

# Arquitetura de Sistemas Embarcados

---

**Edna Barros** ([ensb@cin.ufpe.br](mailto:ensb@cin.ufpe.br))



Centro de Informática – UFPE

## Overview

---

- Processadores de propósito único
  - Customizados
  - Padronizados
- Processadores de periféricos
  - Temporizadores
  - Contadores
  - UART
  - PWM
  - LCD
  - Teclado
  - ....



## Introdução

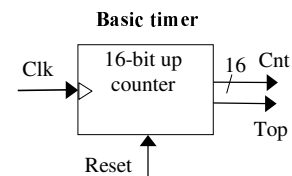
- Processadores de propósito único
  - Realizam computação específica
  - Processadores customizados
    - Projetados para uma unica tarefa
  - *Processadores de propósito único padronizados*
    - “Off-the-shelf” --
    - Exemplos:
      - Periféricos
      - Transmissão serial
      - Conversão analógico-digital

## Temporizadores, contadores, watchdog timers

- Temporizadores: mede intervalos de tempo
  - Geração de saída para eventos temporais
    - Ex: sinal verde deve permanecer por 10 seg.
  - Para medição de eventos em entradas
    - Ex: medida de velocidade de automóvel

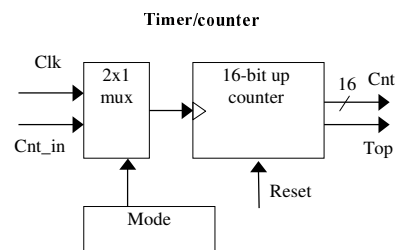
## Temporizadores, contadores, watchdog timers

- Baseado na contagem de pulsos de clock
  - Considere período do clock = 10ns
  - Se contamos 20.000 pulsos de relógio
  - Então se passaram 200 microsegundos
  - Um contador de 16-bits deveria contar até  $65,535 \times 10 \text{ ns} = 655.35 \text{ microseg.}$ , resolução = 10 ns



## Contadores

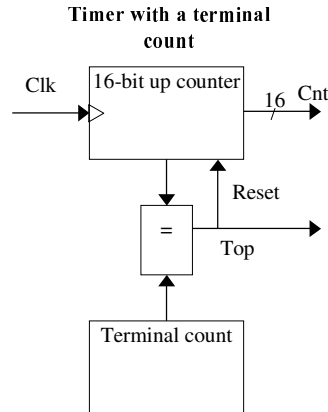
- Contador: semelhante a um temporizador porém conta pulsos de uma entrada genérica (em vez do timer)
  - Ex: contagem de carros passando por um sensor
  - Dispositivo pode ser configurado como temporizador ou contador



## Outras estruturas temporizadoras

- Temporizadores de Intervalos

- Indica quando um intervalo de tempo ocorreu
- Setando o Intervalo desejado
  - *Número de ciclos de clock = Intervalo desejado / Período de clock*

