# Escola Politécnica da Universidade de São Paulo

## Departamento de Engenharia de Energia e Automação Elétricas

# Projeto e Construção de um Controlador Microprocessado de Ponte de Tiristores


**Aluno:** Daniel Mello Moraes Gualberto

**Orientadores:** Clovis Goldemberg
Walter Kaiser

# Índice

## Sumário

No presente trabalho, desenvolveu-se, partindo de um arranjo microcontrolador - PLL (phase-locked loop), um sistema capaz de realizar o disparo de pontes tanto de 6 como 12 tiristores, dotado de lógicas de controle sofisticadas como double pulse, phase-back, ponte avante/reversa, angulo máximo e mínimo, e ainda capaz de compensar a não linearidade entre entrada e saída realizando a operação arco-cosseno. Os sinais de comando podem ser tanto analógicos quanto digitais, permitindo, portanto, a interligação a qualquer sistema de controle.

## Introdução

Pontes de tiristores podem ser controladas através de circuitos analógicos convencionais, destacando-se aqueles que fazem uso do TCA785, que é um circuito integrado desenvolvido pela Siemens e que alcançou grande aceitação. Este tipo de circuito, porem, sofre algumas limitações, principalmente a baixa imunidade a ruído e a resposta não linear entre entrada e saída ( Vdd = K.cos $\alpha$). Complementando, a montagem de lógicas mais sofisticadas como phase-back , limitação de angulo máximo e mínimo e double pulse exigiria a adição de grande quantidade de circuitos extras, aumentando a complexidade (e o custo) desse tipo de solução.

No presente trabalho, desenvolveu-se, partindo de um arranjo microcontrolador-PLL (phase-locked loop), um sistema capaz de realizar o disparo de pontes tanto de 6 como 12 tiristores, dotado de lógicas de controle sofisticadas como as citadas anteriormente, e ainda capaz de compensar a não linearidade entre entrada e saída realizando a operação arco-cosseno.

Foi priorizado o uso de poucos componentes externos a fim de reduzir o custo. Como resultado, um circuito básico pôde ser montado em uma placa de 10x8cm face simples. Alem disso, a solução digital aumenta a precisão do sistema, uma vez que esses circuitos são menos susceptíveis a ruídos e variação de parâmetros. Como um último beneficio, ainda existe a possibilidade de implementar o controle totalmente digital do disparo, facilitando a interface com outros sistemas digitais.

## Especificação funcional

Existem 4 chaves que estabelecem a configuração básica da placa. Estas chaves não devem ser alteradas durante o funcionamento, e só são lidas na inicialização. São elas:

| *Chave* | *Função* |
|---|---|
| **1**<br>**HSER** | **OFF** – Sinal de comando analógico.<br>**ON** – Sinal de comando digital. |
| **2**<br>**PUL12** | **OFF** – Ponte de 6 pulsos.<br>**ON** – Ponte de 12 pulsos. Neste caso, ignora a posição da chave 3. |
| **3**<br>**PREV** | **OFF** – Habilita disparo tanto da ponte avante quanto da ponte reversa.<br>**ON** – Desabilita o disparo da ponte reversa. |
| **4**<br>**ACOS** | **OFF** – Lei de disparo linear.<br>**ON** – Lei de disparo arco-cosseno. |

Existem 4 leds/saídas digitais que fornecem indicação do funcionamento do sistema:

| *Led* | *Estado* |
|---|---|
| **LD1** | Placa OK. Indica que o processador passou pela inicialização. |
| **LD2** | WDT. Acende quando o Watch Dog Timer for ativado. O sistema ficará em estado de espera até ser resetado. |
| **LD3** | Ponte A sendo disparada. No caso de 6 pulsos, esta é a ponte avante. |
| **LD4** | Ponte B sendo disparada. No caso de 6 pulsos, esta é a ponte reversa. |

Existem ainda 5 pinos para entrada de sinais de controle e um para saída:

| *Pino* | *Função* |
|---|---|
| **RST** | Reset do sistema. Entrada com resistor de pull-up.<br>Active low. |
| **AN1** | Entrada analógica. |
| **EN** | Habilita disparo. Entrada com resistor de pull-up.<br>Low - Disparo habilitado.<br>High – Disparo desabilitado. |
| **AV** | Ponte avante/reversa. Entrada com resistor de pull-up.<br>Low – Habilita o disparo da ponte reversa.<br>High – Habilita o disparo da ponte reversa. |
| **PB** | Phase-Back. Entrada com resistor de pull-up.<br>Low – Força condição de ângulo máximo.<br>High – Disparo normal. |
| **CNV** | Conversão. Saída TTL.<br>Pulso de 200ns que indica a realização de uma leitura do angulo de disparo. |

OBS – O pino AV estabelece apenas um permissivo, sendo que a habilitação efetiva é dada pelo pino EN. Obviamente, este pino só terá validade quando o sistema estiver configurado para pontes de 6 tiristores, e a ponte reversa estiver habilitada nas chaves de configuração.

Os pulsos de disparo são apresentados em 12 pinos, denominados de **T1** a **T12**. Estes pinos são saídas em níveis TTL (0V/5V), e podem absorver/fornecer até 20mA cada.

A comunicação serial é realizada através de 5 pinos de I/O:

| *Pino* | *Função* |
|---|---|
| **SDI** | Entrada de dados seriais (Serial Data In). |
| **SDO** | Saída de dados seriais(Serial Data Out). |
| **SCK** | Serial clock. |

| *Pino* | *Função* |
|---|---|
| **RDT** | Saída - Requisição de dados (Request data). |
| **DTR** | Entrada – Dados prontos (Data Terminal Ready). |

# Funcionamento:

### Chaves

Conforme já foi adiantado, as chaves constituem um meio de configurar a placa antes do início do funcionamento. Seu valor não é conferido durante o funcionamento. São os controles de maior prioridade. Por exemplo, caso as chaves indiquem que a ponte reversa está desabilitada e o sinal pino AV indique o disparo da ponte reversa, a ponte reversa não será disparada, já que as chaves tem prioridade sobre pinos de I/O.

### Pinos I/O

Os pinos de I/O funcionam de maneira diversificada. O Pino **EN** é, dentre eles, o de maior prioridade. Qualquer informação passada por outro pino que se oponha a este será desprezada.

O pino **AV** só é lido enquanto os pulsos estão inibidos. A lógica implementada impede que se comute entre as pontes avante e reversa enquanto os disparos estão ativos. Caso se deseje comutar para a ponte reversa enquanto a ponte avante estiver sendo disparada, deve-se antes inibir os pulsos, alterar o estado do pino **AV** e, só então, habilitar novamente o disparo.

O pino **PB**, quando em Low, força o disparo utilizando o ângulo máximo, isto se os disparos estiverem habilitados. Este comando é lido mesmo durante os disparos.

O pino **CNV** apresenta um pulso indicando o início de uma conversão A/D, e é utilizado apenas para depuração.

### Entrada analógica

Caso o sistema esteja configurado para entrada analógica, neste pino deve ser apresentado um sinal entre 0V e 4,9V que controlará o ângulo de

disparo da ponte. Este ângulo será calculado de maneira diversa, dependendo do estado da chave 4. As curvas ângulo de disparo x tensão estão representadas abaixo:

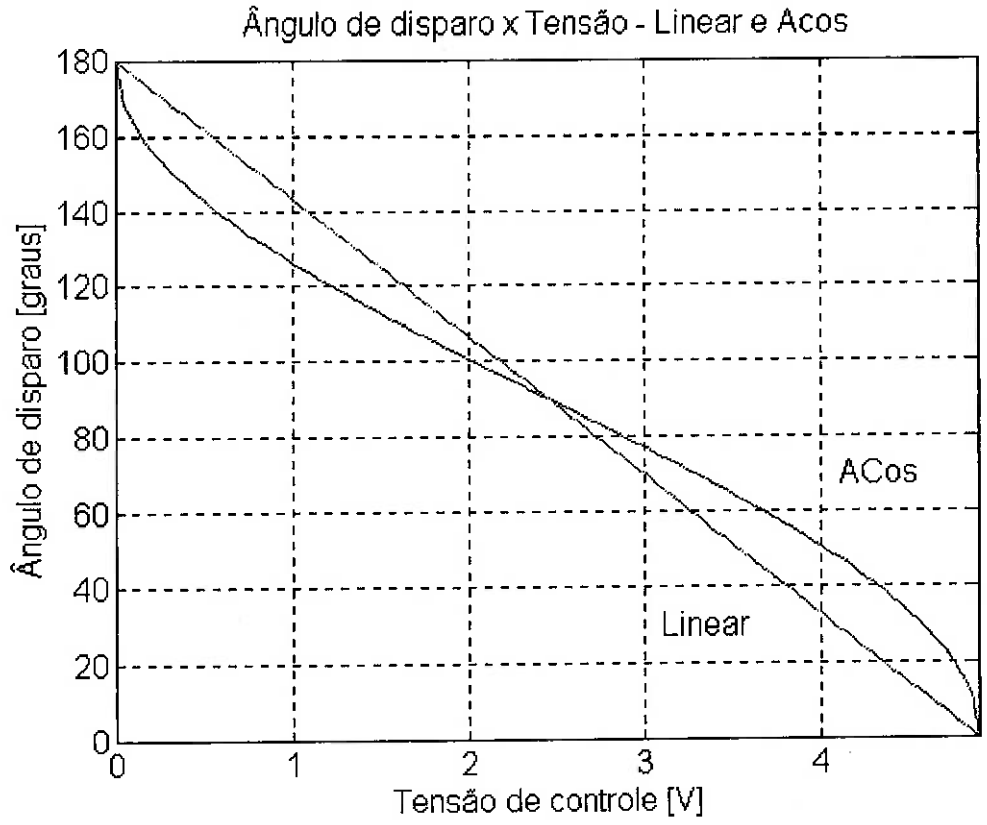



*Ilustração 1*

**Entrada serial**

Por fim, o disparo pode ser controlado digitalmente, por meio de uma entrada serial síncrona de alta velocidade (5Mhz). Neste caso, são transferidos dois bytes. Segue-se a descrição dos bits:

| ***Bit*** | ***Função*** |
|---|---|
| **0 ... 9** | Ângulo de disparo discretizado em 10 bits.<br>Bit 0: LSB<br>Bit 9: MSB |
| **10** | Não é utilizado. |
| **11** | Não é utilizado. |

| *Bit* | *Função* |
|---|---|
| **12** | 1 – A informação dada pelos bits (0..9) deve ser tratada como sendo o $\alpha$max.<br>0 – A informação dada pelos bits (0..9) deve ser tratada como angulo de disparo normal. |
| **13** | 1 – A informação dada pelos bits (0..9) deve ser tratada como sendo $\alpha$min.<br>0 – A informação dada pelos bits (0..9) deve ser tratada como angulo de disparo normal. |
| **14** | Não é utilizado. |
| **15** | Não é utilizado. |

A interface serial funciona da seguinte forma:

Quando os pulsos estão inibidos (Estado de espera):

O pino DTR é continuamente verificado. Caso o dispositivo controlador deseje enviar algum parâmetro, deve elevar o nível deste pino (High). Assim que DTR for detectado, o processador ativará a recepção de 2 bytes. A informação recebida só será considerada caso seja $\alpha$max ou $\alpha$min.

Este modo de funcionamento será permitido mesmo quando o sistema estiver configurado para entrada analógica.

Quando os pulsos estão habilitados:

Quando o processador estiver pronto para receber uma nova informação, O pino RDT será elevado (high). Pelos 4us subsequentes, o pino DTR será verificado. O dispositivo controlador deverá elevar, neste período, este pino para indicar que há nova informação disponível. Assim que DTR for detectado, o processador ativará a recepção de 2 bytes. A informação recebida será considerada tanto como comando, seja $\alpha$max ou $\alpha$min, ou como angulo de disparo normal. Segue o fluxograma:

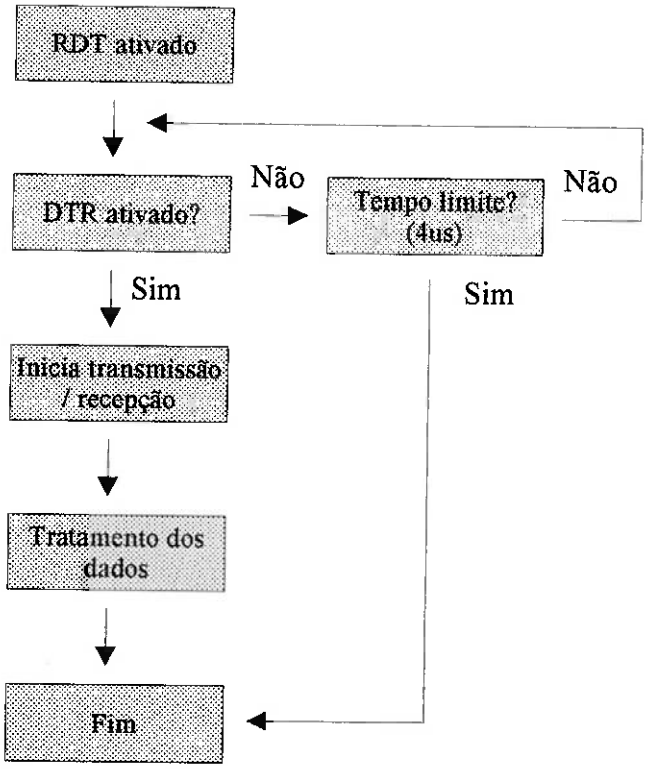



*Ilustração 2*

## Hardware

O sistema completo pode ser descrito sucintamente pelo diagrama de blocos abaixo:

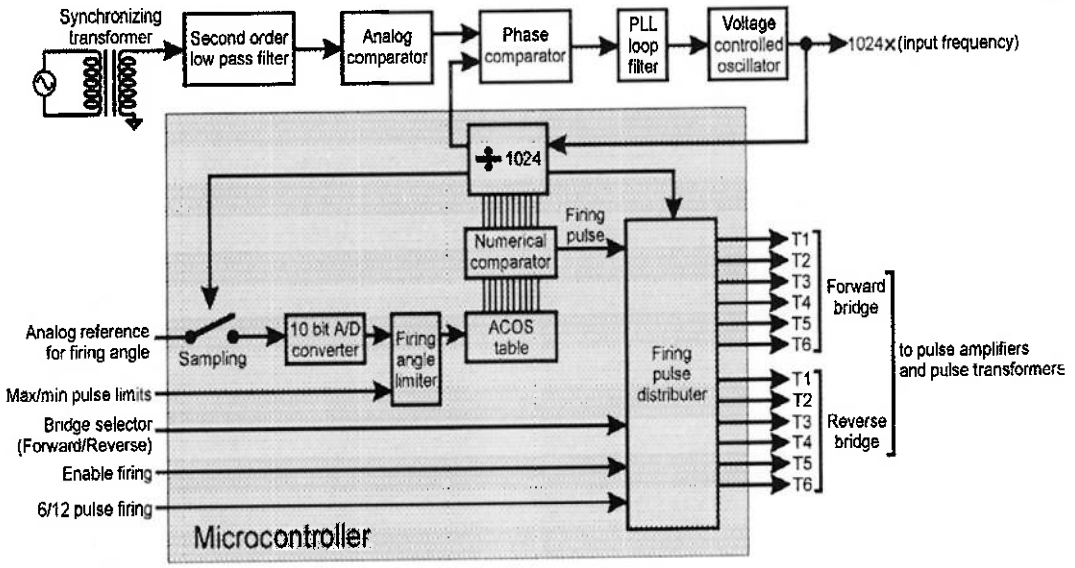



*Ilustração 3*

solução completa, para o efetivo disparo de uma ponte de tiristores envolve uma parte analógica (filtro, comparador, PLL), uma digital (Microcontrolador) e drivers de saída (amplificadores e transformadores de pulso). O presente trabalho enfoca a parte digital desse sistema, mais precisamente o desenvolvimento e depuração do software do microcontrolador.

Foi utilizado o microcontrolador PIC16C774 da Microchip Tecnology Inc. Este dispositivo apresenta como principais características que motivaram sua escolha:

- Ciclo de instrução de 200ns (a 20Mhz).
- Contador de 16Bits.
- Conversor A/D de 12Bits.
- Porta serial síncrona (SSP).
- 34 pinos de I/O.

No Anexo II, estão as páginas introdutórias do catálogo deste componente.

No esquemático seguinte vê-se a versão final do hardware desenvolvido, com a alocação dos pinos de I/O para cada função, e dispositivos auxiliares (resistores de pull-

up, leds, capacitores de desacoplamento e cristal oscilador) diretamente ligados ao processador. No anexo III há uma impressão ampliada deste esquemático.

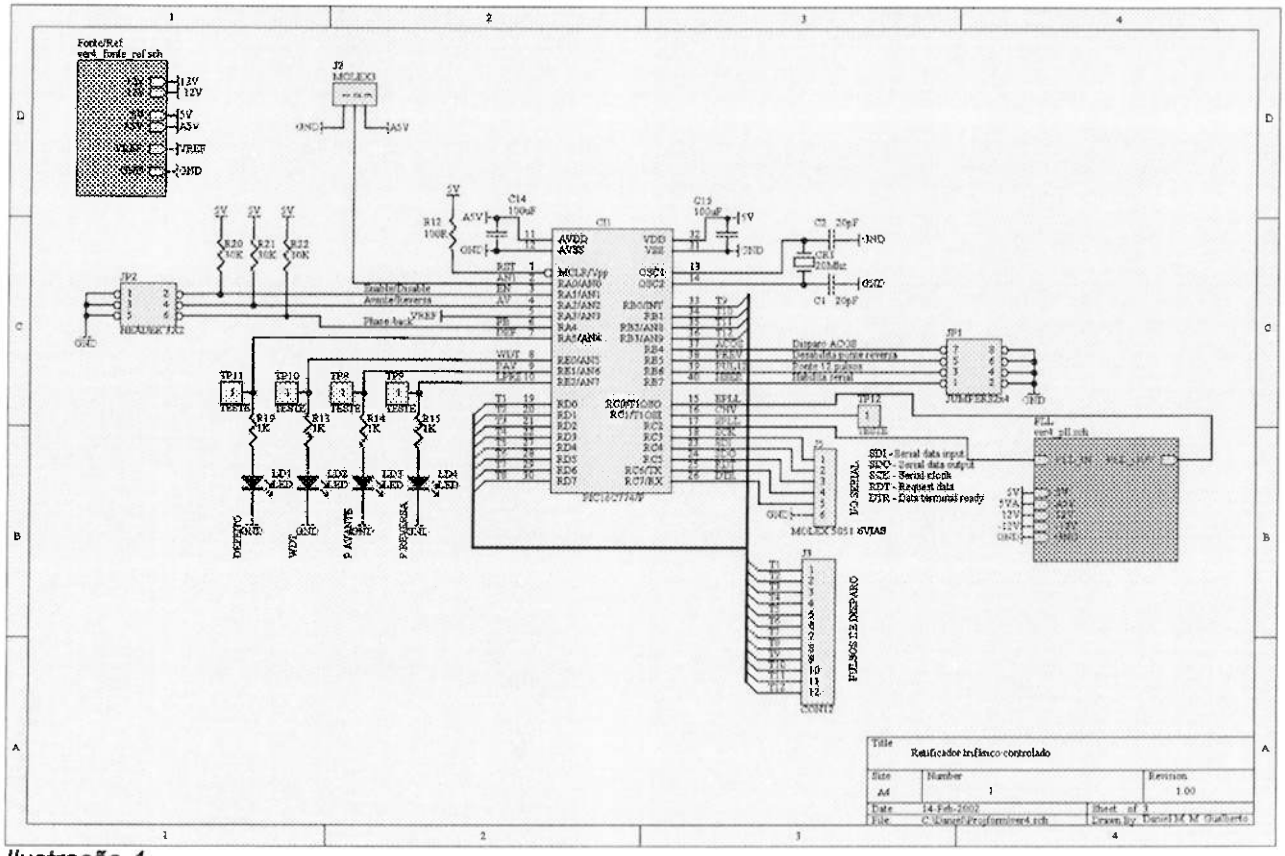



*Ilustração 4*

No esquemático acima existem dois blocos auxiliares: **Fonte/Ref** e **PLL**. Para fins de teste funcional, estes blocos, que não fazem parte do enfoque do trabalho, foram implementados de maneira simplificada, porem suficiente para os testes e depuração do software envolvido. Ainda no anexo III podem ser vistas descrições desses dois blocos.

## Software

### Introdução

O Software foi desenvolvido na linguagem Assembler, utilizando o compilador disponibilizado pelo fabricante. O uso de linguagens de mais alto nível, como o C, foi descartada pois é preciso ter controle absoluto dos tempos gastos pelas rotinas a fim de otimizar o desempenho.

O PIC utiliza a arquitetura RISC (Reduced Instruction Set Computer), com apenas 35 instruções. Todas as instruções gastam apenas um ciclo, com exceção de saltos (branches), que gastam dois ciclos.

### Inicialização

A primeira etapa do programa é configurar os pinos de I/O e periféricos disponíveis no CHIP. A configuração é realizada por meio de vários registradores em RAM. No Apêndice IV estão os valores de todos os registradores que são modificados na inicialização, com uma breve descrição de sua função. Maiores informações podem ser obtidas no manual do processador.

