GLAST Large Area Telescope

Instrument Flight Software
EM2 Design Review
29 January 2004

Functions & Packages I

D.Wood
Naval Research Laboratory / Praxis, Inc.

dwood@xip.nrl.navy.mil

Primary Boot Code (PBC)

Summary of PBC Requirements

- All PBC code and data shall reside in the RAD750 SUROM.
- The PBC code load shall support the SIU and EPU.
- The PBC shall use minimal processor resources.
- PBC shall initialize the RAD750 CPU, memory controller, and PCI bridge to a known state.
- PBC shall attempt to execute SBC if no commands are received on the external interface after 10 minutes from reset.
- The PBC shall provide the capability to reload the SIB EEPROM.
- The PBC shall generate housekeeping telemetry.
- The PBC shall process the SIANCILLARY command packet from the spacecraft for timing.
- The PBC shall recover from errors and exceptions.

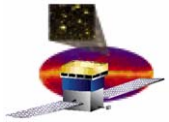
- Detailed requirements are kept in LAT-TD-1806.

Bridge Chip Initialization

- The bridge chip contains an embedded micro-controller (EMC) used to initialize the PPCI bridge chip.
- The EMC code is resident in the SUROM.
- The EMC software configures the PPCI bridge chip memory controller and PCI interface and starts the PPC processor.
- The EMC software also provides exception and error processing support, including the watchdog timer.
- Based on BAE SUROM with GLAST specific changes:
  - Removed EMC-driven DMA service.
  - Turn on memory scrubber in EMC.
  - Removed X2000 SAVE_RAM feature.
  - Change RAM initialization to match PBC needs (first 8 megabytes).
  - Initialize EMC watchdog timer to known value.

Power PC Processor Initialization

- First code to run on Power PC.
- Written in PPC assembly language.
- Overview of processing:
  - Setup memory management unit. The PPC 4 DBATS are used to cover RAM, PPCI registers, PCI memory and SUROM.
  - Configure PPC processor features.
  - Test 8 MB RAM used by PBC on cold boot. Attempt one level of remediation if failures are detected.
  - Test 128MB RAM used by application on request. Attempt one level of remediation if failures are detected.
  - Copy PBC data segment from SUROM to RAM.
  - Initialize BSS segment and stack.
  - Start boot shell (first C code).



Memory Test

- Memory tested in 4 segments: Boot Diagnostics Area, Low Boot Memory, High Boot Memory and Application Memory.
- Tests memory addressing of each 32-bit value and each bit in each memory location.
- On first failure, PID is set to indicate failure, the spare nibble column is swapped in and the test is restarted.
- On second failure, the PID remains set, the test stops and returns to attempt to run PBC code.
- Test results are stored in Boot Diagnostics Area.

Boot Shell Configuration

- The PBC code is required to support the EPU and SIU on a single flight load. The "personality" of the machine is stored in a special area in SUROM. The PBC code will use this value to choose which interface (1553 or LCB) to use for command and telemetry interface.
- The PBC code will configure the devices on the PCI bus.
- Based on the personality stored in SUROM, the PBC will configure the appropriate interface for use as the command and telemetry link.

Boot Shell

- Boot shell is main command processing code.
- Polls active communications interface for commands.
- Sends boot housekeeping telemetry
  - When scheduled on 1553 interface
  - Every 250ms on LCB interface
- Implements initial command timeout period. If no command is received in 10 minutes after boot, PBC will proceed to secondary boot.