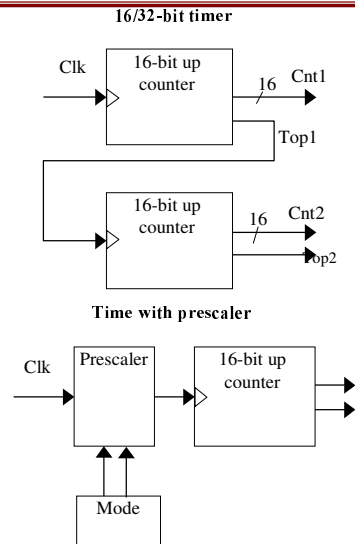


## Outras estruturas temporizadoras

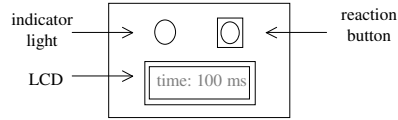
- Contadores em cascata

- Prescaler

- Divisão do clock
- Aumenta o range, diminui a resolução



## Exemplo: Temporizador de Reação



- Medindo o tempo entre acendimento da lampada e o usuário apertar o botão
  - Temporizador de 16-bits, período do clk de 83.33 ns, contador incrementa a cada 6 ciclos
  - Resolução =  $6 \cdot 83.33 = 0.5$  microseg.
  - Range =  $65535 \cdot 0.5$  microseg = 32.77 mileseg.
  - Para contar milesegundos contador deve ser inicializado com 65535 –  $1000/0.5 = 63535$

```

/* main.c */

#define MS_INIT 63535
void main(void){
    int count_milliseconds = 0;

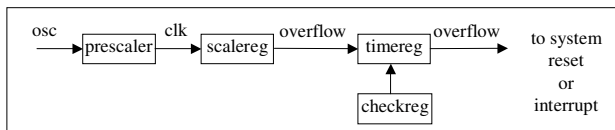
    configure timer mode
    set Cnt to MS_INIT

    wait a random amount of time
    turn on indicator light
    start timer

    while (user has not pushed reaction button){
        if(Top) {
            stop timer
            set Cnt to MS_INIT
            start timer
            reset Top
            count_milliseconds++;
        }
    }
    turn light off
    printf("time: %i ms", count_milliseconds);
}
    
```

## Watchdog timer

- Deve resetar o temporizador a cada X unidades de tempo, caso contrário o temporizador gera um sinal
- Uso comum: detecção de falha, self-reset



```

/* main.c */

main(){
    wait until card inserted
    call watchdog_reset_routine

    while(transaction in progress){
        if(button pressed){
            perform corresponding action
            call watchdog_reset_routine
        }
    }

    /* if watchdog_reset_routine not called every
    < 2 minutes, interrupt_service_routine is
    called */
}
    
```

```

watchdog_reset_routine(){
    /* checkreg is set so we can load value into
    timereg. Zero is loaded into scalereg and
    11070 is loaded into timereg */

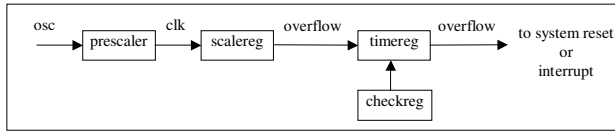
    checkreg = 1
    scalereg = 0
    timereg = 11070
}

void interrupt_service_routine(){
    eject card
    reset screen
}
    
```

## Watchdog timer

- Outro uso: timeouts

- Ex., máquina ATM
- 16-bit timer, 2 microsec. resolution
- *timereg* value =  $2 \cdot (2^{16} - 1) - X = 131070 - X$
- For 2 min.,  $X = 120,000$  microsec.



```

/* main.c */
main(){
  wait until card inserted
  call watchdog_reset_routine

  while(transaction in progress){
    if(button pressed){
      perform corresponding action
      call watchdog_reset_routine
    }
  }

  /* if watchdog_reset_routine not called every
  < 2 minutes, interrupt_service_routine is
  called */
}

```

```

watchdog_reset_routine(){
  /* checkreg is set so we can load value into
  timereg. Zero is loaded into scalereg and
  11070 is loaded into timereg */

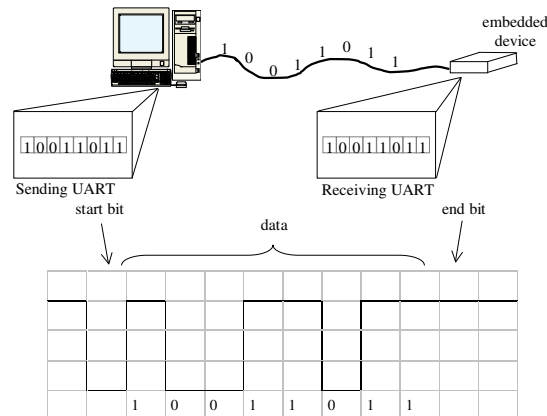
  checkreg = 1
  scalereg = 0
  timereg = 11070
}

void interrupt_service_routine(){
  eject card
  reset screen
}

```

## Transmissão Serial Usando UARTs

- UART: Universal Asynchronous Receiver Transmitter
  - Transmite dado serialmente
  - Recebe dado serialmente e converte para paralelo
- Parity: bit extra para detecção de erros
- Bits de Start e de stop
- Baud rate
  - Mudanças de sinal por segundo
  - Bit rate é usada frequentemente



## Pulse width modulator - PWM

- Gera pulsos com tempo de high/low especificado
- Duty cycle: % do tempo em nível alto
  - Onda quadrada: 50% duty cycle

25% duty cycle – average pwm\_o is 1.25V

50% duty cycle – average pwm\_o is 2.5V.

