LAT Flight Software

Primary Boot Code

Number	LAT-TD-1806-09
Subsystem	Data Acquisition/Flight Software
Supersedes:	LAT-TD-1806-08
Type:	Design Description
Author:	D.L. Wood, R.C. Caperoon, & D. May
Created:	15 April 2004
Updated:	9 March 2005
Printed:	9 March 2005

A description of the LAT Primary Boot Code for the SIU and EPU CPU boards. The detailed design of the primary bootstrap code is presented. The boot code reset operations, command and telemetry operations, and error handling are discussed.

Document Approval

Prepared By:			
	D.Wood	LAT Flight Software	Date
Approved By:			
	G.Haller	LAT Electronics Manager	Date
Approved By:			
	J.J.Russell	LAT Flight Software Manager	Date

Record Of Changes

Version	Date	Title Or Brief Description	Entered By
4	12 January 2002	First signed off release	A.P.Waite
5	6 November 2003	Heavily revised and augmented	R.C. Caperoon
6	3 February 2004	Additional exception info in boot diagnostics Changes in memory maps Addition of PBC configuration items Addition of test case types Changes and clarifications to memory test Clarifications to EEPROM bank header Addition of telemetry item descriptions Changes to exception processing	Ray Caperoon
7	15 April 2004	Scrubbed command and telemetry descriptions.	D.L. Wood
8	29 June 2004	Updated memory maps.	D. May
8+	26 October 2004	Updated reset action lists, configuration data description, and Primary Boot Flags description.	D. May
9	9 March 2005	Struck-through requirements that were obsolete and deleted those that were redundant. Added descriptions of the PPCI BIST and the PBC image selection process. Updated the description of the memory test. Updated the memory maps. Added descriptions of the application-level support and MSG status codes. Removed diagrams from Appendix B.	D. May

Contents

Figures

Tables

0 Introduction

The LAT Primary Boot Code (PBC) is a ROM-based executable which is responsible for bootstrapping the CPU board in both the SIU and EPU crates. The Primary Boot Code resides in the RAD750 board SUROM and can not be modified during flight.

0.0 LAT Boot Code Overview

The LAT RAD750 CPU boards employ a two-stage boot process that covers the time span between when the board is reset or powered on and when the application code is initialized. The Primary Boot Code, discussed in this document, is responsible for the low-level initialization of the RAD750 board, the execution of the PBC boot shell, and the loading and execution of a VxWorks RTOS executable image. The LAT Secondary Boot Code (SBC) covers the initialization of the VxWorks RTOS and the loading and initialization of the LAT application code modules. The two boot processes have been made as independent as possible, so that modifications to the SBC modules may be made without any changes to the PBC.

The LAT CPU boards begin execution of the Primary Boot Code in SUROM in response to the following reset conditions.

- Cold boot cases
    - The RAD750 CPU board has been powered on and the power-on reset signal is asserted.
    - The RAD750 CPU board has been manually hard reset. This is equivalent to the power-on reset scenario.
    - The RAD750 CPU hardware watchdog timer has expired.
- Warm boot cases, as the result of an explicit software command or error.

The Primary Boot Code operates in a similar manner for all of the reset cases. Most differences arise as the result of reporting the cause of the reset.

0.1 Primary Boot Code Requirements

The primary responsibilities of the GLAST LAT boot code are to initialize the RAD750 CPU board and to load and execute the VxWorks RTOS image stored in SIB EEPROM memory. In order to accomplish these primary actions, the LAT boot code is designed to the requirements shown in Figure 1.

These requirements are used to specify the features that must be implemented by the Primary Boot Code and to guide the testing program. Each requirement will have an associated test case. In addition to specifying the PBC requirements, Figure 1 also specifies the test type (one of

system-level test (ST), intrusive test (IT), analysis (A), demonstration (D) or inspection (I)) that will be used to show requirements compliance.

Figure 1 - Primary Boot Code Requirements

Definitions		
1000	The "external interface" for the SIU is the 1553 bus.	
1010	The "external interface" for the EPU is the serial interface on the LCB.	
1020	SUROM refers to the 256 kilobytes of EEPROM that reside on the RAD750. This EEPROM is NOT intended to be re-programmed during flight.	
1030	SIB EEPROM refers to the EEPROM resident on the Spacecraft Interface Board (SIB). This EEPROM is intended to be re-programmed during flight.	
Basics		
2000	All code and data necessary to support the startup diagnostics and PBC SHALL reside in the RAD750 SUROM.	I
2010	~~The same load of PBC SHALL support both the SIU and EPU~~.	I
Modes and Transitions		
3000	Power-up and external reset SHALL cause a restart of the PBC residing in the system's SUROM in the RAD750.	ST
3010	Upon restart of the PBC as a result of power-up or external reset, the PBC SHALL initialize the RAM required to execute the PBC.	I
3020	The soft reset command SHALL cause a restart of the PBC residing in the system's SUROM in the RAD750.	ST
3030	The watchdog reset condition SHALL cause a restart of the PBC residing in the system's SUROM in the RAD750.	ST
3040	Upon restart of the PBC as a result of watchdog reset or warm reset, the PBC SHALL preserve contents of RAM used by the LAT application code to maintain logs, tables, and other data, which may be relevant for FDIR.	ST
3050	The PBC SHALL execute a set of basic diagnostic tests upon initial startup, then enter a state waiting for commands to be received over the external interface.	I
3060	~~If no command directed to the PBC is received on the external interface during a period of time 600 seconds after completing the startup diagnostics, the PBC SHALL load and execute the RTOS stored in SIB EEPROM.~~	ST

3070	~~If any commands directed to the PBC are received on the external interface during a period of time 600 seconds after completing the startup diagnostics, the PBC SHALL enter the boot shell state and wait for further commands on the external interface.~~	ST
	Startup and Startup Diagnostics	
4000	The PBC SHALL initialize the RAD750 CPU, memory controller, and Power PCI bridge chip to a working state.	ST
4010	The PBC SHALL store any system errors occurring during PBC startup in an error log.	IT
4020	The PBC SHALL store the source of the reset in an error log.	ST
4030	The PBC SHALL provide test stimulus to the various hardware components of the CPU crate in order to test the operability of on-board systems. This will include memory tests, interface checking, response checking, and checksums of code segments to determine the validity of the on-board memory.	IT, ST, I
4040	The PBC SHALL test the minimal processor, memory, and interface functionality required for successful PBC operation.	I
4050	The PBC SHALL indicate startup status over the discrete interface lines. The states of the discrete lines and their meanings are defined in this document.	IT
	Command/Telemetry Functionality	
5000	The PBC SHALL provide an initial capability to receive boot command packets from an external interface. These command packets are defined in the *LAT Flight Software Telecommand and Telemetry Formats* document.	ST
5010	The PBC SHALL transmit Boot Housekeeping Telemetry packets over an external interface. These Boot Housekeeping Telemetry packets are defined in the *LAT Flight Software Telecommand and Telemetry Formats* document.	ST
5030	The PBC SHALL provide, upon command, the capability to dump memory data in telemetry. The dump data is contained in the Boot Housekeeping Telemetry packets as documented in the *LAT Flight Software Telecommand and Telemetry Formats* document.	ST
5040	The PBC SHALL provide, upon command, the capability to upload data into any memory mapped address. This includes memory mapped registers, SDRAM, and memory mapped devices on the PCI bus.	ST
5041	The PBC SHALL provide, upon command, the capability to upload data to a PCI device header for a device present on the PCI bus.	ST
5050	When an SIATTITUDE command packet is received on the external interface, the PBC SHALL ignore the command.	ST

5060	When an SIANCILLIARY command packet is received on the external interface, the time information included in the command packet SHALL be recorded for use by the PBC software.	ST
5070	When an SITIMETONE command packet is received on the external interface, the PBC SHALL ignore the command.	ST
5080	~~When a BOOT START command packet is received on the external interface while the PBC is in the initial command timeout state, the PBC SHALL transition into the boot command processing state.~~	ST
5090	When a BOOT RESET command packet is received on the external interface, the PBC SHALL perform a warm reset of the boot software.	ST
5100	When a MEMORY DUMP command packet is received on the external interface, the PBC SHALL start a dump of processor or PCI memory. The dump data is contained in the Boot Housekeeping Telemetry packet as documented in the *LAT Flight Software Telecommand and Telemetry Formats* document.	ST
5110	When a MEMORY DUMP CANCEL command packet is received on the external interface while a memory dump is active, the PBC SHALL stop the memory dump and stop inserting the memory data into the telemetry stream.	ST
5120	When a BOOT ERROR DUMP command packet is received on the external interface, the PBC SHALL insert the error code log information into the telemetry stream. The error code data is contained in the Boot Housekeeping Telemetry packet as documented in the *LAT Flight Software Telecommand and Telemetry Formats* document.	ST
5130	When a BOOT EEPROM-RTOS EXECUTE command packet is received on the external interface, the PBC SHALL load and execute the RTOS stored in SIB EEPROM.	ST
5140	When a BOOT RAM-RTOS EXECUTE command packet is received on the external interface, the PBC SHALL load and execute the RTOS stored in RAM.	ST
	File load	
6000	Requirement deleted, not testable.	
6010	When a FILE UPLOAD START command packet is received on the external interface when the PBC file loader is in the IDLE state, the PBC file loader SHALL enter the LOAD state.	ST
6011	When a FILE UPLOAD START command packet is received on the external interface when the PBC file loader is in any state except the IDLE state, the PBC file loader SHALL reject the command.	ST
6020	When a FILE UPLOAD DATA command packet that IS NOT the last data packet is received on the external interface when the PBC file loader is in the LOAD state, the PBC SHALL store the file data in the command packet into RAM.	ST

6030	When a FILE UPLOAD DATA command packet that IS the last data packet is received on the external interface when the PBC file loader is in the LOAD state, the PBC SHALL store the file data in the command packet into RAM and enter the VALID state.	ST
6031	When a FILE UPLOAD DATA command packet is received on the external interface when the PBC file loader is in any state except the LOAD state, the PBC file SHALL reject the command.	ST
6040	Deleted, since it was a duplicate of requirement 6030.	ST
6050	When a FILE UPLOAD CANCEL command packet is received on the external interface, the PBC SHALL cancel any active file load and enter the IDLE state.	ST
6060	When a FILE UPLOAD COMMIT command packet is received on the external interface, the PBC SHALL copy data previously loaded in the RAM upload buffer to the selected file region in the SIB EEPROM.	ST
6070	Deleted, since it was a duplicate of requirement 6020.	ST
6080	The PBC SHALL accept file uploads across the external interface for the purpose of replacing RTOS and second stage boot executables in SIB EEPROM memory.	ST
6090	The PBC SHALL validate file loads in RAM before committing the file loads to SIB EEPROM.	ST
6100	The PBC SHALL accept file uploads across the external interface for the purpose of bootstrapping a set of temporary RTOS and second stage boot executables from RAM.	ST
6110	The PBC SHALL accept commands across the external interface for the purpose of directing file uploads and controlling manual file operations.	ST
1553 Interface		
7000	The PBC, when running in an SIU, SHALL configure the 1553 interface as a remote terminal (RT) on the 1553 bus.	ST
7010	The PBC, when running in an SIU, SHALL receive CCSDS formatted command packets on the 1553 interface. These commands are documented in the *LAT Flight Software Telecommand and Telemetry Formats* document.	ST
7020	The PBC, when running in an SIU, SHALL send Boot Housekeeping Telemetry packets on the 1553 interface.	ST
LCB Interface		
8000	The PBC, when running in an EPU, SHALL configure the LCB interface for use by the PBC.	ST

8010	The PBC, when running in an EPU, SHALL receive CCSDS formatted command packets on the LCB interface. These commands are documented in the *LAT Flight Software Telecommand and Telemetry Formats* document.	ST
8020	The PBC, when running in an EPU, SHALL send Boot Housekeeping Telemetry packets on the LCB interface.	ST

0.2 Primary Boot Code Design

The LAT Primary Boot Code consists of two major components. The reset initialization code performs all of the low-level setup of the RAD750 board hardware. The command and telemetry boot shell is the boot code's other major component. The boot shell communicates through an external I/O interface, receiving commands and uploaded information and returning telemetry information.

0.3 Acronyms and Definitions

1553 – MIL-STD-1553B serial data bus specification. This is the serial data bus and data protocol implemented for the GLAST mission.

APID – CCSDS packet Application Identifier. This is a numerical code indicating the general type of data in a CCSDS packet.

EMC – Embedded Microcontroller located within the PowerPCI bridge chip. This microcontroller configures the PowerPCI bridge chip at a low-level, before the PowerPC CPU is allowed to run. It also monitors and handles low-level critical errors and watchdog time-outs.

EPU – Event Processing Unit. This is a computer crate that contains a RAD750 board and a LAT Communications Board.

HKP – Real-time Housekeeping telemetry data. This telemetry data relates to the health and safety of the LAT instrument.

LAT – GLAST Large Area Telescope. This is one of two science instruments for the GLAST mission.

LCB – LAT Communications Board. This is a cPCI board that allows the internal components of the LAT to communicate with one another.

PCI – Peripheral Component Interconnect general purpose parallel data bus. For the purposes of the RAD750 boot code, the CompactPCI (cPCI) variant is employed.

PID – Programmable Discrete. The RAD750 CPU board contains 32 channels of digital I/O. The Primary Boot Code uses two channels configured as outputs.

PPC – PowerPC microprocessor. This is the CPU provided on the BAE RAD750 board.

PPCI – PowerPCI bridge chip. This is an ASIC on the BAE RAD750 board which provides a memory controller and a PCI host bridge.

RTOS – Real Time Operating System. This is the VxWorks operating system used by the LAT.

SC – GLAST Spacecraft, as built by Spectrum Astro. Refer to the *GLAST LAT Instrument – Spacecraft Interface Control Document* for the formal specifications of the SC as seen by the LAT.

SDRAM – RAD750 board 128 MB of synchronous DRAM. The SDRAM serves as the RAD750 main memory.

SIB – Spacecraft Interface Board. This is the board that contains the EEPROM that holds the application and RTOS code images. In an SIU, this board also contains the LAT 1553 remote terminal hardware.

SIU – LAT Spacecraft Interface Unit. This is a computer crate that contains a RAD750 board and the 1553 Remote Terminal interface hardware.

SUROM – Startup ROM. This is the 256 KB of EEPROM memory on the RAD750 board that contains the Primary Boot Code. The SUROM is only programmable on the bench through the PPCI JTAG interface.

0.4 Reference Documents

GLAST 1553 Bus Protocol Interface Control Document, Spectrum Astro, Inc.

GLAST LAT Flight Software – Telecommand and Telemetry Formats

GLAST LAT Instrument – Spacecraft Interface Requirements Document, 433-IRD-0001 Revision B, NASA Goddard Space Flight Center, April 2002.

RAD750 Board Hardware User's Manual, *Document #234A533*, BAE Systems, December 2000.

RAD750 Board Software User's Manual, *Document #234A535*, BAE Systems, April 2001.

Embedded Microcontroller (EMC) User's Manual, Document #234A552, BAE Systems, April 2001.

MCP750 RISC Microprocessor User's Manual, Motorola Inc., 1997.

GLAST Spacecraft Interface Board Hardware Specification, LAT Hardware Specification Document.

LAT Communications Board, LAT Hardware Specification Document.

CTDB 1553 Drivers, LAT Flight Software Design Description Document.

LAT Communications Board Driver, Software Architecture and Interfaces, LAT Flight Software Design Description Document.

CCSDS User Manual, LAT Flight Software User Manual.

ZLIB User Manual, LAT Flight Software User Manual.

FILE User Manual, LAT Flight Software User Manual.

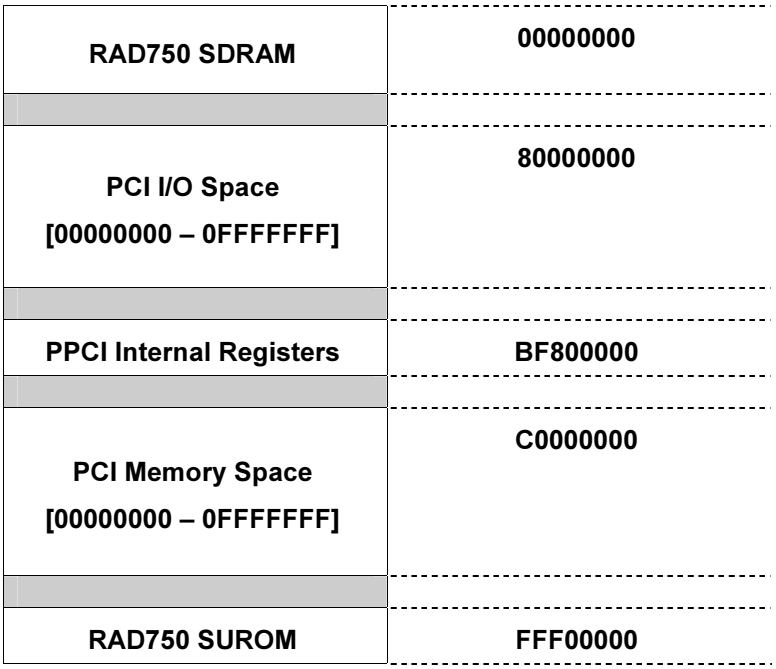
1 Reset Operations

The reset initialization portion of the Primary Boot Code is written in EMC and PPC assembly languages. Its primary duties are to initialize the RAD750 board hardware, perform critical diagnostic tests, and prepare SDRAM memory for the C portion of the boot code. The C portion of the boot code is linked at build time to run at fixed addresses in SDRAM and SUROM. Therefore, the reset initialization code must establish a working area of SDRAM for the C code to run, load the code, and begin execution of the boot shell.

1.0 PPCI Reset

When the master power-on reset signal for the RAD750 board is asserted, the PPCI bridge chip and its embedded EMC controller are reset. This causes not only all of the PPCI control functions to be reset, but also the PCI bus and PPC processor. The cause of the hardware reset is recorded in the PPCI BIST Status Register (at address BF86002C). When reset, the PPCI bridge chip hardware establishes the default memory map shown in Figure 2.

Figure 2 - Primary Boot Code Hardware Memory Map



The PPCI bridge chip holds the PPC processor reset signal in the asserted state until manually cleared by the EMC controller. When the PPCI bridge chip comes out of the reset state, the EMC controller begins execution of instructions starting at SUROM address FFF00020.

1.0.0 PPCI Built-In Self Test (BIST)

The first task that the EMC embedded microcontroller performs after reset is to configure and run the PowerPCI built-in self test (BIST). While the PPCI BIST is running, the EMC spins in a loop waiting for the BIST to complete. At the completion of the self test, the BIST hardware resets the PowerPCI bridge chip and the EMC. This reset causes the EMC to restart execution from FFF00020 again. This time, however, the EMC notices that the BIST has been run, so it does not run it again. The EMC determines that BIST has run by reading the BIST status register located at address BF86002C. If this register indicates a 'Logic BIST Reset', then the EMC knows that the cause of the reset was the completion of a self test.

