A família de microcontroladores 8051

1 Comentários sobre microcontroladores

Um microcontrolador é um componente que tem, num único chip, além de uma CPU, elementos tais como memórias ROM e RAM, temporizadores/contadores, PWM, conversor AD, canais de comunicação e conversores analógico-digitais. Esta característica diferencia os sistemas baseados em microcontroladores daqueles baseados em microprocessadores, onde normalmente se utilizam vários componentes para implementar essas funções. Com isso, os microcontroladores permitem a implementação de sistemas mais compactos e baratos do que aqueles baseados em microprocessadores.

Em contrapartida, as CPUs dos microcontroladores são, menos poderosas do que os microprocessadores. Seu conjunto de instruções costuma se limitar às instruções mais simples, sua freqüência de clock é mais baixa e o espaço de memória endereçável costuma ser bem menor. Vê-se daí que o campo de aplicação dos microcontroladores é diferente daquele dos microprocessadores, e que um sistema que possa ser controlado por um microcontrolador tende a ter menor complexidade e menor custo do que um sistema que exija a capacidade de processamento de um microprocessador.

Exemplos de sistemas onde os microcontroladores encontram aplicação incluem controle de semáforos, balanças eletrônicas, microterminais, telefones públicos, controle de carregadores de baterias, inversores eletrônicos, controles de acesso, taxímetros, sistemas de aquisição de dados de manufatura e eletrodomésticos em geral.

A programação dos microcontroladores é mais simples do que a dos microprocessadores. Isto acontece porque os periféricos *on-chip* dos microcontroladores são acessados de uma forma padronizada e integrada na própria linguagem de programação, dispensando o conhecimento de detalhes externos. Não se deve pensar, porém, que isto simplifique a tarefa do programador em todos os níveis: é necessário que ele conheça bem o hardware conectado ao microcontrolador para poder produzir programas que funcionem corretamente.

Cabe citar ainda uma vantagem particular dos microcontroladores que possuem memória ROM, que é a possibilidade de armazenar programas internamente, dificultando sensivelmente a cópia ilícita do código.

1.1 A família 8051 introduzida pela Intel

Diversos fabricantes produzem microcontroladores da família 8051 (Intel, AMD, Atmel, Dallas, OKI, Matra, Philips, Siemens, SMC, SSI). A Intel iniciou a produção do 8051 em 1981. Em 1982 foram produzidos 2 milhões de unidades, em 1985 foram 18 milhões e em 1993, 126 milhões. A tendência atual é uma participação crescente dos microprocessadores de 8 bits e uma diminuição da fatia de mercado dos microcontroladores de 4 bits.

Além do 8051 propriamente dito, existem variantes como o 8031 (sem memória ROM interna e com apenas 128 bytes de memória RAM), o 8751 (4 kB de memória EPROM) e o 8052 (8 kB de memória ROM, um terceiro timer e 256 bytes de memória RAM). A menos dessas diferenças, os modelos citados são idênticos, e o texto utilizará o termo "8051" de forma genérica, citando as outras versões

apenas onde for necessário. Informações complementares podem ser obtidas em [Inte89], [Siem90] e [Silv94]. : http://www.ustr.net/

1.2 Principais características

- Freqüência de clock de 12 MHz, com algumas versões que alcançam os 40 MHz;
- até 64 kB de memória de dados externa;
- 128 bytes de RAM interna;
- até 64 kB de memória de programa configurável de duas formas mutuamente excludentes:
  - ◊ 4 kB internos (ROM no 8051 e EPROM no 8751) e mais 60 kB externos;
  - ◊ 64 kB externos;
- 4 portas bidirecionais de I/O, cada uma com 8 bits individualmente endereçáveis; duas dessas portas (P0 e P2) e parte de uma terceira (P3) ficam comprometidas no caso de se utilizar qualquer tipo de memória externa;
- 2 temporizadores /contadores de 16 bits;
- 1 canal de comunicação serial;
- 5 fontes de interrupção (dois timers, dois pinos externos e o canal de comunicação serial) com 2 níveis de prioridade selecionáveis por software;
- oscilador de clock interno.

As características citadas são básicas e formam o núcleo da família 8051, que pode ser acrescido de uma ou mais das características especiais mostradas na figura 0.2.

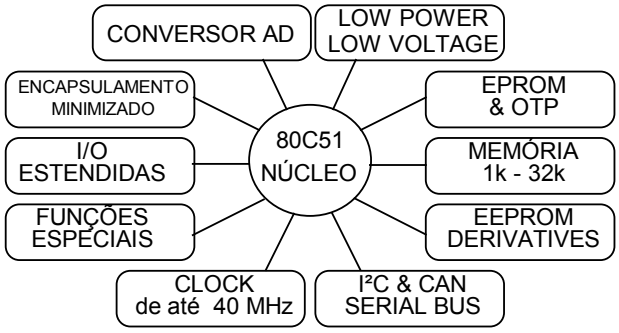


Fig. 0.2 - Características especiais da família 8051

A título de exemplo, a tabela 0.2 apresenta as características de alguns componentes da família 8051 com a indicação de seus fabricantes. O número total de variantes é muito maior. A Philips, por exemplo, produz mais de 40 tipos diferentes.

Tipo	Pinos	Fabr.	RAM	CODE	Notes (LV - low voltage)
MCS251	40	Intel	1K	16K	16 Bit 80x51FX! Preliminar
80C517A	84	Siemens	256	64Kx	ALU;8PWM;CC;2UART;10bA/D
80C537A	84	Siemens	256	32K	ALU;8PWM;CC;2UART;10bA/D
80C535A	68	Siemens	256	64Kx	515+10bA/D;1K XRAM;BRG;OWD
80C515A	68	Siemens	256	32K	515+10bA/D;1K XRAM;BRG;OWD
80535	68	Siemens	256	64Kx	Timer2CaptComp 6ports 8/10bA/D
80515	68	Siemens	256	8K	Timer2 CaptComp 4 ports 8b A/D
80C535	68	Siemens	256	64Kx	Timer2 CaptComp 5 ports 8b A/D
80C51GB	68	Intel	256	64Kx	8051FA+PCA; 8b A/D; SPI
87C51GB	68	Intel	256	8K	8051FA+PCA; 8b A/D; SPI
80C592	68	Philips	256	64Kx	552-I2C+CAN+XRAM
87C598	80	Philips	256	32K	552-I2C+CAN+XRAM
80C552	68	Philips	256	64Kx	10b A/D; I2C; CaptComp; PWM
87C552	68	Philips	256	8K	10b A/D; I2C; CaptComp; PWM

Tab. 0.2 - Alguns integrantes da família MCS51

1.3 Análise externa

O aspecto externo do 8051 é o da figura 0.4. Os pinos com nomes da forma P0.0, P0.1, etc. correspondem às quatro portas de E/S (P0 a P3). Note a dupla utilidade das portas P0 e P2, que ficam comprometidas com o uso de memória externa, assim como os pinos P3.6 e P3.7. O sinal ALE (Address Latch Enable) permite fazer a demultiplexação de dados e endereços na porta P0, conforme mostra o esquema da figura 1.4.

Através do sinal $\overline{\text{PSEN}}$ (program storage enable), o controlador informa o mundo externo se a operação em andamento é uma leitura de instrução (acesso à memória de programa) ou um acesso à memória de dados. Este sinal permite que o processador tenha duas regiões distintas de memória externa, uma para armazenar código e outra para dados. Ambas ocupam os endereços de 0 a FFFFH (64 kB), num total de 128 kB.

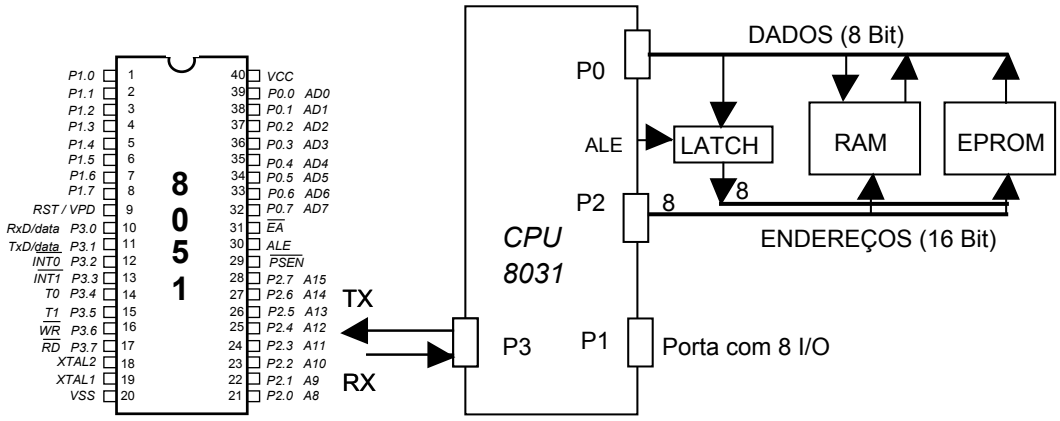


Fig. 0.4 - Aspecto externo do 8051

O pino $\overline{\text{EA}}$ é um sinal de entrada, através do qual o usuário escolhe se será utilizada a memória ROM interna ou se todo o programa será armazenado externamente.

Os pinos da porta P3 também são utilizados para realizar certas funções especiais:

- P3.0 - RxD/data - recepção serial assíncrona ou E/S de dados síncronos;
- P3.1 - TxD/clock - transmissão porta serial assíncrona ou saída de clock p/ dados síncronos;
- P3.2 - INT0 - entrada da interrupção 0 ou bit de controle para o temporizador/contador 0;
- P3.3 - INT1 - entrada da interrupção 1 ou bit de controle para temporizador/contador ;
- P3.4 - T0 - entrada de clock externo para o temporizador/contador 0;
- P3.5 - T1 - entrada de clock externo para o temporizador/contador 1;
- P3.6 - WR - sinal de escrita na memória de dados externa;
- P3.7 - RD - sinal de leitura na memória de dados externa.

A alimentação (5V) é feita pelo pino 40 e o GND é o pino 20; o cristal para o oscilador interno é conectado aos pinos 18 e 19. Finalmente, o pino 9, RST/VPD, é a entrada de reset.

1.4 Modelo de programação simplificado

Desenhar um modelo de programação completo para um microcontrolador é mais difícil do que para um microprocessador, porque os microcontroladores têm, em geral, um número bem maior de registradores, que servem para acessar os componentes periféricos *on-chip*. O modelo da figura 0.6 contempla apenas os registradores de caráter geral.

O registrador A é o acumulador que, como no 8085, é responsável pelas principais operações, sobretudo as lógicas e aritméticas. B é um registrador de caráter geral, assim como os oito registradores R0 a R7. DH e DL também são de uso geral, mas podem ser utilizados como um registrador de 16

bits, que se denomina DPTR. Este registrador é o único que pode conter valores de 16 bits e por isso é freqüentemente utilizado no endereçamento da memória externa, que é sempre indireto. PC contém o endereço da próxima instrução executável e SP aponta para o topo da pilha.

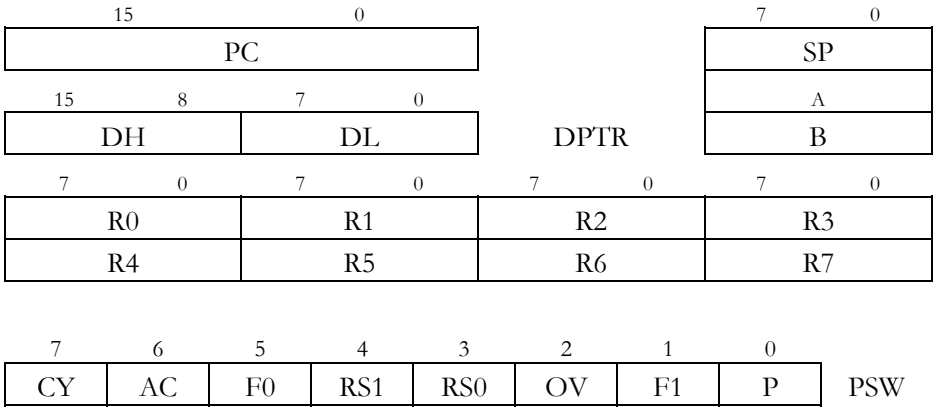


Fig. 0.6 - Registros internos do 8051

O 8051 conta com quatro flags de sistema (carry, auxiliary carry, overflow e parity) e dois flags de usuário (F0 e F1), que o programador pode utilizar como desejar. Os flags estão todos reunidos no registrador PSW *(program status word),* que contém ainda os bits de seleção de banco de registradores, RS0 e RS1. Estes últimos são descritos na seção 0.12.6, que também apresenta mais detalhes sobre os registradores R0 a R7.

Os registradores de controle dos periféricos *on-chip,* que não aparecem no modelo acima, serão detalhados nas seções subseqüentes.

