The On Board Software of the Herschel HIFI Instrument

L. Piazzo[1], A. M. Di Giorgio[2], F. Nuzzolo[1], P. Cerulli-Irelli[2],
L. Dubbeldam[3], D. Ikhenaode[1], S. J. Liu[2], S. Molinari[2] and S. Pezzuto[2]
(1) DIET Dept. - University of Rome "La Sapienza" - Italy
(2) IFSI Inst. - Istituto Nazionale di Astrofisica - Italy
(3) SRON - Netherlands Institute for Space Research - The Netherlands
Corresponding author: L. Piazzo - lorenzo.piazzo@uniroma1.it

***Abstract*—The Heterodyne Instrument for the Far-Infrared onboard the ESA Herschel satellite is a heterodyne spectrometer with unprecedented frequency range, resolution and sensibility. It is composed of two spectrometers, a Local Oscillator Unit, producing the mixing signals, a Focal Plane Unit, perfoming mixing and amplification, and an Instrument Control Unit, hosting a DSP where the On Board Software (OBS) runs. The OBS coordinates the other units, manages the interface with the spacecraft and monitors the instrument status. A synthetic description of the OBS organization and functioning is presented in this paper.**

Keywords- space technology; satellites.

I. INTRODUCTION

The Heterodyne Instrument for the Far-Infrared (HIFI) [1] is one of the instruments onboard the ESA Herschel Space Observatory (HSO) [2], which was launched in May 2009 and is now fully operational. It is a heterodyne spectrometer where the signal from the telescope is mixed with a local reference before being fed to the spectrometers. In this way the wide frequency range from 480 to 1910 GHz can be covered with unprecedented resolution and sensibility. The instrument was developed by a team of engineers, scientists and managers from 12 European and North American countries, founded by their national space agencies.

The instrument is equipped with two spectrometers, namely the Wide Band Spectrometer (WBS) and the High Resolution Spectrometer (HRS). The WBS is an acousto-optical spectrometer covering up to 7 bands within the frequency range with a resolution of 1 MHz, each band being 4 GHz wide. In each band, the HRS, which is a correlator spectrometer, can cover up to 8 sub-bands of 235 MHz with a resolution up to 125 KHz.

A block diagram of the instrument is shown in Figure 1. In addition to the spectrometers, which are duplicated in the Vertical and Horizontal polarizations, there are the Local Oscillator Unit (LOU), the Focal Plane Unit (FPU) and the Instrument Control Unit (ICU). The LOU generates the mixing signals for the 7 bands, deriving them from a reference frequency source. In the FPU the telescope signal is mixed, amplified and fed to the spectrometers. The FPU also hosts a chopper mechanism and a calibration source. The ICU generates the control and timing signals for the other subsystems and realizes the interface between the instrument and the spaceraft's Command and Data Managment Subsystem (CDMS).

The core of the ICU is a DSP running the On Board Software (OBS). The OBS was developed in cooperation by the IFSI Institute of the INAF (Istituto Nazionale di Astrofisica) and by the DIET Department of the University of Rome. It was funded by the Italian Space Agency (ASI). The Netherlands Institute for Space Research (SRON) contributed to the specification activity. The aim of this paper is that of giving a synthetic description of the OBS organisation and functionalities, highlighting the main design issues and implementation choices.

The paper is organised as follows. After a short introduction to the ICU in Section II, the Operating System and its main services are introduced in Section III, while the OBS is described in some detail in Section IV. Finally, Section V considers the development and testing system.

II. THE ICU

The ICU was developed by Carlo Gavazzi Space S.p.A. [3] and the high level block diagram is shown in Figure 2. It is composed by two boards, a CPU board, hosting a DSP and three memory banks (RAM, EEPROM, PROM), and a Payload Interface (PL/IF) board, hosting a 1553 chip, the Low Speed interface Electronic (LSE) and the High Speed interface Electronic (HSE).