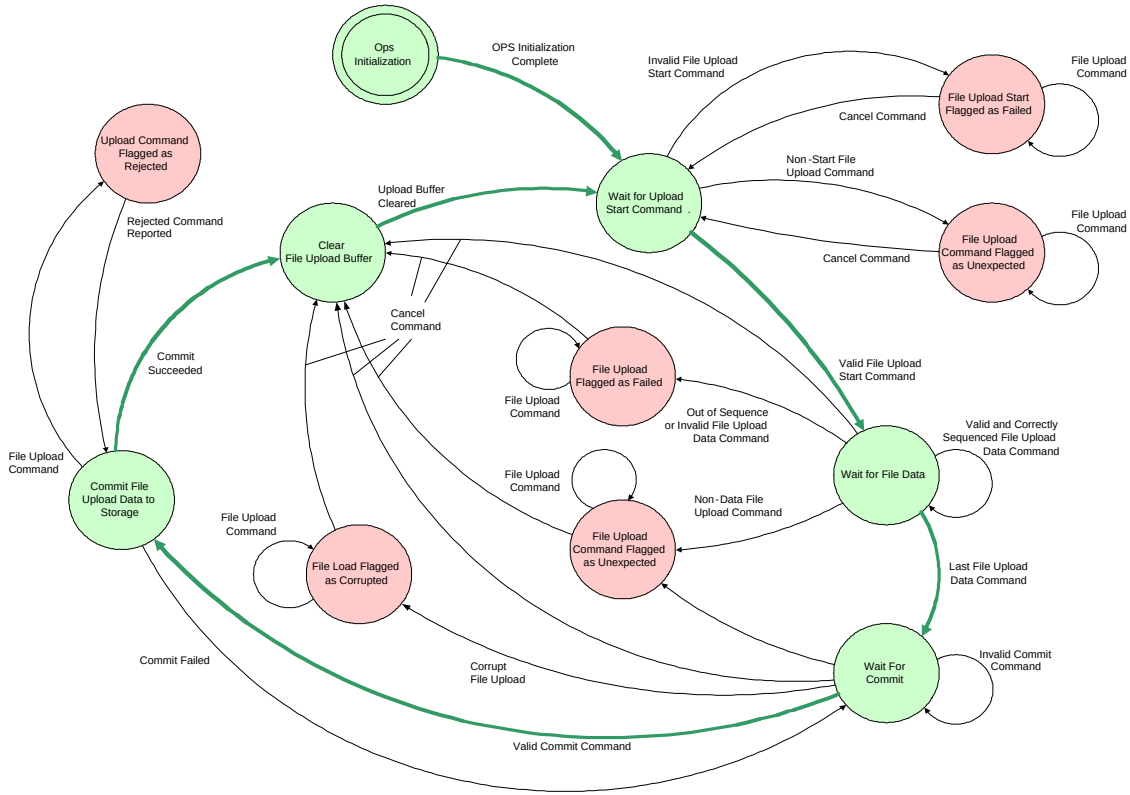
Boot File Operations

- PBC provides access to load the 6 boot file objects as shown here.
- Access to TFFS file system is not provided by PBC.
- No file dump capability is provided, memory dump operations are available to dump file data.

File Number	Description
0	Primary boot RTOS SDRAM Buffer
1	Second stage boot module 0 SDRAM buffer
2	Second stage boot module 1 SDRAM buffer
3	SIB EEPROM boot partition RTOS file
4	SIB EEPROM boot partition second stage boot module 0 file
5	SIB EEPROM boot partition second stage boot module 1 file

File Load State Machine

- **PBC uses the same state machine for file load as the application code.**



Memory Load and Dump

- PBC provides the capability to load data to memory mapped devices (RAM, EEPROM, PPCI registers and PCI memory), PCI device headers and PPC processor registers.
- PBC provides the capability to dump data from memory mapped devices, PCI device headers and PPC processor registers.
- No state machine for memory load, data is committed upon acceptance of memory load command.
- Memory dump data is split up and inserted into boot housekeeping telemetry packets.

Power PC Exception Processing

- PBC provides two sets of PPC exception vectors: ROM based and RAM based.
- ROM based exceptions are used until the PBC boot shell is initialized and configured. These exception vectors record the exception information in the boot diagnostics region and attempt to "plow through" the problem.
- RAM based exceptions are installed at boot shell initialization and used after boot shell is configured. These exception vectors record the exception information in the boot diagnostics region and reset the PBC code.
- Two sets of critical errors are implemented by the RAD750 as interrupts: PCI bus errors and memory controller errors. This requires PBC to enable these interrupts and process them as exceptions.

Boot SUROM Memory Map

- SUROM contains all EMC and PPC code & data.
- PPC data is copied to RAM at initialization.
- PBC personality data stored in SUROM.

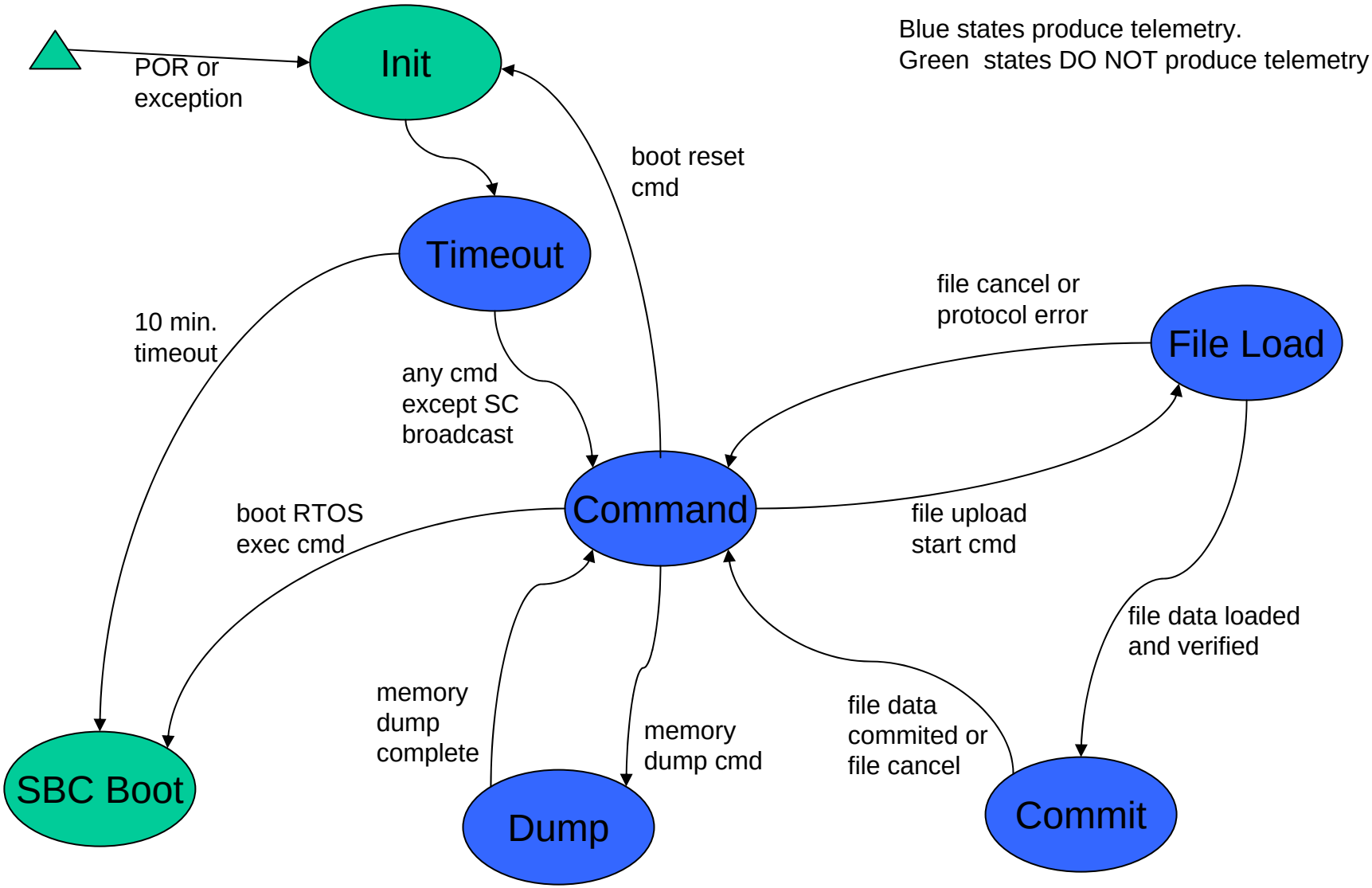
EMC Vector Table	FFF00000
EMC Reset Stub	FFF00020
PPC Reset Stub	FFF00100
PPC Boot Exception Vectors	FFF00200
EMC Startup and Vector Code	FFF01000
PPC Primary Boot Code Segment (*.text and .rodata*)	FFF02000
PPC Primary Boot Initialized Data Segment (*.data*)	*etext*
Unused	*FFF20000 (estimated)*
PBC Personality Data	*FFF3FC00*