75% duty cycle – average pwm\_o is 3.75V.

Centro de Informática

Arquitetura de Sistemas Embarcados 13

## Pulse width modulator - PWM

- Uso comum: controle de tensão para equipamento elétrico
  - Mais simples que conversor DC-DC ou conversor digital-analógico
  - Velocidade de motor, lampadas com controle de luminosidade
- Outro uso: codificação de comandos, recebe temporização do usuário para decodificar

25% duty cycle – average pwm\_o is 1.25V

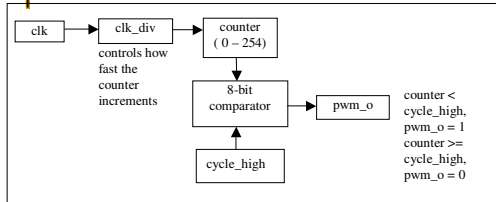
50% duty cycle – average pwm\_o is 2.5V.

75% duty cycle – average pwm\_o is 3.75V.

Centro de Informática

Arquitetura de Sistemas Embarcados 14

# Controlando um motor DC motor com um PWM



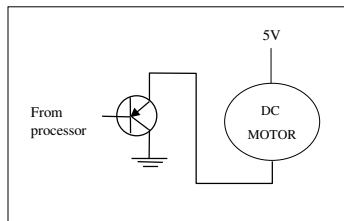
Input Voltage	% of Maximum Voltage Applied	RPM of DC Motor
0	0	0
2.5	50	1840
3.75	75	6900
5.0	100	9200

Relationship between applied voltage and speed of the DC Motor

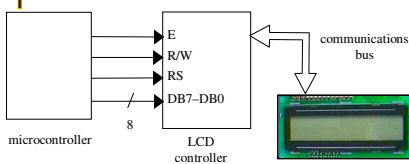
```
void main(void) {
    /* controls period */
    PWMP = 0xFF;
    /* controls duty cycle */
    PWM1 = 0x7F;

    while(1) {};
}
```

The PWM alone cannot drive the DC motor, a possible way to implement a driver is shown below using an MJE3055T NPN transistor.



# Controlador LCD



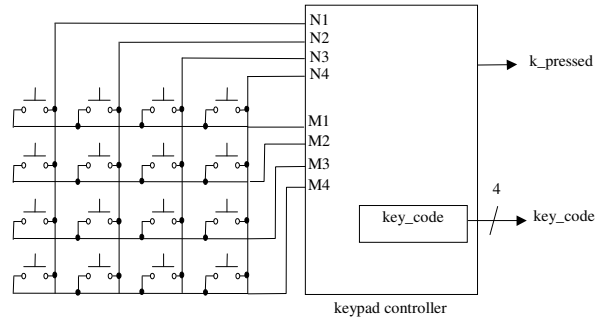
```
void WriteChar(char c){
    RS = 1;           /* indicate data being sent */
    DATA_BUS = c;   /* send data to LCD */
    EnableLCD(45);   /* toggle the LCD with appropriate delay */
}
```

CODES	
I/D = 1 cursor moves left	DL = 1 8-bit
I/D = 0 cursor moves right	DL = 0 4-bit
S = 1 with display shift	N = 1 2 rows
S/C = 1 display shift	N = 0 1 row
S/C = 0 cursor movement	F = 1 5x10 dots
R/L = 1 shift to right	F = 0 5x7 dots
R/L = 0 shift to left	

