Embedded Software Development with eCos –Chapter1   2
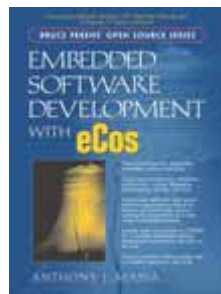
Chapter 1. An Introduction to the eCos World

- eCos Core Components
- Processor and Evaluation Platform Support
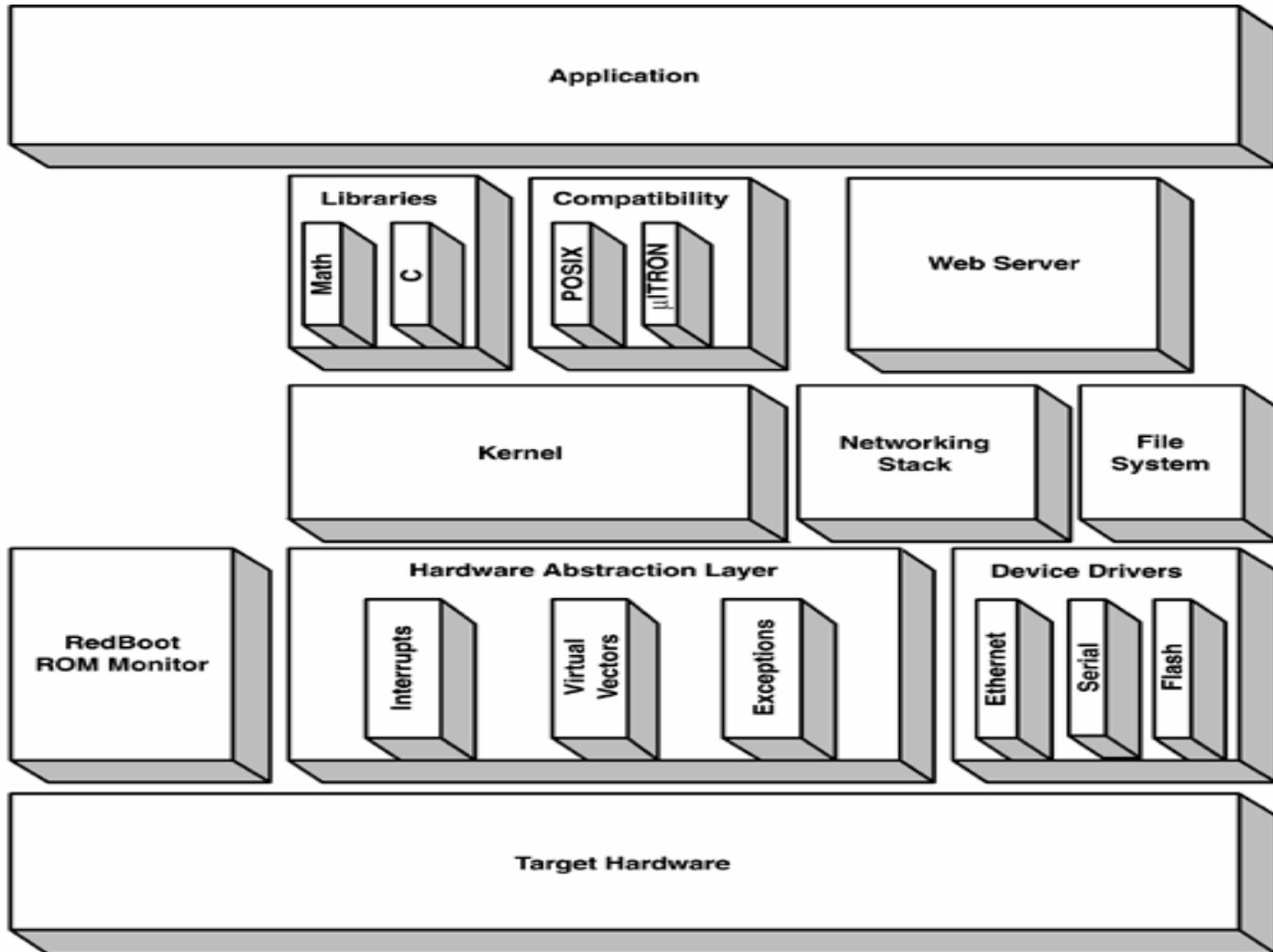- eCos Architecture Overview
- Component Framework

