1 Section: Introducing the
2 Section: Creating a polle
3 Section: Queue-based dri
4 Section: A buffer-based d
5 Section: Configuring DM
6 Section: A buffer-based d
7 Section: Stream buffers

3 Section: Queue-based driver (page 249)

- **Clarification**, page 250-254
  - There's a typo regarding the baud-rate used. The book says it's 9,600, but in the distributed version of `mainUartInterruptQueue.c`, the baud-rate is 256,400.

- **Additional info**, page 252
  - This section describes how the address for `USART2_IRQHandler()` is put in the ISR vector-table.
  - This link explains how the address for `USART2_IRQHandler()` is put in the ISR vector-table:
    - https://electronics.stackexchange.com/questions/425202/arm-cortex-m3-interrupts-and-interrupt-service-routine-interrupt-handler
  - The data-structures related to the ISR vector-table are:
    - The ISR vector-table is defined in:
      - `Drivers\CMSIS\Device\ST\STM32F7xx\Include\stm32f767xx.h`
      - There is a vector-table entry for `USART2_IRQHandler()`
      - There are also linker entries for `USART2_IRQHandler()`:

```
.weak      USART2_IRQHandler
.thumb_set USART2_IRQHandler,Default_Handler
```

    - In the ISR vector-table, the index for `USART2_IRQHandler()` is the USART2 interrupt-number. The interrupt number is defined in:
      - `Drivers\CMSIS\Device\ST\STM32F7xx\Include\stm32f767xx.h`:

```
USART2_IRQn                 = 38,     /*!< USART2 global Interrupt
```

- **Clarification**, page 252
  - This paragraph was hard to follow:
    - "*The xHigherPriorityTaskWoken variable is initialized to false....*"
  - The FreeRTOS documentation clarified it for me. For `xQueueSendFromISR()`, the third parameter is `pxHigherPriorityTaskWoken`, and it's described as:
    - *xQueueSendFromISR() will set *pxHigherPriorityTaskWoken to pdTRUE if sending to the queue caused a task to unblock, and the unblocked task has a priority higher than the currently running task. If xQueueSendFromISR() sets this value to pdTRUE then a context switch should be requested before the interrupt is exited.*

- **Additional info**, page 254
  - The app `mainUartInterruptQueue.c` consists of over 30 function calls in total, and many are system APIs. The following table shows the functions used, and what file and library provides them.
  - Chapter 10 includes a good section on using those libraries. The section is, "Using third-party libraries (STM HAL)" on page 285.
  - The app consists of three program files:
    - `BSP\UartQuickDirtyInit.c`
    - `Chapter_10/Src/mainUartInterruptQueue.c`
    - `Chapter_10\Src\Uart4Setup.c`
  - The system functions used are from five libraries:
    - Arm
    - FreeRTOS
    - SEGGER SystemView
    - STMicroelectronics
    - The book
      - The following files appear to be config-files that were copied from other libraries, and modified:
      - `BSP\Nucleo_F767ZI_Init.c`
      - `Chapter_10\Inc\stm32f7xx_hal_conf.h`
        - See STM's "*User Manual : Description of STM32F7 HAL and low-layer drivers*"

Function/macro provider	Function/macro	Containing file
Arm	NVIC_EnableIRQ	Drivers\CMSIS\Include\core_cm7.h
Arm	NVIC_SetPriority	Drivers\CMSIS\Include\core_cm7.h
Arm	NVIC_SetPriorityGrouping	Drivers\CMSIS\Include\core_cm7.h
book	HWInit	BSP\Nucleo_F767ZI_Init.c
book	initUart2Pins	BSP\UartQuickDirtyInit.c
book	initUart4Pins	BSP\UartQuickDirtyInit.c
book	STM_UartInit	BSP\UartQuickDirtyInit.c
book	startReceiveInt	Chapter_10/Src/mainUartInterruptQueue.c
book	startUart4Traffic	Chapter_10/Src/mainUartInterruptQueue.c
book	uartPrintOutTask	Chapter_10/Src/mainUartInterruptQueue.c
book	USART2_IRQHandler	Chapter_10/Src/mainUartInterruptQueue.c