GLAST Large Area Telescope

Instrument Flight Software
Flight Unit Peer Review
16 September 2004

Software Architecture

J. J. Russell
Stanford Linear Accelerator Center

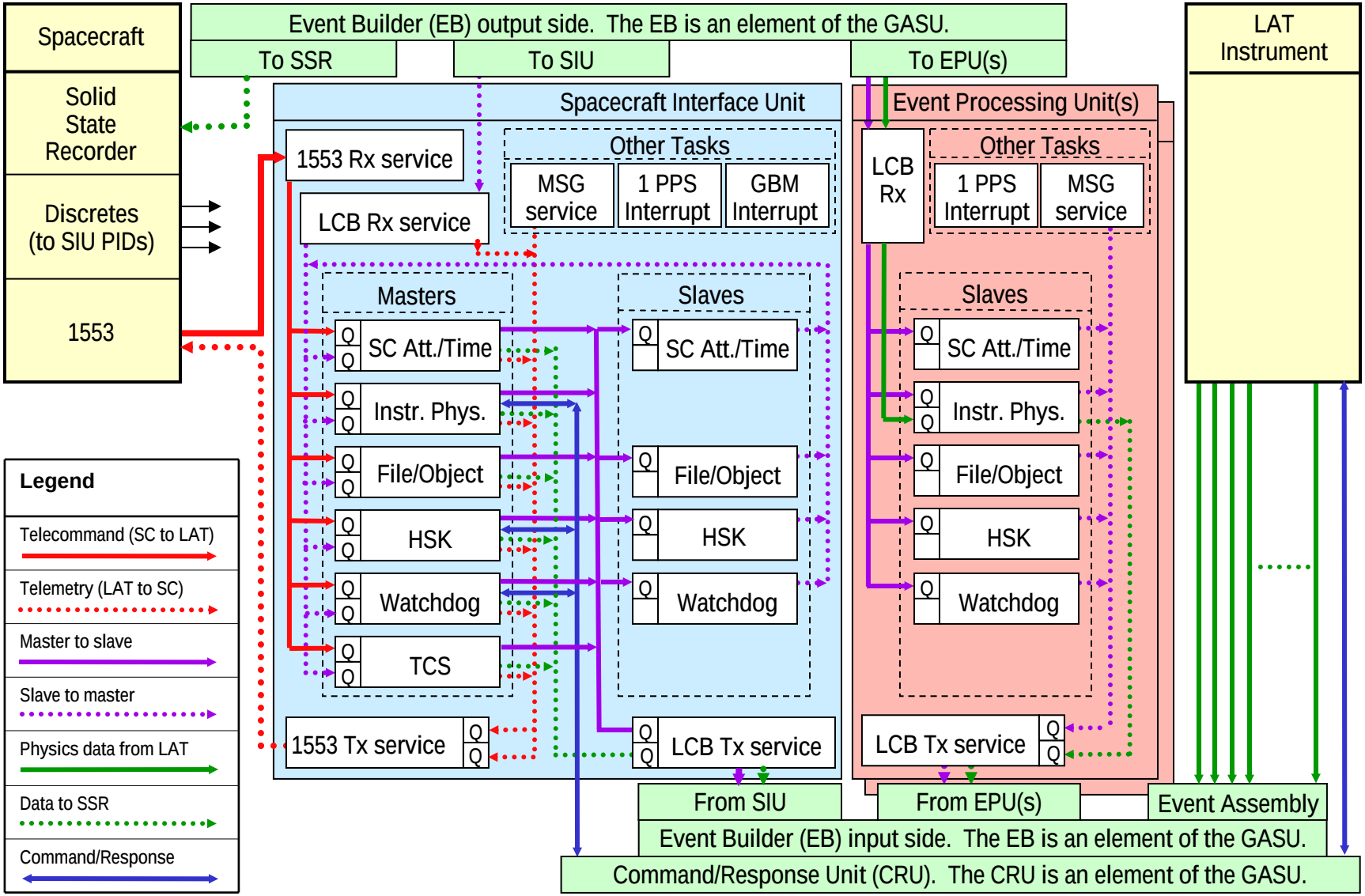
russell@slac.stanford.edu
(650) 926-2583

Overview

- Review final system software architecture
- Identify elements of the final architecture needed for the Flight Unit
  - Identify Flight Unit hardware architectures
  - Map functions to the final system

- Finding more information
  - The LAT flight software web pages are a rich source of information about the current state of the FSW
  - Start at the flight software home page and find links to
    - Flight software specific LATDocs
    - Flight software package documentation of Application Programming Interface (API) (look under “Doxygen”)
    - Introductory tutorials and navigation aids
    - Telecommand and telemetry packet definitions
    - …

