ECE375
External Interrupt

TA:

School of Electrical Engineering and Computer Science

Oregon State University

External Interrupts

- Understand Interrupts
- Demonstrate BumpBot using external Interrupts
- Explore the ATmega32U4 datasheet
- BumpBot counts each whisker and displays on LCD
- BumpBot clears both whiskers' counters on LCD

Interrupts Handling

Right Whisker
(Interrupt Occurs)
INT0

Go Forward
(Main Program)

Interrupts Handling

Right Whisker
(Interrupt Occurs)
INT0

Check Interrupt Vector
2 $0002 INT0

```
.org $0002
rcall    HitRight
         reti
```

Go Forward
(Main Program)

Interrupts Handling

Right Whisker
(Interrupt Occurs)
INT0

Check Interrupt Vector
2 $0002 INT0

```
.org $0002
rcall   HitRight
        reti
```

Go Forward
(Main Program)

Back up and turn away
from the object
(Interrupt Subroutine)

```
ret
```

Interrupts Handling

Right Whisker
(Interrupt Occurs)
INT0

Check Interrupt Vector
2 $0002 INT0

```
.org $0002
rcall   HitRight
        reti
```

Go Forward
(Main Program)

Back up and turn away
from the object
(Interrupt Subroutine)

ret

Interrupts Handling

Right Whisker
(Interrupt Occurs)
INT0

Check Interrupt Vector
2 $0002 INT0

```
.org $0002
rcall   HitRight
        reti
```

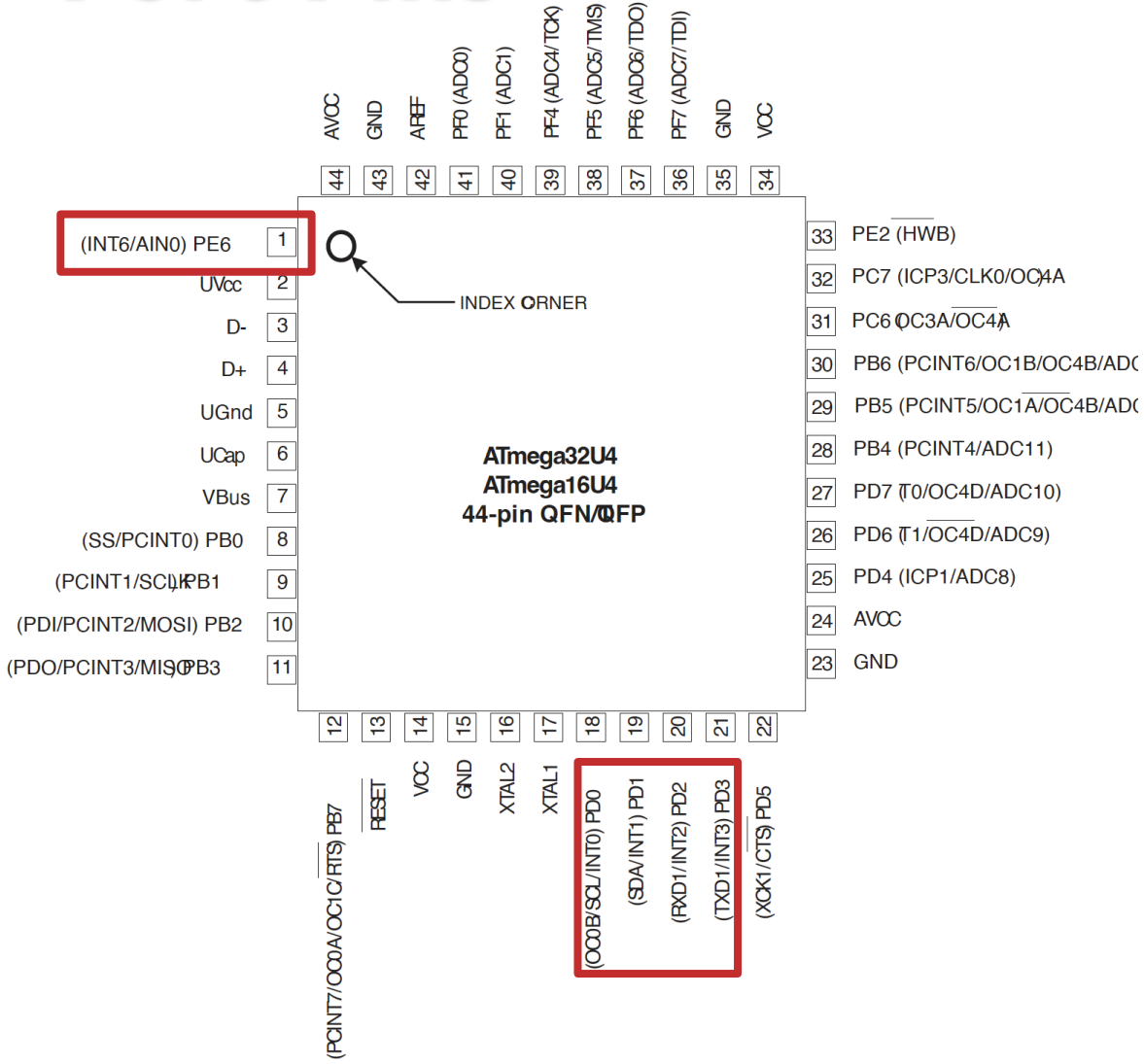
Go Forward
(Main Program)