The CPU board is based on the Analog Devices 21020 DSP [4], [5]. The CPU is clocked at 20MHz, with 50ns

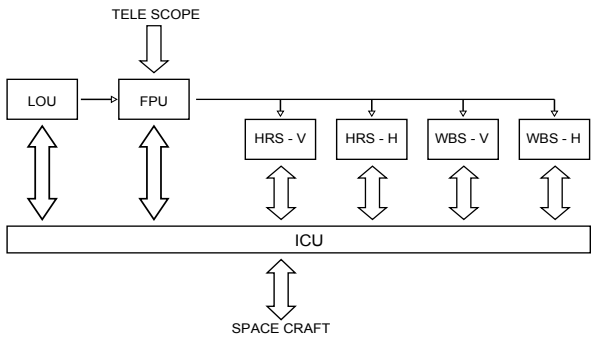

Figure 1. Block diagram of the HIFI instrument.

 ISBN: 978-1-61208-128-1

instruction cycle time and has a processing power of 20 MIPS with 66MFlops (peak) of instruction rate. The board includes a 2 MB data RAM (512k words of 32 bits), a 3 MB program RAM (512k words of 48 bits), a 1MB EEPROM memory (256k words of 32 bits) and a 32 KB PROM. The EEPROM is used to store the OBS whilst the PROM is used to store the Boot Software (BS), which is executed at the switch on. The BS simply loads the OBS into the program RAM, normally by copying it from the EEPROM, and starts its execution.

A. CDMS interface

The interface with the CDMS is based on a protocol structured into three layers. The highest one is the application layer, which is based on the ESA Packet Utilisation Standard (PUS) [6]. This layer defines the structure of the Telemetry (TM) and Telecommand (TC) data packets, which are used to transport the information across the interface.

The two lower layers are the physical and data link layers, which are based on the MIL-STD-1553B standard [8], [7], tailored for the specific requirements of the ESA Herschel mission. These layers regulate the exchange of the TC and TM packets. Communication takes place (at 400KHz with words of 16 bit) in regularly paced time slots and is mastered by the CDMS, which allocates the time slots (64 per second), signals the beginning of a slot by rising the 1553 interrupt (1553-INT) and possibly writes a TC in the 1553 chip memory. In response to the interrupt the OBS can upload the TC or write into the 1553 chip memory a TM packet that will be read by the CDMS at the end of the time slot.

B. Low speed interface

The LSE is the low rate (312.5KHz) interface with the subsystems and is used to transfer control and status data. The LSE has three registers, the transmit register (TXREG), the receive register (RXREG) and the Control Register (CREG), which are mapped into the DSP memory. The CREG is used to control the LSE while the other two are used for communication with the subsystems. Specifically, when the LSE is ready, the OBS can send a Low Speed Command (LSC) to a subsystem by writing it into the TXREG. The LSE switches to the not ready status, decodes the command, dispatch it to the appropriate subsystems and, after a time that varies depending on the LSC and can be as high as 3 msec, returns to the ready status.

A special type of LSC, which can be sent to the FPU and LOU only, is the Housekeeping Request (HKR), which is a LSC requiring the destination unit to report information about its status. In response, the unit writes the requested information into the RXREG where it can be directly read by the OBS. Examples of LSC are reported in Table I.

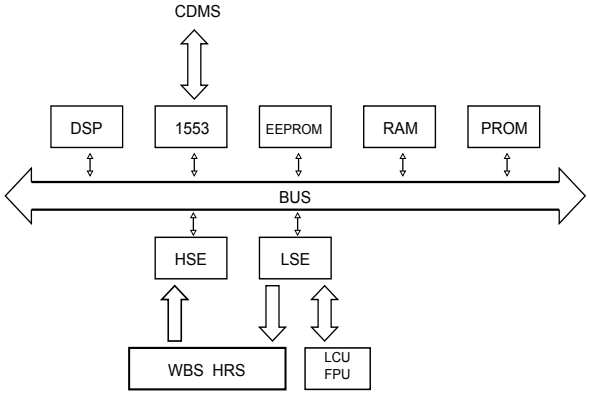

Figure 2. Block diagram of the ICU.

C. High speed interface

The HSE is the high rate (2.5MHz) interface with the spectrometers and is used to gather science and status data. The HSE has four hardware FIFO, of 16K words of 24 bits, together with a control register, which is mapped into the DSP memory and is used to control the HSE. The DSP can request data to a spectrometer by sending a specific, data transfer LSC. When the LSC is received, the spectrometer forms a science data frame, constituted by spectroscopy data and housekeeping data, and writes the frame into the FIFO. The WBS frame is 8210 words long and takes approximately 900 msec to be written while the HRS frame is 4160 words and takes approximately 42 msec.

