



ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Εργαστήριο Μικροελεγκτές - Ενσωματωμένα Συστήματα

2^{ος} ΚΥΚΛΟΣ

Μικροελεγκτές Atmel – AVR

ΑΣΚΗΣΗ 4^η

Χειρισμός οθόνης LCD - Σειριακή Επικοινωνία (UART)

**Γ. Καλτσάς
Καθηγητής**

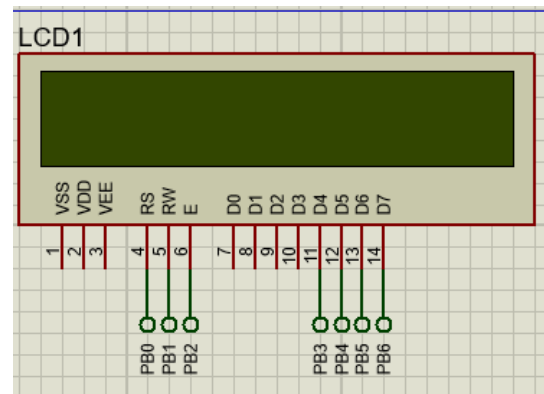
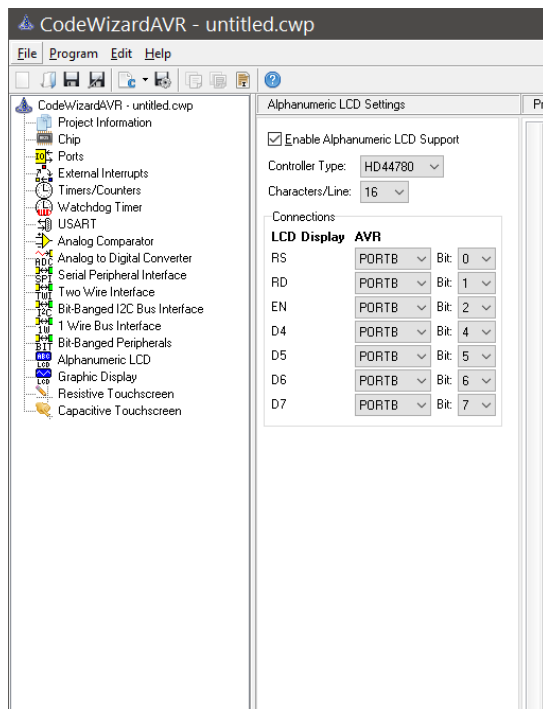
Αθήνα 2019

LCD Οθόνη

Οι συναρτήσεις που θα δούμε στη συνέχεια επιτρέπουν τη διαχείριση LCD οθόνης που χρησιμοποιεί το ολοκληρωμένο Hitachi HD44780 ή κάποιο άλλο παρόμοιο. Τα πρωτότυπά τους βρίσκονται στο αρχείο επικεφαλίδας **alcd.h**, το οποίο θα πρέπει να δηλωθεί με τον τελεστή **#include** πριν τη χρήση των συγκεκριμένων συναρτήσεων.

Οι μορφές του LCD που υποστηρίζονται από το αρχείο **alcd.h** είναι: 1x8, 2x12, 3x12, 1x16, 2x16, 2x20, 4x20, 2x24 και 2x40 χαρακτήρες, όπου ο πρώτος αριθμός δηλώνει το πλήθος των γραμμών και ο δεύτερος το πλήθος των στηλών.

Ομοίως με τα υπόλοιπα περιφερειακά, μπορούμε να χρησιμοποιήσουμε τον CodeWizard του CodeVisionAVR για να αρχικοποιήσουμε κατάλληλα την οθόνη LCD, όπως φαίνεται παρακάτω στο παράδειγμα, που ορίζουμε 16 χαρακτήρες ανά γραμμή και τα αντίστοιχα pins στους ακροδέκτες της PORTB. Παρατηρήστε την ονοματολογία των pins σε σχέση με το σχηματικό του PROTEUS.



Συναρτήσεις LCD υψηλού επιπέδου

Για να χρησιμοποιήσουμε την οθόνη LCD υπάρχουν (ενδεικτικά) οι παρακάτω συναρτήσεις:

```
void lcd_init(unsigned char lcd_columns)
```

Αρχικοποίηση της διάταξης LCD. Πραγματοποιείται καθαρισμός της οθόνης και τοποθέτηση της θέσης εκτύπωσης στην αρχή (γραμμή=0 και στήλη=0). Ο αριθμός των στηλών του LCD θα πρέπει να καθοριστεί (π.χ. 20). Δεν προβάλλεται δρομέας.

Η συγκεκριμένη συνάρτηση πρέπει κλιθεί πριν από όλες τις επόμενες συναρτήσεις υψηλού επιπέδου.

void lcd_clear(void)

Καθαρίζει την οθόνη και τοποθετεί την θέση εκτύπωσης στην αρχή (γραμμή=0 και στήλη=0).

void lcd_gotoxy(unsigned char x, unsigned char y)

Θέτει την τρέχουσα θέση προβολής στην στήλη x και γραμμή y. Ο αριθμός των γραμμών και των στηλών ξεκινά από το 0.

void lcd_putchar(char c)

Προβάλλει τον χαρακτήρα c στην τρέχουσα θέση προβολής.

void lcd_puts(char *str)

Προβάλλει στην τρέχουσα θέση προβολής την γραμματοσειρά str, η οποία βρίσκεται στην SRAM.

void lcd_putsf(char flash *str)

Προβάλλει στην τρέχουσα θέση προβολής την γραμματοσειρά str, η οποία βρίσκεται στην FLASH.

Παραδείγματα χρήσης της LCD

Ο παρακάτω κώδικας αναβοσβήνει σε μια LCD οθόνη 16x2 ένα σταθερό μήνυμα με περίοδο 4 δευτερόλεπτα.

```
#include <mega2560.h>
#include <alcd.h>
#include <delay.h>

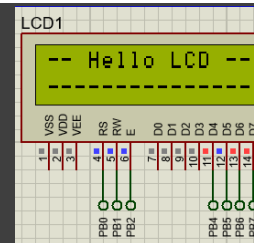
void main(void)
{
    // Crystal Oscillator division factor: 1
    #pragma optimize-
    CLKPR=(1<<CLKPCE);
    CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
    #ifdef _OPTIMIZE_SIZE_
    #pragma optimize+
    #endif

    lcd_init(16); /* 16 columns */

    while (1)
    {
        lcd_gotoxy(0,0); /* go to the first line, first character */

        delay_ms(2000); /* display the message */
        lcd_putsf("-- Hello LCD --\n-----");

        delay_ms(2000); /* and clear */
        lcd_clear();
    }
}
```



Ο κώδικας που ακολουθεί εμφανίζει σε μια οθόνη LCD 2x20 το μήνυμα “Πατήσατε το 1ο πλήκτρο” ή το μήνυμα “Πατήσατε το 2ο πλήκτρο”, εάν πατηθεί αντίστοιχα το 1ο ή το 2ο πλήκτρο της πόρτας D. Τα μηνύματα παραμένουν για 3 δευτερόλεπτα.

```
#include <mega2560.h>
#include <alcd.h>
#include <delay.h>

#define button_0 PIND.0
#define button_1 PIND.1

void main(void)
{
    // Crystal Oscillator division factor: 1
    #pragma optsize-
    CLKPR=(1<<CLKPCE);
    CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
    #ifdef _OPTIMIZE_SIZE_
    #pragma optsize+
    #endif

    DDRD = 0x00;
    PORTD = 0xFF;

    lcd_init(20); /* 20 columns */

    while (1)
    {
        if (button_0 == 0){
            lcd_putsf("You pressed the FIRST button"); /* notice putsf,
saving SRAM */
            delay_ms(3000);
            lcd_clear();
        }

        if (button_1 == 0){
            lcd_putsf("You pressed the SECOND button");
            delay_ms(3000);
            lcd_clear();
        }
    }
}
```

