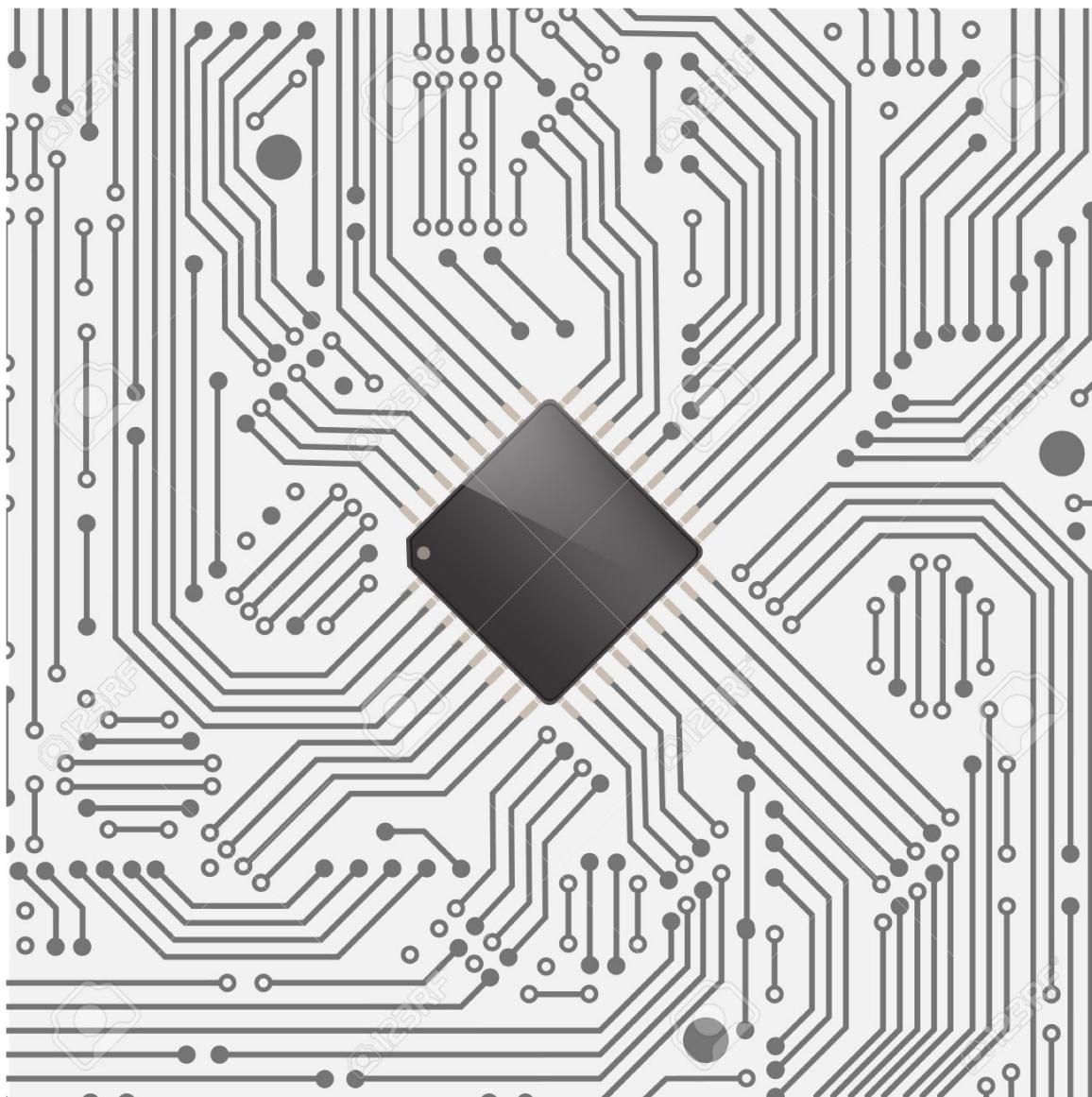


# Simulação do Microcontrolador 80c51 Utilizando o Software PROTEUS.

Trabalho Baseado no Artigo:  
2010-Application of Proteus Virtual System Modelling (VSM) in  
Teaching of Microcontroller

Aluno: Fabiano Aparecido Marino  
NºUsp:7143980

3 de janeiro de 2016





## Sumário

<b>Lista de Figuras</b>	<b>4</b>
<b>1 Proteus</b>	<b>5</b>
1.1 Proteus Interface . . . . .	6
1.2 Proteus ISIS Interface . . . . .	8
<b>2 Simulação 80c51 Assembler</b>	<b>9</b>
2.1 Práticas Botões e Leds . . . . .	9
2.2 Simulação Memória Ram e Rom Externas . . . . .	12
2.3 Comunicação Serial . . . . .	15
2.4 Timers, Interrupções e Contadores : Projeto Conversor AD . . . . .	19
2.4.1 Conversores e Breve Conversão AD e DA . . . . .	19
2.5 Conversor AD e DA Controlado Pelo 80c51 . . . . .	20
2.6 Motor de Passo . . . . .	24
2.7 Display de Sete Segmentos e LCD . . . . .	26
2.7.1 Display LCD . . . . .	28
<b>3 Projetos 80c51 em Linguagem C</b>	<b>31</b>
3.1 Motor de Passo Controlado Pela Interface Serial . . . . .	31
3.2 Frequêncimetro . . . . .	33
3.3 Termômetro com Tabela e Polinômio . . . . .	34
<b>4 Projeto de Dispositivos Decodificados</b>	<b>37</b>
<b>5 Conclusões</b>	<b>39</b>
<b>6 Apêndice1–Códigos–Assembler</b>	<b>40</b>
6.1 Programa:Chaves e Botões . . . . .	40
6.2 Prgrama:Chaves e Botões Com Opção de Piscar Led3 e Led1 . . . . .	41
6.3 Programa que Preenche uma Memória Ram Externa de 64Kbytes . . . . .	43
6.4 Envio Continuo de Caracter, Caracter Considerado 'F' . . . . .	44
6.5 Programa Que Envia Frase Para Serial Dependendo da Letra de Entrada . . . . .	45
6.6 Conversor AD DA ideal . . . . .	49
6.7 Exercício Conversor AD . . . . .	50
6.8 Exercício Motor de Passo . . . . .	52
6.9 Exercício:Display De Sete Segmentos . . . . .	54
6.10 Programa LCD 8 Bits . . . . .	55
6.11 Programa LCD 4 Bits . . . . .	57
<b>7 Programas em Linguagem C Compiladas por SDCC</b>	<b>59</b>
7.1 Botões e Chaves . . . . .	59
7.2 Envio Continuo de Caracter . . . . .	60
7.3 Envio De Caracter Via Interrupção . . . . .	61
7.4 LCD 8 Bits . . . . .	62
7.5 LCD 4 Bits . . . . .	64
7.6 Envio De Frase Dependendo da Letra . . . . .	66
7.7 Frequêncimetro . . . . .	68
7.7.1 Arquivo main.c . . . . .	68
7.7.2 arquivo lcd.c . . . . .	69
7.7.3 arquivo frequencimetro.h . . . . .	71
7.8 Motor de Passo . . . . .	72



7.8.1 Arquivo main.c . . . . .	72
7.8.2 arquivo serialconfig.c . . . . .	77
7.8.3 arquivo MOTORDEPASSO.h . . . . .	77
7.9 Termômetro Indefinido . . . . .	79
7.9.1 arquivo main.c . . . . .	79
7.9.2 arquivo timerinterruptconfig.c . . . . .	81
7.9.3 arquivo TERMOMETRO.h . . . . .	82
7.10 Termômetro com Tabela . . . . .	83
7.10.1 Arquivo main.c . . . . .	83
7.10.2 arquivo interruptconfig.c . . . . .	85
7.10.3 arquivo TERMOMETRO.h . . . . .	86
7.11 Termômetro com Polinômio . . . . .	87
7.11.1 Arquivo main.c . . . . .	87
7.11.2 arquivo TERMOMETRO.h . . . . .	89
7.11.3 arquivo serialtimerconfig . . . . .	90



## Lista de Figuras

1	VSM Proteus Vantagem de Uso . . . . .	5
2	Tela Inicial Proteus . . . . .	6
3	Tela Inicial Proteus ISIS . . . . .	8
4	Esquemático de Simulação Exercício Botões e Leds . . . . .	9
5	Fluxograma do exercício 1 . . . . .	10
6	Simulação Exercício 2 . . . . .	11
7	Hardware Ram e Rom de 8Kbytes . . . . .	13
8	Varrendo Todos os Valores do DPTR . . . . .	14
9	Selecionando Valores Vizualizaveis do DPTR . . . . .	14
10	Esquema de Hardware Para a Simulação da Comunicação Serial . . . . .	15
11	Bits que Representam a Comunicação Serial do Caracter F . . . . .	16
12	Hardware Que Possibilita a Simulação de Envio de Frase Dependente de Botão. . . . .	17
13	Fluxograma do Recebimento de um Caracter sem Interrupção . . . . .	18
14	Simulação Conversores AD DA e 80c51 . . . . .	22
15	Simulação Conversores AD DA e 80c51 . . . . .	22
16	Simulação Conversores AD DA e 80c51 . . . . .	24
17	Hardware Utilizado Para a Simulação do Motor de Passo . . . . .	25
18	Fluxograma Motor de Passo . . . . .	26
19	Fluxograma Display de Sete Segmentos . . . . .	27
20	Esquemático para Simulação dos Displays de Sete Segmentos . . . . .	28
21	Fluxograma Expressando o Funcionamento do Display LCD Tanto em 8 Como em Quattro Bits . . . . .	29
22	Esquemático Expressando o Funcionamento do 80c51 Junto ao LCD na Configuração 4 bits . . . . .	30
23	Esquemático Expressando o Funcionamento do 80c51 Junto ao LCD na Configuração 8 bits . . . . .	30
24	Terminal Serial Simulado no ISIS . . . . .	31
25	Fluxograma Expressando a Construção do Porgrama do Controle Do Motor de Passo . . . . .	32
26	Fluxograma Expressando a Construção do Porgrama do Frequêncimetro . . . . .	33
27	Valor Mostrado no Display Qando a Entrada é 535Hz . . . . .	34
28	Fluxograma Expressando a Construção do Porgrama do Controle Do Termômetro por Tabela . . . . .	34
29	Valor Mostrado No Display Quando o Micro é Programado Como sendo um Termômetro por Tabela. . . . .	35
30	Pretensão de Uso de Microcontroladores de 8 Bits. <b>Estudo Realizado no Ano de 2013 Pela Empresa Embeddded</b> . . . . .	39



## 1 Proteus

O software proteus, oferecido pela empresa Labcenter Electronics, não é gratuito, apesar de haver a versão demo disponível no site da fabricante. Possui diversos componentes tanto na área analógica quanto digital, estando muito deles disponíveis para simulação. Em suas bibliotecas há componentes ideais, que fazem seu princípio de funcionamento sem parâmetros de limitação, e componentes especificados por fabricantes, havendo também a possibilidade de se criar componentes, tanto em esquemático quanto em PCB (Printed Circuit Board) para fins de construção do esquemático produzido. Por enquanto o que se vê de diferencial é a presença de componentes digitais, e isso não se restringe a portas lógicas e flip-flops como encontrase presente em diversos softwares de simulação, mas também com conversores AD e DA de vários fabricantes e microcontroladores programáveis de diversos fabricantes, indo desde de ARM a outros menos robustos como a famosa família de controladores MCU8051. Display de sete segmentos e de cristal líquido também encontram-se presentes, permitindo a simulação conjunta dos componentes citados. Memórias também encontram-se disponíveis para simulação e esquemático podendo ser usadas em simulação conjunta com outros periféricos. Concluindo, fica a critério do projetista como utilizar os vários componentes presentes no simulador.

Um outro fato que leva a utilização do proteus, é a relação que existe entre suas duas interfaces ISIS e ARES, que permite a construção do projeto esquemático no ISIS posteriormente apenas transportando para o ARES com as ligações entre componentes já executada, o que é comum em softwares de PCB. O que diferencia o proteus é que o esquemático já possui sua simulação, assim o PCB só é executado quando o esquemático estiver funcionando, garantindo a primeira a construção do circuito. A figura (1) ilustra essa vantagem.

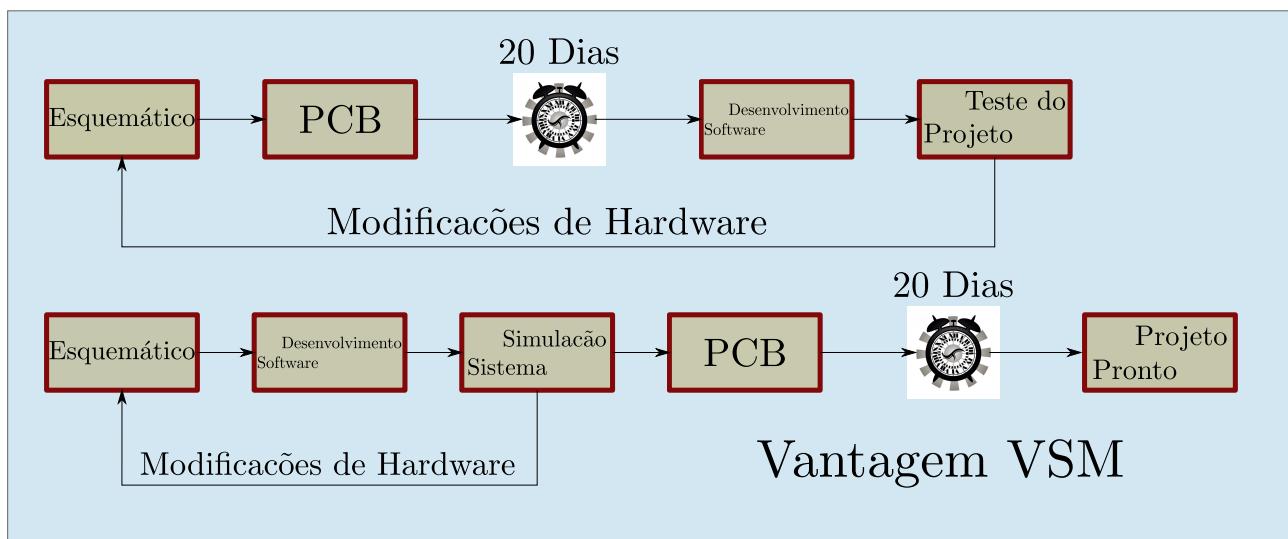


Figura 1: VSM Proteus Vantagem de Uso



## 1.1 Proteus Interface

Outro Link para o Help

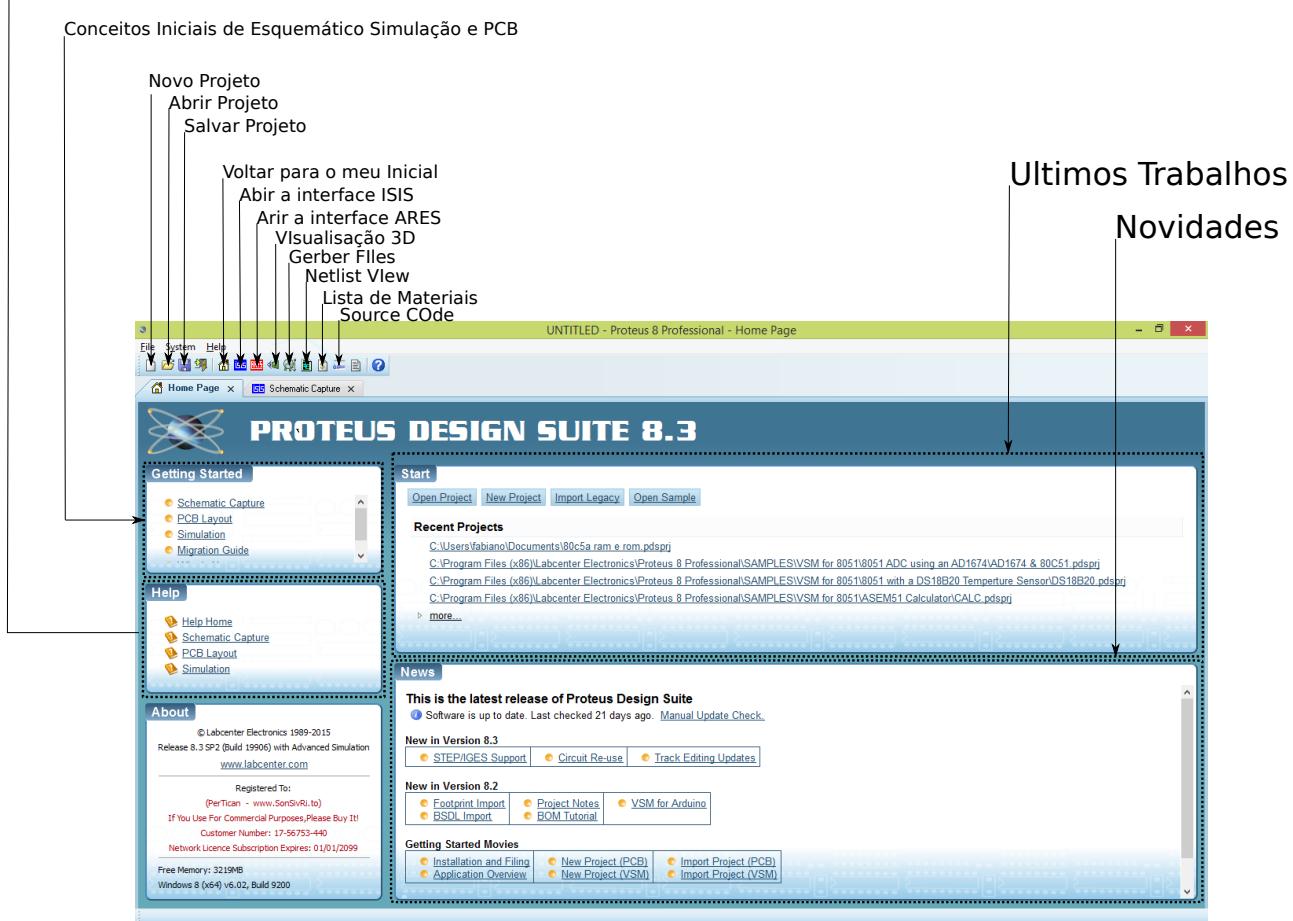
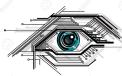


Figura 2: Tela Inicial Proteus

- **Outro link para o help** O help pode ser acessado por dois links, um pela janela indicada o outro pelo sinal de interrogação 2 itens ao lado de **Source Code**.
- **Conceitos iniciais de esquemático e simulação e PCB** Contem informações acerca dos itens citados, PCB, simulação e esquemático. Servindo como se fosse iniciais a serem exemplificado pelo software.
- **Novo Projeto** Inicia um novo projeto com opções de escolher o firmware se for desejado e outras configurações tais layout do papel de esquemático e a possibilidade do emprego de PCB's de arduinos se o desejo do projetista é criar shields para essa plataforma.
- **Abrir Projeto** Escolha de um projeto pronto em algum diretório.
- **Salvar Projeto** Salva o projeto que foi iniciado.
- **Voltar ao Menu Inicial** Volta para a interface expressa na figura(2).
- **Abrir a Interface ISIS** Abre a interface que possibilita a construção e simulação do esquemático.
- **Abrir a interface ARES** Abre a interface que possibilita a construção do PCB.
- **Visualização 3D** Visualiza o modelo real da placa renderizada no modo 3d.



- **Gerber Files** Gera os arquivos que após a construção do modelo PCB serão usados na confecção do produto gerado.
- **Netlist View** Verifica as ligações presentes no esquemático.
- **Lista de materiais** Após a confecção do esquemático pode ser gerado uma lista de materiais que descrevem o que foram usados no esquemático.
- **Source Code** Para simulações onde há a presença de microcontroladores pode-se, diretamente do Proteus colocar o código de funcionamento do microcontrolador.
- **Últimos Trabalhos** Lista dos últimos trabalhos acessados pelo programa.
- **Novidades** Novidades da versão corrente do programa em relação aos anteriores.



## 1.2 Proteus ISIS Interface

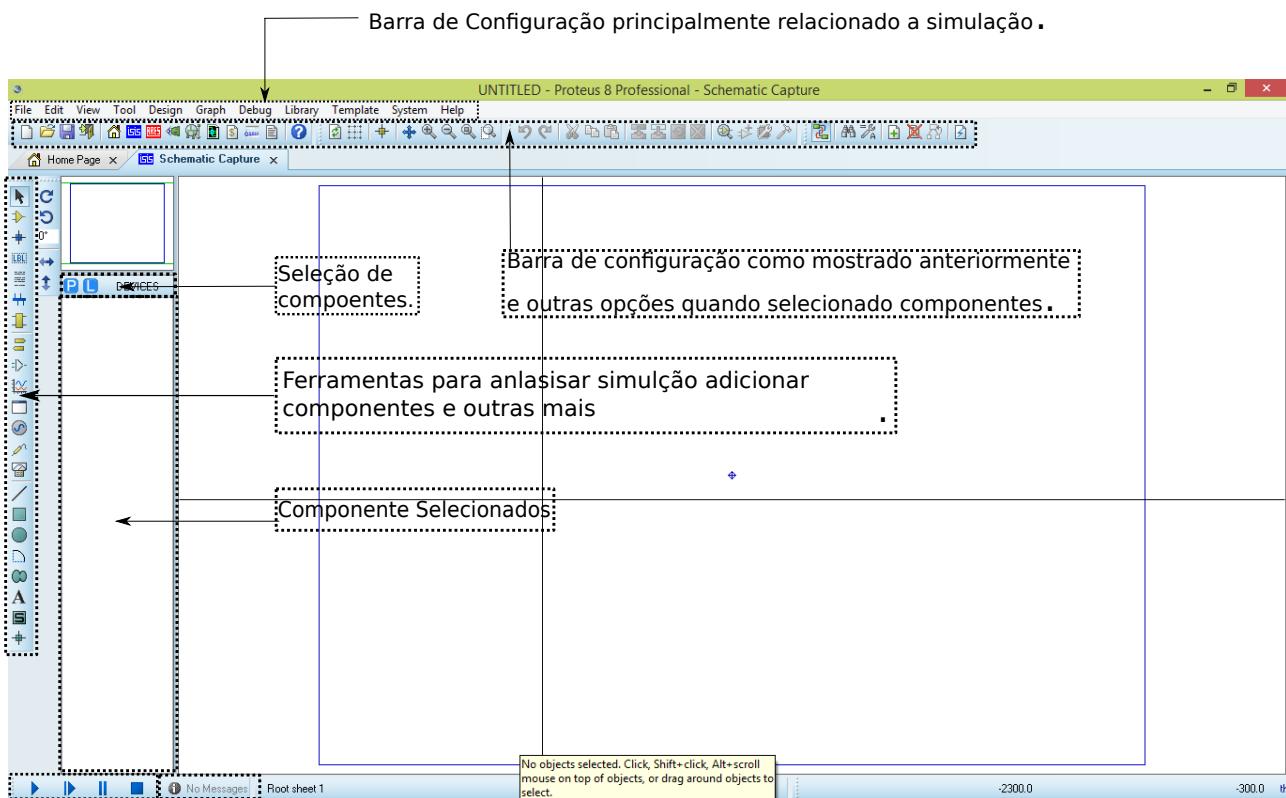


Figura 3: Tela Inicial Proteus ISIS

Explicando os conceitos relacionado as possíveis opções a se escolher na pagina inicial do proteus, resta algumas opções a se explicar quando se esta na plataforma ISIS.

