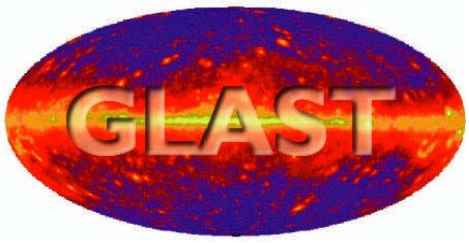


LAT Flight Software

Primary Boot Code

Number	LAT-TD-1806-07
Subsystem	Data Acquisition/Flight Software
Supercedes:	LAT-TD-1806-06
Type:	Design Description
Author:	D.L. Wood & R.C. Caperoon
Created:	15 April 2004
Updated:	16 April 2004
Printed:	16 April 2004

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

Contents

Figures

Tables

0 Introduction

The LAT primary boot is a ROM based executable which is responsible for bootstrapping the CPU boards in both the SIU and EPU crates. The primary boot code resides in the RAD750 CPU board SUROM and is not modifiable 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 primary boot code shell, and the loading and execution of a VxWorks RTOS executable image. The LAT secondary boot 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 secondary boot code modules may be made without any changes to the primary boot code.

The LAT CPU boards begin execution of the primary boot code in SUROM in response to the following reset conditions.

1. Cold boot cases
    a. The RAD750 CPU board has been powered on and the power on reset signal is asserted.
    b. The RAD750 CPU board has been manually hard reset. This is equivalent to the power on reset scenario.
    c. The RAD750 CPU hardware watchdog timer has expired.
2. Warm boot cases as the result of an explicitly 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 following requirements shown in Figure 1.

These requirements are used to specify the features needed to be implemented by the Primary Boot Code and are used 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, intrusive test, analysis, demonstration or inspection; these methods are indicated as ST, IT, A, D or 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-programmable in flight.	
1030	SIB EEPROM refers to the EEPROM resident on the Spacecraft Interface Board (SIB). This EEPROM is intended to be re-programmable in 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 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 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 diagnostics upon initial startup then go into a state waiting for a 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 PCI bridge 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 values 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 packet 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 a PCI device header for a device present on the PCI bus.	ST

5050	When a SIATTITUDE command packet is received on the external interface, the PBC SHALL ignore the command.	ST
5060	When a 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 a 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 BOOT MEMORY DUMP START 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 BOOT MEMORY 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 to 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 file loader 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 file 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	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 file loader store the file data in the command packet into RAM and enter the VALID state.	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	When a FILE UPLOAD DATA command packet is received on the external interface, the PBC SHALL copy data contained in the command to the RAM upload buffer.	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 access 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 command 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 command 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

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 shell is the boot code's other major component. The boot shell communicates through an external I/O interface. The boot shell may receive commands and upload information and return telemetry information.

0.2 Acronyms and Definitions

1553 – MIL-STD-1553B serial data bus specification; in particular the serial data bus and data protocol implemented for the GLAST mission.

APID – CCSDS packet application identifier. A numerical code indicating the general type of data in a CCSDS packet.

HKP – Real-time housekeeping telemetry data; telemetry data which relates to the health and safety of the LAT instrument.

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

LCB – The LAT Communications Board; 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 SIU 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 – The PowerPC microprocessor. In particular the RAD750 processor provided on the BAE RAD750 cPCI CPU board.

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

RTOS – Real Time Operating System; in particular the VxWorks 5.4 operating system used by the LAT.

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

SIB – Spacecraft Interface Board; the board in the SIU crates that contains the LAT 1553 remote terminal hardware.

SIU – The LAT Spacecraft Interface Unit; one of two computer crates which contain a processor and 1553 Remote Terminal interface hardware.

SUROM – Startup ROM; 256 KB of EEPROM memory on the RAD750 CPU boards that holds the primary boot code; the SUROM is only programmable on the bench through the PPCI JTAG interface.

0.3 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 portion of the boot code is written in EMC and PPC assembly languages. Its primary duties are to initialize the board hardware, perform critical diagnostics, 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 stub must establish a working area of SDRAM for the C code to run at, 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 the included 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). After reset the PPCI bridge chip establishes the default PCI memory map as show in .

The PPCI bridge holds the PPC reset signal in the asserted state until manually cleared by the EMC controller. When the PPCI bridge comes out of the reset state, the EMC controller begins execution of instructions starting at SUROM address FFF00020. Before the EMC allows the PPC processor to begin execution of its reset code, it performs the following actions (these actions are also shown in Figure 82):

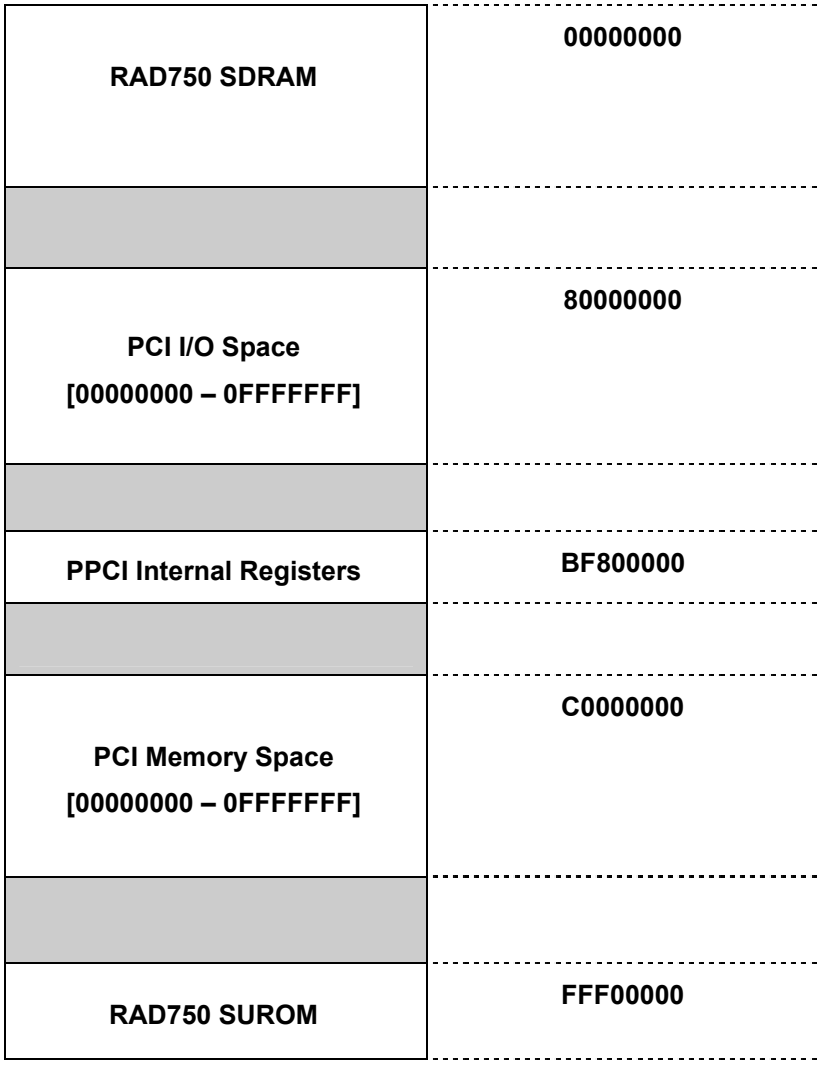
1. The PPCI clock control register is initialized to set the PCI bus clock at 33 MHz, the PPCI timer clock at 8.25 MHz, and the CPU/Memory clock at 33 MHz. The PPC PLL configuration lines are changed from their default state so that the PPC will start executing at 115.5 MHz (3.5x multiplier).
2. The PPCI memory controller Bank 0 is initialized. Bank 0 is set to map the 256 KB of SUROM at address FFF00000 as read-only memory. The SECDED error detection and correction mode is enabled for the SUROM memory bank. Error scrubbing is disabled.
3. The PPCI memory controller Bank 1 is initialized. Bank 1 is set to map the 128 MB of system SDRAM at address 00000000 as read-write memory. Nibble error detection and correction mode is enabled for the SDRAM memory bank. Error scrubbing is enabled.
4. If the current startup is due to a power-on-reset, the area of memory used to store the boot diagnostics is initialized.

5. All memory allocated to boot (except the area of memory used to store the boot diagnostics) is initialized.
6. A self-test of the PPCI bridge chip is performed. The results are logged to the RAM diagnostics area as described in Figure 36.
7. A self-test of the PPC processor is performed. The results are logged to the RAM diagnostics area as described in Figure 36.

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. It will not become active again unless signaled by the PPC processor.

Between the PPCI hardware reset defaults and the EMC initialization code, the hardware memory map shown in Figure 3 is established.

Figure 3 - Primary Boot Code Hardware Memory Map



1.1 CPU Reset

The PPC bootstrap code at the SUROM reset vector address FFF00100 is written in assembly both to access some special instructions needed to initialize the PPC processor and to guarantee that RAM is never accessed before being tested. The first job is to assess the cause of the reboot. If the processor starts executing from address FFF00100 then it is either a power-on reset or hardware reset of the processor board. Otherwise, if the start address is FFF00104, then the CPU is entering a warm boot. The warm boot action flags may then be accessed in register r3. The Primary Boot Flags are set to appropriate values, as described in Section 3.0, and the processor internal registers are setup.

1. The MSR register is set so that the processor uses exception vector base address FFF00000 (in SUROM) and external interrupts are disabled.
2. 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 FFFFFFFF.
3. The PIDs 5 and 6 are set to zero, indicating no error has occurred yet.
4. The results of the RAD750 built-in self-tests are copied from SDRAM into processor registers so that these results are not overwritten by the SDRAM tests.
5. The MMU DBAT registers are initialized to create the SDRAM, SUROM, and PCI I/O memory spaces. The addresses are always mapped directly. The purpose of this step is to allow the data cache to be enabled. The SDRAM and SUROM memory regions are cache enabled. The MMU page tables are disabled as well as all instruction address mapping.
6. The PPC instruction and data caches are not enabled.
7. The PCI_RST# signal is asserted to reset the devices on the PCI bus.
8. The PPC processor branch history table and parallel instruction execution optimizations are enabled.
9. The SDRAM is initialized for boot operations. SDRAM scrubbing and sparing are disabled in preparation to run the memory tests.

These actions are shown in detail in Figure 83, in Appendix B of this document.

1.1.0 PPC Virtual Memory Map

The primary boot code maps the memory regions from the hardware memory map shown in Figure 3 to the PPC MMU as shown in Figure 4. The primary boot code only uses the Data Block Address Translation (DBAT) feature of the Memory Management Unit (MMU) of the PPC processor. The page table entry feature of the MMU is known not to function properly in the RAD750. Five regions are required by the PBC but only 4 DBATS are available. The PCI configuration registers are not mapped by a DBAT. When accessing these registers, the memory manager must be disabled. This region was chosen to be 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.

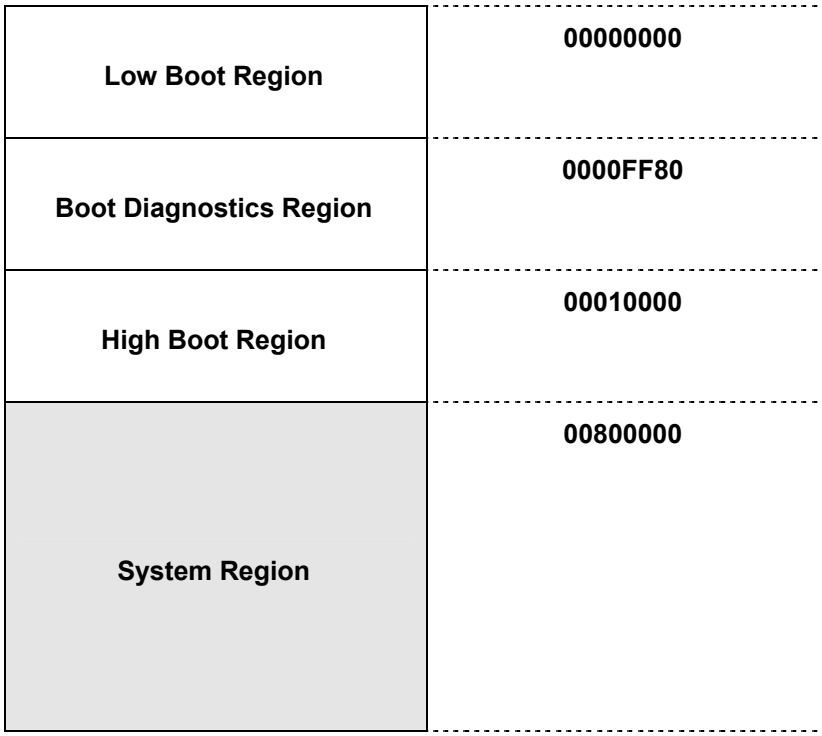
Figure 4 - Primary Boot Code Virtual Memory Map

Region	Addresses	DBAT
RAD750 SDRAM	00000000 7FFFFFFF	DBAT 0
Not Mapped	08000000 7FFFFFFF	
PCI Configuration Registers	80000000 80000CFF	Not managed, see text for explanation
Not Mapped	80000D00 BF7FFFFF	
PPCI Internal Registers	BF800000 BF8FFFFF	DBAT 1
Not Mapped	BF900000 BFFFFFFF	
PCI Memory Space [00000000 – 0FFFFFFF]	C0000000 CFFFFFFF	DBAT 2
Not Mapped	D0000000 FFFEFFFF	
RAD750 SUROM	FFF00000 FFF3FFFF	DBAT 3
Not Mapped	FFF40000 FFFFFFFF	

1.2 Memory Test And Configuration

After the PPC processor and memory interface are initialized, the primary boot code next tries to establish a suitable region of SDRAM for operations. The boot code initially maps SDRAM memory as follows:

Figure 6 - SDRAM Memory Map Before Memory Test



The SDRAM above address 00800000 is not used by the primary boot code.

1.2.0 Memory Regions Tested

The memory test flow is shown in Figure 86 in the Appendix B of this document.

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 Region, Boot Diagnostics Region and High Boot Regions of SDRAM. The memory test determines if the SDRAM in good working order so that the boot shell C environment code can run successfully.

If the Primary Boot Flags indicate that the System Region is to be tested, then a memory test is run on the System Region of SDRAM.

If errors are found in the SDRAM, corrective action is attempted. This action consists of swapping out the spare byte column and retesting SDRAM. If the SDRAM test fails after corrective action, the PID critical status value is set (see Section 3.1.0). Otherwise, the SDRAM is mapped for the boot shell operations. 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 a contiguous working region of good memory is known to exist.

