UNIVERSIDADE FEDERAL DA PARAIBA

CENTRO DE CIENCIAS E TECNOLOGIA

CURSO DE MESTRADO EM ENGENHARIA ELETRICA

Contribuição a Implementação de um Sistema Distribuído em Tempo Real Para Controle De Processos

ANGELO PERKUSICH

CAMPINA GRANDE, PB

DEZEMBRO - 1987

ANGELO PERKUSICH

Contribuição a Implementação de um Sistema Distribuído em Tempo Real Para Controle De Processos

Dissertação apresentada ao Curso de MESTRADO EM ENGENHARIA ELETRICA da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre.

AREA DE CONCENTRAÇÃO: PROCESSAMENTO DA INFORMAÇÃO

GURDIP SINGH DEEP

Orientador

CAMPINA GRANDE, PB

DEZEMBRO 1987

Contribuição a Implementação de um Sistema Distribuído em Tempo Real Para Controle De Processos

P451c Perkusich, Angelo
Contribuicao e implementacao de um sistema distribuido em tempo real para controle de processos / Angelo Perkusich. - Campina Grande, 1987.
88 f.

Dissertacao (Mestrado em Engenharia Eletrica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Sistema Distribuido em Tempo Real 2. Controle de Processos 3. Processamento da Informacao 4. Dissertacao I. Deep, Gurdip Singh, Prof. II. Universidade Federal da Paraiba - Campina Grande (PB)

CDU 621.3(043)

CONTRIBUIÇÃO À IMPLEMENTAÇÃO DE UM SISTEMA DISTRIBUÍDO EM TEMPO REAL PARA CONTROLE DE PROCESSOS

ANGELO PERKUSICH

DISSERTAÇÃO APROVADA EM 30/12/87

MISAEL ELIAS DE MORAES, Dr.Ing.

Maria de Fatima Q. Vieira Turnell

MARIA DE FÁTIMA QUEIROZ VIEIRA TURNELL, Ph.D.
Componente da Banca

PAULO ROBERTO CAMPOS DE ARAÚJO, Mestre
Componente da Banca

CAMPINA GRANDE-PB
DEZEMBRO - 1987

AGRADECIMENTOS

Agradeço aquelas pessoas que direta ou indiretamente contribuiram para que este trabalho pudesse ser concluído. Agradeço em especial:

AO PROF. GURDIP SINGH DEEP, pela orientação, e pelos ensinamentos.

AO PROF. MISAEL ELIAS DE MORAIS, pelo incentivo e apoio que possibilitaram que este trabalho pudesse ser concluído.

AO AMIGO FABIO CUNHA ALBUQUERQUE, pelas valiosas sugestões e principalmente pela sua amizade.

AO PROF. JOSE HOMERO FEITOSA CAVALCANTI, pela orientação.

A minha Mãe, minha esposa Lígia e aos meus filhos Mirko e Andrei.

CONTEUDO

CAPITULO 3

CAPITULO 4

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Neste trabalho propõe-se uma estrutura para um Sistema Distribuído Verticalmente Hierárquico para Controle de Processos. Introduz-se uma extensão ao Diagrama de Fluxo de Dados, como aplicado em SistemaS de Processamento de Dados, para a análise e representação de Sistemas de Automação e Controle, bem como técnicas de decomposição deste diagrama com objetivo de implementar-se o software do sistema de forma otimizada. Apresenta-se ainda a aplicação desta ferramenta, o Diagrama de Fluxo de Dados e Controle, para a análise e representação do Sistema Distribuído Verticalmente Hierárquico, bem como a contribuição a implementação do sistema proposto.

ABSTRACT

A structure of a Vertically Hierarchical Distributed Process Control System is proposed. An extension of the Data Flow Diagram as applied to data processing system is proposed for analysis and representation of Automation and Control Systems. Decomposition techniques to be applied to this diagram for optimized software implementation is also presented. The proposed Data and Control Flow Diagram is applied for the analysis and representation of the proposed Vertically Hierarchical Distributed Process Control System. The contribution for the implememtation of the Vertically Hierarchical Distributed Process Control System is also presented.