- **Componentes Selecionados** Mostram os componentes que já foram adicionados a lista da onde podem ser retirados e colocados no esquemático.
- **Ferramentas para analisar simulação adicionar componentes dentre outras funcionalidades** Nesta seção há a possibilidade de adicionar ao esquemático ferramentas de verificação de simulação tanto no VSM como na simulação não animada. Cursor, seleção de componentes, centralização de imagem em um ponto, adição de bus para a criação de duto de dados, criação de componentes , seleção de ground e power 5v são a função das 8 primeiras possibilidades. Após estes símbolos encontra-se o de simulação não animada tendo diversas possibilidade de simulação analógica ou digital. Letras e formas de figuras também são adicionados por essa barra.
- **Seleção de Componentes** O P significa pick new component e o L Library são dois modo de se procurar componentes um por bibliotecas e outras escolhendo pelo nome e ir filtrando pelas possibilidades de bibliotecas e característica do componente a ser escolhido.
- **Barra de configuração dispensa comentários devido a imagem anterior**



## 2 Simulação 80c51 Assembler

As simulações presentes neste documento baseam-se nas disciplinas Aplicação de Microprocessadores 1 e 2 sendo que o mesmo inicia-se em programação utilizando a linguagem assembler e a subsequente simulação no sistema via ISIS de todos os projetos. Assim o código será descrito via fluxograma e subsequente esquemático, com os resultados, ou somente esquemático que simula o programa. Na próxima seção todos os programas serão simulados em C.

### 2.1 Práticas Botões e Leds

As práticas consistem em simular o comportamento de leds com relação a chaves ligadas na porta P3, nos respectivos pinos P3.5, P3.6 e P3.7, ligando leds de modos diferentes conforme o botão pressionado, estando os leds na porta P1, respectivamente P1.1, P1.2 e P1.3. Isto é indicado na figura (4) sendo o esquemático do 8051 junto aos botões e leds.

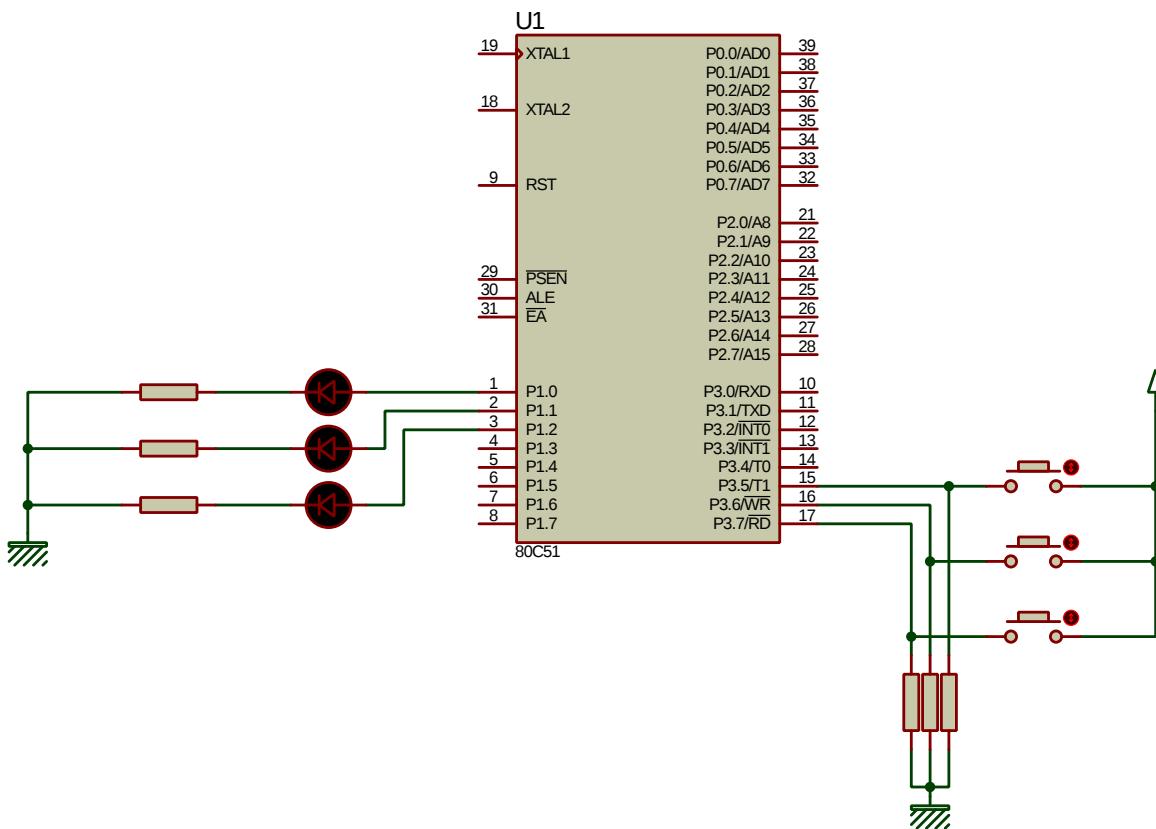
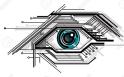


Figura 4: Esquemático de Simulação Exercício Botões e Leds



O fluxograma que representa a programação do problema proposto encontra-se na figura (5) abaixo.

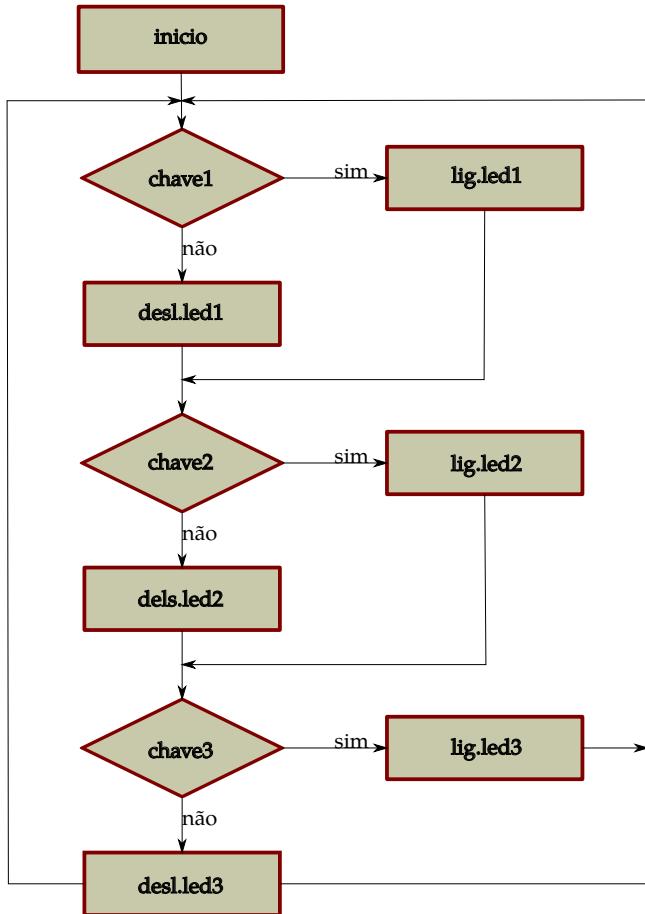


Figura 5: Fluxograma do exercício 1

O problema tem três possibilidades de elaboração, uma é essa expressa na figura (5) outra delas é a listada acima no fluoxograma que acende o led correspondente conforme a chave ligada. Ainda é proposta para essa configuração como exercício os itens presentes abaixo.

- Acender piscar cada led na frequência de 1 Hz para cada chave ligada correspondete ao mesmo.
- Acender um led com a chave1, outro led com a chave2 e piscar o led1 e led3 com a chave3 ligada, na frequência correspondente de 1 Hz cada led. **Atenção** Ligar os leds somente se a chave3 não estiver fechada.



Abaixo na figura (6) encontra-se a simulação do exercício proposto no segundo item mostrando que piscam os leds na frequência de 1 Hz cada quando a chave3 esta pressionada.

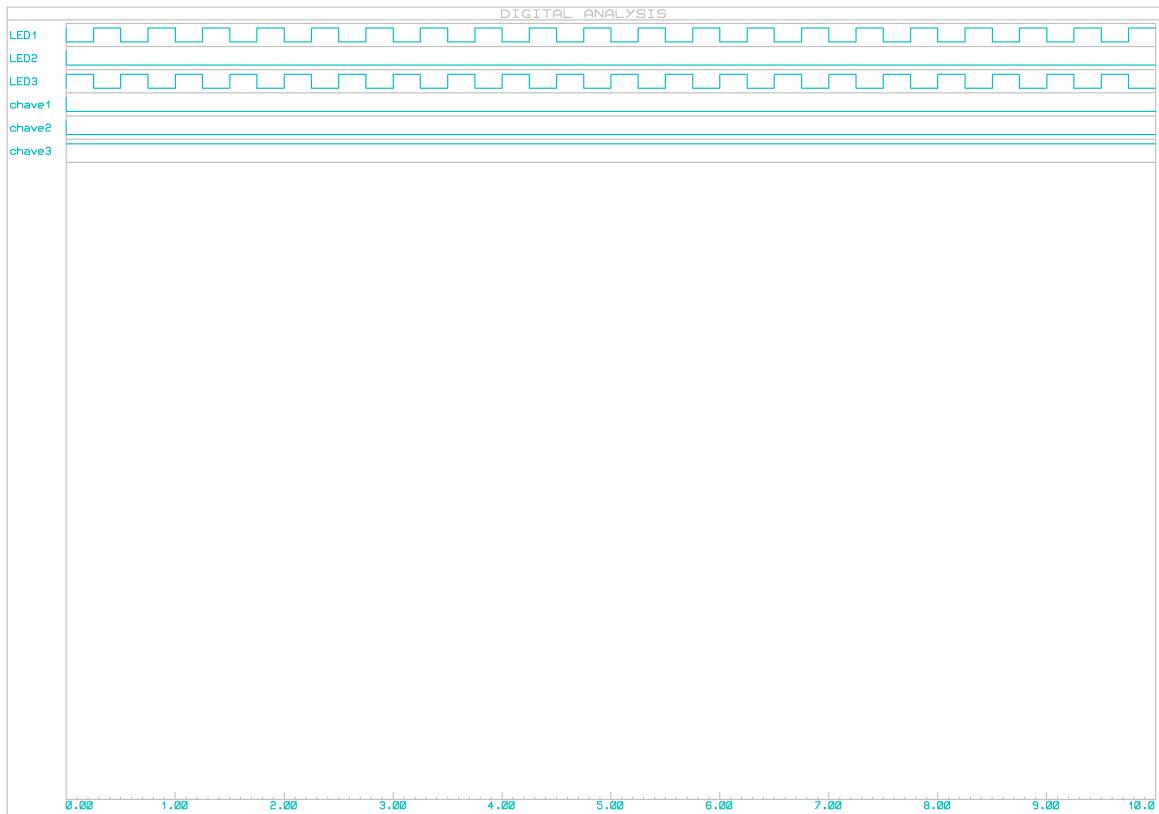


Figura 6: Simulação Exercício 2



## 2.2 Simulação Memória Ram e Rom Externas

As memórias ram e rom externas podem fazer-se necessárias em projetos mais robustos, aonde exige-se mais armazenamento de dados do que a familia de microcontroladores oferece, que no caso, da 80c5x varia de 128 bytes de ram a 512 bytes e rom de 4 Kbytes a 32 Kbytes. Fazendo a expansão é possível endereçar até 64kbytes de ram externa e rom, e com outros modos, é possível também aumentar ainda mais essa possibilidade utilizando um shift-register como selecionador de memórias. Este ultimo modo não é coberto neste trabalho.

Para acessar os conteudos das memórias ram, em assembler no caso, considerando o compilador **MCU IDE 8051** é preciso usar os comando MOVX A,@A+DPTR para acessar um valor externo ao microcontrolador que está endereçado no valor incluso no DPTR, e para colocar um valor no endereço apontado pelo DPTR é usado o comando MOVX @DPTR,A. A memória rom também tem comandos simples, bastando usar no lugar de MOVX o MOVC. Lembrando que faz-se somente leitura da rom e usa-se somente o comando MOVC A,@A+DPTR. O pino EA do micro deve estar ativo para acessar a memória rom externa, devendo haver atenção com este aspecto de esquemático.

Assim, sendo simples os comandos, é importante focar no hardware que as operações exigem, afinal o funcionamento depende da montagem peculiar do esquemático, iniciando uma ideia bastante importante do 80c51 que é a utilização do duto de dados presente na porta P0 como demultiplexada entre endereços e dados. O duto de endereços se estende até a porta P2, totalizando 16 bits de endereçamento o que possibilita uma expansão de até 64k de memórias.

Se caso não for usado todos os bits de endereçamento, os restante são usados como seleção fazendo uma regra em que cada endereço alocado selecione um componente, por um chip chamado decodificador. O esquemático pode ser visto na figura (7) com o microcontrolador uma memória ram de 8k e uma rom de 8k.

É importante salientar o uso dos probes na simulação, eles permitem verificar o valor de um dado pino em um dado instante, havendo a possibilidade de simulações digitais, ou mixas, o que será visto mais adiante. Há a possibilidade de se verificar qual o conteúdo de um duto, apenas utilizando os probes e o bus-wire, crucial na montagem de esquemas afinal este diminui bastante a desorganização do esquemático, porém torna mais minuciosa a construção, afinal cada ligação precisa estar corretamente nomeada.

Foi feito um programa que varre o DPTR de 0000h a ffffh e coloca um valor de 00h a ffh na memória ram. Além disso verifica os pinos de seleção, que são ativos em zero. Como a memória é de 8K, há a necessidade de 13 pinos do microcontrolador para conectá-las, sobrando, ao fim, 3 pinos para seleção, o que torna possível a utilização de um decodificador para endereçar 8 elementos. O resultado da simulação encontram-se nas figuras (8) e (9) uma mostrando a simulação por completo e a outra especificando, em um dado momento, o dado passando pelo duto de dados e de endereços.

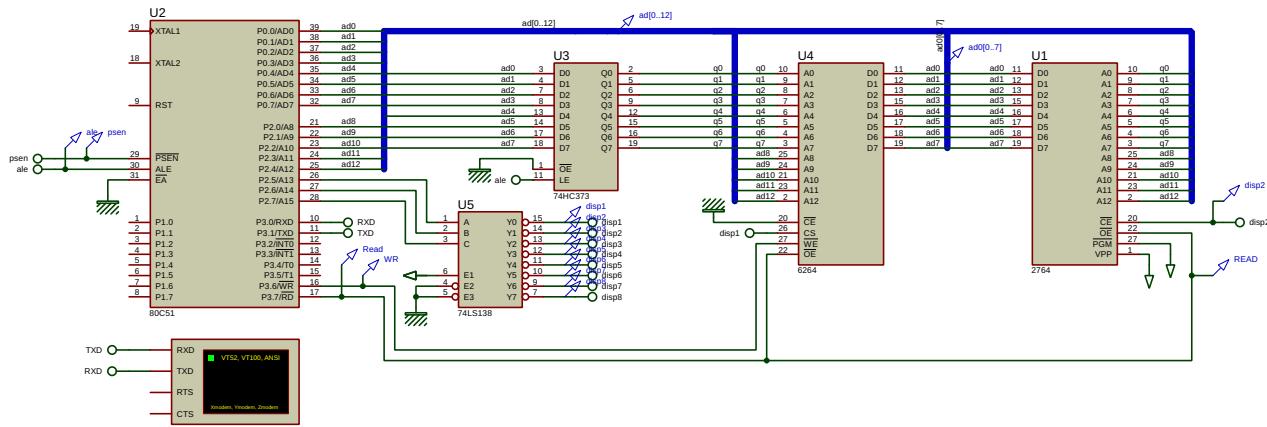


Figura 7: Hardware Ram e Rom de 8Kbytes

Verifica-se, na simulação acima, figura (9) que no valor 080Ch do DPTR é colocado o valor DCh e assim sucessivamente, incrementando o DPTR e o valor a ser colocado na memória. Verifica-se também que o dispositivo selecionado no momento da operação é o dispositivo 7, afinal seu valor vai para zero quando colocado o valor no duto de dados, enquanto todos os demais ficam em nível alto no momento da alocação de dados.

Essa maneira de pensar , selecionando objetos e trabalhando somente com o selecionado, como já foi dito, é muito importante para o 80c51.

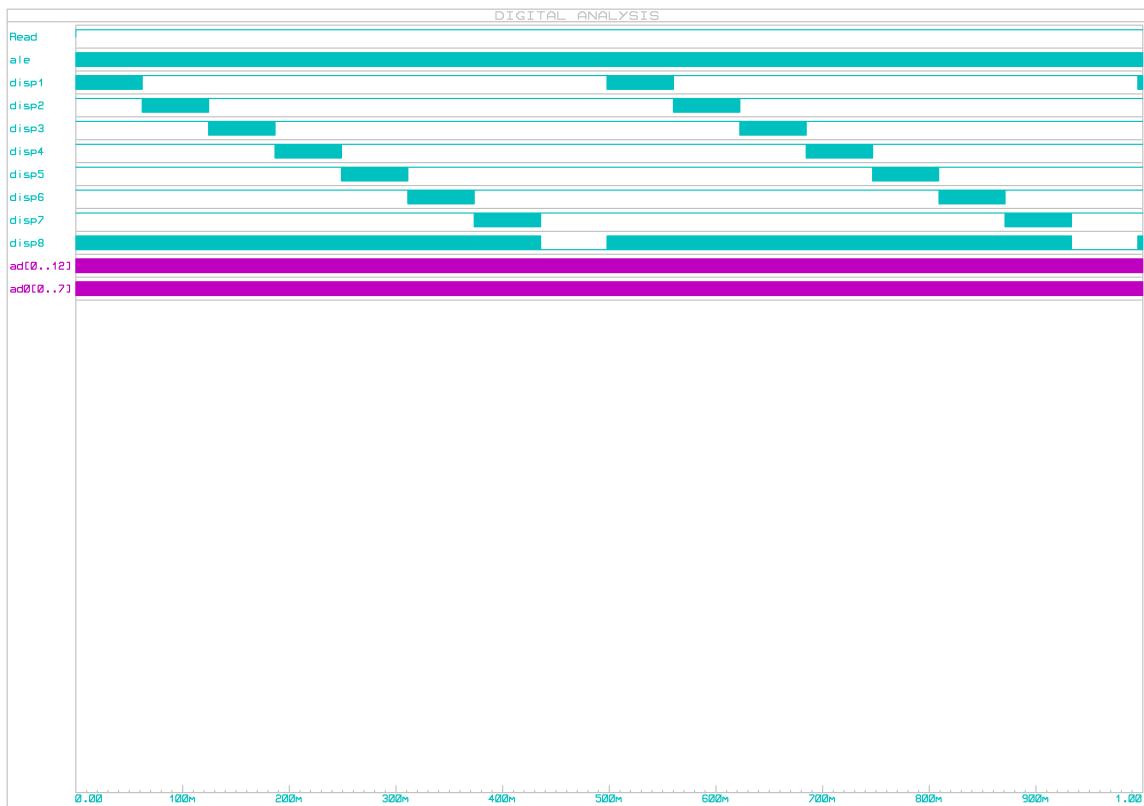


Figura 8: Varrendo Todos os Valores do DPTR

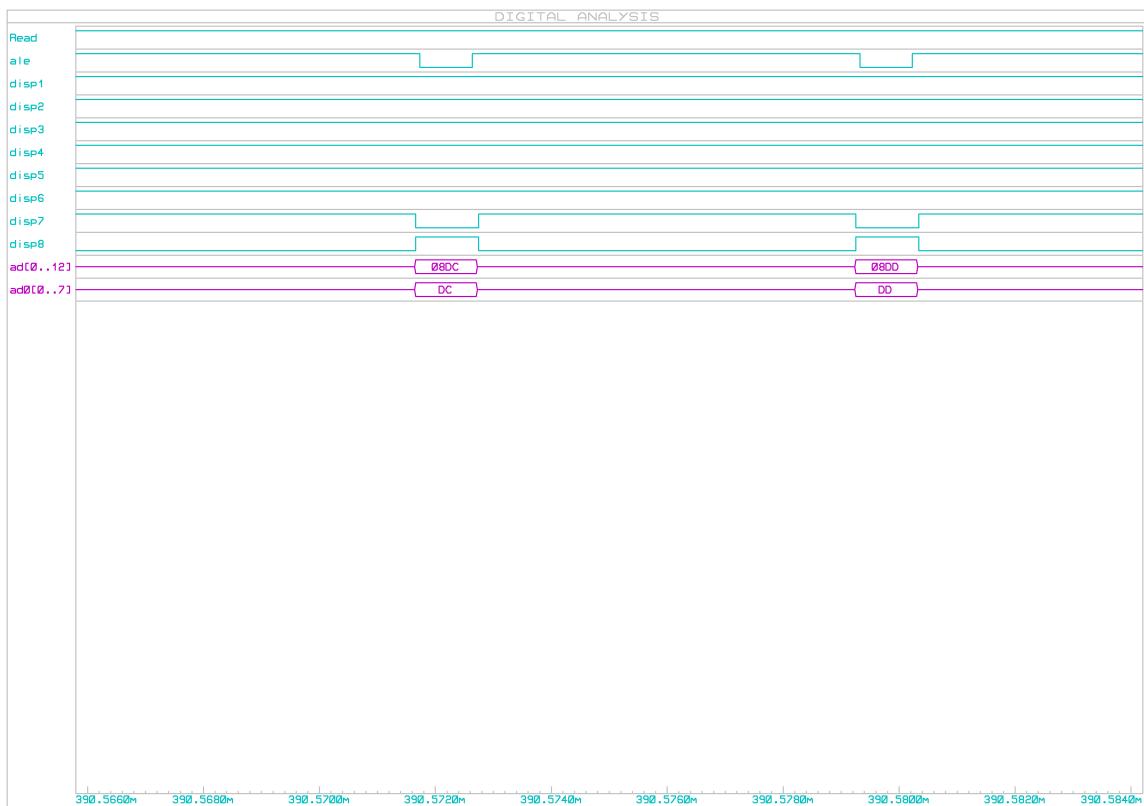


Figura 9: Selecionando Valores Vizualizaveis do DPTR



## 2.3 Comunicação Serial

Pode-se fazer comunicação serial de dados, o que significa enviar um bit por vez, partindo do emissor para o receptor, sendo que desses bits alguns sinalizam o caractere e outros que significam informações de parada, inicio de comunicação dentre outros conforme a configuração da comunicação.

Neste documento foram realizadas as comunicações seriais de 8 bits de caractere e um de inicio, que pode ser configurado no 80c51. Interessante ressaltar que há possibilidade de uma comunicação serial com o 80c51 se da devido ao mesmo possuir uma UART como hardware em sua arquitetura.

A taxa BAUD-RATE de comunicação serial, sendo a quantidade de bit enviada por segundo pelo terminal de comunicação, foi taxada como sendo de 9600( $\frac{\text{bits}}{\text{s}}$ ) porém pode-se configurar o microcontrolador para realizar comunicações com outros BAUD-RATE, lembrando que no Proteus a configuração deve ser feita tanto em software quanto em hardware, modificando no terminal serial, mostrado na figura (10) abaixo.

