Digital electronics

J.-M Friedt

FEMTO-ST/time & frequency department

jmfriedt@femto-st.fr

slides at `jmfriedt.free.fr`

January 18, 2021

Plan

7 lessons/lab sessions (4-hour long schedules):

1. Executive environments: principles and introduction, getting started with FreeRTOS
2. FreeRTOS, RTEMS, Nuttx ... multitasking and associated methods to make sure shared data and resources are kept in known states (mutex & semaphore)
3. Using the scheduler, mutex and semaphores to solve the "philosopher problem"
4. Embedded systems with GNU/Linux – POSIX compatible operating system
   Architecture of an operating system, kernel v.s userspace
   Internet connectivity and networking
5. Accessing hardware resources from userspace – memory translation from physical to virtual address space (Memory Management Unit) – `/dev/mem`
6. Accessing hardware resources from a web server – internet connected instrument
7. From userspace to kernel space: character device (*char device*) for communicating between users and the kernel

Sharing resources: mutex

```
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
#include "semphr.h"
#include "croutine.h"

int global=0;
xSemaphoreHandle xMutex;

void task_rx(void* p)
{   char aff[10];
    char *t=(char*)p;
    int myInt = 0;
    volatile int local;
    for (myInt=0;myInt<8;myInt++)
    { xSemaphoreTake( xMutex, portMAX_DELAY );
      local=global;  // charge la variable
      local++;
      uart_puts(t);  // operation qui prend du temps
      global=local;  // stocke le resultat

      xSemaphoreGive( xMutex );
      vTaskDelay( ( rand() & 0x5 ) );
    }
    aff[0]=' ';aff[1]=global+'0';aff[2]=' ';aff[3]=0;uart_puts(aff);
    while (1) vTaskDelay( ( rand() & 0x5 ) );
}

int main()
{Led_Init(); Usart1_Init(); srand( 567 );
 xMutex = xSemaphoreCreateMutex();
 xTaskCreate(task_rx, (signed char*)"t1", 100, "1111111111111111111111111111111111111111\r→
       ↪\n\0", 1, 0);
 xTaskCreate(task_rx, (signed char*)"t2", 100, "2222222222222222222222222222222222222222\r→
       ↪\n\0", 1, 0);
 vTaskStartScheduler();
 while(1);
 return 0;
}
```

Sharing resources: semaphore

```
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
#include "common.h"
int globale=0;
xSemaphoreHandle event_signal;

void task1(void* p)
{ while (1) {if (xSemaphoreTake(event_signal,500/portTICK_RATE_MS)==pdFALSE)
                uart_puts("not available\r\n\0");
             else uart_puts("sem take\r\n\0");
             while (globale==0); // vTaskDelay( 1/portTICK_RATE_MS );
             globale=0;
             uart_puts("sem take\r\n\0");
            }
}

void task2(void* p)
{
  while (1) {xSemaphoreGive(event_signal); // debloque tache 1
             globale=1;
             uart_puts("sem give\r\n\0");
             vTaskDelay(400/portTICK_RATE_MS );  // remplace 400 /w 700 !
  }
}

int main()
{Usart1_Init();
 uart_puts("depart\r\n\0");
 vSemaphoreCreateBinary( event_signal );   // Create the semaphore
 xSemaphoreTake(event_signal, 0);          // Take semaphore after creating it.
 xTaskCreate(task1, (signed char*)"t1", (256), 0, 2, 0);
 xTaskCreate(task2, (signed char*)"t2", (256), 0, 1, 0);
 vTaskStartScheduler();
 while(1) {}; return 0;
}
```

- Be wary of low-priority producers starving high-priority consumers ... might block execution (deadlock)

