Herschel/PACS On-board Reduction/Compression Software Implementation

Roland Ottensamer[a], A. N. Belbachir[b], H. Bischof[c], H. Feuchtgruber[d], F. Kerschbaum[a], A. Poglitsch[d], C. Reimers[a]

[a]Institute for Astronomy at the University of Vienna
Türkenschanzstraße 17, 1180 Vienna, Austria;
[b]TU Vienna, Computer Aided Automation, PRIP Group
Favoritenstraße 9/183-2, 1040 Vienna, Austria;
[c]TU Graz, Institute for Computer Graphics and Vision
Inffeldgasse 16/II, 8010 Graz, Austria;
[d]Max Planck Institute for Extraterrestric Physics
Postfach 1603, 85740 Garching, Germany

ABSTRACT

This paper describes the design and implementation of the on-board data compression and reduction software for the HERSCHEL[1]/PACS[2] mission (see also A. Poglitsch et al. in this conference) of the European Space Agency (ESA). Lead by the Max Planck Institute for Extraterrestric Physics (MPE) in Garching, Austrian scientists and software engineers participate in the development of the on-board software[3] for the Photodetector Array Camera and Spectrometer (PACS). The novel detectors' high data rates in addition to the distant spacecraft orbit force us to carry out irreversible reduction steps that are normally done on ground and to use highly specialized compression algorithms for lossless compression of the reduced science and the header data.

Keywords: Herschel, PACS, On-board, Software, Reduction, Compression, Infrared

1. INTRODUCTION

ESA's far-infrared and sub-mm cornerstone mission **Herschel Space Observatory (HSO)** is scheduled for launch in 2007. It will be equipped with a 3.5m Cassegrain telescope and house three instruments inside its superfluid Helium cryostat covering the spectral range between 55 and 670μm. The three instruments are built by different European consortia with international cooperation.

Table 1. The scientific payload of the Herschel Space Observatory.

Instrument	PI location	spectral range
Photodetector Array Camera and Spectrometer (PACS)	MPE Garching, GER	55-210 μm
Heterodyne Instrument for the Far Infrared (HIFI)	SRON Groningen, NL	480-1910 GHz
Spectral Photometer Imaging Receiver (SPIRE)	Univ. of Wales/Cardiff, GB	200-670 μm

PACS will conduct dual band photometry and imaging spectroscopy in the 55-210 micron spectral range. The instrument consists of two 25x18 Ge:Ga photoconductor arrays for spectroscopy, read out at 256 Hz and two bolometer arrays with 32x16 and 64x32 pixels for photometry, read out at a frequency of 40 Hz. In both modes, a high raw data flow of up to 4 Mbit/s is generated. This is far above the telemetry downlink bandwidth,

Further author information:
R.O.: E-mail: ottensamer@astro.univie.ac.at
A.N.B.: E-mail: nabil@prip.tuwien.ac.at

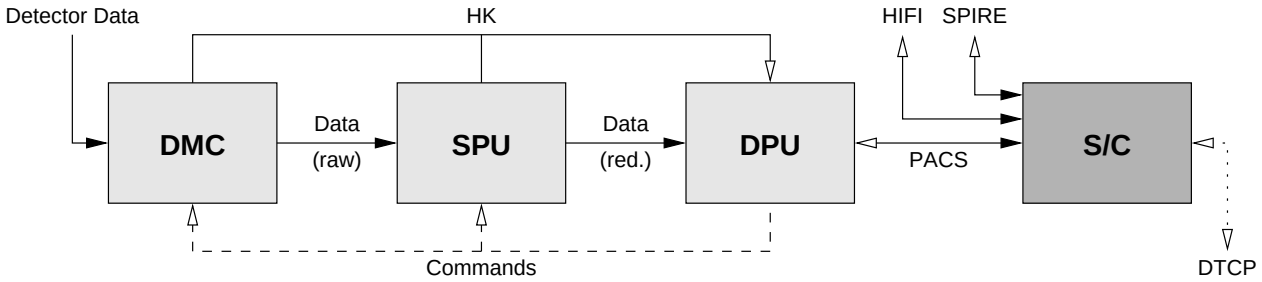

Figure 1. Data flow inside PACS warm electronics. The three instruments are connected to the spacecraft (S/C) mass storage. Inside PACS, the detector arrays are read by the Detector and Mechanism Controller (DMC) built by Centre Spatial de Liège and sent to the SPU, where all reduction and compression is done. After that, the data are sent to the Data Processing Unit (DPU), which has an interface to the spacecraft. The DPU hardware is built by Gavazzi under Istituto di Fisica dello Spazio Interplanetario (IFSI) contract and the DPU Software is also contributed by IFSI. Aside from the nominal science data flow, each subunit is obliged to send Housekeeping (HK) – a set of diagnostic counters and values – at time intervals of 2 seconds.