### Divisão dos pulsos por 1024

Conforme o diagrama de blocos apresentado na secção anterior, uma das funções do microprocessador é dividir a frequência de saída do PLL por 1024. Esta função é executada de forma semi-autônoma, devido a recursos de hardware embutidos no chip.

A saída do PLL é ligada à entrada do Timer1, que é um contador de 16Bits embutido no hardware. Este timer possui um módulo que, após cada incremento, compara o valor do contador (TMR1H e TMR1L) com um registro de referência (CCPR1H e CCPR1L). Uma vez que os registradores são iguais, um pino de I/O muda de estado (no caso, RC2) e a rotina de interrupção é ativada (Caso GIE, Global Interrupt Enable bit, esteja setado). Na rotina de interrupção determinamos o estado do pino de saída na próxima ativação, e resetamos o contador do Timer1.

A lógica acima permite que a divisão de frequência seja realizada corretamente e com baixo uso de CPU, porem existe existe uma limitação intrínseca a ela: O contador contará apenas de 0 a 511 (9 bits). O décimo bit da contagem, porem, poderá ser extraído do registrador que controla o estado do pino de saída. Sabemos que se o pino

de saída estiver em 1, estamos entre 0° e 180°, ou 0 a 511 e, caso esteja em 0, estamos entre 180° e 360°, ou 512 a 1024. Foi criada uma macro capaz de montar um valor de 10bits a partir do valor do contador e do estado do controle do bit de saída.

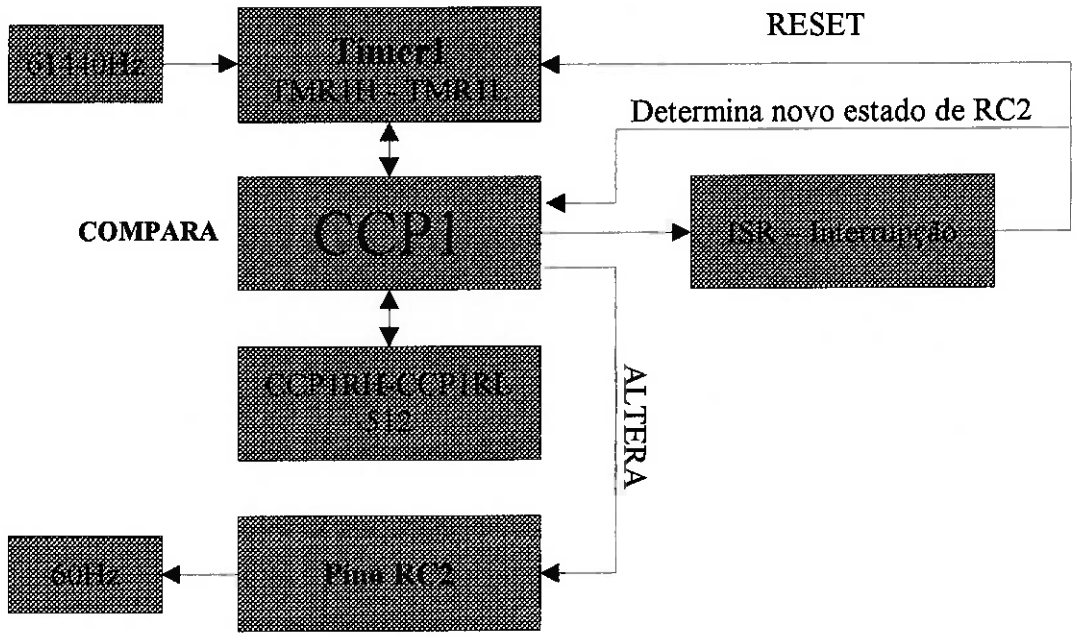



*Ilustração 5*

**Lógica de disparo**

O sistema de disparo trabalha com 1024 níveis discretos para representar 180° elétricos. Com isso, temos uma resolução de aproximadamente 1/5 de grau (0.18°).

Para a conversão de um ângulo $\alpha$ para a representação inerna ao processador n, utilizamos a formula:

$$n = \frac{\alpha \cdot 1024}{180}$$

Na tabela abaixo, temos a representação interna de alguns ângulos importantes (decimal e hexadecimal):

| $\alpha$ | $n$ | |
|---|---|---|
| 0° | 0 | 000h |
| 1° | 6 | 006h |
| 10° | 57 | 039h |
| 30° | 171 | 0ABh |
| 60° | 341 | 155h |
| 90° | 512 | 200h |

| $\alpha$ | $n$ | |
|---|---|---|
| 120° | 683 | 2ABh |
| 150° | 853 | 355h |
| 170° | 967 | 3C7h |
| 180° | 1024 | 400h |
| 210° | 1195 | 4ABh |
| 240° | 1365 | 555h |
| 270° | 1536 | 600h |
| 300° | 1707 | 6ABh |
| 330° | 1877 | 755h |
| 360° | 2048 | 800h |

Um ponto importante a ser explicitado é o mecanismo que torna possível obtermos 10bits de resolução para 180° uma vez que o contador apresenta apenas 9Bits para 180° (ou 10Bits para 360°).

Para aumentar a resolução do sistema, realizamos o disparo hora sincronizado com a borda de subida do clock, hora com a borda de descida. Criamos, assim, um décimo bit. Esse método está ilustrado na figura abaixo:

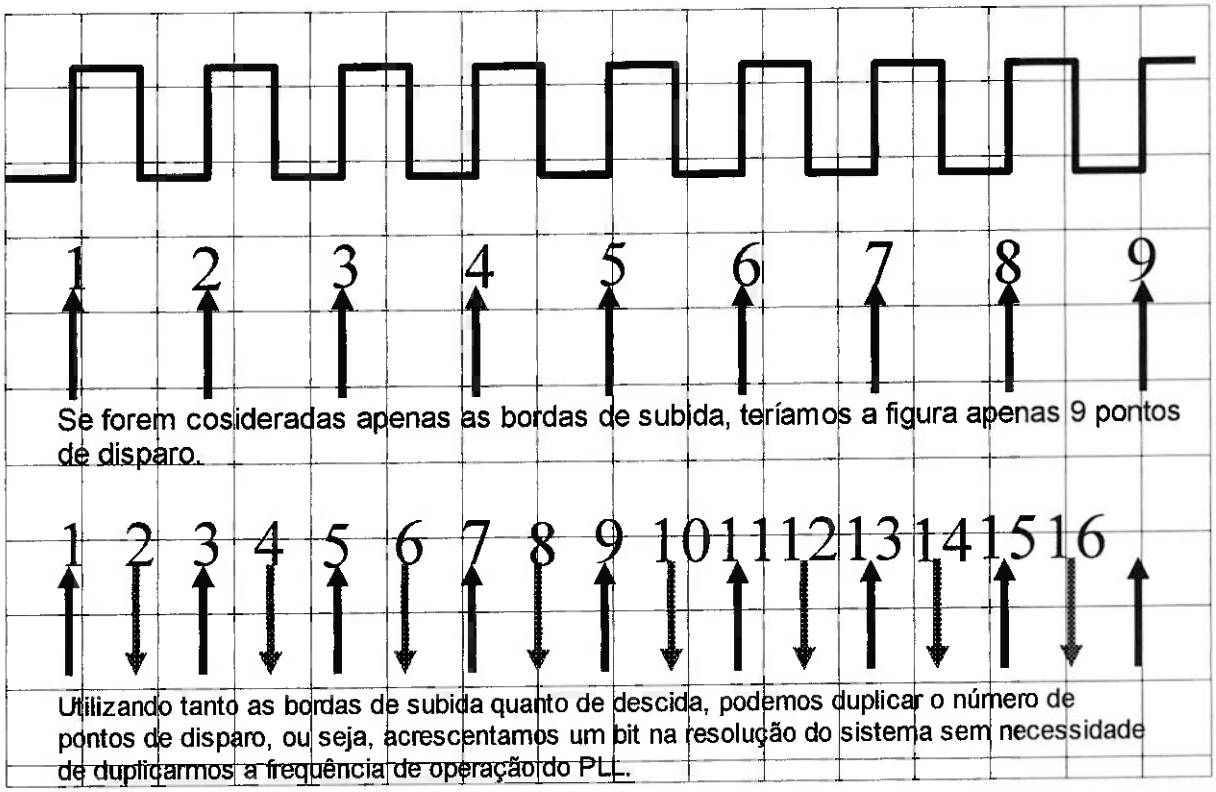



*Ilustração 6*

**Loop de amostragem e disparo**

Existem três subrotinas principais que realizam o disparo, são elas:

- init_av_6 – Dispara a ponte avante (6 pulsos).
- init_rev_6 – Dispara a ponte reversa (6 pulsos).
- init_p12 - Dispara a ponte completa de 12 pulsos.

Cada uma dessas subrotinas é composta de um conjunto de loops, um para cada disparo. Assim, init_av_6 possui 6 loops, init_rev_6 mais 6 loops, e init_p12, 12 loops. Abaixo temos um esquema do funcionamento de init_av_6. As outras subrotinas possuem estruturas semelhantes:

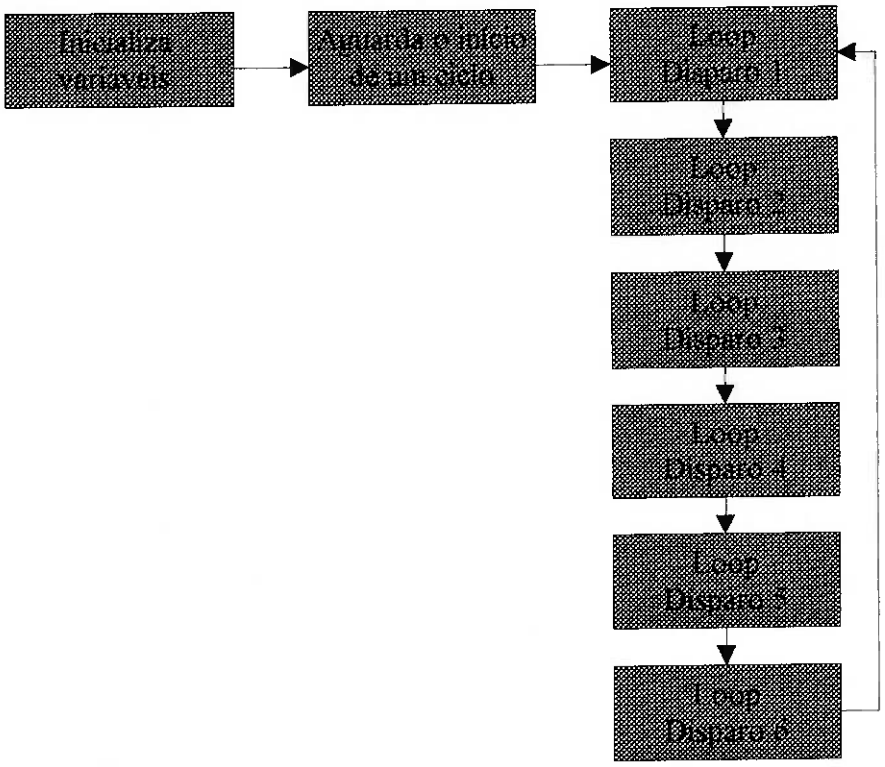



*Ilustração 7*

Esses loops são muito semelhantes, variando apenas alguns parâmetros como os pinos que serão disparados. Abaixo segue a descrição básica desses loops:

1. Inicialização – Determinamos os pinos que serão disparados.
2. Aguarda amostragem – Espera até o momento de realizar a leitura da entrada. A entrada é lida 32 vezes por ciclo o que, em 60Hz, nos dá uma amostra a cada 520us.
3. Amostragem – Lê um novo valor para o ângulo de disparo. A este valor são

aplicados os limites Amax e Amin, e, caso esta opção esteja ativa, é calculado o Arco-cosseno da entrada.

4. Cálculo do disparo – Com o valor lido, calcula alguns parâmetros necessários para determinar o momento do disparo. Modifica variáveis.
5. Decisão – Com base nos parâmetros calculados acima, decide se iremos para a rotina de disparo ou aguardamos uma nova amostra. Caso o momento do disparo esteja longe o suficiente, volta ao passo 2.
6. Aguarda disparo – Aguarda o momento do disparo e dispara os tiristores determinados na inicialização do loop. O pulso de disparo têm largura de 50us.
7. Vai para o próximo loop.

Segue o diagrama de blocos:

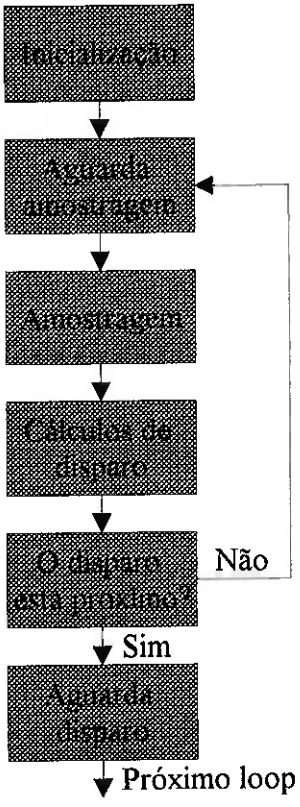



*Ilustração 8*

**Transformação Arco-cosseno**

O microcontrolador em uso não possui capacidade de processamento suficiente para realizar a transformação arco-cosseno em tempo real. Isso já era sabido e a solução é bastante usual: calcula-se de antemão uma tabela de conversão que, uma vez

armazenada em ROM, basta ser consultada.

O armazenamento desta tabela, porem, ocuparia muito espaço:

São 1024 possíveis valores de entrada, cada um com 10Bits. Para armazenar 10Bits, são necessários 2Bytes. Ou seja, a tabela toda ocuparia 2x1024= 2048Bytes= 2KBytes.

O PIC16C774 possui 4K Words (de 14Bits) de ROM e, como para implementar tabelas é necessário usar uma Word inteira para cada Byte de informação, a tabela ocuparia metade de toda a memória disponível. Alem disso, tabelas só podem ser armazenadas em blocos de 255Bytes. Estes fatos não inviabilizam totalmente o uso de uma tabela completa, porem trariam muita complexidade para a pesquisa na mesma.

Foi utilizada uma solução mais prática que resultou em uma tabela de 512 Bytes, ou seja, ¼ do tamanho do método convencional. Observando um gráfico no qual temos tanto a saída linear como a saída arco-cosseno, (ver figura na secção **Especificação funcional**) vemos que:

1. A diferença entre a saída linear e a saída arco-cosseno não é grande. Esta diferença pode ser armazenada em menos de 8 bits. Se, ao invés de armazenarmos o valor inteiro, armazenarmos apenas essa diferença, não são necessários 2 Bytes para cada amostra, mas apenas 1.
2. A diferença entre a saída linear e a saída arco-cosseno é simétrica e invertida com relação à sua metade (A função saída linear – saída arco-cosseno é impar!). Assim, só precisamos armazenar metade da tabela.

No gráfico seguinte vê-se que de 0° a 90°, a diferença (Acos-Linear) é positiva e inferior a 30°, e, de 90° a 180°, a diferença é simétrica com relação à primeira, porem negativa.

A tabela completa pode ser vista no Anexo I, onde temos o código fonte comentado.

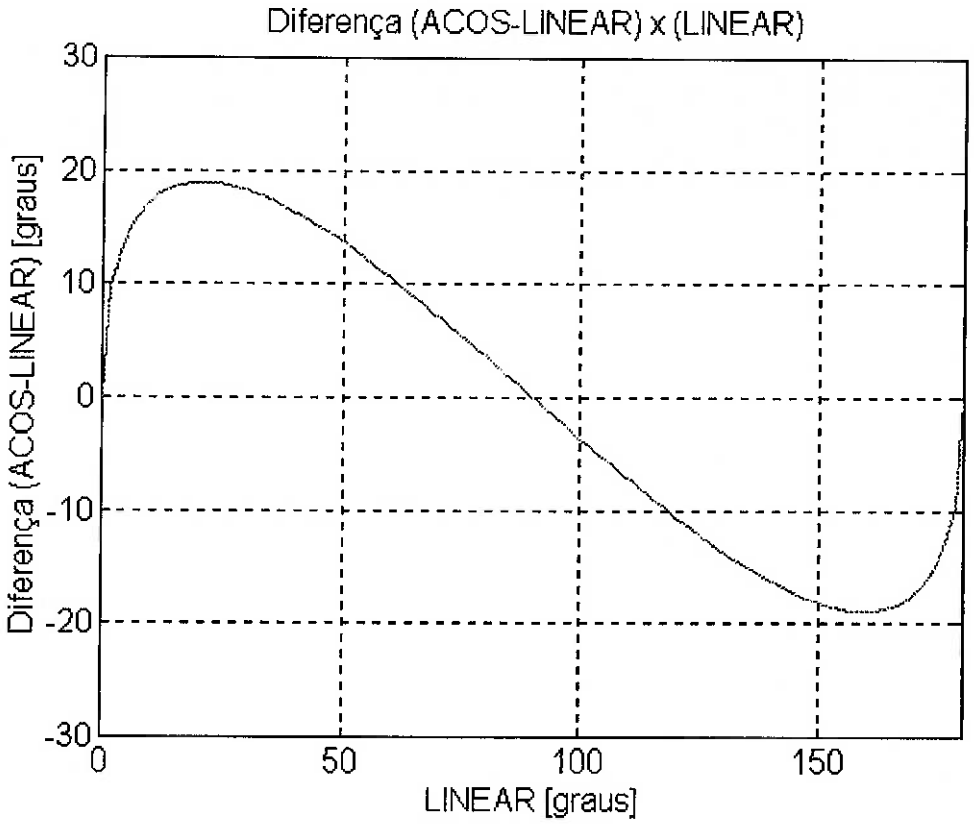



*Ilustração 9*

## Testes funcionais

Nesta secção serão mostradas algumas formas de onda geradas pelo circuito

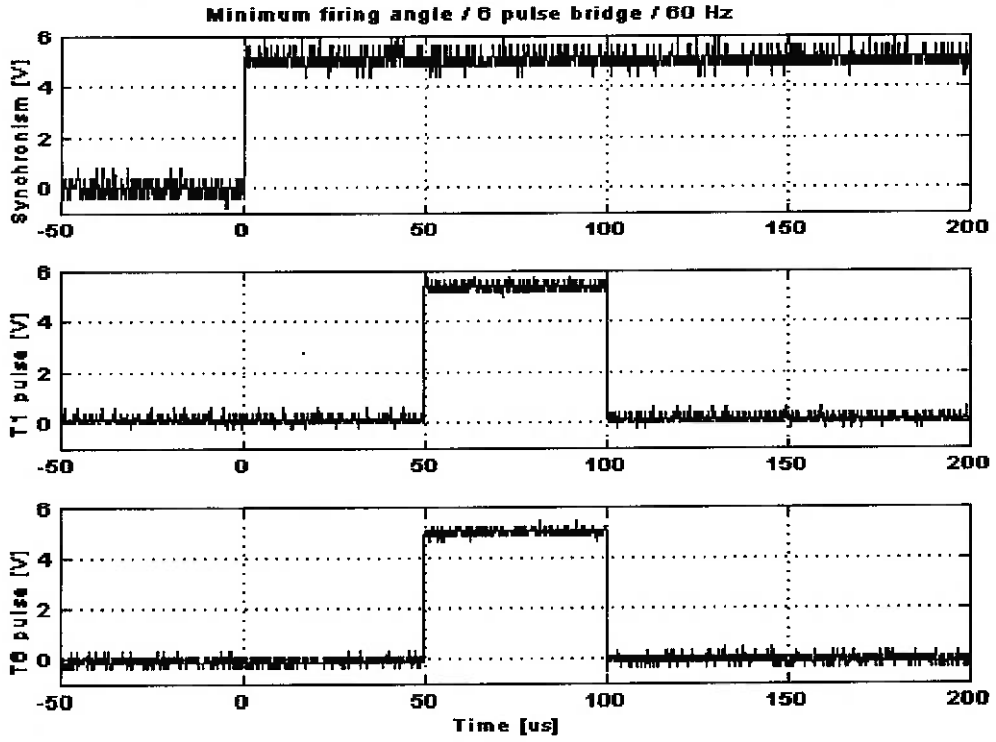



*Ilustração 10*

Ilustração 10 - Mostra as formas de onda nos pinos T1 e T6, juntamente com o sinal de sincronismo. Os pulsos de disparo têm duração de 50us. Neste caso foi utilizado o ângulo mínimo de disparo, que é 1° (aproximadamente 50us).

