Campus de São Carlos

**DESENVOLVIMENTO DE ARQUITETURA DE INSTRUMENTAÇÃO
INTELIGENTE PARA O MONITORAMENTO DE SILOS AGRÍCOLAS**

Autor: Wagner Roberto Balsani

Orientador: Prof. Dr. Paulo Estevão Cruvinel

UNIVERSIDADE DE SÃO PAULO

**ESCOLA DE ENGENHARIA
DE SÃO CARLOS**

também 07/06/99

defina 15.03.99

DESENVOLVIMENTO DE ARQUITETURA DE INSTRUMENTAÇÃO INTELIGENTE PARA O MONITORAMENTO DE SILOS AGRÍCOLAS

Wagner Roberto Balsani

Dissertação apresentada à Escola
de Engenharia de São Carlos da
Universidade de São Paulo para
obtenção do título de Mestre em
Engenharia Elétrica.

ORIENTADOR: Prof. Dr. Paulo Estevão Cruvinel

**São Carlos
1999**

Class.	TESE
Curt.	0570
Tombo	0115/99

S/S 1034886

Ficha catalográfica preparada pela Seção de Tratamento da
Informação do Serviço de Biblioteca - EESC/USP

B196d Balsani, Wagner Roberto
 Desenvolvimento de arquitetura de instrumentação
 inteligente para o monitoramento de silos agrícolas /
 Wagner Roberto Balsani. -- São Carlos, 1999.

 Dissertação (Mestrado) -- Escola de Engenharia de
 São Carlos-Universidade de São Paulo, 1999.
 Área: Engenharia Elétrica.
 Orientador: Prof. Dr. Paulo Estevão Cruvinel.

 1. Microcontroladores. 2. Instrumentação
 inteligente. 3. Protocolos de comunicação. I. Título.

FOLHA DE APROVAÇÃO

Candidato: Engenheiro **WAGNER ROBERTO BALSANI**

Dissertação defendida e aprovada em 15-03-1999
pela Comissão Julgadora:

Doutor **PAULO ESTEVÃO CRUVINEL (Orientador)**
(Pesquisador - EMBRAPA)

Doutor **ANDRÉ TORRE NETO**
(Pesquisador - EMBRAPA)

Prof. Doutor **RUY BARBOZA**
(Escola de Engenharia de São Carlos - Universidade de São Paulo)

Prof. Doutor **DENIS VINICIUS COURY**
Coordenador da Área de Engenharia Elétrica

JOSÉ CARLOS A. CINTRA
Presidente da Comissão de Pós-Graduação da EESC

*Dedico este trabalho aos meus pais
A quem sou eternamente grato.*

Agradecimentos

Ao Prof. Dr. Paulo Estevão Cruvinel pela orientação em todos os momentos.

Aos professores, funcionários e colegas do Departamento de Engenharia Elétrica da Escola de Engenharia de São Carlos.

Ao pesquisador Dr. André Torre Neto, aos técnicos Gilmar Victorino, José Ferrazini Júnior e a todos os funcionários da EMBRAPA Instrumentação Agropecuária.

Àos meus pais Luiz e Mercedes.

À Rose, Rosana e Antonio Carlos.

À minha namorada Adriana, pelo incentivo e compreensão.

Ao CNPq pelo apoio financeiro.

A DEUS.

SUMÁRIO

CAPÍTULO 1

1.1 Introdução	2
1.2 O processo de migração de umidade e condições térmicas em silos e um dos principais modelos tratados na literatura	4
1.3 Modelo matemático de Khankari	6
1.3.1 Equação de balanço de energia	6
1.3.2 Equação de balanço de umidade	7
1.3.3 O papel das isotermas de absorção	10
1.4 Uso de método numérico	12
1.4.1 Discretização da equação de energia	12
1.4.2 Discretização da equação de umidade	13
1.4.3 Tratamento das condições de fronteira	15
1.5 Exemplos de aplicação de modelagem de silos	15
1.6 Objetivos do trabalho	21

CAPÍTULO 2

2.1 Introdução	24
2.2 Instrumentação inteligente	25
2.2.1 Considerações Básicas	25
2.2.2 Vantagens de se Implementar Inteligência em Instrumentação	27
2.2.3 Opções de Implementação de Inteligência	28
2.2.3.1 Microcontroladores	28
2.2.3.2 Computadores de Placa Única (SBC's)	29
2.2.3.3 Sistemas de Computador Compatíveis com PC	29
2.2.4 Facilidades de Desenvolvimento	30
2.3 O Microcontrolador 80535	32
2.3.1 Introdução	32
2.3.2 Hardware do 80535	32

2.3.2.1 CPU	32
2.3.2.2 Organização da Memória.	34
2.3.2.2.1 Memória de Programa	34
2.3.2.2.2 Memória de Dados	35
2.3.2.3 Portas de I/O	40
2.3.2.4 Temporizadores / Contadores	40
2.3.2.4.1 O Temporizador 0 e o Temporizador 1.	40
2.3.2.4.2 O Temporizador 2	41
2.3.2.5 O Temporizador <i>Watchdog</i>	44
2.3.2.6 A Porta Serial	44
2.3.2.6.1 Gerando Taxas de Transmissão	45
2.3.2.7 O Conversor Analógico Digital	46
2.3.2.8 Estrutura de Interrupções	47
2.3.2.8.1 Estrutura de Níveis de Prioridade	49
2.3.2.8.2 Como as Interrupções são Manipuladas	50
2.3.3 O Software do 80535	52
2.3.3.1 Modos de Endereçamento	52
2.3.3.1.1 Endereçamento Direto	52
2.3.3.1.2 Endereçamento Indireto	52
2.3.3.1.3 Instruções de Registrador	52
2.3.3.1.4 Instruções Específicas de Registrador	52
2.3.3.1.5 Constantes Imediatas	53
2.3.3.1.6 Endereçamento Indexado	53
2.4 Protocolos de Comunicação	54
2.4.1 Transmissão de Dados	54
2.4.2 Protocolos de Comunicação	54
2.4.2.1 Tarefas Funcionais de um Protocolo	65
2.4.3 Métodos de Transmissão	56
2.4.3.1 Enlace Físico	58
2.4.3.1.1 Meios de Interconexão	58
2.4.3.1.2 Topologia de Redes	58
2.4.3.2 Mecanismos de Transporte	60
2.4.3.3 Dados de Aplicação	61
2.4.4 Exemplo de Protocolo de Comunicação (Torre-Neto, 1995)	63

CAPÍTULO 3

3.1 Introdução	65
3.2 A Arquitetura Proposta para o Monitoramento de Silos Agrícolas	66
3.2.1 Descrição dos Módulos	66
3.2.1.1 Os Dispositivos de Memória	70
3.2.1.2 O Latch de Endereços	70
3.2.1.3 O Conversor Digital / Analógico	70
3.2.1.4 O Circuito de Interface Serial	70
3.2.1.5 O Display	72
3.2.1.6 O Teclado	72
3.2.1.7 Os Sensores de Temperatura	72
3.2.1.8 A Fonte de Alimentação	74
3.3 O Protocolo de Comunicação	75
3.3.1 O <i>Frame</i> de Informação	75
3.3.2 Descrição dos Campos do <i>Frame</i>	76
3.3.2.1 Campo Início de Quadro (IQ)	76
3.3.2.2 Campo <i>Enquiry</i> (EN)	76
3.3.2.3 Campos Endereço de Origem (EO) e Endereço de Destino (ED)	76
3.3.2.4 Campo Comando (CM)	76
3.3.2.5 Campo Contador de <i>Bytes</i> de Dados (CB)	76
3.3.2.6 Campo de Dados (DD)	77
3.3.2.7 Campo <i>Check-Sum</i> (CS)	77
3.3.2.8 Campo Fim de Quadro (FQ)	77
3.3.3 Os Comandos Implementados	77
3.3.3.1 O Comando “ H ”	77
3.3.3.2 O Comando “ K ”	77
3.3.3.3 O Comando “ T ”	78
3.3.3.4 O Comando “ S ”	78
3.3.3.5 Os Comandos “ A ”, “ B ” e “ C ”	79

CAPÍTULO 4

4.1 Introdução	81
4.2 O funcionamento do sistema	82
4.2.1 Procedimento de operação	82
4.2.2 Tarefas dos módulos de aquisição	82
4.2.3 Tarefas do módulo hospedeiro	83
4.2.4 Funções via teclado	83
4.3 Rotinas dos sistema	85
4.3.1 Rotinas do módulo hospedeiro	85
4.3.1.1 Rotina de inicialização do módulo hospedeiro	85
4.3.1.2 Rotina de interrupção de contadores (relógio) do módulo hospedeiro	86
4.3.1.3 Rotina de interrupção externa (teclado) do módulo hospedeiro	87
4.3.1.4 Rotina de comunicação serial do módulo hospedeiro	88
4.3.2 Rotinas dos módulos de aquisição	89
4.3.2.1 Rotina de inicialização dos módulos de aquisição	89
4.3.2.2 Rotina de interrupção de contadores (relógio) dos módulos de aquisição	89
4.3.2.3 Rotina de aquisição de dados dos módulos de aquisição	90
4.3.2.4 Rotina de comunicação serial dos módulos de aquisição	91
4.4 Diagramas de estado resultantes na concepção sistêmica do projeto	92
4.5 Desempenho do circuito do sensor de temperatura	95
4.6 Detalhamento do arquivo de dados armazenado no <i>host</i> e IBM PC compatível	96
4.7 Conclusões	100
 Apêndice A - Série de Instruções do Microcontrolador 80535	 102
Apêndice B – Códigos fonte das rotinas do módulo hospedeiro	108
Apêndice C - Códigos fonte das rotinas do módulos de aquisição	151
 Referências Bibliográficas	 175
Glossário	181

LISTA DE FIGURAS

CAPÍTULO 1

Figura 1.1 - Fenômeno da Migração de Umidade	5
Figura 1.2 - Variação da Pressão de Vapor com o Teor de Umidade a Temperatura Constante em Volume de Trigo	11
Figura 1.3 - Variação da Pressão de Vapor com a Temperatura a Teor de Umidade Constante em Volume de Trigo	11
Figura 1.4 - Diagrama Esquemático para Solução por Método Numérico	12
Figura 1.5 - Diagrama Esquemático da Distribuição da Grade	16
Figura 1.6 - Perfis de Umidade e de Temperatura após 150 Dias de Simulação de Armazenagem com Teor de Umidade Inicial de 14% e Diferença de Temperatura entre Grãos e o Ambiente de 20°C para condições de : a) Inverno e b) Verão	17
Figura 1.7 - Perfis de Umidade e de Temperatura após 150 Dias de Simulação de Armazenagem com Teor de Umidade Inicial de 14% e Diferença de Temperatura entre Grãos e o Ambiente de 40°C para condições de : a) Inverno e b) Verão	18
Figura 1.8 - Perfis de Umidade e de Temperatura após 150 Dias de Simulação de Armazenagem com Teor de Umidade Inicial de 18% e Diferença de Temperatura entre Grãos e o Ambiente de 20°C para condições de : a) Inverno e b) Verão	19
Figura 1.9 - Perfis de Umidade e de Temperatura após 150 Dias de Simulação de Armazenagem com Teor de Umidade Inicial de 18% e Diferença de Temperatura entre Grãos e o Ambiente de 40°C para condições de : a) Inverno e b) Verão	20
Figura 1.10 - Ilustração de Silos Usados para Armazenamento de Grãos	22
 Capítulo 2	
Figura 2.1 - Arquitetura Convencional	25
Figura 2.2 - Arquitetura Incorporando Unidades Inteligentes	26
Figura 2.3 - Diagrama de Blocos do Microcontrolador 80535	33

Figura 2.4 - Estrutura de Memória do 80535	34
Figura 2.5 - Execução de Memória de Programa Externa	35
Figura 2.6 - (a) Memória de Dados Interna	
(b) Os 128 Bytes Mais Baixos da Memória RAM	36
Figura 2.7 - Mapa de Memória dos Registradores de Função Especial	39
Figura 2.8 - Diagrama de Blocos do Timer 2	42
Figura 2.9 - Diagrama de Blocos do Conversor A/D	46
Figura 2.10 - Diagrama de Temporização de Resposta a Interrupção	51
Figura 2.11 - Modelo de Protocolo OSI da International Standards Organisation (ISO)	56
Figura 2.12 - Modelo de Protocolo Utilizado em Sistemas de Instrumentação e Controle	57
Figura 2.13 - Algumas Topologias Possíveis de Rede Ponto a Ponto: (a) Estrela, (b) Anel,	
(c) Árvore, (d) Completa, (e) Anéis que se interceptam, (f) Irregular	58
Figura 2.14 - Algumas Topologias de Redes por Difusão: (a) Barramento,	
(b) Satélite ou Rádio, (c) Anel	59
Figura 2.15 - Formato de Pacote SDLC (Controle Síncrono de Enlace de Dados)	61
Figura 2.16 - Formato de Mensagem British Standards Institution BS6556 (SAP)	62
Figura 2.17 - Formatos de Mensagem do STP (Southampton Transducer Protocol)	
(a) Dispositivo Mestre e (b) Dispositivo Escravo	62
Figura 2.18 - Formato de Pacotes de Comunicação Definidos no Protocolo	
Projetado por Torre-Neto (1995)	63

Capítulo 3

Figura 3.1 - Arquitetura Proposta para o Sistema de Monitoramento de Silos de	
Armazenagem de Grãos	67
Figura 3.2 - Diagrama de Blocos do Módulo de Aquisição de Dados	68
Figura 3.3 - Diagrama de Blocos do Módulo Host	68
Figura 3.4 - Diagrama Esquemático do Circuito Utilizado para Implementação	
dos Módulos	69
Figura 3.5 - Esquema de Ligação da Interface Serial Padrão EIA RS-485	71
Figura 3.6 - Esquema de Ligação da Interface Serial entre o Módulo Hospedeiro e o PC	71
Figura 3.7 - Diagrama Esquemático para Ligação do <i>Display</i> de Cristal Líquido	72
Figura 3.8 - Diagrama Esquemático para Ligação do Teclado	73
Figura 3.9 - Diagrama Esquemático do Circuito do Sensor de Temperatura	73

Figura 3.10b - Diagrama Esquemático do Circuito da Fonte de Alimentação Utilizada para o Sistema _____	74
Figura 3.11 - Aspecto do <i>Frame</i> do Protocolo de Comunicação Proposto _____	75
Capítulo 4	
Figura 4.1 – Configuração do Teclado _____	84
Figura 4.2 – Diagrama de Estados dos Módulos de Aquisição _____	93
Figura 4.3 – Diagrama de Estados do Módulo de Hospedeiro _____	94
Figura 4.4 – Fotografia do sistema implementado _____	99

LISTA DE TABELAS

CAPÍTULO 2

Tabela 2.1 - Registradores de Função Especial	37
Tabela 2.2 - Pares de Fontes de Interrupção	47
Tabela 2.3 - Relação entre Bits de IP0 e IP1 e Pares de Fonte de Interrupção	49
Tabela 2.4 - Relação entre Valores de Bits de IP0 e IP1 e Níveis de Prioridade	49
Tabela 2.5 - Prioridade para Pedidos de Interrupção Simultâneos de Mesmo Nível de Prioridade	50
Tabela 2.6 - Endereço de Vetor das Fontes de Interrupção	52

CAPÍTULO 4

Tabela 4.1 – <i>Frames</i> Enviados pelo IBM - PC Compatível para Coletar Dados de Aquisições	96
Tabela 4.2 – Exemplo de Arquivo de Aquisições de um dos Módulos	97
Tabela 4.3 – Relatório Resultante do Arquivo de Aquisições de 1 Módulo	98

RESUMO

Neste trabalho descrevemos a arquitetura e um protocolo de transmissão de dados para instrumentação dedicada ao monitoramento de parâmetros climáticos em silos e secadores de grãos. A arquitetura usa processamento distribuído e instrumentação inteligente baseada no microcontrolador 80535. O protocolo é baseado no padrão de comunicação serial multiponto RS 485 e no protocolo ISO.OSI. Atualmente, aproximadamente 20% da produção brasileira de grãos é perdida durante os processos de transporte e armazenamento. O uso de tal sistema permite medições e modelamento de silos e secadores.

Palavras-Chave: Microcontroladores, Instrumentação Inteligente, Protocolos de Comunicação, Sistemas de Aquisição.

ABSTRACT

In this work we describes an architecture and a digital data transmission protocol for an instrumentation dedicated to the monitoring of climatic parameters of grain bins and dryers. The architecture uses distributed processing and intelligent instrumentation based on the 80535 microcontroller. The protocol is based on both the RS 485 multipoint serial communication standard and the ISO.OSI protocol. Presently, approximately 20% of the Brazilian production of grains is lost during the processes of transportation and storing. The use of this system allows measurements and modelling for bins and dryers.

Keywords: Microcontrollers, Intelligent Instrumentation, Communications Protocols.

Capítulo 1

1.1 Introdução

O desenvolvimento da ciência e da tecnologia é um dos principais passos, se não o principal, para que um país em desenvolvimento, como o Brasil, passe para a condição de desenvolvido.

No que diz respeito à agricultura e à agropecuária, que constituem setores de grande importância no país, tal desenvolvimento implica a informatização, mecanização, automação e integração de sistemas e processos. Ferramentas como eletrônica analógica e digital, microeletrônica, computadores e equipamentos mecanizados devem ser aplicadas em grande escala nesse setor, visando uma maior produtividade com menor custo, buscando inclusive minimizar o impacto ambiental (CNPDIA, 1995).

A tendência para as próximas décadas é a aplicação em massa da eletrônica na agricultura, com vistas à agricultura de precisão (Schepers, 1994).

Torre-Neto et al (1997) publicaram um estudo de exemplos de aplicação de instrumentação e automação para a agricultura de precisão, este último um termo bastante recente porém já definido como uma forte tendência para obtenção do aumento da produtividade.

No Brasil, além da importância da aplicação da eletrônica na agricultura durante os processos de produção, também necessita-se de aplicação de novas técnicas no que diz respeito ao armazenamento e ao transporte da produção.

No caso específico da produção de grãos, atualmente cerca de 20% da produção nacional é perdida durante a armazenagem e o transporte. Sabe-se entretanto, que é possível reduzir as perdas, bastando para isso corrigir o sistema de armazenamento e escoamento das safras produzidas. Assim, a melhoria dos sistemas de tratamento e conservação de grãos, como por exemplo no caso dos silos e dos secadores agrícolas, com a aquisição e medida confiável dos dados das variações térmicas, certamente contribuirão para a minimização de grande parte dos problemas de armazenamento com a decisão otimizada no processo de aeração.

A compreensão no processo de secagem de produtos agrícolas em secadores ou em silos requer o monitoramento, dentre outras variáveis, da temperatura do produto, além do que a quantificação desta variável se faz necessária para a validação de modelos matemáticos de

simulação de secagem de grãos.

Para tanto, se faz necessário o modelamento térmico de silos armazenadores de grãos para que se possa, dentre outras coisas, determinar-se pontos de medição de temperatura, para se obter um eficiente monitoramento durante o período de armazenagem, bem como sistemas instrumentais de monitoramento confiáveis que permitam não só medição pontual mas também a avaliação de perfis e gradientes térmicos.

Dois dos principais fatores que governam a qualidade de grãos durante o seu armazenamento são a temperatura e a umidade. Teores de umidade entre 12% e 14% são considerados seguros para o armazenamento de grãos de cereais por um período de até um ano após o processo de secagem dos mesmos (Alagusundaram, 1990).

Variações sazonais, ou seja, de acordo com as estações climáticas ao longo do ano, na temperatura ambiente podem tornar estes índices inseguros devido a migração ou redistribuição de umidade.

O aumento do teor de umidade localizado pode causar o crescimento de fungos e insetos que algumas vezes produzem toxinas que tornam os grãos prejudiciais ao consumo humano. Temperaturas entre 15 °C e 38 °C favorecem o desenvolvimento de insetos e ácaros, sendo que a temperatura ótima para o seu desenvolvimento está em torno dos 30°C .

A natureza qualitativa e quantitativa da migração de umidade depende de alguns fatores, tais como o tipo e a qualidade do grão armazenado, o tamanho do armazém e as condições climáticas.

O planejamento e a gerência do armazenamento requerem predição e entendimento do processo de migração de umidade sob diferentes condições climáticas dos grãos. Tal predição e entendimento podem ser obtidos através da análise de procedimentos experimentais, mas uma alternativa menos dispendiosa e mais abrangente é a predição através de modelos matemáticos.

Os modelos matemáticos também podem ser usados para estudar o comportamento da temperatura no interior de um silo como uma função do tamanho do silo, do material da parede do silo e da localização geográfica do mesmo. O sentido inverso de análise também é possível, no qual através de perfis térmicos desejados determina-se as características físicas do silo.

Vários modelos matemáticos unidimensionais, bidimensionais e também alguns tridimensionais têm sido desenvolvidos, (Muir, 1970; Converse et al., 1973; Yaciuk et al., 1975; Muir et al., 1980; Metzger e Muir, 1983; Longstaff e Banks, 1987; White, 1988; Manbeck e Britton, 1988; Khankari, 1994; Alagusundaram, 1990), sendo que estes últimos

podem prever com bom nível de precisão o efeito do aquecimento de uma região na massa de grão que a circunda. Entretanto observa-se uma lacuna no que tange arquiteturas dedicadas de sistemas de monitoramento térmico de silos de forma a incorporar e realimentar os modelos de previsão e controle disponíveis, com vista a atuação em tempo real.

1.2 O Processo de Migração de Umidade e Condições Térmicas em Silos e um dos Principais Modelos Tratados na Literatura

O fenômeno das isotermas de absorção em conjunto com a natureza higroscópica (capacidade de absorver umidade) dos grãos desempenham papel chave no processo de migração de umidade.

Este processo geralmente é iniciado por gradientes (diferenças) de temperatura que surgem devido às variações sazonais e diárias da temperatura ambiente.

Durante o processo de secagem dos grãos, a temperatura da massa de grãos torna-se relativamente alta e durante o inverno a temperatura próximo às paredes externas dos silos decresce. Este fato proporciona o aparecimento de gradientes de temperatura ao longo dos mesmos. Devido à baixa difusividade térmica do grão, estes gradientes de temperatura permanecem presentes por vários meses.

Estes gradientes de temperatura ocasionam o aparecimento de gradientes de umidade, visto que o vapor d'água se difunde de regiões mais quentes para regiões mais frias, ou seja, a região mais quente perde umidade enquanto que a região mais fria ganha umidade.

Outro fator que contribui para o aparecimento de gradientes de umidade são as correntes de convecção que originam gradientes de densidade do ar.

Portanto o processo de migração de umidade constitui-se do transporte difusivo e convectivo de vapor d'água através de espaços intergranulares do volume do grão.

No caso não higroscópico quase toda umidade migra de uma região quente para uma região fria, deixando para trás uma região seca, com teor de umidade próximo de zero, como mostrado por Eckert e Faghri em 1980 em um experimento com um banco de areia.

No caso de grãos, o fluxo líquido de difusão de vapor d'água é a soma de dois fluxos opostos devido a gradientes de umidade e temperatura, indicando que os grãos têm uma capacidade finita de receber ou fornecer umidade. Portanto, diferente da areia, o grão não

pode exibir uma região com teor de umidade próximo de zero. A figura 1.1 mostra esquematicamente esse processo.

Figura 1.1 - Fenômeno da Migração de Umidade

Os dois principais processos que causam a migração de umidade no armazenamento de grãos são o transporte de vapor d'água nos espaços intergranulares e a interação entre os grãos e o ar que os circunda.

Os processos de transporte de vapor d'água tendem a diminuir os gradientes de vapor d'água e as relações de absorção tendem a manter tais gradientes.

Desta forma, o processo de migração de umidade é dinâmico e ainda, as variações das condições do ambiente (fronteiras) tornam o processo inerentemente instável.

As propriedades de transporte dependentes de temperatura e umidade, o calor latente de evaporação e de condensação do vapor, a natureza não-linear das isotermas de absorção, e a geração de calor e umidade devido a respiração de microorganismos, são fatores que fazem da migração de umidade e transferência de calor problemas casados e não-lineares, não solucionáveis por métodos analíticos.

Um modelo matemático bidimensional foi desenvolvido por Khankari em 1994 bem como realizada sua simulação numérica, para descrever a migração de umidade devida à

difusão. Este modelo foi testado com resultados experimentais e proporcionou a produção de padrões de migração de umidade devido ao processo de difusão.

1.3 Modelo Matemático de Khankari

Para o desenvolvimento do seu modelo, Khankari considerou o sistema quase-monofásico, isto é, existe equilíbrio termodinâmico local entre um grão e o ar que o circunda, o potencial químico em todas as fases é igual, e em escala macroscópica todas as fases dos poros são uniformemente distribuídas.

Khankari ainda considerou o ar inerte no espaço intergranular do volume de grãos e que o transporte de calor e umidade fosse governado principalmente pelas leis de gradiente. Ele não considerou o transporte por convecção e adotou o Sistema Cartesiano de coordenadas.

1.3.1 Equação de Balanço de Energia

No estudo da migração de umidade causada principalmente pela distribuição não uniforme de temperatura na massa de grãos é inevitável que se obtenha a solução da equação de balanço de energia. A equação de energia na forma de conservação de entalpia por unidade de volume (equação (1.1)) destina-se a aplicação de dados de propriedades térmicas de grãos disponíveis em literatura, visto que tais dados tratam o conjunto de grãos mais ar como um volume uniforme, ou seja:

$$(\rho_{\text{bulk}} \cdot C_{\text{bulk}}) \frac{\partial T}{\partial t} = \frac{\partial}{\partial x_j} \left(K_{\text{bulk}} \frac{\partial T}{\partial x_j} \right) + \rho_{\text{bulk}} \cdot h_{fg} \cdot \frac{\partial W_g}{\partial t} \quad (1.1)$$

*Balanço de Energia na Forma de Conservação
de Entalpia por Unidade de Volume de Grão*

onde ρ_{bulk} é a densidade volumétrica do grão (seco) dada na unidade [Kg de grão seco / m³ de grão], C_{bulk} é o calor específico do volume de grão dado em [Joule / Kg . °C], T é a temperatura dada em graus Celsius [°C], t é o tempo dado em segundos [s], K_{bulk} é a condutividade térmica do volume de grão dada em [W / m . °C], h_{fg} é o calor latente de

vaporização/condensação da umidade do grão dado em [J / Kg de H₂O] , W_g é o teor de umidade dado em [Kg de umidade / Kg de grão seco].

1.3.2 Equação de Balanço de Umidade

A equação (1.2) expressa a conservação de umidade total por unidade de volume de grão, ou seja:

$$\frac{\partial}{\partial t} \left[(\varepsilon \cdot \rho_{\text{air}} \cdot W_a) + (\rho_{\text{bulk}} \cdot W_g) \right] = \frac{\partial}{\partial x_j} \left[\frac{D_v \cdot \varepsilon}{\tau} \cdot \frac{\partial}{\partial x_j} (\rho_{\text{air}} \cdot W_a) \right] \quad (1.2)$$

*Conservação de Umidade Total
por Volume de Grão*

onde ε é a porosidade do volume de grão dada em [m³ de ar / m³ de grão], ρ_{air} é a densidade do ar expressa em [Kg / m³], W_a representa o teor de umidade do ar dado em [Kg de umidade / Kg de ar seco], D_v é a difusividade do vapor d'água no ar dada em [m² / s], τ é o fator de tortuosidade do grão.

O termo do lado esquerdo da equação (1.3) representa a taxa de armazenamento da umidade total, enquanto que o termo do lado direito representa o fluxo líquido de difusão de umidade do ar através dos espaços vazios do volume de grãos.

Nesta formulação a difusão direta de umidade através do contato de grão a grão é desprezada.

É sabido que em grãos armazenados, a maioria da umidade total é mantida como umidade do grão (W_g), e uma porção muito pequena é mantida como umidade de ar (W_a).

Por exemplo, em um volume de 1 m³ de milho, com um teor de umidade de 14%, e a uma temperatura de 298 K (25 °C), tem-se 102,2 Kg de umidade do grão e apenas 0,0064 Kg de umidade do ar.

Outro dado importante é que a razão $\rho_{\text{bulk}} / \rho_{\text{air}}$ geralmente fica entre os valores de 500 a 700.

Portanto, baseando-se nas informações acima, pode-se desprezar a umidade do ar quando comparada a umidade do grão (Stewart, 1975).

Assumindo-se que a mistura ar-vapor nas lacunas comporta-se como um gás ideal, a equação (1.2) fica :

$$\rho_{\text{bulk}} \cdot \frac{\partial W_g}{\partial t} = \frac{\partial}{\partial x_j} \left(D_{\text{eff}} \cdot \frac{\partial p_v}{\partial x_j} \right) \quad (1.3)$$

sendo que:

$$D_{\text{eff}} = \frac{D_v \cdot \varepsilon}{\tau \cdot R_v \cdot T_{\text{ab}}} \quad (1.4)$$

onde p_v é a pressão parcial do vapor d'água dada em [Pa], R_v é a constante de gás do vapor d'água expressa em [461,52 J / Kg . °C], T_{ab} é a temperatura absoluta dada em [°C].

Devido ao equilíbrio termodinâmico local a pressão parcial do vapor d'água no grão e no ar a sua volta são assumidas serem iguais.

Assumindo equilíbrio instantâneo :

$$p_v = f(W_g, T) \quad (1.5)$$

Assim, pode-se escrever :

$$\frac{\partial p_v}{\partial x_j} = \left(\frac{\partial p_v}{\partial W_g} \right)_T \cdot \frac{\partial W_g}{\partial x_j} + \left(\frac{\partial p_v}{\partial T} \right)_{W_g} \cdot \frac{\partial T}{\partial x_j} \quad (1.6)$$

Chamando :

$$\left\{ \begin{array}{l} \frac{\partial p_v}{\partial W_g} \Big|_T = \sigma \\ \frac{\partial p_v}{\partial T} \Big|_{W_g} = \omega \end{array} \right. \quad (1.7)$$

e usando-se a equação (1.6), a equação (1.3) pode ser escrita na forma :

$$\rho_{\text{bulk}} \cdot \frac{\partial W_g}{\partial t} = \frac{\partial}{\partial x_j} \left(D_M \cdot \frac{\partial W_g}{\partial x_j} \right) + \frac{\partial}{\partial x_j} \left(D_T \cdot \frac{\partial T}{\partial x_j} \right) \quad (1.8)$$

*Conservação da Umidade do Grão
sobre uma unidade de Volume de Grão*

onde :

$$\begin{cases} D_M = D_{\text{eff}} \cdot \sigma \\ D_T = D_{\text{eff}} \cdot \omega \end{cases} \quad (1.9)$$

Os termos D_M e D_T podem ser considerados difusividades da umidade do grão devido aos gradientes de umidade e aos gradientes de temperatura, respectivamente.

Thorpe em 1980 desenvolveu a seguinte expressão para calcular a difusividade do vapor d'água no ar (D_v):

$$D_v = \frac{9,1 \times 10^{-9} \cdot (T_{\text{ab}})^{2,5}}{(T_{\text{ab}} + 245,18)} \quad (1.10)$$

Brooker et al. em 1974 encontraram os valores de 0,4 [m^3 de ar / m^3 de grão] para a porosidade do volume de grão (ϵ) e 1,53 para o fator de tortuosidade do grão (τ).

Chen e Morey em 1989 concluíram que um bom modelo para isotermas de absorção é dado pela forma de equação de equilíbrio, ou seja:

$$\frac{p_v}{p_s} = 1 - \exp \left[-K_H \cdot (T + C) \cdot (100 \cdot W_g)^N \right] \quad (1.11)$$

*Modelo de Isoterma
de Absorção*

Onde K_H é a constante na equação de Henderson modificada e vale $1,2299 \times 10^{-5}$, C vale 64,346 e N vale 2,5558 (ASAE, 1990b).

A pressão de vapor de saturação (p_s), que é uma função da temperatura, é expressa usando as relações dadas em ASAE, 1990c.

As derivadas parciais σ e ω da equação (1.4) podem ser determinadas pela diferenciação da equação (1.7) com relação a W_g e T respectivamente.

$$\sigma = p_s \cdot \exp \left[-K_H \cdot (T + C) \cdot (100 \cdot W_g)^N \right] \cdot \left[K_H \cdot (T + C) \cdot (100 \cdot W_g)^{N-1} \cdot 100 \cdot N \right] \quad (1.12.a)$$

$$\omega = \frac{p_v}{p_s} \cdot \frac{dp_s}{dT} + p_s \cdot (K_H \cdot M^N) \cdot \left(1 - \frac{p_v}{p_s}\right) \quad (1.12.b)$$

O termo $\frac{dp_s}{dT}$ é determinado da relação de pressão de saturação de vapor (p_s).

Os coeficientes D_M e D_T são dependentes de temperatura e umidade, e desta forma a equação de umidade (equação (1.8)) torna-se não linear e associada à equação de balanço de energia (equação (1.1)).

Na equação (1.7) W_g é escolhida como variável dependente, e p_v é apenas uma variável auxiliar. Esta escolha garante a conservação da umidade do grão (W_g), minimizando o erro, uma vez que W_g é praticamente igual a umidade total, pois a umidade do ar é desprezível quando comparada a umidade do grão (W_g).

1.3.3 O Papel das Isotermas de Absorção

A relação de isoterma de absorção, cujo efeito é incorporado através das difusividades D_M e D_T , é usada para converter a equação de balanço de umidade (equação (1.2)) na forma “padrão” de equação de transporte (equação (1.5)) com uma única variável dependente.

A variação de σ com o teor de umidade e com a temperatura para o trigo, por exemplo é apresentada na figura 1.2, e a variação de ω com o teor de umidade e a temperatura também para o trigo é mostrada na figura 1.3. O aumento da temperatura provoca o aumento de ambos parâmetros. Para teores de umidade acima de 22%, σ torna-se desprezível, enquanto que ω torna-se constante. Portanto, para altos teores de umidade, a migração da umidade será governada apenas pelos gradientes de temperatura. Já para teores de umidade entre 12% e 14%, comumente mantidos durante o período de armazenamento de grãos de cereais, os dois parâmetros σ e ω devem ser considerados. Portanto a migração de umidade é governada pelos gradientes de temperatura e de umidade.

Figura 1.2 - Variação da Pressão de Vapor com Teor de Umidade a Temperatura Constante em Volume de Trigo

Figura 1.3 - Variação da Pressão de Vapor com a Temperatura a Teor de Umidade Constante em Volume de Trigo

1.4 Uso de Método Numérico

As equações de transporte de calor e de umidade são casadas, não lineares, instáveis e envolvem propriedades de transporte variáveis. Patankar (1980, 1991) solucionou estas equações numericamente através de um esquema de volume de controle, através do qual a conservação de umidade, energia e momento são exatamente satisfeitas. Nesse método as formulações matemáticas envolvidas não obscurecem o significado físico de cada termo na equação governante.

Esse método consiste em dividir o domínio de cálculo em volumes de controle que não se sobrepõem tal que haja um volume de controle circundando cada ponto da grade de medida conforme ilustra a figura 1.4. A seguir, cada equação diferencial é discretizada integrando-a sobre cada volume de controle assumindo perfis lineares para variação da variável espacial.

Como resultado, as equações diferenciais são transformadas em equações algébricas simples, chamadas equações de discretização, as quais são resolvidas usando o método de iteração linha-a-linha.

Figura 1.4 - Diagrama Esquemático para Solução por Método Numérico

1.4.1 Discretização da Equação de Energia

A equação de balanço de energia (equação (1.1)) pode ser discretizada da seguinte

forma:

$$a_P T_P = a_E T_E + a_W T_W + a_N T_N + a_S T_S + b \quad (1.13)$$

onde:

$$a_E = \frac{(K_{\text{bulk}})_e \cdot \Delta y}{(\partial x)_e} \quad (1.14.a)$$

$$a_W = \frac{(K_{\text{bulk}})_w \cdot \Delta y}{(\partial x)_w} \quad (1.14.b)$$

$$a_N = \frac{(K_{\text{bulk}})_n \cdot \Delta x}{(\partial y)_n} \quad (1.14.c)$$

$$a_S = \frac{(K_{\text{bulk}})_s \cdot \Delta x}{(\partial y)_s} \quad (1.14.d)$$

$$a_P^0 = \frac{\rho_{\text{bulk}} \cdot C_{\text{bulk}} \cdot \Delta x \cdot \Delta y}{\Delta t} \quad (1.14.e)$$

$$b = \frac{(\rho_{\text{bulk}} \cdot h_{fg}) \cdot (W_g - W_g^0) \cdot \Delta x \cdot \Delta y}{\Delta t} + a_P^0 \cdot T_P^0 \quad (1.14.f)$$

$$a_P = a_E + a_W + a_N + a_S + a_P^0 \quad (1.14.g)$$

No equacionamento anterior e nos que se seguem, os índices n , s , e , w referem-se respectivamente às direções norte, sul, leste e oeste. O índice p refere-se a valores do ponto da grade relativos ao qual os valores estão sendo calculados, e o índice 0 refere-se aos valores iniciais.