eCos Core Components

- Hardware Abstraction Layer (HAL)
- Kernel
- ISO C and math libraries
- Device drivers
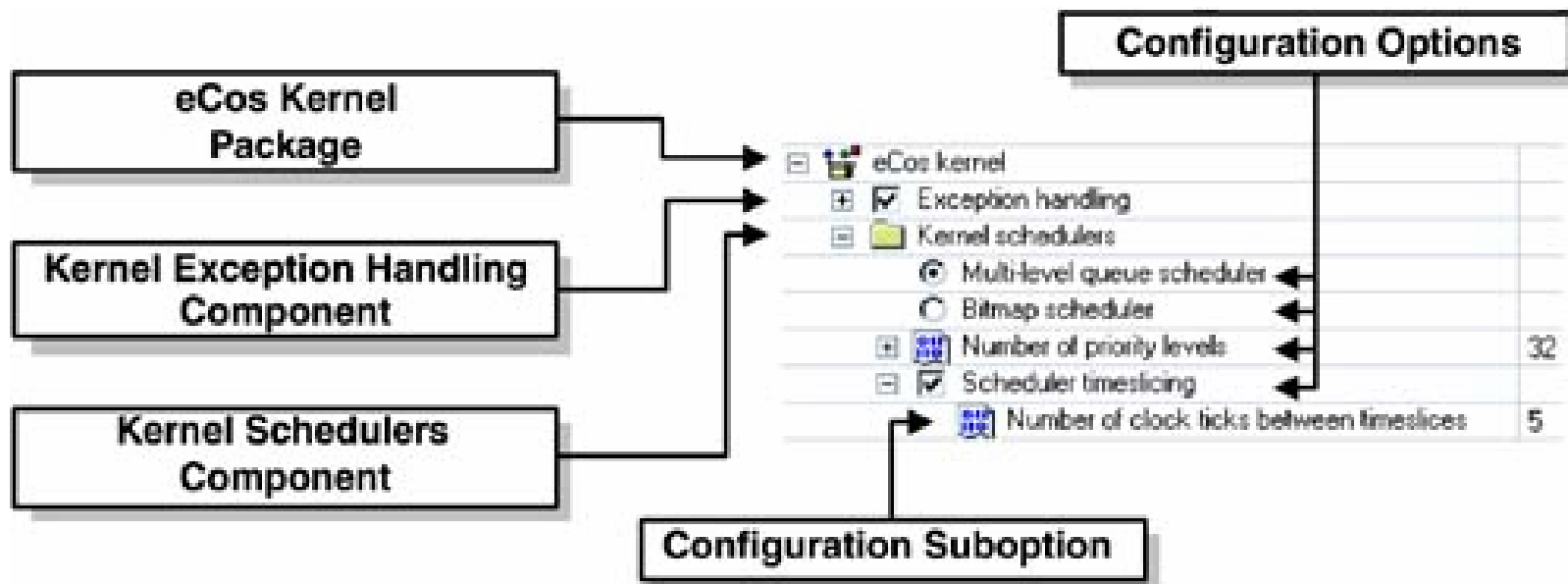- GNU debugger (GDB) support

