

- Courseware (/courses/UTAustinX/UT.6.01x/1T2014/courseware)
- Course Info (/courses/UTAustinX/UT.6.01x/1T2014/info)
- Discussion (/courses/UTAustinX/UT.6.01x/1T2014/discussion/forum)
- Progress (/courses/UTAustinX/UT.6.01x/1T2014/progress)
- Questions (/courses/UTAustinX/UT.6.01x/1T2014/a3da417940af4ec49a9c02b3eae3460b/)
- Syllabus (/courses/UTAustinX/UT.6.01x/1T2014/a827a8b3cc204927b6efaa49580170d1/)
- Embedded Systems Community (/courses/UTAustinX/UT.6.01x/1T2014/e3df91316c544d3e8e21944fde3ed46c/)

To configure an edge-triggered pin, we first enable the clock on the port and configure the pin as a regular digital input. Clearing the **IS** (Interrupt Sense) bit configures the bit for edge triggering. If the **IS** bit were to be set, the trigger occurs on the level of the pin. Since most busy to done conditions are signified by edges, we typically trigger on edges rather than levels. Next we write to the **IBE** (Interrupt Both Edges) and **IEV** (Interrupt Event) bits to define the active edge. We can trigger on the rising, falling, or both edges, as listed in Table 12.5. We clear the **IME** (Interrupt Mask Enable) bits if we are using busy-wait synchronization, and we set the **IME** bits to use interrupt synchronization.

VIDEO 12.3B. EDGE-TRIGGERRED INTERRUPT CONFIGURATION

Help

C12 3b Configuring edge triggered interrupts

YouTube



	9:04 / 9:04	1.0x			
--	-------------	------	--	--	--

JONATHAN VALVANO: Let's look at the steps involved in configuring an edge triggered interrupt. The hardware step is to connect PF4 to the switch. That's the one on our LaunchPad. The software steps are to make it an input, to not do alternative function, not do analog, to not do PCTL, so all these bits will be zero. The interesting step is the set up for the edge trigger. We have the direction bit, the input sense bit, we have this IBE bit, we have this IEV bit, and we have this IME bit.

04/23/2014 04:07 PM

Specifying which edge will interrupt | 12.4 E...

<https://courses.edx.org/courses/UTAustinX/UT...>

And these are the bits that we will set to specify the mode for edge trigger.

In all of the modes, we're going to set the direction register to a zero.

And in our case, we're interested in edges so we were going to set the IS bit equal to zero and that means edge.

The next is we're going to clear the both bit because we don't want both

but we do want the rising edge or the falling edge.

So the IEV bit zero means it's a falling edge, edge triggered.

The IEV bit equal 1 would be a rising edge.

And the last mode we could look at, or use potentially, is the both mode equal to a one.

Doesn't matter what this one is, and in that mode

we're going to get both interrupts, both edges will cause an interrupt.

RAMESH YERRABALLI: So if I was pressing a switch then Jon,

it would interrupt on the touch and the release?

JONATHAN VALVANO: That's what this mode would do.

But the mode we're going to use is this mode here,

because we want just a touch to cause an interrupt.

This last bit will be a one and that says, once you see the interrupt-- once you see the trigger-- cause an interrupt.

So the idea of an arm, is the trigger will cause an interrupt.

So the mode we're going to use, is this mode right here.

Next, let's look at some other things we have to do.

Interrupts use vectors.

Vectors are addresses which tell the system

where to get the software to execute.

And so there are lots of vectors in the system.

The particular vector that we're going to be using

2 of 8
one here located at address
0x000000B8

and this is going to be the port F interrupt service routine.

This vector table are 32-bit addresses, and they begin at location zero.

So at location zero is the initial stack pointer,

at location four is the initial program counter, somewhere in the middle

will be our SysTick vector, and the one that we're interested in

is the port F, the port F interrupt service routine which is located down here in ROM location B8.

RAMESH YERRABALLI: So for each interrupt that is available on the LaunchPad,

there's a corresponding entry in the interrupt vector

table that tells the system what interrupt service

routine to run when that particular interrupt occurs.

JONATHAN VALVANO: Exactly.

And there are hundreds of them.

The one we're interested is this one right here,

the port F interrupt service routine.

Next let's talk about priority.

Priority is like you'd imagine.

Priority means if two things happen at the same time, which one goes first?

Which is most important?

And there's a register on the system called priority seven.

And it turns out that bits 23, 22, and 21 of this register

will specify a number between zero to seven

and set the priority for the port F interrupt.

Each interrupt that's possible has a three bit

location in one of these priority registers to specify it's priority.

