AVR Overview

The AVR is a modified Harvard architecture machine, where program and data are stored in separate physical memory systems that appear in different address spaces, but having the ability to read data items from program memory using special instructions.

Basic families

AVRs are generally classified into following:

tinyAVR — the ATtiny series

1 0.5–16 kB program memory
2 6–32-pin package
3 Limited peripheral set

megaAVR — the ATmega series

1 4–256 kB program memory
2 28–100-pin package
3 Extended instruction set (multiply instructions and instructions for handling larger program memories)
4 Extensive peripheral set

XMEGA — the ATxmega series

1 16–384 kB program memory
2 44–64–100-pin package (A4, A3, A1)
3 32-pin package : XMEGA-E (XMEGA8E5)
4 Extended performance features, such as DMA, "Event System", and cryptography support.
5 Extensive peripheral set with ADCs

Application-specific AVR

megaAVRs with special features not found on the other members of the AVR family, such as LCD controller, USB controller, advanced PWM, CAN, etc.

FPSLIC (AVR with FPGA)

1 FPGA 5K to 40K gates
2 SRAM for the AVR program code, unlike all other AVRs
3 AVR core can run at up to 50 MHz

32-bit AVRs

In 2006 Atmel released microcontrollers based on the 32-bit AVR32 architecture. They include SIMD and DSP instructions, along with other audio- and video-processing features. This 32-bit family of devices is intended to compete with the ARM-based processors. The instruction set is similar to other RISC cores, but it is not compatible with the original AVR or any of the various ARM cores.