LAT FSW Task Architecture



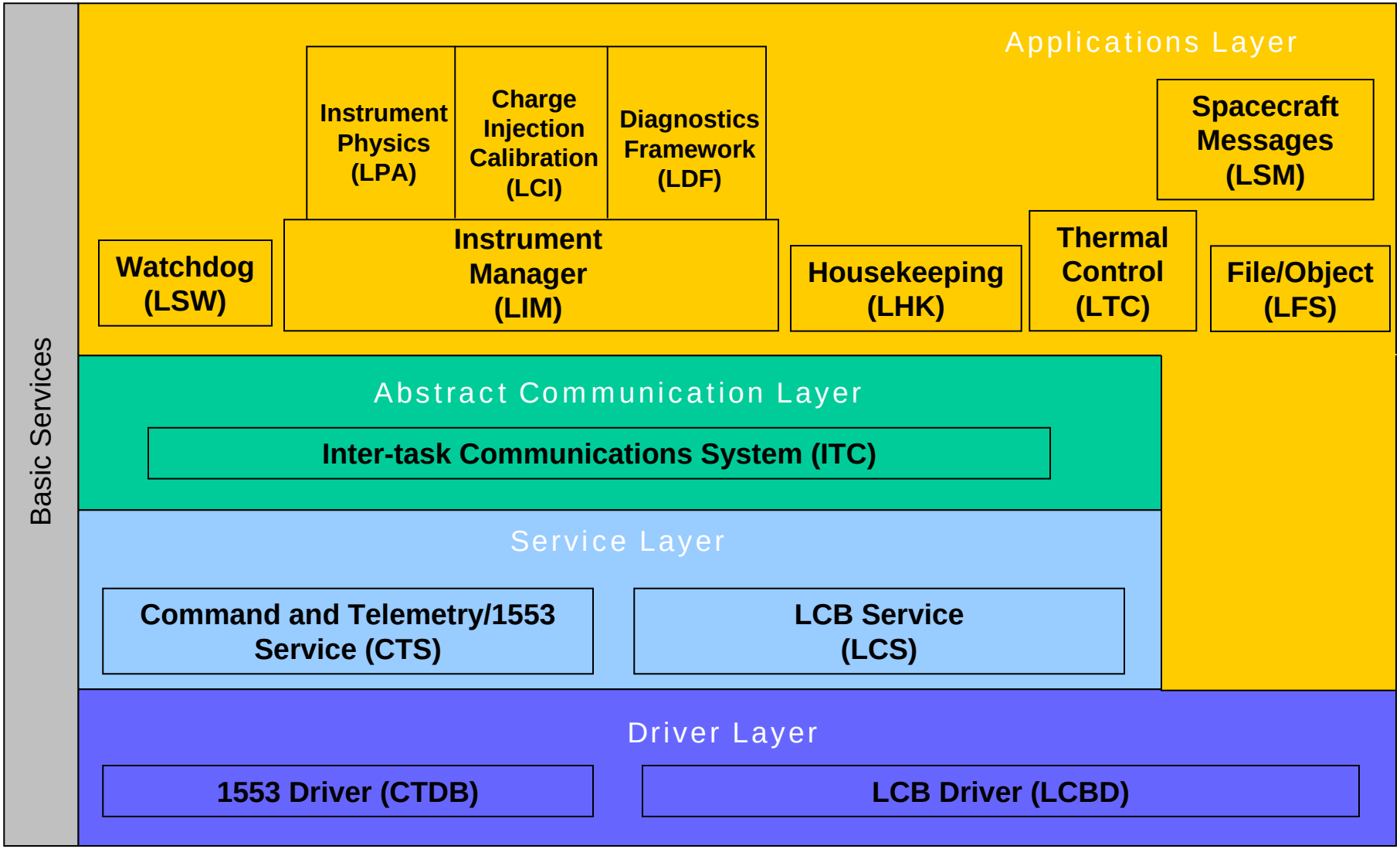
Overview of Tasks

- The flight software runs as a set of VxWorks "tasks"
  - Tasks are analogous to Unix processes, but they share a common address space.
  - In practice, however, FSW tasks communicate with each other via (possibly inter-machine) message queues.

- Flight software tasks can be grouped into three categories: Drivers, Services, and Application Tasks
  - Drivers control operation of interface/communications hardware
  - Services, such as the Command and Telemetry/1553 Receive Service, are used by many other tasks throughout the system
    - Services act as interfaces to the driver/hardware layer
    - Services are not concerned with the types of data they process; instead, they are defined by the source or destination of the data they process
    - An abstract, inter-task communications “service” also falls into this category
      - ITC is not a task so much as a communications “toolbox”
  - Application tasks, such as the File/Object task, perform specialized functions.
    - Application tasks are defined by the type of data they process

- Some flight software functions are divided up between "master" and "slave" tasks.
  - The master tasks (located on the SIU, because of its privileged position) receive, perform higher level verification, and queue commands
  - The slave tasks actually do the work

FSW Layer Architecture



The Driver Layer

- The drivers are accessed by tasks on all layers
  - In rare cases, application tasks access the drivers directly
  - However, most of the interaction with drivers occurs through the Abstract Communication and Service layers

- 1553 Driver
  - Supports exchange of commands and telemetry with the Spacecraft on the MIL-1553 bus
    - Spacecraft acts as bus controller
    - SIU acts as a remote terminal
  - Commands and telemetry messages traveling over this bus are wrapped up in CCSDS packets
  - A proven system
  - A presentation on the 1553 driver is not slotted on today's agenda, but presentation materials are provided on the Peer Review Web page.

- LCB Driver
  - Driver for the LAT Communications Board, a cPCI module installed in every CPU in the LAT
  - Supports communication of event data, configuration commands, and CPU-to-CPU messages within the LAT
  - The LCB driver will be discussed later today by Ed Bacho.

The Abstract Communication Layer

- **Together, the Abstract Communications Layer and the Service Layer (next slide) provide applications with a "hardware-blind" means of communicating with:**
  - **Other applications**
    - **On the same CPU or on other CPUs**
  - **With the Spacecraft**
  - **With the Instrument**
- **The Inter-task Communications system:**
  - **Provides an abstract communications layer**
  - **ITC defines a common communications standard to unify communications between tasks on either the same or different CPUs.**
  - **It also provides the services to build up tasks that are capable of communicating according to the standard.**
  - **ITC is not itself a task, it's a toolbox.**

