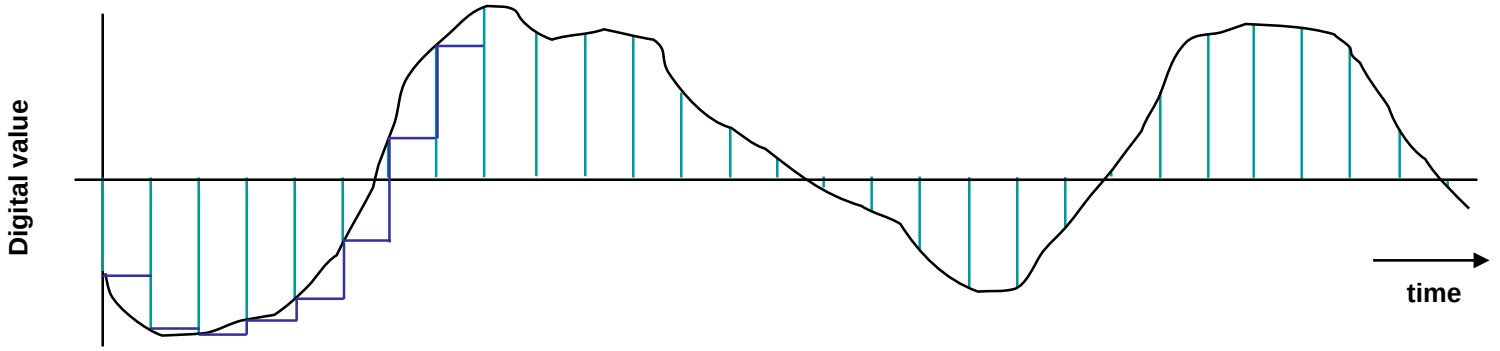
ATmega128

Analog Interfacing

Analog and Digital

- CD digital audio encoding: 2-channel signed 16-bit Linear PCM sampled at 44,100 Hz.

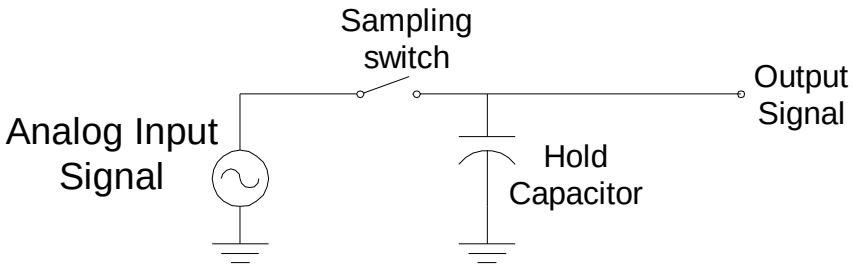
Waveform Sampling and Quantization



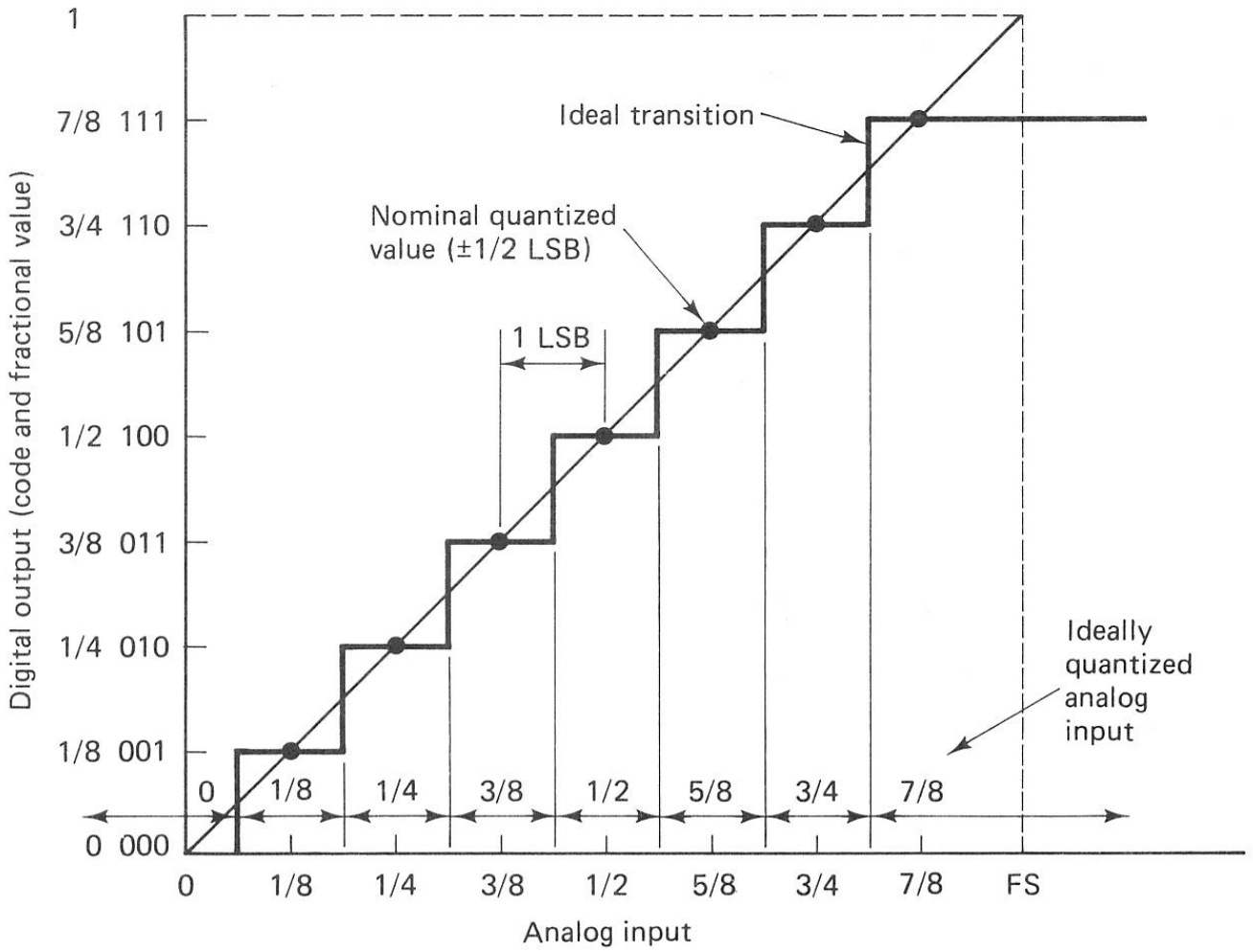
- A waveform is **sampled** at a constant rate – every Δ_t
  - Each such sample represents the instantaneous amplitude at the instant of sampling
  - *"At 37 ms, the input is 1.913419145134514512343 11… V"*
  - Sampling converts a **continuous time** signal to a **discrete time** signal
- The sample can now be **quantized** (converted) into a digital value
  - Quantization represents a **continuous** (analog) value with the closest **discrete** (digital) value
  - *"The sampled input voltage of 1.913419145134514512343 11… V is best represented by the code 0x018, since it is in the range of 1.901 to 1.9980 V which corresponds to code 0x018."*