If the Primary Boot Flags indicate that this is a warm boot or watchdog reset, the memory test is skipped. This allows the boot code to access reboot information stored in the boot diagnostics region (see Section 3.0). This also allows the boot code to access the RAM used by the application code for debugging.

1.2.1 Memory Test Algorithm

Memory regions are tested in four regions, as defined above. For each region, the memory test starts by with an address fill. Each 32-bit word in the memory region is set to its 32-bit address.

Then for each 32-bit word in the region, the address is verified. This test is intended to detect any problems with addressing in the memory.

After the address value is verified, a walking bit test is performed on each location to detect failed memory parts or stuck bits. Each bit of each nibble in the 32-bit location is toggled with an exclusive OR operation. Successive steps of patterns generated by exclusive-or logic accomplish this. This ensures any bit transition and bit value works for every memory location tested.

Finally, after the test is complete, the content of the memory status register is checked. If this register indicates any problems (including correctable single bit errors) this is reported. However problems indicated by the memory controller do not result in a remediation attempt. Some sort of ground intervention would be required to take action if these types of memory errors occur.

Figure 7 shows pseudo-code for the memory test algorithm.

Figure 7 - Memory Test Algorithm

```
Clear memory status register (0xBF8000C0);

/* set initial value */
For each 32-bit memory location (ADDR) in region under test
{
      *ADDR = ADDR;
}

For each 32-bit memory location (ADDR) in region under test:
{
      /* test initial value */
      tv = *ADDR;
      if (tv != ADDR) indicate failure code 7;

      /* step 1 */
      *ADDR = *ADDR ^ 0x11111111;
      tv = *ADDR;
      if (tv != ADDR ^ 0x11111111) indicate failure code 3;

      /* step 2 */
      *ADDR = *ADDR ^ 0x33333333;
      tv = *ADDR;
      if (tv != ADDR ^ 0x22222222) indicate failure code 4;

      /* step 3 */
      *ADDR = *ADDR ^ 0x66666666;
      tv = *ADDR;
      if (tv != ADDR ^ 0x44444444) indicate failure code 5;

      /* step 4 */
      *ADDR = *ADDR ^ 0xCCCCCCCC;
      tv = *ADDR;
      if (tv != ADDR ^ 0x88888888) indicate failure code 6;

      /* step 5 */
      *ADDR = *ADDR ^ 0x88888888;
      tv = *ADDR;
      if (tv != ADDR) indicate failure code 2;
}

tv = *(memory status register);
if (tv != 0) indicate failure 8;
```

As an example, consider the memory location 0x0002F3C4. The following patterns are written and verified:

Figure 8 - Memory Test Example

Initial Value	0x0002F3C4
Step 1	0x1113E2D5

Step 2	0x2220D1E6
Step 3	0x4446B780
Step 4	0x888A7B4C
Step 5	0x0002F3C4

If a failure (other than failure code 8, an error indicated by the memory controller) is encountered, testing of the current block is halted. Remediation is attempted and the whole region is tested again. (See section 1.2.2 below).

1.2.2 Memory Failure Remediation

If any failure is detected during a memory test, a bad byte column index will be generated. If multiple errors have occurred, only one bad byte column index is generated (it will be the most significant byte column), no matter how many columns contain failures. On the first memory failure, 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 on the failed region.

Sparing logic is only programmed once. If a memory test fails a second time, the memory test stops for the region. The results are recorded and any further memory tests are not run.

If memory test fails and remediation is attempted, the output PID value is set to indicate hard memory error. See Figure 43 for detail on PID error codes.

The LAT is using the 3U version of the BAE RAD750. This computer provides one spare memory column. This byte column is shared with the NIBBLE error correction logic. If the spare memory column is used, the error correction must be changed from NIBBLE error correction to SECDED error correction.

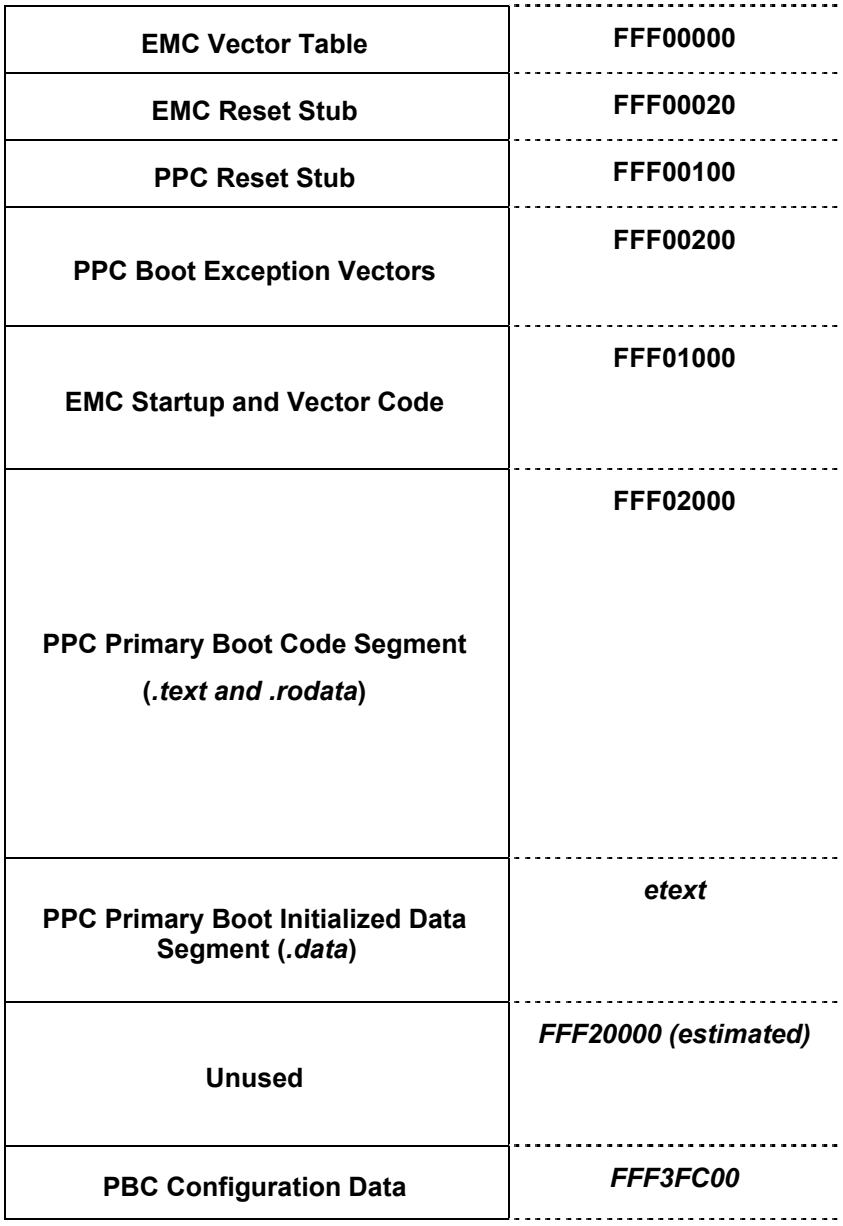
1.2.3 Memory Configuration

After the memory test is complete, active memory scrubbing is enabled in the PPCI memory controller for bank 1 SDRAM. The active memory scrubbing delay timer (address BF80007C) is set to TBD memory clocks.

1.3 SUROM Memory Setup

The majority of the primary boot code consists of the boot shell. The boot shell code and constant data segments (*.text* and *.rodata*), however, remain in SUROM and execute from there 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 10 - SUROM Memory Map



The EMC vector table is an 8-entry table defining the addresses of the EMC vector handlers. Each entry contains an EMC branch instruction that directs execution to one of the vector handlers contained in the main EMC code located at address FFF01000. The EMC reset state of the Vector Anchor Register is FFF00000, so no additional initialization of this register is necessary. The EMC reset stub at address FFF00020 is the location where the EMC begins instruction execution after reset. An EMC branch instruction directs execution to the main EMC startup code at address FFF01000. The PPC reset stub at address FFF00100 is the location where the PPC begins instruction execution after reset. This area is setup as a small branch table. The PPC branch instructions direct the execution to locations in the main startup code

located in the SUROM area at address FFF02000. The PPC boot exception vectors contain the exception handlers for the primary boot code. The code contained in these locations is described in Section 3.2.

The boot shell variable data segments still need to be relocated to a known working area in SDRAM. The initialization values for the boot shell program data variables are contained at a known offset in SUROM. The linker generated symbol *etext* provides the address in SUROM of the program data variables initialization values.

1.3.0 PBC Configuration Data

PBC configuration data is kept in the RAD750 SUROM. This data is used to configure the PBC software, currently this configuration data determines which interface the PBC should use for command and telemetry.

Data items in this region are kept in triplicate to guard against bad memory locations in SUROM causing a mis-configured crate. These values are voted to come up with the configuration value to use. If only two of the three values match, the majority is chosen and an error is reported in telemetry. If all three values are different, a default is chosen and an error is reported in telemetry.

A checksum (not kept in triplicate) for the configuration region is stored in the configuration data. During PBC configuration, the checksum is checked. If the checksum stored in SUROM does not match the calculate checksum, an error is reported in telemetry.

The format and content of the configuration area are shown in Figure 11 and

Figure 12.

Figure 11 - PBC Configuration Data Format

Offset	Mnemonic	Contents
0x000	PBC_CFG_PERS	PBC Personality copy 1 (EPU vs. SIU)
0x004	PBC_SUROM_CHECKSUM	SUROM checksum copy 1.
0x008	PBC_SUROM_LENGTH	Length of checksumed SUROM data (in bytes), copy 1.
0x008-0x103		Reserved
0x104	PBC_CFG_PERS	PBC Personality 2 (EPU vs. SIU)
0x108	PBC_SUROM_CHECKSUM	SUROM checksum copy 2.
0x10C	PBC_SUROM_LENGTH	Length of checksumed SUROM data (in bytes), copy 2.
0x110-0x2F7		Reserved
0x208	PBC_CFG_PERS	PBC Personality 3 (EPU vs. SIU)

0x20C	PBC_SUROM_CHECKSUM	SUROM checksum copy 3.
0x210	PBC_SUROM_LENGTH	Length of checksumed SUROM data (in bytes), copy 3.
0x214-0x3FC		Reserved
0x3FC	PBC_CFG_CHECKSUM	PBC Configuration Data Checksum

Figure 12 - PBC Configuration Contents

Mnemonic	Enumerated Values	Meaning	Default Value On Voting Error
PBC_CFG_PERS	0x0003FFFC	PBC is to act as an SIU and communicate on the 1553 bus.	0x0003FFFC
	0x0C00F3FF	PBC is to act as an EPU and communicate on the LCB.	
PBC_SUROM_CHECKSUM		Checksum of SUROM contents. This checksum uses the adler32 algorithm and is the checksum from the start of SUROM and runs the number of bytes specified by PBC_SUROM_LENGTH.	0x00000000
PBC_SUROM_LENGTH		Length of data to be checksumed in SUROM, in bytes.	0x3FC00
PBC_CFG_CHECKSUM		Checksum of PBC configuration contents. This checksum uses the adler32 algorithm and is the checksum of the contents of the entire PBC configuration region except for this checksum.	0x00000000

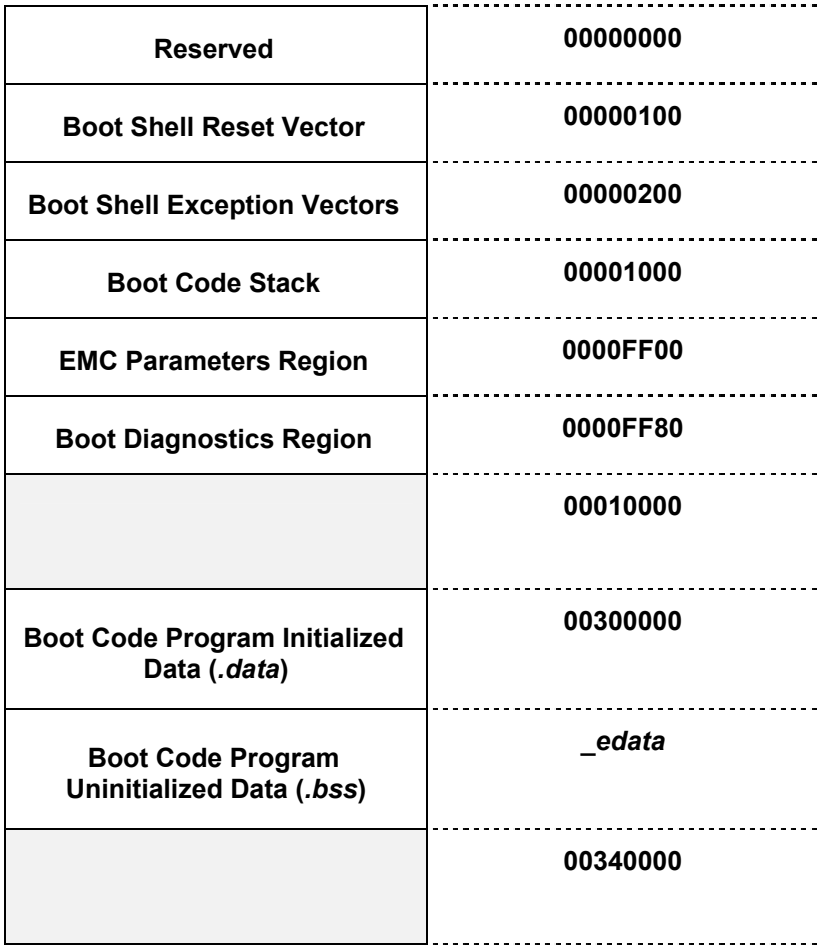
1.4 C Environment Initialization

The last piece of assembly code creates an environment for the execution of the boot shell code, which is written in C. The stack pointer (PPC register *r1*) is set to the appropriate address in

SDRAM. This address is at offset 0000FF80 from the base address of the SDRAM boot region. The stack will grow downwards from this address to lower addresses. The SDRAM that makes up the boot stack is initialized to zero.

