Minutes of the topical meetings on SOFTWARE (s/w)
EURECA Project Design Review
Barcelona, 21-24 November 2005

Attendants Wed 23rd: Jan-Willem den Herder, Jan van der Kuur (SRON); Aliko Mchedlishvili (PSI); Noriko Yamasaki (JAXA); Stéphane Paltani (ISDC); Matt Page (MSSL); Javier Bussons (IFCA).

Attendants Thu 24th: Javier Bussons, Xavier Barcons, Maite Ceballos (IFCA); Aliko Mchedlishvili, Wojtek Hajdas (PSI); Chris Whitford (Leicester); Matt Page (MSSL), Jan-Willem den Herder (SRON).

1) Creation of the software group.

- Goal: design and development of software for quicklook and data analysis. Two phases: until the end of 2007, s/w is a part of the instrument development; after that, show performance improvements due to s/w capabilities and how they might be implemented on board of a space instrument.
- Interested parties and manpower estimates:

  MSSL: ½ person for a year. Matt is to report to his director about our current needs and they will decide on the suitable person. Interested in instrument simulation and test s/w.

  ISDC: up to 2 persons. Stéphane to find out who and for how long. S/W engineer profile with experience on INTEGRAL s/w more likely than astrophysics profile.

  IFCA: Javier at 50%, Maite at 25%, maybe a third person early next year for 9 months (with either a signal processing, information theory or software engineer profile as needed).

  SRON: Jan to provide interaction between s/w and h/w crews as needed. Jan-Willem is helping the s/w group to take shape.

  PSI: Aliko as advisor for the first-level s/w (FPGA level, on-board and EGSE s/w, data reduction in h/w). Maybe an extra ½ person to help with this.

  Leicester: Chris to advise on signal processing algorithms.

2) Identification of key issues and priorities.

First attempt at defining the goals of this group and the steps towards these goals through a "shower of ideas" (most important first):

- Identify, for each test in de Korte's "Instrument Modes" document, the input and output variables as well as the parameters involved in the process. We need to know at any given time which variables are set by the user (and where) and which ones are read out of the instrument (and where). Keep in mind that, for example, we may want to read the actual value of a parameter previously set by the user.
- Data reduction (sometimes indirectly referred to as *rebin*). In many cases the data flow will be so high that data reduction becomes a key issue. Possible combinations of low-pass filters, clever storage of the high-frequency part and, in general, any ideas as to how to decrease the amount of data stored must be studied with high priority.
- Define run, its duration, the component files and the units where the data in them come from, the naming convention, headers, data blocks, etc.
- Logging. Which commands, messages, settings, etc., must go to a log file and how.
- Explore various sets of trigger criteria (mono- or multi-pixel, single- or multi-threshold, ...)
- Routines to deal with pile-up and cross-talk.

3) From conceptual functionality to physical layout.

The discussion then shifted to both the conceptual and the physical structure of the required software. We elaborated further on the data types and processing sequence outlined in Javier's plenary session presentation and came up with Figures 1 and 2, which complement each other and only represent possible approaches.

- In Figure 1, the Command and Control device (C&C) includes the Field-Programmable Gate Array (FPGA) logic, the demultiplexor (Demux) and the Digital Signal Processing (DSP).
- Analog data from all four channels is sent separately to a digitiser (ADC_1) which spits out four separate data sequences (only one is shown), each containing digital, multiplexed data (~20MHz) from a given channel, i.e. 6 (or 7) pixels.
- At this stage, we may want to divide each pixel dataset into low-, medium- and high-frequency subsets (via low-pass filters, triggering and FFT algorithms which may or may not reside in the DSP) in order to reduce the data before it is sent to the Electronics Ground Support Equipment (EGSE). A clever way to do this must be found in order not to lose relevant information.
- If no reduction or triggering is made (Untriggered Mode), the bulky data lot is either sent via IP protocol to the EGSE or directly dumped to a big storage disk. Since the huge rates involved would prevent any command from the EGSE to make it to the C&C, a communication line (RS422) between EGSE and C&C must be arranged.
- In the Triggered Mode, only the demultiplexed data passing a certain hardware trigger is sent to the EGSE (how do we keep data immediately before and after an event?). In this case, the traffic between C&C and EGSE is low-rate.
- Figure 2 aims at complementing Figure 1 by stressing aspects such as the origin of the data (either from readings of input/output voltages and external units or from the Flux-Locked Loop FLL output I(t)), the various data types and corresponding storage units, the spy buffer used for quicklook and the different paths to the user display.
- Readings of voltage settings, intermediate or output voltages and external unit parameters (temperature, etc.) produce very-low-rate data which can all be stored together in Storage A.
- For each channel, the FPGA/Demux board transforms the incoming data set (~20MHz) into 6 sets of ~200kHz demultiplexed data corresponding to the 6 pixels in that channel.
- Untriggered-mode data (huge-rate $I_i(t)$) is dumped to Storage B whereas triggered-mode data (low-rate $I_i(t)$) is stored in Storage C.
- In Untriggered Mode, a data throttle (also known as spy buffer) randomly picks a small fraction of the outgoing data stream (for instance, one every N seconds), accumulates statistics and allows quicklook results to be presented to the user.
- Triggered data undergoes filtering, calibration, event parametrisation, spectrum derivation, correlation search, etc., the results of which are sent to the user display.
- The user must be able to use the data in Storage A for selection, correlation or display in combination with FLL data from Storage B or C.

