ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ

Εργαστηριακή Άσκηση 6η

Hard reset, soft reset και watchdog timer

Σωτήριος Μιχαήλ
2015030140

ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ

Εισαγωγή

Σκοπός είναι η υλοποίηση ενός προγράμματος το οποίο επικοινωνεί μέσω της optional (communication) θύρας RS232 του STK500 με ένα τερματικό (λ.χ. PuTTY) σε PC, και υλοποιεί μερικές απλές εντολές, καθώς και μία οθόνη 7-segment, η οποία δείχνει αριθμούς τους οποίους έχουμε αποθηκεύσει μέσω μίας εκ των προαναφερθέντων εντολών. Σε αυτή την άσκηση, πρέπει επίσης να υλοποιηθεί μία λειτουργία soft και hard reset μέσω του watchdog timer του AVR.

Οι εντολές που υλοποιήθηκαν είναι οι εξής:

Εντολή από PC (ASCII)	**Επεξήγηση Ορίσματος X**	**Ενέργεια Εντολής**	**Απάντηση προς PC (ASCII)**
AT<CR><LF>	-	Απλή Απάντηση OK με το <LF>, καμμία άλλη ενέργεια	OK<CR><LF>
C<CR><LF>		Καθαρισμός οθόνης και απάντηση OK μετά το <LF>	OK<CR><LF>
N<X> <CR><LF>	<X>: 1-8 χαρακτήρες ASCII που όλοι αντιστοιχούν σε δεκαδικό ψηφίο	Καθαρισμός οθονης, αποθήκευση ορισμάτων στην μνημη της οθόνης, απάντηση OK μετά το <LF>	OK<CR><LF>

Η βασική ροή του προγράμματος παρουσιάζεται παρακάτω, σε μορφή ψευδοκώδικα:

```
START
    INITIALIZE_UART( )
    DETECT_RESET_SOURCE( )
    INITIALIZE_WATCHDOG_TIMER()
    INITIALIZE_OUTPUT()
    INITIALIZE_TIMER_0( )
    WHILE(1) {
        UPDATE_DISPLAY()
        IF (RECEIVE_COMMAND) {
            PROCESS_COMMAND()
            REPLY()
            REFRESH_DISPLAY()
        }
    }
```

Παρακάτω παρουσιάζονται περισσότερες λεπτομέρειες για την υλοποίηση κάθε μέρους αυτής της ροής.

Αρχικοποιήσεις

Οι αρχικοποιήσεις για υλοποίηση βασίζονται στο ποιόν μικροελεγκτή χρησιμοποιούμε και στην συχνότητα του ρολογιού του. Σε αυτή τη περίπτωση, έχουμε τον μικροελεγκτή ATmega16L και χρησιμοποιούμε ένα εξωτερικό ρολόι, η λειτουργία του οποίο θα αναλυθεί παρακάτω. Επομένως, πρέπει να υπολογίσουμε ανάλογα κάποιες τιμές για την αρχικοποίηση της διεπαφής U(S)ART καθώς και του timer0, που χρησιμοποιούμε για την ενημέρωση της οθόνης.

Η επικοινωνία με το PC, καθώς και η ενημέρωση της οθόνης γίνονται μέσω interrupts, επομένως, πρέπει να αρχικοποιήσουμε και τους interrupt handlers στην αρχή του προγράμματος, στις σωστές διευθύνσεις, ανάλογα με το interrupt που έχουμε.

Για την υλοποίηση του soft reset, χρησιμοποιούμε τον εσωτερικό watchdog timer του AVR. Όπως και οι άλλες λειτουργίες του AVR, έτσι και ο watchdog timer απαιτεί μία αρχικοποίηση για την λειτουργία του.

Ρολόι του μικροελεγκτή

Σε αυτή την υλοποίηση, χρησιμοποιήθηκε το on-board software clock του STK500, με δύο jumpers, ένα στη θέση XTAL1 και ένα στη θέση OSCSEL, στα PIN 1 & 2. Στον μικροελεγκτή, τα fuse bit τα οποία ελέγχουν το ρολόι πήραν τις τιμές CKSEL = 0000 και SUT = 10, επιλέγοντας “Ext. Clock; Start-up time: 6 CK + 64 ms;” στον προγραμματιστή της συσκευής του Atmel Studio 7.0.

