

Ενσωματωμένα Συστήματα Μικροεπεξεργαστών

ΗΡΥ 411

Σύστημα αυτόματου εντοπισμού ουρανίων σωμάτων με τηλεσκόπιο

Milestone 1

Εξοικείωση με τον AVR και μελέτη της περιστροφικής βάσης

Γιακουμάκης Πάρις-Παύλος

Γκιώνης Νίκος

Πορτοκαλάκης Πέτρος

27 Οκτωβρίου 2017

# Διαδικασία Εξοικείωσης

## Βασικό I/O

Η διαδικασία εξοικείωσης με τον AVRAT90S8515, την αναπτυξιακή πλακέτα STK500 και το software AVRStudio 4 έγιναν με παράλληλη μελέτη των Manuals-Datasheets και πηγών του διαδικτύου, όπως online tutorials και forums, οι οποίες παρουσιάζονται αναλυτικά στο τέλος της αναφοράς. Αρχικά υλοποιήθηκε παράδειγμα που έλεγχε μόνο την έξοδο PORTD, στην οποία είναι συνδεδεμένα τα LEDs του STK500, το οποίο στη συνέχεια εμπλουτίστηκε με τη χρήση των διακοπτών συνδεδεμένων στο PORTB, δημιουργώντας ένα πλήρες πρόγραμμα βασικού I/O. Δοκιμάστηκαν διαφορετικοί συνδυασμοί και πιο σύνθετες λειτουργίες για την σε βάθος κατανόηση του βασικού I/O του AVR.

## Μετρητές

Στη συνέχεια μελετήθηκαν οι μετρητές (timers) με δημιουργία αντίστοιχου παραδείγματος. Ο μετρητής που χρησιμοποιήθηκε ήταν ο 16-bit timer1 καθώς αυτός υποστήριζε όλες τις λειτουργίες που ήταν αναγκαίο να ελεγχθούν. Συνοπτικά, με τον καταχωρητή TCCR1B προσδιοριζόταν ο prescaler, ένας παράγοντας ο οποίος επιβραδύνει το ρολόι που χρησιμοποιεί στους υπολογισμούς του ο timer, και με τον καταχωρητή OCR1A και OCR1Bκαθορίζονταν οι τιμές εκείνες, στις οποίες μόλις φτάσει ο μετρητής, θα σηκώσει το αντίστοιχο flag. Ο καταχωρητής TIFR περιέχει τόσο το προαναφερθέν flag, όσο και flags για overflow των μετρητών. Τέλος για την ενεργοποίηση των interrupts, που θα γίνει αναφορά στη συνέχεια, χρησιμοποιείται ο καταχωρητής TIMSK. To πρόγραμμα που υλοποιήθηκε αξιοποιούσε τα flags του TIFR με χρήση if statements (δεν είχε γίνει ακόμη μελέτη των interrupts σε εκείνη τη φάση) και αφορούσε και στον υπολογισμό χρονικών διαστημάτων με τον αλγόριθμο που ακολουθεί.

Έστω ότι χρειάζεται να μετρηθεί διάστημα ενός δευτερολέπτου με μετρητή των 16 bits. Το ρολόι του AVR έχει τεθεί στο 1MHz. O prescaler έχει επιλεχθεί 1024. Πόσα ticks ρολογιού, δηλαδή ποια τιμή θα έχει πάρει ο timer όταν περάσει το ένα δευτερόλεπτο;

Ticks = ( Clk / Prescaler ) \* time = ( 1000000/1024 ) \* 1 = 976.56

Άρα όταν μετρηθούν 977 ticks θα έχει περάσει 1s.

Για εύκολο υπολογισμό των τιμών αυτών χρησιμοποιήθηκε ένα online calculator που παρουσιάζεται στις πηγές.

