Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

LAB 3

Καλογεράκης Στέφανος | Ζερβάκης Αρης

Σκοπός Εργαστηρίου

Σκοπός του εργαστηρίου είναι σχεδίαση ενός πιο ώριμου συστήματος, που να περιλαμβανει εξωτερικά interrupts, σειριακή θύρα, απλή διεπαφή με το περιβάλλον με διακόπτες (switches) και LED, χρήση εσωτερικής μνήμης SRAM και απλή επεξεργασία και εκτέλεση μερικών εντολών.

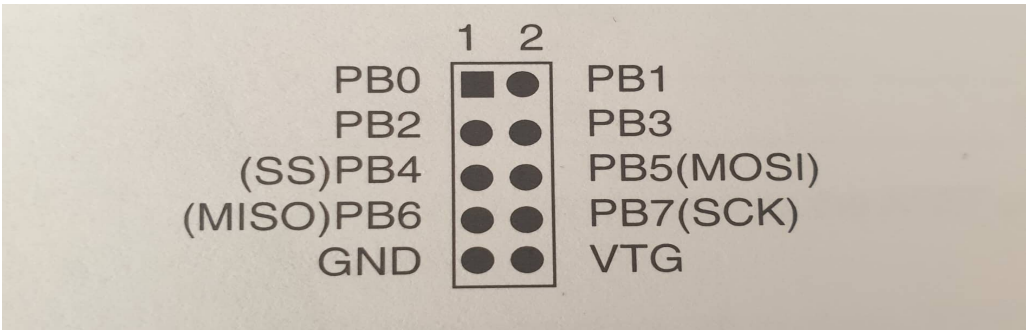
1ο Ερώτημα

Στο πρώτο ερώτημα ζητήθηκε η υλοποίηση ενός μετρητή δακτυλίου 4Bit, χρησιμοποιώντας 4 LED του STK500. Ο μετρητής κατασκευάστηκε αξιοποιώντας τα external interrupts INT0 και INT1, έτσι με το πάτημα του διακόπτη (SW1) το επόμενο LED θα ανάψει ενώ αυτό που ήταν ήδη ανοιχτό θα σβήσει. Οι διακόπτες δεν είναι debounced by default, οπότε δημιουργήθηκε κώδικας για τον λόγο αυτό. Debounce έγινε θέτοντας '0' στο κατάλληλο πεδίο του καταχωρητή GICR, δηλαδή το external interrupt enable, προτού ενεργοποιηθεί ξανά μετα την πάροδο ενός χρονικού διαστήματος που ορίσαμε (500 msec).

Αφού συμβουλευτήκαμε το datasheet του STK500, παρατηρήσαμε πως τα pins PB4 - PB7 στο 40-Pin AVR μπορούν να χρησιμοποιηθούν για την εξωτερική μνήμη, οπότε η επιλογή της εξόδου για τα 4 LED είναι PB0 -PB3.

Επίσης, η παραπάνω επιλογή μας, δεν έρχεται σε σύγκρουση με τους ακροδέκτες των interrupts, ούτε με τους ακροδέκτες της σειριακής θύρας καθώς τα interrupts βρίσκονται στο pin PD3(INT1) και η σειριακή αντιστοιχεί στα pins PD0 και PD1 για RXD, TXD.

Ακολουθεί το datasheet απο το manual του STK500.