The FIFO are accessible to the OBS because the first word of each FIFO is mapped into the DSP memory and can be read. Furthermore, as soon as the word is read, all the other words are shifted downwards so that, by repeatedly reading the first word, the OBS can extract the whole frame. The OBS must be careful not to read when the FIFO is void, to avoid garbage data, and not to request a data transfer while a data transfer is already ongoing, to avoid data misalignment.

Finally, note that the HSE rises the FIFO Half Full interrupt (FHF-INT) when one of the FIFO becomes half full. This interrupt can be used by the OBS to trigger the FIFO flushing.

III. Operating system and libraries

The OBS is written in C language (except for a few lines in DSP assembler) and runs under the Eonic Virtuoso real

Hex	Destination	Action
0xFC00 0003	WBS H/V	Start integration
0xFC00 0005	WBS H/V	Stop integration
0xE400 0006	WBS H	Data transfer
0xE800 0006	WBS V	Data transfer

Table I
Examples of LSC. Commands to start and stop the WBS integration and to require the transfer of the integration buffer.

 ISBN: 978-1-61208-128-1

time operating system (RTOS) [4], [9]. A Virtuoso program is made of several **tasks** running in paralell under the control of the RTOS, which masters the access to the DSP time. The RTOS also provides a set of communication and data management services that can be accessed by the tasks by calling appropriate system functions. The main services exploited by the OBS are the following.

Queue: a software FIFO buffer storing packets of information. There are system functions to add and extract packets and to put a task in a wait state over a queue (i.e., the task is waked up as soon as a packet is put in the queue).

Event: a communication signal between tasks. There are system functions to rise an event and to put a task in a wait state over an event (i.e., the task is waked up when the event is raised).

Timer: a downcounter. There are functions to start a timer and to put a task in a wait state over a timer (i.e., the task is waked up when the countdown reaches zero).

The OBS source code is oganised into a core and several libraries. The main libraries are the 1553, the EEPROM and the Memory Managment libraries. The 1553 library is used to interface with the 1553 chip, providing functions to perform basic tasks, e.g., TC and TM packets upload and download. The EEPROM and the Memory Managment libraries provide functions to handle the EEPROM and the RAM, e.g., writing and reading EEPROM pages, memory dump and CRC check.

IV. The OBS

The OBS main task is the execution of the TC received from the CDMS. This involves accepting and decoding the TC, carrying out the requested operation, gathering and formatting the output into TM packets and passing the TM back to the CDMS. Currently the OBS accepts several tens of TCs, implementing a number functions like spectroscopy, calibration, instrument configuration, OBS configuration and update, memory and time management.

As a second task the OBS regularly monitors the instrument status, possibly taking actions if anomalies are detected, collects housekeeping (HK) data from the instruments, formats the data into TM packets and passes the TM to the CDMS.

The OBS was designed and is conveniently described as a set of logical modules, where a module is a collection of tasks, functions and data. A key role is played by the RTOS queues, which are the preferred communication method among the modules. For efficency reasons, the modules also communicate by direct memory access (i.e., the task running in a module reads or writes data of a second module). The main OBS modules, queues and data flows are depicted in Figure 3 and will be described in the rest of this Section.

A. CDMS interface and TM classification

The TM packets are classified into three types with different priority. Specifically, Event TM (EV-TM) packets,

