Implementation of a Real-Time Operating System on a Small Satellite Platform

Caitlyn Cooke
April 21, 2012
Colorado Space Grant Consortium
Undergraduate Research Symposium
Introduction

- Overview of the Drag and Atmospheric Neutral Density Explorer and its real-time command and data handling architecture
- Benefits of a real-time operating system for our platform
- Hurtles that were presented during final integration
DANDE Overview

Mission Statement
Explore the spatial and temporal variability of the neutral thermosphere at altitudes of 350 - 200 km, and investigate how wind and density variability translate to drag forces on satellites.

\[ F_D = \frac{1}{2} \rho \cdot C_D \cdot A \cdot (V + V_w)^2 = M \cdot a \]

- Spherical payload rotating at 10rpm
  - Spin rate and alignment necessary for successful science
- Eight electrical subsystems
  - EPS, COM, ADC, THM, ACC, NMS, SEP, and CDH
- Ten operational modes to achieve mission success
  - Separation
  - Attitude Control
  - Science
  - Standby
What Constitutes a Real Time Operating System?

- An event driven system that acts on strict time constraints and operational deadline
- Advanced scheduling mechanisms to orchestrate tasks for increased multi-tasking abilities
- Valued for predictability of performance and response time to tasks
- DANDE’s kernel is event driven, preemptive priority scheduling
Common Alternatives for Small Satellites

• Simple control loop
  – One serial loop with a distinct task, or set of subtasks (i.e. ACC and THM data collection)

• Interrupt based control loop
  – Main serial loop that is not sensitive to delays
  – Event triggered interrupt loop for priority tasking (i.e. EPS control firmware)
CDH Hardware Overview

- Atmel NGW100
  - Linux operating system
  - AVR32 DSP
  - 8Mb flash memory
  - 32Mb SDRAM
  - 512 Mb SD card

- Linux 2.6.27.6.atmel.1
  - First version of microcontroller compatible kernels
  - Preemptive kernel structure
  - Improved scheduling algorithm
  - Allowed for faster response to priority events
CDH Software Platform

- Linux 2.6.27.6 Atmel Kernel on a NGW-100 AVR32 DSP
- Linux System Commands
- 50+ Busy Box System Commands
- Bus Message Queue Structure and i2C Bus Communication
- 13 User Processes that Command 10 Operational Modes Designed to Meet Mission Criteria
User Processes

- **Bus Messenger/ Message Queue Handler**
  - Tasking scheduling
  - Information requests
  - Message traffic handling

- **Process and Subsystem Watchdogs**
  - Monitors system configuration
  - Internally corrects for user defined error cases

- **Mode Management**
  - Orchestrates mode changes and handles user scheduling or modes

- **Ground Communications Handler**
  - Provides interface between InControl and space craft
User Processes

- **Data Collection**
  - Engineering diagnostics, battery temperature, RTC broadcaster, NMS data collection and commanding

- **Data Processing**
  - Data compression, min/max/average

- **Attitude Control**
  - Torque rod scheduling form ground for space craft alignment
  - Closed loop spin rate and damping control system
Contribution to Mission Success

• Quick and predictable response to a large number of tasks simultaneously
• Allows for greater complexity of process tasks
  – Control of the very sensitive NMS with sophisticated error case handling
  – Closed loop attitude control system
  – Increased data downlink capability with space craft data processing techniques
• Greater flexibility of system modifications
• On flight system configurability for repairing bugs or altering mission objectives
Error Handling and Reporting System Implementation

- Initial DITL testing showed DANDE could not stay operationally stable for more than 8 hours
- Error reporting and logging mechanisms not verbose enough to diagnose issues
- Extensive logging implemented on multiple layers:
  - System diagnostics (i.e. CPU, memory, load average)
  - Message Queue traffic monitor
  - User process diagnostics
  - Automated system functional test
  - EPS and battery status monitors
  - Data collection frequency timing
Memory Constraints

- Identified memory leaks in multiple system processes:
  - Data collection processes, consuming approx 1Kb/sec:
  - Directory listing Unix command, instantly crashed system upon collection of excessive quantity of data files:
    - Data structure that store file names within the listing command consumes excessive memory
    - The accumulation of 3000 individual raw data files consumed approx. 20Mb of memory

- Actions taken to solve stability issues:
  - All allocated memory in sensor libraries and data collectors released when no longer used
  - New version of the listing command was written to use a static amount of memory, no longer dependent of file numbers
  - Raw data caching implemented to reduce the amount of file accumulation

- Verification of successful fix:
  - All system processes monitored during the next DITL and checked for further leaks
  - New listing command ran with excessive files presents and monitored for memory consumption
Message Concurrency

- EPS was discovered to be turning on/off subsystems at random

Testing Layers of Abstraction:
- CDH message packaging process activity logged. Improper messages were found at this layer.
- Message queue wrapping process activity logged. Provided visibility of all message traffic throughout CDH
- Individual process activity logged. Provided visibility on original generation of messages

Logging Conclusions and Fixes:
- Improper message generation tracked back to command line bus messaging process, utilizing bash scripting methods
- Command line messaging process contained concurrency, causing message traffic confusion
- Random number generator implemented to ensure message instance uniqueness
Load Average Instability

- Initial nominal load average observed at 5-7
  - Unhealthy load caused operational errors such as data product inaccuracy, message logging inaccuracy, and latency in command execution
- Identified multiple system processes that consumed excessive resources
  - Un-throttled infinite loops in multiple system process caused excessive scheduling of tasks and CPU utilization
  - Use of the “tar” Unix command utilizing excessive system memory
  - Data logging and message sending rates excessively high in data collectors
- Actions taken to increase system performance
  - Sleep commands were added abundantly through system to decrease tasks
  - “tar” command replaced with “zip” command, using less resources
  - Data rates reduced significantly, improving load average, decreasing message traffic, and improving data downlink rates
Conclusion

• Added complexity of a RTOS can lead to complexity of critical system error debugging in the integration phase
• Implementation of extensive logging mechanisms early in the design phase can help speed the integration process significantly
• Decision to use an RTOS requires detailed consideration of system design
• In DANDE’s case, Benefits have led to the creation of a more advanced space craft, capable of achieving mission success