Rede de Comunicação para Equipamentos de Teste

RECET

André Luís de Andrade Mendes - Equitel Telecomunicações

Aluno de graduação do curso de Engenharia da Computação da PUC-PR

Luiz Antônio Pavão - Pontifícia Universidade Católica do Paraná - PUC PR.

Professor Mestre da PUC-PR

Manoel Camillo Penna - Centro Federal de Educação Tecnológica do Paraná - CEFET-PR.

Professor Doutor do CEFET-PR

Palavras-chave: rede de comunicação, rede de comunicação industrial, qualidade, coleta de dados, equipamentos de teste.

Resumo: O projeto **RECET** (Rede de Comunicação para Equipamentos de Testes) está sendo desenvolvido pela empresa Equitel S/A - Equipamentos e Sistemas de Telecomunicações. Este projeto visa a interligação em rede dos equipamentos de teste, permitindo a obtenção de dados do processo produtivo com maior confiabilidade e no menor tempo possível.

Abstract: The **RECET** (Rede de Comunicação para Equipamentos de Testes - Communication Network for Automatic Test Equipment) project is being developed within Equitel S/A - Equipamentos e Sistemas de Telecomunicações. This project aim the interconnection of all Automatic Test Equipment, allowing data to be collected with improved reability as quick as possible.

1. Introdução

No chão de fábrica, existem diversos equipamentos de teste, a grande maioria microprocessados, os quais geram dados de qualidade dos produtos fabricados pela Equitel.

A Interface Homem-Máquina de grande parte destes equipamentos é primitiva, obrigando que o operador ou apontadora manipule estes dados, gerando fichas de controle de produção. Estes dados são capturados por um processo de digitação e processados, resultando, assim, na geração de relatórios/gráficos que servem como subsídio para tomada de ações corretivas.

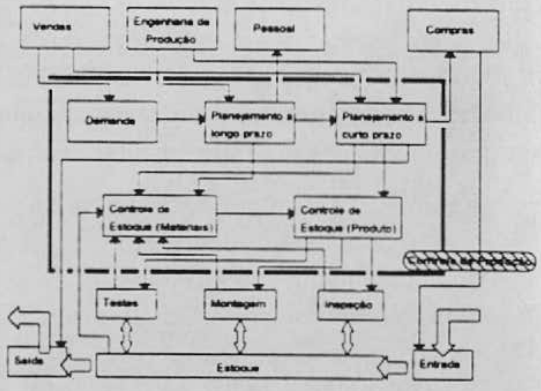


Figura 1. Fluxo de Materiais no Processo de Produção

O ponto a ser alcançado é o aumento da qualidade dos produtos e dados, através da automatização do processo de coleta de dados, motivando-se assim o desenvolvimento da Rede de Comunicação para Equipamentos de Teste (**RECET**). O projeto também visa suprir a necessidade da criação de uma rede a nível de chão de fábrica, com todos os equipamentos componentes interligados em um Sistema de Informações Gerenciais (**SIG**).

Exitem dois pontos deste sistema que merecem cuidados: a manipulação de dados pelo operador e o lapso de tempo entre o fato e uma possível ação corretiva. Quanto menos tempo o operador necessitar para preencher fichas ou digitar dados, mais tempo vai ter para realizar os seus encargos produtivos. Quanto menor o tempo entre a ocorrência de problemas de produção e as ações corretivas, menor será o custo de produção.

O gerenciamento da produção existe basicamente para planejar e controlar os esforços dos empregados e o fluxo de materiais dentro de uma organização [BED87], como nos mostra o diagrama em blocos do processo de produção, figura 1.

1.1. Informações para Gerenciamento da Produção

O conjunto dos dados relativos as atividades de produção juntamente com a atividade de processamento e disponibilização destes dados é chamado de Sistema de Informação do Chão de Fábrica.

Este sistema de informação deve cobrir todos os pontos do processo produtivo, como: estoque, manutenção, produção, inspeção, vendas, recepção, planejamento e gerência. Para que este sistema funcione de maneira eficiente e integrada, precisamos de interconexões através de redes de comunicação nos diversos niveis da estrutura fabril.

O tempo e a qualidade dos dados coletados no chão de fábrica têm um impacto significante na eficiência do processo produtivo, pois caso ocorra algum problema, e este não seja reportado em tempo hábil, podemos ter inconsistências no controle de estoque ou máquinas paradas durante muito tempo.

Uma quantidade enorme de informação é gerada no processo de manufatura, tais como: dados de qualidade, dados quantitativos, dados qualitativos, ocupação do equipamento, índices de produção, etc; cada um deles direcionados a seus respectivos setores. Um dos grandes problemas existentes em um processo de manufatura automatizado é garantir uma aquisição dos dados sempre válida e sem a existência de um lapso de tempo entre a geração do dado e a sua coleta pelo sistema de informação.

1.2. Sistema de Comunicação para a Coleta de Dados

Nos diversos niveis do processo fabril, existem atividades no controle do processo que necessitam de diferentes sistemas de comunicação de dados. Por exemplo, enquanto controles do tipo movimentos de eixos (que requerem controle em tempo real) necessitam tempos de amostragem ou resposta de alguns poucos milissegundos, o controle do funcionamento da fábrica com análises estatísticas (operação *off-line*) não requer respostas imediatas [MOL85]. Com a grande variedade de tarefas requeridas na estrutura multifuncional da fábrica, como o exemplo de características de tempo citado anteriormente, o sistema de comunicação deve ter como embasamento, os seguintes princípios:

- ✓ subdivisão funcional do sistema em um número de equipamentos "inteligentes", com características diferentes, em função das diferentes tarefas;
- ✓ obtenção de um projeto de sistema de comunicação o mais imune possível a falhas;
- ✓ uma implementação gradual, provendo um crescimento das funções mais elementares para as mais complexas;
- ✓ um crescimento reduzido dos custos em *hardware*, visto que cada unidade está mais próxima de sua função.

O objetivo principal do projeto é viabilizar a troca de informações com os equipamentos de teste. Inicialmente com os equipamentos de teste desenvolvidos pelo setor de Desenvolvimento de Meios de Teste, e posteriormente com equipamentos de outras procedências. A parametrização dos testes e a aquisição (COLETA) dos dados gerados pelo equipamento de teste são alguns exemplos de informações que podem ser trocadas remotamente através da rede para equipamentos de teste. Com esta ferramenta pretende-se viabilizar:

- ✓ a atualização de parâmetros de teste;
- ✓ a disponibilização de informações sobre testes;
- ✓ a disponibilização de informações sobre produção;
- ✓ a disponibilização de informações a nível de desempenho;

- ✓ a disponibilização de análise estatística dos dados;
- ✓ a disponibilização destes dados para os bancos de dados corporativos;
- ✓ a transparência de atuação para o operador do equipamento;
- ✓ a integração com novos desenvolvimentos internos;
- ✓ a ampla e amistosa análise e apresentação de resultados;
- ✓ e o equipamento de teste deve ser autônomo, devendo funcionar sem a placa de rede e caso a placa de rede esteja conectada, não deve interferir no funcionamento do equipamento.

O **RECET** permite reduzir/eliminar o esforço manual para a coleta de dados da produção. Como o processamento das informações se torna mais rápido, transparente e ágil, geramos relatórios de maneira imediata. Deste modo as ações corretivas podem ser extremamente rápidas, tanto a nível de problemas de produção seriada, como de manutenção preventiva/corretiva dos equipamentos de teste.

1.3. Arquitetura

O projeto **RECET** é formado por um conjunto de soluções integradas de *software* e *hardware* que interligará os diversos equipamentos de teste para coletar os dados gerados, incluindo um aplicativo de controle e uma rede de comunicação. O aplicativo de controle consiste em um conjunto de programas gerenciais que produzem um conjunto de relatórios e gráficos para o controle do processo produtivo e a tomada de decisões.

Figura 2. Arquitetura proposta para a rede de equipamentos de teste

A rede de comunicação para equipamentos de teste consiste de um microcomputador gerenciador interligado aos equipamentos de teste através de uma rede de comunicação industrial (Figura 2).

