High Performance Instrument & Payload EGSE

Andy Armitage, Roger Patrick, Leif Hansen Paul Allen

Terma B.V
Schuttersveld 9
2316 XG Leiden, The Netherlands
E-mail:aba@terma.com

ABSTRACT

For some spacecraft, the instrument or payload is more complex than the satellite platform.

Often the EGSE for such instruments are required to handle data at rates far beyond the typical S-Band TMTC rate, e.g. to process space-wire or X-band TMTC data, or science data in real time.

Usually, these instrument EGSE have to simulate the spacecraft platform without the on board computer, or onboard software, and we have to deal with a spacecraft platform database that is frequently changing during the system integration and testing.

This paper discusses some of the features we would recommend for this type of project. When considering requirements and architecture for future common EGSE and MCS software (EGS_CC), these should not be forgotten.

INTRODUCTION

In our recent industrial experience, payload and instrument EGSE have proved far more challenging in terms of performance and sophistication than even the major spacecraft-level checkout systems. At the same time, customer expectations of an EGSE budget at instrument level are lower than at system level.

We consider here spacecraft where there may be a single "dominant" payload that constitutes by itself much of the mass, power consumption, electrical interfacing and software complexity of the spacecraft.

The instrument or payload supplier wishes to validate his instrument in the absence of the spacecraft platform, so that he can deliver a validated subsystem for integration on the spacecraft prime. The supplier also needs to supply a database and ideally some working test scripts and synoptic pictures. The usual job for the payload EGSE is to simulate the presence of the platform, including all the "intimate" internal interfaces of the spacecraft.

The test system (EGSE) for such an instrument typically has to manage

- Direct electrical power provision, discrete commanding, discrete and analogue measurement sampling, including thermistor and heater sampling and stimulation
- The onboard data handling bus, typically MIL-STD-1553-B and its mission-specific protocols
- The mission specific tailoring of ECSS Packet Utilisation Standard, usually applicable
- A science data interface that can generate telemetry at very high rates using typically space-wire or channel-link.

This paper discusses the software aspects of the test system.

Of course the electrical front-end equipment to such a payload is very significant, and this presents further challenges:

- the electrical front-end EGSE suppliers vary from one project to another, and the different suppliers do not support the same interfaces. Even when a "standard" interface is used, the content is always different for each payload
- the hardware elements of the EGSE justifiably consume the majority of the available budget

Instrument detectors often generate high-rate packetized telemetry that is sent directly to an onboard mass memory of the spacecraft platform that would typically be sent to an X-band telemetry link, whereas housekeeping is forwarded by the spacecraft platform to an S-band link.

For scientific instruments the EGSE end user often requires the capability to take a "Quick Look" into the generated science packets. The goal here is not to perform deep scientific analysis of the content, but to quickly validate and investigate possible problems with the content. Example scenarios are to validate that a test pattern injected to the detector is indeed generated at the instrument output, or to electrically / optically stimulate the detector and check for specific signals in the science data output. Such a "Quick Look" tool usually has to provide for some interpretation of the science, some graph plotting and some image display.

In our project experience, the end user requires a "Live" science data view, that presents the latest images, plots and derived data in real time (can be a major performance challenge), and an offline data view that allows investigative functions on archived data (hence a challenge to provide sophistication and flexibility).

In some instrument EGSE, real-time hardware-in-the-loop closed loop testing responses are required, that also require the presence of a hard-real time software simulation platform to guarantee a stimulus response within typically one millisecond of a signal.

There is typically an instrument controller, which may be deployed as an electronic box programmed in VHDL, or as a full onboard computer exchanging CCSDS standard TM and TC packets, PUS packets, its own dedicated mission time line, onboard control procedures, memory management. There may be onboard data encryption. In some cases there is a mix of intelligent (packet aware) and dumb (not packet aware) payload subsystems on the same bus. This requires huge flexibility in the EGSE.

