ELEC 2220 Computer Systems
Homework #19
Due: Monday, July 27

The Final Project requires precise timing of signal waveforms to be generated. Therefore, the purpose of this exercise is to design a simple program that sets up and uses timer interrupts to control two LEDs. The program is to operate as follows.

1. The blue LED is to be blinked on and off at a rate of one change per one-half second (on for half a second, off for half a second.) The timing must be precise (hardware timer-controlled).
2. The orange LED is to be blinked on and off at a rate of one change per two seconds (on for two seconds, off for two seconds.) The timing must be precise (hardware timer-controlled).
3. The green LED is to be blinked on and off at a rate of approximately one change per second (on for one second, off for one second.) This timing can be approximate (software-controlled).
4. The timing of the blue and orange LEDs is to be controlled by programmable timer 6 (TIM6). The timer should interrupt the program when it is time to change the blue LED (every half second). The state of the blue LED should be changed by the interrupt handler on every interrupt. The state of the orange LED should be changed every fourth interrupt.
5. The "main program" should continuously execute your previously-designed software delay loop to change the green LED every second.

While testing and debugging, try out the "Watch" and "Logic Analyzer" debug windows (to be demonstrated in class), to observe one or more of your memory variables. Among other things, you should verify the timing of your waveforms. Watch and Logic Analyzer windows are described in STM Application Note 230 (*STMicroelectronics: Cortex-M4 Training STM32F4-Discovery evaluation board using ARM Keil MDK5 toolkit*). Sections 12 and 13 describe Watch windows. Sections 14 and 18 describe setup and use of the Logic Analyzer.

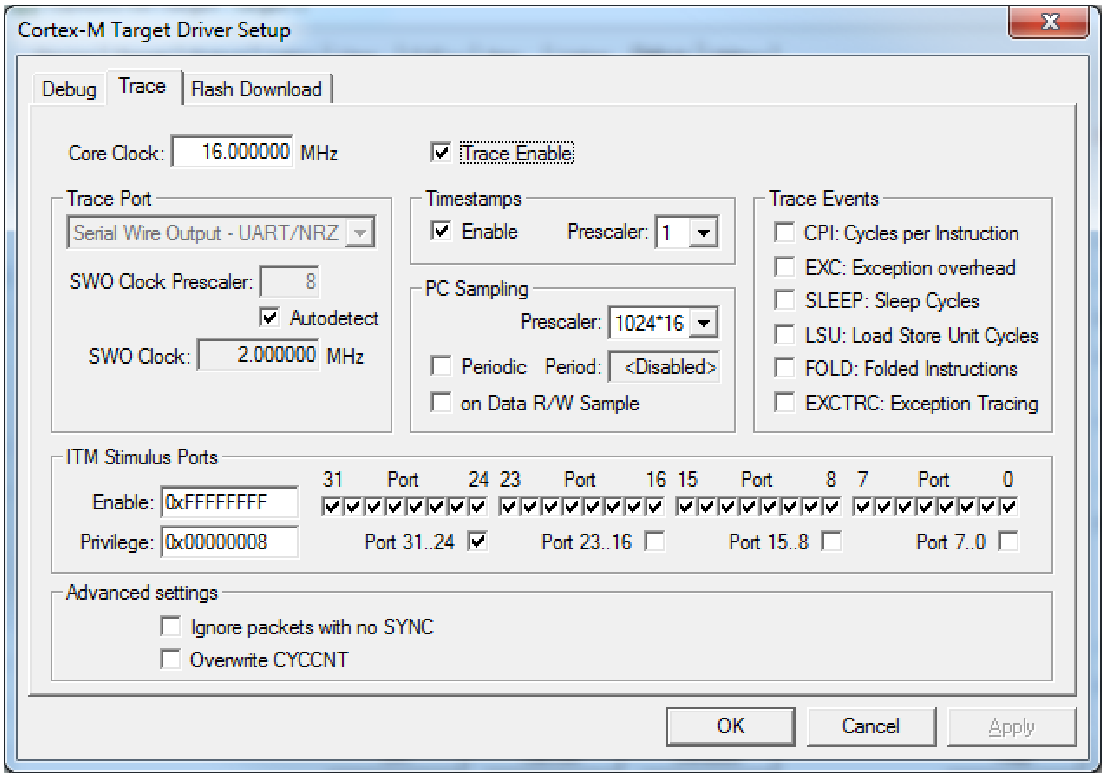
Do the following to enable use of the logic analyzer.