Sharing resources: semaphore

```
#include "FreeRTOS.h"
#include "task.h"
#include "semphr.h"
#include "common.h"
int globale=0;
xSemaphoreHandle event_signal;

void task1(void* p)
{ while (1) {if (xSemaphoreTake(event_signal,500/portTICK_RATE_MS)==pdFALSE)
                uart_puts("not available\r\n\0");
             else uart_puts("sem take\r\n\0");
             while (globale==0); // vTaskDelay( 1/portTICK_RATE_MS );
             globale=0;
             uart_puts("sem take\r\n\0");
            }
}

void task2(void* p)
{
  while (1) {xSemaphoreGive(event_signal); // debloque tache 1
             globale=1;
             uart_puts("sem give\r\n\0");
             vTaskDelay(400/portTICK_RATE_MS );  // remplace 400 /w 700 !
  }
}

int main()
{Usart1_Init();
 uart_puts("depart\r\n\0");
 vSemaphoreCreateBinary( event_signal );   // Create the semaphore
 xSemaphoreTake(event_signal, 0);          // Take semaphore after creating it.
 xTaskCreate(task1, (signed char*)"t1", (256), 0, 2, 0);
 xTaskCreate(task2, (signed char*)"t2", (256), 0, 1, 0);
 vTaskStartScheduler();
 while(1) {}; return 0;
}
```

- Might not be obvious in case of dynamically allocated priority

Thread/fork

The challenge of sharing resources is valid for any multitasking environment

- changing context is a computationally intensive operation [1]
- **thread**: uses the same resources than the father but has been allocated its own stack ⇒ share resources and eliminates the challenge of communication between processes (⇒ risk of data corruption)
- **fork**: duplicate a process by inheriting all attributes from the father ⇒ own virtual memory ⇒ no problem of data corruption and ownership
- for systems with no Memory Management Unit (MMU), `vfork` as used in uClinux

[1] `https://computing.llnl.gov/tutorials/pthreads/`

Threads

A father creates many sons running the same task (e.g. server – answering connection requests over the network interface)

- These sons share resources/variables
- Handling multiple clients/windows in servers or graphical interfaces.
- Method to divide a process in independent sub-tasks
- Consistent access to variabales shared by all threads: mutex, binary version of semaphores [2]

`thread` implementation under GNU/Linux: the `pthread` library
$\Rightarrow$ compile with `-lpthread` linker option

[2] http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html

Thread/mutex

```c
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>

#define NTHREADS 10

void *thread_function(void *);
pthread_mutex_t mutex1 = PTHREAD_MUTEX_INITIALIZER;
int  counter = 0;

int main()
{int d[10];
 pthread_t thread_id[NTHREADS];
 int i, j;

 for (i=0; i < NTHREADS; i++)
     {d[i]=i;
      pthread_create( &thread_id[i], NULL, thread_function, &d[i] );} // CREATION THREADS
 for(j=0; j < NTHREADS; j++) {pthread_join( thread_id[j], NULL);}     // ATTENTE MORT THREADS
 printf("Final counter value: %d\n", counter);
 return(0);
}

void *thread_function(void *d)
{printf("Thread number %d: %lx\n", *(int *)d, pthread_self());
 pthread_mutex_lock( &mutex1 );
 usleep(500000); // 500 ms
 counter++;
 pthread_mutex_unlock( &mutex1 );
}
```

What happens if `usleep()` is taken *out of* the mutex?

```
$ time ./thread
...
real    0m5.001s
user    0m0.004s
sys     0m0.000s
```

```
$ time ./thread
...
real    0m0.503s
user    0m0.000s
sys     0m0.000s
```

Example: RTEMS on STM32

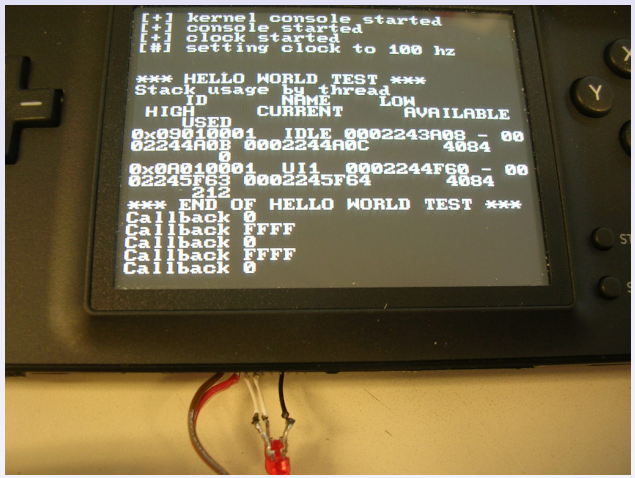
```c
#include <rtems/test.h>
#include <bsp.h> /* for device driver prototypes */
#include <stdio.h>
#include <stdlib.h>

const char rtems_test_name[] = "HELLO WORLD";

rtems_task Init(
  rtems_task_argument ignored
)
{
  rtems_test_begin();
  printf( "Hello World\n" );
  rtems_test_end();
  exit( 0 );
}

#define CONFIGURE_APPLICATION_DOES_NOT_NEED_CLOCK_DRIVER
#define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER

#define CONFIGURE_MAXIMUM_TASKS             1
#define CONFIGURE_USE_DEVFS_AS_BASE_FILESYSTEM

#define CONFIGURE_RTEMS_INIT_TASKS_TABLE

#define CONFIGURE_INITIAL_EXTENSIONS RTEMS_TEST_INITIAL_EXTENSION

#define CONFIGURE_INIT
#include <rtems/confdefs.h>
```

