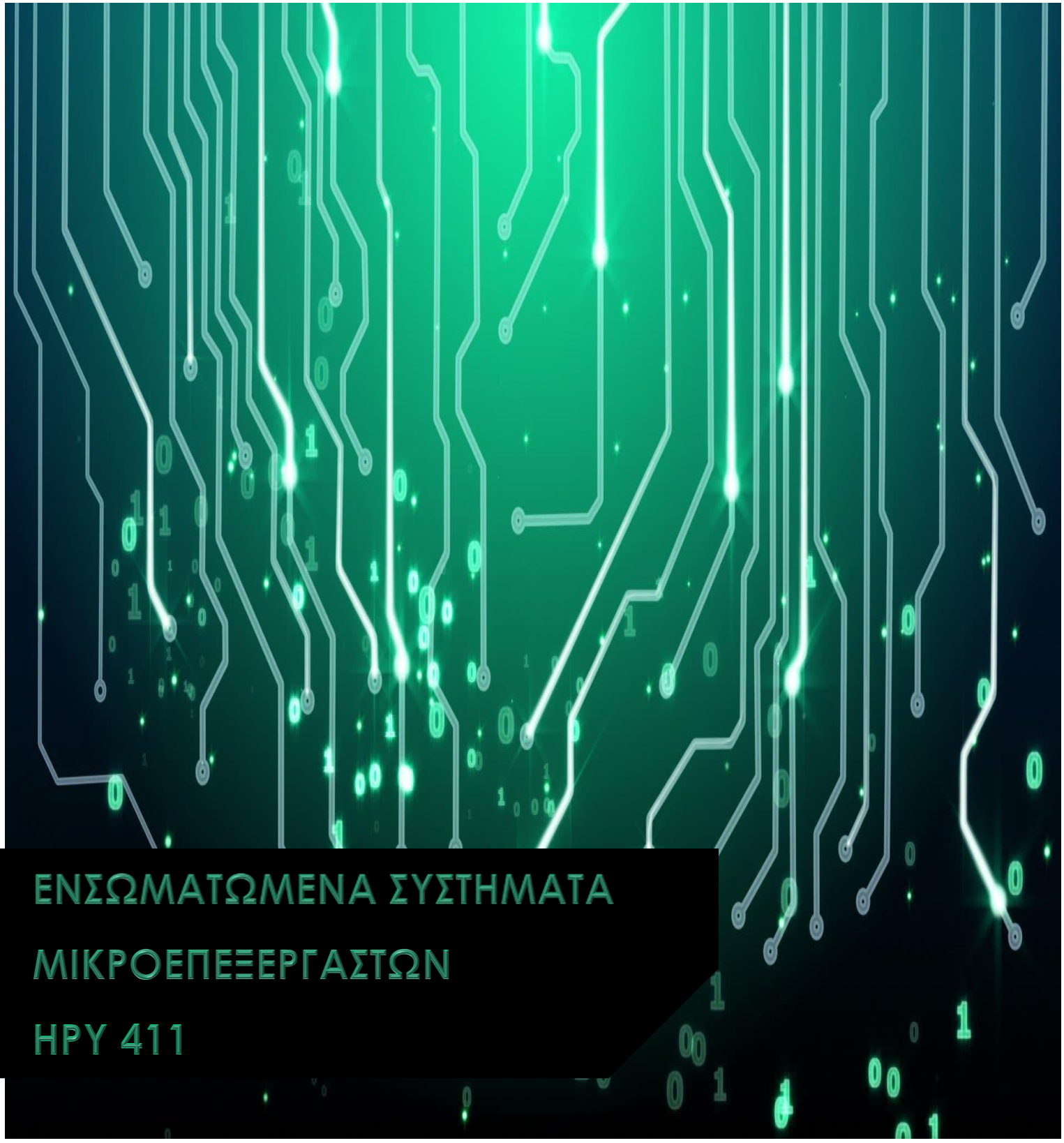

ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ ΗΡΥ 411

LAB41140566

Milestone 1: Othello Game Project

Φλέγγας Γεώργιος 2014030161

Χατζηπέτρος Αλέξανδρος 2013030151

8/4/2019

ΗΜΜΥ

Εισαγωγή:

