Electron Losses and Fields Investigation

Mission Critical Design Review
Command and Data Handling

Jordan Miller

Los Angeles, California
February 17, 2016
- Flight Computer Board (FCPCB)
  - Flight Computer PIC18F8722
  - Watchdog PIC18F6722

- Solar Battery Power Boards (SBPCB) x2
  - Power PIC18LF6722 (x2)

- ADCS Board (ACB)
  - ADCS Main PIC24FJ256GA108
  - ADCS Peripheral Controller PIC18LF25K22

- CDH is responsible for the software for the specified Avionic Boards
  - This software interfaces with each other and non-avionics
DELIVERABLES

- Software that meets requirements
  - Programs for each PIC
    - Flight Computer, Watchdog, 2x Power Boards, ADCS Main, Peripheral Controller

- Spreadsheet of command and telemetry frames
  - Used for issuing commands to the spacecraft

- Memory-Map spreadsheets

- Interface Control Document (*ELF_CDH_ICD_v06*)

- Test procedures and scripts
The software design flows down from the requirements and is influenced by the avionics architecture.

- Avoid closed-loop algorithms
  - Push complexity to the ground

- Use of Memory Maps to simplify data collection and to modify software parameters

- Iterative, test-driven development

- Modular Software
  - Libraries that are shared among processors
- Subsets of safe mode
- Normal Operations modes
Scheduled resets are full power resets

State after reset determined by value in the EEPROM
• Processes commands from the ground
• Gathers and stores housekeeping data
• Gathers and transfers science data
• “Central Hub” for other boards
• Interfaces with radio to send science and housekeeping data to ground
  • Frames data to proper format for radio
• Hosts the scheduler
▪ Monitors Flight Computer

▪ Capable of resetting the system if the Flight Computer is stalled
  ▪ “Heartbeat” pulses are sent between the Watchdog and Flight Computer

▪ Switch Flight Computer software
  ▪ Flight computer will hold two copies of the software
    ▪ “Side A, Side B”

▪ Monitors critical temperature telemetry and determines if it should reset the spacecraft
**Power Board x2**

- Collect telemetry
  - Solar cell arrays
  - Batteries
  - Telemetry stored in double buffer system

- Regulate power

- Send telemetry to Flight Computer
ADCS Board (ACB)

- Collects data from ADCS Peripheral Controller
- Runs Control Law Algorithm
- Controls orientation of the satellite
Mission CDR
Command and Data Handling

Software and Algorithms
- Frame format is common to all processors
- 255 available operational codes
- Data section may be used for parameters
- Escape codes used to escape special codes
• Bytes are transmitted one by one, and are parsed into the frame, which consists of the OP code, payload, and the CRC. This frame is then used by the PIC to execute the command.

• Parser acts as a state machine to reduce time spent in the interrupt handler.
- Allows spacecraft to be able to execute commands at certain times
  - Schedule entry: 8 bytes long
    - Timestamp: 6 bytes
      - Execution/stale flags located here
    - Address to script: 2 bytes
  - Entries checked every sleep-wake cycle
- Issues
  - Slow due to flash memory pushing
  - It pushes the execution/stale bits at the end -> potential duplicate execution
• Scripts
  • sequence of commands that the Flight Computer can accept and execute
  • Scripts are variable in length – a special stop script command specifies the end of the script

• Tests
  • Scripts were written into the scheduler along with stale, executed, and normal entries and it behaved correctly
Other Features

- **File system**
  - Periodic data stored in file system to allow easy retrieval
  - Location: Flight Computer; ADCS Main (planned), Watchdog (planned)

- **Uplink Authentication**
  - Prevent unauthorized access to our spacecraft using SHA-1 hash, a salt, and a secret key

- **SideA/SideB**
  - Two copies of the same code to address program corruption
  - Location: Flight Computer