1.0.1 PBC Image Selection

There are four, redundant images of the Primary Boot Code stored within the RAD750 SUROM. At the beginning of each image is a 32-byte header that contains the image's length and adler32 checksum (along with other, unused information). Following the completion of PPCI BIST, the EMC finds a valid PBC image and begins executing it. The EMC does this by verifying the checksums of the images and executing the first valid image that it finds. If it finds no valid image, the EMC executes the first image by default.

The EMC stores the status of the image selection process in its General-Purpose Register 10. The bits within this 32-bit status value have the following meaning:

- bits 0-15: Number of EMC vectors handled while verifying images.
- bits 16-26: Unused.
- bit 27: Set if no valid image was found and the default image was selected.
- bit 28: Set if the 4th image was checked because the first 3 images were invalid.
- bit 29: Set if the 3rd image was checked because the first 2 images were invalid.
- bit 30: Set if the 2nd image was checked because the first image was invalid.
- bit 31: Set if the 1st image was checked (should always be set).

At reset, the EMC's Vector Anchor Register points to a default vector table located at address FFF00000. When the EMC begins executing the selected PBC image, however, it changes the value of the Vector Anchor Register so that it points to a vector table within the selected image.

1.0.2 PPCI Configuration

Before the EMC allows the PPC processor to begin execution of its reset initialization code, it performs the following actions:

- The PPCI clock control register is initialized to set the PCI bus clock at 33 MHz, the PPCI timer clock at 8.25 MHz, and the CPU/Memory clock at 33 MHz. The PPC PLL configuration lines are changed from their default state so that the PPC will start executing at 115.5 MHz (3.5x multiplier).
- The PPCI memory controller Bank 0 is configured to map the 256 KB of SUROM at address FFF00000 as read-only memory. The ROM timing parameters are configured, the SECDED error detection and correction mode is selected, and error scrubbing is disabled for this memory bank.
- The PPCI memory controller Bank 1 is configured to map the 128 MB of system SDRAM at address 00000000 as read-write memory. The SDRAM addressing mode and timing parameters are configured, the nibble error detection and correction mode is selected, and error scrubbing is disabled for this memory bank.

- The User Defined registers (A, B, and C) are configured to enable machine checks, P60x errors, snoop_ws and cache snooping, and DBW0.
- EMC vector interrupts 0, 1, 5, 6, and 7 are enabled.
- A built-in self test (BIST) of the RAD750 CPU is performed. The results are logged to one of the EMC registers where the PPC processor can retrieve them later. Unlike the PPCI BIST, however, the EMC is not reset at the completion of the RAD750 BIST.

Once the EMC has completed its reset operations, the reset signal to the PPC processor is released and the PPC begins execution at the standard address FFF00100. The EMC controller then enters the "monitor" state and does not become active again unless signaled by the PPC processor.

1.1 CPU Reset

When released from reset, the PowerPC CPU begins executing code at the reset exception address FFF00100. The first task that the CPU performs when it executes the code at this address is to determine which PBC image it should execute. It determines this by reading the EMC Vector Anchor Register, which points to the EMC vector table located within the selected PBC image. By convention, the 32-bit word immediately before the EMC vector table in each image contains the CPU's entry point into the image (placed there by the pbc_makefile that builds the images). By reading this entry point address, the CPU determines which image it should execute.

The reset initialization bootstrap code for the CPU is located at the image entry point. This code is written in assembly to (1) allow access to special instructions needed to initialize the CPU and (2) ensure that RAM is not accessed before being tested. The first task for this bootstrap code is to determine the cause of the reset. For 'cold' resets (power-on, hardware, watchdog, etc.), the CPU sets the Primary Boot Flags appropriately. For 'warm' resets (commanded), the CPU expects to find the Primary Boot Flags in the r3 register. Once the Primary Boot Flags are established, the CPU configures itself and the RAD750 board as follows.

- The MSR register is set so that the processor uses exception vector base address FFF00000 (in SUROM) and external interrupts are disabled.
- The PPCI watchdog timer counter (address BF880058) is set for the maximum timeout period of approximately 70 minutes. To do this, the watchdog timer counter is set to the maximum value of FFFFFFFFF.
- The PIDs 5 and 6 are set to zero, indicating no error has occurred yet.
- The HID0 register is set to enable machine checks and disable everything else, including cache operation and machine optimizations.
- The MMU DBAT registers are initialized to create the SDRAM, SUROM, PPCI register, and PCI I/O memory spaces, in which addresses are mapped directly (i.e. not translated). The SDRAM and SUROM memory regions are cache enabled. The MMU page tables are disabled, as is instruction address mapping.
- The PPC instruction and data caches are not enabled at this time.
- The PCI_RST# signal is asserted to reset the devices on the PCI bus.
- The PPC branch history table and parallel instruction execution optimizations are enabled.
- SDRAM scrubbing is disabled in preparation for running the memory tests.

- A spare RAM column is enabled if requested in the Primary Boot Flags.
- The RAM is tested.
- The BIST results, Primary Boot Flags, and memory test results are stored in the boot diagnostics region of RAM.
- Passive memory scrubbing is enabled in the PPCI memory controller for bank 1 SDRAM.

1.1.0 PPC Virtual Memory Map

The Primary Boot Code uses the Data Block Address Translation (DBAT) feature of the CPU's Memory Management Unit (MMU) to map the memory regions from the hardware memory map shown in Figure 2 to those shown in Figure 3. The page table entry feature of the MMU is not used because it does not function properly in the RAD750 CPU.

Five blocks of memory are required by the PBC but only 4 DBAT entries are available. Therefore, one of the blocks – the PCI configuration register block – is not mapped by a DBAT. To access the registers in this block, the memory manager must be disabled. This block was chosen as the exception to minimize the number of accesses made by the PPC while the memory manager is disabled. These registers are needed only to enable configuration accesses on the PCI bus and are only done during PBC configuration.

Figure 3 - Primary Boot Code Virtual Memory Map

Region	Address	DBAT
RAD750 SDRAM	00000000	DBAT 0
Not Mapped	08000000	
PCI Configuration Registers	80000000	Not managed, see text for explanation
Not Mapped	80000D00	
PPCI Internal Registers	BF800000	DBAT 1
Not Mapped	BF900000	
PCI Memory Space [00000000 – 0FFFFFFF]	C0000000	DBAT 2
Not Mapped	D0000000	
RAD750 SUROM	FFF00000	DBAT 3
Not Mapped	FFF40000	

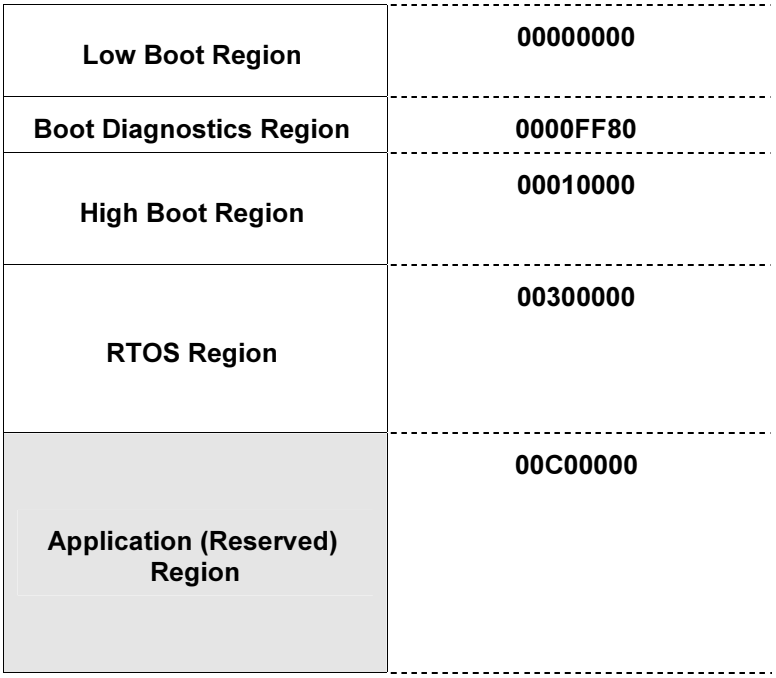
1.2 Memory Test

During the reset initialization procedure, the Primary Boot Code tests the SDRAM that it will use for operation. As with the processor initialization code, the memory test code is written in PPC assembly in order to guarantee that SDRAM is not accessed before it has been tested.

1.2.0 Memory Regions Tested

For the purposes of this memory test, the PBC defines five regions of the SDRAM memory, as follows:

Figure 4 - SDRAM Regions



If the Primary Boot Flags indicate that this is a cold boot (power on or hardware reset), then a memory test is run on the Low Boot, Boot Diagnostics, High Boot, and RTOS regions of SDRAM.

If this is an OCB reset, then a memory test is run only on the Low Boot and High Boot regions. This allows the boot code to access reboot and debug information stored in the boot diagnostics, RTOS, and application regions. An OCB reset is one that is initiated by the EMC in response to a watchdog timeout, checkstop, EMC critical error, or vector 1 interrupt (vector 1 interrupts are commanded by the CPU by writing directly to the Vector Control register).

If the Primary Boot Flags indicate that this is a warm boot, the memory test is run only on the regions requested within the Primary Boot Flags.

If errors are found in the SDRAM, corrective action is attempted. This action consists of swapping out the spare byte column, setting the PID critical status value (see Section 3.1.0), and retesting SDRAM.

1.2.1 Memory Test Algorithm

For each region of memory, the following memory test algorithm is performed:

1. Write each 32-bit word in the region with its 32-bit address.
2. Clear any pending memory controller errors.
3. Test each 32-bit word within the region, as follows:
    a. Read and verify that the word contains the address value written in step 1. Report ‘address’ failure (code 7) if the value is not correct.
    b. Exclusive-or the 32-bit value with 11111111, which inverts the least-significant bit (bit 3) of each nibble within the word. Verify the result and report ‘ones’ failure (code 3) if the value is not correct.
    c. Exclusive-or the 32-bit value with 33333333, which restores bit 3 and inverts bit 2 of each nibble within the word. Verify the result and report ‘twos’ failure (code 4) if the value is not correct.
    d. Exclusive-or the 32-bit value with 66666666, which restores bit 2 and inverts bit 1 of each nibble within the word. Verify the result and report ‘fours’ failure (code 5) if the value is not correct.
    e. Exclusive-or the 32-bit value with CCCCCCCC, which restores bit 1 and inverts the most-significant bit (bit 0) of each nibble within the word. Verify the result and report ‘eights’ failure (code 6) if the value is not correct.
    f. Exclusive-or the 32-bit value with 88888888, which restores bit 0 of each nibble within the word. Verify the result and report ‘zeroes’ failure (code 2) if the value is not correct.
    g. Invert the 32-bit value so that it contains a different value than it started with. This value is not verified.
4. Check for memory controller errors, including correctable errors. Report ‘controller’ failure (code 8) if the memory controller indicates that an error occurred.

As an example, consider the memory location 0002F3C4. The following patterns are written and verified:

Figure 5 - Memory Test Example

Step 1	0002F3C4
Step 3b	1113E2D5
Step 3c	2220D1E6
Step 3d	4446B780
Step 3e	888A7B4C
Step 3f	0002F3C4
Step 3g	FFFD0C3B

1.2.2 Memory Failure Remediation

If a failure is encountered during the memory test, testing of the current region is halted. For failures other memory controller errors (code 8), remediation is attempted and the memory test is restarted from the beginning.

The remediation process begins by determining the column index of the data byte that contained an incorrect value. If multiple errors have occurred, only the index of the most significant incorrect byte is considered. If a spare byte is not already enabled, the bad byte column index is used to program the PPCI read and write sparing logic registers. After the sparing logic is programmed, the memory test is run again from the beginning.

Sparing logic is programmed only once. If a memory test fails a second time, or if the initial Primary Boot Flags enabled a spare byte before the memory test was run, the memory test stops. The results are recorded and no further memory tests are run.

If the memory test fails and remediation is attempted, the output PID value is set to indicate hard memory error (see section 3.1.0).

The LAT uses the 3U version of the BAE RAD750 board, which provides one spare memory column. This byte column is shared with the NIBBLE error correction logic, however. Therefore, when the spare memory column is used, the error correction is changed from NIBBLE error correction to SECDED error correction, which does not require the use of a spare memory column.

1.3 SUROM Memory Map

The majority of the Primary Boot Code consists of the boot shell. The boot shell's code and constant data segments (*.text* and *.rodata*) remain in SUROM, from which the CPU executes them directly. Also contained within the SUROM is the PPCI EMC code, which always executes directly from SUROM – both during reset operations and during normal operations if one of the EMC vectors is invoked. Figure 6 shows the overall memory map for the SUROM. The values in the right column of the figure are absolute addresses.

The **EMC Default Vector Table** is an 8-entry table defining the addresses of the EMC vector handlers that are used during the image selection procedure. Each 32-bit entry contains an EMC branch instruction that directs execution to a single common vector handler located within the EMC image selection code.

The **EMC Image Selection Code** is divided into two portions of the SUROM. It is divided to avoid the CPU reset stub and exception handlers, which have locations that are fixed by the PowerPC architecture. This image selection code is described in section 1.0.1.

The **CPU Reset Stub and Boot Exception Vectors** contain the initial code executed by the CPU after reset. As described in section 1.1, this code determines which PBC image has been selected by the EMC, and then jumps to the chosen image's entry point.

The **PBC Build Information** contains a NULL-terminated text string that describes where and when the Primary Boot Code was built. This is immediately followed by another NULL-terminated text string that lists the names and versions of the packages that are included in the PBC.

The bulk of the SUROM is consumed by the four redundant **PBC Images**. Figure 7 shows the memory map for these images. The values in the right column of the figure are offsets from the start of the image.

Figure 6 – SUROM Memory Map

EMC Default Vector Table	**FFF00000**
EMC Image Selection Code (1/2)	**FFF00020**
CPU Reset Stub and Boot Exception Vectors	**FFF00100**
PBC Build Information	**FFF01000**
EMC Image Selection Code (2/2)	**FFF01100**
PBC Image 0	**FFF01400**
PBC Image 1	**FFF10C00**
PBC Image 2	**FFF20400**
PBC Image 3	**FFF2FC00**
Unused	**FFF3F400**

Figure 7 – PBC Image Memory Map (Four Copies)

EMC Startup and Vector Code	**0000**
CPU Primary Boot Code (*.text and .rodata*)	**0800**
CPU Primary Boot Initialized Data (*.data*)	***etext***

The **EMC Startup and Vector Code** includes the PPCI configuration code described in section 1.0.2. It also includes the EMC vector table and vector handlers that are invoked if an EMC vector occurs while the CPU is executing code.

The **CPU Primary Boot Code** contains the *.text* and *.rodata* section of the PBC. It is from this portion of the selected PBC image that the CPU executes the boot shell.

The **CPU Primary Boot Initialized Data** contains the *.data* section of the PBC. During the C environment initialization, the CPU copies this portion of the PBC image to the appropriate location within the SDRAM.

1.4 C Environment Initialization

The last piece of assembly code run during reset initialization creates an environment for the execution of the boot shell code, which is written in C. To create this environment, the CPU performs the following steps:

- Moves the initialized data from the CPU Primary Boot Initialized Data portion of the SUROM into the Boot Code Program Initialized Data (*.data*) region of SDRAM.
- Clears the Boot Code Program Uninitialized Data (*.bss*) region of SDRAM.
- Clears the Boot Code Stack region of SDRAM.
- Sets the stack pointer (PPC register r1) to the top of the Boot Code Stack region, from which if grows downward toward lower addresses.

Figure 8 – RAM Boot Region Memory Map After C Environment Initialization

Reserved	**00000000**
Boot Shell Reset and Exception Vectors	**00000100**
Boot Code Stack	**00003000**
RTOS Parameter Region	**0000FD00**
EMC Parameter Region	**0000FF00**
Boot Diagnostics Region	**0000FF80**
	00010000
Boot Code Program Initialized Data (*.data*)	**002E8000**
Boot Code Program Uninitialized Data (*.bss*)	***_edata***
	00300000

The final task for the reset initialization code is to jump to the PBC_BootShell() C function in SUROM to start the boot shell. A 'boot type' code is passed to this function as the first (and only)

argument (within register r3 by convention). The boot type code indicates the source of the boot. It is available for use by the PBC, which passes its value to the Secondary Boot Code (SBC) when the RTOS is started. The SBC, in turn, makes this value available to the application code.

For hardware, watchdog, exception, and boot code panic restarts, the values of the boot type code are pre-defined, as shown in Figure 9. For warm restarts, the application code specifies the boot type as a 'VxWorks Start'.

Figure 9 - Boot Type Codes

Value	Meaning	Description
0	VxWorks Start	Processor was rebooted by the VxWorks application.
1	Cold Start	Processor was started from a power-on or hardware reset.
2	Watchdog Start	Processor was started from a watchdog timeout, checkstop, critical EMC error, or vector 1 interrupt.
3	Panic Start	Boot code has detected a panic situation and restarted. Possible panic sources are: • Boot Shell returned to caller. • Secondary Boot Code returned to PBC.
4	Exception Start	Processor was started due to an exception in the PBC.
5	Commanded Start	Reboot was commanded from the ground.

2 Boot Shell

The LAT Primary Boot Code shell application processes a simple list of commands and outputs a limited set of telemetry. The boot shell also contains the ability to receive file uploads and store those files in SIB EEPROM memory or in temporary SDRAM buffers. The boot shell listens for commands and upload blocks from the either the 1553 remote terminal interface (SIU) or LCB unsolicited data interface (EPU). The commands are limited to only what is absolutely necessary for booting the VxWorks RTOS and for updating the SIB EEPROM with a new RTOS image or other critical information. In its normal mode of operation, the boot shell waits for incoming commands without loading or executing the RTOS.

2.0 Initialization

The boot shell first creates a system memory configuration, initializes the I/O communications interface and executes the final boot diagnostics. It also configures some of the PPCI registers as follows:

- Ignore PCI master aborts since PCI device scans will produce master abort cycles.
- Enable OCB machine checks, disable 60X error interrupts, and disable 60X machine checks (the latter do not function properly on the RAD750).
- Enable internal machine check interrupts and prevent NMIs from generating machine checks.

2.0.0 SDRAM Memory Layout

The boot shell establlishes the layout of SDRAM memory as shown in Figure 10 below. The values in the right column are offsets from the beginning of SDRAM.

