Operating System Abstraction Layer (OSAL)

Flight Software Workshop
Nicholas J Yanchik
November 6, 2007
Agenda

- What is the OSAL?
- Where does it fit in our current FSW architecture?
- How does it work?
- Directory structure
- What functionality does the OSAL provide?
- OSAL releases
- Metrics
- Open Source Software
- Future Plans
What is the Operating System Abstraction Layer?
- A small layer of software that allows programs to run on many different operating systems and hardware platforms
- Independent of the underlying OS & hardware
- Self-contained

Why do we want it?
- Removes dependencies from any one operating system
- Promotes portable, reusable flight software
- Core FSW can be built for multiple processors and operating systems
- Example: different missions require different hardware & operating system

What does it do?
- Allows developers to write and maintain one version of code
- Allows for easy reuse across different missions with different hardware
- Bonus: Allows for desktop development of flight software; reduces impact of potential hardware delays
Where Does It Fit in Our Current Flight Software Architecture?

- Application 1
- Application 2
- ... Application N
- Core Flight Executive (cFE)
- OS Abstraction Layer
- Real Time Operating System
- Drivers
- Board Support Package
- Flight Computer Hardware
How Does It Work?

- Implemented by make files
- Compiles in only the files needed for a specific OS/architecture
Directory Structure

- osapi
  - os
    - OS type: linux, rtems, vxworks, os x
      - osapi.c
      - osfilesys.c
  - arch
    - platform: coldfire, ppc, x86
      - board: mac, mcp750
        - os: rtems, vxworks
          - bsp, exe
  - prolog.mk
Functionality - Standard API's

- Abstractioned ID's and information
  - All entities named

- Task API
  - Create, Delete, Exit, Delay, Set Priority, Get Info, Register, Get ID, Get ID by Name

- Queue API
  - Create, Delete, Get (w/ timeout), Put, Get ID, Get ID by Name, Get Info,

- Semaphore API
  - Binary Semaphores
  - Counting Semaphores
  - Mutexes
  - Create, Delete, Take, Give, Get Info, Timed Wait, Get ID by Name

- Misc API
  - Millisecs to System Ticks, Ticks to MicroSecs, Get Time, Interrupt Disable/Enable and Lock/Unlock, Printing utility
Functionality (2) - File System API's

- Abstracted FS
  - The file system has the same interface to the user no matter the underlying OS

- File System API
  - Make FS, Remove FS, Init FS, Mount, Unmount, Get Physical Device Name

- File API
  - Create, Remove, Open, Close, Read, Write, Lseek, Rename, Copy, Move Files
  - Make, Remove, Open, Close, Read Files
  - Get Info on File Descriptors
  - Send Shell Command to a file
OSAL Releases

- **Version 1.0 (Released August 2004)**
  - Developed by Alan Cudmore / code 582
  - Currently being used on SDO
  - Open source via a Flight Software Branch Technology Initiative
  - Capabilities: Creation of OS resources, Interrupt and Exception API, Hardware and memory API

- **Version 2.0 (Released July 2005)**
  - Used with the cFE for LRO (and previously HRV)
  - Additional Capabilities: dynamic object creation, deletion of resources, file system layer, networking functions, general API improvements with parameters and error codes

- **Version 2.10 (Release Before 2008)**
  - Currently being used by the cFE, LRO mission, SDO mission, ESA EDROOM, DISILCAS.
  - Additional Capabilities: Counting semaphores
  - Enhancements made to almost all aspects of the OSAL, including file system, task, queue, and semaphore code
• Executable Lines of code: 8168

• Average Lines per BSP: 1500
  - Number of distinct BSP's: 7
  - VxWorks on MCP750 skews results

• Number of OS's supported: 4
  - VxWorks
  - RTEMS
  - Linux
  - OS X

• Number of boards supported: 6
  - M5282lite  – Intel Mac
  - m5235bcc  – PPC Mac
  - Mcp750    – x86 Desktops
Open Source Software

- **Version 2.0**

- **NASA Open Source License (2004)**
  - Allows users to redistribute code, but must include source code
  - Allows additions to software, but additions must be the work of the author
  - Requests users to register software
  - Requests users to inform us of modifications
Future Plans

- Continue development of threaded model

- Support Current Customers

- Develop Version 3.0
  - Conversion from Thread Model to Process Model
  - Shared Memory API