RS	R/W	DB <sub>7</sub>	DB <sub>6</sub>	DB <sub>5</sub>	DB <sub>4</sub>	DB <sub>3</sub>	DB <sub>2</sub>	DB <sub>1</sub>	DB <sub>0</sub>	Description
0	0	0	0	0	0	0	0	0	1	Clears all display, return cursor home
0	0	0	0	0	0	0	0	1	*	Returns cursor home
0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and/or specifies not to shift display
0	0	0	0	0	0	1	D	C	B	ON/OFF of all display(D), cursor ON/OFF (C), and blink position (B)
0	0	0	0	0	1	S/C	R/L	*	*	Move cursor and shifts display
0	0	0	0	1	DL	N	F	*	*	Sets interface data length, number of display lines, and character font
1	0	WRITE DATA								Writes Data



## Controlador de Teclado

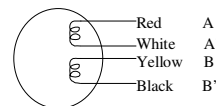
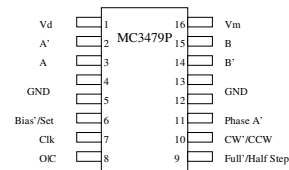


N=4, M=4

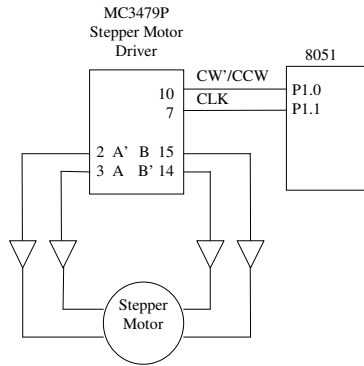
## Controlador de motor de passo

- Motor de passo: rotaciona número fixo de graus dado um sinal de passos
  - Em contraste: motor DC rotaciona quando potência é aplicada
- Rotação obtida pela aplicação de tensão especifica nas bobinas
- Controlador mais simples

Sequence	A	B	A'	B'
1	+	+	-	-
2	-	+	+	-
3	-	-	+	+
4	+	-	-	+
5	+	+	-	-



## Motor de Passo com Controlador (driver)



```

/* main.c */
sbit clk=P1^1;
sbit cw=P1^0;

void delay(void){
    int i, j;
    for (i=0; i<1000; i++)
        for (j=0; j<50; j++)
            i = i + 0;
}
    
```

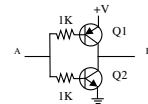
```

void main(void){

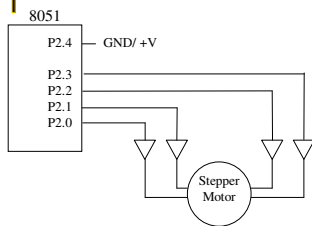
    /*turn the motor forward */
    cw=0;          /* set direction */
    clk=0;        /* pulse clock */
    delay();
    clk=1;

    /*turn the motor backwards */
    cw=1;          /* set direction */
    clk=0;        /* pulse clock */
    delay();
    clk=1;
}
    
```

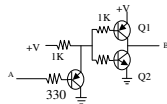
The output pins on the stepper motor driver do not provide enough current to drive the stepper motor. To amplify the current, a buffer is needed. One possible implementation is pictured to the left. Q1 is an MJE3055T NPN transistor and Q2 is an MJE2955T PNP transistor. A is connected to the 8051 microcontroller and B is connected to the stepper motor.



## Motor de Passo sem Controlador (driver)



A possible way to implement the buffers is located below. The 8051 alone cannot drive the stepper motor, so several transistors were added to increase the current going to the stepper motor. Q1 are MJE3055T NPN transistors and Q3 is an MJE2955T PNP transistor. A is connected to the 8051 microcontroller and B is connected to the stepper motor.



```

/*main.c*/
sbit notA=P2^0;
sbit isA=P2^1;
sbit notB=P2^2;
sbit isB=P2^3;
sbit dir=P2^4;
    
```

```

void delay(){
    int a, b;
    for(a=0; a<5000; a++)
        for(b=0; b<10000; b++)
            a=a+0;
}
    
```

```

void move(int dir, int steps) {
    int y, z;
    /* clockwise movement */
    if(dir == 1){
        for(y=0; y<=steps; y++){
            for(z=0; z<=19; z+4){
                isA=lookup[z];
                isB=lookup[z+1];
                notA=lookup[z+2];
                notB=lookup[z+3];
                delay();
            }
        }
    }
}
    
```

```

/* counter clockwise movement */
    
```

```

if(dir==0){
    for(y=0; y<=steps; y++){
        for(z=19; z>=0; z - 4){
            isA=lookup[z];
            isB=lookup[z-1];
            notA=lookup[z-2];
            notB=lookup[z-3];
            delay( );
        }
    }
}

void main(){
    int z;
    int lookup[20] = {
        1, 1, 0, 0,
        0, 1, 1, 0,
        0, 0, 1, 1,
        1, 0, 0, 1,
        1, 1, 0, 0 };
    while(1){
        /*move forward, 15 degrees (2 steps) */
        move(1, 2);
        /* move backwards, 7.5 degrees (1step)*/
        move(0, 1);
    }
}
    
```