Αυτή η αλλαγή έγινε έτσι ώστε να μηδενιστεί το σφάλμα επικοινωνίας μέσω USART, όπως εξηγείται περετέρω στο χωρίο “Διεπαφή USART” αυτής της αναφοράς.

Διευθύνσεις μνήμης

Στις προδιαγραφές του προγράμματος, ορίζεται η αποθήκευση των δεδομένων εισόδου τα οποία προορίζονται να εμφανιστούν στην οθόνη, δηλαδή ψηφία σε ASCII, στη μνήμη RAM του ελεγκτή. Έτσι ώστε να αποθηκευθούν τα δεδομένα στη μνήμη RAM και να μην αρχικοποιούνται κάθε φορά που έχουμε ένα soft reset, στον ορισμό του buffer χρησιμοποιούμε ένα attribute του avr-gcc compiler, με το οποίο του δίνουμε την οδηγία να τοποθετήσει τον buffer στη θέση “.noinit” της RAM. Αυτή η θέση μνήμης **δεν** αρχικοποιείται με κάθε soft reset του watchdog timer, και έτσι η οθόνη συνεχίζει να παρουσιάζει τον αποθηκευμένο αριθμό καθώς γίνεται soft reset. Σε περίπτωση hard reset (ή αλλιώς “cold start”), που γίνεται είτε μέσω του power switch, είτε με το κουμπί reset του STK500, είτε όταν η τάση του συστήματος ή πλακέτας πέσει κάτω του ορίου brown-out (το οποίο μπορεί να οριστεί από τον χρήστη μέσω των fuses του μικροελεγκτή), αρχικοποιούμε τις θέσεις “noinit” της RAM. Ο εντοπισμός του είδους που έχει λάβει χώρα γίνεται με τον έλεγχο των bits του καταχωρητή κατάστασης συστήματος MCUCSR. Κάθε είδους θέτει και ένα register ενεργό, στη περίπτωση του watchdog timer soft reset, ο καταχωρητής WDRF ενεργοποιείται.

Για την εμφάνιση του ψηφίου που αποθηκεύεται σε μορφή κωδικού ASCII, πρέπει να γίνει μία μετατροπή του ψηφίου από κώδικα ASCII σε ακέραιο αριθμό, μέσω της συνάρτησης atoi, η οποία είναι μέρος της standard βιβλιοθήκης της C. Ο ακέραιος αυτός χρησιμοποιείται ως δείκτης για ένα lookup table το οποίο έχει αποθηκευμένους όλους τους κωδικούς seven segment, έτσι ώστε να εμφανίζονται τα ψηφία στην οθόνη seven segment.

Η συνάρτηση που ελέγχει το είδος του reset που έχει συμβεί, μαζί με την συνάρτηση αρχικοποίησης του UART, καθώς η πρώτη χρησιμοποιεί την δεύτερη, πρέπει να εκτελεστεί **πριν** την main σε κάθε reset. Έτσι, δίνουμε στον avr-gcc οδηγία να τοποθετήσει τις δύο συναρτήσεις αυτές σε θέσεις μνήμης παραπάνω της main, έτσι ώστε να εκτελούνται εκείνες πριν την main, κατά σειρά που θέτουμε εμείς. Αυτό γίνεται μέσω του attribute “initX”, όπου X η σειρά εκτέλεσης. Επίσης πριν την main εκτελείται και η αρχικοποίηση του watchdog timer μετά από κάθε reset.

Interrupts

Για την ορθή λειτουργία της οθόνης, χρησιμοποιούμε τον timer 0 και το interrupt υπερχείλισης του. Για τον ελεγκτή αυτού του interrupt σε κώδικα C χρησιμοποιείται το macro "TIMER0_OVF_vect".

Για την επικοινωνία UART μέσω της θύρας RS232, θέλουμε να έχουμε ένα interrupt κάθε φορά που έχουμε μία ολοκληρωμένη μετάδοση στον buffer του UART, τον UDR. Επομένως, θέτουμε τα bits RXC, TXC του καταχωρητή UCSRA σε 1, και τα RXEN, TXEN και RXCIE του καταχωρητή UCSRB σε 1. Το RXCIE είναι το bit το οποίο ενεργοποιεί το interrupt για μία ολοκληρωμένη μετάδοση, και απαιτεί να είναι και το RXC bit ενεργό. Ο ελεγκτής του RXCIE interrupt βρίσκεται στη διεύθυνση 0x16 και για τον κώδικα C χρησιμοποιείται το macro "USART_RXC_vect".