The values of any of the boot shell initialized program variables (*.data* section) are copied from the appropriate region of SUROM into the region of SDRAM at address 00300000. The *.bss* section will be located immediately following the *.data* section in the selected SDRAM boot region. The program data sections are always linked to start at address 00300000 from the base of the SDRAM boot region The SDRAM at address The SDRAM boot region selected will then be mapped as follows:

Figure 14 - RAM Boot Region Memory Map After C Environment Initialization



The location of the initialized data values in SUROM is found by the linker generated symbolic address *etext*. The size of the primary boot code data variable region is found by the difference of the linker generated symbol *_edata* and the known SDRAM address of the section 00300000. The size of the boot program data uninitialized data section (*.bss*) is found by the difference of the linker symbols *_end* and *_edata*. The *.bss* section in SDRAM is set to '0'.

The last thing for the reset initialization code to do is to jump to the appropriate C function entry point in SUROM to start the boot shell. The entry point address will be at a known address in

SUROM. The boot type will be passed to the C entry point as the first (and only) argument (register r3 by convention).

This boot type is available for use in the PBC, although no use is defined for now. This boot type is also passed to the Secondary Boot Code and can be made available to the application code for use. The boot type indicates a source of the boot. For hardware, watchdog, exception or boot code panic restarts, the values are pre-defined. For warm restarts, the application will use the VxWorks start type. The boot type codes are described below:

Figure 15 - Boot Type Codes

Value	Meaning	Description
0	VxWorks Start	Processor was rebooted from VxWorks application.
1	Cold Start	Processor was started from a power-on or hardware reset.
2	Watchdog Start	Processor was started from a watchdog reset.
3	Panic Start	Boot code has detected a panic situation and restarted. Possible panic sources are: • Boot Shell returns to caller. • Secondary entry point returns to caller.
4	Exception Start	Processor was started due to and exception in the PBC.
5	Commanded Start	Reboot was commanded by ground command

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 needed for booting the VxWorks RTOS and for replacing the RTOS image in SIB EEPROM and other critical SIB EEPROM information. In normal mode of operations, the boot shell simply waits for a short timeout period to make sure the external I/O interface has no incoming commands. After the timeout period, the default VxWorks RTOS image is loaded from SIB EEPROM to SDRAM, and execution of the RTOS is started.

2.0 Initialization

The boot shell first creates a system memory configuration, initializes the I/O communications interface and executes the final boot diagnostics.

2.0.0 SDRAM Memory Layout

The boot shell creates a layout of SDRAM memory as shown in Figure 17 below. The values shown in the right column are offsets from the SDRAM boot region base address selected by the assembly language reset stub. The base address of the SDRAM region to use is passed as a parameter to the boot shell initialization function.

Figure 17 - RAM Memory Map After Boot Shell Initialization

Reserved	00000000
Boot Shell Reset Vector	00000010
Boot Shell Exception Vectors	00000200
Boot Code Stack	00001000
EMC Parameter Region	0000FF00
Boot Diagnostics Region	0000FF80
Second Stage Boot RAM Module 0	00010000
Second Stage Boot RAM Module 1	00020000
VxWorks RTOS Load Region	00030000
Boot Shell File Upload Buffer	00200000
Boot Code Program Data	00300000
I/O Output Buffer	00320000
I/O Input Buffer	00321000
Boot Shell Memory Heap	00322000
LCB Input Ring Buffer	00400000 007FFFFF

The boot stack and program data segments were initialized by the boot reset stub. The diagnostics area contains pertinent information about the boot state (see Section 3.0). The secondary boot module areas are for storing temporary or trial versions of the secondary boot modules (see Section 2.0.2). The VxWorks RTOS load region is where the RTOS text and data segments will reside when it is loaded and started. The RTOS absolute binary image is first inflated to this address and jumping to address 00030000, where the RTOS initialization entry point is located, begins execution of the RTOS. The RTOS executable images are set at build time to run at this SDRAM address.

The boot shell file upload buffer is where upload file data is temporarily stored while waiting for the upload to complete and the file commit command to be issued. The file upload buffer is also used as a storage area to copy the RTOS load image from SIB EEPROM, since the ZLIB inflation routine must run on data stored in a byte-accessible memory region. Finally, the upload buffer may also serve as the place to store a temporary version of an RTOS executable that is to be run but not committed to SIB EEPROM.

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 will use this area to construct GLAST 1553 telemetry packet blocks when sending out housekeeping telemetry packets. The EPU LCB driver will use 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 will use this area to receive incoming telecommand packets. The EPU LCB driver will use 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 44 KB of temporary memory to perform each file inflation. This allows different software components of the boot code to remain somewhat modular. The original heap pointer starts at offset 00321000 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 ring buffer SDRAM region exists to support the reception of telecommand input packets through the LAT event fabric. The LCB 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. The PCI bus reset is now de-asserted, releasing the I/O boards in the backplane from the reset state. If the boot shell is starting from a warm boot, it will assert the PCI bus reset signal for TBD milliseconds. All of the I/O boards in the cPCI backplane should now be ready for configuration. The prescience of all 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 will program 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 either PCI Memory space.
2. The PCI configuration control register will be enabled for target Memory operations as well as for bus mastering capability.
3. The PCI configuration cache line size and latency timer members will be 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 PPCI bridge will be setup 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 will utilize this capability to provide a DMA ring buffer to receive telecommand packet input.

2.0.1.0 SIU PCI Setup

If the boot shell is operating in either SIU mode, it will program 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, and will never use it. The SIU PCI memory maps and SIB assignments are shown below

Figure 19 - SIU PCI Memory Space Address Map

<table>
<tr><td rowspan="4">SIB BAR 0</td><td>SIB Control/Status Registers
[00000000 – 001FFFFF]</td><td>C0000000</td></tr>
<tr><td>SIB Summit 1553 Controller Internal Registers
[00400000 – 0040007F]</td><td>C0400000</td></tr>
<tr><td>SIB 1553 Shared Memory
[00600000 – 0061FFFF]</td><td>C0600000</td></tr>
<tr><td>SIB EEPROM Bank 0
[00800000 – 00BFFFFF]</td><td>C0800000</td></tr>
</table>

		C0C00000
	SIB EEPROM Bank 1 [00C00000 – 00FFFFFF]	

2.0.1.1 EPU PCI Setup

If the boot shell is operating in EPU mode, it will program 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 21 - 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 partitions the boot regions of SIB EEPROM memory. The boot regions of SIB EEPROM memory are located at the base address of the SIB EEPROM banks, at CPU addresses C0800000 and C0C00000. The boot region contains an EEPROM bank header, one copy of the VxWorks RTOS executable image as well as two secondary boot modules. An area of the SIB EEPROM may be used as NVRAM (Non-Volatile RAM) area for the Secondary Boot. The format of this NVRAM area is not in the scope of this document.

The boot shell only deals directly with the RTOS executable image file, which must be stored at the designated address, possibly in ZLIB compressed format. The NVRAM and boot module files hold critical data needed for the secondary boot process such as the secondary boot executable and secondary boot application initialization script. These modules are loaded by the RTOS itself when it initializes.

Figure 23 - SIB EEPROM Boot Region Memory Map

EEPROM Bank Header	C0x00000
RTOS NVRAM	C0x00100 (nominal)
VxWorks RTOS Executable Image	C0x00200 (nominal)

Secondary Boot Module 0	**C0x60000 (nominal)**
Secondary Boot Module 1	**C0x70000 (nominal)**

The EEPROM Bank Header is stored at the first address of the SIB EEPROM Boot Region. The EEPROM bank header is 256 bytes longs and consists of:

1. A bank header checksum. This checksum uses the adler32 algorithm and is the checksum of the contents of the entire bank header minus the checksum itself (the checksum 252 bytes)
2. The RTOS File Offset. This is an offset in bytes from the base of the SIB EEPROM Boot Region to the address in SIB EEPROM where the Secondary Boot RTOS file is stored. The boot code expects a file header describing the RTOS file at this address, immediately followed by the RTOS file data.
3. The RTOS File Size. This is a size in bytes of the Secondary Boot RTOS file; including both the RTOS file header and RTOS file data.
4. The Segment 0 File Offset. This is an offset in bytes from the base of the SIB EEPROM Boot Region to the address in SIB EEPROM where the Secondary Boot Segment 0 file is stored. The boot code expects a file header describing the Segment 0 file at this address, immediately followed by the Segment 0 file data.
5. The Segment 0 File Size. This is a size in bytes of the Secondary Boot Segment 0 file, including both the Segment 0 file header and Segment 0 file data.
6. The Segment 1 File Offset. This is an offset in bytes from the base of the SIB EEPROM Boot Region to the address in SIB EEPROM where the Secondary Boot Segment 1 file is stored. The boot code expects a file header describing the Segment 1 file at this address, immediately followed by the Segment 1 file data.
7. The Segment 1 File Size. This is a size in bytes of the Secondary Boot Segment 1 file, including both the Segment 1 file header and Segment 1 file data.
8. The NVRAM Offset. This is an offset in bytes from the base of the SIB EEPROM Boot Region to the address in SIB EEPROM where the NVRAM data is stored.
9. The NVRAM File Size. This is a size in bytes of the NVRAM data.

The EEPROM Bank Header is shown below:

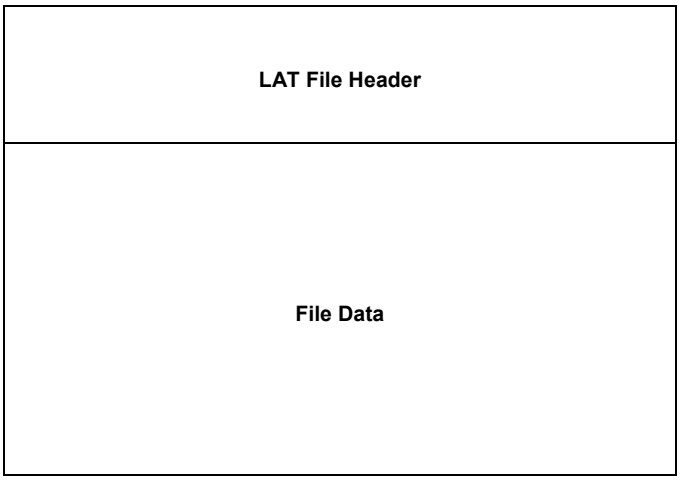
Figure 24 - EEPROM Bank Header

Offset	Mnemonic	Description
0x00	BHDR_CHECKSUM	Bank Header Checksum
0x04	BHDR_RTOS_PTR	Offset in bytes to RTOS file from the start of the SIB EEPROM.

0x08	BHDR_RTOS_SIZE	Size in bytes of RTOS file stored in SIB EEPROM.
0x0C	BHDR_SEG0_PTR	Offset in bytes to boot segment zero file from the start of the SIB EEPROM.
0x10	BHDR_SEG0_SIZE	Size in bytes of boot segment zero file stored in SIB EEPROM.
0x14	BHDR_SEG1_PTR	Offset in bytes to boot segment one file from the start of the SIB EEPROM.
0x18	BHDR_SEG1_SIZE	Size in bytes of boot segment one file stored in SIB EEPROM.
0x1C	BHDR_NVRAM_PTR	Offset in bytes to NVRAM from the start of the SIB EEPROM.
0x20	BHDR_NVRAM_SIZE	Size in bytes of NVRAM stored in SIB EEPROM.

Each of the files stored in the SIB EEPROM boot regions 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 file data. The file data itself is stored in SIB EEPROM immediately following the file header.

Figure 26 – SIB EEPROM Boot Region File Layout



2.0.3 PBC Configuration and Communications Setup

The PBC uses the configuration data stored in SUROM to choose a communications path to use for command and telemetry. The following steps are taken to configure PBC:

1. A checksum is calculated for the PBC configuration data region. If this checksum does not match the value of PBC_CFG_CHECKSUM stored in the configuration data, an error is reported.
2. A checksum is calculated for the contents of SUROM (excluding the configuration data). If this checksum does not match the value of PBC_SUROM_CHECKSUM stored in the configuration data, an error is reported.
3. The value of PBC_CFG_PERS is extracted from the configuration data region. If this value indicates the SUROM is configured for EPU operations, the boot communications are setup on the LBC interface. If the configuration data indicates that the SUROM is to be configured for SIU operations, the boot communications are setup on the 1553 interface.

See section 1.3.0 for details about how PBC configuration data is extracted from the SUROM.

The LAT primary boot code does not support the full functionality of the LAT CCSDS data packet protocol. The boot code only recognizes the boot shell operational command packets, ground upload packets, and the SC the 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 will initialize the boot mode 1553 driver to provide the external interface. For crates configured to run as an EPU, the boot shell will initialize the boot mode LCB driver to provide the external interface.

Any errors encountered during the communications driver initialization will be 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 must continually call the polled mode driver's query function in order to detect new 1553 bus activity. The boot version of the 1553 driver accepts all telecommand input packets, but never enables the boot code to send out a telecommand transmit packets. The polled mode driver will insert 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 must use the polled mode version of the LCB driver. The boot shell must continually call the polled mode driver's query function in order 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:

1. Check the external interface driver (1553 or LCB) for an indication that either a new telecommand packet has arrived or a communications error was detected.
2. Check 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, do so and send it to the external interface driver (1553 or LCB).
3. If a file commit operation targeting SIB EEPROM is in operation, check to see if the last location write cycle has completed. If so, start the write programming cycle for the next SIB EEPROM location if any remain.
4. Reset the RAD750 PPCI watchdog timer to a timeout value of 600 seconds.

When the RTOS image is executed (by explicit command), the primary boot code environment is destroyed and 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.

The boot shell state machine is shown graphically in Appendix A in Figure 79, Figure 80 and Figure 81. The basic state machine shows the transitions related to RTOS startup. The file load state machines shows details about transitions made during file load operations. The memory write state machine shows details about memory write to EEPROM regions.

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. The boot code needs to process the telecommand packets to obtain the following information:

1. Operational commands to provide control over the initialization process.
2. Upload data for replacement of SIB EEPROM segments.
3. Time information so that telemetry output may be time tagged.

Telecommands not listed in Sections 2.2.0, 2.2.1, 2.2.3, 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. The operational command packets will be rejected immediately if any of the fixed header fields have an incorrect value. All command packets should 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 at least a two-byte boundary. The trailing checksum must be correct for the telecommand packet to be accepted. 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 29 - Boot Shell Operational Telecommands

APID	Function Code	Description
0x640	0	Boot Start. Used as a NOOP command.
	1	Boot Reset. Forces a reset of the primary boot code into a known restart state.
	2	Boot Error Dump. Dump the value of a queued error work by the primary boot code.
	3	Boot RTOS Execute. Begins execution of an RTOS image and the secondary 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 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 30 - Boot Shell File Upload Telecommands

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

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 will write the contents of the temporary RAM file upload buffer into the final storage destination and report any errors encountered during the write operation.

