TCNT is 16-bit up counter (see **TSCR2**)
PTT is 8-bit bi-directional I/O port
DDRT is the associated direction register for Port T (0 means input, 1 means output)
PTM is 6-bit bi-directional I/O port
DDRM is the associated direction register for Port M (0 means input, 1 means output)

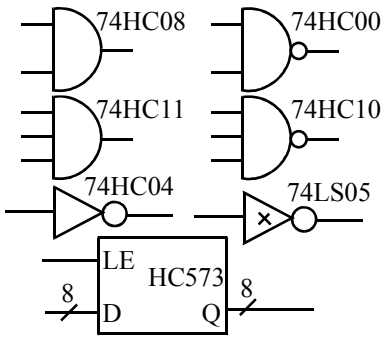
TSCR1 is a timer control register
bit 7 **TEN**, 1 means allow timer to function normally, 0 means disable timer including **TCNT**
TSCR2 is 8-bit timer control register
bits 2,1,0 **PR2**, **PR1**, **PR0**, select rate, let **n** be the 3-bit number
without PLL TCNT is 4MHz/2^{n}, with PLL TCNT is 24MHz/2^{n}, **n** ranges from 0 to 7

CRGFLG real time interrupt flag register
bit 7 **RTIF** real time interrupt flag, set on RTI timeout, cleared by write to this register with bit set
CRGINT real time interrupt control register
bit 7 **RTIE** real time interrupt enable, 1 means interrupt on RTIF, 0 means RTI interrupts will not occur
RTICTL real time interrupt control register, M clock is 4 MHz
bits 6-4 **RTR6**, **RTR5**, **RTR4**, select rate, let **n** be the 3-bit number, **n** ranges from 1 to 7
bits 3-0 **RTR3**, **RTR2**, **RTR1**, **RTR0**, select rate, let **m** be the 4-bit number, **m** ranges from 0 to 7
interrupt period is 64μs*(**m**+1)*2^{n}

SCIDRL 8 bit data serial data register
SCIBD is 16-bit SCI baud rate register, let **n** be the 16-bit number Baud rate is 12MHz/**n**
SCICR1 is 8-bit SCI control register
bit 4 M, Mode, 0 = One start, eight data, one stop bit, 1 = One start, eight data, ninth data, one stop bit
SCICR2 is 8-bit SCI control register
bit 7 TIE, Transmit Interrupt Enable, 0 = TDRE interrupts disabled, 1 = interrupt whenever TDRE set
bit 5 RIE, Receiver Interrupt Enable, 0 = RDRF interrupts disabled, 1 = interrupt whenever RDRF set
bit 3 TE, Transmitter Enable, 0 = Transmitter disabled, 1 = SCI transmit logic is enabled
bit 2 RE, Receiver Enable, 0 = Receiver disabled, 1 = Enables the SCI receive circuitry.
SCISR1 is 8-bit SCI status register
bit 7 TDRE, Transmit Data Register Empty Flag
Set if transmit data can be written to SCDR
Cleared by **SCISR1** read with TDRE set followed by **SCIDRL** write.
bit 5 RDRF, Receive Data Register Full
set if a received character is ready to be read from **SCIDRL**
Clear the RDRF flag by reading **SCISR1** with RDRF set and then reading **SCIDRL** .

ATDDIEN ADC digital enable register, 1 to make corresponding pin digital, 0 to make corresponding pin analog
PTAD is 8-bit bi-directional I/O port
DDRAD is the associated direction register for digital pins of Port AD (0 means input, 1 means output)

```
0xFFD6    interrupt 20 SCI
0xFFDE    interrupt 16 timer overflow
0xFFE0    interrupt 15 timer channel 7
0xFFE2    interrupt 14 timer channel 6
0xFFE4    interrupt 13 timer channel 5
0xFFE6    interrupt 12 timer channel 4
0xFFE8    interrupt 11 timer channel 3
0xFFEA    interrupt 10 timer channel 2
0xFFEC    interrupt 9  timer channel 1
0xFFEE    interrupt 8  timer channel 0
0xFFF0    interrupt 7  real time interrupt
```



Place your answers on pages 5 and 6.

