SNAKE GAME

Made by :

Adam HARB

SUMMARY

DESCRIPTION

1. Presentation of the system

The Integrated Circuit Obstacle Simulation (ICOBS) project aims to leverage hardware and software modifications to create a compelling video game using the C programming language. The project entails enhancing the hardware architecture to manage multiple sprites effectively and writing corresponding software code to control these sprites within the game environment.

Hardware Modifications:

Sprite Management: Addition of necessary elements in the hardware architecture to accommodate and manage other sprites effectively (IP sources and VGA_BASIC_ROM.vhd).

Position Control Registers: Integration of registers dedicated to controlling the position of the second sprite, enabling precise manipulation within the game world (VGA_BASIC_ROM.vhd and VGA_BASIC_ROM_Top.vhd).

VGA Controlling: Facilitating communication between the AHB-Lite bus and the VGA controller, allowing for the display of graphics on a VGA monitor based on data received from the AHB-Lite bus (ahblite_my_vga.vhd).

Definitions Library: A package to provide a convenient way to manage constants, utility functions, and component declarations for a VGA-based project, promoting modularity and reusability in the codebase. (VGA_Generic_Package.vhd).

Software Development:

The software development process involves utilizing a bootloader for testing purposes and implementing VGA peripherals for sprite control. The progression is structured as follows:

Basic Sprite Display: Initial code to display a fixed sprite on the screen, establishing foundational visual elements (sprite_egg).

Sprite Movement: Developing code to manage a sprite that moves autonomously, akin to a screen saver, introducing dynamic sprite behavior (prom_sprite_1).

Collision Management: Advancing to code managing two sprites moving with collision detection and handling, adding complexity to gameplay mechanics (sprite_egg / sprite_snake and prom_sprite_1/sprite_egg).

Game Development:

The game's objective is to illustrate the full functionality of the system. Key components of the game include:

User-Controlled Movement: Enabling players to navigate a sprite within the visualized area of the screen (vidon) using input controls like buttons and switches.

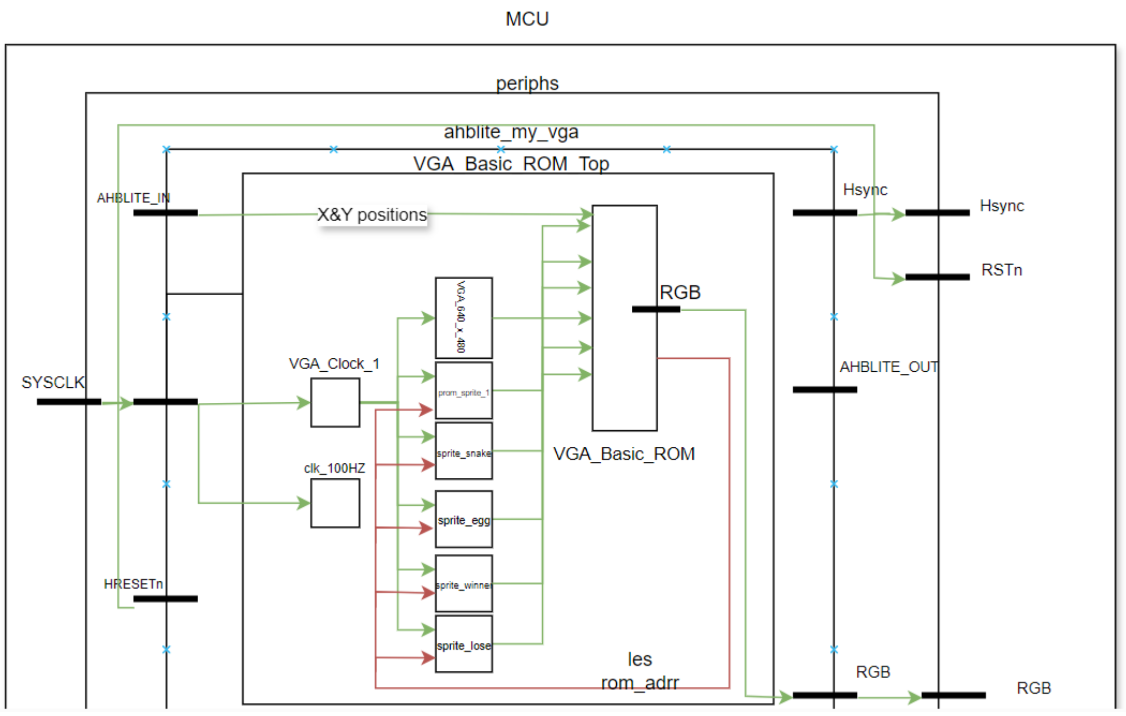
Objective and Victory Conditions: The player has an objective which is collecting the eggs while avoiding the fire , leading to victory or defeat based on their score.

Additional Elements: Incorporating optional features such as managing multiple sprites, background imagery, start and end screens, and scoring mechanisms.

