

PacketLib: A C++ Library for Scientific Satellite Telemetry Applications

A. Bulgarelli, F. Gianotti, M. Trifoglio

CNR/IASF sezione di Bologna - Via P. Gobetti 101, 40129 Bologna, Italy

Abstract. PacketLib is a C++ open-source software library for writing applications which deal with satellite telemetry source packets, provided that the packets are compliant with the CCSDS Telemetry and Telecommand Standards. The library is being used within the Italian Space Agency (ASI) mission AGILE for simulation, graphical display, processing and decoding of the telemetry generated by the Test Equipment (TE) of two AGILE detectors. From an input stream of bytes, the library is able to recognize automatically the source packets (described by a simple configuration file), and provides a simple access to each packet field by means of an object oriented interface. In the same way the library writes source packets to output stream. Various types of input and output streams are abstracted by a software layer.

1. Introduction

The equipment to support the test and calibration of scientific payload requires tailor-made software applications which are similar in overall design and functionality but are different in details. These applications must work with CCSDS satellite telemetry. The telemetry (TM) and telecommand (TC) packets are usually divided into two sections: (*i*) the Header section, which is the only mandatory part of a packet and contains the packet identifier, the packet length, and the Source Sequence Counter, and (*ii*) the Data Field section containing either the TM scientific and house-keeping data or the TC data. Although the CCSDS standard does not specify the Data Field internal structure, usually it contains an Header which precedes the data. In turn, the data are grouped according to a logical sequence of bits which represent well defined information.

Based on these considerations, we have designed PacketLib, a C++ software library running on Unix platform which can be used as the basis for building applications able to handle the source packets, down to the level of the single logical structure contained in the data field.

The PacketLib¹ is aimed at providing a reusable software library for satellite telemetry production and processing and a rapid development for Test Equipment (TE), Electric Ground Support Equipment (EGSE) and Ground Segment

¹<http://www.bo.iasf.cnr.it/~GSE>

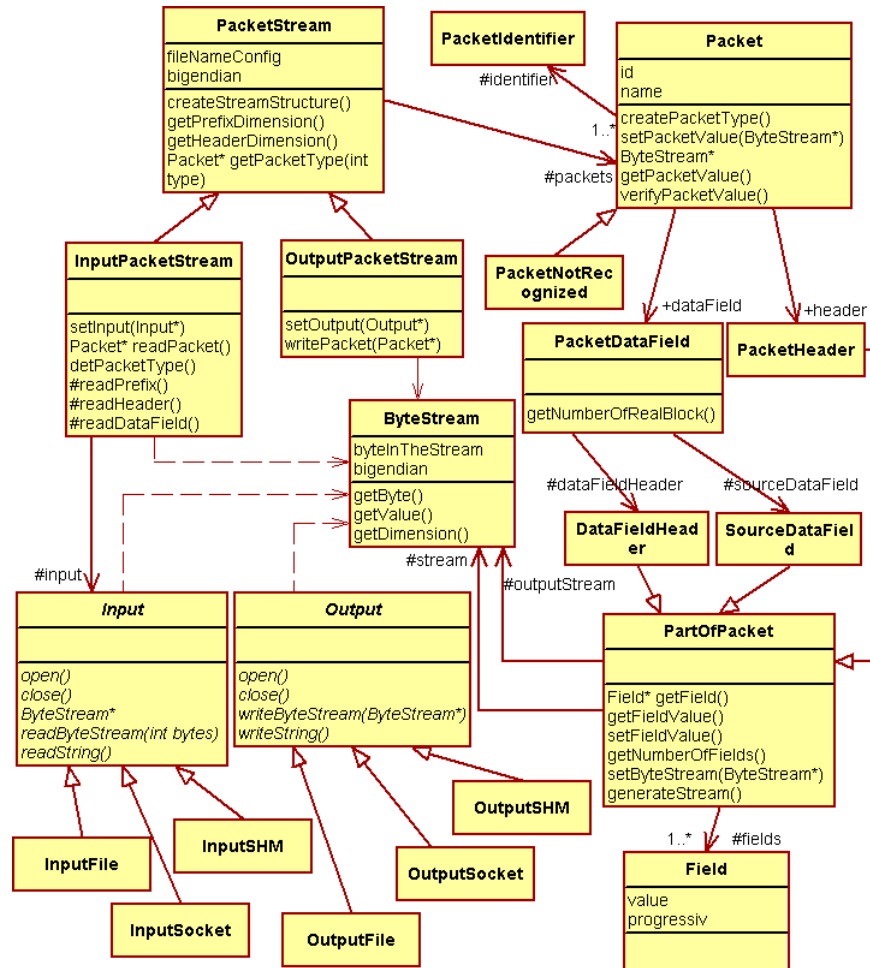


Figure 1. The class diagram of PacketLib.

applications. It has been developed in the context of the Italian Space Agency (ASI) AGILE space mission (Tavani et al. 2001).