Figure 10 - RAM Memory Map After Boot Shell Initialization

Reserved	**00000000**
Boot Shell Reset and Exception Vectors	**00000100**
Boot Code Stack	**00003000**
RTOS Parameter Region	**0000FD00**
EMC Parameter Region	**0000FF00**
Boot Diagnostics Region	**0000FF80**
Second Stage Boot RAM Module 0	**00010000**
Second Stage Boot RAM Module 1	**00070000**
VxWorks RTOS RAM Load Region	**00080000**
Boot Shell File Upload Buffer	**00140000**
LCB Input Ring Buffer	**00200000**
I/O Output Buffer	**002A0000**
I/O Input Buffer	**002A1000**
Boot Shell Memory Heap	**002A2000**
Boot Code Program Data	**002E8000**
VxWorks RTOS Image	**00300000**
	00C00000

The **Boot Code Stack** and **Boot Code Program Data** regions are initialized by the boot shell initialization code. The **Boot Diagnostics Region** contains pertinent information about the boot state (see Section 3.0). The **Second Stage Boot RAM Modules** are for storing temporary, or trial, versions of the secondary boot modules (see Section 2.0.2).

The **VxWorks RTOS Image** region is where the RTOS text and data segments reside when it is loaded and started. The RTOS absolute binary image is inflated to this address and then executed by jumping to the start of the VxWorks RTOS Image, where the RTOS initialization entry point is located. The RTOS executable images are set at build time to run at this SDRAM address.

The **Boot Shell File Upload Buffer** is where uploaded file data is temporarily stored while waiting for the upload to complete and the file commit command to be issued. The **VxWorks RTOS RAM Load Region** is used to store a temporary version of an RTOS executable that is to be run but not committed to SIB EEPROM. This region is also used as a staging area where an RTOS load image from SIB EEPROM is copied before it is inflated, since the ZLIB inflation routine must run on data stored in a byte-accessible memory region.

The **I/O Output Buffer** serves as a common output region for both the SIU 1553 driver and the EPU LCB driver. The SIU 1553 driver uses this area to construct GLAST 1553 telemetry packet blocks when sending out housekeeping telemetry packets. The EPU LCB driver uses this area to contain the export command list for sending out housekeeping telemetry packets. The I/O output buffer base address meets the LCB DMA alignment requirements for export command lists.

The **I/O Input Buffer** serves as a common input region for both the SIU 1553 driver and EPU LCB driver. The SIU 1553 driver uses this area to receive incoming telecommand packets. The EPU LCB driver uses this area to extract incoming telecommand packets from the LCB ring buffer.

The **Boot Shell Memory Heap** satisfies the small number of dynamic memory allocations within the boot code. These allocations occur within the ZLIB inflate process, which requires approximately 100 KB of temporary memory to perform each file inflation. The original heap pointer starts at the beginning of the Boot Shell Memory Heap region and advances as each call to *malloc()* requests more memory. The heap *free()* function is a null operation since the heap does not need to be maintained beyond the start of the RTOS execution.

The **LCB Input Ring Buffer** region exists to support the reception of telecommand input packets through the LAT event fabric. The LCB input ring buffer base address meets the LCB DMA alignment requirements for unsolicited data result buffers.

2.0.1 PCI Setup

After the boot shell memory resources are mapped, the next stage in the startup sequence is the initialization of the I/O interfaces. If the boot shell is starting from a cold boot, the PCI bus has been held in reset by the PPCI bridge chip since hardware reset. If the boot shell is starting from a warm boot, however, the PCI bus has been held in reset since just before the memory test. The PCI bus reset is now de-asserted, releasing the I/O boards in the backplane from the reset state. Next, the PCI interface is configured by setting various PPCI registers as follows:

- BAR1 prefetch and speculative read are enabled.
- OSR ordering, the latency timer, and BAR2 writes are disabled from the PCI bus.
- The cache line size is set to 8.

- PCI protocol checks, PCI arbiter latency timeout checks, and PCI data phase timeout checks are enabled. The PCI arbiter latency timeout is set to its maximum value, but the PCI data phase timeout is set to its minimum value.
- PCI read parity checks are disabled.
- BAR1 is set to 40000000 and BAR2 is set to 3FF00000.
- PCI bus memory space accesses, PCI master accesses, PCI parity error processing, and the SERR# driver are enabled.
- Address/data stepping is disabled.

At this point, all of the I/O boards in the cPCI backplane are ready for configuration. The presence of cPCI boards in the backplane is ascertained by scanning the PCI configuration space headers. The boot shell recognizes two cPCI boards, the SIB and the LCB.

The boot shell programs certain PCI configuration members before the I/O drivers are initialized. A driver for use with the boot shell can assume the following items in the PCI configuration header have already been setup at initialization time:

1. All relevant BAR registers are assigned valid addresses in PCI Memory space.
2. The PCI configuration control register is enabled for target Memory operations as well as for bus mastering capability.
3. The PCI configuration cache line size and latency timer members are set to default values.

Note that the interrupt line members of the PCI configuration headers will be left at ‘0’. Drivers for use with the boot shell do not need interrupt assignments.

The PowerPCI bridge chip is configured to provide access to the RAD750 SDRAM from the cPCI bus at PCI memory space address 40000000. The LCB driver for the EPU version of the boot shell utilizes this capability to provide a DMA ring buffer to receive telecommand packet input.

2.0.1.0 SIU PCI Setup

In an SIU crate, the boot shell programs the necessary members of the SIB PCI configuration header to allow the 1553 and SIB EEPROM drivers to initialize themselves. The SIU boot shell does not configure the LCB board, since it does not use it. The SIU PCI memory maps and SIB assignments are shown below

Figure 11 - SIU PCI Memory Space Address Map

SIB BAR 0	**SIB Control/Status Registers [00000000 – 001FFFFF]**	**C0000000**
	SIB Summit 1553 Controller Internal Registers [00400000 – 0040007F]	**C0400000**
	SIB 1553 Shared Memory [00600000 – 0061FFFF]	**C0600000**
	SIB EEPROM Bank 0 [00800000 – 00BFFFFF]	**C0800000**
	SIB EEPROM Bank 1 [00C00000 – 00FFFFFF]	**C0C00000**

2.0.1.1 EPU PCI Setup

In an EPU crate, the boot shell programs the necessary members of the SIB PCI configuration header to allow the SIB EEPROM driver to initialize itself. The EPU boot shell also configures the LCB board so that the LCB boot driver can initialize itself. The EPU PCI memory maps and SIB and LCB assignments are shown below

Figure 12 - EPU PCI Memory Space Address Map

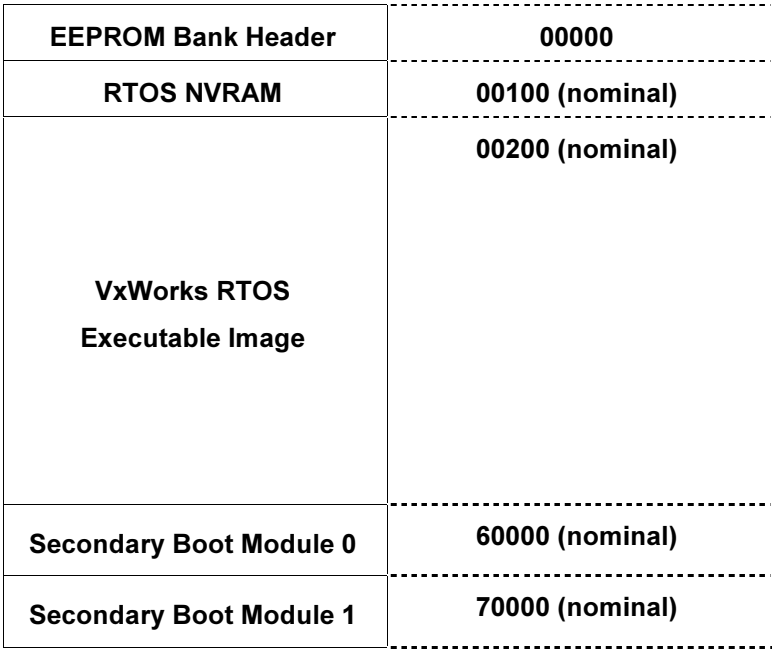
SIB BAR 0	**SIB Control/Status Registers [00000000 – 001FFFFF]**	**C0000000**
	SIB Summit 1553 Controller Internal Registers [00400000 – 0040007F]	**C0400000**
	SIB 1553 Shared Memory [00600000 – 0061FFFF]	**C0600000**
	SIB EEPROM Bank 0 [00800000 – 00BFFFFF]	**C0800000**
	SIB EEPROM Bank 1 [00C00000 – 00FFFFFF]	**C0C00000**
LCB BAR 0	**LCB Internal Registers and FIFO's [01000000 – 01003FFF]**	**C1000000**

2.0.2 SIB EEPROM Memory Setup

At initialization time, the boot shell configures the boot partitions of the two SIB EEPROM banks. These boot partitions are located at the bases of the SIB EEPROM banks – at CPU addresses C0800000 and C0C00000. As shown in Figure 13, the boot partition contains an EEPROM bank header, NVRAM for the RTOS, one copy of the VxWorks RTOS executable image, and two secondary boot modules. The values in the right column of the figure are offsets from the start of the SIB EEPROM bank.

The boot shell deals directly only with the VxWorks RTOS executable image, which may be stored in ZLIB compressed format. The NVRAM and boot modules hold critical data needed for the secondary boot process such as the secondary boot executable and the secondary boot application initialization script. These modules are loaded by the RTOS itself when it starts.

Figure 13 - SIB EEPROM Boot Partition Memory Map



The EEPROM Bank Header is stored at the first address of the SIB EEPROM Boot Partition. It is 256 bytes long and consists of:

- A Bank Header Checksum. This checksum uses the adler32 algorithm and is the checksum of the contents of the entire bank header except the checksum itself (the checksum over 252 bytes)
- The RTOS File Offset. This is the offset, in bytes, from the base of the SIB EEPROM Boot Partition to the VxWorks RTOS Executable Image.
- The RTOS File Size. This is the size, in bytes, of the VxWorks RTOS Executable Image, including both the header and data portions of the image.
- The Segment 0 File Offset. This is the offset, in bytes, from the base of the SIB EEPROM Boot Partition to the Secondary Boot Module 0 image.
- The Segment 0 File Size. This is the size, in bytes, of the Secondary Boot Module 0 image, including both the header and data portions of the image.

- The Segment 1 File Offset. This is the offset, in bytes, from the base of the SIB EEPROM Boot Partition to the Secondary Boot Module 1 image.
- The Segment 1 File Size. This is the size, in bytes, of the Secondary Boot Module 1 image, including both the header and data portions of the image.
- The NVRAM Offset. This is the offset, in bytes, from the base of the SIB EEPROM Boot Partition to the RTOS NVRAM.
- The NVRAM File Size. This is the size, in bytes, of the RTOS NVRAM.

Figure 14 - EEPROM Bank Header

Offset	Mnemonic	Description
0x00	BHDR_CHECKSUM	Bank Header Checksum.
0x04	BHDR_RTOS_PTR	Offset in bytes to the VxWorks RTOS Executable Image.
0x08	BHDR_RTOS_SIZE	Size in bytes of the VxWorks RTOS Executable Image.
0x0C	BHDR_SEG0_PTR	Offset in bytes to the Secondary Boot Module 0.
0x10	BHDR_SEG0_SIZE	Size in bytes of the Secondary Boot Module 0.
0x14	BHDR_SEG1_PTR	Offset in bytes to the Secondary Boot Module 1.
0x18	BHDR_SEG1_SIZE	Size in bytes of the Secondary Boot Module 1.
0x1C	BHDR_NVRAM_PTR	Offset in bytes to the RTOS NVRAM.
0x20	BHDR_NVRAM_SIZE	Size in bytes of the RTOS NVRAM.

Each of the images stored in the SIB EEPROM boot partition is prefixed by the standard LAT file header. The format of the header is shown in the LAT Flight Software Telecommand and Telemetry Formats Document. This header provides a length, checksum, and type information for the image data. The image data itself is stored in SIB EEPROM immediately following the file header.

2.0.3 PBC Configuration and Communications Setup

The LAT Primary Boot Code does not support the full functionality of the LAT CCSDS data packet protocol. The boot code recognizes only the boot shell operational command packets, file upload packets, memory dump and load command packets, and the SC ancillary message packets. On the telemetry output side, the boot code only supports the output of one fixed length housekeeping status packet.

For the crates configured to run as an SIU, the boot shell initializes the boot mode 1553 driver to provide the external interface. For crates configured to run as an EPU, the boot shell initializes the boot mode LCB driver to provide the external interface.

Any errors encountered during the communications driver initialization are reported on the PID status channels (see Section 3.1.0).

2.0.3.0 1553 Remote Terminal Setup

The SIU version of the Primary Boot Code utilizes the Summit remote terminal hardware interface located on the SIB board to provide communications capabilities. Because the SIU boot code runs with interrupts disabled and does not contain any multitasking support, the boot shell must use the polled mode version of the Summit/SIB driver. The boot shell initializes the remote terminal sub-address descriptor table in device-shared memory through the 1553 driver, and then begins listening for messages on the 1553 bus. The boot shell periodically calls the polled mode driver's query function to detect new 1553 bus activity. The boot version of the 1553 driver accepts telecommand input packets, but never enables the boot code to send out telecommand transmit packets. The polled mode driver inserts the boot shell HKP packet into the GLAST 1553 telemetry packet block.

2.0.3.1 LCB Communications Setup

The EPU version of the Primary Boot Code utilizes the LCB hardware interface to the LAT event fabric to provide communications capabilities. Because the EPU boot code runs with interrupts disabled and does not contain any multitasking support, the boot shell uses the polled mode version of the LCB driver. The boot shell periodically calls the polled mode driver's query function to detect new unsolicited data activity.

2.1 Command and Telemetry State Machine

The boot shell enters a simple command and telemetry state machine once it has started. This state machine implements an interactive mode of operation, and all activity is directed from telecommand packets sent across the external interface. In this state, the boot shell performs a polling loop that performs the following actions:

- Checks the external interface driver (1553 or LCB) for an indication that either a new telecommand packet has arrived or a communications error was detected.
- Checks the HKP telemetry state. For the SIU, the 1553 driver indicates if the last HKP packet has been sent and it is possible to send a new one. For the EPU, a new HKP packet is sent out 4 times per second. If it is time to generate a new HKP packet, the boot shell does so and sends it to the external interface driver (1553 or LCB).
- If a file commit operation targeting SIB EEPROM is in progress, the boot shell checks to see if the last write cycle has completed. If so, it starts the programming cycle for the next SIB EEPROM location if any remain.
- If a Memory Load operation is in progress, the boot shell writes the next chunk of data to its target location.
- Reads 128 words of data from SDRAM starting at the current scrub address, then increments the scrub address, wrapping to 0 when the ending address of 00300000 is reached.
- Checks certain error status registers within the PPCI and reports any error conditions that may be indicated.
- Resets the RAD750 PPCI watchdog timer to a timeout value of 600 seconds.

When an RTOS image is executed (by explicit command), the secondary boot process begins. Before executing the RTOS to start the secondary boot code, the RAD750 watchdog timer is set to a period of 600 seconds. This is to protect the secondary boot code from fatal errors.

2.2 Telecommand Input

The boot shell accepts input information in the form of standard GLAST CCSDS telecommand packets. The boot shell only recognizes a limited subset of the complete LAT 1553 telecommand list. It processes the telecommand packets to obtain the following information:

- Operational commands to provide control over the initialization process.
- File upload data for replacement of SIB EEPROM segments.
- Time information so that telemetry output may be time tagged.

Telecommands not listed in Sections 2.2.0, 2.2.1, 2.2.2, or 2.2.3 are not recognized by the Primary Boot Code and will be signaled with an operational command error in telemetry.

Each CCSDS telecommand packet is contained within a single 1553 data message or a single LCB DMA block. The first word of the telecommand header is always the first word of the 1553 data message or the first word of the LCB DMA transfer following the LATp status word.

The SIU telecommand packets are limited in total size by the GLAST 1553 protocol to no more than 62 bytes. The EPU telecommand packets are limited by the size of one LCB DMA transfer block to no more than 4096 bytes. Each telecommand packet is prefixed by a standard 8-byte header and is trailed by a 2-byte checksum. This leaves 52 bytes of actual command data in SIU telecommand packets and 4086 bytes of actual command data in EPU telecommand packets.

The boot shell expects certain telecommand header fields to have known values. Operational command packets are rejected immediately if any of the fixed header fields have an incorrect value. All command packets must be version ‘1’, type ‘telecommand’, and with secondary header ‘true’. The packet length member value must be limited to no more than the boot code operational mode allows. The total packet size must also be aligned to a two-byte boundary and the trailing checksum must be correct. The command packet secondary header consists of a single 16-bit word, of which 15 bits provide a command function code.

2.2.0 Operational Commands

All operational commands arrive as standalone CCSDS telecommand packets. The operational commands accepted by the boot shell are enumerated below:

Figure 15 – Boot Shell Operational Telecommands

APID	Function Code	Description
0x640	0	Boot Start. No-op which simply increments the command counter.
	1	Boot Reset. Forces a reset of the PBC into a known restart state.
	2	Boot Error Dump. Dumps the value of an error word queued by the PBC.
	3	Boot RTOS Execute. Begins execution of an RTOS image and the second-stage boot process.

2.2.1 File Uploads

The boot shell is capable of receiving file uploads. These uploads are useful for replacing old or corrupted RTOS images or secondary boot files stored in SIB EEPROM. It is also possible to test a new RTOS or secondary boot module without committing it to SIB EEPROM by uploading the file to RAM and running it directly from there. Since the RTOS files tend to be rather large, it may be tedious to use the boot shell upload facility often. Instead, the capability is present to fix any critical errors in the on-board RTOS files that prevent the LAT instrument from functioning properly. Because the RTOS code is primarily third-party and because those portions that are not have had an early development cycle, it is expected that the upload feature will be used during flight sparingly. More likely candidates for replacement are the secondary boot modules.

All upload data and commands arrive as CCSDS telecommand packets. A telecommand APID value, 0x641, has been reserved to designate file upload telecommands. Each upload is directed by a set of CCSDS telecommands that frame the actual data packets. The upload telecommand packet types are listed by function code below.

Figure 16 – Boot Shell File Upload Telecommands

APID	Function Code	Description
0x641	0	File Upload Start. Announces the beginning of a new file upload and provides its total size.
	1	File Upoad Cancel. Cancels an outstanding file upload.
	2	File Upload Commit. Writes the upload data to its final storage destination.
	3	File Upload Data. Actual file upload data packet.

Note that the boot shell only supports a subset of the nominal mode file upload telecommand set. Once the File Upload Commit command has been accepted, the boot shell writes the contents of the temporary RAM file upload buffer into the final storage destination and reports any errors encountered during the write operation.

The boot shell only recognizes a limited set of File ID values for the File Upload Commit telecommand. The device and directory numbers for the File ID parameter must both be ‘0’. The file number portion of the File ID parameter enumerates the various directly mapped boot SDRAM and SIB EEPROM file regions.

2.2.2 Memory Management Commands

The primary boot shell supports the dumping and loading of all on-board memory though a small set of memory management commands. While memory dumps may be one of the more common primary boot shell operations, the memory load capability is fairly limited and is intended only for critical error situations.

All memory management telecommands arrive as a single CCSDS telecommand packet. A single telecommand APID value, 0x644, has been reserved to designate memory management telecommands. The commands accepted by the boot shell are enumerated by function code in the table below.

