EEET2481 Embedded System Design and Implementation

Semester 1, 2025

Assessment 3 Report

# **1. Introduction**

We present the development and demonstration of two embedded projects on the Nuvoton M487 development board:

**Exercise 1:** Rewrite the Tutorial-8 LCD demo.

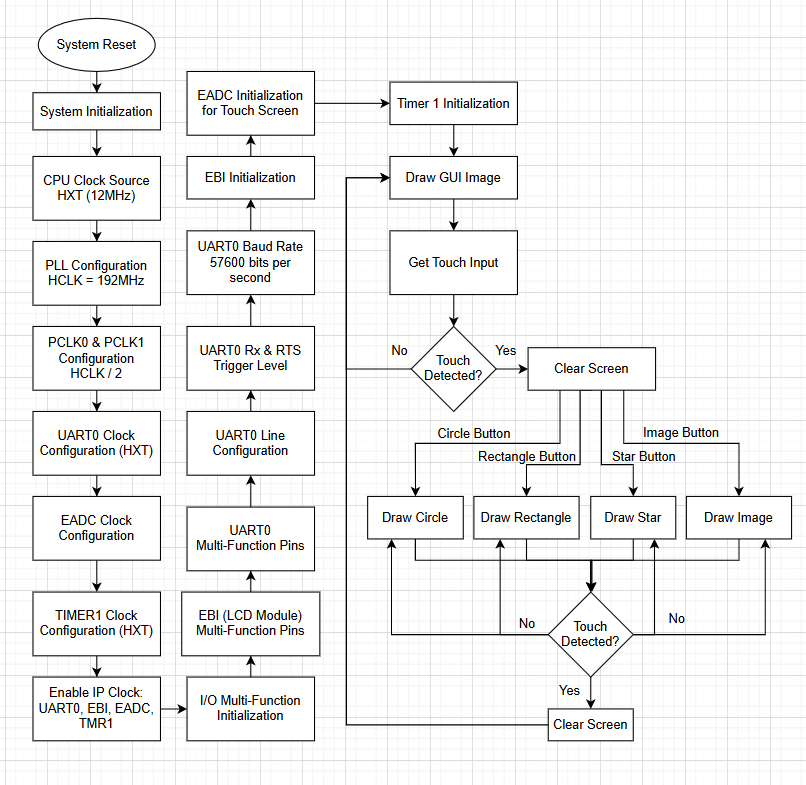
**Exercise 2:** Implements a full Tetris game with screen display and (joystick + buttons) for controls.

Github link for exercise 2: <https://github.com/truongbaan/NuvoM487Tetris-An-and-Minh>

# **2. Exercise 1: Rewriting Tutorial 8**

This task involved re-implementing hardware initialization without using predefined parameters and adding new LCD drawing functions.