Ο κώδικας που ακολουθεί τοποθετεί αστερίσκους σε μια οθόνη LCD 2x20 ξεκινώντας από την πρώτη στήλη της πρώτης γραμμής και καταλήγοντας στην εικοστή στήλη της δεύτερης γραμμής. Στη συνέχεια τοποθετεί κενά στην LCD οθόνη ξεκινώντας από την εικοστή στήλη της δεύτερης γραμμής και καταλήγοντας στην πρώτη στήλη της πρώτης γραμμής.

```
#include <mega2560.h>
#include <alcd.h>
#include <delay.h>

#define button_0 PIND.0
```

```
#define button_1 PIND.1

void main(void)
{

    int p,i;

    // Crystal Oscillator division factor: 1
    #pragma optsize-
    CLKPR=(1<<CLKPCE);
    CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
    #ifdef _OPTIMIZE_SIZE_
    #pragma optsize+
    #endif

    lcd_init(20);

    while (1)
    {

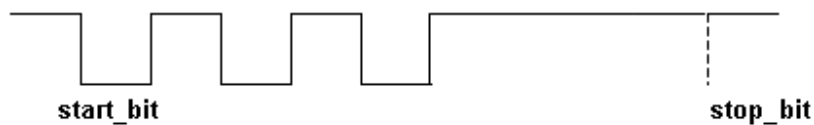
        for(p=0; p<=1 ;p++) {
            for(i=0; i<20; i++) {
                lcd_gotoxy(i,p);
                lcd_putsf("*");
                delay_ms(200);
            }
        }

        for(p=1 ;p>=0; p--) {
            for(i=19; i>=0; i--) {
                lcd_gotoxy(i,p);
                lcd_putsf(" ");
                delay_ms(200);
            }
        }

    }
}
```

Ασύγχρονη Επικοινωνία

H Universal Asynchronous Receiver Transmitter (UART) επικοινωνία σχετίζεται με την ασύγχρονη σειριακή επικοινωνία μεταξύ δύο συστημάτων. Η επικοινωνία μέσω UART απαιτεί να καθοριστεί ο ρυθμός αποστολής δεδομένων και ο αριθμός των bit δεδομένων (5-8 bits) που θα μεταδίδονται μεταξύ των start και stop bits. Η έναρξη αποστολής των δεδομένων σηματοδοτείται με την αλλαγή της τιμής του start_bit. Συγκεκριμένα, η τιμή του αλλάζει από 1 σε 0. Ο τερματισμός των δεδομένων σηματοδοτείται με το stop_bit, του οποίου η τιμή είναι 1. Για παράδειγμα, αν διαμορφώσουμε τον ρυθμό μετάδοσης σε 9600 bps και 8-bit δεδομένων ο χρόνος μετάδοσης για κάθε bit είναι $1/9600 = \sim 100 \mu s$. Τα 8-bit δεδομένα κωδικοποιούνται ανάλογα με την τιμή τους. Για παράδειγμα, το byte 10101111 κωδικοποιείται ως εξής:



Πριν από το `stop_bit`, ο πομπός μπορεί προαιρετικά να μεταδώσει ένα bit ισοτιμίας (`parity_bit`), το οποίο χρησιμοποιείται από τον παραλήπτη για έλεγχο σφαλμάτων στη μετάδοση. Η τιμή του καθορίζεται από το είδος της ισοτιμίας, άρτιας ή περιττής. Στο προηγούμενο παράδειγμα, σε περίπτωση άρτιας ισοτιμίας η τιμή του bit ισοτιμίας είναι 0, έτσι ώστε ο συνολικός αριθμός των “1” (μαζί με το bit ισοτιμίας) να είναι άρτιος, ενώ σε περίπτωση περιττής ισοτιμίας η τιμή του είναι 1, έτσι ώστε ο συνολικός αριθμός των “1” (μαζί με το bit ισοτιμίας) να είναι περιττός.

Σειριακή Επικοινωνία στον AVR

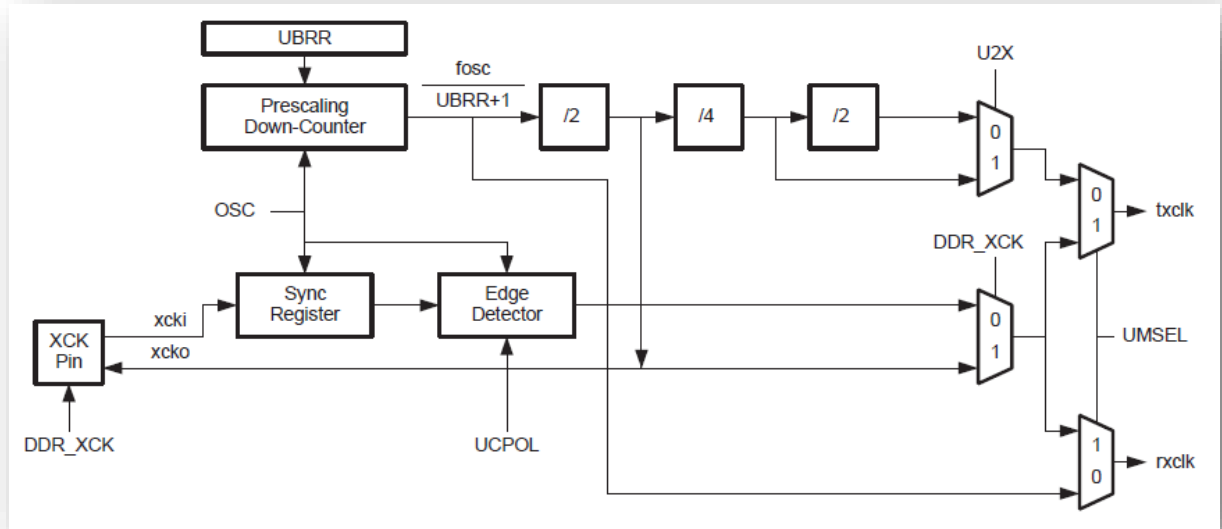
Ο ATmega2560 έχει διαθέσιμες 4 σειριακές θύρες, την UART0, την UART1, την UART2 και την UART3. Για κάθε μια που θέλει ο χρήστης να χρησιμοποιήσει, πρέπει να ρυθμιστεί αρχικά ο ρυθμός μετάδοσης συμβόλων (baudrate) με τον καταχωρητή UBRRn (Baud Rate Generator).

BAUD Baud rate (in bits per second, bps)

 f_{osc} System Oscillator clock frequency

UBRRn Contents of the UBRRHn and UBRLn Registers, (0-4095)

Στο παρακάτω σχήμα φαίνεται η συνδεσμολογία του συστήματος για να παραχθεί ο απαραίτητος ρυθμός από το ρολόι του συστήματος προς τα συστήματα χρονισμών των γραμμών αποστολής (Tx) και λήψης (Rx).



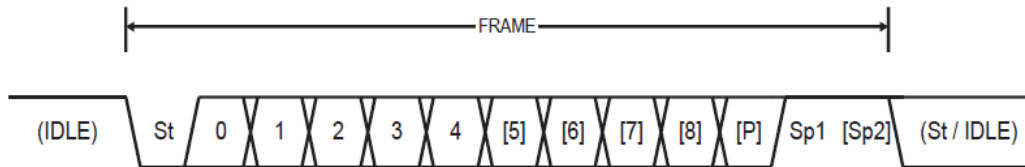
Η σειριακή επικοινωνία UART μπορεί να λειτουργήσει με τρεις διαφορετικούς τρόπους στους μικροελεγκτές AVR: a) Asynchronous mode ($U2X_n = 0$), b) Asynchronous Double Speed mode ($U2X_n = 1$) και c) Synchronous Master Mode

Στον παρακάτω πίνακα συνοψίζονται οι σχέσεις που παρέχουν τον ρυθμό μετάδοσης (baudrate) σε σχέση με το περιεχόμενο του καταχωρητή UBRR σε κάθε περίπτωση:

Operating Mode	Equation for Calculating Baud Rate ⁽¹⁾	Equation for Calculating UBRR Value
Asynchronous Normal mode ($U2X_n = 0$)	$BAUD = \frac{f_{osc}}{16(UBRR_n + 1)}$	$UBRR_n = \frac{f_{osc}}{16BAUD} - 1$
Asynchronous Double Speed mode ($U2X_n = 1$)	$BAUD = \frac{f_{osc}}{8(UBRR_n + 1)}$	$UBRR_n = \frac{f_{osc}}{8BAUD} - 1$
Synchronous Master mode	$BAUD = \frac{f_{osc}}{2(UBRR_n + 1)}$	$UBRR_n = \frac{f_{osc}}{2BAUD} - 1$

Είναι επίσης απαραίτητο να ρυθμιστούν κι άλλες παράμετροι, όπως το start bit, τα data bits, το parity και τα stop bits. Μια τυπική ρύθμιση που χρησιμοποιούν πολλά συστήματα είναι: 9600 8-N-1 σε asynchronous mode (ο συμβολισμός παραπέμπει σε 9600 baudrate, 8 data bits, No parity bit και 1 stop bit).

