Astronomical Data Analysis Software and Systems X
ASP Conference Series, Vol. 238, 2001
F. R. Harnden Jr., F. A. Primini, and H. E. Payne, eds.

DISCoS—Detector-Independent Software for On-Ground Testing and Calibration of Scientific Payloads Using the ESA Packet Telemetry and Telecommand Standards

F. Gianotti and M. Trifoglio

Istituto TeSRE/CNR - Via P. Gobetti 101, 40129 Bologna, Italy

Abstract. The ESA Packet Telemetry and Telecommand Standards satisfy the requirements of a great variety of scientific space missions, thus providing a standard basis for cost-effective and technically compatible developments of on–board and on-ground data handling systems for a wide range of projects. This paper describes the design and the implementation of detector-independent software, running on Unix platforms, for near real-time acquisition, archiving, and basic processing of ESA-based telemetry and telecommand data produced during on-ground testing and calibration of various space-borne X– and γ–ray detectors.

1. Introduction

Every scientific payload that adopts the ESA Packet Telecommand (TC) and Telemetry (TM) Standards uses the TC and TM source packet structures to receive the commands which determine the payload set up and operations, and to send the data generated by its subsystems. These standards dictate the guidelines for the packet structure, formed by a header, a data field and a trailer. They also specify the information to be included in the header and in the trailer in order to allow the verification and decoding of the packet content. Among them, the Application Identifier (APID) is required to identify the subsystem which is the destination or the source of the TC or the TM packet, respectively. In addition, for a given APID, the Type/Subtype keywords identify the specific operating mode under which the data have been produced, and, then, the structure of the data contained in the data field.

Suitable Electrical Ground Support Equipment (EGSE) hardware and software are required in order to test and verify, before launch, all the payload functionality and performances. To this purpose, the EGSE provides the simulation of the relevant missing on–board subsystems, generates and sends to the payload the TC packets, receives and archives all the TM packets. By inspecting in near real time the TM reports and the TM housekeeping packets, the EGSE verifies the correct execution of the TC, and is able to monitor the payload health, as required to support the basic engineering tests. In addition, it is important that EGSE software allow the EGSE operator to easily verify all the different scientific operating modes, and, in particular, to verify and calibrate the scientific performance of the detectors illuminated with X– and γ–ray sources or beams.

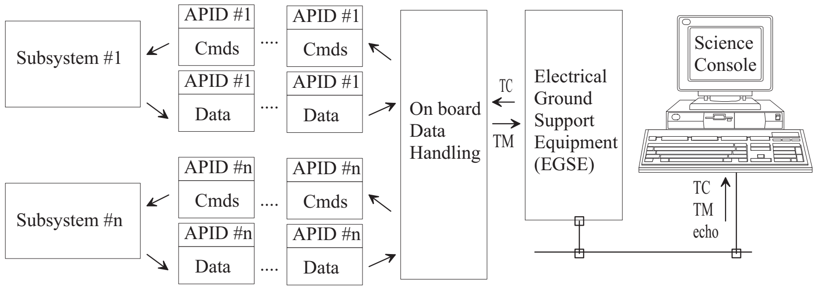

Figure 1. TM and TC packets data flow during ground tests.

2. DESIGN CONCEPT

The use of the ESA TC/TM Standards, allowed us to adopt the common design concept sketched in Figure 1, where the EGSE is limited to the engineering functionality, while the scientific functionality is provided by the Science Console, which receives in near real time, from the EGSE, a copy of all the TC and TM packets. The EGSE was procured by the industry, and the Science Console software was in charge of the scientific team responsible for the instrument acceptance test and calibration.

The DISCoS (Detector Independent Science Console Subsystem) software we present in the following sections, is the part of the Science Console software which was integrated with various detector specific software written in order to unpack the information contained in the TC/TM packets and to perform the quick look on the scientific data. The current version, which is being developed for the AGILE mission (Trifoglio et al. 2000), profits from the experience gained with the previous versions which have been exploited for the XMM–EPIC mission (Trifoglio et al. 1997; Trifoglio et al. 1998), and the INTEGRAL–IBIS mission (La Rosa et al. 1999; Trifoglio et al. 1999).

3. SOFTWARE ARCHITECTURE AND IMPLEMENTATION

The DISCoS software consists of the C programs Monitor, Receiver, Archiver and Provider, which allow the Science Console to acquire, verify, and archive in near real-time the TC/TM packets in one set of files for each measurement setup, and to reconstruct, either in live or in playback mode, the various streams of TM scientific packets pertaining to particular operating modes, i.e those having the same APID/Type/Subtype. Figure 2 sketches how these programs interact together and with the unpacking programs (Processors) and the quick look programs (Quick Look Analysis and Graphical Display) to be written by the DISCoS users.

Once started from a shell script, every second the Monitor updates a screen window once per second with status information and relevant parameters that the concerned programs write on the Shmmon shared memory by using specific routines. In addition, the Monitor program allows the operator to generates fake start/stop TC, to be used by the Science Console in order to close and open the measurements files independently from the TC packets generated by the EGSE.

