
# GLAST Large Area Telescope

## Instrument Flight Software
## EM2 Design Review
## 29 January 2004

## Functions & Packages II

**A.P.Waite**
**Stanford Linear Accelerator Center**

**apw@slac.stanford.edu**
**(650) 926-2075**


# Configuration

# Configuration: Overview

- **Purpose**
  - **To place and maintain the instrument in a known configuration**
  - **To report the configuration**
    - **Knowledge of instrument configuration is *essential* to a successful physics analysis**
- **The instrument appears to flight software as a collection of registers**
  - **A *large* collection**

| Subsystem / Block | Chip / Sub-block | Number in Instrument | Register Count | Register documentation |
|---|---|---|---|---|
| TKR | GTFE | 13824 | 69120 | LAT-SS-00169 |
| | GTRC | 1152 | 2304 | LAT-SS-00170 |
| CAL | GCFE | 3072 | 21504 | LAT-TD-00424 |
| | GCRC | 256 | 2048 | LAT-SS-00208 |
| ACD | GAFE | 216 | 2376 | LAT-SS-00363 |
| | GARC | 12 | 528 | |
| TEM | GTCC | 128 | 768 | LAT-TD-00605 |
| | GCCC | 64 | 448 | |
| | GTIC | 16 | 272 | |
| | Common | 16 | 112 | |
| GASU | AEM | 1 (primary and redundant) | 6 | LAT-TD-00639 |
| | CRU | 1 (primary and redundant) | 4 | LAT-TD-01547 |
| | EBM | 1 (primary and redundant) | 31 | LAT-TD-01546 |
| | GEM | 1 (primary and redundant) | 59 | LAT-SS-01596 |
| PDU | | | 7 | LAT-SS-01543 |
| | | | 99587 | |

# Configuration: Mechanism

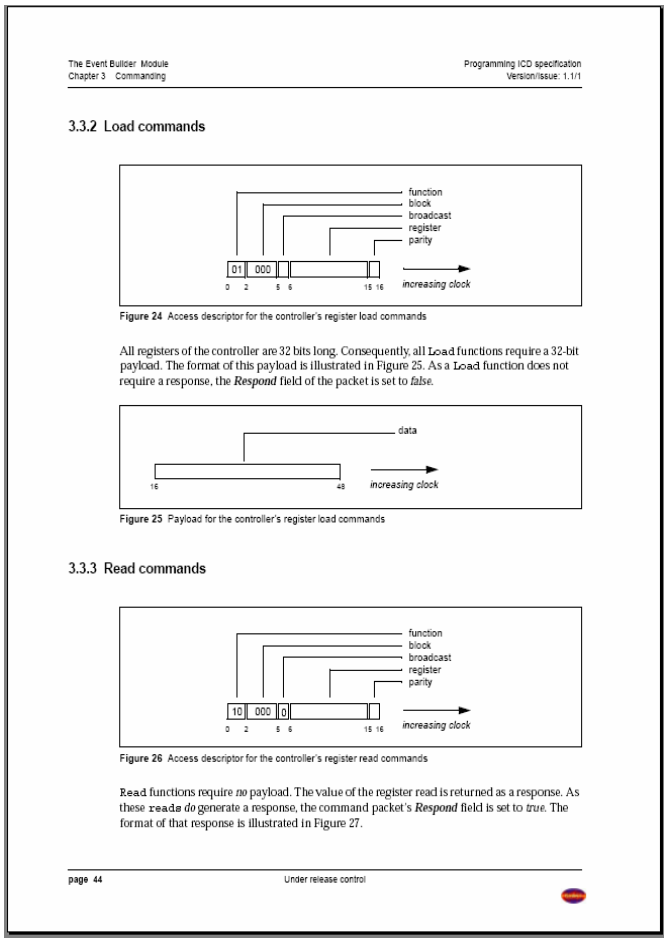
- **Communicating with registers**
  - Communicating with registers is very uniform throughout the system
  - Instrument flight software supports the register operations
    - Read register
    - Write register
    - "Dataless"
      - Special commands like reset
  - Instrument communications are "bit serial"
    - Read / write / dataless commands are constructed as bit strings in CPU memory
    - Bit strings are sent to LAT Communications Board (LCB) for transmission to the instrument
    - Responses from the instrument are presented back in CPU memory by the LCB
  - Construction of the bit strings is the responsibility of packages DEM and DAB
    - DEM: ACD, CAL, & TKR "front-end" registers (GxFE and GxRC) and TEM registers
    - DAB: All elements of GASU (AEM, CRU, EBM and GEM) and PDU
  - Instrument communications are the responsibility of package LCB

- **Further Documentation**
  - LAT-TD-00606 — LAT Inter-module Communications
  - LAT-TD-01380 — LCB Driver
  - LCB — Package LCB APIs
  - DEM — Package DEM APIs
  - DAB — Package DAB APIs
  - DAB Programmer's Guide

# Configuration: Example

Extract from LAT-TD-01546

EBM Programming ICD



The Event Builder Module
Chapter 3 Commanding

Programming ICD specification
Version/Issue: 1.1/1

3.3.2 Load commands



**Figure 24** Access descriptor for the controller's register load commands

All registers of the controller are 32 bits long. Consequently, all `Load` functions require a 32-bit payload. The format of this payload is illustrated in Figure 25. As a `Load` function does not require a response, the ***Respond*** field of the packet is set to *false*.



**Figure 25** Payload for the controller's register load commands

3.3.3 Read commands



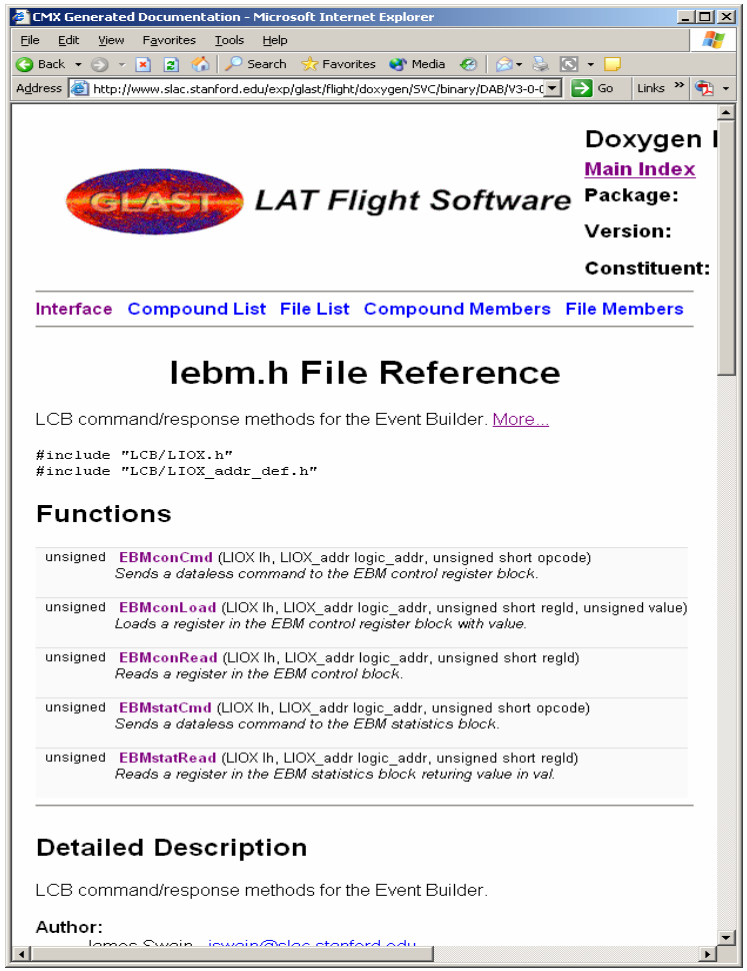
**Figure 26** Access descriptor for the controller's register read commands

