ISSN: 1984-3151

CONTROLADOR LÓGICO PROGRAMÁVEL UTILIZANDO PIC 18F4550

PROGRAMMABLE LOGIC CONTROLLER USING MICROCONTROLER PIC 18F4550

Warley A. Eleutério[1]; Wagner A. A. Hovadich[2]; Eduardo Q. Braga[3]

1 Engenheiro Eletricista. UNIBH, 2011. Projetista de sistemas elétricos e Automação – C&W Projetos e consultoria. Belo Horizonte, Minas Gerais. warley@cewprojetos.com.br.

2 Engenheiro Eletricista. UNIBH, 2011. Projetista de sistemas de Automação – IHM Engenharia. Belo Horizonte, Minas Gerais. wagner.hovadick@gmail.com.

3 Mestre em Engenharia elétrica. UFMG, 2007. Centro Universitário de Belo Horizonte. Belo Horizonte, Minas Gerais; eduardo.braga@prof.unibh.br.

Recebido em: 30/10/2011 - Aprovado em: 16/12/2011 - Disponibilizado em: 30/12/2011

RESUMO: Desde a criação dos relés eletromecânicos os processos industriais vêm se modernizando, principalmente após a descoberta dos microprocessadores que impulsionaram o desenvolvimento dos controladores lógicos programáveis. Os controladores tornaram os processos produtivos flexíveis do ponto de vista de manutenção, expansão e principalmente aumentando a qualidade dos produtos e qualidade de vida dos envolvidos, já que esta tecnologia vem retirando os homens de processos insalubres e tornando estes processos inteligentes e independentes. A cada dia os controladores ficam mais poderosos e consequentemente mais caros, o que vem tornando estes dispositivos inviáveis para processos simples e com baixo valor agregado. Este fato motivou o desenvolvimento de um controlador de baixo custo de produção e de fácil implementação para profissionais que atuam nestes processos mais simples. Para o desenvolvimento deste trabalho foi utilizado o Microcontrolador PIC18F4550. Depois de concluídas as simulações foi observada a possibilidade de se aumentar os módulos de entrada e saída do controlador, o que torna seu custo/beneficio ainda maior.
PALAVRAS-CHAVE: Controladores Lógicos Programáveis. PIC18F4550. Microcontrolador.

ABSTRACT: Since the creation of electromechanical relays industrial processes are being modernized, especially after the discovery of the microprocessors that drive the development of programmable logic controllers. Controllers made flexible production processes in terms of maintenance, expansion and mainly by increasing product quality and quality of life of those involved, as this technology is taking the men of unhealthy processes and making these processes intelligent and independent. Every day the drivers become more powerful and therefore more expensive, which is making these devices impractical for simple processes and low added value. This fact motivated the development of a controller of low production cost and easy implementation for professionals who work in simple cases. To develop this study we used the PIC18F4550 microcontroller. After completing the simulations, we observed the possibility of increasing the input and output modules of the controller, which makes their cost/benefit even more.
Keywords: Programmable Logic Controllers. PIC18F4550. Microcontroller.

---

1 INTRODUÇÃO

Até no final dos anos 70 sempre que se desejava controlar algum processo industrial, desde os mais simples aos mais complexos, a alternativa utilizada eram os controles de processos a relés. Este tipo de controle teve seu início em 1968, sendo utilizado pela

primeira vez na divisão hidramática da *General Motors.* Embora este tipo de controle funcionasse de maneira satisfatória, sempre que fosse necessária alguma alteração no processo, onde este sistema estava sendo empregado, era necessário que todo o sistema fosse re-projetado e novamente implementado. Isto demandava além de um alto custo, muito tempo. (PLÍNIO, 2008)

Segundo PLÍNIO (2008), no final dos anos 60, os engenheiros e projetistas perceberam que seria possível criar um hardware que de uma forma mais flexível pudesse tornar este processo de controle menos oneroso e que pudesse ser modificado em menor tempo, pois a implementação dos controles passa a ser via software e não mais com os relés. Desta forma surgiram os primeiros Controladores Lógico Programáveis – CLP's.

Em sua estrutura básica, o CLP deve ser composto por um controlador central, CPU, que fará o controle e monitoramento do processo através de dispositivos de I/O. (PLÍNIO, 2008)

Para que seja possível sua utilização em ambientes industriais, o hardware do CLP deve ter a capacidade de suportar uma alta tensão e corrente em seus terminais de I/O, possuir um sistema de memória não volátil, para que caso a alimentação de seu circuito seja interrompida, não sejam perdidos seus parâmetros, nem programa. Além disto é desejável que ele seja robusto, suportando altas temperaturas e manuseio. (PETRUZELLA, 2005),

As vantagens em se utilizar um CLP são inúmeras e ultrapassam as que motivaram sua criação. Além da redução nos custos e no tempo de montagem os CLP's também oferecem confiabilidade, flexibilidade, a capacidade de realizar uma grande variedade de tarefas de controle, desde ações simples e repetitivas até a manipulação de dados complexos, a possibilidade de comunicação com outros CLP's e computadores e, também, a capacidade de identificar e indicar, para seu operador, a existência de algum problema, facilitando desta forma à manutenção. (PLÍNIO, 2008)

1.1 PROBLEMA DE PESQUISA

Embora uma das ideias principais do CLP fosse o de reduzir custos, devido a evolução e alta complexidade de alguns grandes processos, o avanço da tecnologia e o surgimento de microprocessadores cada vez mais poderosos, os CLP's passaram a realizar além de controles lógicos, tarefas com alto grau de complexidade. Este avanço, entretanto, fez com que o custo destes equipamentos se tornasse muito alto, inviabilizando o seu uso em tarefas mais simples ou que não possuam um alto valor agregado.

De acordo com Eng. André Oliveira, Msc (*National Instruments*), · 77% dos CLPs são utilizados em pequenas aplicações, cerca de 72% das E/S dos CLPs são digitais e 80% dos desafios em aplicações com CLPs são solucionados com um conjunto de 20 instruções ladder.

1.2 MOTIVAÇÃO

As grandes vantagens oferecidas pelo CLP, seu alto custo de aquisição e a necessidade de se controlar processos de baixo valor agregado, fizeram surgir o interesse em pesquisar e demonstrar que um CLP pode ser desenvolvido e controlado por microcontrolador, de forma a torná-lo simples e de baixo custo.

1.3 OBJETIVOS

1.3.1 OBJETIVO GERAL

Criar um CLP utilizando microcontrolador de baixo custo, que seja versátil e de fácil manuseio.

1.3.2 OBJETIVOS ESPECÍFICOS

- Criar projeto de hardware necessário para a implementação do projeto, tais como: CPU, módulos de entrada e saída digitais e interface de comunicação serial.
- Desenvolver código de programação em linguagem C (Firmware) para o funcionamento do microcontrolador PIC18F4550.
- Criar mecanismos para a programação do CLP.
- Demonstrar a possibilidade de monitoramento de processo (Supervisório).
- Realizar teste do CLP com simuladores.
- Avaliar a viabilidade econômica do projeto.

1.3.3 JUSTIFICATIVA

A ideia do controle automático de processo se iniciou na década de 20, quando Henry Ford criou a linha de produção para a fabricação de automóveis e foi impulsionado com o desenvolvimento de dispositivos semicondutores. Desde então, os controladores (CLP) vem se modernizando a cada dia. A automação tem um importante papel na sociedade, pois vem proporcionando além de melhorias na qualidade dos produtos, uma melhora na qualidade de vida da sociedade, uma vez que substitui o homem em trabalhos repetitivos e insalubres.

Este estudo visa demonstrar o quanto os microcontroladores são úteis e podem estar presentes mesmo para as mais simples aplicações, transformando o ambiente em um local mais confortável, podendo contribuir para qualidade de um determinado processo e até mesmo ajudando na preservação do meio ambiente. Como exemplo de preservação, pode-se citar o simples controle de iluminação, ventilação e ar condicionado de um local onde existem vários ambientes isolados.

A relevância do tema está em tornar acessível os dispositivos de controle, tendo em vista a inviabilidade financeira de aquisição de controladores automáticos existentes no mercado para aplicações de baixa complexidade.

2 REFERENCIAL TEÓRICO

2.1 CONTROLADORES LÓGICOS PROGRAMÁVEIS

Segundo IEC 61131-1, *International Electrotechnical Commission* (2003), os CLP's podem ser descritos como:

Sistema eletrônico operado digitalmente, projetado para o uso em ambiente industrial, que usa uma memória programável para a armazenagem interna de instruções orientadas para o usuário implementar funções específicas, tais como lógica, sequencial, temporização, contagem e aritmético, para controlar, através de entradas e saídas digitais ou analógicas, vários tipos de máquinas ou processos. O controlador programável e seus periféricos associados são projetados para serem facilmente integráveis em um sistema de controle industrial e usados com facilidade em todas suas funções específicas. (IEC 61131-1, 2003)

Segundo Petruzella (2005), um CLP é um equipamento eletrônico microcontrolado que deve possuir uma interface amigável com o usuário e que tem como função executar controles de vários tipos e de diferentes níveis de complexidade.

