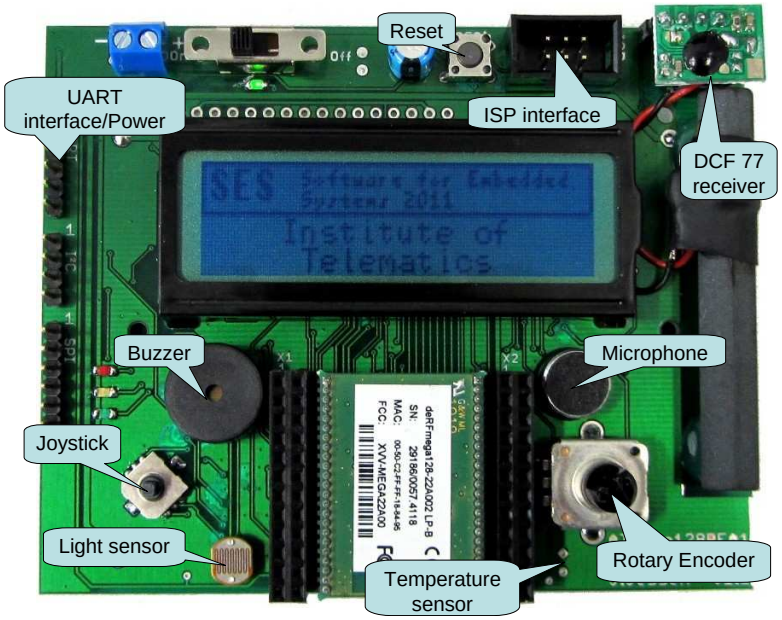
Tools & Tints

April 12th, 2016

In the following labs, we will put several components of the SES board into operation. In this lab however, we'll have an easy start with the LEDs. The SES board can be powered by 3 options: battery, external power supply or UART power supply. For ease of use we will use the UART power supply. The UART FTDI cable combines bidirectional serial communication and 5 V power supply. To power the board connect the 6-pin connector to the SES board, so that the black wire is connected to pin 1 of the UART pin header (upper pin on the left side of board). For flashing your program connect the ISP connector to the corresponding header of the board. The connector is polarity protected. It can remain connected, even while your program is running on the board.

Check polarity of the UART connector to avoid short circuits!



Task 1.1 : Get the Party Started

Read the *Eclipse AVR Tutorial* (Stud.IP) and install the software needed for AVR programming. Execute the example given in the tutorial.

Task 1.2 : Take a look at Atmel's ATmega128RFA1

In general, you should be aware of the resources and capabilities of your embedded system. For this purpose please try to find out more about the ATmega128RFA1! Important parameters are:

- size of programmable flash
- size of internal EEPROM
- size of internal SRAM

- available ports
- available counters/timers
- peripherals (e.g., UART, ADC)

Use the provided material *ATmega128RFA1 Datasheet.pdf*, which you can download from Stud.IP.

Task 1.3 : Who's In, Who's Out

Do you remember the LED blinker program of the *Eclipse AVR Tutorial*? In that program, two global variables `PORTG` and `DDRG` are used. Explain their functionality!

Task 1.4 : Looping Louie

On some occasions, e.g. the initialization of a display, delays have to be inserted in the program before performing the next operation. A very simple way of doing this is busy waiting, i.e. running a loop to burn processor cycles. You have already seen the function `_delay_ms`. In this exercise you have to make your own implementation of a waiting function. The function `void wait(uint16_t millis)` takes a 16 bit unsigned integer as the input parameter, corresponding to the delay of approximately `millis` milliseconds. You can test your function with the blinking program of the *Eclipse AVR Tutorial*.

For this task, assume a processor frequency of 16 MHz (16 million clock cycles per second). Furthermore an arbitrary busy waiting function is shown in the code snippet below. The challenge is to determine the required clock cycles for the C program. This can be achieved by compiling the source code whereupon a file with the ending **.lss* is created. In this file the assembler code for each function is listed separately. A short description of the mnemonics and the required clock cycles can be found in the document *AVR Instruction Set Manual* which can be retrieved from Stud.IP.

```
#include <stdint.h>
#include <avr/io.h>

void shortDelay (void)
{
    uint16_t i; // 16 bit unsigned integer

    for (i = 0x0100; i > 0 ; i--) {
        // prevent code optimization by using inline assembler
        asm volatile ( "nop" ); // one cycle with no operation
    }
}
```

Task 1.5 : Around the clock

To test the correctness of your delay loop implement a second LED program. After exactly 1 second toggle the LED. Check your program by synchronizing it with a watch and count the seconds for exactly 1 minute.