INSTITUTO DE ENSINO SUPERIOR DE BRASÍLIA (IESB)

CHARLES HENRIQUE GONÇALVES SANTOS

VIABILIDADE DE UMA IMPRESSORA BRAILLE DE BAIXO CUSTO

BRASÍLIA (DF)
DEZ/2008

CHARLES HENRIQUE GONÇALVES SANTOS

VIABILIDADE DE UMA IMPRESSORA BRAILLE DE BAIXO CUSTO

Trabalho de Graduação apresentado ao Instituto de Ensino Superior de Brasília (IESB) como requisito para obtenção do título de Engenheiro de Computação.

Orientador: MSc. Saulo Pimentel Wulhynek

BRASÍLIA (DF)
DEZ/2008

CHARLES HENRIQUE GONÇALVES SANTOS

VIABILIDADE DE UMA IMPRESSORA BRAILLE DE BAIXO CUSTO

Trabalho de Graduação apresentado ao Instituto de Ensino Superior de Brasília (IESB) como requisito para obtenção do título de Engenheiro de Computação.

INSTITUTO DE ENSINO SUPERIOR DE BRASÍLIA (IESB)
DEPARTAMENTO DE ENGENHARIA

Data de aprovação	MSc. Saulo Pimentel Wulhynek
MSc. Adriana de Carvalho Drummond	MSc. Luis Gustavo de A. Carvalho

Ao Programa Universidade para Todos – ProUni.

Aos deficientes visuais, inspiração para este trabalho.

RESUMO

Na Era da Informação, o acesso a determinadas tecnologias ainda é privilégio de poucos. Estão excluídos destas tecnologias, por exemplo, aqueles que não detêm todas as faculdades sensoriais, como os deficientes visuais. Embora esforços tenham sido realizados no sentido de reduzir esse quadro de exclusão, notadamente no que se refere ao desenvolvimento de sistemas amplificadores de telas, de saída de voz e de saída em Braille, o acesso deste grupo a determinados tipos de informação ainda é limitado.

Nesse sentido, e com o propósito de possibilitar aos deficientes visuais o acesso a recursos já disponíveis para a população em geral, foi desenvolvido o estudo sobre a Viabilidade de Produção de Impressora Braille de Baixo Custo, a partir de impressoras já em desuso. Para tanto, foram necessários conhecimentos sobre normas e aplicações do sistema Braille para a Língua Portuguesa, programação de sistemas embarcados, estudos sobre o funcionamento e mecânica de impressoras a jato de tinta e aplicação de diversos componentes de eletrônica analógica/digital tais como microcontroladores, motores de passo, motores de corrente contínua, solenóides, servo-motores, sensores, codificadores lineares e interfaces paralela/serial.

Os resultados mostram que, em termos técnicos e econômicos, a produção de uma impressora Braille de baixo custo a partir do aproveitamento do sistema mecânico de impressoras em desuso é possível. Este processo pode ser realizado por meio da adaptação da cabeça de impressão com uso de solenóide ou servo-motores e da substituição da controladora lógica.

Palavras-chave: sistema Braille, deficiência visual, impressoras, solenóides, servo-motores.

ABSTRACT

On the Information Era, the access to certain technologies is a privilege to few people. Those who don't have all the human senses, as a visual deficient are excluded from these technologies. Although some efforts have been done to reduce this exclusion problem, especially referring to the development of screen amplifier systems, voice output and Braille output, the access of this group to determined types of information is still limited.

In this sense and with the purpose of giving, to the visual deficient's, resources already available to people in general, a study was developed aiming the feasibility of production of low cost Braille Printer, from unused pieces. To that, knowledge of rules and applications of the Braille System for Portuguese Language, embedded systems programming, study of the functioning and mechanics of inkjet printers have been undertaken. In addition, application of several analogic/digital electronic components as microcontrollers, step-motors, DC motors, solenoids, servo-motors, sensors, linear encoders and parallel/serial interfaces have been used.

The results show that, technically and economically, the production of a low cost printer for a Braille system from mechanical use of unused printers is possible. This process can be carried out through the adaptation of the printer head with the use of solenoid or servo-motors and the substitution of the logical controller.

Key words: *Braille system, visual deficiency, printers, solenoids, servo-motors.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

INTRODUÇÃO

O mundo vive a Era da Informação. A população mundial é bombardeada por notícias instantâneas, independente se estas acontecem com um ente próximo no Brasil ou um desconhecido no Japão. Os recursos existem e, como forma de evitar a exclusão social, deseja-se que os meios tecnológicos viabilizem o acesso à informação para todos. Este desejo está longe de ser a realidade do Brasil, mesmo levando em consideração somente pessoas sem limitações de qualquer natureza. O que dizer então das pessoas que sofrem alguma limitação?

Uma das características da economia global é a valorização da alta tecnologia, que tanto pode promover a inclusão como a exclusão digital, reforçando as desigualdades. Isso porque somente o indivíduo que tem acesso a ela ou que pode produzi-la é valorizado. Sob esse aspecto, pode-se dizer que o fator desenvolvimento tecnológico é inversamente proporcional à igualdade social [1].

De acordo com a Organização Mundial de Saúde, os portadores de algum tipo de deficiência representam um décimo da população brasileira. Destes, são portadores de deficiência visual apenas 0,5% (Tabela 1).

Tabela 1: Distribuição das diferentes áreas de deficiência no Brasil [2, adaptado]

ÁREAS DE DEFICIÊNCIA	POPULAÇÃO	PORCENTAGEM
Deficiência Mental	7.250.000	5,00%
Deficiência Física	2.900.000	2,00%
Deficiência Auditiva	2.175.000	1,50%
Deficiência Múltipla	1.450.000	1,00%
Deficiência Visual	725.000	0,50%
TOTAL	**14.500.000**	**10,00%**

Embora numericamente sejam minoria, os deficientes visuais – que compreendem, além dos cegos, aqueles que possuem visão subnormal[*] – são os que têm maiores dificuldades em termos de manipulação da informação.

De fato, se entendermos a interface de *software* como uma superfície de contato entre o lado humano e o lado do sistema computacional, as interfaces com os sistemas computacionais que temos hoje são essencialmente dependentes do bom funcionamento de nossos sistemas perceptual, cognitivo e motor. Fazemos uso principalmente da visão – para leitura da tela, e do sistema motor – para uso do teclado e do *mouse*. Pessoas com deficiências nesses sistemas têm o acesso à informação bastante dificultado [3].

[*] Cego é aquele de dispõe de 20/200, ou 10%, de visão no melhor olho, após correção; e portador de visão sub-normal, aquele que dispõe de 20/70 (quase 30%) nas mesmas condições [37].

Alguns esforços isolados têm contribuído sobremaneira para que pessoas com limitações visuais possam usufruir dos benefícios oferecidos pela computação pessoal. Cita-se, como exemplo, o DOSVOX[*], *software* gratuito capaz de falar alguns textos ao usuário durante o uso do computador. Outro que merece destaque é o *ViaVoice*[†], *software* comercial que, além de falar ao usuário, permite que o mesmo entre com os comandos por meio da voz, beneficiando tanto os deficientes visuais quanto alguns deficientes físicos. Esses sistemas fazem parte de um conjunto maior de sistemas desenvolvidos para pessoas com deficiência visual envolvendo *hardware*, *software* e outros tipos de equipamento. Podem ser classificados em: sistemas amplificadores de telas, sistemas de saída de voz (como nos exemplos citados), e sistemas de saída em Braille – impressoras e terminais de acesso [3].

No entanto, não existe material que favoreça ou facilite o acesso desse grupo (deficientes visuais) a determinados tipos de informação e recursos já disponíveis para a população geral, como uma impressora Braille a um preço acessível. No mercado formal uma impressora de finalidade similar tem preço elevado, chegando ao valor de um veículo popular (ANEXO A).

Assim, a prova da viabilidade de uma impressora Braille de baixo custo – objetivo deste trabalho – possibilitará aos deficientes visuais, que já têm acesso a um computador pessoal, a impressão de seus próprios trabalhos. A tarefa de impressão, aparentemente corriqueira para os usuários de computadores sem limitações visuais, não é tão evidente para os deficientes visuais. Estes precisam se deslocar às escolas ou instituições específicas[‡] para ter seus documentos impressos. Além disso, este trabalho também propõe o aproveitamento de impressoras a jato de tinta do fabricante Hewlett-Packard (HP), já em desuso, que provavelmente virariam lixo eletrônico.

Em sentido acadêmico, o estudo da viabilidade de uma impressora Braille de baixo custo envolve atividades como: estudo das normas e aplicações do sistema Braille para a Língua Portuguesa, estudos sobre o funcionamento e mecânica de impressoras a jato de tinta e aplicação de componentes como solenóide linear, motor de passo, motor de corrente contínua, servo-motor, sensores, codificadores lineares, componentes de eletrônica analógica/digital, interface paralela e interface serial. Para a união destes componentes é necessário o conhecimento de eletrônica básica, programação em sistemas embarcados na linguagem *assembly* ou *C* para microcontroladores da família PIC 16F877A e utilização de aferidores básicos de medidas (multímetro, paquímetro, etc.).

---

[*] UFRJ (http://nce.ufrj.br/aau/dosvox)
[†] IBM (http:// www.ibm.com/speech/demo)
[‡] Em Brasília, por exemplo, tem-se conhecimento da existência de apenas uma escola que possui impressora Braille, qual seja, Centro de Ensino Especial de Deficientes Visuais (CEEDV), situado na Av. L2 sul, Quadra 612 projeção J – Área Especial.

Objetivo

O objetivo deste trabalho é avaliar a viabilidade técnica e econômica de se produzir uma impressora Braille de baixo custo, a partir do aproveitamento do sistema mecânico de impressoras em desuso, com a conseqüente redução dos custos de produção e eliminação de lixo eletrônico.

Estrutura do trabalho

O Capítulo 1 estuda o sistema Braille, como e por quem foi desenvolvido, as ferramentas existentes para escrever em Braille, as grafias existentes no sistema Braille e as distância de conforto para impressões em Braille.

O Capítulo 2 trata das diferentes tecnologias para impressão, dando ênfase nos estudos da impressora a jato de tinta quanto ao funcionamento da cabeça de impressão, conjunto de alimentação do papel e unidade lógica de controle.

O Capítulo 3 estuda conceitos das interfaces de comunicação da porta paralela e da porta serial com o circuito integrado MAX232. Deste estudo chegou-se a conclusão que a porta serial é a mais indicada, pois facilita sobremaneira uma possível migração para a porta USB.

O Capítulo 4 estuda os atuadores presentes na impressora, os quais sejam: motor de corrente contínua, motor de passo, solenóide e servo-motor. Além dos atuadores, foram estudados PWM, circuitos integrados L293D/ULN2003A para a integração dos atuadores ao PIC.

O Capítulo 5 estuda o funcionamento dos sensores presentes em uma impressora a jato de tinta, que são basicamente dois: o sensor de papel, que usa uma chave ótica, e o sensor de deslocamento do carro que usa um *photointerrupter* e um *encoder strip*.

O Capítulo 6 apresenta os microcontroladores de uma forma genérica, detalhando a arquitetura do PIC16F877A. Por der de extrema importância o conceito de interrupção em sistemas embarcados, as 14 interrupções deste microcontrolador foram especificadas.

O Capítulo 7 detalha o procedimento realizado para a execução do projeto, dando meios suficientes para que o mesmo possa ser reproduzido e melhorado por quem tenha as habilidades necessárias. Este capítulo relata os materiais utilizados, o porquê da impressora escolhida, o esquemático do circuito da controladora lógica bem como o procedimento para a sua montagem, a forma de adaptação do solenóide/servo-motor no cartucho de tinta e os fluxogramas e código fonte do *firmware* desenvolvido.

O Capítulo 8 relata como foram realizados os testes e apresenta um comparativo entre impressões Braille em diferentes tecnologias. Trata ainda sobre a vida útil de uma cabeça de impressão e sobre as dificuldades encontradas na execução do projeto.

No capítulo seguinte apresenta-se a conclusão do projeto, um comparativo de custos para a produção em série deste protótipo com uma impressora nova e algumas recomendações para trabalhos futuros.

1 SISTEMA BRAILLE

1.1 História do Braille

O sistema Braille é um código universal de leitura tátil e de escrita, inventado na França por Louis Braille, em 1825. Louis, aos três anos de idade, adquiriu a deficiência visual ao ferir-se com um instrumento de trabalho do pai, produtor de selas. Aos dez anos iniciou os estudos na escola para cegos de Paris e, aos 15 anos, dedicou-se a encontrar um sistema que possibilitasse ao cego escrever em relevo. É curioso constatar que, no processo de criação do sistema Braille, Louis foi inspirado pelos desenhos em relevo que enfeitavam as selas confeccionadas pelo pai com o instrumento que o cegou [4].

Ainda estudante, conheceu uma invenção denominada sonografia (ou código militar), cujo objetivo era possibilitar a comunicação noturna entre oficiais nas campanhas de guerra. Baseava-se em doze sinais de linhas e pontos em relevos, que representavam sílabas na língua francesa. A partir de então, Louis Braille começou a desenvolver estudos que resultaram em 1837 na estrutura básica do Sistema, utilizada mundialmente até os dias atuais [5].

O aparelho usado por Louis Braille consistia de uma prancha, régua com janelas[*], que se encaixam nas extremidades laterais da prancha, e a punção[†]. O papel é introduzido entre a prancha e a reglete e, pressionando-o com a punção, de trás para frente, escrevem-se os pontos em relevo (Figura 1.1). Os textos em Braille também podem ser produzidos com máquinas de datilografia especiais (Figura 1.2) ou com impressoras Braille comerciais (Figura 1.3).

Figura 1.1: Reglete e prancha

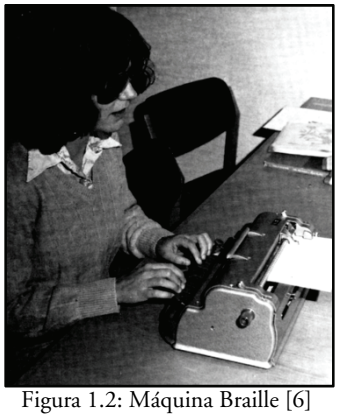


Figura 1.2: Máquina Braille [6]

Figura 1.3: Impressora Braille [6]

No Brasil, o sistema Braille foi adotado em 1854, com a criação do Imperial Instituto dos Meninos Cegos – conhecido hoje como Instituto Benjamin Constant (IBC). A partir de 1940,

[*] A reglete "é um dispositivo metálico ou plástico, constituído de uma placa frisada ou com cavidades circulares rasas e de uma régua ou placa com retângulos vazados, para a produção manual de sinais Braille" [7].
[†] A punção "é um estilete constituído de uma ponta metálica e de um cabo em plástico, madeira ou metal, usado especificamente para a produção de pontos em relevo em regletes" [7].

algumas modificações foram impostas, devido à reforma ortográfica da Língua Portuguesa. Pela ausência de definições governamentais, as alterações no Sistema ficaram à mercê dos esforços de professores, técnicos especializados no sistema Braille e de instituições ligadas à educação dos deficientes visuais.

1.2 O sistema Braille

O sistema Braille é composto por matrizes de seis pontos – três linhas por duas colunas, totalizando 63 representações[*] distintas. Cada matriz recebe o nome de cela e estas, lado-a-lado, formam as linhas táteis com algum significado lógico para os leitores do sistema Braille. Nota-se que 63 símbolos são insuficientes para representar toda a grandiosidade de uma língua qualquer. Para contornar esta limitação, alguns símbolos são representados por sinais compostos, o que aumenta sobremaneira as possibilidades de representação.

O sistema Braille pode ser empregado por extenso, ou seja, escrevendo-se palavra por palavra, letra por letra. Este sistema é conhecido como escrita Braille de grau 1 ou escrita Braille integral, e será a base para o desenvolvimento deste trabalho.

Existe a escrita Braille de grau 2, onde a representação de conjunções, preposições, pronomes, prefixos, sufixos e grupo de letras são representados por meio de uma ou mais celas Braille. A principal razão de seu emprego é reduzir o volume dos livros em Braille e permitir maior rendimento na leitura e escrita Braille [5]. Na Figura 1.4 pode-se observar a forma de codificação da escrita Braille, onde os pontos escuros representam o relevo da folha e os números são indicativos das posições dos relevos.

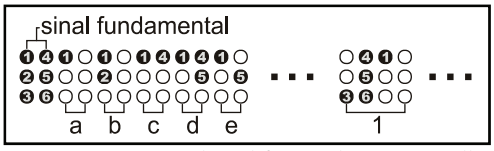


Figura 1.4: Forma de codificação da escrita Braille

1.3 Grafias do sistema Braille

O conjunto de seis pontos ⠿, numerados de um a seis – que é contado de cima para baixo e da esquerda para direita – é denominado símbolo fundamental, base para os demais símbolos (Figura 1.4). O espaço que ele ocupa, ou qualquer outro sinal variante dele, denomina-se cela Braille ou célula Braille. Há quem considere o espaço em branco como uma cela vazia, o que elevaria o número de representações do Sistema em uma unidade [7].

[*] Combinação dos seis elementos: $2^6 - 1 = 63$. A unidade excluída é a representação em branco.

Além dos possíveis graus de escrita, o sistema Braille está dividido em grafias com representações próprias. Ou seja, os lingüistas desenvolvem uma Grafia Braille para a Língua Portuguesa, os matemáticos desenvolvem uma Grafia Braille para a Matemática, os químicos desenvolvem uma Grafia Braille para a Química; os músicos desenvolvem uma Grafia Braille para a Música, etc. Inúmeras são as possibilidades de se representar o mundo real no sistema Braille [8]. Pode-se, inclusive, produzir diagramas táteis (Figura 1.5).

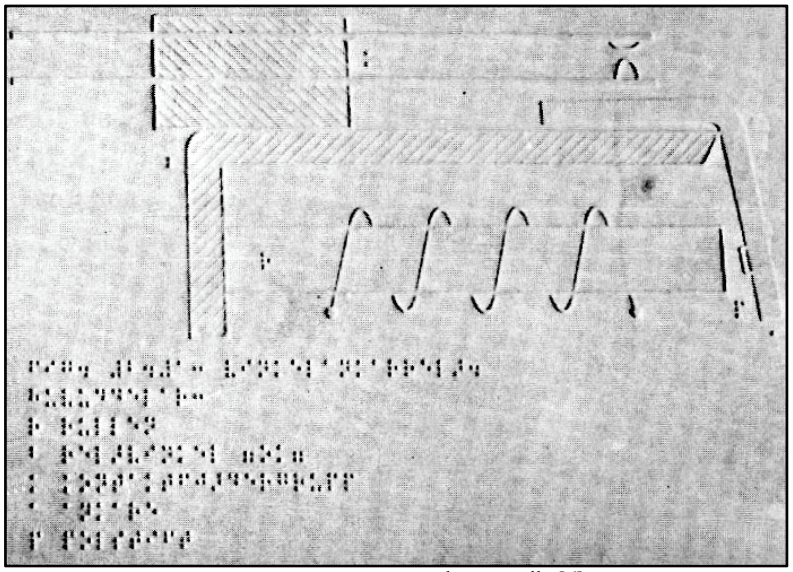


Figura 1.5: Diagrama tátil em Braille [6]

Para o nosso propósito, utilizaremos somente a Grafia Braille para a Língua Portuguesa. A correspondência entre código Braille e Língua Portuguesa é ilustrada no Quadro 1.1.

Quadro 1.1: sistema Braille da Língua Portuguesa [9, adaptado]

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**
[1]	[12]	[14]	[145]	[15]	[124]	[1245]	[125]	[24]	[245]
k	l	m	n	o	p	q	r	s	t
[13]	[123]	[134]	[1345]	[135]	[1234]	[12345]	[1235]	[234]	[2345]
u	v	w	x	y	z	ç	á	é	í
[136]	[1236]	[2456]	[1346]	[13456]	[1356]	[12346]	[12356]	[123456]	[34]
ó	ú	â	ê	ô	ã	õ	à	ü	Sinal de maiúscula
[346]	[23456]	[16]	[126]	[1456]	[345]	[246]	[1246]	[1256]	[46]
Sinal de número	Sinal de reticência	.	,	;	:	?	!	-	–
[3456]	[3][3][3]	[3]	[2]	[23]	[25]	[26]	[235]	[36]	[36][36]
0	1	2	3	4	5	6	7	8	9
[3456][245]	[3456][1]	[3456][12]	[3456][14]	[3456][145]	[3456][15]	[3456][124]	[3456][1245]	[3456][125]	[3456][24]

1.4 Distância de conforto

Para os videntes – pessoas com visão normal – existe um tamanho de letra que é considerado confortável que, segundo recomendações da Norma ABNT NBR 14724, é fonte com tamanho 12 e espaçamento entre linhas de 1,5. Assim também o é para o sistema Braille. A produção de textos em Braille com pontos muito próximos dificulta a percepção tátil, pontos muito distantes podem se perder em meio às celas. Algumas referências a essas distâncias já foram identificadas: Portal do Envelhecimento[*] (Figura 1.6a) e *American Braille Technical Specifications* (Figura 1.6b). Observa-se que a primeira especificação nada diz sobre a distância das linhas.

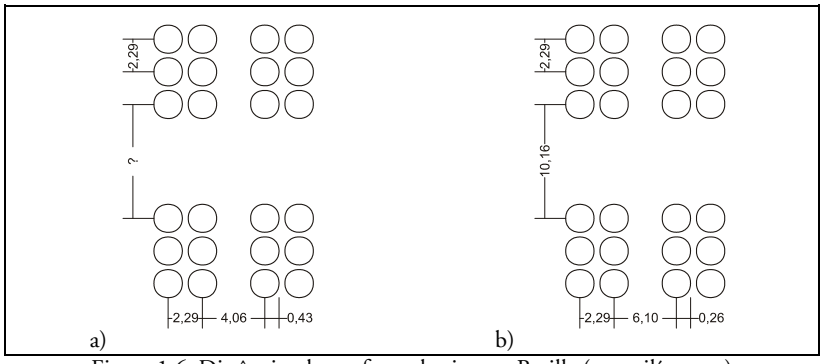


Figura 1.6: Distâncias de conforto do sistema Braille (em milímetros)

[*] http://www.portaldoenvelhecimento.net/artigos/artigo1378.htm, citado em: 25 de maio de 2008.

2 IMPRESSORAS

Existem diversas impressoras no mercado, classificadas de acordo com o mecanismo de impressão [10]:

- Impressoras matriciais: utilizam cabeça de impressão com uma série de agulhas que se movimentam de acordo com pulsos elétricos sobre uma fita de tinta, marcando o papel [10]. A tecnologia é antiga, mas ainda é utilizada por ser indispensável para a impressão de formulários contínuos e por ter baixo custo de impressão.
- Impressoras a jato de tinta: a cabeça de impressão borrifa tinta sobre o papel por meio de microconduítes [10].
- Impressoras a *laser:* também conhecida como impressora de página, utiliza um feixe *laser* para marcar, em um cilindro fotossensível, os pontos que devem receber tinta do toner [10].
- Impressoras Braille: funcionam de modo similar às impressoras matriciais, mas têm cabeça de impressão com agulhas maiores e não dependem de tinta, pois marcam o papel diretamente, formando leves relevos.

Das impressoras listadas, interessa-nos estudar a jato de tinta, pois apresenta preço acessível e é a mais comum no mercado.

Uma impressora a jato de tinta é composta por uma unidade central de processamento, um sistema para processamento de dados, uma ou mais cabeças de impressão, um mecanismo para carregamento de papel, uma ou mais interfaces de comunicação, três motores – dois de passo e um motor de corrente contínua – além de sensores para realimentar o sistema, também conhecidos como *feedback* (Figura 2.1).

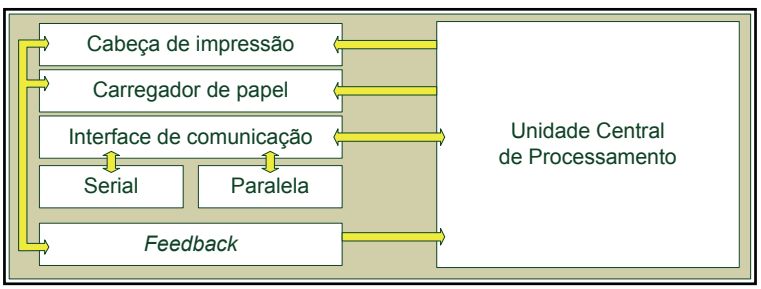


Figura 2.1: Diagrama de blocos de uma impressora a jato de tinta

Para entender o funcionamento de uma impressora a jato de tinta, é importante conhecer os termos técnicos e visualizar aspectos de componentes gerais que compõe o sistema. O site

norte-americano *How Stuff Works* (como as coisas funcionam) [11] especifica de forma satisfatória os termos e aspectos de uma impressora jato de tinta:

1. Conjunto da cabeça de impressão

   a. Cabeça de impressão: é a parte mais importante de uma impressora a jato de tinta. Contém uma série de esguichos que são usados para lançar gostas de tinta sobre o papel (Figura 2.2).

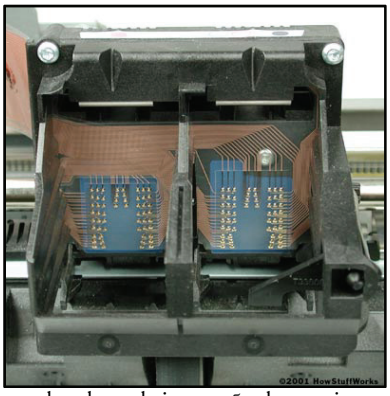


Figura 2.2: Compartimento da cabeça de impressão de uma impressora a jato de tinta [11]

   b. Cartuchos de tinta: as características dependem do fabricante e do modelo, podendo ter diferentes combinações.

   c. Motor da cabeça de impressão: um motor de corrente contínua move o conjunto da cabeça de impressão de um lado para o outro do papel. Como este motor não é de passo, precisa de um sistema de controle, o *encoder strip*, que será tratado adiante.

   d. Motor do sistema de repouso: algumas impressoras têm um sistema para estacionar[*] o conjunto da cabeça de impressão. Este motor é o responsável por esta movimentação.

   e. *Belt carriage*: é uma fita de borracha dentada que lembra muito a correia dentada de um veículo automotor; é usada para prender o conjunto da cabeça de impressão ao motor que a traciona (Figura 2.3).

[*] Entenda-se estacionar por mover o conjunto da cabeça de impressão para um local pré-determinado, de modo que seja impedida de se mover acidentalmente (tal como o freio de mão de um veículo automotor). Isto evita o vazamento da tinta e/ou o ressecamento quando a impressora fica inativa por longos períodos.

Figura 2.3: *Belt carriage* e *encoder strip* de uma impressora a jato de tinta [11]

f. *Encoder strip*: é uma fita fixa na carcaça da impressora que tem marcações intercaladas, cuja resolução é definida pelo número de linhas por polegada (ver Figura 2.3). A cabeça de impressão tem um foto-sensor que lê tais marcações, enviando os pulsos gerados para a unidade central de processamento.

2. Conjunto de alimentação do papel

   a. *Paper feeder:* é responsável pela alimentação de papel no sistema.

   b. *Rollers*: são responsáveis por puxar o papel do alimentador para o sistema de impressão (Figura 2.4).

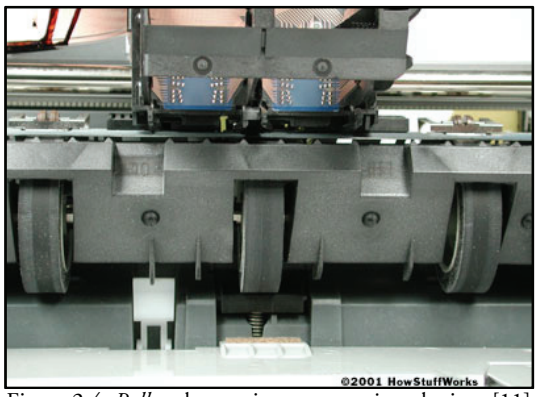


Figura 2.4: *Rollers* de uma impressora a jato de tinta [11]

   c. Motor do alimentador de papel: é um motor de passo que impulsiona os *rollers* com precisão, de forma a garantir que imagens contínuas sejam corretamente impressas.

3. Alimentação do sistema: é fornecida por um transformador cuja tensão e corrente de operação variam conforme o fabricante e o modelo da impressora. As tensões secundárias são reguladas pelo circuito de controle.

4. Interfaces: a porta paralela e a serial ainda são usadas, mas atualmente a interface de comunicação predominante nas impressoras é o barramento serial universal, ou *universal serial bus* – USB.

5. Unidade lógica de controle: é uma placa sofisticada responsável por interligar todos os componentes anteriores, além de decodificar as informações enviadas pelo computador. É composta por microcontroladores, reguladores de tensão, interfaces de entrada/saída, componentes de eletrônica analógica/digital, entre outros (Figura 2.5).

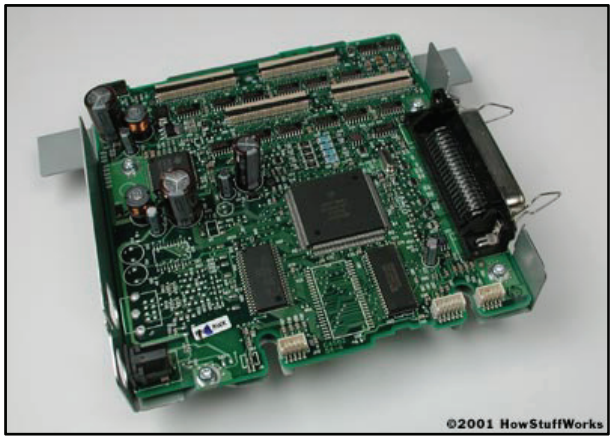


Figura 2.5: Circuito de controle de uma impressora a jato de tinta [11]

3 INTERFACES DE COMUNICAÇÃO

3.1 Porta paralela

A porta paralela, também conhecida por interface paralela, permite a transferência de dados do computador para algum periférico externo, *byte* a *byte*. A interface paralela tradicional é unidirecional, conhecida como SPP (*standard parallel port)*, e permite transferências de dados do computador para o periférico, mas não o caminho inverso. No entanto, existem novas tecnologias – EPP (*enhanced parallel port*) e ECP (*extended capabilities port*) – que tornam possível a operação da porta paralela no modo bidirecional [10].

O Modo SPP, como já citado, é unidirecional e admite taxa de transferência máxima que pode ser de 150 KB/s [10] ou 200 KB/s [12]. Esta velocidade foi projetada apenas para conectar impressoras, já que estas trabalham a baixas taxas de transferência devido à necessidade de monitoramento constante dos registradores. Para transmitir um *byte* é necessário monitorar o sinal *busy*, escrevê-lo nos registradores de dados, enviar um sinal *strobe* e ficar aguardando pelo sinal *acknowledge*, que é enviado pela impressora após o processamento do *byte* recebido [12].