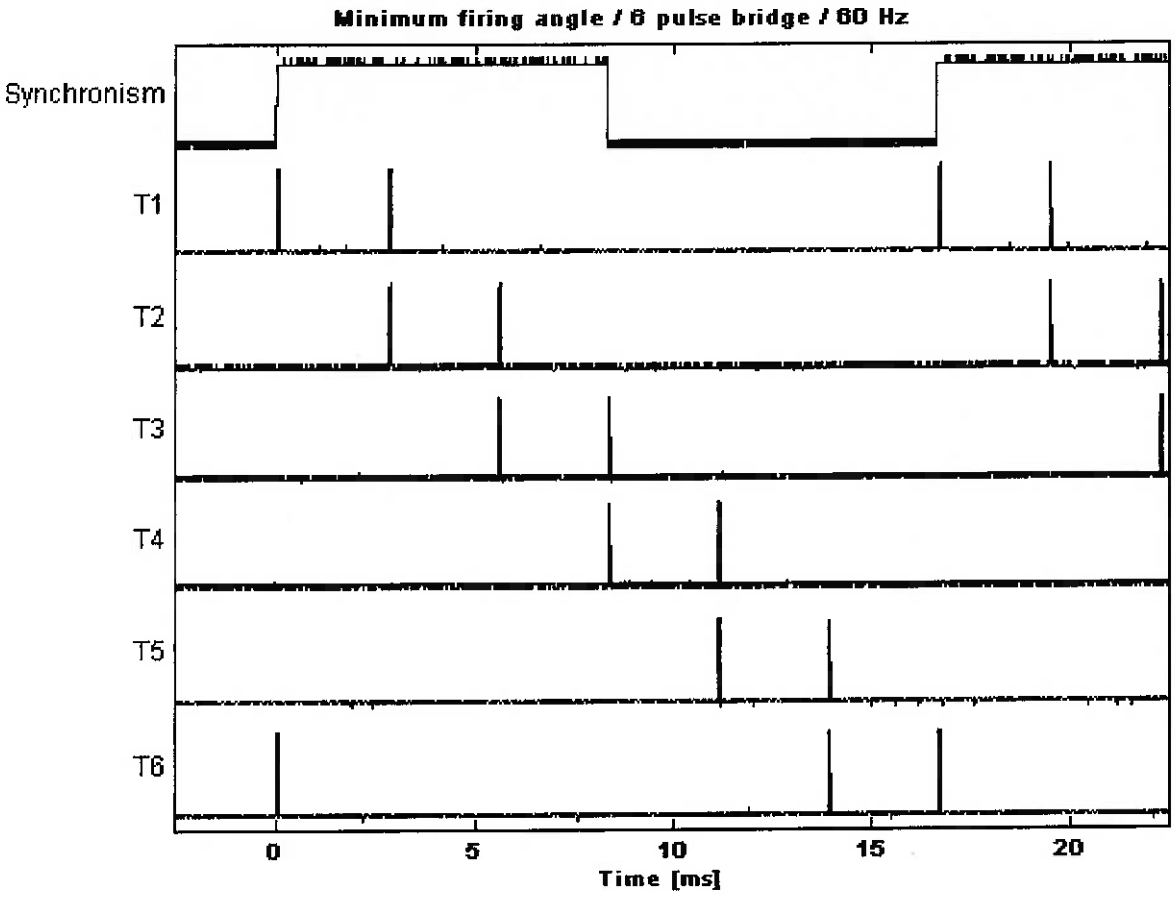



*Ilustração 11*

Ilustração 11 - Mostra os seis pinos da ponte A durante o disparo da ponte avante com o mínimo ângulo de disparo (Tensão de entrada 0V). Pode-se perceber que o tiristor anterior é sempre redisparado (double pulse), para ambientes de condução descontínua.

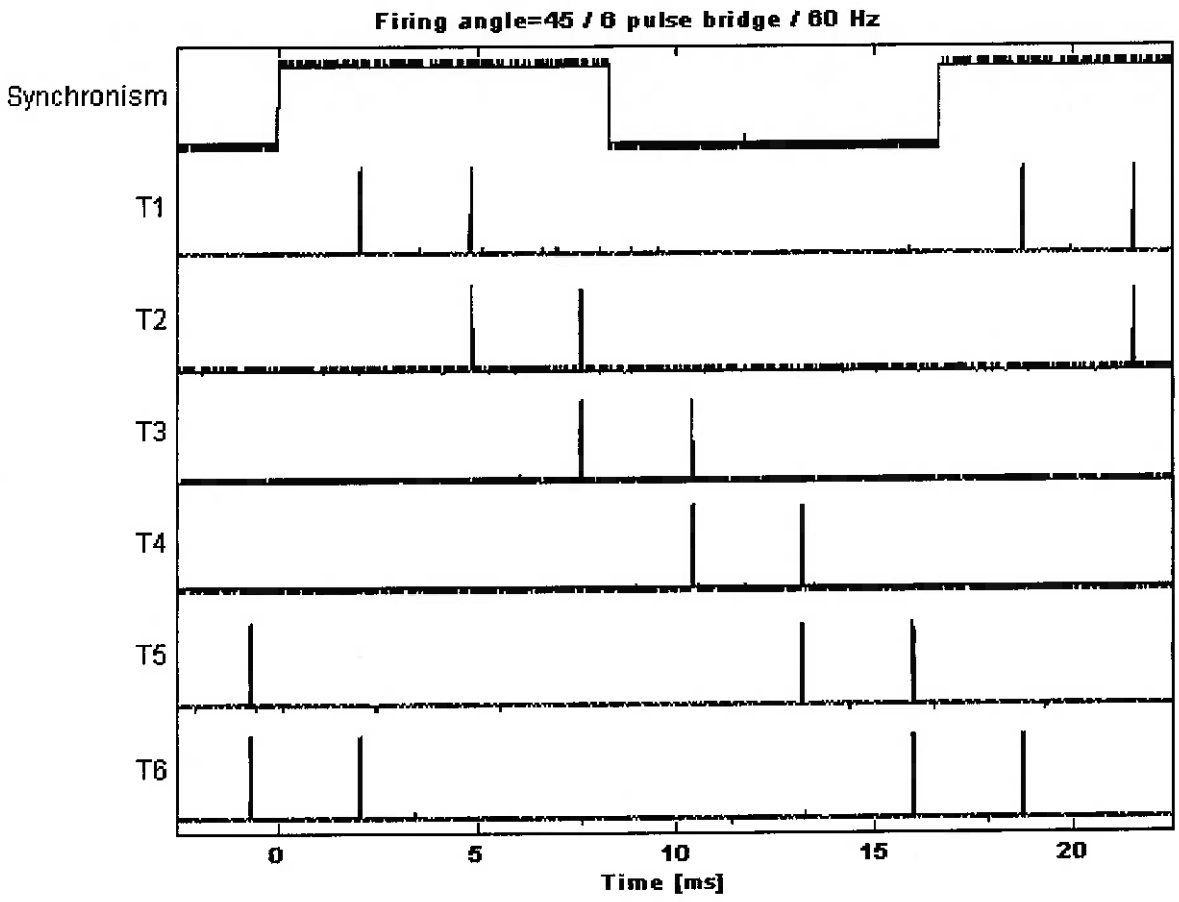



*Ilustração 12*

Ilustração 12 – Mostra o disparo da ponte A com ângulo de 45° (Tensão de entrada 1,23V).

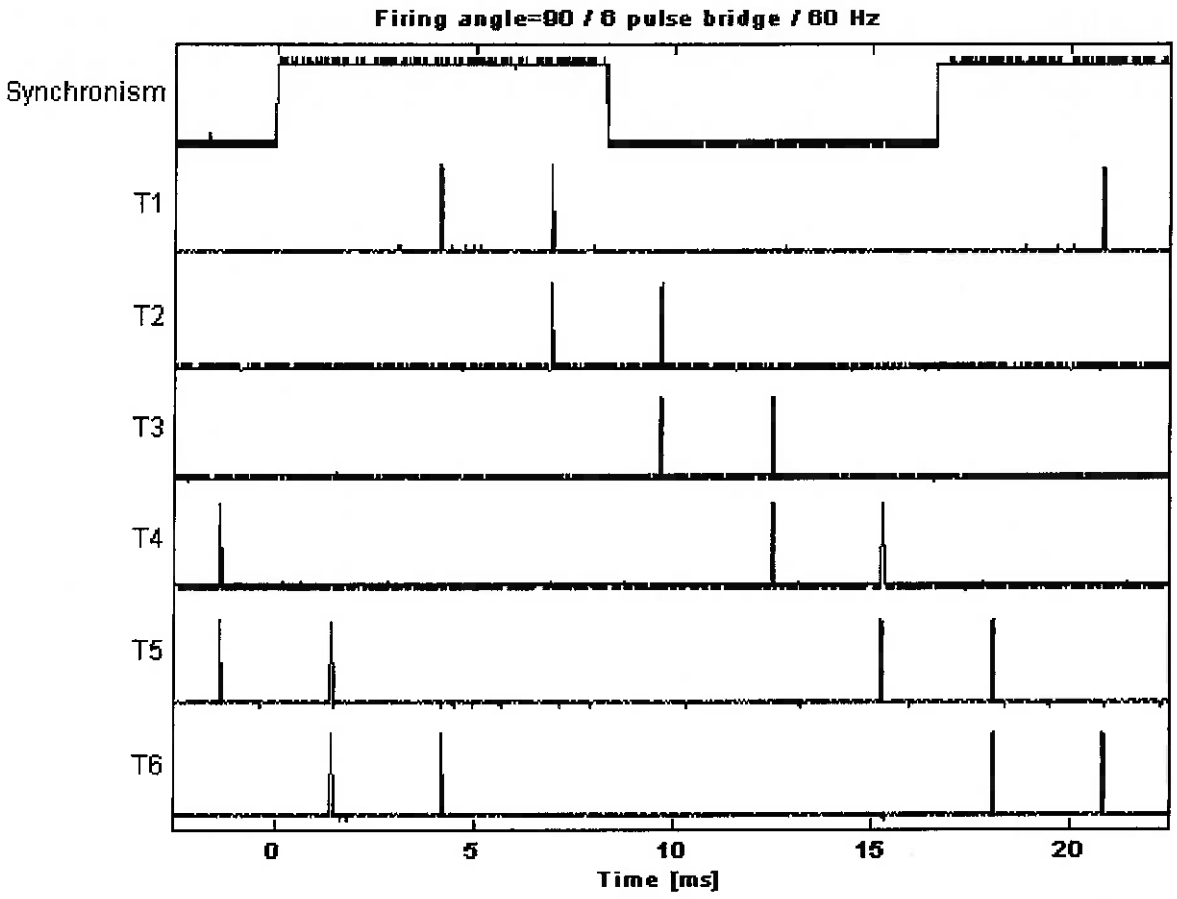



*Ilustração 13*

Ilustração 13 – Mostra o disparo da ponte A com ângulo de 90° (Tensão de entrada 2,45V).

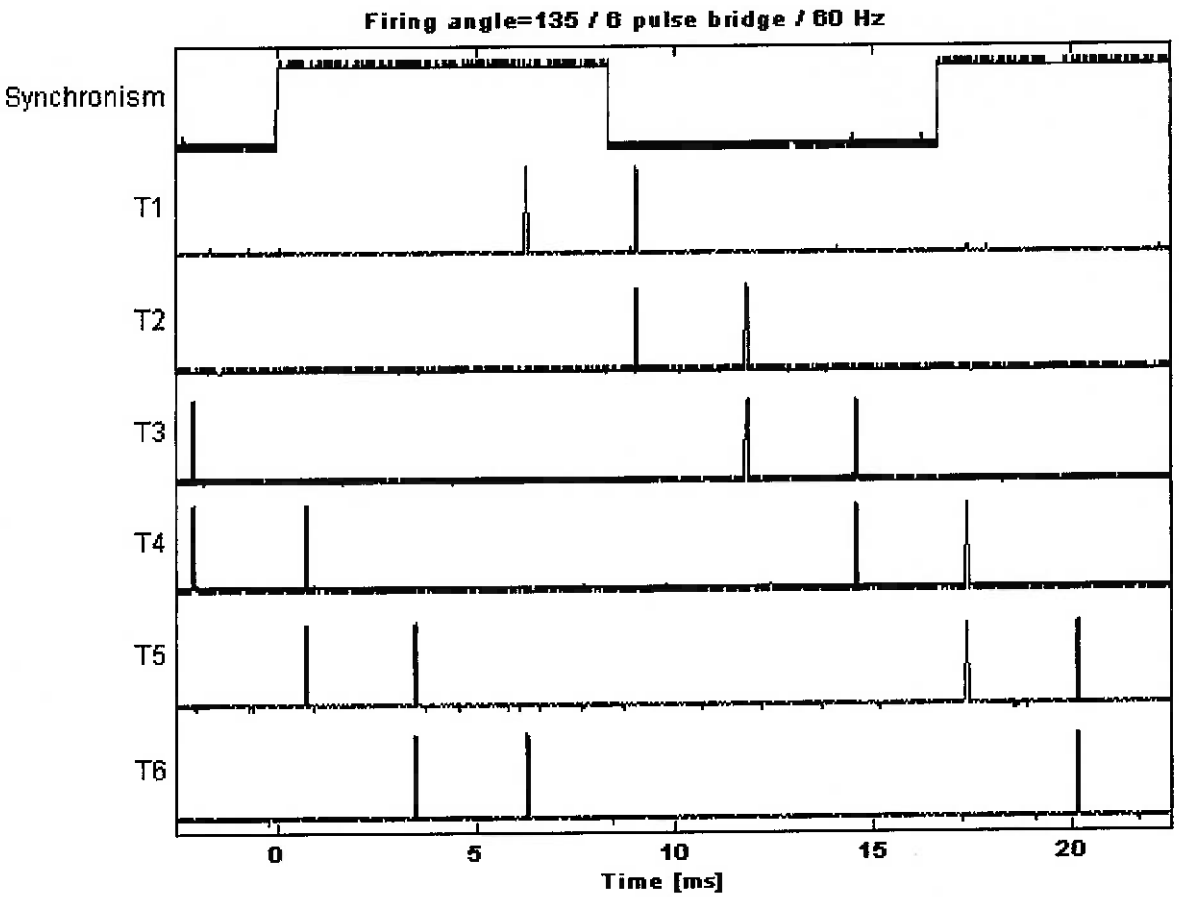



*Ilustração 14*

Ilustração 14 – Mostra o disparo da ponte A com ângulo de 135° (Tensão de entrada 3,68V).

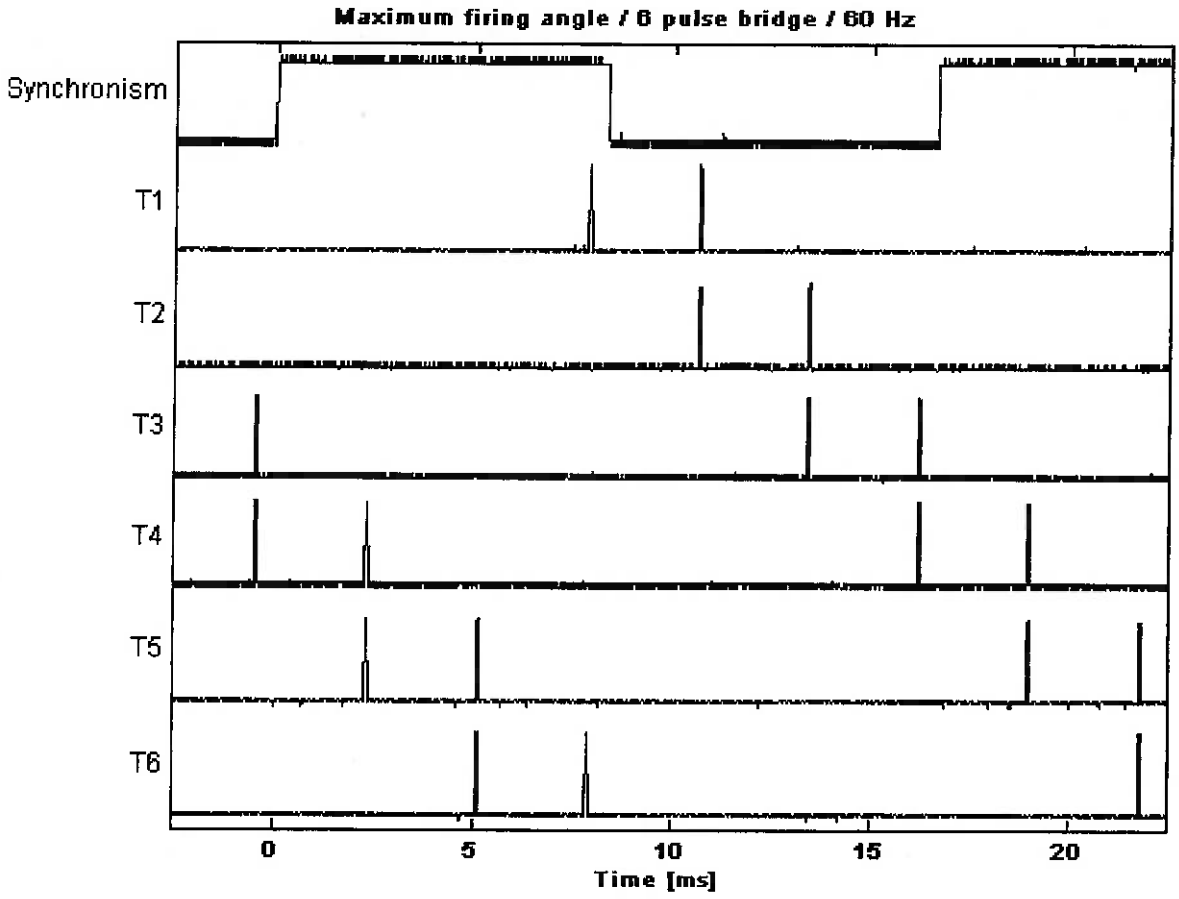



*Ilustração 15*

Ilustração 15 – Mostra o disparo da ponte A com ângulo máximo (170°) (Tensão de entrada 5V).

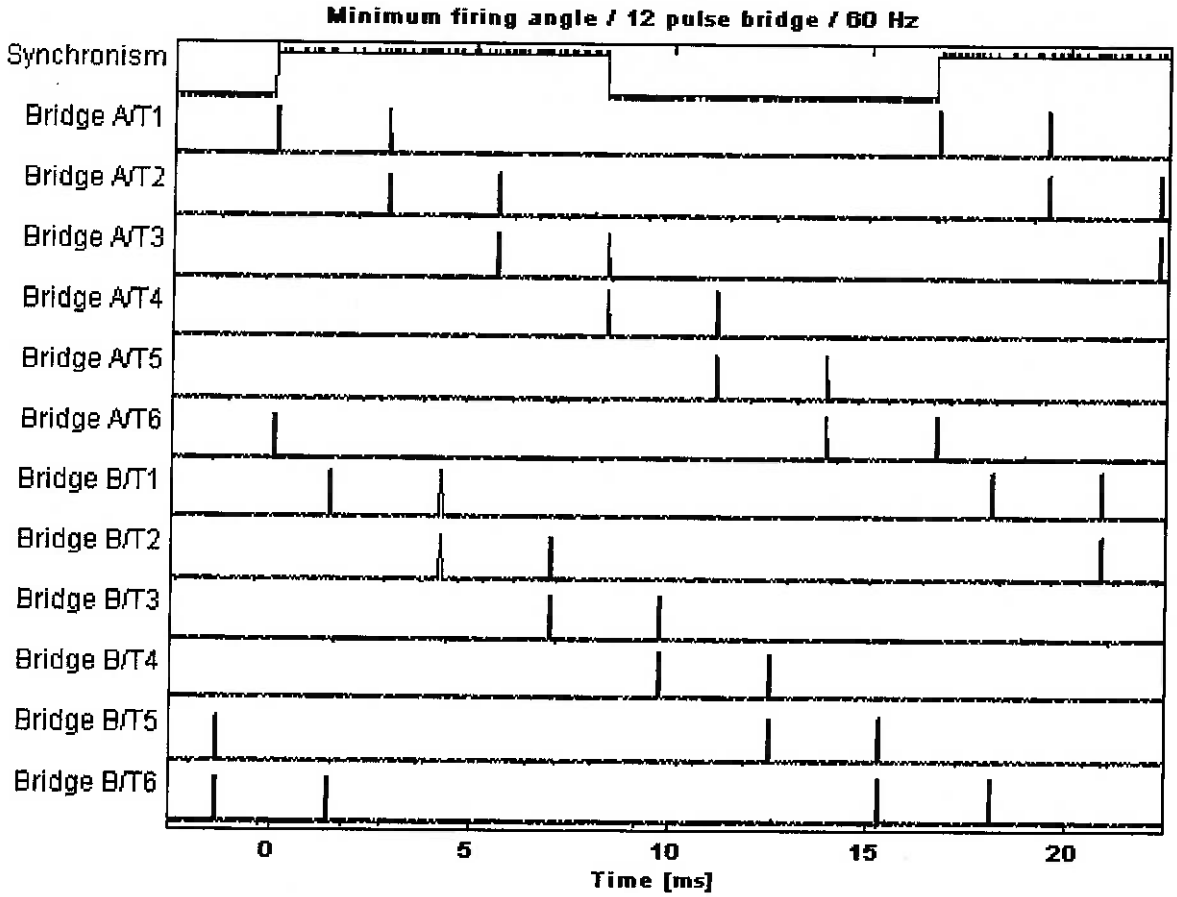



*Ilustração 16*

Ilustração 16 – Mostra o disparo das pontes A e B, com sistema configurado para 12 pulsos. Percebe-se a defasagem de 30° entre as pontes. Utilizou-se o mínimo ângulo de disparo.