1.4.2 Discretização da Equação de Umidade

Similarmente, a equação de balanço de umidade (equação (1.8)) pode ser discretizada da seguinte forma:

$$a_P W_{gP} = a_E W_{gE} + a_W W_{gW} + a_N W_{gN} + a_S W_{gS} + b \quad (1.15)$$

onde:

$$a_E = \frac{(D_M)_e \cdot \Delta y}{(\partial x)_e} \quad (1.16.a)$$

$$a_W = \frac{(D_M)_w \cdot \Delta y}{(\partial x)_w} \quad (1.16.b)$$

$$a_N = \frac{(D_M)_n \cdot \Delta x}{(\partial y)_n} \quad (1.16.c)$$

$$a_S = \frac{(D_M)_s \cdot \Delta x}{(\partial y)_s} \quad (1.16.d)$$

$$a_P^0 = \frac{\rho_{\text{bulk}} \cdot \Delta x \cdot \Delta y}{\Delta t} \quad (1.16.e)$$

$$b = S_E \cdot (T_E - T_P) - S_W \cdot (T_P - T_W) - S_N \cdot (T_N - T_P) - S_S \cdot (T_P - T_S) - a_P^0 \cdot W_{gP}^0 \quad (1.16.f)$$

Na equação (1.16.f), os parâmetros S_E , S_W , S_N , S_S e a_P são dados por:

$$S_E = \frac{(D_T)_e \cdot \Delta y}{(\partial x)_e} \quad (1.17.a)$$

$$S_W = \frac{(D_T)_w \cdot \Delta y}{(\partial x)_w} \quad (1.17.b)$$

$$S_N = \frac{(D_T)_n \cdot \Delta x}{(\partial y)_n} \quad (1.17.c)$$

$$S_S = \frac{(D_T)_s \cdot \Delta x}{(\partial y)_s} \quad (1.17.d)$$

$$a_P = a_E + a_W + a_N + a_S + a_P^0 \quad (1.17.e)$$

1.4.3 Tratamento das Condições de Fronteira

Neste estudo, a migração de umidade foi considerada como sendo uma redistribuição da umidade inicial devido aos gradientes de temperatura, portanto todas as fronteiras do domínio de cálculo são assumidas serem impermeáveis ao fluxo de umidade:

$$\frac{\partial p_v}{\partial x_j} = \left(\sigma \cdot \frac{\partial W_g}{\partial x_j} \right) + \left(\omega \cdot \frac{\partial T}{\partial x_j} \right) = 0 \quad \text{sobre todas as fronteiras} \quad (1.18)$$

Isto é implementado fazendo-se com que a pressão parcial do vapor d'água (p_v) sobre todas as fronteiras seja igual àquela dos pontos da grade adjacentes às fronteiras.

1.5 Exemplos de Aplicação de Modelagem de Silo

Khankari em 1994 realizou modelamentos de um silo cilíndrico com 10 metros de diâmetro por 10 metros de altura. Para isto foi utilizado o sistema de coordenadas cilíndricas bidimensional.

As condições climáticas adotadas foram condições extremas de inverno e de verão. A simulação foi realizada para um período de 150 dias para cada caso. Foram feitas simulações para duas séries de diferencial de temperatura ($0\text{ }^{\circ}\text{C} \rightarrow 20\text{ }^{\circ}\text{C}$ e $-10\text{ }^{\circ}\text{C} \rightarrow 30\text{ }^{\circ}\text{C}$) e para dois teores de umidade inicial (14% e 18%).

A fronteira mais baixa do silo foi considerada como sendo adiabática, enquanto que a fronteira mais alta e as paredes externas do silo foram consideradas isotérmicas a temperatura ambiente constante.

Por simetria, os cálculos foram realizados num perfil de metade do silo, dividido numa grade não uniforme de 22x22 pontos. A figura 1.5 mostra o esquema de distribuição da grade.

As figuras 1.6 a 1.9 mostram distribuições de umidade e temperatura para as várias condições adotadas no modelamento. Os perfis de temperatura no interior do silo nos dará subsídios para a distribuição dos sensores de temperatura dentro dos silos.

Como resultados, Khankari observou que durante o inverno a umidade move-se para as fronteiras lateral e superior do silo, concentrando-se numa faixa estreita próxima a borda. Durante o verão a umidade move-se das bordas superior e lateral do silo para o seu interior.

Ele também observou que aumentando-se a diferença de temperatura de 20 °C para 40 °C, aumenta-se também o movimento de umidade; e que aumentando-se o teor de umidade inicial de 14% para 18%, aumenta-se a magnitude das mudanças.

Na metade inferior do silo a temperatura varia somente na direção radial. Na metade superior do silo, a temperatura varia em ambas direções.

Isto ocorre em consequência de se adotar a fronteira inferior do silo como sendo adiabática.

Figura 1.5 - Diagrama Esquemático de Distribuição da Grade

UMIDADE

TEMPERATURA

(1.6.a)

(1.6.b)

Figura 1.6 - Perfis de Umidade e de Temperatura após 150 Dias de Simulação de Armazenagem com Teor de Umidade Inicial de 14% e Diferença de Temperatura entre Grãos e o Ambiente de 20°C para Condições de: a) Inverno e b) Verão

(1.7.a)

(1.7.b)

Figura 1.7 - Perfis de Umidade e de Temperatura após 150 Dias de Simulação de Armazenagem com Teor de Umidade Inicial de 14% e Diferença de Temperatura entre Grãos e o Ambiente de 40°C para Condições de: a) Inverno e b) Verão

UMIDADE

TEMPERATURA

(1.8.a)

(1.8.b)

Figura 1.8 - Perfis de Umidade e de Temperatura após 150 Dias de Simulação de Armazenagem com Teor de Umidade Inicial de 18% e Diferença de Temperatura entre Grãos e o Ambiente de 20°C para condições de: a) Inverno e b) Verão

(1.9.a)

(1.9.b)

Figura 1.9 - Perfis de Umidade e de Temperatura após 150 Dias de Simulação de Armazenagem com Teor de Umidade Inicial de 18% e Diferença de Temperatura entre Grãos e o Ambiente de 40°C para Condições de: a) Inverno e b) Verão

1.6 Objetivos do Trabalho

O monitoramento da temperatura do grão durante sua armazenagem em silos é também importante na decisão de execução de aeração, quando forem detectadas áreas de aquecimento da massa armazenada. A figura 1.10 ilustra exemplos de silos usados para armazenamento de grãos.

O objetivo final deste estudo é dar fundamentos para o desenvolvimento de uma arquitetura em instrumentação com unidades remotas inteligentes destinadas à aquisição de medidas de temperatura em silos para grãos, o que tem importância no que diz respeito à conservação da qualidade destes grãos durante o período de armazenamento. Uma arquitetura deste tipo possibilitará o monitoramento constante das temperatura no interior do silo, com cortes de perfis através de técnicas de imagens térmicas com o objetivo de se manter a qualidade dos grãos, potencializando a análise em tempo real de modelos matemáticos utilizados para a predição dos processos de migração de umidade e condições térmicas nos mesmos.

Um sistema deste tipo poderá dirigir ações que culminarão na redução efetiva do volume de perdas de produção por motivos de armazenamento inapropriado.

Figura 1.10 - Ilustração de Silos Usados para Armazenamento de Grãos

Capítulo 2

2.1 Introdução

Neste segundo capítulo serão abordados aspectos teóricos e práticos de temas e dispositivos nos quais este trabalho está baseado.

Dentre eles, será discutida a instrumentação inteligente e suas técnicas de implementação, tais como microcomputadores de placa única, sistemas de computadores compatíveis com PC e microcontroladores, com ênfase especial para o microcontrolador 80535, que é empregado neste estudo como fornecedor de inteligência para o sistema de instrumentação.

Por fim serão apresentados alguns aspectos de protocolos de comunicação utilizados em instrumentação e controle.

2.2 Instrumentação Inteligente

2.2.1 Considerações Básicas

Recentemente tem-se obtido controladores e sensores inteligentes através do emprego de computadores em uma única placa, ou com o uso de microcontroladores. Estes dispositivos vêm se tornando cada vez mais baratos e completos em termos das tarefas que devem ser executadas. O objetivo é minimizar a interação destes sensores ou controladores com o computador *host*, fazendo com que eles trabalhem de forma autônoma e independente, reservando para o *host* a tarefa mais nobre da análise de dados e a exibição em tempo-real, deixando tarefas triviais e repetitivas associadas a aquisição e processamento dos dados brutos por conta da inteligência distribuída. A comunicação entre as unidades inteligentes e o *host* é feita através de um protocolo de transmissão de dados e controle pré-definido (Howell, 1990).

Na figura 2.1 tem-se o esquema de um sistema convencional que não emprega inteligência distribuída, no qual um computador é responsável pela geração do estímulo apropriado e pela interpretação de informação de estado e de dados experimentais. Desta forma o computador está sendo usado de forma ineficiente, realizando operações triviais em intervalos freqüentes e o computador geralmente não está disponível para outras tarefas desejadas como análise e manipulação de dados.

Figura 2.1 - Arquitetura Convencional

Na figura 2.2 vemos um sistema que incorpora unidades inteligentes entre o computador *host* e os sensores ou controladores, permitindo gerenciamento ou interpretação

de uma ou mais interfaces, aliviando o *host* de tarefas repetitivas e triviais.

Figura 2.2 - Arquitetura Incorporando Unidades Inteligentes

Visto que várias unidades inteligentes possam ser requeridas num único sistema, elas devem realmente ser bem estruturadas e de custos compensáveis, visto que cada uma delas pode ser associada a somente um número limitado de sensores ou controladores. Outra importância é a facilidade de desenvolvimento do *hardware* e do *software* de aplicação constituintes das unidades inteligentes, podendo ser modulares e dedicados a tarefas específicas, repetitivas e mesmo triviais.

O advento dos instrumentos inteligentes com capacidades grandemente ampliadas em relação a instrumentação convencional levou o usuário a esperar por mais dados além de simples leituras, como por exemplo condições de alarme e informações de diagnóstico. Linearização de transdutores e calibração automática têm sido práticas comuns em instrumentos microprocessados ou microcontrolados. Níveis de calibração podem ser indicados através de entradas do instrumento independentes do canal de comunicação (Atkinson, 1987).

Os métodos de aplicação de sinais de calibração são diversos, entre eles, magnético, óptico, ultra-sônico.

Portanto, um instrumento chamado inteligente apresenta, se não todas, algumas das características relacionadas a seguir:

- I. compensação de sensibilidade;
- II. auto-linearização;
- III. auto-calibração;
- IV. auto-diagnóstico;
- V. informação de estado;
- VI. *display* de leitura/estado do instrumento;
- VII. capacidades de transmissão remota (Atkinson, 1987).

2.2.2 Vantagens de se Implementar Inteligência em Instrumentação

Dentre as vantagens do uso de inteligência local em instrumentação tem-se:

- liberação do *host* para tarefas nobres como exibição de dados em tempo-real e análise de dados complexos.
- controle do *hardware* local com fornecimento de informação de estado e diagnóstico através de auto-checagem que é de muita importância em sistemas complexos de instrumentação.
- medição e correção automática de defeitos relativos aos sensores, como *offsets*, *drifts* de tempo e de temperatura, como também não linearidades. Possibilidade de auto-verificação periódica através de constantes armazenadas em memória local.
- compilação de sumário dos dados adquiridos e transmissão do mesmo de forma conveniente (fatores de correção, conversão para unidades apropriadas já realizada na inteligência local).
- reconfiguração dinâmica do instrumento, realizada através do *host*, possibilitando uma instrumentação que se adapte a diferentes situações.
- possibilidade de criação de grandes sistemas, com grande número de unidades inteligentes que possam se comunicar entre si ou com o *host*, uma vez que o desempenho deste último praticamente independe do número de unidades. Para isto, pode-se utilizar padrões de comunicação serial populares como o RS232C ou o RS485, os quais possibilitam grande facilidade e baixo custo de instalação, admitindo distâncias consideráveis entre uma unidade e o *host*, muito importante em aplicações industriais e de aquisição de dados em ambientes hostis, ou ainda em ambientes agropecuários.
- a versatilidade da instrumentação inteligente traz vantagens significantes para aplicações industriais, como verificação independente da operação do instrumento e emissão de condições de alarme, aumentando a confiabilidade e durabilidade da instrumentação (Howell, 1990).

Entretanto, para se incorporar inteligência local num instrumento ou sistema, deve-se ter convicção de que a complexidade adicional é justificada. Dificuldades surgem no que diz respeito a projeto, construção, detecção e correção de falhas, manutenção de sistemas inteligentes, que serão aumentadas pela sofisticação de *hardware* e *software* de controle, pela gerência do instrumento e pelos protocolos de comunicação.

2.2.3 Opções de Implementação de Inteligência

Existem vários meios de se implementar inteligência, dentre eles sistemas de controle digital baseados em tecnologia SSI, MSI, e LSI, microcomputadores comerciais, computadores de placa única e microcontroladores. Ambos apresentam vantagens e, portanto, deve-se adequá-las a cada aplicação, com compromisso entre custo, facilidades de desenvolvimento, tamanho físico e capacidades: poder de processamento, memória, provisões de I/O e dispositivos de suporte.

2.2.3.1 Microcontroladores

Uma das maiores vantagens de se implementar inteligência através de microcontroladores está na simplicidade de *hardware*, visto que este componente apresenta vários dispositivos típicos num único circuito integrado (por isso é chamado de computador de *chip* único), evitando o árduo processo de desenvolvimento e detecção de erros de *hardware*. Outras vantagens são o seu reduzido tamanho físico e facilidade de desenvolvimento de *software* de aplicação, visto que existem ferramentas de desenvolvimento compreensivo para a maioria das famílias, tornando esta etapa rápida e barata.

Um microcontrolador geralmente possui uma CPU, memória limitada *on-chip* (RAM e ROM), UARTs (transmissores/receptores assíncronos universais) com gerador de taxa de transmissão, suporte para DMA (acesso direto a memória) e interrupções, contadores /temporizadores, I/O digitais, geralmente através de várias portas bidirecionais, programáveis a nível de *bit*, *nibble* e *byte*.

Dentro da maioria das famílias de microcontroladores existem dispositivos melhorados, destinados a aplicações específicas, que além das características acima pode apresentar algumas outras tais como contadores/temporizadores melhorados, I/O paralelas e interfaces digitais de alta velocidade, conversor A/D multiplexado, temporizadores *watchdogs*, saída de pulsos e *drivers* de *display*.

Tudo isto possibilita a construção de sistemas de controle complexos adicionando-se apenas alguns componentes externos ao microcontrolador.

Outra característica existente em alguns poucos microcontroladores é a inclusão de “*software* em silício”, no qual a ROM já vem programada com algum sistema operacional ou interpretador de linguagem de alto-nível.

Microcontroladores têm sido amplamente empregados comercialmente em indústria

automotiva, controle de processos, utensílios domésticos entre outros.

Fabricantes têm anunciado sensores industriais e unidades de condicionamento de sinais baseados em microcontroladores, possibilitando auto-calibragem, auto-escala, linearização, compensação, melhorando a acuracidade e estabilidade do dispositivo.

As vantagens dos microcontroladores sobre os microprocessadores são evidentes, e suas aplicações devem aumentar cada vez mais. Já existem microcontroladores de 8 *bits* ou de 32 *bits* baseados em arquitetura RISC e desempenho de 5 MIPS (milhões de instrução por segundo). Quando conveniente, são empregados microcontroladores de 4 bits cuja operação também é bastante eficiente. Também existem microcontroladores baseados em processadores digitais de sinais, possibilitando o desenvolvimento de sistemas DSP em um único circuito integrado, como por exemplo o DSC320C14 e o TMS320C40 entre outros.

2.2.3.2 Computadores de Placa Única (SBC's)

Os computadores de placa única são muito utilizados em ambiente industrial por fornecer solução rápida e conveniente para muitos problemas. Geralmente constituem-se de uma CPU de 8, 16 ou 32 bits, I/O paralela e serial, contador/temporizador, RAM suficiente para software de aplicação, ROM que pode conter monitores, assembler/debugger e interpretadores de linguagem de alto nível, tudo numa única placa de tamanho pré-definido. Geralmente apresentam possibilidade de melhorar suas capacidades através de cartões de I/O, cartões estes que podem ou não incorporar inteligência local (através de microprocessadores), para execução de determinadas tarefas. Para gerenciar e programar a distribuição de inteligência no sistema foram desenvolvidas linguagens específicas, como por exemplo a linguagem CLIP, que é uma linguagem de controle para periféricos inteligentes.

Entre as vantagens dos SBC's, tem-se disponibilidade comercial, hardware pronto e facilidade de software de desenvolvimento. Entretanto, eles também apresentam desvantagens significantes, evidentes principalmente em aplicações de larga-escala, entre as quais tem-se o alto custo, tanto da placa como do hardware de suporte, como cartões adicionais e fonte de alimentação.

2.2.3.3 Sistemas de Computador Compatíveis com PC

O recente advento de computadores compatíveis com PC baratos tem removido certas restrições à implementação de instrumentos ou sistemas em torno de computadores comerciais dedicados. Tais restrições existiam no que diz respeito a custo e tamanho físico. As vantagens são claras pois além do PC fornecer facilidades para desenvolvimento como teclado, monitor,

disk drivers, a sua utilização também dispensa preocupações com projeto de embalagens e fontes de alimentação. Outra vantagem é a disponibilidade de utilização de qualquer linguagem desejada ou pacotes de software para o programa de aplicação, e após finalizar este programa, ele pode ser diretamente incorporado na memória do PC através de EPROM ou no arquivo AUTOEXEC.BAT. A comunicação com o “host” também é bastante simples, feita através de barramentos seriais disponíveis no PC. Quando o tamanho físico é preocupação principal, já é possível implementar um PC em placa única.

Portanto, a aplicação deste tipo de hardware vem se tornando cada vez mais barata e amplamente disponível.

2.2.4 Facilidades de Desenvolvimento

A facilidade de desenvolvimento é um ponto crucial e a ela deve ser dada grande importância ao se comparar os diferentes métodos de se implementar inteligência, antes de se tomar uma decisão definitiva a respeito do hardware a ser utilizado.

Geralmente o processo de desenvolvimento de um software de aplicação é composto de 3 etapas :

- 1) criação do código fonte, num editor de texto qualquer, para que este seja compilado ou *assembled*, dando origem ao código objeto.
- 2) carregar o código objeto na RAM do sistema (se possível após simulação), executá-lo e corrigi-lo.
- 3) gravar o programa final em EPROM, finalizando-se assim o sistema.

Já existem SBC's e microcontroladores com interpretadores de linguagem de alto nível, mas as facilidades proporcionadas para correção e controle em tempo-real variam muito. Ao se optar por empregar linguagem de alto nível deve-se levar em conta que o tamanho do código gerado é maior que o tamanho de um código gerado por linguagem de baixo nível como Assembly por exemplo. Neste caso, existem Assembler's comercialmente disponíveis que oferecem testes adequados e facilidades de correção (Howell, 1990)

A aplicação de instrumentação inteligente em um sistema complexo de aquisição de dados e controle geralmente resulta em significativa melhoria no desempenho do sistema, fornecendo também facilidades que dificilmente ou nunca se alcançaria usando técnicas convencionais. Existem vários métodos de se implementar tal inteligência, como vimos anteriormente, cuja escolha apropriada depende principalmente da aplicação a qual o sistema

se destina. Com os benefícios da incorporação de inteligência dedicada tornando-se cada vez mais evidentes, o interesse neste tipo de implementação irá aumentar cada vez mais.

2.3 O Microcontrolador 80535

2.3.1 Introdução

O microcontrolador 80535 é de 8 bits e baseia-se na arquitetura do microcontrolador 8051. Ele mantém todas as características operacionais do 8051, incorporando algumas melhorias que resultam em significativo aumento de flexibilidade de seu uso para projetos dedicados.

As principais características do microcontrolador 80535 são memória RAM de 256 bytes, 6 portas de 8 *bits*, resultando 48 linhas de I/O, 3 temporizadores / contadores de 16 *bits*, sendo um desses temporizadores com capacidade de recarga, captura, e comparação, porta serial *full-duplex*, 12 fontes de interrupção com 4 níveis de prioridade, conversor analógico/digital de 8 *bits*, temporizador *watchdog* de 16 *bits*, tensão para 40 *bytes* de RAM no caso de falha na alimentação, processador booleano, 256 locações endereçáveis por *bit*, maioria das instruções executadas em 1 μ s, capacidade de endereçamento de espaço de memória de programa de 64K e capacidade de endereçamento de espaço de memória de dados também de 64K.

A figura 2.3 mostra o diagrama de blocos do 80535.

2.3.2 O Hardware do 80535

2.3.2.1 CPU

O 80535 é eficiente tanto como um controlador como um processador aritmético. Ele apresenta grandes facilidades para aritmética binária e em código BCD e também para manipulação de *bits*. A sua série de instruções é composta por 44% de instruções de um *byte*, 41% de instruções de dois *bytes* e 15 % de instruções de três *bytes*, possibilita o uso eficiente de sua Memória de Programa (a versão 80515 apresenta uma ROM interna de 8K). A maioria de suas instruções são executadas num único ciclo de máquina. Com um cristal de 12 MHz, 58% das instruções são executadas em 1 μ s.

Figura 2.3 - Diagrama de Blocos do Microcontrolador 80535

2.3.2.2 Organização da Memória

O 80535 manipula espaços de endereço de memória de acordo com a figura 2.4.

Figura 2.4 - Estrutura de Memória do 80535

2.3.2.2.1 Memória de Programa

O 80535 não tem ROM interna, de modo que o pino $\overline{\text{EA}}$ deve ser mantido em nível baixo ao se usar este dispositivo. Isto deve ser feito através de conexão externa de $\overline{\text{EA}}$ a V_{SS} . O sinal de leitura $\overline{\text{PSEN}}$ é usado para todas as buscas de programa externo. A configuração de *hardware* para execução de programa externo é mostrada na figura 2.5. As Portas 0 e 2 são usadas como linhas de I/O dedicadas a função de barramento. A Porta 0 funciona como um barramento de Endereço/Dados multiplexado. Ela emite o *byte* menos significativo do Contador de Programa como um endereço, e depois flutua esperando a chegada do *byte* de código da Memória de Programa. Enquanto o *byte* de endereço está presente na Portas 0 o sinal ALE (Habilitação de *Latch* de Endereço) faz com que este *byte* seja escrito no *latch* de

endereço. A Porta 2 emite o *byte* mais significativo do Contador de Programa. O sinal PSEN faz com que o *byte* de código seja lido pelo microcontrolador através da Porta 0.

Como o 80535 não possui Memória de Programa Interna, as Portas 0 e 2 ficam com o seu uso restrito à leitura de Memória de Programa Externa.

Figura 2.5 - Execução de Memória de Programa Externa

2.3.2.2.2 Memória de Dados

O espaço de Memória de Dados consiste de um espaço de memória interno e outro externo.

A Memória de Dados Externa pode ser expandida até 64K e pode ser acessada por instruções que usam um endereço de 8 ou 16 *bits*.

A Memória de Dados interna está dividida em três blocos fisicamente separados e distintos:

- os 128 *bytes* mais baixos de RAM;
- os 128 *bytes* mais altos de RAM;
- e a área de 128 *bytes* de registradores de funções especiais (SFR).

A figura 2.6 ilustra o mapeamento da memória de dados interna.

**Figura 2.6 - (a) Memória de Dados Interna
(b) Os 128 bytes mais Baixos da RAM Interna**

Uma vez que os 128 *bytes* mais altos de Memória de Dados e a área de SFR compartilham as mesmas locações de endereço, elas somente são acessadas através de diferentes modos de endereçamento. Os 128 *bytes* mais baixos de Memória de Dados pode ser acessado por endereçamento direto ou indireto. Os 128 *bytes* mais altos de RAM podem ser acessados através de endereçamento indireto via registrador, e os registradores de funções especiais são acessados somente por endereçamento direto.

Quatro bancos de 8 registradores cada um ocupam as locações de 0 a 1Fh na área mais baixa da RAM. Os próximos 16 *bytes*, de 20h a 2Fh, contém 128 locações de *bits* diretamente acessíveis. O Apontador de Pilha pode ser alocado em qualquer lugar no espaço de endereço da Memória de Dados Interna, e sua profundidade pode chegar a 256 *bytes*.

Todos os registradores, exceto o Contador de Programa e os quatro bancos de 8 registradores, residem na área de registradores de funções especiais. Os 41 registradores de funções especiais incluem registradores aritméticos, apontadores, e registradores que fornecem uma interface entre a CPU e as funções periféricas contidas no circuito integrado. Também há 128 *bits* diretamente endereçáveis na área de SFR. Os registradores de funções especiais estão listados na tabela 2.1 e o seu mapeamento na Memória de Dados está ilustrado na figura 2.7.

Tabela 2.1 - Registradores de Função Especial (Parte)

Endereço (HEX)	Símbolo	Nome	Valor após Power-On/Reset
* 80	P0	Porta 0	11111111
81	SP	Stack Pointer	00000111
82	DPL	Apontador de Dados, Byte BAIXO	00000000
83	DPH	Apontador de Dados, Byte Alto	00000000
87	PCON	Registrador de Controle de Potência	0xxxxxxx
* 88	TCON	Registrador de Controle de Timer	00000000
89	TMOD	Registrador de Modo de Timer	00000000
8A	TL0	Timer 0, Byte BAIXO	00000000
8B	TL1	Timer 1, Byte BAIXO	00000000
8C	TH0	Timer 0, Byte ALTO	00000000
8D	TH1	Timer 1, Byte ALTO	00000000
* 90	P1	Porta 1	11111111
* 98	SCON	Registrador de Controle da Porta Serial	00000000
99	SBUF	Registrador Buffer da Porta Serial	Indeterminado
* 0A0	P2	Porta 2	11111111
* 0A8	IEN0	Registrador 0 de Habilitação de Interrupção	00000000
0A9	IP0	Registrador 0 de Prioridade de Interrupção	00000000
* 0B0	P3	Porta 3	11111111
* 0B8	IEN1	Registrador 1 de Habilitação de Interrupção	00000000
0B9	IP1	Registrador 1 de Prioridade de Interrupção	00000000
* 0C0	IRCON	Reg. de Controle de Pedido de Interrupção	00000000
0C1	CCEN	Reg. de Habilitação. de Comparação/Capture	00000000
0C2	CCL1	Reg. 1 de Comp./Cap., Byte BAIXO	00000000
0C3	CCH1	Reg. 1 de Comp./Cap., Byte ALTO	00000000
0C4	CCL2	Reg. 2 de Comp./Cap., Byte BAIXO	00000000

Tabela 2.1 - Registradores de Função Especial (Cont.)

Endereço (HEX)	Símbolo	Nome	Valor após Power-On/Reset
0C5	CCH2	Reg. 2 de Comp./Cap., Byte ALTO	00000000
0C6	CCL3	Reg. 3 de Comp./Cap., Byte BAIXO	00000000
0C7	CCH3	Reg. 3 de Comp./Cap., Byte ALTO	00000000
* 0C8	T2CON	Reg. de Controle do Timer 2	00000000
0CA	CRCL	Reg. de Comp./Recarga/Cap., Byte BAIXO	00000000
0CB	CRCH	Reg. de Comp./Recarga/Cap., Byte ALTO	00000000
0CC	TL2	Timer 2, Byte BAIXO	00000000
0CD	TH2	Timer 2, Byte ALTO	00000000
* 0D0	PSW	Registrador Palavra de Status de Programa	00000000
* 0D8	ADCON	Reg. de Controle do Conversor A/D	00000000
0D9	ADDAT	Reg. de Dados do Conversor A/D	00000000
0DA	DAPR	Reg. de Programa do Conversor A/D	00000000
* 0E0	ACC	Acumulador	00000000
* 0E8	P4	Porta 4	11111111
* 0F0	B	Registrador B	00000000
* 0F8	P5	Porta 5	11111111

OBS.: Os registradores marcados com asterisco (*) são endereçáveis tanto pelos bits como pelo byte.

O Acumulador

O registrador Acumulador é denominado por ACC. Entretanto, os mnemônicos para instruções específicas do acumulador se referem a ele simplesmente como A.

O Registrador B

É usado durante as operações de multiplicação e divisão, podendo ser usado como registrador de uso geral para as outras instruções.

O Registrador PSW (Palavra de Status de Programa)

Este registrador contém informação de status de programa.

2.3.2.3 Portas de I/O

O 80535 tem seis portas, P0 a P5, de 8 *bits*. As Portas 1 a 5 quando configuradas como entradas tornam-se ALTAS e fornecerão corrente quando tornadas BAIXAS externamente. A Porta 0 flutuará quando configurada como entrada.

A Porta 0 e a Porta 2 são usadas para acessar as memórias de programa e de dados externas, sendo que durante este acesso, a Porta 0 emite o *byte* de baixa ordem (menos significativo) e a Porta 2 emite o *byte* de alta ordem (mais significativo).

2.3.2.4 Temporizadores / Contadores

O 80535 contém três temporizadores/contadores de 16 *bits* os quais são úteis para muitas aplicações.

Na função temporizador o registrador é incrementado todo ciclo de máquina. Ele pode ser visto como um contador de ciclos de máquina.

Na função contador o registrador é incrementado em cada transição de nível lógico de 1 para 0 do pino de entrada externa T0 (para o Timer0) ou T1 (para o Timer1). Nesta função, o reconhecimento de uma transição de nível lógico 1 para nível lógico 0 do sinal externo toma o tempo de dois ciclos de máquina, portanto a taxa de contagem máxima é 1/24 da frequência do oscilador (lembrar que a frequência de operação da máquina é 1/12 da frequência do oscilador).

2.3.2.4.1 O Temporizador 0 e o Temporizador 1

Os Temporizadores 0 e 1 podem ser configurados em quatro diferentes modos de operação:

- Modo 0: temporizador / contador de 8 *bits* com pré-escala de 32:1. A contagem está presente no registrador THx e a pré-escala de 32:1 é implementada através dos cinco primeiros *bits* do registrador TLx. Quando todos os *bits* de THx passam de nível lógico 1 para 0 o *flag* de interrupção de Timer TFX (no registrador TCON) é setado. TRx é um *bit* de controle de Timer no registrador TCON.
- Modo 1: é idêntico ao Modo 0 exceto que é um temporizador / contador de 16 *bits*. THx apresenta os 8 *bits* mais significativos da contagem e TLx apresenta os 8 *bits* menos significativos da contagem.
- Modo 2: configura o registrador contador do *timer* como um contador de 8 *bits*

(TL1) com recarga automática. Quando ocorre *overflow* de TLx o *flag* TFx é setado, e o valor pré-ajustado por *software* de THx é carregado em TLx e THx permanece inalterado.

- **Modo 3:** Neste modo de operação os Temp./Cont. 0 e 1 apresentam funcionamentos distintos. O Temp./Cont. 0 é configurado como um temp./cont. de 8 bits e um temp. de 8 bits, enquanto que o Temp./Cont. 1 neste modo mantém-se inoperante.

2.3.2.4.2 O Temporizador 2

Além de operar como temporizador ou contador, o Timer 2 também apresenta as seguintes características:

- recarga de 16 *bits*.
- comparação de 16 *bits*.
- captura de 16 *bits*.

A figura 2.8 apresenta um diagrama de blocos do circuito do timer 2.

O Timer 2 pode operar como temporizador, contador de evento ou ainda como temporizador gatilhado.

Na função de temporizador, a contagem é incrementada a cada ciclo de máquina ou a cada dois ciclos de máquina, utilizando-se uma pré-escala de 2:1.

Na função de Temporizador gatilhado, o pino de entrada externa T2 (P1.7) funciona como um gatilho para a entrada do timer 2. Se T2=0, o procedimento de contagem é interrompido. Se T2 está alto, a contagem prossegue. Este argumento é útil para medições de largura de pulso.

Na função de contador, o registrador do timer 2 é incrementado em resposta a transição de 1 para 0 no pino de entrada externa T2 (P1.7). Da mesma forma que ocorre nos Timers 0 e 1, a taxa de contagem máxima é 1/24 da frequência do oscilador.

Em qualquer uma das três funções apresentadas, quando ocorre um *overflow* do registrador do timer 2, isto é, quando a contagem passa de todos os *bits* iguais a 1 para todos os *bits* iguais a 0, o *flag* de *overflow* do timer 2 TF2 (*bit* 6 de IRCON) é setado, podendo gerar uma interrupção.

O Modo Recarga

O modo de recarga do Timer 2 apresenta dois modos de funcionamento. No modo 0, quando ocorre um *overflow* no Timer 2, além de setar o *flag* TF2, ele também efetua uma recarga do registrador do Timer 2 (TL2/TH2) com o valor pré-ajustado por *software* presente

ALTO, retornando para nível BAIXO quando o timer 2 sofrer um *overflow*.

No modo 1, a transição do sinal de saída é determinado por *software*, ou seja, o usuário pode selecionar se o sinal faz uma transição de 0 para 1 ou uma transição de 1 para 0 quando o valor do registrador do timer 2 se igualar ao valor comparado. Assim que os valores comparados se igualam, o valor escrito no primeiro *latch* através do *software* é transferido para o *latch* da direita e consequentemente para o pino da porta 1.

Em ambos os modos de operação, o valor presente na porta 1 é modificado no mesmo ciclo de máquina em que ocorre a igualdade dos valores comparados.

Na função comparação, o pino correspondente da porta 1 atua como saída. Portanto, uma instrução de escrita num pino habilitado para função comparação não terá efeito. Já uma instrução de leitura será efetuada normalmente.

O Modo Captura

Existem dois modos diferentes de operação para a função captura. No modo 0, a captura será causada por um evento externo, que pode ser:

- uma transição positiva nos pinos CC1 a CC3. Neste caso a captura do valor do registrador do Timer 2 ocorrerá nos registradores CC1 a CC3 respectivamente.
- uma transição positiva ou negativa no pino CC0 (dependendo se I3FR=1 ou I3FR=0, respectivamente). Neste caso a captura ocorrerá no registrador CRC.

Neste modo de operação, uma transição nas entradas de captura externas CC0 a CC3 causará a mudança de nível lógico de 0 para 1 dos *flags* de pedido de interrupção externa correspondentes IEX3 a IEX6. Se a interrupção correspondente estiver habilitada, um sinal de captura externo fará com que a CPU aponte para a rotina de serviço de interrupção apropriada.

No modo 1 a captura ocorre em resposta a uma instrução MOV para o *byte* de baixa ordem de um registrador de captura, sendo que o valor escrito no registrador é irrelevante. Neste modo não é gerado pedido de interrupção.

Em ambos os modos de operação, o valor capturado é o valor presente no mesmo ciclo de máquina em que ocorre a captura (ciclo de máquina seguinte ao evento que provocou a captura).

Podem ser estabelecidos modos individuais de operação para cada registrador de captura.

2.3.2.5 O Temporizador *Watchdog*

O 80535 possui um sistema para segurança contra falha de *hardware* ou de *software*, através de um temporizador *watchdog*. Se o *software* não zerar este temporizador no mínimo a cada 65532 μ s, um *reset* de *hardware* interno será inicializado. O *watchdog* também previne um mal funcionamento de *hardware* mais longo que 65 ms (com um oscilador de 12 MHz), pois desta forma o *software* também não zeraria o temporizador.

O temporizador *watchdog* é um contador de 16 *bits* que é incrementado a cada ciclo de máquina. Após um *reset* externo , ele é desabilitado e zerado. O contador é inicializado setando-se o *bit* SWDT (bit 6 de IEN1). Após isto, o temporizador não pode ser parado por *software*. Ele somente pode ser zerado primeiramente setando o *bit* WDT (bit 6 de IEN0) e a seguir setando SWDT. O bit WDT será automaticamente zerado durante o terceiro ciclo de máquina após ele ter sido setado. Esta dupla instrução para se zerar o *watchdog* foi implementada para se evitar que ele fosse acidentalmente zerado. Portanto, após inicializado, o temporizador deve ser zerado periodicamente para prevenir um *overflow*.

Se o *software* não zerar o *watchdog*, quando o contador estiver em FFFCh um *reset* interno será inicializado, que demora quatro ciclos de máquina. Este *reset* só difere do *reset* externo pelo fato de que o temporizador *watchdog* não é desabilitado e o *bit* WTDS (status do temporizador *watchdog* - bit 6 de IP0) é setado. Este *bit* permite o *software* verificar que o *reset* foi causado por *overflow* do temporizador *watchdog*.