Já o Modo EPP está presente em todos os computadores atuais e traz duas características importantes: comunicação bidirecional e aumento significativo na taxa de transferência, que fica em torno de 800 KB/s [10]. Neste modo, a porta paralela é composta por oito registradores, sendo que três destes já existiam no modo SPP e outros cinco foram criados com o novo protocolo [12].

Diferente dos anteriores, o Modo ECP permite maior velocidade, com taxas de transferência que podem chegar a 2 MB/s, pois implementa um protocolo de sinalização que adapta a velocidade de transferência de acordo com a suportada pelo hospedeiro. O protocolo ECP, além de dados, envia comandos e isto sugere *hardware* e *software* mais complexos [12]. Este modo é um avanço em relação ao EPP, pois aumenta a taxa máxima de transferência por meio de técnicas diversas [10]. Uma dessas técnicas é a compactação de dados via algoritmo RLE[*] (*run length encoding* – codificação de largura corrida); outra é a utilização do DMA (*direct memory access* – acesso a memória direto) que possibilita a comunicação direta da porta paralela com a memória.

No computador, a porta paralela é conhecida como LPT*x* (*line print terminal*), onde *x* varia de acordo com a quantidade de portas existentes – normalmente uma nos computadores de

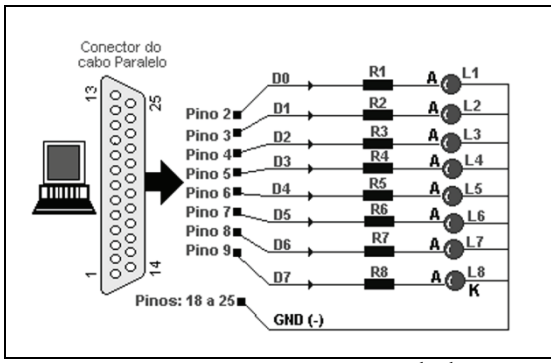
[*] Método que consiste em agrupar seqüências repetidas de caracteres.

mesa e nenhuma nos computadores portáteis. Os endereços de I/O (*input/output* - entrada/saída) e registradores que tornam possível a comunicação do computador com a porta paralela LPT1 estão listados no Quadro 3.1.

Quadro 3.1: Endereços da porta paralela LPT1 [12, adaptado]

ENDEREÇO (hexadecimal)	REGISTRADOR
378h	Dados (8 bits)
379h	Estado (5 bits)
37Ah	Controle (4 bits)

Apesar da porta paralela ter sido projetada originalmente pela IBM* para conectar impressoras nos computadores IBM-PC, esta pode ser utilizada como entrada ou saída genérica. Ou seja, qualquer circuito projetado de acordo com os padrões da IBM pode ser utilizado para se conectar à porta paralela. Essa característica despertou o interesse na utilização desta porta para o presente estudo. Um exemplo de projeto pode ser visto Figura 3.1.




```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#define LPT1 0x378
int main(void) {
  unsigned char Valor=128;  //Em binário: 10000000
  while(Valor > 0 ) {
    outportb(LPT1, Valor);  // Envia para a Porta LPT1
    printf("\Ascender o próximo LED...");
    getch( );
    Valor = Valor >> 1; //Move o bit 1 para a direita
  }
}
```

Figura 3.1: Exemplo de projeto para comunicação com a LPT1 [13, adaptado]

Os registradores da porta paralela dividem-se em registrador de dados, registrador de estados e registrador de controle. A Figura 3.2 ilustra a relação entre os *bits* dos registradores que compõem a porta e a sua interface física, o conector DB-25.

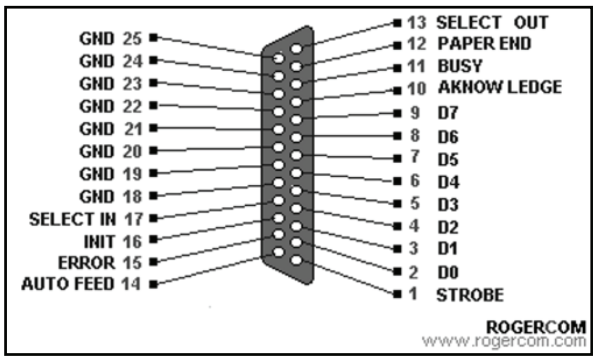


Figura 3.2: DB-25 [13]

O registrador de dados, que se localiza no endereço de I/O 378h (ver Quadro 3.1), é o responsável pela transferência dos dados a serem impressos (em *bytes*). Os pinos 2 a 9 da porta

* *International Business Machines <http://www.ibm.com>.*

paralela correspondem aos *bits* D0 a D7 do registrador de dados (Figura 3.2). Estes pinos são capazes de fornecer 2,6 *m*A e drenar 24 *m*A de corrente elétrica.

Para escrever o nível lógico "alto" em um *bit* do registrador de dados, deve-se aplicar uma tensão entre 3,1 V e 5 V. Por outro lado, para escrever o nível lógico "baixo", aplica-se uma tensão de 0 a 0,4 V [13]. Nos modos avançados (EPP ou ECP) da porta paralela, pode-se fazer com que o registrador de dados atue como entrada ativando o *bit* 5 (I/O 37Ah – Quadro 3.1) do registrador de controle [12].

O registrador de estados se localiza no endereço de I/O 379h (Quadro 3.1) e informa as condições de operação da impressora. Os estados são:

- *acknowledge* (pino 10): *bit* informado pela impressora, com duração típica de 10 μs em nível lógico baixo, quando esta está pronta para receber novos dados após o envio de dados anteriores.
- *busy* (pino 11): indica que a impressora está ocupada. A impressora emite esse sinal após o recebimento de cada dado, quando o *buffer*[*] está cheio, quando estiver fora de operação ou quando há um erro.
- *paper end* (pino 12): indica que a impressora está sem papel.
- *select out* (pino 13): indica que a impressora está pronta para receber informações.
- *error* (pino 15): indica (nível baixo) que houve erros na impressora, que podem ser falta de papel, impressora fora de operação ou um estado de erro genérico.

O registrador de controle se localiza no endereço I/O 37Ah (ver Quadro 3.1) e permite verificar se os dados estão prontos para serem transmitidos (*strobe*, pino 1). Além disso, faz com que a impressora mova o papel para o início da próxima linha (*autofeed*, pino 14), pause trabalhos de impressão (*select in*, pino 17) e reinicialize e zere o *buffer* da impressora (*init*, pino 16) [12].

Outro conceito relevante utilizado na porta paralela é o *handshake*. Este, que pode ser traduzido literalmente como aperto de mão, é a confirmação que a impressora fornece ao computador após a leitura e processamento de cada *byte* (Figura 3.3).

[*] Memória interna da impressora utilizada para armazenar temporariamente os dados, o que libera o microcomputador mais rapidamente para execução de outras tarefas [10].

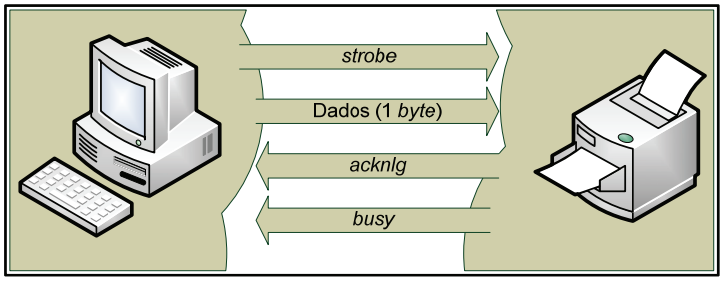


Figura 3.3: *Handshake* entre o computador e a impressora

O sinal *busy*, quando em nível alto, indica que a impressora está ocupada e não pode receber mais dados até que termine de processar o anterior. Ao desocupar, o computador coloca um *byte* no barramento de dados, aguarda a estabilização e envia um sinal em nível alto no pino *strobe* para alertar a impressora sobre o novo dado. A impressora lê o dado e informa, por meio do sinal *acknowledge*, que terminou a leitura. Neste ponto, coloca o sinal *busy* com nível baixo e libera o computador para colocar outro *byte* no barramento de dados. Esse ciclo se repete até o fim da comunicação entre o microcomputador e a impressora [12].

3.2 Porta serial

A porta serial, também chamada de porta de comunicações[*], faz parte dos computadores há mais de vinte anos. É considerada a conexão externa mais básica que existe em um computador. No entanto está sendo abandonada devido à facilidade das conexões USB [14].

As portas de comunicação são baseadas na interface RS-232, que estabelece padrões físicos de comunicação pela porta serial [12]. São baratas, de programação simplificada, admite a utilização de longos cabos[†] e pode ser usada em conjunto com microcontroladores de baixo custo [15], possibilitando sua aplicação para o nosso caso concreto. Os endereços de I/O das interfaces padrão RS-232, em um computador, podem ser vistos no Quadro 3.2.

Quadro 3.2: Endereços convencionais das portas seriais no Windows [15, adaptado].

Porta	Endereço	IRQ
COM1	3F8h	4
COM2	2F8h	3
COM3	3E8h	4 ou 11
COM4	2E8h	3 ou 10

A interface RS-232 pode se comunicar de duas formas: a transmissão síncrona e a transmissão assíncrona. Na transmissão síncrona os dispositivos envolvidos usam um único *clock*, que é gerado por um deles ou por um dispositivo externo. Todos os *bits* transmitidos são sincronizados com este *clock* [15] (Figura 3.4).

[*] Nos computadores, são conhecidas como dispositivos do tipo COM.
[†] O comprimento dos cabos pode ser de até 15,24 m [12].

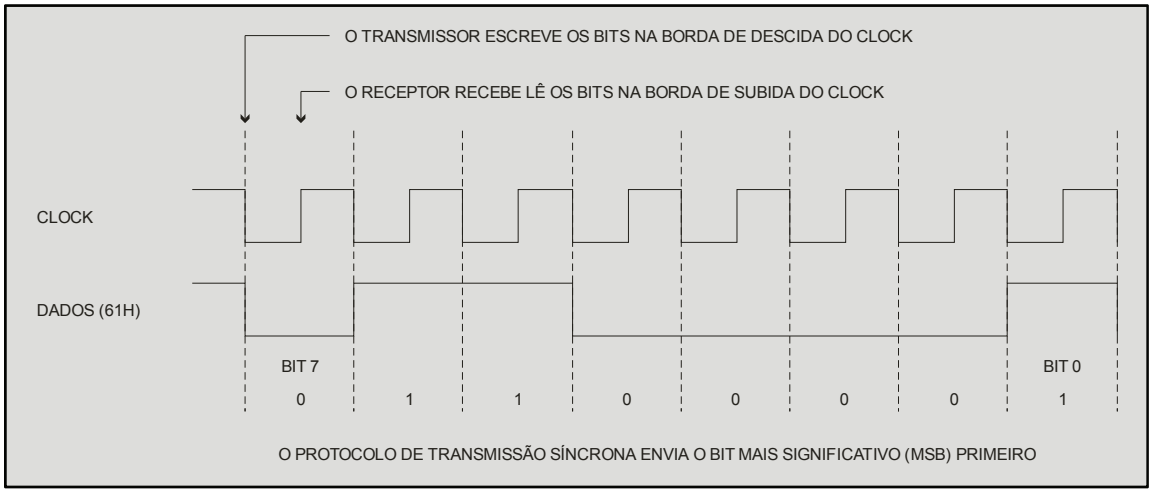


Figura 3.4: Transmissão serial síncrona [15, adaptado]

A transmissão assíncrona significa que "pelo mesmo fio onde os dados são transmitidos, são transmitidos os sinais de sincronismo, isto é, os sinais que indicam início e fim da transmissão" [10]. Neste tipo de transmissão, não é necessário *clock* pré-determinado para a transmissão dos *bits*. Cata *byte* transmitido inclui um *start bit* (*bit* de início) que sincroniza os *clocks* e um ou mais *stop bits* (*bits* de parada) para finalizar a transmissão [15] (Figura 3.5).

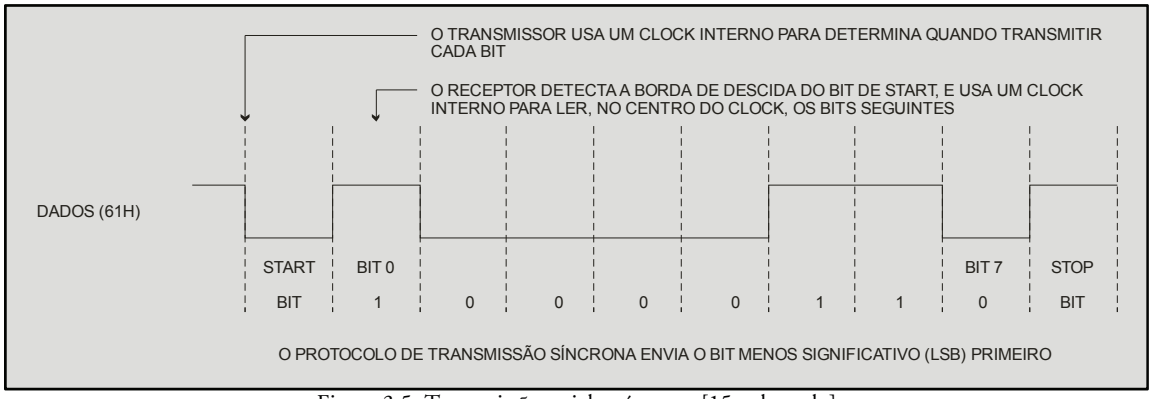


Figura 3.5: Transmissão serial assíncrona [15, adaptado]

A RS-232 usa a transmissão assíncrona e o padrão 8-N-1[*] para enviar os *bits*, por ser este o mais popular. Neste padrão, o transmissor envia 1 *start bit* seguido de 8 *bits* de dados sem verificação de paridade (Figura 3.5). A velocidade de transmissão de um *byte* é de um décimo da taxa de transmissão informada, pois se considera também os *bits* de controle.

De acordo com Zelenovsky e Mendonça [12],

> As portas seriais do PC são totalmente programáveis e somente permitem a comunicação assíncrona. É possível programar os *bits* de partida, os *bits* de paridade e os bits de parada. Um gerador de taxa de comunicação ("baud-rate") programável permite a operação a até 115.200 bauds. Pode-se transmitir caracteres com 5, 6, 7 ou 8 bits, com 1, 1½ ou 2 bits de parada. Um sistema de interrupções priorizáveis controla as interrupções de transmissão, recepção, erro ou estado de linha. Existe internamente um

[*] 8-N-1 = [oito *bits*]-[sem paridade]-[um *start bit* para cada *byte* de dados].

> recurso para diagnóstico, através da função de "loop-back". O coração da interface serial é o circuito 8250, ou seu equivalente funcional [12 p. 494].

A conversão de dados paralelos em seriais no barramento do sistema e vice-versa é uma funcionalidade do UART (*universal asynchronous receiver and transmiter* – receptor e transmissor assíncrono universal) [10] [14]. O UART tem *buffer* para aumentar o desempenho dos dispositivos, e é organizado no formato FIFO (*first in, first out* – primeiro a entrar é o primeiro a sair). Quando este *buffer* enche uma interrupção é gerada no sistema. O seu tamanho típico é de quatro *bytes* [10], mas pode ser também de 16 *bytes* [15].

O UART suporta conexões unidirecionais ou bidirecionais e os padrões da interface RS-232, tais como *handshaking* (similar ao da porta paralela, visto na seção 3.1) e sinais de controle.

No conector DB-9[*] (Figura 3.6) a tensão em um pino é considerada em nível lógico alto quando for mais negativa que –3 V (faixa conhecida como –12 V); e será considerada em nível lógico baixo quando a tensão for mais positiva que +3 V (faixa conhecida como +12 V). As regiões entre +3 V e –3 V, abaixo de –15 V e acima de +15 V, são consideradas níveis lógicos inválidos [12].

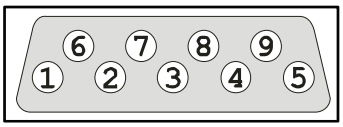


Figura 3.6: Conector DB-9

O Quadro 3.3 lista as funções de cada pino do conector DB-9. Convém lembrar que, na prática, somente os pinos *receive data*, *transmit data* e *signal ground* são suficientes para se estabelecer uma conexão serial [16].

Quadro 3.3: Relação entre pino e função no DB-9 [11, adaptado]

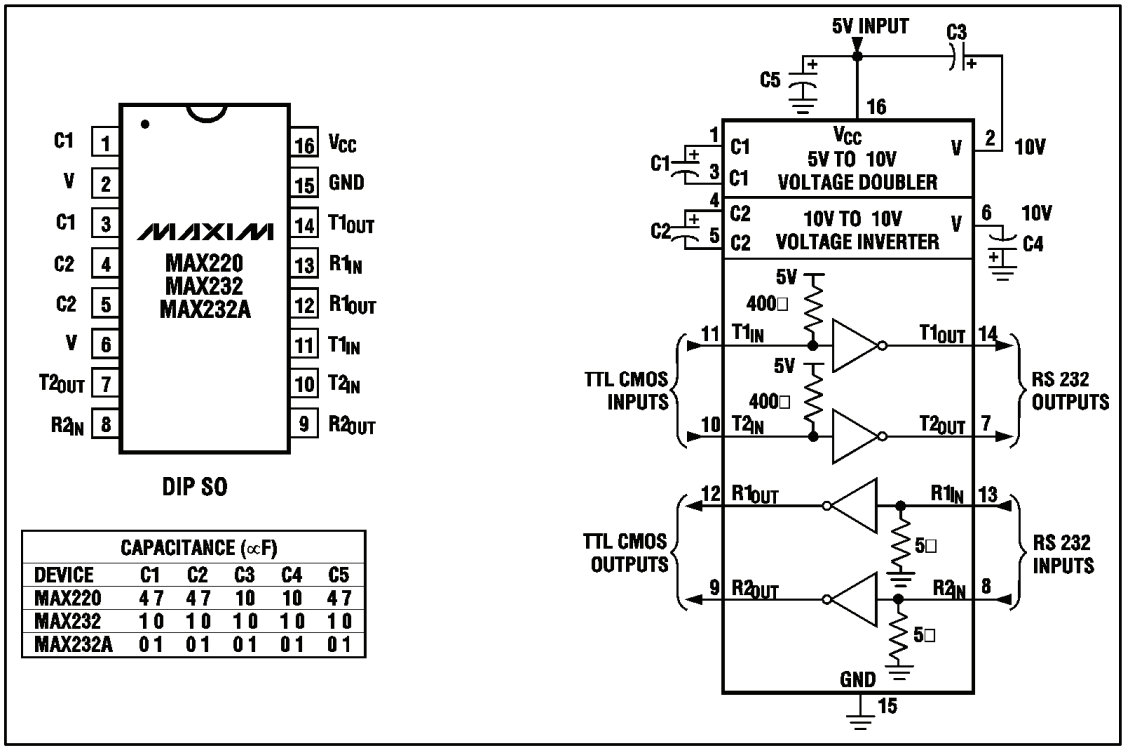
Pino	Nome do pino	Função
1	*Carrier detect* (*detector de portadora)*	Determina se o dispositivo está conectado
2	*Receive data* (recebimento de dados)	O computador recebe dados do dispositivo
3	*Transmit data* (transmissão de dados)	O computador envia dados ao dispositivo
4	*Data terminal ready* (terminal pronto)	O computador avisa ao dispositivo que está pronto para iniciar a comunicação
5	*Signal ground* (sinal de terra)	Pino ligado a terra
6	*Data set ready* (dispositivo pronto)	O dispositivo avisa ao computador que está pronto para iniciar a comunicação
7	*Request to send* (solicitação de envio)	O computador pergunta ao dispositivo se pode enviar informações
8	*Clear to send* (pronto para enviar)	O dispositivo avisa ao computador que as informações podem ser enviadas
9	*Ring indicator* (indicador de sinal)	Assim que a ligação é estabelecida, o computador reconhece o aviso

[*] Existe também o conector DB-25 para a porta serial.

3.2.1 O circuito integrado MAX232

Os microcontroladores, diferentemente da porta serial, trabalham com tensões que variam de 0 V a 5 V – contra uma variação de –12 V a +12 V da porta serial. Por isso necessitam de conversão dos níveis lógicos, de forma que a porta serial e o microcontrolador possam se comunicar.

O circuito integrado MAX232[*] (Figura 3.7) converte a faixa de tensão de –12 V em sinal lógico alto (5 V) e a faixa de tensão de +12 V em sinal lógico baixo (0 V) e vice-versa. Em outras palavras, pode-se dizer que o MAX232 contém dois *drivers* de entrada que convertem tensões no padrão da família lógica TTL (*transistor-transistor logic*) em tensões compatíveis com a interface RS-232. Possui ainda dois receptores que recebem tensões compatíveis com os padrões da interface RS-232 e as convertem em tensões do padrão da família TTL [15].



CAPACITANCE (∝F)					
DEVICE	C1	C2	C3	C4	C5
MAX220	47	47	10	10	47
MAX232	10	10	10	10	10
MAX232A	01	01	01	01	01

Figura 3.7: MAX232 [17]

[*] Para mais detalhes do circuito lógico MAX232, ver ANEXO B.

4 ATUADORES

Chama-se atuador qualquer dispositivo utilizado para transformar pulsos elétricos em ações mecânicas, comandadas por um processador ou microcontrolador. Neste capítulo são descritos os atuadores de interesse para este projeto. São eles: motor de corrente contínua, motor de passo, solenóide cilíndrico e servo-motor.

4.1 Motor de corrente contínua

Os motores de corrente contínua ou motores DC (*direct current*) são usados largamente na indústria, desde a fabricação de brinquedos infantis a veículos automotores. Os motores DC* são de baixo custo, de fácil aplicabilidade e disponíveis em vários tamanhos e formas [18].

A estrutura de um motor DC básico com escovas é composta de seis partes: armadura ou rotor, comutador, escovas, eixo, imã de campo e uma fonte de alimentação de corrente contínua (Figura 4.1). Conforme ilustrado, o motor gira quando o condutor enrolado na armadura é percorrido por uma corrente em meio a um campo magnético [19].

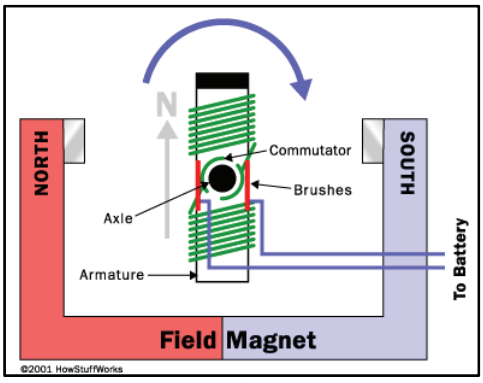


Figura 4.1: Motor DC básico com escovas [20]

Nas impressoras a jato de tinta, o motor DC é responsável pela tração da cabeça de impressão. Sua estrutura e aspecto físico são apresentados nas figuras abaixo.

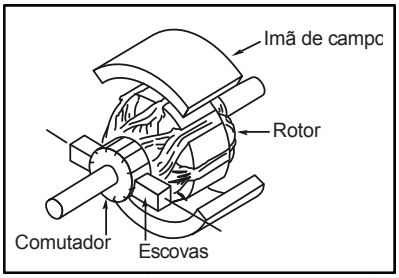


Figura 4.2: Interior de um motor DC [19, adaptado]

Figura 4.3: Motor DC modelo C2164-60045 [21]

* Para mais detalhes do motor DC, ver ANEXO C.

O motor DC foi projetado para converter energia elétrica em energia mecânica, porém seu controle não é trivial (Figura 4.4). Deve-se considerar que ele possui inércia em seu rotor: quando a corrente é interrompida, este permanece girando mais um tempo [19]. O tempo de inércia depende da velocidade de rotação, que é proporcional à tensão média aplicada. Algumas formas de tratar a inércia serão assuntos das seções seguintes.

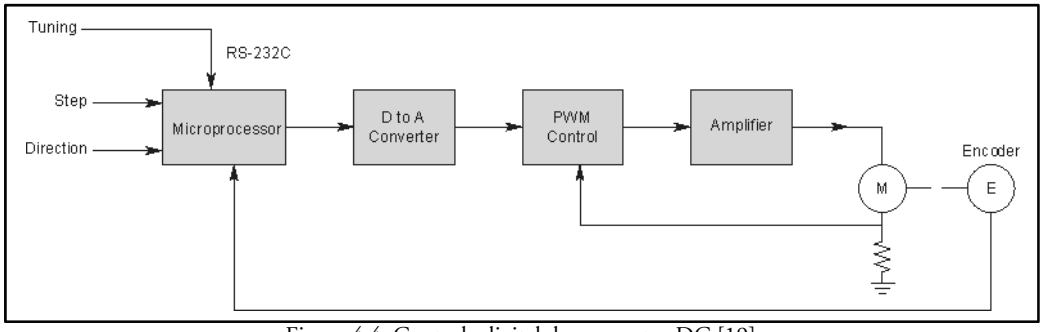


Figura 4.4: Controle digital de um motor DC [19]

4.1.1 PWM

Quando se trabalha com eletrônica analógica interagindo com eletrônica digital é necessário, em alguns casos, utilizar a modulação por largura de pulsos – PWM (*pulse-width modulation*) [19]. Usa-se o PWM em situações onde se deseja converter a informação em um canal de comunicação ou em situações que é necessário controlar a corrente entregue a uma carga. Por exemplo, se tenho uma tensão de 12 V e desejo regular esta em 6 V para alimentar um determinado componente eletrônico, aplico um *duty cycle* de 50%.

PWM é uma onda com formato quadrado de período específico (Figura 4.5) [16]. Para a utilização de PWM nos microcontroladores deve-se calcular (ou testar empiricamente) os períodos e o *duty cycle* (tempo de trabalho). Uma aplicação típica do PWM é o controle de velocidade de motores de corrente contínua.

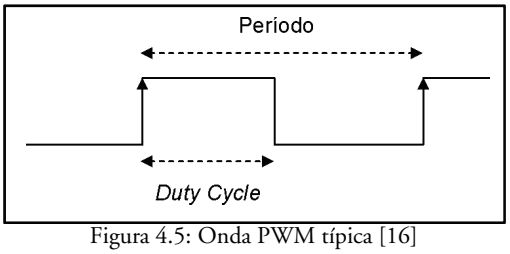


Figura 4.5: Onda PWM típica [16]

Alguns exemplos de aplicação são para o controle de um motor DC, gerar saída analógica para um conversor digital-analógico, variar a luminosidade de uma lâmpada, aplicação em telecomunicações e para gerar efeitos de áudio. A restrição fica para a aplicação de altas freqüências, pois haverá grandes perdas devido aos sucessivos chaveamentos.

4.1.2 O circuito integrado L293D

Um projeto envolvendo motores DC muitas vezes requer que o motor gire nos dois sentidos. Para que isto ocorra, é necessário que haja inversões no fluxo de corrente aplicada, o que normalmente se faz por uma combinação de chaves, conhecidas como ponte-H [22] (Figura 4.6).

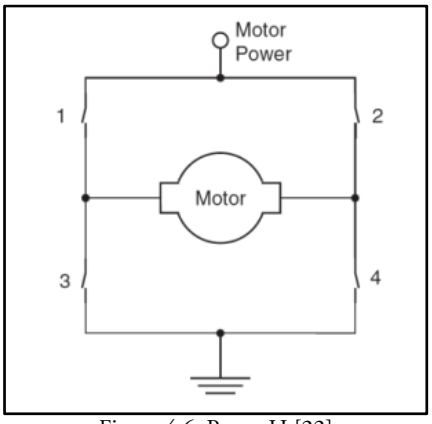


Figura 4.6: Ponte-H [22]

A ponte-H pode ser implementada utilizando apenas chaves e diodos, mas é recomendável utilizar o circuito integrado L293D* – um circuito integrado da *Texas Instruments* que opera com corrente de até 1 *ampère* com tensões de 4,5 a 36 *volts* [22].

O circuito integrado L293D pode ser utilizado para ativar dispositivos, desde que estes não ultrapassem a corrente máxima [23] (Figura 4.7). A Figura 4.8 ilustra um exemplo prático de utilização do L293D como ponte-H.

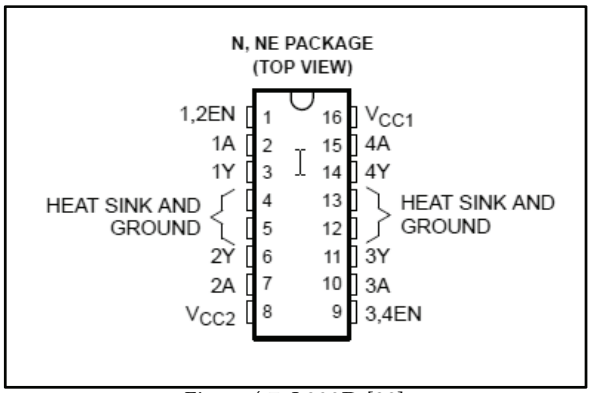


Figura 4.7: L293D [23]

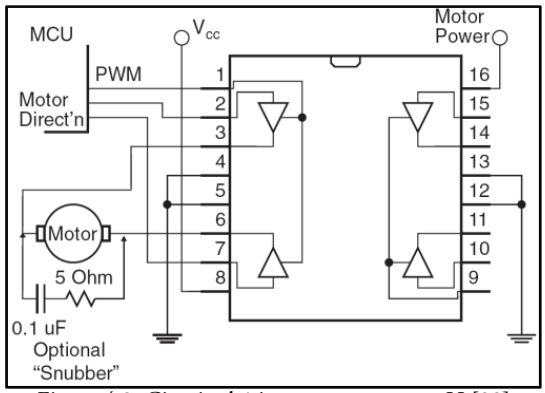


Figura 4.8: Circuito básico com uma ponte-H [22]

4.2 Motor de passo

Motor de passo* é um dispositivo que serve para converter informação digital em saída mecânica, ou seja, rotação com passos definidos. Esta característica fornece interface natural com a eletrônica digital.

* Para mais detalhes do circuito lógico L293D, ver ANEXO D.

A construção de um motor de passo é simples. Consiste em um estator[†] com ranhuras, equipado com duas ou mais bobinas e uma estrutura de rotor[‡] sem rolamento. Se não houver um imã permanente preso ao rotor, o motor de passo é classificado como de relutância[§]. Caso contrário, é classificado como motor de passo com imã permanente [24]. Os motores de passo das impressoras em análise são de relutância (Figura 4.9). Por este motivo os motores de imã-permanente não serão objeto de estudo.