O microcomputador executa a função de gerenciamento da rede de equipamentos, recebendo dos equipamentos os dados de teste (AQUISIÇÃO), fazendo tratamento matemático dos dados (ANÁLISE), gerando relatórios (APRESENTAÇÃO) e possibilitando a transmissão de novos limites de teste (PARAMETRIZAÇÃO) aos equipamentos. O microcomputador pode ser conectado a outras redes, disponibilizando estes dados aos níveis superiores da planta, possibilitando assim uma maior integração dos dados. A conexão dos equipamentos de teste na rede efetua-se através de placas inteligentes de comunicação de forma a possibilitar a troca de informações com a **CPU** (*central processing unit* - unidade central de processamento) do equipamento sem interferir na performance dos testes.

1.4 Metodologia de desenvolvimento

O desenvolvimento desse projeto é dividido basicamente em duas atividades: desenvolvimento de *software* e do *hardware*. Estas atividades são bastante distintas quanto a metodologia de desenvolvimento utilizada. Na atividade de *hardware* foi utilizado o padrão de desenvolvimento utilizado na Equitel S/A, ou seja:

- ✓ Descritivo funcional da placa a ser projetada
- ✓ Diagrama em blocos

- ✓ Documentação elétrica
- ✓ Lista de peças
- ✓ Projeto da placa de circuito impresso
- ✓ Montagem
- ✓ Depuração
- ✓ Documentação final

Para a atividade de desenvolvimento do *software*, foi utilizada a metodologia de Projeto Estruturado, aplicada em todas as fases do projeto. As técnicas empregadas para obtermos o máximo no projeto estruturado foram:

- ✓ Diagrama de fluxo de dados (**DFD**)
- ✓ **GRAFCET**
- ✓ Máquina de Estados (**ME**)

Os principais objetivos a serem alcançados pela atividade de desenvolvimento são a confiabilidade e a manutenibilidade do código gerado, sem entretando negligenciar os aspectos relativos a eficiência. O código produzido deve ser compacto e de alto desempenho.

2. Características técnicas e estrutura

Nesta seção apresentamos as principais características que devem ser incluídas na placa inteligente de comunicação (**PIC**) e na placa de **CPU** do equipamento de teste. Apresentamos ainda os blocos que compõem o sistema, abrangendo o *hardware* e *software*.

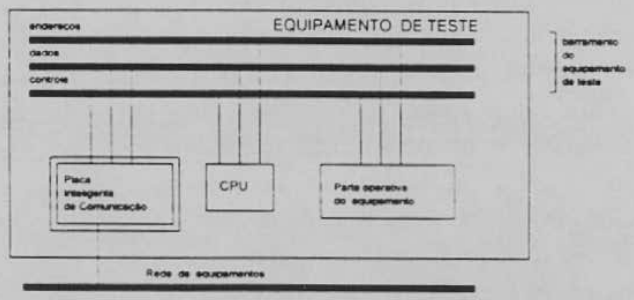


Figura 3. Estrutura do equipamento de teste conectado a rede de equipamentos

2.1. *Hardware*

2.1.1. Características

Para atender as necessidades dos equipamentos de teste desenvolvemos uma placa inteligente de comunicação que acessa a **CPU** através do seu barramento digital (Figura 3).

A placa de comunicação, apesar de ser microprocessada, terá um comportamento passivo a nível de barramento. O barramento do equipamento é controlado por um único processador, no caso, o microprocessador da **CPU** do equipamento de teste. A **CPU** executa o trabalho a que o equipamento destina-se e no momento que tiver dados a serem enviados, verifica a existência da placa inteligente de comunicação. Caso exista, repassa estes dados a placa de comunicação que encarrega-se de fazê-los chegar corretamente no microcomputador gerenciador da rede.

Na Figura 4, vemos um diagrama com os principais blocos de *hardware* que compõem a placa inteligente de comunicação. A placa de comunicação tem um sistema mínimo de suporte a um microprocessador (memórias **RAM** e **EPROM**) e a lógica necessária para a decodificação da placa. O sistema de troca de dados com a **CPU** do equipamento será através de *DUAL PORT* **RAM**, para agilizar a troca de dados entre os processadores.

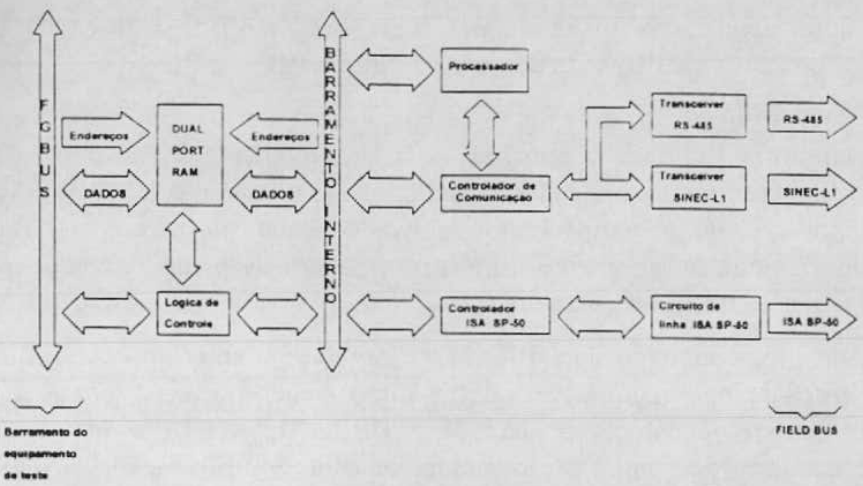


Figura 4. Diagrama em blocos do *hardware* da placa inteligente de comunicação

O *hardware* é preparado na placa para comportar os principais padrões internacionais de meios físicos de *Fieldbus*, que são : RS-485, meio físico da rede SINEC-L1 e o meio físico do *Fieldbus* ISA SP50.

O microcontrolador escolhido foi o µC8051. Os fatores principais que levaram a esta escolha foram: hardware compacto e de baixo custo, um conjunto de recursos internos suficientes para o projeto (*timers*, canal serial, endereçamento separado) e ampla plataforma de desenvolvimento.

A utilização de uma *Dual Port* **RAM** visa agilizar o processo de troca de informação (dados) entre os dois lados do sistema, o equipamento e a placa. Com esta estratégia o processador do equipamento escreve um bloco de dados e o processador da placa retira-os sem precisar de um protocolo rígido de transferência *byte à byte* através de interrupção. É a solução mais econômica, do ponto de vista de consumo, pois poderíamos utilizar **FIFOs** para agilizar essa troca, mas o seu consumo é muito grande, chegando a 200mA por *chip*, inviabilizando a sua utilização.

Utilizaremos um par trançado como meio físico para o barramento de campo, pois tem o menor custo entre todas as opções e soluciona o nosso problema uma vez que o volume de dados a ser transmitido é baixo. Como não transmitiremos programas e também não haverá a opção de *boot* remoto. Devido as características do projeto, não precisaremos de alta velocidade no meio físico e assim limitaremos a algumas opções de velocidade no meio físico: 9,6K, 31,25K e 125K bits/s.

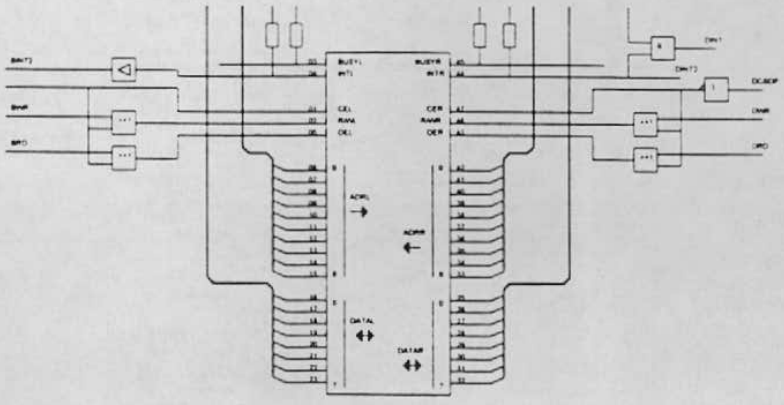


Figura 5 - Estrutura da DUAL PORT RAM

2.1.2. Estrutura

O *hardware* da placa de **RECET** possui diversas estruturas ou blocos que determinam a característica de operação da placa de rede.

