CSE 379 Lab 6: Timers

Daniel Jin (djin3)
Motaz Khalifa (motazkha)

---

1) Purpose of the Program

a) Objective

The objective of this program is to utilize timers and interrupts to build our own Snake Game. Users will be able to play snake on a 40x15 game board, using keys i, j, k, and m to move up, right, left, and down respectively. If the snake runs into either a wall or itself, the game is over, and the program will terminate.

b) Debugging Steps

Our implementation required the direction flag, location, and score to be stored in memory. Therefore, we used and monitored the memory browser extensively. Since the location and score could possibly reach values where 3 hexadecimal digits would be needed to represent them, we experimented with storing and loading bytes (STRB/LDRB), half words (STRH/LDRH), and entire registers (STR/LDR). We monitored memory addresses that corresponded to the location of the next asterisk, the current score, the address/offset of the current location, and the current direction.

c) How To Run The Program

Open PuTTy using the same process as previous labs. In Code Composure Studios, click Debug → Resume. You will see — in PuTTy —an introduction to Snake. To play the game, make sure the PuTTy window is selected; otherwise, i, j, k, and m will not do anything. The rules for this game are similar to all other Snake games: navigate through the game board without hitting either a wall or your tail. Doing so will result in game over and the program terminating.

d) Outside References

The Tiva™ TM4C123GH6PM Microcontroller Data Sheet was referenced for information about the necessary registers for this lab.

2) Division of Work

Daniel wrote *timer_init*, *initial_print*, and *timer_interrupt_clear*.. Motaz wrote the core implementation of *Timer0_Handler* and *UART0_Handler. print_update* and other subroutines are either taken straight from the previous lab or modified from previous labs.

3) Logic

a) Summary of Main Program Flowchart

lab_6:

This subroutine first initializes UART0, the timer, the interrupts, and prints the initial board and initial score. The subroutine then goes into an infinite loop, constantly checking the "quit" flag. If the "quit" flag is a 1, the program terminates. If not, the subroutine continues in its infinite loop, waiting for a timer interrupt or UART interrupt.

b) Summary of Subroutine Flowcharts

i) UART0_Handler

This subroutine interrupts the program when a key on the keyboard is pressed; in this lab, the keys pressed are either i, j, k, or m. This subroutine checks to see which of the four is pressed, and updates the direction flag to a number corresponding to each letter; 1 corresponds to k, 2 corresponds to j, 3 corresponds to i, and 4 corresponds to m. Before returning to the main program, the interrupt is cleared.

ii) Timer0_Handler

This subroutine first clears the timer interrupt, as it should be cleared at the beginning of the handler. Then the handler looks at the direction flag, looks at the location where the new asterisk would be in that direction, and looks at the character at that location. If the character is not a space (i.e. the snake has hit a wall or itself) then a "quit" flag is changed to 1. If the character is a space, the space is replaced with an asterisk. Finally, this subroutine prints to PuTTy the updated status of the board and the score.

iii) timer_init

This subroutine initializes the necessary registers for the timer to "activate." This subroutine connects the clock to timer, sets up the timer for 32-bit mode, periodic mode, sets the timer to interrupt every quarter of a second, enables the time, and sets the NVIC interrupt.

iv) timer_interrupt_clear

This subroutine clears the timer interrupt by writing to the GPTM Interrupt Clear Register (GPTMICR).

v) initial_print

This program simply prints to PuTTy the initial state of the board and the score. The asterisk/snake starts in the center of the board, and the score is 0.

vi) print_update

This program increments the score, and prints out the score and the board as is.