Sample and Hold Devices

- Some A/D converters require the input analog signal to be held constant during conversion, (eg. successive approximation devices)
- In other cases, peak capture or sampling at a specific point in time necessitates a sampling device.
- This function is accomplished by a sample and hold device as shown to the right:
- These devices are incorporated into some A/D converters



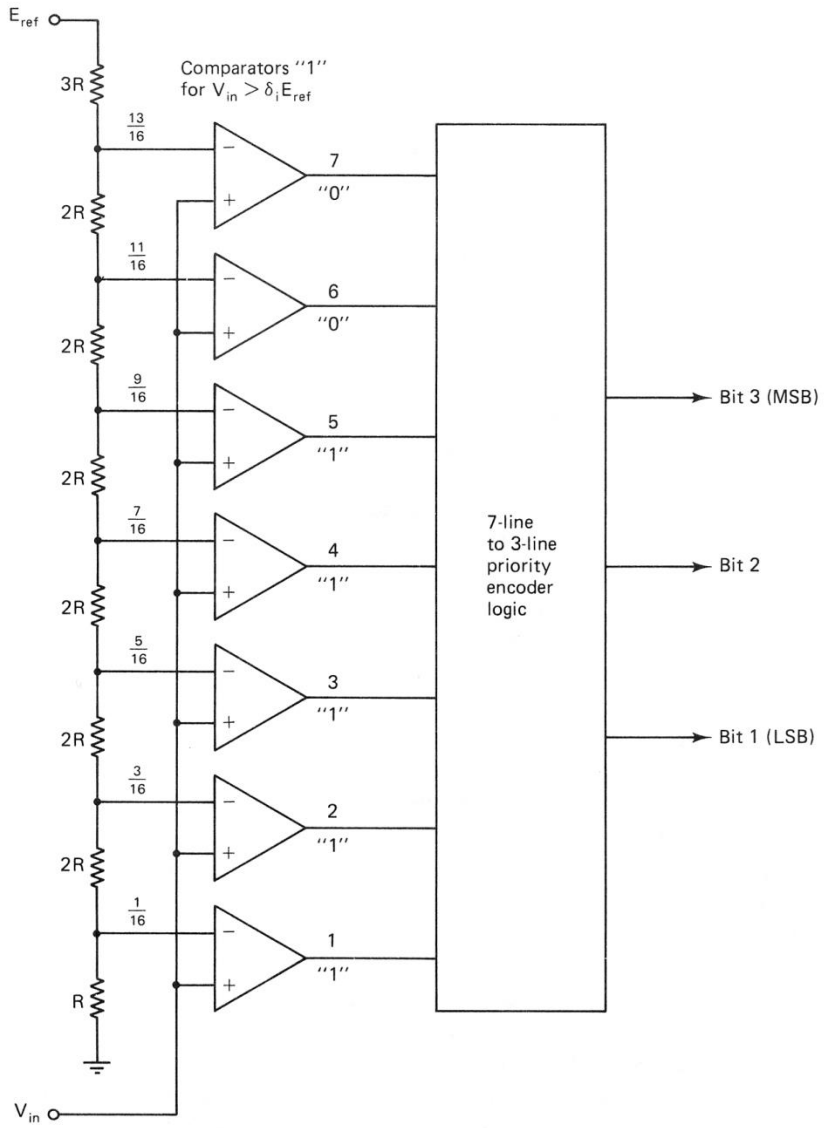