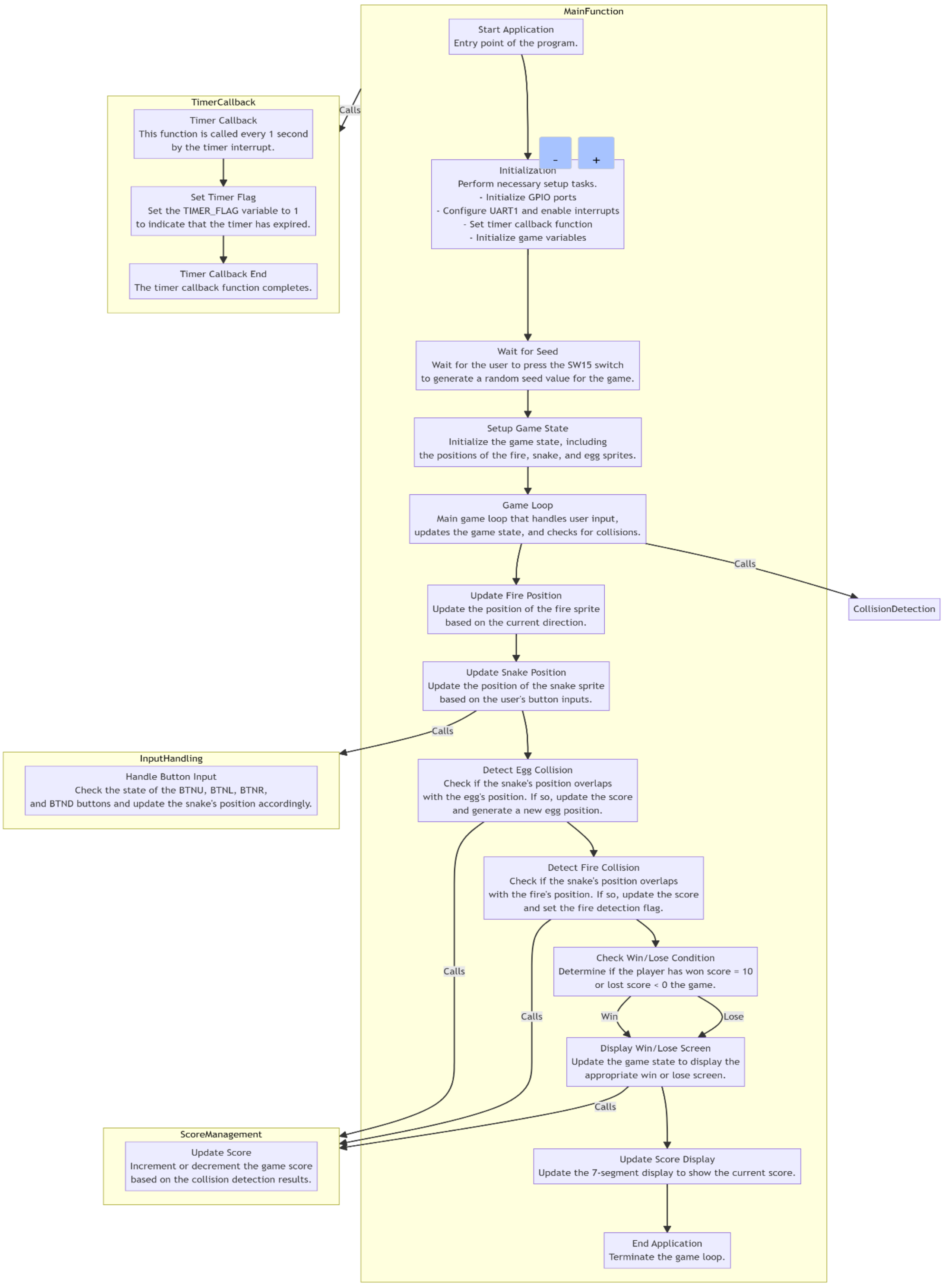
System Overview:

The developed system integrates hardware and software components to create an interactive gaming experience. Players interact with the game through user inputs, controlling sprites within the game world. The hardware enhancements enable smooth sprite management, while the software implementation adds depth to gameplay mechanics, including sprite movement and collision detection.

