We begin this section with a description of the problem. Each time we touch a switch a counter will be incremented.

VIDEO 12.3A. EDGE-TRIGERRED INTERRUPT

C12 3a Use edge-triggered interrupts YouTube

1:31 / 1:31 **1.0x**

DR. JONATHAN VALVANO: I'm Jon Valvano.

In this video we will use edge triggered interrupts

to solve a real time problem.

The problem we're going to solve is we're

going to have a switch attached to PF4, and when we touch the switch,

there will be a falling edge on PF4.

The action I wish to perform is to increment a counter.

This is a software action, and so the interrupt

is going to trigger a software action.

And the time between the hardware event and the software action

is called latency.

This is a time delay between a desire and an action.

And if this latency is less than a bound, in other words

it's always less than some number, then we will classify this system as real time.

In order to make this real time, we're going to use the interrupts.

The interrupt is going to cause a trigger on the falling edge of PF4.

We'll acknowledge the interrupt, which is to clear the trigger,

and then we'll increment the variable.

After incrementing the variable, we'll return back to where we came from.

Next we'll show you the steps involved in initializing

the edge triggered interrupts.

Help

Synchronizing software to hardware events requires the software to recognize when the hardware changes states from busy to done. Many times the busy to done state transition is signified by a rising (or falling) edge on a status signal in the hardware. For these situations, we connect this status signal to an input of the microcontroller, and we use edge-triggered interfacing to configure the interface to set a flag on the rising (or falling) edge of the input. Using edge-triggered interfacing allows the software to respond quickly to changes in the external world. If we are using busy-wait synchronization, the software waits for the flag. If we are using interrupt synchronization, we configure the flag to request an interrupt when set. Each of the digital I/O pins on the TM4C family can be configured for edge triggering. Table 12.4 shows the registers needed to set up edge triggering for Port A. The differences between members of the TM4C family include the number of ports (e.g., the TM4C123 has ports A – F) and the number of pins in each port (e.g., the TM4C123 only has pins 4 – 0 in Port F). For more details, refer to the datasheet for your specific microcontroller. Any or all of digital I/O pins can be configured as an edge-triggered input. When writing C code using these registers, include the header file for your particular microcontroller (e.g., **tm4c123ge6pm.h**). To use any of the features for a digital I/O port, we first enable its clock in the Run Mode Clock Gating Control Register 2 (RCGC2). For each bit we wish to use we must set the corresponding **DEN** (Digital Enable) bit. To use edge triggered interrupts we will clear the corresponding bits in the **PCTL** register, and we will clear bits in the **AFSEL** (Alternate Function Select) register. We clear **DIR** (Direction) bits to make them input. On the TM4C123, only pins PD7 and PF0 need to be unlocked. We clear bits in the **AMSEL** register to disable analog function.

Address	7	6	5	4	3	2	1	0	Name
$4000.43FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTA_DATA_R
$4000.4400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTA_DIR_R

$4000.4404	IS	IS	IS	IS	IS	IS	IS	IS	GPIO_PORTA_IS_R
$4000.4408	IBE	IBE	IBE	IBE	IBE	IBE	IBE	IBE	GPIO_PORTA_IBE_R
$4000.440C	IEV	IEV	IEV	IEV	IEV	IEV	IEV	IEV	GPIO_PORTA_IEV_R
$4000.4410	IME	IME	IME	IME	IME	IME	IME	IME	GPIO_PORTA_IM_R
$4000.4414	RIS	RIS	RIS	RIS	RIS	RIS	RIS	RIS	GPIO_PORTA_RIS_R
$4000.4418	MIS	MIS	MIS	MIS	MIS	MIS	MIS	MIS	GPIO_PORTA_MIS_R
$4000.441C	ICR	ICR	ICR	ICR	ICR	ICR	ICR	ICR	GPIO_PORTA_ICR_R
$4000.4420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTA_AFSEL_R
$4000.4500	DRV2	DRV2	DRV2	DRV2	DRV2	DRV2	DRV2	DRV2	GPIO_PORTA_DR2R_R
$4000.4504	DRV4	DRV4	DRV4	DRV4	DRV4	DRV4	DRV4	DRV4	GPIO_PORTA_DR4R_R
$4000.4508	DRV8	DRV8	DRV8	DRV8	DRV8	DRV8	DRV8	DRV8	GPIO_PORTA_DR8R_R
$4000.450C	ODE	ODE	ODE	ODE	ODE	ODE	ODE	ODE	GPIO_PORTA_ODR_R
$4000.4510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTA_PUR_R
$4000.4514	PDE	PDE	PDE	PDE	PDE	PDE	PDE	PDE	GPIO_PORTA_PDR_R
$4000.4518	SLR	SLR	SLR	SLR	SLR	SLR	SLR	SLR	GPIO_PORTA_SLR_R
$4000.451C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTA_DEN_R
$4000.4524	CR	CR	CR	CR	CR	CR	CR	CR	GPIO_PORTA_CR_R
$4000.4528	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	GPIO_PORTA_AMSEL_R
	31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0	
$4000.452C	PMC7	PMC6	PMC5	PMC4	PMC3	PMC2	PMC1	PMC0	GPIO_PORTA_PCTL_R
$4000.4520	LOCK (32 bits)								GPIO_PORTA_LOCK_R

Table 12.4. Some TM4C port A registers. We will clear PMC bits to used edge triggered interrupts.

Help