Reliable and Self-Configurable Flight Software Architecture for CubeSats

Te-Chuan Huang, Cheng-Ting Wu, and Jyh-Ching Juang

Department of Electrical Engineering
National Cheng Kung University, Tainan, Taiwan

Photo by NASA (July, 2014)
Outline

- Introduction
- PHOENIX Nanosatellite
- Flight Software Design
- Resilient Software Design
- Testing and Verification
- Conclusion
Outline

- Introduction
  - PHOENIX Nanosatellite
  - Flight Software Design
  - Resilient Software Design
  - Testing and Verification
  - Conclusion
Introduction

- **CubeSat**
  - Standardized nanosatellite
  - Inexpensive cost and shorter development time
  - Commercial Off-The-Shelf (COTS) products available

- **On Board Data Handling (OBDH) Subsystem**
  - Hardware - On Board Computer (OBC)
  - Software - Flight Software (FSW)
Introduction

• Flight Software (FSW)
  • Limited resource
  • Unpredictable space environment
  • Responsible for many task
  • Hard to modify after launch

• Several techniques of design and implementation of reliable FSW are presented.
Outline

- Introduction
- **PHOENIX Nanosatellite**
- Flight Software Design
- Resilient Software Design
- Testing and Verification
- Conclusion
PHOENIX is one of the CubeSat in QB50 project.
PHOENIX Nanosatellite

- QB50
  - EU’s FP7 project managed by VKI
  - Carrying out atmospheric research within the lower thermosphere

- PHOENIX
  - Subsystem: EPS, OBDH, COM, ADCS, ANTS & GPSR.
  - Science payload: INMS and SEUV

- OBDH
  - GomSpace NanoMind A712D
  - FreeRTOS operating system
Outline

- Introduction
- PHOENIX Nanosatellite
- **Flight Software Design**
  - Resilient Software Design
  - Testing and Verification
- Conclusion
Flight software is embedded in the On Board Data Handling (OBDH) subsystem in CubeSat and serves as a brain of the satellite.
Flight Software Design
- Requirements -

• By analyzing the requirements, a system view from top to down helps the development of software.
  • Functional Requirements
  • Interface Requirements
  • Operational Requirements
  • Software Reliability Requirements
  • Software Safety Requirements
Flight Software Design - Architecture -

- Three layered architectures

<table>
<thead>
<tr>
<th>Task Layer</th>
<th>Function Layer</th>
<th>Library Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode Task</td>
<td>File Read/Write</td>
<td>liba712</td>
</tr>
<tr>
<td>Initial Task</td>
<td>HK Collection</td>
<td>libcsp</td>
</tr>
<tr>
<td>INMS_E Task</td>
<td>TC TM Handle</td>
<td>libutil</td>
</tr>
<tr>
<td>Schedule Task</td>
<td>Subsystem Control</td>
<td>libftp</td>
</tr>
<tr>
<td>SEUV_C Task</td>
<td>Report Generate</td>
<td>libgosh</td>
</tr>
<tr>
<td>Anom Task</td>
<td>Parameter Set</td>
<td>liblog</td>
</tr>
<tr>
<td>Format Task</td>
<td></td>
<td>libparam</td>
</tr>
<tr>
<td>GPS Task</td>
<td></td>
<td>libstorage</td>
</tr>
<tr>
<td>WOD Task</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beacon Task</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HK Task</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INMS_T Task</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INMS_S Task</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INMS_R Task</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Flight Software Design - Operation Mode -

- Mode Transition
  - Initial Mode
  - Early Orbit Phase (EOP) Mode
  - Stabilize Mode
  - Nominal Mode
  - Safe Mode
  - Recovery Mode

AUTO: flight software trigger
COM: telecommand trigger
Anomaly: anomaly trigger
Outline

- Introduction
- PHOENIX Nanosatellite
- Flight Software Design
- Resilient Software Design
- Testing and Verification
- Conclusion
Resilient Software Design

