WHERE IS MY PROGRAM?

- Does my microcontroller run C language code?
- Where does it store its instructions?
- How does the microcontroller know where to begin?
- What happens during a reset?
- How does a computer *program* a microcontroller
- Are there going to be burgers in today's class?

MICROCONTROLLER PROGRAMMING

Daniel Martínez

I7266 - Programación de Sistemas Embebidos

CUCEI - UDG

MEMORY

Do you remember?

The 21st night of September?

Love was changing the minds of pretenders

While chasing the clouds away

MEMORY CLASSIFICATIONS

- Volatility
- Access method
- Read/write characteristics
- Physical location
- Functionality
- Technology
- Usage in CPU
- Data organization

VOLATILITY

- Volatile Memory:
  - Loses its stored information when power is turned off. Examples include RAM (Random Access Memory).
- Non-Volatile Memory:
  - Retains data even when power is turned off. Examples include CDs, flash memory, and hard drives.

ACCESS METHOD

- Random Access:
  - Allows data to be accessed in any random order. (RAM)
- Sequential Access:
  - Requires accessing data in a sequential manner. Examples include tape drives and optical drives.

READ/WRITE CHARACTERISTICS

- Read-Only:
  - Typically used for storing permanent or semi-permanent data, and the data cannot be easily modified. Examples include some optical drives.
- Read/Write:
  - Allows both reading and writing of data. Examples include RAM and Flash memory.

PHYSICAL LOCATION

- Internal Memory:
  - Embedded within the microprocessor or closely connected to it. Examples include Cache memory or SRAM.
- External Memory:
  - Separate from the microprocessor and connected via buses. Examples include EEPROM, hard drives, etc.

FUNCTIONALITY

- Primary Memory:
  - Fast and used for temporary storage (e.g., RAM).
- Secondary Memory:
  - Slower but provides larger storage capacity (e.g., hard drives).

TECHNOLOGY

- Dynamic Random Access Memory (DRAM):
  - Requires periodic refresh to maintain data integrity.
- Static Random Access Memory (SRAM):
  - More stable and does not require frequent refreshing.
- Electrically Erasable Programmable ROM (EEPROM):
  - Non-volatile but needs UV light to be erased.
- Flash Memory:
  - Non-volatile memory commonly used in storage devices.

USAGE INSIDE CPU

- Cache Memory:
  - High-speed memory used by the CPU to store frequently accessed instructions and data.
- Registers:
  - Fast, small, and internal memory locations within the CPU.

HELLO WORLD!

Our first program

NOT SO FAST…

THE MINIMAL SYSTEM

What is needed for a microcontroller to work?

A minimal system refers to the basic set of components required to create a functional electronic system.

The components in a minimal system may vary depending on the specific application, for embedded systems, some common elements include:

- **Microcontroller or Microprocessor:**
  - The central processing unit responsible for executing instructions and controlling the overall operation of the system.
- **Clock Source:**
  - A clock oscillator or crystal to provide the system with a timing reference, ensuring synchronization of operations.

- Memory: This includes both volatile memory (RAM) and non-volatile memory (ROM, Flash), for storing data and program instructions, respectively.
- Input/Output (I/O) Components: These could be sensors, actuators, communication interfaces, or other peripherals that allow the system to interact with its environment.

- Power Supply: A source of electrical power to provide the necessary voltage levels for the components in the system.
- Basic Support Components: Resistors, capacitors, and other passive components that are necessary for circuit stability and functionality.

ATMEGA328P MINIMAL SYSTEM

POWER SUPPLY

- VCC
- GND
- AVCC
- AREF

CLOCK SOURCE

- Low Power Crystal Oscillator
- Full Swing Crystal Oscillator
- Low Frequency Crystal Oscillator
- Internal 128kHz RC Oscillator
- Calibrated Internal RC Oscillator
- External Clock

RESET

HELLO WORLD!

No BS this time!

HELLO WORLD!

whut???

```c
#define F_CPU 16000000U

#include <avr/io.h>

int main(void)
{
    DDRD &= ~(1 << PORTD2);

    while (1)
    {
        PORTD |= (1 << PORTD2);
    }

}
```

TALKING COMPUTER LANGUGE

HEXADECIMAL

- The hexadecimal numbering system (*hex* for short) is used to represent *nibbles*.
  - Knowledge check:
    - A **bit** is a binary digit, i.e., *1* or *0*.
    - A **nibble** is a group of four bits, e.g.,*1011*.
    - A **byte** is a group of two nibbles (or eight bits), e.g., *11001111*
- Hexadecimal uses the first ten digits, plus, A, B, C, D, E and F to represent quantities. Let’s see how it is used.

