ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ

Εργαστηριακή Άσκηση 4η

Έλεγχος οθόνης μέσω διεπαφής UART
σε AVR Assembly και C

Σωτήριος Μιχαήλ
2015030140

ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

Εισαγωγή

Σκοπός είναι η υλοποίηση ενός προγράμματος το οποίο επικοινωνεί μέσω της optional (communication) θύρας RS232 του STK500 με ένα τερματικό (λ.χ. PuTTY) σε PC, και υλοποιεί μερικές απλές εντολές, καθώς και μία οθόνη 7-segment, η οποία δείχνει αριθμούς τους οποίους έχουμε αποθηκεύσει μέσω μίας εκ των προαναφερθέντων εντολών. Η άσκηση αυτή αποτελεί συνέχεια της 3ης άσκησης, με την διαφορά πως τώρα η υλοποίηση της επικοινωνίας με τη θύρα RS232, καθώς και οι αρχικοποιήσεις όλων των καταχωρητών I/O γίνονται μέσω C.

Οι εντολές που υλοποιήθηκαν είναι οι εξής:

Εντολή από PC (ASCII)	**Επεξήγηση Ορίσματος X**	**Ενέργεια Εντολής**	**Απάντηση προς PC (ASCII)**
AT<CR><LF>	-	Απλή Απάντηση OK με το <LF>, καμμία άλλη ενέργεια	OK<CR><LF>
C<CR><LF>		Καθαρισμός οθόνης και απάντηση OK μετά το <LF>	OK<CR><LF>
N<X> <CR><LF>	<X>: 1-8 χαρακτήρες ASCII που όλοι αντιστοιχούν σε δεκαδικό ψηφίο	Καθαρισμός οθονης, αποθήκευση ορισμάτων στην μνημη της οθόνης, απάντηση OK μετά το <LF>	OK<CR><LF>

Μία ακόμη εντολή με μορφή “D”, η οποία επιστρέφει τα δεδομένα από τον buffer της RAM στο τερματικό του PC και απαντά OK, υλοποιήθηκε για λόγους αποσφαλμάτωσης.

Η βασική ροή του προγράμματος παρουσιάζεται παρακάτω, σε μορφή ψευδοκώδικα:

```
START
    INITIALIZE_STACK( )
    INITIALIZE_UART( )
    INITIALIZE_TIMER_0( )
    WHILE(1) {
        UPDATE_DISPLAY()
        IF (RECEIVE INTERRUPT) {
            RECEIVEINTERRUPTHANDLER()
            REPLY()
        }
    }
```

Παρακάτω παρουσιάζονται περισσότερες λεπτομέρειες για την υλοποίηση κάθε μέρους αυτού του προγράμματος.

Αρχικοποιήσεις

Οι αρχικοποιήσεις για υλοποίηση σε μικροελεγκτή βασίζονται στο ποιόν μικροελεγκτή χρησιμοποιούμε και στην συχνότητα του ρολογιού του. Σε αυτή τη περίπτωση, έχουμε τον μικροελεγκτή ATmega16L και χρησιμοποιούμε το εσωτερικό του ρολόι στο 1MHz. Επομένως, πρέπει να υπολογίσουμε ανάλογα κάποιες τιμές για την αρχικοποίηση της διεπαφής U(S)ART καθώς και του timer0, που χρησιμοποιούμε για την ενημέρωση της οθόνης.

Η επικοινωνία με το PC, καθώς και η ενημέρωση της οθόνης γίνονται μέσω interrupts, επομένως, πρέπει να αρχικοποιήσουμε και τους interrupt handlers στην αρχή του προγράμματος, στις σωστές διευθύνσεις, ανάλογα με το interrupt που έχουμε.

Οι αρχικοποιήσεις για τη 4η εργαστηριακή άσκηση έχουν υλοποιηθεί σε γλώσσα C, σύμφωνα με τον ψευδοκώδικα που παρουσιάστηκε παραπάνω.