Processor and Evaluation Platform Support

- ARM
- Fujitsu FR-V
- Hitachi H8/300
- Intel x86
- Matsushita AM3x
- MIPS
- NEC V8xx
- PowerPC
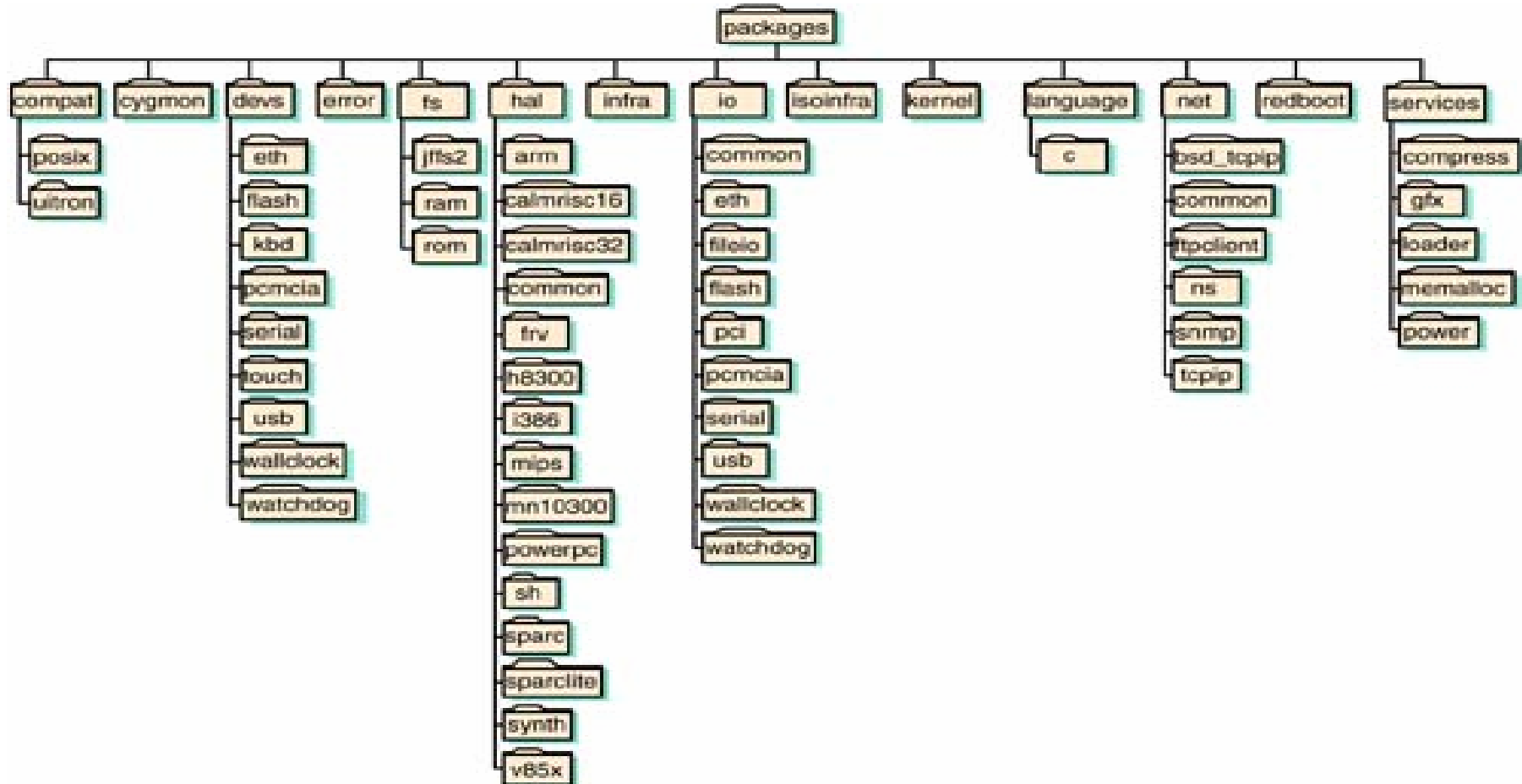- Samsung CalmRISC16/32
- SPARC
- SPARClite
- SuperH

Architecture Overview

Component Framework

High-level component repository directory structure snapshot

Component Repository Directory Structure Descriptions

Directory	Description
compat	Contains packages for the POSIX (IEEE 1003.1) and mITRON 3.0 compatibility.
cygmon	Package contents for Cygmon standalone debug monitor.[a]
devs	Includes all device driver hardware-specific components such as serial, Ethernet, and PCMCIA.
error	Contains common error and status code packages. This allows commonality among packages for error and status reporting.
fs	Includes the ROM and RAM file system packages
hal	Incorporates all HAL target hardware packages
infra	Contains the eCos infrastructure such as common types, macros, tracing, assertions, and startup options
io	Packages for all generic hardware-independent Input/Output (I/O) system support, such as Ethernet, flash, and serial, which is the basis for system device drivers

Component Repository Directory Structure Descriptions(Cont.)

Directory	Description
isoinfra	Contains package that provides support for ISO C libraries (such as stdlib and stdio) and POSIX implementations
kernel	Includes the package that provides the core functionality (such as the scheduler, semaphores, and threads) of the eCos kernel
language	Incorporates the packages for the ISO C and math libraries, which allows the application to use well-known standard C library functions and the floating-point mathematical library.
net	Packages for basic networking support including TCP, UDP and IP, and the SNMP protocol and agent support libraries based on the UCD-SNMP project.
redboot	Contains package for the RedBoot standalone debug ROM monitor
services	Includes packages for dynamic memory allocation and support for compression and decompression library.

Chapter 2. The Hardware Abstraction Layer

- HAL_ENABLE_INTERRUPTS() macro.
- HAL Structure
- HAL Directory Structure
- HAL startup procedure

HAL_ENABLE_INTERRUPTS() macro.

Listing 2.1 ARM architecture implementation of

```
#define HAL_ENABLE_INTERRUPTS()        \
    asm volatile (                     \
      "mrs r3,cpsr;"                   \
      "bic r3,r3,#0xC0;"               \
      "msr cpsr,r3"                    \
      :                                \
      :                                \
      : "r3"                           \
      );
```

HAL_ENABLE_INTERRUPTS() macro.