Also, 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 support 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 a single CCSDS telecommand packet. A set of telecommand APID values, 0x642 and 0x644, has been reserved to designate memory management telecommands. The operations commands accepted by the boot shell are enumerated by function code in below.

Figure 32 - Boot Shell Memory Management Telecommands

APID	Function Code	Description
0x642	0	Memory Write. Update memory contents at the selected address(es) with telecommand data.
	1	PCI Device Header Write. Update contents of selected PCI Device Header with telecommand data.
	2	Processor Register Write. Update contents of selected processor register with telecommand data.
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 memory dump.
	5	PCI Device Header Dump. Dump the selected PCI device header.
	6	Processor Register Dump. Dump the processor general purpose and control 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 as valid all of the ancillary telecommand packets that the SC regularly delivers to the GLAST instruments. A telecommand APID value, 0x701, has been reserved to designate SC ancillary telecommands shows the listing of these telecommands.

Figure 33 - 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.

The boot shell will accept all of the telecommands listed in Figure 33, but will only process the contents of the SIANCILLARY telecommand message.

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. The boot HKP telemetry packets are indicated by a fixed application ID value of 0x200. Because of the limited size, only a small amount of information can be downlinked while the boot shell is executing. For the SIU, a new HKP packet is prepared at a 4 Hz rate. A new packet is prepared when the 1553 driver indicates that the last telemetry block has been transmitted to the SC.

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

The boot shell HKP packets contain a boot software mode indicator, which provides status on the state that the boot code is in. The primary boot code mode values are shown below.

Figure 34 - 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 will remain in this mode until the file upload is complete or has been canceled.
4	The boot code has received a File Upload Commit command targeting SIB EEPROM and is writing the file upload data into SIB EEPROM memory storage.
5	Reserved.
6	The boot code is booting the RTOS.

The primary boot shell is able to dump the contents of memory in each output telemetry packets. There are two memory dumps: the foreground memory dump and the background memory dump.

The foreground memory dumps commence as the result of a Boot Memory Dump Start operational command. The memory dump continues in the HKP telemetry output until either the requested amount of memory has been dumped or a Boot Memory Dump Cancel operational command is issued to the boot shell.

The background memory dump is active at any time when the foreground memory dump is not active. The background memory dump will continuously dump the contents of the boot diagnostics area (address 0x0FF80 -> 0x0FFFF).

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 will contain only this HKP packet, with the remainder of the telemetry block set to 0. The SC bus controller will attempt to read a new telemetry block at 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. Diagnostic errors are errors encountered during the startup process that cannot be reported immediately because the command and telemetry boot shell has not been started yet. Operational errors are encountered during command operations and are reported in the boot code 1553 telemetry output. Exceptions are 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 on 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 also use this area to report critical errors that may not be able to be reported by the normal telemetry channels. Both the LAT boot code and LAT applications may also share this area to report diagnostics information that must be preserved across reboots. Figure 36 shows the layout of the primary boot code diagnostics area.

Figure 36 – SDRAM Boot Diagnostics Area

Offset (bytes)	Mnemonic	Description	Data Source
0x00	DIAGS_PRIM_BOOT_FLAGS	Primary Boot Flags. Described in Figure 38.	Set by startup diagnostics or reboot request.
0x04	DIAGS_SEC_BOOT_FLAGS	Secondary Boot Flags. Described in Figure 40.	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	PBC, SBC or application exception handler.
0x10	DIAGS_EXC_SRR0_REG	Exception SSR0 Register Contents	PBC, SBC or application exception handler.
0X14	DIAGS_EXC_SRR1_REG	Exception SSR1 Register Contents	PBC, SBC or application exception handler.
0X18	DIAGS_EXC_DAR_REG	Exception DAR Register Contents	PBC, SBC or application exception handler.
0X1C	DIAGS_EXC_DSISR_REG	Exception DSISR Register Contents	PBC, SBC or application exception handler.
0x20	DIAGS_PCI_STATUS2_REG	Exception PCI Status 2 Register Contents	PBC, SBC or application exception handler.
0x24	DIAGS_MEM_STATUS_REG	Exception Memory Status Register Contents	PBC, SBC or application exception handler.
0x28	DIAGS_EXC_TASK_ID	Exception task ID.	SBC or application exception handler.
0x2C	DIAGS_MT_RESULTS_1	First Pass Memory Test Results	Startup diagnostics.
0x30	DIAGS_MT_ADDR_1	First Pass Memory Test Failure Address	Startup diagnostics.
0x34	DIAGS_MT_ACTUAL_1	First Pass Memory Test Failure Data	Startup diagnostics.
0x38	DIAGS_MT_RESULTS_2	Second Pass Memory Test Results	Startup diagnostics.
0x3C	DIAGS_MT_ADDR_2	Second Pass Memory Test Failure Address	Startup diagnostics.
0x30	DIAGS_MT_ACTUAL_2	Second Pass Memory Test Failure Data	Startup diagnostics.
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	Startup diagnostics.
0x54-0x5C	DIAGS_BIST_PPCI	PPCI BIST Results	Startup diagnostics.
0x5C	Reserved		

0x60-0x7C	DIAGS_APP_INFO	Application Information	Secondary boot or Application.

The Primary Boot Flags member provides a summary of the reboot cause. The bit fields are shown below (bit 31 is LSB).

Figure 38 – Primary Boot Flags

Bits	Mnemonic	Description	Data Source/Destination
0 - 1		Reserved.	
2	DIAGS_PBF_HARD_RESET	This flag is adopted from the standard VxWorks boot flags. When set, it indicates that the CPU was reset in hardware. If this flag is set, the other flags indicating software settings will be forced to known values. The set of hardware reset cause flags (bits 8 – 11) may be interrogated to find the source of the hardware reset.	Set by PBC startup diagnostics.
3	DIAGS_PBF_BOOT_FAST	This flag is adopted from the standard VxWorks boot flags. When set, it indicates that the primary boot code should immediately load the default RTOS image and start the secondary boot process. This flag is set to '0' in the event of a hardware reboot.	Input to PBC boot shell.
4 - 7		Reserved.	
8	DIAGS_PBF_POR_RESET	A hardware reset flag indicating that the RAD750 CPU board was reset by the external power on reset signal. This flag is copied from the corresponding bit in the RAD750 BIST Status Register.	Set by PBC startup diagnostics based on data in BIST status register.
9	DIAGS_PBF_PPCI_ERR_RESET	A hardware reset flag indicating that the RAD750 CPU board was reset by an internal error in the PPCI bridge chip. This flag is copied from the corresponding bit in the RAD750 BIST Status Register.	Set by PBC startup diagnostics based on data in BIST status register.
10	DIAGS_PBF_PPCI_RESET	A hardware reset flag indicating that the RAD750 CPU board was reset by the PPCI on chip bus software reset command. This flag is copied from the corresponding bit in the RAD750 BIST Status Register	Set by PBC startup diagnostics based on data in BIST status register.

11	DIAGS_PBF_JTAG_RESET	A hardware reset flag indicating that the RAD750 CPU board was reset by an external JTAG master interface. This flag is copied from the corresponding bit in the RAD750 BIST Status Register.	Set by PBC startup diagnostics based on data in BIST status register.
12 - 15		Reserved.	
16	DIAGS_PBF_EXC_RESET	The CPU has been rebooted as the result of a software instruction exception. This flag is set to '0' in the event of a hardware reboot.	Set by PBC startup diagnostics or exception vector.
17	DIAGS_PBF_SEC_RESET	The CPU has been rebooted as the result of a secondary boot process critical error. This flag is set to '0' in the event of a hardware reboot.	Set by SBC.
18	DIAGS_PBF_APP_RESET	The CPU has been reboot intentionally by one of the applications. The DIAGS_APP_INFO may contain additional information.	
19 – 23		Reserved.	
24	DIAGS_PBF_MEM_ERROR	Indicates a memory test failure. Set by the startup diagnostics memory test.	Set by PBC startup diagnostics.
25-30		Reserved.	
31	DIAGS_PBF_RES_MEM_TEST	Test reserved memory. If the startup diagnostics see this flag set, a memory test will be run on reserved SDRAM (addresses over 0x00800000) are tested. This allows for remediation to be attempted for errors detected in high memory.	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 40 - Secondary Boot Flags

Bits	Mnemonic	Description
0 - 1	DIAGS_SBF_RTOS_SOURCE	RTOS image source flag indicates the location from which the RTOS executable was taken when starting the secondary boot process. 0 = RTOS image was sourced from RAM temporary area; 1 = RTOS image was sourced from SIB EEPROM lower bank boot partition; 2 = RTOS image was sourced from SIB EEPROM upper bank boot partition.

1 - 2	DIAGS_SBF_MODO_SOURCE	Flags indicate 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 lower bank boot parition; 2 = SSB Module 0 should be sourced from SIB EEPROM upper bank boot parition
3 - 4	DIAGS_SBF_MOD1_SOURCE	Flags indicate 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 lower bank boot partition; 2 = SSB Module 1 should be sourced from SIB EEPROM lower bank boot partition
5 - 31		Reserved.

The Exception Vector, Exception SSR0 Register, and Exception SSR1 Register members of the diagnostics region are set when the CPU has been rebooted as the result of a software instruction exception. These members will contain valid values when the *SE* primary boot flag is set. Otherwise, the primary boot code will set these members to '0'.

Results of memory tests are included in the diagnostics region. Data is included for a first pass and an optional second pass. The contents of the second pass are only populated if a failure was found in the first pass and remediation was attempted. The contents include a set of test results for each pass, the address of where the first (if any) error was detected and the "read-back" data that was in error.

The results of the memory test include the status of each of the four memory areas tested and a column index (1 indicates problem in least significant column, 8 indicates problem in most significant column). The four memory test regions are described in Figure 6.

The memory test results bit field is described in Figure 41 (bit 31 is the LSB).

Figure 41 - Memory Test Results

Bits	Description
0-3	Results of Low Boot Region test. Result codes described in Figure 42.
4-7	Results of High Boot Region test. Result codes described in Figure 42.
8-11	Results of Boot Diagnostics Region test. Result codes described in Figure 42.
12-15	Results of Reserved Region test. Result codes described in Figure 42.
16-23	Index of column column that was determined to be bad and was swapped out as part of remediation.
24-31	Reserved.

Figure 42 - Memory Test Result Code

Code	Description
0	Memory test was not run.
1	Memory test has passed.
2	Memory test failed in final walking bit step.
3	Memory test failed in first walking bit step.
4	Memory test failed in second walking bit step.
5	Memory test failed in third walking bit step.
6	Memory test failed in fourth walking bit step.
7	Memory test failed in address test step.
8	Memory controller indicated error.
9-14	Reserved.
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 occur before telemetry communications are available. Operational errors 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 error occurs when the primary boot code initialization process encounters a problem that would prevent successful boot and which happen before the boot primary communications 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 43 - Critical Startup Error PID Settings

PID 5 Value	PID 6 Value	Status
0	0	No error.
0	1	Memory test indicates hard error in boot region of SDRAM.

1	0	The primary communications channel (1553 or LCB) could not be initialized.
1	1	Reserved

When the primary boot code reports a critical startup error, the PID lines are set to the appropriate status values, and the boot code enters a stall loop, doing nothing more that resetting the watchdog timer. This guarantees that the error condition status is available for users to examine. Once the boot code enters the stall state, the crate must either be hard reset or power cycled to continue.

3.1.1 Operational Errors

Operational errors as the result of primary boot code normal procedures 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 will maintain its value for examination until a new Boot Error Dump telecommand is received. This will remove the next error report from a primary boot code internal queue and place it in the next outgoing HKP telemetry packet. If no new errors are available for report, the error report word in the HKP packet is set to an appropriate value.

3.2 Exceptions

The PBC will implement exception handlers for both the PPC and EMC processors. All possible exceptions on both machines will have an exceptions handler, even though the same exception handler code may be used more many or most exceptions.

3.2.0 EMC Exceptions

Exception handling on the EMC processor is different from the PPC exceptions in the fact that these exceptions exist in SUROM and are the exceptions that must execute while the PPC 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 44 - EMC Exception Vectors

Vector	Source
Vector 0	Operation request from CPU.
Vector 1	No assigned use.
Vector 2	No assigned use.

Vector 3	No assigned use.
Vector 4	No assigned use.
Vector 5	No assigned use.
Vector 6	EMC invalid instruction or access error. Unmaskable.
Vector 7	Critical error or watchdog timer expired. Unmaskable.

Vector 0 allows for 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 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 will be stored in RAM at the address 0x0000FF80. 128 will be reserved to share data used in the exception.

Vector 6 is an unmaskable exception to the EMC. It is active if an illegal EMC instruction is encountered. When this vector is activated, a hardware reset of the EMC and PPC is activated.

Vector 7 is an unmaskable exception to the EMC. It is active if PPC watchdog timer expires or if a "critical" error in encountered in the PPC or PPCI bridge. When this vector is activated, a hardware reset of the EMC and PPC is activated.

The LAT flight software does not use vectors 1-5. If any of these vectors are activated, a hardware reset of the EMC and PPC is activated.

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 will provide exception handlers in RAM for all exceptions. If may be appropriate for the SBC to use the exception handlers provided by the primary boot code.

3.2.1.0 Startup Exception Handler

While the PBC is running the startup diagnostics, one exception handler is used to process all PPC exceptions. The exception handler is installed for all exception vectors located in the RAD750 SUROM (starting at address 0xFFF00200). This exception vector stores the following information into the boot in 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.

The handler also increments an exception counter kept in the boot diagnostics region. The handler then increments 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 starts running in the boot shell mode 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 0x00000200). The RAM version of the primary boot code exception handler stores the following information into the boot in 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.

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 0xFFF00104 with the boot type code set to "Exception Start" (see Figure 15 for more detail).

4 Appendix A

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

4.0 Boot Telecommands

Boot telecommands direct the boot operations of either the SIU or EPU primary boot code.