2. The Architecture

PacketLib is structured into two main layers: the Telemetry Management layer, which interfaces the application, and the I/O Abstraction layer, which interfaces the Operating System. The former allows the application to address the various elements forming the packet data stream, without having to deal with the specific structure. The latter abstracts from the specific input and output mechanisms (e.g., sockets, files, and shared memories).

This approach allows a more rapid development of the user applications without having to deal with the kind of input and output sources, which could be changed at run time. Indeed the library could be used either to process or to generate a telemetry stream.

PacketLib abstracts the byte stream and all the details characterizing a given packet structure by means of the Object Model represented in the simplified UML class diagram shown in Figure 1.

In particular, for each kind of packet stream, the library requires one ASCII file containing the description of the overall stream, and a set of ASCII files containing all the parameters defining the various packets, one for each kind of packet identifier. At run time, PacketLib loads these files and builds the representation of the stream and the packets structures into memory.

3. An Example

The real functioning of the library can be easily understood with the following little coding example:

```
1.  InputPacketStream ips;
2.  char* parameters; //set the parameters of input
3.  Input *in = new InputSocket(parameters);
4.  in.open();
5.  ips.createStreamStructure("conf.stream");
6.  ips.setInput(in);
7.  Packet* p = ips.getPacket();
8.  cout << "The value of field 4 of header is " <<
    p->header->getFieldValue(4) << endl;
9.  cout << "The value of field 2 of source data field is " <<
    p->dataField->sourceDataField->getFieldValue(2) << endl;
```

Line 1 instantiates the object that represents the input byte stream, and line 5 loads the configuration files and creates into memory the byte stream structure and the packets structure (header, data field and fields). Lines 2, 3, and 4 create a telemetry input source and open it. To change the input source without modifying the rest of the code, it is only necessary to modify the line 3 by instantiating another type of input (e.g., InputFile instead of InputSocket), and opening it with the correct parameters. Line 6 links the input byte stream with the input source, and the remaining code extracts the required packet information from the input stream. In particular, line 7 reads a complete packet, whereas lines 8–9 print the value of the various fields.

Summarising, lines 2–4,6 provide the I/O Abstraction layer, while lines 1,5,7–9 are related to the Telemetry Management layer.

This example demonstrates that with a few lines of code it is possible to connect the application with an input byte stream and to obtain an object representing a packet with all the field value decoded. The library is able to manage either big-endian and little-endian format.

For a full comprehension of how the library works, it is necessary to understand how the input is processed. After having read the packet Header (which is of fixed length), the library knows the length of the subsequent Data Field. The actual structure of the Data Field is derived by its identifier which consists of one or more fields containing some predefined values. Typically, the APID field of CCSDS standard is used, but the library has not limitation, and allows identifiers which include others fields (e.g., in the AGILE case the Type/Subtype fields of the Data Field Header).

The identification of the various packets is performed by looking in the packets for one of the possible identifiers defined in the *.packet ASCII files. Identified packets are unpacked and the various fields are read into the packet structure defined in the *.packet files, where can be directly addressed as shown in lines 8–9. In the negative case, an object representing a generic not recognized packet is created, where the Data Field byte sequence is copied without any structure.

4. Some PacketLib-Based Applications

Various PacketLib-based applications have been built in the AGILE framework: (i) a telemetry generator which is able to simulate the output of the Digital Front End (DFE) of Minicalorimeter, one of the instruments of the mission; (ii) the QPacketViewer which recognizes the packets of a telemetry stream and shows all the various fields; and (iii) the DISCoS ProcessorLib which extends the DISCoS system (Gianotti et al. 2001), in order to process the telemetry scientific data and convert them into FITS file.

The generator allowed us to write and test the processing software without the presence of the instrument’s DFE. The QPacketViewer has been used for a preliminary analysis of the telemetry produced by the DFE TE. With these tools it has been possible to realize the analysis software (Quick Look or off-line analysis) in parallel with the building of the detectors, allowing, at the same time, a more accurate test of the overall software of the TE (either in functional and performance terms). This has enabled us to perform a more accurate analysis of all the possible errors into the telemetry data before using the processing software.

5. Conclusions

We plan to extend the library in order to manage a generalized type of packet that will include all the possible AGILE packet type formats. In addition, the library flexibility will be improved by increasing the number of I/O communication channel types.

At the moment the library is used with success in the development of TE software for the AGILE space mission and it will be used for the AGILE Payload EGSE. It enables a rapid development of the TE and EGSE applications based on object oriented architecture.

References

- Tavani, M. et al. 2001, Proceedings of the 5th Compton Symposium, AIP Conf. Proceedings, ed. M. McConnell, Vol. 510, 476
- Gianotti, F. & Trifoglio, M. 2001, in ASP Conf. Ser., Vol. 238, Astronomical Data Analysis Software and Systems X, ed. F. R. Harnden, Jr., F. A. Primini, & H. E. Payne (San Francisco: ASP), 245
- CCSDS, Packet Telemetry - CCSDS 102.0-B-5, November 2000