Education Ministry of Moldova

Technical University

Computer Science and Microelectronics

Report

Embedded Systems

laboratory work #1

Author:
Bîrcu Maxim

Supervisor:
Bragarenco Andrei

November 20, 2016

Introduction

Topic

Introduction to Micro Controller Unit programming and implementation of serial communications over UART Universal Asynchronous Receiver/Transmitter

Objectives

1. Initiation in MCU
2. Study of UART serial communication.
3. Creating PCB in Proteus
4. Write a 8 program for Atmega32 MCU for created circuit.

Objectives

Write a C program and schematics for Micro Controller Unit (MCU) using Universal asynchronous receiver/transmitter. For writing program, use ANSI-C Programming Language with AVR Compiler and for schematics use Proteus, which allow us to simulate real example.

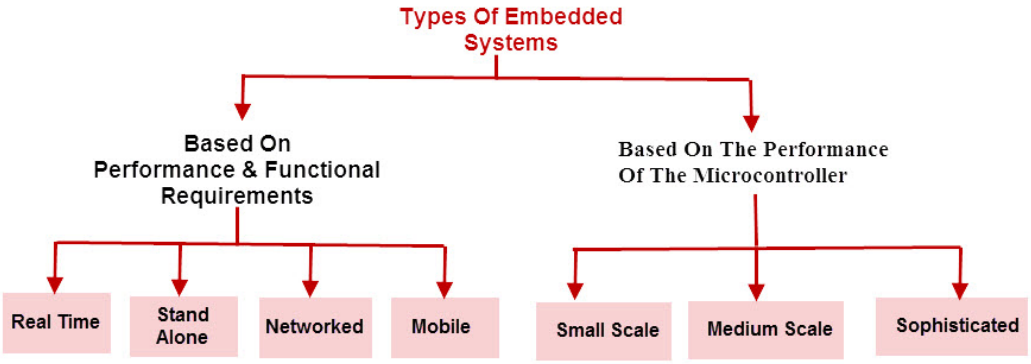
1 Embedded Systems

1.1 Definition

An Embedded system is a combination of computer hardware and software. As with any electronic system, this system requires a hardware platform and that is built with a microprocessor or micro-controller.The Embedded system hardware includes elements like user interface, Input/Output interfaces, display and memory, etc.Generally, an embedded system comprises power supply, processor, memory, timers, serial communication ports and system application specific circuits.

1.2 Types

Embedded systems can be classified into different types based on performance, functional requirements and performance of the micro-controller.



1. Embedded systems are classified into four categories based on their performance and functional requirements:

    - Stand alone embedded systems
    - Real time embedded systems
    - Networked embedded systems
    - Mobile embedded systems

2. Embedded Systems are classified into three types based on the performance of the micro-controller such as

    - Small scale embedded systems
    - Medium scale embedded systems
    - Sophisticated embedded systems

2 Micro Controller

2.1 Definition

A micro-controller (or MCU, short for micro-controller unit) is a small computer (SoC) on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of Fierroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. Micro-controllers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

2.2 Usage

Micro-controllers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other embedded systems. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, micro-controllers make it economical to digitally control even more devices and processes. Mixed signal micro-controllers are common, integrating analog components needed to control non-digital electronic systems.

2.3 How does the micro-controller operate?

Even though there is a large number of different types of micro-controllers and even more programs created for their use only, all of them have many things in common. Thus, if you learn to handle one of them you will be able to handle them all. A typical scenario on the basis of which it all functions is as follows:

1. Power supply is turned off and everything is still the program is loaded into the micro-controller, nothing indicates what is about to come

2. Power supply is turned on and everything starts to happen at high speed! The control logic unit keeps everything under control. It disables all other circuits except quartz crystal to operate. While the preparations are in progress, the first milliseconds go by.

3. Power supply voltage reaches its maximum and oscillator frequency becomes stable. SFRs are being filled with bits reflecting the state of all circuits within the micro-controller. All pins are configured as inputs. The overall electronics starts operation in rhythm with pulse sequence. From now on the time is measured in micro and nanoseconds.

4. Program Counter is set to zero. Instruction from that address is sent to instruction decoder which recognizes it, after which it is executed with immediate effect.

5. The value of the Program Counter is incremented by 1 and the whole process is repeated several million times per second.

2.4 Special Function Registers (SFR)

Special function registers are part of RAM memory. Their purpose is predefined by the manufacturer and cannot be changed therefore. Since their bits are physically connected to particular circuits within the micro-controller, such as A/D converter, serial communication module etc., any change of their state directly affects the operation of the micro-controller or some of the circuits. For example, writing zero or one to the SFR controlling an input/output port causes the appropriate port pin to be configured as input or output. In other words, each bit of this register controls the function of one single pin.

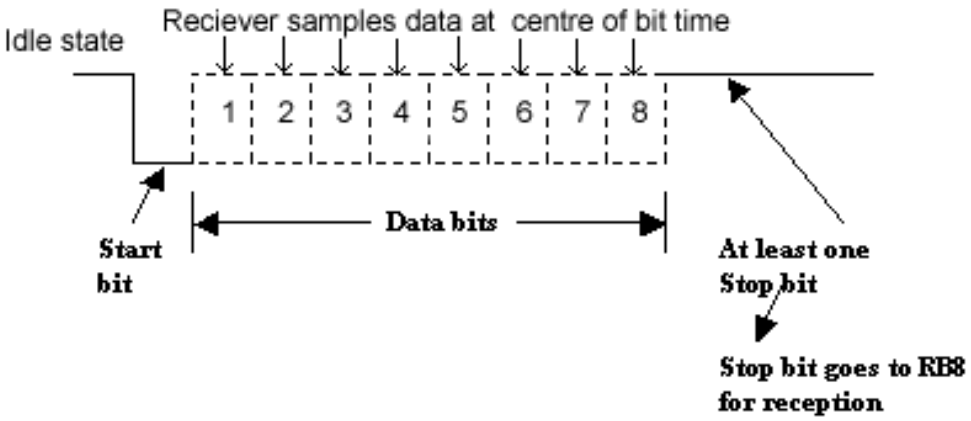
2.5 Program Counter

Program Counter is an engine running the program and points to the memory address containing the next instruction to execute. After each instruction execution, the value of the counter is incremented by 1. For this reason, the program executes only one instruction at a time just as it is written. However... the value of the program counter can be changed at any moment, which causes a "jump" to a new memory location. This is how subroutines and branch instructions are executed. After jumping, the counter resumes even and monotonous automatic counting +1, +1, +1...

3 UART

3.1 Definition

A universal asynchronous receiver/transmitter , is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. The electric signaling levels and methods (such as differential signaling, etc.) are handled by a driver circuit external to the UART.



4 Resources

4.1 Atmel Studio

Atmel Studio 7 is the integrated development platform (IDP) for developing and debugging Atmel® SMART ARM®-based and Atmel AVR® microcontroller (MCU) applications. Studio 7 supports all AVR and Atmel SMART MCUs. The Atmel Studio 7 IDP gives you a seamless and easy-to-use environment to write, build and debug your applications written in C/C++ or assembly code. It also connects seamlessly to Atmel debuggers and development kits. Additionally, Atmel Studio includes Atmel Gallery, an online apps store that allows you to extend your development environment with plug-ins developed by Atmel as well as by third-party tool and embedded software vendors. Atmel Studio 7 can also able seamlessly import your Arduino sketches as C++ projects, providing a simple transition path from Makerspace to Marketplace.

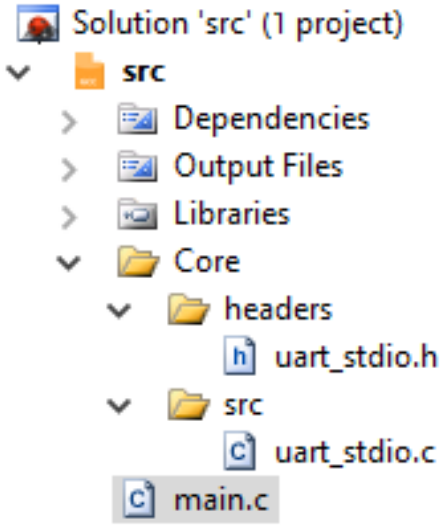
4.2 Proteus

The Proteus Design Suite is an Electronic Design Automation (EDA) tool including schematic capture, simulation and PCB Layout modules. It is developed in Yorkshire, England by Lab center Electronics Ltd with offices in North America and several overseas sales channels. The software runs on the Windows operating system and is available in English, French, Spanish and Chinese languages.

5 Solution

5.1 Project Structure

Project contains a header and source file for uart driver that was written in order to interact with peripheral which in our case is a virtual terminal.



5.2 UART dependencies

- **stdio.h** - UART as a STD stream for IO Library
- **io.h** - MACRO definition for registers which makes our driver to work not only on ATMega32 but on more devices.

5.3 UART Driver

Procedures && Functions:

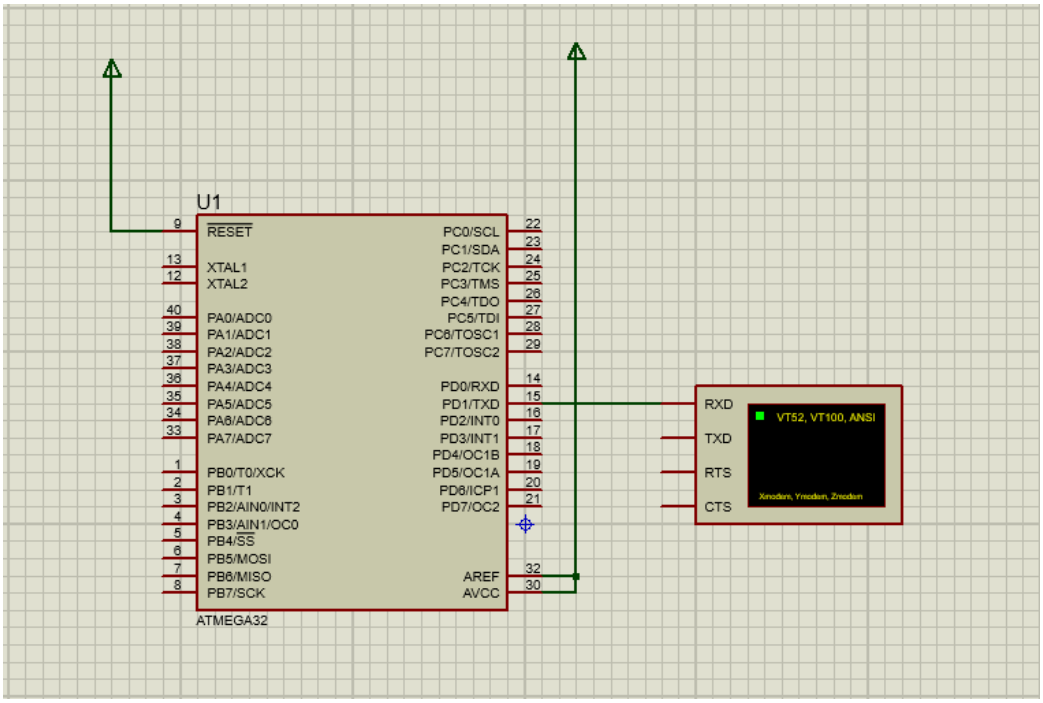
1. **void uart_stdio_Init(void);** - initializing uart Baud frequency in order to make peripheral device to understand our signals correct.
2. **void Int uart_PutChar(char c, FILE *stream);** - printing on peripheral

5.4 Main program flow

1. Global variable declarations.
2. UART Driver initialization
3. Start infinite loop (Controller life cycle)
   (a) Increment counter variable
   (b) Print counter to output(UART)
   (c) Make a delay

5.5 Circuit in Proteus

I've connected MCU to Virtual Terminal. Because I use only data transmission on one direction (OUTPUT) we need to make sure that our MC Tx is connected to Peripheral Rx.



5.6 Simulation

Conclusion

In this laboratory work I've learned basic concepts of MCU programming in C language and building a simple printed circuit using Proteus. I've implemented a simple counter and displayed it on a virtual terminal using UART and even if it was a simple program it was quite good for introduction to Embedded Systems world. It made me understand that I have a lot of work in feature cause , I've become very excited to study this filed.

Apendix

Main

```
/*
 * main.c
 *
 * Created: 10/22/2016 8:39:04 PM
 *  Author: max
 */

#include <stdio.h>
#include "Core/headers/uart_stdio.h"
#include <util/delay.h>

int main(void)
{
    while(1)
    {
        uart_stdio_init();

        int i = 0;

        while(1){
                i++;
                printf("Counter %d \n",i);
                _delay_ms(1000);
        }
        return 1;
    }
}
```

Uart_stdio

```
/*
 * uart_stdio.c
 *
 * Created: 10/22/2016 8:42:39 PM
 *  Author: max
 */
#include <stdio.h>
#include <avr/io.h>

#define UART_BAUD  9600
#define F_CPU 1000000UL

int uart_stdio_putchar(char c, FILE *stream)
{
        if (c == '\n')
        uart_stdio_putchar('\r', stream);
        while(~UCSRA & (1 << UDRE));
        UDR = c;

        return 0;
}

FILE uart_str = FDEV_SETUP_STREAM(uart_stdio_putchar, NULL, _FDEV_SETUP_

void uart_stdio_init(){
        stdout = stdin = &uart_str;

        #if F_CPU < 2000000UL && defined(U2X)
        UCSRA = _BV(U2X);               /* improve baud rate error by usi
        UBRRL = (F_CPU / (8UL * UART_BAUD)) - 1;
        #else
        UBRRL = (F_CPU / (16UL * UART_BAUD)) - 1;
        #endif
        UCSRB = _BV(TXEN) | _BV(RXEN); /* tx/rx enable */
}
```