HEXADECIMAL

- In *hex* after the first ten numbers (from 0 to 9), the decimal values 10, 11, 12, 13, 14 and 15 are represented by A, B, C, D, E and F, respectively.
- Since four digits in binary can represent up to 15 (the binary number *1111*) then a single hex character can represent four bits.
- For example, the binary number *1100*, (a.k.a. 12 in decimal) is represented by the character *C*.
- This means we can represent a full byte with just two hex characters!

HEXADECIMAL

- To represent a byte in the common computer science slang, for example, 10100101 in binary (or 165 in decimal) we use the hexadecimal notation 0xA5, where:
  - “0x” means “we are talking hexadecimal”
  - “A” represents the first four bits (from left to right) in binary:
    - 1010 in binary is 10 in decimal. Since it is in the second character position, its value in decimal is multiplied by 16. (10·16=160)
  - “5” represents the last four bits (least significative) in binary:
    - 0101 in binary is, unsurprisingly, 5 in decimal.

MEMORY

One more time...

WE ALREADY KNOW… DON’T WE?

- We have already seen different types of memory used in computer systems, however, we have yet to learn the uses for these kind of memory.
- Of all the classifications we explored, there are four key memory types:
  - ROM
  - RAM
  - Registers
  - Cache

ROM

Back to the burgers...

- In our “big company” examples, we know we must follow certain steps to achieve a desired output, these steps usually don’t change. In computer systems, the place where we store our instructions is called ROM or Program Memory.
- ROM, in embedded systems, is located inside the microcontroller and it is not usually updated. It is intended to be programmed once and to need the very least number of updates possible.
- ROM is a type of non-volatile memory, since we need it to keep its contents for a long time.

RAM

More RAM, more FPS.

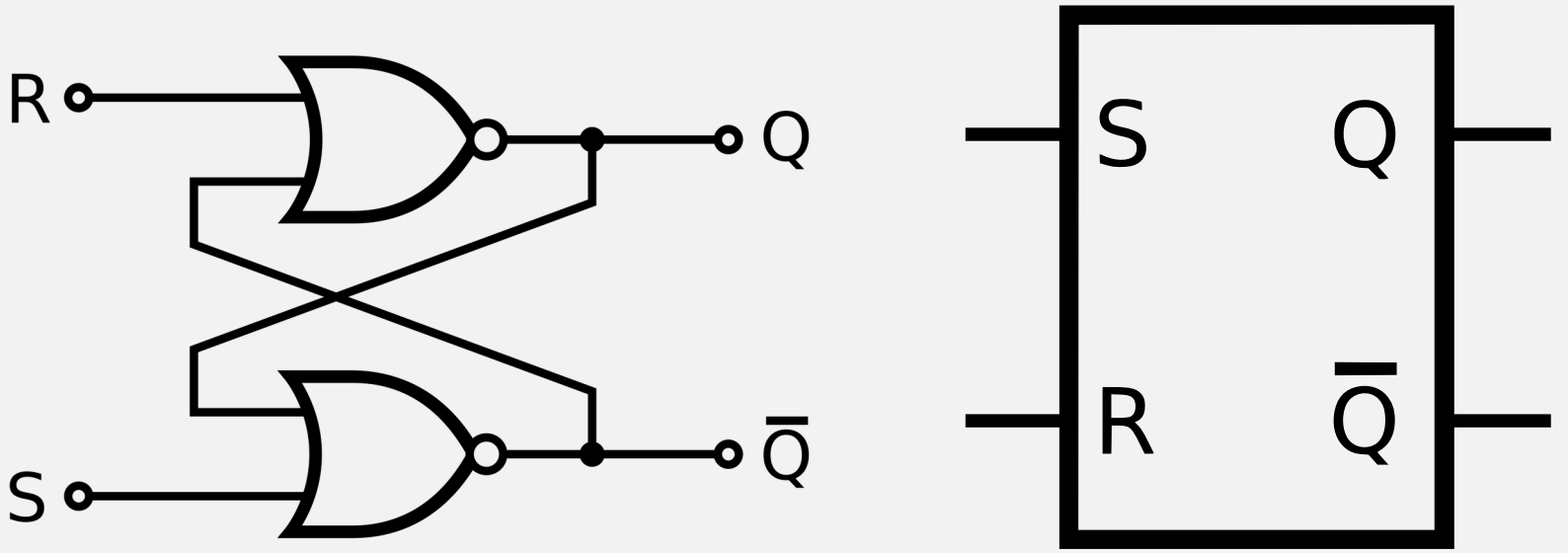
- RAM is our temporary storage; in our “big company” we need places to do what we are doing.
- We cannot prepare a burger on our hands, (and if we can, we should not), we need at least a table where we can place our ingredients.
- RAM, in embedded systems, is also located inside the microcontroller and it is written all the time while the microcontroller is working, it stores the data we need during the execution of a program.
- RAM is a usually a kind of volatile memory.