Για το πρότζεκτ του μαθήματος Ενσωματωμένα Συστήματα Μικροεπεξεργαστών, μας ζητήθηκε η υλοποίηση ενός παιχνιδιού Othello με την χρήση του αναπτυξιακού STK500 το οποίο διαθέτει τον μικροελεγκτή ATMega16L. Το παιχνίδι θα διαθέτει τόσο την ικανότητα να παίξει ένας άνθρωπος εναντίον του AVR, καθώς και να αντιμετωπίσει κάποιο άλλο STK500.
Στόχος του πρώτου milestone αποτέλεσε, η υλοποίηση της διεπαφής σειριακής θύρας για υποστήριξη όλων των εντολών και απαντήσεων, του ανάμματος LED για τις απαραίτητες περιπτώσεις, υλοποίηση χρονομετρητή και η υλοποίηση του board.

Επιλογές και σχεδίαση:

Αφού μελετήσαμε την εκφώνηση και κατανοήσαμε κάποιες από τις βασικές ιδέες λειτουργίας του παιχνιδιού, αρχίσαμε την υλοποίηση του πρωτοκόλλου επικοινωνίας, φροντίζοντας να εφαρμόσουμε τις απαραίτητες ρυθμίσεις που ζητήθηκαν, ώστε να υπάρχει ενιαία πλατφόρμα (κρύσταλλος 10MHz, 9600baud, 8Bits, 1Stop Bit, No Parity).

Σε αυτό το κομμάτι, μεγάλο ρόλο έπαιξε η αξιοποίηση του κώδικα που γράψαμε για το **3ο εργαστήριο**. Χρησιμοποιήσαμε τις συναρτήσεις :

- `void UART_Init()` για την αρχικοποίηση της θύρας επικοινωνίας
- `void USART_Transmit( unsigned char data )` για την μετάδοση πληροφορίας
- `void USART_Transmit_Str(char data[])` για την μετάδοση string πληροφορίας
- `void AVR_Reciever(char Data[])` για την λήψη πληροφορίας από το pc

Ακόμα δημιουργήσαμε την νέα συνάρτηση `void AVR_Reciever(char Data[])`, η οποία θα επεξεργάζεται τα μηνύματα που δέχεται το STK500 από το pc χάρις στο USART_RXC_vect και θα συνεχίζει στις απαραίτητες ενέργειες (η υλοποίηση των περισσότερων ενεργειών θα γίνει στο milestone 2). Ο τρόπος με τον οποίο αναγνωρίζει κάθε εντολή, είναι να συγκρίνει κάθε χαρακτήρα που θα αποθηκεύει στο Data[] κατά την ανάγνωση, με τον ανάλογο κωδικό ASCII. Ιδιαίτερη προσοχή δόθηκε στην περίπτωση που γίνεται λήψη του OK<CR>. Για να μπορέσουμε να αντιμετωπίσουμε σωστά την εντολή αυτή, προσθέσαμε στον κώδικα μας το `char LastTrasmit[10]`, το οποίο αποθηκεύει την τελευταία εντολή που μετέδωσε το STK500. Παρακάτω ακολουθεί ένα χαρακτηριστικό κομμάτι κώδικα που αναπαριστά 2 από τις περιπτώσεις που το STK500 δέχεται την εντολή αυτή, καθώς και το πως ανιχνεύει ότι είναι αυτή η εντολή:

```
else if ((Data[reader] == 79)&&(Data[reader+1] == 75)&&(Data[reader+2] == 13))
// OK<CR>
{
  //Reply Case based on what avr transmitted last
  if ((LastTrasmit[0] == 77 )&&(LastTrasmit[1] == 80)&&(LastTrasmit[2] == 13))
  { //MP<CR>
    //My Pass, answer after avr declares that it has no move
  }
  else if((LastTrasmit[0] == 87 )&&(LastTrasmit[1] == 78)&&(LastTrasmit[2] == 13))
  { //WN<CR>
    //I Win, answer after avr declares that its the winner open led 1
    PORTB ^= (1<<PB1);
  }
```

