Εργαστήριο 3

Ονομα: Μαυρογιώργης Δημήτρης
ΑΜ: 2016030016
ΗΡΥ 411 - Ενσωματωμένα Συστήματα Μικροεπεξεργατών
Πολυτεχνειο Κρητης

April 19, 2021

Σκοπός

Σκοπός του τέταρτου εργαστηρίου είναι να υλοποιήσουμε σε κώδικα C τις αρχικοποιήσεις των TIMER0 και USART, κρατώντας αυτούσιους τους κώδικες των interrupt handler που είχαν υλοποιηθεί στο εργαστήριο 2 και 3 αντίστοιχα. Στόχος μας είναι, δηλαδή, να συνδυάσουμε τον κώδικα C των αρχικοποιήσεων και του κώδικα assembly των handler προκειμένου να υλοποίησουμε τις προδιαγραφές που είχαν περιγραφθεί στα δύο προηγούμενα εργαστήρια.

Περιγραφή της υλοποίησης

Στο συγκερκιμένο εργαστήριο οι αλλαγές που χρειάστηκε να γίνουν είναι να κάνουμε τις αρχικοποιήσεις των δύο handler σε C και στη main του προγράμματος να χρησιμοποιήσουμε ένα infinite loop. Ειδικότερα, οι αρχικοποιήσεις των τιμών των καταχωρητών είναι ακριβώς ίδιες μου του προηγούμενου εργαστηρίου. Οι αρχικοποιήσεις αυτές έγιναν σε δύο ξεχωριστές συναρτήσεις και το μοναδικό που αλλάζει είναι ότι καλούνται στο main πρόγραμμα.

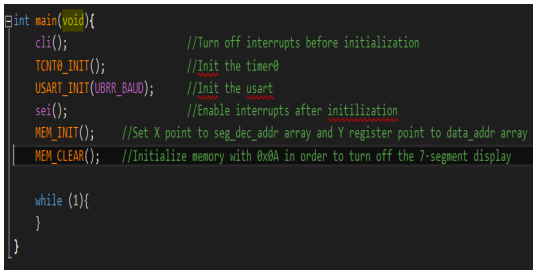
Όσον αφορά τον κώδικα assembly, η υλοποιήσεις των handler είναι η ίδια με του εργαστηρίου 3, με μοναδική αλλαγή τη μετονομασία τους από TIMER0_OVF και USART_RXC σε TIMER0_OVF_vect και USART_RXC_vect αντίστοιχα. Οι συγκεκριμένοι handler, δηλώθηκαν ως global για να μπορούν να αναγνωριστούν από τη C.

Παράλληλα, δηλώθηκε και η συνάρτηση MEM_INIT ως global, για την αρχικοποίηση των X και Y registers έτσι, ώστε να κρατούν τις διευθύνσεις των array που είναι αποθηκευμένες οι αποκωδικοποιήσεων των αριθμών για τα 7-segment και τα δεδομένα προς εμφάνιση αντίστοιχα.

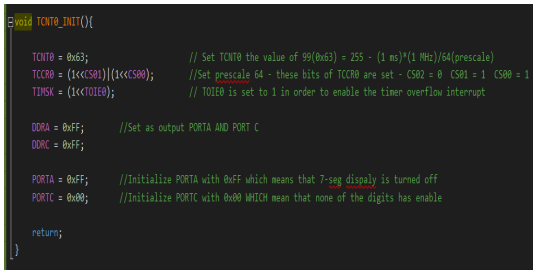
Κατόπιν, για να είναι σβησμένα τα 7-segment display, έγινε ένα clear της μνημης με τη συνάρτηση MEM_CLEAR, δηλαδή αποθηκεύτηκαν και στις 8 θέσεις το 0x0A. Για να μπορέσουμε να καλέσουμε και αυτή τη συνάρτηση στη C, δηλώθηκε και αυτή ως global. Τέλος, όλες οι δηλώσεις με .equ των γραμμάτων των εντολών, που δηλώθηκαν στην assembly του 3ου εργαστηρίου, έγιναν με #define.

Παρακάτω φαίνονται οι κώδικες C και οι αλλαγές του κώδικα Assembly, όπως περιγράφτηκαν προηγουμένως.

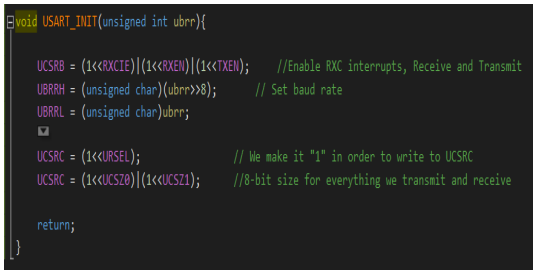
(a) C code for defines global variables and function signatures