`Read` functions require *no* payload. The value of the register read is returned as a response. As these `reads` *do* generate a response, the command packet's ***Respond*** field is set to *true*. The format of that response is illustrated in Figure 27.

page 44 Under release control

Extract from package DEM API documentation

Callable interface to EBM registers



CMX Generated Documentation - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://www.slac.stanford.edu/exp/glast/flight/doxygen/SVC/binary/DAB/V3-0-0

Doxygen 1

Main Index

LAT Flight Software

Package:

Version:

Constituent:

Interface Compound List File List Compound Members File Members

## lebm.h File Reference

LCB command/response methods for the Event Builder. More...

```
#include "LCB/LIOX.h"
#include "LCB/LIOX_addr_def.h"
```

### Functions

unsigned **EBMconCmd** (LIOX lh, LIOX_addr logic_addr, unsigned short opcode)
*Sends a dataless command to the EBM control register block.*

unsigned **EBMconLoad** (LIOX lh, LIOX_addr logic_addr, unsigned short regId, unsigned value)
*Loads a register in the EBM control register block with value.*

unsigned **EBMconRead** (LIOX lh, LIOX_addr logic_addr, unsigned short regId)
*Reads a register in the EBM control block.*

unsigned **EBMstatCmd** (LIOX lh, LIOX_addr logic_addr, unsigned short opcode)
*Sends a dataless command to the EBM statistics block.*

unsigned **EBMstatRead** (LIOX lh, LIOX_addr logic_addr, unsigned short regId)
*Reads a register in the EBM statistics block returing value in val.*

### Detailed Description

LCB command/response methods for the Event Builder.

**Author:**

# Configuration: Status

- Configure CAL, TKR front-end electronics
  - Implemented for COMM I/O boards
    - Packages GTEM, GNAT, … (working, being used in field, part of EM1 build)
  - Re-implemented for LCB
    - Packages DEM, LCB
    - In use at SLAC for hardware commissioning
- Configure ACD front end electronics
  - Emulation implemented for EM1 to let ACD proceed with their testing
    - Packages GAEM, GNAT, … (working, being used in field, part of EM1 build)
  - Re-implemented for LCB and for real hardware
    - Packages DEM, LCB
    - In use at SLAC for hardware commissioning
- Configure GASU
  - Implemented for COMM I/O boards (initial commissioning only)
    - Packages GDAB, … (working, used at SLAC, now being superceded)
  - Re-implemented for LCB
    - Handles all elements of GASU (CRU, AEM, GEM, EBM)
    - Packages DAB, LCB
    - In use at SLAC for hardware commissioning
- Configure PDU
  - Part of package DAB (see previous bullet)
- Configure XBRD
  - Implemented for EM1
    - Packages GGLT, GNAT, … (working, being used in field, part of EM1 build)
    - A test-stand only package: no further development expected

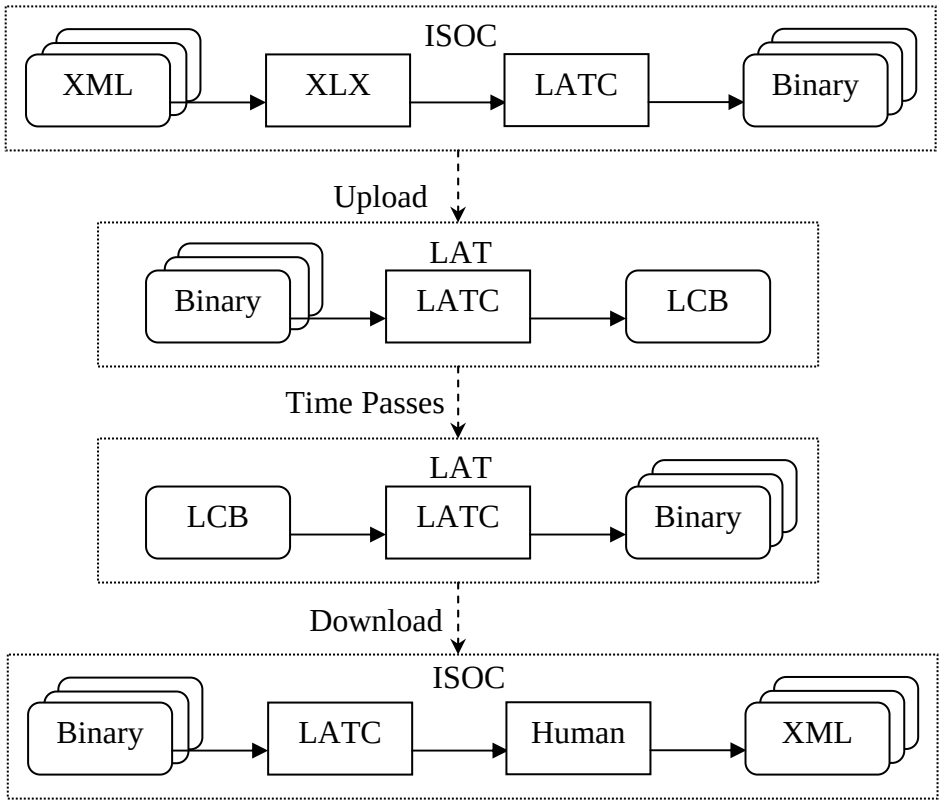
- DEM, DAB and LCB will be part of I&T code distribution for new generation of LCB-based test stands

# Configure From Files: Overview

- Full instrument configuration description is large
  - Impractical to configure instrument with a series of telecommands
  - Impractical to record instrument configurations with a “simple” register dump
- Instrument flight software has developed methods for segmenting and compressing configurations
  - Segmentation
    - Define a file format, describing a hierarchical description of all registers
    - Allow any single file to populate only a small segment of this hierarchy
    - Reading a series of files builds up a full configuration
      - The first file is usually the “golden” instrument configuration (which can use wildcards heavily)
      - Subsequent files then annotate deviations from the golden configuration
      - A particular target configuration (e.g., trigger configuration) can be put in a file by itself and reused
  - Compression
    - The above files start in XML format
      - Easy to describe, code manage, and exchange with other LAT groups
      - Flight software and I&T are converging the XML file format
    - A (host/ground based) tool converts the XML to a compact binary format
    - The resulting binary format is compressed using ZLIB, producing a “LATC” file
- On board handling
  - On board software reads a set of LATC-compressed files to build an “in–memory” configuration:
    - To set the instrument configuration
    - To verify the instrument configuration
    - To dump the instrument configuration

# Configure From Files: Data Flow

- **Extracted from LAT-SS-01597 LAT Configuration**



# Configure From Files: Status

- Status
  - All the functional blocks described have been implemented
    - Package XLX converts files XML <-> LATC
    - Package LATC does all compressed file handling
      - Read file(s)
      - Write file(s)
      - Set instrument to configuration
      - Verify instrument against configuration

- Left to do
  - Expand the XML “dictionary” to describe GASU, PDU, redundant paths, …
  - Re-implement the LATC register read/write functions to use LCB communications

- Further Documentation
  - LAT-SS-01597 LAT Configuration
  - LATC Developer Guide
  - LATC Programmer Guide
  - XLX Programmer Guide


# Event Delivery

# Event Delivery

- Two very distinct blocks of functionality
  - Deliver *events* from target hardware though flight software to test stands
  - Deliver *traffic* through science data interface to spacecraft (ISIS)