O mesmo autor afirma que em sua estrutura básica os CLP's (FIG. 1) são constituídos de:

1. Entradas

2. Saídas

3. Unidade Central de Processamento (Central Processing Unit – CPU)

4. Memória para o programa e armazenamento de dados

5. Fornecimento de alimentação

6. Dispositivo de programação

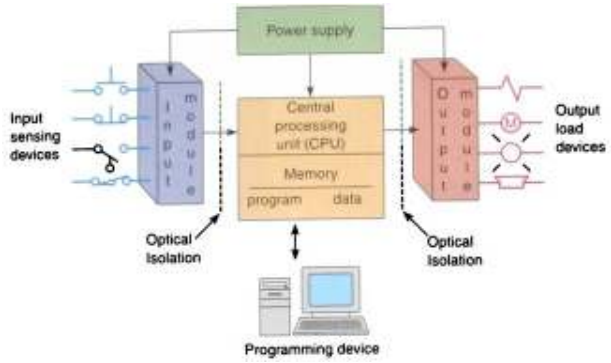


Figura 1 – Estrutura Típica do PLC

Fonte - PETRUZELLA, 2005.

As entradas são os terminais conectados ao CLP que criam o meio pelo qual os dispositivos que estão em campo podem receber e enviar sinais necessários para o controle do processo. (PLÍNIO, 2008)

O mesmo autor ressalta que os sinais recebidos em módulo de entrada podem ser do tipo discreto (Chaves, botoeiras, contatos de relés, teclados) ou do tipo analógico (sensores de vazão, transdutores de corrente, medidores de pressão).

Plínio (2008) ainda afirma que estes sinais são recebidos dos equipamentos de campo e são, em geral, de 120Vca ou em alguns casos de 24Vcc. As portas de entrada do CLP recebem estes sinais vindos do campo e a manipulam de forma que possam ser utilizados pelo CLP. Esta manipulação é necessária porque os componentes internos de um CLP são digitais e operam com baixo nível de tensão (5Vcc – 12Vcc) e, portanto, devem ser protegidos de flutuações de tensão. Para que os componentes internos fiquem eletricamente isolados dos terminais de entrada, os CLP's empregam um isolador ótico, que usa a luz para acoplar os sinais de um dispositivo elétrico a outro.

O mesmo autor ensina que as saídas são o meio através do qual o CLP efetua o controle do processo acionando cargas ou sinalizando algum evento. Os módulos de saída podem ser a relés, a transistores ou tiristores. As saídas a relés podem ser utilizadas para sinais CC ou CA, pois a comutação acontece através de seus terminais NC (Normalmente fechado) ou NA (Normalmente aberto), o de saída, entretanto, está mais sujeita a desgastes e é mais lenta que os outros. As saídas a transistores são rápidas e não estão sujeitas ao mesmo desgaste que as saídas a relés, porém não pode comutar cargas que demandam grande quantidade de corrente e sinais CA. Já as saídas a tiristores são semelhantes às saídas a transistores, porém comutam exclusivamente sinais AC e podem comutar cargas com correntes mais altas.

Conforme Plínio (2008), a CPU (Unidade Central de Processamento) é o principal componente do CLP, pois recebe as informações (FIG. 2), executa todos os algoritmos programados em linguagem de programação, por exemplo, do tipo ladder e joga nas saídas os comandos necessários ao processo que se deseja controlar. Através das entradas, o hardware recebe o sensoriamento de campo, processa o algoritmo armazenado em sua memória e aciona as saídas, respectivamente. Esta unidade possui internamente memória de dados, programas, valores pré-definidos, acumuladores de contadores/ temporizadores e ainda outras constantes e variáveis que a torna um excelente controlador para infinitas aplicações.

Figura 2 – Sistema básico de estrutura/Controle Fonte - PLÍNIO, 2008.

O mesmo autor ainda revela que controle eletrônico do CLP é alimentado através de tensão continua, para tanto, existem conversores para transformação dos diferentes tipos de alimentação (Potencial) existentes atualmente. O sensoriamento de campo também deve ser alimentado. A tensão de alimentação em alguns casos é proveniente de fontes ininterruptas para que quando em uma possível falta de energia o operador do sistema tenha o verdadeiro status da condição e ainda para que informações de nível 2 (controle de processo) não sejam perdidas. Em alguns casos algumas cargas/sensores são alimentadas também por fontes ininterruptas, exemplos deste tipo de aplicação são fornos siderúrgicos que precisam manter seu sistema de arrefecimento ou dispositivos de segurança humana onde falhas são inaceitáveis.

Para a elaboração dos algoritmos, cada fabricante disponibiliza um software de programação dedicado, este software é instalado em um sistema computacional normal. Através deste software é possível criar, editar, documentar, armazenar e localizar as falhas dos diagramas ladder, gerando também relatórios impressos. As instruções dos softwares são baseadas em símbolos gráficos para as várias funções. Não é necessário o conhecimento das linguagens mais avançadas de programação para se usar o software, bastando um entendimento genérico dos diagramas elétricos funcionais. (PLÍNIO, 2008)

2.2 MICROPROCESSADORES E MICROCONTROLADORES

Os microprocessadores ou CPU (*Central Processing Unit*) surgiram em 1971 fabricados pela Intel Corporation com a grande novidade de possuir em um só encapsulamento vários componentes, que até então só poderiam ser encontrados separadamente, desta forma, tornou-se mais fácil a implementação, comunicação, tamanho, consumo de energia e preço de novos projetos. (SOUZA,2007)

Um microprocessador é um circuito extremamente complexo que pode possuir internamente entre alguns milhares (*Z80, Zilog Corporation -1976*) a milhões de transistores. Estes transistores internos constituem diversos circuitos lógicos, tais como, timers, contadores, registradores e outros dependendo do modelo escolhido. A disposição interna destes circuitos é tal que permite ao microprocessador executar operações lógicas aritméticas e de controle.(SOUZA,2007)

O Microprocessador é o elemento principal de coordenação de todo o sistema. Ele recebe instruções de manipulação dos dados, realiza as tarefas e gera os resultados.(SOUZA,2007)

De acordo com Gimenez (2002), o microcontrolador, pode ser considerado como sendo um computador de um único chip, ou seja, trata-se de um sistema computacional completo composto por onde estão incluídos uma CPU (*Central Processor Unit*), memória de dados e programa, sistema de clock, portas de I/O (*Input/Output*), além de outros possíveis periféricos. Tais como, módulos de temporização e conversores A/D entre outros, integrados em um mesmo componente.

Segundo Pereira (2005), os microcontroladores são os componentes de processamento digitais mais difundidos e utilizados na atualidade. Estes podem ser encontrados desde em pequenos aparelhos domésticos até em grandes e complexos aparelhos, como os satélites por exemplo. Hoje diversos fabricantes podem ser encontrados, estes produzem diversos modelos que facilitam a criação de novas aplicações, esta diversidade por sua vez, exige uma grande pesquisa com o intuito de escolher aquele microcontrolador que melhor atenda aos requisitos de seu projeto.

Existem duas principais arquiteturas de microcontroladores, Havard e Von-Newmann. A arquitetura Havard tem como principal característica a

existência de dois barramentos internos, um para acesso a memória de dados e o outro para acesso à memória de programa, isto resulta em um aumento na capacidade de fluxo de dados. Já na arquitetura Von-Newmann existe apenas um barramento que é compartilhado pela memória de dados e pela memória de programa, o que resulta em uma limitação na banda de operação. (MIYADAIRA, 2009)

Em relação ao conjunto de instruções de um microcontrolador (SET), ele pode ser classificado como RISC (*Reduced Instruction Set Computer*) ou CISC (*Complex Instruction Set Computer*). (Miyadaira, 2009)

Segundo Morimoto (2007), um microcontrolador CISC (*Complex Instruction Set Computer*), tem a capacidade de executar centenas de instruções complexas diferentes, sendo extremamente versátil. Os microcontroladores RISC (*Reduced Instruction Set Computer*), ao contrário dos CISC, são capazes de executar apenas algumas poucas instruções (em torno de 35 a 75 dependendo do modelo de microcontrolador). Justamente por isso, os chips baseados nesta arquitetura são mais simples e muito mais baratos. Outra vantagem dos processadores RISC, é que, por terem um menor número de circuitos internos, podem trabalhar a frequências mais altas, sendo assim um processador RISC é capaz de executar tais instruções muito mais rapidamente

De uma maneira geral, as memórias de programa existentes nos microcontroladores são do tipo FLASH (*Electrically-Erasable Programmable Read-Only Memory*), ROM (*Read Only Memory*), EPROM (*Erasable Programmable Read Only Access*) ou OTP (*One Time Programmable*). Todas estas memórias são do tipo não volátil, ou seja, o código de programa armazenado não é perdido caso a alimentação do circuito seja interrompido. (IDOETA, 2001)

As principais características das memórias de programa acima citadas são:

ROM: Uma vez gravada pelo fabricante, este tipo de memória não permite que seu conteúdo seja alterado pelo usuário. Os microcontroladores que possuem este tipo de memória, em geral, apresentam baixo custo em relação aos que possuem os outros tipos de memórias citadas e são recomendados quando o código de programa já está consolidado, não havendo mais necessidade de modificá-lo, e quando for desejável a produção do circuito em grande escala. (IDOETA, 2001)

EPROM: Pode ser apagado e reprogramado muitas vezes, porém para apagar o código residente é necessária a exposição da janela de quartzo do componente a luz ultravioleta por um determinado tempo que é definido pelo fabricante do microcontrolador. Este tipo de componente, devido ao seu processo de fabricação, tem um custo superior aos que possuem outros tipos de memória. (IDOETA, 2001)