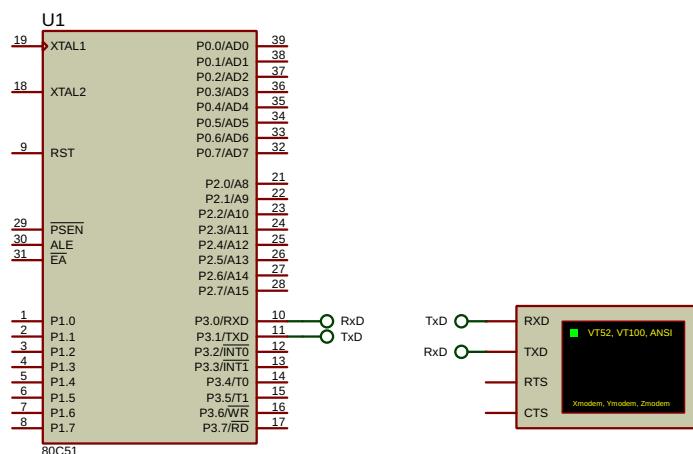


Figura 10: Esquema de Hardware Para a Simulação da Comunicação Serial



Há as configurações de BAUD-RATE variando conforme protocolos de comunicação sendo dois bastante conhecidos de comunicação RS-232 e o RS-485. Também pode-se configurar o microcontrolador para realização comunicação com diferentes quantidade de bits. Foram fixados 10 bits, e taxa de BAUD-RATE de 9600( $\frac{Bits}{s}$ ). A figura (11) ilustra como se dá o comportamento dos bits da comunicação quando é enviado para o terminal serial. O caracter enviado é o 'F'.



Figura 11: Bits que Representam a Comunicação Serial do Caracter F

Como possibilidades de simulação ficam a critérios de escolha de configuração de outros valores de BAUD-RATE, e o item listado abaixo:

- Fazer um programa que envia uma frase dependendo do caracter enviado, ainda contando com um caracter que imprima todas as frases armazenadas.

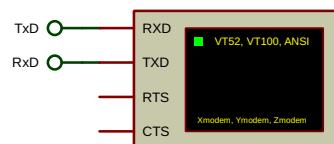
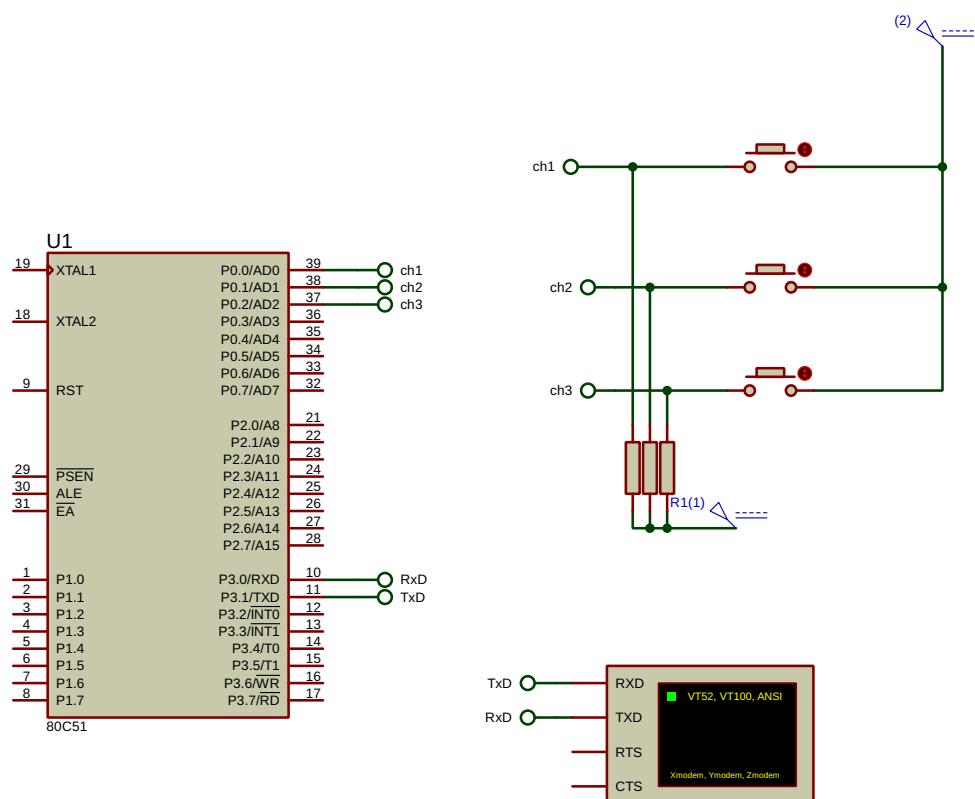


Figura 12: Hardware Que Possibilita a Simulação de Envio de Frase Dependente de Botão.



Abaixo na figura (13) encontra-se o fluxograma que ilustra o envio continuo de um caracter.O envio de uma frase exige a utilização de um ponteiro que aponte para a frase fazendo com que o conteúdo do ponteiro seja passada para o registrador SBUF. Assim, além de verificar se a transmissão de cada caracter é preciso identificar mais um if(cjne) verificando o fim do envio da frase.Para fazer as simulações com a interrupção bastam configurá-la, lenbrando que é preciso habilitar a recepção, por meio do bit REN.

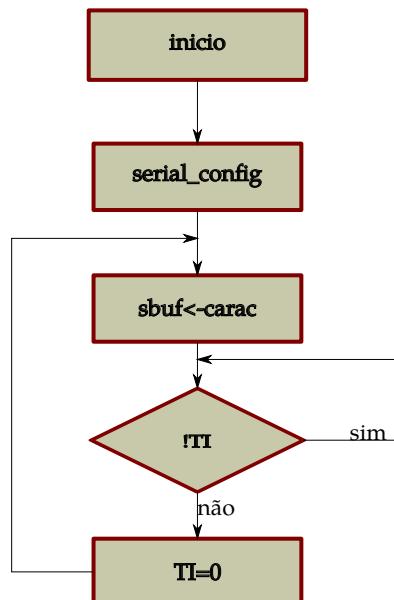


Figura 13: Fluxograma do Recebimento de um Caracter sem Interrupção



## 2.4 Timers, Interrupções e Contadores : Projeto Conversor AD

Dentre os elementos que constituem o microcontrolador 80c51 há dois timers de 16 bits cada e a presença de 5 tipos de interrupção, sendo uma delas extra. Sempre que utilizá-las é preciso que se faça sua configuração e escolha conforme o trabalho a ser executado.

Os timers presentes são **timer1** e o **timer0** ambos de 16 bits configurados pelos registrados DPHx e DPLx sendo x 1 ou 0, correspondendo ao timer em questão.

As interrupções estão relacionadas aos hardwares com as quais funcionam, no caso de recepção e transmissão serial, ao bit de finalização da transmissão recepção da serial. As externas relacionam-se com os pinos INT0 e INT1 e o overflow dos timers que se relacionam com bit TF, indicando estouro de contagem.

Como possibilidades de projeto pode-se fazer a geração de formas de ondas quadradas com qualquer dos timers utilizando um esquemático que seja apenas o 80c51 ou contar quanto tempo uma onda fica em nível alto, utilizando o timer como temporizador, para isso basta que se faça a configuração do microcontrolador e adicione probes no esquemático para visualizá-la. Pode-se utilizar o osciloscópio disponível na barra de ferramentas, caso opte-se pela simulação animada.

Outra possibilidade de projeto é fazer um medidor de tempo em nível alto de uma onda e imprimir na serial, lembrando que será preciso fazer a conversão hexa asc do valor presente no contador. Além disso há o fato de que os timers são de dezesseis bits, o acumulador e registradores são de oito bits, devendo retirar os valores dos registradores DPhx e DPLx.

O projeto do conversor AD não irá dar enfase em critérios de conversão, como precisão e se os tempos de amostragem estão satisfazendo os critérios de desempenho da aplicação em questão, mas sim a utilização de uma maneira bastante prática o uso de interrupções, utilizando a interrupção de qualquer dos timers além de uma externa.

### 2.4.1 Conversores e Breve Conversão AD e DA

Conversão analógica digital envolve vários conceitos, como tempo de amostragem, resolução, tempo de conversão para limites de frequência que podem ser convertidos. Isto será deixado de lado nesta análise, sendo o objetivo somente fazer funcionar o conversor AD, pegar o valor convertido e colocar em outra porta do microcontrolador aonde está conectado um conversor DA, com a qual será feita comparação entre os sinais de entrada e de saída.

O esquemático que possibilita esta simulação encontra-se na figura (??) que trata-se do microcontrolador 80c51 junto a um conversor ADC0808 e DAC0808. Utilizando geradores de onda do proteus para simular os cloks e pulso de start que encontra-se no esquemático citado.

O esquemático que simula o conversor AD junto ao microcontrolador, fazendo uso de uma fonte, um node de geração de pulsos, acessado na barra de ferramenta a esquerda no ISIS, cujos valores são 680k para o clok e 5k para o start é o mesmo citado anteriormente. A programação do micro é apenas colocar na porta P2 o valor convertido que encontra-se na porta P1, enfim o código faz P2=P1. Em P1 encontra-se um conversor AD, que converter para analógico o sinal digitalizado. Na figura (14) encontra-se a simulação da conversão ad realizada.



## 2.5 Conversor AD e DA Controlado Pelo 80c51

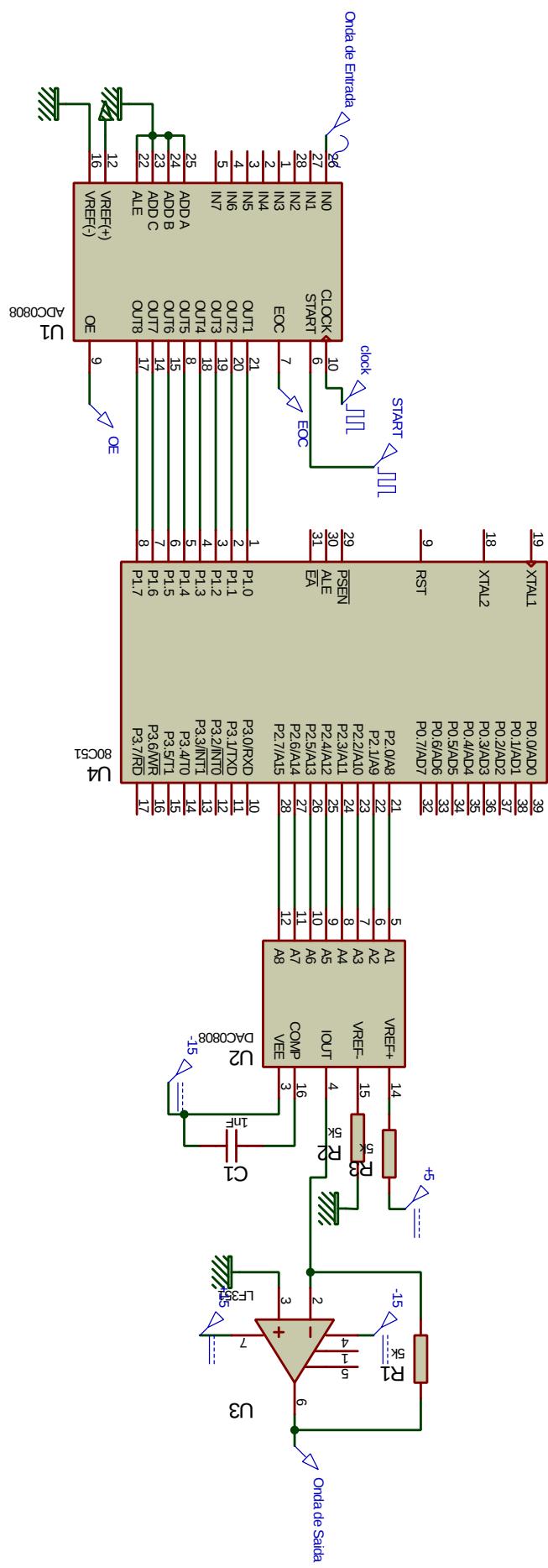
É preciso que haja a leitura do datasheet do conversor ADC0808 e DAC0808 para verificar os valor que deve ser inserido como clock que não pode ser alcançado pelo micorontrolador 80c51 levando a necessidade de se trablahar em uma frequêcia mais baixa,além de carregar o timer no máxino ffefh, para que haja interrupção no seu estouro.Ao fim da conversão é gerado um pulso em nível alto no pino EOC do conversor AD. O conversor DA apenas converte para analógico o sinal digital, assim sua única limitação é quanto ao tempo de conversão, oque, para testes não significara muito caso sejam convertidos sinais de baixa frequêcia.

Assim, tem-se como objetivo gerar uma onda quadrada por meio da interrupção de um dos timers e configurar alguma interrupção externa sensível a borda de subida do EOC, e dentro da interrupção ler o valor convertido.O fluxograma da figura (15) expressa o controle do conversor AD.

Verifica-se pelo datasheet do ADC0808 que o mesmo trabalha com clock tipico de 680kHz.Diz também que o pulso de start tem que ter duração maior que 200ns e que o tempo de conversão será no valor de 90 a 100us.Dessa forma há a necessidade de se trabalhar com valores menores de clock e aumentar o pulso de start para controlá-lo por meio do 80c51 afinal nem carregando o timerx com ffefh não é um tempo minimo de overflow que gera 680kHz.

Na simulação feita foi carregado timer1 com o máximo ffefH, inicia o timer1, espera 10us e da um pulso de start após isto espera o EOC ir para 1, acionando a interrupção.Na interrupção tirar o valor da P1 e passar para P2.Após isto reinicia-se o ciclo.

Na figura (??) encontra-se o hardware que possibilita esta simulação.Deste modo tem-se o fluxograma e o hardware mínimo para simulação. A figura (16) expressa o resultado da conversor de uma forma de onda construida a mão pelo gerado pwlin.Além de desenhar a forma de onda é possível programa-la por meio da linguagem HDL.



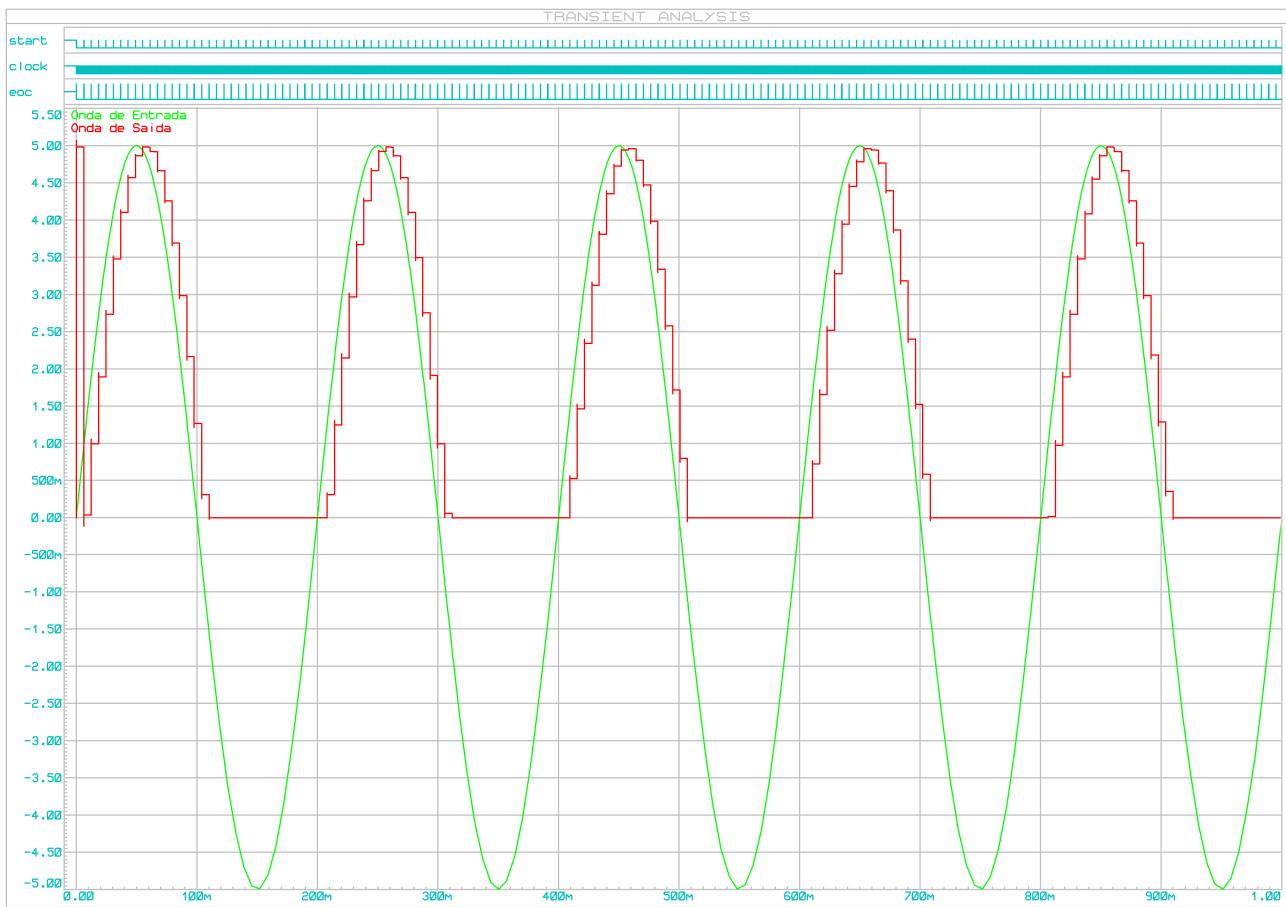
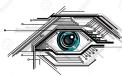


Figura 14: Simulação Conversores AD DA e 80c51

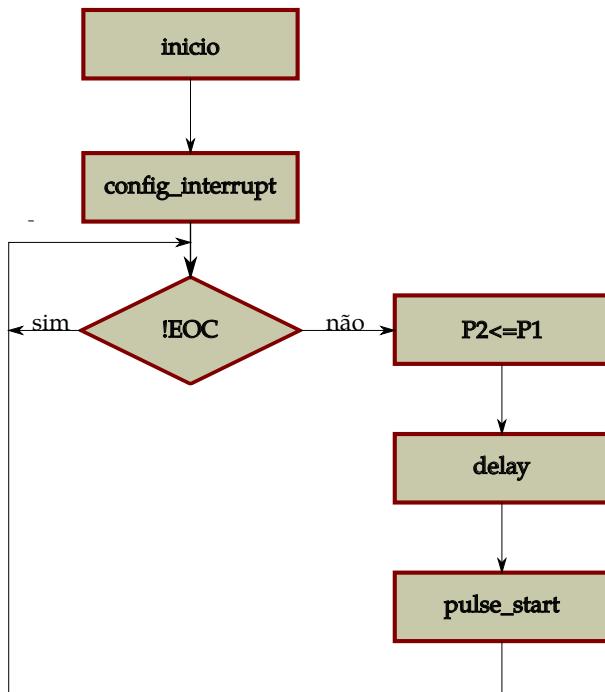
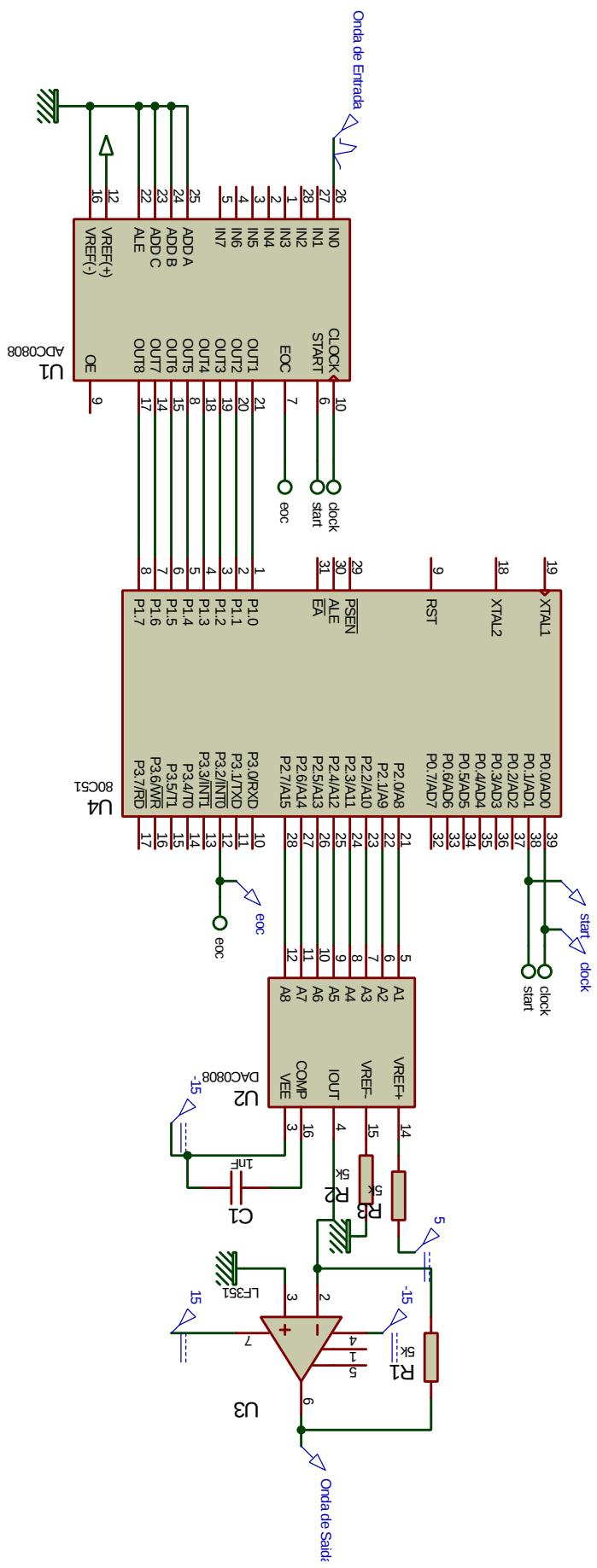


Figura 15: Simulação Conversores AD DA e 80c51



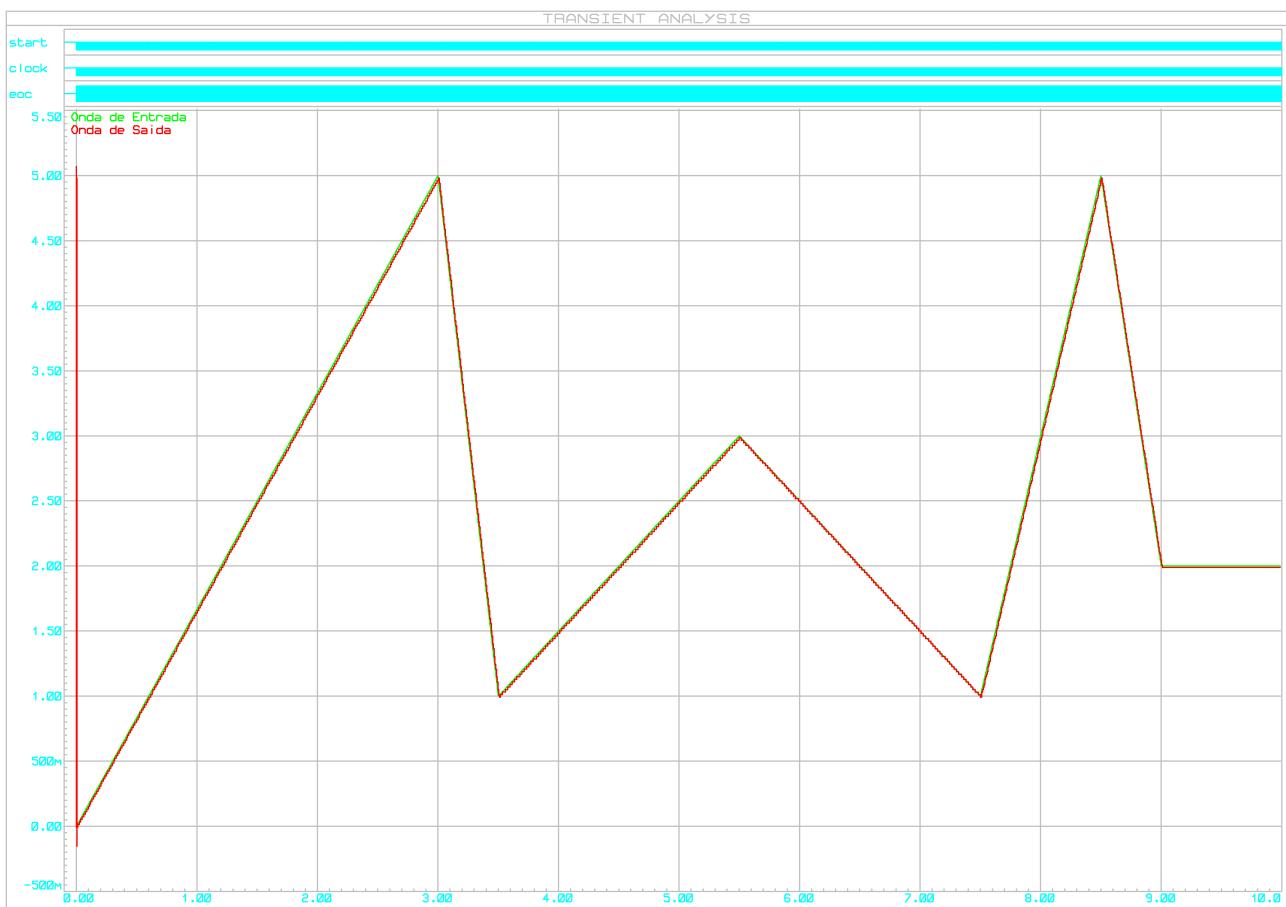
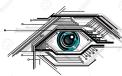


Figura 16: Simulação Conversores AD DA e 80c51

## 2.6 Motor de Passo

Uma possível aplicação de controle do 80c51 é controlando o motor de passo. Um motor de passo é um conversor eletromecânico que da uma rotação, cujo sentido e velocidade de rotação, dentro de certo limites, são configuráveis. Como o motor é um dispositivo que exige correntes elevadas com relação as que as portas do 80c51 podem oferecer, é preciso mediá-lo por meio de um driver, sendo os chips utilizados para se construir o driver o L297 como controlador e o l298 como dispositivo de potência. Uma leitura nos datasheet desses componentes é importante para a realização desta prática.