- Test stand event delivery
  - A pass through from the hardware to the test stand
    - Does not filter
    - Does not reformat
  - Implemented for COMM I/O boards
    - Packages OES (Online Event Server), GGLT, GNAT, … (working, being used in field, part of EM1 build)
  - Re-implemented for LCB
    - Packages OES, LCB, …
    - In use at SLAC for hardware commissioning
- ISIS traffic delivery
  - Spectrum Astro wish to test the science data interface
    - Protocol
    - Integrity
    - Bandwidth
  - For this purpose, they do not need physics data
  - Instead, LAT flight software will provide a data stream emulation
    - Test patterns will be generated on the CPU and written to science data interface
    - Test pattern will be configurable
    - Data rate will be configurable

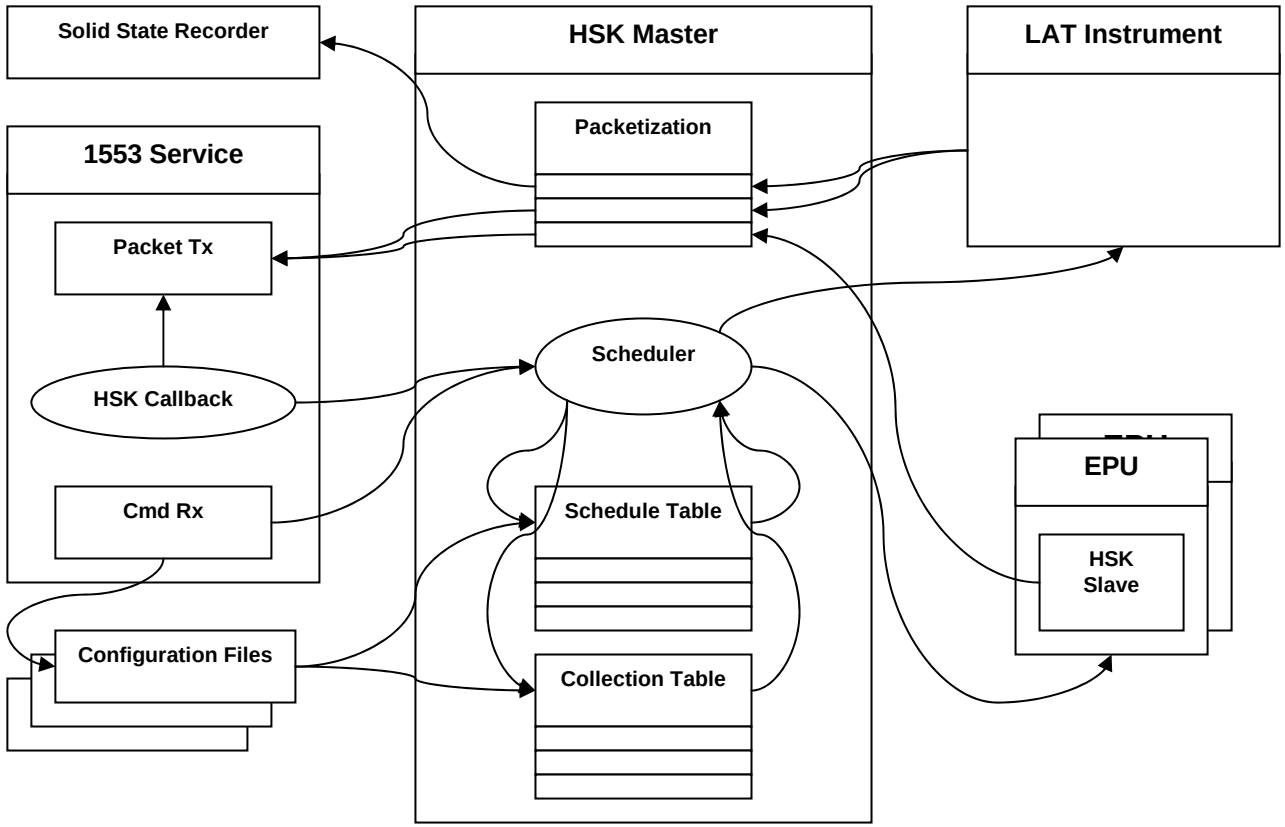

# Housekeeping

# Housekeeping: Overview

- Monitor the state and health of the instrument:
  - Hardware monitors
    - Switch positions, temperatures, voltages, currents, …
    - Hardware command counts, trigger counts, livetime, …
  - Software monitors
    - Software command counts, idle time, memory usage, …
- Do this independently of the event acquisition path
  - Hardware monitoring primarily via LCB command/response path
  - Software monitoring by exchange of messages between CPUs
  - Housekeeping runs as independent task(s)
- Design document: LAT-TD-02905
  - Enumerates all the measurands available in hardware
  - Begins to enumerate CPU metrics
  - Lays out telemetry packets
  - Describes the operations of the housekeeping master task
    - Based on a time-slot scheduler
    - Slots assigned a block of housekeeping to gather
      - Slot to block-of-data mapping is table driven, so easy to modify
      - Some blocks can be scheduled more frequently than others
    - Some slots held open to accommodate commanded collection of data
  - As described, can telemeter complete LAT state/health information in ~20 seconds
    - But the list of measurands is likely to grow
- Status
  - First pass implementation in developer private area

# Housekeeping: Scheduler

- **Scheduler diagram extracted from LAT-TD-02905**



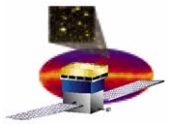
# Housekeeping: Telemetry Packet

- **Telemetry packet description extracted from [LAT-TD-02905]**

**Table 19** TEM Environmental Packet

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Version=0 | | | T=0 | SH=1 | | | | | | APID | | | | | |
| 1 | SF=3 | | | | | | Sequence Count | | | | | | | | | |
| 2 | Packet Length=116 | | | | | | | | | | | | | | | |
| 3 | Timestamp Seconds MSW | | | | | | | | | | | | | | | |
| 4 | Timestamp Seconds LSW | | | | | | | | | | | | | | | |
| 5 | Timestamp Subseconds MSW | | | | | | | | | | | | | | | |
| 6 | Timestamp Subseconds LSW | | | | | | | | | | | | | | | |
| 7 | Timestamp delta MSW | | | | | | | | | | | | | | | |
| 8 | Timestamp delta LSW | | | | | | | | | | | | | | | |
| 9 | | | | Spare | | | | | Validity | | | | Channel | | | Group |
| 10 | | | ADC1 | | | | | | ADC0 | | | | | | | |
| 11 | | | ADC2 | | | | | | | | | ADC1 | | | | |
| 12 | | | | | ADC3 | | | | | | | | | ADC2 | | |
| 13 | | | ADC5 | | | | | | ADC4 | | | | | | | |
| 14 | | | ADC6 | | | | | | | | | ADC5 | | | | |
| 15 | | | | | ADC7 | | | | | | | | | ADC6 | | |
| 16 | | | ADC1 | | | | | | ADC0 | | | | | | | |
| 17 | | | ADC2 | | | | | | | | | ADC1 | | | | |
| 18 | | | | | ADC3 | | | | | | | | | ADC2 | | |
| 19 | | | ADC5 | | | | | | ADC4 | | | | | | | |
| 20 | | | ADC6 | | | | | | | | | ADC5 | | | | |
| 21 | | | | | ADC7 | | | | | | | | | ADC6 | | |
| 22 | | | ADC1 | | | | | | ADC0 | | | | | | | |
| 23 | | | ADC2 | | | | | | | | | ADC1 | | | | |
| 24 | | | | | ADC3 | | | | | | | | | ADC2 | | |
| 25 | | | ADC5 | | | | | | ADC4 | | | | | | | |
| 26 | | | ADC6 | | | | | | | | | ADC5 | | | | |
| 27 | | | | | ADC7 | | | | | | | | | ADC6 | | |
| 28 | | | ADC1 | | | | | | ADC0 | | | | | | | |
| 29 | | | ADC2 | | | | | | | | | ADC1 | | | | |
| 30 | | | | | ADC3 | | | | | | | | | ADC2 | | |
| 31 | | | ADC5 | | | | | | ADC4 | | | | | | | |
| 32 | | | ADC6 | | | | | | | | | ADC5 | | | | |
| 33 | | | | | ADC7 | | | | | | | | | ADC6 | | |
| 34 | | | ADC1 | | | | | | ADC0 | | | | | | | |
| 35 | | | ADC2 | | | | | | | | | ADC1 | | | | |
| 36 | | | | | ADC3 | | | | | | | | | ADC2 | | |
| 37 | | | ADC5 | | | | | | ADC4 | | | | | | | |
| 38 | | | ADC6 | | | | | | | | | ADC5 | | | | |
| 39 | | | | | ADC7 | | | | | | | | | ADC6 | | |
| 40 | | | ADC1 | | | | | | ADC0 | | | | | | | |
| 41 | | | ADC2 | | | | | | | | | ADC1 | | | | |
| 42 | | | | | ADC3 | | | | | | | | | ADC2 | | |
| 43 | | | ADC5 | | | | | | ADC4 | | | | | | | |
| 44 | | | ADC6 | | | | | | | | | ADC5 | | | | |
| 45 | | | | | ADC7 | | | | | | | | | ADC6 | | |
| 46 | | | ADC1 | | | | | | ADC0 | | | | | | | |
| 47 | | | ADC2 | | | | | | | | | ADC1 | | | | |
| 48 | | | | | ADC3 | | | | | | | | | ADC2 | | |
| 49 | | | ADC5 | | | | | | ADC4 | | | | | | | |
| 50 | | | ADC6 | | | | | | | | | ADC5 | | | | |
| 51 | | | | | ADC7 | | | | | | | | | ADC6 | | |
| 52 | | | ADC1 | | | | | | ADC0 | | | | | | | |
| 53 | | | ADC2 | | | | | | | | | ADC1 | | | | |
| 54 | | | | | ADC3 | | | | | | | | | ADC2 | | |
| 55 | | | ADC5 | | | | | | ADC4 | | | | | | | |
| 56 | | | ADC6 | | | | | | | | | ADC5 | | | | |
| 57 | | | | | ADC7 | | | | | | | | | ADC6 | | |