AVR ATmega328 Microcontroller High-Level Block Diagram

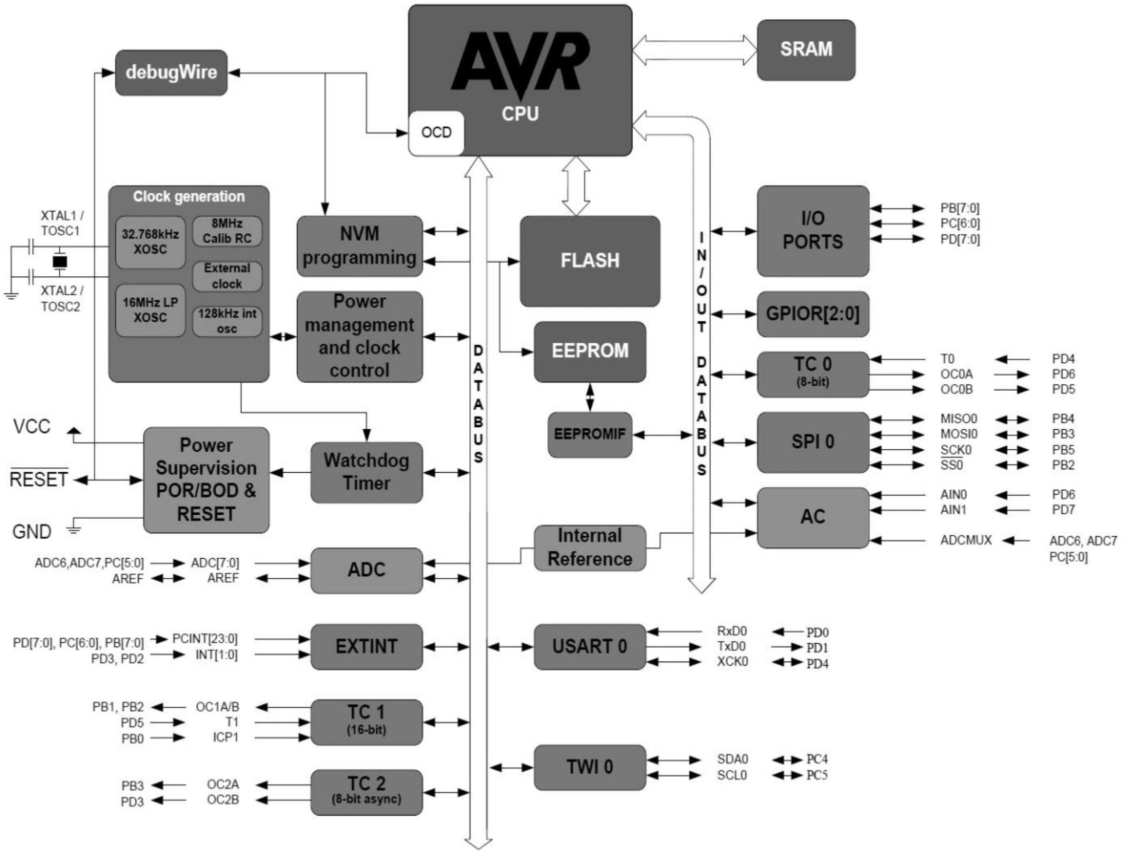

Fig 1.2 ATmega 328 block diagram

ATmega328 Pin-out

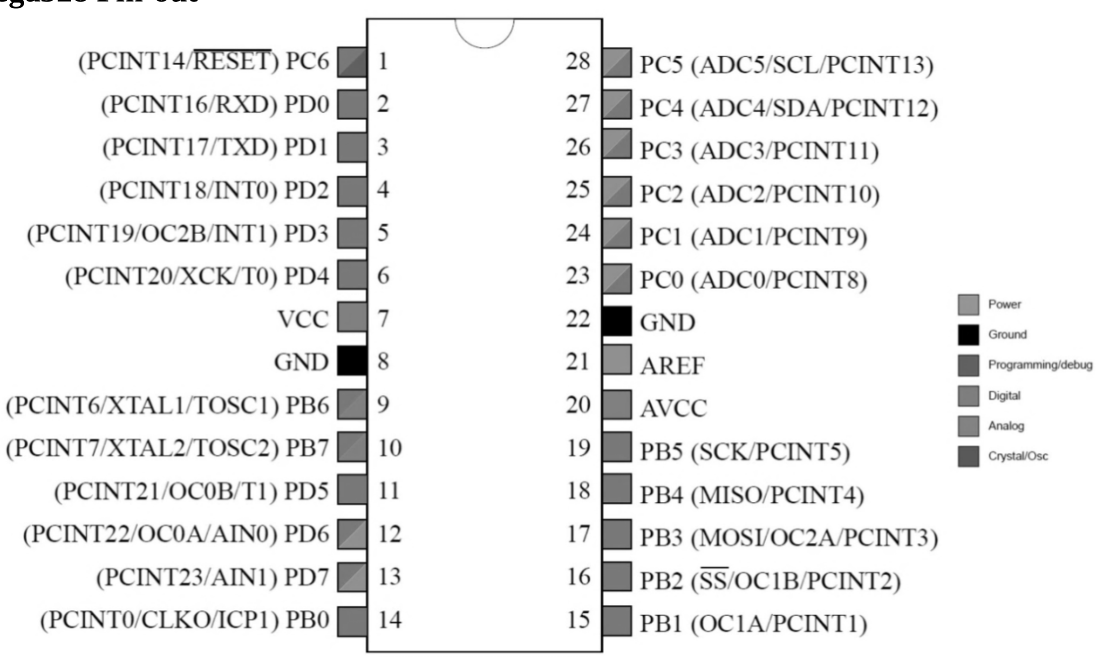

Fig 1.3 ATmega 328 Pin Diagram

Device architecture

Flash, EEPROM, and SRAM are all integrated onto a single chip, removing the need for external memory in most applications. Some devices have a parallel external bus option to allow adding additional data memory or memory-mapped devices. Almost all devices (except the smallest TinyAVR chips) have serial interfaces, which can be used to connect larger serial EEPROMs or flash chips.

Program memory

Program instructions are stored in non-volatile flash memory. Although the MCUs are 8-bit, each instruction takes one or two 16-bit words.

The size of the program memory is usually indicated in the naming of the device itself (e.g., the ATmega64x line has 64 kB of flash, while the ATmega32x line has 32 kB).

There is no provision for off-chip program memory; all code executed by the AVR core must reside in the on-chip flash. However, this limitation does not apply to the AT94 FPSLIC AVR/FPGA chips.

Internal data memory

The data address space consists of the register file, I/O registers, and SRAM.

Internal registers

The AVRs have 32 single-byte registers and are classified as 8-bit RISC devices.

In the tinyAVR and megaAVR variants of the AVR architecture, the working registers are mapped in as the first 32 memory addresses (0000_{16}–$001F_{16}$), followed by 64 I/O registers (0020_{16}–$005F_{16}$). In devices with many peripherals, these registers are followed by 160 “extended I/O” registers, only accessible as memory-mapped I/O (0060_{16}–$00FF_{16}$).

