UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEMIC UNICAMP

"SISTEMA DE CONTROLE MICROPROCESSADO PARA TANQUES PARA *WET-ETCHING/CLEANING* EM MICROELETRÔNICA"

AUTORA: *Silvia Elisabeth Sauaia Lopes*
ORIENTADOR: *Dr. José Antonio Siqueira Dias*

Dissertação apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de "Mestre em Engenharia Elétrica".

DEZEMBRO DE 1995

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

L881s

Lopes, Silvia Elisabeth Sauaia
Sistema de controle microprocessado para tanques para Wet-Etching/Cleaning em microeletrônica / Silvia Elisabeth Sauaia Lopes.-- Campinas, SP: [s.n.], 1996.

Orientador: José Antonio Siqueira Dias.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica.

1. Controladores PID. 2. Controle de temperatura. 3. * 8051/8031 (Microcontrolador). I. Dias, José Antonio Siqueira. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica. III. Titulo.

DEDICATÓRIA

Dedico este trabalho aos meus pais, Manuel e Julia, e ao meu esposo Ramiro.

AGRADECIMENTOS

Ao Professor Dr. José Antonio Siqueira Dias, pela confiança e oportunidade de sua orientação.

Ao Professor Carlos Alberto Murari Pinheiro, da Escola Federal de Engenharia de Itajubá (EFEI), pelos ensinamentos, sem os quais não seria possível a realização deste trabalho, pela amizade e compreensão.

Aos amigos do laboratório do DON (EFEI), Túlio e Graça, pela colaboração.

Aos colegas do CEB da UNICAMP, especialmente ao Alexandre, pela oportunidade da utilização do laboratório e pelo apoio.

Aos saudosos amigos Carlos Gilberto de Figueiredo e Pedro Markevicius, pela ajuda e incentivo ao longo deste trabalho e pela oportunidade de sua amizade.

RESUMO

Tanques para banho à temperatura constante necessitam de um sistema de controle para monitoração e controle de sua temperatura de operação e demais funções. O controle da temperatura deve ser rígido e preciso; condições de alarme e desligamento automático devem ser previstos.

O presente trabalho pretende estudar, implementar e testar um protótipo de um sistema de controle microprocessado para tais tanques. Este trabalho apresenta um controlador digital do tipo PID, baseado na arquitetura do microcontrolador 8051 da Intel, com alto desempenho, robusto, eficiente e simples, características estas comprovadas através de testes práticos realizados no final do projeto.

ABSTRACT

Tanks for constant temperature bath need a temperature and related functions monitoring and control system. Temperature control must be constant and precise; alarm and automatic switching off conditions must be provided.

This work is to study, implement and test a microprocessor controller's prototype for such tanks. This work presents a digital controller with a PID control scheme, based in the architecture of the Intel's 8051 microcontroller, with high performance, strong, efficient and simple, characteristics verified through practical tests made at the end of the project.

ÍNDICE

CAPÍTULO 1 - INTRODUÇÃO

1.1 - MICROPROCESSADORES EM CONTROLE DE PROCESSOS

Ninguém desconhece, hoje em dia, a importância adquirida pelos dispositivos produzidos pela indústria da microeletrônica, em particular, os tão falados microprocessadores. Da mesma forma é impossível ignorar a participação crescente da informática em praticamente todos os ramos da atividade humana. Neste caso o microprocessador veio realimentar fortemente a tendência existente, possibilitando, por seu baixo custo, a disseminação intensa das aplicações de computadores digitais.

A área de Controle de Processos que, já por tradição, é das que incorporam com maior rapidez as inovações tecnológicas, com o advento dos microprocessadores, teve possibilidade de aplicar em nível industrial as técnicas mais recentemente desenvolvidas. O impacto foi de tal ordem que, em vários de seus setores, é nítida a sensação de ter uma ferramenta muito poderosa, que as metodologias existentes não exploram completamente. Isto se traduz no grande esforço de pesquisa realizado em todas as instituições ligadas à área, em todo o mundo.

A simplicidade da idéia central da automação é o controle com realimentação, que pode ser resumido na sequência: mede-se o estado do processo a ser controlado, compara-se com o valor de referência (condição desejada deste estado) e altera-se o estado para que se reduza o erro resultante da comparação, acionando o controle. Um sistema para o controle de processos dinâmicos em malha-fechada, a grosso modo, pode ser esquematizado conforme a Figura 1.1.

Levando-se em consideração este esquema, o bloco controlador era constituído, quase que exclusivamente, por elementos contínuos ou analógicos até o fim da década de 50. Durante a década de 60 iniciaram-se as primeiras realizações digitais (através de computadores), as quais se mostraram muito promissoras, mas também muito dispendiosas. Ao final da década de 60 e no início da década de 70, o preço dos computadores sofreu uma sensível redução pelo emprego da tecnologia de

circuitos integrados, ainda que em pequena escala. Este fato consolidou de vez o emprego de controladores à base de sistemas de processamento digital da informação [1].

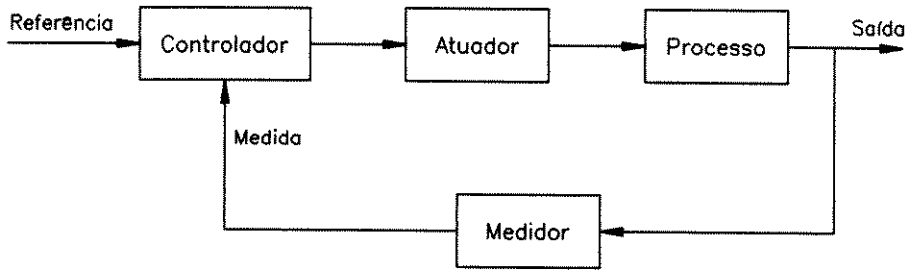


Fig. 1.1 - Esquema básico de um processo controlado.

A segunda metade da década de 70 e o início da década de 80 presenciaram o aparecimento dos processadores digitais de baixíssimo custo, o que popularizou sobre maneira a utilização de controladores digitais e abriu novas perspectivas para o tratamento de sistemas complexos através, por exemplo, da descentralização do processamento [1].

Paralelamente ao desenvolvimento da tecnologia construtiva dos sistemas digitais verificou-se o estabelecimento da metodologia adequada à utilização destes sistemas na área de Controle de Processos. As pesquisas dirigiram-se inicialmente à compatibilização das diferentes características do processo (contínuo por natureza) e do computador (discreto por natureza), visando a aplicação dos métodos de controle já conhecidos para sistemas contínuos.

1.1.1 - Aplicações em Controle de Processos Industriais

A utilização de computadores na área de Controle de Processos evoluiu de maneira muito rápida nos últimos vinte anos. São empregados em funções que variam desde a simples substituição de controladores analógicos, controle digital direto, até as de grande complexidade como controladores com otimização, controladores adaptativos, etc.

Com relação à forma da automação já se comentou anteriormente a crescente tendência da substituição de elementos

analógicos por sistemas digitais mais flexíveis e capazes. Os microprocessadores são utilizados para implementar circuitos dedicados de controle com uma série de vantagens em relação a circuitos e lógicas discretos.

O uso de microprocessadores nos equipamentos deu início a uma nova era na informática industrial, resultando em equipamentos com recursos suficientes para atender às mais complexas aplicações. Entre estes recursos, destaca-se a programabilidade.

A grande vantagem dos sistemas de controle programáveis é que com simples modificações em programas, pode-se alterar substancialmente as características do controle, com alteração mínima do *hardware*.

1.1.2 - Programação

Quando surgiram os primeiros microprocessadores, seu uso era muito prejudicado pelo fato de sua programação ser feita basicamente em linguagem de máquina (códigos em hexadecimal) ou em *assembly* (mnemônicos). Esta situação está no entanto sendo modificada muito rapidamente, pela crescente evolução na própria área de eletrônica, e a programação pode ser feita também em linguagens de alto nível.

1.1.3 - Controladores Discretos com Comportamento PID

O esquema de controle apresentado na Figura 1.1 pode ser realizado de múltiplas formas por processadores digitais, desde um simples controlador liga-desliga (do inglês: *On-Off*) até um complexo algoritmo adaptativo. Os controladores do tipo PID (Proporcional-mais-Integral-mais-Derivativo), por sua grande utilização industrial foram dos primeiros a ter um assemelhado digital.

Neste trabalho, será utilizada a técnica de controle digital do tipo PID baseada nos microcontroladores dedicados, devido aos diversos recursos disponíveis neste dispositivos e, também, devido às facilidades de implementação de um sistema de controle digital simples porém confiável.

1.2 - OBJETIVOS DO TRABALHO

Este trabalho propõe o estudo e implementação de um circuito discreto microprocessado que deverá monitorar e controlar as funções de um tanque para *wet-etching/cleaning*. O tanque comercial da HEATEFLEX para banho à temperatura constante foi usado como modelo para o presente trabalho. A seguir, são apresentadas as suas principais características:

- Modelo: UCT200-2 (SIZE-2), 1KW/120VAC, 50/60Hz
- Volume: 5,5 litros
- Elemento térmico: 120VAC, 1000W, 8,3A, 14,4 ohms
- Sensor de temperatura (PROCESS): Termopar tipo J
- Sensor de temperatura excessiva (HI-LIMIT): Termopar tipo J
- Sensor de nível: Tubo sensor de nível do líquido a ar
- Temperatura de operação máxima: 180°C

A partir dos estudos realizados, chegou-se a um sistema de controle baseado na arquitetura do microcontrolador 8051 da Intel.

O principal objetivo deste sistema é controlar precisamente a temperatura do banho de ácido, de modo que a mesma permaneça constante e com o valor previamente determinado durante o tempo que durar o banho. Este rígido controle sobre a temperatura é uma exigência dos processos realizados no tanque, bem como uma variação da temperatura de operação de no máximo 1%. A faixa de temperatura deste controlador será de 0,0°C a 128,0°C e sua resolução de 0,5°C. As principais temperaturas envolvidas no processo serão:

- 100°C: temperatura de operação para a maior parte dos processos a serem realizados (microeletrônica)
- 125°C: temperatura máxima (desencadeia proteção geral)
- 60°C: temperatura de abertura da válvula do dreno

Para a monitoração da temperatura será utilizado um sensor de temperatura de precisão e uma malha de controle PID, implementada por *software*, proporcionará o controle da

temperatura do banho. A informação do nível do líquido será dada por um sensor de nível para ambiente agressivo. Para maior segurança no escoamento deverá ser previsto uma função, que estabelecerá um intertravamento entre a temperatura da solução e a abertura da válvula do dreno, visando não danificar a tubulação.

Este sistema deverá possuir ainda teclado, *display* e LEDs, para atuação, visualização e sinalização, respectivamente; objetivando maior interface com o usuário e melhor comodidade para entrada de dados e referências. A fim de aumentar a segurança do equipamento, deverão ser implementadas funções tais como: alarmes, desligamento automático, entre outras.

1.3 - ORGANIZAÇÃO DO TRABALHO

Neste capítulo foi feita uma breve introdução à aplicação de microprocessadores em sistemas de controle de processos. As características do tanque comercial usado como modelo e os objetivos deste trabalho também estão contidos neste capítulo.

O Capítulo 2 apresenta o controlador microprocessado para o tanque para *wet-etching/cleaning* desenvolvido neste trabalho, bem como seu princípio de operação. São discutidas também neste capítulo as principais características e necessidades deste sistema, tais como, temperaturas críticas, alarmes, interface com operador, monitoramento do nível do líquido, intertravamento entre a temperatura do banho e a abertura do dreno, etc.

No Capítulo 3 é descrito o *hardware* do controlador digital, sendo dada maior ênfase ao microcontrolador 8031 e aos CIs 8155 e 8279, por se tratarem de circuitos integrados programáveis. No Apêndice A é apresentado o circuito completo do protótipo construído neste trabalho.

No Capítulo 4 os sistemas de controle de processos digitais em malha-fechada são discutidos mais detalhadamente. A composição física do sistema de controle e a discretização das ações de controle proporcional, integral e derivativa, necessária para a implementação do algoritmo de controle, também são

discutidas neste capítulo. Finalmente é apresentado o programa de controle PID elaborado neste projeto, na forma de fluxograma para uma melhor compreensão. O *software* do controlador PID escrito em linguagem C encontra-se no Apêndice B.

No Capítulo 5 são apresentados os resultados práticos obtidos com o sistema de controle proposto e também os valores dos parâmetros ajustáveis. As conclusões e sugestões para melhoramento do protótipo são discutidas neste capítulo.

CAPÍTULO 2 - SISTEMA DE CONTROLE MICROPROCESSADO

2.1 - PRINCÍPIO DE FUNCIONAMENTO

Este sistema de controle pertence a uma nova geração de controladores baseados em microprocessador. Foi projetado de modo a tirar vantagem dos mais recentes avanços tecnológicos, com o objetivo de proporcionar uma atuação em controle de processos mais precisa, eficiente e confiável. Seu processador é o microcontrolador 8031 da Intel, que, além das funções do microprocessador, incorpora portas de I/O (*Input/Output*), temporizadores e memória interna.

Os tanques para *wet-etching/cleaning* contêm um elemento resistivo (aquecedor) imerso que, ao ser percorrido por uma corrente elétrica, eleva a temperatura do banho de ácido. Este elemento deve estar em solução antes de ser energizado e deve ser mantido mergulhado na mesma enquanto energizado. Sérios danos podem ocorrer caso o equipamento seja ligado sem que o aquecedor esteja imerso em solução. Para atender estas exigências, foi previsto um intertravamento (por *software* e por *hardware*) com o nível do líquido, a fim de garantir uma operação segura.

A Figura 2.1 mostra o tanque e o controlador, de uma forma esquemática, para facilitar o entendimento do sistema:

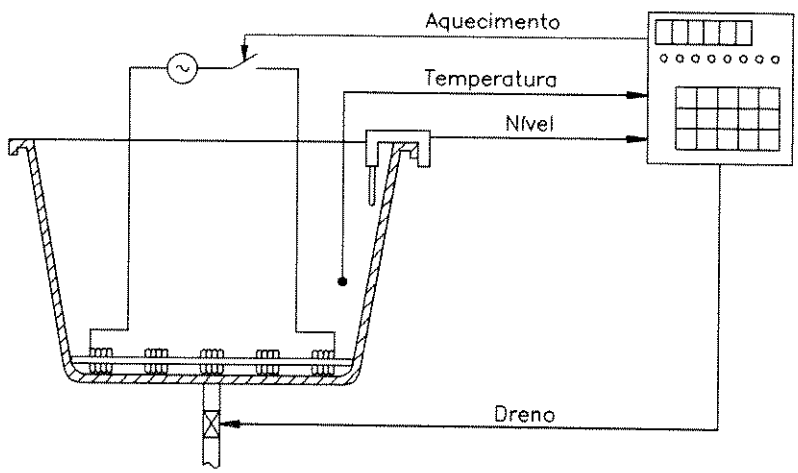


Fig. 2.1 - Esquema simplificado do tanque para banho de ácido com o controlador de temperatura.

Pode-se notar que o controlador possui duas entradas principais: **Nível** e **Temperatura**. A primeira é proveniente de um circuito sensor de nível de líquido remoto, que informa ao controlador se o nível do líquido está adequado ou baixo. A segunda é a informação da temperatura do líquido. O controlador monitora a temperatura do banho utilizando um sensor de precisão e controla a temperatura com uma malha de controle de três modos padrão (PID).

O controlador possui duas saídas principais: **Dreno** e **Aquecimento**. Ao receber um comando para acionar o escoamento do líquido, o controlador verifica se a temperatura se encontra abaixo de um valor pré-definido (60°C). Caso afirmativo, envia um comando para o circuito da válvula do dreno, permitindo sua abertura. Isto assegura que a tubulação não será danificada. A segunda saída é responsável pelo aquecimento do líquido. Funciona como uma "chave" que controla o aquecedor, ligando e desligando a energia, a fim de manter a temperatura do banho constante e com o valor previamente especificado.

Várias funções de alarme e *status* foram incorporadas para monitorar os diversos parâmetros do sistema.

A Figura 2.2 a seguir mostra o *hardware* do controlador na forma de diagrama de blocos:

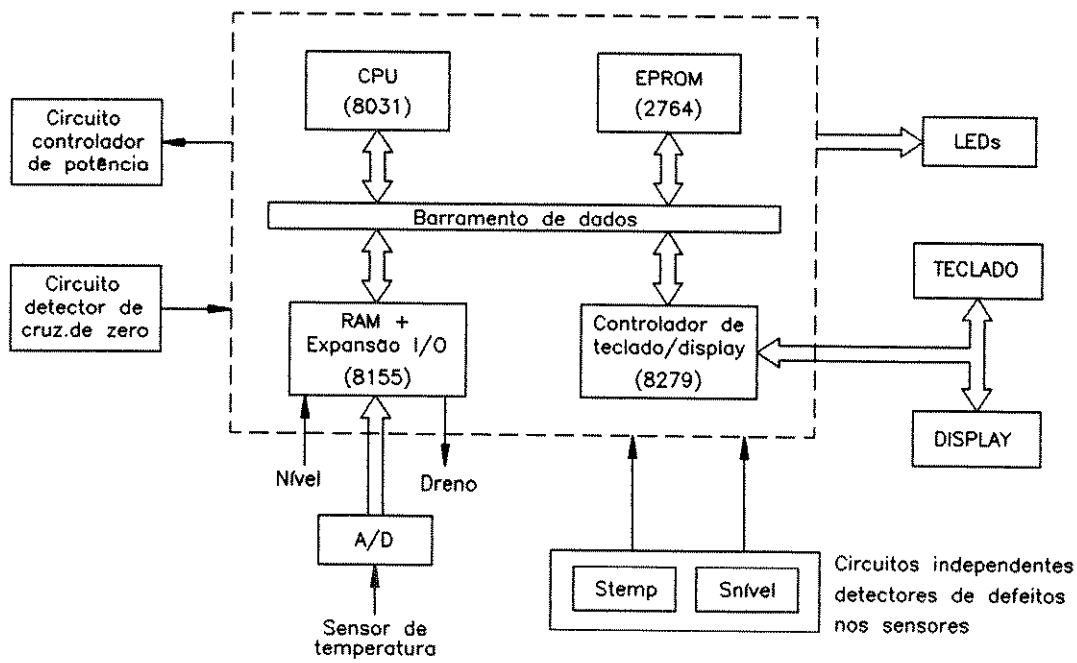


Fig. 2.2 - Diagrama de blocos do controlador microprocessado.

A CPU de controle do sistema é responsável pela centralização e processamento das informações dos blocos periféricos (teclado, *display*, sensores, etc.), bem como pelo tratamento destes dados segundo um programa de controle armazenado na memória EPROM. Uma memória RAM auxiliar amplia a capacidade de memória, uma vez que a memória interna do 8031 pode não ser suficiente à medida que o *software* de controle evolui. Uma expansão de I/O se torna necessária em virtude do número de entradas e saídas já previstas e de possíveis alterações futuras. O controlador de teclado/*display* é utilizado como interface entre a CPU e a unidade de teclado/*display*, auxiliando na tarefa de tratamento de teclados e na varredura de *displays* de sete segmentos.

O sensor de temperatura envia ao microcontrolador, através de um conversor analógico/digital, a informação da temperatura do processo na forma de tensão.

Um TRIAC é usado para controlar a potência a ser entregue ao aquecedor. O TRIAC é gatilhado cíclica e continuamente. O sinal no *gate* do TRIAC provém de um circuito astável que, por sua vez, é controlado pelo microprocessador através de uma rotina PID. Isto permite ao microprocessador manter a temperatura com a precisão requerida. O controlador de potência contém, ainda, um circuito *snubber* para evitar que o TRIAC seja disparado por correntes capacitivas. O circuito de detecção de cruzamento de zero (*zero-cross*) possui duas finalidades: impedir altas correntes ao ligar o TRIAC e assegurar que nenhuma interferência por rádio frequência (RFI) seja gerada quando a carga é chaveada.

Dois circuitos independentes remotos detectam um possível defeito no sensor de nível ou no sensor de temperatura. O microcontrolador monitora um limite máximo de temperatura (125°C), que não pode ser excedido. Caso uma dessas três condições ocorra, o TRIAC é desligado automaticamente (por *hardware*) e uma sinalização adequada é fornecida.

Um teclado e um *display* proporcionam a interação entre o operador e o equipamento, de modo que o operador possa introduzir dados ou parâmetros a serem utilizados pelo programa de controle, bem como fazer ajustes do sistema. O teclado é

matricial, com 15 teclas. O *display* numérico de seis dígitos possibilita a visualização da temperatura do processo. Este *display* também incorpora diagnóstico múltiplo e funções de *setup* que podem ser ativadas, via teclado ou pelo microcontrolador, durante as diversas condições de alarme e *setup*. Sete LEDs discretos indicam o *status* do sistema e anunciam os vários alarmes.

Apenas uma fonte de tensão contínua de +5 volts (1A) é necessária para o funcionamento do sistema. Um LED sinaliza uma queda de tensão da fonte abaixo de 4,3V.

O protótipo consiste de 3 placas de circuitos: uma para o painel, outra para o controle e outra para a parte de potência. Os principais componentes da placa de controle são: o conversor A/D, as memórias EPROM e RAM, a interface teclado/*display* e o microprocessador. A placa de potência contém um transformador (110V-9V) e fornece a fonte DC isolada (+5V). Além disso, contém o circuito de detecção de zero e o circuito controlador de potência. A placa do painel, mostrada na Figura 2.3, contém os *displays* de 7 segmentos, os LEDs e também o teclado.

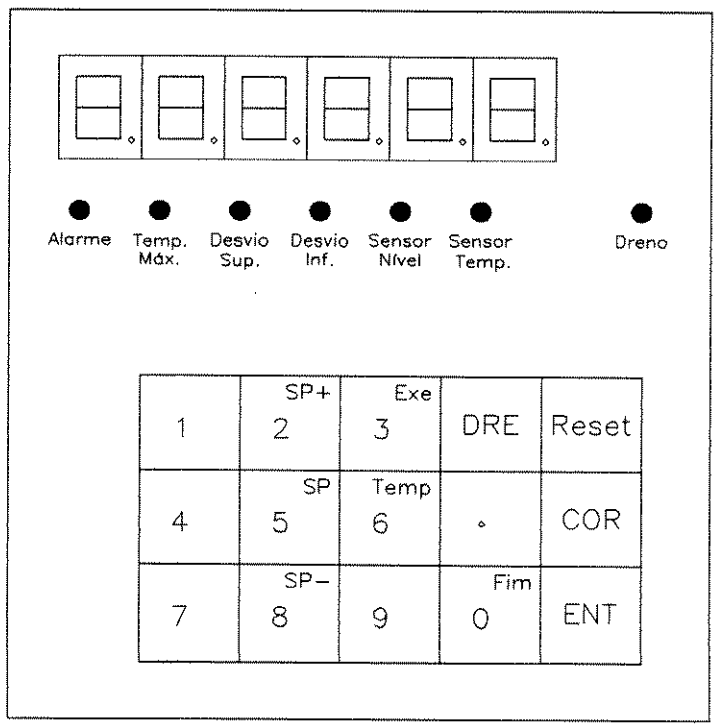


Fig. 2.3 - Painel do protótipo do controlador.

2.2 - PRINCÍPIO DE OPERAÇÃO

A seguir será feita uma explanação sobre o modo de operação do equipamento, com o objetivo de esclarecer os diversos passos a serem seguidos para uma operação correta e segura. Primeiramente serão apresentados os modos de operação e *status* do equipamento, em seguida todas as condições de alarme que poderão ocorrer e, finalmente, uma explicação geral e detalhada da operação do controlador de temperatura microprocessado.

2.2.1 - Modos de *Status* e Operação

Os modos de *status* e operação principais são indicados no painel, por LEDs ou pelo *display* de processo. São eles: NORMAL, ALARME, INICIAL, DRENO e EXECUÇÃO.

NORMAL - Este modo é a condição normal para o sistema. Indica que o sistema está operando dentro dos parâmetros definidos. Neste modo nenhum LED está aceso.

ALARME - As várias condições de alarme são ativadas por muitas fontes e anunciadas pelo *display* e pelos LEDs. O indicador mostra que um alarme ocorreu e que o sistema está operando sob as condições especiais requeridas por aquele alarme. Quando sinalizado pelo *display*, neste lê-se "ERRO" e em seguida o código do alarme: "DR" (dreno) ou "NI" (nível). A única maneira de sair deste modo é corrigir a situação que gerou o alarme e apertar a tecla <RESET>, reinicializando o sistema.

