Ενσωματωμένα Συστήματα Μικροεπεξεργαστών – ΗΡΥ411

Αναφορά Εργαστηρίου 5 – Κώδικας C στον Atmel AVR

Ομάδα: LAB41145877
Κιούλος Ευάγγελος 2016030056

Εισαγωγή:

Σκοπός της πέμπτης εργαστηριακής άσκησης ήταν η εξοικείωση με την γλώσσα C στον Atmel AVR. Σε αυτό το εργαστήριο υλοποιήθηκαν σε γλώσσα C και οι ρουτίνες εξυπηρέτησης των interrupt για την σειριακή θήρα και την οθόνη. Η αρχικοποίηση του προγράμματος και το κυρίως πρόγραμμα χρησιμοποιήθηκαν χωρίς αλλαγές από το προηγούμενο εργαστήριο.

Υλοποίηση Κώδικα:

Αρχικά, προστέθηκε η συνάρτηση “void init_7_seg(void)” η οποία αποθηκεύει στις θέσεις “0x0070 – 0x007A” της μνήμης RAM, τα ψηφία 0 έως 9 και 0x0A που χρειάζονται για την αποκωδικοποίηση από BCD σε seven segment. Η παραπάνω συνάρτηση καλείται στην αρχή του προγράμματος. Επιπλέων, προστέθηκε η συνάρτηση “void shift_RAM(void)”, η οποία μεταφέρει τα στοιχεία που είναι αποθηκευμένα στις θέσεις “0x0060 – 0x0067” της μνήμης RAM μία θέση προς τα δεξιά.

Στη ρουτίνα εξυπηρέτησης του interrupt TIMER0 overflow, που είναι υπεύθυνη για την λειτουργία της οθόνης, αρχικά δηλώνουμε τους παρακάτω pointers.

```
// points at RAM START
unsigned char *ram_pos = (unsigned char *)0x60;
// points at address 0x68, used for storing which digit is shown
unsigned char *digitShown = (unsigned char*) 0x68;
// points at address 0x69, used for storing the value of the ring counter
unsigned char *ring_counter = (unsigned char*) 0x69;
// points at address 0x70, the first address of the stored seven segment digits
unsigned char *seven_seg = (unsigned char*) 0x70;
```

Ο pointer “ram_pos” δείχνει στην διεύθυνση 0x60 και χρησιμοποιείται για να διαβάσουμε το στοιχείο που θέλουμε να προβάλουμε στην οθόνη. Ο pointer “digitShown” δείχνει στην θέση “0x68”, περιέχει μία τιμή από το 0 μέχρι το 7 και χρησιμοποιείται για να δείξουμε ποιο ψηφίο προβάλλεται στην οθόνη. Για παράδειγμα, αν η τιμή του “digitShown” είναι “0x02”, τότε στην οθόνη προβάλλεται το ψηφίο που είναι αποθηκευμένο στην διεύθυνση “ram_pos + *digitShown = 0x0060 + 0x02 = 0x0062”. Ο pointer “ring_counter” δείχνει στην διεύθυνση 0x0069, στην οποία αποθηκεύουμε την τιμή του ring counter. Ο pointer “seven_seg” δείχνει στην πρώτη διεύθυνση στην οποία είναι αποθηκευμένα τα seven segment ψηφία.

Στην συνέχεια, προσθέτοντας στην διεύθυνση που δείχνει ο pointer “ram_pos” την τιμή που έχει ο pointer “digitShown”, διαβάζουμε από την μνήμη το ψηφίο που θέλουμε να προβάλλουμε στην οθόνη και αποθηκεύουμε αυτή την τιμή στην μεταβλητή “data” όπως φαίνεται και παρακάτω.

```
// used for reading data we want to display
unsigned char data = ram_pos[(unsigned char)*digitShown];
```