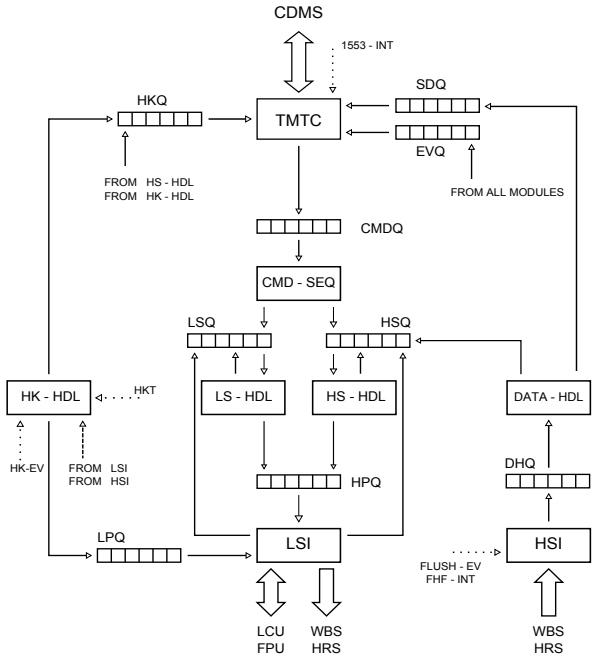

Figure 3. The OBS modules, queues and main data flows. Solid arrows represent queue packets. Dashed arrows represent direct memory access. Dotted arrows are events, interrupts and timers.

reporting errors, alarms and warnings, are high priority. Housekeeping TM (HK-TM) packets, reporting instrument status data, are medium priority and Science Data TM (SD-TM) packets, reporting spectrometers' readouts, are low priority.

The interface with the CDMS is managed by the TMTC module. The module has three input queues, namely the Event Queue (EVQ), used for EV-TM, the Housekeeping Queue (HKQ), used for HK-TM, and the Science Data Queue (SDQ), used for SD-TM. Any OBS module willing to send a TM packet simply needs to put the packet into one of these queues.

Processing is carried out by task $tmtc$, which is waked up when the 1553-INT interrupt is raised. Upon wake up $tmtc$ checks if there is a TC packet in the 1553 chip. If yes, the packet is extracted, and passed to the CMD-SEQ module's input queue. If not, $tmtc$ orderly checks the EVQ, the HKQ and the SDQ and if a pending packet is found, it is uploaded to the 1553 chip. Eventually the task goes back to a wait state.

B. Low speed interface

The low speed interface with the instruments is managed by the LSI module. The module has two input queues, the High Priority (HPQ) and the Low Priority (LPQ) where any OBS module can put a packet with a LSC to send. When the LSC is an HKR, the packet also contains the memory address where the LSI module shall write the HK data. An

 ISBN: 978-1-61208-128-1

additional type of packet that can be put in the queues is the Feedback Request (FBR), which forces the LSI module to produce feedback to some other module of the OBS and is useful to synchronise the modules.

Processing is carried out by task ls, which is in a wait state on the input queues. As soon as a packet is put in one of the queues the task wakes up and processes all the packets in the queues, starting from the HPQ. For a LSC the processing simply involves writing the command to the TXREG. For HKR it also includes the readout of the RXREG and the copying of the HK data in the memory addres specified by the packet. A FBR tipically involves putting a packet into the input queue of another module or rising an event. When the queues are void, the task goes back to a wait state.

C. High speed interface and science data processing

The high speed interface with the spectrometers is managed by the HSI module, hosting tasks $hs0$, $hs1$ and an input queue for $hs1$. Science data frames are extracted from the FIFO in blocks shorter than half a FIFO. The low level reading is carried out by task $hs0$, which is in a wait state over the FHF-INT. When the interrupt is raised, $hs0$ extracts a block from the FIFO, concatenates it with the previously extracted blocks and checks whether a whole science data frame has been extracted. If yes, $hs0$ put a packet with the frame into the input queue of $hs1$.

Task $hs1$ is in a wait state over its iput queue and is waked up by the packet. This task performs initial processing of the science data. It extracts the HK information and copies it into the the HK-HDL module by direct memory access. If required, it also performs the coaddition of the frame with previously received frames. Then it checks if a fully coadded science frame has been formed. If yes, it put a packet with the frame into the input queue of module DATA-HDL.

The procedure just described is simple, prevents FIFO overflow (because the FIFO are automatically emptied when they are half full) and guarantees that no garbage data is read (because a block is less than half FIFO). However it always leaves a block in the FIFO (because a block is too short to trigger the FHF-INT) while there are times when the OBS needs to completely flush the FIFO. For this reason, the HSI module also hosts task hs_flush, which, upon wake up, checks the FIFO, extracts the pending blocks and passes the frames to $hs1$. That task is in a wait state over the FLUSH-EV event and can be activated by any module of the OBS by rising that event.