CAPITULO 1

INTRODUÇÃO

1.1. Computador em Sistemas de Automação e Controle

A aplicação de controle automático em Sistemas de Automação e Controle (SAC) remonta da década de 30 com as primeiras aplicações do controle pneumático. Já na década de 60 eram disponíveis diversos sistema de controle eletrônicos, notadamente analógicos, hidráulicos e peneumáticos. A aplicação de computadores em Sistemas de Automação e Controle teve início na década de 60, porém devido ao alto custo dos sistemas computacionais sua aplicação era proibitiva. Com o surgimento dos microprocessadores no início da década de 70 e o rápido desenvolvimento da tecnologia em microeletrônica, os computadores puderam ser aplicados em Sistemas de Automação em Controle por Computador de forma competitiva.

Este rápido desenvolvimento da microeletrônica e consequentemente dos microprocessadores proporcionou

profundas mudanças nos conceitos operacionais dos SAC. Melhores interfaces homem-máquina puderam ser implementadas, notadamente pelo uso de vídeos gráficos em salas de controle. A aplicação de técnicas sofisticadas de análise matemática tornaram-se possíveis, desta forma melhorando o gerenciamento global dos SAC [1] [2]. Não só mudanças nos conceitos operacionais ocorreram, mas também evolução de arquiteturas para SAC por computador, bem como a surgiu a necessidade de ferramentas eficientes para o projeto de software destes sistemas.

1.2. Conceitos, Elementos e Arquitetura dos Sistemas de Automação e Controle

Desde as primeiras aplicações do computador em SAC tem havido uma evolução tanto conceitual como de arquitetura para sistemas de automação e controle por computador.

Conceitualmente um computador pode ser incluído em uma malha de controle de três formas:

- "off line";
- "on line" em malha aberta;
- "on line" em malha fechada.

Quando o computador opera "off line" toda a interação é feita através do operador. Ou seja, o operador atua como elemento de entrada de dados, o computador processa estes dados e os devolve ao operador através de algum meio de saída. O operador de posse dos dados de saída

profundas mudanças nos conceitos operacionais dos SAC. Melhores interfaces homem-máquina puderam ser implementadas, notadamente pelo uso de vídeos gráficos em salas de controle. A aplicação de técnicas sofisticadas de análise matemática tornaram-se possíveis, desta forma melhorando o gerenciamento global dos SAC [1] [2]. Não só mudanças nos conceitos operacionais ocorreram, mas também evolução de arquiteturas para SAC por computador, bem como a surgiu a necessidade de ferramentas eficientes para o projeto de software destes sistemas.

1.2. Conceitos, Elementos e Arquitetura dos Sistemas de Automação e Controle

Desde as primeiras aplicações do computador em SAC tem havido uma evolução tanto conceitual como de arquitetura para sistemas de automação e controle por computador.

Conceitualmente um computador pode ser incluído em uma malha de controle de três formas:

- "off line";
- "on line" em malha aberta;
- "on line" em malha fechada.

Quando o computador opera "off line" toda a interação é feita através do operador. Ou seja, o operador atua como elemento de entrada de dados, o computador processa estes dados e os devolve ao operador através de algum meio de saída. O operador de posse dos dados de saída

efetua as ações necessárias à correção do comportamento do processo.

Quando o computador opera "on line" em malha aberta os dados são adquiridos diretamente do processo por meio de alguma interface, o computador processa estes dados e devolve-os ao operador para que este tome as ações corretivas junto ao processo.

Por fim, no modo "on line" em malha fechada o computador desempenha todas as ações, ou seja entrada de dados, processamento e atuação direta sobre o processo.

Os elementos de um sistema de automação e controle por computador básicamente são:

- aquisição de dados;
- atuação;
- controle;
- supervisão.

Estes elementos são discutidos em detalhe no Capítulo 2.

Arquiteturalmente a aplicação do computador em SAC tem evoluído desde o uso de uma única máquina central até os sistemas distribuídos de controle digital direto. Três tipos de arquitetura tem sido utilizadas nas aplicações em SAC: arquitetura centralizada, arquitetura descentralizada e arquitetura distribuída. No Capítulo 2 descrevem-se estas

três arquiteturas.

Com a evolução da aplicação de computadores em SAC surgiu também o conceito de controle de processos em tempo real. Neste caso de aplicação do computador para sistemas "on line" em malha fechada são introduzidos alguns conceitos. Basicamente estes conceitos dizem respeito a gerênciamento de processos em tempo real, execução concorrente de programas e existência de um relógio em tempo real, RTC [1] [2].