which is normally restricted to 120 kbit/s due to the L2-orbit of the spacecraft in about four times the Moon's distance.

Taking the needed signal to noise ratio into account, the required compression factor of typically 40 cannot be achieved with lossless compression techniques only.[4] That is why on-board data reduction steps, that are normally done on ground, are needed.[5]

Processing of such an amount of data requires dedicated hardware on board. The subunit which is built for this purpose is the Signal Processing Unit (SPU). Figure 1 illustrates the warm electronics subunits that are involved in the dataflow. PACS has four independent SPU boards built by CRISA under contract by the Instituto de Astrofísica de Canarias (IAC). Our Spanish colleagues also provide the startup software and low-level drivers. Two boards are always operational – one for the long wavelength arrays and one for the short wavelength arrays – while the other boards serve as backup units. Each SPU board has a copy of the **High Level Software (HLSW)** loaded into EEPROM. Different versions of the software can also be uploaded during the Daily Telecommunication Period (DTCP). This ensures that the mission-critical HLSW can be updated during operations.

2. REQUIREMENTS AND DESIGN

The main function of the HLSW is to provide data reduction and compression for all foreseen operating modes of the instrument. This also implies for example, that the communication with other subunits is handled too. The main functions of the HLSW are:

- execution of commands from the spacecraft Data Processing Unit (DPU)
- data acquisition from the Detector and Mechanism Controller (DMC)
- science data reduction and/or compression
- lossless compression of the DMC data headers
- transmission of the processed data to the DPU
- implementation of servicing, diagnostic and test modes
- remote access to RAM and EEPROM

