National Aeronautics and Space Administration

PLATFORM LAYER UPDATES FOR THE CAELUM (7.0) RELEASE OF THE CORE FLIGHT SYSTEM

2021 FLIGHT SOFTWARE WORKSHOP

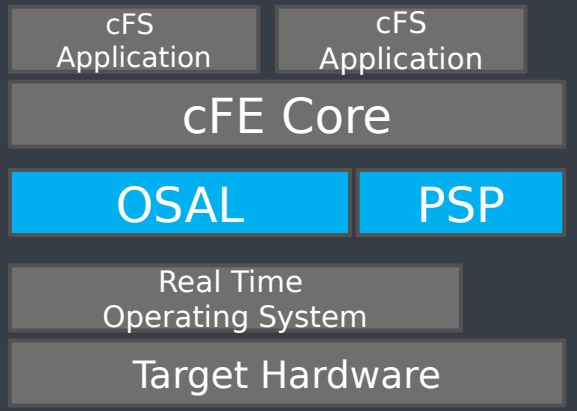
ALAN CUDMORE
NASA/GODDARD SPACE FLIGHT CENTER
CODE 582

AGENDA

- INTRODUCTION
- OPERATING SYSTEM ABSTRACTION LAYER (OSAL)
- PLATFORM SUPPORT PACKAGE (PSP)
- SYSTEM STARTUP
- EXCEPTION HANDLING
- CMAKE FILES
- PORTING TIPS
- SUMMARY

INTRODUCTION

- THE CFS ***PLATFORM LAYER*** *CONSISTS OF THE OPERATING SYSTEM ABSTRACTION LAYER (**OSAL**) AND THE PLATFORM SUPPORT PACKAGE (**PSP**)*
- THE **OSAL** PROVIDES A PORTABLE APPLICATION PROGRAMMING INTERFACE (API) TO THE UNDERLYING OPERATING SYSTEM SERVICES
- THE **PSP** PROVIDES GLUE LOGIC AND STARTUP CODE NECESSARY TO MAKE THE CFS WORK ON A SPECIFIC COMBINATION OF AN OPERATING SYSTEM AND PROCESSOR CARD
- TOGETHER THE OSAL AND PSP ALLOW THE CFS TO RUN WITHOUT MODIFICATIONS ON MULTIPLE TARGETS, AND MAKE IT EASIER TO



OS ABSTRACTION LAYER (OSAL) - FEATURES

- THE OSAL PROVIDES
  - A PORTABLE OPERATING SYSTEM API FOR REAL TIME OPERATING SYSTEM FEATURES
    - TASKS, QUEUES, SEMAPHORES, MUTEXES, FILES, TIMERS, AND NETWORK SOCKETS, ETC.
  - SUPPORT FOR THE FOLLOWING OPERATING SYSTEMS
    - **LINUX** – BOTH 32-BIT AND 64-BIT, MULTIPLE ARCHITECTURES
    - **RTEMS** – VERSION 4.11 AND 5.1
    - **VXWORKS** – VERSION 6.X AND 7.0
  - STARTUP CODE FOR ABSTRACTED APPLICATIONS
    - OSAL BOARD SUPPORT PACKAGE (BSP)
      - CURRENT OSAL BSPS INCLUDE
        - GENERIC LINUX
        - GENERIC VXWORKS
        - PC RTEMS
    - A PORTABLE APPLICATION ENTRY POINT (EXAMPLE: ”MAIN” ON LINUX, “INIT” ON RTEMS)

OSAL - LAYERED ARCHITECTURE

- STARTING WITH VERSION 5.1 (INCLUDED WITH CFS BOOTES), THE OSAL PROVIDES A LAYERED ARCHITECTURE
  - MINIMIZES DUPLICATE CODE
  - ASSURES CONSISTENT IMPLEMENTATION FOR EACH PORT (ERROR CHECKING, LOCKS)
  - MAKES PORTING OSAL TO A NEW OPERATING SYSTEM (OS) EASIER
  - SLOC FOR RTEMS IMPLEMENTATION:
    - OSAL 4.2.1A: 5577
    - OSAL 5.1 (MAIN BRANCH): 3196
- THE OSAL API IMPLEMENTATION CONSISTS OF
  - SHARED TOP LEVEL API SOURCE FILES
    - **SRC/OS/SHARED**
  - OPERATING SYSTEM SPECIFIC LOW LEVEL IMPLEMENTATION FILES
    - **SRC/OS/POSIX, SRC/OS/RTEMS, SRC/OS/VXWORKS**
  - PORTABLE LOW LEVEL IMPLEMENTATION FILES THAT ARE SHARED AMONG MORE THAN ONE OS
    - **SRC/OS/PORTABLE**
    - THIS DIRECTORY ALSO CONTAINS ”STUB” IMPLEMENTATION FILES TO EXCLUDE FEATURES THAT A TARGET OS MIGHT NOT SUPPORT

