LAB #6: “SNAKE” GAME

CSE379: MICROPROCESSORS

Dee Gao & Carmen Tam

UBIT: CTAM2 (50194532) AND DGAO (50244179)

Description

Division of Work

We worked together on the documentation and the timer initialization. Dee worked on the game logic and UI, while Carmen focused on debugging the timer issues and revising code to ensure everything is working properly.

Purpose of the Program

The main objective of Lab 6 is to familiarize ourselves with timer interrupts. To do this, the task is to code a simplified version of the game *Snake* by having the game update itself through the timer interrupt automatically every 0.25 seconds.

Debugging Steps

We had issues with initializing the timer interrupt because it was not firing automatically. We had to keep checking that we followed all the initialization steps correctly very carefully. When we had an issue with the timer interval firing faster than we wanted, we realized that we did not clear the timer interrupt properly. After fixing that, our program was working properly.
There was also an issue with our game board refreshing too quickly in the sense that we could see the cursor moving around the screen while the game was active. We tried to change the baud rate and also tried various methods of "refreshing" the screen, but we essentially only got the cursor to move around less than what we originally had, which is acceptable enough for us.

Outside Material

We referred to the Timer lecture slides and the Tiva™ TM4C123GH6PM Microcontroller Data Sheet.

Instructions

Flash the program and you should see the starting game screen show up on the terminal. Adjust the terminal screen to the size of the game board to ensure that the game board refreshes properly. Press w, a, s, or d to start the game and attempt to keep moving around without touching the borders or the snake itself. Upon losing, you will see the ‘Game Over’ screen and your score. If you want to play again, you can press the ‘p’ key to restart the game.

Logic

The game’s functionality is focused around a timer interrupt that fires every 0.25 seconds. The UART interrupt is solely responsible for reading the key that the user presses and storing it within a register or restarting the game if the key pressed was a ‘p.’

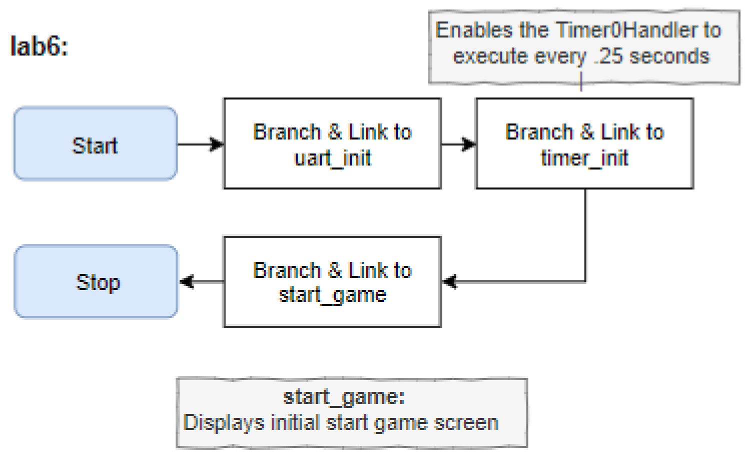
Simultaneously, the timer interrupt goes into the game_play subroutine, which checks which direction was pressed by the user and adding an asterisk to the snake in the direction pressed. It does this by storing the asterisk in memory once the key is checked, resetting the cursor to the top left corner of the screen, and redisplaying or overwriting the game board to reflect the new snake. The snake will continue to grow in the direction specified by the user until the next location is not ‘empty’ or a space. If the space is not empty, the program displays the ‘Game Over’ screen and the score.

Subroutines

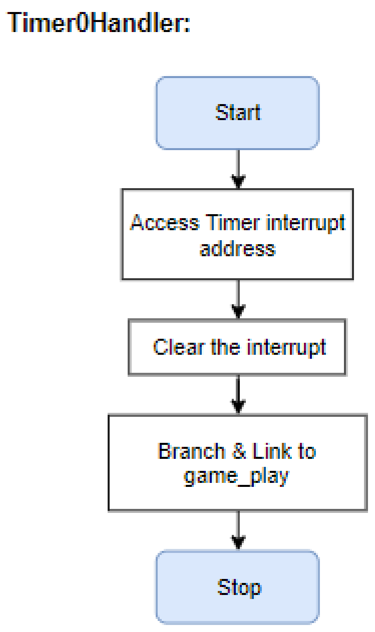
1. *UART0Handler*: For the UART interrupt, it simply reads the character pressed on the keyboard and stores it in a register. Afterwards, it checks if the user pressed ‘p,’ in which case, the program would restart the game.
2. *Timer0Handler*: The timer interrupt triggers every 0.25 seconds and every time it does, it checks the current direction that the snake is going in and adds an asterisk in that direction next to the previous asterisk. In other words, the snake keeps growing in the direction that the user presses. It does this by storing the asterisk in the proper place in memory and reprinting the game board repeatedly.