A figura (17) expõe o hardware completo com o driver do motor, lembrando que para encontrar o motor basta entrar na seleção de componentes e digitar motor, que surgira as opções, após isto basta escolher o stepper motor.

A simulação consiste em fazer o motor girar para o sentido horário e esperar até que uma interrupção externa ocorra, como isto não é possível de simular no ISIS colocou-se o botão junto a INT0, quando pressionado o botão a interrupção é atendida e o motor espera por 5s, e gira novamente no sentido anti-horário até haver novamente uma interrupção no mesmo pino. Após isto o motor para por dez segundos e reinicia o ciclo.

Esta implementação pode ser a automação de um a esteira que para por tempos diferentes quando um objeto, ou produto, passa por um determinado sensor que envia um sinal para o microcontrolador. As paradas da esteira podem ser entendidas como tempos em que o produto esteja sendo modificado.

Abaixo, na figura (18), encontra-se o fluxograma que representa o código do controle do motor de passo. É preciso conhecer os drivers para executar a simulação, então é dito novamente que a leitura dos datasheets é indispensável.

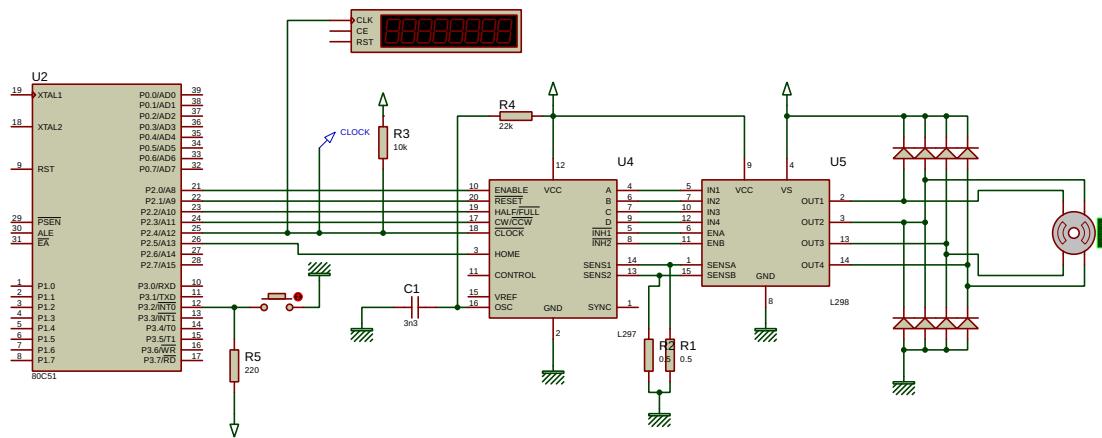


Figura 17: Hardware Utilizado Para a Simulação do Motor de Passo

Como é necessário gerar uma onda de frequência aproximada de 10Hz para fazer com que o motor de passo se movimente, ligando o micro no driver, nos pinos corretos, é preciso usar a interrupção de um dos timers para realizar esta tarefa.

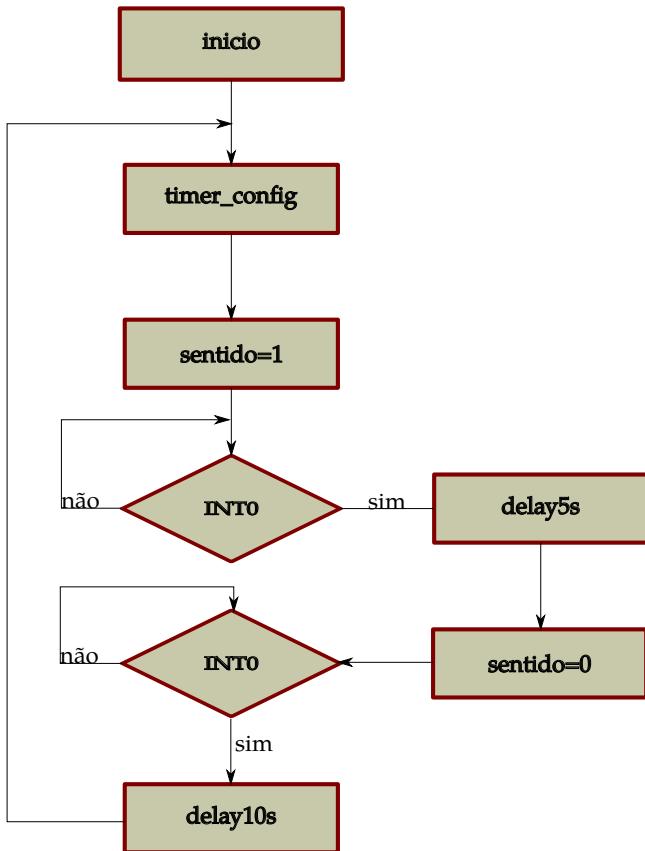


Figura 18: Fluxograma Motor de Passo

## 2.7 Display de Sete Segmentos e LCD

Há a disponibilidade de displays de sete segmentos para simulação animada no ISIS. Os displays devem ser multiplexados e o valor a se colocar no display deve corresponder a conversão hexa-decimal do valor em hexa em questão. Pode-se também, simplesmente adicionar um conversor BCD a saída do microcontrolador e ligá-lo por sua vez aos displays. Assim basta somente multiplexá-los e colocar o valor desejado na porta aonde encontra-se o conversor. É também preciso esperar um tempo para o display ficar ligado com o respectivo valor. Na prática o ser humano enxerga até no máximo 30Hz, havendo um atraso que não ultrapasse essa frequência de acendimento de cada display, não há problemas em configurar o delay.

Abaixo na figura (19) encontra-se o fluxograma que traduz o código a ser usado para expressar um valor constante a quatro display de sete segmentos.

Abaixo na figura (20) encontra-se a montagem do esquemático necessário para a utilização do display de sete segmentos, considerando o uso de um conversor BCD.

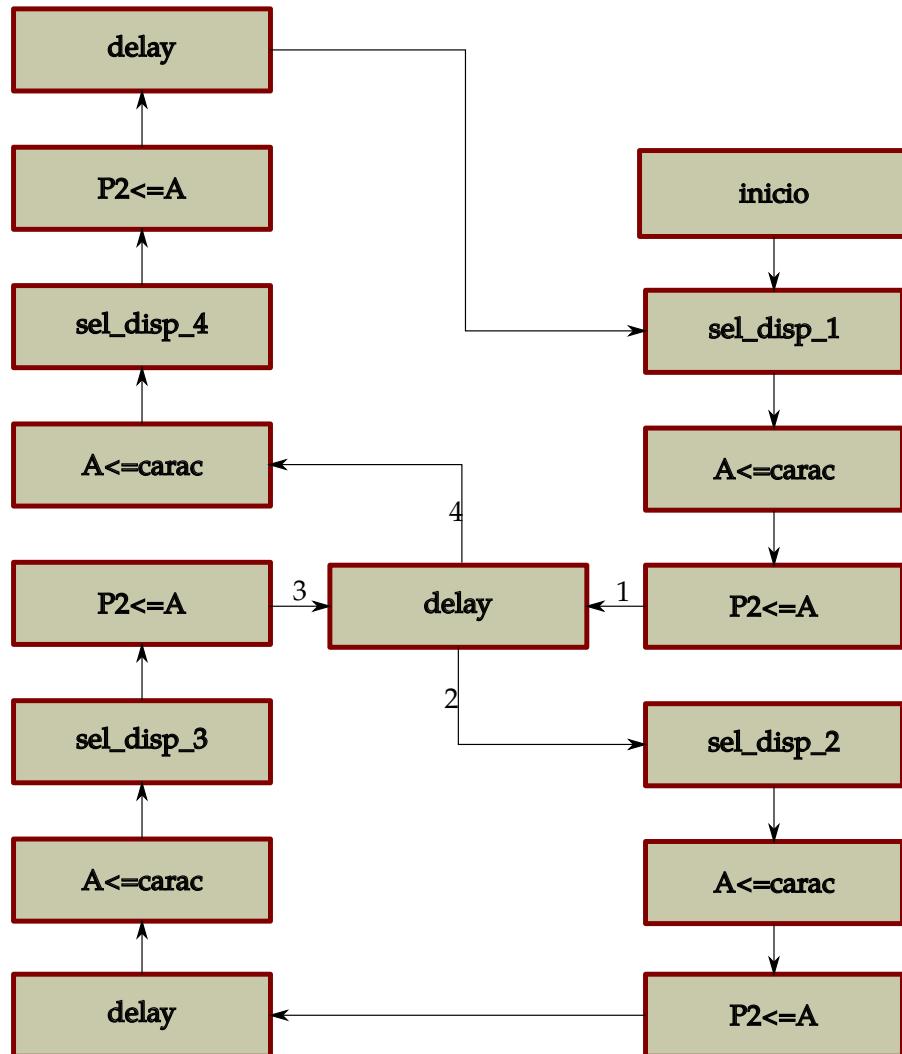


Figura 19: Fluxograma Display de Sete Segmentos

Para que os displays apareçam com seus caracteres rapidamente é preciso mudar as configurações do Set Animations Options do ISIS na barra de configurações opção System. Faça com que os frames da simulação apareçam mais rápido, modifique as configurações e veja o resultado.

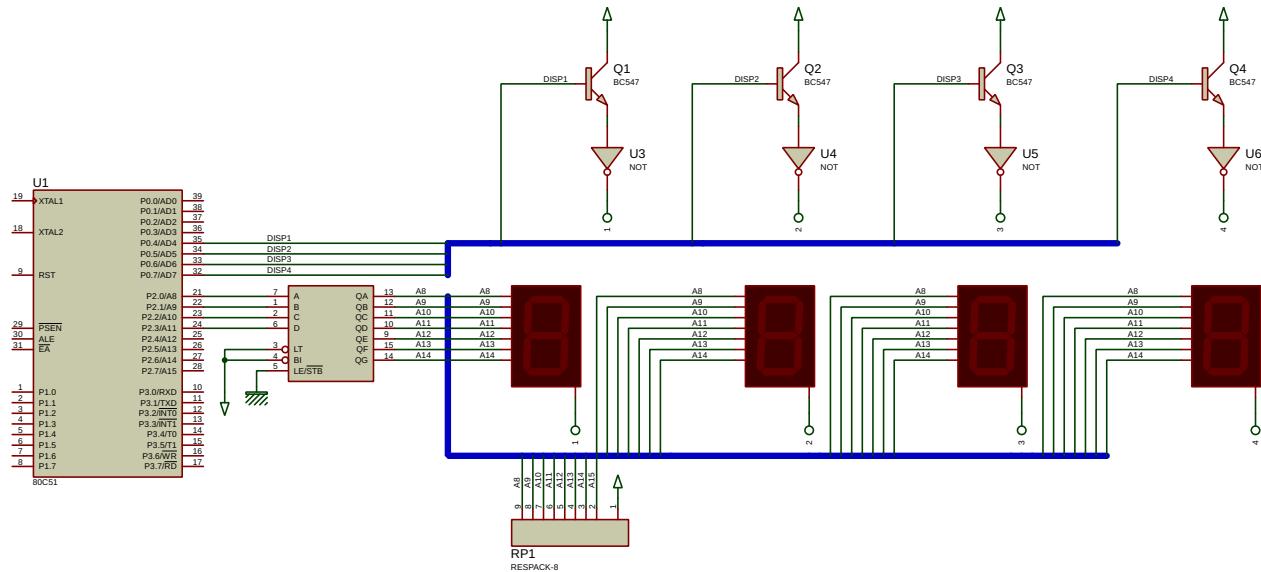


Figura 20: Esquemático para Simulação dos Displays de Sete Segmentos

### 2.7.1 Display LCD

Liquid Crystal Display são largamente utilizados e fornecem um meio de interface visual com o usuário. Não raro encontra-se projetos com lcd, possiblizando o fornecimento de informações que estão sendo processadas para o usuário utilizá-la.

O uso do display exige que seja escolhida sua configuração quanto a quantos pinos serão utilizados, assim, pode-se operar tanto no modo 4 bits de dados e instruções quanto 8 bits de barramento. Há ainda os bits de controle E-Enable, RW-ReadWrite e RS que indica a leitura de dados ou instruções. Há disponibilidade de vários display no ISIS, e o mais comum LM016 de duas linhas, também está na biblioteca.

É preciso ter em mente primeiramente como será a configuração do display, duas, uma linha, pixels da matriz, e enviar os dados de configuração ao display. É importante criar uma partição que envie instruções e outra que envie caracteres ao lcd, afinal há uma diferença nos bits de controle em nível alto e baixo para escrever cada uma dessas informações.

A figura (21) indica o fluxograma que trata como inicializar o display e escrever uma frase.

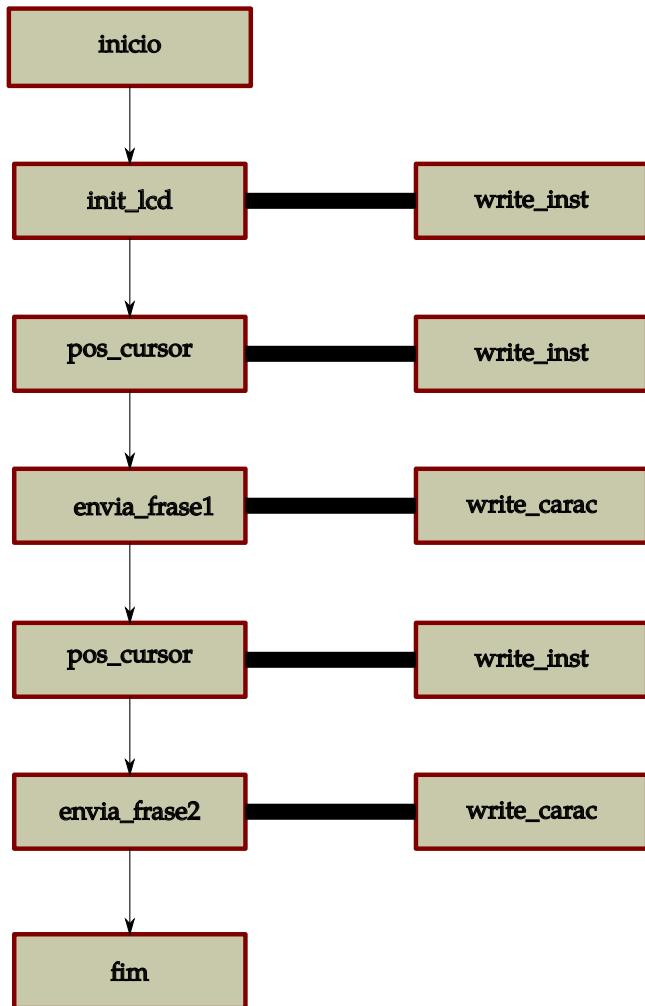


Figura 21: Fluxograma Expressando o Funcionamento do Display LCD Tanto em 8 Como em Quatro Bits

As duas proximas figuras (22) (23) ilustram o hardware já com o display em funcionamento tanto no modo 4 bits como em 8 bits.

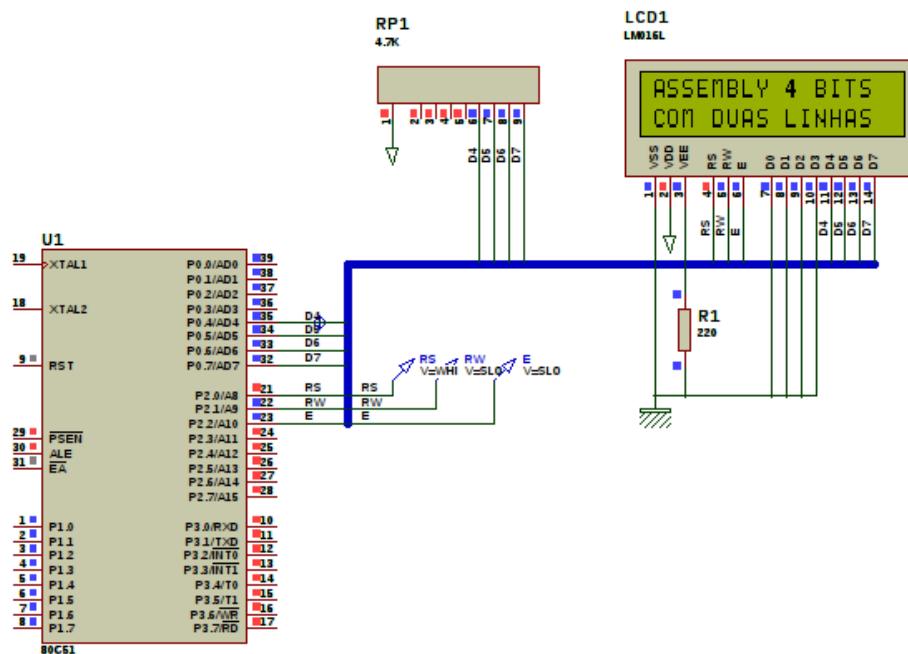


Figura 22: Esquemático Expressando o Funcionamento do 80c51 Junto ao LCD na Configuração 4 bits

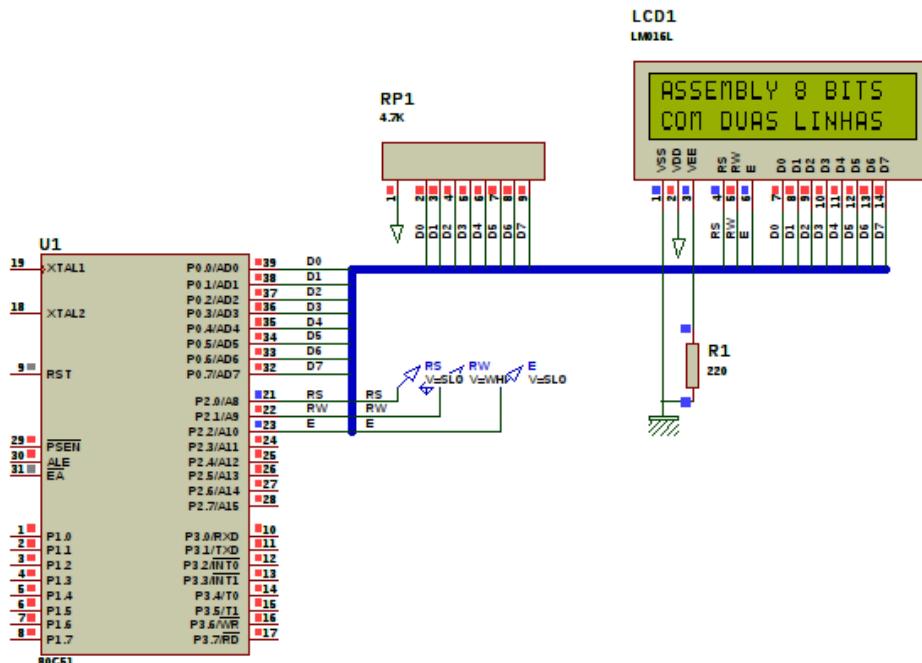


Figura 23: Esquemático Expressando o Funcionamento do 80c51 Junto ao LCD na Configuração 8 bits



### 3 Projetos 80c51 em Linguagem C

Pode-se trabalhar com o microcontrolador 80c51 em linguagem C, utilizando para isto um compilador para a linguagem. Junto ao Proteus ISIS há uma janela que se chama source code, nela é possível selecionar um compilador previamente instalado no computador e programar o micro direto do proteus, não exigindo mais que o código deva ser carregado no micro, bastando apenas que se compile o programa e de play na execução.

A escolha do compilador pode ser feita antes ou depois de se construir o esquemático, podendo clicar com o botão esquerdo em cima do micro, ir em Edit Firmware e escolher o compilador que irá usar, após isto basta programar na janela que irá abrir.

O compilador escolhido para a programação em C é o **SDCC**, que compila programas de alguns microcontroladores da família MCS80c51, incluindo o utilizado em todos os projetos até o momento.

Para uma breve introdução em linguagem C para o 80c51, reescrever todos os programas em assembler ditados até o momento é uma boa maneira de se iniciar na linguagem aplicada ao micro. Os projetos aqui apresentados levam em consideração que isto tenha sido feito, apresentando resultados mais rebuscados com o microcontrolador, como um frequêncimetro, um termômetro e um modo diferente de controle do motor de passo, utilizando a interface serial. Outros projetos também encontram-se descritos.

