

Capítulo 12

Projeto 5 – Controle de Motores de Passo

A crescente popularidade dos motores de passo deve-se a sua adaptação à lógica digital. Estes dispositivos são usados em inúmeras aplicações, tais como: mesas XY, periféricos de computadores (unidades de disco, impressoras, scanners, plotters), célula de manufatura integrada e sistemas robóticos (de manipulação e móveis). Nesta última aplicação, com o auxílio dos motores de passo, pode-se criar interfaces entre o PC e o robô de modo a realizar o movimento desejado.

Neste projeto é apresentado um circuito controlador de motores de passo usando o microcontrolador PIC 16F84.

12.1 Circuito e Funcionamento

Geralmente, os motores comuns (por exemplo, de corrente contínua, de corrente alternada, de indução) possuem apenas dois estados de operação: parado ou em rotação. Quando os motores estão em rotação, o giro é feito com velocidade constante. Entretanto, os motores de passo têm três estados de operação: parado, ativado com travamento do rotor (bobinas energizadas) ou giro em etapas. O movimento dos motores pode ser brusco ou suave, dependendo da frequência e da amplitude dos passos em relação ao estado inercial. Baseando-se nisso, os motores de passo recebem uma classificação especial em relação aos comuns, sendo adequados àquelas situações em que se necessita ter o controle preciso do movimento, a partir de sinais provenientes de um circuito controlador. Os sinais enviados ao motor pelo circuito controlador devem obedecer a uma ordem específica de pulsos e estarem perfeitamente sincronizados.

Observando-se o diagrama esquemático da Figura 12.1, verifica-se que o circuito controlador e os motores de passo (Motor 1 - J2 e Motor 2 - J3) são alimentados com +12V_{CC} (J1). A tensão de alimentação do circuito controlador, ao passar pelo regulador de tensão (78L05 - U1), é transformada em +5V_{CC}, habilitando, assim, o microcontrolador PIC 16F84 (CI1).

Como os motores de passo trabalham com +12V_{CC} e o microcontrolador PIC 16F84 gera pulsos de +5V_{CC}, é necessário amplificar o sinal de tensão usando-se um driver de corrente (ULN2803A - CI3). Esse driver de corrente transforma os pulsos de +5V_{CC} provenientes do microcontrolador PIC 16F84 em pulsos de +12V_{CC}, possibilitando o funcionamento dos motores de passo.

Para realizar a interface de comunicação serial entre o PC e o circuito controlador, utiliza-se o conversor de nível de tensão padrão RS232 para TTL/CMOS (MAX232 - CI2) acrescido de quatro capacitores (1 μF / +50V_{CC}) em conjunto com o conector DB-9 (X1). O circuito integrado dedicado MAX232 estabelece a conversão dos níveis de tensão do microcontrolador PIC 16F84 (0V_{CC} e +5V_{CC}) para os níveis de tensão da porta serial do computador padrão RS232 (-12V_{CC} e +12V_{CC}) e vice-versa.

A lista de materiais para a implementação do circuito controlador está relacionada na Tabela 12.1.

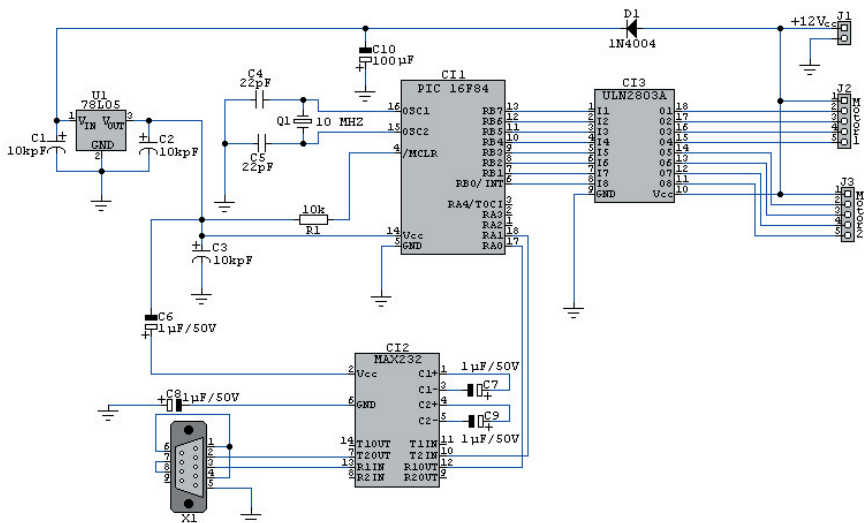


Figura 12.1 – Diagrama esquemático do circuito controlador de motores de passo.

12.2 Programação do Microcontrolador PIC 16F84

Após a declaração das variáveis, o programa utilizado no microcontrolador PIC 16F84 é desviado para a rotina de início. A princípio, seleciona-se o banco 0 (zero) e configura-se o registrador PORTA com o bit 0 (pino 0 da porta A) como entrada e os outros bits como saída. Todos os bits do registrador PORTB (pinos da porta B) são também configurados como saída. No registrador de opções (OPTION REG) configuram-se os resistores pull-ups internos para a porta B, visto que outras configurações são irrelevantes para a aplicação. No registrador INTCON habilita-se somente o bit 5 relacionado ao tratamento de interrupções do registrador contador de 8 bits (TMR0). Na seqüência, seleciona-se o banco 1 para começar o tratamento dos dados.