INICIAL - Neste modo os CIs programáveis são inicializados, os LEDs apagados e o aquecedor desligado. Por segurança, a unidade sempre inicializa neste modo. Os *displays* inicialmente acendem todos os seus segmentos devendo ser lido no visor "8.8.8.8.8.8.". O objetivo deste procedimento é alertar para a existência de algum segmento queimado, que poderia alterar o valor dos dados apresentados no *display*. Em seguida, é exibido o código "OP" (operação). O operador deverá então apertar a tecla <EXE> ou a tecla <DRE> para prosseguir.

DRENO - A função DRENO providencia um ciclo de dreno intertravado com a temperatura do banho. No programa de controle, pode-se programar a temperatura, acima da qual a função DRENO não operará.

Quando neste modo, o relé do dreno foi acionado e o líquido está sendo escoado. O LED 'Dreno' está aceso e o *display* mostra a temperatura do líquido quando da ativação deste modo. Vale ressaltar que a temperatura se encontra abaixo de 60°C (variável TEMP_DRE) e o aquecedor desligado. A tecla <DRE> deve ser usada para iniciar o ciclo de dreno e a tecla <RESET> deve ser pressionada ao término do escoamento ou para interromper o ciclo.

EXECUÇÃO - Quando neste modo, deve-se primeiramente fornecer os valores dos parâmetros solicitados através do *display*. A Tabela 2.1 a seguir apresenta os códigos destes parâmetros:

CÓDIGO	DESCRIÇÃO
SP	*Set Point* do processo
GA	Ganho
TI	Constante de Integração
TD	Constante de Derivação

Tabela 2.1 - Códigos dos parâmetros a serem ajustados.

O valor da grandeza referente ao código no *display* deve ser fornecido e, em seguida, a tecla <ENT> deve ser pressionada. Se houver necessidade de uma correção, deve-se apertar a tecla <COR> e fornecer o novo valor.

O *display* de processo mostra a temperatura do banho todo o tempo durante operação normal, porém se a tecla <SP> for pressionada, este mostrará o *Set Point* do processo. A tecla <TEMP> faz com que o *display* retorne à condição normal, ou seja, mostre a temperatura do processo. As teclas <SP+> e <SP-> alteram o valor do *Set Point*, porém funcionam apenas quando o *Set Point* está sendo visualizado.

A tecla <FIM> é usada para interromper a rotina PID. Sempre que o microprocessador estiver executando a rotina de controle, esta tecla possibilitará sua interrupção. O *display* voltará a exibir o código "OP".

Outra maneira de sair deste modo é através da tecla <RESET>. Neste caso, o sistema é reinicializado, voltando-se à condição inicial.

2.2.2 - Condições de Alarme

Como mostrado na Figura 2.3, há seis LEDs de alarme. Eles são essencialmente anunciantes visuais de funções irregulares no sistema. O primeiro, **'Alarme'**, acende sempre que uma condição de alarme ocorrer.

O segundo é o **'Temp. Máx.'**. Este LED acende sempre que o limite máximo de temperatura (125°C) for atingido. O microprocessador monitora continuamente este valor e se detectado, desliga a saída para o aquecedor e ativa o alarme. Por segurança, este desligamento é feito automaticamente por *hardware* também. O valor deste limite máximo é um parâmetro ajustável do programa (variável TEMP_MAX).

O próximo alarme é o **'Desvio Sup.'**. É ativado toda vez que a temperatura do processo atingir 5°C acima do *Set Point*. O valor 5°C é um parâmetro ajustável do programa (variável LIM_SP).

O alarme **'Desvio Inf.'** atua da mesma maneira que o alarme **'Desvio Sup.'**, exceto que compara a temperatura do processo com um valor de 5°C abaixo do *Set Point*. Como a temperatura inicialmente resulta do modo de pré-aquecimento, este alarme seria ativado uma primeira vez antes de se atingir o *Set Point*. O *software* de controle prevê esta situação, impedindo que este LED se acenda nesta condição.

O quinto alarme **'Sensor Nível'** é para um sensor de nível defeituoso. Indica que o sensor está aberto ou não conectado. Circuitos especiais têm sido incorporados para monitorar o sensor para o caso de um circuito aberto. Se esta entrada for ativada, o LED do alarme acende e a saída para o aquecedor é desligada automaticamente (por *hardware*).

O último alarme **'Sensor Temp.'** atua do mesmo modo que o alarme **'Sensor Nível'**, porém monitorando um sensor de temperatura defeituoso.

Todos os alarmes ativam o LED de *status* do alarme assim como o LED **'Alarme'**. Conforme comentado no ítem 2.2.1, alguns alarmes têm *displays* visuais adicionais para melhor definir ou chamar a atenção para eles. Onde aplicável, a saída para o aquecedor é desligada para proteger o equipamento de qualquer possível dano.

2.2.3 - Detalhamento da Operação

Ao ligar o equipamento, todos os segmentos do *display* deverão acender mostrando "8.8.8.8.8.8." no visor. Em seguida, aparecerá o código "OP" e o operador deverá apertar a tecla referente a operação desejada: tecla <EXE> ou tecla <DRE>. Nenhuma outra tecla é atendida nesta etapa. Abaixo é descrito o que cada uma destas teclas executa:

EXECUÇÃO

Se for desejado iniciar o banho de ácido à temperatura constante, deve-se apertar a tecla <EXE>. O sistema verificará primeiramente o nível do líquido.

Se o nível estiver baixo, o *display* mostrará alternadamente a palavra "ERRO" e o código "NI". Neste caso, o nível do líquido deve ser corrigido e o sistema reinicializado, apertando-se a tecla <RESET>.

Se o nível estiver correto, o *display* mostrará os códigos dos parâmetros a serem ajustados. Deve-se fornecer o valor correspondente a cada código no *display* e apertar a tecla <ENT>. A tecla <COR> possibilita a correção do dado. Após pressionada a tecla <ENT>, o *display* exibe o valor recebido para uma eventual verificação.

O sistema inicializa o controle propriamente dito da temperatura do banho e o *display* passa a exibir a temperatura do processo. Nesta etapa apenas algumas teclas estão habilitadas: <SP>, <SP+>, <SP->, <TEMP> e <FIM>.

Caso seja desejado visualizar o valor do *Set Point*, deve-se apertar a tecla <SP>. Para que o *display* volte a exibir a temperatura do banho, basta pressionar a tecla <TEMP>. Se houver a necessidade de alteração do valor do *Set Point* durante o processo, deve-se visualizar o *Set Point* e, em seguida, apertar a tecla <SP+> para incrementar seu valor ou a tecla <SP-> para decrementar.

Se a temperatura do banho atingir o valor do *Set Point* acrescido do desvio admissível, os LEDs 'Desvio Sup.' e 'Alarme' se acendem. Por outro lado, se a temperatura atingir o valor do *Set Point* diminuido do desvio admissível, os LEDs 'Desvio Inf.' e 'Alarme' se acendem.

Se a temperatura do banho atingir a temperatura máxima permitida, o processo é interrompido. Os LEDs 'Temp. Máx.' e 'Alarme' se acendem, o aquecedor é desligado e o *display* volta a exibir o código "OP".

Se os LEDs 'Sensor Nível' ou 'Sensor Temp.' se acenderem durante o banho de ácido, isto indica que foi detectado um defeito em um dos sensores. Neste caso, o aquecedor é desligado automaticamente. Se o nível do líquido começar a diminuir, o aquecedor também é desligado automaticamente.

Quando for desejado interromper o processo, a tecla <FIM> deve ser pressionada. O aquecedor é desligado e o *display* volta a exibir o código "OP".

Caso a tecla <RESET> venha a ser pressionada durante o processo, aparentemente tem-se o mesmo efeito da tecla <FIM>, porém o *display* volta a exibir "8.8.8.8.8.8.", ou seja, volta-se às condições iniciais.

DRENO

Se for desejado esvaziar o tanque, deve-se apertar a tecla <DRE>. O sistema verificará a temperatura do líquido, a qual será mostrada no *display*. Neste ponto podem ocorrer uma das situações abaixo:

Se a temperatura estiver acima da temperatura permitida para escoamento, o *display* mostrará a palavra "ERRO" e, em seguida, o código "DR". Instantes depois, o *display* voltará a

exibir o código "OP". O operador deverá então esperar pelo resfriamento do líquido até a temperatura especificada para escoamento e apertar a tecla <DRE> novamente.

Se a temperatura estiver dentro do limite permitido para escoamento do líquido, o sistema envia um comando para o circuito da válvula do dreno habilitando sua abertura e acende o LED 'Dreno'. A temperatura do líquido continua sendo exibida no *display*. Quando o líquido estiver escoado completamente, a tecla <RESET> deve ser pressionada, reinicializando o sistema.

Para finalizar, a Tabela 2.2 abaixo apresenta todos os códigos, e respectivos significados, que aparecem no *display*:

CÓDIGO	DESCRIÇÃO
OP	Operação desejada
ERRO	Sinalização de erro
NI	Nível incorreto
DR	Dreno: abertura inválida
SP	*Set Point* do processo
GA	Ganho
TI	Constante de Integração
TD	Constante de Derivação

Tabela 2.2 - Códigos exibidos no *display*.

CAPÍTULO 3 - *HARDWARE*

3.1 - ARQUITETURA DA FAMÍLIA DE MICROCONTROLADORES 8051

3.1.1 - Descrição Funcional

Um microcontrolador possui as mesmas funções de um microprocessador com a vantagem de trazer incorporados temporizadores, portas de I/O, memórias, etc., permitindo assim a construção de sistemas compactos e tão poderosos quanto os baseados em microprocessadores. Desta forma, traz vantagens tanto no tamanho reduzido quanto na facilidade de *software*, dado que seus periféricos são vistos pela CPU interna como memória.

O CI 8051 é um microcontrolador de 8 *bits* fabricado pela Intel [2][3]. Devido as suas características de *hardware* e *software* é usado como um poderoso controlador, permitindo controle em tempo real, execução de complexas operações lógicas e aritméticas (multiplicação, divisão, etc.), manipulação de dados de 16 *bits* e expansão de memória, além de trabalhar com bancos de registradores nominais e com *bits* individualmente endereçáveis na RAM, como será visto mais adiante.

As principais características da arquitetura do 8051 são apresentadas a seguir, e serão detalhadas nos próximos ítens para uma melhor compreensão do funcionamento interno do 8051:

- oscilador interno
- 4 K*bytes* de ROM (interna)
- 128 *bytes* de RAM (interna)
- 21 Registros de Funções Especiais
- 32 linhas de I/O
- capacidade de endereçar até 64 K*bytes* de Memória de Dados externa
- capacidade de endereçar até 64 K*bytes* de Memória de Programa externa
- 2 Temporizadores/Contadores programáveis de 16 *bits*
- porta serial *full-duplex*
- estrutura de interrupção com 5 fontes e 2 níveis de prioridade

O termo "8051" é usado genericamente para referir aos microcontroladores 8051, 8031 e 8751. O 8031 é um 8051 sem a ROM interna; todas as instruções vêm da memória externa. O 8751 é um 8051 com EPROM ao invés de ROM. Por razões de disponibilidade e custo, o 8031 foi o microcontrolador escolhido para o projeto em questão, sendo assim as particularidades referentes ao 8751 e ao 8051 não serão discutidas neste trabalho. A Figura 3.1 mostra o diagrama de blocos (a) e a pinagem (b) do microcontrolador 8051:

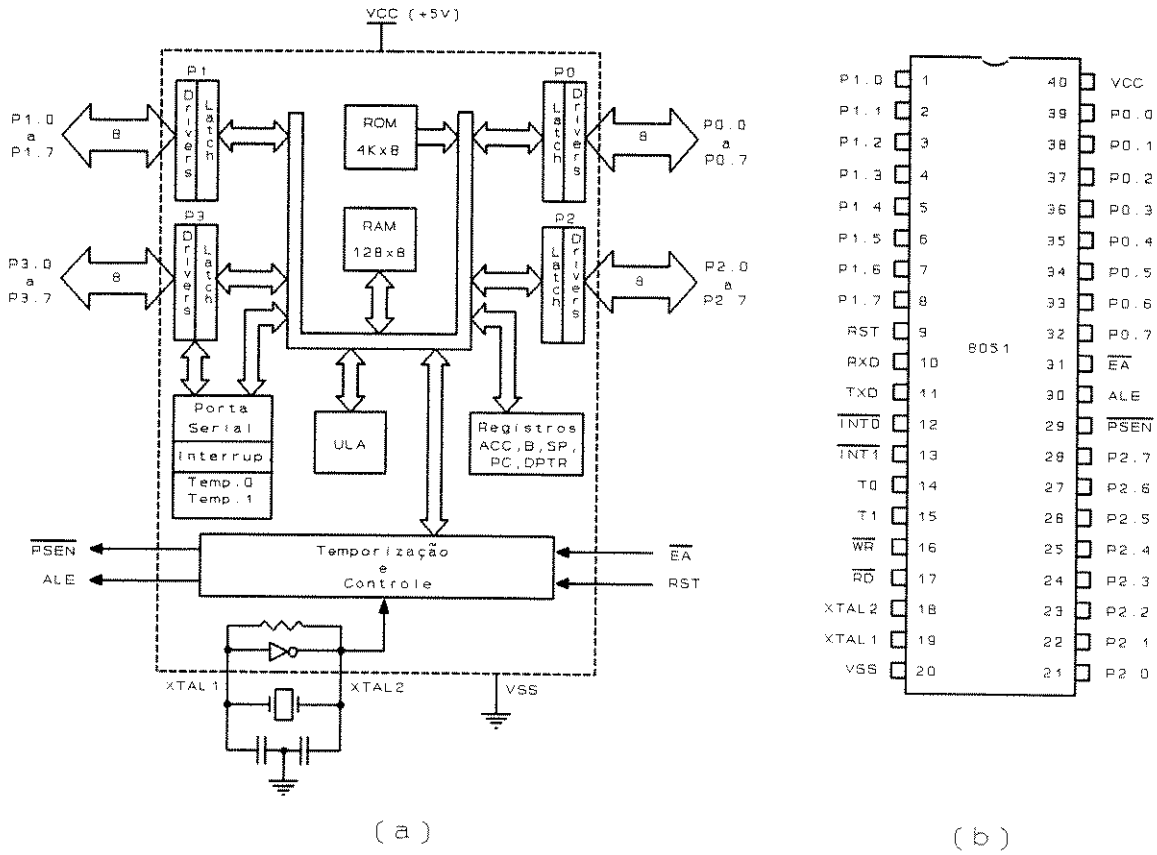


Fig. 3.1 - *Hardware* do CI 8051
(a) - Diagrama de blocos
(b) - Configuração dos pinos

3.1.2 - Descrição dos Pinos da Família 8051

VSS Potencial terra do circuito.

VCC Tensão de alimentação (+5 volts).

P0 a P3 Quatro portas de I/O de 8 *bits* bidirecionais. Os pinos da Porta 3 são multifuncionais, servindo também as funções especiais descritas a seguir:

Pino		Função Alternativa
P3.0	RXD	: entrada da porta serial
P3.1	TXD	: saída da porta serial
P3.2	$\overline{\text{INT0}}$	: interrupção externa 0
P3.3	$\overline{\text{INT1}}$	: interrupção externa 1
P3.4	T0	: entrada externa do Temp. 0
P3.5	T1	: entrada externa do Temp. 1
P3.6	$\overline{\text{WR}}$	: *strobe* de escrita na Memória de Dados externa
P3.7	$\overline{\text{RD}}$	: *strobe* de leitura da Memória de Dados externa

RST Um nível alto neste pino por 2 ou mais ciclos de máquina efetua o *reset* do dispositivo.

ALE Saída habilitadora do *latch* de endereço. Usado para "travar" o *byte* menos significativo do endereço durante o acesso a memória externa.

$\overline{\text{PSEN}}$ Saída habilitadora da leitura da Memória de Programa externa. Não é ativado (permanece em nível alto) durante as buscas na Memória de Programa interna.

$\overline{\text{EA}}$ Entrada de seleção de memórias.

XTAL1 Entrada do amplificador do oscilador interno.

XTAL2 Saída do amplificador do oscilador interno. Se for usado um oscilador externo, recebe o sinal do mesmo.

3.1.3 - Circuito do Oscilador e *Clock*

O 8051 possui um circuito oscilador interno com amplificador inversor. O sinal de sincronismo (*clock*) da CPU pode ser gerado por este oscilador ou por uma fonte externa.

Para que o *clock* seja gerado pelo oscilador interno basta conectar um cristal, entre os pinos XTAL1 e XTAL2, e dois capacitores de realimentação cujos valores recomendados são da ordem de 30pF [4]. A faixa de frequência para este oscilador a cristal é de 1,2MHz a 12MHz [2].

Se for utilizado um oscilador externo, deve-se aterrar o pino XTAL1 e injetar o sinal externo no pino XTAL2, que desta forma irá diretamente para o circuito interno de temporização e

controle. A Figura 3.2 a seguir mostra as duas configurações para o oscilador no 8051:

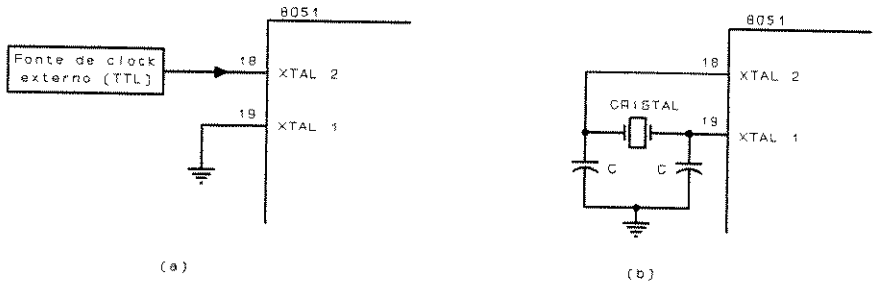


Fig. 3.2 - Geração do *clock* no 8051 com oscilador externo (a) e com oscilador interno (b).

Um ciclo de máquina no 8051 consiste de 12 períodos do oscilador [2]. A maior parte das instruções são executadas em um ciclo de máquina. As instruções de multiplicação e divisão são as únicas que levam mais de dois ciclos para serem completadas; ocupam quatro ciclos de máquina.

3.1.4 - Organização da Memória

O 8051 trabalha separadamente com Memória de Programa e com Memória de Dados. O mapa da memória do 8051 está ilustrado na Figura 3.3:

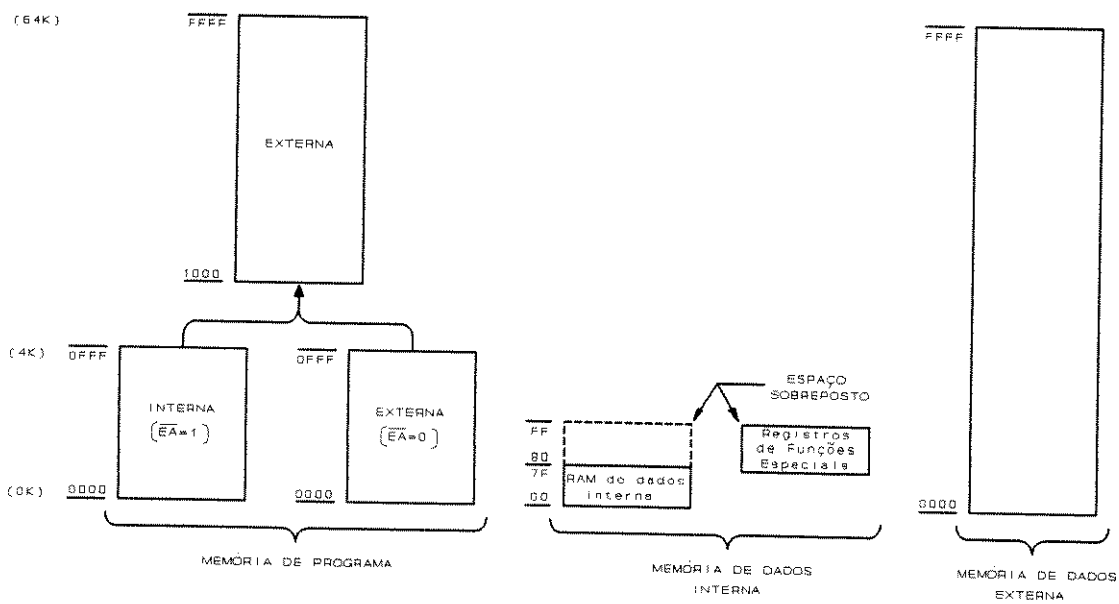


Fig. 3.3 - Mapa da memória do 8051.

A Memória de Programa pode ser de até 64 *Kbytes*, sendo que os primeiros 4 *Kbytes* estão na ROM interna (invioláveis). Se o pino $\overline{EA}$ for mantido em nível alto, a CPU endereça a ROM interna a menos que o PC (*Program Counter*) exceda 0FFFH. Os 60 *Kbytes* restantes (localizações 1000H a FFFFH) são buscados externamente. Se $\overline{EA}$ for mantido em nível baixo, a CPU trabalha somente com a Memória de Programa externa (64 *Kbytes*). No 8031, $\overline{EA}$ deve ser ligado a nível lógico 0 externamente.

A Memória de Dados consiste de 128 *bytes* da RAM interna mais 21 Registros de Funções Especiais, sendo ainda possível uma expansão para mais 64 *Kbytes* de memória externa. A Memória de Dados interna usa endereçamento de 8 *bits*. A RAM interna possui 128 *bits* individualmente endereçáveis, que podem ser usados como *flags* no *software* ou para processamento (*Booleano*) de 1 *bit*. Esta capacidade de endereçar *bit* é uma característica importante do 8051, pois facilita a programação. Onze dos Registros de Funções Especiais também têm *bits* individualmente endereçáveis.

3.1.5 - Registros de Funções Especiais

Os Registros de Funções Especiais estão listados abaixo. O registro ACC é equivalente ao acumulador dos microprocessadores de 8 *bits*, o mesmo acontecendo para os registros B, PSW e SP.

- ACC* — Acumulador
- B* — Registro B
- PSW* — Palavra de *Status* do Programa
- SP — *Stack Pointer*
- DPTR — *Data Pointer* (consiste de DPH e DPL)
- P0,P1,P2,P3 * — Porta 0, Porta 1, Porta 2, Porta 3
- IP* — Prioridade da Interrupção
- IE* — Habilitador da Interrupção
- TMOD — Modo de Operação do Temporizador/Contador
- TCON* — Controle do Temporizador/Contador
- TH0,TL0,TH1,TL1 — Temporizador 0 e Temporizador 1
- SCON* — Controle da Porta Serial
- SBUF — *Buffer* de Dados Serial
- PCON — Controle da Potência

Os registros marcados com "*" são tanto *byte* quanto *bit* endereçáveis. A CPU trata os Registros de Funções Especiais da mesma forma que a RAM. O mais interessante é que registros da CPU, tais como ACC e B, são também acessíveis como simples posições de memória. A seguir, será dada uma breve descrição de alguns registros:

- *Data Pointer* (DPTR)

Registro de 16 *bits* que consiste de um *byte* mais significativo (**DPH**) e um menos significativo (**DPL**). Sua função é armazenar um endereço de 16 *bits*.

- Portas 0 a 3 (P0, P1, P2, P3)

Estes registros são posições da RAM, que contêm os dados das portas de I/O.

- Temporizadores/Contadores (TH1, TL1, TH0, TL0)

Estes registros contêm os valores a serem contados pelos Temporizadores/Contadores.

- *Buffer* de Dados Serial (SBUF)

Registro no qual a porta serial armazenará o dado recebido ou escreverá o dado a ser transmitido.

- Registros de *Status* e Controle (IP, IE, TMOD, TCON, SCON, PCON)

Estes registros contêm *bits* de *status* e controle para o sistema de interrupção, para os temporizadores e para a porta serial. Serão descritos com maiores detalhes oportunamente.

3.1.6 - Operação das Portas de I/O

As quatro portas paralelas do 8051 fornecem as 32 linhas de I/O, sendo que cada linha pode ser usada independentemente como entrada ou como saída. Uma linha configurada como entrada pode ser reconfigurada como saída e vice-versa. As portas podem ser endereçadas individualmente como portas de 8 *bits* cada ou *bit* a *bit*. As Portas 0 e 2 podem também ser utilizadas para a expansão de memória, porém, neste caso não poderão mais ser utilizadas como portas de I/O.