- **Early Orbit Operations**
  - Heavily relies on the scheduler

- **Radio interface with Flight Computer**
  - Uplink, not downlink, has been tested
<table>
<thead>
<tr>
<th>Test #</th>
<th>Feature</th>
<th>DM Verification Date</th>
<th>Test</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Command parsing</td>
<td>07/10/14</td>
<td>Send a command to FC to change the LEDs from GSE</td>
<td>success</td>
</tr>
<tr>
<td>2</td>
<td>Scheduler</td>
<td>08/29/14</td>
<td>Write some entries into the scheduler that execute written scripts at certain timestamps</td>
<td>success</td>
</tr>
<tr>
<td>3</td>
<td>Power board (1) Interface</td>
<td>12/19/14</td>
<td>Have FC request and receive power data; also write into power boards memory map</td>
<td>success</td>
</tr>
<tr>
<td>4</td>
<td>File System: read packet (1 power board)</td>
<td>7/24/2015</td>
<td>Have FC send request to power board for data then send request from GSE to read packet</td>
<td>success</td>
</tr>
<tr>
<td>5</td>
<td>File System: two power boards</td>
<td>8/5/2015</td>
<td>Have FC request packets from two power boards, storing the data in file system; then request for the data</td>
<td>success</td>
</tr>
<tr>
<td>6</td>
<td>Radio Interface:FC sends and receives commands from He-100</td>
<td>8/11/15-8/25/15</td>
<td>FC sends and receives commands which include noop, read/write configuration from Radio</td>
<td>success</td>
</tr>
<tr>
<td>7</td>
<td>IDPU Interface:FC reads from IDPU</td>
<td>9/15/2015</td>
<td>Have FC read IDPU version</td>
<td>success</td>
</tr>
<tr>
<td>8</td>
<td>IDPU Interface:FC writes to IDPU</td>
<td>9/15/2015</td>
<td>Have FC write IDPU LEDS</td>
<td>success</td>
</tr>
<tr>
<td>9</td>
<td>IDPU Interface:FC boots IDPU</td>
<td>9/15/2015</td>
<td>Have FC send boot</td>
<td>success</td>
</tr>
<tr>
<td>10</td>
<td>IDPU Interface:FC reads &amp; writes booted IDPU</td>
<td>9/15/2015</td>
<td>Have FC send command and get MSP430 echo</td>
<td>success</td>
</tr>
<tr>
<td>11</td>
<td>IDPU Interface: FC reads IDPU with PWR on same bus</td>
<td>9/15/2015</td>
<td>Have FC read IDPU version</td>
<td>success</td>
</tr>
<tr>
<td>12</td>
<td>IDPU Interface:FC writes IDPU with PWR on same bus</td>
<td>9/15/2015</td>
<td>Have FC write IDPU LEDS</td>
<td>success</td>
</tr>
<tr>
<td>13</td>
<td>ADCS Interface:FC writes to ADCS Main</td>
<td>9/16/2015</td>
<td>Have FC write a command and receive an ACK from ADCS Main</td>
<td>success</td>
</tr>
<tr>
<td>14</td>
<td>Implement sideA/sideB using a script</td>
<td>9/30/2015</td>
<td>Run a script that adds sideA/sideB to Flight Computer (mostly without input); test to see if it switches sides upon reset when the toggle flag is enabled</td>
<td>success - reading from the EEPROM in the interrupt redirection header prevented the Flight Computer from parsing through a series of commands fast enough (for some reason)</td>
</tr>
<tr>
<td>15</td>
<td>Have Flight Computer receive a byte that was transmitted from He-100 Radio</td>
<td>10/31/2015</td>
<td>Use TNC to send a byte to the He-100 that is connected to the Flight Computer</td>
<td>success</td>
</tr>
<tr>
<td>16</td>
<td>Uplink authentication key validity</td>
<td>09/01/15</td>
<td>Send command with correct/incorrect SHA-1 verify only correct commands are executed</td>
<td>success</td>
</tr>
<tr>
<td>17</td>
<td>Uplink authentication resynchronization</td>
<td>09/02/15</td>
<td>Send command generated with incorrect salt and verify that ground software can resynchronize from NACK</td>
<td>success</td>
</tr>
<tr>
<td>18</td>
<td>ADCS MRM data collection</td>
<td>10/07/2015</td>
<td>ACB collects MRM data, sends to ADCS Main, ADCS main prints to GSE</td>
<td>success</td>
</tr>
<tr>
<td>19</td>
<td>CRC validation</td>
<td>??</td>
<td>(one of the first few tests) isn't executed</td>
<td>success</td>
</tr>
<tr>
<td>20</td>
<td>Memory mirroring validation</td>
<td>08/19/14</td>
<td>Write incorrect data to one mirror in flash, verify error is flagged</td>
<td>success</td>
</tr>
</tbody>
</table>
Mission CDR
Command and Data Handling