A/D Converter



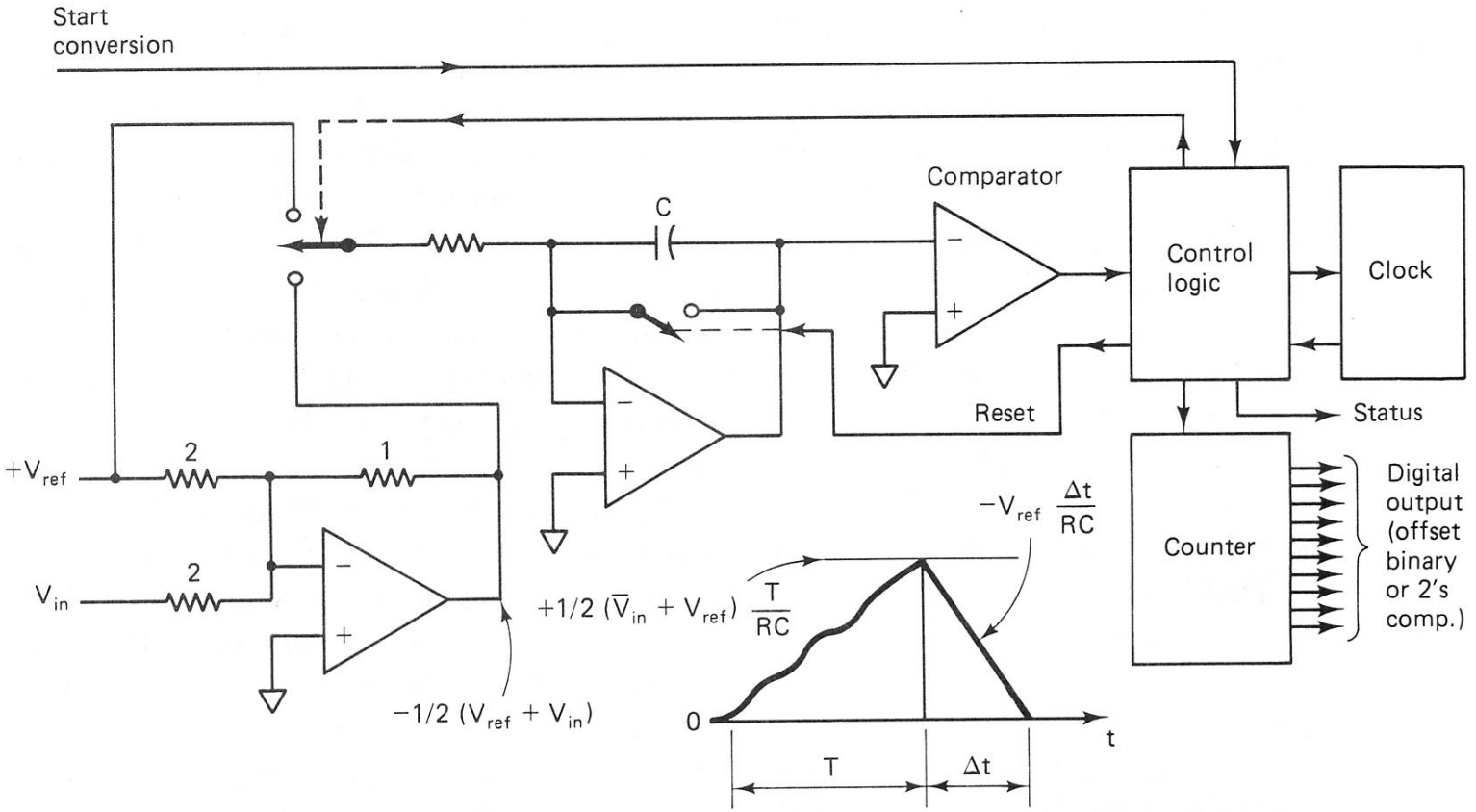
A/D Converter

- Multi-comparator “flash”
- Integrating
- Tracking (counter-comparator)
- Successive approximation

Multi-comparator “flash”



Integrating

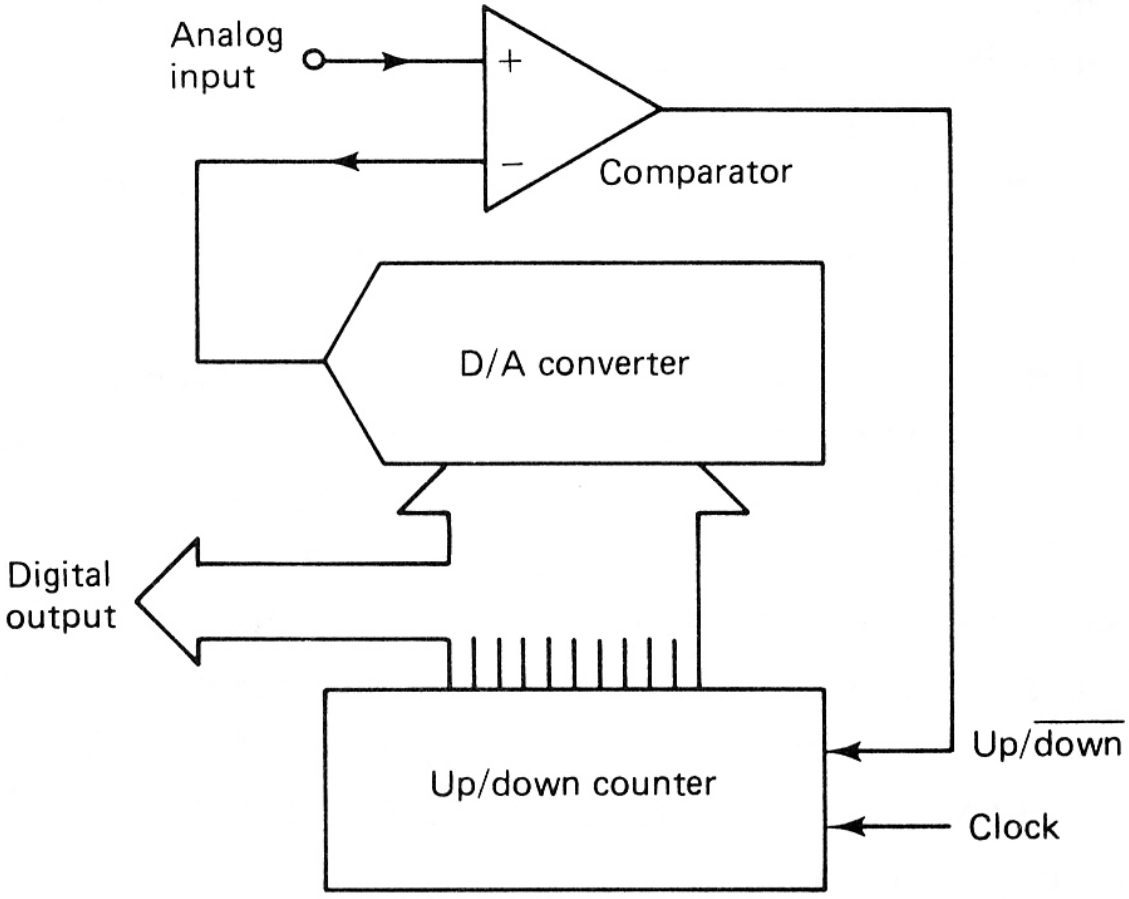


Comparator is tripped when

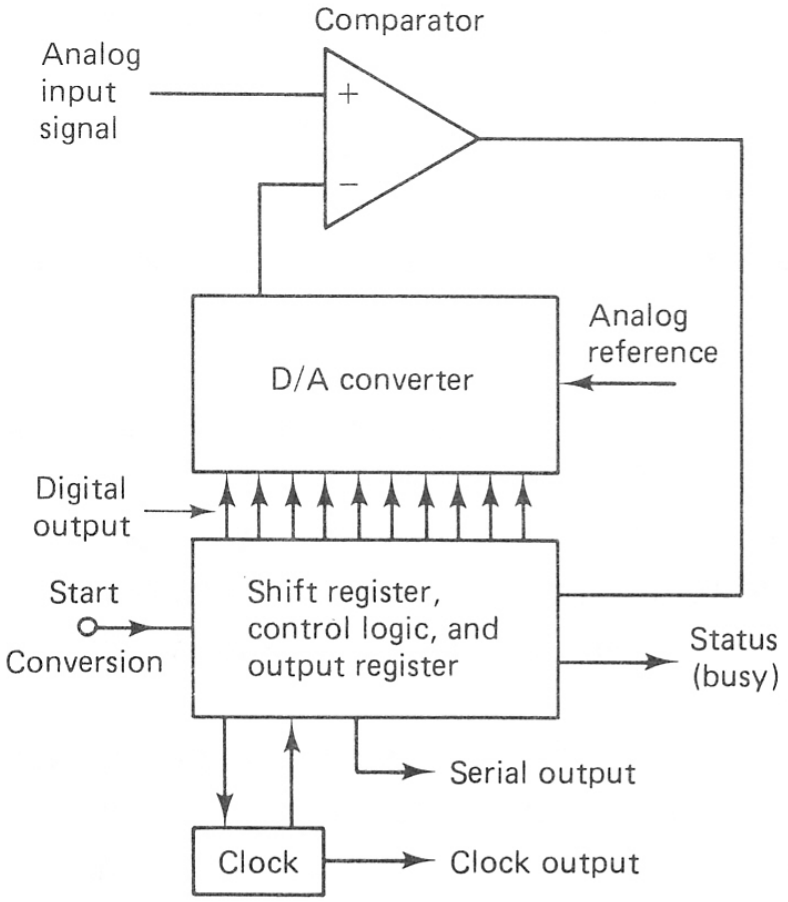
$$+1/2\ (\overline{V}_{in} + V_{ref})\ \frac{T}{RC} = V_{ref}\ \frac{\Delta t}{RC}$$