REGISTERS

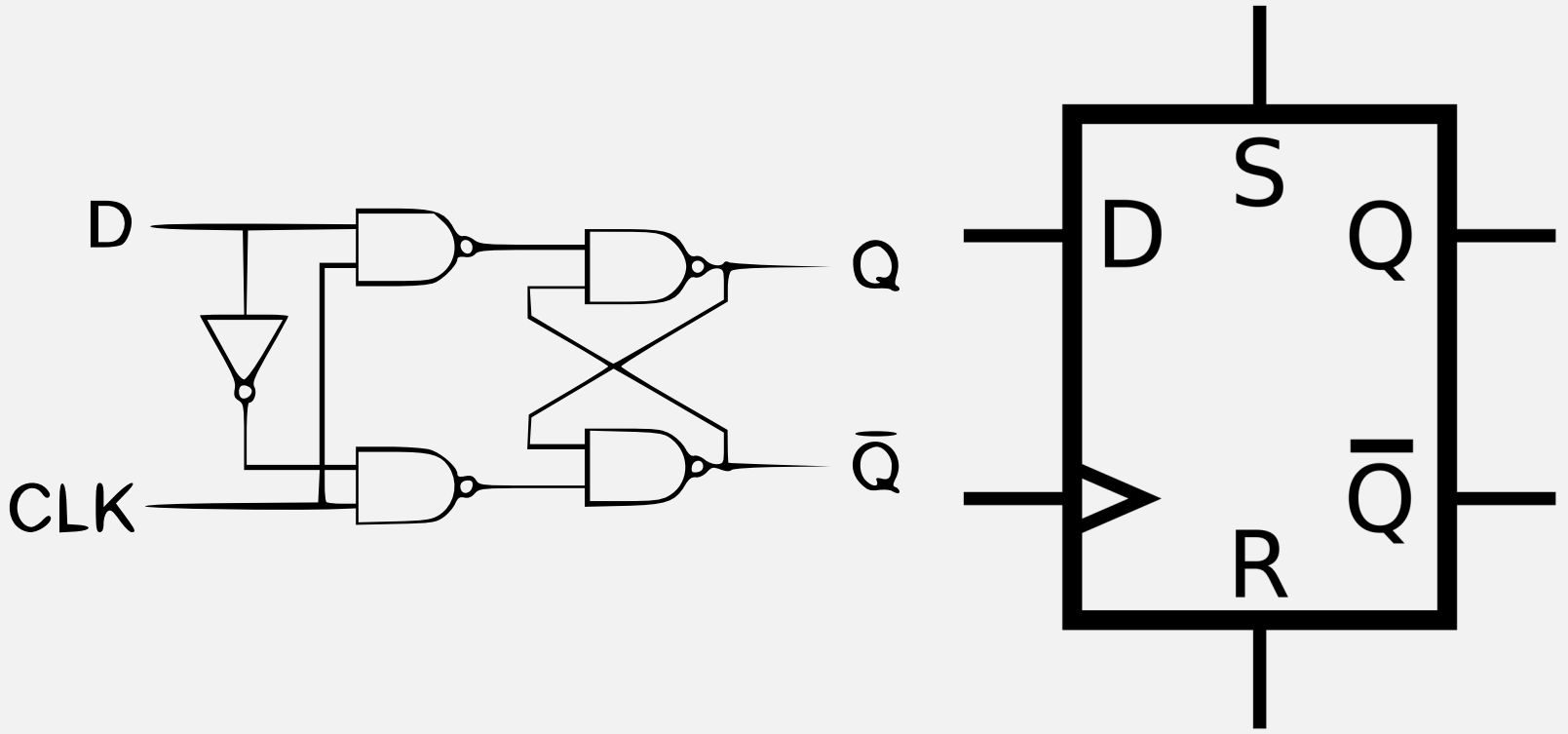
More RAM, more FPS.