Boot RAM Memory Map

- PBC limited to first 8 megabytes of memory.
- Provides storage for:
  - Ram exception vectors
  - Boot diagnostics region
  - File upload buffers
  - Dynamic allocation heap
  - PBC code program data (copied from SUROM at initialization)
  - Hardware I/O buffers
- Dynamic allocation needed by decompression routines. Allocation is "one way" only (i.e no recovery of resources).
- All other memory reserved by RTOS and application code.

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

Boot States


Blue states produce telemetry.
Green states DO NOT produce telemetry
POR or exception
Init
boot reset cmd
Timeout
10 min. timeout
any cmd except SC broadcast
file cancel or protocol error
File Load
Command
boot RTOS exec cmd
file upload start cmd
file data loaded and verified
memory dump complete
memory dump cmd
file data commited or file cancel
SBC Boot
Dump
Commit

Boot Commands (1 of 5)

- **APID 0x640 - Boot Operational Telecommands**
  - **Function Code 0 - Boot Start**
    - **Cancel timeout period; also acts as a NOOP command**
  - **Function Code 1 - Boot Reset**
    - **Provides warm restart of PBC code**
  - **Function Code 2 - Boot Error Dump**
    - **Dump PBC error codes to telemetry**
  - **Function Code 3 - Boot RTOS Execute**
    - **Start execution of Secondary Boot Code**

Boot Commands (2 of 5)

- **APID 0x641 - File Load Telecommands**
  - **Function Code 0 - File Upload Start**
    - **Announce new file upload**
  - **Function Code 1 - File Upload Cancel**
    - **Cancel active file upload**
  - **Function Code 2 - File Upload Commit**
    - **Commit file data to EEPROM**
  - **Function Code 3 - File Upload Data**
    - **Contains the file data**

Boot Commands (3 of 5)

- **APID 0x642 - Memory Load Telecommands**
  - **Function Code 0 - Memory Write**
    - **Load data to memory-mapped device (RAM, EEPROM, PPCI Registers).**
  - **Function Code 1 - PCI Device Header Write**
    - **Load data to PCI device header. Used for testing, diagnostics or fault resolution.**
  - **Function Code 2 - Processor Register Write**
    - **Load data to PPC processor register. Used for testing, diagnostics or fault resolution.**

Boot Commands (4 of 5)

- **APID 0x644 - Memory Dump Telecommands**
  - **Function Code 0 - Memory Data Dump**
    - **Dump data from memory-mapped device (RAM, EEPROM, PPCI Registers)**
  - **Function Code 1 - Memory Dump Cancel**
    - **Cancel active memory dump**
  - **Funciton Code 5 - PCI Device Header Dump**
    - **Dump contents of PCI device header**
  - **Function Code 6 - Processor Register Dump**
    - **Dump PPC processor registers.**

Boot Commands (5 of 5)

- APID 0x701 - SC Broadcast Telecommands
  - Function Code 1 - SIATTITUDE
    - Ignored by PBC
  - Function Code 2 - SIANCILLARY
    - Ignored by PBC
  - Function Code 3 - SITIMETONE
    - Used to set current time by PBC.

Boot Housekeeping Telemetry

- Boot sends out one telemetry packet 4 times per second.
- Contains command, file load and boot state information.
- Contains boot error codes.
- Contains memory dump data.
- Currently contains 28 bytes reserved for expansion. If not used, this area will be allocated to memory dump data.

Offset	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
0	Version=0			T=0	SH =1	APID = 0x200										
2	SF=3	Sequence Count														
4	Packet Length = 109															
6	Timestamp Seconds MSW															
8	Timestamp Seconds LSW															
10	Timestamp Sub-Seconds MSW = 0															
12	Timestamp Sub-Seconds LSW = 0															
14	Boot Software Mode															
16	Total Error Count															
	Queued Error Count															
18	Error Word MSW															
20	Error Word LSW															
22	Telecommand Packet Receive Count															
24	Boot Operational Telecommand Accept Count															
26	File Management Telecommand Accept Count															
28	Boot Operational Telecommand Sequence Count															
30	File Management Telecommand Sequence Count															
32	File Upload State															
34	File Upload Packet Count															
36	File Upload Error MSW															
38	File Upload Error LSW															
40																
42																
44																
46																
48																
50																
52																
54																
56																
58																
60																
62																
64																
66																
68																
70																
72																
74																
76	Memory Dump Word Count															
78	Memory Dump Address MSW															
80	Memory Dump Address LSW															
82	Memory Dump Word 0 MSW															
84	Memory Dump Word 0 LSW															
86	Memory Dump Word 1 MSW															
88	Memory Dump Word 1 LSW															
90	Memory Dump Word 2 MSW															
92	Memory Dump Word 2 LSW															
94	Memory Dump Word 3 MSW															
96	Memory Dump Word 3 LSW															
98	Memory Dump Word 4 MSW															
100	Memory Dump Word 4 LSW															
102	Memory Dump Word 5 MSW															
104	Memory Dump Word 5 LSW															
106	Memory Dump Word 6 MSW															
108	Memory Dump Word 6 LSW															
110	Memory Dump Word 7 MSW															
112	Memory Dump Word 7 LSW															
114	[HKP Packet Checksum]															

