Modular Infrastructure for Rapid Flight Software Development

Craig Pires
NASA Ames Research Center
Overview

• Background
• Flight Software Development Process
• Simulink Model Overview
• Integration with cFE
Background

- Small Spacecraft Investigation
  - Modular CommonBus Spacecraft
- Hover Test Vehicle (HTV) Development
- Next Step - Lunar Atmosphere and Dust Environment Experiment (LADEE)
  - Joint ARC/GSFC Mission
  - Lunar Orbiter, Launch 2012
Flight Software Infrastructure Development

- Model Based Approach for Application Unique Software
- Latest Developments
  - Mathworks Simulink/RTW Embedded Coder
  - Integration of GSFC ITOS GDS Tool
  - Integration of GSFC Core Flight Executive (cFE)
  - Demonstrated on HTV
Hover Test
Flight Software Development Process Overview
Model Based Development Approach
- Develop Models of FSW, Vehicle, and Environment in Simulink
- Automatically generate Software using RTW/EC.
- Integrate with hand-written and heritage software.
- Iterate while increasing fidelity of tests – Workstation Sim (WSIM), Processor-In-The-Loop (PIL), Hardware-in-the-Loop (HIL)

Iterate Early and Often
Automatic Code Generation

- Simulink supports two way trace-ability between models and generated code
- Code Easy to read, well commented
Simulink Model Overview
Simulink HTV Architecture

FSW Auto-Coded and integrated with CFE

Flight Software

Flight Bus (Data)

Flight Hardware

Environmental Link

Ground Station

Ground Data Station
Simulink FSW Model

Command Processing:
- Receives commands via CDH (TCP/IP or RS422).
- Compiled in script allows flexible sequencing.
- Processes and Sets Control Modes.

Vehicle Health Monitoring:
- Command Checking
- Sensor Limit Checking
- Hardware status

State Estimation:
- Receives sensor data.
- Low Pass Filters
- Auto generated Kalman Filter.

Telemetry:
Passes data to the CDH so that it can be transmitted via TCP/IP or RS422.

Prop Management:
- Fires thrusters based on commands and control mode.

GN&C:
- Guidance System sets desired angles based on position error.
- Guidance System maintains desired vertical velocity.
- Control System uses Bang-Bang approach to maintain desired angle.

Simulink Bus Creator
Simulink Flight Hardware Model

Sensor Models
- Analogs (Temperature, Pressure)
- LN200 IMU
- VIZ Camera System

Thruster dynamic forces and torques.

Mass and Inertia Characteristics of Vehicle
Simulink Environment Model

Command and Downlink Delays

6DOF Position and Rotational Propagation

External Forces on Vehicle (Tether, platform)

Gravitational Forces

Vehicle Initial Conditions
cFE Simulink Integration
cFE – Core Flight Executive

• Goddard Space Flight Center Developed
• Derived from Legacy Missions
• Flexible infrastructure for Space Flight Software

• Components:
  – Executive Services
  – Event Services
  – Time Services
  – Table Services
  – Software Bus Services
cFE Simulink Development Goals

• Utilize cFE with no changes
• Automate process during Code Generation.
• Subsystem Blocks generate to cFE Applications that run at desired rates
• Simulink Apps/Blocks Communicate via cFE Software Bus
Layered Architecture Approach

Simulink Generated Mission Unique Application Layer

Generic Services Layer (GSFC cFS) & Hand Code

System Support Layer (GSFC cFE)

OS Services Layer (VxWorks OS, GSFC OS Abstraction Layer)

Physical (Hardware) Layer
cFE Simulink Key Ideas

• Modular Tasks (vs. Monolythic)
  – Pros:
    • More Flexible
    • Simplifies Task Replacement
    • Easier Debugging – can look at messages between tasks
  – Cons:
    • Harder to implement
    • More overhead due to more tasks and messages

• Mathworks Template (TLC) File
  – Executed during Code Generation Process
  – Allows customization of created code
  – Leveraged to autocode cFE Apps from Simulink
cFE Simulink Implementation

• Simulink Bus translates to cFE Message
• RTW/EC generates Task Description
• Master Timer Generates “Tick” to Schedule Apps and generate Output Messages
• Receive Structure Msgs update local App Input Values
• Apps also Respond to Other Command and Housekeeping Messages
cFE Simulink Autocode Process