1.5 Organização da memória de programa

A memória de programa do 8051 pode ocupar até 64 kB. Aqui há duas opções: se a memória ROM interna do controlador for utilizada, então esta será mapeada nos primeiros 4 kB deste espaço de endereçamento (endereços 0000H a 0FFFH), e os demais 60 kB (1000H a FFFFH) serão externos. Caso não se deseje utilizar a ROM interna, então toda a memória será externa.

A escolha é feita por hardware, de acordo com o nível de tensão aplicado ao pino EA. Nível lógico zero seleciona memória de programa externa para todos os endereços; com nível 1, as instruções são lidas da memória interna se seu endereço for menor do que 1000H (primeiros 4 kB).

1.6 Organização da memória de dados

1.6.1 Memória de dados interna

A memória interna é dividida em três blocos fisicamente distintos, conforme ilustra a figura 0.8. Há dois blocos de RAM, de 128 bytes cada, mapeados nos endereços 00H-7FH e 80H-FFH, e mais cerca de 20 registradores de funções especiais, espalhados em endereços da faixa 80H-FFH. A distinção entre os dois blocos cujos endereços coincidem é feita pelos modos de endereçamento, também indicados na figura.

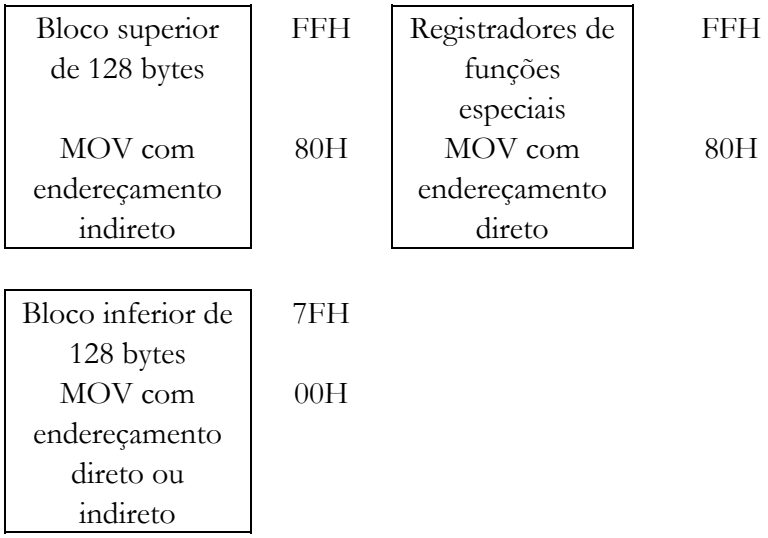


Fig. 0.8 - Organização da memória de dados interna do 8051

1.6.2 Memória de dados externa

A memória de dados externa pode ocupar até 64 kB. Portanto, há coincidências de endereços com toda a faixa de memória de programa e com os endereços 00H a FFH da memória de dados interna. Estes conflitos são resolvidos por dois mecanismos distintos:

1. conforme mencionado na seção 1.3, a distinção entre *memória de programa* e *memória de dados* é feita pelo sinal PSEN, que o hardware utiliza para habilitar o banco de memória correspondente. Este sinal é ativado na leitura de instruções da memória na utilização da instrução MOVC, transfere dados entre registradores e a memória de programa;
2. para distinguir entre *memória de dados interna* e *memória de dados externa*, o microcontrolador oferece duas instruções distintas: MOV, para acessar a memória interna, e MOVX (move external), para a memória externa. É interessante notar que esta última instrução exige sempre endereçamento indireto. Por exemplo, a instrução

   ```
   MOV 20H,A
   ```

   armazena na posição 20H da memória RAM interna o conteúdo do acumulador. Por outro lado, para armazenar o mesmo conteúdo na posição 20H da memória de dados externa, o procedimento seria:

   ```
   MOV R0,#20H              ;coloca o endereço em R0
   MOVX @R0,A               ;copia A para a posição em R0
   ```

1.6.3 Mais detalhes sobre a RAM interna

Os primeiros 48 bytes da RAM interna (00H a 2FH) apresentam ainda algumas particularidades, conforme mostra a figura 0.10. Os endereços 00H a 1FH compreendem quatro bancos de oito registradores cada. Em cada banco, os registradores são denominados R0, R1, ... R7 (os nomes são repetidos). Estes podem ser endereçados por seus nomes nas instruções (como em `MOV A,R0`), ou então diretamente através dos seus endereços (`MOV A,00H`). No primeiro caso, o processador precisa decidir ainda a qual banco a instrução se refere, por causa da repetição dos nomes. Esta *seleção de banco* é feita configurando os bits RS1 e RS0 do registrador PSW de acordo com a tabela 0.4.

7F	7E	7D	7C	7B	7A	79	78	2FH
...								...
0F	0E	0D	0C	0B	0A	09	08	21H
07	06	05	04	03	02	01	00	20H
R0 - R7 Banco 3								1FH 18H
R0 - R7 Banco 2								17H 10H
R0 - R7 Banco 1								0FH 08H
R0 - R7 Banco 0								07H 00H

Fig. 0.10 - Organização da parte baixa da RAM interna

RS1	RS0	Banco
0	0	0
0	1	1
1	0	2
1	1	3

Tab. 0.4 - Seleção dos bancos de registradores do 8051

Acima dos bancos de registradores há uma região de 16 bytes (endereços 20H a 2FH) cujos bits podem ser endereçados individualmente, através das instruções SETB (set bit), CLR (clear) e CPL (complement). Os endereços desses 128 bits vão de 00 a 7FH, começando do bit menos significativo do byte 20H e terminando no bit mais significativo do byte 2FH. A instrução **SETB 00** vai setar o bit LSB do byte da posição de memória 20H. A instrução **CLR 22** faz o que ?

1.6.4 Os registradores de funções especiais

Os registradores de funções especiais incluem posições de acesso às portas de E/S, registradores de interrupção, registradores da porta serial, temporizadores e registradores aritméticos. Aqueles situados em endereços múltiplos de 8 também podem ser endereçados bit a bit Os registradores são descritos resumidamente a seguir; os interessados em obter maiores detalhes são convidados a consultar uma das referências bibliográficas.

- P0 (80H), P1(90H), P2 (A0H) e P3(B0H) correspondem a posições de RAM contendo os dados das portas de E/S; os bits individuais são endereçados como P0.0, P0.1, etc.;
- TH1 (8DH), TL1 (8BH), TH0 (8CH) e TL0 (8AH) contêm os valores das contagens dos temporizadores/contadores 1 e 0, respectivamente;
- TCON (88H) e TMOD (89H) são os registradores de controle e modo de operação dos temporizadores/contadores;
- PCON (87H) permite adaptar o chip a uma situação na qual não há processamento, mas onde se quer manter os conteúdos das memórias internas (falha de alimentação, por exemplo);
- SCON (98H) e SBUF (99H) permitem, respectivamente, programar a porta de comunicação serial e armazenar o dado recebido ou a ser transmitido;
- IE (A8H) e IP (B8H) são registradores associados à gestão de interrupção (habilitação e prioridade);
- SP (81H), PSW (D0H), A (E0H), B (F0H), DPH (83H) e DPL (82H) foram apresentados na seção 0.8.

Endereço Endereço

F8	B								**FF**
F0	B								**F7**
E8									**EF**
E0	ACC								**E7**
D8									**DF**
D0	PSW								**D7**
C8									**CF**
C0									**C7**
B8	IP								**BF**
B0	P3								**B7**
A8	IE								**AF**
A0	P2								**A7**
98	SCON	SBUF							**9F**
90	P1								**97**
88	TCON	TMOD	TL0	TL1	TH0	TH1			**8F**
80	P0	SP	DPL	DPH				PCON	**87**

1.7 Modos de endereçamento

O conjunto de instruções oferece diversos modos de endereçamento, projetados de modo a agilizar o acesso à RAM interna e também para facilitar as operações de manipulação de bits e bytes.

1.7.1 Endereçamento imediato

O valor do operando faz parte do corpo da instrução e segue o *opcode* na memória de programa. O operando deve ser precedido do símbolo '#', a fim de evitar a confusão com o modo direto. Exemplo: MOV B,#255 - faz o registrador B igual a FFH.

1.7.2 Endereçamento direto

Neste modo, a instrução especifica o endereço do operando, que deve ser um número de 8 bits. Conforme a seção 1.6, somente dados dos primeiros 128 bytes da memória RAM interna e dos registradores de funções especiais são endereçados deste modo. Exemplo: MOV A,25H - traz para o acumulador o conteúdo do byte 25H da RAM interna.

1.7.3 Endereçamento de bits individuais

As instruções que manipulam bits individuais especificam este bit de forma direta, quer sob a forma de um número (os endereços dos bits individualmente endereçáveis da figura 0.10), quer sob a forma de uma abreviatura, como no caso das portas de E/S ou de outros bits dos registradores de funções especiais. Exemplos: SETB 0FH - seta o bit mais significativo do byte 21H da memória interna; CLR P1.0 - zera o bit menos significativo da porta 1.

1.7.4 Endereçamento indireto

A instrução especifica um registrador, cujo conteúdo é o endereço do operando. Este modo pode ser utilizado para endereçar tanto a memória interna quanto a memória externa. Se o endereço do operando for de 8 bits, os registradores que podem ser especificados no corpo da instruções são R0, R1 ou SP. Caso o endereço seja de 16 bits, o registrador a ser utilizado tem que ser DPTR. Exemplos: MOV @R1,#15H - coloca o valor 15H no byte endereçado por R1; MOV @DPTR,A - copia o valor do acumulador para o byte endereçado por DPTR.

1.7.5 Endereçamento indexado

Este modo serve apenas para endereçar a memória de programa (instrução MOVC). Nesse tipo de acesso, o endereço do operando é dado pela soma do conteúdo de um registrador de base, que pode ser DPTR ou PC, com o conteúdo do acumulador. Dessa forma, DPTR ou PC apontam para a base

de uma tabela enquanto o acumulador seleciona um elemento dentro da tabela. Exemplos: MOVC A,@A+DPTR - o acumulador indica o offset de um byte dentro da tabela que inicia em DPTR e que deve ser copiado para o acumulador; MOVC B,@A+PC - o registrador B recebe uma cópia do valor do byte que está A posições à frente do PC.

1.7.6 Endereçamento dos registradores R0 a R7

Os opcodes das instruções de acesso aos registradores R0 a R7 utilizam três dos seus oito bits para especificar o registrador endereçado. Desta forma, este modo consegue instruções mais curtas do que o modo direto, por dispensar byte com o endereço. O banco de registradores referenciado é aquele que está selecionado pelos bits RS1 e RS0 no instante em que a instrução é executada. Exemplo: MOV A,R3 - copia R3 para A.

1.7.7 Endereçamento implícito de registradores

Algumas instruções trabalham sempre com um certo registradores, e portanto nenhum byte é necessário para endereçar o operando. Exemplo: a instrução DA A atua sempre sobre o acumulador, e por isso não se pode especificar um registrador diferente de A..

1.8 Funcionamento da pilha

É importante notar que **SP é um registrador de 8 bits**, e que por isso o tamanho da pilha do 8051 é bem menor do que o das pilhas normalmente usadas com outros microprocessadores. Além disso, SP endereça somente a RAM interna.

O funcionamento da pilha também é diferente, pois aqui o armazenamento se faz no sentido dos **endereços crescentes** e os bytes são empilhados um a um, de acordo com as seguintes regras:

- instrução PUSH: SP é incrementado e o byte é escrito na posição apontada por SP;
- instrução POP: o byte da posição apontada por SP é copiado para o destino e em seguida SP é decrementado;
- chamadas de sub-rotinas e interrupções: os dois bytes que formam o endereço de retorno são empilhados, com o byte menos significativo no endereço mais baixo;
- retorno de sub-rotinas e interrupções: os dois bytes que formam o endereço de retorno são copiados para o PC.

Desta forma, o registrador **SP aponta sempre para o topo da pilha**.

O valor de SP após um reset é 07H, de modo que a pilha começa, por default, no byte de memória interna 08H, onde inicia também o banco de registradores 1. Portanto, se o programa precisar trabalhar com mais de um banco de registradores, deve escolher outro local para a pilha.

1.9 As interrupções

O 8051 tem cinco fontes de interrupção: duas entradas externas (INT0 e INT1), dois temporizadores/contadores (Timer 0 e Timer 1) e o canal de comunicação serial. Conforme descrito na seção 0.6, as entradas das interrupções externas e o canal serial ocupam pinos da porta P3.

Os endereços de desvio das interrupções são fixos e dados pela tabela 0.6.

O 8051 permite habilitar ou desabilitar cada interrupção individualmente através dos bits EX0, ET0, EX1, ET1 e ES do registrador IE (Interrupt Enable), que aparece na figura 0.12. Esses bits controlam as interrupções externa 0, do timer 0, externa 1, do timer 1 e do canal serial, respectivamente. O bit EA (Enable All) permite habilitar (1) ou desabilitar (0) todas as interrupções de uma só vez. Para que uma interrupção esteja habilitada, tanto o bit correspondente quanto EA precisam estar setados.