Finally in current ESA programs, the instrument or payload manufacturer is required by the Agency or the prime contractor to provide validation evidence of the operability of their unit in a form that is compatible with system-level testing, and mission operations. Ideally the TMTC databases, scripts, and display definitions will be in a form that can be directly re-used, or re-used with minimal adaptation

PERFORMANCE FUNDAMENTALS

Although we may quail at the challenges listed in the previous section, we have to recall that the data rates seen today from spacecraft payloads and instruments are still far below the maximum data rates of the high-rate onboard electrical interfaces. Furthermore, in comparison with the data rates required in other "telemetry" processing applications (e.g. medical tomography, geological survey sampling, vehicle crash testing) the data rates required today are still at a level considered trivial.

The highest sampling rates seen in the spacecraft EGSE industry today are those used for launcher EGSE, however these sampling rates only need to be sustained for a short period of a few minutes. This means that an architecture can reasonably be based on temporarily storing all samples in memory. For payload and instrument EGSE, the sampling rates are continuously high, and must also be sustained for at least several days and up to a few weeks in a thermal vacuum test.

We are greatly helped by the fact that computer hardware has evolved to the point where a single multi-core 64-bit workstation using PCI-express, several gigabytes of memory and a hardware accelerated graphics card supporting 4 displays provides a platform with performance and a low price that we could only dream of 10 years ago. The memory, CPU, disk archiving throughput and capacity are such that there is no fundamental reason why we could not support the data rate requirements seen today.

The fundamental issue we face today is that software architectures designed only for processing S-band telemetry at relatively low rates are so riddled with design bottlenecks that they are of little practical use to the instrument manufacturer. It is simply not viable to impose a software kernel on the instrument supplier that is neither flexible nor performant.

In this category we include the well known legacy monitoring and control kernels such as SCOS2000 and Open Center. Only by distorting their intended use cases can these systems be adapted for instrument and payload EGSE. While this is not fundamentally impossible, it is expensive to deliver and maintain, and frustrating for the end user.

FUTURE COMMON CORE

ESA has recognised this issue and is in the process of initiating new developments to address the need for a common EGSE and mission control kernel that could be used at all levels across European industry.

Terma has valid experience to share in this evolution. Initially, our approach was to adapt the ESA SCOS2000 mission control system to EGSE use, and to invest in various performance and functionality improvements. This demonstrated some of the benefits that can be derived from a common TMTC kernel. However several years before the time of writing this paper, it became clear that this approach would have a limited scope for long-term exploitation, and Terma began developing a "light" system that maintains compatibility with SCOS2000 while improving flexibility and performance. By incremental development we were able to succeed in this development, and today all our projects at instrument and payload level are based on this new platform. The features described in this paper are based on this experience.

As we attempt to look into the future, we can see that price-performance demands will only increase. In order to be viable at instrument or payload level, any new "common core" kernel must offer a fundamental step-change in performance and flexibility. This can be achieved by different strategies

- selection of underlying technologies that are fundamentally best-performance
- maximize use of parallelism
- most efficient possible inter-process or inter-thread communication (or shared memory)
- identify appropriate tasks for hardware-acceleration
- possibility to configure a stripped down "light" version of the common core
- descope "luxury" functionality that has high performance cost

While this paper contains example features that are especially valuable at payload and instrument level, it also implies some warnings. Firstly, a "common core" without sufficient performance for payload or instrument EGSE level processing cannot truly be a common core. Furthermore, during a long development process of a common core, in parallel, performance requirements will further increase. This will carry the risk that the new system might be obsolete by the time it is completed.

To avoid these traps, the author would recommend that performance is kept at the top of the priority list, time to market (the inverse of complexity) is minimised, even if this is at the expense of functionality. The bottom line is that performance, availability, reliability, and ease-of-use will determine take-up.

ARCHIVING

Archiving is at the heart of spacecraft monitoring & control systems and is one of the chief determinants of performance. At the time of writing, hard disk writing performance is fundamentally capable of storing data far above the rates generated by spacecraft units, and we expect this to remain the case for some time to come.