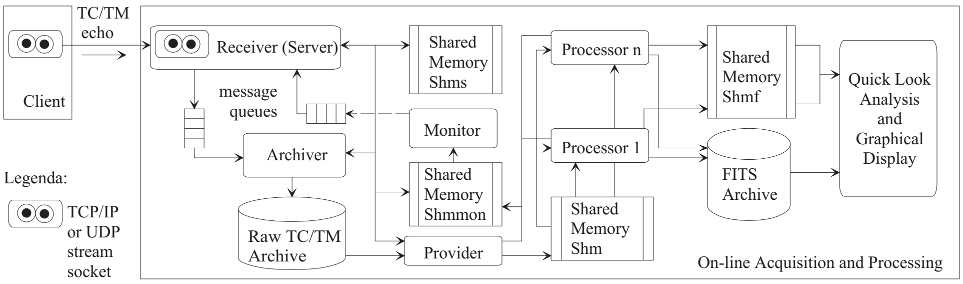

Figure 2. The Science Console software architecture.

The Receiver is the program which interfaces the EGSE through a TCP/IP or UDP socket on an Ethernet LAN 10/100 BaseT. Once started from a shell script, it waits as a Server, and, in the TCP/IP case, establishes with the EGSE a stream socket connection. Hence, using a fork the Receiver generates the Archiver program. As long as the connection is active, the Receiver performs a reading loop. On each iteration, by using a non blocking call it reads from the Monitor task message queue the next fake TC packet, if any; and by using a blocking call, it reads from the LAN in order to acquire the next TC/TM packet. For each packet, the Receiver inspects the header in order to verify the Application Identifier (APID), the Source Sequence Counter (SSC) and the packet length. Hence, by using a blocking call, the Receiver sends through a message queue the packet to the Archiver program, and restarts reading. In case a Start/Stop measurement command is detected, the Receiver waits until the Archiver message queue is empty, then sends a SIGINT signal to terminate the running Archiver, forks a new Archiver, and eventually restarts reading.

Every time it is started, the Archiver opens a new set of raw files: one to contain all TC/TM packets, and one for the TC and the TM Housekeeping only. Their locations and names are automatically derived from the progressive number (Run ID) which identifies the measurement. An additional suffix is used to identify the files containing the data generated during the instrument configuration. By using blocking calls, the Archiver reads each TC/TM packet from the message queue and writes each packet to the local disk using low level C–Unix Unformatted I/O routines. Upon receiving a SIGINT signal from the Receiver, the Archiver completes the reading and archiving of any TC/TM packet from the message queue, closes the raw files, and then terminates itself.

The Provider reads and sorts the TC/TM packets from the raw file. This program is run by a shell script either in live or in playback mode. In the former, the raw file name is derived from the current Run ID and the forthcoming TC/TM packets are read upon receiving the SIGUSR1 signal from the Receiver. In the playback mode, the program starts reading from the raw file, whose name has been provided by the script. For each packet, the Provider verifies the correctness of the packet header, sorts the packet by APID, and writes the packet in the column of the shared memory Shm assigned to the APID. Indeed, this shared memory is managed as a two–dimensional circular buffer capable of containing some hundreds of packets for each APID. Information exchanged with the Provider, through the shared memory Shms, allows each Processor to read new TC/TM packets having the required APID as soon as they are available in the shared memory Shm. A synchronization mechanism guarantees that no TC/TM packet is overwritten until it has been read by the related Processor. Unless a time-out occurs, before overwriting the Provider waits for a SIGUSR2 signal generated by the concerned Processor indicating that the new data have been already read.

In order to allow interfacing with detector-specific programs, the DISCoS software includes a set of C, Fortran and IDL callable routines. They allow the user to write the Processors as separated programs which have access to the monitor window and to the various source packet streams sorted by APID, without having to deal with either the EGSE interfacing or the TC/TM packet acquisition, verification and archiving. Usually, one Processor program is run for each APID. For each sorted stream, the Processor has access to 100% of the acquired packets, regardless of input rate and processing to be performed on the packet. A typical Processor reads the TC/TM packets from the shared memory Shm, verifies and archives the instrument data extracted from the packet data field in a format suitable for further off–line analysis (e.g., FITS format). Other DISCoS routines allows the Processor to communicate these data, through the shared memory Shmf, to another user's program, written in IDL, which is devoted to quick look purposes. Depending on the selected mechanism, on each call the quick look program receives the unpacked data related to either the next or the last packet processed by the Processor. A typical quick look program produces and displays in near real time further data products (e.g., time profile, spectra, and images) allowing the user interaction.

4. CONCLUSIONS

The DISCoS software presented herein has demonstrated its effectiveness and re–usability in the Science Consoles developed for several EGSE and Test platforms which have been designed adopting the ESA Packet Telemetry and Telecommand Standards for instrument data and commands transport structure. The software architecture and implementation have allowed a simple porting from the original HP–UX Unix workstation platform to Linux PC platforms with the GNU Compiler Collection (GCC).

References

La Rosa, G. et al. 1999, Proc. AIP Conference, 510, 693

Trifoglio, M. et al. 1997, Proc. of the Fifth Workshop Data Analysis in Astronomy, ed. V. Di Gesu' et al., World Scientific, 233

Trifoglio, M. et al. 1998, Proc. SPIE, 3445, 558

Trifoglio, M. et al. 1999, Proc. SPIE, 3765, 572

Trifoglio, M. et al. 2000, Proc. SPIE, 4140, 478