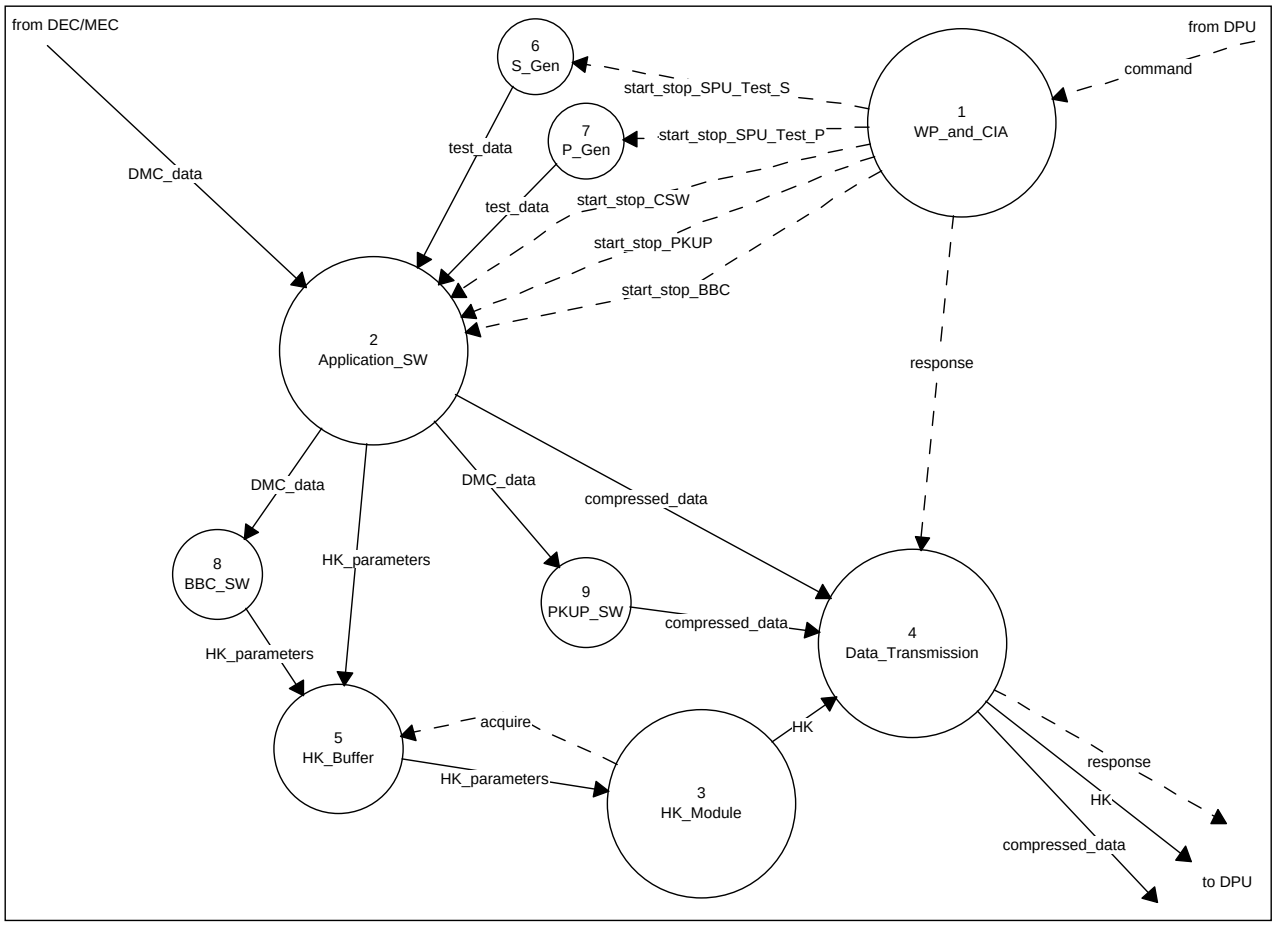

Figure 2. SPU HLSW architectural design at top level. The main task is the Watch Process (WP) that signals start and stop to all other threads. The WP automatically interrupts any ongoing task if a command from the DPU is received.

A variety of sequences, modes and algorithms, commandable and/or adaptable to the data input and uploadable sets of parameters and tables ensure a maximum of flexibility with a minimum of user interaction.

The SPU HLSW is developed according to ESA engineering software standards using the Yourdon-DeMarco Structured Analysis method with the Hatley-Pirbhai real-time extensions.[6] During the design phase, data from the Infrared Space Observatory (ISO) have been used to evaluate the concept.[7] The required HLSW functionality has been decomposed into three main tasks (see figure 2):

- The Watch Process (WP) listens to the DPU link. It interrupts the Application SW whenever a command is received for its acknowledgment. Basically, any running activity inside the SPU is interrupted whenever a command is received from the DPU.
- The **Application Software (ASW)** performs data reduction and compression according to the DMC header received within the raw data. Its responsibility is to achieve the required compression ratio according to telemetry requirements and to the compression mode.
- The Housekeeping (HK) module generates HK, which is sent to the DPU in 2 second intervals.

The actual task of the HLSW – *science data reduction and compression* – is achieved in the ASW module. Here, the various processing steps from header separation to ramp fitting are done. Figure 3 illustrates a nominal