PBC Configuration Management

- PBC package contains the following functionality:
  - EMC Initialization / PPC Initialization / Memory Test
  - Exception Vectors
  - Boot Shell Command and Telemetry
- PBC package depends of the following packages/constituents
  - VXW (headers files only)
  - PBI (header files only)
  - PBS (pbs_boot)
  - ZLIB (zlib_inflate_boot)
  - CCSDS (ccsds_pkt_boot)
  - FILE (file_hdr_boot & file_upl_boot)
  - MEM (memboot)
  - CTDB (sumt_rt_poll_sib_boot)
  - LCB (lcbp)

Development Environment

- Using GNU tools as supplied by Wind River on a Sun workstation for PPC assembly and C development.
- Using EMC tools as supplied by BAE for EMC code development.
- Using Corelis in-circuit emulator for instruction level debugging on PPC processor (source-level debugging has been problematic)
- Using Corelis emulator and a combination of BAE supplied and NRL created tools to program the SUROM.
- Spacecraft Instrument Interface Simulator available for testing the 1553 interface.

Boot Code Test Approach

- **Intrusive testing for low-level features (e.g. memory test, processor initialization, exception processing) will be done with emulator.**
- **1553 interface testing will be done with the SIIS. All functions that can be stimulated and verified over the command and telemetry interface will be done in this manner.**
- **Testing on the LCB interface is TBD.**
- **The there will be a test case mapped to each detailed requirement that is specified in section 0.1 of the Primary Boot Code Document (LAT-TD-01806).**

Status

- PBC package (EMC Initialization, PPC Initialization, Memory Test, Boot Shell, 1553 Interface and Exception Vectors) is about 90% coded, ready for testing.
- pbs_boot, zlib_inflate_boot, ccsds_pkt_boot, file_hdr_boot, sumt_rt_poll_sib_boot constituents all have been integrated.
- Memory package (load and dump) is 75% coded.
- File load package is designed, 50% coded.

- We currently have functional boot code that initializes the RAD750, generates telemetry, accept commands and successfully executes secondary boot stored in SIB EEPROM.

Issues

- EPU test approach at NRL - how will a system level test be implemented and executed over the LCB interface?
- Memory test validation - how do we get enough confidence that memory test works? (we may have an option to try our PBC code on a broken RAD750 with bad memory from the SECCHI project)
- How soon can we execute and test the PBC code on a flight RAD750? We may find that the RAD750 evaluation board behaves differently than the RAD750 flight article
  - First flight article RAD750 expected end of May

Forward Work

- Implement file load package.
- Add EDAC status to memory test.
- Implement EEPROM commit functionality.
- Implement configuration based on PBC personality.
- Integrate PBC with LCB Driver.
- Write the PBC test plan.
- Execute the PBC test plan.
- Estimated completion date: March/April.

Secondary Boot Code (SBC)

Secondary Boot Overview

- **Primary boot handles initialization from power-on or reset to start of RTOS execution**
  - **Software contained in SUROM**
  - **No RTOS services available**
- **Secondary boot handles initialization from start of RTOS execution to application control of instrument**
  - **Software contained in EEPROM or delivered by file upload to primary boot**
  - **RTOS services available**
- **Secondary boot described in LAT-TD-02150**