OSAL - LAYERED ARCHITECTURE EXAMPLE (1)

Shared: osapi-task.c

```
int32 OS_TaskCreate ( …
{
      …
      return_code = OS_TaskCreate_Impl(...
      …
}
```

POSIX: os-impl-tasks.c

```
int32 OS_TaskCreate_Impl ( …
{
      …
      …
}
```

Contains code that is common to all ports and would otherwise have to be duplicated for each supported OS

Contains POSIX specific implementation code for creating an OSAL task

OSAL - LAYERED ARCHITECTURE EXAMPLE (2)

Shared: osapi-socket.c

```
int32 OS_SocketOpen ( …
{
      …
      return_code = OS_SocketOpen_Impl(...
      …
```

Contains code that is common to all ports and would otherwise have to be duplicated for each supported OS

Selected by CMakeLists.txt

Network ON

Network OFF

Portable: os-impl-**bsd**-sockets.c

```
int32 OS_SocketOpen_Impl ( …
{
    ...
```

Contains Portable implementation code for creating a BSD socket

Portable: os-impl-**no**-sockets.c

```
int32 OS_SocketOpen_Impl ( …
{
    ...
```

Contains “No-Op” implementation for targets that don’t support the sockets API.

OSAL - BOARD SUPPORT PACKAGE (BSP)

- ON PREVIOUS VERSIONS OF THE OSAL, THE BSP DIRECTORY CONTAINED STARTUP CODE BUILD RULES, AND TOOLCHAIN OPTIONS FOR A SPECIFIC TARGET
  - THE BSP CODE WAS USED FOR EXAMPLES AND STANDALONE OSAL APPLICATIONS
  - THE CFS USED THE PLATFORM SUPPORT PACKAGE (PSP) PROVIDED THE STARTUP CODE, BUILD RULES, AND TOOLCHAIN OPTIONS IN PLACE OF THE OSAL BSP
- STARTING WITH OSAL 5.1, THE OSAL BSP CODE CONTAINS THE ENTRY POINT AND STARTUP CODE FOR THE CFE
  - THIS BRINGS CONSISTENCY TO THE STARTUP CODE AND FACILITATES UNIT TESTS
- OSAL BSP CODE INCLUDES:
  - OS/TARGET SPECIFIC STARTUP CODE (**GENERIC-LINUX**, **GENERIC-VXWORKS**, **PC-RTEMS**)
  - SHARED BSP FILES TO PREVENT CODE DUPLICATION
  - BUILD RULES AND DEFINITIONS

PLATFORM SUPPORT PACKAGE (PSP) - FEATURES

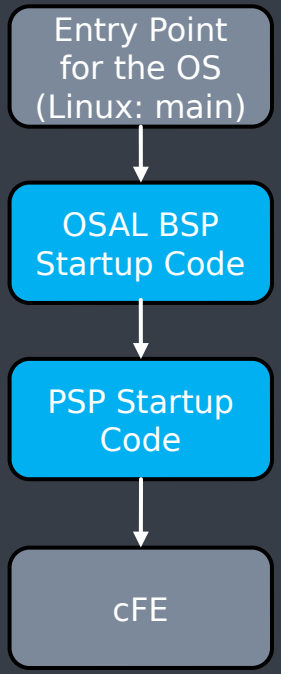
- THE PSP PROVIDES THE NECESSARY FUNCTIONS NEEDED TO ADAPT THE CFS TO A PARTICULAR OPERATING SYSTEM AND HARDWARE PLATFORM COMBINATION
  - EXAMPLES: VXWORKS/MCP750, POSIX/PC-LINUX
  - THE FUNCTIONALITY DOES NOT BELONG IN A GENERIC OS ABSTRACTION, BUT IS STILL NECESSARY TO SUPPORT THE CFS ON A SPECIFIC HARDWARE BOARD
- THE PSP PROVIDES THE FOLLOWING THROUGH STARTUP CODE AND FUNCTION LIBRARIES:
  - STARTUP OF CFS
  - WATCHDOG API
  - RESTART API
  - EXCEPTION SUPPORT
  - HARDWARE TIMER SUPPORT
  - FILE SYSTEM MAPPING
  - MEMORY FOR CRITICAL DATA STORES AND CFE CORE VOLATILE DISK
- § THE PSP ALSO PROVIDES SHARED IMPLEMENTATION FILES
  - FSW/SHARED