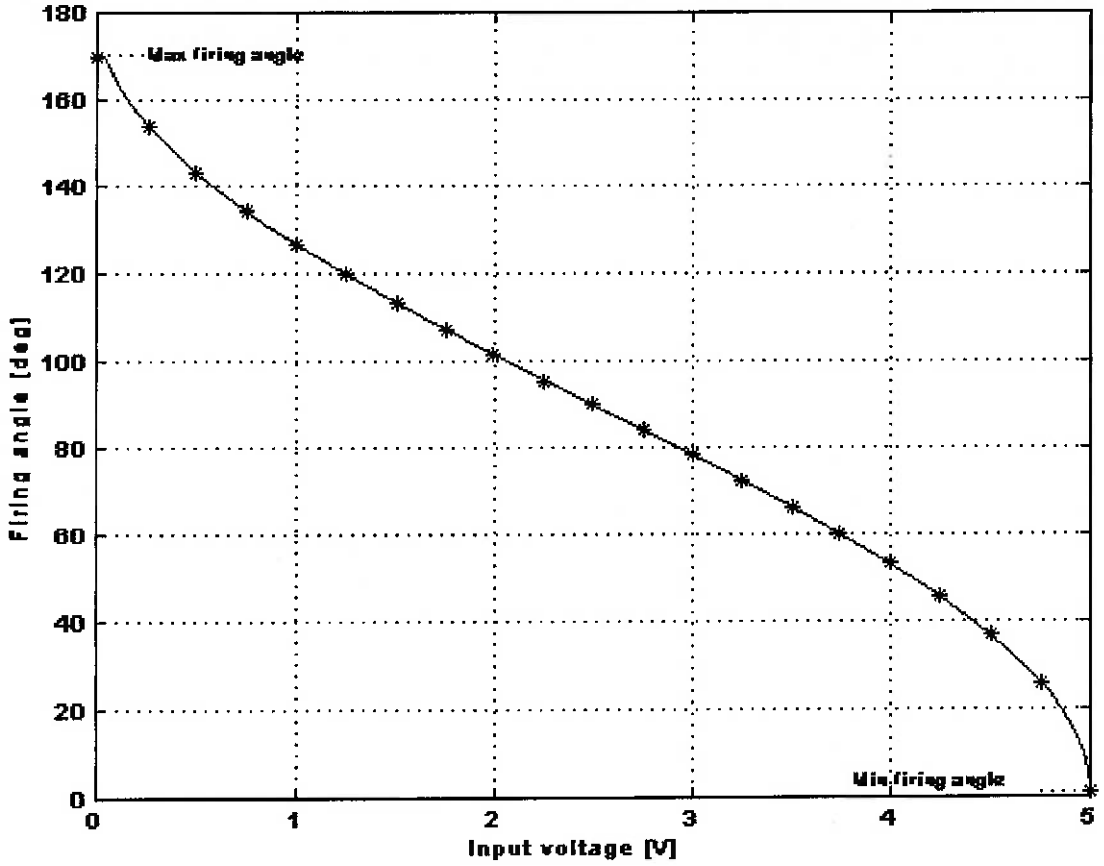



*Ilustração 17*

Ilustração 17 – Foi medido o ângulo de disparo para 21 níveis discretos de tensão de entrada, com o sistema configurado para lei de disparo arco-cosseno (Ver tabela abaixo). Estes pontos estão representados por * na ilustração acima. A curva traçada é a curva arco-cosseno teórica. Verifica-se que o sistema realiza corretamente a conversão.

| *Vin(mV)* | *T(us)* | *alpha(º)* |
|---|---|---|
| 4900 | 48 | 1,04 |
| 4660 | 1180 | 25,49 |
| 4410 | 1700 | 36,72 |
| 4164 | 2108 | 45,53 |
| 3921 | 2460 | 53,14 |
| 3672 | 2776 | 59,96 |
| 3432 | 3068 | 66,27 |
| 3187 | 3352 | 72,4 |
| 2940 | 3628 | 78,36 |

| *Vin(mV)* | *T(us)* | *alpha(º)* |
|---|---|---|
| 2697 | 3892 | 84,07 |
| 2447 | 4168 | 90,03 |
| 2207 | 4428 | 95,64 |
| 1950 | 4700 | 101,52 |
| 1714 | 4968 | 107,31 |
| 1479 | 5244 | 113,27 |
| 1218 | 5560 | 120,1 |
| 985 | 5860 | 126,58 |
| 735 | 6216 | 134,27 |
| 485 | 6632 | 143,25 |
| 248 | 7112 | 153,62 |
| 0 | 7868 | 169,95 |

Nos Anexos **III** e **V** podem ser vistos respectivamente os diagramas esquemáticos e placa de circuitos (PCB) do dispositivo utilizado para os testes acima.

## Especificações técnicas

| *Parâmetro* | *Valor* |
|---|---|
| Processador | PIC16C774 |
| Frequência de operação | 20Mhz |
| Resolução | 10Bits para 180° elétricos<br>0,178° elétricos<br>8,14us (60Hz) |
| Frequência de trabalho | 40Hz a 100Hz |
| Amostragem<br>Frequência (60Hz)<br>Período(60Hz) | 32 leituras por ciclo<br>1,9KHz<br>520us |
| Consumo (5V) | < 50mA + I/O |
| Alimentação | 4,5V < V < 5,5V |
| Precisão de disparo (digital) | 800ns |
| Watch Dog Timer timeout | 7ms < WDT < 33ms |
| Largura do pulso de disparo | 50us |
| Corrente máxima por pino I/O | 25mA ( $\Sigma$<200mA ) |
| Frequência de clock serial | 5Mhz |

## Conclusão

Utilizando um microcontrolador de baixo custo foi possível implementar um circuito disparador de ponte retificadora trifásica com recursos bastante sofisticados.

## Bibliografia


1. LEONHARD, W.; *Control of Electrical Drives*; Springer-Verlag;1985.
2. BOSE, B.K.; *Power Electronics and AC Drives*, Prentice-Hall, 1986.
3. BUXBAUM, A; SCHIERAU, K; STRAUGHEN, A; *Design of Control Systems for DC Drives*, Springer-Verlag; 1990.
4. PILLAI, S.K.; *A First Course on Electrical Drives*; Wiley Eastern Limited; 1982.