2.3.2.6 A Porta Serial

A porta serial do 80535 permite a comunicação *full-duplex*, isto é, permite receber e transmitir dados simultaneamente, entre microcontroladores ou entre microcontroladores e dispositivos periféricos. A porta serial pode operar em quatro modos.

- **Modo 0:** Modo Registrador de Deslocamento. Dados seriais entram e saem através de RxD. TxD produz o *clock* de deslocamento. Oito *bits* são transmitidos/recebidos - os primeiros oito *bits* de dados (*byte* menos significativo). A taxa de transmissão é fixada em 1/12 da frequência de oscilação.
- **Modo 1:** Dez *bits* são transmitidos (através de RxD) ou recebidos (através de TxD) - um *start bit* (0), oito *bits* de dados (primeiro o *byte* menos significativo), e um *stop bit* (1). Na recepção, o *stop bit* vai para o *bit* RB8 do registrador de função especial SCON. Neste modo de operação a taxa de transmissão é variável.
- **Modo 2:** Onze *bits* são transmitidos (através de RxD) ou recebidos (através de TxD)

- um *start bit* (0), oito *bits* de dados (primeiro o *byte* menos significativo), um nono *bit* de dados programável, e um *stop bit* (1). Na transmissão, o nono *bit* de dados (TB8 em SCON) pode ser programado com o valor 0 ou 1, podendo por exemplo ser usado como *bit* de paridade. Na recepção, o nono *bit* de dados vai para RB8 e o *stop bit* é ignorado. A taxa de transmissão neste modo de operação é programável para 1/32 ou 1/64 da frequência de oscilação.
- **Modo 3:** Onze *bits* são transmitidos (através de RxD) ou recebidos (através de TxD)
 - um *start bit* (0), oito *bits* de dados (primeiro o *byte* menos significativo), um nono *bit* de dados programável, e um *stop bit* (1). O Modo 3 é semelhante ao Modo 2 em todos os aspectos, exceto a taxa de transmissão, que no Modo 3 é variável.

As taxas de transmissão variáveis podem ser geradas pelo Temporizador 1 ou um gerador de taxa de transmissão interno.

Em todos os quatro modos a transmissão é iniciada por qualquer instrução que usa SBUF como registrador de destino. A recepção é iniciada no Modo 0 pelas condições RI=0 e REN=1. Nos outros modos a recepção é iniciada por um *start bit* chegando se REN=1.

2.3.2.6.1 Gerando Taxas de Transmissão

Porta Serial no Modo 0

A taxa de transmissão no Modo 0 é fixa em :

$$\text{Tx. Transm. Modo0} = \frac{\text{Freq. Osc.}}{12}$$

Porta serial no Modo 1

O Modo 1 apresenta taxa de transmissão variável e é gerada pelo Timer 1.

Para este objetivo, o Timer 1 é usado no Modo 2 (Auto-Recarga). A taxa de transmissão é determinada pelo *overflow* do Timer 1 e pelo valor do *bit* SMOD no registrador PCON, como se segue:

$$\text{Tx. Transm.} = \frac{2^{\text{SMOD}} \times \text{Freq. Osc.}}{32 \times 12 \times [256 - (\text{TH1})]}$$

Na maioria das vezes o usuário conhece o valor da taxa de transmissão e necessita saber o valor de recarga para TH1 a ser programado. A equação para calcular TH1 pode ser escrita como:

$$TH1 = 256 - \frac{2^{SMOD} \times \text{Freq. Osc.}}{384 \times \text{Tx. Transm.}}$$

Porta Serial no Modo 2

Neste modo a taxa de transmissão é fixa em 1/32 ou 1/64 da frequência do oscilador dependendo do valor do bit SMOD no registrador PCON.

SMOD = 1 $\Rightarrow$ Tx. Transm. = 1/32 Freq. Osc.

SMOD = 0 $\Rightarrow$ Tx. Transm. = 1/64 Freq. Osc.

Porta Serial no Modo 3

Neste modo a taxa de transmissão é variável e ajustada exatamente igual ao Modo 1.

2.3.2.7 O Conversor Analógico/Digital

O 80535 é provido de um Conversor A/D de 8 bits com oito canais de entrada analógica multiplexados. Este conversor tem ainda um circuito *sample and hold* e oferece a vantagem de possibilitar programação de tensões de referência por *software*.

A figura 2.9 mostra o diagrama de blocos do conversor A/D.

O registrador de função especial ADCON é usado para seleccionar um dos oito canais de entrada analógica a ser convertida, para especificar única ou contínuas conversões, e checar o *bit* de estado BSY, que indica se uma conversão está sendo realizada ou não.

Figura 2.9 - Diagrama de Blocos do Conversor A/D

O registrador de função especial ADDAT contém o dado digital de 8 *bits* resultante da conversão. O dado permanece em ADDAT até ele ser sobrescrito pelo próximo dado convertido.

O registrador de função especial DAPR é utilizado para programar as tensões internas de referência IV_{AREF} e IV_{AGND} do conversor A/D, sendo cada uma destas tensões programadas por quatro *bits* de DAPR.

2.3.2.8 Estrutura de Interrupções

A estrutura de interrupção do microcontrolador 80535 fornece 12 fontes de interrupção organizadas em 6 pares, e 4 níveis de prioridade. A tabela 2.2 lista os pares de fontes de interrupção do 80535. Cada fonte de interrupção tem sua própria localização de vetor no espaço de endereço de memória de programa de 00h a 06Bh.

Tabela 2.2 - Pares de Fontes de Interrupção

Interrupção Externa 0	-	Interrupção do Conversor A/D
Interrupção do Timer 0	-	Interrupção Externa 2
Interrupção Externa 1	-	Interrupção Externa 3
Interrupção do Timer 1	-	Interrupção Externa 4
Interrupção da Porta Serial	-	Interrupção Externa 5
Interrupção do Timer 2	-	Interrupção Externa 6

As interrupções externas 0 e 1 (INT_0 e INT_1) podem ser ativadas por nível ou por transição negativa de acordo com os *bits* IT0 e IT1 no registrador TCON. Os *flags* que geram estas interrupções são IE0 e IE1 respectivamente. Se a interrupção for ativada por transição, o *flag* que a originou é zerado pelo *hardware* assim que a rotina de interrupção é apontada. Se a interrupção for ativada por nível, então a fonte de pedido de interrupção controla o *flag* de pedido de interrupção.

As interrupções do Timer 1 e do timer 2 são geradas pelos *flags* TF0 e TF1, de acordo com o funcionamento dos mesmos. Estes *flags* são zerados por *hardware* assim que a rotina de interrupção é acessada.

A interrupção da porta serial é gerada pela lógica OU dos *flags* TI e RI. Estes *flags* não

são zerados por *hardware* e após a interrupção, estes *flags* devem ser zerados por *software* na própria rotina de interrupção.

A interrupção do Timer 2 é gerada pela lógica OU dos *flags* TF2 e EXF2. Estes *flags* não são zerados por *hardware* e após a interrupção, estes *flags* devem ser zerados por *software* na própria rotina de interrupção.

Quando uma conversão A/D é iniciada através de uma instrução de escrita no registrador DAPR, uma interrupção será gerada pelo *flag* IADC no registrador IRCON assim que o valor convertido for movido para o registrador ADDAT ao final da conversão. O *flag* de interrupção do conversor A/D IADC deve ser zerado por *software*.

A interrupção externa 2 (INT₂) pode ser ativada por transição negativa ou positiva no pino correspondente, dependendo do *bit* I2FR no registrador T2CON. Esta interrupção é gerada pelo *bit* IEX2 no registrador IRCON. O *bit* IEX2 é zerado por *hardware* quando a rotina de interrupção é acessada.

A interrupção externa 3 (INT₃) também pode ser ativada por transição negativa ou positiva no pino correspondente, dependendo do *bit* I3FR no registrador T2CON. O *flag* que gera esta interrupção é IEX3 no registrador IRCON. Este *bit* também pode ser setado se ocorrer um evento de comparação no pino P1.0/ INT₃/CC₀, isto é, o conteúdo do registrador do timer 2 se igualar ao conteúdo do registrador CRC. O *flag* IEX3 é zerado por *hardware* assim que a rotina de interrupção é acessada.

As interrupções externas 4, 5 e 6 (INT₄, INT₅, e INT₆) são ativadas somente por transição positiva no pino correspondente. Os *flags* que geram estas interrupções são IEX4, IEX5, e IEX6, respectivamente, pertencentes ao registrador IRCON. Estes *flags* também podem ser setados se ocorrerem eventos de comparação nos pinos de saída correspondentes P1.1/ INT₄/CC₁, P1.2/ INT₅/CC₂, e P1.3/ INT₆/CC₃. Estes *flags* também são zerados por *hardware* após a rotina de interrupção ser acessada.

Todos os *bits* que geram interrupções podem ser setados ou zerados por *software* a fim de gerar interrupções ou cancelar aquelas pendentes. Exceção é feita aos *flags* IE0 e IE1, cujas interrupções podem ser ativadas por nível, e neste caso, as interrupções externas 0 e 1 só podem ser geradas por *software* escrevendo um 0 nos pinos INT₀ e INT₁ respectivamente.

Cada uma destas interrupções podem ser habilitadas ou desabilitadas individualmente através do devido ajuste dos registradores IEN0 e IEN1. O registrador IEN0 possui o *bit* EAL, que quando zerado, desabilita todas as interrupções de uma só vez.

2.3.2.8.1 Estrutura de Níveis de Prioridade

Cada par de fontes de interrupção pode ser programado individualmente para um de quatro níveis de prioridade. Esta programação é feita através de *bits* presentes nos registradores IP0 e IP1. A tabela 2.3 relaciona os *bits* com os pares de fontes de interrupção e a tabela 2.4 relaciona os *bits* com os níveis de prioridade.

Tabela 2.3 - Relação entre Bits de IP0 e IP1 e Pares de Fonte de Interrupção

Bits		Par de Interrupção Correspondente
IP1.0	IP0.0	IE0 / IADC
IP1.1	IP0.1	TF0 / IEX0
IP1.2	IP0.2	IE1 / IEX3
IP1.3	IP0.3	TF1 / IEX4
IP1.4	IP0.4	RI+TI / IEX5
IP1.5	IP0.5	TF2+EXF2 / IEX6

Tabela 2.4 - Relação entre Valores de Bits de IP0 e IP1 e Níveis de Prioridade

IP1. x	IP0. x	Nível de Prioridade
0	0	Nível de Prioridade 0 (Mais Baixo)
0	1	Nível de Prioridade 1
1	0	Nível de Prioridade 2
1	1	Nível de Prioridade 3 (Mais Alto)

Uma interrupção pode ser interrompida por outra interrupção de maior nível de prioridade, mas não por uma interrupção de mesmo nível de prioridade ou menor. Se dois ou mais pedidos de interrupção ocorrerem simultaneamente, será primeiramente atendida aquela que tiver maior prioridade. Para o caso de pedidos de interrupção simultâneos de mesmo nível de prioridade, existe um esquema interno, conforme ilustra a tabela 2.5, no qual o nível de

prioridade decresce de cima para baixo, e dentro de cada par de fontes de interrupção o nível de prioridade decresce da esquerda para a direita. Deve-se lembrar que este esquema só é válido para pedidos de interrupção simultâneos de mesmo nível de prioridade.

Tabela 2.5 - Prioridade para Pedidos de Interrupção Simultâneos de Mesmo Nível de Prioridade

ALTA → BAIXA		Prioridade
Par de de Interrupção Fonte		
IE0	IADC	ALTA ↓
TF0	IEX2	
IE1	IEX3	
TF1	IEX4	
RI + TI	IEX5	
TF2 + EXF2	IEX6	BAIXA

2.3.2.8.2 Como as Interrupções São Manipuladas

Os *flags* de interrupção são amostrados em S5P2 de todo ciclo de máquina, e as amostras são votadas durante o ciclo de máquina seguinte, gerando uma instrução LCALL para a rotina de interrupção apropriada se houver algum *flag* setado. Esta instrução será bloqueada se houver qualquer uma das seguintes condições:

- 1) Uma interrupção de igual ou maior prioridade estiver em progresso.
- 2) O ciclo corrente (de votação) não é o ciclo final na execução da instrução em progresso. Isto assegura que a instrução em progresso será completada antes da chamada da rotina de interrupção.
- 3) A instrução em progresso é RETI ou qualquer acesso aos registradores IEN0, IEN1, IP0, ou IP1. Isto garante que, para estes casos, no mínimo mais uma instrução será executada antes da chamada de qualquer rotina de interrupção.

O ciclo de votação / seqüência LCALL é ilustrado na figura 2.10.

Figura 2.10 - Diagrama de Temporização de Resposta a Interrupção

O processador reconhece um pedido de interrupção executando uma instrução LCALL gerada por *hardware* para a rotina de interrupção apropriada. Esta instrução empurra o conteúdo do Contador de Programa para o Apontador de Pilha (mas não salva PSW) e recarrega o Contador de Programa com um endereço que depende da fonte de interrupção, conforme a tabela 2.6.

Tabela 2.6 - Endereço de Vetor das Fontes de Interrupção

Fonte	Endereço de Vetor
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH
IADC	0043H
IEX2	004BH
IEX3	0053H
IEX4	005BH
IEX5	0063H
IEX6	006BH

A execução procede daquela locação até que uma instrução RETI seja encontrada. Então o Processador reconhece que a interrupção terminou e pega os 2 *bytes* do topo do Apontador de Pilha e recarrega-os no Contador de Programa, que faz com que o programa interrompido continue de onde parou.

2.3.3 O Software do 80535

2.3.3.1 Modos de Endereçamento

2.3.3.1.1 Endereçamento Direto

No endereçamento direto o operando é especificado por um campo de endereço de 8 *bits* na instrução. Somente a RAM de Dados interna e os registradores de Função Especial podem ser endereçados diretamente.

Ex.: MOV A, 50h (código E5 50)

2.3.3.1.2 Endereçamento Indireto

No endereçamento indireto a instrução especifica um registrador que contém o endereço do operando. As memórias RAM interna e externa podem ser endereçadas indiretamente.

O registrador de endereço para endereços de 8 *bits* pode ser R0 ou R1 do banco de registrador selecionado, ou o Apontador de Pilha. O registrador de endereço para endereços de 16 *bits* somente pode ser o registrador apontador de dados de 16 *bits* DPTR.

Ex.: MOV @R0, #20h (código 76 20)
MOVX @DPTR, A (código F0)

2.3.3.1.3 Instruções de Registrador

Certas instruções acessam os registradores R0 a R7 do banco de registradores corrente (selecionado através de PSW), através de 3 *bits* presentes no *opcode* da instrução.

Ex.: INC R0 (código 08)
INC R1 (código 09)

2.3.3.1.4 Instruções Específicas de Registrador

Algumas instruções são específicas para um certo registrador, que é apontado no *opcode* da instrução. Por exemplo, algumas instruções operam sempre com o acumulador, ou

com o Apontador de Dados (DPTR), não sendo necessário apontá-los através de *bytes* de endereço.

Ex.: MUL AB (código A4)

2.3.3.1.5 Constantes Imediatas

O valor de uma constante pode seguir o *opcode* na Memória de Programa. Por exemplo,

MOV A, #100 (código 74 64)

carrega o Acumulador com o número decimal 100. O mesmo número poderia ser especificado em hexadecimal como 64h.

2.3.3.1.6 Endereçamento Indexado

Somente a Memória de Programa pode ser acessada com endereçamento indexado. Este modo de endereçamento tem a finalidade de leitura de *look-up tables* na Memória de Programa. Um registrador de 16 *bits* (DPTR ou Contador de Programa) aponta para a base da tabela, e o Acumulador é ajustado com o número de entrada da tabela. O endereço de entrada da tabela na Memória de Programa é formado pela adição do dado do Acumulador ao apontador da base.

Ex.: MOVC A, @A+DPTR (código 93)

2.4 Protocolos de Comunicação

2.4.1 Transmissão de Dados

O grande problema na transmissão de dados é que tem havido uma falta de padronização de bases globais, sendo que diferentes indústrias de diferentes nacionalidades têm adotado diferentes técnicas para transmissão de dados. Para dados analógicos, geralmente se transmite um valor como 4mA a 20mA , 0.4V a 2V , 5Hz a 15Hz, etc. Mas quando se trata de instrumentos inteligentes, o meio de comunicação deve ser capaz de transmitir não apenas valores analógicos mas sim transmitir também um protocolo de comunicação digital.

Os problemas encontrados para isto são os diferentes padrões de comunicação digital atualmente empregados e como melhor representar um dado na forma digital. Portanto, há dois assuntos distintos ao se discutir instrumentação digital. Um deles é o uso de inteligência através de microprocessadores para melhorar as capacidades do instrumento, e o outro constitui-se dos protocolos de comunicação.

A melhor saída é separar a tarefa de gerenciamento da operação interna do instrumento, daquela relacionada às transmissão de dados. Assim, como resultado ter-se-ia um instrumento com a possibilidade de escolha de formatos de transmissão. Para isso, pode-se pensar em usar mais que um microprocessador por instrumento, sendo que um deles se incumbiria do gerenciamento da comunicação. Isto se faz necessário devido à grande complexidade do protocolo de comunicação, e a meta seria a possibilidade de se obter uma gama de instrumentos inteligentes com uma gama de interfaces de comunicação intercambiáveis (Atkinson, 1987).

2.4.2 Protocolo de Comunicação

Um protocolo de comunicação consiste de uma série de regras cuja finalidade é assegurar que a informação a ser trocada entre duas ou mais unidades está sendo enviada e recebida corretamente, de modo que não haja diferença entre os dados enviados e os dados recebidos.

2.4.2.1 Tarefas Funcionais de um Protocolo de Comunicação

Há muitos tipos diferentes de protocolos de comunicação, desde aqueles cuja preocupação é a simples troca de mensagens entre duas partes conectadas por um único enlace, até aqueles cuja finalidade é a comunicação entre computadores conectados numa rede. Mesmo assim, todos eles incorporam os elementos de controle de erro, controle de seqüência, controle de fluxo, controle de intervalo e controle de início. Esses elementos são definidos como:

- **Controle de Erro:** é a ação necessária para que o receptor receba uma cópia correta da informação transmitida quando um erro for detectado.
- **Controle de Seqüência:** a maioria dos sistemas de comunicação envolvem a troca de uma série inteira de mensagens, e não apenas uma simples mensagem, e cada mensagem, por sua vez, pode ser dividida em blocos menores, que precisam estar numa seqüência correta. Portanto, o protocolo deve possuir algum tipo de identificação de seqüência para que os pacotes ou mensagens sejam processados na seqüência correta, no caso de eles serem recebidos numa ordem incorreta.
- **Controle de Fluxo:** se a fonte gera informações mais rápido do que elas possam ser recebidas pelo destinatário, se faz necessário algum meio de controlar a produção ou o fluxo de informação através da rede de comunicação, para que esta não fique congestionada.
- **Controle de Intervalo (*Time-out*):** diz respeito a ação a ser tomada se o fluxo de informação é interrompido. Alguns protocolos determinam que a fonte receba um sinal de reconhecimento vindo do receptor para indicar que o mesmo recebeu corretamente a mensagem enviada pela fonte. Se este sinal demora a chegar, a ação de controle de intervalo faz com que a mensagem seja retransmitida ou a comunicação abandonada. Este tempo de espera varia muito com o nível de complexidade da rede de comunicação.
- **Controle de Início (*Start-up*):** serve para assegurar que as duas ou mais unidades comunicantes estejam aptas a iniciar a comunicação. Esta verificação pode ser feita através de um *handshake* (Willis, 1986).

2.4.3 Métodos de Transmissão

A comunicação entre um instrumento e um sistema central de aquisição de dados pode ser dividida em vários níveis identificáveis, que vão desde o formato dos dados (nível mais alto) até o tipo de fio utilizado para interligar as unidades (nível mais baixo). Entre estes dois extremos estão os métodos pelos quais a transmissão de dados é iniciada, verificada e reconhecida.

A ISO (International Standards Organisation) definiu um modelo padrão para este nivelamento, que é composto de 7 camadas, denominado OSI (Open Systems Interconnection). Este modelo geralmente não é utilizado totalmente, sendo o mais comum encontrar-se apenas algumas destas camadas, servindo apenas como referência. A figura 2.11 mostra o padrão OSI definido pela ISO.

Figura 2.11 - Modelo de Protocolo OSI da International Standards Organisation (ISO).

As camadas são definidas como:

- 1) Camada Física: lida com aspectos físicos, tais como características mecânicas, elétricas, funcionais e procedimentais de acesso ao meio. Por exemplo, níveis de tensão para representar os níveis lógicos 0 e 1, sinais de controle para indicar o estado do circuito físico para poder sincronizar a taxa de dados, propriedades mecânicas dos plugs de conexão, designação dos pinos dos plugs de conexão.
- 2) Camada de Enlace de Dados: é concernente à transferência confiável de informação ao longo do enlace físico. Envia blocos de dados (*frames*) com a sincronização

necessária, controle de erro e controle de fluxo. Visto que a camada física meramente aceita uma sequência de bits sem se importar com o significado ou a estrutura, cabe à camada de enlace de dados criar e reconhecer os limites do *frame*.

- 3) Camada de Rede: fornece às camadas superiores independências da transmissão de dados e tecnologias de chaveamento utilizadas para os sistemas de conexão. Responsável pelo estabelecimento, manutenção e finalização das conexões. Toma a mensagem da fonte, divide-a em pacotes e organiza a transmissão destes pacotes ao longo da rede para o destino desejado. Portanto, é responsável pelo controle de sequência e de fluxo dos pacotes de informação. Esta camada é indispensável para que ocorra a perfeita comunicação entre redes heterogêneas.
- 4) Camada de Transporte: procura ocultar as características de rede dependentes das camadas inferiores, ou seja, sua finalidade é apresentar transferência de dados confiável e transparente.
- 5) Camada de Sessão: sessão é o período de tempo no qual duas unidades permanecem conectadas. Esta camada tem por finalidade o estabelecimento e o gerenciamento do caminho de conexão entre duas unidades.
- 6) Camada de Apresentação: manipula a conversão de códigos, compressão de informação, codificação de dados sigilosos, etc.
- 7) Camada de Aplicação: é o nível mais alto e é o ambiente no qual os programas do usuário operam e se comunicam. Tem por função vencer as incompatibilidades entre terminais e arquivos na transferência da informação.

Em instrumentação, geralmente se encontram modelos de poucas camadas, como o modelo de 3 camadas mostrado na figura 2.12, que separa os dados de aplicação do mecanismo de transporte empregado para efetuar a passagem de dados do instrumento ao sistema de aquisição de dados e vice-versa. A camada mais baixa define o meio e o método de transmissão (fibra óptica, fio de cobre, ondas de rádio).

Figura 2.12 - Modelo de Protocolo Utilizado em Sistemas de Instrumentação e Controle

2.4.3.1 Enlace Físico

2.4.3.1.1 Meios de Interconexão

Devido as suas propriedades de auto-blindagem, o par trançado de fios e o cabo coaxial são os meios de interconexão preferidos. Fibras ópticas têm tido algumas aplicações industriais. Sistemas de transmissão sem fio também tem tido algumas aplicações com bons resultados em indústrias e em ambientes agropecuários, principalmente neste último em consequência das grandes distâncias que devem ser cobertas.

2.4.3.1.2 Topologia de Redes

De maneira geral existem dois tipos de projetos de redes de comunicação, que serão apresentados a seguir.

Canais Ponto a Ponto

Neste caso a comunicação é feita através de um canal que une dois pontos. Se dois dispositivos que não estão interconectados diretamente desejam trocar informações, a comunicação deverá ser feita indiretamente, através de outros dispositivos. Quando uma mensagem é enviada de um ou mais dispositivos intermediários, esta mensagem é recebida inteiramente em cada dispositivo, armazenada até que a linha de saída requerida esteja livre, e então passada adiante. A figura 2.13 apresenta várias topologias possíveis da rede de canais ponto a ponto.

Figura 2.13 – Algumas Topologias Possíveis de Rede Ponto a Ponto: (a) Estrela; (b) Anel; (c) Árvore; (d) Completa; (e) Anéis que se interceptam; (f) Irregular.

Redes locais geralmente apresentam uma topologia simétrica. Em contraste, redes de longa distância têm tipicamente topologias irregulares.

A rede estrela é usada em sistemas analógicos de transmissão e permite a sobreposição das linhas de alimentação do instrumento à linha de dados, o que justifica a redundância na quantidade de fios utilizados em muitas aplicações.

A configuração anel tem sido bastante empregada em automação de escritórios, mas a grande desvantagem no que diz respeito ao seu uso em instrumentação está no fato de que ela conta com que todas as unidades estejam em operação para habilitar a transmissão de dados, ou seja, se uma unidade falhar, a comunicação não é possível. Portanto, esta configuração deixa muito a desejar no que diz respeito à confiabilidade.

Canais de Difusão

Um segundo tipo de arquitetura de comunicação utiliza difusão (*broadcasting*). A maioria das redes locais são desse tipo. Os sistemas por difusão têm um único canal de comunicação, compartilhado por todos os dispositivos conectados a rede. Pacotes enviados de qualquer terminal são recebidos por todos os outros. Um campo de endereço no pacote especifica o destinatário. Ao receber o pacote cada dispositivo verifica o campo de endereço; os dispositivos cujos endereços não coindidem com aquele presente na mensagem simplesmente a ignoram. Os sistemas por difusão também permitem endereçar uma mensagem para todos os terminais através do uso de um código especial no campo de endereçamento. Quando um pacote com este código é transmitido, ele é recebido e processado por todos os terminais. A figura 2.14 apresenta algumas das possibilidades para redes por difusão.

Figura 2.14 – Algumas Topologias Possíveis de Rede por Difusão:
(a) Barramento; (b) Satélite ou Rádio; (c) Anel.

A configuração barramento é o método no qual vários instrumentos compartilham um meio de comunicação comum. Por isso, é necessário efetuar a separação de sinais individuais de cada unidade conectada. Para isto, existem várias técnicas, dentre elas as que separam o sinal em tempo, TDM (Multiplexação por Divisão de Tempo), e as que separam os sinais em frequência, FDM (Multiplexação por Divisão de Frequência), sendo esta última bastante complicada quanto à exigência de *hardware*. A mais utilizada é a TDM, que se utiliza de um protocolo para evitar a colisão de informações de unidades diferentes.

Num sistema por satélite ou por estações de rádio cada terminal tem uma antena através da qual pode transmitir e receber.

Num sistema de difusão em anel, cada bit se propaga por conta própria, sem esperar pelo resto do pacote ao qual pertence. Como todos os outros sistemas por difusão necessita-se de alguma regra para arbitrar acessos simultâneos ao anel.

Na comunicação digital de dados, o método mais adequado é a configuração *multi-drop* ou barramento, usado para implementar grandes trilhas de comunicação de dados, mas a configuração estrela é muito usada atualmente para redes de dados digitais em indústrias (Atkinson, 1987).

2.4.3.2 Mecanismos de Transporte

A maioria das técnicas utilizadas para comunicação entre unidades empregam operações mestre/escravo, nas quais a unidade mestre controla a comunicação. Dentre elas, existem aquelas em que a unidade que desempenha a função de mestre é fixa, e também aquelas em que as unidades podem se revezar na função de mestre. Obviamente o primeiro método é mais simples.

Além de determinar a ordem de transmissão num caminho de transmissão multiplexado, o mecanismo de transporte também é responsável pelo roteamento de informação entre as estações, que implica na adição de informação de endereço em qualquer dado transmitido. Também deve ser incluído algum meio de estabelecer a validade dos dados, geralmente feito através de cálculo e anexação de códigos de checagem. Como consequência de tudo isso, os dados devem ser transmitidos em pacotes denominados *frames*, com *flags* reconhecíveis de início e final de mensagem, para assegurar que dados de aplicação não sejam confundidos com informação de endereço ou código de checagem.

Há muitos padrões e sub-séries e variações destes padrões atualmente utilizados no campo de transmissão de dados.

O SDLC (Controle Síncrono de Enlace de Dados) - IBM 1986, é mostrado na figura

2.15, que é uma técnica de protocolo orientado por *bit* e tem campos de *bytes* de *flag*, de endereço, de controle e de informação. Uma variação do SDLC é o HDLC (Controle de Alto-nível de Enlace de Dados) - ISO 1976, que permite o uso de endereço estendido e de campos de informação com comprimento variável. Apesar de haver uma grande variedade de protocolos híbridos entre o SDLC e o HDLC, que não se classificam como nenhum dos dois, eles têm sido bastante utilizados em indústrias.

Figura 2.15 - Formato de Pacote SDLC

A maior complicação de protocolos orientados por *bit*, é a possibilidade de a sequência de *bits* no campo de informação representar o padrão dos *flags* de início/fim de mensagem. Para evitar esta confusão, emprega-se a técnica de *bit-stuffing* que consiste na inserção de um *bit* extra no campo de informação para quebrar qualquer sequência de dados que pudesse gerar confusão, *bit* este que depois é automaticamente removido.

Geralmente as tarefas de formatação de dados em pacotes, reconhecimento de endereço de estação, verificação de campo de checagem, realização de *bit-stuffing* são feitas por *hardware* VLSI dedicados.

Existem protocolos orientados por *byte*, como o SAP (Protocolo Assíncrono Simples) BS6556 - British Standards Institution 1985, mostrado na figura 2.16, que baseia-se em técnicas análogas a do SDLC e HDLC com a diferença de que trabalha com blocos de 8 bits (um *byte*) e necessita de operação de *byte-stuffing* para evitar confusões entre *bytes* de dados e *bytes* de controle.

Um meio de se evitar a necessidade de utilizar técnicas de *stuffing*, é através da utilização de técnicas de transmissão de informação codificada, por exemplo, em código ASCII, na qual é possível definir um único caracter de início de mensagem que nunca aparecerá no campo de dados. Outra vantagem desta técnica é a de apresentar uma saída legível diretamente no barramento de comunicação. A figura 2.17 mostra o protocolo STP (Southampton Transducer Protocol) - Atkinson 1984.

2.4.3.3 Dados de Aplicação

Ao se mencionar os possíveis formatos dos dados de aplicação encontra-se grande

difficuldade devido a total falta de padronização e a grande quantidade de modelos existentes.

O padrão SAP apresenta uma explanação detalhada em seu documento interno, que especifica não somente o formato de dados analógicos e digitais mas também detalhes do formato de mensagens no que diz respeito ao tipo de transdutor, fator de multiplicação, código de falha e de estado, etc.

Já o protocolo STP não faz distinção entre mecanismo de transporte e dados de aplicação, propiciando maior liberdade na elaboração dos mesmos (Atkinson, 1987).

Figura 2.16 - Formato de Mensagem British Standards Institution BS6556 (SAP)

Figura 2.17 - Formatos de Mensagem do STP: (a) Dispositivo Mestre (b) Dispositivo Escravo

2.4.4 Exemplo de Protocolo de Comunicação

Torre-Neto(1995) desenvolveu um protocolo de comunicação para um sistema de monitoramento de variáveis edafo-ambientais baseado na estrutura OSI (Open Systems Interconnection), com algumas simplificações. O protocolo desenvolvido apresenta relações assimétricas do tipo mestre/escravo operando no modo *poll/response*, isto é, as unidades escravos só transmitem mediante um pedido emitido pela unidade mestre.

Foram implementadas apenas as camadas meio físico, enlace e aplicação.

O meio físico adotado foi o padrão serial multiponto RS-485 via cabo.

Na camada enlace foi especificado um formato de pacotes de comunicação, conforme ilustrado na figura 2.18, além de procedimentos para detecção de falhas e recuperação de informação.

*TDI : Transmissor Digital Inteligente. Nome adotado no trabalho de Torre Neto (1995) para o circuito responsável pela aquisição e transmissão de leituras dos sensores. Constituem as unidades escravos.

Figura 2.18 - Formato de Pacotes de Comunicação Definidos no Protocolo Projetado por Torre Neto (1995)

Na camada aplicação foi projetado um conjunto de comandos para controlar a operação da comunicação, sendo este controle efetuado pela unidade mestre de modo a garantir a correta operação do sistema.

Capítulo 3

3.1 Introdução

Neste capítulo serão abordados aspectos práticos da implementação da arquitetura de instrumentação inteligente proposta para a realização do monitoramento de variáveis térmicas em silos destinados à armazenagem de grãos.

Além de aspectos da arquitetura, também serão abordados aspectos do protocolo de comunicação adotado para a efetivação de uma comunicação correta e confiável entre as várias unidades que compõem essa arquitetura.

Por fim será apresentada uma descrição do funcionamento do sistema, bem como as tarefas e funções de cada uma das partes que integram o mesmo.

3.2 A Arquitetura Projetada para o Sistema Monitoramento de Silos Agrícolas

Com o objetivo de se realizar o monitoramento de temperatura de silos de armazenagem de grãos, propõe-se a implementação de um sistema cuja arquitetura baseia-se em módulos de aquisição de dados juntamente com outro módulo que desempenha a função de *host*, isto é, hospedeiro do sistema. Tanto os módulos de aquisição quanto o módulo hospedeiro são semelhantes em sua concepção, e serão apresentados mais adiante.

Cada um dos módulos de aquisição de dados é capaz de monitorar 8 sensores. Em princípio, a quantidade de módulos a se implementar é limitada pelo protocolo de comunicação, que apresenta um campo de dados para a identificação dos módulos, e o número de módulos é limitado pelo comprimento deste campo.

O módulo hospedeiro do sistema é conectado a um PC para apresentação dos dados na forma de relatórios.

Nos teste foram utilizados três módulos de aquisição mais um módulo hospedeiro, sendo este último conectado a um PC utilizando-se o padrão RS-232 de comunicação serial de dados. Para a troca de informações entre o módulo hospedeiro e os módulos de aquisição utilizou-se o padrão RS-485 de comunicação serial que permite a troca de informações entre mais de dois dispositivos interligados através de um par de fios metálicos.

A figura 3.1 ilustra o esquema da arquitetura inteligente projetada para o monitoramento de silos agrícolas.

3.2.1 Descrição dos Módulos

Todos os módulos (tanto os módulos de aquisição quanto o módulo hospedeiro) do sistema são oriundos de um circuito cujo processador é o microcontrolador 80535. As figuras 3.2 e 3.3 mostram o diagrama de blocos dos módulos de aquisição e hospedeiro respectivamente. A figura 3.4 apresenta o diagrama esquemático detalhado do circuito que é a base para implementação dos módulos.

Figura 3.1 - Arquitetura Proposta para o Sistema de Monitoramento de Silos de Armazenagem de Grãos.

Figura 3.2 - Diagrama de Blocos do Módulo de Aquisição de Dados

Figura 3.3 - Diagrama de Blocos do Módulo Hospedeiro

Figura 3.4 - Diagrama Esquemático do Circuito Utilizado para Implementação dos Módulos .

3.2.1.1 Dispositivos de Memória

- A) Memória de Programa (EPROM) : Utilizada para o armazenamento do programa que rege o funcionamento do sistema de aquisição. São empregados dois circuitos integrados de 32 *Kbytes* alcançando, desta forma, a capacidade máxima de endereçamento do microcontrolador 80535, que é de 64 *Kbytes*.
- B) Memória de Dados (RAM) : Utilizada para a gravação de dados de aplicação do sistema, dados estes que posteriormente são transferidos para o módulo hospedeiro do sistema. Também são empregados dois circuitos integrados de 32 *Kbytes*, permitindo uma autonomia de funcionamento do sistema por um período de tempo relativamente longo.

A figura 3.4 ilustra o circuito de conexão entre o 80535 e os dispositivos de memória do sistema.

3.2.1.2 O Latch de Endereços

O circuito de acesso à memória utiliza um *latch* para captura do *byte* de baixa ordem do endereço de memória (programa ou dados), pois o 80535 utiliza a porta P0 tanto para saída de endereço (*byte* de baixa ordem) como para entrada saída de dados. Este dispositivo é então utilizado para multiplexar no tempo o endereço estabelecido pelo microcontrolador com a entrada/saída de dados de leitura/escrita, respectivamente. O *latch* também está ilustrado na figura 3.4.

3.2.1.3 O Conversor Digital / Analógico

Circuito utilizado para o caso em que haja necessidade de se efetuar algum tipo de ação de controle contínuo. O conversor utilizado é o Max 543, que é do tipo 12 *bits*, possui entrada serial e utiliza apenas 3 pinos do 80535. O esquema de conexão do conversor D / A é mostrado na figura 3.4. Também é utilizado um amplificador operacional LM324, visto que o Max 543 não possui amplificador interno.

3.2.1.4 O Circuito de Interface Serial

Para a comunicação entre o módulo hospedeiro e os módulos de aquisição foi utilizado o circuito integrado DS3695, que opera com as especificações do padrão EIA RS-485, permitindo comunicação bidirecional através de um único meio (par trançado de fios). A figura

3.5 apresenta o esquema de ligação da interface serial dos módulos.