# Telecommand / Telemetry

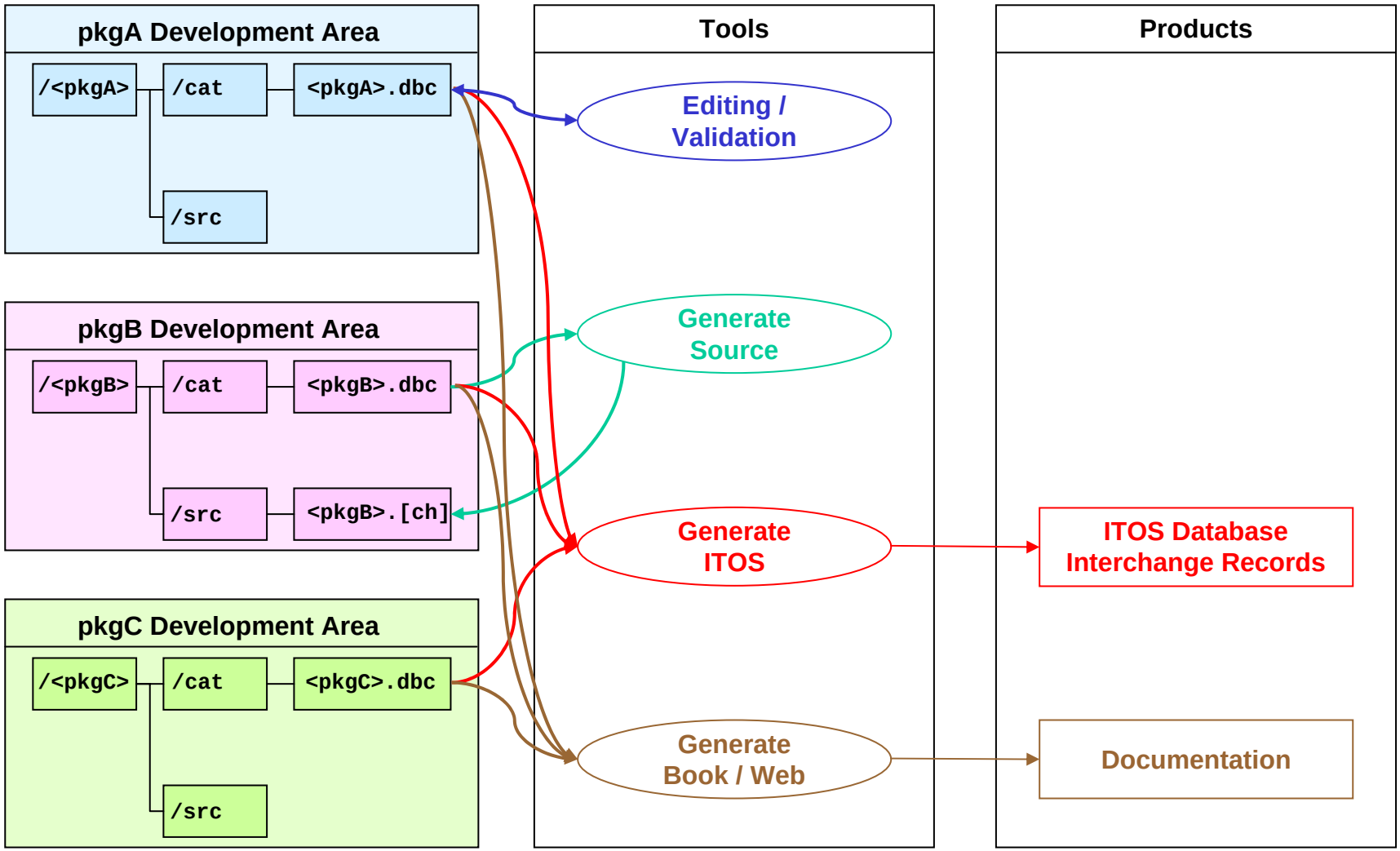


# Telecommand / Telemetry: Overview

- **Telecommand / telemetry database and services**
  - **Constraints**
    - **During the mission MOC will use ITOS**
    - **During satellite integration, Spectrum Astro will use AstroRT**
    - **LAT must produce a command and telemetry database document**
    - **LAT must produce code corresponding to the C&T database document**
  - **Many products, but all derived from the same basic information**
    - **A maintenance nightmare, if all products are maintained independently**

- **Design**
  - **A tools-based approach**
    - **Define a private, XML-based representation of this information**
    - **Provide an editor tool to manipulate the XML files**
    - **Provide extraction tools to produce**
      - **ITOS database interchange records**
      - **AstroRT compatible ITOS database interchange records**
      - **Telecommand and telemetry web pages and book**
      - **Source code for inclusion by FSW applications**
  - **Integration into developer's environment**
    - **Any package defining telecommands or telemetry will have a `/cat` directory**
    - **Developer will create package-specific telecommand / telemetry XML files in `/cat`**
    - **XML files are code managed under our present system**
    - **Deriving code source files from XML files will be built into our build scheme**
  - **Books and web pages are created by farming `/cat` directories in all packages**

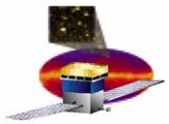
# Telecommand / Telemetry: Workflow



# Telecommand / Telemetry: Status

- Status
  - Proof of principle implementations completed for:
    - Editing XML files
    - Extracting ITOS database exchange records
    - Building a telecommand / telemetry web pages and book
      - Web access to telecommand / telemetry book and web pages
    - Dropping out source code
- Future
  - Converge on ITOS vs. AstroRT database representations
    - AstroRT nominally reads ITOS database records, but
      - Doesn't understand all of them
      - Mistranslates in some cases (bit fields in telecommands)
  - Finalize the XML file format to satisfy all consumers
  - Finalize the tools