Tools and Testing
**Development Tools**

- **MPLAB X IDE v2.20**
  - Linux
  - Embedded C
  - XC8 Compiler v1.33B, XC16 Compiler
    - Free Version of the compilers

- **MPLAB ICD3**
  - Used to load program onto PICs

- **Development boards**
  - PICDEM PIC18 EXPLORER Demo Board
  - EXPLORER 16 Development Board

- **Prototyping boards**
  - FCPCB rev J, SBPCB rev G, ACB rev A
Bench tests
- Tests performed on development and engineering boards
- Test modules
- System level tests

Unit Tests (Planned)
- Automated tests for testing PIC modules
  - Command parsing, I2C, SPI, CRC
- Unit tests evolve and grow alongside software
- Will help prevent bugs from recurring with new versions of software
- Only one unit test written so far
- Received recommendation to use the flash memory chip for storing test data and feeding it into different interfaces (for testing ICs)
- GSE software created to test programs
  - Computer program to print out anything sent over from the PIC to a Linux computer over UART
  - Computer program that sends and receives frames over UART
- GSE software changes during PIC development
- The GSE software used for testing will be separate from the GSE software used during integration testing, environmental testing, and flight.
<table>
<thead>
<tr>
<th>RFA</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get in contact with Reviewer to discuss Spacecraft Resets</td>
<td>Closed</td>
</tr>
<tr>
<td>Produce state transition diagrams (modes), as well as data flow diagrams</td>
<td>In progress</td>
</tr>
<tr>
<td>Specifics on how requirements are verified and tested</td>
<td>Closed</td>
</tr>
<tr>
<td>Diagram for Logic in Code</td>
<td>Closed</td>
</tr>
<tr>
<td>Flat Sat diagram and organization</td>
<td>Closed</td>
</tr>
<tr>
<td>Define CNDH Requirement 12 - &quot;Engineering magnetometer telemetry shall be collected at a sufficient rate to implement the ADCS control law algorithm as specified by the ADCS subsystem.&quot;</td>
<td>Closed</td>
</tr>
<tr>
<td>Flash Memory Chip Performance</td>
<td>Closed</td>
</tr>
<tr>
<td>Check transaction time between ADCS controller request and Honeywell magnetometer</td>
<td>Moved to ADCS subsystem</td>
</tr>
<tr>
<td>Amend CNDH Requirement 11 - &quot;C&amp;DH shall be capable of jumping the clock on command from the ground station&quot;</td>
<td>Closed</td>
</tr>
<tr>
<td>CNDH requirement for RTCC drift – <em>Tied to RFA to amend Requirement 11</em></td>
<td>Closed</td>
</tr>
<tr>
<td>Operator Specification document – provide constraints document to OPs team</td>
<td>In progress</td>
</tr>
<tr>
<td>Early Orbit Timers – detail what timers are involved in early orbit operations during mission CDR</td>
<td>Resolved</td>
</tr>
</tbody>
</table>
Currently Completed

- PIC basics
  - Timers, interrupts, UART, analog-to-digital converter
- Command processing logic
- ADCS Peripheral Controller
  - Collect and send MRM data
- Flight Computer flash memory interface
- Software I2C and SPI libraries
- Power Board software
  - Periodic collection from various IC’s
    - Analog, SPI, I2C ICs
- Watchdog
  - Heartbeat and resetting functionality
- Flight Computer
  - Scheduler
  - File System
  - Uplink Authentication
  - Interfaces with 2 Power boards, IDPU, ACB, Radio (basic with He-100)
  - SideA/SideB – script to make the process easier
- Flight Computer Software
  - Early Orbit Operation functions
  - In-flight reprogramming (built upon sideA/sideB)
- Radio (He-82)
  - Test commands received via uplink
  - Implement downlink; error correction (FEC)
- Watchdog Software
  - Adding UART interface
  - File system
  - Monitoring TMP275s for resets
- ADCS Software
  - Control Law Algorithm
  - Adding more features (file system, hardware I2C, etc)
- Deliverables and requirements
- Integrating and testing the entire system
- Optimizing/Improving existing features
**Flight Computer Software**
- Early Orbit Operation functions
- In-flight reprogramming (built upon sideA/sideB)

