

# Microprocessors and - controllers

- [audiobook of data sheet ATtiny212\\_412](#)
- [recitation](#)
  - [presentation](#)
- [literature explaining working with AVR](#)s
- [microcontroller vs -processor](#)
- [download platformio for vscode](#)

## MCU

An **MCU** is a group of **microcontroller units** that share a common architecture, design, and often the same core features

- **ARM Cortex-M** is a popular MCU family, where multiple manufacturers like **STMicroelectronics**, **NXP**, and **Microchip** produce MCUs that are based on the ARM Cortex-M core.
- **AVR** is another MCU family, famously used in the **Arduino** platform.
- **PIC** is a family of MCUs from **Microchip Technology**, with a range of devices from small 8-bit controllers to more powerful 16-bit and 32-bit versions.

Within each MCU family, you'll typically see models that differ by features like:

- **Flash memory size**
- **RAM size**
- **Clock speed**
- **Number of GPIO pins**
- **Peripheral support** (e.g., UART, SPI, I2C, timers, etc.)

### In-System Programming (ISP)

- the act of programming a microcontroller while it is already mounted on the board
- Contrary to Pre-Programming. You program the contro

CMSIS-DAP Devices are all devices that can write programs into a microcontroller's memory using JTAG or SWD. Ier before soldering it somewhere.

# Available Controllers

At least 5 of each:

- SEEDSTUDIO XIAO SAMD21 NO HDRS
- SEEDSTUDIO XIAO ESP32C3 NO HDRS
- SEEDSTUDIO XIAO ESP32S3 NO HDRS
- SEED STUDIO XIAO RP2040 ARDUINO
- ATSAM11C14A-SSUT
- ATSAM21E18A-AUT
- ATTINY412-SSFR
- ATTINY1624-SSFR
- ATTINY3226-SU
- AVR128DB32-I/PT

	Xiao RP2040	Pico RP2040	ESP32 C3	ESP32 S3	SAMD 21	ATSA MD11 C14A- SSUT	ATSA MD21 E18A- AUT	ATTIN Y412- SSFR	ATTIN Y1624 -SSFR	ATTIN Y3226 -SU	AVR1 28DB3 2-I/PT
Type	ARM Cortex M0+ 32Bit	ARM Cortex M0+ 32Bit	ESP- RISC-V 32Bit	Xtensa ® 32- bit LX7	ARM- Cortex M0+ 32Bit	ARM Cortex- M0+ 32Bit	ARM Cortex M0+ 32Bit	AVR® RISC 8- bit	AVR® RISC 8- bit	AVR® RISC 8- bit	AVR® RISC 8- bit
Frequ ncy	133MH z	133MH z	160 MHz	240 MHz	48MHz	48MHz	48MHz	20MHz	20MHz	20MHz	24MHz
SRAM	264KB	264KB	400 KB	8MB	32KB	4KB	32KB	256B	2KB	3KB	16KB
onboar d memor y	2MB	2MB	4MB	8MB	256KB	16KB	256KB	4KB	16KB	32KB	128KB
I/O- Pins	11	26	11	11	11	12	26	6	12	18	25/26 (1x only In)
ADC	4	3	3	9	11	5	6	6	9	15	13
DAC	0	0	0	0	1	1	1	1	0	0	1
Packag e						SOIC- 14	TQFP- 32	SOIC-8	SOIC- 14	SOIC- 20	TQFP- 32

	Xiao RP2040	Pico RP2040	ESP32 C3	ESP32 S3	SAMD 21	ATSA MD11 C14A- SSUT	ATSA MD21 E18A- AUT	ATTIN Y412- SSFR	ATTIN Y1624 -SSFR	ATTIN Y3226 -SU	AVR1 28DB3 2-I/PT
Price	4,68 \$	5 \$	4,99 \$	7,49 \$	5,4 \$	1,58 \$	4,03 \$	0,59 \$	1,01 \$	1,29 \$	2,06 \$
FPU availab le?											

# RP2040 (Raspberry Pi Pico)

**Toolchain:**

- **Primary:** Pico SDK (C/C++), CMake
- **Alternatives:** Arduino IDE (via Arduino-Pico Core), MicroPython/CircuitPython

**Workflow:**

1. Write code in C/C++ or Python.
2. Build with CMake (Pico SDK) or Arduino IDE.
3. Flash via USB (UF2 bootloader) or SWD debugger (e.g., Picoprobe).

**Efficiency Tips:**

- Use Visual Studio Code with the Pico SDK extension for CMake integration.
- Leverage Picoprobe (a second Pico) for debugging.