#### 3.1 Motor de Passo Controlado Pela Interface Serial

O hardware utilizado para a simulação do motor de passo é o mesmo apresentado na figura (17) junto aos pinos TX e RX ligados ao terminal serial virtual, como apresentado na figura (10).

A simulação consiste em exibir um menu no terminal serial que possibilite ao usuário entrar com caracteres que irão identificar um possível funcionamento do motor de passo. As opções escolhidas estão listadas na figura (24) que mostra o terminal com o menu escrito, esperando para receber um caractere e iniciar o funcionamento.

```
Virtual Terminal
1 PASSO SENTIDO HORARIO CLIQUE      A
1 PASSO SENTIDO ANTI-HORARIO CLIQUE   B
1/2 VOLTA SENTIDO HORARIO CLIQUE     C
1/2 VOLTA SENTIDO ANTI-HORARIO CLIQUE D
1 VOLTA SENTIDO HORARIO CLIQUE       E
1 VOLTA SENTIDO ANTI-HORARIO CLIQUE  F
5 VOLTAS SENTIDO HORARIO CLIQUE     G
5 VOLTAS SENTIDO ANTI-HORARIO CLIQUE H
MODO DE APRENDIZAGEM COM 5 DADOS DIGITE I
```

Figura 24: Terminal Serial Simulado no ISIS

Abaixo na figura (25) encontra-se um fluxograma do funcionamento do programa.

A execução do programa se deu corretamente, lembrando que existe a necessidade de se ajustar o passo do motor e saber como a informação enviada para o driver modificam os passos do motor, para

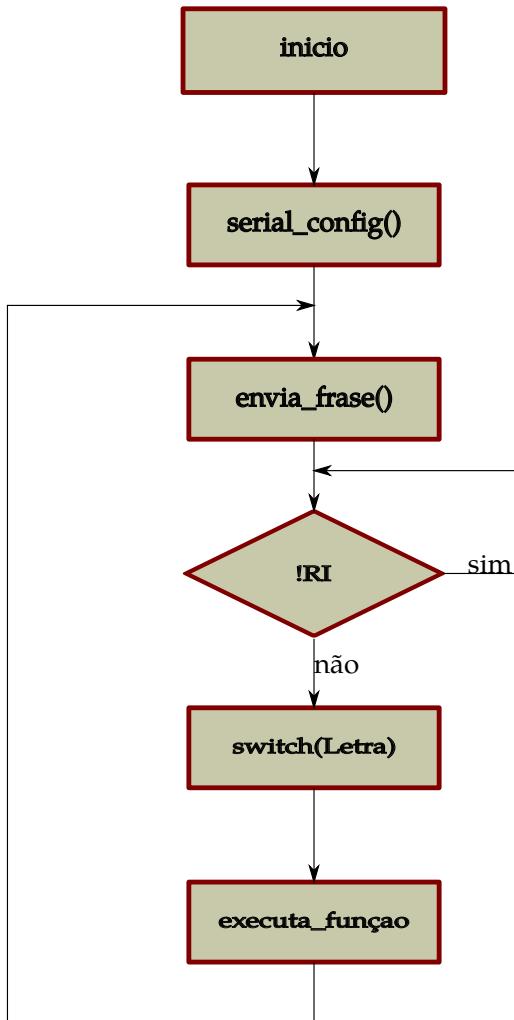
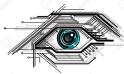


Figura 25: Fluoxograma Expressando a Construção do Programa do Controle Do Motor de Passo

cobrir a tarefa de girar o motor nos ângulos propostos. Foi utilizada na simulação 10º para uma boa visualização do giro do motor. Porém existe grande chance de não haver motores com este passo no mercado. Assim, antes de simular, verifique, caso tenha interesse na realização do projeto, o passo do motor real e colocar nas configurações do proteus, bastando clicar sobre o motor e configurar o passo do mesmo.

Outras opções para o motor também estão disponíveis.



### 3.2 Frequêncimetro

É possível implementar um frequêncimetro com o 80c51, utilizando o fato de que o mesmo possui um contador de pulsos e permite programar um delay, seja via timer ou atraso.

Assim basta deixá-lo contando por um segundo e exibir o valor da contagem. Encontra-se o fato de que o contador é de dezesseis bits, havendo a necessidade de se converter o numero para decimal, fazendo assim a operação de trabalhar com dois registradores, DPH e DPL, traduzindo-os em um só número. O projeto fica completo quando adiciona-se um display que exibe a frequência medida. Como já sabemos o programa do lcd, basta utilizá-lo em conjunto com o main.c do frequêncimetro, possibilitando o uso de funções do lcd.

O fluxograma que exibe o comportamento do programa que controla o sistema encontrase na figura (26).

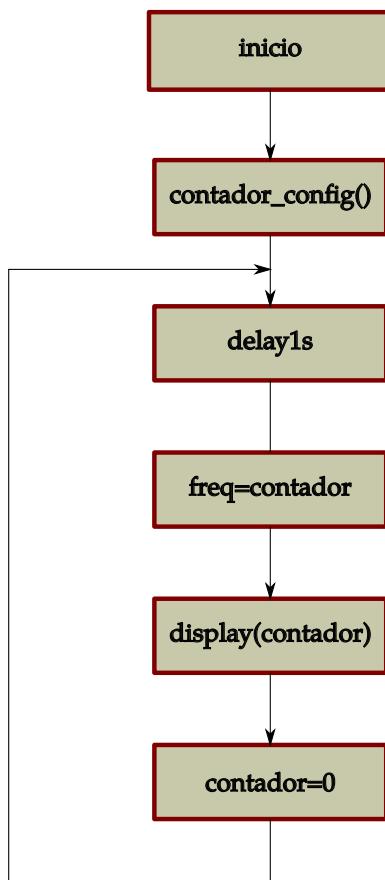


Figura 26: Fluoxograma Expressando a Construção do Porgrama do Frequêncimetro

O funcionamento do programa se deu corretamente com poucos porcento de erro. Lembrando que o micro é capaz de medir no minímo 1 Hz e no máximo a metade de frequência do oscilador do micro. Abaixo na figura (27) encontra-se o display com uma interessante ferramenta que é a de seleção de esquemático. Selecionando o que você deseja ver durante a simulação animada após isso bastando somente cliar em mensagem durante a simulação, com isto irá aparecer somente os itens selecionados. Verifica-se que a frequência exibida no display é de 537Hz enquanto a de entrada é no valor de 535Hz, havendo boa precisão na contagem.

O hardware é apenas o visto para ligar o display, seja no modo 8 bits ou 4 colocando um gerador de onda quadrada no pino do contador, que é T0 ou T1.

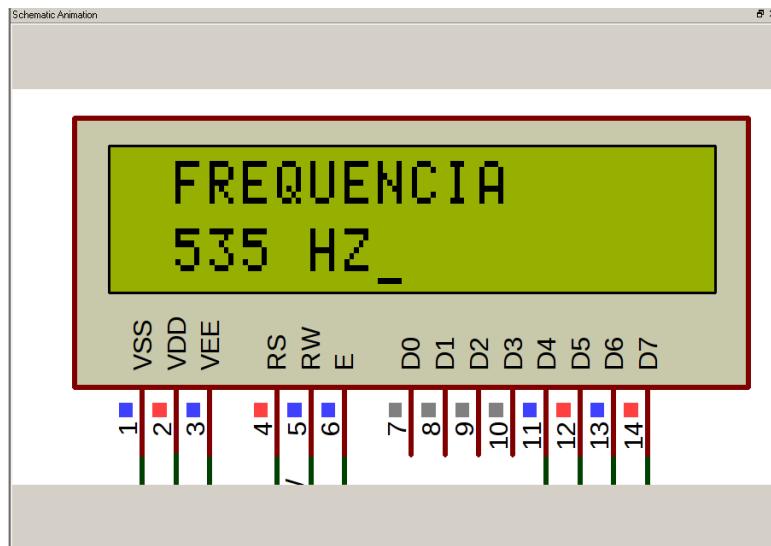


Figura 27: Valor Mostrado no Display Quando a Entrada é 535Hz

### 3.3 Termômetro com Tabela e Polinômio

A conversão digital analógica permite processamento ou tabelamento de dados digitalizados, permitindo que se faça uma relação entre o digitalizado e uma informação, que no presente caso, refere-se a temperatura que o sinal analógico está indicando.

Como já foi comentado a respeito da programação do conversor AD e sua configuração basta dizer que o processo de construir um termômetro é construir um vetor de char, sendo cada char a temperatura do valor lido no conversor AD. Desta forma tem-se a seguinte configuração do programa, expressa no fluxograma da figura (28) abaixo.

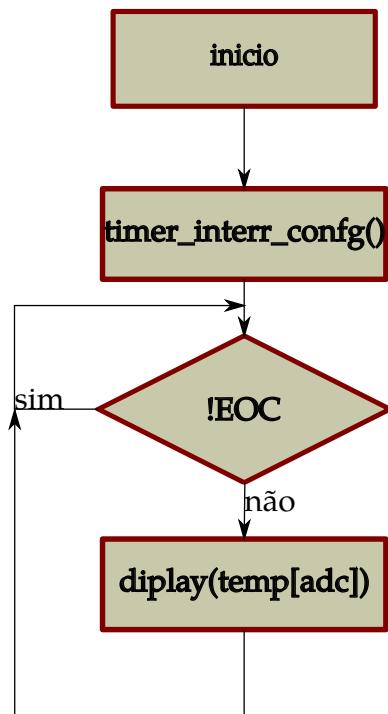


Figura 28: Fluxograma Expressando a Construção do Programa do Controle Do Termômetro por Tabela



Para o programa em questão foi montada uma tabela que ia de  $-15^{\circ}\text{C}$  a  $+15^{\circ}\text{C}$ , para valores de 00h no converor a 1fh, dai para cima mostra limite superior no display.

Para fazer modificações basta modificar os valores na tabela de acordo com o endereço especificado pelo conversor AD.

Abaixo na figura (??) encontra-se o hardware utilizado para realizar a simulação bem como o resultado dado nos display, expresso na figura (29). O termômetro pode ser construído partir de um polinômio que recebe o valor DA convertido processa-o e entrega o valor da temperatuda relacionada. Quanto ao fluxograma de projeto muda somente o fato de que o display imprime a o valor **polinômio(adc)**.

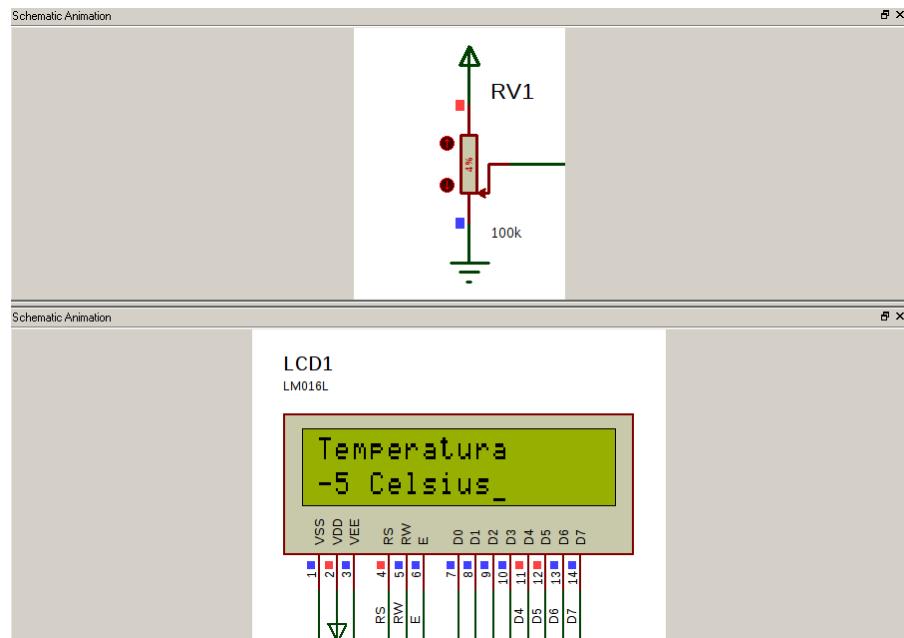
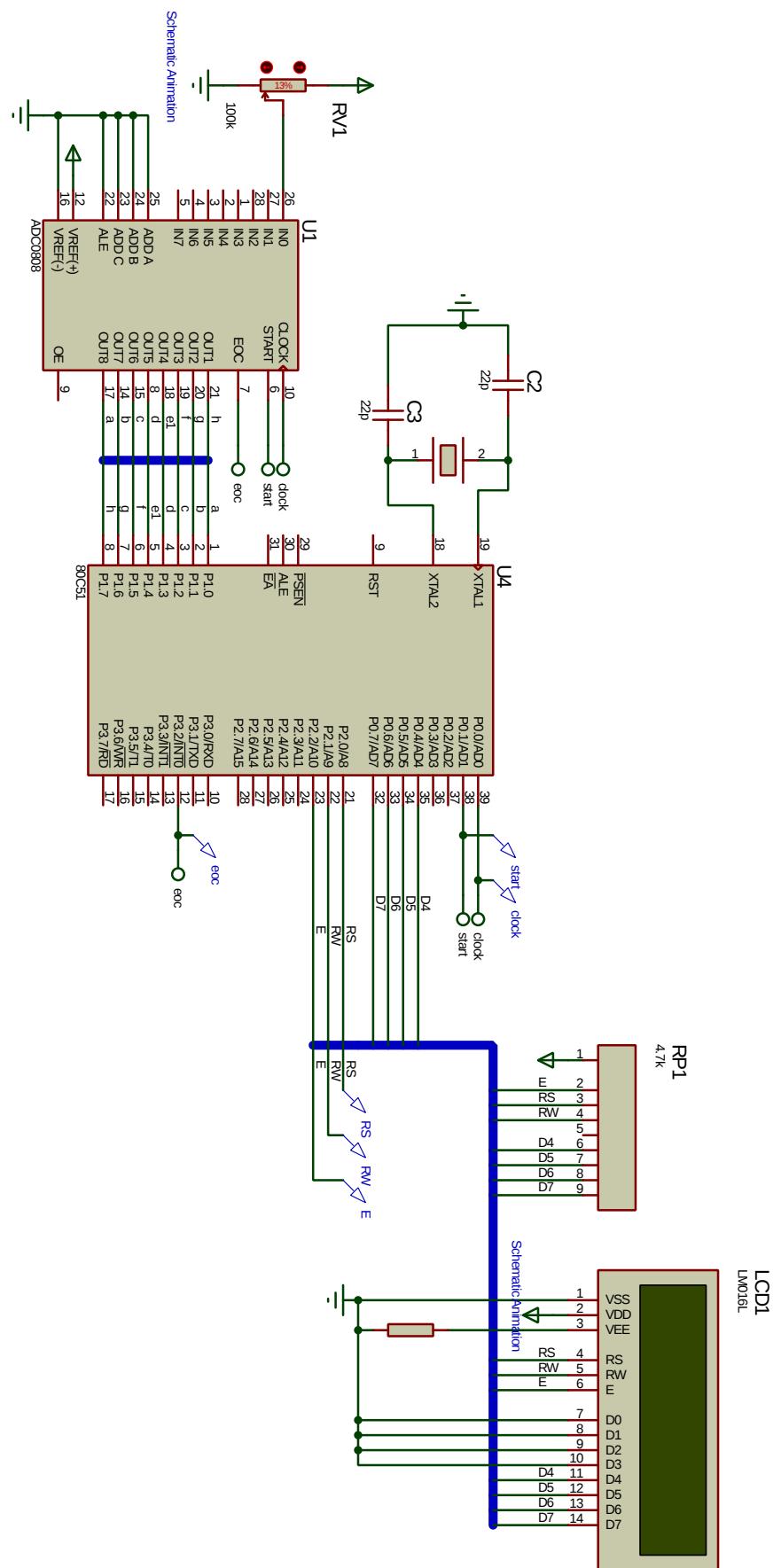


Figura 29: Valor Mostrado No Display Quando o Micro é Programado Como sendo um Termômetro por Tabela.



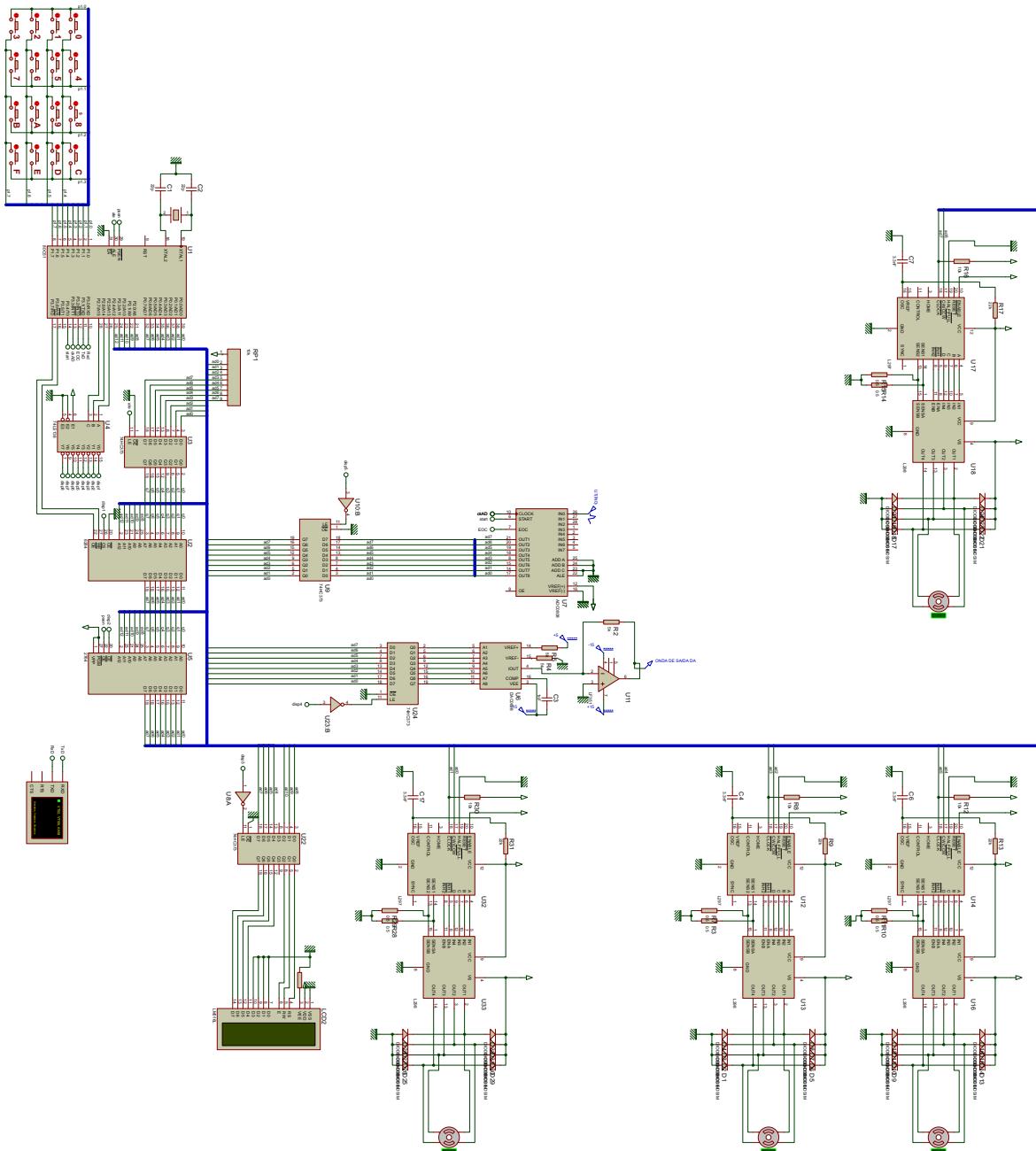


## 4 Projeto de Dispositivos Decodificados

Como vem sido expresso neste trabalho, é muito importante ligar varios dutos da porta P0 do micro de decodificá-la, permitindo assim o uso de varios dispositivos pelo microcontrolador.

Levando em consideração a prática dessa estrutura o projeto seguinte consiste em montar um circuito com o 80c51 que conteha os itens baixo, e simulá-los para verificar seu funcionamento.

- Memória de programa externa mínima de 8k
- Memória de dados externa mínima de 8k
- Teclado Matricial de 16 Caracteres
- Quatro Motores de Passo com Acionamento Independentes
- Um Sensor Óptico Conectado a Entrada T0
- Um Conversor AD de 8 Bits
- Um conversor DA de 8 Bits
- Interface Serial de 8 Bits





Como pode ser visto pelo esquemático, apresentando na figura (??) o mesmo constitue-se de elementos todos estudados nos demais tópicos do projeto total, podendo ser simulado utilizando qualquer um dos códigos feitos, porém fazendo previa decodificação do elemento ou utilizar o endereçamento pelo registrador DPTR, sendo que cada elemento do decodificador, seleciona o dispositivo desejado. Cada passo entre um e outro dispositivo do decodificado possui uma faixa de 1ffffh.

Portanto tem-se:

0000h–1ffffh disp1  
2000h–3ffffh disp2  
4000h–5ffffh disp3  
6000h–7ffffh disp4  
8000h–9ffffh disp5  
a000h–bffffh disp6  
c000h–dffffh disp7  
e000h–fffffh disp8

Outro modo é usar a função MOVC A, @A+DPTR com o DPTR direcionando para o endereço que irá selecionar o referido dispositivo a que se deseja transmitir ou receber um dado.

## 5 Conclusões

No que diz respeito a simulação de circuitos digitais, principalmente a utilização do microcontrolador 80c51, foi dado uma abordagem inicial de como utilizar o sistema PROTEUS ISIS para simular os elementos constituintes do circuito. Lembrando que a gama de microcontroladores disponíveis no ISIS é gigante, sendo que, o objetivo deste relatório é introduzir o usuário ao simulador, deixando a ele novas aventuras, como o aprendizado de novos microcontroladores mais usados no mercado, como PIC's, microcontroladores atmega, base da plataforma arduino. Abaixo, na figura (30), encontra-se um gráfico falando sobre qual microcontrolador de 8 bits é provável de se usar no próximo projeto em questão, aonde vê-se que não há pretenção no uso do 80c51. Assim, aprendendo somente o 80c51 é ter apenas uma base de microcontroladores. O software Proteus ISIS possibilita o aprendizado de novos micros mais usados. Caso tenha sido feita todas as simulações presentes no projeto, o aluno terá uma boa experiência com simulações no proteus, havendo apenas a necessidade de se transitar para um novo microcontrolador, ou seja, uma nova arquitetura e compilador, crucial para dar caminho a sua jornada junto a outro microcontroladores.