For questions 1-7, the definition is given and you are asked to give the correct term described by that definition. Since there are more terms than definitions, not all terms will be used. *Answer each as A through Z.*

A) latency	H) critical section	O) friendly	V) tristate
B) high-speed CMOS	I) buffered I/O	P) stabilization	W) nonintrusive
C) Schottky	J) polled interrupt	Q) I/O bound	X) invasive
D) atomic	K) vectored interrupt	R) CPU bound	Y) intrusive
E) reentrant	L) fixed-point	S) instrument	Z) open collector
F) volatile	M) desk check	T) blind cycle	
G) busy waiting	N) nonvolatile	U) real-time	

(5) Question 1. A condition when the bandwidth of the system, which is the amount of data processed per second, is limited by the speed of the input device.

(5) Question 2. An interrupt scheme where each individual flag that can request an interrupt has a unique interrupt vector.

(5) Question 3. A condition when the debugger itself significantly alters the operation of software/hardware system.

(5) Question 4. A debugging technique that fixes all its inputs to specific values and can be repeated over and over.

(5) Question 5. A type of logic that can be high, low or off.

(5) Question 6. The time between new input being ready and the time when the new input is read.

(5) Question 7. An I/O interfacing technique that uses a FIFO queue to pass data between the foreground and the background.

---

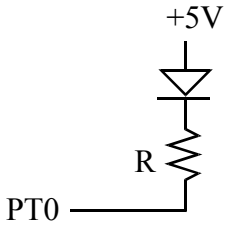
(5) Question 8. Consider the situation where interrupt-driven I/O uses a FIFO to transmit data between the foreground and the background. The producer generates data and puts it into the FIFO, concurrently with the consumer getting data from the FIFO and processing it. Observations of the system reveal the FIFO becomes full. Which best describes the situation?

A) The problem can always be solved by increasing the size of the FIFO.
B) The problem can never be solved by increasing the size of the FIFO.
C) The problem can always be solved by increasing the producer rate.
D) Sometimes the problem can be solved by increasing the size of the FIFO, but other times it will be necessary to increase the consumer rate or decrease the producer rate.
E) The problem can always be solved by increasing the interrupt rate.

(5) Question 9. The resolution of an 8-bit unsigned binary fixed-point number is 2^{-2}, which equals 1/4. What is the value of the number if the integer stored in memory is 41?

(5) Question 10. Assume you are going to use a 16-bit signed decimal fixed-point number system to represent the values from -100 to +100. What is the smallest fixed-point resolution you could use?

(5) Question 11. You are given the 6812 voltage and current parameters for PT0: V_{OH}, V_{OL}, V_{IH}, V_{IL}, I_{OH}, I_{OL}, I_{IH}, and I_{IL}. The desired LED voltage is V_D and the desired current is I_D. Assume the LED current is small enough, so the LED can be interfaced directly to the 6812 PT0 pin as shown. Give the equation to calculate the resistance R in terms of V_D, I_D, V_{OH}, V_{OL}, V_{IH}, V_{IL}, I_{OH}, I_{OL}, I_{IH}, and I_{IL}.



(5) Question 12. List the events in proper order as a **`RTI`** interrupt causes the computer to switch from foreground to background? Do not include events explicitly caused by executing software in either the foreground or in the background, just the hardware events occurring during the context switch.