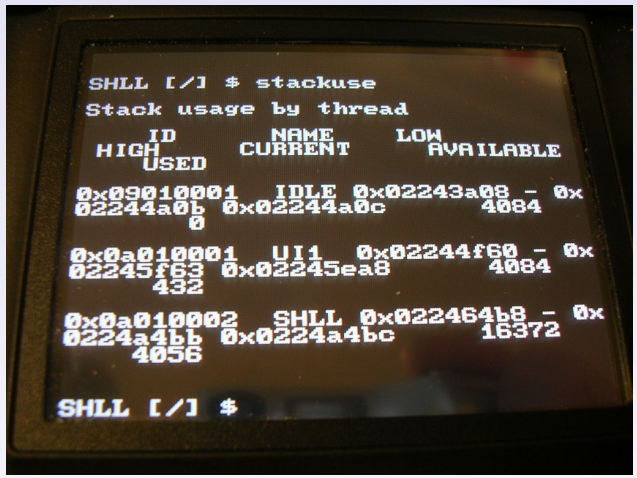
70536 arm-rtems4.11/c/stm32f105rc/testsuites/samples/hello/hello.bin[3]

[3] http://wiki.rtems.com/wiki/index.php/STM32F105

Example: RTEMS on NDS

- POSIX emulation layer for RTEMS: en working framework close to the familiar Unix environment
- hardware access abstraction: framebuffer, network (WiFi)
- TCP/IP stack for high bandwidth communication
- efficient use of available resources (4 MB RAM)



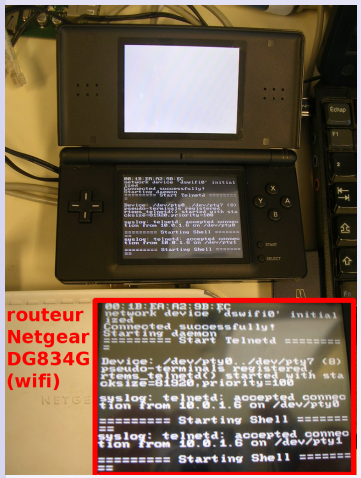


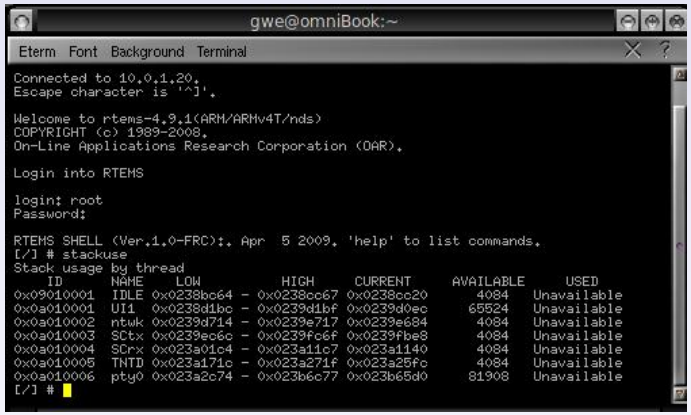
BSP (Board Support Package) RTEMS ported to the NDS by M. Bucchianeri, B. Ratier, R. Voltz and C. Gestes

`http://www.rtems.com/ftp/pub/rtems/current_contrib/nds-bsp/manual.html`

Example: RTEMS on NDS

- POSIX emulation layer for RTEMS: en working framework close to the familiar Unix environment
- hardware access abstraction: framebuffer, network (WiFi)
- TCP/IP stack for high bandwidth communication
- efficient use of available resources (4 MB RAM)