Ένα frame μηνύματος UART μοιάζει με το παρακάτω:



St Start bit, always low.

(n) Data bits (0 to 8).

P Parity bit. Can be odd or even.

Sp Stop bit, always high.

IDLE No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

Καταχωρητές – Ρυθμίσεις UART

Bit	7	6	5	4	3	2	1	0	
	RXCn	TXCn	UDREN	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

UCSRnA (Control and Status Register A)

- Bit 7 – RXCn: USART Receive Complete
- Bit 6 – TXCn: USART Transmit Complete
- Bit 5 – UDREN: USART Data Register Empty
- Bit 4 – FEn: Frame Error
- Bit 3 – DORn: Data OverRun
- Bit 2 – UPEn: USART Parity Error
- Bit 1 – U2Xn: Double the USART Transmission Speed
- Bit 0 – MPCMn: Multi-processor Communication Mode

UCSRnB (Control and Status Register B)

Bit	7	6	5	4	3	2	1	0	
	RXCIE_n	TXCIE_n	UDRIE_n	RXEN_n	TXEN_n	UCSZ_{n2}	RXB8_n	TXB8_n	UCSR_{nB}
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 – RXCIE_n: RX Complete Interrupt Enable n
- Bit 6 – TXCIE_n: TX Complete Interrupt Enable n
- Bit 5 – UDRIE_n: USART Data Register Empty Interrupt Enable n
- Bit 4 – RXEN_n: Receiver Enable n
- Bit 3 – TXEN_n: Transmitter Enable n
- Bit 2 – UCSZ_{n2}: Character Size n
- Bit 1 – RXB8_n: Receive Data Bit 8 n
- Bit 0 – TXB8_n: Transmit Data Bit 8 n

UCSRnC (Control and Status Register C)

Bit	7	6	5	4	3	2	1	0	
	UMSEL_{n1}	UMSEL_{n0}	UPM_{n1}	UPM_{n0}	USBS_n	UCSZ_{n1}	UCSZ_{n0}	UCPOL_n	UCSR_{nC}
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

- Bits 7:6 – UMSEL_{n1:0} USART Mode Select
- Bits 5:4 – UPM_{n1:0}: Parity Mode
- Bit 3 – USBS_n: Stop Bit Select
- Bit 2:1 – UCSZ_{n1:0}: Character Size
- Bit 0 – UCPOL_n: Clock Polarity

UMSEL _{n1}	UMSEL _{n0}	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM) ⁽¹⁾

UPM _{n1}	UPM _{n0}	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

UCSZ _{n2}	UCSZ _{n1}	UCSZ _{n0}	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

USBS _n	Stop Bit(s)
0	1-bit
1	2-bit

Παράδειγμα αρχικοποίησης UART (Asynchronous Normal mode): $UBRR_n = \frac{f_{osc}}{16BAUD} - 1$

```
// AVR Core Clock frequency: 8,000000 MHz
// USART0 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART0 Receiver: On
// USART0 Transmitter: On
// USART0 Mode: Asynchronous
// USART0 Baud Rate: 9600

UCSR0A = (0<<RXC0) | (0<<TXC0) | (0<<UDRE0) | (0<<FE0) | (0<<DOR0) | (0<<UPE0) | (0<<U2X0) |
(0<<MPCM0);

UCSR0B = (0<<RXCIE0) | (0<<TXCIE0) | (0<<UDRIE0) | (1<<RXEN0) | (1<<TXEN0) | (0<<UCSZ02) | (0<<RXB80) |
(0<<TXB80);

UCSR0C = (0<<UMSEL01) | (0<<UMSEL00) | (0<<UPM01) | (0<<UPM00) | (0<<USBS0) | (1<<UCSZ01) | (1<<UCSZ00) |
(0<<UCPOL0);

UBRR0H = 0x00;
UBRR0L = 0x33; // (Dec: 51)
```

Παράδειγμα αρχικοποίησης UART(Asynchronous Double Speed mode): $UBRR_n = \frac{f_{osc}}{8BAUD} - 1$

```
// AVR Core Clock frequency: 8,000000 MHz
// USART2 initialization
// Communication Parameters: 8 Data, 2 Stop, No Parity
// USART2 Receiver: On
// USART2 Transmitter: On
// USART2 Mode: Asynchronous
// USART2 Baud Rate: 14400 (Double Speed Mode)

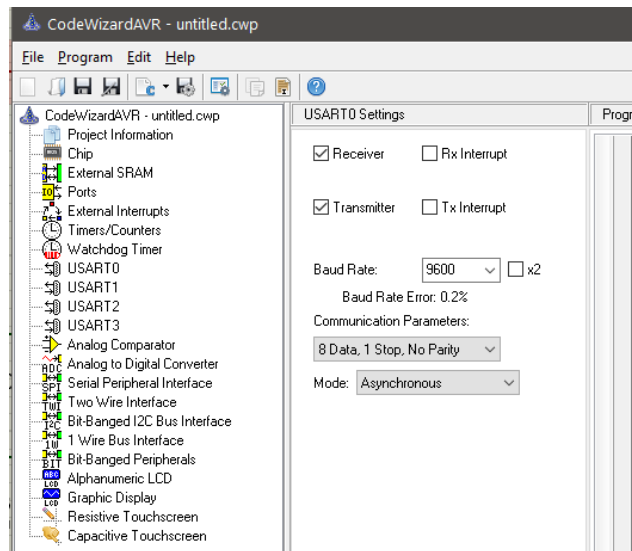
UCSR2A = (0<<RXC2) | (0<<TXC2) | (0<<UDRE2) | (0<<FE2) | (0<<DOR2) | (0<<UPE2) | (1<<U2X2) |
(0<<MPCM2);

UCSR2B = (0<<RXCIE2) | (0<<TXCIE2) | (0<<UDRIE2) | (1<<RXEN2) | (1<<TXEN2) | (0<<UCSZ22) | (0<<RXB82) |
(0<<TXB82);

UCSR2C = (0<<UMSEL21) | (0<<UMSEL20) | (0<<UPM21) | (0<<UPM20) | (1<<USBS2) | (1<<UCSZ21) | (1<<UCSZ20) |
(0<<UCPOL2);

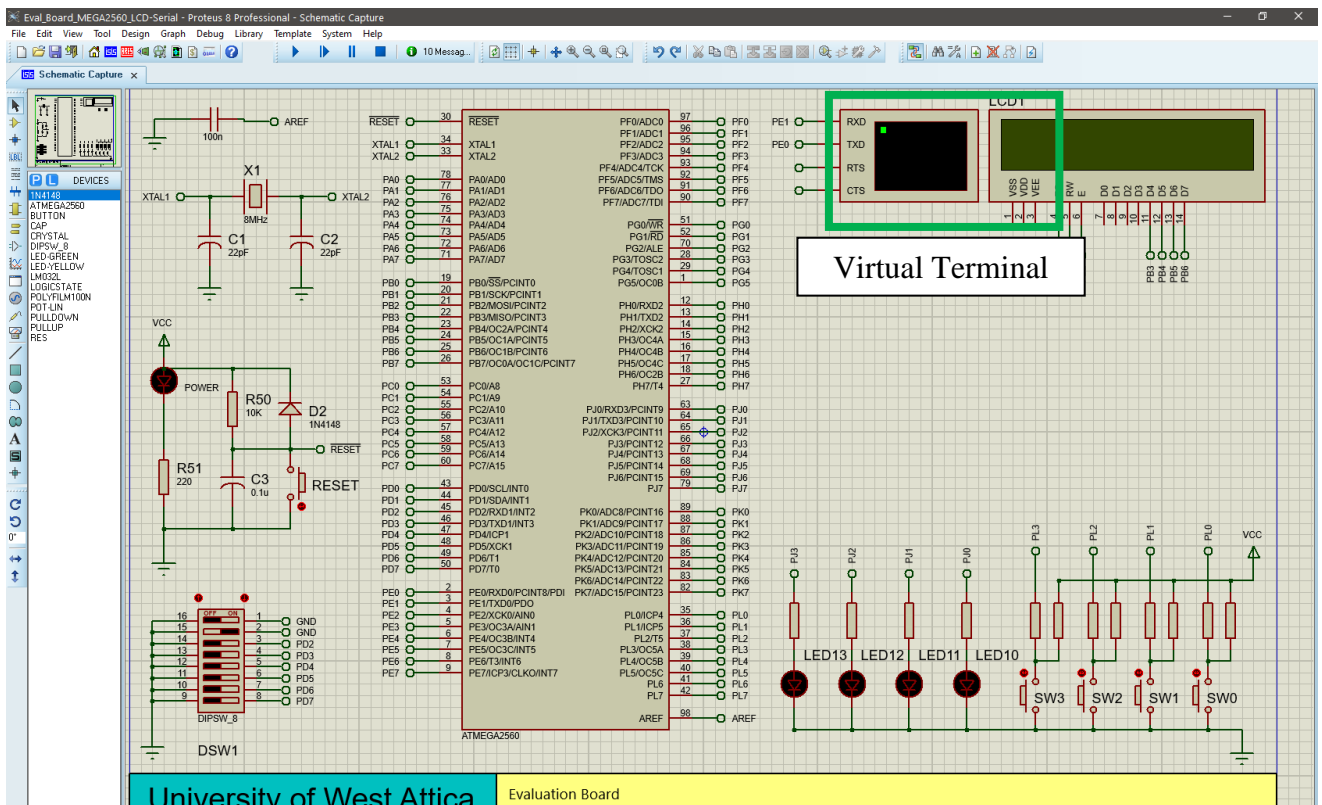
UBRR2H=0x00;
UBRR2L=0x44; // (Dec: 68)
```

