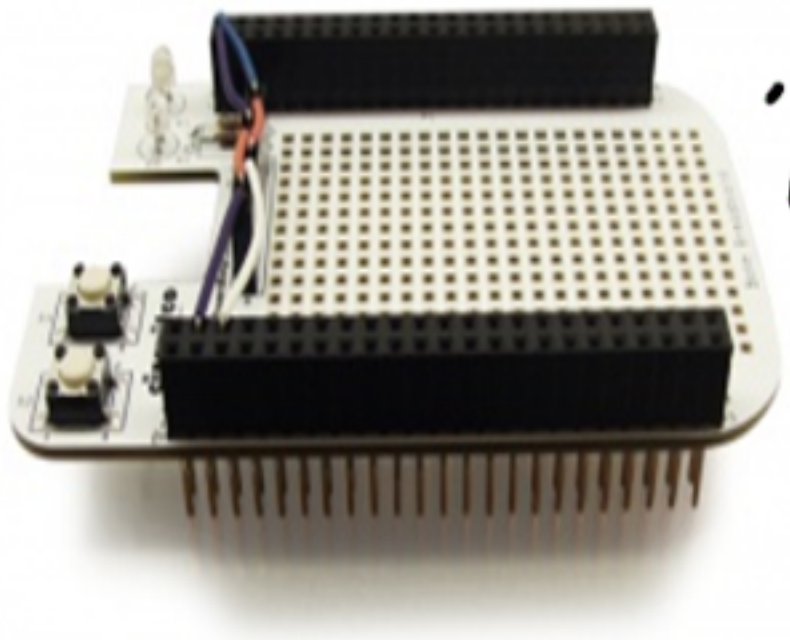
SE3910 – REAL TIME SYSTEMS

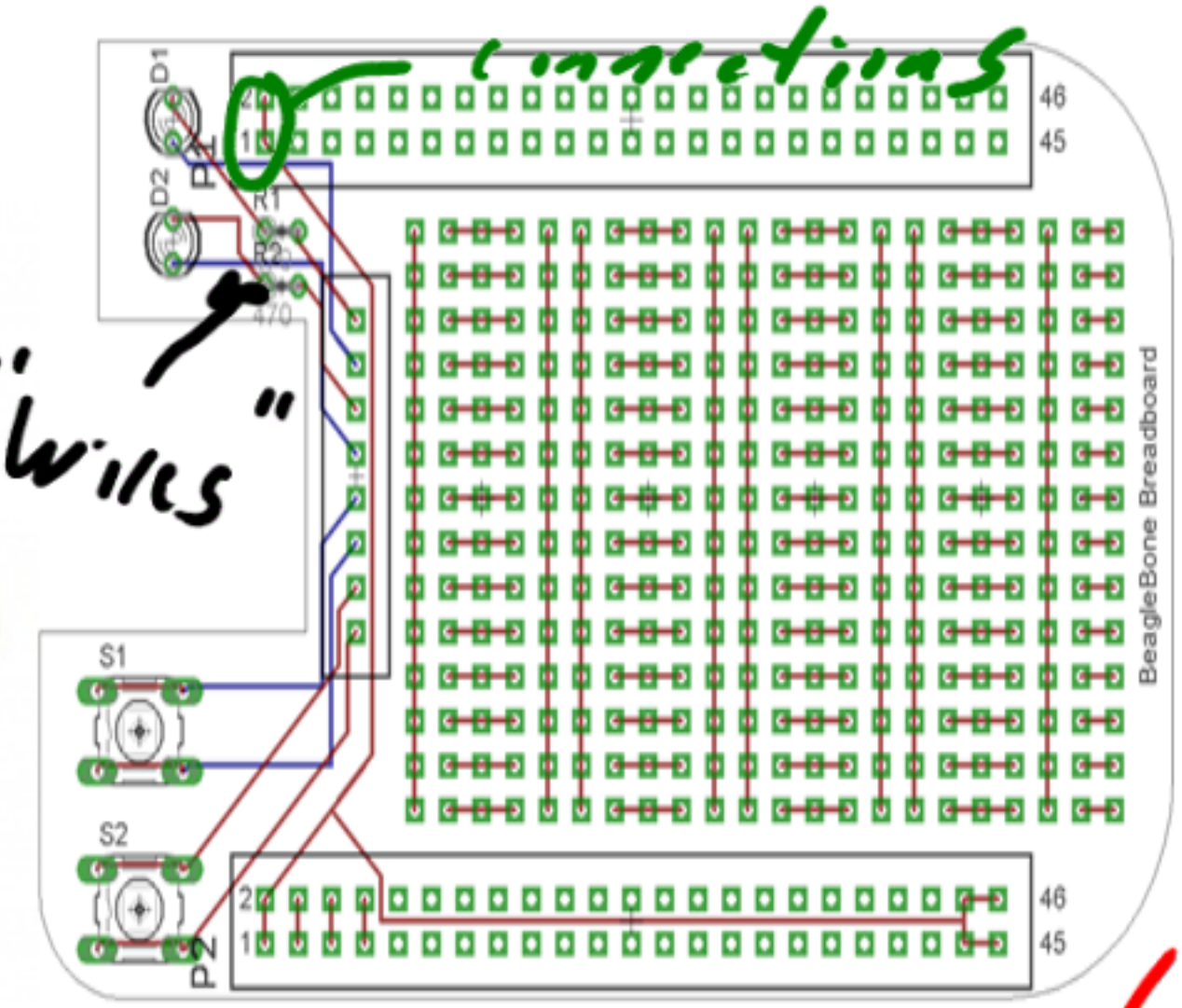
Hardware, Interrupts, and Vicious Dogs

OBJECTIVES

- Explain the concept of a cape
- Understand how to read a basic schematic
- Explain the concept of a dropping resistor
- Explain the concept of a pull up and a pull down resistor
- Explain the difference between polling and interrupts
- Explain how an interrupt service routine is handled
- Explain the concept of a system on a chip
- Explain the purpose for a watchdog timer

THE LAB CAPE



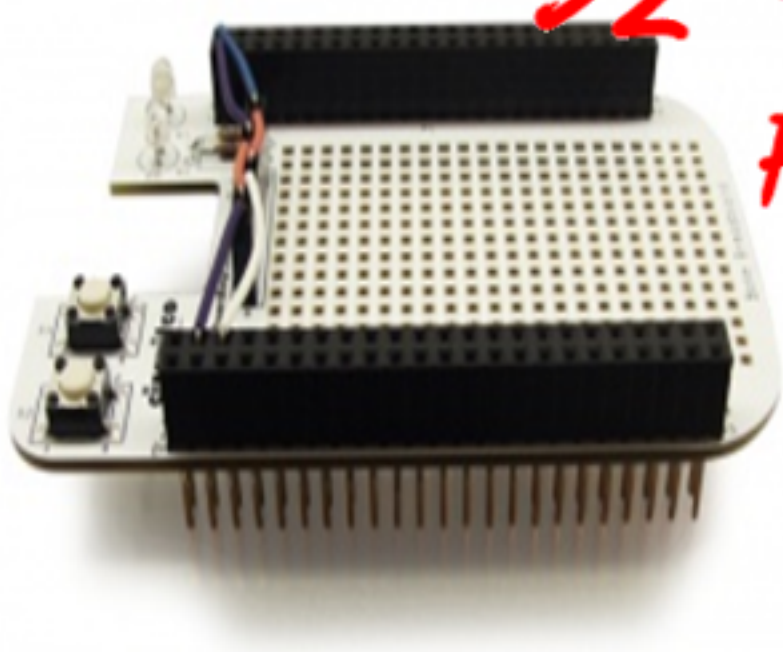


Connections

"wires"

