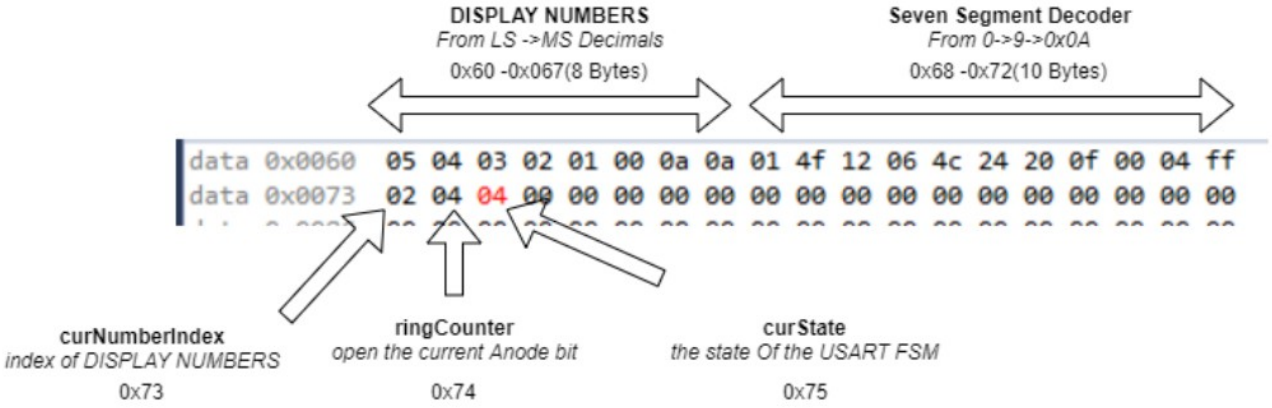
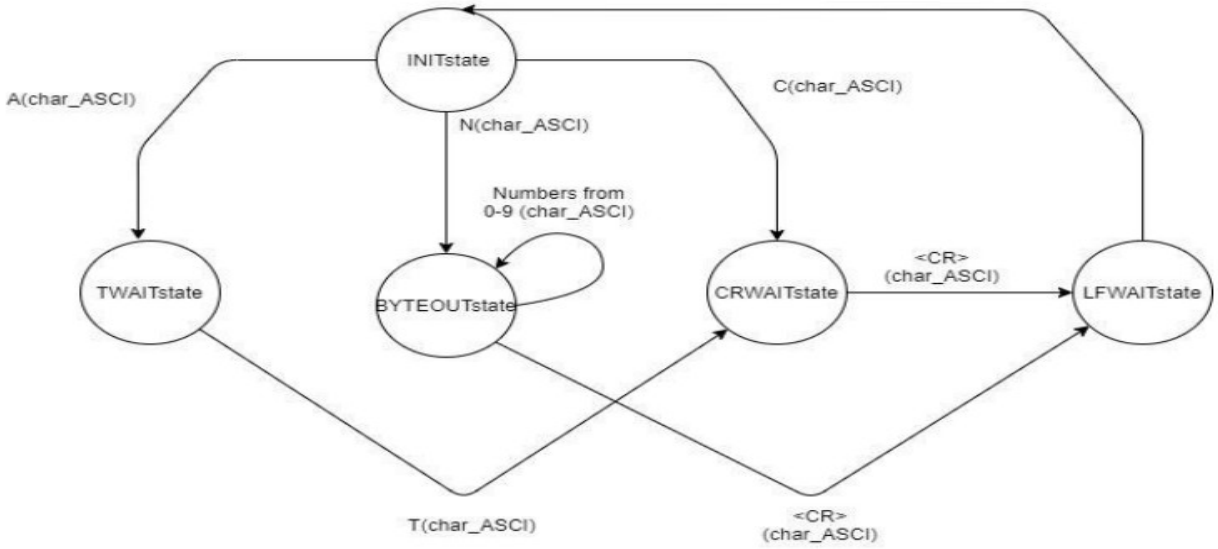
Θεόδωρος Μπάρκας 2016030050 LAB5

Σκοπός άσκησης

Στη Συγκεκριμένη άσκηση σκοπός ήταν η κατανόηση λειτουργίας και η εξοικείωση με τον Watchdog timer. Καθώς και η εξοικείωση με Cold Starts - Warm Starts μεσώ της κατανόησης και της εν μέρη υλοποίησης τους.