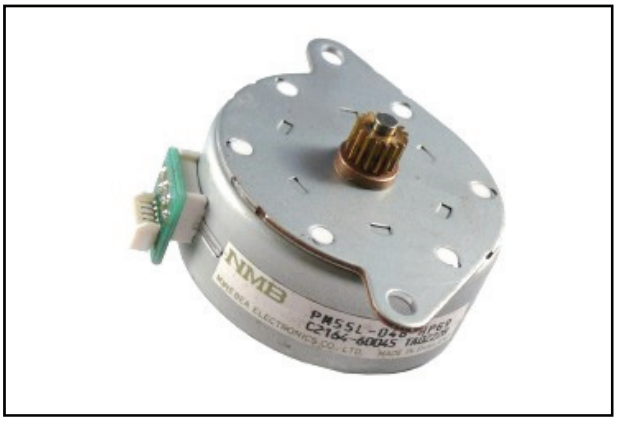


Figura 4.9: Motor de passo C2162-60006 [21]

O motor de passo de relutância desenvolve torque por meio da excitação, em seqüência, das bobinas do estator em razão da grande diferença entre a relutância magnética, o percurso de eixo direto e o percurso do eixo de quadratura. Em uma configuração típica (Figura 4.10) percebe-se a existência de um estator de oito pólos que resulta em quatro bobinas de fase (B1, B2, B3 e B4) e um rotor de seis pólos (R1, R2, R3, R4, R5 e R6). Esta configuração permite a obtenção de um passo polar de 15º [24].

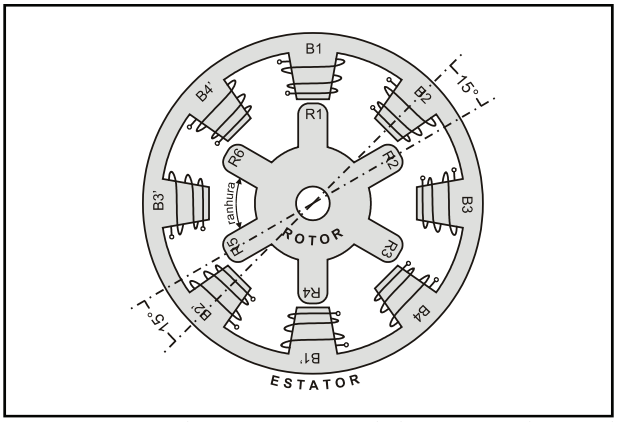


Figura 4.10: Motor de passo com quatro bobinas e rotor de seis pólos

---

[*] Para mais detalhes do motor de passo, ver ANEXO E.
[†] Trave fixa onde as bobinas são enroladas.
[‡] Conjunto eixo-imã que roda na parte móvel do motor.
[§] Análogo em circuitos magnéticos à resistência em circuitos elétricos, exceto por não consumir energia.

O motor de passo C2161-60006 é responsável pelo movimento do carro de impressão das impressoras da série 600 da HP (ver Figura 4.9). Tem como característica dar 48 passos até que uma volta se complete (360º). Assim, é fácil ver que o tamanho do passo polar – ou precisão de giro do motor – é de 7,5º.

4.2.1 Identificação dos fios

Um contratempo para quem deseja projetar circuitos com motores de passo é descobrir a seqüência correta de ligação dos fios, pois não existe um padrão nas cores e/ou esta informação não está presente nos *datasheets*.

Deve-se saber, de antemão, se o motor é de cinco ou seis fios. Nos motores de cinco fios um é de alimentação e quatro de retorno, sendo um para cada bobina (Figura 4.11). No motor de seis fios dois são para alimentação, sendo um para cada dupla de bobinas e os outros quatro para retorno.

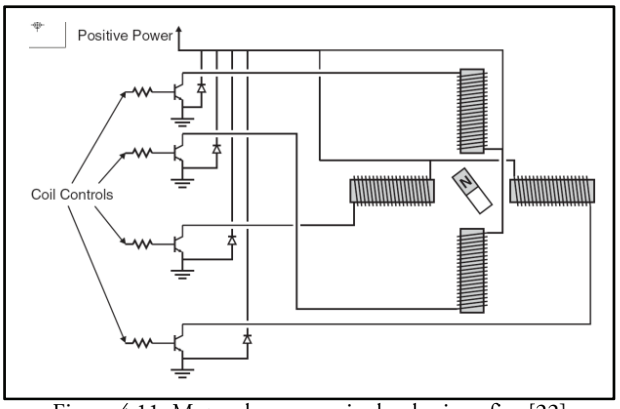


Figura 4.11: Motor de passo unipolar de cinco fios [22]

Identificado o número de fios, deve-se descobrir qual(is) é(são) o(s) de alimentação. Isto pode ser feito medindo-se a resistência entre todos os fios, dois a dois: a resistência entre o fio de alimentação e cada um dos fios de retorno deve ser igual à metade da resistência entre dois fios de retorno [25].

Identificado o(s) fio(s) de alimentação, deve-se descobrir a seqüência correta dos quatro fios de retorno, que será a mesma independente se o motor é de cinco ou seis fios. Para tanto, deve-se proceder à montagem do sistema utilizando o circuito lógico ULN2003A ou equivalente. Em seguida, alimentar o motor no(s) fio(s) comum(ns) e, finalmente, enviar nível lógico alto para os quatro fios restantes em seqüência. A primeira seqüência deve ser escolhida ao acaso, no entanto as demais devem seguir alguma regra lógica de forma que todas as possibilidades sejam testadas. Em algum momento há de se perceber que o motor deu quatro passos seguidos, o que indica que a seqüência correta foi encontrada.

4.2.2 O circuito integrado ULN2003A

O acionamento seqüencial das bobinas faz com que o motor gire um determinado número de passos ou, se for uma seqüência cíclica, gire de forma síncrona. Para tanto, deve-se projetar um circuito que forneçam o controle necessário, com base em sinais lógicos enviados por um microcontrolador. Um circuito integrado projetado para esta finalidade é o ULN2003A*, cuja aplicação em um projeto prático pode ser visto na Figura 4.12.

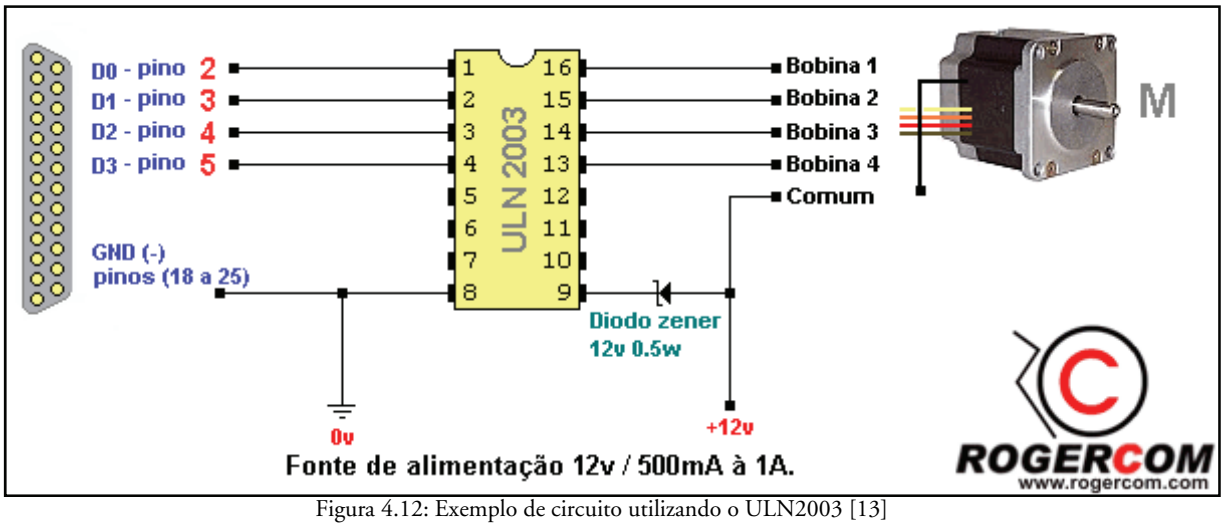


Figura 4.12: Exemplo de circuito utilizando o ULN2003 [13]

O circuito integrado ULN2003A permite tensão máxima de entrada de 30 V a 25 *m*A. Para as saídas, a tensão máxima é de 50 V a 500 *m*A (com picos de até 600 *m*A).

4.3 Solenóide

Os solenóides são dispositivos que convertem energia elétrica em energia mecânica, transformando um sinal elétrico em movimento linear. São formados por uma carcaça metálica (cilindro, tubo), um êmbolo de ferro móvel e uma bobina. Quando a corrente passa pela bobina gera um campo magnético que, por sua vez, movimenta o êmbolo. A força† gerada por esta movimentação é afetada por duas variáveis: número total de voltas e resistência da bobina [26].

Os solenóides cilíndricos são divididos em três classes: os de ação simples, que são ativados enquanto houver corrente na bobina (Figura 4.13); os de ação dupla que têm êmbolos para os dois lados e funcionam de modo similar aos de ação simples; e os de estado duplo, onde cada pico de corrente ativa ou desativa o solenóide [26].

* Para mais detalhes do circuito integrado ULN2003A, ver ANEXO F.
† Medida em Newtons (1 N = 0.1Kgf = 0.225 lbf).

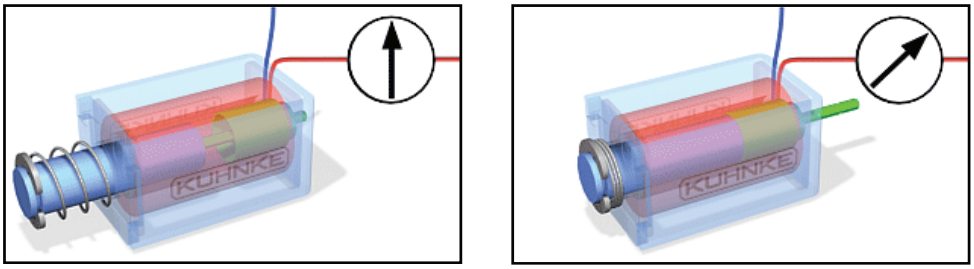


Figura 4.13: Funcionamento de um solenóide cilíndrico de ação simples (*push*) [27]

Ao utilizar o solenóide* em um projeto, os efeitos do calor devem ser considerados. Para tensão constante de aplicação, o aumento de temperatura na bobina reduzirá a força do êmbolo. Em outras palavras, a utilização de tensão constante eleva a temperatura e o calor torna a bobina menos eficiente e diminui a vida útil do solenóide. Convém lembrar que o tempo médio de vida de um solenóide é de 25 milhões de ciclos [28].

Outro conceito importante, quando se fala de solenóides, é o *duty cycle*. Este é determinado pela razão entre o período de trabalho do solenóide e o período total. Por exemplo, se um solenóide trabalha um segundo e fica três em repouso, o *duty cycle* é de 25% (1/[1+3] = ¼).

Também deve-se considerar que existe limite para o período de energização. No exemplo anterior, em que o tempo de trabalho é de apenas um segundo, não há energização suficiente para causar danos. Por outro lado, se o solenóide permanece energizado dez de quarenta minutos, portanto o mesmo *duty cycle*, o tempo de energização é de seiscentos segundos, suficiente para ocasionar a queima do solenóide [28].

4.4 Servo-motor

Servo-motores são utilizados para o controle remoto de modelos de aviões, carros, helicópteros, barcos, etc., devido à precisão no controle de posição (90º a 180º) [29]. Outra característica que chama a atenção é o elevado torque que apresentam. Além disso, por serem facilmente encontrados no mercado, os servo-motores se constituem em alternativa para o solenóide (descrito no na seção 4.3). Neste caso, alguma mecânica extra se faz necessária para utilizar este tipo de motor, uma vez que ele não é linear como o solenóide.

O circuito de controle do servo-motor recebe o sinal de posicionamento e regula o ângulo de acordo com o sinal. Este controle de ângulo dentro do servo-motor é feito por um resistor linear (Figura 4.14) [30].

* Para mais detalhes do solenóide, ver ANEXO G.

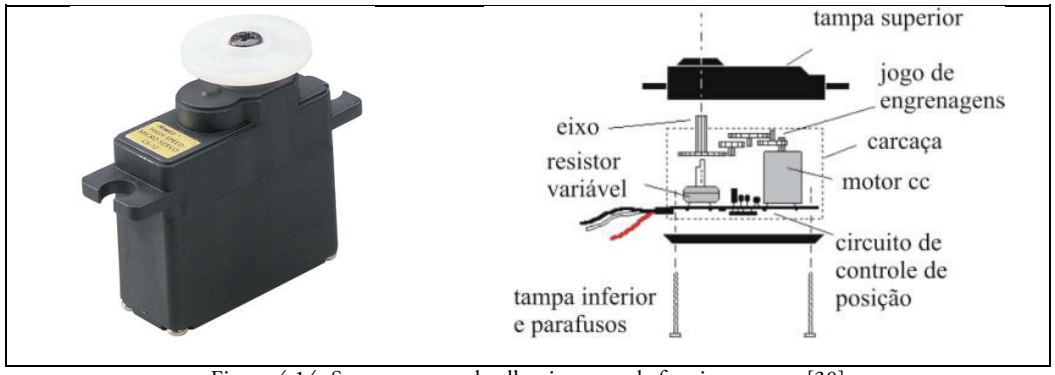


Figura 4.14: Servo-motor e detalhes internos de funcionamento [30]

Três fios ligam o servo-motor: um para alimentação de +5 V, um para controle de posição e o fio terra. A posição é controlada de acordo com a largura do pulso, que pode ser de 1 *ms* (–90º), 1,5 *ms* (0º) ou 2 *ms* (90º) [29] (Figura 4.15).

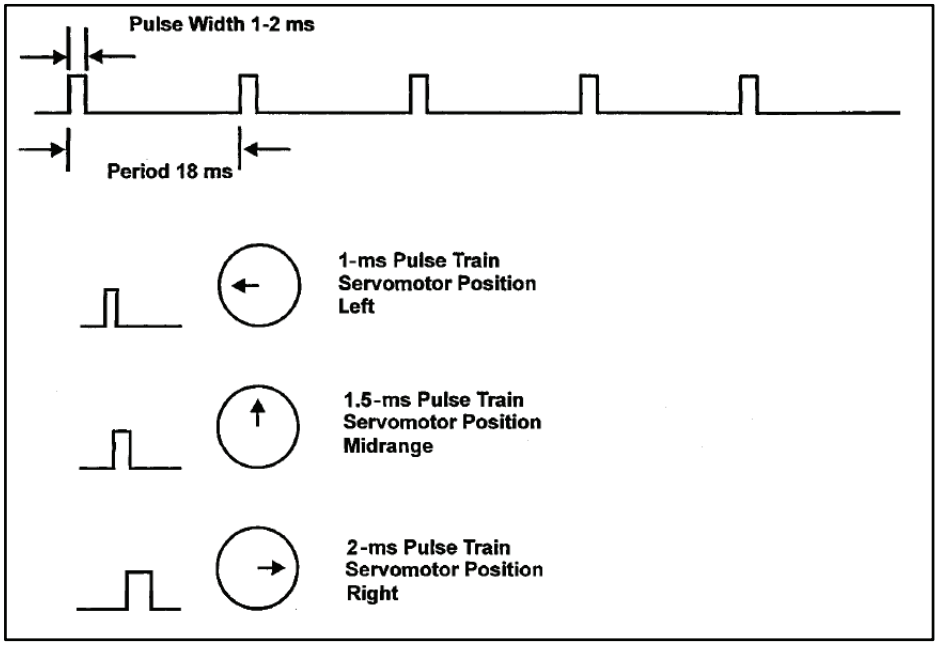


Figura 4.15: Pulsos para o controle de posição de um servo-motor [29]

5 SENSORES

5.1 Sensor de papel

Nas impressoras a jato de tinta da linha HP o motor de passo tem a função de carregar e descarregar o papel. No entanto, para o correto posicionamento do papel, bem como para a detecção da sua falta ou atolamento, o sistema possui uma chave ótica, também conhecida como *photointerrupter* (interruptor de luz).

Esta chave é ativada por uma alavanca que, mecanicamente, detecta a passagem do papel e interrompe a chave ótica [31]. Uma visão do componente eletrônico e do diagrama genérico de implementação é apresentada nas figuras a seguir.

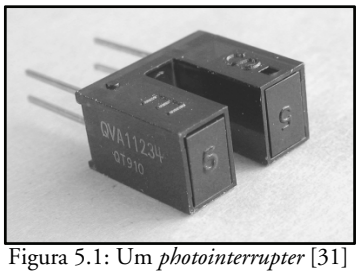


Figura 5.1: Um *photointerrupter* [31]

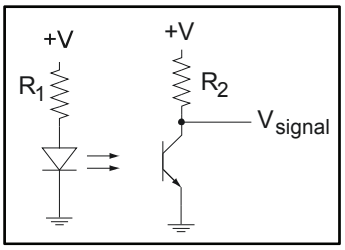


Figura 5.2: Esquema de um *photointerrupter* [31]

5.2 Sensor de deslocamento

Em sistemas mecânicos geralmente é desejado o controle de posição. Para esta finalidade existem diversos tipos de sensores: magnéticos, de contato, resistivos, óticos, etc. Caso o objetivo seja controlar a posição de forma precisa – que é o caso das impressoras a jato de tinta – o sensor mais indicado é o ótico. Este pode ser encontrado sob duas formas: absoluto e incremental [19].

Nesta seção trataremos apenas do codificador ótico incremental, também chamado de *encoder strip*, por se tratar do controle presente nas impressoras, inclusive naquela utilizada para adaptação ao sistema Braille.

Um *encoder strip* (Figura 5.3) é usado quando é necessário ter uma medição direta do movimento linear. Ele é composto por uma escala linear de comprimento variável (de poucos milímetros a mais de um metro) e um leitor ótico, o *photoineterrupter* com função de *encoder*. A limitação para a utilização deste sensor é a alta velocidade, pois o seu controle está limitado à freqüência de 100 KHz [19].

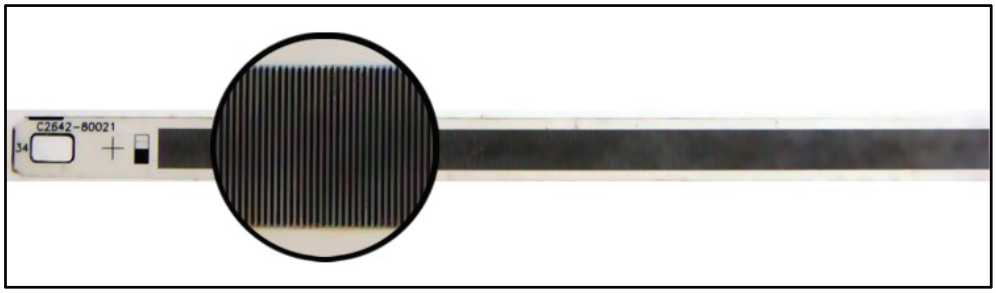


Figura 5.3: *Encoder strip* C2642-80021

Existe também o *encoder* circular incremental (Figura 5.4), onde a fase observada entre o canal A e B é utilizada para determinar a direção do motor (de acordo com a tabela de estados) [32]. Destaca-se que a tabela de estados do *encoder* circular se aplica também ao *encoder strip*.

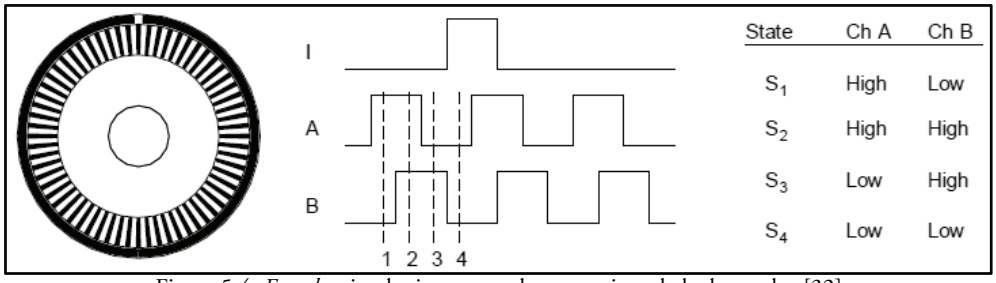


Figura 5.4: *Encoder* circular incremental e respectiva tabela de estados [32]

6 MICROCONTROLADORES

O microcontrolador é um circuito integrado que tem funcionalidades mínimas para controlar o funcionamento de produtos tecnológicos que requerem algum tipo de interação, tais como celulares, DVD *players*, câmeras, etc. Apesar de pequeno, trata-se de um complexo componente que depende da perfeita integração entre *hardware* e *software* com o qual está interfaceado [33].

Os microcontroladores têm todos os componentes necessários para atender à programação de sistemas embarcados: uma CPU (*central processing unit* – unidade de processamento central), memória e portas de entrada e saída. Uma característica fundamental deste componente é que ele deve ser programado de acordo com o circuito no qual está inserido. Esta programação deve ser convertida em linguagem máquina e gravada na memória interna.

No mercado existem diversos modelos e famílias de microcontroladores. Para o desenvolvimento deste trabalho foi escolhido o PIC 16F877A do fabricante *Microchip*[*] por ser este um microcontrolador de porte médio com várias portas de entrada e saída, linguagem de programação relativamente simples, material abrangente e ambientes de programação e depuração gratuitos (Figura 6.1).

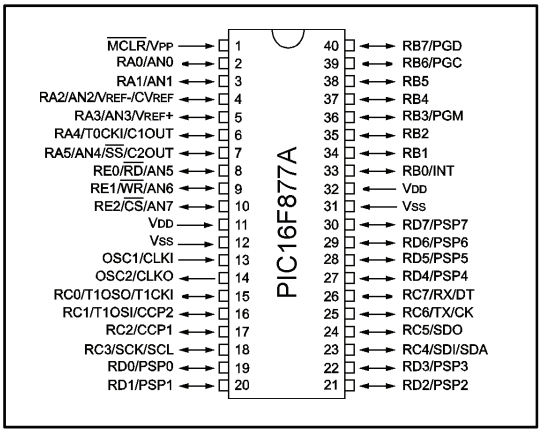


Figura 6.1: O microcontrolador PIC 16F877A [34]

6.1 Arquitetura do PIC16F877

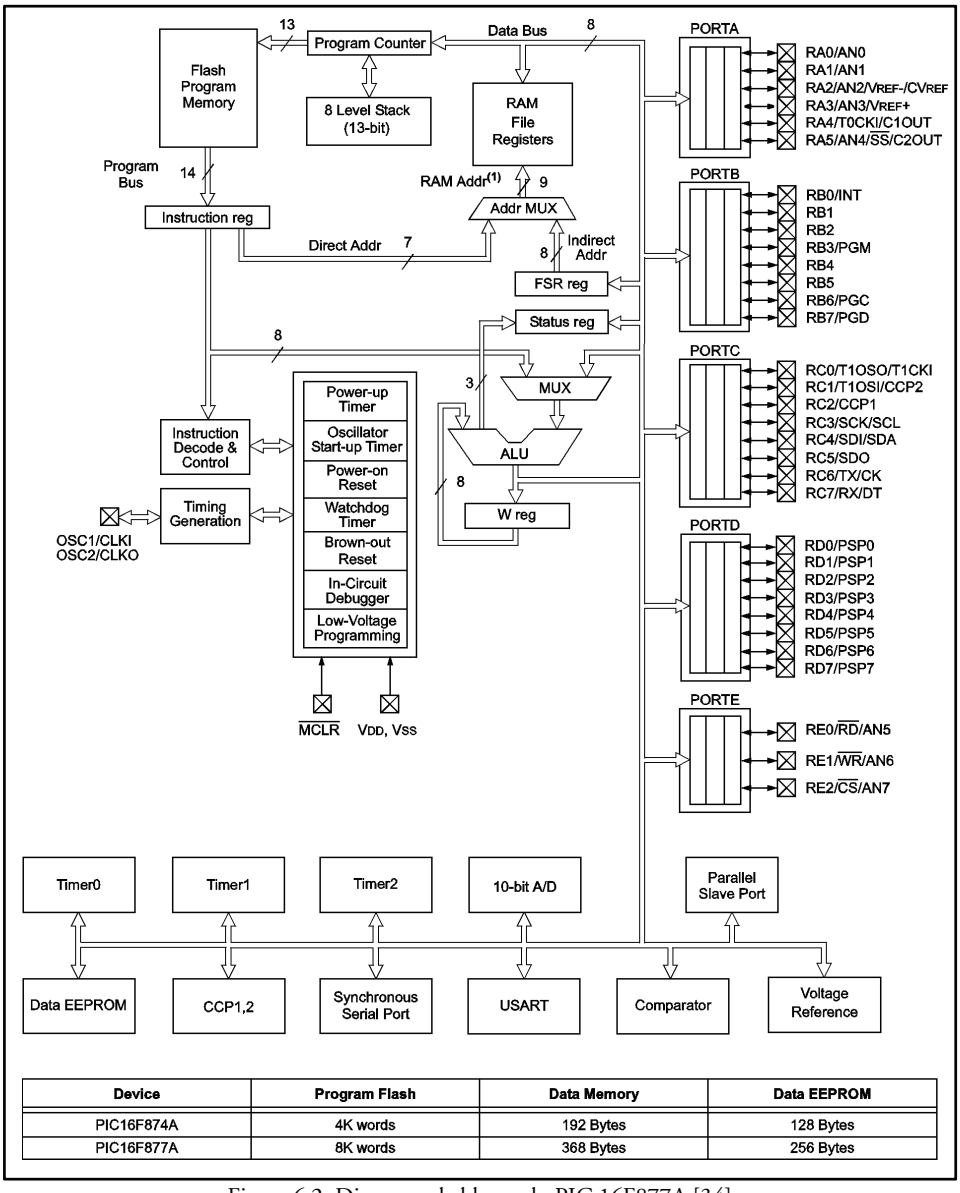
A arquitetura do PIC 18F877A é *Havard*, ou seja, não existe apenas um barramento interno para memória e dados, que é uma característica da arquitetura *Von-Neumann*. Na arquitetura *Havard* existem dois barramentos, um para os dados e outro para as instruções. O barramento de dados tem oito bits e o de instruções pode ter doze, quatorze ou dezesseis bits [34].

[*] http://www.microchip.com

Este tipo de arquitetura, assim como a de *Von-Neumann*, permite a operação no modo *pipeline*, ou seja, enquanto uma instrução é decodificada, outra está sendo buscada na memória.

A tecnologia empregada no PIC 16F877A* é a RISC (*reduced instruction set computer* – computador com conjunto de instruções reduzido). Assim, estes microcontroladores possuem apenas 35 instruções de máquina [34].

As entradas e saídas estão agrupadas em cinco portas: A(5), B(8), C(8), D(8) e E(3), totalizando 32 entradas e saídas. A Figura 6.2 mostra o diagrama de blocos do PIC 16F877A, onde pode ser visto como as entradas e saídas se comportam internamente. As interrupções existentes neste microcontrolador serão vistas seção seguinte.



Device	Program Flash	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

Figura 6.2: Diagrama de blocos do PIC 16F877A [34]

* Para mais detalhes do PIC 16F877A, ver ANEXO H.

6.2 Interrupções do PIC 16F877A

As interrupções são responsáveis por desviar a execução de um programa para uma parte específica do programa. A interrupção, quando couber, pode ser disparada por borda de subida, borda de descida ou em ambos os casos.

Caso não seja possível usar interrupções, deve-se gastar ciclos de processamento do microcontrolador para ficar "consultando", ou fazendo *pooling* em determinada porta e executar o desvio quando a alteração ocorrer. Este método é eficiente quando se trata de programas simples. No entanto, em programas complexos, a utilização da interrupção é mais indicada, pois o evento é tratado pelo *hardware*.

Nos microcontroladores da família PIC, uma interrupção só ocorre quando o sinal da interrupção global e o sinal da interrupção que se deseja monitorar estão ativos. O PIC 16F877A tem quatorze interrupções, agrupadas nas categorias [34]:

- Interrupções de *timer*: TMR0, TMR1 e TMR2 que servem para contar tempo e variações de um sinal externo;
- Interrupção externa: específica da porta RB0/INT, é usada para gerar interrupção por meio de sinais enviados por dispositivos externos. Esta interrupção pode ser configurada para detecção da borda de subida ou descida do sinal aplicado;
- Interrupção por mudança de estado: específica para as portas RB4 a RB7. A interrupção ocorre tanto na borda de subida quanto na borda de descida. Assim, é necessário desenvolver algoritmos que detectem qual porta gerou a interrupção e se esta foi de subida ou descida;
- Interrupção da porta paralela: trata-se de uma interrupção ligada diretamente à porta paralela no modo *slave* e acontece sempre que uma operação de leitura ou escrita é disparada;
- Interrupção dos conversores analógico/digital: esta interrupção ocorre sempre que uma conversão é finalizada;
- Interrupção USART: esta interrupção, que é dividida em recebimento e transmissão, ocorre para indicar o término de uma recepção ou o término de uma transmissão pela porta serial do microcontrolador;
- Interrupção MSSP (*master synchronous serial port)*: esta interrupção é disparada sempre que um dado é enviado ou recebido na porta serial. Possui dois modos de

comunicação, o SPI e o I^2C, que são protocolos de comunicações muito utilizados para interfaceamento com chips com funções específicas, tais como EEPROM's, RTC (*real time counter* – relógio de tempo real);

- Interrupção do CCP (*capture, compare, PWM*): esta interrupção é a mais importante para quem trabalha com motores DC. Com ela pode-se capturar o nível de tensão, comparar e determinar a velocidade do motor por PWM. As portas no PIC habilitadas para esta função são as RC1 e RC2.

7 MATERIAIS, MÉTODOS E PROCEDIMENTOS

7.1 Materiais utilizados

Para o desenvolvimento do projeto foram utilizados:

- Impressoras: diversas[*] impressoras a jato de tinta da marca *Hewlett-Packard*;
- Circuitos integrados: um microcontrolador da família PIC (16F877A), um circuito integrado ULN2003A, um circuito integrado L293D e um circuito integrado MAX232N;
- Componentes eletrônicos: capacitores, resistores, reguladores de tensão, fios, placa de ilhas, *leds*, diodos, soquetes, *buzzer*, servo-motor, solenóide, *jumpers*, chaves e conectores diversos;
- Ferramentas de trabalho: *protoboard*, ferro de solda (30 W), suporte do ferro de solda, pasta para solda, estanho (1 mm), sugador de solda, microretífica *Dremel*[†], ferramentas básicas de bancada, óculos de proteção, pulseira anti-estática e um *MicroICD*[‡];
- Ferramentas de medição: paquímetro digital *Stainless* (150 mm), régua de precisão *Trident flex-inox* (40 cm) e multímetro digital (tensão, corrente, resistência e teste de continuidade);
- Materiais diversos: *etil vinil acetato* (EVA) e reglete *Howe Press Perkins* (modelo 140);
- *Softwares*: *MPLAB IDE*[§], *CSS*[**], *PICKit*[††] e o *PIC Simulator IDE*[‡‡].