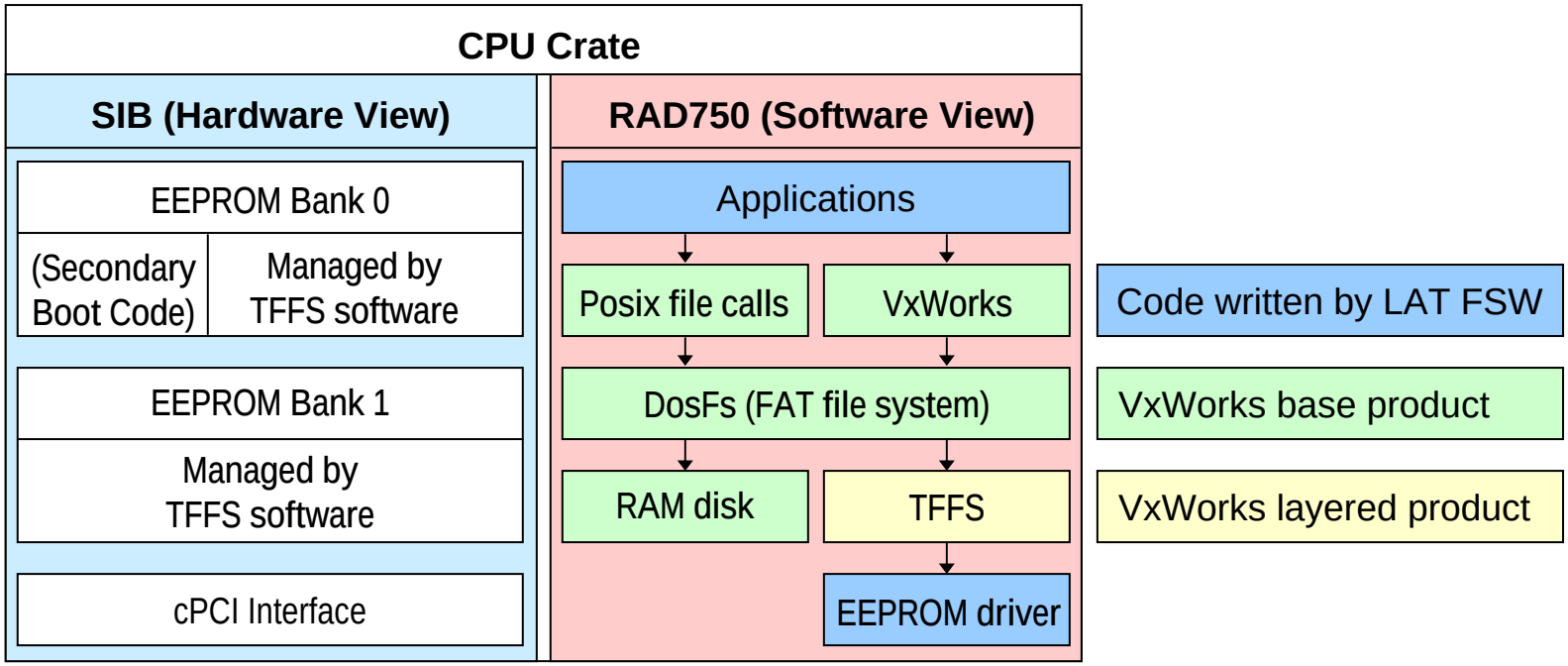
SBC Requirements

- Configure and initialize VxWorks RTOS
  - VxWorks 5.5 kernel
  - BSP and drivers for RAD750 CPU board hardware
- Provide file system non-volatile storage
  - Small code object modules (variable length)
  - Small configuration objects (variable length)
- Load application object modules
- Call application initialization functions
- Report diagnostics and error from secondary boot process
  - Fatal errors reboot and leave behind information in boot diagnostics area

SBC Software Organization

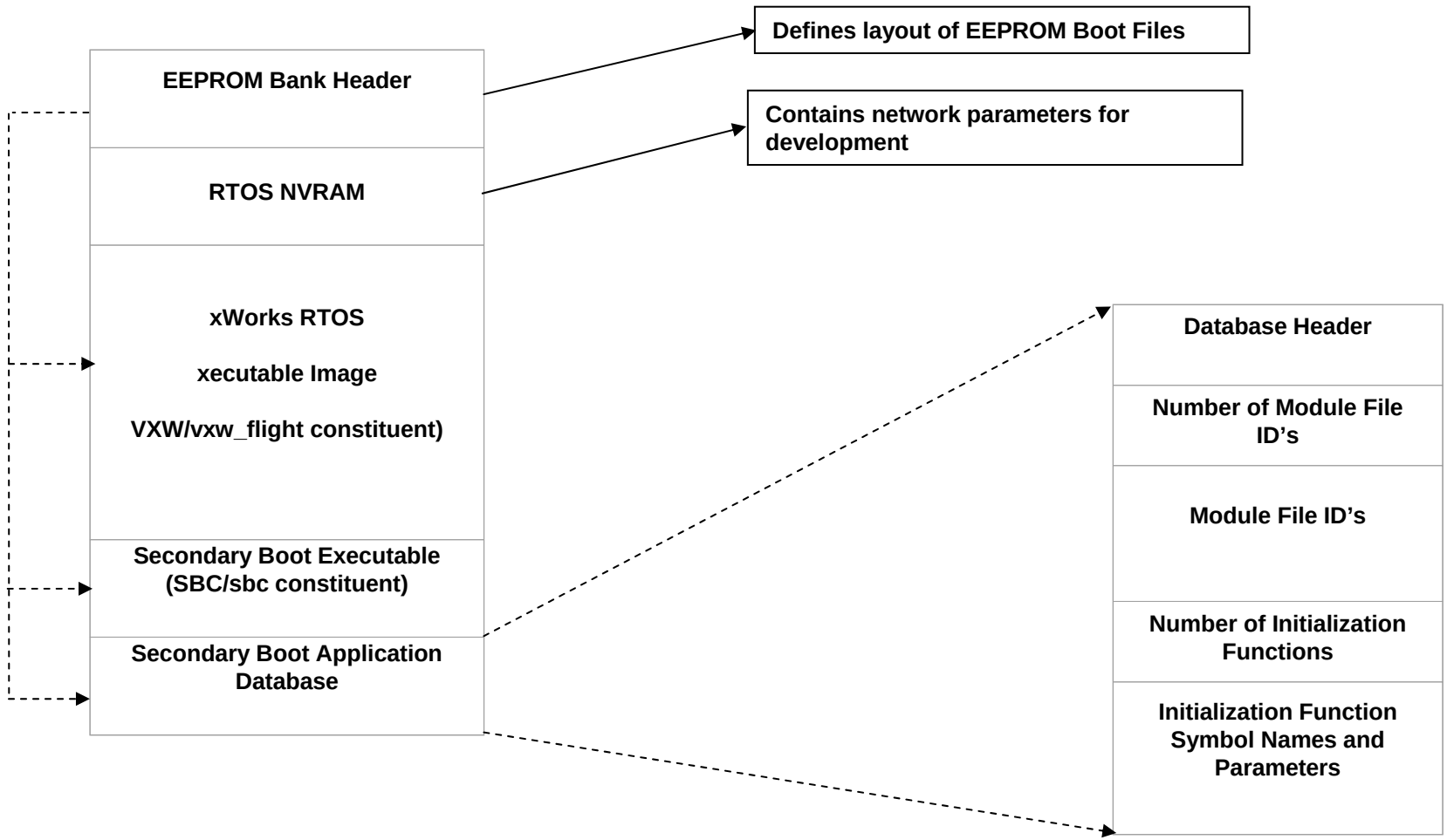
- Package VXW contains VxWorks BSP
  - Constituent: vxw_flight - provides kernel, BSP, and drivers for RAD750 board accessories
    - vxw_flight constituent also contains a small number of application modules statically linked with it
      - ZLIB/zlib_inflate, FILE/file_hdr
  - TFFS EEPROM driver is part of BSP image
  - Package VXW also contains many BSP variants
    - Development configurations run debugger, shell, networking, etc…
    - Flight configuration is subset of development configurations
    - Similar configurations available for COTS CPU boards
- Package SBC contains application loader
  - Constituent: sbc – provides application initialization

