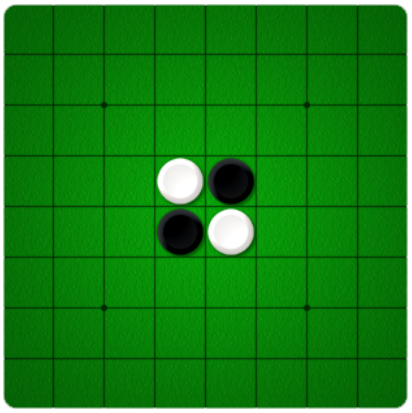
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

EMBEDDED SYSTEMS

PROJECT - AN OTHELLO GAME

REPORT OF MILESTONE 3

25 May 2019
Professor: A. Dollas



Working Group LAB41140573
Vasilakopoulos Panagiotis A.M:2015030048
Maragkaki Maria A.M:2015030153

PROJECT PURPOSE

The purpose of this Project is the implementation of the Othello game where it can play a game against another human player or against another STK500. The Othello game is played on a 8x8 chessboard with cells of the same color. On the horizontal axis the columns are numbered from 1 to 8 and on the vertical axis the lines named from A to H. There are two-colored checkers, black and white. Each player choose one color. At the start of the game 4 checkers are placed in the middle of the chessboard. Two white in the positions D4 and E5 and two black in the positions D5 and E4. The first player places a checker near of the opponent's checker. If the opponent's checkers are closed into the first's player checkers, then they change color. The goal is at the end of the game one of the players to have the most checkers on his color. If the number of checkers is the same then the game is tie. The game ends when all of the squares have a checker or when there are no valid moves.

PURPOSE OF MILESTONE 3

The purpose of the third and last Milestone was the completion of the Othello game, with full support of the strategy that AVR will play and full communication protocol. The strategy of second Milestone was the AVR plays the first valid move he had found. In this Milestone AVR tries to find the best move that will give to him the most amount of checkers.

STRATEGY

First of all were implemented and initialized 15 chessboards in the memory with the same logic as the second Milestone. More specifically in one cell of the memory i.e. in 8 bits correspond 4 cells of the chessboard. At the most-significant bits are written the colors of the cells (black or white) and at the least-significant bits are written the enable bits where they show if the cell is empty or full. The main chessboard is copied to each of the 15 chessboards. The strategy is that AVR takes every valid move that is saved and plays it individually in one parallel chessboard. E.g. the first valid move is played in the first parallel chessboard and respectively for the second. Afterwards it counts in each chessboard the sum of its checkers. Then according to the applicable greedy algorithm, AVR decides to play the specific legal move, that will offer the maximum sum of AVR's tokens. This is done by the following procedure. The microcontroller scans the valid moves and for each move that its finds, it increments a counter, which corresponds to the parallel chess board numbering. Then plays this move in the specific chessboard, stores his score and compares it with the maximum score. In the end of this procedure, decides the move that corresponds to his maximum score.

This algorithm is more efficient in speed and in storage space in program memory (flash memory) than the Monte Carlo algorithm. However, works well only if the amount of the available moves is less than or equal with 15.

The space requirements for this design are: 32 bytes for the communications buffers, 16 bytes to store the valid moves of both players, 16 bytes for the main chessboard and 15x16 bytes for the parallel chessboards. So the internal storage memory must be strictly above 304 bytes. The sram has 1 kbyte of space, so we guarantee that there is not a space requirement fault.