OTP: É tolerada apenas uma programação, diferentemente da ROM esta programação pode ser feita pelo usuário, possui menor custo que as memórias do tipo EPROM e FLASH. (IDOETA, 2001)

FLASH: É o tipo de memória mais flexível dentre os outros tipos de memória de programa, pode ser apagada e reprogramada eletricamente, até 1.000.000,00 de vezes, dependendo do fabricante, os microcontroladores que possuem este tipo de memória são extremamente úteis em aplicações, onde é necessária a modificação do programa constantemente, pois podem, em grande maioria, ser reprogramados no próprio circuito. (IDOETA, 2001)

O outro tipo de memória encontrada nos microcontroladores é a memória de dados, definida como memória RAM (*Random Access Memory*), esta memória é do tipo volátil e é responsável por armazenar as variáveis e constantes do sistema. Como os dados constantes nesta memória são perdidos em caso de desligamento da alimentação do

circuito, os valores das variáveis de sistema devem ser carregados sempre que o sistema for iniciado. (IDOETA, 2001)

Segundo Gimenez (2002), para a comunicação com o mundo externo, os microcontroladores possuem pinos dedicados, denominados I/O (Input – Output). O sentido do fluxo de dados nestes pinos é definido como entrada (I) ou saída (O). Os pinos de saída são utilizados para o controle de periféricos do sistema e os pinos de entrada serão responsáveis por receber os sinais vindos dos periféricos para que o microcontrolador possa tomar as decisões atribuídas a aquela situação.

A velocidade de processamento de um microcontrolador está diretamente relacionada à frequência de seu sinal de *CLOCK*. Este sinal pode ser gerado internamente pelo microcontrolador através de um circuito RC interno (porém sem grande precisão) ou por um oscilador externo, a cristal de quartzo, com alta precisão. (GIMENEZ, 2002).

2.2.1 MICROCONTROLADORES PIC

Os microcontroladores PIC são uma família de microcontroladores fabricados pela *Microchip Technology,* que apresentam uma estrutura interna do tipo Havard e o conjunto de instruções tipo RISC, seu barramento de dados é sempre de 8 bits e o barramento de instruções pode ser de 12, 14 e 16 bits dependendo do modelo do microcontrolador. Este tipo de arquitetura permite que uma instrução seja buscada na memória, enquanto outra instrução ainda estiver sendo executada, isto torna o processamento mais rápido. Outra vantagem é que como o barramento de instruções é maior que o barramento de dados, o OPCODE da instrução já inclui o dado e o local onde ela irá operar, isto significa que apenas uma posição de memória é utilizada a cada instrução o que economiza memória de programa. (SOUZA, 2000).

2.2.2 MICROCONTROLADORES PIC 18F4550

O PIC18F4550 é um microcontrolador de 8 bits atual com arquitetura Havard e conjunto de instruções tipo RISC, ele possui uma memória interna de 32 Kbytes para armazenamento do programa residente e 2048 bytes de memória RAM. Sua tensão de alimentação pode ser da ordem de 4 a 5,5 Volts e sua frequência de operação é de até 48MHz, a esta frequência ele é capaz de executar até 12 milhões de instruções por segundo. (MIYADAIRA, 2009).

O mesmo autor afirma que este modelo de Microcontrolador possui 40 pinos dos quais 35 podem ser configurados como portas I/O e diversos periféricos tais como memória EEPROM de 256 bytes, um módulo CCP e ECCP um módulo SPI e um módulo I2C. Possui também 13 conversores A/D com 10 bits de resolução cada e tempo de amostragem programável, 02 comparadores analógicos, uma comunicação EUSART, um timer de 8 bits e três timers de 16 bits cada, um módulo de detecção de tensão alta/baixa (HLVD) e um módulo USB 2.0 com a capacidade de operar nos modos low-speed (1,5Mbps) ou full-speed (12Mbps).

Segundo Miyadaira (2009), o PIC18F4550 conta com o recurso PIPELINE para aumentar a velocidade de processamento. Este recurso faz com que ao mesmo tempo em que uma instrução é executada, a próxima instrução é localizada e carregara no registro de instruções (IR) em um ciclo de máquina. Devido a este comportamento é possível afirmar que uma instrução é executada efetivamente em apenas um ciclo de máquina.

O ciclo de máquina do microcontrolador PIC18F4550 pode ser dividido em 04 fases (FIG. 3), de Q1 a Q4, resultando em uma velocidade de ciclo de máquina

equivalente a ¼ do valor do oscilador. (MIYADAIRA, 2009).

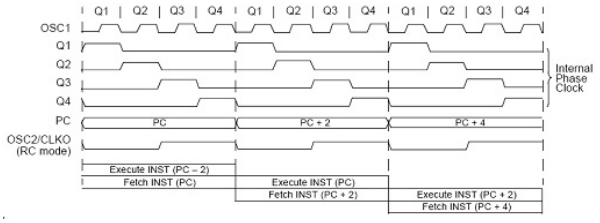


Figura 3 - Ciclo de máquina do PIC18F4550
Fonte - *Datasheet* PIC18F4550

É possível verificar que cada ciclo de máquina executa dois eventos de forma paralela (FIG. 3), um ciclo de localização (Fetch) e um ciclo de execução (Execute). O Ciclo de localização incrementa o program counter (PC) durante o ciclo Q1, a instrução é então decodificada e executada durante os ciclos Q2 a Q4. A memória de dados é lida em Q2 e escrita em Q4. (MICROCHIP, 2009).

Pinagem do Microcontrolador:

Pinagem do Microcontrolador PIC18F4550 (FIG. 4):

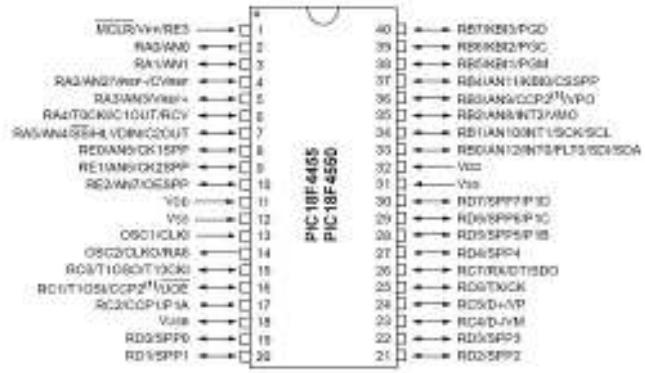


Figura 4 - Pinagem do PIC18F4550 (*Datasheet* PIC18F4550)

A descrição de cada pino do microcontrolador pode ser encontrada no documento de referência *Datasheet* do fabricante (FIG. 4). É possível perceber que devido a grande versatilidade do microcontrolador existem vários pinos que desempenham mais de uma função. A escolha de qual será a função do pino dependerá do projeto a ser desenvolvido. (MIYADAIRA, 2009).

2.2.3 LINGUAGEM DE PROGRAMAÇÃO C E O MPLAB

Para que os microcontroladores executem as tarefas desejadas é necessário que ele seja programado. Existem diversas linguagens de programação as mais comuns em microcontroladores são o assembly, basic e C.

O C é uma linguagem de programação genérica desenvolvida em texto e é utilizada para criação de programas diversos e apesar de sua complexidade de programação vem se tornando popular devido a sua versatilidade. Nesta linguagem, para cada ação ou comando desejado existe um código textual específico, armazenado em alguma biblioteca. Esta programação é textual e além dos milhares de comandos disponíveis ainda faz distinção de letras maiúsculas e minúsculas no código (Case sensitive), o que torna ainda mais difícil esta programação.

Segundo Dornelles (1997), a linguagem C é uma linguagem de alto nível, genérica. Foi desenvolvida para programadores, tendo como meta características de flexibilidade e portabilidade. O C é uma linguagem que nasceu juntamente com o advento da teoria de linguagem estruturada e do computador pessoal. Assim tornou-se rapidamente uma linguagem "popular" entre os programadores. O C foi usado para desenvolver o sistema operacional UNIX, e ainda hoje está sendo usada para desenvolver novas linguagens.

Para que o programa fonte seja interpretado pelo microcontrolador é necessário que ele seja convertido para código de máquina e posteriormente gravado em sua memória interna, o MPLAB é uma plataforma de desenvolvimento fornecida gratuitamente pela Microchip (www.microchip.com) que tem como função gerar os códigos que poderão ser gravados no

microcontrolador PIC. Esta plataforma integra em apenas um ambiente todo o processo de gerência do projeto, desde a edição do programa fonte, compilação e simulação até a gravação do Microcontrolador PIC. (MARTINS, 2011)

2.2.4 LINGUAGEM DE PROGRAMAÇÃO LADDER

O ladder é uma linguagem de programação gráfica, em forma de diagrama elétrico, de fácil criação e interpretação por estudantes e profissionais que atuam com eletricidade. Esta linguagem representa ligações físicas entre componentes (sensores e atuadores) que são conectados as entradas e saídas do dispositivo e que assumem posições de memória de dados dentro do dispositivo. Estas memórias assumem o status tanto da real condição dos sensores externos como do comando que será transmitido aos atuadores, sendo então todas as memórias de dados variáveis do processo que se deseja controlar. Todas estas variáveis (entradas e saídas) são tratadas dentro do programa, assim como variáveis de controle internas ao controlador, como, por exemplo, contadores ou memórias auxiliares necessárias à lógica, que se deseja criar. Um exemplo de linguagem de programação pode ser visto na FIG. 5.