Optou-se por desenvolver os códigos fonte em linguagem *C* utilizando o compilador CSS, pois esta apresentou a melhor relação desempenho/estabilidade entre os compiladores para C, além da forte similaridade com o *C ANSI*. Em substituição ao CSS, outros compiladores *C* para microcontroladores podem ser usados.

Dentre as alternativas cita-se os conceituados *HI-TECH* (http://www.htsoft.com) e *MikroC* (http://www.mikroe.com), com versões básicas gratuitas.

---

[*] HP P-2200, cedida gentilmente pelo colega Guilherme Banuth; HP 420C cedida gentilmente pelo colega Giordanno Martins; HP 690 cedida gentilmente pelo colega Paulo Gontijo; duas HP 692 uma cedida pelo orientador e outra pelo graduando.
[†] Ferramenta para corte, furação, lixamento e polimento.
[‡] Ferramenta para gravação de microcontroladores da família PIC (http://www.microgenios.com.br).
[§] Ambiente de desenvolvimento para os microcontroladores da família PIC (http://www.microchip.com).
[**] Compilador C para os microcontroladores da família PIC (http://www.ccsinfo.com).
[††] *Software* para gravação do código compilado para os microcontroladores da família PIC (http://www.microgenios.com.br).
[‡‡] *Software* para simulação de códigos gerados para microcontroladores da família PIC (http://www.oshonsoft.com).

7.2 Métodos e procedimentos

7.2.1 Seleção da impressora

Para a seleção da impressora a ser utilizada dentre as disponíveis, foi realizado um estudo detalhado em cada uma delas. Este estudo envolveu a desmontagem das impressoras e a caracterização de suas peças quanto às dimensões, ao funcionamento dos componentes eletrônicos e à adequação aos objetivos do trabalho. Conforme será apresentado no Capítulo 8, impressora selecionada foi a HP 420[*] (figuras abaixo).

Figura 7.1: Impressora HP Deskjet 420

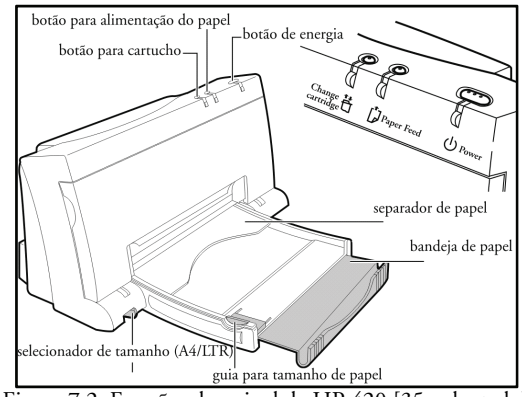


Figura 7.2: Funções do painel da HP 420 [35, adaptado]

Antes de planejar o circuito, vários testes foram realizados para aferir o perfeito funcionamento de cada um dos componentes eletrônicos da HP 420C. Estes testes foram realizados basicamente com a utilização de um *protoboard* – uma matriz de contatos onde foi possível montar circuitos sem a necessidade de soldagem – e com um multímetro digital. Importa destacar que os mesmos testes foram realizados após a soldagem dos componentes na nova controladora lógica, como forma de verificação da perfeita montagem do circuito final.

7.2.2 *Hardware*

O diagrama de blocos da impressora a jato de tinta (ver Figura 2.1) serviu como inspiração para o desenvolvimento do diagrama de blocos para a impressora adaptada (Figura 7.3).

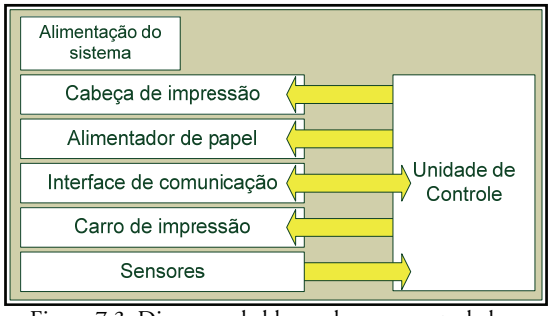


Figura 7.3: Diagrama de blocos da nova controladora

[*] O diagrama de montagem da HP 420 e da HP 692 pode ser consultado no ANEXO I.

No diagrama de blocos da impressora adaptada tem-se: alimentação do sistema, cabeça de impressão, alimentador de papel, carro de impressão, sensores, interface de comunicação e unidade de controle. As seções seguintes caracterizam cada bloco individualmente, com o seu respectivo esquemático de circuito.

Alimentação do sistema

O esquemático do circuito de alimentação do sistema pode ser visto na Figura 7.4. Um *led* indicador foi incluído no circuito com a única função de verificar a passagem de corrente. Esse procedimento foi bastante útil por permitir a verificação imediata de curto-circuitos e do correto funcionamento dos reguladores de tensão.

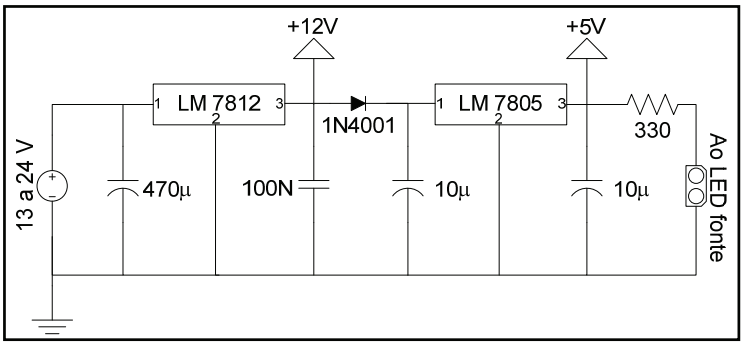


Figura 7.4: Esquemático do circuito do alimentador do sistema

A tensão de entrada, depois de passar por um regulador de tensão, o circuito integrado LM 7812, é reduzida para 12 V, que será usada para alimentar os motores, o solenóide e um segundo regulador de tensão, o circuito integrado LM 7805. Este, por sua vez, regula novamente a tensão para 5 V que é utilizada para alimentar os demais componentes do sistema.

A alimentação do sistema recebe tensão contínua de 24 V a 400 *m*A, proveniente da fonte original da impressora HP 420C. No entanto, esta tensão está no limite da energia máxima de entrada, o que provoca aquecimento excessivo no LM 7812 com grande desperdício de energia. Para evitar a queima do regulador, ou mesmo anormalidades no sistema por excesso de calor neste componente, um dissipador de calor alumínio foi adaptado nos dois reguladores.

Cabeça de impressão

Com o intuito de definir métricas de comparação entre projetar uma cabeça de impressão utilizando o solenóide ou o servo-motor, as duas opções foram testadas. Para tanto, dois cartuchos vazios foram preparados para adaptação, um para o solenóide e outro para o servo-motor. Todas as operações indicadas a seguir (corte, furação, polimento, etc) foram feitas utilizando a ferramenta *Dremel*.

Nos dois cartuchos foram retiradas as tampas por meio de corte, pois a cola utilizada não era removível. Em seguida, procedeu-se à limpeza do cartucho com água corrente, de forma que todos os resíduos de tinta fossem completamente eliminados.

Para determinar a força média necessária para a marcação do relevo procedeu-se com a escrita Braille de forma manual, usando reglete e punção, sobre uma balança de precisão e a força média observada para a marcação do ponto foi de 0,9 Kgf ou 9 N. Ocorre, porém, que este procedimento foi realizado após a aquisição do solenóide[*] escolhido para aplicação no projeto, e este apresentou força insuficiente. Em razão disto, e levando em consideração recomendações[†] do fabricante, a tensão foi dobrada na tentativa, bem sucedida, de aumentar a força. O servo-motor[‡], por sua vez, foi adquirido após a medição da força e este foi um dos critérios para a escolha.

Para a inserção do solenóide (Figura 7.5), procedeu-se à furação do cartucho de tinta, conforme medidas indicadas. O solenóide não tem sistema de retorno automático, daí a necessidade de adaptação de um fio de aço, utilizado como mola, preso na lateral do cartucho para agregar tal funcionalidade, essencial para que este possa funcionar como cabeça de impressão.

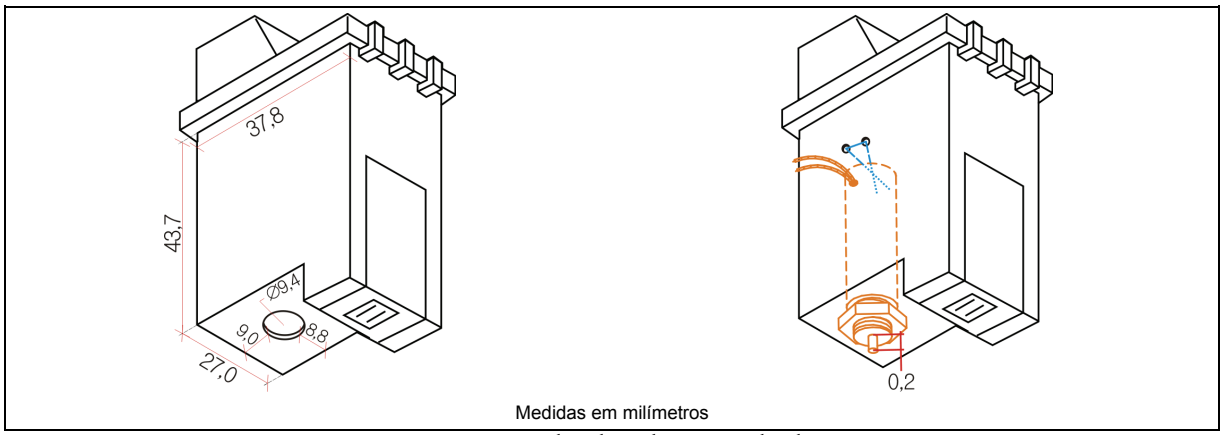


Figura 7.5: Inserção do solenóide no cartucho de tinta

Para a inserção do servo-motor, procedeu-se à furação da parte frontal do cartucho, com as medidas indicadas no Figura 7.6a. Observa-se na figura que os furos superiores não têm medidas indicadas, mas estes devem ser feitos na direção dos parafusos do servo-motor. Diferentemente do solenóide, o servo-motor não tem uma cabeça na forma da punção. Dessa forma, para que o servo-motor pudesse ser utilizado, foi feita uma mecânica com partes de uma caneta e *clip* de prender papel. Após desmontar a caneta, os resíduos de tinta foram eliminados com água corrente. Na montagem aproveitou-se o tubo da caneta como "guia" do sistema. A

[*] Solenóide modelo 120-420-620-540 / 6V.
[†] No *datasheet* o fabricante informa que é possível utilizar o dobro da tensão, 12 V, desde que o *duty cycle* seja de 25%. A curva de força correspondente permite, no máximo, 400gF.
[‡] Hobbico CS-12 #HCAM0110 (torque = 2,54 kg/cm).

cabeça desta, inserida por meio de um *clip*, foi ligada ao disco do servo-motor simulando, desta forma, uma punção (Figura 7.6b).

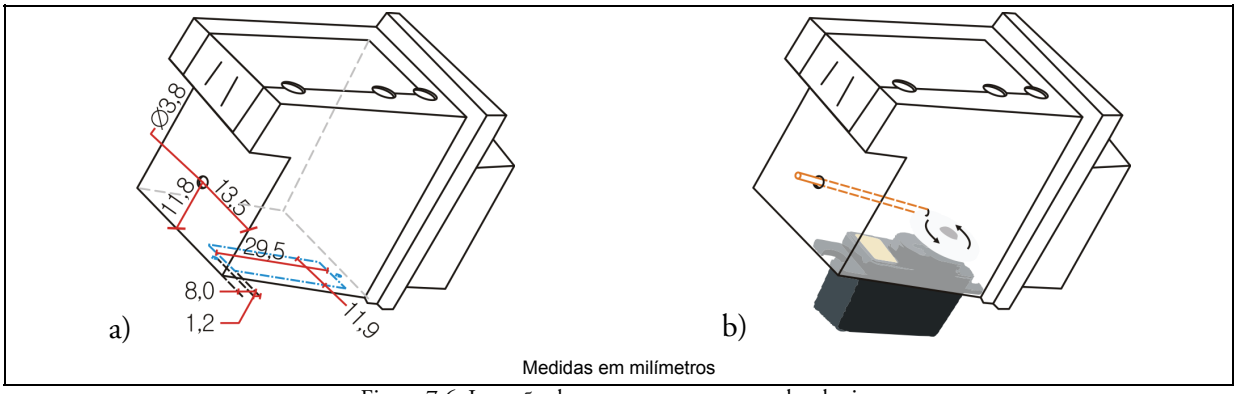


Figura 7.6: Inserção do servo-motor no cartucho de tinta

A Figura 7.7 ilustra o esquemático do circuito para controle da cabeça de impressão. Deve-se lembrar que o ULN2003A controla tanto o solenóide quanto o motor de passo, mas foram separados para fins didáticos.

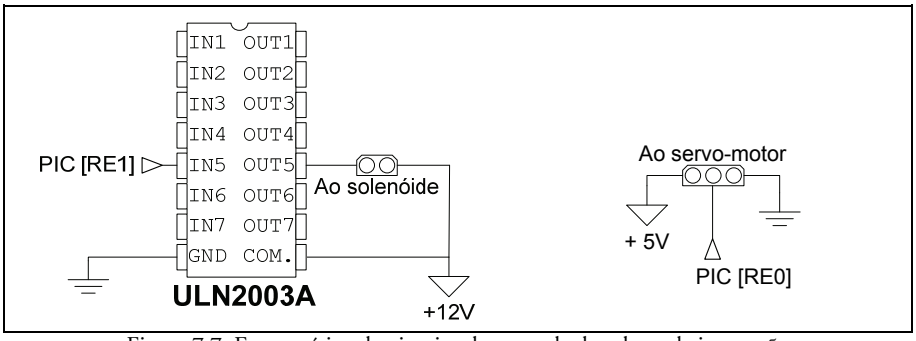


Figura 7.7: Esquemático do circuito de controle da cabeça de impressão

Para a simulação dos sulcos existentes na reglete, uma tira de EVA de 20 cm de comprimento, 2 cm de largura e 0,25 cm de espessura foi colada ao batente. Para que esta não impedisse a passagem do papel, suas bordas foram chanfradas. A tira de EVA, que consiste em uma borracha sintética, se moldará à medida que as páginas forem impressas em Braille (Figura 7.8).

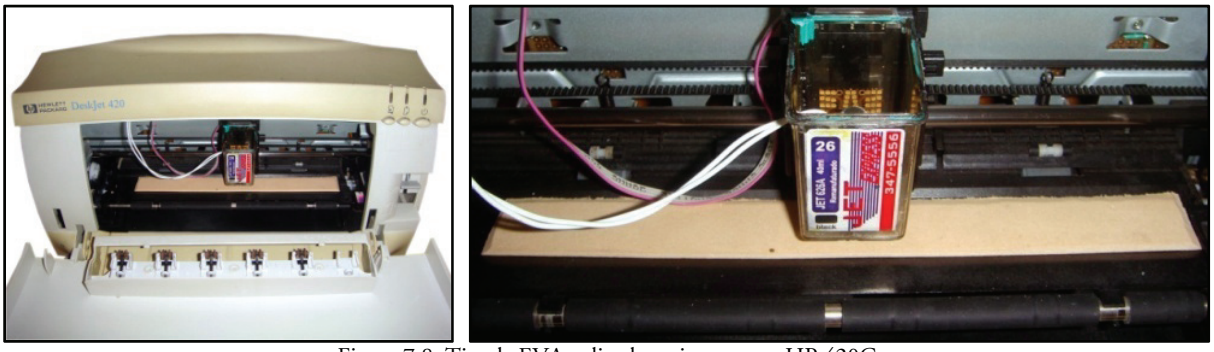


Figura 7.8: Tira de EVA aplicada na impressora HP 420C

Alimentador de papel

O alimentador de papel consiste de um motor de passo e da mecânica de alimentação. Por esta mecânica não estar documentada, o entendimento do alimentador se deu pela observação do funcionamento de impressoras semelhantes em bom estado. Notou-se que o motor gira para trás, por um determinado número de passos, até que o alimentador esteja completamente levantado. Em seguida, gira em sentido contrário, carregando o papel. O número de passos necessários para levantar completamente o alimentador foi descoberto de modo empírico.

A Figura 7.9 ilustra o esquemático do circuito para controle do motor de passo. Conforme já mencionado, o ULN2003A ilustrado é o mesmo que controla o solenóide, e só foi separado para fins didáticos.

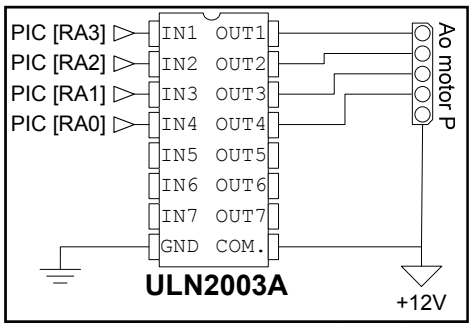


Figura 7.9: Esquemático do circuito para controle do alimentador de papel

Carro de impressão

O carro de impressão, componente que demandou maior tempo de pesquisa, consiste em um motor DC, uma fita codificadora (*encoder strip*) e um leitor de fita decodificadora (*photointerrupter* com função de *encoder*).

O controle do motor foi feito utilizando o circuito integrado L293D, que é um *drive* que atua como ponte-H, possibilitando que o motor gire nos dois sentidos, de acordo com as tensões aplicadas nos pinos 3A/4A. É importante ressaltar que este circuito integrado minimiza o efeito de inércia ao "frear" o motor, de acordo com o Quadro 7.1.

Quadro 7.1: Combinações possíveis nos pinos 3A e 4A do L293D

Nível do pino 3A	Nível do pino 4A	Estado do motor
Alto	Baixo	Gira para a direita
Baixo	Alto	Gira para a esquerda
Alto	Alto	Freia instantaneamente
Baixo	Baixo	Freia instantaneamente

O controle de motor DC projetado foi simplificado e, por isto, não levou em consideração a velocidade de rotação para realimentar do sistema. No entanto, para a impressão de pontos perfeitamente alinhados, um projeto mais complexo de controle é necessário.

O esquemático do circuito de controle para o carro de impressão pode ser visto na Figura 7.10. Observa-se que os pinos 3A/4A foram ligados às portas RC1 e RC2, pois são portas que permitem saídas PWM no PIC 16F877A.

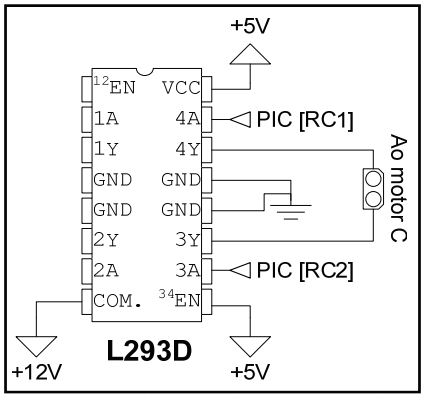


Figura 7.10: Circuito de controle para o carro de impressão

Sensores

Os sensores são a realimentação (*feedback*) do sistema. Para o projeto têm-se basicamente dois sensores: o que identifica se existe papel carregado e o que determina a distância de deslocamento da cabeça de impressão. Conforme citado na seção "7.2.1 Seleção da impressora", o sensor de papel (*photointerrupter*) e o sensor de posição (*photointerrupter* com função de *encoder*) apresentaram mau funcionamento.

O sensor de papel foi facilmente adquirido em lojas de componentes eletrônicos (modelo genérico), adaptado e substituído. O esquemático do circuito, ilustrado na Figura 7.11, é de fácil compreensão: a porta RD3 do PIC 16F877A receberá nível lógico baixo enquanto a base do transistor for alimentada pela luz emitida pelo *led*. Quando a luz é interrompida (sistema mecânico de alavanca ligada ao papel), o sinal na porta RD3 passa para o nível lógico alto, indicando a presença de papel.

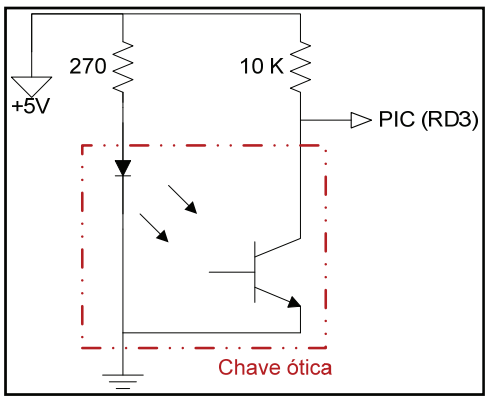


Figura 7.11: Esquemático do circuito do sensor de papel

O sensor de posição*, responsável pela leitura do *encoder strip*, por se tratar de componente proprietário da HP, não tem seu *datasheet* disponibilizado para consulta. Dessa forma, componentes semelhantes de duas outras impressoras foram retirados e desmontados para conhecimento e análise de suas funções (Figura 7.12). Nesta etapa, bastante dispendiosa, mas essencial para a realização deste trabalho, foram identificados que os pinos 1 e 3 são de alimentação e estão ligados com capacitor de desacoplamento. Os pinos 2 e 4 são as saídas A e B, conforme estudado e descrito na seção 0, Figura 5.4.

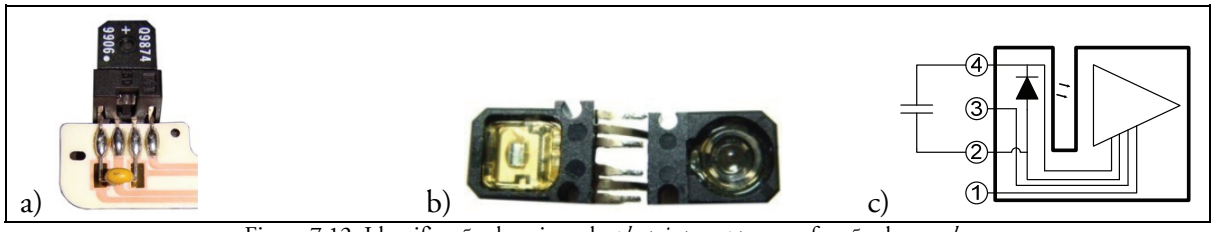


Figura 7.12: Identificação dos pinos do *photointerrupter* com função de *encoder*

Uma vez identificado o comportamento do sensor de posição, bastou integrá-lo ao projeto, por meio de um cabo *flat* retirado de um computador em desuso. A utilização deste cabo originou da necessidade de se ter uma conexão maleável e fina o suficiente para não atrapalhar o deslocamento do carro de impressão. O esquemático do circuito do sensor de posição da cabeça de impressão pode ser visto na Figura 7.13.

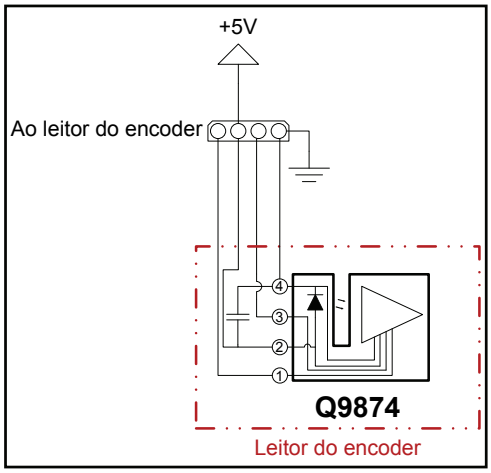


Figura 7.13: Esquemático do circuito do sensor de posição da cabeça de impressão

Interface

Entende-se por interface qualquer das formas com que o um projeto interage com o ambiente externo. No projeto têm-se três interfaces: a de comunicação da impressora Braille com o computador (porta serial), a que permite interação do usuário com a impressora (recarregar papel, ligar, desligar, etc.) e a que permite a programação do microcontrolador (*MicroICD*).

* *Photointerrupter* com função de *encoder* (#Q987A).

O esquemático do circuito de controle para a interface de comunicação com a impressora é apresentado na Figura 7.14. Observa-se que os pinos *R1o* e *T1i* foram ligados ao RC7 e RC6 do PIC, específicos para a comunicação serial.

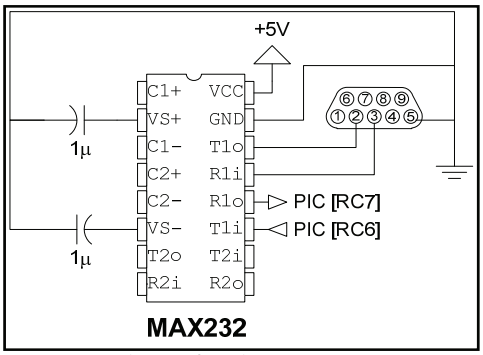


Figura 7.14: Circuito de interface de comunicação com o computador

A interface com o usuário, conforme esquemático do circuito ilustrado na Figura 7.15a, foi desenvolvida com o intuito de aproveitar o painel de controle original da impressora escolhida para adaptação. Ressalta-se, entretanto, que dos três botões existentes apenas dois foram utilizados no projeto.

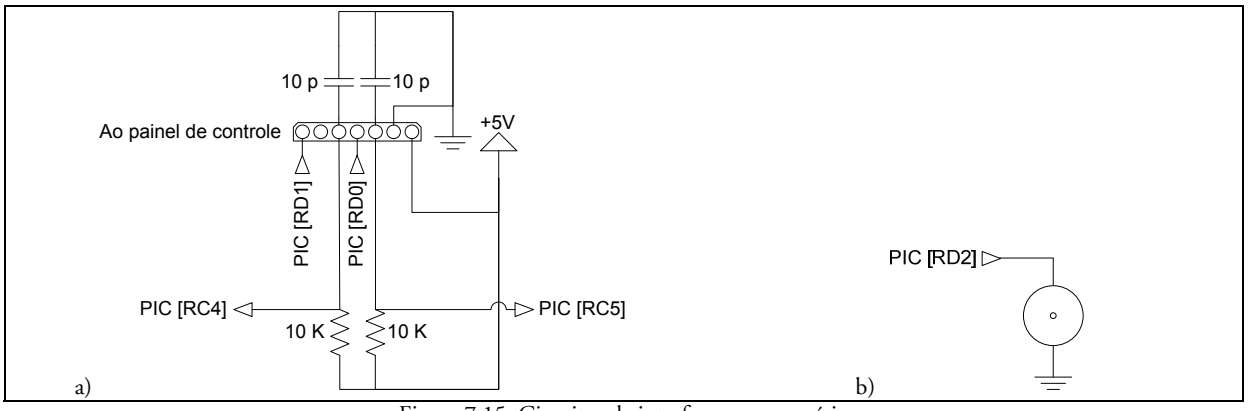


Figura 7.15: Circuitos de interface como usuário

Como a proposta do projeto visa beneficiar os deficientes visuais, é interessante que estes percebam o estado da impressora pela emissão de sinais sonoros. Para tanto foi incluído na proposta do projeto um *buzzer* piezoelétrico (Figura 7.15b). A interface de programação foi projetada de acordo com instruções do fabricante do *MicroICD*. O circuito da interface de programação está ilustrado na Figura 7.16.

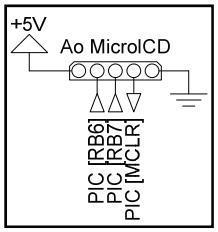


Figura 7.16: Circuito da interface de programação

Unidade de controle

Conforme dito em capítulos anteriores, a controladora lógica é responsável pela integração de todas as partes de uma impressora (Figura 7.17). O "cérebro" da controladora é a unidade de controle que, no projeto, trata-se de um microcontrolador da família PIC.

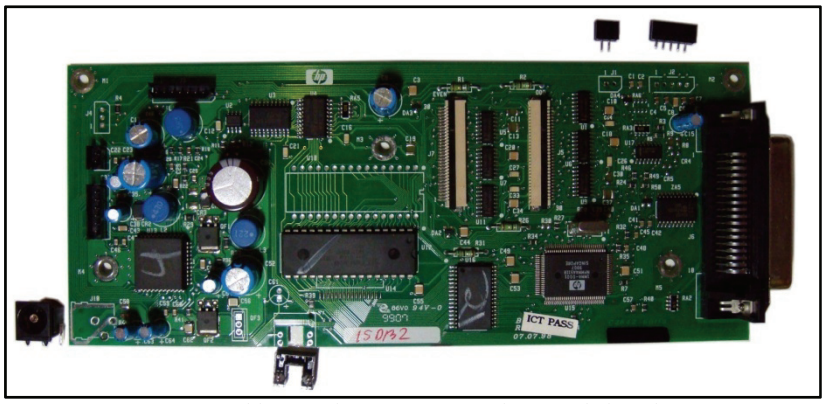


Figura 7.17: "Dessoldagem" dos componentes da controladora da HP 420C

Assim, para o perfeito funcionamento do sistema foi necessário projetar uma controladora lógica, que integrasse todos os blocos mostrados anteriormente. O projeto levou em consideração características presentes na controladora original, tais como receber os dados do computador, controlar os motores que posicionam o papel, controlar os motores que movimentam a cabeça de impressão, realimentar o sistema por meio de sensores, etc. O esquemático do circuito da unidade de controle, que consiste do microcontrolador PIC 16F877A, de um cristal, da alimentação e do botão de reinicialização é apresentado na Figura 7.18.

