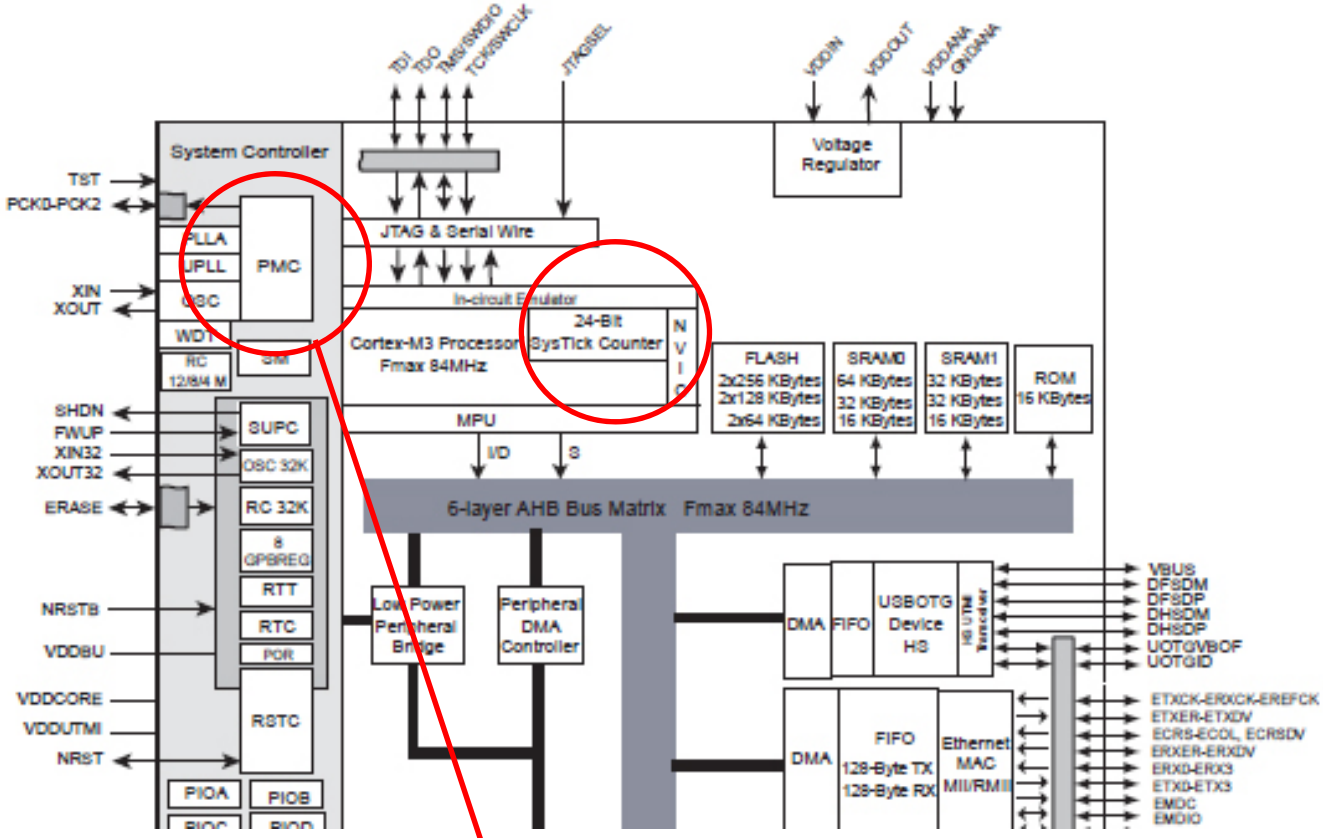
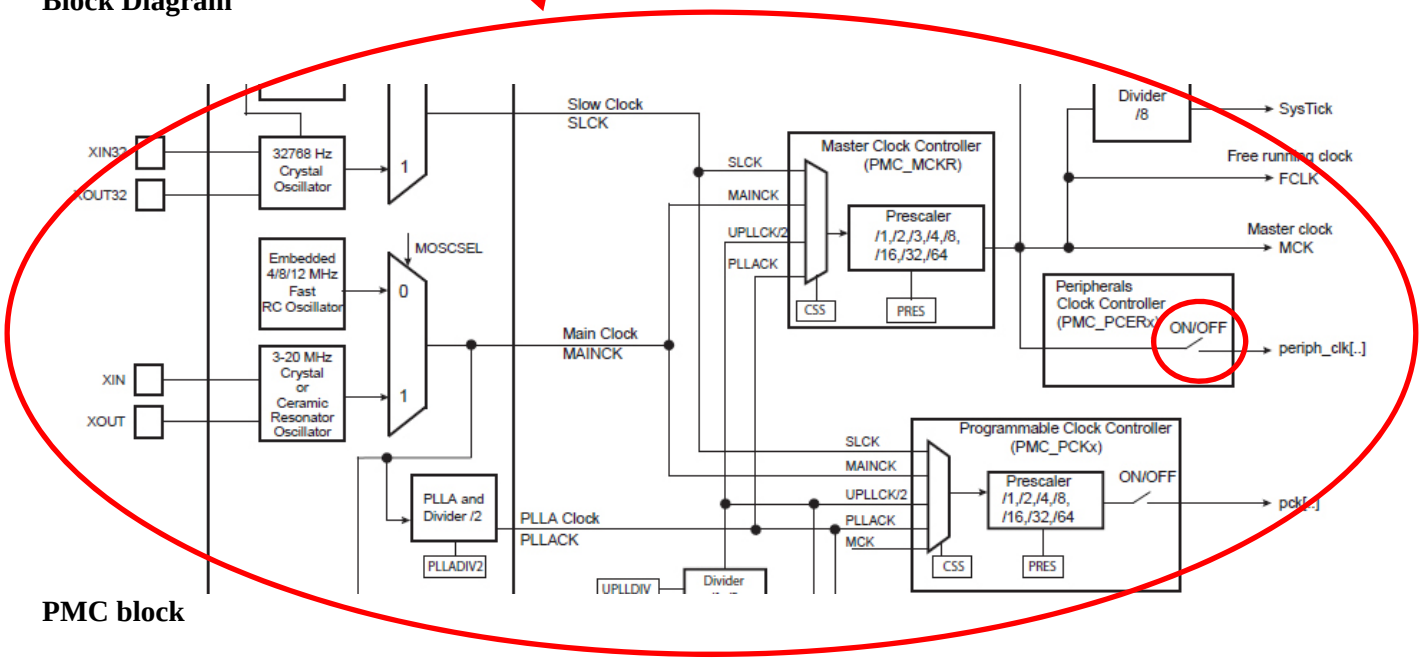
Computer Systems Engineering I
Tip to HW Lab 1