Σε κρίσιμα κομμάτια του κώδικα, όπως είναι η αντιγραφή του buffer εισόδου ή η επεξεργασία της εντολής, όπου δε θέλουμε να υπάρχει η πιθανότητα καταστροφής δεδομένων στη RAM, πρέπει να φράσσεται η εκτέλεση των interrupt handlers. Αυτό γίνεται μέσω *atomic block,* της βιβλιοθήκης atomic του avr-gcc.

Watchdog timer

Ο watchdog timer είναι ένας μετρητής παρατήρησης, ο οποίος έχει εξ'ολοκλήρου δικό του κύκλωμα υλοποίησης, με δικό του ρολόι, το οποίο βασίζεται στη τάση που υπάρχει στη παροχή του AVR (όπως παρουσιάζεται στον παρακάτω πίνακα). Χρησιμοποιείται όταν θέλουμε το πρόγραμμα να επαναφέρεται αυτόματα σε περίπτωση που "κρεμάσει".

Για την χρήση του, χρειάζεται η ενεργοποίηση του, η οποία γίνεται εάν θέσουμε '1' το bit WDE του καταχωρητή ελέγχου του watchdog timer, WDTCR. Για να ελέγξουμε κάθε πότε συμβαίνει ένα soft reset, πρέπει να ορίσουμε τον prescaler του ρολογιού του watchdog timer, το οποίο γίνεται μέσω των bits WDP2..0 του WDTCR. Σε αυτή την άσκηση, έχουμε ένα soft reset κάθε δευτερόλεπτο εάν δεν έχουμε είσοδο ή έξοδο χαρακτήρα μέσω της σειριακής θύρας. Σύμφωνα με τις προδιαγραφές της άσκησης, σε κάθε είσοδο χαρακτήρα έχουμε επαναφορά του watchdog timer έτσι ώστε να μη συμβεί soft reset. Η επαναφορά γίνεται μέσω του *wdt_reset()* της βιβλιοθήκης wdt του avr-gcc.

WDP2	WDP1	WDP0	Typical Time-out at VCC = 5.0V
0	0	0	16.3ms
0	0	1	32.5ms
0	1	0	65ms
0	1	1	0.13s
1	0	0	0.26s
1	0	1	0.52s
1	1	0	1.0s
1	1	1	2.1s

Διεπαφή U(S)ART

Σε αυτή την εργαστηριακή άσκηση, χρησιμοποιούμε το USART ως απλό UART, δηλαδή, έχουμε ασύγχρονη επικοινωνία με το PC. Ο στόχος για την ταχύτητα επικοινωνίας με το PC είναι τα 9600 baud. Για να μηδενιστεί το σφάλμα επικοινωνίας μέσω USART, το ρολόι του μικροελεγκτή ρυθμίστηκε στα 3.6864 MHz. Αυτό έγινε διότι, λόγω του τύπου με τον οποίο υπολογίζουμε το UBRRL (ο οποίος παρουσιάζεται παρακάτω), δημιουργείται ένα σφάλμα, καθώς πάντα πρέπει να στρογγυλοποιούμε το αποτέλεσμα αυτού του τύπου προς τον κοντινότερο ακέραιο, με αποτέλεσμα ο ρυθμός μετάδοσης να μην είναι απόλυτα συγχρονισμένος με το ρολόι του μικροελεγκτή.

$$UBRRL = \frac{f_{oscillator}}{16\,x\,target_{baud}}$$

Εάν όμως θέσουμε το ρολόι σε ένα πολλαπλάσιο των 1.8432 MHz, μπορούμε να μηδενίσουμε το σφάλμα αυτό, όπως φαίνεται και στον παρακάτω πίνακα, για την τιμή ρολογιού 3.6864 MHz, η οποία χρησιμοποιείται σε αυτή την υλοποίηση. Έτσι, μπορούμε πολύ γρήγορα να τροποποιήσουμε το πρόγραμμα του μικροελεγκτή έτσι ώστε να έχουμε οποιονδήποτε ρυθμό επικοινωνίας δίχως σφάλμα.

