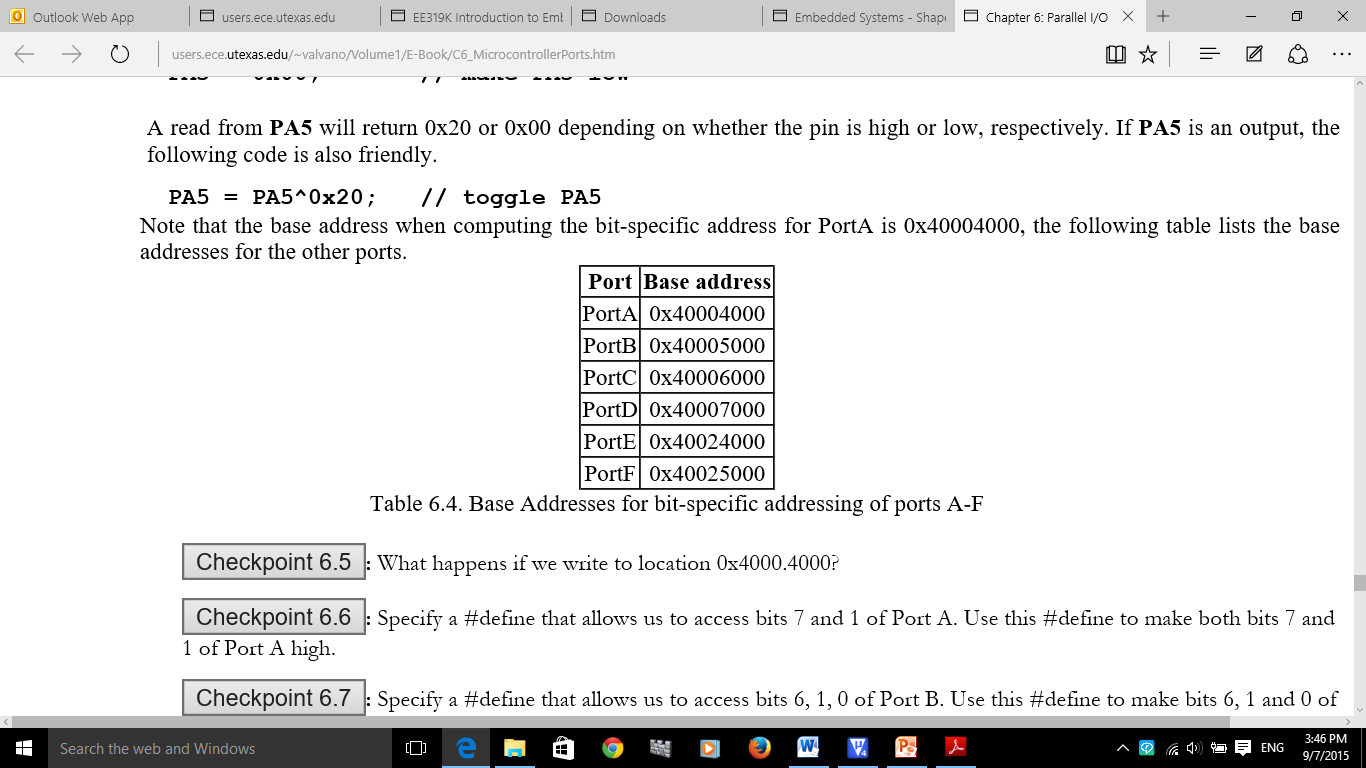
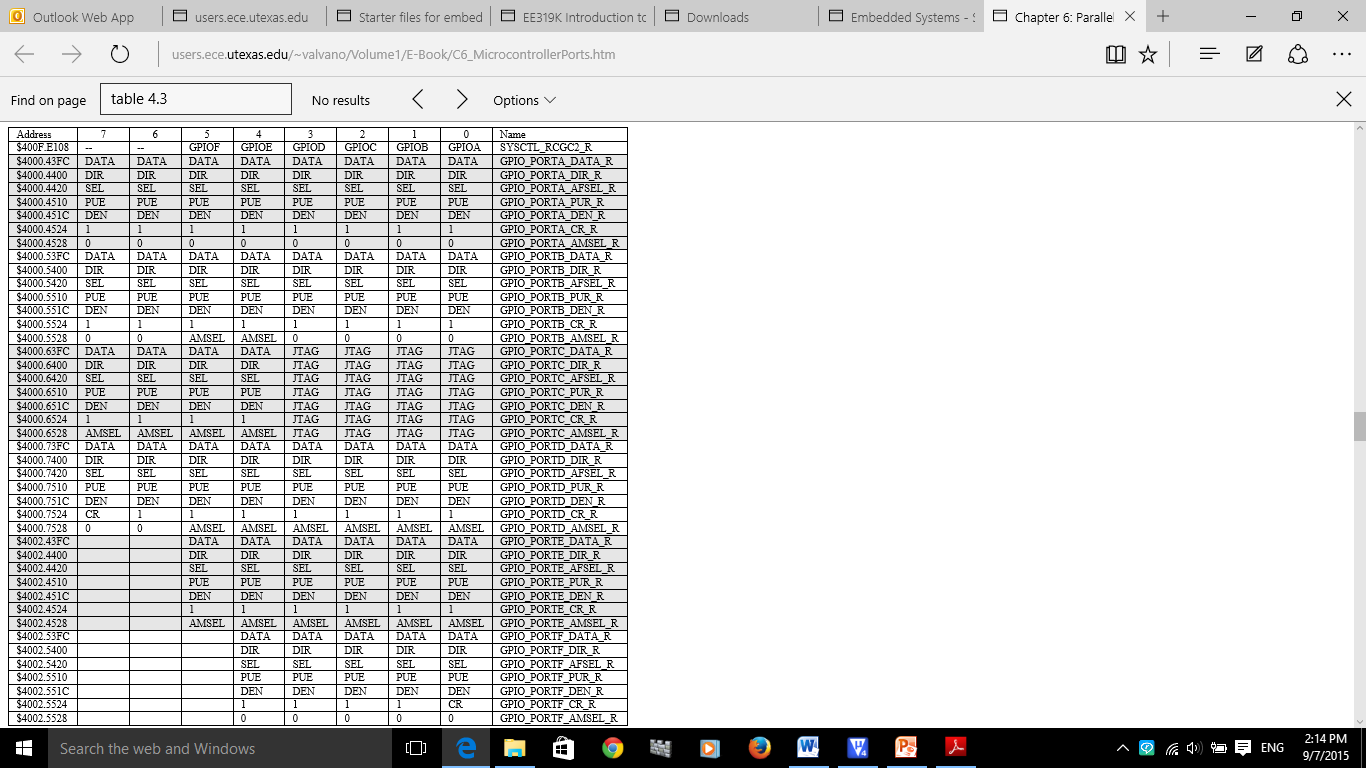
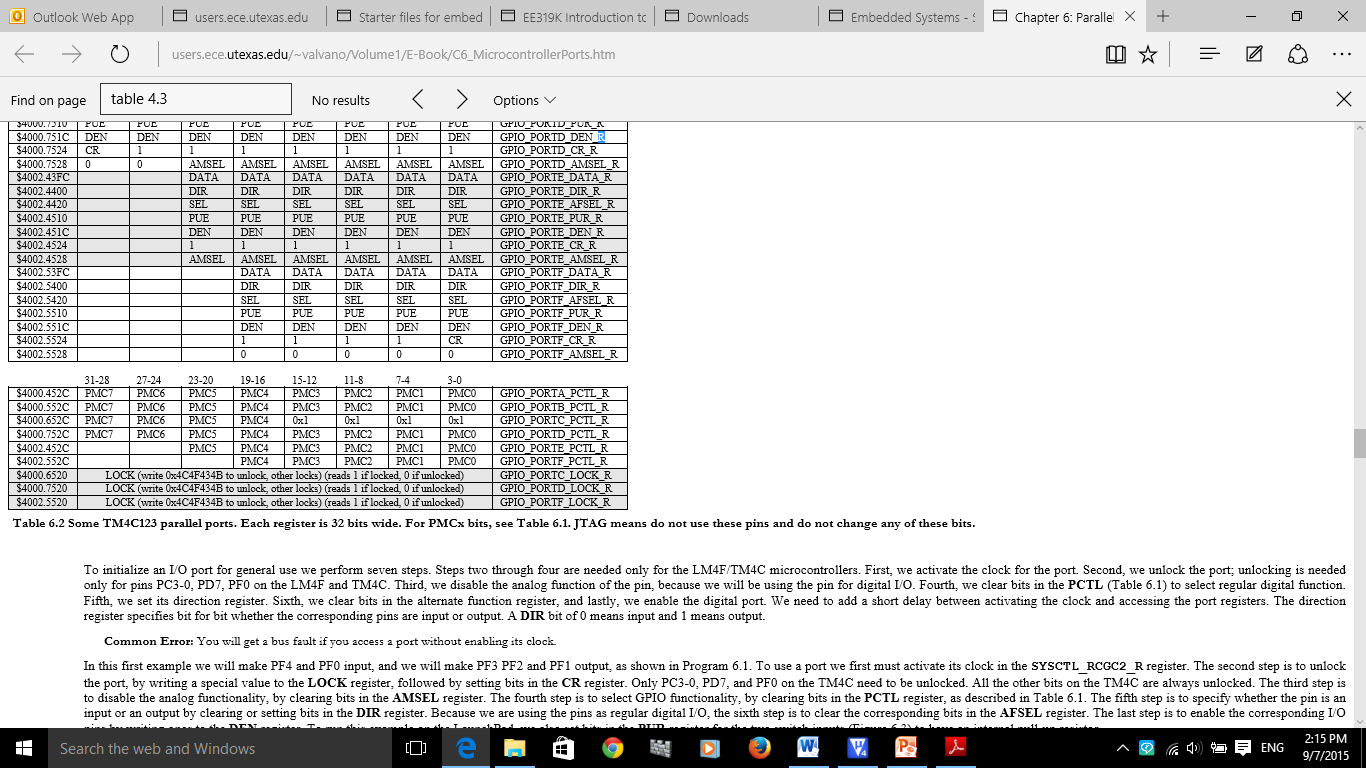
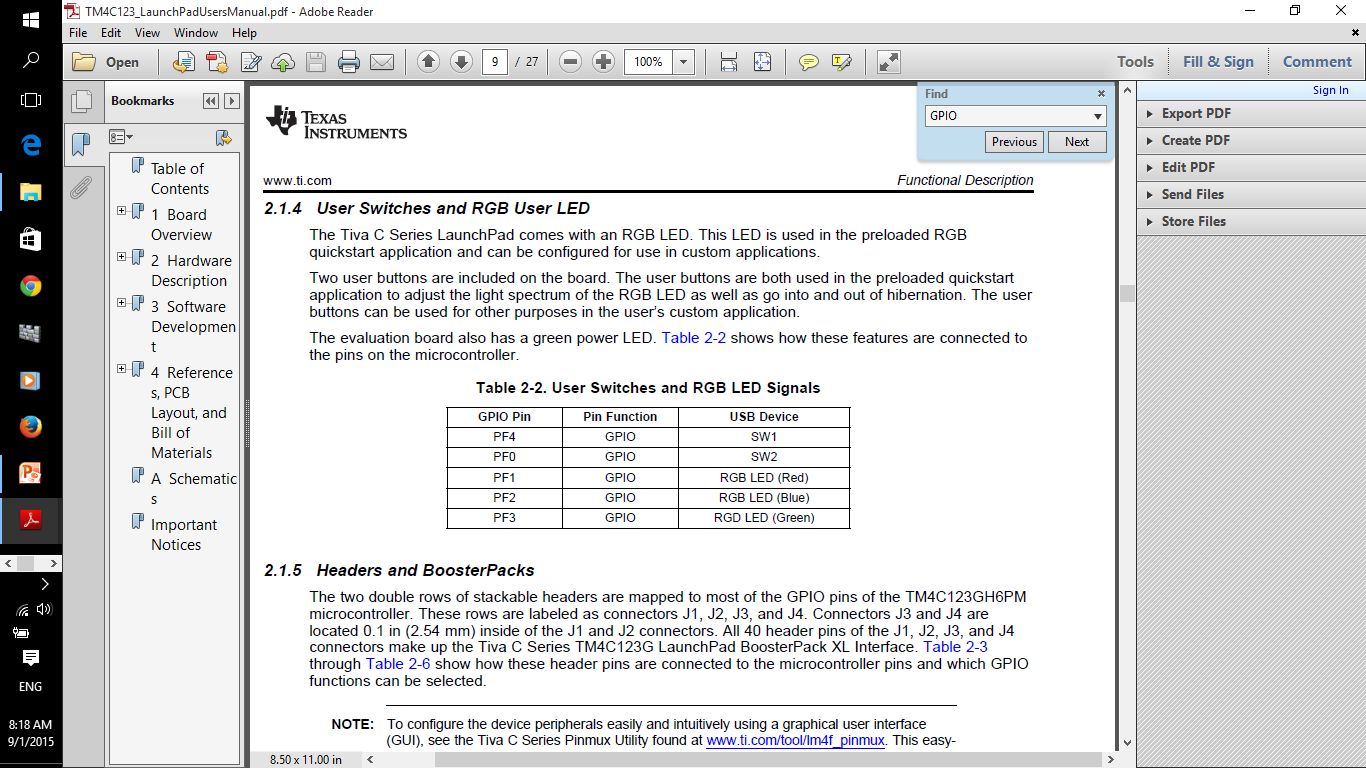
CSCE 347 Exam Information Sheet