1. Open *Options for Target*, select the Debug tab, click on Settings (next to the box showing ST-Link Debugger), and select the *Trace* tab (see Figure 1 on next page). On this tab, elect Trace Enable, unselect Periodic and EXCTRC, enter 16 as the Core Clock frequency (in MHz), and click OK. Note that the default startup files for our board configure the system clock to run at 16MHz.
2. Copy the information on page 3 of this assignment into a plain text file STM32_SWO.ini, put that in your project directory, and then on the Debug tab (see Figure 2 on the next page) select this as the debugger Initialization File. (Or, you can add this text to your own debugger initialization file.)
3. Refer to the above-mentioned document for instructions on configuring and using the Logic Analyzer.

<u>**Deliverables:**</u>

1. Submit a printout of the source program in class.

2. Demonstrate the operation of the program via a Zoom meeting. Alternatively, you may upload to Canvas or email me a **video** that demonstrates the above behavior. I must see and hear you in the video, describing the operation of the board.



Core Clock: 16.000000 MHz
[✓] Trace Enable

Trace Port: Serial Wire Output - UART/NRZ
SWO Clock Prescaler: 8
[✓] Autodetect
SWO Clock: 2.000000 MHz

Timestamps: [✓] Enable Prescaler: 1

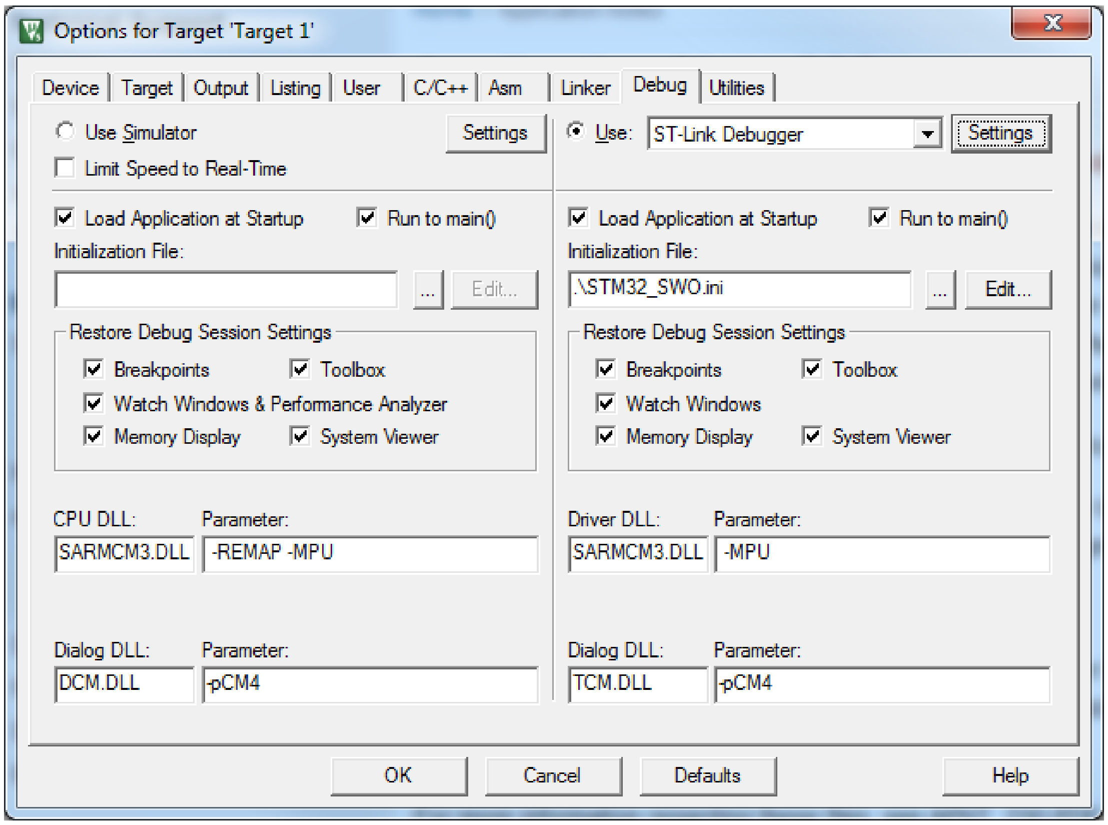
PC Sampling: Prescaler: 1024*16
[ ] Periodic Period: <Disabled>
[ ] on Data R/W Sample