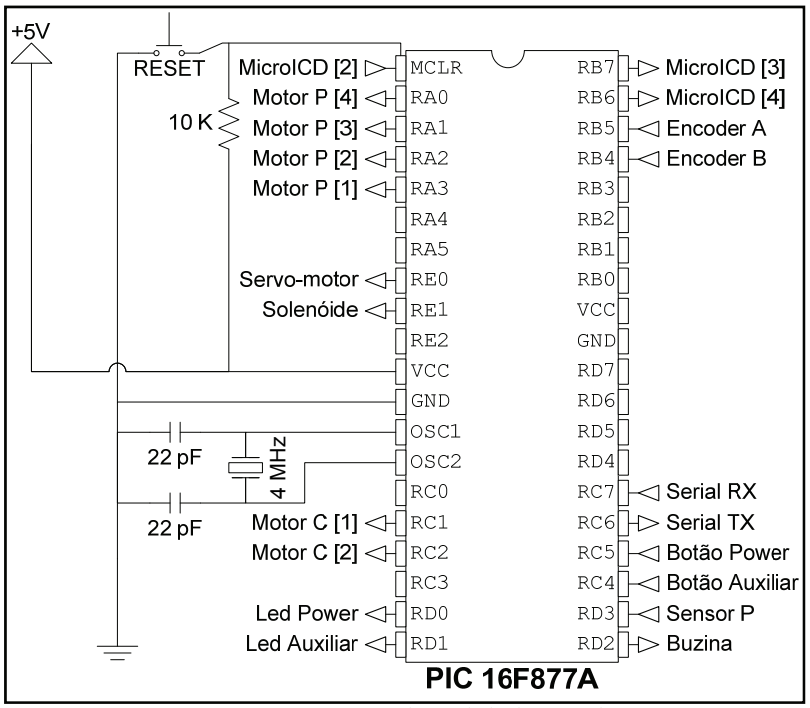


Figura 7.18: Circuito da unidade de controle

7.2.3 Montagem do circuito

Por se tratar de uma análise de viabilidade, a montagem do circuito se deu em uma placa de ilhas, pois a produção de uma placa industrial seria inviável por não se saber de antemão o melhor posicionamento para os componentes eletrônicos. A fim de evitar soldagens desnecessárias, uma simulação da posição dos componentes foi produzida em um programa gráfico, com as dimensões reais da placa de ilhas utilizada (Figura 7.19). As linhas amarelas representam tensões de +12 V, as linhas vermelhas representam tensões de +5 V, as linhas pretas representam o fio terra e, por fim, as linhas verdes representam tensões TTL.

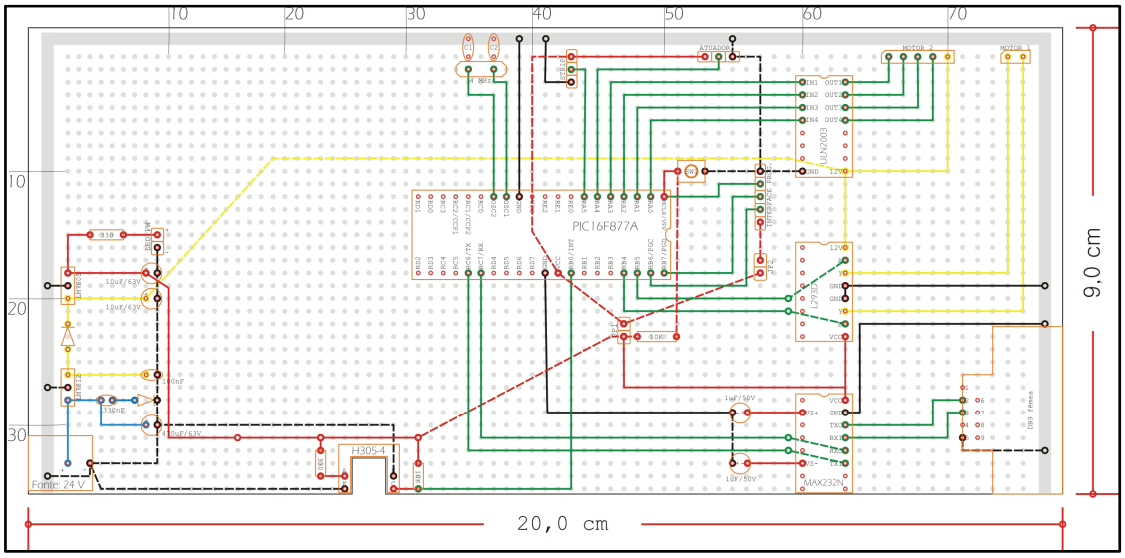


Figura 7.19: Análise da disposição dos componentes no circuito na placa de ilhas

A Figura 7.20 mostra o aspecto final da placa em escala de cinza, já instalada na impressora. Observa-se que a ilustração foi divida em blocos, tais quais apresentados no diagrama de blocos e estudados no início deste capítulo.

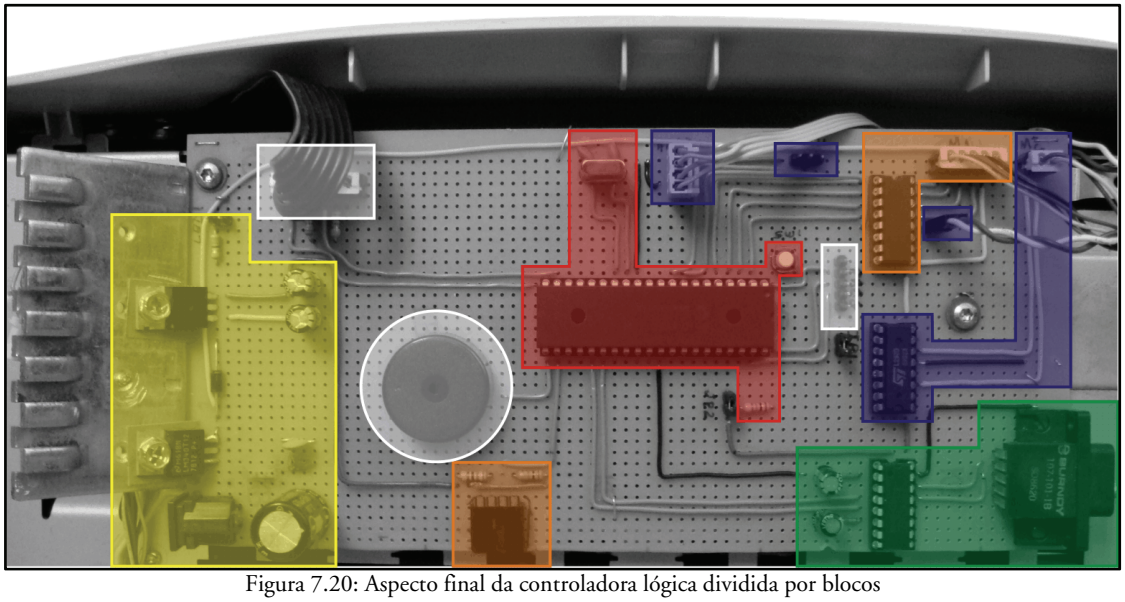


Figura 7.20: Aspecto final da controladora lógica dividida por blocos

A primeira parte a ser montada foi a alimentação do sistema (bloco amarelo). Após a montagem efetuou-se medições de tensões e correntes, bem como aferição do nível de aquecimento dos reguladores que, naquele momento, se mostrou aceitável.

A segunda parte a ser montada foi a unidade de controle (bloco vermelho). Um soquete foi usado para evitar a dessoldagem caso o microcontrolador viesse a queimar por erro de montagem do projeto, o que realmente aconteceu. Para não deixar o trabalho repetitivo, cabe dizer aqui que o os circuitos lógicos MAX232, ULN2003A e L293D foram ligados por meio de soquetes, com o mesmo objetivo do microcontrolador. No entanto, a única perda em todo o projeto foi a citada anteriormente.

A terceira parte a ser montada foi a interface de comunicação serial com o circuito lógico MAX232 (bloco verde). Neste bloco houve dificuldade em inserir o conector DB-9 fêmea na placa, em razão da falta de alinhamento dos pinos.

A quarta parte a ser montada foi o alimentador de papel (bloco laranja). Esta montagem se deu na seguinte ordem: sensor de papel, circuito integrado ULN2003A e conector do motor de passo. É importante destacar que descobrir a seqüência correta dos fios do motor de passo demandou bastante tempo.

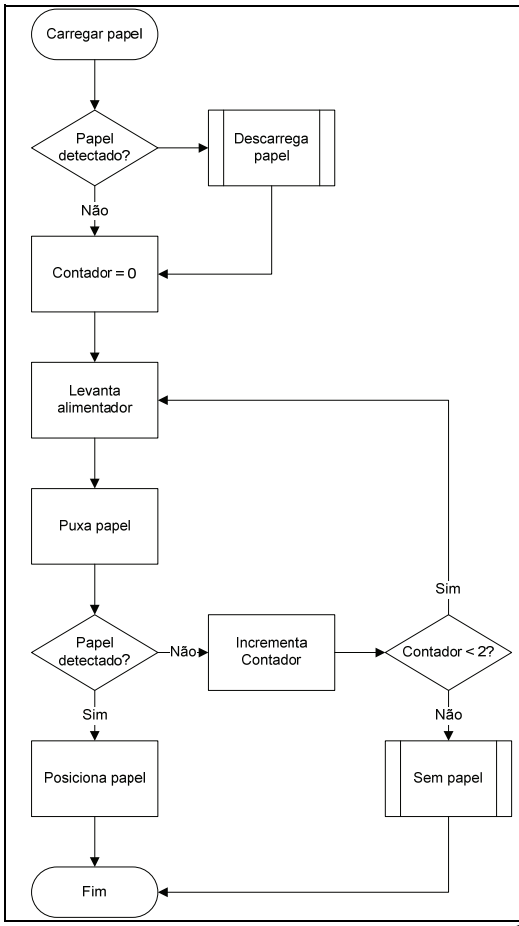
A quinta parte a ser montada foi a cabeça de impressão (bloco azul), na seguinte ordem: circuito integrado L293D, conector do motor DC, conector do sensor de movimento, conector do solenóide e conector do servo-motor.

A sexta e última parte foi a montagem das interfaces de usuário e de programação (bloco branco). Esta etapa, aparentemente simples, requereu bastante tempo, haja vista que a idéia era aproveitar as características da impressora original. Assim, antes de proceder com a soldagem da interface com o usuário na ordem correta, foi necessário estudar o comportamento do painel de controle (*leds* e botões) original. Além disso, foi adicionado o *buzzer* para a emissão de sinais sonoros e a interface de programação.

Concluída a montagem, novos testes indicaram que quando a controladora lógica estava em pleno funcionamento, havia superaquecimento nos reguladores de tensão, motivo da inclusão do dissipador.

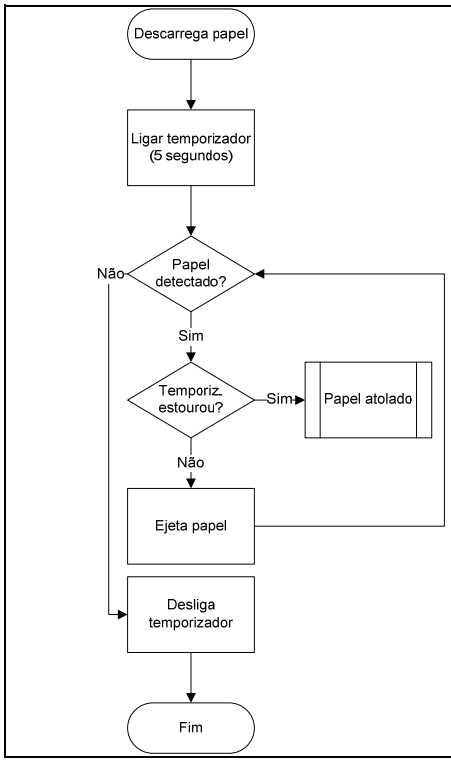
7.2.4 *Software*

Para auxiliar o desenvolvimento, vários fluxogramas de dados foram produzidos e estes, juntamente com o algoritmo estruturado, seguem nas figuras seguintes.



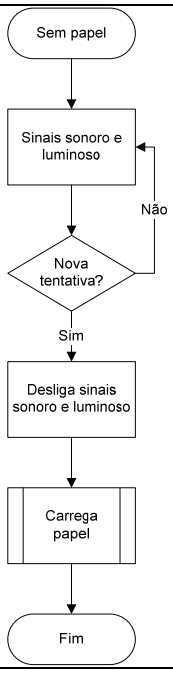

```
função carrega papel
  se papel detectado
    chama descarrega papel
  fim-se
  enquanto estiver sem papel faça
    se tentativa inferior a duas
      incrementa tentativa
      levanta alimentador
      gira motor papel 140x para carga
    fim-se
    se estiver sem papel
      chama sem papel
    fim-se
  fim-enquanto
fim-função
```

Figura 7.21: Fluxograma da função carrega papel



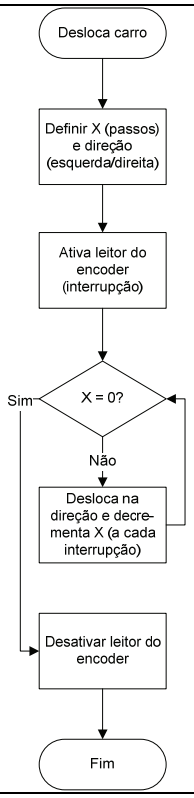

```
função descarrega papel
  liga temporizador
  se papel detectado
    gira motor do papel para ejeção
    se temporizador estourou
      se papel detectado
        chama papel atolado
      fim-se
    fim-se
  fim-se
  desliga temporizador
fim
```

Figura 7.22: Fluxograma da função descarrega papel



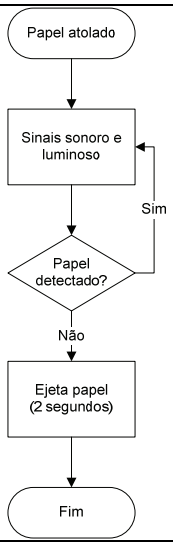

```
função sem papel
  enquanto não houver nova tentativa
    emite sinal sonoro
    emite sinal luminoso
  fim-enquanto
  chama carrega papel
fim
```

Figura 7.23: Fluxograma da função sem papel




```
função desloca carro(x, direção)
  ativa leitor do encoder
  enquanto x > 0
    se direção direita
      gira motor para direita
    senão
      gira motor do carro para esquerda
    fim-se
    decrementa x na leitura do encoder
  fim-enquanto
  desativa leitor do encoder
fim
```

Figura 7.24: Fluxograma da função desloca carro




```
função papel atolado
  enquanto papel detectado
    emite sinal sonoro
    emite sinal luminoso
  fim-enquanto
  gira motor para ejetar papel por 2 s
fim
```

Figura 7.25: Fluxograma da função papel atolado


```
função inicializa carro
  ajusta PWM para 30%
  liga temporizador em 1,5 s
  enquanto temporizador não estoura
    gira motor do carro para a direita
  fim-enquanto
  desliga temporizador
fim
```

Figura 7.26: Fluxograma da função inicializa carro

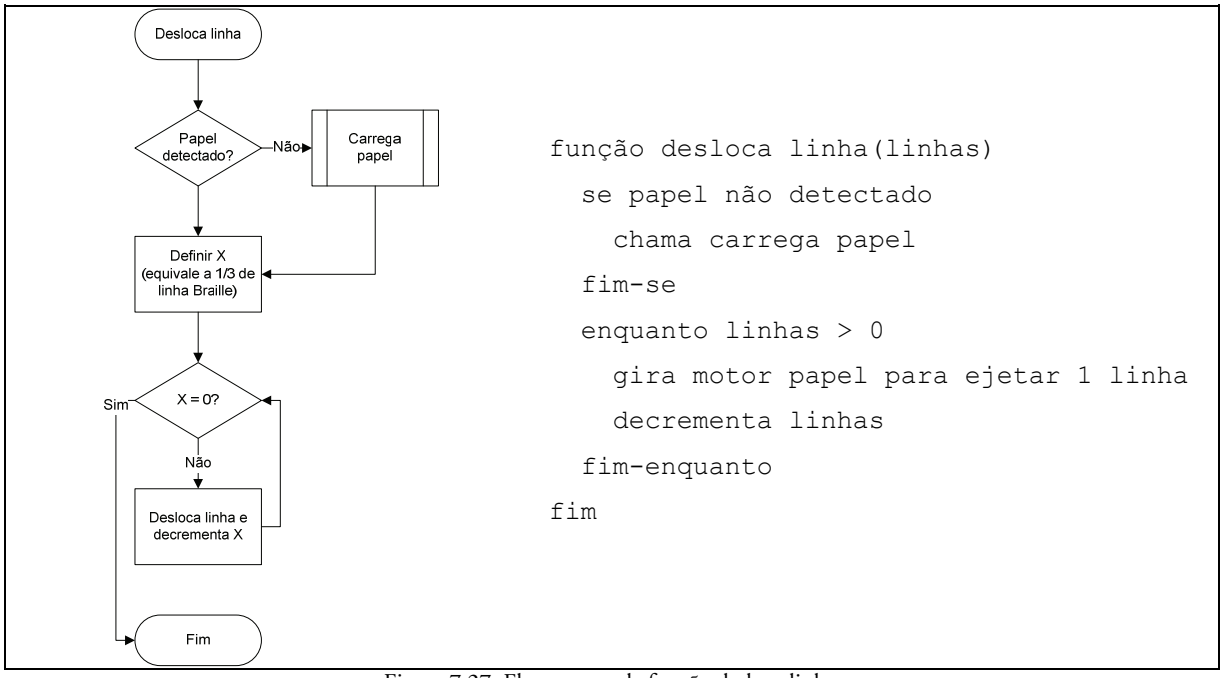


Figura 7.27: Fluxograma da função desloca linha

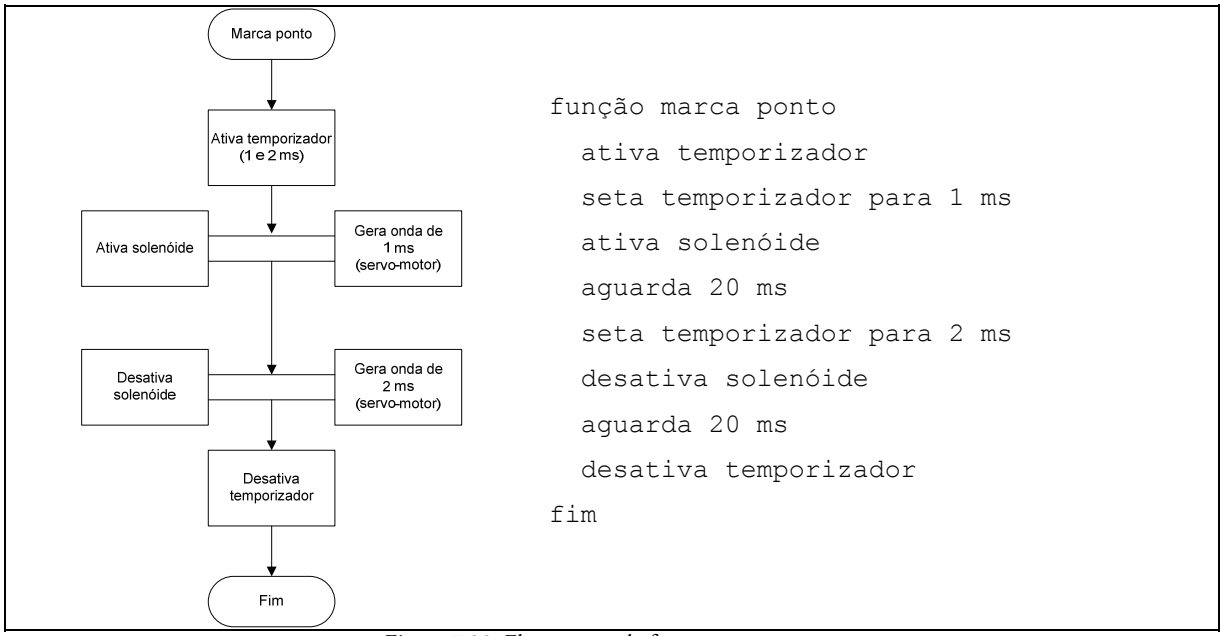


Figura 7.28: Fluxograma da função marca ponto

Conforme já mencionado, a linguagem escolhida para programar o microcontrolador foi a linguagem *C*, no ambiente do *MPLAB IDE* e compilado com o *CSS*. Depois de compilado, o código foi gravado no microcontrolador por meio do *PICKit*, *software* que trabalha em conjunto como *MicroICD*.

O código fonte pode ser consultado nos APÊNDICES A e B. Ressalta-se que a codificação não seguiu à risca os fluxogramas por motivos técnicos/práticos.

Finalmente, cabe destacar que uma das metas do projeto era desenvolver uma interface de usuário utilizando programação VBA (*Visual Basic for Applications*) no *Microsoft Word*. O objetivo disto seria dividir o texto em linhas correspondentes ao tamanho suportado pela impressora (cerca de vinte celas Braille por linha). Esta idéia, que não chegou a ser desenvolvida, originou-se de uma visita feita a uma escola especializada no ensino para deficientes visuais, onde a divisão das linhas era feita de forma manual, em um *software* de impressão em Braille.

Outra meta, também não alcançada, era desenvolver a camada de *software* da interface de comunicação. Esta, apesar de parecer primordial em um primeiro momento, não constituiu etapa impeditiva para o alcance do objetivo do projeto. Isto porque uma simulação de envio de dados foi incorporada ao *firmware*, o qual é acionado ao se pressionar determinado botão da impressora em teste.

8 RESULTADOS E DISCUSSÃO

Conforme apresentado no Capítulo 7, a escolha da impressora HP 420 se deu em razão de existir, neste modelo, um batente (Figura 8.1) que facilita o funcionamento da cabeça de impressão. É importante destacar que, em razão do tempo disponível, essa característica se mostrou essencial para o desenvolvimento das atividades. Entretanto, dada a dificuldade em encontrar este modelo em bom funcionamento no mercado de impressoras usadas, a série 600 é mais indicada ao projeto, desde que se produza o batente acima mencionado.

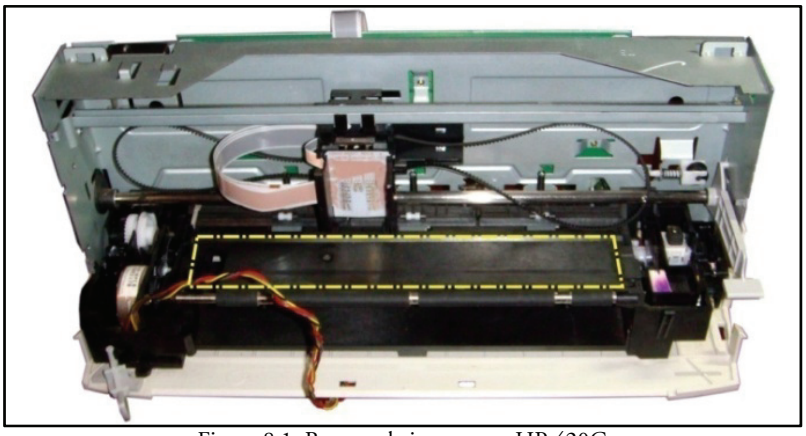


Figura 8.1: Batente da impressora HP 420C

Uma vez selecionada a impressora, os testes realizados para a aferição de seus componentes eletrônicos demonstraram que alguns deles estavam em perfeitas condições de uso, enquanto que outros apresentaram anomalias no funcionamento. Este foi o caso do sensor de papel (*photointerrupter*) e do sensor de deslocamento (*photointerrupter* com função de *encoder*), que foram substituídos por componentes de igual função. O primeiro deles foi substituído por um novo, com dimensões equivalentes. O segundo foi aproveitado de outra impressora.

Após a substituição dos componentes acima referidos, apenas o sensor de papel se mostrou satisfatório. O sensor de deslocamento, por sua vez, funcionou de maneira aleatória, não sendo possível programá-lo. Esse problema se deu provavelmente em razão da impossibilidade de aproveitar todo o conjunto, sensor de deslocamento e *encoder strip*, já que este não apresentava dimensões compatíveis. Para solucionar este problema, utilizou-se recurso de temporização do microcontrolador para o correto posicionamento do carro.

Concluído o projeto de *hardware* e *software*, o funcionamento da impressora foi atestado, por meio da impressão de diversas páginas em Braille. O resultado final pode ser visto na Figura 8.2, que compara a impressão Braille da impressora adaptada com as impressões derivadas da reglete de uma impressora Braille comercial.

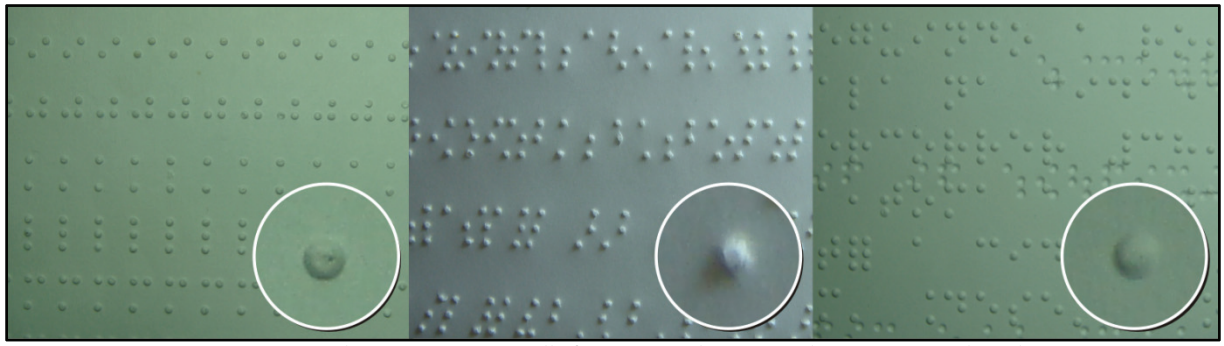


Figura 8.2: Comparação de impressões em Braille feitas com reglete (esquerda), impressora adaptada (centro) e impressora Braille comercial (direita)

Observa-se na figura acima que a impressora adaptada produziu textos com relevos acentuados, podendo ser utilizada para a escrita Braille. A falta de alinhamento nas celas se deve à impossibilidade de utilização dos sensores de deslocamento. Conforme já mencionado, a superação deste problema se deu com a utilização de recursos de temporização do microcontrolador, método que se mostrou inapropriado para o caso.

Uma alternativa para corrigir o alinhamento vertical, sem a necessidade de utilizar sensores, é a adaptação de outro motor de passo, já que a distância entre os pontos Braille não requerem a resolução fornecida pelas impressoras a jato de tinta. Este recurso foi utilizado para espaçar as linhas das celas e se mostrou adequado.

Com relação ao carregamento de papel, observou-se que, devido ao ressecamento dos *rollers*, situação comum em todos os tipos de impressora, há falhas aleatórias no alimentador de papel. Esse problema, no entanto, só pode ser resolvido com a substituição do *roller* por um novo.

Conforme esperado, o desempenho da impressora adaptada foi inferior ao de uma impressora Braille comercial e superior a produção de textos com reglete. No primeiro caso, destaca-se que a impressora comercial imprime uma cela de uma só vez, enquanto que a adaptada necessita repetir o mesmo procedimento três vezes para a impressão da mesma cela. Isto porque a cabeça de impressão adaptada tem somente uma "punção". Acredita-se ser possível adaptar o mesmo mecanismo com "três punções", o que aumentaria a velocidade de impressão de forma substancial.

Quanto a durabilidade da cabeça de impressão sabe-se que a vida útil de um solenóide é de 25 milhões de ciclos. Uma folha impressa em Braille comporta 22 linhas com 28 celas por linha. Tomando por base a impressão de uma cela cheia (seis marcações), teríamos 3.696 marcações em uma folha. Dividindo-se a vida útil do solenóide por esta quantidade, chega-se à conclusão que é possível imprimir cerca de 7.000 páginas. Considerando-se que um texto Braille

não é composto apenas por celas cheias, a durabilidade calculada anteriormente pode ser aumentada significativamente.

Finalmente, ressalta-se que as métricas levantadas para comparação entre solenóide e servo-motor indicaram que ambos os componentes podem ser utilizados para o fim pretendido. Não obstante, o solenóide é mais indicado que o servo-motor devido às seguintes características:

- maior velocidade de funcionamento, em razão da possibilidade de adaptação de um sistema automático de retorno sem que haja necessidade de especificação deste procedimento no *firmware*;
- facilidade de adaptação ao cartucho original da impressora em análise, evitando excessiva manipulação, e, por conseguinte, a possível inutilização do cartucho;
- similaridade do êmbolo com uma punção e baixo consumo de energia.

A única vantagem observada na utilização do servo-motor foi o torque, que é mais elevado que o do solenóide. No entanto, esta pode ser superada com a utilização de um solenóide de maior porte, o que acarretaria uma pequena elevação nos custos do projeto (APÊNDICE C).

Para realizar uma comparação entre a impressora Braille proposta e uma impressora Braille comercial básica, foi calculado o custo por unidade fabricada considerando uma impressora nova (Quadro 8.1).

Quadro 8.1: Custo por unidade fabricada (considerando uma impressora nova para adaptação)

Descrição	Custo (R$)
Circuitos integrados	70,00
Solenóide com mola de retorno	110,00
Componentes eletrônicos	100,00
Impressora a jato de tinta básica	200,00
TOTAL	**480,00**

Após levantado o custo por unidade fabricada, foram comparados outros parâmetros para atestar que, mesmo utilizando uma impressora nova para adaptação, a produção de uma impressora Braille de baixo custo é viável (Quadro 8.2).

Quadro 8.2: Impressora Braille comercial e a impressora Braille adaptada

Impressora Braille comercial Basic S	Impressora Braille adaptada
• Imprime em formulário contínuo	• Imprime em folha A4
• Impressão em um lado da folha	• Impressão em um lado da folha
• 150 páginas por hora	• 10 páginas por hora
• 49 caracteres por segundo	• 1,1 caracteres por segundo
• Aceita comandos por voz	• Emite sinais de alerta sonoros
• Dimensões: 52 x 24 x 12 cm	• Dimensões: 35 x 18 x 16 cm
• Peso: 8 kg	• Peso: 3 kg
• Custo: superior a R$ 15.000,00	• Custo: inferior a R$ 500,00

Analisando as características das duas impressoras pode-se dizer que a primeira é 15 vezes mais rápida e 30 vezes mais cara que a segunda. Aceitar comando de voz é algo desejado, mas não impeditivo para a correta utilização de uma impressora Braille. Para finalizar, levando em consideração as dimensões e pesos das impressoras comparadas, a impressora adaptada pode ser considerada uma impressora Braille portátil.

CONCLUSÃO E RECOMENDAÇÕES

Após realizados os procedimentos e testes descritos nos capítulos anteriores, conclui-se que o objetivo do projeto, "Viabilidade de uma impressora Braille de baixo custo" foi atingido. A adaptação de uma impressora para a escrita Braille é viável.

Entretanto, para o perfeito funcionamento da máquina, o projeto precisa de aprimoramentos. Um deles se refere à precisão no posicionamento do carro de impressão, que influencia sobremaneira na eficácia do sistema, pois a leitura Braille só é possível se as celas Braille forem impressas com o devido alinhamento.

Além do alinhamento, deseja-se a impressora produza relevos uniformes, o que pode ser conseguido por meio da produção de um batente rígido perfurado nas distâncias de conforto, a exemplo da reglete, que melhoraria a marcação dos pontos. Neste caso, as impressoras da série 600 passariam a ser as mais indicadas para adaptação, caso se utilize uma impressora usada. Um inconveniente, neste caso, seria a impossibilidade de se produzir impressões gráficas uma vez que os pontos de impressão serão determinados pelo batente.

