# Satellite Simulation on RaspberryPI v2.0

Generated by Doxygen 1.8.6

Fri Mar 7 2014 13:00:15

ii CONTENTS

# Contents

1	Hiera	archical Index	1
	1.1	Class Hierarchy	1
2	Class	s Index	2
:	2.1	Class List	2
3	File I	Index	3
;	3.1	File List	4
4	Class	s Documentation	4
	4.1	action Struct Reference	4
		4.1.1 Detailed Description	5
		4.1.2 Member Data Documentation	5
4	4.2	Analyseur Class Reference	5
		4.2.1 Detailed Description	6
		4.2.2 Member Function Documentation	6
	4.3	Arg_Manager Struct Reference	6
		4.3.1 Detailed Description	7
	4.4	Arg_Non_Fonctionnel Struct Reference	7
		4.4.1 Detailed Description	8
	4.5	Arg_Recovery Struct Reference	8
		4.5.1 Detailed Description	9
	4.6	arguments_f_thread_watchdogM Struct Reference	9
	4.7	CArgument Class Reference	10
	4.8	CarteComm Class Reference	10
		4.8.1 Detailed Description	10
		4.8.2 Member Function Documentation	11
		4.8.3 Member Data Documentation	11
	4.9	CarteRecep Class Reference	11
		4.9.1 Member Function Documentation	12
		4.9.2 Member Data Documentation	12
	4 10	CConfig Class Reference	13
			13
		4.11.1 Detailed Description	14
		4.11.2 Member Function Documentation	14
		4.11.3 Member Data Documentation	20
	4.12		20
			20
		COMMUNICATION_VECTOR Struct Reference	21
•	T. 14	CONTINUE TO TO IT OF OUR OF THE CONTINUE TO TH	۷ ا

4.15	CPartition Class Reference	21
4.16	Cport_service Class Reference	21
4.17	CQueuing Class Reference	22
4.18	CSampling Class Reference	22
4.19	geolocation Struct Reference	22
	4.19.1 Detailed Description	22
4.20	gui_for_emulator.gui_for_ARINC_653 Class Reference	23
	4.20.1 Member Data Documentation	24
4.21	gui.gui_for_ARINC_653 Class Reference	24
4.22	Horloge Class Reference	25
	4.22.1 Detailed Description	25
	4.22.2 Member Function Documentation	25
	4.22.3 Member Data Documentation	26
4.23	Manager Class Reference	26
	4.23.1 Detailed Description	27
	4.23.2 Member Function Documentation	27
4.24	Memoire_stable Class Reference	34
	4.24.1 Detailed Description	34
	4.24.2 Member Function Documentation	34
4.25	order Struct Reference	34
	4.25.1 Detailed Description	35
4.26	orientation Struct Reference	35
	4.26.1 Detailed Description	35
4.27	Plan Class Reference	35
	4.27.1 Detailed Description	36
	4.27.2 Member Function Documentation	36
	4.27.3 Member Data Documentation	40
4.28	RebootFlag Class Reference	40
	4.28.1 Member Function Documentation	41
4.29	SAMPLING_PORT_STATUS_TYPE Struct Reference	42
4.30	Table Class Reference	42
	4.30.1 Detailed Description	43
	4.30.2 Member Function Documentation	43
4.31	Type_Message Struct Reference	45
File	Documentation	45
5.1	/home/william/ARINC653v2/sources/Chaire_SE_Student/Analyseur.h File Reference	<b>45</b>
5.1	5.1.1 Detailed Description	46
5.2	/home/william/ARINC653v2/sources/Chaire_SE_Student/CarteComm.h File Reference	47
J.L	5.2.1 Detailed Description	48
	o.e. Dotained Description	+0

