

UART Snake Game

ECE 425: Microprocessor Systems



Written By: Samira Cordero-Morales

Professor Aaron Joseph Nanas

May 9, 2025

Introduction

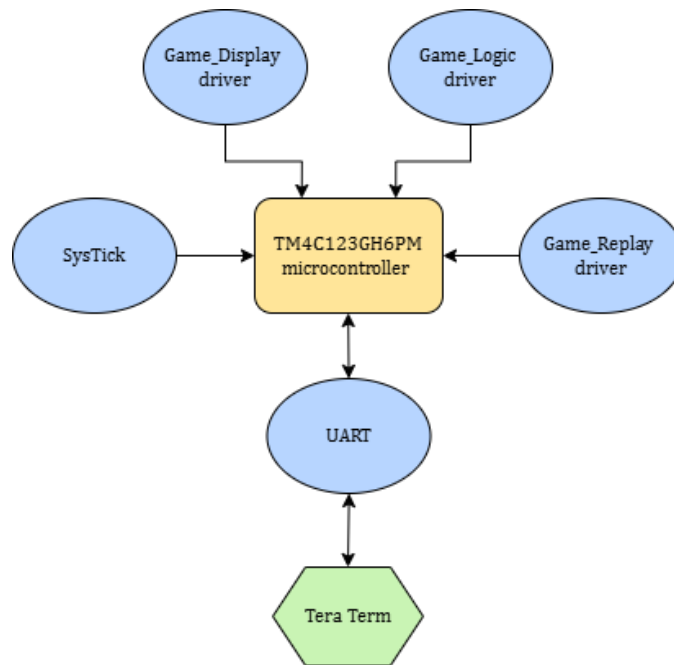
This project explores the basics of Universal Asynchronous Receiver / Transmitter (UART), a serial communication protocol that allows devices to exchange data without a clock signal, and it launches a functional snake game on the Tera Term terminal. The UART module and its registers are configured within the TM4C123GH6PM microcontroller. Other drivers are implemented to establish the game's display, logic, and replayability. Another peripheral named SysTick is configured for timing control.

Background and Methodology

The objectives of my project are to successfully launch a functional and interactive snake game with the TM4C123GH6PM microcontroller, implement basic controls to control the moving snake around, and make the game replayable whether the user wins or loses. The main embedded systems concepts being applied to my project are configuring peripherals in MCUs, synchronizing operations of components, and developing embedded displays. The outcome of my project is the launch of a functional Snake Game on the Tera Term terminal that can be easily interacted with and replayed by the user.

The hardware components used to achieve my project's objectives are a laptop that can run Tera Term, a TM4C123GH6PM microcontroller, and a USB-A to Micro-USB Cable. The software components used to achieve my project's objectives are Keil μ Vision IDE and Tera Term. The TM4C123GH6PM microcontroller peripherals I used for my project are UART and SysTick.

Block Diagram



Components Used

Hardware	Software
Laptop that can run Tera Term	Keil μ Vision IDE
Tiva C Series TM4C123G LaunchPad	Tera Term
USB-A to Micro-USB Cable	

Pinout Used

TM4C123G LaunchPad Pin	Description
PA1	UART Transmitter (TX)
PA0	UART Receiver (RX)

Analysis and Results

I gathered several drivers from my UART lab session for my final project and added a new function to my UART driver. These drivers are known as *UART0* and *SysTick_Delay* drivers. The new function in my UART driver checks if UART0's Receive FIFO is empty or not. If it is empty, the function returns a "true", and if it is fully, it returns a "full". This function allows the user to enter inputs from the keyboard.

```

251 // NEW FUNCTION-NOT FROM UART LAB!
252 bool UART0_Char_Available(void)
253 {
254     return (UART0->FR & UART0_RECEIVE_FIFO_EMPTY_BIT_MASK) == 0;
255 }

```

Figure 1: Implementation of new function in the *UART0* driver.