3.6864 Mhz		
Bit Rate	**UBRR**	**% of error**
300	767	0.0
600	383	0.0
1200	191	0.0
2400	95	0.0
4800	47	0.0
9600	23	0.0
14400	15	0.0
19200	11	0.0
28800	7	0.0
38400	5	0.0
57600	3	0.0
76800	2	0.0
115200	1	0.0
230400	0	0.0

Για την σωστή επικοινωνία με το PC μέσω του RS232, πρέπει επίσης να ορίσουμε και τη διαμόρφωση των μηνυμάτων που μεταφέρονται. Αυτό γίνεται μέσω των bits UCSZ2..0 του καταχωρητή UCSRC. Καθώς ο καταχωρητής UCSRC μοιράζεται τη διεύθυνσή του με έναν άλλον καταχωρητή για εξοικονόμηση διευθύνσεων, χρειάζεται να θέσουμε και το URSEL bit σε 1, έτσι ώστε να γράψουμε τις σωστές τιμές για τα UCSZ2..0. Σε αυτήν την άσκηση, τα μηνύματα έχουν 1 stop bit και δεν έχουν parity. Επομένως, γράφουμε 1 στα bits UCSZ1, UCSZ0 και URSEL.

Timer0

Μέσω του μετρητή αυτού, θέλουμε να ανανεώνουμε και τα 8 ψηφία της οθόνης κάθε 4 ms. Χρησιμοποιήθηκε ο prescaler $f_{clk}/1024$, θέτοντας 1 τα bit CS00 και CS02, και η σωστή τιμή του TCCR0 για overflow κάθε 4 ms προκύπτει από τον τύπο:

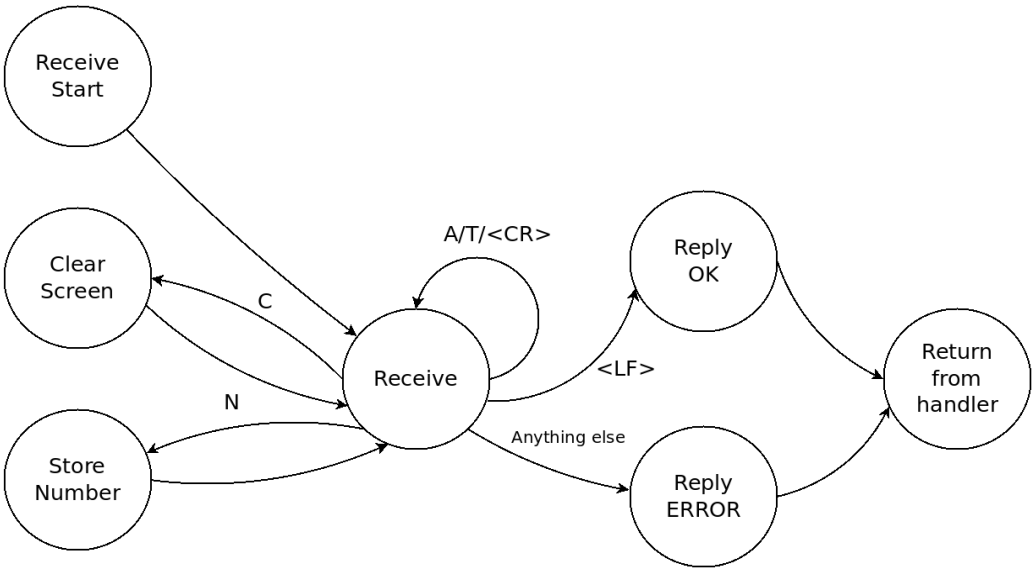
$$f_{timer0} = f_{clk}/1024 \approx 3.68\,kHz \Rightarrow T_{timer0} \approx 0.27\,ms \Rightarrow (0.27\,ms\,x\,(256 - TCCR0))\,x\,8 \approx 4\,ms \Rightarrow TCCR0 = 254$$

Η υλοποίηση του κυκλικού counter ο οποίος ελέγχει πιο ψηφίο εκ των οκτώ της οθόνης οδηγείται, το οποίο αλλάζει κάθε 0,27 ms, γίνεται στον interrupt handler του timer0, μέσω bit manipulation των bit εξόδου της θύρας C. Θέτουμε 1 σε ένα και μόνο bit, τη θέση το οποίο καθορίζει μία μεταβλητή η οποία αυξάνεται με κάθε ενεργοποίηση του interrupt handler και επιστρέφει στο 0 εάν ξεπεράσει τη τιμή 7, και έπειτα αντιστρέφουμε όλα τα bit, καθώς έχουμε LEDs τα οποία είναι κοινής ανόδου, και επομένως είναι ενεργά στο 0.