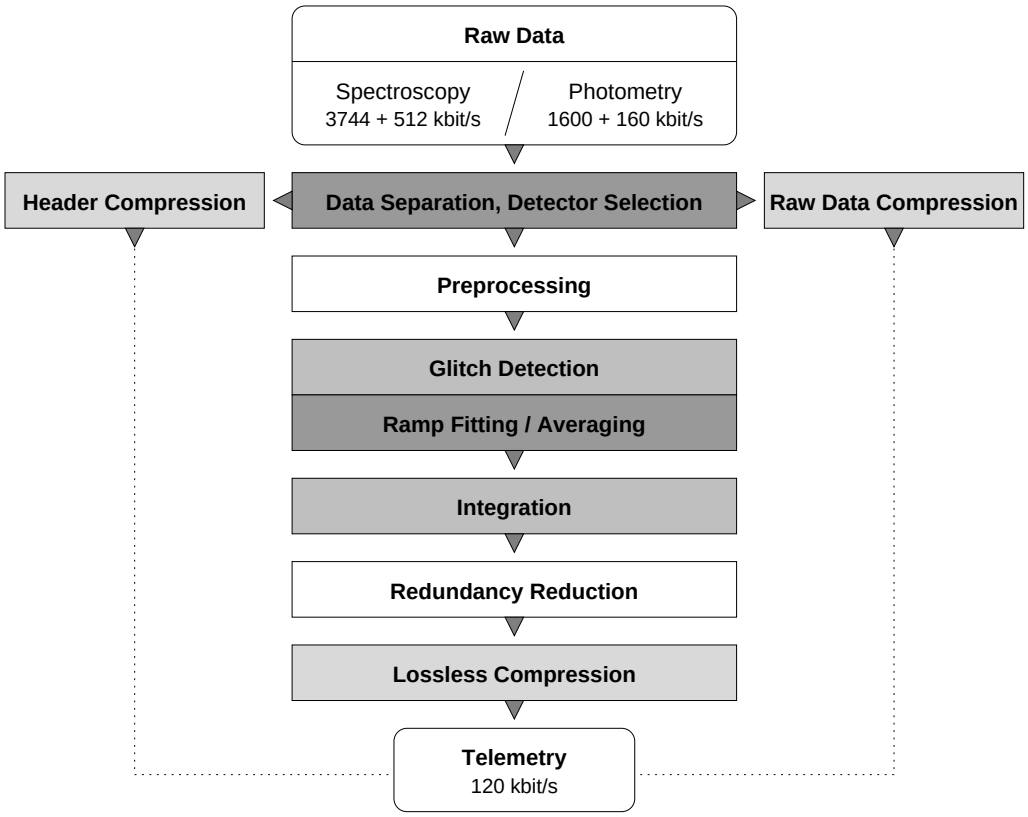

Figure 3. A typical reduction and compression sequence. Modules where raw data are irreversibly reduced are filled with dark grey colour.

compression/reduction sequence as it would be executed in a *default* mode. As already stated before, the full nominal sequence is not the only mode of operation. All processing steps are optional – it's even possible not to do anything at all with the data, though that case is most unfavorable (see section 1). There is also not just *one* lossless compression algorithm applied to all the data before they are sent. For example, the reduced science data in spectroscopy may be encoded with an implementation of arithmetic coding, whereas in photometry this is hardly possible due to the required CPU power. The central algorithm that is used for compression of the science frame headers is outlined in appendix A.

3. IMPLEMENTATION

Each SPU board has a Temic 21020 DSP running at 18 MHz, 7 MB of RAM and one SMCS chip for the interfaces. These restraints had already been taken into account during the early design phase. The HLSW modules are with only one exception written in ANSI C. This exception is the interrupt service routine that had to be coded in assembler.

The current implementation (as of May 2004) of the SPU HLSW consists of more than 11000 lines of code, but compiles to an executable of less than 250 kB including the Virtuoso^{TM}operating system. For proper operation, most of the 3 MB of PRAM (see figure 4) are required though, because of the many processing tables.

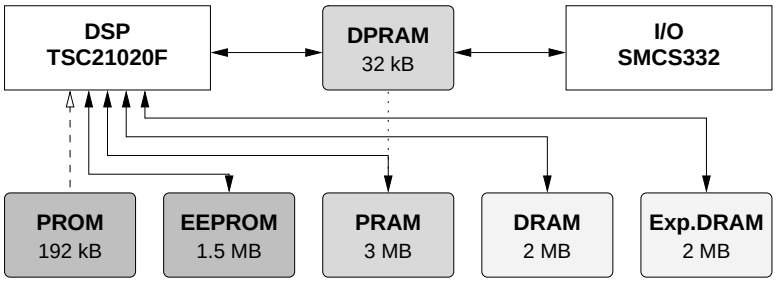

Figure 4. A SPU board has a total of 3 MB of PRAM where the HLSW is located and 4 MB of DRAM for data processing. Depending on the current operation mode, it is only possible to cache a few seconds of input data.

3.1. Drivers and Operating System

To handle the task management – especially with all the simultaneously operating high speed communication links – the HLSW runs under the Virtuoso^{TM}real-time operating system from WindRiver (previously developed by EONIC). Virtuoso^{TM}is available for several processors, most of them are DSPs. It is linked with the executable during compilation. The drivers to access the SMCS and other hardware are provided by the hardware manufacturer.