## Interrupts

Επόμενο βήμα ήταν η μελέτη των interrupts, η οποία έγινε σε συνδυασμό με τους timers. Συγκεκριμένα ενεργοποιώντας ταinterrupts μέσω του SREG, χρησιμοποιώντας τη συνάρτηση sei( ); της χρήσιμης γενικώς για τα interrupts βιβλιοθήκης <avr/interrupts.h>, αλλά και τα interrupts των timers συγκεκριμένα μέσω του TIMSK (τα bits OCIE1A, OCIE1B για τα interrupts σύγκρισης και το bit TOIE1 για το interrupt λόγω overflow), υλοποιήθηκαν βασικές interrupt service routines, που εκτελούσαν απλές λειτουργίες όπως η αλλαγή της κατάστασης των εξόδων στο PORTB. Χρήσιμη ήταν επίσης η μελέτη του αρχείου io8515.h, που υπάρχει στο directory του WinAVR (του GCC που χρησιμοποιείται) για τον εντοπισμό των κατάλληλων interrupt vectors, τα οποία χρησιμοποιούνται ως όρισμα στις αντίστοιχες ISRs.

## External Interrupts

Τα external Interrupts μελετήθηκαν με χρήση των διακοπτών του STK500 στις θέσεις PD2 και PD3 του PORTD που αντιστοιχούν στα INT0 και ΙΝΤ1. Βασικός καταχωρητής που ενεργοποιεί τα external interrupts είναι ο GIMSK. Επίσης χρησιμοποιήθηκε ο καταχωρητής MCUCR, του οποίου τα τέσσερα LSΒ (δύο για κάθε ένα εκ των INT0, INT1) ρυθμίζουν σε ποιο σημείο του παλμού εισόδου θα ενεργοποιείται το interrupt.Το πρόγραμμα που υλοποιήθηκε, αφού ενεργοποιήθηκαν τα external Interrupts με την εντολή sei( ); για τον καταχωρητή SREG και με την ενεργοποίηση των δύο MSB του GIMSK, εκτελούσε την απλή λειτουργία της ενεργοποίησης των LEDs με τον ένα διακόπτη και του σβησίματός τους με τον δεύτερο. Η διαδικασία αυτή γινόταν μέσα στις αντίστοιχες Interrupt Service Routines. Επίσης επιλέχθηκε να ενεργοποιούνται τα interrupts στο rising edge του παλμού εισόδου.

## Watchdog timer

Τέλος μελετήθηκε και ο Watchdog timer, του οποίου η χρήση βασίζεται στον καταχωρητή WDTCR. O Watchdog timer έχει ξεχωριστό ρολόι από τον υπόλοιπο AVR, οπότε ο χρόνος που θα μετράει πριν κάνει reset καθορίζεται από ξεχωριστό prescaler μέσω του WDTCR. Επίσης έχει ένα bit για enable του μετρητή, και ένα bit για enable του κλεισίματος του μετρητή, προκειμένου να μην απενεργοποιηθεί κατά λάθος ο Watchdog timer. Ιδιαίτερα χρήσιμη είναι και η βιβλιοθήκη <avr/wdt.h> η οποία περιέχει βασικές συναρτήσεις ελέγχου του Watchdog timer, όπως η wdt\_reset( ); η οποία μηδενίζει την τιμή του, και κατά συνέπεια ξαναρχίζει το μέτρημα μέχρι το reset, ή η wdt\_disable( ); η οποία εκτελεί τον αλγόριθμο απενεργοποίησης του Watchdog timer. Το σχετικό πρόγραμμα που υλοποιήθηκε μετέβαλε τα LEDs από την αρχική τους κατάσταση (όλα ΟΝ) μέσω απλού I/O με διακόπτη, και ύστερα από προκαθορισμένο διάστημα 1.9 seconds γινόταν reset στο κύκλωμα.