1.3. Projeto de Software para Sistemas de Automação e Controle.

Nas últimas décadas tem-se reconhecido o software como sendo um produto que deve ser projetado, implementado, documentado e mantido. Os crescentes custos para desenvolvimento e manutenção do software de sistemas tem forçado o desenvolvimento de ferramentas mais eficientes para o projeto de software e principalmente aqueles aplicados a SAC, pois estes tendem a ser mais complexos pela própria natureza da aplicação. Esta complexidade é principalmente traduzida pela necessidade de programação em tempo real onde as ações do sistema não são definidas pelo projetista mas sim pelo ambiente. Deste modo os eventes ocorrem de forma assíncrona em tempo real [2], devendo ser analizados e processados de forma crítica no tempo.

O projeto do software deve pressupor a aplicação de uma ferrameneta eficiente para a análise, implementação, documentação e manutenção dos programas de gerenciamento e aplicativos dos SAC. Devendo prover meios pelos quais possa ser definida com clareza a estrutura lógica do SAC.

Devem ainda ser disponíveis meios pelos quais possam ser empregadas eficientemente as diversas linguagens de programação disponíveis. Isto é, após a análise do SAC e definição de sua estrutura lógica, devem haver ferramentas para decomposição desta estrutura lógica, ou seja que derivem uma estrutura orientada para a natureza da linguagem ou linguagens utilizadas na programação. Estas estruturas derivadas ou decompostas podem ser as mais diferentes, desde um fluxograma até uma decomposição visando a aplicação de linguagens modulares objeto orientadas [41]. No capítulo 3 são abordados aspectos relacionados com a decomposição da estrutura lógica de sistemas.

Outro conceito importante de software para SAC são os programas gerenciadores em tempo real ou executivos em tempo real. Estes programas provêem as estruturas de controle necessárias ao gerenciamento da execução das tarefas. Este gerenciamento frequentemente é dividido em duas estruturas distintas de controle:

- o escalonador e o;
- acionador de tarefas.

O escalonador é responsável pela ordenação das

tarefas em uma estrutura de dados, que pode ser implementada por uma fila, uma lista encadeada, etc. Esta ordenação deve ser feita seguindo-se alguma regra de precedência pré determinada.

O acionador executa o acionamento das tarefas sequencialmente no tempo respeitando a informação de precedência contida na estrura de dados preparada pelo escalonador.

Pode-se classificar um executivo em tempo real segundo a prioridade de execução das tarefas ou quanto a suspensão temporária de tarefas [53].

Quanto a prioridade, um executivo em tempo real pode ser classificado como sem prioridade, quando não existe prioridade entre as tarefas. Com prioridade estática quando as prioridades são diferentes, porém fixadas a priori. Prioridade dinâmica, quando a prioridade das tarefas pode alterar-se ao longo do tempo. E prioridade mista, quando coexistem prioridades estáticas e dinâmicas.

Quanto a suspensão temporária da tarefas um executivo pode ser não preemptivo, quando cada tarefa é executada completamente, não aceitando interrupções na sua execução. Preemptivos, quando as tarefas podem ser interrompidas durante a execução, sem obrigatoriedade de ser mantida sua execução após a interrupção.

1.4. Objetivos do trabalho

Este trabalho tem como objetivo principal a contribuição a implementação de um sistema distribuído para aplicação em controle de processos. A evolução deste trabalho passa por três etapas:

i) Caracterização dos sistemas distribuídos para controle de processos;

ii) proposta de uma possível arquitetura;

iii) introdução de uma ferramenta eficiente para o projeto de software para automação e controle, o Diagrama de Fluxo de Dados e Controle, DFDC, descrito no Capítulo 3.

Na área de Automação e Controle desconhece-se a aplicação de ferramentas sistemáticas para o projeto de software. Na verdade existem diversas ferramentas disponíveis para o prejeto de software, que são citadas no Capítulo 3. Porém estas ferramentas vem sendo intensivamente aplicadas à àrea de Processamento de Dados. Na bibliografia encontrou-se apenas uma aplicação intensiva de ferramenta de projeto de software que poderia ser aplicada indistintamente tanto para ambientes de Processamento de Dados como de Automação e Controle. Esta ferramenta baseia-se no uso de Redes de Petri modificadas [54]. Porém conclui-se do trabalho que a sua aplicação a sistemas complexos resultará em uma complexidade muito elevada, além de não fornecer uma metodologia global para o projeto do software.