Assignment 1:

1) Connect the button to port D and pin 1 according to the wiring diagram. [Appendix A]
2) Start the peripheral clock by writing to the PMC [sam3x8e_complete page 6 and 544]



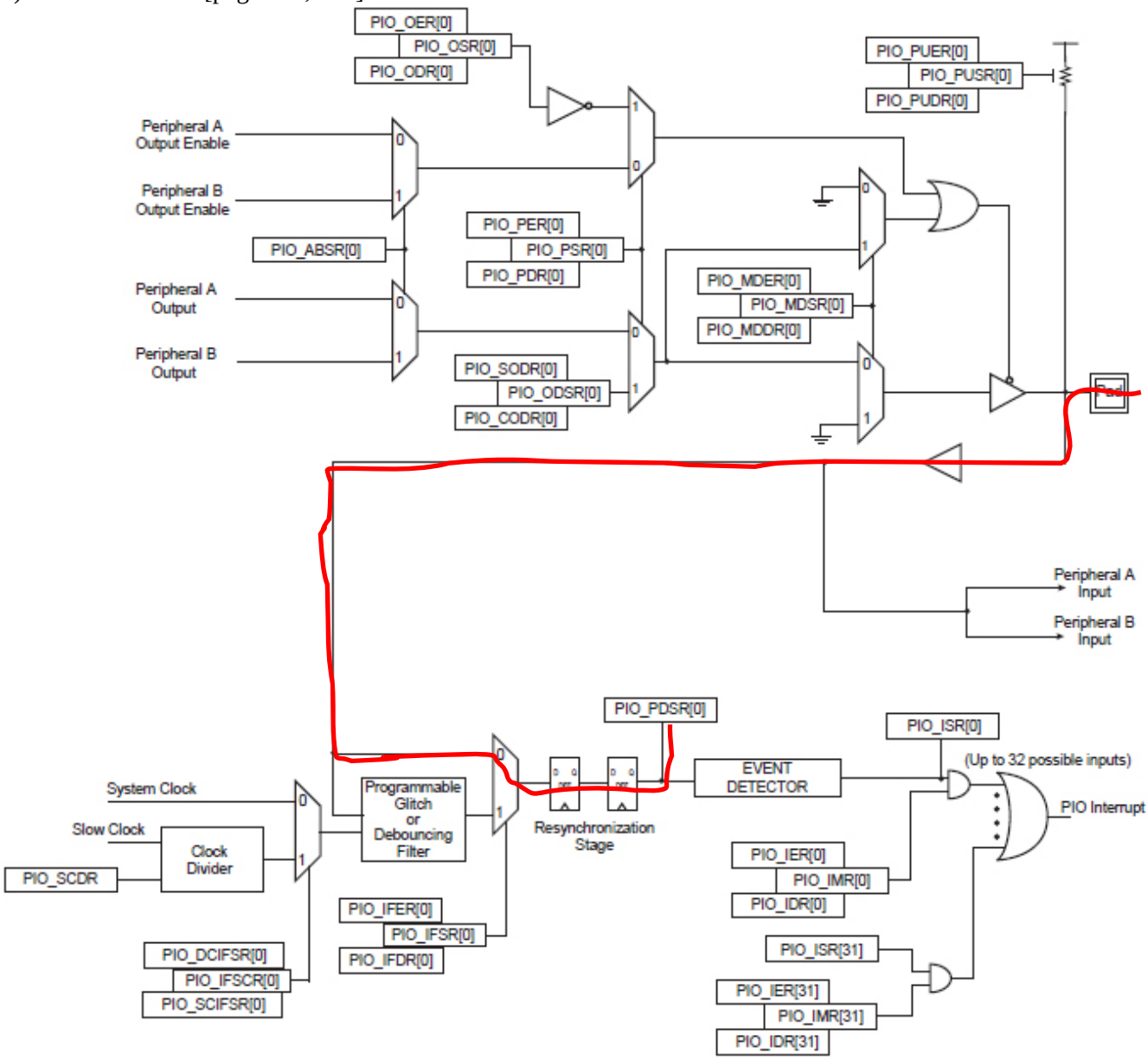
Block Diagram



PMC block

Port D has a specific instance ID number (PID number) [page 47]. Check if you should use PMC_PCER0 or PMC_PCER1 to start the peripheral clock and check which bit that has to be set.[page 561 or 583]

3) Enable the PIO [page 644, 656]



4) Read digital value from PIO [page 644, 662]

Use brake points and the debugger to check if it works. Remember the “while loop”, if you don’t have an operating system (OS) installed, here and now there is no OS. **Always** have an endless while loop within the main.

Assignment 2:

1) Connect a LED to port D pin 3 according to the wiring diagram.
2) Enable the PIO (PIO_PER) [page 644, 656]
3) Disable the internal pull up resistor (PIO_PUDR) [page 644, 666]
4) Make the pin as output (PIO_OER) [page 644, 657]

5) Write the function Set_Led that should turn the LED on and off.

Use brake points and the debugger to check if it works. Use the watch window to check a variable value by do a right click over the variable and select Add to Watch.

Assignment 3:

Write a program that turn on the LED when the button is pressed and off when the button is released by using the function from assignment 1 and 2.
Check the function by press the button and look.