5

1 Hierarchical Index 1

	5.3	/home/william/ARINC653v2/sources/Chaire_SE_Student/CarteRecep.h File Reference	48
		5.3.1 Detailed Description	49
	5.4	/home/william/ARINC653v2/sources/Chaire_SE_Student/Checkpoint.h File Reference	50
		5.4.1 Detailed Description	51
	5.5	/home/william/ARINC653v2/sources/Chaire_SE_Student/Horloge.h File Reference	51
		5.5.1 Detailed Description	52
	5.6	/home/william/ARINC653v2/sources/Chaire_SE_Student/Manager.h File Reference	53
		5.6.1 Detailed Description	53
	5.7	/home/william/ARINC653v2/sources/Chaire_SE_Student/Memoire_stable.h File Reference	54
		5.7.1 Detailed Description	54
	5.8	/home/william/ARINC653v2/sources/Chaire_SE_Student/RebootFlag.h File Reference	55
		5.8.1 Detailed Description	55
	5.9	/home/william/ARINC653v2/sources/Chaire_SE_Student/Table.h File Reference	56
		5.9.1 Detailed Description	57
Ind	ex		58
1	Hie	rarchical Index	
1.1	Cla	ass Hierarchy	
Thi	s inhe	eritance list is sorted roughly, but not completely, alphabetically:	
	actio	n	4
	Anal	yseur	5
	Arg_	Manager	6
	Arg_	Non_Fonctionnel	7
	Arg_	Recovery	8
	argui	ments_f_thread_watchdogM	9
	CArg	ument	10
	Carte	eComm	10
	Carte	еRесер	11
	CCor	nfig	13
	Chec	kpoint	13
	CMaı	nager	20
	com	munication_obj	20
	СОМ	MUNICATION_VECTOR	21
	CPar	tition	21

Cport_service	21
CQueuing	22
CSampling	22
geolocation	22
Horloge	25
Manager	26
Memoire_stable	34
order	34
orientation	35
Plan	35
RebootFlag	40
SAMPLING_PORT_STATUS_TYPE	42
<b>Table</b> Tk	42
gui.gui_for_ARINC_653	24
gui_for_emulator.gui_for_ARINC_653	23
Type_Message	45
2 Class Index	
2.1 Class List	
Here are the classes, structs, unions and interfaces with brief descriptions:	
action Contents an instruction	4
Analyseur Analyses and compress the images from the camera	5
Arg_Manager Content all the objects necessary to the main manager thread	6
Arg_Non_Fonctionnel Content all the objects necessary to the non functionnal thread (including Watchdog)	7
Arg_Recovery Content all the objects necessary to the recovery thread	8
arguments_f_thread_watchdogM	9
CArgument	10
CarteComm Emulates a TM controller	10

3 File Index

CarteRecep	11
CConfig	13
Checkpoint Creation and management of the checkpoints	13
CManager	20
communication_obj	20
COMMUNICATION_VECTOR	21
CPartition	21
Cport_service	21
CQueuing	22
CSampling	22
geolocation Structure containing the geolocation of a picture	22
gui_for_emulator.gui_for_ARINC_653	23
gui.gui_for_ARINC_653	24
Horloge Emulates the clock used by a partition	25
Manager Used in Master and Slave as the mission manager object. It regroups all the functionnalities and the objects necessary to manage the mission	26
Memoire_stable Image memory	34
order Structure of the order from managers partition	34
orientation Structure containing the attitude of the satellite	35
Plan Imports and stocks the instructions	35
RebootFlag	40
SAMPLING_PORT_STATUS_TYPE	42
Table Buffer of checkpoints	42
Type_Message	45

