GLAST Large Area Telescope

Instrument Flight Software
Flight Unit Peer Review
16 September 2004

Event Integrity and Delivery

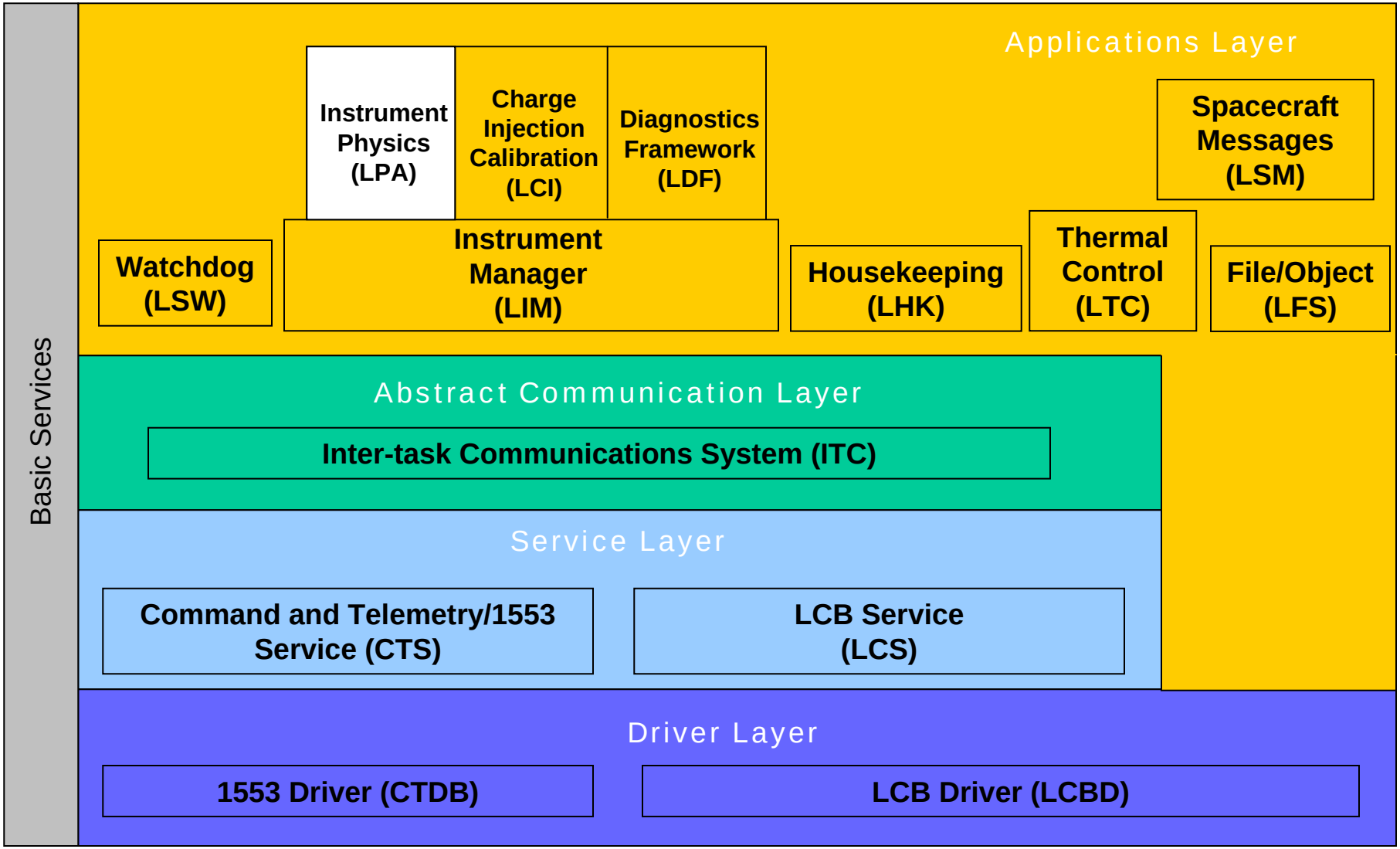
J.J. Russell
Stanford Linear Accelerator Center

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

Event Delivery: Requirements

- **Flight Software General Requirements**
  - **Interface from the Event Builder (5.2.1.2)**
    - **The EPU FSW shall receive fully assembled events directly into EPU memory from the Event Builder, formatted according to the custom hardware and software protocols defined in TEM, AEM, CRU, EBM, GEM, and PDU Programming ICDs.**
  - **Event Monitoring and Delivery (5.3.8)**
    - **The EPU FSW shall monitor event data for integrity and to track changes in event and detector statistics and notify the SIU via CPU-to-CPU protocol in the event of an error or anomaly.**