Διευθύνσεις μνήμης

Στις προδιαγραφές του προγράμματος, ορίζεται η αποθήκευση των δεδομένων εισόδου τα οποία προορίζονται να εμφανιστούν στην οθόνη, δηλαδή ψηφία σε ASCII, στη μνήμη RAM του ελεγκτή. Έτσι ώστε να αποθηκευθούν τα δεδομένα στη μνήμη RAM, ο buffer στον οποίο αποθηκεύονται ορίζεται ως "volatile". Με αυτή την οδηγία, ο avr-gcc θα χρησιμοποιεί τη RAM για την αποθήκευση του buffer και όχι registers, όπως θα γινόταν με μία απλή μεταβλητή της C.

Για την εμφάνιση του ψηφίου που αποθηκεύεται σε μορφή κωδικού ASCII, πρέπει να γίνει μία μετατροπή σε BCD, το οποίο γίνεται με μία μάσκα στα 4 λιγότερο σημαντικά bits, και ένα lookup table, το οποίο περιέχει τους κωδικούς για την εμφάνιση των ψηφίων στην οθόνη 7-segment. Η διαδικασία της μετατροπής με τη μάσκα των 4 bit γίνεται μέσω assembly.

Για το πρόγραμμα

Καθώς αυτή η έκδοση του προγράμματος είναι σε C, η στοίβα δεν χρειάζεται να αρχικοποιηθεί, καθώς αυτό γίνεται αυτόματα από τον μεταγλωττιστή. Για να χρησιμοποιηθεί ο κώδικας Assembly παράλληλα με τον κώδικα C, οι ενέργειες που θέλουμε να υλοποιηθούν σε Assembly χωρίζονται σε συναρτήσεις, οι οποίες βρίσκονται σε ένα ξεχωριστό αρχείο κώδικα Assembly (serial_display_asm.S). Οι συναρτήσεις αυτές είναι δηλωμένες ως εξωτερικές στο κώδικα C, και ο μεταγλωττιστής συνδυάζει τα δύο αρχεία κατά την μεταγλώττιση. Τα ορίσματα που στέλνει ο κώδικας C βρίσκονται στους καταχωρητές με ζυγό αριθμό, ξεκινώντας από τον R24 και πηγαίνοντας προς τις υψηλότερες θέσεις μνήμης (όρισμα 1 = R24, όρισμα 2 = R22 κ.ο.κ.). Η τιμή που επιστρέφει μία συνάρτηση σε Assembly αποθηκεύεται στον καταχωρητή R25, ο οποίος πρέπει να εκκενώνεται πριν από την εντολή επιστροφής του κώδικα Assembly. Οι καταχωρητές αυτοί είναι "caller-saved", δηλαδή χρησιμοποιούνται ελεύθερα από τον κώδικα Assembly, και αν χρειάζεται, πρέπει να αποθηκεύονται και να επαναφέρονται από τον κώδικα που καλεί τις συναρτήσεις που τους χρησιμοποιούν, σε αυτή τη περίπτωση, τον κώδικα C.

Για τα interrupts

Για την ορθή λειτουργία της οθόνης, χρησιμοποιούμε τον timer 0 και το interrupt υπερχείλισης του. Ο ελεγκτής για αυτό το interrupt επομένως βρίσκεται στη διεύθυνση 0x12, και για τον κώδικα C χρησιμοποιείται το macro "TIMER0_OVF_vect"

Για την επικοινωνία UART μέσω της θύρας RS232, θέλουμε να έχουμε ένα interrupt κάθε φορά που έχουμε μία ολοκληρωμένη μετάδοση στον buffer του UART, τον UDR. Επομένως, θέτουμε τα bits RXC, TXC του καταχωρητή UCSRA σε 1, και τα RXEN, TXEN και RXCIE του καταχωρητή UCSRB σε 1. Το RXCIE είναι το bit το οποίο ενεργοποιεί το interrupt για μία ολοκληρωμένη μετάδοση, και απαιτεί να είναι και το RXC bit ενεργό. Ο ελεγκτής του RXCIE interrupt βρίσκεται στη διεύθυνση 0x16 και για τον κώδικα C χρησιμοποιείται το macro "USART_RXC_vect".