Για την υλοποίηση της εργαστηριακής άσκησης, θα χρησιμοποιήσουμε τα εργαλεία CodeVisionAVR-CodeWizardAVR και PROTEUS. Αρχικά θα πρέπει να ρυθμίσουμε το project ώστε να ενεργοποιείται η επιθυμητή hardware serial port. Στον ATmega2560 βρίσκεται στα pins 0 (RX) & 1(TX) της PORTE. Παρακάτω, φαίνεται η ρύθμιση της UART στο CodeWizardAVR.



Σε όλα τα αρχικά παραδείγματα, δίνεται ο απαραίτητος κώδικας που ρυθμίζει τους καταχωρητές της UART κι έτσι ο προγραμματιστής μπορεί να πάρει τον κώδικα της αρχικοποίησης έτοιμο. Σε οποιαδήποτε άλλη περίπτωση, αν ο προγραμματιστής χρειαστεί να αρχικοποιήσει κάποια σειριακή και δεν έχει πρόσβαση στις πληροφορίες για τους καταχωρητές, μπορεί να χρησιμοποιήσει τον CodeWizard όπως φαίνεται στην εικόνα παραπάνω.

Στην συνέχεια, μέσω του PROTEUS, μπορούμε να τρέξουμε τον εκάστοτε κώδικα. Θα χρησιμοποιήσουμε το **template Eval_Board_MEGA2560_LCD-Serial**. Το template αυτό έχει έτοιμη την απαραίτητη συνδεσμολογία και ένα virtual terminal, ένα εικονικό δηλαδή παράθυρο που συνδέει απ'ευθείας το πληκτρολόγιο του υπολογιστή του χρήστη με την UART του μικροελεγκτή.



Αν για κάποιον λόγο δεν εμφανιστεί το παράθυρο του virtual terminal, είναι διαθέσιμο αφού ο χρήστης τρέξει την προσομοίωση μέσω του Debug > Virtual Terminal.

Συναρτήσεις Τύπου Χαρακτήρων (char)

Για να χρησιμοποιηθούν οι παρακάτω συναρτήσεις, είναι απαραίτητο να δηλωθεί το αρχείο που ορίζονται τα πρωτότυπά τους (με την εντολή `#include <ctype.h>`). Με τις συναρτήσεις αυτές, ο προγραμματιστής μπορεί να πραγματοποιεί ελέγχους π.χ. για το αν οι χαρακτήρες που λαμβάνει το σύστημα από ένα εξωτερικό περιφερειακό (UART) είναι αναγνώσιμο κείμενο, αποτελούν λέξεις, είναι αλφαριθμητικοί συνδυασμοί κ.α..

unsigned char isascii(char c)

Επιστρέφει την τιμή 1 αν η μεταβλητή “c” είναι χαρακτήρας ASCII (0...127)

unsigned char iscntrl(char c)

Επιστρέφει την τιμή 1, εάν η μεταβλητή “c” είναι χαρακτήρας ελέγχου (control character) (0...31 ή 127).

unsigned char isdigit(char c)

Επιστρέφει την τιμή 1, εάν η μεταβλητή “c” είναι δεκαδικό ψηφίο.

unsigned char isspace(char c)

Επιστρέφει την τιμή 1, εάν η μεταβλητή “c” είναι ένας εκ των χαρακτήρων: space, CR, HT.

char toascii(char c)

Επιστρέφει το ASCII ισοδύναμο της μεταβλητής “c”.

Προκαθορισμένες Συναρτήσεις Εισόδου/Εξόδου (Standard C I/O Functions)

Για να χρησιμοποιηθούν οι παρακάτω συναρτήσεις, είναι απαραίτητο να δηλωθεί το αρχείο που ορίζονται τα πρωτότυπά τους (με την εντολή `#include <stdio.h>`). Πριν την χρήση αυτών των συναρτήσεων θα πρέπει να έχουν εκτελεστεί τα παρακάτω βήματα:

- ✓ Αρχικοποίηση του ρυθμού μετάδοσης δεδομένων (baud rate) της ασύγχρονης πόρτας (UART).
- ✓ Ενεργοποίηση του ασύγχρονου εκπομπού (UART transmitter).
- ✓ Ενεργοποίηση του ασύγχρονου δέκτη (UART receiver).

Οι σημαντικότερες συναρτήσεις αυτής της κατηγορίας είναι:

char getchar(void)

Επιστρέφει τον χαρακτήρα που λαμβάνεται από την ασύγχρονη πόρτα (UART), χρησιμοποιώντας συνεχή παρακολούθηση (polling). Ουσιαστικά περιμένει να πιεστεί κάποιος χαρακτήρας του πληκτρολογίου και επιστέφει το πλήκτρο που πατήθηκε σε ASCII μορφή.

void putchar(char c)

Αποστέλλει τον χαρακτήρα της παραμέτρου “c” στην ασύγχρονη πόρτα (UART), χρησιμοποιώντας συνεχή παρακολούθηση (polling).

Βασικό Πρόγραμμα Σειριακής Επικοινωνίας

$$UBRR_n = \frac{f_{osc}}{16BAUD} - 1$$

```
#include <mega2560.h>
#include <stdio.h>

/* quartz crystal frequency [Hz] */
#define xtal 8000000L

/* Baud rate */
#define baud 9600

void main(void)
{
    char k;
    // Crystal Oscillator division factor: 1
    #pragma optsize-
    CLKPR=(1<<CLKPCE);
    CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
    #ifdef _OPTIMIZE_SIZE_
    #pragma optsize+
    #endif

    /* ...
       init other peripherals
       ...
    */

    // USART0 initialization
    // Communication Parameters: 8 Data, 1 Stop, No Parity
    // USART0 Receiver: On
    // USART0 Transmitter: On
    // USART0 Mode: Asynchronous
    // USART0 Baud Rate: 9600
    UCSR0A=0x00;
    UCSR0B=0x18;
    UCSR0C=0x06;

    /* similarly:
    UCSR0A = (0<<RXCE0) | (0<<TXCE0) | (0<<UDRE0) | (0<<FE0) | (0<<DOR0) |
    (0<<UPE0) | (0<<U2X0) | (0<<MPCM0);

    UCSR0B=(0<<RXIE0) | (0<<TXIE0) | (0<<UDRIE0) | (1<<RXEN0) | (1<<TXEN0) |
    (0<<UCSZ02) | (0<<RXB80) | (0<<TXB80);
```

```
UCSR0C=(0<<UMSEL01) | (0<<UMSEL00) | (0<<UPM01) | (0<<UPM00) | (0<<USBS0) |
(1<<UCSZ01) | (1<<UCSZ00) | (0<<UCPOL0);
*/

UBRR0H=(xtal/16/ baud-1) >> 8; // same as 0x00
UBRR0L=(xtal/16/ baud-1) & 0xFF; // same as 0x33 (dec: 51)

while (1)
{
    k = getchar(); /* Λήψη χαρακτήρα */
    putchar(k);    /* Αποστολή του ίδιου χαρακτήρα
                    (echo) */
}
}
```

void puts(char *str)

Εκτυπώνει (χρησιμοποιώντας την συνάρτηση **putchar**) τη γραμματοσειρά **str**, η οποία είναι αποθηκευμένη στην SRAM, ακολουθούμενη από το χαρακτήρα νέας γραμμής

void putsf(char flash *str)

Εκτυπώνει (χρησιμοποιώντας την συνάρτηση **putchar**) τη γραμματοσειρά **str**, η οποία είναι αποθηκευμένη στην FLASH, ακολουθούμενη από το χαρακτήρα νέας γραμμής.

