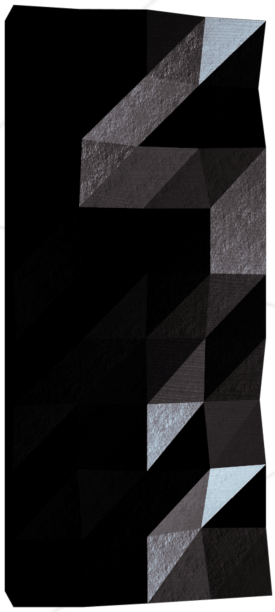

Embedded pool

Module07 : EEPROM

contact@42chips.fr

Summary: Because writing too many times at the same spot is what made cars crash

Version: 1

Chapter I

Introduction

An EEPROM (Electrically Erasable Programmable Read-Only Memory) is a type of non-volatile memory that can be used to store data on a microcontroller.

It can be written to and erased multiple times, and retains its data when the power is turned off.

EEPROMs are useful for storing data that needs to be retained even when the microcontroller is not powered, such as configuration settings or calibration data.

They are slower and have less capacity than other types of memory, such as SRAM or flash memory, but are still useful in a variety of applications.

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

Tutorial

	Exercise 00
What a mess!	
Turn-in directory : *ex*00/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, avr/eeprom.h`	

Write a program that displays the status of the entire EEPROM on the UART in hexdump format.

```
0000000 6f43 7262 3a61 5020 4948 4c4c 5049 2045
0000010 2121 2021 654a 7320 6961 2073 756f 7420
0000020 2075 6574 6320 6361 6568 2073 0a21 6956
0000030 6e65 2073 6369 2069 7571 2065 656a 7420
0000040 2065 7562 6574 6520 636e 6c75 656c 2120
0000050 2121 500a 6968 6c6c 7069 3a65 5420 2061
0000060 7567 7565 656c 2120 2121 000a 0000 0000
...
```

`Do not assume the EEPROM is set at 0 when you didn't write to it yet.`

	Exercise 01
Sniper !	
Turn-in directory : $ex01/$	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, avr/eeprom.h`	

`The number of write cycles on an EEPROM is limited, but not the number of reads..`

Write a command-line program on the UART.

When given a valid address on the EEPROM in hexadecimal it writes a hexadecimal byte to the EEPROM, then displays the EEPROM contents with the byte in red.

If the byte already had this value, nothing should happen.

```
00000010 7f
00000000 4a6f 686e 2046 6974 7a67 6572 616c 6420
00000010 7f65 6e6e 6564 792c 2064 6974 204a 6163
00000020 6b20 4b65 6e6e 6564 792c 0a63 6f6d 6d75
00000030 6ec3 a96d 656e 7420 6170 7065 6cc3 a920
00000040 4a6f 686e 2046 2e20 4b65 6e6e 6564 7920
00000050 6574 2070 6172 2073 6573 2069 6e69 7469
00000060 616c 6573 204a 464b 2c0a 6ec3 a920 6c65
00000070 2032 3920 6d61 6920 3139 3137 20c3 a020
00000080 4272 6f6f 6b6c 696e 6520 284d 6173 7361
00000090 6368 7573 6574 7473 290a 6574 206d 6f72
000000a0 7420 6173 7361 7373 696e c3a9 206c 6520
000000b0 3232 206e 6f76 656d 6272 6520 3139 3633
000000c0 20c3 a020 4461 6c6c 6173 2028 5465 7861
000000d0 7329 0a00 0000 0000 0000 0000 0000 0000
...
```

Chapter IV

This is not EEPROMalloc

	Exercise 02
Um Ackchyually	
Turn-in directory : *ex*02/	
Files to turn in : `Makefile, *.c, *.h`	
Allowed functions : `avr/io.h, avr/eeprom.h`	

`We strongly advise you to use magic numbers to indicate the presence of your data in the EEPROM. It is also best to use a magic number that's not in the standard the standard ASCII table. For example 7f.`

Write a command line interface on your microcontroller's UART port.

It can store a key/value pair of strings up to 32 standard ASCII characters long, which must not be lost on reboot.
It can take 4 commands:

- `READ` : Takes a key, retrieves the associated value and displays it. If the pair does not exist, return `empty`.
- `WRITE` : Takes a key and a value, attempts to store them in the remaining EEPROM space without overwriting any pairs that have not been deleted. If successful, returns the address in hexadecimal, otherwise returns no space left. If the pair already exists, return already exists.
- `FORGET` : Take a key, delete the key/value pair if found. Otherwise returns not found.
- `PRINT` : Displays EEPROM contents in hexdump -C format.

```
> READ "lol"
empty
> WRITE "lol" "i don't know"
done
> READ "lol"
"i don't know"
> PRINT
00000000 1f4c 6120 7069 7363 696e 6520 656d 6265 | The embedded po|
00000010 6464 6564 2063 2765 7374 2074 726f 7020 |l is so much fu |
00000020 6269 656e 2021 0a00 0000 0000 0042 0000 |n !.      B  |
...
```