EE443 - Embedded Systems

Exercise - 5

Interrupts

1. List the series of operations that takes place in responding an interrupt call.

2. List three advantages of using interrupts compared to polling techniques in responding to external events and in timing arrangements.

3. Describe the two common methods for vectoring interrupts (directing interrupt calls to the target ISR addresses).

4. Sort the following interrupt sources according to the interrupt priority requirements (from the highest priority to the lowest):

- UART interrupt: Indicates a byte is received that should be read from the buffer before another byte arrives. Maximum data rate is 10KByte/s.
- Timer interrupt: Requires an ADC sample to be taken within 10µs after the interrupt.
- Keyboard interrupt: Indicates a key is pressed by the user.
- Position encoder interrupt: ISR counts the number of pulses received at the interrupt input. Maximum pulse rate is 100 per second.

5. A microcontroller performs an operation in predetermined time intervals controlled by a timer interrupt. List all specifications and other factors related to the **a) clock generator**, **b) timer**, **c) microprocessor hardware**, and **d) software** that can affect the timing accuracy.

6. You have an 8-bit microprocessor that can read/write memory one byte at a time. ISR1 increments a 16-bit variable named "**`P1count`**" everytime an Int1 interrupt request is received. The main program reads **`P1count`** and uses its value in a calculation.

a) What can go wrong in the code given below?

b) How can you fix it?

```
MainProg:
---
---
---
Load    A, P1countL;
Load    B, P1countH;
---
---
Load    A, Sum;
Add     A, ADCout;
---
---
---
```

```
ISR1:
---
Load    A, P1countL;
Add     A, #0x01
Store   A, P1countL;
JumpIC  CalcH;
Return;
CalcH:
Load    A, P1countH;
Add     A, #0x01
Store   A, P1countH;
---
Return;
```

7. Calculate the memory size required for stack operations for the following two cases. Microprocessor responds to only one interrupt besides the regular subprogram calls in both cases. Assume that a normal subprogram call requires 15 bytes and an ISR call requires 10 bytes of stack storage.

a)

```
MainProg       Sub1           Sub2           Sub3
----           ----           ----           ----
Loop:          ----           ----           ----
----           ----           Call Sub3      ----
Call Sub1      ----           ----           ----
----           ----           ----           ----
Call Sub2      return         return         return
----
GOTO Loop
```

b)

```
MainProg       Sub1           Sub2
----           ----           ----
Loop:          ----           ----
----           Call Sub2      ----
Call Sub1      ----           ----
Call Sub1      ----           ----
----           return         return
Call Sub2
----
GOTO Loop
```

c) Repeat the stack calculations for the second case given above, but this time assume that there are two interrupts. Calculate the memory size required for stack operations for the following two cases:

Case-1: Interrupts are not allowed in ISR's.

Case-2: Interrupts are allowed in ISR's.

8. A semaphore is used to transfer data from ISR1 to ISR2 as follows.

If the previously stored data were used by ISR2, then ISR1 reads 16-bit data from an ADC and stores them in the memory. Otherwise, ISR1 returns without reading ADC.

ISR2 reads the data stored by ISR1 and calculates an output value if a new value is received from the ADC since the last calculation. Otherwise, ISR2 returns without doing any calculation.