**Radio (He-82)**
- Test commands received via uplink
- Implement downlink; error correction (FEC)

**Watchdog Software**
- Adding UART interface
- File system
- Monitoring TMP275s for resets

**ADCS Software**
- Control Law Algorithm
- Adding more features (file system, hardware I2C, etc)

**Deliverables and requirements**
- Integrating and testing the entire system
- Optimizing/Improving existing features
BACKUP SLIDES AHEAD

Mission CDR
Command and Data Handling

BACK UP SLIDES
<table>
<thead>
<tr>
<th>REQ ID</th>
<th>Requirement</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&amp;DH-01</td>
<td>C&amp;DH shall be capable of storing housekeeping data</td>
<td>File system design. V&amp;V tests #4 &amp; #5, July/August 2015</td>
</tr>
<tr>
<td>C&amp;DH-03</td>
<td>C&amp;DH shall be capable of processing commands from the ground station</td>
<td>Uplink processing design. V&amp;V tests #6 &amp; #15 August/October 2015</td>
</tr>
<tr>
<td>C&amp;DH-04</td>
<td>C&amp;DH shall be capable of disabling the spacecraft transmitter on command from the ground station</td>
<td>TODO</td>
</tr>
<tr>
<td>C&amp;DH-06</td>
<td>C&amp;DH shall be capable of interfacing with the communications system to transmit housekeeping and science data</td>
<td>In progress, V&amp;V #15 October 2015</td>
</tr>
<tr>
<td>C&amp;DH-29</td>
<td>The C&amp;DH subsystem shall be designed to operate at a dose of 5 krad/yr or greater</td>
<td>Hardware design. Upset mitigation/reset plan.</td>
</tr>
<tr>
<td>C&amp;DH-30</td>
<td>Early orbit operations shall be autonomous</td>
<td>TODO</td>
</tr>
<tr>
<td>Requirement ID</td>
<td>Requirement Description</td>
<td>Implementation Details</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>C&amp;DH-07</td>
<td>C&amp;DH shall be capable of putting the spacecraft in safe mode</td>
<td>Implemented by design</td>
</tr>
<tr>
<td>C&amp;DH-09</td>
<td>C&amp;DH shall be capable of resetting itself in the event of an error</td>
<td>Implemented with watchdog timer Test with Watchdog heartbeat V&amp;V #14 Sept 2015</td>
</tr>
<tr>
<td>C&amp;DH-10</td>
<td>C&amp;DH shall be capable of executing scheduled events sent by ground station</td>
<td>Implemented and tested with scheduler V&amp;V #1 June/July 2015</td>
</tr>
<tr>
<td>C&amp;DH-11</td>
<td>C&amp;DH shall be capable of jumping the clock on command from the ground station</td>
<td>TODO</td>
</tr>
<tr>
<td>C&amp;DH-12</td>
<td>Engineering magnetometer telemetry shall be collected at 32 samples per second or faster to implement the ADCS control law algorithm as specified by the ADCS subsystem.</td>
<td>Implemented and tested V&amp;V #18 Oct 2015</td>
</tr>
<tr>
<td>C&amp;DH-13</td>
<td>All software shall support a sleep-wake cycle in order to conserve power.</td>
<td>Implemented by design</td>
</tr>
<tr>
<td>C&amp;DH-14</td>
<td>All software shall respond to a command with a latency of less than 100 ms.</td>
<td>TODO</td>
</tr>
</tbody>
</table>
**REQUIREMENTS**