- The registers inside a microcontroller or microprocessor are small memory cells that are meant to hold some very important data that is useful for general purposes.
- In our “big company” these are like the boards where we place important company updates.
- Registers are usually implemented with some special logic cells like latches or flip-flops.
- When using the microcontroller, we will understand better how these registers are used and why.

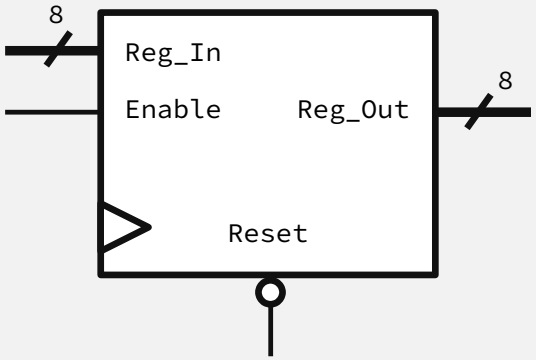
SR LATCH

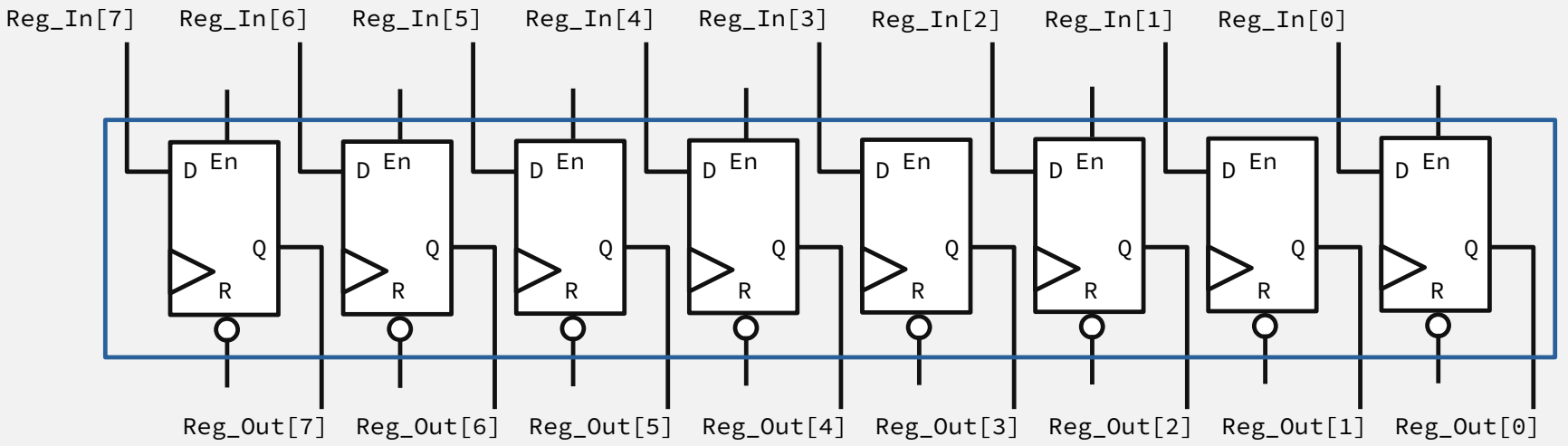


D FLIP-FLOP



8-BIT REGISTERS





ADDRESSES

Where you at rn?

ADDRESSES AND RELATED CONCEPTS

- A memory address is a unique identifier for a specific location in the microcontroller's memory.
- We use addresses to identify *addressable units*. In most systems the smallest addressable unit is the byte, so, each memory cell can store a byte.
- The *address space* is the range of possible memory addresses that out control unit can generate.
- *Word Addressing* is accessing to memory with the addresses for words, this means, groups of bytes of a predefined size.
- *Endianness* refers to the order of multi-byte datatypes. We have *Little-Endian* and *Big-Endian*.