Já para a comunicação entre o módulo hospedeiro e o microcomputador foi utilizado um circuito integrado MAX232 que respeita as especificações EIA RS232-C. A interface serial entre o módulo hospedeiro e o PC é mostrado na figura 3.6.

Figura 3.5 - Esquema de Ligação da Interface Serial Padrão EIA RS-485

Figura 3.6 - Esquema de Ligação da Interface Serial entre o Módulo Hospedeiro e o PC.

3.2.1.5 O Display

O *display* é implementado apenas no módulo hospedeiro. O modelo utilizado é um *display* de cristal líquido de 16 caracteres x 2 linhas. O circuito de conexão entre o *display* e o módulo hospedeiro é mostrado na figura 3.7.

Figura 3.7 - Diagrama Esquemático para Ligação do Display de Cristal Líquido

3.2.1.6 O Teclado

Também implementado apenas no módulo hospedeiro, o teclado utilizado é do tipo 16 teclas em matriz 4x4, utilizando a Porta 4 de entrada/saída do microcontrolador 80535. Portas NAND são utilizadas para gerar interrupção ao se acionar uma tecla. A figura 3.8 mostra o esquema de ligação entre o teclado e o módulo hospedeiro.

3.2.1.7 Os Sensores de Temperaturas

Os sensores utilizados são do tipo PT100, implementados em configuração de Ponte de Wheatstone. Foi implementado um circuito de condicionamento do sinal proveniente do sensor de temperatura, através da utilização do amplificador operacional LM725 na configuração de amplificador diferencial, o qual amplifica a diferença de potencial existente nos ramos da ponte de Wheatstone. a função deste circuito, mostrado na figura 3.9, é condicionar a variação de tensão proporcional a variação de temperatura sentida pelo sensor, entre 0V e 5V, para então ser aplicada a entrada do conversor analógico/digital do microcontrolador 80535.

Figura 3.8 - Diagrama Esquemático para Ligação do Teclado

Figura 3.9 – Diagrama esquemático do circuito do sensor de temperatura

3.2.1.8A Fonte de Alimentação

A fonte de alimentação utilizada foi uma fonte regulada ajustada com tensões simétricas de +5V, -5V, +12V e -12V. A figura 3.10 ilustra o diagrama esquemático da estrutura de ligações da fonte de alimentação desenvolvida.

Figura 3.10 - Diagrama Esquemático do Circuito da Fonte de Alimentação Utilizada para o Sistema .

3.3 O Protocolo de Comunicação

O protocolo de comunicação proposto apresenta as seguintes características:

O enlace físico é feito através de par trançado de fios. O padrão a ser utilizado será o RS485.

A comunicação entre os módulos é obtida através de sua interface serial de dados a uma taxa de 9600 bauds. Como visto anteriormente, a taxa de transmissão é determinada via programação do microcontrolador. Foram implementadas funções de controle de fluxo através de *start* e *stop bytes*, controle de erro através de *check sum*, e também de *time-out*.

3.3.1 O Frame de Informação

O protocolo será direcionado por *byte* e a estrutura do *frame* proposta tem o aspecto mostrado na figura 3.11.

IQ	EN	EO	ED	CM	CB	DD	CS	FQ
----	----	----	----	----	----	----	----	----

Figura 3.11 - Aspecto do Frame do Protocolo de Comunicação

Onde:

IQ : Início de Quadro → 1 *byte*.

EN : *Enquiry* → 1 *byte*.

EO : Endereço de Origem → 2 *bytes*.

ED : Endereço de Destino → 2 *bytes*.

CM : Comando → 1 *byte*.

CB : Contador de *Bytes* de Dados → 4 *bytes*.

DD : Dados → até 64 KB (65536 *bytes*).

CS : *Check-Sum* → 2 *bytes*.

FQ : Fim de Quadro → 1 *byte*.

A transmissão de dados será feita em ASCII, o que possibilita a leitura e entendimento diretos dos mesmos através de um monitor, que facilita muito a fase de desenvolvimento.

3.3.2 Descrição dos Campos do *Frame*

3.3.2.1 Campo Início de Quadro (IQ)

Para este campo, cujo comprimento é 1 *Byte*, foi adotado o caractere “ * ”.

3.3.2.2 Campo *Enquiry* (EN)

Este campo, que tem o comprimento de 1 *Byte*, pode apresentar os seguintes caracteres:

- ? : quando o *frame* é transmitido pelo módulo hospedeiro e dirigido aos módulos de aquisição ou ao microcomputador.
- @ : quando o *frame* é transmitido por um dos módulos de aquisição e dirigido ao módulo hospedeiro.
- ! : quando o *frame* é transmitido pelo microcomputador e dirigido ao módulo hospedeiro.

3.3.2.3 Campos Endereço de Origem (EO) e Endereço de Destino (ED)

Para estes campos, ambos com comprimento de 2 *Bytes*, foram adotados os seguintes caracteres:

- FA : endereço do Módulo Host.
- FC : endereço do PC.
- 01 : endereço do Módulo de Aquisição 1.
- 12 : endereço do Módulo de Aquisição 2.
- 23 : endereço do Módulo de Aquisição 3.
- FF : endereço universal (destinado a todos os módulos de aquisição).

3.3.2.4 Campo Comando (CM)

Este campo tem o comprimento de 1 *Byte*. Os comandos implementados serão apresentados mais adiante.

3.3.2.5 Campo Contador de *Bytes* de Dados (CB)

Este campo é formado por 4 (quatro) caracteres que representam um registrador de 2 *Bytes* (0000h a FFFFh), que determina a quantidade de *Bytes* existentes no Campo de Dados do respectivo *frame*.

3.3.2.6 Campo de Dados (DD)

Este é o campo no qual são transmitidos os dados de um *frame*. Vale notar que num *frame* onde campo Contador de Bytes de Dados (CD) for “0000”, não haverá o Campo de Dados (DD). Nota-se também que, em teoria, o Campo de Dados pode ter no máximo 65536 Bytes, uma vez que o Contador de Bytes de Dados (CD) pode ter o valor FFFFh. Mais adiante veremos o conteúdo deste campo para os respectivos frames de comandos.

3.3.2.7 Campo Check-Sum (CS)

Este campo, representado por dois caracteres, mostra um número de 1 Byte que é a soma dos códigos dos caracteres transmitidos nos campos EO, ED, CM, CB e DD.

Através deste campo, realiza-se a detecção de erro de transmissão/recepção do *frame*, comparando-se o seu valor com a mesma soma realizada durante a recepção do respectivo *frame*.

3.3.2.8 Campo Fim de Quadro (FQ)

Para este campo, cujo comprimento é 1 Byte, foi adotado o caractere “ # ”.

3.3.3 Os Comandos Implementados

3.3.3.1 O Comando “ H ”.

Este comando é enviado pelo módulo Hospedeiro individualmente para cada um dos módulos de aquisição. Com este comando o Hospedeiro pede o valor atualizado do relógio (Dia/Hora) do módulo de aquisição ao qual foi endereçado. Junto a este comando não é enviado Campo de Dados e ele requer resposta.

O campo de Dados para o comando H

Um *frame* que contém o comando H não apresenta o campo de Dados. A seguir temos um exemplo de *frame* contendo o comando H:

*?FA01H0000F0#

3.3.3.2 O Comando “ K ”.

É o comando enviado pelos módulos de aquisição em resposta a um comando “H” recebido.

O campo de Dados para o comando K

Num *frame* que apresenta o comando K, o seu campo de Dados contém o valor atualizado do relógio do módulo de aquisição que o enviou. Neste caso o campo de Dados é composto por 6 (seis) caracteres, da seguinte maneira:

DDHHMM

que representam as variáveis de tempo respectivamente dias, horas e minutos (valores em hexadecimal). A seguir é apresentado um Exemplo de *frame* contendo o comando K:

*@01FAK00060100001A#

3.3.3.3 O Comando “ T ”.

Este comando é enviado pelo módulo hospedeiro a todos os módulos de aquisição, através de endereçamento universal, para que seus relógios sejam ajustados para os valores contidos no Campo de Dados enviado no mesmo *frame*. Este comando não requer resposta.

O campo de Dados para o comando T

Num *frame* que apresenta o comando T, o campo de Dados contém o valor atualizado do relógio do módulo Hospedeiro para que os módulos de aquisição atualizem os seus respectivos relógios. A seguir temos um exemplo de *frame* com o comando T :

*?FAFFT0006010C0462#

3.3.3.4 O Comando “ S ”.

Este comando é enviado pelo Hospedeiro individualmente para os módulos de aquisição, pedindo para que estes enviem de volta valores armazenados durante um período de medição (1 dia). A resposta dos módulos de aquisição ao Hospedeiro também contém o comando “ S ”. Neste caso o Campo de Dados é formado por pequenos pacotes, cada um deles referente a uma seção de aquisições, compostos da seguinte maneira:

DDHHMMSSXXSSXXSSXXSSXXSSXXSSXXSSXX

onde DDHHMM corresponde ao dia e horário da seção de medição e cada um dos 8 (oito) conjuntos SSXX corresponde ao número de identificação e respectivo valor de medição do sensor. Notar que cada valor XX é a média de um determinado número consecutivo de aquisições, média esta calculada nos próprios módulos de aquisição, com o intuito de

minimização de erros.

Portanto, se são realizadas seções de aquisição a cada 30 (trinta) minutos, e uma vez por dia o módulo hospedeiro pede que os módulos de aquisição enviem os resultados desta medição, o campo de Dados do frame de resposta a este comando terá 48 (quarenta e oito) pacotes iguais ao mostrado acima.

3.3.3.5 Os Comandos “ A ”, “ B ” e “ C ”.

Estes comandos são recebidos pelo Hospedeiro para que ele descarregue para o computador os valores respectivamente dos módulos de aquisição 1 , 2 e 3 armazenados durante 2 (dois) dias de medições. Os comandos de respostas são os próprios.

Capítulo 4

4.1 Introdução

Este capítulo trata dos resultados obtidos ao se implementar a arquitetura e o protocolo destinados ao sistema para monitoramento de silos agrícolas. Também contém as principais conclusões, comentários e proposições decorrentes do desenvolvimento deste trabalho.

4.2 O Funcionamento do Sistema

A arquitetura foi montada com quatro módulos, sendo três módulos de aquisição e um módulo hospedeiro do sistema.

Foi desenvolvido um sistema operacional e instalado no módulo hospedeiro, a fim de gerenciar as funções desempenhadas por ele e pelos módulos de aquisição, que possuem em suas memórias um sistema para comunicação com o hospedeiro e execução das tarefas de aquisição.

A interface com o operador é feita através de um teclado numérico, um *display* e o próprio PC conectado ao sistema.

4.2.1 Procedimentos de Operação

a) Após o sistema ser energizado, o módulo hospedeiro e os módulos de aquisição executam suas rotinas de inicialização e logo em seguida o hospedeiro realiza uma troca de mensagens de reconhecimento para os módulos de aquisição, exibindo no *display* uma mensagem para o operador;

b) A partir deste instante o sistema passa a realizar suas tarefas pré-programadas;

c) Em seguida o operador deve acionar no teclado a tecla F2 (função de comando T) a fim de ajustar o relógio do sistema. Os valores são inseridos no sistema através de 7 (sete) dígitos via teclado, sendo que os 3 (três) primeiros correspondem ao dia (seqüencial) e os outros 4 (quatro) dígitos seguintes correspondem a hora e minuto respectivamente;

d) A partir das 12:10h do segundo dia de aquisições, o operador deve fazer a coleta dos dados referentes às aquisições dos módulos para o IBM - PC Compatível, tomando os dados acessíveis. Esta tarefa é realizada enviando-se pacotes com os comandos A, B e C, através do Terminal[®] do Microsoft Windows. Esta operação deve ser repetida a cada 2 (dois) dias.

4.2.2 Tarefas dos Módulos de Aquisição

Os módulos de aquisição têm como principal tarefa a leitura dos níveis de tensão elétrica presentes em suas entradas analógicas, provenientes dos sensores de temperatura ou

outros.

Esta tarefa compõe-se de etapas distintas:

- a) realização de um determinado número consecutivo de leituras de cada um dos sensores com a finalidade de minimização de erros;
- b) cálculo da média das séries de medições para cada um dos sensores;
- c) armazenamento destas médias identificadas de acordo com o seu respectivo sensor de origem, bem como o dia e hora nos quais a série foi realizada. Estes dados são armazenados em alocações pré-determinadas da memória RAM dos módulos de aquisição.

Estas séries de medições são realizadas em intervalos regulares de tempo preestabelecidos, via software. No experimento foi adotado um intervalo de tempo entre cada série de medições de 30 (trinta) minutos.

Quando solicitado pelo módulo hospedeiro, o módulo de aquisição envia um pacote contendo informações um determinado número de séries de aquisições.

4.2.3 Tarefas do Módulo Hospedeiro

O módulo hospedeiro tem como tarefa principal gerenciar a interface do sistema a qual se dá via operador sendo realizada através do conjunto teclado e *display*, a troca de informações com os módulos de aquisição, solicitando que estes enviem periodicamente informações de aquisição. Estas informações serão organizadas e armazenadas para depois serem enviadas ao IBM - PC Compatível conectado ao sistema, estando assim disponibilizadas ao operador quando solicitado.

No projeto definiu-se que a cada 24 (vinte e quatro) horas o módulo hospedeiro envia mensagem aos módulos de aquisição para que estes transmitam ao mesmo as informações referentes a 1 (um) dia de aquisição, ou seja, 48 (quarenta e oito) séries de aquisição por módulo, uma vez que a cada hora são executadas duas séries de medições.

4.2.4 Funções Via Teclado

O teclado utilizado é do tipo matricial de 16 teclas, cuja configuração é apresentada na figura 4.1.

1	2	3	F1
4	5	6	F2
7	8	9	F3
F6	0	F5	F4

Figura 4.1 – Configuração do Teclado

Foram implementadas 3 (três) funções acessadas via teclado.

F1 : Função H - Quando esta função é acessada, o módulo hospedeiro envia o comando “H” individualmente para cada um dos módulos de aquisição e estes retornam seus respectivos valores de dia e hora, que são exibidos no *display* de cristal líquido.

F2 : Função T - Esta função tem como objetivo ajustar os relógios dos módulos. Quando esta função é acessada, espera-se que o operador tecle em seguida três dígitos para atualização do dia, dois dígitos para hora e dois dígitos para minuto. Os valores válidos são:

dia: 001 a 255.

hora: 00 a 23.

minuto: 00 a 59.

Após receber estes valores, o módulo hospedeiro ajusta seu relógio e envia o comando “T” para que os módulos de aquisição ajustem os seus respectivos relógios.

F4 : Função D - Esta função faz a atualização do *display*.

4.3 Rotinas do Sistema

Nesta seção, serão apresentadas as rotinas desenvolvidas para os módulos de aquisição e para o módulo hospedeiro do sistema. As listagens dos códigos-fonte em Assembly do 80535 das rotinas desenvolvidas encontram-se no Apêndice B (módulo hospedeiro) e no Apêndice C (módulos de aquisição) deste trabalho.

4.3.1 Rotinas do Módulo Hospedeiro

As rotinas do módulo hospedeiro foram divididas em grupos de acordo com as suas finalidades. Foram desenvolvidas as seguintes rotinas.

4.3.1.1 Rotina de Inicialização do Módulo Hospedeiro

A rotina de inicialização do módulo hospedeiro foi implementada pelo algoritmo abaixo descrito:

Begin

Testa RAM externa

Inicializa interrupções

Inicializa display

Inicializa porta serial

Inicializa contadores (modo recarga com overflow a cada 50 ms)

Envia comando H para Módulo1

Envia comando H para Módulo2

Envia comando H para Módulo3

Loop espera de evento.

Para interrupção de contador

Vai Para rotina de interrupção de contador.

Fim Para

Para interrupção externa (teclado acionado)

Vai Para rotina de int. externa

FimPara

Para interrupção de porta serial
 Vai Para rotina de int. de porta serial.
 Fim Para
 Volta para Loop
End.

4.3.1.2 Rotina de Interrupção de Contadores (Relógio) do Módulo Hospedeiro

Quando uma interrupção é gerada pelo *overflow* do contador o programa é desviado para sua respectiva rotina de interrupção. O *overflow* do contador ocorre a cada 50 ms.

Begin
 Incrementa contador de interrupções
 Se contador de interrupções = 20 ($20 * 50 \text{ ms} = 1 \text{ segundo}$)
 Então Faz contador de interrupções = 0
 Incrementa contador de segundos
 Se contador de segundos = 60
 Então Faz contador de segundos = 0
 Incrementa contador de minutos
 Se contador de minutos = 60
 Então Faz contador de minutos = 0
 Incrementa contador de horas
 Se contador de horas = 24
 Então Faz contador de horas = 0
 Incrementa contador de dias
 Fim Se
 Fim Se
 Fim Se
 Atualiza relógio do display
 Se relógio = 12:10:00
 Então Envia frame com comando S para Módulo 1
 Recebe frame do Módulo 1 com dados de aquisição
 Se frame válido
 Então Faz back-up de dados do Módulo 1
 Senão Envia frame com comando S para Módulo 1
 Fim Se

```

        Envia frame com comando S para Módulo 2
        Recebe frame do Módulo 2 com dados de aquisição
        Se frame válido
            Então Faz back-up de dados do Módulo 2
            Senão Envia frame com comando S para Módulo 2
        Fim Se
        Envia frame com comando S para Módulo 3
        Recebe frame do Módulo 3 com dados de aquisição
        Se frame válido
            Então Faz back-up de dados do Módulo 3
            Senão Envia frame com comando S para
                Módulo 3
        Fim Se
    Fim Se
Fim Se
Retorna
End.

```

4.3.1.3 Rotina de Interrupção Externa (Teclado) do Módulo Hospedeiro

Quando uma tecla for acionada no teclado o pino INT0 do 80535 muda de estado e então uma interrupção é gerada, desviando o programa para o endereço de sua rotina de interrupção externa.

Begin

```

    Identifica tecla acionada
    Se tecla = função válida
        Então Para tecla = Função H
            Envia frame com comando H para o módulo 1
            Recebe frame do Módulo 1 seu estado
            Se frame válido
                Então Escreve estado do Módulo 1 no display
                Senão Envia frame com comando H para Módulo 1
            Fim Se
            Envia frame com comando H para Módulo 2
            Recebe frame do Módulo 2 com seu estado

```


Se frame válido
 Então Escreve estado do Módulo2 no display
 Senão Envia frame com comando H para Módulo 2

Fim Se

Envia frame com comando H para Módulo 3

Recebe frame do Módulo 3 com seu estado

Se frame válido
 Então Escreve estado do Módulo 3 no display
 Senão Envia frame com comando H para Módulo 3

Fim Se

Fim Para

Para tecla = Função T

Lê Dia, Hora, Minuto através teclado

Atualiza Dia, Hora, Minuto

Envia comando T para todos os módulos de aquisição

Fim Para

Para tecla = Função D

Faz refresh do display

Fim Para

Senão Escreve mensagem de função inválida no display

Fim Se

Retorna

End.

4.3.1.4 Rotina de Comunicação Serial do Módulo Hospedeiro

No estado de espera o módulo hospedeiro fica ajustado para receber informação proveniente do PC. Assim que uma informação trafegar pelo barramento é gerada uma interrupção que desvia o programa do módulo para uma rotina de interrupção.

Begin

Captura frame do barramento serial

Para frame = válido

Para comando = A

Envia para PC frame com dados de aquisições do módulo 1

Fim Para

```
    Para comando = B
        Envia para PC frame com dados de aquisições do módulo 2
    Fim Para
    Para comando = C
        Envia para PC frame com dados de aquisições do módulo 3
    Fim Para
Fim Para
Retorna
End.
```

4.3.2 Rotinas dos Módulos de Aquisição

4.3.2.1 Rotina de Inicialização dos Módulos de Aquisição

```
Begin
    Testa RAM externa
    Inicializa interrupções
    Inicializa porta serial
    Inicializa conversor A/D
    Inicializa contadores (modo recarga com overflow a cada 50 ms)
    Loop espera de evento.
        Para interrupção de contador
            Vai Para rotina de interrupção de contador.
        Fim Para
        Para interrupção de porta serial
            Vai Para rotina de int. de porta serial.
        Fim Para
    Volta para Loop
End.
```

4.3.2.2 Rotina de Interrupção de Contadores (Relógio) dos Módulos de Aquisição

```
Begin
    Incrementa contador de interrupções
```

Se contador de interrupções = 20 ($20 * 50 \text{ ms} = 1 \text{ segundo}$)

Então Faz contador de interrupções = 0

Incrementa contador de segundos

Se contador de segundos = 60

Então Faz contador de segundos = 0

Incrementa contador de minutos

Se contador de minutos = 60

Então Faz contador de minutos = 0

Incrementa contador de horas

Se contador de horas = 24

Então Faz contador de horas = 0

Incrementa contador de dias

Fim Se

Fim Se

Fim Se

Se contador de minutos = 29 ou =59

Então Chama rotina de aquisição

Fim Se

Se relógio = 12:05

Então Faz back-up de aquisições diárias

Fim Se

Fim Se

Retorna

End.

4.3.2.3 Rotina de Aquisição de Dados dos Módulos de Aquisição

Begin

Para N=1 até 8

Faz 20 aquisições do sensor N

Calcula média dos 20 valores

Escreve dados referentes ao sensor N

Incrementa apontador de escrita

Próximo N

Fim Para

End.

4.3.2.4 Rotina de Comunicação Serial dos Módulos de Aquisição

Begin

Captura frame do barramento serial

Para frame = válido

Para comando = H

Envia para módulo hospedeiro frame com informação de estado 2

Fim Para

Para comando = T

Faz ajuste do relógio

Fim Para

Para comando = S

Envia para módulo hospedeiro frame com dados de aquisições de 1 dia

Fim Para

Fim Para

Retorna

End.

4.4 Diagramas de Estado Resultantes na Concepção Sistêmica do Projeto

Após a inicialização do sistema como um todo, os módulos de aquisição entram em estado de espera. São dois os tipos de eventos que resultam alguma ação dos módulos. Os eventos relacionados com o relógio e com a comunicação serial. Os eventos relacionados com o relógio proporcionam aos módulos de aquisição a realização de determinadas ações em horários predefinidos. As ações desencadeadas via relógio são a leitura dos sensores e a escrita de *back-up* de dados de aquisições. Os eventos relacionados com a comunicação serial são aqueles onde os módulos recebem alguma informação enviada pelo módulo hospedeiro através do barramento serial, reconhecendo a informação e, se requerido, enviando uma mensagem de resposta. A figura 4.2 ilustra o diagrama de estados dos módulos de aquisição.

Da mesma forma, encontra-se similar procedimento no módulo hospedeiro. Eventos de relógio do módulo hospedeiro relacionam o envio de mensagens para os módulos de aquisição e escritas de *back-up*. Eventos de comunicação serial do módulo hospedeiro constituem-se de mensagens recebidas do PC através do barramento serial. Um outro tipo de evento através do qual o módulo hospedeiro realiza alguma ação é o evento de teclado, para executar funções predefinidas. A figura 4.3 ilustra o diagrama de estados do módulo hospedeiro.

Figura 4.2 – Diagrama de Estados dos Módulos de Aquisição

Figura 4.3 – Diagrama de Estados do Módulo de Hospedeiro

4.5 Desempenho do Circuito do Sensor de Temperatura

O circuito do sensor de temperatura, que utiliza como sensor um PT100 em configuração de ponte de Wheatstone e como condicionador de sinal um amplificador operacional LM725 na configuração de amplificador diferencial apresentou uma resposta de variação de temperatura de 20,6 mV / °C. Com o conversor analógico / digital do 80535 ajustado para a faixa de 0 - 5V, obteve-se a seguinte quantização:

$$1 \text{ bit} = 5\text{V} / 256 \rightarrow 1 \text{ bit} = 19,5 \text{ mV}$$

$$\text{Se } 1^\circ\text{C} = 20,6 \text{ mV} \rightarrow 1 \text{ bit} = 0,948^\circ\text{C}$$

A arquitetura desenvolvida permite melhorar esta relação de duas formas, ou seja, alterando-se o fator de amplificação do circuito condicionador de sinal do sensor ou alterando-se a faixa de operação do conversor A/D do microcontrolador. Para o caso de medições de temperatura de grande massas de grãos, como ocorre em silos de armazenamento, a quantização de 0,5 °C é suficiente, quantização esta que pode ser obtida pelo sistema.

4.6 Detalhamento de Arquivo de Dados Armazenado no Host e IBM – PC Compatível

A coleta dos dados de aquisição do módulo host para o PC é feita através do Windows Terminal[®], ajustado com taxa de transmissão de 9600 bauds, 8 bits de dados, sem paridade, 1 bit de parada e controle de fluxo Xon/Xoff.

A tabela 4.1 contém os frames enviados pelo IBM - PC Compatível para que o módulo hospedeiro retorne os dados de aquisição, respectivamente dos módulos 1, 2 e 3.

Tabela 4.1 – Frames Enviados pelo IBM - PC Compatível para Coletar Dados de Aquisições

*!FCFAA000011#
*!FCFAB000012#
*!FCFAC000013#

A tabela 4.2 apresenta um exemplo de arquivo de dados de sensores de um dos módulos do sistema enviados para o IBM - PC Compatível.

Com arquivos deste tipo disponíveis, através de uma macro desenvolvida em Excel[®], os dados de aquisição são apresentados na forma de relatórios, como no exemplo da tabela 3.3.

A figura 4.4 apresenta uma fotografia do sistema implementado.

Tabela 4.3 – Relatório Resultante do Arquivo de Aquisições de 1 Módulo

Relatório de Aquisição		Início do Experimento:		Período: 09/12/98 - 8 :59 h a					
Módulo: 1		09/12/98		10/12/98 - 10 :59 h					
Data	Hora	Sensor 00	Sensor 01	Sensor 02	Sensor 03	Sensor 04	Sensor 05	Sensor 06	Sensor 07
09/12/98	8 : 59	31,3	31,3	30,3	31,3	30,3	31,3	31,3	31,3
09/12/98	9 : 29	31,3	31,3	30,3	31,3	30,3	31,3	31,3	31,3
09/12/98	9 : 59	31,3	31,3	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	10 : 29	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	10 : 59	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	11 : 29	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	11 : 59	32,2	32,2	31,3	32,2	31,3	32,2	32,2	32,2
09/12/98	12 : 29	32,2	32,2	31,3	32,2	31,3	31,3	32,2	32,2
09/12/98	12 : 59	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	13 : 29	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	13 : 59	31,3	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	14 : 29	31,3	31,3	30,3	31,3	31,3	31,3	31,3	31,3
09/12/98	14 : 59	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	15 : 29	31,3	31,3	30,3	31,3	30,3	31,3	31,3	31,3
09/12/98	15 : 59	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	16 : 29	31,3	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	16 : 59	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	17 : 29	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	17 : 59	31,3	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	18 : 29	32,2	32,2	31,3	32,2	31,3	32,2	31,3	31,3
09/12/98	18 : 59	31,3	32,2	30,3	31,3	30,3	31,3	31,3	31,3
09/12/98	19 : 29	31,3	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	19 : 59	32,2	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	20 : 29	31,3	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	20 : 59	31,3	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	21 : 29	31,3	32,2	30,3	32,2	31,3	31,3	31,3	31,3
09/12/98	21 : 59	31,3	31,3	30,3	32,2	30,3	31,3	31,3	31,3
09/12/98	22 : 29	31,3	32,2	30,3	31,3	30,3	31,3	31,3	31,3
09/12/98	22 : 59	31,3	31,3	30,3	31,3	30,3	31,3	31,3	31,3
09/12/98	23 : 29	31,3	31,3	30,3	31,3	30,3	31,3	31,3	31,3
09/12/98	23 : 59	31,3	31,3	30,3	31,3	30,3	31,3	31,3	30,3
10/12/98	0 : 29	31,3	31,3	30,3	31,3	30,3	30,3	31,3	30,3
10/12/98	0 : 59	31,3	31,3	30,3	31,3	30,3	31,3	30,3	30,3
10/12/98	1 : 29	31,3	31,3	30,3	31,3	30,3	30,3	31,3	30,3
10/12/98	1 : 59	31,3	31,3	30,3	31,3	30,3	30,3	30,3	30,3
10/12/98	2 : 29	31,3	31,3	30,3	31,3	30,3	30,3	30,3	31,3
10/12/98	2 : 59	32,2	31,3	30,3	32,2	31,3	31,3	31,3	31,3

10/12/98	3 : 29	31,3	31,3	30,3	31,3	30,3	31,3	31,3	31,3
10/12/98	3 : 59	31,3	31,3	30,3	31,3	30,3	31,3	31,3	31,3
10/12/98	4 : 29	31,3	31,3	29,4	31,3	30,3	31,3	31,3	31,3
10/12/98	4 : 59	31,3	31,3	29,4	31,3	30,3	31,3	31,3	31,3
10/12/98	5 : 29	31,3	31,3	29,4	31,3	30,3	30,3	30,3	30,3
10/12/98	5 : 59	31,3	31,3	29,4	31,3	30,3	30,3	30,3	30,3
10/12/98	6 : 29	31,3	31,3	29,4	31,3	30,3	30,3	30,3	30,3
10/12/98	6 : 59	31,3	31,3	29,4	31,3	30,3	30,3	30,3	30,3
10/12/98	7 : 29	31,3	31,3	29,4	31,3	30,3	30,3	30,3	30,3
10/12/98	7 : 59	30,3	31,3	29,4	30,3	29,4	29,4	29,4	30,3
10/12/98	8 : 29	30,3	30,3	29,4	30,3	29,4	30,3	30,3	30,3
10/12/98	8 : 59	31,3	31,3	29,4	31,3	30,3	30,3	30,3	30,3
10/12/98	9 : 29	30,3	31,3	29,4	31,3	29,4	30,3	30,3	30,3
10/12/98	9 : 59	30,3	30,3	29,4	30,3	29,4	30,3	30,3	30,3
10/12/98	10 : 29	30,3	30,3	29,4	31,3	30,3	30,3	30,3	30,3
10/12/98	10 : 59	30,3	30,3	29,4	30,3	30,3	30,3	30,3	30,3

Figura 4.4 - Fotografia do sistema implementado.

4.7 Conclusões

O desenvolvimento deste trabalho proporcionou o estabelecimento de conhecimentos em interfaces de sistemas, técnicas de programação e desenvolvimento de software e hardware para arquitetura inteligente baseada no microcontrolador 80535.

Dentre as principais conclusões destacam-se:

a) A opção por se interconectar módulos de aquisição através de um barramento serial gerenciados por um módulo hospedeiro permitiu a obtenção de um sistema versátil e flexível direcionado ao ambiente agropecuário e agroindustrial;

b) A arquitetura projetada com interconexão de módulos de aquisição apresentou-se de forma bastante satisfatória e consistiu de uma opção otimizada para a aquisição de dados no ambiente agropecuário e agroindustrial;

c) A implementação do padrão de comunicação serial RS-485 permitiu a interconexão multiponto e apresentou-se de forma confiável, com a vantagem de permitir a conexão de um grande número de módulos de aquisição. Entretanto, o sistema projetado também permite a conexão serial pelo padrão RS-232 que é amplamente difundido, constituindo desta forma um sistema flexível e bastante versátil.

d) A comunicação entre módulo hospedeiro e IBM - PC Compatível que foi primeiramente programada para ser executada via operador (a cada 2 dias), também pode ser feita através da conexão momentânea de um terminal tipo Lap Top. Adicionalmente, a partir de resultados obtidos, observou-se que esta é uma tarefa bastante flexível para o operador visto que após as 12:10h do segundo dia de aquisição, ele ainda tem uma tolerância de tempo de 24 horas, até que o back-up de segurança de aquisições do período de um dia seja refeito. Esta tarefa pode também ser automatizada, via rotina especializada, entretanto, neste caso o IBM - PC Compatível ficaria dedicado a tarefa de coleta final de informação dos sensores.

e) Conforme projetado, o sistema possibilita que sejam realizadas 96 (noventa e seis) aquisições até que os dados referente às medições de cada módulo sejam coletadas para o IBM - PC Compatível. A definição da quantidade de aquisições possíveis entre duas coletas de dados para o IBM - PC Compatível está intimamente ligada com o intervalo em que cada aquisição é realizada, variando conforme a aplicação a que o sistema se destina.

f) Dentre as características elétricas do sistema temos:

Faixa de leitura: 248 °C (máxima).

Resolução : 0,9 °C (podendo ser programado para até 0,4 °C).

Número de canais : 8 canais por módulo de aquisição, devendo os sinais provenientes dos sensores estarem condicionados na faixa entre 0 e 5V.

Consumo : 150 mA x 5V (módulo hospedeiro + 3 módulos de aquisição)

Tempo de transferência : aproximadamente 1,6 segundos.

Capacidade de armazenamento : 96 séries de aquisições. Com intervalo entre aquisições de 30 minutos, deve-se realizar a transferência de dados armazenados para o PC a cada 2 dias.

g) Como idéias para trabalhos futuros e melhoria do sistema, sugere-se a implementação de um relógio de hardware externo ao chip do microcontrolador 80535 e o desenvolvimento e implementação de outras funções acessadas via teclado, tal como execução de uma série de aquisições com a respectiva visualização do resultado no display do sistema ou ainda visualização da última aquisição realizada pelos módulos, em quaisquer dos pontos de medição. Também foi previsto e se faz necessário para o sistema a implementação de proteção contra transientes e um sistema de *no-break* ou baterias para a eventual falta de energia.

h) A arquitetura projetada contribui prontamente para a flexibilização da aquisição de dados de temperatura (ou outros) em silos agrícolas utilizados para armazenagem de grãos para o imediato uso em modelos de predição e isotermas e migração de umidade, para a adequação de um otimizado processo de aeração. A utilização de tal sistema implica na redução de perdas no armazenamento de grãos, ocasionadas pelo deterioramento dos mesmos, devido a variações na temperatura e umidade da massa armazenada.