# ANEXO I

**Código Comentado**

```
;       Arquivo: DISPARO.ASM
;
; Autor:Daniel Mello Moraes Gualberto
; Out/2001
;

        #define _DISPARO_ASM_

        list    p=PIC16C774
        #include p16c774.inc
        #include disparo.inc
;*****************************************************************************
; Observações:
; - Na ISR, assumo que SEMPRE estamos no Banco0 de RAM.
;   Portanto, quando precisar sair do Banco0, desativar as interrupções.
;*****************************************************************************

;*****************************************************************************
; Variáveis para salvar contexto
;*****************************************************************************
BANCOX  UDATA
w_temp          RES     1       ; armazenamento de W
pclath_temp     RES     1       ; armazenamento de PCLATH
status_temp     RES     1       ; armazenamento de STATUS
fsr_temp        RES     1       ; armazenamento de FSR

;*****************************************************************************
; Variáveis
;*****************************************************************************
BANCO0  UDATA
contdisp        RES     1       ; Contador do tempo de disparo.
anguloh         RES     1       ; Byte mais significativo do angulo de disparo base.
angulol         RES     1       ; Byte menos significativo do angulo de disparo base.
preangh         RES     1       ; Byte mais sig. do Angulo de disparo-1 (atual).
preangl         RES     1       ; Byte menos sig. do Angulo de disparo-1 (atual).
predisph        RES     1       ; Byte mais sig. de preangh>>1.
predispl        RES     1       ; Byte menos sig. de preangl>>1.
proxconvh       RES     1       ; Byte mais sig. de proxconv.
proxconvl       RES     1       ; Byte menos sig. de proxconv.
compconvh       RES     1       ; Byte mais sig. de compconv.
compconvl       RES     1       ; Byte menos sig. de compconv.
contah          RES     1       ; Byte mais sig. do contador, apos macro montach.
contal          RES     1       ; Byte menos sig. do contador (temporario).
dpbit           RES     1       ; O bit2 armazena o decimo primeiro bit do contador.
tiristorh       RES     1       ; Seleciona os tiristores que serão disparados (9 a 12).
tiristorl       RES     1       ; Seleciona os tiristorews que serão disparados (1 a 8).
flags1          RES     1       ; flags indicadores do estado.
amaxh           RES     1       ; angulo máximo de disparo (high)
amaxl           RES     1       ; angulo máximo de disparo (low)
aminh           RES     1       ; angulo minimo de disparo (high)
aminl           RES     1       ; angulo minimo de disparo (low)
postabela       RES     1       ; posicao de procura nas tabelas.
envh            RES     1       ; Byte a ser enviado pela serial (high)
envl            RES     1       ; Byte a ser enviado pela serial (low)
rech            RES     1       ; Byte recebido pela serial (high)
recl            RES     1       ; Byte recebido pela serial (low)
cont_resp       RES     1       ; Conta o tempo de resposta na comunicação serial.

; flags1
;[BIT7][BIT6][BIT5][BIT4][BIT3][BIT2][BIT1][BIT0]
;[hser][pul2][hrev][acos][    ][    ][    ][disp]
hser    EQU     7               ; 0 - controle serial / 1 - controle analógico
pul2    EQU     6               ; 0 - ponte 12 pulsos / 1 - ponte 6 pulsos
hrev    EQU     5               ; 0 - desabilita ponte reversa / 1 - habilita ponte rever
sa
acos    EQU     4               ; 0 - disparo arcos / 1 - disparo linear
disp    EQU     0               ; Usada como resultado de verifica_disparo

;*****************************************************************************
;  Vetor de interrupção
;*****************************************************************************

STARTUP CODE
```

```
        GOTO    inicio
        nop
        nop
        nop
        BCF     TMR1H,1
PROG1   CODE

interrupcao
        BCF     PIR1,CCP1IF
        BTFSC   CCP1CON,CCP1M0
        GOTO    ccp1_bitset
ccp1_bitclear
        BSF     CCP1CON,CCP1M0
        BSF     dpbit,2
        RETFIE
ccp1_bitset
        BCF     CCP1CON,CCP1M0
        RETFIE

;*************************************************************************
; Macro: montach
; Na verdade, o contador TMR1 conta apenas de 0 a 511 (512 posições, 9bits). O
; décimo bit vem de CCP1CON. Se o bit 0 de CCP1CON estiver setado, TMR1
; está entre 0 e 511, se estiver resetado, estamos entre 512 e 1023. Portanto,
; o bit0 de CCP1CON representa o décimo bit da contagem invertido!.
; Nesta rotina, invertemos o bit0 de CCP1CON, posicionamos uma posição para a
; esquerda e somamos com TMR1H, para formar a contagem de 10bits.
; Esta macro realiza ainda outra correção para formar o byte mais significativo
; da contagem. Por exemplo, quando estou disparando a 170°, terei angulo=484,
; porem o sexto tiristor (6 pulsos) será disparado em 170°+300°=470°, ou
; 484+853=1337. Este número tem mais de 10bits, 11bits para ser exato.
; Para realizar o disparo de um tiristor do ciclo anterior, precisamos de
; mais um bit, no caso o decimo primeiro bit. Este bit está na variável dpbit,
; que tbm é somada à contagem para formar então o byte mais significativo da
; contagem (contah).
; Gasta 8 ciclos (1.6us 20Mhz).
;*************************************************************************

montach macro
        BCF     INTCON,GIE
        RLF     CCP1CON,w
        XORLW   B'00000010'
        ANDLW   B'00000010'
        ADDWF   TMR1H,w
        ADDWF   dpbit,w         ; Soma o decimo primeiro bit
        MOVWF   contah
        BSF     INTCON,GIE
    endm

;*************************************************************************
;  Macro: montapredisp
; Esta macro soma anguloh e angulol com o parâmetro inc.
; O resultado é colocado em preangh e preangl.
; Depois, elimina um bit de preang e coloca o resultado em predisp.
; Gasta 14 ciclos (2.8us 20Mhz).
;*************************************************************************

montapredisp    macro  inc
        MOVLW   HIGH(inc)
        MOVWF   preangh
        MOVLW   LOW(inc)
        ADDWF   angulol,w
        MOVWF   preangl
        MOVF    anguloh,w
        BTFSC   STATUS,C
        INCFSZ  preangh,f
        ADDWF   preangh,f
        BCF     STATUS,C
        RRF     preangh,w       ; Remove o bit menos significativo de preang
        MOVWF   predisph        ; e coloca o resultado em predisp
        RRF     preangl,w
        MOVWF   predispl
    endm
```

```
;*******************************************************************************
;  Macro: montaproxconv
; Esta macro soma proxconv com 32 (0x020), para formar o endereco da proxima
; conversão (32 = 11.25°). OBS: Como proxconv é usado para comparação com o
; timer, não utiliza 10bits para 180°, e sim 10bits para 360°.
; Gasta 6 ciclos (1.2us a 20Mhz)
;*******************************************************************************

montaproxconv   macro
        MOVLW   0x20
        ADDWF   proxconvl,f
        MOVLW   0x00
        BTFSC   STATUS,C
        MOVLW   0x01
        ADDWF   proxconvh,f
   endm

;*******************************************************************************
;  Macro: montacompconv
; Esta macro soma proxconv com 10 (0x00A), para formar a contagem limite para
; ao invés de esperarmos pela proxima conversão, esperarmos pelo proximo
; disparo. OBS: Como compconv é usado para comparação com o
; predisp, não utiliza 10bits para 180°, e sim 10bits para 360°.
; Consideramos que 190 pulsos são pelo menos 97us, ou 485 instruções a 20Mhz.
; Este tempo foi obtido considerando a frequência da rede máxima de 100Hz.
; Com rede em 60Hz, teremos 162us.
; Gasta 8 ciclos (1.6us a 20Mhz)
;*******************************************************************************

montacompconv   macro
        MOVLW   0x0A
        ADDWF   proxconvl,w
        MOVWF   compconvl
        MOVLW   0x00
        BTFSC   STATUS,C
        MOVLW   0x01
        ADDWF   proxconvh,w
        MOVWF   compconvh
   endm

;*******************************************************************************
;  Macro: limangulo
; Esta macro limita angulo entre amin e amax
; No pior caso, gastará 21 ciclos (4.2us a 20mHz)
;*******************************************************************************

limangulo macro num
        MOVF    angulol,w
        SUBWF   amaxl,w
        MOVF    anguloh,w
        BTFSS   STATUS,C
        INCFSZ  anguloh,w
        SUBWF   amaxh,w
        BTFSC   STATUS,C
        GOTO    limangulo_min#v(num)
        MOVF    amaxl,w
        MOVWF   angulol
        MOVF    amaxh,w
        MOVWF   anguloh
limangulo_min#v(num)
        MOVF    aminl,w
        SUBWF   angulol,w
        MOVF    aminh,w
        BTFSS   STATUS,C
        INCFSZ  aminh,w
        SUBWF   anguloh,w
        BTFSC   STATUS,C
        GOTO    limangulo_fim#v(num)
        MOVF    aminl,w
        MOVWF   angulol
        MOVF    aminh,w
        MOVWF   anguloh
```

```
limangulo_fim#v(num)
    endm

;*****************************************************************************
; mdisparo <num>
; Esta macro implementa disparo# e disparo_imediato#
; Estas rotinas realizam o disparo de 50us.
; Os valores que serão colocados em PORTD e PORTB, indicando os tiristores que serão
; disparados, devem estar em tiristorh e tiristorl.
; Só retorna após o pulso estar completo.
; Uso macro para poupar tempo de execução com chamada retorno de subrotina.
;*****************************************************************************

mdisparo macro num

disparo#v(num)
        BTFSC   PORTC,0
        GOTO    disparo#v(num)
        BTFSC   PORTA,1
        GOTO    disparo_fim#v(num) ; Verifica se os disparos estão inibidos
        BCF     INTCON,GIE         ; Desabilita as interrupções
        BTFSC   preangl,0
        GOTO    disparo_loop_1impar#v(num)
disparo_loop#v(num)
        BTFSS   PORTC,0
        GOTO    disparo_loop#v(num)
disparo_imediato#v(num)
        MOVF    tiristorl,w
        MOVWF   PORTD
        MOVF    tiristorh,w
        MOVWF   PORTB
        MOVLW   D'82'
        MOVWF   contdisp
disparo_loop_pul#v(num)
        DECFSZ  contdisp,f
        GOTO    disparo_loop_pul#v(num)
        GOTO    disparo_fim#v(num)
disparo_loop_1impar#v(num)
        BTFSS   PORTC,0
        GOTO    disparo_loop_1impar#v(num)
disparo_loop_2impar#v(num)
        BTFSC   PORTC,0
        GOTO    disparo_loop_2impar#v(num)
disparo_imediato_impar#v(num)
        MOVF    tiristorl,w
        MOVWF   PORTD
        MOVF    tiristorh,w
        MOVWF   PORTB
        MOVLW   D'82'
        MOVWF   contdisp
disparo_loop_pul_impar#v(num)
        DECFSZ  contdisp,f
        GOTO    disparo_loop_pul_impar#v(num)
disparo_fim#v(num)
        CLRF    PORTD
        CLRF    PORTB
        BSF     INTCON,GIE      ; Habilita as interrupções
   endm

;*****************************************************************************
; Rotina principal
;*****************************************************************************

inicio
        CLRF    STATUS
        ; ** Inicializa as Portas **
        CLRF    PORTA
        CLRF    PORTB
        CLRF    PORTC
        CLRF    PORTD
        CLRF    PORTE
        BSF     STATUS,RP0      ; B1
        MOVLW   B'00011111'     ; B1
```

```
        MOVWF   TRISA           ; B1
        MOVLW   B'11110000'     ; B1
        MOVWF   TRISB           ; B1
        MOVLW   B'10010001'     ; B1
        MOVWF   TRISC           ; B1
        CLRF    TRISD           ; B1
        CLRF    TRISE           ; B1
        ; ** Verifica se tivemos WDT reset **
        BTFSC STATUS,NOT_TO     ; B1
        GOTO    cont_config     ; B1
        BCF     STATUS,RP0      ; B1 Volta ao banco 0
        BSF     PORTE,0         ; Acende o LED de WDT
wdt_loop
        GOTO    wdt_loop
cont_config
        ; ** Configura PCON **
        BSF     PCON,NOT_POR    ; B1
        BSF     PCON,NOT_BOR    ; B1
        ; ** Configura SSPSTAT **
        MOVLW   B'01000000'     ; B1
        MOVWF   SSPSTAT         ; B1
        ; ** Configura OPTION_REG **
        CLRWDT                  ; B1
        MOVLW   B'01011000'     ; B1
        MOVWF   OPTION_REG      ; B1
        BCF     STATUS,RP0      ; B1
        ; ** INTCON **
        MOVLW   B'01000000'
        MOVWF   INTCON
        ; ** PIE1 **
        BSF     STATUS,RP0      ; B1
        MOVLW   B'00000100'     ; B1
        MOVWF   PIE1            ; B1
        ; ** PIE2 **
        CLRF    PIE2            ; B1
        ; ** LVDCON **
        MOVLW   B'00000100'
        MOVWF   LVDCON          ; B1
        BCF     STATUS,RP0      ; B1
        ; ** ADCON0 **
        MOVLW   B'10000001'
        MOVWF   ADCON0
        ; ** ADCON1 **
        BSF     STATUS,RP0      ; B1
        MOVLW   B'10111110'     ; B1
        MOVWF   ADCON1          ; B1
        BCF     STATUS,RP0      ; B1
        ; ** CCP1 **
  if (!debug)
        MOVLW   B'00001000'     ; CCP1 compare mode
        MOVWF   CCP1CON
  else
        MOVLW   B'00001001'     ; CCP1 compare mode
        MOVWF   CCP1CON
  endif
        MOVLW   0x02
        MOVWF   CCPR1H          ; Período de 512.
        CLRF    CCPR1L
        ; ** SSPCON **
        MOVLW   B'00100000'
        MOVWF   SSPCON
        ;***************************
        ; ** INICIALIZA VARIAVEIS **
        ;***************************
        MOVLW   0x00            ; amin (1°)
        MOVWF   aminh
        MOVLW   0x06
        MOVWF   aminl
        ;===========================
        MOVLW   0x03            ; amax (170°)
        MOVWF   amaxh
        MOVLW   0xC7
        MOVWF   amaxl
```

```
        ;============================
        BSF     PORTA,5         ; Acende o LED que indica placa funcionando.
        MOVF    PORTB,w         ; Copia o conteúdo dos jumpers para a variável
        ANDLW   0xF0            ; flags1
        MOVWF   flags1
        ; ** TIMER1 **
        CLRF    TMR1L
        CLRF    TMR1H
        MOVLW   B'00000011'
        MOVWF   T1CON
        BSF     INTCON,GIE
        ;============================
aguarda_habilitar
        CLRWDT
        BTFSC   PORTC,7         ; Verifica se o disp. de controle deseja enviar comando.

        CALL    le_serial_desabilitado
        BTFSC   PORTA,1
        GOTO    aguarda_habilitar
        ;============================
        MOVF    amaxh,w         ; Inicializa o ângulo de disparo com amax
        MOVWF   anguloh         ; Coloca amax tbm no buffer de transmissão.
        MOVWF   envh
        MOVF    amaxl,w
        MOVWF   angulol
        MOVWF   envl
        ;============================
        BTFSS   flags1,pu12
        GOTO    init_p12
        BTFSC   PORTA,2
        GOTO    init_av_6
        BTFSC   flags1,hrev
        GOTO    init_rev_6
        GOTO    aguarda_habilitar

;*****************************************************************************
; Ponte avante
;*****************************************************************************

verifica_av_6   macro
        CLRWDT
        BTFSC   PORTA,1
        GOTO    termina_av_6
   endm

init_av_6
        CLRF    tiristorl       ; tiristor.
        CLRF    tiristorh
        ;============================
        CLRF    proxconvl       ; proxconv, de início com 1024!
        MOVLW   0x04
        MOVWF   proxconvh
        ;============================
        CLRF    dpbit
aguarda_inicio_av_6
        CLRWDT
        BTFSS   dpbit,2
        GOTO    aguarda_inicio_av_6
        ;============================
        BSF     PORTE,1         ; Acende o LED P.Avante
loop_princ_av_6
disp1_av_6
        MOVLW   B'00100001'     ; Prepara tiristores para o primeiro disparo
        MOVWF   tiristorl
        CLRF    tiristorh
disp1_av_6_loop
        CALL    aguarda_conversao
        montapredisp 0x07FE     ; É como se estivessemos 360° à frente!
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp1_av_6_loop
        CALL    aguarda_disparo
        verifica_av_6
```

```
        ;=== Neste ponto, com certeza dpbit estará setado! Devemos reseta-lo
        BCF     dpbit,2
        BCF     proxconvh,2
disp2_av_6
        MOVLW   B'00000011'     ; Prepara tiristores para o segundo disparo
        MOVWF   tiristorl
        CLRF    tiristorh
disp2_av_6_loop
        CALL    aguarda_conversao
        montapredisp 0x0153
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp2_av_6_loop
        CALL    aguarda_disparo
        verifica_av_6
disp3_av_6
        MOVLW   B'00000110'     ; Prepara tiristores para o terceiro disparo
        MOVWF   tiristorl
        CLRF    tiristorh
disp3_av_6_loop
        CALL    aguarda_conversao
        montapredisp 0x02A9
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp3_av_6_loop
        CALL    aguarda_disparo
        verifica_av_6
disp4_av_6
        MOVLW   B'00001100'     ; Prepara tiristores para o quarto disparo
        MOVWF   tiristorl
        CLRF    tiristorh
disp4_av_6_loop
        CALL    aguarda_conversao
        montapredisp 0x03FE
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp4_av_6_loop
        CALL    aguarda_disparo
        verifica_av_6
disp5_av_6
        MOVLW   B'00011000'     ; Prepara tiristores para o quinto disparo
        MOVWF   tiristorl
        CLRF    tiristorh
disp5_av_6_loop
        CALL    aguarda_conversao
        montapredisp 0x0553
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp5_av_6_loop
        CALL    aguarda_disparo
        verifica_av_6
disp6_av_6
        MOVLW   B'00110000'     ; Prepara tiristores para o sexto disparo
        MOVWF   tiristorl
        CLRF    tiristorh
disp6_av_6_loop
        CALL    aguarda_conversao
        montapredisp 0x06A9
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp6_av_6_loop
        CALL    aguarda_disparo
        verifica_av_6
fim_loop_av_6
        GOTO    loop_princ_av_6
termina_av_6
        BCF     PORTE,1         ; apaga o LED P.Avante
        GOTO    aguarda_habilitar

;*****************************************************************************
; Ponte reversa
;*****************************************************************************
```

```
verifica_rev_6  macro
        CLRWDT
        BTFSC   PORTA,1
        GOTO    termina_rev_6
    endm

init_rev_6
        CLRF    tiristorl       ; tiristor.
        CLRF    tiristorh
        ;============================
        CLRF    proxconvl       ; proxconv, de início com 1024!
        MOVLW   0x04
        MOVWF   proxconvh
        ;============================
        CLRF    dpbit
aguarda_inicio_rev_6
        CLRWDT
        BTFSS   dpbit,2
        GOTO    aguarda_inicio_rev_6
        ;============================
        BSF     PORTE,2         ; Acende o LED P.Reversa
loop_princ_rev_6
disp1_rev_6
        MOVLW   B'01000000'     ; Prepara tiristores para o primeiro disparo
        MOVWF   tiristorl
        MOVLW   B'00001000'
        MOVWF   tiristorh
disp1_rev_6_loop
        CALL    aguarda_conversao
        montapredisp 0x07FE     ; É como se estivessemos 360° à frente!
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp1_rev_6_loop
        CALL    aguarda_disparo
        verifica_rev_6
        ;=== Neste ponto, com certeza dpbit estará setado! Devemos reseta-lo
        BCF     dpbit,2
        BCF     proxconvh,2
disp2_rev_6
        MOVLW   B'11000000'     ; Prepara tiristores para o segundo disparo
        MOVWF   tiristorl
        MOVLW   B'00000000'
        MOVWF   tiristorh
disp2_rev_6_loop
        CALL    aguarda_conversao
        montapredisp 0x0153
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp2_rev_6_loop
        CALL    aguarda_disparo
        verifica_rev_6
disp3_rev_6
        MOVLW   B'10000000'     ; Prepara tiristores para o terceiro disparo
        MOVWF   tiristorl
        MOVLW   B'00000001'
        MOVWF   tiristorh
disp3_rev_6_loop
        CALL    aguarda_conversao
        montapredisp 0x02A9
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp3_rev_6_loop
        CALL    aguarda_disparo
        verifica_rev_6
disp4_rev_6
        MOVLW   B'00000000'     ; Prepara tiristores para o quarto disparo
        MOVWF   tiristorl
        MOVLW   B'00000011'
        MOVWF   tiristorh
disp4_rev_6_loop
        CALL    aguarda_conversao
        montapredisp 0x03FE
        CALL    verifica_disparo
```

```
        BTFSS   flags1,disp
        GOTO    disp4_rev_6_loop
        CALL    aguarda_disparo
        verifica_rev_6
disp5_rev_6
        MOVLW   B'00000000'     ; Prepara tiristores para o quinto disparo
        MOVWF   tiristorl
        MOVLW   B'00000110'
        MOVWF   tiristorh
disp5_rev_6_loop
        CALL    aguarda_conversao
        montapredisp 0x0553
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp5_rev_6_loop
        CALL    aguarda_disparo
        verifica_rev_6
disp6_rev_6
        MOVLW   B'00000000'     ; Prepara tiristores para o sexto disparo
        MOVWF   tiristorl
        MOVLW   B'00001100'
        MOVWF   tiristorh
disp6_rev_6_loop
        CALL    aguarda_conversao
        montapredisp 0x06A9
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp6_rev_6_loop
        CALL    aguarda_disparo
        verifica_rev_6
fim_rev_6
        GOTO    loop_princ_rev_6
termina_rev_6
        BCF     PORTE,2         ; Apaga o LED P.Reversa
        GOTO    aguarda_habilitar

;*****************************************************************************
; Ponte 12 pulsos
;*****************************************************************************

verifica_p12    macro
        CLRWDT
        BTFSC   PORTA,1
        GOTO    termina_p12
   endm

init_p12
        CLRF    tiristorl       ; tiristor.
        CLRF    tiristorh
        ;=========================
        CLRF    proxconvl       ; proxconv, de início com 1024!
        MOVLW   0x04
        MOVWF   proxconvh
        ;=========================
        CLRF    dpbit
aguarda_inicio_p12
        CLRWDT
        BTFSS   dpbit,2
        GOTO    aguarda_inicio_p12
        ;=========================
        BSF     PORTE,1         ; Acende o LED P.Avante
        BSF     PORTE,2         ; Acende o LED P.Reversa
loop_princ_p12
disp1_p12
        MOVLW   B'00100001'     ; Prepara tiristores para o primeiro disparo
        MOVWF   tiristorl
        MOVLW   B'00000000'
        MOVWF   tiristorh
disp1_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x07FE     ; É como se estivessemos 360° à frente!
        CALL    verifica_disparo
        BTFSS   flags1,disp
```

```
        GOTO    disp1_p12_loop
        CALL    aguarda_disparo
        verifica_p12
        ;=== Neste ponto, com certeza dpbit estará setado! Devemos reseta-lo
        BCF     dpbit,2
        BCF     proxconvh,2
disp2_p12
        MOVLW   B'01000000'     ; Prepara tiristores para o segundo disparo
        MOVWF   tiristorl
        MOVLW   B'00001000'
        MOVWF   tiristorh
disp2_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x00A9
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp2_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp3_P12
        MOVLW   B'00000011'     ; Prepara tiristores para o terceiro disparo
        MOVWF   tiristorl
        MOVLW   B'00000000'
        MOVWF   tiristorh
disp3_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x0153
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp3_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp4_p12
        MOVLW   B'11000000'     ; Prepara tiristores para o quarto disparo
        MOVWF   tiristorl
        MOVLW   B'00000000'
        MOVWF   tiristorh
disp4_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x01FE
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp4_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp5_p12
        MOVLW   B'00000110'     ; Prepara tiristores para o quinto disparo
        MOVWF   tiristorl
        MOVLW   B'00000000'
        MOVWF   tiristorh
disp5_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x02A9
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp5_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp6_p12
        MOVLW   B'10000000'     ; Prepara tiristores para o sexto disparo
        MOVWF   tiristorl
        MOVLW   B'00000001'
        MOVWF   tiristorh
disp6_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x0353
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp6_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp7_p12
        MOVLW   B'00001100'     ; Prepara tiristores para o setimo disparo
```

```
        MOVWF   tiristorl
        MOVLW   B'00000000'
        MOVWF   tiristorh
disp7_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x03FE
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp7_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp8_p12
        MOVLW   B'00000000'     ; Prepara tiristores para o oitavo disparo
        MOVWF   tiristorl
        MOVLW   B'00000011'
        MOVWF   tiristorh
disp8_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x04A9
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp8_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp9_P12
        MOVLW   B'00011000'     ; Prepara tiristores para o nono disparo
        MOVWF   tiristorl
        MOVLW   B'00000000'
        MOVWF   tiristorh
disp9_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x0553
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp9_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp10_p12
        MOVLW   B'00000000'     ; Prepara tiristores para o decimo disparo
        MOVWF   tiristorl
        MOVLW   B'00000110'
        MOVWF   tiristorh
disp10_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x05FE
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp10_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp11_p12
        MOVLW   B'00110000'     ; Prepara tiristores para o decimo primeiro disparo
        MOVWF   tiristorl
        MOVLW   B'00000000'
        MOVWF   tiristorh
disp11_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x06A9
        CALL    verifica_disparo
        BTFSS   flags1,disp
        GOTO    disp11_p12_loop
        CALL    aguarda_disparo
        verifica_p12
disp12_p12
        MOVLW   B'00000000'     ; Prepara tiristores para o decimo segundo disparo
        MOVWF   tiristorl
        MOVLW   B'00001100'
        MOVWF   tiristorh
disp12_p12_loop
        CALL    aguarda_conversao
        montapredisp 0x0753
        CALL    verifica_disparo
        BTFSS   flags1,disp
```

```
        GOTO    disp12_p12_loop
        CALL    aguarda_disparo
        verifica_p12
fim_p12
        GOTO    loop_princ_p12
termina_p12
        BCF     PORTE,1         ; Apaga o LED P.Avante
        BCF     PORTE,2         ; Apaga o LED P.Reversa
        GOTO    aguarda_habilitar

;*****************************************************************************
;  Esta rotina verifica se devemos disparar ou aguardar a proxima conversão
;*****************************************************************************

verifica_disparo
        montaproxconv
        montacompconv
        BCF     flags1,disp
        MOVF    predispl,w
        SUBWF   compconvl,w
        MOVF    predisph,w
        BTFSS   STATUS,C
        INCFSZ  predisph,w
        SUBWF   compconvh,w
        BTFSC   STATUS,C
        BSF     flags1,disp
        RETURN

;*****************************************************************************
;  Esta rotina aguarda até o momento do disparo e dispara
;*****************************************************************************

aguarda_disparo
        ;=============================================================================
        ; Se predispl for zero, só precisamos comparar predisph e contah
        ; verifica_contah é uma rotina dedicada a isso.
        ;=============================================================================
        MOVF    predispl,f
        BTFSC   STATUS,Z
        GOTO    verifica_contah
        ;=============================================================================
        ; Em poucos casos a rotina acima será acionada, na maioria das vezes
        ; passaremos por aqui.
        ; Primeiro, armazenamos uma copia válida de TMR1H(criado por montach)
        ; e TMR1L em contah e contal. Dizemos que deve ser uma cópia válida pois
        ; verificamos se não houve alteração de TMR1L após armazenamento.
        ;=============================================================================
refaz_conta
        MOVF    TMR1L,w
        MOVWF   contal
        montach
        MOVF    contal,w
        SUBWF   TMR1L,w
        BTFSS   STATUS,Z
        GOTO    refaz_conta
        ;=============================================================================
        ; Agora comparo contah com predisph.
        ; Caso seja maior, disparo imediatamente. Caso seja igual, coloco
        ; o valor de contal em w e verifico se este é o momento de disparar,
        ; comparando contal com predispl.
        ;=============================================================================
        MOVF    contah,w
        SUBWF   predisph,w
        BTFSS   STATUS,C
        GOTO    disparo_imediato1
        BTFSS   STATUS,Z
        GOTO    loop_comp_h
        MOVF    contal,w
        SUBWF   predispl,w
        BTFSS   STATUS,C
        GOTO    disparo_imediato1
        BTFSS   STATUS,Z
        GOTO    loop_comp_l
```

```
        GOTO    disparo1
        ;=======================================================================
        ; Finalmente, caso não tenhamos disparado nas rotinas acima, que
        ; representam casos especiais em que o angulo de disparo era inferior
        ; ou igual à contagem atual, entro nas rotinas que realizam a operação
        ; normal, que é um loop para comparar apenas o byte mais significativo,
        ; e outro para comparar apenas o menos significativo.
        ;=======================================================================
loop_comp_h
        montach
        MOVF    predisph,w
        SUBWF   contah,w
        BTFSS   STATUS,C
        GOTO    loop_comp_h
loop_comp_l
        MOVF    predispl,w
        SUBWF   TMR1L,w
        BTFSS   STATUS,C
        GOTO    loop_comp_l
        mdisparo 1
        RETURN

; ============================================================
; = Neste caso, preangl é zero, só precisamos comparar preangh =
; ============================================================
; Chegando aqui, na primeira comparação, verificamos se
; contah é maior ou igual a predisph. Caso afirmativo, dispara
; imediatamente.
; Senão, entramos em um loop que comparará continuamente se
; contah é maior ou igual a predisph e, quando for, vai para
; a rotina de disparo.
; ============================================================

verifica_contah
        montach
        MOVF    predisph,w
        SUBWF   contah,w
        BTFSC   STATUS,C
        GOTO    disparo_imediato1
verifica_contah_loop                    ; Este loop gasta 13 ciclos.
        montach                         ;8
        MOVF    predisph,w              ;9
        SUBWF   contah,w                ;10
        BTFSS   STATUS,C                ;11
        GOTO    verifica_contah_loop    ;13
        GOTO    disparo1

;*************************************************************************
;  Conversão A/D - 26us
;  p/ limitar o ângulo de disparo gastamos mais 4.2us
; Esta rotina realiza uma conversão A/D (de 12 bits), elimina 2 bits, deixando
; 10 bits no total, e limita esse valor entre amin e amax.
; Para inverter o valor, são gastos mais 0.6us
; No pior caso, são gastos 30.8us
;*************************************************************************

conversao       macro
   if (!debug)
        BSF     PORTC,1         ; **DEBUG** Pulsa portc1
        BCF     PORTC,1
        BTFSS   flags1,hser     ; verifica se teremos entrada serial ou analógica
        GOTO    le_serial_disparo
        BSF     ADCON0,GO       ; Inicia a conversão
conv_loop
        BTFSC   ADCON0,NOT_DONE ; Aguarda a conversão terminar
        GOTO    conv_loop
        BCF     STATUS,C
        RRF     ADRESH,w
        MOVWF   anguloh
        BCF     INTCON,GIE      ; Desabilita interrupções, para irmos ao banco1
        BSF     STATUS,RP0
        RRF     ADRESL,w
        BCF     STATUS,RP0
```

```
        BSF     INTCON,GIE      ; Habilita interrupções.
        MOVWF   angulol
        BCF     STATUS,C
        RRF     anguloh,f       ; Elimina mais um bit menos significativo.
        RRF     angulol,f       ; deixando no total 10 bits.
conv_proc_final
        ;=====================
        COMF    angulol,f       ; Inverte o valor de 10 bits!
        MOVLW   b'00000011'
        XORWF   anguloh,f
        ;=====================
        BTFSS   PORTA,4         ; Phase-back
        BSF     anguloh,7       ; Setando este bit, forçamos amax
        BTFSS   flags1,acos
        CALL    transforma_acos ; Caso disparo
        limangulo 1             ; Limita o ângulo de disparo (max 21 ciclos).
        BTFSS   flags1,hser     ; Se temos entrada serial, devo colocar o ângulo calculad
o
        GOTO    copia_angulo    ; no buffer de saída
conv_final
   else
        MOVLW   D'06'
        MOVWF   angulol
        CLRF    anguloh
   endif
   endm

;=======================================================
; Copia o ângulo calculado para o buffer de saída
; 8 ciclos, já com a chamada (1,6us)

copia_angulo
        MOVF    anguloh,w
        MOVWF   envh
        MOVF    angulol,w
        MOVWF   envl
        GOTO    conv_final

;*************************************************************************
;  le_serial_disparo
; Lê informação da porta serial durante funcionamento normal (disparo).
; Caso a informação for amax ou amin, coloca os valores recebidos no buffer para
; envio na proxima transmissão.
;*************************************************************************

le_serial_disparo
        MOVLW   D'5'
        MOVWF   cont_resp
        BSF     PORTC,6         ; Ativa RDT -> Request data
loop_le_ser
        BTFSC   PORTC,7
        GOTO    le_bytes
        DECFSZ  cont_resp,f
        GOTO    loop_le_ser
        BCF     PORTC,6         ; Não houve resposta do dispositivo serial
        GOTO    conv_final
le_bytes
        BCF     PIR1,SSPIF      ; Limpa a flag SSPIF
        MOVF    envh,w
        MOVWF   SSPBUF          ; Coloca a parte mais sig. do byte a ser enviado em SSPBU
F
loop_le_byteh
        BTFSS   PIR1,SSPIF
        GOTO    loop_le_byteh
        MOVF    SSPBUF,w        ; Armazena o byte recebido (high)
        MOVWF   rech
        BCF     PIR1,SSPIF      ; Limpa a flag SSPIF
        MOVF    envl,w
        MOVWF   SSPBUF          ; Coloca a parte mais sig. do byte a ser enviado em SSPBU
F
loop_le_bytel
        BTFSS   PIR1,SSPIF
        GOTO    loop_le_bytel
```

```
        MOVF    SSPBUF,w        ; Armazena o byte recebido (low)
        MOVWF   recl
        BTFSC   rech,4
        GOTO    le_serial_disparo_amax
        BTFSC   rech,5
        GOTO    le_serial_disparo_amin
        MOVF    rech,w
        ANDLW   B'00000011'
        MOVWF   anguloh         ; Se chegamos aqui, a informação é um ângulo de disparo
        MOVF    recl,w
        MOVWF   angulol
        BCF     PORTC,6
        GOTO    conv_proc_final

;=============================================================
; A informação recebida é ângulo mínimo
; Neste caso, copia o comando recebido para envh e envl
; e coloca o valor em aminh e aminl

le_serial_disparo_amin
        MOVF    rech,w
        MOVWF   envh
        ANDLW   B'00000011'
        MOVWF   aminh
        MOVF    recl,w
        MOVWF   envl
        MOVWF   aminl
        BCF     PORTC,6
        GOTO    conv_final

;=============================================================
; A informação recebida é ângulo máximo
; Neste caso, copia o comando recebido para envh e envl
; e coloca o valor em amaxh e amaxl

le_serial_disparo_amax
        MOVF    rech,w
        MOVWF   envh
        ANDLW   B'00000011'
        MOVWF   amaxh
        MOVF    recl,w
        MOVWF   envl
        MOVWF   amaxl
        BCF     PORTC,6
        GOTO    conv_final

;*****************************************************************************
;  le_serial_desabilitado
; Esta subrotina lê um comando (amin ou amax) pela serial. Caso o comando seja
; de ângulo de disparo, a informação é desprezada.
;*****************************************************************************

le_serial_desabilitado
        BSF     PORTC,6
        BCF     PIR1,SSPIF      ; Limpa a flag SSPIF
        MOVF    envh,w
        MOVWF   SSPBUF          ; Coloca a parte mais sig. do byte a ser enviado em SSPBU
F
loop_le_des_byteh
        BTFSS   PIR1,SSPIF
        GOTO    loop_le_des_byteh
        MOVF    SSPBUF,w        ; Armazena o byte recebido (high)
        MOVWF   rech
        BCF     PIR1,SSPIF      ; Limpa a flag SSPIF
        MOVF    envl,w
        MOVWF   SSPBUF          ; Coloca a parte mais sig. do byte a ser enviado em SSPBU
F
loop_le_des_bytel
        BTFSS   PIR1,SSPIF
        GOTO    loop_le_des_bytel
        MOVF    SSPBUF,w        ; Armazena o byte recebido (low)
        MOVWF   recl
        BTFSC   rech,4
```

```
        GOTO    le_serial_des_amax
        BTFSC   rech,5
        GOTO    le_serial_des_amin
        MOVF    rech,w
le_serial_des_fim
        BCF     PORTC,6
        RETURN

;=========================================================
; A informação recebida é ângulo mínimo
; Neste caso, copia o comando recebido para envh e envl
; e coloca o valor em aminh e aminl

le_serial_des_amin
        MOVF    rech,w
        MOVWF   envh
        ANDLW   B'00000011'
        MOVWF   aminh
        MOVF    recl,w
        MOVWF   envl
        MOVWF   aminl
        GOTO    le_serial_des_fim

;=========================================================
; A informação recebida é ângulo máximo
; Neste caso, copia o comando recebido para envh e envl
; e coloca o valor em amaxh e amaxl

le_serial_des_amax
        MOVF    rech,w
        MOVWF   envh
        ANDLW   B'00000011'
        MOVWF   amaxh
        MOVF    recl,w
        MOVWF   envl
        MOVWF   amaxl
        GOTO    le_serial_des_fim

;*****************************************************************************
;  aguarda_conversao
; Esta subrotina aguarda o momento de uma nova leitura do angulo de disparo e
; realiza esta leitura.
;*****************************************************************************

aguarda_conversao
        MOVF    TMR1L,w
        MOVWF   contal
        montach
        MOVF    contal,w
        SUBWF   TMR1L,w
        BTFSS   STATUS,Z
        GOTO    aguarda_conversao
        MOVF    proxconvl,w
        SUBWF   contal,w
        MOVF    proxconvh,w
        BTFSS   STATUS,C
        INCFSZ  proxconvh,w
        SUBWF   contah,w
        BTFSS   STATUS,C
        GOTO    aguarda_conversao
        conversao
        RETURN

;*****************************************************************************
;  transforma_acos
; Esta subrotina realiza a transformação acos no angulo de disparo.
; Gasta 30 ciclos aproximadamente, ou seja, 6us.
;*****************************************************************************

transforma_acos
        RRF     anguloh,w
        RRF     angulol,w
        BTFSS   anguloh,1
```

```
        GOTO    acos_menor_90
acos_maior_90                   ; O angulo é maior que 90°, devemos subtrair a posição nã
o linear.
        SUBLW   0xFF
        MOVWF   postabela       ; Armazena a posição de busca na tabela.

        BTFSC   angulol,0
        GOTO    acos_maior_tab2
acos_maior_tab1
        MOVLW   D'01'
        SUBWF   postabela,f
        BTFSS   STATUS,C
        RETURN                  ; Neste caso, angulo não sofre alteração
        MOVLW   HIGH(tabela1)
        MOVWF   PCLATH
        MOVF    postabela,w
        CALL    tabela1
        SUBWF   angulol,f
        BTFSS   STATUS,C
        DECF    anguloh,f
        RETURN
acos_maior_tab2
        SUBLW   0xFF
        BTFSC   STATUS,Z
        RETURN                  ; Neste caso, angulo não sofre alteração
        MOVLW   HIGH(tabela2)
        MOVWF   PCLATH
        MOVF    postabela,w
        CALL    tabela2
        SUBWF   angulol,f
        BTFSS   STATUS,C
        DECF    anguloh,f
        RETURN
acos_menor_90                   ; O angulo é menor que 90°, devemos somar a posição não l
inear
        MOVWF   postabela       ; Armazena a posição de busca na tabela.
        BTFSC   angulol,0
        GOTO    acos_menor_tab2
acos_menor_tab1
        MOVLW   D'01'
        SUBWF   postabela,f
        BTFSS   STATUS,C
        RETURN                  ; Neste caso, angulo não sofre alteração
        MOVLW   HIGH(tabela1)
        MOVWF   PCLATH
        MOVF    postabela,w
        CALL    tabela1
        ADDWF   angulol,f
        BTFSC   STATUS,C
        INCF    anguloh,f
        RETURN
acos_menor_tab2
        SUBLW   0xFF
        BTFSC   STATUS,Z
        RETURN                  ; Neste caso, angulo não sofre alteração
        MOVLW   HIGH(tabela2)
        MOVWF   PCLATH
        MOVF    postabela,w
        CALL    tabela2
        ADDWF   angulol,f
        BTFSC   STATUS,C
        INCF    anguloh,f
        RETURN

;*****************************************************************************
;  Tabela 1 - ARCO-COSSENO - Valores pares
;*****************************************************************************

TABELA1 CODE

tabela1
        ADDWF   PCL,f
        DT        D'027',D'037',D'044',D'050',D'055',D'059',D'062'      ;0
```

```
        DT D'066',D'069',D'071',D'074',D'076',D'078',D'080',D'082'       ;16
        DT D'084',D'085',D'087',D'088',D'090',D'091',D'092',D'093'       ;32
        DT D'094',D'095',D'096',D'097',D'098',D'099',D'099',D'100'       ;48
        DT D'101',D'101',D'102',D'102',D'103',D'103',D'104',D'104'       ;64
        DT D'105',D'105',D'105',D'106',D'106',D'106',D'106',D'107'       ;80
        DT D'107',D'107',D'107',D'107',D'107',D'108',D'108',D'108'       ;96
        DT D'108',D'108',D'108',D'108',D'108',D'108',D'108',D'108'       ;112
        DT D'108',D'107',D'107',D'107',D'107',D'107',D'107',D'107'       ;128
        DT D'107',D'106',D'106',D'106',D'106',D'106',D'105',D'105'       ;144
        DT D'105',D'105',D'104',D'104',D'104',D'104',D'103',D'103'       ;160
        DT D'103',D'102',D'102',D'102',D'101',D'101',D'101',D'100'       ;176
        DT D'100',D'100',D'099',D'099',D'098',D'098',D'098',D'097'       ;192
        DT D'097',D'096',D'096',D'096',D'095',D'095',D'094',D'094'       ;208
        DT D'093',D'093',D'092',D'092',D'091',D'091',D'090',D'090'       ;224
        DT D'089',D'089',D'088',D'088',D'087',D'087',D'086',D'086'       ;240
        DT D'085',D'085',D'084',D'084',D'083',D'083',D'082',D'082'       ;256
        DT D'081',D'080',D'080',D'079',D'079',D'078',D'078',D'077'       ;272
        DT D'076',D'076',D'075',D'075',D'074',D'073',D'073',D'072'       ;288
        DT D'072',D'071',D'070',D'070',D'069',D'069',D'068',D'067'       ;304
        DT D'067',D'066',D'065',D'065',D'064',D'064',D'063',D'062'       ;320
        DT D'062',D'061',D'060',D'060',D'059',D'058',D'058',D'057'       ;336
        DT D'056',D'056',D'055',D'054',D'054',D'053',D'052',D'052'       ;352
        DT D'051',D'050',D'050',D'049',D'048',D'048',D'047',D'046'       ;368
        DT D'046',D'045',D'044',D'044',D'043',D'042',D'042',D'041'       ;384
        DT D'040',D'039',D'039',D'038',D'037',D'037',D'036',D'035'       ;400
        DT D'035',D'034',D'033',D'032',D'032',D'031',D'030',D'030'       ;416
        DT D'029',D'028',D'027',D'027',D'026',D'025',D'025',D'024'       ;432
        DT D'023',D'022',D'022',D'021',D'020',D'020',D'019',D'018'       ;448
        DT D'017',D'017',D'016',D'015',D'015',D'014',D'013',D'012'       ;464
        DT D'012',D'011',D'010',D'009',D'009',D'008',D'007',D'007'       ;480
        DT D'006',D'005',D'004',D'004',D'003',D'002',D'001',D'001'       ;496

;*****************************************************************************
;  Tabela 2 - ARCO-COSSENO - Valores ímpares
;*****************************************************************************

TABELA2 CODE

tabela2
        ADDWF   PCL,f
        DT D'019',D'032',D'041',D'047',D'052',D'057',D'061',D'064'       ;1
        DT D'067',D'070',D'073',D'075',D'077',D'079',D'081',D'083'       ;17
        DT D'085',D'086',D'088',D'089',D'090',D'092',D'093',D'094'       ;33
        DT D'095',D'096',D'097',D'097',D'098',D'099',D'100',D'100'       ;49
        DT D'101',D'102',D'102',D'103',D'103',D'104',D'104',D'104'       ;65
        DT D'105',D'105',D'106',D'106',D'106',D'106',D'107',D'107'       ;81
        DT D'107',D'107',D'107',D'107',D'107',D'108',D'108',D'108'       ;97
        DT D'108',D'108',D'108',D'108',D'108',D'108',D'108',D'108'       ;113
        DT D'108',D'107',D'107',D'107',D'107',D'107',D'107',D'107'       ;129
        DT D'107',D'106',D'106',D'106',D'106',D'106',D'105',D'105'       ;145
        DT D'105',D'105',D'104',D'104',D'104',D'103',D'103',D'103'       ;161
        DT D'103',D'102',D'102',D'102',D'101',D'101',D'100',D'100'       ;177
        DT D'100',D'099',D'099',D'099',D'098',D'098',D'097',D'097'       ;193
        DT D'097',D'096',D'096',D'095',D'095',D'094',D'094',D'094'       ;209
        DT D'093',D'093',D'092',D'092',D'091',D'091',D'090',D'090'       ;225
        DT D'089',D'089',D'088',D'088',D'087',D'087',D'086',D'086'       ;241
        DT D'085',D'085',D'084',D'083',D'083',D'082',D'082',D'081'       ;257
        DT D'081',D'080',D'080',D'079',D'078',D'078',D'077',D'077'       ;273
        DT D'076',D'076',D'075',D'074',D'074',D'073',D'073',D'072'       ;289
        DT D'071',D'071',D'070',D'070',D'069',D'068',D'068',D'067'       ;305
        DT D'066',D'066',D'065',D'064',D'064',D'063',D'063',D'062'       ;321
        DT D'061',D'061',D'060',D'059',D'059',D'058',D'057',D'057'       ;337
        DT D'056',D'055',D'055',D'054',D'053',D'053',D'052',D'051'       ;353
        DT D'051',D'050',D'049',D'049',D'048',D'047',D'047',D'046'       ;369
        DT D'045',D'045',D'044',D'043',D'043',D'042',D'041',D'040'       ;385
        DT D'040',D'039',D'038',D'038',D'037',D'036',D'036',D'035'       ;401
        DT D'034',D'033',D'033',D'032',D'031',D'031',D'030',D'029'       ;417
        DT D'029',D'028',D'027',D'026',D'026',D'025',D'024',D'024'       ;433
        DT D'023',D'022',D'021',D'021',D'020',D'019',D'018',D'018'       ;449
        DT D'017',D'016',D'016',D'015',D'014',D'013',D'013',D'012'       ;465
        DT D'011',D'011',D'010',D'009',D'008',D'008',D'007',D'006'       ;481
        DT D'005',D'005',D'004',D'003',D'003',D'002',D'001'              ;497
```

```
;*******************************************************************************
;  Byte de configuração
;*******************************************************************************

CONFIG CODE

        dw      _HS_OSC & _PWRTE_ON & _WDT_ON & _CP_OFF & _BODEN_OFF
;       dw      _HS_OSC & _PWRTE_ON & _WDT_ON & _BODEN_OFF & _CP_ALL

        END
```