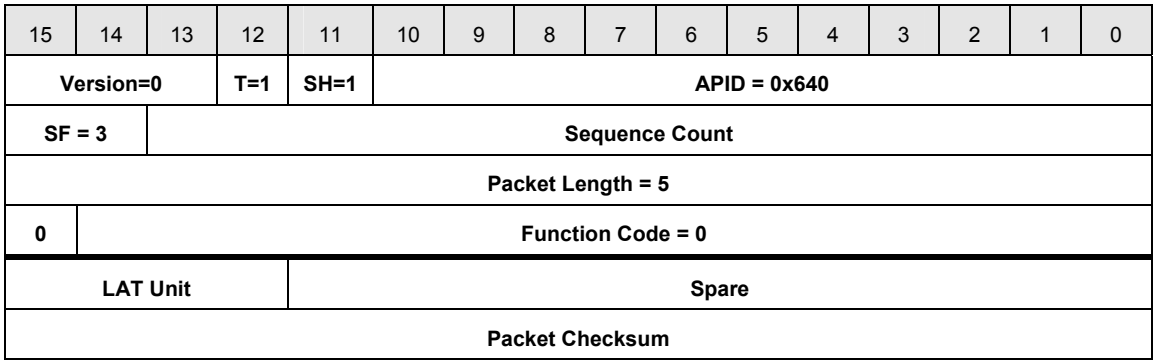
Table 6 - Boot Telecommand Function Codes

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

4.0.0 Boot Start

The Boot Start command is present to provide a harmless no-op command to test the command and telemetry operations of the boot shell.

Figure 45 - Boot Command Start Telecommand Format

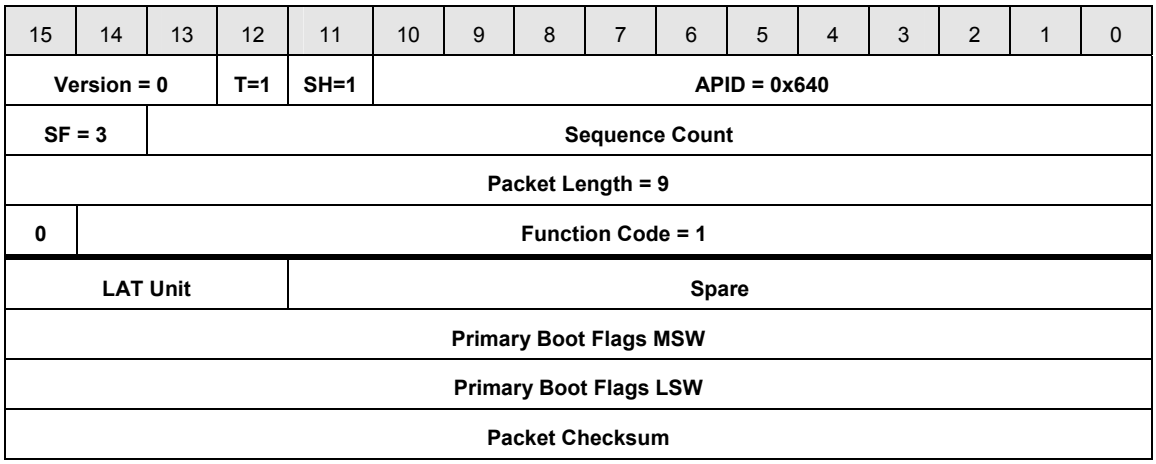


LAT Unit – A code indicating the LAT unit to receive the Boot Start command. Expected values are SIU, EPU_0, EPU_1 or EPU_2.

4.0.1 Boot Reset

The Boot Reset command causes the primary boot code to return to the warm 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 46 - Boot Reset Telecommand Format



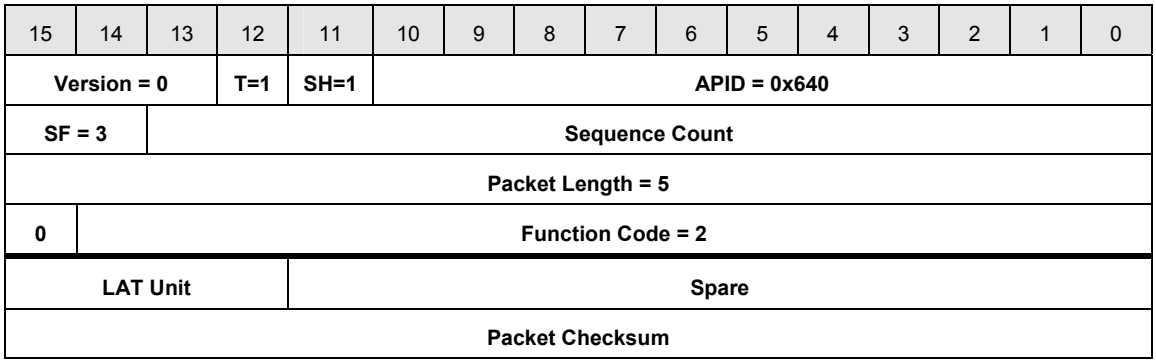
LAT Unit – A code indicating the LAT unit to receive the Boot Reset command. Expected values are SIU, EPU_0, EPU_1 or EPU_2.

Primary Boot Flags – Bit field indicating what actions the boot code should take when restarting. Reference Figure 38 for definitions of this bit field.

4.0.2 Boot Error Dump

The primary boot shell maintains a queue of up to TBD 32-bit error code words that describes error conditions that have been encountered. The error words remain in the queue until forced out into the HKP telemetry packets by this command. This command will cause one error word to be sent in the next available HKP packet. The error code work will remain visible in subsequent HKP packets until this command is issued again. The number of queued error words still outstanding is also contained in the HKP telemetry packet.

Figure 47 - Boot Error Dump Telecommand Format

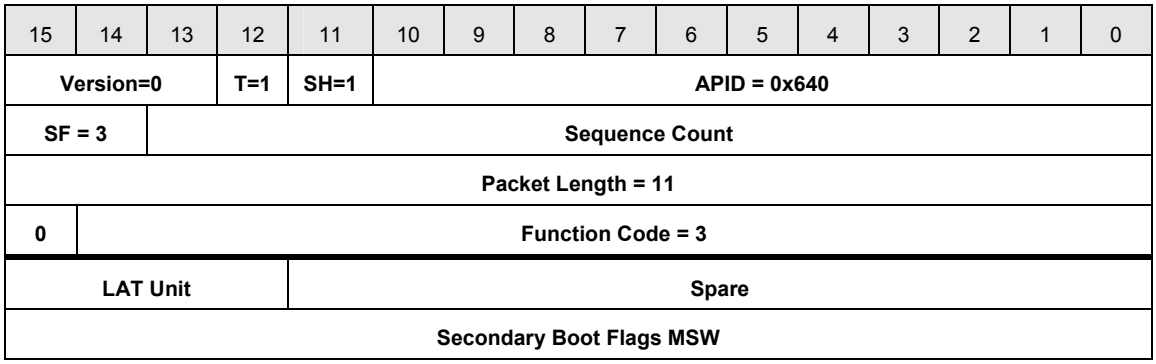


LAT Unit – A code indicating the LAT unit to receive the Boot Error Dump command. Expected values are SIU, EPU_0, EPU_1 or EPU_2.

4.0.3 Boot RTOS Execute

The Boot RTOS Execute command manually boots an RTOS image. The Secondary Boot Flags parameter gives the source of the RTOS executable image, as well as other information which will be 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 will generate an error.

Figure 48 - Boot RTOS Execute Telecommand Format



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Secondary Boot Flags LSW															
Packet Checksum															

LAT Unit – A code indicating the LAT unit to receive the Boot RTOS Execute command. Expected values are SIU, EPU_0, EPU_1 or EPU_2.

Secondary Boot Flags – A bit mask of secondary boot options. See Figure 40 for a definition.

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. Then, the boot shell jumps to the entry point of the RTOS executable, ending the execution of the boot shell itself and beginning the start of the secondary boot process. The Boot RTOS Execute command also has the ability to direct the secondary boot process by specifying the location to use for the second stage boot file modules.

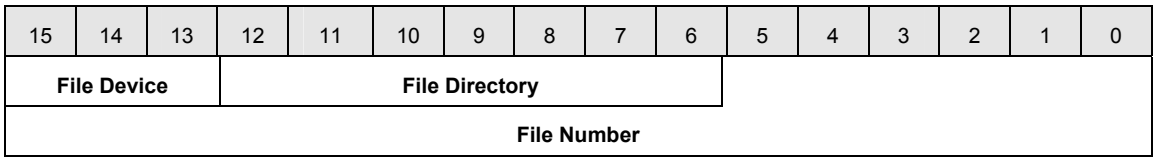
4.1 File Load

Table 10 - File Load Telecommand Function Codes

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

All of the on board LAT files are designated by a 32-bit file ID. The ID is composed of three bit fields as shown below.

Figure 50 - File ID Fields



A file ID word in the File Upload Commit telecommand directs the primary boot code where to store the file object. The File Device field designates a particular storage device. For the primary boot code, which does not support true file systems, the File Device and File Directory fields should always be set to ‘0’. The File Number field enumerates a list of temporary RAM or EEPROM boot partition file storage locations.

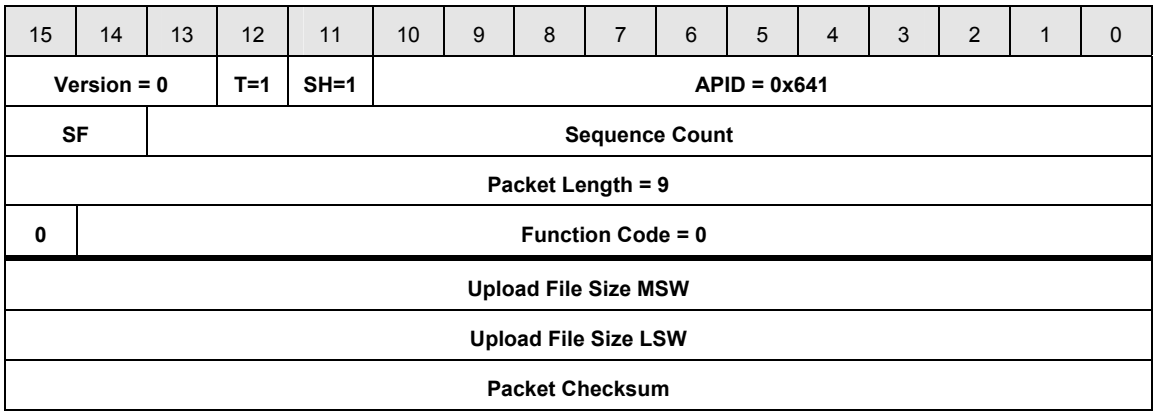
Table 12 - Boot Device File Numbers

File Number	Description
0	Primary boot RTOS SDRAM buffer
1	Secondary boot module 0 SDRAM buffer.
2	Secondary boot module 1 SDRAM 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

4.1.0 File Upload Start

The File Upload Start command initializes the file upload state machine. A new string of File Upload Data packets may be initiated after this command successfully completes. A previous upload must not be in progress when this command is sent. The file data commands that follow the upload start command will place file data in a temporary RAM upload buffer.

Figure 51 - File Upload Start Telecommand Format

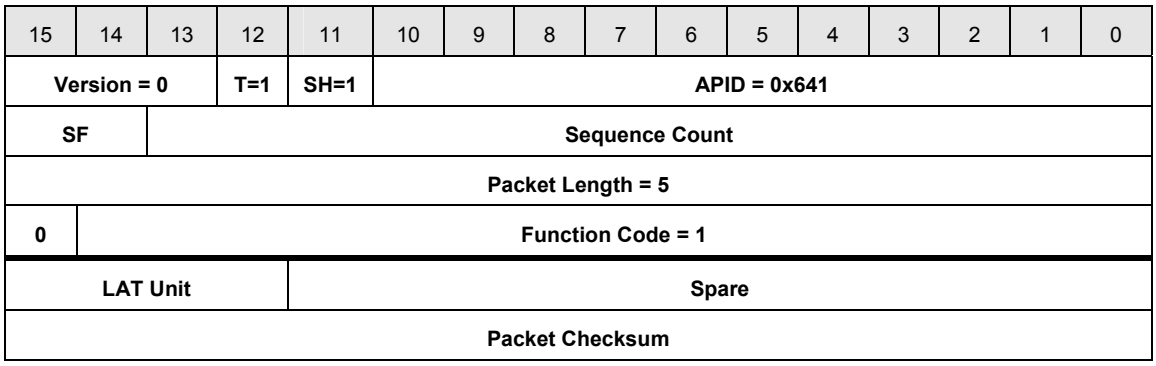


Upload File Size – The size in bytes of file data which will be contained in the entire upload sequence.

4.1.1 File Upload Cancel

The File Upload Cancel command deletes all of the data currently held in the temporary file upload buffer and resets the file upload state machine. This command may be used to cancel an erroneous upload attempt.

Figure 53 - File Upload Cancel Telecommand Format



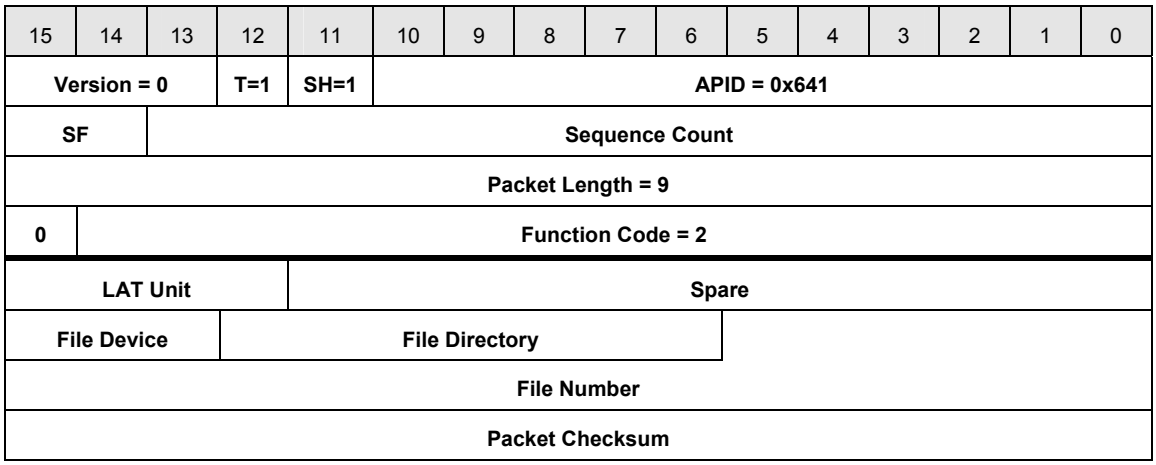
LAT Unit – A code indicating the LAT unit on which the file upload will be cancelled.

4.1.2 File Upload Commit

The File Upload Commit command copies the file data contents from the file upload buffer to the actual device storage location as indicated by the File ID parameter. The file upload packets must have all arrived and in good order; otherwise this command will fail with an error. If the file

to be committed is large and is being written to EEPROM, this command may introduce a significant delay in commanding operations.