A operação das portas como I/O é muito simples. Se a porta estiver atuando como saída, uma escrita em seu registro alterará automaticamente o conteúdo presente nos pinos. Se estiver atuando como entrada, uma operação de leitura colocará o estado presente nos pinos dentro de seu registro. Uma variação do valor presente nos pinos só será sentida pelo sistema quando for efetuada uma nova leitura. Isto acontece porque, na realidade, cada porta consiste de um *latch* (registros P0 a P3), *drivers* de saída e *buffers tri-state* de entrada (Figura 3.1).

As Portas 1, 2 e 3 possuem *pull-ups* internos, os quais são FETs e não resistores lineares. São chamadas de portas quase-bidirecionais, pois, quando configuradas como entrada cada pino é levado a nível alto pelo *pull-up* interno, mas poderá ser levado a nível baixo por uma fonte externa. Ou seja, o pino tem sempre um estado definido (0 ou 1), de forma que é possível medir seu nível apesar da porta estar sendo utilizada como entrada.

A Porta 0 difere por não ter *pull-ups* internos e quando configurada como entrada terá seu nível flutuando na ausência de um nível fixo no pino. Como saída, a Porta 0 tem saídas com *dreno* aberto, exceto quando usada como barramento de endereço/dados.

As Portas 1, 2 e 3 podem alimentar 3 entradas LS TTL e a Porta 0 pode alimentar 8 entradas LS TTL. As Portas 1, 2 e 3 podem acionar entradas MOS sem *pull-ups* externos. A Porta 0 necessita de *pull-ups* externos para acionar entradas MOS, exceto quando estiver sendo usada como barramento de endereço/dados.

Conforme visto no ítem 3.1.2, os pinos da Porta 3 são capazes de desempenhar funções alternativas, não relacionadas às funções da porta, tais como: expansão da Memória de Dados, canal serial, interrupções externas e sinal externo para os temporizadores. Se qualquer uma destas funções for utilizada, a Porta 3 não poderá mais ser acessada como porta de I/O para *byte*, apenas como *bit* endereçável.

3.1.7 - Acesso a Memória Externa

O acesso a Memória de Programa externa utiliza o sinal $\overline{\text{PSEN}}$ como *strobe* de leitura e o acesso a Memória de Dados externa utiliza os sinais $\overline{\text{RD}}$ ou $\overline{\text{WR}}$ (funções alternativas de P3.7 e P3.6).

A Memória de Programa usa endereçamento de 16 *bits* e a Memória de Dados externa pode usar tanto endereçamento de 16 *bits* quanto de 8 *bits*.

Sempre que um endereço de 16 *bits* é usado, o *byte* mais significativo do endereço é colocado na Porta 2, onde é mantido enquanto durar o ciclo de leitura ou escrita. Se for usado um endereço de 8 *bits*, o conteúdo da Porta 2 permanece inalterado por todo o ciclo da memória externa, facilitando a paginação.

O *byte* menos significativo do endereço é multiplexado no tempo com o *byte* de dados na Porta 0. Nesta situação os pinos da Porta 0 não são saídas com dreno aberto e não necessitam de *pull-ups* externos. O sinal ALE é responsável por colocar o *byte* menos significativo do endereço no *latch* externo, separando-o do *byte* de dados.

3.1.8 - Temporizadores

O 8051 possui 2 registros de 16 *bits*, Temporizador 0 e Temporizador 1, que podem ser usados como temporizadores ou contadores de eventos. São programados pelo *software* e operam de maneira completamente independente aos demais sistemas do microcontrolador. Na função "temporizador", o registro é incrementado a cada ciclo de máquina. Pode-se dizer que é um contador de ciclos de máquina. Na função "contador", o registro é incrementado a cada transição negativa (1 para 0) do sinal externo, aplicado no pino T0 ou T1 (funções alternativas da Porta 3).

Cada Temporizador/Contador (T/C) possui 4 modos de operação possíveis. Os Modos 0, 1 e 2 são os mesmos para ambos os Temporizadores, porém o Modo 3 é diferente.

- MODO 0

O temporizador configurado no Modo 0 é um contador ou temporizador de 8 *bits*, com divisor de frequência de até 32 vezes. Na Figura 3.4 é mostrado o Temp. 1 operando no Modo 0. O valor inicial da contagem é escrito no registro TH1 por *software*. O sinal de contagem é dividido pelo valor presente nos *bits* 0 a 4 do registro TL1. Como exemplo, se o valor do divisor for 15, tem-se que a cada 15 contagens do sinal de entrada, o registro TH1 sofre UM incremento.

Ao ocorrer o *overflow* no registro TH1, o Temp. 1 gerará um pedido de interrupção, que será ou não aceito pela CPU.

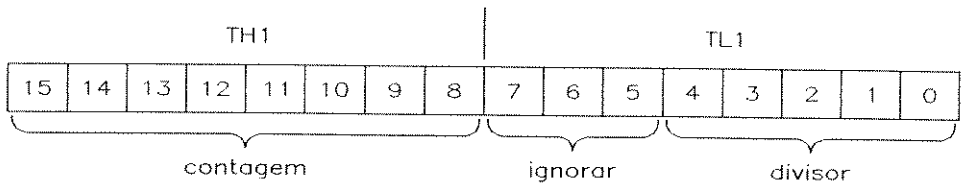


Fig. 3.4 - Temporizador 1 no Modo 0.

- MODO 1

No Modo 1 o temporizador opera como um contador ou temporizador de 16 *bits*, utilizando o par de registros THx e TLx para efetuar a contagem. Neste modo pode-se contar até 65535 vezes, com valor inicial programável por *software*.

- MODO 2

Neste modo o temporizador é configurado como um contador ou temporizador de 8 *bits* (TL1) com recarga automática. Um *overflow* de TL1 não somente gera um pedido de interrupção, como também recarrega TL1 com o conteúdo de TH1, que é pré-definido por *software* para qualquer valor de 1 *byte* desejado.

- MODO 3

O Temporizador 1, configurado no Modo 3, pára sua contagem e o Temporizador 0 neste modo comporta-se como dois contadores de 8 *bits* separados (TH0 e TL0).

Dois Registros de Funções Especiais, TCON e TMOD, são usados para controlar as funções e definir os modos de operação dos Temporizadores/Contadores.

- TCON: Registro de Controle do Temporizador

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

- TFx — este *bit* é levado a nível lógico 1 pelo *hardware* sempre que ocorrer um *overflow* no Temp. "x", gerando um pedido de interrupção. Zerado pelo *hardware* quando o processador transfere o controle para a rotina de interrupção.

- TRx — liga/desliga o Temp. "x". Quando TRx=1, o Temp. "x" inicia a contagem, caso contrário pára a contagem.

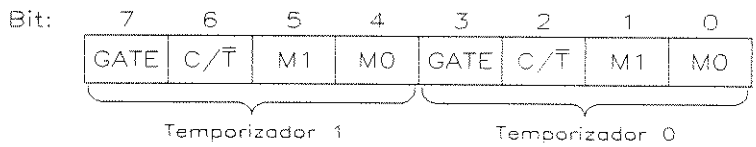
- ITx — indica o modo de ativação da interrupção externa "x".

- IEx — sinaliza um pedido da interrupção externa "x".

Nas descrições acima "x" pode receber os valores "0" ou "1". Para obter a descrição dos *bits* TR0 e TR1, por exemplo, basta substituir "x" por "0" em toda a descrição de TRx e, em uma segunda etapa, por "1".

Os *bits* IT0, IE0, IT1 e IE1 não se aplicam aos temporizadores, são usados para o controle das interrupções externas (ítem 3.1.10).

- TMOD: Registro de Controle do Modo do Temporizador



- M1 e M0 especificam o Modo:

M1	M0	Modo	Descrição
0	0	0	contador de 8 *bits*
0	1	1	contador de 16 *bits*
1	0	2	contador de 8 *bits* com auto-recarga
1	1	3	2 contadores de 8 *bits* (Temp. 0)

- $C/\overline{T}$ — seleciona a função do Temporizador/Contador. Se $C/\overline{T}=1$, a função será "contador" (sinal externo). Se $C/\overline{T}=0$, a função será "temporizador" (sinal interno).

- GATE determina como o T/C será habilitado. Se GATE=0, o T/C "x" será habilitado (estará contando) quando o *bit* TRx for igual a 1. Se GATE=1, o T/C "x" somente será habilitado se o pino $\overline{\text{INTX}}$ e o *bit* TRx forem 1.

Assim, se GATE=1, pode-se utilizar o T/C para contar a largura de pulsos externos. Basta colocar a entrada externa no pino de interrupção correspondente e fazer com que o T/C funcione com sinal interno. Desta maneira, somente haverá contagem quando o sinal externo for 1. O *software*, neste caso, encarregar-se-á de calcular o tempo decorrido entre dois pulsos externos.

3.1.9 - Interface Serial

A interface serial é uma das poucas características do 8051 que não será utilizada neste trabalho, por isso será tratada superficialmente. Para maiores detalhes consultar o manual do fabricante (referências [2][3]).

A porta serial do 8051 é *full-duplex*, o que significa que pode transmitir e receber dados simultaneamente. Possibilita comunicação com periféricos e também comunicação assíncrona entre processadores. O registro SCON proporciona o controle da porta de comunicação serial conforme desejado.

Esta porta utiliza o pino RXD (P3.0) como pino de recepção e o pino TXD (P3.1) como pino de transmissão. O pino RXD é amostrado periodicamente e, quando a porta serial detecta um *start bit* válido, o dado correspondente é carregado no registro SBUF. Um pedido de interrupção é gerado para informar à CPU que um dado foi recebido. A CPU acessa este dado executando uma operação de leitura em SBUF. Na transmissão serial, um dado é escrito no registro SBUF pela CPU e enviado através do pino TXD. Um pedido de interrupção informa à CPU o término da transmissão.

3.1.10 - Interrupções

Neste projeto em particular nenhuma interrupção será utilizada, logo todas as interrupções deverão ser desabilitadas. O 8051 possui cinco fontes de interrupção:

- Interrupção Externa 0 (pino $\overline{\text{INT0}}$ da Porta 3)
- Interrupção Externa 1 (pino $\overline{\text{INT1}}$ da Porta 3)
- Temporizador 0 (*overflow*)
- Temporizador 1 (*overflow*)
- Porta Serial

Cada interrupção pode ser individualmente habilitada ou desabilitada. Pode-se também desabilitar todas de uma só vez. As interrupções podem ter apenas dois níveis de prioridade. Uma interrupção de maior prioridade interrompe outra de menor prioridade, mesmo que esta tenha chegado primeiro. O processador responde à requisição ativa de maior prioridade, desviando para um certo endereço fixo, no qual começa a rotina de serviço da interrupção em questão. A execução prossegue a partir deste endereço, até que a instrução RETI seja encontrada.

Os Registros de Funções Especiais IP e IE possibilitam a escolha de qual ou quais interrupções serão habilitadas (ou desabilitadas) e qual a prioridade de cada uma.

- IE: Registro Habilitador de Interrupção

Bit:	7	6	5	4	3	2	1	0
	EA	X	X	ES	ET1	EX1	ET0	EX0

onde X = irrelevante (pode ser 0 ou 1)

- EA — EA=0 desabilita todas as interrupções. Se EA=1, cada fonte de interrupção é habilitada ou desabilitada em função dos *bits* de controle individuais a seguir.

- ES — ES=1 habilita a interrupção da Porta Serial. Se ES=0, a interrupção da Porta Serial é desabilitada.

- ETx — ETx=1 habilita a interrupção do Temporizador "x". Se ETx=0, a interrupção do Temp. "x" é desabilitada.

- EX1 — EXx=1 habilita a interrupção externa "x". Se EXx=0, a interrupção externa "x" é desabilitada.

- IP: Registro de Prioridade da Interrupção

Bit:	7	6	5	4	3	2	1	0
	X	X	X	PS	PT1	PX1	PT0	PX0

Os *bits* PS, PT1, PX1, PT0 e PX0 quando em nível lógico 1, indicam prioridade alta para a interrupção da Porta Serial, do Temporizador 1, Externa 1, Temporizador 0 e Externa 0, respectivamente.

3.1.11 - *Reset*

Quando uma condição de *reset* ocorre, o processamento é interrompido e o sistema reinicializado. O *reset* no 8051 é ativado sempre que o pino RST permanecer em nível alto, por pelo menos 2 ciclos de máquina. Neste caso, a CPU responde executando um *reset* interno, que consiste em preencher os registros internos com valores pré-determinados, a saber:

Registro	Conteúdo
PC e DPTR	0000H
A, B, PSW, TMOD, TCON e SCON	00H
TH1, TL1, TH0 e TL0	00H
SP	07H
P0, P1, P2 e P3	FFH
IP	(XXX00000)
IE	(0XX00000)
PCON	(0XXXXXXX)
SBUF	indeterminado

Um *reset* automático pode ser obtido quando a alimentação é ligada, conectando-se um capacitor de 10μF entre os pinos VCC e RST [2]. Na Figura 3.5 pode-se observar que o 8051 possui um resistor de 8KΩ ligado internamente entre seu pino de *reset* e o terra [4]. Quando a potência é ligada, o capacitor comporta-se inicialmente como um curto-circuito, provocando uma circulação de corrente pelo resistor interno e uma tensão inicial no pino

RST da ordem de VCC (+5V). À medida que a carga no capacitor aumenta, a tensão no pino RST decresce. Após algum tempo, o capacitor estará carregado e não circulará mais corrente pelo resistor, o que tornará a tensão no pino RST nula. Esta configuração providenciará um *reset* seguro ao ligar.

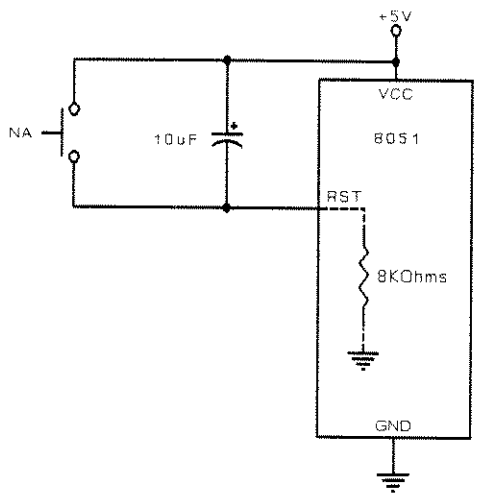


Fig. 3.5 - *Reset* ao ligar e *reset* forçado no 8051.

O circuito da Figura 3.5 implementa também um *reset* forçado, que é obtido com a chave de contato momentâneo normalmente aberta (NA), em paralelo com o capacitor. Ao pressionar e soltar a chave, provoca-se uma nova operação de *reset*.

3.1.12 - Configuração do 8031

Neste projeto o microcontrolador 8031 é implementado com oscilador interno, uma vez que não existe no sistema nenhuma fonte de *clock* externo que possa ser aproveitada, ou seja, que atenda a faixa de frequência de 1,2MHz a 12MHz. O cristal utilizado é de 5MHz e os capacitores de 33pF cada.

Como o 8031 não possui ROM interna e a RAM interna pode ser insuficiente, torna-se necessário utilizar a capacidade de expansão de memória. Assim, as Portas 0 e 2 são usadas como barramento (*bus*) de endereço/dados. A Memória de Programa externa é uma EPROM de 8 K*bytes* (CI 2764) [5].

Ambos os temporizadores operam com 16 *bits*, sendo que o Temp. 0 possui a função de temporização e o Temp. 1 funciona

com sinal externo. São utilizados na malha de controle PID para os propósitos de temporização. O Temporizador 0 constitui um relógio de tempo real, utilizado para medir o período de amostragem Δt (Capítulo 4, ítem 4.1.4). O Temporizador 1 trabalha em conjunto com outro temporizador e sua função será discutida no ítem 3.2.6. Assim sendo, os registros TMOD e TCON devem ser carregados com os seguintes valores:

TMOD:	0	1	0	1	0	0	0	1	= 51H
TCON:	0	1	0	1	0	0	0	0	= 50H

Dos recursos disponíveis no 8051, somente a interface serial e as interrupções não são utilizadas neste projeto. No caso das interrupções torna-se necessário desabilitá-las, assim, o registro IE deve possuir o seguinte valor:

IE:	0	0	0	0	0	0	0	0	= 00H

As Portas 1 e 3 são configuradas como portas de I/O *bit* endereçáveis. Controlam os LEDs de alarme e a saída para o aquecedor. Juntamente com as portas lógicas, implementam o desligamento automático do aquecedor no caso de temperatura excessiva, nível do líquido incorreto ou defeito em um dos sensores. Os pinos P1.0 e RXD são levados a nível lógico 1 pelo *software*, se a temperatura do banho atingir seu limite máximo, e o pino TXD é responsável pelo aquecimento do líquido. Estas portas, também, controlam a RAM externa e a interface teclado/*display*, assunto dos próximos ítens.

As portas lógicas e o LATCH são CIs TTL [8]. A porta lógica AND foi implementada com portas NOR e INV, otimizando o circuito. A Figura 3.6 mostra a configuração do *hardware* do 8031 para este projeto:

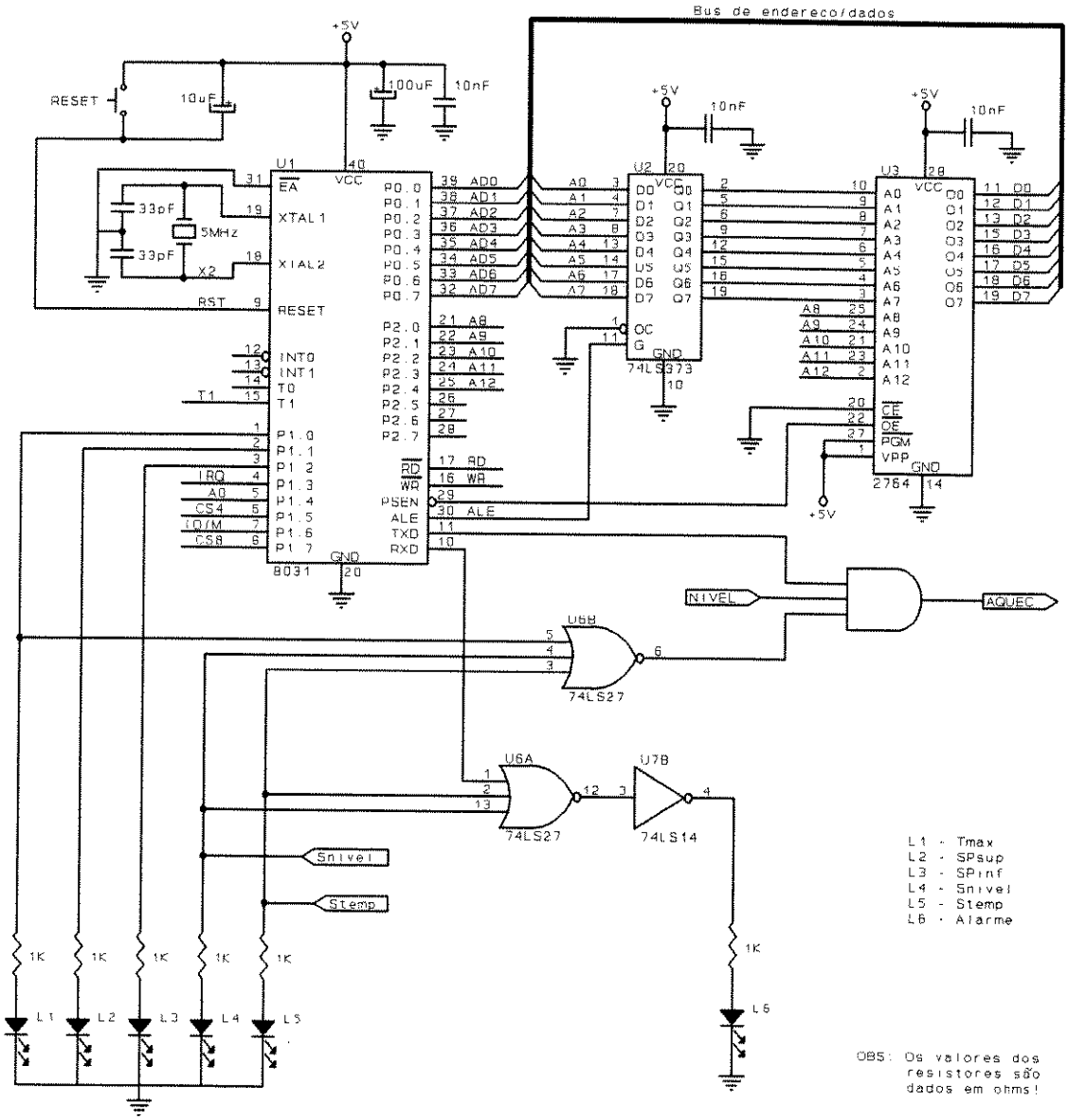


Fig. 3.6 - Esquema de ligação do 8031 e da memória EPROM.

3.2 - MEMÓRIA DE DADOS EXTERNA E EXPANSÃO DE I/O

3.2.1 - Descrição Funcional

Uma vez que a RAM interna do 8031 pode não ser suficiente à medida que o *software* evolui, torna-se necessário expandir a memória de dados. Uma expansão da capacidade de I/O também deve ser prevista, em virtude do número de entradas e saídas utiliza-

das no projeto. Estas considerações levaram à escolha do circuito integrado 8155 da Intel que contém, em sua arquitetura interna, além de uma memória de dados, portas de I/O e temporizador [6]. O CI 8155 é um dispositivo programável, designado para o uso com microprocessadores de 8 *bits* da linha Intel, e sua utilização reduz significativamente o *hardware* do controlador.

A memória de dados é uma RAM estática de 256 *bytes*. A seção de I/O consiste de duas portas de 8 *bits* (Portas A e B) e uma porta de 6 *bits* (Porta C), que podem ser configuradas como entrada ou saída e, ainda, ser utilizadas em modo de confirmação (*handshake mode*). O 8155 conta, também, com um temporizador de 14 *bits* programável, que produz um sinal ao término da contagem.

As portas de I/O possuem registros internos (PA, PB e PC), que guardam os dados durante as operações de I/O. O 8155 contém ainda um registro de comando e um registro de *status*, responsáveis pela programação e controle das portas de I/O e, também, do temporizador. Para propósitos de endereçamento, a seção de I/O incorpora estes cinco registros (comando, *status*, PA, PB, PC), mais o temporizador. A Figura 3.7 a seguir mostra o diagrama de blocos (a), bem como a pinagem (b) do 8155:

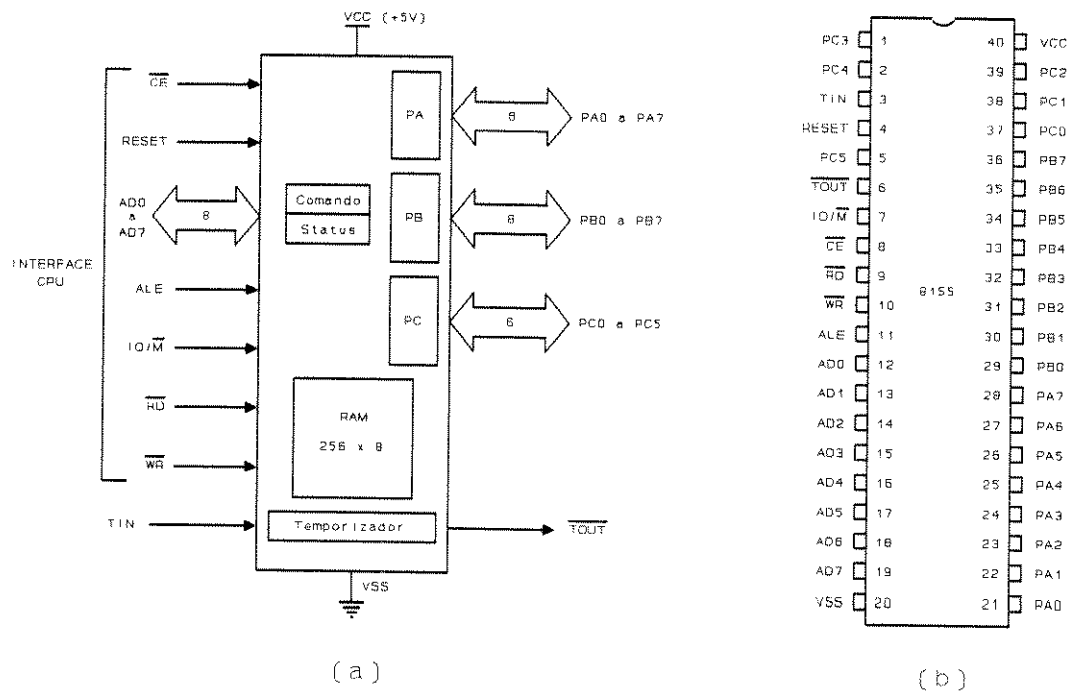


Fig. 3.7 - *Hardware* do CI 8155
(a) - Diagrama lógico
(b) - Configuração dos pinos

3.2.2 - Descrição dos Pinos do 8155

A maioria dos pinos do 8155 são auto-explicativos. Os pinos VSS, VCC e ALE possuem o mesmo significado dos já descritos para o 8031. A seguir, será dada uma descrição dos demais pinos:

AD0-AD7 Estas linhas constituem o barramento de endereço/dados. O *byte* de endereço pode endereçar tanto a memória quanto a seção de I/O, dependendo da entrada IO/$\overline{M}$.