Interrupção	Endereço
Interrupção externa 0	0003H
Timer 0	000BH
Interrupção externa 1	0013H
Timer 1	001BH
Canal serial	0023H

Tab. 0.6 - Endereços de desvio das interrupções

7	6	5	4	3	2	1	0
EA	...	...	ES	ET1	EX1	ET0	EX0

Fig. 0.12 - O registrador IE - Interrupt **E**nable

A cada interrupção está associado um de dois níveis de prioridade (alto ou baixo), de acordo com o registrador IP (Interrupt Priority), que aparece na figura 0.14. Estes níveis são tais que um tratador de interrupção só pode ser interrompido por um pedido de interrupção de nível superior. Os bits PX0, PT0, PX1, PT1 e PS permitem setar a prioridade das interrupções associadas aos elementos INT0, Timer 0, INT1, Timer 1 e canal serial, respectivamente.

7	6	5	4	3	2	1	0
...	...	...	PS	PT1	PX1	PT0	PX0

Fig. 0.14 - O registrador IP - Interrupt **P**riority

Além disso, os bits de 0 a 3 do registrador TCON (figura 0.16) permitem programar o modo de reconhecimento das interrupções externas INT0 e INT1. Os bits IT0 e IT1 (bits 0 e 2) determinam o modo de reconhecimento das interrupções, se por nível ou por transição. Quando um destes bits está em 0, a interrupção correspondente tem garantia de ser detectada se o pino permanecer em 0 por pelo menos 12 períodos de clock; quando em 1, a interrupção correspondente acontece quando ocorre uma transição de 1 para 0. Os bits IE0 e IE1 (bits 1 e 3) são flags que sinalizam internamente os pedidos de interrupção. São colocados automaticamente em 1 quando há uma transição de 1 para 0 no pino correspondente e em 0 quando essa interrupção é atendida.

1.10 Os temporizadores / contadores

O 8051 é dotado de dois temporizadores/contadores internos programáveis, deste ponto em diante chamados simplesmente de *timers*. São controlados por software e também por sinais externos aplicados ao microcontrolador. Os timers são geralmente utilizados para a geração de eventos após decorrido um determinado intervalo de tempo *(timeouts)*, na geração de pedidos de interrupção periódicos, ou ainda na contagem e na medição da largura de pulsos externos. Os registradores associados à programação dos timers, TCON e TMOD, aparecem nas figuras 0.16 e 0.18.

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Fig. 0.16 - O registradorTCON - Timer **CON**trol

7	6	5	4	3	2	1	0
Gate1	C/T1	M1.1	M0.1	Gate0	C/T0	M1.0	M0.0

Fig. 0.18 - O registrador TMOD - Timer **MOD**es

Os bits TF0 e TF1 (5 e 7) de TCON são flags associados aos Timers 0 e 1, respectivamente. Cada vez que ocorre um overflow (passagem de FFH para 0) na contagem de um timer, o bit correspondente é setado, o que gera um pedido de interrupção (que será atendido se a interrupção correspondente estiver habilitada). TR0 e TR1 (bits 4 e 6) permitem habilitar ou desabilitar (parar) a contagem, respectivamente para Timer 0 e Timer 1. Os bits de 0 a 3 deste registrador dizem respeito às interrupções e foram discutidos na seção 0.18.

O registrador TMOD (figura 0.16b) define o modo de funcionamento dos timers, discutido logo adiante.

1.10.1 Habilitação da contagem

Diz-se que um timer está *habilitado* quando estão satisfeitas todas as condições para que sua contagem progrida com o passar do tempo. A habilitação dos timers depende dos valores atribuídos aos bits C/T.x e GATE.x, bem como dos sinais externos aplicados aos pinos T.x e INT.x (entenda-se, deste ponto em diante, 'x' como um nome genérico para 0 ou 1).

As condições de habilitação de um timer são as seguintes:

- o bit TR.x deve estar em 1;
- se o bit GATE.x estiver em 0, o timer estará habilitado;
- caso contrário (bit GATE.x = 1), a habilitação pode ser controlada pelo usuário através do pino externo INT.x, da seguinte forma:
    - ◊ INT.x = 0: contador desabilitado;
    - ◊ INT.x = 1: contador habilitado.

Satisfeitas estas condições, a contagem avançará de acordo com a configuração do timer:

- temporizador (bit C/T.x = 0): a contagem é incrementada a cada 12 ciclos do oscilador interno, o que significa que o contador avança a cada ciclo de instrução do microcontrolador;
- contador (bit C/T.x = 1): neste modo, a contagem não avança de acordo com o oscilador interno, mas sim a cada transição descendente de um sinal externo, colocado no pino T.x (não confundir com o pino INT.x, utilizado na habilitação).

1.10.2 Aplicações

A possibilidade de se habilitar um timer externamente através do pino INT.x permite medir a largura de pulsos externos. Para tanto, conecta-se o sinal cuja largura se deseja medir ao pino INT.x, com o timer correspondente configurado como temporizador (C/T.x = 0) e com o bit GATE.x = 1. Desta forma, a contagem avançará enquanto o sinal externo estiver em nível alto, obtendo-se uma contagem proporcional à duração do pulso. A largura do pulso pode então ser calculada a partir do período do sinal de clock.

É possível também contar pulsos externos. Para tanto, conecta-se o sinal a medir ao pino T.x, com o contador configurado como contador (C/T.x = 1) e com o bit GATE.x = 0. Desta forma, a contagem avançará a cada pulso do sinal externo.

Os dois mecanismos acima podem ainda ser combinados para contar pulsos de um sinal externo somente durante um intervalo de tempo determinado por outro sinal externo. Para tanto, configura-se o timer como contador (bit C/T.x = 1) com o bit GATE.x=1. Desta forma, a contagem avançará de acordo com os pulsos recebidos no pino Tx, mas somente enquanto o nível em INT.x estiver em 1.

1.10.3 Modos de operação

Cada timer pode operar em um de quatro modos de operação diferentes. O modo de operação é definido pelos valores dos bits M0.x e M1.x do registrador TMOD.

Modo 0 (M1.x = 0 e M0.x = 0)

Neste modo, os registradores TLx e THx são vistos como um contador de 13 bits, formado por THx e pelos 5 bits menos significativos de TLx. Os 3 bits mais significativos de TLx são indeterminados e devem ser ignorados. O registrador THx pode ser visto como um contador de 8 bits, cuja freqüência de contagem é igual a 1/32 da freqüência de clock. Este modo existe por razões de compatibilidade com a família MCS48, mais antiga.

Modo 1 (M1.x = 0 e M0.x = 1)

O funcionamento deste modo é semelhante ao do modo 0. A única diferença é que o par de registradores TLxTHx é visto como um contador de 16 bits.

A figura 0.20 representa o funcionamento dos timers nos modos 0 e 1. Note a representação do bit C/T.x, em forma de uma chave que seleciona a fonte de pulsos de clock para o timer e também a saída "overflow", que gera o pedido de interrupção associado. Essa figura mostra ainda o mecanismo de habilitação da contagem, descrito no início desta seção.

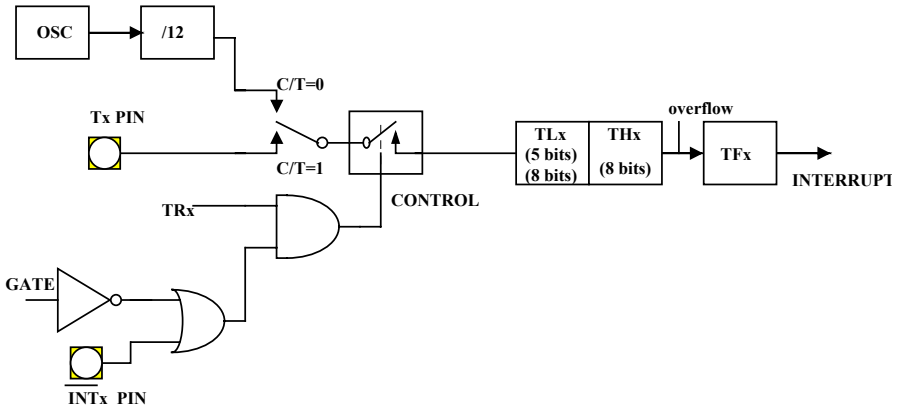


Fig. 0.20 - Funcionamento dos timers nos modos 0 e 1

Modo 2 (M1.x = 1 e M0.x = 0)

Neste modo**,** o registrador TLx funciona como um contador recarregável de 8 bits, conforme a figura 0.22. O registrador THx contém o valor que é recarregado automaticamente em TLx sempre que a contagem deste sofre um overflow (passagem de FFH para 00). O overflow faz também com que o flag TFx seja setado e portanto gera um pedido de interrupção, que será atendido de acordo com as regras de habilitação das interrupções. O valor de THx não sofre qualquer alteração e TLx retoma a contagem a partir do valor carregado.

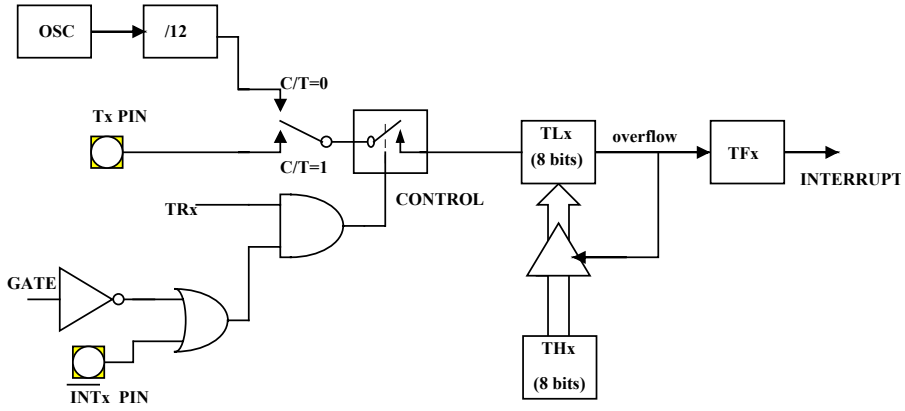


Fig. 0.22 - Funcionamento dos timers no modo 2

Modo 3 (M1.x = 1 e M0.x = 1)

TL0 e TH0 funcionam como dois contadores independentes de 8 bits, conforme a figura 0.24. Note que TR1 e TF1 são desviados do Timer 1 e utilizados para controlar TH0.

O par de registradores TL1TH1 continua funcionando como um contador de 16 bits. Contudo, não se pode desabilitar sua contagem através de TR1, e TF1 não reage quando ocorre overflow da contagem em TL1TH1. Para desabilitar a contagem de TL1TH1 quando TL0TH0 estiver no modo 3, deve-se colocar TL1TH1 também no modo 3. Então, se o Timer 0 estiver no modo 3 e o Timer 1 for colocado no modo 3, o par TL1TH1 pára de contar e preserva seu valor até que outro modo de funcionamento seja escolhido para o Timer 1.

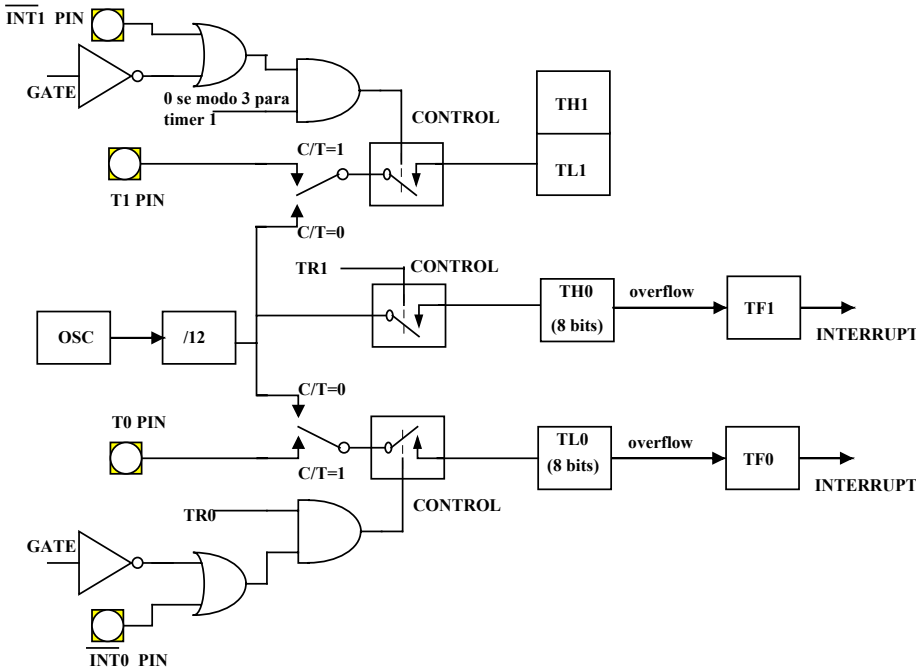


Fig. 0.24 - Timer 0 no modo 3 e alterações no timer 1

1.11 O canal serial

A interface serial no 8051 conta com dois registradores de dados, um deles utilizado na transmissão e outro na recepção. O conjunto de instruções, contudo, referencia ambos pelo nome SBUF. A distinção entre eles é feita de acordo com a natureza da operação, escrita ou leitura. Desta forma, escrever em SBUF implica no envio do byte escrito através da interface serial; analogamente, a leitura desse registrador retorna o último byte recebido.

