DT 4013 Computer system engineering II

requirements not listed in `kernel-spec-march-2019.pdf`, `kernel_function`

concerning global variables and properties

4th March 2019

1 Required global variables

```c
int    Ticks;        /* global sys tick counter */
int    KernelMode;  /* can equal either INIT or RUNNING */
TCB   * PreviousTask, * NextTask;  /* changed on the 4th of March */
list  * ReadyList, * WaitingList, * TimerList;
```

globalVariables.c

2 Format for `main` when you test your kernel

```c
void main(void)
{
   SystemInit();
   SysTick_Config(100000); /* changed on 4th March 2019*/
   SCB->SHP[((uint32_t)(SysTick_IRQn) & 0xF)-4] =  (0xE0);
   isr_off();

   init_kernel();

   /*
      here you can create tasks
      and mailboxes
   */

   run();
   /*  the LoadContext() inside run shall enable interrupts */
}
```

formatForMain.c

3 Change the API header file

Use the header file: `kernel_functions_march_2019.h`

You need to use this modified header file, because we have changed the definition of the TCB. Now the TCB is smaller, because only the the stack pointer, and the 8 registers (R4 to

R11) are regularly saved and stored in the TCB fields dedicated to registers. The remaining registers are saved and retrieved from the stack. Those registers (R0 to R3, R12, LR, PC, PSR) are stored and retrieved by the hardware when

1. you call the assembly function `SwitchContext()`, or,
2. you call the assembly functions `isr_off()`, `isr_on()` or,
3. a Sys tick interrupt is serviced.

The stack storage and retrieval during `isr_off()`, `isr_on()` does not change the TCB involved. In other words, the TCB of the running task is the same before and after `isr_off()` is called. Ditto for `isr_on()`.

4 Change the linker configuration file

Use the header file: `sam3x8-sram_BigHeap.icf`

You need to use this because line number 40 has been edited to:

`define symbol __size_heap__ = 0x2000 ;`

You should place this file in your project folder, and tell the IAR IDE to use this as the linker configuration file. The way to tell tha is to go to:

Project->Options->Linker->Configuration

There, mark the box: override default (probably is already marked so). And make sure that the filed below points to the modified `icf` file in your project folder.

5 Initializing TCBs

```c
exception create_task( void(*taskBody)(), unsigned int deadline)
{
    TCB * new_tcb;
    new_tcb = (TCB *) calloc(1,size(TCB));
    /* you must check if calloc was successful or not ! */

    new_tcb->PC           = taskBody;
    new_tcb->SPSR         = 0x21000000;
    new_tcb->Deadline     = deadline;

    new_tcb->StackSeg[STACK_SIZE - 2] = 0x21000000;
    new_tcb->StackSeg[STACK_SIZE - 3] = (unsigned int) taskBody;
    new_tcb->SP           = &( new_tcb->StackSeg[STACK_SIZE - 9] );

    /* and add the other stuff you should be doing in this function
       */
}
```

initializingTCB.c

6 Stop coding in the IF-FIRST format

Because the software architecture of the assembly file has changed, you do context switching by calling:

- `LoadContext_In_Run()` in the function `run()`
- `LoadContext_In_Terminate()` in the function `terminate()`
- `SwitchContext()` every where else

The old functions `SaveContext()` `LoadContext()` are no longer available. Here is an illustration of how to change from the old coding style to the new style:

```c
void set_deadline(unsigned int d)
{
  static volatile int first;
  first = 1;
  isr_off();
  SaveContext();
  if (first == 1)
  {
    first = 0;
    RunningTask->Deadline = d;
    sort(ReadyList);
    RunningTask =
       ReadyList->pHead-pNext->pTask;
    LoadContext();
  }
}
```

ifFirst.c

```c
void set_deadline(unsigned int d)
{
 isr_off();
 NextTask->Deadline = d;
 PreviousTask = NextTask;
 sort(ReadyList);
 NextTask =
     ReadyList->pHead->pNext->pTask;
 SwitchContext();
}
```

switchContext.c

Keep in mind that `SwitchContext()` enables interrupts towards the end of its execution !

7 Required format for `run()`

```c
void run(void)
{
 NextTask = ReadyList->pHead->pNext->pTask;

 LoadContext_In_Run();
  /*       -- supplied to you in the assembly file ---------<
   * does not save any registers,
   * but simply restores registers from saved values
   * from the TCB of NextTask
   */
}
```

run.c

8 Required format for `terminate()`

```c
void terminate(void)
{
 isr_off();
 leavingObj = extract(ReadyList->pHead->pNext);
   /* detaches from the ReadyList and returns
    * the list object of the currently running task
    */
 NextTask = ReadyList->pHead->pNext->pTask;

 switch_to_stack_of_next_stack();
   /* switches to the stack of the task to be loaded
    */
 free(leavingObj->pTask);
 free(leavingObj);

 LoadContext_In_Terminate();
  /*        -- supplied to you in the assembly file --------<
   * does not save any registers; in specific, does not save the
   * process stack pointer (psp)
   * but simply restores registers from saved values
   * from the TCB of NextTask
   * note: the stack pointer is restored from NextTask->SP
   */
}
```

terminate.c

9 Use `memcpy` to copy data in the communication functions

In the communication functions (such as `send_wait()` for example), you have to copy data from the sender's area to the receiver's area. To do so in the mailbox called `mBox`, say:

`memcpy( receiving_pointer, sending_pointer , mBox->nDataSize)`

Please see http://www.cplusplus.com/reference/cstring/memcpy/ for more details.

10 Avoid using `printf` to debug

See https://www.embedded.com/design/prototyping-and-development/4025013/Say-no-to-stack-ov

If you must use `printf`, then make sure that interrupts are disabled before calling it.

11 Include files for writing an application program

Please have the following at the start of the C file of your application program:

`#include "system_sam3x.h"; #include "at91sam3x8.h";`