GPIO





****



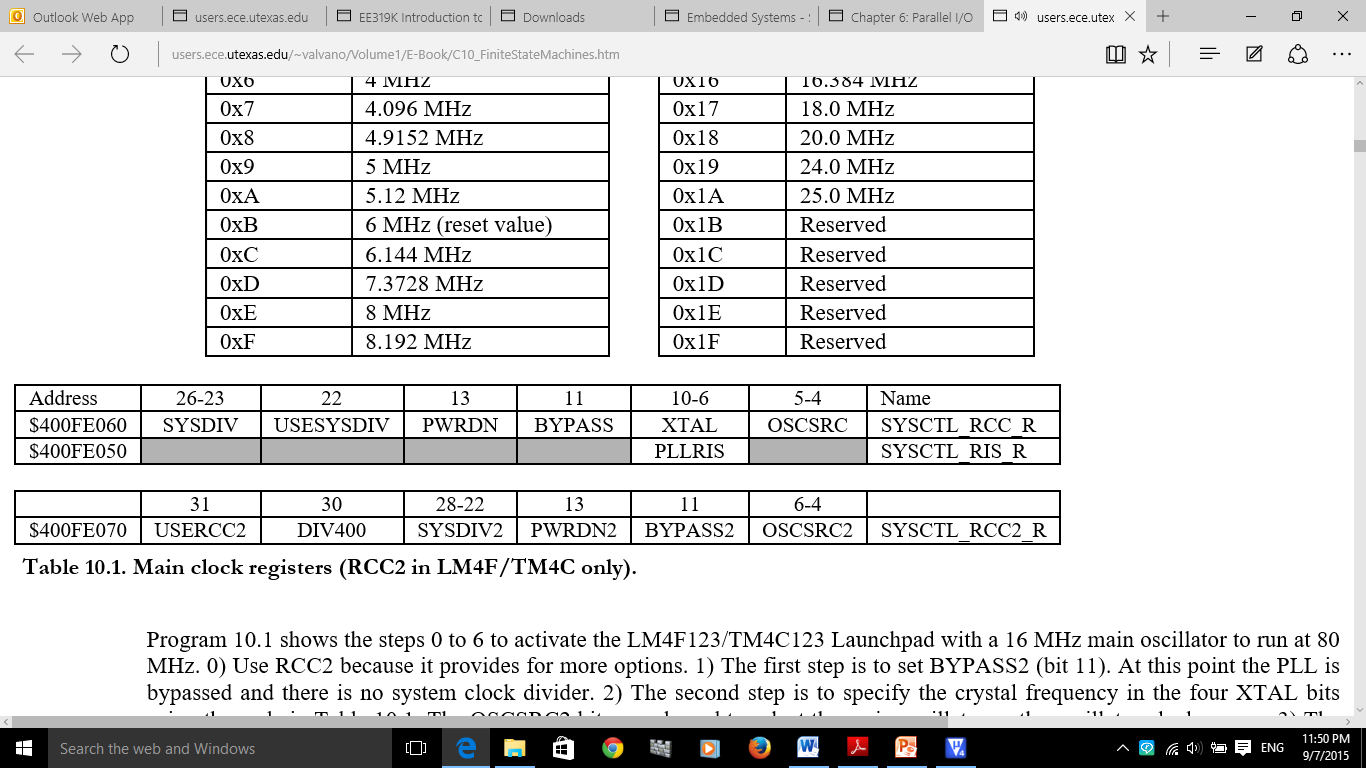
**Initialization Ritual for GPIO (executed once at beginning)**

