GLAST Large Area Telescope

Instrument Flight Software
Flight Unit Design Review
16 September 2004

Software Watchdog

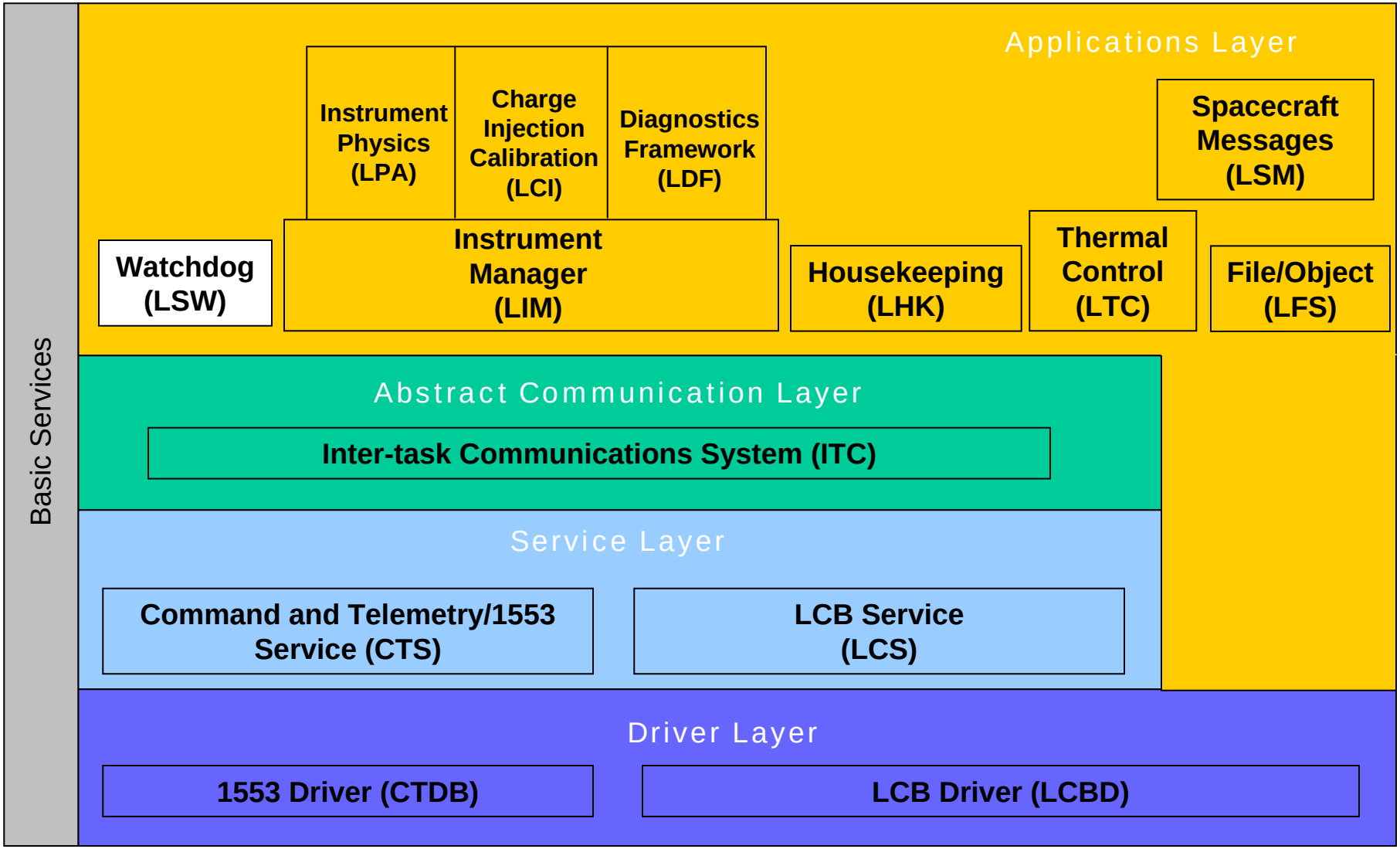
Steve Mazzoni
Stanford Linear Accelerator Center