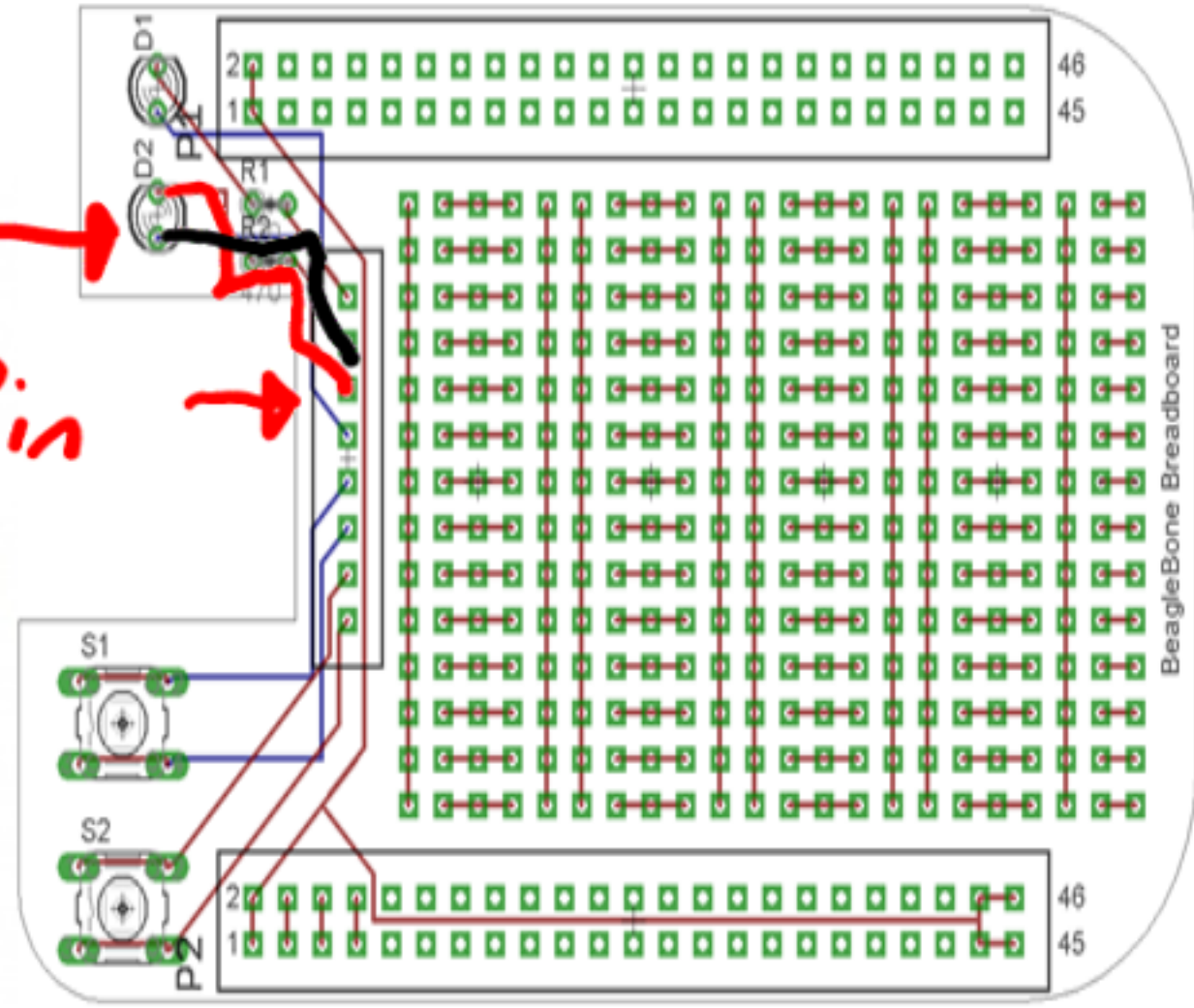
Board Layout

THE LAB CAPE

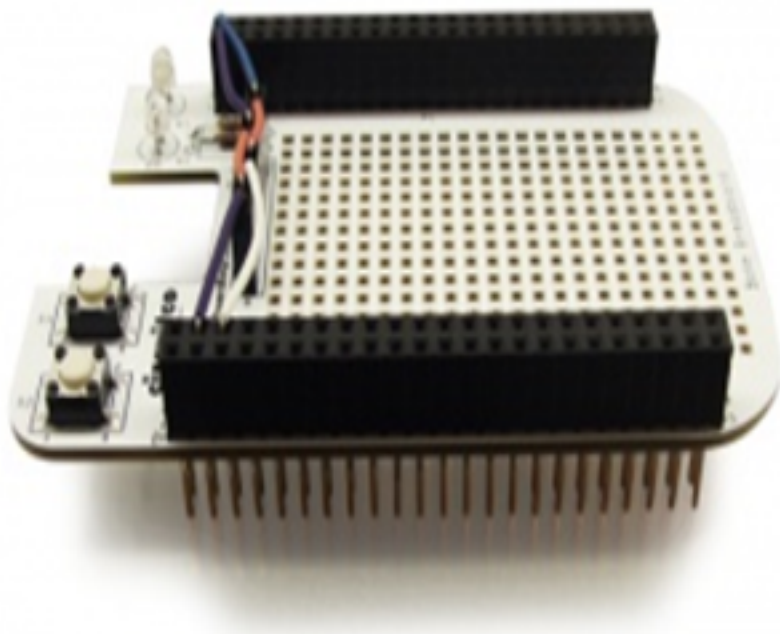


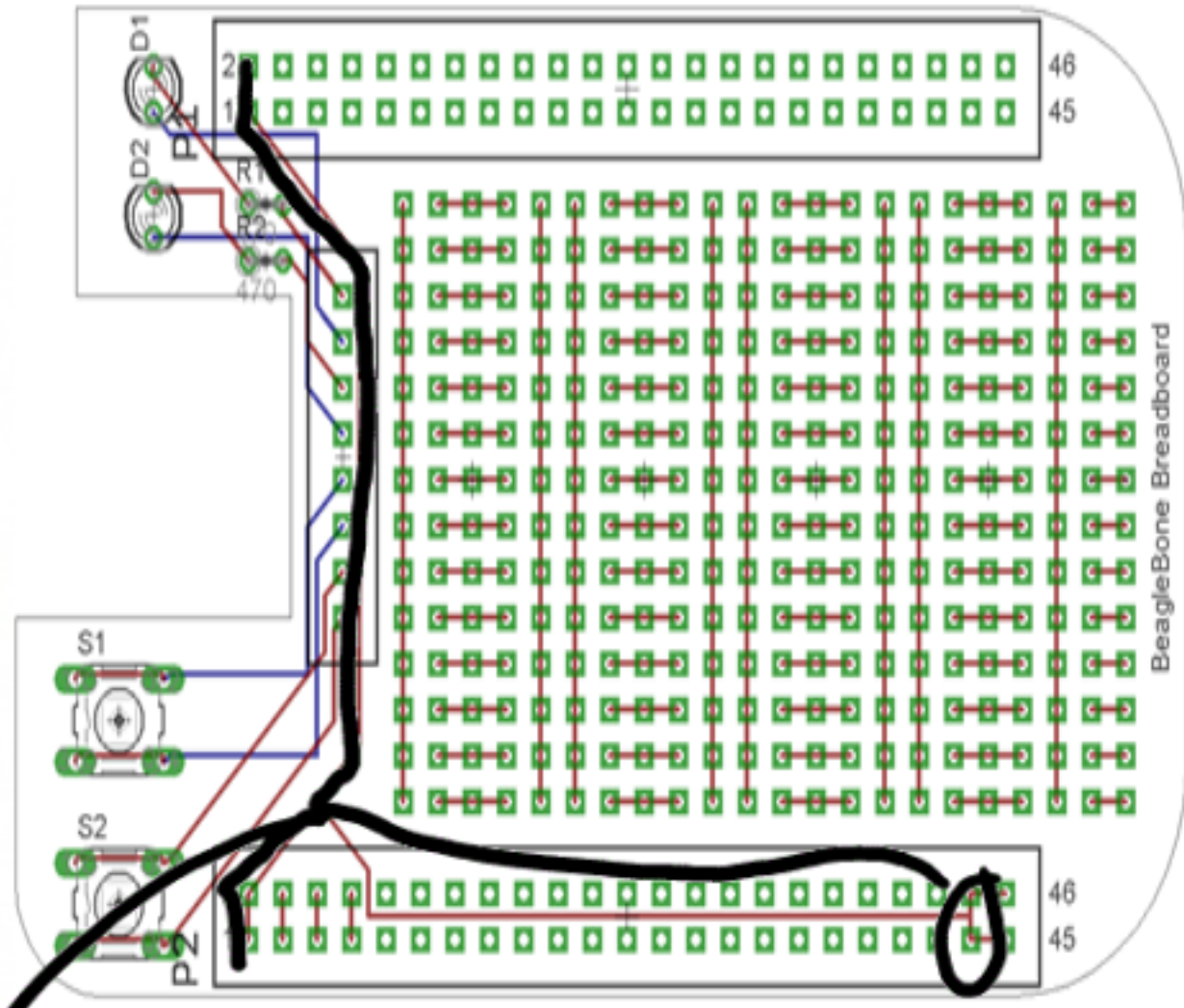
LED2

Pin



THE LAB CAPE

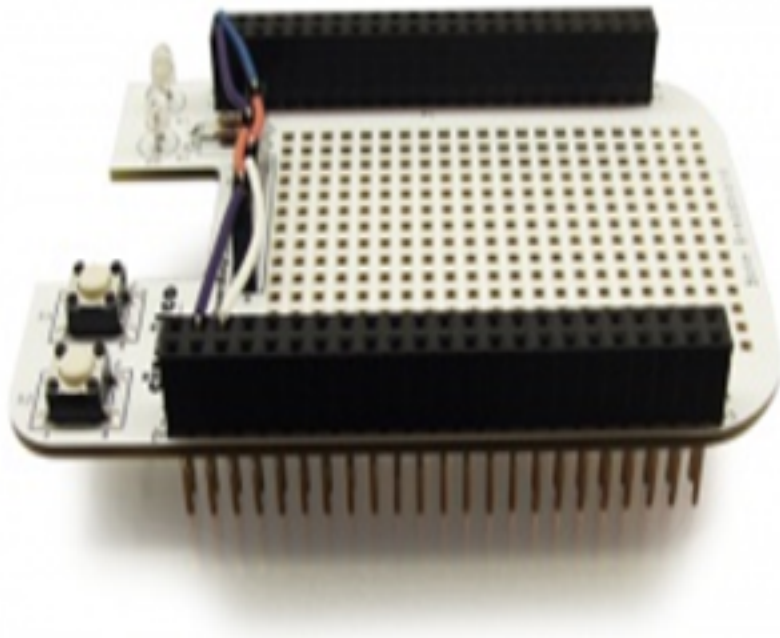


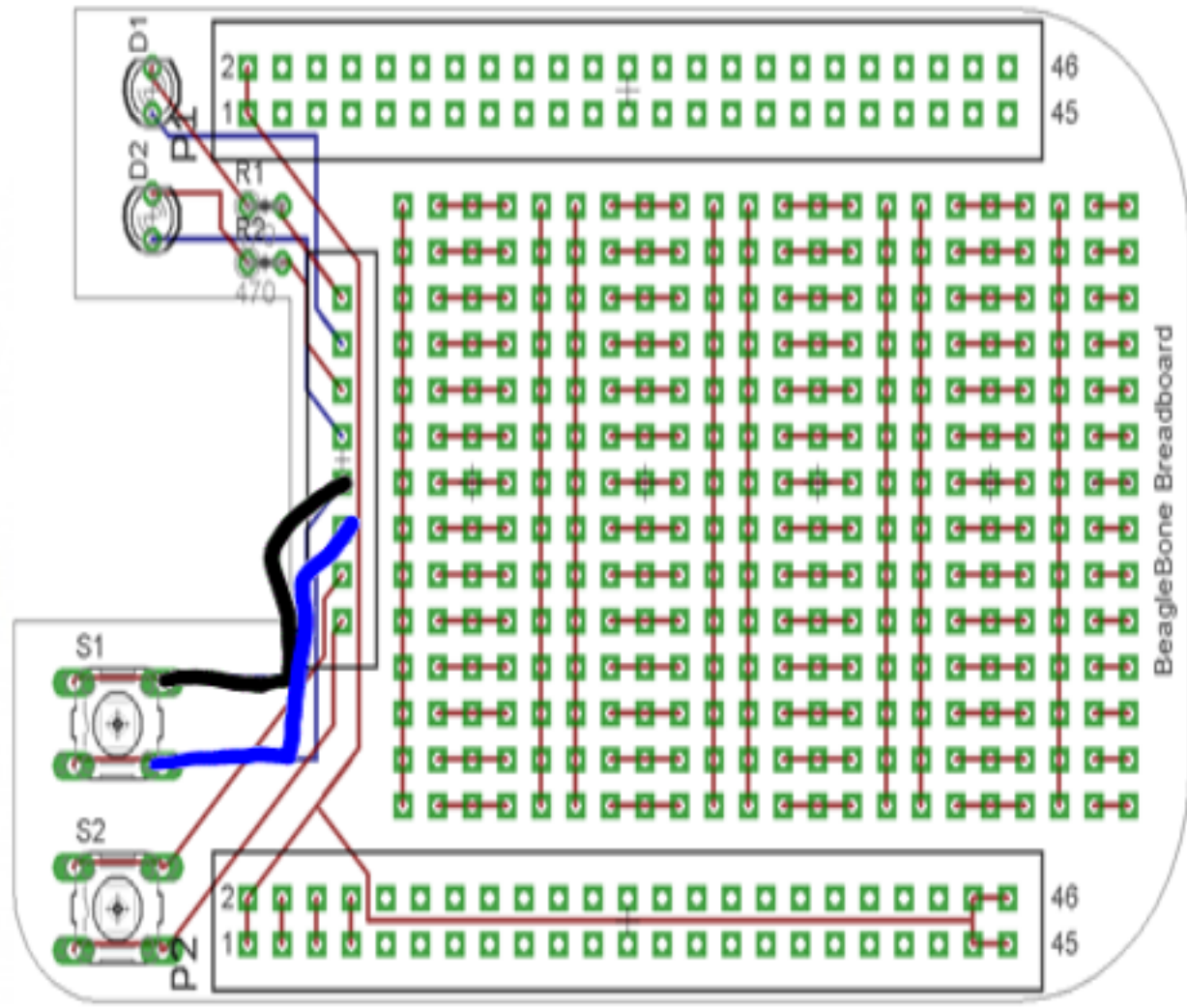


Ground Connection

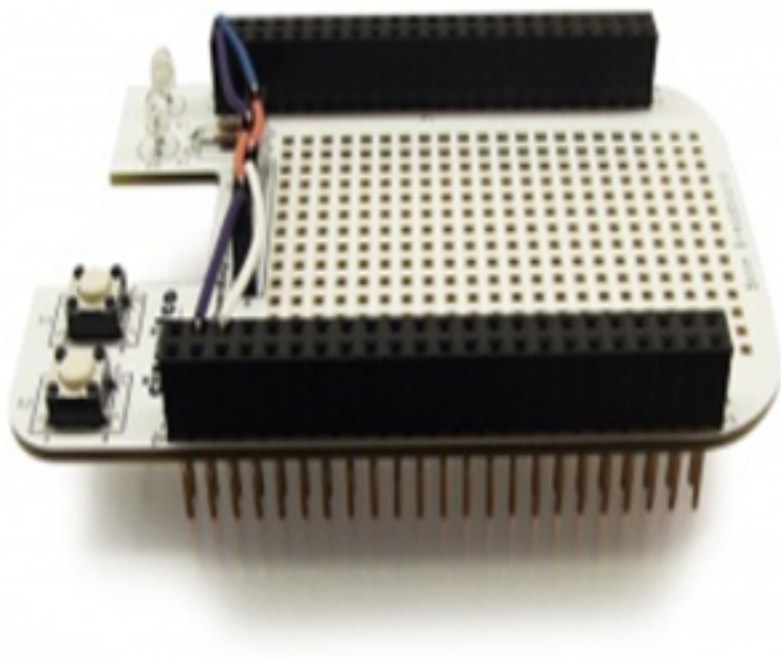
MSOE UNIVERSITY

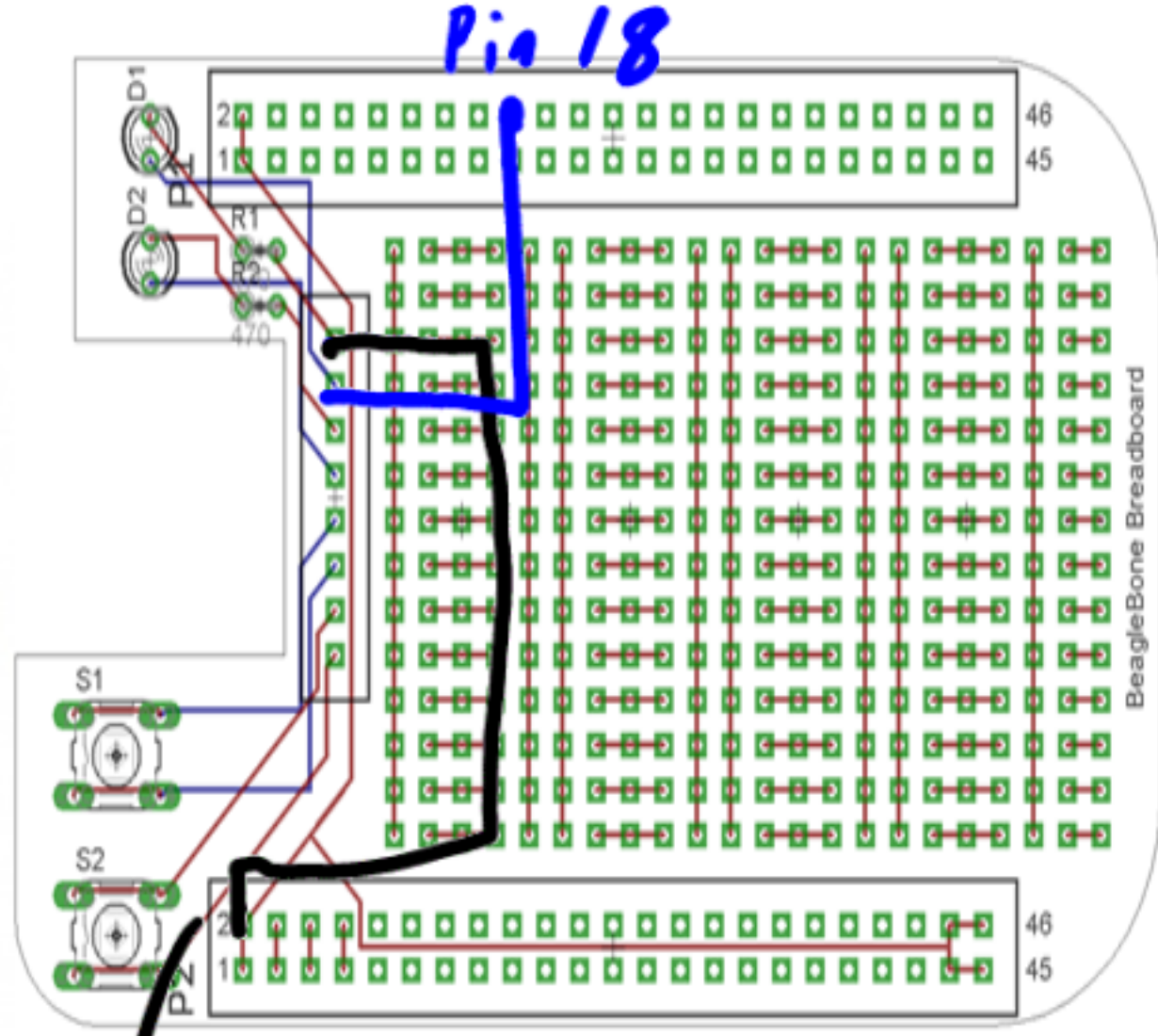
THE LAB CAPE





THE LAB CAPE





Pin 2

Names For Signals

LAB CAPE SCHEMATIC

P1

GPIO1_6	1 2		GPIO1_7
GPIO1_2	3 4		GPIO1_3
