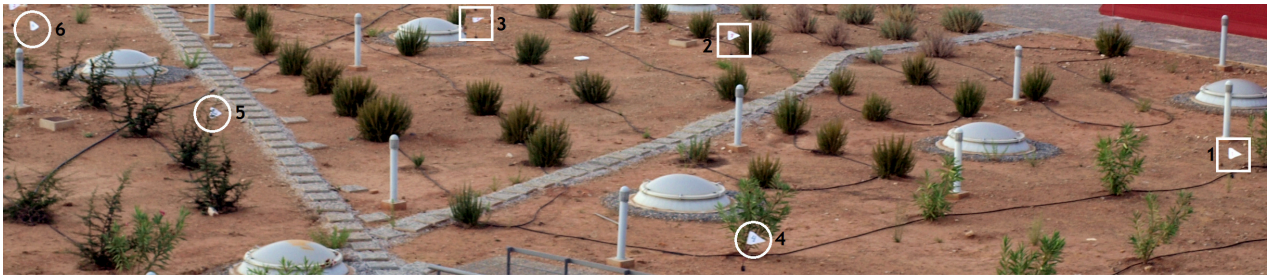
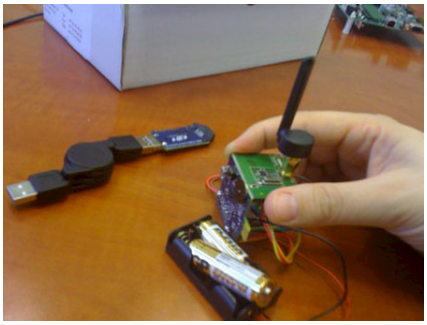
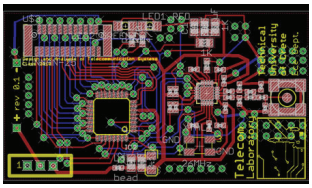
Πολυτεχνείο Κρήτης
Τμήμα Ηλεκτρονικών Μηχ\κών & Μηχ\κών Η\Υ
Χειμερινό Εξάμηνο 2010-2011

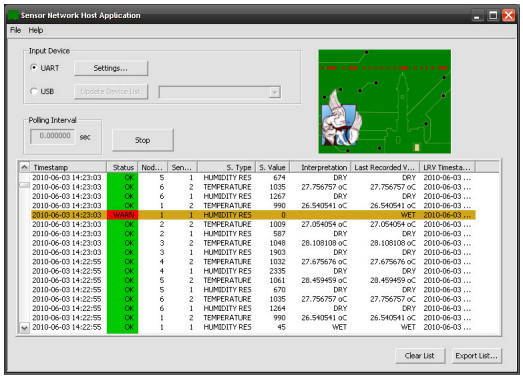
ΤΗΛ 412 Ανάλυση και Σχεδίαση (Σύνθεση) Τηλεπικοινωνιακών Διατάξεων

7ου Εξαμήνου









Σειριακή Επικοινωνία Υπολογιστή με Πλακέτα Ανάπτυξης

Υποστηρικτικό Υλικό

Διδάσκων: Άγγελος Μπλέτσας
aggelos@telecom.tuc.gr

version 0.1.0

1 Εισαγωγή

1.1 Σειριακή Διεπαφή (J5)

Η πλακέτα ανάπτυξης C8051F320 που χρησιμοποιείται στα πλαίσια του εργαστηρίου του μαθήματος, είναι εξοπλισμένη με ένα RS-232 κύκλωμα πομποδέκτη καθώς και ένα DB-9 connector.
Τα παραπάνω μπορούν να χρησιμοποιηθούν, με σκοπό την καλύτερη αποσφαλμάτωση και τον έλεγχο ορθής λειτουργίας των συστημάτων που υλοποιούμε μέσω της εκτύπωσης στην οθόνη μηνυμάτων ή τιμών μεταβλητών.