Tabela 12.1 – Materiais referentes ao circuito controlador de motores de passo

Quantidade	Tipo	Nomenclatura	Designação
3	10 KpF	C1 a C3	Capacitores
2	22 pF	C4 e C5	Capacitores
4	1 μ F / 50 V	C6 a C9	Capacitores
1	100 μ F	C10	Capacitor
1	1N4004	D1	Diodo
1	Conector	J1	Alimentação do circuito
2	Conectores	J2 e J3	Motores de passo 1 e 2
1	10 MHZ	Q1	Cristal
1	10 K Ω	R1	Resistor
1	78L05	U3	Transistor
1	DB-9	X1	Conector
1	PIC16F84	C11	Microcontrolador
1	MAX232	C12	Conversor de nível de tensão
1	ULN2803A	C13	Driver de corrente

Inicializam-se as variáveis de estado dos motores com 'D', de modo que, ao energizar o circuito, os mesmos estejam desligados. As variáveis de sentido dos motores são inicializadas com 'H', para que quando os motores forem ligados girem no sentido horário. As variáveis de TIMER, que definem as pausas entre os pulsos, são inicializadas com o valor 1 (menor pausa entre os pulsos, maior velocidade de rotação do motor). As variáveis auxiliares de tempo também ser-

vem para definir as pausas, as quais são incrementadas de 1 em 1. Quando tais variáveis atingem o valor igual ao valor contido nas variáveis de `TIMER`, o pulso é gerado e elas são reinicializadas. Zera-se o conteúdo das variáveis restantes e ativa-se a interrupção do registrador contador de 8 bits (`TMR0`).

Inicializa-se o registrador contador de 8 bits (`TMR0`) com o valor 190, pois este registrador estoura a cada 255 passos, o que acarreta em aproximadamente 510 ciclos de máquina. Cada ciclo corresponde a cerca de 400 ns. Para trabalhar com a velocidade de 9.600 bauds na comunicação serial, é necessário que cada bit recebido seja lido em 104 μ s aproximadamente. Logo, é preciso que a porta serial seja lida em um intervalo bem menor, possibilitando a recepção e o tratamento dos dados sem perda de bits. Com o registrador contador de 8 bits (`TMR0`) sendo inicializado em 190, tem-se somente mais 65 passos para o estouro do mesmo. Isso resulta em um tempo de aproximadamente 52 μ s para cada estouro, o que é bastante razoável.

Nesse ponto do programa, entra-se na rotina principal, chamada de rotina responsável pela geração de passos. Terminada a sua execução, volta-se para a rotina principal, onde o programa ficará em um laço (loop).

A rotina de geração de passo verifica, por meio das variáveis de estado, se o motor 1 está ligado. Caso o motor não esteja ligado, testa-se o motor 2 e depois vai para a rotina de pausa (que gasta aproximadamente 4 ms). Logo, se um dos motores estiver ligado, é verificado se é o momento da geração de passo ou de se aguardar mais um tempo. Se for o momento da geração de passo, verifica-se o sentido de rotação dos motores e desvia-se para as rotinas específicas (rotinas de sentidos horário e anti-horário).

Escolhido o sentido de rotação dos motores, chama-se a rotina de geração de passo. Esta rotina gera o primeiro passo, pulso a pulso, alterando somente os bits desejados do registrador `PORTB` (pinos da porta B), pois, se todos os bits fossem alterados, ocorreria o travamento do outro motor. Ao final, é colocado o valor que define o próximo passo a ser executado. O esquema de funcionamento dos motores dá-se nos dois sentidos de rotação, mas, nesse ponto do programa, surge a seguinte pergunta: como é realizada a comunicação serial?

As rotinas pertinentes à comunicação serial são dependentes do estouro da interrupção do registrador contador de 8 bits (`TMR0`). Na ocorrência de uma interrupção, há um desvio no programa para o tratamento da mesma. Após o tratamento, retorna-se ao ponto onde o programa havia sido interrompido.

Dentro do tratamento da interrupção, as variáveis de ambiente são salvas para não alterar o funcionamento do programa. Na seqüência, é verificada a ocorrência do start bit (bit inicial de qualquer comunicação serial). Caso tenha ocorrido o start bit, desvia-se para a rotina de recepção e tratamento dos dados.