THE MEMORY MAP

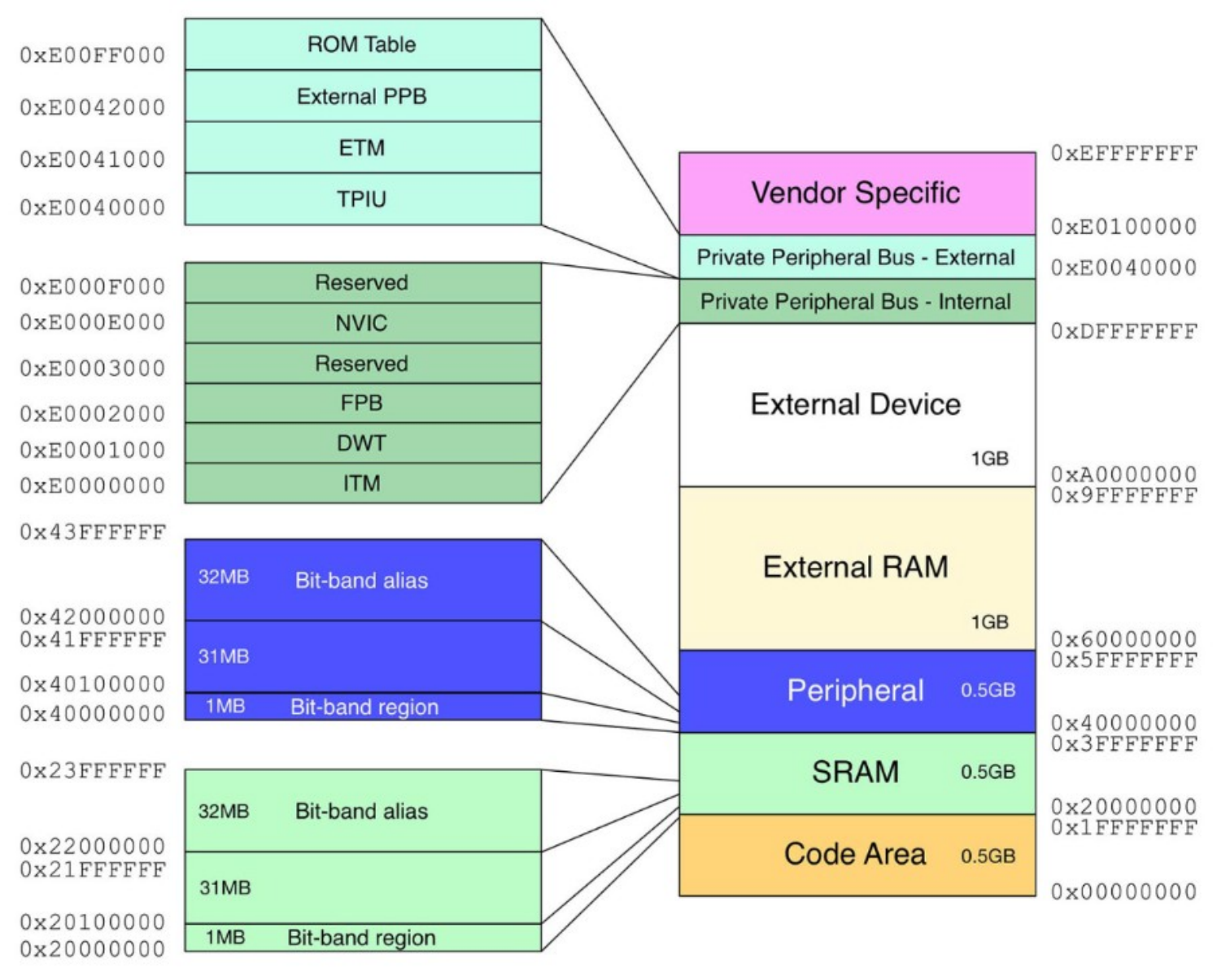
WHAT IS A MEMORY MAP?

