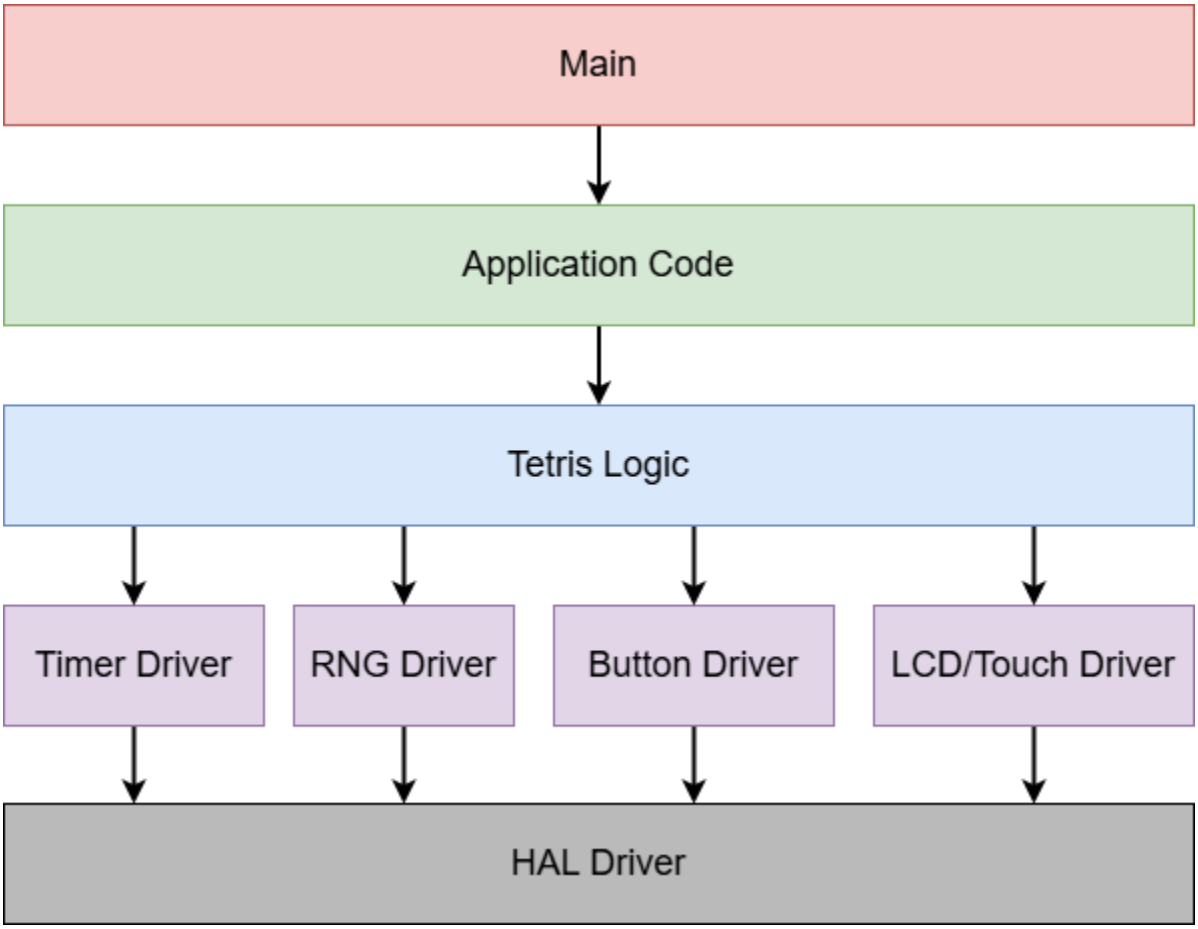
Code Documentation - *Tetris*

GitHub Link: https://github.com/ArjunD112/ECEN2370-FinalProject

---

Code Hierarchy



Button Driver

Button_Init_InterruptMode()

- Arguments: void
- Return Type: void

Initializes the user button peripheral and enables global interrupts for it.

Timer Driver

Timer7Init()

- Arguments: void
- Return Type: void

Initializes Timer 7 with a frequency of 1 Hz.

Timer7DeInit()

- Arguments: void
- Return Type: void

Deinitializes Timer 7.

StartTimer()

- Arguments: void
- Return Type: void

Initializes and starts Timer 7 and enables global interrupts for it. Calls **Timer7Init()**.