Organizou-se a apresentação deste trabalho de modo a mostar esta evolução e os objetivos alcançados. No capítulo 2 apresentam-se conceitos relacionados com sistemas distribuídos para controle de processos, bem como apresenta-se uma proposta para implementação de um sistema distribuído verticalmente hierárquico para controle de processos, o SDVC. No capítulo 3 introduz-se uma ferramenta eficiente para o projeto e implementação de software de SAC, bem como sua aplicação ao sistema proposto. Em seguida apresenta-se a contribuição a implementação do SDVC e finalmente apresentam-se conclusões e perspectivas futuras para este trabalho.

CAPITULO 2

SISTEMAS DISTRIBUIDOS DE CONTROLE

2.1. Introdução

Durante a evolução dos sistemas digitais de controle e aquisição de dados observa-se a necessidade de melhorar o desempenho dos sistemas de controle automático dos processos envolvidos na operação destes sistemas. Esta evolução tem levado a adoção de estratégias de controle e de arquiteturas que proporcionem um maior grau de confiabilidade bem como uma maior simplicidade no projeto do sistema de controle. Sob esta ótica podemos adotar três tipos de arquiteturas para um sistema de controle:

- Arquitetura centralizada;
- arquitetura descentralizada, e;
- arquitetura distribuída.

Ao adotar-se uma arquitetura centralizada, figura 2.1, objetiva-se que as unidades interrelacionadas constituintes do processo sejam integradas e controladas por

um mecanismo de controle e supervisão centralizado sofisticado e complexo. Este mecanismo deverá dispor de informações sobre o estado global do sistema a ser controlado.

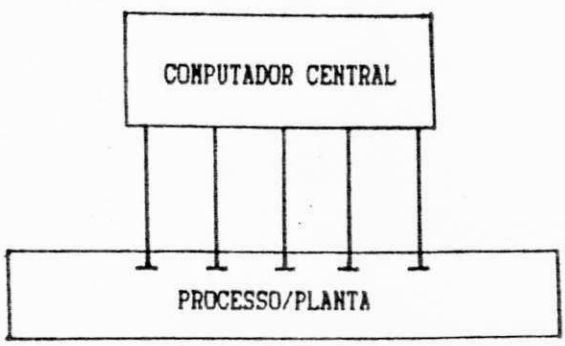


Figura 2.1. Sistema centralizado de controle.

A adoção de uma arquitetura centralizada acarreta um aumento na sofisticação do sistema, resultando em aumento do tamanho e complexidade do hardware do sistema [3]. Esta complexidade do hardware é principalmente caracterizada pela necessidade da utilização de computadores de alto custo, o que torna proibitivo e inviável a utilização de técnicas de redundância a fim de tornar o sistema tolerante a falhas. Além disto, a centralização do sistema pode fazer com que os custos de expansão tornem-se elevados, pois a introdução de uma nova malha de controle e aquisição de dados pode levar a necessidade de reprojetar parte ou todo o sistema. Os custos para a análise, implementação e manutenção do software do

sistema de controle tornam-se elevados, devido ao custo de programação, pois a centralização acarreta aumento da complexidade do Sistema de Automação e Controle como um todo. Além disto, pode haver diminuição do tempo de resposta do sistema, pois os programas de controle e aplicação serão extensos e deverão ser executados em um único processador.

Os sistemas de controle que adotam uma arquitetura descentralizada [4] totalmenente paralela, figura 2.2, caracterizam-se pela adoção de técnicas de decomposição do sistema de controle em subsistemas menores, ditos módulos [5,6]. Estes módulos executam a função de controle independentemente e sem cooperação com os outros módulos, desta forma permitindo a utilização de máquinas menores, notadamente microcomputadores [7,8]. O fato de cada processador ou unidade de controle desempenhar sua função localmente, não havendo comunicação entre os controladores pode levar ao surgimento de dificuldades na operação do sistema como um todo, uma vez que as malhas de controle são independentes. Esta independência das malhas de controle faz com que não estejam disponíveis de forma imediata os dados locais de cada malha de controle. Esta indisponibilidade dos dados locais pode ser contornada utilizando-se um sistema supervisório independente do sistema de controle, de forma que os dados locais de cada malha de controle possam ser acessados através dos bancos de dados locais de cada malha. Porém este acesso pode acarretar um aumento na complexidade

das malhas de controle, pois deverão ser criados mecanismos que possibilitem este acesso aos bancos de dados locais dos controladares.