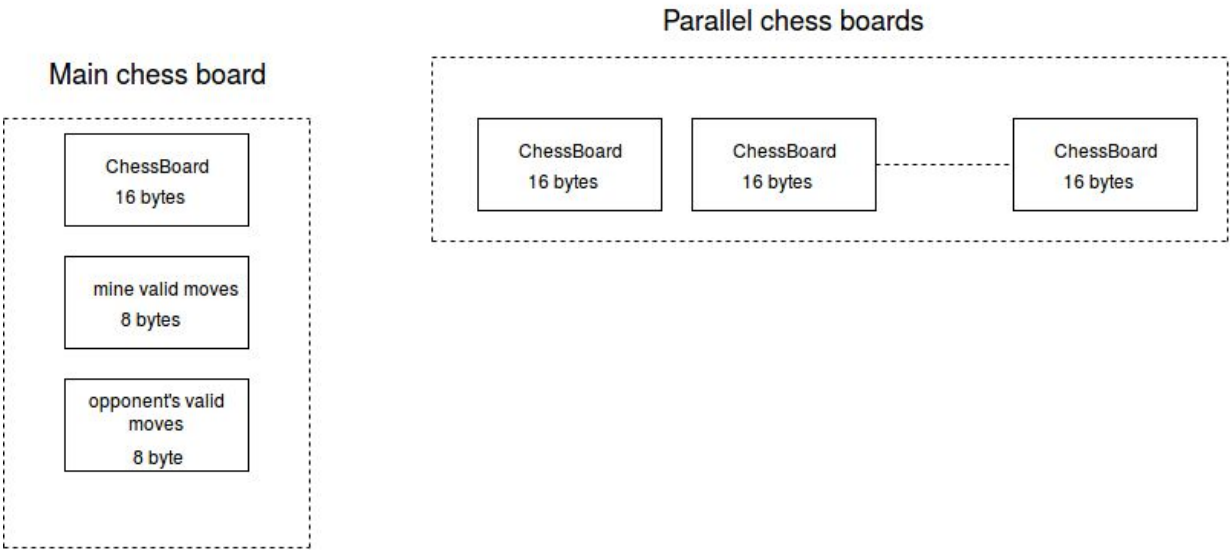


Image 1: Implementation of chessboards

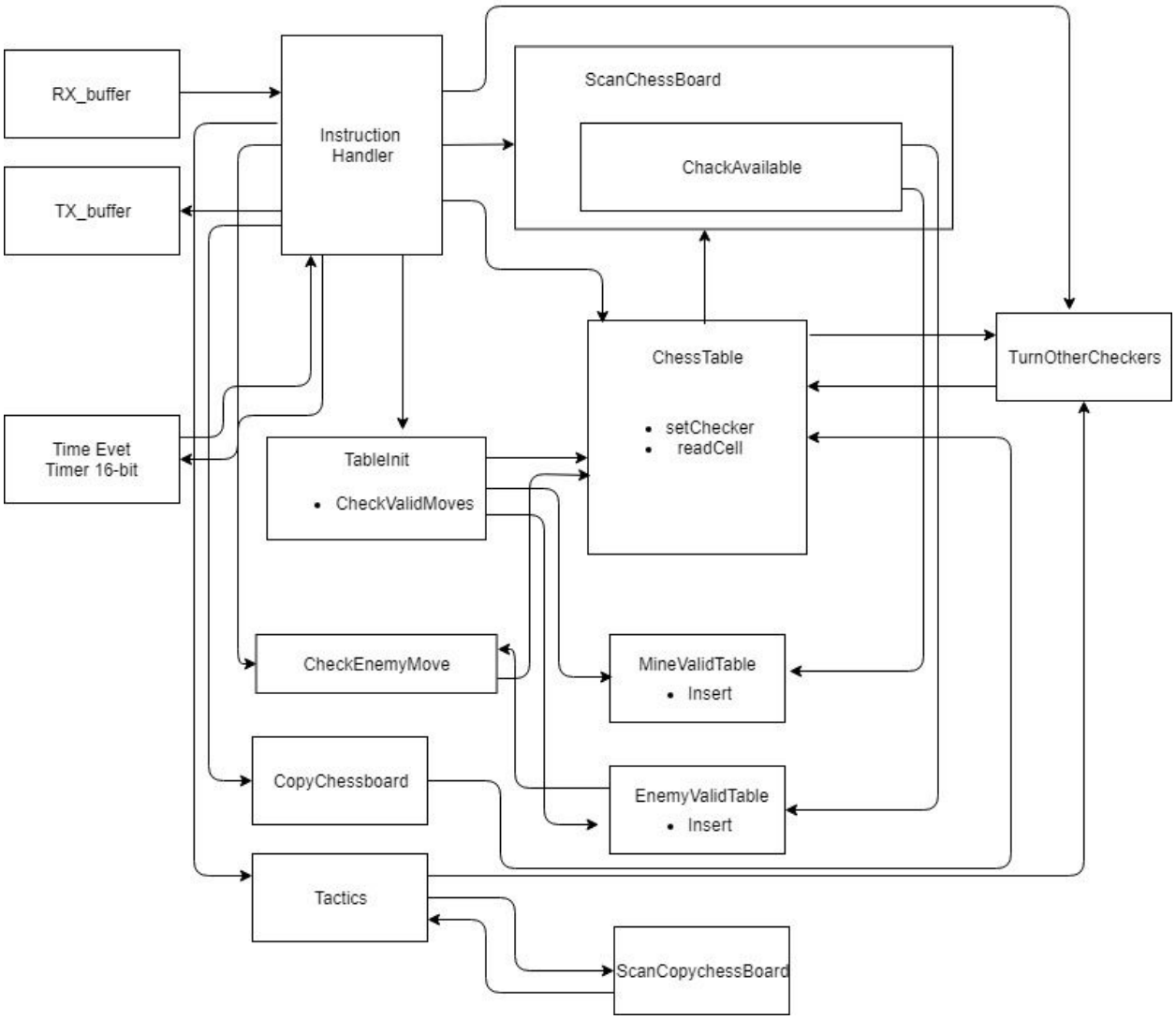


Image 2: FInal Block Diagram