# Inter-Task Communications

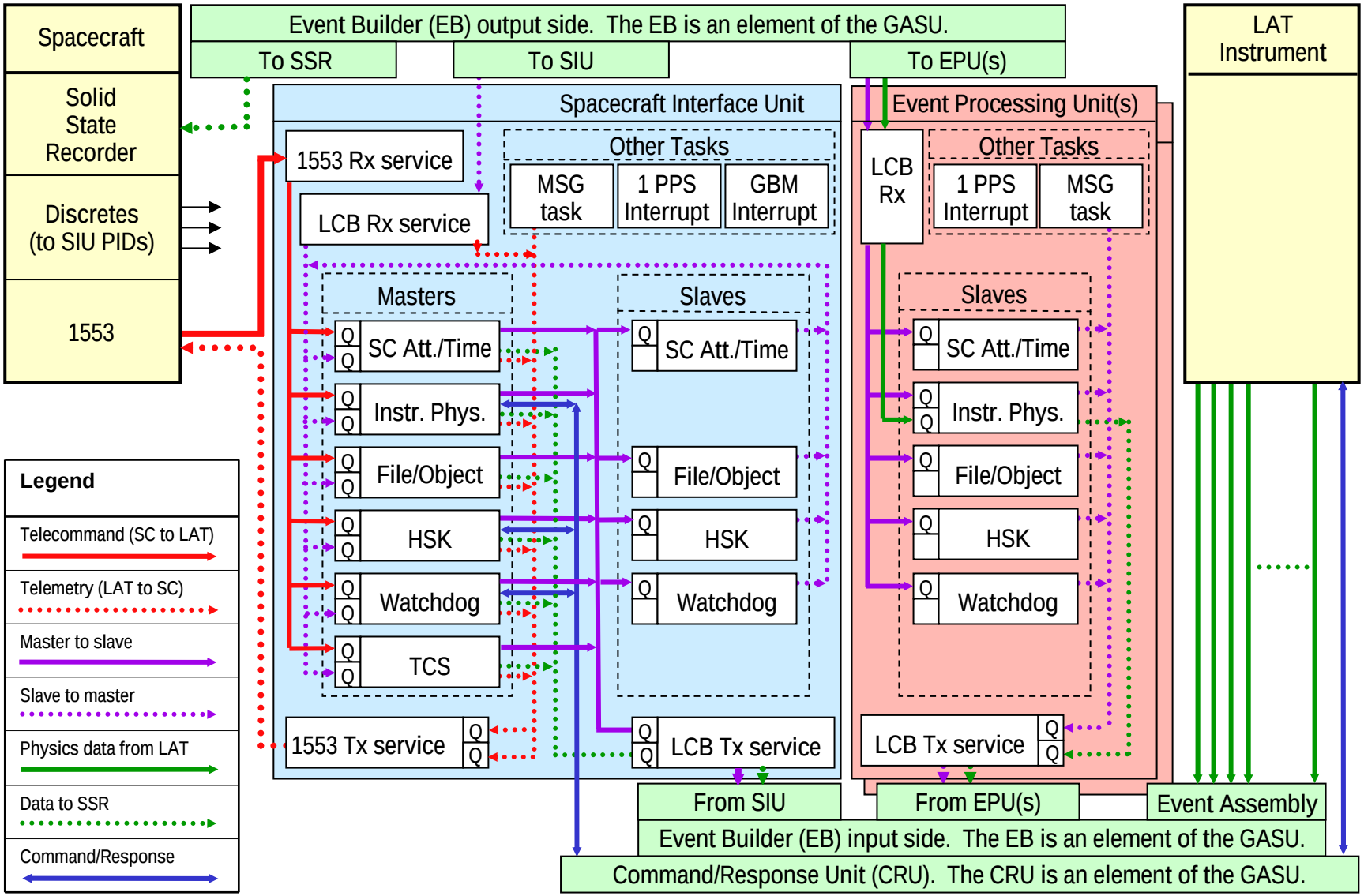


# Inter-Task Communications: Overview

- **Task communications / task frameworks**
  - **Package to provide developers with standard methods for setting up tasks with communication queues:**
    - **Define and initialize a task's incoming queues**
    - **Receive and dispatch messages on incoming queues**
    - **Send messages to other tasks**
  - **In other words, the *arrows* on the architecture diagram**

- **Forward Planning**
  - **While not strictly needed for EM2, the package is being designed to be upwardly compatible with multi-CPU operation:**
    - **Send messages to other tasks on other CPUs**
    - **Provide task-synchronization mechanism**
      - **The "scatter/gather" problem**
      - **Master task must sometimes synchronize responses from multiple slaves**

# LAT FSW Architecture



# Inter-Task Communications: Status

- **Package must deal with multiple formats/protocols**
  - **1553-mediated CCSDS telecommands from spacecraft**
  - **1553-mediated CCSDS telemetry to spacecraft**
  - **LCB-mediated unsolicited CPU to CPU communications**
  - **LCB-mediated unsolicited instrument to CPU communications**
  - **LCB-mediated CPU to spacecraft Solid State Recorder (SSR) communications (the science data interface)**

- **Status**
  - **Toolbox from which to build this package already available in package PBS**
    - **Prioritized queues with queue masking and timeouts**
    - **Interlocked operations on queues**
    - **…**
  - **Now running "use case" exercises for all tasks communicating via queues**
  - **Identifying split between single CPU and multi-CPU implementation**
    - **Implement single CPU for EM2, but provide all the hooks for multi-CPU**
  - **With most of our low level utilities, drivers, and services coming under control, this is the next package to be implemented**


# Software Watchdog

# Software Watchdog

- **Software watchdog**
  - **Primary function: Update the hardware watchdog periodically to pre-empt a hardware watchdog reset**
  - **Primary problem: This is too easy in a multi-tasking system!**
    - **Other tasks could be dead or dying but if the software watchdog task is healthy, it will continue to update the hardware watchdog**
    - **Real design challenge is not "how to refresh the hardware watchdog", but "deciding if the hardware watchdog *should* be refreshed"**

- **Status**
  - **Preliminary design document written**
    - **Identifies the above problem and provides a solution**
      - **Other tasks register a "progress indicating" routine with the software watchdog task**
      - **Before refreshing the hardware watchdog, software watchdog will run through registered routines to check that all tasks are making progress**
      - **If task(s) are not making progress, software watchdog performs a software reset**
    - **This reduces the hardware watchdog function to catching code spins at high priority level or a lock up at interrupt level**
      - **A software-initiated reset is always preferable to a hardware reset**
  - **Once the concept is grasped, this is not a large coding project**
    - **Has been assigned to our new software engineer (Steve Mazzoni)**

