NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GODDARD SPACE FLIGHT CENTER

Operating System Abstraction Layer (OSAL)

Configuration Guide

582-2007-00

October 23, 2007 (Version 1.0)

Authors:

A. Cudmore/Flight Software Engineer/Code 582 — Date

N. Yanchik/Flight Software Engineer/Code 582 — Date

Approvals:

Elaine Shell/ 582 Branch Head — Date

Acknowledgements

Revision History

Revision Number	**Release Date**	**Changes to Prior Revision**	**Approval**
1.0	10/17/07	Initial Release.	A. Cudmore

Table Of Contents

1 Introduction

1.1 Scope

The purpose of this document is to provide guidelines and conventions for the configuration and deployment of the Operating System Abstraction Layer (OSAL) to a desired platform or platforms.

1.2 Background

The goal OS Abstraction Layer is to promote the creation of portable and reusable real time embedded system software. Given the necessary OS abstraction layer implementations, the same embedded software should compile and run on a number of platforms ranging from spacecraft computer systems to desktop PCs.

1.3 Applicable Documents

Document ID	Document Title

1.4 Acronyms

Acronym	Description
OS	Operating System
API	Application Programming Interface
CM	Configuration Management
CPU	Central Processing Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
HW, H/W	Hardware
RAM	Random-Access Memory
SW, S/W	Software
TBD	To Be Determined

1.5 Glossary of Terms

The following table defines the terms used throughout this document. These terms are identified as proper nouns and are capitalized.

Term	Definition
Application (APP)	A generic term for a computer program in a desktop or embedded system. An Application is generally not part of the operating system.
Application Programmer's Interface (API)	A set of routines, protocols, and tools for building software applications
Board Support Package (BSP)	A collection of user-provided facilities that interface an OS and the cFE with a specific hardware platform. The BSP is responsible for hardware initialization.
Core Flight Executive (cFE)	A runtime environment and a set of services for hosting FSW Applications

Cyclic Redundancy Check	A polynomial based method for checking that a data set has remained unchanged from one time period to another.
Developer	Anyone who is coding a software Application.
Hardware Platform	The target hardware that hosts the an Operating System and Applications.
Interface Control Document	A document that describes the software interface, in detail, to another piece of software or hardware.
I/O Data	Any data being written to and read from an I/O port. No structure is placed on the data and no distinction as to the type of I/O device. I/O data is defined separately from memory data because it has a separate API and it's an optional interface of the cFE.
Log	A collection of data that an application stores that provides information to diagnose and debug FSW problems.
Memory Data	Any data being written to and read from memory. No structure is placed on the data and no distinction as to the type of memory is made.
MMU	Memory Management Unit. A piece of hardware that manages virtual memory systems. It automatically translates addresses into physical addresses so that an application can be linked with one set of addresses but actually reside in a different part of memory.
Network	A connection between subsystems used for communication purposes.
Platform	See "Hardware Platform" above.
User	Anyone who interacts with the a Software Application or system in its operational state. A user can be a developer, a tester, , an operator, or a maintainer.

2 How to Configure, Build, and Run the OSAL

The OSAL distribution includes a complete development environment with support for a number of processors and operating systems. The OSAL development environment has been designed to isolate the portable OS source code from the OSAL applications, configuration parameters, and build products. The development environment is an example of how to configure and build portable software using the OSAL code, but it is by no means a requirement to use the OSAL. The included platforms for the OSAL can be used as starting points for other boards and CPUs.

The following sections provide instructions on how to:

- Setup the build environment
- Configure the build directory for an OSAL application
- Configure a OSAL Application
- Build the OSAL Application
- Load the OSAL Application on to the target platform
- Run the OSAL Application on the target platform

2.1 Setup the Build Environment

This section details the steps needed to setup the OSAL source distribution and prepare the host development environment to build the OSAL.