Figure 17 – Boot Shell Memory Management Telemcommands

APID	Function Code	Description
0x644	0	Memory Data Dump. Dumps the contents of a range of memory into a series of telemetry packets.
	1	Memory Dump Cancel. Cancels the current memory dump.
	2	PCI Device Header Dump. Dumps the selected PCI device header.
	3	Processor Register Dump. Dumps the CPU internal registers.
	4	Memory Write. Updates memory contents at the selected address.
	5	PCI Device Header Write. Writes a value to a PCI device header location.
	6	Processor Register Write. Writes values to CPU internal registers.

Note that the boot shell only supports a subset of the nominal memory management telecommand set.

2.2.3 SC Ancillary Messages

The boot shell recognizes all of the ancillary telecommand packets that the SC regularly delivers to the GLAST instruments. The SIANCILLARY telecommand, however, is the only one that the boot shell actively processes. A telecommand APID value, 0x701, has been reserved to designate SC ancillary telecommands, as shown in the table below.

Figure 18 – SC Ancillary Telecommands

APID	Function Code	Description
0x701	1	SC Attitude (SIATTITUDE). Contains latest SC attitude information.
	2	SC Ancillary (SIANCILLARY). Contains latest SC position and mode.
	3	SC Time (SITIMETONE). Contains synchronization time information for GPS PPS.

2.3 Telemetry Output

The boot shell is able to output housekeeping (HKP) telemetry. The boot shell uses this telemetry both as a keep-alive heartbeat and as a way of reporting operational status and error reports. The boot shell telemetry output is constrained to a single, fixed size CCSDS telemetry packet, with a fixed application ID value of 0x200. Because of the limited size of this packet, only a small amount of information can be downlinked by the boot shell.

For the SIU, a new HKP packet is prepared when the 1553 driver indicates that the last telemetry block has been transmitted to the SC. This occurs at a rate of 4 packets per second. On the EPU, the PBC automatically sends a new HKP packet every 250 ms, which also results in a rate of 4 packets per second.

The boot shell sets certain CCSDS HKP telemetry header fields to known values. All boot shell HKP packets are version '0', type 'telemetry', and with secondary header 'true'. The sequence counter increments in a regular manner. The packet length member is always set to '109' to indicate that the LAT HKP telemetry packets is exactly 116 bytes in size total. The telemetry packet secondary header consists of two 32-bit timestamp counters. For SIU, the seconds counter is always set to the same value as the seconds field in the most recently received SIANCILLARY telecommand packet, and the sub-seconds value is always set to '0'.

The boot shell HKP packets contain a boot software mode indicator, which provides status on the current state of the boot code, as shown in the figure below.

Figure 19 – Primary Boot Code Modes

Mode Value	Description
0	The CPU has powered and is initializing, but is not yet ready to accept commands.
1	<No longer used>
2	The boot code is in interactive mode and awaiting commands.
3	The boot code has received a File Upload Start command and is waiting for the upload data packets of the file to arrive. The boot code remains in this mode until the file upload is complete or has been canceled.
4	The boot code has received a File Upload Commit command and is writing the file upload data into memory.
5	Reserved.
6	The boot code is booting the RTOS.

The primary boot shell is able to include a portion of a memory dump within each output telemetry packet. There are two types of memory dumps: the foreground memory dump and the background memory dump.

The foreground memory dump commences as the result of a Memory Data Dump, a PCI Device Header Dump, or a Processor Register Dump telecommand. The memory dump continues in the HKP telemetry output until either the requested amount of memory has been dumped or a Memory Dump Cancel telecommand is received by the boot shell.

The background memory dump is active at any time the foreground memory dump is not active. The background memory dump continuously dumps the contents of the boot diagnostics area (address 0000FF80 -> 0000FFFF).

The SIU HKP telemetry packets are sent as the first packet in a GLAST 1553 telemetry data block. The telemetry block output from the SIB remote terminal contains only this HKP packet, with the remainder of the telemetry block set to 0. The SC bus controller attempts to read a new telemetry block at a rate of 4 Hz. The SIB 1553 polled mode driver is responsible for managing the telemetry block flow control counter at the head of the telemetry blocks.

3 Diagnostics and Errors

The LAT boot code recognizes three basic types of errors: startup, operational, and exception. Startup errors are those errors encountered during the startup process that cannot be reported immediately because the command and telemetry boot shell has not yet started. Operational errors are those encountered during command operations and are reported in the boot code 1553 telemetry output. Exceptions are those errors that escape software detection but are recognized by a hardware component, such as the exception conditions defined by the PPC processor.

3.0 Diagnostics

The Primary Boot Code is able to report a small set of diagnostic information that is recorded during the reset process. This information is in addition to the operational errors reported in the telemetry output packet. The boot code diagnostics information is kept in a reserved 128-byte area at address 0000FF80 in SDRAM. Both the primary and secondary boot codes use this area to report critical errors that they may not be able to report in the normal telemetry channels. Both the LAT boot code and LAT applications may share this area to report diagnostic information that must be preserved across reboots. Figure 20 shows the layout of the Boot Diagnostics area.

Figure 20 – SDRAM Boot Diagnostics Area

Offset (bytes)	Mnemonic	Description	Data Source
0x00	DIAGS_PRIM_BOOT_FLAGS	Primary Boot Flags. Described in Figure 21.	Set during reset initialization or by reboot request.
0x04	DIAGS_SEC_BOOT_FLAGS	Secondary Boot Flags. Described in Figure 22.	Set by PBC from parameters specified in the Boot RTOS Execute Telecommand.
0x08	DIAGS_EXC_COUNT_OFF	Counter of exceptions that have occurred. Reset on cold start.	PBC, SBC, or application exception vector.
0x0C	DIAGS_EXC_VEC	Exception Vector of most recent exception.	PBC, SBC, or application exception handler.
0x10	DIAGS_EXC_SRR0_REG	SRR0 Register value at most recent exception.	PBC, SBC, or application exception handler.
0X14	DIAGS_EXC_SRR1_REG	SRR1 Register value at most recent exception.	PBC, SBC, or application exception handler.
0X18	DIAGS_EXC_DAR_REG	DAR Register value at most recent exception.	PBC, SBC, or application exception handler.
0X1C	DIAGS_EXC_DSISR_REG	DSISR Register value at most recent exception.	PBC, SBC, or application exception handler.
0x20	DIAGS_PCI_STATUS2_REG	PCI Status 2 Register value at most recent exception.	PBC, SBC, or application exception handler.
0x24	DIAGS_MEM_STATUS_REG	Memory Status Register value at most recent exception.	PBC, SBC, or application exception handler.
0x28	DIAGS_EXC_TASK_ID	Task ID at most recent exception.	PBC, SBC, or application exception handler.
0x2C	DIAGS_MT_RESULTS_1	First Pass Memory Test Results.	Reset initialization.

0x30	DIAGS_MT_ADDR_1	First Pass Memory Test Failure Address.	Reset initialization.
0x34	DIAGS_MT_ACTUAL_1	First Pass Memory Test Failure Data.	Reset initialization.
0x38	DIAGS_MT_RESULTS_2	Second Pass Memory Test Results.	Reset initialization.
0x3C	DIAGS_MT_ADDR_2	Second Pass Memory Test Failure Address.	Reset initialization.
0x40	DIAGS_MT_ACTUAL_2	Second Pass Memory Test Failure Data.	Reset initialization.
0x44	DIAGS_SEC_BOOT_FAIL_ERR	Secondary Boot Error Code.	Secondary boot.
0x48	DIAGS_SEC_BOOT_FAIL_IDX	Secondary Boot Error Index.	Secondary boot.
0x4C	DIAGS_SEC_BOOT_FAIL_DATA	Secondary Boot Error Data.	Secondary boot.
0x50	DIAGS_BIST_750	RAD750 BIST Results.	Reset initialization.
0x54-0x5C	DIAGS_BIST_PPCI	PPCI BIST Results.	Reset initialization.
0x5C	DIAGS_MT_COUNT	Number of SDRAM bytes tested.	Reset initialization.
0x60-0x7C	DIAGS_APP_INFO	Application Information.	Secondary boot or application.

The Primary Boot Flags value is set to direct the operation of the Primary Boot Code and to provide status from the reset initialization stage of the PBC. The bit fields are shown below (bit 31 is LSB).

Figure 21 – Primary Boot Flags

Bits	Mnemonic	Description	Data Source/Destination
0-1		Reserved.	
2	DIAGS_PBF_HARD_RESET	Adopted from the standard VxWorks boot flags. When set, it indicates that the CPU was reset by the hardware and the hardware reset cause flags (bits 8 – 11) contain the cause of the hardware reset.	Set during PBC reset initialization.
3	DIAGS_PBF_BOOT_FAST	Adopted from the standard VxWorks boot flags. When set, it indicates that the PBC should immediately load the default RTOS image and start the secondary boot process.	Input to PBC boot shell or reset to '0' by PBC after a hardware reset.
4-7		Reserved.	
8	DIAGS_PBF_JTAG_RESET	A hardware reset cause flag indicating that the CPU board was reset by the external JTAG master interface.	Set during PBC reset initialization based on data in the BIST status register.
9	DIAGS_PBF_PPCI_RESET	A hardware reset cause flag indicating that the CPU board was reset by the PPCI software reset command.	Set during PBC reset initialization based on data in the BIST status register.
10	DIAGS_PBF_PPCI_ERR_RESET	A hardware reset cause flag indicating that the CPU board was reset by an internal error in the PPCI bridge chip. This is also set in the event of an EMC watchdog timeout.	Set during PBC reset initialization based on data in the BIST status register.
11	DIAGS_PBF_POR_RESET	A hardware reset cause flag indicating that the CPU board was reset by the external power-on reset signal.	Set during PBC reset initialization based on data in the BIST status register.
12-15		Reserved.	

16	DIAGS_PBF_EXC_RESET	The CPU has been rebooted as the result of an exception.	Set by exception vector or reset to '0' by PBC after a hardware reset.
17	DIAGS_PBF_SEC_RESET	The CPU has been rebooted as the result of a secondary boot process critical error.	Set by SBC or reset to '0' by PBC after a hardware reset.
18	DIAGS_PBF_APP_RESET	The CPU has been rebooted by the application code. The DIAGS_APP_INFO may contain additional information.	Set by application code or reset to '0' by PBC after a hardware reset.
19	DIAGS_PBF_PRI_RESET	The CPU has been rebooted as the result of a PBC critical error or ground command.	Set by PBC or reset to '0' by PBC after a hardware reset.
20-23	DIAGS_PBF_MEM_SPARE_SELECT	Select a spare memory column. If the DIAGS_PBF_MEM_SPARE_ENABLE flag is set, the PBC startup code uses the value in this field to select a spare RAM column before running memory tests.	Set by application code before warm restart.
24	DIAGS_PBF_MEM_ERROR	Indicates a memory test failure.	Set during PBC reset initialization.
25-26		Reserved.	
27	DIAGS_PBF_MEM_SPARE_ENABLE	Enable a spare memory column. If set, the PBC startup code enables a spare RAM column before running memory tests.	Set by application code before warm restart.
28	DIAGS_PBF_DIAG_MEM_TEST	When set, a memory test is run on the boot diagnostic region of SDRAM.	Set by application code before warm restart or by PBC during cold reboot.
29	DIAGS_PBF_BOOT_MEM_TEST	When set, a memory test is run on the LOW and HIGH boot regions of SDRAM.	Set by application code before warm restart or by PBC during cold reboot.
30	DIAGS_PBF_RTOS_MEM_TEST	When set, a memory test is run on the RTOS region of SDRAM.	Set by application code before warm restart or by PBC during cold reboot.
31	DIAGS_PBF_RES_MEM_TEST	When set, a memory test is run on the reserved region of SDRAM.	Set by application code before warm restart.

The Secondary Boot Flags are set by the Primary Boot Code to direct the operation of the secondary boot process. The bit fields are shown below (bit 31 is LSB).

Figure 22 - Secondary Boot Flags

Bits	Mnemonic	Description
0 – 1	DIAGS_SBF_RTOS_SOURCE	RTOS image source flag indicating the location from which the RTOS executable should be taken when starting the secondary boot process. 0 = RTOS image should be sourced from RAM temporary area 1 = RTOS image should be sourced from SIB EEPROM upper bank boot partition 2 = RTOS image should be sourced from SIB EEPROM lower bank boot partition.
2 – 3	DIAGS_SBF_MODO_SOURCE	Flags indicating the location from which the secondary boot process should source module 0. 0 = SSB Module 0 should be sourced from RAM temporary area 1 = SSB Module 0 should be sourced from SIB EEPROM upper bank boot parition 2 = SSB Module 0 should be sourced from SIB EEPROM lower bank boot parition
4 – 5	DIAGS_SBF_MOD1_SOURCE	Flags indicating the location from which the secondary boot process should source module 1. 0 = SSB Module 1 should be sourced from RAM temporary area 1 = SSB Module 1 should be sourced from SIB EEPROM upper bank boot partition 2 = SSB Module 1 should be sourced from SIB EEPROM lower bank boot partition
6 – 15		Reserved.
16 – 31		Reserved for use by the secondary boot code (SBC).

The exception members of the boot diagnostics region (Exception Vector, Exception SRR0, etc.) are set when the CPU is rebooted as the result of an exception. These members contain valid values only when the DIAGS_PBF_EXC_RESET Primary Boot Flag is set.

In certain error situations, the EMC is called upon to reset the RAD750. When this occurs, the EMC reads various PPCI error status registers and stores their values in the boot diagnostics area. Figure 23 summarizes the affected locations within the boot diagnostic area. Subsequent figures describe the details of these register values.

Figure 23 – EMC Use of Boot Diagnostics Area

Offset (bytes)	Mnemonic	Description
0x08	DIAGS_EXC_COUNT_OFF	Set to 0.
0x0C	DIAGS_EXC_VEC	EMC Exception Vector in the format FFFF000v, where 'v' is the vector number (0-7).
0x10	DIAGS_EXC_SRR0_REG	Value of Vector Interrupt Status register at address Bf88008C.
0X14	DIAGS_EXC_SRR1_REG	Value of Misc Interrupt Status register at address BF880094.
0X18	DIAGS_EXC_DAR_REG	Value of Memory Error Log 1 register at address BF8000C8.
0X1C	DIAGS_EXC_DSISR_REG	Value of Memory Error Log 2 register at address BF8000DC.
0x20	DIAGS_PCI_STATUS2_REG	Value of PCI Status 2 register at address BF870050.
0x24	DIAGS_MEM_STATUS_REG	Value of Memory Status register at address BF8000C0.
0x28	DIAGS_EXC_TASK_ID	Value of MCP Collection register at address BF88002C and 60X Error Status register at address BF810010.

Figure 24 – EMC Exception Info at Offset 0x0C (Exception Vector)

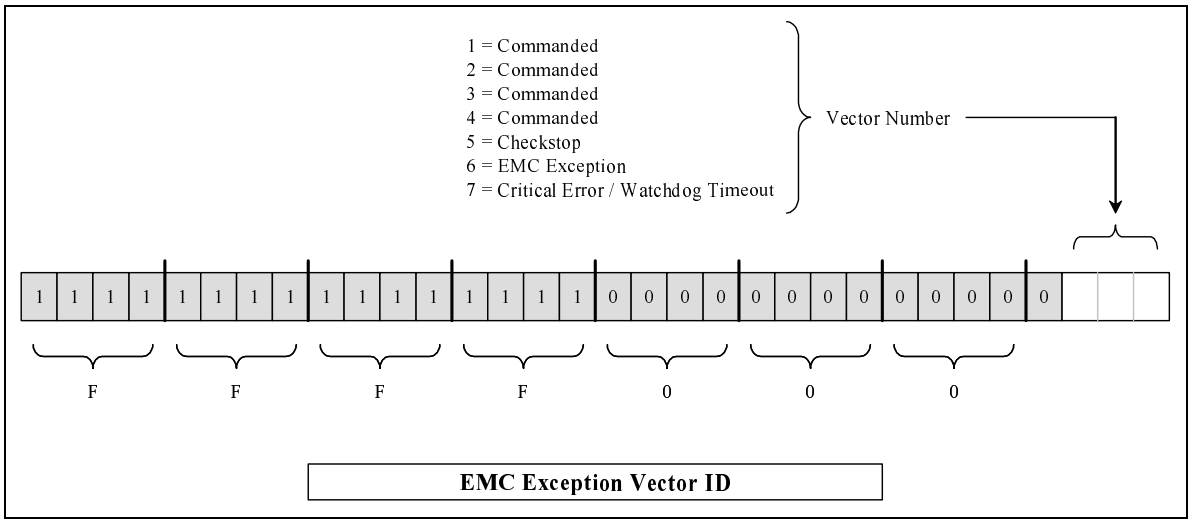


Figure 25 – EMC Exception Info at Offset 0x10 (SRR0)

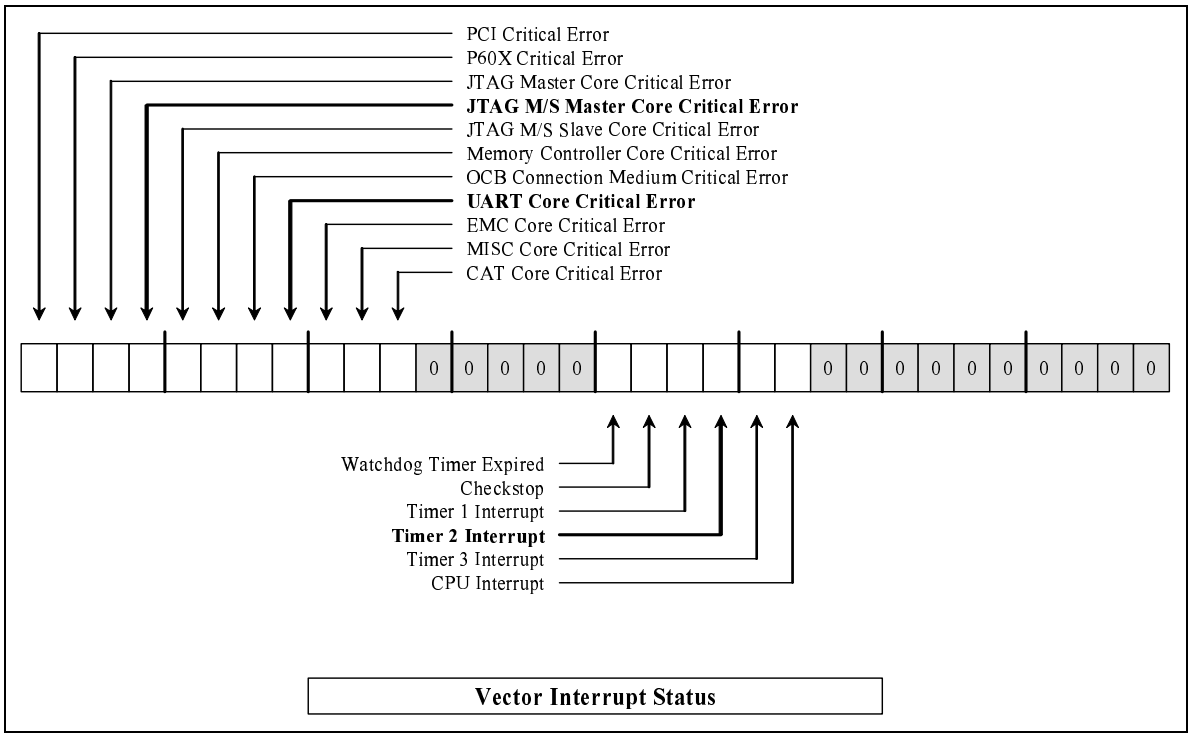


Figure 26 – EMC Exception Info at Offset 0x14 (SRR1)