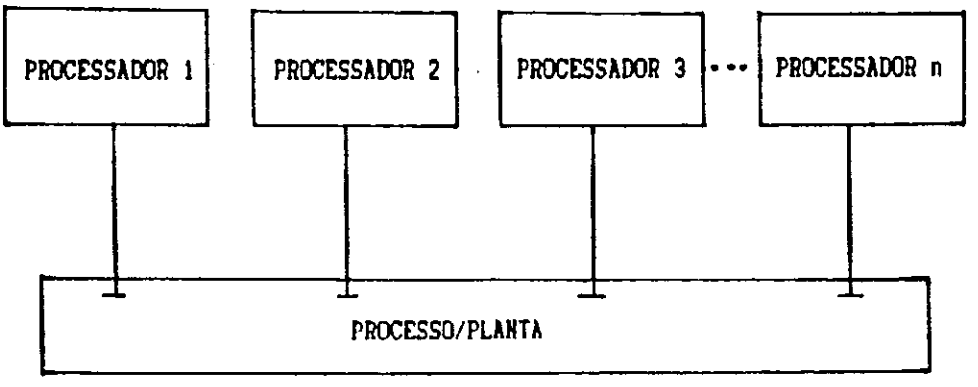


Figura 2.2. Sistema descentralizado totalmente paralelo.

Nos sistemas distribuídos de controle, adota-se uma estratégia descentralizada e cooperante para a solução do problema de controle, utilizando minicomputadores e/ou microcomputadores interligados por algum meio de comunicação [8-16].

A seguir apresentam-se as características e evolução dos sistemas distribuídos de controle de processos. Apresenta-se também uma proposta de arquitetura, bem como a caracterização de um Sistema Distribuído Verticalmente Hierárquico para Controle de Processos [17].

2.2. Sistema Distribuídos de Controle

De uma forma genérica um sistema distribuído deve

ser constituído de dois ou mais processadores separados e interligados por algum meio de comunicação. O objetivo de interligar processadores é proporcionar que um problema de controle possa ser cooperativamente solucionado. O meio de comunicação poderá ser uma rede de comunicações em estrela, em anel, ou barramento, etc [18].

Em um sistema genérico de controle distribuído, algumas funções básicas devem ser implementadas [19]. Estas funções são:

- Aquisição de dados;
- controle das variáveis do processo;
- comunicação com o operador;
- supervisão, e;
- controle de sequência.

2.2.1. Aquisição de dados

A função de aquisição de dados consiste na leitura periódica de sinais analógicos ou digitais obtidos do processo através de transdutores conectados à planta. Estes sinais representam as variáveis do processo sob controle, os quais são utilizadas pelo algoritmo de supervisão com o objetivo de monitorar o estado do processo ou gerenciar e avaliar o processo. Estas variáveis podem ser as mais diversas, tais como: temperatura, pressão, vazão, nível, posição, velocidade, etc.

Devido a natureza dinâmica dos processos os sinais analógicos dos sensores são amostrados e digitalizados. Sinais digitais vindos de chaves de limite, sinais indicando estados, etc, são lidos das interfaces e podem causar uma interrupção do processo, caso tenha sido especificado em projeto. Os sinais provenientes dos sensores são transformados para unidades de engenharia, sofrem filtragem, verificam-se as condições de alarme e são temporariamente armazenados em bancos de dados na memória para posterior processamento. Há muitas alternativas para cada passo de processamento e estas dependem do tipo do sensor, variabilidade do sinal, ruído, interferências que alterem o sinal, a importância do sensor no processo, os valores anteriores do sinal e o status atual do alarme. Uma vez que a aquisição de dados do processo é executada através dos diversos módulos do sistema, o banco de dados do sistema também pode ser distribuído.

Geralmente grandes sistemas envolvem um grande número de sinais tornando os bancos de dados extensos. Alguns sinais são calculados periódicamente a partir da combinação de outros sinais. Os sinais calculados, assim como os sinais obtidos dos sensores são armazenados em bancos de dados. O banco de dados deve ser atualizado em tempo real.