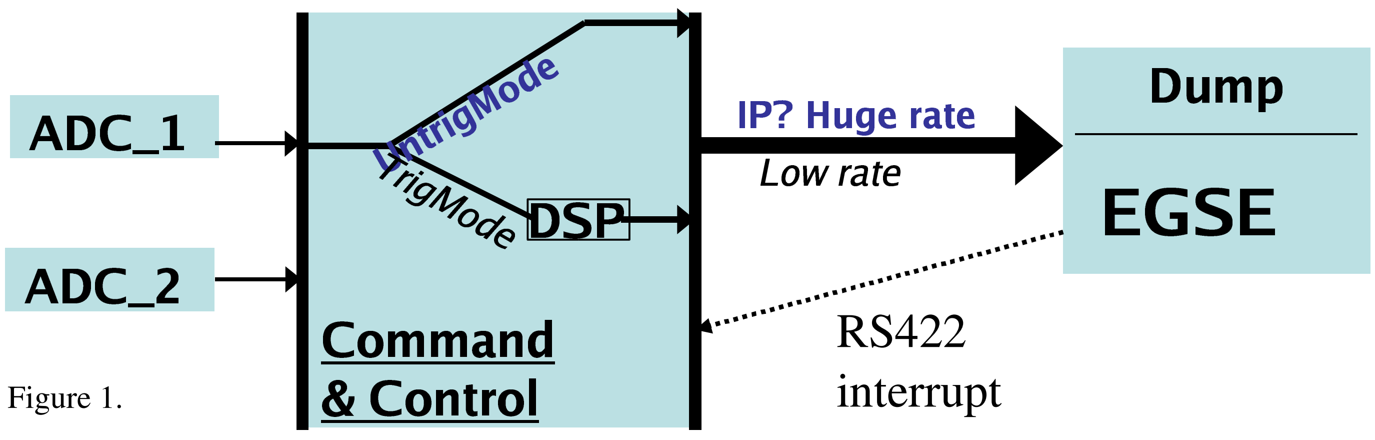

Figure 1.