O controle do canal serial é feito pelo registrador SCON, apresentado na figura 0.26, e pelo bit SMOD do registrador PCON, que aparece na figura 0.28. Os bits SM0 e SM1 do registrador SCON selecionam o modo de funcionamento, de acordo com a tabela 0.8.

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Fig. 0.26 - O registrador PCON

7	6	5	4	3	2	1	0
SMOD	...	...	...	GF1	GF0	PD	IDL

Fig. 0.28 - O registrador PCON

SM0	SM1	Modo	Tipo	Bits	Taxa de TX
0	0	0	Síncrona, HD	8	Fclock/12
0	1	1	Assíncrona, FD	10	Variável
1	0	2	Assíncrona, FD		Fclock/32 ou /64
1	1	3	Assíncrona, FD		Variável

Tab. 0.8 - Modos de funcionamento do canal serial

Modo 0 (SM0 = 0 e SM1 = 0)

Este modo implementa a comunicação síncrona de palavras de 8 bits. As palavras são transmitidas e recebidas através do pino RxD, o que significa que, neste modo, apenas a comunicação *half-duplex* (HD, transmissão nos dois sentidos, mas não simultânea) é possível. O sinal de clock necessário para o sincronismo é enviado pelo pino TxD. A taxa de transmissão é fixa e igual a 1/12 da freqüência do clock do sistema.

Nos demais modos, os dados são enviados através do pino TxD e recebidos através do pino RxD. Assim, esses modos permitem comunicação *full-duplex* (FD, transmissão simultânea nos dois sentidos).

Modo 1 (SM0 = 0 e SM1 = 1)

A palavra transmitida é composta por 10 bits: um start bit (nível lógico 0), oito bits de dados, e um stop bit (nível lógico 1). A taxa de transmissão é dada pela equação

$$Tx = \frac{2^{SMOD}}{32} \times \frac{f_{clock}}{12(256 - TH1)}, \qquad (0.2)$$

onde *SMOD* é o bit 7 do registrador PCON (figura 0.28b) e TH1 é o registrador mais significativo da contagem do timer 1.

Modo 2 (SM0 = 1 e SM1 = 0)

Cada palavra de dados é composta de 11 bits. O bit adicional enviado é o bit TB8 de SCON. Na recepção, é este bit que se lê em RB8. A taxa de transmissão pode ser escolhida entre 1/64 (SMOD = 0) ou 1/32 (SMOD = 1) da freqüência de clock do sistema.

Modo 3 (SM0 = 1 e SM1 = 1)

Igual ao modo 2 exceto pela taxa de transmissão, dada também pela equação 0.2.

O bit SM2 do registrador SCON tem diferentes interpretações, dependendo do modo de operação selecionado:

- no modo 0, não tem qualquer efeito, devendo permanecer em 0;
- no modo 1, inibe (SM2 = 1) ou habilita (0) a geração de um pedido de interrupção da porta serial quando da recepção de um stop bit inválido;
- nos modos 2 e 3, permite habilitar a comunicação entre vários 8051.

REN (Reception ENable) habilita a recepção. Quando está em 1, o primeiro start bit em RxD implica recepção de um dado em SBUF.

TI é o bit de requisição de interrupção da transmissão. É setado pelo hardware após a transmissão do oitavo bit de dados quando no modo 0, e no início da transmissão do stop bit nos outros modos.

O bit RI é o bit de requisição de interrupção na recepção. É setado pelo hardware no momento da recepção do oitavo bit de dados no modo 0, ou durante a recepção de um stop bit nos outros modos. TI e RI devem ser reinicializados pelas rotinas de tratamento das respectivas interrupções de modo a habilitar novas interrupções.

1.12 Sistema de desenvolvimento Avocet

Um sistema de desenvolvimento é um sistema que permite fazer a edição, depuração e teste de programas para um determinado tipo de processador. A maioria dos sistemas de desenvolvimento para microprocessadores e microcontroladores se baseiam no PC. Normalmente são necessários um editor de programas, um cross-assembler e/ou um compilador e um linker, além de simuladores ou emuladores.

Muitos cross-assemblers permitem a inclusão de macros, que são trechos de programas fontes aos quais é atribuído um nome. Este nome pode então ser usado em qualquer parte do programa como se fosse uma instrução.

1.12.1 Sistema de desenvolvimento Avocet

Dentre as diversas ferramentas de software para o desenvolvimento de programas para a família 8051, uma das mais conhecidas é o conjunto da Avocet, ilustrado na figura 0.30. Praticamente todos os fornecedores de software para este tipo de desenvolvimento apresentam o mesmo elenco de programas utilitários.

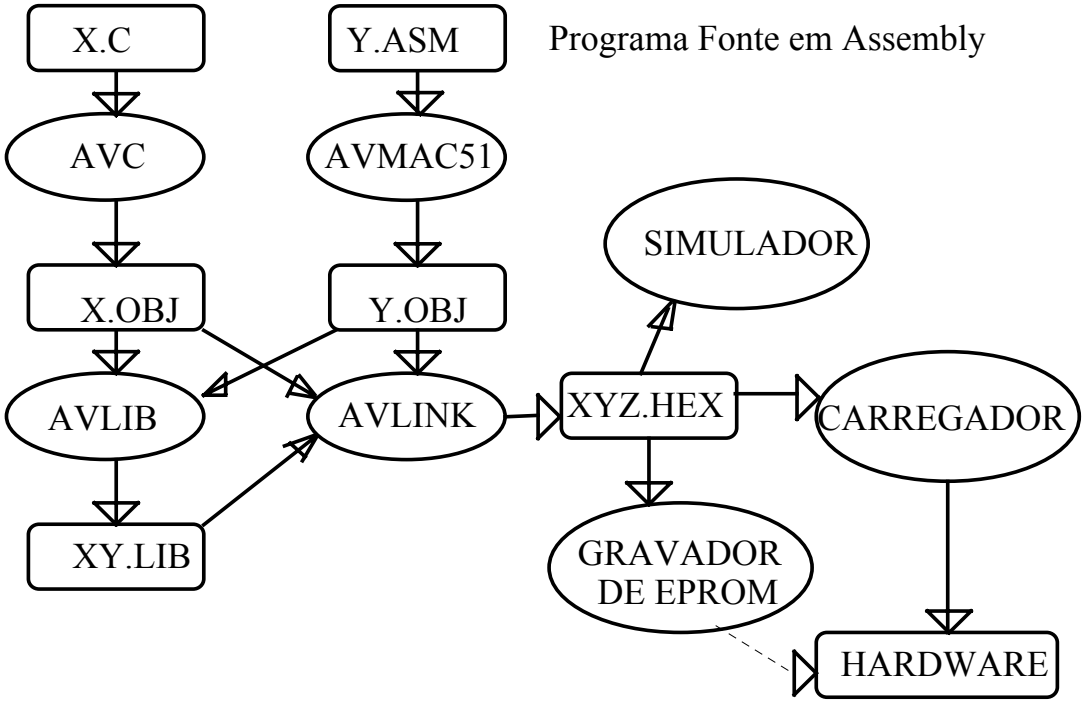


Fig. 0.30 - Conjunto de ferramentas de desenvolvimento

1.12.2 Roteiro de elaboração de um programa

Esta seção apresenta os passos a serem seguidos para a elaboração de um programa para microcontroladores da família 8051, utilizando as ferramentas da Avocet.

1.12.2.1 Edição do programa-fonte

Utilizando um editor de texto simples, que permita gravar arquivos em formato ASCII, criar o programa fonte. O nome do arquivo deve ter no máximo oito letras e extensão .ASM.

1.12.2.2 Montagem

A sintaxe para chamar o assembler é, no caso do exemplo acima:

AVMAC51 <*nome*>

Nesta etapa, são gerados os arquivos <*nome*>.obj e <*nome*>.prn contendo, respectivamente, os códigos hexadecimais correspondentes às instruções e a listagem para consulta dos resultados.

1.12.2.3 Linkagem

A sintaxe para chamar o linker é, no caso do exemplo acima:

AVLINK51 <*nome*>=<*novo nome*>

Nesta etapa, são gerados os arquivos <*novo nome*>.hex e <*novo nome*>.map contendo, respectivamente, os códigos hexadecimais em arquivo-texto e o mapa de linkagem. O sistema oferece, através da sintaxe <*nome*>=<*novo nome*>, a oportunidade de criar um programa final (.HEX) com nome diferente do arquivo-fonte, mas esta facilidade é raramente útil. Normalmente utiliza-se a sintaxe

AVLINK51 <*nome*>=<*nome*>.

1.12.2.4 Simulação

Um simulador é um programa que roda em um determinado computador e simula a operação de uma outra CPU. O simulador da Avocet é chamado através do comando **AVSIM51**

1.12.2.5 Gravação em EPROM

Uma vez depurado e funcionando, o programa está pronto para ser colocado na EPROM do sistema que utiliza o microcontrolador.

1.12.2.6 Utilização do simulador AVSIM51

- Chame o programa: AVSIM51;
- escolha a opção C (8031);
- tecle **L** (**L**oad) para carregar um programa;
- digite **P** (**P**rogram) para escolher um programa;
- digite o nome do programa a ser carregado, com a extensão .HEX, e finalize com Enter;
- tecle ESC. Esta tecla desloca o cursor alternadamente entre a linha de comandos (na parte inferior da tela) e a parte superior da tela, onde são exibidos os registradores do 8031;
- após o primeiro toque da tecla ESC, o cursor estará no campo de registradores, mais especificamente sobre o campo PC (Program Counter). Digite o endereço do início do programa;
- a partir deste ponto, estão disponíveis os comandos da tabela 0.10;
- para sair do programa de simulação, digite ESC, **Q** (**Q**uit), **E** (**E**xit).

Controle de execução e exibição		Movimentação do cursor	
F1	RUN	Esc	Regs. / linha de comando
F10	Single Step (passo a passo)	F7	Cursor - Hex, Bin, ASCII
F9	UNDO - volta um comando	Ctrl Pg Up	Scroll lock
F2	Move Breakpoint UP	Ctrl A / B	Registradores A / B
F3	Set breakpoint	Ctrl D	Data Pointer
F4	Move Breakpoint DOWN	Ctrl CXFO	Flags C/AC/FO/OV
F5	Velocidade de simulação	Ctrl I	Interrupções
Alt F5	Alternar apresentação de símbolos / valores hexadecimais	Ctrl P	PC
F6	Atualizar todas as janelas / somente janelas com trace ligado	Ctrl R	Banco de Registradores

Tab. 0.10 - Comandos do simulador AVSIM51

A representação de constantes na tela do simulador se dá segundo a convenção da tabela 0.12.

Sistema de numeração	Notação
Decimal	255
Hexadecimal	$FF ou 0FFH
Binário	%1111 ou 1111B

Tab. 0.12 - Representação de constantes no AVSIM51

1.13 Exercícios

1. Estude as instruções que aparecem no programa abaixo e procure determinar o valor do acumulador ao final da execução. Crie um arquivo-fonte com o programa, gere o arquivo .hex e simule-o no AVSIM51. São necessárias as diretivas ORG e END, que constam dos anexos.

```
      MOV R1,#3
      MOV R3,#4
      INC @R1
      MOV A,@R1
      MOV DPTR,#MENS
      MOVC A,@A+DPTR
      SJMP $
MENS  DB "1234567890ABCDE",0
      END
```

2. Use o simulador para ver a troca dos bancos de registradores no programa abaixo**(BANCO.asm)**

```
      ORG 2000H;
      MOV A,12            ; confirme no simulador o que faz esta instrução ?
INICIO: CLR RS0
      CLR RS1
      MOV R0,A            ; DADOS NO BANCO0  RS1=RS0=00
      MOV R1,A
      MOV R2,#00H
      MOV R3,#00H
      INC A
      SETB RS0            ; DADOS NO BANCO1  RS1 RS0 = 01
      MOV R0,A
      MOV R1,A
```

```
      MOV R2,#11H
      MOV R3,#11H
      INC A
      CLR RS0             ; DADOS NO BANCO2  RS1 RS0 = 10
      SETB RS1
      MOV R0,A
      MOV R1,A
      MOV R2,#22H
      MOV R3,#22H
      INC A
      SETB RS0            ; DADOS NO BANCO3  RS1  RS0 = 11
      MOV R0,A
      MOV R1,A
      MOV R2,#33H
      MOV R3,#33H
      INC A
      JMP INICIO
      END
```

3. O programa abaixo lê a porta P1. Se o valor lido for par, incrementa R2; em caso contrário, incrementa R1. Os dados pares são colocados na porta P2 e os dados ímpares na porta P3; todos os dados são armazenados seqüencialmente a partir da posição de memória 20H

PARIDADE.ASM e P1.DAT

```
      ORG 2000H
      MOV R0,#20H         ;INICIO DO ARMAZENAMENTO
      MOV R1,#0H          ;CONTADOR DOS IMPARES
      MOV R2,#0H          ;CONTADOR DOS PARES
INICIO:MOV A,P1
      MOV @R0,A           ;GRAVA DADO
      ANL A,#01           ;VER SE PAR OU IMPAR
      JNZ IMPAR
PAR:  INC R2
      MOV A,@R0
      MOV P2,A
      INC R0
      JMP INICIO
IMPAR:INC R1
      MOV A,@R0
      INC R0
      MOV P3,A            ; MOSTRA DADO
      JMP INICIO
      END
```

Este programa utiliza a porta P1 como porta de entrada, dentro do simulador. Esta entrada de dados pode ser feita por meio de um arquivo (ideal quando existem muitos dados) ou com uma entrada manual, neste caso desloque o cursor até a porta P1 e digite ali o valor pretendido.