2.2.2. Controle das variáveis do processo

Nos módulos a efetivação da ação de controle envolve o valor medido ou calculado da variável do processo, o valor

de referência ("set-point") e o algoritmo de controle. O algoritmo de controle calcula as modificações no valor da variável de entrada ou variável controlada do processo a partir do valor medido ou calculado da variável do processo e o valor de referência a fim de reduzir o erro. A complexidade do algoritmo de controle varia consideravelmente devido a sua dependência de características dinâmicas do processo, como: controle com realimentação, controle em cascata, etc. Sendo assim o banco de dados deve conter os apontadores para algoritmos específicos de controle, os parâmetros de algoritmos e as informações sobre os algoritmos de controle. Um módulo pode ser constituído de um conjunto de algoritmos de controle executados em cascata no caso do controle em cascata.

A atualização dos parâmetros de controle nas malhas ocorre periodicamente, as vezes utilizando dados dos módulos remotos. Em um mesmo módulo podem ser adquiridos os dados necessários e gerados os sinais de saída para o atuador. O fato de somente alguns dados dentro de um mesmo módulo terem de ser adquiridos via módulos remotos evita atrazos de comunicação que ocorreriam quando o meio de comunicação estivesse sendo intensamente utilizado. Isto pode resultar em degradação suave do desempenho uma vez que em uma dada malha de controle o módulo pode manter o processo na condição operacional existente no momento da falha de comunicação.

2.2.3. Comunicação com o operador

Com o objetivo de tornar o sistema de controle interativo, existe a necessidade de uma comunicação centralizada com o operador, incluindo acesso a todas as informações presentes no banco de dados. Além disso, o sistema deve dispor de meios para alterar o fluxo das operações tais como: partida; alteração no valor de referência; etc. Esses meios devem possibilitar modificação dos algoritmos e configuração de controle, apresentação de condições de alarme, verificação das funções do processo e amostragem estatística das variáveis do processo. Apesar do banco de dados ser distribuído nos módulos, as cópias fiéis destes dados são disponíveis e atualizadas no módulo de comunicação com o operador e são atualizadas para apresentação na tela, ou modifição e as vezes nova partida do sistema.

A comunicação com o operador, é feita geralmente via uma tela de terminal de vídeo gráfico, teclado, som, funções especiais, tais como controle de abertura ou fechamento de válvulas utilizando caneta ótica, etc [20]. Esta comunicação permite que o sistema opere sem a necessidade de um computador de supervisão ou ainda que as funções de supervisão sejam incorporadas a outras funções.

2.2.4. Supervisão

Devido a determinados aspectos que serão oportunamente abordados, como por exemplo a têndencia atual

de implementar-se sistemas de controle multinível, os sistemas de controle modernos estão sendo estruturados hierarquicamente. As funções de alta velocidade e críticas no tempo, estão situadas nos níveis mais baixos da hierarquia e as funções mais complexas e menos críticas no tempo estão em níveis mais altos. Desta forma, as funções de aquisição e controle digital direto são alocadas em um nível mais baixo, e as funções nos altos níveis podem incluir otimização de desempenho da operação, gerenciamento de energia e de planta, escalonamento, etc. Funções de controle mais extensivas, geralmente estão residentes no computador do nível mais alto da hierarquia, de modo que os programas de aplicação tem acesso a todo banco de dados. Estes programas de aplicação variam dependendo da aplicação. Deste modo, o computador supervisor, no mais alto nível da hierarquia, é programado segundo a natureza da aplicação, em linguagem de alto nível e deve possuir um sistema operacional em tempo real.

O computador de supervisão tem como função otimizar a operação dos conntroladores da planta. Esta otimização pode vizar por exemplo minimizar o consumo de energia ou maximizar a eficiência da produção.

2.2.5. Controle de Sequência

O controle de sequência é responsável pelo estabelecimento da ordem ou sequência de ocorrência dos eventos na operação da planta. Desta forma definindo os

estágios ou fases, bem como a sequência de execução destas fases durante o processo produtivo.

2.3. Arquitetura de Sistemas Distribuídos de Controle

Como dito na secção 2.1, um sistema distribuído de controle é caracterizado pela comunicação efetiva entre os diversos componentes constituintes do sistema. Três arquiteturas são consideradas para sistemas distribuídos de controle:

- Sistema distribuído horizontalmente;
- sistema distribuído verticalmente;
- sistema misto.