Conforme citado na fase de definições, o projeto da placa é baseado no microcontrolador 8051, devido a facilidade de utilização e capacidade de recursos que este possui.

Serão utilizados diversos recursos de *hardware* para agilizar a troca de dados entre os equipamentos de testes e a **PIC** e do outro lado, entre a **PIC** e o **PC**.

O principal ponto considerado no projeto de *hardware* é a utilização de uma estrutura de *Dual Port* **RAM** (figura 5), visando a otimização da transferência dos dados. Para conseguir um desempenho satisfatório utilizamos o mapeamento em endereços desta **DPRAM** no barramento do equipamento. Assim o equipamento escreve os dados nesta região (através do uso de instruções de manipulação de memória) e o processador de placa de comunicação inteligente pode ter desta região da mesma maneira.

A estrutura de funcionamento da **DPRAM** é baseada em interrupções, sendo elaborado um esquema de troca de mensagens através destas interrupções, ou seja assim que o processador do equipamento acabar de colocar os dados na **DPRAM**, este gera uma interrupção para o processador da placa de comunicação manipular estas informações e da mesma maneira no sentido inverso.

A placa tem um sistema de reinicialização (***RESET***) composto de de duas partes, o *POWER-ON* e o *POWER-DOWN*. No instante do acionamento o circuito que faz o controle do ***RESET*** é o **VP1000**, que além de liberar o processador num instante onde está garantido que todo o sistema esta energizado, faz também o *WATCHDOG*.

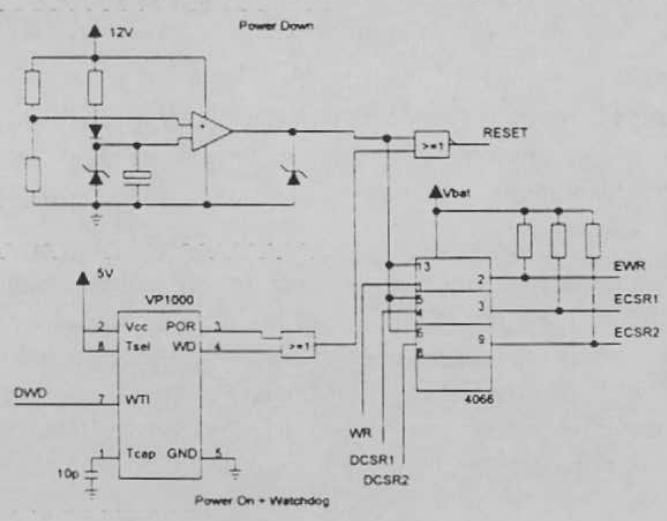


Figura 6 - Estrutura do sistema de RESET

No momento de desligamento temos o controle através de um sistema de comparação, que mantém todos os **CIs** das memórias de dados e o pino de **WR** desligados, impedindo que os dados armazenados nelas sejam destruídos (figura 6).

2.2. *Software*

2.2.1. Cararcterísticas

O *software* está organizado em dois grandes blocos (figura 7), um que executa na **CPU** do equipamento de testes e outro que executa na placa de comunicação. Como a placa de comunicação é dependente do equipamento de teste no que diz respeito ao fluxo de dados no barramento interno, é necessário a inclusão de uma camada de usuário nos dois blocos de *softawre*, para suportar a troca de dados entre ambos.

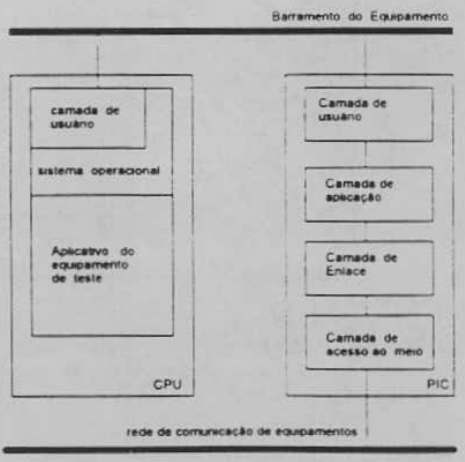


Figura 7. Estrutura em camadas do *software* de comunicação do equipamento de teste

Para que a placa de comunicação seja independente do equipamento de teste no que tange ao fluxo de dados na rede, ela deve ter todas as camadas definidas no padrão *fieldbus*, mais a camada de usuário, que

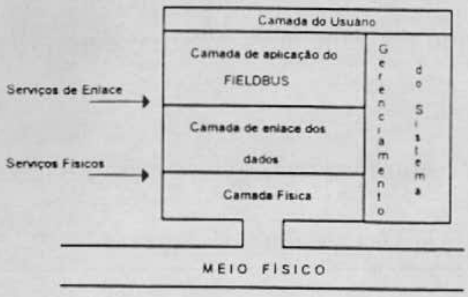


Figura 8. Estrutura em camadas do *Fieldbus*

é responsável pela comunicação de placa com aplicativo do equipamento de teste ou do microcomputador que faz a gerência da rede.

A camada de usuário contém a interface entre a placa de rede e o equipamento na forma de funções que são ativadas pelo aplicativo quando este quer se comunicar. Como a placa de rede é autônoma, o *software* aplicativo do equipamento contém apenas as chamadas de transferência de dados para a placa ou chamadas que retornam o *STATUS* da **PIC** no instante em que a inferência foi requisitada, sem qualquer interrupção no fluxo de processamento principal do código de teste.

As camadas de *software* da placa de comunicação deste projeto estão baseadas no *Fieldbus Reference Model* (**FRM**) que definem a arquitetura de comunicação para sistemas. O FRM é um padrão internacional, baseado no **RM-OSI** (*Reference Model - Open Systems Interconnection*), que subdivide e padroniza o sistema de interconexão em um conjunto de camadas, conforme ilustrado na Figura 8 [ISA91-2].

Todas estas camadas serão baseadas no padrão ISA-SP50 que segue o **FRM** e provê os seguintes serviços para uma rede de chão de fábrica [ISA91-1]:

- ✓ as primitivas e eventos de cada serviço;
- ✓ os parâmetros associados a cada ação primitiva e seus eventos e a forma que devem ter;
- ✓ os inter-relacionamentos entre eles e as seqüências válidas para esse eventos e ações.

As funções a serem disponibilizadas aos usuários da placa de rede serão as seguintes:

- ✓ solicita *STATUS*
- ✓ envia dados
- ✓ envia parâmetros
- ✓ verifica erros

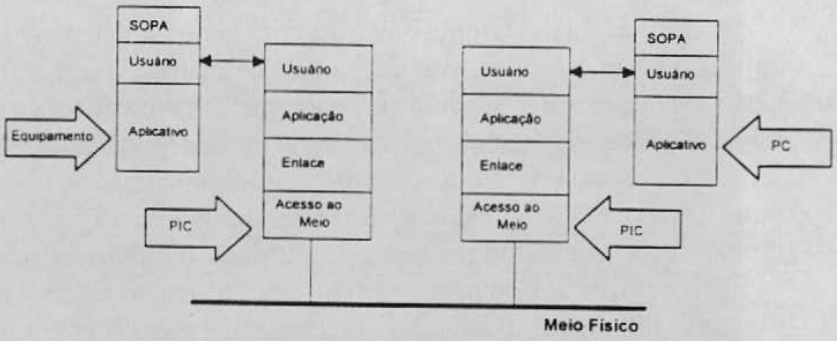


Figura 9. Diagrama simplificado da estrutura de *software*

Basicamente, faremos dois tipos de comunicação: uma entre a **PIC** e o equipamento de teste e uma entre duas **PICs**. O primeiro interfaceamento será feito usando um padrão desenvolvido durante o projeto, resolvido diretamente pelo *hardware* desenvolvido. Para a comunicação **PIC-PIC** usamos um subconjunto de funções do **FMS** (*Fieldbus Messages Services*) da SP50 [ISA91-2], que é um derivado do **MMS** (*Manufacturing Messages Services*).