The Service Layer

- The Command and Telemetry/1553 Service (1553 Tx and Rx)
  - It provides the service layer between applications (high level tasks) and the 1553 hardware driver
  - Built using the ITC toolbox
  - It provides communication:
    - Spacecraft to CPU
    - CPU to spacecraft
- The LCB Service (LCB Tx and Rx)
  - It provides the service layer between applications (high level tasks) and the LCB hardware driver.
  - Built using the ITC toolbox
  - It provides communication:
    - CPU to CPU
    - CPU to SSR
- The Service Layer and the Abstract Communications Layer will be covered later today by Sergio Maldonado

The Applications Layer

- Applications perform the work required to operate and monitor the instrument, and collect science data
- Housekeeping
  - Housekeeping interrogates the instrument housekeeping systems (currents, voltages, counters, …) and gathers CPU metrics, then places this data into housekeeping 1553 stream
  - Presented today by Sergio Maldonado
- Instrument Manager
  - Responsible for instrument mode control (e.g., Safe Mode)
    - Therefore, controls power up and power down sequence as well
      - Will be presented by James Swain
  - Controls access to and control over a related set of applications:
    - Diagnostics Framework
      - Provides an infrastructure for defining and running instrument diagnostics
      - Presented by James Swain
    - Instrument Physics
      - Collect and filter data, detect GRBs, and monitor instrument deadtime
      - Presented by JJ Russell
    - Charge Injection Calibration
      - Calibrates the instrument
      - Presented by James Swain

The Applications Layer (cont'd)

- **Spacecraft Messages and Attitude/Time**
  - **These apps receive and process attitude, time tone, and ancillary messages from the Spacecraft. This data allows all data from the LAT to be time stamped.**
    - **Attitude data to support Gamma-ray burst tracking and notification**
  - **Handle responses to signals on discrete lines**
  - **Will be presented today by Steve Mazzoni**
- **File/Object and Boot Code**
  - **Primary and secondary boot process**
  - **File/Object receives requests for operations on the RAM and EEPROM based file systems (file upload, dump, copy, delete, …), interrogates the file system (directory dumps, file checksum checks, …)**
  - **Discussed later by Dan Wood**
- **Watchdog**
  - **Provides a software watchdog capability**
  - **Not covered in today's session, but presentation materials are available on the Peer Review Web page.**
- **Thermal Control**
  - **Read instrument temperature sensors and, based on a set of configurable parameters, apply a Lockheed algorithm to determine whether to activate heaters.**
  - **Presented later today by Steve Mazzoni**

FSW Consumers

Consumers: I&T Test Stands

- Field test stands for ACD, CAL, TKR & DAQ
  - Functionality to test subsystems
    - Configuration
    - Event read-out
    - Housekeeping stream
  - ACD has special hardware needs, being dependent on
    - All elements of the GASU
      - CRU, AEM, GEM, EBM
    - Through CRU/AEM, access to
      - ACD front end electronics (GAFE, GARC)
- Field maintenance is the responsibility of LAT I&T
  - FSW responsible for providing embedded system software only

Consumers: I&T Tower and GASU Integration

- All the needs of the Test Stands
  - This is where they will be checking out new software to be delivered to the field
- Plus the ability to “Build up the instrument”
  - In practice, this means
    - Supporting Multi-towers
    - A clean interface to the GASU triggering system

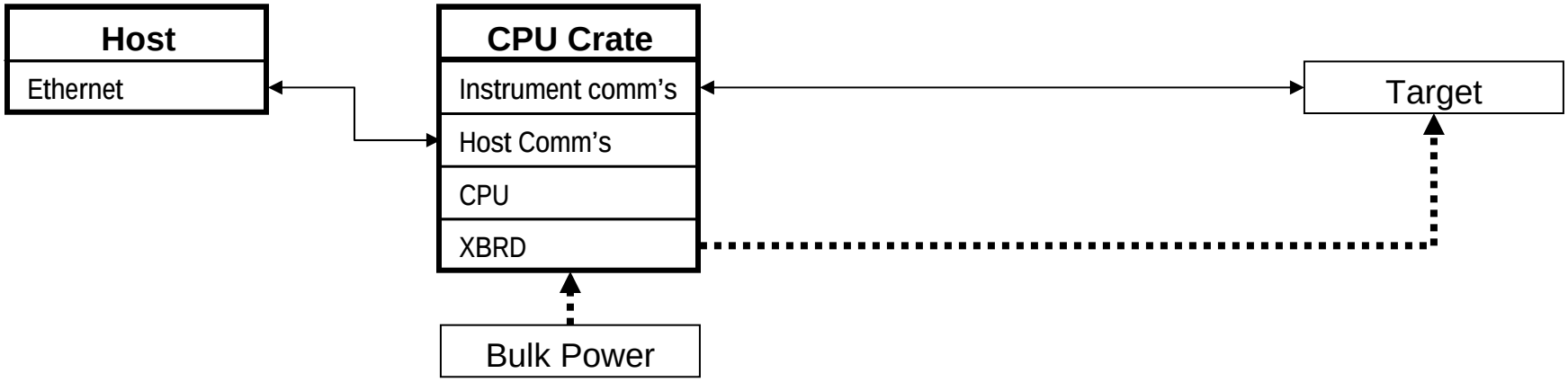
Consumers: ISIS

- SIU Boot Sequence
- PID Interface Testing
  - Both inputs and outputs
- 1553 Interface Testing
  - Exchange of commands and telemetry
  - Keep track of statistics, i.e. count messages
  - Housekeeping stream
- Science Interfaces
  - Soliciting of known test patterns