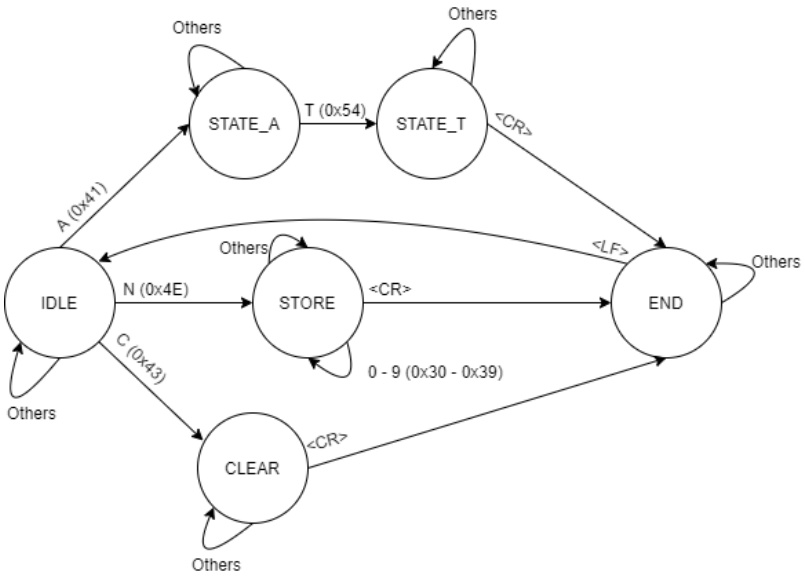
Στην συνέχεια ολισθαίνουμε την τιμή του pointer “ring_counter”, την αποθηκεύουμε στην μνήμη και ελέγχουμε αν η τιμή του είναι “0x00”, εάν είναι τότε την θέτουμε στην τιμή “0x01”.

```
*ring_counter = *ring_counter << 1; // shift ring counter and store new value

// if value of ring counter is 0x00, reset it to 0x01
if(*ring_counter == 0x00){
    *ring_counter = 0b00000001;
}
```

Έπειτα, αυξάνουμε την τιμή του pointer “digitShown” και ελέγχουμε αν έχει υπερβεί την τιμή ‘7’. Εάν ναι, τότε θέτουμε την τιμή του στην τιμή “0x00”. Τέλος, για την αποκωδικοποίηση από BCD σε seven segment, τα ψηφία seven segment είναι αποθηκευμένα στην μνήμη με την σειρά από 0x00 έως 0x0A στις θέσεις 0x0070 με 0x007A αντίστοιχα. Επομένως για να εμφανίσουμε το ψηφίο seven segment που θέλουμε, προσθέτουμε την τιμή BCD που διαβάστηκε από την μνήμη στην διεύθυνση 0x0070, που είναι η διεύθυνση στην οποία είναι αποθηκευμένο το πρώτο ψηφίο seven segment και εμφανίζουμε στην οθόνη την τιμή που υπάρχει στην διεύθυνση που προέκυψε. Για παράδειγμα, εάν διαβάσουμε από την μνήμη την BCD τιμή 0x05, τότε στην οθόνη θα εμφανίσουμε την τιμή που υπάρχει στην διεύθυνση “0x0070 + 0x05 => 0x0075”, δηλαδή στην περίπτωση μας την τιμή “0b10010010” που θα εμφανίσει το ψηφίο “5” στην οθόνη. Μετά την αποκωδικοποίηση, εμφανίσουμε στην οθόνη το ζητούμενο ψηφίο.

Στη ρουτίνα εξυπηρέτησης του interrupt USART RXC, δεν χρησιμοποιήθηκε η υλοποίηση των προηγούμενων εργαστηρίων αλλά δημιουργήθηκε η μηχανή πεπερασμένων καταστάσεων που φαίνεται παρακάτω.



Η κατάσταση στην οποία βρισκόμαστε αποθηκεύεται στην διεύθυνση 0x006A της μνήμης RAM. Στην κατάσταση “IDLE”, με είσοδο “A” δεν έχουμε έξοδο και μεταβαίνουμε στην κατάσταση “A”, με είσοδο “N” καθαρίζουμε την οθόνη και μεταβαίνουμε στην κατάσταση “STORE”, με είσοδο “C” καθαρίζουμε την οθόνη και μεταβαίνουμε στην κατάσταση “CLEAR”. Για οποιαδήποτε άλλη είσοδο μένουμε στην ίδια κατάσταση. Στην κατάσταση “A”, με είσοδο “T” δεν έχουμε έξοδο και μεταβαίνουμε στην κατάσταση “T” ενώ για οποιαδήποτε άλλη είσοδο μένουμε στην ίδια κατάσταση. Στην κατάσταση “T”, με είσοδο “<CR>” δεν έχουμε