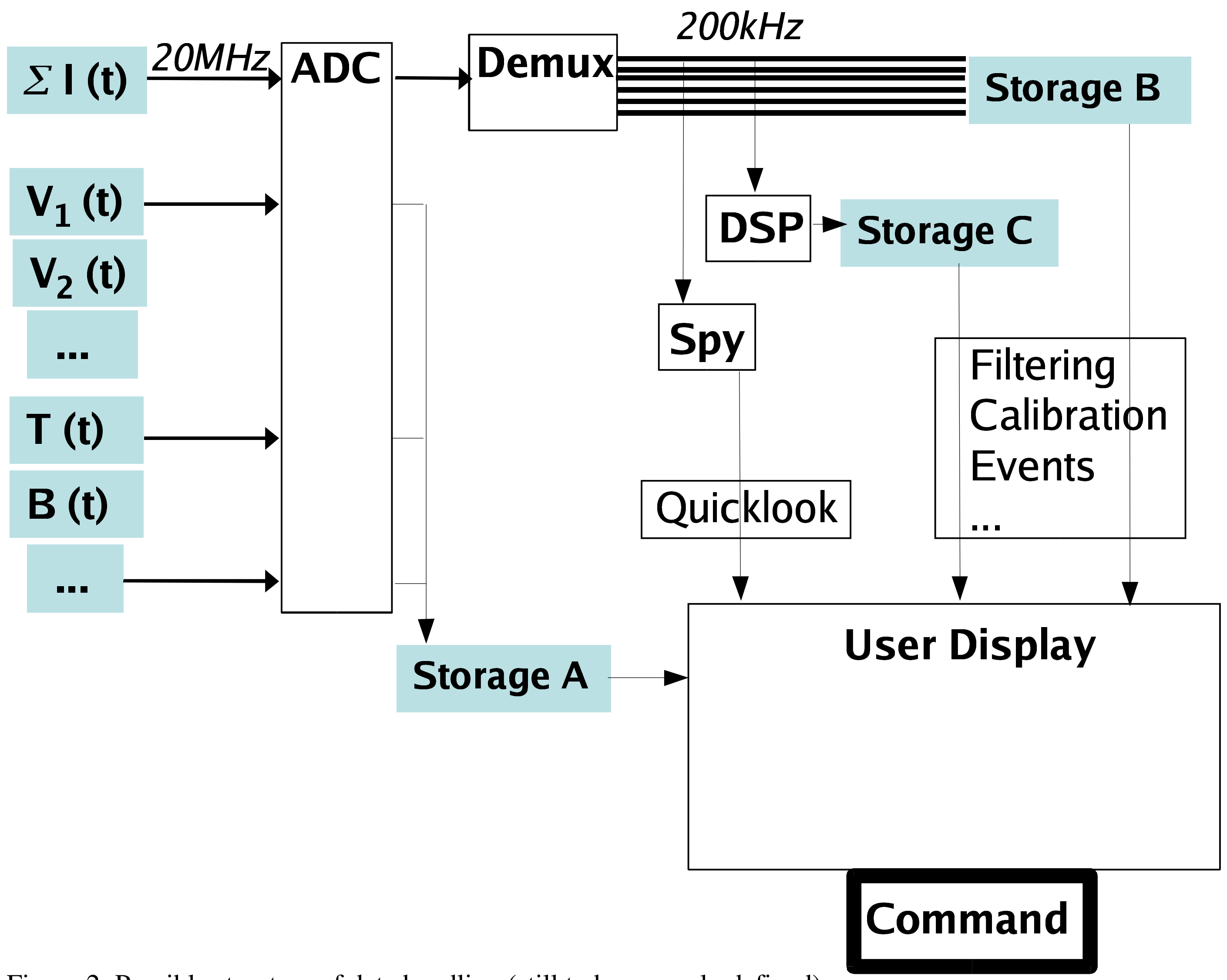

Figure 2: Possible structure of data handling (still to be properly defined).

4) Software requirements.

- Find commonalities between routines in order to recycle as much code as possible. In terms of function calling, parameters or structures to be passed across routines, global/local variables, etc., many operating modes may be thought of as simple V(t) measurements.
- Raw data format is CCSDS; we must think of the appropriate point to switch to FITS format (demux'ed data before or after optimum filtering?).
- Language: IDL, other or mixture? For now we think that IDL (or equivalent) should be the general language although it may call C subroutines for low-level functionality. Both MSSL and ISDC to think about this and about division into user interface, display functions, processing functions, etc. Groups running IDL are: MSSL, JAXA, PSI, IFCA, SRON. If ISDC s/w is used, then IDL may be impractical (ISDC s/w is Fortran and C++). Using compiled languages (C/C++) with some scripting languages (UNIX shell, Perl, Python) should be considered.
- FITS and pre-FITS files must have a standard header and history. Maite to look into this.
- Platform: everybody agrees on Linux.
- CVS shall control the master s/w repository, including standard datasets for s/w testing. To this end, a generic account will be created at IFCA for all users but each user will work in his/her local machine for security reasons. Every institute will have their own space in the repository.

5) **Open questions and actions.**

- We must first define the top-level s/w requirements and corresponding tasks. Two documents are expected: a re-vamp of the current generic requirements text and a detailed test-by-test s/w description.
- Once the tasks themselves have been identified and placed on a timeline, we shall proceed to their distribution among group members/institutions.
- We still need to identify which functions belong to the EGSE and which to the C&C or to off-line analysis. We do not have a clear picture of the EGSE/Instrument layout.
- Where exactly do the housekeeping data and the data from external units join the rest of the data if at all?
- The protocol used for CCSDS frame sending/receiving must be defined.

- Summary of actions (some already mentioned in the text):

• Bussons: lead production of Software Requirements and Test-by-Test Functional Breakdown documents; find out if there will be an IDL license at BESSY.
• Paltani: summary of stuff we can recycle from INTEGRAL.
• Paltani, Page: division into user interface, offline vs interactive vs automatic shells, modularity of display functions, processing functions, etc. Pros and cons of various languages (IDL, alternatives) and environments.
• Aliko, Jan-Willem: come up with a clear physical device layout (EGSE, Command & Control, FPGA, Demux, DSP, communications between them)
• Maite: list of items for file headers and file history
• Somebody: find out exactly when and how we must do the data reduction
• ALL: meet in late January or February to finalise documents and distribute tasks.