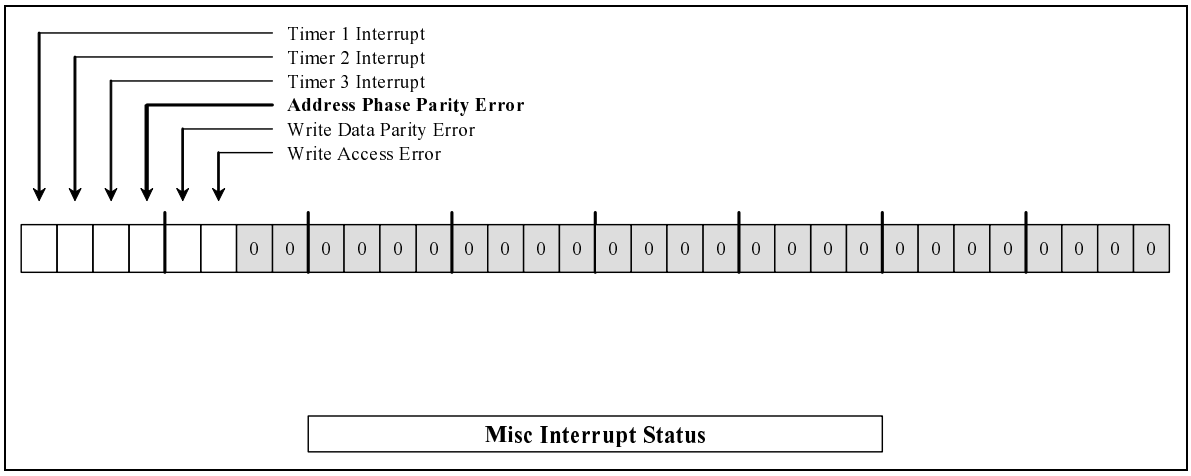


Figure 27 – EMC Exception Info at Offset 0x18 (DAR)

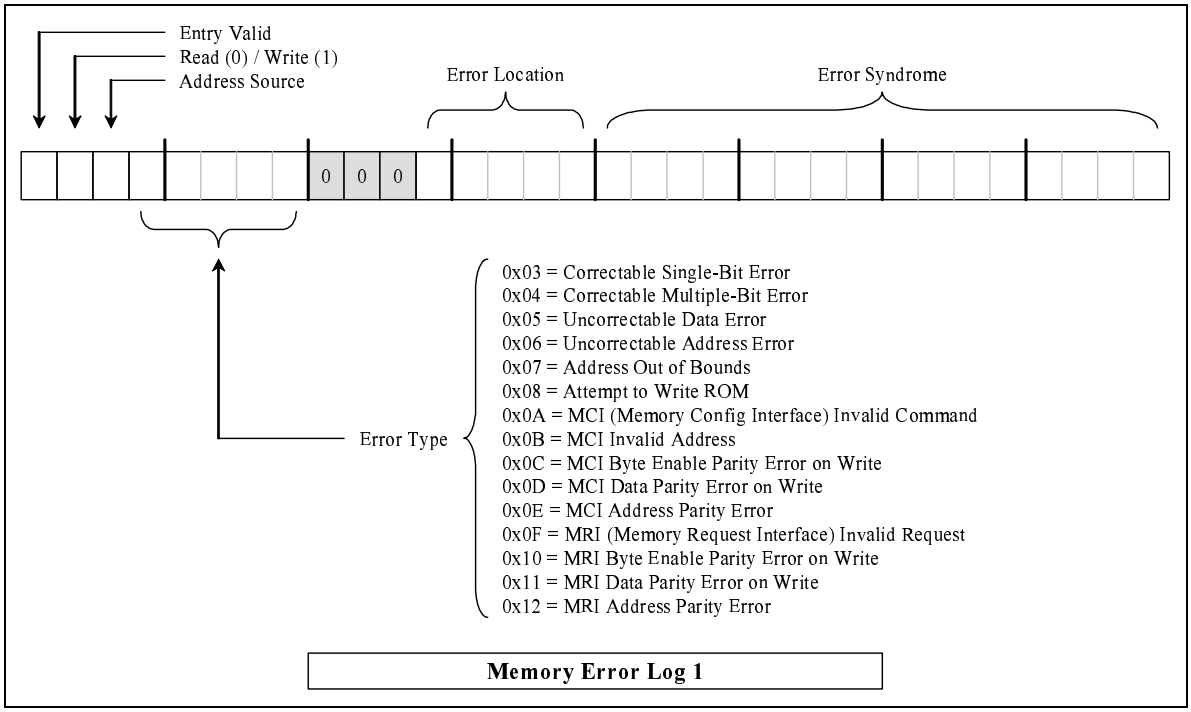


Figure 28 – EMC Exception Info at Offset 0x1C (DSISR)

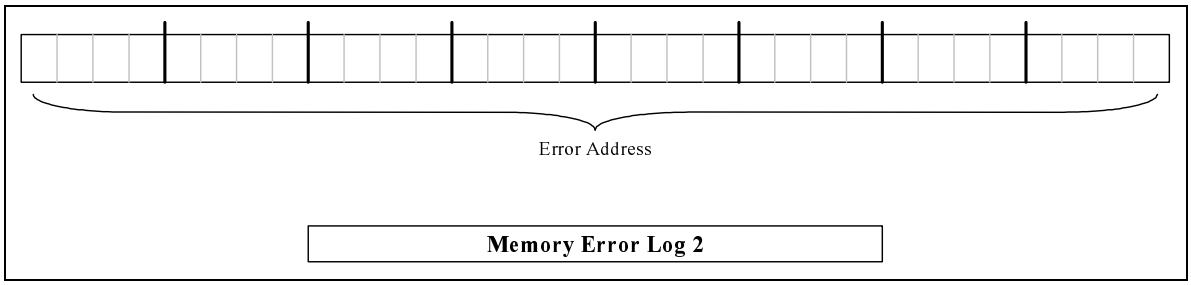


Figure 29 – EMC Exception Info at Offset 0x20 (PCI Status 2)

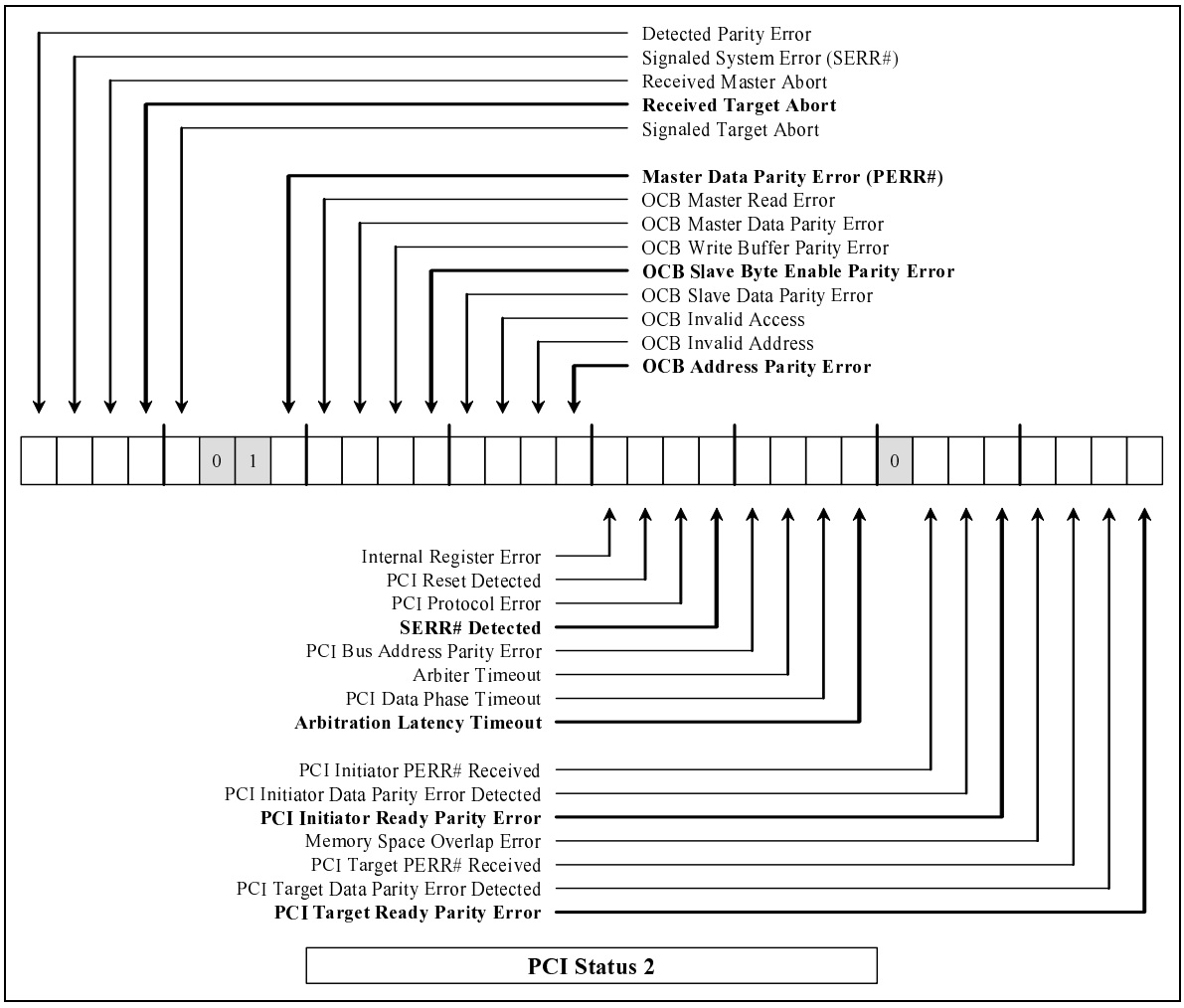


Figure 30 – EMC Exception Info at Offset 0x24 (Mem Status)

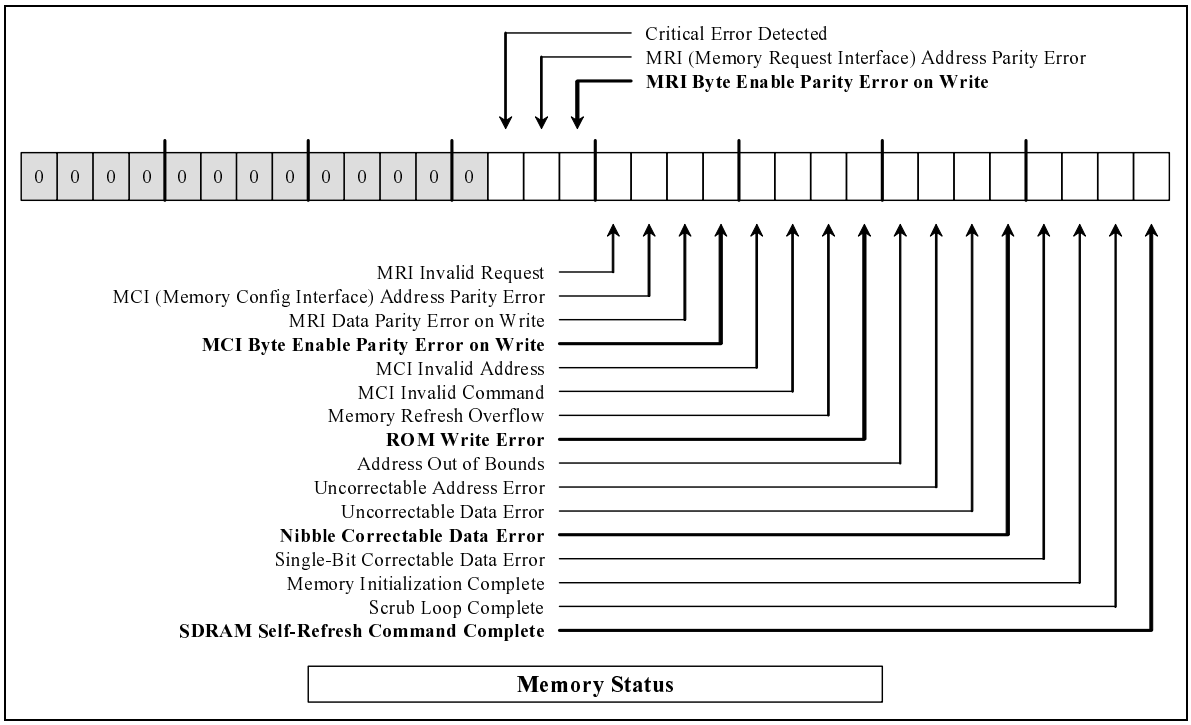
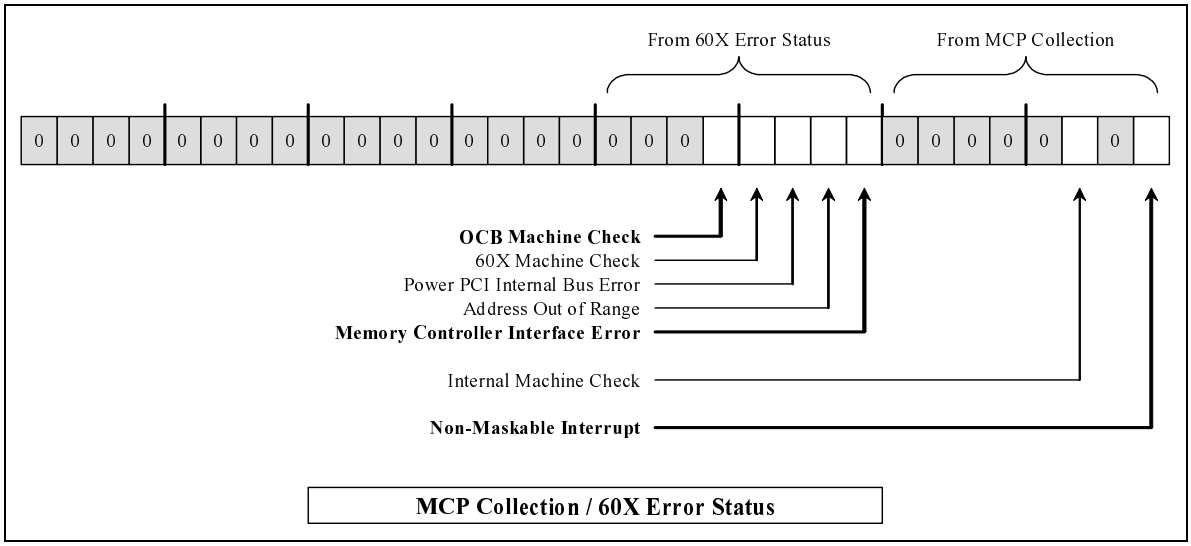


Figure 31 – EMC Exception Info at Offset 0x28 (Task ID)



Results of the memory test are included in the boot diagnostics region. Data is included for a first pass and an optional second pass. The contents of the second pass are populated only if a failure is found when a spare memory column was enabled. This is usually the case only when a failure is detected during the first pass of the memory test. If a spare column was enabled via the Primary Boot Flags, however, only a single pass of the memory test will be executed and its results will be stored in the second pass results region.

The types of memory test data that are included for each test pass are (1) a result word, (2) the address of the first error detected, and (3) the "read-back" data that was in error. The format of the result word is shown in Figure 32 (bit 31 is the LSB). It includes the result code for each of the five memory areas tested (see Figure 4), the index of the column containing a failure (1-8 or LSB-MSB), and the number of errors reported by the PPCI memory controller. The memory test result codes are described in Figure 33.

Figure 32 - Memory Test Result Word

Bits	Description
0-3	Result code of Low Boot Region test.
4-7	Result code of High Boot Region test.
8-11	Result code of Boot Diagnostics Region test.
12-15	Result code of Reserved/Application Region test.
16-19	Result code of RTOS Region test.
20-23	Index of column that was determined to be bad and was swapped out as part of remediation.
24-31	Number of errors reported by the PPCI memory controller.

Figure 33 - Memory Test Result Codes

Code	Description
0	Memory test was not run.
1	Memory test passed.
2	Memory test failed in final (‘zeroes’) walking bit step.
3	Memory test failed in first (‘ones’) walking bit step.
4	Memory test failed in second (‘twos’) walking bit step.
5	Memory test failed in third (‘fours’) walking bit step.
6	Memory test failed in fourth (‘eights’) walking bit step.
7	Memory test failed in address test step.
8	Memory controller indicated error.
9-13	Reserved.
14	Memory was initialized but not tested.
15	Memory test was skipped.

3.1 Errors

Primary Boot Code errors fall into two broad categories, critical startup errors and operational errors. Critical startup errors are those that occur before telemetry communications are available. Operational errors are those that occur during boot shell operations, usually as the result of a telecommand. Operational errors are reported in the boot shell HKP telemetry.

3.1.0 Critical Startup Errors

Critical startup errors occur when the Primary Boot Code initialization process encounters a problem that would prevent successful boot before the boot primary communication path is established. In these cases, the LAT Primary Boot Code utilizes two of the PID channels on the RAD750 board to report the startup failure.

Figure 34 - Critical Startup Error PID Settings

PID 5 Value	PID 6 Value	Status
0	0	No error.
0	1	Memory test failure.
1	0	The primary communication channel (1553 or LCB) could not be initialized.
1	1	Memory test failure and primary communication channel initialization failure.

When the Primary Boot Code reports a critical startup error, the PID lines are set to the appropriate status values. This guarantees that the error condition status is available for users to examine.

3.1.1 Operational Errors

Operational errors that occur during Primary Boot Code normal operation are reported in the HKP telemetry packet. Each HKP packet can report one error, which is encoded as a 32-bit word in the packet. The error report word maintains its value for examination until a Boot Error Dump telecommand is received. When the boot shell receives this command, it removes the next error report from a PBC internal queue and places it in the next outgoing HKP telemetry packet. If no new errors are available to report, the error report word in the HKP packet is set to 00000000.

3.2 Exceptions

The PBC implements exception handlers for both the PPC and EMC processors. All possible exceptions on both machines have an exception handler, even though the same exception handler code may be used for more than one exception.

