A GENERIC EGSE

Martin Grim(1)*, Bart van Kuik*(1)*, Frederic Lemmel*(1)*, Joris van Rantwijk*(1)

SRON, Netherlands Institute for Space Research (1)
Sorbonnelaan 2, 3584 CA Utrecht, The Netherlands
Emails:
w.m.grim@sron.nl, *b.van.kuik@sron.nl*, *f.j.f.lemmel@sron.nl*, *j.f.van.rantwijk@sron.nl*

ABSTRACT

This paper describes a generic and versatile EGSE designed to allow a smooth transition of command, control, telemetry, analysis and visualization from early lab systems and development models to full-fledged instruments integrated on a satellite.

Key concepts used to achieve this versatility are ‘modularity’, ‘hardware abstraction’ and the use of a ‘flexible communication layer’. Project specifics are dealt via dedicated scripting.

The SRON ‘Generic EGSE’ is successfully deployed in several short and long-term projects (Eureca [1], Tropomi [2] FEE development, Spica/Safari [3] development).

CONTEXT

The ‘Generic EGSE’ was developed at SRON with the ultimate goal to operate as a standardized tool to help the user to command, control, visualize and analyse instruments in all different stages of a certain project. From early lab systems, prototypes and engineering models to full flight models integrated into satellites.

Initially, this ‘Generic EGSE’ was developed to support detector development for Eureca [1], Xeus [4], IXO/Athena [5], but was quickly adopted in many other SRON projects. For the Tropomi [2] Front End Electronics, this EGSE is used throughout development from EM (Engineering Model) to FM (Flight Model). For Safari [3], this EGSE will be used for the entire EM, DM (Development Model) and QM (Qualification Model) phases. For the Safari [3] flight model the JAXA GSTOS [6] system is expected to be used.

At the start of the development of the ‘Generic EGSE’ an investigation was held on the variety of UCE/EGSE systems. It appeared that most of these were not suited to support lab and prototype systems which are in a great state of flux. An EGSE must be able to quickly support a new topology (adding/removing equipment, migrating to different cryostats, etc.) of system under test (SUT). Also, investigations showed most systems to lack proper capabilities to display (level 0) telemetry in all kind of different ways.

Closely coupled, most investigated EGSE systems were not able to view telemetry after desired computations immediately during a measurement, which is for early detector development, prototypes and engineering models essential.

These two observations, quickly adaptable to a changed SUT and better online analysis, are the basis for the 'Generic EGSE'. A vital addition is that the flexibility of the 'Generic EGSE' is available to the different users (experimenters, developer, and operator) at all times.

BUILDING BLOCKS

The 'Generic EGSE' is compound of the following elements as described in Fig. 1. :

- A EGSE server that is the central application. It functions mainly as a graphical user interface for the end-user to command, control and visualise the system under test. The EGSE server also functions as router for the data flow towards the hardware daemons.
- Hardware daemon (HW Daemon) applications handle communication with all equipment of the system under test (instrument itself, network analysers, power supplies, etc.). The hardware daemons perform the necessary protocol conversions between the internal generic EGSE protocol (CCSDS based) to the protocol of the equipment (e.g. GPIB) and hide the actual interface (serial, Ethernet, SpaceWire, etc.) used.
- A database that contains all specific parameters for the project.
- A scripting language (python) to control the system under test and to perform calculations with.