## Conversão AD ou DA

---

- AD: conversão analógico para digital
- DA: conversão digital para analógica

Arquitetura de Sistemas Embarcados 21

## Conversão Analógico-Digital

---

<p><math>V_{max} = 7.5V</math></p> <table border="0"><tr><td>7.0V</td><td>1111</td></tr><tr><td>6.5V</td><td>1110</td></tr><tr><td>6.0V</td><td>1101</td></tr><tr><td>5.5V</td><td>1100</td></tr><tr><td>5.0V</td><td>1011</td></tr><tr><td>4.5V</td><td>1010</td></tr><tr><td>4.0V</td><td>1001</td></tr><tr><td>3.5V</td><td>1000</td></tr><tr><td>3.0V</td><td>0111</td></tr><tr><td>2.5V</td><td>0110</td></tr><tr><td>2.0V</td><td>0101</td></tr><tr><td>1.5V</td><td>0100</td></tr><tr><td>1.0V</td><td>0011</td></tr><tr><td>0.5V</td><td>0010</td></tr><tr><td>0V</td><td>0001</td></tr><tr><td></td><td>0000</td></tr></table> <p style="text-align: center;">proportionality</p>	7.0V	1111	6.5V	1110	6.0V	1101	5.5V	1100	5.0V	1011	4.5V	1010	4.0V	1001	3.5V	1000	3.0V	0111	2.5V	0110	2.0V	0101	1.5V	0100	1.0V	0011	0.5V	0010	0V	0001		0000	<p style="text-align: center;">analog to digital</p>	<p style="text-align: center;">digital to analog</p>
7.0V	1111																																	
6.5V	1110																																	
6.0V	1101																																	
5.5V	1100																																	
5.0V	1011																																	
4.5V	1010																																	
4.0V	1001																																	
3.5V	1000																																	
3.0V	0111																																	
2.5V	0110																																	
2.0V	0101																																	
1.5V	0100																																	
1.0V	0011																																	
0.5V	0010																																	
0V	0001																																	
	0000																																	

Arquitetura de Sistemas Embarcados 22

## Conversão Digital Analógico

**Simplified binary weighted resistor DAC**

**R - 2R ladder DAC**

Arquitetura de Sistemas Embarcados 23

## Conversão Digital Analógico

### Four-Bit D/A Converter

One way to achieve [D/A conversion](#) is to use a [summing amplifier](#).

Inputs in volts are weighted in the summing amplifier to produce the corresponding analog voltage.

$$1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 = 13$$

Scaled resistances into a summing junction.

A 12 bit DAC of this type would require the largest scaling resistor to be 2048 times the smallest, so this approach quickly becomes impractical.

This approach is not satisfactory for a large number of bits because it requires too much precision in the summing resistors. This problem is overcome in the [R-2R network](#) DAC.

Arquitetura de Sistemas Embarcados 24

## Conversão Digital Analógico

### R-2R Ladder DAC

The [summing amplifier](#) with the R-2R ladder of resistances shown produces the output

$$V_{out} = \frac{R_f}{R} V_{ref} \left[ \frac{D_0}{16} + \frac{D_1}{8} + \frac{D_2}{4} + \frac{D_3}{2} \right]$$

where the D's take the value 0 or 1. The digital inputs could be [TTL](#) voltages which close the [switches](#) on a logical 1 and leave it grounded for a logical 0. This is illustrated for 4 bits, but can be extended to any number with just the resistance values R and 2R.