Actual SRAM starts after these register sections, at address 0060_{16} or, in devices with “extended I/O”, at 0100_{16}.

Even though there are separate addressing schemes and optimized opcodes for accessing the register file and the first 64 I/O registers, all can still be addressed and manipulated as if they were in SRAM.

The very smallest of the tinyAVR variants use a reduced architecture with only 16 registers (r0 through r15 are omitted) which are not addressable as memory locations. I/O memory begins at address 0000_{16}, followed by SRAM. In addition, these devices have slight deviations from the standard AVR instruction set. Most notably, the direct load/store instructions (LDS/STS) have been reduced from 2 words (32 bits) to 1 word (16 bits), limiting the total direct addressable memory (the sum of both I/O and SRAM) to 128 bytes. Conversely, the indirect load instruction's (LD) 16-bit address space is expanded to also include non-volatile memory such as Flash and configuration bits; therefore, the LPM instruction is unnecessary and omitted.

In the XMEGA variant, the working register file is not mapped into the data address space; as such, it is not possible to treat any of the XMEGA's working registers as though they were SRAM. Instead, the I/O registers are mapped into the data address space starting at the very beginning of the address space. Additionally, the amount of data address space dedicated to I/O registers has grown substantially to 4096 bytes (0000_{16}–$0FFF_{16}$). As with previous generations, however, the fast I/O manipulation instructions can only reach the first 64 I/O register locations (the first 32 locations for bitwise instructions). Following the I/O registers, the XMEGA series sets aside a 4096 byte range of the data address space, which can be used optionally for mapping the internal EEPROM to the data address space (1000_{16}–$1FFF_{16}$). The actual SRAM is located after these ranges, starting at 2000_{16}.

GPIO ports

Each GPIO port on a tiny or mega AVR drives up to eight pins and is controlled by three 8-bit registers: DDR*x*, PORT*x* and PIN*x*, where *x* is the port identifier.

DDR*x*: Data Direction Register configures the pins as either inputs or outputs.

PORT*x*: Output port register. Sets the output value on pins configured as outputs. Enables or disables the pull-up resistor on pins configured as inputs.

PIN*x*: Input register, used to read an input signal. On some devices (but not all, check the datasheet), this register can be used for pin toggling: writing a logic one to a PIN*x*bit toggles the corresponding bit in PORT*x*, irrespective of the setting of the DDR*x* bit.[8]

xmegaAVR have additional registers for push/pull, totem-pole and pull-up configurations.

EEPROM

Almost all AVR microcontrollers have internal EEPROM for semi-permanent data storage. Like flash memory, EEPROM can maintain its contents when electrical power is removed.

In most variants of the AVR architecture, this internal EEPROM memory is not mapped into the MCU's addressable memory space. It can only be accessed the same way an external peripheral device is, using special pointer registers and read/write instructions, which makes EEPROM access much slower than other internal RAM.

However, some devices in the SecureAVR (AT90SC) family use a special EEPROM mapping to the data or program memory, depending on the configuration. The XMEGA family also allows the EEPROM to be mapped into the data address space.

Since the number of writes to EEPROM is not unlimited — Atmel specifies 100,000 write cycles in their datasheets — a well-designed EEPROM write routine should compare the contents of an

EEPROM address with desired contents and only perform an actual write if the contents need to be changed.

Note that erase and write can be performed separately in many cases, byte-by-byte, which may also help prolong life when bits only need to be set to all 1s (erase) or selectively cleared to 0s (write).

Program execution

Atmel's AVRs have a two-stage, single-level pipeline design. This means the next machine instruction is fetched as the current one is executing. Most instructions take just one or two clock cycles, making AVRs relatively fast among eight-bit microcontrollers.

The AVR processors were designed with the efficient execution of compiled C code in mind and have several built-in pointers for the task.

MCU speed

The AVR line can normally support clock speeds from 0 to 20 MHz, with some devices reaching 32 MHz Lower-powered operation usually requires a reduced clock speed. All recent (Tiny, Mega, and Xmega, but not 90S) AVRs feature an on-chip oscillator, removing the need for external clocks or resonator circuitry. Some AVRs also have a system clock prescaler that can divide down the system clock by up to 1024. This prescaler can be reconfigured by software during run-time, allowing the clock speed to be optimized.

