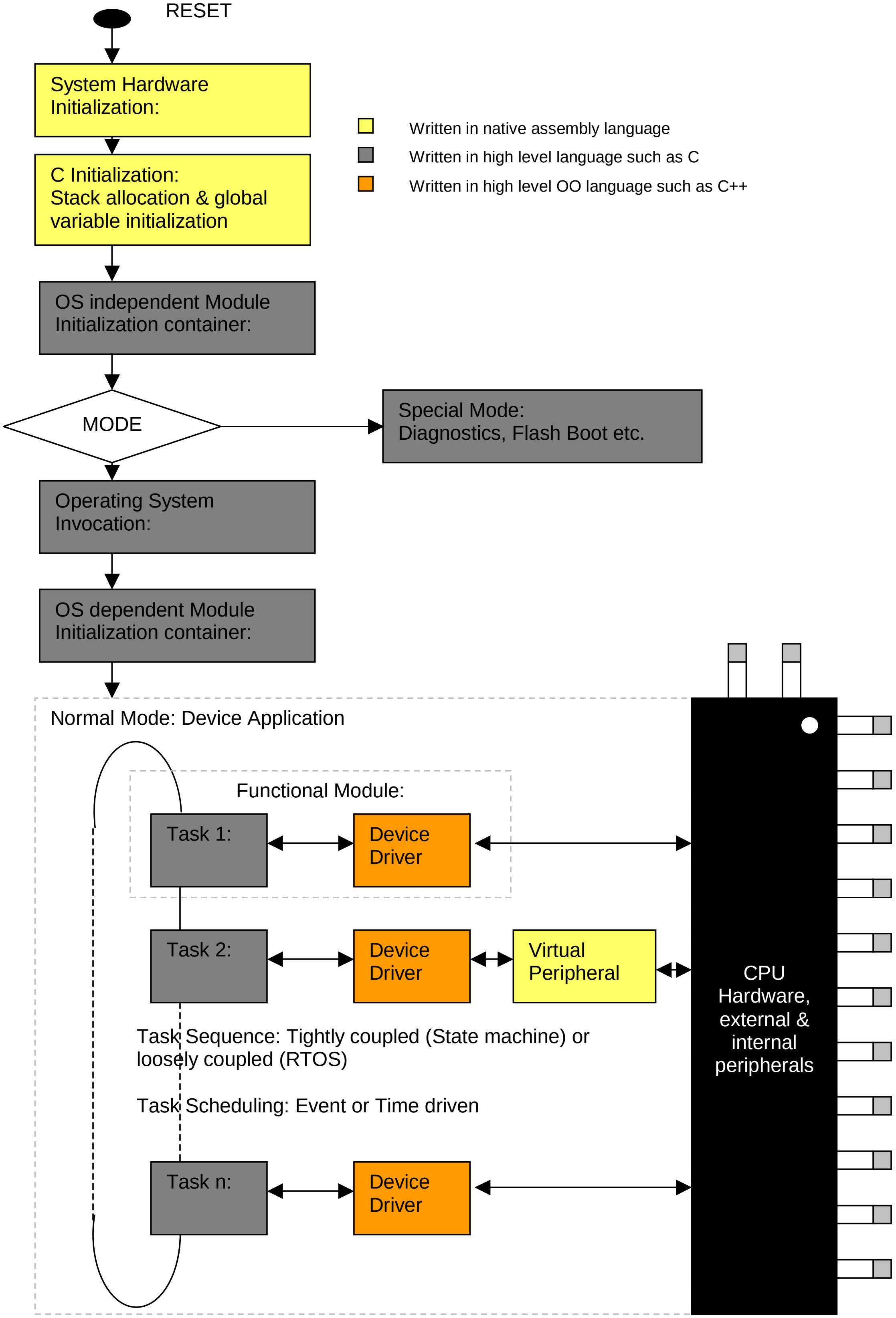
ARCHITECTURE SPECIFICATION

Embedded System Software Profile

embedded
technology

Rev 1.2 / 13.04.2002


RESET
System Hardware Initialization:
C Initialization: Stack allocation & global variable initialization
OS independent Module Initialization container:
MODE
Special Mode: Diagnostics, Flash Boot etc.
Operating System Invocation:
OS dependent Module Initialization container:
Normal Mode: Device Application
Functional Module:
Task 1:
Device Driver
Task 2:
Device Driver
Virtual Peripheral
Task n:
Device Driver
CPU Hardware, external & internal peripherals