* 1. Turn on Port x clock in **SYSCTL\_RCGCGPIO\_R/ SYSCTL\_RCGC2\_R** Wait two bus cycles (two **NOP**)
  2. For Port F or Port D only: Unlock PF0 (PD7 also needs unlocking)
  3. Clear *AMSEL* to disable analog
  4. Clear *PCTL* to select GPIO
  5. Set *DIR* to 0 for input, 1 for output
  6. Clear *AFSEL* bits to 0 to select regular I/O
  7. Set *PUR* bits to 1 to enable internal pull-up
  8. Set *DEN* bits to 1 to enable data pins

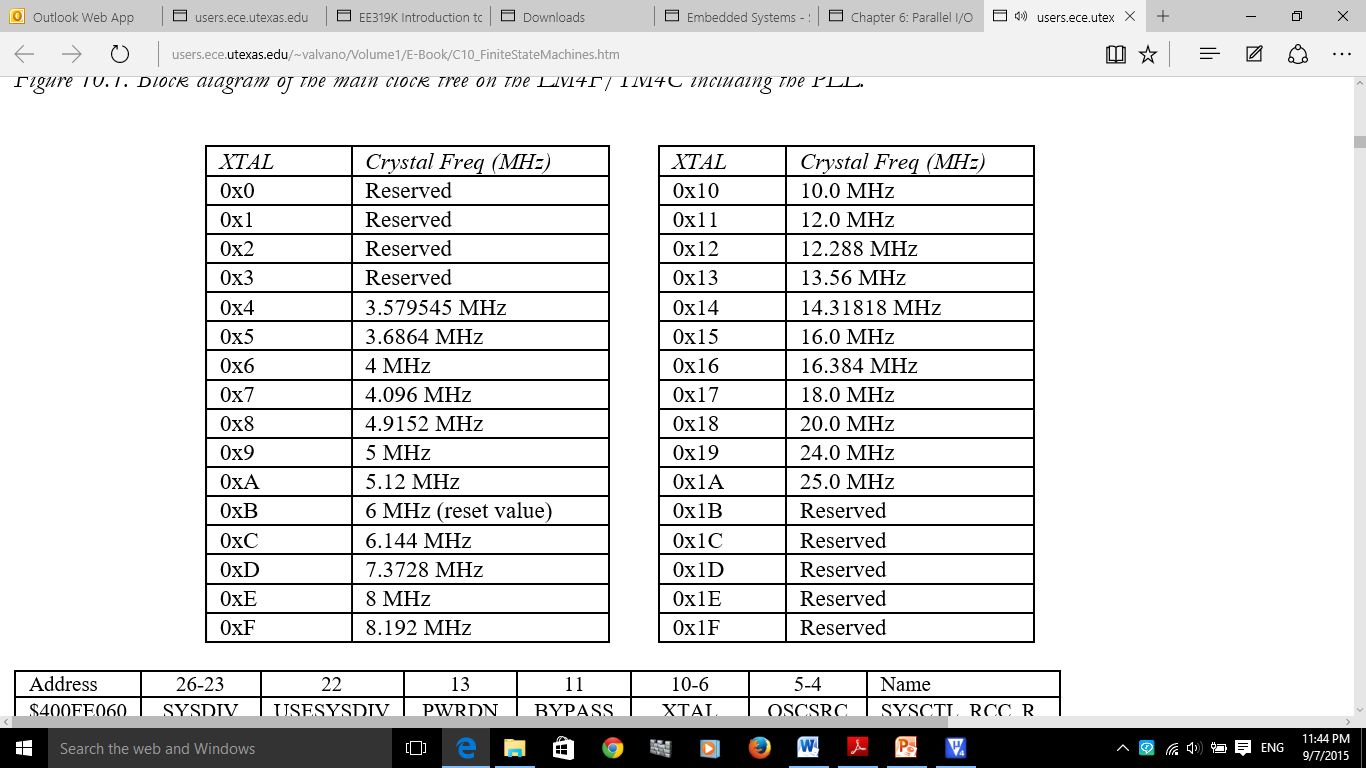
PLL and SysTick

**Initialize PLL: Use RCC2 because it provides for more options.**

* 1. Set BYPASS2 (bit 11). At this point the PLL is bypassed and there is no system clock divider.
  2. Specify the crystal frequency in the four XTAL bits using the code in Table 10.1. The OSCSRC2 bits are cleared to select the main oscillator as the oscillator clock source.
  3. Clear PWRDN2 (bit 13) to activate the PLL.
  4. Configure and enable the clock divider using the 7-bit SYSDIV2 field.
  5. Wait for the PLL to stabilize by waiting for PLLRIS (bit 6) in the **SYSCTL\_RIS\_R** to become high.
  6. Connect the PLL by clearing the BYPASS2 bit.