Apêndice A

A. Série de Instruções do Microcontrolador 80535

A.1 Operações Aritméticas

Mnemônico		Descrição	Byte	Duração em Ciclos de Máquina
ADD	A, Rn	Adiciona Registrador ao Acumulador	1	1
ADD	A, direto	Adiciona byte direto ao Acumulador	2	1
ADD	A, a Ri	Adiciona RAM indireta ao Acumulador	1	1
ADD	A, #dado	Adiciona dado imediato ao Acumulador	2	1
ADDC	A, Rn	Adiciona Registrador ao Acumulador com "Carry"	2	1
ADDC	A, direto	Adiciona byte direto ao Acumulador com "Carry"	2	1
ADDC	A, a Ri	Adiciona RAM indireta ao Acumulador com "Carry"	1	1
ADDC	A, #dado	Adiciona dado imediato ao Acumulador com "Carry"	2	1
SUBB	A, Rn	Subtrai Registrador do Acumulador com "Borrow"	1	1
SUBB	A, direto	Subtrai byte direto do Acumulador com "Borrow"	2	1
SUBB	A, a Ri	Subtrai RAM indireta do Acumulador com "Borrow"	1	1
SUBB	A, #dado	Subtrai dado imediato do Acumulador com "Borrow"	2	1
INC	A	Incrementa o Acumulador	1	1
INC	Rn	Incrementa Registrador	1	1
INC	direto	Incrementa byte direto	2	1
INC	@Ri	Incrementa RAM indireta	1	1
DEC	A	Decrementa o Acumulador	1	1
DEC	Rn	Decrementa Registrador	1	1
DEC	direto	Decrementa byte direto	2	1
DEC	@ Ri	Decrementa RAM indireta	1	1
INC	DPTR	Incremente Apontador de Dados	1	2
MUL	AB	Multiplica A e B	1	4
DIV	AB	Divide A por B	1	4
DA	A	Ajuste Decimal do Acumulador	1	1

A.2 Operações Lógicas

Mnemônico		Descrição	Byte	Duração em Ciclos de Máquina
ANL	A, Rn	AND do Registrador para o Acumulador	1	1
ANL	A, direto	AND do byte direto para o Acumulador	2	1
ANL	A, @ Ri	AND de RAM indireta para o Acumulador	1	1
ANL	A, #dado	AND de dado imediato para o Acumulador	2	1
ANL	direto, A	AND do Acumulador para byte direto	2	1
ANL	direto, #dado	AND de dado imediato para byte direto	3	2
ORL	A, Rn	OR do Registrador para o Acumulador	1	1
ORL	A, direto	OR do byte direto para o Acumulador	2	1
ORL	A, @ Ri	OR de RAM indireta para o Acumulador	1	1
ORL	A, #dado	OR de dado imediato para o Acumulador	2	1
ORL	direto, A	OR do Acumuladpr para byte direto	2	1
ORL	direto, #dado	OR de dado imediato para byte direto	3	2
XRL	A, Rn	OR-Exclusivo do Registrador para o Acumulador	1	1
XRL	A, direto	OR-Exclusivo do byte direto para o Acumulador	2	1
XRL	A, @ Ri	OR-Exclusivo de RAM indireta para o Acumulador	1	1
XRL	A, #dado	OR-Exclusivo de dado imediato para o Acumulador	2	1
XRL	direto, A	OR-Exclusivo do Acumuladpr para byte direto	2	1
XRL	direto, #dado	OR-Exclusivo de dado imediato para byte direto	3	2
CRL	A	Zera o Acumulador	1	1
RL	A	Rotaciona o Acumulador para a Esquerda	1	1
RLC	A	Rotaciona o Acumulador para a Esquerda passando pelo "Carry"	1	1
RR	A	Rotaciona o Acumulador para a Direita	1	1
RRC	A	Rotaciona o Acumulador para a Direita passando pelo "Carry"	1	1
SWAP	A	Troca "nibbles" no Acumulador	1	1

A.3 Transferência de Dados

Mnemônico		Descrição	Byte	Duração em Ciclos de Máquina
MOV	A, Rn	Move Registrador para o Acumulador	1	1
MOV	A, direto	Move byte direto para o Acumulador	2	1
MOV	A, @ Ri	Move RAM indireta para o Acumulador	1	1
MOV	A, #dado	Move dado imediato para o Acumulador	2	1
MOV	Rn, A	Move Acumulador para Registrador	1	1
MOV	Rn, direto	Move byte direto para Registrador	2	2
MOV	Rn, #dado	Move dado imediato para Registrador	2	1
MOV	direto, A	Move Acumulador para byte direto	2	1
MOV	direto, Rn	Move Registrador para byte direto	2	2
MOV	direto, direto	Move byte direto para byte direto	3	2
MOV	direto, @ Ri	Move RAM indireta para byte direto	2	2
MOV	direto, #dado	Move dado imediato para byte direto	3	2
MOV	@ Ri, A	Move Acumulador para RAM indireta	1	1
MOV	@ Ri, direto	Move byte direto para RAM indireta	2	2
MOV	@ Ri, #dado	Move dado imediato para RAM indireta	2	1
MOV	DPTR, #dado16	Carrega Apontador de Dados com Constante de 16 bits	3	2
MOVC	A, @ A+DPTR	Move byte de Código relativo ao DPTR para o Acumulador	1	2
MOVC	A, @ A+PC	Move byte de Código relativo ao Contador de Progr. para o Acumulador	1	2
MOVX	A, @ Ri	Move RAM Externa (endereço de 8 bits) para o Acumulador	1	2
MOVX	A, @ DPTR	Move RAM Externa (endereço de 16bits) para o Acumulador	1	2
MOVX	@ Ri, A	Move o Acumulador para RAM Externa (endereço de 8 bits)	1	2

Mnemônico		Descrição	Byte	Duração em Ciclos de Máquina
MOVX	@ DPTR, A	Move o Acumulador para RAM Externa (endereço de 16 bits)	1	2
PUSH	direto	Empurra byte direto para a pilha	2	2
POP	direto	Traz byte direto da pilha	2	2
XCH	A, Rn	Troca Registrador com Acumulador	1	1
XCH	A, direto	Troca byte direto com Acumulador	2	1
XCH	A, @ Ri	Troca RAM indireta com o Acumulador	1	1
XCHD	A, @ Ri	Troca dígito de baixa ordem da RAM indireta com o Acumulador	1	1

A.4 Manipulação de Variáveis Booleanas

Mnemônico		Descrição	Byte	Duração em Ciclos de Máquina
CLR	C	Zera o "Carry"	1	1
CLR	bit	Zera bit direto	2	1
SETB	C	Coloca nível lógico "1" no "Carry"	1	1
SETB	bit	Coloca nível lógico "1" em bit direto	2	1
CPL	C	Complementa o "Carry"	1	1
CPL	bit	Complementa bit direto	2	1
ANL	C, bit	AND de bit direto para o "Carry"	2	2
ANL	C, /bit	AND de complemento de bit direto para o "Carry"	2	2
ORL	C, bit	OR de bit direto para o "Carry"	2	2
ORL	C, /bit	OR de complemento de bit direto para o "Carry"	2	2
MOV	C, bit	Move bit direto para o "Carry"	2	1
MOV	bit, C	Move o "Carry" para bit direto	2	2
JC	rel	Salta se "Carry" = 1	2	2
JNC	rel	Salta se "Carry" = 0	2	2
JB	bit, rel	Salta se bit direto = 1	3	2
JNB	bit, rel	Salta se bit direto = 0	3	2
JBC	bit, rel	Salta se bit direto = 1 e zera o bit	3	2

A.5 Instruções de Desvio de Programa

Mnemônico	Descrição	Byte	Duração em Ciclos de Máquina
ACALL end 11	Desvio Absoluto para Subrotina	2	2
LCALL end 16	Desvio Longo para Subrotina	3	2
RET	Retorno de Subrotina	1	2
RETI	Retorno de Interrupção	1	2
AJMP end 11	Salto Absoluto	2	2
LJMP end 16	Salto Longo	3	2
SJMP rel	Salto Curto (endereço relativo)	2	2
JMP @ A+DPTR	Salto Indireto relativo ao DPTR	1	2
JZ rel	Salta se Acumulador é zero	2	2
JNZ rel	Salta se Acumulador não é zero	2	2
CJNE A, direto, rel	Compara byte direto com Acumulador e salta se não for igual	3	2
CJNE A, #dado, rel	Compara dado imediato com Acumulador e salta se não for igual	3	2
CJNE Rn, #dado, rel	Compara dado imediato com Registrador e salta se não for igual	3	2
CJNE @ Ri, #dado, rel	Compara dado imediato com dado indireto e salta se não for igual	3	2
DJNZ Rn, rel	Decrementa Registrador e salta se não for zero	2	2
DJNZ direto, rel	Decrementa byte direto e salta se não for zero	3	2
NOP	Nenhuma operação	1	1

Apêndice B

```

.....
*
*      wagner roberto balsani
*      mestrado em engenharia eletrica
*      universidade de sao paulo
*      escola de engenharia de sao carlos
*      orientador: prof. dr. paulo estevao cruvinel
*
*
.....

```

```

=====
REC_QUA_FROM_PCEQU 1800H ;
SND_CMD_H EQU 2000H ;
SEND_CMD_T EQU 22C4H ;
SND_CMD_S EQU 23ECH ;
LCD8BITS EQU 2800H ; endereços hexadecimais das rotinas
WRLCD_COM EQU 2881H ; gravadas nas EEPROM's.
WRLCD_DATA EQU 2875H ;
PRTLCD EQU 291EH ;
BINASC EQU 1B08H ;

```

```

; variaveis.h

```

```

; alocação de variaveis na ram interna

```

```

; bytes:

```

```

CONT_INT EQU 060H ; contador de interrupcoes do timer 2
SEGUNDOS EQU 061H ; contador de segundos
MINUTOS EQU 062H ; contador de minutos
HORAS EQU 063H ; contador de horas
DIAS EQU 064H ; contador de dias (byte baixo)
APONT_MEM_AQ_HI EQU 066H ; apontador de escrita de dados de aquisição (byte alto)
APONT_MEM_AQ_LO EQU 067H ; apontador de escrita de dados de aquisição (byte baixo)
SENSOR EQU 068H ; utilizado durante envio de dados ao host

```

```

; bits:

```

```

SEGFLAG EQU 01H ; sinalizador de incremento de segundos
MINFLAG EQU 02H ; sinalizador de incremento de minutos
HORFLAG EQU 03H ; sinalizador de incremento de horas
DIAFLAG EQU 04H ; sinalizador de incremento de dias
SERFLAG EQU 05H ; sinalizador de comunicação serial
FUNFLAG EQU 06H ; sinalizador de função de teclado
TEC_VAL EQU 07H ; sinalizador de validade da tecla acionada
NUMBFLAG EQU 08H ; sinalizador de número teclado
READYNUM EQU 09H ; sinalizador de fim de entrada de dígitos
TFUNFLAG EQU 0AH ; sinalizador de função T
HFUNFLAG EQU 0BH ; sinalizador de função H
SFUNFLAG EQU 0CH ; sinalizador de função S

```

```

; bytes:

```

```

RXST EQU 069H ; registrador de status da ram externa
INIT_ADDR_HI EQU 06AH ; registradores de endereço da área de
INIT_ADDR_LO EQU 06BH ; dados a serem transmitidos

```

```

SUNDA1 EQU 06CH
SUNDA2 EQU 06DH

```

```

ZERO_COUNTER EQU 06EH
LINE EQU 070H
ADDRESS EQU 06FH
ZERO_BIT EQU 071H
NUMERO EQU 072H
AUX1 EQU 073H
AUX2 EQU 074H
SOBRA EQU 075H
CONTDIG EQU 076H
CONTSEC EQU 077H

```

; variáveis utilizadas para emissão de dados via serial

S_CHECK_SUM	EQU 078H	:	
S_COMANDO	EQU 079H	:	
S_CONT_DADOS_4	EQU 07AH	:	
S_CONT_DADOS_3	EQU 07BH	:	
S_CONT_DADOS_2	EQU 07CH	:	variáveis
S_CONT_DADOS_1	EQU 07DH	:	
S_CONT_DADOS_HI	EQU 07EH	:	utilizadas
S_CONT_DADOS_LO	EQU 07FH	:	
S_CHECK_SUM_HI	EQU 080H	:	para
S_CHECK_SUM_LO	EQU 081H	:	
CH_SUM_CALC	EQU 082H	:	comunicação
CONT_DADOS_4	EQU 083H	:	
CONT_DADOS_3	EQU 084H	:	entre
CONT_DADOS_2	EQU 085H	:	
CONT_DADOS_1	EQU 086H	:	os
CONT_DADOS_HI	EQU 087H	:	
CONT_DADOS_LO	EQU 088H	:	módulos
CH_SUM_HEX	EQU 089H	:	
R_COMANDO	EQU 08AH	:	
CHECK_SUM_HI	EQU 08BH	:	
CHECK_SUM_LO	EQU 08CH	:	

S_INIT_DATA_ADDR_HI	EQU 08DH	:
S_INIT_DATA_ADDR_LO	EQU 08EH	:
R_INIT_DATA_ADDR_HI	EQU 08FH	:
R_INIT_DATA_ADDR_LO	EQU 090H	:

programa principal

ORG 1000H

INICIALIZA:

MOV	IPO, #0	; baixa prioridade para todas as interrup.
MOV	IP1, #0	
ORL	IP1, #00100000B	
ORL	IPO, #00000001B	
MOV	PSW, #00H	
MOV	SP, #30H	
LCALL	LCD8BITS	; chama rotina de inicialização do display
MOV	RO, #02H	
LCALL	WRLCDCOM	; chama rotina de envio de comando p/ o display
LCALL	PRTLCD	; chama rotina de escrita no display
DB	"INICIANDO...", 0	
MOV	RO, #0CH	
LCALL	WRLCDCOM	
LCALL	RAMXTEST	; chama rotina de teste de memória RAM do módulo
LCALL	SDELAY	; chama rotina de espera (1 segundo)
LCALL	INIT_SER	; chama rotina de inic. da porta serial do 80535
LCALL	SDELAY	
LCALL	INIT_TIMER2	; chama rotina de inic. Do timer2 do 80535
LCALL	SDELAY	
LCALL	SND CMD_H	; chama rotina de envio do comando H p/ os módulos
LCALL	SDELAY	
MOV	ZERO_COUNTER, #0	
MOV	ADDRESS, #0	; inicializa

```

MOV     ZERO_BIT, #0      ; variáveis
MOV     NUMERO, #0        ; para operação
CLR     FUNFLAG           ; do teclado
CLR     TEC_VAL           ;
CLR     NUMBFLAG          ;

```

```

LCALL   INIT_TECL         ; chama rotina de inicialização do teclado
LCALL   SDELAY

```

LOOP:

```

JB      HFUNFLAG, GODOHFUN ; este é um loop de espera
JB      SFUNFLAG, GODOSFUN  ; até que uma tecla de função
JB      TFUNFLAG, GODOTFUN  ; seja acionada
SJMP    LOOP

```

GODOSFUN:

```

LCALL   SNDCMD_S          ; ao reconhecer a tecla da função S um
CLR     SFUNFLAG           ; comando S é enviado
SJMP    LOOP

```

GODOHFUN:

```

LCALL   SNDCMD_H          ; ao reconhecer a tecla da função H um
CLR     HFUNFLAG           ; comando H é enviado
SJMP    LOOP

```

GODOTFUN:

```

; quando a tecla de função T é acionada, o módulo espera até
; que outros 7 (sete) dígitos sejam teclados, sendo que os 3
; (três) primeiros serão o novo valor decimal para a variável
; "DIAS", os outros 4 (quatro) dígitos teclados serão os novos
; valores para as variáveis "HORAS" e "MINUTOS" respectivamente.
; Após receber os 7 dígitos, o módulo ajusta o seu relógio e
; envia uma mensagem de comando T para que os módulos de
; aquisição ajustem seus relógios com os novos valores.

```

LOOP1DIA:

```

MOV     A, CONTDIG
CJNE    A, #6, LOOP1DIA
MOV     A, NUMERO
MOV     B, #100
MUL     AB
MOV     DIAS, A

```

LOOP2DIA:

```

MOV     A, CONTDIG
CJNE    A, #5, LOOP2DIA
MOV     A, NUMERO
MOV     B, #10
MUL     AB
ADDC    A, DIAS
MOV     DIAS, A

```

LOOP3DIA:

```

MOV     A, CONTDIG
CJNE    A, #4, LOOP3DIA
MOV     A, NUMERO
ADD     A, DIAS
MOV     DIAS, A

```

TLOOP:

```

MOV     A, CONTDIG
CJNE    A, #03, TLOOP
MOV     A, NUMERO
MOV     B, #10
MUL     AB
MOV     HORAS, A

```

TLOOP1:

```

MOV     A, CONTDIG

```



```

CJNE    A, #02, TLOOP1
MOV     A, NUMERO
ADD     A, HORAS
MOV     HORAS, A

TLOOP2:
MOV     A, CONTDIG
CJNE    A, #01, TLOOP2
MOV     A, NUMERO
MOV     B, #10
MUL     AB
MOV     MINUTOS, A

TLOOP3:
MOV     A, CONTDIG
CJNE    A, #0, TLOOP3
MOV     A, NUMERO
ADD     A, MINUTOS
MOV     MINUTOS, A
CLR     TFUNFLAG
CLR     NUMBFLAG

MOV     DPH, #09FH
MOV     DPL, #00H
MOV     A, DIAS
LCALL   BINASC
MOVX    @DPTR, A
INC     DPTR
MOV     A, R2
MOVX    @DPTR, A
INC     DPTR
MOV     A, HORAS
LCALL   BINASC
MOVX    @DPTR, A
MOV     A, R2
INC     DPTR
MOVX    @DPTR, A
INC     DPTR
MOV     A, MINUTOS
LCALL   BINASC
MOVX    @DPTR, A
MOV     A, R2
INC     DPTR
MOVX    @DPTR, A
SETB    SEGFLAG
SETB    MINFLAG
SETB    HORFLAG
SETB    DIAFLAG
LCALL   SEND_CMD_T
LJMP    LOOP

;=====
;=====
;=====
; rotina de inicialização da porta serial
;=====

INIT_SER:

CLR     RI           ; zera flags de interrupção de recepção
CLR     TI           ; e de transmissão

SETB    ADCON.7      ; serial com gerador de baud rate interna(timer1)
MOV     TMOD, #020H  ; timer 1 modo 2 (auto recarga de th1 em tl1)
MOV     TH1, #0F3H   ; ajuste para 9600 bauds
MOV     PCON, #080H

MOV     SCON, #00H
SETB    SM1          ; modo1: 10 bits(startb+8b+stopb)
SETB    REN          ; habilita recepção

```

```

MOV     TCON, #00H
SETB    TR1           ; dispara timer1

SETB    P5.6          ; ajusta pc para transmissão
CLR     P5.7          ; e o host para recepção

MOV     R0, #02H
LCALL   WRLCDCOM
LCALL   PRTLCD
DB      "PORTA SERIAL OK!", 0

CLR     RI
CLR     TI
SETB    ES
CLR     SERFLAG
RET

```

```

=====
;
; subrotina init_timer2
; ==> inicializa o timer 2 como auto reload modo 0
;      gera uma interrupcao por overflow a cada 50 ms
;      valor de recarga do registrador do timer2: 3cb0h
;
=====

```

INIT_TIMER2:

```

MOV     APONT_MEM_AQ_HI, #0A0H
MOV     APONT_MEM_AQ_LO, #00H
MOV     SEGUNDOS, #00H
MOV     MINUTOS, #00H
MOV     HORAS, #00H
MOV     DIAS, #01H
SETB    SEGFLAG
SETB    MINFLAG
SETB    HORFLAG
SETB    DIAFLAG

MOV     CRCH, #03CH
MOV     CRCL, #0B0H
MOV     TH2, #03CH
MOV     TL2, #0B0H
MOV     T2CON, #00010001H ; timer2 auto reload modo0

MOV     R0, #02H
LCALL   WRLCDCOM
LCALL   PRTLCD
DB      "RELOGIO OK !!", 0
SETB    EAL
SETB    ET2           ; habilita int. do timer2
RET

```

```

=====
;
; rotina de interrupcao do timer 2 (TIMER2_INT)
; ==> a cada interrupção o contadores de interrupções é incrementado.
;      a cada 20 interrupções o contador de interrupções é zerado e o
;      contador de segundos é incrementado.
;      a cada 60 segundos o contador de segundos é zerado e o
;      contador de minutos é incrementado.
;      a cada 60 minutos o contador de minutos é zerado e o
;      contador de horas é incrementado.
;      a cada 24 segundos o contador de horas é zerado e o
;      contador de dias é incrementado.
;      a cada segundo o display é atualizado.
;
=====

```

```

TIMER2_INT:
CLR      TF2
PUSH     PSW
MOV      PSW, #0      ; seleciona banco de registradores 0
PUSH     ACC
PUSH     DPH
PUSH     DPL

INC      CONT_INT
MOV      A, CONT_INT

CJNE     A, #20, EHFIM
MOV      CONT_INT, #00
INC      SEGUNDOS
INC      CONTSEC
SETB     SEGFLAG
MOV      R0, SEGUNDOS
CJNE     R0, #60, EHFIM
MOV      SEGUNDOS, #00
INC      MINUTOS
SETB     MINFLAG
MOV      R0, MINUTOS
CJNE     R0, #60, EHFIM
MOV      MINUTOS, #00
INC      HORAS
SETB     HORFLAG
MOV      R0, HORAS
CJNE     R0, #24, EHFIM
MOV      HORAS, #00
INC      DIAS
SETB     DIAFLAG
EHFIM:
JB       SERFLAG, JAFOI
JNB      SEGFLAG, JAFOI
JNB      DIAFLAG, PHORAS
CLR      DIAFLAG
MOV      R2, #00
MOV      R3, DIAS
LCALL    LONG_BINASC
MOV      R0, #0CDH
LCALL    WRLCDCOM
MOV      A, R6
MOV      R0, A
LCALL    WRLCDDATA
MOV      A, R5
MOV      R0, A
LCALL    WRLCDDATA
MOV      A, R4
MOV      R0, A
LCALL    WRLCDDATA

PHORAS:
JNB      HORFLAG, PMINUTOS
CLR      HORFLAG
MOV      R0, #0C0H
LCALL    WRLCDCOM
MOV      A, HORAS
LCALL    PRT10

PMINUTOS:
JNB      MINFLAG, PSEGUNDOS
CLR      MINFLAG
MOV      R0, #0C3H
LCALL    WRLCDCOM
MOV      A, MINUTOS
LCALL    PRT10

MOV      A, MINUTOS
CJNE     A, #10, PSEGUNDOS
MOV      A, HORAS
CJNE     A, #12, PMINUTOS
SETB     SFUNFLAG
PSEGUNDOS:

```

```

CLR      SEGFLAG
MOV      R0, #0C6H
LCALL    WRLCDCOM
MOV      A, SEGUNDOS
LCALL    PRT10
JAFOI:
POP      DPL
POP      DPH
POP      ACC
POP      PSW
RETI

```

```

;=====
; rotina LIMPA_MSG → limpa linha de mensagem do display
;=====

```

```

LIMPA_MSG:
MOV      R0, #02H
LCALL    WRLCDCOM
LCALL    PRTLCD
DB       " ", 0
MOV      R0, #02H
LCALL    WRLCDCOM
RET

```

```

;=====
; rotina RAMXTESTE → esta rotina realiza um teste de leitura
; e escrita na memória ram externa do módulo.
; ao final do teste é mostrado o resultado no LCD
;=====

```

```

RAMXTEST:
PUSH     PSW
PUSH     ACC
MOV      PSW, #00H
MOV      DPTR, #0000H
ENCHE:
MOV      A, #0AAH
MOVX     @DPTR, A
MOV      R4, DPL
MOV      R5, DPH
INC      DPTR
CJNE     R4, #0FFH, ENCHE
CJNE     R5, #0FFH, ENCHE
LCALL    SDELAY
MOV      R0, #08CH
LCALL    WRLCDCOM
MOV      R0, #02AH
LCALL    WRLCDDATA
MOV      DPTR, #0000H

```

```

ESVAZIA:
MOVX     A, @DPTR
MOV      R4, DPL
MOV      R5, DPH
INC      DPTR

CJNE     A, #0AAH, ERRO
CJNE     R4, #0FFH, ESVAZIA
CJNE     R5, #0FFH, ESVAZIA
LCALL    SDELAY
MOV      R0, #08DH
LCALL    WRLCDCOM
MOV      R0, #02AH
LCALL    WRLCDDATA
MOV      DPTR, #0000H

```

```

ENCHE1:
MOV      A, #055H
MOVX     @DPTR, A
MOV      R4, DPL
MOV      R5, DPH

```



```

INC      DPTR
CJNE     R4, #0FFH, ENCHE1
CJNE     R5, #0FFH, ENCHE1
LCALL    SDELAY
MOV      R0, #08EH
LCALL    WRLCDCOM
MOV      R0, #02AH
LCALL    WRLCDDATA
MOV      DPTR, #0000H

```

```

ESVAZIA1:
MOVX     A, @DPTR
MOV      R4, DPL
MOV      R5, DPH
INC      DPTR
CJNE     A, #055H, ERRO
CJNE     R4, #0FFH, ESVAZIA1
CJNE     R5, #0FFH, ESVAZIA1
LCALL    SDELAY
MOV      R0, #08FH
LCALL    WRLCDCOM
MOV      R0, #02AH
LCALL    WRLCDDATA
MOV      RXST, #4BH
LCALL    SDELAY
LJMP     TESTADO

```

```

ERRO:
MOV      RXST, #46H
LCALL    SDELAY
MOV      R0, #080H
LCALL    WRLCDCOM
LCALL    PRTLCD
DB        "FALHA RAM EXT. !",0

```

```

TESTADO:
LCALL    SDELAY
MOV      R0, #080H
LCALL    WRLCDCOM
LCALL    PRTLCD
DB        "RAM EXTERNA OK!!",0
POP      ACC
POP      PSW
RET

```

```

;=====
;=====

```

```

;=====
;=====

```

```

; rotina LONG_BINASC
; conversao de número binário de 2 (dois) bytes para o seu código
; ascii correspondente;
;
;   entrada:          saida:
;   r2 - hbyte        r7 - milhar
;   r3 - lbyte        r6 - centena
;                   r5 - dezena
;                   r4 - unidade
;
;=====

```

```

LONG_BINASC:
MOV      R7, #030H
MOV      R6, #030H
MOV      R5, #030H
MOV      R4, #030H
MOV      SOBRA, #00H

```

```

TEST_MIL:
CLR      C
MOV      A, R2
CJNE     A, #03H, TEST
SJMP     TEST4

```

```

TEST:

```

```

        JNC      MAIOR_IGUAL
        SJMP     CENTENA
TEST4:
        CLR      C
        MOV      A, R3
        CJNE     A, #0E8H, TEST5
        SJMP     MAIOR_IGUAL
TEST5:
        JNC      MAIOR_IGUAL
        SJMP     CENTENA
MAIOR_IGUAL:
        CLR      C
        MOV      A, R2
        SUBB     A, #03H
        MOV      AUX1, A
        CLR      C
        MOV      A, R3
        SUBB     A, #0E8H
        MOV      R3, A
        JNC      SEGUE
        DEC      R2
SEGUE:
        INC      R7
        SJMP     TEST_MIL
CENTENA:
        CLR      C
        MOV      A, R2
        CJNE     A, #00H, MAIOR_IGUAL1
        MOV      A, R3
        CJNE     A, #00H, TSOBRA
        LJMP     UNIDADE
TSOBRA:
        ADD      A, SOBRA
        MOV      SOBRA, #00H
        MOV      R3, A
        CLR      C
        CJNE     A, #00H, VAIEM
        INC      R2
        SJMP     CENTENA
VAIEM:
        CLR      C
        CJNE     A, #064H, TEST1
        SJMP     MAIOR_IGUAL1
TEST1:
        JNC      MAIOR_IGUAL1
        SJMP     DEZENA
MAIOR_IGUAL1:
        CLR      C
        MOV      A, R3
        SUBB     A, #064H
        JNC      SEGUE3
        DEC      R2
SEGUE3:
        MOV      R3, A
        CJNE     A, #00H, SEGUE1
        MOV      A, R2
        CLR      C
        CJNE     A, #00H, SEGUE4
        INC      R6
        LJMP     UNIDADE
SEGUE4:
        DEC      R2
        MOV      R2, #OFFH
        INC      SOBRA
SEGUE1:
        INC      R6
        SJMP     CENTENA
DEZENA:
        CLR      C
        MOV      A, R3
        CJNE     A, #0AH, TEST2
        SJMP     MAIOR_IGUAL2
TEST2:

```

```

        JNC      MAIOR_IGUAL2
        SJMP     UNIDADE
MAIOR_IGUAL2:
        CLR      C
        SUBB     A, #0AH
        MOV      R3, A
        INC      R5
        SJMP     DEZENA
UNIDADE:
        MOV      A, R3
        ADD      A, R4
        MOV      R4, A

        RET

```

```

=====
; rotina INIT_TECL → executa inicialização do teclado, cuja operação
; é feita através da interrupção int0 do 80535.
=====

```

```

INIT_TECL:
        MOV      R0, #02H
        LCALL    WRLCDCOM
        LCALL    PRTLCD
        DB       "TECLADO OK !!",0

        CLR      TFUNFLAG
        CLR      HFUNFLAG
        CLR      SFUNFLAG

        MOV      P4, #0F0H
        SETB     ITO
        SETB     EAL
        SETB     EXO
        CLR      NUMBFLAG

        RET

```

```

=====
; rotina INT_0 → executada quando da ocorrência de uma interrupção int0
; ou seja, quando uma tecla do teclado for acionada.
; esta rotina tem por finalidade identificar a tecla acionada.
; se for uma tecla correspondente a uma função de teclado
; válida, seu respectivo flag será setado.
=====

```

```

INT_0:
        CLR      EXO
        PUSH     PSW
        PUSH     ACC
        PUSH     DPH
        PUSH     DPL
        LCALL    DELAY
        JNB      TFUNFLAG, VAILOGO
        DEC      CONTDIG
VAILOGO:
        MOV      LINE, P4
        LCALL    SCAN
        DJNZ     ZERO_COUNTER, REJECT

TECACIO:
        MOV      ADDRESS, ZERO_BIT
        MOV      P4, #0FH
        XCH      A, ZERO_BIT
        MOV      A, P4
        SWAP     A
        MOV      LINE, A
        XCH      A, ZERO_BIT

```

```

LCALL    SCAN
DJNZ     ZERO_COUNTER, REJECT
XCH      A, ZERO_BIT
SWAP     A
ORL      ADDRESS, A
XCH      A, ZERO_BIT
SETB     TEC_VAL
LCALL    LIMPA_MSG
LCALL    MOSTRA_TECLA
LCALL    SDELAY

```

```

REJECT:
MOV      P4, #0F0H
SETB     EX0

POP      DPL
POP      DPH
POP      ACC
POP      PSW
RETI

```

```

; debounce do teclado

```

```

DELAY:
MOV      R7, #03AH
RODA:
MOV      R6, #0FFH
RODA1:
DJNZ     R6, RODA1
DJNZ     R7, RODA
RET

```

```

; varredura do teclado

```

```

SCAN:
MOV      A, LINE
MOV      ZERO_COUNTER, #00
JB       ACC.4, ONE
INC      ZERO_COUNTER
MOV      ZERO_BIT, #01
ONE:
JB       ACC.5, TWO
INC      ZERO_COUNTER
MOV      ZERO_BIT, #02
TWO:
JB       ACC.6, THREE
INC      ZERO_COUNTER
MOV      ZERO_BIT, #03
THREE:
JB       ACC.7, FOUR
INC      ZERO_COUNTER
MOV      ZERO_BIT, #04
FOUR:
RET

```

```

; mostrar tecla acionada ou retorna o numero teclado

```

```

MOSTRA_TECLA:
PUSH     PSW
PUSH     DPH
PUSH     DPL
PUSH     ACC

MOV      R7, ADDRESS
JB       NUMBFLAG, NUMBERS
LF1:
CJNE     R7, #014H, LF2
LCALL    FUNCAO_H
LJMP     TCHAW1

```



```

LF2:
  CJNE    R7, #024H, LF3
  LCALL   FUNCAO_T
  LJMP    TCHAW1

LF3:
  CJNE    R7, #034H, LF4
  LCALL   FUNCAO_L
  LJMP    TCHAW1

LF4:
  CJNE    R7, #044H, LNOTE
  LCALL   FUNCAO_D
  LJMP    TCHAW1

LNOTE:
  LCALL   PRTLCD
  DB      "TECLA INVALIDA!",0
  LJMP    TCHAW1

NUMBERS:
  LCALL   PRTLCD
  DB      "NUMERO ",0
  CJNE    R7, #011H, EHD0IS
  MOV     NUMERO, #1
  LJMP    TCHAW

EHD0IS:
  CJNE    R7, #012H, EHTRES
  MOV     NUMERO, #2
  LJMP    TCHAW

EHTRES:
  CJNE    R7, #013H, EHQUATRO
  MOV     NUMERO, #3
  LJMP    TCHAW

EHQUATRO:
  CJNE    R7, #021H, EHCINCO
  MOV     NUMERO, #4
  LJMP    TCHAW

EHCINCO:
  CJNE    R7, #022H, EHSEIS
  MOV     NUMERO, #5
  LJMP    TCHAW

EHSEIS:
  CJNE    R7, #023H, EHSETE
  MOV     NUMERO, #6
  LJMP    TCHAW

EHSETE:
  CJNE    R7, #031H, EHOITO
  MOV     NUMERO, #7
  LJMP    TCHAW

EHOITO:
  CJNE    R7, #032H, EHNOVE
  MOV     NUMERO, #8
  LJMP    TCHAW

EHNOVE:
  CJNE    R7, #033H, EHZERO
  MOV     NUMERO, #9
  LJMP    TCHAW

EHZERO:
  CJNE    R7, #042H, NO_NUMB
  MOV     NUMERO, #0
  LJMP    TCHAW

NO_NUMB:
  LCALL   PRTLCD
  DB      "INVALIDO!",0
  CLR     TFUNFLAG
  CLR     NUMBFLAG
  SJMP    TCHAW1

TCHAW:
  MOV     A, NUMERO
  ADD     A, #030H
  MOV     R0, A
  LCALL   WRLCDDATA

```

```
TCHAW1:
    POP     ACC
    POP     DPL
    POP     DPH
    POP     PSW
    RET
```

```
; subrotina para a função H
```

```
FUNCAO_H:
    MOV     R0, #02H
    LCALL   WRLCDCOM
    LCALL   PRTLCD
    DB      " FUNCAO H ...",0
    SETB    HFUNFLAG
    RET
```

```
; subrotina para a função T
```

```
FUNCAO_T:
    MOV     R0, #02H
    LCALL   WRLCDCOM
    LCALL   PRTLCD
    DB      " FUNCAO T ...",0
    SETB    TFUNFLAG
    SETB    NUMBFLAG
    MOV     CONTDIG, #7
    RET
```

```
; subrotina para a função L (não implementada)
```

```
FUNCAO_L:
    MOV     R0, #02H
    LCALL   WRLCDCOM
    LCALL   PRTLCD
    DB      " FUNCAO L ...",0
    RET
```

```
; subrotina para a função D → realiza o refresh do LCD
```

```
FUNCAO_D:
    MOV     R0, #02H
    LCALL   WRLCDCOM
    LCALL   PRTLCD
    DB      " FUNCAO D ...",0
    LCALL   SDELAY
    MOV     R0, #0C0H
    LCALL   WRLCDCOM
    MOV     A, HORAS
    LCALL   PRT10
    MOV     R0, #03AH
    LCALL   WRLCDDATA
    MOV     A, MINUTOS
    LCALL   PRT10
    MOV     R0, #03AH
    LCALL   WRLCDDATA
    MOV     A, SEGUNDOS
    LCALL   PRT10
```

```

LCALL    PRTLCD
DB       " DIA ",0
MOV      R2, #00H
MOV      R3, DIAS
LCALL    LONG_BINASC
MOV      A, R6
MOV      R0, A
LCALL    WRLCDDATA
MOV      A, R5
MOV      R0, A
LCALL    WRLCDDATA
MOV      A, R4
MOV      R0, A
LCALL    WRLCDDATA
RET

```

PRT10:

; esta subrotina mostra no display um número decimal de dois dígitos
; correspondente ao byte presente no acumulador

```

MOV      B, #10
DIV      AB
SWAP     A
ADD      A, B
LCALL    BINASC
MOV      R0, A
LCALL    WRLCDDATA
MOV      A, R2
MOV      R0, A
LCALL    WRLCDDATA
RET

```

;subrotina SDELAY → executa uma espera de tempo de 1 segundo

```

SDELAY:
PUSH     ACC
MOV      A, #091H

SD_OLP:
INC      A
LCALL    MDELAY
LCALL    MDELAY
LCALL    MDELAY
LCALL    MDELAY
LCALL    MDELAY
LCALL    MDELAY
LCALL    MDELAY
LCALL    MDELAY
NOP
NOP
NOP
NOP
NOP
JNZ      SD_OLP
MOV      A, #033H
SD_ILP:
DJNZ     ACC, SD_ILP
POP      ACC
RET

MDELAY:
PUSH     ACC
MOV      A, #0A6H
MD_OLP:
INC      A

```

```

NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
JNZ      MD_OLP
NOP
POP      ACC
RET

```

```

; programa finalreq.asm

```

```

; ==> Este programa contém as rotinas referentes a comunicação
; serial dos módulos de aquisição. Mediante um sinal no bar-
; ramento, é gerada uma interrupção e o módulo captura o frame
; (desde que o endereço de destino seja o seu), e executa uma
; ação de acordo com o campo comando do frame.

```

```

;*****
;          wagner roberto balsani
;      mestrado em engenharia eletrica
;      universidade de sao paulo
;      escola de engenharia de sao carlos
;      orientador: prof. dr. paulo estevao cruvinel
;*****

```

```

; programa rec_quadro.asm;

```

```

; esta rotina, a partir de uma interrupção gerada pela presença
; de informação na porta serial, recebe, monta e confere o quadro
; de comunicação, e se necessário encaminha para rotina de resposta
; ao comando presente no quadro de comunicação.

```

```

; esta rotina é específica para módulos de aquisição de dados.

