**Microcontrollers**

Table of Contents

[AVR 4](#_Toc100172922)

[Categories 4](#_Toc100172923)

[Naming Convention 5](#_Toc100172924)

[ATMega16 6](#_Toc100172925)

[Features 6](#_Toc100172926)

[Pins 7](#_Toc100172927)

[Ports 8](#_Toc100172928)

[Oscillator 9](#_Toc100172929)

[CPU 9](#_Toc100172930)

[Status Register 10](#_Toc100172931)

[Timers and Counters 11](#_Toc100172932)

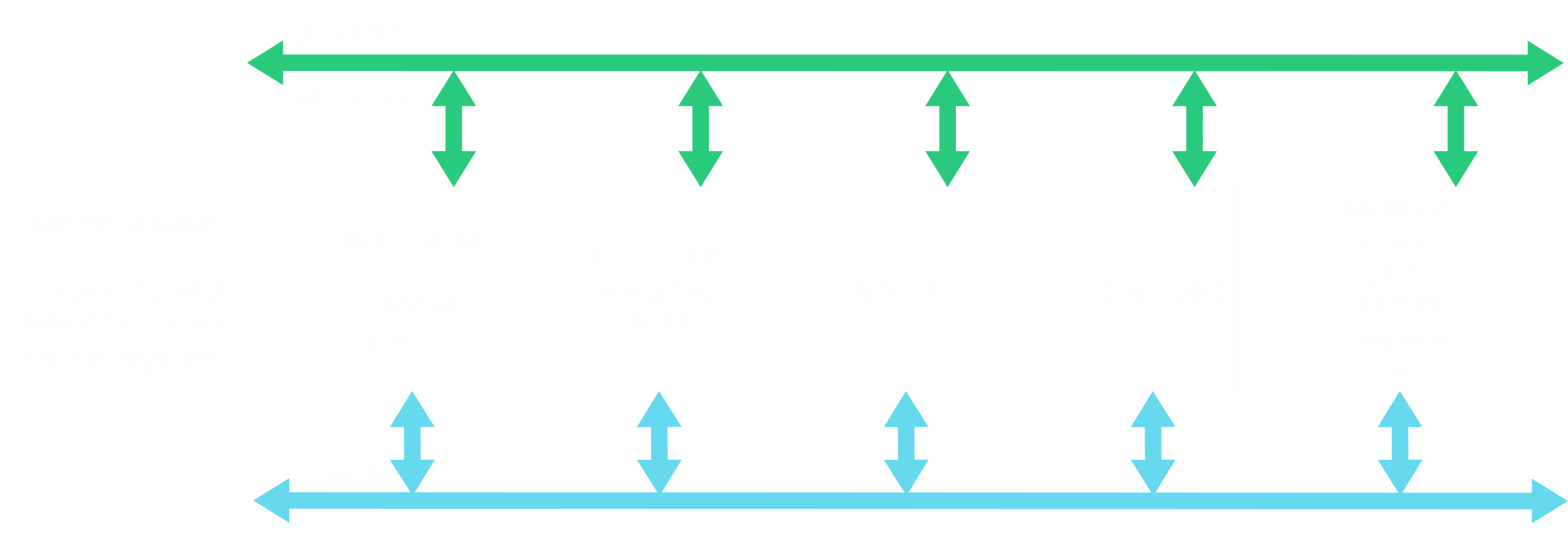
[USART 11](#_Toc100172933)

[Two-Wire Interface 11](#_Toc100172934)

[ADC Interface 12](#_Toc100172935)

[Serial Peripheral Interface 12](#_Toc100172936)