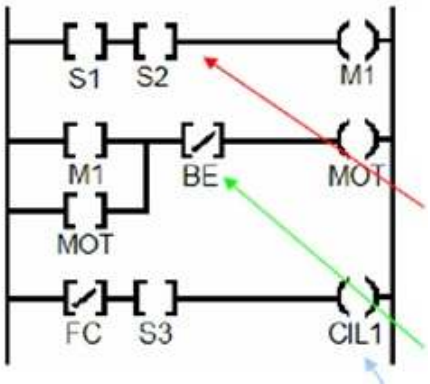


Figura 5 – Diagrama em Ladder
Fonte: CORTELETTI, 2007.

Segundo CORTELETTI (2007), para o diagrama da FIG. 5, tem-se o controle de 3 elementos, sendo estes M1, MOT1, e CIL1. Estes elementos podem ser BOBINAS (ATUADORES) ou MEMÓRIAS (relés internos).

Os elementos S1, S2, BE, VC, e S3 só aparecem ao lado esquerdo do diagrama, no formato de colchetes [ ], o que pressupõe que sejam sensores (entradas).

Na primeira linha, observa-se que a regra do programa define que a saída M1 irá ativar somente se os sensores S1 e S2 estiverem AMBOS ligados.

Na segunda linha deste programa, observa-se que a regra determina que a saída MOT irá ligar se BE estiver DESLIGADO (a barra significa inversão) e se M1 ou MOT estiver acionado (ao menos um destes)

Na terceira linha, observa-se que o atuador CIL1 irá ativar caso o sensor FC estiver DESLIGADO (novamente observa-se a barra), e se o sensor S3 estiver acionado. (CORTELETTI, 2007)

2.3 FIRMWARE DE UM DISPOSITIVO

Firmware é o conjunto de instruções operacionais normalmente desenvolvidos com operações muito básicas de baixo nível sem as quais o dispositivo não seria capaz de executar suas tarefas. Basicamente os Firmwares são códigos lógicos que interpretam os sinais físicos dos periféricos e os tornam transparentes no ponto de vista de programação e dos demais pontos do programa, que está em execução simplificando então a utilização destes periféricos. O Firmware desenvolvido para o PIC18F4550, aqui já entendido como PLC é o conjunto de instruções lógicas pré-definidas, que tem por objetivo simplificar a programação em linguagem C. Assim como um firmware torna simples a programação de um periférico em um dispositivo mais complexo, para cada função lógica disponível no Firmware desenvolvido existe um código em C correspondente, que receberá quando chamado as variáveis que deverá tratar.

2.4 SUPERVISÃO

Por padrão o Firmware criado envia através da porta serial os estados das entradas e das saídas a cada ciclo de leitura de entrada e atualização de saídas em formato hexadecimal. Este valor pode ser tratado e analisado isoladamente.

3 METODOLOGIA

Após este estudo foi iniciada as pesquisas bibliográficas a fim de determinar qual tipo de hardware PLC poderia ser criado de forma que seu custo final não fosse muito alto. Foi determinada então a construção de um PLC utilizando um microcontrolador da família Microchip. Para determinar o modelo a ser utilizado, procurou-se por um microcontrolador que fosse moderno, bem completo e que fosse barato.

Escolheu-se então o PIC18F4550 que possui um custo benefício muito bom, pois já possui integrado a ele interfaces RS485, USB e RS232, 4 timers, entradas e saídas analógicas . Tudo isto faz com que o custo final do projeto possa ser reduzido devido ao fato de ser necessário um menor número de componentes externos.

Após o processo de escolha do microcontrolador, iniciou-se o processo de elaboração do hardware do modulo de CPU e I/O.

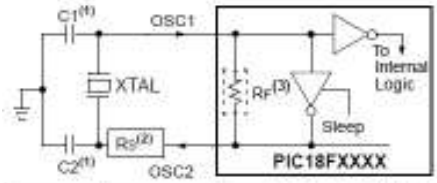
3.1 DESENVOLVIMENTO DO HARDWARE

O Hardware foi desenvolvido em três blocos distintos, sendo os blocos Módulo principal, módulo de entrada digital e módulo de saída digital.

Para o **modulo principal** (Apêndice 1) foram considerados um CI PIC18F4550, dois capacitores 15pF, cristal de clock 20MHz, dois resistores de 10K omhs, dois resistores de 10 omhs, 2 botões, um capacitor de 470nF, cinco capacitores de 1uF, um CI MAX232, uma porta USB, uma porta RS232 CONN-D9F, conector 71918-114LF, conector TN140 2 pinos.

O CI PIC18F4550 é o elemento principal do módulo, responsável por receber as entradas, tratá-las conforme programa desejado e enviar o resultado às saídas.

Os **dois capacitores de 15pF** juntamente com o **cristal de clock 20MHz** são responsáveis por produzir a frequência de chaveamento para que o PIC possa executar suas tarefas. Este conjunto pode ser comparado ao processador de um computador. Este conjunto é típico do cristal utilizado e o conjunto é recomendado pelo *datasheet* do PIC184550 (FIG. 6).



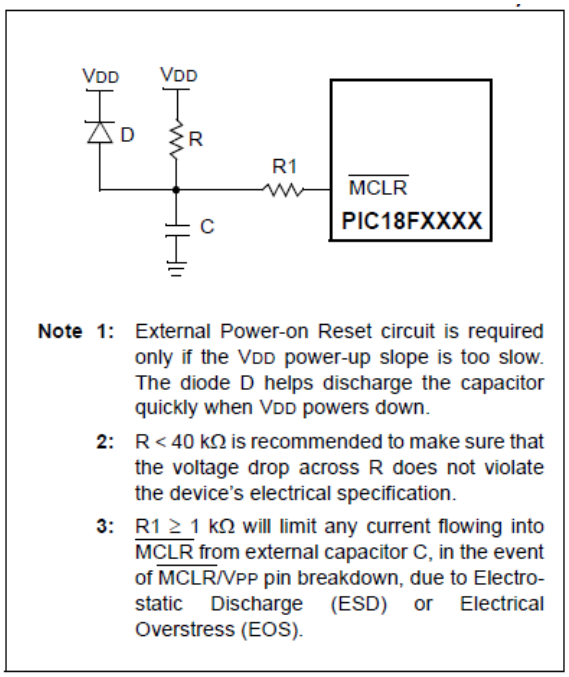
Osc Type	Crystal Freq	Typical Capacitor Values Tested:	
		C1	C2
XT	4 MHz	27 pF	27 pF
HS	4 MHz	27 pF	27 pF
	8 MHz	22 pF	22 pF
	20 MHz	15 pF	15 pF

Figura 6 - Circuito do cristal de chaveamento

O **botão LOADER** juntamente com o **resistor de 10K omhs** são responsáveis por ativar o processo de carregamento do programa dentro do PIC18f4550.

O **capacitor de 470nF** é utilizado por recomendação do *datasheet* do PIC18f4550.

O **botão reset** é responsável por reinicializar o PIC18f4550 (FIG. 7). O **resistor de 10K omhs** juntamente com o **capacitor 1uF** são responsáveis por auxiliar o comando reset através do botão e também garantir o reset quando o modulo principal for desliga/ligado.



Note 1: External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

3: R1 ≥ 1 kΩ will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown, due to Electro-static Discharge (ESD) or Electrical Overstress (EOS).

Figura 7 – Circuito de Power reset.

O **CI MAX232** é responsável por fazer a conversão do sinal TTL para a linguagem serial e os **capacitores de 1uF** conectados a ele são recomendação do respectivo *datasheet* (FIG. 8).

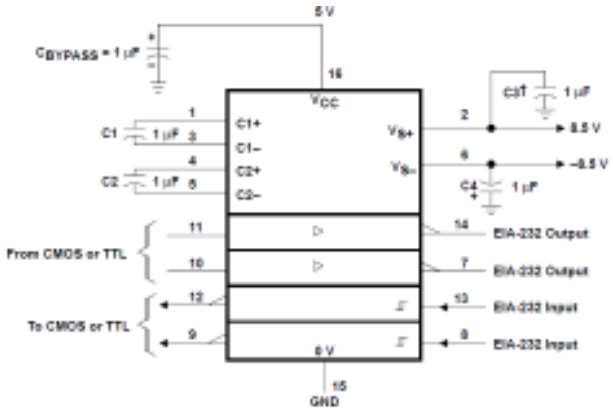


Figura 8 - Circuito de operação típica do CI MAX 232

O **conector CONN-D9F** é interface de rede RS232 com o circuito do módulo principal.

A **porta USB** é a interface de rede USB com o circuito do módulo principal e os **resistores de 10 omhs** conectados a esta porta **l**imitam a corrente em caso de curto circuito nesta porta. Estes resistores são recomendação do livro (Microcontroladores Pic18, Aprenda e Programe em LINGUAGEM C, MIYADAIRA), para proteção tanto do PIC18F4550 como da interface USB.

Para o **modulo de entrada digital** (Apêndice 2) foram considerados um CI 74LS241, conector 71917-114LF 14 pinos e individualmente para cada entrada resistores de 10K ohms, dois resistores de 2k2 ohms, um resistor de 22komh, um LED emissor de luz e um transistor fotoacoplado 4N25.