Tem-se ainda que a velocidade de impressão ficou aquém da desejada, mas esta pode ser aumentada em até quatro vezes apenas efetuando ajustes no circuito da controladora lógica, na mecânica da impressora adaptada e no *firmware*. Pode-se aumentar, ainda, esta velocidade em três vezes caso seja possível produzir uma cabeça de impressão que marque três pontos ao mesmo tempo. Assim, considerando que as duas soluções anteriores foram implementadas com sucesso, a velocidade do de impressão da impressora adaptada passaria de 10 páginas por hora para 120 página por hora.

Os elementos utilizados no projeto foram reais e, caso não tivessem sido aproveitados, tornar-se-iam lixos eletrônicos. Dessa forma, os custos de produção foram compatíveis com a proposta de baixo custo do projeto, que teve um custo total de R$ 433,00 (quatrocentos e trinta e três reais).

Caso se deseje adaptar uma impressora nova, o custo de produção unitário seria R$ 480,00 (quatrocentos e oitenta reais). Apesar de ter sido incluído uma impressora nova nos custos, exclui-se itens que foram adquiridos em fase de projeto e não serão necessários para a produção em série. Esta opção de adaptação eliminaria grande parte das dificuldades encontradas, além de aumentar a vida útil da impressora. No entanto, deve-se lembrar que não foi incluído o custo de mão de obra, o que elevaria o custo de produção sobremaneira. Por outro lado,

considerando que uma impressora Braille comercial custa cerca de quinze mil reais, ainda assim essa alternativa seria de baixo custo.

Trabalhos futuros

- Análise da viabilidade com impressoras de outros fabricantes, para tentar baixar ainda mais o custo de produção de uma unidade;
- Controle preciso de posicionamento do carro de impressão, pois impressão Braille com celas desalinhadas é inútil;
- Camada de comunicação USB, pois é uma interface de alta velocidade e de grande aceitação no mercado;
- Integração com o software Braille fácil ou similar para que o deficiente visutal não tenha que aprender a usar mais uma ferramenta para a impressão de seus trabalhos em Braille;
- Protótipo utilizando uma impressora nova, levando em consideração o mínimo de modificações na mecânica. Deseja-se chegar ao ponto de substituir apenas a controladora lógica e o cartucho de tinta por um com o solenóide;
- Implementação de comandos por voz no *firmware* para que o deficiente visual tenha maiores opções de controle, e não dependa de apenas dois ou três botões de controle.

REFERÊNCIAS

1. **BATISTA, Cristina Abranches Mota.** Inclusão da pessoa portadora de deficiência no mercado formal de trabalho e sua correlação com a prática de responsabilidade social. [A. do livro] Armindo dos Santos de Sousa Teodósio. *Gestão Inclusiva: primeiro, segundo e terceiro setor.* Belo Horizonte : Armazém de Idéias, 2003, pp. 49-70.

2. **BRASIL, Ministério da Justiça.** *Mídia e Deficiência: Manual de Estilo.* [Online] CORDE, Secretaria dos Direitos da Cidadania Nacional para Integração da Pessoa de Deficiência, 2003. [Citado em: 28 de março de 2008.] http://www.mj.gov.br/sedh/ct/CORDE/dpdh/sicorde/midia.asp.

3. **BARANAUSKAS, M. Cecília C. e MANTON, Maria Teresa Eglér.** Acessibilidade em Ambientes Educacionais: para Além das Guidelines. [A. do livro] Antoni Augusto Fasolo Quevedo, José Raimundo de Oliveira e Maria Teresa Eglér Manton. *Mobilidade, Comunicação e Educação: Desafios à Acessibilidade.* Campinas : WVA Editora, 2000, pp. 133-148.

4. **CERTIC - Centro de Engenharia de Reabilitação e Acessibilidade.** *História do Braille.* [Online] [Citado em: 24 de maio de 2008.] http://www.acessibilidade.net/mecbraille/braille.php.

5. **SENAI.** *Programa SENAI de Ações Inclusivas.* [Online] 2007. [Citado em: 5 de maio de 2008.] http://www.senai.br/psai/braille_contato.asp.

6. **HAMPSHIRE, Berry.** *La práctica del braille - El braille como meio de comunicación.* Paris : UNESCO, 1981. 187 p.

7. **BRASIL, Ministério da Educação.** *Grafia Braille para a Língua Portuguesa.* 2ª edição. Brasília : SEESP, 2006. 106 p.

8. —. *Normas técnicas para a produção de textos em Braille.* 2ª edição. Brasília : SEESP, 2006. 73 p.

9. **UnB.** Sistema Braille da Língua Portuguesa. Brasília : Laboratório de atendimento ao deficiente visual - LDV. ldv@fe.unb.br.

10. **TORRES, Gabriel.** *Hardware: Curso Completo.* 4ª edição. Rio de Janeiro : Axcel Books, 2001. 1398 p.

11. **HOWSTUFFWORKS.** *How Inkjet Printers Work.* [Online] 2001. [Citado em: 22 de setembro de 2008.] http://computer.howstuffworks.com/inkjet-printer2.htm.

12. **ZELENOVSKY, Ricardo e MENDONÇA, Alexandre.** *PC: Um Guia Prático de Hardware e Interfaceamento.* 3ª edição. Rio de Janeiro : MZ, 2002. 1020 p.

13. **MESSIAS, Antônio Rogério.** Porta Paralela. *ROGERCOM.* [Online] 2006. [Citado em: 24 de abril de 2008.] http://www.rogercom.com/pparalela.

14. **HOWSTUFFWORKS.** *How Serial Ports Works.* [Online] 2001. [Citado em: 12 de setembro de 2008.] http://computer.howstuffworks.com/serial-port.htm.

15. **AXELSON, Jan.** *Serial Port Complete: Programming and Circuits for RS-232 and RS-485 Links and Networks.* Madison : Lakeview Research, 2000. 321 p.

16. **IBRAHIM, Dogan.** *Advanced PIC Microcontroller Projects in C - From USB to ZIGBEE with the PIC 18F Series.* Burlington : Elsevier, 2008. 560 p.

17. **MAXIM.** +5V-Powered, Multichannel RS-232 - Drivers/Receivers. *MAX220-MAX249.* [Online] 2003. [Citado em: 24 de 9 de 2008.] http://www.maxim-ic.com.

18. **MICROCHIP.** *Brushed DC Motor Fundamentals.* [Online] 2004. [Citado em: 5 de julho de 2008.] 10 p. http://ww1.microchip.com/downloads/en/AppNotes/00905a.pdf.

19. **ALLEGRO.** *Compumotor Engineering Reference Guide.* [Online] 1997. [Citado em: 26 de setembro de 2008.] http://www.allegromicro.com/en/products/Design/compumot/.

20. **HOWSTUFFWORKS.** *How Electric Motors Work.* [Online] 2001. [Citado em: 13 de setembro de 2008.] http://electronics.howstuffworks.com/motor1.htm.

21. **SMC Eletronics.** *HP Motors.* [Online] [Citado em: 4 de junho de 2008.] http://www.smcelectronics.com/printrep.htm.

22. **PREDKO, Myke.** *Programming and Customizing the PIC Microcontroller.* New York : Mc Graw Hill, 2008. 1293 p.

23. **TEXAS INSTRUMENTS.** Datasheet. *Quadruple half-drivers L293, L293D.* [Online] 2002. [Citado em: 18 de junho de 2008.] http://www.datasheetcatalog.org/datasheet/texasinstruments/l293d.pdf.

24. **TORO, DEL.** *Fundamentos de Máquinas Elétricas.* Rio de Janeiro : Prentice Hall, 1990. 550 p.

25. **GEDAE - Grupo de Estudo e Desenvolvimento de Aplicações Eletrônicas.** *Controle de motores de passo unipolares de quatro fases de ímã permanente (5 ou 6 fios).* [Online] 2005. [Citado em: 3 de junho de 2008.] http://br.geocities.com/gedaepage/Doc/MP_5fios.htm.

26. **GUARDIAN, Eletric Manufaturing CO.** *Linear Solenoids.* [Online] [Citado em: 30 de maio de 2008.] http://www.guar dian-electric.com/sole.htm.

27. **TSC, The Solenoid Company.** *Linear Solenoids.* [Online] 2007. [Citado em: 4 de junho de 2008.] http://www.thesolenoidcompany.com/solenoidassistant_linear_singleacting.php.

28. **JOHNSON, Eletric.** *Rotary Solenoids e Linear Solenoids.* [Online] [Citado em: 29 de maio de 2008.] http://www. solenoids.com.

29. **IOVINE, John.** *PIC Robotics: A beginner's Guide to Robotics Projects Using the PICmicro.* New York : McGraw-Hill, 2004. 289 p.

30. **GEDAE - Grupo de Estudo e Desenvolvimento de Aplicações Eletrônicas.** *Controle de Servomotores de Posição tipo RC.* [Online] 20__. [Citado em: 26 de setembro de 2008.] http://br.geocities.com/gedaepage/Doc/Artigo_Sevo.htm.

31. **BISHOP, Robert H.** *The Mecatronics Handbook.* Austin : CRC Press, 2002. 1229 p.

32. **BORENSTEIN, J., EVERETT, H. R. e FENG, L.** *"Where I am" - Sensors and Methods for Mobile Robot Positioning.* Michigan : The University of Michigan, 1996. 282 p.

33. **BATES, Martin.** *Interfacing PIC Microcontrollers: Embedded Design by Interactive Simulation.* San Diego : Elsevier, 2006. 313 p.

34. **MICROCHIP.** Datasheet. *PIC16F877A.* [Online] 2004. [Citado em: 21 de abril de 2008.] http://ww1.microchip. com/downloads/en/devicedoc/40044b.pdf.

35. **HEWLETT PACKARD.** HP United States. *Computer, Servers, Laptops, Printers and more.* [Online] [Citado em: 28 de 10 de 2008.] http://www.hp.com/country/us/en/welcome_acc_supp.html.

36. —. *HP Parts Reference Guide.* [Online] 2004. [Citado em: 28 de abril de 2008.] http://www.hp.com/go/ hpparts.

37. **MASINI, Elcie F. Salzano.** *O perceber e o relacionar-se do deficiente visual: orientando professores especializados.* Brasília : Corde, 1994. 161 p.

APÊNDICE A: Código fonte – arquivo *fw_braille_printer.h*

```
#include <16F877A.h>                    //Inclusão de bib1lioteca
#device ADC=8                           //Identificação do PIC
#fuses NOWDT                            //Sem Watch Dog Timer
#fuses XT                               //Oscilador de Crystal
#fuses NOPUT                            //Sem temporizador Power Up
#fuses NOPROTECT                        //Código não protegido
#fuses NODEBUG                          //Não permitir debuger
#fuses NOBROWNOUT                       //Sem reset por brown out
#fuses NOLVP                            //Alta tensão para programação
#fuses NOCPD                            //Sem proteção EE
#fuses NOWRT                            //Sem proteção de memória
#use DELAY(CLOCK=4000000)               //Oscilador de 4 MHz
#use FAST_IO(A)                         //O programador trata I/O, porta A
#use FAST_IO(B)                         //O programador trata I/O, porta B
#use FAST_IO(C)                         //O programador trata I/O, porta C
#use FAST_IO(D)                         //O programador trata I/O, porta D
#use FAST_IO(E)                         //O programador trata I/O, porta E
#priority RB, RTCC                      //Prioridade para a interrupção RB
#define MOTOR_PAPEL PORTA               //Porta A dedicada ao motor de passo
#define PUNCAO_SOL PIN_E0               //Punção no RE0
#define PUNCAO_SER PIN_E1               //Servo-motor no RE1
#define SENSOR_ENC_A PIN_B4             //Sensor A do encoder no RB4
#define SENSOR_ENC_B PIN_B5             //Sensor B do encoder no RB5
#define MOTOR_CARRO_E PIN_C1            //Motor DC, esquerda, no RC1
#define MOTOR_CARRO_D PIN_C2            //Motor DC, direita, no RC2
#define BOTAO_AUX PIN_C4                //Botão auxiliar no RC4
#define BOTAO_PRI PIN_C5                //Botão principal no RC5
#define SERIAL_TX PIN_C6                //Transmissão serial no RC6
#define SERIAL_RX PIN_C7                //Recepção serial no RC7
#define LED_PRI PIN_D0                  //LED primário no RD0
#define LED_AUX PIN_D1                  //LED auxíliar no RD1
#define BUZZER PIN_D2                   //Sinal sonoro no RD2
#define SENSOR_PAPEL PIN_D3             //Sensor de papel no RD3
#define esquerda 0                      //Esquerda = 0
#define direita 1                       //Direita = 0
#define cima 1                          //Cima = 1
#define baixo 0                         //Baixo = 0
#define dc_off 0                        //Duty Cycle = 0% -> off
#define dc_10 102                       //Duty Cycle = 10%
#define dc_15 153                       //Duty Cycle = 15%
#define dc_50 512                       //Duty Cycle = 50%
#define brl_sublinha 7                  //1/3 linha braille
#define brl_linha 15                    //1 linha braille
#define meia_cela 8                     //meia cela
#define cela 12                         //cela
```

APÊNDICE B: Código fonte – arquivo *fw_braille_printer.c*

```
1  /*********************************************************
2  IESB - Instituto de Ensio Superior de Brasília
3
4  Curso...........: Engenharia de Computação
5  Projeto.........: Firmware para Impressora Braille
6  Autor...........: Charles Henrique Gonçalves Santos
7  Descrição.......: Firmware da impressora Braille
8  Orientador......: Prof. MSc Saulo Pimentel Wulhynek
9  Local...........: Brasília (DF), 27 de outrobro de 2008
10 Microcontrolador: PIC16F877A
11 Oscilador.......: XT, 4.0000 MHz
12 Compilador......: CCS C, 4.0.14
13 Versão do fw....: 1.00b
14 Última revisão..: 15/11/2008
15 *********************************************************/
16
17 /*********************************************************
18 Inclusões
19 *********************************************************/
20 #include <fw_braille_printer.h>
21 #include <string.h>
22
23 /*********************************************************
24 Variáveis Globais
25 *********************************************************/
26 const char Passo[] =
27 {
28   //1 seq. = 7,50° polar
29   //48 * 4 -> 192 seqs. para uma volta completa
30   0b00001000, 0b00000100, 0b00000010, 0b00000001
31 };
32 const char uPasso[] =
33 {
34   //1 seq. = 3,75° polar
35   //48 * 8 -> 384 seqs. para uma volta completa
36   0b00001000, 0b00001100, 0b00000100, 0b00000110,
37   0b00000010, 0b00000011, 0b00000001, 0b00001001
38 };
39
40 /*********************************************************
41 Escopo das Funções
42 *********************************************************/
43 void inicializa_sistema();
44 void carrega_papel();
45 void descarrega_papel();
46 void desloca_carro(short direcao, int distancia);
47 void inicializa_carro();
48 void desloca_linha(int linhas);
49 byte tabela_braille(char letra);
50 void imprime_buffer(char *buffer_braille);
51 void imprime_grub(int qtd_pontos, short direcao, short marcar, short
       mover_linha);
52 void marca_ponto();
53 void teste_impressao_texto();
54 void teste_impressao_grafico();
55
56 void main()
57 {
58   /*******************************************************
59   Código principal
60   *******************************************************/
61
62   inicializa_sistema();
63   output_high(LED_PRI);
```

```
64
65    while(true)
66    {
67      while(!input(BOTAO_PRI))
68      {
69        //debouncer
70        delay_ms(100);
71        teste_impressao_texto();
72      }
73
74      while(!input(BOTAO_AUX))
75      {
76        //debouncer
77        delay_ms(100);
78        teste_impressao_grafico();
79      }
80    }
81  }
82
83
84  void inicializa_sistema()
85  {
86    /*********************************************************
87    Objetivo: Configurar os parâmetros iniciais do sistema:
88    » Inicialização das portas
89    » Portas de entrada e saída
90    » Configuração dos registradores
91    » Configuração das interrupções
92    Entradas: Nenhuma
93    Saídas..: Nenhuma
94    *********************************************************/
95
96    /*********************************************************
97    Inicialização das portas
98    *********************************************************/
99    output_A(0x00); //Inicializa porta A
100   output_B(0x00); //Inicializa porta B
101   output_C(0x00); //Inicializa porta C
102   output_D(0x00); //Inicializa porta D
103   output_E(0x00); //Inicializa porta E
104
105   /*********************************************************
106   Definição das entradas e saídas das portas
107   » 0 = saída
108   » 1 = entrada
109   set_tris_x(0b01001100) -> RX0 (saída)
110   -> RX1 (saída)
111   -> RX2 (entrada)
112   -> RX3 (entrada)
113   -> RX4 (saída)
114   -> RX5 (saída)
115   -> RX6 (entrada)
116   -> RX7 (saída)
117   *********************************************************/
118   set_tris_a(0b00000000); //Configura porta A
119   set_tris_b(0b00110000); //Configura porta B
120   set_tris_c(0b10110000); //Configura porta C
121   set_tris_d(0b00001000); //Configura porta D
122   set_tris_e(0b00000000); //Configura porta E
123
124   /*********************************************************
125   Configuração dos registradores
126   *********************************************************/
127   setup_adc_ports(NO_ANALOGS);
128   setup_adc(ADC_OFF);
129   setup_psp(PSP_DISABLED);
130   setup_spi(FALSE);
131   //Timer1 -> 32.7 ms
```

```
132   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_128);
133   setup_timer_1(T1_DISABLED);
134   //Timer2 -> 136 us
135   setup_timer_2(T2_DIV_BY_1, 135, 1);
136   setup_comparator(NC_NC_NC_NC);
137   setup_vref(FALSE);
138   //PWM -> 0 = off; 1024 = 100%
139   setup_ccp1(ccp_pwm);
140   setup_ccp2(ccp_pwm);
141   set_pwm1_duty(dc_off);
142   set_pwm2_duty(dc_off);
143   //enable_interrupts(GLOBAL);
144 }
145
146 void carrega_papel()
147 {
148   int i, j, tentativa;
149   short sem_papel = true;
150
151   while (sem_papel == true)
152   {
153     for (tentativa = 0; tentativa < 2; tentativa++)
154     {
155       //Posiciona papel no alimentador
156       for(i=0; i < 140; i++)
157       {
158         for (j = 0; j < 8; j++)
159         {
160           output_a(uPasso[j]);
161           delay_ms(3);
162         }
163       }
164       delay_ms(300);
165
166       //Tenta carregar
167       for(i = 0; i < 250; i++)
168       {
169         for (j = 0; j < 4; j++)
170         {
171           output_a(Passo[3-j]);
172           delay_ms(3);
173         }
174       }
175
176       //Verifica se carregou
177       if (input(SENSOR_PAPEL))
178       {
179         sem_papel = false;
180         break;
181       }
182     }
183     if (!input(SENSOR_PAPEL))
184     {
185       while (input(BOTAO_AUX))
186       {
187         output_high(BUZZER);
188         output_high(LED_AUX);
189         delay_ms(200);
190         output_low(BUZZER);
191         output_low(LED_AUX);
192         delay_ms(200);
193       }
194     }
195   }
196 }
197
```

```
198 void descarrega_papel()
199 {
200
201   int i, j, contador = 0;
202
203   while (input(SENSOR_PAPEL))
204   {
205     for(i = 0; i < 10; i++)
206       for (j = 0; j < 4; j++)
207       {
208         Output_A(Passo[3-j]);
209         delay_ms(3);
210       }
211
212       contador++;
213
214       if (contador == 100)
215       {
216         while (true)
217         {
218           output_high(BUZZER);
219           output_high(LED_AUX);
220           output_high(LED_PRI);
221           delay_ms(1000);
222           output_low(BUZZER);
223           output_low(LED_AUX);
224           output_low(LED_PRI);
225           delay_ms(2000);
226         }
227       }
228   }
229   for(i = 0; i < 200; i++)
230   {
231     for (j = 0; j < 4; j++)
232     {
233       Output_A(Passo[3-j]);
234       delay_ms(3);
235     }
236   }
237 }
238
239 void desloca_carro(short direcao, int distancia)
240 {
241   int contador = 0;
242   while (contador < distancia)
243   {
244     if (direcao == direita)
245       set_pwm1_duty(dc_15);
246     else
247       set_pwm2_duty(dc_15);
248     delay_us(1500);
249     set_pwm1_duty(dc_off);
250     set_pwm2_duty(dc_off);
251     contador++;
252   }
253   delay_ms(200);
254 }
255
256 void inicializa_carro()
257 {
258   set_pwm2_duty(dc_10);
259   delay_ms(1300);
260   set_pwm2_duty(dc_off);
261   set_pwm1_duty(dc_50);
262   delay_ms(90);
263   set_pwm1_duty(dc_off);
264   delay_ms(500);
265 }
```

```
266 void desloca_linha(int linhas)
267 {
268   int i, j;
269   for(i = 0; i <= linhas; i++)
270   {
271     for (j = 0; j < 4; j++)
272     {
273       output_a(Passo[3-j]);
274       delay_ms(3);
275     }
276   }
277 }
278 byte tabela_braille(char letra)
279 {
280
281   /*******************************************************
282   Forma de conversão
283   a = [1]; b = [12]; d = [145] é = [123456]
284   braille_letra = [ x x x x x x - -]
285   a em braille_letra = [10000000]
286   b em braille_letra = [11000000]
287   d em braille_letra = [10011000]
288   é em braille_letra = [11111100]
289   76543210
290
291   *******************************************************/
292
293   switch (letra)
294   {
295     case 'a': return 0b10000000; break;
296     case 'b': return 0b11000000; break;
297     case 'c': return 0b10010000; break;
298     case 'd': return 0b10011000; break;
299     case 'e': return 0b10001000; break;
300     case 'f': return 0b11010000; break;
301     case 'g': return 0b11011000; break;
302     case 'h': return 0b11001000; break;
303     case 'i': return 0b01010000; break;
304     case 'j': return 0b01011000; break;
305     case 'k': return 0b10100000; break;
306     case 'l': return 0b11100000; break;
307     case 'm': return 0b10110000; break;
308     case 'n': return 0b10111000; break;
309     case 'o': return 0b10101000; break;
310     case 'p': return 0b11110000; break;
311     case 'q': return 0b11111000; break;
312     case 'r': return 0b11101000; break;
313     case 's': return 0b01110000; break;
314     case 't': return 0b01111000; break;
315     case 'u': return 0b10100100; break;
316     case 'v': return 0b11100100; break;
317     case 'w': return 0b01011100; break;
318     case 'x': return 0b10110100; break;
319     case 'y': return 0b10111100; break;
320     case 'z': return 0b10101100; break;
321     case ' ': return 0b00000000; break;
322     case 'é': return 0b11111100; break;
323   }
324 }
```

```
325 void imprime_buffer(char *buffer_braille)
326 {
327   int sublinha, max, i;
328   byte letra;
329
330   max = strlen(buffer_braille);
331
332   //Execução de três sublinhas
333   //1ª sublinha: pontos 1 e 4 das 20 celas
334   //2ª sublinha: pontos 2 e 5 das 20 celas
335   //3ª sublinha: pontos 3 e 6 das 20 celas
336   inicializa_carro();
337   for (sublinha = 1; sublinha < 4; sublinha++)
338   {
339     for(i = 0; i < max; i++)
340     {
341       letra = tabela_braille(buffer_braille[i]);
342       if(sublinha == 1)
343       {
344         if(bit_test(letra, 7)) marca_ponto();
345         desloca_carro(direita, meia_cela);
346         if(bit_test(letra, 4)) marca_ponto();
347         if (i < max - 1) desloca_carro(direita, cela);
348       }
349       if(sublinha == 2)
350       {
351         if(bit_test(letra, 3)) marca_ponto();
352         desloca_carro(esquerda, meia_cela);
353         if(bit_test(letra, 6)) marca_ponto();
354         if (i < max - 1) desloca_carro(esquerda, cela);
355       }
356       if(sublinha == 3)
357       {
358         if(bit_test(letra, 5)) marca_ponto();
359         desloca_carro(direita, meia_cela);
360         if(bit_test(letra, 2)) marca_ponto();
361         if (i < max - 1) desloca_carro(direita, cela);
362       }
363     }
364     desloca_linha(brl_sublinha);
365   }
366   desloca_linha(brl_linha);
367 }
368
369 void imprime_grub(int qtd_pontos, short direcao, short marcar,
    short mover_linha)
370 {
371   int i;
372   for (i = 0; i < qtd_pontos; i++)
373   {
374     if (i > 0) desloca_carro(direcao, 8);
375     if (marcar == true) marca_ponto();
376   }
377   if (mover_linha == true) desloca_linha(10);
378 }
379
380 void marca_ponto()
381 {
382   output_high(PUNCAO_SOL);
383   delay_ms(20);
384   output_low(PUNCAO_SOL);
385   delay_ms(50);
386 }
387
```

```
388 void teste_impressao_texto()
389 {
390   char buffer[20];
391 
392   carrega_papel();
393   desloca_linha(brl_sublinha);
394   desloca_linha(brl_sublinha);
395   desloca_linha(brl_sublinha);
396   strcpy(buffer, "viabilidade");
397   imprime_buffer(buffer);
398   strcpy(buffer, "de");
399   imprime_buffer(buffer);
400   strcpy(buffer, "uma");
401   imprime_buffer(buffer);
402   strcpy(buffer, "impressora");
403   imprime_buffer(buffer);
404   strcpy(buffer, "braille");
405   imprime_buffer(buffer);
406   strcpy(buffer, "de");
407   imprime_buffer(buffer);
408   strcpy(buffer, "baixo");
409   imprime_buffer(buffer);
410   strcpy(buffer, "custo");
411   imprime_buffer(buffer);
412   descarrega_papel();
413 }
414 void teste_impressao_grafico()
415 {
416   carrega_papel();
417   desloca_linha(brl_sublinha);
418   desloca_linha(brl_sublinha);
419   desloca_linha(brl_sublinha);
420 /* ########################################
421    ##################################    ##
422    ##############################      ####
423    ##########################        #######
424    ####################          ##########
425    ################            ############
426    ##########              ################
427    #####                 ##################
428    #                   ####################
429                      ######################
430                    ######################
431                ##################################
432             ###################################
433           ###################################
434        ########################################
435    ########################################
436    ##    ##         ###       ##       ####
437    ##    ##    ######    #######   ##   ###
438    ##    ##         ###      ###       ###
439    ##    ##    ############   ##   ###   ##
440    ##    ##         ##       ###         ##
441    ######################################## */
442   inicializa_carro();
443   imprime_grub(40, direita , true , true ); //linha 1
444   imprime_grub(2 , esquerda, true , false); //linha 2
445   imprime_grub(4 , esquerda, false, false);
446   imprime_grub(34, esquerda, true , true );
447   imprime_grub(30, direita , true , false); //linha 3
448   imprime_grub(6 , direita , false, false);
449   imprime_grub(4 , direita , true , true );
450   imprime_grub(7 , esquerda, true , false); //linha 4
451   imprime_grub(8 , esquerda, false, false);
452   imprime_grub(25, esquerda, true , true );
453   imprime_grub(20, direita , true , false); //linha 5
454   imprime_grub(10, direita , false, false);
455   imprime_grub(10, direita , true , true );
```

```
456   imprime_grub(13, esquerda, true , false); //linha 6
457   imprime_grub(12, esquerda, false, false);
458   imprime_grub(15, esquerda, true , true );
459   imprime_grub(10, direita , true , false); //linha 7
460   imprime_grub(14, direita , false, false);
461   imprime_grub(16, direita , true , true );
462   imprime_grub(18, esquerda, true , false); //linha 8
463   imprime_grub(17, esquerda, false, false);
464   imprime_grub(5 , esquerda, true , true );
465   imprime_grub(1 , direita , true , false); //linha 9
466   imprime_grub(18, direita , false, false);
467   imprime_grub(21, direita , true , true );
468   imprime_grub(24, esquerda, true , false); //linha 10
469   imprime_grub(16, esquerda, false, true );
470   imprime_grub(14, direita , false, false); //linha 11
471   imprime_grub(26, direita , true , true );
472   imprime_grub(29, esquerda, true , false); //linha 12
473   imprime_grub(11, esquerda, false, true );
474   imprime_grub(8 , direita , false, false); //linha 13
475   imprime_grub(32, direita , true , true );
476   imprime_grub(34, esquerda, true , false); //linha 14
477   imprime_grub(6 , esquerda, false, true );
478   imprime_grub(3 , direita , false, false); //linha 15
479   imprime_grub(37, direita , true , true);
480   imprime_grub(40, esquerda, true , true ); //linha 16
481   imprime_grub(2 , direita , true , false); //linha 17
482   imprime_grub(4 , direita , false, false);
483   imprime_grub(2 , direita , true , false);
484   imprime_grub(9 , direita , false, false);
485   imprime_grub(3 , direita , true , false);
486   imprime_grub(7 , direita , false, false);
487   imprime_grub(2 , direita , true , false);
488   imprime_grub(7 , direita , false, false);
489   imprime_grub(4 , direita , true , true );
490   imprime_grub(3 , esquerda, true , false); //linha 18
491   imprime_grub(3 , esquerda, false, false);
492   imprime_grub(2 , esquerda, true , false);
493   imprime_grub(3 , esquerda, false, false);
494   imprime_grub(7 , esquerda, true , false);
495   imprime_grub(4 , esquerda, false, false);
496   imprime_grub(6 , esquerda, true , false);
497   imprime_grub(4 , esquerda, false, false);
498   imprime_grub(2 , esquerda, true , false);
499   imprime_grub(4 , esquerda, false, false);
500   imprime_grub(2 , esquerda, true , true );
501   imprime_grub(2 , direita , true , false); //linha 19
502   imprime_grub(4 , direita , false, false);
503   imprime_grub(2 , direita , true , false);
504   imprime_grub(9 , direita , false, false);
505   imprime_grub(3 , direita , true , false);
506   imprime_grub(6 , direita , false, false);
507   imprime_grub(3 , direita , true , false);
508   imprime_grub(8 , direita , false, false);
509   imprime_grub(3 , direita , true , true );
510   imprime_grub(2 , esquerda, true , false); //linha 20
511   imprime_grub(3 , esquerda, false, false);
512   imprime_grub(3 , esquerda, true , false);
513   imprime_grub(3 , esquerda, false, false);
514   imprime_grub(2 , esquerda, true , false);
515   imprime_grub(4 , esquerda, false, false);
516   imprime_grub(12, esquerda, true , false);
517   imprime_grub(3 , esquerda, false, false);
518   imprime_grub(2 , esquerda, true , false);
519   imprime_grub(4 , esquerda, false, false);
520   imprime_grub(2 , esquerda, true , true );
521   imprime_grub(2 , direita , true , false); //linha 21
522   imprime_grub(4 , direita , false, false);
523   imprime_grub(2 , direita , true , false);
```

```
524   imprime_grub(9 , direita , false, false);
525   imprime_grub(2 , direita , true , false);
526   imprime_grub(7 , direita , false, false);
527   imprime_grub(3 , direita , true , false);
528   imprime_grub(9 , direita , false, false);
529   imprime_grub(2 , direita , false, true );
530   imprime_grub(40, esquerda, true , true ); //linha 22
531   descarrega_papel();
532 }
```

APÊNDICE C: Custo do projeto

Quantidade	Descrição	Preço unitário (R$)	Preço total (R$)
1	Botão de contato	2,30	2,30
1	Buzzer	3,55	3,55
1	Capacitor cerâmico de 100nF	0,30	0,30
2	Capacitor cerâmico de 10pF	0,25	0,50
2	Capacitor cerâmico de 22pF	0,25	0,50
2	Capacitor eletrolítico de 10mF/25V	1,50	3,00
1	Capacitor eletrolítico de 470mF/50V	2,50	2,50
2	Capacitores eletrolíticos de 1mF/50V	1,00	2,00
1	Chave ótica	4,20	4,20
1	Circuito integrado L293D	7,20	7,20
1	Circuito integrado MAX232	9,60	9,60
1	Circuito integrado ULN2003A	2,48	2,48
1	Diodo 1N4001	0,30	0,30
1	Diveros (solda, fios, fita adesiva, jumper)	30,00	30,00
1	EVA	0,50	0,50
1	Led	1,40	1,40
1	LM 7805	1,86	1,86
1	LM 7812	1,86	1,86
1	MicroICD	99,00	99,00
1	Cristal de 4 MHz	2,10	2,10
1	PIC 16F877A	35,50	35,50
1	Placa de ilha de 9x20	12,50	12,50
4	Resistor 10K Ω	0,30	1,20
1	Resistor 270 Ω	0,30	0,30
1	Resistor 330Ω	0,30	0,30
1	Servo-motor Hobbico HCAM0110	86,00	86,00
1	Solenóide 120-420-620-540 6V DC	106,00	106,00
3	Soquete de 16 furos	3,08	9,24
1	Soquete de 40 furos	6,80	6,80
		TOTAL	**432,99**

ANEXO A: Orçamento de impressoras Braille

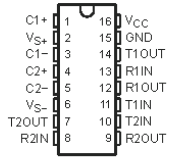
Orçado em 21 de novembro de 2006.