Trace Events:
[ ] CPI: Cycles per Instruction
[ ] EXC: Exception overhead
[ ] SLEEP: Sleep Cycles
[ ] LSU: Load Store Unit Cycles
[ ] FOLD: Folded Instructions
[ ] EXCTRC: Exception Tracing

ITM Stimulus Ports:
Enable: 0xFFFFFFFF
31 Port 24 23 Port 16 15 Port 8 7 Port 0
[✓][✓][✓][✓][✓][✓][✓][✓] [✓][✓][✓][✓][✓][✓][✓][✓] [✓][✓][✓][✓][✓][✓][✓][✓] [✓][✓][✓][✓][✓][✓][✓][✓]
Privilege: 0x00000008
Port 31..24 [✓] Port 23..16 [ ] Port 15..8 [ ] Port 7..0 [ ]

Advanced settings:
[ ] Ignore packets with no SYNC
[ ] Overwrite CYCCNT

Figure 1. Enter Core Clock frequency, select Trace Enable, unselect Periodic and EXCTRC.



( ) Use Simulator
[ ] Limit Speed to Real-Time
[✓] Load Application at Startup [✓] Run to main()
Initialization File:
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows & Performance Analyzer
[✓] Memory Display [✓] System Viewer
CPU DLL: SARMCM3.DLL Parameter: -REMAP -MPU
Dialog DLL: DCM.DLL Parameter: -pCM4

(●) Use: ST-Link Debugger
[✓] Load Application at Startup [✓] Run to main()
Initialization File: .\STM32_SWO.ini
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows
[✓] Memory Display [✓] System Viewer
Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM4

Figure 2. Select Initialization File: STM32_SWO.ini

```
/******************************************************************************/
/* STM32_SWO.ini: STM32 Debugger Initialization File                           */
/******************************************************************************/
// <<< Use Configuration Wizard in Context Menu >>>                           //
/******************************************************************************/
/* This file is part of the uVision/ARM development tools.                    */
/* Copyright (c) 2005-2009 Keil Software. All rights reserved.                */
/* This software may only be used under the terms of a valid, current,        */
/* end user licence from KEIL for a compatible version of KEIL software       */
/* development tools. Nothing else gives you the right to use this software.  */
/******************************************************************************/


FUNC void DebugSetup (void) {
// <h> Debug MCU Configuration
//   <o1.0>    DBG_SLEEP     <i> Debug Sleep Mode
//   <o1.1>    DBG_STOP      <i> Debug Stop Mode
//   <o1.2>    DBG_STANDBY   <i> Debug Standby Mode
//   <o1.5>    TRACE_IOEN    <i> Trace I/O Enable
//   <o1.6..7> TRACE_MODE    <i> Trace Mode
//             <0=> Asynchronous
//             <1=> Synchronous: TRACEDATA Size 1
//             <2=> Synchronous: TRACEDATA Size 2
//             <3=> Synchronous: TRACEDATA Size 4
//   <o1.8>    DBG_IWDG_STOP <i> Independant Watchdog Stopped when Core is halted
//   <o1.9>    DBG_WWDG_STOP <i> Window Watchdog Stopped when Core is halted
//   <o1.10>   DBG_TIM1_STOP <i> Timer 1 Stopped when Core is halted
//   <o1.11>   DBG_TIM2_STOP <i> Timer 2 Stopped when Core is halted
//   <o1.12>   DBG_TIM3_STOP <i> Timer 3 Stopped when Core is halted
//   <o1.13>   DBG_TIM4_STOP <i> Timer 4 Stopped when Core is halted
//   <o1.14>   DBG_CAN_STOP  <i> CAN Stopped when Core is halted
// </h>
  _WDWORD(0xE0042004, 0x00004027);  // DBGMCU_CR
}

DebugSetup();                       // Debugger Setup
```