c) Registers and Addresses

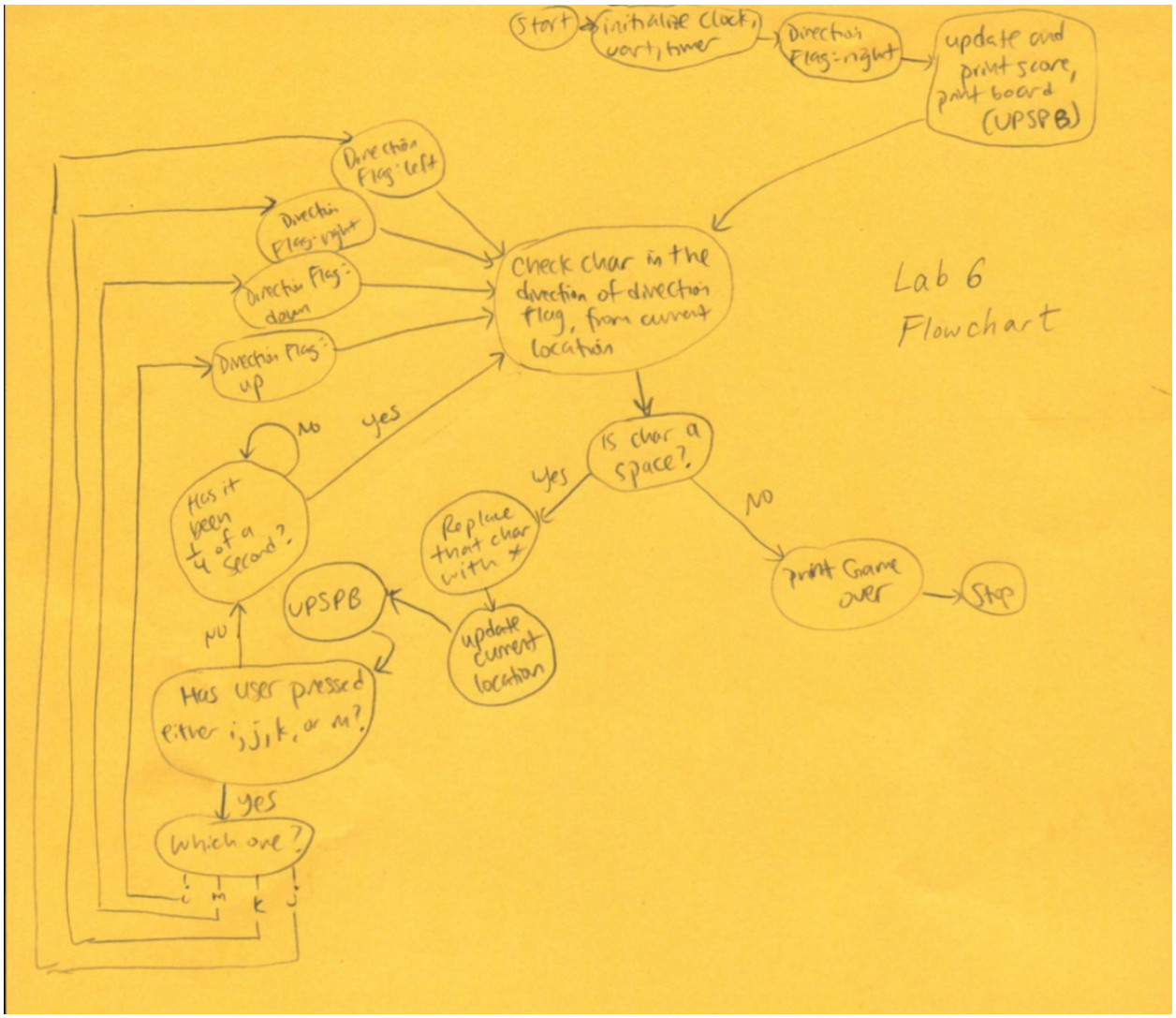
The necessary registers needed for the initialization of UART interrupts are the UART NVIC (0xE000E100) and the UART Interrupt Mask Register (UARTIM; offset 0x038). The RTIM bit in UARTIM (bit 6) and bit 5 in the NVIC enables the interrupt. UART Interrupt Clear (UARTICR; offset 0x044) is the interrupt clear register. The RXIC (bit 4) is needed to clear the corresponding interrupt. The base address for UART0 is 0x4000C000.

The necessary registers needed for the initialization of the timer interrupts are the timer NVIC (0xE000E100) and the General-Purpose Timer Run Mode Clock Gating Control (RCGCTIMER; 0x400FE604). Bit 19 in the NVIC sets the timer, and the R0 bit (bit 0) in RCGCTIMER connects the clock to the timer. The GPTM Configuration Register (GPTMCFG; offset 0x000) sets up the timer for 32-bit mode by writing 0x0 to GPTMCFG (bits 2:0). The GPTM Timer A Mode Register (GPTMTAMR; offset 0x004) sets the timer to periodic mode by writing 0x2 to the TAMR bits (bits 1:0). The GPTM Timer A Interval Load Register (GPTMTAILR; offset 0x028) sets up an interval by writing the desired interval into the register; all 32 bits of this register are to hold the interval. The GPTM Interrupt Mask (GPTMIMR; offset 0x018) sets the timer to interrupt whenever the interval set in GPTMTAILR is reached if the TATOIM bit (bit 0) is 1. The GPTM Control Register's (GPTMCTL; offset 0x00C) TAEN bit (bit 0) either enables or disables timer A. When configuring timer A, timer A must be disabled. The GPTM Interrupt Clear Register (GPTMICR; offset 0x024) clears the interrupt in timer A by writing a 1 to the TATOCINT bit (bit 0).