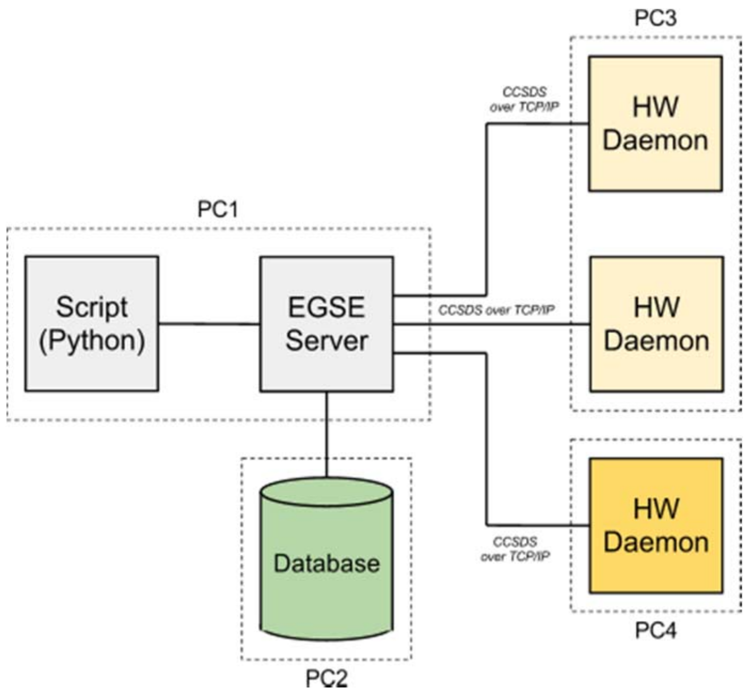


Fig.1. An EGSE environment

DISTRIBUTED ARCHITECTURE

The ‘Generic EGSE’ is designed as a distributed architecture. An XML configuration file describes the complete EGSE topology. At start-up, the EGSE server and all necessary hardware daemons are launched and configured as desired. Adding equipment to the SUT only requires an additional entry in the XML configuration, which is fairly trivial. An example of a distributed EGSE environment is depicted at Fig.1.

The main advantage of a distributed architecture is its modularity. It is easy to build a EGSE environment and to add or to replace an element to this topology (the EGSE server, the database, HW daemons) depending of the evolution of the project.

HARDWARE ABSTRACTION

The ‘Generic EGSE’ makes use of hardware abstraction by means of a simple relational database with all parameters of the system under test. Each parameter is defined in terms of an application identifier to which it belongs (e.g. which actual piece of hardware), values it can take, conversion polynomials, warning levels, consequent actions, etc.

Any action to be executed to the system under test is done in terms of ‘get’ or ‘set’ actions. Upon every telecommand issued by the commanding script, a telemetry response is sent by the hardware holding either a single value (for simple get/set actions) or a range of values (retrieval of block data).

The EGSE server nor the hardware daemons nor the underlying protocol (see below) have any knowledge of the parameters.

Adding hardware results in adding a new table to the database with parameters describing the newly added hardware. This is fairly trivial work which can be done by the operator who extents the system under test.

INTERNAL PROTOCOL

The ‘Generic EGSE’ uses a CCSDS based protocol for communication between the script, EGSE server and hardware daemons. This protocol defines a generic and flexible structure: each telecommand or telemetry packet consist of one or multiple triplets containing: an action, a parameter and one value or more if necessary.

More precisely, the header of a CCSDS based packet is always composed of :

- A Primary Header that contains information about the receiver.
- A Secondary Header that contains a timestamp and checksum of the packet.

Then, the action, parameter and value(s) are encoded in:

- A Data Header that describes the type of action (set or get), the parameter and the type of data.
- A optional Data Header that contains more information about the data.
- A single data value or a block of data that contains the value(s) to send or to receive.

The CCSDS based packet is ended by a trailer and can contain one or more Data Header/Optional Header/Data (see Fig. 2.).