É importante resaltar os requisitos exigidos pelo *software* : eficiência, visto que não deve interferir no tempo do teste; facilidade de uso e interfaceamento, para que outros projetos existentes possam utilizá-los; otimização do tamanho do código, uma vez que temos limitação de capacidade de memória no 8051; padronização, como vamos desenvolver *software* em diversas plataformas diferentes teremos que utilizar padrões que existam em todas elas. A Figura 9 mostra o diagrama da estrutura de software do sistema.

2.2.2. Estrutura

A parte de *software* do projeto **RECET** é dividido em camadas conforme definida na fase de especificação.

I - Camada do Usuário

No *software* da **RECET** a camada de usuário é responsável pela troca de informações, ou seja, é o ponto de entrada dos dados no sistema de rede. Esta camada faz o interfaceamento entre o aplicativo de teste e a rede propriamente dita, permitindo que sejam passados os dados dos equipamentos de testes aos níveis superiores da rede corporativa.

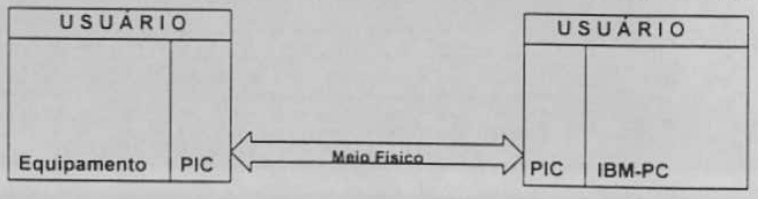


Figura 10 - Posição da camada do usuário

Os comandos que serão executados pela camada de usuário são:

para a camada de usuário do equipamento de teste

- geração de alarmes; - recebimento: parâmetros de teste e horário; - transmissão: dados dos testes, *status* do equipamento, ocupação do *buffer* da placa e parâmetros de teste.

para a camada de usuário do **IBM-PC**

- geração de alarmes; - transmissão: parâmetros de teste e horário; - recebimento: dados de teste, *status* do equipamento, ocupação do *buffer* da placa e parâmetros de teste.

a) geração de alarmes

A camada de usuário da placa que ficará no equipamento de teste deverá indicar para o aplicativo que a esta utilizando o nível de ocupação dos seus *buffers* de transmissão,.isto porque os equipamentos de testes não armazenam os dados coletados. Caso o servidor não esteja coletando os dados, estes *buffers* podem encher e os dados começar a serem sobrepostos, uma vez que os *buffers* são organizados em uma estrutura circular. Esta indicação também deverá ser repassada ao servidor para que se possa fazer um monitoramento em tempo real da ocupação dos *buffers* nos equipamentos. Caso algum equipamento esteja com sobrecarga de informações, o servidor poderá requisitar dados deste em maior quantidade, aliviando assim os seus *buffers*, sem prejudicar o atendimento aos demais equipamentos e aumentando a performance geral do sistema.

b) recebimento

Este serviço é responsável pela coleta de dados dos diversos equipamentos de testes que estejam interligados na rede **RECET** e, também, é responsável pelo tratamento dos serviços de parametrização dos programas de testes dos equipamentos e a garantia da sincronização dos relógios das diversas placas de rede que estejam na rede. No **PC** este serviço deverá tratar também os *bytes* de *status* e ocupação para que o programa gerenciador seja capaz de manipulá-los e seja capaz de identificar qual dos nós da rede (equipamento de teste) estão sobrecarregados para que este seja atendido com maior freqüência.

c) transmissão

Este serviço é responsável pela comunicação de dados propriamente dita. Implementa funções tais como a transmissão dos dados de teste, do *status* do equipamento, da ocupação do *buffer* e dos parâmetros que estão definidos para aquele teste.

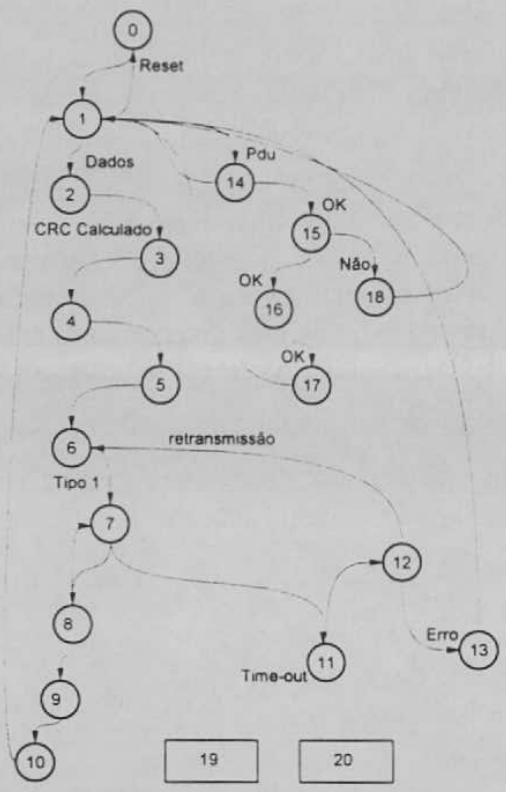
II - Camada de Enlace

A camada de enlace é definida para fornecer suporte à troca de mensagens críticas no tempo, originadas de e destinadas para os dispositivos num ambiente de automação. O termo "crítico no tempo" é usado para representar a presença de uma janela de tempo, na qual uma ou mais ações devem tem que ser completadas num certo tempo definido previamente. Caso essas ações não sejam finalizadas nesta janela de tempo, pode fazer com que a camada requisitante falhe.

O serviço da camada de enlace é caracterizado pelo seguinte conjunto de primitivas.

`-Dl_ConnectRequest:-` requisita a um nó da rede a sua identificação.
`-Dl_ConnectIndication:-` informa ou responde a uma requisição de identificação.
`-Dl_ConnectResponse:-` requisita os endereços dos nós.
`-Dl_ConnectConfirm:-` retorna o endereço do nó.
`-Dl_ResetRequest:-` requisita um *reset* da camada de enlace.
`-Dl_ResetIndication:-` confirma que o pedido de *reset* foi aceito.
`-Dl_ResetConfirm:-` confirma que o *reset* foi executado.
`-Dl_DisconnectRequest:-` solicita desconexão.
`-Dl_DisconnectIndication:-` confirma solicitação de pedido de desconexão.
`-Dl_DataPut:-` transmite dados
`-Dl_DataGet:-` recebe dados
`-Dl_DataCreate:-` cria o *buffer* de dados.
`-Dl_DataDelete:-` remove o *buffer* de dados.
`-Dl_Event:-` reporta em qual dos estados se encontra a camada de enlace naquele instante.
`-Dl_FaultIndication:-` reporta as condições de erro da camada de enlace.
`-Dl_Bind:-` provê suporte para tratamento das seções críticas do sistema, travando o recurso até a liberação pela rotina `Unbind`.
`-Dl_Unbind:-` libera o travamento dos recursos feitos pela rotina `Bind`.

O diagrama seguinte representa a máquina de estados da camada de enlace.