3.2. Communication

One SPU communicates on three links: raw data are permanently received from DMC, compressed data are sent to the DPU and commands can also be received any time from the DPU. The SPU uses a Scalable Multichannel Communications Subsystem (SMCS) 332 chip which implements the *Spacewire* standard. It is nominally configured to handle up to 10 Mbit/s on each link. To meet the real-time requirements it was necessary to implement several input and output buffers as *circular buffers*.

3.3. Detector Selection

In many cases it may not be needed to use the full array for observation. In addition to that, the detector will degrade over time and individual pixels will become defunct. So it is most reasonable to apply a pixel selection mask on the detector arrays. Depending on the number of selected pixels, the raw data could even be transmitted without reduction.

Detector selection is handled by the SPU with uploadable tables. These tables have a unique identification number that allows to reconstruct the array geometry on ground. Detector selection tables are stored within the ground segment software – the Herschel Common Science System (HCSS). Figure 5 shows the tool which is used to create or modify detector selection tables.

3.4. Transitional Effects

Before any data can be fit it must be ensured that no transitional effects go into the reduction. This especially addresses external glitches in the detected signal caused by impacts of ionized particles on the detectors. Several algorithms for glitch detection have been tested and can be selected, but a combination of this step with the actual fit turned out to be most feasible considering the limited CPU power.

Transitions can also be caused by the instrument itself. The data must be synchronized with the chopper and other instrumental parameters and constantly monitor measures like the grating position, for instance. If an unexpected transition occurs, the data processed so far are sent and a new sequence is started without losing any science frames.

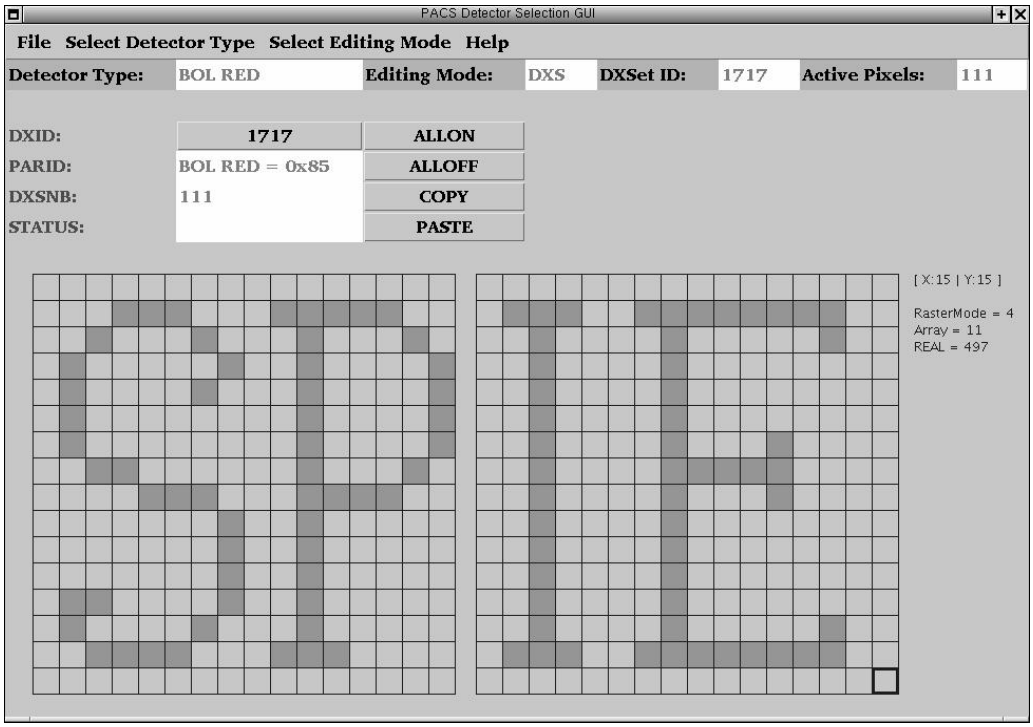

Figure 5. A screenshot of the Detector selection table graphical user interface within the HCSS. This tool is used to select active/inactive pixels for consideration in the data stream. In this example, a table with 111 selected pixels for the long wavelength bolometer array is generated for uplink through telecommand and storage in the local database.