## **Flowchart**



# **3. Exercise 2 – Tetris Game on M487 Board**

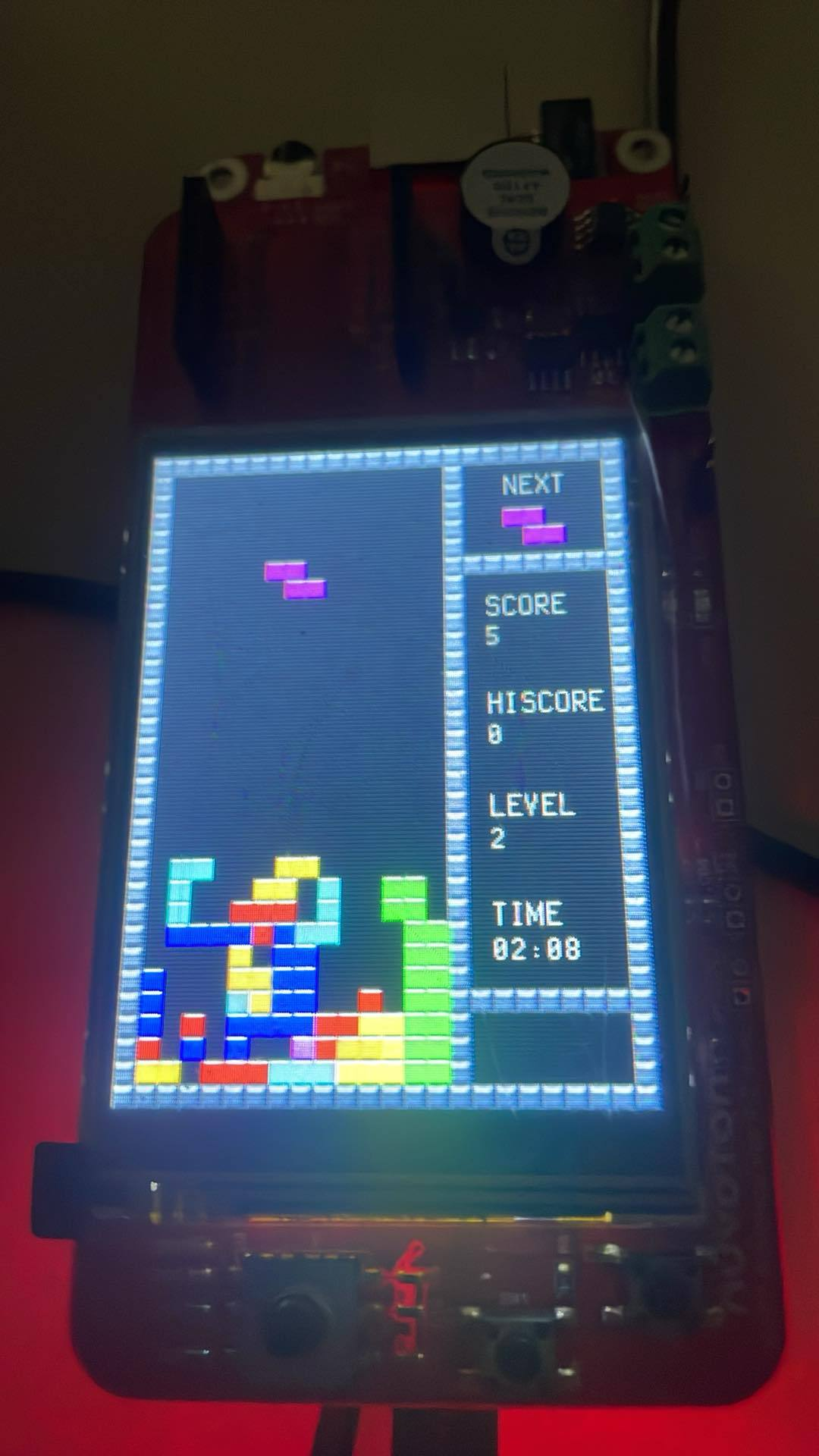
In Exercise 2, we built a classic Tetris game on the M487 board. Key features include:

* Game field: 24×32 units (10×10 pixel blocks), with a 14-unit-wide play area and 7-unit-wide info panel.
* Controls: 4-way joystick moves and rotates pieces; a single push-button (SW1) pauses and advances between game stages; another button (SW2) for hard-drop.

**Stages in the game:**

* **Start Screen**: This will present the image of tetris (You can either touch screen to start or use any button to enter the game)
* **GamePlay**: falling pieces, scoring, level progression.
* **Pause Mechanic**: freezes time and displays pause. (not really a screen but pause stage)
* **Game Over**: when blocks reach the top, the game ends and displays Game Over Screen.
* **High Score**: shows ranked top scores. Press any buttons to return to the GamePlay.