0 - Inicialização
1 - Idle (esperando)
2 - Calcula CRC
3 - Monta DLPdu
4 - Transfere dados para os *buffers* da camada de enlace
5 - Trava *buffer* da DLPdu
6 - Urgência define o tamanho da Pdu
7 - Libera pacote de acordo com a urgência
8 - Testa o envio de todos os bytes
9 - Libera *buffer*
10 - Avisa aplicação
11 - Verifica *time-out*
12 - Retransmissão
13 - Avisa aplicação
14 - Verifica tamanho
15 - Verifica CRC
16 - Monta APdu
17 - Libera APdu
18 - Avisa erro para o transmissor
19 - Evento atual
20 - Tratamento de erro

III - Camada Física

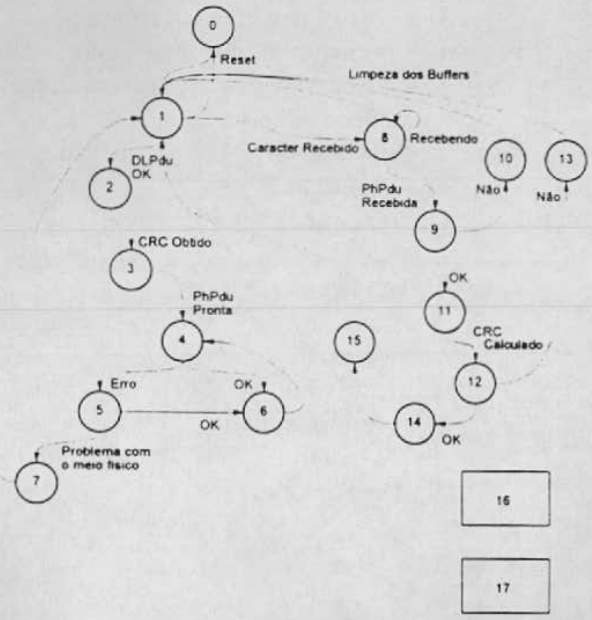
A camada física recebe os dados da camada de enlace, faz o empacotamento, adicionando as informações que são necessárias como o preâmbulo e delimitadores, e transmite para o meio físico. Estes sinais são recebidos pelas outras camadas físicas, onde são retiradas e decodificadas estas informações adicionais. Caso a mensagem seja para esse nó, é passada para a camada de enlace.

Outra função da camada física é garantir a integridade dos dados antes da validação de erros da camada de enlace e manter a interoperabilidade entre as diversas camadas físicas que compõem a rede.

O serviço da camada física é caracterizado pelas seguintes primitivas:

- **PH_CharacteristicsIndication:**-informa as características da camada física
- **PH_DataRequest:**-requisição de dados
- **PH_DataIndication:**- confirmação de dados recebidos
- **PH_RESET:**- reinicializa todas as estruturas e funções da camada física
- **PH_SetValue:**- modifica algum dos parâmetros da camada física
- **PH_GetValue:**- retorna o valor do parâmetro especificado
- **PH_Event:**- reporta em qual dos estados se encontra a camada física
- **PH_FaultIndication:**- reporta as condições de erro que por acaso possam ter ocorrido na camada física
- **PH_CalcCRC:**- calcula o CRC do bloco de dados que passará pela camada física
- **PH_CheckCRC:**- verifica se o CRC dos dados da camada física é válido ou não
- **PH_Packet:**- empacota os dados com as informações da camada física
- **PH_Unpacket:**- retira as informações colocadas pela camada física

A máquina de estados da camada física é representada pelo diagrama a seguir:




```
0 - Inicialização
1 - Idle (esperando)
2 - Calcula CRC
3 - Monta Pdu
4 - Envia 1 byte
5 - Retransmissão
6- Avança para o próximo byte
7 - Temporização
8 - Monta Pdu
9 - Verifica endereçamento
10 - Descarta dados
11 - Calcula CRC
12 - Compara CRC
13 - Pede retransmissão
14 - Envia OK!
15 - Monta buffer.
16 - Evento atual
17 - Tratamento de erro
```

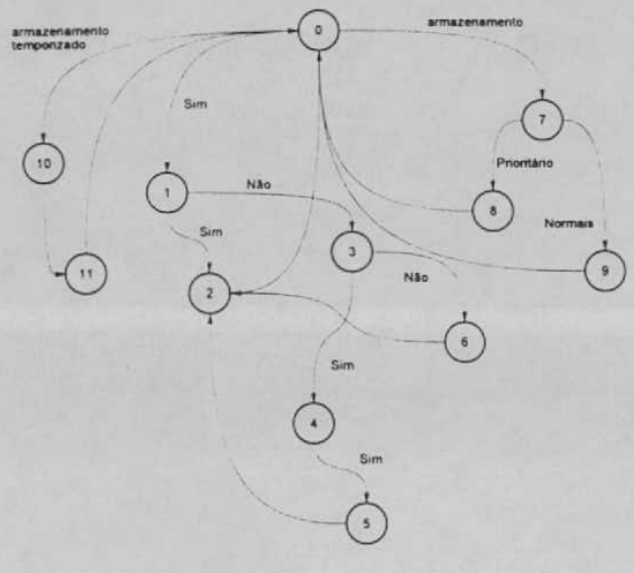
IV-Gerenciamento do Sistema

O gerenciamento do sistema é a camada que gerencia a **CPU** e a alocação dos recursos para as demais camadas. Esta camada é responsável pelo controle dos estados das demais camadas e pelo controle dos *timers* e dos *buffers* de memória.

O serviço do gerenciamento do sistema é definido pelas seguintes primitivas:

- **MS_CreateInstance:**- coloca o próximo comando da referida máquina de estados na fila de execução do sistema de gerenciamento.
- **MS_DeleteInstance:**- retira da fila de execução o comando armazenado anteriormente, mantendo os demais comandos na ordem em que devem ser executados.
- **MS_GetInstanceAttribute:**- retorna as características do comando armazenado.
- **MS_SetInstanceAttribute:**- altera as características do comando armazenado.
- **MS_DefineBuffer:**- aloca dinâmicamente áreas de memória (*buffers*) para serem utilizadas pelas rotinas da **BIOS** de rede.
- **MS_DeleteBuffer:**- devolve a área de memória alocada ao banco de memória do sistema de gerenciamento.
- **MS_CreatTimer:**- armazena um comando na fila de comandos temporizados que será executado quando transcorrer o tempo especificado.
- **MS_DeleteTimer:**- retira um comando armazenado na fila de comandos temporizados antes que este seja executado.
- **MS_Scheduler:**- controla a execução do sistema de gerenciamento.

Segue a máquina de estados do sistema de gerenciamento



0 - Testa se existe comando
1 - Comando prioritário
2 - Executa comando
3 - Verifica temporização
4 - Existe algum *time-out*
5 - Acerta fila de comandos temporários
6 - Verifica fila de comandos
7 - Verifica se é um comando prioritário ou normal
8 - Coloca na fila de comandos prioritários
9 - Coloca na fila de comandos normais
10 - Coloca na fila de comandos temporizados
11 - Ordena fila de comandos temporizados

3. Implementação

Nesta seção mostramos as principais características do *hardware* do qual o projeto **RECET** será composto.

3.1. *Hardware*

Para a implementação da placa inteligente de comunicação foi necessário observar uma série de compromissos e limitações a nível de mecânica, a nível de dimensões dos cartões, a nível de barramento elétrico dos cartões, devido a necessidade de padronização com outros produtos da empresa.

3.1.1. Padrão Mecânico

O padrão mecânico utilizado é o *rack* 19", definido pela norma **ASA** C.83.9 (**IEC** publ. 297) em correspondência à norma **DIN** 41494. Dentro deste padrão, a maioria das aplicações utiliza a seguinte composição: *rack* 19" formato duplo *Eurocard*, comportando 14 cartões de dimensões 233,4 mm x 160,0 mm.

Desta forma, a implementação da **PIC** está restrita a um cartão de circuito impresso de dimensões 233,4x160,0 mm e a altura deste cartão não pode superar a 5,08 mm, que é a distância entre os cartões.

3.1.2. Padrão Elétrico

Como padrão elétrico foram definidas as seguintes características: as placas possuem dois conectores *euro* na parte posterior sendo o conector superior destinado a conexão do barramento do equipamento e o conector inferior livre para o uso de acordo com a necessidade de cada projeto. Este padrão segue a descrição mecânica do barramento **VME-BUS**.

No caso da placa **PIC**, o conector superior é usado para a interface com o barramento do equipamento e o conector inferior não é utilizado.