What determines archiving software performance is usually the software method with which archived data is distributed to applications inside the overall system. We have considered the following issues

- most "live" user interfaces are in any case constrained by what can reasonably be presented to the operator. When watching a packet history display, log window, or science image, the operator does not need or want to see all data that has arrived, but only the relevant (usually latest) subset which could be the last 30 packets or log messages, or the most recent packets that compose one complete instrument image. Retrieving only the latest subset required for a display can greatly reduce demands on the archive
- many applications request access to the same data, e.g. packets or parameters, and often this is the most recently saved. In this scenario the operating system can help us, because disk pages that are most-recently and most-frequently accessed will be maintained in the buffer cache. Although a system call to read a specific page has a cost, in the vast majority of cases, the data requested will already be in memory.
- when an operator asks to retrieve or examine historical data, e.g. to step back and forth through packet or log displays, he is no longer in "live" mode, and then we are somewhat independent of the rate at which data is arriving. In this mode, we are more concerned that stepping back and forth, and performing filtered searches should be as efficient as possible. Nevertheless for an operator, waiting even a few hundred milliseconds in response to a tape-recorder button click can still be acceptable on a heavily loaded system. Historical clients can be decoupled from live ones, and historical clients have different priorities.
- If a client subscribes to more data than it can absorb, then the scope of any backlog or performance problem must be limited to only that client, and have no fundamental effect on the system archiving performance or distribution mechanism
- When the archive is massively loaded, it can be acceptable that indexed searching and filtering does not fully keep up with the actually archived data. Since the act of creating archive search indexes can itself be relatively slow, it is acceptable to allow archive indexing to catch up as best it can, so long as it does eventually catch up. Ensuring that archive indexing is offloaded to different CPU from the task performing the storage gives the archive indexing the maximum resources available to keep up. The effect that a client might see in such a scenario is that a filtered historical search does not see the very latest packets or logs. This can be perfectly acceptable since the operator is most likely trying to step to data from seconds ago, or older, not a few milliseconds ago.

It is vital to decouple the act of archiving (saving to non-volatile storage) from distribution. The first priority is to store, since stored data could, even in the event of a system crash be replayed and analysed. Assuming that data can be flushed to disk at an adequate rate, the key issue is to make the clients who are interested in the arriving data aware of new data in the most efficient way possible. At this point we must be extremely careful to ensure that the storage and distribution process is not itself impeded by the addition of a large number of "live" clients or "greedy" clients.

This is the point where legacy systems have created a fundamental bottleneck. If the data distribution mechanism relies on any kind of round-robin transmission of packets (e.g. sending packets one-by-one over TCP/IP sockets) then the system will slow down for every client added and eventually be overloaded. Further the such a system could be slowed down if one of the receiving clients does not read data from its socket fast enough.

Some form of non-blocking publish-subscribe mechanism is needed that is completely independent of the number of clients, and will not be blocked by a slow client.

Terma based its recent archive architecture on the distribution of multicast UDP packets called "heartbeats" which are sent at regular intervals. Each heartbeat could in theory announce the arrival of a stupendous number of records, far more than any client could process. However in reaction to receiving a heartbeat, any client may retrieve only the subset needed, or skip a specific heartbeat until it is ready to process more data. Actual data retrieval consists of simply reading the archive hard drive.

The Terma approach was rather crude, since we wished to be independent of commercial middleware tools. However this approach has been proven in some of the latest high-end EGSE. This solution has made it possible to process high-rate X-band data (e.g. recorded housekeeping) or science data, in systems that previously were not capable of doing so. It also has the benefit that the only fundamental tools needed by an archive client are the ability receive UDP packets and to read a hard drive.

This approach also allows for integration with third party tools and front end equipment. In the distribution direction, it is possible for other third party "subscribers" to listen-in on the heartbeat with no effect to the archive. Conversely we have also used this approach when the heartbeat source is an electronic front end saving science data, and the subscribers are "Quick Look" applications in the EGSE.