έξοδο και μεταβαίνουμε στην κατάσταση “END” ενώ για οποιαδήποτε άλλη είσοδο μένουμε στην ίδια κατάσταση. Στην κατάσταση “CLEAR”, με είσοδο “<CR>” δεν έχουμε έξοδο και μεταβαίνουμε στην κατάσταση “END” ενώ για οποιαδήποτε άλλη είσοδο μένουμε στην ίδια κατάσταση. Στην κατάσταση “STORE”, με είσοδο “<CR>” δεν έχουμε έξοδο και μεταβαίνουμε στην κατάσταση “END”, με είσοδο κάποιο χαρακτήρα με τιμή ASCII από “0x30” έως “0x39”(που αντιστοιχεί στα ψηφία 0 έως 9), ολισθαίνουμε κατά ένα byte τις τιμές που είναι αποθηκευμένες στις διευθύνσεις 0x0060 έως 0x0067 και αποθηκεύουμε στην διεύθυνση 0x0060 τα τέσσερα τελευταία bits του χαρακτήρα που διαβάσαμε και μένουμε στην κατάσταση “STORE”. Για οποιαδήποτε άλλη είσοδο μένουμε στην ίδια κατάσταση. Στην κατάσταση “END”, για είσοδο “<LF>” μεταδίδουμε τους χαρακτήρες “OK<CR><LF>” και μεταβαίνουμε στην κατάσταση “IDLE”. Για οποιαδήποτε άλλη είσοδο μένουμε στην ίδια κατάσταση.

Στην αρχή της εκτέλεσης της ρουτίνας, διαβάζουμε από την μνήμη την κατάσταση στην οποία βρισκόμαστε και στο τέλος της ρουτίνας αποθηκεύουμε στην μνήμη την επόμενη κατάσταση.

Χάρτης Μνήμης:

Παρακάτω, βλέπουμε τις θέσεις μνήμης RAM που χρησιμοποιούνται από το πρόγραμμα.

0x0060	0x0071
0x0061	0x0072
0x0062	0x0073
0x0063	0x0074
0x0064	0x0075
0x0065	0x0076
0x0066	0x0077
0x0067	0x0078
0x0068	0x0079
0x0069	0x007A
0x006A	. . .
. . .	. . .
0x0070	0x045F

Στις διευθύνσεις με “ανοιχτό μπλε” χρώμα αντιστοιχούν οι θέσεις που χρησιμοποιούνται για την αποθήκευση των χαρακτήρων που προβάλλονται στην οθόνη. Στην διεύθυνση 0x0068 με “πορτοκαλί” χρώμα, αντιστοιχεί η διεύθυνση στην οποία αποθηκεύεται το ψηφίο που προβάλουμε. Στην διεύθυνση 0x0069 με “πορτοκαλί” χρώμα, αντιστοιχεί η διεύθυνση στην οποία αποθηκεύεται η τιμή του ring counter. Στην διεύθυνση 0x006A με “πορτοκαλί” χρώμα, αντιστοιχεί η διεύθυνση στην οποία αποθηκεύεται η τωρινή κατάσταση. Η διεύθυνση με “γκρι” χρώμα αντιστοιχεί στην θέση του stack.

Αποτελέσματα προσομοίωσης:

Για την προσομοίωση του προγράμματος, χρησιμοποιήθηκαν τα αρχεία “.stim” από το προηγούμενο εργαστήριο, με την διαφορά ότι χρησιμοποιείται ο καταχωρητής PORTB για την εισαγωγή χαρακτήρων στον AVR, αντί για τον r15 που χρησιμοποιήθηκε στα προηγούμενα εργαστήρια. Παρατηρήθηκε ότι το αποτέλεσμα της προσομοίωσης ήταν το ίδιο με το αποτέλεσμα των προηγούμενων εργαστηρίων. Τα αρχεία “.stim” καθώς και το αρχείο “.log” που χρησιμοποιήθηκαν περιλαμβάνονται στον παραδοτέο φάκελο.