$$\frac{\Delta t}{T}\left(\begin{array}{l}\text{proportional to}\\ \text{full-scale count}\end{array}\right) = 1/2\left(\frac{\overline{V}_{in}}{V_{ref}} + 1\right)$$

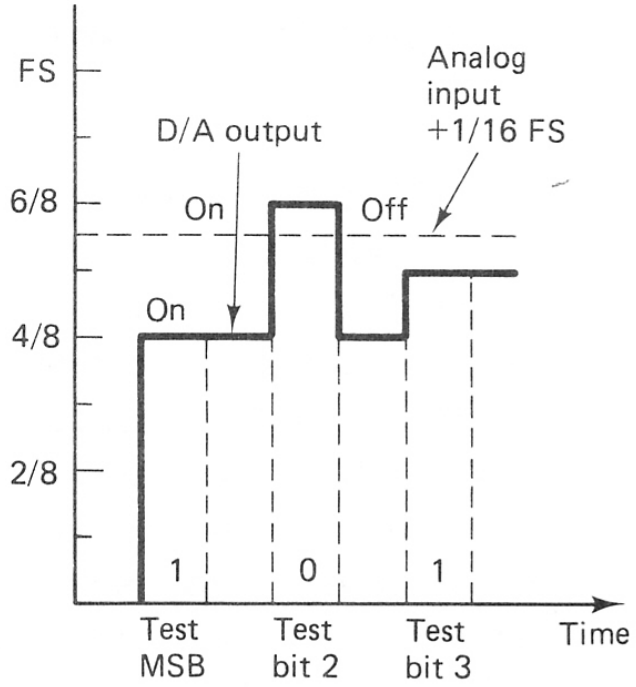
Tracking (counter-comparator)



Successive approximation



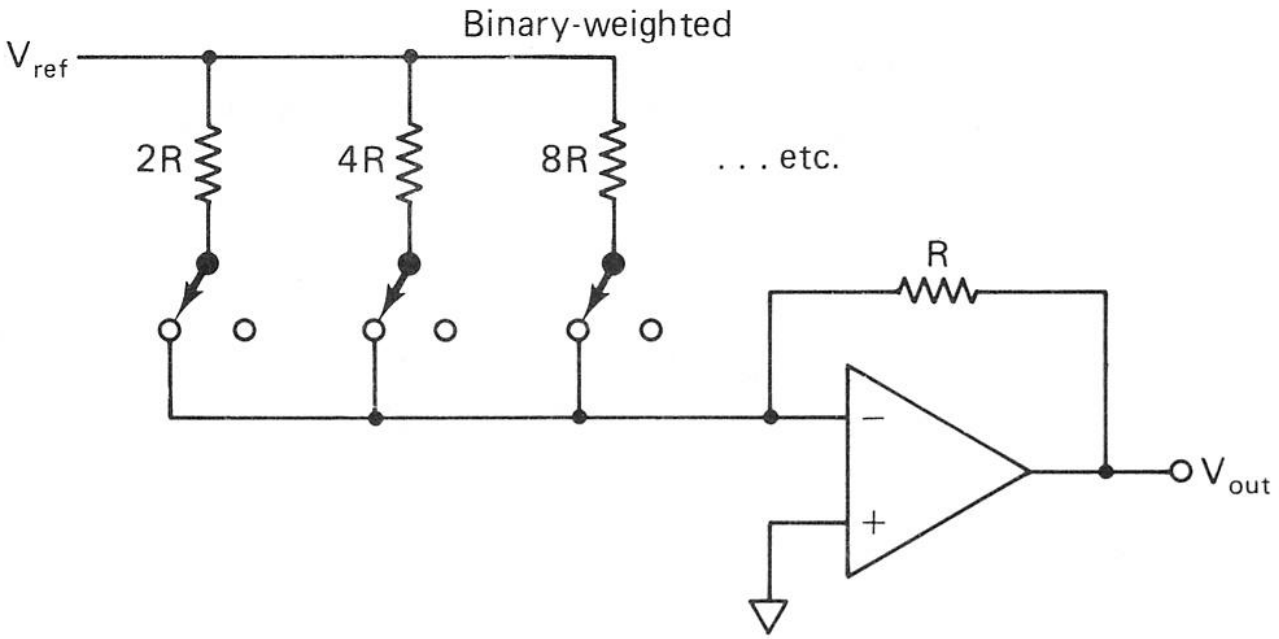
(a)



(b)