- Several methods proposed to increase the reliability of the FSW
  - Modular Programming
  - Redundancy Memory Design
  - Anomaly Handling
  - Self-Configurable Architecture
  - On Board Scheduling Management
  - In Orbit Software Update
Resilient Software Design
- Modular Programming -

• Features
  • Divide an application into several modules
  • Independent, interchangeable modules
  • Plug and play idea
  • Increase the flexibility, maintainability, and reusability of FSW
  • Common functions can be used in future satellite programs
Resilient Software Design
- Redundancy Memory -

• SD card partition – /sd0 & /sd1

• Crippled mode
  • Backup solution
  • Use flash memory instead
  • Limited storage place

Crippled Mode

<table>
<thead>
<tr>
<th>type_DATA.dat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  2  3  4</td>
</tr>
<tr>
<td>5  6  7  8</td>
</tr>
<tr>
<td>9 10  ...   N-1</td>
</tr>
<tr>
<td>...  ...  N</td>
</tr>
</tbody>
</table>

Nominal Mode

<table>
<thead>
<tr>
<th>type_DATA/</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000.dat  00000030.dat</td>
</tr>
<tr>
<td>00000060.dat  00000100.dat</td>
</tr>
<tr>
<td>0000200.dat   0000030.dat</td>
</tr>
</tbody>
</table>
Resilient Software Design - Self Configurable Architecture -

- The behavior of the FSW is influenced by some system parameters.
- Tune/adjust system parameters depending on the situation.

**System Parameters**
- Safe Mode Threshold
- Over Current Threshold
- Temperature Range Threshold
- Initial OBC time

**Monitor Task**
- Margin Library
- Value Pool
- Update Algorithm
- Analyzing Mechanism

**CubeSat Performance**

Parameter control the performance of CubeSat

Task monitors the parameter and update itself

Give feedback to the monitor task
Some unpredictable situations occur in space, it may cause the fatal error in FSW.

In case that some major issues happened in flight software, a backup solution is proposed by uploading new firmware on OBC.

All the step must be done by telecommand from GS.
Resilient Software Design
- In Orbit Software Update -

NanoMind image (660.8 KB) → LZO algorithm → Compressed NanoMind image (377.1 KB)

Split Program on Ground

Part1 15000 B
Part2 15000 B
Part3 15000 B
Part4 15000 B
Part25 15000 B
Part26 2055 B

OBC SD card

RF signal 100%
Resilient Software Design
- In Orbit Software Update -

New firmware
• Results:
  • each part will be uploaded within 180 seconds
  • More than 20 contacts are needed.
  • Successfully update the flight software all by GS
Outline

- Introduction
- PHOENIX Nanosatellite
- Flight Software Design
- Resilient Software Design
- Testing and Verification
- Conclusion
Testing and Verification

- Several tools are used to conduct the testing campaign and verification process.
  - Discover errors or defects in FSW.
  - Verify that the design meets the requirements.
  - Evaluate the performance of FSW.
Testing and Verification
- Test Configuration Setup -

• Software Debug Interface
  • Self defined command
  • GOSH interface

• Tools
  • EGSE – ISIS Generic Interface System
  • I²C protocol analyzer
  • STM32F429 Discovery Board
Testing and Verification
- Test Campaign -

- Reference Functional Test
  - Basic Functional Test
- Environmental Test
  - Data Recording
- End to End Communication Test
  - Uplink large file, downlink large file
- 32 Hour Test
  - Continues operation, simulate contacting, mode transition
- Anomaly Handling
  - Handle anomaly during the mission
- Mission Verification
  - Focus on operation of each mode
- Telecommand Test
  - All telecommands (169/169) have been implemented and tested.
Outline

- Introduction
- PHOENIX Nanosatellite
- Flight Software Design
- Resilient Software Design
- Testing and Verification
- Conclusion
Conclusion

• Flight Software
  • A comprehensive software architecture is designed and implemented.
  • A few methods for increasing the reliability of FSW are proposed.

• Testing and Verification
  • Several testing campaigns are conducted and passed successfully.
  • Some anomaly are detected, and update the related action in FSW.
Thank You for attention!