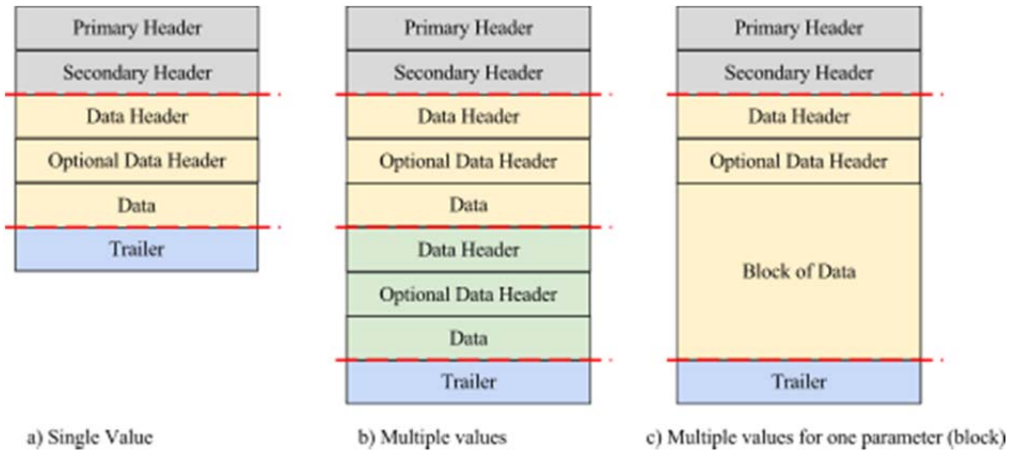


Fig. 2. CCSDS based protocol with single, multiple values and block of values.

The protocol terminating entity (either FPGA or embedded software in case of hardware or the hardware daemon in case of a protocol conversion, see below) interprets the action for the given parameter with given value. All these entities are able to deal with the structure of the protocol.

Thus, the protocol doesn't define a datagram with fixed parameters at fixed positions, but a datagram of which the contents can be changed at any time at the cost of slightly more overhead.

PROTOCOL CONVERSIONS

The ‘Generic EGSE’ supports a wide range of hardware: from off the shelf power supplies, digital current/voltage meters, network analysers and so on, to dedicated electronics for a (future) space instrument.

The ‘Generic EGSE’ uses internally a CCSDS based protocol for all data communication. However, off the shelf equipment or dedicated instruments often differ from this protocol and hence a protocol conversion is required. Also, not every piece of equipment is connected identically: nowadays one can choose from many different interfaces: SpaceWire, RS232/RS422, (optical) Ethernet, GPIB, and so on.

The hardware daemons run distributed on a PC locally to the equipment under test. As described in Fig. 3, these daemons contain the following units:

- a TCP/IP connection core to send and to receive CCSDS packet over TCP/IP.
- A CCSDS core.
- A Finite State Machine (FSM), the control entity of the hardware daemon.
- A flexible translation layer to convert the internal CCSDS protocol to the protocol used by the equipment (e.g. RMAP) and vice versa.
- An interface to communicate with the instrument (e.g. SpaceWire).
- A 'log to file' unit.
- A housekeeping core.

Besides, upon execution the daemons "choose" what kind of interface with which properties shall be used, based on the data in the XML configuration.

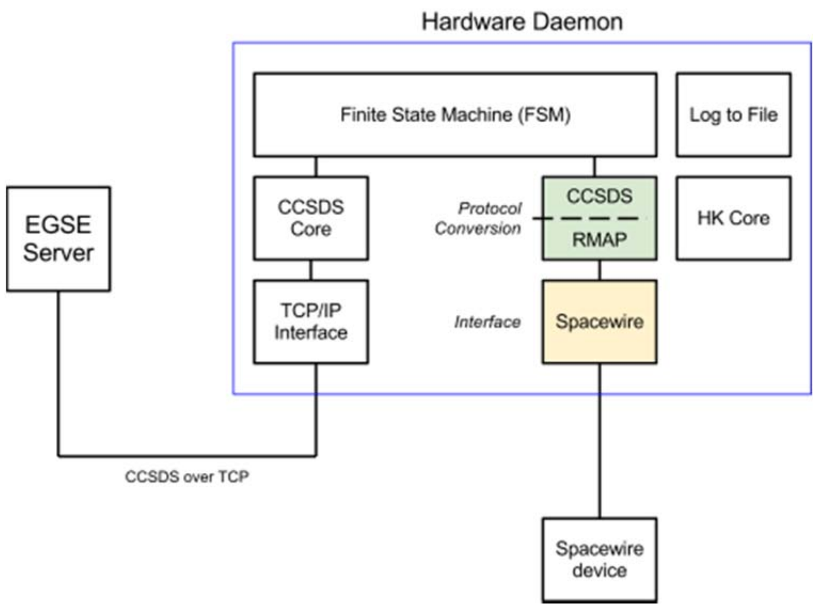


Fig. 3. Hardware Daemon with an example protocol conversion from CCSDS to RMAP over SpaceWire.