Για χάριν της καλύτερης εξοικείωσης σας με την δημιουργία σειριακής επικοινωνίας μεταξύ του υπολογιστή και της πλακέτας ανάπτυξης σας δίνεται το project με όνομα UART_EXAMPLE το οποίο βρίσκεται στον φάκελο με τον κωδικό του μαθήματος στον υπολογιστή του κάθε πάγκου του εργαστηρίου. Για την επεξεργασία του κώδικα θα χρησιμοποιηθεί η πλατφόρμα της Sillabs

Επιπλέον σας δίνεται το header file C8051F320.h που περιέχει ορισμούς σταθερών παραμέτρων του μικροελεγκτή μας.

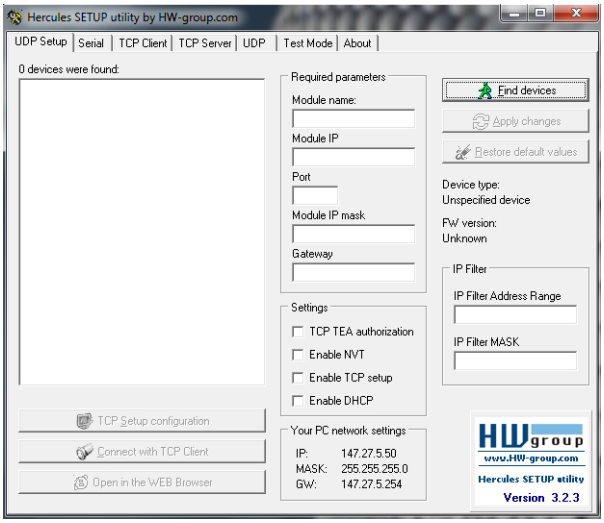
Για την επεξεργασία των κωδίκων και την φόρτωση τους στον μικροελεγκτή θα χρησιμοποιηθεί η πλατφόρμα της Silabs ενώ σαν τερματικό θα χρησιμοποιηθεί το Hercules, μία standalone εφαρμογή, το εκτελέσιμο αρχείο της οποίας είναι επίσης τοποθετημένο στον φάκελο με τον κωδικό του μαθήματος.

2 Εφαρμογή Hercules

2.1 Εγκατάσταση και Λειτουργία

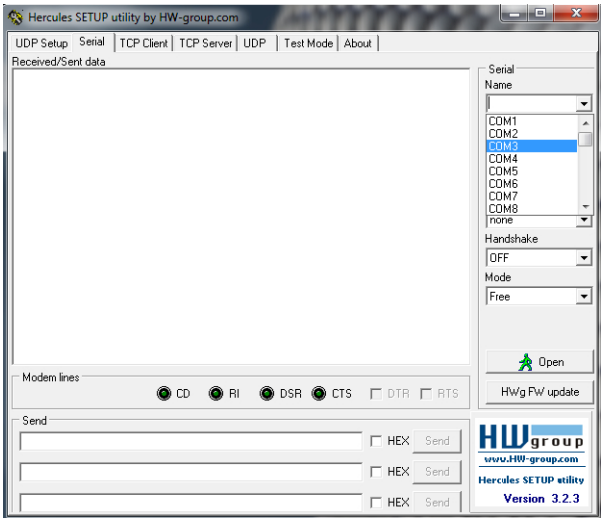
Όπως αναφέραμε παραπάνω το εκτελέσιμο αρχείο της εφαρμογής Hercules είναι τοποθετημένο στον φάκελο με όνομα τον κωδικό του μαθήματος.

Αφότου εκκινήσετε την εφαρμογή, θα σας εμφανιστεί ένα παράθυρο όπως αυτό απεικονίζεται στο Σχήμα 1.



Σχήμα 1: Εκκίνηση τερματικού παραθύρου της εφαρμογής Hercules.