Μετατροπές άσκησης

Έβγαλα από την συνάρτηση INIT τις αρχικοποιήσεις της μνήμης όπως τα SSD-Decoder Related Data και τους αρχικούς αριθμούς που προβάλλονται : θέσεις 0x60 εως 0x72 ,καθώς και τις θέσεις 0x73 και 0x74 που αντιπροσωπευόσουν το curNumberIndex και ringCounter αντίστοιχα(βλ. πάνω σχήμα).Τα στοιχειά αυτά που έβγαλα τα αρχικοποιώ πλέον σε μια άλλη συνάρτηση η οποία ονομάζεται dataInit. Επομένως ο τωρινός κώδικας μου εμπεριεχει:

- την void INIT() η οποία είναι υπεύθυνη για τις αρχικοποιήσεις του State της FSM καθώς και για την αρχικοποίηση της USART ,TIMER0,DDRx (OUTs).
  Επίσης στη συνάρτηση INIT προστεθήκαν τα Control Bits του WDT δηλαδή μεταβλήθηκε ο καταχωρητής WDTCR. `WDTCR|=(1<<WDE)|0x03; //watchdog timer to 32.5 ms`

- την void dataINIT() η οποία είναι υπεύθυνη για τις υπόλοιπες αρχικοποιήσεις της RAM εκτός της curState.

Επίσης υλοποιήσαμε μια συνάρτηση void USART_TRANSMIT_RESET()
Για να γίνει Fire η συγκεκριμένη συνάρτηση μόνο στην κατάσταση RESET βάλαμε τον παρακάτω κωδικά στη main():

```
int main(void)
{
    unsigned char *curState=(unsigned char *) 0x75;//set state init
    if(*curState==(unsigned char)PREINITstate){
        dataInit();
        INIT();
    }
    else{
        INIT();
        USART_Transmit_Reset();
    }
    while (1)
    {
    }
}
```

και άλλαξα τα states values ως εξης :

```
#define PREINITstate 0x00
#define INITstate 0x05
#define TWAITstate 0x01
#define BYTEOUTstate 0x02
#define CRWAITstate 0x03
#define LFWAITstate 0x04
```

Εκμεταλλεύτηκα δηλαδή το ότι γνωρίζω ότι η ram κατά το Boot του συστήματος χωρίς Watchdog timer είναι αρχικοποιούμενη στη θέση 0x00. Έτσι μπορούμε να καταλάβουμε πότε κάνει Restart ή πότε κάνει άπλα Power On όπως φαίνεται και στη παραπάνω main.

Σε περίπτωση που διακοπεί και δεν ολοκληρωθεί η μετάδοση μεσώ USART μέσα στο προβλεπόμενο χρόνο που προβλέπει ο Watchdog timer Prescale(ορίζεται μέσα στην void INIT()) ή στη περίπτωση που απλώς τύχει να μην γίνει μετάδοση για το προβλεπόμενο χρονικό διαστημα. Επαναφερουμε το state στο INITstate και κατόπιν ξανά αρχικοποιούμε τα OUTs (συνάρτηση INIT) .
Κατόπιν ενημερώνω με το μήνυμα RESET για την επανεκκίνηση(συνάρτηση USART_TRANSMIT_Reset()) .