Finally, the DATA-HDL module completes the processing of the science data, by means of task $data\_hdl$, which is in a wait state over its input queue, the Data Handler queue (DHQ). The processing varies[1]and may include computing the total power in given bands, spectrum scaling to adjust the dynamic range, breaking the frame into a several SD-TM packets and writing the packets into the SDQ.

D. Housekeeping

HK production is carried out by task hk_hdl of the HK-HDL module. This task is in a wait state over the Housekeeping Timer (HKT). The countdown of the HKT is started the first time when a specific TC, switching on HK production, is received by the OBS. When the counter reaches zero, task hs_hdl is waked up and its first action is to start again the HKT countdown with a given, programmable initial value. In this way the task is activated on a regular basis and the activation rate is controlled by the initial value loaded in the HKT. Typical rates are from 1 to 0.25 Hz.

Task hk_hdl has to produce an HK-TM packet each time it is activated. Concerning the HK data from the spectrometers, these are written by the LSI module into the HK-HDL module every time that a science data frame is received. Therefore, provided that frames are regularly produced, task hk_hdl finds the HK in its memory and does not need to explicitely acquire them. However the task has to guarantee a regular frame production. Therefore the task's second step is to check if the spectrometers are being used by some other OBS module[2]. If yes, the frames will be produced by the other module's transfer requests. If not, hk_hdl issues a data transfer request, by placing the corresponfing packet into the LPQ.

The third step of the task is to send a sequence of HKR to the FPU and the LOU in order to acquire their HK. This is done by placing the corresponding packets into the LPQ. The sequence is closed by a FBR packet, requiring LS to rise the event HK-EV. Next the task goes into a wait state over the HK-EV. When LS has processed all the HKR and rises the HK-EV, the task is waked up and all the HK data are there in its memory. The task can format a HK-TM packet, put the packet in the HKQ and go back into a wait state over the HKT.

E. TC classification

There are several issues concerning TC execution. Firstly note that the TCs are received with a maximum rate of four per second, are executed in the order in which they are received and, in normal operation, they are paced so that a new TC is received only when the previous TC was completed. However the OBS needs to be robust against the case where a new command is received while an old one is still running, which can happen by mistake or because the CDMS wants to stop the running command. In such a case, normally, the old command needs to be aborted before the new one is started, even though some commands can be executed in parallel, as we see later.

Secondly, when the TC requires the use of the HSI, before starting the TC execution the OBS needs to check whether

[1]The OBS maintains a global variable, $activity\_id$, indicating the current activity, which is used by DATA-HDL to decide the appropriate processing.

[2]By checking $activity\_id$.

 ISBN: 978-1-61208-128-1

the HSI is busy or not. In fact the HSI may be transferring data requested by the HK-HDL module or by an older TC. If the HSI is busy, the OBS has to guarantee a safe transition of the HSI to the new command, in order to avoid misalignment or garbage data.

Finally, while most commands have time requirements on the order of the msec, spectrometry commands have time requirements on the order of the usec, which are difficult to gurantee even within the Virtuoso RTOS.

In order to deal with these issues, the TCs are divided by the OBS into four classes, having different requirements. The classes are reported in the following list and will be discussed in the next subsections.

Short TC (S-TC): execution time less than 250 msec. Does not access the spectrometers. Example: OBS configuration, HK handling (switch on, switch off, rate setting).

Low Speed TC (LS-TC): execution time more than 250 msec. Does not access the spectrometers. Example: memory dumping, FPU and LCU configuration and monitoring.

High Speed TC (HS-TC): execution time more than 250 msec. Access the spectrometers with loose time requirements (msec). Example: WBS and HRS calibration and monitoring.

Reat Time TC (RT-TC): execution time more than 250 msec. Access the spectrometers with strict time requirements (usec). Example: spectrometry.

F. Short TC and TC front-end

The front-end of the command execution is module CMD-SEQ, hosting task cmd_seq. This task is in a wait state on its input queue, the Command queue (CMDQ). As soon as TMTC put a TC packet in the queue the task wakes up and performs basic integrity checks on the TC (e.g., checksum, length). If these are passed, it decodes the command and checks its type. If the TC is a short one it is immediately executed, within the cmd_seq task, possibly in parallel with another, non short TC already running. This is possible because short commands are completed before the next TC is received and do not access the spectrometers.