SCRIPTING

Every system under test is controlled via a set of python scripts. By means of a GUI addition of python a commanding menu can be made holding all desired actions to be executed by a user. The script and menu are tailor made for each project.

The script holds routines for configuring the system under test and routines for analysing the telemetry. In a typical test the user first configures the system as desired, then starts the measurement (by triggering a start command) after which the hardware will generate telemetry. This telemetry is forwarded by the hardware daemon to the EGSE server to the script. The script can now perform all kinds of mathematical analysis.

Automatic scripts for regression tests are supported as well: all user intervention is now scripted into a series of configuration actions, telemetry retrieval, possible analysis and storage of data.

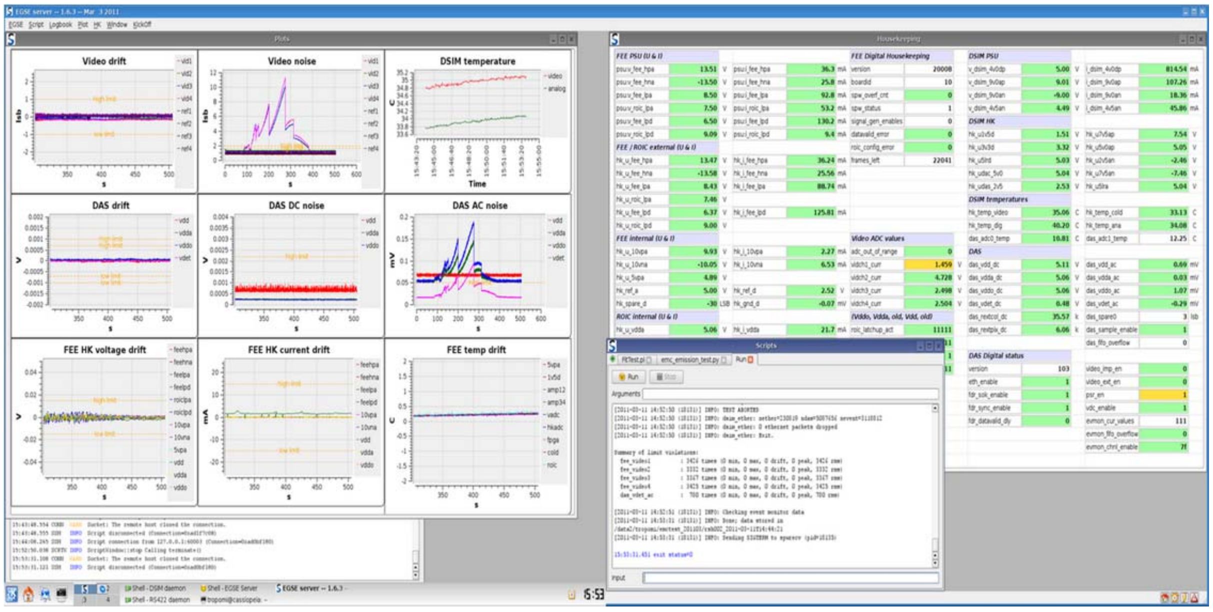
A set of python libraries is provided to support several functions like set, get or plot. These libraries are straightforward to use and they function as a bridge between the commands available to user and the CCSDS based protocol carrying the commands towards the EGSE server and HW daemons.

GRAPHICAL USER INTERFACE

As said, the EGSE server holds a graphical user interface. This GUI is very light weight and shows the status of all daemons and hardware belonging to the system under test. Also, the GUI allows for manipulation of the database and of the scripts.

By means of the GUI, the user can define and show housekeeping for arbitrary parameters defined in the database. Limit checking on these housekeeping parameter is automatically performed and, if defined, consequent actions are taken automatically as well. Again the consequent action is easily configured by the user.

The GUI also allows for defining graphs of arbitrary parameters as seen in Fig. 4. As the EGSE server listens in on all communication containing telemetry data and the plots are automatically updated when new data is available for the defined graphs. If a mathematical calculation is to be performed first by the script, the script has means to push the updated telemetry data back to the EGSE server which will then display the updated telemetry.



