Arquitetura de Sistemas Embarcados

Edna Barros *(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

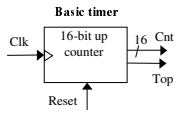
- Processadores de propósito único
  - Realizam computação específica
  - Processadores customizados
    - Projetados para uma unica tarefa
  - *Processadores de propósito único padronizados*
    - "Off-the-shelf" --
    - Exemplos:
      - Periféricos
      - Transmissão serial
      - Conversão analógico-digital

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

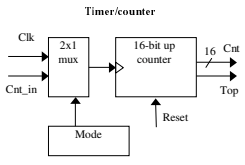
Temporizadores, contadores, watchdog timers

- Baseado na contagem de pulsos de clock
  - Considere periodo 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*10 ns = 655.35 microseg., resolução = 10 ns



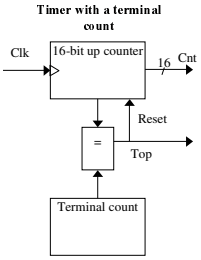
Contadores

- 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



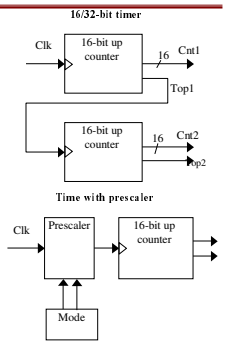
Outras estruturas temporizadoras

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

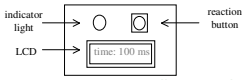


Outras estruturas temporizadoras

- Contadores em cascata
- Prescaler
  - Divisão do clock
  - Aumenta o range, diminui a resolução



Exemplo: Temporizador de Reação

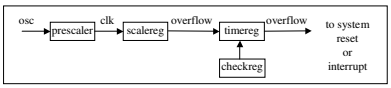


- Medindo o tempo entre acendimento da lampada e o usuário apertar o botão
  - Temporizador de 16-bits, período do clk de 83.33 ns, contador incrementa a cada 6 ciclos
  - Resolução = 6*83.33=0.5 microseg.
  - Range = 65535*0.5 microseg = 32.77 mileseg.
  - Para contar milesegundos contador deve ser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

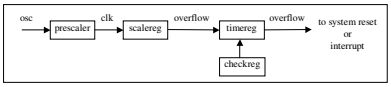



```
/* main.c */

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 = $2*(2^{16}-1)-X = 131070-X$
  - For 2 min., X = 120,000 microsec.

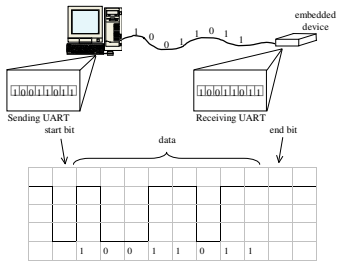



```
/* main.c */

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
}
```

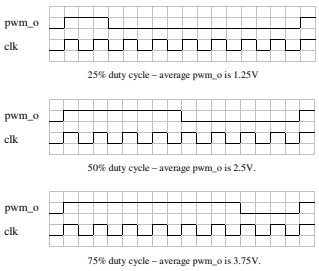

Transmissão Serial Usando UARTs

- 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



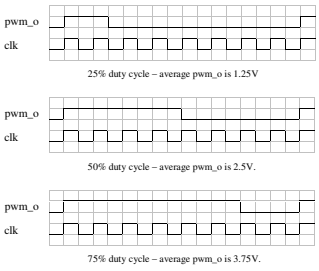
Pulse width modulator - PWM

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

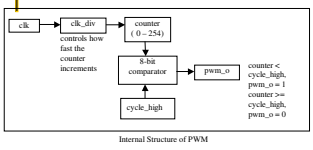


Pulse width modulator - PWM

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



Controlando um motor DC motor com um PWM



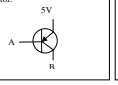
Internal Structure of PWM

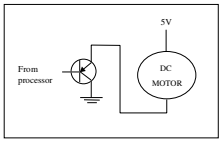
Input Voltage	% of Maximum Voltage Applied	RPM of DC Motor
0	0	0
2.5	50	1840
3.75	75	6900
5.0	100	9200

Relationship between applied voltage and speed of the DC Motor

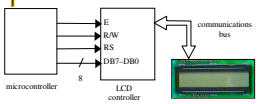
```
void main(void){

   /* 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

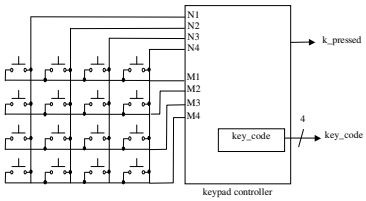



```
void WriteChar(char c){

  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_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0	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

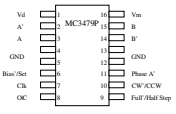


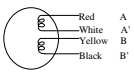
N=4, M=4

Controlador de motor de passo

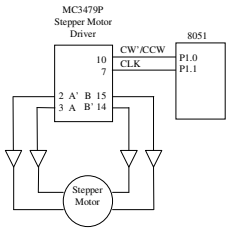
- Motor de passo: rotaciona número fixo de graus dado um sinal de passos
  - Em contraste: motor DC rotaciona quando potência é aplicada
- Rotação obtida pela aplicação de tensão especifica nas bobinas
- Controlador mais simples

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





Motor de Passo com Controlador (driver)

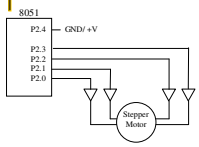



```
/* main.c */

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;
}
```

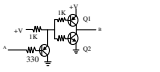
```
void main(void){

  */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 of the buffers 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.