If the TC is not a short one, task cmd_seq updates the data on the currently running command, possibly aborting a previously running TC, and forwards the TC either to the LS-HDL or to the HS-HDL module by copying it into a packet and putting the packet into the corresponding input queue.

G. Low speed TC

LS-TC are dispatched to and executed by the LS-HDL module, hosting task ls_hdl, which is in a wait state over its input queue, the Low Speed queue (LSQ). In order to simplify the command aborting and increase the code structuring, every LS-TC is broken down into a sequence of steps. Every step is triggered when a specific packet is extracted from the queue, the first step being triggered by the LS-TC itself. Furthermore, every step ensures that a packet triggering the successive step is put in the LSQ, so that the whole command is executed.

To clarify the execution, let us discuss the processing of a simple LS-TC requiring a set of HK data from the LOU. The first step copies the TC parameters (i.e., the set of HK data to be required) into the internal memory of the LS-HDL module and put into the LSQ the packet triggering the second step. The second step put a sequence of HKR into the HPQ in order to gather the required HK data and closes the sequence with a FBR, requiring LS to put into the LSQ the packet triggering the third step. When the third step is triggered, the LS module has already processed all the HKR of the sequence. Therefore the third step can format the HK data into a HK-TM packet and put the packet into the HKQ.

H. High speed TC

HS-TC are dispatched to and executed by the HS-HDL module, hosting task hs_hdl. This task is in a wait state over its input queue, the High Speed queue (HSQ). Also HS-TC are broken down into smaller steps and their execution is almost identical to the execution of a LS-TC. One difference is that the first step has to check whether the HSI is busy or not. If there are no pending frames in the HSI[3], it put into the HSQ the packet triggering the second step. If there are pending frames, the first step waits a programmable time[4] and posts the HS-TC back into the HSQ in order to be triggered again. A second difference is that the successive step is typically triggered by feedback from the DATA-HDL module and not from the LS one.

To clarify the the processing let us describe (a simplified version of) the WBS calibration. The first step is triggered when CMD-SEQ put the TC in the HSQ. When the HSI is free, the first step triggers the second step by placing the corrsponding packet into the HSQ. The second step issues a sequence of LSC commands to LS in order to perform a WBS integration followed by a data transfer request. The science data frame is extracted by the HSI module and passed to the DATA-HDL module, which computes the frame total power, breaks the frame into SD-TM packets, put the packets into the SDQ and eventually put a feedback packet into the HSQ triggering the third calibration step. The third step exploits the total power to set the WBS attenuators, formats an HK-TM packet with a calibration report and put the packet into the HKQ.

I. Real Time TC and the Virtual Machine

Due to the strict time requirements on the RT-TC, these TCs cannot be executed using the RTOS services, which

[3]The OBS maintains four counters indicating how many frames are pending within each FIFO. These counters are increased when a data transfer is requested by any module and decreased when the HSI extracts a frame.

[4]The task also issues a FLUSH-EV in order to flush the FIFO and updates $activity\_id$ in order to stop any data transfer request to the spectrometers.

 ISBN: 978-1-61208-128-1

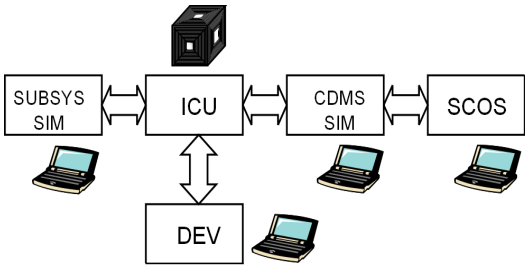

Figure 4. The development and testing system.

have a timing jitter on the order of the msec. In order to execute these commands, the OBS implements a dedicated command interpreter, termed the Virtual Machine (VM). The VM is called by the Interrupt Service Routine (ISR) of the TIME-INT interrupt, which is the DSP highest priority interrupt, raised when a hardware timer, based on the high resolution clock, reaches zero. At every call, the VM executes one or more steps of a program written in VM assembly language and restarts the timer with a proper inital value, in order to trigger its next execution round. The assembly includes basic programming instructions (e.g., assignment, if, $goto$). Furthermore the VM can send LSC to the spectrometers by directly accessing the TXREG. In this way the VM can excute all the spectroscopy commands. There are different VM programs for the different commands, and these are handled by the OBS, which stores the programs as binary vectors and can upload them into the VM memory for execution. Furthermore the OBS can switch on or off the VM by enabling or disabling the TIME-INT.