3.2.0 EMC Exceptions

Exception handling on the EMC processor is different from CPU exceptions in that these exceptions exist in SUROM and are the exceptions that must execute while the CPU is in both the boot and operational modes. These exceptions are the only piece of PBC code that is executed while not in boot mode.

The EMC processor has 8 exception vectors:

Figure 35 - EMC Exception Vectors

Vector	Source
Vector 0	Operation request from the CPU.
Vector 1	Reset request from the CPU.
Vector 2	No assigned use.
Vector 3	No assigned use.
Vector 4	No assigned use.
Vector 5	Checkstop indication from the CPU.
Vector 6	EMC invalid instruction or access error. Unmaskable.
Vector 7	Critical error or watchdog timer expired. Unmaskable.

Vector 0 allows the PPC to request the EMC processor to perform operations. The only EMC operation used by the LAT FSW is the clock-set operation. This operation allows the PPC to change the processor clock rate. This code is implemented by BAE to support the sysSetCpuClk() call. For this code to work, data must be shared between the EMC and the PPC. This data is stored in the 128-byte EMC Parameter Region of RAM shown in Figure 10.

Vector 1 allows the PPC to request the EMC to reset the RAD750. When invoked, the EMC saves the state of certain PPCI registers in the boot diagnostics area before resetting the board. Figure 23 describes the registers that are saved.

Vector 5 is invoked when the PPC indicates a checkstop condition. When activated, the EMC saves the state of certain PPCI registers in the boot diagnostics area before resetting the board. Figure 23 describes the registers that are saved.

Vector 6 is an unmaskable exception to the EMC. It is active if an illegal EMC instruction is encountered. When this vector is activated, the EMC saves the state of certain PPCI registers in the boot diagnostics area before resetting the board. Figure 23 describes the registers that are saved.

Vector 7 is an unmaskable exception to the EMC. It is active if the PPC watchdog timer expires or a "critical" error is encountered in the PPC or PPCI bridge. When this vector is activated, the EMC saves the state of certain PPCI registers in the boot diagnostics area before resetting the board. Figure 23 describes the registers that are saved.

The LAT flight software does not use vectors 2-4. If any of these vectors are activated, the EMC saves the state of certain PPCI registers in the boot diagnostics area before resetting the board. Figure 23 describes the registers that are saved.

3.2.1 PPC Exceptions

Exception handlers on the PPC processor can exist in either the SUROM or RAM. It is assumed that when in operational mode, the application code will provide exception handlers in RAM for all exceptions.

3.2.1.0 Startup Exception Handler

While the PBC is running the reset initialization code, one exception handler is used to process all PPC exceptions. This exception handler is installed for all exception vectors located in the RAD750 SUROM (starting at address FFF00200). This handler stores the following information into the assigned fields in the boot diagnostics SDRAM region:

- Exception vector number,
- PPC SRR0 register contents,
- PPC SRR1 register contents,
- PPC DAR register contents,
- PPC DSISR register contents,
- PPCI PCI status register contents and
- PPCI memory status register contents.
- PPC LR (link register) value – stored in APP_INFO[0] location.
- PPC SP (stack pointer) value – stored in APP_INFO[1] location.

The handler also increments an exception counter kept in the boot diagnostics region. The handler then advances the exception program counter to the next instruction and returns. This gives the PBC a chance to "plow though" to the boot shell mode in a transient error situation.

3.2.1.1 Boot Shell Exception Handler

When the PBC runs the boot shell and has established telemetry communications, exception handlers are installed into RAM. A single exception handler is installed for all exception vectors located in the RAD750 RAM (starting at address 00000200). The RAM version of the Primary Boot Code exception handler stores the same information into the boot diagnostics SDRAM region as is stored by the startup exception handler.

The handler also increments an exception counter kept in the boot diagnostics region. The Primary Boot Code then sets the software exception flag in the Primary Boot Flags member, clears all other Primary Boot Flags, and restarts itself from the warm boot address FFF00104 with the boot type code set to "Exception Start" (see Figure 9 for more detail).

4 Application-Level Support

The PBC package provides a small number of functions that are used at the application level. These functions are mainly involved with forwarding telecommands from an SIU to an EPU and telemetry from the EPUs to the spacecraft and SSR. The primary interface to the application level support is through the two functions PBC_initialize() and PBC_shutdown().

4.0 PBC_initialize()

The PBC_initialize() function allocates resources used by the PBC package and sets configuration values to their default state. It accepts two parameters:

- a pointer to the task description block of the task to which the PBC telecommand handlers should be attached
- the ID of the task to which the PBC telecommand handlers should be attached

These parameters are redundant, in that they provide two methods for specifying the PBC command handler task. If the first parameter is non-NULL, it will be used to attach the PBC command handlers to the corresponding task (and the second parameter will be ignored). If the first parameter is NULL, however, then PBC_initialize() will call an ITC function to translate the second parameter into a task description block pointer, which it will then use to attach the PBC command handlers. Finally, if the first parameter is NULL and the second parameter is not a valid task ID (e.g. -1), then PBC_initialize() will not attach the PBC command handlers to any task. In the latter case, it is assumed that some other application-level code will perform this attachment.

PBC_initialize() returns a MSG status code.

4.1 PBC_shutdown()

The PBC_shutdown() function frees resources allocated by the PBC package. It requires no parameters and it returns a MSG status code.

4.2 PBC Telecommand Handlers

The PBC package provides a handler for each of the Boot telecommands. These handlers accept the telecommands as they arrive on the SIU and forward them to the appropriate EPU. If the current CPU is the target of a Boot telecommand, however, all but the LPBCRESET handler report an error. The LPBCRESET handler, on the other hand, accepts the command and resets the CPU if the current CPU is the target.

4.3 SCP

In an SCP environment, the PBC package provides functions that assist in sending Boot telecommands and displaying Boot Housekeeping telemetry. Table 1 describes these functions.

Table 1 – PBC Package SCP Functions

Function	Description
PBC_dispatchTlm()	Displays the contents of a Boot Housekeeping telemetry packet. This function can be called from within the SCP callback function that is registered to receive telemetry from CTX (currently SCP_dispatchTlm()). Since Boot Housekeeping telemetry arrives at a 4 Hz rate, the PBC_setDisplayLevelTlm() function can be used to specify how often the packets are displayed.
PBC_sendBoot()	Sends the LPBCRTOSEXEC telecommand (see section 6.0.3).
PBC_sendErrDump()	Sends the LPBCERRDUMP telecommand (see section 6.0.2).
PBC_sendReset()	Sends the LPBCRESET telecommand (see section 6.0.1).
PBC_sendStart()	Sends the LPBCSTART telecommand (see section 6.0.0).
PBC_setDisplayLevelCmd()	Changes the value of the internal PBC_displayLevelCmd variable, which determines the amount of information displayed when a Boot telecommand is sent, as follows: 0 = display no information 1 = display command body 2 = display command header and body 3+ = display command header, body, and checksum This function requires a single parameter, which is the new value of the variable. It returns the variable's previous value.
PBC_setDisplayLevelTlm()	Changes the value of the internal PBC_displayLevelTlm variable, which determines how often the contents of the Boot Housekeeping telemetry packet is displayed, as follows: 0 = never display 1 = display when error count values change 2 = display when file upload state or boot software mode change 3 = display when file upload packet count changes 4 = display when anything but the scrub address changes 5+ = display when anything changes For values less than 6, Boot Housekeeping telemetry packets will be displayed no more often than once per second, no matter what values change. This function requires a single parameter, which is the new value of the variable. It returns the variable's previous value.

5 Error Reporting

The PBC package functions support the MSG status reporting system. The following message codes are defined.

Table 2 – PBC Package MSG Codes – Success and Information

MSG Code	Description	Parameters (if any)
SUCCESS	Function succeeded.	
REBOOT	The LPBCRESET telecommand handler is rebooting the CPU.	

Table 3 – PBC Package MSG Codes – Errors

MSG Code	Description	Parameters (if any)
ALLOCMTX	Unable to allocate a mutex.	
BADALLOC	Unable to allocate memory.	• Size of allocation request, in bytes • String describing the intended use of the memory
BADAPID	Internal software error.	
BADATACH	Unable to attach handler callback functions.	String describing the type of callback functions
BADDTACH	Unable to detach telecommand handler functions using ITC_detachApid().	
BADFREE	Unable to free memory.	
BADSTATE	The PBC application-level code is not in the correct state for the requested operation.	• Current state • Expected state
BHDRCHK	An SIB EEPROM bank header checksum was incorrect.	
CMDBDFC	A telecommand was received that is not supported by the PBC in its current mode.	

CMDFAIL	An operation that should have terminated PBC execution (e.g. loading the RTOS or rebooting the CPU), returned to the PBC.	
CMDMODE	A telecommand was not accepted because the PBC was not in the correct mode of operation.	
CMDREJ	A telecommand was rejected because it was not recognized by the PBC or it was corrupted.	
CMDUNSP	A telecommand was received that is not supported by the PBC in its current mode.	
COMBDFID	A file upload telecommand specified an invalid file ID value.	
COMSIZE	A file upload telecommand specified a file size that is too large for the file's destination.	
COMSTATE	The PBC file upload state machine was in an invalid state.	
COMTO	The PBC was unable to commit a file to SIB EEPROM after trying 1000 times.	
CSDSAPID	Unable to set the APID within a CCSDS packet.	MSG status code from CCSDS_pktHdrSetApid()
CSDSHDR	Unable to create a CCSDS packet header.	MSG status code from CCSDS_pktHdrCreate()
CSDSSUM	Unable to insert a checksum into a CCSDS packet.	MSG status code from CCSDS_pktChksumInsert()
EBADSEND	Unable to send a packet with ITC_send().	MSG status code from ITC_send()
EEBADADR	An access to SIB EEPROM was attempted at an invalid address – most likely due to an incorrect memory map table entry.	
EEBDBANK	Internal software error.	
EENOTRDY	Unable to access SIB EEPROM – probably because the PBC EEPROM driver code was not initialized successfully.	
EEUNLOCK	Writing to the SIB EEPROM has already been enabled.	
EEWEFAIL	Unable to enable SIB EEPROM writes.	
EPKTSHRT	A telecommand packet was too short.	• Minimum required length of the packet, in bytes • Actual length of the packet, in bytes

ERRSTCLR	Unable to clear a PPCI error register bit.	
EQIALLOC	Unable to allocate an ITC queue item.	
FILECHK	An RTOS image checksum was incorrect.	
FREEMTX	Unable to destroy a mutex.	
GENERROR	Generic error within an SCP function.	String describing the error
INFLERR	A ZLIB inflation function returned an error.	
INVARGS	An invalid RTOS image source was specified in an LPBCRTOSEXEC command, or an invalid SIB EEPROM bank ID was passed to an internal function.	
LOCKMTX	Unable to lock a mutex.	
MEMST5 – 18	One of the memory status register bits 5-18 was set.	
NOTRDY	The PBC application-level code has not been initialized successfully by a call to PBC_initialize().	
NULLPTR	A NULL pointer value was passed to a function, probably due to an internal software error.	
PCIST0 – 6, PCIST8 – 9, PCIST_10 – _19, PCIST20 – 24, PCIST27 – 29, PCIST30_ – 31_	One of the PCI status 2 register bits 0-6, 8-24, or 27-31 was set.	
PKTSIZE	A telecommand packet was not the correct length.	
RAMINIT	Unable to initialize RAM before loading the RTOS.	
SECRTRN	The secondary boot code entry function returned to the primary boot code.	
SELFAIL	The PBC image selection code detected one or more invalid PBC images.	
SIBFAIL	No recognized SIB board was found.	
UNLCKMTX	Unable to unlock a mutex.	
UNKNOWN	Internal software error.	

6 Appendix A

This appendix contains the formats of the boot telecommands and telemetry.

6.0 Boot Telecommands

Boot telecommands direct the boot operations of the SIU and EPU Primary Boot Code. Table 4 summarizes the boot telecommands.

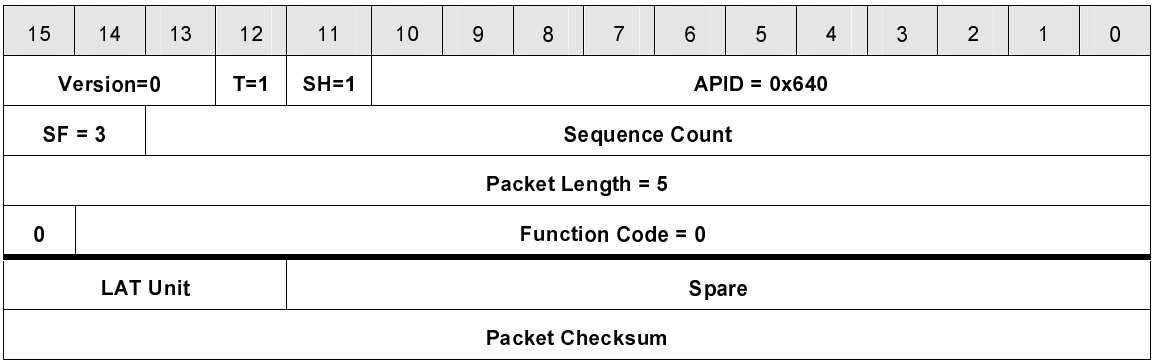
Table 4 – Boot Telecommand Function Codes

APID	Function Code	Description
0x640	0	Boot Start. No-op which simply increments the command counter.
	1	Boot Reset. Forces a reset of the PBC into a known restart state.
	2	Boot Error Dump. Dumps the value of an error word queued by the PBC.
	3	Boot RTOS Execute. Begins execution of an RTOS image and the second-stage boot process.

6.0.0 Boot Start (LPBCSTART)

The Boot Start telecommand provides a harmless no-op command to test the command and telemetry operations of the boot shell.

Figure 36 – Boot Start (LPBCSTART) Telecommand Format



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

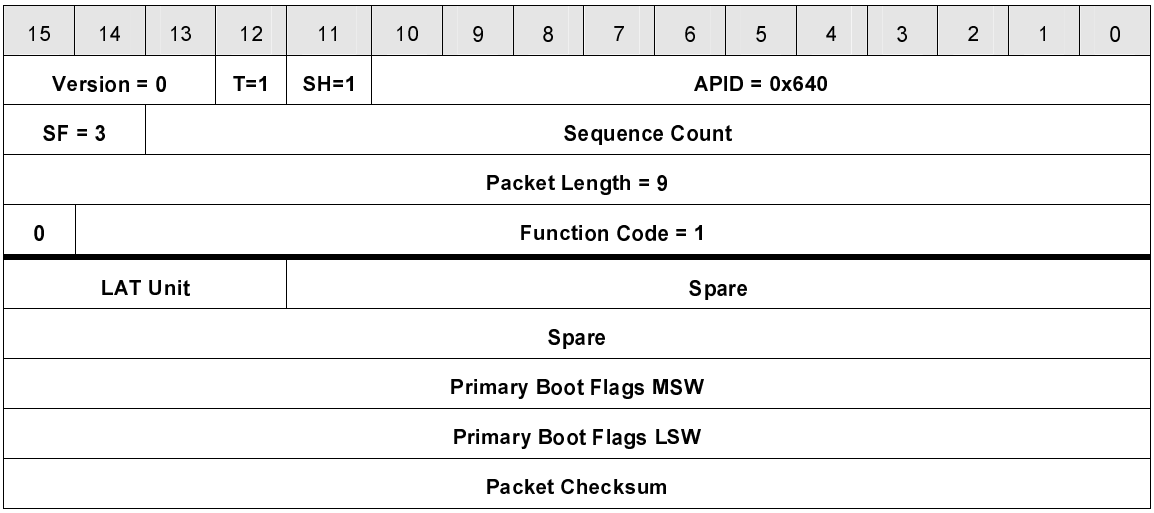
Within the SCP environment, the **PBC_sendStart()** function can be used to send this telecommand. This function requires the following parameter:

- unit: the 'LAT unit' value.

6.0.1 Boot Reset (LPBCRESET)

The Boot Reset telecommand causes the Primary Boot Code to return to the warm-start entry point of the SUROM reset code. The command allows the user to specify the 32-bit Primary Boot Flags value to use upon restart.

Figure 37 – Boot Reset (LPBCRESET) Telecommand Format



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

Primary Boot Flags – A bit field indicating the actions the boot code should take when restarting. See Figure 21 for definitions of this bit field.

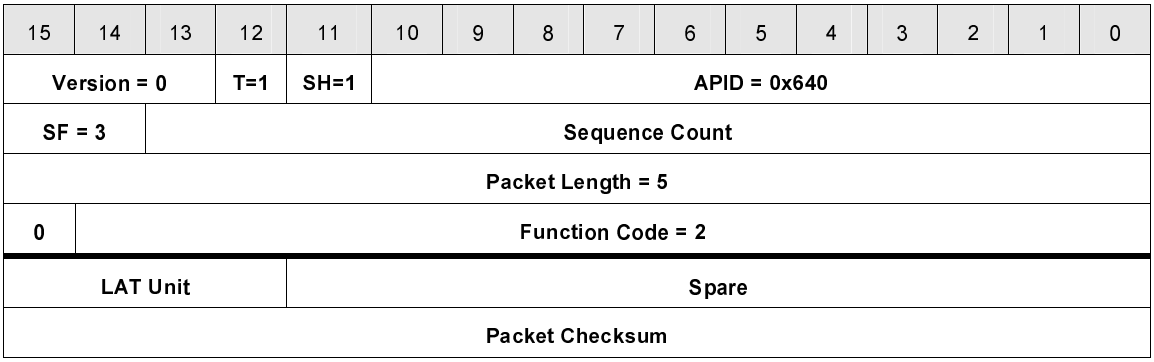
Within the SCP environment, the **PBC_sendReset()** function can be used to send this telecommand. This function requires the following parameters:

- unit: the 'LAT unit' value.
- flags: the 'Primary Boot Flags' value.

6.0.2 Boot Error Dump (LPBCERRDUMP)

The primary boot shell maintains a queue of up to 256, 32-bit error code words that describe error conditions that the PBC has encountered. The error words remain in the queue until forced out into the HKP telemetry packets by this command. This command causes one error word to be sent in the next available HKP telemetry packet. The error code word remains visible in subsequent HKP telemetry packets until this command is issued again. The number of queued error words still outstanding is also contained in the HKP telemetry packet.

Figure 38 – Boot Error Dump (LPBCERRDUMP) Telecommand Format



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

Within the SCP environment, the **PBC_sendErrDump()** function can be used to send this telecommand. This function requires the following parameter:

- unit: the 'LAT unit' value.

6.0.3 Boot RTOS Execute (LPBCRTOSEXEC)

The Boot RTOS Execute command manually boots an RTOS image. The Secondary Boot Flags parameter specifies the source of the RTOS executable image, as well as other information which is passed to the secondary boot code. If the boot shell determines that the source memory region does not contain a valid VxWorks executable image, this command generates an error.