RESET Um nível alto neste pino efetua o *reset* do 8155 e inicializa as três portas de I/O como entrada.

$\overline{CE}$ Um nível baixo neste pino habilita o 8155.

PA a PC Portas de I/O de propósitos gerais.

IO/$\overline{M}$ Seleciona a memória se em nível baixo e a seção de I/O se em nível alto.

TIN Entrada do temporizador/contador.

$\overline{TOUT}$ Saída do temporizador/contador.

3.2.3 - Princípio de Operação

O barramento de dados e endereço do 8155 é multiplexado e deve ser conectado diretamente ao barramento de endereço/dados do microprocessador. O 8155 possui um *latch* de endereço interno, para simplificar esta conexão. A Figura 3.8 mostra a conexão do CI 8155 com um microprocessador de 8 *bits* da linha Intel:

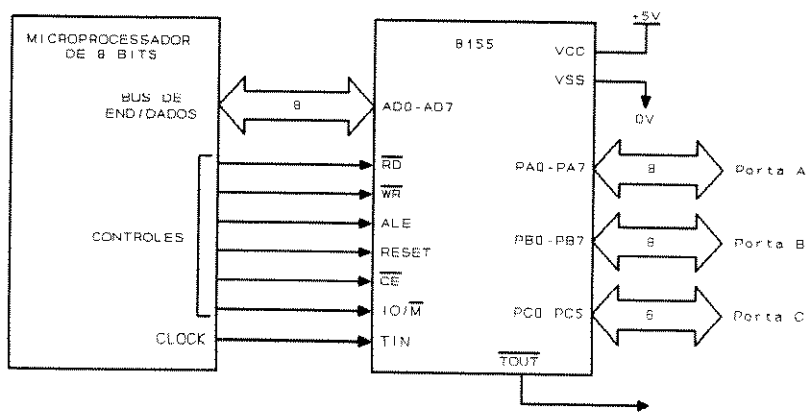


Fig. 3.8 - Diagrama de blocos da conexão do 8155 com um microprocessador de 8 *bits*.

Para realizar uma operação com a memória do 8155, deve-se manter a entrada IO/$\overline{M}$ em nível baixo e para realizar uma operação de I/O, deve-se manter esta entrada em nível alto. Assim, se IO/$\overline{M}$=0, durante uma operação de leitura, o conteúdo da RAM será lido; caso contrário, será lido o conteúdo de um dos registros de I/O. O mesmo se aplica a uma operação de escrita.

É importante notar que basta um comando da CPU para alterar o modo de operação deste componente. Assim, num instante pode-se estar fazendo uma operação de memória e no instante seguinte uma operação de I/O. Tal como no 8031, uma porta de I/O pode ser configurada como porta de entrada e, logo em seguida, reconfigurada como porta de saída e vice-versa.

Durante as operações de memória, envolvendo o 8155, todos os 8 *bits* de endereço/dados são utilizados, porém, durante as operações de I/O somente os 3 *bits* menos significativos do endereço (A2, A1 e A0) são utilizados. A Tabela 3.1 a seguir mostra o endereço para cada registro interno. O endereço de I/O (A7 a A0) deve ser enviado com $\overline{CE}$=0 e IO/$\overline{M}$=1, de modo a selecionar o registro apropriado.

A7	A6	A5	A4	A3	A2	A1	A0	SELEÇÃO
X	X	X	X	X	0	0	0	Registro de Comando/*Status*
X	X	X	X	X	0	0	1	Porta A
X	X	X	X	X	0	1	0	Porta B
X	X	X	X	X	0	1	1	Porta C
X	X	X	X	X	1	0	0	Temporizador (registro TL)
X	X	X	X	X	1	0	1	Temporizador (registro TH)

onde X = irrelevante (pode ser 0 ou 1)

Tabela 3.1 - Esquema de endereçamento dos registros de I/O.

O endereço XXXXX000 é compartilhado pelos registros de comando e de *status*, ou seja, ambos os registros são acessados

através do mesmo endereço. Se estes registros forem endereçados durante uma operação de escrita, o dado é escrito no registro de comando. Se forem endereçados durante uma operação de leitura, o conteúdo do registro de *status* se torna disponível nas linhas AD0-AD7. É válido observar que não se pode ler o registro de comando nem escrever no registro de *status*.

Com relação a operação das portas, se uma determinada porta estiver atuando como porta de entrada, para acessar seu conteúdo, deve-se realizar uma operação de leitura no respectivo endereço de I/O. Por outro lado, se estiver atuando como saída, para escrever o dado na porta, deve-se efetuar uma operação de escrita no endereço apropriado.

3.2.4 - Registros de Comando e de *Status*

O registro de comando controla o modo de operação das portas de I/O e do temporizador, e o registro de *status* fornece informações sobre o *status* dos mesmos. Para a configuração das portas e do temporizador conforme desejado, um comando apropriado deve ser escrito no registro de comando. A seguir é mostrado o formato deste registro:

Bit:	7	6	5	4	3	2	1	0
	TM_2	TM_1	IEB	IEA	PC_2	PC_1	PB	PA

Os *bits* TM_2 e TM_1 controlam o temporizador e serão descritos no próximo ítem. Os *bits* IEB e IEA, quando em nível lógico 1, habilitam as interrupções das Portas B e A, respectivamente. Os outros 4 *bits* definem o modo de operação das portas.

A Porta A é configurada como porta de entrada se PA=0 e como porta de saída se PA=1. O mesmo se aplica a Porta B com relação ao *bit* PB. A Porta C pode ser configurada como porta de entrada ou porta de saída, ou ainda, como porta de controle para as Portas A e B. Os *bits* PC_2 e PC_1 determinam como age a Porta C:

PC_2	PC_1	Modo de Operação
0	0	Porta de entrada de 6 *bits*
0	1	Controle da Porta A e porta de saída de 3 *bits*
1	0	Controle das Portas A e B
1	1	Porta de saída de 6 *bits*

Quando PC_2PC_1=01, metade da Porta C é usada como uma porta de *handshaking* para a Porta A e a outra metade como porta de saída. Quando PC_2PC_1=10, a Porta C fornece *handshaking* para as Portas A e B. Uma vez que não há necessidade de utilizar *handshaking* neste projeto, este modo não será detalhado.

3.2.5 - Programação do Temporizador

O temporizador do 8155 é um contador decrescente de 14 *bits*, que conta os pulsos no pino de entrada TIN, e gera um sinal de fim de contagem no pino de saída $\overline{\text{TOUT}}$. Este sinal pode ser uma onda quadrada ou um pulso, dependendo da configuração escolhida.

O temporizador é composto por um registro de 16 *bits*, sendo necessários 2 *bytes* de endereço para acessá-lo. O endereço XXXXX100 acessa o *byte* menos significativo do temporizador e o endereço XXXXX101, o *byte* mais significativo (Tabela 3.1). A Figura 3.9 a seguir mostra o formato deste registro:

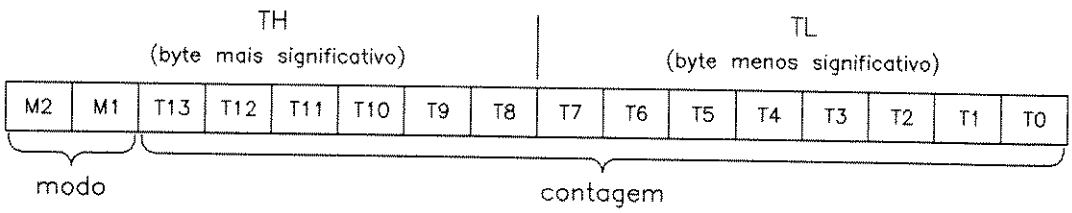


Fig. 3.9 - Formato do temporizador.

Os *bits* 0-13 definem o valor da próxima contagem e os *bits* 14-15 determinam o modo de saída do temporizador, ou seja, o formato do sinal de fim de contagem. Existem quatro modos possíveis:

M2	M1	Modo de saída
0	0	Onda quadrada simples
0	1	Onda quadrada periódica
1	0	Pulso simples
1	1	Pulso periódico

A operação do temporizador é controlada pelos *bits* TM_2 e TM_1, do registro de comando, os quais são usados para iniciar ou parar o contador. Há quatro alternativas:

TM_2	TM_1	Modos de operação
0	0	**NOP:** não afeta a operação do contador
0	1	**STOP:** pára o contador
1	0	**STOP AFTER TC** (*Terminal Count*): pára imediatamente após o final da contagem ser atingido
1	1	**START:** carrega o modo de saída e o valor da contagem e inicia a contagem

O registro do temporizador não é inicializado com um valor em particular quando ocorre um *reset* via *hardware*, porém, o temporizador **não** pára sua contagem. Portanto, a contagem não pode ser iniciada após uma condição de *reset*, e sim somente após um comando START.

3.2.6 - Configuração do 8155

As Portas A e C são configuradas como portas de entrada e a Porta B como porta de saída. A Porta A recebe o valor digital da temperatura do banho enviado pelo conversor analógico-digital. A porta C recebe as informações do circuito de detecção de zero e do nível do líquido. A Porta B controla o conversor A/D e, também, a saída para a válvula do dreno. Assim, o registro de comando deve ser inicialmente carregado com o seguinte valor:

0	0	0	0	0	0	1	0	= 02H

O temporizador do 8155 atua em conjunto com o Temporizador 1 do 8031 que, por sua vez, é utilizado na malha de controle PID

para gerar o tempo que o aquecedor deve permanecer ligado (Capítulo 4, ítem 4.2.2). Foram configurados de modo que um ciclo de contagem completo do Temp. 1 dure cerca de 3 minutos, a fim de evitar ciclos de chaveamento de curta duração que podem gerar ruídos de rádio frequência (RF). A Figura 3.10 mostra o esquema deste conjunto:

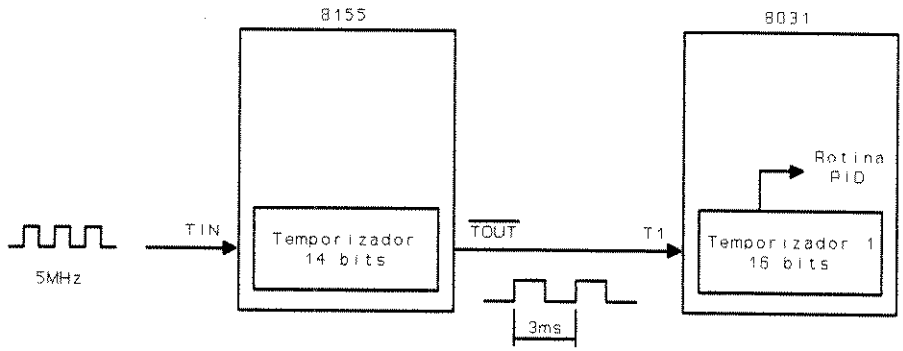


Fig. 3.10 - Esquema de conexão dos temporizadores.

O sinal de entrada do temporizador do 8155 é proveniente do cristal utilizado no oscilador do 8031 e, portanto, sua frequência é 5MHz. O sinal de saída é aplicado na entrada externa do Temporizador 1 do microcontrolador. O temporizador do 8155 fornece um sinal de saída periódico (onda quadrada) e opera com um valor de contagem igual a 16383 (3FFFH), o que resulta em um sinal de saída com período de aproximadamente 3ms. Os valores a serem carregados no registro do temporizador, de modo a obter esta configuração, são:

TH:	0	1	1	1	1	1	1	1	= 7FH
TL:	1	1	1	1	1	1	1	1	= FFH

Os sinais $\overline{CE}$ e IO/$\overline{M}$ estão conectados a Porta 1 do microcontrolador 8031 e o pino RESET é conectado ao pino RESET do 8031. O esquema de ligação do CI 8155 será mostrado no próximo ítem, juntamente com o circuito do conversor A/D.

3.3 - CONVERSÃO ANALÓGICA/DIGITAL

Para a monitoração da temperatura do banho foi utilizado um sensor de temperatura de precisão, o LM135 da National. Este sensor é facilmente calibrável, opera na faixa de temperatura de -55°C a +150°C, possui baixo custo e boa precisão e apresenta **saída linear**, sendo calibrado diretamente em Kelvin. Operando como um diodo zener de 2 terminais, o LM135 fornece uma tensão em seus terminais diretamente proporcional a temperatura absoluta, na taxa de +10mV/K [10].

Para a conversão da tensão fornecida pelo sensor em um valor digital foi utilizado o conversor analógico/digital ADC0802 também da National. Trata-se de um conversor A/D de 8 *bits* com gerador de *clock* interno e tempo de conversão de 100μs. Este conversor opera com uma tensão de alimentação de +5V e com uma faixa de tensão analógica de entrada de 0V a +5V. Apresenta fácil interface com todos os microprocessadores, podendo ser tratado como uma posição de memória ou uma porta de I/O [10].

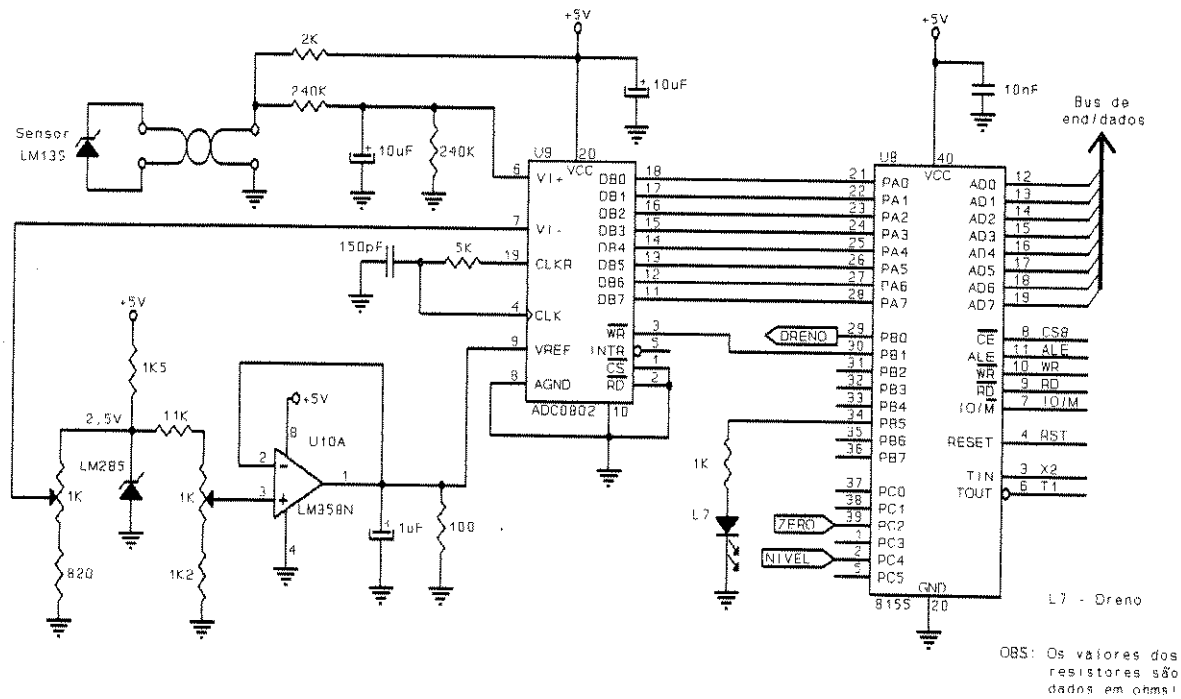


Fig. 3.11 - Esquema de ligação do 8155 e do conversor A/D.

A Figura 3.11 apresenta o circuito de conversão analógica/digital deste projeto, juntamente com o CI 8155. O circuito de conversão está configurado conforme recomendação do próprio fabricante, e tem por objetivo converter o valor analógico da temperatura em um valor digital a ser utilizado pelo microprocessador. Os valores dos resistores e capacitores foram ajustados, de tal modo que, a faixa de temperatura a ser medida é de 0°C a +128°C [10].

A CPU envia um comando ao conversor A/D, através da Porta B do 8155, para que o mesmo inicie a conversão. Após decorrido o tempo de conversão, a CPU lê o dado na saída do conversor através da Porta A do 8155.

Sendo a variação de temperatura é de 0°C a +128°C e o conversor A/D de 8 *bits*, o que fornece 256 combinações possíveis, cada *bit* representa 0,5°C, possibilitando que a resolução do controlador seja de 0,5°C. Portanto, tem-se a seguinte correspondência entre os valores analógicos e digitais da temperatura, respectivamente:

0,0°C	-	0 0 0 0 0 0 0 0
0,5°C	-	0 0 0 0 0 0 0 1
1,0°C	-	0 0 0 0 0 0 1 0
1,5°C	-	0 0 0 0 0 0 1 1
2,0°C	-	0 0 0 0 0 1 0 0
	até	
127,5°C	-	1 1 1 1 1 1 1 0
128,0°C	-	1 1 1 1 1 1 1 1

3.4 - INTERFACE TECLADO/DISPLAY

3.4.1 - Descrição Funcional

A utilização do teclado e do *display*, além de proporcionar maior interação entre o operador e o equipamento, permite ao operador introduzir dados ou parâmetros a serem utilizados pelo programa de controle. O CI 8279 da Intel é utilizado como interface entre a CPU e a unidade de teclado/*display*, aliviando a CPU de fazer a varredura do teclado

e do *display* e, também, de reativar (*refresh*) o *display* [7]. Trata-se de um dispositivo programável, projetado para o uso com microprocessadores de 8 *bits* da linha Intel.

O teclado previsto para operar com o 8279 pode ser formado por uma matriz de até 64 teclas. O *display* pode conter até 16 dígitos, podendo-se utilizar tanto *displays* numéricos quanto alfanuméricos, ou ainda, simples indicadores como um banco de LEDs, por exemplo. O 8279 possui duas memórias (RAM) internas: uma delas armazena o código dos dígitos a serem enviados ao *display* e a outra, do tipo FIFO, armazena o código das teclas pressionadas.

A cada tecla pressionada, o 8279 executa o *debouncing* e ativa uma linha de interrupção, avisando a CPU que existe um dado armazenado na RAM do teclado. A CPU tanto pode ler quanto escrever na RAM do *display*, sendo que ambas as operações podem ser feitas com auto-incremento do endereço.

O 8279 oferece ainda outras opções para a seção do teclado, tais como, matriz de sensores ou entrada *strobed*; opções estas que não serão tratadas neste trabalho, uma vez que o teclado utilizado neste projeto é um teclado matricial. A Figura 3.12 mostra o diagrama de blocos (a) e a pinagem (b) do CI 8279:

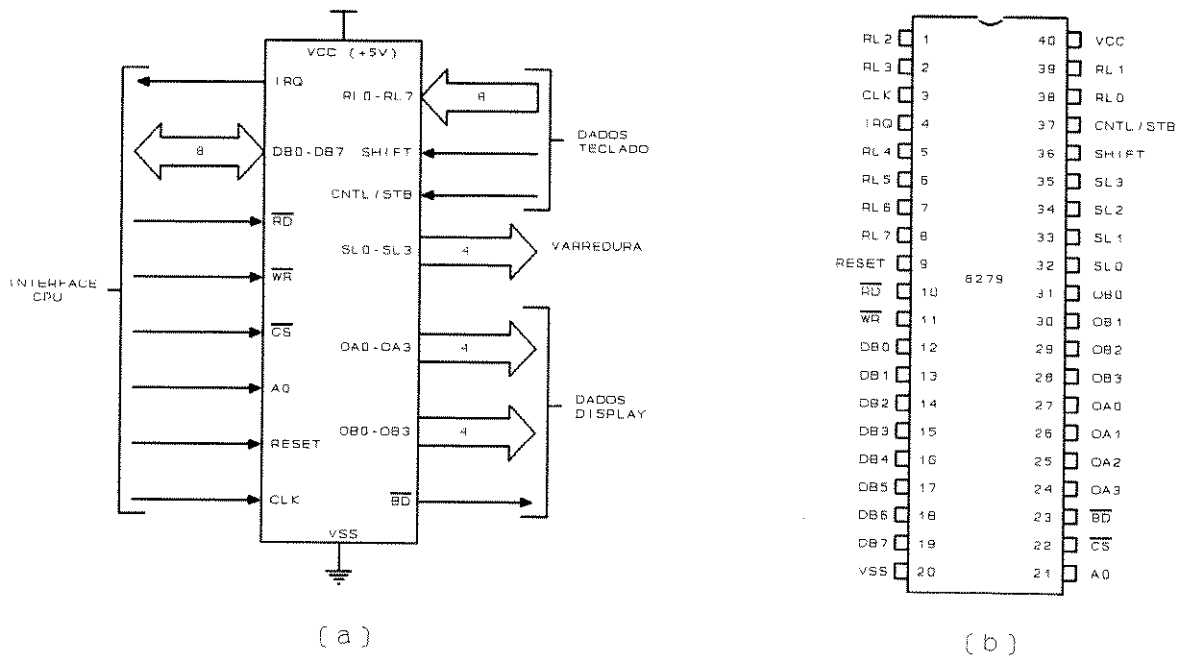


Fig. 3.12 - *Hardware* do CI 8279
(a) - Diagrama de blocos
(b) - Configuração dos pinos

3.4.2 - Descrição dos Pinos do 8279

A seguir, será dada uma descrição dos pinos de maior interesse para o projeto em questão:

DB0-DB7 Estas linhas bidirecionais constituem o barramento de dados. São transmitidos por estas linhas *bytes* de dados e de comando.

CLK Frequência externa usada para gerar a temporização interna.

RESET Um nível alto neste pino reinicializa o 8279.

$\overline{\text{CS}}$ Um nível baixo neste pino habilita o 8279.

A0 Um nível alto nesta linha indica que os sinais recebidos ou enviados se tratam de um comando ou *status*. Um nível baixo indica que são dados.

IRQ Avisa a CPU que há dado válido na RAM do teclado.

SL0-SL3 Estas linhas são usadas para fazer a varredura do teclado e dos dígitos do *display*. Podem ser codificadas (1 de 16) ou decodificadas (1 de 4).

RL0-RL7 Estas linhas de retorno são conectadas às linhas de varredura através das chaves do teclado.

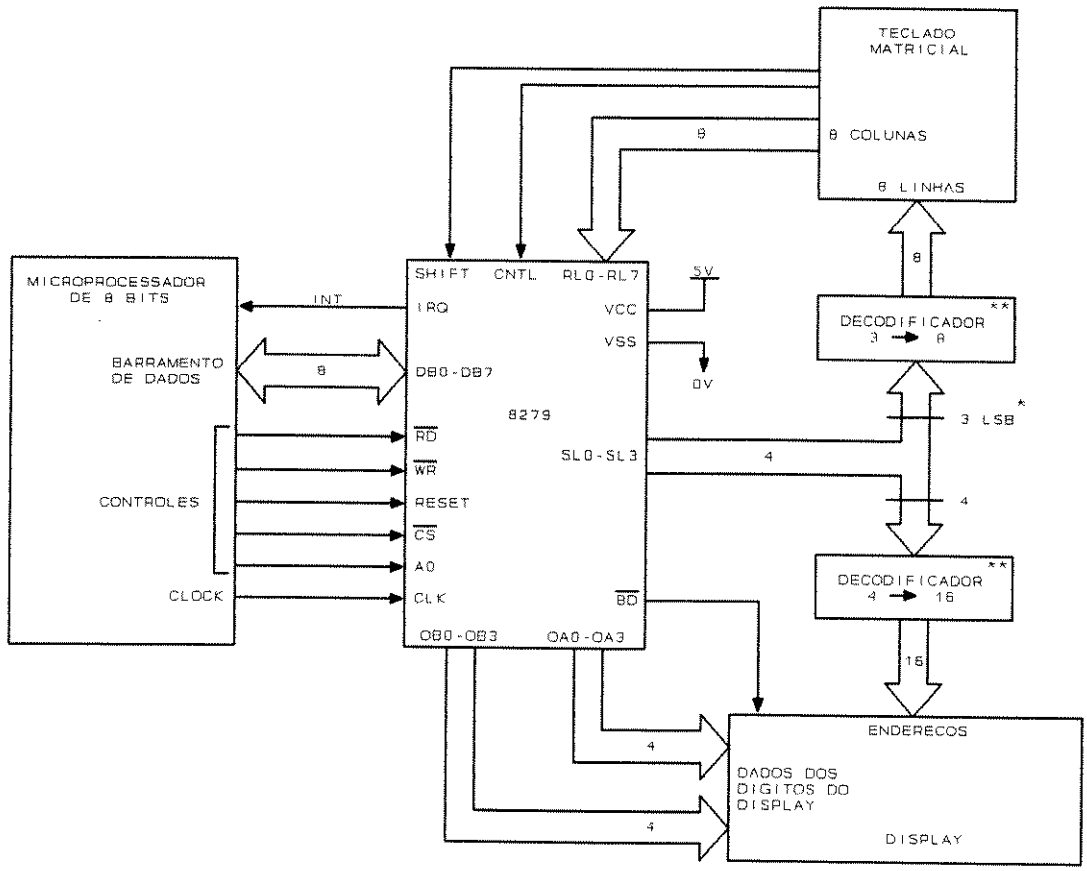
OA0-OA3 **OB0-OB3** Estas duas portas são as saídas para o *display* e podem ser usadas como uma única porta de saída de 8 *bits*. O dado nestas saídas é sincronizado às linhas de varredura, para multiplexar os dígitos do *display*.