# 3.1 File List

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

/home/william/ARINC653v2/sources/Chaire_SE_Student/Analyseur.h Analyse and compress the images from the camera	45
/home/william/ARINC653v2/sources/Chaire_SE_Student/CarteComm.h Image transmission device	47
/home/william/ARINC653v2/sources/Chaire_SE_Student/CarteRecep.h Plan reception device	48
/home/william/ARINC653v2/sources/Chaire_SE_Student/Checkpoint.h  Backup structure to save the states of each images	50
/home/william/ARINC653v2/sources/Chaire_SE_Student/Horloge.h Emulate the clock used by a partition	51
/home/william/ARINC653v2/sources/Chaire_SE_Student/Manager.h Backup structure to save the states of each images	53
/home/william/ARINC653v2/sources/Chaire_SE_Student/Memoire_stable.h Emulate the memory used for image storage	54
/home/william/ARINC653v2/sources/Chaire_SE_Student/Plan.h	??
/home/william/ARINC653v2/sources/Chaire_SE_Student/RebootFlag.h	55
/home/william/ARINC653v2/sources/Chaire_SE_Student/Table.h Storage object for instruction	56
/home/william/ARINC653v2/sources/include/CArgument.h	??
/home/william/ARINC653v2/sources/include/CBasefunction.h	??
/home/william/ARINC653v2/sources/include/CCommunication.h	??
/home/william/ARINC653v2/sources/include/CConfig.h	??
/home/william/ARINC653v2/sources/include/CManager.h	??
/home/william/ARINC653v2/sources/include/communication_obj.h	??
/home/william/ARINC653v2/sources/include/CPartition.h	??
/home/william/ARINC653v2/sources/include/Cport_service.h	??
/home/william/ARINC653v2/sources/include/CQueuing.h	??
/home/william/ARINC653v2/sources/include/CSampling.h	??

# 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.

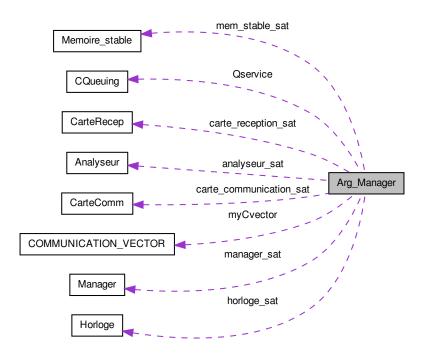
The documentation for this class was generated from the following files:

- /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.

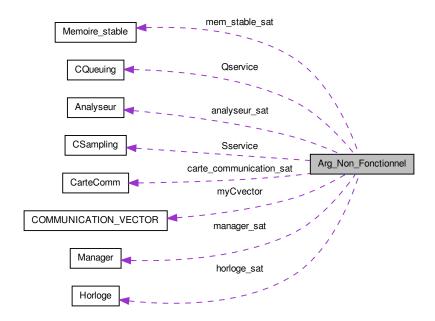
The documentation for this struct was generated from the following files:

- /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).

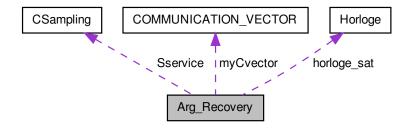
The documentation for this struct was generated from the following files:

- /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

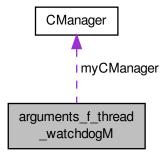
Content all the objects necessary to the recovery thread.

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.

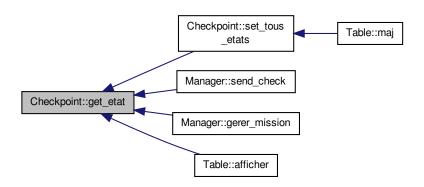
#### **Parameters**

int	State index (from 0 to 4).
IIIL	State index (iron o to 4).

#### Returns

Image's state.

Here is the caller graph for this function:



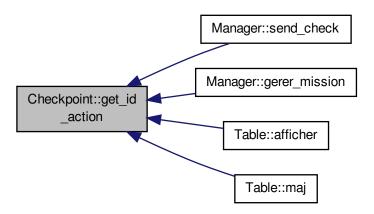
4.11.2.2 int Checkpoint::get\_id\_action ( )

Accessor id\_action.

Returns

id\_action.

Here is the caller graph for this function:



4.11.2.3 int Checkpoint::get\_id\_image\_sauvegarde ( )

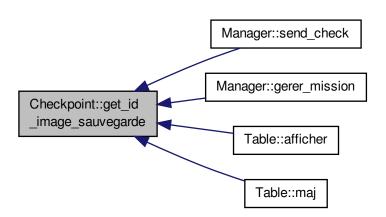
Accessor id\_image\_sauvegarde.