Consumers: Flight Software Test-Bed

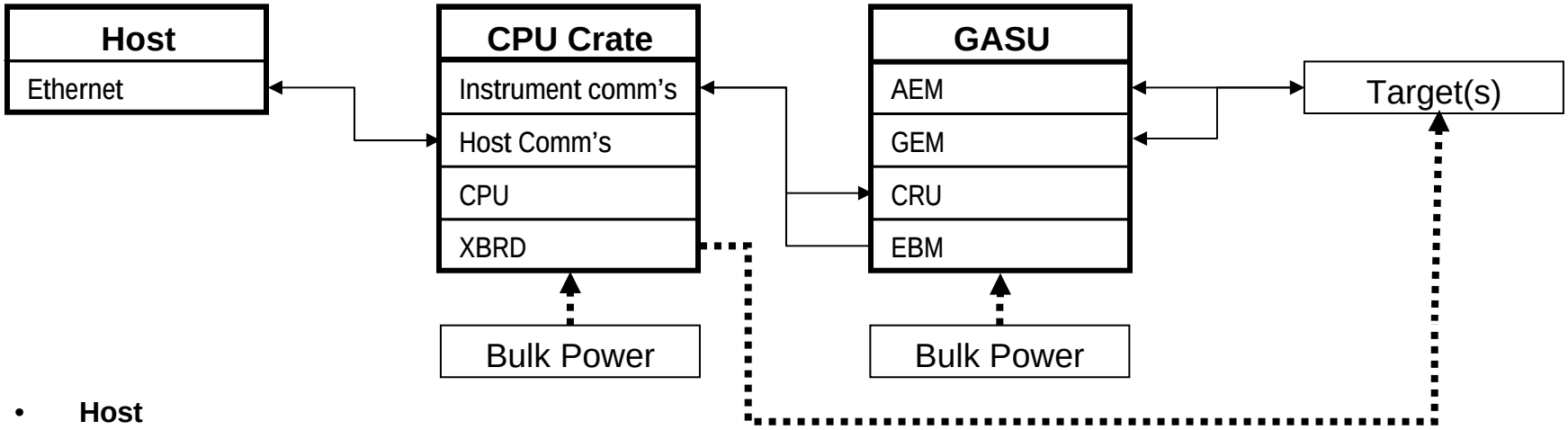
- **Instrument Flight Software (ourselves)**
  - **Development of instrument flight software final deliverable**
  - **Development done on the test-bed**
    - **A full flight-like DAQ system (except sensors)**
    - **Sensors emulated by Front End Simulators (FES)**

TKR or CAL Test Stand Hardware Configuration



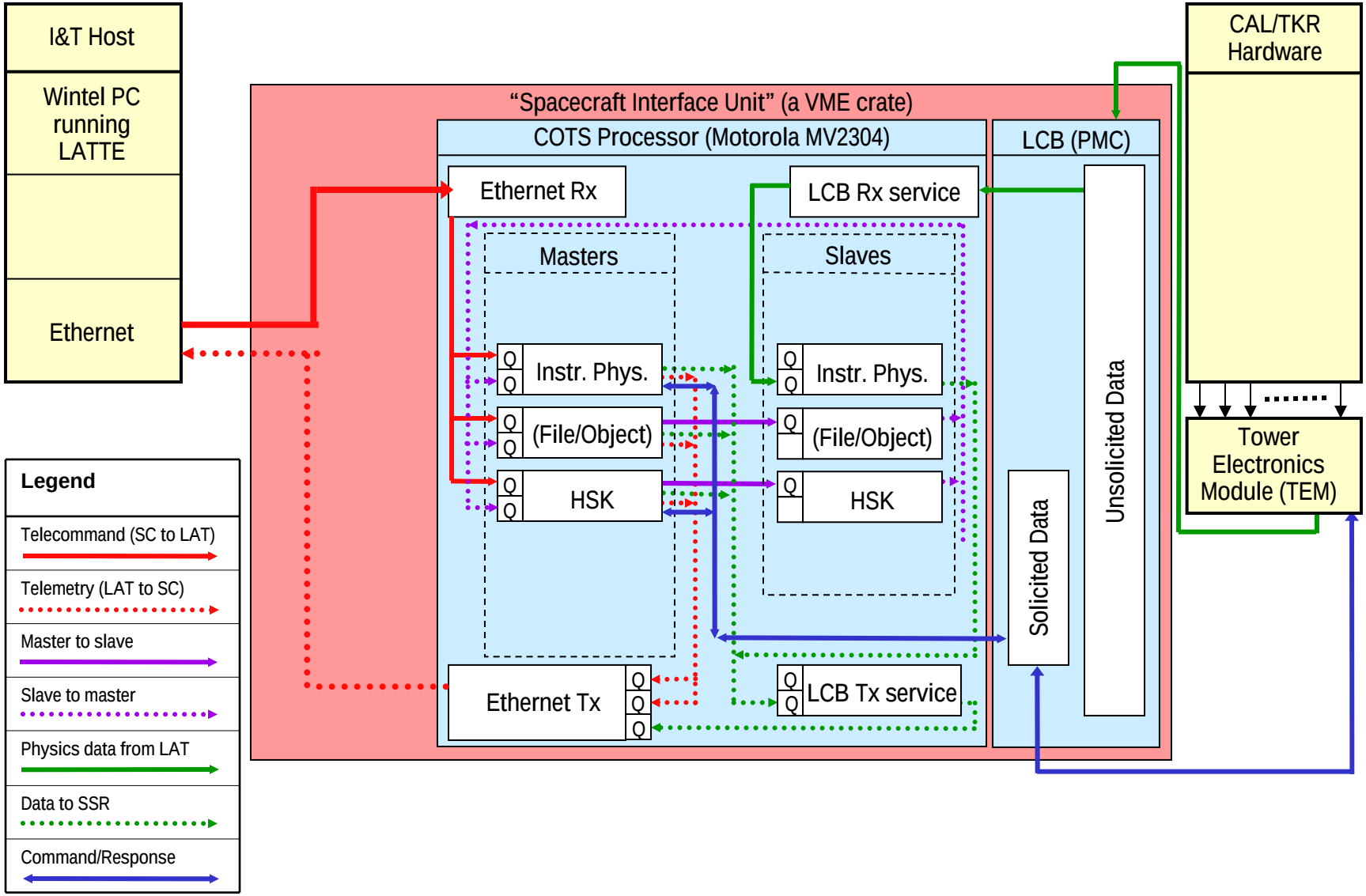
- **Host**
  - **Wintel PC running I&T LATTE software**
  - **Ethernet communications (both command/response and event acquisition)**
- **CPU Crate**
  - **VME chassis**
  - **Instrument communications: LAT communications board (LCB) (PMC form factor)**
  - **Host communications: Ethernet (built in to COTS CPU)**
  - **CPU: Motorola MV2304 COTS processor**
  - **XBRD: Minimal emulations of the trigger function and the power distribution function**
- **GASU**
  - **None**
- **Power Distribution**
  - **Transition board (XBRD)**
- **Target(s)**
  - **Sensors under test, read by a Tower Electronics Module (TEM)**