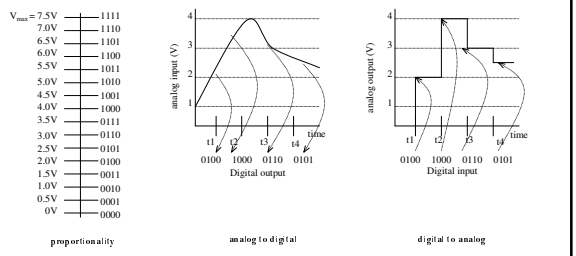

```
/*main.c*/
sbit notA=P2^0;
sbit isA=P2^1;
sbit notB=P2^2;
sbit isB=P2^3;
sbit dir=P2^4;

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<=step; 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

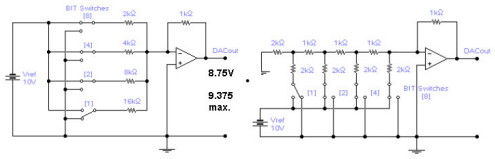
- AD: conversão analógico para digital
- DA: conversão digital para analógica

Conversão Analógico-Digital



Conversão Digital Analógico

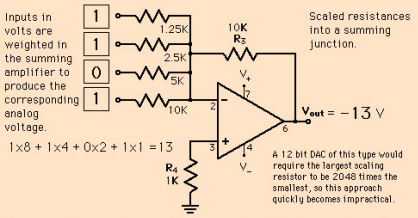
Simplified binary weighted resistor DAC **R - 2R ladder DAC**



Conversão Digital Analógico

Four-Bit D/A Converter

One way to achieve D/A conversion is to use a summing amplifier.



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.

Conversão Digital Analógico

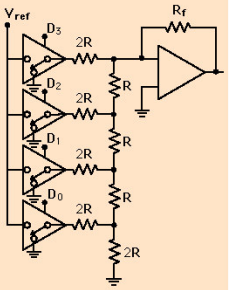
R-2R Ladder DAC

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

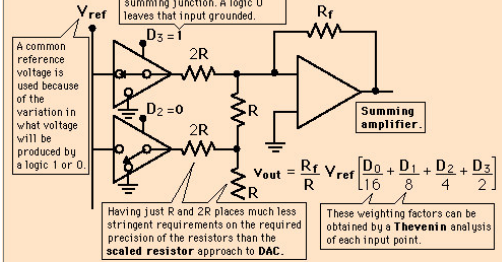


Conversão Digital Analógico

R-2R Ladder DAC Details

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

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

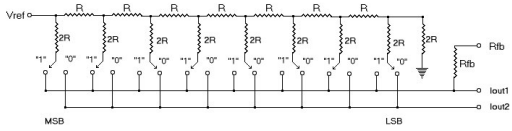


$$V_{out} = \frac{R_f}{R} V_{ref} \left[\frac{D_0}{16} + \frac{D_1}{8} + \frac{D_2}{4} + \frac{D_3}{2} \right]$$

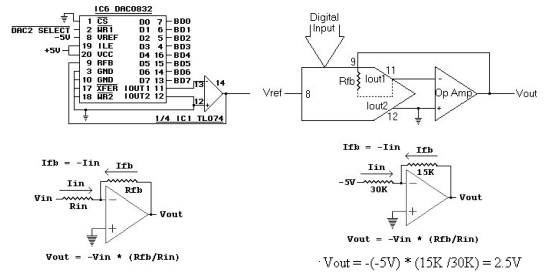
Having just R and 2R places much less stringent requirements on the required precision of the resistors than the **scaled resistor** approach to **DAC**.

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

Conversão Digital Analógico



Conversão Digital Analógico

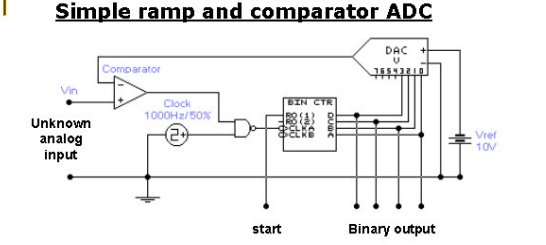


Vout = -(Vref * (Digital Input)$_{10}$)/256 (the 10 means "base 10 number").

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



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/(28-1) Encoding: 01010101
d= 85

Successive-approximation method

½(V_{max} – V_{min}) = 7.5 volts V_{max} = 7.5 volts.	0	0	0	0	0	0	0	0	
½(7.5 + 0) = 3.75 volts V_{min} = 3.75 volts.	0	1	0	0	0	0	0	0	
½(7.5 + 3.75) = 5.63 volts V_{max} = 5.63 volts	0	1	0	0	0	0	0	0	
½(5.63 + 3.75) = 4.69 volts V_{min} = 4.69 volts.	0	1	0	1	0	0	0	0	
½(5.63 + 4.69) = 5.16 volts V_{max} = 5.16 volts.	0	1	0	1	0	0	0	0	
½(5.16 + 4.69) = 4.93 volts V_{min} = 4.93 volts.	0	1	0	1	0	1	0	0	
½(5.16 + 4.93) = 5.05 volts V_{max} = 5.05 volts.	0	1	0	1	0	1	0	0	
½(5.05 + 4.93) = 4.99 volts	0	1	0	1	0	1	0	1	

Conversão AD – Aproximação sucessivas

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