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:



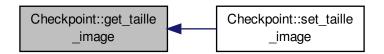
4.11.2.5 int Checkpoint::get\_taille\_image ( )

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</p>

Here is the caller graph for this function:



4.11.2.7 void Checkpoint::set\_etat ( int numero\_etat, bool v\_etat )

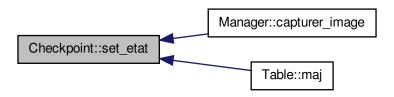
Single state mutator.

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 )

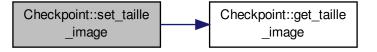
Image weight mutator.

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 )

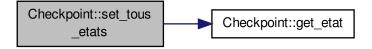
All-states mutator.

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

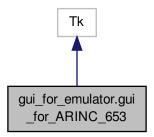
Structure containing the geolocation of a picture.

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

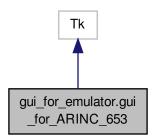
- /home/william/ARINC653v2/sources/Chaire\_SE\_Student/Leica/Leica.cpp
- /home/william/ARINC653v2/sources/Chaire\_SE\_Student/Scao/Scao.cpp

# 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')

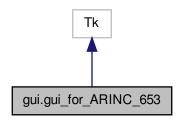
Creation d'un boutton dans 'menu\_bottom' b = Button(menu\_bottom,text='execute',command=self.affiche\_commande) b.grid(row=1,column=3,sticky='e') Creation d'un boite d'edition dans 'nenu\_top' entre = Entry(menu\_bottom,width=20,relief='sunken') entre.insert(END,'Texte a afficher') entre.grid(row=1,column=2,sticky='ew')

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

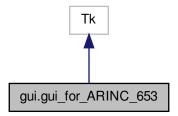
/home/william/ARINC653v2/sources/GUI/gui\_for\_emulator.py

# 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
------------

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.

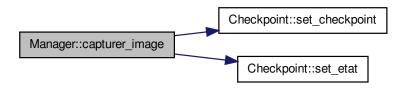
char*	name of the image
int	Weight of the image (px)

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

string	: Partition name
string	: Partition mode
Horloge*	Clock
Analyseur*	Image analyser
Memoire	Image memory
stable*	
CarteComm*	Image transmission device
CarteRecep*	Plan transmission device

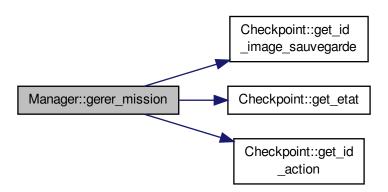
COMMUNICATI- Communication information vector ON\_VECTOR

# Returns

0 in case of error.

Lecture de la prochaine action du plan

Here is the call graph for this function:

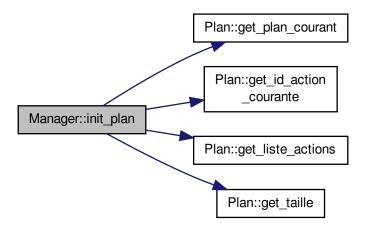


4.23.2.6 void Manager::init\_plan ( Plan pl )

Replace the current plan by a new one.

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

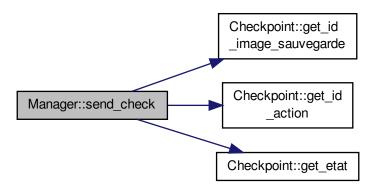
4.23.2.11 void Manager::send\_check ( Checkpoint check )

Send a checkpoint to the backup partition.

#### **Parameters**

Checkpoint	Checkpoint to be send.

Here is the call graph for this function:



4.23.2.12 void Manager::set\_mode ( string smode )

Mutator on the mode.

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:

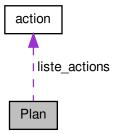
/home/william/ARINC653v2/sources/Chaire\_SE\_Student/Scao/Scao.cpp

### 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.

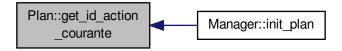
## 4.27.2.3 int Plan::get\_id\_action\_courante ( )

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
	ınt	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.

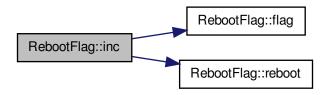
Returns
flag_value

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.