```

```

; o quadro utilizado pelo protocolo de comunicação é
; constituído dos seguintes campos:

```

IQ	EN	EO	ED	CM	CB	DADOS	CS	FQ
----	----	----	----	----	----	-------	----	----

```

; onde:

```

```

; IQ : início de quadro —> 1 byte (*)
; EN : enquiry —> 1 byte (?)hospedeiro para modulo de aquisição
;      (@) modulo de aquis. para hospedeiro
; EO : endereço de origem —> 2 bytes
; ED : endereço de destino —> 2 bytes
; CM : comando —> 1 byte
; ST : status —> 1 byte( presente no sentido modulo—>host)
; CB : contador de bytes de dados —> 2 bytes (00 a ff)

```



```

; DADOS : campo de dados  --> ate 255 bytes
; CS : check-sum          --> 2 bytes (00 a ff)
; FQ : fim de quadro      --> 1 byte (#)

```

```

=====
; constantes

```

```

INICIO_QUADRO      EQU 02AH      ; '*'
PC_ENQ             EQU 021H      ; '!'
HOST_ENQ           EQU 03FH      ; '?'
MODUL_ENQ          EQU 040       ; '@'
FIM_QUADRO         EQU 023H      ; '#'
PC_ADDR_HI         EQU 046H      ; 'F' endereço específico
PC_ADDR_LO         EQU 043H      ; 'C' para o pc
HOST_ADDR_HI       EQU 046H      ; 'F' endereço específico
HOST_ADDR_LO       EQU 041H      ; 'A' para o host

```

```

=====
; programa principal

```

```

ORG 1800H

```

```

REC_QUA_FROM_PC:

```

```

PUSH    PSW
PUSH    ACC
PUSH    DPL
PUSH    DPH

CLR      IEN0.4      ; desabilita interrupção via serial
SETB    SERFLAG      ; seta flag de aviso de que esta havendo comunicação
SETB    RS1          ; seleciona banco de reg. número 3

```

```

MOV      R0, #CH_SUM_CALC      ; inicializa check-sum
MOV      @R0, #00H
MOV      AUX1, #00H
MOV      R0, #R_INIT_DATA_ADDR_HI
MOV      @R0, #09FH
MOV      R0, #R_INIT_DATA_ADDR_LO
MOV      @R0, #080H

```

```

LCALL    GET_CHAR
CJNE     A, #INICIO_QUADRO, GO_FIM_REC_I

```

```

LCALL    GET_CHAR
CJNE     A, #PC_ENQ, GO_FIM_REC_I      ; verifica campo EN

```

```

LCALL    GET_CHAR
CJNE     A, #PC_ADDR_HI, GO_FIM_REC_I
MOV      R0, #CH_SUM_CALC
ADD      A, @R0      ; verifica o campo EO
MOV      @R0, A

```

```

LCALL    GET_CHAR
CJNE     A, #PC_ADDR_LO, GO_FIM_REC_I
MOV      R0, #CH_SUM_CALC
ADD      A, @R0
MOV      @R0, A

```

```

LCALL    GET_CHAR
CJNE     A, #HOST_ADDR_HI, GO_FIM_REC_I
MOV      R0, #CH_SUM_CALC
ADD      A, @R0

```

```

MOV    @R0, A

LCALL  GET_CHAR
CJNE   A, #HOST_ADDR_LO, GO_FIM_REC_I
MOV    R0, #CH_SUM_CALC
ADD    A, @R0
MOV    @R0, A
SJMP   SEGUE_I

GO_FIM_REC_I:
LJMP   FIM_REC_I

SEGUE_I:
LCALL  GET_CHAR
MOV    R0, #R_COMANDO      ; recebe o campo CM
MOV    @R0, A

MOV    R0, #CH_SUM_CALC    ;
ADD    A, @R0
MOV    @R0, A              ; e inclui no check-sum

LCALL  GET_CHAR
MOV    R0, #CONT_DADOS_4
MOV    @R0, A
MOV    R0, #CH_SUM_CALC
MOV    AUX1, @R0
ADD    A, AUX1
MOV    AUX1, A

LCALL  GET_CHAR
MOV    R0, #CONT_DADOS_3
MOV    @R0, A
ADD    A, AUX1
MOV    AUX1, A

LCALL  GET_CHAR
MOV    R0, #CONT_DADOS_2
MOV    @R0, A
ADD    A, AUX1              ; recebe o campo CB
MOV    AUX1, A              ;

LCALL  GET_CHAR              ; e inclui no check-sum
MOV    R0, #CONT_DADOS_1
MOV    @R0, A
ADD    A, AUX1              ; se este campo for diferente
MOV    R0, #CH_SUM_CALC      ; de zero
MOV    @R0, A

LCALL  CONT_DADOS_ASCTOHEX    ;

MOV    R0, #CONT_DADOS_LO    ; entao eh acessada rotina
MOV    A, @R0
CJNE   A, #00H, TEM_DADOS_I   ; para montar o campo DADOS
MOV    R0, #CONT_DADOS_HI
MOV    A, @R0
CJNE   A, #00H, TEM_DADOS_I

SJMP   SEGUE3_I

TEM_DADOS_I:
LCALL  MONT_DATA_FIELD

SEGUE3_I:
LCALL  GET_CHAR
MOV    R0, #CHECK_SUM_HI
MOV    @R0, A
LCALL  GET_CHAR              ; recebe o campo CS
MOV    R0, #CHECK_SUM_LO
MOV    @R0, A

LCALL  GET_CHAR
CJNE   A, #FIM_QUADRO, FIM_REC_I

```

```

LCALL    CH_SUM_ASCTOHEX      ; e compara com o
MOV      R0, #CH_SUM_CALC
MOV      AUX1, @R0
CJNE     A, AUX1, FIM_REC_I

LCALL    IDENT_CMND0          ; do campo CM

FIM_REC_I:

POP      DPH
POP      DPL
POP      ACC
POP      PSW

SETB     IEN0.4                ; habilita interrupção
CLR      SERFLAG              ; zera flag de aviso de serial

RETI                                ; retorna para o programa original

```

```

;-----
; subrotina ident_cmndo ==> identifica o comando recebido e acessa
; a subrotina correspondente ao mesmo.
;-----

```

```

IDENT_CMND0:
MOV      R0, #R_COMANDO
MOV      A, @R0
CJNE     A, #041H, NOT_A      ; teste para comando A (44h)
LCALL    COMANDO_A
RET

NOT_A:
CJNE     A, #042H, NOT_B
LCALL    COMANDO_B
RET

NOT_B:
CJNE     A, #043H, NOT_C
LCALL    COMANDO_C

NOT_C:
RET

```

```

;-----
; subrotina mont_data_field ==> monta o campo de dados recebido no frame
;-----

```

```

MONT_DATA_FIELD:
MOV      R0, #R_INIT_DATA_ADDR_HI
MOV      A, @R0
MOV      DPH, A
MOV      R0, #CONT_DADOS_HI
ADD      A, @R0
MOV      R0, #R_FIM_DATA_ADDR_HI
MOV      @R0, A

MOV      R0, #R_INIT_DATA_ADDR_LO
MOV      A, @R0
MOV      DPL, A
MOV      R0, #CONT_DADOS_LO
ADD      A, @R0
MOV      R0, #R_FIM_DATA_ADDR_LO
MOV      @R0, A

MOV      R0, #CH_SUM_CALC
MOV      AUX1, @R0

```

```

GO_REC:
    LCALL    GET_CHAR
    MOVX     @DPTR, A
    ADD      A, AUX1
    MOV      AUX1, A
    INC      DPTR
    MOV      R0, #R_FIM_DATA_ADDR_LO
    MOV      A, @R0
    CJNE     A, DPL, AINDA
    MOV      R0, #R_FIM_DATA_ADDR_HI
    MOV      A, @R0
    CJNE     A, DPH, AINDA
    MOV      R0, #CH_SUM_CALC
    MOV      @R0, AUX1
    RET
AINDA:
    SJMP     GO_REC

```

```

;-----
COMANDO_A:
    MOV      R0, #S_COMANDO
    MOV      @R0, #041H
    MOV      DEST_ADDR_HI, #046H
    MOV      DEST_ADDR_LO, #043H
    MOV      R0, #S_INIT_DATA_ADDR_HI
    MOV      @R0, #080H
    MOV      R0, #S_INIT_DATA_ADDR_LO
    MOV      @R0, #00H
    MOV      R0, #S_CONT_DADOS_HI
    MOV      @R0, #0EH
    MOV      R0, #S_CONT_DADOS_LO
    MOV      @R0, #040H
    SETB     P5.7                ; ajusta host para transmissão
    CLR      P5.6                ; ajusta pc para recepção
    LCALL    SEND_PACK
    CLR      P5.7                ; ajusta host para recepção
    SETB     P5.6                ; ajusta pc para transmissão
    RET

```

```

;-----
COMANDO_B:
    MOV      R0, #S_COMANDO
    MOV      @R0, #042H
    MOV      DEST_ADDR_HI, #046H
    MOV      DEST_ADDR_LO, #043H
    MOV      R0, #S_INIT_DATA_ADDR_HI
    MOV      @R0, #0D0H
    MOV      R0, #S_INIT_DATA_ADDR_LO
    MOV      @R0, #00H
    MOV      R0, #S_CONT_DADOS_HI
    MOV      @R0, #0EH
    MOV      R0, #S_CONT_DADOS_LO
    MOV      @R0, #040H
    SETB     P5.7
    CLR      P5.6
    LCALL    SEND_PACK
    CLR      P5.7
    SETB     P5.6

    RET

```

```

;-----
COMANDO_C:
    MOV      R0, #S_COMANDO
    MOV      @R0, #043H
    MOV      DEST_ADDR_HI, #046H
    MOV      DEST_ADDR_LO, #043H

```



```

MOV    R0, #S_INIT_DATA_ADDR_HI
MOV    @R0, #0F0H
MOV    R0, #S_INIT_DATA_ADDR_LO
MOV    @R0, #00H
MOV    R0, #S_CONT_DADOS_HI
MOV    @R0, #0EH
MOV    R0, #S_CONT_DADOS_LO
MOV    @R0, #040H
SETB   P5.7
CLR    P5.6
LCALL  SEND_PACK
CLR    P5.7
SETB   P5.6

```

```
RET
```

```

=====
; subrotina send_pack ==> envia pela serial um pacote de informação
; em resposta a um pedido do hospedeiro.
=====

```

```
SEND_PACK:
```

```

MOV    R0, #S_CHECK_SUM
MOV    @R0, #00H
MOV    AUX1, #00H

MOV    A, #INICIO_QUADRO
LCALL  SEND_CHAR      ; envia caractere de início de quadro

MOV    A, #HOST_ENQ
LCALL  SEND_CHAR      ; envia caractere de enquiry

MOV    A, #HOST_ADDR_HI
LCALL  SEND_CHAR
MOV    R0, #S_CHECK_SUM
MOV    AUX1, @R0
ADD    A, AUX1
MOV    AUX1, A
MOV    A, #HOST_ADDR_LO
LCALL  SEND_CHAR      ; envia endereço de origem
                          ; (endereço do módulo)
ADD    A, AUX1
MOV    AUX1, A

MOV    A, DEST_ADDR_HI
LCALL  SEND_CHAR
ADD    A, AUX1
MOV    AUX1, A
MOV    A, DEST_ADDR_LO
LCALL  SEND_CHAR      ; envia endereço de destino
                          ; (end. do hospedeiro)
ADD    A, AUX1
MOV    AUX1, A

MOV    R0, #S_COMANDO
MOV    A, @R0
LCALL  SEND_CHAR
ADD    A, AUX1
MOV    AUX1, A
                          ; envia comando

LCALL  S_CONT_DADOS_HEXTOASC
MOV    R0, #S_CONT_DADOS_4
MOV    A, @R0
LCALL  SEND_CHAR
ADD    A, AUX1
MOV    AUX1, A
MOV    R0, #S_CONT_DADOS_3
MOV    A, @R0
LCALL  SEND_CHAR
ADD    A, AUX1
MOV    AUX1, A
MOV    R0, #S_CONT_DADOS_2
MOV    A, @R0

```

```

LCALL    SEND_CHAR
ADD      A, AUX1
MOV      AUX1, A
MOV      R0, #S_CONT_DADOS_1
MOV      A, @R0
LCALL    SEND_CHAR
ADD      A, AUX1
MOV      R0, #S_CHECK_SUM
MOV      @R0, A

MOV      R0, #S_CONT_DADOS_LO
MOV      A, @R0
CJNE     A, #00, CHAMASEND
MOV      R0, #S_CONT_DADOS_HI
MOV      A, @R0
CJNE     A, #00, CHAMASEND
SJMP     CHAMACS

CHAMASEND:
LCALL    SEND_DATA_FIELD
CHAMACS:
LCALL    CHECK_SUM_HEXTOASC

MOV      R0, #S_CHECK_SUM_HI
MOV      A, @R0
LCALL    SEND_CHAR
MOV      R0, #S_CHECK_SUM_LO
MOV      A, @R0
LCALL    SEND_CHAR

MOV      A, #FIM_QUADRO
LCALL    SEND_CHAR
CLR      TI
RET

```

```

=====
; subrotina send_data_field ==> envia pacote de dados requisitado
; pelo hospedeiro.
=====

```

```

SEND_DATA_FIELD:
MOV      R0, #S_CONT_DADOS_HI
MOV      A, @R0
MOV      R0, #S_INIT_DATA_ADDR_HI
ADD      A, @R0
MOV      R0, #S_FIM_DATA_ADDR_HI
MOV      @R0, A

MOV      R0, #S_CONT_DADOS_LO
MOV      A, @R0
MOV      R0, #S_INIT_DATA_ADDR_LO
ADD      A, @R0
MOV      R0, #S_FIM_DATA_ADDR_LO
MOV      @R0, A

SEGSTART:
MOV      R0, #S_INIT_DATA_ADDR_HI
MOV      A, @R0
MOV      DPH, A
MOV      R0, #S_INIT_DATA_ADDR_LO
MOV      A, @R0
MOV      DPL, A
MOV      R0, #S_CHECK_SUM
MOV      AUX1, @R0

GO_SEND:

```

```

MOVX    A, @DPTR
LCALL   SEND_CHAR
ADD     A, AUX1
MOV     AUX1, A
INC     DPTR

MOV     R0, #S_FIM_DATA_ADDR_LO
MOV     A, @R0
CJNE    A, DPL, AINDATEM
MOV     R0, #S_FIM_DATA_ADDR_HI
MOV     A, @R0
CJNE    A, DPH, AINDATEM

MOV     R0, #S_CHECK_SUM
MOV     @R0, AUX1
RET

```

```

AINDATEM:
SJMP    GO_SEND

```

```

=====
; subrotina get_char ==> recebe caractere presente na porta serial
; colocando-o no acumulador.
=====

```

```

GET_CHAR:
MOV     CONTSEC, #00H
GET_CHAR1:
MOV     A, CONTSEC
CJNE    A, #5, GOGET
SETB    TIMEOUTFLAG
RET

GOGET:
JNB     RI, GET_CHAR1
MOV     A, SBUF
MOV     SBUF, A
CLR     RI
RET

```

```

=====
; subrotina send_char ==> envia caractere presente no acumulador através
; da porta serial.
=====

```

```

SEND_CHAR:
MOV     SBUF, A
FICA:
JNB     TI, FICA
CLR     TI
RET

```

```

=====
; subrotina cont_dados_asctohex ==> converte os dois caracteres ascii
; correspondentes a 0-9 ou a-f do campo
; CD num único byte composto por dois
; nibbles de valores correspondentes.
=====

```

```

CONT_DADOS_ASCTOHEX:
PUSH    ACC
PUSH    PSW

MOV     R0, #CONT_DADOS_4
MOV     A, @R0

```

```

LCALL    ASCBIN
SWAP     A
MOV      B, A
MOV      R0, #CONT_DADOS_3
MOV      A, @R0
LCALL    ASCBIN
ORL      A, B
MOV      R0, #CONT_DADOS_HI
MOV      @R0, A
MOV      R0, #CONT_DADOS_2
MOV      A, @R0
LCALL    ASCBIN
SWAP     A
MOV      B, A
MOV      R0, #CONT_DADOS_1
MOV      A, @R0
LCALL    ASCBIN
ORL      A, B
MOV      R0, #CONT_DADOS_LO
MOV      @R0, A

```

```

POP      PSW
POP      ACC

```

```

RET

```

```

=====
: subrotina ch_sum_asctohex ==> converte os dois caracteres ascii
:   correspondentes a 0-9 ou a-f do campo
:   CS num único byte composto de dois
:   nibbles de valores correspondentes.
=====

```

CH_SUM_ASCTOHEX:

```

MOV      R0, #CHECK_SUM_HI
MOV      A, @R0
LCALL    ASCBIN
SWAP     A
MOV      B, A
MOV      R0, #CHECK_SUM_LO
MOV      A, @R0
LCALL    ASCBIN
ORL      A, B
MOV      R0, #CH_SUM_HEX
MOV      @R0, A
RET

```

```

=====
: subrotina s_cont_dados_hextoasc ==> converte os dois bytes contadores
:   de dados (high e low) nos quatro
:   códigos ascii correspondentes
:   (pares 4 e 3, e 2 e 1).
=====

```

S_CONT_DADOS_HEXTOASC:

```

MOV      R0, #S_CONT_DADOS_HI
MOV      A, @R0
LCALL    BINASC
MOV      R0, #S_CONT_DADOS_4
MOV      @R0, A
MOV      A, R2
MOV      R0, #S_CONT_DADOS_3
MOV      @R0, A
MOV      R0, #S_CONT_DADOS_LO
MOV      A, @R0
LCALL    BINASC
MOV      R0, #S_CONT_DADOS_2
MOV      @R0, A

```



```

MOV    A, R2
MOV    R0, #S_CONT_DADOS_1
MOV    @R0, A
RET

```

```

=====
; subrotina check_sum_hextoasc ==> converte o byte check_sum do quadro
; a ser enviado no par de código ascii
; correspondente ( high e low )
=====

```

```

CHECK_SUM_HEXTOASC:
MOV    R0, #S_CHECK_SUM
MOV    A, @R0
LCALL  BINASC
MOV    R0, #S_CHECK_SUM_HI
MOV    @R0, A
XCH    A, R2
MOV    R0, #S_CHECK_SUM_LO
MOV    @R0, A
RET

```

```

=====
; subrotina ascbn ==> esta rotina pega o caractere ascii presente
; no acumulador e o converte num numero binário de 4 bits que ,
; recolocado no acumulador.
=====

```

```

ASCBIN:
ADD    A, #0D0H          ; se char < 30 --> erro
JNC    NOTNUM
CLR    C                  ; verifica se o char eh 0-9
ADD    A, #0F6H          ; e ajusta-o, caso contrario
JC     HEXTRY             ; salta
ADD    A, #0AH
RET

```

```

HEXTRY:
CLR    ACC.5              ; verifica se o char eh a-f
CLR    C
ADD    A, #0F9H           ; se for ajusta-o
JNC    NOTNUM
CLR    C                  ; caso contrario existe
ADD    A, #0FAH
JC     NOTNUM             ; um erro
ANL    A, #0FH
RET

```

```

NOTNUM:
RET

```

```

=====
; subrotina binasc ==> esta rotina pega o conteudo do acumulador e o
; converte em dois numeros hexadecimais ascii correspondentes
; o ascii correspondente ao nibble menos significativo vai para r2
; e o ascii correspondente ao nibble mais significativo vai para o acc.
=====

```

```

BINASC:
MOV    R2, A
ANL    A, #0FH
ADD    A, #0F6H

```

```

JNC      NOADJ1
ADD      A, #07H
NOADJ1:
ADD      A, #03AH

XCH      A, R2
SWAP     A
ANL      A, #0FH
ADD      A, #0F6H
JNC      NOADJ2
ADD      A, #07H
NOADJ2:
ADD      A, #03AH
RET

```

```

source_hi      equ 0a0h
source_lo      equ 0a1h
dest_hi        equ 0a2h
dest_lo        equ 0a3h
bound_hi       equ 0a4h
bound_lo       equ 0a5h

```

```

; constantes

```

```

inicio_quadro  equ 02ah ; ""
pc_enq         equ 021h ; "!"
host_enq       equ 03fh ; "?"
modul_enq      equ 040h ; "@"
fim_quadro     equ 023h ; "#"
pc_addr_hi     equ 030h ; 'f' endereço específico
pc_addr_lo     equ 031h ; 'c' para o PC
host_addr_hi   equ 046h ; 'f' endereço específico
host_addr_lo   equ 041h ; 'a' para o host

```

```

send_pack      equ 198dh
get_char       equ 1a60h
cont_dados_asctohex equ 1a7dh
mont_data_field equ 18e5h
ch_sum_asctohex equ 1aaeh
wrlcdcom       equ 2881h
wrlcddata      equ 2875h
prtlcd         equ 291eh
limpa_msg      equ 1232h
long_binasc    equ 131bh
sdelay         equ 15e2h
ascbin         equ 1aech
binasc         equ 1b08h

```

```

ORG 2000H

```

```

SND CMD_H:
PUSH      PSW
PUSH      ACC
PUSH      DPH
PUSH      DPL

```

```

CLR      IEN0.4
SETB     RS1
SETB     SERFLAG

MOV      R0, #S_COMANDO
MOV      @R0, #048H          ; 'H'
MOV      R0, #S_CONT_DADOS_HI
MOV      @R0, #00H
MOV      R0, #S_CONT_DADOS_LO
MOV      @R0, #00H
MOV      DEST_ADDR_HI, #030H ; '0'
MOV      DEST_ADDR_LO, #031H ; '1'
MOV      R0, #R_INIT_DATA_ADDR_HI
MOV      @R0, #09FH
MOV      R0, #R_INIT_DATA_ADDR_LO
MOV      @R0, #080H

VOLTA_QUADRO:
CLR      P5.6                ; ajusta pc para recepção
SETB     P5.7                ; e host para transmissão
LCALL    SEND_PACK
CLR      P5.7                ; ajusta host para recepção
LCALL    REC_QUADRO
JB       TIMEOUTFLAG, VOLTA_QUADRO
JB       ERCSFLAG, VOLTA_QUADRO
JB       ERFRMFLAG, VOLTA_QUADRO
LCALL    LIMPA_MSG
LCALL    LIMPA_MSG
MOV      R0, #R_COMANDO
MOV      A, @R0
CJNE     A, #04BH, EHERRORAM
LCALL    PRTLCD
DB       "MODULO 1 OK!!",0
SJMP     SHOWT

EHERRORAM:
CJNE     A, #46H, VOLTA_QUADRO
LCALL    PRTLCD
DB       "MOD1 FALHA XRAM!",0

SHOWT:
LCALL    SDELAY
LCALL    LIMPA_MSG
LCALL    LIMPA_MSG
LCALL    LIMPA_MSG
MOV      R0, #R_INIT_DATA_ADDR_LO
MOV      A, @R0
MOV      DPL, A
MOV      R0, #R_INIT_DATA_ADDR_HI
MOV      A, @R0
MOV      DPH, A
MOV      R0, #08DH
LCALL    WRLCD COM

MOVX     A, @DPTR
LCALL    ASCBIN
MOV      R2, A
INC      DPTR
MOVX     A, @DPTR
LCALL    ASCBIN
SWAP     A
ORL      A, R2
SWAP     A
MOV      R2, #00H
MOV      R3, A
LCALL    LONG_BINASC
MOV      A, R6
MOV      R0, A
LCALL    WRLCDDATA
MOV      A, R5
MOV      R0, A
LCALL    WRLCDDATA
MOV      A, R4
MOV      R0, A
LCALL    WRLCDDATA

```

```

MOV     R0, #080H
LCALL   WRLCDCOM
INC     DPTR
MOVBX   A, @DPTR
LCALL   ASCBIN
MOV     R2, A

INC     DPTR
MOVBX   A, @DPTR
LCALL   ASCBIN
SWAP    A
ORL     A, R2
SWAP    A
MOV     B, #0AH
DIV     AB
SWAP    A
ADD     A, B
LCALL   BINASC
MOV     R0, A
LCALL   WRLCDDATA
MOV     A, R2
MOV     R0, A
LCALL   WRLCDDATA

MOV     R0, #03AH
LCALL   WRLCDDATA

MAISMIN:
INC     DPTR
MOVBX   A, @DPTR
LCALL   ASCBIN
MOV     R2, A

INC     DPTR
MOVBX   A, @DPTR
LCALL   ASCBIN
SWAP    A
ORL     A, R2
SWAP    A
MOV     B, #0AH
DIV     AB
SWAP    A
ADD     A, B
LCALL   BINASC
MOV     R0, A
LCALL   WRLCDDATA
MOV     A, R2
MOV     R0, A
LCALL   WRLCDDATA

MOV     DEST_ADDR_HI, #031H    ; '1'
MOV     DEST_ADDR_LO, #032H    ; '2'
VOLTA_QUADRO1:
SETB    P5.7                    ; ajusta host para transmissão
LCALL   SEND_PACK
CLR     P5.7                    ; ajusta host para recepção
LCALL   REC_QUADRO
JB      TIMEOUTFLAG, VOLTA_QUADRO1
JB      ERCSFLAG, VOLTA_QUADRO1
JB      ERFRMFLAG, VOLTA_QUADRO1
LCALL   SDELAY
LCALL   LIMPA_MSG
LCALL   LIMPA_MSG
MOV     R0, #R_COMANDO
MOV     A, @R0
CJNE    A, #04BH, EHERRORAM1
LCALL   PRTLCD
DB      "MODULO 2 OK !!",0
LCALL   SDELAY

```



```

    LCALL    LIMPA_MSG
    LCALL    LIMPA_MSG
    SJMP     SHOWT2
EHERRORAM1:
    CJNE     A, #046H, VOLTA_QUADRO1
    LCALL    PRTLCD
    DB       "MOD2 FALHA XRAM!".0

SHOWT2:
    MOV      R0, #R_INIT_DATA_ADDR_LO
    MOV      A, @R0
    MOV      DPL, A
    MOV      R0, #R_INIT_DATA_ADDR_HI
    MOV      A, @R0
    MOV      DPH, A
    MOV      R0, #08DH
    LCALL    WRLCDCOM

    MOVX     A, @DPTR
    LCALL    ASCBIN
    MOV      R2, A
    INC      DPTR
    MOVX     A, @DPTR
    LCALL    ASCBIN
    SWAP     A
    ORL      A, R2
    SWAP     A
    MOV      R2, #00H
    MOV      R3, A
    LCALL    LONG_BINASC
    MOV      A, R6
    MOV      R0, A
    LCALL    WRLCDDATA
    MOV      A, R5
    MOV      R0, A
    LCALL    WRLCDDATA
    MOV      A, R4
    MOV      R0, A
    LCALL    WRLCDDATA

    MOV      R0, #080H
    LCALL    WRLCDCOM
    INC      DPTR
    MOVX     A, @DPTR
    LCALL    ASCBIN
    MOV      R2, A

    INC      DPTR
    MOVX     A, @DPTR
    LCALL    ASCBIN
    SWAP     A
    ORL      A, R2
    SWAP     A
    MOV      B, #0AH
    DIV      AB
    SWAP     A
    ADD      A, B
    LCALL    BINASC
    MOV      R0, A
    LCALL    WRLCDDATA
    MOV      A, R2
    MOV      R0, A
    LCALL    WRLCDDATA

    MOV      R0, #03AH
    LCALL    WRLCDDATA

    INC      DPTR
    MOVX     A, @DPTR
    LCALL    ASCBIN
    MOV      R2, A

```

```

INC      DPTR
MOVX     A, @DPTR
LCALL    ASCBIN
SWAP     A
ORL      A, R2
SWAP     A
MOV      B, #0AH
DIV      AB
SWAP     A
ADD      A, B
LCALL    BINASC
MOV      R0, A
LCALL    WRLCDDATA
MOV      A, R2
MOV      R0, A
LCALL    WRLCDDATA

NEXTMOD1:
MOV      DEST_ADDR_HI, #032H    ; '2'
MOV      DEST_ADDR_LO, #033H    ; '3'
VOLTA_QUADRO2:
SETB     P5.7                    ; ajusta host para transmissão
LCALL    SEND_PACK
CLR      P5.7                    ; ajusta host para recepção
LCALL    REC_QUADRO
JB       TIMEOUTFLAG, VOLTA_QUADRO2
JB       ERCSFLAG, VOLTA_QUADRO2
JB       EFRMFLAG, VOLTA_QUADRO2
LCALL    SDELAY
LCALL    LIMPA_MSG
LCALL    LIMPA_MSG
MOV      R0, #R_COMANDO
MOV      A, @R0

CJNE     A, #04BH, EHERRORAM2
LCALL    PRTLCD
DB       "MODULO 3 OK !!",0
LCALL    SDELAY
LCALL    LIMPA_MSG
LCALL    LIMPA_MSG
SJMP     SHOWT3

EHERRORAM2:
CJNE     A, #046H, VOLTA_QUADRO2
LCALL    PRTLCD
DB       "MOD3 FALHA XRAM!",0

SHOWT3:
MOV      R0, #R_INIT_DATA_ADDR_LO
MOV      A, @R0
MOV      DPL, A
MOV      R0, #R_INIT_DATA_ADDR_HI
MOV      A, @R0
MOV      DPH, A
MOV      R0, #08DH
LCALL    WRLCDCOM

MOVX     A, @DPTR
LCALL    ASCBIN
MOV      R2, A
INC      DPTR
MOVX     A, @DPTR
LCALL    ASCBIN
SWAP     A
ORL      A, R2
SWAP     A
MOV      R2, #00H
MOV      R3, A
LCALL    LONG_BINASC
MOV      A, R6
MOV      R0, A

```

```

LCALL  WRLCDDATA
MOV    A, R5
MOV    R0, A
LCALL  WRLCDDATA
MOV    A, R4
MOV    R0, A
LCALL  WRLCDDATA

```

```

MOV    R0, #080H
LCALL  WRLCDCOM
INC    DPTR
MOVX   A, @DPTR
LCALL  ASCBIN
MOV    R2, A

```

```

INC    DPTR
MOVX   A, @DPTR
LCALL  ASCBIN
SWAP   A
ORL    A, R2
SWAP   A
MOV    B, #0AH
DIV    AB
SWAP   A
ADD    A, B
LCALL  BINASC
MOV    R0, A
LCALL  WRLCDDATA
MOV    A, R2
MOV    R0, A
LCALL  WRLCDDATA

```

```

MOV    R0, #03AH
LCALL  WRLCDDATA

```

```

INC    DPTR
MOVX   A, @DPTR
LCALL  ASCBIN
MOV    R2, A

```

```

INC    DPTR
MOVX   A, @DPTR
LCALL  ASCBIN
SWAP   A
ORL    A, R2
SWAP   A
MOV    B, #0AH
DIV    AB
SWAP   A
ADD    A, B
LCALL  BINASC
MOV    R0, A
LCALL  WRLCDDATA
MOV    A, R2
MOV    R0, A
LCALL  WRLCDDATA

```

```

POP    DPL
POP    DPH
POP    ACC
POP    PSW

```

```

SETB   P5.6           ; ajusta pc para transmissao
CLR    SERFLAG
SETB   ES

```

```

RET

```

SND_CMD_T:

```

PUSH PSW
PUSH ACC
PUSH DPH
PUSH DPL

CLR IEN0.4
SETB SERFLAG
SETB RS1

MOV R0, #S_COMANDO
MOV @R0, #054H
MOV R0, #S_CONT_DADOS_HI
MOV @R0, #00H
MOV R0, #S_CONT_DADOS_LO
MOV @R0, #06H
MOV R0, #S_INIT_DATA_ADDR_HI
MOV @R0, #09FH
MOV R0, #S_INIT_DATA_ADDR_LO
MOV @R0, #00H

MOV DEST_ADDR_HI, #046H ; 'F' endereço
MOV DEST_ADDR_LO, #046H ; 'F' universal

SETB P5.7 ; ajusta host para transmissão
CLR P5.6 ; ajusta pc para recepção
LCALL SEND_PACK
CLR P5.7 ; ajusta host para recepção
SETB P5.6 ; ajusta pc para transmissão

POP DPL
POP DPH
POP ACC
POP PSW
CLR SERFLAG
SETB ES

RET

```

REC_QUADRO:

```

CLR ERCSFLAG
CLR ERFRMFLAG
CLR TIMEOUTFLAG
MOV R0, #CH_SUM_CALC ; inicializa check-sum
MOV @R0, #00H
MOV AUX1, #00H

LCALL GET_CHAR
JB TIMEOUTFLAG, GO_FIM_REC2
CJNE A, #INICIO_QUADRO, GO_FIM_REC

LCALL GET_CHAR
JB TIMEOUTFLAG, GO_FIM_REC2
CJNE A, #MODUL_ENQ, GO_FIM_REC ; verifica campo EN

LCALL GET_CHAR
JB TIMEOUTFLAG, GO_FIM_REC2
CJNE A, DEST_ADDR_HI, GO_FIM_REC
MOV R0, #CH_SUM_CALC
ADD A, @R0 ; verifica o campo EO
MOV @R0, A
LCALL GET_CHAR
JB TIMEOUTFLAG, GO_FIM_REC2
CJNE A, DEST_ADDR_LO, GO_FIM_REC
MOV R0, #CH_SUM_CALC
ADD A, @R0
MOV @R0, A

LCALL GET_CHAR
JB TIMEOUTFLAG, GO_FIM_REC2

```



```

CJNE    A, #HOST_ADDR_HI, GO_FIM_REC
MOV     R0, #CH_SUM_CALC
ADD     A, @R0
MOV     @R0, A

LCALL   GET_CHAR
JB      TIMEOUTFLAG, GO_FIM_REC2
CJNE    A, #HOST_ADDR_LO, GO_FIM_REC
MOV     R0, #CH_SUM_CALC
ADD     A, @R0
MOV     @R0, A
SJMP    SEGUE

GO_FIM_REC:
SETB    ERFRMFLAG

GO_FIM_REC2:
LJMP    FIM_REC

SEGUE:
LCALL   GET_CHAR
JB      TIMEOUTFLAG, GO_FIM_REC2
MOV     R0, #R_COMANDO           ; recebe o campo CM
MOV     @R0, A

MOV     R0, #CH_SUM_CALC
ADD     A, @R0
MOV     @R0, A                   ; e inclui no check-sum

LCALL   GET_CHAR
JB      TIMEOUTFLAG, GO_FIM_REC2
MOV     R0, #CONT_DADOS_4
MOV     @R0, A
MOV     R0, #CH_SUM_CALC
MOV     AUX1, @R0
ADD     A, AUX1
MOV     AUX1, A

LCALL   GET_CHAR
JB      TIMEOUTFLAG, GO_FIM_REC2
MOV     R0, #CONT_DADOS_3
MOV     @R0, A
ADD     A, AUX1
MOV     AUX1, A

LCALL   GET_CHAR
JB      TIMEOUTFLAG, GO_FIM_REC2
MOV     R0, #CONT_DADOS_2
MOV     @R0, A
ADD     A, AUX1                 ; recebe o campo CB
MOV     AUX1, A

LCALL   GET_CHAR                 ; e inclui no check-sum
JB      TIMEOUTFLAG, GO_FIM_REC2
MOV     R0, #CONT_DADOS_1
MOV     @R0, A
ADD     A, AUX1                 ; se este campo for diferente
MOV     R0, #CH_SUM_CALC        ; de zero
MOV     @R0, A

LCALL   CONT_DADOS_ASCTOHEX

MOV     R0, #CONT_DADOS_LO       ; entao é acessada rotina
MOV     A, @R0
CJNE    A, #00H, TEM_DADOS       ; para montar o campo dados
MOV     R0, #CONT_DADOS_HI
MOV     A, @R0
CJNE    A, #00H, TEM_DADOS

SJMP    SEGUE3

TEM_DADOS:
LCALL   MONT_DATA_FIELD

```

```

SEGUE3:
  LCALL  GET_CHAR
  JB     TIMEOUTFLAG, FIM_REC
  MOV    R0, #CHECK_SUM_HI
  MOV    @R0, A
  LCALL  GET_CHAR           ; recebe o campo CS
  MOV    R0, #CHECK_SUM_LO
  MOV    @R0, A

  LCALL  GET_CHAR
  JB     TIMEOUTFLAG, FIM_REC
  CJNE   A, #FIM_QUADRO, FIM_REC1
  SJMP   PULULA

FIM_REC1:
  SETB   ERFRMFLAG
  SJMP   FIM_REC

PULULA:
  LCALL  SDELAY

  LCALL  CH_SUM_ASCTOHEX    ; e compara com o
  MOV    R0, #CH_SUM_CALC
  MOV    AUX1, @R0
  CJNE   A, AUX1, ERRO_CHSUM ; conteúdo do campo CM

FIM_REC:
  CLR    TI
  CLR    RI

  RET                    ; retorna para o programa original

ERRO_CHSUM:
  SETB   ERCSFLAG
  SJMP   FIM_REC

;-----

SND_CMD_S:

  PUSH   PSW
  PUSH   ACC
  PUSH   DPH
  PUSH   DPL

  CLR    IEN0.4
  SETB   RS1
  SETB   SERFLAG

  MOV    R0, #S_COMANDO
  MOV    @R0, #053H          ; 'S'
  MOV    R0, #S_CONT_DADOS_HI
  MOV    @R0, #00H
  MOV    R0, #S_CONT_DADOS_LO
  MOV    @R0, #00H
  MOV    DEST_ADDR_HI, #030H ; '0'
  MOV    DEST_ADDR_LO, #031H ; '1'
  MOV    R0, #R_INIT_DATA_ADDR_HI
  MOV    @R0, #0A0H
  MOV    R0, #R_INIT_DATA_ADDR_LO
  MOV    @R0, #00H

VOLTA_QUADROSS:
  SETB   P5.7                ; ajusta host para transmissão
  CLR    P5.6                ; ajusta pc para recepção
  LCALL  SEND_PACK
  CLR    P5.7                ; ajusta host para recepção
  LCALL  REC_QUADRO
  JB     TIMEOUTFLAG, VOLTA_QUADROSS
  JB     ERCSFLAG, VOLTA_QUADROSS
  JB     ERFRMFLAG, VOLTA_QUADROSS
  LCALL  LIMPA_MSG
  LCALL  LIMPA_MSG
  MOV    R0, #R_COMANDO