Since all operations (excluding multiplication and 16-bit add/subtract) on registers R0–R31 are single-cycle, the AVR can achieve up to 1 MIPS per MHz, i.e. an 8 MHz processor can achieve up to 8 MIPS. Loads and stores to/from memory take two cycles, branching takes two cycles. Branches in the latest "3-byte PC" parts such as ATmega2560 are one cycle slower than on previous devices.

Development

AVRs have a large following due to the free and inexpensive development tools available, including reasonably priced development boards and free development software. The AVRs are sold under various names that share the same basic core, but with different peripheral and memory combinations. Compatibility between chips in each family is fairly good, although I/O controller features may vary.

See external links for sites relating to AVR development.

Features

Current AVRs offer a wide range of features:

1. Multifunction, bi-directional general-purpose I/O ports with configurable, built-in pull-up resistors
2. Multiple internal oscillators, including RC oscillator without external parts
3. Internal, self-programmable instruction flash memory up to 256 kB (384 kB on XMega)
4. In-system programmable using serial/parallel low-voltage proprietary interfaces or JTAG
5. On-chip debugging (OCD) support through JTAG or debugWIRE on most devices
6. Internal data EEPROM up to 4 kB
7. Internal SRAM up to 16 kB (32 kB on XMega)
8. The external data space is overlaid with the internal data space, such that the full 64 kB address space does not appear on the external bus and accesses to e.g. address 0100_{16} will access internal RAM, not the external bus.
9. 8-bit and 16-bit timers
10. PWM output (some devices have an enhanced PWM peripheral which includes a dead-time generator)
11. Input capture that record a time stamp triggered by a signal edge
12. Analog comparator
13. 10 or 12-bit A/D converters, with multiplex of up to 16 channels
14. 12-bit D/A converters
15. A variety of serial interfaces, including
    I²C compatible Two-Wire Interface (TWI)
    Synchronous/asynchronous serial peripherals (UART/USART) (used with RS-232, RS-485, and more)
    Serial Peripheral Interface Bus (SPI)
    Universal Serial Interface (USI): a multi-purpose hardware communication module that can be used to implement an SPI,[10] I²C[11][12] or UART interface.

16. Watchdog timer (WDT)
17. Multiple power-saving sleep modes
18. Lighting and motor control (PWM-specific) controller models
19. CAN controller support
20. USB controller support
21. Ethernet controller support
22. LCD controller support
23. DMA controllers and "event system" peripheral communication.

PIC16F877A –Simplified Features	
CPU	8-bit PIC
Number of Pins	40
Operating Voltage (V)	2 to 5.5 V
Number of I/O pins	33
ADC Module	8ch, 10-bit
Timer Module	8-bit(2), 16-bit(1)
Comparators	2
DAC Module	Nil
Communication Peripherals	UART(1), SPI(1), I2C(1), MSSP(SPI/I2C)
External Oscillator	Up to 20Mhz
Internal Oscillator	Nil
Program Memory Type	Flash
Program Memory (KB)	14KB
CPU Speed (MIPS)	5 MIPS
RAM Bytes	368
Data EEPROM	256 bytes

PIC16F877A

This powerful (200 nanosecond instruction execution) yet easy-to-program (only 35 single word instructions) CMOS FLASH-based 8-bit microcontroller packs Microchip's powerful PIC® architecture into an 40 package and is upwards compatible with the PIC16C5X, PIC12CXXX and PIC16C7X devices. The PIC16F877A features 256 bytes of EEPROM data memory, self programming, an ICD, 2 Comparators, 8 channels of 10-bit Analog-to-Digital (A/D) converter, 2 capture/compare/PWM functions, the synchronous serial port can be configured as either 3-wire Serial Peripheral Interface (SPI™) or the 2-wire Inter-Integrated Circuit (I^2C™) bus and a Universal Asynchronous Receiver Transmitter (USART).

How to Select your PIC Microcontroller

Microchip provides a waste verity of Microcontrollers from PIC family. Each MCU has its own advantage and disadvantage. There are many parameters that one has to consider before selecting a MCU for his project. The below points are just suggestions which might help one to select a MCU.