[More Detail](#)

Arquitetura de Sistemas Embarcados 25

## Conversão Digital Analógico

### R-2R Ladder DAC Details

A common reference voltage is used because of the variation in what voltage will be produced by a logic 1 or 0.

A logic 1 (e.g., TTL 5V) closes the switch, applying Vref to the summing junction. A logic 0 leaves that input grounded.

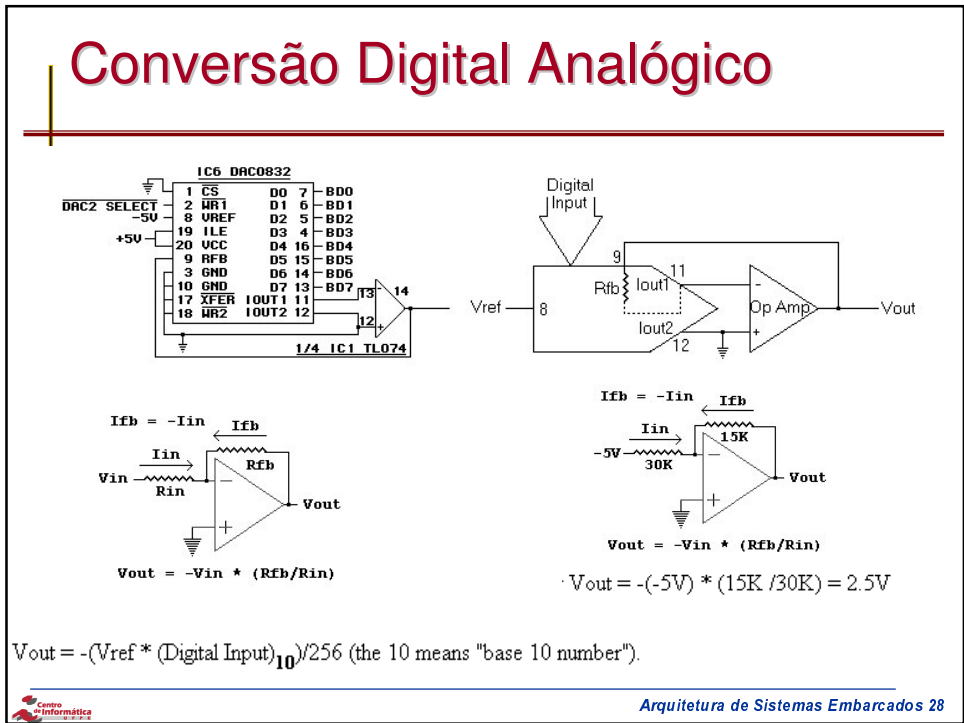
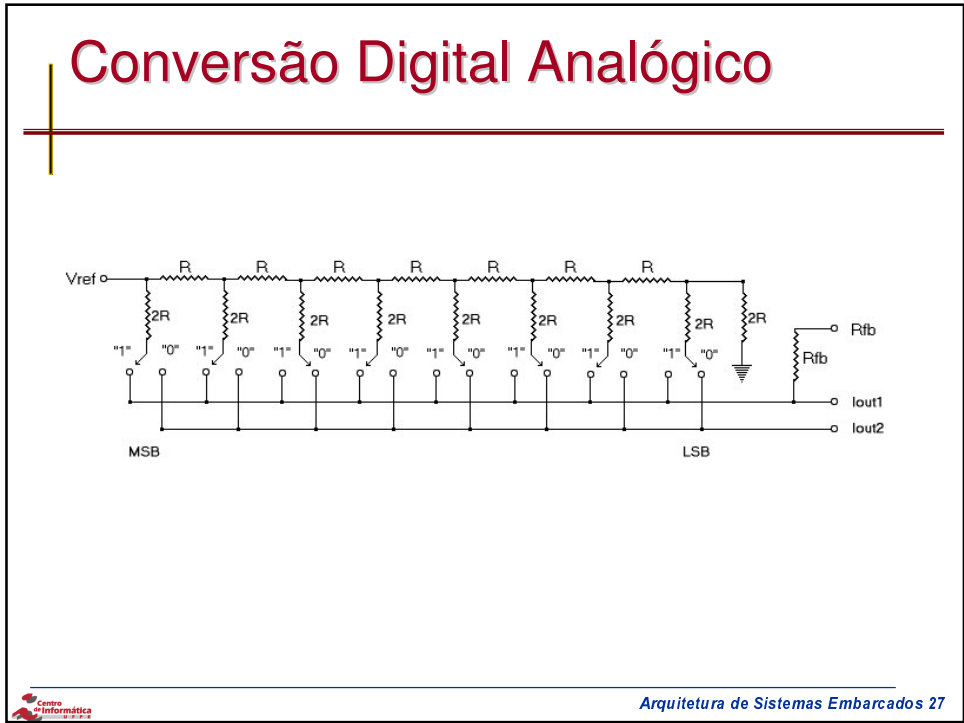
Summing amplifier.