2. Diagram of the hardware architecture



3. Flowchart of the C code


MainFunction
Start Application
Entry point of the program.
Initialization
Perform necessary setup tasks.
- Initialize GPIO ports
- Configure UART1 and enable interrupts
- Set timer callback function
- Initialize game variables
-
+
Calls
TimerCallback
Timer Callback
This function is called every 1 second by the timer interrupt.
Set Timer Flag
Set the TIMER_FLAG variable to 1 to indicate that the timer has expired.
Timer Callback End
The timer callback function completes.
Wait for Seed
Wait for the user to press the SW15 switch to generate a random seed value for the game.
Setup Game State
Initialize the game state, including the positions of the fire, snake, and egg sprites.
Game Loop
Main game loop that handles user input, updates the game state, and checks for collisions.
Calls
CollisionDetection
Update Fire Position
Update the position of the fire sprite based on the current direction.
Update Snake Position
Update the position of the snake sprite based on the user's button inputs.
Calls
InputHandling
Handle Button Input
Check the state of the BTNU, BTNL, BTNR, and BTND buttons and update the snake's position accordingly.
Detect Egg Collision
Check if the snake's position overlaps with the egg's position. If so, update the score and generate a new egg position.
Detect Fire Collision
Check if the snake's position overlaps with the fire's position. If so, update the score and set the fire detection flag.
Calls
Check Win/Lose Condition
Determine if the player has won score = 10 or lost score < 0 the game.
Calls
Win
Lose
Display Win/Lose Screen
Update the game state to display the appropriate win or lose screen.
Calls
ScoreManagement
Update Score
Increment or decrement the game score based on the collision detection results.
Update Score Display
Update the 7-segment display to show the current score.
End Application
Terminate the game loop.

4. Explanation of the overall operation

Here is a detailed Mermaid flow diagram with explanations for the code:Explanation of the Mermaid flow diagram:

Main Function:

- *Start*: Entry point of the program.

- Initialization*:* Perform necessary setup tasks, such as initializing GPIO ports, configuring UART1, enabling interrupts, setting the timer callback function, and initializing game variables.

- Wait For Seed: Wait for the user to press the SW15 switch to generate a random seed value for the game.

- Setup Game State: Initialize the game state, including the positions of the fire, snake, and egg sprites.

- Game Loop: The main game loop that handles user input, updates the game state, and checks for collisions.

- Update Fire Position: Update the position of the fire sprite based on the current direction.

- Update Snake Position: Update the position of the snake sprite based on the user's button inputs.

- Detect Egg/Snake Collision: Check if the snake has collided with the egg. If so, update the score and generate a new egg position.

- Detect Fire/Snake Collision: Check if the snake has collided with the fire. If so, update the score and set the fire detection flag.

- Check Win/Lose: Determine if the player has won (score = 10) or lost (score < 0) the game.

- Display Win/Lose: If the player has won or lost, update the game state to display the appropriate win or lose screen.

- Update Score on Display: Update the 7-segment display to show the current score.

- Delay: Introduce a short delay to control the game's frame rate.

In conclusion the flow diagram represents the overall structure and logic of the provided code, with subgraphs for the main function, collision detection, position updates, and display of the game state. The explanations within the diagram provide detailed insights into the functionality of each component, helping the user understand the code's behavior and structure.

5. Link between the software and the hardware

The link between software and hardware is established through well-defined interfaces, protocols, and communication channels.

Software sends commands or data to the hardware to initiate specific actions or retrieve information. On te other hand, Hardware responds to software requests by executing the required operations and providing feedback or results.

This bidirectional communication ensures synchronization and coordination between the software and hardware components, enabling seamless interaction and operation of the overall system.

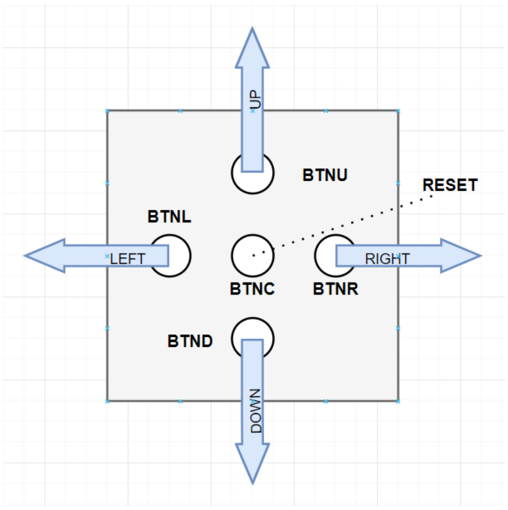
Ultimately, the collaboration between software and hardware enables the execution of the video game, with software driving gameplay and logic while hardware handles the display, input/output, and other low-level operations.

6. How to Play?

Welcome to SNAKE GAME ! Follow these instructions to play:

Start the Game: Switch the SW15 switch to position 1 to initiate the game.

Navigate the Snake: Utilize the directional buttons – BTNU (Up), BTNR (Right), BTND (Down), and BTNL (Left) – to control the movement of the snake.



Objective: The aim is to guide the snake to collect eggs scattered across the screen.

Scoring: Each time the snake picks up an egg, you earn 1 point.

Egg Respawn: After collecting an egg, a new one will appear at a random position on the screen.

Avoid the Fire: Be cautious and avoid touching the fire hazards.

Penalties: Accidentally touching the fire results in a deduction of 1 point from your score.

Score Display: Monitor your score on the 7-segment display.

Victory Condition: Reach a score of 10 to emerge victorious.

Defeat Condition: If your score drops below 0, the game ends in defeat.

Reset: To start over, press the BTNC button to reset the game.

Most Important: Have fun playing!