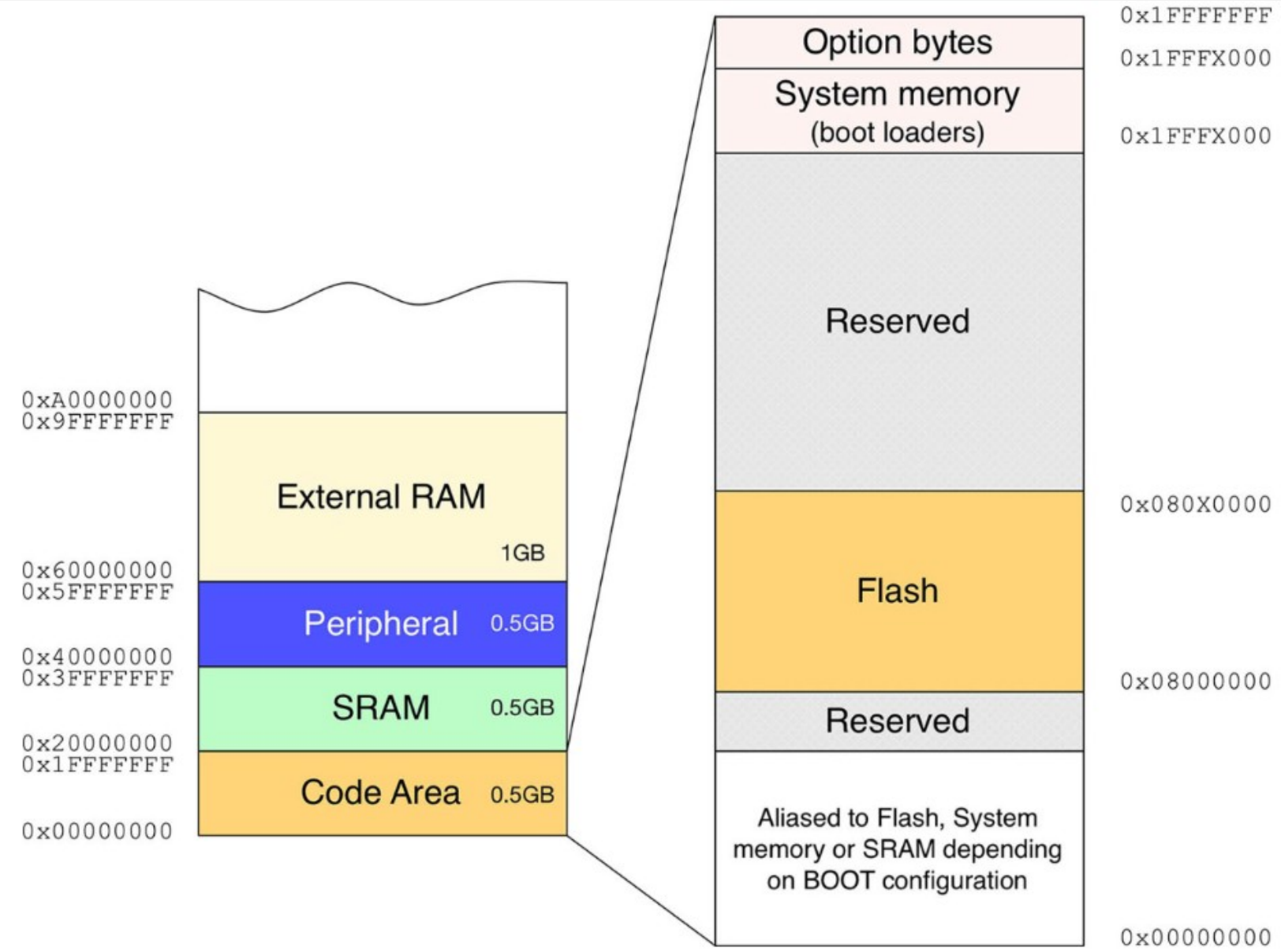
- A memory map is the way a to describe how a computing system has its memory organized.
- It can represent different physical memory devices into one single address space.
- It can also represent multiple regions of a single memory device into the address map.
- We usually care about certain special *memory regions* that we will cover in detail later, this is the overview:

MEMORY REGIONS

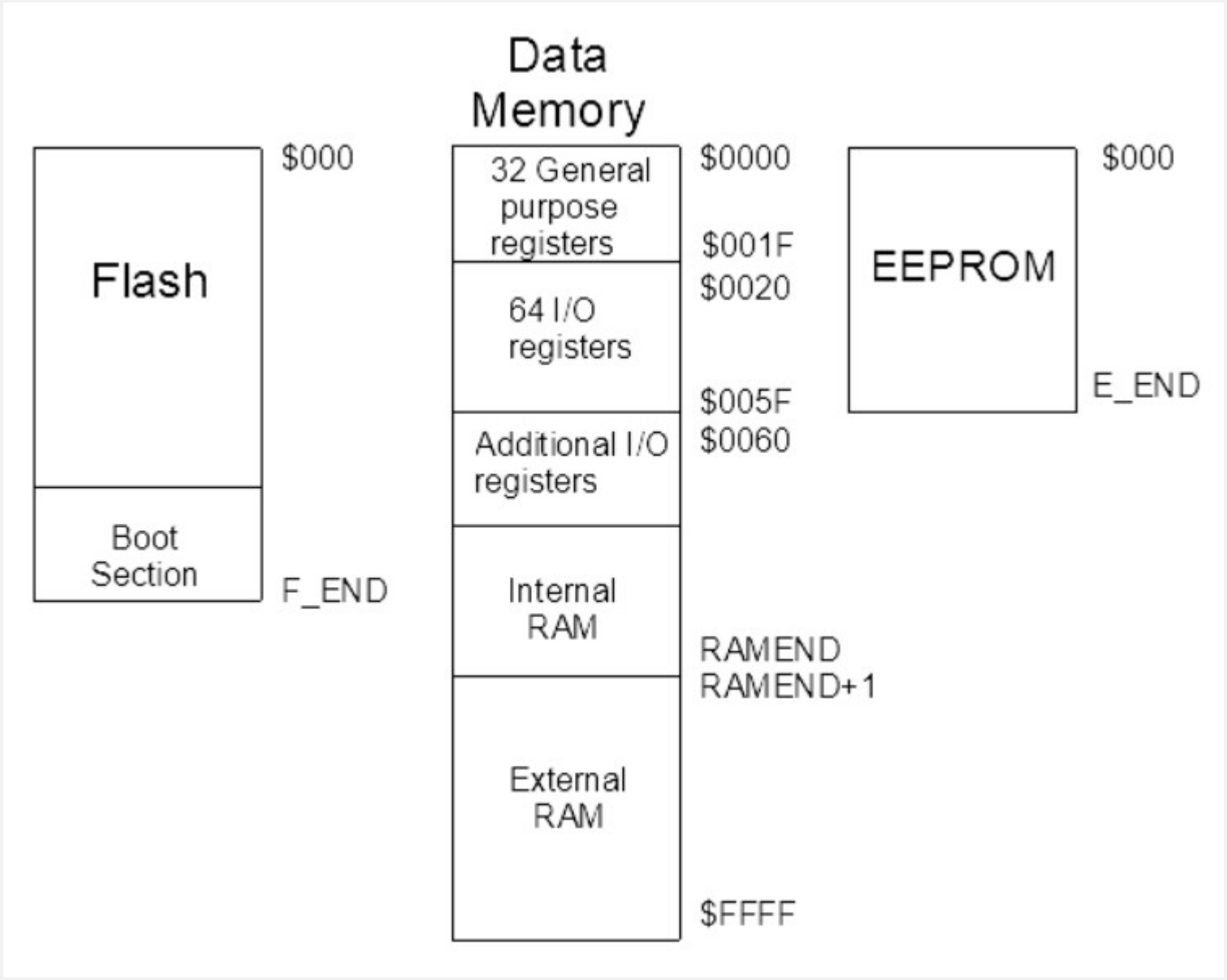
- Program memory
  - This is the region where the code is stored, this refers to the ROM device.
- Data memory
  - This is the region where variables and constants are stored, this is usually in the RAM device.
- Registers
  - These are also mapped here.