## **3.1 Game Design Summary**



**Figure 1**: Actual Game Screen

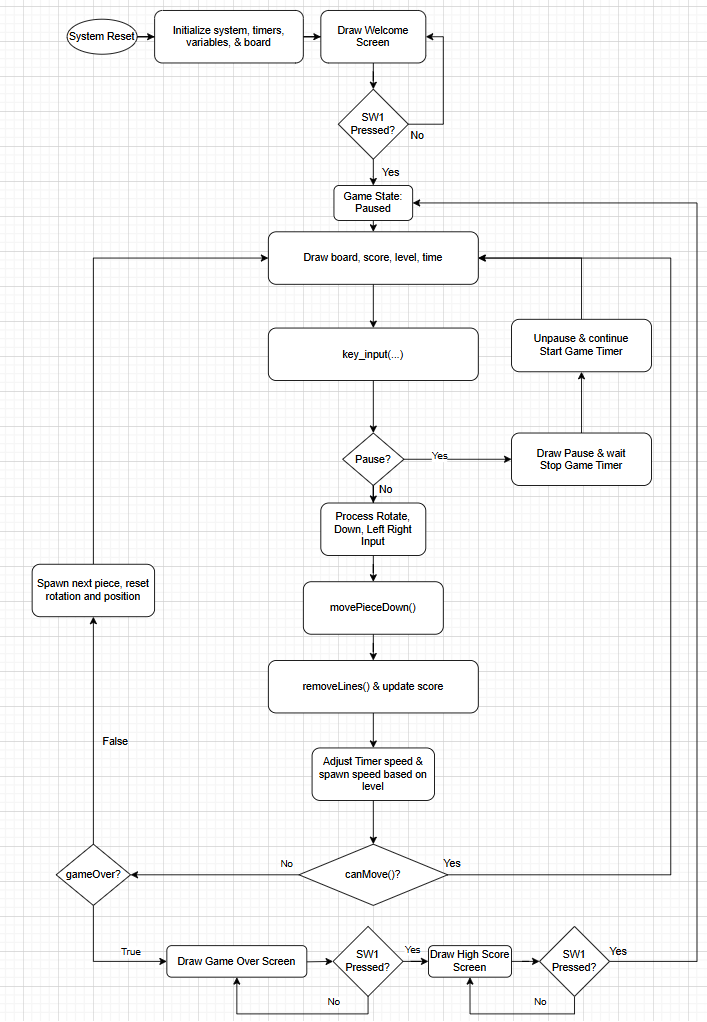
**Play Field:**

* Dark grid of dots (24 rows × 14 columns) framed by a 1-unit-thick border.
* Active pieces are rendered in their unique colors; settled blocks form the pile.

**Info Panel:**

* **NEXT**: previews the upcoming piece in its color and orientation.
* **SCORE / HISCORE**: current score increments by 1 per line cleared; HISCORE persists between runs.
* **LEVEL**: starts at 1 and increases every 5 lines (max 10), which accelerates piece drop speed.
* **TIME**: elapsed runtime in mm:ss using Timer 0 at 1 Hz.
* **PAUSE**: reminder that SW1 toggles pause using an interrupt.