Τέλος μέσα σε κάθε USART κάνουμε reset τον WDT.

```
//reset WDT
__asm__(
"WDR\n"
);
```

COLD START/WARM START

Εφόσον κάνουμε μόνο ένα απαραίτητο μέρος των αρχικοποιήσεων μας είμαστε πιο κοντά στο να πούμε ότι κάνουμε ένα Warm Start εκτός του Power On που τρέχει όταν το state της FSM είναι το PREINITstate .Αυτό φαίνεται καθώς διατηρούμε τα δεδομένα της μνήμης :

- Τα curNumberIndex και ringCounter (0x73,0x74).
- Τα SSD-Decoder Related Data (0x67-0x72).
- Τα τελευταία εισαχθέντα Decimal Digit Related Data (0x60-0x67).

Σε κάθε περίπτωση σε περίπτωση λάθους εισόδου χρήστη ή κάποιας κατάστασης που εντοπίσαμε και δεν ξέρουμε πως να να αντιμετωπίσουμε θα μπορούσαμε θα μπορουσαμε να θεσουμε το state στο PREINITstate και να ξανά εκτελέσουμε Cold Start.
Η διάφορα μεταξύ των δυο βρίσκεται στο ότι στη μια περίπτωση κρατάμε τα δεδομένα στη μνήμη και δεν τα σβήνουμε στο Reset(COLD START).
Ενώ στην άλλη κάνουμε Initialize μόνο τα OUTs του μικροελεγκτης μας και κρατάμε τη μνήμη(WARM START).

TESTING

- Για να γίνει σωστά το Testing έφτιαξα ένα Test το οποίο τερματίζει σε λάθος χαρακτήρα (UDR RECIEVE)αφήνοντας έτσι το State σε μια κατάσταση διαφορετική του INITstate και παρακολούθησα τη συμπεριφορά του προγράμματος στο τελευταίο Interrupt της USART (με 1 Break point στο τελος του USART Interrupt).Κατόπιν επιβεβαίωσα ότι μπαίνει στο σωστό if Statement της main και κοίταξα να δω αν είναι σωστά ορισμένες οι τιμές τις μνήμης RAM .
- Επίσης άλλα 2 Break Points έχω βάλει στην If της main όπως φαίνεται στην εικόνα της main παραπάνω .
- Αλλο 1 Break Point στο τελος της TIMER0_OVF Interrupt.

Ο prescaler ειναι ορισμενος με **WDP2=0,WDP1=1,WDP0=1** το οποίο στην προσομοίωση μας αναπαριστάτε σε χρόνο 13ms . Αξίζει να σημειωθεί εδώ ότι το interrupt του TCNT0_OVF γίνεται κάθε 4ms με την τωρινή υλοποίηση.

Πιο συγκεκριμένα για να δοκιμάσουμε τη σωστή λειτουργιά του WDT είναι να περάσουμε απο την θήρα USART N<INPUTS> χωρίς να δώσουμε <CR> και <LF> .Έτσι όταν σταματήσουμε να δίνουμε εισόδους στη USART περιμένουμε ο Watchdog timer να δώσει reset και όταν γίνει αυτό θα θέλαμε να

δούμε:

- Να μπαίνει σε σωστό If statement στη main(WARM START) (1)
- τις τελευταίες τιμές γραμμένες στη μνήμη (<INPUTS>), curNumberIndex ,ringCounter (2)
- αλλαγή του curState της FSM (3)

Επομένως το Stimuli File που θα χρησιμοποιήσω είναι το εξής :

```
R15 = 0x4E
UCSRA = 0b10000000
#3000
R15 = 0x30
UCSRA = 0b10000000
#3000
R15 = 0x31
UCSRA = 0b10000000
#3000
R15 = 0x32
UCSRA = 0b10000000
#3000
R15 = 0x33
UCSRA = 0b10000000
#3000

$log TCNT2
$startlog lab3.log
#10000
$stoplog
```