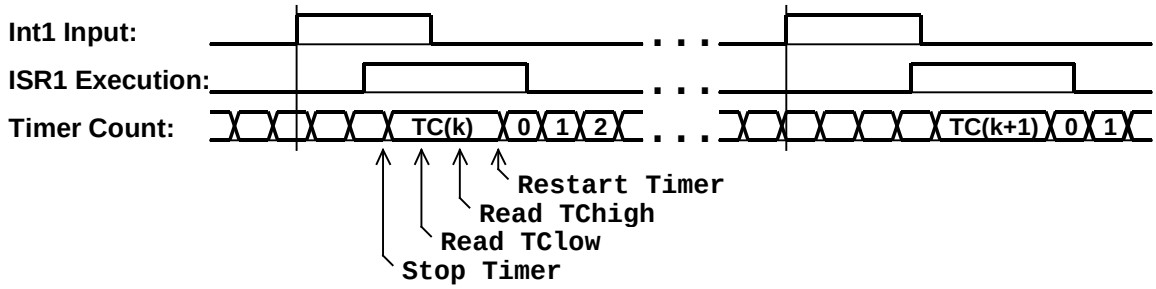
You should allow other interrupts to be received during execution of ISR1 and ISR2. You can block interrupts during the critical operations using the following instructions:
DisInt : Disable Interrupts (blocks all interrupts until an EnInt is executed)
EnInt : Enable Interrupts

a) Write the step-by-step operations in two ISRs that require transfer of data from ISR1 to ISR2 using a semaphore.

b) What can you do if the calculation of output value in ISR2 takes significant time and if you don't want to block ISR1 interrupt during that time?

9. A 16-bit timer is used for measuring the time between the rising edges of short pulses. The pulse signal is used as **Int1** interrupt input to the microcontroller that initiates a call to **ISR1**. Consider the following interrupt service routine and the timing diagram to answer the questions below:

```
ISR1:
Push    A;              save register-A
Stop    Timer;          stop the timer
Read    A, TimerLow;    read timer low byte
Store   A, TClow;       store low byte in memory
Read    A, TimerHigh;   read timer high byte
Store   A, TChigh;      store high byte in memory
Start   Timer;          restart the timer
Pop     A;              restore register-A
Clear   Int1Flag;       clear Int1 interrupt flag
Return;
```



a) Which sensitivity type is preferable for the **Int1** interrupt input? Why?

b) Why it may take longer to start **ISR1** after the timer interrupt is received?

c) Calculate the timing accuracy in ns with the following conditions (assume that **Int1**-to-**ISR1** delay is constant). Indicate source and amount of every timing error.

- The longest pulse interval to be measured is 20ms.
- Timer clock frequency is 2MHz.
- Controller clock frequency has +/-50ppm absolute accuracy at 25ºC.
- Temperature dependence of the controller clock frequency is 1ppm/ºC, and the system operating temperature range is +5ºC minimum to +85ºC maximum.
- Jitter at **Int1** rising edge is +/-200ns with respect to the previous rising edge.

d) Calculate the additional timing error as a result of the varying **Int1**-to-**ISR1** delay with the following conditions:

- Clock frequency of the microprocessor in the controller is 40MHz.
- Execution of machine instructions takes between minimum 4 and maximum16 clock cycles.

10. Consider the following interrupt service routine that works together with the **ISR1** in the question given above. **ISR2** loads the timer data stored in memory by **ISR1**, and calculates an output result

```
ISR2:
Push    A;              save register-A
Push    B;              save register-B
Load    A, TClow;       load low byte of timer count
Load    B, TChigh;      load high byte of timer count
----
----    calculate output using data in registers A and B
----
Pop     A;              restore register-A
Pop     B;              restore register-B
Clear   Int2Flag;       clear Int2 interrupt flag
Return;
```

a) What can go wrong if **ISR1** has higher priority?

b) What can go wrong if **ISR2** has higher priority?

c) Assume **ISR1** has higher priority, and write a modified **ISR2** to prevent calculations on corrupt timer count data by blocking interrupts for a minimal time.

d) How does blocking interrupts in **ISR2** affect the timing accuracy obtained in **ISR1**?

e) Write modified versions of **ISR1** and **ISR2** using the semaphore, **TCready**, to prevent calculations on corrupt timer count data. You can use the following instructions to modify and test the semaphore:

```
Set     TCready;        sets the TCready flag
Clear   TCready;        clears the TCready flag
JmpIfTCready  <target>; jumps to <target> if TCready is set
```