TIMER4	5 6		TIMER7
TIMER5	7 8		TIMER6
GPIO1_13	9 10		GPIO1_12
EHRPWM2B	11 12		GPIO1_26
GPIO1_15	13 14		GPIO1_14
GPIO1_27	15 16		GPIO2_1
EHRPWM2A	17 18		GPIO1_31
GPIO1_30	19 20		GPIO1_5
GPIO1_4	21 22		GPIO1_1
GPIO1_0	23 24		GPIO1_29
GPIO2_22	25 26		GPIO2_24
GPIO2_23	27 28		GPIO2_25
UART5_CTSN	29 30		UART5_RTSN
UART4_RTSN	31 32		UART3_RTSN
UART4_CTSN	33 34		UART3_CTSN
UART5_TXD	35 36		UART5_RXD
GPIO_12	37 38		GPIO2_13
GPIO2_10	39 40		GPIO2_11
GPIO2_8	41 42		GPIO2_9
GPIO2_6	43 44		GPIO2_7
	45 46		

FEMALE HEADER 2x23

DGND DGND

P2

VDD_3V3EXP	1 2		VDD_3V3EXP
VDD_5V	3 4		VDD_5V
SYS_5V	5 6		SYS_5V
PWR_BUT	7 8		SYS_RESETN
UART4_RXD	9 10		GPIO1_28
UART4_TXD	11 12		EHRPWM1A
UART3_RXD	13 14		EHRPWM1B
I2C1_SCL	15 16		I2C1_SDA
I2C2_SCL	17 18		I2C2_SDA
UART2_TXD	19 20		UART2_RXD
UART3_TXD	21 22		UART1_TXD
GPIO3_21	23 24		UART1_RXD
GPIO3_19	25 26		SPI1_CS0
SPI1_D0	27 28		SPI1_D1
SPI1_SCLK	29 30		VDD_ADC
AIN4	31 32		
AIN6	33 34		AIN5
AIN2	35 36		AIN3
AIN0	37 38		AIN1
CLKOUT2	39 40		GPIO0_7
	41 42		
	43 44		
	45 46		