D/A Converter

Disadvantage:
Wide range of resistor values required

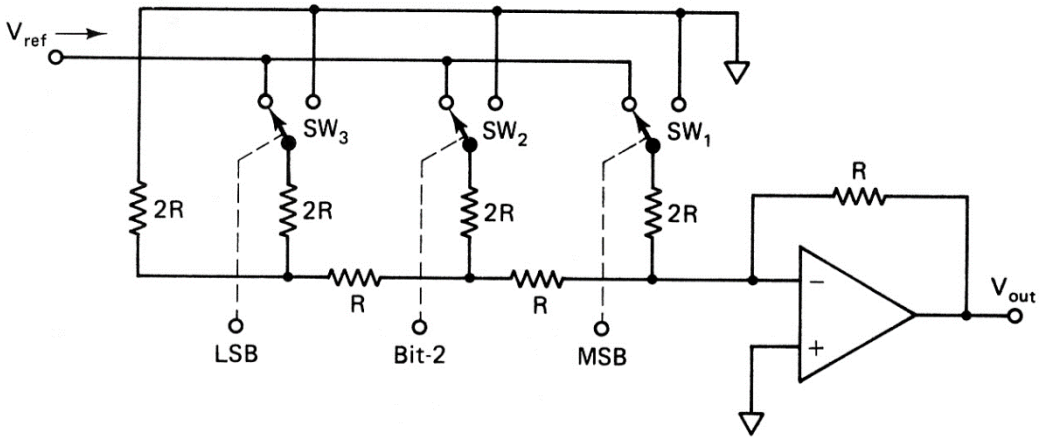


D/A Converter

Standard R-2R

Advantage:
Simple resistor ratios

Disadvantage:
Switch biasing

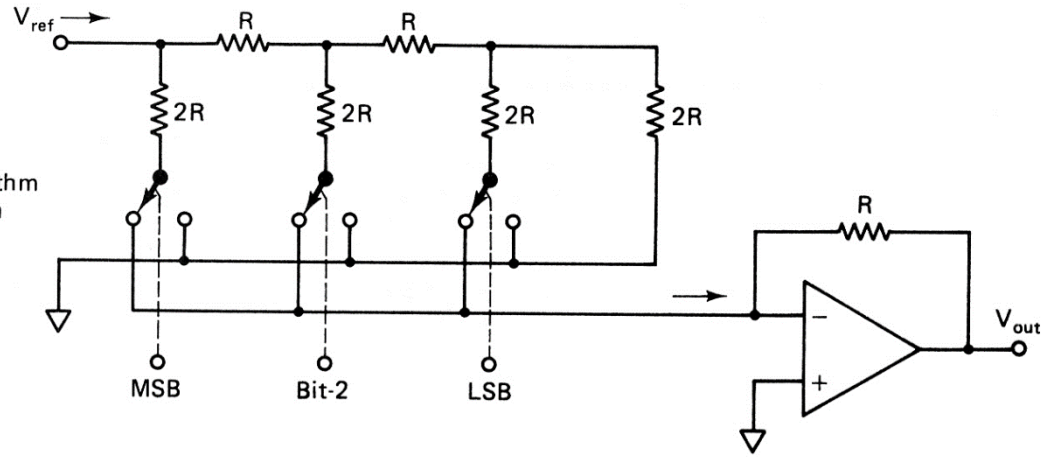


(a)

Inverted R-2R

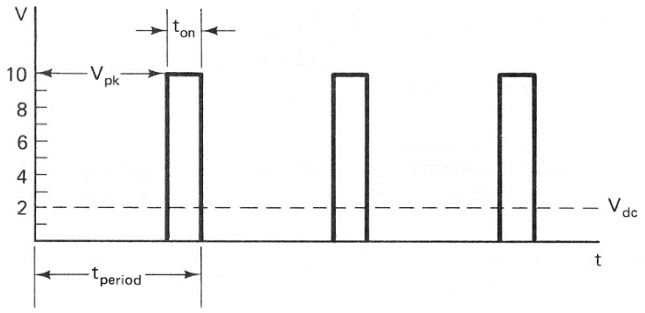
Advantage:
Simplicity

Disadvantages:
Trimming algorithm
Error interaction

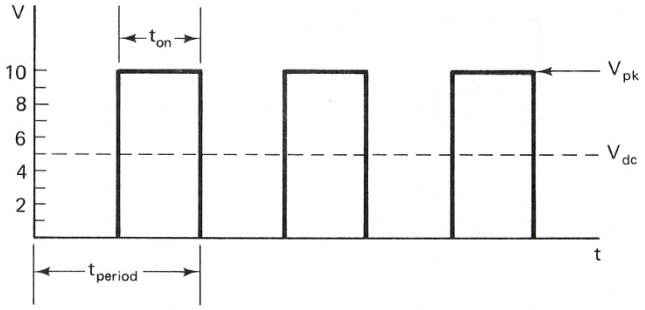


(b)

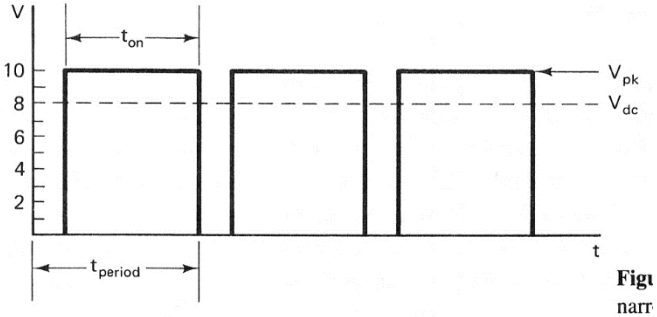
PWM (Pulse Width Modulation)



(a)



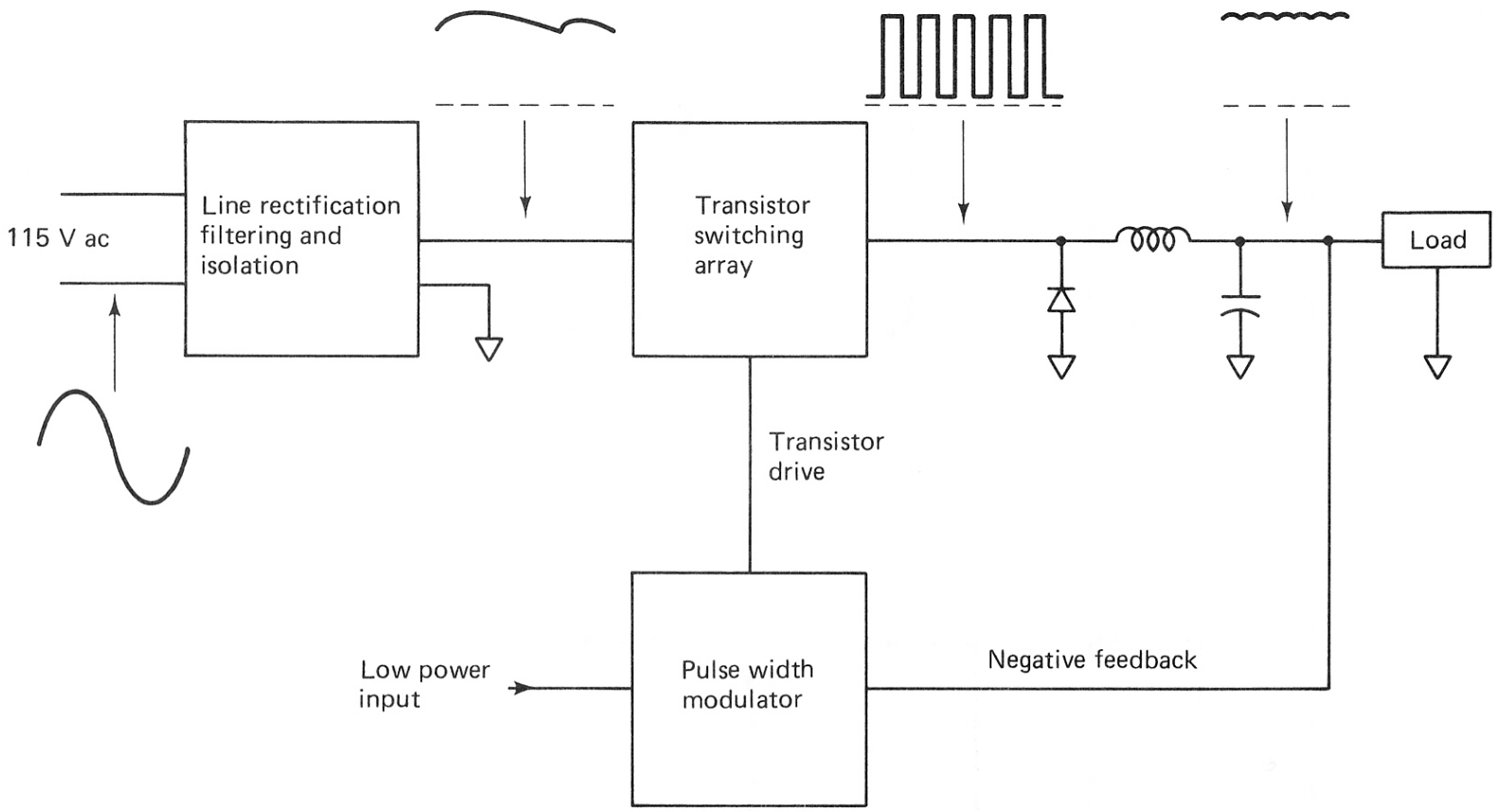
(b)