Παράδειγμα: Δημιουργία Menu. Να ζητείται να εισαχθεί ένας αριθμός.

- Εάν ο αριθμός είναι το «1» να δημιουργείται down counter modulo 16 στην πόρτα B.
- Εάν ο αριθμός είναι το «2» να δημιουργείται ripple counter στην πόρτα B.
- Σε οποιαδήποτε άλλη περίπτωση να προβάλλεται στο μήνυμα “insert number”.

```
#include <mega2560.h>
#include <stdio.h>
#include <delay.h>
#include <ctype.h>

#define xtal 8000000L
#define baud 9600

void main(void)
{
    int i;
    char k;

    // Crystal Oscillator division factor: 1
    #pragma optsize-
```

```

CLKPR=(1<<CLKPCE);
CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
#ifdef _OPTIMIZE_SIZE_
#pragma optsize+
#endif

DDRB=0xFF;
UCSR0A=0x00;
UCSR0B=0x18;
UCSR0C=0x06;

UBRR0H=(xtal/16/115200-1) >> 8;
UBRR0L=(xtal/16/115200-1) & 0xFF;

while (1)
{
    do { /* wait for number as input */
        putsf("\n\r Give number");
        k=getchar();
    } while (isdigit(k)!= 1);

    /* we got number, because isdigit(k) returned 1 (k is a digit)
       and we left while */

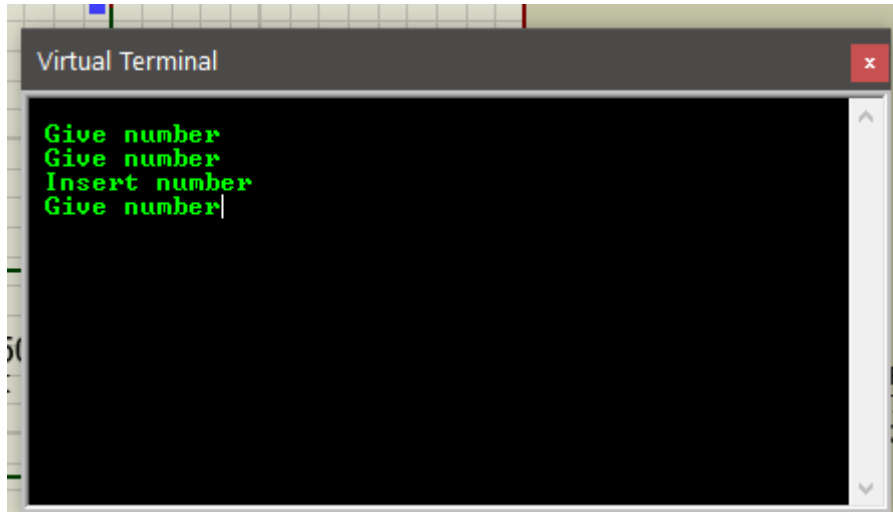
    /* is k 1? */
    if (k == '1'){
        /* down counter mod16 */
        for(i=16; i>=0; i--){
            PORTB=i;
            delay_ms(200);
        }
    }

    /* or is k 2? */
    else if (k == '2') {
        /* ripple counter */
        for (i=1;i<=128;i*=2) {
            PORTB=i;
            delay_ms(200);
        }
    }

    /* or is k any other number? */
    else {
        putsf("\n\r Insert number ");
    }
} // while
} // main

```

Παρακάτω φαίνονται στιγμιότυπα από την λειτουργία του menu στο PROTEUS. Να σημειωθεί ότι ο χρήστης δεν έχει local echo των πλήκτρων που πατά, δηλαδή δεν εμφανίζονται τα γράμματα που πατά στο πληκτρολόγιο. Αποστέλλονται κατευθείαν στην σειριακή.



```
void printf(char flash *fmtstr [, arg1, arg2, ...])
```

Η συνάρτηση συμπεριφέρεται όμοια με την αντίστοιχη συνάρτηση της ANSI C. Εκτυπώνει κείμενο (χρησιμοποιώντας την συνάρτηση **putchar**) μορφοποιημένο, με βάση τους προσδιοριστές μορφοποίησης που εμπεριέχονται στην γραμματοσειρά **fmtstr**. Η γραμματοσειρά **fmtstr** θεωρείται σταθερά και πρέπει να βρίσκεται στην μνήμη FLASH.

Οι **βασικοί προσδιοριστές** μορφοποίησης που υποστηρίζονται είναι λιγότεροι από αυτούς της αντίστοιχης συνάρτησης της ANSI C και είναι οι ακόλουθοι:

%c:	εκτυπώνει την αντίστοιχη μεταβλητή σαν χαρακτήρα ASCII
%d, %i:	εκτυπώνει την αντίστοιχη μεταβλητή σαν προσημασμένο ακέραιο
%u:	εκτυπώνει την αντίστοιχη μεταβλητή σαν μη προσημασμένο ακέραιο
%e:	εκτυπώνει την αντίστοιχη μεταβλητή σαν αριθμό κινητής υποδιαστολής με την μορφή [-]d.ddd e[±]dd
%f:	εκτυπώνει την αντίστοιχη μεταβλητή σαν αριθμό κινητής υποδιαστολής με την μορφή [-]ddd.ddd
%x:	εκτυπώνει την αντίστοιχη μεταβλητή σαν μη προσημασμένο δεκαεξαδικό ακέραιο χρησιμοποιώντας μικρούς (όχι κεφαλαίους) χαρακτήρες
%X:	εκτυπώνει την αντίστοιχη μεταβλητή σαν μη προσημασμένο δεκαεξαδικό ακέραιο χρησιμοποιώντας κεφαλαίους χαρακτήρες
%s:	εκτυπώνει την αντίστοιχη μεταβλητή σαν γραμματοσειρά, η οποία είναι αποθηκευμένη στην SRAM
%p:	εκτυπώνει την αντίστοιχη μεταβλητή σαν γραμματοσειρά, η οποία είναι αποθηκευμένη στην FLASH
%%:	εκτυπώνει τον χαρακτήρα “%”

Η συνάρτηση **printf** υποστηρίζει ορισμένους χαρακτήρες ελέγχου, οι οποίοι ενσωματώνονται στην γραμματοσειρά **fmtstr** και προσδιορίζονται με τον **χαρακτήρα διαφυγής “\”**. Οι συνηθέστερα χρησιμοποιούμενοι από αυτούς είναι:

\n	Μετάβαση στην επόμενη γραμμή
\r	Μετάβαση στην αρχή της ίδιας γραμμής

Παράδειγμα:

```
#include <mega2560.h>
#include <stdio.h>
#include <delay.h>
#include <ctype.h>
#define xtal 8000000L
#define baud 9600

void main(void)
{
    int i=5;
    char k='a';
    char str1[10]="hello";
    float vl=12.12345;

    // Crystal Oscillator division factor: 1
    #pragma optsize-
    CLKPR=(1<<CLKPCE);
    CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
    #ifdef _OPTIMIZE_SIZE_
    #pragma optsize+
    #endif

    UCSR0A=0x00;
    UCSR0B=0x18;
    UCSR0C=0x06;
    UBRR0H=(xtal/16/ baud-1) >> 8;
    UBRR0L=(xtal/16/ baud-1)&0xFF;

    while (1)
    {
        printf("\r i=%d, k=%c, str1=%s, val=%.2f, val=%.2e", i,k,str1,vl,vl);
        delay_ms(1000);
    }
}
```

char *gets(char *str, unsigned char len)

Εκτελεί διαδικασία εισόδου χρησιμοποιώντας την εντολή **getchar**. Περιμένει είσοδο από το πληκτρολόγιο και ότι πληκτρολογηθεί μέχρι να πατηθεί το πλήκτρο *Enter*, αποθηκεύεται σαν γραμματοσειρά στην μεταβλητή **str**. Ο χαρακτήρας που αντιστοιχεί στο *Enter* αντικαθίσταται με το “0”.

Το μέγιστο μήκος της γραμματοσειράς προσδιορίζεται από τον αριθμό **len**. Εάν πληκτρολογηθούν πάνω από **len** χαρακτήρες, τότε αποθηκεύονται οι “*len*” πρώτοι, και η γραμματοσειρά κλείνει με τον χαρακτήρα “0”.