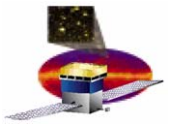
smazzoni@slac.stanford.edu
650-926-4140

Software Watchdog: Requirements

- **Flight Software General Requirements:**
  - **Watchdog (5.3.3.9.2)**
    - **Once booting of a unit is complete, the FSW shall provide a heartbeat to a hardware watchdog that reboots the unit if the heartbeat is not received within 10 seconds.**

FSW Layer Architecture





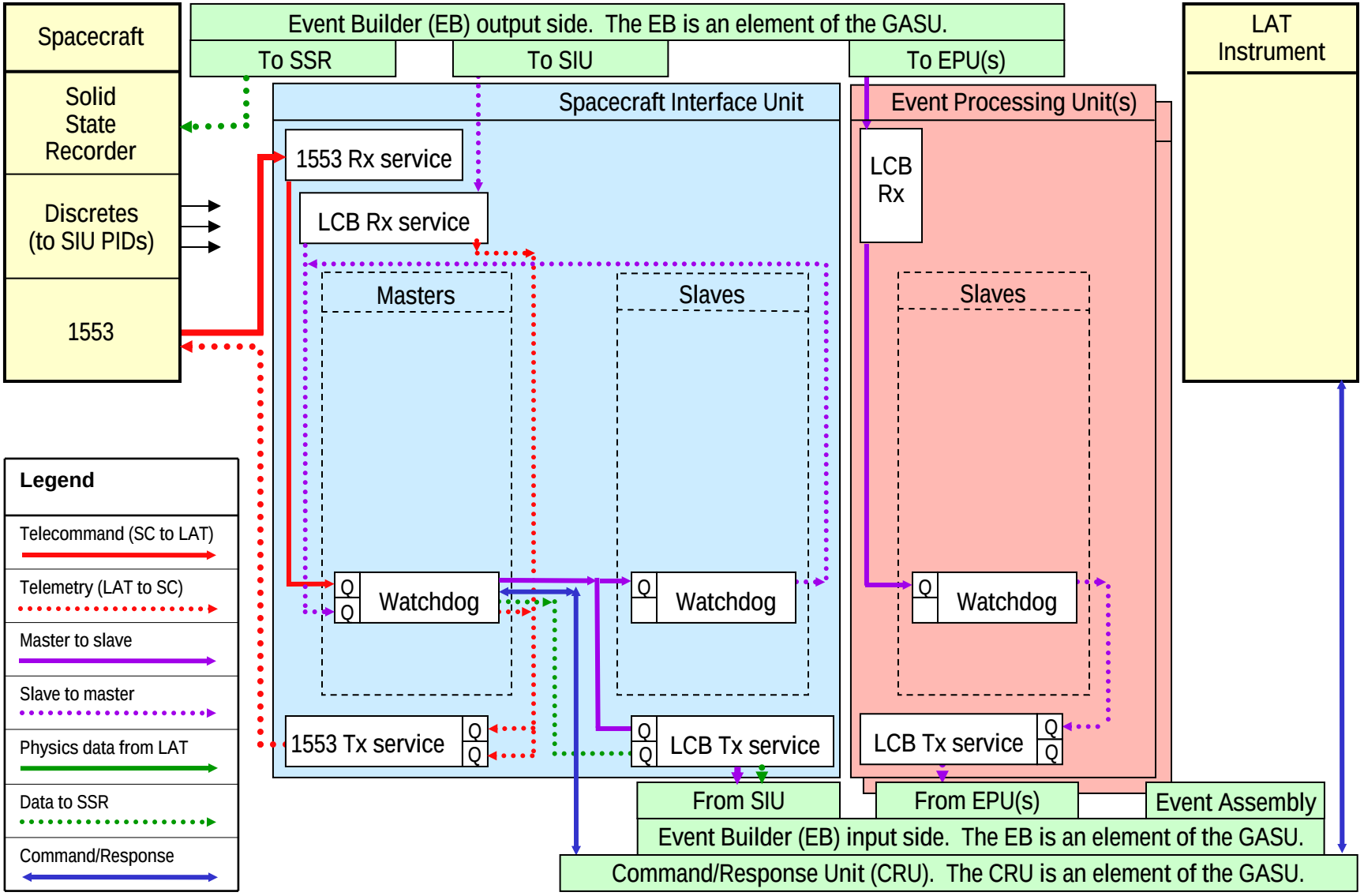
Software Watchdog: Functional Components