2.1.1 Setup the OSAL Source Distribution

Get a copy of the OSAL source distribution directory on your build machine. The source distribution has the following directories:

OSAL source distribution directories

Directory	Description
osal	The top level OSAL source distribution directory. OSAL version 2.10 is being used as an example.
osal/src	The src directory contains all OSAL source and make rules.
osal/src/apps	The apps directory contains the sample and test applications for the osal.
osal/src/arch	The arch directory contains the architecture (processor) specific code for the OSAL as well as the BSP (Board Support Package) code to make the OSAL run on a particular platform. Everything in this directory is used to adapt the OSAL and Applications to a particular hardware platform. This directory also contains the startup code for the example programs. The included platforms are generic enough that they may be easy to port to other platforms and processor architectures. For example: The arch/coldfire/mcf5235/rtems board support package was ported to an ARM processor running RTEMS with minimal effort.
osal/src/inc	The inc directory contains system wide include files that are used by the OSAL on all platforms. Currently only the common-types.h file is in this directory.
osal/src/make	The make directory contains common makefiles for building the OSAL and it's applications.
osal/src/os	The os directory is the heart of the OSAL, containing the implementation of the OSAL for each supported operating system. There is a sub-directory for each supported operating system in this directory. The OSAL include files are also contained in this directory (src/os/inc).

osal/build	The build directory contains a framework for building an OSAL application. The files in this directory allow easy customization and configuration for any supported OS or platform for the OSAL. By changing a few variables in a file, the OSAL examples and test can be built for any of the supported platforms.
osal/doc	The doc directory contains the documentation and release notes for the OSAL.

The osal directory can go just about anywhere on a host development system.

Example directory structure locations

Host Operating System	**Example Directory**	**Notes**
Windows/Cygwin	/home/osaluser/osal	1. Building in Cygwin requires the "bash" command shell. Cygwin provides a virtual unix style directory directory structure, so "/home/osaluser/osal" might translate to C:\cygwin\home\osaluser\osal. 2. Putting the cFE directory under a directory with spaces is not recommended. i.e. ("My Documents")
Windows/vxWorks 6 Development Shell	C:\osalproject\osal	1. Building on Windows with the vxWorks 6.x development tools requires using the "vxWorks Development Shell". The system will not build on a standard Cygwin Shell, or a windows DOS prompt.
Linux	/home/osaluser/osal	
Mac OS X	/Users/osaluser/osal	

2.1.2 Create System Environment Variable(s)

The OSAL development environment requires one system environment variable to be set in order to build the example programs. The directory also contains a shell script "setvars.sh" to set the environment to the current OSAL directory.

Environment Variables Needed by the cFE

Environment Variable	**Value (in Linux as an example)**	**Notes**
OSAL_SRC	/home/osaluser/osal/src	The location of the OS Abstraction Layer source code. This directory can be moved anywhere as long as the environment variable is set accordingly.

Example Environment Variable for Different Development Hosts

Host Operating System	**Example Environment Variables**	**Notes**
Windows/Cygwin	$ export OSAL_SRC=/home/osaluser/osal/src	1. The Windows/Cygwin environment variables are "Bash" shell variables, not Windows

		environment variables.
Windows/vxWorks 6 Development Shell	% set OSAL_SRC=C:/osalproject/osal/src	1. These environment variables can be set in the Windows control panel under system/environment variables. 2. Note the forward slash directory separators in the DOS environment variables. Because the vxWorks tools are half DOS and half-Unix, they don't seem to like the DOS style backslash.
Linux	$ export OSAL_SRC=/home/osaluser/osal/src	These settings can be set in the user's .bash_profile
Mac OS X	$ export OSAL_SRC=/home/osaluser/osal/src	These settings can be set in the user's .bash_profile

2.2 Configure the Build Directory for the OSAL application

The build directory is where the OSAL is configured and compiled for a particular processor, board, and OS. The build directory is designed to hold the OSAL configuration for the selected platform. The **core** directory is where the core OS code, bsp code, and hardware abstraction layer (hal) are built. They are left in the core directory for for the applications to link against. The build directory can have multiple OSAL applications to build for a particular platform. The OSAL distribution contains directories for example and test applications. Multiple build directories can be used to configure the OSAL for different platforms in the same environment, each with it's own unique OSAL configuration.

2.3 Configure the 'build' Directory

In order to build the OSAL for one of the supported platforms, the OSAL build directory must be properly configured. This involves editing a couple of configuration files and setting up one or more sample applications that use the OSAL API.