$$V_{out} = \frac{R_f}{R} V_{ref} \left[ \frac{D_0}{16} + \frac{D_1}{8} + \frac{D_2}{4} + \frac{D_3}{2} \right]$$

Having just R and 2R places much less stringent requirements on the required precision of the resistors than the **scaled resistor** approach to DAC.

These weighting factors can be obtained by a **Thevenin** analysis of each input point.

Arquitetura de Sistemas Embarcados 26

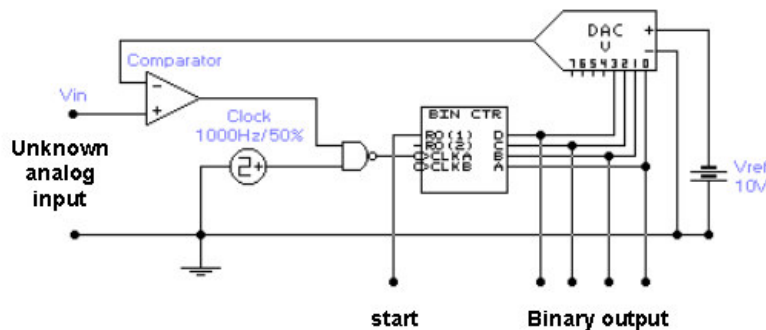


## Conversão Analógico Digital

- Exemplo: valor de entrada no range de 0 a 7.5 que deve ser representada com 4 dígitos. Qual a codificação ideal?
- Fórmula Geral
  - $e/V_{\max} = d/(2^n - 1)$

## Conversão AD

### Simple ramp and comparator ADC



## Conversão Analógico Digital usando Aproximações sucessivas

Given an analog input signal whose voltage should range from 0 to 15 volts, and an 8-bit digital encoding, calculate the correct encoding for 5 volts. Then trace the successive-approximation approach to find the correct encoding.

$$5/15 = d/(2^8-1) \quad \text{Encoding: } 01010101$$

$$d = 85$$

### Successive-approximation method

$\frac{1}{2}(V_{\max} - V_{\min}) = 7.5$ volts $V_{\max} = 7.5$ volts.	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	$\frac{1}{2}(5.63 + 4.69) = 5.16$ volts $V_{\max} = 5.16$ volts.	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
$\frac{1}{2}(7.5 + 0) = 3.75$ volts $V_{\min} = 3.75$ volts.	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	$\frac{1}{2}(5.16 + 4.69) = 4.93$ volts $V_{\min} = 4.93$ volts.	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
$\frac{1}{2}(7.5 + 3.75) = 5.63$ volts $V_{\max} = 5.63$ volts.	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	$\frac{1}{2}(5.16 + 4.93) = 5.05$ volts $V_{\max} = 5.05$ volts.	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
$\frac{1}{2}(5.63 + 3.75) = 4.69$ volts $V_{\min} = 4.69$ volts.	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	$\frac{1}{2}(5.05 + 4.93) = 4.99$ volts	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

## Conversão AD – Aproximação sucessivas

- **Fast process - 1 - 100µsecs**
- **Result always n clocks after start**
- **Used extensively for 12-16bit DAQ systems**