3.1.3. Barramento FG-BUS

O barramento padronizado pelo setor é um subconjunto do **VME-BUS**. As adaptações feitas foram a limitação do barramento de endereços para 16 *bits*, a limitação do barramento de dados para 8 *bits* e a limitação de conter apenas uma **CPU** para a geração dos sinais de controle. Com essas limitações, o barramento pode ser comportado em um conector *euro* com 2 fileiras de pinos (A e C), possuindo 64 linhas disponíveis.

A placa **PIC**, para estabelecer comunicação com a **CPU** do equipamento de teste, teve que utilizar esta definição, ocupando 2K endereços a nível de barramento **FG-BUS**.

3.1.4. Conectores

Seguindo a linha de padronização, procura-se sempre que possível a utilização dos *euro* conectores, devido as suas características mecânicas e densidade de pinos. Devido também a sua indicação para uso no sistema *Eurocard*, dispomos de diversos fabricantes nacionais que fornecem regularmente este produto.

As especificações obedecem as normas **DIN** 41612 versão C e **IEC** 603/2. Os conectores possuem corpo em poliéster reforçado com fibras de vidro e anti-chamas, com terminais em liga de bronze e acabamento em ouro.

Utilizamos este sistema de conectores na **PIC** para efetuar a conexão com o *back-plane* e entre a **PIC** e a placa de *driver* de rede.

3.1.5. Circuito Impresso

As placas de circuito impresso, projetadas e confeccionadas para o setor, apresentam algumas particularidades que são descritas nos próximos itens. Normalmente as placas são dupla-face com furos metalizados e o impresso em classe B.

A placa de circuito impresso possue um base de fibra de vidro com espessura de 1.6 mm. É utilizada a placa de dupla-face cobreada, tendo cada camada de cobre uma espessura de 1 onça, que corresponde a 35 micras (1 micron = 1μm).

O projeto do circuito impresso da placa **PIC**, foi desenvolvido utilizando a classe B, que define a distância mínima entre trilhas. Nesta classe, a distância entre trilhas e entre trilhas e furos metalizados deve ser igual ou superior a 12 mills (1 mill = 0.001 polegada).

Esta classe de impresso permite que entre dois terminais de um circuito integrado passe uma trilha.

Após o projeto do circuito impresso em sistema **CAD**, é gerado um arquivo de furação que fornece os dados para uma furadeira **CNC** executar a furação da placa virgem de circuito impresso dupla-face.

Em seguida, é executado o processo de metalização dos furos, que consiste de processos físico-químicos para deposição de cobre nas paredes dos furos. As máscaras do lado de componentes e solda que contêm as trilhas de interligação dos componentes, são geradas pela plotagem em papel poliéster.

Com as máscaras em poliéster, são gerados diasos para permitir a transferência destas imagens para o circuito impresso por processo foto-químico, seguindo para o processo de corrosão.

Na corrosão, o cobre que não está protegido, é retirado quimicamente da base de fibra de vidro. Segue a deposição, via banho, de uma liga de estanho-chumbo sobre toda a placa para facilitar o processo de soldagem, na hora de montagem e para evitar a oxidação do cobre.

A próxima etapa consiste em processos serigráficos para acabamento, sendo feitas as máscaras de solda e componentes, que protegem as trilhas do circuito impresso e a serigrafia da máscara de componentes com indicativo para a montagem. Resumidamente, este é o processo de confecção de placas de circuito impresso.

3.1.6. Disposição Mecânica

Conforme a especificação inicial, foi mantida a característica da **PIC** de possuir mais de um meio físico para a rede de comunicação. Devido a densidade de componentes, não foi possível projetar uma placa contendo todos *drivers* de rede. A solução adotada foi o desenvolvimento de

uma placa base contendo todos os circuitos possíveis da placa de rede, e uma placa de *driver* de rede contendo todos os circuitos particulares àquela rede.

Desta forma, quando desejarmos trocar a implementação da rede de comunicação, a nível de *hardware*, somente trocamos a placa de *driver* de rede que está montada na placa base. Assim foi possível delimitar a placa inteligente de comunicação no formato desejado (233,3x160,0 mm), ocupando uma posição no bastidor do equipamento de teste.

Na figura 11 vemos a disposição das duas placas, a placa base e placa do *driver* de rede SP50 num cartão padrão. A conexão entre as placas base e de *driver* é executada através de um mini-conecetor com duas filas de 16 pinos. Este conector contêm linhas de alimentação para os *drivers*, possue o barramento de dados da placa base, linhas decodificadas de endereços, sinais de controle de escrita e leitura, linhas de interrupções para a placa base, e as linhas do canal serial do microcontrolador.

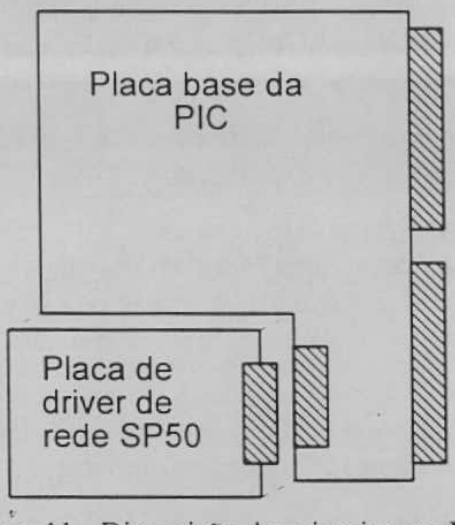


Figura 11 - Disposição dos circuitos na **PIC**

Verificamos que dispondo destes sinais é possível executar o interfaceamento da placa base com os outros *drivers* de rede a serem desenvolvidos.

A placa base contêm toda a estrutura de *hardware* da **PIC**, exceto os circuito de *driver* de rede. Com a estratégia adotada de possuírmos uma placa com o *driver* de rede, a placa base tem uma forma irregular, apesar de ter as medidas externas no formato duplo *eurocard* com profundidade de 160,0 mm.

A Placa de *driver* de rede SP50 de dimensões reduzidas (80x110 mm), contém todos os circuitos necessários para interligar o *driver* de rede (*chipset* **SMAR FB1050**) *Fieldbus* **ISA** SP50, ao barramento da placa base.

O conjunto da placa base e o *driver* de rede compõem a placa inteligente de comunicação.

A conexão desta placa com a placa base é executada via mini *euro* conector fêmea com 32 pinos. Foram previstos na placa, alguns pontos de teste, devido a importância de alguns sinais do *driver*, também, alguns LEDs foram colocados, que podem fornecer indicação de alguns estados ou situações.

A conexão do *driver* de rede ao meio físico propriamente dito foi executado de acordo com a camada física da SP50, que determina a utilização de um conector DB-9 macho no equipamento.

Esta placa foi desenvolvida para atender as especificações da camada física do *Fieldbus* **ISA** SP50. Para esta implementação foi utilizado o *chipset* da camada física, o **CI SMAR FB1050**.

Este componente está na forma de encapsulamento PLCC com 44 pinos. Foi concebido para ser interfaceado com o microprocessador, possuindo um endereço de escrita para transmitir dados, um endereço de leitura de dado recebido, linhas de *STATUS* e a saída e entrada de sinal serial modulado em código *Manchester*.

O *chip* ainda gera sinais de interrupção de *buffer* de transmissão vazio e *buffer* de recepção cheio. Esta placa possue dimensões reduzidas e acopla-se a placa base, via mini *euro* conector, formando assim a placa **PIC** com *driver* para rede SP50.

3.1.7. Interfaceamento com o barramento FG-BUS

A **PIC** troca informações com a **CPU** do equipamento através de *DUAL PORT* **RAM**. O circuito integrado **IDT 7130** foi o *chip* utilizado, sendo sua capacidade de 1 *Kbytes*. Devido a limitação do *chip* de não permitir o acesso ao mesmo tempo pelos dois barramentos, decidimos utilizar dois *chips*, ficando definido que um é utilizado para transferir dados da **CPU** para a **PIC** e o outro é utilizado no sentido inverso de transferência. Desta forma, conseguimos um comportamento *full-duplex* de troca de dados.