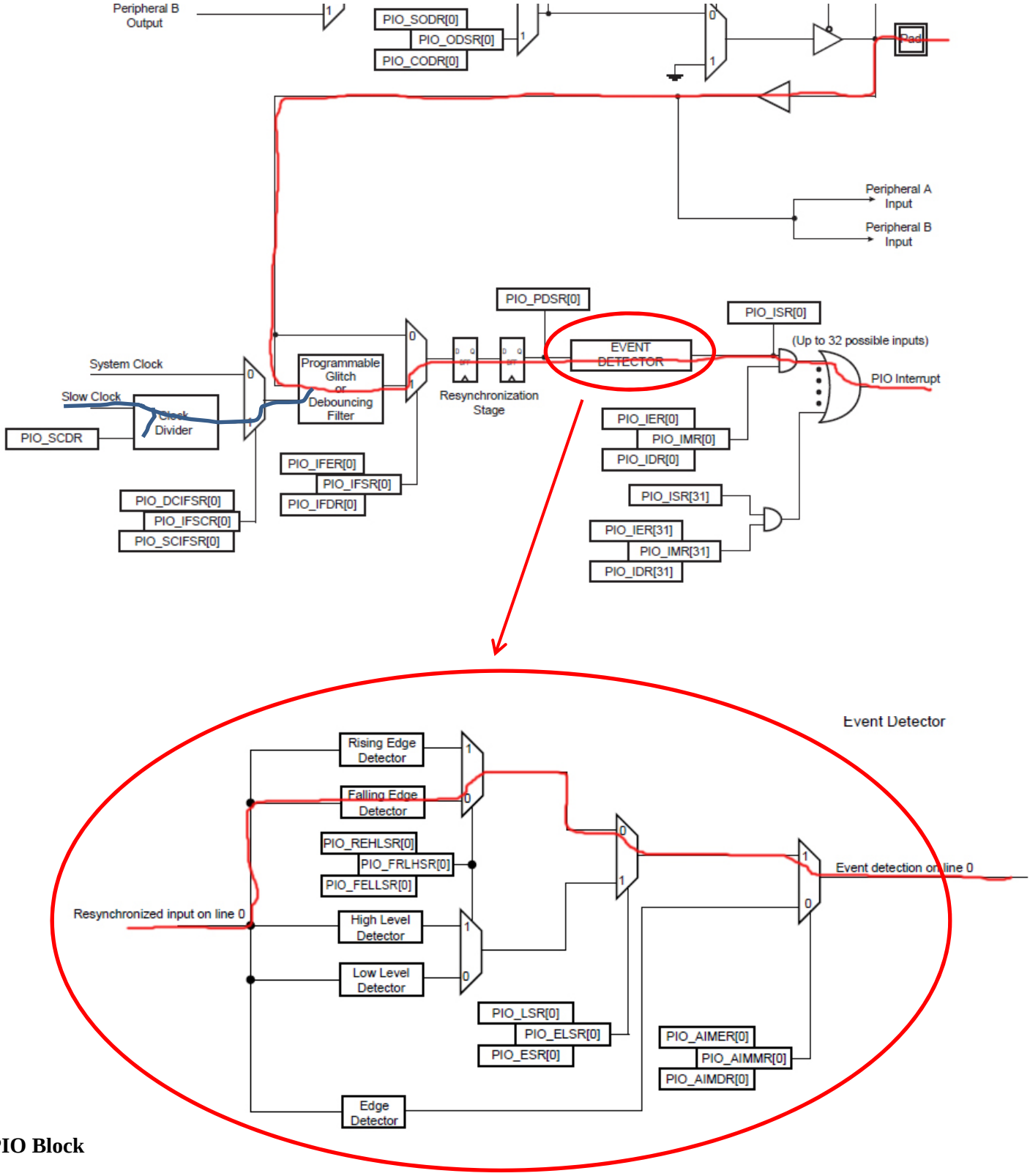
Assignment 4:

1) To start the Systick you don't need to do anything with the PMC. It's always running.
2) Write and add the function void SysTick_Handler(void){}. Always write the interrupt function and have that part ready before you start the interrupt.
3) In "core_cm3.h" there is an inline function static __INLINE uint32_t SysTick_Config(uint32_t ticks) that do the hard work for us. The only thing that has to be done is to decide "ticks". "Choose" a value on ticks so that the interrupt Systick is executed every ms. A hint, to choose a value, try to figure out the speed of the master clock.
4) Write a function that initialize and start the SysTick timer with interrupts. Write a program that use the timer to blink the LED. Blink LED ones per sec.