FEMALE HEADER 2x23

AGND

DGND DGND

P3

8-PIN HEADER

R1 470 D1

R2 470 D2

S1

S2

TH1 TH2 TH3 TH4

TH5 TH6 TH7 TH8

TH9 TH10

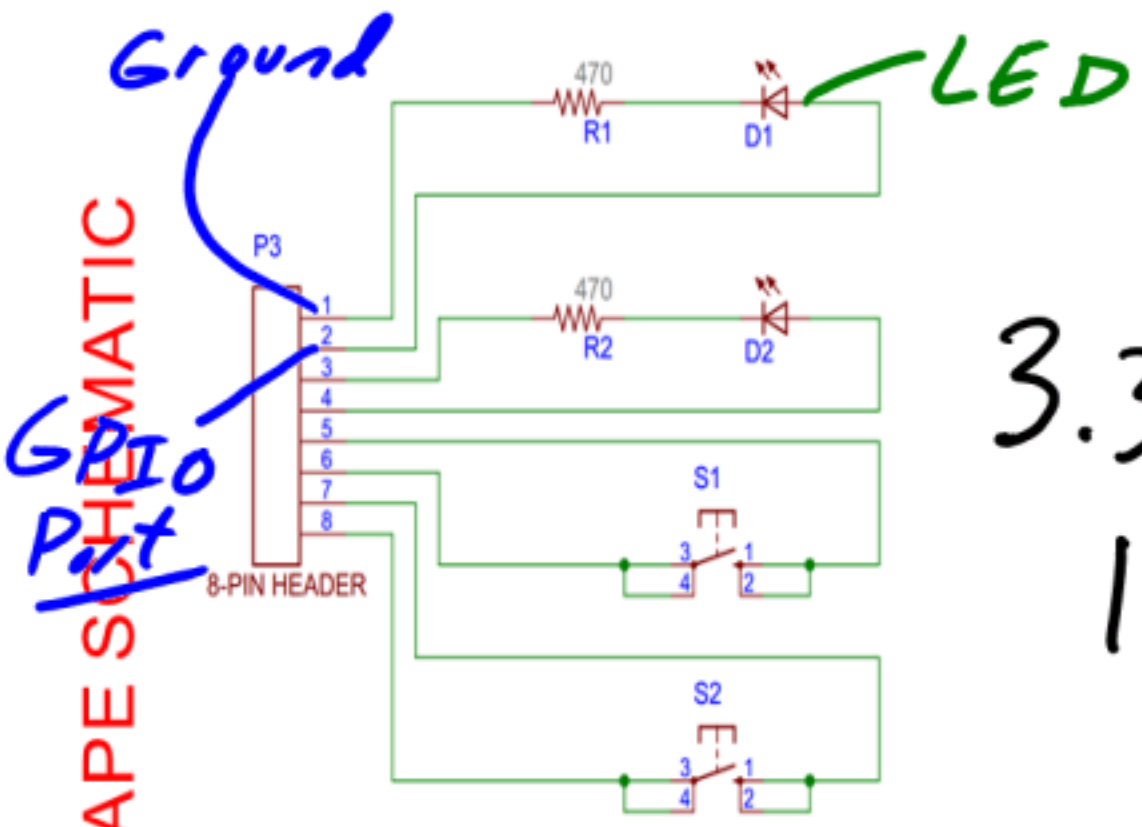
TITLE: Proto

Document Number:

REV:

MSOE UNIVERSITY

LAB CAPE SCHEMATIC



ITM	QTY	REF	PART	DESCRIPTION	DISTR	PART #
1	1		PCB	2-LAYER 3.4" x 2.15"		
2	2	P1, P2	2x23 HDR FE	DUAL - STRAIGHT SOCKET STRIP .025" SQ. PINS	MLE	SSHQ-123-D-08-F-LF
3	1	P3	8 PIN HDR FE	CONN HEADER FEMALE 8 POS 0.1" GOLD	DK	PPPC081LFBN-RC
4	2	S1, S2	BUTTONS	SWITCH TACT SPST-NO .05A 24V	DK	B3F-1000
5	2	D1, D2	LEDS	LED GREEN 3MM 568NM 20mA 2.12V	DK	WP7104SGC
6	2	R1, R2	RESISTOR	RES 470 OHM 1/4W 5% CF MINI	DK	CFM14JT470R
7	1	BR1	BREADBOARD	WHT ADHSV SLDRLSS BRDBRD 170 TIE PT 1.8" x 1.37"	PLU	1490
8	1		JMPR WIRE KIT	10 x 14 LNGTH .1" .2" .3" .4" .5" .6" .7" .8" .9" 1" 2" 3" 4" 5"	SQR	