2.3.1 Define the CPU, Operating System, and Processor Board

In the build directory, edit the **'osal-config.mak'** file and set the options for your target. The default settings in the osal-config.mak are for running vxWorks6.4 on an generic PowerPC board.

osal-config.mak Settings

osal-config.mak variable	Valid selections	Notes
HWARCH	x86, ppc, coldfire	
PLATFORM	genppc, pc, mac, mcf5235	
OS	vxworks6, rtems, osx, linux	1. VxWorks 5.5 is no longer supported. 2. use linux on cygwin
BSP	vxworks6.4, linux, osx, rtems	Use linux on cygwin

Note that not all combinations are valid. See the Platform Specific Section for more information on each supported cFE target.

2.3.2 Configure the OSAL Parameter File

The file **osconfig.h** has configuration parameters for tailoring the OSAL parameters. Most parameters set upper bounds on the number of OS objects that can be created. The OSAL keeps track of allocated OS objects using fixed size tables.

OSAL configuration parameters

Parameter	Description
OS_MAX_TASKS	The maximum number of tasks that can be created in the running OSAL application.
OS_MAX_QUEUES	The maximum number of queues that can be created in the running OSAL application.
OS_MAX_COUNT_SEMAPHORES	The maximum number of counting semaphores that can be created in the running OSAL application.
OS_MAX_BIN_SEMAPHORES	The maximum number of binary semaphores that can be created in the running OSAL application.
OS_MAX_MUTEXES	The maximum number of mutexes that can be created in the running OSAL application
OS_MAX_PATH_LEN	The maximum length for an absolute path length in the OSAL File API.
OS_MAX_API_NAME	The maximum length for an individual file name in the OSAL File API.
OS_BUFFER_SIZE	The maximum size of a formatted text message

	for the OS_printf API.
OS_BUFFER_MSG_DEPTH	The maximum number of messages buffered by the OS_printf API.
OS_UTILITY_TASK_ON	Turns on a utility task that will read the statements to print from the OS_printf function. If this define is commented out OS_printf will print the text under the context of the caller. The utility task will use TBD resources from the above allocated resources.
OS_UTILITYTASK_STACK_SIZE	The size of the stack for the utility task.
OS_UTILITYTASK_PRIORITY	The priority of the utility task.

2.4 Configure one or more OSAL Applications

Once the OSAL is configured and ready to build, a OSAL application can be configured in the build directory. Multiple OSAL applications can be created in this directory. The application source code can come from the src/apps directory, or the applications can be contained completely within the build directory. The OSAL source distribution has a set of test and example applications in the src/apps directory and a set of corresponding application directories and makefiles in build directory.

2.4.1 Configure a sample application in the build directory

The following show the files needed for a sample OSAL application in the build directory.

Sample OSAL Applications and the associated files

File	Description
build/example1	Directory for the included OSAL example Application.
build/example1/Makefile	Makefile for the example OSAL app. Because the source is in the src/apps/example1 directory, there is no need to include it here. The Makefile will find it using the OSAL_SRC environment variable. The source could be copied here in order to customize it.
build/new_osal_app	Directory for a new OSAL application.
build/new_osal_app/Makefile	Makefile for a new OSAL application.
build/new_osal_app/new_osal_app.c	Source file for the new OSAL application.
build/new_osal_app/new_osal_app.h	Header file for the new OSAL application.

The Application Makefiles have a specific format, so it is best to copy one of the application Makefiles from the build directory, such as build/example1.

2.4.2 Configure the application's main entry point

The OSAL development environment provides the main entry point/startup code for the Application. This code is located in the src/arch/<cpu-arch>/<platform>/<os>/bsp directory. The startup code will call the Application's entry point which is named: void OS_Application_Startup(void)

2.5 Build the OSAL core and Applications

Once the OSAL Core and Applications are set up in a build directory, everything can be compiled. The OSAL Core or any of the Applications can be built from individual make files, or they can be built from the top-level Makefile in the build directory.