O conector 71917-114LF, 14 pinos tem a função de conectar eletricamente o módulo principal ao modulo de entrada.

O **resistor de 10K** tem a função *Pull Dow* que garante nível lógico zero "0" na entrada quando a mesma não estiver acionada.

O **resistor de 22K** tem a função de dessensibilizar a entrada digital a tornando menos susceptível a ruídos provenientes do circuito externo.

O **resistor de 2K2 em série com o diodo** tem a função de sinalizar que a entrada esta em nível alto e o último **resistor de 2K2** tem a função de limitar a corrente que acionará o fotoacoplador.

O Fotoacoplador (CI 4N25) tem a função de isolar eletricamente o circuito externo do circuito interno da placa de entrada e modulo principal.

Para o **módulo de saída digital** (Apêndice 3) foram considerados um Flip Flop (CI 74F374), conector 71917-114LF, 14 pinos e individualmente para cada saída resistores de 390 ohms, 360 ohms, 470 ohms, 39 ohms, capacitores de 50nF, 10nF, conector TN140 2 pinos, tiristor fotoacoplado MOC3021, tiristor Q7004L4.

O conector 71917-114LF, 14 pinos tem a função de conectar eletricamente o módulo principal ao módulo de saída.

O CI 74F374 possui internamente 8 flipflop's tipo D (FIG. 9), com as respectivas entradas digitais (D0 a

D7), uma entrada habilita leitura (CP) comum aos 8 flipflop's que na transição de estado baixo para alto transferem os valores presentes em D para a memória interna do flipflop e um habilita saída (OE) também comum aos 8 flipflop's, que quando em nível lógico alto faz com que as saídas fiquem em alta impedância, porém para a aplicação em questão, considera-se inativa esta função por estar diretamente aterrado. Durante o ciclo de execução do programa, o módulo principal enviará aos flipflop's os comandos que serão recebidos em suas entradas "D", seguido de um pulso de clock (CP), fazendo com que os valores presentes em "D" sejam transferidos a saída "O". O Flipflop manterá a saída fixa ate o próximo ciclo onde o processo se dará novamente.

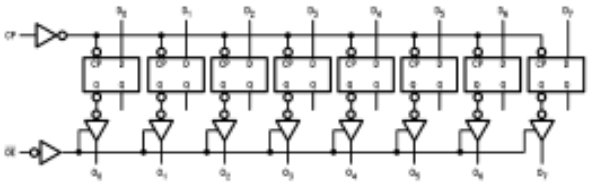


Figura 9 - circuito equivalente do CI 74F374

O resistor de 390 omhs tem a função de limitar a corrente tanto na saída do **CI 74F374** como no anodo do fotoacoplador **MOC3021. O resistor de 360 omhs** também tem a função de limitar a corrente tanto na saída do fotoacoplador **MOC3021** como no circuito de gate do tiristor **BT136.**

O tiristor **BT136** tem a função de chavear a carga a ser acionada.

O **Resistor de 470 omhs** juntamente com o **capacitor de 50nF**, assim como o **resistor de 39 omhs** juntamente com o **capacitor de 10nF,** têm a função de **snubber** no circuito. Esta função é responsável por eliminar ruídos no circuito fazendo com que estes não proporcionem acionamentos indevidos gerados por estes ruídos.

O **Conector TN140** é responsável por fazer a interface elétrica entra a placa de saída e a carga a ser acionada.

Todos os componentes mencionados bem como o circuito de saída após o acoplador (FIG. 10) fazem parte de recomendação do fabricante e podem ser visto no *datasheet* do fotoacoplador **MOC3021.**

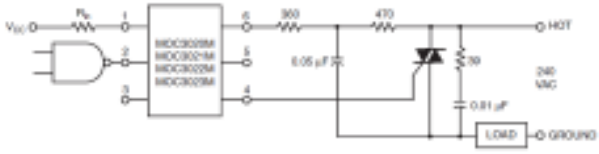


Figura 10 - recomendação *datasheet* do CI MOC 321

3.1.1 DESENVOLVIMENTO DO FIRMWARE

O Firmware **"main.c"** é o programa padrão que estará presente na memória do PIC. Este firmware define quais portas serão utilizadas como entradas e saídas, atribuindo estas aos PORT's físicos do PIC18F4550, bem como aos demais periféricos necessários ao funcionamento para aplicação em questão. Este firmware também é responsável por chamar outras bibliotecas necessárias ao funcionamento do PIC como, por exemplo, a biblioteca padrão do PIC e também outras bibliotecas. que possuem diversas funções utilizadas no programa. As funções lógicas existentes dentro deste Firmware são as portas and, Or, inversor, exor, exnor, temporizador On-Delay e Off-Delay. O programa escrito em linguagem C neste Firmware encontra-se no Apêndice 4.

3.1.2 DESENVOLVIMENTO DO MECANISMO DE PROGRAMAÇÃO

O objetivo deste mecanismo é tornar a programação do CLP mais simples, tendo em vista que os métodos utilizados para programação do PIC não são triviais. O

mecanismo de programação é baseado em circuitos lógicos de eletrônica digital, linguagem de programação Ladder e linguagem de programação C, sendo esta última utilizada com maior aplicação na criação do firmware do controlador.

Normalmente os programas para microcontroladores são desenvolvidos em linguagem C, e tendo em vista as dificuldades já mencionadas, será utilizado um firmware padrão para o controlador, objeto do estudo, com funções já definidas. Estas funções quando desejadas são chamadas dentro do programa que será escrito em linguagem C. Este código pode ser escrito diretamente em linguagem C sem a necessidade de memórias lógicas e pode também ser obtido através do diagrama Ladder posteriormente sendo reescrito em linguagem C com os recursos do firmware desenvolvido.

Para o desenvolvimento do programa em Ladder, deve-ser considerar as proibições ou "em condições de operação" e possibilidades de comandos ou ordens externas, que devem ser obedecidas em caso da não proibição dos sensores ou lógica. As variáveis são tratadas duas a duas, sendo fundamentais as memórias aqui denominadas memórias lógicas (de uso interno do controlador) para se chegar ao programa que tratará as memórias aqui denominadas memórias físicas (sensores e atuadores externos).

Após o desenvolvimento do programa em Ladder é possível determinar todas as memórias que serão utilizadas no programa (lógicas e físicas), para assim declará-las para o controlador (programa em C após a aplicação do mecanismo).

Para ilustrar este mecanismo foi considerada a programação do controle de um portão (FIG. 11), que mostra o portão com seus respectivos atuadores.

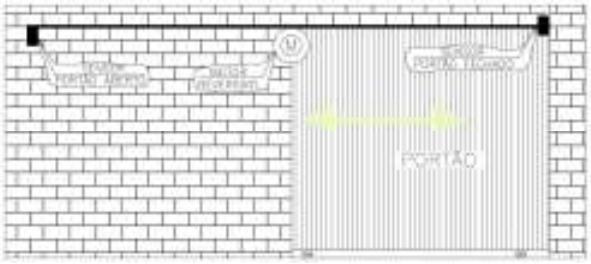


Figura 11 – portão com seus respectivos sensores e motor.

O desenho elétrico para acionamento de todo o sistema (FIG. 12).

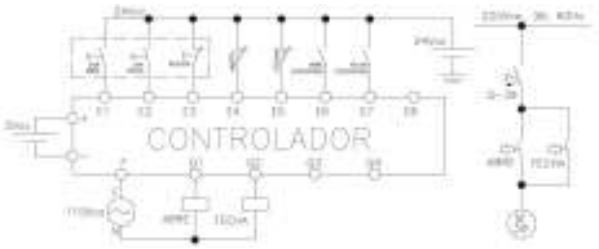


Figura 12 – Desenho elétrico básico

Na linguagem Ladder para o controle deste portão (FIG. 13), é possível se determinar todas as memórias necessárias caso o programa em C seja extraído do ladder.

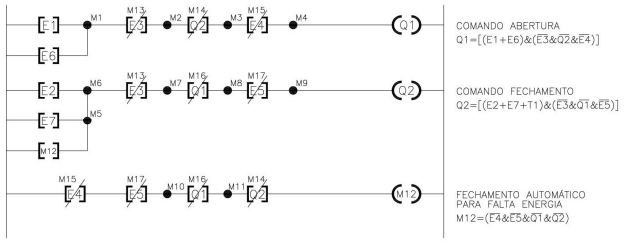


Figura 13 - Lógica em ladder e digital equivalente.

Baseando-se então no diagrama ladder para o desenvolvimento do programa têm-se as seguintes memórias utilizadas, sendo as memórias físicas entendidas como entradas digitais identificadas pela letra "E", memórias lógicas por "M", temporizadores

por “T” e atuadores entendidas como saídas digitais pela letra “Q”. Basicamente a conversão deste circuito para linguagem C é feita pela análise de circuito série/paralelo, onde os elementos em paralelo podem ser analisados como portas lógicas or, exor, exnor e os elementos em série analisados como portas And, Nand e Temporizador.

E1 - Botão liga abre.
E2 - Botão liga fecha.
E3 - Botão liga desliga.
E4 - Sensor portão aberto.
E5 - Sensor portão fechado.
E6 - Retroaviso real abrindo.
E7 - Retroaviso real fechando.