FEE PSU (U & I)						*FEE Digital Housekeeping*		
psu.v_fee_hpa	13.51	V	psu.i_fee_hpa	36.3	mA	version	20008	
psu.v_fee_hna	-13.50	V	psu.i_fee_hna	25.8	mA	boardid	10	
psu.v_fee_lpa	8.50	V	psu.i_fee_lpa	92.8	mA	spw_overf_cnt	0	
psu.v_roic_lpa	7.50	V	psu.i_roic_lpa	53.2	mA	spw_status	1	
psu.v_fee_lpd	6.50	V	psu.i_fee_lpd	130.2	mA	signal_gen_enables	0	
psu.v_roic_lpd	9.09	V	psu.i_roic_lpd	9.4	mA	datavalid_error	0	
FEE / ROIC external (U & I)						roic_config_error	0	
hk_u_fee_hpa	13.47	V	hk_i_fee_hpa	36.24	mA	frames_left	22041	
hk_u_fee_hna	-13.58	V	hk_i_fee_hna	25.56	mA			
hk_u_fee_lpa	8.43	V	hk_i_fee_lpa	88.74	mA			
hk_u_roic_lpa	7.46	V						
hk_u_fee_lpd	6.37	V	hk_i_fee_lpd	125.81	mA			
hk_u_roic_lpd	9.00	V						
FEE internal (U & I)						*Video ADC values*		
hk_u_10vpa	9.93	V	hk_i_10vpa	2.27	mA	adc_out_of_range	0	
hk_u_10vna	-10.05	V	hk_i_10vna	6.53	mA	vidch1_curr	1.459	V
hk_u_5vpa	4.89	V				vidch2_curr	4.728	V
hk_ref_a	5.00	V	hk_ref_d	2.52	V	vidch3_curr	2.498	V
hk_spare_d	-30	LSB	hk_gnd_d	-0.07	mV	vidch4_curr	2.504	V
ROIC internal (U & I)						*(Vddo, Vdda, old, Vdd, old)*		
hk_u_vdda	5.06	V	hk_i_vdda	21.7	mA	roic_latchup_act	11111	

DSIM PSU					
v_dsim_4v0dp	5.00	V	i_dsim_4v0dp	814.54	mA
v_dsim_9v0ap	9.01	V	i_dsim_9v0ap	107.26	mA
v_dsim_9v0an	-9.00	V	i_dsim_9v0an	18.36	mA
v_dsim_4v5an	4.49	V	i_dsim_4v5an	45.86	mA
DSIM HK					
hk_u1v5d	1.51	V	hk_u7v5ap	7.54	V
hk_u3v3d	3.32	V	hk_u5v0ap	5.05	V
hk_u5lrd	5.03	V	hk_u2v5an	-2.46	V
hk_udac_5v0	5.04	V	hk_u7v5an	-7.46	V
hk_udas_2v5	2.53	V	hk_u5lra	5.04	V
DSIM temperatures					
hk_temp_video	35.06	C	hk_temp_cold	33.13	C
hk_temp_dig	40.20	C	hk_temp_ana	34.08	C
das_adc0_temp	10.81	C	das_adc1_temp	12.25	C
DAS					
das_vdd_dc	5.11	V	das_vdd_ac	0.69	mV
das_vdda_dc	5.06	V	das_vdda_ac	0.03	mV
das_vddo_dc	5.06	V	das_vddo_ac	1.07	mV
das_vdet_dc	0.48	V	das_vdet_ac	-0.29	mV
das_rextcol_dc	35.57	k	das_spare0	3	lsb
das_rextpix_dc	6.06	k	das_sample_enable	1	
			das_fifo_overflow	0	
DAS Digital status					
version	103		video_imp_en	0	
eth_enable	1		video_ext_en	0	
fdr_sok_enable	1		psr_en	1	
fdr_sync_enable	1		vdc_enable	1	
fdr_datavalid_dly	0		evmon_cur_values	111	
			evmon_fifo_overflow	0	
			evmon_chnl_enable	7f	

```
[2011-03-11 14:52:50 (18131)] INFO: TEST ABORTED
[2011-03-11 14:52:50 (18131)] INFO: daim_ether: nether=130019 ndam=5007454 nevent=1110012
[2011-03-11 14:52:50 (18131)] INFO: daim_ether: 0 ethernet packets dropped
[2011-03-11 14:52:50 (18131)] INFO: daim_ether: Exit.

Summary of limit violations:
  fee_video1        : 3424 times (0 min, 0 max, 0 drift, 0 peak, 3424 rms)
  fee_video2        : 3332 times (0 min, 0 max, 0 drift, 0 peak, 3332 rms)
  fee_video3        : 3367 times (0 min, 0 max, 0 drift, 0 peak, 3367 rms)
  fee_video4        : 3423 times (0 min, 0 max, 0 drift, 0 peak, 3423 rms)
  das_vdet_ac       :  700 times (0 min, 0 max, 0 drift, 0 peak, 700 rms)

[2011-03-11 14:52:51 (18131)] INFO: Checking event monitor data
[2011-03-11 14:53:31 (18131)] INFO: Done; data stored in
/data2/tropomi/emctest_201103/rsh002_2011-03-11T14:44:21
[2011-03-11 14:53:31 (18131)] INFO: Sending SIGTERM to egserv (pid=18133)

15:53:31.451 exit status=0
```

```
15:43:48.554 COMM  WARN  Socket: The remote host closed the connection.
15:43:48.555 SSM   INFO  Script disconnected (Connection=0xadf7c08)
15:44:08.245 SSM   INFO  Script connection from 127.0.0.1:60003 (Connection=0xad8bf180)
15:52:50.038 SCRTW INFO  ScriptWindow::stop Calling terminate()
15:53:31.108 COMM  WARN  Socket: The remote host closed the connection.
15:53:31.121 SSM   INFO  Script disconnected (Connection=0xad8bf180)
```

