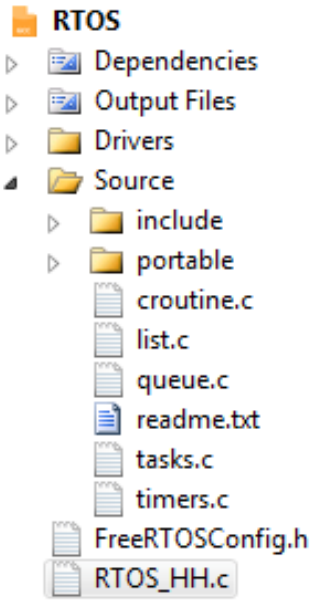
AMS Lab Exercise 7
"Free RTOS"

Purpose
To understand and have "Hands On" the FreeRTOS real time operating system.

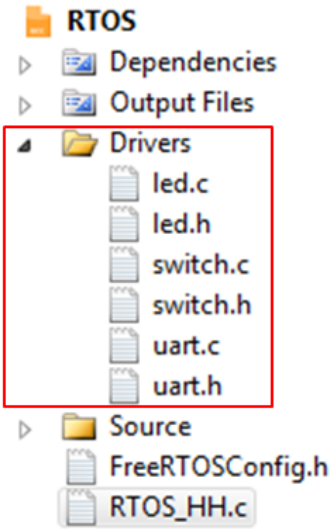
Literature
- www.freertos.org
- Demo program "RTOS_HH" (zipped at Blackboard).
- The "Free RTOS" lesson.

Exercise, Part 1
Start by downloading the demo program "RTOS_HH" from AMS Blackboard and unzip it to a proper location. Notice: Zipped Atmel Studio 6 project!

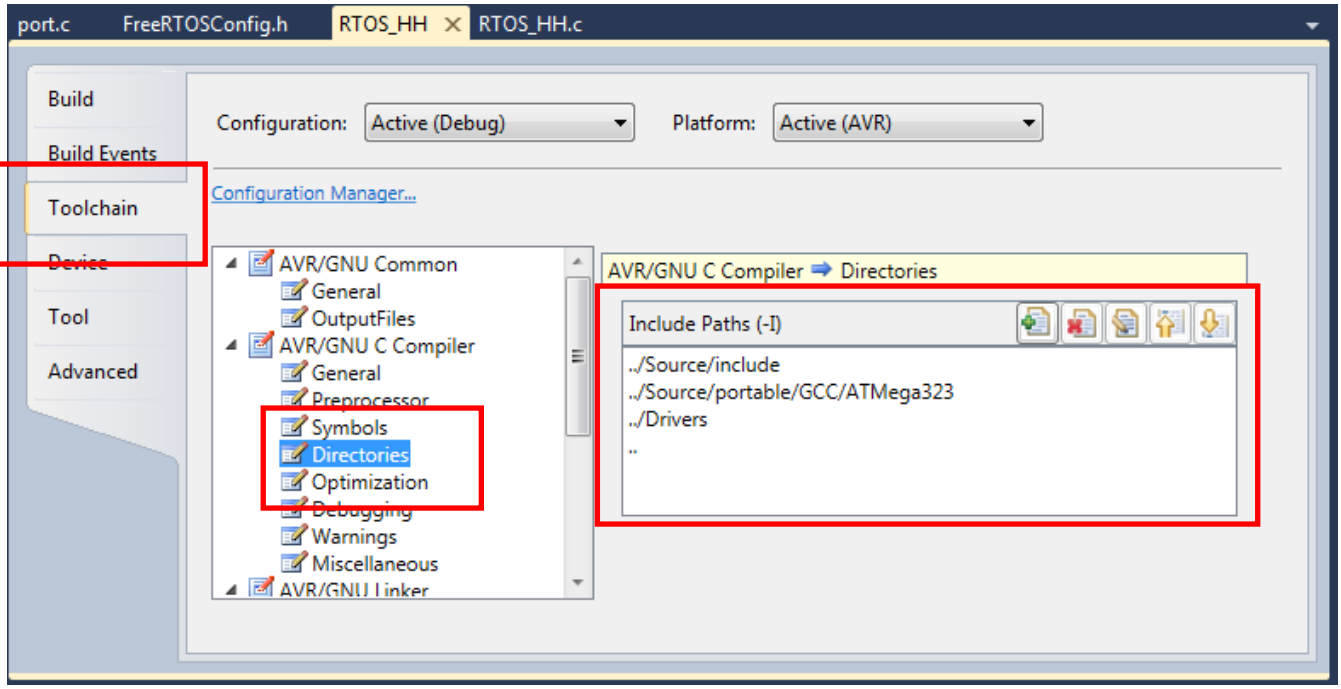
Then open the project and study its file structure:



Embedded in the project (folder "drivers") you will find some useful low level application drivers:



The file structure demands this setting for the project “Toolchain” -> ”Directories”:



Also study the FreeRTOS configuration file “FreeRTOSConfig.h”:

```
#define configUSE_PREEMPTION        1
#define configUSE_IDLE_HOOK         0
#define configUSE_TICK_HOOK         0
#define configCPU_CLOCK_HZ          ( ( unsigned long ) 3686400 )
#define configTICK_RATE_HZ          ( ( portTickType ) 1000 )
#define configMAX_PRIORITIES        ( ( unsigned portBASE_TYPE ) 1 )
#define configMINIMAL_STACK_SIZE    ( ( unsigned short ) 85 )
#define configTOTAL_HEAP_SIZE       ( (size_t ) ( 3500 ) )
#define configMAX_TASK_NAME_LEN     ( 8 )
#define configUSE_TRACE_FACILITY    0
#define configUSE_16_BIT_TICKS      1
#define configIDLE_SHOULD_YIELD     1
#define configQUEUE_REGISTRY_SIZE   0

/* Co-routine definitions. */
#define configUSE_CO_ROUTINES       0
#define configMAX_CO_ROUTINE_PRIORITIES ( 2 )

/* Set the following definitions to 1 to include the API function, or zero
to exclude the API function. */

#define INCLUDE_vTaskPrioritySet        0
#define INCLUDE_uxTaskPriorityGet       0
#define INCLUDE_vTaskDelete             0
#define INCLUDE_vTaskCleanUpResources   0
#define INCLUDE_vTaskSuspend            0
#define INCLUDE_vTaskDelayUntil         1
#define INCLUDE_vTaskDelay              1
```

The program simply creates two tasks, each blinking a LED:

```
/*****************************************************
FreeRTOS demo program.
Implementing 2 tasks, each blinking a LED.

STK500 setup:
  * PORTC connected to LEDS.

Henning Hargaard 13.2.2016
*****************************************************/
#include <avr/io.h>
#include "FreeRTOS.h"
#include "task.h"
#include "led.h"

void vLEDFlashTask1( void *pvParameters )
{
portTickType xLastWakeTime;
xLastWakeTime=xTaskGetTickCount();
  while(1)
  {
    toggleLED(0);
    vTaskDelayUntil(&xLastWakeTime,1000);
  }
}

void vLEDFlashTask2( void *pvParameters )
{
portTickType xLastWakeTime;
xLastWakeTime=xTaskGetTickCount();
  while(1)
  {
    toggleLED(1);
    vTaskDelayUntil(&xLastWakeTime,500);
  }
}

int main(void)
{
  initLEDport();
  xTaskCreate( vLEDFlashTask1, ( signed char * ) "LED1", configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY, NULL );
  xTaskCreate( vLEDFlashTask2, ( signed char * ) "LED2", configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY, NULL );
  vTaskStartScheduler();
  while(1)
  {}
}
```

Test the program at STK500, and then make some changes (for example create more tasks with other functionalities).

Exercise, Part 2

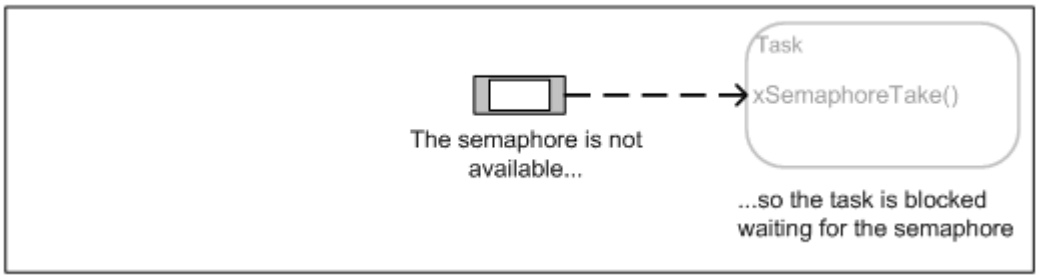
Extent the demo program (from part 1) with 2 more tasks.

Also a binary semaphore has to be declared:

```
xSemaphoreHandle xSemaphore1 = NULL;
```

- The first (extra) task shall wait for switch SW0 to be pressed.
  When this is the case, the semaphore shall be given.
  Use the switch port driver to read the switch.

- The second task shall wait for the semaphore (take it).
  When taken, LED7 shall flash (on for a few milliseconds).
  Use the LED driver to control LED7.



Exercise, Part 3

Write a program having 3 tasks.

Also the program has a global variable called "count" (type unsigned char), being a common resource for two of the tasks. Therefore this variable has to be protected using a (binary) semaphore!

A queue for 10 bytes is used for data transfer between the tasks.

- The first task shall decrement "count" when switch SW0 is pressed.
  Then the value of "count" shall be put onto the queue.

- The second task shall increment "count" when SW1 is pressed.
  Then the value of "count" shall be put onto the queue.

- The third task must receive items from the queue.
  Each time a new item is received, the value of the element shall be displayed at the LEDs (use the LED driver). Also the same value shall be sent as a text string to an attached terminal (use the UART driver function SendInteger()).

- Use an interrupt (for example external INT0 edge-triggered) to reset "count" to 0.
  Also the value 0 shall be put onto the queue.
  Remember to use special FreeRTOS functions when called form an ISR.

Hints:
Declaration part:

```
unsigned char count = 0;
xSemaphoreHandle xSemaphoreCount = NULL;
xQueueHandle xQueue1 = NULL;
```

Part of initialization:

```
initSwitchPort();
initLEDport();
InitUART(115200,8);
// Create semaphore for protecting the "count" variable
vSemaphoreCreateBinary( xSemaphoreCount );
// Create a queue capable of containing 10 unsigned char values.
xQueue1 = xQueueCreate( 10, sizeof(unsigned char) );
```

Note: <queue.h> must be included.