StopTimer()

- Arguments: void
- Return Type: uint16_t

Stops Timer 7, returns time remainder in CNT register, deinitializes Timer 7. Calls **Timer7DeInit()**.

TIM_ClearInterruptFlag()

- Arguments: void
- Return Type: void

Clears UIF flag in Timer 7's SR.

RNG Driver

RNG_Init()

- Arguments: void
- Return Type: void

Initializes the RNG peripheral.

RNG_DeInit()

- Arguments: void
- Return Type: void

Deinitializes the RNG peripheral.

RNG_GenRandNum()

- Arguments: void
- Return Type: uint32_t

Initializes the RNG peripheral, generates a random 32-bit number, transforms the result into a number between 2 and 8 (inclusive), deinitializes the RNG peripheral, and returns

the result. Calls **RNG_Init()** and **RNG_DeInit()**. Note that the repeated initializing/deinitializing is critical to function properly.

LCD Driver

This code was provided to us by the instructional staff, but one extra function, **DrawHorizontalLine()**, was added and the rest were unchanged.

LCD_Draw_Horizontal_Line()

- Arguments: uint16_t *x*, uint16_t *y*, uint16_t *len*, uint16_t *color*
- Return Type : void

Draws a horizontal line to the LCD screen beginning at (*x*, *y*) and ending at (*x* + *len*, *y)* which has color *color*.

Tetris Logic

Tetrominoe Struct

- uint8_t Name
- uint8_t Structure[4][4]
- uint16_t Color
- uint8_t Width
- uint8_t Height
- int8_t XPosition
- int8_t YPosition

Board Struct

- int8_t Field[14][12]

BuildTetrominoe()

- Arguments: uint8_t *c*, Board *b*
- Return Type: Tetrominoe

Builds a 4x4 matrix and sets the properties corresponding to the argument *c*. See the header file for the macros for each type of tetromino. Surveys the current gameboard *b* and checks spawn for overlap with previously set tetrominoes. If no overlap, draws the new tetromino on the LCD screen and returns the new tetromino. Else, the local variable *end*, which acts as a status flag, is set to true, and nothing is drawn. However, the loop in **main()** will draw the end screen. Calls **CheckOverlap()** and **DrawTetrominoe()**.

RotateTetrominoe()

- Arguments: Tetrominoe *oldTetrominoe*, Board *b*
- Return Type: Tetrominoe

Clears *oldTetrominoe* from LCD screen. Rotates *oldTetrominoe* by taking the transpose of its matrix, then multiplying the horizontally-mirrored identity matrix by the transpose. The result is a 90° clockwise rotation. Checks overlap of result with existing set tetrominoes in *b*. If no overlap, rotation is executed and new tetromino is drawn and returned. Else, *oldTetrominoe* is drawn and returned. Calls **CheckOverlap()** and **DrawTetrominoe()**.

ShiftTetrominoe()

- Arguments: Tetrominoe *oldTetrominoe*, Board *b*, uint8_t *dir*
- Return Type: Tetrominoe

Clears *oldTetrominoe* from LCD screen. Shifts *oldTetrominoe* left, right, or down, depending on *dir*. Checks overlap of result with existing set tetrominoes in *b*. If no overlap, shift is executed and new tetromino is drawn and returned. Else, *oldTetrominoe* is drawn and returned. Calls **CheckOverlap()** and **DrawTetrominoe()**.

NewTetrominoe()

- Arguments: Board *b*
- Return Type: Tetrominoe

Uses RNG to randomly generate a new tetromino. Calls **RNG_GenRandNum()** and **BuildTetrominoe()**.

SetTetrominoe()

- Arguments: Tetrominoe *t*, Board *b*
- Return Type: Board
- Sets cells in *b* occupied by *t* to the value of the *Name* property of *t*. Draws updated gameboard *b*, checks for level clear(s), and returns updated gameboard *b*. Calls **UpdateBoard()** and **CheckTetris()**.

CheckCollision()

- Arguments: Tetrominoe *t*, Board *b*
- Return Type: bool
- Checks for collision between occupied cells *after a single hypothetical shift downwards* and previously set tetrominoes. If there is a collision ahead, returns *true*, else returns *false*.

