Embedded pool

Module02: UART Protocol

contact@42chips.fr

Summary: Kickstart my UART

Version: 1

Chapter I

Introduction

International Morse code (or the International Morse alphabet) is a code that allows text to be transmitted using short and long pulses, whether produced by signs, light, an electrical signal, sound, or gesture.

This code is often attributed to Samuel Morse, but several people dispute this and tend to attribute the language's paternity to his assistant, Alfred Vail.

Invented in 1832 for telegraphy, this character coding assigns a unique combination of intermittent signals to each letter, number, and punctuation mark.
Morse code is considered the precursor of digital communications.

Morse code is primarily used by the military as a means of transmission, often encrypted, as well as in the civilian world for certain automatic broadcasts, such as:

- radio beacons in aviation,
- call sign of maritime stations,
- international emitters (atomic clocks),
- or for maritime signaling by certain radar transponders.

Morse code is also practiced by amateurs such as:

- radio amateurs,
- scouts (audible and visual Morse code),
- divers and mountaineers (visual Morse code),
- puzzle solvers,
- as well as the default message reception ringtone for Nokia mobile phones ('SMS SMS' in Morse code).

Chapter II

General instructions

Unless explicitly stated otherwise, the following instructions will be valid for all assignments.

- The language used for this project is C.
- It is not necessary to code according to the 42 norm.
- The exercises are ordered very precisely from the simplest to the most complex. Under no circumstances will we consider or evaluate a complex exercise if a simpler one is not perfectly successful.
- You must not leave any files other than those explicitly specified by the exercise instructions in your directory during peer evaluation.
- All technical answers to your questions can be found in the `datasheets` or on the Internet. It is up to you to use and abuse these resources to understand how to complete your exercise.
- You must use the datasheet of the microcontroller provided to you and comment on the important parts of your program by indicating where you found the clues in the document, and if necessary, explaining your approach. Don't write long blocks of text, keep it clear.
- Do you have a question? Ask your neighbor to the right or left. You can ask in the dedicated channel on the Piscine's Discord, or as a last resort, ask a staff member.

Chapter III

Write

	Exercise 00
write(2)	
Turn-in directory : *ex*00/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	

- The AVR ATmega328P microcontroller has 1 UART device that you must use in this exercise to communicate with a computer.
- On the PC, the `screen` program is used to read the serial port from a terminal.
- You must write a function `uart_init` that initializes the UART.
- A function `uart_tx` that writes a character to the PC's serial port.
- The MCU's UART must be configured as 115200 8N1.
- UBRRnL must be calculated based on `UART_BAUDRATE` and `F_CPU`.
- The program should write 'Z' to the serial port at 1Hz (do as you wish).

```
void uart_tx(char c);
```

	Exercise 01
	print_str
Turn-in directory : *ex*01/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	

- You must write a function `uart_printstr` that will be called every 2 seconds to display `Hello World!` on the serial port.
- The MCU's UART must be configured to 115200 8N1.
- The infinite loop of the program should remain empty.

```
void uart_printstr(const char* str);
```

```
Hello World!
Hello World!
Hello World!
...
```

Chapter IV

Read

	Exercise 02
read(2)	
Turn-in directory : *ex*02/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	

- Now you will have to write a function `uart_rx` that waits to receive a character on the MCU's UART port and then returns it.
- You must write a program that uses your `uart_rx` function.
- It should write the received characters from `uart_rx` to the serial port using your `uart_tx` function (ex00).

```
char uart_rx(void);
```

	Exercise 03
Uppercase/Lowercase	
Turn-in directory : *ex*03/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	

- You must write a program that sends an echo on the serial port, but transforms lowercase characters into uppercase and uppercase characters into lowercase before sending them back.
- Attention, this time instead of using your `uart_rx`, you must use an interruption to detect that a new character is on the UART port.
- The infinite loop of the program must remain empty.

Chapter V

Bonus: WOPR

	Exercise 04
Login	
Turn-in directory : *ex*04/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, util/delay.h, avr/interrupt.h`	

- Create 2 strings, a username and a password.
- Display a prompt on the serial port that asks for the username and password.
- When typing the username, there should be an echo.
- same for the password but with '*' instead of characters.
- The `Backspace` key deletes a character.
- The `Enter` key validates the input.
- If the username and password are correct, the program displays the welcome text and makes the LEDs blink.

`Bonus points if you add a dramatic effect. ;)`

- Otherwise, the program displays the error message.

```
Enter your login:
   username: spectre
   password: ******
Bad combination username/password

Enter your login:
   username: spectre
   password: ******
Hello spectre!
Shall we play a game?
```