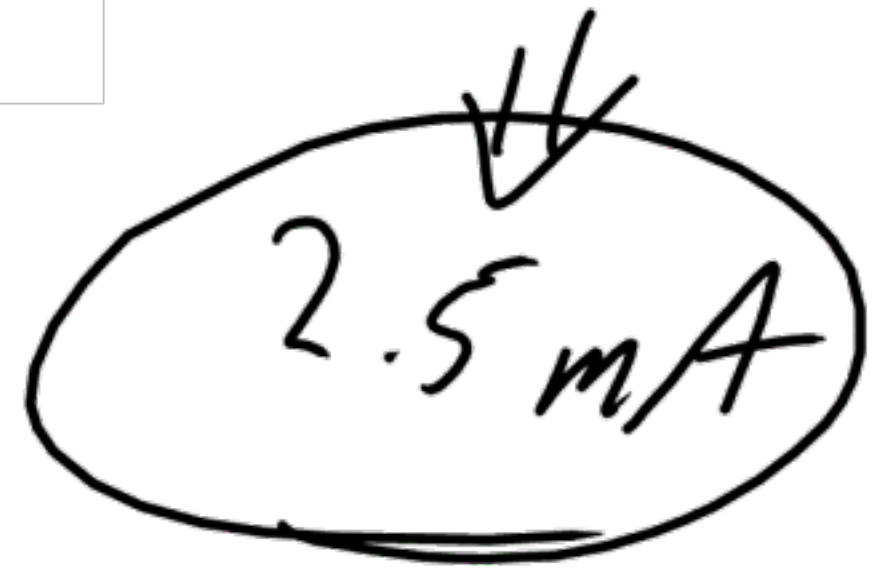
$$V = IR$$

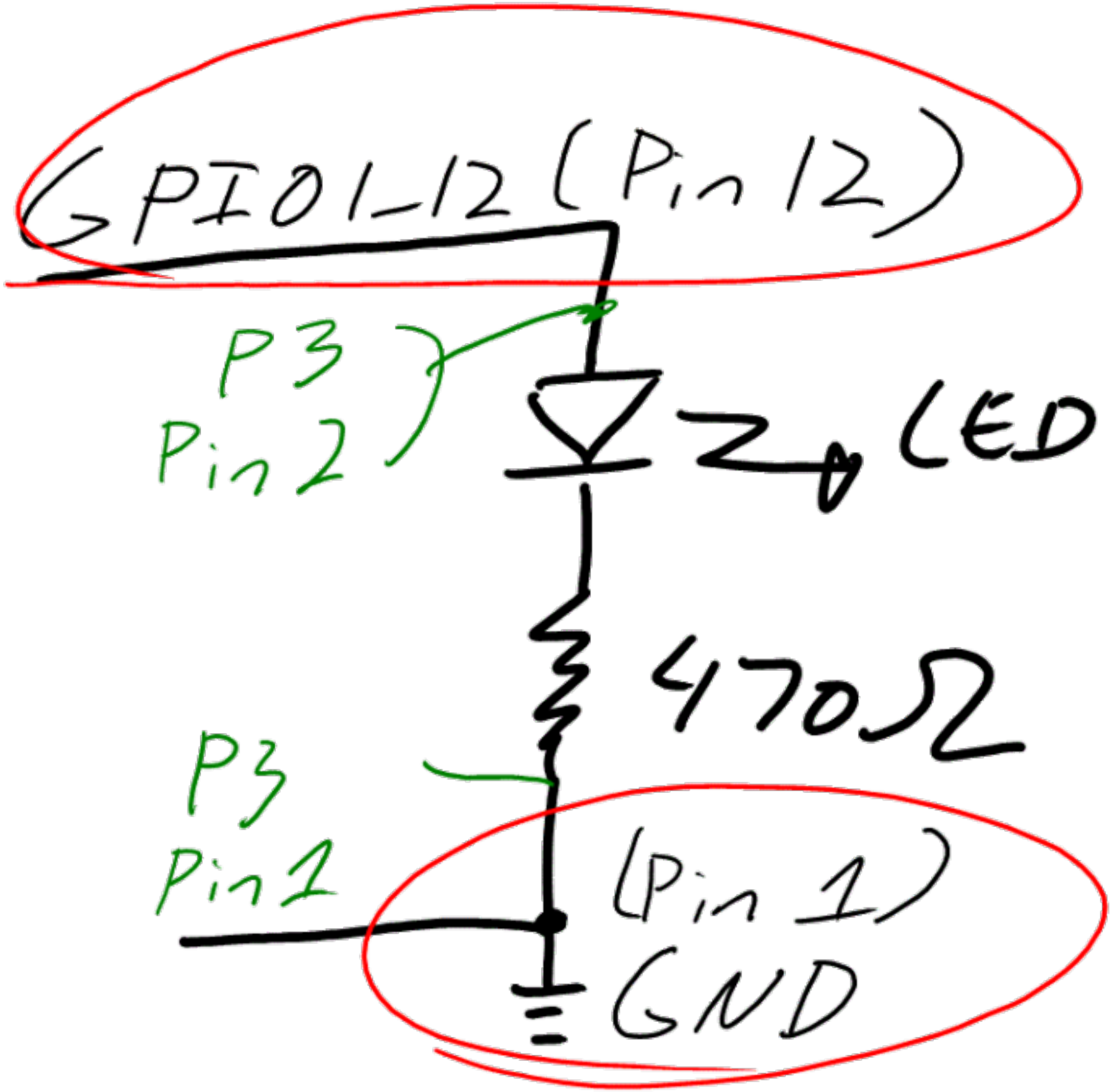
$$3.3 - 2.12 = I \times 470$$

$$1.18 = 470I$$

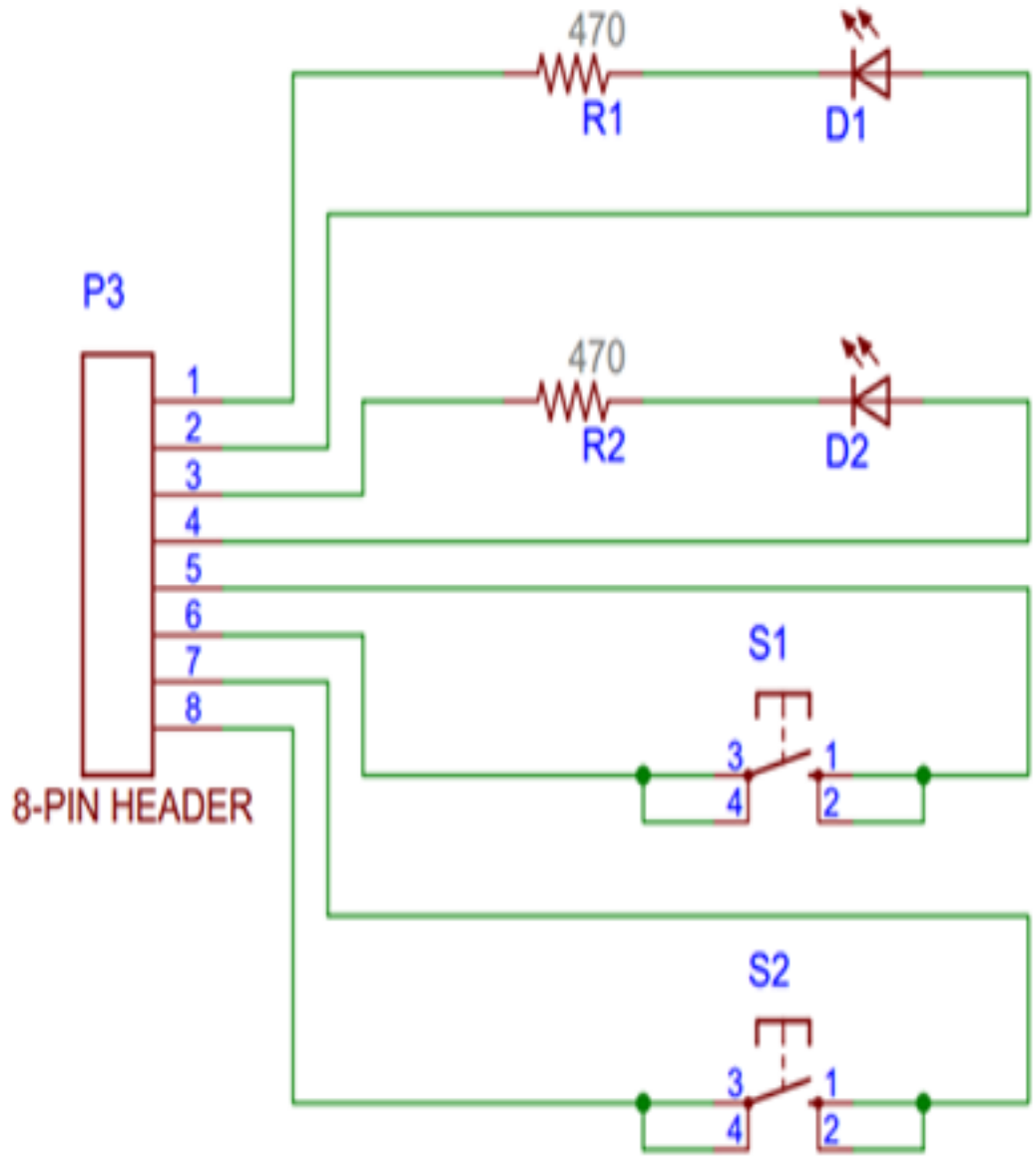
$$\frac{1.18}{470} = I$$

$$I = .0025$$




GPIO1_12 (Pin 12)
P3
Pin 2
LED
470Ω
P3
Pin 1
(Pin 1)
GND

THE SCHEMATIC WE BUILT WITH THE PULL UP RESISTOR



POLLING VERSUS INTERRUPTS

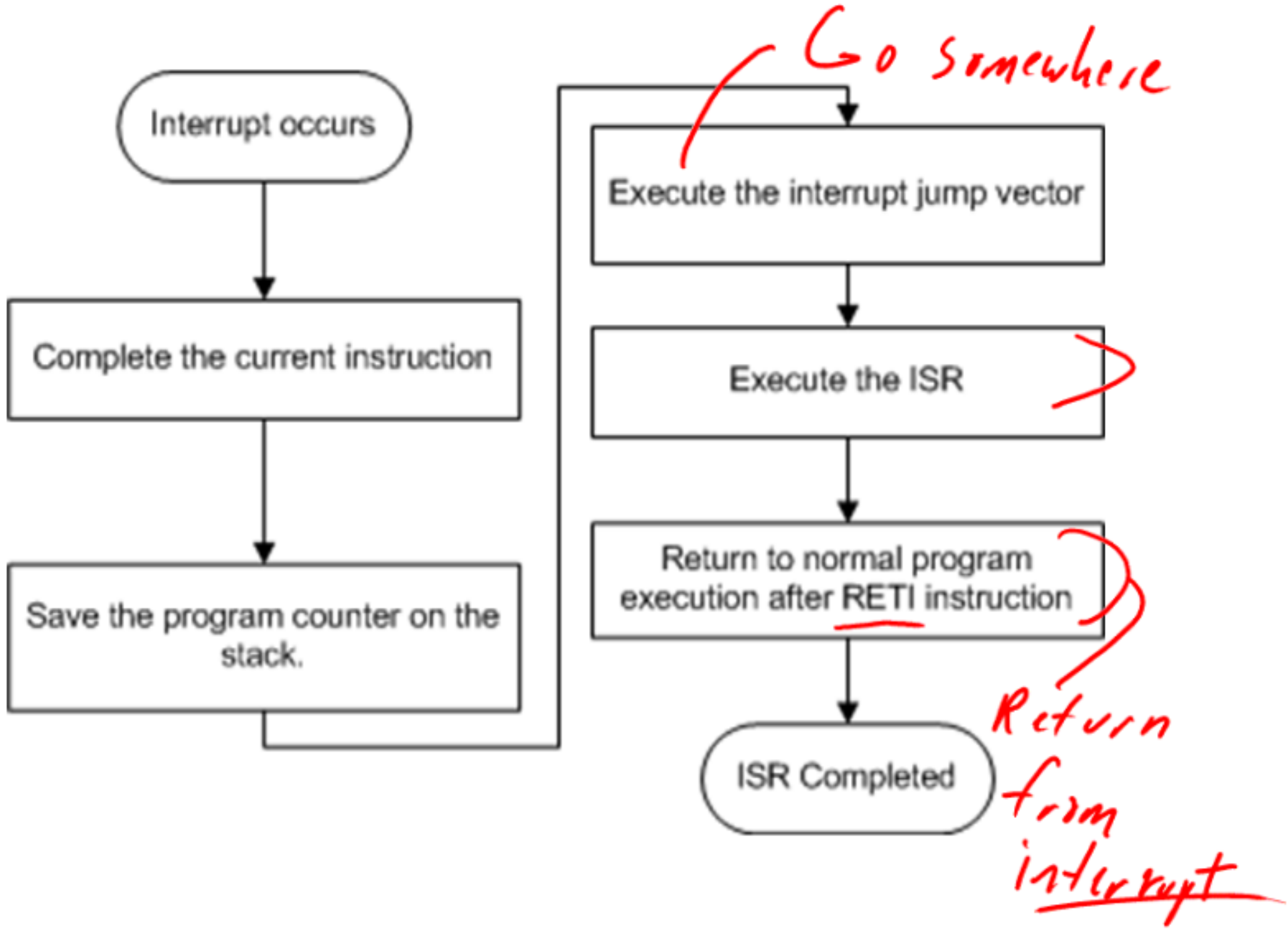
- Polled IO System
  - the status of the I/O device is checked periodically(regularly)
  - I/O activity is software controlled; only
  - accessible status and data registers are needed in the hardware side.
- Interrupt Driven
  - Latency can be made less uncertain without increasing the loading on the CPU
  - Question: How do we manage multiple interrupts?

