How RMAP improves in-flight update of on-board software via SpaceWire

SpaceWire Missions and Applications, Short Paper

Holger Michel, Adrian Belger, Björn Fiethe, Tobias Lange, Harald Michalik
Institute of Computer and Network Engineering
Technische Universität Braunschweig
Braunschweig, Germany
michel@ida.ing.tu-bs.de

Martin Kolleck
Max Planck Institute for Solar System Research
Göttingen, Germany

***Abstract*—Modern space probes such as Solar Orbiter employ a SpaceWire network to connect to on-board computer (OBC), solid state mass memory (SSMM), and scientific instruments. Management of SpaceWire links within scientific instruments is typically performed by a data processing module (DPM) featuring a space qualified processor that is executing on-board software. To adapt to changing mission requirements, account for failures and fix possible software bugs, the ability of uploading and patching instrument software is mandatory. However uploading and over-writing of the software's boot image cannot securely be performed by the software itself. If over-writing the boot image fails, the remaining image might be corrupted. So the processor may not be able to reboot successfully and no further upload would be possible. Therefore reception of uploaded patches must be performed by an independent entity. Currently, this is accomplished by a dedicated boot loader in separate memory area, to be qualified according to ECSS criticality category B. This boot loader processes uploading of patches and copies them to the second boot area, where the actual software including the operating system is stored. Due to the opportunity of modern processors to handle SpaceWire RMAP accesses (e.g. SpW-RTC, UT699, GR712RC [1], or upcoming NGMP [2]), it would be possible to perform uploading and patching of the instrument software independent of software execution using RMAP. This would dramatically simplify the development, eliminate the need for a class-B qualified boot loader, and will inherently improve reliability, as reception of patches would entirely be performed by hardware. This paper presents a possible update and patch process for boot images using hardware based RMAP features. Furthermore implications of the standard ECSS services affecting such patching routines are discussed.**

***Index Terms*—SpaceWire, RMAP, in-flight update, boot loader, CCSDS PUS.**

I. Introduction

In modern scientific instruments a data processing module (DPM) handles processing of science data, instrument control, and telecommand (TC), telemetry (TM) and housekeeping (HK) communication. To receive TC and send TM and HK to the on-board computer (OBC) and solid state mass memory (SSMM) modern space probes such as Solar Orbiter or BepiColombo are equipped with a SpaceWire network. The DPM typically features a processor running instrument software to process TC, generate TM and HK, and perform instrument control. The instrument software is stored in a boot memory contained in the DPM and is booted automatically on power up. Due to changing scientific mission requirements or handling of unexpected difficulties with the instrument, this software may need to be exchanged or updated. Furthermore, if in disregard of instrument changing scientific requirements, the instrument was equipped with software that cannot be fixed and updated it would be required to completely qualify the software to almost highest ECSS criticality category. This causes additional effort and limits possibilities like dynamic memory allocation. Dynamic memory allocation in turn is essential for fast external interfaces using direct memory access (DMA).

II. Telecommand (TC) and Housekeeping (HK) Standards and Structure

The standard for data structures in TC, TM, and HK packets in ESA spacecrafts is ECSS-E-70-41A [3], which is based on guidelines agreed on in the Consultative Committee for Space Data Systems (CCSDS) such as reference document [4]. The standard ECSS-E-70-41A [3] is a packet utilization standard (PUS), defining the packet structure and a set of standard services. For TC packets arriving at a scientific instrument, the defined structure is depicted in Figure 1. Also this PUS [4] defines a set of standard services. These services are functions of the commanded entity e.g. the instruments DPM consisting of a command, an action, and if applicable a reply. The standard services are listed in Table I.

For a particular spacecraft the contractor building the spacecraft platform performs a tailoring of this standard and selects mandatory and optional services, the payload instruments must be able to perform.