Back up and turn away
from the object
(Interrupt Subroutine)

External Interrupts

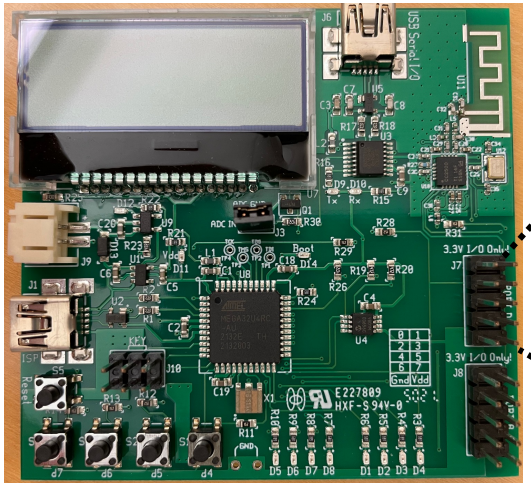
- The External Interrupts are triggered by the INT3:0 and INT6 pins.
  - PIND3:0 = INT3:0
  - PINE6 = INT6

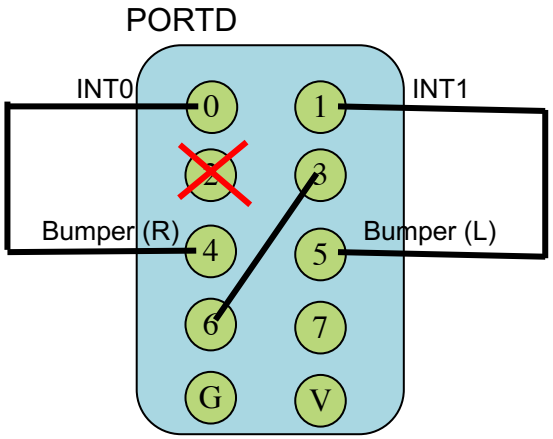
I/O Port Pins



Recall that we use PD4-7 for push buttons

Tekbot Bumper Switch Connection





To trigger interrupts with the push buttons,
plug jumpers into the board to connect:
PD0 <=> PD4
PD1 <=> PD5
PD3 <=> PD6

Do NOT jump PD2. The pushbuttons cannot pull the PD2 logic low.

External Interrupts

- The External Interrupts are triggered by the INT3:0 and INT6 pins.