ACD Test Stand Hardware Configuration

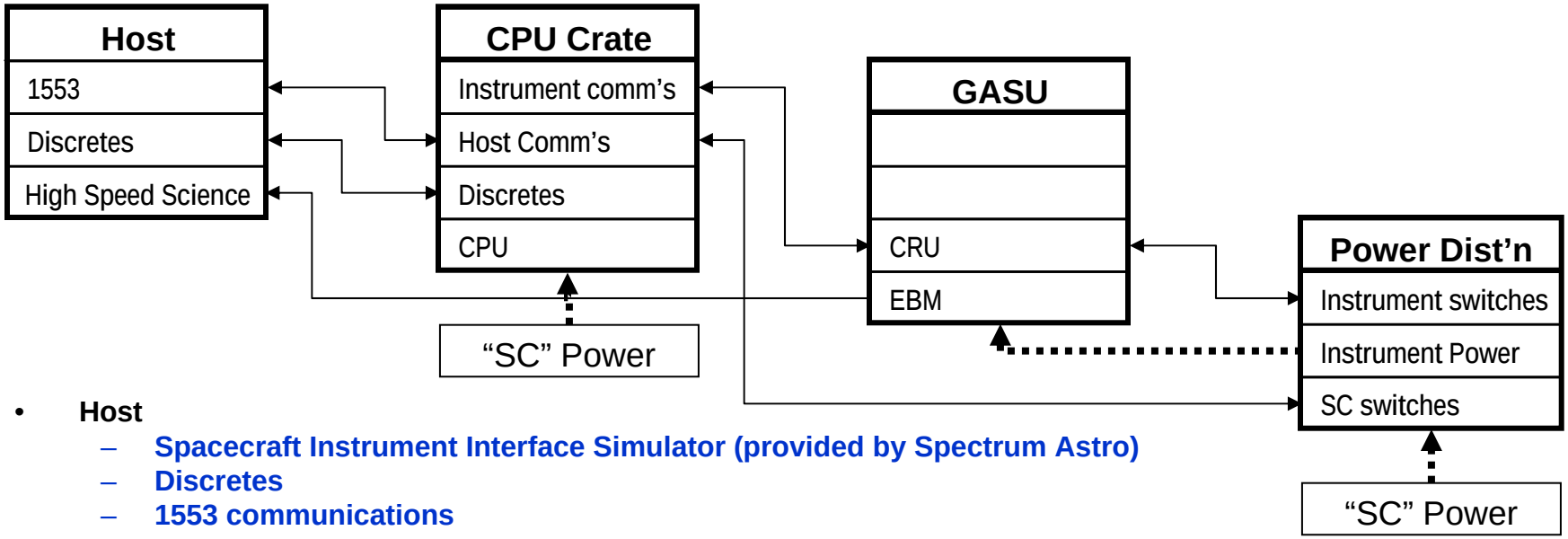


- **Host**
  - **Wintel PC running I&T LATTE software**
  - **Ethernet communications (both command/response and event acquisition)**
- **CPU Crate**
  - **VME chassis**
  - **Instrument communications: LAT communications board (LCB) (PMC form factor)**
  - **Host communications: Ethernet (built in to COTS CPU)**
  - **CPU: Motorola MV2304 COTS processor**
  - **XBRD: Emulation of the power distribution function**
- **GASU**
  - **Full GASU implementation**
    - **AEM, GEM, CRU, EBM**
    - **Primary and redundant sides**
- **Power Distribution**
  - **Transition board (XBRD) to target hardware**
  - **Simple bulk 28V supply for GASU**
- **Target(s)**
  - **Sensors under test, triggered by GEM and read back by FREE board(s) via the GASU AEM unit**