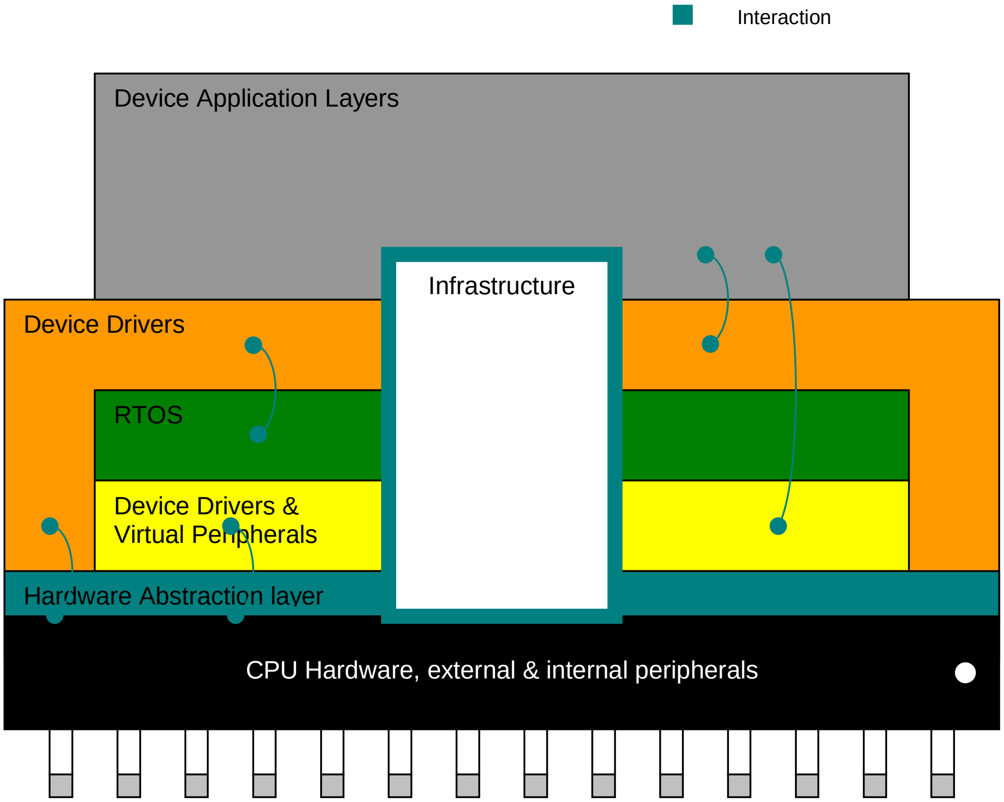
🟨 Written in native assembly language

▨ Written in high level language such as C

🟧 Written in high level OO language such as C++

Task Sequence: Tightly coupled (State machine) or loosely coupled (RTOS)

Task Scheduling: Event or Time driven


■ Interaction
Device Application Layers
Infrastructure
Device Drivers
RTOS
Device Drivers &
Virtual Peripherals
Hardware Abstraction layer
CPU Hardware, external & internal peripherals

TYPICAL MEMORY ORGANIZATION (Simple project)

RO Base →	RO space (text)	Executable image	**ROM (FLASH)**
	RW data copy		
	▨	Unused ROM space	
	Exception soft vectors	Software vectors space	**SRAM**
	Interrupt soft vectors		
	▨	Unused SRAM space	
	User stack	Stack space	
	SVC stack		
	Undef stack		
	Abort stack		
	IRQ stack		
	FIQ stack		
RW Base →	RW data (Initialized data)	Data space	**DRAM (SDRAM)**
	ZI data (Zero initialized data)		
	↑ Heap start address	Heap space	

Where:

□ Used memory space
▨ Unused memory space

Note: RO Base (program start address) & RW Base (Data start address) are 2 parameter set in the Linker flags.

Executable image:
Contains the following:

1. RO space or the text area, which is the executable code with the constant variables (*const*).
   Eg1. If(bFlag == nSuspended) return False;
   Eg2. const unsigned int wPortID = 0x365261;

2. RW data copy, which is the copy of all the initialize values of the RW data.
   Eg1. unsigned char bData = 23; /* 23, a copy of the RW initialize value kept in the RW data copy area */

Unused ROM space: This may not be present if the image exactly fits the size of the ROM.

Software vector space: This area makes it possible to change Interrupt/Exception Service (ISR/ESR) handlers at run-time. The start-up assembly file should have their hardware vector check the corresponding soft vector location & branch to the pointed to location (service function).
Contains the following:

1. Exception soft vectors – All the processor exceptions have a corresponding soft vector location here.
   Eg1. pISR_DABORT defined in k41.h (processor header file - ARM from Samsung)
2. Interrupt soft vectors – All the peripheral interrupts have a corresponding soft vector location here.
   Eg1. pISR_URX1 – UART1 Receiver interrupt soft vector.

Unused SRAM space: This buffer space may need not be maintained between the Software vector space & the messy stack space.

Stack space: This space is separated into chunks & allocated for the stack for each processor mode. The start-up assembly file needs to individually initialize the Stack Pointer (SP) for each processor mode with the corresponding Stack start address (High address – in the case of descending stack. Low address – in the case of ascending stack).
What goes in here?

1. Register saves between function calls including the return address (not in the case of ARM for it has the lr, so the return address of the previous function call is saved).
2. Local variables & function passed argument variables are allocated in stack space after the argument registers (CPU registers) are exhausted.

Note: For more information about these for the ARM, check the APCS for ARM.

Data space: The compiler (C & C++) allocates space for Global variables & the *static* variables in this space.
Contains the following:

1. RW data – These are the value initialized global variables. The initialize values are copied from the RW data copy region in ROM.
   Eg1. U32 FLDD_wStartAdr = 0x0a000000;
2. ZI data – These are the zero initialized variables.
   Note:The range information of this region is picked from the image placed next to the RW data copy region (this may be particular to the ARM).
   Eg1. U16 phwSamplePtr;   /* un-initialized pointer variable */

Heap Space: This space is utilized for allocating run-time memory. (malloc( ) – in C & key word, new in C++). The start of Heap is located just after the Data space.