## **3.2 Flowchart and FSM**



**Figure 2:** Tetris Game Logic Flow Chart

## **3.3 Game Mechanics (Functions)**

| **Function** | **Category** | **Purpose / Usage** |
| --- | --- | --- |
| **SYS\_Init** | Initialization | Configure system clock, EBI, UART0 pins (MFP), and basic GPIO settings |
| **GPIO\_Init** | Initialization | Set up all GPIO ports/pins (joystick inputs, SW1, LCD control lines) before use. |
| **Timer0\_Init** | Initialization | Configure Timer0 in periodic mode at 1 Hz to track elapsed play time (mm:ss). |
| **Timer1\_Init** | Initialization | Configure Timer1 for the falling‐piece “tick” interrupts, with reload values based on current level (0.5 s → 0.05 s). |
| **disableSelectiveInterrupts** | Interrupt Management | Temporarily disable specific NVIC interrupt lines while critical sections run |
| **enableSelectiveInterrupts** | Interrupt Management | Re-enable previously disabled NVIC interrupt lines |
| **TMR0\_IRQHandler** | ISR | Fired every 1 s—increments the played‐time counter and requests a redraw of the time display. |
| **TMR1\_IRQHandler** | ISR | Fired at the current drop interval—sets a flag so the main loop will advance the active piece downward. |
| **GPA\_IRQHandler** | ISR | Handles joystick “UP” (rotate) and “DOWN” (hard drop) button presses on port A |
| **GPC\_IRQHandler** | ISR | Handles joystick “LEFT” and “RIGHT” movement presses on port C. |
| **GPG\_IRQHandler** | ISR | Handles SW1 presses on port G for Pause/Stage‐change events. |
| **Default\_Handler** | ISR Fallback | Catches any unexpected interrupts routed to this default vector. |
| **HardFault\_Handler** | Fault Handler | Low‐level HardFault entry point that captures register state. |
| **HardFault\_HandlerC** | Fault Handler | C routine that logs or handles a HardFault once control has been transferred here. |
| **createBoard** | Game Logic | Initialize the 2D play‐field array (24×32) to all zeros (empty). |
| **printBoard** | Rendering | Draw the current play‐field matrix to the LCD, including settled pieces and the active piece. |
| **generatePiece** | Game Logic | Randomly select one of the seven Tetris shapes, assign it as the “next” piece and prepare the new active piece. |
| **canMove** | Collision Detection | Check whether the active piece can legally move or rotate to a given position without colliding or exiting the boundary |
| **movePieceDown** | Movement | For auto moving down and for speed up going down |
| **movePieceLeft** | Movement | The controlled piece goes left by 1 |
| **movePieceRight** | Movement | The controlled piece goes right by 1 |
| **rotatePiece** | Movement | Rotate the active piece 90° clockwise in response to an UP joystick press (with collision correction). |
| **computeDropDistance** | Game Logic | Calculate how many empty rows lie below the active piece (used by hard‐drop). |
| **hardDrop** | Movement | Instantly move the active piece to its lowest legal position and settle it into the play‐field |
| **removeLines** | Game Logic | Scan the play‐field for any full horizontal lines; remove them and shift all above rows down, updating the score |
| **isGameOver** | Game Logic | Check if any block cell has reached the top row |
| **key\_input** | Input Handling | Central polling function for joystick and SW1 flags; debounces and routes to movement, rotate, pause, or stage‐change actions. |
| **insertScore** | High Score | Insert the current final score into the leaderboard array in sorted order, capping to the top n entries |
| **drawScoreScreen** | Rendering | Display the HIGH SCORE stage layout with the sorted leaderboard on the LCD. |
| **drawNextPiece** | Rendering | Render the “next piece” preview box in the info area |
|  |  |  |
| **drawGameScore** | Rendering | Draw the current game score value in the status area |
| **drawTopScore** | Rendering | Draw the all‐time high score (from previous sessions) in the status area. |
| **drawLevel** | Rendering | Show the current level (1–10) in the status area |
| **drawTime** | Rendering | Show the elapsed play time (mm:ss) using Timer0 data |
| **drawPause** | Rendering | Overlay “PAUSED” indicator and instructions when the game is suspended. |
| **drawUnpause** | Rendering | Remove the pause overlay and resume normal display |
| **drawGameOver** | Rendering | Overlay the “GAME OVER” screen and prompt to view high scores |
| **gameplay** | Core | The main game‐state loop: checks flags (timer, input), moves/rotates pieces, calls collision, line‐clear, rendering, and state transitions |
| **main** | Starting point | Calls all initializers, then enters the game. (start at game start screen) |

## **3.4 Timing and Waveforms**

Each of the key timings for the corresponding timers used in the project were verified using an oscilloscope through the timer interrupt toggling a GPIO (PB.2 in this case) to measure the timing. Some of the measurements (TIMER0 & TIMER1) were taken over a complete cycle therefore the measurement should be divided by 2 to get the timing of the actual delay generated by the interrupt.