TFFS/EEPROM File System Architecture



- **TFFS provides wear-leveling of EEPROM locations**
  - **Multiple updates of the same file or file system meta-data is spread across all locations**
  - **Provides a map of the status of all EEPROM locations**
  - **Provides access to raw EEPROM driver functions (memory dump, memory poke)**
- **EEPROM Driver manages interaction with Austin EEPROM chips on SIB board**
  - **Write programming and completion status**
  - **Manages write lock and unlock of SIB EEPROM banks**

Boot EEPROM Organization



Application Initialization

- Create RAM disk file system partition
- Mount TFFS file system partitions
- Validate application database boot file
- Read list of application object module files
- Validate each object module file
- Inflate each object module file
- Invoke VxWorks dynamic loader for each object module file
- Read list of application initialization functions
- Call each application initialization function with parameters
- Examine status code returned by functions for fatal errors
- Report any errors in boot diagnostics area
- Reboot if error is fatal
- Otherwise, applications are now in control

File system / EEPROM Driver Testing

- File system / EEPROM driver testing
  - Two functional tests developed and in use
    - sib_eeprom_test – performs raw read and write operations on all EEPROM locations
      - Thorough test of hardware interface
      - Thorough test of SIB EEPROM driver functions
      - Does not require file system formatting
    - tffs_func_test – performs common file system operations as employed by application software
      - Create, read, write, copy, and delete file objects
      - Create and delete directory objects
      - Obtain file and directory status
      - Requires DOS/TFFS formatting
      - Short and long form of test

SBC Status (1)

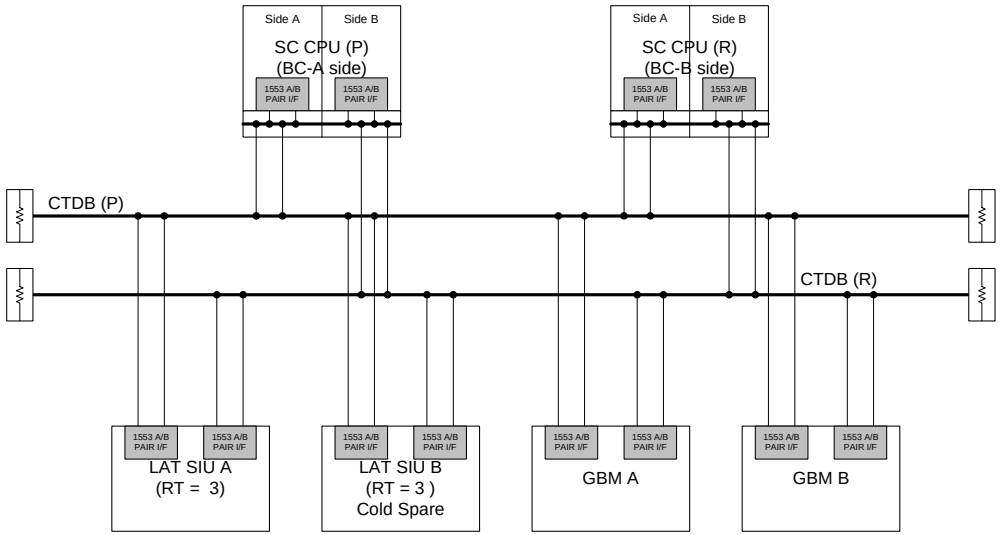
- VxWorks BSP for RAD750 is available and in use
  - Final flight configuration may still need small adjustments
  - Latest workarounds for BAE errata incorporated
- TFFS / EEPROM running and working at functional level
  - Genuine file system with all expected POSIX functionality
  - Issue: write performance is painfully slow
  - Issue: advertised fault tolerant features do not appear to work
    - Bad blocks detected by driver are not automatically marked
    - May require more software interfaces to handle manual marking of bad blocks
    - Media format is documented and understood

SBC Status (2)

- **SBC Application Loader is developed and functional**
  - **Currently, SBC package provides simple script parser to produce binary database file**
    - **Move to XML script to maintain commonality with other LAT configuration formats**
  - **Demonstration of the application initialization process has been done**
  - **Work needs to be done to provide a more formal test method of the SBC software**

1553 Communications

LAT 1553 Interface



- **MIL_STD_1553B (1553) bus is primary interface for exchanging information between LAT, SC, and GBM**
  - **Telecommands from SC and GBM**
  - **Telecommands to SC and GBM**
  - **Telemetry to SC (housekeeping, diagnostic, alert)**
- **SC will act as bus controller (BC) node**
- **Each SIU can act as remote terminal (RT) node**
- **Bus protocol and schedule under control of SC**
  - **GLAST 1553 Bus Protocol Interface Control Document**
- **All traffic will consist of CCSDS packets (CP_PDU).**