- The External Interrupts can be triggered by a falling, a rising edge, or a low level.
  - EICRA (INT3:0) and EICRB (INT6)

External Interrupts

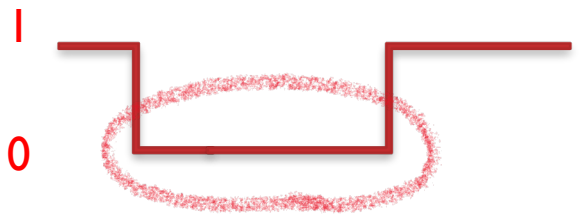
- The External Interrupts are triggered by the INT3:0 and INT6 pins.

- The External Interrupts can be triggered by a falling, a rising edge, or a low level.
  - EICRA (INT3:0) and EICRB (INT6)

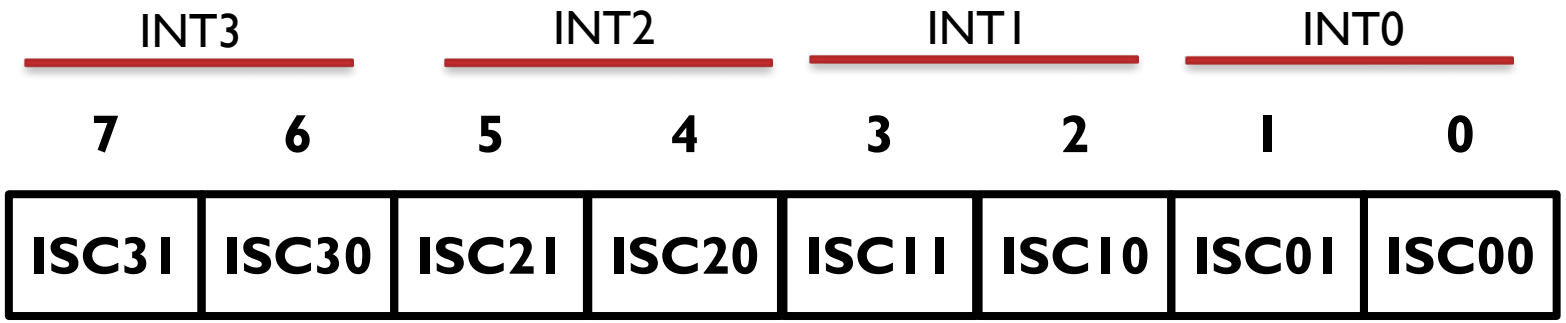
External Interrupts

- The External Interrupts are triggered by the INT3:0 and INT6 pins.

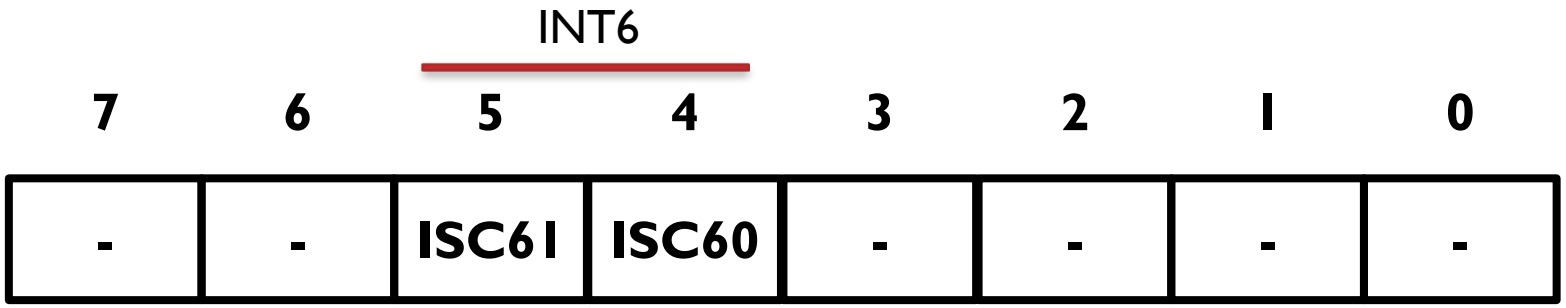
- The External Interrupts can be triggered by a falling, a rising edge, or a low level.
  - EICRA (INT3:0) and EICRB (INT6)