FSW Layer Architecture

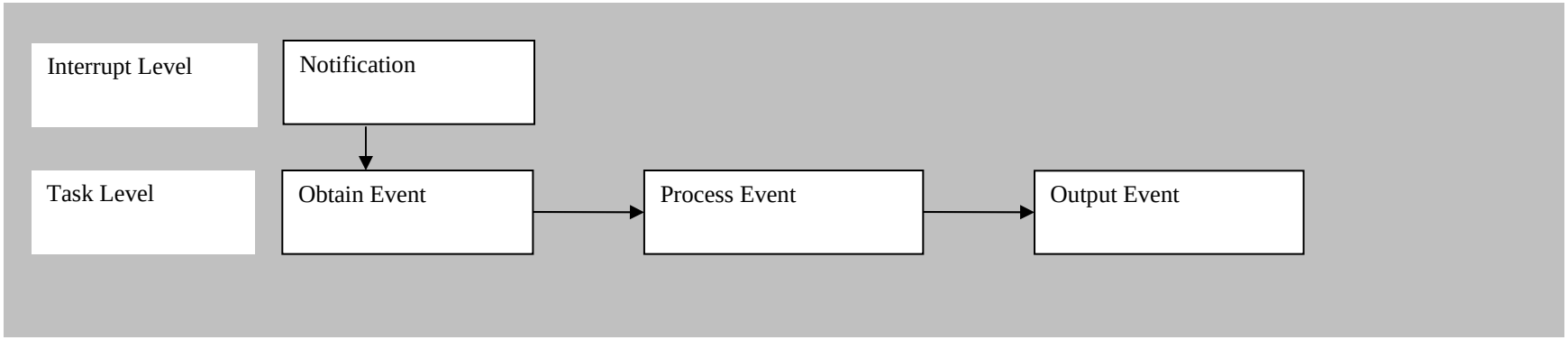


Event Delivery: Functional Components

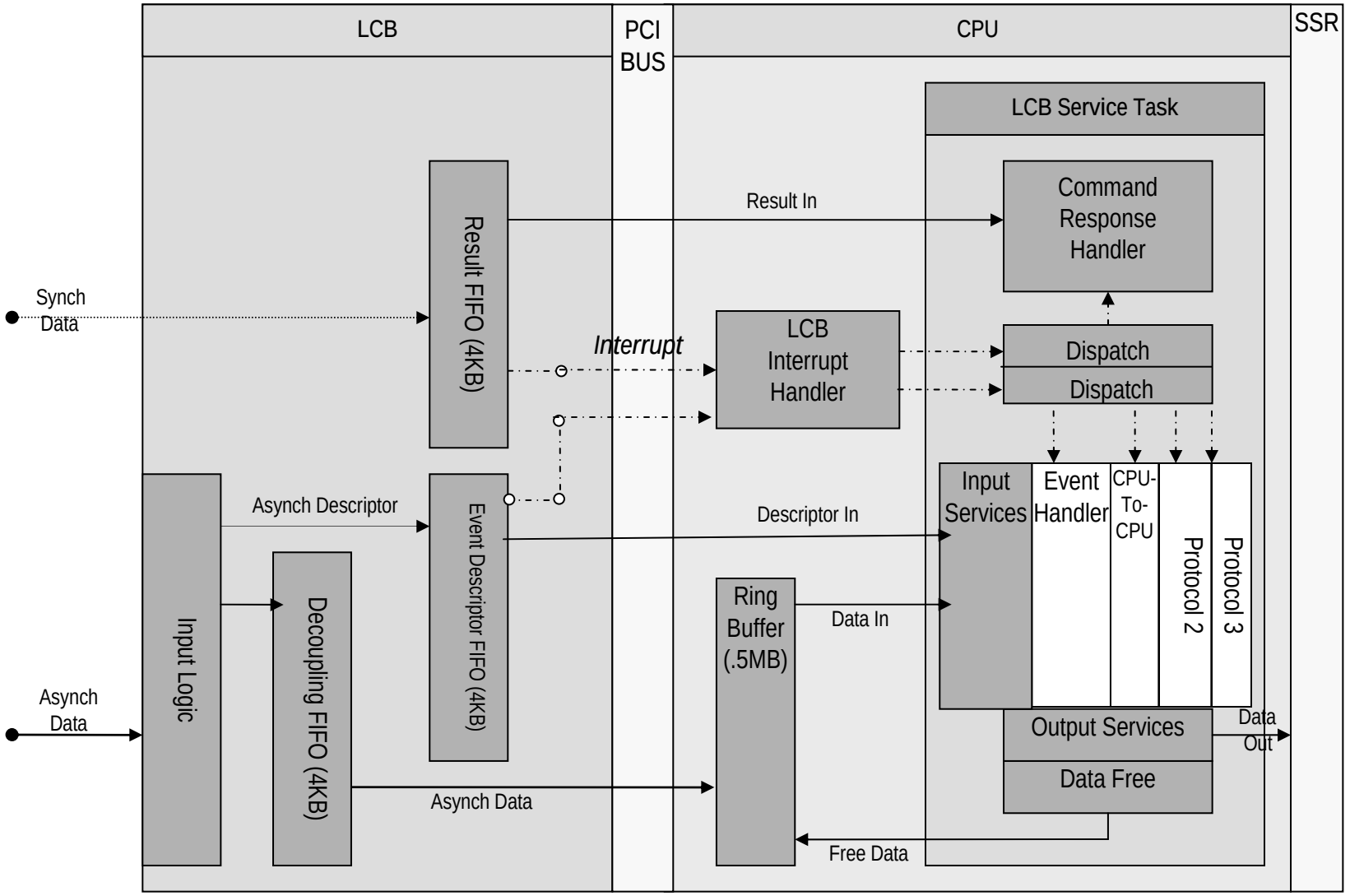
- **Functional Inputs from the Event Builder Module**
  - **The Event Delivery function receives interrupt notification via the LAT Communications Board when data is available**
  - **Once notified, the Event Delivery function begins to receive the asynchronous event data streams via the LAT Communications Board**
- **Functional Processing in FSW**
  - **The Event Delivery function routes event data for processing of various kinds. Event data can be simply passed through to output, be checked for integrity, or be filtered, compressed, etc.**
  - **The Event Delivery function delivers enough information for user code to reassemble entire events.**
- **Functional Outputs to Spacecraft**
  - **Event data packets are delivered as a synch word plus the usual CCSDS packet to the Solid State Recorder on the Spacecraft**
  - **Science data for GRB detection**