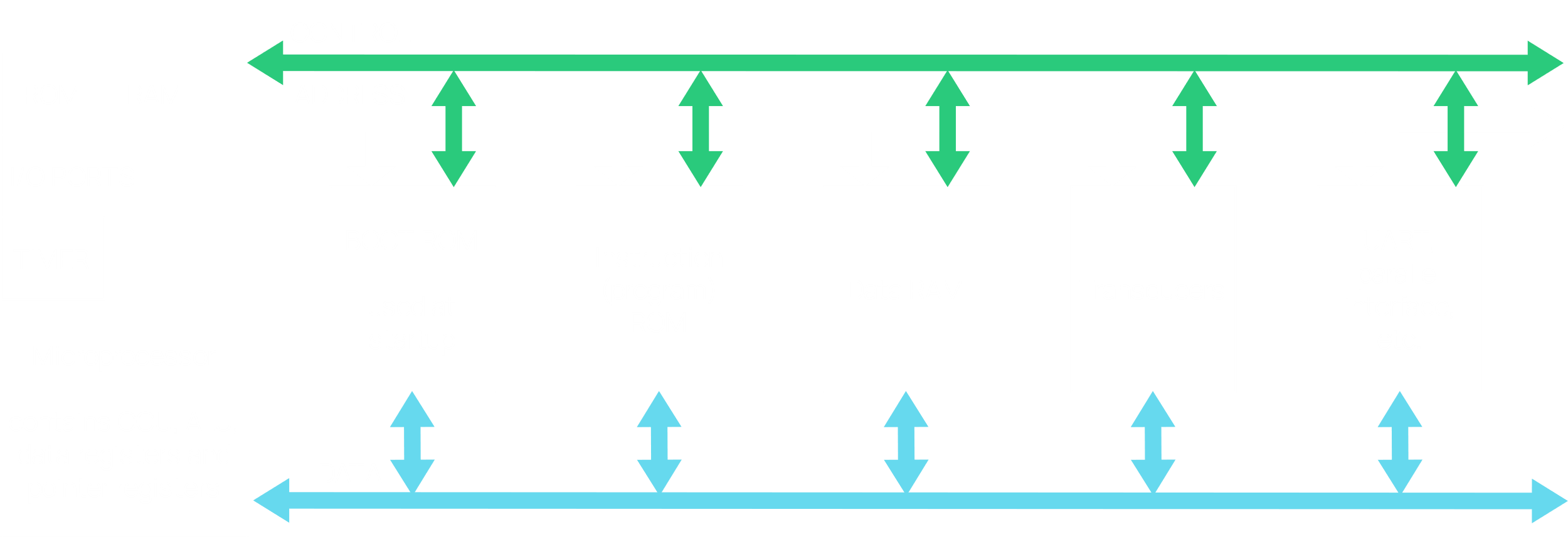
When we use a microprocessor, we need many different peripherals and interfaces to interact with. Each of these require a separate IC, all of which are placed on the **motherboard**. If we connect different ICs to each other, there are a total of interconnections. With an increasing number of ICs, the complexity of the interconnections thus increases exponentially.



For lightweight applications, such as simple sensors, this complexity is an unnecessary burden. A larger number of interconnections introduces lots of issues:

* Increased mechanical failure rates
* Increased design time
* Increased cost
* Increased board size
* Increased requirement for backwards compatibility

**Microcontrollers** aim to solve these limitations. A **limited amount** of the commonly used resources are placed directly inside the chip. This limited amount is relatively small, but it is often enough for simpler applications.



By doing this:

* The number of pins required is decreased
* The design time is decreased
* The board size is decreased
* The cost is decreased
* The reliability is increased
* The need for backwards compatibility is decreased, since it is easy to simply replace the entire microcontroller

|  |  |
| --- | --- |
| **Microprocessor** | **Microcontroller** |
| CPU is stand-alone. RAM, ROM, I/O and timers are separate. | CPU, RAM, ROM, I/O and timers are all on a single chip. |
| The designer gets to decide the amount of RAM, ROM and the number of I/O ports. | There is a fixed amount of RAM, ROM and number of I/O ports. |
| Expensive | Cost-effective |
| Versatile | Good for applications where cost, power and space are critical |
| General-purpose | Single-purpose |

## AVR

There are many different variants of microcontroller architectures. The one we will be using as reference is the **Atmel AVR**.

AVR stands for Alf Egil Bogen and Vegard Wollan’s RISC Processor, or Advanced Virtual RISC. It is an **8-bit**, **RISC**, **single-chip** microcontroller and uses a modified Harvard Architecture. Under the Harvard Architecture, the **program memory** and the **data memory** are stored separately with sperate physical address spaces.

The AVR was one of the first microcontroller families to use an **on-chip flash memory**, whereas other microcontrollers of the time used one-time programmable ROM, EPROM or EEPROM.

### Categories

These microcontrollers are available in three categories:

1. **Tiny AVR** – These have less memory and a smaller size. They are suitable for simpler applications.
2. **Mega AVR** – These are the most popular ones with a good amount of memory and a higher number of in-built peripherals. They are suitable for moderate to complex applications.
3. **XMega AVR** – These are used commercially for complex applications requiring a large program memory and high speed. They include support for DMA and event systems, meaning they can detect specific events, such as an unusually high room temperature, which could be caused by a fire.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **No. of Pins** | **Flash Memory Size** | **Special Features** |
| Tiny AVR | 6 – 32 | 0.5 – 8 KB | Small Size |
| Mega AVR | 28 – 100 | 4 – 256 KB | Extended Peripherals |
| XMega AVR | 44 – 100 | 16 – 384 KB | DMA, Event Systems |

