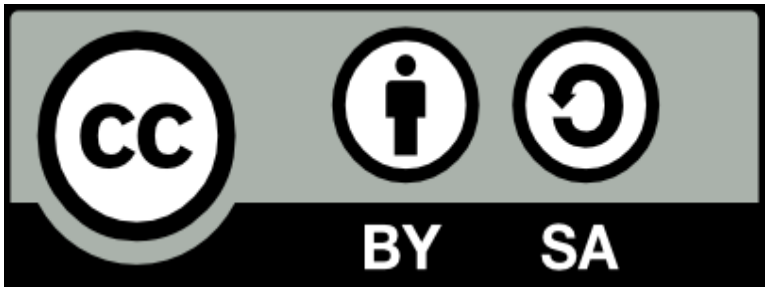
FreeRTOS

An introduction

Atilla Filiz



whoami

Atilla Filiz

Embedded Software Engineer
Essensium - Mind

http://mind.be

Contents

- Introduction
- RT Basics
- Configuration
- API
  - Tasks
  - Hooks
  - Interrupts and semaphores
  - Delaying
- Licensing and flavours

Contents

- Introduction
- RT Basics
- Configuration
- API
  - Tasks
  - Hooks
  - Interrupts and semaphores
  - Delaying
- Licensing and flavours

Introduction

FreeRTOS

- Is A lightweight operating system.
- Is Designed to run on microcontrollers.
- Supports memory protection
  - But not TLBs
- Based on absolute priorities
- A lot of features are optional (even preemption!)
- Configuration via single C header: FreeRTOSConfig.h

Why?

Why not?

- SAM4E16E runs at 120MHz
  - Can handle many tasks
  - Multiple tasks → Need scheduling
  - → Need sync/communication

Real-time systems

- Tasks have deadlines
- Tasks may or may not be periodic
- Key: managing priorities
  - May be importance based
  - May be deadline based (mathematically optimal)
- Predictability is crucial!

RTOS Basics

- FreeRTOS does not magically make a system real-time.
- FreeRTOS gives a mechanism to
  - Prioritize: Higher priority task always preempts lower
  - Timing: Delay with a fixed period
- Combine the two above to design a real-time system
- Use queues to pass messages between tasks, use semaphores to manage resources.
- Ports: HW specific code → its own board dir

FreeRTOS Feature summary

- Pre-emptive or co-operative operation
- Very flexible task priority assignment
- Queues
- Binary semaphores
- Counting semaphores
- Recursive semaphores
- Mutexes
- Tick hook functions
- Idle hook functions
- Stack overflow checking
- Trace hook macros

Configuration

FreeRTOSConfig.h

```
#define configUSE_PREEMPTION                    1
#define configUSE_IDLE_HOOK                     1
#define configUSE_TICK_HOOK                     0
#define configTICK_RATE_HZ                      ( ( portTickType ) 1000 )
#define configMINIMAL_STACK_SIZE                ( ( unsigned portSHORT ) 4 )
#define configTOTAL_HEAP_SIZE                   ( ( size_t ) ( 32 * 1024 ) )
#define configMAX_TASK_NAME_LEN                 ( 16 )
#define configUSE_TRACE_FACILITY                1
#define configUSE_16_BIT_TICKS                  0
#define configIDLE_SHOULD_YIELD                 1
#define configUSE_CO_ROUTINES                   0
#define configUSE_MUTEXES                       1
…
```

API: Tasks

Minimal main()

```
int main(void)
{
    sysclk_init();
    board_init();
    vTaskStartScheduler();
    While(1); //will never reach here
}
```

API: Tasks

```
static void myTask( void *pvParameters );
...

main(){
   ...
       xTaskCreate( myTask, "My Task",
   configMINIMAL_STACK_SIZE, NULL, /* parameter*/
   mainPRINT_TASK_PRIORITY, NULL /* task handle
   */);
...

    vTaskStartScheduler();
```

API: Hooks

- Defined as weak symbols, just redefine and use them

- void vApplicationIdleHook(void);
- void vApplicationTickHook(void);
- void vApplicationMallocFailedHook(void);
- void vApplicationStackOverflowHook(TaskHandle_t xTask,
- signed char *pcTaskName);

API: Interrupts and Semaphores

- vSemaphoreCreateBinary(xSemaphore)
- xSemaphoreGive(xSemaphore);
- xSemaphoreGiveFromISR(xSemaphore, pxHigherPriorityTaskWoken) // type boolean*
- void vPortYieldFromISR(void)
- xSemaphoreTake(xSemaphore, xBlockTime)
- Similar with queues (send, receive, ...fromISR)

API: Delaying

- void vTaskDelay(const TickType_t xTicksToDelay);
  // Just delay a number of ticks