DEFINITIONS

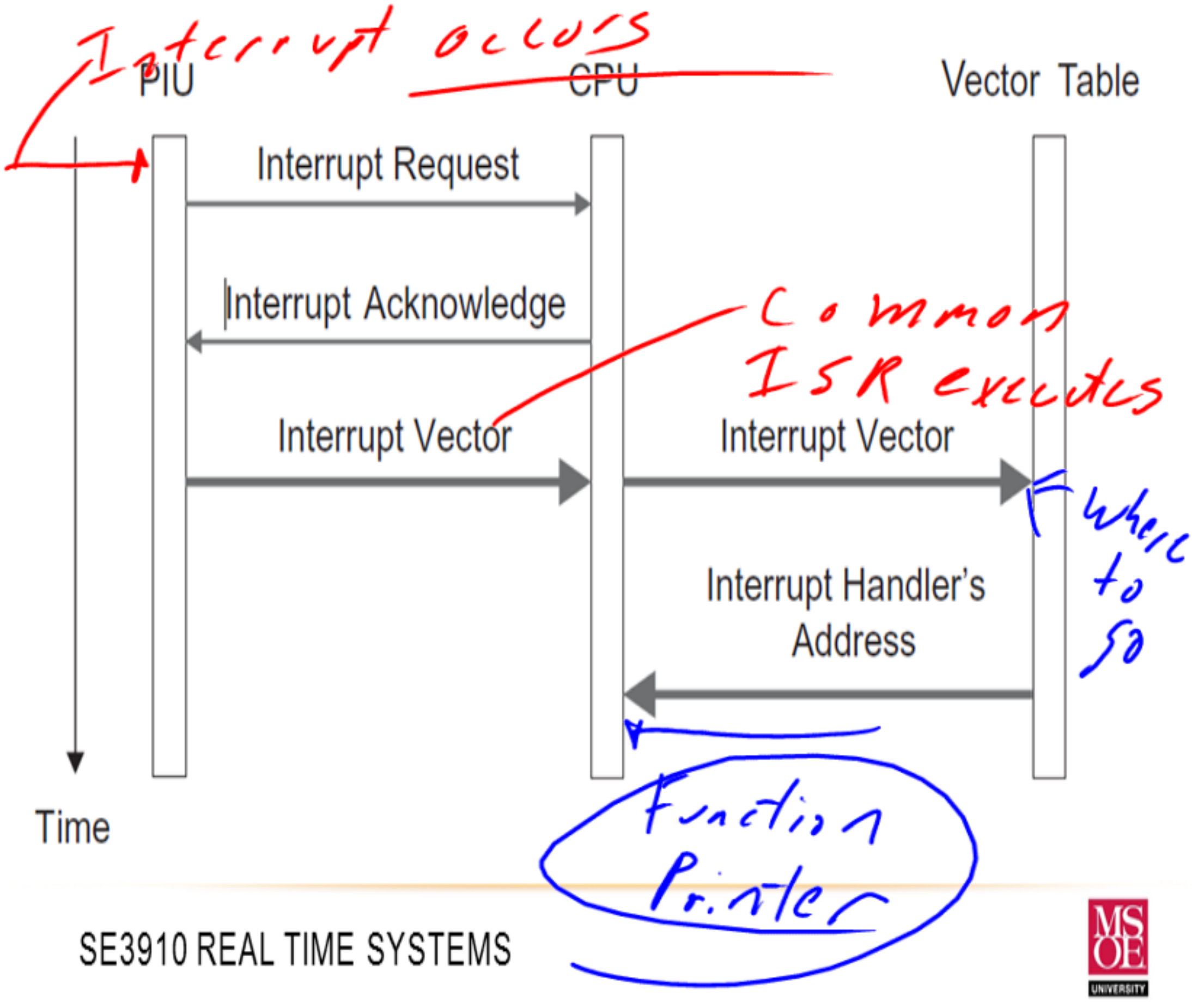
- Interrupt — Something happening
  - An event in hardware that triggers the processor to jump from its current program counter to a specific point in the code.
- Interrupt Service Routine (ISR)
  - The function that is called or the particular assembly code that is executed when the interrupt happens is called the Interrupt Service Routine (ISR).
- Interrupt flag (IFG)
  - this is the bit that is set that triggers the interrupt, leaving the interrupt resets this flag to the normal state.
- Interrupt Enable
  - Control bit that tells the processor that a particular interrupt should or should not be ignored.
- Interrupt Vector Table
  - A table in memory which maps ISRs to interrupts.