### Naming Convention

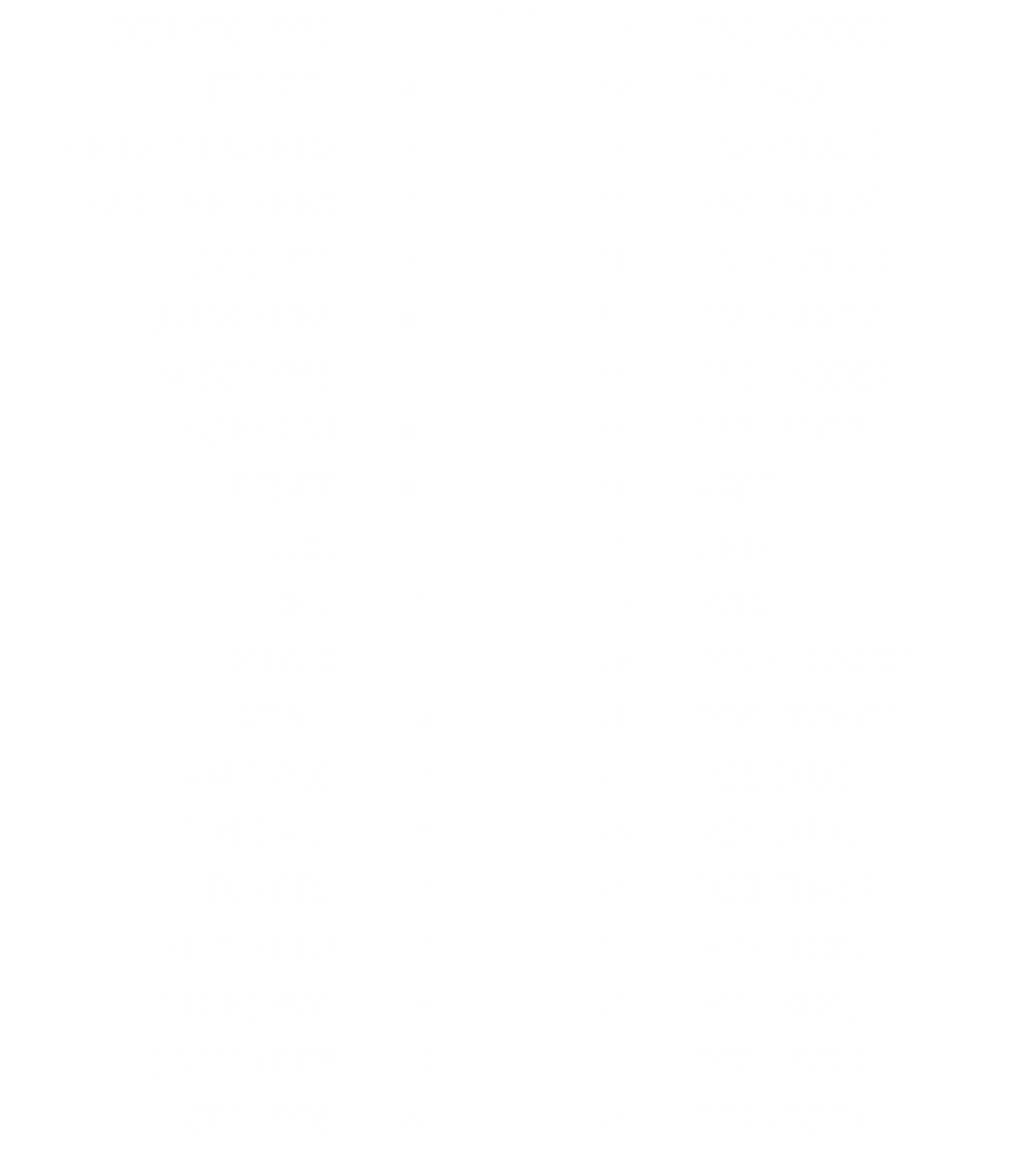
All AVR microcontrollers tend to use a specific naming convention. Consider the ATMega16 for example. It has three parts, AT referring to Atmel, the manufacturer, Mega, referring to the AVR category, and 16, the flash memory size.