Memórias lógicas:

M1 ⟵(E6+E1)
M2 ⟵(M1 & M13)
M3 ⟵(M2 & M14)
M4 ⟵(M3 & M15)
M5 ⟵(E7+M12)
M6 ⟵(M5&E2)
M7 ⟵(M6&M13)
M8 ⟵(M7&M16)
M9 ⟵(M8&M17)
M10 ⟵(M15&M17)
M11 ⟵(M10&M16)
M12 ⟵(M11&M14)
M13 ⟵(/E3)
M14 ⟵(/Q2)
M15 ⟵(/E4)
M16 ⟵(/Q1)
M17 ⟵(/E6)

Atuadores físicos:

Q1 - Comando abre para motor.
Q2 - Comando fecha para motor.

Baseando-se então no mecanismo de programação, obtém-se o código “usuário.c” (Apêndice 5) que deverá ser compilado e gravado na memória do PIC18f4550, neste momento já entendido como PLC. O código gerado é reconhecido pelo firmware “main.c” com o nome de “usuário.c”.

3.1.3 VIABILIDADE

Foi realizada uma pesquisa de mercado para se obter o custo de um PLC. Na pesquisa foi considerado o PLC LOGO, de fabricação Siemens, sendo este PLC o modelo mais simples que este fabricante possui e para efeito de comparação, foram cotados em diversas lojas de eletrônica de Belo Horizonte os componentes considerados no projeto e chegando aos valores especificados a seguir.

O PLC LOGO foi cotado na empresa DW Material elétrico e seu custo gira em torno de R$ 480,00.

Os componentes para confecção das placas em versão protótipo ficam aproximadamente em torno de R$ 180,00 e as demais placas aproximadamente R$ 80,00.

4 CONCLUSÃO

Depois de encerradas as simulações com êxito foi constatado a funcionalidade do projeto em aplicações simples, como a do exemplo citado e ainda que exista a possibilidade de expansão dos módulos digitais de entrada e saída, é necessária apenas a revisão das rotinas de leitura e escrita dentro Firmware. Esta constatação aumenta ainda mais o custo benefício do controlador e reforça a ideia inicial da pesquisa, onde é possível se desenvolver um controlador de baixo custo e ainda torná-lo simples do ponto de vista de aplicação.

REFERÊNCIAS

CORTELETTI, D. LINGUAGEM LADDER para microcontroladores microchip PIC. Disponível em: <http://www.mecatronica.org.br/disciplinas/programacao/019/LDMICRO_TUTORIAL.pdf> Acesso em: 03 mar. 2011

DATASHEET 6-Pin DIP Optoisolators Transistor Output, 1995. – 4N25

DATASHEET 6-PIN DIP RANDOM-PHASE OPTOISOLATORS TRIAC DRIVER OUTPUT (250/400 VOLT PEAK, 2002) MOC3021

DATASHEET DM54LS240 / DM74LS240, DM54LS241/DM74LS241 Octal TRI-STATEÉ Buffers/Line Drivers/Line Receivers, abril 1992.

DATASHEET MAX232, MAX232I DUAL EIA-232 DRIVERS/RECEIVERS, outubro de 2002.

DATASHEET Microcontrolador PIC18F4550, Microchip Technology, 2009.

DATASHEET Octal D-Type Flip-Flop with TRI-STATEÉ Outputs 54F/74F374, maio de 1995.

GIMENEZ, S.P **Microcontroladores 8051.** 1. ed. São Paulo: Pearson Education do Brasil, 2002

IDOETA, Ivan V.; CAPUANO, Francisco G. Elementos **de eletrônica digital.** 40.ed. São Paulo: Érica, 2001

International Electrotechnical Commission. **IEC-61131-1:** Programmable controllers General information , 2003.

MARTINS, Henrique. R ; TORRES, Fernando. E **Apostila Curso de Sistemas Microcontrolados, baseado no PIC18F4550**. [S.L:s.n], 2011.

MIYADAIRA, A. N. **Microcontroladores Pic18, Aprenda e Programe em LINGUAGEM C.** 1. ed. São Paulo: Érica, 2009.

MORIMOTO, E. C. Processadores RISC X Processadores CISC. Disponivel em:<http://www.hardware.com.br/artigos/risc-cisc/> Acesso em: 10 maio. 2011.

PEREIRA, F **Microcontroladores HCS08, Teoria e Prática**. 1.ed. São Paulo: Érica, 2005.

PETRUZELLA, F. D. **Programmable Logic Controllers.** 3. ed. Boston: McGraw Hill, 2005.

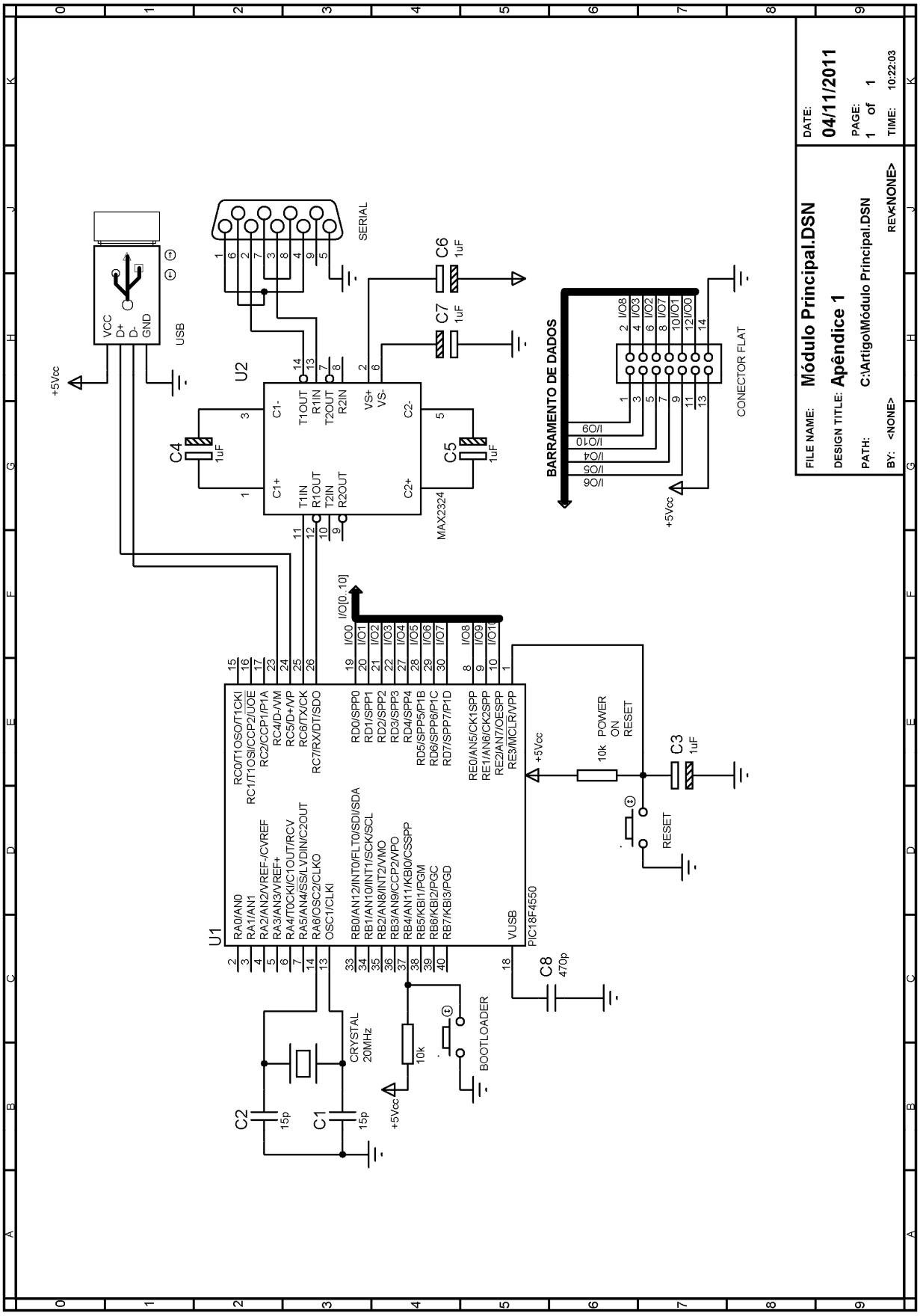
PLÍNIO, M. Introdução aos Controladores Lógicos Programáveis. Disponível em: <http://www.muriloplinio.eng.br/attachments/File/UNIFACS/AC/CLP_EXTRA.pdf > Acesso em: 03 mar. 2011