RAMESH YERRABALLI: So priority does not just

pertain to two things happening at the same time.

3 of 8
bits of one priority-- one interrupt service routine is currently running,

and if another interrupt were to occur,
the question

of whether this interrupt continues or is
suspended

and this new interrupt that occurs, gets
processed.

The priority answers that question.

If a higher priority interrupt were to occur,
then the current interrupt service routine
is suspended.

The new interrupt service routine runs,
and then

control returns to the interrupt service
routine that was suspended.

JONATHAN VALVANO: Exactly.

So a high priority interrupt service
routine--

let's say this one is a priority one, and this
one is a priority

two-- a high priority interrupt can
preempt a low priority interrupt
service routine.

In this example, we're going to set the
priority for this interrupt to two.

Which means a zero or one could
preempt, but a two through seven would
not.

So we saw the vector and the priority.

Next we're going to have to look at an
enable for this interrupt.

There are two enables.

There's one specific enable in the nested
vectored interrupt controller,
enable register zero.

And it turns out that bit 30 will have to be
set at one

to enable the port F interrupts.

And the second, is a global interrupt
enable for all interrupts.

And that exists in the PRIMASK register,
which exists in the processor.

And that's the I bit.

And the I bit has to be zero to enable.

And if it happened to be a one that would
mean disable.

All right, next let's look inside at the port
F.

There are three registers in the port F
edge triggered mode.

Each of the bits in port F could be edge triggered.

And the one we're interested in, is port F, bit four.

So we're interested in bit four of these registers.

And the IM is the arm bit which we will set one, once in the initialization

such that we want port F to have an edge triggered interrupt.

The actual trigger bit is right here.

There's the trigger bit.

And we can look at if you want to.

The trigger bit here is in the RIS-- raw interrupt status register--

and there's the trigger bit.

And we have one more bit, bit four of the interrupt clear register.

If we write a one to this bit that will

Help

The hardware sets an **RIS** (Raw Interrupt Status) bit (called the trigger) and the software clears it (called the acknowledgement). The triggering event listed in Table 12.5 will set the corresponding **RIS** bit in the **GPIO_PORTA_RIS_R** register regardless of whether or not that bit is allowed to request an interrupt. In other words, clearing an **IME** bit disables the corresponding pin's interrupt, but it will still set the corresponding **RIS** bit when the interrupt would have occurred. The software can acknowledge the event by writing ones to the corresponding **IC** (Interrupt Clear) bit in the **GPIO_PORTA_IC_R** register. The **RIS** bits are read only, meaning if the software were to write to this register, it would have no effect. For example, to clear bits 2, 1, and 0 in the **GPIO_PORTA_RIS_R** register, we write a 0x07 to the **GPIO_PORTA_IC_R** register. Writing zeros into **IC** bits will not affect the **RIS** bits.

DIR	AFSEL	PMC	IS	IBE	IEV	IME	Port mode
0	0	0000	0	0	0	0	Input, falling edge trigger, busy wait
0	0	0000	0	0	1	0	Input, rising edge trigger, busy wait
0	0	0000	0	1	-	0	Input, both edges trigger, busy wait
0	0	0000	0	0	0	1	Input, falling edge trigger, interrupt
0	0	0000	0	0	1	1	Input, rising edge trigger, interrupt
0	0	0000	0	1	-	1	Input, both edges trigger, interrupt

Table 12.5. Edge-triggered modes.

For input signals we have the option of adding either a pull-up resistor or a pull-down resistor. If we set the corresponding **PUE** (Pull-Up Enable) bit on an input pin, the equivalent of a 13 k Ω to 30 k Ω resistor to +3.3 V power is internally connected to the pin. Similarly, if we set the corresponding **PDE** (Pull-Down Enable) bit on an input pin, the equivalent of a 13 k Ω to 35 k Ω resistor to ground is internally connected to the pin. We cannot have both pull-up and a pull-down resistor, so setting a bit in one register automatically clears the corresponding bit in the other register.

A typical application of pull-up and pull-down mode is the interface of simple switches. Using these modes eliminates the need for an external resistor when interfacing a switch. Compare the interfaces on Port A to the interfaces on Port B illustrated in Figure 12.4. The PA2 and PA3 interfaces will use software-configured internal resistors, while the PB2 and PB3 interfaces use actual resistors. The PA2 and PB2 interfaces in Figure 12.4a) implement negative logic switch inputs, and the PA3 and PB3 interfaces in Figure 12.4b) implement positive logic switch inputs.

CHECKPOINT 12.5

What do negative logic and positive logic mean in this context?