Também pode-se associar um arquivo a essa porta. Cada vez que o programa acessa a porta, um novo caracter do arquivo é lido. O valor lido é o valor hexadecimal do caracter, incluindo espaços em branco e caracteres de fim de linha (0DH, 0AH). Opcionalmente, pode-se fazer com que a leitura do arquivo recomece do início sempre que o programa chegar ao fim do arquivo, o que garante uma fonte ininterrupta de dados para simulação.

Para o programa acima, crie com um editor de texto um arquivo chamado P1.DAT, contendo a linha:

0123456789

e um "ENTER" no final.

Monte o programa e carregue-o no AVSIM51. Para associar o arquivo de entrada à porta P1, siga os passos abaixo:

- ◊ use o comando **IO**, opção Open;
- ◊ digite o nome do arquivo de entrada, **P1.DAT**;

- ◊ perguntado se a leitura deve recomeçar do início, responda **Yes**;
- ◊ perguntado se deseja criar um arquivo de saída, responda somente com **Enter** (não cria arquivo de saída);
- ◊ perguntado sobre o modo de **IO TRIGGER**, selecione **Opcode access**;
- ◊ perguntado sobre o modo de transferência de IO, digite **P1,IN**.

Em seguida, comece a simulação. Observe que, cada vez que o programa faz um acesso de leitura à porta P1, um novo caracter do arquivo P1.DAT é lido, como se estivesse presente na porta nessa hora. Os dados escritos nas portas P2 e P3 podem ser observados no simulador. Opcionalmente, v. pode criar um arquivo de saída e examinar os dados colocados lá depois que encerrar a simulação.

Uma vez entendido o funcionamento do programa acima use a instrução ***CJNE Rn,#dado,endereço*** para fazer com que o programa fique em um loop infinito (JMP $) quando tiver encontrado 3 números do mesmo tipo.

Crie um outro arquivo de dados P2.DAT, faça os mesmos testes e explique a diferença existente nos dois arquivos.!!!!!!!

1.14 Referências bibliográficas

[Mors88] Morse, Stephen P.: Microprocessadores 8086/8088 Arquitetura, projeto, sistemas e programação. Editora Campus, Rio de Janeiro (1988)

[Inte89] INTEL CORPORATION. 8-bit Embedded Controller Handbook. Santa Clara, 1989.

[Siem90] Microcomputer Components SAB 80C515 / 80C535. Siemens A.G. (1990)

[Silv94] da Silva Jr., Vidal Pereira: Aplicações Práticas do Microcontrolador 8051. Editora Érica Ltda., Tatuapê - SP - Brasil (1994). ISBN 085-7194-194-7

[Nico00] Nicolosi Denys: Microcontrolador 8051 detalhado. Editora Érica Ltda., Tatuapê - SP - Brasil (2000). ISBN 085-7194-721-x

www.eel.ufsc.br\eel7030

2 - Conjunto de instruções do 8051

2.1 Abreviaturas utilizadas na tabela de instruções

Abreviatura	Interpretação
Rn	registrador R0 a R7 do banco selecionado por RS1 e RS0
@Ri	endereçamento indireto do byte endereçado por Ri; i ≤ 1
#dado8	endereçamento imediato; dado8 é uma constante de 8 bits
#dado16	endereçamento imediato; dado16 é uma constante de 16 bits
Direto	endereçamento direto; direto é um endereço da RAM interna
Bit	endereçamento direto de um bit da memória RAM interna ou SFR
End16	endereço de 16 bits (LJMP e LCALL)
End11	endereço de 11 bits (SJMP e SCALL)
End. rel.	endereçamento relativo (SJMP e todos os JMPs condicionais)

2.2 Principais diretivas

Diretiva	Significado	Função	Exemplo
ORG	Origin	Definir ende. montagem	ORG 2000H
EQU	Equate	Criar sinônimos	CR EQU 0DH; LF EQU 0AH
DB <valor>	Define byte	Criar variáveis	CONTAGEM DB 00H
DB <valor>,..	Define byte	Criar vetores	TABELA DB 15H, 22H, 35H
DB <string>	Define byte	Criar strings ASCII	MSG DB “Mensagem 1”
%INCLUDE	Include	Incluir arquivos-fonte	%INCLUDE RET.ASM
END	End	Indica final de programa	END

2.3 As instruções

As tabelas a seguir apresentam o conjunto de instruções do microcontrolador 8051.

As colunas “B” e “C” indicam, respectivamente, o número de bytes e o número de ciclos de clock de cada instrução. As colunas “CY”, “AC” e “OV” de cada tabela indica como cada instrução afeta, respectivamente, os flags de carry, auxiliary carry e overflow, de acordo com a seguinte notação:

- 0: o flag é resetado nesta operação;
- 1: o flag é setado nesta operação
- *: o flag é afetado e seu valor pode ser 0 ou 1, dependendo do resultado da operação. ***Observe que só algumas instruções alteram os flags!!!!.***
- -: o flag não é afetado.

2.3.1 Instruções aritméticas

Mnem.	Operando	Descrição	B	C	CY	AC	OV
ADD	A,Rn	Soma registrador ao acumulador	1	1	*	*	*
ADD	A,direto	Soma byte ao acumulador	2	1	*	*	*
ADD	A,@Ri	Soma byte no endereço Ri a A	1	1	*	*	*
ADD	A,#dado8	Soma dados imediatos a A	2	1	*	*	*
ADDC	A,Rn	Soma o registrador e o carry a A	1	1	*	*	*
ADDC	A,direto	Soma byte e o carry a A	2	1	*	*	*

ADDC	A,@Ri	Soma byte no endereço Ri e carry a A	1	1	*	*	*
ADDC	A,#dado8	Soma dados imediatos e o carry a A	2	1	*	*	*
SUBB	A,Rn	Subtrai Rn e borrow de A: A = A-CY-Rn	1	1	*	*	*
SUBB	A,direto	Subtrai byte e borrow do acumulador	2	1	*	*	*
SUBB	A,@Ri	Subtrai byte no end. Ri e borrow de A	1	1	*	*	*
SUBB	A,#dado8	Subtrai dados imediatos e borrow de A	2	1	*	*	*
INC	A	Incrementa acumulador	1	1	-	-	-
INC	Rn	Incrementa registrador	1	1	-	-	-
INC	Direto	Incrementa byte	2	1	-	-	-
INC	@Ri	Incrementa byte no endereço Ri	1	1	-	-	-
DEC	A	Decrementa acumulador	1	1	-	-	-
DEC	Rn	Decrementa registrador	1	1	-	-	-
DEC	Direto	Decrementa byte	2	1	-	-	-
DEC	@Ri	Decrementa byte no endereço Ri	1	1	-	-	-
INC	DPTR	Incrementa datapointer	1	2	-	-	-
MUL	AB	Multiplica A e B, resultado em BA	1	4	0	-	*
DIV	AB	Divide A/B, quociente em A e resto em B	1	4	0	*	0
DA	A	Ajuste do acumulador após adição BCD	1	1	*	*	-

2.3.2 Instruções lógicas

Mnem.	Operando	Descrição	B	C	CY	AC	OV
ANL	A,Rn	AND entre registrador e acumulador	1	1	-	-	-
ANL	A,direto	AND entre byte e acumulador	2	1	-	-	-
ANL	A,@Ri	AND entre o conteúdo do endereço Ri e A	1	1	-	-	-
ANL	A,#dado8	AND entre dado imediato e o acumulador	2	1	-	-	-
ANL	direto,A	AND entre acumulador e byte	2	1	-	-	-
ANL	dir,#dado8	AND entre dado imediato e byte	3	2	-	-	-
ORL	A,Rn	OR entre registrador e acumulador	1	1	-	-	-
ORL	A,direto	OR entre byte e acumulador	2	1	-	-	-
ORL	A,@Ri	OR entre o conteúdo do endereço Ri e A	1	1	-	-	-
ORL	A,#dado8	OR entre dado imediato e o acumulador	2	1	-	-	-
ORL	direto,A	OR entre acumulador e byte	2	1	-	-	-
ORL	dir,#dado	OR entre byte e dado imediato	3	2	-	-	-
XRL	A,Rn	XOR entre o acumulador e Registrador	1	1	-	-	-
XRL	A,direto	XOR entre acumulador e byte	2	1	-	-	-
XRL	A,@Ri	XOR entre A e o conteúdo do endereço Ri	1	1	-	-	-
XRL	A,#dado8	XOR p/ dado imediato e o acumulador	2	1	-	-	-
XRL	direto,A	XOR entre acumulador e byte	2	1	-	-	-
XRL	dir,#dado	XOR entre byte e dado imediato	3	2	-	-	-
CLR	A	Zera o acumulador	1	1	-	-	-
CPL	A	Complementa o acumulador	1	1	-	-	-
RL	A	Rotaciona o acumulador para a esquerda	1	1	-	-	-
RLC	A	Rotaciona A para a esquerda pelo carry	1	1	*	-	-
RR	A	Rotaciona A à direita	1	1	-	-	-
RRC	A	Rotaciona A à direita pelo carry	1	1	*	-	-
SWAP	A	Permuta os bits (3-0) e (7-4) de A	1	1	-	-	-

2.3.3 Instruções de manipulação de bits

Mnem.	Operando	Descrição	B	C	CY	AC	OV
CLR	C	Reseta CY	1	1	0	-	-
CLR	bit	Reseta o bit endereçado	2	1	-	-	-
SETB	C	Seta o CY	1	1	1	-	-
SETB	bit	Seta o bit endereçado	2	1	-	-	-
CPL	C	Complementa CY	1	1	*	-	-
CPL	bit	Complementa o bit endereçado	2	1	-	-	-
ANL	C,bit	AND entre o bit endereçado e CY	2	2	*	-	-
ANL	C,/bit	AND entre CY e complemento do bit	2	2	*	-	-
ORL	C,bit	OR entre CY e o bit endereçado	2	2	*	-	-
ORL	C,/bit	OR entre CY e complemento do bit	2	2	*	-	-
MOV	C,bit	Copia bit endereçado para CY	2	1	*	-	-

MOV	bit,C	Copia CY para bit endereçado	2	2	-	-	-

2.3.4 Instruções de movimentação de dados

Mnem.	Operando	Descrição	B	C	CY	AC	OV
MOV	A,Rn	Move registrador para acumulador	1	1	-	-	-
MOV	A,direto	Move byte para acumulador	2	1	-	-	-
MOV	A,@Ri	Move conteúdo do endereço Ri para A	1	1	-	-	-
MOV	A,#dado8	Move dados imediatos para acumulador	2	1	-	-	-
MOV	Rn,A	Move acumulador para registrador	1	1	-	-	-
MOV	Rn,direto	Move byte para registrador	2	2	-	-	-
MOV	Rn,#dado8	Move dados imediatos para registrador	2	1	-	-	-
MOV	direto,A	Move acumulador para byte	2	1	-	-	-
MOV	direto,Rn	Move registrador para byte	2	2	-	-	-
MOV	direto,direto	Move da byte1 para a byte2	3	2	-	-	-
MOV	direto,@Ri	Move conteúdo do endereço Ri p/ byte	2	2	-	-	-
MOV	dieto,#dado8	Move dados imediatos para byte	3	2	-	-	-
MOV	@Ri,A	Move A para o endereço em Ri	1	1	-	-	-
MOV	@Ri,direto	Move byte para o endereço em Ri	2	2	-	-	-
MOV	@Ri,#dado8	Move dado imediato p/ endereço em Ri	2	1	-	-	-
MOV	DPTR,#dad16	Carrega DPTR com constante de 16 bits	3	2	-	-	-
MOVC	A,@A+DPTR	Move byte endereçado por DPTR+A para A	1	2	-	-	-
MOVC	A,@A+PC	Move byte endereçado por PC+A para A	1	2	-	-	-
MOVX	A,@Ri	Move RAM externa (end. de 8 bits) p/ A	1	2	-	-	-
MOVX	A,@DPTR	Move RAM externa (end. de 16 bits) p/ A	1	2	-	-	-
MOVX	@Ri,A	Move A p/ RAM externa (end. de 8 bits)	1	2	-	-	-
MOVX	@DPTR,A	Move A p/ RAM externa (end. de 16 bits)	1	2	-	-	-
PUSH	Direto	Move byte para a pilha (stack)	2	2	-	-	-
POP	Direto	Retira byte da pilha (stack)	2	2	-	-	-
XCH	A,Rn	Permuta A com Rn	1	1	-	-	-
XCH	A,direto	Permuta A com byte	2	1	-	-	-
XCH	A,@Ri	Permuta A com o conteúdo do endereço Ri	1	1	-	-	-
XCHD	A,@Ri	Permuta nibble inferior de A c/ byte	1	1	-	-	-