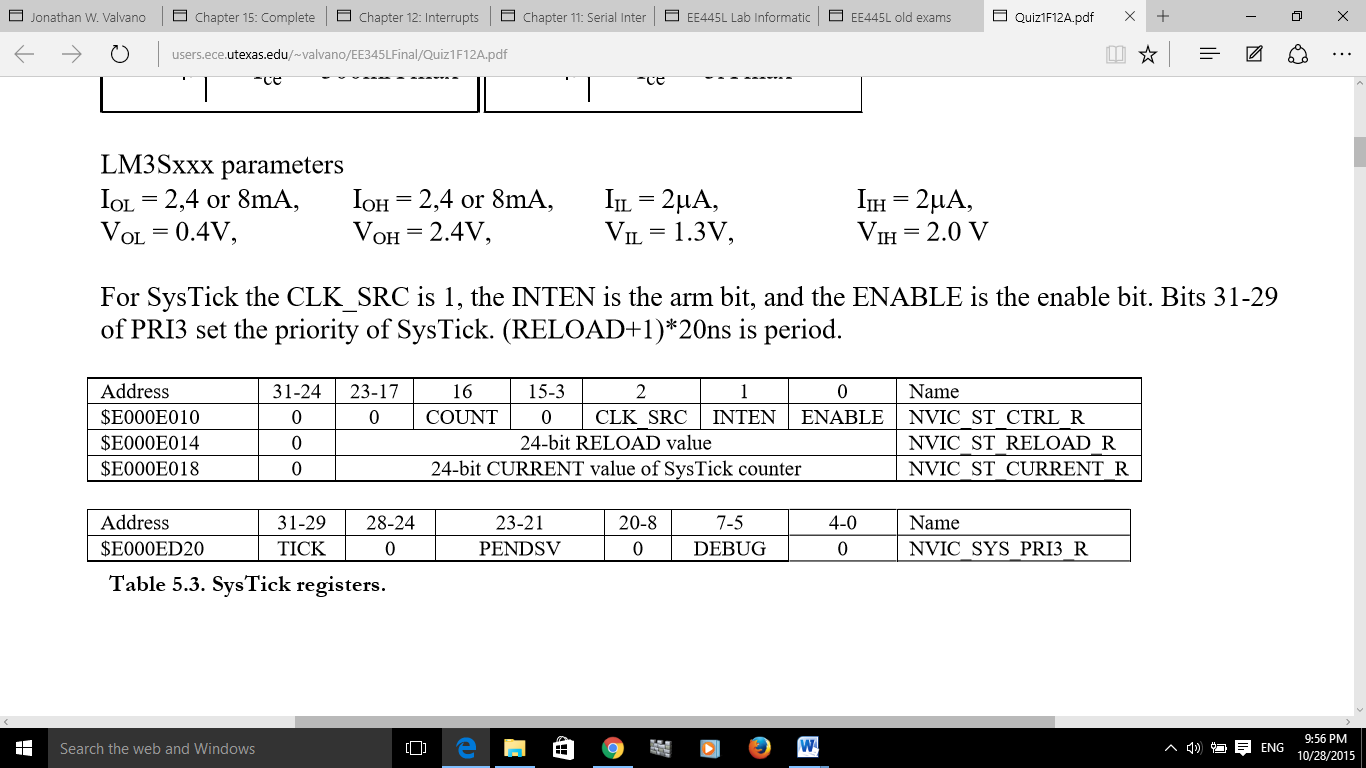
**Table 4.7b. Main clock registers for TM4C123**

****

**Table 4.7a. XTAL field used in the SYSCTL\_RCC\_R register of the TM4C123**

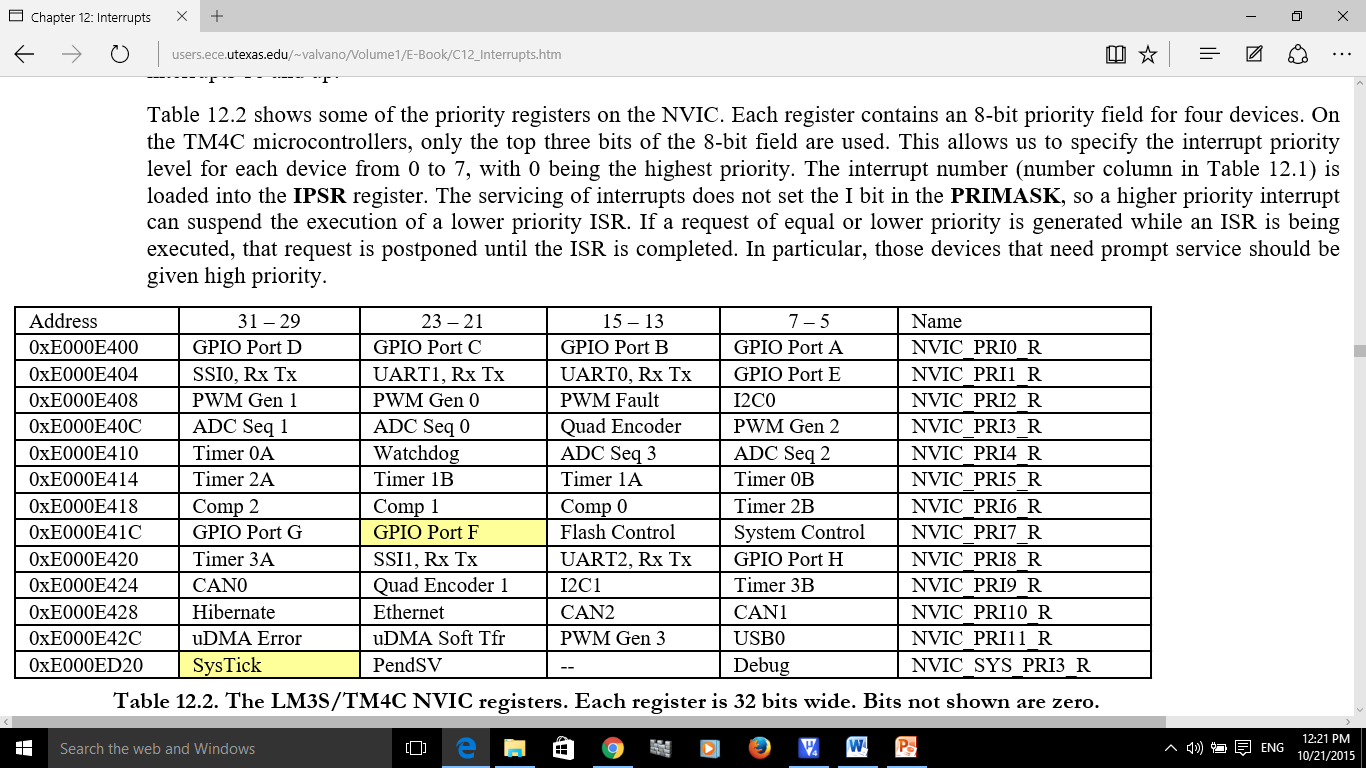
**Initialize SysTick**

* 1. Clear ENABLE to stop counter
  2. Update RELOAD value
  3. Clear the counter via NVIC\_ST\_CURRENT\_R
  4. Set CLK-SRC=1 and specify interrupt action via INTEN in NVIC\_ST\_CTRL\_R
  5. Note operation of phase-locked loop (PLL) in determining clock period