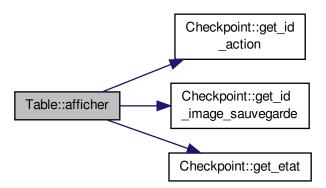
#### 4.30.2 Member Function Documentation

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** 

	income id
l int	I Image id
,,,,,	inage 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.

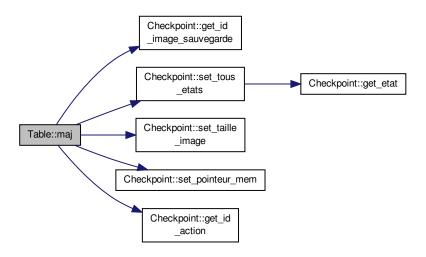
**Parameters** 

checkpoint	checkpoint to be added
------------	------------------------

#### Returns

1 if operation was successful, 0 if not

Here is the call graph for this function:



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

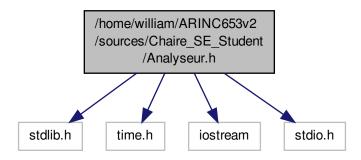
## 5 File Documentation

## 5.1 /home/william/ARINC653v2/sources/Chaire\_SE\_Student/Analyseur.h File Reference

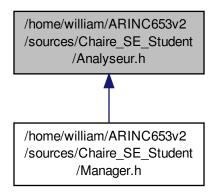
Analyse and compress the images from the camera.

```
#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

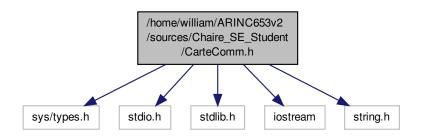
**Author** 

Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

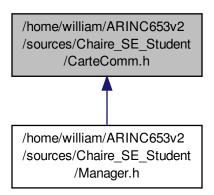
## 5.2 /home/william/ARINC653v2/sources/Chaire\_SE\_Student/CarteComm.h File Reference

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** 

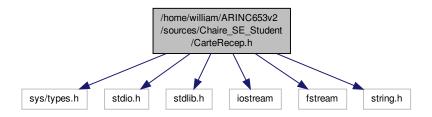
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.3 /home/william/ARINC653v2/sources/Chaire\_SE\_Student/CarteRecep.h File Reference

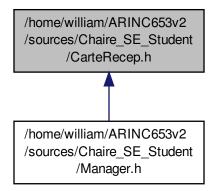
## Plan reception device.

```
#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

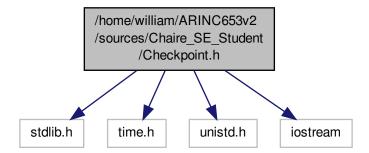
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.4 /home/william/ARINC653v2/sources/Chaire\_SE\_Student/Checkpoint.h File Reference

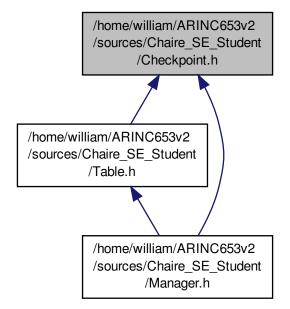
Backup structure to save the states of each images.

#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:



#### 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** 

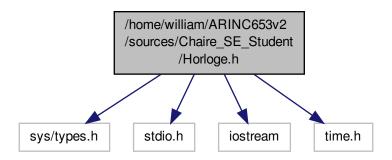
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.5 /home/william/ARINC653v2/sources/Chaire\_SE\_Student/Horloge.h File Reference

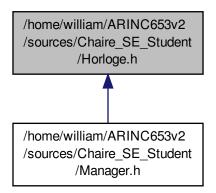
Emulate the clock used by a partition.

```
#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 <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**

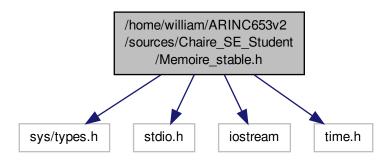
Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.7 /home/william/ARINC653v2/sources/Chaire\_SE\_Student/Memoire\_stable.h File Reference

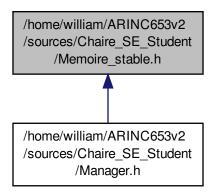
Emulate the memory used for image storage.