4) Flowcharts

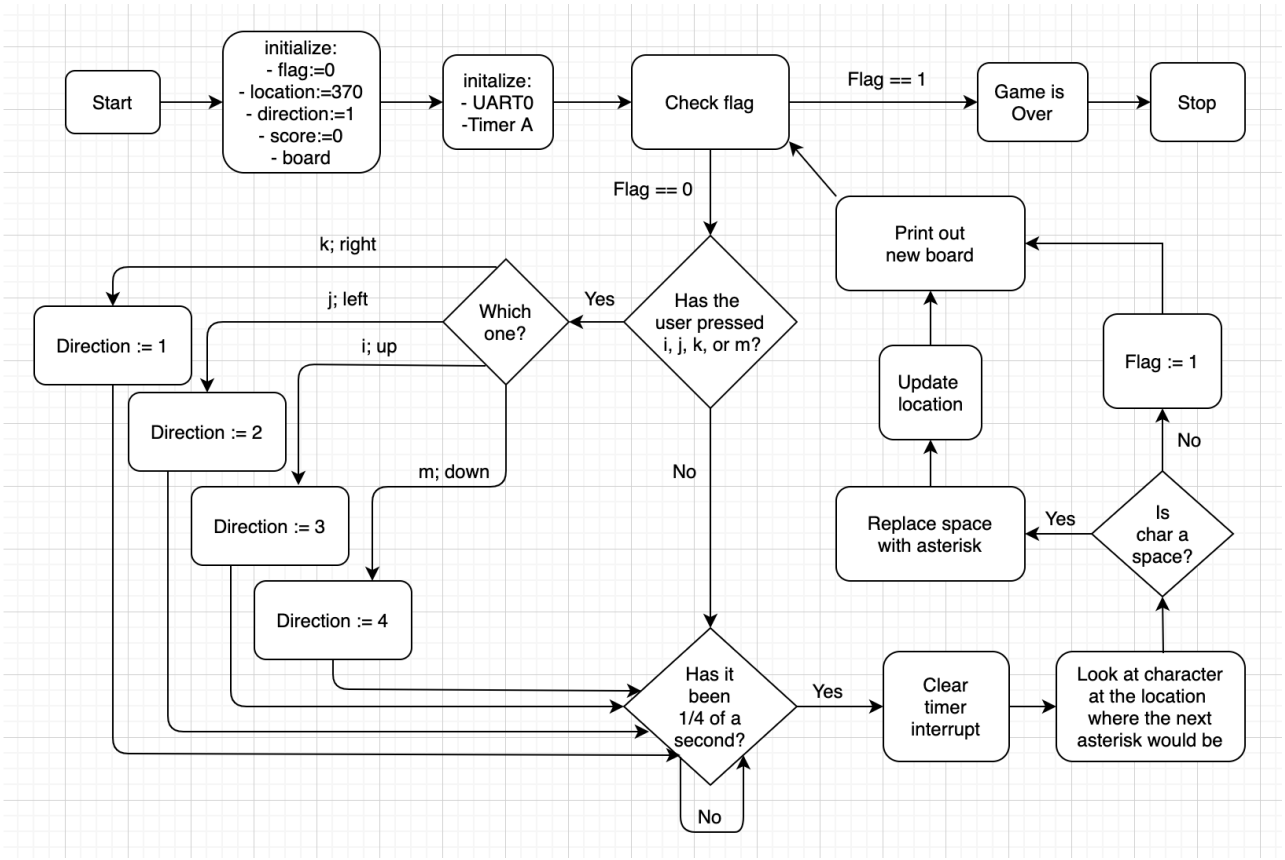
All flowcharts are generated by draw.io.