I created 3 drivers in order to achieve my project's objectives known as *Game_Display*, *Game_Logic*, and *Game_Reply*.

```

Game_Display.h
1 /**
2  * @file Game_Display.h
3  *
4  * @brief Header code for the Game_Display driver.
5  *
6  * This file contains the function definitions for the Game_Display driver.
7  * The #ifndef & #define preprocessor directives helps avoid compiler errors due to other files
8  * containing Game_Display.h. This suggestion was made by ChatGPT.
9  *
10 * @author Samira Cordero-Morales
11 */
12
13 #ifndef game_display_header
14 #define game_display_header
15 /**
16  * @brief The Draw_Function function draws the grid of the snake game, the snake, and the food.
17  *
18  * Boolean is used to check if a cell is filled with a 'O' or '*'. If neither, the program
19  * fills the empty cell with a '.'.
20  *
21  * @param None
22  *
23  * @return None
24  */
25 void Draw_Game(void);
26 #endif

```

Figure 2: Code documentation of *Game_Display* driver.

```
Game_Logic.h
1  /**
2   * @file Game_Logic.h
3   *
4   * @brief Header code for the Game_Logic driver.
5   *
6   * This file contains the function definitions for the Game_Logic driver.
7   * The #ifndef & #define preprocessor directives helps avoid compiler errors due to other files
8   * containing Game_Logic.h. This suggestion was made by ChatGPT.
9   *
10  * @author Samira Cordero-Morales
11  */
12
13 // game_logic.h
14 #ifndef game_logic_header
15 #define game_logic_header
16 #include <stdint.h>
17 #include <stdbool.h>
18
19 #define GRID_WIDTH 20
20 #define GRID_HEIGHT 10
21 #define MAX_SNAKE_LENGTH 50
22 #define INITIAL_SNAKE_LENGTH 3
23
24 typedef enum
25 {
26     UP,
27     DOWN,
28     LEFT,
29     RIGHT
30 } Direction;
31
32 typedef struct
33 {
34     uint8_t x;
35     uint8_t y;
36 } Coord;
37
```

Figure 3: Beginning code documentation of *Game_Replay* driver

```

38 extern Coord food;
39 extern Coord snake[MAX_SNAKE_LENGTH];
40 extern uint8_t snake_length;
41 extern Direction current_direction;
42 extern uint32_t game_score;
43
44 /**
45  * @brief The Food_Init function initializes the placement of the food.
46  *
47  * The coordinates of the food were determined by a random number, a modulo operator,
48  * and the grid's width/length.
49  *
50  * @param None
51  *
52  * @return None
53  */
54 void Food_Init(void);
55
56 /**
57  * @brief The Game_Init function initializes the game state, snake, food, and game score.
58  *
59  * The coordinates of the food were determined by a random number, a modulo operator,
60  * and the grid's width/length.
61  *
62  * @param None
63  *
64  * @return None
65  */
66 void Game_Init(void);
67
68 /**
69  * @brief The Snake_Move function updates the snake's position and movement based on its
70  * current direction.
71  *
72  * @param None
73  *
74  * @return None
75  */
76 void Snake_Move(void);
77
78 /**
79  * @brief The Snake_Grow function increments the snake's length and game score.
80  *
81  * @param None
82  *
83  * @return None
84  */
85 void Snake_Grow(void);
86
87 /**
88  * @brief The Check_Collision function checks if the snake hits a wall or itself.
89  *
90  * @param None
91  *
92  * @return False if a collision has NOT occurred; true if a collision has occurred.
93  */
94 bool Check_Collision(void);
95 #endif

```

Figure 4: Continued code documentation of *Game_Logic* driver.

```

1 /**
2  * @file Game_Replay.h
3  *
4  * @brief Header code for the Game_Replay driver.
5  *
6  * Adds the prototype of the Play_Again function.
7  * The #ifndef & #define preprocessor directives helps avoid complier errors due
8  * to other files containing Game_Replay.h.
9  *
10 * @author Samira Cordero-Morales
11 */
12
13 #ifndef game_replay_header
14 #define game_replay_header
15 #include <stdbool.h>
16
17 /**
18  * @brief The Play_Again function displays a prompt asking if the user wants to play again
19  * and checks the user's response.
20  *
21  * If the user enters an invalid input, the function prompts the user again.
22  *
23  * @param None
24  *
25  * @return False if the user entered "N/n"; true if the user entered "Y/y".
26  */
27 bool Play_Again(void);
28 #endif

```

Figure 4: Code documentation of *Game_Replay* driver.