PSP – SUPPORTED PLATFORMS

- THE CURRENT PSP RELEASE SUPPORTS THE FOLLOWING PLATFORMS
  - **PC-LINUX**
    - A GENERIC LINUX PLATFORM IMPLEMENTATION FOR CFS TEST AND DEVELOPMENT
    - DESPITE THE “PC” NAME, IT WILL RUN ON A NUMBER OF 32-BIT AND 64-BIT LINUX PLATFORMS INCLUDING VIRTUAL MACHINES, ARM BASED SINGLE BOARD COMPUTERS LIKE THE RASPBERRY PI
    - IT CAN BE USED AS A STARTING POINT FOR NEARLY ANY EMBEDDED LINUX PLATFORM
  - **MCP750-VXWORKS**
    - A VERY SPECIFIC VXWORKS PLATFORM FOR AN INTERNAL CFS TESTING PLATFORM
    - CONSISTS OF A MCP750 POWER PC PROCESSOR CARD AND VXWORKS 6.X (LATEST IS 6.9)
    - CAN BE ADAPTED TO OTHER VXWORKS TARGETS
  - **PC-RTEMS**
    - FOR TESTING ON QEMU X86 MODEL RUNNING RTEMS 4.11 OR 5.1
    - CAN BE ADAPTED TO OTHER RTEMS TARGETS
- IT IS BEYOND THE SCOPE OF THE CFS PROJECT TO MAINTAIN EXTERNAL PSPS

SYSTEM STARTUP – INTRODUCTION

- HOW DOES THE CFS START?
- WHAT IS THE RELATIONSHIP BETWEEN THE STARTUP CODE IN THE OSAL BSP AND THE PSP?



SYSTEM STARTUP – STANDALONE OSAL EXAMPLE

OSAL BSP

OSAL: src/bsp/generic-linux/src/bsp_start.c

```
int main ( …
{
        OS_BSP_Initalize

        OS_Application_Startup

        OS_Application_Run
```

(1) OSAL Entry Point: main

OSAL Example

OSAL: src/examples/tasking-example/tasking-example.c

```
int OS_Application_Startup ( …
{

   …
}


void OS_Application_Run (void)
{
    ...

}
```

(2) OSAL Application Entry Point: OS_Application_Startup

(3) Create sample tasks

(4) OSAL Application: OS_Application_Run

SYSTEM STARTUP – CFS

OSAL
BSP

OSAL:
src/bsp/generic-linux/src/bsp_start.c

```
int main ( …
{
      OS_BSP_Initalize

      OS_Application_Startup

      OS_Application_Run
```

(1) OSAL Entry Point:
main

PSP

PSP: fsw/pc-linux/src/cfe_psp_start.c

```
int32 OS_Application_Startup ( …
{

      CFE_ES_Main (...
      …
}


void OS_Application_Run (void)
{
    ...

}
```

(2) PSP Entry Point:

OS_Application_Startup

(3) cFE Entry Point:
CFE_ES_Main

(4) PSP:
OS_Application_Run

EXCEPTION HANDLING

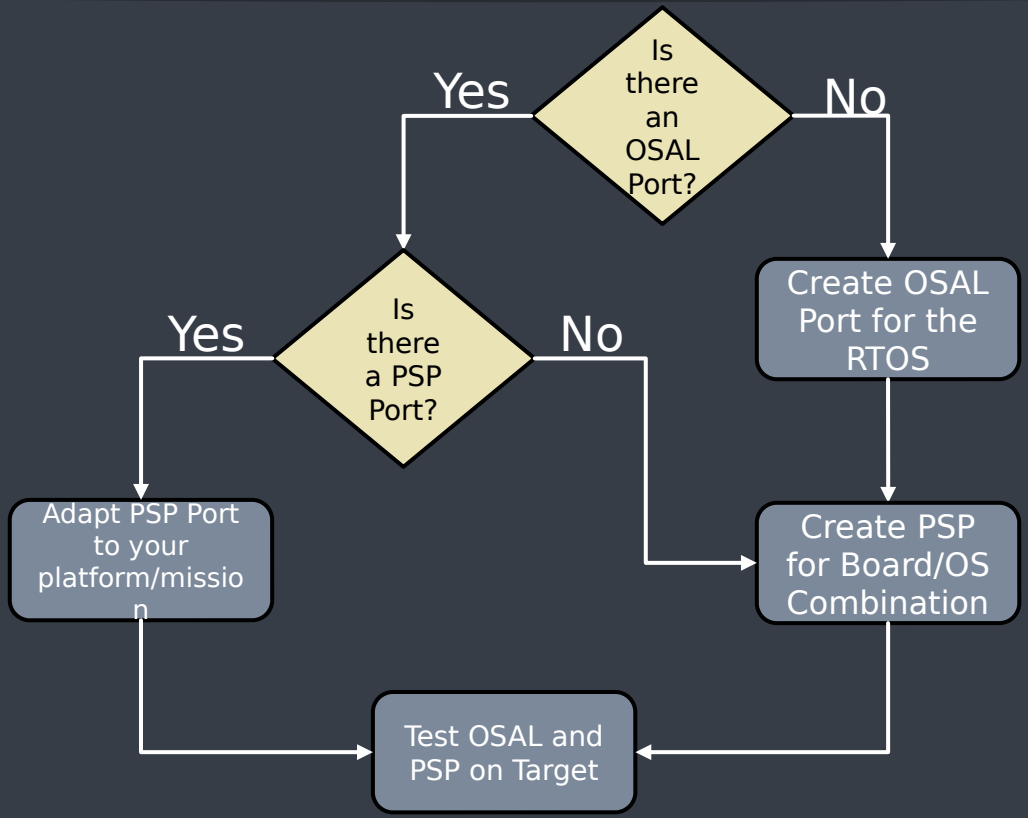
- LIKE MUCH OF THE CODE IN PSP AND OSAL, THE EXCEPTION HANDLING CODE HAS BEEN IMPROVED
- PREVIOUSLY, THE PSP WOULD CATCH AN EXCEPTION, AND CALL A CFE EXECUTIVE SERVICES (ES) FUNCTION TO LOG THE EXCEPTION AND HANDLE THE RESTART
  - THIS CAUSED PROBLEMS WHEN THE PSP CALLED ES FUNCTIONS IN AN INTERRUPT CONTEXT
  - IT ALSO VIOLATES LAYERING GUIDELINES BY HAVING THE PSP CALL ES FUNCTIONS (OTHER THAN THE ENTRY POINT)
- NOW THE EXCEPTION HANDLING IS COMPLETELY MANAGED IN THE PSP, AND ES WILL CALL INTO THE PSP TO CHECK FOR EXCEPTIONS:
  - AS PART OF ITS BACKGROUND TASK, ES POLLS THE PSP TO SEE IF ANY EXCEPTION EVENTS OCCURRED
  - IF SO, ES WILL CALL INTO THE PSP TO COLLECT THE EXCEPTION INFORMATION/CONTEXT
  - ES WILL THEN CALL THE PSP TO TAKE THE APPROPRIATE ACTION SUCH AS A RESET

CMAKE FILES

- CMAKE IS THE BUILD TOOL USED FOR CONFIGURING AND BUILDING THE CFS
- IMPORTANT CMAKE FILES FOR THE CFS PLATFORM LAYER:

Location	File	Purpose
Mission Defs	**default_osconfig.cmake**	replacement for osconfig.h header. Selects OSAL features such as network
Mission Defs	**toolchain_xyz.cmake**	Selects compiler, common options, OSAL port, OSAL BSP, and PSP
Mission Defs	**arch_build_custom.cmake**	Target compile options – currently used for warnings, but can be used for other options
OSAL Src (src/os/posix)	**build_options.cmake**	Options such as libraries to link and unit test/coverage options
OSAL Src	**CMakeLists.txt**	Selects OSAL implementation source files including portable files
OSAL BSP	**build_options.cmake**	BSP files and OSAL library to link
PSP (fsw/pc-linux)	**CMakeLists.txt**	Selects the PSP source files for this PSP
PSP	**make/ build_options.cmake**	PSP specific options

PORTING TIPS – HOW TO START

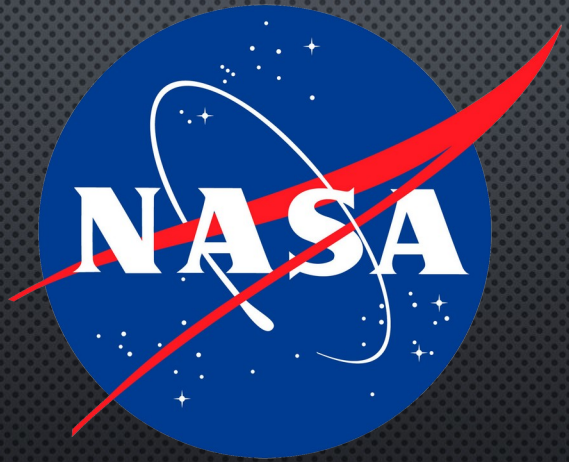


PORTING TIPS – A 1 MINUTE GUIDE

- PORT THE OSAL (IF NECESSARY)
  - START BY CLONING THE CLOSEST OS IMPLEMENTATION FOR YOUR TARGET OS
  - PORT THE "**IMPL**" FILES FOR THAT OS
  - ADJUST CMAKE FILES TO SELECT CORRECT PORTABLE MODULES FOR FEATURES LIKE FILES, NETWORK, ETC.
- PORT THE PSP
  - FOR A PORT THAT HAS AN EXISTING OSAL PORT, AN EXISTING PSP CAN OFTEN BE USED TO BRING UP THE CFS ON YOUR TARGET
    - 90% OF THE EFFORT IS IN 10% OF THE PSP PORT
  - FOCUS ON THE STARTUP CODE FIRST, THEN START IMPLEMENTING INTERFACES SUCH AS TIMER, WATCHDOG, MEMORY ACCESS, ETC.
- CREATE A TOOLCHAIN FILE – REQUIRED FOR CROSS COMPILATION

SUMMARY

- OTHER IMPROVEMENTS
  - UNIT TESTS IMPROVEMENTS
  - OSAL TIMER IMPROVEMENTS
- WHERE TO FIND THE OSAL AND PSP
  - OSAL: HTTPS://GITHUB.COM/NASA/OSAL
  - PSP: HTTPS://GITHUB.COM/NASA/PSP
  - CFS BUNDLE OFFERS PRE-PACKAGED INTEGRATION: HTTPS://GITHUB.COM/NASA/CFS
- HOW CAN I GET INVOLVED?
  - LOOK AT THE TICKETS IN THE REPOSITORIES
  - FIX PROBLEMS AND MAKE IMPROVEMENTS!
  - CONTRIBUTOR LICENSE AGREEMENT IS REQUIRED, BUT ALLOWS ANYONE TO CONTRIBUTE
- WHAT'S NEXT?
  - CERTIFICATION AND CAELUM RELEASE
  - INTEGRATION OF SYMMETRIC MULTI-PROCESSOR (SMP) FEATURES


NASA

ACRONYMS

API	application programming interface
App	Software Application
ARM	Advanced RISC Machines
BSD	Berkeley Software Distribution
BSP	Board Support Package
cFS	core Flight System
ES	Executive Services
GSFC	Goddard Space Flight Center
NASA	National Aeronautics and Space Administration
OS	Operating System
OSAL	Operating System Abstraction Layer
PC	Personal Computer
POSIX	Portable Operating System Interface
PSP	Platform Support Package
QEMU	Quick Emulator
RISC	Reduced Instruction Set Computer
RTEMS	Real-Time Executive for Multiprocessor Systems
RTOS	Real Time Operating System