Build Commands

Shell command	Description
`$ cd build`	Change to the build directory.
`$ make`	Build the OSAL Core, and all Applications
`$ make clean`	Clean the OSAL Core, and all Applications
`$ cd core; make`	Build the OSAL Core files only.
`$ cd example1; make`	Build the example1 Application only. NOTE: The OSAL Files have

	to be compiled in order for the Application to link.
$ make depend	Recalculate the dependencies on the OSAL Core files and apps

Once the OSAL Applications are built, they are ready to load and execute on the target. The filename of the executable is dependent on the OS it is built for.

OSAL Application executable name

Target Operating System	Application executable name	Notes
vxWorks 6.x dynamic link	example1.elf	The vxWorks PowerPC platforms use a dynamically loaded object without the kernel.
Linux	example1.bin	
Mac OS X	example1.bin	
Windows/Cygwin	example1.bin	
Rtems/Coldfire	example1.nxe	This is a static linked executable, linked with the RTEMS kernel and BSP.

2.6 Load and Run the OSAL Applications

- Depending on the Target, it is usually straightforward to run an OSAL Application on a target platform. On desktop platforms, it is just a matter of running the executable program. On vxWorks, the example programs are loadable modules.

2.6.1 Load the OSAL Application Executable on the Target

On desktop targets the cFE Core can be run from the directory where it was compiled. On embedded targets, the Application has to be loaded into a remote file system, or booted over the network. On the vxWorks PowerPC targets, the Application can be loaded into the EEPROM or Flash disk after the vxWorks kernel is booted. On RTEMS targets, the Application can be loaded using the CEXP dynamic loader or it can be linked in with an RTEMS Binary. See the target specific sections for details on each platform.

2.6.2 Setup the Target File Systems

Because the OSAL runs on many different platforms, it must be able to deal with different file system types and different paths. The OSAL accomplishes this by using a file system abstraction. The abstracted OSAL file system is similar to a UNIX file system, where the root directory starts with "/" and all disks are mounted on directory trees. For example:

- /ram0/apps/ → RAM disk 0, apps subdirectory
- /ram1/data/ → RAM disk 1, data subdirectory
- /hd0/tables/ → Hard Disk 0, tables subdirectory

Using this abstraction, a file "datafile1.dat" on RAM disk 1 might be accessed from the OSAL by using the path "/ram1/data/datafile1.dat". Using the host vxWorks tools, the path to the same file would be: "RAM:0/data/myfile.dat". If the OSAL is running on a Linux development workstation, the file might be located at: "/tmp/ramdev1/data/myfile.dat". The important part is that the OSAL Application can access the files using a generic path, allowing the software to remain portable.

There are a few ways to map these host file systems to OSAL file systems:

- **Map existing target file systems to a OSAL path**. This is one of the most common ways to map the Non-Volatile disk to the OSAL. The OSAL relies on the target OS to create/mount a file system and it simply is given a mapping to the disk to allow the OSAL to access it.
- **Create EEPROM/Flash/ATA File systems**. The OSAL has the ability on some targets to format or initialize a EEPROM or ATA disk device. This is less commonly used.
- **Create RAM File Systems**. The OSAL can create RAM disks on the vxWorks targets. The OSAL will create or re-initialize the RAM disk for the vxWorks targets.

The following table shows examples of these file system mappings on various hosts:

OSAL File system mapping

Target Operating system	**cFE File system path**	**Target OS File system path**	**Notes**
vxWorks 6.x	/ram	RAM:0/	Most vxWorks targets
	/cf	CF:0/ or CF:1/	MCP750
	/cf	EEP:0/	RAD750 target
Linux	/ram	/tmp/ramdev0	Multiple users on the same development machine should remap this to use a local directory.
	/cf	/tmp/eedev0	
Mac OS X	/ram	/tmp/ramdev0	Multiple users on the same development machine should remap this to use a local directory.
	/cf	/tmp/eedev0	
Windows/Cygwin	/ram	/tmp/ramdev0	On Cygwin, the directory is a Cygwin shell directory. /tmp/ramdev0 will probably map to: C:\cygwin\tmp\ramdev0
	/cf	/tmp/eedev0	C:\cygwin\tmp\eedev0

2.6.3 Start the OSAL Application on the Target

Starting an OSAL Application is a highly target dependant activity. The following table gives examples of how to start an Application on various platforms. For full details see the notes for each section.