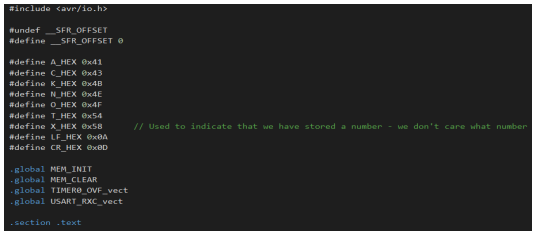
(b) C code for main



(c) C code for TIMER0 Initialization

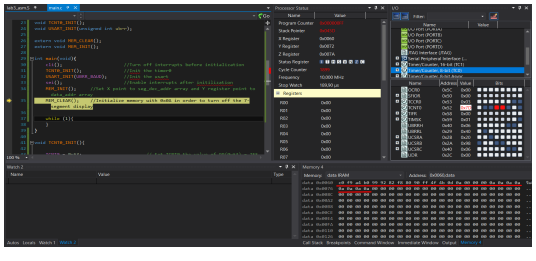


(d) C code for USART RXC Initialization

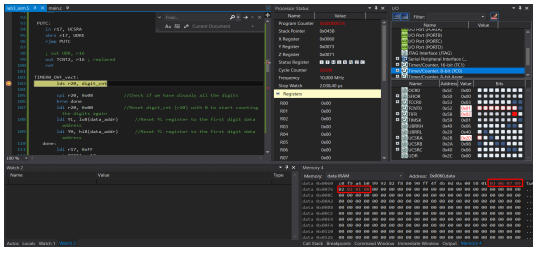


(e) Assembly code changes

Οι προσομοιώσεις του συγκεκριμένου εργαστηρίου είναι ακριβώς ίδιες με του προηγούμενου, εφόσον δεν έγιναν καθόλου αλλαγές στην υλοποίηση των TIMER0 και USRART handler. Ειδικότερα, στις παρακάτω εικόνες φαίνεται ότι γίνεται σωστά το initialization των καταχωρητών του TIMER0 και USART, καθώς και η αποθήκευση των αποκωδικοποιήσεων για τα 7-segment και του μηνυματος OK<CR><LF>. Επίσης, βλέπουμε ότι εκτελείται σωστά ο USART handler και οι τιμές οι οποίες λαμβάνονται αποθηκεύονται στη μνήμη, ενώ παράλληλα φαίνεται ότι εκτελείται σωστά και ο TIMER0 handler. Τέλος, όσον αφορά την αποστολή του OK<CR><LF>, μετά τη λήψη των αριθμώ, γίνεται η εγγραφή του στο αρχείο lab4.log, γεγονός που επιβεβαιώνει τη σωστή λειτουργία.



(a) Atmel Studio 7 - Initialization of TIMER0 and USART



(b) Atmel Studio 7 - Execution of N01229763<CR><LF>

Επιπρόσθετα, κατα την προσομοίωση του κώδικα C, μπορούμε να δούμε τον κώδικα Assembly που δημιουργήθηκε για τις αρχικοποιήσεις. Όπως φαίνεται παρακάτω ο κώδικας αρχικοποίησης σε assembly του 3ου εργαστηρίου και ο κώδικα που δημιούργησε ο compiler είναι παρόμοιοι, με μικρές διαφορές. Οι κυριότερες διαφορές είναι ότι τα ονόματα των καταχωρητών μεταφράστηκαν στις διευθύνσεις τους και μία άλλη αλλαγή είναι ότι χρησιμοποιούνται διαφορετικοί καταχωρητές (πχ στο 3ο Lab είχε χρησιμοποιηθεί για τον TIMER0 ο R16, ενώ στο συγκεκριμένο Lab χρησιμοποιείται ο R24).

```
   TCNT0 = 0x63;                    // Set TCNT0 the value of 99(0x63) = 255 - (1 ms)*(1 MHz)/64(prescale)
000000DE  LDI R24,0x63      Load immediate
000000DF  OUT 0x32,R24      Out to I/O location
   TCCR0 = (1<<CS01)|(1<<CS00);       //Set prescale 64 - these bits of TCCR0 are set - CS02 = 0  CS01 = 1  CS00 = 1
000000E0  LDI R24,0x03      Load immediate
000000E1  OUT 0x33,R24      Out to I/O location
   TIMSK = (1<<TOIE0);              // TOIE0 is set to 1 in order to enable the timer overflow interrupt
000000E2  LDI R24,0x01      Load immediate
000000E3  OUT 0x39,R24      Out to I/O location
   DDRA = 0xFF;      //Set as output PORTA AND PORT C
000000E4  SER R24       Set Register
000000E5  OUT 0x1A,R24      Out to I/O location
   DDRC = 0xFF;
000000E6  OUT 0x14,R24      Out to I/O location
   PORTA = 0xFF;      //Initialize PORTA with 0xFF which means that 7-seg dispaly is turned off
000000E7  OUT 0x1B,R24      Out to I/O location
   PORTC = 0x00;      //Initialize PORTC with 0x00 WHICH mean that none of the digits has enable
000000E8  OUT 0x15,R1       Out to I/O location
000000E9  RET       Subroutine return
```

(a) Assembly generated code for TIMER0 initialization

```
   UCSRB = (1<<RXCIE)|(1<<RXEN)|(1<<TXEN);     //Enable RXC interrupts, Receive and Transmit
000000EA  LDI R18,0x98      Load immediate
000000EB  OUT 0x0A,R18      Out to I/O location
   UBRRH = (unsigned char)(ubrr>>8);       // Set baud rate
000000EC  OUT 0x20,R25      Out to I/O location
   UBRRL = (unsigned char)ubrr;
000000ED  OUT 0x09,R24      Out to I/O location
   UCSRC = (1<<URSEL);                // We make it "1" in order to write to UCSRC
000000EE  LDI R24,0x80      Load immediate
000000EF  OUT 0x20,R24      Out to I/O location
   UCSRC = (1<<UCSZ0)|(1<<UCSZ1);      //8-bit size for everything we transmit and receive
000000F0  LDI R24,0x06      Load immediate
000000F1  OUT 0x20,R24      Out to I/O location
000000F2  RET       Subroutine return
```

(b) Assembly generated code for USART initialization