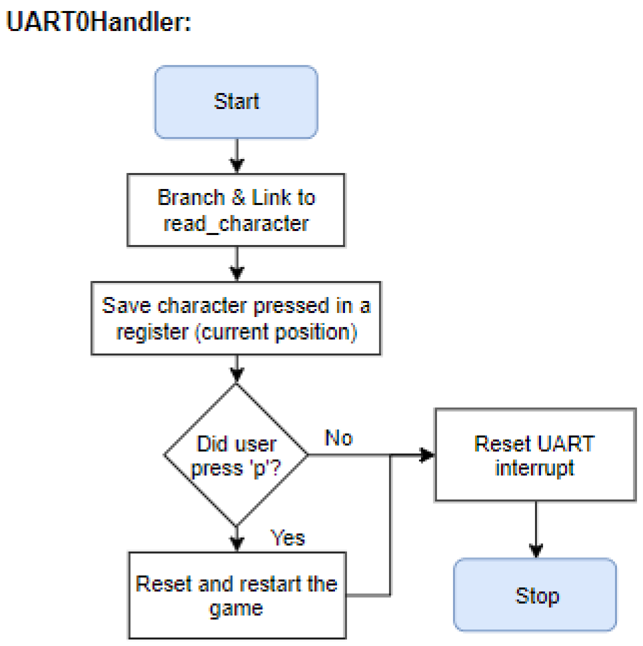
Flowcharts

Main flowchart:

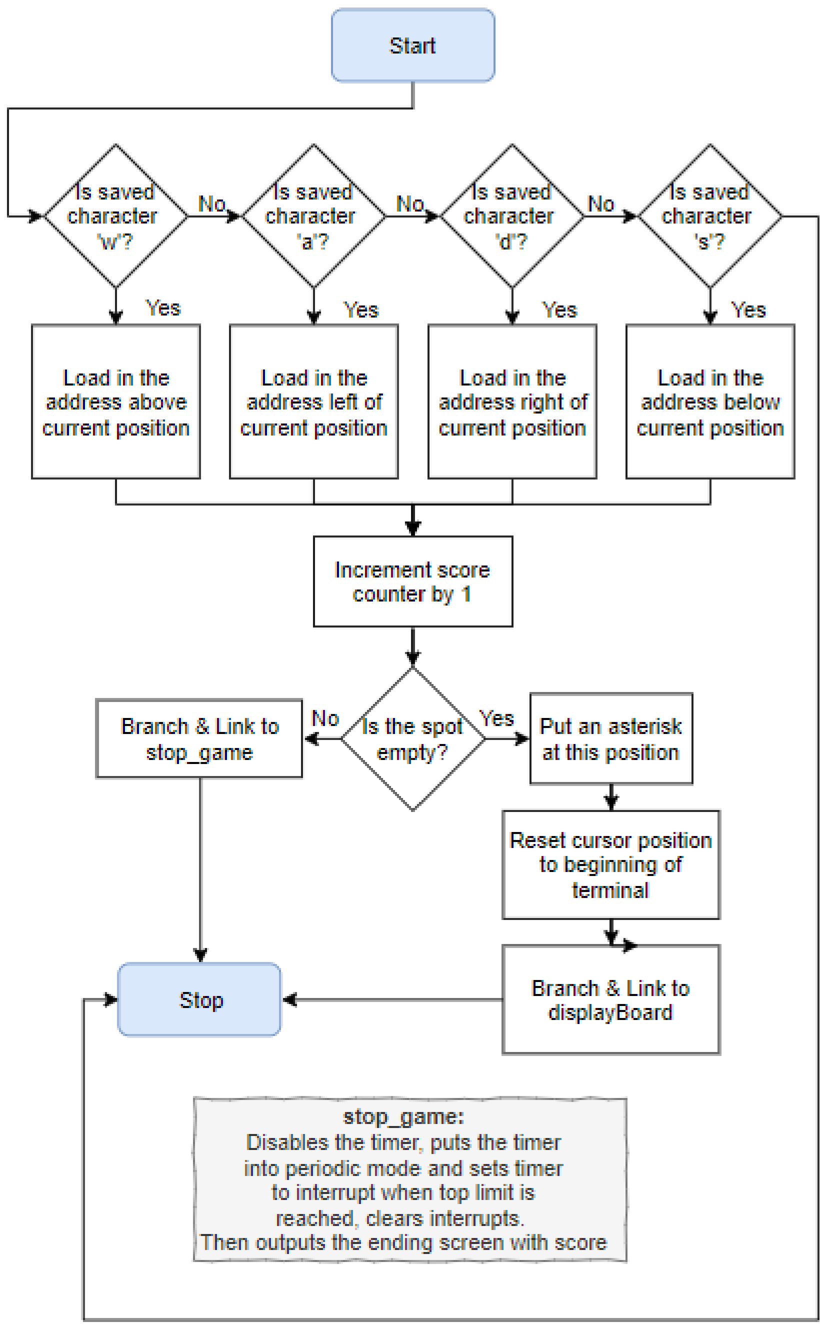

lab6:
Enables the Timer0Handler to execute every .25 seconds
Start
Branch & Link to uart_init
Branch & Link to timer_init
Stop
Branch & Link to start_game
start_game:
Displays initial start game screen

Subroutine flowchart:




UART0Handler:
Start
Branch & Link to read_character
Save character pressed in a register (current position)
Did user press 'p'?
No
Yes
Reset UART interrupt
Reset and restart the game
Stop

game_play:



Start
Is saved character 'w'?
No
Is saved character 'a'?
No
Is saved character 'd'?
No
Is saved character 's'?
Yes
Yes
Yes
Yes
Load in the address above current position
Load in the address left of current position
Load in the address right of current position
Load in the address below current position
Increment score counter by 1
Branch & Link to stop_game
No
Is the spot empty?
Yes
Put an asterisk at this position
Reset cursor position to beginning of terminal
Stop
Branch & Link to displayBoard

stop_game:
Disables the timer, puts the timer into periodic mode and sets timer to interrupt when top limit is reached, clears interrupts.
Then outputs the ending screen with score