3.5. Reduction and Integration

In the reduction module the raw data from the detectors are averaged or fit depending on the instrument operating and compression mode. In photometry, samples on a chopper plateau are averaged, whereas in spectroscopy a more dedicated method has to be used. A series of photoconductor samples within a reset interval is read out in a non-destructive way, so the resulting signal is a *ramp*. The ramp or parts of it are then fit with the least squares method. Several alternative algorithms can be commanded, one for instance splits a ramp into several *sub-ramps* that are averaged with their mean. Some compression modes require additional integration to satisfy the required downlink rate.

3.6. Lossless Compression

Three kinds of data must be compressed with lossless algorithms in the SPU. For this purpose, a handful of algorithms are implemented and applied accordingly to the data.

- **Science Frame Headers**
  Each raw data frame comes with a 64 Byte header that contains the instrument setup. This header is required on ground, but at a rate of 256 Hz in spectroscopy it adds up to 128 kbit/s for each of the two spectroscopic arrays, already exceeding the overall bandwidth! To get rid of the headers, a simple yet most effective method has been developed. First of all, the header frames are subtracted from each other. That way, constants are reduced to zeros and counters are reduced to constants. After that, all words containing zeros are removed by storing their original position in a binary mask. Alternatively, Run-length encoding[8] could be used, but it does not perform that well in this case. The remaining data can be easily packed with an entropy encoder e.g. *RZip* (see appendix A).

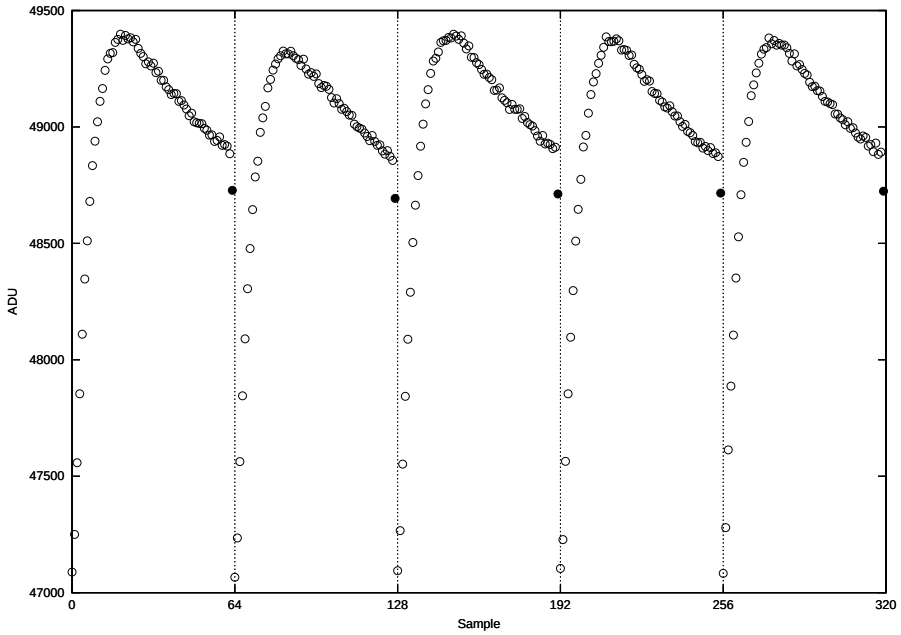

Figure 6. Some of the first real ramps, obtained during February 2003 in the MPE lab.

- **Reduced Science Data**
  The reduced data still have some properties we can take advantage of to encode the entropy. First of all, the data are sorted in a profitable way, taking reset intervals of the detector into account. The next step is to take advantage of redundancies by calculating dynamic differences. After that, the data are processed with *RZip* and finally encoded with a variant of arithmetic coding, though this demanding method is not available for all compression modes due to the restraints on processing power.

- **Additional Raw Channels**
  Depending on the selected observation and compression mode, the number of selected pixels and the observed target, the telemetry bandwidth allows for additional raw data from a few pixels. These raw data are used on ground to verify the on-board reduction. The method how the additional raw data are compressed does not differ too much from the way the reduced science is compressed.

