Embedded_session_5

Embedded System Session 5 - ADC & DAC

Created: 2025-07-17
Author: Fares Hesham Mahmoud
Tags: AVR, Embedded system, ADC, DAC
Status: #Formatted

اللهمَّ علِّمنا ما ينفعنا، وانفعنا بما علمتنا، وزدنا علمًا. وافتح علينا فتحًا عظيمًا.

ADC (Analog-to-Digital Converter)

Key Terminology

Power Supply & Reference

- AVCC: Analog external power supply into the microcontroller
- AREF: Analog external reference supply (AVCC gets compared to it every unit time)

ADC Parameters

Parameter	Formula	Example (10-bit, 5V ref)
Resolution	Number of bits	10 bits
Sample Size	2^resolution	2^10 = 1024
Step Size	Reference Voltage / Sample Size	5V / 1024 = 4.88 mV
Analog Value	Digital Value × Step Size	Digital × 4.88 mV

Analog to Digital Conversion Process

1. Sampling

After each period of time T, we read the value of the analog signal (take a sample every period).

Nyquist Theorem: Sample frequency should be double the frequency of the analog signal.

2. Hold

The sampled value is held until the next sample arrives (creates a bar graph shape).

3. Quantization

Divide the range of values depending on the sample size in memory.

Quantization Examples

- 2-bit: `00, 01, 10, 11` (4 levels)
- 4-bit: 16 levels
- 8-bit: 256 levels
- 10-bit: 1024 levels

Quantization Error

When you choose a small memory size to store samples, sometimes unequal values get treated as if they were the same, causing accuracy errors.

DAC (Digital-to-Analog Converter)

R-2R Ladder Network

For a 4-bit system with values [R, 2R, 4R, 8R]:

Formula:

$$_t = \frac{2_c\ 8}{2} = \frac{_{ref} \times _i}{}$$

Weighted Resistor Method

Alternative DAC implementation using weighted resistors.

ADC Sampling Modes

Interrupt Flag Behavior

- The flag is raised automatically when conversion is complete

- Global interrupt enable and peripheral interrupt enable are only needed for ISR functionality
- Without enabling interrupts, you can still use polling method

Free Running Mode

- Needs a first trigger to start (set register value based on datasheet)
- After initial trigger, continues automatically
- Continuous conversion

Single Conversion Mode

- Performs conversion only once when triggered
- Waits for another trigger before next conversion
- More power-efficient for occasional measurements

ADC Circuit Types

RAMP ADC (Counter-Type)

- Conversion time varies according to the value being converted
- Lower values (like 10) take less time than higher values (like 1023)
- Simple but slower for high values

SAR ADC (Successive Approximation Register)

- Constant conversion time
- Converts 10 in the same time as 1023
- More efficient and commonly used

ADC Registers

ADMUX (ADC Multiplexer Selection)

Controls reference voltage and input channel selection.

Bits	Function
REFS1:REFS0	Reference Selection
ADLAR	Left Adjust Result
MUX4:MUX0	Input Channel Selection

Reference Voltage Selection:

REFS1	REFS0	Reference
0	0	AREF, Internal Vref turned off
0	1	AVCC with external capacitor
1	0	Reserved
1	1	Internal 2.56V with external capacitor

ADCSRA (ADC Control and Status Register A)

Bit	Function	Description
ADEN	ADC Enable	Set to enable ADC
ADSC	ADC Start Conversion	Set to start conversion
ADATE	ADC Auto Trigger Enable	Enable auto-triggering
ADIF	ADC Interrupt Flag	Set when conversion complete
ADIE	ADC Interrupt Enable	Enable ADC interrupt
ADPS2:ADPS0	ADC Prescaler Select	Clock division factor

Prescaler Selection:

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Practical Considerations

ADC Best Practices >

- Choose appropriate prescaler for ADC clock (50-200 kHz recommended)
- Allow settling time after changing input channels
- Use appropriate reference voltage for your measurement range

- Consider noise filtering for critical measurements

Basic ADC Code Structure

```
// ADC Initialization
void ADC_Init(void) {
    ADMUX = (1<<REFS0);             // AVCC reference
    ADCSRA = (1<<ADEN) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0); //
Enable, prescaler 128
}

// Single conversion
uint16_t ADC_Read(uint8_t channel) {
    ADMUX = (ADMUX & 0xF8) | (channel & 0x07);  // Select channel
    ADCSRA |= (1<<ADSC);            // Start conversion
    while(ADCSRA & (1<<ADSC));      // Wait for completion
    return ADC;                     // Return result
}
```

References

- ATmega32 Datasheet
- ADC Theory and Applications
- Digital Signal Processing Fundamentals