RTOS Debug Demo Notes

1. Starting point:
   1. Configuration
      1. threads.h
         1. THREADS\_ARE\_SAME\_PRIORITY = 1
         2. DEMO\_DEFAULT\_READ\_ACCEL\_STK\_SZ = 1
      2. RTX\_Config.h
         1. Default thread stack size 256 bytes
         2. Use Stack overrun checking
         3. Use Stack usage watermark
   2. Result: Program freezes
      1. Load into debugger, run
      2. Check RTX RTOS window
      3. Generic stack overflow problem indicated
2. Increase default stack size to 512 bytes
   1. Configuration
      1. RTX\_Config.h
         1. Default thread stack size 512 bytes
   2. Result: Program freezes later
      1. Load into debugger, run
      2. Stuck in ostxErrorNotify
      3. Check RTX RTOS window: Thread\_Read\_Accel red, Stack Overrun detected
3. Increase default stack size to 768 bytes
   1. Configuration
      1. RTX\_Config.h
         1. Default thread stack size 768 bytes
   2. Result: Program runs
4. Now no audio
   1. Same configuration
   2. Debug:
      1. Load into debugger, run
      2. Set breakpoint in Thread\_Sound\_Manager after osDelay call.
      3. Breakpoint is never hit.
      4. Stop program. Examine RTX RTOS window, expand Threads node:
         1. Thread Sound Manager is present
         2. Thread Refill Sound Buffer is not present
      5. Examine values of thread IDs in threads.c (line 26)
         1. osThreadId\_t values. All are non-zero except t\_Refill\_Sound\_Buffer
      6. Set breakpoint at threads.c:Create\_OS\_Objects
         1. Run to breakpoint
         2. Observe that osThreadNew returns 0 for last thread. Is error indicator.
5. Reduce default stack size, but keep large stack for Thread\_Read\_Accelerometer
   1. Configuration
      1. RTX\_Config.h
         1. Default thread stack size 256 bytes
         2. Use Stack overrun checking
         3. Use Stack usage watermark
      2. threads.h
         1. DEMO\_DEFAULT\_READ\_ACCEL\_STK\_SZ = 0
         2. #define READ\_ACCEL\_STK\_SZ 768
      3. threads.c
         1. Add .stack\_size = READ\_ACCEL\_STK\_SZ to attributes for Read\_Accelerometer
   2. Result: Audio works
6. How much stack space is really needed?
   1. Set Listing options: Linker Listing and Callgraph
   2. Examine Objects/[project].htm
   3. Look for Thread\_ functions
      1. Thread\_Read\_Accelerometer: 152 + ?
         1. Call Chain = Thread\_Read\_Accelerometer ⇒ \_\_2sprintf ⇒ \_printf\_char\_common ⇒ \_\_printf
      2. Thread\_Read\_TS: 136
      3. Thread\_Refill\_Sound\_Buffer: 24
      4. Thread\_Sound\_Manager: 40
      5. Thread\_Update\_Screen: 80
   4. Can put in custom sizes for last four threads
   5. And also look for IRQHandler (
      1. DMA0: 56
      2. PIT: 8
      3. TPM0: 0
   6. Note on ISRs and stacks (See Chapter 4, p.100 – Entering a handler)
      1. CPU pushes 8 registers onto currently used stack (PSP or MSP)
      2. CPU switches to MSP and handler mode. Main stack (MSP, size defined in startup\_..s) must be large enough to hold ISR stack frames