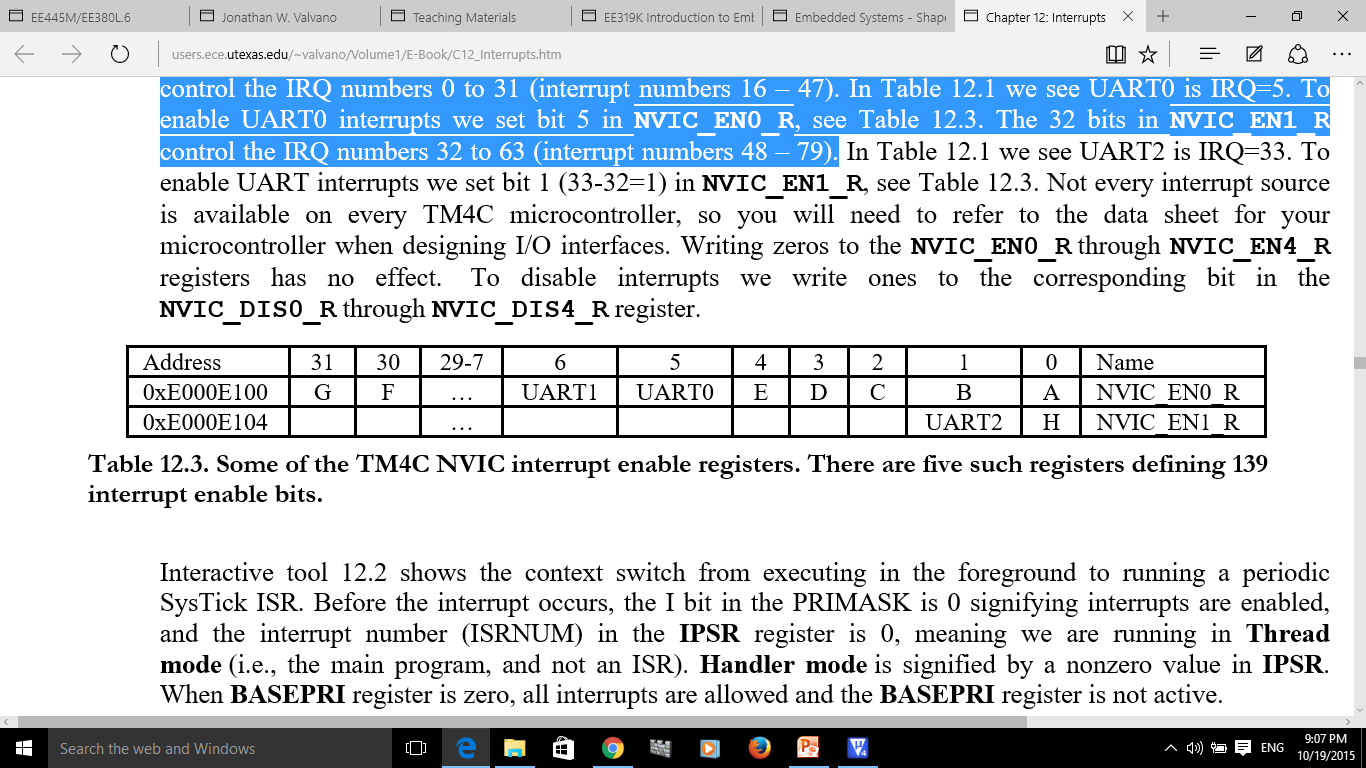


SysTick Registers

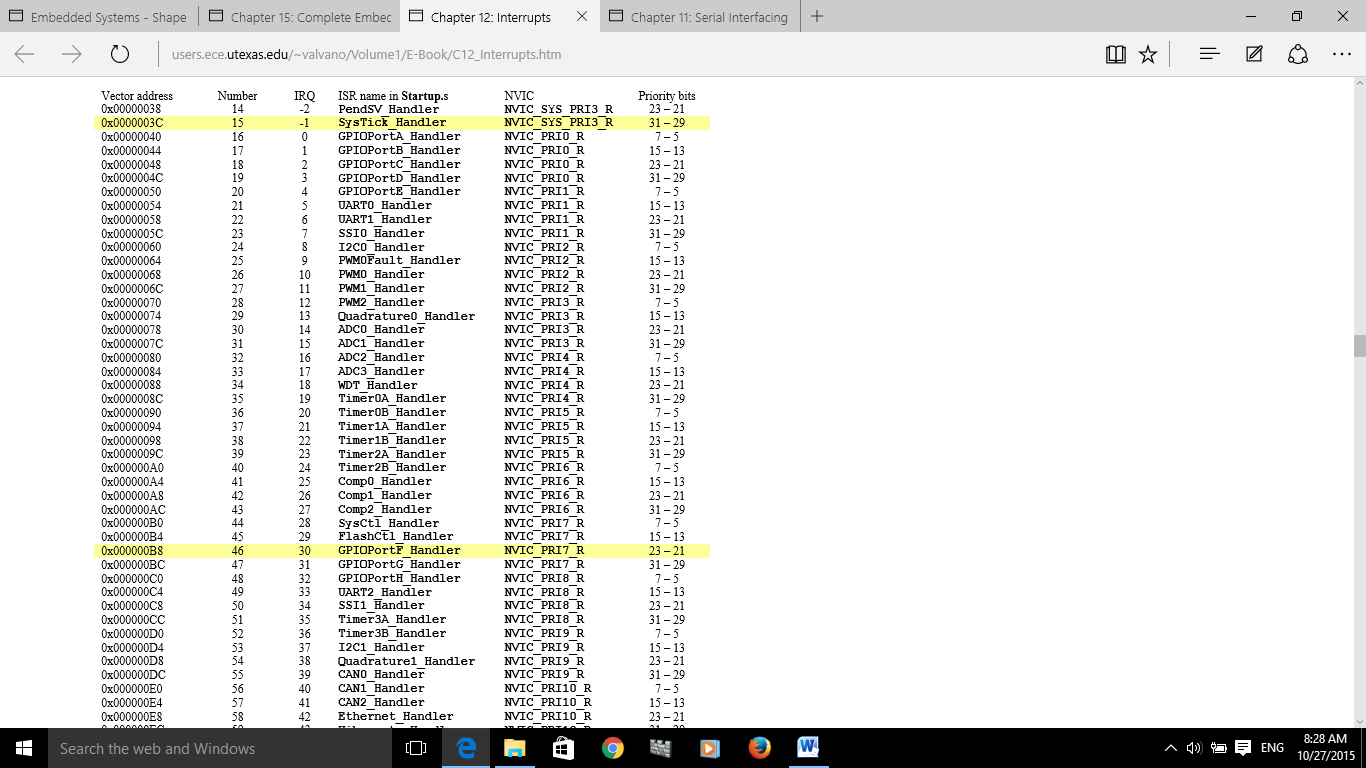
**Interrupt related tables: Table 9.1 – Table 9.6**



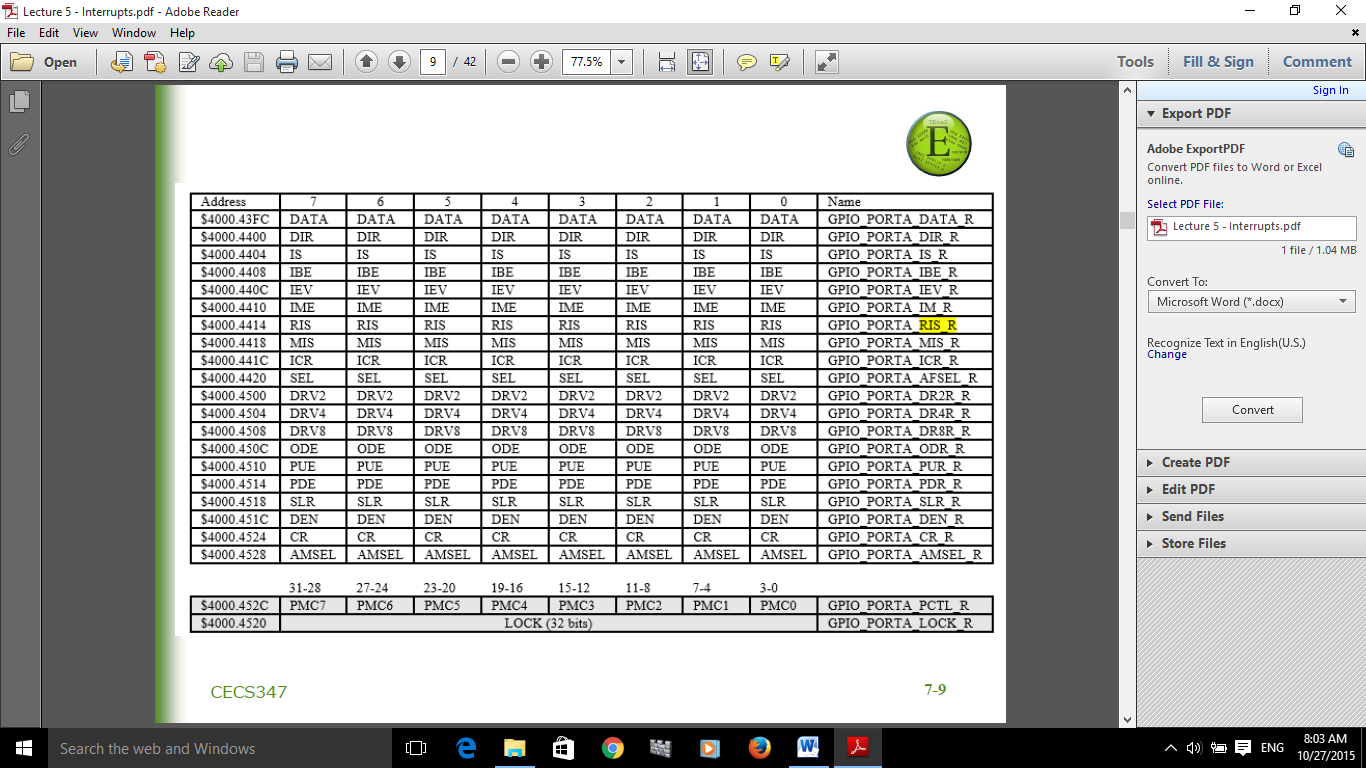
**Table 9.2 The TM4C NVIC registers. Each register is 32 bits wide. Bits not shown are zero.**