3.4.3 - Princípio de Operação

O 8279 é conectado diretamente ao barramento de dados do microprocessador e a comunicação é feita através deste barramento e de algumas linhas de controle. A Figura 3.13 mostra a conexão do 8279 com um sistema microprocessado genérico.

O 8279 possui dois modos de operação. No modo decodificado, as linhas de varredura fornecem uma sequência de 0's. No modo codificado, as linhas de varredura fornecem uma contagem binária, que deverá ser convertida em uma sequência de 0's por um decodificador externo. Quando operando no modo decodificado, o 8279 pode manipular até 32 teclas e um *display*

de até 4 dígitos; no modo codificado pode manipular até 64 teclas e um *display* de até 16 dígitos.



* Não se conecta o decoficador do teclado com o MSB das linhas de varredura.
** Não é utilizado no modo DECODIFICADO.

Fig. 3.13 - Conexão do 8279 com um sistema microprocessado.

O teclado matricial é conectado ao 8279 através das linhas de retorno (RL0-RL7) e das linhas de varredura (SL0-SL3). As linhas de retorno são mantidas em nível lógico 1 por *pullups* internos. A cada linha de varredura ativada, todas as linhas de retorno são examinadas. Quando uma tecla é pressionada, as linhas de retorno e varredura, referentes a esta tecla são curto-circuitadas e, portanto, a linha de retorno passa a apresentar nível lógico **0**. O circuito de varredura determina, assim, qual das teclas da matriz foi pressionada.

Quando o circuito de *debounce* detecta uma chave fechada, espera cerca de 10ms para checar se a referida chave permanece fechada [7]. Caso afirmativo, o código desta tecla é transferido para a memória FIFO (8 x 8). Em seguida, a linha de interrupção IRQ é levada a nível alto, para informar a CPU que

existe dado na FIFO. As teclas pressionadas posteriormente terão seus códigos armazenados em posições sucessivas da memória. Quando a CPU lê o dado na memória, a linha IRQ volta a nível baixo, retornando a nível alto se ainda houver informação na RAM. A CPU tem acesso aos dados armazenados na RAM do teclado, executando uma operação de leitura com A0, $\overline{CS}$ e $\overline{RD}$ em nível baixo.

O *display* é conectado ao 8279 através das linhas de saída (OA0-OA3, OB0-OB3) e das linhas de varredura (SL0-SL3). Estas são responsáveis pela multiplexagem do *display*, pois, como há somente um barramento de saída para o *display*, é necessário que os códigos correspondentes a cada dígito sejam multiplexados. Assim, cada dígito é acionado durante um pequeno intervalo de tempo dentro de um ciclo de varredura do *display*. A varredura contínua possibilita que os dígitos sejam reativados periodicamente.

A RAM do *display* conta com 16 palavras de 8 *bits* e recebe os códigos dos dígitos a serem mostrados no *display*, sendo que cada palavra corresponde a um dígito. Portanto, se for desejado mostrar o número "0" no i-ésimo dígito, deve-se escrever o código de sete segmentos correspondente a este número, na i-ésima posição da RAM. A CPU acessa esta memória através de uma operação de escrita (ou leitura) com A0, $\overline{CS}$ e $\overline{WR}$ (ou $\overline{RD}$) em nível baixo.

3.4.4 - Registros de Comandos

Para que o 8279 possa operar normalmente é necessário que seja inicializado. Os comandos a seguir programam os modos de operação do 8279, sendo enviados pela CPU com $\overline{CS}$ e $\overline{WR}$ em nível baixo e A0 em nível alto. Os 3 *bits* mais significativos das palavras de comando abaixo, especificam qual registro interno será selecionado:

- **Modo do Teclado/*Display***

Bit:	7	6	5	4	3	2	1	0
	0	0	0	D	D	K	K	K

onde DD define o modo de operação *display* e KKK o modo de operação do teclado:

DD

00 8 dígitos de 8 segmentos - entrada à esquerda
01 16 dígitos de 8 segmentos - entrada à esquerda
10 8 dígitos de 8 segmentos - entrada à direita
11 16 dígitos de 8 segmentos - entrada à direita

KKK

000 teclado matricial, modo codificado - *2-key lockout*
001 teclado matricial, modo decodificado - *2-key lockout*
010 teclado matricial, modo codificado - *N-key rollover*
011 teclado matricial, modo decodificado - *N-key rollover*

A denominação entrada à esquerda/direita, refere-se a ordem em que os dígitos serão apresentados no *display*. A denominação *2-key lockout/N-key rollover*, refere-se ao modo de detecção de uma tecla pressionada. É importante notar que, quando o teclado for configurado no modo decodificado, o *display* é reduzido a 4 dígitos, independente do modo escolhido para o *display*; em compensação, elimina-se a necessidade de um circuito de decodificação externo.

- **Frequência**

Bit:	7	6	5	4	3	2	1	0
	0	0	1	P	P	P	P	P

Esta palavra recebe o valor pelo qual deve ser dividida a frequência externa (pino CLK), para gerar a temporização interna. O divisor é um número inteiro programável (*bits* PPPPP), cuja faixa vai de 2 a 31. Deve ser escolhido de modo a produzir uma frequência interna em torno de 100kHz, o que proporciona um tempo de varredura do teclado de 5,1ms e um tempo de *debounce* de 10,3ms [7].

- ***Clear***

Bit:	7	6	5	4	3	2	1	0
	1	1	0	C_D	C_D	C_D	C_F	C_A

Este comando determina o código usado para apagar todos

os dígitos do *display*. Os *bits* C_D (*Clear Display*) selecionam este código, conforme mostrado a seguir:

C_D	C_D	C_D	Código	
0	X	-	00H (0000 0000)	onde X é irrelevante
1	0	-	20H (0010 0000)	
1	1	-	FFH (1111 1111)	

↑ Habilita apagar o *display*, quando igual a 1

Se o *bit* C_F (*Clear* FIFO) é igual a 1, o registro de *status* da FIFO é zerado e a linha IRQ vai para nível baixo. O *bit* C_A (*Clear All*) tem o efeito combinado de C_D e C_F. Quando habilitado, apaga o *display* e também zera o registro de *status* da FIFO.

Após a inicialização, o 8279 está pronto para operar junto ao microprocessador. Durante a operação normal outros dois comandos são amplamente utilizados: escrita na RAM do *display* e leitura da RAM do teclado.

- **Leitura da RAM do teclado**

Bit:	7	6	5	4	3	2	1	0
	0	1	0	AI	X	A	A	A

A CPU deve enviar este comando para iniciar a leitura da FIFO. Todas as leituras subsequentes com A0=0, serão da FIFO, até que outro comando seja enviado. No modo teclado matricial, o *flag* de auto-incremento (AI) e os *bits* de endereço da RAM (AAA) são irrelevantes.

- **Escrita na RAM do *display***

Bit:	7	6	5	4	3	2	1	0
	1	0	0	AI	A	A	A	A

A CPU habilita o 8279 para uma escrita na RAM do *display*, escrevendo este comando. Todas as escritas subsequentes

com A0=0, serão para a RAM do *display*, até que outro comando seja enviado. Os *bits* de endereço AAAA selecionam uma das 16 linhas da RAM. Se AI=1, o endereço será incrementado automaticamente.

3.4.5 - Configuração do 8279

Para o projeto em questão um *display* de 6 dígitos é adequado. Cada dígito é mostrado por um *display* de 7 segmentos com ponto decimal. O teclado é composto por 15 teclas organizadas em uma matriz de 3 linhas x 5 colunas. Pelo fato do *display* possuir mais de 4 dígitos, torna-se necessário utilizar um decodificador externo (CI 74LS156) [8] e, consequentemente, o 8279 deve operar no modo codificado.

Portanto, a configuração para o *display* é 8 dígitos de 8 segmentos, com entrada pela direita, e para o teclado, com varredura codificada *N-key rollover;* resultando na seguinte palavra de comando para o modo do teclado/*display*:

0	0	0	1	0	0	1	0

= 12H

A frequência externa no pino CLK é gerada por um oscilador, cuja frequência é aproximadamente 700KHz (valor prático). Este valor dividido por 7, resultará em uma frequência interna da ordem de 100KHz. Assim, a palavra de comando para a frequência receberá o seguinte valor:

0	0	1	0	0	1	1	1

= 27H

A determinação do valor da palavra para apagar o *display* é feita, sabendo-se que, os dígitos são apagados quando todas as linhas de saída estiverem em nível baixo (00H), o que resulta em:

1	1	0	1	0	0	0	0

= D0H

A Figura 3.14 a seguir mostra o *hardware* da interface teclado/*display*. Foram utilizados transistores e resistores como *drivers* de corrente, melhorando assim a visibilidade do *display*.

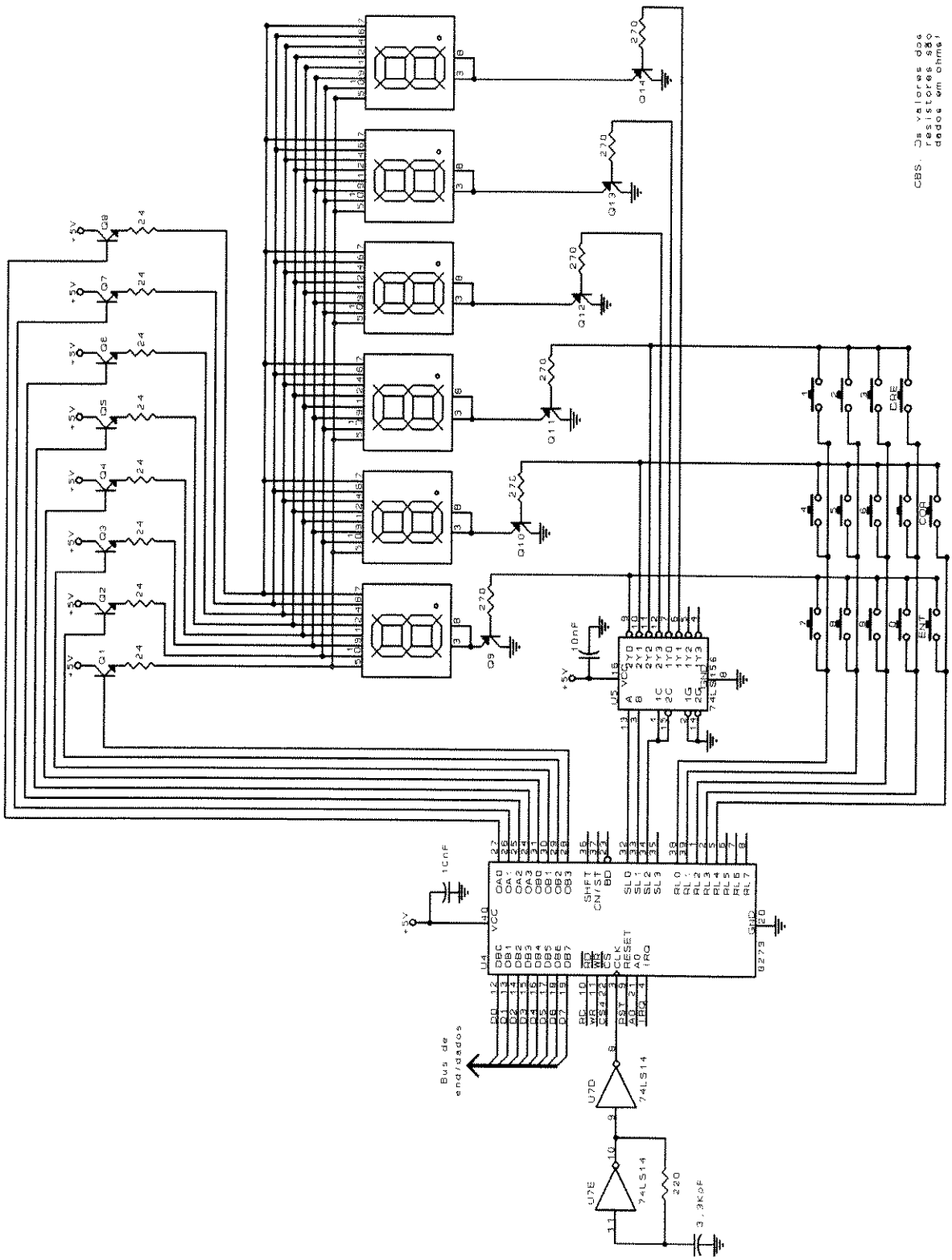


Fig. 3.14 - Esquema de ligação do 8279, do teclado e do *display*.

3.5 - INTERFACE DE POTÊNCIA

3.5.1 - Circuito Controlador de Potência

Este circuito tem por objetivo controlar a potência entregue ao aquecedor, de acordo com um comando da CPU. A Figura 3.15 abaixo apresenta seu esquema:

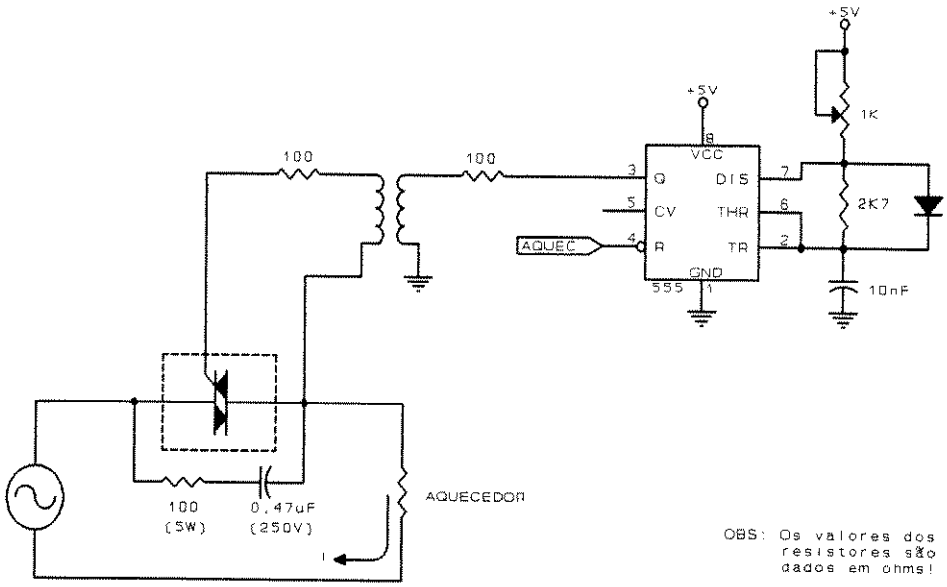


Fig. 3.15 - Esquema de ligação do circuito controlador de potência.

O CI LM555 da National é um temporizador que, neste caso, está operando como astável [10]. Foi configurado de modo a produzir na saída (pino 3) o trem de pulsos mostrado a seguir:

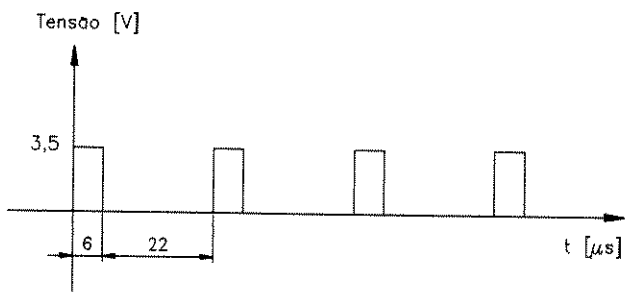


Fig. 3.16 - Pulso de saída do 555 operando como astável.

Pode-se notar na Figura 3.16 que o trem de pulsos na saída do temporizador possui uma frequência de aproximadamente 36kHz, a fim de garantir o disparo do TRIAC e uma boa resposta do transformador de pulso. Enquanto o TRIAC estiver sendo gatilhado pelo trem de pulsos, uma corrente circula pelo aquecedor elevando a temperatura do banho. O TRIAC utilizado neste projeto é o TIC226 da Texas Instruments, projetado para uma corrente de operação máxima de 8A RMS [9]. O circuito controlador de potência contém ainda um circuito *snubber*, cujo propósito é evitar que o TRIAC seja disparado por correntes capacitivas.

O pino 4 (*Reset*) do 555 é usado para retornar a saída ao estado baixo. Um nível baixo neste pino força a saída ao nível lógico **0** independente das entradas e pode ser usado para terminar um pulso de saída prematuramente. A CPU controla a potência a ser entregue ao aquecedor através deste pino. Quando for desejado desligar o aquecedor, um nível lógico **0** deve ser aplicado no pino de *Reset* do 555; caso contrário, um nível alto deve ser aplicado. A Figura 3.17 a seguir apresenta o diagrama de tempo para alguns dos sinais do circuito controlador de potência:

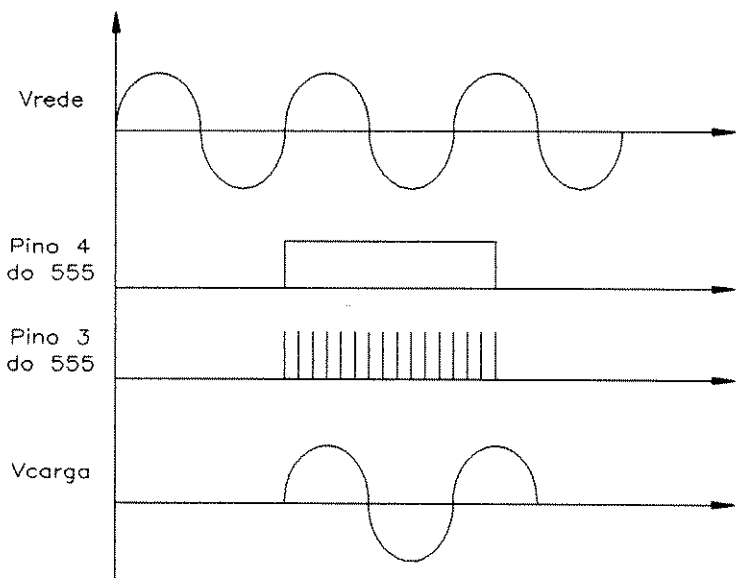


Fig. 3.17 - Diagrama de tempo dos sinais do circuito controlador de potência.

3.5.2 - Fonte e Circuito Detector de Cruzamento de Zero

A Figura 3.18 mostra a implementação da fonte de tensão contínua de +5V (1A) necessária para o funcionamento do sistema. Enquanto a fonte estiver ligada, um LED permanece aceso para sinalização. Caso ocorra uma queda da tensão da fonte para +4,3V, este LED se apaga, indicando um problema com a alimentação do circuito.

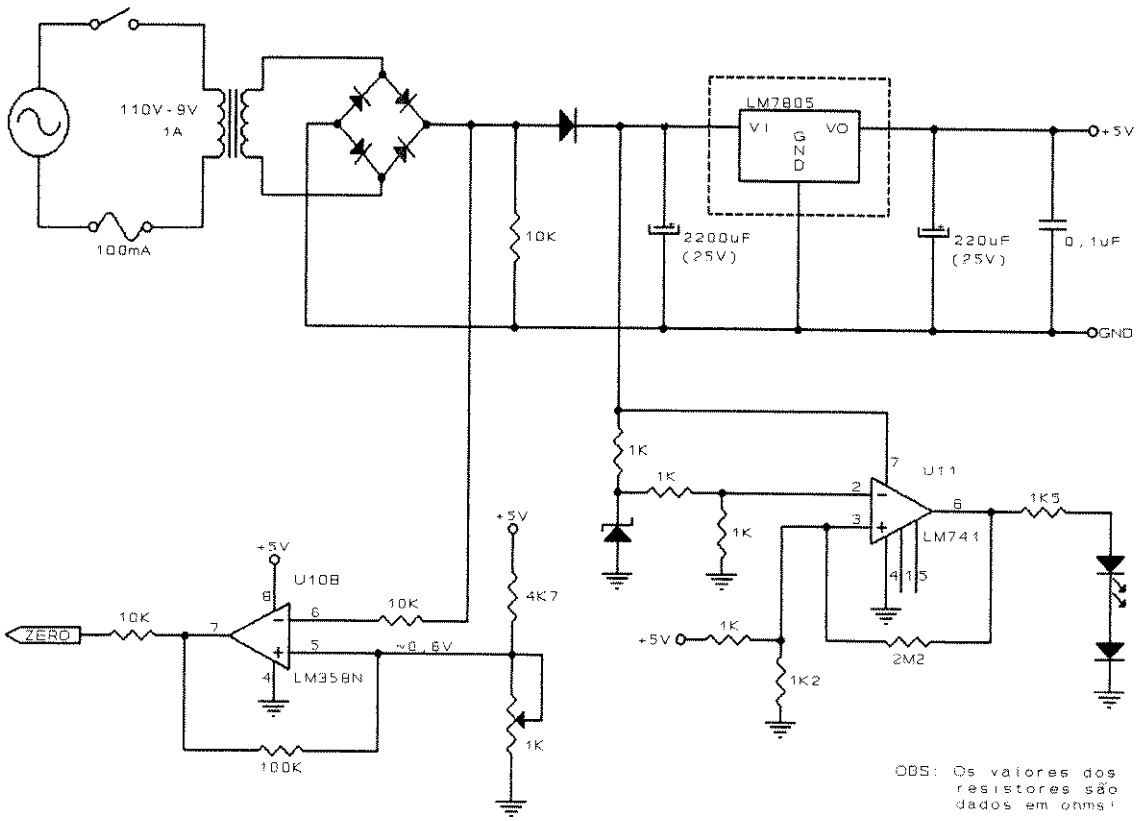


Fig. 3.18 - Esquema de ligação da fonte de +5V e do circuito detector de cruzamento de zero.

Foi implementado também um circuito detector de cruzamento de zero, a fim de evitar altas correntes ao ligar o TRIAC. Este circuito assegura também que nenhuma interferência por rádio frequência (RFI) será gerada quando a carga for ligada ou desligada. A Figura 3.19 apresenta um diagrama de tempo para os principais sinais dos circuitos da Figura 3.18:

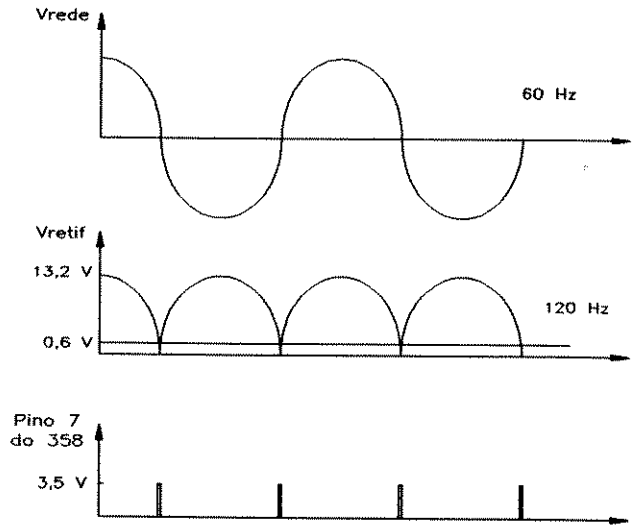


Fig. 3.19 - Diagrama de tempo dos sinais dos circuitos da fonte e do detector de zero.