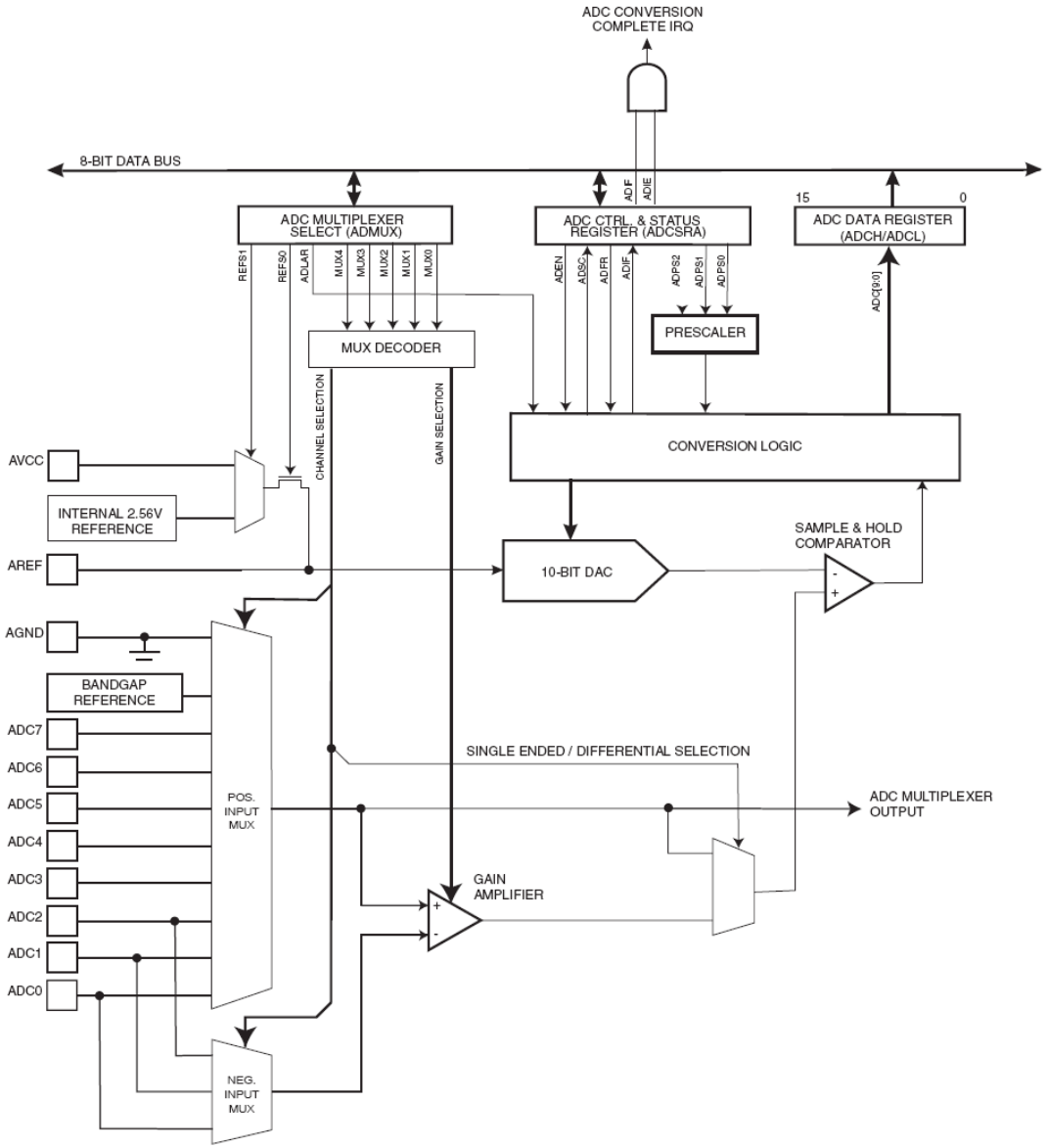
(c)

Figu
narr
(c) \

PWM (Pulse Width Modulation)



ATmega128 ADC



ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7:6 – REFS1:0: Reference Selection Bits**

These bits select the voltage reference for the ADC, as shown in Table 97. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set). The internal voltage reference options may not be used if an external reference voltage is being applied to the AREF pin.

Table 97. Voltage Reference Selections for ADC

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

- **Bit 5 – ADLAR: ADC Left Adjust Result**

The ADLAR bit affects the presentation of the ADC conversion result in the ADC Data Register. Write one to ADLAR to left adjust the result. Otherwise, the result is right adjusted. Changing the

ADC Multiplexer Selection Register

- **Bits 4:0 – MUX4:0: Analog Channel and Gain Selection Bits**

The value of these bits selects which combination of analog inputs are connected to the ADC. These bits also select the gain for the differential channels. See Table 98 for details. If these bits are changed during a conversion, the change will not go in effect until this conversion is complete (ADIF in ADCSRA is set).