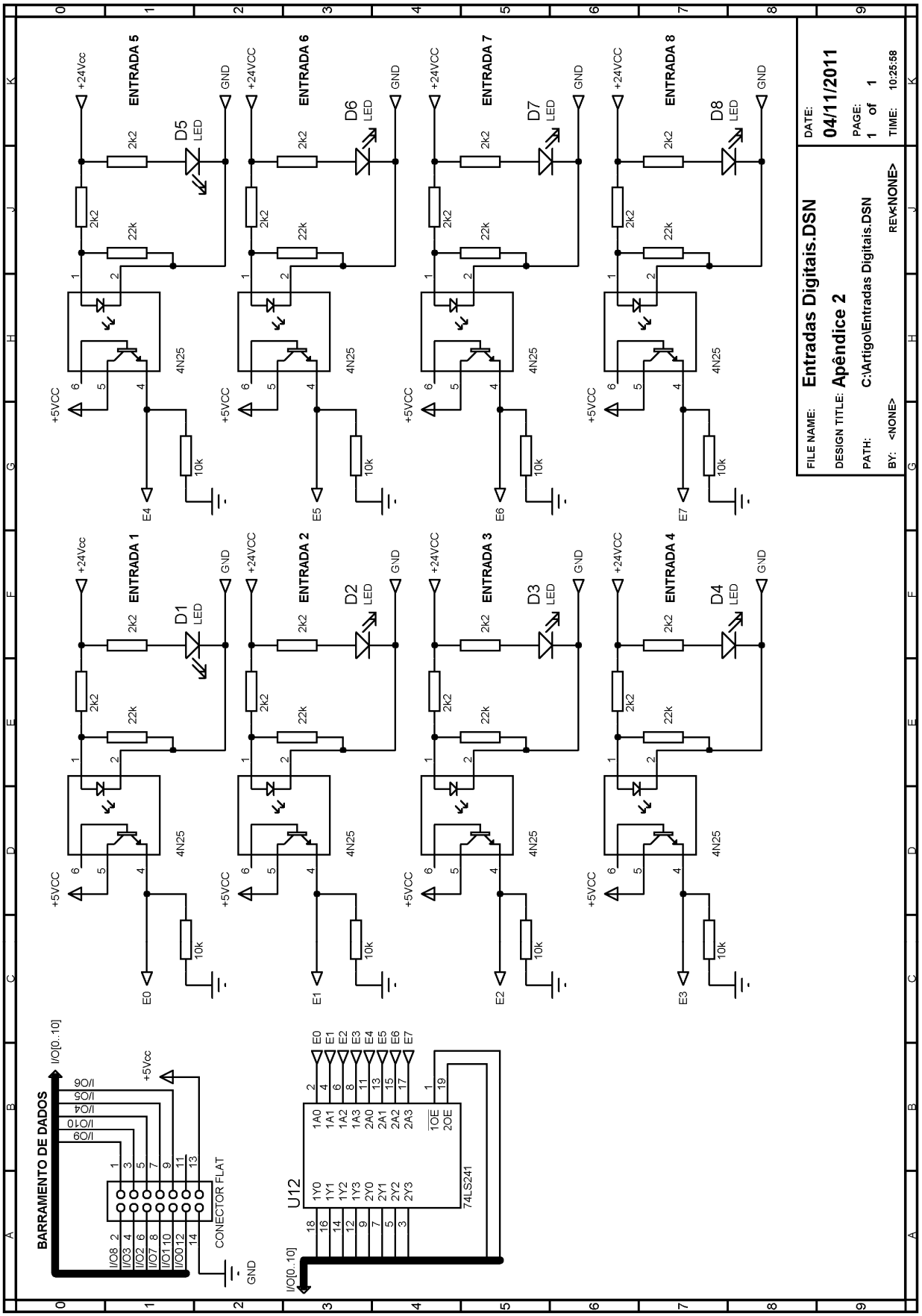
SOUZA, D. J. **Desbravando o PIC16F84.** 4. ed. São Paulo: Érica, 2000

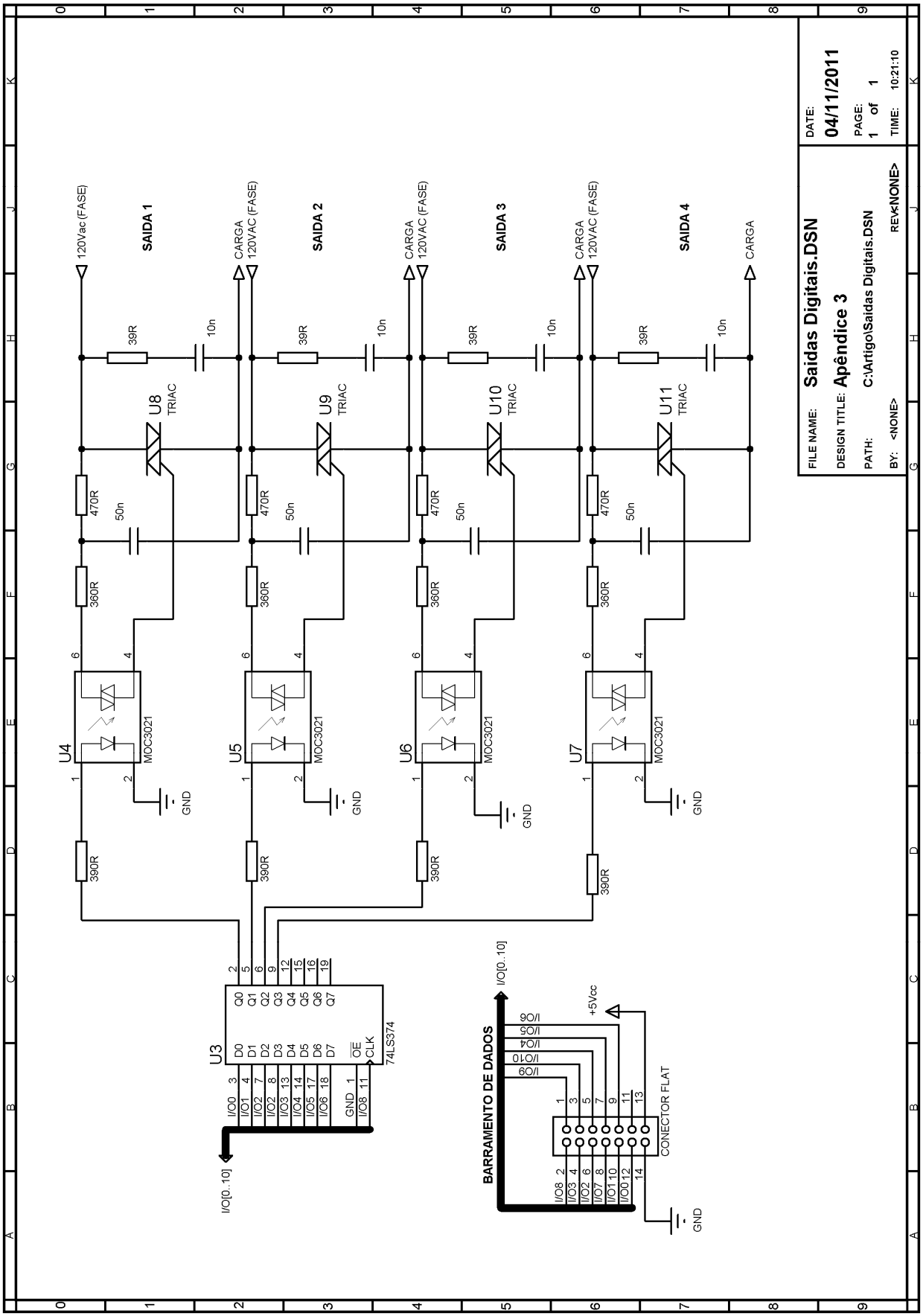
SOUZA, Vitor A**. A história e as diferenças entre um microcontrolador e um microprocessador**,2007. Disponivel em: <http://www.cerne-tec.com.br/artigos.htm>. Acesso em: 05 mar. 2011.

APÊNDICES

- Apêndice 1 - Modulo Principal.
- Apêndice 2 - Módulo de entrada Digital.
- Apêndice 3 - Módulo de Saída Digital.
- Apêndice 4 - Firmware.
- Apêndice 5 - Lógica do usuário.


U1
PIC18F4550
U2
MAX2324
SERIAL
USB
VCC
D+
D-
GND
+5Vcc
C1 15p
C2 15p
CRYSTAL 20MHz
C3 1uF
C4 1uF
C5 1uF
C6 1uF
C7 1uF
C8 470p
10k
10k POWER ON RESET
RESET
BOOTLOADER
BARRAMENTO DE DADOS
CONECTOR FLAT
I/O[0..10]
FILE NAME: Módulo Principal.DSN
DESIGN TITLE: Apêndice 1
PATH: C:\Artigo\Módulo Principal.DSN
BY: <NONE>
REV:<NONE>
DATE: 04/11/2011
PAGE: 1 of 1
TIME: 10:22:03


ENTRADA 1
ENTRADA 2
ENTRADA 3
ENTRADA 4
ENTRADA 5
ENTRADA 6
ENTRADA 7
ENTRADA 8
D1
D2
D3
D4
D5
D6
D7
D8
LED
+24Vcc
+24VCC
+5VCC
GND
2k2
22k
10k
4N25
E0
E1
E2
E3
E4
E5
E6
E7
BARRAMENTO DE DADOS
I/O[0..10]
I/O9
I/O10
I/O4
I/O5
I/O6
I/O8
I/O3
I/O2
I/O7
I/O1
I/O0
CONECTOR FLAT
U12
74LS241
1A0
1A1
1A2
1A3
2A0
2A1
2A2
2A3
1Y0
1Y1
1Y2
1Y3
2Y0
2Y1
2Y2
2Y3
1OE
2OE
FILE NAME: Entradas Digitais.DSN
DESIGN TITLE: Apêndice 2
PATH: C:\Artigo\Entradas Digitais.DSN
BY: <NONE>
REV:<NONE>
DATE: 04/11/2011
PAGE: 1 of 1
TIME: 10:25:58