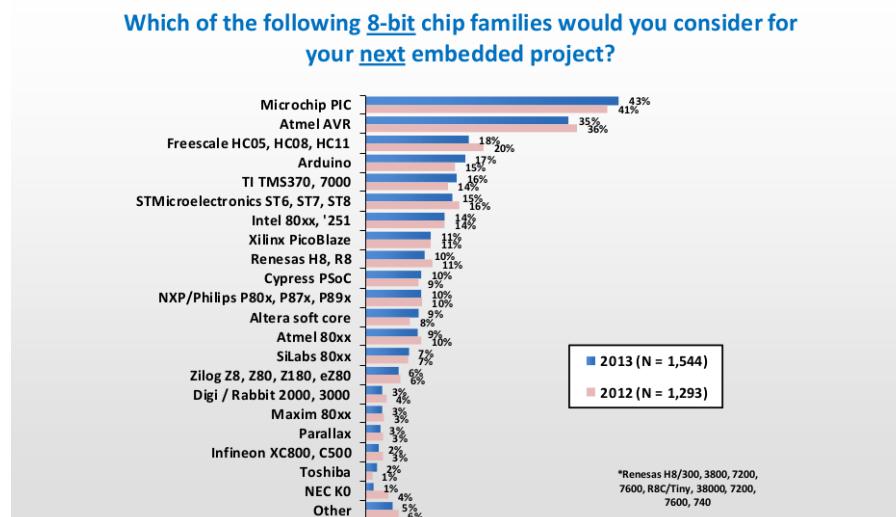
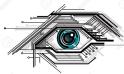


Figura 30: Pretensão de Uso de Microcontroladores de 8 Bits. Estudo Realizado no Ano de 2013 Pela Empresa Embedded



## 6 Apêndice1–Códigos–Assembler

### 6.1 Programa:Chaves e Botões

```
ORG 0000H;  
CHAVE1 EQU P3.5;  
CHAVE2 EQU P3.6;  
CHAVE3 EQU P3.7;  
LED1    EQU P1.0;  
LED2    EQU P1.1;  
LED3    EQU P1.2;  
  
MOV P3,#00H;  
  
CHAVE11:  
    JB CHAVE1,LED11;  
    CLR LED1;  
CHAVE22:  
    JB CHAVE2,LED22;  
    CLR LED2;  
CHAVE33:  
    JB CHAVE3,LED33;  
    CLR LED3;  
    SJMP CHAVE11;  
LED11:  
    SETB LED1;  
    SJMP CHAVE22;  
LED22:  
    SETB LED2;  
    SJMP CHAVE33;  
LED33:  
    SETB LED3;  
    SJMP CHAVE11;  
  
END;
```



## 6.2 Prgrama:Chaves e Botões Com Opção de Piscar Led3 e Led1

```
ORG 0000H;  
CHAVE1 EQU P3.5;  
CHAVE2 EQU P3.6;  
CHAVE3 EQU P3.7;  
LED1    EQU P1.0;  
LED2    EQU P1.1;  
LED3    EQU P1.2;  
  
MOV P1,#00H;  
MOV P3,#00;  
  
CHAVE11:  
    JB CHAVE1,LED11;  
    CLR LED1;  
CHAVE22:  
    JB CHAVE2,LED22;  
    CLR LED2;  
CHAVE33:  
    JB CHAVE3,LED33;  
    CLR LED3;  
    SJMP CHAVE11;  
  
LED11:  
    SETB LED1;  
    LCALL DELAY05S;  
    CLR LED1;  
    LCALL DELAY05S;  
    JB CHAVE1,LED11;  
    SJMP CHAVE22;  
LED22:  
    SETB LED2;  
    LCALL DELAY05S;  
    CLR LED2;  
    LCALL DELAY05S;  
    JB CHAVE1,LED22;  
    SJMP CHAVE33;  
LED33:  
    SETB LED3;  
    LCALL DELAY025S;  
    CLR LED3;  
    SETB LED1;  
    LCALL DELAY025S;  
    CLR LED1;  
    JB CHAVE1,LED33;  
    SJMP CHAVE11;  
  
DELAY05S:  
    MOV     R2, #004h  
    MOV     R1, #0E8h  
    MOV     R0, #0F6h
```



```
NOP
DJNZ    R0, $
DJNZ    R1, $-5
DJNZ    R2, $-9
MOV     R0, #0F9h
DJNZ    R0, $
NOP
RET;
```

DELAY025S:

```
MOV     R2, #01Ah
MOV     R1, #0B1h
MOV     R0, #017h
NOP
DJNZ    R0, $
DJNZ    R1, $-5
DJNZ    R2, $-9
MOV     R0, #06Eh
DJNZ    R0, $
NOP
RET;
```

END;



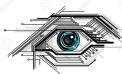
### 6.3 Programa que Preenche uma Memória Ram Externa de 64Kbytes

```
ORG 0000H;  
MOV DPTR,#0000H;  
MOV A,#00H;  
  
RAM:  
    MOVX @DPTR,A;  
    INC DPTR;  
    INC A;  
    SJMP RAM;  
  
END;
```



#### 6.4 Envio Continuo de Caracter, Caracter Considerado 'F'

```
ORG 0000H;  
LCALL SERIAL_CONFIG;  
SERIAL:  
    MOV SBUF, 'F';  
    JNB TI,$;  
    CLR TI;  
    SJMP SERIAL;  
  
SERIAL_CONFIG:  
    MOV TMOD,#20h;  
    MOV TH1,#253;  
    MOV TL1,#253d  
    MOV SCON,#01000000b;  
    SETB TR1;  
    RET;  
  
END;
```



## 6.5 Programa Que Envia Frase Para Serial Dependendo da Letra de Entrada

```
ORG 0000H;  
LJMP INICIO;  
  
ORG 0023H  
CLR ES;  
MOV A,SBUF;  
CLR RI;  
CLR 20H.1;  
RETI;  
*****  
INICIO:  
  
    SETB 20H.1;  
    LCALL SERIAL_CONFIG;  
    LCALL INTER_CONFIG;  
    MOV DPTR,#0000H;  
INICIO2:  
    SETB ES;  
    SETB 20H.1;  
    JB 20H.1,$;  
  
    CJNE A,#'A',TESTEB;  
    LCALL ENV_FRASE1;  
    SJMP INICIO2;  
TESTEB:  
    CJNE A,#'B',TESTEC;  
    LCALL ENV_FRASE2;  
    SJMP INICIO2;  
TESTEC:  
    CJNE A,#'C',TESTED  
    LCALL ENV_FRASE3;  
    SJMP INICIO2;  
TESTED:  
    CJNE A,#'D',TESTEE;  
    LCALL ENV_FRASE4;  
    SJMP INICIO2;  
TESTEE:  
    CJNE A,#'E',TESTEF;  
    LCALL ENV_FRASE5;  
    SJMP INICIO2;  
TESTEF:  
    CJNE A,#'F',ERROLETRA;  
    LCALL IMPRIME;  
    SJMP INICIO2;  
  
ERROLETRA:  
    LCALL ERRO3;  
    SJMP INICIO2;  
*****  
ENV_FRASE1:
```



```
MOV DPTR,#FRASE1;
LCALL ENVIO_SERIAL;
SJMP FIM;

ENV_FRASE2:
MOV DPTR,#FRASE2;
LCALL ENVIO_SERIAL;
SJMP FIM;

ENV_FRASE3:
MOV DPTR,#FRASE3;
LCALL ENVIO_SERIAL;
SJMP FIM;

ENV_FRASE4:
MOV DPTR,#FRASE4;
LCALL ENVIO_SERIAL;
SJMP FIM;

ENV_FRASE5:
MOV DPTR,#FRASE5;
LCALL ENVIO_SERIAL;
SJMP FIM;

ERRO:
MOV DPTR,#ERRO2;
LCALL ENVIO_SERIAL;
SJMP FIM;

ERRO3:
MOV DPTR,#ERRO1;
LCALL ENVIO_SERIAL;
SJMP FIM;

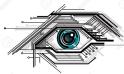
FRASEINICIO1:
MOV DPTR, #FRASEINICIO;
LCALL ENVIO_SERIAL;

FIM:
RET;

;*****  
IMPRIME:
CLR A;
MOV DPH,#01H;
CLR A;
MOVX A,@DPTR;
CJNE A,#05H,TESTEMM;

TESTEMM:
JNC ERRO3;
DEC DPH;
MOV A,DPL;
SUBB A,#05H;
MOV DPL,A;
MOV B,#00H;

CONTINUA:
MOVX A,@DPTR;
MOV SBUF,A;
JB TI,$;
CLR TI;
```



```
INC DPTR;
INC B;
MOV @R0,B;
CJNE @R0,#05H,CONTINUA;
RET;

;*****ENVIO_SERIAL:
CLR A;
MOVC A,@A+DPTR;

ENVIO: MOV SBUF,A;
JNB TI,$;
CLR TI;

INC DPTR;
CLR A;
MOVC A,@A+DPTR;
CJNE A,#'$',ENVIO;
RET;

INTER_CONFIG:
SETB ES
SETB EA
RET;

SERIAL_CONFIG:
MOV TMOD,#20H
MOV TH1,#253D
MOV TL1,#253D
MOV SCON,#01000000B
SETB TR1
SETB REN
RET;

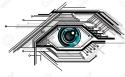
;*****DIRECCIONA_FRASE:
CJNE A,#'A',CARACTERB;
MOV A,#'A';
RET;

CARACTERB:
CJNE A,#'B',CARACTERC;
MOV A,#'B';
RET;

CARACTERC:
CJNE A,#'D',CARACTERD;
MOV A,#'D';
RET;

CARACTERD:
CJNE A,#'E',CARACTERF;
MOV A,#'E';
RET;

CARACTERF:
```



```
MOV A,# 'F';
RET;

FRASE1: DB '1',0AH,0DH,'$';
FRASE2: DB '2',0AH,0DH,'$';
FRASE3: DB '3',0AH,0DH,'$';
FRASE4: DB '4',0AH,0DH,'$';
FRASE5: DB '5',0AH,0DH,'$';
ERRO1: DB '6',0AH,0DH,'$';
ERRO2: DB '7',0AH,0DH,'$';
FRASEINICIO: DB 'ENTRE_SOMENTE_COM_CARACTERES_A_B_C_D_E_F',0AH,0DH,
'ENTRANDO_COM_A_LETRA_F_IMPRIME_AS_ULTIMAS_CINCO_FRASES_QUE_FORAM
REPASSADAS',0AH,0DH,'$';
END;
```



## 6.6 Conversor AD DA ideal

ORG 0000H;

LOOP:

MOV P2,P1;  
SJMP LOOP;

END;



## 6.7 Exercício Conversor AD

```
CLOCK EQU P3.3;
EOC     EQU P3.2;

ORG 0000H;
SJMP INICIO;

ORG 0003H
MOV A,P0;
RETI;

ORG 001BH;
CPL CLOCK;
LCALL INICIA_TIMER1;
RETI;

INICIO:
SETB P2.5;
CLR P2.6;
SETB P2.7;

CLR CLOCK;
LCALL INTERRUPTION_CONFIG;
LCALL INICIA_TIMER1;
LCALL DELAY10US;

LOOP:
JNB EOC,$;
CPL P1.1;
CLR EX0;

MOV P0,A;
SETB EX0;
LCALL DELAY10US;
SJMP LOOP;

INTERRUPTION_CONFIG:
SETB EA;
SETB EX0;
SETB ET0;
SETB PT1;
SETB ET0;
MOV TCON,#10H;
RET;

INICIA_TIMER1:
MOV TH1,#0FFH;
MOV TL1,#0FEH;
SETB TR1;
```



RET;

DELAY10US:

MOV R0, #003h

NOP

DJNZ R0, \$

NOP

NOP

RET;

END;



## 6.8 Exercício Motor de Passo

```
CLOCK EQU P3.3;  
EOC EQU P3.2;  
  
ORG 0000H;  
SJMP INICIO;  
  
ORG 0003H  
MOV A,P0;  
RETI;  
  
ORG 001BH;  
CPL CLOCK;  
LCALL INICIA_TIMER1;  
RETI;  
  
INICIO:  
SETB P2.5;  
CLR P2.6;  
SETB P2.7;  
  
CLR CLOCK;  
LCALL INTERRUPTION_CONFIG;  
LCALL INICIA_TIMER1;  
LCALL DELAY10US;  
LOOP:  
JNB EOC,$;  
CPL P1.1;  
CLR EX0;  
  
MOV P0,A;  
SETB EX0;  
LCALL DELAY10US;  
SJMP LOOP;  
  
INTERRUPTION_CONFIG:  
SETB EA;  
SETB EX0;  
SETB ET0;  
SETB PT1;  
SETB ET0;  
MOV TCON,#10H;  
RET;  
  
INICIA_TIMER1:  
MOV TH1,#0FFH;  
MOV TL1,#0FEH;  
SETB TR1;
```



RET;

DELAY10US:

MOV R0, #003h

NOP

DJNZ R0, \$

NOP

NOP

RET;

END;



## 6.9 Exercício:Display De Sete Segmentos

```
DISP1 EQU P0.4;
DISP2 EQU P0.5;
DISP3 EQU P0.6;
DISP4 EQU P0.7;
DADOS EQU P2;

ORG 0000H;
CLR DISP1;
CLR DISP2;
CLR DISP3;
CLR DISP4;

INICIO :
    SETB DISP1;
    MOV DADOS,#01h ;
    LCALL DELAY;
    CLR DISP1;

    SETB DISP2;
    MOV DADOS,#05H;
    LCALL DELAY;
    CLR DISP2;

    SETB DISP3;
    MOV DADOS,#08H;
    LCALL DELAY;
    CLR DISP3;

    SETB DISP4;
    MOV DADOS,#03H;
    LCALL DELAY;
    CLR DISP4;
    SJMP INICIO ;

DELAY:
    MOV      R1, #05Fh
    MOV      R0, #05Fh
    NOP
    DJNZ    R0, $
    DJNZ    R1, $-5
    NOP
    RET;

END;
```



## 6.10 Programa LCD 8 Bits

```
RS EQU P2.0;
RW EQU P2.1;
E  EQU P2.2;

.

ORG 0000H;
INICIO:
    LCALL INIT_LCD ;
    MOV A,#00H;
    LCALL POS_LCD;
    MOV DPTR,#TAB1;
    LCALL ENVIA_FRASE;

    MOV A,#40H;
    LCALL POS_LCD;
    MOV DPTR,#TAB2;
    LCALL ENVIA_FRASE;
    SJMP $;

INIT_LCD:
    MOV A,#38H;
    LCALL WRITEINSTRUCOES;
    MOV A,#0EH;
    LCALL WRITEINSTRUCOES;
    MOV A,#06H;
    LCALL WRITEINSTRUCOES;
    MOV A ,#01H;
    LCALL WRITEINSTRUCOES;
    RET;

ENVIA_FRASE:
    CLR A;
    MOVC A,@A+DPTR;
CONTINUA:
    LCALL WRITEDADOS;
    INC DPTR;
    CLR A;
    MOVC A,@A+DPTR;
    CJNE A,# '$' ,CONTINUA;
    RET;
    SJMP $;

WRITEINSTRUCOES:
    SETB E;
    CLR RS;
    CLR RW;
    MOV DADOS_INSTRUCOES,A;
    CLR E;
```



```
LCALL WAITLCD;  
RET;
```

WRITEDADOS:

```
SETB E;  
SETB RS;  
CLR RW;  
MOV DADOS_INSTRUCOES,A;  
CLR E;  
LCALL WAITLCD;  
RET;
```

WAITLCD:

```
MOV DADOS_INSTRUCOES,#0FFH;  
CLR E;  
CLR RS;  
SETB RW;  
SETB E;  
NOP;  
MOV A, DADOS_INSTRUCOES;  
ANL A,#80H;  
CJNE A,#00H,WAITLCD;  
CLR E;  
RET;
```

POS\_LCD:

```
ADD A,#80H;  
LCALL WRITEINSTRUCOES;  
RET;
```

```
TAB1: DB 'ASSEMBLY_8_BITS_$'  
TAB2: DB 'COM_DUAIS_LINHAS_$'  
END;
```



## 6.11 Programa LCD 4 Bits

```
RS EQU P2.0;
RW EQU P2.1;
E  EQU P2.2;

ORG 0000H;
LCALL INIT_LCD ;
MOV A,#00H;
LCALL POS_LCD ;
MOV DPTR,#TAB1;
LCALL ENVIA_FRASE;
MOV A,#40H;
LCALL POS_LCD ;
MOV DPTR,#TAB2;
LCALL ENVIA_FRASE;
MOV P1,#00H;
SJMP $;

INIT_LCD:
    MOV A,#32H;
    LCALL WRITEINSTRUCOES;
    MOV A,#28H;
    LCALL WRITEINSTRUCOES;
    MOV A,#0EH;
    LCALL WRITEINSTRUCOES;
    MOV A,#06H;
    LCALL WRITEINSTRUCOES;
    MOV A ,#01H;
    LCALL WRITEINSTRUCOES;
    RET;

WRITEINSTRUCOES:
    SETB E;
    CLR RS;
    CLR RW;
    MOV R7,A;
    ANL A,#0F0H;
    MOV DADOS_INSTRUCOES,A;
    CLR E;
    LCALL WAITLCD;
    SETB E;
    MOV A,R7;
    ANL A,#0FH;
    SWAP A;
    MOV DADOS_INSTRUCOES,A;
    CLR E;
    LCALL WAITLCD;
    SETB E;
    RET;
```



WRITEDADOS:

```
SETB E;  
SETB RS;  
CLR RW;  
MOV R7,A;  
ANL A,#0F0H;  
MOV DADOS_INSTRUCOES,A;  
CLR E;  
LCALL WAITLCD;  
SETB E;  
MOV A,R7;  
ANL A,#0FH;  
SWAP A;  
MOV DADOS_INSTRUCOES,A;  
CLR E;  
RET;
```

WAITLCD:

```
MOV R1, #006h  
MOV R0, #0E4h  
NOP  
DJNZ R0, $  
DJNZ R1, $-5  
NOP  
NOP  
NOP  
NOP  
RET;
```

POS\_LCD:

```
ADD A,#80H;  
LCALL WRITEINSTRUCOES;  
RET;
```

ENVIA\_FRASE:

```
CLR A;  
MOVC A,@A+DPTR;
```

CONTINUA:

```
LCALL WRITEDADOS;  
INC DPTR;  
CLR A;  
MOVC A,@A+DPTR;  
CJNE A,#'$',CONTINUA;  
RET;
```

TAB1: DB 'ASSEMBLY\_4\_BITS\_\$'  
TAB2: DB 'COM\_DUAS\_LINHAS\_\$'  
END;



## 7 Programas em Linguagem C Compiladas por SDCC

### 7.1 Botões e Chaves

```
/* Main.c file generated by New Project wizard
 *
 * Created:    qui dez 10 2015
 * Processor:  80C51
 * Compiler:   SDCC for 8051
 */
#include <mcs51reg.h>

void main(void)
{
    while(1){
        P1=P0;
    }
}
```



## 7.2 Envio Continuo de Caracter

```
/* Main.c file generated by New Project wizard
*
* Created: qua dez 9 2015
* Processor: 80C51
* Compiler: SDCC for 8051
*/
#include <mcs51reg.h>

void serial_config(void);

void main(void)
{
    serial_config();
envia:
    SBUF='a';
    while (!TI);
        TI=0;
    goto envia;
}

void serial_config(void){
    TMOD=0x20;
    SCON=0x50;
    TH1=253;
    TL1=253;
    TR1=1;
}
```



### 7.3 Envio De Caracter Via Interrupção

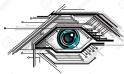
```
/* Main.c file generated by New Project wizard
*
* Created: qua dez 9 2015
* Processor: 80C51
* Compiler: SDCC for 8051
*/
#include <mcs51reg.h>
void serial_config(void);
void interrupt_config(void);
void serial_interrupcao(void) __interrupt(4);

void serial_interrupcao(void) __interrupt(4){
    RI=0;
    SBUF='a';
    while (!TI);
    TI=0;
}

void main(void)
{
    serial_config();
    interrupt_config();
    while (1);
}

void serial_config(void){
TMOD=0x20;
SCON=0x50;
TH1=253;
TL1=253;
TR1=1;
REN=1;
}

void interrupt_config(void){
EA=1;
ES=1;
}
```



## 7.4 LCD 8 Bits

```
/* Main.c file generated by New Project wizard
 *
 * Created:    qui dez 10 2015
 * Processor:  80C51
 * Compiler:   SDCC for 8051
 */

#include <mcs51reg.h>
#define RS P2_0
#define RW P2_1
#define E  P2_2

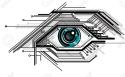
void lcd_init(void);
void write_instrucoes(unsigned char );
void write_caracteres(unsigned char );
void wait_lcd(void);
void posiciona_cursor(unsigned char );
void write_frases(unsigned char *);

__code unsigned char frase1 []="DISPLAY_LCD_EM_C";
__code unsigned char frase2 []="COM_8_BITS";

void main(void)
{
    lcd_init();
    write_frases(frase1);
    posiciona_cursor(0x40);
    write_frases(frase2);
    while(1);
}

void lcd_init(void){
    write_instrucoes(0x38);
    write_instrucoes(0x0E);
    write_instrucoes(0x06);
    write_instrucoes(0x01);
}

void write_instrucoes(unsigned char instrucoes){
    E=1;
    RS=0;
    RW=0;
    P0=instrucoes;
    E=0;
    wait_lcd();
}
```



```
}
```

```
void write_caracteres(unsigned char caracteres){
    E=1;
    RS=1;
    RW=0;
    P0=caracteres;
    E=0;
    wait_lcd();
}
```

```
void wait_lcd(void){
    unsigned int wait_flag;
    do{
        P0=0xff;
        E=0;
        RS=0;
        RW=1;
        E=1;
        wait_flag=P0;
    }while( wait_flag & 0x80 );
}
```

```
void posiciona_cursor(unsigned char pos_cursor){
    pos_cursor+=0x80;
    write_instrucoes(pos_cursor);
}
```

```
void write_frases(unsigned char *frases){
    do{
        write_caracteres(*frases++);
    }while(*frases);
}
```



## 7.5 LCD 4 Bits

```
* Main.c file generated by New Project wizard
*
* Created: sex dez 11 2015
* Processor: 80C51
* Compiler: SDCC for 8051
*/
#include <mcs51reg.h>

#define RS P2_0
#define RW P2_1
#define E P2_2

void init_lcd(void);
void write_instrucoes(unsigned int);
void write_caracteres(unsigned char);
void pos_cursor(unsigned int);
void pos_cursor(unsigned int);
void write_lcd(unsigned char *);
void wait_lcd(void);

__code unsigned char frase1 []="LCD_4_BITS_EM_C";
__code unsigned char frase2 []="COMDUAS_LINHAS";

void main(void)
{
    init_lcd();
    write_lcd(frase1);
    pos_cursor(0x40);
    write_lcd(frase2);
    while(1);
}

void init_lcd(void){
    write_instrucoes(0x32);
    write_instrucoes(0x28);
    write_instrucoes(0x0E);
    write_instrucoes(0x06);
    write_instrucoes(0x01);
}

void write_instrucoes(unsigned int instrucoes){
    unsigned int aux_inst;
    E=1;
    RS=0;
    RW=0;
    aux_inst=instrucoes;
    instrucoes=(instrucoes & 0xf0);
    P0=instrucoes;
```



```
E=0;
wait_lcd ();
E=1;
instrucoes=aux_inst;
instrucoes=(instrucoes & 0x0f);
instrucoes=instrucoes<<4;
P0=instrucoes;
E=0;
wait_lcd ();
E=1;
}
void write_caracteres(unsigned char caracter){
    unsigned char aux_carac;
    E=1;
    RS=1;
    RW=0;
    aux_carac=caracter;
    caracter=(caracter & 0xf0);
    P0=caracter;
    E=0;
    wait_lcd ();
    E=1;
    caracter=aux_carac;
    caracter=(caracter & 0x0f);
    caracter=caracter<<4;
    P0=caracter;
    E=0;
    wait_lcd ();
}
void pos_cursor(unsigned int pos){
    pos+=0x80;
    write_instrucoes(pos);
}
void write_lcd(unsigned char *frases){
    do{
        write_caracteres(*frases++);
    }while(*frases);
}
void wait_lcd(void){
    unsigned long int j;
    for(j=0; j<=50; j++);
}
```