a) Rough Draft



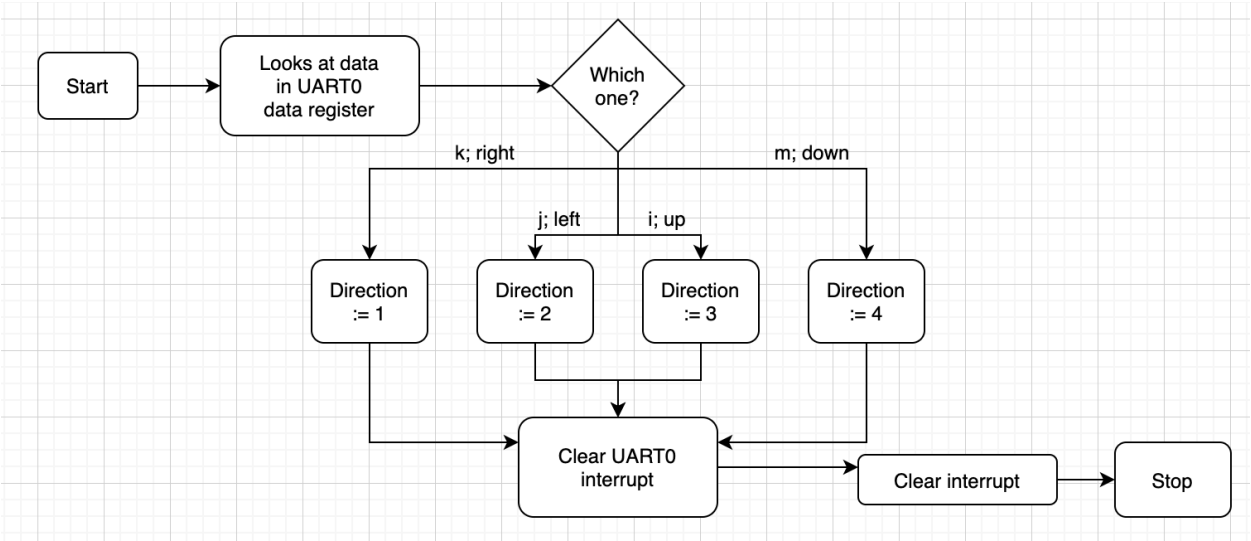
b) Main Program Flowchart

lab_6:

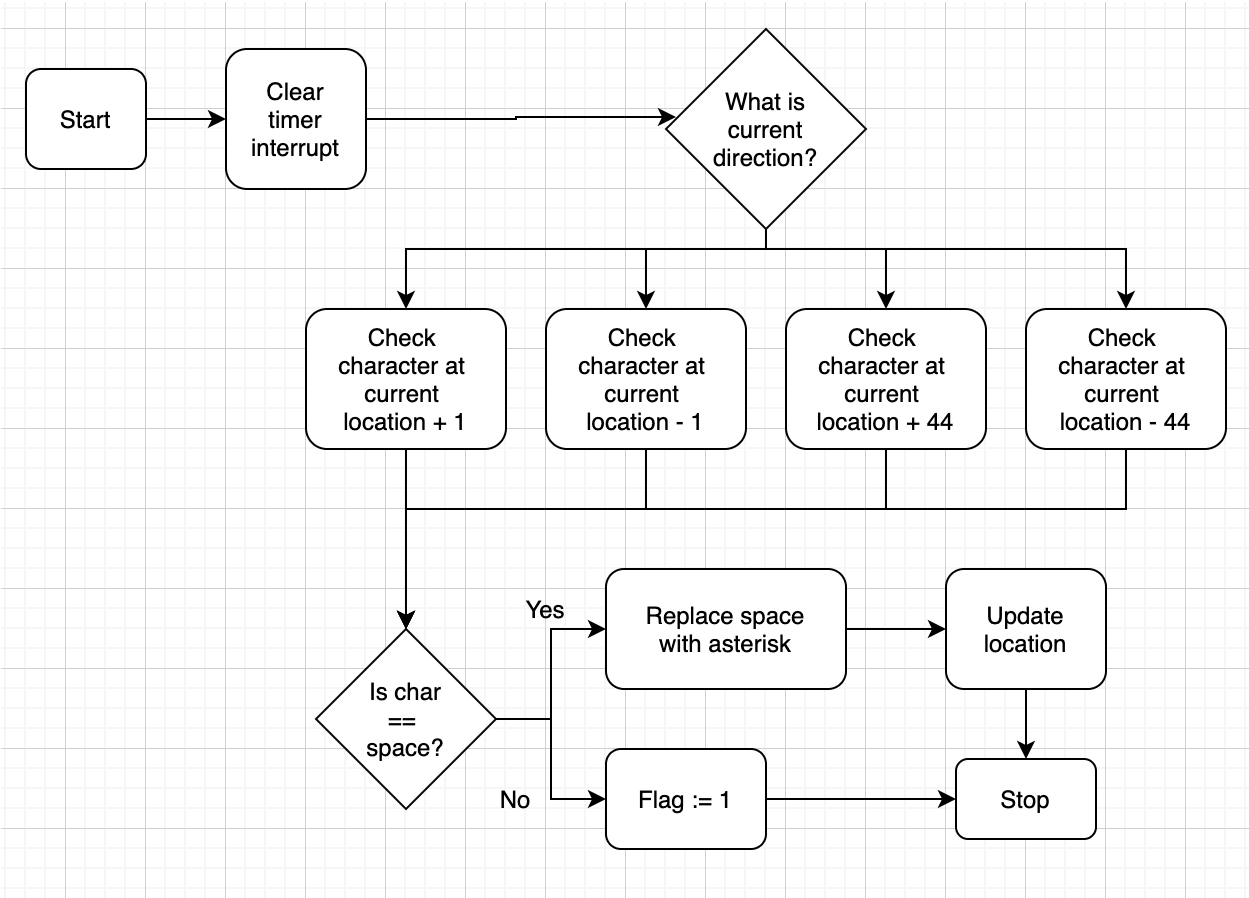


c) Subroutine Flowcharts

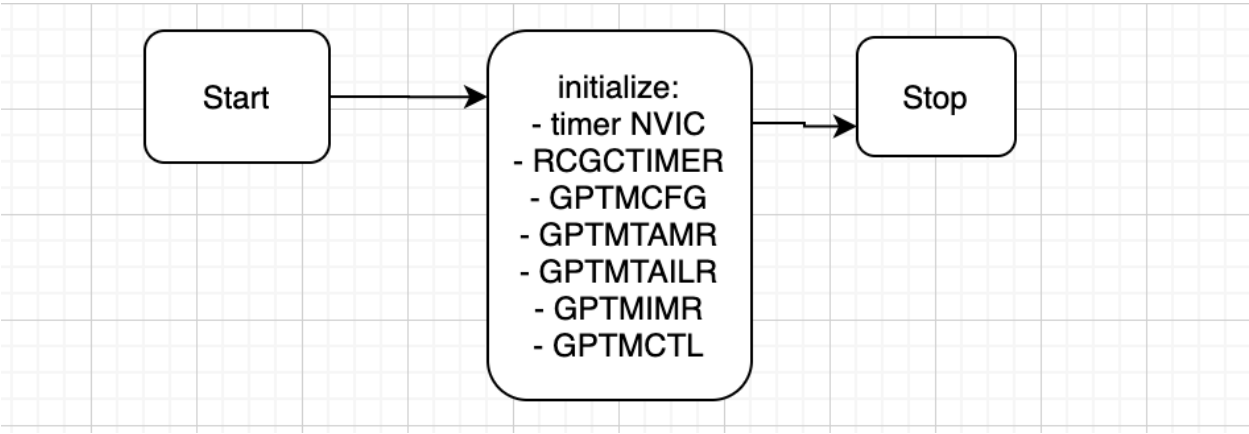
i) UART0_Handler



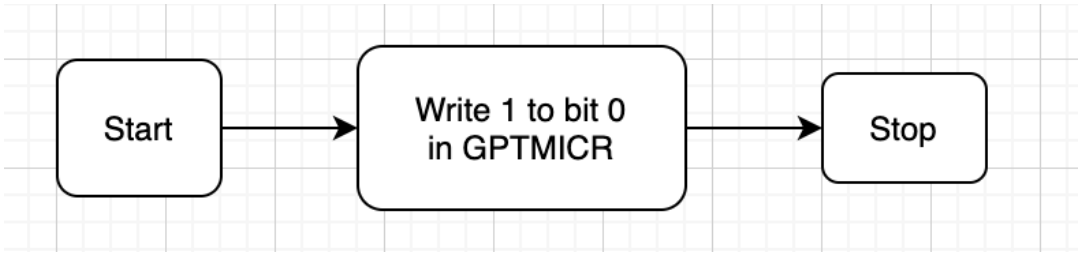
ii) Timer0_Handler



iii) timer_init



iv) timer_interrupt_clear



v) initial_print, *print_update*