**Table 9.3 Two of the five NVIC interrupt enable registers.**



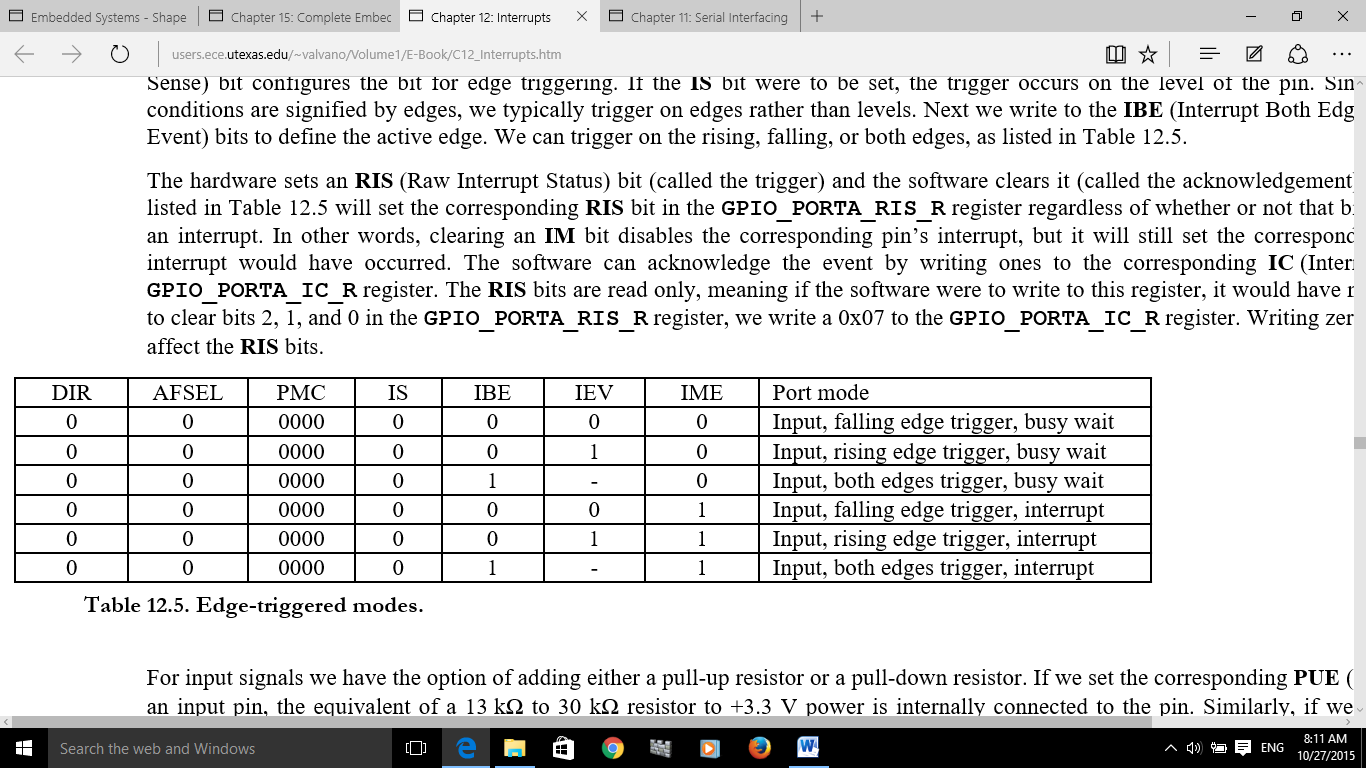
**Table 9.1 Some of the interrupt vectors for the TM4C123.**



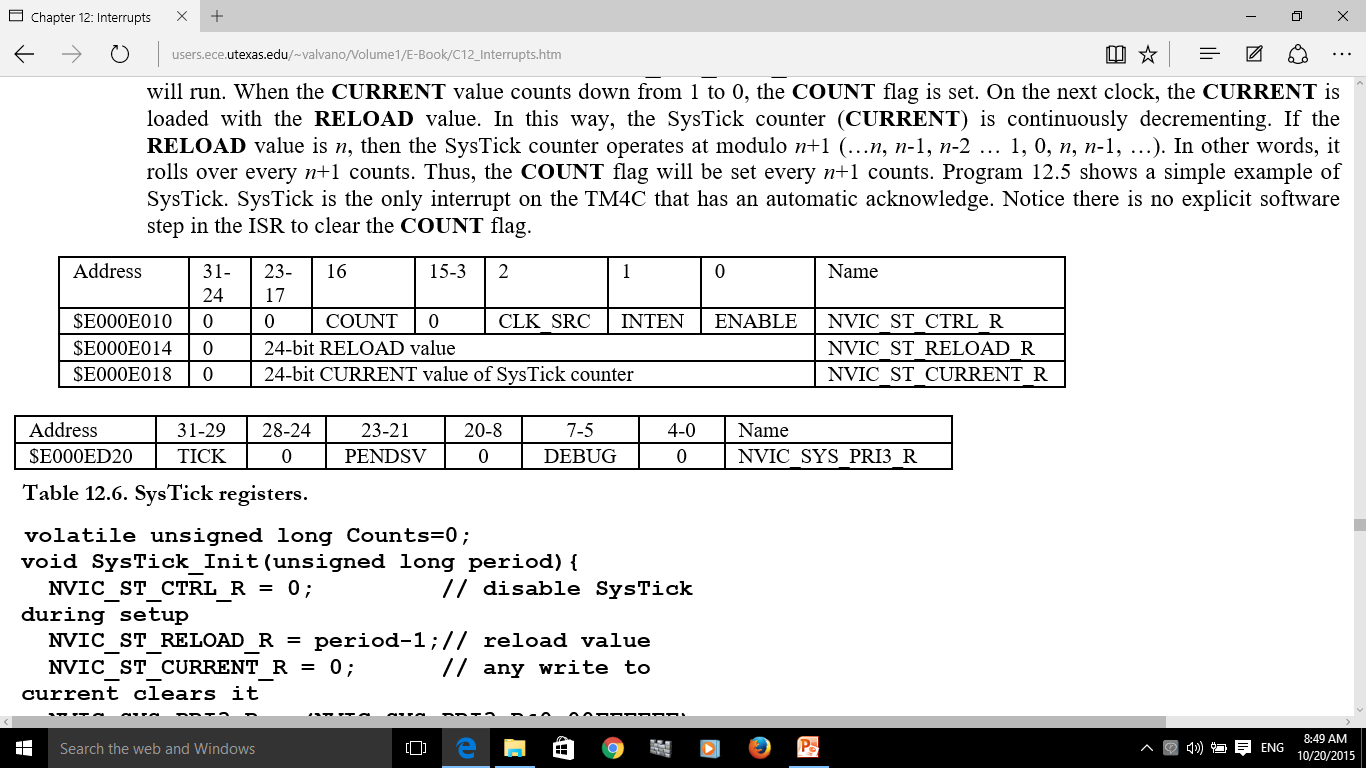
**Table 9.4 Some TM4C123 Port A registers.**