## 7.6 Envio De Frase Dependendo da Letra

```
/* Main.c file generated by New Project wizard
*
* Created: ter dez 8 2015
* Processor: 80C51
* Compiler: SDCC for 8051
*/
#include <mcs51reg.h>

void serial_interrupcao() __interrupt(4);
void serial_config(void);
void interrupt_config(void);
void envia_frase(unsigned char frases []);

volatile unsigned char letra;
volatile unsigned int interruption=0;
__code unsigned char frase1 []="letra_a\n\r";
__code unsigned char frase2 []="letra_b\n\r";
__code unsigned char frase3 []="letra_c\n\r";
__code unsigned char frase4 []="letra_desconhecida\n\r";

void serial_interrupcao() __interrupt(4){
    RI=0;
    letra=SBUF;
    interruption=1;
}

void main(void)
{
    serial_config();
    interrupt_config();
inicio:
    while(interruption==0);

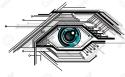
    switch(letra){

        case 'a':
            envia_frase(frase1);
            break;

        case 'b':
            envia_frase(frase2);
            break;

        case 'c':
            envia_frase(frase3);
            break;

        default:
```



```
    envia_frase( frase4 );
}

interruption=0;
goto inicio ;
}

void serial_config(void){
TMOD=0x20;
SCON=0x50;
TH1=253;
TL1=253;
TR1=1;
REN=1;
}

void interrupt_config(void){
EA=1;
ES=1;
}

void envia_frase(unsigned char *frases){
unsigned int i=0;
do{
    SBUF=frases [ i ];
    while (!TI);
        TI=0;
        i++;
    }while( frases [ i ]!= '\0' );
}
```



## 7.7 Frequêncimetro

### 7.7.1 Arquivo main.c

```
#include "frequencimetro.h"

--code unsigned char frase1 []="FREQUENCIA";
--code unsigned char frase2 []="HZ";

short int seg_nibble;

void main(void){
    signed short int i;
    unsigned char aux1,aux2,aux3,aux4;
    unsigned char th1_sof,t11_sof,resto,vec[6];
    unsigned int atraso=131,freq_dec;

    init_lcd();
    timer_contador_config();
    write_lcd(frase1);
    TH1=0x00;
    TL1=0x00;

    while(1){
        delay_ms(atraso);
        th1_sof=TH1;
        t11_sof=TL1;

        aux1=(th1_sof & 0xf0);
        aux1=aux1>>4;

        aux2=(th1_sof & 0x0f);
        aux3=(t11_sof & 0xf0);
        aux3=aux3>>4;

        aux4=(t11_sof & 0x0f);

        freq_dec= aux1*powf(16,3)+aux2*powf(16,2)+aux3*powf(16,1)+aux4*powf(16,0);

        i=0;

        while(freq_dec){
            resto=freq_dec%10;
            resto=hex_asc_converter(resto);
            vec[i]=resto;
            freq_dec=freq_dec/10;
            i++;
        }

        seg_nibble=0;
        pos_cursor(0x40);
```



```
while(i>=-1){
    write_caracteres(vec[i]);
    i--;
}

write_lcd(frase2);

TH1=0x00;
TL1=0x00;
}

void timer_contador_config(void){
TMOD=0x50;
TR1=1;
}

unsigned char hex_asc_converter(unsigned char dph_dpl_hex_asc){
if(seg_nibble==1){
    dph_dpl_hex_asc=dph_dpl_hex_asc>>4;
    dph_dpl_hex_asc=(dph_dpl_hex_asc & 0x0f);
}
else
    dph_dpl_hex_asc=(dph_dpl_hex_asc & 0x0f);

if(dph_dpl_hex_asc < 0x0a)
    dph_dpl_hex_asc+=0x30;

else
    dph_dpl_hex_asc+=0x37;
return(dph_dpl_hex_asc);}

void delay_ms(unsigned int itime) {
unsigned int i,j;
for (i=0; i<itime; i++)
for (j=0; j<500; j++);
}

7.7.2 arquivo lcd.c
```

```
#include "frequencimetro.h"
```

```
void init_lcd(void){
    write_instrucoes(0x32);
    write_instrucoes(0x28);
    write_instrucoes(0x0E);
    write_instrucoes(0x06);
    write_instrucoes(0x01);
}
```



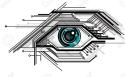
```
void write_instrucoes(unsigned int instrucoes){
    unsigned int aux_inst;
    E=1;
    RS=0;
    RW=0;
    aux_inst=instrucoes;
    instrucoes=(instrucoes & 0xf0);
    P0=instrucoes;
    E=0;
    wait_lcd();
    E=1;
    instrucoes=aux_inst;
    instrucoes=(instrucoes & 0x0f);
    instrucoes=instrucoes<<4;
    P0=instrucoes;
    E=0;
    wait_lcd();
    E=1;
}

void write_caracteres(unsigned char caracter){
    unsigned char aux_carac;
    E=1;
    RS=1;
    RW=0;
    aux_carac=caracter;
    caracter=(caracter & 0xf0);
    P0=caracter;
    E=0;
    wait_lcd();
    E=1;
    caracter=aux_carac;
    caracter=(caracter & 0x0f);
    caracter=caracter<<4;
    P0=caracter;
    E=0;
    wait_lcd();
}

void pos_cursor(unsigned int pos){
    pos+=0x80;
    write_instrucoes(pos);
}

void write_lcd(unsigned char *frases){
    do{
        write_caracteres(*frases++);
    }while(*frases);
}

void wait_lcd(void){
```



```
unsigned long int j;  
for(j=0; j<=50; j++);  
}
```

### 7.7.3 arquivo frequencimetro.h

```
#ifndef b  
#define b  
  
#include <mcs51reg.h>  
#include <math.h>  
  
#define RS P2_0  
#define RW P2_1  
#define E P2_2  
  
void init_lcd(void);  
void write_instrucoes(unsigned int);  
void write_caracteres(unsigned char);  
void pos_cursor(unsigned int);  
void pos_cursor(unsigned int);  
void write_lcd(unsigned char *);  
void wait_lcd(void);  
void timer_contador_config(void);  
unsigned char hex_asc_converter(unsigned char);  
void delay_ms(unsigned int);  
float powf(const float x, const float);  
  
extern short int seg_nibble;  
extern __code unsigned char frase1[];  
extern __code unsigned char frase2[];  
  
#endif
```



## 7.8 Motor de Passo

### 7.8.1 Arquivo main.c

```
/* main.c file generated by New Project wizard
 * Created: seg dez 14 2015
 * Processor: 80C51
 * Compiler: SDCC for 8051
 */

#include "MOTOR_DE_PASSO.h"

#define enable P2_0
#define reset P2_1
#define halffull P2_2
#define sentido P2_3
#define clock P2_4
#define home P2_5

__code unsigned char frase1 [] = "1.PASSO.SENTIDO.HORARIO.CLIQUE.A\n\r";
__code unsigned char frase2 [] = "1.PASSO.SENTIDO.ANTI-HORARIO.CLIQUE.B\n\r";
__code unsigned char frase3 [] = "1/2.VOLTA.SENTIDO.HORARIO.CLIQUE.C\n\r";
__code unsigned char frase4 [] = "1/2.VOLTA.SENTIDO.ANTI-HORARIO.CLIQUE.D\n\r";
__code unsigned char frase5 [] = "1.VOLTA.SENTIDO.HORARIO.CLIQUE.E\n\r";
__code unsigned char frase6 [] = "1.VOLTA.SENTIDO.ANTI-HORARIO.CLIQUE.F\n\r";
__code unsigned char frase7 [] = "5.VOLTAS.SENTIDO.HORARIO.CLIQUE.G\n\r";
__code unsigned char frase8 [] = "5.VOLTAS.SENTIDO.ANTI-HORARIO.CLIQUE.H\n\r";
__code unsigned char frase9 [] = "MODO.DE.APRENDIZAGEM.COM.5.DADOS.DIGITE.I\n\r";

volatile unsigned int int_atendida;
volatile unsigned char escolha;

void main(void){

    unsigned int i;
    unsigned char vec[5];

    halffull=0;
    enable=1;

    serial_config();
inicio:
    home=0;
    printf(frase1);
    printf(frase2);
    printf(frase3);
    printf(frase4);
    printf(frase5);
    printf(frase6);
    printf(frase7);
    printf(frase8);
    printf(frase9);}
```



```
interrupt_config ();
int_atendida=0;
while (! int_atendida );
int_atendida=0;

switch(escolha){
    case 'a':
        sentido=0;
        clock=0;
        delay33ms();
        clock=1;
        delay33ms();
    break;

    case 'b':
        sentido=1;
        clock=0;
        delay33ms();
        clock=1;
        delay33ms();
    break;

    case 'c':
        sentido=0;
        for (i=0;i<=17;i++){
            clock=0;
            delay33ms();
            clock=1;
            delay33ms();
        }
    break;

    case 'd':
        sentido=1;
        for (i=0;i<=17;i++){
            clock=0;
            delay33ms();
            clock=1;
            delay33ms();
        }
    break;

    case 'e':
        sentido=0;
        for (i=0;i<=35;i++){
            clock=0;
            delay33ms();
            clock=1;
            delay33ms();
        }
    break;
```



```
case 'f':
    sentido=1;
    for ( i=0;i<=35;i++){
        clock=0;
        delay33ms ();
        clock=1;
        delay33ms ();
    }
break;

case 'g':
    sentido=0;
    for ( i=0;i<=179;i++){
        clock=0;
        delay33ms ();
        clock=1;
        delay33ms ();
    }
break;

case 'h':
    sentido=1;
    for ( i=0;i<=179;i++){
        clock=0;
        delay33ms ();
        clock=1;
        delay33ms ();
    }
break;

case 'i':
    printf ("_ENTRE_COM_CINCO_MOVIMENTOS\n\r");
    int_atendida=0;
    goto aprendizado;
    volta_aprendizado:
break;

default:
    printf ("_LETRA_DESCOHECIDA\n\r");
}
printf ("\n\r..\n\r..\n\r");
goto inicio;

aprendizado:
for ( i=0;i<5;i++){
    while (!int_atendida);
    SBUF=escolha;
    while (!TI);
    TI=0;
    delay ();
```



```
    vec [ i ] = escolha ;
    int_atendida = 0 ;
}

for ( i = 0; i < 5; i ++ ) {

    switch ( vec [ i ] ) {
        case 'a':
            sentido = 0;
            clock = 0;
            delay33ms ();
            clock = 1;
            delay33ms ();
        break;

        case 'b':
            sentido = 1;
            clock = 0;
            delay33ms ();
            clock = 1;
            delay33ms ();
        break;

        case 'c':
            sentido = 0;
            for ( i = 0; i <= 17; i ++ ) {
                clock = 0;
                delay33ms ();
                clock = 1;
                delay33ms ();
            }
        break;

        case 'd':
            sentido = 1;
            for ( i = 0; i <= 17; i ++ ) {
                clock = 0;
                delay33ms ();
                clock = 1;
                delay33ms ();
            }
        break;

        case 'e':
            sentido = 0;
            for ( i = 0; i <= 35; i ++ ) {
                clock = 0;
                delay33ms ();
                clock = 1;
                delay33ms ();
            }
    }
}
```



```
break;

case 'f':
    sentido=1;
    for ( i=0;i <=35;i++){
        clock=0;
        delay33ms ();
        clock=1;
        delay33ms ();
    }
break;

case 'g':
    sentido=0;
    for ( i=0;i <=179;i++){
        clock=0;
        delay33ms ();
        clock=1;
        delay33ms ();
    }
break;

case 'h':
    sentido=1;
    for ( i=0;i <=179;i++){
        clock=0;
        delay33ms ();
        clock=1;
        delay33ms ();
    }
break;

default:
    printf (" .LETRA DESCOHECIDA\n\r ");
}
}
goto volta_aprendizado;
}

void delay33ms(void){
    unsigned int j,k;
    for (j=0;j <=63;j++)
        for (k=0;k <=50;k++);
    }

void delay(void){
    unsigned int m,n;
    for (m=0;m <=650;m++)
        for (n=0;n <=50;n++);
    }
```



### 7.8.2 arquivo serialconfig.c

```
#include "MOTORDEPASSO.h"

void serial_config(void){
    TMOD=0x20;
    SCON=0x50;
    TH1=253;
    TL1=253;
    TR1=1;
    REN=1;
}

void printf(unsigned char *frases){
    unsigned int i=0;
    do{
        SBUF=*frases++;
        while(!TI);
        TI=0;
    }while(*frases);
}

#include "MOTORDEPASSO.h"

void serial_interrupt(void) __interrupt(4){
    RI=0;
    escolha=SBUF;
    int_atendida=1;
    P1++;
}

void interrupt_config(void){
    EA=1;
    ES=1;
}
```

### 7.8.3 arquivo MOTORDEPASSO.h

```
#ifndef b
#define b

#include <mcs51reg.h>

void serial_config(void);
void printf(unsigned char *);
void interrupt_config(void);
void serial_interrupt(void) __interrupt(4);
void delay33ms(void);
void delay(void);

extern volatile unsigned int int_atendida;
extern volatile unsigned char escolha;
```



```
extern __code unsigned char frase1 [];
extern __code unsigned char frase2 [];
extern __code unsigned char frase3 [];
extern __code unsigned char frase4 [];
extern __code unsigned char frase5 [];
extern __code unsigned char frase6 [];
extern __code unsigned char frase7 [];
extern __code unsigned char frase8 [];
extern __code unsigned char frase9 [];
```

```
#endif
```



## 7.9 Termômetro Indefinido

### 7.9.1 arquivo main.c

```
include "TERMOMETRO.h"

volatile unsigned char temp;
unsigned short int seg_nibble;

//interrupt o do timer0
void timer0_inter(void) __interrupt(1){
clock=!clock;
TH0=0xff;
TL0=0xfe;
}

//interrupt o externa 0 (pino int0)
void external_inter(void) __interrupt(0)
{
    temp=P1;
}

void main(void)
{ unsigned char aux;

init_lcd();
printf_lcd("Temperatura");
timer_interrupt_config();

while(1){

tempo_de_pulsos();
pulse_start();

while(!eoc);
EX0=0;
aux=temp;
temp=hex_asc_converter(temp);
seg_nibble=0;
pos_cursor(0x40);
write_caracteres(temp);

temp=hex_asc_converter(aux);
seg_nibble=1;
write_caracteres(temp);
EX0=1;
}
}

unsigned char hex_asc_converter(signed char dph_dpl_hex_asc){

if(seg_nibble==1){
```



```
dph_dpl_hex_asc=dph_dpl_hex_asc>>4;
dph_dpl_hex_asc=(dph_dpl_hex_asc & 0x0f);
}
else
    dph_dpl_hex_asc=(dph_dpl_hex_asc & 0x0f);

    if( dph_dpl_hex_asc < 0x0a)
        dph_dpl_hex_asc+=0x30;

    else
        dph_dpl_hex_asc+=0x37;
    return(dph_dpl_hex_asc);}

void pulse_start(void){
    unsigned int j;
    start=0;
    for(j=0;j<28;j++);
    start=1;
}

void tempo_de_pulsos(void){
    unsigned int i;
    for(i=0;i<400;i++);
}

#include "TERMOMETRO.h"

void init_lcd(void){
    write_instrucoes(0x32);
    write_instrucoes(0x28);
    write_instrucoes(0x0E);
    write_instrucoes(0x06);
    write_instrucoes(0x01);
}

void write_instrucoes(unsigned int instrucoes){
    unsigned int aux_inst;
    E=1;
    RS=0;
    RW=0;
    aux_inst=instrucoes;
    instrucoes=(instrucoes & 0xf0);
    P0=instrucoes;
    E=0;
    wait_lcd();
    E=1;
    instrucoes=aux_inst;
    instrucoes=(instrucoes & 0x0f);
    instrucoes=instrucoes<<4;
    P0=instrucoes;
```



```
E=0;
wait_lcd ();
E=1;
}

void write_caracteres(unsigned char caracter){
    unsigned char aux_carac;
    E=1;
    RS=1;
    RW=0;
    aux_carac=caracter;
    caracter=(caracter & 0xf0);
    P0=caracter;
    E=0;
    wait_lcd ();
    E=1;
    caracter=aux_carac;
    caracter=(caracter & 0x0f);
    caracter=caracter <<4;
    P0=caracter;
    E=0;
    wait_lcd ();
}

void pos_cursor(unsigned int pos){
    pos+=0x80;
    write_instrucoes(pos);
}

void printf_lcd(unsigned char *frases){
    do{
        write_caracteres(*frases++);
    }while(*frases);
}

void wait_lcd(void){
    unsigned long int j;
    for(j=0; j<=50; j++);
}
```

### 7.9.2 arquivo timerinterruptconfig.c

```
include "TERMOMETRO.h"

void timer_interrupt_config(void){
    EA=1;
    EX0=1;
    ET0=1;
    TH0=0xff;
    TH1=0xfe;
    TR0=1;
```



```
PT0=1;  
IT0=1;  
}
```

### 7.9.3 arquivo TERMOMETRO.h

```
#ifndef b  
#define b  
  
#include <mcs51reg.h>  
//define LCD  
#define RS P2_0  
#define RW P2_1  
#define E P2_2  
  
//define conversor AD.  
#define clock P0_0  
#define eoc P3_2  
#define start P0_1  
  
  
void external_inter(void) __interrupt(0);  
void timer0_inter(void) __interrupt(1);  
void timer_interrupt_config(void);  
void init_lcd(void);  
void write_instrucoes(unsigned int);  
void write_caracteres(unsigned char);  
void pos_cursor(unsigned int);  
void printf_lcd(unsigned char *);  
void wait_lcd(void);  
void timer_interrupt_config(void);  
void pulse_start(void);  
void tempo_de_pulos(void);  
unsigned char hex_asc_converter(signed char);  
#endif
```



## 7.10 Termômetro com Tabela

### 7.10.1 Arquivo main.c

```
#include "TERMOMETRO.h"

volatile short int temp;
unsigned short int seg_nibble;

--code unsigned char *vec[0x1f]={"-15", "-14", "-13", "-12", "-11",
"-10", "-9", "-8", "-7", "-6", "-5", "-4", "-3", "-2", "-1", "0", "1", "2",
"3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15"};
```

```
void main(void)

{unsigned short int lim_sup=1;
 unsigned int u;
 init_lcd();
 printf_lcd("Temperatura");
 timer_interrupt_config();

while(1){

tempo_de_pulos();
pulse_start();

while(!eoc);
for(u=0;u<10000;u++);
EX0=0;

if(temp>=0x1f){
    if(lim_sup){
        write_instrucoes(0x01);
        lim_sup=0;
        pos_cursor(0x00);
        printf_lcd("LIMITE-SUPERIOR");
    }
else{
    lim_sup=1;
    write_instrucoes(0x01);
    pos_cursor(0x00);
    printf_lcd("Temperatura");
    pos_cursor(0x40);
    printf_lcd(vec[temp]);
    printf_lcd(" ºCelsius");
}
}

void pulse_start(void){
    unsigned int j;
```



```
start=0;
for(j=0;j <28;j++);
start=1;
}

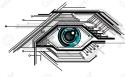
void tempo_de_pulsos(void){
    unsigned int i;
    for(i=0;i <400;i++);
}

#include "TERMOMETRO.h"

void init_lcd(void){
    write_instrucoes(0x32);
    write_instrucoes(0x28);
    write_instrucoes(0x0E);
    write_instrucoes(0x06);
    write_instrucoes(0x01);
}

void write_instrucoes(unsigned int instrucoes){
    unsigned int aux_inst;
    E=1;
    RS=0;
    RW=0;
    aux_inst=instrucoes;
    instrucoes=(instrucoes & 0xf0);
    P0=instrucoes;
    E=0;
    wait_lcd();
    E=1;
    instrucoes=aux_inst;
    instrucoes=(instrucoes & 0x0f);
    instrucoes=instrucoes<<4;
    P0=instrucoes;
    E=0;
    wait_lcd();
    E=1;
}

void write_caracteres(unsigned char caracter){
    unsigned char aux_carac;
    E=1;
    RS=1;
    RW=0;
    aux_carac=caracter;
    caracter=(caracter & 0xf0);
    P0=caracter;
    E=0;
    wait_lcd();
    E=1;
```



```
caracter=aux_carac;
caracter=(caracter & 0x0f);
caracter=caracter<<4;
P0=caracter;
E=0;
wait_lcd ();
}

void pos_cursor(unsigned int pos){
    pos+=0x80;
    write_instrucoes(pos);
}

void printf_lcd(unsigned char *frases){
    do{
        write_caracteres(*frases++);
    }while(*frases);
}

void wait_lcd(void){
    unsigned long int j;
    for(j=0; j<=50; j++);
}
```

#### 7.10.2 arquivo interruptconfig.c

```
include "TERMOMEIRO.h"

void timer_interrupt_config(void){
    EA=1;
    EX0=1;
    ET0=1;
    TH0=0xff;
    TH1=0xfe;
    TR0=1;
    PT0=1;
    IT0=1;
}

//interrupto do timer0
void timer0_inter(void) __interrupt(1){
    clock=!clock;
    TH0=0xff;
    TL0=0xfe;
}

//interrupto externo 0 (pin 0 int0)
void external_inter(void) __interrupt(0)
{
    temp=P1;
}
```



### 7.10.3 arquivo TERMOMETRO.h

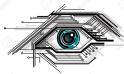
```
#ifndef b
#define b

#include <mcs51reg.h>
//define LCD
#define RS P2_0
#define RW P2_1
#define E  P2_2

//define conversor AD.
#define clock P0_0
#define eoc    P3_2
#define start  P0_1

extern volatile short int temp;

void external_inter(void) __interrupt(0);
void timer0_inter(void) __interrupt(1);
void timer_interrupt_config(void);
void init_lcd(void);
void write_instrucoes(unsigned int );
void write_caracteres(unsigned char );
void pos_cursor(unsigned int );
void printf_lcd(unsigned char *);
void wait_lcd(void);
void timer_interrupt_config(void);
void pulse_start(void);
void tempo_de_pulsos(void);
unsigned char hex_asc_converter(signed char );
#endif
```