3.7. Other Modes

The HLSW must also support the peak-up sequence of the satellite and other routines for recalibration. This includes the *Background Canceling* mode that is needed to recalibrate the temperature of the reference channels in the bolometer. In this mode the DMC and the SPU autonomously reconfigure the instrument setup. The HLSW has to perform these servicing tasks largely in an autonomous way, but at least these modes are not too demanding for CPU power.

4. TESTS, VERIFICATION AND CONCLUSION

Since the integration of the HLSW on the SPU hardware mid-2002 and the acceptance and delivery of the SW to the consortium in spring 2003, a whole lot of tests in all diversity have been conducted at MPE (see figure 6). We used the findings to improve the HLSW and to respond to the evolution of our detectors. This process of iteration is still not finished. The central points are dealing with the detector non-linearity and its response to glitches. With support from the PACS Instrument Control Center, new methods are still being investigated. Yet most advanced methods require too much CPU power, that is simply not available. Nevertheless, many modules

have proven of value, like our communication tasks. The header compression turned out to be most efficient, yielding lossless compression ratios of typically 40(!) and more, leaving many standard algorithms[9] that we tried far behind. We also experienced that the vast majority of standard lossless compression algorithms[8] are hardly useful for any other kind of our data. In addition to our efforts, studies have been made to evaluate transform coding methods for on-board compression and to apply reduction and compression concepts to infrared cameras of other space observatories.[10]

APPENDIX A. RZIP

RZip is a lossless compression algorithm that we developed for DMC header compression. It is not intended for compression of any other data, though it turned out to be useful in other contexts as well. The emphasis was on writing an algorithm that runs *fast* on the DSP and compresses the science frame headers as efficient as possible.

The strategy of *RZip* for searching redundancies in the input buffer is closely related to the data granularity. Our header data words are 32-bit wide. Therefore, a symbol size of 32 bit ensures a good chance in finding reoccurring equal symbols. Another important factor to consider is the wordsize of the CPU or - in case of PACS - the DSP. Most DSP instruction sets only support 32-bit granularity.

So, *RZip* focuses on 32-bit words. Given an arbitrary 32-bit symbol of a data buffer, a logical question can be posed, "Does it reoccur, or not?" If so, "where in the buffer or how often does it repeat?"

Basically, *RZip* takes a symbol and looks ahead for recurrence within a certain index range. The index difference of the two occurrences is encoded taking already coded indices into account. After that, the next occurrence of the symbol is sought if not the end of the buffer is encountered. In case there are no more occurrences, the source buffer is investigated for the next symbol.

The distances can be encoded in different ways. One way is to use the maximum distance as an indicator for no more recurrences. In case of PACS a binary flag after a symbol in the encoded data stream indicates either that an offset will follow or that there are no more occurrences for the current symbol.

Two parameters determine the performance of the algorithm:

- The size of a symbol quantifies the number of bits per symbol. In case of PACS this is fixed at 32 bit per symbol.
- Δ sets the width of the range to look ahead for recurring symbols. For instance, a Δ of 4 means that $2^4 = 16$ indices will be checked.

To get a little more into the algorithm, an explanatory example run is given:

Let `SOURCE` be the data buffer to be compressed. Let `DEST` be the destination buffer where the compressed data will be put.

Δ is the parameter that determines the number of bits to use for encoding ranges. In the following example we choose 2, therefore the effective offset counter δ will be 0..3 ($= 2^2 - 1$). The size of a symbol shall be 32 bit.

Our `SOURCE` (symbol buffer) may look like:

```
A A B C A A C C B B  SOURCE
```

There is also need for a workbuffer. At the beginning of the algorithm, it has to be cleared.

```
0 0 0 0 0 0 0 0 0 0  WORK
```

a) Select the first unused (workbuffer = 0) Symbol and the workbuffer to 1.

```
A A B C A A C C B B  SOURCE
1 0 0 0 0 0 0 0 0 0  WORK
```

b) Look ahead if the symbol recurs within δ. If yes, code 1 within 1 bit and δ within Δ bits. Set the proper position of the found symbol in the workbuffer to 1. Reset the δ to 0 and continue until no further occurrences are found, then code 0 in 1 bit.

c) Go back to a) until the end of the buffer.