Show Answer

Help

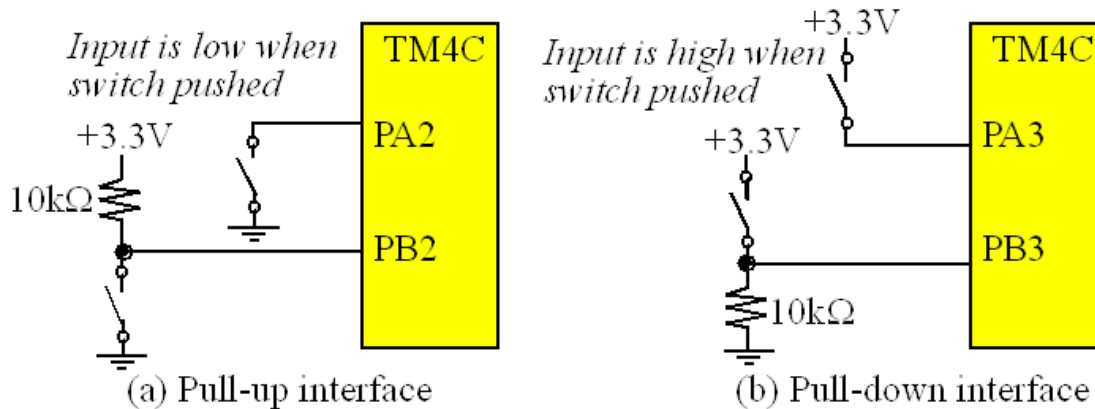


Figure 12.4. Edge-triggered interfaces can generate interrupts on a switch touch.

Using edge triggering to synchronize software to hardware centers around the operation of the trigger flags, **RIS**. A busy-wait interface will read the appropriate **RIS** bit over and over, until it is set. When the **RIS** bit is set, the software will clear the **RIS** bit (by writing a one to the corresponding **IC** bit) and perform the desired function. With interrupt synchronization, the initialization phase will arm the trigger flag by setting the corresponding **IME** bit. In this way, the active edge of the pin will set the **RIS** and request an interrupt. The interrupt will suspend the main program and run a special interrupt service routine (ISR). This ISR will clear the **RIS** bit and perform the desired function. At the end of the ISR it will return, causing the main program to resume. In particular, five conditions must be simultaneously true for an edge-triggered interrupt to be requested:

- The trigger flag bit is set (RIS)
- The arm bit is set (IME)
- The level of the edge-triggered interrupt must be less than BASEPRI
- The edge-triggered interrupt must be enabled in the NVIC_EN0_R
- The I bit, bit 0 of the special register PRIMASK, is 0

CHECKPOINT 12.6

What values do you write into DIR, AFSEL, PUE, and PDE to configure the switch interfaces of PA2 and PA3 in Figure 12.4?

Hide Answer


For PA2, we need input with pull-up. DIR bit 2 is low (input), AFSEL bit 2 is low (not alternate), PUE bit 2 high (pull-up) and PDE bit 2 low (not pull-down). For PA3, we need input with pull-down. DIR bit 3 is low (input), AFSEL bit 3 is low (not alternate), PUE bit 3 low (no pull-up) and PDE bit 3 high (pull-down).

Table 12.4 listed the registers for Port A. The other ports have similar registers. We will begin with a simple example that counts the number of rising edges on Port F bit 4 (Program 12.4). The initialization requires many steps. (a) The clock for the port must be enabled. (b) The global variables should be initialized. (c) The appropriate pins must be enabled as inputs. (d) We must specify whether to trigger on the rise, the fall, or both edges. In this case we will trigger on the rise of PF4. (e) It is good design to clear the trigger flag during initialization so that the first interrupt occurs due to the first rising edge after the initialization has been run. We do not wish to count a rising edge that might have occurred during the power up phase of the system. (f) We arm the edge-trigger by setting the corresponding bits in the **IM** register. (g) We establish the priority of Port F by setting bits 23 – 21 in the **NVIC_PRI7_R** register as listed in Table 9.2. We activate Port F interrupts in the NVIC by setting bit 30 in the **NVIC_EN0_R** register, Table 12.3. There is no need to unlock PF4.



MOOCs are interactive and subjects include computer science, public health,
Specifying which edge will interrupt | 12.4 E...

 <https://courses.edx.org/courses/UTAustinX/UT...>
(<https://twitter.com/edXOnline>)

 (<https://plus.google.com/108235383044095082735/posts>)

 (<http://youtube.com/user/edxonline>)
© 2014 edX, some rights reserved.

Terms of Service and Honor Code -
Privacy Policy (<https://www.edx.org/edx-privacy-policy>)