INTERRUPT

ISR HANDLING PROCESS



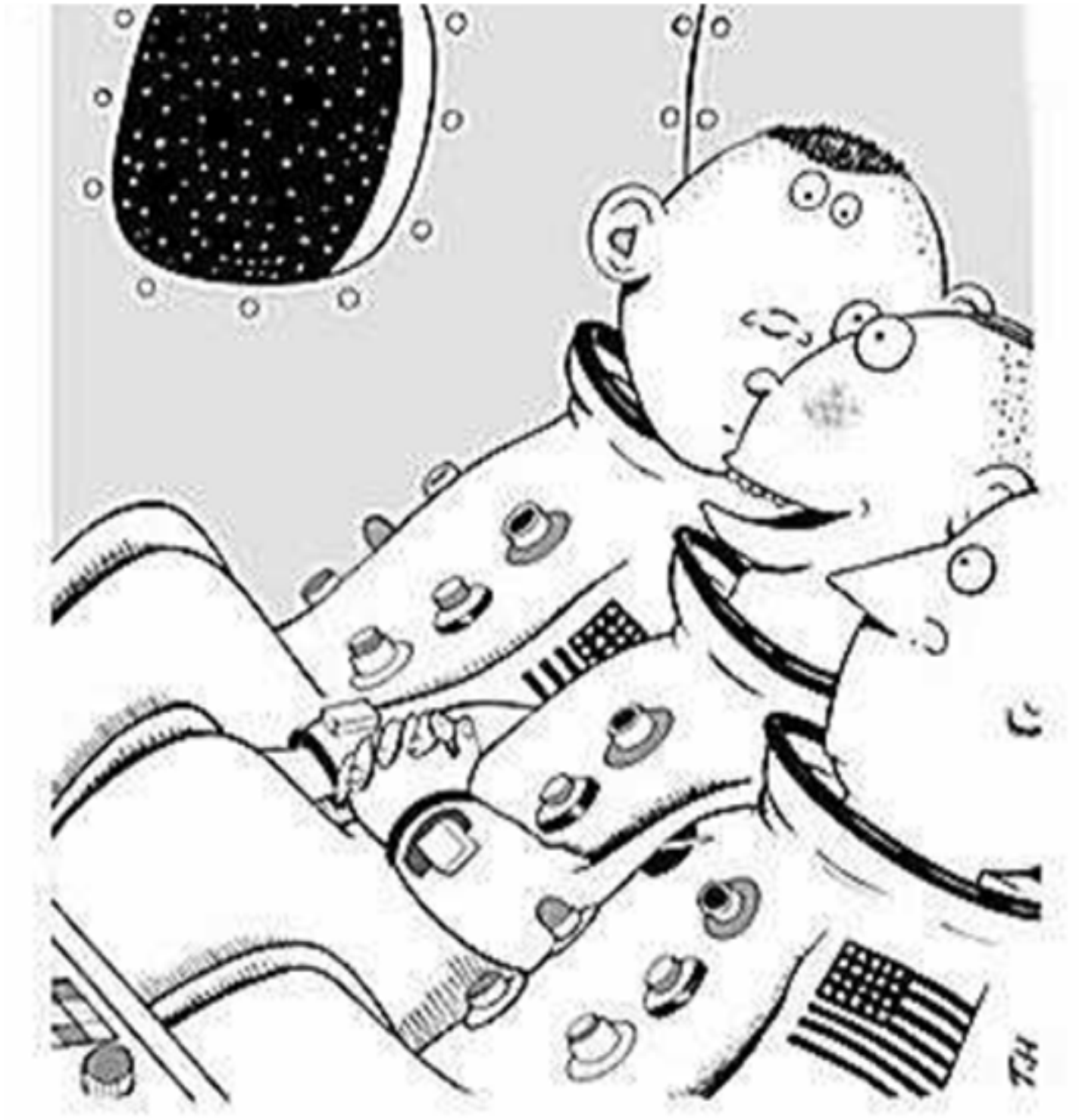
INTERRUPT HANDLING WITH PIU



INTERRUPT SERVICE ROUTINE HANDLING

- The interrupt-request line is activated.
- The interrupt request is latched by the CPU hardware (~).
- The processing of the ongoing instruction is completed (~).
- The content of program counter register (PCR) is pushed to stack.
- The content of status register (SR) is pushed to stack.
- The PCR is loaded with the interrupt handler's address.
- The interrupt handler is executed (~).
- The original content of SR is popped from stack.
- The original content of PCR is popped from stack.

POLLING

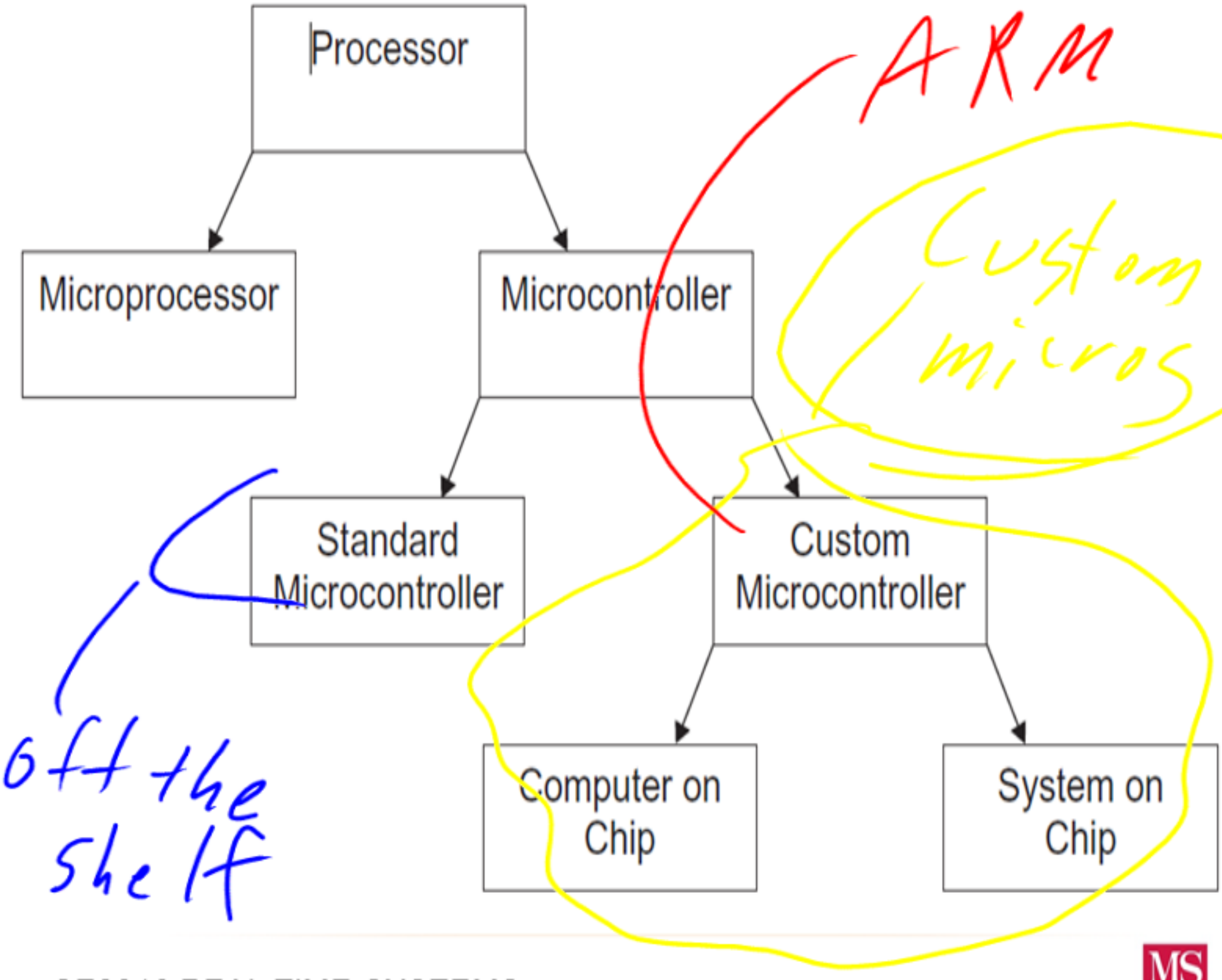


"Are we there yet? Are we there yet? Are we there yet?"

INTERRUPTS

No! Shut up and I'll tell you when we are there!

THE RELATIONSHIP BETWEEN PROCESSOR TECHNOLOGIES



THE CLEMENTINE

- In 1994, a deep space probe, the Clementine, was launched to make observations of the moon and a large asteroid (1620 Geographos).
- After months of operation, a software exception caused a control thruster to fire for 11 minutes, which depleted most of the remaining fuel and caused the probe to rotate at 80 RPM.
- Control was eventually regained, but it was too late to successfully complete the mission.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and incur significant cost impact.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software and [illegible] be truly robust.
- In [illegible]s, embedded devices operate in [illegible]on and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injur

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injur

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software and failures to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injury

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In most cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injury

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software and [illegible] be truly robust.
- In [illegible]s, embedded devices operate in [illegible]on and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injur

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injur

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software and failures to be truly robust.
- In many applications, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injury.

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In most cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injury

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software and [illegible] be truly robust.
- In [illegible]s, embedded devices operate in [illegible]on and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injur

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injur

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software and failures to be truly robust.
- In many cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injury

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software anomalies to be truly robust.
- In most cases, embedded devices operate in total isolation and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injury

WHY USE A WATCHDOG

- Embedded systems must be able to cope with both hardware and software and [illegible] be truly robust.
- In [illegible]s, embedded devices operate in [illegible]on and are not accessible to an operator.
- Manually resetting a device in this scenario when its software "hangs" is not possible.
- In extreme cases, this can result in damaged hardware or loss of life and injur

WATCHDOG TIMER STRUCTURE