Lab 11: SysTick

Purpose/Scope: This lab will introduce you to the system timer SysTick on the Cortex-M4 processor, and how it can be utilized to provide a hardware interrupt.

Concepts: Interrupt
Register maps
Peripheral base addresses and register offsets

Answers to pre-lab questions are to be completed before coming to your scheduled lab period, for submittal at the beginning of the lab. Keep a record of your answers to use in the lab.

References:

- DUI0553A – Section 4.4 (pg 4-33)

Note: The 4 pages in the reference above constitute the basis for this lab, and as such, are of great importance for you to thoroughly understand before coming to lab.

It is equally important to use the register and bit field (function) names assigned in the documentation exactly as presented (including capitalization). Since these names are already defined for the Cortex-M4, there is no need to provide further documentation if they are used as defined.

Pre-Lab 11

An extremely important concept for any microprocessors course is the interrupt, both software and hardware. Interrupts are another link to the external world for the microprocessor and are inextricably linked to operating system design. It's time to get down and dirty with this concept.

Please read the following short introduction to the concepts of software and hardware interrupts:

Intro to Interrupts - hardware and software

Now that you know what we are talking about, it's time to actually play with a hardware generated interrupt. In this lab, you will set up the microcontroller to execute code immediately upon receiving pre-programmed, re-occurring, periodic hardware timer events (think of an alarm clock set to wake you up day after day).

In this pre-lab, you will describe the system timer hardware (alarm clock) that will pre-empt the microprocessor's execution of your C main program. Please fill out the information below for the SysTick system timer using the document link on the first page above. This won't take long.

Deliverables

Part A: System Timer Register Names

What is the base address of this peripheral? _0x_E000E010_____
(hint: address of the first register in the peripheral)

Use Section 4.4 of the Cortex-M4 User Guide (link provided above) to find the necessary information on the four system timer registers to fill in the following table:

Register Name	Address	Offset*	Description
SYST_CSR	0xE000E010	+0	SysTick Control and status register
SYST_RVR	0xE000E014	+4	Systick Reload Value Register
SYST_CVR	0xE000E018	+8	SysTick Current value Register
SYST_CALIB	0xE000E01C	+12	SysTick Calibration Value

* Offset from the base address

Part B: System Timer Field Names

Each timer register has one or more bit fields associated with it. Fill in the following table for all nine non-reserved bit fields:

Name	Assoc. Register	Function	Offset*	Size (bits)
COUNTFLAG	SYST_CSR	Returns 1 if timer counted to 0 since last read	16	1
CLKSOURCE	SYST_CSR	Indicates clock source	2	1
TICKINT	SYST_CSR	Enables SysTick exception request	1	1
ENABLE	SYST_CSR	Enables the counter	0	1
RELOAD	SYST_RVR	Value to load into the SYST_CSV register when the counter is enabled and when it reaches 0	0	24
CURRENT	SYST_CVR	Reads return the current value of SysTick counter	0	24
NOREF	SYST_CALIB	Indicates whether device provides a ref clock to processor	31	1
SKEW	SYST_CALIB	Indicates whether the TENMS value is exact	30	1
TENMS	SYST_CALIB	Reload value for 10ms timing subject to sys clock skew errors	0	24