Figure 55 - File Upload Commit Telecommand Format



LAT Unit – A code indicating the LAT unit on which the file will be deleted..

File Device – A code indicating the file storage device.

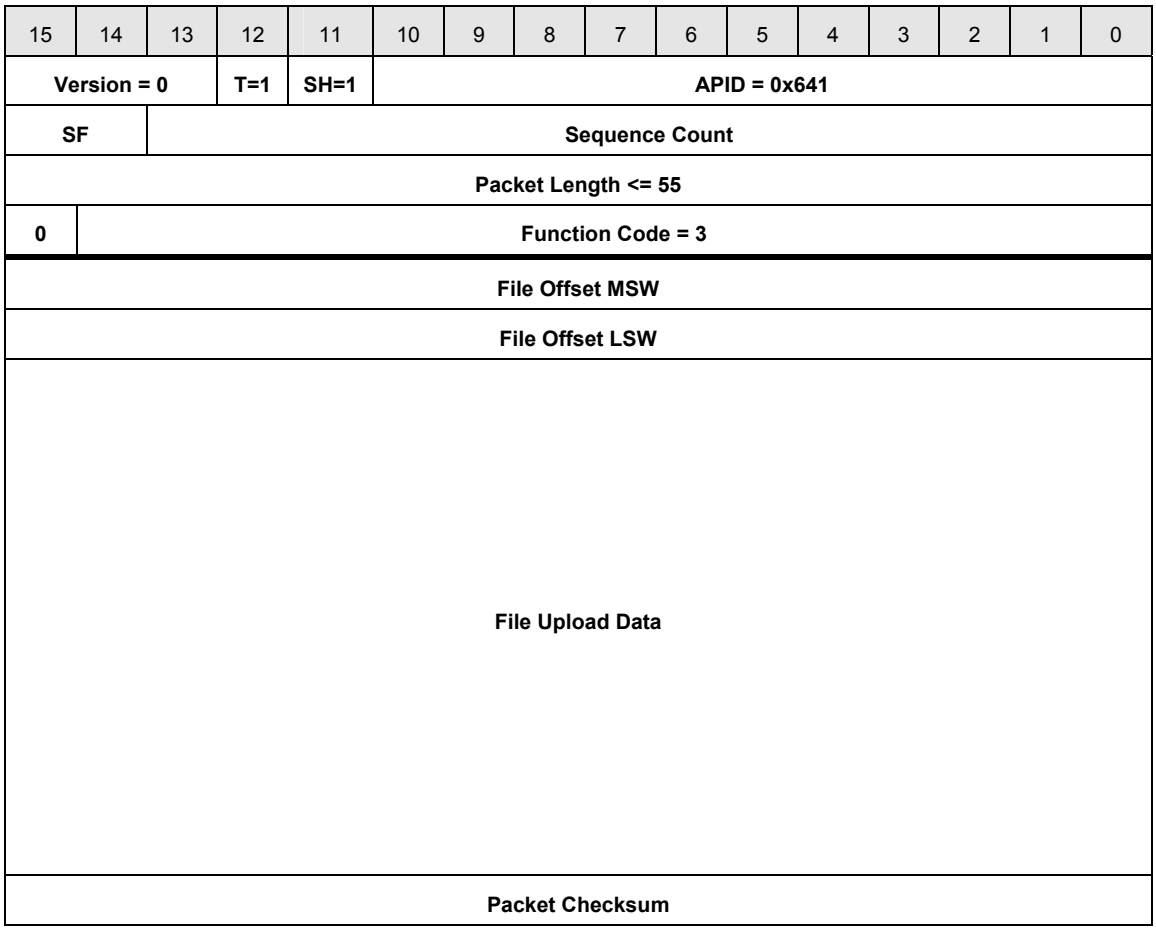
File Directory - A code indicating the file storage directory.

File Number – A code indicating which file in a directory will be written with the contents of the file upload buffer.

4.1.3 File Upload Data

Each upload arrives as a string of File Upload Data telecommand packets. The total size of each upload telecommand packet does not exceed 62 bytes. The file data is collected into a temporary RAM upload buffer.

Figure 57 - File Upload Data Telecommand Format



File Offset – A offset in bytes from the beginning of the file that the first data byte in this packet represents.

Each File Upload Data telecommand packet contains up to 48 bytes actual file data. The file data is extracted sequentially from the original source file and inserted into consecutive upload telecommand packets.

4.3 Memory Load Telecommands

Table 13 - Memory Load Telecommand Function Codes

APID	Function Code	Description
0x642	0	Memory Write. Update memory contents at the selected address(es).
	1	PCI Device Header Write. Writes a value into a PCI Header Register.
	2	Processor Register Write. Writes a value into a Processor Register.

4.3.0 Memory Write

The memory write command writes the contents of the telecommand Memory Write Data Words into the designated region of memory. The boot shell can only write memory in units of 32-bit words.

Figure 64 - Memory Write 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 = 0x642**										
SF = 3			**Sequence Count**												
Packet Length <= 55															
0	**Function Code = 0**														
LAT Unit					**Spare**										
Memory Write Word Count															
Memory Write Address MSW															
Memory Write Address LSW															
Memory Write Data Word 0 MSW															
Memory Write Data Word 0 LSW															
Memory Write Data Word 1 MSW															
Memory Write Data Word 1 LSW															
Memory Write Data Word 2 MSW															
Memory Write Data Word 2 LSW															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Memory Write Data Word 3 MSW															
Memory Write Data Word 3 LSW															
Memory Write Data Word 4 MSW															
Memory Write Data Word 4 LSW															
Memory Write Data Word 5 MSW															
Memory Write Data Word 5 LSW															
Memory Write Data Word 6 MSW															
Memory Write Data Word 6 LSW															
Memory Write Data Word 7 MSW															
Memory Write Data Word 7 LSW															
Memory Write Data Word 8 MSW															
Memory Write Data Word 8 LSW															
Memory Write Data Word 9 MSW															
Memory Write Data Word 9 LSW															
Memory Write Data Word 10 MSW															
Memory Write Data Word 10 LSW															
Packet Checksum															

LAT Unit – A code indicating the LAT unit that will update the contents of the memory locations. Expected values are SIU, EPU_0, EPU_1 or EPU_2.

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

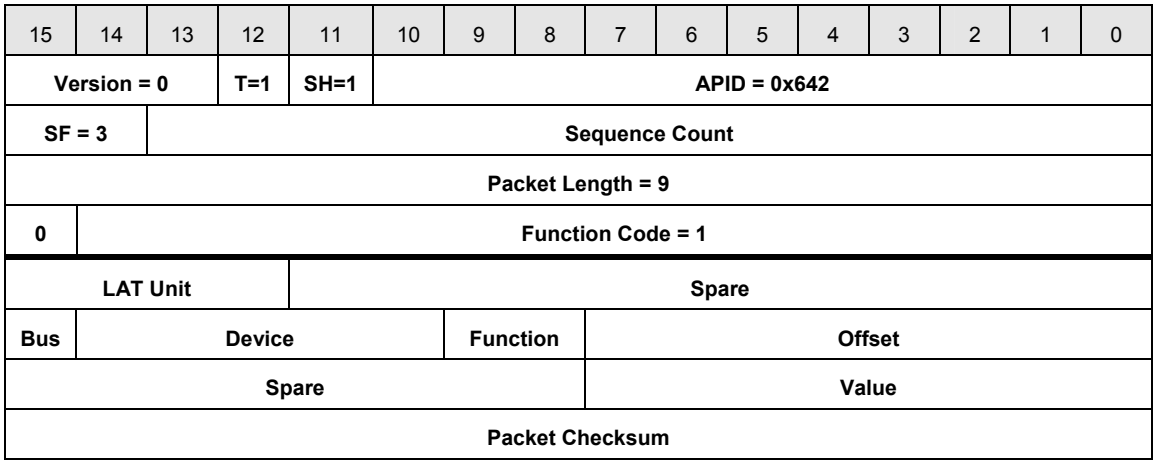
Memory Write Address – The address of the memory to write. Because the memory write format only supports 32-bit alignment, this address must start the dump region on a four byte boundary.

Memory Write Data – The 1 to 11 32-bit data values to write into memory.

4.3.1 PCI Device Header Write

The PCI Device Header Write command writes a byte value into the designated PCI Header Register.

Figure 65 - PCI Device Header Write Telecommand Format



LAT Unit – A code indicating the LAT unit that will update the contents of the specified PCI Device Header Register.

Bus – PCI Bus Select. Expected value: 0.

Device – PCI Device Select. Will vary based on system configuration (test vs flight)

Function – PCI Device Function Select. Expected value: 0.

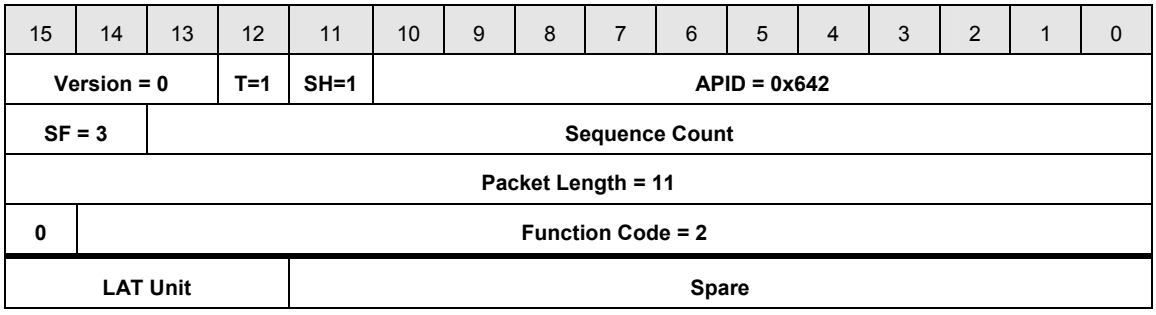
Offset – PCI Device Register Select. Dependent on device.

Value - 8 bit value to write into the selected PCI header register.

4.3.2 Processor Register Write

The Processor Register Write Command writes a 32 bit value into the designated Processor Register.

Figure 66 - Processor Register Write Telecommand Format



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Register Select															
Register Value MSW															
Register Value LSW															
Packet Checksum															

LAT Unit – A code indicating the LAT unit that will update the contents of the specified PCI Header Register..

Register Select – The processor register to load.

Register Value – Register Value. Value constraints will vary based on selected register.

LAT Unit –

LAT Unit – code ndicating the LAT unit that will dump the File Header. Expected

4.4 Memory Dump Telecommands

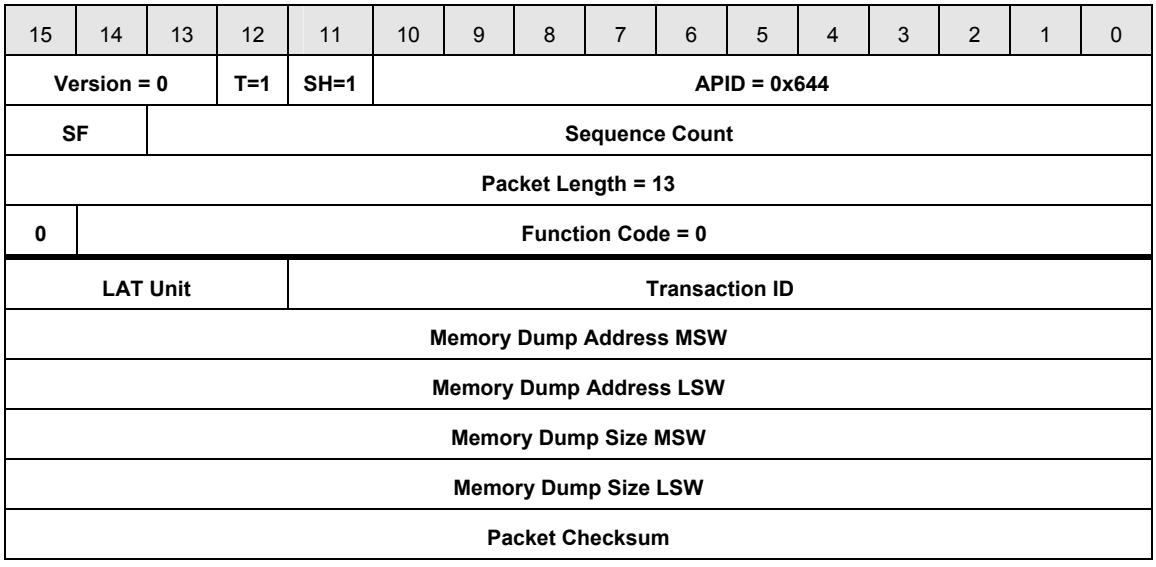
Table 15 - Memory Dump Telecommand Function Codes

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 memory dump.
	5	PCI Device Header Dump. Dump the selected PCI device header.
	6	Processor Register Dump. Dump the processor general purpose and control registers.

4.4.0 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.

Figure 71 - Memory Data Dump Telecommand Format



LAT Unit – A code indicating the LAT unit that will dump the Memory Contents.

Transaction ID – A value that will be echoed in the memory dump telemetry. This value should be used by the ground system to help differentiate between the downlink results of subsequent memory dump requests.

Memory Dump Address – The starting address of the memory region to dump. The address must be 32-bit aligned.

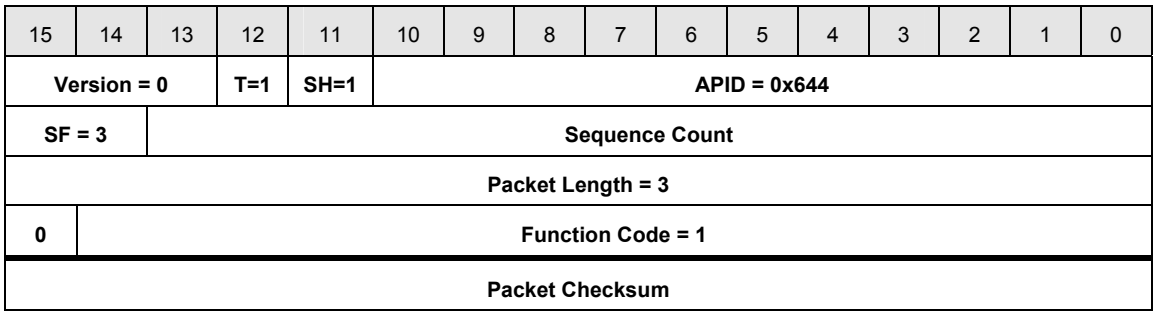
Memory Dump Size – The number of 32-bit words to dump.