TKR, CAL, ACD Test Stand Software Configuration

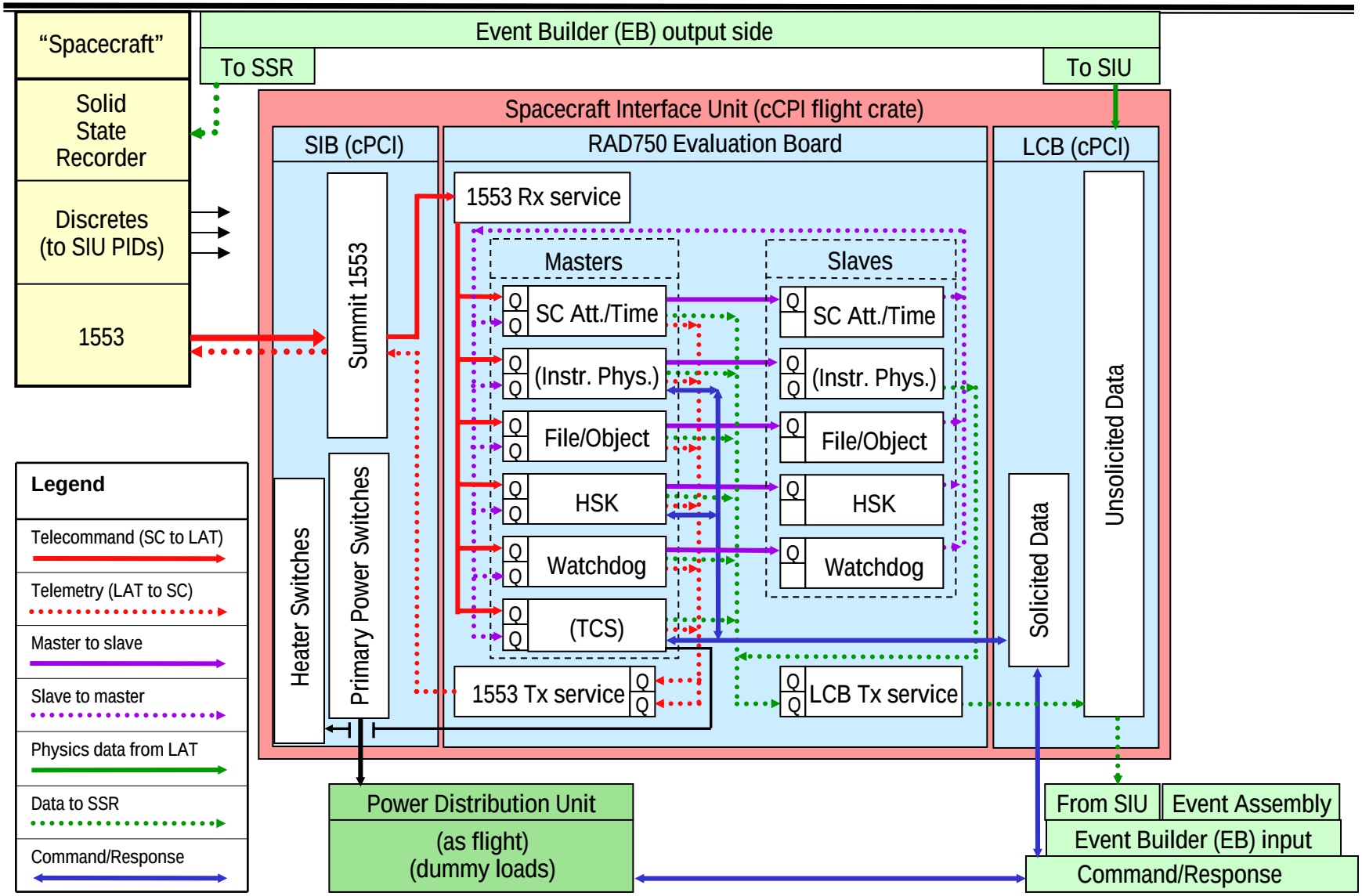


ISIS Hardware Configuration

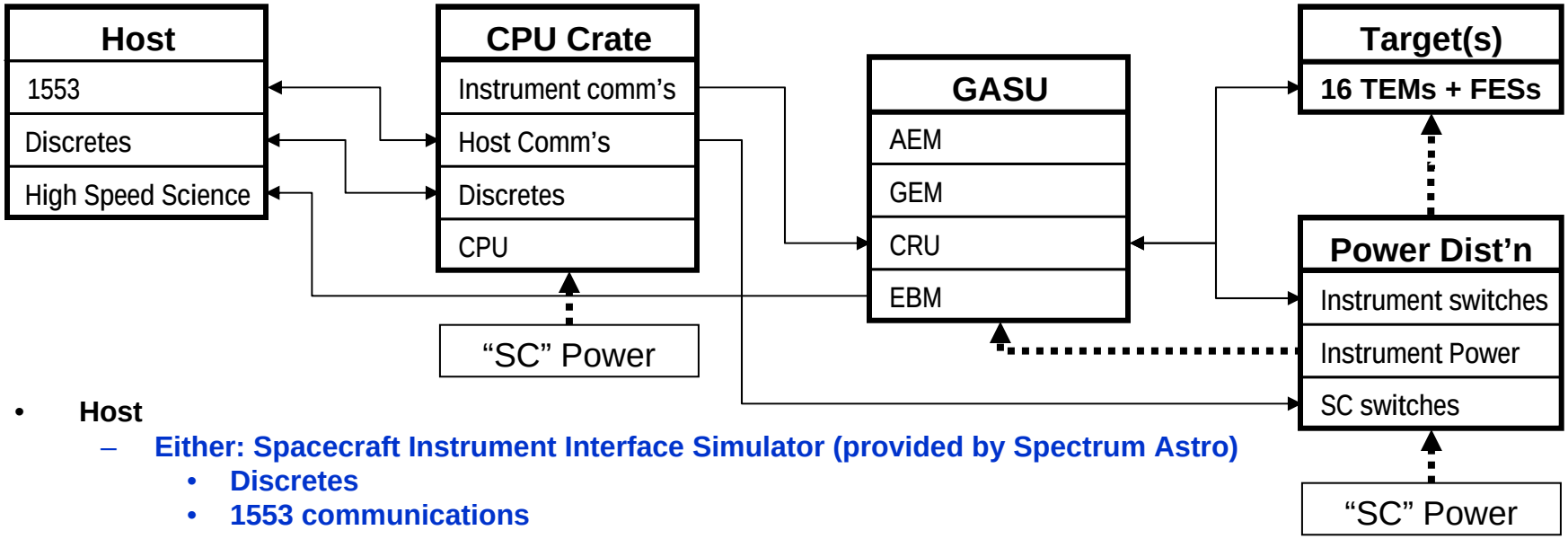


- **Host**
  - **Spacecraft Instrument Interface Simulator (provided by Spectrum Astro)**
  - **Discretes**
  - **1553 communications**
  - **High speed science interface**
- **CPU Crate**
  - **cPCI flight equivalent chassis**
  - **Instrument communications: LAT communications board (LCB) (cPCI form factor)**
  - **Host communications: Storage and Interface Board (SIB) (also does power bootstrap)**
  - **CPU: RAD750 processor**
- **GASU (GASU-lite) (primary side only)**
  - **Command/response unit (CRU)**
  - **Event Builder Module (EBM)**
- **Power Distribution**
  - **Full PDU implementation (primary and redundant) but with dummy loads for most instrument targets**
- **Target(s)**
  - **None**

ISIS Software Configuration