## ATMega16

### Features

* 16 KB in-system self-programmable flash program memory
* 512 bytes EEPROM
* 1 KB internal SRAM
* 32 8-bit general purpose working registers
* 32 programmable I/O lines out of total 40 pins in DIP
* 8-channel, 10-bit ADC
* Two 8-bit timers or counters
* One 16-bit timer or counter
* Programmable serial USART
* 4 PWD channels
* Operating voltages between 4.5V and 5.5V
* Speed grades between 0 and 16 MHz

### Pins



There are four ports, , , and . We will discuss these shortly. There are also some generic pins like and , which we will skip over. The only ones being discussed here are the ones that are new:

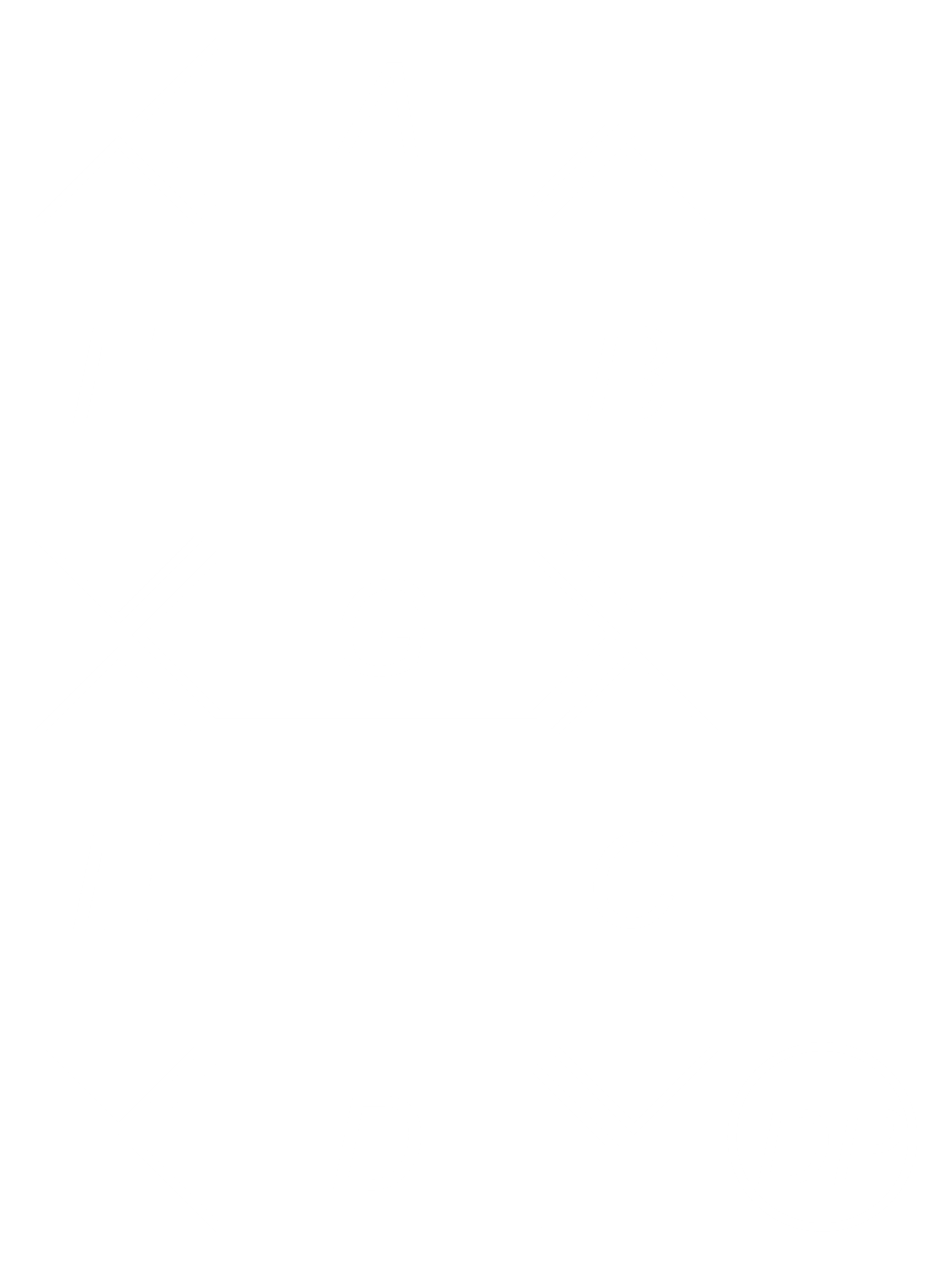
* – This pin supplies voltage to Port A and the A/D Converter.
* – This pin provides the analogue reference point for the A/D Converter.
* and – The ATMega16 has an internal oscillator, but external oscillators can be connected using these pins.

### Ports

There are four ports in the ATMega16, Ports A – D. Each port has 8 pins, meaning there are 32 pins in total.