BSP (Board Support Package) RTEMS ported to the NDS by M. Bucchianeri, B. Ratier, R. Voltz and C. Gestes

`http://www.rtems.com/ftp/pub/rtems/current_contrib/nds-bsp/manual.html`

Example RTEMS: framebuffer

An RTEMS application must declare the resources it will use

```
#include <bsp.h>
#include <rtems/fb.h>
#include <rtems/console.h>
#include <rtems/clockdrv.h>
#include "fb.h"

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <rtems/mw_fb.h>

static struct fb_screeninfo fb_info;

inline void draw_pixel(int x, int y, int color)
{
  uint16_t* loc = fb_info.smem_start;
  loc += y * fb_info.xres + x;

  *loc = color;    // 5R, 5G, 5B
  *loc |= 1 << 15; // transparence ?
}

void draw_ppm()
{int x,y,bpp;char r,g,b;
 int l=0;
 for (y=0;y<161;y++)
   for (x=0;x<296;x++) {
     r=image[l++];g=image[l++];b=image[l++];
     bpp=(((int)(r&0xf8))>>3)+(((int)(g&0xf8))<<2)+ \
         (((int)(b&0xf8))<<8);
     if (x<256) draw_pixel(x, y, bpp);
   }
}
```

```
rtems_task Init(rtems_task_argument ignored)
{
  struct fb_exec_function exec;
  int fd = open("/dev/fb0", O_RDWR);

  if (fd < 0) {printk("failed\n");exit(0);}

  exec.func_no = FB_FUNC_ENTER_GRAPHICS;
  ioctl(fd, FB_EXEC_FUNCTION, (void*)&exec);
  ioctl(fd, FB_SCREENINFO, (void*)&fb_info);

  draw_ppm();
  while (1) ;

  exec.func_no = FB_FUNC_EXIT_GRAPHICS;
  ioctl(fd, FB_EXEC_FUNCTION, (void*)&exec);
  close(fd);printk("done.\n");exit(0);
}

/* configuration information */
#define CONFIGURE_HAS_OWN_DEVICE_DRIVER_TABLE

rtems_driver_address_table Device_drivers[] =
{ CONSOLE_DRIVER_TABLE_ENTRY,
  CLOCK_DRIVER_TABLE_ENTRY,
  FB_DRIVER_TABLE_ENTRY,
  { NULL,NULL, NULL,NULL,NULL, NULL }
};

#define CONFIGURE_LIBIO_MAXIMUM_FILE_DESCRIPTORS      10
#define CONFIGURE_RTEMS_INIT_TASKS_TABLE
#define CONFIGURE_MAXIMUM_TASKS 1
#define CONFIGURE_INIT
#include <rtems/confdefs.h>
```

Mixes abstraction ("/dev/fb0") and low level access ("*loc=color").