ARM CORTEX-M MEMORY MAP

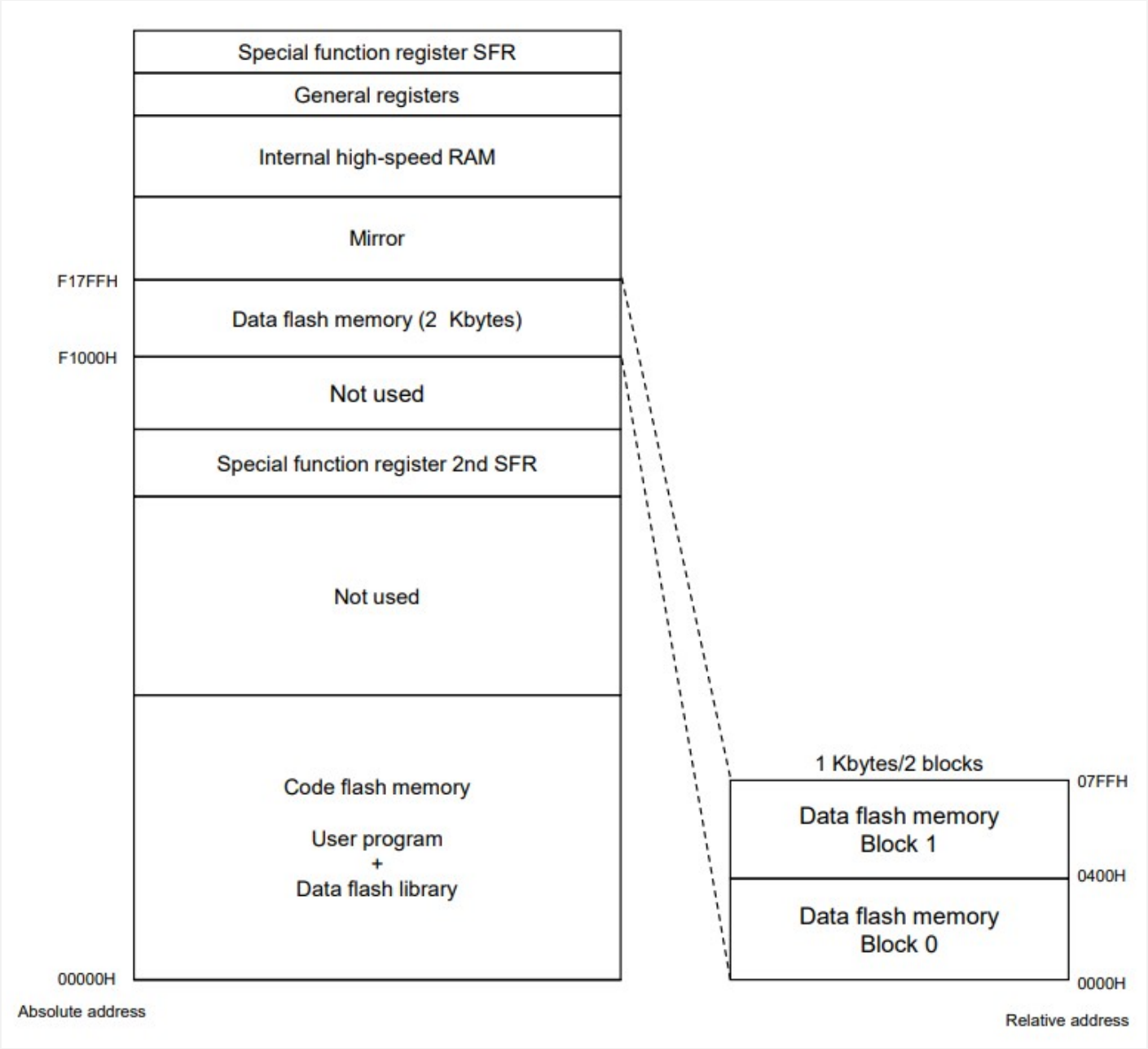

0xE00FF000
ROM Table
0xE0042000
External PPB
0xE0041000
ETM
0xE0040000
TPIU
0xE000F000
Reserved
0xE000E000
NVIC
0xE0003000
Reserved
0xE0002000
FPB
0xE0001000
DWT
0xE0000000
ITM
0x43FFFFFF
32MB
Bit-band alias
0x42000000
0x41FFFFFF
31MB
0x40100000
1MB
Bit-band region
0x40000000
0x23FFFFFF
32MB
Bit-band alias
0x22000000
0x21FFFFFF
31MB
0x20100000
1MB
Bit-band region
0x20000000
Vendor Specific
0xEFFFFFFF
0xE0100000
Private Peripheral Bus - External
0xE0040000
Private Peripheral Bus - Internal
0xDFFFFFFF
External Device
1GB
0xA0000000
0x9FFFFFFF
External RAM
1GB
0x60000000
0x5FFFFFFF
Peripheral
0.5GB
0x40000000
0x3FFFFFFF
SRAM
0.5GB
0x20000000
0x1FFFFFFF
Code Area
0.5GB
0x00000000


0xA0000000
0x9FFFFFFF
External RAM
1GB
0x60000000
0x5FFFFFFF
Peripheral
0.5GB
0x40000000
0x3FFFFFFF
SRAM
0.5GB
0x20000000
0x1FFFFFFF
Code Area
0.5GB
0x00000000
Option bytes
System memory
(boot loaders)
Reserved
Flash
Reserved
Aliased to Flash, System memory or SRAM depending on BOOT configuration
0x1FFFFFFF
0x1FFFX000
0x1FFFX000
0x080X0000
0x08000000
0x00000000

AVR MEMORY MAP


Data Memory
Flash
Boot Section
$000
F_END
32 General purpose registers
64 I/O registers
Additional I/O registers
Internal RAM
External RAM
$0000
$001F
$0020
$005F
$0060
RAMEND
RAMEND+1
$FFFF
EEPROM
$000
E_END

AVR MEMORY MAP


Special function register SFR
General registers
Internal high-speed RAM
Mirror
F17FFH
Data flash memory (2 Kbytes)
F1000H
Not used
Special function register 2nd SFR
Not used
Code flash memory
User program
+
Data flash library
00000H
Absolute address
1 Kbytes/2 blocks
07FFH
Data flash memory
Block 1
0400H
Data flash memory
Block 0
0000H
Relative address

NOTICES!

WhatsApp and classroom groups, missing data, class formality.

THANKS!

Please feel free to ask any related questions at any time.