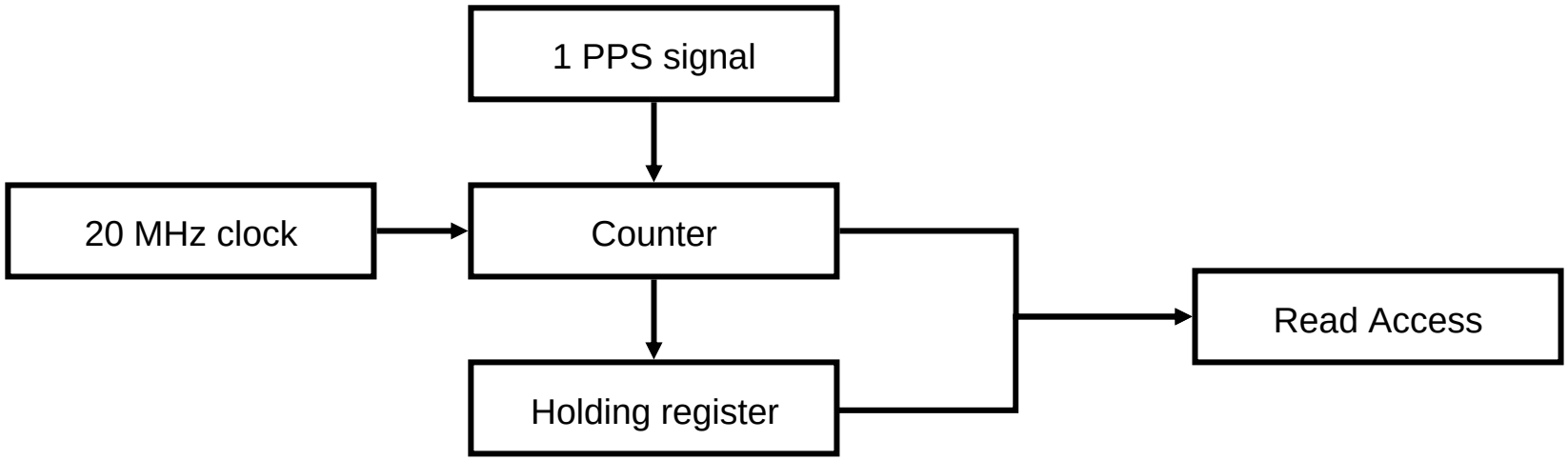

# Wall Clock Time

# Wall Clock Time: Overview

- **Wall clock time**
  - **LAT FSW is required to time tag all photon events to one microsecond**
  - **Wall clock time is defined for us by the spacecraft**
    - **A once a second message "At the tone, the time will be …"**
    - **A once a second time hack (a dedicated signal line)**
  - **FSW has chosen to extend this precision to all time tags**
    - **All times from all sources (events or CPUs) will be reported to better than one microsecond, wall clock time**
  - **Implementation requires a small amount of software and a large amount of hardware support**

- **Principles of operation**
  - **Same for both event time tagging and CPU time tagging**
    - **Drive a counter from the instrument 20 MHz clock**
    - **Strobe the current value of this counter into a holding register with the spacecraft 1 PPS signal**
    - **Reading the counter provides "current time" (in counter units)**
      - **Stamp this value into triggered events**
    - **CPU uses 1 PPS signal interrupt to read strobed values in holding registers, using the data to build tables of wall clock time (provided by the spacecraft messages) against counter value**
    - **Counter time can be converted to wall clock time by interpolating/extrapolating through these tables**

# Wall Clock Time: Operation




- **Event time tags**
  - **Counter and holding register physically located on GEM**
  - **When 1 PPS received, current counter value is strobed to holding register (SIU reads this back when it receives its own 1 PPS interrupt)**
  - **When event triggered, current value of counter is part of event data**
- **Documented in**
  - **LAT-SS-01596**

- **CPU time tags**
  - **Counter and holding register are provided on the RAD750 (specially configured PIDs)**
  - **When 1 PPS received, current counter value strobed to holding register (all CPUs read this back when they receive their own 1 PPS interrupt)**
- **Documented in**
  - **BAE Hardware manual section 4.8**
  - **LAT PID assignments**

# Wall Clock Time: Status

- **Status**
  - **Design understood**
    - **Spacecraft attitude/time master task distributes spacecraft messages to all slaves**
    - **Spacecraft attitude/time slave task populates wall clock time side of table(s)**
    - **1 PPS task (all CPUs) populates counter side of table(s)**
      - **With strobed GEM counter (SIU only)**
      - **With strobed PID counter (SIU and EPU)**
    - **Accessor routines convert GEM counter and PID counter times to wall clock time through built up tables**


# Function / Package Mapping

# Function / Package Mapping

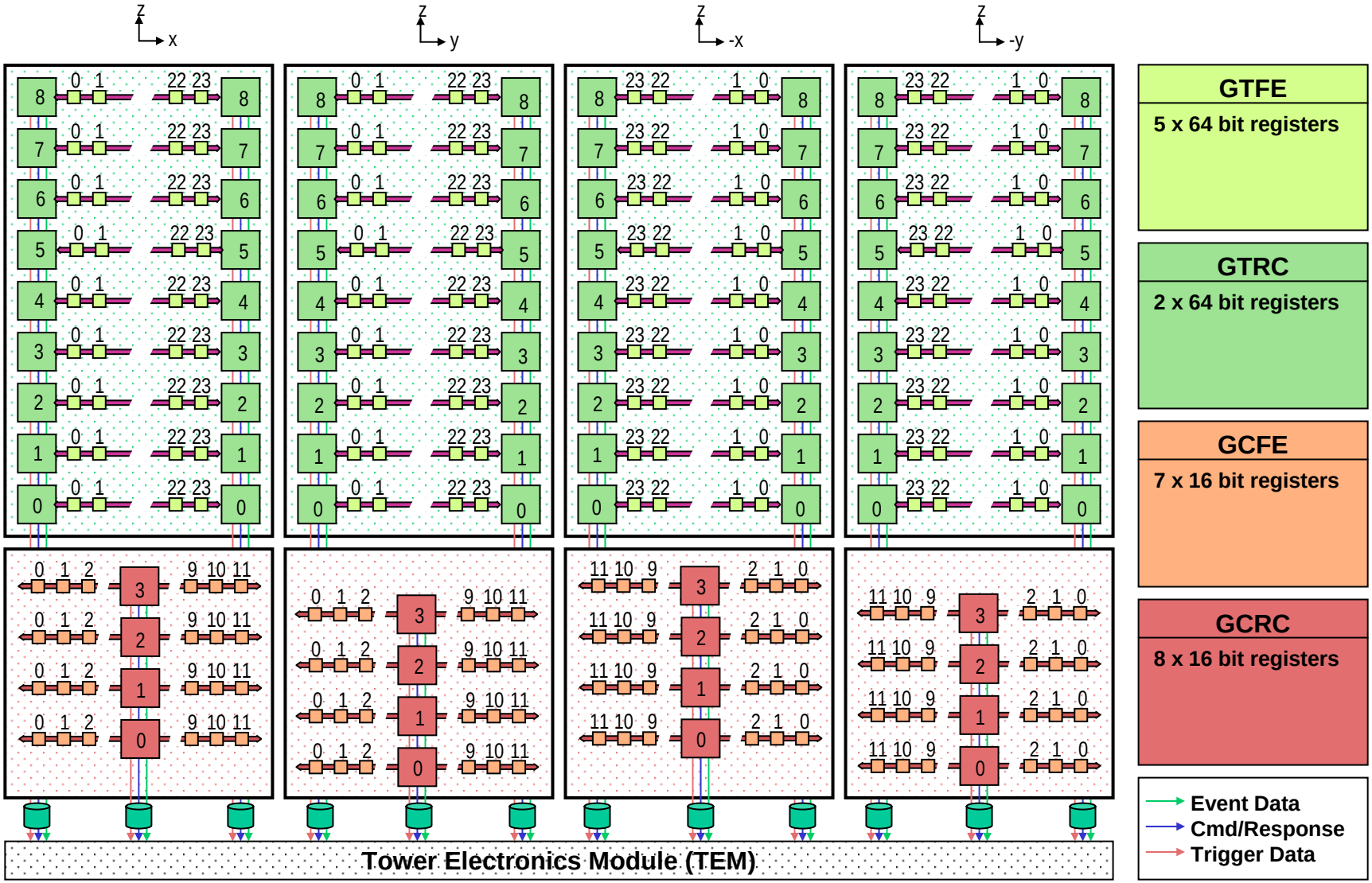
| | | Package | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | (CTD) | (GPS) | (HSK) | (ITC) | (LSW) | CCSDS | CLI | CMX | CTDB | DAB | DEM | EXPAT | FILE | GGLT | GNAT | GRL | LATC | LCB | MSG | OCS | OES | PBC | PBI | PBS | VXW | XLX | ZLIB |
| Function | Configure TKR and CAL front end electronics | | | | | | | | Y | | | Y | | | | | | | Y | Y | | | | Y | Y | Y | | |
| | Configure ACD front end electronics | | | | | | | | Y | | | Y | | | | | | | Y | Y | | | | Y | Y | Y | | |
| | Configure GASU (CRU, GEM, EBM, AEM) | | | | | | | | Y | | Y | | | | | | | | Y | Y | | | | Y | Y | Y | | |
| | Configure PDU | | | | | | | | Y | | Y | | | | | | | | Y | Y | | | | Y | Y | Y | | |
| | Configure XBRD | | | | | | | | Y | | | | | | Y | Y | | | Y | Y | | | | Y | Y | Y | | |
| | Configure by compressed file | | | | | | | | Y | | | | Y | | | | | Y | | Y | | | | Y | Y | Y | Y | Y |
| | Real event delivery (instrument to CPU) | | | | | | | | Y | | Y | Y | | | Y | | | | Y | Y | Y | Y | | Y | Y | Y | | |
| | Housekeeping data stream | | | (Y) | | | | Y | Y | | Y | Y | | | Y | | Y | | Y | Y | | | | Y | Y | Y | | |
| | RAD750 boot and crate initialization | | | | | | Y | Y | Y | Y | | | | Y | | | Y | | | Y | | | Y | Y | Y | Y | | Y |
| | 1553 bus communications | | | | | | Y | Y | Y | | | | | | | | Y | | | Y | | | | Y | Y | Y | | |
| | Telecommand/telemetry database and services | (Y) | | | | | | | Y | | | | | | | | | | | | | | | | | | | |
| | Emulated event delivery (to science data interface) | | | | | | | | Y | | | | | | | | | | | | | | | | | | | |
| | CPU internal communications/task frameworks | | | | (Y) | | | | Y | | | | | | | | | | | | | | | | | | | |
| | Software watchdog | | | | | (Y) | | | Y | | | | | | | | | | | | | | | | | | | |
| | Wall clock time services (GPS) | | (Y) | | | | | | Y | | | | | | | | | | | | | | | | | | | |