Για την διεπαφή U(S)ART

Σε αυτή την εργαστηριακή άσκηση, χρησιμοποιούμε το USART ως απλό UART, δηλαδή, έχουμε ασύγχρονη επικοινωνία με το PC. Ο στόχος για την ταχύτητα επικοινωνίας με το PC είναι τα 9600 baud, αλλά έπειτα από πειραματισμό με στην πλατφόρμα STK500, η ταχύτητα επικοινωνίας για την οποία η επικοινωνία με τη θύρα RS232 ήταν επιτυχής, ήταν 2400 baud, λόγω του υψηλού ποσοστού σφάλματος επικοινωνίας για τη χαμηλή ταχύτητα ρολογιού (1MHz) που έχουμε. Επομένως, πρέπει να φορτώσουμε τη κατάλληλη τιμή στον καταχωρητή UBBRL. Για να υπολογίσουμε τη τιμή αυτή, χρησιμοποιείται ο παρακάτω τύπος:

$$UBBRL = \frac{f_{oscillator}}{16\,x\,target_{baud}} = \frac{1\,MHz}{16\,x\,2400} = 26.042 \approx 26 = 0\,x\,1\,A\,(hex)$$

Για την σωστή επικοινωνία με το PC μέσω του RS232, πρέπει να ορίσουμε και τη διαμόρφωση των μηνυμάτων που μεταφέρονται. Αυτό γίνεται μέσω των bits UCSZ2..0 του καταχωρητή UCSRC. Καθώς ο καταχωρητής UCSRC μοιράζεται τη διεύθυνσή του με έναν άλλον καταχωρητή για εξοικονόμηση διευθύνσεων, χρειάζεται να θέσουμε και το URSEL bit σε 1, έτσι ώστε να γράψουμε τις σωστές τιμές για τα UCSZ2..0. Σε αυτήν την άσκηση, τα μηνύματα έχουν 1 stop bit και δεν έχουν parity. Επομένως, γράφουμε 1 στα bits UCSZ1, UCSZ0 και URSEL.

Για τον timer 0

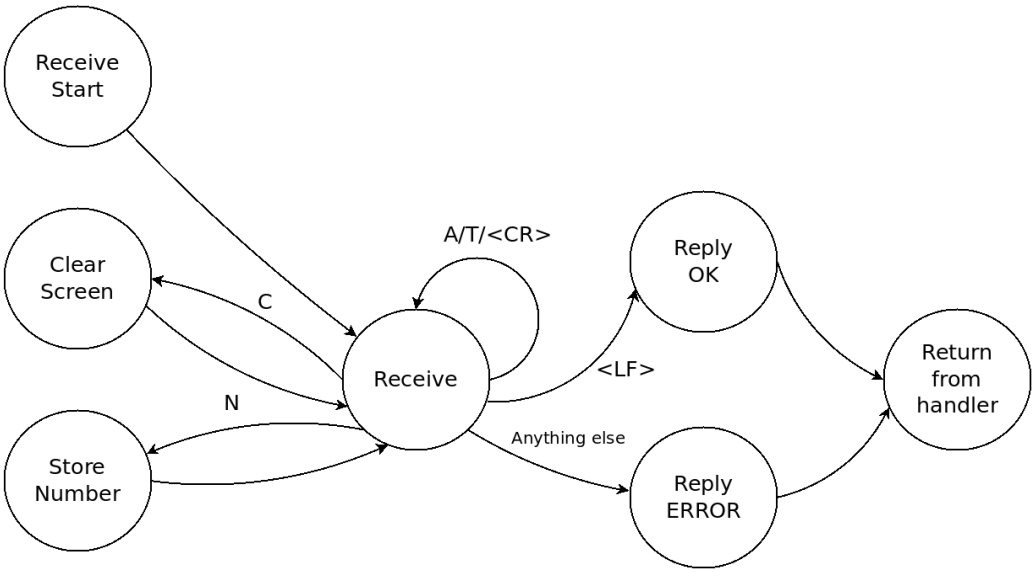
Μέσω του μετρητή αυτού, θέλουμε να ανανεώνουμε την οθόνη κάθε 4 ms. Καθώς έχουμε ρολόι 1 MHz αντί των 10 MHz, έπρεπε να γίνει μία αλλαγή στην αρχικοποίηση του timer, σε σχέση με τις 2 προηγούμενες εργαστηριακές ασκήσεις. Χρησιμοποιήθηκε ο prescaler $f_{clk}/1024$, θέτοντας 1 τα bit CS00 και CS02, και η σωστή τιμή του TCCR0 για overflow κάθε 4 ms προκύπτει από τον τύπο:

$$f_{timer0} = f_{clk}/1024 \approx 1\,kHz \Rightarrow T_{timer\,0} \approx 1\,ms \Rightarrow 1\,ms\,x\,(256 - TCCR0) = 4\,ms \Rightarrow TCCR0 = 4$$

Ενώ η αρχικοποίηση του timer 0 υλοποιείται με C, η διαχείριση του κυκλικού μετρητή για την ενεργοποίηση της μίας εκ των οκτώ οθονών για κάθε τιμή που αποθηκεύεται γίνεται μέσω του κώδικα Assembly.

Διαχείριση RXCIE interrupt και είσοδος μέσω RS232

Ο RXCIE interrupt handler υλοποιεί την είσοδο δεδομένων από το PC μέσω τις θύρας RS232, και επομένως, όλες τις εντολές των προδιαγραφών, μέσω κλήσεων κατάλληλων βοηθητικών συναρτήσεων. Ο διαχειριστής αυτού το interrupt είναι ουσιαστικά μία μηχανή πεπερασμένων καταστάσεων, η οποία αλλάζει κατάσταση ανάλογα τον χαρακτήρα που λαμβάνεται από το UDR. Η είσοδος του πρώτου χαρακτήρα κάθε εντολής ενεργοποιεί το interrupt και επομένως τον handler, και για την είσοδο των επόμενων χαρακτήρων κάθε εντολής γίνεται μέσω polling του καταχωρητή UDR. Η βασική ροή του RXCIE interrupt handler παρουσιάζεται με το παρακάτω διάγραμμα καταστάσεων:



Για τον καθαρισμό της οθόνης μέσω της εντολής “C<CR><LF>”, μία βοηθητική συνάρτηση θέτει την οθόνη σε 0xFF, και καθαρίζει την μνήμη στην οποία βρίσκονται τα ψηφία εισόδου αποθηκευμένα.

Έξοδος μέσω RS232

Οι δύο συναρτήσεις απάντησης προς το PC, όπου η REPLY απαντά “OK<CR><LF>” για μία επιτυχή αναγνώριση και εκτέλεση εντολής, και η ERREPLY απαντά “ERROR<CR><LF>” για μία αποτυχία αναγνώρισης εντολής, υλοποιούνται με την uart_send(), μία συνάρτηση η οποία παίρνει ως όρισμα έναν χαρακτήρα, και τον αποστέλλει στο UDR. Η συνάρτηση uart_send() παρουσιάζεται ως αυτούσιο κομμάτι κώδικα:

```
void uart_send(unsigned char ch)
{
      while (! (UCSRA & (1 << UDRE)));
      UDR = ch;
}
```

Οθόνη 7-segment

Η υλοποίηση της οθόνης, καθώς και του timer που χρησιμοποιείται για την ανανέωσή της, έχει καλυφθεί εκτενώς στις αναφορές των προηγούμενων εργαστηρίων. Η διαφορά σε αυτή την υλοποίηση είναι πως ο κώδικας που ελέγχει τα PORT A και C που ελέγχουν την οθόνη, χωρίζονται σε κομμάτια κώδικα C και Assembly.