```

```

MOV     A, @R0
CJNE    A, #053H, VOLTA_QUADROSS
LCALL   PRTLCD
DB      "DESCARGA MOD1 !!",0

LCALL   SDELAY
MOV     R0, #SOURCE_HI
MOV     @R0, #0A0H
MOV     R0, #SOURCE_LO
MOV     @R0, #00H
MOV     A, DIAS_B
MOV     B, #2
DIV     AB
MOV     A, B
CJNE    A, #0, DIAIMPAR
MOV     R0, #DEST_HI
MOV     @R0, #0B7H
MOV     R0, #DEST_LO
MOV     @R0, #020H
SJMP    AJUSTADO
DIAIMPAR:
MOV     R0, #DEST_HI
MOV     @R0, #0B0H
MOV     R0, #DEST_LO
MOV     @R0, #00H
AJUSTADO:
LCALL   TRANSFER
LCALL   LIMPA_MSG
LCALL   PRTLCD
DB      "BACK-UP MODULO 1!",0

MOV     DEST_ADDR_HI, #031H      ; '1'
MOV     DEST_ADDR_LO, #032H      ; '2'
MOV     R0, #R_INIT_DATA_ADDR_HI
MOV     @R0, #0C0H
MOV     R0, #R_INIT_DATA_ADDR_LO
MOV     @R0, #00H

VOLTA_QUADROSS1:
SETB    P5.7                    ; ajusta host para transmissão
LCALL   SEND_PACK
CLR     P5.7                    ; ajusta host para recepção
LCALL   REC_QUADRO
JB       TIMEOUTFLAG, VOLTA_QUADROSS1
JB       ERCSFLAG, VOLTA_QUADROSS1
JB       ERFRMFLAG, VOLTA_QUADROSS1
LCALL   SDELAY
LCALL   LIMPA_MSG
LCALL   LIMPA_MSG
MOV     R0, #R_COMANDO
MOV     A, @R0
CJNE    A, #053H, VOLTA_QUADROSS1
LCALL   PRTLCD
DB      "DESCARGA MOD2 !!",0

LCALL   SDELAY
MOV     R0, #SOURCE_HI
MOV     @R0, #0C0H
MOV     R0, #SOURCE_LO
MOV     @R0, #00H
MOV     A, DIAS_B
MOV     B, #2
DIV     AB
MOV     A, B
CJNE    A, #0, DIAIMPAR1
MOV     R0, #DEST_HI
MOV     @R0, #0D7H
MOV     R0, #DEST_LO
MOV     @R0, #020H
SJMP    AJUSTADO1
DIAIMPAR1:

```

```

MOV     R0, #DEST_HI
MOV     @R0, #0D0H
MOV     R0, #DEST_LO
MOV     @R0, #00H
AJUSTADO1:
LCALL   TRANSFER
LCALL   LIMPA_MSG
LCALL   PRTLCD
DB      "BACK-UP MODULO 2",0

MOV     DEST_ADDR_HI, #032H           ; '2'
MOV     DEST_ADDR_LO, #033H           ; '3'
MOV     R0, #R_INIT_DATA_ADDR_HI
MOV     @R0, #0E0H
MOV     R0, #R_INIT_DATA_ADDR_LO
MOV     @R0, #00H

VOLTA_QUADROSS2:
SETB    P5.7                          ; ajusta host para transmissão
LCALL   SEND_PACK
CLR     P5.7                          ; ajusta host para recepção
LCALL   REC_QUADRO
JB      TIMEOUTFLAG, VOLTA_QUADROSS2
JB      ERCSFLAG, VOLTA_QUADROSS2
JB      ERFRMFLAG, VOLTA_QUADROSS2
LCALL   SDELAY
LCALL   LIMPA_MSG
LCALL   LIMPA_MSG
MOV     R0, #R_COMANDO
MOV     A, @R0

CJNE    A, #053H, VOLTA_QUADROSS2
LCALL   PRTLCD
DB      "DESCARGA MOD3 !!",0

LCALL   SDELAY
MOV     R0, #SOURCE_HI
MOV     @R0, #0E0H
MOV     R0, #SOURCE_LO
MOV     @R0, #00H
MOV     A, DIAS_B
MOV     B, #2
DIV     AB
MOV     A, B
CJNE    A, #0, DIAIMPAR2
MOV     R0, #DEST_HI
MOV     @R0, #0F7H
MOV     R0, #DEST_LO
MOV     @R0, #020H
SJMP    AJUSTADO2
DIAIMPAR2:
MOV     R0, #DEST_HI
MOV     @R0, #0F0H
MOV     R0, #DEST_LO
MOV     @R0, #00H
AJUSTADO2:
LCALL   TRANSFER
LCALL   LIMPA_MSG
LCALL   PRTLCD
DB      "BACK-UP MODULO 3",0

LCALL   SDELAY

POP     DPL
POP     DPH
POP     ACC
POP     PSW

SETB    P5.6                          ; ajusta pc para transmissão
CLR     SERFLAG
CLR     TI

```



```

CLR    RI
SETB   ES

RET

```

TRANSFER:

```

SETB   RS0
SETB   RS1

MOV     R0, #SOURCE_HI
MOV     A, @R0
MOV     R4, A
MOV     R0, #SOURCE_LO
MOV     A, @R0
MOV     R5, A
MOV     R0, #DEST_HI
MOV     A, @R0
MOV     R6, A
MOV     R0, #DEST_LO
MOV     A, @R0
MOV     R7, A
MOV     R0, #BOUND_LO
MOV     @R0, #00
MOV     R0, #BOUND_HI
MOV     @R0, #00

```

TRANSFER1:

```

MOV     R0, #BOUND_LO
MOV     A, @R0
CJNE    A, #020H, GO_TRANSFER
MOV     R0, #BOUND_HI
MOV     A, @R0
CJNE    A, #07H, GO_TRANSFER
CLR     RS0
CLR     RS1
RET

```

GO_TRANSFER:

```

MOV     DPH, R4
MOV     DPL, R5
MOVX    A, @DPTR
MOV     DPH, R6
MOV     DPL, R7
MOVX    @DPTR, A
INC     R5
CJNE    R5, #00H, MENOS
INC     R4

```

MENOS:

```

INC     R7
CJNE    R7, #00H, MENOS1
INC     R6

```

MENOS1:

```

MOV     R0, #BOUND_LO
INC     @R0
CJNE    @R0, #00H, VAITRANSF
MOV     R0, #BOUND_HI
INC     @R0

```

VAITRANSF:

```

LJMP    TRANSFER1

```

=====

=====

```

;      programa principal
;-----

; constantes

LCD_DATA EQU 090H      ; Port 1 é usada para dados
LCD_RS   EQU 0B4H      ; p3.4 LCD RS (Register Select)
LCD_RW   EQU 0B5H      ; p3.5 LCD RW (Read/Write)
LCD_E    EQU 0F8H      ; p5.0 LCD E (Enable)

; instruções de sistema

CONFIG EQU 38H          ; 2 linhas, matriz caractere de 5 por 7
ENTRYMODE EQU 6         ; desloca o cursor, não desloca a linha

; instruções de controle de cursor

OFFCUR EQU 0CH
LINECUR EQU 0EH
BLINKCUR EQU 0DH
COMBNCUR EQU 0FH
HOMECUR EQU 02H
SHLFCUR EQU 10H
SHRTCUR EQU 14H

; instruções de controle de display

CLRDSPEQU 01H
OFFDSP EQU 0AH
ONDSP EQU 0EH
SHLFDSP EQU 18H
SHRTDSP EQU 1CH

; endereço de subrotinas

MDELAY EQU 1611H

;-----
ORG 2800H

LCALL RESETLCD ; duplicate power on reset
LCALL INITLCD  ; initialize fonts, #lines, Entry Mode

MOV A, #2
MOV B, #0
LCALL PLACECUR

LCALL PRTLCD
DB "00:00:00 DIA 000",0

RET

; fim do programa principal

;-----
; subrotinas
;-----

;=====
; subrotina CRSHLF
; esta subrotina desloca o cursor para a esquerda.
; o número de posições a ser deslocado é colocado no acumulador.
; input : accumulator indicates number of notches to shift

```

```

; output : none
;=====
;
CRSHLF:
    JZ     RET_SCL
    MOV    R0, #SHLFCUR          ; control word for left shift
    LCALL  WRLCDCOM
    DEC    ACC
    SJMP   CRSHLF
RET_SCL:
    RET

;=====
; subrotina CRSHRT
; esta subrotina desloca o cursor para a direita.
; o número de posições a ser deslocado é colocado no acumulador.
; input  : accumulator indicates number of notches to shift
; output : none
;=====
;
CRSHRT:
    JZ     RET_SCR
    MOV    R0, #SHRTCUR          ; control word for right shift
    LCALL  WRLCDCOM
    DEC    ACC
    SJMP   CRSHRT
RET_SCR:
    RET

;=====
; subrotina PLACECUR
; esta rotina ajusta a posição do cursor.
; a posição do cursor é colocada no registrador b
; a posição do cursor começa em 0, o acumulador dá o número da linha.
; input  : accumulator indicates line number (1, 2)
;         : b register holds the cursor position
; output : none
;=====
;
PLACECUR:
    DEC    ACC                   ; acc=0 for line=1
    JNZ    LINE2                 ; if acc=0 then first line
    MOV    A, B
    ADD    A, #080H              ; construct control word for line 1
    SJMP   SETCUR
LINE2:
    MOV    A, B
    ADD    A, #0C0H              ; construct control word for line 2
SETCUR:
    MOV    R0, A                 ; place control word
    LCALL  WRLCDCOM              ; issue command
    RET

;=====
; subrotina SETCGRAM
; this routine sets the Character Generator RAM address. the
; character [0..7] is placed in the accumulator. the font row
; is placed in the b register. the instructions 40h to 7Fh
; selects the CGRAM addresses 0 to 3Fh (0 to 63 decimal).
;
; the 5x7 character font is assumed. that is, each character
; location contains a matrix of 40 dots arranged in 8 rows of
; 5 dots each. the 7 rows are for the characters and the 8-th
; row for the underline cursor. CGRAM address 0 corresponds
; to the first (font) row of character 0, address 7 corresponds
; to the 8-th (last) row of character 0, and address 3Fh
; corresponds to the 8-th row of character 7.
;
; input  : accumulator indicates character code [0..7]
;         : b register holds font row [0..7]

```

```

; output : none
;=====
;
SETCGRAM:
  PUSH    B
  MOV     B, #8
  MUL     A, B           ; multiply character number by 8
  POP     B             ; b holds row number
  ADD     A, B           ; a holds CGRAM address
  ADD     A, #40H        ; convert to instruction

  MOV     R0, A          ; place instruction
  LCALL   WRLCDCOM       ; issue command
  RET

```

```

;=====
; subrotina DSPSHLF
; esta rotina desloca o conteúdo do LCD para a esquerda.
; o número de caracteres a ser deslocado é colocado no acumulador.
; input : accumulator indicates number of notches to shift
; output : none
;=====
;
DSPSHLF:
  JZ     RET_SDL
  MOV    R0, #SHLFDSP    ; control word for left shift
  LCALL  WRLCDCOM
  DEC    A
  SJMP   DSPSHLF
RET_SDL:
  RET

```

```

;=====
; subrotina DSPSHRT
; esta rotina desloca o conteúdo do LCD para a esquerda.
; o número de caracteres a ser deslocado é colocado no acumulador.
; input : accumulator indicates number of notches to shift
; output : none
;=====
;
DSPSHRT:
  JZ     RET_SDR
  MOV    R0, #SHRTDSP    ; control word for right shift
  LCALL  WRLCDCOM
  DEC    A
  SJMP   DSPSHRT
RET_SDR:
  RET

```

```

;=====
; subrotina WRLCDDATA
; escreve uma palavra de dados para o LCD
; o dado deve ser colocado em R0 no programa de chamada
;=====
;
WRLCDDATA:
  CLR    LCD_E           ; select send Data
  SETB   LCD_RS          ; select write operation
  CLR    LCD_RW
  MOV    LCD_DATA, R0    ; load command into port
  LCALL  PULSEWAIT       ; pulse the Enable line
  RET

```

```

;=====
; subrotina WRLCDCOM
; escreve uma palavra de comando para o LCD
; LCD_RS, LCD_RW, e R0 devem ser ajustados no programa de chamada.

```



```

;-----
WRLCDCOM:
CLR    LCD_E
CLR    LCD_RS      ; select send Command
CLR    LCD_RW      ; select Write operation
MOV    LCD_DATA, R0 ; load command into port
LCALL  PULSEEWAIT  ; pulse the Enable line
RET

;=====
; subroutine rdLCDstat
; reads the status byte (Busy Flag and Address Counter)
; the status byte is returned in the accumulator
; input  : none
; output : status byte in accumulator
;-----
RDLCDSTAT:
MOV    R0, #OFFH      ; prepare port for input
CLR    LCD_E
CLR    LCD_RS
SETB   LCD_RW
SETB   LCD_E          ; pulse the Enable line
MOV    A, R0
CLR    LCD_E
RET

;=====
; subrotina PULSEEWAIT
; gera um pulso positivo na linha de habilitação (enable line) do LCD.
; espera que o Busy Flag seja zerado para retornar.
; input  : none
; output : none
; destroys : LCD_RW, LCD_RS, LCD_DATA
;-----
PULSEEWAIT:
CLR    LCD_E
SETB   LCD_E          ; pulse the Enable line
CLR    LCD_E

MOV    LCD_DATA, #OFFH ; prepare port for input
SETB   LCD_RW          ; prepare RW for read operation

PUSH   ACC             ; save accumulator contents
PEW:
INC    B
SETB   LCD_E          ; start the Enable pulse
MOV    A, LCD_DATA    ; read status
CLR    LCD_E          ; end Enable pulse
LCALL  MDELAY
LCALL  MDELAY
LCALL  MDELAY
LCALL  MDELAY

JB     ACC.7, PEW      ; loop while Busy Flag is set
POP    ACC             ; restore accumulator
RET

;=====
; subrotina INITLCD - inicializa o LCD
;-----
INITLCD:
CLR    LCD_RS      ; LCD Register Select line
CLR    LCD_RW      ; Read / Write line
CLR    LCD_E       ; Enable line

MOV    R0, #38H    ; Function Set - 2 lines 5x7 fonts

```

```

LCALL    WRLCDCOM

MOV      R0, #0EH          ; display ON
LCALL    WRLCDCOM

MOV      R0, #06H          ; set Entry Mode
LCALL    WRLCDCOM          ; increment cursor to right, no display shift

MOV      R0, #01H          ; clear display, home cursor
LCALL    WRLCDCOM

RET

```

```

=====
; subrotina RESETLCD - faz o reset do LCD
;
;
=====

```

```

RESETLCD:
CLR LCD_RS          ; LCD Register Select line
CLR LCD_RW          ; Read / Write line
CLR LCD_E           ; Enable line

MOV      R0, #30H    ; step 1
SETB    LCD_E        ; start Enable pulse
CLR      LCD_E        ; end Enable pulse
LCALL    MDELAY
LCALL    MDELAY
LCALL    MDELAY
LCALL    MDELAY

MOV      LCD_DATA, #30H ; step 2
SETB    LCD_E        ; start Enable pulse
CLR      LCD_E        ; end Enable pulse
LCALL    MDELAY

MOV      LCD_DATA, #30H ; step 3
SETB    LCD_E        ; start Enable pulse
CLR      LCD_E        ; end Enable pulse
LCALL    MDELAY

MOV      R0, #CONFIG  ; function set
LCALL    WRLCDCOM

MOV      R0, #08H     ; display off
LCALL    WRLCDCOM

MOV      R0, #01H     ; clear display, home cursor
LCALL    WRLCDCOM

MOV      R0, #ENTRYMODE ; set ENTRY MODE
LCALL    WRLCDCOM      ; increment cursor, do not shift display

RET

```

```

=====
; subrotina PRTLCD
; pega a cadeia imediatamente a seguir a chamada e mostra no LCD.
; a cadeia é lida com a instrução mov a, @a+dptr
; a cadeia deve terminar com o caractere 0.
; após a terminação esta rotina salta para a instrução imediatamente após a cadeia.
;
; input  : none
; output : none
; destroys : acc, dptr
;
=====

```

```

PRTLCD:
POP      DPH          ; pop return address into dptr
POP      DPL

PRTNEXT:

```

```
CLR      A                ; set offset = 0
MOVC     A, @A+DPTR       ; get chr from code memory
CJNE     A, #0, CHROK     ; if chr = 0 then return
SJMP     RETPRTLCD
CHROK:
MOV      R0, A
LCALL    WRLCDDATA        ; send character
INC      DPTR             ; point at next character
LJMP     PRTNEXT          ; loop till end of string
RETPRTLCD:
MOV      A, #1H           ; point to instruction after string
JMP      @A+DPTR          ; return with a jump instruction

; =====
; fim do programa
```

Apêndice C


```
rec_quadro equ 1800h
```

```
*****
*                                     *
*      wagner roberto balsani        *
*      mestrado em engenharia eletrica *
*      universidade de sao paulo      *
*      escola de engenharia de sao carlos *
*      orientador: prof. dr. paulo estevao cruvinel *
*                                     *
*****
```

```
=====
;
; variaveis.h
;
; alocao de variaveis na ram interna
```

```
; bytes:
```

CONT_INT	EQU 060H	; contador de interrupcoes do timer 2
SEGUNDOS	EQU 061H	; contador de segundos
MINUTOS	EQU 062H	; contador de minutos
HORAS	EQU 063H	; contador de horas
DIAS	EQU 064H	; contador de dias (byte baixo)
CONTSEC	EQU 065H	; contador de segundos utilizado p/ time-out
APONT_MEM_AQ_HI	EQU 066H	; apontador de escrita de dados de aq. (byte alto)
APONT_MEM_AQ_LO	EQU 067H	; apontador de escrita de dados de aq. (byte baixo)
SENSOR	EQU 068H	; utilizado durante envio de dados ao host
RXST	EQU 069H	; registrador de status da ram externa
INIT_ADDR_HI	EQU 06AH	
INIT_ADDR_LO	EQU 06BH	
CONT_SUM	EQU 06CH	; var. que controla o numero de parcelas da soma
SOMA_H	EQU 06DH	; byte mais significativo da soma
SOMA_L	EQU 06EH	; byte menos significativo da soma
DIVISOR	EQU 06FH	; no c lc. da m,dia cont,m n.o de aq. de cada sensor
DIVIDENDO_H	EQU 070H	; valor obtido na soma dos valores das aq.
DIVIDENDO_L	EQU 071H	
QUOCIENTE_H	EQU 072H	
QUOCIENTE_L	EQU 073H	
RESTO_H	EQU 074H	; vari veis auxiliares da operac/ao de divis/ao
RESTO_L	EQU 075H	
QUOCIENTE	EQU 076H	; resultados finais obtidos
RESTO	EQU 077H	; na divis/ao
AUX1	EQU 078H	

```
;
; vari veis utilizadas para emiss/ao de dados via serial
```

S_CHECK_SUM	EQU 080H	:
S_COMANDO	EQU 081H	:
S_CONT_DADOS_4	EQU 082H	:
S_CONT_DADOS_3	EQU 083H	:
S_CONT_DADOS_2	EQU 084H	:
S_CONT_DADOS_1	EQU 085H	:
S_CONT_DADOS_HI	EQU 086H	:
S_CONT_DADOS_LO	EQU 087H	:
S_CHECK_SUM_HI	EQU 088H	:
S_CHECK_SUM_LO	EQU 089H	:
CH_SUM_CALC	EQU 08AH	:
CONT_DADOS_4	EQU 08BH	:
CONT_DADOS_3	EQU 08CH	:
CONT_DADOS_2	EQU 08DH	:
CONT_DADOS_1	EQU 08EH	:
CONT_DADOS_HI	EQU 08FH	:

```

CONT_DADOS_LO      EQU 090H
CH_SUM_HEX         EQU 091H
R_COMANDO          EQU 092H
CHECK_SUM_HI       EQU 093H
CHECK_SUM_LO       EQU 094H

S_INIT_DATA_ADDR_HI EQU 095H
S_INIT_DATA_ADDR_LO EQU 096H
R_INIT_DATA_ADDR_HI EQU 097H
R_INIT_DATA_ADDR_LO EQU 098H

IS_CONT_DADOS_HI   EQU 099H
IS_CONT_DADOS_LO   EQU 09AH

; bits

SEGFLAG            EQU 01H      ; sinalizador de incremento de segundos
MINFLAG            EQU 02H      ; sinalizador de incremento de minutos
HORFLAG            EQU 03H      ; sinalizador de incremento de horas
DIAFLAG            EQU 04H      ; sinalizador de incremento de dias
SERFLAG            EQU 05H      ; sinalizador de comunicac#o#o serial
TIMEOUTFLAG        EQU 0FH      ; sinalizador de time-out

=====

; =====
; programa principal
; =====

ORG 1000H

MOV IP0, #001000000B
MOV IP1, #00000001B
MOV PSW, #00H
MOV SP, #30H

LCALL RAMXTEST

LCALL INIT_SER

LCALL INIT_TIMER2      ; inicializa timer 2

LOOP:
    JNB MINFLAG, TCHAU
    CLR MINFLAG
    MOV A, MINUTOS
    CJNE A, #29, TESTE59
    LCALL ADC
    SJMP VAIMED
TESTE59:
    CJNE A, #59, TESTE1205
    LCALL ADC
VAIMED:
    LCALL MEDIA
    LCALL WRITE_ACQ_DATA
    SJMP TCHAU
TESTE1205:
    CJNE A, #5, TCHAU
    MOV A, HORAS
    CJNE A, #12, TCHAU
    LCALL TRANSFER
TCHAU:
    SJMP LOOP

=====

```

; INICIALIZACAO DA PORTA SERIAL

INIT_SER:

```
CLR RI      ; zera flags de interrupção de recepção
CLR TI      ; e de transmissão

SETB ADCON.7 ; serial com gerador de baud rate interna(timer1)
MOV TMOD, #020H ; timer 1 modo 2 (auto recarga de th1 em tl1)
MOV TH1, #0F3H ; ajuste para 4800 bauds
MOV PCON, #080H ;

MOV SCON, #00H ;
SETB SM1      ; modo1: 10 bits(startb+8b+stopb)
SETB REN      ; habilita recepção

MOV TCON, #00H ;
SETB TR1      ; dispara timer1

CLR P5.7      ; ajusta módulo para recepção

SETB EAL      ;
SETB ES      ; habilita interrupção via serial

RET
```

```
;=====
; subrotina init_timer2
;==> inicializa o timer 2 como auto reload modo 0
; gera uma interrupção por overflow a cada 50 ms
; valor de recarga do registrador do timer2: 3cb0h
;=====
```

INIT_TIMER2:

```
MOV APONT_MEM_AQ_HI, #0B0H
MOV APONT_MEM_AQ_LO, #00H
MOV SEGUNDOS, #00H
MOV MINUTOS, #00H
MOV HORAS, #00H
MOV DIAS, #01H
MOV CONTSEC, #00H
CLR SEGFLAG
CLR MINFLAG
CLR HORFLAG
CLR DIAFLAG

MOV CRCH, #03CH
MOV CRCL, #0B0H
MOV TH2, #03CH
MOV TL2, #0B0H
MOV T2CON, #00010001H ; timer2 auto reload modo0
SETB EAL
SETB ET2      ; habilita int. do timer2
RET
```

```
;=====
;=====
; rotina de interrupção do timer 2
;==> incrementa contadores de interrupção, segundos
; horas e dias
```

```

;-----
TIMER2_INT:
    CLR TF2
    PUSH PSW
    MOV PSW, #0 ; seleciona banco de registradores 0
    PUSH ACC
    PUSH DPH
    PUSH DPL

    INC CONT_INT
    MOV A, CONT_INT

    CJNE A, #20, EHFIM
    MOV CONT_INT, #00
    INC SEGUNDOS
    INC CONTSEC
    SETB SEGFLAG
    MOV R0, SEGUNDOS
    CJNE R0, #60, EHFIM
    MOV SEGUNDOS, #00
    INC MINUTOS
    SETB MINFLAG
    MOV R0, MINUTOS
    CJNE R0, #60, EHFIM
    INC HORAS
    MOV MINUTOS, #00
    SETB HORFLAG
    MOV R0, HORAS
    CJNE R0, #24, EHFIM
    MOV HORAS, #00
    INC DIAS
    SETB DIAFLAG

EHFIM:
    POP DPL
    POP DPH
    POP ACC
    POP PSW
    RETI

```

```

;=====

```

```

;=====
; este programa mostra os valores de tensão presentes nas
; 8 entradas do conversor A/D do 80535
; estas voltagens são provenientes de 8 pontos da placa rware
; ajustáveis entre 0 e 5 volts através de potenciômetros
; o programa realiza várias aquisições para cada sensor
; a fim de eliminar erro através do cálculo de médias
;
;
;=====
; programa principal
;
;-----
; conexões:
; sinais dos 8 potenciômetros analog0 a analog7 conectados
; as 8 entradas an0 a an7 através de resistores
; de valores entre 1kohm e 10kohm.
;
;-----
; as voltagens vistas nos pinos do conversor A/D do
; 80535 são mostradas na tela do monitor com aproximação
; decimal de duas casas.
;
; jumpers jp2 e jp3 devem ser instalados.
;-----

```



```
ADC:
    PUSH PSW
    PUSH ACC
    PUSH DPH
    PUSH DPL

    SETB RSO
    MOV INIT_ADDR_HI, #0A0H
    MOV INIT_ADDR_LO, #00H
```

```
BEGIN:
    LCALL INICICADO
    MOV DPH, INIT_ADDR_HI
    MOV DPL, INIT_ADDR_LO
    MOV R6, #15
    MOV R5, #8
```

```
GOON:
    MOV DAPR, #00H
```

```
    MOV R7, #8
```

```
WW2:
    DJNZ R7, WW2
```

```
    DJNZ R6, INKEPT
    INC DPTR
    MOV R6, #15
    INC ADCON
    DJNZ R5, INKEPT
    CLR RSO
```

```
    POP DPL
    POP DPH
    POP ACC
    POP PSW
    RET
```

```
INKEPT:
    INC DPTR
    SJMP GOON
```

```
=====
; subrotinas inicicad - inicializam conversões A/D
=====
```

```
INICICADO:
    CLR EADC      ;desabilita interrupções do conversor a/d
    CLR IADC      ;reseta flag de interrupção do cad

    SETB EADC     ; adc interrupt mask bit
    MOV A, ADCON  ; le registrador de controle do cad
    ANL A, #0D0H  ; seleciona canal 0, uma conversão
    MOV ADCON, A

    RET
```

```
=====
; subrotinas ftintcad
=====
```

```
FTINTCADO:
    PUSH PSW
    PUSH ACC
```

```

CLR IADC      ; zera flag de int. do conv. a/d
MOV  A, ADDAT ; lê valor analógico
MOVX @DPTR, A ; armazena valor de analog0

POP  ACC
POP  PSW
RETI

;-----

;=====
; este programa efetua a soma das medições realizadas
; referentes a um sensor.
; o resultado da soma consiste de um numero de dois bytes
; que será armazenado nas variáveis soma_h e soma_l

;-----

MEDIA:
    PUSH PSW ;
    PUSH ACC ; salva registradores
    PUSH DPL ;
    PUSH DPH ;

PULA1:
    CLR C
    MOV INIT_ADDR_HI, #0A0H ;
    MOV INIT_ADDR_LO, #00H ; inicializa variáveis

SEGUEF:
    MOV SOMA_H, #00H ;
    MOV SOMA_L, #00H ;
    MOV CONT_SUM, #1 ; inicializa parcelas da soma
    MOV DPTR, #0000H ;
    MOV DPL, INIT_ADDR_LO ; e
    MOV DPH, INIT_ADDR_HI ;
    MOVX A, @DPTR ; pega a primeira parcela (valor da 1a aq.)
    MOV SOMA_L, A ;

MAIS_UMA:
    MOV A, INIT_ADDR_LO ;
    ADD A, CONT_SUM ;
    MOV DPL, A ;
    MOVX A, @DPTR ; pega próxima parcela da soma (próxima aq.)
    ADDC A, SOMA_L ; e soma

    JNC NO_ENCHEU ; testa se ocorreu overflow em soma_l
    CLR C ; se enche zera o carry
    INC SOMA_H ; e incrementa soma_h

NO_ENCHEU:
    MOV SOMA_L, A ; atualiza soma_l
    INC CONT_SUM ;
    MOV R1, CONT_SUM ; e verifica se j foram somadas
    CJNE R1, #15, SEG_SUM ; todas as 15 parcelas (aquisições)

FIM_SUM:
    SJMP DIVISAO ; se já vai para o cálculo da média

SEG_SUM:
    SJMP MAIS_UMA ; se não vai para a próxima parcela

DIVISAO:
    MOV DIVISOR, #15 ; ajusta número de aquisições

    MOV DIVIDENDO_H, SOMA_H ;
    MOV DIVIDENDO_L, SOMA_L ; carrega o resultado da soma

```

```

MOV QUOCIENTE, #0 ;
MOV QUOCIENTE_H, #0 ;
MOV QUOCIENTE_L, #0 ;
MOV RESTO, #0 ; inicializa variáveis
MOV RESTO_H, #0 ;
MOV RESTO_L, #0 ;

MOV A, #OFFH ;
MOV B, DIVISOR ; calcula o resultado (quociente e resto)
DIV AB ; da divisão de um byte cheio(ffh)
MOV QUOCIENTE_H, A ; pelo divisor
MOV RESTO_H, B ;

MOV A, DIVIDENDO_H ; a cada unidade de dividendo_h o quociente
MOV B, QUOCIENTE_H ; ser aumentado do valor obtido em quociente_h
MUL AB ;
MOV QUOCIENTE, A ;

MOV A, DIVIDENDO_H ;
MOV B, RESTO_H ;
INC B ; o mesmo acontece com resto e resto_h
MUL AB ; o incremento de b corresponde ao vai1
MOV RESTO, A ;

MOV A, DIVIDENDO_L ;
MOV B, DIVISOR ; calcula o resultado da divisão de dividendo_l
DIV AB ; pelo divisor
ADD A, QUOCIENTE ; e adiciona ao quociente atual
MOV QUOCIENTE, A ;

MOV A, B ;
ADD A, RESTO ; pega o resto e soma ao resto atual
MOV RESTO, A ;

MOV A, RESTO ;
MOV B, DIVISOR ; divide o resto pelo divisor e acrescenta o
DIV AB ; resultado ao quociente
ADD A, QUOCIENTE ;
MOV QUOCIENTE, A ;
MOV RESTO, B ; o resto fica em resto

MOV A, RESTO ; e se for maior que a metade
CJNE A, #8, TESTA ; do divisor aumenta uma unidade ao quociente
SJMP ARREDONDA ;

TESTA:
JC EHFIMDIV

ARREDONDA:
INC QUOCIENTE

EHFIMDIV:
MOV A, INIT_ADDR_LO ;
ADD A, #0FH ;
MOV DPL, A ; escreve o quociente no endereço de rascunho
MOV DPH, INIT_ADDR_HI ; destinado .... m,dia
mov a, quociente
MOVX @DPTR, A
MOV RO, DPL ;
CJNE RO, #OFFH, VAIF ; e testa se já foram calculadas 8 médias

POP DPH ;
POP DPL ; se já recupera registradores e retorna
POP ACC ;
POP PSW ;
RET ;

VAIF:
INC RO ; se não vai calcular a próxima soma
MOV INIT_ADDR_LO, RO ;
LJMP SEGUEF ;

```

```

=====
;
;
;
=====
WRITE_ACQ_DATA:

```

```

    PUSH PSW
    PUSH ACC
    PUSH DPL
    PUSH DPH

    MOV SENSOR, #0
    MOV A, DIAS
    LCALL WRITE_DATA
    MOV A, HORAS
    LCALL WRITE_DATA
    MOV A, MINUTOS
    LCALL WRITE_DATA

```

```

GO_WRITE:

```

```

    MOV A, SENSOR
    LCALL WRITE_DATA
    MOV A, SENSOR
    SWAP A
    ORL A, #0FH
    MOV DPH, #0A0H
    MOV DPL, A
    MOVX A, @DPTR
    LCALL WRITE_DATA
    INC SENSOR
    MOV A, SENSOR
    CJNE A, #8, GO_WRITE

```

```

    POP DPH
    POP DPL
    POP ACC
    POP PSW
    RET

```

```

WRITE_DATA:

```

```

    MOV DPH, APONT_MEM_AQ_HI
    MOV DPL, APONT_MEM_AQ_LO
    MOVX @DPTR, A
    INC APONT_MEM_AQ_LO
    MOV A, APONT_MEM_AQ_LO
    CJNE A, #00H, VOLTA
    INC APONT_MEM_AQ_HI

```

```

VOLTA:

```

```

    RET

```

```

=====
;
;
;
=====
; TESTE DE LEITURA E ESCRITA NA RAM EXTERNA

```

```

RAMXTEST:

```

```

    PUSH PSW
    PUSH ACC

    MOV PSW, #00H
    MOV DPTR, #09100H

```

```

ENCHE:

```

```

    MOV A, #0AAH
    MOVX @DPTR, A

```



```

MOV R4, DPL
MOV R5, DPH
INC DPTR
CJNE R4, #0FFH, ENCHE
CJNE R5, #0FFH, ENCHE
MOV DPTR, #09f00H

ESVAZIA:
MOVX A, @DPTR
MOV R4, DPL
MOV R5, DPH
INC DPTR

CJNE A, #0AAH, ERRO
CJNE R4, #0FFH, ESVAZIA
CJNE R5, #0FFH, ESVAZIA
MOV DPTR, #09f00H

ENCHE1:
MOV A, #055H
MOVX @DPTR, A
MOV R4, DPL
MOV R5, DPH

INC DPTR
CJNE R4, #0FFH, ENCHE1
CJNE R5, #0FFH, ENCHE1
MOV DPTR, #09f00H

ESVAZIA1:
MOVX A, @DPTR
MOV R4, DPL
MOV R5, DPH
INC DPTR
CJNE A, #055H, ERRO
CJNE R4, #0FFH, ESVAZIA1
CJNE R5, #0FFH, ESVAZIA1

MOV RXST, #4BH
LJMP TESTADO

ERRO:
MOV RXST, #46H
TESTADO:
POP ACC
POP PSW
RET
;
;=====
TRANSFER:

MOV R4, #0B0H
MOV R5, #00H
MOV R6, #0D0H
MOV R7, #00H

TRANSFER1:
CJNE R4, #0B3H, GO_TRANSFER
CJNE R5, #090H, GO_TRANSFER
MOV APONT_MEM_AQ_HI, #0B0H
MOV APONT_MEM_AQ_LO, #00H
RET
GO_TRANSFER:
MOV DPH, R4
MOV DPL, R5
MOVX A, @DPTR
MOV DPH, R6
MOV DPL, R7
MOVX @DPTR, A
INC R5
CJNE R5, #00H, MENOS

```

```

INC R4
MENOS:
INC R7
CJNE R7, #00H, MENOS1
INC R6
MENOS1:
LJMP TRANSFER1

```

```

; programa finalreq.asm
; ==> Este programa contém as rotinas referentes a comunicação
; serial dos módulos de aquisição. Mediante um sinal no
; barramento é gerada uma interrupção, o módulo captura o frame
; (desde que o endereço de destino seja o seu) e executa uma
; ação de acordo com o campo comando do frame.

```

```

=====
send_pack equ 2000h

```

```

=====
; programa rec_quadro.asm      ultima atualização: 06/12/1997
; esta rotina, a partir de uma interrupcao gerada pela presença
; de informação na porta serial, recebe, monta e confere o quadro
; de comunicação, e se necessário encaminha para rotina de resposta
; ao comando presente no quadro de comunicação.
; esta rotina é especifica para módulos de aquisição de dados.