Table 98. Input Channel and Gain Selections

MUX4..0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
00000	ADC0	N/A		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000(1)		ADC0	ADC0	10x
01001		ADC1	ADC0	10x

ADC Control & Status Register

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

- **Bit 5 – ADFR: ADC Free Running Select**

When this bit is written to one, the ADC operates in Free Running mode. In this mode, the ADC samples and updates the data registers continuously. Writing zero to this bit will terminate Free Running mode.

ADC Control & Status Register

- **Bit 4 – ADIF: ADC Interrupt Flag**

This bit is set when an ADC conversion completes and the data registers are updated. The ADC Conversion Complete Interrupt is executed if the ADIE bit and the I-bit in SREG are set. ADIF is cleared by hardware when executing the corresponding interrupt handling vector. Alternatively, ADIF is cleared by writing a logical one to the flag. Beware that if doing a read-modify-write on ADCSRA, a pending interrupt can be disabled. This also applies if the SBI and CBI instructions are used.

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

- **Bits 2:0 – ADPS2:0: ADC Prescaler Select Bits**

These bits determine the division factor between the XTAL frequency and the input clock to the ADC.

Table 99. ADC Prescaler Selections

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

ADC Data Register

ADLAR = 0:

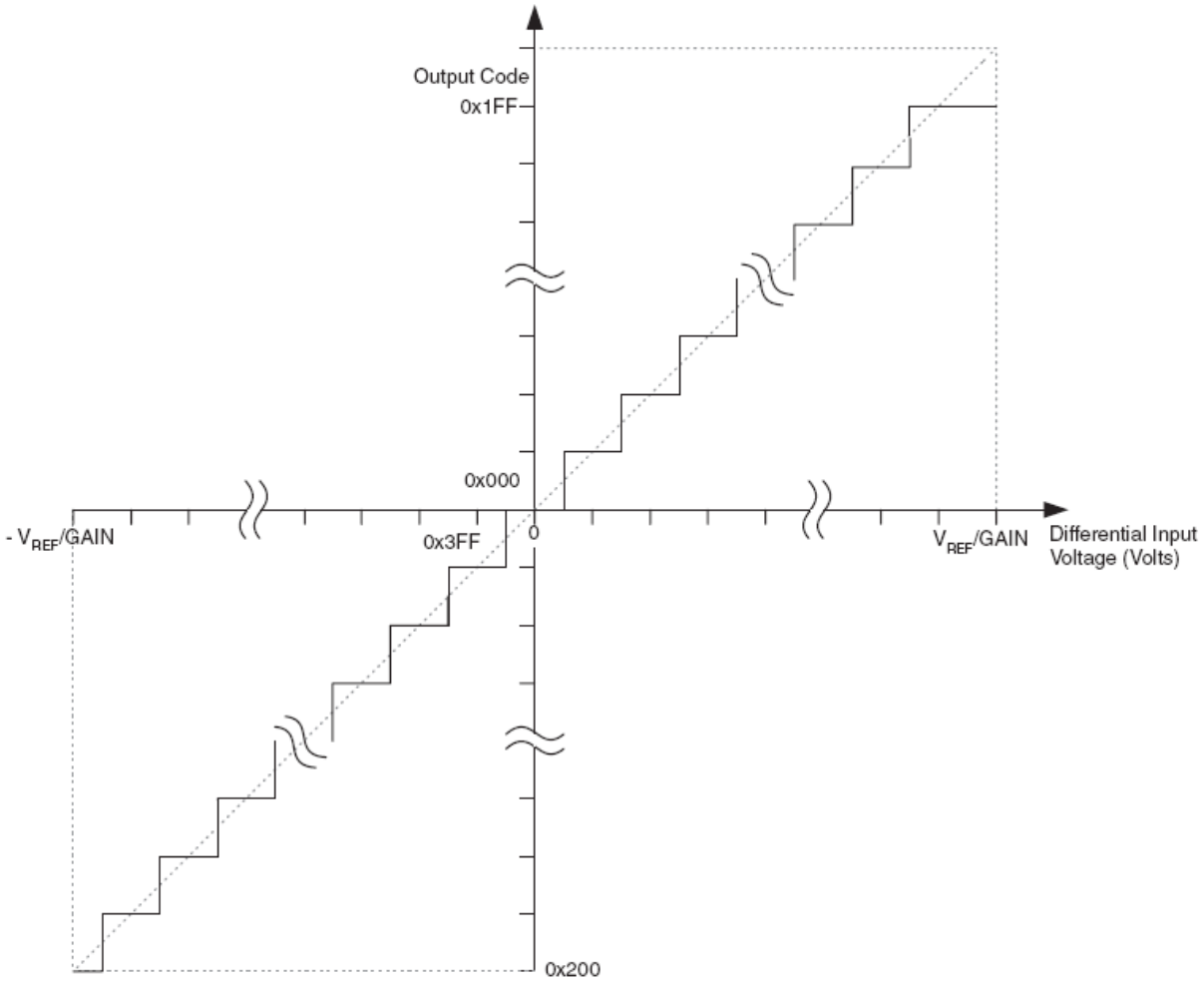
Bit	15	14	13	12	11	10	9	8	
	–	–	–	–	–	–	ADC9	ADC8	ADCH
	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ADLAR = 1:

Bit	15	14	13	12	11	10	9	8	
	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R	R	R	R	R	
	R	R	R	R	R	R	R	R	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