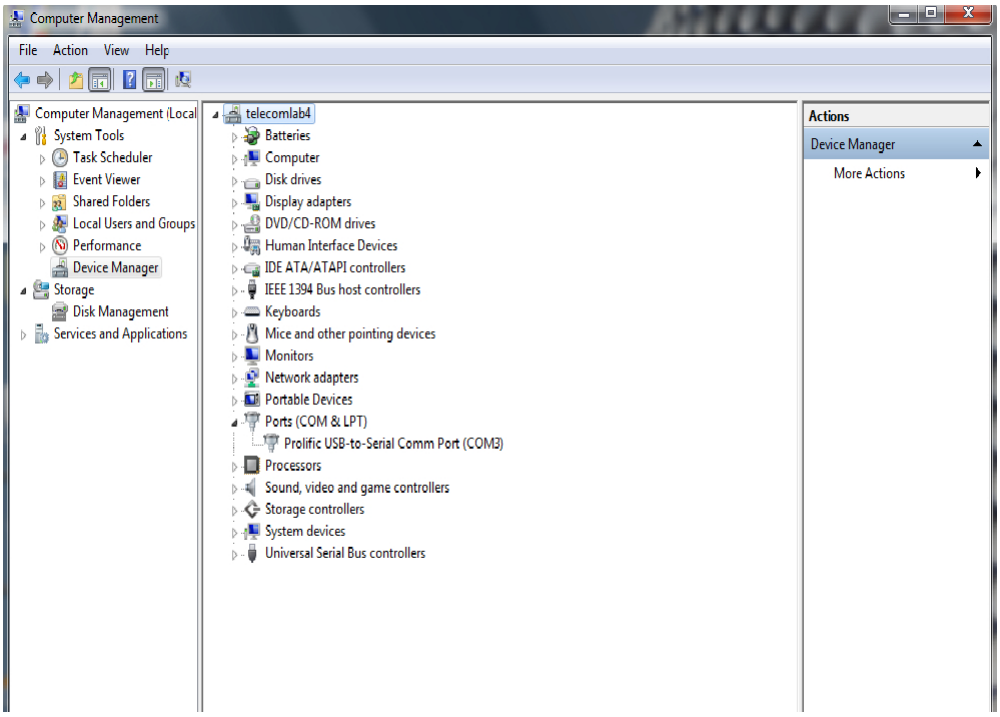
Στην συνέχεια επιλέγοντας την καρτέλα *Serial* καλείστε να ρυθμίσετε κάποιες παραμέτρους στο δεξί μέρος του παραθύρου, όπως αυτό της σειριακής θύρας (π.χ. COM1).



Σχήμα 2: Εφαρμογή Hercules (συνέχεια).

Έτσι για την ορθή ρύθμιση των παραμέτρων, χρειάζεται αφού πρώτα συνδέσετε τον Serial to USB

μετατροπέα στην αντίστοιχη θύρα, να μεταβείτε στον Device Manager του υπολογιστή σας. Στην καρτέλα για τις σειριακές θύρες (Ports (COM & LPT)) αφού δείτε τον αριθμό της εικονικής σειριακής θύρας που δημιουργήθηκε, τον επιλέγετε στην αντίστοιχη καρτέλα της εφαρμογής Hercules.



Σχήμα 3: Ρύθμιση παραμέτρων στην εφαρμογή Hercules.

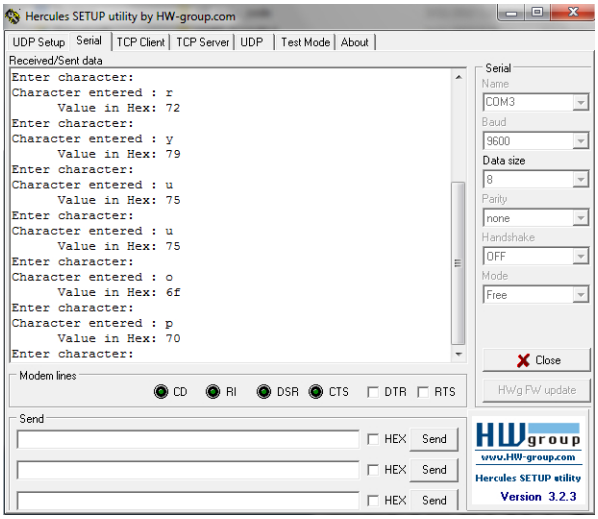
Έπειτα, σας δίνεται η δυνατότητα να επιλέξετε διαφορετική τιμή για το Baud Rate κάτι το οποίο δεν πρέπει να ξεχάσετε να δηλώσετε σαν σταθερά στον κώδικα σας. Η default τιμή του είναι 9600 και είναι αυτή που χρησιμοποιείται στο παράδειγμα το οποίο θα εξασκηθείτε.