## 7.11 Termômetro com Polinômio

### 7.11.1 Arquivo main.c

```
#include "TERMOMETRO.h"

volatile unsigned char temp;
unsigned short int seg_nibble;
signed short int negativo;

void main(void)

{
    unsigned char poly2 ,poly3 ;
    init_lcd();
    printf_lcd("Temperatura");
    timer_interrupt_config();

    while(1){

        tempo_de_pulsos();
        pulse_start();

        while (!eoc);
        EX0=0;

        if (temp>0x1f){
            write_instrucoes(0x01);
            pos_cursor(0x00);
            printf_lcd("LIMITE-SUPERIOR");}

        else{
            write_instrucoes(0x01);
            pos_cursor(0x00);
            printf_lcd("Temperatura");
            pos_cursor(0x40);
            poly2=polynomio(temp);

            if (negativo){
                printf_lcd("-");
                poly3=poly2;
                seg_nibble=1;
                poly2=0xff-poly2;
                poly2=dec_asc_converter(poly2);

                write_caracteres(poly2);
                seg_nibble=0;
                poly3=0xff-poly3;
                poly2=dec_asc_converter(poly3);
                write_caracteres(poly2);}

            else{
                poly3=poly2;
```



```
    seg_nibble=1;
    poly2=dec_asc_converter(poly2);
    write_caracteres(poly2);
    seg_nibble=0;
    poly2=dec_asc_converter(poly3);
    write_caracteres(poly2);}
}

EX0=1;
}

void pulse_start(void){
    unsigned int j;
    start=0;
    for (j=0;j<28;j++);
    start=1;
}

void tempo_de_pulsos(void){
    unsigned int i;
    for (i=0;i<400;i++);
}

unsigned char polynomio(volatile unsigned char temp2)
{
    signed char poly;

    poly= temp2-16;

    if (poly<0)
        negativo=1;

    else
        negativo=0;

    poly=(unsigned char)poly;

    return (poly);}

unsigned char dec_asc_converter(unsigned char poly4){

    if (seg_nibble==1){
        poly4=(poly4 & 0xf0);
        poly4=poly4>>4;
    }
    else
        poly4=(poly4 & 0x0f);
```



```
    if ( poly4<0x0a )
        poly4+=0x30;

    else
        poly4+=0x37;

    return( poly4 );}
```

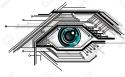
### 7.11.2 arquivo TERMOMETRO.h

```
#include "TERMOMETRO.h"

void init_lcd(void){
    write_instrucoes(0x32);
    write_instrucoes(0x28);
    write_instrucoes(0x0E);
    write_instrucoes(0x06);
    write_instrucoes(0x01);
}

void write_instrucoes(unsigned int instrucoes){
    unsigned int aux_inst;
    E=1;
    RS=0;
    RW=0;
    aux_inst=instrucoes;
    instrucoes=(instrucoes & 0xf0);
    P0=instrucoes;
    E=0;
    wait_lcd();
    E=1;
    instrucoes=aux_inst;
    instrucoes=(instrucoes & 0x0f);
    instrucoes=instrucoes<<4;
    P0=instrucoes;
    E=0;
    wait_lcd();
    E=1;
}

void write_caracteres(unsigned char caracter){
    unsigned char aux_carac;
    E=1;
    RS=1;
    RW=0;
    aux_carac=caracter;
    caracter=(caracter & 0xf0);
    P0=caracter;
    E=0;
    wait_lcd();
    E=1;
}
```



```
caracter=aux_carac;
caracter=(caracter & 0x0f);
caracter=caracter<<4;
P0=caracter;
E=0;
wait_lcd ();
}

void pos_cursor(unsigned int pos){
    pos+=0x80;
    write_instrucoes(pos);
}

void printf_lcd(unsigned char *frases){
    do{
        write_caracteres(*frases++);
    }while(*frases);
}

void wait_lcd(void){
    unsigned long int j;
    for(j=0; j<=50; j++);
}
```

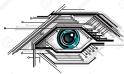
### 7.11.3 arquivo serialtimerconfig

```
#include "TERMOMEIRO.h"

void timer_interrupt_config(void){
    EA=1;
    EX0=1;
    ET0=1;
    TH0=0xff;
    TH1=0xfe;
    TR0=1;
    PT0=1;
    IT0=1;
}

//interrupto do timer0
void timer0_inter(void) __interrupt(1){
    clock=!clock;
    TH0=0xff;
    TL0=0xfe;
}

//interrupto externo 0 (pino int0)
void external_inter(void) __interrupt(0)
{
    temp=P1;
```



## 8 Apêndice2: Simulação no Proteus do Sistema Motor DC

### 9 Revisão de trabalhos feito no Proteus

Apesar não ser tão extensa quanto a bibliografia de trabalhos que usam o matlab, o proteus tem uma gama de artigos publicados que fazem o seu uso considerável, principalmente quando há a utilização de microcontroladores no sistema em questão, afinal uma grande potencialidade do proteus é a presença de vários microcontroladores e periféricos, sendo que alguns artigos defendem o uso do proteus como ferramenta para o ensino de microcontroladores [1][2]. A presença de vários tipos de motores provem a possibilidade de simula-los, como pode ser visto no seguinte trabalho[3]. Controle de cargas e sistemas de potência baseado em microcontroladores PIC (Programmable Integrate Circuit) também encontram-se desenvolvidos[4]. Simulação do uso de pesticidas emitidos por meio de circuitos microcontrolados[5]. A simulação e construção de display de leds em caracteres de chineses também já foi implementada [6] e outras utilidades utilizando displays [7].

Trabalhos mais impactantes também foram feitos, que utilizam praticamente somente o proteus como software de simulação, como exemplo tem-se a construção de um microcomputador [8], o controle de carga (explicar melhor) [9]. A construção de um termostato [10] o controle de temperatura e humidade baseado no proteus [11] avanços na tecnologia de pizoeletroico [12], a construção de um voltímetro digital por meio de microcontroladore [13], busca de máxima potência utilizando sistemas microcontrolados [4] e muitos outros trabalhos que utilizam diretamente e indiretamente o proteus como simulador.

## 10 Simulações em Malha aberta

Primeiramente as simulações visam verificar os resultados dos softwares , proteus e matlab, com relação aos dados reais e a ambos e entre si. Na figura (??) encontra-se os resultados da simulação em malha aberta feita com a função de transferência do motor,bem como a curva obtida experimentalmente e, na figura (??) encontra-se o erro obtido em relação as duas simulações com os dados obtidos experimentalmente.Como pode-se observar o erro é tolerante em ambas as simulações.

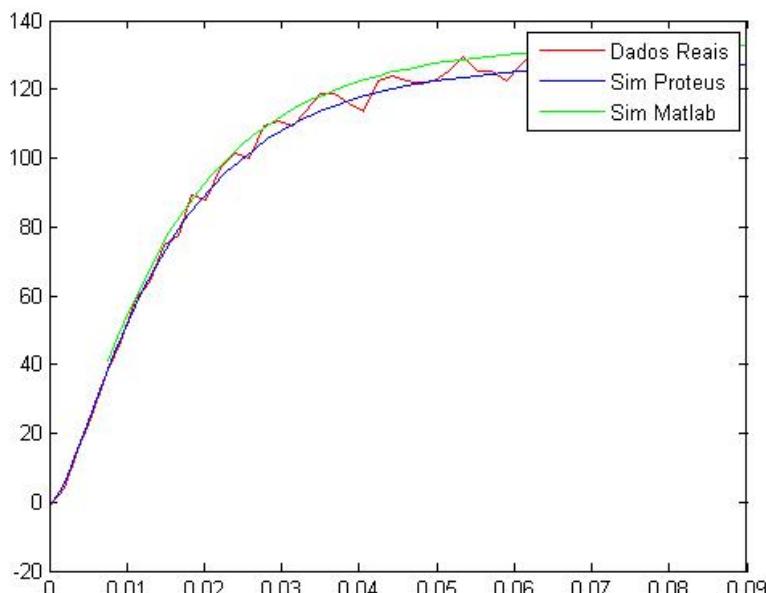


Figura 31: Dados Reais , Simulação Proteus e Matlab

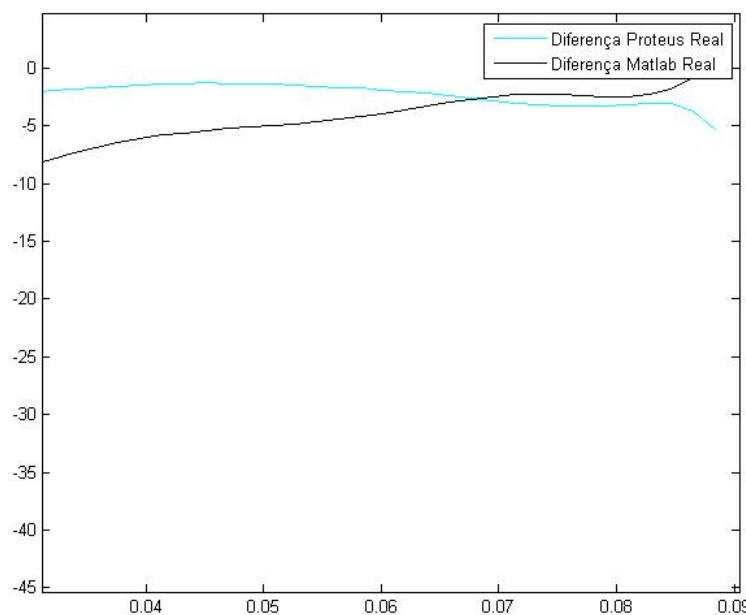
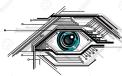


Figura 32: Erro Com Relação aos Dados Reais da Simulação no Proteus e Matlab em Malha Aberta

Assim, verifica-se que ambas as simulações obtiveram erros razoáveis com os dados obtidos experimentalmente. Porém, cabe salientar que foi usado no proteus os blocos de função de transferência, que não é seu ponto forte, afinal não há uma grande gama de funções. No que diz respeito a simulação do controle do motor em malha fechada, caberá o uso de componentes reais do Proteus, sendo este seu ponto forte. Isto é conteúdo dos próximos tópicos.



## 11 Simulações do Controle em Malha Fechada.

A simulação foi realizada no software Proteus utilizando os componentes e resistências que compõem o controlador, utilizando blocos para representar o PWM e o ganho do tacogerador, além da planta do motor.A figura (??) ilustra a construção feita.

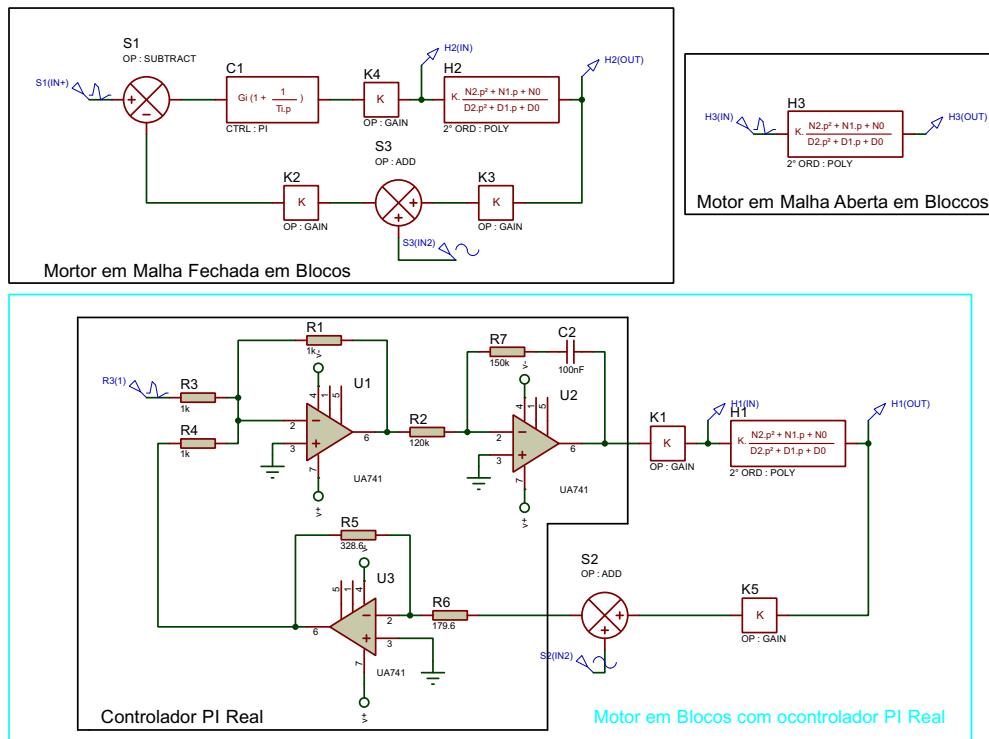


Figura 33: Estrutura Simulada no Proteus com os Componentes Reais do Controlador

Foram feitas as simulações nos softwares Matlab e Proteus e plotadas juntamente com o intuito de verificar a precisão das simulações nos softwares, principalmente no proteus, por ser de menor reputação.A resposta real também foi plotada para verificar a precisão.Tal construção de simulações encontra-se na figura abaixo (??).

Por meio das comparações feitas entre os simuladores e dados obtidos no laboratório, verifica-se que são confiáveis as simulações no proteus para blocos de controle e amp-ops.

## 12 Simulação PWM no Proteus e Construção da Placa de Controle

O PWM é uma construção essencial no controle do motor, e seu funcionamento, no quesito de componentes analógicos, se da por meio da comparação da forma de onda de uma dente de serra com uma tensão de referência.Conforme varia-se a tensão de referência varia-se o duty cycle do PWM.A figura (??) explicita o circuito do PWM utilizado, sendo que a figura (??) expressa a simulação do componente.A simulação foi realizada com uma tensão de entrada no valor de 5,96 Volts, gerando o duty cycle precisamente como mostra a tabela (??) que é de valor 52us.

Como foi dito, o software Proteus possibilita a construção da placa de controle, além de sua simulação.Assim, a construção do projeto se dá quando todas as simulações estarem batendo com os resultados esperados, expresando essa vantagem como explícito na figura (??).O circuito em sua forma de simulação e renderização para o circuito real encontram-se nas figuras (??), (??) abaixo.Com esta placa o controle do motor pode ser feito apenas conectando os elementos expressos na figura da placa

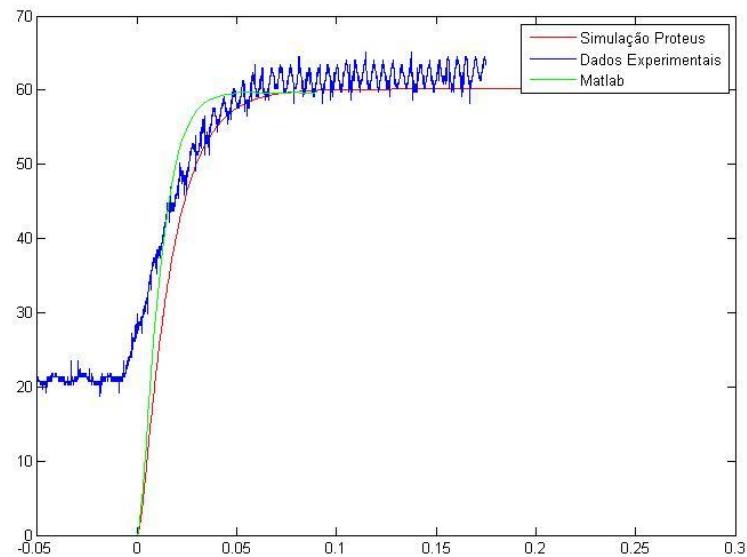


Figura 34: Estrutura Simulada no Proteus e Matlab e Dados Obtidos Experimentalmente

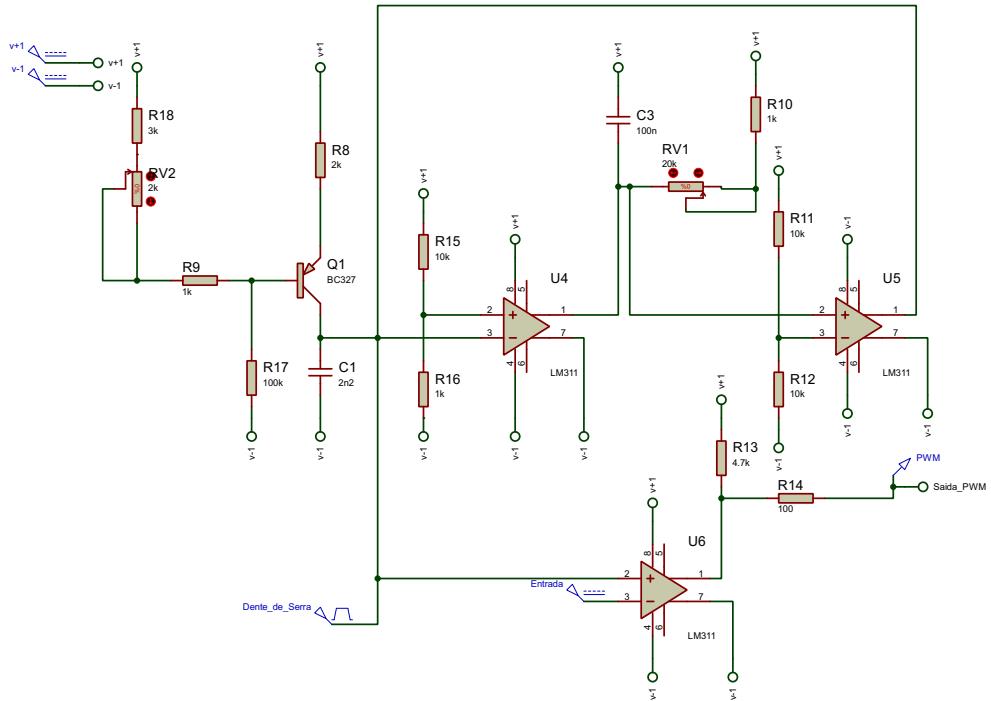


Figura 35: Circuito PWM

real renderizada, sendo que TC é a saída do Tacogerador, +15V o 15 volts e -15V o menos 15 volts,  $D_S$  a entrada dente de serra, se não se desejar colocar um componente que gere tal forma de onda na placa como o LM555 que é um dos CI's capaz de gerar essa forma de onda, IN é tensão de referência para o PWMS e a saída do PWM que vai para o terminal do motor.

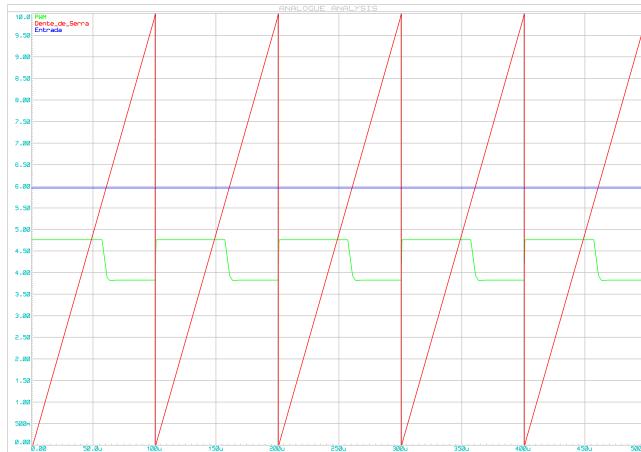


Figura 36: Simulação do Circuito PWM da Figura Acima.

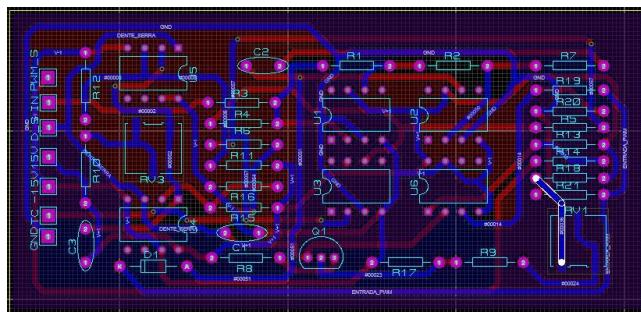


Figura 37: Imagem na Forma de PCB Construída no Software Proteus Ares

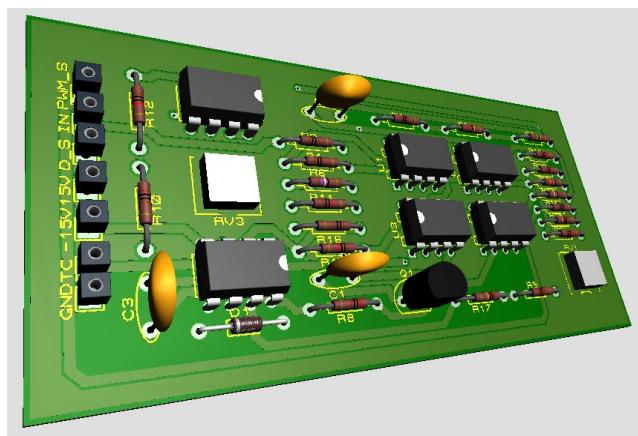
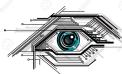


Figura 38: Placa Renderizada do Circuito PCB Acima.

### 13 Conclusão Simulação Proteus

Como pode ser visto pela comparação das simulações, observa-se que o proteus possui grande fidelidade quanto a simulações, havendo pequena diferença entre a simulação no matlab, dados reais. O fato de poder-se simular e construir o circuito simulado é uma grande vantagem, afinal garante-se que o sistema terá o funcionamento desejado. Assim, conclui-se que tal software é um bom simulador para auxiliar em projetos de controle.



## Referências

- [1] Li Wang Bo Su. Application of proteus virtual system modelling (vsm) in teaching of microcontroller. *International Conference on E-Health Networking, Digital Ecosystems and Technologies*, pages 375–378, 2010.
- [2] Dražen Cika and Darko Grundler. Proteus virtual system modelling used for microcontroller education. *MIPRO*, pages 1034–1038, 2010.
- [3] Dr. Dahaman Ishak Mohamad Nasrul Abdul Satar. Application of proteus vsm in modelling brushless dc motor drives. *International Conference on Mechatronics*, 2011.
- [4] Anwar Shabir Ahsan Shahid. Microchip based embedded system design for achievement of high power factor in electrical power systems. *IEEE*.
- [5] Ji Na Liu Wei Zhou Ying-chang Sun Rong-xia, Guo Liang. The simulation design of the ozonizer base on proteus. *The 1st International Conference on Information Science and Engineering*, pages 5265–5267.
- [6] Guo Jinying Ding Yanchuang. Led display screen design and proteus simulation based on single-chip microcomputer. *IEEE*.
- [7] LI Zong-qiang LIU Li-jun. Design and simulation of led clock circuit based on proteus. *International Conference on Computer, Mechatronics, Control and Electronic Engineering*, pages 315–316, 2010.
- [8] W. Chen S. K. Tolpygo J. R. Friedman, V. Patel and J. E. Lukens. The construction of single-chip microcomputer virtual experiment platform based on proteus. *The 5th International Conference on Computer Science and Education*, 2010.
- [9] Ahmad Shukri Bin Fazil Rahman and Abdul Rahim Bin Abdul Raza. Proteus based simulation of a charge controller. *IEEE Conference on Power and Energy*, pages 539–543, 2010.
- [10] Song Kefei HAN Zhenwei. Design of thermostat system based on proteus simulation software. *International Conference on Electronic and Mechanical Engineering and Information Technology*, pages 1901–1904, 2011.
- [11] Pan Jinfeng i Xu Xiumei. The simulation of temperature and humidity control system based on proteus. *International Conference on Mechatronic Science, Electric Engineering and Computer*, pages 1896–1898, 2011.
- [12] Rasli Abd Ghani Noorradiyah Ismail. Advance devices using piezoelectric harvesting energy. *IEEE Student Conference on Research and Developmentr*, pages 451–453, 2013.
- [13] Sarvesh SS Rawat Karthik Srinivasam Vikram Puri Jartin Aroram, Gagandeep. Design and develepmment of digital voltmeter using differents techniques.