Microcontroller Overview

Jake Garrison
475 2024

Why Use an Microcontroller (MCU)?

github notes

Cost-effectiveness: Integration less components more board space.

Power efficiency: Low power crucial for portable devices.

Small size: Compact for space-constrained applications.

Real-time control: Precise timing and fast response for critical tasks.

Dedicated functionality: Optimized for specific tasks, efficient.

Wide range of applications: From IoT devices and industrial automation to robotics, consumer electronics, medical devices, and aerospace systems.

Factors to Consider When Choosing an MCU

github notes

1. **Application Needs:**
   a. Define project goals (what it does).
   b. List all inputs (sensors, buttons) and outputs (actuators, displays).
   c. Determine processing needs (simple, moderate, complex).
2. **Features:**
   a. Essential peripherals (ADC, DAC, timers, etc.).
   b. I/O requirements (analog, digital).
   c. Communication interfaces (UART, SPI, I2C, USB, CAN, etc.).

I2C Address List

3. **Memory Architecture:**
   a. Modified Harvard (most common): Separate caches, single address space.
   b. Harvard: Separate memories, potentially faster.
4. **Bit Size (Processing Power):**
   a. 8-bit: Simple, low power. (AVR, PIC16)
   b. 16-bit: Balanced. (MSP430)
   c. 32-bit: High performance. (ARM Cortex-M, PIC32)
5. **Networking/Communication:**
   a. Choose protocols (Wi-Fi, Bluetooth, etc.) based on needs.
   b. Address security (HTTPS, TLS).

Factors to Consider When Choosing an MCU

6. **Power:**
    a. Ensure voltage compatibility.
    b. Choose regulator (linear, switching).
    c. Utilize low-power modes (sleep, interrupts).
7. **I/O Pins:**
    a. Ensure enough GPIO.
    b. Consider specialized I/O (PWM, ADC).
8. **Physical Size:**
    a. Through-hole (DIP) for Prototyping.
    b. Surface Mount (SMD): Compact designs.
    c. Modules: Pre-assembled, easier but potentially less flexible.
9. **Memory:**
    a. Flash: Code storage (Non-volatile).
    b. RAM: Active data. Cleared when power off (SRAM, Volatile).
    c. EEPROM/FRAM: Stores small amounts of data e.g., config, creds (Non-volatile).
10. **Development Ecosystem:**
    a. IDE/Toolchain, Libraries, Community, Debugging tools, Licenses.
    b. Language: C/C++, Assembly, MicroPython.

Considerations for Low Power Projects

Power Management: Crucial for battery life.

- Utilize sleep modes, low-power peripherals, and efficient code.
- Consider Power Management Integrated Circuits (PMICs) for battery management.

Wireless: (BLE, Wi-Fi, Thread, Zigbee, NFC etc.)

- Select the appropriate wireless protocol based on range, data rate, and power needs.

Displays: Prioritize low-power displays (e.g., e-ink)

- consider size and interface (SPI, I2C).

Processing (AI/DSP):

- Select MCUs with sufficient processing power, FPU, and DSP.

Big List of Microcontrollers

For a larger list of options grouped by vendor, see github notes **Popular Microcontrollers**

Some Recommendations (by application)

Ultra-Low Power (Minimal Resources, simple sensors, long battery life):

- **Texas Instruments MSP430:** Exceptional power efficiency for basic tasks.
- **Microchip ATTiny85:** Tiny and ultra-low power, for space-constrained.

Cost-Effective 8-bit (Simple projects, beginners):

- **Microchip AVR (ATmega328P, etc.):** Widely used, Arduino IDE compatibility.
- **PIC16/PIC18:** Microchip's cost-effective options.
- **STM8:** Cost-effective 8-bit option from STMicroelectronics.

Real-Time Applications & DSP (Signal processing, time-sensitive):

- **ATmega32U4:** Built-in USB for HID and other low latency USB applications.
- **Teensy Family (e.g., Teensy 4.0, 4.1, LC):** Audio processing, robotics, uses Arduino IDE
- **STM32F4 Series (e.g., F407VG, F446RE):** Features (FPU, DSP instructions), affordable.
- **STM32H7 Series:** Significantly more processing power for demanding real-time DSP, more expensive.

Some Recommendations (by application)

Balanced Power/Performance with Wireless (Wearables, data logging, IoT):

- **Nordic nRF52840:** More powerful than the nRF52832, with additional wireless protocols.
- **Raspberry Pi Pico W:** Affordable dual-core with Wi-Fi, easy to use with MicroPython.

High Performance with Wireless (Complex wearables, edge compute, robotics):

- **Espressif ESP32/ESP32-S3:** Dual-core with Wi-Fi/Bluetooth. S3 adds a camera interface.
- **Arduino Nano RP2040 Connect:** Integrated Wi-Fi, Bluetooth, and onboard sensors.

Image/Audio AI (on-device machine learning, power hungry):

- **ESP32-S3:** Good balance of performance, memory, and cost.
- **STM32F7/H7:** Powerful for DSP and AI due to FPU and dedicated DSP instructions.
- **Microchip SAM D51:** Cortex-M4F with FPU suitable for some DSP/AI tasks
- **SparkFun Edge:** Cortex-M4F with built-in mic, accel, camera and tflite support
- **Dedicated AI Accelerators:** For complex AI, consider dedicated hardware (e.g., Google Coral) used in conjunction with a suitable MCU.

MCU On-Device Machine Learning Frameworks

github notes

AI/ML Frameworks for Microcontrollers:

- **TensorFlow Lite:**
  - now called **LiteRT**
  - Supported MCUs
  - Examples
- **Other TinyML Frameworks:**
  - **uTensor:** Lightweight, memory-efficient, supports various platforms.
  - **CMSIS-NN:** ARM's optimized neural network kernels for Cortex-M.
    i. Can be used with TensorFlow Lite.
  - **Glow:** Compiler for neural networks, includes MCU backend.