CMPE-250
Assembly and Embedded Programming
Spring 2021
Laboratory Exercise Ten:
Timer Driver Input Timing

This exercise develops a timer driver for the KL05Z board. It uses the KL05 periodic interrupt timer (PIT) with channel zero. The objective of this exercise is to implement interrupt-based timing measurements accurate to within 0.01 s, (i.e., 10 ms). An interrupt service routine (ISR) and a driver program are written, and they are run on the KL05Z board.

Prelab Work

1. Write the subroutine Init_PIT_IRQ to initialize the KL05 as discussed in class and presented in the class notes for an interrupt every 0.01 s from PIT channel 0. You must write the subroutine so that no registers have changed value after return.
2. Write PIT_ISR, the ISR for the PIT module. On a PIT interrupt, if the (byte) variable RunStopWatch is not zero, PIT_ISR increments the (word) variable Count; otherwise it leaves Count unchanged. In either case, make sure the ISR clears the interrupt condition before exiting. You must write the ISR so that no registers have changed value after return.
3. Write the change required in the KL05 vector table to "install" PIT_ISR.

Application

Certain tasks requres timing information. This assignment implements hardware timing accurate to 0.01 s to time user input for each of three different pieces of information It asks the user for three string inputs, and for each input it shows the time the user took to complete the string response.

Program Specification

Write a program that works with the subroutine and ISR from prelab work to time user input to within approximately 0.01s. The program must perform the following sequence of operations and must meet the requirements that follow.

1. Use Init_UART0_IRQ from Lab Exercise Nine to initialize the KL05 for interrupt-based serial I/O with UART0 at 9600 baud using a format of one start bit, eight data bits, no parity, and one stop bit—as in Lab Exercise Nine. (Also include UART0_ISR from Lab Exercise Nine, and "install" it in the KL05 vector table.)
2. Initialize the (byte) Boolean stopwatch run variable RunStopWatch to zero (0).
3. Initialize the (word) stopwatch clock count variable Count to zero (0).
4. Use Init_PIT_IRQ from prelab work to initialize the KL05 PIT to generate an interrupt from channel zero every 0.01s.
5. Output the character string below to the terminal screen followed by a carriage return character (0x0D) and a line feed character (0x0A) to advance the cursor to the beginning of the next line, and then output ">" as a prompt, as shown below.

```
Enter your name.
>
```

6. Clear the stop watch clock count variable Count to zero (0), and set the Boolean stopwatch run variable RunStopWatch to one (1).
7. Input a character string from the terminal keyboard, as shown below for the example where "John Doe" was typed by the user.

```
Enter your name.
>John Doe
```

8. Clear the Boolean stopwatch run variable RunStopWatch to zero (0).
9. Output "<" to the terminal as a marker, then print the time it took the user to enter the text in step 7, (i.e., the value of the stop watch clock count variable Count), print the time unit factor " x 0.01 s", and finally move the cursor to the beginning of the next line, as shown below where it took the user 5 s to input the text "John Doe."

```
Enter your name.
>John Doe
<500 x 0.01 s
```

10. Repeat steps 5–9 two times with the alternate prompt text shown below instead of the text shown in step 5.
    - Enter the date.
    - Enter the last name of a 250 lab TA.

11. Output "Thank you. Goodbye!" to the terminal, and then move the cursor to the beginning of the next line, as shown below. (Note: with terminal I/O via the serial driver, you should wait for the transmit queue to empty before ending the program, as the output to the terminal screen is asynchronous with PutStringSB.)

```
Enter your name.
>John Doe
<500 x 0.01 s
Enter the date.
>3/29/2021
<750 x 0.01 s
Enter the last name of a 250 lab TA.
>Bacchetta
<1242 x 0.01 s
Thank you.  Goodbye!
```

In addition, the program must meet these requirements.

- The program is to use two variables for timing and control.
  - Count: word variable for stopwatch clock count variable.
  - RunStopWatch: byte variable for Boolean stopwatch timing state—zero when not timing and nonzero when timing.
- It must use the subroutine and ISR from prelab work for timing accurate to within approximately 0.01 s. The following is a summary of the parameter specifications for the subroutine. (Note: unlike the subroutine, the ISR does not have parameters.)

  Init_PIT_IRQ

  Input parameter:

  (none)

  Output parameter:

  (none)
- It must use the serial driver, (i.e., subroutines, ISR, and receive and transmit queues), from Lab Exercise Nine for character I/O.
- It must use subroutines from Lab Exercise Six for string and number I/O.
- It must have properly commented subroutines, including a comment header block that includes these components.
  - Description of functionality
  - List of subroutines called
  - Lists of registers
    - Input parameters
    - Output parameters
    - Modified registers (after return, not preserved by subroutine)

Lab Procedure

1. Create a new directory and Keil MDK-ARM project for this exercise.
2. Write a properly commented and properly formatted KL05 assembly language program to implement a timer driver for input timing, which functions according to the preceding specification.
3. Build the program with Keil MDK-ARM to create a listing file and a linker map file.
4. Test your program. Make sure to test timing of all three inputs and program shutdown.
5. Examine the KL05 listing and memory map produced for your program, and determine the exact memory range, (i.e., the starting address and the address of the last byte), for these components, (to include and discuss in the results section of your report):
   a. Executable code, (i.e., total `MyCode AREA`),
   b. PIT ISR code,
   c. Constants in ROM (e.g., prompt strings), and
   d. RAM used.
6. Demonstrate steps 4 and 5 for your lab instructor. Following demonstration, capture the terminal screen output to include and discuss in the results section of your report.
7. Submit your source file, listing file, map file, report, and grading sheet to the myCourses assignments for this exercise.

Baseline Metrics

The metrics in the table below can be used to compare your solution to a baseline solution.

Language	Routine	Instructions/LOC	Code Size (bytes)
Assembly	main program*	39	100
C	main	19	110
Assembly	Init_PIT_IRQ	32	64
C	Init_PIT_IRQ	11	92
Assembly	PIT_ISR	14	28
C	PIT_IRQHandler	6	30

*Not including instructions from provided program template.

Report

Write a report consisting of the following sections. Your writing should follow the rules of professional technical writing and should meet the specifications in "Laboratory and Report Guidelines" on myCourses.

- Abstract
- Procedure
- Results
  - ☞ The memory ranges specified in step 5 should be part of your report's results section. (Remember to introduce and explain them in text rather than merely pasting them into the report.)
  - ☞ The screen capture from step 6 should be part of your report's results section. (Remember to introduce and explain it in text rather than merely pasting it into the report.)
- Conclusion

Submission

myCourses Assignments
Separate assignment for each item below • Report (including cover sheet) • Source code (.s) • Listing (.lst) • Map (.map)

Late submissions are penalized 20% per day late, where "day late" is calculated by rounding submitted time to the nearest greater or equal integer number of days late (i.e., ceiling function). Note: To receive any grading credit for this lab exercise you must earn points for *both* the demonstration *and* the report.