1553 RT Driver Requirements (1)

- **Requirements derived from GLAST 1553 ICD**
  - **Support data wraparound**
  - **Support Notice II mode codes**
  - **Receive up to 20 CCSDS telecommands / second and deliver to consumer applications**
  - **Collect and send up to 5 CCSDS telecommands / second from producer applications (application driver only)**
  - **Collect and send CCSDS telemetry packets from producer applications**
    - **Insert telemetry packets into telemetry message blocks (4 / second)**
    - **Allow for one real-time housekeeping telemetry packet for each telemetry block**

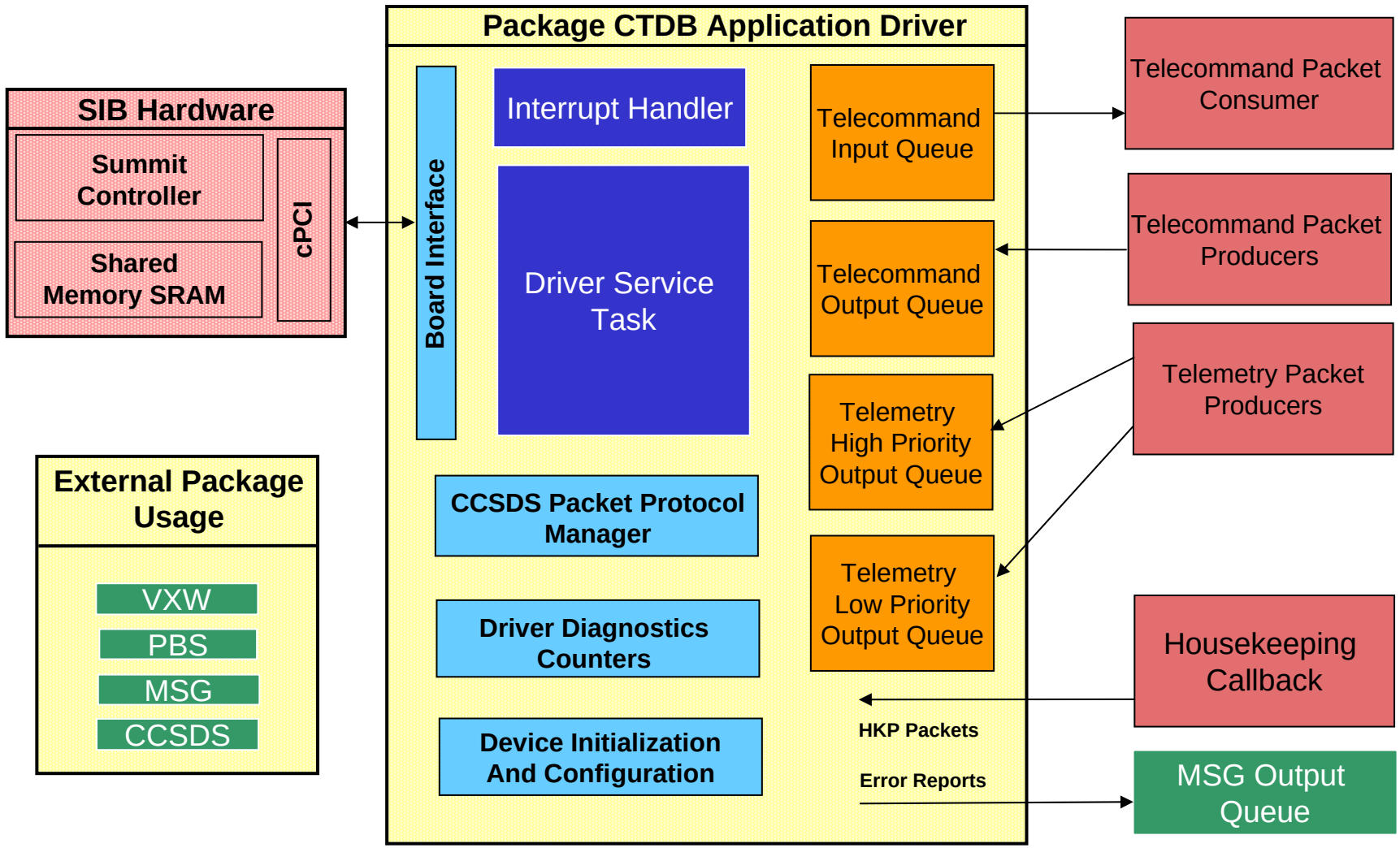
1553 RT Driver Requirements (2)

- **Internal requirements from LAT FSW**
  - **De-couple bus schedule activity from application processing**
    - **One exception is real-time housekeeping packet, which should be delivered synchronous to the bus schedule**
    - **Guarantee that the HKP packet is always delivered at head of telemetry blocks**
  - **Log 1553 bus errors as detected by the Summit controller**
  - **Keep communications diagnostics counters (application only)**
  - **Perform timeout on telemetry and telecommand send polling by BC (application only)**
  - **Provide priority levels for outgoing telemetry packets (application only)**
- **See CTDB 1553 Drivers document**