Η συνάρτηση επιστρέφει δείκτη στην γραμματοσειρά **str**.

signed char scanf(char flash *fmtstr [, arg1 address, arg2 address, ...])

Η συνάρτηση συμπεριφέρεται όμοια με την αντίστοιχη συνάρτηση της ANSI C. Εκτελεί διαδικασία εισόδου μορφοποιημένου κειμένου χρησιμοποιώντας την συνάρτηση **getchar**. Η μορφοποίηση πραγματοποιείται με βάση τους προσδιοριστές μορφοποίησης που εμπεριέχονται στην γραμματοσειρά **fmtstr**. Η γραμματοσειρά **fmtstr** θεωρείται σταθερά και πρέπει να βρίσκεται στην μνήμη FLASH.

Οι προσδιοριστές μορφοποίησης που υποστηρίζονται είναι οι ακόλουθοι:

%c: είσοδος της αντίστοιχης μεταβλητής σαν χαρακτήρας ASCII

%d: είσοδος της αντίστοιχης μεταβλητής σαν ακέραιος

%i: είσοδος της αντίστοιχης μεταβλητής σαν ακέραιος

%u: είσοδος της αντίστοιχης μεταβλητής σαν μη προσημασμένος ακέραιος

%x: είσοδος της αντίστοιχης μεταβλητής σαν μη προσημασμένος δεκαεξαδικός ακέραιος

%s: είσοδος της αντίστοιχης μεταβλητής σαν γραμματοσειρά, η οποία είναι αποθηκευμένη στην SRAM

Παράδειγμα:

```
#include <mega2560.h>
#include <stdio.h>
#include <delay.h>
#include <ctype.h>
#define xtal 4000000L
#define baud 9600

void main(void)
{

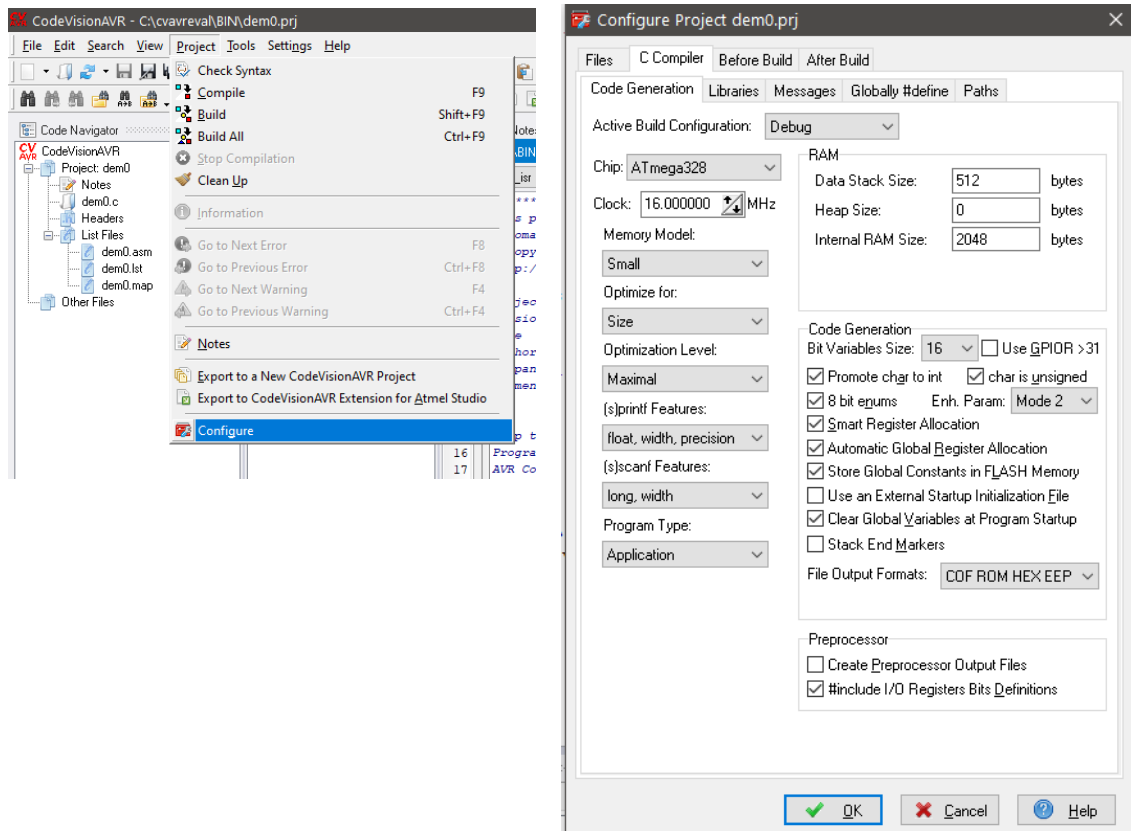
    int i=5;
    char k='a';
    char str1[10];

    // Crystal Oscillator division factor: 1
    #pragma optsize-
    CLKPR=(1<<CLKPCE);
    CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) | (0<<CLKPS0);
    #ifdef _OPTIMIZE_SIZE_
    #pragma optsize+
    #endif

    UCSRA=0x00;
    UCSRB=0x18;
    UCSR0C=0x06;
    UBRR0H=(xtal/16/ baud-1) >> 8;
    UBRR0L=(xtal/16/ baud-1)&0xFF;

    while (1)
    {
        putsf("\n Insert String: \n\r"); // print string from FLASH
        gets(str1,9); // scanf("%s", str1);
        putsf("\n String = \n\r");
        puts(str1);
        printf("\n\r i=%d, k=%c, str1=%s", i,k,str1);
        delay_ms(1000);
    }
}
```

Για να χρησιμοποιηθούν οι συναρτήσεις printf() και scanf() στο CodeVision, θα πρέπει να γίνουν οι αντίστοιχες ρυθμίσεις στο CodeWizardAVR. Project -> Configure κι αλλάζουμε τις τιμές των (s)printf και (s)scanf όπως φαίνεται παρακάτω.



Γενικό παράδειγμα εισόδου & εξόδου δεδομένων:

```
#include <mega2560.h>
#include <stdio.h>

/* uart settings */
#define xtal 8000000L
#define baud 9600

void main(void)
{
    char ch;

    /* we need to use the keyword flash, for the compiler
    to store the char array as static in FLASH */
    flash char *str2;

    /* these char arrays are stored in RAM and can be modified */
    char str1[10], *str3;
    int i;

    // Crystal Oscillator division factor: 1
    #pragma optimize-
    CLKPR=(1<<CLKPCE);
    CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) |
    (0<<CLKPS0);
    #ifdef _OPTIMIZE_SIZE_
    #pragma optimize+

```

```
#endif

UCSR0A=0x00;
UCSR0B=0x18;
UCSR0C=0x06;
UBRR0H=(xtal/16/baud-1) >> 8;
UBRR0L=(xtal/16/baud-1)&0xFF;

str2="Hello from flash";

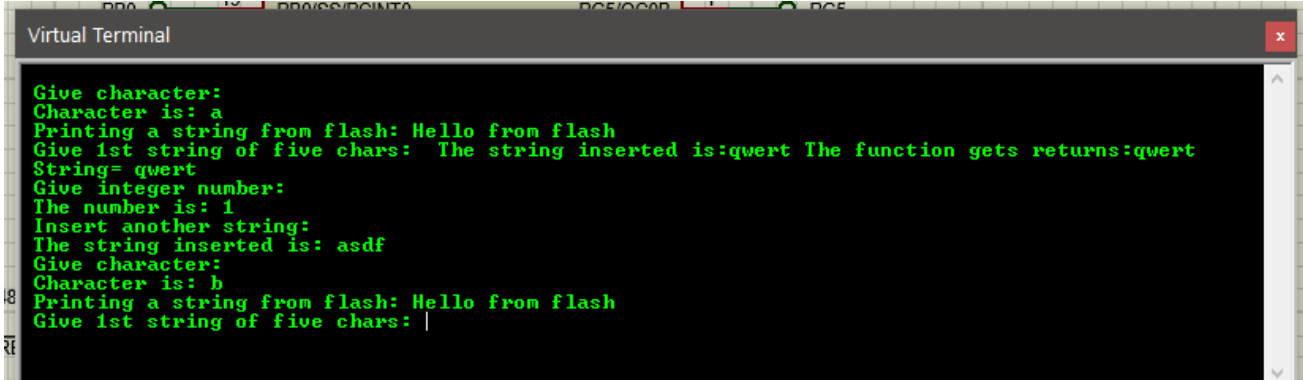
while (1)
{
    printf("\n\r Give character: ");
    ch=getchar();
    printf("\n\r Character is: ");
    putchar(ch);

    putsf("\n\r Printing a string from flash: ");
    putsf(str2);

    printf("\n\r Give 1st string of five chars: ");
    str3=gets(str1,5);
    putsf("\n The string inserted is:");
    puts(str1);
    putsf("\n The function gets returns:");
    puts(str3);
    printf("\n\r String= %s", str1);

    printf("\n\r Give integer number: ");
    scanf("%d",&i);
    getchar();
    printf("\n\r The number is: %d ",i);

    printf("\n\r Insert another string: ");
    scanf("%s",str1);
    getchar();
    printf("\n\r The string inserted is: %s",str1);
}
}
```



```
Virtual Terminal
Give character:
Character is: a
Printing a string from flash: Hello from flash
Give 1st string of five chars: The string inserted is:qwert The function gets returns:qwert
String= qwert
Give integer number:
The number is: 1
Insert another string:
The string inserted is: asdf
Give character:
Character is: b
Printing a string from flash: Hello from flash
Give 1st string of five chars: |
```