A nível de barramento **FG-BUS**, a **PIC** ocupa 2K endereços que podem ser selecionados entre sete faixas de endereçamento, ficando a critério do usuário a definição da faixa de endereçamento. A nível do barramento interno da **PIC**, os *DUAL PORT* **RAMs** estão mapeados nos dois últimos segmentos de 1K da página de dados do microcontrolador 8051.

3.1.8. Memórias

Conforme a especificação, a **PIC** foi dotada de 64K *bytes* de memória de programa em **EPROM** e 62 *Kbytes* de memória de dados em **SRAM** mantidos por bateria.

Os dois últimos *kilobytes* da memória de dados é ocupado pelos *DUAL PORT* **RAM**, de forma a agilizar a troca de dados.

Foi desenvolvido uma página para alocar os dispositivos de **IO** sendo que esta é sobreposta a página de dados. Isto acarreta um cuidado especial nas rotinas de acesso a **IO** e nas rotinas de interrupção, evitando que ocorram *deadlock* no microcontrolador.

Como dispositivos mapeados na página de **IO**, estão uma **EEPROM** de 8K ou 32 *Kbytes* ou uma *FLASH* **EPROM** de 32 *Kbytes*; o relógio de tempo real (CI MK 48T02), LEDs indicativos, e chaves para indicar programação por hardware.

3.1.9. Dispositivo de Lógica Programável

Foi necessário a utilização de dispositivos de lógica programável para executar a geração de linhas de controle.Uma primeira **PAL**, foi utilizada para endereçar os **DPRAMs** no barramento **FG-BUS**. Foi utilizada uma **PAL20L8**, como sendo decodificador e comparador de endereços, para a geração de linhas de *chip enable left*, *read/write left* e *output eneble left*, além de controlar a linha de direção de fluxo de sinal do **CI74LS245** que buferiza o barramento de dados.

Outros dois dispositivos de lógica programável foram necessários, para gerar as linhas de seleção dos dispositivos do barramento do microcontrolador 8051.

A denominada **PAL2**, executa a decodificação e geração das linhas de controle das memórias e **DPRAM**. A denominada **PAL3**, executa a decodificação e geração das linhas de controle dos dispositivos de **I/O**. Verificamos que as equações da **PAL2** e as equações da **PAL3**, poderiam ser implementadas por uma **PAL16L8** que seria configurada por uma entrada, informando se deve operar como **PAL2** ou como **PAL3**.

Desta forma, geramos as equações funcionais das duas **PALs** e de acordo com uma entrada que está fixa em nível lógico zero para a **PAL2** e fixa em nível lógico um para a **PAL3**, temos as correspondentes decodificações. Com este trabalho, temos na **PIC**, duas **PAL** com a mesma programação, facilitando o trabalho de duplicação e diminuindo o trabalho de documentação.

3.2. Software

O programa que implementa as rotinas básicas de entrada e saída (**BIOS**) da placa de comunicação do projeto **RECET**, foi escrito em linguagem C. Esta característica nos permitiu desenvolver e testar a maior parte do programa num ambiente **IBM-PC** mais poderoso que o ambiente disponível sobre o µc8051.

Com a utilização de uma linguagem de alto nível pudemos implementar vários pontos de qualidade de *software*, tais como: modularidade, encapsulamento, uniformidade de notação, entre outros.

A portabilidade do código foi conseguida através do uso de compilação condicionada, necessária devido a utilização de diferentes compiladores, uma vez que instruções ou funções que temos num ambiente nem sempre encontramos no outro.

A seguir detalharemos cada um dos recursos e algoritmos utilizados nos diversos módulos do projeto **RECET**.

3.2.1 Sistema de Gerenciamento

No sistema de gerenciamento fizemos uma implementação dos algoritmos definidos através de filas de prioridades representando assim todos os pontos das máquinas de estados.

Neste módulo do sistema implementamos a parte de *software* que controla o funcionamento de toda a rede. Assim foi necessário a criação de um sistema de filas com prioridades.

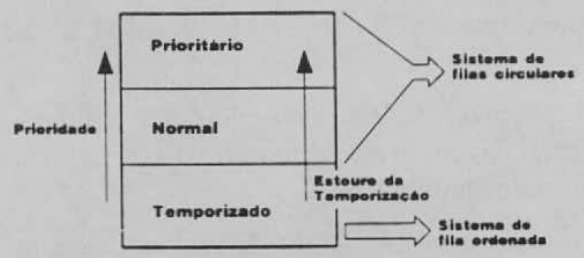


Figura 12 - Estrutura das filas do sistema de gerenciamento

Como podemos ver na figura 12, o sistema foi dividido em três camadas, sendo uma com a máxima prioridade, onde normalmente os comandos a serem executados são disparados no próximo ciclo de execução do sistema. Nesta fila são colocados apenas os comandos que tem máxima urgência na execução.

A fila de processos normais é a fila onde são colocados os demais comandos para que o escalonador do sistema se encarregue de dispará-los no momento apropriado. Esta fila funciona como uma **FIFO** circular (*First In - First Out*), conforme vemos na figura 13. Isto introduz uma primeira limitação no sistema,isto é, que só podemos ter um número finito de processos pendentes para ser executados. Este limite foi obtido através de testes, tendo sido estimado 15 entradas como o número de entradas mais adequado.

A fila de processos temporizados contém uma entrada para cada comando que deverá sofrer um atraso de tempo antes de sua execução. A estrutura adotada foi de uma fila ordenada, de modo a garantir a agilidade e desempenho no processo de escalonamento. Cada comando é inserido na fila ordenado pelo seu tempo de saída, ver Figura 14.

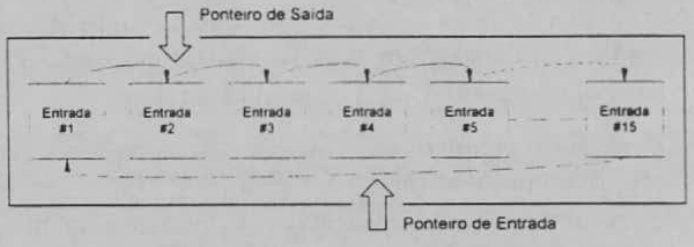


Figura 13 - Estrutura das filas circulares

Mantendo a fila sempre em ordem crescente de tempo, o escalonador precisa verificar apenas a cabeça da fila para determinar se precisa ou não executar aquele processo. Quando o processo tem o seu tempo vencido, o escalonador retira-o da fila de processos temporizados e o coloca na fila de processos prioritários. A fila de processos prioritários será testada a seguir, garantindo que o processo que veio da fila de processos temporizados seja executado no mesmo ciclo do sistema em que este foi retirado da fila dos processos temporizados.

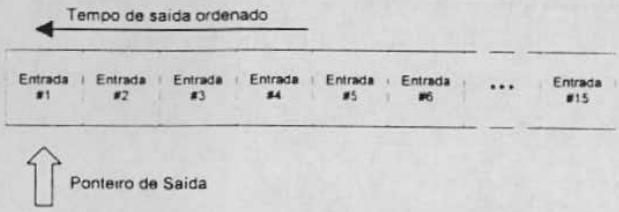


Figura 14 - Estrutura da fila ordenada

No sistema de gerenciamento devemos utilizar um relógio para controlar a temporização, e este foi implementado por *hardware*, sendo é acessado através de um circuito específico chamado de *timekeeper*. Esta implementação garante um relógio bastante confiável e com atrasos compatíveis.

3.2.2 Camada Física

Este módulo de **BIOS** de rede, assim como os demais, é dividido em duas partes, uma de rotinas básicas e outra com o sistema de controle daquele módulo. Com esta técnica podemos implementar o conceito de encapsulamento num compilador para linguagem C padrão **ANSI**, uma vez que as variáveis só podem ser acessadas pelas funções implementadas naquele módulo.