RT-TC are dispatched to the HS-HDL module and are handled like HS-TC with a single step. The step is similar to the first step of an HS-TC. It checks the HSI status and put the RT-TC back in the HSQ until the HSI is flushed. When the HSI is found ready, the step loads the appropriate VM program and start the VM execution by enabling the TIME-INT. From this point on the TC is executed by the VM, which typically issues a long sequence of integration and data transfer requests. The only modules of the OBS involved are the HSI, which extracts the data frames and put them into the DHQ, DATA-HDL, which breaks the frames into SD-TM packets and put the packets into the SDQ, and TMTC, which passes the TM to the CDMS.

V. Development and testing

Tha OBS development was carried out in parallel by several developers, exploiting the Concurrent Versioning System (CVS) [10]. It started in year 2000 and is now in the maintenance phase.

The development and testing system is shown in Figure 4. The core of the system is the electric model of the ICU, which is a prototype of the actual ICU, featuring all the characteristics of the final device. A second piece of hardware is the subsystem simulator (SUBSIS-SIM), which is a board mounted on a PC, capable of emulating the signals and timing of the other HIFI units (LOU, FPU, WBS and HRS) at the electric level. These two components allow to test the interface between the ICU and the instrument. A second PC (CDMS-SIM) was equipped with an off the shelf 1553 board and a software simulator of the CDMS, allowing to emulate the interface with spacecraft. The CDMS-SIM was connected via a LAN to a third PC (SCOS) running the Satellite Control and Operation System (SCOS2000) application, which is the standard ESA mission control software [11]. In this way the whole chain, from the mission control facility to the instrument, could be emulated. An additional PC (DEV) hosted the DSP development kit, which included a JTAG probe directly connected to the DSP, allowing to place breakpoints to pause the execution and inspect the DSP memory and registry.

A test procedure involved issuing a sequence of TCs from the SCOS application and gathering the resulting TM. Since the output produced by the subsystem simulator (both the science and the HK data) can be set to given values, the resulting TM can be checked against the expected TM, thereby allowing to prove the correct functioning of the OBS. A number of test procedures, covering a wide range of functionalities, were included in a test plan and the test plan was carried out at each software update, in order to validate the OBS. Naturally additional tests were carried out at the SRON and ESA premises using the real hardware.

References

[1] T. de Graauw et al., ”The Herschel-Heterodyne Instrument for the Far-Infrared (HIFI)”, Astronomy and Astrophysics, Vol. 518, L6-12, July 2010.

[2] G. Pilbratt et al., ”Herschel Space Observatory”, Astronomy and Astrophysics, Vol. 518, L1-5, July 2010.

[3] www.cgspace.it/index.php?option=com_content &task=view&id=178 (last accessed 1/28/2011)

[4] T. K. Pike et al. ”OMI building blocks and developments for future high performance processing systems in the aerospace and automotive fields: DSP, SMCS and VIRTUOSO”, Proc. of the IEEE Conf. on Electronics, Circuits, and Systems (ICECS), Vol 1, pp. 610-613, 1996.

[5] www.analog.com/en/embedded-processing-dsp/adsp-21xx/processors/index.html (last accessed 1/28/2011)

[6] ESA Ground systems and operations - Telemetry and Telecommand Packet Utilization- ECSS-E-70-41A, January 2003.

[7] MIL-Std.-1553 B, Digital Internal Time Division Command/Response Multiplex Data Bus, Issue Notice 2, 8 September 1986

[8] E. H. Philips, ”The Electric Jet.”, Aviation Week and Space Technology, Vol 5. No. 2, Feb. 2007.

[9] S. Hustin, M. Potkanjak, E. Verhulst and W. Wolf, ”Real-Time Operating Systems for Embedded Computing”, Proc. of the 1998 IEEE/ACM international conference on Computer-aided design, pp. 388-392, 1998.

[10] www.tortoisecvs.org/ (last accessed 1/28/2011)

[11] www.egos.esa.int/portal/egos-web/products/MCS/SCOS2000/ (last accessed 1/28/2011)

 ISBN: 978-1-61208-128-1