Συναρτήσεις της `stdlib.h`

Οι συναρτήσεις αυτής της κατηγορίας μετατρέπουν τις παραμέτρους με τις οποίες συντάσσονται, από ένα συγκεκριμένο τύπο σε έναν άλλο. Τα πρωτότυπά τους βρίσκονται στο αρχείο επικεφαλίδας **`stdlib.h`**. Το συγκεκριμένο αρχείο θα πρέπει να δηλωθεί με τον τελεστή **`#include`** πριν τη χρήση των προκαθορισμένων συναρτήσεων βιβλιοθήκης.

`int atoi(char *str)`

Μετατρέπει την γραμματοσειρά *str* σε ακέραιο (int).

`long int atol(char *str)`

Μετατρέπει την γραμματοσειρά *str* σε *long integer*.

`void itoa(int n, char *str)`

Μετατρέπει τον ακέραιο *n* σε χαρακτήρες της γραμματοσειράς *str*.

`void ltoa(long int n, char *str)`

Μετατρέπει τον *long integer* *n* σε χαρακτήρες της γραμματοσειράς *str*.

`void ftoa(float n, unsigned char decimals, char *str)`

Μετατρέπει τον αριθμό κινητής υποδιαστολής (*float*) *n* σε χαρακτήρες της γραμματοσειράς *str*. Ο αριθμός των δεκαδικών ψηφίων που θα απεικονιστεί προσδιορίζεται από τον δεύτερο τελεστή της συνάρτησης. *z*

`void ftoe(float n, unsigned char decimals, char *str)`

Μετατρέπει τον αριθμό κινητής υποδιαστολής (*float*) *n* σε χαρακτήρες της γραμματοσειράς *str*. Ο αριθμός των δεκαδικών ψηφίων που θα απεικονιστεί προσδιορίζεται από τον δεύτερο τελεστή της συνάρτησης. Η αναπαράσταση του αριθμού γίνεται με μορφή εκθετική (π.χ. 12.35e-5).

`float atof(char *str)`

Μετατρέπει τους χαρακτήρες της γραμματοσειράς *str* σε αριθμό κινητής υποδιαστολής.

`int rand (void)`

Παράγει έναν ψευδο-τυχαίο αριθμό μεταξύ 0 και 32767.

`void srand(int seed)`

Θέτει την αρχική τιμή, η οποία χρησιμοποιείται από την συνάρτηση παραγωγής ψευδο-τυχαίων αριθμών (**`rand`**).

Οι παραπάνω συναρτήσεις είναι πολύ χρήσιμες για τον χειρισμό και τις λειτουργίες του E/E με χρήση αριθμών κινητής υποδιαστολής (*float*). Η συνάρτηση *scanf* δε μπορεί να χειριστεί παραμέτρους *float*, αλλά με την βοήθεια των παραπάνω συναρτήσεων μπορούμε να μετατρέπουμε μεταβλητές τύπου *float* σε γραμματοσειρές.

Παράδειγμα

```
#include <mega2560.h>
#include <stdio.h>
#include <ctype.h>
```

```
#include <stdlib.h>

#define xtal 8000000L
#define baud 9600

void main(void)
{
    char st1[10];
    int i;
    float fnum;

    // Crystal Oscillator division factor: 1
    #pragma optsize-
    CLKPR=(1<<CLKPCE);
    CLKPR=(0<<CLKPCE) | (0<<CLKPS3) | (0<<CLKPS2) | (0<<CLKPS1) |
(0<<CLKPS0);
    #ifdef _OPTIMIZE_SIZE_
    #pragma optsize+
    #endif

    UCSRA=0x00;
    UCSRB=0x18;
    UCSR0C=0x06;
    UBRR0H=(xtal/16/ baud-1) >> 8;
    UBRR0L=(xtal/16/ baud-1)&0xFF;

    while (1)
    {
        printf("\n\r Insert integer number: ");
        gets(st1,5);
        printf("\n\r String is :%s", st1);
        printf("\n\r String printed from puts is: ");
        puts(st1);

        i=atoi(st1);
        printf("\n\r Number is : %d", i);

        printf("\n\r Insert decimal number: ");
        scanf("%s",st1);
        getchar();

        printf("\n\r String is :%s", st1);
        fnum=atof(st1);

        fnum *= 2;

        ftoa(fnum, 2, st1);
        printf("\n\r The double of the inserted float is: %s", st1);
    }
}
```

Συναρτήσεις της math.h

unsigned int **abs**(int x)

Επιστρέφει την απόλυτη τιμή του ακεραίου x.

unsigned long labs(*Long int x*)

Επιστρέφει την απόλυτη τιμή της *long integer* παραμέτρου *x*.

int max(*int a, int b*)

Επιστρέφει τον μεγαλύτερο από τους δύο ακεραίους *a* και *b*.

long int lmax(*Long int a, Long int b*)

Επιστρέφει την μεγαλύτερη από τις δύο *long integer* παραμέτρους *a* και *b*.

float fmax(*float a, float b*)

Επιστρέφει τον μεγαλύτερο από τους δύο αριθμούς κινητής υποδιαστολής *a* και *b*.

int min(*int a, int b*)

Επιστρέφει τον μικρότερο από τους δύο ακεραίους *a* και *b*.

long int lmin(*Long int a, Long int b*)

Επιστρέφει την μικρότερη από τις δύο *long integer* παραμέτρους *a* και *b*.

float fmin(*float a, float b*)

Επιστρέφει τον μικρότερο από τους δύο αριθμούς κινητής υποδιαστολής *a* και *b*.

signed char sign(*int x*)

Επιστρέφει τις τιμές -1, 0, 1 εάν ο ακέραιος *x* είναι αρνητικός, μηδέν ή θετικός, αντίστοιχα.

signed char lsign(*Long int x*)

Επιστρέφει τις τιμές -1, 0, 1 εάν η τιμή της *long integer* παραμέτρου *x* είναι αρνητική, μηδέν ή θετική, αντίστοιχα.

signed char fsign(*float x*)

Επιστρέφει τις τιμές -1, 0, 1 εάν ο αριθμός κινητής υποδιαστολής *x* είναι αρνητικός, μηδέν ή θετικός, αντίστοιχα.

unsigned char isqrt(*unsigned int x*)

Επιστρέφει την τετραγωνική ρίζα του μη προσημασμένου ακεραίου *x*.

unsigned int lsqrt(*unsigned Long x*)

Επιστρέφει την τετραγωνική ρίζα της μη προσημασμένης *long integer* παραμέτρου *x*.

float sqrt(*float x*)

Επιστρέφει την τετραγωνική ρίζα του μη προσημασμένου αριθμού κινητής υποδιαστολής *x*.

float floor(*float x*)

Επιστρέφει τον μικρότερο ακέραιο, ο οποίος είναι πλησιέστερος στον αριθμό κινητής υποδιαστολής *x*.

float ceil(*float x*)

Επιστρέφει τον μεγαλύτερο ακέραιο, ο οποίος είναι πλησιέστερος στον αριθμό κινητής υποδιαστολής *x*.

float fmod(*float x, float y*)

Επιστρέφει το υπόλοιπο της διαίρεσης: *x/y*.

float modf(*float x, float *ipart*)

Διασπά τον αριθμό κινητής υποδιαστολής x στο ακέραιο και το δεκαδικό μέρος του. Το δεκαδικό μέρος επιστρέφεται από την συνάρτηση σαν προσημασμένος αριθμός κινητής υποδιαστολής. Το ακέραιο μέρος αποθηκεύεται στον αριθμό κινητής υποδιαστολής *ipart*.

float exp(float x)

Επιστρέφει τον παράγοντα: e^x .

float log(float x)

Επιστρέφει τον φυσικό λογάριθμο του αριθμού κινητής υποδιαστολής x .

float log10(float x)

Επιστρέφει τον λογάριθμο βάσης 10, του αριθμού κινητής υποδιαστολής x .

float pow(float x, float y)

Επιστρέφει τον παράγοντα: x^y .

float sin(float x)

Επιστρέφει το ημίτονο του αριθμού κινητής υποδιαστολής x , όπου η ακτίνα εκφράζεται σε ακτίνια (radians).

float cos(float x)

Επιστρέφει το συνημίτονο του αριθμού κινητής υποδιαστολής x , όπου η ακτίνα εκφράζεται σε ακτίνια (radians).

float tan(float x)

Επιστρέφει την εφαπτομένη του αριθμού κινητής υποδιαστολής x , όπου η ακτίνα εκφράζεται σε ακτίνια (radians).

