Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

Project 4

Ημερομηνία Παράδοσης: 7 Νοεμβρίου 2021

Αμπλιανίτης Κωνσταντίνος

2017030014

Σκοπός της Άσκησης:

Σκοπός του εργαστηρίου είναι η μετατροπή του κώδικα από Assembly σε γλώσσα C λαμβάνοντας υπόψιν τους πόρους του μικροελεγκτή.

Περιγραφή της τεχνολογίας που χρησιμοποιήθηκε:

Για την υλοποίηση της συγκεκριμμένης εργαστηριακής άσκησης χρησιμοποιήθηκε το Microchip Sudio (version 7.0.2542) της Microchip Technology.

Περιγραφή επίλυσης της άσκησης:

Αρχείο C

Ζητούμενο της εκφώνησης ήταν η δημιουργία ενός αρχείο .c το οποίο να περιελάμβανε όλες τις αρχικοποιήσεις. Πρώτα έγιναν οι αυτές που είναι υπεύθηνες για τον Timer που έχει επιλεχθεί προς χρήση και το UART. Η υλοποίηση των παραπάνω έγινε εύκολα, μέσω της δημιουργίας δύο συναρτήσεων USART_Init και TIMER0_INIT. Και οι δύο συναρτήσεις είναι τύπου void και δεν έχουν ορίσματα. Οι αρχικοποιήσεις τόσο του Timer όσο και του UART παραμένουν ίδιες με αυτές του προηγούμενου εργαστηρίου. Όσον αφορά το κομμάτι της μνήμης, η δεύσμευση του χώρου για κάθε ένα από τα seven segment displays όσο και για τις πιθανές καταστάσεις τους γίνεται με την χρήση arrays. Η ίδια διαδικασία ισχύει και για την αρχικοποίηση των πιθανόν καταστάσεων και για τις τιμές των rotate_reg και του offset_reg. Στην main του αρχείου έγιναν οι αρχικοποιήσεις των DDRA και DDRC με σκοπό να μπορεί να γίνει η χρήση των αντίστοιχων PORTS και τέλος υλοποιήθηκε και ο ατέρμονος βρόχος.

Αρχείο Assembly

Για την διαχείρηση των interrupts χρησιμοποιήθηκε αρχείο γραμμένο σε γλώσσα Assembly. Ο λόγος είναι ότι με αυτόν τον τρόπο γίνεται καλύτερη διαχείρηση των πόρων. Συγκριτικά με τον κώδικα της Assembly για το εργαστήριο 3 υπάρχουν ορισμένες διαφορές. Αρχικά η ονομασία ορισμέων καταχωρητών (συγκεκριμένα των καταχωρητών rotate_reg και address_adder) έχει αφαιρεθεί αφού δεν υπήρχε τρόπος να παραμείνει και οι καταχωρητές χρησιμοποιούνται ως r20 και r21. Μία ακόμα αλλαγή αφορά το όνομα του interrupt handler **getc** καθώς υπάρχει ομόνυμη συνάρτηση στη c και έτσι καθισταταί αδύνατο να χρησιμοποιηθεί. Επιπλέον, εφόσον δεν μπορεί να είναι γνωστό ποιούς καταχωρητές θα χρησιμοποιήσει ο compiler της C για την πρόσβαση στην μνήμη η σύμβασή

που υπήρχε ότι ο X θα αφορά μόνο το main πρόγραμμα καταρρέει. Αντ'αυτού γίνεται φόρτωση της κατάλληλης διεύθυνσης κάθε φορά που είναι απαραίτητη η φόρτωση δεδομένων από την μνήμη. Τέλος, δεδομένης της παραπάνω αλλαγής έγινε μία προσθήκη μίας μεταβλητής τιμής (offset_reg) που είναι υπεύθηνη για να κρατά την θέση της οθόνης που θα βρεθεί το portC στην επόμενη κλίση του Timer.

Σύνδεση C με Assembly

Για την σύνδεση των δύο αρχείων, χρείαστηκε να γίνουν αλλαγές τόσο στο πρώτο όσο και στο δεύτερο αρχείο.