Καταχωρητές και μεταβλητές

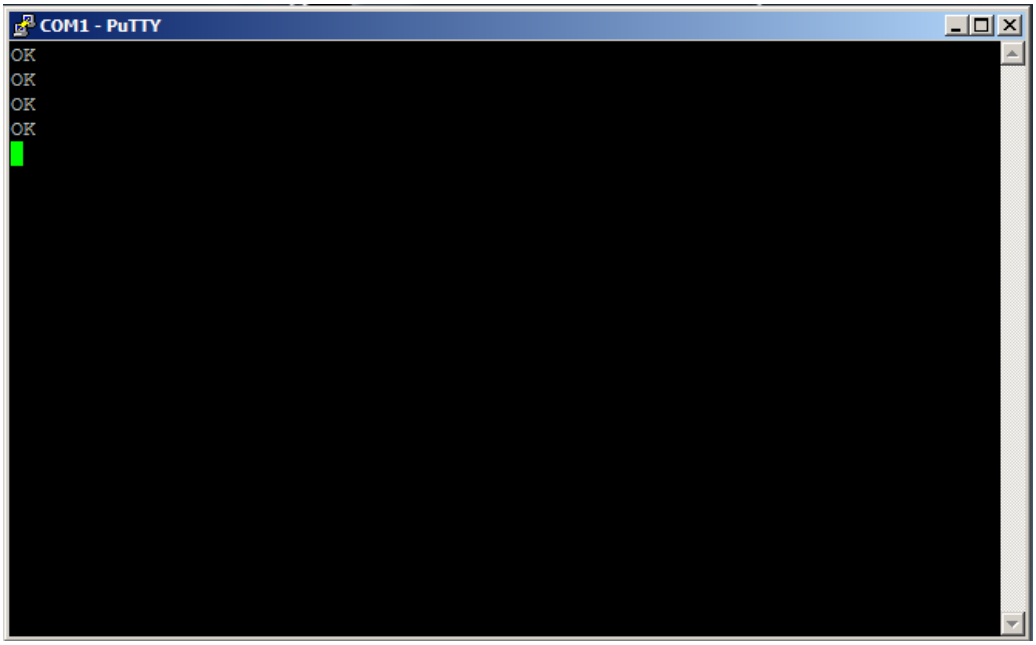
Για την ευκολότερη κατανόηση του κώδικα, παρατίθεται ένας πίνακας με τους καταχωρητές που χρησιμοποιούνται στο πρόγραμμα, και την ειδική χρήση του καθενός, εάν υπάρχει:

Καταχωρητής	Χρήση
R16	Γενικής προσωρινής χρήσης
R17	Γενικής προσωρινής χρήσης
R18	Γενικής προσωρινής χρήσης
R19	Γενικής προσωρινής χρήσης
R20	Γενικής προσωρινής χρήσης
R21	Γενικής προσωρινής χρήσης
R22	Όρισμα από κώδικα C
R23	Γενικής προσωρινής χρήσης
R24	Όρισμα από κώδικα C
R25	Τιμή επιστροφής συνάρτησης