Αφού ανιχνευθεί το OK<CR>, ελέγχει το περιεχόμενο του LastTrasmit[] και συγκινεί με τις πιθανές εντολές που μπορεί να μετέδωσε τελευταίο το STK500 και όταν βρει την σωστή περίπτωση προχωρά στην εκτέλεση των απαραίτητων ενεργειών. Στην προκείμενες περιπτώσεις έχουμε τις εντολές My Pass στην οποία δεν εκτελούμε κάποια ενέργεια και I Win στην οποία θα πρέπει να ανάψουμε το LED1. Αντίστοιχα για τις περιπτώσεις I Lose και Tie, ανάβουν τα LED2 και LED3 αντίστοιχα. Τα leds αυτά είναι συνδεδεμένα με τον PORTB και έχουν αρχικοποιηθεί ώστε να είναι σβηστά. Για να ελέγξουμε την ορθή λειτουργία του παραπάνω, περάσαμε στο LastTrasmit[] το WN<CR> και στείλαμε από το PC απλά την εντολή OK<CR> και άναψε το LED1.

Στην συνέχεια, προχωρήσαμε στην υλοποίηση του timer που θα κρατάει τον χρόνο μέσα στον οποίο πρέπει ο κάθε παίχτης να κάνει την κίνηση του. Για αυτό, αξιοποιήσαμε την μελέτη που κάναμε για τους timers στο 1ο εργαστήριο. Με βάση την χρήση του Overflow Interrupt TIMER0_OVF_vect και χρησιμοποιώντας prescaler 1024, προχωρήσαμε στην εφαρμογή της ακόλουθης φόρμουλας:

$$Ftimer = \frac{CPU\ \ freq}{Prescaler} = 10000000/1024 = 9765.625$$

$$Ttick = \frac{1}{Ftimer} = 0.0001024$$

$$Ttotal = 255 * Ttick = 0.026112$$

$$OVFCount = \frac{Time\ we\ want}{Ttotal}$$

Το παραπάνω για την default τιμή 2 sec, δίνει 76. Έτσι αρχικοποιήσαμε την μεταβλητή `volatile uint8_t t_limit` ως 76. Σε περίπτωση που θέλουμε να δώσουμε άλλη τιμή μέσα από την εντολή ST<SP>[1-9]<CR>, αφού ανιχνευθεί η περίπτωση από την AVR_Reciever και ελεγχθεί ότι η τιμή που δόθηκε είναι αποδεχθεί τότε την διαιρούμε με το 0.026112 και θέτουμε το κατάλληλο OVFCount στο t_limit. Ο timer θα μηδενίζεται κάθε φορά που ξεκινάει ο γύρος του κάθε παίχτη.

Εν τέλη, υλοποιήσαμε το Board μας με τις συναρτήσεις `void BoardInit(),void PrintBoard()`. Το board μας θα είναι ουσιαστικά ένας δισδιάστατος πίνακας 8x8, `volatile uint8_t board[8][8]`. Με την χρήση της BoardInit, όλες του οι θέσεις αρχικοποιούνται ως *<SP>(32)* και στην συνέχεια οι θέσεις [3,3] και [4,4] παίρνουν την τιμή *W(87)* συμβολίζοντας το White, ενώ οι [3,4] και [4,3] την τιμή *B(66)* για το Black. Με την χρήση της PrintBoard μετά την αρχικοποίηση θα έχουμε το ακόλουθο αποτέλεσμα: