ATMEGA328/Arduino Uno - c_memory
https://github.com/teaksoon/lmaewapm

MEMORY

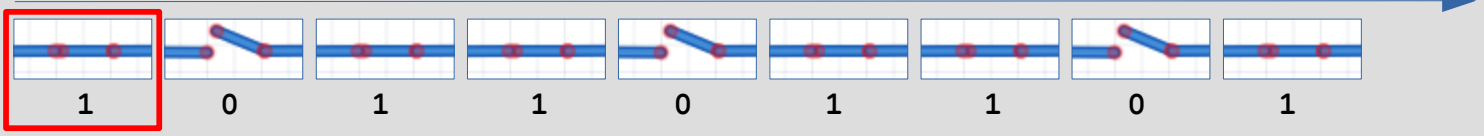
Inside the ATMEGA328 micro-controller chip, is a massive electrical circuit. Bulk of the massive circuit are "switches" made from different types tiny transistors. They are mostly used as **"MEMORY", to store information**

We can visualize the "MEMORY" as a long sequence of "switches" that can be turned ON or OFF (each represented by 1 or 0)

A sequence of 1s and 0s a in specific order, represents a piece of information

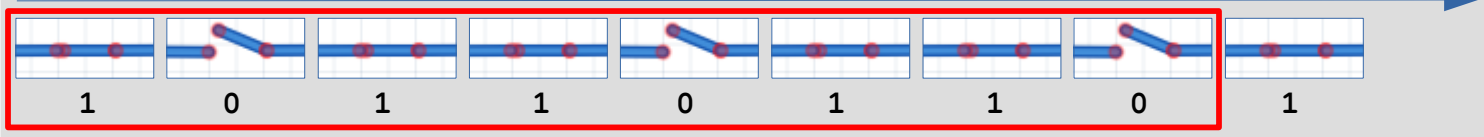
What are BIT and BYTE ?

BIT



Each physical "switch" in the "MEMORY" known as **a "BIT"**
Each "BIT" can be either **ON(represented by 1)** or **OFF(represented by 0)**

BYTE



A Group of **8-BITS** in sequence is known as **a "BYTE"**

Types of MEMORY in the ATMEGA328P micro-controller

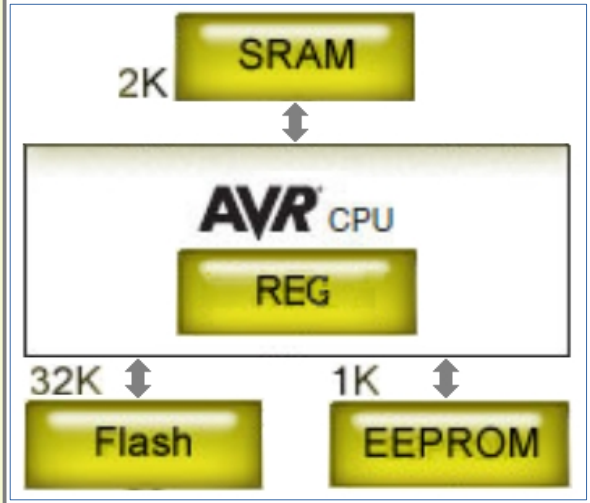
There are 4 TYPES of MEMORY inside the ATMEGA328P micro-controller chip

- CPU MEMORY (REGISTERS, our Regular Computer have this same thing **)**

- WORKING MEMORY (SRAM, similar to our Regular Computer RAM, but is inside the chip **)**

- PROGRAM MEMORY (FLASH, similar to our Regular Computer SSD/HARDDISK, but is inside the chip, for permanent storage of PROGRAM **)**

- STORAGE MEMORY (EEPROM, similar to our Regular Computer SSD/HARDDISK, but is inside the chip, for permanent storage of ANY DATA **)**



2K, 32K and 1K are in BYTE

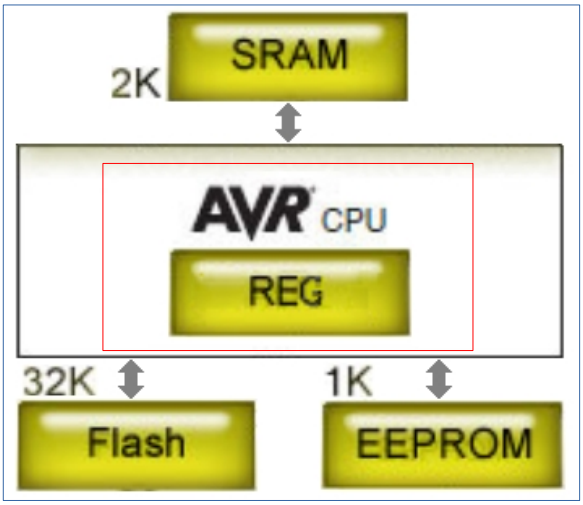
ATMEGA328/Arduino Uno – CPU MEMORY (REGISTER) –
https://github.com/teaksoon/lmaewapm

– CPU MEMORY (REGISTERS, our Regular Computer have this same thing **)**

This "MEMORY" is part of the micro-controller CPU (Central Processing Unit)

This MEMORY can be visualized as a long sequence of BITS. This long sequence of BITS has already been broken into many "BLOCKS of multiple BITS"

Each "BLOCK" is known as a "REGISTER"(refer Datasheet for each of them). The number of BITS in each "REGISTER" can differ. The REGISTERS in the ATMEG328P micro-controller are 8-BITS, that is why the ATMEGA328P is known as an 8-BIT MICRO-CONTROLLER



2K, 32K and 1K are in BYTE

The CPU MEMORY (REGISTER) are special, when they are changed they will normally cause the micro-controller perform a task. We can read or update the REGISTERS from our PROGRAM codes

In our earlier tutorial topic, you probably have seen the following code,

SBI 0x04, 5
"***SBI***" is an insruction from the "**AVR INSTRUCTION SET**"
"***0x04***" is the address of **an 8-BIT REGISTER (DDRB) from the CPU MEMORY**

When the CPU sees the code "***SBI 0x04, 5***" , it will **turn** the **5th BIT of the DDRB REGISTER to 1. When the 5th bit of DDRB REGISTER in the CPU MEMORY is 1**, the Arduino Pin 13(PB5) will become an **OUTPUT Pin**

NOTE: "SBI" instruction can only turn a BIT to 1, if you wish to turn the 5th bit of DDRB REGISTER to 0, we use a different instruction from the "AVR INSTRUCTION SET" on the same REGISTER. See the following line of code,

CBI 0x04, 5
"***CBI***" is an instruction from the "**AVR INSTRUCTION SET**"
"***0x04***" is the address of **an 8-BIT REGISTER (DDRB) from the CPU MEMORY**

When the CPU sees the code "***CBI 0x04, 5***" , it will **turn** the **5th BIT of the DDRB REGISTER to 0. When the 5th bit of DDRB REGISTER in the CPU MEMORY is 0**, the Arduino Pin 13(PB5) will become an **INPUT Pin**

Refer to the ATMEGA328P datasheet for the rest of the REGISTER and what each of its BITS are used for

If we do not wish to use the "AVR INSTRUCTION SET", we can also manipulate the CPU MEMORY (REGISTER) directly by using the C-Language,

DDRB = DDRB | (1<<5);
C-Language PROGRAM code ***DDRB = DDRB | (1<<5);*** // same as ***SBI 0x04, 5***

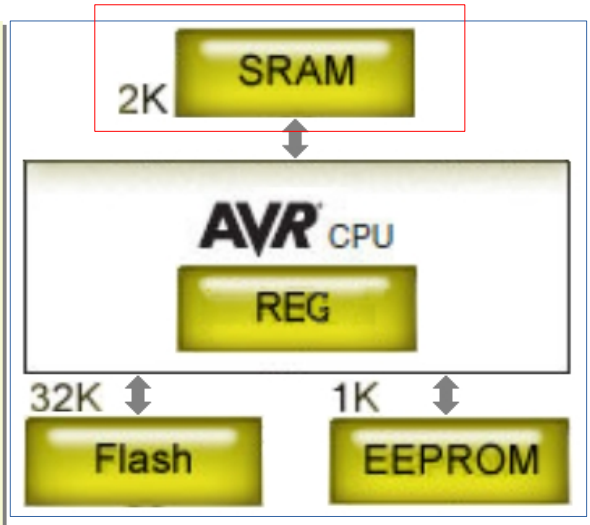
DDRB = DDRB & ~(1<<5);
C-Language PROGRAM code ***DDRB = DDRB & ~(1<<5);*** // same as ***CBI 0x04, 5***

ATMEGA328/Arduino Uno – WORKING MEMORY (SRAM) –
https://github.com/teaksoon/lmaewapm

– WORKING MEMORY (SRAM, similar to our Regular Computer RAM, but is inside the chip **)**

This MEMORY can be visualized as a long sequence of BITS

The "WORKING MEMORY" is for us to use in our PROGRAM as temporary working storage



2K, 32K and 1K are in BYTE

To use the WORKING MEMORY (SRAM) in our PROGRAM, we will need "Reserve in BLOCKS of multiple BITS". Each "BLOCK" of "Reserved MEMORY" is commonly known as a "variable". Each newly created "variable" is given a "name", "datatype" with some "properties/features".

The "datatype" will determine the number of BITS in each Variable, the C-Language basic "datatype" are,

char (8-BIT)
int (16-BIT, can be different for non 8-BIT micro-controllers)
Short (16-BIT, normally do not use this, it is slower than int)
long (32-BIT)
float (32-BIT)
double (32-BIT, for ATMEGA328, it is same as float, we use float)

signed (The stored number can have **Positive and Negative Numbers**)
unsigned (The stored number can have **Positive Numbers Only**)

When a "variable" is first created, it can be assigned with an initial value and after that, we can change the value from our PROGRAM. However, the Variable can also be created with different properites/features such as making it read only ("constant variable") and so on...

A Typical Usage of a "variable" from the WORKING MEMORY:

In the code below, we create a "variable" name "cycle_counter", which has 16-BITS in it. It is set to be able to store positive numbers only and we put an initial value of 0 into it

```
unsigned int cycle_counter = 0;
```

... later in other parts of our PROGRAM, we code the following...

In the code below, we change value stored in cycle_counter "variable" to 1

```
cycle_counter = 1;
```

The code below, we retrieve the value stored inside the cycle_counter "variable" and then add 1 to it

```
cycle_counter = cycle_counter+1;
```

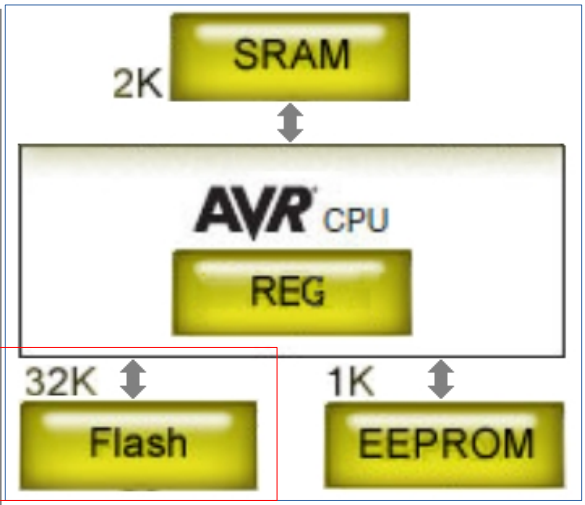
There are many ways to create and use "variable" – WORKING MEMORY in our PROGRAM, we will learn about them as we go along

ATMEGA328/Arduino Uno – PROGRAM MEMORY (FLASH) –
https://github.com/teaksoon/lmaewapm

– PROGRAM MEMORY (FLASH, similar to our Regular Computer SSD/HARDDISK, but is inside the chip, for permanent storage of PROGRAM **)**

This MEMORY can be visualized as a long sequence of BITS. The long sequence of BITS has already been broken into many BLOCKS of 8-BIT(BYTE)

This "MEMORY" is for us to store our PROGRAM permanently. Even when the micro-controller is powered-off, whatever stored in this memory remains



2K, 32K and 1K are in BYTE

ATMEGA328 PROGRAM MEMORY, Generally we do not care much about the PROGRAM MEMORY apart from monitoring our PROGRAM size so that it does not exceed what is physically available.