External Interrupt Control Register



EICRA



EICRB

ISC*n*1:0 External Interrupt *n* Sense Control Bits
00 – Low level generates an interrupt
01 – Any edge generates an interrupt request (for ISC3-0)
Any logical change generates an interrupt request (for ISC6)
10 - Falling Edge generates an interrupt request
11 – Rising Edge generates an interrupt request

External Interrupt Control Register

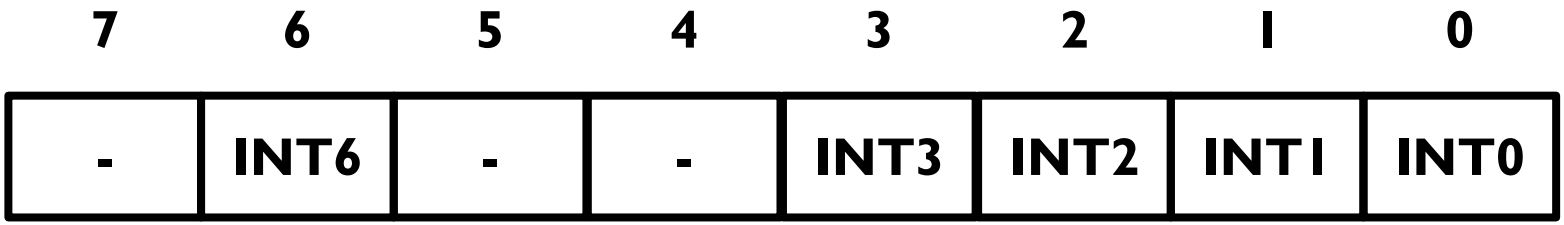
ISCn1	ISCn0	Description
0	0	The low level of INTn generates an interrupt request.
0	1	Any edge of INTn generates asynchronously an interrupt request.
1	0	The falling edge of INTn generates asynchronously an interrupt request.
1	1	The rising edge of INTn generates asynchronously an interrupt request.

Note: 1. n = 3, 2, 1, or 0.
When changing the ISCn1/ISCn0 bits, the interrupt must be disabled by clearing its Interrupt Enable bit in the EIMSK Register. Otherwise an interrupt can occur when the bits are changed.

INT0

```
ldi mpr, 0b0000_0010
sts EICRA, mpr
```

External Interrupt Mask Register



EIMSK

INT0

```
ldi mpr, 0b0000_0001

out EIMSK, mpr
```

Determining Source of Interrupt

- When an interrupt occurs each source of interrupt is mapped to a vector.

Vector #	Program Address	Source
1	$0000	RESET
2	$0002	External Interrupt Request 0 (INT0)
3	$0004	External Interrupt Request 1 (INT1)
4	$0006	External Interrupt Request 2 (INT2)
5	$0008	External Interrupt Request 3 (INT3)
…	…	…
8	$000E	External Interrupt Request 6 (INT6)
…	…	…
22	$002A	Timer/Counter0 Compare Match A
23	$002C	Timer/Counter0 Compare Match B
21	$002E	Timer/Counter0 Overflow
…	…	…