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&amp;DH-15</td>
<td>All software must be tolerant to single event upsets. (SEUs)</td>
<td>Implemented through memory mirroring, ECC &amp; CRC, reset &amp; watchdog plan. Tested indirectly. Direct tests TODO.</td>
</tr>
<tr>
<td>C&amp;DH-16</td>
<td>The C&amp;DH subsystem shall not exceed the mass allocated by Systems</td>
<td>Boards were weighed on receipt from Aerospace.</td>
</tr>
<tr>
<td>C&amp;DH-17</td>
<td>The C&amp;DH subsystem shall not exceed the power allocated in the ELFIN system power budget</td>
<td>TODO</td>
</tr>
<tr>
<td>C&amp;DH-18</td>
<td>Throughput speed to the Communications subsystem shall meet or exceed 38.4k baud.</td>
<td>By design.</td>
</tr>
<tr>
<td>C&amp;DH-19</td>
<td>Real time calendar clock (RTCC) data shall be provided to the IDPU as the first command after the IDPU is powered up. This command shall arrive with 200 ms latency and +/− 100 ms absolute variation (jitter) from the 1 second rollover on the avionics RTCC.</td>
<td>TODO</td>
</tr>
<tr>
<td>C&amp;DH-20</td>
<td>The C&amp;DH subsystem shall not transmit any RF transmissions for the first 45 minutes after separation from the launch vehicle</td>
<td>TODO</td>
</tr>
<tr>
<td>C&amp;DH-21</td>
<td>The C&amp;DH subsystem shall not actuate any deployables during the first 30 minutes after separation from the launch vehicle</td>
<td>TODO</td>
</tr>
</tbody>
</table>
### REQUIREMENTS

<table>
<thead>
<tr>
<th>C&amp;DH-22</th>
<th>C&amp;DH shall be capable of resetting and switching program sides if the Avionics does not receive a command from the ground in TBD amount of time</th>
<th>TODO</th>
</tr>
</thead>
<tbody>
<tr>
<td>C&amp;DH-23</td>
<td>The C&amp;DH subsystem shall not exceed the data allocated in the ELFIN system data budget</td>
<td>Packet sizes as implemented compared to data budget.</td>
</tr>
<tr>
<td>C&amp;DH-24</td>
<td>All EPS components shall be kept within their operational thermal limits during all modes of operation</td>
<td>TODO, collaboration with thermal team.</td>
</tr>
<tr>
<td>C&amp;DH-25</td>
<td>The C&amp;DH subsystem shall be capable of operating with a single power board</td>
<td>Hardware design</td>
</tr>
<tr>
<td>C&amp;DH-27</td>
<td>The C&amp;DH subsystem shall be capable of beaconing</td>
<td>TODO</td>
</tr>
<tr>
<td>C&amp;DH-28</td>
<td>The C&amp;DH subsystem shall be designed for a 6 month lifetime</td>
<td>Hardware design</td>
</tr>
</tbody>
</table>
- Apache Subversion (SVN)
- Tags generated for stable releases of flight software
Feature on FCPCB:

File System
Periodic data stored in file system to allow easy retrieval

Each packet data type gets its own circular buffer in the file system

Each packet has a 6 byte timestamp

Packets are inserted in chronological order

If no more room is left in the current page, the packet is inserted into the next available page of the circular buffer
A request asks for all the packets between a certain time period of a certain type.

first searches for the newest packet that fits in the date range using a linear search.

then searches for all the older packet that fits in the date range using a linear search.

The time it takes to access a page is hugely more than the time it takes to find a packet in a page, meaning that the linear search speed is insignificant in finding the range.
• Code was implemented and tested on PC before being ported to the PIC
  ▪ The flash memory interface has to be integrated with the PC version of the code
• The Flight Computer requested information from a power board and that was stored in the file system. Then a command was sent to retrieve this packet
• More packets from different sources (two power boards and Flight Computer telemetry) were stored and requested
• Issues
  ▪ Previously, a page was taken directly from the flash and stored in a runtime memory buffer. However, due to lack of space on the PIC, a 1024 byte buffer could not be used to do this operation. As a result, a quarter can only be used at a time
Feature on FCPCB:

Uplink Authentication
- Prevent unauthorized access to our spacecraft using SHA-1 hash, a salt, and a secret key
  - Hash: 20 bytes
    - Verified on spacecraft from command received via radio
  - Salt/Counter – 2 bytes
    - Ground version
      - Incremented when command sent
    - Flight Computer version
      - Increment when valid command received
  - Secret Key – 6 bytes
If salts no longer match, the Flight Computer will not execute the incoming command due to failing to match its calculated hash to the hash of the command.