First, all `A`s are coded.

```
A A B C A A C C B B  SOURCE
1 1 0 0 1 1 0 0 0 0  WORK
A y0. . y2y0 n  DEST
```

The next symbol to code is `B`.

```
    A A B C A A C C B B  SOURCE
    1 1 1 0 1 1 0 0 0 0  WORK
Ay0y2y0n B n  DEST
```

Next one is `C`.

```
    A A B C A A C C B B  SOURCE
    1 1 1 1 1 1 1 1 0 0  WORK
Ay0y2y0nBn C . . y0y0 n  DEST
```

And finally, `B` again.

```
    A A B C A A C C B B  SOURCE
    1 1 1 1 1 1 1 1 1 1  WORK
Ay0y2y0nBnCy0y0n B y0 n  DEST
```

In this example, 10 symbols of 32 bit size are encoded to 4 symbols plus 10 flags plus 6 ranges à $\Delta = 2$ bit. So, the input stream was 320 bit and the output stream is 150 bit. Therefore, the achieved compression ratio in this case is 2.13.

Note that the difference between `A A` will be encoded 0 (0 symbols are between them). The difference between `A X X A` will be encoded 2 if the `X` have not been coded before and if a range of 2 is allowed due to the set Δ (this should be the case). In case you have already coded all `A`s, the difference between the `B`s in `B A C B` will be encoded as 1 (the `A`s are already invisible due to the mask in the workbuffer).

Once a buffer has been compressed with a set of parameters, it can be encoded another time with different parameters. For example, DMC header compression works best with $\Delta = 6$ applied *twice*. Using more than three iterations did not yield any more compression in most cases.

ACKNOWLEDGMENTS

This work was supported by the Austrian Federal Ministry for Transport, Innovation and Technology.

REFERENCES

1. "The herschel mission, scientific objectives and this meeting," in *The Promise of the Herschel Space Observatory*, G. L. Pilbratt, J. Cernicharo, H. M. Heras, T. Prusti, and R. Harris, eds., *ESA SP* **460**, 2001.
2. A. Poglitsch, N. Geis, and C. Waelkens, "Photoconductor array camera and spectrometer for first," in *UV, Optical and IR Space Telescopes and Instruments VI*, *SPIE* **4013**, 2000.
3. R. Ottensamer, A. N. Belbachir, H. Bischof, F. Kerschbaum, and C. Reimers, "The austrian herschel/pacs on-board reduction work package," in *Hvar Obs. Bull.*, **23**, 2002.
4. H. Bischof, A. N. Belbachir, D. Hönigmann, and F. Kerschbaum, "A data reduction concept for first/pacs," in *UV, Optical and IR Space Telescopes and Instruments VI*, *SPIE* **4013**, 2000.
5. R. Ottensamer, A. N. Belbachir, H. Bischof, H. Feuchtgruber, F. Kerschbaum, C. Reimers, and E. Wieprecht, "The herschel/pacs on-board data reduction concept," in *Astronomical Data Analysis Software and Systems XI*, D. A. Bohlender, D. Durand, and T. H. Handley, eds., *ASP Conference Proceedings* **Vol. 281**, 2002.
6. D. J. Hatley and I. Pirbhai, *Strategies for Real-Time System Specification*, Dorset House Publishing, New York, 1987.
7. F. Kerschbaum, H. Bischof, A. N. Belbachir, T. Lebzelter, and D. Hönigmann, "Evaluation of first/pacs data compression on iso data," in *UV, Optical and IR Space Telescopes and Instruments VI*, *SPIE* **4013**, 2000.
8. F. Kordes, R. Hogendoorn, and J. Marchand, *Handbook of data compression algorithms*, TM-06, ESA, 1990.
9. C. C. for Space Data Systems, *Lossless Data Compression (Green Book)*, CCSDS Secretariat, NASA, Washington, D.C., 1997.
10. C. Reimers, A. N. Belbachir, H. Bischof, R. Ottensamer, H. Feuchtgruber, F. Kerschbaum, and A. Poglitsch, "A feasibility study of on-board data compression for infrared cameras of space observatories," in *7th International Conference on Pattern Recognition*, 2004.