2.3.5 Instruções de controle de fluxo

Mnem.	Operando	Descrição	B	C	CY	AC	OV
ACALL	end11(2k)	Chamada de sub-rotina	2	2	-	-	-
LCALL	end16 (64k)	Chamada de sub-rotina	3	2	-	-	-
RET		Retorno de sub-rotina	1	2	-	-	-
RETI		Retorno de interrupção	1	2	-	-	-
AJMP	end11 (2k)	Desvio incondicional	2	2	-	-	-
LJMP	end16 (64k)	Desvio incondicional	3	2	-	-	-
SJMP	end. rel.	Desvio incondicional	2	2	-	-	-
JMP	@A+DPTR	Desvio indireto relativo a DPTR	1	2	-	-	-
JZ	end. rel.	Desvio se A = 0	2	2	-	-	-
JNZ	end rel.	Desvio se A ≠ 0	2	2	-	-	-
JC	end rel.	Desvio se CY = 1	2	2	-	-	-
JNC	end rel.	Desvio se CY =0	2	2	-	-	-
JB	bit,end. rel.	Desvio se bit = 1	3	2	-	-	-
JNB	bit,end. rel.	Desvio se bit = 0	3	2	-	-	-
JBC	bit,end. rel.	Desvio se bit =1; zera bit	3	2	-	-	-
CJNE	A,direto,end. rel.	Desvio se A ≠ byte	3	2	*	-	-
CJNE	A,#dado8,end. rel.	Desvio se A ≠ dado imediato	3	2	*	-	-
CJNE	Rn,#dado8,end. rel.	Desvio se Rn ≠ dado imediato	3	2	*	-	-
CJNE	@Ri,#dado8,end. rel.	Desvio se byte endereçado ≠ dado	3	2	*	-	-
DJNZ	Rn,end.rel	Decrementa Rn; desvio se Rn ≠ 0	2	2	-	-	-
DJNZ	direto,end. rel.	Decrementa byte e desvia se ≠ 0	3	2	-	-	-
NOP		Instrução sem efeito	1	1	-	-	-

3 O Kit de desenvolvimento da Datapol

3.1 Apresentação

O sistema 9431 tem uma CPU 8031 e pode comunicar-se com um PC através do canal serial, através do qual é possível transferir programas desenvolvidos no PC para o kit. Este conta ainda com outros componentes, listados na tabela 4.2 juntamente com os endereços que ocupam.

Componente	Endereços
EPROM 2764 (16 kB)	0000H - 1FFFH
RAM 62xx (32 kB)	5000H - 3FFFH
LCD	4000H - 4002H

Tab. 2.2 - Componentes periféricos do kit WF9431

O display de cristal líquido (LCD) possui duas linhas conforme mostra a figura 3.1.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

Fig. 2.2 – Display de Cristal Liquido

3.2 Restrições

As portas P0 e P2, bem como os pinos P3.6 e P3.7, são usados para acessar a memória externa. Os bits P3.0 e P3.1 e o bit 7 do registrador PCON são utilizados na transmissão de dados pelo canal serial e portanto só podem ser utilizados por programas que, depois de carregados no kit, não façam mais uso da comunicação com o PC.

3.3 Rotinas do Monitor e desvio de Interrupções

Na EPROM existem diversas sub-rotinas que dão suporte à execução de tarefas tais como a entrada e saída de dados via teclado e a apresentação de mensagens no LCD. Estas sub-rotinas estão listadas na tabela. A chamada dessas subrotinas é feita mediante a instrução LCALL <*nome da sub-rotina*>. Na tabela a seguir estão listados os nomes das rotinas, seus endereços, além de uma breve descrição de suas funções e o tempo de execução. Explique como você pode conferir se este tempo de execução está correto !!

Nome	Endereço	Função	tempo
RAM	EQU 5000H	INICIO DA RAM	
INTE0	EQU 4230H	DESVIO INT. EXTERNA 0-***Tecla INTER***	
INTT0	EQU 4240H	DESVIO DA INTERRUPÇÃO TIMER 0	
INTE1	EQU 4250H	DESVIO DA INT. EXTERNA 1 Tecla INTER	

INTT1	EQU 4260H	DESVIO DA INTERRUPÇÃO TIMER 1	
REGI&TI	EQU 4270H	DESVIO DA INT. da PORTA SERIAL	
SUBROUTINAS	***Endereço***	***Função***	
CLR_DSP	EQU 10AAH	APAGA DISPLAY	1ms
MENS	EQU 110FH	MENSAGEN no LCD, (DPTR aponta mens.)	
AC_DSP	EQU 10E7H	IMPRIME NUMERO no DISPLAY, (A)	
DPT_DSP	EQU 1121H	IMPRIME NUMERO no display (DPTR)	
DSP_DAT	EQU 10FFH	IMPRIME CARACTER no LCD (A, ASCII)	
DSP_COM	EQU 109AH	MANDA COMANDO (A) PARA DISPLAY	
LE_TEC	EQU 1002H	LE UMA TECLA digitada no teclado (A)	
LE_DAD0	EQU 0F00H	LE DUAS TECLAS digitadas no teclado (A)	
LE_DAD1	EQU 0F27H	LE DUAS TECLAS E ESPERA ENTER (A)	
ASCII	EQU 114CH	Passa ACUMULADOR para ASCII (R1 R2)	
AD	EQU 145FH	LE O SINAL DO CONVERSOR (DPTR e A)	
DA	EQU 1471H	ENVIA (A) PARA O CONVERSOR DA	
DELAY	EQU 11C8H	ATRASO FIXO DO PROGRAMA MONITOR	

Tab. 3.3 – Rotinas do Monitor

Tanto os endereços das subrotinas como os demais encontram-se definidos como sinônimos no arquivo ENDERED.ASM. Convém incluir este arquivo no início de todos os programas desenvolvidos para o kit Datapool 9431 com a seguinte diretiva

```
%INCLUDE ENDERED.ASM          colocada no início do arquivo-fonte.
```

O redirecionamento das interrupções, no simulador, é conseguido através das linhas finais do arquivo ENDERED.ASM, reproduzidas abaixo. No KIT isto está gravado na EPROM.

```
        ORG 0003H           ;INT 0
        LJMP 4230H          ;INT 0 redirecionada
        ORG 000BH           ;T0
        LJMP 4240H          ;T0 redirecionada
        ORG 0013H           ;INT 1
        LJMP 4250H          ;INT 1 redirecionada
        ORG 001BH           ;T1
        LJMP 4260H          ;T1 redirecionada
```

Note que estas linhas não têm efeito sobre programas transferidos para o kit, pois este tem EPROM nos endereços modificados. A utilidade deste redirecionamento está em permitir a execução de programas no simulador. Por isso, programas que utilizem tratadores de interrupção devem seguir a estrutura dada abaixo:

```
        ORG RAM
        SJMP INICIO
        ORG  INTE0
        ...                 ;tratador da interrupção externa 0
        ORG  INTT1
        ...                 ;tratador da interrupção do timer 1
INICIO: ...                 ;instruções do programa
        END
```

Um outro problema que aparece no desenvolvimento de programas para o kit é que o simulador AVSIM51 não possui as sub-rotinas de serviço e portanto não pode simulá-las. Então, se um programa escrito para chamar sub-rotinas da eprom for executado no simulador, é preciso tomar cuidado para que o programa não se perca quando saltar para uma dessas sub-rotinas. A solução adotada aqui

consiste em incluir, no início do programa, o arquivo RET.ASM, que coloca instruções RET em todos os endereços listados na tabela Desta forma, as chamadas das sub-rotinas são ignoradas durante a simulação e não atrapalham. Mais uma vez, este arquivo só tem efeito no simulador, pois o kit tem EPROM nesses endereços.

3.5 Sistema de Desenvolvimento

Para fazer o desenvolvimento de programas para o 8031 são necessários:

- um editor, EDIT;
- um montador, AVMAC51;
- um linkador, AVLINK51;
- um conversor de arquivo HEX (texto) para BIN (binário), HEXBIN;
- um programa denominado **9431** para transferir o arquivo do PC para o kit**.**

Para agilizar a compilação, pode-se criar um arquivo bat com o conteúdo listado abaixo.

```
avmac51 %1
@pause
avlink51 %1=%1
hexbin %1.hex %1.bin I
```

Supondo ser o nome dado a este arquivo **av.bat**, pode-se digitar

av <*nome do arquivo fonte*>

para criar o programa.

Para efetuar a transferência do programa para o kit, conecte-o ao PC pela interface serial e:

Digite **9431** no PC

Ligue o Kit no **modo teclado** e digite **Serial** e **0** para carregar o programa.

No PC deve-se definir o programa que será transferido e depois enviar o mesmo.

Para rodar o programa no KIT digitar: **EXEC 5000 ENTER**

3.6 - Exemplos

3.6.1 Utilização da interrupção do timer 0

O programa abaixo utiliza o timer 0 para controlar a escrita das palavras "BOM" e "DIA" no LCD, que alternam a cada segundo. **SegundoD.asm**

```
;**************************************************segundoD.asm
; Timer0 controla escrita de mensagem BOM DIA a cada segundo no Display
; Ulisses Percegona Neto 05.06.02
;**********************************************************
%include ENDERED.ASM
TIMER0H   EQU 00BH      ; valores para o timer0
TIMER0L   EQU 0DCH

        ORG INTT0
                LJMP INTR0
        ORG RAM
                LJMP INICIO
;INTERRUPCAO DO TIMER 0
INTR0:      INC R0
                CJNE R0,#16h,RECARGA     ; 16 Int perfazem 1 segundo
                MOV  R0,#00
;POSICIONA CURSOR PRIMEIRA AREA LCD
                CJNE R1,#0,DIA  ; seleciona a mensagem
BOM:      MOV DPTR,#MENS_1
```

```
                MOV A,#82H      ; seleciona linha 1 e coluna 2
                LCALL DSP_COM
                LCALL MENS
                MOV R1,#1       ; alterna a mensagem
                LJMP RECARGA
DIA:    MOV DPTR,#MENS_2
                MOV A,#82H      ; seleciona linha 1 e coluna 2
                LCALL DSP_COM
                LCALL MENS
                MOV R1,#0       ; alterna a mensagem
RECARGA:        MOV TH0,#TIMER0H
                MOV TL0,#TIMER0L
                RETI

INICIO: LCALL CLR_DSP
;R0 = USADO PARA CONTROLE DE ESTOURO DO TIMER 0
        MOV R0,#0    ;  16 ESTOUROS = 1 segundo
        MOV R1,#0    ;  R1 =USADO PARA INDICAR 0=BOM!!!!! 1=!!!!!DIA
        MOV TMOD,#01h   ; TIMER 0 NO MODO DE CONTAGEM 1
        MOV TH0,#TIMER0H        ;Valor inicial do Timer0
        MOV TL0,#TIMER0L
        SETB ET0   ;        HABILITAR TIMER 0 GERAR INTERRUPCAO
        SETB EA    ;        HABILITA TODAS INTERRUPCOES
        SETB TR0   ;        DISPARA TIMER0
        SJMP $     ;        AGUARDA INTERRUPCAO
MENS_1: DB 13,"!!!!!Bom!!!!!"
MENS_2: DB 13,"!!!!!Dia!!!!!"
        END
; Freq Cristal = 12MHZ  ==> Periodo =0.00000008333S (1/12MHZ)
; O TIMER incrementa a cada 12 períodos de clock
; Quero uma pausa de 1 segundo,  Logo, necessito de 1000000 períodos  (1s/12Periodo)
; Como o topo de contagem máximo do TIMER e' 65536 (0..65535)
; Necessitarei de 15,2588 estouros do TIMER de 65536.
; Como o microcontrolador trabalha apenas com números inteiros usamos 16 contagens,  65536*15 = 1048580
; 1048580 ultrapassa 1000000 de 43576.  Esta diferença se tira em cada contagem, ou seja, o TIMER não partirá de 0
; 65536 - 3036 = 62500 que multiplicado por 16 da 1000000
; 3036 HEXA e' 0BDCH.   Logo, sempre Inicializo TH0=0BH e TL0=DCH
```