# ANEXO II

**Páginas introdutórias do catálogo do PIC16C774**

MICROCHIP

# PIC16C77X

## 28/40-Pin, 8-Bit CMOS Microcontrollers w/ 12-Bit A/D

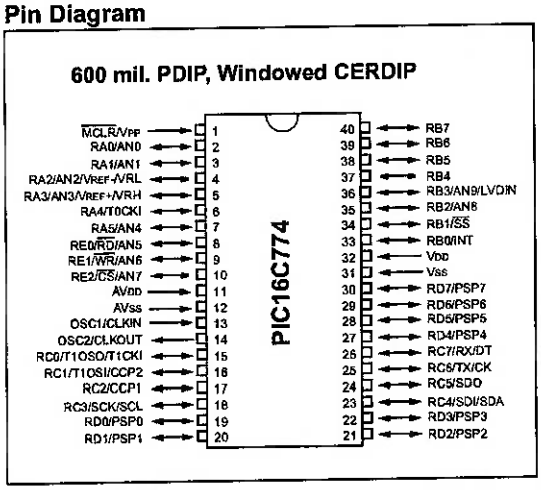
### Microcontroller Core Features:

- High-performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 4K x 14 words of Program Memory, 256 x 8 bytes of Data Memory (RAM)
- Interrupt capability (up to 14 internal/external interrupt sources)
- Eight level deep hardware stack
- Direct, indirect, and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code-protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low-power, high-speed CMOS EPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ISCP)
- Wide operating voltage range: 2.5V to 5.5V
- High Sink/Source Current 25/25 mA
- Commercial and Industrial temperature ranges
- Low-power consumption:
  - < 2 mA @ 5V, 4 MHz
  - 22.5 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

* Enhanced features

### Pin Diagram

**600 mil. PDIP, Windowed CERDIP**



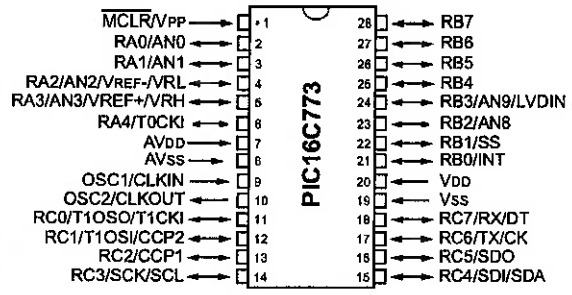

### Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during sleep via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
- Capture is 16-bit, max. resolution is 12.5 ns, Compare is 16-bit, max. resolution is 200 ns, PWM max. resolution is 10-bit
- * 12-bit multi-channel Analog-to-Digital converter
- * On-chip absolute bandgap voltage reference generator
- * Synchronous Serial Port (SSP) with SPI™ (Master Mode) and I²C™
- * Universal Synchronous Asynchronous Receiver Transmitter, supports high/low speeds and 9-bit address mode (USART/SCI)
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls
- * Programmable Brown-out detection circuitry for Brown-out Reset (BOR)
- * Programmable Low-voltage detection circuitry

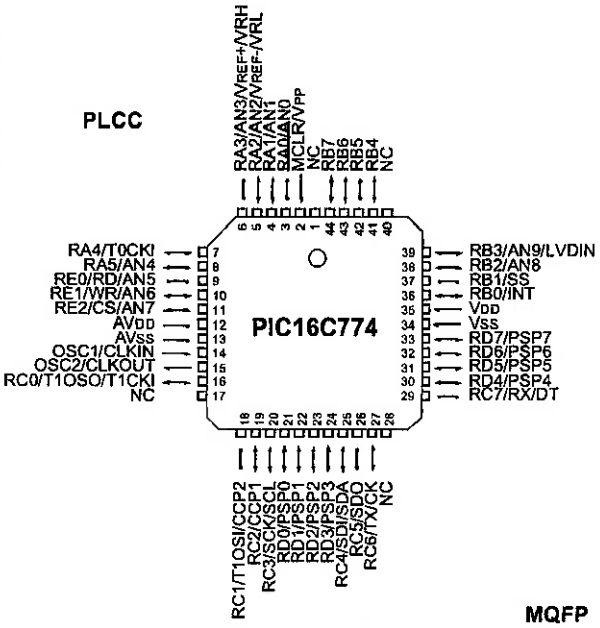
**This is an advanced copy of the data sheet and therefore the contents and specifications are subject to change based on device characterization.**

## Pin Diagrams

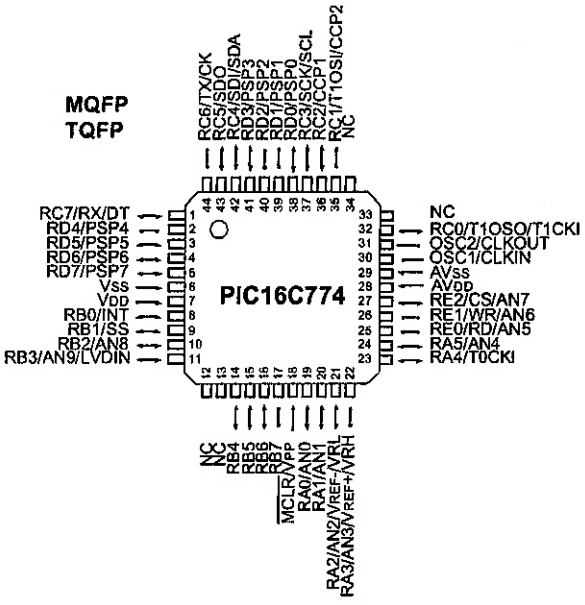
### 300 mil. SDIP, SOIC, Windowed CERDIP, SSOP




### PLCC




### MQFP
### TQFP



| Key Features PICmicro™ Mid-Range Reference Manual (DS33023) | PIC16C773 | PIC16C774 |
|---|---|---|
| Operating Frequency | DC - 20 MHz | DC - 20 MHz |
| Resets (and Delays) | POR, BOR, MCLR, WDT (PWRT, OST) | POR, BOR, MCLR, WDT (PWRT, OST) |
| Program Memory (14-bit words) | 4K | 4K |
| Data Memory (bytes) | 256 | 256 |
| Interrupts | 13 | 14 |
| I/O Ports | Ports A,B,C | Ports A,B,C,D,E |
| Timers | 3 | 3 |
| Capture/Compare/PWM modules | 2 | 2 |
| Serial Communications | MSSP, USART | MSSP, USART |
| Parallel Communications | — | PSP |
| 12-bit Analog-to-Digital Module | 6 input channels | 10 input channels |
| Instruction Set | 35 Instructions | 35 Instructions |

## Table of Contents



*To Our Valued Customers*

**Most Current Data Sheet**

To obtain the most up-to-date version of this data sheet, please check our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number. e.g., DS30000A is version A of document DS30000.

**Errata**

An errata sheet may exist for current devices, describing minor operational differences (from the data sheet) and recommended workarounds. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (602) 786-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

**Corrections to this Data Sheet**

We constantly strive to improve the quality of all our products and documentation. We have spent a great deal of time to ensure that this document is correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears in error, please:

- Fill out and mail in the reader response form in the back of this data sheet.
- E-mail us at webmaster@microchip.com.

We appreciate your assistance in making this a better document.

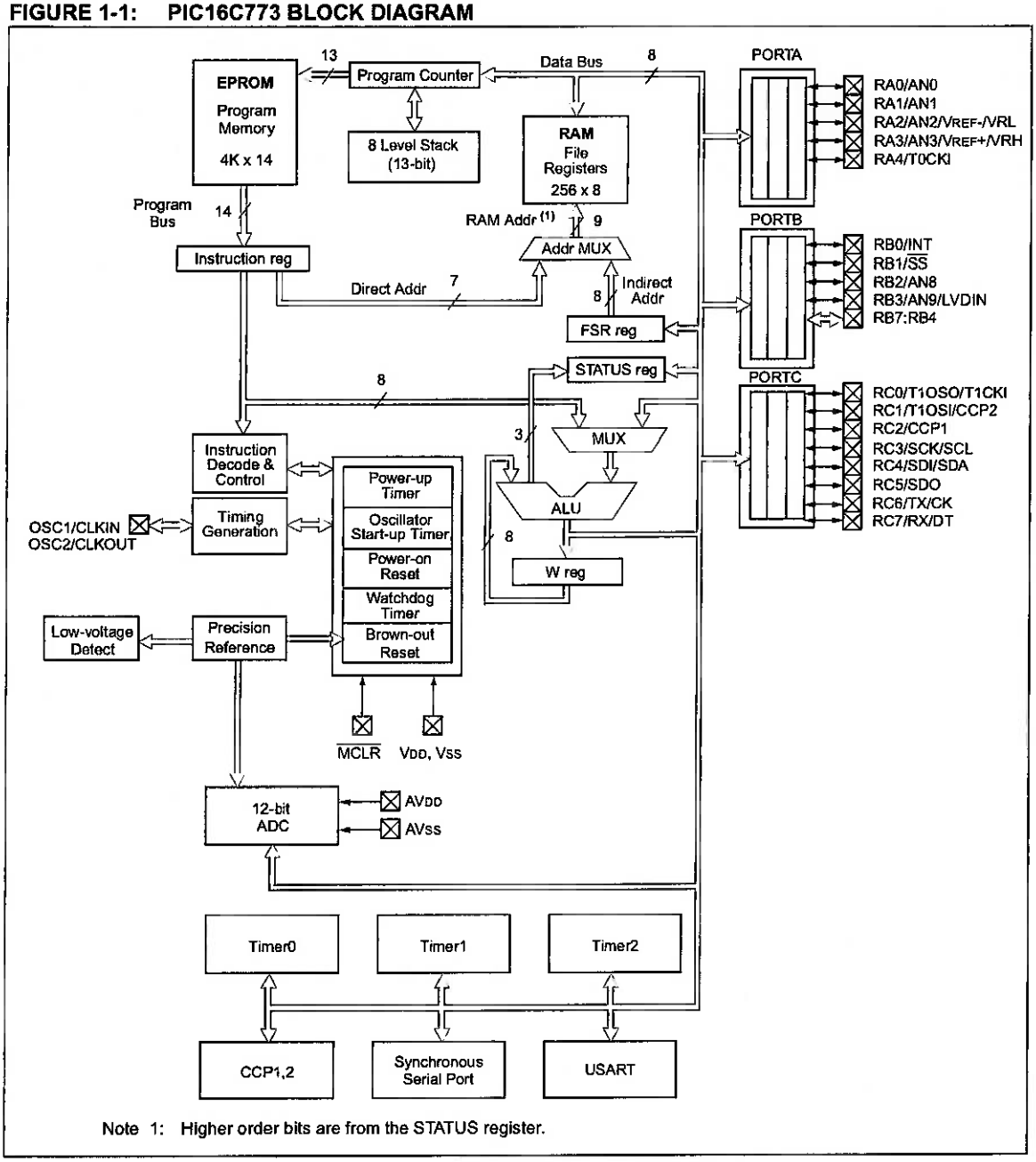
## 1.0 DEVICE OVERVIEW

This document contains device-specific information. 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 website. 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.