Fig. 4. Example of a EGSE server with plots and housekeeping visualization.

The EGSE server can sustain a data flow of 320 Mbits/s with plotting and housekeeping visualization.

PLATFORM

The generic EGSE consists of two C++ applications (the EGSE server and the hardware daemon) based on Qt under Linux. However, nowadays the generic EGSE is being migrated into a multi-platform system supporting Linux, OSX and Windows. Plotting of graphs is performed by means of Qwt. The database uses MySQL and for the scripting python with PyQt is used. Mathematical analysis is done with NumPy.

As the applications run on normal PCs no dedicated hardware is necessary. Nowadays, the capabilities of off the shelf computers is such that these are fast, flexible and easy exchangeable.

CONCLUSION

The SRON 'Generic EGSE' is a very flexible tool that allows for a smooth transition from early lab systems to full-fledged instruments. The use of hardware abstraction via a database with parameters supports this smooth transition, as does the distributed architecture of the EGSE server with its hardware daemons and the flexible internal CCSDS protocol.

The hardware daemons support a wide range of protocols and interfaces, allowing for quick addition of new hardware without the necessity for new development.

System control is achieved via get/set telecommands issued via a commanding script, which is under control of the user. Housekeeping with limit checking and consequent actions is supported fully. Immediate online analysis of the (mathematically changed) telemetry is a key feature which allows for quick insights into the system under tests.

REFERENCES

[1] 'Eureca (EURopean-JapanEse Calorimeter Array)', http://www.sron.nl/instrument-studies-hea-menu-340/eureca-hea-menu-893.

[2] 'TROPOMI (TROPOsferic Monitoring Instrument)', http://www.sron.nl/sentineltropomi-eos-missions-2302.html .

[3] 'SAFARI', http://www.sron.nl/spica-safari-missionsmenu-2253.html.

[4] 'XEUS', http://www.sron.nl/xeus-missionsmenu-1913.html.

[5] 'IXO (International X-ray Observatory), ATHENA (Advanced Telescope for High-Energy Astrophysics)', http://www.sron.nl/ixoathena-hea-menu-2035.html.

[6] 'GSTOS (Generic Spacecraft Test and Operations Software)', http://www.c-soda.isas.jaxa.jp/sog/new_gaibu_Eng/gstos/gstos1.html.