In any future architecture we would strongly recommend that the archiving and distribution mechanism is based on a publish-subscribe mechanism where the archiving source is completely unaware and completely decoupled from the clients. This could be achieved via use of appropriate *high-performance* off-the-shelf tools or by the basic mechanism described here.

SCIENCE PROCESSING

The instrument EGSE clients that are most demanding in terms of data processing are typically the "Quick Look" applications, especially when they attempt to display images, graphs or plots of live data and even processing several streams in parallel.

To repeat, when these applications cannot keep up with the data arrival rate, it is usually acceptable to display only data from the latest set of packets from a complete image. When the user requests to drop out of live mode, the packets can be retrieved at an almost leisurely pace.

In our projects we have used the commercial tool MATLAB for science processing because

- MATLAB is popular with the consumer of this type of data, and the end user is often familiar with how to program MATLAB scripts

- Since the science data format is always unique to a given instrument, we can define useful MATLAB libraries for the specific mission. The expert customer is then free to extend or write his own batch processing scripts in MATLAB
- MATLAB image, plotting and mathematical libraries are superior to any that could be custom developed. Further a huge global ecosystem of end users means that many free libraries can be downloaded from the web.

We have added MATLAB libraries to receive and process the above mentioned archive heartbeats, and also some basic libraries allowing MATLAB to independently extract and calibrate telemetry parameters according to the ESA SCOS2000 MIB database format.

In all projects to date, MATLAB performance has been sufficient to process the full live science data rate; however we always maintain the design principle that in the event of a flood of science data, the Quick Look tool will skip to showing the latest images. This ensures that the system is robust to unforeseen bursts of data.

MATLAB has limitations that were somewhat disappointing. For example we do not find the MATLAB tool-boxes for parallel processing to be in any way superior to running multiple instances of a MATLAB application. When running on a multi-core CPU we allow MATLAB instances to run on different CPU cores, which is both simple and avoids the additional cost the MATLAB tool-box.

DATABASE PARTITIONING

For instrument manufacturers, dealing with a TMTC database can be a cause of frustration and lost time. While we cannot remove the complexity of understanding and populating a MIB or space system model, there are practical ways that we can help to do this job more quickly than in legacy systems. Recall that if TMTC descriptions were not needed, most instrument suppliers would be happier to use tools such as LabView, Test Stand or even VB script.