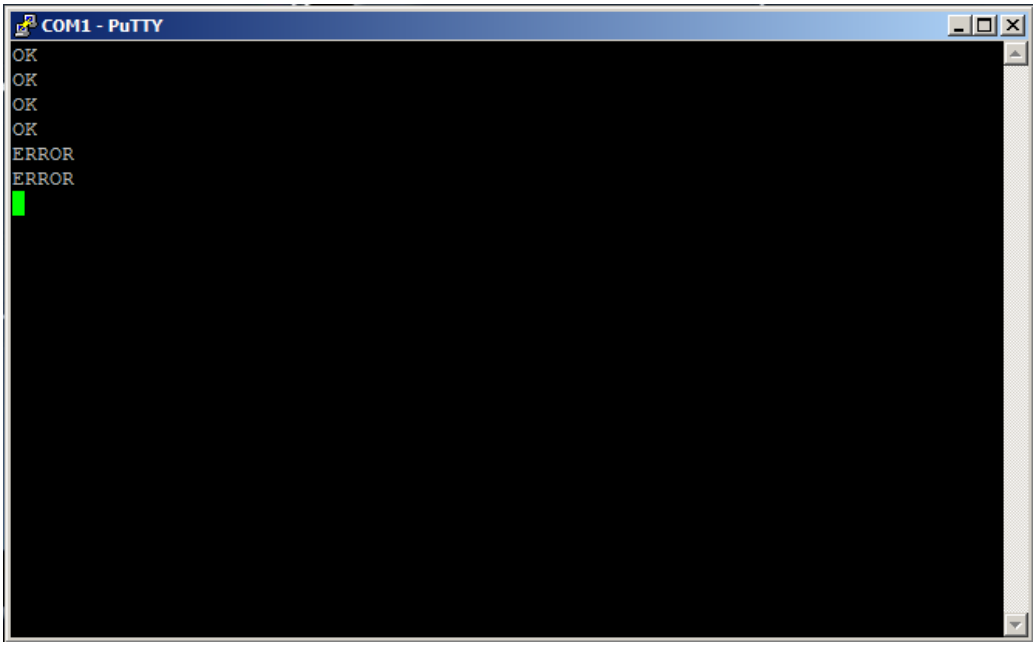
Στον κώδικα C, οι μεταβλητές έχουν όλες δηλωθεί ως χαρακτήρες χωρίς πρόσημο, “unsigned char”. Αυτό γίνεται διότι αυτός ο τύπος μεταβλητής χρησιμοποιεί το λιγότερο δυνατό χώρο στον μικροελεγκτή, και κάνει την υλοποίηση όσο πιο μικρή γίνεται. Το ίδιο ισχύει και τη χρήση του “for(;;) { }” στη θέση του “while(1)” για την υλοποίηση του ατέρμονος βρόχου που διατηρεί τη λειτουργία του μικροελεγκτή επ’αορίστον. Οι επαναλήψεις με “for” χωρίς μεταβλητές χρησιμοποιούν λιγότερο χώρο στον AVR. Έτσι, μπορούν να χρησιμοποιηθούν ελεγκτές πολύ μικρότεροι του ATmega16L, καθώς η άσκηση αυτή χρησιμοποιεί μόνο το 3.2% της μνήμης δεδομένων, και 5.5% της μνήμης προγράμματος.

Διεξαγωγή εργαστηρίου – προσομοίωση και επιβεβαίωση λειτουργίας

Στην προηγούμενη εργαστηριακή άσκηση εμφανίστηκε ένας περιορισμός της προσομοίωσης, καθώς δεν ήταν δυνατό να προσομοιώσουμε δεδομένα τα οποία εισέρχονται από τη θύρα RS232. Για αυτή την εργαστηριακή άσκηση όμως, η πλατφόρμα STK500 χρησιμοποιήθηκε για την επιβεβαίωση λειτουργίας της υλοποίησης σε πραγματικό μικροελεγκτή. Η θύρα RS232 SPARE χρησιμοποιήθηκε για την αποστολή των εντολών στον μικροελεγκτή. Η θύρα αυτή ενεργοποιείται με την σύνδεση των RXC-TXC pin στα PD0 και PD1 του PORT D. Η επικοινωνία του PC με τον AVR γίνεται μέσω του PuTTY, στα 2400 baud. Παρακάτω φαίνεται πως ο AVR αναγνωρίζει τις εντολές και απαντά OK σε κάθε μία από αυτές (Η εντολή "D" απαντά μόνο "OK" καθώς δεν είχε φορτωθεί κάποιος αριθμός):



Με την είσοδο κάποιας μη αναγνωρίσιμης εντολής, ή την είσοδο παραπάνω από 8 ψηφίων για την εντολή N<X>, το πρόγραμμα επιστρέφει "ERROR", όπως φαίνεται παρακάτω, οπού δοκιμάστηκαν και οι δύο προαναφερθείσες περιπτώσεις:



Καθώς δεν υπάρχει διαθέσιμη οθόνη 7-segment LED με 8 ψηφία, η έξοδος οδηγήθηκε στα LEDs του STK500. Ο ρυθμός ανανέωσης της οθόνης όμως είναι πολύ υψηλός, στο σημείο που, με τα LEDs του STK500, ο αριθμός που εμφανίζεται, εμφανίζεται μόνο για κλάσματα του δευτερολέπτου, ακόμη και αρκετά χαμηλό ρυθμό ανανέωσης από τον timer0. Έτσι δεν ήταν δυνατό να υπάρξει μία φωτογραφία για την επιβεβαίωση της εξόδου.