Packet Header (48 Bits)							Packet Data Field (Variable)			
Packet ID				Packet Sequence Control		Packet Length	Data Field Header (most services)	Source Data	Spare (Optional)	Packet Error Control (Optional)
Version Number (=0)	Type (=0)	Data Field Header Flag	Applica-tion Process ID	Grouping Flags	Source Sequence Count					
3	1	1	11	2	14					
16				16		16	Variable	Variable	Variable	Not CCSDS, but ECSS

Figure 1 : CCSDS packet structure as refined by ECSS-E-70-41A [3]

Table I : Table 1 standard service defined by ECSS-E-70-41A [2]

Service Type	Service Name
1	Telecommand verification service
2	Device command distribution service
3	Housekeeping & diagnostic data reporting service
4	Parameter statistics reporting service
5	Event reporting service
6	Memory management service
7	Not used
8	Function management service
9	Time management service
10	Not used
11	On-board operations scheduling service
12	On-board monitoring service
13	Large data transfer service
14	Packet forwarding control service
15	On-board storage and retrieval service
16	Not used
17	Test service
18	On-board operations procedure service
19	Event-action service

Within that standard set of services, the service that can be used to update the instrument software is service 6 subtype 2 “Load Memory using Absolute Addresses service”. After an upload and storing of a new software version has finished, the DPM would simply have to be rebooted. There is no standard service for rebooting a payload instrument; one possibility is to use the service 8 subtype 1 “Perform function” or to use a set of private services, if they are allocated for the mission, to implement the reboot function. It is not sufficient if a boot loader supports only these two services, instead the boot loader must also implement a set of minimal standard PUS services, so that spacecraft requirements for nominal operation are fulfilled and the spacecraft allows further operation and does not power down the instrument, Table II lists an exemplary set of services.

Table II : Exemplary collection of a set of services

Minimal services that need to be supported			
Service 1: TC Verification Service			
TM	1	1	TC acceptance success report
TM	1	2	TC acceptance failure report
TM	1	7	TC execution success report
TM	1	8	TC execution failure report
Service 6: Memory Management Service			
TC	6	2	Load data into memory area using absolute address
TC	6	5	Dump memory area using absolute address
TM	6	6	Memory dump using absolute address Report
TC	6	9	Check memory area using absolute address
TM	6	10	Memory check using absolute address Report
Services for which the boot loader may need to generate a reply that avoids tripping error detection by the OBC			
Service 3: Housekeeping and Diagnostic Data Reporting Service			
TM	3	25	Housekeeping Parameter Report
Service 5: Event Reporting Service			
TM	5	1	Normal / Progress Report
TM	5	2	Error / Anomaly Report - Low Severity –Warning
TM	5	3	Error / Anomaly Report - Medium Severity - Ground Action
TM	5	4	Error / Anomaly Report - High Severity - On-board Action
Service 9: Time Management Service			
Service 17: Test Service			
TM	17	1	Connection Test Response
TM	17	2	Connection Test Response Report
Service 19: Event-Action Service			
TC	19	1	Add an Event to the Detection List
TC	19	4	Enable Actions
TC	19	5	Disable Actions

III. CCSDS / PUS SERVICES IN SPACEWIRE PACKETS

In SpaceWire packets a protocol identifier defines the packet type [6]. RMAP is assigned to the protocol identifier value 0x01 and the CCSDS packet transfer protocol is assigned to the protocol identifier value 0x02. [7] defines how packets of the CCSDS packet transfer protocol are transmitted through a SpaceWire network by appending addressing, protocol identifier, a reserved and user application byte at the start of the packet and an EOP marker at the end of the packet, see Figure 2.

	Target Spw Address	Target Spw Address	Target Spw Address
Target Logical Address	Protocol Identifier	Reserved = 0x00	User Application
CCSDS Packet (First Byte)	CCSDS Packet	CCSDS Packet	CCSDS Packet
Target Spw Address	...	...	CCSDS Packet
CCSDS Packet	CCSDS Packet (Last Byte)	EOP	

Figure 2 : SpaceWire packet transporting a CCSDS packet as defined by [6] ECSS-E-ST-50-53C

IV. BOOT MEMORY OPTIONS AND ARCHITECTURE

In order to avoid the effort of qualifying the entire instrument software to highest ECSS criticality level and allow for updates during space flight, current DPMs (such as for the

Polarimetric and Helioseismic Imager (PHI) instrument on Solar Orbiter) have a two stage boot process, as depicted in the system in Figure 3. The default boot memory address range (0x0000 0000-0x0FFF FFFC) of the employed LEON processor connects to a non-volatile memory containing a minimal boot loader plus an additional boot memory which is larger in storage and the content of which can be exchanged. In the case of Solar Orbiter PHI DPM a minimal PROM memory could be implemented within the Microsemi RTAX2000 system FPGA and a redundant NOR-flash is used to store the second boot image, which includes an RTEMS operating system and the complete instrument software. The basic boot loader will need to initialize processor registers and the SpaceWire interface and implement a basic driver for the SpaceWire interface. As the SpaceWire interface in processors such as the GR712 RC [1] uses direct memory access a substantial amount of software complexity and therefore boot loader size is required. Subsequently the boot loader needs to perform basic TC and TM handling and check if an update of the boot software needs to be performed. As NOR-flash cannot be written directly like a simple SRAM device, a driver performing defined program sequences also needs to be integrated in the boot loader.

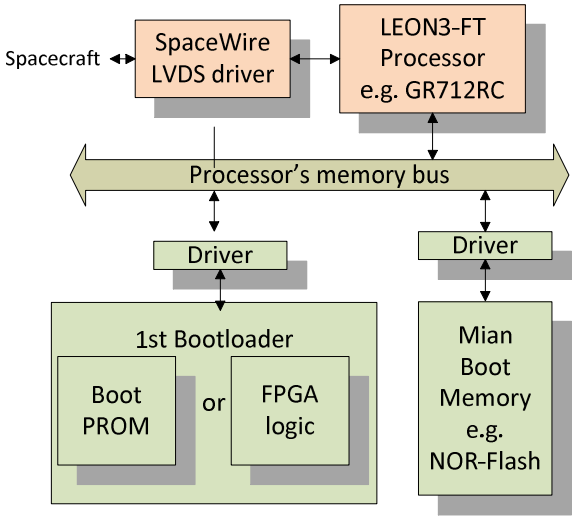

Figure 3 : Instrument data processing module boot memory set-up

V. Remote Memory Acces Protocol (RMAP)

The SpaceWire Remote Memory Access Protocol (RMAP) is a protocol that works over SpaceWire. This Protocol allows reading and writing memory remotely in a SpaceWire node. RMAP is defined in ECSS-E-ST-50-52C [5]. A memory write transaction is depicted in Figure 4 and it consists of SpaceWire addressing, protocol identifier, instruction, key, reply address, initiator logical, transaction identifier, address, data length, data and CRC-byte. In many radiation hard processors, such as e.g. the Aeroflex Gasiler GR712RC [1] the Aeroflex UT700 etc. the RMAP protocol is supported directly in hardware. In these devices even the complete address space from the processor bus can be read and written by RMAP. Therefore all the cores on the processor's AMBA bus including the debug support unit (DSU) can be reached. On ground the software debugger (GRMON) can connect to the processor through the SpaceWire interface without requiring an additional debug connector. Also on ground the second boot memory in the NOR-flash is written by uploading the boot image and a small program to the processor's working memory and then starting the small program that copies the boot image from working memory to the NOR-flash.

	Target Spw Address	Target Spw Address	Target Spw Address
Target Logical Address	Protocol Identifier	Instruction	Key
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Reply Address	Reply Address	Reply Address	Reply Address
Initiator Logical Addess	Transaction Identifier (MS)	Transaction Identifier (LS)	Extended Address
Address (MS)	Address	Address	Address (LS)
Data Length (MS)	Data Length	Data Length (LS)	Header CRC
Data	Data	Data	Data
Data	...	...	Data
Data	Data CRC	EOP	

Figure 4 : SpaceWire packet containing an RMAP write command (as given by ECSS-E-ST-50-52C [4])

VI. Using RMAP for Update of Instrument Software

As the boot memory update procedure on ground only uses the SpaceWire interface, which is also available via the space craft's OBC in flight, it seems logical to also use this procedure to update the boot software during flight operation. The only changes that would be necessary are, to create a separate RMAP command for the processor initialization that GRMON performs in the update process on ground and use RMAP's safety and error checking capabilities. However as RMAP provides access to all registers including debug support unit (DSU) this is not a problem. Thus a possible update procedure via RMAP for e.g. the GR712RC could consist of:

1) Stopping the software execution of the processor, by simply writing to the DSU register "break now"

2) Initializing processor register via DSU including program counter and Ancillary State Registers (ASRs)