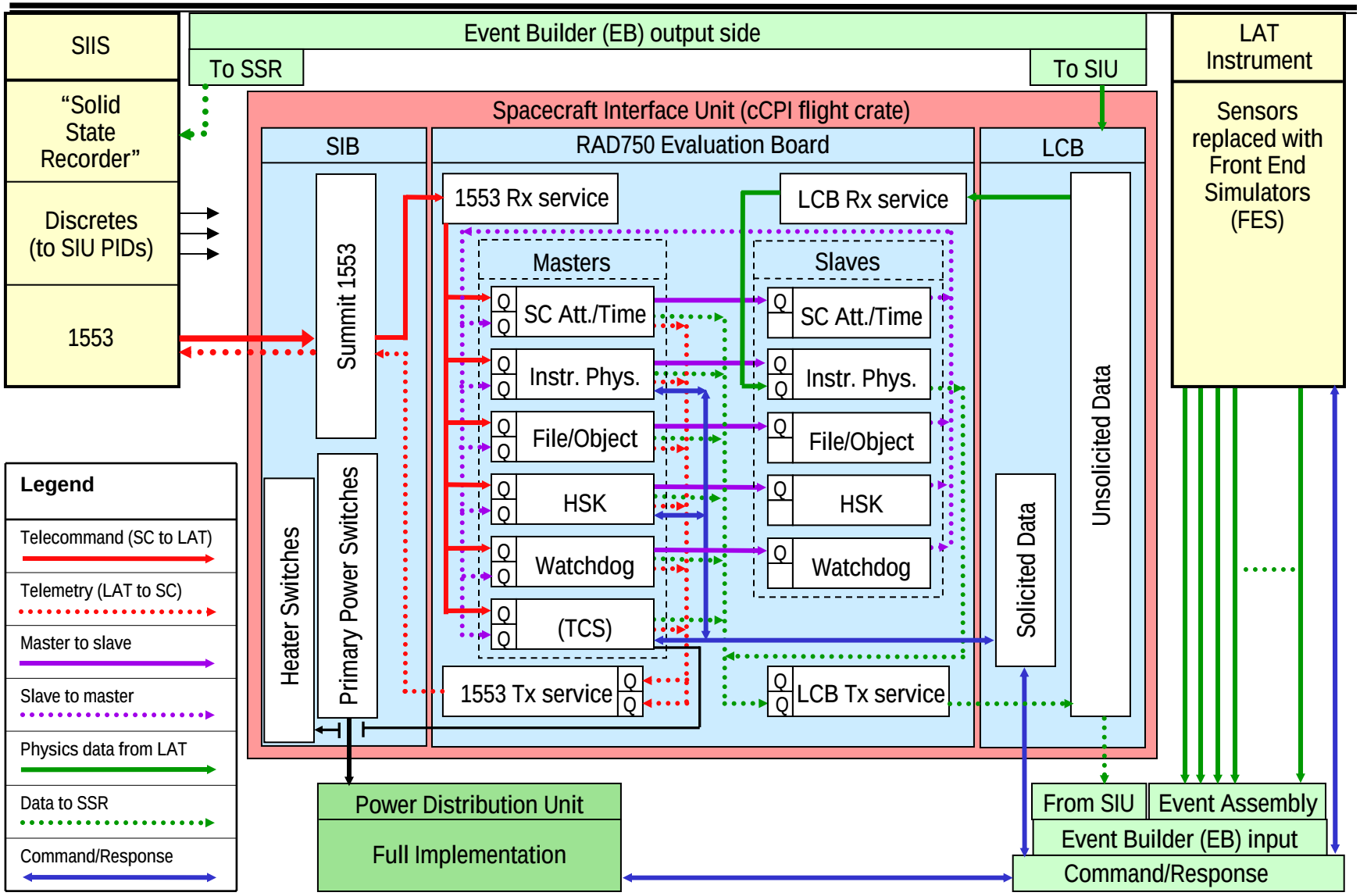


Test-Bed Hardware Configuration



- **Host**
  - **Either: Spacecraft Instrument Interface Simulator (provided by Spectrum Astro)**
    - **Discretes**
    - **1553 communications**
    - **High speed science interface**
  - **Or: Ethernet simulation**
- **CPU Crate**
  - **cPCI flight equivalent chassis or commercial cPCI chassis**
  - **Instrument communications: LAT communications board (LCB) (cPCI form factor)**
  - **Host communications: Storage and Interface Board (SIB) (also does power bootstrap)**
  - **CPU: RAD750 processor or Motorola MCP750 COTS processor**
- **GASU**
  - **Full GASU implementation (including primary and redundant sides)**
- **Power Distribution**
  - **Full PDU implementation (including both primary and redundant sides)**
- **Target(s)**
  - **16 Tower Electronics Modules (backed by Front End Simulators)**