Check by counting blinking and compare with some other watch.

Assignment 5:

1) Write and add the function void PIOD_Handler(void). Within the interrupt handler read PIO_ISR and decide if it was the button that initiated the interrupt. Use a global variable that indicate an interrupt has occur (increment by 1 or something like that). **Don't do anything unnecessary inside an interrupt.**



PIO Block

Follow the blue and red line from the pin/PAD through the PIO block and identify which register that has to be manipulated.

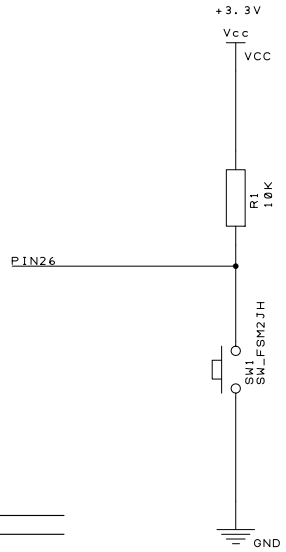
2) Enable PIO_AIMER
3) Enable PIO_IFER
4) Enable PIO_DIFSR

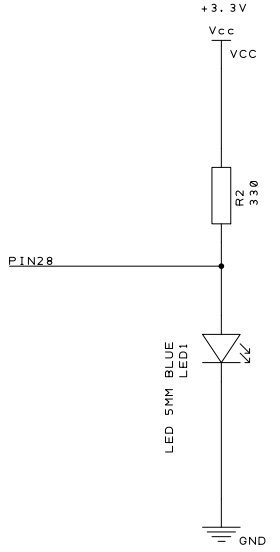
5) Set PIO_SCDR to a value. Try with different values until you are satisfied with the bouncing. It means the filter will be clocked with different clock speed. [page 648]
6) Read PIO_ISR to clear old event.
7) Use the inline function to clear the pending interrupts
   static __INLINE void NVIC_ClearPendingIRQ(IRQn_Type IRQn)
8) Use the inline function to set the priority
   static __INLINE void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)
9) Use the inline function to enable the interrupt
   static __INLINE void NVIC_EnableIRQ(IRQn_Type IRQn)
10) Enable the pin by PIO_PER
11) Enable the interrupt by PIO_IER

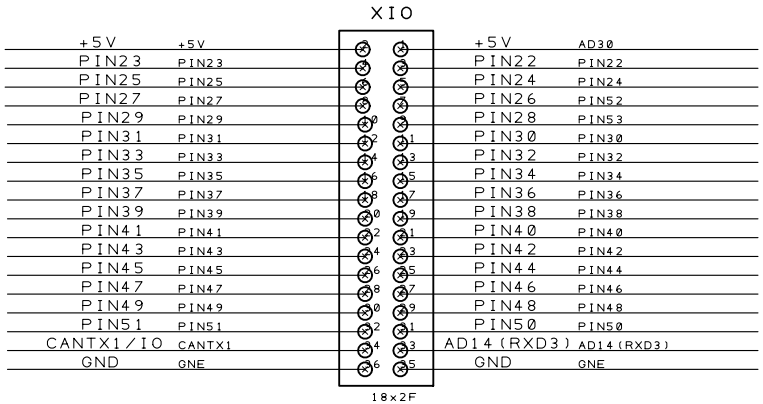
Check the function by looking and press one time on the button. Printf is an option and alternative way to check what's going on. Printf can print a string to the IDE console window. Do this by select "View" and "Terminal I/O" in debug mode.

LAB 1

2013-10-22







E	D	C	B	A	Drawn	Check	Projection Do Not Scale		
Drn	Drn	Drn	Drn	Project			Client		
Chk	Chk	Chk	Chk	Title			Filename	Drawing No.	Sheet of