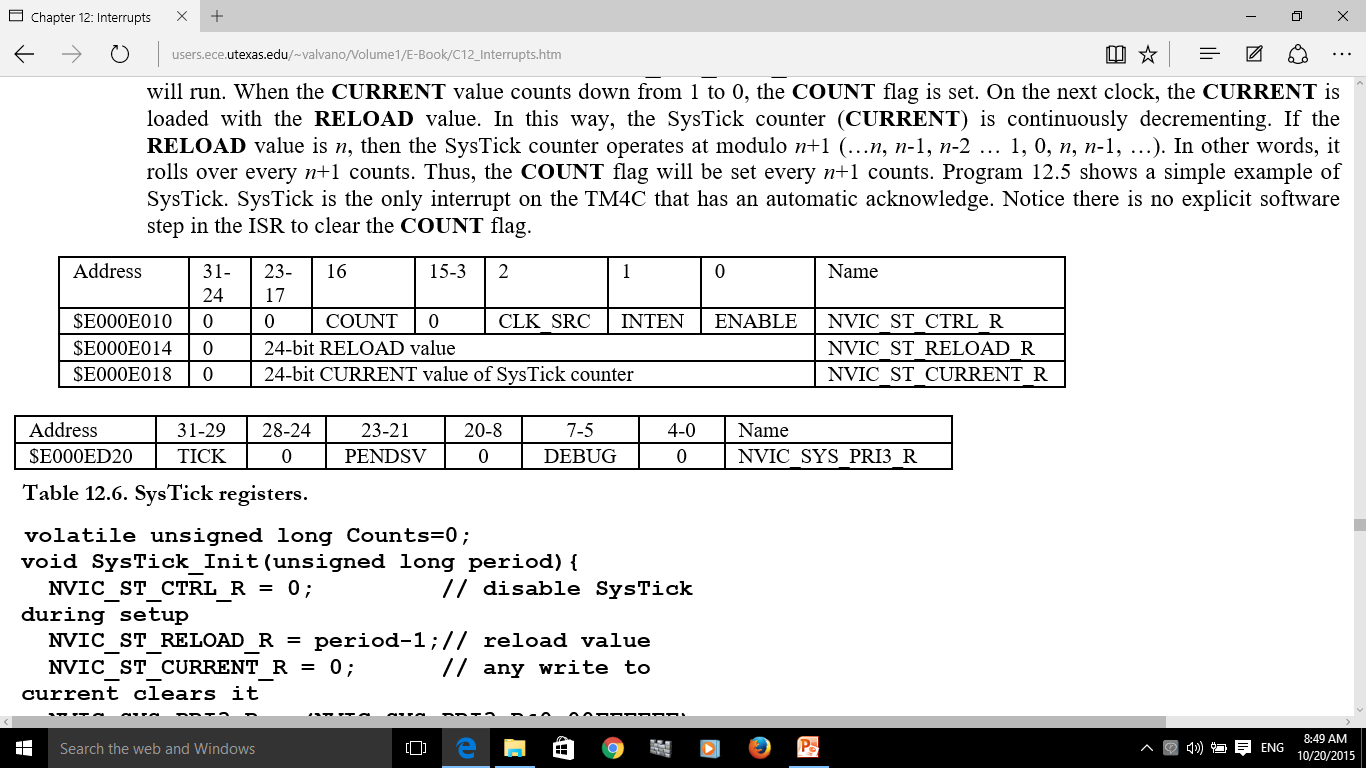
**GPIO\_PORTx\_ICR\_R:** set by software for acknowledge

**GPIO\_PORTx\_IRS\_R:** set by hardware when triggered, cleared by hardware when software sets corresponding **GPIO\_PORTx\_ICR\_R** bit(s).



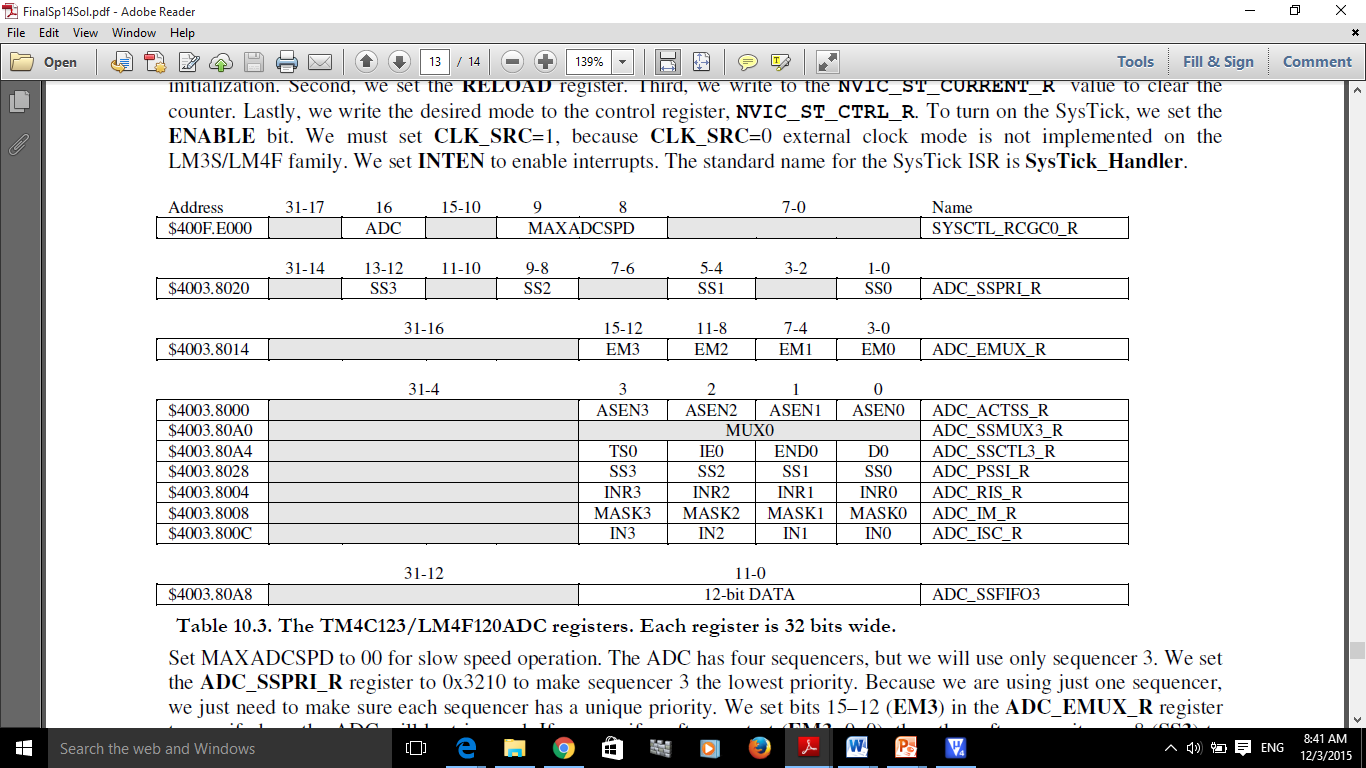
**Table 9.5 Edge-triggered and level-active interrupt modes (set IME=1 to arm interrupt)**