120Vac (FASE)
SAIDA 1
CARGA 120VAC (FASE)
SAIDA 2
SAIDA 3
SAIDA 4
CARGA
39R
10n
470R
50n
360R
390R
U8 TRIAC
U9 TRIAC
U10 TRIAC
U11 TRIAC
U4 MOC3021
U5 MOC3021
U6 MOC3021
U7 MOC3021
GND
U3 74LS374
I/O[0..10]
BARRAMENTO DE DADOS
+5Vcc
CONECTOR FLAT
FILE NAME: Saidas Digitais.DSN
DESIGN TITLE: Apêndice 3
PATH: C:\Artigo\Saidas Digitais.DSN
BY: <NONE>
REV:<NONE>
DATE: 04/11/2011
PAGE: 1 of 1
TIME: 10:21:10

Apêndice 4 (1 de 2)- Firmware.TXT

```
//Projeto para publicação de Artigo técnico
//Titulo: CONTROLADOR LÓGICO PROGRAMÁVEL UTILIZANDO PIC18F4550
//FIRMWARE
#include <18f4550.h> // Inclui cabeçario padrão para o pic.
#use delay(clock=20000000) // Define diretriz para utilização de rotinas de tempo.
#use rs232(baud=9600, xmit=PIN_C6,rcv=PIN_C7) // Habilita funções Serial
short E1,E2,E3,E4,E5,E6,E7,E8,Q1,Q2,Q3,Q4; //Define I/Os físicos
short M1,M2,M3,M4,M5,M6,M7,M8,M9,M10,M11,M12,M13,M14,M15; //Define memorias
virtuais
short M16,M17,M18,M19,M20,M21,M22,M23,M24,M25,M26,M27,M28;//Define memorias
virtuais
int Sts_Entradas,Sts_Saidas; //Define variáveis para comunicação Serial
int tmr_ocupado, Tempo, seg; //Define variáveis para controle do temporizador.
int a[3],r1,r2,r3,r4; //Define variáveis para comunicação Serial
short porta_and(short O1, short O2)//Logica Porta AND
{
return O1&O2; //Faz operação logica entre argumentos recebidos,
//retorna para variavel que a chamou.
}
short inversor(short M0)//Logica Porta NOT
{
return ~M0;//Faz operação logica entre argumentos recebidos,
//retorna para variavel que a chamou.
}
short porta_or(short O1, short O2)//Logica Porta OR
{
return O1||O2;//Faz operação logica entre argumentos recebidos,
//retorna para variavel que a chamou.
}s
hort porta_exor(short O1, short O2)//Logica Porta EX-OR
{
return O1^O2;//Faz operação logica entre argumentos recebidos,
//retorna para variavel que a chamou.
}
short porta_exnor(short O1, short O2)//Logica Porta EX-NOR
{
return O1|O2;//Faz operação logica entre argumentos recebidos,
//retorna para variavel que a chamou.
}
#include <logica_usuario.h>
#INT_TIMER0 //Rotina de interrupção do TMR0.
void tempo(int conta) //Função tempo com variavel local conta
{
if(conta==610) // Testa variável conta, caso "conta=610" faça, se não, pule.
{
if(seg==0) // Testa variavel seg, caso "seg=0" faça, se não, pule.
{
Q1=0; // Executa logica
tmr_ocupado=1; //Libera temporizador
}
conta=0; //Zera variável conta.
seg--; //Decrementa variavel segundo.
}
conta++; //Incrementa variável conta.
} //fim da rotina de interrupção do TMR1.
void main()
{
enable_interrupts(GLOBAL); //Habilita interrupções globais
```

```
enable_interrupts(INT_TIMER0); //Habilita interrupção do TMR0
output_bit(PIN_E0,0); //Inicializa pinos de controle
output_bit(PIN_E1,0); //dos modulos de entradas e
output_bit(PIN_E2,1); //saídas.
while(1) //Ciclo de Scan Infinito
{
//Leitura das Entradas
output_bit(PIN_E1,1); //Habilita entradas 1Y do ls241
output_bit(PIN_E2,0); //Habilita entradas 2Y do ls241
E1=input(!PIN_D0); //Lê RD0 e armazena em E1
E2=input(!PIN_D1); //Lê RD1 e armazena em E2
E3=input(!PIN_D2); //Lê RD2 e armazena em E3
E4=input(!PIN_D3); //Lê RD3 e armazena em E4
E5=input(!PIN_D4); //Lê RD4 e armazena em E5
E6=input(!PIN_D5); //Lê RD5 e armazena em E6
E7=input(!PIN_D6); //Lê RD6 e armazena em E7
E8=input(!PIN_D7); //Lê RD7 e armazena em E8
Sts_Entradas=!input_d(); //Armazena status das entradas para enviar a serial
output_bit(PIN_E1,0); //Desabilita entradas 1Y do ls241
output_bit(PIN_E2,1); //Desabilita entradas 2Y do ls241
printf("%c",Sts_Entradas); //Envia status das entradas para a serial.
//Final da leitura das entradas
//Início da lógica do usuário
logica_usuario();
//Fim da lógica do usuário
//Inicializa atualização das saídas.
output_bit(PIN_D0,Q1); //Carrega em RD0 o valor de Q1
output_bit(PIN_D1,Q2); //Carrega em RD1 o valor de Q2
output_bit(PIN_D2,Q3); //Carrega em RD2 o valor de Q3
output_bit(PIN_D3,Q4); //Carrega em RD3 o valor de Q4
output_bit(PIN_D4,0); //Limpa bit não utilizado.
output_bit(PIN_D5,0); //Limpa bit não utilizado.
output_bit(PIN_D6,0); //Limpa bit não utilizado.
output_bit(PIN_D7,0); //Limpa bit não utilizado.
output_bit(PIN_E0,1); //Executa um pulso de
delay_us(0.06); //60ns no pino de clock
output_bit(PIN_E0,0); //do ls374 para carregar saídas.
```

Apêndice 4 (2 de 2)- - Firmware.TXT

```
//Final da atualização das saídas.
//Prepara atualização das saídas na porta serial.
a[0]=Q1; //Preenche a posição 0 do vetor com o valor de Q1
a[1]=Q2; //Preenche a posição 1 do vetor com o valor de Q2
a[2]=Q3; //Preenche a posição 2 do vetor com o valor de Q3
a[3]=Q4; //Preenche a posição 3 do vetor com o valor de Q4
R1=a[0]&0x01; //R1=Resultado da operação logica and com a posição 0 do vetor.
R2=a[1]&0x01; //R2=Resultado da operação logica and com a posição 1 do vetor.
if(R2==1) //Se R2 for 1
{R2=2;} //R2 vale 2 (0010b)
else //Senão
{R2=0;} //R2 vale 0 (0000b)
R3=a[2]&0x01; //R3=Resultado da operação logica and com a posição 2 do vetor.
if(R3==1) //Se R3 for 1
{R3=4;} //R3 vale 2 (0100b)
else //Senão
{R3=0;} //R3 vale 0 (0000b)
R4=a[2]&0x01; //R4=Resultado da operação logica and com a posição 3 do vetor.
if(R4==1) //Se R4 for 1
{R4=8;} //R4 vale 2 (1000b)
```

```
else //Senão
{R4=0;} //R4 vale 0 (0000b)
Sts_Saidas=R4+R3+R2+R1;
printf("%c",Sts_Saidas); //Envia status das saídas para a serial.
}
}
```

Apêndice 5 - Lógica do usuário.txt

```
//UNI-BH - Centro Universitário de Belo Horizonte
//Projeto para publicação de Artigo técnico
//Titulo: CONTROLADOR LÓGICO PROGRAMÁVEL UTILIZANDO PIC18F4550
//LOGICA DO USUÁRIO
void logica_usuario()
{
//DESCRIÇÃO DE SINAIS
//E1=botão abre
//E2=botão fecha
//E3=botão desliga
//E4=sensor portão aberto
//E5=sensor portão fechado
//Q1=saida abre portão
//Q2=saida fecha portão
//INICIA LOGICA DO USUÁRIO
M13=inversor(E3);//logica de segurança do botão desliga
M15=inversor(E4);//logica de segurança do sensor portão aberto
M17=inversor(E5);//logica de segurança do sensor portão fechado
M16=inversor(Q1);//intertravamento portão abrindo
M14=inversor(Q2);//intertravamento portão fechando
//logica de abertura
M1=porta_or(E1,E6); //botão abrir portão ou selo abrindo "real"
M2=porta_and(M1,M13);//logica de segurança botão desliga
M3=porta_and(M2,M14);//logica de segurança portão fechando
M4=porta_and(M3,M15);//logica de segurança portão totalmente aberto
Q1=M4;//comando abre portão
//logica de fechamento
M5=porta_or(M12,E7); //temporizador ou selo fechando "real"
M6=porta_or(M5,E2); //botão fechar portão
M7=porta_and(M6,M13);//logica de segurança botão desliga
M8=porta_and(M7,M16);//logica de segurança portão abrindo
M9=porta_and(M8,M17);//logica de segurança portão totalmente fechado
M10=porta_and(M15,M17);//logica de fechamento automático-posição intermediaria
aberto fechado
M11=porta_and(M10,M16);//logica de fechamento automático-comando abre portão
desligado
M12=porta_and(M11,M14);//logica de fechamento automático-comando fecha portão
desligado
Q2=M9;
//FINAL DA LÓGICA DO USUÁRIO
}
```