- If you are a beginner who is learning PIC then, selecting a MCU that has good online community support and wide applications will be a good choice. PIC16F877A and PIC18F4520 are two such MCUs
- Consider the operating voltage of your system. If they are 5V then select a 5V MCU some sensors or devices work and communicate on 3.3V in such case a 3.3V MCU can be selected
- If size and price is a limitation then you can choose small 8-pin MCUs like PIC12F508. These are also comparatively cheaper.
- Based on the sensors and actuators used in your project, verify which modules you might need in for MCU. For example is you are reading many Analog voltages then make sure PIC has

enough ADC channels and supportive resolution. The details of all modules are given in the table above.

- If you project involves communication protocols like UART, SPI ,I2C, CAN etc make sure you PIC can support them. Some MCU can support more than one module of the same protocol

PIC16F877A –Detailed Features	
CPU	8-bit PIC
Architecture	8
Program Memory Size (Kbytes)	14
RAM (bytes)	368
EEPROM/HEF	256/HEF
Pin Count	40
Max. CPU Speed (MHz)	20
Peripheral Pin select (PPS)	No
Internal Oscillator	No
No. Of comparators	2
No. Of Operational Amplifier	0
No. Of ADC channels	14
Max ADC Resolution (bits)	10
ADC with Computation	No
Number of DAC Converter	0
Max DAC resolution	0
Internal Voltage Reference	Yes
Zero Cross Detect	No
No. Of 8-bit timers	2
No. Of 16-bit Timers	1
Signal Measurement Timer	0
Hardware Limit Timer	0
No. Of PWM outputs	0
Max PWM resolution	10
Angular Timer	No
Math Accelerator	No
No. Of UART module	1
No. Of SPI Module	1
No. Of I2C module	1
No. Of USB Module	0
Windowed Watchdog Timer (WWDT)	No
CRC/Scan	No
Numerically Controlled Oscillator	0
Cap. Touch Channels	11
Segment LCD	0
Minimum Operating Temperature (*C)	-40
Maximum Operating Temperature	125

(*C)	
Minimum Operating Voltage (V)	2
Maximum Operating Voltage (V)	5.5
High Voltage Capable	No

Applications

- Replacement for Arduino Module
- Ideal for more advanced level A/D applications in automotive, industrial, appliances and consumer applications

Main Difference between AVR, ARM, 8051 and PIC Microcontrollers

	8051	PIC	AVR	ARM
Bus width	8-bit for standard core	8/16/32-bit	8/32-bit	32-bit mostly also available in 64-bit
Communication Protocols	UART, USART,SPI,I2C	PIC, UART, USART, LIN, CAN, Ethernet, SPI, I2S	UART, USART, SPI, I2C, (special purpose AVR support CAN, USB, Ethernet)	UART, USART, LIN, I2C, SPI, CAN, USB, Ethernet, I2S, DSP, SAI (serial audio interface), IrDA
Speed	12 Clock/instruction cycle	4 Clock/instruction cycle	1 clock/ instruction cycle	1 clock/ instruction cycle
Memory	ROM, SRAM, FLASH	SRAM, FLASH	Flash, SRAM, EEPROM	Flash, SDRAM, EEPROM
ISA	CLSC	Some feature of RISC	RISC	RISC
Memory Architecture	Von Neumann architecture	Harvard architecture	Modified	Modified Harvard architecture
Power Consumption	Average	Low	Low	Low
Families	8051 variants	PIC16,PIC17, PIC18, PIC24, PIC32	Tiny, Atmega, Xmega, special purpose AVR	ARMv4,5,6,7 and series
Community	Vast	Very Good	Very Good	Vast
Manufacturer	NXP, Atmel, Silicon Labs, Dallas, Cyprus, Infineon, etc.	Microchip Average	Atmel	Apple, Nvidia, Qualcomm, Samsung Electronics, and TI etc.
Cost (as compared to features provide)	Very Low	Average	Average	Low
Other Feature	Known for its Standard	Cheap	Cheap, effective	High speed operation Vast
Popular Microcontrollers	AT89C51, P89v51, etc.	PIC18fXX8, PIC16f88X, PIC32MXX	Atmega8, 16, 32, Arduino Community	LPC2148, ARM Cortex-M0 to ARM Cortex-M7, etc.