How to start an OSAL Application on Various Target Systems:

"Target" operating system	**How to start the cFE**
RTEMS	Loaded through GDB/BDM using a shell script: "debug.sh"
vxWorks 6.2 / RAD750	Started from the vxWorks Target Shell commands: Vx> ld < example1.elf `Vx>  OS_BSPMain`
Linux	Start directly from the linux shell: $ ./example1.bin
Mac OS X	Start directly from the OS X shell: $ ./example1.bin
Windows/Cygwin	Start directly from the Cygwin shell: $ ./example1.bin

Target Specific Instructions

This section provides details on how to load and run each of the supported OSAL configurations.

2.7 Generic PPC / vxWorks 6.4 Platform:

The Generic PPC applications will work on both the Motorola MCP750 and the BAE RAD750 running vxWorks 6.4. On this platform, the OSAL Applications are built as dynamic loadable vxWorks modules, rather than being linked to the vxWorks kernel/BSP. The OSAL Applications are loaded into the compact flash disk on the MCP750, so it can be started from a vxWorks shell or startup script after the kernel comes up.

2.7.1 OSAL Configuration for the Generic PPC / VxWorks 6.4

cfe-config.mak Settings

Prolog.mak variable	Required selection	Notes
HWARCH	PPC	
PLATFORM	genppc	
OS	vxworks6	
BSP	vxworks6.4	

2.7.2 File System Mappings on the MCP750 PPC Board

The cFE uses the following file system mappings for the MCP750 PPC Board. The file system mappings are defined in the bsp_voltab.c file in the src/arch/ppc/genppc/vxworks6.4/bsp directory:

OSAL File System Mappings

OSAL "device"	File System Type	OSAL Path	Host Path	Notes
/ramdev0	Real RAM Disk (vxWorks)	/ram	RAM:0/	
/eedev0	File System Mapped (FS_BASED)	/cf	eep:0/	This is the Compact Flash drive on the MCP750
/ramdev1 – /ramdev5	Real RAM Disk	N/A	N/A	Unused table entries for applications to create new RAM disks
/ssedev0 - /ssrdev2	File System Mapped (FS_BASED)	N/A	/ssr:0/SSR1 - /ssr:0/SSR3	Unused table entries for applications to map Hard Disk device directories to "pseudo" SSR file systems.

2.7.3 How to run the OSAL Applications on the MCP750 or RAD750

1. Load the kernel. The custom vxWorks kernel is loaded into the MCP750 via TFTP. We use a vxWorks boot image (Rather than the Motorola boot monitor/loader) to boot the MCP750 board, TFTP the "real" kernel to RAM, and execute it. This vxWorks boot image also sets the network settings for the "real" kernel image. On our OSAL/cFE development system, we keep the loadable vxWorks kernel image in a TFTP directory on the development workstation. So the vxWorks kernel image goes in /tftpboot/cpu1/cfecpu1.st. **($ cp /opt/workspace/mcp750image/default/vxWorks /tftpboot/cpu1/cfecpu1.st)**

2. Copy the "example1.elf" (or other executable name) loadable module into the non-volatile disk. On the MCP750, this is done simply by FTPing the example1.elf file to the target:

```
$ ftp 192.168.1.4
 ftp> username: target
 ftp> password: password
 ftp>  cd "CF:0"
ftp> binary
ftp> put example1.elf
```

3. Load the example Application in the vxWorks shell:

```
vx> cd "CF:0"
vx> ld < example1.elf
```

4. Run the example Application in the vxWorks shell:

```
vx> OS_BSPMain
```

(The entry point for the examples and test programs is always OS_BSPMain)

2.8 Axiom M5235 BCC / RTEMS 4.7:

The OSAL supports the Axiom 5235 BCC single board computer with an RTEMS 4.7 board support package. The tests and examples are built as static RTEMS executable programs for the board and can be loaded using the DBUG monitor or BDM port. The OSAL Makefiles assume the RTEMS 4.7 toolchain is installed on the host system in /opt/rtems-4.7 and the mcf5235 BSP is used. The files in arch/coldfire/mcf5235/rtems/make can be modified for a different environment.