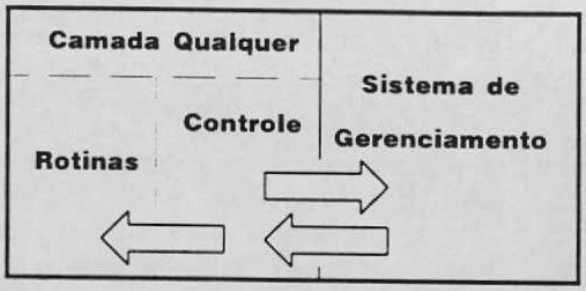


Figura 15 - Estrutura dos módulos da BIOS

A estrutura dual implementada em todas as camadas da **BIOS** de rede, é mostrada na figura 15, onde observamos que a parte de controle da camada faz uma chamada ao sistema de gerenciamento informando que precisa de algum recurso. Recurso aqui deve-se entender como: **CPU**, memória, etc.... Assim que estes estejam disponíveis, o sistema de gerenciamento informa a parte de controle através de uma mensagem única para o tipo de chamada executada, que então dispara uma das rotinas básicas que manipulam diretamente o *hardware*, ou as variáveis daquele módulo, garantindo assim que não exista conflito entre as diversas operações.

Na camada física, até as interrupções de *hardware* são transformadas em mensagens para o sistema de controle desta camada, visando garantir a eficiência no tratamento de uma interrupção.

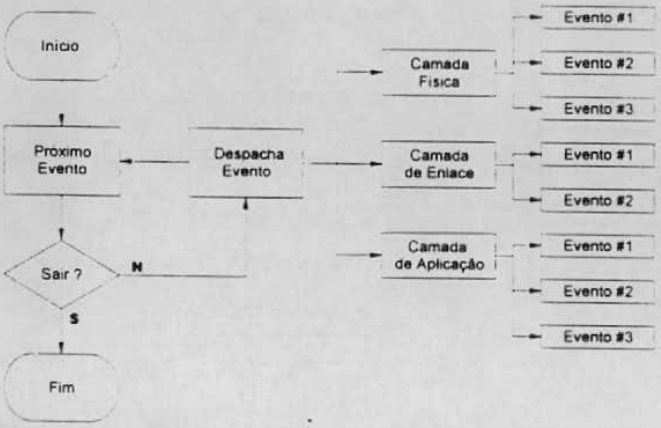


Figura 16 - Estrutura lógica de operação das camadas da BIOS

A figura 16, mostra o funcionamento do sistema de controle como um todo, através de um fluxograma de funcionamento do sistema de gerenciamento acoplado as diversas camadas do programa de **BIOS**. Como é possível observar, as camadas ficam esperando num laço infinito por uma mensagem que ao ser encontrada dispara o processo correspondente, agindo em algum dado ou fazendo algum processamento, que depois de terminado retorna para o laço infinito para esperar uma nova mensagem. Como todos os sistemas de controle das camadas estão neste mesmo processo, podemos considerar que este é um sistema multitarefa, ou seja é capaz de realizar diversos processos ao mesmo tempo.

Esta maneira de implementação garante que nenhuma das camadas ficará sem ser atendida, uma vez que a maioria dos processos têm que ser subdivididos em rotinas pequenas, que normalmente são executadas rapidamente. Como exemplo desta implementação, temos na figura 17 uma parte do código do laço de mensagens do módulo da camada física.

```
void PH_Main()
{
  switch (ucMSEvento)
  {
    case MSG_RESET:
      PH_InitAndRestart();
      MS_SendMessage(PH_Main,MSG_IDLE,0,0);
      break;
    case MSG_IDLE:
      MS_SendMessage(PH_Main,MSG_IDLE,0,0);
      break;
    case MSG_PDUMOUNT:
      PH_Packet((char*)&acBuffer,uiMSParm1,uiMSParm2);
      MS_SendMessage(PH_Main,MSG_TX,0,0);
      break;
    case MSG_TX:
      break;
    case MSG_TIMEOUT:
      PH_FaultIndication("TIMEOUT... \n");
      MS_CreateTimer(PH_Main,MSG_TIMEOUT,0,0,100);
      break;
    case MSG_RX:
      break;
  }
  PHSetEvento(ucMSEvento);
}
```

Figura 17 - Laço de mensagens da camada física

Desta maneira a organização do software de uma camada nesta estrutura dual, facilita extremamente a implementação da máquina de estados, aumentando a eficiência do processo de codificação.

3.2.3 Camada de Enlace

Na camada de enlace a estrutura dual apresentada na camada física também foi seguida, existindo assim uma divisão de módulos que implementam as rotinas de manipulação de dados, com o módulo que contêm as rotinas de controle da camada.

Uma das funções primordiais desta camada é garantir a confiabilidade dos dados transmitidos, sendo utilizado um esquema de **CRC**. Visando garantir um bom desempenho para o sistema, utilizamos o método de procura em tabelas. Para isso foi criada uma tabela com todos os **CRCs** pré-calculados no padrão **CCITT** para o polinômio de 16 bits. O programa acessa o código relativo àquele byte, não mais necessitando refazer o cálculo a todo o instante.

O resultado final deste **CRC** é a soma de todos os códigos pré-calculados individuais de cada byte componente do pacote de transmissão, limitada em 16 bits. A estrutura deste algoritmo pode ser visto na figura 18.

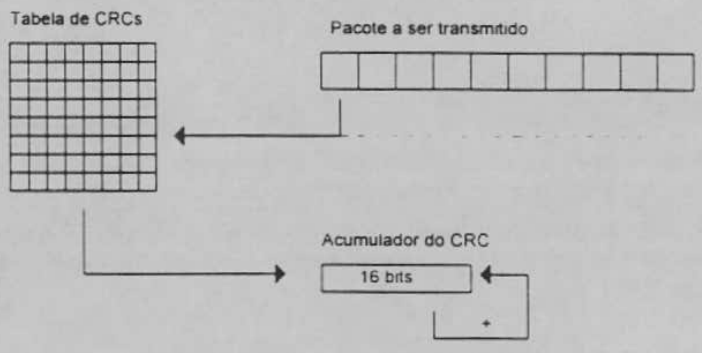


Figura 18 - Esquema do funcionamento da montagem do CRC

Na camada de enlace, bem como nos demais módulos, utilizamos variáveis estáticas, visando evitar que a manipulação de variáveis de um módulo interfira no funcionamento dos demais módulos. Assim, para que tenhamos acesso a alguma variável de um módulo, foram disponibilizadas rotinas para fazer este acesso, o que garante o baixo acoplamento entre os módulos.

4. Conclusão

O projeto RECET virá para tentar solucionar um problema que os equipamentos de testes possuem, a coleta de dados de forma rápida e eficiente. A sua implementação, atualmente, está na fase de integração do software da BIOS com a placa de comunicação (PIC), o hardware do sistema, para que assim possam ser feitos os devidos testes de funcionamento. Depois destes testes iniciais, este hardware deverá ser acoplado a um equipamento de teste para que o desenvolvimento das camadas de usuário e aplicação sejam iniciados.

Após a implantação do projeto, podemos prever que existirá um aumento da qualidade e da confiabilidade dos dados gerados, uma vez que não existiram mais as imperfeições causadas por uma coleta manual, além, é claro da velocidade em que estes dados chegarão até o ponto de tomada de decisão, justificando dessa maneira o tempo de desenvolvimento deste projeto.

Bibliografia

[BED87], Bedworth, David D.;
Bailey, James E.
Integrated Production Control Systems
John Wiley & Sons - 1987
PP: 18-58

[MOL85], Mollo, M.
A Distributed Control Architeture for a Flexible MAnufacturing System Flexible Manufacturing Systems (International Trends in Manufacturing Tech)
Bedford, UK & Berlin IFS (Publications) LTD and Springer-Verlag, 1985
PP: 153-163

[ISA91-1],ISA SP50 - 359H
Data Link Service Definition
September, 1991

[ISA91-2], ISA SP50 - 360E
Data Link Protocol Specification
1991