(Y): Package exists in developer private area or not yet created

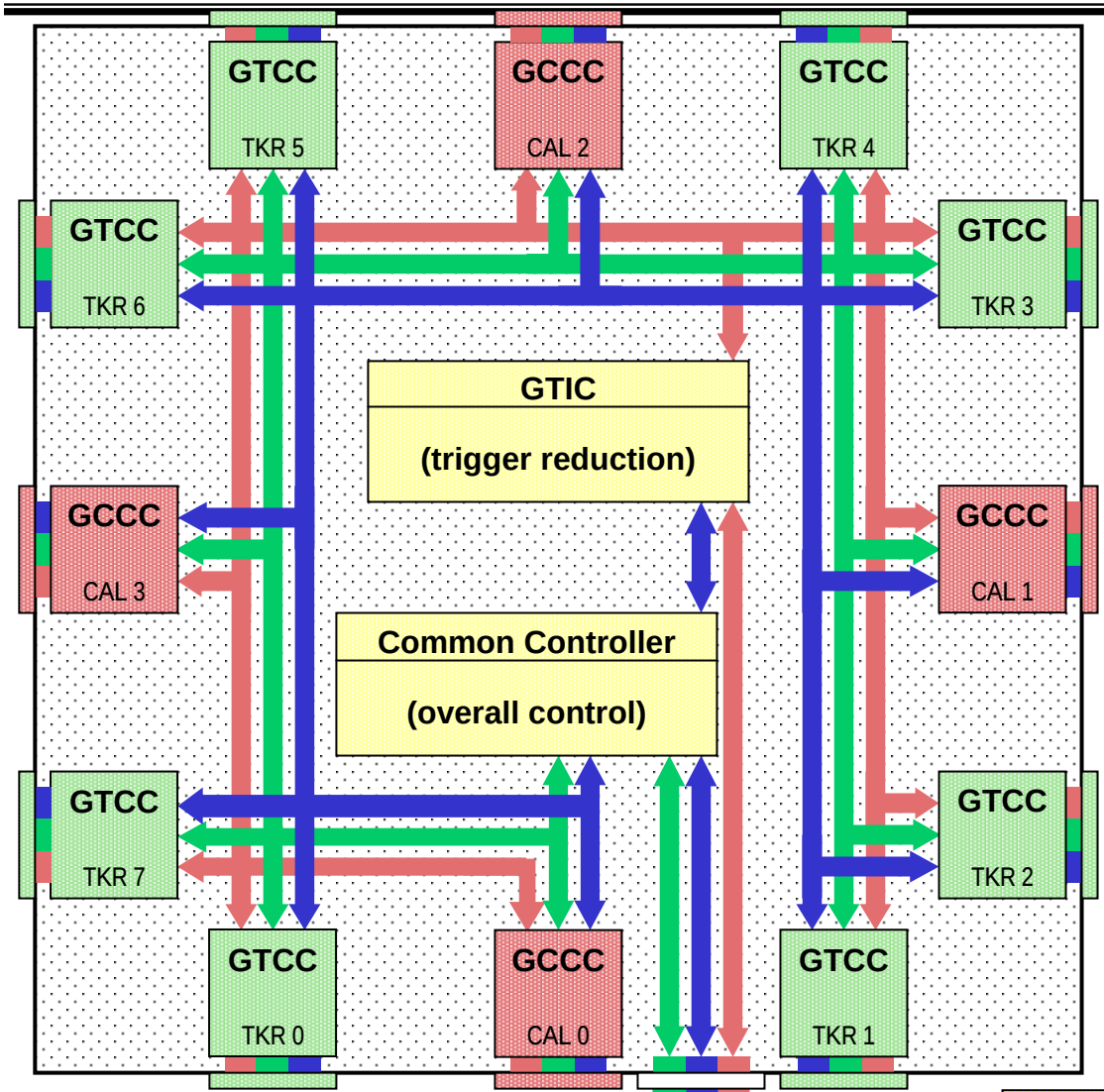
- **Access to package API documentation (main index)**