3.6.2 Leitura do Teclado

O programa abaixo (par.asm) lê números até 256 do teclado do PC. Os números ímpares aparecem na área 1 do LCD e os pares na área 2. O programa termina quando tiverem sido digitados 4 números do mesmo tipo (pares ou ímpares).

```
%INCLUDE ENDERED.ASM        ;ENDERECOS DAS ROTINAS NA EPROM
         ORG RAM            ; ENDERECO DA RAM
         LJMP INICIO        ; INICIO DO PROGRAMA
         ORG 2003H          ; INTERRUPCAO INT0
         RETI               ; SEM INTERRUPÇÃO
         ORG 200BH          ; INTERRUPCAO DO TIMER 0
         RETI               ; NÃO USADO
INICIO:  MOV DPTR,#MENS1
         CALL MENTER
         LCALL APAGA        ; LIMPAR DISPLAY
         MOV R1,#4          ; CONTADOR DE NÚMEROS IMPARES
         MOV R4,#4          ; CONTADOR DE NÚMEROS PARES
LACO:    LCALL LEBYTE       ; ENTRADA DO TECLADO
         CALL APAGA         ;
         PUSH ACC           ; SALVAR VALOR LIDO
         ANL A,#1           ; VER SE É PAR OU IMPAR
         CJNE A,#1,PAR      ; DESVIA SE PAR
IMPAR:   MOV A,#0           ; DEFINIR AREA1
         CALL AREA1
         POP ACC            ; RECUPERA VALOR LIDO
```

```
        CALL NUMLCD         ; MOSTRAR VALOR IMPAR
        DJNZ R1,LACO        ; CONTADOR
        MOV DPTR,#MENIMP    ; ATINGIU N IMPARES
        SJMP FIM
PAR:    MOV A,#0            ; POSICIONAR AREA2
        CALL AREA2
        POP ACC             ; RECUPERAR VALOR
        CALL NUMLCD         ; IMPRIMIR PARES NA AREA2
        DJNZ R4,LACO
        MOV  DPTR,#MENPAR   ; ATINGIU N PARES
FIM:    CALL APAGA
        CALL MENLCD         ; MOSTRA MENSAGEM DE FINAL ATINGIDO
        CALL MENTER
        SJMP $              ; PERMANECE PRESO NESTE LACO
MENS1   DB ' ENTRE COM UM NUMERO < 255 ',CR,LF
        DB ' O PROGRAMA IRA DETECTAR SE SAO PARES OU IMPARES ',CR,LF
        DB ' O PROGRAMA TERMINA COM A ENTRADA DE 4 DO MESMO TIPO',CR,LF,0
MENPAR  DB '4 PARES ',0
MENIMP  DB '4IMPARES',0
        END
```

3.6.3 Exercicios

1. Fazer um programa para apagar as duas áreas do display e escrever a letra A na terceira coluna da 1 LINHA e o numero 2 na 4 coluna da 2.
2. Fazer um contador hexadecimal que coloque este valor na porta P1.
3. Alterar o contador acima para fazer uma contagem decimal e mostrar os valores de forma legível no display de cristal líquido (incluir um delay) para permitir a leitura.
4. Mostrar o conteúdo de 16 posições de memória, a partir de uma posição de memória solicitada ao operador pelo teclado, no terminal de vídeo. Defina uma tecla para avançar para a linha seguinte e outra para retroceder.
5. Fazer um programa que gere uma onda quadrada na porta P1.7.
6. Gerar uma onda quadrada, em P1.0, com a maior freqüência possível. Qual é esta freqüência ?
7. Fazer um programa que gere uma onda retangular na porta P1.7.
8. Fazer um programa para ler a porta P1 e mostrar o resultado em hexadecimal, binário e decimal no LCD.
9. Contar o numero de vezes que a saída P1.7 foi colocada em nível baixo. Mostrar este valor em decimal na LINHA1 e em hexadecimal na LINHA 2.
10. Incrementar o conteúdo da posição de memória 20H de uma unidade cada vez que a porta P1.7 baixar para o nível zero. Se o bit P1.6 for para o nível baixo este valor deve ser decrementado em uma unidade. Mostrar o conteúdo desta posição de memória no vídeo e no display.
11. Fazer um programa que gere uma onda quadrada na saída P1.7 usando o temporizador T0.
12. Gerar uma forma de onda retangular na saída P1.6 do 8031. A onda retangular deve ter um nível alto durante 25 microsegundos e um nível baixo durante 40 microsegundos.
13. Gerar uma forma de onda retangular na saída P1.6 do 8031. A onda retangular deve ter um nível alto durante 350 milisegundos e um nível baixo durante 180 milisegundos.
14. Gerar uma onda quadrada na saída P1.7 usando o temporizador T0. O programa principal deve ler um número no teclado, carregar para o temporizador (THx) e mostrar o valor hexadecimal da tecla no terminal e no LCD.
15. Usar o timer para gerar uma onda quadrada P1.7 com freqüência variável e proporcional aos 4 bits menos significativos de P1.

16. Fazer um programa que possa medir a freqüência de um sinal colocado na saída P1.7. Quais são os limites e erros nestas medições.
17. Medir o intervalo de tempo em que a onda retangular gerada no exercício 7 permanece em nível alto. Quais são os limites existentes nesta medida de tempo ?
18. Fazer a leitura de 8 números da porta P2 e colocar os valores na memória interna a partir do endereço 40H. Classificar estes números por ordem crescente e coloca-los nesta ordem a partir da posição de memória 50H. As duas seqüências também devem ser mostradas no terminal de vídeo e no display LCD. Criar um arquivo com os dados para a sua utilização no simulador.
19. Ler um número na porta P1 que representa o número de bytes que devem ser somados a partir do endereço 2000H. Colocar o resultado nos endereços 20 e 21H da memória interna do microcontrolador

4 O kit de desenvolvimento WF

4.1 Apresentação

O sistema WF8031 v. 2.0 tem uma CPU 8031. Pode comunicar-se com um PC através do canal serial, através do qual é possível transferir programas desenvolvidos no PC para o kit. Este conta ainda com outros componentes, listados na tabela 2.2 juntamente com os endereços que ocupam.

Componente	Endereços
EPROM 2732 (8 kB)	0000H - 1FFFH
RAM 6264 (8 kB)	2000H - 3FFFH
LCD	4000H - 4002H

Tab. 4.1 - Componentes periféricos do kit WF

O display de cristal líquido (LCD) possui uma linha, dividida em duas áreas de oito colunas, conforme mostra a figura **2.2**.

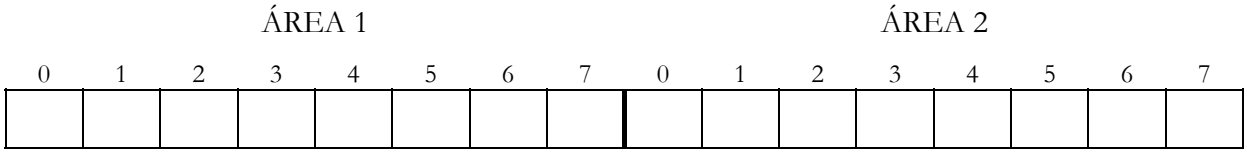


Fig. 4.1 – O Display de Cristal Liquido do Kit WF

4.2 Restrições

As portas P0 e P2, bem como os pinos P3.6 e P3.7, são usados para acessar a memória externa. Os bits P3.0 e P3.1 e o bit 7 do registrador PCON são utilizados na transmissão de dados pelo canal serial e portanto só podem ser utilizados por programas que, depois de carregados no kit, não façam mais uso da comunicação com o PC. Além disso, o timer 1 é utilizado para determinar a taxa de transmissão do

canal serial e portanto também só pode ser usado em outras aplicações se não houver necessidade de comunicação através do canal serial.

É preciso tomar muito cuidado com instruções que manipulem os registradores SCON, PCON, TMOD e TCON para evitar que as configurações do timer 1 e do canal serial não sejam alteradas involuntariamente. Se isto acontecer, a comunicação com o PC pode apresentar problemas.

4.3 – Rotinas do Monitor

Na EPROM existem diversas sub-rotinas que dão suporte à execução de tarefas tais como a entrada e saída de dados via terminal do PC e a apresentação de mensagens no LCD. Estas sub-rotinas estão listadas na tabela 2.4. A chamada dessas subrotinas é feita mediante a instrução LCALL <*nome da sub-rotina*>. Na tabela a seguir estão listados os nomes das rotinas, seus endereços, além de uma breve descrição de suas funções e o tempo de execução. Explique como você pode conferir se este tempo de execução está correto !!

Nome	Endereço	Descrição	tempo
APAGA	0089H	Apaga display	1,5ms
APAGAT	0196H	Apaga terminal de vídeo	1,5ms
LEBYTE	00D0H	Lê numero digitado no teclado (byte fica em a)	
NUMTER	0164H	Imprime numero no terminal, (valor de a)	1,1ms
NUMLCD	013EH	Imprime numero no display (valor de a)	40µs
CARLCD	0071H	Imprime caracter no display (valor de a)	60µs
AREA1	00^{A}4H	Posiciona cursor do display na área 1, (coluna em A)	40µs
AREA2	00^{A}DH	Posiciona cursor do display na área 2, (coluna em A)	40µs
MENLCD	00B6H	Imprime mensagem no display (inicio=DPTR, 0 =fim)	64µs/D
APACUR	009BH	Apaga cursor do LCD	
ACECUR	01BDH	Acende cursor do LCD	
HOMCUR	0092H	Home cursor do LCD	
OUTLCD	0059H	Comando para LCD	
CARTER	01^{A}AH	Imprime caracter no terminal de vídeo (dado de A)	1,1ms/D
MENTER	0118H	Imprime mensagem n terminal de vídeo, inicio=DPTR	1,1ms/D
LENUM	00DDH	Lê numero digitado no terminal,(numero em A)	
LEMENS	00F3H	Lê mensagem digitada no terminal, (DPTR=endereço)	
TECPRES	01C6H	Faz CY = 1 se alguma tecla foi pressionada	
CODTECL	01DAH	Lê o código da tecla pressionada, (valor em A)	

Tab. 4.2 – Subrotinas da Eprom do Kit Wf

4.4 Alguns endereços importantes

Além dos endereços das sub-rotinas apresentadas acima, são importantes os endereços listados na tabela 4.6.

Descrição	Endereço
Início da RAM	2000H
Redirecionamento da interrupção externa 0 (P3.2)	2003H
Redirecionamento da interrupção do timer 0	200BH
Redirecionamento da interrupção externa 1 (P3.3)	2013H
Redirecionamento da interrupção do timer 1	201BH

Tab. 4.3 – Redirecionamento das Interrupções

Tanto os endereços da tabela 4.4 quanto os da tabela 4.6 encontram-se definidos como sinônimos no arquivo ENDERECO.ASM. Convém incluir este arquivo no início de todos os programas desenvolvidos para o kit WF com a seguinte diretiva

```
%INCLUDE ENDERECO.ASM
```

colocada no início do arquivo-fonte.

O redirecionamento das interrupções, no simulador, é conseguido através das linhas finais do arquivo ENDERECO.ASM, reproduzidas abaixo. No KIT isto está gravado na EPROM.

```
        ORG 0003H           ;INT 0
        LJMP RAM+0003H      ;INT 0 redirecionada
        ORG 000BH           ;T0
        LJMP RAM+000BH      ;T0 redirecionada
        ORG 0013H           ;INT 1
        LJMP RAM+0013H      ;INT 1 redirecionada
        ORG 001BH           ;T1
        LJMP RAM+001BH      ;T1 redirecionada
        ORG RAM
```

Note que estas linhas não têm efeito sobre programas transferidos para o kit, pois este tem EPROM nos endereços modificados. A utilidade deste redirecionamento está em permitir a execução de programas no simulador. Por isso, programas que utilizem tratadores de interrupção devem seguir a estrutura dada abaixo:

```
        ORG RAM
        SJMP INICIO
        ORG 2003H
        ...                 ;tratador da interrupção externa 0
        ORG 200BH
        ...                 ;tratador da interrupção do timer 0
INICIO: ...                 ;instruções do programa
        END
```

Um outro problema que aparece no desenvolvimento de programas para o kit WF é que o simulador AVSIM51 não possui as sub-rotinas de serviço e portanto não pode simulá-las. Então, se um programa escrito para chamar sub-rotinas da tabela 4.4 for executado no simulador, é preciso tomar cuidado para que o programa não se perca quando saltar para uma dessas sub-rotinas. A solução adotada aqui consiste em incluir, no início do programa, o arquivo RET.ASM, que coloca instruções RET em todos os endereços listados na tabela 4.4. Desta forma, as chamadas das sub-rotinas são ignoradas durante a simulação e não atrapalham. Mais uma vez, este arquivo só tem efeito no simulador, pois o kit tem EPROM nesses endereços.

4.5 Desenvolvimento de Programas

Para fazer o desenvolvimento de programas para o 8031 são necessários:

- um editor, EDIT;

- um montador, AVMAC51;
- um linkador, AVLINK51;
- um conversor de arquivo HEX (texto) para BIN (binário), HEXBIN;
- um programa para transferir o arquivo do PC para o kit, LINK8031.

Para agilizar a compilação, pode-se criar um arquivo bat com o conteúdo listado abaixo.

```
avmac51 %1
@pause
avlink51 %1=%1
hexbin %1.hex %1.bin I
```

Supondo ser o nome dado a este arquivo av.bat, pode-se digitar

av <*nome do arquivo fonte*>

para criar o programa.