No registrador W coloca-se o valor 8 (o qual determina a quantidade de bits a serem recebidos) e verifica-se a chegada do start bit no registrador PORTA (porta A). Recebido o start bit, o programa rotaciona o byte (que identifica o tipo de tarefa a ser realizada) para a direita. A variável AUX recebe o valor do registrador W e, logo em seguida, é chamada a rotina de pausa (que gasta aproximadamente 104 μ s devido à velocidade de 9.600 bauds para a comunicação serial). Então, é verificado se o bit recebido é 0 (zero) ou 1 (um) para poder escrevê-lo corretamente no byte. Decrementa-se a variável AUX (que recebeu o valor 8 porque o byte equivale a 8 bits) e é verificado se ela é zerada. Se a variável AUX ainda não zerou, repete-se a operação de recepção de bit; caso contrário, desvia-se para a operação que aguarda a chegada do stop bit. Enquanto o stop bit não for recebido, a rotina não sai do laço (loop).

Após terminar a recepção do stop bit, verifica-se qual tarefa foi recebida: tarefa A para a recepção de tarefa comum e B para recepção de tarefa que depende de outro dado (como a tarefa de alteração de velocidade, a qual recebe primeiro o tipo de tarefa e depois o novo valor de velocidade do motor). Se a tarefa recebida for comum, desvia-se para a decodificação da tarefa; caso contrário, desvia-se para a recepção do byte restante.

A decodificação das tarefas é realizada de um modo bem simples: move-se o valor correspondente da tarefa a ser verificada para o registrador W e faz-se uma operação OU-EXCLUSIVO entre o valor do registrador W e o byte recebido. Para saber se houve igualdade entre os dois valores, verifica-se o bit Z do registrador de estados (STATUS). Se a operação for falsa, então desvia-se para o próximo teste; senão há uma alteração nas variáveis de estado que controlam o motor.

Como pode ser visto na Tabela 12.2, tem-se a variável STATUS_MOTOR que realiza o controle liga/desliga dos motores, a variável SENTIDO_MOTOR que realiza o controle do sentido de rotação dos motores e a variável TIMER_MOTOR que procede ao controle de pausa entre os pulsos dos motores.

Tabela 12.2 – Variáveis de estado do programa do PIC 16F84 para o controle de motores de passo

Variáveis de estado	Função	Byte correspondente
STATUS_MOTOR1	Liga/Desliga Motor 1	“L”/ “D”
STATUS_MOTOR2	Liga/Desliga Motor 2	“L”/ “D”
SENTIDO_MOTOR1	Horário/Anti-Horário Motor 1	“H”/ “A”
SENTIDO_MOTOR2	Horário/Anti-Horário Motor 2	“H”/ “A”

Ao término da decodificação e da atualização das variáveis, há um retorno para a rotina de interrupção, a qual reinicializa o registrador contador de 8 bits (TMR0) com o valor 190 e devolve os valores dos registradores W e de estados (STATUS) com o intuito de dar continuidade ao programa a partir do ponto em que foi interrompido.

12.3 Software Controlador no PC

O software desenvolvido para Windows tem uma interface amigável e auto-explicativa, como mostra Figura 12.2.



Figura 12.2 – Interface do software controlador para Windows.

No item da interface ‘Chave-Geral’ há o controle liga/desliga de ambos os motores. No item da interface ‘Motor 1’ (idem ‘Motor 2’) há o controle individual para esse motor contendo quatro chaves funcionais: liga, desliga, inversão do sentido de rotação do motor e alteração de velocidade. O controle de velocidade funciona de tal maneira que, quanto maior o valor colocado (entre 1 e 127), menor será a velocidade do motor, pois esse valor refere-se ao intervalo entre o pulso das bobinas do motor. Existe ainda um controle no canto inferior esquerdo que permite ao usuário definir qual porta serial que deseja utilizar.

O funcionamento do software consiste basicamente no envio de bytes para a porta serial. Para facilitar a sua compreensão (ver Tabela 12.3), o seguinte exemplo é fornecido: caso seja desejável ligar o motor 1, deve-se clicar no botão liga deste motor, de modo a enviar um byte ('C') à porta serial. Este byte é recebido e decodificado pelo microcontrolador para efetuar a tarefa LIGA do motor 1. No caso da alteração de velocidade do motor 1, dois bytes são enviados à porta serial, sendo que o primeiro deles informa o tipo de tarefa (alteração de velocidade) e o segundo define o novo valor de velocidade.

Tabela 12.3 – Relação de bytes enviados do PC para o PIC 16F84

Bytes	Função
A	Alteração de velocidade do Motor 1 (1º byte = tarefa, 2º byte = dado)
B	Alteração de velocidade do Motor 2 (1º byte = tarefa, 2º byte = dado)
C	Liga Motor 1 (STATUS_MOTOR1= "L")
D	Liga Motor 2 (STATUS_MOTOR2= "L")
E	Desliga Motor 1 (STATUS_MOTOR1= "D")
F	Desliga Motor 2 (STATUS_MOTOR2= "D")
G	Sentido do Motor 1 (SENTIDO_MOTOR1 = "H" ou "A")
H	Sentido do Motor 2 (SENTIDO_MOTOR2 = "H" ou "A")
I	Liga motores (1 e 2) simultaneamente (STATUS_MOTOR = "L")
J	Desliga motores (1 e 2) simultaneamente (STATUS_MOTOR = "D")