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 data dump map is maintained on board to designate the addresses of the legal memory dump data regions and the constraints of those regions.

4.4.1 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 currently not in progress.

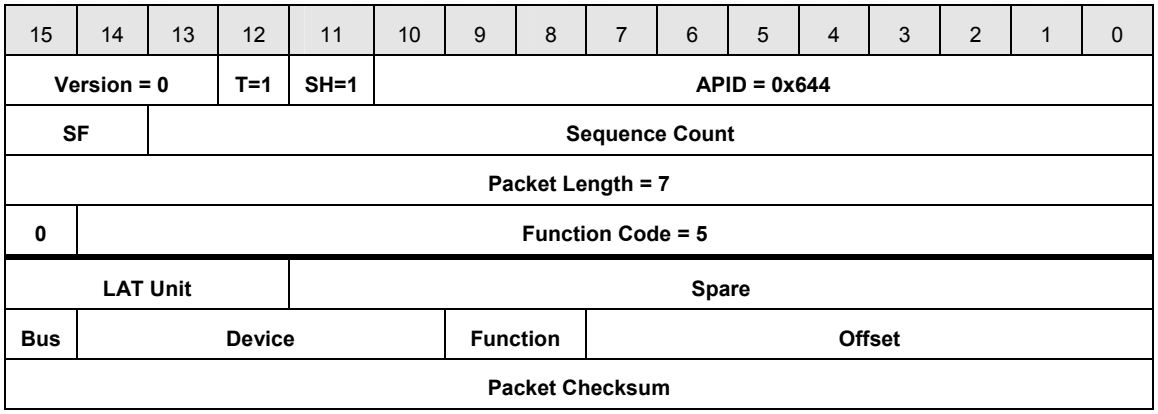
Figure 72 - Memory Dump Cancel Telecommand Format



4.4.2 PCI Device Header Dump

The PCI Device Header Dump telecommand causes a telemetry packet containing the selected device header to be sent to out.

Figure 76 - PCI Device Header Dump Telecommand Format



LAT Unit – A code indicating the LAT unit that will dump the selected PCI Device Header.

Bus – PCI Bus Select. Expected value: 0.

Device – PCI Device Select. Will vary based on system configuration (test vs flight).

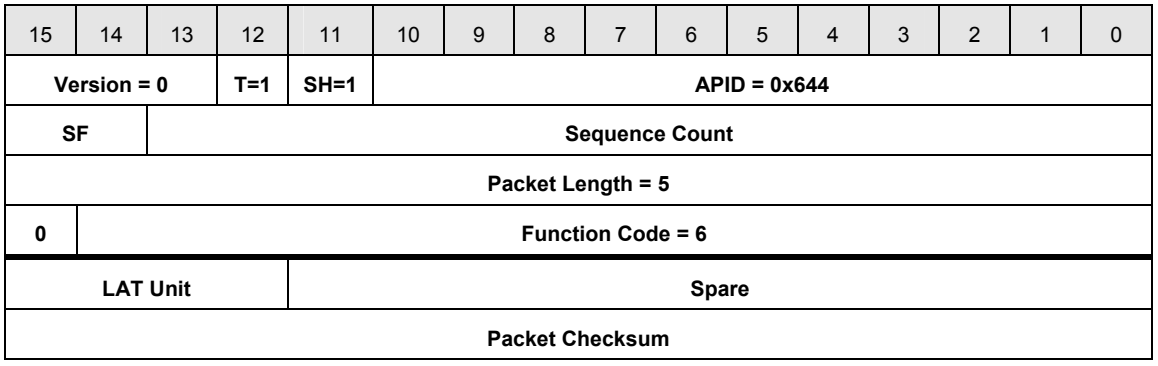
Function – PCI Device Function Select. Expected value: 0.

Offset – PCI Device Register Select. Dependent on device.

4.4.3 Processor Register Dump

The Process Register Dump telecommand causes a telemetry packet containing the processor general purpose registers and control registers to be sent to out.

Figure 77 - Processor Register Dump Telecommand Format



LAT Unit – A code indicating the LAT unit that will dump the Processor Register contents.

4.5 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 78 – Boot Housekeeping Telemetry Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Version = 0			**T=0**	**SH=1**	**APID = 0x200**										
SF=3	**Sequence Count**														
Packet Length = 109															
Timestamp Seconds MSW															
Timestamp Seconds LSW															
Timestamp Sub-Seconds MSW															
Timestamp Sub-Seconds LSW															
Software Mode															
Total Error Count															
Queued Error Count															
Error Word MSW															
Error Word LSW															
Received Telecommand Count															
Boot Telecommand Acceptance Count															
File Telecommand Acceptance Count															
Boot Telecommand Sequence Count															
Last Command APID												**Last Command Function Code**			
Latest Error Word MSW															
Latest Error Word LSW															
File Upload State															
File Upload Packet Count															
Latest File Upload Error MSW															
Latest File Upload Error LSW															
Memory Dump Word Count															
Memory Dump Address MSW															
Memory Dump Address LSW															
Memory Dump Data Word 0 MSW															
Memory Dump Data Word 0 LSW															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Memory Dump Data Word 1 MSW															
Memory Dump Data Word 1 LSW															
Memory Dump Data Word 2 MSW															
Memory Dump Data Word 2 LSW															
Memory Dump Data Word 3 MSW															
Memory Dump Data Word 3 LSW															
Memory Dump Data Word 4 MSW															
Memory Dump Data Word 4 LSW															
Memory Dump Data Word 5 MSW															
Memory Dump Data Word 5 LSW															
Memory Dump Data Word 6 MSW															
Memory Dump Data Word 6 LSW															
Memory Dump Data Word 7 MSW															
Memory Dump Data Word 7 LSW															
Memory Dump Data Word 8 MSW															
Memory Dump Data Word 8 LSW															
Memory Dump Data Word 9 MSW															
Memory Dump Data Word 9 LSW															
Memory Dump Data Word 10 MSW															
Memory Dump Data Word 10 LSW															
Memory Dump Data Word 11 MSW															
Memory Dump Data Word 11 LSW															
Memory Dump Data Word 12 MSW															
Memory Dump Data Word 12 LSW															
Memory Dump Data Word 13 MSW															
Memory Dump Data Word 13 LSW															
Memory Dump Data Word 14 MSW															
Memory Dump Data Word 14 LSW															
Memory Dump Data Word 15 MSW															
Memory Dump Data Word 15 LSW															

5 Appendix B

This appendix contains diagrams referred to by the text in this document.