```
#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

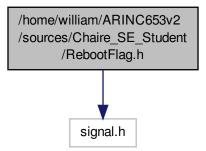
2.0

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**

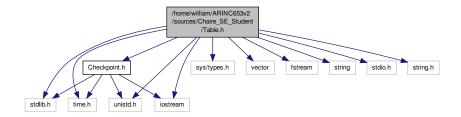
### Lucie BEAUSSART Thomas BETOUS Abdelkader BOUARFA William EXCOFFON

## 5.9 /home/william/ARINC653v2/sources/Chaire\_SE\_Student/Table.h File Reference

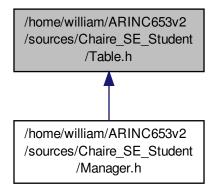
### Storage object for instruction.

```
#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

/home/william/ARINC653v2/sources/Chaire_SE Student/Analyseur.h, 45	CPartition, 21 CQueuing, 22
/home/william/ARINC653v2/sources/Chaire_SE	CSampling, 22
Student/CarteComm.h, 47	capturer_image
/home/william/ARINC653v2/sources/Chaire_SE	Manager, 27
Student/CarteRecep.h, 48	CarteComm, 10
/home/william/ARINC653v2/sources/Chaire_SE	active, 11
Student/Checkpoint.h, 50	activer_carte, 11
/home/william/ARINC653v2/sources/Chaire_SE	desactiver_carte, 11
Student/Horloge.h, 51	envoyer, 11
/home/william/ARINC653v2/sources/Chaire_SE	get_etat, 11
Student/Manager.h, 53	CarteRecep, 11
/home/william/ARINC653v2/sources/Chaire_SE	activer_carte, 12
Student/Memoire_stable.h, 54	desactiver_carte, 12
/home/william/ARINC653v2/sources/Chaire_SE	etat, 12
Student/RebootFlag.h, 55	get_etat, 12
/home/william/ARINC653v2/sources/Chaire_SE	recevoir_plan, 12
Student/Table.h, 56	Checkpoint, 13
	etat, 20
action, 4	get_etat, 14
angle_prise_vue, 5	get_id_action, 14
date_action, 5	get_id_image_sauvegarde, 15
duree_action, 5	get_pointeur_mem, 15
id_images_a_envoyer, 5	get_taille_image, 16
nature, 5	id_action, 20
nbre_images_a_envoyer, 5	id_image_sauvegarde, 20
numero_action, 5	pointeur_mem, 20
numero_image, 5	set_checkpoint, 16
active	set_etat, 17
CarteComm, 11	set_pointeur_mem, 17, 18
activer_carte	set_taille_image, 18
CarteComm, 11	set_tous_etats, 19
CarteRecep, 12	taille_image, 20
actualiser_table	communication_obj, 20
Manager, 27	compresser_image
afficher T-11- 40	Analyseur, 6
Table, 43	Cport_service, 21
afficher_action	Creer_Plan
Plan, 36	Manager, 28
Analyseur, 5	
compresser_image, 6	date2seconds
traiter_image, 6	Horloge, 25
angle_prise_vue	date action
action, 5	action, 5
Arg_Manager, 6	desactiver_carte
Arg_Non_Fonctionnel, 7	CarteComm, 11
Arg_Recovery, 8	CarteRecep, 12
arguments_f_thread_watchdogM, 9	duree action
big_bang	action, 5
Horloge, 26	,
11011090, 20	envoyer
CArgument, 10	CarteComm, 11
CConfig, 13	etat
CManager, 20	CarteRecep, 12
COMMUNICATION_VECTOR, 21	Checkpoint, 20

INDEX 59

flag	RebootFlag, 41
RebootFlag, 41	init_plan
	Manager, 29
geolocation, 22	
gerer_mission	labelVariable
Manager, 28	gui_for_emulator::gui_for_ARINC_653, 24
get_action	liste_actions
Plan, 37	Plan, 40
get_check	
Table, 43	maj
get_dernier_check	Table, 44
Table, 43	Manager, 26
get_etat	actualiser_table, 27