CheckOverlap()

- Arguments: Tetrominoe *t*, Board *b*
- Return Type: bool

Checks for overlap of *t* with set tetrominoes in *b*. If there is an overlap, returns *true*, else returns *false*.

CheckRow()

- Arguments: Board *b*
- Return Type: bool

Surveys *b* for filled rows. If there is a filled row, returns *true*, else returns *false*.

CheckTetris()

- Arguments: Board *b*
- Return Type: Board

Clears filled rows and shifts upper rows down, redraws and returns new board. Calls **CheckRow()** and **UpdateBoard()**.

InitBoard()

- Arguments: void
- Return Type: Board

Draws empty board to LCD screen and initializes and returns empty gameboard with boundaries defined. Calls **DrawBoard()**.

DrawBoard()

- Arguments: void
- Return Type: void

Draws gridlines, walls, and floor to the LCD screen. Calls functions from provided code and **LCD_Draw_Horizontal_Line()**.

UpdateBoard()

- Arguments: Board *b*
- Return Type: void

Surveys *b* and draws a block of corresponding color in each cell to the LCD screen. Calls **DrawBlock()**.

DrawBlock()

- Arguments: uint16_t *x*, uint16_t *y*, uint16_t *color*
- Return Type: void

Draws a square at the coordinates (*x*, *y*) of color *color* to the LCD screen. Calls **LCD_Draw_Vertical_Line()**.

DrawTetrominoe()

- Arguments: Tetrominoe *tetrominoe*, uint16_t *color*
- Return Type: void

Draws *tetrominoe* to LCD screen. Calls **DrawBlock()**.

DrawStartScreen()

- Arguments: void
- Return Type: void

Draws animation and start screen to LCD screen.

DrawEndScreen()

- Arguments: void
- Return Type: void

Draws end screen and displays time elapsed.

ReturnEnd()

- Arguments: void
- Return Type: bool

Returns local Boolean variable *end*.

IncTime()

- Arguments: void
- Return Type: void

Increments local uint16_t *time* by the frequency of Timer 7.

Application Code

Much of this code was provided, only additions were made, which are noted below.

ApplicationInit()

- Arguments: void
- Return Type: void

IN ADDITION: Calls **Button_Init_InterruptMode** and **DrawStartScreen()**.

Touchscreen Interrupt Handler (EXTI15_10_IRQHandler)

- Arguments: void
- Return Type: void

IN ADDITION: Upon a fired touchscreen interrupt, will check whether the game is started or not by checking the state of *started*, a local Boolean initialized as *false*. If the game is not started (*started* == *false*), the start screen is cleared, the gameboard is drawn, Timer 7 is started, *started* is set to *true*, and the game begins. Calls **InitBoard()**, **NewTetrominoe()**, and **StartTimer()**. Else, the x-coordinate of the touch will be determined. If the x-coordinate is on the left half of the screen, the live tetromino will shift left. Else, it will shift right. Calls **ShiftTetrominoe()**.

Button Interrupt Handler (EXTI0_IRQHandler)

- Arguments: void
- Return Type: void

Disables button IRQ, rotates tetromino 90° clockwise, clears pending interrupt bit in EXTI peripheral, enables button IRQ. Calls **HAL_NVIC_EnableIRQ()**, **HAL_NVIC_DisableIRQ()**, and **RotateTetrominoe()**.

Timer 7 Interrupt Handler (TIM7_IRQHandler)

- Arguments: void
- Return Type: void

Disables Timer 7 IRQ and checks for a tetromino collision. If there is a collision, the tetromino is set, the board is updated, and a new tetromino is spawned. Calls **CheckCollision()**, **SetTetrominoe()**, and **NewTetrominoe()**. Else, the tetromino is shifted down. Calls **ShiftTetrominoe()**. Increments the time elapsed, clears the pending interrupt bit for Timer 7, and enables Timer 7 IRQ. Calls **IncTime()** and **TIM_ClearInterruptFlag()**. Calls **HAL_NVIC_EnableIRQ()** and **HAL_NVIC_DisableIRQ()**.

Main

main()

- Arguments: void
- Return Type : void

IN ADDITION : In the infinite loop, an *end* flag is checked. If *true*, draws the end screen and disables touchscreen interrupt. Else, does nothing. Calls **Return TypeEnd()**, **DrawEndScreen()**, **HAL_NVIC_DisableIRQ()**.