Figure 79- Boot Shell Basic State Machine

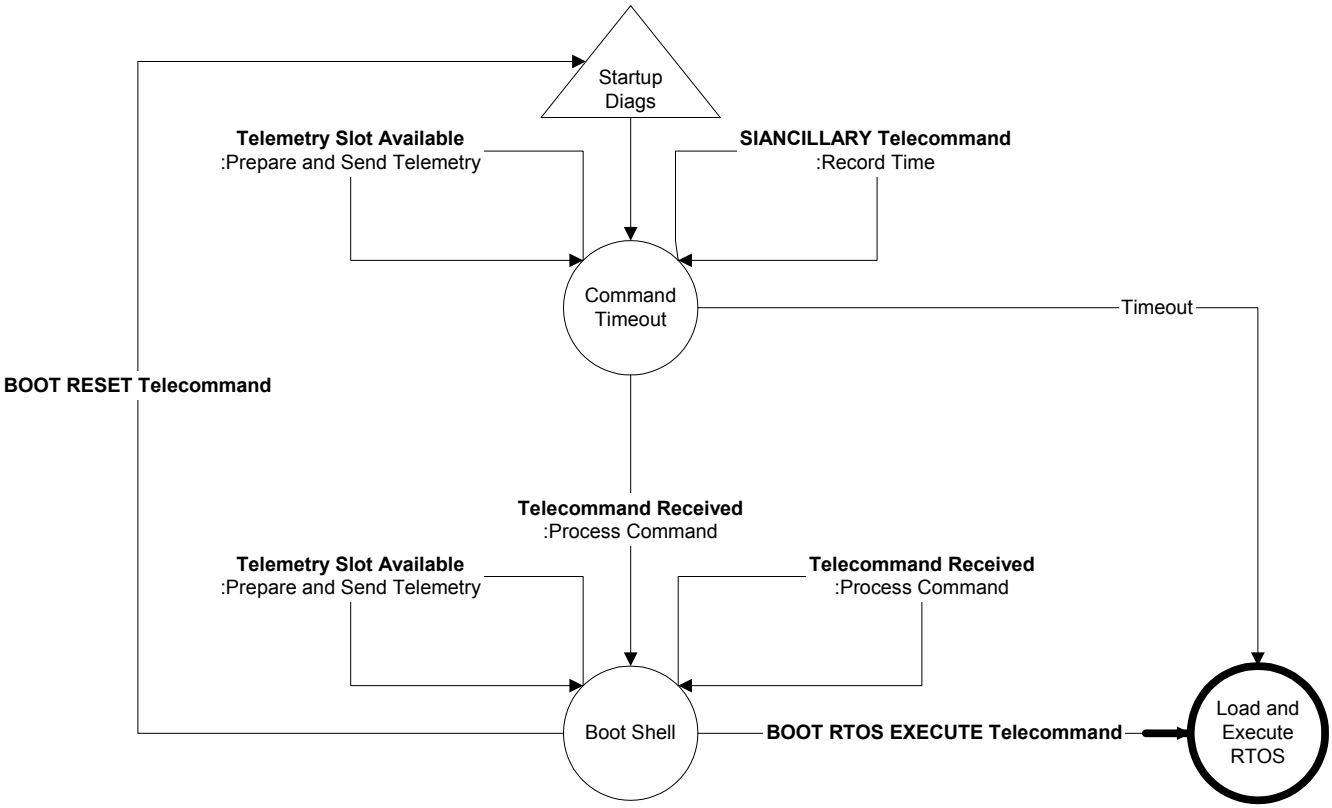


Figure 80 - Boot Shell File Load State Machine

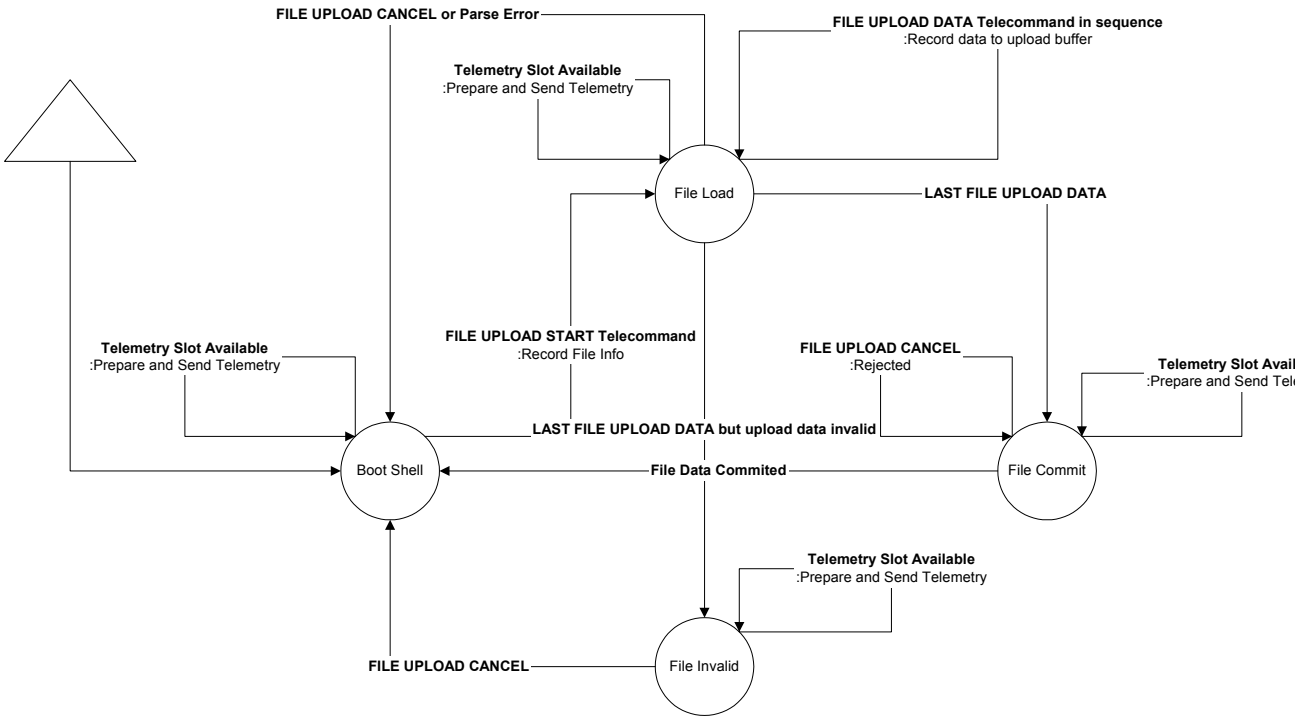


Figure 81 - Boot Shell Memory Write State Machine

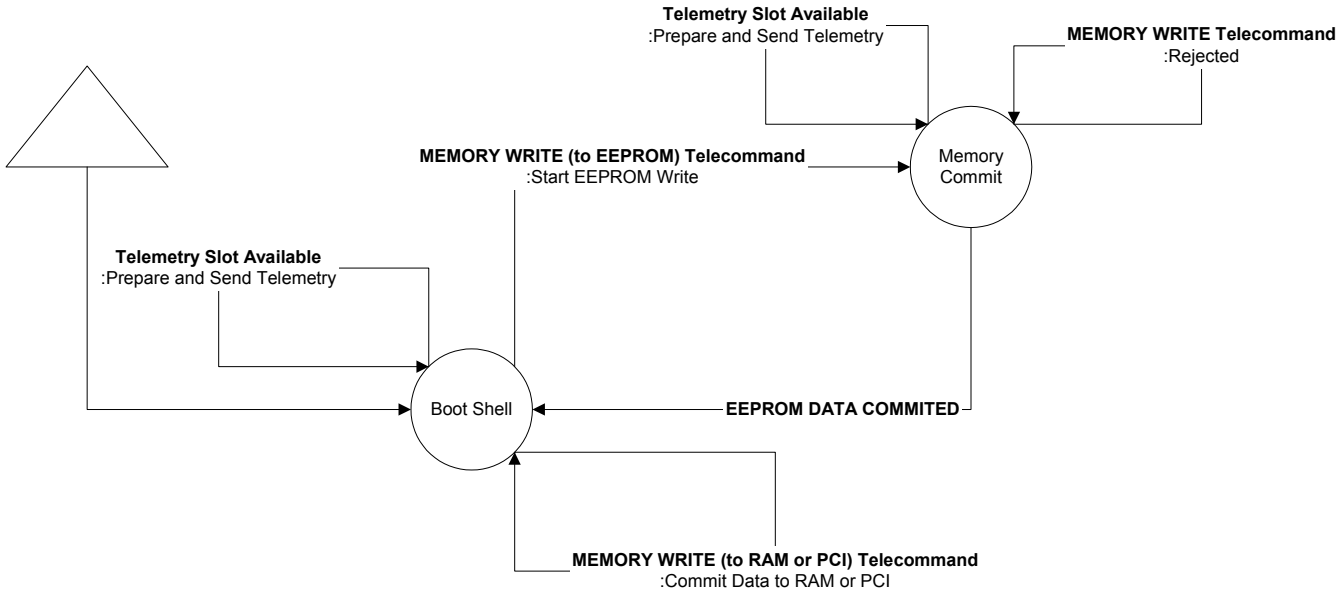


Figure 82 - EMC Boot Flow

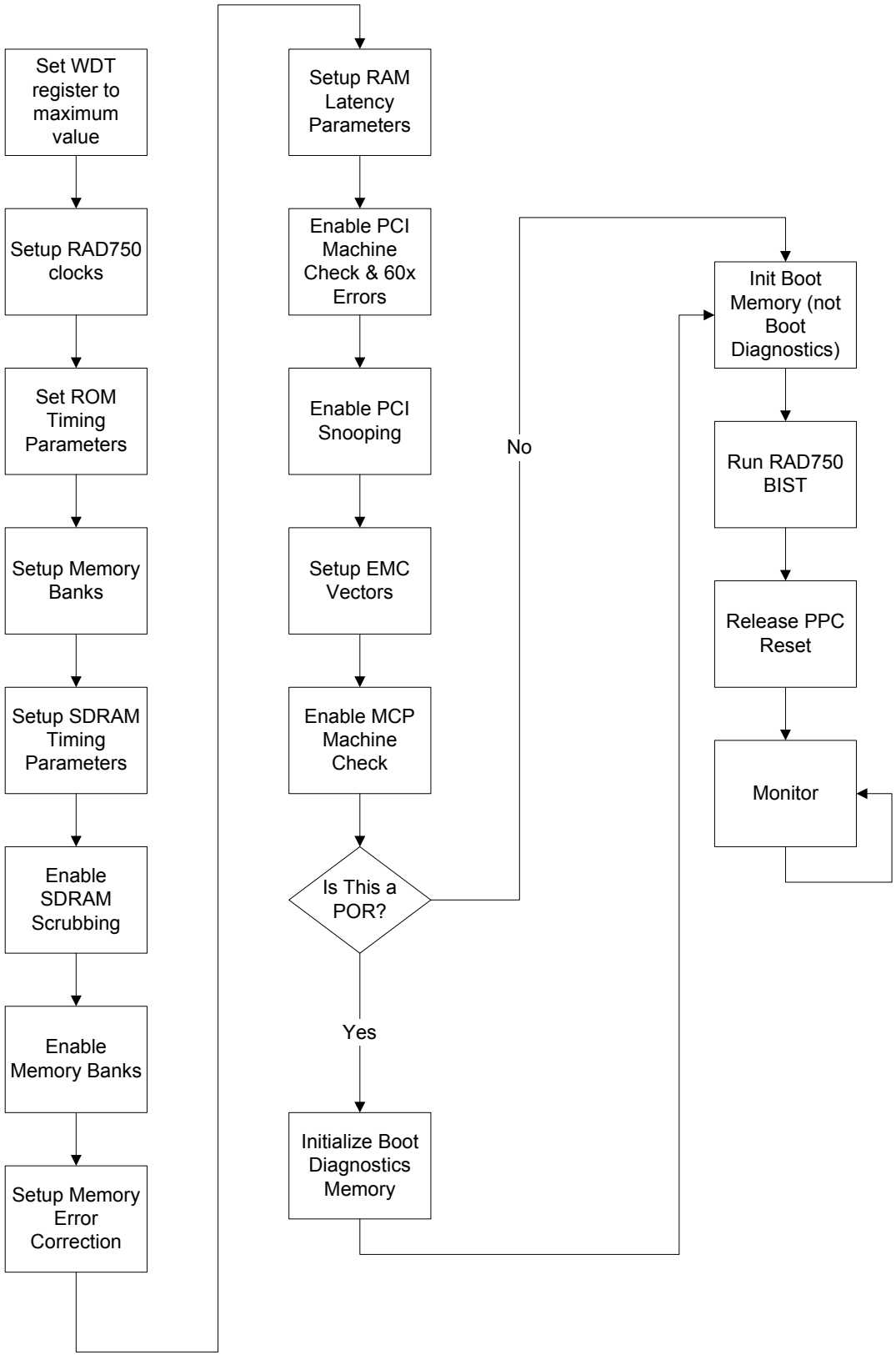


Figure 83 - PPC Boot Flow

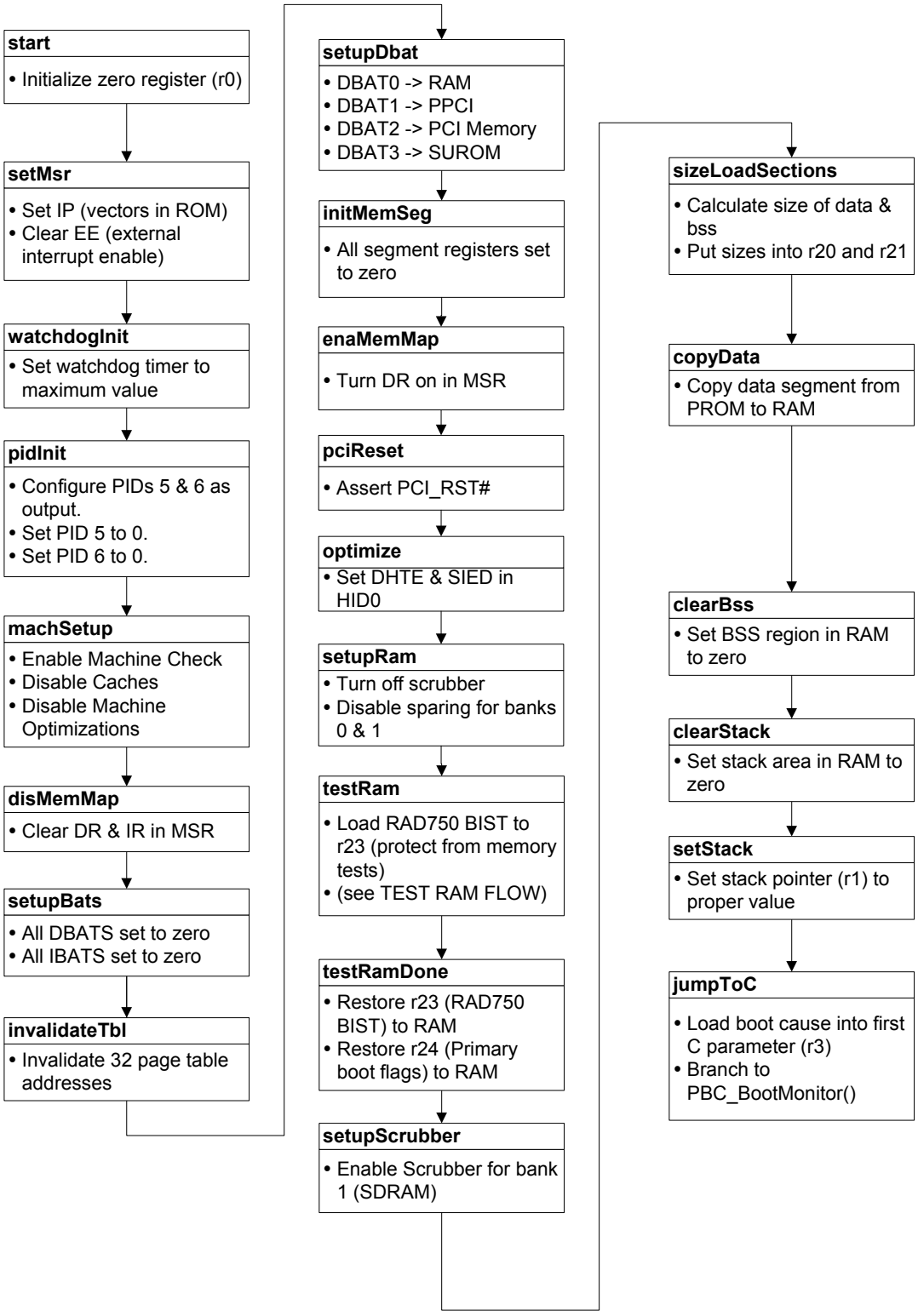


Figure 84 - Cold Boot Flow

COLD START FLOW

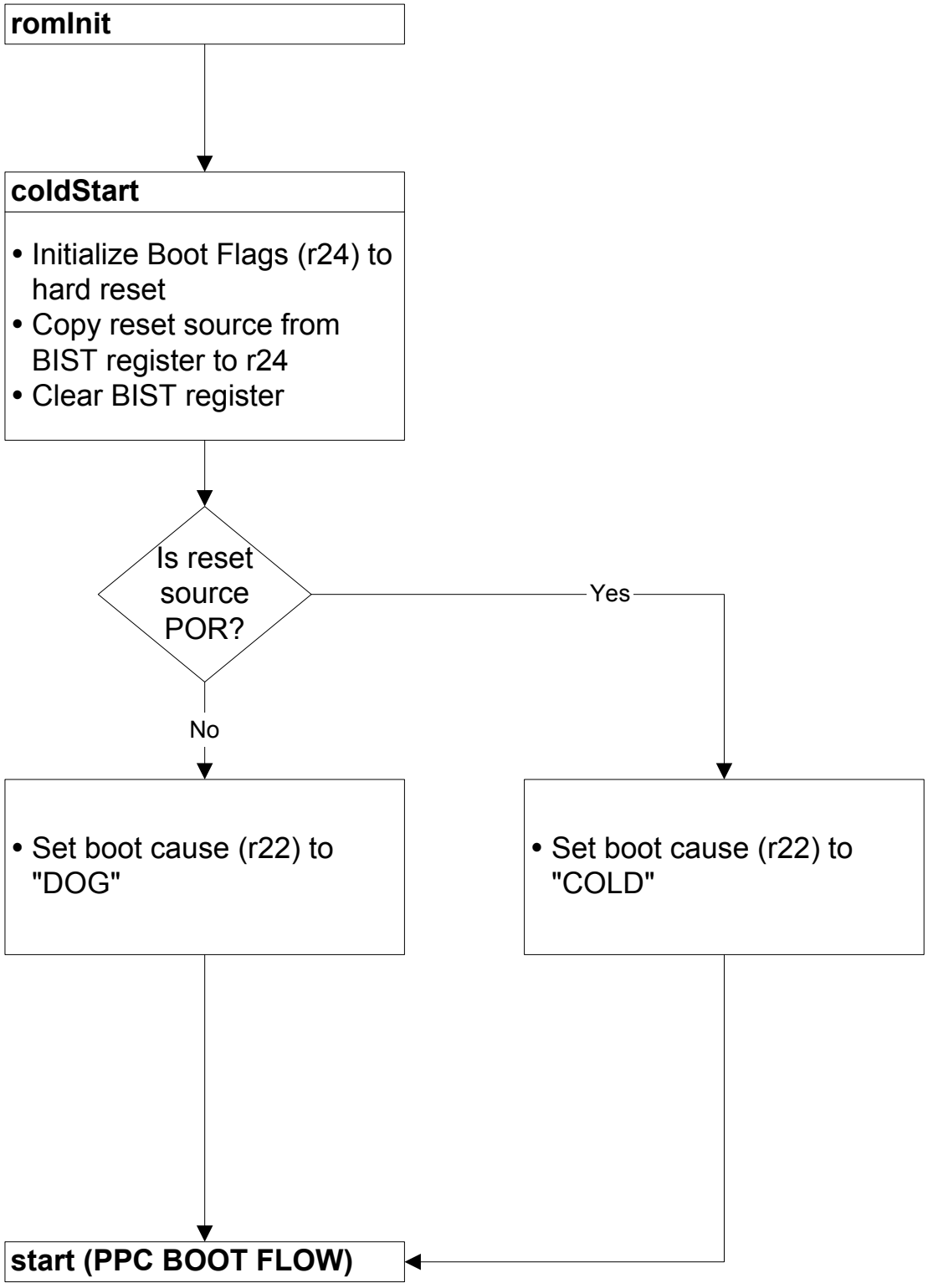


Figure 85 - Warm Boot Flow

WARMSTART FLOW

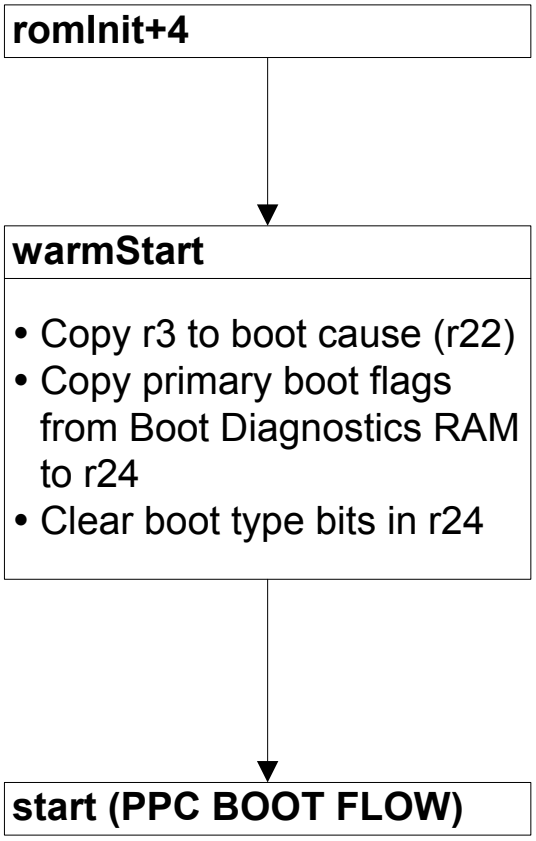


Figure 86 - Memory Test Flow