With these drivers ready, I created the snake game through the main source file of my Keil project. This file has the most code lines because there are a lot I needed to implement: the game display, rules, game controls, and the state of the game.

```

/**
 * @file main.c
 *
 * @brief Main source code for the UART Snake Game program.
 *
 * This file contains the main entry point and function definitions for the UART Snake Game program.
 * It interfaces with the Tiva C Series TM4C123G LaunchPad and provides a display of a snake game
 * on the Tera Term terminal.
 *
 * @note For more information regarding the UART module, refer to the
 * Universal Asynchronous Receivers / Transmitters (UARTs) section
 * of the TM4C123GH6PM Microcontroller Datasheet.
 * - Link: https://www.ti.com/lit/gpn/TM4C123GH6PM
 *
 * @author Samira Cordero-Morales
 */

#include "TM4C123GH6PM.h"
#include "SysTick_Delay.h"
#include "UART0.h"
#include "Game_Display.h"
#include "Game_Logic.h"
#include "Game_Replay.h"
#include <stdbool.h>

int main(void)
{
    SysTick_Delay_Init();
    UART0_Init();

    while(1) // Outer while loop for replayability
    {
        UART0_Output_String("\x1B[2J"); // Clears TeraTerm screen
        UART0_Output_String("\x1B[H"); // Place cursor on top left before printing
        UART0_Output_String("UART Snake Game");
        UART0_Output_Newline();
        UART0_Output_String("Use W, A, S, and D keys to control the moving snake.");
        UART0_Output_Newline();
        UART0_Output_String("For every 10 points, the snake moves faster! Can you reach 50
points?!");

        UART0_Output_Newline();
        UART0_Output_String("Press the SPACEBAR to start the Snake Game! ");

        while(1) // Inner while loop to start the game
        {
            if (UART0_Char_Available())
            {
                char start_game = UART0_Input_Character();
                if (start_game == ' ')
                {

```



```

        break; // Start Snake Game
    }
}

UART0_Output_String("\x1B[2J"); // Clears TeraTerm screen
UART0_Output_String("\x1B[H"); // Place cursor on top left before printing
Game_Init();
uint32_t snake_delay_ms = 200;

while(1) // Inner while loop for playing one game
{
    if (UART0_Char_Available())
    {
        char input = UART0_Input_Character();
        switch (input)
        {
            case 'W':
            case 'w':
            {
                if (current_direction != DOWN)
                {
                    current_direction = UP;
                }
                break;
            }
            case 'S':
            case 's':
            {
                if (current_direction != UP)
                {
                    current_direction = DOWN;
                }
                break;
            }
            case 'A':
            case 'a':
            {
                if (current_direction != RIGHT)
                {
                    current_direction = LEFT;
                }
                break;
            }
            case 'D':
            case 'd':
            {
                if (current_direction != LEFT)
                {

```

```

current_direction = RIGHT;
    }
    break;
}
}

Snake_Move();

// If the snake catches the food
if (snake[0].x == food.x && snake[0].y == food.y)
{
    Snake_Grow();
    Food_Init();

    // For every 10 points, increase snake's speed
    snake_delay_ms = 200 - (game_score / 10) * 40;
    if (snake_delay_ms < 40)
    {
        snake_delay_ms = 40;
    }
}

if (Check_Collision())
{
    UART0_Output_String("\nGAME OVER! Collision hit!");
    UART0_Output_Newline();
    UART0_Output_String("Your final score is ");
    UART0_Output_Unsigned_Decimal(game_score);
    UART0_Output_String(" points.");
    UART0_Output_Newline();
    break; // Exit inner while loop
}

// When the user reaches 50 points
if (game_score >= 50)
{
    UART0_Output_String("\x1B[2J"); // Clears TeraTerm screen
    UART0_Output_String("\x1B[H"); // Place cursor on top left before

    UART0_Output_Newline();
    UART0_Output_String("CONGRATULATIONS! You won the UART Snake
Game!");

    UART0_Output_Newline();
    UART0_Output_String("Final Score: ");
    UART0_Output_Unsigned_Decimal(game_score);
    UART0_Output_Newline();
    break; // Exit inner while loop
}

