9th January 2015

Swiss Space Center

CHEOPS : CCSDS PARSING MODULE

project in space technologies

Prepared by:

Julien Graisse

Checked/Approved/Signed by:

Anton Ivanov

Swiss Space Center
EPFL - Lausanne

09.01.2015

RECORD OF REVISIONS

ISS/REV	Date	Modifications	Created/modified by

RECORD OF REVISIONS

GLOSSARY

INTRODUCTION

THE CHEOPS PROJECT
OBJECTIVES OF THIS PROJECT

DESIGN ASSUMPTION

REQUIREMENTS
DEVELOPMENT TOOLS
The programming language
Development environment

SOFTWARE IMPLEMENTATION

THE PACKET PARSER
THE ERROR HANDLER
THE PACKET HANDLER

DISCUSSION

CONCLUSION

APPENDIX: DOCUMENTATION

INSTITUTIONS

GLOSSARY

CCSDS: The Consultative Committee for Space Data Systems

ESA: European Space Agency

MARSIS: low frequency, pulse-limited radar sounder and altimeter used on the ESA Mars Express mission.

RAM: Random Access Memory. Memory used by running processes on all common computers

SSO: Swiss Space Office

telemetry: Automated communications process by which measurements are made and data collected at remote points and transmitted to receiving equipment for monitoring.

INTRODUCTION

THE CHEOPS PROJECT

CHEOPS (Characterising ExOPlanets Satellite) is a mission organized by the SSO and the ESA. It consists of the development of a space telescope dedicated to examining and characterising transiting exoplanets. The telescope, mounted on a s-class satellite platform will be the first mission dedicated to search for transits by means of ultrahigh precision photometry on bright stars already known to host planets.

The project was selected in October 2012 and is planned to be launched in 2017 or 2018.

MISSION SUMMARY

Primary Goal	Characterize transiting exoplanets on known bright and nearby host stars
Targets	Known exoplanet host stars with a V-magnitude < 12.5 (goal: 13) anywhere on the sky
Wavelength	Visible range : 400 to 1100 nm
Telescope	33 cm reflective an-axis telescope
Orbit	Sun-synchronous Low Earth Orbit, LTAN 6am, altitude 620-800 km
Lifetime	3.5 years
Type S-class mission	S-class mission

Source: http://cheops.unibe.ch/index.php/article-2

OBJECTIVES OF THIS PROJECT

The idea of this project is to start the development of a CCSDS-compliant messages parser. The Parser program consists of a group of module handling binary telemetry data sent from the satellite. At this early stage of the mission, no simulation data from CHEOPS was available for testing. This project is built working on MARSIS simulation data.

The program executes the following steps:

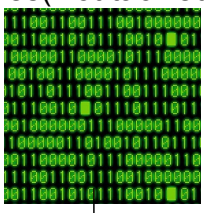
1. Read an incoming bit stream of data. At this stage of the development, the data is MARSIS simulation data read from a local file.

2. The bit stream read from file is parsed into CCSDS-compliant readable packets.

3. The packets are sent to a packet handler which executes any desirable operation defined by the mission. At this point all it does is:

-Checking the data isn't corrupted and the packet valid.

- Store mission's payload in FITS files(Not tackled here, last semester's project)



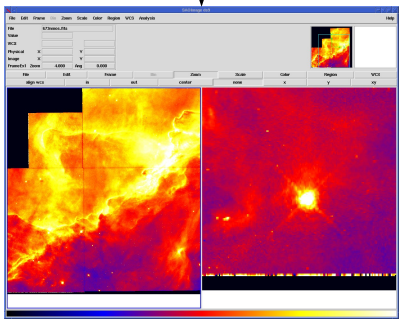
Packet Parser module

SOURCE PACKET HEADER (48 bits)								PACKET DATA FIELD (Variable)	
PACKET ID					PACKET SEQUENCE CONTROL		PACKET LENGTH	DATA FIELD HEADER	SOURCE DATA
Version Number	Type	Data Field Header Flag	Application Process ID		Segment. Flags	Source Sequence Count (single counter for each single Application Process ID)	(octets in Packet Data Field −1)		
			Proc. ID	Pack. Cat.					
000_b	0	1	$(77 \circ 80)_h$	12_d	11_b	$(0 \circ 2^{14}-1)_b$	Max $(4106-1)_b$		
3 b	1 b	1 b	7 b	4 b	2 b	14 b	16 b	80 b	
2 B					2 B		2 B	10 B	Variable (Max 4096 B)

Error Handler module

Packet Handler module

Outputs Scientific Data



DESIGN ASSUMPTION

REQUIREMENTS

Name: Error Detector

Type: Behavioural and maintenance requirements

Status: draft

Verification method: Test

Parent: None

Description: Any kind of error or exception in the composition of a packet must be detected.