A typical DB utilisation cycle is illustrated in Fig. 1

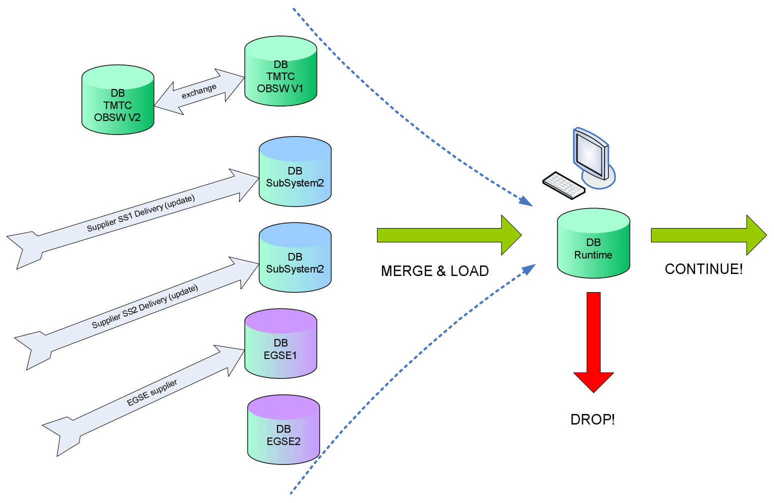


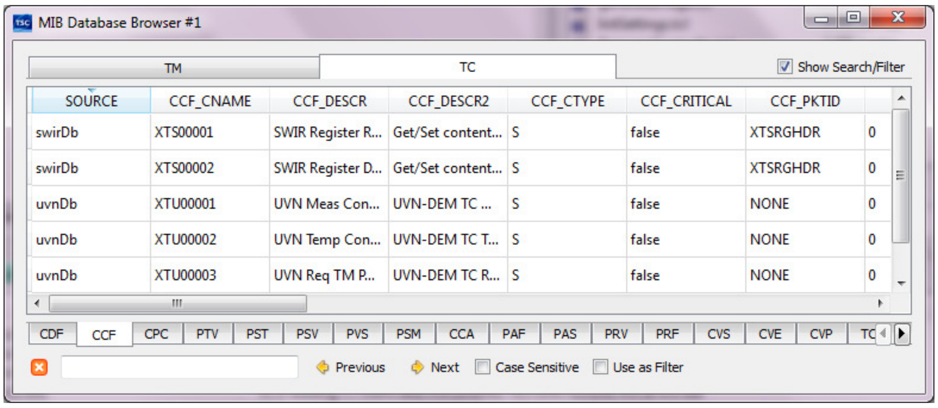
Fig. 1. Database Merge, Load, Drop, Continue Cycle

In Terma systems the database related optimisations comprise

- databases from different sources (e.g. platform, instrument, unit, EGSE) may be kept apart and are merged on-the-fly. An arbitrary number and locations of databases can be merged. This means that when one database is updated or replaced, the end user need only replace or reconfigure the load path to the new set of files for the unit being changed.

- databases can be dropped or loaded on the fly, without restarting a session. It is not even necessary to terminate all test sequences, since these will be automatically suspended and resumed during the drop-load cycle.

The currently loaded database can be directly examined in the user interface, sorted, searched and filtered, including the logical source of the data. Given that TMTC definition names can be cryptic, it can be very useful to be able to search and filter what is actually loaded. An example simple runtime database browser is illustrated in Fig. 2



SOURCE	CCF_CNAME	CCF_DESCR	CCF_DESCR2	CCF_CTYPE	CCF_CRITICAL	CCF_PKTID	
swirDb	XTS00001	SWIR Register R...	Get/Set content...	S	false	XTSRGHDR	0
swirDb	XTS00002	SWIR Register D...	Get/Set content...	S	false	XTSRGHDR	0
uvnDb	XTU00001	UVN Meas Con...	UVN-DEM TC ...	S	false	NONE	0
uvnDb	XTU00002	UVN Temp Con...	UVN-DEM TC T...	S	false	NONE	0
uvnDb	XTU00003	UVN Req TM P...	UVN-DEM TC R...	S	false	NONE	0

Fig. 2. Database Browser

In current systems, the database is based on the ESA SCOS2000 MIB, in future this is likely to change. The principle remains that the end user must be adequately supported when the database consists of many separate parts, is being developed and frequently changing.

SIMULATION & SVF SUPPORT

All payload & instrument EGSE include some element of simulation, since we are usually "simulating" the presence of the spacecraft platform. Whether a classic simulation platform such as EuroSim is required depends on the requirements for closed-loop-hard-real-time-hardware-in-the-loop interfacing. For some quite complex EGSE, the simulation can be done entirely in test scripts, which have adequate, but not hard-real-time performance.

For Software Validation Facilities (SVF) we may have other needs for a simulation platform, because we may want to emulate the onboard computer and its surrounding equipment and environment.

In any scenario including such a separate simulator, we need to allow test automation that allows adequate control, monitoring, error injection, loading and saving of breakpoints, start, stop, pause and resume controls from the test language. Typically this is achieved using a language plugin dedicated to the simulation platform. For the future, Terma is considering a generic language plugin for any simulation platform, so that the scripts produced are independent of the simulation platform. We are not sure of the demand for this feature, and would welcome inputs from end-users.