# Appendix

# TKR & CAL Front End



# Tower Electronics Module (TEM)




**GTCC**
6 x 32 bit registers

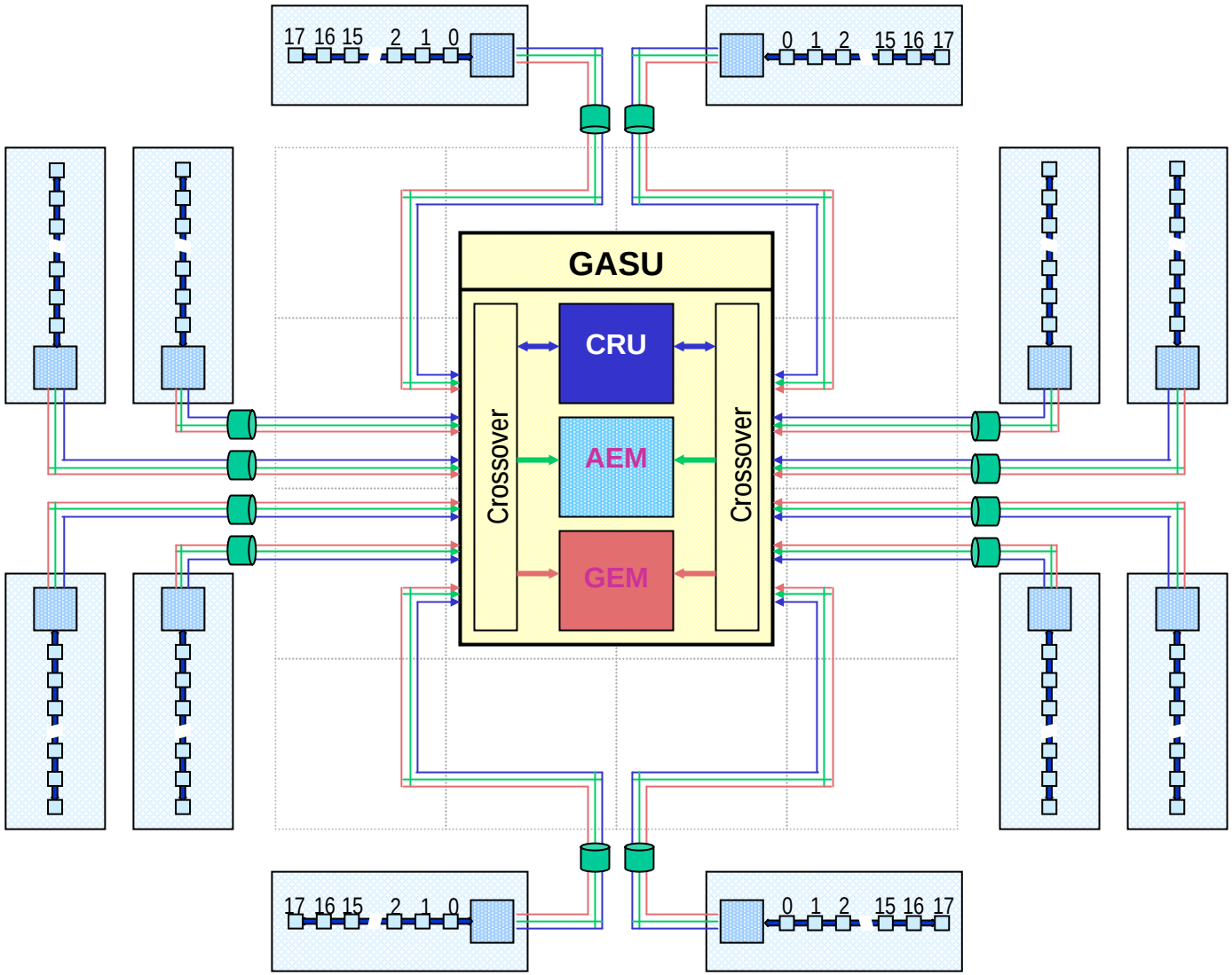
**GCCC**
7 x 32 bit registers

**GTIC**
11 x 32 bit registers
2 x 18 bit registers
2 x 16 bit registers
1 x 3 bit register
1 x 112 bit register

**Common**
7 x 32 bit registers

Event Data
Cmd/Response
Trigger Data

# ACD



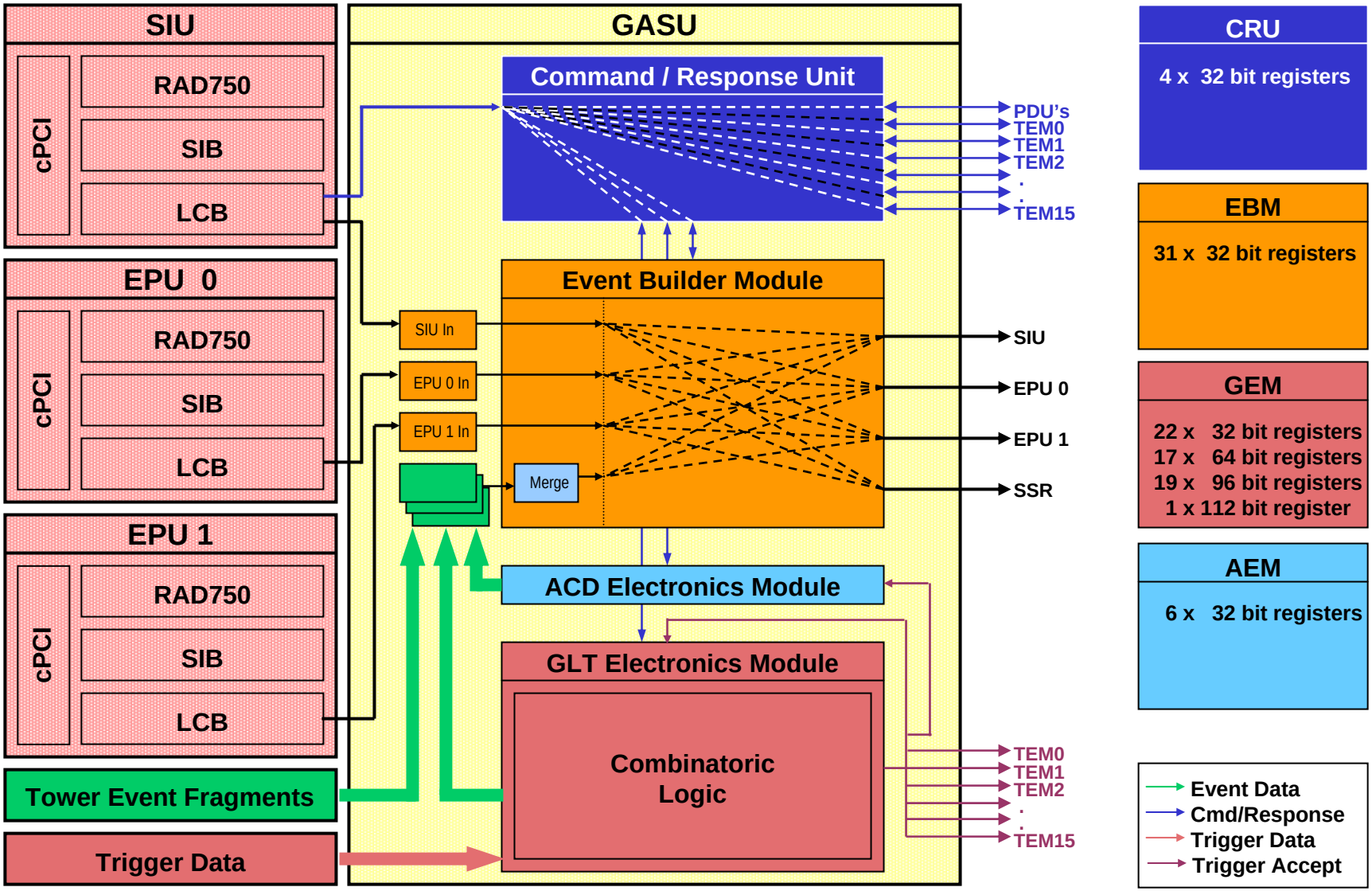

**GAFE**
11 x 16 bit registers

**GARC**
44 x 16 bit registers

**AEM**
6 x 32 bit registers

Event Data
Cmd/Response
Trigger Data

# GASU


SIU
cPCI
RAD750
SIB
LCB
EPU 0
cPCI
RAD750
SIB
LCB
EPU 1
cPCI
RAD750
SIB
LCB
Tower Event Fragments
Trigger Data
GASU
Command / Response Unit
PDU's
TEM0
TEM1
TEM2
TEM15
Event Builder Module
SIU In
EPU 0 In
EPU 1 In
Merge
SIU
EPU 0
EPU 1
SSR
ACD Electronics Module
GLT Electronics Module
Combinatoric Logic
TEM0
TEM1
TEM2
TEM15
CRU
4 x 32 bit registers
EBM
31 x 32 bit registers
GEM
22 x 32 bit registers
17 x 64 bit registers
19 x 96 bit registers
1 x 112 bit register
AEM
6 x 32 bit registers
Event Data
Cmd/Response
Trigger Data
Trigger Accept