Ασκήσεις

1. Να γραφεί ένα πρόγραμμα το οποίο να εμφανίζει τους αριθμούς 0-9 στη θέση 0,0 της οθόνης.
 - 1α. Τροποποιήστε το πρόγραμμα ώστε να εμφανίζονται στην οθόνη οι αριθμοί 0-99.
2. Να γραφεί ένα πρόγραμμα το οποίο να εμφανίζει τους πεζούς χαρακτήρες του αγγλικού αλφάβητου στη θέση 0,0 και τα κεφαλαία γράμματα στη δεύτερη γραμμή της οθόνης.
3. Να γραφεί ένα πρόγραμμα το οποίο να ζητά από τον χρήστη να εισάγει έναν μονοψήφιο αριθμό από το σειριακό τερματικό. Στη συνέχεια θα πρέπει να εμφανίζεται ο αριθμός που εισήχθηκε στην πρώτη γραμμή της LCD οθόνης και στην δεύτερη γραμμή να εμφανίζεται ο χαρακτήρας «*» τόσες φορές, όσες ο εισηγμένος αριθμός. Για παράδειγμα, όταν στην πρώτη γραμμή εμφανίζεται ο αριθμός 4 στην δεύτερη γραμμή να εμφανίζονται ****.
4. Να γραφεί ένα πρόγραμμα το οποίο να εμφανίζει το μήνυμα Hello world στην οθόνη.
 - 4α. Τροποποιήστε το πρόγραμμα ώστε το μήνυμα να εμφανίζεται στην οθόνη κάθε 1 sec.
 - 4β. Τροποποιήστε το πρόγραμμα ώστε να εμφανίζεται ένας-ένας ο χαρακτήρας του μηνύματος εισάγοντας κάποια χρονική καθυστέρηση. Όταν εμφανιστεί ο τελευταίος χαρακτήρας, η οθόνη να καθαρίζει και η διαδικασία εμφάνισης των χαρακτήρων να επαναλαμβάνεται.
 - 4γ. Τροποποιήστε το πρόγραμμα ώστε αν ο τελευταίος χαρακτήρας του μηνύματος είναι ο d, οι χαρακτήρες του μηνύματος να εμφανίζονται με αντίστροφη σειρά, αλλιώς να εμφανίζεται το μήνυμα απευθείας.
 - 4δ. Τροποποιήστε το πρόγραμμα ώστε ο χαρακτήρας e να αντικαθίσταται με a και το κενό με το +.
 - 4ε. Τροποποιήστε το πρόγραμμα ώστε οι δύο λέξεις να εμφανίζονται σε ξεχωριστές γραμμές.

5. Να γραφεί ένα πρόγραμμα το οποίο να διαβάσει συνεχώς χαρακτήρες από την σειριακή πόρτα και να τους εμφανίζει στην lcd οθόνη.
- 5α. Να τροποποιήσετε το παραπάνω πρόγραμμα ώστε αν ο χρήστης εισάγει το 'z', το πρόγραμμα να εμφανίζει το 'a'. Αλλιώς, να εμφανίζει τον επόμενο ASCII χαρακτήρα.
- 5β. Να τροποποιήσετε το παραπάνω πρόγραμμα ώστε να εμφανίζει μόνο τα πεζά γράμματα.
- 5γ. Να τροποποιήσετε το παραπάνω πρόγραμμα ώστε σε περίπτωση που ο χρήστης εισάγει ένα πεζό γράμμα, να εμφανίζει όλους τους χαρακτήρες από το a μέχρι τον χαρακτήρα που εισήγαγε ο χρήστης.
6. Να γραφεί ένα πρόγραμμα το οποίο να ελέγχει τις τιμές των ακροδεκτών 6 και 7 της πόρτας D και ανάλογα με τον συνδυασμό τους να εμφανίζει στο σειριακό τερματικό ανάλογο μήνυμα: Zero, One, Two και Three.
7. Να γραφεί ένα πρόγραμμα το οποίο να διαβάσει συνεχώς χαρακτήρες από την σειριακή πόρτα και να αποθηκεύει σε έναν πίνακα μόνο τα πεζά γράμματα του αλφαβήτου. Όταν αποθηκευτούν πέντε χαρακτήρες, το πρόγραμμα να εμφανίζει το περιεχόμενο του πίνακα στην lcd οθόνη και να συνεχίζει με την ανάγνωση νέας πεντάδας.
8. Να γραφεί ένα πρόγραμμα το οποίο να διαβάσει συνεχώς έναν ακέραιο από την σειριακή πόρτα και αν ανήκει στο [0, 255] να τον εμφανίζει στην lcd οθόνη, αλλιώς να εμφανίζει το μήνυμα "No".
- 8α. Να τροποποιήσετε το πρόγραμμα, ώστε να εμφανίζει τον αριθμό σε δυαδική μορφή στην δεύτερη γραμμή της οθόνης.
9. Να γραφεί πρόγραμμα που να επιτρέπει στον χρήστη να ελέγχει τους ακροδέκτες της PORTD μέσω της UART. Δηλαδή, με μηνύματα της μορφής «x: ON» ή «x: OFF» να ενεργοποιείται ο αντίστοιχος "x" ακροδέκτης κι ο μικροελεγκτής να απαντά μέσω της UART «x:OK» αφού ενεργοποιήσει τον ακροδέκτη. Ένα παράδειγμα επικοινωνίας ακολουθεί, στο οποίο ο χρήστης ενεργοποιεί την έξοδο στον ακροδέκτη 1 και απενεργοποιεί την έξοδο στον ακροδέκτη 5.
 - > 1: ON (χρήστης)
 - < 1: OK (μικροελεγκτής)
 - > 5: OFF (χρήστης)
 - < 5: OK (μικροελεγκτής)

10. Να υλοποιηθεί μια εντολή της μορφής «x?», η οποία να επιστρέφει στον χρήστη την κατάσταση του ακροδέκτη x. Ένα παράδειγμα επικοινωνίας ακολουθεί:
 - > 1? (χρήστης)
 - < 1: ON (μικροελεγκτής)
 - > 5? (χρήστης)
 - < 5: OFF (μικροελεγκτής)
11. Να δημιουργηθεί πρόγραμμα το οποίο να ζητά από τον χρήστη να εισάγει σειριακά 10 αριθμούς κινητής υποδιαστολής και στο τέλος να προβάλει τον μεγαλύτερο, τον μικρότερο και τον μέσο όρο τους.
12. Να γραφεί ένα πρόγραμμα το οποίο για κάθε πενήντα θερμοκρασίες που διαβάζει από την σειριακή πόρτα, να εμφανίζει στην lcd οθόνη τον μέσο όρο τους με ακρίβεια ενός δεκαδικού ψηφίου.
- 12α. Να τροποποιήσετε το πρόγραμμα, ώστε για κάθε πενήντάδα θερμοκρασιών που διαβάζει να εμφανίζει στην lcd οθόνη τον μέσο όρο των θερμοκρασιών με τιμή στο [10, 20]. Αν δεν εισαχθεί καμία θερμοκρασία σε αυτό το διάστημα, το πρόγραμμα να εμφανίζει "No". Για παράδειγμα, αν διαβαστούν οι τιμές 12, 14 και οι άλλες 48 τιμές είναι έξω από το [10, 20], το πρόγραμμα να εμφανίζει 13.