Mechanisms to address this:

- File system packet containing housekeeping data for the Flight Computer will also contain the current salt
  - Can be used to verify salt during a pass
- Nacks
  - Nack command will contain Flight Computer’s salt in the payload
First, commands were tested without the salt incremented
- The Flight Computer wouldn’t accept the command if the hash was incorrect

Incrementing salts were then added
- Salts were incremented on both the GSE and Flight Computer side
- The Flight Computer was able to Nack back the correct salt value due to incorrect hash

Issues
- Calculating the hash to compare to the incoming hash has slowed down the command processing process – this caused an overflow in the command circular buffer as several big commands (with the hash) sent at once were not being processed and removed fast enough
  - Will need to test throughput
Feature on FCPCB:

Side A / Side B
- Two copies of the same code to address program corruption

- **Interrupt vectors** - program jumps to this location when an interrupt occurs, and is then directed to the Interrupt Indirection section.

- **Side A** – The first copy of the program in memory; includes the interrupt handler functions associated with this copy.

- **Side B** – The second copy of the program in memory; includes the interrupt handler functions associated with this copy.

- **Interrupt Indirection** – Determines which side’s interrupt handler to go to after an interrupt has occurred

- **Side Determination** – Executes initially upon startup or (potentially) directly after a reset; determines which side to run the program of based off EEPROM (side and toggle bits) and code toggle (from Watchdog)
The general process to create two copies of the same program in memory has been automated using a Perl script and thoroughly tested.

The Perl script takes the original program as input and generates a HEX file that upon compilation results in a double sided version of the same program.

The resulting program successfully switches sides upon a reset.

The resulting program executes correctly on either side.
Feature on FCPCB:

Communications: Radio Interface
• Will be Helium-82 on board, currently testing with Helium-100

• Sending commands to the radio via UART
  ▪ Command Header: 8 bytes
    ▪ Includes a checksum of the header
  ▪ Payload: extra info needed for the command
    ▪ Includes checksum of entire message

• Radio feedback
  ▪ Radio responds to command with a message of the same format
    ▪ Command Header:
      ▪ If no payload size, sends 0x0A if acknowledged/0xFF if not acknowledged
    ▪ Payload: requested information if applicable
MD5 checksum needed to write configuration into radio’s flash
  ▪ Calculate this over the radio configuration structure

Flight Computer version of code
  ▪ Calculations for MD5 checksum use too much program memory
  ▪ Calculate MD5 on GSE first to make it easier on the PIC
    ▪ Command to configure the radio sent from GSE includes MD5 checksum in its payload
Ability to parse messages from the radio
  - Iterates through packet header to extract Flight Computer commands within radio packet’s payload
  - Built upon existing command parser

Testing
  - Configured TNC (Kantronics KPC-9162+)
  - Connected PC → TNC → IC910H → Helium-100 → Flight Computer
    - Sent a simple command from PC to modify Flight Computer
      - Command packeted by TNC, sent through radio, parsed by Flight Computer
  - These tests only covered uplink, not downlink

Plan for downlink implementation
  - Raw Transfer Mode
  - Reed Solomon code with 4-frame interleaver
  - Packet format is defined but more development needed
Feature on FCPCB:

Early Orbit Operations
- Details what the satellite does before normal operations take place
  - Will be described and discussed in more detail during the Operations team’s presentation

- Timers based off of RTCC in conjunction with Flight Computer’s Scheduler
  - Scheduler provides flexibility and simplicity for implementation
  - Entries missed can be mitigated with redundant entries and reschedule commands/entries