Example RTEMS: GPIO

Low level access: "(*(volatile uint16_t*)0x08000000)=l;"

```
#include <bsp.h>
#include <stdlib.h>
#include <stdio.h>
#include <nds/memory.h>

rtems_id timer_id;
uint16_t l=0;

void callback()
{ printk("Callback %x\n",l);
  (*(volatile uint16_t*)0x08000000)=l;
  l=0xffff-l;
  rtems_timer_fire_after(timer_id, 100, callback, NULL);
}

rtems_task Init(rtems_task_argument ignored)
{ rtems_status_code status;
  rtems_name timer_name = rtems_build_name('C','P','U','T');

  printk( "\n\n*** HELLO WORLD TEST ***\n" );
// sysSetCartOwner(BUS_OWNER_ARM9);
  (*(vuint16*)0x04000204) = ((*(vuint16*)0x04000204) \
         & ~ARM7_OWNS_ROM);

  status = rtems_timer_create(timer_name,&timer_id);
  rtems_timer_fire_after(timer_id, 1, callback, NULL);
  rtems_stack_checker_report_usage();
           // requires #define CONFIGURE_INIT

  printk( "*** END OF HELLO WORLD TEST ***\n" );
  while(1)
     ;
  exit( 0 );
}

/* configuration information */
#define CONFIGURE_APPLICATION_NEEDS_CONSOLE_DRIVER
#define CONFIGURE_APPLICATION_NEEDS_CLOCK_DRIVER
#define CONFIGURE_RTEMS_INIT_TASKS_TABLE

/* configuration information */
#define CONFIGURE_MAXIMUM_DEVICES                     40
#define CONFIGURE_MAXIMUM_TASKS                       100
#define CONFIGURE_MAXIMUM_TIMERS                      32
#define CONFIGURE_MAXIMUM_SEMAPHORES                  100
#define CONFIGURE_MAXIMUM_MESSAGE_QUEUES              20
#define CONFIGURE_MAXIMUM_PARTITIONS                  100
#define CONFIGURE_MAXIMUM_REGIONS                     100
/* This seetings overwrite the ones defined in confdefs.h */
#define CONFIGURE_MAXIMUM_POSIX_MUTEXES               32
#define CONFIGURE_MAXIMUM_POSIX_CONDITION_VARIABLES   32
#define CONFIGURE_MAXIMUM_POSIX_KEYS                  32
#define CONFIGURE_MAXIMUM_POSIX_QUEUED_SIGNALS        10
#define CONFIGURE_MAXIMUM_POSIX_THREADS               128
#define CONFIGURE_MAXIMUM_POSIX_TIMERS                10
#define CONFIGURE_LIBIO_MAXIMUM_FILE_DESCRIPTORS      200

#define STACK_CHECKER_ON
#define CONFIGURE_INIT

#include <rtems/confdefs.h>
```

J.-M Friedt, G. Goavec-Merou, *Interfaces matérielles et OS libres pour Nintendo DS : DSLinux et RTEMS*, GNU/Linux Magazine France Hors Série 43, aout 2009

Practical demonstration

FreeRTOS on STM32 : `github.com/jmfriedt/tp_freertos`

- using the cross-compilation toolchain (`arm-...-gcc`) and tools for transferring the binary file to the microcontroller (baremetal C examples)
- specificity of the STM32 architecture: high granularity of the clock configuration
- 32 bit microcontroller able to run an executive environment
- many version of this microcontroller core: the *linker* script defines what memory resources are available (`ld/` directory).
- datasheet ... core manual [4] and processor datasheet [5]
- FreeRTOS on STM32: an embedded "OS" providing functionalitites such as tasks, scheduler, mutex, semaphores stack corruption and process status
- runs with qemu (`https://github.com/beckus/qemu_stm32`):

Table 5.High-density STM32F103xx pin definitions (continued)

Pins						Pin name	Type[1]	I / O Level[2]	Main function[3] (after reset)	Alternate functions[4]	
LFBGA144	LFBGA100	WLCSP64	LQFP64	LQFP100	LQFP144					Default	Remap
K1	H1	-	-	20	31	V_{REF-}	S		V_{REF-}		
L1	J1	F7 (8)	-	21	32	V_{REF+}	S		V_{REF+}		
M1	K1	G8	13	22	33	V_{DDA}	S		V_{DDA}		
J2	G2	F6	14	23	34	PA0-WKUP	I/O		PA0	WKUP/USART2_CTS[9] ADC123_IN0 TIM2_CH1_ETR TIM5_CH1/TIM8_ETR	
K2	H2	E6	15	24	35	PA1	I/O		PA1	USART2_RTS[9] ADC123_IN1/ TIM5_CH2/TIM2_CH2[9]	
L2	J2	H8	16	25	36	PA2	I/O		PA2	USART2_TX[9]/TIM5_CH3 ADC123_IN2/ TIM2_CH3 [9]	
M2	K2	G7	17	26	37	PA3	I/O		PA3	USART2_RX[9]/TIM5_CH4 ADC123_IN3/TIM2_CH4[9]	
G4	E4	F5	18	27	38	V_{SS_4}	S		V_{SS_4}		
F4	F4	G6	19	28	39	V_{DD_4}	S		V_{DD_4}		

```
qemu_stm32/arm-softmmu/qemu-system-arm -M stm32-p103 -serial stdio \
    -serial stdio -serial stdio -kernel usart_cm3.bin
```

[4] `http://www.st.com/web/en/resource/technical/document/reference_manual/CD00171190.pdf`

[5] `http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00191185.pdf`