**Figure 3:** TIMER0 - Game Timer - 1 Second Counter



**Figure 4:** TIMER1 - Game Speed Delay - Level 1 - 0.5 Second



**Figure 5:** TIMER1 - Game Speed Delay - Level 2 - 0.45 Second



**Figure 6:** TIMER1 - Game Speed Delay - Level 3 - 0.4 Second



**Figure 7:** TIMER1 - Game Speed Delay - Level 4 - 0.35 Second



**Figure 8:** TIMER1 - Game Speed Delay - Level 5 - 0.3 Second



**Figure 9:** TIMER1 - Game Speed Delay - Level 6 - 0.25 Second



**Figure 10:** TIMER1 - Game Speed Delay - Level 7 - 0.2 Second



**Figure 11:** TIMER1 - Game Speed Delay - Level 8 - 0.15 Second



**Figure 12:** TIMER1 - Game Speed Delay - Level 9 - 0.1 Second

**Figure 13:** TIMER0 - Game Speed Delay - Level 10 - 0.05 Second

****

**Figure 14:** TIMER2 - Game Piece Spawn Delay - Level 1 - 0.5 Second

****

**Figure 15:** TIMER2 - Game Piece Spawn Delay - Level 2 - 0.45 Second

****

**Figure 16:** TIMER2 - Game Piece Spawn Delay - Level 3 - 0.4 Second

****

**Figure 17:** TIMER2 - Game Piece Spawn Delay - Level 4 - 0.35 Second

****

**Figure 18:** TIMER2 - Game Piece Spawn Delay - Level 5 - 0.3 Second

****

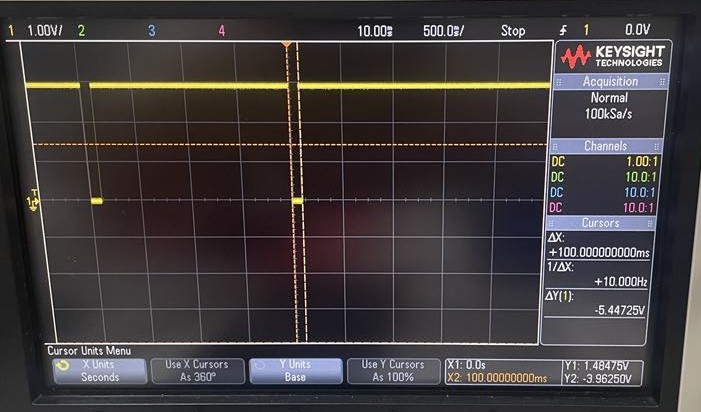
**Figure 19:** TIMER2 - Game Piece Spawn Delay - Level 6 - 0.25 Second

****

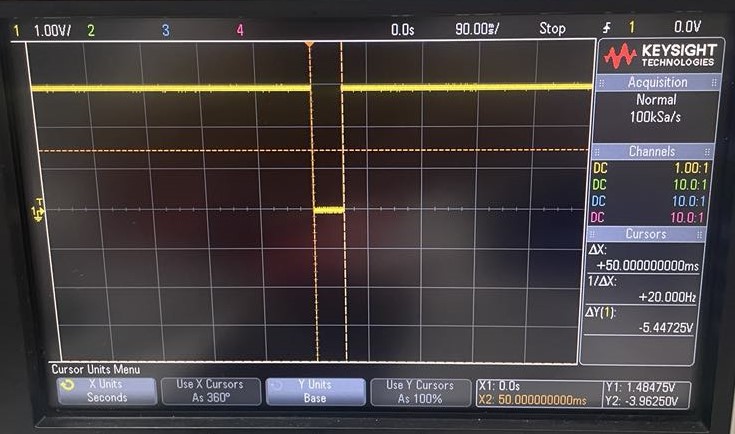
**Figure 20:** TIMER2 - Game Piece Spawn Delay - Level 7 - 0.2 Second

****

**Figure 21:** TIMER2 - Game Piece Spawn Delay - Level 8 - 0.15 Second

****

**Figure 22:** TIMER2 - Game Piece Spawn Delay - Level 9 - 0.1 Second

****

**Figure 23:** TIMER2 - Game Piece Spawn Delay - Level 10 - 0.05 Second

# **4. Team Contribution Summary**

| Team Member | Responsibility | Contribution Score |
| --- | --- | --- |
| Truong Ba An (s3999568) | Tetris game logic, Game Logic debugging | 50% |
| Minh Duc Vu (s4054648) | LCD draw functions, Tetris game button controls, Hardware & game debugging | 50% |