If the RTOS executable file header indicates compression, the RTOS executable image is first inflated using the ZLIB library. The uncompressed executable sections from the RTOS image are then loaded into the RAM area reserved for VxWorks execution. Finally, the boot shell jumps to the entry point of the RTOS executable, ending the execution of the boot shell itself and beginning the secondary boot process. The Boot RTOS Execute command also has the ability to direct the secondary boot process by specifying the location of the second stage boot modules.

Figure 39 – Boot RTOS Execute (LPBCRTSOEXEC) Telecommand Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Version=0			T=1	SH=1	APID = 0x640										
SF = 3			Sequence Count												
Packet Length = 11															
0			Function Code = 3												
LAT Unit					Spare										
Spare															
Secondary Boot Flags MSW															
Secondary Boot Flags LSW															
Packet Checksum															

LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

Secondary Boot Flags – A bit field of secondary boot options. See Figure 22 for a definition of this bit field.

Within the SCP environment, the **PBC_sendBoot()** function can be used to send this telecommand. This function requires the following parameters:

- unit: the 'LAT unit' value.
- rtos: the location of the RTOS image to be booted (0 = RAM, 1 = SIB EEPROM lower bank, 2 = SIB EEPROM upper bank).
- ssb0: the location of the SBC module 0 image to use (0 = RAM, 1 = SIB EEPROM lower bank, 2 = SIB EEPROM upper bank).
- ssb1: the location of the SBC module 1 image to use (0 = RAM, 1 = SIB EEPROM lower bank, 2 = SIB EEPROM upper bank).

The Secondary Boot Flags' value is constructed from the 'rtos', 'ssb0', and 'ssb1' parameters.

6.1 File Upload Telecommands

The file upload telecommands load RTOS images and secondary boot modules into the SIB EEPROM and RAD750 SDRAM. Table 5 summarizes the file upload telecommands accepted by the Primary Boot Code, which is a subset of those available from the FILE package. See the documentation for that package for a more complete description of these commands.

Table 5 – File Upload Telecommand Function Codes

APID	Function Code	Description
0x641	0	File Upload Start. Announce the beginning of a new file upload and provides its total size.
	1	File Upoad Cancel. Cancel an outstanding file upload.
	2	File Upload Commit. Write the upload data to its final storage destination.
	3	File Upload Data. Actual file upload data packet.

6.1.0 File Upload Start

The File Upload Start command starts the file upload process. A new series of File Upload Data packets may be sent after this command successfully completes. A previous upload process must not be in progress when this command is sent, otherwise the PBC will report an error without disturbing the state of the previous upload process.

Figure 40 – File Upload Start Telecommand Format

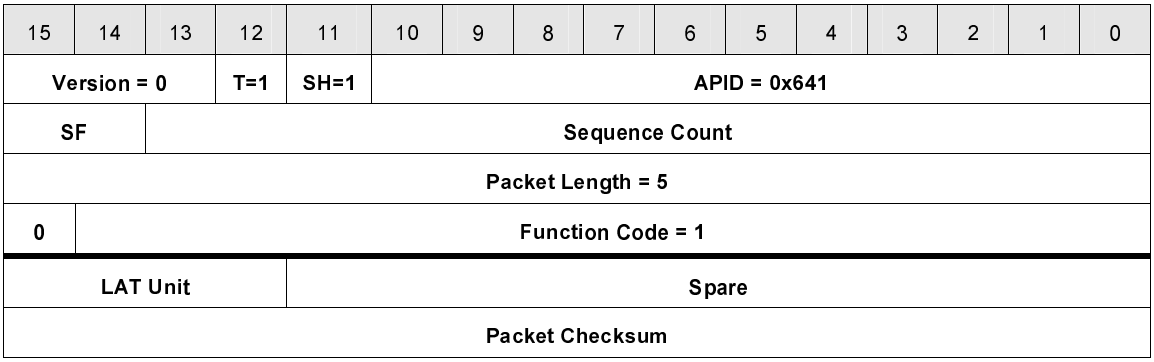
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Version = 0			T=1	SH=1	APID = 0x641										
SF			Sequence Count												
Packet Length = 9															
0	Function Code = 0														
Upload File Size MSW															
Upload File Size LSW															
Packet Checksum															

Upload File Size – The size in bytes of the file that will be uploaded by a subsequent series of File Upload Data commands.

6.1.1 File Upload Cancel

The File Upload Cancel command aborts a file upload process by deleting all data currently held in the file upload buffer and reseting the file upload state machine. This command may be used to cancel an erroneous upload attempt.

Figure 41 – File Upload Cancel Telecommand Format

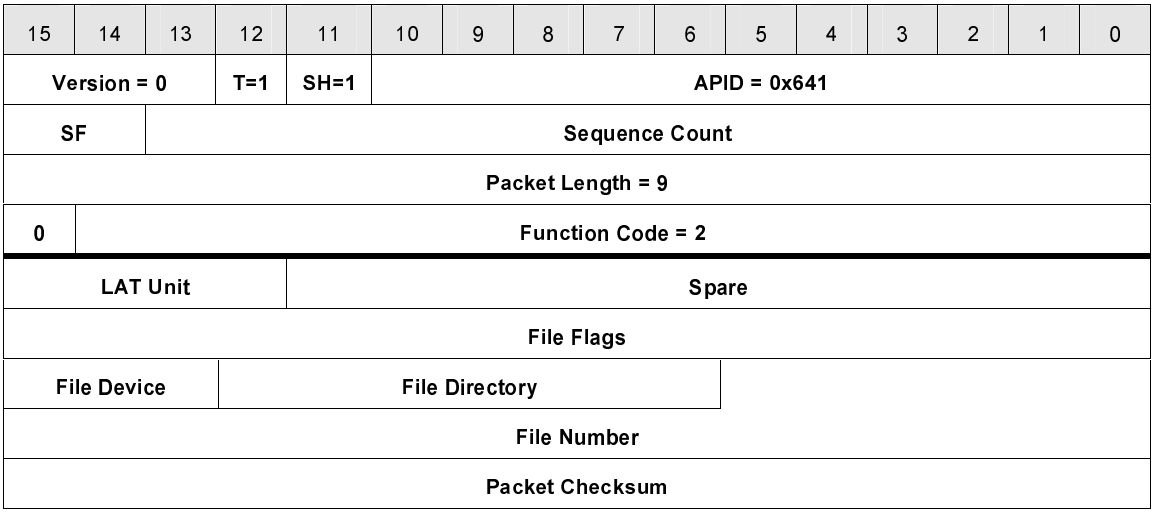


LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

6.1.2 File Upload Commit

The File Upload Commit command copies the file data contents from the file upload buffer to the actual device storage location indicated by the File Device, File Directory, and File Number parameters. All File Upload Data packets must have arrived in good order prior to receiving this command, otherwise this command will fail with an error.

Figure 42 – File Upload Commit Telecommand Format



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

File Flags – A code indicating options for controlling the commit operation.

File Device – A code indicating the file storage device. For the PBC, this must be '0'.

File Directory – A code indicating the file storage directory. For the PBC, this must be '0'.

File Number – A code indicating where the contents of the file upload buffer should be stored, as described in the table below.

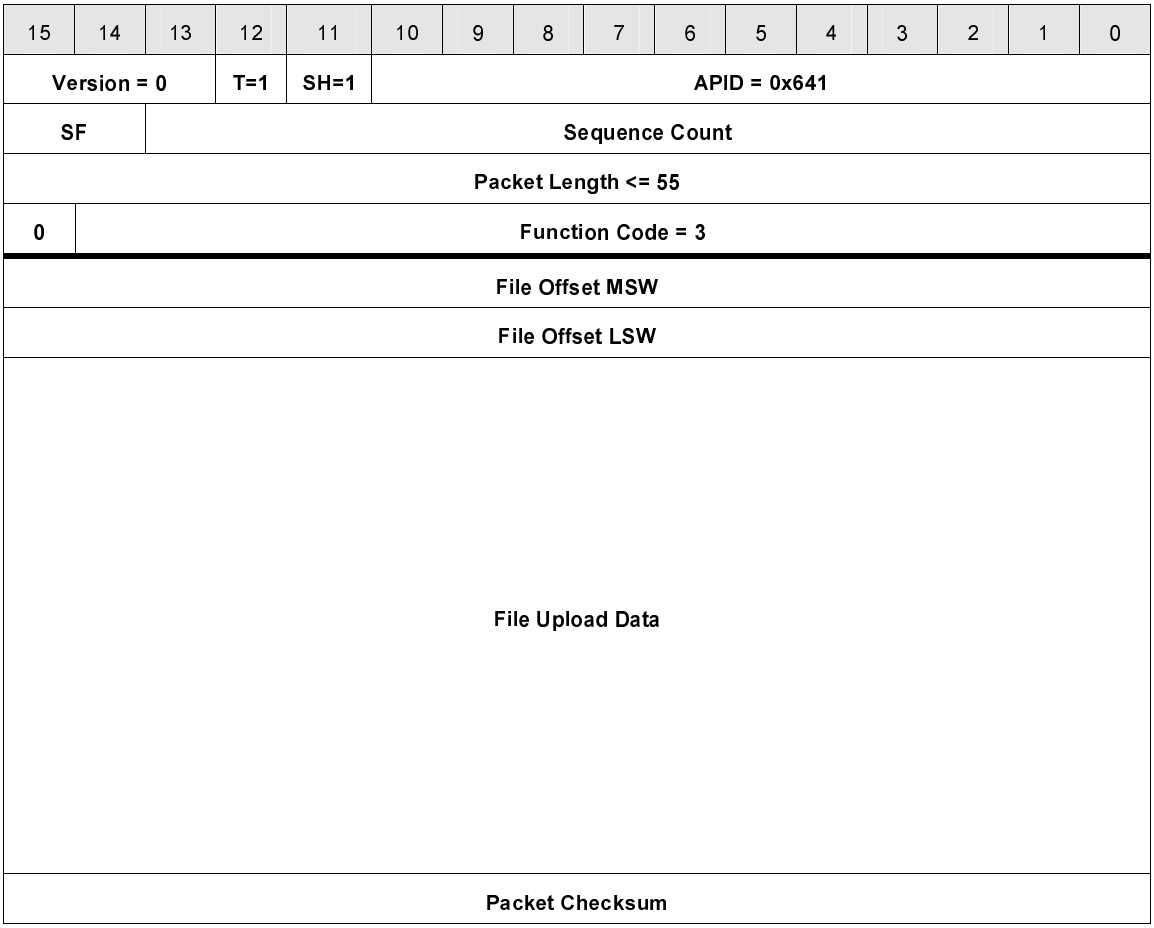
Table 6 – File Upload File Numbers

File Number	Description
0	RAM RTOS buffer.
1	RAM secondary boot module 0 buffer.
2	RAM secondary boot module 1 buffer
3	SIB EEPROM lower bank boot partition RTOS file.
4	SIB EEPROM lower bank boot partition secondary boot module 0 file
5	SIB EEPROM lower bank boot partition secondary boot module 1 file
6	SIB EEPROM upper bank boot partition RTOS file.
7	SIB EEPROM upper bank boot partition secondary boot module 0 file
8	SIB EEPROM upper bank boot partition secondary boot module 1 file

6.1.3 File Upload Data

Each file upload arrives as a series of File Upload Data telecommand packets. The total size of each upload telecommand packet may not exceed 62 bytes. The file data contained within these packets is collected in a temporary RAM upload buffer.

Figure 43 – File Upload Data Telecommand Format



File Offset – The offset in bytes from the beginning of the upload file from which the data in this packet was extracted.

File Upload Data – Up to 48 bytes of data extracted from the upload file.

6.2 Memory Load Telecommands

The memory load telecommands load arbitrary data into various locations within the SIU. These locations include SDRAM, SIB EEPROM, PPCI registers, PCI device headers, and processor registers. Table 7 summarizes the memory load telecommands accepted by the Primary Boot Code, which is a subset of those available from the MEM package. See the documentation for that package for a more complete description of these commands.

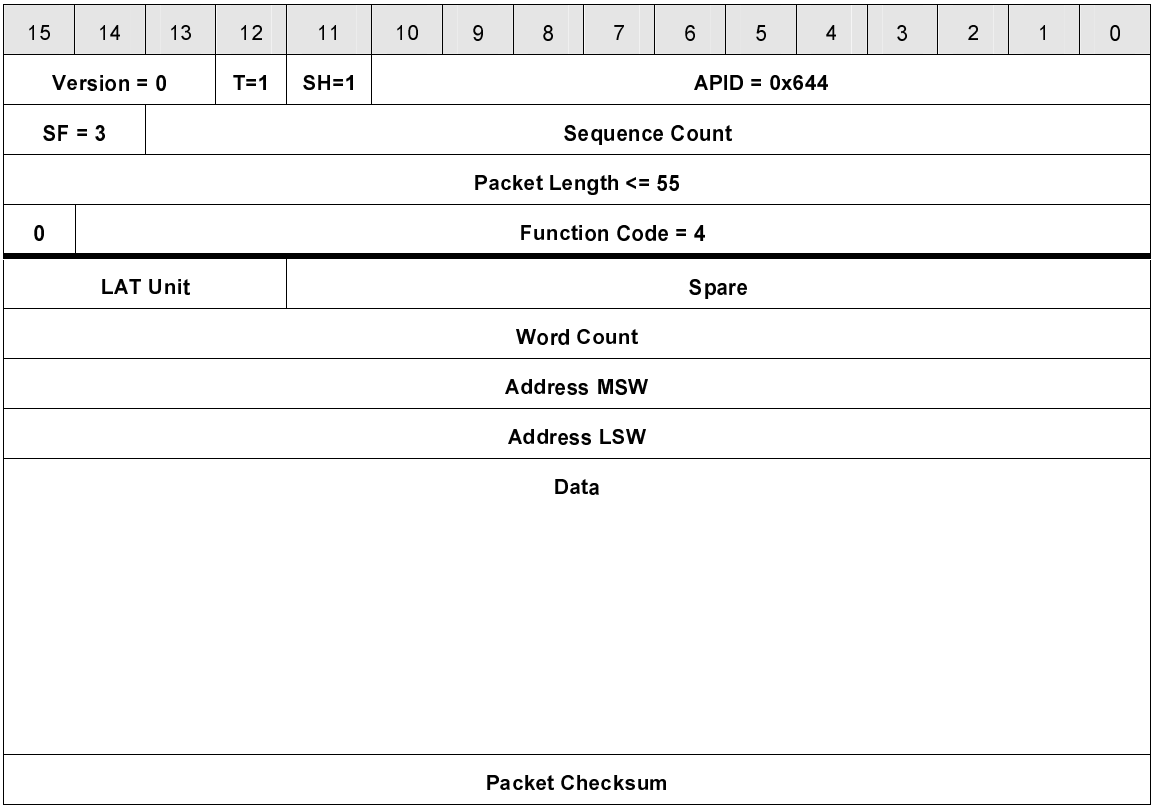
Table 7 – Memory Load Telecommand Function Codes

APID	Function Code	Description
0x644	4	Memory Write. Update memory contents at the selected address.
	5	PCI Device Header Write. Write a value to a PCI configuration header location.
	6	Processor Register Write. Write values to processor registers.

6.2.0 Memory Write

The Memory Write command updates the contents of 32-bit locations within the processor's address space. Due to alignment constraints of the target memory, the Memory Write command may fail with an error if the address or word count parameters violate those constraints. A small memory map is maintained by the PBC to designate the addresses of the legal memory regions and the constraints of those regions.

Figure 44 – Memory Write Telecommand Format



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

Word Count – The number of 32-bit words to write.

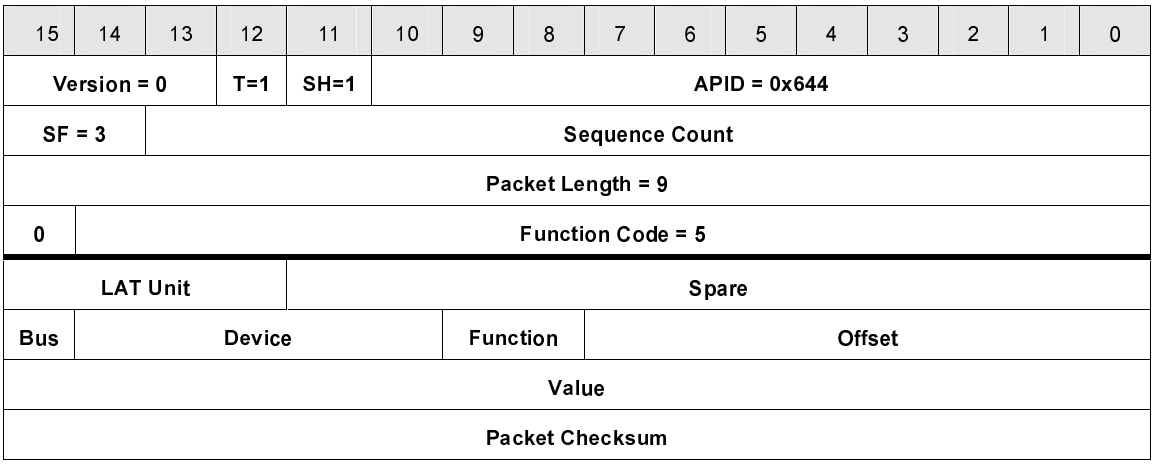
Address – The address at which the data should be written. For the PBC, this address must be aligned to a 32-bit boundary.

Data – Up to 11, 32-bit data values to write.

6.2.1 PCI Device Header Write

The PCI Device Header Write command writes a 16-bit value into the designated PCI configuration header location.

Figure 45 – PCI Device Header Write



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

Bus – PCI Bus Select. For the PBC, this must be '0'.

Device – PCI Device Select. This value will vary based on system configuration (test vs flight).

Function – PCI Device Function Select. For the PBC, this must be '0'.