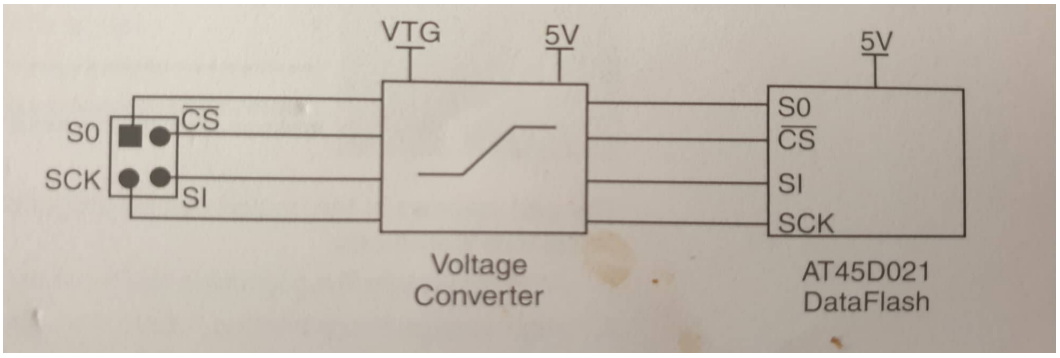
2ο Ερώτημα

Στο δεύτερο ερώτημα μας ζητήθηκε να γίνει πρόσβαση στην εσωτερική SRAM για διάβασμα και γράψιμο. Υλοποιήθηκε κατασκευάζοντας ένα μικρό τμήμα κώδικα το οποίο αναλαμβάνει να αποθηκεύσει μερικούς χαρακτήρες στις πρώτες θέσεις της SRAM στις οποίες μπορούμε να αποθηκεύσουμε δεδομένα στη διεύθυνση 0x0060, καθώς προηγούμενες θέσεις είναι δεσμευμένες απο τον AVR.

```
volatile char *a = (char *)malloc(sizeof(char)*10);
a=0x0060;

a[0]= 'T';
a[1] ='e';
a[2] ='s';
a[3] ='t';
```

Στο εγχειρίδιο του STK500 βλέπουμε τη συνδεσμολογία των pins S0,CS',SCK,SI με τα αντίστοιχα του dataflash chip που είναι συμβατό με το STK500.



3ο Ερώτημα

Στο τελευταίο ερώτημα μας ζητήθηκε να δημιουργήσουμε ενα πρωτόκολλο εντολών με χρήση της σειριακής θύρας και τα Interrupt αυτής.

Οι εντολές που υποστηρίζονται είναι :

1. AT<CR> : Απλό πρωτόκολλο XON/XOFF
2. MW<SP> X<SP>Y<CR> : Αποθήκευση του αριθμού Y στην μνήμη σε διεύθυνση που καθορίζεται από τον αριθμό X.
3. MR<SP>X<CR>: Ανάγνωση από την διεύθυνση μνήμης που καθορίζει το όρισμα X
4. SUM<SP>X<SP>Y<CR>: Άθροισμα αριθμών που έχουν αποθηκευτεί στην μνήμη

Αν σε οποιαδήποτε εντολή, κατα την σύνταξη της, ο χρήστης κάνει κάποιο σφάλμα, τότε εκτυπώνεται κατάλληλο μήνυμα λάθους.

Ξεκινώντας την υλοποίηση, αρχικά, εντοπίζουμε το <CR> στο σημείο που δεχόμαστε τα interrupts. Αφού γίνει ο παραπάνω εντοπισμός, ανακατευθυνόμαστε στην συνάρτηση που είναι υπεύθυνη για την ανάλυση των φράσεων που εισαγάγαμε πριν το <CR>. Η ανάλυση ξεκινάει μία θέση μετά, και η σύγκριση εκτείνεται έως το νεό <CR> που συναντάμε.

Οποιαδήποτε σύγκριση βρεί format η κωδικό γράμμα διαφορετικό από τα πρότυπα, η διαδικασία θα διακοπεί με κατάλληλο μήνυμα λάθους και γίνεται ενημέρωση του pointer στο buffer στην επόμενη θέση απο το τελευταίο τελευταίου CR (CR+1).

Πιο συγκεκριμένα, διαβάζουμε byte-byte τον buffer μας, συγκρίνοντας αρχικά τα 2 ή 3 πρώτα γράμματα για να καταλήξουμε σε ποια εντολή βρισκόμαστε ώστε να μειώσουμε όσο μπορούμε τον αριθμό των συγκρίσεων. Αφού φτάσουμε στο σημείο με τις ακέραιες παραμέτρους, διαβάζουμε κάθε αριθμό ξεχωριστά, ο οποίος στην συνέχεια θα μετατραπεί σε uint8_t.

Έτσι πλέον μπορούμε να αποθηκεύσουμε και να διαβάσουμε χρησιμοποιώντας το ερώτημα 2. Η αποστολή των ακεραίων στο terminal γίνεται με τον αντίστροφο τρόπο από αυτόν που χρησιμοποιήθηκε για το διάβασμα και το concatenation (καθώς χρησιμοποιούμε τα div και mod για το σπάσιμο του αριθμού σε μονοψήφια) και τον μετατρέψουμε σε char ώστε να σταλεί στο terminal.