We normally just use this MEMORY to store our PROGRAM

To use this MEMORY, all we need to do is: Click "Upload" from our Arduino IDE Sofware, our PROGRAM will be automatically stored into the PROGRAM MEMORY. Our PROGRAM will stay there permanently until it is being replaced by our next "Upload"

PROGRAM MEMORY is a FLASH Memory, it has 10,000 "write-cycle" limit, meaning if this memory is "changed" for more than 10,000 times, new update into this MEMORY can become un-reliable. Since we only Upload PROGRAM into it, very unlikely we will need to upload more than 10,000 times

There is no limit on now many times we can read from this MEMORY

Apart from **our PROGRAM**, the PROGRAM MEMORY can also store another optional PROGRAM known as **"BOOTLOADER"**

- If a "BOOTLOADER" PROGRAM exist, the CPU will run the "BOOTLOADER" PROGRAM first when it is powered-up. Once the "BOOTLOADER" completes, the CPU will continue to run our PROGRAM

- If a "BOOTLOADER" PROGRAM does not exist, the CPU will run our PROGRAM immediately when it is powered-up

When we use the Arduino Uno Board, a "BOOTLOADER" PROGRAM is already exist in our ATMEGA328P micro-controller PROGRAM MEMORY (most likely is the "Optiboot" PROGRAM). The Arduino IDE Software needs to use the "BOOTLOADER" for the "Upload" process when using the USB connection

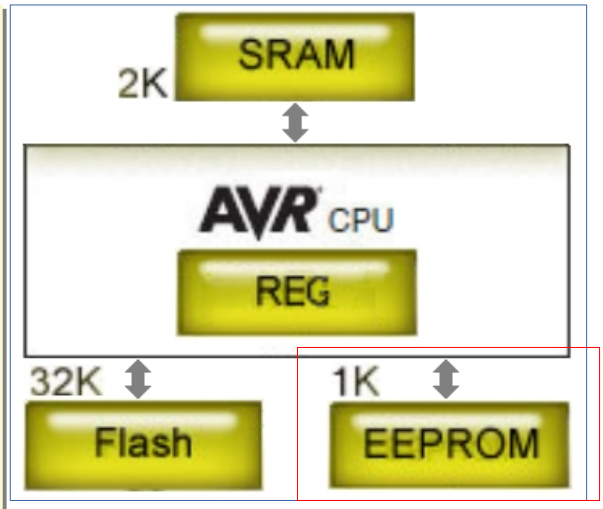
NOTE:
If we wish to use the PROGRAM MEMORY as WORKING MEMORY, we can still do it from our C-Language PROGRAM. However, we normally avoid writing to it from our PROGRAM because of the 10,000 "write-cycle" limit

ATMEGA328/Arduino Uno – STORAGE MEMORY (EEPROM) –
https://github.com/teaksoon/lmaewapm

– STORAGE MEMORY (EEPROM, similar to our Regular Computer SSD/HARDDISK, but is inside the chip, for permanent storage of ANY DATA **)**

This MEMORY can be visualized as a long sequence of BITS. The long sequence of BITS has already been broken into many BLOCKS of 8-BIT(BYTE)

This "MEMORY" is for us to store ANY DATA permanently. Even when the micro-controller is powered-off, whatever stored in this memory remains



2K, 32K and 1K are in BYTE

The STORAGE MEMORY (EEPROM) can be manipulated based on each BYTE index position, BYTE-0, BYTE-1, BYTE-2 and so on...

We can update this MEMORY from our C-Language PROGRAM codes

The EEPROM has a 100,000 "write-cycle" limit, meaning if this memory is "changed" for more than 100,000 times, new update into this MEMORY can become un-reliable.

There is no limit on now many times we can read from this MEMORY