There a two devices (PIC16C773 and PIC16C774) covered by this datasheet. The PIC16C773 devices come in 28-pin packages and the PIC16C774 devices come in 40-pin packages. The 28-pin devices do not have a Parallel Slave Port implemented.

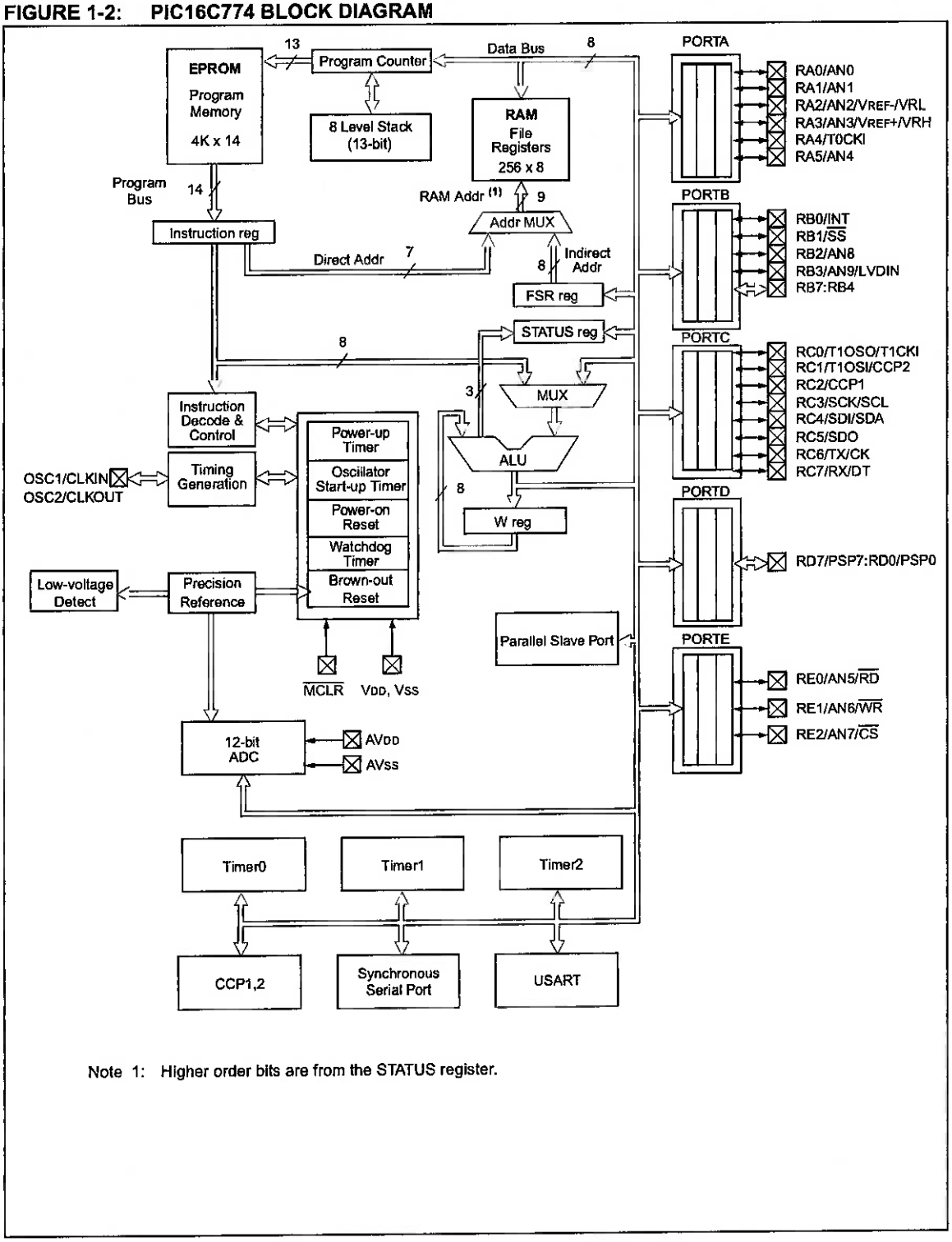
The following two figures are device block diagrams sorted by pin number; 28-pin for Figure 1-1 and 40-pin for Figure 1-2. The 28-pin and 40-pin pinouts are listed in Table 1-1 and Table 1-2, respectively.

**FIGURE 1-1: PIC16C773 BLOCK DIAGRAM**




Note 1: Higher order bits are from the STATUS register.

**FIGURE 1-2: PIC16C774 BLOCK DIAGRAM**




Note 1: Higher order bits are from the STATUS register.

**TABLE 1-1 PIC16C773 PINOUT DESCRIPTION**

| Pin Name | DIP, SSOP, SOIC Pin# | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|
| OSC1/CLKIN | 9 | I | ST/CMOS(3) | Oscillator crystal input/external clock source input. |
| OSC2/CLKOUT | 10 | O | — | Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, the OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate. |
| MCLR/VPP | 1 | I/P | ST | Master clear (reset) input or programming voltage input. This pin is an active low reset to the device. |
| | | | | PORTA is a bi-directional I/O port. |
| RA0/AN0 | 2 | I/O | TTL | RA0 can also be analog input0 |
| RA1/AN1 | 3 | I/O | TTL | RA1 can also be analog input1 |
| RA2/AN2/VREF-/VRL | 4 | I/O | TTL | RA2 can also be analog input2 or negative analog reference voltage input or internal voltage reference low |
| RA3/AN3/VREF+/VRH | 5 | I/O | TTL | RA3 can also be analog input3 or positive analog reference voltage input or internal voltage reference high |
| RA4/T0CKI | 6 | I/O | ST | RA4 can also be the clock input to the Timer0 module. Output is open drain type. |
| | | | | PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. |
| RB0/INT | 21 | I/O | TTL/ST(1) | RB0 can also be the external interrupt pin. |
| RB1/SS | 22 | I/O | TTL/ST(1) | RB1 can also be the SSP slave select |
| RB2/AN8 | 23 | I/O | TTL | RB2 can also be analog input8 |
| RB3/AN9/LVDIN | 24 | I/O | TTL | RB3 can also be analog input9 or the low voltage detect input reference |
| RB4 | 25 | I/O | TTL | Interrupt on change pin. |
| RB5 | 26 | I/O | TTL | Interrupt on change pin. |
| RB6 | 27 | I/O | TTL/ST(2) | Interrupt on change pin. Serial programming clock. |
| RB7 | 28 | I/O | TTL/ST(2) | Interrupt on change pin. Serial programming data. |
| | | | | PORTC is a bi-directional I/O port. |
| RC0/T1OSO/T1CKI | 11 | I/O | ST | RC0 can also be the Timer1 oscillator output or Timer1 clock input. |
| RC1/T1OSI/CCP2 | 12 | I/O | ST | RC1 can also be the Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output. |
| RC2/CCP1 | 13 | I/O | ST | RC2 can also be the Capture1 input/Compare1 output/PWM1 output. |
| RC3/SCK/SCL | 14 | I/O | ST | RC3 can also be the synchronous serial clock input/output for both SPI and $I^2C$ modes. |
| RC4/SDI/SDA | 15 | I/O | ST | RC4 can also be the SPI Data In (SPI mode) or data I/O ($I^2C$ mode). |
| RC5/SDO | 16 | I/O | ST | RC5 can also be the SPI Data Out (SPI mode). |
| RC6/TX/CK | 17 | I/O | ST | RC6 can also be the USART Asynchronous Transmit or Synchronous Clock. |
| RC7/RX/DT | 18 | I/O | ST | RC7 can also be the USART Asynchronous Receive or Synchronous Data. |
| AVSS | 8 | P | | Ground reference for A/D converter |
| AVDD | 7 | P | | Positive supply for A/D converter |
| VSS | 19 | P | — | Ground reference for logic and I/O pins. |
| VDD | 20 | P | — | Positive supply for logic and I/O pins. |

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured for the multiplexed function.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

**TABLE 1-2 PIC16C774 PINOUT DESCRIPTION**

| Pin Name | DIP Pin# | PLCC Pin# | QFP Pin# | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| OSC1/CLKIN | 13 | 14 | 30 | I | ST/CMOS(4) | Oscillator crystal input/external clock source input. |
| OSC2/CLKOUT | 14 | 15 | 31 | O | — | Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate. |
| $\overline{MCLR}$/VPP | 1 | 2 | 18 | I/P | ST | Master clear (reset) input or programming voltage input. This pin is an active low reset to the device. |
| | | | | | | PORTA is a bi-directional I/O port. |
| RA0/AN0 | 2 | 3 | 19 | I/O | TTL | RA0 can also be analog input0 |
| RA1/AN1 | 3 | 4 | 20 | I/O | TTL | RA1 can also be analog input1 |
| RA2/AN2/VREF-/VRL | 4 | 5 | 21 | I/O | TTL | RA2 can also be analog input2 or negative analog reference voltage input or internal voltage reference low |
| RA3/AN3/VREF+/VRH | 5 | 6 | 22 | I/O | TTL | RA3 can also be analog input3 or positive analog reference voltage input or internal voltage reference high |
| RA4/T0CKI | 6 | 7 | 23 | I/O | ST | RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type. |
| RA5/AN4 | 7 | 8 | 24 | I/O | TTL | RA5 can also be analog input4 |
| | | | | | | PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs. |
| RB0/INT | 33 | 36 | 8 | I/O | TTL/ST(1) | RB0 can also be the external interrupt pin. |
| RB1/$\overline{SS}$ | 34 | 37 | 9 | I/O | TTL/ST(1) | RB1 can also be the SSP slave select |
| RB2/AN8 | 35 | 38 | 10 | I/O | TTL | RB2 can also be analog input8 |
| RB3/AN9/LVDIN | 36 | 39 | 11 | I/O | TTL | RB3 can also be analog input9 or input reference for low voltage detect |
| RB4 | 37 | 41 | 14 | I/O | TTL | Interrupt on change pin. |
| RB5 | 38 | 42 | 15 | I/O | TTL | Interrupt on change pin. |
| RB6 | 39 | 43 | 16 | I/O | TTL/ST(2) | Interrupt on change pin. Serial programming clock. |
| RB7 | 40 | 44 | 17 | I/O | TTL/ST(2) | Interrupt on change pin. Serial programming data. |

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured for the multiplexed function.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

**TABLE 1-2 PIC16C774 PINOUT DESCRIPTION (Cont.'d)**

| Pin Name | DIP Pin# | PLCC Pin# | QFP Pin# | I/O/P Type | Buffer Type | Description |
|---|---|---|---|---|---|---|
| | | | | | | PORTC is a bi-directional I/O port. |
| RC0/T1OSO/T1CKI | 15 | 16 | 32 | I/O | ST | RC0 can also be the Timer1 oscillator output or a Timer1 clock input. |
| RC1/T1OSI/CCP2 | 16 | 18 | 35 | I/O | ST | RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output. |
| RC2/CCP1 | 17 | 19 | 36 | I/O | ST | RC2 can also be the Capture1 input/Compare1 output/PWM1 output. |
| RC3/SCK/SCL | 18 | 20 | 37 | I/O | ST | RC3 can also be the synchronous serial clock input/output for both SPI and $I^2C$ modes. |
| RC4/SDI/SDA | 23 | 25 | 42 | I/O | ST | RC4 can also be the SPI Data In (SPI mode) or data I/O ($I^2C$ mode). |
| RC5/SDO | 24 | 26 | 43 | I/O | ST | RC5 can also be the SPI Data Out (SPI mode). |
| RC6/TX/CK | 25 | 27 | 44 | I/O | ST | RC6 can also be the USART Asynchronous Transmit or Synchronous Clock. |
| RC7/RX/DT | 26 | 29 | 1 | I/O | ST | RC7 can also be the USART Asynchronous Receive or Synchronous Data. |
| | | | | | | PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus. |
| RD0/PSP0 | 19 | 21 | 38 | I/O | ST/TTL(3) | |
| RD1/PSP1 | 20 | 22 | 39 | I/O | ST/TTL(3) | |
| RD2/PSP2 | 21 | 23 | 40 | I/O | ST/TTL(3) | |
| RD3/PSP3 | 22 | 24 | 41 | I/O | ST/TTL(3) | |
| RD4/PSP4 | 27 | 30 | 2 | I/O | ST/TTL(3) | |
| RD5/PSP5 | 28 | 31 | 3 | I/O | ST/TTL(3) | |
| RD6/PSP6 | 29 | 32 | 4 | I/O | ST/TTL(3) | |
| RD7/PSP7 | 30 | 33 | 5 | I/O | ST/TTL(3) | |
| | | | | | | PORTE is a bi-directional I/O port. |
| RE0/$\overline{RD}$/AN5 | 8 | 9 | 25 | I/O | ST/TTL(3) | RE0 can also be read control for the parallel slave port, or analog input5. |
| RE1/$\overline{WR}$/AN6 | 9 | 10 | 26 | I/O | ST/TTL(3) | RE1 can also be write control for the parallel slave port, or analog input6. |
| RE2/$\overline{CS}$/AN7 | 10 | 11 | 27 | I/O | ST/TTL(3) | RE2 can also be select control for the parallel slave port, or analog input7. |
| AVss | 12 | 13 | 29 | P | | Ground reference for A/D converter |
| AVDD | 11 | 12 | 28 | P | | Positive supply for A/D converter |
| Vss | 31 | 34 | 6 | P | — | Ground reference for logic and I/O pins. |
| VDD | 32 | 35 | 7 | P | — | Positive supply for logic and I/O pins. |
| NC | — | 1,17,28, 40 | 12,13, 33,34 | | — | These pins are not internally connected. These pins should be left unconnected. |

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

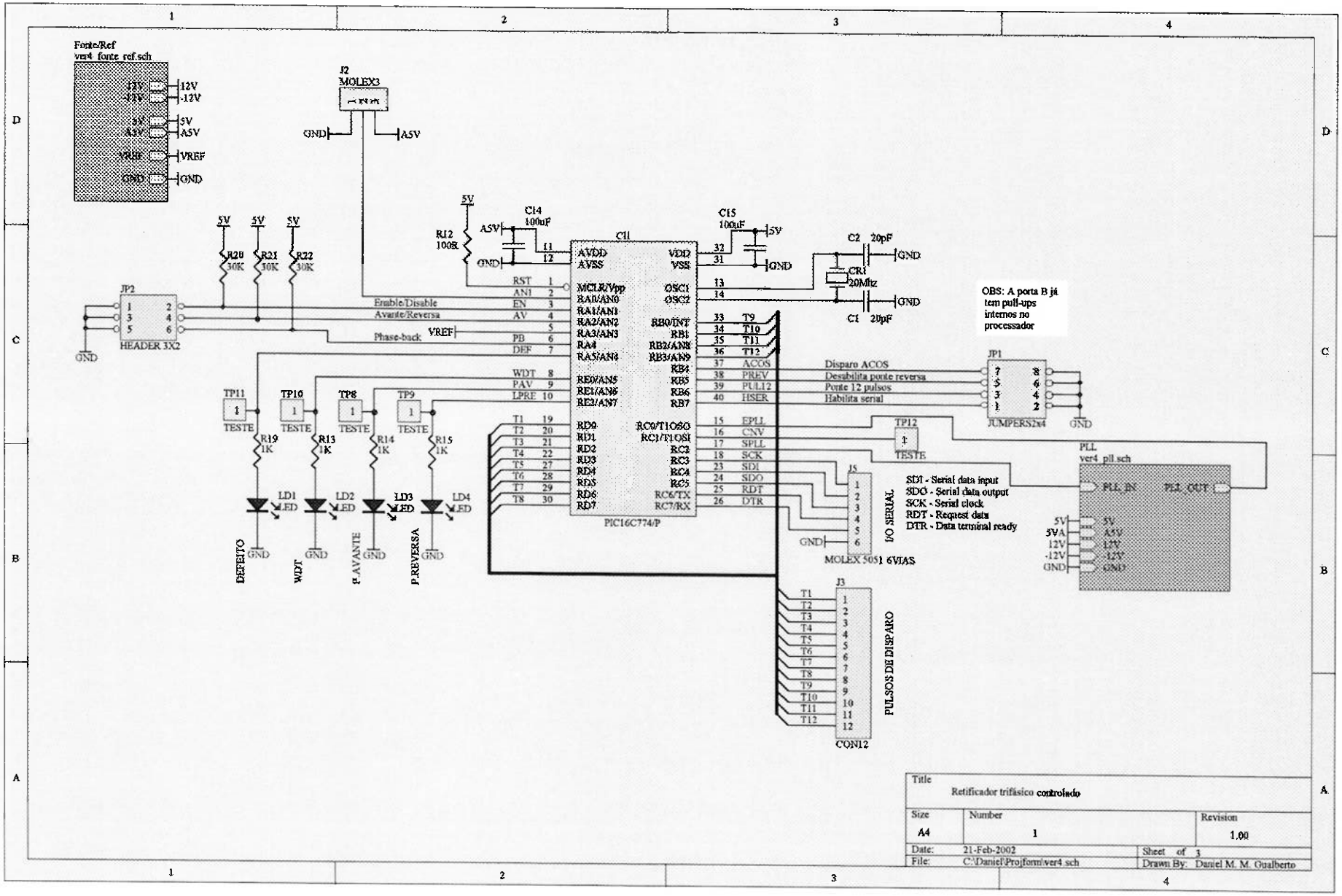
Note 1: This buffer is a Schmitt Trigger input when configured for the multiplexed function.
2: This buffer is a Schmitt Trigger input when used in serial programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