```

```

=====
; o quadro utilizado pelo protocolo de comunicação é
; constituído dos seguintes campos;

```

IQ	EN	EO	ED	CM	CB	DADOS	CS	FQ
----	----	----	----	----	----	-------	----	----

onde:

```

IQ   : inicio de quadro  --> 1 byte (*)
EN   : enquiry           --> 1 byte (?) hospedeiro para modulo de aquisicao
      :                  (@) modulo de aquis. para hospedeiro
EO   : endereco de origem --> 2 bytes
ED   : endereco de destino --> 2 bytes
CM   : comando           --> 1 byte
CB   : contador de bytes de dados --> 2 bytes (00 a ff)
DADOS : campo de dados --> ate 255 bytes
CS   : check-sum        --> 2 bytes (00 a ff)
FQ   : fim de quadro     --> 1 byte (#)

```

```

=====
; constantes

```

```

INICIO_QUADRO EQU 02AH ; '*'

```

```

HOST_ENQ    EQU 040H ; '@'
MODUL_ENQ   EQU 03FH ; '?'
FIM_QUADRO  EQU 023H ; '#'
MY_ADDR_HI  EQU 030H ; '0' endereço específico
MY_ADDR_LO  EQU 031H ; '1' para cada módulo
HOST_ADDR_HI EQU 046H ; 'F'
HOST_ADDR_LO EQU 041H ; 'A'
UNIV_ADDR_HI EQU 046H ; 'F'
UNIV_ADDR_LO EQU 046H ; 'F'

```

```

=====
; programa principal
=====

```

```

ORG      1800H

REC_QUA:

    PUSH    PSW
    PUSH    ACC
    PUSH    DPL
    PUSH    DPH

    CLR      IEN0.4           ; desabilita interrupção via serial
    SETB     SERFLAG         ; seta flag de aviso de comunicação
    SETB     RS1             ; seleciona banco de reg. número 3

    MOV      R0, #CH_SUM_CALC ; inicializa check-sum
    MOV      @R0, #00H
    MOV      AUX1, #00H
    MOV      R0, #R_INIT_DATA_ADDR_HI
    MOV      @R0, #09FH
    MOV      R0, #R_INIT_DATA_ADDR_LO
    MOV      @R0, #00H

    LCALL    GET_CHAR
    CJNE     A, #INICIO_QUADRO, GO_FIM_REC

    LCALL    GET_CHAR
    JB       TIMEOUTFLAG, JAERA
    CJNE     A, #MODUL_ENQ, GO_FIM_REC ; verifica campo EN

    LCALL    GET_CHAR
    JB       TIMEOUTFLAG, JAERA
    CJNE     A, #HOST_ADDR_HI, GO_FIM_REC
    MOV      R0, #CH_SUM_CALC
    ADD      A, @R0           ; verifica o campo EO
    MOV      @R0, A          ; (este campo deve conter

    LCALL    GET_CHAR
    JB       TIMEOUTFLAG, JAERA
    CJNE     A, #HOST_ADDR_LO, GO_FIM_REC
    MOV      R0, #CH_SUM_CALC
    ADD      A, @R0
    MOV      @R0, A

    LCALL    GET_CHAR
    JB       TIMEOUTFLAG, JAERA
    CJNE     A, #MY_ADDR_HI, L_UNIV_HI
    SJMP     SEGUE           ; verifica o campo ED
L_UNIV_HI:
    CJNE     A, #UNIV_ADDR_HI, GO_FIM_REC ; (este campo deve conter
SEGUE:
    MOV      R0, #CH_SUM_CALC
    ADD      A, @R0           ; o endereço universal )
    MOV      @R0, A

    LCALL    GET_CHAR        ; e

```

```

JB  TIMEOUTFLAG, JAERA
CJNE A, #MY_ADDR_LO, L_UNIV_LO
SJMP SEGUE2 ; também
JAERA:
GO_FIM_REC:
LJMP FIM_REC

L_UNIV_LO:
CJNE A, #UNIV_ADDR_LO, GO_FIM_REC ; inclui no check-sum
SEGUE2:
MOV R0, #CH_SUM_CALC
ADD A, @R0
MOV @R0, A

LCALL GET_CHAR
JB  TIMEOUTFLAG, JAERA
MOV R0, #R_COMANDO ; recebe o campo CM
MOV @R0, A
MOV R0, #CH_SUM_CALC
ADD A, @R0
MOV @R0, A ; e inclui no check-sum

LCALL GET_CHAR
JB  TIMEOUTFLAG, JAERA
MOV R0, #CONT_DADOS_4
MOV @R0, A
MOV R0, #CH_SUM_CALC
MOV AUX1, @R0
ADD A, AUX1
MOV AUX1, A

LCALL GET_CHAR
JB  TIMEOUTFLAG, JAERA
MOV R0, #CONT_DADOS_3
MOV @R0, A
ADD A, AUX1
MOV AUX1, A

LCALL GET_CHAR
JB  TIMEOUTFLAG, JAERA
MOV R0, #CONT_DADOS_2
MOV @R0, A
ADD A, AUX1 ; recebe o campo CB
MOV AUX1, A

LCALL GET_CHAR ; e inclui no check-sum
JB  TIMEOUTFLAG, JAERA
MOV R0, #CONT_DADOS_1
MOV @R0, A
ADD A, AUX1 ; se este campo for diferente
MOV R0, #CH_SUM_CALC ; de zero
MOV @R0, A

LCALL CONT_DADOS_ASCTOHEX

MOV R0, #CONT_DADOS_LO ; então eh acessada rotina
MOV A, @R0
CJNE A, #00H, TEM_DADOS ; para montar o campo DADOS
MOV R0, #CONT_DADOS_HI
MOV A, @R0
CJNE A, #00H, TEM_DADOS

SJMP SEGUE3

TEM_DADOS:
LCALL MONT_DATA_FIELD
JB  TIMEOUTFLAG, FIM_REC

SEGUE3:
LCALL GET_CHAR

```



```

JB  TIMEOUTFLAG, FIM_REC
MOV      RO, #CHECK_SUM_HI
MOV      @RO, A
LCALL    GET_CHAR      ; recebe o campo CS
JB  TIMEOUTFLAG, FIM_REC
MOV      RO, #CHECK_SUM_LO
MOV      @RO, A

LCALL    GET_CHAR
JB  TIMEOUTFLAG, FIM_REC
CJNE     A, #FIM_QUADRO, FIM_REC

LCALL    MDELAY

LCALL    CH_SUM_ASCTOHEX ; e compara com o
MOV      RO, #CH_SUM_CALC
MOV      AUX1, @RO
CJNE     A, AUX1, ERRO_CHSUM

LCALL    IDENT_CMNDO    ; chama rotina identificação do campo CM
FIM_REC:
POP      DPH
POP      DPL
POP      ACC
POP      PSW

SETB     IEN0.4         ; habilita interrupção
CLR      SERFLAG        ; zera flag de aviso de serial
CLR      TI
RETI                      ; retorna para o programa original

ERRO_CHSUM:
SJMP     FIM_REC

;
; fim do programa principal
;-----

;
; subrotina ident_cmndo ==> identifica o comando recebido e acessa
; a subrotina correspondente ao mesmo.
;-----
IDENT_CMNDO:
MOV      RO, #R_COMANDO
MOV      A, @RO
CJNE     A, #054H, NOT_T      ; teste para comando T (54h)
LCALL    COMANDO_T
RET
NOT_T:
CJNE     A, #053H, NOT_S      ; teste para comando S (53h)
LCALL    COMANDO_S
RET
NOT_S:
CJNE     A, #048H, NOT_H      ; teste para comando H (48h)
LCALL    COMANDO_H
RET
NOT_H:
RET

;
; subrotina mont_data_field ==> monta o campo de dados recebido no frame
;-----
MONT_DATA_FIELD:
MOV      RO, #R_INIT_DATA_ADDR_HI
MOV      A, @RO
MOV      DPH, A
MOV      RO, #R_INIT_DATA_ADDR_LO
MOV      A, @RO

```

```

MOV      DPL, A
MOV      R0, #CONT_DADOS_HI
MOV      A, @R0
MOV      R7, A
INC      R7
MOV      R0, #CONT_DADOS_LO
MOV      A, @R0
MOV      R6, A
MOV      R0, #CH_SUM_CALC
MOV      AUX1, @R0
GO_REC:
LCALL    GET_CHAR
JB       TIMEOUTFLAG, BYE
MOVX     @DPTR, A
ADD      A, AUX1
MOV      AUX1, A
INC      DPTR
DJNZ     R6, GO_REC
DJNZ     R7, GO_REC
MOV      R0, #CH_SUM_CALC
MOV      @R0, AUX1
BYE:
RET

```

```

;subrotina comando_t ==> ajusta dia e hora de acordo com dados recebidos via serial.

```

```

COMANDO_T:
MOV      R0, #R_INIT_DATA_ADDR_HI
MOV      A, @R0
MOV      DPH, A
MOV      R0, #R_INIT_DATA_ADDR_LO
MOV      A, @R0
MOV      DPL, A
MOVX     A, @DPTR
LCALL    ASCBIN
SWAP     A
MOV      B, A
INC      DPTR
MOVX     A, @DPTR
LCALL    ASCBIN
ORL      A, B
MOV      DIAS, A

INC      DPTR
MOVX     A, @DPTR
LCALL    ASCBIN
SWAP     A
MOV      B, A
INC      DPTR
MOVX     A, @DPTR
LCALL    ASCBIN
ORL      A, B
MOV      HORAS, A

INC      DPTR
MOVX     A, @DPTR
LCALL    ASCBIN
SWAP     A
MOV      B, A
INC      DPTR
MOVX     A, @DPTR
LCALL    ASCBIN
ORL      A, B
MOV      MINUTOS, A

```

RET

```

=====
; subrotina comando_h ==> envia para o hospedeiro um frame de
; comunicação contendo no campo de comando
; o conteúdo do registrador de status da
; ram externa do módulo
; ( 'k'--> ok! ; 'f'--> falha! )
=====

```

COMANDO_H:

```

MOV      A, RXST
MOV      R0, #S_COMANDO
MOV      @R0, A
MOV      R0, #S_INIT_DATA_ADDR_HI
MOV      @R0, #09FH
MOV      R0, #S_INIT_DATA_ADDR_LO
MOV      @R0, #00H
MOV      R0, #S_CONT_DADOS_HI
MOV      @R0, #00H
MOV      R0, #S_CONT_DADOS_LO
MOV      @R0, #06H
MOV      R0, #IS_CONT_DADOS_HI
MOV      @R0, #00
MOV      R0, #IS_CONT_DADOS_LO
MOV      @R0, #03H
MOV      DPTR, #9F00H
MOV      A, DIAS
MOVX     @DPTR, A
INC      DPTR
MOV      A, HORAS
MOVX     @DPTR, A
INC      DPTR
MOV      A, MINUTOS
MOVX     @DPTR, A

SETB     P5.7          ; ajusta módulo para transmissão

LCALL    SEND_PACK

CLR      P5.7          ; ajusta módulo para recepção
RET

```

```

=====
; subrotina comando_s ==> envia pacote de dados referente a aquisições
; de um período preestabelecido.
=====

```

COMANDO_S:

```

MOV      R0, #S_INIT_DATA_ADDR_HI
MOV      @R0, #0D0H
MOV      R0, #S_INIT_DATA_ADDR_LO
MOV      @R0, #00H

MOV      R0, #IS_CONT_DADOS_LO
MOV      @R0, #090H
MOV      R0, #IS_CONT_DADOS_HI
MOV      @R0, #03H

MOV      R0, #S_CONT_DADOS_LO
MOV      @R0, #020H
MOV      R0, #S_CONT_DADOS_HI
MOV      @R0, #07H

MOV      R0, #S_COMANDO

```

```

MOV      @R0, #053H      ; 'S'
SETB     P5.7            ; ajusta módulo para transmissão
LCALL    SEND_PACK
CLR      P5.7            ; ajusta módulo para recepção
RET

```

```

;=====
; subrotina get_char ==> recebe caractere presente na porta serial
; colocando-o no acumulador.
;=====

```

```

GET_CHAR:
MOV      CONTSEC, #00H
CLR      TIMEOUTFLAG
GET_CHAR1:
MOV      A, CONTSEC
CJNE     A, #5, GOGET
SETB     TIMEOUTFLAG
RET
GOGET:
JNB      RI, GET_CHAR1
MOV      A, SBUF
CLR      RI
RET

```

```

;=====
; subrotina cont_dados_asctohex ==> converte os dois caracteres ascii
; correspondentes a 0-9 ou a-f do campo
; CD num único byte composto por dois
; nibbles de valores correspondentes.
;=====

```

```

CONT_DADOS_ASCTOHEX:
PUSH     ACC
PUSH     PSW

MOV      R0, #CONT_DADOS_4
MOV      A, @R0
LCALL    ASCBIN
MOV      B, A
MOV      R0, #CONT_DADOS_3
MOV      A, @R0
LCALL    ASCBIN
ORL      A, B
MOV      R0, #CONT_DADOS_HI
MOV      @R0, A
MOV      R0, #CONT_DADOS_2
MOV      A, @R0
LCALL    ASCBIN
SWAP     A
MOV      B, A
MOV      R0, #CONT_DADOS_1
MOV      A, @R0
LCALL    ASCBIN
ORL      A, B
MOV      R0, #CONT_DADOS_LO
MOV      @R0, A

```



```
POP      PSW
POP      ACC

RET
```

```
=====
; subrotina ch_sum_asctohex ==> converte os dois caracteres ascii
; correspondentes a 0-9 ou a-f do campo
; CS num unico byte composto por dois
; nibbles de valores correspondentes.
=====
```

```
CH_SUM_ASCTOHEX:
MOV      R0, #CHECK_SUM_HI
MOV      A, @R0
LCALL    ASCBIN
SWAP     A
MOV      B, A
MOV      R0, #CHECK_SUM_LO
MOV      A, @R0
LCALL    ASCBIN
ORL      A, B
MOV      R0, #CH_SUM_HEX
MOV      @R0, A
RET
```

```
=====
; subrotina s_cont_dados_hextoasc ==> converte os dois bytes contadores
; de dados (high e low) nos quatro
; códigos ascii correspondentes
; (pares 4 e 3, e 2 e 1).
=====
```

```
S_CONT_DADOS_HEXTOASC:
MOV      R0, #S_CONT_DADOS_HI
MOV      A, @R0
LCALL    BINASC
MOV      R0, #S_CONT_DADOS_4
MOV      @R0, A
MOV      A, R2
MOV      R0, #S_CONT_DADOS_3
MOV      @R0, A

MOV      R0, #S_CONT_DADOS_LO
MOV      A, @R0
LCALL    BINASC
MOV      R0, #S_CONT_DADOS_2
MOV      @R0, A
MOV      A, R2
MOV      R0, #S_CONT_DADOS_1
MOV      @R0, A

RET
```

```
=====
; subrotina check_sum_hextoasc ==> converte o byte check_sum do quadro
; a ser enviado no par de código ascii
; correspondente ( high e low )
=====
```

```

CHECK_SUM_HEXTOASC:
    MOV     R0, #S_CHECK_SUM
    MOV     A, @R0
    LCALL   BINASC
    MOV     R0, #S_CHECK_SUM_HI
    MOV     @R0, A
    XCH     A, R2
    MOV     R0, #S_CHECK_SUM_LO
    MOV     @R0, A
    RET

```

```

;=====
; subrotina ascbin ==> esta rotina pega o caractere ascii presente
; no acumulador e o converte num numero binário de 4 bits que ,
; recolocado no acumulador.
;=====

```

```

ASCBIN:
    ADD     A, #0D0H      ; se char < 30 --> erro
    JNC     NOTNUM
    CLR     C              ; verifica se o char eh 0-9
    ADD     A, #0F6H      ; e ajusta-o, caso contrario
    JC      HEXTRY        ; salta
    ADD     A, #0AH
    RET

```

```

HEXTRY:
    CLR     ACC.5          ; verifica se o char eh a-f
    CLR     C
    ADD     A, #0F9H      ; se for ajusta-o
    JNC     NOTNUM
    CLR     C              ; caso contrário existe
    ADD     A, #0FAH
    JC      NOTNUM        ; um erro
    ANL     A, #0FH
    RET

```

```

NOTNUM:
    RET

```

```

;=====
; subrotina binasc ==> esta rotina pega o conteúdo do acumulador e o
; converte em dois números hexadecimais ascii correspondentes
; o ascii correspondente ao nibble menos significativo vai para r2
; e o ascii correspondente ao nibble mais significativo vai para o acc.
;=====

```

```

BINASC:
    MOV     R2, A
    ANL     A, #0FH
    ADD     A, #0F6H
    JNC     NOADJ1
    ADD     A, #07H
NOADJ1:
    ADD     A, #03AH

    XCH     A, R2
    SWAP    A
    ANL     A, #0FH
    ADD     A, #0F6H
    JNC     NOADJ2

```

```

        ADD      A, #07H
NOADJ2:
        ADD      A, #03AH
        RET

```

```

;=====
; subrotina mdelay ==> esta rotina gera um atraso de um milisegundo
;=====

```

```

MDELAY:
        PUSH     ACC
        MOV      A, #0A6H

```

```

MD_OLP:
        INC      A
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        JNZ      MD_OLP

```

```

        NOP
        POP      ACC
        RET

```

```

;=====
; fim de programa

```

```

;*****
;          *
;      wagner roberto balsani      *
;      mestrado em engenharia eletrica      *
;      universidade de sao paulo      *
;      escola de engenharia de sao carlos      *
;      orientador: prof. dr. paulo estevao cruvinel      *
;          *
;*****

```

```

;=====
binasc equ 1a7Fh
ascbin equ 1a63h

```

```

;=====
; constantes
;=====

```

```

inicio_quadro equ 02ah ;""
host_enq      equ 040h ; '@'
modul_enq     equ 03fh ; '?'
fim_quadro    equ 023h ; '#'
my_addr_hi    equ 030h ; '0' endereço específico
my_addr_lo    equ 031h ; '1' para cada módulo
host_addr_hi   equ 046h ; 'f'
host_addr_lo   equ 041h ; 'a'
univ_addr_hi   equ 046h ; 'f'
univ_addr_lo   equ 046h ; 'f'

```

```

;-----
; org 2000h
;-----

;=====
; subrotina send_pack ==> envia pela serial um pacote de informação
; em resposta a um pedido do hospedeiro.
;=====

SEND_PACK:

    MOV R0, #S_CHECK_SUM
    MOV @R0, #00H
    MOV AUX1, #00H

    MOV A, #INICIO_QUADRO ;
    LCALL SEND_CHAR ; envia caractere de inicio de quadro

    MOV A, #HOST_ENQ ;
    LCALL SEND_CHAR ; envia caractere de enquiry

    MOV A, #MY_ADDR_HI ;
    LCALL SEND_CHAR ;
    MOV R0, #S_CHECK_SUM ;
    MOV AUX1, @R0 ;
    ADD A, AUX1 ;
    MOV AUX1, A ; envia endereço de origem
    MOV A, #MY_ADDR_LO ; (endereço do módulo)
    LCALL SEND_CHAR ;
    ADD A, AUX1 ;
    MOV AUX1, A ;

    MOV A, #HOST_ADDR_HI ;
    LCALL SEND_CHAR ;
    ADD A, AUX1 ;
    MOV AUX1, A ; envia endereço de destino
    MOV A, #HOST_ADDR_LO ; (end. do hospedeiro)
    LCALL SEND_CHAR ;
    ADD A, AUX1 ;
    MOV AUX1, A ;

    MOV R0, #S_COMANDO ;
    MOV A, @R0 ;
    LCALL SEND_CHAR ;
    ADD A, AUX1 ; envia comando
    MOV AUX1, A ;

    LCALL S_CONT_DADOS_HEXTOASC;
    MOV R0, #S_CONT_DADOS_4 ;
    MOV A, @R0 ;
    LCALL SEND_CHAR ;
    ADD A, AUX1 ;
    MOV AUX1, A ;
    MOV R0, #S_CONT_DADOS_3 ;
    MOV A, @R0 ;
    LCALL SEND_CHAR ;
    ADD A, AUX1 ;
    MOV AUX1, A ;
    MOV R0, #S_CONT_DADOS_2 ;
    MOV A, @R0 ;
    LCALL SEND_CHAR ;
    ADD A, AUX1 ;
    MOV AUX1, A ; envia contadores de bytes
    MOV R0, #S_CONT_DADOS_1 ; do campo de dados
    MOV A, @R0 ;
    LCALL SEND_CHAR ;
    ADD A, AUX1 ;
    MOV R0, #S_CHECK_SUM
    MOV @R0, A

```



```

        LCALL SEND_DATA_FIELD    ; chama rotina de envio de
                                ; campo de dados

        LCALL CHECK_SUM_HEXTOASC ;

        MOV  R0, #S_CHECK_SUM_HI ;
        MOV  A, @R0
        LCALL SEND_CHAR          ; envia check-sum
        MOV  R0, #S_CHECK_SUM_LO ;
        MOV  A, @R0
        LCALL SEND_CHAR          ;

        MOV  A, #FIM_QUADRO      ;
        LCALL SEND_CHAR          ; envia caractere de fim de quadro

        CLR  TI                  ; zera flag de envio da serial

        RET

;-----

;=====
; subrotina send_data_field ==> envia pacote de dados requisitado
; pelo hospedeiro.
;=====

SEND_DATA_FIELD:
    MOV  R0, #S_CONT_DADOS_HI ; verifica se existem dados a ser
    MOV  A, @R0
    MOV  R7, A
    MOV  R0, #S_CONT_DADOS_LO ;
    MOV  A, @R0
    MOV  R6, A
    CJNE R6, #00H, START_SEND ; enviados
    CJNE R7, #00H, START_SEND ;
    RET

START_SEND:
    MOV  R0, #IS_CONT_DADOS_HI
    MOV  A, @R0
    MOV  R7, A
    MOV  R0, #IS_CONT_DADOS_LO
    MOV  A, @R0
    MOV  R6, A
    DEC  R6
    CJNE R6, #0FFH, SEGSTART
    DEC  R7
SEGSTART:
    MOV  R0, #S_INIT_DATA_ADDR_HI
    MOV  A, @R0
    MOV  DPH, A
    MOV  R0, #S_INIT_DATA_ADDR_LO
    MOV  A, @R0
    MOV  DPL, A
    MOV  R0, #S_CHECK_SUM
    MOV  AUX1, @R0

GO_SEND:
    MOVX A, @DPTR
    LCALL BINASC
    LCALL SEND_CHAR
    ADD  A, AUX1
    MOV  AUX1, A
    MOV  A, R2
    LCALL SEND_CHAR
    ADD  A, AUX1
    MOV  AUX1, A
    CJNE R6, #00H, GO_GO_SEND

```

```

DEC R6
CJNE R7, #00H, GO_GO_SEND1
MOV R0, #S_CHECK_SUM
MOV @R0, AUX1
RET

```

```

GO_GO_SEND:
DEC R6
INC DPTR
SJMP GO_SEND

```

```

GO_GO_SEND1:
DEC R7
INC DPTR
SJMP GO_SEND

```

```

=====
; subrotina send_char ==> envia caractere presente no acumulador através
; da porta serial.
=====

```

```

SEND_CHAR:
MOV SBUF, A
FICA:
JNB TI, FICA
CLR TI
RET

```

```

=====
; subrotina s_cont_dados_hextoasc ==> converte os dois bytes contadores
; de dados (high e low) nos quatro
; códigos ascii correspondentes
; (pares 4 e 3, e 2 e 1).
=====

```

```

S_CONT_DADOS_HEXTOASC:
MOV R0, #S_CONT_DADOS_HI
MOV A, @R0
LCALL BINASC
MOV R0, #S_CONT_DADOS_4
MOV @R0, A
MOV A, R2
MOV R0, #S_CONT_DADOS_3
MOV @R0, A

MOV R0, #S_CONT_DADOS_LO
MOV A, @R0
LCALL BINASC
MOV R0, #S_CONT_DADOS_2
MOV @R0, A
MOV A, R2
MOV R0, #S_CONT_DADOS_1
MOV @R0, A

RET

```

```

=====
; subrotina check_sum_hextoasc ==> converte o byte check_sum do quadro
; a ser enviado no par de código ascii
; correspondente ( high e low )
=====

```

```

CHECK_SUM_HEXTOASC:
    MOV R0, #S_CHECK_SUM
    MOV A, @R0
    LCALL BINASC
    MOV R0, #S_CHECK_SUM_HI
    MOV @R0, A
    XCH A, R2
    MOV R0, #S_CHECK_SUM_LO
    MOV @R0, A
    RET
    
```

—

Referências Bibliográficas

- ALAGUSUNDARAM, K.; JAYAS, D.S.; WHITE, N.D.G.; MUIR, W.E. (1990). Finite difference model of three-dimensional heat transfer in grain bins. *Canadian Agricultural Engineering*, v.32, p.315-321.
- ALLOCA, J.; STUART, A. (1984). *Transducer: theory and applications*. Virginia, Reston.
- ANDERSON, J.A.; BABBITT, J.D.; MEREDITH, W.O.S. (1943). The effect of temperature differential on the moisture content of stored wheat. *Canadian Journal of Research, Section C: Botanical Sciences*, v.21, p.297-306.
- ASAE STANDARDS (1990)a. Thermal Properties of Grain and Grain Products. 37.ed. St. Joseph.
- ASAE STANDARDS (1990)b. Moisture Relationship of Grains. 37.ed. St. Joseph.
- ASAE STANDARDS (1990)c. Psychrometric Data. 37.ed. St. Joseph.
- ASTHEIMER, R.W.; SCHWARZ, F. (1968). Thermal imaging using pyroelectric detectors. *Applied Optics*, v.7, n.9, p.1687-1695.
- ATKINSON, J.K. (1987). Communications protocols in instrumentation. *Journal of Physics E. Scientific Instruments*, v.20, p.484-491.
- BALSANI, W.R.; CRUVINEL, P.E.; TORRE-NETO, A. (1997). Monitoramento de parâmetros climáticos em silos agrícolas com arquitetura inteligente. SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE, 3., Vitória, 1997. *Anais*. p. 202-206.
- BIBBERO, R.J. (1977). *Microprocessors in instruments and control*. New Jersey, Wiley.
- BRIGNELL, J.E.; YOUNG, R. (1979). Computer-aided measurement. *Journal of Physics E, Scientific Instruments*, v.12, p.455-463.
- BRIGNELL, J.E.; DOREY, A.P. (1983). Sensors for microprocessor-based applications.

Journal of Physics E. Scientific Instruments, v.16, p.952-958.

BROOKER, D.B.; BAKKER-ARKEMA, F.W.; HALL, C.W. (1974). *Drying cereal grains*. Westport, AVI Pub.

CHEN, C.C.; MOREY, R.V. (1989). Comparison for four EMC/ERH equations. *Transactions of the ASAE*, v.32, n.3, p.983-990.

CIARCIA, S. (1988). Why microcontrollers? Part 1. *Byte*, p.239-247, Aug, 1988.

CIARCIA, S. (1988). Why microcontrollers? Part 2. *Byte*, p.303-312, Sept, 1988.

PROGRAMA de AUTOMAÇÃO (1995). São Carlos, CNPDIA - EMBRAPA.

CONVERSE, H.H.; GRAVES, A.H.; CHUNG, D.S. (1973). Transient heat transfer within wheat stored in a cylindrical bin. *Transactions of the ASAE*, v.16, p.129-133.

DIXON, J. (1988). Radiation thermometry. *Journal of Physics E. Scientific Instruments*, v.21, p.425-436.

DORTMANS, H.M.J.M. (1981). Application of microprocessors. *Journal of Physics E. Scientific Instruments*, v.14, p.777-782.

ECKERT, E.R.G.; FAGHRI, M.A. (1980). A general analysis of moisture migration caused by temperature differences in an unsaturated porous Medium. *International Journal of Heat and Mass Transfer*, v.23, p.1613-1623.

GRIFFITHS, H.J. (1964). Bulk storage of grain: a summary of factors governing control of deterioration. CISRO Division of Mechanical Engineering. Report ED8, Melbourne, Australia.

HOWELL, S.K.; HAMILTON, T.D.S. (1990). Intelligent instrumentation. *Measurement Science and Technology*, v.1, p.1265-1273.

- JIANG, S.; JOFRIET, J.C.; BUCHANAN-SMITH, J. (1988). Temperature observations in a bottom-unloading Concrete-Silo. *Canadian Agricultural Engineering*, v.30, p.249-255.
- KHANKARI, K.K.; MOREY, R.V.; PATANKAR, S.V. (1994) Mathematical model for moisture diffusion in stored grain due to temperature gradients. *Transactions of the ASAE*, v.37, n.5, p.1591-1604.
- LO, K.M.; CHEN, C.S.; CLAYTON, J.T.; ADRIAN, D.D. (1975). Simulation of temperature and moisture changes in wheat storage due to weather variability. *Journal of Agricultural Engineering Research*, v.20, p.47-53.
- LONGSTAFF, R.A.; BANKS, H.J. (1987). Simulation of temperature fluctuations near the surface of grain bulks. *Journal of Stored Products Research*, v.23, p.21-30.
- LOURO, A.H.F. (1995). *Projeto de um sistema inteligente baseado no microcontrolador 80535 para aquisição de dados e controle em ambiente agropecuário*. São Carlos. 158p. Dissertação (Mestrado) - Centro de Ciências Exatas e de Tecnologia, Universidade Federal de São Carlos.
- MANBECK, H.B.; BRITTON, M.G. (1988). Predictions of bin wall temperature declines. *Transactions of the ASAE*, v.31, p.1767-1773.
- METZGER, J.F.; MUIR, W.E. (1983). Computer model of two dimensional conduction and forced convection in stored grain. *Canadian Agricultural Engineering*, v.25, p.119-125.
- MICROCONTROLLER HANDBOOK (1988) Sunnyvale, Advanced Micro Devices.
- MUIR, W.E. (1970). Temperature in grain bins. *Canadian Agricultural Engineering*, v.12, p.21-24.
- MUIR, W.E.; FRASER., B.M.; SINHA, R.N. (1980). Simulation model of two dimensional heat transfer in controlled atmosphere grain bins. In: Shejbal, J., ed. *Controlled atmosphere storage of grains*. Amsterdam, Elsevier Sci. p.385-397.

-
- PATANKAR, S.V. (1980). *Numerical heat transfer and fluid flow*. New York, Hemisphere.
- PATANKAR, S.V. (1991). *Computation of conduction and duct flow heat transfer*. Maple Grove, Innovative Research.
- PAYNE, P.A. (1983). Sensors and their applications. *Journal of Physics E. Scientific Instruments*, v.16, p.947-951.
- R-535/R-WARE - USER'S GUIDE Version 1.1 (1991). Gainesville, Rigel Corporation.
- SCHEPERS, J.S. (1994). New diagnostic tools for tissue testing. *Communications in Soil Science and Plant Analysis*, v.25, n.7,8, p.817-826.
- SEYER, M.D. (1984). *RS 232 made easy: connecting computers, printers, terminals, and modems*. Englewood Cliffs, Prentice-Hall.
- SINCLAIR, I.R. (1992). *Sensors and transducers: a guide for technicians*. 2.ed. Oxford , Butterworth/Heinemann.
- STEWART, J.A. (1975). Moisture migration during storage of preserved, high moisture grains. *Transactions of the ASAE*, v.18, n.2, p.387-393, 400.
- SYDENHAM, P.H. (1985). *Transducers in measurement and control*. 3.ed. Bristol, Adam Hilger.
- THORPE, G.R. (1980). Moisture diffusion through bulk grain. *Journal of Stored Products Research*, v.17, p.39-42.
- TANENBAUM, A.S. (1994). *Redes de computadores*. Rio de Janeiro, Campus.
- TORRE NETO, A. (1995). *Estudo e implementação de um sistema de monitoramento remoto de variáveis edafô-ambientais*. São Carlos. 152p. Tese (Doutorado) - Instituto de Física de São Carlos, Universidade de São Paulo.

- TORRE-NETO, A.; CRUVINEL, P.E.; SLAETS, J.F.W.; CRESTANA, S. (1997). Remote monitoring of enviromental variables for modelling of pesticide transport in soil. *Applied Engineering in Agriculture*, v.13, n.1, p. 155-122.
- TORRE-NETO, A.; CRUVINEL, P.E.; INAMASU, R.Y.; CRESTANA, S. (1997). Tecnologias de ponta na agricultura: instrumentação e automação para a agricultura de precisão. CONGRESSO BRASILEIRO DE ENGENHARIA AGRÍCOLA, 26., Campina Grande, 1997. *Anais*. p. 75-95.
- WHITE, G.G. (1988). Temperature changes in bulk stored wheat in sub-tropical Australia. *Journal of Stored Products Research*, v.24, p.5-11.
- WILLIS, N. (1986). *Computer architecture and communications*. London, Tab Books.
- YACIUK, G.; MUIR, W.E.; SINHA, R.N. (1975). A simulation model of temperatures in stored grain. *Journal of Agricultural Engineering Research*, v.20, p.245-258.

Glossário

ASCII - American Standard Code for Information Interchange: Padrão de código de 7 bits para comunicação de dados. O código completo tem 128 caracteres incluindo letras maiúsculas e minúsculas, dígitos, símbolos e caracteres de controle. Também é usada uma subserie de 96 caracteres que exclui os caracteres de controle de comunicação.

Assembly - Linguagem de baixo nível para um modelo particular de computador ou para uma gama de modelos de computadores.

Bit - dígito binário (0 ou 1).

Bit stuffing - Uma provisão de protocolo SDLC e HDLC pela qual uma unidade de transmissão adiciona e uma unidade de recepção suprime um bit (0) após qualquer sequência de 5 bits iguais a 1 (um). O onjetivo é assegurar que nenhuma sequência nos dados do usuário seja confundida com um flag (01111110).

Byte - cadeia binária de comprimento padrão: 8 bits.

Backplane - O lado posterior de um painel de controle ou de display. O lado que contém a fiação de conexão.

Check-sum - Valor obtido pela aplicação de um algoritmo para um bloco de dados que é transmitido ou armazenado com os dados para uso de checagem em passo subsequente.

Clock - Sinal composto de pulsos precisa e igualmente espaçados com o objetivo de controlar a operação da máquina.

Código BCD - Binary code decimal: método de representação de números decimais em notação binária no qual cada dígito decimal é representado por quatro dígitos binários entre 0000 (zero) e 1001 (nove). Note a diferença entre BCD e binário puro: por exemplo, 87 é 1000 0111 em BCD e 1010111 em binário puro.

CPU - Unidade central de processamento.

DC - Corrente direta (direct current).

Disk-drive - Unidade de disco magnético.

Display driver: dispositivo que dirige a função de acionar um display. Tipicamente consiste de um gerador de varredura e um decodificador de segmentos que conecta os segmentos individuais necessários para formar cada caracter no display.

Dispositivos de I/O - dispositivo utilizado para manipular transferências de e para uma unidade microprocessada ou computador.

DMA - acesso direto a memória.

Drain - Dreno: terminal ou área positiva de silício num transistor de efeito de campo. O lugar através do qual os elétrons fluem durante a condução.

DSP - Processamento de sinais digitais.

Flag - (1) Um formato padrão de bits (01111110) que inicia e termina cada frame. Usado para sincronização e identificação do frame pelo hardware de recepção. (2) Indicador que mostra a existência de uma certa condição quando “setado” e a inexistência desta condição quando zerado.

Frame - Em SDLC é a unidade de transferência de dados inicializada e finalizada com um flag.

Full-duplex - Designação de um circuito, operação ou unidade funcional que fornece transmissão simultânea de dados em ambas direções entre duas partes.

Gate - a conexão elétrica para a junção de controle de um transistor ou tiristor.

Handshake - Termo aplicado às trocas iniciais que são feitas entre modems num barramento de comunicação de dados preparatórias para a transmissão de dados.

Host - O computador principal ou aquele que exerce a função de controle numa rede de dados na qual existem outros computadores, ou ainda o computador principal numa instalação com vários processadores.

ISO – International Standards Organisation

Latch - É um flip-flop: trigger com apenas duas condições estáveis.

Look up table - Tabelas que contém locações cujo conteúdo se deseja acessar.

MIPS - Milhões de instruções por segundo.

MOSFET - Metal-Oxide-Silicon Field Effect Transistor.

Nibble - metade de um byte: 4 bits.

On-chip - no mesmo chip de circuito integrado.

OSI – Open Systems Interconnection

Poll/response - tipo de comunicação na qual um dispositivo só envia uma mensagem quando solicitado por um outro dispositivo mestre.

Pull-up - Dispositivo usado para elevar a tensão em certo ponto de um circuito eletrônico.

RAM - Random Access Memory. Memória de dados.

RISC - Reduced instruction set computer: computador com série de instruções reduzidas.

ROM - Read Only Memory. Memória de programa.

Star bit - Em transmissão assíncrona de dados é um bit (0) que indica para o hardware de recepção que os bits seguintes fazem parte de um caractere.

Stop bit - Bit que indica fim de uma sequência de bits ou bytes. Usado como delimitador numa transmissão.

Trigger - um circuito com mais de um estado estável no qual uma transição de um estado estável para outro é iniciada por um único pulso.

UART - Transmissor receptor assíncrono universal: dispositivo de circuito integrado que realiza funções de interface entre um barramento paralelo e um dispositivo assíncrono serial tal como um modem ou periférico.