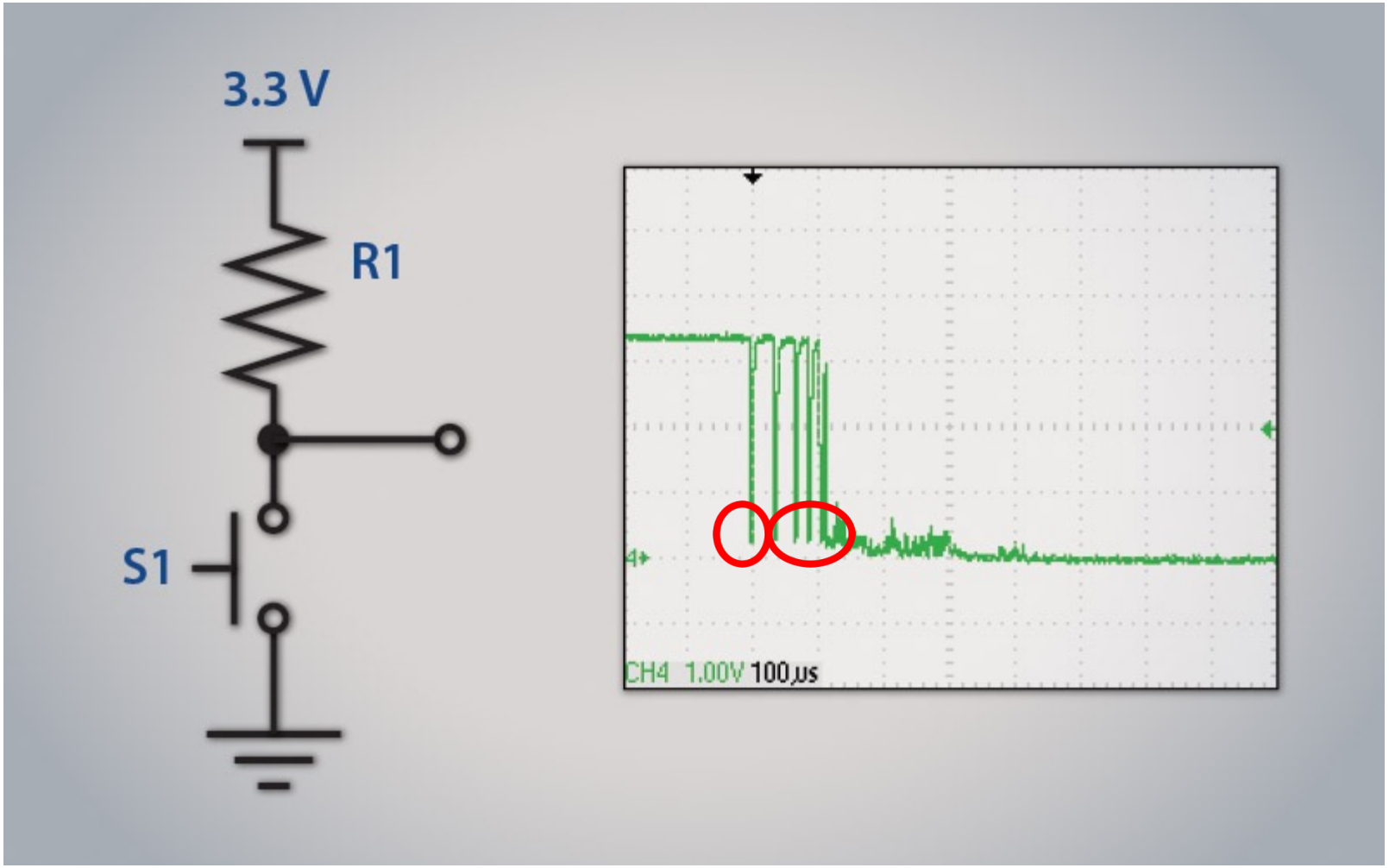
Priority

High