1). **Αλλαγές στο αρχείο C.**

Αρχικά στο αρχείο C πρέπει να εισαχθούν οι βιβλιοθήκες <avr/io.h> υπεύθυνη για τα inputs/ouputs καθώς και η <avr/interrupt.h> η οποία θα εισάγει τις μεταβλητές για την διαχείρηση των interrupts. Επιπλέον, εισάγονται δύο interrupt service routines(ISR) με πρωότο όρισμα τη σταθερά που αντιστοιχεί στο interrupt που είναι επιθυμητό και δεύτερο το ISR_NAKED με σκοπό την διατήρηση του return address στο αρχείο της C. Οι δύο συναρτήσεις καλούν τους αντίστοιχους interrupt handlers που είναι γραμμένοι σε Assembly και βρίσκονται στο αρχείο .s. Επιπλέον είναι απαραίτητη τη χρήση του reti() με σκοπό να μην γίνεται εκτέλεση κάθε φορά και των δύο interrupt handlers.

```c
#include <avr/io.h>
#include <avr/interrupt.h> // call the interrupt direction library
#include <stdlib.h>

// the fucntions that are written in assembly code
extern void usart_recieve(void);
extern void timer0Ovf(void);

// practically tell the compiler that this is for the
ISR(TIMER0_OVF_vect,ISR_NAKED)
{
    timer0Ovf();
    reti();
}

ISR(USART_RXC_vect,ISR_NAKED)
{
    usart_recieve();
    reti();
}


// create the two arrays as i did in assembly
uint8_t seven_seg_status[11] = {0b00000011,0b10011111,0b00100101,0b00001101,0b10011001,0b00100101,0b01000001,0b00011101,0b00000001,0b00001001,0b11111111};
uint8_t port_C_segment[8];
//store the rotate register as global to be able to get accessed by the extern values
uint8_t rotate_reg = 0b00000001;
// will be usefull for keeping in wich segment i am each time
uint8_t offset_reg =0b00000000;
```

2). **Αλλαγές στο αρχείο .s**

Για το αρχείο .s αρχικά γίνονται τα απαραίτητα defines ώστε να δουλεύουν τα i/o. Έπειτα, χωρίζεται ο κώδικας σε δύο sections. Το data section και το code section. Στο data section γίνεται η αρχικοποίηση των interrupt handlers με .global έτσι

ώστε να μπορούν να είναι visible στο υπόλοιπο project και κατά συνέπεια στο αρχείο .c.

```
timer0Ovf:
            ; i load the value that will take the specific portC
            ;ld adress_adder, X+
            ldi ZL, lo8(offset_reg)
            ldi ZH, hi8(offset_reg)
            ld r22, Z // load the offset of the ports .. This will show in witch display i was before the interrupt

            // save the  value to a reg to save it for next
            mov r23,r22
            // increase for the next use
            inc r22
            st Z, r22

            // load the porc displays positions
            ldi YL, lo8(port_C_segment)
            ldi YH, hi8(port_C_segment)

            // find wich position
            add YL,r23

            ld  r21,Y

            ; change the values
            ; process to load the value of the rotate reg
            ldi ZL, lo8(seven_seg_status)
            ldi ZH, hi8(seven_seg_status)

            ;go to the proper address in the memory
            add ZL, r21

            ld r2, Z


            ldi XL,lo8(rotate_reg)
            ldi XH,hi8(rotate_reg)
            ld  r20,X
```