→ Την πρώτη φορά (Cold Start) μπαίνει στο σωστό Statement που αναλογεί σε Cold Start Handler

```
int main(void)
{
    unsigned char *curState=(unsigned char *) 0x75;//set state init
    if(*curState==(unsigned char)PREINITstate){
        dataInit();
        INIT();
    }
    else{
        INIT();
        USART_Transmit_Reset();
    }
    while (1)
    {
    }
}
```

→ Τρέχω μερικές φόρες με break point στον Timer0_OVF interrupt ώστε να αλλάξουν τα περιεχόμενα της μνήμης έως ότου φτάσω να έχω αυτή τη μνήμη:

```
data 0x0060  05 06 07 08 09 00 01 02 01 4f 12 06 4c 24 20 0f 00 04 ff
data 0x0073  01 02 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Θυμίζω ότι 0x73→ curNumberIndex , 0x74→ ringCounter , 0x75→ curState , 0x76→ last USART received value

Θυμίζω επίσης και τα States της FSM

```
#define PREINITstate 0x00
#define INITstate 0x05
#define TWAITstate 0x01
#define BYTEOUTstate 0x02
#define CRWAITstate 0x03
#define LFWAITstate 0x04
```

→ Κατόπιν τρέχω το stimuli file .Ο Debugger σταματάει και στο Break Point στο τέλος της USART RECIEVE Interrupt έχουμε :

```
data 0x0060  0a 0a 0a 0a 0a 0a 0a 0a 01 4f 12 06 4c 24 20 0f 00 04 ff
data 0x0073  01 02 02 4e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

εδώ βλέπουμε και τον 0x76 που παίρνει την τιμή που εισηχθεί από την USART

→ Το τελευταίο Interrupt που δίνουμε μεσώ του stimuli file:

```
data 0x0060  03 02 01 00 0a 0a 0a 0a 01 4f 12 06 4c 24 20 0f 00 04 ff
data 0x0073  01 02 02 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

βλέπουμε ότι είμαστε στο state : BYTEOUTstate
Παρατηρούμε ότι το curState=0x02=BYTEOUTstate

→ Αφού συνεχίσουμε τον Debugger σταματάει για άλλες 3 φόρες στο Timer0_OVF interrupt .Το οποίο είναι λογικό αφού αυτό το Interrupt δίνετε κάθε =4ms → 4ms*3=12ms<13ms (WDT).

```
data 0x0060  03 02 01 00 0a 0a 0a 0a 01 4f 12 06 4c 24 20 0f 00 04 ff
data 0x0073  04 10 02 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

βλέπουμε ότι ο curNumberIndex=0x04 και ο ringCounter=0b00010000 .

→ Συνεχίζοντας τον Debugger μπαίνουμε στο δεύτερο If statement (Warm Start)

```
int main(void)
{
    unsigned char *curState=(unsigned char *) 0x75;//set state init
    if(*curState==(unsigned char)PREINITstate){
        dataInit();
        INIT();
    }
    else{
        INIT();
        USART_Transmit_Reset();
    }
    while (1)
    {
    }
}
```

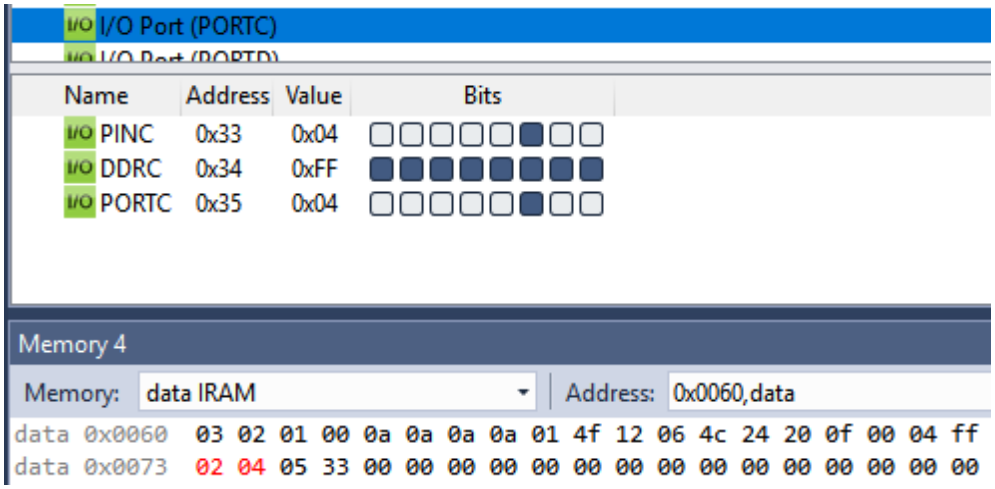
και η μνήμη όπως φαίνεται παρακάτω δεν αλλάζει:

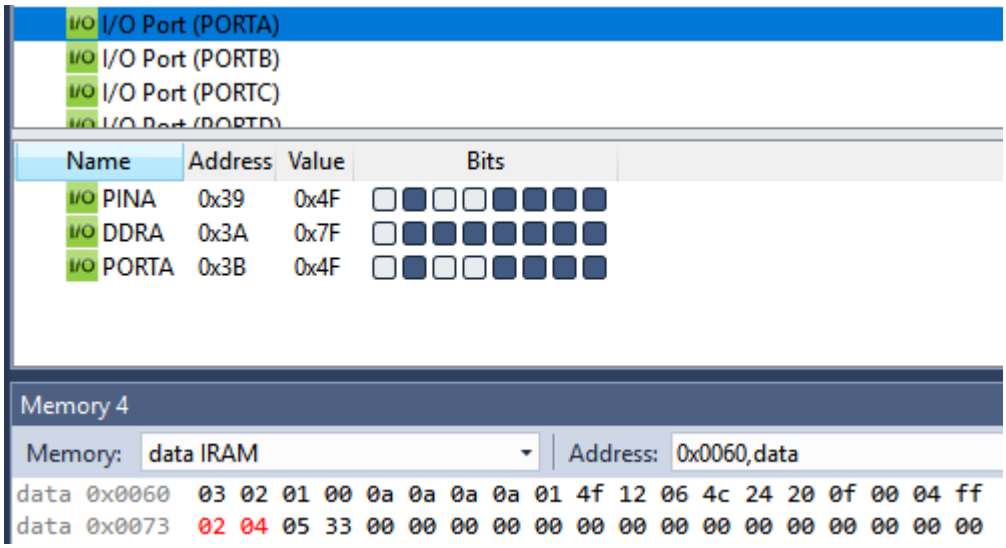
```
data 0x0060  03 02 01 00 0a 0a 0a 0a 01 4f 12 06 4c 24 20 0f 00 04 ff
data 0x0073  04 10 02 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

→Παρόλα αυτά χάνονται οι τιμές από τα OUTs και για αυτό καλούμε ξανά την INIT() που είναι υπεύθυνη να τα επαναχρησιμοποιήσει καθώς και για να θέσει το curState=INITstate=0x05

```
data 0x0060  03 02 01 00 0a 0a 0a 0a 01 4f 12 06 4c 24 20 0f 00 04 ff
data 0x0073  04 10 05 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

→ Έπειτα ελέγχουμε τη σωστή λειτουργιά του Display προχωρώντας το Debugger εως ότου φτάσει στο τέλος της Timer0_OVF interrupt με ανοιχτή την 3η άνοδο η οποία θα πρέπει να δείχνει το 3. Όντως:




I/O Port (PORTA)
I/O Port (PORTB)
I/O Port (PORTC)
Name
Address
Value
Bits
PINA
0x39
0x4F
DDRA
0x3A
0x7F
PORTA
0x3B
0x4F
Memory 4
Memory:
data IRAM
Address:
0x0060,data
data 0x0060 03 02 01 00 0a 0a 0a 0a 01 4f 12 06 4c 24 20 0f 00 04 ff
data 0x0073 02 04 05 33 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00