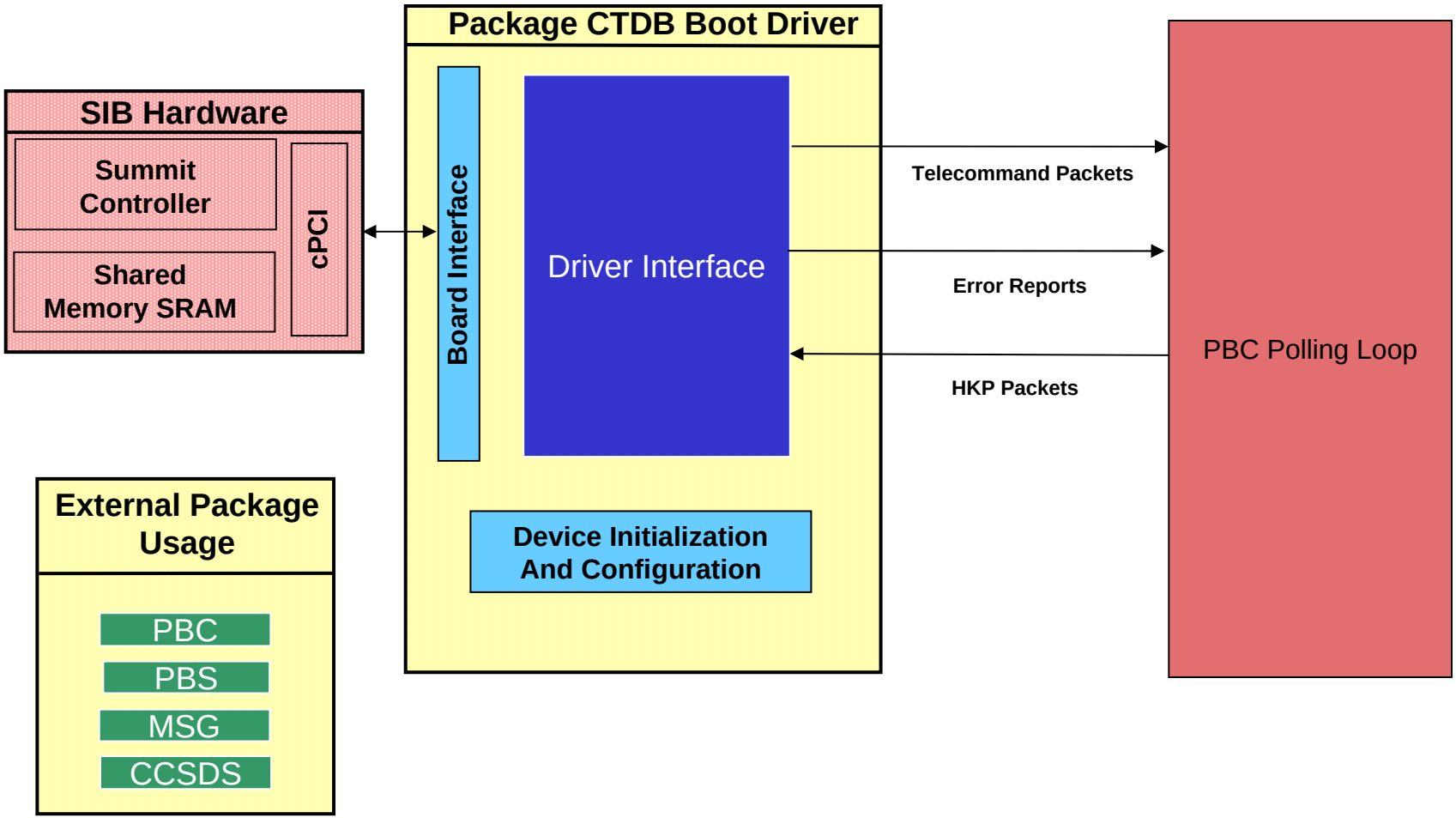
1553 Software Organization

- 1553 drivers and related software contained in package CTDB
  - API for client software is the same among variants
- RT Drivers for SIB board
  - Constituent: sumt_rt_sib – application mode driver
  - Constituent: sumt_rt_poll_sib – boot mode driver
- RT Drivers for COTS Alphi PMC 1553 board
- BC Drivers for testing and development
- IP/Ethernet Simulator for testing and development
- Unit Test for hardware-independent portions of driver available
- Functional tests for drivers using SIIS/SDIS available
- Documentation for design overview and API available
- 1553 RT software runs on SIU crate only

1553 RT Driver (Application) Architecture



1553 RT Driver (Boot) Architecture



1553 Driver Inputs/Outputs

- Inputs
  - Driver configuration, input at initialization
    - 1553 addressing
    - Queue sizes
  - Outgoing telecommand packets to SC or GBM
  - Outgoing telemetry packets to SC
- Outputs
  - Incoming telecommand packets from SC or GBM
  - Error reports for API calls (out of range parameter, etc …)
  - Error reports for 1553 bus illegal activity
  - Diagnostics counters (application driver only)
    - Packet counts
    - Byte counts
    - Error counts

1553 RT Driver Testing (1)

- CTDB package unit test covers 'CCSDS Packet Protocol Manager' portion of driver
  - Simulate raw 1553 input messages and extract CCSDS telecommand packets
  - Simulate CCSDS telemetry output packets and construct raw 1553 output messages
- Two functional tests for RT driver
  - Functional tests have two components
    - LTX script which loads target software and records target logs
    - SIIS/SDIS Perl script which performs closed-loop testing
  - ctdb_link_test – Tests that driver has properly initialized the device and complies with standard
    - Uses special 1553 link level telecommands and telemetry provided by SIIS and SC FSW

1553 RT Driver Testing (2)

- Verifies data wraparound with multiple patterns
- Verifies all unused subaddresses are illegal
- Versions for both application and boot drivers

– ctdb_com_test – Tests that the driver can perform basic CCSDS packet communications

- Uses LAT commuications diagnostic telecommands and telemetry
- SIIS Perl script sends a telecommand to LAT RT and waits for the expected telemetry reply packet from LAT RT
- Versions for both application and boot drivers
- Issue: Testing of telecommand send functionality
- Issue: Testing of “dump style” telemetry

- Interactive test applications for development (both RT and BC)

1553 Software Status

- Application and boot RT drivers and API designed, documented, and reviewed
- Boot driver integrated into primary boot code (service for package PBC)
- Repeatable tests developed
  - Not quite full coverage of functionality
- Recent work involves integrating and testing driver with EM model SIB board
- Application driver still needs to begin integration