```

```

        // Game display
        UART0_Output_String("\x1B[2J"); // Clears TeraTerm screen
        UART0_Output_String("\x1B[H"); // Place cursor on top left before printing

        UART0_Output_String("UART Snake Game");
        UART0_Output_Newline();
        UART0_Output_String("Use W, A, S, and D keys to control the moving snake.");
        UART0_Output_Newline();
        UART0_Output_String("For every 10 points, the snake moves faster! Can you
reach 50 points?!");

        UART0_Output_Newline();
        UART0_Output_Newline();

        UART0_Output_String("Game Score: ");
        UART0_Output_Unsigned_Decimal(game_score);
        UART0_Output_Newline();
        UART0_Output_String("Delay Speed: ");
        UART0_Output_Unsigned_Decimal(snake_delay_ms);
        UART0_Output_String(" ms");

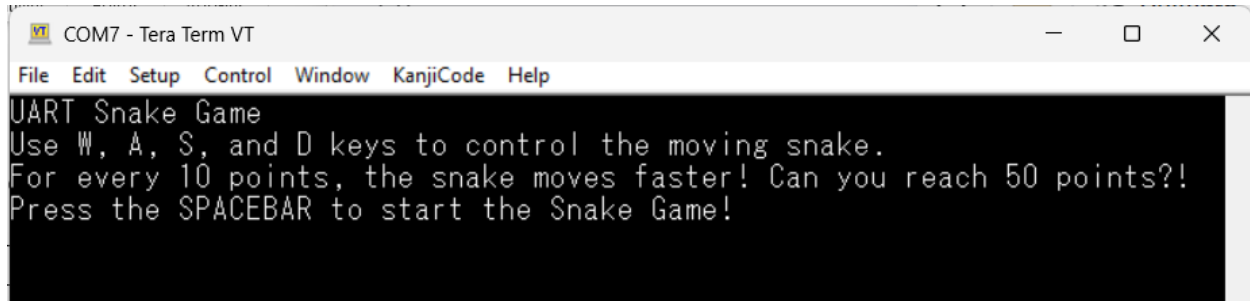
        Draw_Game();
        SysTick_Delay1ms(snake_delay_ms);
    }

    // After winning or losing the game,
    if (!Play_Again())
    {
        UART0_Output_Newline();
        UART0_Output_String("\nThank you for playing! Exiting Snake Game...");
        break;
    }
}
return 0;
}

```

Main source file of my Snake Game Keil Project

Using my USB-A to Micro-USB Cable to connect the Tiva C Series TM4C123G LaunchPad to my laptop, I built the program, verified its configuration settings, and flash my Snake Game program to the LaunchPad. Once the download is complete, I pressed the RESET button on the LaunchPad, and the Tera Term terminal displays the following output. Note that the Tera Term terminal has the same settings as the UART0 module, such as the baud rate, length of data bits, parity, and stop bits.

A screenshot of a Tera Term VT terminal window. The title bar reads "COM7 - Tera Term VT". The menu bar includes "File", "Edit", "Setup", "Control", "Window", "KanjiCode", and "Help". The terminal text is as follows:

```
UART Snake Game
Use W, A, S, and D keys to control the moving snake.
For every 10 points, the snake moves faster! Can you reach 50 points?!
Press the SPACEBAR to start the Snake Game!
```

The snake game won't start until the user presses the spacebar. Video demonstrations of the game are on my [presentation](#). The first video demonstrates one round of the snake game, and the second video demonstrates if the user wants to play the snake game again.

Conclusion

From this project, I expanded my understanding of the programming language, C. I developed drivers to organize the code of my snake game program. I installed libraries to randomize the food's position in the game or check the user's inputs. Once I had the basic snake game displayed on the Tera Term terminal, I was able to expand the complexity of the game. As the user scores higher, I increased the difficulty of the game by decreasing the SysTick delay of the moving snake. This way, there is a goal for the user to achieve and win the game. I then implemented code for game replayability, so that the user doesn't have to press RESET on the LaunchPad every time-they can get back to the game easily. My initial proposal was to use a graphical display for the project, but I was still able to display the game through UART and the Tera Term terminal and allow the user to interact with the snake game easily.