No Apêndice A encontra-se o esquema de ligação completo do protótipo envolvendo todos os ítens discutidos neste capítulo, ou seja, o circuito principal (constituído pelo microprocessador, interface teclado/*display*, memórias e circuito de conversão A/D), o circuito controlador de potência, a fonte e o circuito detector de cruzamento de zero.

CAPÍTULO 4 - *SOFTWARE*

4.1 - CONTROLADOR PID DIGITAL

4.1.1 - Introdução

O controle eletrônico contínuo, em malha fechada, desenvolveu-se inicialmente através de circuitos analógicos, comparando-se um sinal de referência com o sinal da grandeza a ser controlada, usando-se métodos de realimentação. Atualmente pode ser implementado com circuitos digitais com diversos recursos e vantagens em relação aos analógicos. De uma forma geral pode-se visualizar um sistema de controle contínuo do tipo PID em malha fechada conforme o diagrama da Figura 4.1 abaixo:

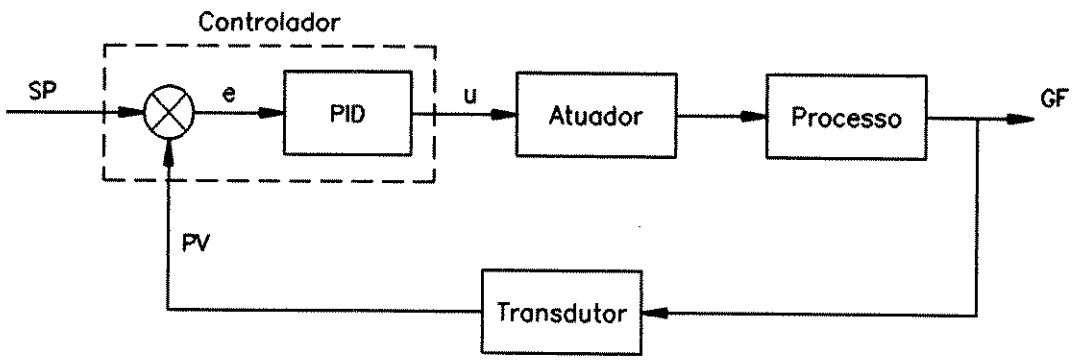


Fig. 4.1 - Sistema básico para controle de processos.

Um sinal de entrada de referência (**SP** - *Set Point*) é comparado com um sinal de realimentação (**PV** - *Process Variable*) da grandeza real controlada, proveniente de um transdutor (velocidade, temperatura, pressão, etc.). O erro resultante (**e**) é amplificado e filtrado, produzindo um sinal de controle (**u**), o qual por sua vez opera o atuador. O atuador é um elemento que altera a entrada para o processo de acordo com o sinal de controle, modificando a grandeza física controlada do processo (**GF**), a fim de reduzir o erro a um valor nulo ou muito pequeno.

O elemento de medida (transdutor) é um dispositivo que converte a variável de saída em uma outra variável conveniente, de modo que possa ser comparada com a entrada de referência. Este elemento é o ramo de realimentação do sistema em malha fechada.

Em sistemas digitais são usados microprocessadores para a implementação do controlador PID, assim como de funções auxiliares que não são fáceis de serem realizadas analogicamente, tais como: geração de curvas de *Set Point*, filtros no laço de realimentação, ganho (P) adaptativo e filtros (I e D) auto sintonizados para minimização do erro.

4.1.2 - Dispositivos de Interface com o Processo

De fundamental importância para a aplicação de microprocessadores em processos são os dispositivos de interface. Isto vem como decorrência das naturezas distintas do processador (digital) e do processo (contínuo).

Inicialmente observe-se que a instrumentação do processo, composta essencialmente por sensores e por atuadores, será aqui considerada como parte integrante do processo, o que é equivalente a dizer que este terá apenas entradas e saídas elétricas. Estes sinais elétricos poderão ser contínuos, quando refletem variáveis do processo, ou digitais, quando são derivados de condições pré-estabelecidas sobre variáveis do processo, por exemplo a violação de valor mínimo ou máximo, gerando um resultado binário do tipo sim ou não. Pode-se assim classificar quatro tipos básicos de sinais:

a) sinal contínuo de entrada no processo
b) sinal contínuo de saída do processo
c) sinal digital de entrada no processo
d) sinal digital de saída do processo

Os sinais dos tipos c e d são da mesma natureza que os sinais existentes nos microprocessadores e na maioria das vezes necessitam apenas ajustes de nível para sua total adaptação. Estes sinais são recebidos ou enviados para o microprocessador através dos dispositivos de interface paralela, tais como, portas de I/O.

Os sinais do tipo a são gerados pelo algoritmo de controle em forma binária e devem ser postos sob forma analógica. O dispositivo responsável por esta transformação é o conversor D/A (Digital/Analógico), o qual para cada palavra binária em sua entrada, origina uma tensão ou corrente proporcional ao valor binário.

Os sinais do tipo b são provenientes do processo em forma analógica (tensão ou corrente) e devem ser postos como uma palavra binária cujo valor seja proporcional àquele. O conversor A/D (Analógico/Digital) é o responsável por esta transformação.

Desta forma, quando os sinais de controle (**u**) e realimentação (**PV**) são analógicos, devem ser usados conversores D/A e A/D, respectivamente. O sistema de controle englobando ambos os casos é mostrado na Figura 4.2 a seguir:

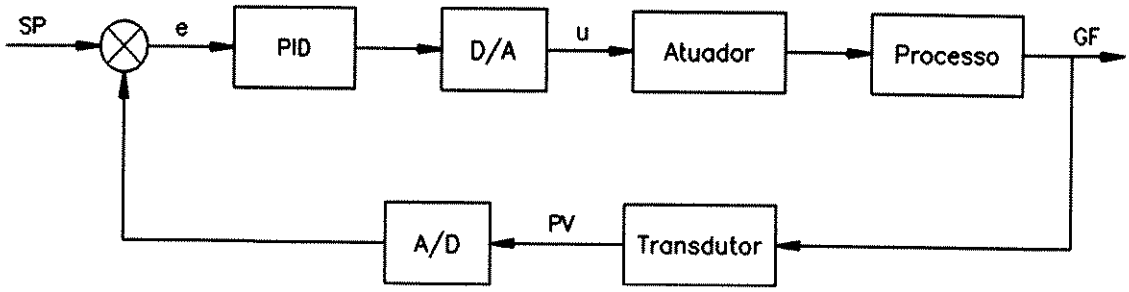


Fig. 4.2 - Diagrama do sistema de controle para sinais de controle e realimentação analógicos.

4.1.3 - Esquema Utilizado para o Controlador de Temperatura

O controle da temperatura de tanques ou fornos aquecidos eletricamente é uma das aplicações de um controlador microprocessado com comportamento do tipo PID. O esquema utilizado neste projeto é apresentado na Figura 4.3 abaixo:

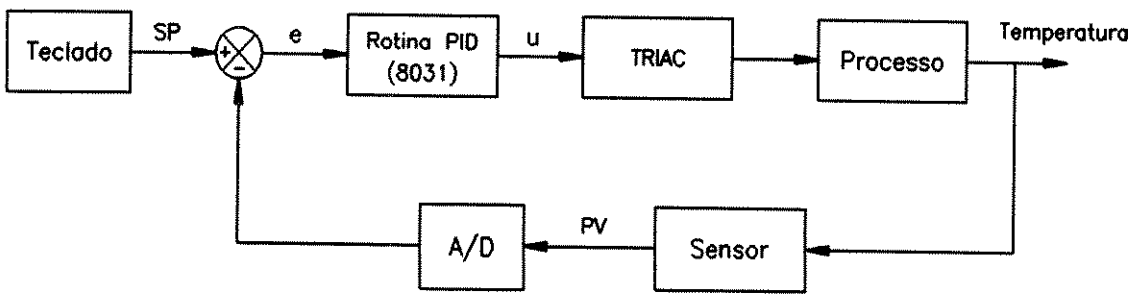


Fig. 4.3 - Diagrama de blocos do sistema de controle para o tanque para *wet-etching/cleaning*.

O microprocessador inicializa o controle PID lendo o valor da temperatura do banho. Em seguida, compara o valor lido com o valor de referência (SP) e estabelece um sinal de erro. Em seguida, é gerado um sinal de controle que determina se o aquecedor deve ou não ser energizado, de modo a igualar a temperatura do banho ao valor de referência.

O atuador neste caso é o circuito controlador de potência, cujo principal elemento é o TRIAC, e o sinal de controle (u) um sinal digital de entrada no processo. Quando em nível lógico 1, o TRIAC é gatilhado permitindo a passagem de corrente pelo aquecedor e, consequentemente, elevando a temperatura do banho. O sinal de saída do processo, por ser um sinal contínuo, necessita de um conversor A/D. O valor de referência é um parâmetro fornecido pelo operador através do teclado.

4.1.4 - Discretização da Lei de Controle

A resposta final do controle PID é o resultado da soma das ações de três controladores independentes, o proporcional (P), o integral (I) e o derivativo (D).

O controlador proporcional é essencialmente um amplificador com um ganho ajustável. O sinal de controle fornecido por este controlador é diretamente proporcional ao sinal de erro e representa o quanto a variável a ser controlada difere do valor de referência (SP). É caracterizado por uma maior sobre elevação (*overshoot*), quando o ganho é elevado. A expressão que descreve a ação de controle proporcional para um sistema contínuo é dada por:

$$P(t) = Ge(t) \qquad (4.1)$$

onde, G é o ganho proporcional e e(t) o sinal de erro.

Para um sistema digital a equação (4.1) deve assumir a forma discreta, dada por:

$$P(t_j) = Ge(t_j) \qquad (4.2)$$

onde, t_j é o instante de amostragem e $e(t_j)$ é valor do erro no instante t_j, como mostra a Figura 4.4a a seguir:

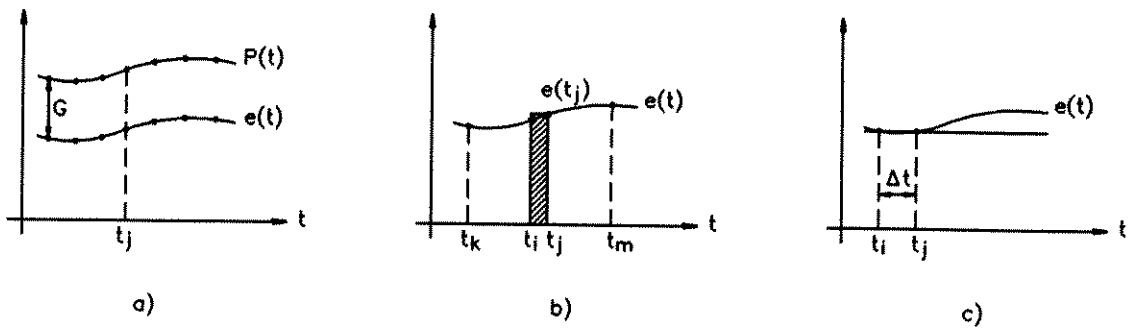


Fig. 4.4 - Discretização dos sinais de controle (a) proporcional, (b) integral e (c) derivativo.

O controlador com ação de controle integral tem por função integrar o sinal de erro e(t), e quando associado ao controle proporcional, garante um erro em regime nulo. A ação integral pode ser descrita para um sistema contínuo como:

$$I(t) = \frac{1}{\tau_i} \int_{t_k}^{t_m} e(t)\, dt \qquad (4.3)$$

onde, τ_i é a constante de integração conhecida como constante de tempo de integração, t_k e t_m definem o intervalo para a integração, sendo t_k o instante inicial e t_m o instante final.

Para um sistema digital a Equação (4.3) pode ser discretizada substituindo-se a integral por um somatório, seguindo-se a aproximação retangular [1], como mostra a Figura 4.4b, o que resulta:

$$I(t_j) = \frac{1}{\tau_i} \sum_{t_j=t_k}^{t_m} \left\{ e(t_j)\,(t_j - t_i) \right\} = \frac{1}{\tau_i} \sum_{t_j=t_k}^{t_m} \left\{ e(t_j)\ \Delta t \right\} \qquad (4.4)$$

onde, t_i é o instante imediatamente anterior ao instante de amostragem t_j e Δt o intervalo de amostragem ($\Delta t = t_j - t_i$).

A ação de controle derivativa responde à taxa de variação do sinal de erro e permite ao controlador uma ação antecipatória, podendo produzir uma correção do sistema antes que o sinal de erro torne-se demasiadamente grande. Quando adicionado ao controlador proporcional, o controlador derivativo aumenta a sensibilidade do sistema. A ação de controle derivativa para um

sistema contínuo pode ser descrita pela expressão:

$$D(t) = \tau_d \frac{de(t)}{dt} \qquad (4.5)$$

onde, τ_d é a constante de derivação conhecida como constante de tempo derivativo.

Para um sistema digital a Equação (4.5) pode ser discretizada substituindo-se a derivada por uma diferença de primeira ordem, seguindo-se a aproximação retangular [1]. Deste modo, tem-se:

$$D(t_j) = \tau_d \frac{e(t_j) - e(t_i)}{t_j - t_i} = \tau_d \frac{e(t_j) - e(t_i)}{\Delta t} \qquad (4.6)$$

onde, $e(t_j)$ e $e(t_i)$ são os valores dos erros nos instantes t_j e t_i, respectivamente, como mostra a Figura 4.4c.

Nas expressões digitais, quanto menor o valor de Δt em relação à constante de tempo do sistema a ser controlado, maior a precisão dos resultados. Isso também é válido para o número de *bits* do sistema digital envolvido na representação dos números (quanto maior o número de *bits* melhor a precisão).

O controlador PID pode ser obtido combinando-se as três ações de controle, sendo que esta ação combinada possui as vantagens de cada uma das ações de controle individuais. Isoladamente, estas ações de controle apresentam uma série de problemas, mas combinadas apresentam características próprias que garantem robustez e eficiência. Para um sistema contínuo o controle PID pode ser descrito por:

$$u(t) = Ge(t) + \frac{1}{\tau_i} \int_{t_k}^{t_m} e(t)\, dt + \tau_d \frac{de(t)}{dt} \qquad (4.7)$$

onde, u(t) é a função de controle.

Utilizando-se as equações discretizadas (4.2), (4.4) e (4.6), obtém-se a função de controle PID para um sistema digital:

$$u(t_j) = Ge(t_j) + \frac{1}{\tau_i} \sum_{t_j=t_k}^{t_m} \left\{ e(t_j)\ \Delta t \right\} + \tau_d \frac{e(t_j) - e(t_i)}{\Delta t} \qquad (4.8)$$

Enquanto a ação de controle derivativa possui a vantagem de ser antecipatória, tem as desvantagens de amplificar os sinais de ruído e causar um efeito de saturação no atuador.

A função de controle PID quando empregada em um sistema que apresenta uma descontinuidade no sinal de erro e(t) pode levar o sistema a instabilidade, devido a presença da parte derivativa. Pois, na descontinuidade, o termo $\{e(t_j)-e(t_i)\}$ da equação (4.6) poderá ter um valor muito grande, e como o termo Δt deve ser pequeno por causa da precisão, o valor inicial para a parte derivativa tenderá a um valor grande. Na prática, nos sistemas analógicos o valor da parte derivativa é limitado à ±VCC e nos sistemas digitais este valor é limitado ao tamanho da palavra do microprocessador. Para contornar este problema, é comum a utilização de um modelo de derivador segundo a Equação (4.9):

$$D(t) = \frac{\tau_d}{\tau_f} e(t) - \frac{1}{\tau_f} \int_{t_k}^{t_m} D(t)\, dt \qquad (4.9)$$

onde, τ_f é uma constante que, neste projeto em particular, receberá o mesmo valor da constante de derivação ($\tau_f=\tau_d$).

Para um sistema digital a Equação (4.9) pode ser discretizada por processo análogo aos das Equações (4.2) e (4.4), o que resulta em:

$$D(t_j) = \frac{\tau_d}{\tau_f} e(t_j) - \frac{1}{\tau_f} \sum_{t_j=t_k}^{t_m} \left\{ D(t_j)\ \Delta t \right\} \qquad (4.10)$$

A Figura 4.5 apresenta o diagrama de blocos do controlador PID para esta situação, ou seja, com uma ação de controle derivativa com filtragem:

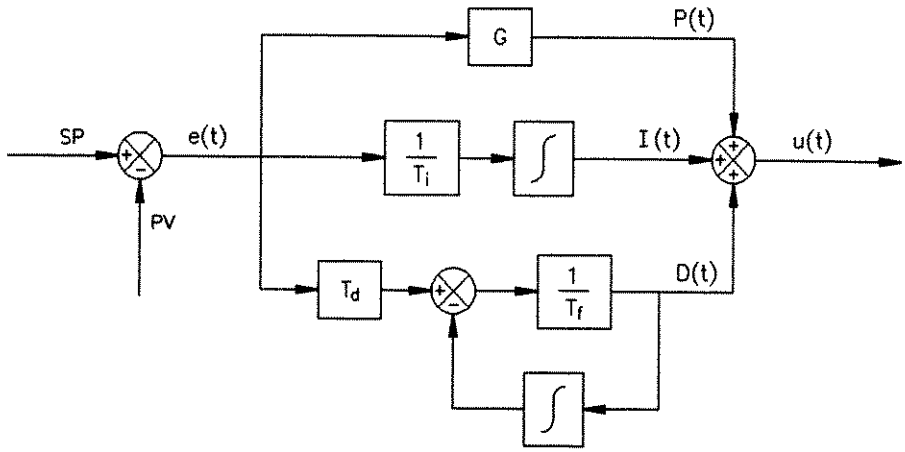


Fig. 4.5 - Malha de controle PID para um sistema que apresenta descontinuidade no sinal de erro e(t).

A expressão que representa a ação de controle PID mostrada na Figura 4.5, é dada por:

$$u(t_j) = Ge(t_j) + \frac{1}{\tau_i}\sum_{t_j=t_k}^{t_m}\left\{e(t_j)\Delta t\right\} + \frac{\tau_d}{\tau_f}e(t_j) - \frac{1}{\tau_f}\sum_{t_j=t_k}^{t_m}\left\{D(t_j)\Delta t\right\} \quad (4.11)$$

Esta expressão reflete o tratamento dado pelo algoritmo de controle PID adotado neste projeto.

4.2 - PROGRAMA DE CONTROLE

O *software* de controle é o responsável pelo controle da temperatura através de uma rotina PID, bem como pela recepção e tratamento dos dados, monitoração dos parâmetros, sinalização, atuação no circuito controlador de potência, abertura da válvula do dreno, e também contém as rotinas de operação do teclado e do *display*.

O programa de controle foi desenvolvido em linguagem C e compilado para o "*assembler*" do 8051 através do *software* AvCase [11]. A listagem completa do programa encontra-se no APÊNDICE B deste trabalho.

O programa é composto por uma função principal (main) e por várias outras funções (subrotinas), o que torna o programa estruturado e mais fácil de ser entendido. A seguir, serão

apresentadas as principais subrotinas e também o programa principal de um ponto de vista funcional, onde os detalhes de implementação são omitidos.

4.2.1 - Subrotinas do programa de controle

- INICIALIZ: subrotina de inicialização

É responsável pelas inicializações dos CIs programáveis 8031, 8155 e 8279, conforme as configurações definidas nos ítens 3.1.12, 3.2.6 e 3.4.5 do Capítulo 3, respectivamente. Esta subrotina também desliga o aquecedor, desabilita a abertura do dreno, apaga os LEDs e acende todos os segmentos do *display*.

- ESPERA: subrotina de temporização

Esta subrotina fornece um atraso de aproximadamente 0,5 segundo, necessário para a visualização do dado no *display*.

- Subrotinas de interface com o usuário

Estas subrotinas possibilitam ao programa de controle comunicar com o operador através da unidade teclado/*display*:

- **CLRDISPLAY:** apaga o conteúdo do *display*, enviando o código D0 (all zeros) para o 8279 (ítem 3.4.5 do Capítulo 3).
- **DISPLAY:** converte um *byte* em sua representação em código de sete segmentos e envia ao *display*.
- **TECLADO:** quando uma tecla é pressionada, o 8279 armazena na RAM do teclado um código que representa a posição da tecla na matriz. Esta subrotina converte o código da tecla pressionada no valor da referida tecla.
- **TECL_DEC:** executa o atendimento do teclado. Consiste em ler as teclas pressionadas, realizar o tratamento das teclas, acionar o *display* (escrever no *display* a tecla pressionada) e armazenar o número digitado em uma variável interna (NUMDEC).
- **DEC_DISP:** escreve no *display* um número decimal.

- READ_TEMP: subrotina de leitura da temperatura

É responsável pela leitura do valor da temperatura do banho. Envia um comando ao conversor A/D, através da Porta B do 8155, para que o mesmo inicie a conversão. Espera cerca de 200μs e, em seguida, lê a Porta A do 8155 armazenando o valor em uma variável interna (TEMP).

- NIVEL: subrotina de verificação do nível

Lê a Porta C do 8155 e verifica o *bit* que recebe a informação do nível do líquido. Se este *bit* estiver em nível lógico **0**, o que significa que o nível não está adequado, esta rotina sinaliza esta situação e impede que o programa prossiga. Caso contrário, o programa segue normalmente.

- ZERO_CROSS: subrotina de detecção de cruzamento de zero

É responsável por detectar a passagem por zero da forma de onda da tensão da rede. Esta subrotina lê a Porta C do 8155 e verifica o *bit* que recebe a informação do cruzamento por zero. Enquanto este *bit* não apresentar nível lógico **1**, esta subrotina será repetida.

4.2.2 - Programa Principal

O programa principal engloba a rotina de controle PID e, para uma melhor compreensão, será tratado com maior nível de detalhamento. Este programa primeiramente executa a subrotina INICIALIZ e, em seguida, espera que o operador aperte a tecla <EXE> ou a tecla <DRE>. Nenhuma outra tecla terá efeito algum.

Caso seja apertada a tecla <DRE>, o programa executa a subrotina READ_TEMP e exibe o valor da variável de processo (PV), ou seja, da temperatura do banho no *display*. Se este valor for inferior ao valor da variável TEMP_DRE o programa habilita a abertura da válvula do dreno, sinaliza esta situação e é interrompido neste ponto. Caso contrário, o programa sinaliza a ocorrência deste erro de operação e volta a esperar por uma ação

do operador. O fluxograma da Figura 4.06 abaixo mostra a lógica de funcionamento do programa:

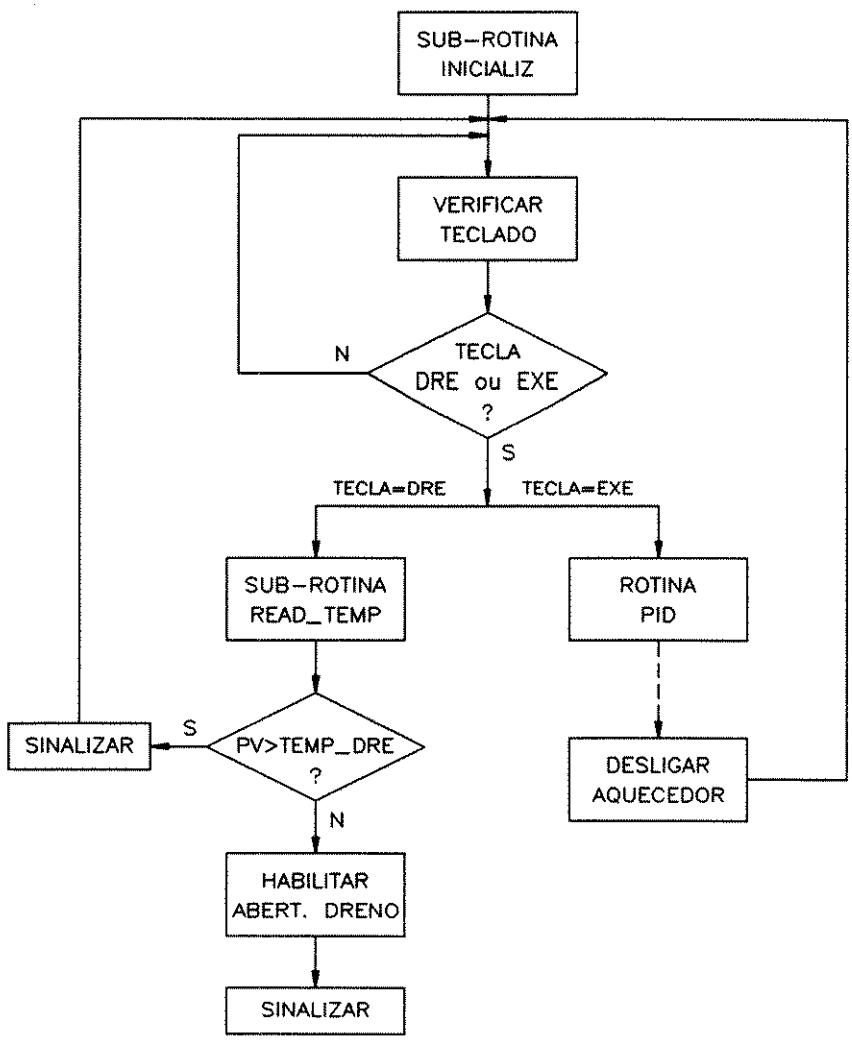


Fig. 4.06 - Fluxograma do programa principal.