Συνδυαστικό πρόγραμμα

Για το σκοπό της παρουσίασης του πρώτου milestone, αλλά και για την καλύτερη κατανόηση όλων των παραπάνω δημιουργήθηκε τέλος ένα πρόγραμμα που συνδυάζει τις παραπάνω λειτουργίες. Συγκεκριμένα, αρχικά όλα τα LEDs ήταν ανοιχτά. Με χρήση απλού I/O μέσω διακόπτη, σβήνουν όλα τα LEDs. Με χρήση του INT0 external interrupt, ανάβει το LED2 και ξεκινάει να μετράει ένας απλός μετρητής, για ένα second. Εάν περάσει ο χρόνος αυτός, τότε, μέσω compare interrupt σβήνουν όλα τα LEDs. Με χρήση του INT1 external interrupt σταματάει ο μετρητής που ενδεχομένως έχει ανοίξει από το INT0, ανάβει το LED2 και ενεργοποιείται ο Watchdog στα δύο δευτερόλεπτα. Ο χρήστης έχει τη δυνατότητα να καθυστερήσει το reset μέσω του INT0, το οποίο περιέχει και τη λειτουργία wdt\_reset.

## Περιστροφική βάση AS20-RS485

Όσον αφορά στην περιστροφική βάση, έγινε μια πρώτη μελέτη των συνδέσεων που διαθέτει και του κώδικα χειρισμού της. Η βάση επικοινωνεί μεσω RS485, επομένως αξιοποιώντας τον UART του AVR θα γίνει η επικοινωνία μικροελεγκτή-βάσης. OUART χρησιμοποιεί τα PD1 για transmit και PD0 για receive, ενώ η βάση έχει απλό RS485 δύο αγωγών. Οι κώδικες χειρισμού έχουν δημιουργηθεί για νεότερη εκδοση του AVR (πιθανώς ATmega), οπότε έγινε μια προσαρμογή αυτών στους καταχωρητές και τις προδιαγραφές του AT90S8515.

## Πρωτόκολλο SPI

Το πρωτόκολλο επικοινωνίας SPI χρησιμοποιείται μεταξύ μικροελεγκτών ή μεταξύ μικροελεγκτή και περιφερειακών. Για την υλοποίησή του συνδέονται ο ένας 8-bit καταχωρητής ολίσθησης του κυρίως μικροελεγκτή (master) με τον 8-bit καταχωρητή ολίσθησης του δευτερεύοντος μικροελεγκτή ή περιφερειακού (slave). Η σύνδεση γίνεται ως εξής: To PB7 (SCK) του master είναι η έξοδος-ρολόι που δίνεται στο αντίστοιχο pin του slave. Ο master γράφει τα δεδομένα στον SPI data register, τα οποία μεταφέρονται μέσω της σύνδεσης των PB5(MOSI) των master και slave. Η αντίθετη μεταφορά δεδομένων γίνεται μέσω της σύνδεσης των PB6(MISO). Επιπλέον υπάρχουν οι SPI control register, για τον έλεγχο της σύνδεσης, καθώς και ο SPI Status Register για τα flags. Συνολικά τα PINS που χρειάζεται για όλες τις λειτουργίες του το SPI είναι τα PB4, PB5, PB6 και PB7 του PORTD.

## Πρωτόκολλο RS232

Η σειριακή θύρα του STK500 “RS232 Spare” χρησιμοποιείται συνδέοντας τα σήματα Receive (RXD) και transmit(TXD) του πρωτοκόλλου RS232 με τα PD0 και PD1 Pins του UART. Τα υπόλοιπα σήματα του RS232 αφορούν στον έλεγχο της επικοινωνίας μέσω της σύνδεσης. Αυτάείναιτα Data carrier detect (DCD), Data terminal ready (DTR), Data set ready (DSR), Request to send (RTS), Clear to send (CTS), Ring indicator (RI). Επίσης υπάρχει και το Signalground (GND). Επομένως η σύνδεση ενός περιφερειακού στην “RS232 Spare” θύρα δεσμεύει τα PD0 και PD1 του PORTD.