Para efetuar a transferência do programa para o kit, conecte-o ao PC pela interface serial e digite:

LINK8031 EXEMPLO *n*,

onde *n* identifica a porta serial utilizada pelo PC na comunicação, de acordo com a tabela 4.8. Quando estiver sendo utilizado o conector padrão RS-232C, a porta escolhida deve ser a primeira disponível.

n	Porta selecionada
1	03F0H (COM 1)
2	02F0H (COM 2)
3	03E0H (COM 3)
4	02E0H (COM 4)

Tab. 4.8 - Erro! A origem da referência não foi encontrada.

O LINK8031 pedirá então ao usuário que pressione o botão de RESET do kit, para que se faça a transferência do programa. Em seguida, o LINK8031 pede que o usuário pressione uma tecla do PC para iniciar a execução do programa no kit. Uma vez alcançado este estado, o PC passa a funcionar como terminal de entrada e saída para o programa carregado no kit. Para sair do modo terminal, pressionam-se as 2 teclas SHIFT simultaneamente.

4.6 Exemplos

4.6.1 Utilização da interrupção do timer 0

O programa abaixo utiliza o timer 0 para controlar a escrita das palavras "BOM" e "DIA" no LCD, que alternam a cada segundo. **Segundo.asm**

```
%INCLUDE ENDERECO.ASM        ;endereços das rotinas na EPROM
TIMER0H  EQU 10H             ;valores para o timer 0
TIMER0L  EQU 01H

         ORG RAM
         LJMP INICIO

;INTERRUPCAO DO TIMER 0
         ORG 200BH
         INC R0
         CJNE R0,#15,RECARGA ;15 Int perfazem 1 segundo
         MOV  R0,#00
         MOV A,#0            ;cursor na coluna 0
```

```
        LCALL AREA1          ;da área 1 do LCD
        CJNE R1,#0,DIA       ;seleciona a mensagem
BOM:    MOV DPTR,#MSGBOM
        LCALL MENLCD
        MOV R1,#1            ;alterna a mensagem
        SJMP RECARGA
DIA:    MOV DPTR,#MSGDIA
        LCALL MENLCD
        MOV R1,#0            ;alterna a mensagem
RECARGA:MOV TH0,#TIMER0H
        MOV TL0,#TIMER0L
        RETI

INICIO: LCALL APAGA
                             ;R0 CONTA OVERFLOWS DO TIMER 0
        MOV R0,#0            ;15 ESTOUROS = 1 segundo
        MOV R1,#0            ;R1 seleciona a msg a ser apresentada
        MOV TMOD,#1          ;timer 0 no modo de contagem 1
        MOV TH0,#TIMER0H     ;valor inicial do timer 0
        MOV TL0,#TIMER0L
        SETB ET0             ;habilita interrupção do timer 0
        SETB EA              ;habilita todas as interrupções
        SETB TR0             ;liga timer 0
        SJMP $
MSGBOM  DB "!!!!!BOM",0
MSGDIA  DB "DIA!!!!!",0
        END

; Comentários sobre a obtenção dos valores de TH0 e TL0
; A freqüência do cristal = 11.059MHZ==> Período = 90,424 ns.
; O timer é incrementado a cada ciclo de instrução (12 períodos de clock).
; Queremos uma pausa de 1 segundo.
; Logo, é preciso contar 921584 ciclos (1s / (12 * 90,424 * 10exp-9)).
; Como o limite de contagem do TIMER no modo 1 (16 bit) é 65536 seriam
; necessárias 14,062 contagens de 65536.
; Como o número de contagens deve ser inteiro, usamos 15 contagens.
; Cada contagem deve ser então de 921584 / 15 = 61439.
; Para tanto, o timer 0 deve partir de 65536-61439 = 4097 = 1001H.
; Logo, inicializamos TH0 com 10H e TL0 com 01H.
```

4.6.2 Leitura do Teclado

O programa abaixo (par.asm) lê números até 256 do teclado do PC. Os números ímpares aparecem na área 1 do LCD e os pares na área 2. O programa termina quando tiverem sido digitados 4 números do mesmo tipo (pares ou ímpares).

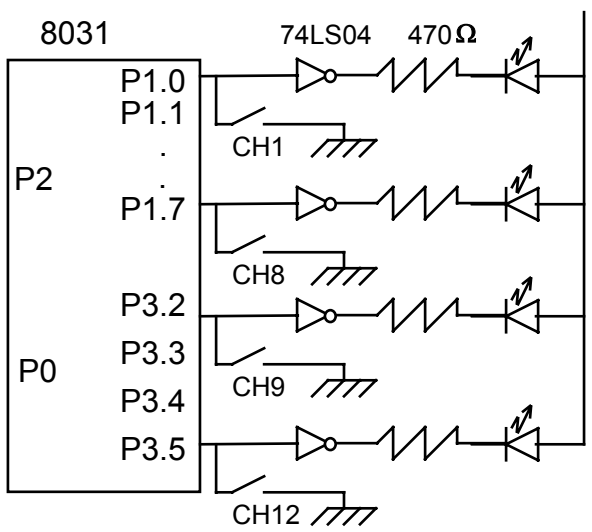
```
%INCLUDE ENDERECO.ASM        ;ENDERECOS DAS ROTINAS NA EPROM
        ORG RAM              ; ENDERECO DA RAM
        LJMP INICIO          ; INICIO DO PROGRAMA
        ORG 2003H            ; INTERRUPCAO INT0
        RETI                 ; SEM INTERRUPÇÃO
        ORG 200BH            ; INTERRUPCAO DO TIMER 0
        RETI                 ; NÃO USADO
INICIO: MOV DPTR,#MENS1
        CALL MENTER
        LCALL APAGA          ; LIMPAR DISPLAY
        MOV R1,#4            ; CONTADOR DE NÚMEROS IMPARES
        MOV R4,#4            ; CONTADOR DE NÚMEROS PARES
LACO:   LCALL LEBYTE         ; ENTRADA DO TECLADO
        CALL APAGA           ;
        PUSH ACC             ; SALVAR VALOR LIDO
        ANL A,#1             ; VER SE É PAR OU IMPAR
        CJNE A,#1,PAR        ; DESVIA SE PAR
IMPAR:  MOV A,#0             ; DEFINIR AREA1
        CALL AREA1
        POP ACC              ; RECUPERA VALOR LIDO
        CALL NUMLCD          ; MOSTRAR VALOR IMPAR
```

```
        DJNZ R1,LACO        ; CONTADOR
        MOV DPTR,#MENIMP    ; ATINGIU N IMPARES
        SJMP FIM
PAR:    MOV A,#0            ; POSICIONAR AREA2
        CALL AREA2
        POP ACC             ; RECUPERAR VALOR
        CALL NUMLCD         ; IMPRIMIR PARES NA AREA2
        DJNZ R4,LACO
        MOV  DPTR,#MENPAR   ; ATINGIU N PARES
FIM:    CALL APAGA
        CALL MENLCD         ; MOSTRA MENSAGEM DE FINAL ATINGIDO
        CALL MENTER
        SJMP $              ; PERMANECE PRESO NESTE LACO
MENS1   DB ' ENTRE COM UM NUMERO < 255 ',CR,LF
        DB ' O PROGRAMA IRA DETECTAR SE SAO PARES OU IMPARES ',CR,LF
        DB ' O PROGRAMA TERMINA COM A ENTRADA DE 4 DO MESMO TIPO',CR,LF,0
MENPAR  DB '4 PARES ',0
MENIMP  DB '4IMPARES',0
        END
```

4.6.3 Exercíos para fazer

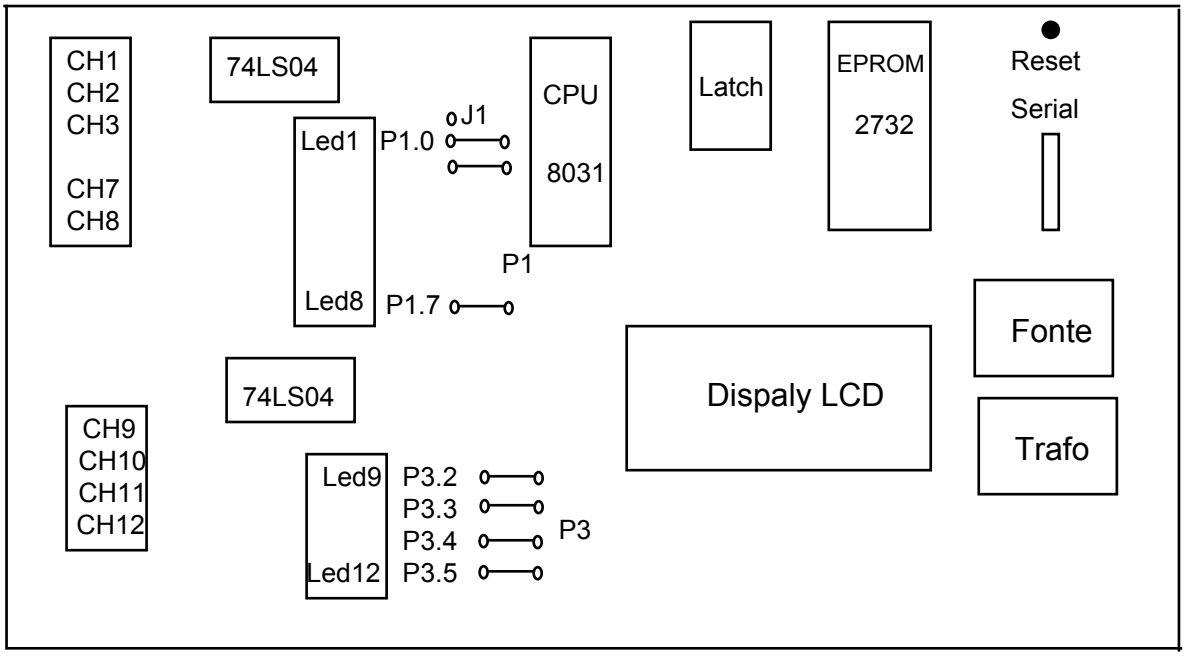
1) Fazer um programa para apagar as duas áreas do display e escrever a letra A na terceira coluna da área 1 e o numero 2 na 4 coluna da área 2.
2) Fazer um contador hexadecimal que coloque este valor na porta P1.
3) Alterar o contador acima para fazer uma contagem decimal e mostrar os valores de forma legível no display de cristal líquido e no terminal de vídeo (incluir um delay) para permitir a leitura.
4) Mostrar o conteúdo de 16 posições de memória, a partir de uma posição de memória solicitada ao operador pelo teclado, no terminal de vídeo. Defina uma tecla para avançar para a linha seguinte e outra para retroceder.
5) Fazer um programa que gere uma onda quadrada na porta P1.7.
6) Gerar uma onda quadrada, em P1.0, com a maior freqüência possível. Qual é esta freqüência ?
7) Fazer um programa que gere uma onda retangular na porta P1.7.
8) Fazer um programa para ler a porta P1 e mostrar o resultado em hexadecimal, binário e decimal no terminal de vídeo.
9) Contar o numero de vezes que a saída P1.7 foi colocada em nível baixo. Mostrar este valor em decimal na área 1 e no vídeo e em hexadecimal na área 2 e no vídeo.
10) Incrementar o conteúdo da posição de memória 20H de uma unidade cada vez que a porta P1.7 baixar para o nível zero. Se o bit P1.6 for para o nível baixo este valor deve ser decrementado em uma unidade. Mostrar o conteúdo desta posição de memória no vídeo e no display.
11) Fazer um programa que gere uma onda quadrada na saída P1.7 usando o temporizador T0.
12) Gerar uma forma de onda retangular na saída P1.6 do 8031. A onda retangular deve ter um nível alto durante 25 microsegundos e um nível baixo durante 40 microsegundos.
13) Gerar uma forma de onda retangular na saída P1.6 do 8031. A onda retangular deve ter um nível alto durante 350 milisegundos e um nível baixo durante 180 milisegundos.
14) Gerar uma onda quadrada na saída P1.7 usando o temporizador T0. O programa principal deve ler um número no teclado do PC, carregar para o temporizador (THx) e mostrar o valor hexadecimal da tecla no terminal e no LCD.
15) Usar o timer para gerar uma onda quadrada P1.7 com freqüência variável e proporcional aos 4 bits menos significativos de P1.

ESQUEMA DA PLACA DE LEDS E CHAVES PARA EXPANSÃO DO KIT 8031



O circuito ao lado permite utilizar as portas como entradas e saídas.
Se a chave estiver aberta o LED indica o estado da saída.
Se a chave estiver fechada o LED estará apagado independente do valor da saída.
Com a chave aberta a entrada estará em alto e com a chave fechada a entrada estará em nível baixo.

LAY-OUT DO KIT 8031 COM A PLACA DE EXPANSÃO



A placa com os Leds e as chaves está sobrecarregando a fonte e fazendo com que a tensão de alimentação caia a ponto de impedir a transmissão serial para o PC. Por isso deve-se retirar o Jumper J1 para fazer o Download do programa e após recolocar os jumper caso seja necessário.