Item	Descrição	Preço unitário
01	IMPRESSORA BASIC S: Impressora Braille, em formulário contínuo, impressão de um lado da folha (single side), 170 pph (páginas por hora), 49 cps (caracteres por segundo). Impressão em alto relevo - mesmo com papeis de baixa gramatura; capacidade para imprimir textos (máximo 42 caracteres por linha) e gráficos; sistema Braille de 6 ou 8 pontos; permite ajuste para papeis com diferentes tamanhos e gramaturas; oferece sistema de memória para ate 400 cópias; painel de controle de fácil operação tanto para pessoas videntes como para os cegos; auxílio sonoro dos comandos por voz (opcional nas línguas portuguesa ou inglesa); dimensões 521x246x120 mm, peso 8 kg; acompanha software gerenciador de impressão, drive para impressão Braille de dentro do Word e aplicativo “Draw” para impressão de desenhos.	R$ 17.500,00
02	IMPRESSORA BASIC D: Impressora Braille, para folha contínua frente e verso (double side, impressão dos dois lados da folha). Permite não só 50% de redução na quantidade de papel, bem como, nos custos, reduzindo tempo de trabalho para encadernação, manuseio e espaço de armazenamento. Velocidade - 340 pph (páginas por hora), 99 cps (caracteres por segundo); impressão em alto relevo - mesmo com papeis de baixa gramatura; capacidade para imprimir textos (máximo 42 caracteres por linha) e gráficos; sistema Braille interpontado de 6 ou 8 pontos; permite ajuste para papeis com diferentes tamanhos e gramaturas; oferece sistema de memória para ate 400 cópias; painel de controle de fácil operação tanto para pessoas videntes como para os cegos; auxílio sonoro dos comandos por voz (opcional nas línguas portuguesa ou inglesa); possui - sistema de multicópia independente do micro; dimensões 521x246x120mm, peso 8kg; acompanha software gerenciador de impressão, drive para impressão Braille de dentro do Word e aplicativo “Draw” para impressão de desenhos.	R$ 19.000,00
03	IMPRESSORA 4X4 PRO: Impressão frente e verso (double side, dois lados da folha) em folhas soltas. Formato brochura. Com este novo modelo desenvolvido pela index, ficou mais fácil produzir automaticamente revistas (brochuras), livros e jornais em Braille, fazendo com que até produções de baixo volume sejam feitas com baixo custo (baixo custo é mesmo a palavra de ordem, manutenções só devem ser feitas após 500.000 cópias impressas). A impressora Braille index 4x4 pro é a ferramenta perfeita para produzir: livros escolares, material de marketing, relatórios e informações públicas. Simplesmente, prepare um arquivo, imprima e grampeie, está pronto, não se preocupe, a 4x4 pro já fez toda a formatação para você, velocidade de impressão, 380 pph (tamanho A4), 100 cps (caracteres por segundo), nível de ruído abaixo de 60 db(a) - a index 4x4 PRO oferece alternativas de impressão em papeis de formato A4 até A3, com opção de 1, 2 ou 4 páginas ao mesmo tempo (sempre no formato paisagem); permite operar com papeis de gramatura (90-180gsm^2); comprimento (297-490mm), largura (150-297mm); formatações possíveis, 18 a 39 caracteres por linha e 15 a 29 linhas por página; sistema Braille interpontado de 6 ou 8 pontos; permite impressão de texto e gráficos; oferece sistema de memória para ate 200 páginas Braille; é construída com painel de controle de fácil operação (chaves de membrana) com caracteres em texto e Braille para perfeita identificação dos comandos, tanto para pessoas videntes como para os cegos; auxílio sonoro dos comandos por voz (disponível nas diversas línguas inglesa-default); possui sistema de multicópia independente do micro, 10 tabelas Braille pré-instaladas e 10 livres para definição pelo usuário, interfaces - paralelo (centronics) e serial; alimentação 100-240vac - 50/60 hz, consumo 10w stand-by e máximo de 130w; dimensões do conjunto montado com gabinete acústico hxlxprof. 1520x750x160mm, peso 57kg; acompanha software gerenciador de impressão, drive para impressão Braille de dentro do Word e aplicativo “Draw” para impressão de desenhos.	R$ 29.000,00
04	IMPRESSORA EVEREST D: Impressora Braille, para folha solta, impressão frente e verso (double side, impressão dos dois lados da folha ao mesmo tempo), 324 pph (páginas por hora) 95 cps (caracteres por segundo modo double side). Impressão em alto relevo - mesmo com papeis de baixa gramatura; capacidade para imprimir textos (máximo 42 caracteres por linha) e gráficos; facilidade de ajuste de espaçamento entre pontos, celas e linhas, permite impressão; Braille para leitores com diferentes níveis de aprendizado; sistema Braille interpontado de 6 ou 8 pontos; permite ajuste para papeis com diferentes tamanhos e gramaturas; oferece sistema de memória para ate 400 cópias; painel de controle de fácil operação com caracteres em texto e Braille para perfeita identificação dos comandos, tanto para pessoas videntes como para os cegos; auxílio sonoro dos comandos por voz (opcional nas línguas portuguesa ou inglesa); possui sistema de multicópia independente do micro, 10 tabelas Braille pré-instaladas e 10 livres para definição pelo usuário, interfaces - paralelo (centronics) e serial; dimensões 563x175x433mm, peso 14kg; acompanha software gerenciador de impressão, drive para impressão Braille de dentro do Word e aplicativo “Draw” para impressão de desenhos.	R$ 25.000,00

Fonte: Civiam Comércio Importação Exportação Ltda (http://www.civiam.com.br)

- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0-μF Charge-Pump Capacitors
- Operates Up To 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- ESD Protection Exceeds JESD 22
  – 2000-V Human-Body Model (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1-μF Charge-Pump Capacitors is Available With the MAX202
- Applications
  – TIA/EIA-232-F, Battery-Powered Systems, Terminals, Modems, and Computers

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/EIA-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/EIA-232-F inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/EIA-232-F levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube of 25	MAX232N	MAX232N
	SOIC (D)	Tube of 40	MAX232D	MAX232
		Reel of 2500	MAX232DR	
	SOIC (DW)	Tube of 40	MAX232DW	MAX232
		Reel of 2000	MAX232DWR	
	SOP (NS)	Reel of 2000	MAX232NSR	MAX232
–40°C to 85°C	PDIP (N)	Tube of 25	MAX232IN	MAX232IN
	SOIC (D)	Tube of 40	MAX232ID	MAX232I
		Reel of 2500	MAX232IDR	
	SOIC (DW)	Tube of 40	MAX232IDW	MAX232I
		Reel of 2000	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1) . . . –0.3 V to 6 V
Positive output supply voltage range, V_{S+} . . . V_{CC} – 0.3 V to 15 V
Negative output supply voltage range, $V_{S–}$ . . . –0.3 V to –15 V
Input voltage range, V_I: Driver . . . –0.3 V to V_{CC} +0.3 V
Receiver . . . ±30 V
Output voltage range, V_O: T1OUT, T2OUT . . . $V_{S–}$ – 0.3 V to V_{S+} +0.3 V
R1OUT, R2OUT . . . –0.3 V to V_{CC} +0.3 V
Short-circuit duration: T1OUT, T2OUT . . . Unlimited
Package thermal impedance, θ_{JA} (see Notes 2 and 3): D package . . . 73°C/W
DW package . . . 57°C/W
N package . . . 67°C/W
NS package . . . 64°C/W
Operating virtual junction temperature, T_J . . . 150°C
Storage temperature range, T_{stg} . . . –65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltages are with respect to network GND.
2. Maximum power dissipation is a function of $T_J(max)$, θ_{JA}, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) – T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability.
3. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions

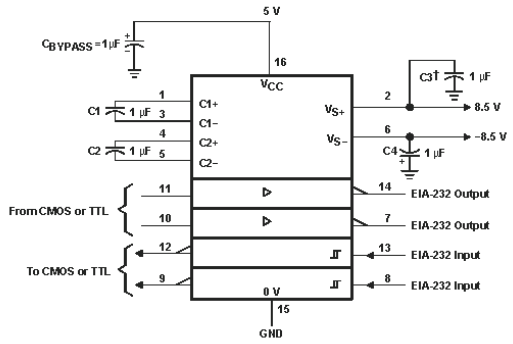
			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN,T2IN)		2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)				0.8	V
R1IN, R2IN	Receiver input voltage				±30	V
T_A	Operating free-air temperature	MAX232	0		70	°C
		MAX232I	–40		85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature (unless otherwise noted) (see Note 4 and Figure 4)

PARAMETER		TEST CONDITIONS	MIN	TYP‡	MAX	UNIT
I_{CC}	Supply current	V_{CC} = 5.5 V, All outputs open, T_A = 25°C		8	10	mA

‡ All typical values are at V_{CC} = 5 V and T_A = 25°C.
NOTE 4: Test conditions are C1–C4 = 1 μF at V_{CC} = 5 V ± 0.5 V.

APPLICATION INFORMATION



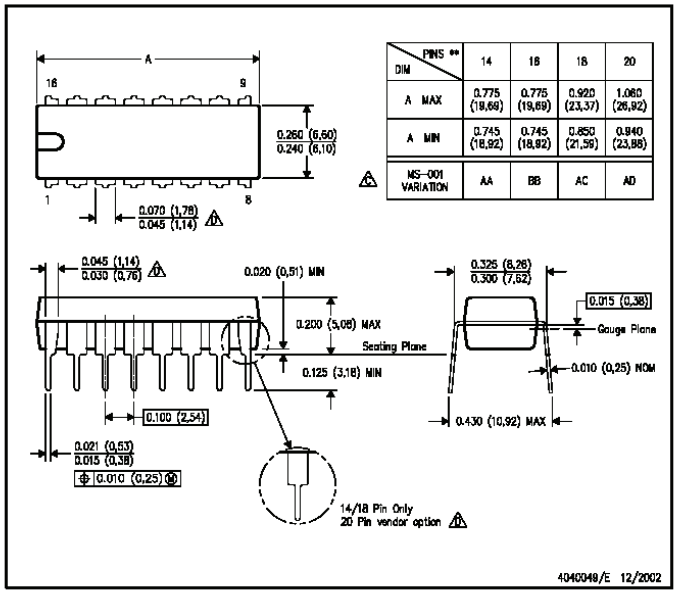
† C3 can be connected to V_{CC} or GND.

NOTES: A. Resistor values shown are nominal.
B. Nonpolarized ceramic capacitors are acceptable. If polarized tantalum or electrolytic capacitors are used, they should be connected as shown. In addition to the 1-µF capacitors shown, the MAX202 can operate with 0.1-µF capacitors.

Figure 4. Typical Operating Circuit

N (R–PDIP–T**) PLASTIC DUAL–IN–LINE PACKAGE

16 PINS SHOWN



DIM \ PINS **	14	16	18	20
A MAX	0.775 (19,69)	0.775 (19,69)	0.920 (23,37)	1.060 (26,92)
A MIN	0.745 (18,92)	0.745 (18,92)	0.850 (21,59)	0.940 (23,88)
MS–001 VARIATION	AA	BB	AC	AD

NOTES: A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS–001, except 18 and 20 pin minimum body length (Dim A).
D. The 20 pin end lead shoulder width is a vendor option, either half or full width.

ANEXO C: Principais páginas do *datasheet* – Motor DC (#HC385XLG)

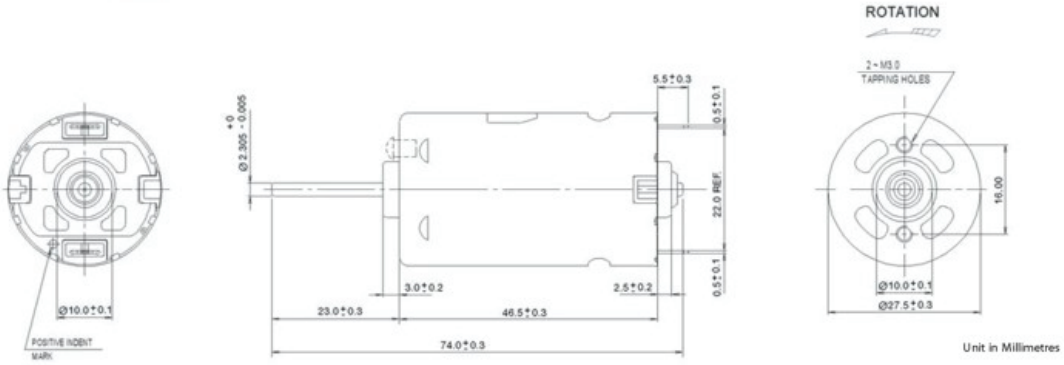
JOHNSON ELECTRIC

Product Reference Guide

Motor for Foot Bath (Water Jet)

P.M.D.C. Motor

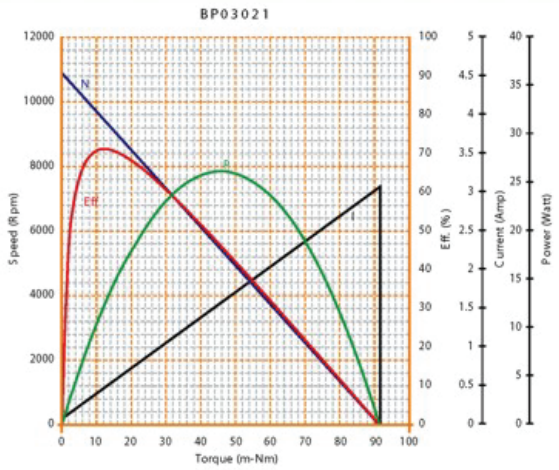
BP03021: High power with compact size



Unit in Millimetres

Performance (testing under room temperature) :

Motor Series	Motor Code	Nom. Volt. (V)	Electrical Performance at													Motor Characteristics		
			No Load		Stall		Maximum Efficiency					Maximum Power						
			Speed (Rpm)	Cur. (Amp)	Tor. (mNm)	Cur. (Amp)	Eff. (%)	Tor. (mNm)	Speed (Rpm)	Cur. (Amp)	Power (W)	Tor. (mNm)	Speed (Rpm)	Cur. (Amp)	Power (W)	Kt (mNm/A)	R (Ohms)	Reg. (Rpm/mNm)
HC385XLG	BP03021	36.0	10900	0.073	91.56	3.066	71.31	12.24	9500	0.473	12.146	45.78	5500	1.570	26.221	30.589	11.741	119.422



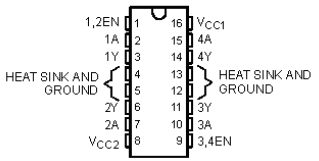
Remarks:
1. Specifications here are for reference only and subject to change without prior notice.
2. Having full capability in motor design, development and manufacture, Johnson Electric is capable to tailor-design motors according to customer's requirement.

Johnson Electric Group
6-22 Dai Shun Street, Tai Po Industrial Estate, Tai Po, New Territories, Hong Kong
Tel : (852) 2663 6688 Fax : (852) 2897 2054 Web Site : www.johnsonelectric.com Email : salessupport@johnsonelectric.com

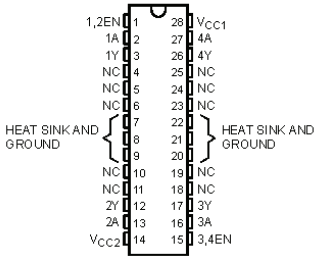
Fonte: http://www.eminebea.com/usa/eMinebea
Acesso em 6 de novembro de 2008

- **Featuring Unitrode L293 and L293D Products Now From Texas Instruments**
- **Wide Supply-Voltage Range: 4.5 V to 36 V**
- **Separate Input-Logic Supply**
- **Internal ESD Protection**
- **Thermal Shutdown**
- **High-Noise-Immunity Inputs**
- **Functional Replacements for SGS L293 and SGS L293D**
- **Output Current 1 A Per Channel (600 mA for L293D)**
- **Peak Output Current 2 A Per Channel (1.2 A for L293D)**
- **Output Clamp Diodes for Inductive Transient Suppression (L293D)**

N, NE PACKAGE
(TOP VIEW)



DWP PACKAGE
(TOP VIEW)



description

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

On the L293, external high-speed output clamp diodes should be used for inductive transient suppression.

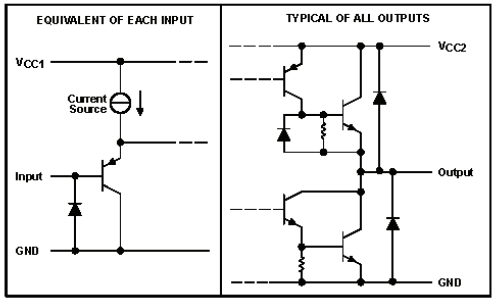
A V_{CC1} terminal, separate from V_{CC2}, is provided for the logic inputs to minimize device power dissipation.

The L293 and L293D are characterized for operation from 0°C to 70°C.

L293, L293D
QUADRUPLE HALF-H DRIVERS

SLRS008B – SEPTEMBER 1986 – REVISED JUNE 2002

schematics of inputs and outputs (L293D)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage, V_{CC1} (see Note 1)	36 V
Output supply voltage, V_{CC2}	36 V
Input voltage, V_I	7 V
Output voltage range, V_O	–3 V to V_{CC2} + 3 V
Peak output current, I_O (nonrepetitive, t ≤ 5 ms): L293	±2 A
Peak output current, I_O (nonrepetitive, t ≤ 100 µs): L293D	±1.2 A
Continuous output current, I_O: L293	±1 A
Continuous output current, I_O: L293D	±600 mA
Continuous total dissipation at (or below) 25°C free-air temperature (see Notes 2 and 3)	2075 mW
Continuous total dissipation at 80°C case temperature (see Note 3)	5000 mW
Maximum junction temperature, T_J	150°C
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	260°C
Storage temperature range, T_{stg}	–65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. All voltage values are with respect to the network ground terminal.
2. For operation above 25°C free-air temperature, derate linearly at the rate of 16.6 mW/°C.
3. For operation above 25°C case temperature, derate linearly at the rate of 71.4 mW/°C. Due to variations in individual device electrical characteristics and thermal resistance, the built-in thermal overload protection may be activated at power levels slightly above or below the rated dissipation.

APPLICATION INFORMATION

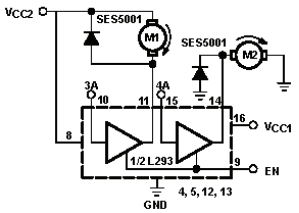


Figure 4. DC Motor Controls (connections to ground and to supply voltage)

EN	3A	M1	4A	M2
H	H	Fast motor stop	H	Run
H	L	Run	L	Fast motor stop
L	X	Free-running motor stop	X	Free-running motor stop

L = low, H = high, X = don't care

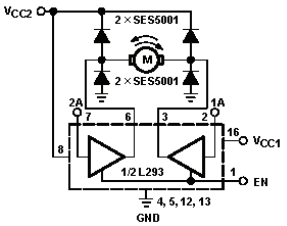


Figure 5. Bidirectional DC Motor Control

EN	1A	2A	FUNCTION
H	L	H	Turn right
H	H	L	Turn left
H	L	L	Fast motor stop
H	H	H	Fast motor stop
L	X	X	Fast motor stop

L = low, H = high, X = don't care

APPLICATION INFORMATION

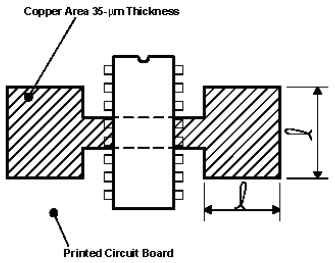


Figure 7. Example of Printed Circuit Board Copper Area (used as heat sink)

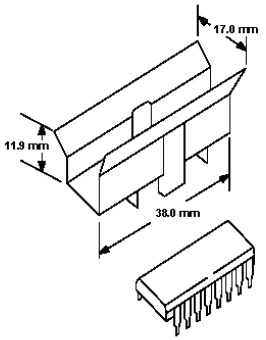


Figure 8. External Heat Sink Mounting Example (θ_{JA} = 25°C/W)

ANEXO E: Principais páginas do *datasheet* – Motor de passo (#PM55L-048)

NMB-MAT
Minebea Motor Manufacturing Corporation
eMINEBEA.COM

PM55L-048

PM Motor
PM Type

Reference Characteristics

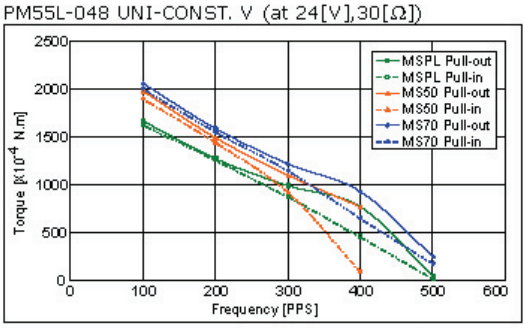
Motor Size	PM55L-048	
Number of Steps per Rotation	48(7.5°/Step)	
Drive Method	2-2 PHASE	
Drive Circuit	UNIPOLAR CONST. VOLT.	BIPOLAR CHOPPER
Drive Voltage	24[V]	24[V]
Current/Phase		800[mA]
Coil Resistance/Phase	30[Ω]	5.5[Ω]
Drive IC	2SC3346	UDN2916B-V
Magnet Material	Ferrite plastic magnet (MSPL) Polar anisotropy ferrite sintered magnet (MS50) Nd-Fe-B bonded magnet (MS70)	
Insulation Resistance	100M[Ω] MIN	
Dielectric Strength	AC 500[V] 1[min]	
Class of Insulation	CLASS E	
Operating Temp.	-10[°C] ~ 50[°C]	
Storage Temp.	-30[°C] ~ 80[°C]	
Operating Hum.	20[%] RH ~ 90[%] RH	

Applications

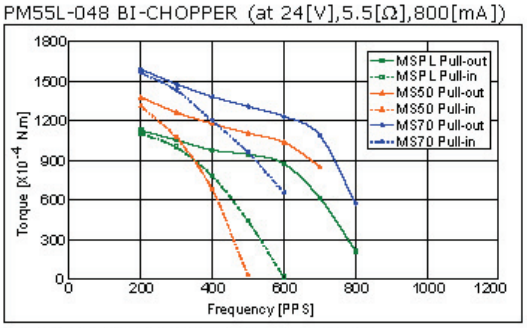
OA Equipment : Printers / Scanners
Industrial equipment : Flow control valves
Toys : Slot machines
Home automation appliances : Sewing machines

Torque Characteristics

PM55L-048 UNI-CONST. V (at 24[V],30[Ω])

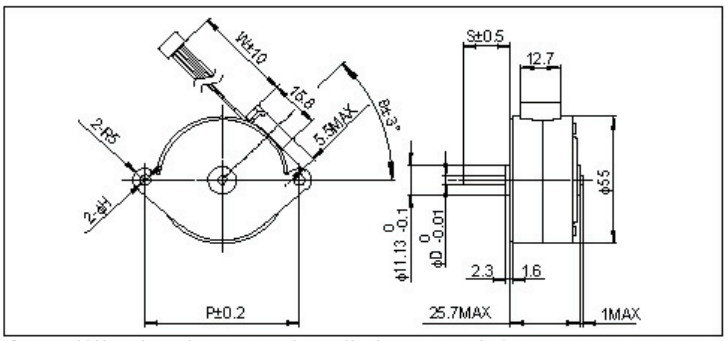


PM55L-048 BI-CHOPPER (at 24[V],5.5[Ω],800[mA])



These torque values are reference only. Heat radiation conditions and temperature rise effect by duty are different on each equipment, therefore please select motors after considering the heat conditions in the actual equipment.

Dimensions



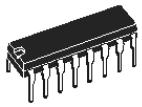
If you would like to know this Dimensions(D,S,W,θ), Please see Standerd Dimensions in our Home Page.

Fonte: http://www.eminebea.com/usa/eMinebea
Acesso em 6 de novembro de 2008

ULN2001A-ULN2002A
ULN2003A-ULN2004A

SEVEN DARLINGTON ARRAYS

- SEVEN DARLINGTONS PER PACKAGE
- OUTPUT CURRENT 500mA PER DRIVER (600mA PEAK)
- OUTPUT VOLTAGE 50V
- INTEGRATED SUPPRESSION DIODES FOR INDUCTIVE LOADS
- OUTPUTS CAN BE PARALLELED FOR HIGHER CURRENT
- TTL/CMOS/PMOS/DTL COMPATIBLE INPUTS
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY LAYOUT



DIP16

ORDERING NUMBERS: ULN2001A/2A/3A/4A

SO16

ORDERING NUMBERS: ULN2001D/2D/3D/4D

DESCRIPTION

The ULN2001A, ULN2002A, ULN2003 and ULN2004A are high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

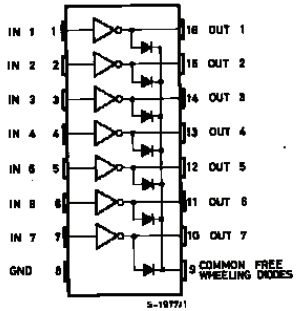
The four versions interface to all common logic families :

ULN2001A	General Purpose, DTL, TTL, PMOS, CMOS
ULN2002A	14-25V PMOS
ULN2003A	5V TTL, CMOS
ULN2004A	6–15V CMOS, PMOS

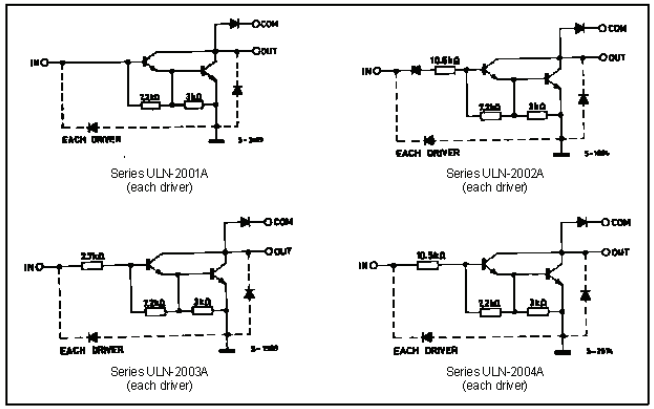
These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high power buffers.

The ULN2001A/2002A/2003A and 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D/2002D/2003D/2004D.