# Βιβλιογραφικές Αναφορές

## Manuals και Datasheets από το επίσημο site της Atmel:

* UserGuide για το STK500

<http://www.atmel.com/images/doc1925.pdf>

* Manual για τον AT90S8515

<http://www.atmel.com/Images/DOC0841.PDF>

## Διαδικτυακές πηγέςγια AVR:

* Σειρά tutorials για τη χρήση του AVR και τον προγραμματισμό του σε C

<https://www.youtube.com/watch?v=BKg2hoD89jY&list=PLRuRKN7_FVgvhdj6JmelCihl6jmKHc_Xt>

* Σειρά tutorials για τη χρήση του AVR και τον προγραμματισμό του σε C

<https://www.youtube.com/watch?v=9ADxPRjZI4Q&list=PLA6BB228B08B03EDD>

* Online Calculator για AVR timer

<http://eleccelerator.com/avr-timer-calculator/>

## Πηγές για την περιστροφική βάση AS20-RS485

* Κώδικες χειρισμού από εργαστήριο
* Manual με τις βασικές συνδέσεις από εργαστήριο
* Χρήσιμα links για το RS-485

<http://www.windmill.co.uk/rs485.html>

<https://en.wikipedia.org/wiki/RS-485>

# Παράρτημα

Ο κώδικας που έχει φορτωθεί στον AT90S8515 για την παρουσίαση του πρώτου milestone είναι ο ακόλουθος:

#define F\_CPU 1228800UL

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <avr/wdt.h>

ISR(INT0\_vect){

PORTB=0xFB; //OPEN LED 2

TCCR1B = (1<<CS12) | (1<<CS10)|(1<<CTC1); //Prescaler 1024, Start Counter

wdt\_reset(); //and clear on compare

}

ISR(INT1\_vect){

TCCR1B = 0; //Close simple Counter

PORTB=0xFB; //Open Led 2

WDTCR=0x0F; //Enable watchdog, 2 second reset. Can be delayed stopped with Int0!

}

ISR(TIMER1\_COMPA\_vect){

PORTB =0xFF; //If INT0 is pressed, after 1 second the Led closes

TCCR1B = 0; //Stop Counter

}

void switchPolling(){

if((~PIND) & 0b00000001==1 ){

PORTB=0xFF;

}

}

void init(){

DDRB = 0xFF; // Configure port B as output

DDRD = 0x00;

sei();

TIMSK = (1<<OCIE1A) | (1<<OCIE1B);

GIMSK=0b11000000;

OCR1A = 1200; //1 second

MCUCR=0x0F; //Interrupt at rising edge

}

int main(){

init();

while(1){

switchPolling();

}

}

Ο προσαρμοσμένος κώδικας χειρισμού της βάσης στα δεδομένα του AT90S8515 είναι ο εξής:

* *Συνάρτηση που βάζει το byte προς μετάδοση στον UART data register, περιμένοντας να αδειάσει:*

void UART\_Transmit(unsigned char data)

{

// Wait for empty transmit buffer

while (!( USR & (1<<UDRE)));

// Put data into buffer, sends the data

UDR = data;

}

* *Συνάρτηση που αξιοποιεί την UART\_Transmit για την μετάδοση 8 byte*:

void gimbal\_transmit(int byte1,int byte2,int byte3,int byte4,int byte5,int byte6,int byte7)

{

UART\_Transmit(byte1);

UART\_Transmit(byte2);

UART\_Transmit(byte3);

UART\_Transmit(byte4);

UART\_Transmit(byte5);

UART\_Transmit(byte6);

UART\_Transmit(byte7);

}

* *Οι συναρτήσεις left( ), right ( ) κ.ο.κ. δεν έχουν μεταβληθεί. Ενδεικτικά η left():*

void left(void)

{

gimbal\_transmit(0xFF,0x01,0x00,0x04,0x20,0x00,0x25);

}

* *Παράδειγμα χρήσης της βάσης. Ενεργοποιείται ο UART και η βάση στρίβει αριστερά:*

int main(){

UCR = (1<<RXEN) | (1<<TXEN);

UBRR= 12;

while(1){

left( );

}

}

Σημείωση: Οι κώδικες της βάσης δεν έχουν ελεγχθεί, διότι δεν έχουμε ακόμη προμηθευτεί το κατάλληλο τροφοδοτικό για την βάση. Είμαστε σε επικοινωνία με τον κ. Κιμιωνή για το ζήτημα.