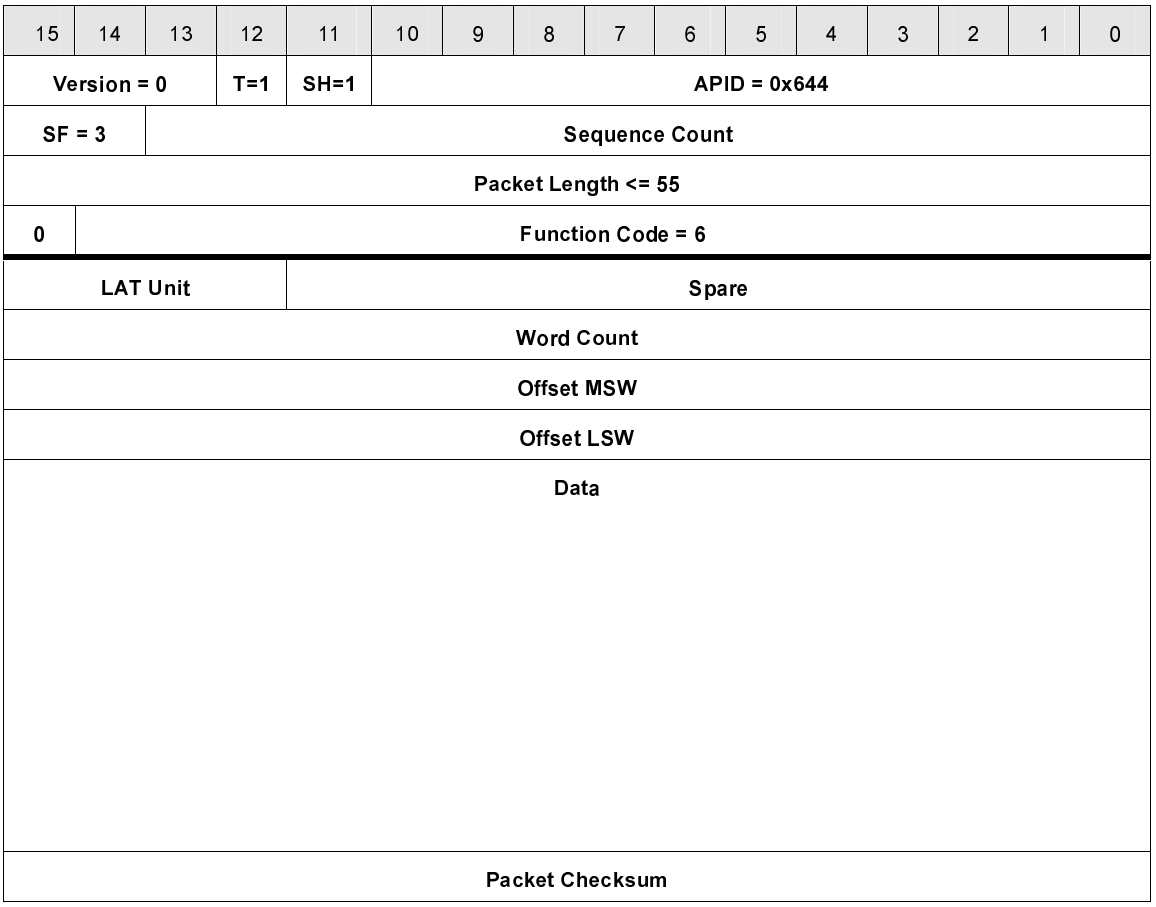
Offset – The offset from the beginning of the PCI device's configuration header at which the 16-bit value should be written.

Value – The 16-bit value to write into the selected PCI configuration header location.

6.2.2 Processor Register Write

The Processor Register Write telecommand writes 32-bit values into the designated processor registers.

Figure 46 – Processor Register Write Telecommand Format



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

Word Count – The number of 32-bit register values to write.

Offset – The starting offset within the processor register block to write the data. For the PBC, this value must be aligned to a 32-bit boundary. See Figure 47 for the locations of the various registers within the processor register block.

Data – Up to 11, 32-bit data values to write into the processor registers.

Figure 47 – Processor Register Offsets

Offset	Register	Offset	Register	Offset	Register	Offset	Register
0x000	GPR 0	0x0b0	IBAT 2U	0x160	FPR 1L	0x210	FPR 23L
0x004	GPR 1	0x0b4	IBAT 2L	0x164	FPR 2U	0x214	FPR 24U
0x008	GPR 2	0x0b8	IBAT 3U	0x168	FPR 2L	0x218	FPR 24L
0x00c	GPR 3	0x0bc	IBAT 3L	0x16c	FPR 3U	0x21c	FPR 25U
0x010	GPR 4	0x0c0	DBAT 0U	0x170	FPR 3L	0x220	FPR 25L
0x014	GPR 5	0x0c4	DBAT 0L	0x174	FPR 4U	0x224	FPR 26U
0x018	GPR 6	0x0c8	DBAT 1U	0x178	FPR 4L	0x228	FPR 26L
0x01c	GPR 7	0x0cc	DBAT 1L	0x17c	FPR 5U	0x22c	FPR 27U
0x020	GPR 8	0x0d0	DBAT 2U	0x180	FPR 5L	0x230	FPR 27L
0x024	GPR 9	0x0d4	DBAT 2L	0x184	FPR 6U	0x234	FPR 28U
0x028	GPR 10	0x0d8	DBAT 3U	0x188	FPR 6L	0x238	FPR 28L
0x02c	GPR 11	0x0dc	DBAT 3L	0x18c	FPR 7U	0x23c	FPR 29U
0x030	GPR 12	0x0e0	SDR1	0x190	FPR 7L	0x240	FPR 29L
0x034	GPR 13	0x0e4	PVR	0x194	FPR 8U	0x244	FPR 30U
0x038	GPR 14	0x0e8	DAR	0x198	FPR 8L	0x248	FPR 30L
0x03c	GPR 15	0x0ec	SPRG 0	0x19c	FPR 9U	0x24c	FPR 31U
0x040	GPR 16	0x0f0	SPRG 1	0x1a0	FPR 9L	0x250	FPR 31L
0x044	GPR 17	0x0f4	SPRG 2	0x1a4	FPR 10U	0x254	HID 0
0x048	GPR 18	0x0f8	SPRG 3	0x1a8	FPR 10L	0x258	HID 1
0x04c	GPR 19	0x0fc	DSISR	0x1ac	FPR 11U	0x25c	MMCR 0
0x050	GPR 20	0x100	SRR 0	0x1b0	FPR 11L	0x260	MMCR 1
0x054	GPR 21	0x104	SRR 1	0x1b4	FPR 12U	0x264	PMC 1
0x058	GPR 22	0x108	DEC	0x1b8	FPR 12L	0x268	PMC 2
0x05c	GPR 23	0x10c	DABR	0x1bc	FPR 13U	0x26c	PMC 3
0x060	GPR 24	0x110	EAR	0x1c0	FPR 13L	0x270	PMC 4
0x064	GPR 25	0x114	SR 0	0x1c4	FPR 14U	0x274	SIA
0x068	GPR 26	0x118	SR 1	0x1c8	FPR 14L	0x278	IABR
0x06c	GPR 27	0x11c	SR 2	0x1cc	FPR 15U	0x27c	L2CR
0x070	GPR 28	0x120	SR 3	0x1d0	FPR 15L	0x280	ICTC
0x074	GPR 29	0x124	SR 4	0x1d4	FPR 16U	0x284	THRM 1
0x078	GPR 30	0x128	SR 5	0x1d8	FPR 16L	0x288	THRM 2
0x07c	GPR 31	0x12c	SR 6	0x1dc	FPR 17U	0x28c	THRM 3
0x080	CR	0x130	SR 7	0x1e0	FPR 17L	0x290 - 0x2fc	<unused>
0x084	FPSCR	0x134	SR 8	0x1e4	FPR 18U		
0x088	XER	0x138	SR 9	0x1e8	FPR 18L		
0x08c	LR	0x13c	SR 10	0x1ec	FPR 19U		
0x090	CTR	0x140	SR 11	0x1f0	FPR 19L		
0x094	TBL	0x144	SR 12	0x1f4	FPR 20U		
0x098	TBU	0x148	SR 13	0x1f8	FPR 20L		
0x09c	MSR	0x14c	SR 14	0x1fc	FPR 21U		
0x0a0	IBAT 0U	0x150	SR 15	0x200	FPR 21L		
0x0a4	IBAT 0L	0x154	FPR 0U	0x204	FPR 22U		
0x0a8	IBAT 1U	0x158	FPR 0L	0x208	FPR 22L		
0x0ac	IBAT 1L	0x15c	FPR 1U	0x20c	FPR 23U		

6.3 Memory Dump Telecommands

The memory dump telecommands read blocks of data from various locations within the SIU and send the data in a series of boot housekeeping telemetry packets. The locations that can be dumped include SDRAM, SIB EEPROM, PPCI registers, PCI device headers, and processor registers. Table 8 summarizes the memory dump telecommands accepted by the Primary Boot Code, which is a subset of those available from the MEM package. See the documentation for that package for a more complete description of these commands.

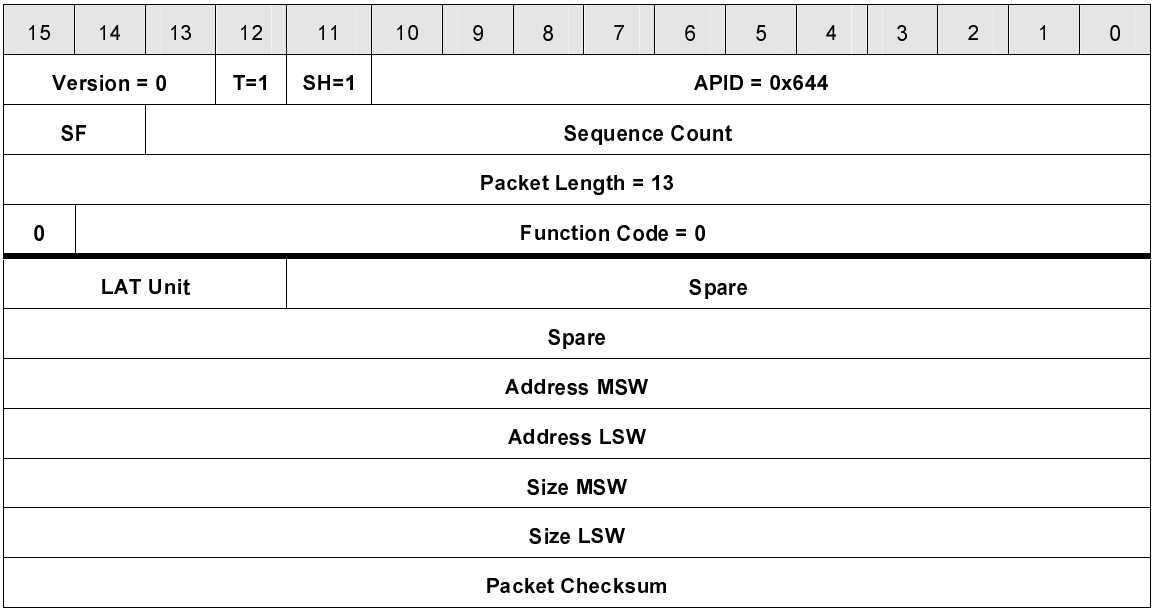
Table 8 – Memory Dump Telecommand Function Codes

APID	Function Code	Description
0x644	0	Memory Data Dump. Dump the contents of a range of memory into a series of telemetry packets.
	1	Memory Dump Cancel. Cancel the current memory dump.
	2	PCI Device Header Dump. Dump the selected PCI device header.
	3	Processor Register Dump. Dump the processor registers.

6.3.0 Memory Data Dump

The Memory Data Dump telecommand causes the contents of a specified memory region to be sent out as a sequence of telemetry packets. Due to alignment constraints of the target memory, the Memory Data Dump command may fail with an error if the address or size parameters violate those constraints. A small memory map is maintained by the PBC to designate the addresses of the legal memory regions and the constraints of those regions.

Figure 48 – Memory Data Dump Telecommand Format



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

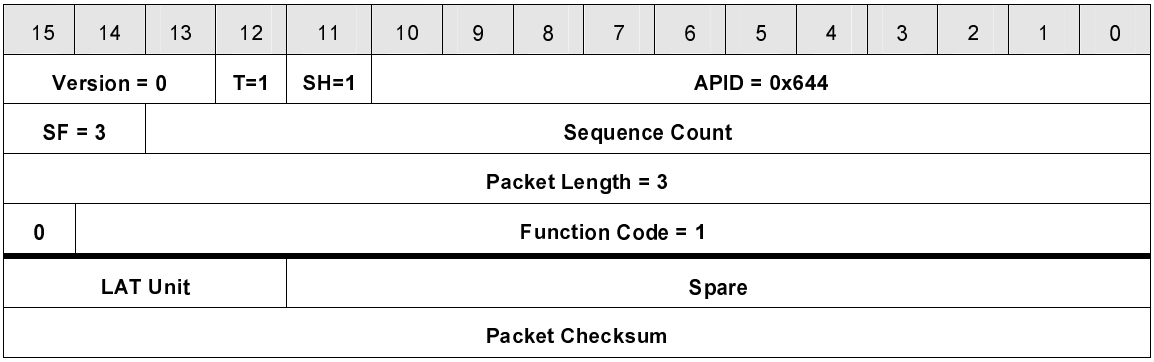
Address – The starting address of the memory region to dump. For the PBC, the address must be aligned to a 32-bit boundary.

Size – The number of 32-bit words to dump.

6.3.1 Memory Dump Cancel

The Memory Dump Cancel command stops any further memory dump data from being sent in telemetry and resets the memory dump state machine. The command will generate an error if a memory dump is not currently in progress.

Figure 49 – Memory Dump Cancel Telecommand Format

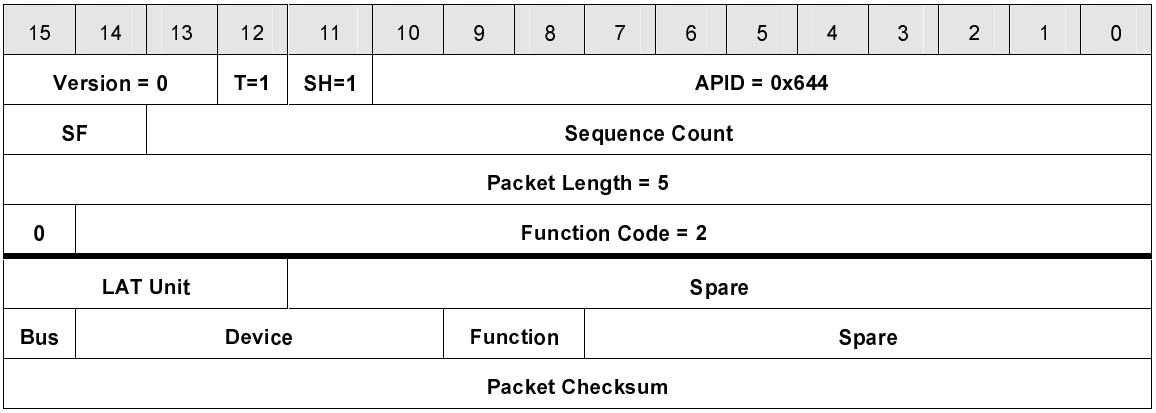


LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

6.3.2 PCI Device Header Dump

The PCI Device Header Dump telecommand causes the contents of the specified PCI device header to be sent out as a series of telemetry packets.

Figure 50 – PCI Device Header Dump Telecommand Format



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

Bus – PCI Bus Select. For the PBC, this must be '0'.

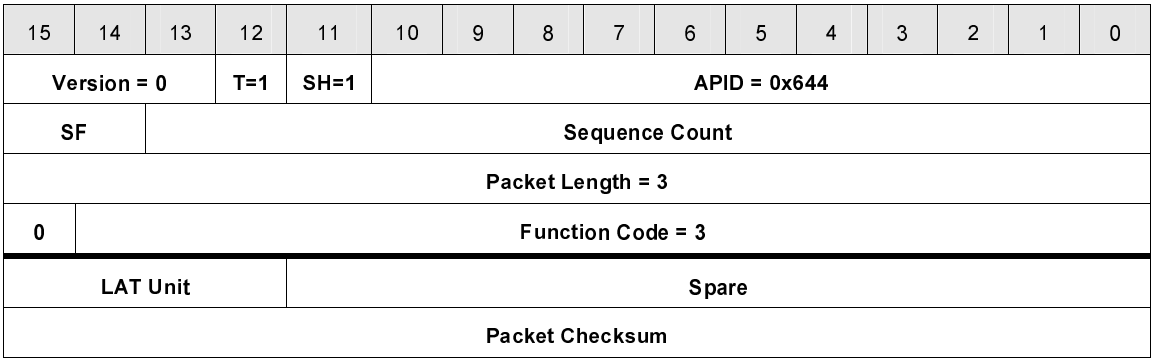
Device – PCI Device Select. This value will vary based on system configuration (test vs flight).

Function – PCI Device Function Select. For the PBC, this must be '0'.

6.3.3 Processor Register Dump

The Processor Register Dump telecommand causes the contents of the processor's internal registers to be sent out as a series of telemetry packets. The register values are arranged in the format described by Figure 47.

Figure 51 – Processor Register Dump Telecommand Format



LAT Unit – ID of the CPU that should execute the command (SIU, EPU 0, etc.), as defined by ITC.

6.4 Boot Housekeeping Telemetry

The Primary Boot Code sends only one type of telemetry packet – the Boot Housekeeping Telemetry packet. The boot shell uses this telemetry both as a keep-alive heartbeat and as a way of reporting operational status and error reports.

Figure 52 – Boot Housekeeping Telemetry Format

offset (hex)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00	Version = 0			T=0	SH=1	APID = 0x200										
02	SF=3			Sequence Count												
04	Packet Length = 109															
06	Timestamp Seconds MSW															
	Timestamp Seconds LSW															
0a	Timestamp Sub-Seconds MSW															
	Timestamp Sub-Seconds LSW															
0e	Software Mode															
10	Total Error Count															
12	Queued Error Count															
14	Error Word MSW															
	Error Word LSW															
18	Received Telecommand Count															
1a	Accepted Telecommand Count															
1c	Latest Error Word MSW															
	Latest Error Word LSW															
20	Last Command Function Code							Last Command APID								
22	Spare															
24	Spare															
26	File Upload State															
28	File Upload Packet Count															
2a	Scrub Address MSW															
2c	Boot Type															
2e	Memory Dump Word Count															
30	Memory Dump Address MSW															
	Memory Dump Address LSW															
34	Memory Dump Data Word 0 MSW															
	Memory Dump Data Word 0 LSW															
38	Memory Dump Data Word 1 MSW															

offset (hex)	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Memory Dump Data Word 1 LSW															
3c	**Memory Dump Data Word 2 MSW**															
	Memory Dump Data Word 2 LSW															
40	**Memory Dump Data Word 3 MSW**															
	Memory Dump Data Word 3 LSW															
44	**Memory Dump Data Word 4 MSW**															
	Memory Dump Data Word 4 LSW															
48	**Memory Dump Data Word 5 MSW**															
	Memory Dump Data Word 5 LSW															
4c	**Memory Dump Data Word 6 MSW**															
	Memory Dump Data Word 6 LSW															
50	**Memory Dump Data Word 7 MSW**															
	Memory Dump Data Word 7 LSW															
54	**Memory Dump Data Word 8 MSW**															
	Memory Dump Data Word 8 LSW															
58	**Memory Dump Data Word 9 MSW**															
	Memory Dump Data Word 9 LSW															
5c	**Memory Dump Data Word 10 MSW**															
	Memory Dump Data Word 10 LSW															
60	**Memory Dump Data Word 11 MSW**															
	Memory Dump Data Word 11 LSW															
64	**Memory Dump Data Word 12 MSW**															
	Memory Dump Data Word 12 LSW															
68	**Memory Dump Data Word 13 MSW**															
	Memory Dump Data Word 13 LSW															
6c	**Memory Dump Data Word 14 MSW**															
	Memory Dump Data Word 14 LSW															
70	**Memory Dump Data Word 15 MSW**															
	Memory Dump Data Word 15 LSW															