PIN CONNECTION



SCHEMATIC DIAGRAM



Series ULN-2001A (each driver)

Series ULN-2002A (each driver)

Series ULN-2003A (each driver)

Series ULN-2004A (each driver)

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_o	Output Voltage	50	V
V_{in}	Input Voltage (for ULN2002A/D - 2003A/D - 2004A/D)	30	V
I_c	Continuous Collector Current	500	mA
I_b	Continuous Base Current	25	mA
T_{amb}	Operating Ambient Temperature Range	– 20 to 85	°C
T_{stg}	Storage Temperature Range	– 55 to 150	°C
T_j	Junction Temperature	150	°C

THERMAL DATA

Symbol	Parameter		DIP16	SO16	Unit
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	70	120	°C/W

ELECTRICAL CHARACTERISTICS (T_{amb} = 25°C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit	Fig.
I_{CEX}	Output Leakage Current	V_{CE} = 50V T_{amb} = 70°C, V_{CE} = 50V T_{amb} = 70°C for ULN2002A V_{CE} = 50V, V_I = 6V for ULN2004A V_{CE} = 50V, V_I = 1V			50 100 500 500	µA µA µA µA	1a 1a 1b 1b
$V_{CE(sat)}$	Collector-emitter Saturation Voltage	I_C = 100mA, I_B = 250µA I_C = 200 mA, I_B = 350µA I_C = 350mA, I_B = 500µA		0.9 1.1 1.3	1.1 1.3 1.6	V V V	2 2 2
$I_{I(on)}$	Input Current	for ULN2002A, V_I = 17V for ULN2003A, V_I = 3.85V for ULN2004A, V_I = 5V V_I = 12V		0.82 0.93 0.35 1	1.25 1.35 0.5 1.45	mA mA mA mA	3 3 3 3
$I_{I(off)}$	Input Current	T_{amb} = 70°C, I_C = 500µA	50	65		µA	4
$V_{I(on)}$	Input Voltage	V_{CE} = 2V for ULN2002A I_C = 300mA for ULN2003A I_C = 200mA I_C = 250mA I_C = 300mA for ULN2004A I_C = 125mA I_C = 200mA I_C = 275mA I_C = 350mA			 13 2.4 2.7 3 5 6 7 8	V	5
h_{FE}	DC Forward Current Gain	for ULN2001A V_{CE} = 2V, I_C = 350mA	1000				2
C_I	Input Capacitance			15	25	pF	
t_{PLH}	Turn-on Delay Time	0.5 V_I to 0.5 V_o		0.25	1	µs	
t_{PHL}	Turn-off Delay Time	0.5 V_I to 0.5 V_o		0.25	1	µs	
I_R	Clamp Diode Leakage Current	V_R = 50V T_{amb} = 70°C, V_R = 50V			50 100	µA µA	6 6
V_F	Clamp Diode Forward Voltage	I_F = 350mA		1.7	2	V	7

Figure 8: Collector Current versus Input Current

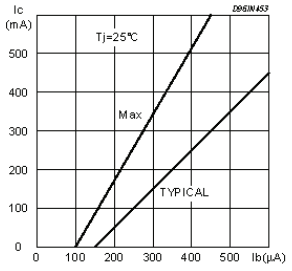


Figure 9: Collector Current versus Saturation Voltage

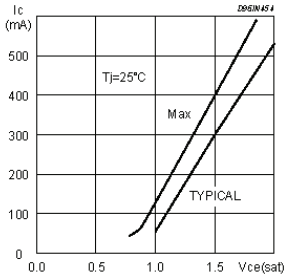


Figure 10: Peak Collector Current versus Duty Cycle

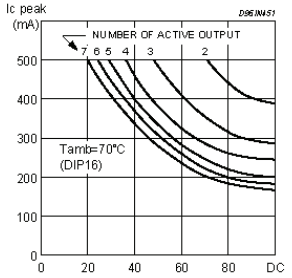
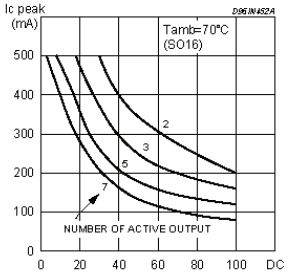
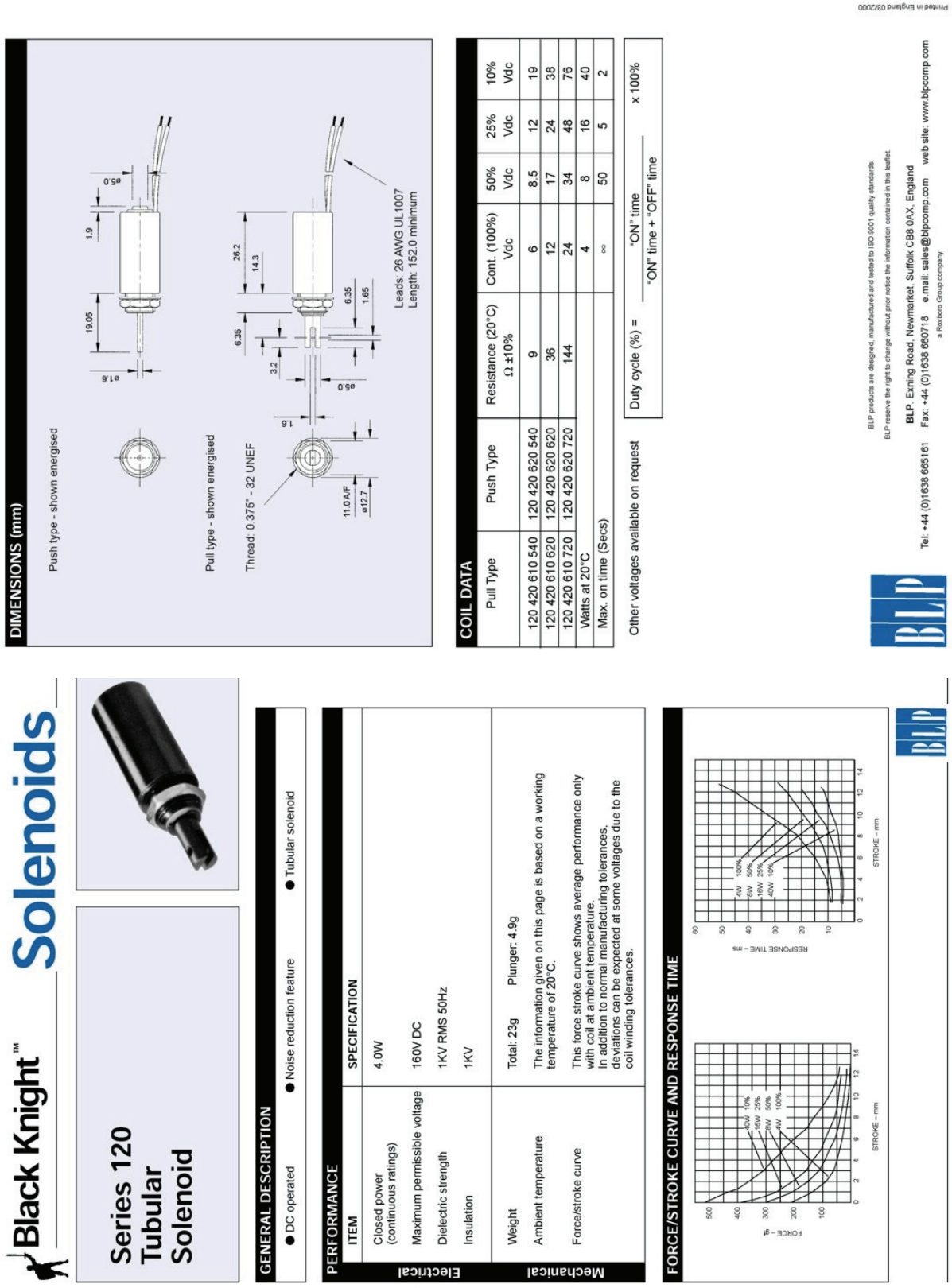


Figure 11: Peak Collector Current versus Duty Cycle

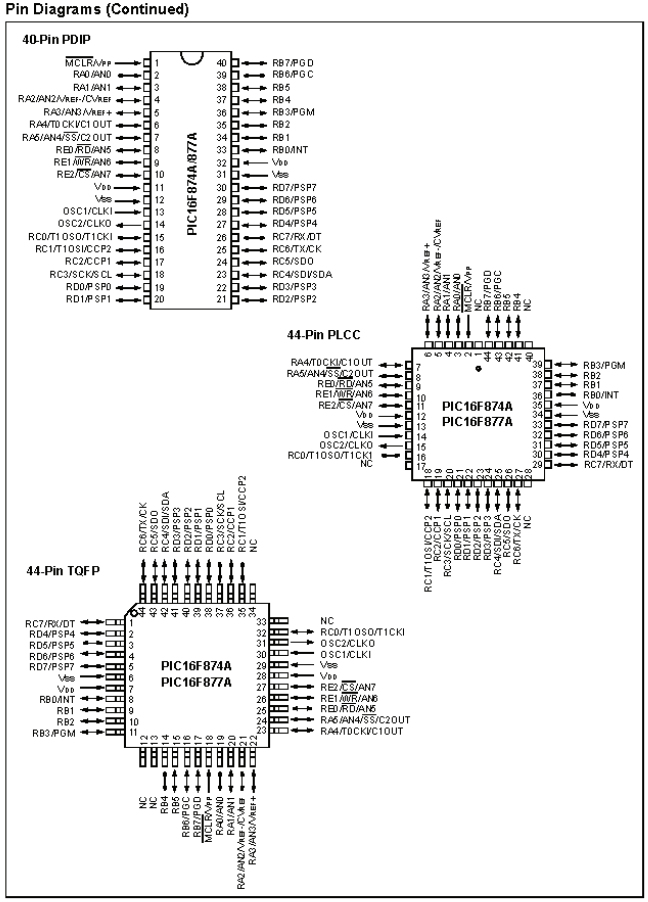


ANEXO G: Principais páginas do *datasheet* – Solenóide (#120420620540)



Fonte: http://www.farnell.com/datasheets/103300.pdf
Acesso em 6 de novembro de 2008

Pin Diagrams (Continued)



1.0 DEVICE OVERVIEW

This document contains device specific information about the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

PIC16F873A/876A devices are available only in 28-pin packages, while PIC16F874A/877A devices are available in 40-pin and 44-pin packages. All devices in the PIC16F87XA family share common architecture with the following differences:

- The PIC16F873A and PIC16F874A have one-half of the total on-chip memory of the PIC16F876A and PIC16F877A
- The 28-pin devices have three I/O ports, while the 40/44-pin devices have five
- The 28-pin devices have fourteen interrupts, while the 40/44-pin devices have fifteen
- The 28-pin devices have five A/D input channels, while the 40/44-pin devices have eight
- The Parallel Slave Port is implemented only on the 40/44-pin devices

The available features are summarized in Table 1-1. Block diagrams of the PIC16F873A/876A and PIC16F874A/877A devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2 and Table 1-3.

Additional information may be found in the PICmicro® Mid-Range Reference Manual (DS33023), which may be obtained from your local Microchip Sales Representative or downloaded from the Microchip web site. The Reference Manual should be considered a complementary document to this data sheet and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

TABLE 1-1: PIC16F87XA DEVICE FEATURES

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
01h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	43, 150
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	47, 150
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	48, 150
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	49, 150
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
0Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	26, 150
0Dh	PIR2	—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF	-0-0 0--0	28, 150
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	60, 150
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	--00 0000	57, 150
11h	TMR2	Timer2 Module Register								0000 0000	62, 150
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	61, 150
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	79, 150
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	82, 82, 150
15h	CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	63, 150
16h	CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	63, 150
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	64, 150
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	112, 150
19h	TXREG	USART Transmit Data Register								0000 0000	118, 150
1Ah	RCREG	USART Receive Data Register								0000 0000	118, 150
1Bh	CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	63, 150
1Ch	CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	63, 150
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	64, 150
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	133, 150
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	$\overline{GO/DONE}$	—	ADON	0000 00-0	127, 150

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC <12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	43, 150
86h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
87h	TRISC	PORTC Data Direction Register								1111 1111	47, 150
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	48, 151
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	50, 151
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
8Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	25, 151
8Dh	PIE2	—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE	-0-0 0--0	27, 151
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	29, 151
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	83, 151
92h	PR2	Timer2 Period Register								1111 1111	62, 151
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	79, 151
94h	SSPSTAT	SMP	CKE	$D/\overline{A}$	P	S	$R/\overline{W}$	UA	BF	0000 0000	79, 151
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	111, 151
99h	SPBRG	Baud Rate Generator Register								0000 0000	113, 151
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	135, 151
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	141, 151
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	133, 151
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	128, 151

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC <12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 2											
100h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
101h	TMR0	Timer0 Module Register								xxxx xxxx	55, 150
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	30, 150
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	45, 150
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
10Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	39, 151
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	39, 151
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						--xx xxxx	39, 151
10Fh	EEADRH	—	—	—	—(5)	EEPROM Address Register High Byte				---- xxxx	39, 151
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	31, 150
181h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	23, 150
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	30, 150
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	22, 150
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	31, 150
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	45, 150
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	30, 150
18Bh(3)	INTCON	GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	24, 150
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	34, 151
18Dh	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	39, 151
18Eh	—	Reserved; maintain clear								0000 0000	—
18Fh	—	Reserved; maintain clear								0000 0000	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8>, whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873A/876A devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD and TRISE are not implemented on PIC16F873A/876A devices, read as '0'.

5: Bit 4 of EEADRH implemented only on the PIC16F876A/877A devices.

4.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual (DS33023).

4.1 PORTA and the TRISA Register

PORTA is a 6-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, the value is modified and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open-drain output. All other PORTA pins have TTL input levels and full CMOS output drivers.

Other PORTA pins are multiplexed with analog inputs and the analog VREF input for both the A/D converters and the comparators. The operation of each pin is selected by clearing/setting the appropriate control bits in the ADCON1 and/or CMCON registers.

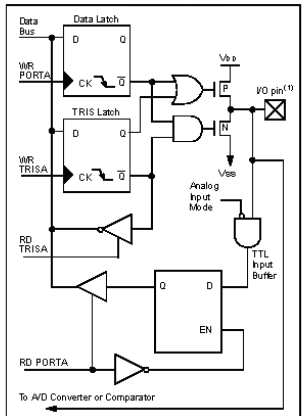
Note: On a Power-on Reset, these pins are configured as analog inputs and read as '0'. The comparators are in the off (digital) state.

The TRISA register controls the direction of the port pins even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 4-1: INITIALIZING PORTA

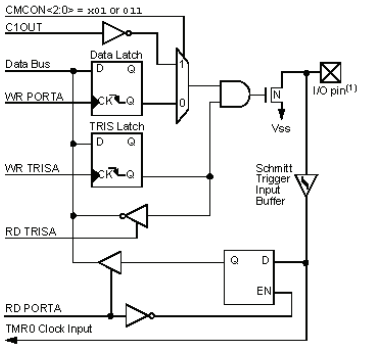
```
BCF     STATUS, RP0     ;
BCF     STATUS, RP1     ; Bank0
CLRF    PORTA           ; Initialize PORTA by
                        ; clearing output
                        ; data latches
BSF     STATUS, RP0     ; Select Bank 1
MOVLW   0x06            ; Configure all pins
MOVWF   ADCON1          ; as digital inputs
MOVLW   0xCF            ; Value used to
                        ; initialize data
                        ; direction
MOVWF   TRISA           ; Set RA<3:0> as inputs
                        ; RA<5:4> as outputs
                        ; TRISA<7:6>are always
                        ; read as '0'.
```

FIGURE 4-1: BLOCK DIAGRAM OF RA3:RA0 PINS



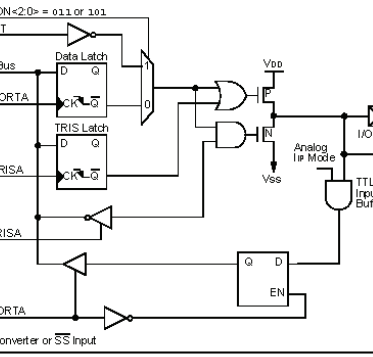
Note 1: I/O pins have protection diodes to Vdd and Vss.

FIGURE 4-2: BLOCK DIAGRAM OF RA4/T0CKI PIN



Note 1: I/O pin has protection diodes to Vss only.

FIGURE 4-3: BLOCK DIAGRAM OF RA5 PIN



Note 1: I/O pin has protection diodes to VDD and Vss.

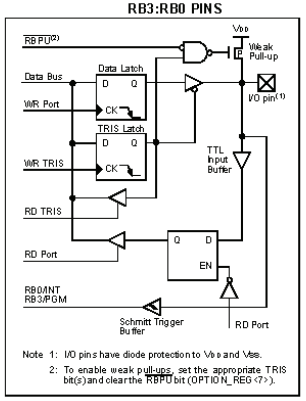
4.2 PORTB and the TRISB Register

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Three pins of PORTB are multiplexed with the In-Circuit Debugger and Low-Voltage Programming function: RB3/PGM, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in **Section 14.0 "Special Features of the CPU"**.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit RBPU (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

FIGURE 4-4: BLOCK DIAGRAM OF RB3:RB0 PINS



Note 1: I/O pins have diode protection to VDD and Vss.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (OPTION_REG<7>).

Four of the PORTB pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB port change interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from Sleep. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.
b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

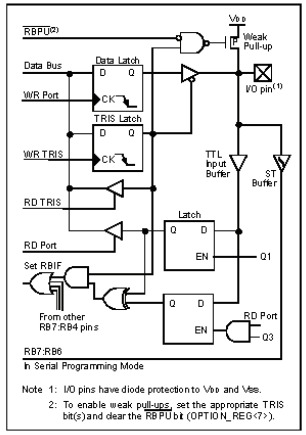
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

This interrupt-on-mismatch feature, together with software configurable pull-ups on these four pins, allow easy interface to a keypad and make it possible for wake-up on key depression. Refer to the application note, *AN552, "Implementing Wake-up on Key Stroke"* (DS00552).

RB0/INT is an external interrupt input pin and is configured using the INTEDG bit (OPTION_REG<6>).

RB0/INT is discussed in detail in **Section 14.11.1 "INT Interrupt"**.

FIGURE 4-5: BLOCK DIAGRAM OF RB7:RB4 PINS



Note 1: I/O pins have diode protection to VDD and Vss.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (OPTION_REG<7>).

4.3 PORTC and the TRISC Register

PORTC is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a High-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

PORTC is multiplexed with several peripheral functions (Table 4-5). PORTC pins have Schmitt Trigger input buffers.

When the I^2C module is enabled, the PORTC<4:3> pins can be configured with normal I^2C levels, or with SMBus levels, by using the CKE bit (SSPSTAT<6>).

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (BSF, BCF, XORWF) with TRISC as the destination, should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

FIGURE 4-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<2:0>, RC<7:5>

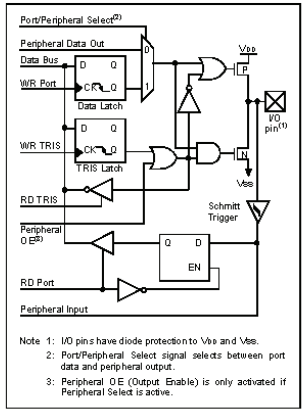
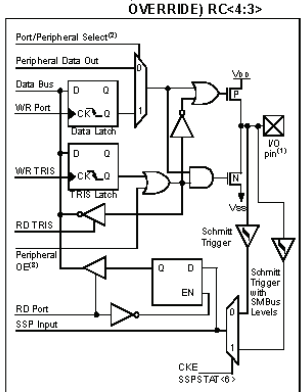


FIGURE 4-7: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<4:3>



4.4 PORTD and TRISD Registers

Note: PORTD and TRISD are not implemented on the 28-pin devices.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

PORTD can be configured as an 8-bit wide microprocessor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

FIGURE 4-8: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)

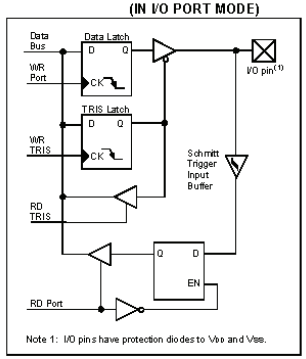


TABLE 4-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit 0	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 0.
RD1/PSP1	bit 1	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 2.
RD3/PSP3	bit 3	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 3.
RD4/PSP4	bit 4	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 4.
RD5/PSP5	bit 5	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 5.
RD6/PSP6	bit 6	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 6.
RD7/PSP7	bit 7	ST/TTL(1)	Input/output port pin or Parallel Slave Port bit 7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 4-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

4.5 PORTE and TRISE Register

Note: PORTE and TRISE are not implemented on the 28-pin devices.

PORTE has three pins (RE0/$\overline{RD}$/AN5, RE1/$\overline{WR}$/AN6 and RE2/$\overline{CS}$/AN7) which are individually configurable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become the I/O control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make certain that the TRISE<2:0> bits are set and that the pins are configured as digital inputs. Also, ensure that ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

Register 4-1 shows the TRISE register which also controls the Parallel Slave Port operation.

PORTE pins are multiplexed with analog inputs. When selected for analog input, these pins will read as '0's.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

Note: On a Power-on Reset, these pins are configured as analog inputs and read as '0'.

FIGURE 4-9: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)

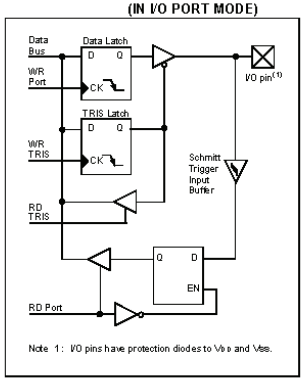


TABLE 4-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit 0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected).
RE1/$\overline{WR}$/AN6	bit 1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected).
RE2/$\overline{CS}$/AN7	bit 2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input
Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

8.3 PWM Mode (PWM)

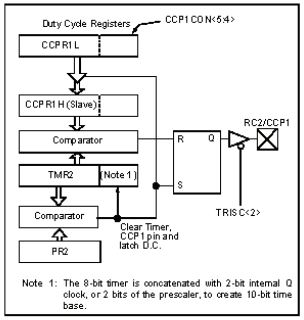
In Pulse Width Modulation mode, the CCPx pin produces up to a 10-bit resolution PWM output. Since the CCP1 pin is multiplexed with the PORTC data latch, the TRISC<2> bit must be cleared to make the CCP1 pin an output.

Note: Clearing the CCP1CON register will force the CCP1 PWM output latch to the default low level. This is not the PORTC I/O data latch.

Figure 8-3 shows a simplified block diagram of the CCP module in PWM mode.

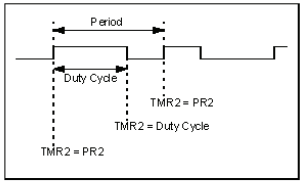
For a step-by-step procedure on how to set up the CCP module for PWM operation, see **Section 8.3.3 "Setup for PWM Operation"**.

FIGURE 8-3: SIMPLIFIED PWM BLOCK DIAGRAM



A PWM output (Figure 8-4) has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 8-4: PWM OUTPUT



8.3.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula:

$$\text{PWM Period} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note: The Timer2 postscaler (see **Section 7.1 "Timer2 Prescaler and Postscaler"**) is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

8.3.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR1H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitch-free PWM operation.

When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCP1 pin is cleared.

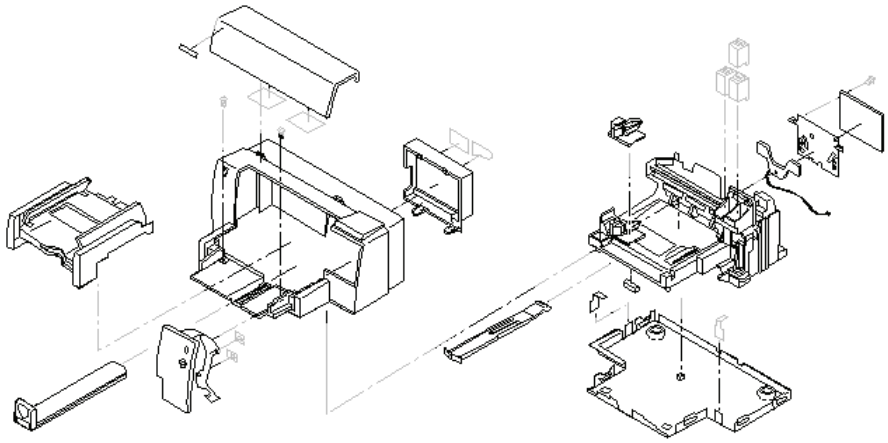
The maximum PWM resolution (bits) for a given PWM frequency is given by the following formula.

EQUATION 8-1:

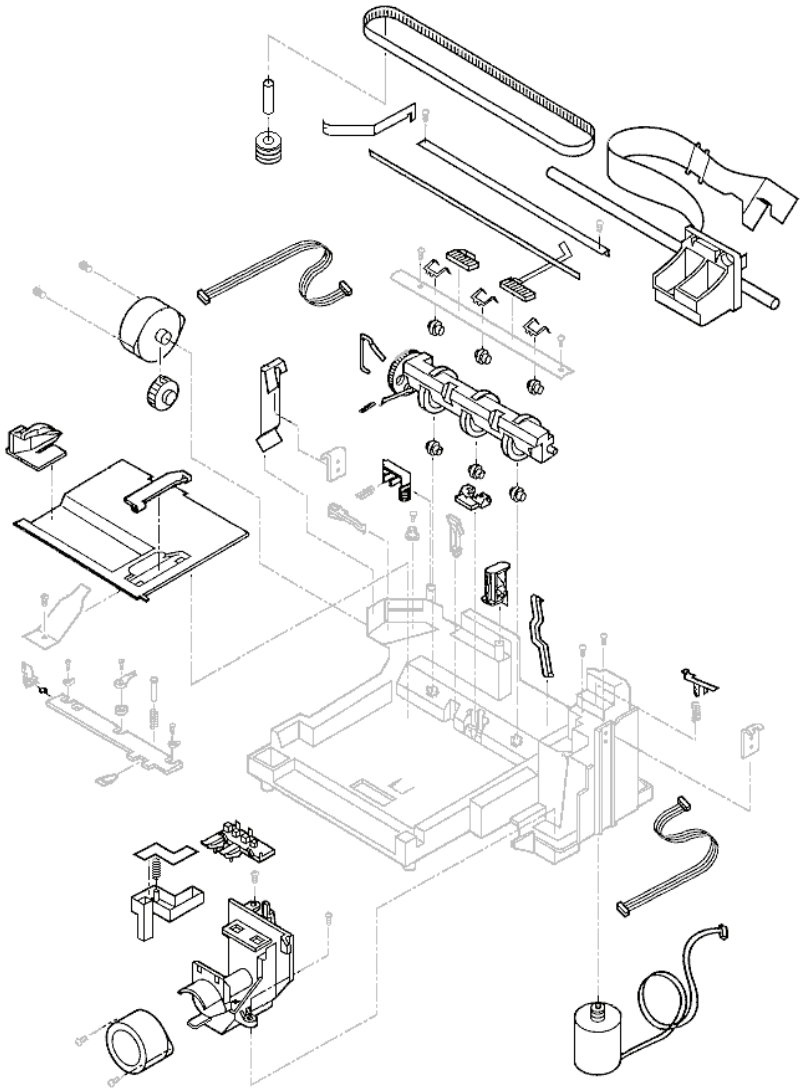
$$\text{Resolution} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

Note: If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

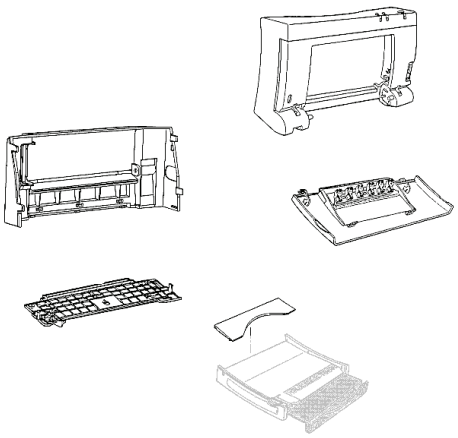
ANEXO I: Desmontagem das impressoras 692C e 420C da HP



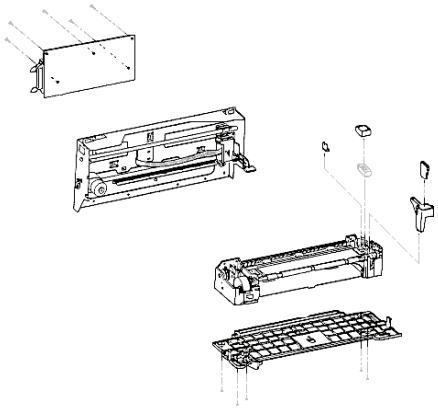
DeskJet 692C - Complete Printer Assembly [36]



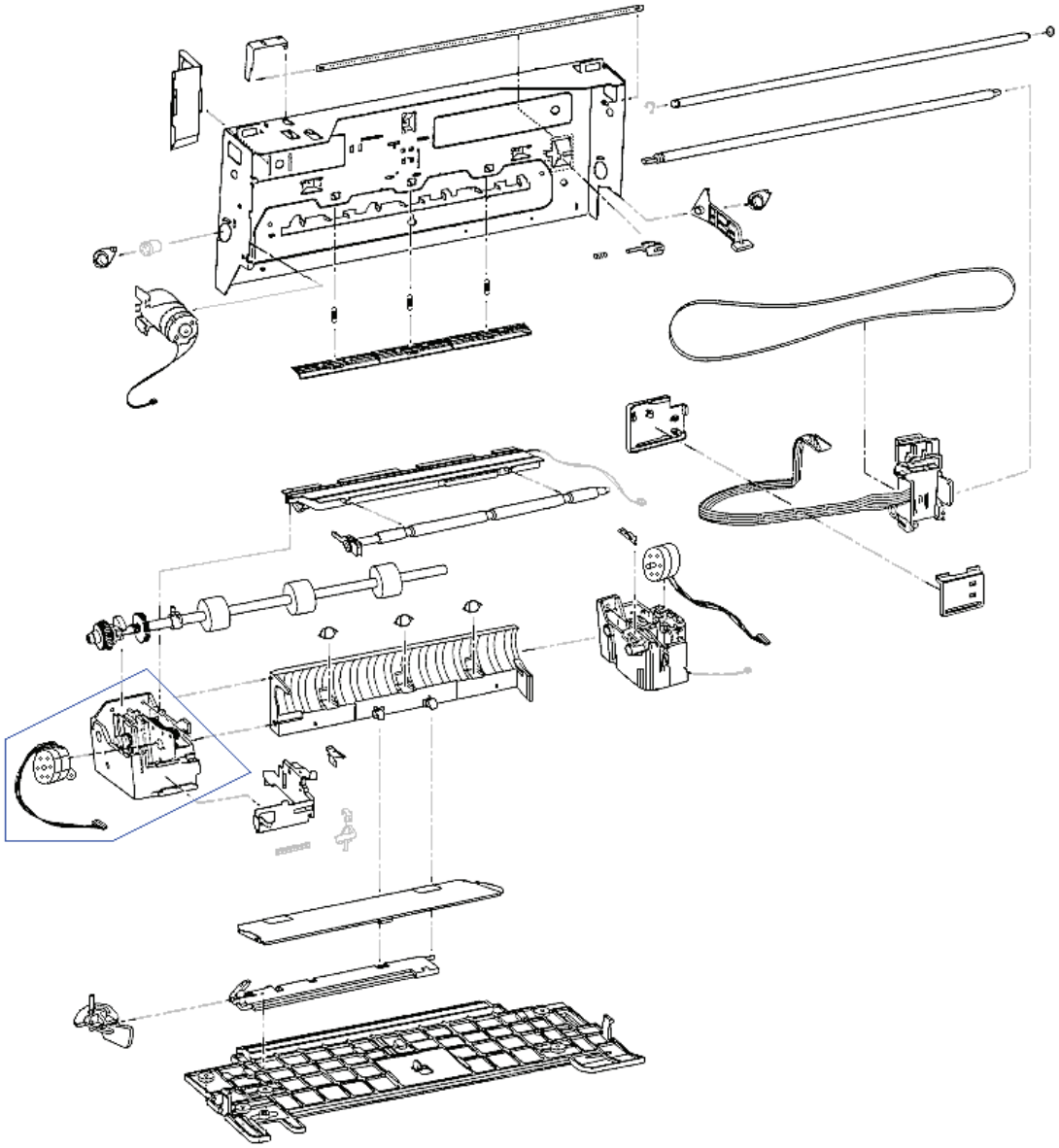
DeskJet 692C - Print Mechanism Assembly [36]



DeskJet 420C - External Case Parts [36]



DeskJet 420C - Internal Components [36]



DeskJet 420C - Print Mechanism [36]