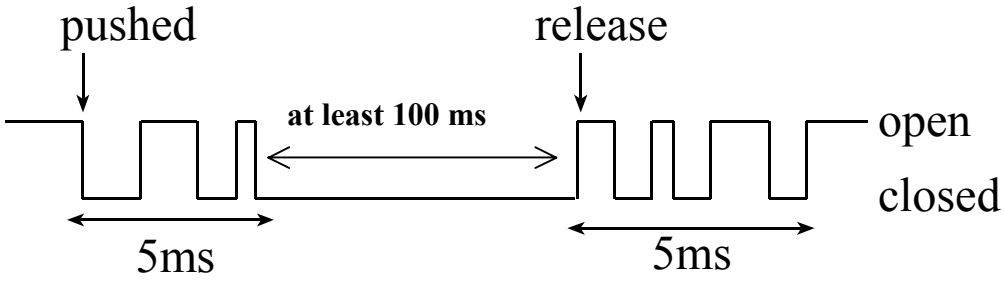
- **A)** The PC is loaded with the 16-bit contents of **`$FFF0`**
- **B)** The **`RTI`** interrupts are armed by setting the **`RTIE`** bit in the **`CRGINT`** register
- **C)** The PC, Y, X, B, A, CC registers are pulled from the stack
- **D)** The CC, A, B, X, Y, PC registers are pushed on the stack
- **E)** The **`I`** bit is set to one (disable)
- **F)** The PC is set equal to **`$FFF0`**
- **G)** A write with bit 7=0 occurs to **`CRGFLG`**, clearing **`RTIF`**
- **H)** A write with bit 7=1 occurs to **`CRGFLG`**, clearing **`RTIF`**
- **I)** The periodic timer clock times out, setting **`RTIF`**
- **J)** The **`I`** bit is cleared to zero (enable)

(5) Question 13. A signed fixed point system has a range of values from -50 to +50 with a resolution of 2^{-8}. Note: 2^{-8} equals 1/256.With which of the following data types should the software variables be allocated? When more than one answer is possible choose the most space efficient type.

A) `char`	**D)** `unsigned char`	**G)** `float`
B) `short`	**E)** `unsigned short`	**H)** `double`
C) `long`	**F)** `unsigned long`	

(35) Question 14. You will be using the switch attached to PAD7 and the LED attached to PT0. The LED should be initially off, and the LED should be turned on (and left on) when the operator pushes the switch five (5) times. The goal of this question is to write RTI interrupt-based software that counts the number of times the switch is pushed, and turns the LED on if the switch is pushed five (5) times.

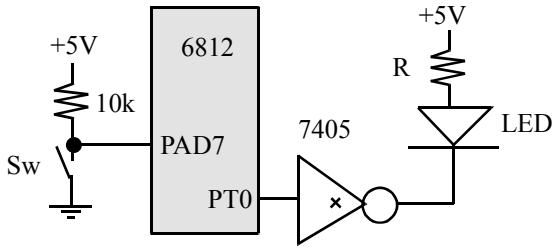
When the switch is pushed, the input on PAD7 is low. When the switch is released the PAD7 input will be high. A count requires the switch to first be pushed then to be released. The switch bounces for about 5ms each time it is pushed and each time it is released.



When your software writes a '1' to the PT0 output, the LED will come on. The main program is fixed as

```
void main(void){
  RTIinit();
  while(1){};
}
```

and can not be changed.



Your **`RTIinit`** will initialize any required global variables, turn off the LED, arm RTI, and enable interrupts. Select the RTI interrupt period to be 16384μs by setting **m**=1 and **n**=7 in the following equation $64\mu s*(\mathbf{m}+1)*2^{\mathbf{n}}$.

Your **`RTIhandler`** will test the value of the switch, counting the number of times it has been pushed and released. When the count goes to 5, the LED is turned on.

In this system, there is no software other than my main program, your ritual, and your RTI interrupt service routine. Nevertheless, your software must be friendly.

Jonathan W. Valvano First:_____ Last:_____
March 1, 2006, 1:00pm-1:50pm. This is a closed book exam. You have 50 minutes, so please allocate your time accordingly. ***Please read the entire quiz before starting***. Only this piece of paper (pages 5 and 6) will be turned in.

(5) Question 1.	
(5) Question 2.	
(5) Question 3.	
(5) Question 4.	
(5) Question 5.	
(5) Question 6.	
(5) Question 7.	
(5) Question 8.	
(5) Question 9.	
(5) Question 10.	
(5) Question 11.	
(5) Question 12.	
(5) Question 13.	

(35) Question 14. Show your **`RTIinit()`** and any required global variables

Show your **`RTIhandler()`**