CarteComm, 11	capturer_image, 27
CarteRecep, 12	Creer_Plan, 28
Checkpoint, 14	gerer_mission, 28
get_flag_value	init_plan, 29
RebootFlag, 41	order_transfer, 30
get_id_action	orienter_satellite, 30
Checkpoint, 14	receive_transfer_report, 30
get_id_action_courante	recuperer_dernier_check, 32
Plan, 37	send_check, 32
get_id_image_sauvegarde	set_mode, 32
Checkpoint, 15	set_partition, 33
get_liste_actions	table_get_check, 33
Plan, 37	table_id_image_to_position, 33
get_num_check	table_position_to_id_image, 33
Table, 43	trouver_action, 33
get_plan_courant	Memoire_stable, 34
Plan, 37	stocker, 34
get_pointeur_mem	vider, 34
Checkpoint, 15	
get_taille	nature
Plan, 38	action, 5
get_taille_image	nbre_images_a_envoyer
Checkpoint, 16	action, 5
get_temps	next_action
Horloge, 26	Plan, 38
gui.gui_for_ARINC_653, 24	numero_action
gui_for_emulator.gui_for_ARINC_653, 23	action, 5
gui_for_emulator::gui_for_ARINC_653	numero_image
labelVariable, 24	action, 5
	andan 04
Horloge, 25	order, 34
big_bang, 26	order_transfer
date2seconds, 25	Manager, 30
get_temps, 26	orientation, 35
id action	orienter_satellite
id_action	Manager, 30
Checkpoint, 20	Dian OF
id_action_courante	Plan, 35
Plan, 40	afficher_action, 36
id_image_sauvegarde	get_action, 37
Checkpoint, 20	get_id_action_courante, 37
id_image_to_position	get_liste_actions, 37
Table, 44	get_plan_courant, 37
id_images_a_envoyer	get_taille, 38
action, 5	id_action_courante, 40
INC	liste_actions, 40

60 INDEX

next_action, 38	Memoire_stable, 34
plan_courant, 40	T.11 40
set_action, 38	Table, 42
set_id_action_courante, 39	afficher, 43
set_liste_actions, 39	get_check, 43
set_plan, <mark>39</mark>	get_dernier_check, 43
set_plan_courant, 39	get_num_check, 43
set_plan_recouvrement, 40	id_image_to_position, 4-
set_taille, 40	maj, 44
taille_plan, 40	position_to_id_image, 4
plan_courant	table_get_check
Plan, 40	Manager, 33
pointeur_mem	table_id_image_to_position
Checkpoint, 20	Manager, 33
position_to_id_image	table_position_to_id_image
Table, 45	Manager, 33
,	taille_image
RebootFlag, 40	Checkpoint, 20
flag, 41	taille_plan
get_flag_value, 41	Plan, 40
inc, 41	traiter_image
receive_transfer_report	Analyseur, 6
Manager, 30	trouver_action
recevoir_plan	Manager, 33
CarteRecep, 12	Type_Message, 45
•	Type_Message, 45
recuperer_dernier_check	vider
Manager, 32	Memoire_stable, 34
send_check	momo <u>-</u> otable, or
Manager, 32	
set_action	
Plan, 38	
set_checkpoint	
Checkpoint, 16	
set_etat	
Checkpoint, 17	
set_id_action_courante	
Plan, 39	
set_liste_actions	
Plan, 39	
set_mode	
Manager, 32	
set_partition	
Manager, 33	
set_plan	
Plan, 39	
set_plan_courant	
Plan, 39	
set_plan_recouvrement	
Plan, 40	
set_pointeur_mem	
Checkpoint, 17, 18	
set_taille	
Plan, 40	
set_taille_image	
<del>-</del>	
Checkpoint, 18	
set_tous_etats	
Checkpoint, 19	
stocker	