- **Functional Inputs**
  - The Watchdog registers callbacks, through which other system tasks report their status/progress
  - The Watchdog can accept telecommands to set operational parameters or instruct the Watchdog to force a timed software or hardware reboot of a CPU
- **Functional Processing**
  - The Watchdog checks progress indicators reported by other system tasks, and decides whether to (1) initiate a software reboot or (2) update the hardware watchdog to prevent a hardware reboot
- **Functional Outputs**
  - At regular intervals, the Watchdog function writes a value into the hardware watchdog timer present on the CPU to prevent the hardware watchdog from resetting the CPU
  - The Watchdog function sends telemetry reporting the Watchdog's current operation mode, number of tasks checked, and a timestamp for the last time a watchdog cycle was run
  - For debugging purposes, records the identity of tasks failing their progress check and the identity of the last task checked prior to reboot or reset

Design Overview

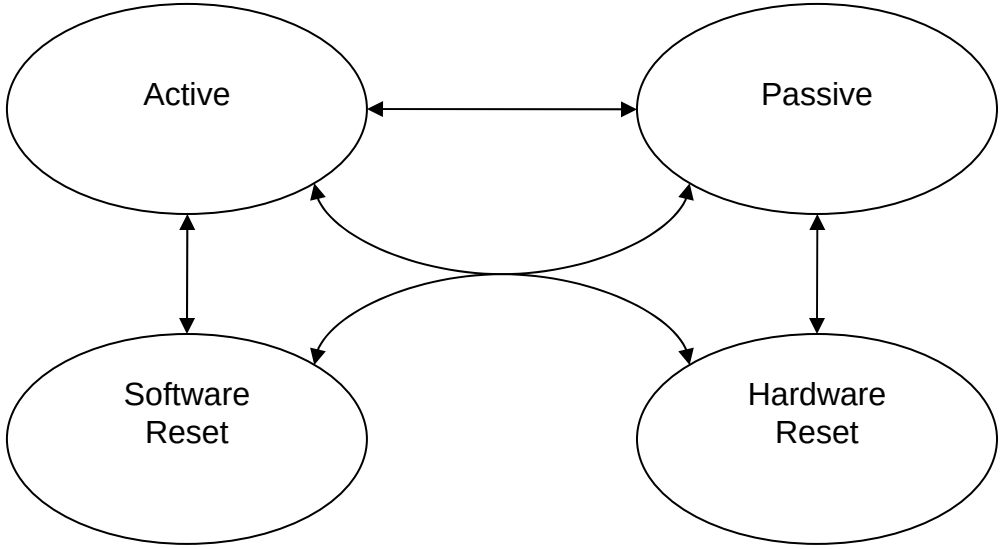
- The software Watchdog is a very simple task, run at high priority.
  - It wakes periodically and resets the hardware watchdog register to a value greater than the software watchdog wake-up period
  - The task itself knows nothing about other tasks in the system. The software watchdog simply provides a uniform facility for tasks to monitor themselves.
- Any task can register a callback routine (and a pointer) with the software Watchdog.
  - When it wakes up, the Watchdog simply runs down the list of registered routines and calls them back in turn
  - The return code from the callback is examined by the Watchdog to determine whether the task targeted by the callback is making progress
    - If any callback indicates that the task is not making progress, the software watchdog will initiate a software reboot once all callbacks have been called
    - If all tasks report progress, the software watchdog will reset the hardware watchdog