2.8.1 OSAL Configuration for the Axiom M5235 BCC / RTEMS 4.7

cfe-config.mak Settings

Prolog.mak variable	Required selection	Notes
HWARCH	coldfire	
PLATFORM	mcf5235	
OS	rtems	
BSP	rtems	

2.8.2 File System Mappings on the Axiom M5235 BCC / RTEMS 4.7

The cFE uses the following file system mappings for the M5235 BCC Board with RTEMS. The file system mappings are defined in the bsp_voltab.c file in the src/arch/coldfire/mcf5235/rtems/bsp directory:

OSAL File System Mappings

OSAL "device"	File System Type	OSAL Path	Host Path	Notes
/ramdev0		/	/	Mapped to the IMFS root directory
/eedev0	File System Mapped (FS_BASED)	/cf	/	Mapped to the IMFS root directory
/ramdev1 – /ramdev5	unused	N/A	N/A	Unused table entries for applications to create new RAM disks. RTEMS does not currently have support for creating new RAM disks.
/ssedev0 - /ssrdev2	File System Mapped (FS_BASED)	N/A	/ssr:0/SSR1 - /ssr:0/SSR3	Unused table entries for applications to map Hard Disk device directories to "pseudo" SSR file systems.

2.8.3 How to run the OSAL Applications on the Axiom M5235 BCC with RTEMS 4.7

When the example application and test programs are all built as static executables for the M5235BCC board. The example programs can be loaded in the following ways:

- Using the BDM port through the GNU debugger. If the board is connected to the host PC with a BDM debugger cable, then the example programs can be loaded and run from there. For our environment we use the GNU Debugger with BDM Tools patches to allow GDB to talk directly to the BDM interface. The BDM tools project is at: http://bdmtools.sourceforge.net . There are support files included in src/arch/coldfire/mcf5235/rtems/bsp/rtems-supoort to help with loading and running via GDB/BDM.
- Using the BDM port to load the example in the Flash memory on the board. Using the bdmtools "bdmflash" program or another program such as CFFlasher for Windows, you could burn a flash version of the examples and boot the board directly. To do this, you must set the jumper on the

board to disable the DBUG monitor and you must use the correct linker script in RTEMS to target the application in Flash instead of RAM.

- Using the UART port and the DBUG monitor. The DBUG monitor on the board can be used to load a program into RAM.
- Using TFTP and the DBUG monitor. The sample applications can be transferred over the network to the board using the DBUG monitor.

It is a good idea to get the board working with the standard RTEMS demos before building and loading the OSAL Applications. Once the “hello world” RTEMS demo can be built, loaded and executed, you will have no problem loading the OSAL Applications.

2.9 Macintosh / OS X Platform:

2.9.1 OSAL Configuration for the Macintosh / OS X Platform

Prolog.mak Settings

Prolog.mak variable	Required selection	Notes
HWARCH	PPC / x86	Both PPC and x86 are supported.
PLATFORM	mac	
OS	osx	
BSP	osx	

Additional cFE Core configuration notes:

2.9.2 How to Run the cFE on the Macintosh / OS X Platform

1. To run an OSAL Application, simply execute the binary from a shell prompt:

build/example1]$ ./example1.bin

2.10 PC / Linux Platform

2.10.1 OSAL Configuration for the PC / Linux Platform

Prolog.mak Settings

Prolog.mak variable	Required selection	Notes
HWARCH	x86	
PLATFORM	pc	
OS	linux	
BSP	linux	

Additional cFE Core configuration notes:

2.10.2 How to Run the cFE on the PC / Linux Platform

1. To run an OSAL Application, simply execute the binary from a shell prompt:

build/example1]$ ./example1.bin

2.11 PC / Cygwin-Windows Platform

2.11.1 OSAL Configuration for the PC / Cygwin-Windows Platform

Prolog.mak Settings

Prolog.mak variable	Required selection	Notes
HWARCH	x86	
PLATFORM	pc	
OS	linux	The Linux OSAL port is used for cygwin. Cygwin provides the POSIX APIs needed for the OSAL.
BSP	linux	

Additional cFE Core configuration notes:

2.11.2 How to Run the cFE on the PC / Cygwin Platform

1. To run an OSAL Application, simply execute the binary from a shell prompt:

build/example1]$ ./example1.bin