Event Delivery: Overview

- One can think of the Event Delivery function as simply moving data through a pipeline, with hardware at the input valve and different hardware at the output valve.
  - Delivery of events is strongly hardware driven: the LAT Communications Board operates on both the input (LAT) side and the output (Solid State Recorder) side. The Event Delivery function is mainly concerned with efficient use of the LCB.
  - The Event Delivery function tries not to concern itself with controlling the input of event data; if events come in, they are processed.
- Event data moves through the pipeline in three primary stages:
  - 1. Notification
  - 2. Processing
  - 3. Disposition



Event Delivery FSW: System Architecture



Event Delivery: Event Notification and Data Servicing

- The LCB Interrupt Handler is served an interrupt:
  - Whenever data arrives in the synchronous FIFO
  - Whenever a high-water mark of asynchronous data builds up in the ring buffer
  - The Interrupt Handler then disables the interrupt
- The LCB Service then determines which FIFO, the synchronous results FIFO or the asynchronous descriptor FIFO, has data. It does this by trying to remove an item from each of the FIFOs in turn.
- If the response FIFO is non-empty, the user-supplied response handling routine shall be called back to handle the data.
- If the descriptor FIFO is non-empty, one of four (one for each LATp protocol type) user-supplied routines is called back.
  - Because the data for all four protocols is piled up in the common ring buffer, the software has no choice but to retrieve the data descriptors in order. This in-order paradigm applies also to returning the data to the ring-buffer.
- The LCB Service Task shall continue to read both FIFOs until they are empty. When both FIFOs are empty, the LCB service routine shall re-enable the interrupts and wait until another interrupt is received.

Event Delivery: Event Processing

- The Event Delivery function performs no “processing” of events. It routes event data descriptors to other functions for separate processing.
- Event Description
  - The LCB service task shall pass a descriptor of the event, containing its length and a pointer to its location in the ring buffer, to any event processing routine.
  - The event processing routine shall use this information to locate and unpack the event for further processing.
- Event Fate
  - The event processing stage shall determine the fate of event.
  - The event processing can dispose of an event in any one or all three of the following ways
    - 1. Return the event to the ring-buffer
    - 2. Pass the event to an output stage
    - 3. Transform the information
  - The event processor is also free to output more than one representation of the event. For example, the filter will, for a subset of events
    - 1. Reject the original event with no output
    - 2. For accepted events, compress the event data and queue it to be written to the SSR
    - 3. For gamma candidates, compose a small data block to pass on to the science analysis software.

Event Disposition

- There are two primary aspects to event disposition:
  - 1. Freeing the memory associated with the input event
  - 2. Writing the output event or some representation of it to an output device(s)
- Event Output
  - The Event Output stage shall be decoupled from the event processing stage if this stage would introduce unacceptable latency.
  - Because of the serial processing nature of the ring buffer, the processing of one event directly impacts the latency of
    - 1. Processing the next event
    - 2. Processing the next response descriptor
  - Therefore, if the writing of an event to the output device causes unacceptable latencies, the event will be placed on a queue to a separate task. Figures of merit for acceptable and unacceptable are on the order of 50 µsecs.
- Event Freeing
  - Either the Event Processing stage or the Event Output stage shall return the event to the ring buffer
  - Thus, other code must provide for a way to free event data in the buffer in proper order

Forward Work

- **Coding is 50% complete**
- **Event delivery with data generated by a Front End Simulators and a full set of hardware is scheduled for January 2005**
- **Coding and unit testing work complete 1/05/2005**