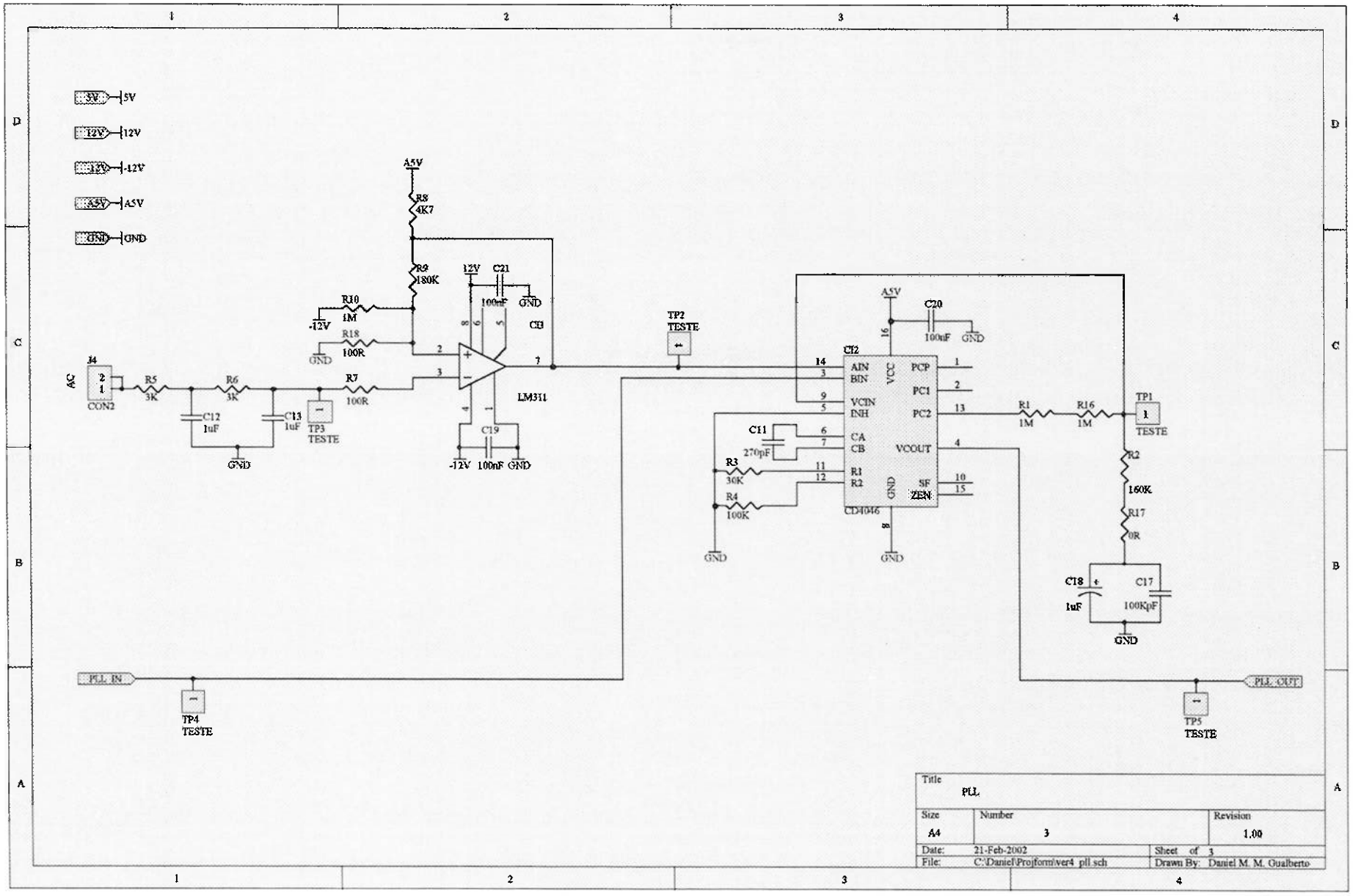
**NOTES:**

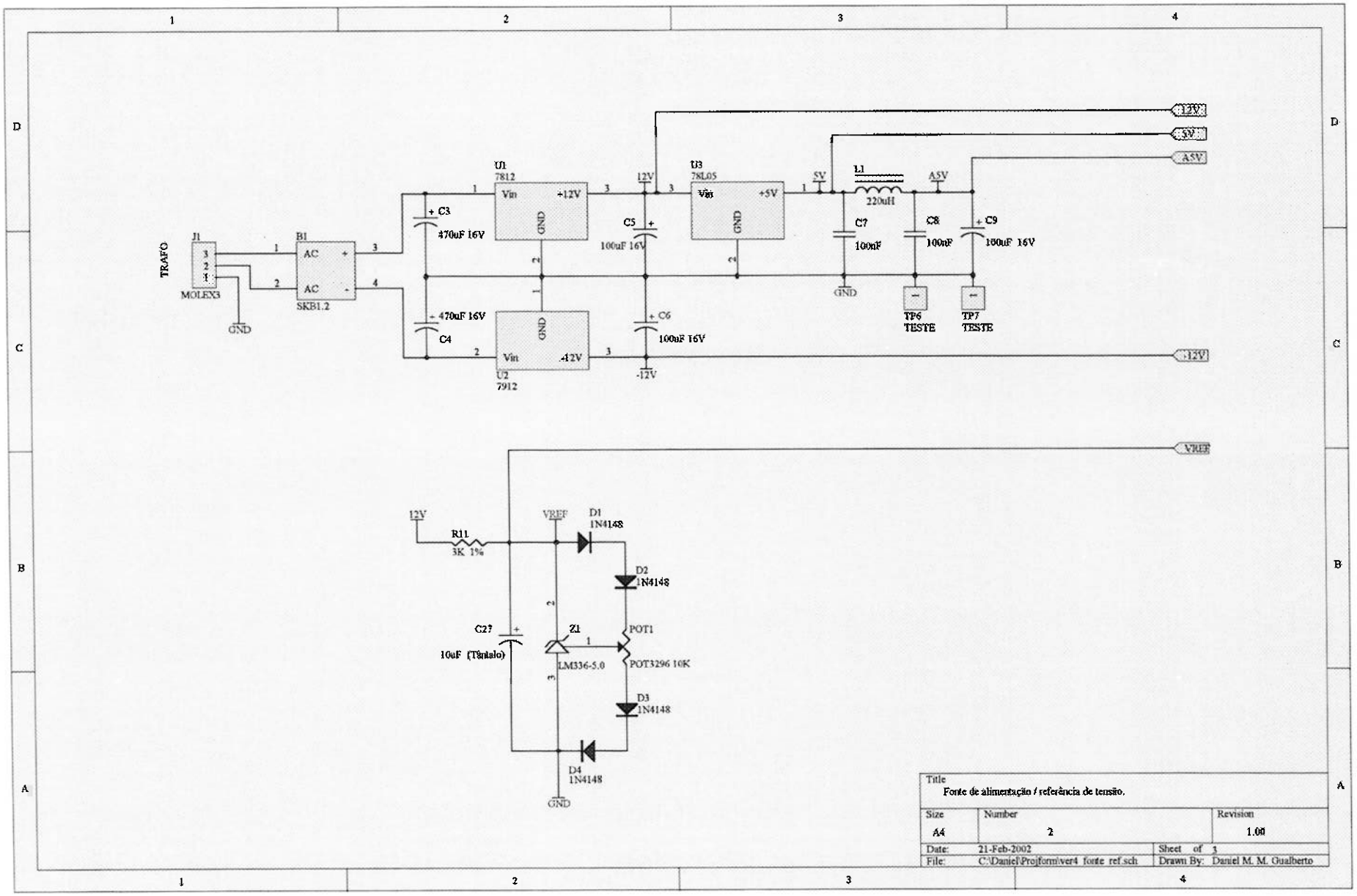
 

# ANEXO III

**Esquemático dos circuito implementados**


Fonte/Ref
ver4_fonte_ref.sch
12V
-12V
5V
ASV
VREF
GND
J2
MOLEX3
GND
ASV
5V
C14
100uF
R12
100R
C11
C15
100uF
C2 20pF
CR1
20Mhz
C1 20pF
OBS: A porta B já tem pull-ups internos no processador
R20 30K
R21 30K
R22 30K
JP2
HEADER 3X2
Enable/Disable
Avante/Reversa
Phase-back
VREF
EN
AV
PB
DEF
RST
MCLR/Vpp
AVDD
AVSS
VDD
VSS
OSC1
OSC2
RB0/INT
RB1
RB2/AN8
RB3/AN9
RB4
RB5
RB6
RB7
RC0/T1OSO
RC1/T1OSI
RC2
RC3
RC4
RC5
RC6/TX
RC7/RX
PIC16C774/P
WDT
PAV
LPRE
RE0/AN5
RE1/AN6
RE2/AN7
RD0
RD1
RD2
RD3
RD4
RD5
RD6
RD7
ACOS
PREV
PUL12
HSER
EPLL
CNV
SPLL
SCK
SDI
SDO
RDT
DTR
Disparo ACOS
Desabilita ponte reversa
Ponte 12 pulsos
Habilita serial
JP1
JUMPERS2x4
TP11
TP10
TP8
TP9
TESTE
R19 1K
R13 1K
R14 1K
R15 1K
LD1 LED
LD2 LED
LD3 LED
LD4 LED
DEFEITO
WDT
P.AVANTE
P.REVERSA
TP12
J5
I/O SERIAL
SDI - Serial data input
SDO - Serial data output
SCK - Serial clock
RDT - Request data
DTR - Data terminal ready
MOLEX 5051 6VIAS
PLL
ver4_pll.sch
PLL_IN
PLL_OUT
5VA
J3
PULSOS DE DISPARO
CON12
Title
Retificador trifásico controlado
Size
A4
Number
1
Revision
1.00
Date:
21-Feb-2002
Sheet of 3
File:
C:\Daniel\Projfonn\ver4.sch
Drawn By:
Daniel M. M. Gualberto


5V
12V
-12V
A5V
GND
A5V
R8
4K7
R9
180K
12V
C21
100nF
GND
R10
1M
-12V
R18
100R
GND
CI3
J4
AC
CON2
R5
3K
R6
3K
C12
1uF
C13
1uF
GND
TP3
TESTE
R7
100R
LM311
C19
-12V
100nF
GND
TP2
TESTE
A5V
C20
100nF
GND
CI2
AIN
BIN
VCC
PCP
PC1
PC2
VCIN
INH
CA
CB
VCOUT
R1
R2
GND
SF
ZEN
CD4046
C11
270pF
R3
30K
R4
100K
GND
GND
R1
1M
R16
1M
TP1
TESTE
R2
160K
R17
0R
C18
1uF
C17
100KpF
GND
PLL IN
TP4
TESTE
PLL OUT
TP5
TESTE
Title
PLL
Size
A4
Number
3
Revision
1.00
Date:
21-Feb-2002
Sheet of 3
File:
C:\Daniel\Projform\ver4_pll.sch
Drawn By:
Daniel M. M. Gualberto


TRAFO
J1
MOLEX3
B1
SKB1.2
AC
AC
GND
C3
470uF 16V
C4
470uF 16V
U1
7812
Vin
+12V
GND
U2
7912
Vin
-12V
GND
12V
C5
100uF 16V
C6
100uF 16V
-12V
U3
78L05
Vin
+5V
GND
5V
L1
220uH
A5V
C7
100nF
C8
100nF
C9
100uF 16V
GND
TP6
TESTE
TP7
TESTE
12V
5V
A5V
-12V
VREF
12V
R11
3K 1%
VREF
D1
1N4148
D2
1N4148
C27
10uF (Tântalo)
Z1
LM336-5.0
POT1
POT3296 10K
D3
1N4148
D4
1N4148
GND
Title
Fonte de alimentação / referência de tensão.
Size
A4
Number
2
Revision
1.00
Date:
21-Feb-2002
Sheet of 3
File:
C:\Daniel\Proj\form\ver4 fonte ref.sch
Drawn By:
Daniel M. M. Gualberto

# ANEXO IV

**Valores de inicialização dos registradores do PIC16C774**

## Anexo IV – Valores de inicialização dos registradores do PIC16C774

### Port A:

| *Pino* | *Função* | *TRISA* | *PORTA* |
|---|---|---|---|
| RA0 | Entrada – AN1 | 1 | 0 |
| RA1 | Entrada - EN | 1 | 0 |
| RA2 | Entrada - AV | 1 | 0 |
| RA3 | Entrada - VREF | 1 | 0 |
| RA4 | Entrada - PB | 1 | 0 |
| RA5 | Saída – DEF (Led) | 0 | 0 |
| RA6 | X | 0 | 0 |
| RA7 | X | 0 | 0 |

### Port B:

| *Pino* | *Função* | *TRISB* | *PORTB* |
|---|---|---|---|
| RB0 | Saída - T9 | 0 | 0 |
| RB1 | Saída - T10 | 0 | 0 |
| RB2 | Saída - T11 | 0 | 0 |
| RB3 | Saída - T12 | 0 | 0 |
| RB4 | Entrada - ACOS | 1 | 0 |
| RB5 | Entrada - PREV | 1 | 0 |
| RB6 | Entrada - PUL12 | 1 | 0 |
| RB7 | Entrada - HSER | 1 | 0 |

### Port C

| *Pino* | *Função* | *TRISC* | *PORTC* |
|---|---|---|---|
| RC0 | Entrada - EPLL | 1 | 0 |
| RC1 | Saída - CNV | 0 | 0 |
| RC2 | Saída - SPLL | 0 | 0 |
| RC3 | Saída - SCK | 0 | 0 |
| RC4 | Entrada - SDI | 1 | 0 |
| RC5 | Saída - SDO | 0 | 0 |
| RC6 | Saída - RDT | 0 | 0 |
| RC7 | Entrada - DTR | 1 | 0 |

## Port D:

| *Pino* | *Função* | *TRISD* | *PORTD* |
|---|---|---|---|
| RD0 | Saída - T1 | 0 | 0 |
| RD1 | Saída - T2 | 0 | 0 |
| RD2 | Saída - T3 | 0 | 0 |
| RD3 | Saída - T4 | 0 | 0 |
| RD4 | Saída - T5 | 0 | 0 |
| RD5 | Saída - T6 | 0 | 0 |
| RD6 | Saída - T7 | 0 | 0 |
| RD7 | Saída - T8 | 0 | 0 |

## Port E:

| *Pino* | *Função* | *TRISD* | *PORTD* |
|---|---|---|---|
| RE0 | Saída – WDT (Led watch dog) | 0 | 0 |
| RE1 | Saída – PAV (Led ponte avante) | 0 | 0 |
| RE2 | Saída – PRE (Led ponte reversa) | 0 | 0 |

## OPTION_REG

| *Bit* | *Função* | *Valor* |
|---|---|---|
| 7 | Habilita Pull-Ups p/ PortB | 0 |
| 6 | INT na borda de subida | 1 |
| 5 | Timer 0 usa clock interno | 0 |
| 4 | Timer 0 usa borda de subida | 1 |
| 3 | Prescaler para WDT | 1 |
| 2 | Prescaler de 1:1 p/ WDT | 0 |
| 1 | | 0 |
| 0 | | 0 |

## INTCON

| *Bit* | *Função* | *Valor* |
|---|---|---|
| 7 | GEI – global interrupt enable | 0 |
| 6 | PEIE – Peripheral interrupt enable | 1 |
| 5 | T0IE – TMR0 overflow Interrupt enable | 0 |
| 4 | IINTE – RB0/INT External interrupt enable | 0 |
| 3 | RBIE – RB port change interrupt enable | 0 |

| Bit | Função | Valor |
|---|---|---|
| 2 | T0IF – TMR0 overflow interrupt flag | 0 |
| 1 | INTF – External interrupt flag | 0 |
| 0 | RBIF – RB port change interrupt flag | 0 |

## PIE1

| Bit | Função | Valor |
|---|---|---|
| 7 | PSP interrupt enable | 0 |
| 6 | A/D converter interrupt enable | 0 |
| 5 | USART receive interrupt enable | 0 |
| 4 | USART transmit interrupt enable | 0 |
| 3 | Syncronous serial port interrupt enable | 0 |
| 2 | CCP1IE – CCP1 interrupt enable | 1 |
| 1 | TMR2 to PR2 match interrupt enable | 0 |
| 0 | TMR1 overflow interrupt enable | 0 |

## PIR1

| Bit | Função | Valor |
|---|---|---|
| 7 | PSP interrupt flag | 0 |
| 6 | A/D converter interrupt flag | 0 |
| 5 | USART receive interrupt flag | 0 |
| 4 | USART transmit interrupt flag | 0 |
| 3 | Syncronous serial port interrupt flag | 0 |
| 2 | CCP1IE – CCP1 interrupt flag | 0 |
| 1 | TMR2 to PR2 match interrupt flag | 0 |
| 0 | TMR1 overflow interrupt flag | 0 |

## PIE2

| Bit | Função | Valor |
|---|---|---|
| 7 | Low voltage detect interrupt enable | 0 |
| 6 | X | 0 |
| 5 | X | 0 |
| 4 | X | 0 |
| 3 | Bus collision interrupt enable | 0 |

| Bit | Função | Valor |
|---|---|---|
| 2 | X | 0 |
| 1 | X | 0 |
| 0 | CCP2 interrupt enable | 0 |

## Voltage Reference Module

- Desabilitado
- Low Voltage detect desabilitado

## LVDCON

| Bit | Função | Valor |
|---|---|---|
| 7 | X | 0 |
| 6 | X | 0 |
| 5 | X | 0 |
| 4 | Low-Voltage Detect Power Enable | 0 |
| 3 | 2,5 V (Limite arbitrário) | 0 |
| 2 | | 1 |
| 1 | | 0 |
| 0 | | 0 |

## REFCON

| Bit | Função | Valor |
|---|---|---|
| 7 | VRH enable | 0 |
| 6 | VRL enable | 0 |
| 5 | VRH output enable | 0 |
| 4 | VRL output enable | 0 |
| 3 | X | 0 |
| 2 | X | 0 |
| 1 | X | 0 |
| 0 | X | 0 |

## Módulo A/D

- Referência superior externa (AN3)
- Referência inferior interna (GND)
- 1 canal
- A/D ativo

## ADCON0

| *Bit* | *Função* | *Valor* |
|---|---|---|
| 7 | Fosc/32 | 1 |
| 6 | | 0 |
| 5 | Canal 0 selecionado | 0 |
| 4 | | 0 |
| 3 | | 0 |
| 2 | A/D not in progress | 0 |
| 1 | Canal 0 selecionado | 0 |
| 0 | A/D ligado | 1 |

## ADCON1

| *Bit* | *Função* | *Valor* |
|---|---|---|
| 7 | Resultado justificado pela direita | 1 |
| 6 | Vref+ externa | 0 |
| 5 | Vref- GND | 1 |
| 4 | | 1 |
| 3 | 1 Canal | 1 |
| 2 | | 1 |
| 1 | | 1 |
| 0 | | 0 |

## Master Syncronous Serial Port

## SSPSTAT

| *Bit* | *Função* | *Valor* |
|---|---|---|
| 7 | Entrada amostrada no meio do ciclo | 0 |
| 6 | Dado transmitido na borda de subida | 1 |
| 5 | X | 0 |
| 4 | X | 0 |
| 3 | X | 0 |
| 2 | X | 0 |
| 1 | X | 0 |
| 0 | X | 0 |

## SSPCON

| *Bit* | *Função* | *Valor* |
|---|---|---|
| 7 | Não detectar colisão | 0 |
| 6 | Overflow de recepção | 0 |
| 5 | Habilita porta serial síncrona | 1 |
| 4 | Estado de espera (IDLE) | 0 |
| 3 | SPI Master Mode<br>Clock=Fosc/4 = 5Mhz | 0 |
| 2 | | 0 |
| 1 | | 0 |
| 0 | | 0 |

## Timer1 (TMR1 Module)

## T1CON

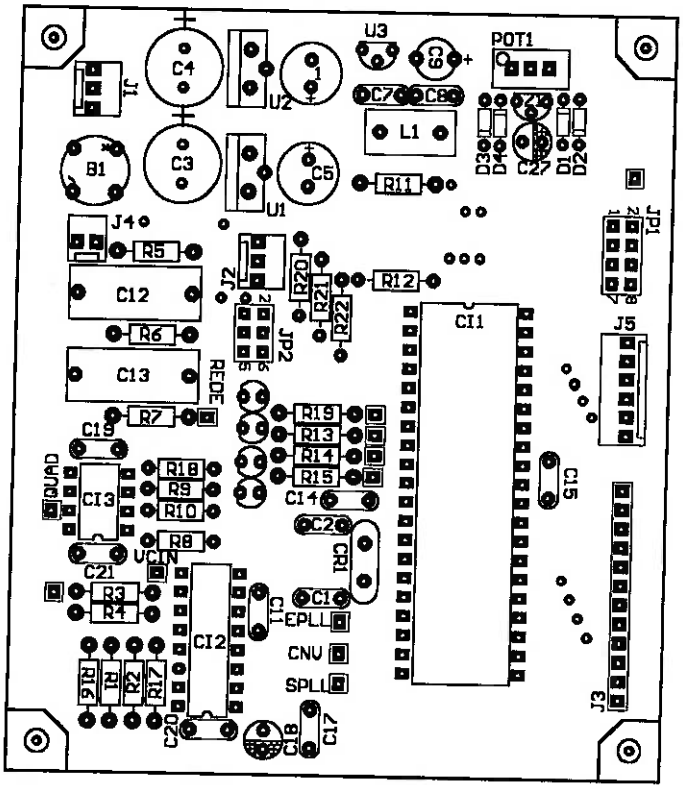
| *Bit* | *Função* | *Valor* |
|---|---|---|
| 7 | X | 0 |
| 6 | X | 0 |
| 5 | Prescale 1:1 | 0 |
| 4 | | 0 |
| 3 | TMR1 Oscilador desativado | 0 |
| 2 | Sincronizar clock | 0 |
| 1 | Borda de subida | 1 |
| 0 | Ativa TMR1 | 1 |

## CCP1CON

| *Bit* | *Função* | *Valor* |
|---|---|---|
| 7 | X | 0 |
| 6 | X | 0 |
| 5 | X | 0 |
| 4 | X | 0 |
| 3 | Compare Mode<br>Set output on match | 1 |
| 2 | | 0 |
| 1 | | 0 |
| 0 | | 0 |

# ANEXO V

**Placa de circuito de teste (PCB)**


U3
POT1
C4
C9
J1
U2
C7
C8
L1
D3
D4
C6
C7
D1
D2
B1
C3
C5
U1
R11
JP1
J4
R5
J2
R20
R21
R12
C12
R22
CI1
J5
JP2
R6
C13
REDE
R19
R7
R13
C19
R14
R18
R15
QUAD
CI3
R9
C14
C15
R10
C2
CR1
R8
VCIN
C21
C1
R3
C11
EPLL
R4
CI2
CNV
R16
R1
R2
R17
SPLL
J3
C20
C18
C17