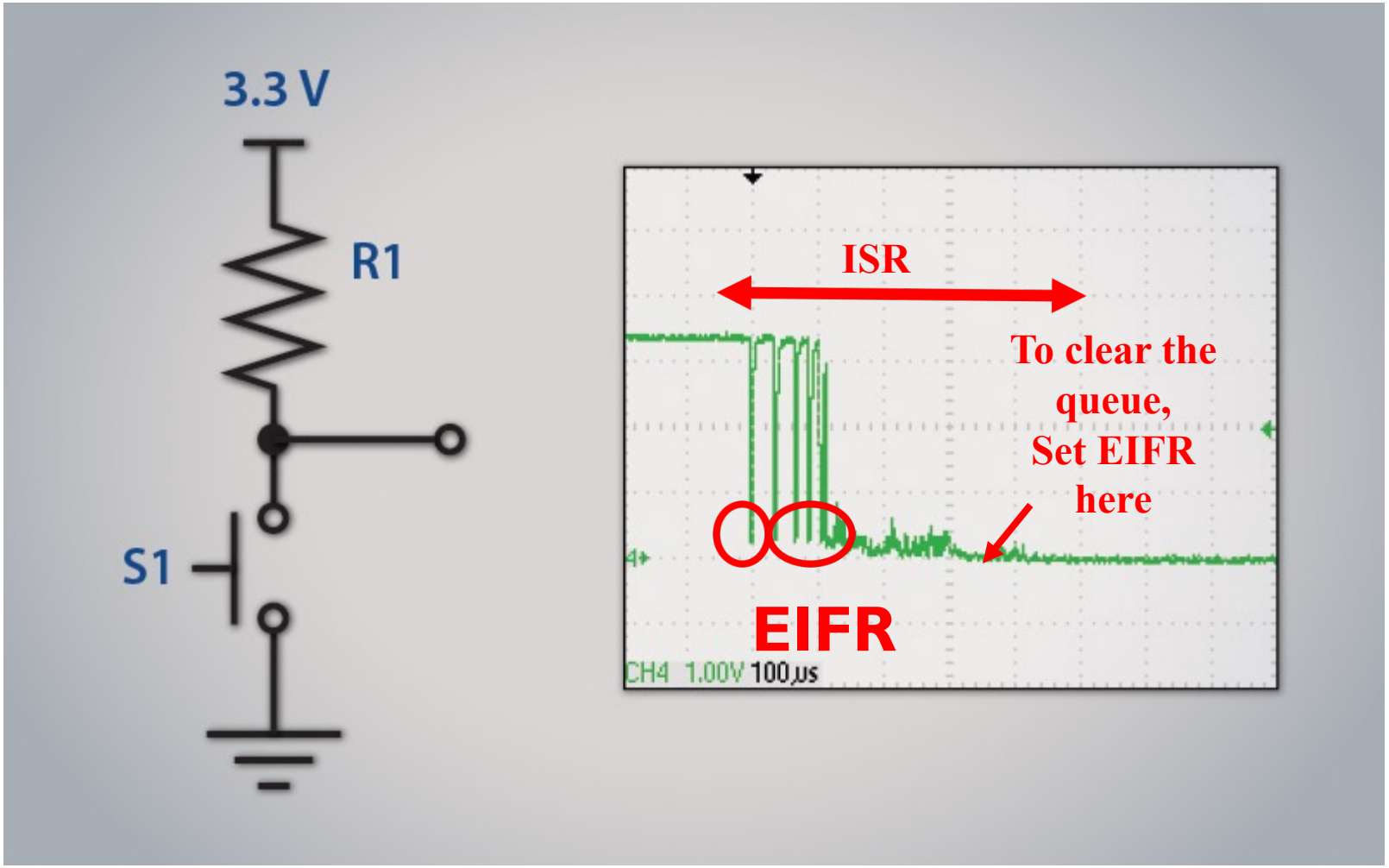
Low

There are 43 vectors!