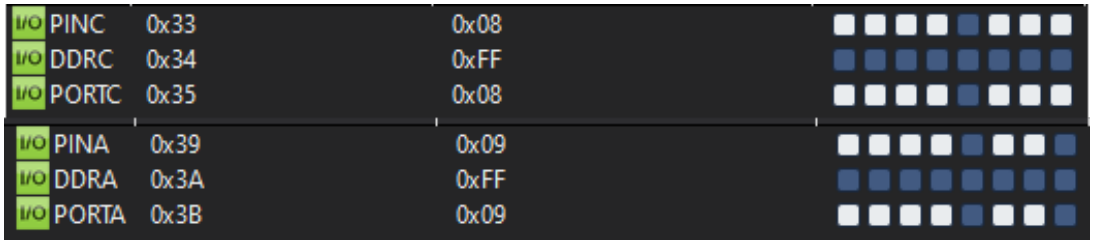
Αποτελέσματα

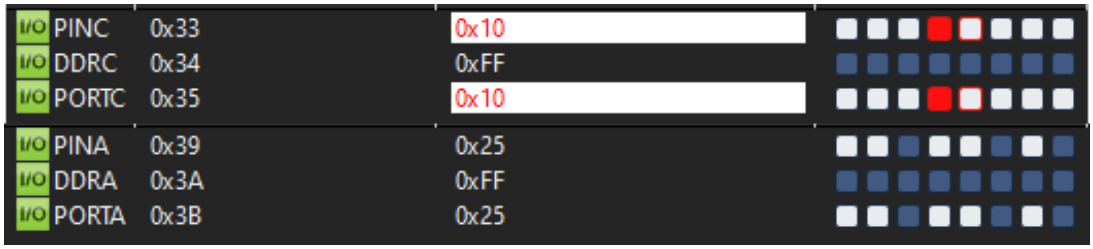
Ενδεικτικά αποτελέσματα για το εργαστήριο:

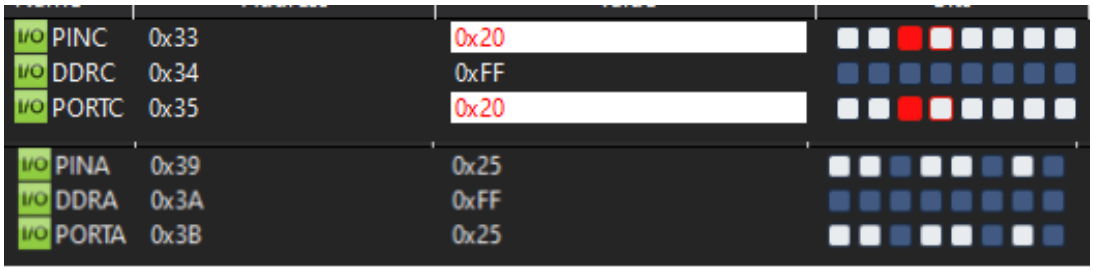
```
#16034
TCNT2 = 0x47
#12
TCNT2 = 0x4b
#12
TCNT2 = 0x0d
#12
TCNT2 = 0x0a
#13964
TCNT2 = 0x47
#12
TCNT2 = 0x4b
#12
TCNT2 = 0x0d
#12
TCNT2 = 0x0a
#19963
TCNT2 = 0x47
#12
TCNT2 = 0x4b
#12
TCNT2 = 0x0d
#12
TCNT2 = 0x0a
#29964
TCNT2 = 0x47
#12
TCNT2 = 0x4b
#12
TCNT2 = 0x0d
#12
TCNT2 = 0x0a
```

Όπως φαίνεται παραπάνω τα OK<CR><LF> είναι και πάλι τέσσερα όσα και τα διαφορετικά inputs που δίνονται από το stimuli file (χρησιμοποιήθηκε το ίδιο αρχείο με το εργαστήριο 3).

Ενδεικτικά αποτελέσματα για τον δεύτερο αριθμό







Κατάσταση της μνήμης

```
data 0x0060  40 03 9f 25 0d 99 25 41 1d 01 09 ff 06 03 06 07 09 02 02 01 0a 00 00 00 00 00 00 00 00 00 00 00 00
data 0x0082  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x00A4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x00C6  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
data 0x00E8  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Πρώτο κατα σειρά είναι το rotate_reg. Ακολουθούν οι θέσεις μνήμης των καταστάσεων των seven segment, ο αριθμός ένδειξης επόμενης οθόνης και τέλος οι θέσεις μνήμης για τα PORTC.