FSW Architecture (Watchdog-specific)



Software Watchdog: Operational Modes

- **The Watchdog supports the following modes. Transitions (caused by telecommand) are shown below.**
  - **Active**
    - **“Standard” mode, in which task progress is checked and**
  - **Passive**
    - **Check task progress and always reset the WDT**
  - **Timed Software Reset**
    - **Load time at which software reboot should be performed**
  - **Timed Hardware Reset**
    - **Load specified period into hardware watchdog and never reset**



Software Watchdog: Masters and Slaves

- **The Watchdog tasks collect and evaluate information from other tasks. If the information indicates the presence of a problem, they call a "bug check" mechanism. In addition, they issue the CPU's "heartbeat". If the hardware watchdog does not receive this on a timely basis, it will initiate a hard reset of the CPU's crate.**

- **The master task:**
  - **receives requests for software watchdog change-of-state requests, redistributing them to each targeted slave.**

- **Each slave task:**
  - **Implements the software watchdog function for its CPU (e.g., interrogating registered tasks for progress).**
  - **In passive mode, the slave always refreshes the hardware watchdog.**
  - **In active mode, the slave only refreshes the hardware watchdog if all subsystems report progress; otherwise, it performs a "warm boot".**

Diagnosing the Watchdog In Flight

- **LSW will store information that persists across software reboots or hardware resets that can assist in determining the task responsible for the reboot. The information may be accessed by a memory dump.**
- **The persistent storage items are:**
  - **Circular (FIFO) buffer containing the name of 5 tasks that were not making progress during the last check cycle, i.e. the buffer is cleared after at the start of each progress check cycle.**
  - **Name of last task checked.**
    - **This name is stored immediately preceding the call of each task callback.**
    - **If a task "hangs", its name will be saved in persistent storage, which can be downloaded when the system is started up again.**

Watchdog Telecommands

- **Telecommands**
  - **Though commands and telemetry definitions have not yet been coded using the LCAT tool, the following general types of telecommands will be provided for use by developers and test personnel on the ground; they are not expected to be used in flight.**
    - **Set software watchdog to active mode: This is the bread and butter case. If any task reports a lack of progress, initiate a software reboot.**
    - **Set software watchdog to passive mode: This one is for developers. When debugging a task, it might appear to the software watchdog that the task is not making forward progress. In this mode, the software watchdog will perform all its normal functions, but will always reset the hardware watchdog.**
    - **Set software watchdog to software reset in xxx milliseconds: This is for testers. The software watchdog will follow the software reboot logic after the requested interval, even if all tasks are reporting forward progress.**

Watchdog Telecommands (cont'd)

- **Set software watchdog to hardware reset in xxx milliseconds: This is another one for testers. The software watchdog will set up the hardware watchdog to expire after the requested interval and then do nothing to reset it.**
- **Set software watchdog refresh period to xxx milliseconds and load hardware watchdog with period yyy milliseconds (yyy > xxx): This is another way for developers to put off any software/hardware resets into the far future. It also means that there's no rush to figure out what the best periods for these are.**

– **Diagnostics/debugging data in persistent storage retrieved using memory dump telecommands (not part of the LSW package)**
– **Telecommands are forwarded by the SIU Watchdog master task to all Watchdog slave tasks on all CPUs.**

Watchdog Telemetry

- **Telemetry**
  - **Provided by LSW to signify normal operation. The persistent storage structure is used for debugging after a reboot or reset. Telemetry from LSW includes:**
    - **Current mode.**
    - **Number tasks checked.**
    - **A timestamp for the last time a watchdog cycle was run.**

LSW Software Package Organization and Configuration Management

- The LSW Package contains the following components:
  - Shareables
    - liblsw – provides the watchdog functionality on both SIUs and EPUs

- LSW directly uses the following packages/constituents:
  - PBS
  - MSG