3) Initializing the interrupt controller, memory configuration registers, and the GPIO controller

4) Disabling breakpoints and the debug mode by writing to the DSU control register and disabling the DSU Debug Mode Mask register

5) Uploading a program that writes boot memory content (see step 6) from working memory into the NOR-flash memory; alternatively this program could already be stored in another memory area and just copied to the working memory to minimize upload data volume, but still being replaceable by a newer version if needed

6) Uploading the new boot image via RMAP to the working memory. This could also be optimized by only uploading addresses and chunks of data where a difference to the current boot image occurs (patch)

7) Finally starting the copy to NOR-flash process by writing to the DSU break and single step register, when copying the data has finished the program can cause the processor to boot from the new boot image or cause a processor reset via a register in the supervisor FPGA.

As RMAP features a key byte and can be used with target logical address and RMAP cores are able to check these two bytes, they can be used as a security check to prevent any accidental triggering of the steps mentioned above, like e.g. stopping software execution.

VII. RMAP Access Problems

Despite appearing pretty straight forward an RMAP based software update procedure has some problems. Firstly in the many cases such as the GR712 processor the hardware RMAP support can be disabled by software, whereby a corrupted software image could potentially block any further accesses and disable software updates. Also software can set the SpaceWire logical address in the SpaceWire core of the GR712, which results in the SpaceWire core discarding any data that does not start with this logical address. Both of these problems could be somewhat alleviated by ensuring correct core settings through the first boot loader and halting the processor execution or including a wait before the second boot loader is started for a sufficient amount of time. As this is only a register access it will not increase the size of the first boot loader by a lot.

VIII. Conclusion

In the case of Solar Orbiter the ground operation team and particularly the OBC only offers CCSDS PUS services for payload instruments. Reasoning this with safety checks performed on the OBC and SSMM, which require a match between a SpaceWire packet's logical address and the application ID (APID) inside the CCSDS data packet, see Figure 1. However an RMAP write packet, which is depicted in Figure 4, starts with instruction, key and, reply address bytes and therefore cannot contain a CCSDS type header including APID which could be used for this safety check. Furthermore the Spacecraft would need to allow sending a packet with the protocol ID of RMAP (0x01). As the reply address in RMAP has 12 Bytes and logical addressing of a single byte is used it would be feasible to use the remaining 11 bytes for such header information. However this only applies to a reply, as in a request the position of an APID is the first byte of the reply address which is used for routing the reply packet.

Despite these difficulties an RMAP based update procedure would have several advantages. Such an update procedure would completely eliminate the need of any instrument software in the update process and thereby be inherently more reliable. Furthermore, if no software is required, this software does not need to be stored, which frees valuable FPGA resources or eliminates the need of an additional PROM, which would simplify system architecture and processor bus load as depicted in Figure 3. Additionally it would reduce development effort and costs, because no boot loader would need to be developed and qualified.

In conclusion RMAP would be an elegant, effective and more reliable mean of updating the instrument software, but there is a lack of harmonization between the two standard protocols of SpaceWire RMAP and the CCSDS PUS services and missing support by the OBC platform at least in the exemplary case of Solar Orbiter.

References

[1] GR712RC – User's Manual, Aeroflex Gaisler AB, http://www.gaisler.com, Issue 2.3, May 2014

[2] LEON4-N2X - Data Sheet and User's Manual, Aeroflex Gaisler AB, http://www.gaisler.com, Issue 2.3, May 2014

[3] CCSDS 102.0-B-5 Packet Telemetry, Blue Book, Issue 5, November 2000

[4] ECSS-E-70-41A "Ground systems and operations - Telemetry and telecommand packet utilization", European Cooperation for Space Standardization http://www.ecss.nl/, January 2003

[5] ECSS-E-ST-50-52C "SpaceWire - Remote memory access protocol", European Cooperation for Space Standardization http://www.ecss.nl/, February 2010

[6] ECSS-E-ST-50-51C "SpaceWire protocol identification", European Cooperation for Space Standardization http://www.ecss.nl/, February 2010

[7] ECSS-E-ST-50-53C "SpaceWire - CCSDS packet transfer protocol", European Cooperation for Space Standardization http://www.ecss.nl/, February 2010