- Possible solutions, for concurrent operations on the `flag` variable:
  - A mutex could be used, when accessing `flag`.
  - ARM does have atomic instructions, which might work here.
    - http://www.doulos.com/knowhow/arm-embedded/implementing-semaphores-on-arm-processors/
    - https://stackoverflow.com/questions/11894059/atomic-operations-in-arm

```
  flag = 1;  //set 'flag' to 1 to "signal" BlueTaskB to run
0800454A   LDR            R7, =flag                   ; [PC, #48] [0x0800457C] =0x20005310
0800455C   MOVS           R3, #1
0800455E   STR            R3, [R7, #0]
```

```
  while(!flag){}
0800458A   LDR            R5, =flag                   ; [PC, #60] [0x080045C8] =0x20005310
08004596   LDR            R3, [R5, #0]
08004598   CMP            R3, #0
0800459A   BEQ            0x08004596                  ; <BlueTaskB>+0x12
```

- **Bug** in the code (`mainPolledExample.c`), page 183
  - Problem (minor bug):
    - There's a compiler warning for `mainPolledExample.c`, for line 124, which is:
      - `while(!flag);`
    - Warning message:

```
../Src/mainPolledExample.c:124:6: warning: this 'while' clause does not guard...
[-Wmisleading-indentation]
      while(!flag);
      ^~~~~
```

  - Solution: `while(!flag){}`

- **Clarification**, page 185
  - The program described here is `mainSemTimeBound.c`

- **Clarification** (`mainSemTimeBound.c`), page 185
  - `StmRand(3,7);`
    - Returns a random number between 3 and 7, inclusive.
  - The only documentation I could find for `StmRnd()` is from STM32CubeIDE. It can be accessed by right-clicking on the function, and selecting "`Open Definition`."

- **Additional info** (`mainSemTimeBound.c`), page 185
  - In `GreenTaskA`, `numLoops` is changed on every iteration of the loop. This makes it non-trivial to figure-out expected behavior:
    - How much time is there between calls to `xSemaphoreGive()`?
    - How often will `xSemaphoreTake()` time-out?
  - Results from using SystemView, and running the program for over 2 minutes:
    - `xSemaphoreTake()` timed-out 40% of the time (107 times), and succeeded 60% of the time (145 times).

- **Clarification** (`mainSemTimeBound.c`), page 185-186
  - The code shown on page 185 differs from the code on GitHub (`mainSemTimeBound.c`). The code on GitHub has calls to SEGGER_SYSVIEW_PrintfHost(), and those calls are not in the code on page 185.
  - The figure on page 186 was created from running the code on GitHub. It shows the messages from calling SEGGER_SYSVIEW_PrintfHost().

- **Bug** in the book, page 186
  - There are two related errors in the figure:
  - The events shown in the Terminal window are not the same events shown in the Events-List and Timeline windows. The events in Terminal window have different timestamps than the events in the other two windows.
  - In the Terminal window, the events labeled "2" and "3" do not correspond with the events labeled "2" and "3" in the Timeline window. For each window, the time between those events is very