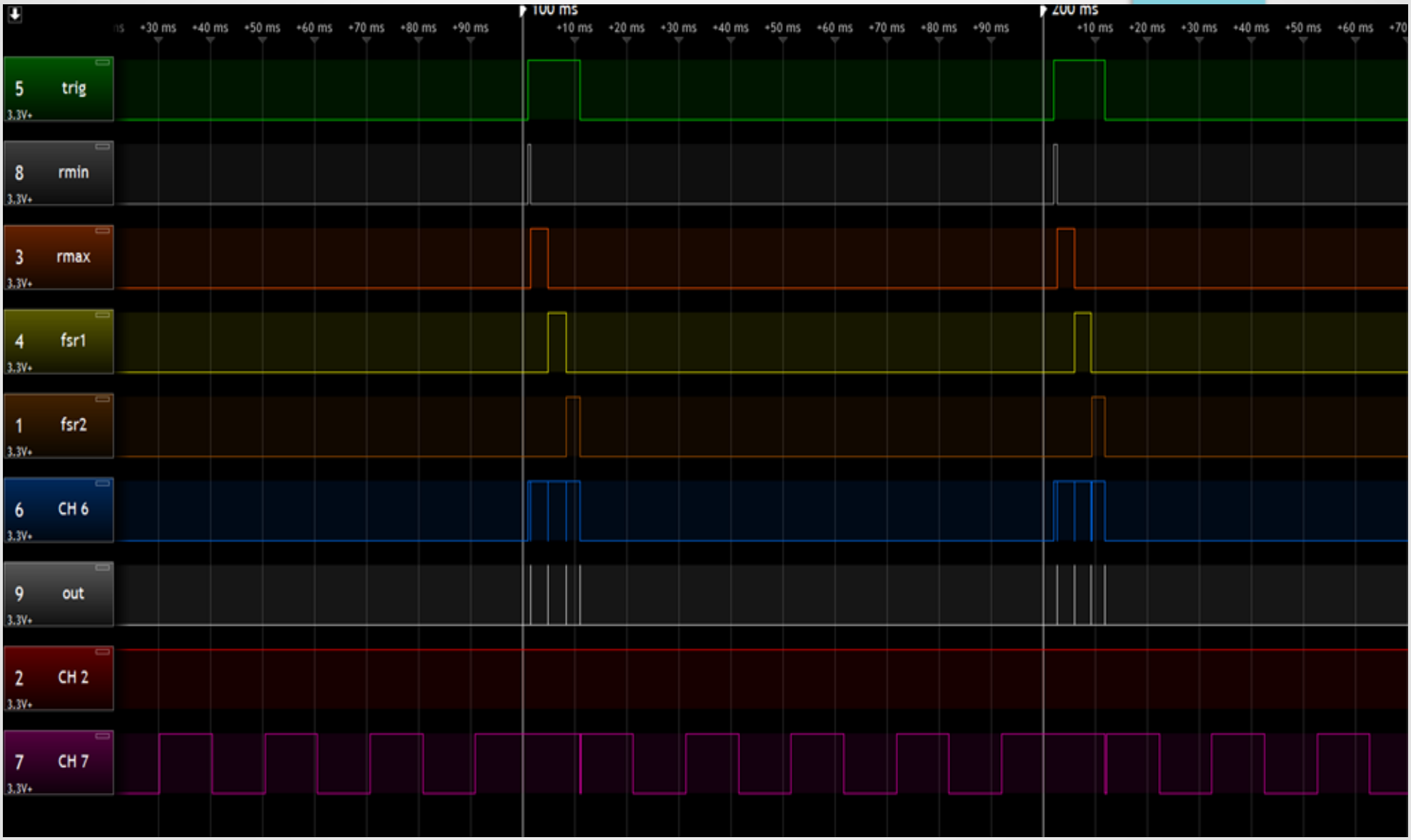
- void vTaskDelayUntil(TickType_t *pxPreviousWakeTime,
- const TickType_t xTimeIncrement);

  // Magic delay function that increments own first parameter

API: Delaying

```
// Perform an action every 10 ticks.
 void vTaskFunction( void * pvParameters )
 {
 TickType_t xLastWakeTime;
 const TickType_t xFrequency = 10;
    // Initialise the xLastWakeTime variable with the current time.
    xLastWakeTime = xTaskGetTickCount();
    for( ;; )
    {
        // Wait for the next cycle.
        vTaskDelayUntil( &xLastWakeTime, xFrequency );
        // Perform action here.
    }
 }
```

Delaying


100 ms
200 ms
5 trig
8 rmin
3 rmax
4 fsr1
1 fsr2
6 CH 6
9 out
2 CH 2
7 CH 7
3.3V+

License and Flavours

“Modified GPL”

- Change FreeRTOS → Must release source
- Link with own app → Can keep app secret
- FreeRTOS part should still be provided with same license
- Usage of FreeRTOS must be mentioned
- FreeRTOS+
- Open source but not free.
- OpenRTOS
- Full “commercial” licence to keep all propriatery
- SafeRTOS
- With safety certifications

Questions?

www.mind.be

www.essensium.com

Essensium NV
Mind - Embedded Software Division
Gaston Geenslaan 9, B-3001 Leuven
Tel : +32 16-28 65 00
Fax : +32 16-28 65 01
email : info@essensium.com

PC Demo

- Download http://interactive.freertos.org/attachments/token/r6d5gt3998niuc4/?name=Posix_GCC_Simulator_6.0.4.zip
- Go to Release
- Make all
- Add -lrt to the linker flags if necessary