Τέλος με το κουμπί open η σειριακή θύρα θα 'ανοίξει' και θα είναι έτοιμη για την λήψη δεδομένων από την πλακέτα ανάπτυξης.

3 Περιγραφή Δοθέντος Λογισμικού

Στο παράδειγμα που σας δίνεται, μπορείτε να εισάγετε κάποιο χαρακτήρα και το τερματικό θα εκτυπώσει την είσοδο σας από το πληκτρολόγιο σε δεκαεξαδική μορφή.

Ένα παράδειγμα υλοποιήσης εμφανίζεται στο Σχήμα 4.



Σχήμα 4: Παράδειγμα F32x_UART_STDIO.c στην εφαρμογή Hercules.

3.1 Κώδικας F32x_UART_STDIO.c

Ορισμός των header files που περιέχουν απαραίτητες δηλώσεις καταχωρητών καθώς και συναρτήσεων απαραίτητων για διάβασμα από το πληκτρολόγιο και εκτύπωση στην οθόνη.

```
#include <c8051f320.h>                  // SFR declarations
#include <stdio.h>
#include <common.h>
```

Στην συνέχεια ορίζονται 2 σταθερές όπως η συχνότητα του εσωτερικού ταλαντωτή η οποία τίθεται στα 12MHz ενώ ο ρυθμός μεταφοράς δεδομένων στην σειριακή επικοινωνία ορίζεται στα 9600bps.

```
#define SYSCLK      12000000            // SYSCLK frequency in Hz
#define BAUDRATE        9600            // Baud rate of UART in bps
```

Ακόμη ορίζεται ένας buffer με το όνομα myData τύπου BYTE ο οποίος θα αποθηκευτεί σ΄ ένα άλλο κομμάτι της μνήμης (xdata) για το οποίο έχει γίνει εκτενής περιγραφή στην αναφορά του προηγούμενου εργαστηρίου.

```
BYTE xdata myData[]= {20, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14};
```

Ορισμός των πρωτοτύπων των συναρτήσεων που μερικές από τις οποίες θα χρησιμοποιηθούν στο παρακάτω πρόγραμμα (UART0_Init, SYSCLK_Init, PORT_Init).

```
void SYSCLK_Init (void);
void UART0_Init (void);
void PORT_Init (void);
void Timer2_Init (int);
```

Στην κύρια συνάρτηση του προγράμματος μας αφού πρώτα ορίσουμε μια μεταβλητή τύπου χαρακτήρα, απενεργοποιήσουμε τον watchdog timer και καλέσουμε τις 3 απαραίτητες συναρτήσεις για την αρχικοποίηση των Ports, του εσωτερικού ταλαντωτή και της σειριακής επικοινωνίας αντίστοιχα, καλούμε την συνάρτηση printf με ορίσματα τις προτάσεις που θέλουν να εκτυπωθούν στην οθόνη. Να επισημάνουμε ακόμη ότι η συνάρτηση getChar() χρησιμοποιείται για το διάβασμα χαρακτήρων από το πληκτρολόγιο.

```
void main (void)
{
   unsigned char inputcharacter;
   int byteAsInt;
   int iter;
   PCA0MD &= ~0x40;                    // WDTE = 0 (clear watchdog timer
                                       // enable)
   PORT_Init();                        // Initialize Port I/O
   SYSCLK_Init ();                     // Initialize Oscillator
   UART0_Init();
   while (1)
   {
      printf ("\nEnter character: ");
      inputcharacter = getkey ();
      printf("\nCharacter entered : %c",inputcharacter);
      printf("\n      Value in Hex: %bx",inputcharacter);
   printf("\n");
   printf("Buffer myData:");
   for (iter=1;iter<=(sizeof(myData)-1);iter++){
     byteAsInt=myData[iter];
   printf("%d ",byteAsInt);
  }//end of for
   printf("\n");
   }
}
```