Listing 2.2 PowerPC architecture implementation of
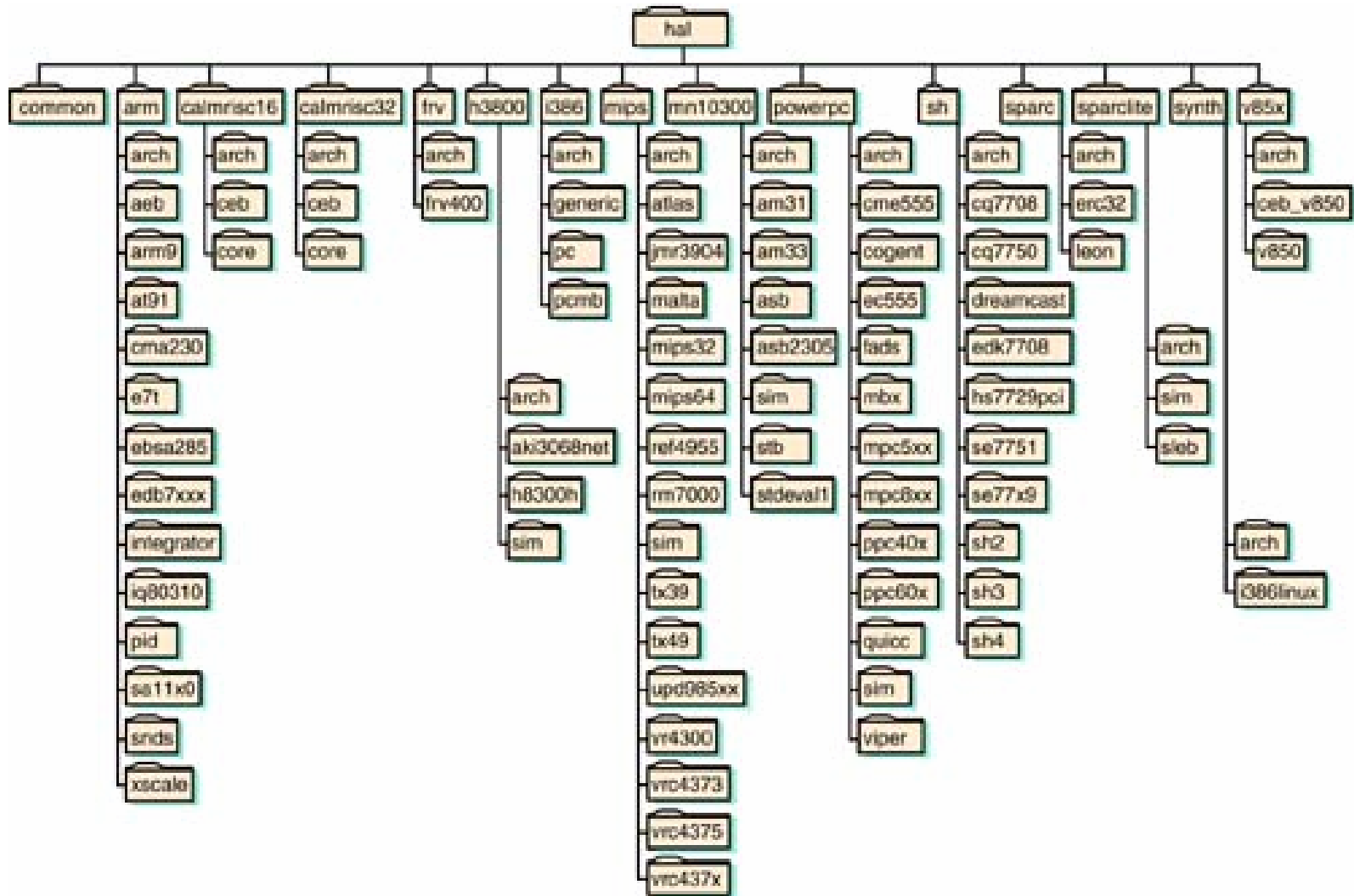
```
#define HAL_ENABLE_INTERRUPTS()        \
    CYG_MACRO_START                    \
    cyg_uint32 tmp1, tmp2;             \
    asm volatile (                     \
       "mfmsr  %0;"                    \
       "ori    %1,%1,0x8000;"          \
       "rlwimi %0,%1,0,16,16;"         \
       "mtmsr  %0;"                    \
       : "=r" (tmp1), "=r" (tmp2));    \
    CYG_MACRO_END
```
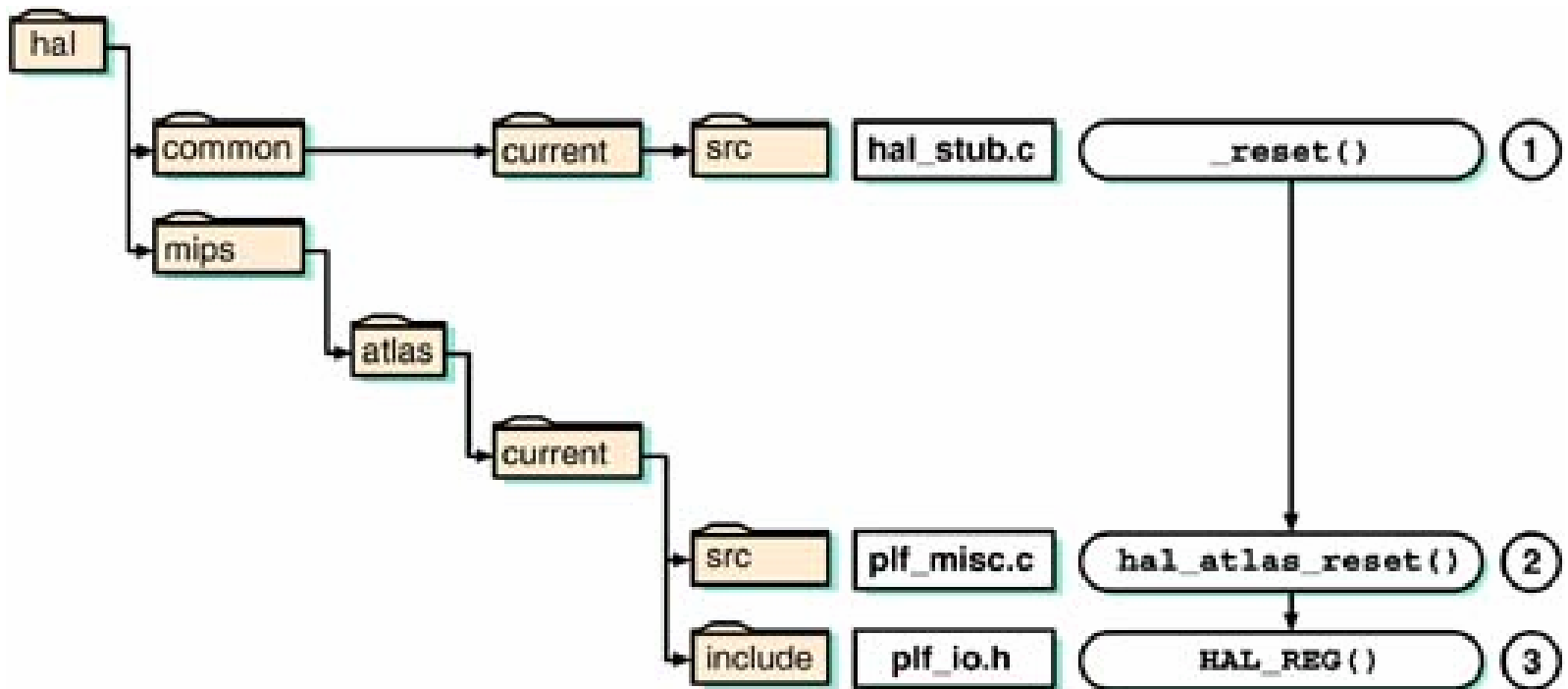
HAL Structure

- The HAL consists of four separate modules
  - Common HAL   (hal/common)
  - Architecture   (hal/<architecture>/arch)
  - Platform   (hal/<architecture>/<platform>)
  - Variant   (hal/<architecture>/<platform>)

HAL Directory Structure

Example
HAL Function Call Trace

HAL startup procedure

Hard or soft reset occurs   the processor jumps to its reset vector

Handles setting processor-specific registers   such as disabling instruction and data caches

Vector.S   starting point for all HAL initialization

Setting interrupt registers   disabling the processor watchdog setting real-time clock registers…etc.

This reserves a storage area for saving processor state information when an interrupt occurs

HAL startup procedure

(8) **Clear bss section** — HAL initialization process is to clear the BSS section

(9) **Setup C function call stack**

(10) **hal_platform_init** — Hal_aux.c file   calls hal_if_init() -> initializes the virtual vector table

(11) **hal_MMU_init** — Handles translations of logical addresses to physical addresses

(12) **hal_enable_caches**

(13) **hal_IRQ_init**

(14) **cyg_hal_invoke_constructors**

(15) **initialize_stub** — Installs the standard trap handlers and initializes the hardware for debug.

(16) **cyg_start On to kernel startup...**