RTW/EC

Sequencer
VHM
State Est.
Thermal
Payload
Prop Pyro
Power
GN&C
Telemetry

Sequencer.c + IF.h
VHM.c + IF.h
State_Est.c + IF.h
Thermal.c + IF.h
Payload.c + IF.h
Prop_Pyro.c + IF.h
Power.c + IF.h
GNC.c + IF.h
Telemetry.c + IF.h

Compile & Link

FSW

HC_Module.c

CFE INTERFACE.c
Drivers.c

ITOS DBX
Simulink Bus becomes cFE Message

'simulink_bus', ...
sprintf("', ...'
    '{ins_msg}', 3, 'int16', -1, 'real', 'Sample'}); ...
    '{ins_mode}', 1, 'int16', -1, 'real', 'Sample'}); ...
    '{ins_data}', 1, 'int16', -1, 'real', 'Sample'}); ...
    '{ins_counts}', 3, 'int16', -1, 'real', 'Sample'}); ...
    '{ins_checksum}', 1, 'int16', -1, 'real', 'Sample'}); ...
cFE Simulink Message Flow

- 100 Hz Tick
  - Sequencer
  - State Est.
  - Prop Pyro
  - GN&C

- 100 Hz Tick
  - Sequencer
  - State Est.
  - Prop Pyro
  - GN&C

- 100 Hz Tick
  - Sequencer
  - State Est.
  - Prop Pyro
  - GN&C

Messages
Struct App_Inputs In
Struct App_Outputs Out
App_Init() {
    Initialize_App_Inputs()
    Subscribe_SB_Msgs(Tick, AppMsgs,...)
    Simulink_Init(In, Out)
}
App_Main(){
    App_Init()
    while(1) {
        sb_receive_msg(msg, timeout)
        if (msg == tick) {
            Simulink_Step(dt, In, Out)
            sb_send_msg(Out) /* app update */
        } else {
            If (msg == app_update) /* Process other App Msgs */
            App_Update_Inputs(msg, Out)
            else Process_Msg(msg) /* HK, Cmds, etc... */
        }
    }
}
New Efforts

- 3DOF Simulator
- Command Queueing
- Parameter Tables
- Command & Telemetry Dictionary - XTCE
- Snapshot/Snapshot Recall
- Latency Reduction
  - Output Message triggers “Step” of Next Module
  - Retains Modularity
Summary

• NASA Ames developing infrastructure for rapid flight software development
• Model based process leverages Mathworks Simulink, RTW-EC
• Developed modular approach to integrate auto-generated code with GSFC’s cFE.
• Successfully demonstrated on HTV
• Being Utilized on NASA’s LADEE mission
Backup
cFE IMU App Loop

IMU_Main(){
    while(1) {
        struct imu_input_str imu_in
        read_msg_que(imu_in, timeout) /* VxWorks Msg Que */
        sb_send_msg(imu_msg)
        Send_tick()
    }
}

Cnt = 0;
Send_tick() {
    sb_send_msg(400HZ_Tick)    /* Do we need 400HZ Tick or key off of IMU Data? */
    if ((Cnt % 2) == 0) sb_send_msg(200HZ_Tick)
    if ((Cnt % 4) == 0) sb_send_msg(100HZ_Tick)
    if ((Cnt % 40) == 0) sb_send_msg(10HZ_Tick)
    if ((Cnt % 400) == 0) sb_send_msg(1HZ_Tick)

    Cnt++;
}

/* Note: Other Apps same as IMU without the Send_tick() */
• Simulink/SystemBuild Only (No Autocode)
• Early in development process
• Algorithm Development
• Requirements Analysis
Processor-in-the-Loop Simulation

- Models autocode and running on RT processors
- Inexpensive “flight-like” processor
- Tests autocode process & integration with C&DH software
- Integration with Telemetry Software allows early development/testing of downlink
- Can be used for initial code size and resource utilization analysis
• Flight code runs on Flight Avionics EDU
• Provides testing of FSW with Avionics I/O
• Definitive answers on resource utilization
• Highest fidelity simulations for verification/validation
Motivation for Moving to Simulink

- Industry appears to be moving that direction.
- Mathworks Extensive support network.
- Mathworks tools for Requirements management, Documentation, and V&V.
- Bus concept makes model management easier.
- Monolithic SystemBuild models not conducive to Reuse and V&V.