Έτσι καλούμαστε να εισάγουμε σαν είσοδο κάποιο χαρακτήρα και το πρόγραμμα μας θα εκτυπώσει την τιμή του χαρακτήρα σε δεκαεξαδική μορφή.

Επιπροσθέτως, ένας βρόγχος επανάληψης for υλοποιείται, με σκοπό την εκτύπωση στο τερματικό των στοιχείων του buffer που ορίστηκε παραπάνω.

Παρακάτω ακολουθούν οι κώδικες των συναρτήσεων που καλούνται κατά την διάρκεια του προγράμματος.

Αρχικά με την κλήση της PORT_Init θα ενεργοποιήσουμε το P0.4 ως push-pull ψηφιακή έξοδο και το P0.5 ως open-drain.

```
void PORT_Init (void)
{
   P0MDOUT |= 0x10;                    // enable UTX as push-pull output
   XBR0    = 0x01;                     // Enable UART on P0.4(TX) and P0.5(RX)
   XBR1    = 0x40;                     // Enable crossbar and weak pull-ups
}
```

Την συνάρτηση SYSCLK_Init την έχετε επίσης συναντήσει σε προηγούμενα εργαστήρια και χρησιμοποιείται για να καθορίσουμε ως ρολόι του συστήματος τον εσωτερικό ταλαντωτή.

```
void SYSCLK_Init (void)
{
   OSCICN |= 0x03;                     // Configure internal oscillator for
                                       // its maximum frequency
   CLKSEL = 0x20;
   RSTSRC  = 0x04;                     // Enable missing clock detector

}
```

Τέλος η συνάρτηση UART0_Init χρησιμοποιείται για την εγκαθίδρυση σειριακής επικοινωνίας ανάμεσα στον υπολογιστή και στην πλακέτα ανάπτυξης με ρυθμό μεταφοράς που ορίζεται στις πρώτες γραμμές του προγράμματος.

```
void UART0_Init (void)
{
   SCON0 = 0x10;                       // SCON0: 8-bit variable bit rate
                                       //        level of STOP bit is ignored
                                       //        RX enabled
                                       //        ninth bits are zeros
                                       //        clear RI0 and TI0 bits
   if (SYSCLK/BAUDRATE/2/256 < 1) {
      TH1 = -(SYSCLK/BAUDRATE/2);
      CKCON &= ~0x0B;                  // T1M = 1; SCA1:0 = xx
      CKCON |=  0x08;
   } else if (SYSCLK/BAUDRATE/2/256 < 4) {
      TH1 = -(SYSCLK/BAUDRATE/2/4);
      CKCON &= ~0x0B;                  // T1M = 0; SCA1:0 = 01
      CKCON |=  0x01;
```

```
   } else if (SYSCLK/BAUDRATE/2/256 < 12) {
      TH1 = -(SYSCLK/BAUDRATE/2/12);
      CKCON &= ~0x0B;                  // T1M = 0; SCA1:0 = 00
   } else {
      TH1 = -(SYSCLK/BAUDRATE/2/48);
      CKCON &= ~0x0B;                  // T1M = 0; SCA1:0 = 10
      CKCON |=  0x02;
   }

   TL1 = TH1;                          // init Timer1
   TMOD &= ~0xf0;                      // TMOD: timer 1 in 8-bit autoreload
   TMOD |=  0x20;
   TR1 = 1;                            // START Timer1
   TI0 = 1;                            // Indicate TX0 ready
}
```