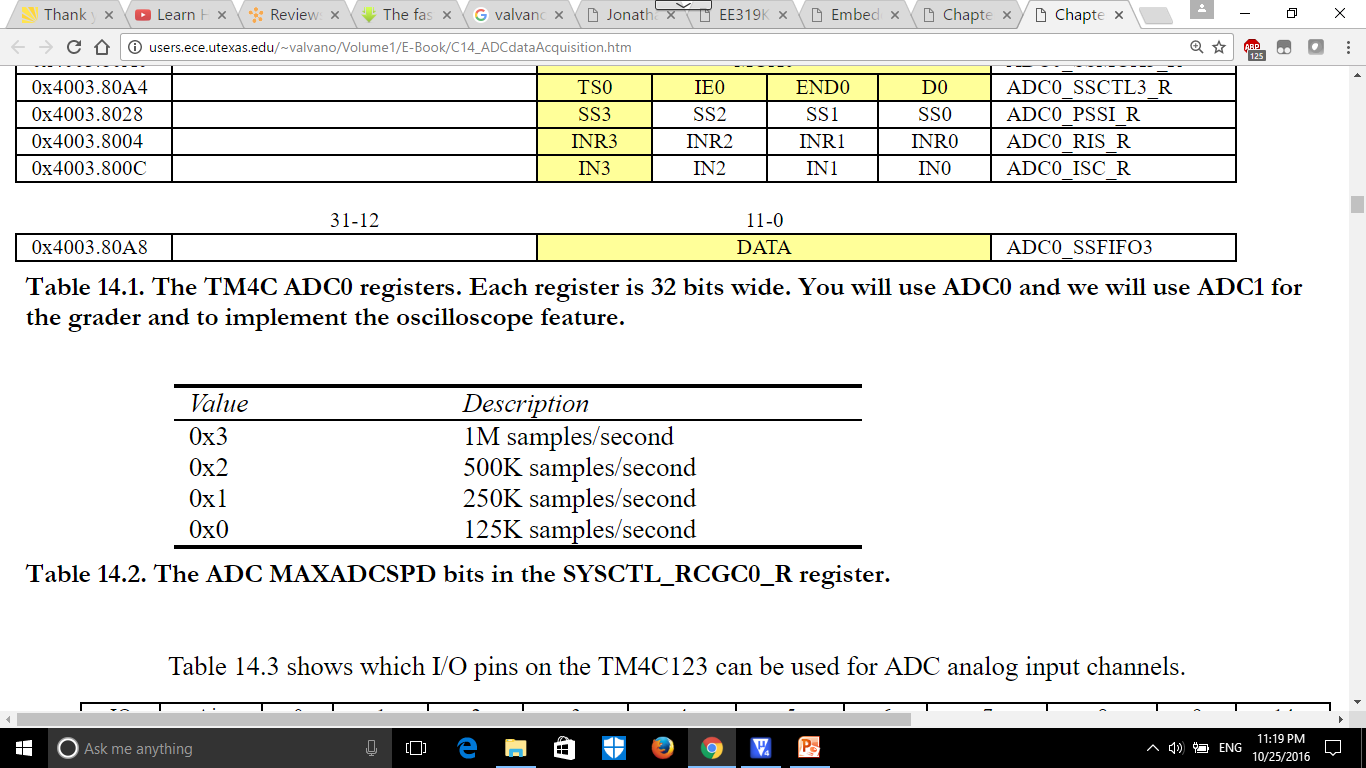
**Table 9.6 SysTick Registers**

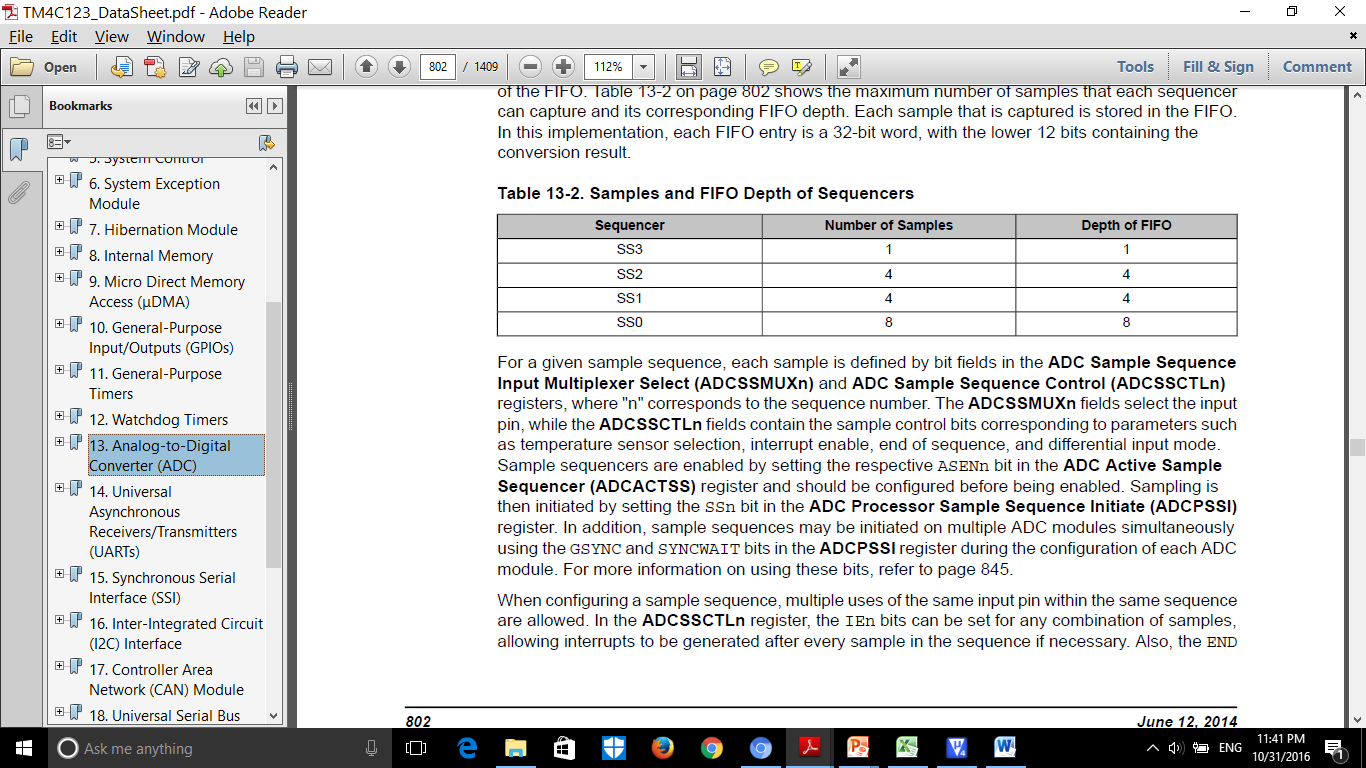
**ADC Registers**



**Table 10.3. The TM4C123/LM4F120ADC registers. Each register is 32 bits wide.**

Set MAXADCSPD to 00 for slow speed operation. The ADC has four sequencers, but we will use only sequencer 3. We set the **ADC\_SSPRI\_R** register to 0x3210 to make sequencer 3 the lowest priority. Because we are using just one sequencer, we just need to make sure each sequencer has a unique priority. We set bits 15–12 (**EM3**) in the **ADC\_EMUX\_R** register to specify how the ADC will be triggered. If we specify software start (**EM3**=0x0), then the software writes an 8 (**SS3**) to the **ADC\_PSSI\_R** to initiate a conversion on sequencer 3. Bit 3 (**INR3**) in the **ADC\_RIS\_R** register will be set when the conversion is complete. We can enable and disable the sequencers using the **ADC\_ACTSS\_R** register. There are 11 on the TM4C123/LM4F120. Which channel we sample is configured by writing to the **ADC\_SSMUX3\_R** register. The **ADC\_SSCTL3\_R** register specifies the mode of the ADC sample. Clear **TS0**. We set **IE0** so that the **INR3** bit is set on ADC conversion, and clear it when no flags are needed. We will set **IE0** for both interrupt and busy-wait synchronization. When using sequencer 3, there is only one sample, so **END0** will always be set, signifying this sample is the end of the sequence. Clear the **D0** bit. The **ADC\_RIS\_R** register has flags that are set when the conversion is complete; assuming the **IE0** bit is set. Do not set bits in the **ADC\_IM\_R** register because we do not want interrupts. Write one to **ADC\_ISC\_R** to clear the corresponding bit in the **ADC\_RIS\_R** register.



****