Διαχείριση RXCIE interrupt και είσοδος μέσω RS232

Με κάθε είσοδο ενός χαρακτήρα, ενεργοποιείται ο διαχειριστής interrupt RXC. Όταν έχουμε να νέο χαρακτήρα στον καταχωρητή UDR, ο διαχειριστής τον αποθηκεύει στο rx, το οποίο είναι ένας volatile buffer. Όταν εισαχθεί ένα newline (\n), τότε μία ενεργοποιείται μία σημαία rx_complete η οποία συμβολίζει πως έχει εισαχθεί μία εντολή. Η κατάσταση αυτής της σημαίας ελέγχεται συνεχώς από τον κυρίως βρόχο στην συνάρτηση main. Εάν η σημαία είναι ενεργή, η main καλεί μία συνάρτηση η οποία αντιγράφει τον buffer σε μία βοηθητική μεταβλητή, αμέσως μετά καλεί την συνάρτηση η οποία επεξεργάζεται την εντολή που μόλις εισήχθη, με όρισμα την βοηθητική μεταβλητή που κρατά τα δεδομένα τα οποία μόλις εισήχθησαν, αρχικοποιεί τον rx, και κατεβάζει τη σημαία rx_complete. Έτσι, μπορούμε να λάβουμε να δεδομένα ενώ επεξεργάζεται η εντολή.

Η επεξεργασία της εντολής είναι ουσιαστικά μία μηχανή πεπερασμένων καταστάσεων, η οποία αλλάζει κατάσταση ανάλογα τον χαρακτήρα, η βασική ροή της οποίας παρουσιάζεται με το παρακάτω διάγραμμα καταστάσεων:



Για τον καθαρισμό της οθόνης μέσω της εντολής “C<CR><LF>”, μία βοηθητική συνάρτηση θέτει την οθόνη σε 0xFF, και καθαρίζει την μνήμη στην οποία βρίσκονται τα ψηφία εισόδου αποθηκευμένα.

Έξοδος μέσω RS232

Η αποστολή χαρακτήρων μέσω του UART υλοποιείται με την uart_putc(), μία συνάρτηση η οποία παίρνει ως όρισμα έναν χαρακτήρα, και τον αποστέλλει στο UDR. Η συνάρτηση uart_putc() παρουσιάζεται ως αυτούσιο κομμάτι κώδικα:

```
void uart_putc(unsigned char ch)
{
      while (! (UCSRA & (1 << UDRE)));
      UDR = ch;
}
```

Οθόνη 7-segment

Η υλοποίηση της οθόνης, καθώς και του timer που χρησιμοποιείται για την ανανέωσή της, έχει καλυφθεί εκτενώς στις αναφορές των προηγούμενων εργαστηρίων. Η διαφορά σε αυτή την υλοποίηση είναι πως ο κώδικας που ελέγχει τα PORT A και C που ελέγχουν την οθόνη είναι όλα υλοποιημένα σε C.

Καταχωρητές και μεταβλητές

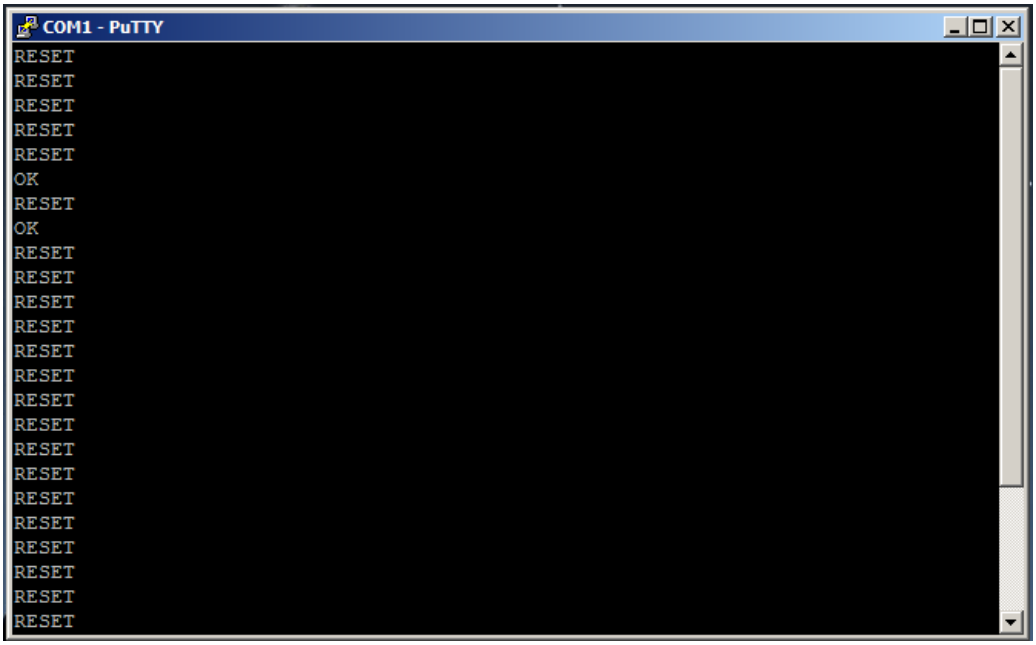
Για την ευκολότερη κατανόηση του κώδικα, παρατίθεται ένας πίνακας με τις καθολικές μεταβλητές που χρησιμοποιούνται στο πρόγραμμα, και την χρήση της κάθε μίας:

Μεταβλητή	Χρήση
ram_data	Δεδομένα της εντολής “N<X>\r\n”. Βρίσκεται στη θέση “noinit” της RAM
active_segment	Δείχνει ποιο εκ των 8 display είναι ενεργό
rx_n	Δείκτης του rx
rx	Volatile buffer ο οποίος κρατά τα δεδομένα εισόδου από UART
rx_len	Μήκος ακολουθίας χαρακτήρων εισόδου. Χρησιμοποιείται για την αντιγραφή

Στον κώδικα C, οι μεταβλητές έχουν όλες δηλωθεί ως χαρακτήρες χωρίς πρόσημο, “unsigned char”. Αυτό γίνεται διότι αυτός ο τύπος μεταβλητής χρησιμοποιεί το λιγότερο δυνατό χώρο στον μικροελεγκτή, και κάνει την υλοποίηση όσο πιο μικρή γίνεται. Το ίδιο ισχύει και τη χρήση του “for(;;) { }” στη θέση του “while(1)” για την υλοποίηση του ατέρμονος βρόχου που διατηρεί τη λειτουργία του μικροελεγκτή επ’αορίστον. Οι επαναλήψεις με “for” χωρίς μεταβλητές χρησιμοποιούν λιγότερο χώρο στον AVR.

Διεξαγωγή εργαστηρίου – προσομοίωση και επιβεβαίωση λειτουργίας

Στην προηγούμενη εργαστηριακή άσκηση εμφανίστηκε ένας περιορισμός της προσομοίωσης, καθώς δεν ήταν δυνατό να προσομοιώσουμε δεδομένα τα οποία εισέρχονται από τη θύρα RS232. Για αυτή την εργαστηριακή άσκηση όμως, η πλατφόρμα STK500 χρησιμοποιήθηκε για την επιβεβαίωση λειτουργίας της υλοποίησης σε πραγματικό μικροελεγκτή. Η θύρα RS232 SPARE χρησιμοποιήθηκε για την αποστολή των εντολών στον μικροελεγκτή. Η θύρα αυτή ενεργοποιείται με την σύνδεση των RXC-TXC pin στα PD0 και PD1 του PORT D. Η επικοινωνία του PC με τον AVR γίνεται μέσω του PuTTY, στα 19600 baud. Παρακάτω φαίνεται πως ο AVR αναγνωρίζει τις εντολές και απαντά OK σε κάθε μία από αυτές, ενώ ταυτόχρονα γίνονται soft resets από τον watchdog timer κάθε 1 δευτερόλεπτο:



Με την είσοδο κάποιας μη αναγνωρίσιμης εντολής, ή την είσοδο παραπάνω από 8 ψηφίων για την εντολή N<X>, το πρόγραμμα επιστρέφει “ERROR”.

Καθώς δεν υπάρχει διαθέσιμη οθόνη 7-segment LED με 8 ψηφία, η έξοδος οδηγήθηκε στα LEDs του STK500. Ο ρυθμός ανανέωσης της οθόνης όμως είναι πολύ υψηλός, στο σημείο που, με τα LEDs του STK500, ο αριθμός που εμφανίζεται, εμφανίζεται μόνο για κλάσματα του δευτερολέπτου, ακόμη και αρκετά χαμηλό ρυθμό ανανέωσης από τον timer0. Έτσι δεν ήταν δυνατό να υπάρξει μία φωτογραφία για την επιβεβαίωση της εξόδου.