Avoiding Queued Interrupts



Avoiding Queued Interrupts



External Interrupt Flag Register

- Write "1" in order to clear the queue to EIFR
  - ldi mpr, 0b0000_0001
  - out EIFR, mpr

ATmega32U4 I/O registers

- ATmega32U4 I/O registers for External Interrupts
  - External Interrupt Control Register A – EICRA
  - External Interrupt Control Register B – EICRB
  - External Interrupt Mask Register – EIMSK
  - External Interrupt Flag Register – EIFR

  - sei ; set interrupt

I/O Port Registers

These I/O registers can be accessed using `IN`/`OUT` instructions

Address	Name
0x17 (0x37)	Reserved
0x16 (0x36)	TIFR1
0x15 (0x35)	TIFR0
0x14 (0x34)	Reserved
0x13 (0x33)	Reserved
0x12 (0x32)	Reserved
0x11 (0x31)	PORTF
0x10 (0x30)	DDRF
0x0F (0x2F)	PINF
0x0E (0x2E)	PORTE
0x0D (0x2D)	DDRE
0x0C (0x2C)	PINE
0x0B (0x2B)	PORTD
0x0A (0x2A)	DDRD
0x09 (0x29)	PIND
0x08 (0x28)	PORTC
0x07 (0x27)	DDRC
0x06 (0x26)	PINC
0x05 (0x25)	PORTB
0x04 (0x24)	DDRB
0x03 (0x23)	PINB
0x02 (0x22)	Reserved
0x01 (0x21)	Reserved
0x00 (0x20)	Reserved

0x30 (0x50)	ACSR
0x2F (0x4F)	Reserved
0x2E (0x4E)	SPDR
0x2D (0x4D)	SPSR
0x2C (0x4C)	SPCR
0x2B (0x4B)	GPIOR2
0x2A (0x4A)	GPIOR1
0x29 (0x49)	PLLCSR
0x28 (0x48)	OCR0B
0x27 (0x47)	OCR0A
0x26 (0x46)	TCNT0
0x25 (0x45)	TCCR0B
0x24 (0x44)	TCCR0A
0x23 (0x43)	GTCCR
0x22 (0x42)	EEARH
0x21 (0x41)	EEARL
0x20 (0x40)	EEDR
0x1F (0x3F)	EECR
0x1E (0x3E)	GPIOR0
0x1D (0x3D)	EIMSK
0x1C (0x3C)	EIFR
0x1B (0x3B)	PCIFR
0x1A (0x3A)	Reserved
0x19 (0x39)	TIFR4
0x18 (0x38)	TIFR3

0x3F (0x5F)	SREG
0x3E (0x5E)	SPH
0x3D (0x5D)	SPL
0x3C (0x5C)	Reserved
0x3B (0x5B)	RAMPZ
0x3A (0x5A)	Reserved
0x39 (0x59)	Reserved
0x38 (0x58)	Reserved
0x37 (0x57)	SPMCSR
0x36 (0x56)	Reserved
0x35 (0x55)	MCUCR
0x34 (0x54)	MCUSR
0x33 (0x53)	SMCR
0x32 (0x52)	PLLFRQ
0x31 (0x51)	OCDR/ MONDR

Demo Check

- BumpBot Behavior using Interrupts
  - Need to avoid queued interrupts
- LCD displays two counters
  - Count Right/Left Whiskers
  - Implement clearing each counter
    - Hint: Use Bin2ASCII function in LCDDriver.asm to display decimal numbers.
    - It must be able to display both counters greater than 10.
    - Do not show any garbage data when increment/clear the counters.
- Implement 3 interrupts properly
  - INT0 and INT1 for counting Right/Left Whiskers
  - INT3 for clearing Right/Left whisker counters
  - Do not use INT2.

Checklists

- Demo Checklist
  - Standard BumpBot behavior observed
  - Must use interrupts, not polling
  - Queued interrupts explicitly avoided
  - Nested interrupts not enabled
  - Correctly configured INT0 – INT3 to use falling-edge sense control
  - Correctly wired PD0-7 to use interrupt

Questions?