Caso seja apertada a tecla <EXE>, o programa iniciará a rotina de controle PID propriamente dita. Esta rotina é executada repetidamente, sendo interrompida somente quando ocorrer uma temperatura excessiva ou quando o operador apertar a tecla <FIM>. O fluxograma da Figura 4.07 a seguir mostra a rotina PID com maiores detalhes. Uma descrição detalhada das tarefas envolvidas permite uma melhor compreensão do algoritmo.

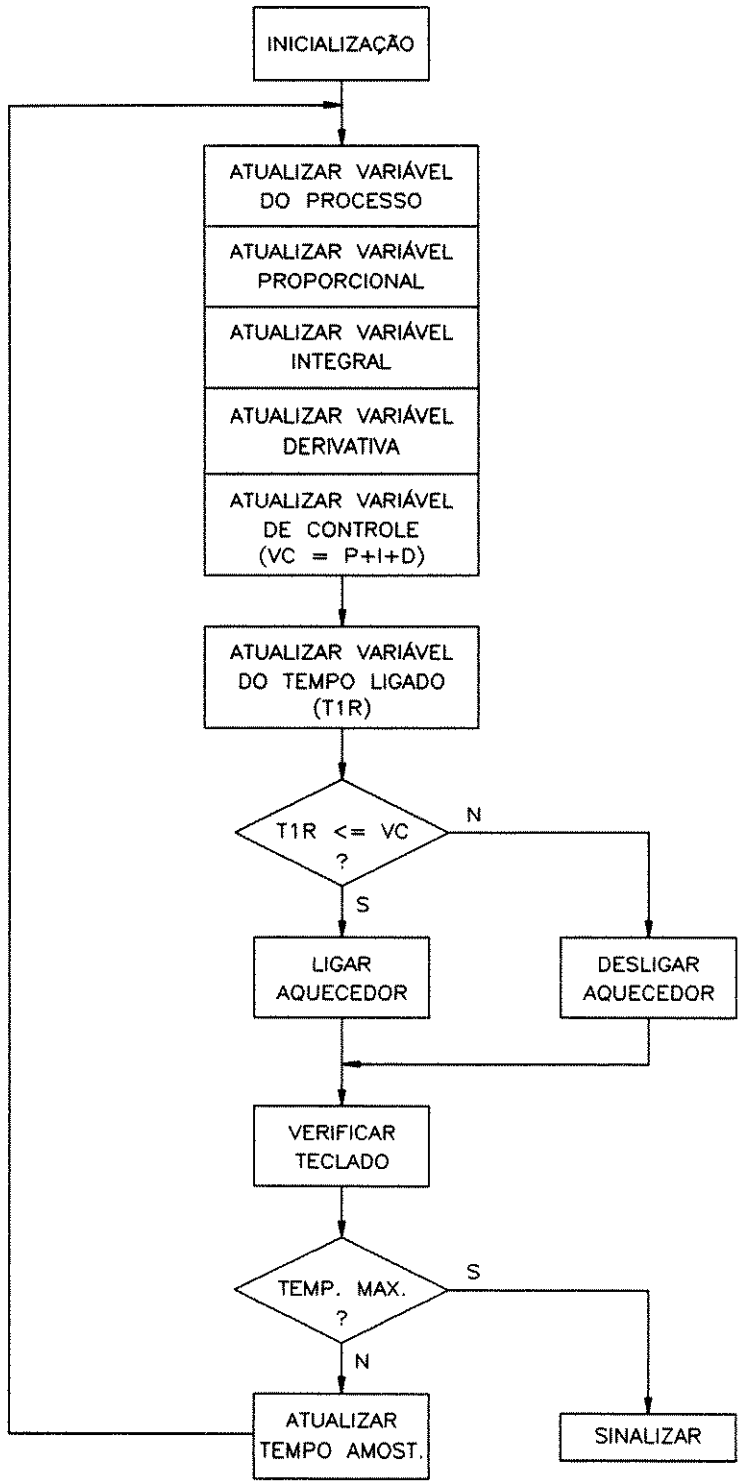


Fig. 4.07 - Fluxograma do algoritmo PID.

INICIALIZAÇÃO DA ROTINA PID

Primeiramente, a rotina de controle PID executa a subrotina NIVEL a fim de verificar se o nível do líquido está adequado à operação. Em seguida, recebe os valores dos parâmetros via teclado: SP, GA, TI, TD (onde TI=$1/\tau_i$ e TD=$1/\tau_d$). Nesta etapa também são inicializadas as variáveis do programa.

ATUALIZAÇÃO DA VARIÁVEL DO PROCESSO

A subrotina READ_TEMP é executada e o valor atual da temperatura do banho armazenado na variável PV. O erro resultante da comparação entre este valor e o valor de referência (SP) é armazenado na variável E. A cada período de amostragem estes valores são atualizados, para que um novo valor da variável de controle seja gerado.

ATUALIZAÇÃO DA VARIÁVEL PROPORCIONAL

A variável proporcional é o produto do sinal de erro (E) pelo ganho (GA), Equação (4.2). A cada período de amostragem um novo valor da variável proporcional é gerado.

ATUALIZAÇÃO DA VARIÁVEL INTEGRAL

O cálculo da variável integral segue a fórmula apresentada na Equação (4.4). O valor calculado é somado ao valor da memória integral, uma vez que se trata de um somatório. Ao valor da memória integral são impostos limites inferior e superior. A cada período de amostragem um novo valor desta variável é gerado.

ATUALIZAÇÃO DA VARIÁVEL DERIVATIVA

O cálculo da variável derivativa segue a fórmula apresentada na Equação (4.10). O valor calculado é somado ao valor da memória derivativa, ao qual também são impostos limites inferior e superior. A cada período de amostragem um novo valor desta variável é gerado. Tanto a variável integral quanto a derivativa são dependentes do período de amostragem.

ATUALIZAÇÃO DA VARIÁVEL DE CONTROLE

A variável de controle (VC) é calculada somando-se os valores das variáveis proporcional, integral e derivativa,

Equação (4.11). Da mesma forma que as memórias integral e derivativa, a variável de controle possui um limite máximo e um limite mínimo. Seu valor é atualizado a cada período de amostragem.

ATUALIZAÇÃO DA VARIÁVEL DO TEMPO LIGADO

O Temporizador 1 do 8031 é responsável por indicar o tempo que o aquecedor permanece ligado. Este tempo é armazenado na variável T1R e atualizado a cada período de amostragem.

TESTE DO TEMPO LIGADO

O tempo que o aquecedor deve permanecer ligado é proporcional à variável de controle. Assim, se T1R for menor que VC, o aquecedor deve ser ligado, caso contrário deve ser desligado. Antes porém de ligar o aquecedor, a subrotina ZERO_CROSS é executada, garantindo que o TRIAC será disparado somente na passagem por zero da forma de onda da tensão da rede.

VERIFICAÇÃO DO TECLADO

Nesta etapa o programa verifica se há tecla pressionada, porém atenderá somente às teclas: <FIM>, <SP>, <TEMP>, <SP+> e <SP->. Nesta etapa são feitos também os testes de desvio do *Set Point*, sendo sinalizado quando da sua ocorrência.

TESTE DE TEMPERATURA MÁXIMA

A variável do processo é comparada com a variável TEMP_MAX, que indica a temperatura máxima permitida. Caso esta temperatura tenha sido atingida, uma sinalização é fornecida, a rotina PID encerrada e o aquecedor desligado.

ATUALIZAÇÃO DO TEMPO DE AMOSTRAGEM

O Temporizador 0 do 8031 temporiza a duração da rotina PID, gerando o período de amostragem a ser utilizado no próximo ciclo da rotina.

CAPÍTULO 5 - RESULTADOS E CONCLUSÕES

5.1 - CONSIDERAÇÕES FINAIS

Neste trabalho implementou-se um protótipo de um controlador microprocessado com a finalidade de controlar as funções de um tanque para *wet-etching/cleaning*. Infelizmente não foi possível testar este sistema de controle no tanque para *wet-etching/cleaning* propriamente dito, e por isso algumas alterações se fizeram necessárias no decorrer do projeto. Aparentemente isto seria um problema, porém, tornou-se uma vantagem, uma vez que este controlador se tornou mais versátil, podendo inclusive controlar outros tipos de tanques ou sistemas com apenas algumas modificações.

O sistema de controle desenvolvido apresenta-se adequado do ponto de vista de controle (*hardware* e *software*) e seu comportamento pode ser avaliado com o protótipo em operação. A implementação deste protótipo permitiu extrair resultados experimentais essenciais para a validação do sistema.

Os testes foram realizados em um sistema de aquecimento com as seguintes características:

- Volume: 500ml
- Líquido: Água
- Temperatura inicial do líquido: 25°C
- Elemento térmico: 120VAC; 500W; 4,16A; 28,8 ohms

O teste consistiu em aquecer a água até a temperatura de 50°C (*Set Point*), variando-se o ganho e medindo-se o tempo que o sistema leva para estabilizar. Foram efetuadas várias medidas do tempo de estabilização para diferentes valores das constantes TI e TD. A Figura 5.1 apresenta a família de curvas para TI variando entre 0,5s e 50s, com TD constante e igual a 0,1s. A Figura 5.2 mostra a família de curvas análoga para TD variando entre 0,5s e 50s, com TI constante e igual a 0,1s.

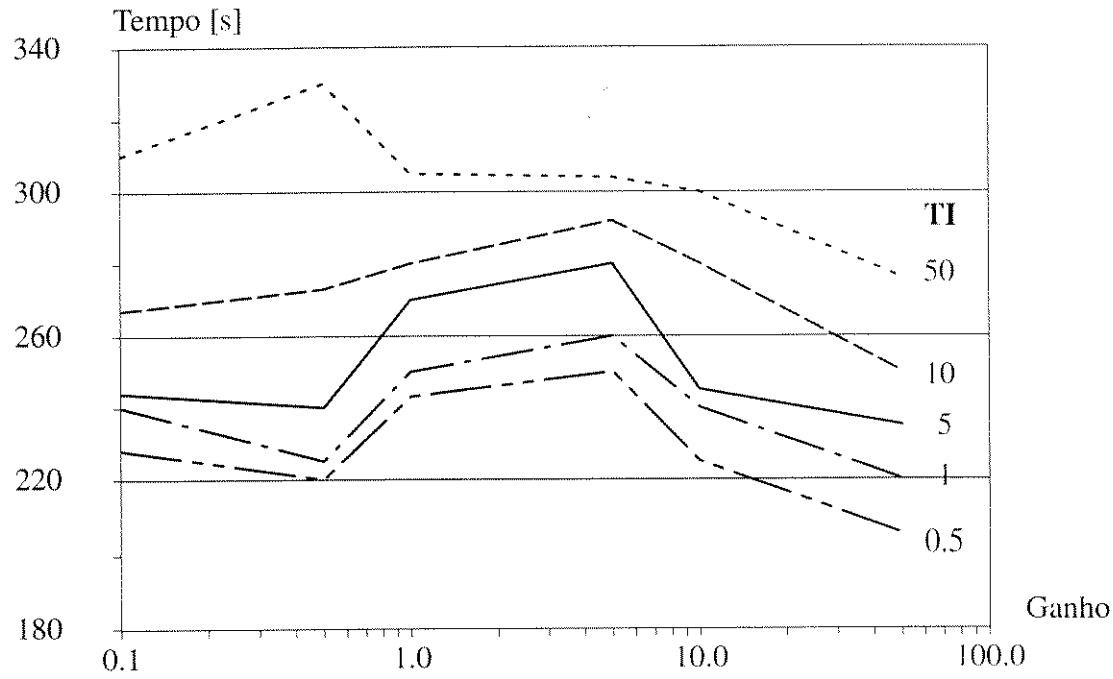


Fig. 5.1 - Família de curvas para TD constante.

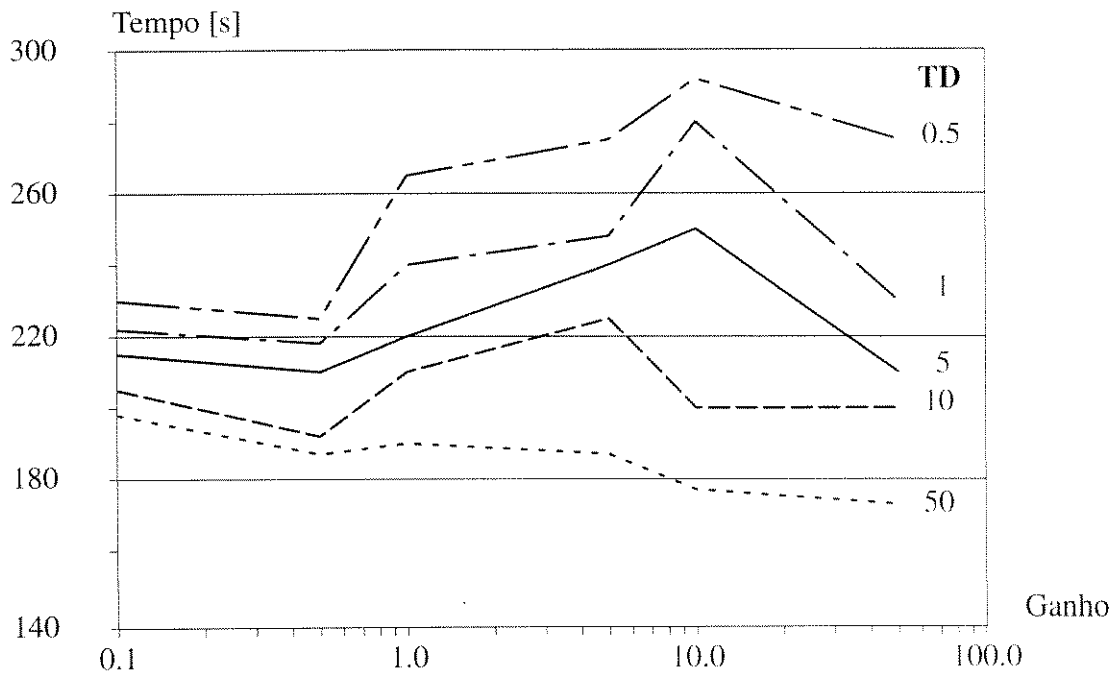


Fig. 5.2 - Família de curvas para TI constante.

Pode-se verificar na Figura 5.1 que, quanto maior o valor de TI maior o tempo de estabilização da temperatura. Na Figura 5.2 tem-se que, quanto maior o valor de TD menor o tempo de estabilização. Observou-se ainda durante as medidas que, para valores de TD maiores do que 5s, a temperatura de operação oscila entre 49,5°C e 52°C, e o mesmo ocorre para valores de TI inferiores a 0,5s. Portanto, não são valores recomendáveis, uma vez que ultrapassam a tolerância de variação máxima da temperatura de operação (1%), comentada no Capítulo 1.

Os valores do ganho e das constantes TI e TD que proporcionaram uma variação da temperatura de operação de no máximo 1% foram: GA=5, TI=1s e TD=1s.

Os resultados obtidos revelam que o controlador desenvolvido neste trabalho possui as características desejadas, e o desempenho obtido nos testes realizados demonstram a eficiência do mesmo. Todas as condições de alarmes e *status* estabelecidas no Capítulo 2 são atendidas e o modo de operação está de acordo conforme descrito no referido capítulo.

Pode-se concluir, portanto, que este controlador possui as características fundamentais dos controladores PID de uso geral. Pode ser utilizado em uma série de sistemas de controle onde o erro de regime, a confiabilidade e a robustez sejam parâmetros importantes, como é o caso do controle de processos industriais.

5.2 - SUGESTÕES PARA APRIMORAMENTOS

Por se tratar de um protótipo, quando finalizado o trabalho, verificou-se que algumas melhorias poderiam ter sido introduzidas. A seguir, serão comentadas algumas sugestões de melhoramentos, para uma futura utilização deste trabalho como base para novos trabalhos envolvendo controladores PID microprocessados.

Um conversor analógico/digital com maior número de *bits*, aumentaria a precisão e o fundo de escala do equipamento, melhorando o desempenho do sistema.

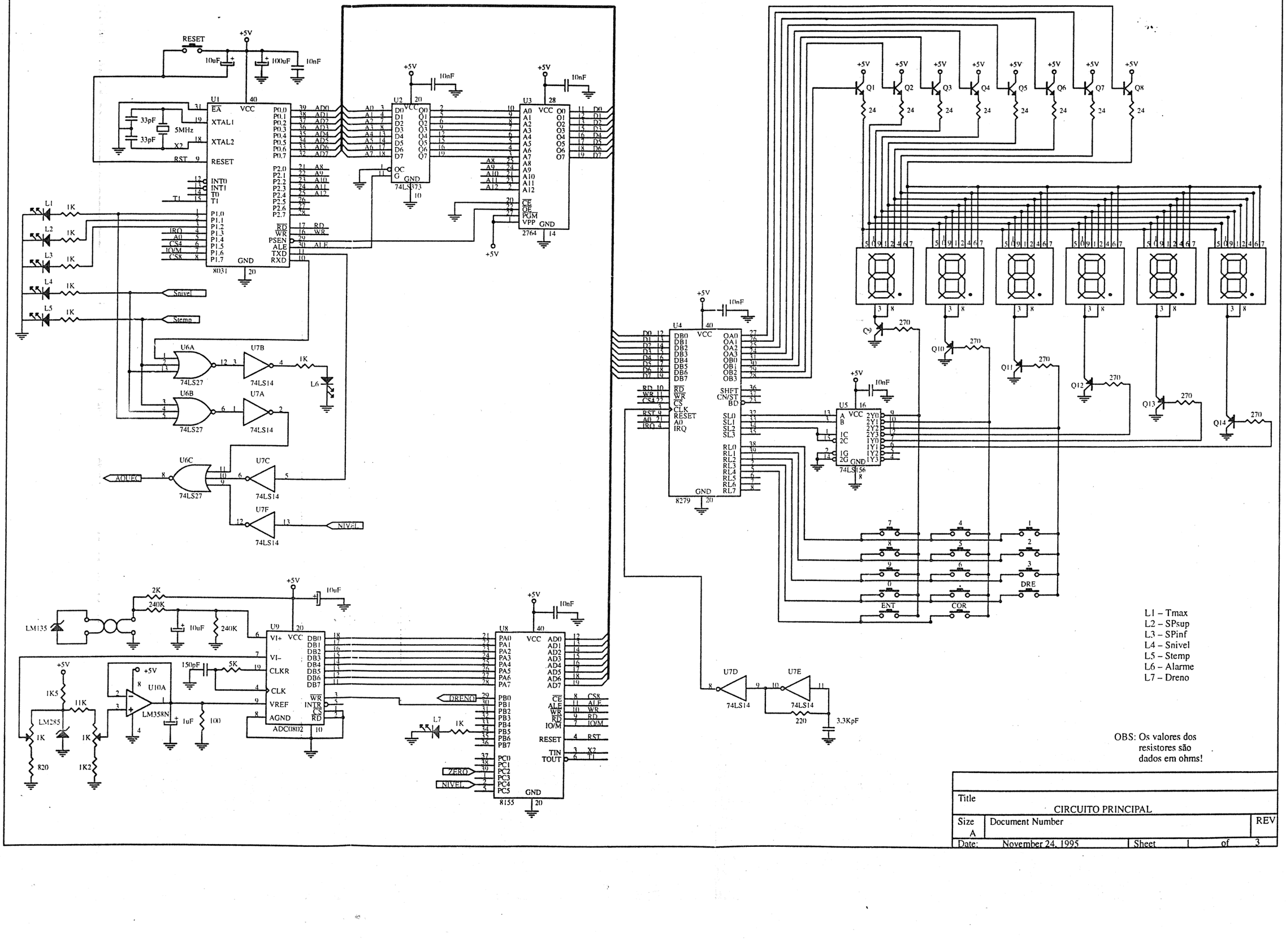
Poderiam ser introduzidos, um sensor exclusivo para monitoração de temperatura excessiva, garantindo uma maior confiabilidade para o sistema, e um aviso sonoro para anunciar as várias condições de alarme juntamente com a parte visual.

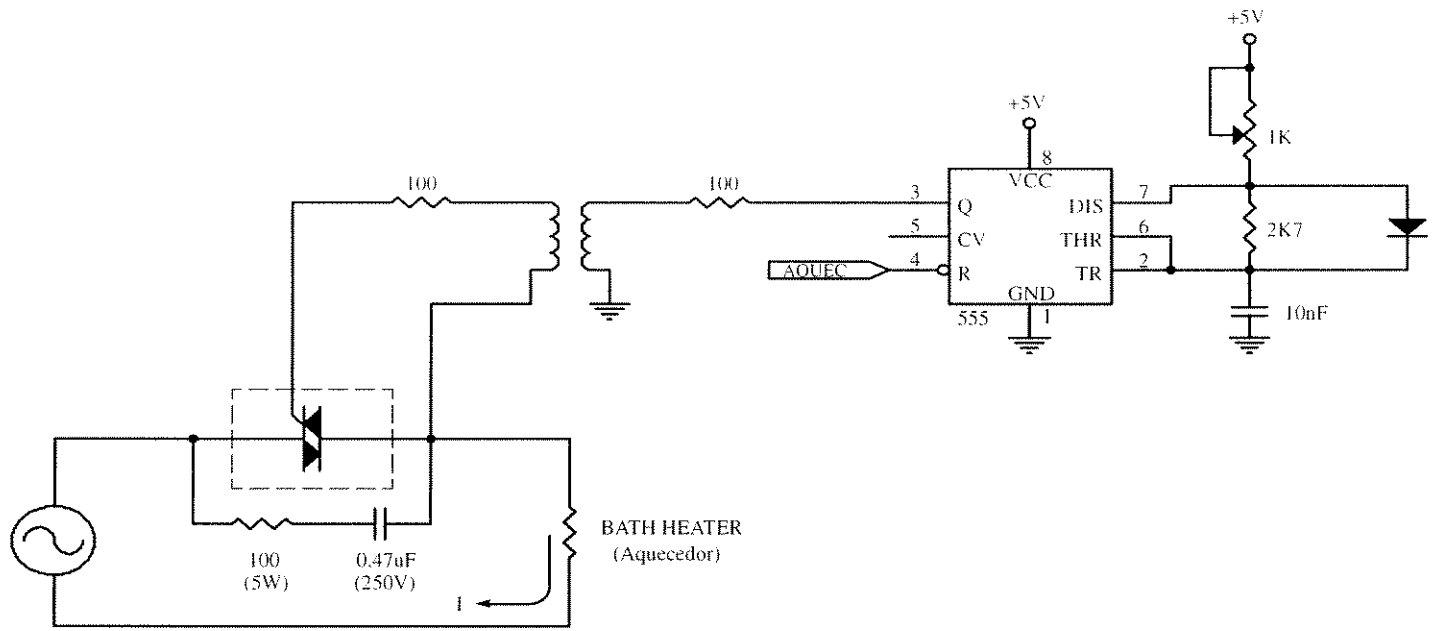
No protótipo em questão o sensor é calibrado por *hardware*. Um ajuste para calibração por *software* poderia ser implementado, permitindo a eliminação de erros do sistema e do sensor de maneira mais confortável. Estes erros podem provocar uma diferença entre a temperatura atual do banho e a temperatura mostrada no *display*. Este ajuste consistiria da introdução de mais um parâmetro fornecido via teclado, que levaria o *display* a uma concordância com a temperatura atual do banho.

Finalmente, como última sugestão, para uma produção em escala industrial o conjunto formado pelo microprocessador 8031, pelo *latch* 74LS373 e pela memória EPROM 2764, poderia ser substituído pelo microprocessador 8051 devidamente programado pelo fabricante. Isto simplificaria o *hardware* e protegeria o programa. Porém, o programa deveria ser modificado de modo a ocupar no máximo 4K *bytes* de memória ROM.