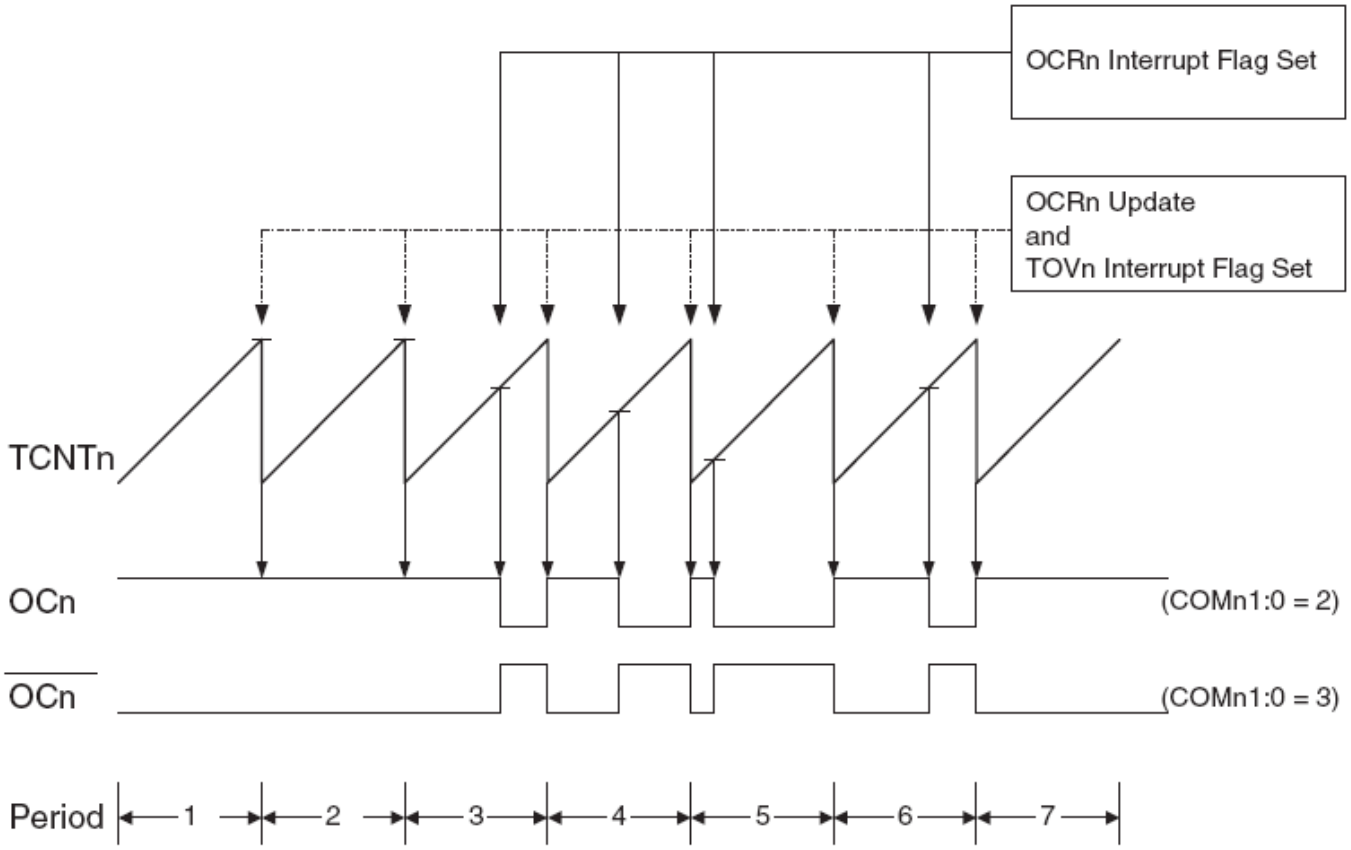
ADC Differential Mode

Figure 119. Differential Measurement Range



Timer PWM mode

Figure 39. Fast PWM Mode, Timing Diagram



Timer PWM mode

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – FOC0: Force Output Compare**

The FOC0 bit is only active when the WGM bits specify a non-PWM mode. However, for ensuring compatibility with future devices, this bit must be set to zero when TCCR0 is written when operating in PWM mode. When writing a logical one to the FOC0 bit, an immediate compare match is forced on the waveform generation unit. The OC0 output is changed according to its COM01:0 bits setting. Note that the FOC0 bit is implemented as a strobe. Therefore it is the value present in the COM01:0 bits that determines the effect of the forced compare.

A FOC0 strobe will not generate any interrupt, nor will it clear the timer in CTC mode using OCR0 as TOP.

The FOC0 bit is always read as zero.

- **Bit 6, 3 – WGM01:0: Waveform Generation Mode**

These bits control the counting sequence of the counter, the source for the maximum (TOP) counter value, and what type of waveform generation to be used. Modes of operation supported by the Timer/Counter unit are: Normal mode, Clear Timer on Compare match (CTC) mode, and two types of Pulse Width Modulation (PWM) modes. See Table 52 and "Modes of Operation" on page 98.

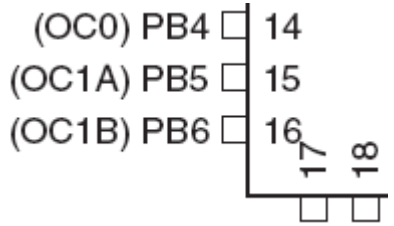
Timer PWM mode

Table 52. Waveform Generation Mode Bit Description

Mode	WGM01[1] (CTC0)	WGM00[1] (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0 at	TOV0 Flag Set on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	BOTTOM	MAX

Table 54. Compare Output Mode, Fast PWM Mode[1]

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at BOTTOM, (non-inverting mode)
1	1	Set OC0 on compare match, clear OC0 at BOTTOM, (inverting mode)



Timer PWM mode

Table 56. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	clk_{T0S}/(No prescaling)
0	1	0	clk_{T0S}/8 (From prescaler)
0	1	1	clk_{T0S}/32 (From prescaler)
1	0	0	clk_{T0S}/64 (From prescaler)
1	0	1	clk_{T0S}/128 (From prescaler)
1	1	0	clk_{T0S}/256 (From prescaler)
1	1	1	clk_{T0S}/1024 (From prescaler)

Bit	7	6	5	4	3	2	1	0	
	OCR0[7:0]								OCR0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register contains an 8-bit value that is continuously compared with the counter value (TCNT0). A match can be used to generate an output compare interrupt, or to generate a waveform output on the OC0 pin.

Sample Code (1)

```
int main(void)
{
    MCUCR = 0x80; // Address and data bus

    DDRD = 0xf0;
    DDRB = 0x10;

    ADCH = 0x00;
    ADCL = 0x00;
    ADMUX = 0x00;

    LCD_init();
    TCCR0 = 0x7c; /* Timer0 Fast PWM clk/64 */
    while(1)
    {
        ADCSR = 0xc7;
        while((ADCSR & 0x10)==0);
        ADC_L = ADCL;
        ADC_H = ADCH;
        PORTD = ~(ADC_L << 4);
        OCR0=(ADC_L>>2) | (ADC_H<<6);
```

Sample Code (2)

```
            ADC_RESULT = 0x00;
            ADC_RESULT = ADC_RESULT | ADC_H;
            ADC_RESULT = 0x0300 & (ADC_RESULT << 8);
            ADC_RESULT = ADC_RESULT | ADC_L;

            c4 = ADC_RESULT / 1000;
            c3 = (ADC_RESULT % 1000) / 100;
            c1 = ADC_RESULT % 100;
            c2 = c1/10;
            c1 = c1 % 10;
            LCD_pos(0,0);
            LCD_STR((char*)"ADC:");
            LCD_CHAR(c4+0x30);
            LCD_CHAR(c3+0x30);
            LCD_CHAR(c2+0x30);
            LCD_CHAR(c1+0x30);
            _delay_ms(5);
    }
}
```