One extremely useful feature for simulation and SVF support is the ability to drive test script timing and telecommand verification from an external time signal derived from the simulation or emulation platform. These platforms can pause, resume, accelerate and slow down by user command or because of free-running execution. Unless special support for this scenario is implemented, then test scripts and telecommand verification will fail because of variations in the timing of the simulated unit. We name this feature "SVF mode"; when enabled, the kernel onboard time model is driven from an external time pulse from the simulator. If this pulse pauses, resumes, accelerates or slows down, then the corresponding scripts and telecommand verifications will follow the external time signal in preference to the local computer clock. This allows the user to verify his test scripts and databases (containing TC verification timers) as though the simulated unit were running in real time, even though it varies. For the future we would like to consider also the possibility of saving the complete timer, TM parameter and TC verification state, as the equivalent to a simulation breakpoint; whether this can realistically be achieved remains to be seen.

Another extremely useful simulation feature is the ability to generate simulated TM packets from their database definition, even when the contents are not acquired through TM packets. We can consider an acquisition sampling system that acquires samples via a mixture of electrical front-end equipment, simulation platforms, GPIB/SCPI, and even SNMP. The scenario is shown in Fig. 3.

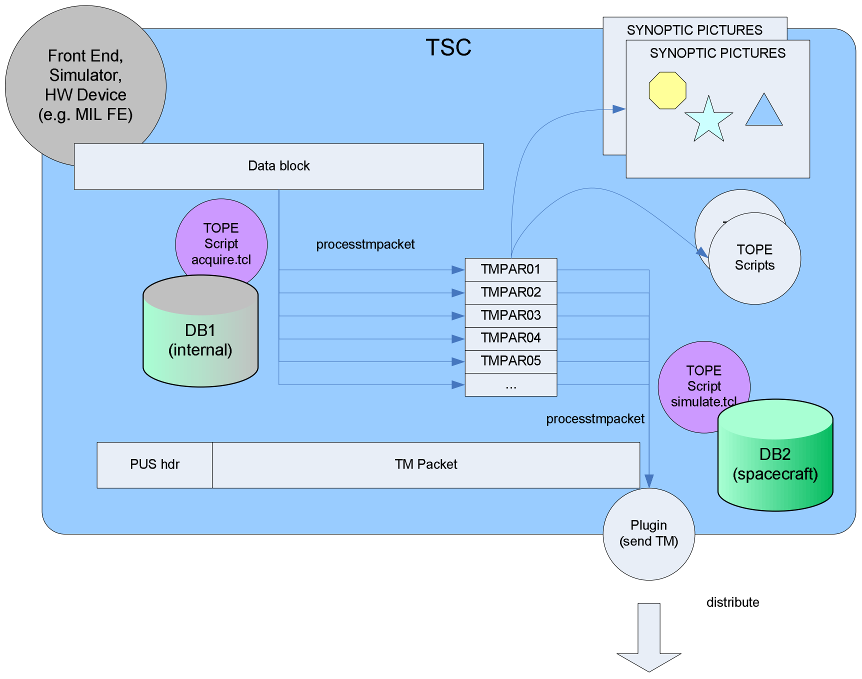


Fig. 3. TM simulation

In this scenario, data (either directly as raw parameters or as blocks not packets) is acquired via one or other device interface. These can be locally published according as raw parameters or blocks internally within the system. For blocks, these are processed into parameters "as though" they were prefixed with a classic TM packet header and can then be processed locally using test scripts and displayed exactly as though they came from a TM packet. Beyond this, the system is also capable of using another packet definition (e.g. a spacecraft platform packet) to place the same parameter samples in a simulated packet, generated according to its database definition. This means that the instrument EGSE (simulating the platform to the instrument) can also simulate the platform to some other system (e.g. the central checkout system) by sending this generated packet elsewhere. The generated TM packet looks just as though it were generated by the spacecraft onboard computer. If the TMTC database of the platform changes, then the layout of the generated packet would also change accordingly.