Name: Error Recognition

Type: Behavioural and maintenance requirements

Status: draft

Verification method: Test

Parent: Error Detector

Description: Any detected error must be characterized and described in order to be understood.

Name: Error Storage

Type: Behavioural and maintenance requirements

Status: Issued

Verification method: Test

Parent: Error Detector

Description: Any detected error must be stored in a log file.

Name: CCSDS formatting

Type: Functional requirement

Status: issued

Verification method: Inspection

Parent: None

Description: Any generated packet must respect the CCSDS standard.

Name: Independence of modules.

Type: Architectural requirement

Status: Issued

Verification method: Inspection

Parent: None

Description: The different components of the program must be interconnected independent modules that could be reused in other programs.

DEVELOPMENT TOOLS

The programming language

It has been decided to work with C++ for the CHEOPS mission. Unlike last semester's project, no other tool than the C++ standard library is required.

Development environment

OS

The project is developed on linux, the platform on which it is supposed to run. No special requirement is needed for the machine. The application has been tested so far on a common laptop with the following characteristics:

Ubuntu Linux 12.04

intel(r) core(tm) i3-2310m cpu @ 2.10ghz dual-core

4Gb RAM

Code redistribution

This project is under the GNU GPLv3 licence. The source code is accessible on github at the following address:

https://github.com/merodrem/CHEOPS

SOFTWARE IMPLEMENTATION

In this section, the different modules are detailed. The list of classes they contain isn't exhaustive. The point is to give an idea of important features.

THE PACKET PARSER

Class

```
PacketParser(const char* filepath)
    methods
        TM_Packet nextPacket(char* &scientific_data)
        bool hasnext()
```

The packet parser's role is to convert bitstreams read from a file into CCSDS packets. The container selected to represent packets in C++ is simply a **structure**. Thoses structures are defined in the file Packet.h of the module.

The only interesting feature for this module is the nextPacket method returning the last read and parsed packet. It is quite similar to JAVA standard library's file readers in the concept. This method needs a reference to a a pointer where the scientific data will be dynamically allocated (the field has a variable length). Warning: it is the programmer's duty to free the pointer provided when not needed anymore to avoid memory leaks.

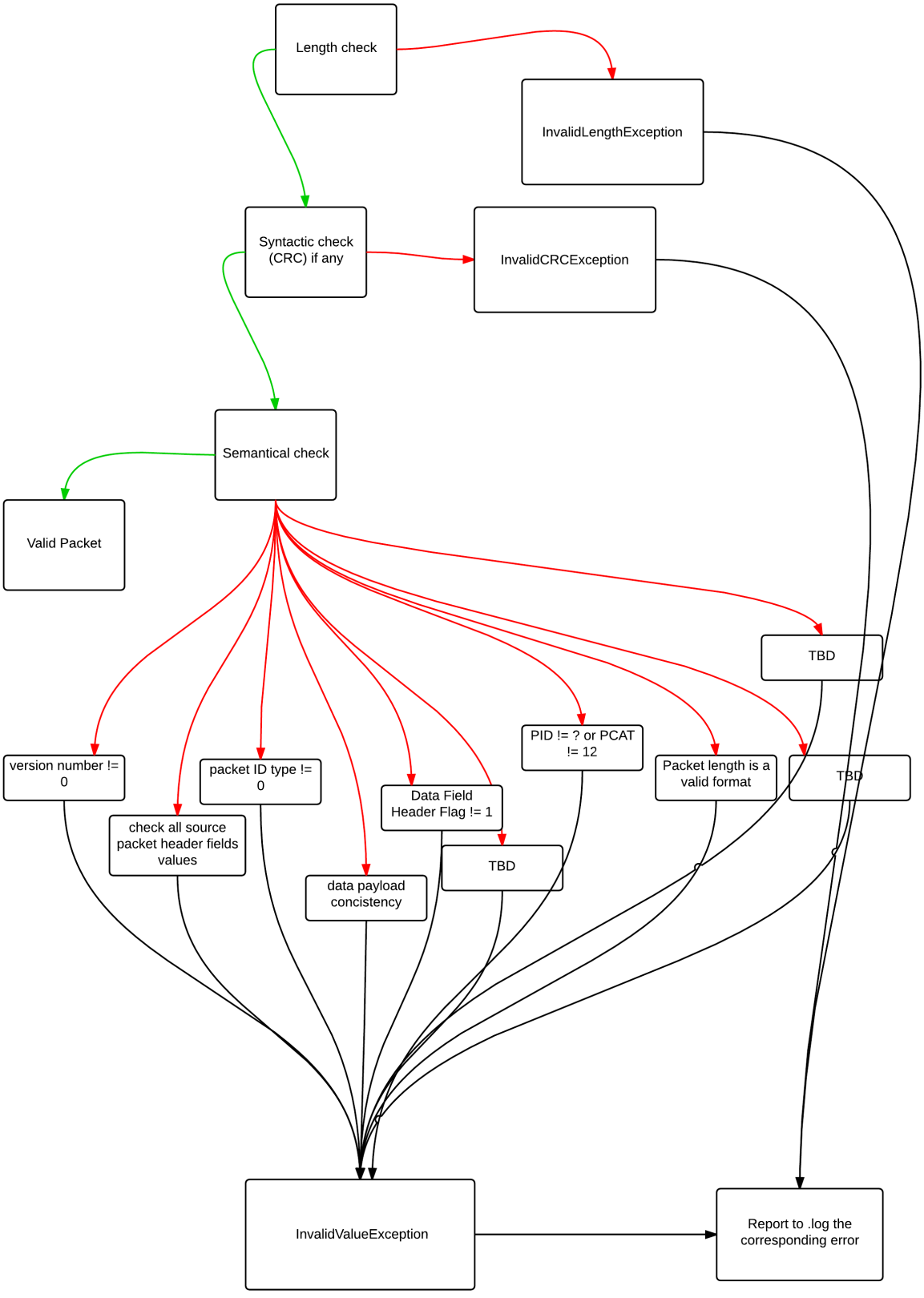
Note about bitfields in C++:

One should know that the fields themselves are packed in different orders and locations depending on the compiler. So, using standard functions such as memcpy() won't be portable.

THE ERROR HANDLER

The following page represents the error detection scheme. Different checks are operated to detect any syntactic, semantic or consistency error. For now, errors are logged in a file, but later the goal is to keep them in a database.

No need to explain the architecture and organization of files in the module. It is quite straightforward with the schema following:


Length check
InvalidLengthException
Syntactic check (CRC) if any
InvalidCRCException
Semantical check
Valid Packet
TBD
version number != 0
packet ID type != 0
check all source packet header fields values
data payload concistency
Data Field Header Flag != 1
TBD
PID != ? or PCAT != 12
Packet length is a valid format
TBD
InvalidValueException
Report to .log the corresponding error

THE PACKET HANDLER

The Packet Handler is the playground of this project. It receives structures representing telemetry packets from the Packet Parser module and is allowed to do with it anything the mission's specifications require to do with collected Data. At this stage it is still quite empty as no real task has been submitted already. This project focused on parsing Data. Managing it like, with the FITS format, was last semester's topic.

Class

```
PacketHandler(const char* file);
```

methods

```
PacketHandler(const char* file);
void readWholeFile();
```

The readWholeFile() was no part of the tasks, only here for testing purpose. Of course this class also has attributes as a PacketParser or an ErrorHandler, but those are transparent to end users of the class, respecting the encapsulation protocol. Checking for errors being a higher-level operation than parsing, it has been decided to do it here. It also makes logging errors easier (the parser module doesn't have to notify the packet handler of errors).

DISCUSSION

More than really developing tools to integrate into CHEOPS whole software, this project was rather exercising on topics that will have to be mastered when developing the actual software. But for now, due to a lack of tasks on CHEOPS, the project works on MARSIS simulation data which is slightly different.

The code of this project is functional the given tasks have been implemented and the structure of the software is laid. But there are still many points to improve:

- The parsing of binary streams into structures is too hard-coded. It makes the code hard to be understood for anyone willing to work on it.
- Cruel lack of documentation beside this report. Doxygen is being considered in order to palliate to this problem.
- Only the structure of semantic tests with a few examples has been implemented. But there are many many different semantic constraints on CCSDS packets and they have to be added. It is an easy but long work to try to spot them all.

CONCLUSION

This project was a first step in the development of a CCSDS parsing module for CHEOPS. The idea was to build tools working on bitstreams for next tasks. Those tools consist in a module parsing bitstrings in structures representing packets, an error detecting module for those generated packets, and a packet handling module managing everything.

The tasks have been developed in C++ on ubuntu linux. Due to a delay in the general progress of the CHEOPS mission, no simulation data was available to work on and an old mission had to be used to test the code. As a general comment about the progress, tasks are functional but must be refined and continued in the next steps.

This semester project was a bit to small to cover the whole system and over all a bit too early in the mission's development stage. lots of features, rules, organization,..., were still to be determined. The development was often pending waiting for available resources to test or to get organized. Nevertheless, this work represent a good start for further developments. It was also a good grip and learning of bitfield manipulation which wasn't as easy as predicted.

APPENDIX: DOCUMENTATION

Institutions

The CHEOPS Project

http://cheops.unibe.ch/

The Swiss Space Center

http://space.epfl.ch/

The ESA

www.esa.int

The SSO

http://www.sbfi.admin.ch/themen/01371/index.html?lang=en

The CCSDS standard official website

http://public.ccsds.org/