# ESP32 (Espressif)

**Toolchain:**

- **Primary:** ESP-IDF (C/C++), PlatformIO
- **Alternatives:** Arduino IDE (via ESP32 Core)

**Workflow:**

1. Develop in C/C++ (ESP-IDF) or Arduino framework.
2. Build with ESP-IDF CLI or PlatformIO.
3. Flash via USB (esptool.py) or OTA updates.

**Efficiency Tips:**

- PlatformIO streamlines ESP-IDF/Arduino workflows.
- Use ESP-Prog or JTAG for advanced debugging.

## SAMD21/SAMD11 (Atmel/Microchip)

### Toolchain:

- **Primary:** Atmel/Microchip Studio (C/C++)
- **Alternatives:** Arduino IDE (via SAMD Core)

### Workflow:

1. Code in C/C++ (Microchip Studio) or Arduino.
2. Build and flash via USB (UF2 bootloader) or EDBG/SWD.

### Efficiency Tips:

- Use Arduino IDE for simplicity; enable verbose upload for debugging.
- For low-level control, use CMSIS libraries in Microchip Studio.

## Attiny/AVR128 (AVR Family)

### Toolchain:

- **Primary:** AVR-GCC + AVRdude
- **Alternatives:** Arduino IDE (via ATTiny Core)

### Workflow:

1. Write code in C/C++ or Arduino.
2. Compile with AVR-GCC or Arduino IDE.
3. Flash via ISP programmer (e.g., USBasp, Arduino-as-ISP).

### Efficiency Tips:

- Use PlatformIO for project management.
- For tinyAVR (e.g., ATtiny85), optimize code size with `-Os` compiler flag.

## General Workflow Optimization Tips

### 1. Unified Environments:

- **PlatformIO** (VS Code) supports all listed MCUs, reducing toolchain setup time.

- **Arduino IDE** (with board managers) simplifies entry-level development.
2. **Debugging Tools:**
    - **SWD/JTAG:** Use for RP2040, ESP32, SAMD21 (e.g., Segger J-Link, CMSIS-DAP).
    - **Serial Monitor:** Essential for ESP32/RP2040 debugging.
  3. **Version Control:**
    - Use `git` for code management; track dependencies (e.g., submodules for Pico SDK).
  4. **Automation:**
    - Write Makefiles or use PlatformIO scripts for CI/CD pipelines.

QFP mit Beinchen seitlich

TQFP ohne Beinchen

Soic-8 8 Beinchen

Punkt oder Kerbe kennzeichnen Pin 1

# Putting code onto it

## In-System Programming (ISP)

- the act of programming a microcontroller while it is already mounted on the board
- Contrary to Pre-Programming. You program the contro

CMSIS-DAP Devices are all devices that can write programs into a microcontroller's memory using JTAG or SWD. Ier before soldering it somewhere.

## Serial Peripheral Interface (SPI)

- A standard synchronous serial communication interface used for short-distance communication between a main device and one or more peripheral devices.
- synchronous (needs clock signal), serial communication
- multiple secondary devices are possible
- full-duplex
- pins:
  - MOSI (main out secondary in)
  - MISO (main in secondary out)
  - SCK (clock)
  - SS (secondary select): select which secondary device to communicate with

## Serial Peripheral Data Interface (SPDI)

- do not find any source for it
- Good Article GERMAN

## Joint Test Action Group (JTAG)

- electronics manufacturers committee
- they developed a protocol with the same name
- mostly used for programming ARM cores
- daisy-chaining possible
- pins
  - TMS: mode select
  - TCLK: clock
  - TDO: data out
  - TDI: data in
  - nRESET: reset (optional)

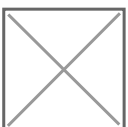


### Serial Wire Debug (SWD)

- two-pin variant of the JTAG protocol -> replaced JTAG
- daisy-chaining not possible
- most common on newer ARM chips
- pins:
  - swdio: in/out
  - swclk: clock

### Unified Program and Debug Interface (UPDI)

- proprietary
- single-wire,
- bi-directional, half-duplex
- asynchronous
- can use off-the shelf UART adapters
- used to program AVR microcontrollers released since 2016, ATTINY412, ATTINY164.
- more detailed blog article
- debugging is hidden behind a proprietary interface, but even though you can snoop on the protocol with just a serial adapter and even though the large scale structure of the protocol is known too
- even more detailed article
- i did not find out what part of the protocol actually is proprietary and what not.



# Difference between microcontroller and -processor

...

## Programmierer:

### ISP-Programmierer

AVRISP MKII



FabISP



### JTAG-Programmierer

ATMEL-ICE



<http://pub.fabcloud.io/programmers/summary/>

---

Revision #43

Created 12 February 2025 22:02:08 by Jakob

Updated 27 February 2025 17:10:18 by Matthias