Test-Bed Software Configuration



Functions to Support Current Consumers

- Functions evident in previous diagrams
  - Configure TKR and CAL front end electronics
  - Configure ACD front end electronics
  - Configure GASU (CRU, GEM, EBM, AEM)
  - Configure PDU
  - Configure XBRD
  - Configuration by compressed file
  - Real event delivery to CPU
  - Housekeeping data stream
  - RAD750 boot and crate initialization
  - 1553 bus communications
  - Command/Telemetry database and services
  - Emulated event delivery across science data interface (ISIS)
- Derived functions internal to instrument flight software
  - CPU internal communications/task frameworks
  - Software watchdog

Additional Functions to Deliver with Flight Unit

- **Going forward:**
  - **Mode Control, Diagnostics, and Safety**
  - **Charge Injection Calibration**
  - **GRB Detection and Response**
  - **Deadtime Monitoring**
  - **Thermal Control**

Function/Consumer Mapping

	I&T: Test Stands		ISIS	I&T: Add'l Integration	FSW Test-bed	Flight Unit Deliverable to I&T
	TKR,CAL	ACD				
Configure TKR and CAL front end electronics	Y			Y	Y	Y
Configure ACD front end electronics		Y		Y	Y	Y
Configure GASU (CRU, GEM, EBM, AEM)		Y	Y	Y	Y	Y
Configure PDU			Y	Y	Y	Y
Configure XBRD	Y	Y		Y		
Configure by compressed file			Y	Y	Y	Y
Real event delivery (instrument to CPU)	Y	Y		Y	Y	Y
Housekeeping data stream	Y	Y	Y	Y	Y	Y
RAD750 boot and crate initialization			Y	Y	Y	Y
1553 bus communications			Y	Y	Y	Y
Telecommand/telemetry database and services			Y	Y	Y	Y
Emulated event delivery (to science data interface)			Y			
CPU internal communications/task frameworks			Y	Y	Y	Y
Software watchdog			Y	Y	Y	Y
Wall clock time services (GPS)			Y	Y	Y	Y

- **Only one specialized function (not in final code base)**
  - **Emulate event delivery (Spectrum Astro interested in rate/protocol/integrity, not physics data)**

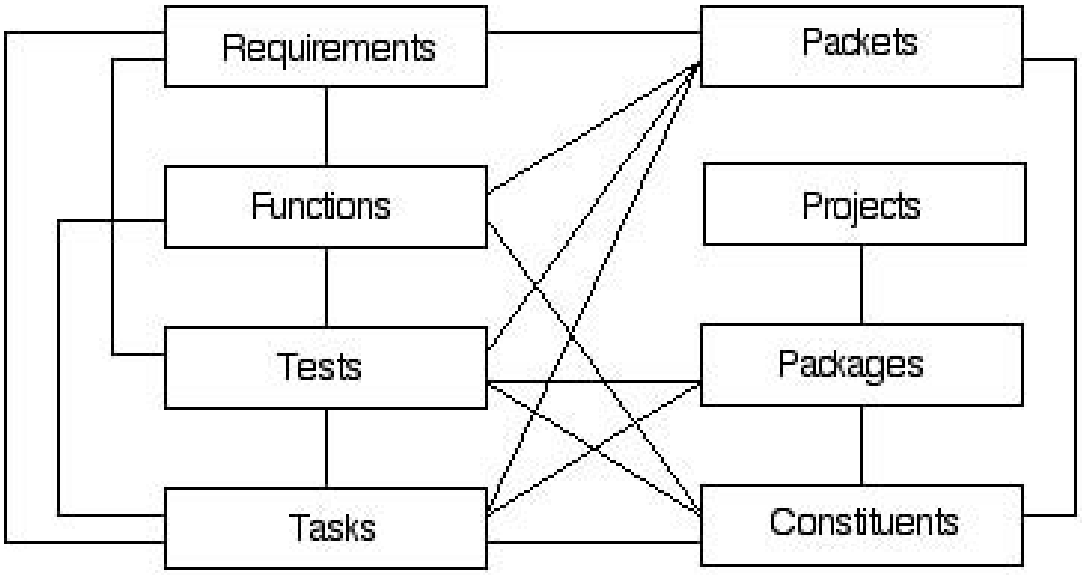
Function/Consumer Mapping (cont'd)

	I&T: Test Stands		ISIS	I&T: Add'l Integration	FSW Test-bed	Flight Unit Deliverable to I&T
	TKR,CAL	ACD				
Mode Control, Safety, and Diagnostics					Y	Y
Charge Injection Calibration				Y	Y	Y
GRB Detection and Response					Y	Y
Deadtime Monitoring				Y	Y	Y
Thermal Control						Y

Backup

Functions & Packages

- **The diagram below illustrates the major entities which FSW must track.**
  - **The top-level requirements are dictated by the goals of the scientific mission and the engineering and physical realities of spaceflight. Lower-level requirements are developed to add necessary detail. Most requirements relate to particular functions (e.g., Thermal Control) that must be performed.**
  - **The FSW group is developing tests to verify that its software meets the relevant requirements. Along with verifying the underlying hardware, these tests verify that the operational functions of the flight software are being performed correctly by the FSW tasks.**
  - **Tasks are assembled, at run time, from libraries of dynamically-loadable constituents. For code-maintenance purposes, constituents are collected into packages and projects.**
  - **Most FSW communication (e.g., between tasks, with the spacecraft) is performed by means of packets. The code to generate, send, accept, and process these is defined in the relevant constituents.**



Functions & Packages (cont'd)

Let's take the Housekeeping function as an example. The entities associated with Housekeeping are shown at right.