APÊNDICE A - *HARDWARE* DO CONTROLADOR

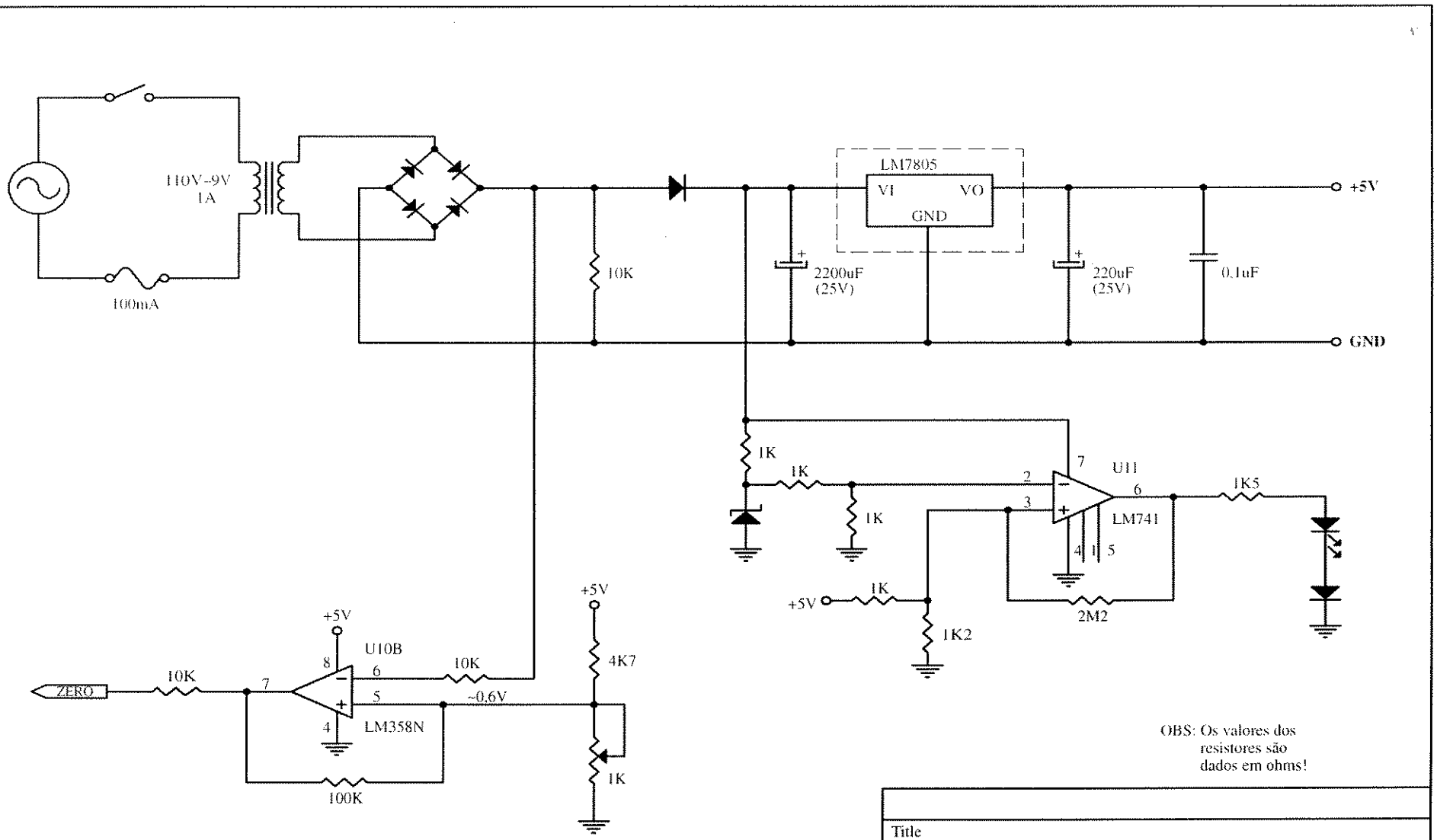
ESQUEMA DE LIGAÇÃO DO SISTEMA DE CONTROLE MICROPROCESSADO


RESET
+5V
10uF
100uF
10nF
U1
8031
EA
VCC
XTAL1
XTAL2
RST
RESET
33pF
5MHz
X2
P0.0 AD0
P0.1 AD1
P0.2 AD2
P0.3 AD3
P0.4 AD4
P0.5 AD5
P0.6 AD6
P0.7 AD7
P2.0 A8
P2.1 A9
P2.2 A10
P2.3 A11
P2.4 A12
INT0
INT1
T0
T1
P1.0
P1.1
P1.2 IRQ
P1.3 A0
P1.4 CS4
P1.5 IO/M
P1.6 CS8
P1.7
RD
WR
PSEN
ALE
TXD
RXD
GND
20
U2
74LS373
D0 D1 D2 D3 D4 D5 D6 D7
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
OC
G
GND
U3
2764
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12
O0 O1 O2 O3 O4 O5 O6 O7
CE
OE
PGM
VPP
L1 1K
L2 1K
L3 1K
L4 1K
L5 1K
Snivel
Stemp
U6A
U6B
U6C
U7A
U7B
U7C
U7F
74LS27
74LS14
L6
AOUEC
NIVEL
U4
8279
DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7
OA0 OA1 OA2 OA3 OB0 OB1 OB2 OB3
SHFT
CN/ST
BD
CLK
RESET
IRQ
SL0 SL1 SL2 SL3
RL0 RL1 RL2 RL3 RL4 RL5 RL6 RL7
U5
74LS156
Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8
24
Q9 Q10 Q11 Q12 Q13 Q14
270
U7D
U7E
220
3,3KpF
U9
ADC0802
VI+
VI-
CLKR
CLK
VREF
AGND
DB0 DB1 DB2 DB3 DB4 DB5 DB6 DB7
WR
INTR
CS
RD
LM135
2K
240K
10uF
150pF
5K
U10A
LM358N
1K5
11K
LM285
1K
820
1K2
1uF
100
U8
8155
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7
AD0 AD1 AD2 AD3 AD4 AD5 AD6 AD7
PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7
PC0 PC1 PC2 PC3 PC4 PC5
CE
ALE
WR
RD
IO/M
RESET
TIN
TOUT
CS8
RST
X2
T1
DRENO
ZERO
NIVEL
L7
U5
74LS156
1C
2C
1G
2G
DRE
ENT
COR
L1 – Tmax
L2 – SPsup
L3 – SPinf
L4 – Snivel
L5 – Stemp
L6 – Alarme
L7 – Dreno
OBS: Os valores dos resistores são dados em ohms!
Title
CIRCUITO PRINCIPAL
Size
A
Document Number
REV
Date: November 24, 1995
Sheet 1 of 3



OBS: Os valores dos resistores são dados em ohms!

Title: CIRCUITO CONTROLADOR DE POTÊNCIA		
Size: A	Document Number	REV
Date: November 24, 1995	Sheet 2 of 3	



OBS: Os valores dos resistores são dados em ohms!

Title: FONTE / CIRCUITO DETECTOR DE CRUZAMENTO DE ZERO		
Size: A	Document Number	REV
Date: November 24, 1995	Sheet 3 of 3	

APÊNDICE B - *SOFTWARE* DO CONTROLADOR

LISTAGEM DO PROGRAMA DE CONTROLE DESENVOLVIDO EM LINGUAGEM C

```
struct port1
{
   unsigned   CS8:1,   /* P1.7 */
             IO_M:1,   /* P1.6 */
              CS4:1,   /* P1.5 */
               A0:1,   /* P1.4 */
              IRQ:1,   /* P1.3 */
             LED3:1,   /* P1.2: SPinf */
             LED2:1,   /* P1.1: SPsup */
             LED1:1;   /* P1.0: Tmax  */
};
struct port3
{
   unsigned
             dummy:6,
             HEAT:1,   /* P3.1: AQUEC  */
             LED6:1;   /* P3.0: Alarme */
};
#define  PORT1 ( *(struct port1 *) 0x90 )
#define  PORT3 ( *(struct port3 *) 0xB0 )
#define  TCON  ( *(char *) 0x88 )
#define  TMOD  ( *(char *) 0x89 )
#define  TL0   ( *(char *) 0x8A )
#define  TL1   ( *(char *) 0x8B )
#define  TH0   ( *(char *) 0x8C )
#define  TH1   ( *(char *) 0x8D )
#define  IE    ( *(char *) 0xA8 )
#define  DRE 10
#define  ENT 11
#define  COR 12

#define  OUTPORT0 ( *(char *) 0x0100 )
void outbyte0 ( char valor0 )
{
   OUTPORT0 = valor0;
}
#define  INPORT1 ( (char *) 0x0101 )
char inbyte1 ( )
{
   return ( *INPORT1);
}
#define  OUTPORT2 ( *(char *) 0x0102 )
void outbyte2 ( char valor2 )
{
   OUTPORT2 = valor2;
}
#define  INPORT3 ( (char *) 0x0103 )
char inbyte3 ( )
{
   return ( *INPORT3);
}
```

```
#define  OUTPORT4 ( *(char *) 0x0104 )
void outbyte4 ( char valor4 )
{
   OUTPORT4 = valor4;
}
#define  OUTPORT5 ( *(char *) 0x0105 )
void outbyte5 ( char valor5 )
{
   OUTPORT5 = valor5;
}

const int MIL=1000, CEM=100, DEZ=10;
const unsigned int _2=2, _12=12, _256=256;
const unsigned int INT_MAX=65535, TEMP_MAX=125, TEMP_DRE=60;
const float DEZ_R=10.0, ZERO_R=0.0, T_INI=0.020, LIM_SP=5.0;
const float LIM_INF=0.0, LIM_SUP=4095.0, TCLK=0.2e-6;
unsigned char wd;
float tecl_dec(), read_temp();

void espera (void)
{
   int i;
   for ( i=0; i<10000; i++);
}

void inicializ (void)
{
   int i;
   /*  Inicializacao do 8031 */
   IE = 0;
   TMOD = 0x51;
   TCON = 0x50;
   PORT1.LED1 = 0;
   PORT1.LED2 = 0;
   PORT1.LED3 = 0;
   PORT3.LED6 = 0;
   PORT3.HEAT = 0;
   /*  Inicializacao do 8279 */
   PORT1.CS4  = 0;
   PORT1.A0   = 1;
   espera ();
   outbyte0 ( 0x12 );
   outbyte0 ( 0x27 );
   outbyte0 ( 0x90 );
   PORT1.A0   = 0;
   for ( i=1; i<9; i++ ) outbyte0 ( 0xFF );
   espera ();
   espera ();
   PORT1.CS4  = 1;
   /*  Inicializacao do 8155 */
   PORT1.CS8  = 0;
   PORT1.IO_M = 1;
   outbyte0 ( 0x02 );
   outbyte2 ( 0x02 );
   outbyte4 ( 0xFF );
   outbyte5 ( 0x7F );
   outbyte0 ( 0xC2 );
```

```
    PORT1.CS8  = 1;
}

void clrdisplay (void)
{
    char i;

    PORT1.CS4  = 0;
    PORT1.A0   = 1;
    outbyte0 ( 0xD0 );
    for ( i=0; i<3; i++ ) ;
    PORT1.CS4  = 1;
    wd = 0x96;
}

void display ( char c, char p )
{
    char d;

    if ( c == 0 ) d = 0xF3;
    if ( c == 1 ) d = 0x60;
    if ( c == 2 ) d = 0xB5;
    if ( c == 3 ) d = 0xF4;
    if ( c == 4 ) d = 0x66;
    if ( c == 5 ) d = 0xD6;
    if ( c == 6 ) d = 0xD7;
    if ( c == 7 ) d = 0x70;
    if ( c == 8 ) d = 0xF7;
    if ( c == 9 ) d = 0xF6;

    if ( c =='A') d = 0x77;
    if ( c =='D') d = 0xE5;
    if ( c =='E') d = 0x97;
    if ( c =='G') d = 0xD7;
    if ( c =='I') d = 0x40;
    if ( c =='N') d = 0x45;
    if ( c =='O') d = 0xC5;
    if ( c =='P') d = 0x37;
    if ( c =='R') d = 0x05;
    if ( c =='S') d = 0xD6;
    if ( c =='T') d = 0x87;
    if ( c =='=') d = 0x84;
    if ( c =='-') d = 0x04;
    if ( c ==' ') d = 0x00;

    if ( p ) d = d + 0x08;

    PORT1.CS4  = 0;
    PORT1.A0   = 1;
    outbyte0 ( wd );
    PORT1.A0   = 0;
    outbyte0 ( d );
    PORT1.CS4  = 1;
    wd++;
    if ( wd == 0x98 ) wd = 0x90;
}
```

```
char teclado ( )
{
   char dado, digito, linha, coluna;

   PORT1.CS4  = 0;
   PORT1.A0   = 1;
   outbyte0 ( 0x40 );
   PORT1.A0   = 0;
   dado = inbyte1 ();
   PORT1.CS4  = 1;
   linha  = dado & 0x38;
   linha >>= 3;
   coluna = dado & 0x07;
   if ( coluna == 0 )
   {
      if ( linha == 0 ) digito = 7;
      if ( linha == 1 ) digito = 4;
      if ( linha == 2 ) digito = 1;
   }
   if ( coluna == 1 )
   {
      if ( linha == 0 ) digito = 8;
      if ( linha == 1 ) digito = 5;
      if ( linha == 2 ) digito = 2;
   }
   if ( coluna == 2 )
   {
      if ( linha == 0 ) digito = 9;
      if ( linha == 1 ) digito = 6;
      if ( linha == 2 ) digito = 3;
   }
   if ( coluna == 3 )
   {
      if ( linha == 0 ) digito = 0;
      if ( linha == 1 ) digito ='.';
      if ( linha == 2 ) return ( DRE );
   }
   if ( coluna == 4 )
   {
      if ( linha == 0 ) return (ENT);
      if ( linha == 1 ) return (COR);
   }

   return  ( digito );
}

void error (void)
{
   clrdisplay ();
   display ('E',0);
   display ('R',0);
   display ('R',0);
   display ('O',0);
   espera ();
}
```

```
float tecl_dec ()
{
   char counti=0, countf=0, flag=0, fim=0;
   char cent=0, dez=0, uni=0, frac=0;
   char tecla;
   float numdec;

   while ( PORT1.IRQ == 0 ) ;
   clrdisplay ();
   while ( fim == 0 )
   {
      if ( PORT1.IRQ == 0 ) continue;
      tecla = teclado ();
      if ( tecla == COR ) return (-1);
      if ( tecla == DRE || tecla == ENT )
      {
         if (tecla==ENT && (counti != 0 || countf != 0))
         fim = 1;
         else
         {
            error();
            return (-1);
         }
      }
      if ( tecla == '.' )
         if ( flag == 1 ) continue;
         else
         {
            flag = 1;
            clrdisplay ();
            if ( cent != 0 ) display ( cent, 0 );
            if ( dez != 0 || cent != 0 ) display ( dez, 0 );
            if ( uni == 0 && counti == 0 ) display ( ' ',1);
            else display ( uni, 1 );
         }
      if ((tecla >= 0 && tecla <= 9) && flag == 0 )
      {
         if ( counti > 2 )
         {
            error ();
            return (-1);
         }
         if ( counti == 2 )
         {
            cent = dez;
            dez = uni;
         }
         if ( counti == 1 ) dez = uni;
         uni = tecla;
         counti++;
         display ( tecla, 0 );
      }
      if ((tecla >= 0 && tecla <= 9) && flag == 1 )
      {
         if ( countf > 0 ) continue;
         frac = tecla;
         countf++;
```

```
            display ( frac, 0 );
         }
      }
      numdec = cent*CEM + dez*DEZ + uni + frac/DEZ_R;
      return ( numdec );
   }

   void dec_disp ( float n )
   {
      int cent, dez, uni, frac, nint;

      n = n * DEZ_R;
      nint = (int) n;

      cent = nint/MIL;
      nint = nint%MIL;
      if ( cent != 0 ) display ( cent, 0 );
      dez = nint/CEM;
      nint = nint%CEM;
      if ( dez != 0 || cent != 0 )  display ( dez, 0 );
      uni = nint/DEZ;
      frac = nint%DEZ;
      display ( uni, 1 );
      display ( frac, 0 );
   }

   float read_temp ()
   {
      char intr;
      unsigned char t;
      float temp;

      PORT1.CS8 = 0;
      outbyte2 ( 0 );
      outbyte2 ( 2 );
      for ( intr=0; intr<3; intr++ ) ;
      t = ( unsigned char ) inbyte1 ();
      PORT1.CS8 = 1;
      temp = (float) t/2.;
      return ( temp );
   }

   char prog_dre ( )
   {
      char tecla;

      clrdisplay ();
      display ( 0, 0);
      display ('P',0);
      while ( 1 )
      {
         if ( PORT1.IRQ == 0 ) continue;
         tecla = teclado ();
         if ( tecla == 3 ) return ('P');
         if ( tecla == DRE ) return ('D');
      }
   }
```

```
void nivel ( void )
{
   char nivel;

   PORT1.CS8 = 0;
   nivel = inbyte3 ();
   PORT1.CS8 = 1;
   nivel = nivel & 0x10;
   while ( nivel == 0 )
   {
      error ();
      clrdisplay ();
      display ('N',0);
      display ('I',0);
      espera ();
   }
}

void zero_cross ( void )
{
   char zero;

   do {
      PORT1.CS8 = 0;
      zero = inbyte3 ();
      PORT1.CS8 = 1;
      zero = zero & 4;
   } while ( zero != 4 );
}

void main ( void )
{
   unsigned char tl0, th0, tl1, th1;
   unsigned int  t1a, t1b, t1r, VCi, PVi;
   char modo, count, flag, pid, tecla, disp, splim;
   float SP, PV, E, G, P, I, T, TI, ID, D, TD, TF, VC;
   wd = 0x96;

   inicializ ();
   while ( 1 )
   {
      modo = prog_dre ();
      if ( modo == 'D' )
      {
         PV = read_temp ();
         clrdisplay ();
         dec_disp ( PV );
         espera ();
         PVi = (unsigned int) PV;
         if ( PVi > TEMP_DRE )
         {
            error ();
            clrdisplay ();
            display ('D',0);
            display ('R',0);
            espera ();
         }
```

```
    else
    {
       PORT1.CS8 = 0;
       outbyte2 ( 0x23 );
       PORT1.CS8 = 1;
       for ( ; ; ) ;
    }
    continue;
}

/* Verificacao do nivel */
nivel ();

/* Recebimento dos parametros via teclado */
do {
   clrdisplay ();
   display ('S',0);
   display ('P',0);
   SP = tecl_dec ();
} while ( SP == -1 );
clrdisplay ();
dec_disp ( SP );
espera ();
do {
   clrdisplay ();
   display ('G',0);
   display ('A',0);
   G = tecl_dec ();
} while ( G == -1 );
clrdisplay ();
dec_disp ( G );
espera ();
do {
   clrdisplay ();
   display ('T',0);
   display ('I',0);
   TI = tecl_dec ();
} while ( TI == -1 );
clrdisplay ();
dec_disp ( TI );
espera ();
do {
   clrdisplay ();
   display ('T',0);
   display ('D',0);
   TD = tecl_dec ();
} while ( TD == -1 );
clrdisplay ();
dec_disp ( TD );
espera ();

/* Inicializacao de variaveis */
disp = 'P';
T  = T_INI;
I  = ZERO_R;
ID = ZERO_R;
D  = ZERO_R;
```

```
TF = TD;
splim = 0;
count = 0;
flag = 0;
pid = 1;

/* Rotina PID */
while ( pid == 1 )
{
   TH0=TL0=0;
   PV = read_temp ();
   E  = SP-PV;
   P  = G*E;
   I  = I+E*T/TI;
   if ( I < LIM_INF ) I = LIM_INF;
   if ( I > LIM_SUP ) I = LIM_SUP;
   ID = ID+D*T;
   if ( ID < LIM_INF ) ID = LIM_INF;
   if ( ID > LIM_SUP ) ID = LIM_SUP;
   D  = (E*TD-ID)/TF;
   VC = P+I+D;
   if ( VC < LIM_INF ) VC = LIM_INF;
   if ( VC > LIM_SUP ) VC = LIM_SUP;
   VCi = (unsigned int) VC;
   if ( flag == 0 )
   {
      flag = 1;
      th1 = TH1;
      tl1 = TL1;
      t1a = _256*th1 + tl1;
   }
   th1 = TH1;
   tl1 = TL1;
   t1b = _256*th1 + tl1;
   if ( t1b >= t1a ) t1r = t1b-t1a;
   else t1r = INT_MAX-t1a+t1b+_2;
   if ( t1r <= VCi && E > ZERO_R )
   {
      zero_cross ();
      PORT3.HEAT = 1;
   }
   else
   {
      flag = 0;
      PORT3.HEAT = 0;
   }
   if (PORT1.IRQ == 1)
   {
      tecla = teclado ();
      if (tecla == 0) pid = 0;
      if (tecla == 5) disp = 'S';
      if (tecla == 6) disp = 'P';
      if ((tecla == 2 || tecla == 8) && disp == 'S')
      {
         if (tecla == 2) SP = SP + 1;
         if (tecla == 8) SP = SP - 1;
         splim = 0;
```

```
            }
         }
         if ( PV == SP ) splim = 1;
         if ( splim == 1 )
         {
            if ( PV <= (SP-LIM_SP))
            {
               PORT1.LED3 = 1;
               PORT3.LED6 = 1;
            }
            if ( PV >= (SP+LIM_SP))
            {
               PORT1.LED2 = 1;
               PORT3.LED6 = 1;
            }
         }
         count++;
         if ( count == DEZ )
         {
            clrdisplay ();
            if (disp == 'S') dec_disp ( SP );
            else dec_disp ( PV );
            count = 0;
         }
         PVi = (unsigned int) PV;
         if ( PVi >= TEMP_MAX )
         {
            PORT3.LED6 = 1;
            PORT1.LED1 = 1;
            pid = 0;
         }
         th0 = TH0;
         tl0 = TL0;
         /* Periodo de Amostragem */
         T = (float)(th0*_256+tl0)*_12*TCLK;
      }
      PORT3.HEAT = 0;
   }
}
```

REFERÊNCIAS E BIBLIOGRAFIA

Referências

[1] **Andrade, M. L., Magalhães, L. P. e Tozzi, C. L.** - Micro-Processadores em Controle de Processo, 4º Congresso Brasileiro de Automática. Campinas, Mini-Cursos:73-102, 1982.

[2] **Intel** - MCS-51 Family of Single-Chip Microcomputers User's Manual. Santa Clara, CA, Jul.,1981.

[3] **Intel** - 8-Bit Embedded Controller Handbook. Santa Clara, CA, 1989, Cap:5, 6, 9, 10.

[4] **Silva Jr., V. P.** - Microcontrolador 8051 Hardware & Software (Projeto Real). São Paulo, Érica, 1990, 143p.

[5] **Intel** - Memory Components Handbook. Santa Clara, CA, 1984, p:4.31-4.49.

[6] **Intel** - Component Data Catalog. Santa Clara, CA, Jan., 1981, p:6.30-6.44.

[7] **Intel** - Peripheral Components. Santa Clara, CA, 1991, p: 3.215-3.230.

[8] **Texas Instruments** - TTL Logic, Standard TTL, Schottky, Low Power Schottky Databook. Dallas, Texas, Mar., 1988.

[9] **Texas Instruments** - The Power Semiconductor Databook for Design Engineers. 1ª ed., Dallas, Texas, s.d., p:7.61-7.64.

[10] **National Semiconductor** - Linear Databook. Santa Clara, CA, 1980.

[11] **Avocet Systems** - AvCase User's Guide. U.S.A., 1989.

Bibliografia

[12] **Gibson, G. A. e Yu-cheng, L.** - Microcomputer Systems: The 8086/8088 Family. Architecture, Programming and Design. Englewood Cliffs, N.J., Prentice-Hall, 1984, Cap:3, 4, p:53-206.

[13] **Intel** - SDK-85 System Design Kit User's Manual. Santa Clara, CA, 1978.

[14] **Intel** - MCS-80/85 Family User's Manual. Santa Clara, CA, 1979, p:357-389.

[15] **Avocet Systems** - AVMAC 8051 Family User's Manual Rev. 1.1. U.S.A., May, 1986, 80p.

[16] **Avocet Systems** - AVSIM 8051 User's Manual - Ver. 1.1. U.S.A., May, 1986. 81p.

[17] **Fröhr, F. e Orttenburger, F.** - Técnicas de Controle Eletrônico, Siemens S.A., São Paulo, Nobel, 1990.

[18] **Ogata, K.** - Engenharia de Controle Moderno, N.J., Prentice Hall do Brasil, Rio de Janeiro, 1982. 929p.