**Port A** exists specifically to deal with **analogue data**. It connects to sensors and sends the data to the A/D Convertor.

**Ports B – D** work with **peripherals**. For example, say we connect a keyboard to the 8 pins of Port B. ASCII characters are 8-bit characters, so this is perfect. For the character ‘A’, we would need to send the values through pins to , from left to right.



We could also display this value in a 7-segment display connected to a different port, say Port C. There are 8 pins, so the left-most one is ignored, marked as DP. To display the character ‘A’, we need to light up the segments A, B, C, E, F and G. Thus, we can send the value through pins to respectively, corresponding to DP, G, F, E, D, C, B, A.

Similarly, if we wanted to display 2 characters at the same time, we would need to use another 7-segment display, connected to Port D.

Note that the peripherals are **directly connected** to the microcontroller, without any interface in between.

All the ports are 8-bit bi-directional I/O ports. Even port A serves as one when it is not being used with the A/D Converter. The ports B, C and D also serve the functions of various special features of the ATMega16.

### Oscillator

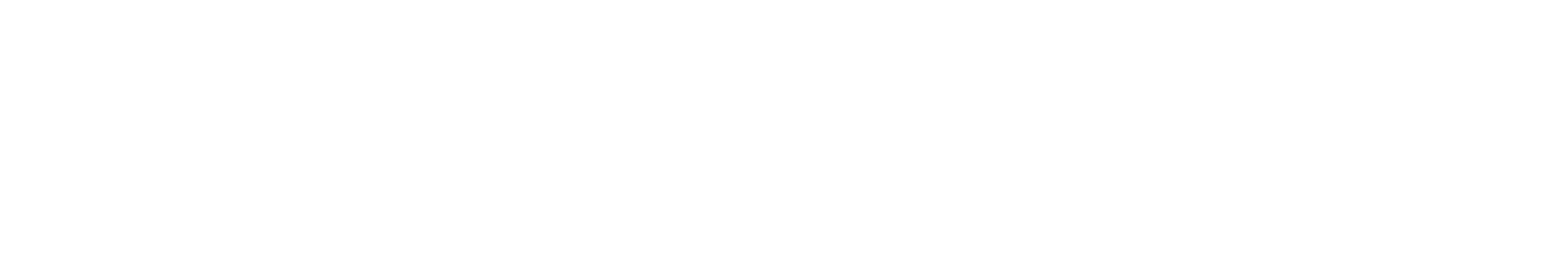
The ATMega16 can use an **internal** or **external clock signal**. By default, it uses the internal oscillator at . It can be programmed to use the internal one at , , or . If an external crystal oscillator is used, the maximum frequency is .

### CPU

The **CPU** includes:

* A program counter
* An instruction register
* An instruction decoder
* 32 general purpose registers (GPR), to , allowing 8-bit x 32 GPR pipelining
* An ALU
* A status register

### Status Register



* Carry Flag (C) – Indicates a carry caused by the operation.
* Zero Flag (Z) – Indicates a zero result.
* Negative Flag (N) – Indicates a negative result.
* Two’s Complement Overflow Flag (V) – Supports two’s complement arithmetic.
* Sign-Bit Flag (S) – This is the XOR of the N and V flags.
* Half-Carry Flag (H) – This indicates a half carry, like the auxiliary carry flag from microprocessors.
* Bit-Copy Storage Flag (T) – The Bit Copy instructions use this flag as the source/destination for the operated bit.
* Global Interrupt Flag (I) – Setting this flag enables interrupts.

### Timers and Counters

**Timers** and **Counters** are the most commonly used complex peripherals. The ATMega16 consists of two 8-bit and one 16-bit timer/counter.

Timers and counters can be thought of as **binary up-counters**. In timing mode, it counts time-periods. In counting mode, it counts events or pulses.

Timers are useful for generating precision actions, e.g. to create time delays between operations.

### USART

The **Universal Synchronous Asynchronous Receiver and Transmitter** (USART) is an interface that works with an external device capable of communicating **serially**, i.e. bit-by-bit data transmission.

### Two-Wire Interface

A **Two-Wire Interface** (TWI) can be used to set up a network of devices. Multiple devices can be connected over the TWI to form a network. The devices can simultaneously transmit and receive data using their own unique address.

### ADC Interface

The ATMega16 has an **8-channel ADC** with a resolution of 10-bits to convert analogue data from sensors to digital data.

### Serial Peripheral Interface

The **Serial Peripheral Interface** (SPI) is used for serial communication between two devices using a common clock. The SPI has a higher data-transmission rate than the USART.