2.3.1. Sistema distribuído horizontalmente

Em um sistema distribuído horizontalmente [10] todos os processadores encontram-se em um mesmo nível hierárquico interligados por uma rede de comunicação. Neste caso existe comunicação inter processadores, e não há hierarquia entre os processadores. No projeto de um sistema distribuído horizontalmente a quantidade de informações trocada ou seja as transações entre os processadores deve ser limitada, caso contrário o sistema de comunicação tende a saturar. Cada computador ou processador é responsável pelo processamento dos dados vindos de uma área bem determinada da planta, executando a aquisição de dados e enviando comandos de controle aos atuadores ligados ao processo. Os

programas dedicados estão armazenados na memória local e em cada um há necessidade de alguma informação sobre um outro sub-processo, a qual é solicitada através da rede de comunicação. Deve-se salientar o paralelismo do processamento dos dados nas diversas unidades de processamento. Na figura 2.3 mostra-se a estrutura esquemática de um sistema distribuído horizontalmente.

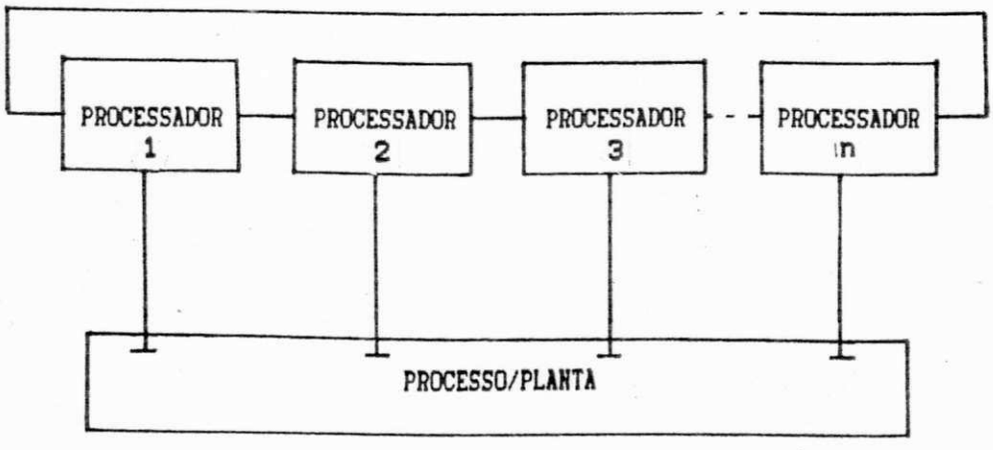


Figura 2.3. Sistema distribuído horizontalmente.

2.3.2. Sistema distribuído verticalmente

Um sistema distribuído verticalmente é caracterizado pela divisão do sistema de controle em diversos módulos ou níveis hierárquicos. Cada nível emprega um ou mais computadores com capacidade e recursos ajustados às funções específicas que desempenham. No nível mais baixo, isto é, no nível do processo e funções de controle direto, o computador desempenha as funções de aquisição de dados do processo,

envio de comandos para os atuadores, execução de algorítmos específicos de controle, ajuste de "set-point" do controlador a partir dos dados fornecidos pelo operador ou vindo do nível hierárquico superior. Segundo Yoshii e Kuroda [3] neste nível pode-se implementar as funções básicas de controle manual e de interação com o processo, bem como a comunicação com os níveis superiores.

Os computadores no segundo nível, isto é, o de coordenação, são responsáveis pelo gerenciamento de recursos para os processadores do nível mais baixo. Eles são responsáveis pelo acompanhamento do estado das variáveis, estabelecimento do ponto de operação de todos os sub-processos conectados ao sistema e finalmente sincronização do controle destes subprocessos. O processador deste nível deve possuir maiores recursos para armazenamento bem como as interfaces com o operador. Este nível deve desempenhar funções semelhantes ao computador de supervisão descrito na secção 2.2.4. A otimização da planta como um todo pode ser efetuada neste nível de hierarquia. Neste nível são determinados parâmetros para definir o ponto de operação do processo bem como os demais parâmetros necessários a obtenção de melhor rendimento da planta através de técnicas padrões. Na ausência de outros níveis mais elevados, a política global do processo produtivo da planta também é determinada por este nível. Obviamente isto leva em consideração recursos físicos instalados e disponíveis, as matérias primas, recursos humanos, fontes de energia,

demanda do mercado, etc.

2.3.3. Sistema distribuído misto

Na arquitetura mista, tem-se distribuição horizontal e vertical. Na figura 2.4, as linhas pontilhadas, linhas de comunicação, atribuem a distribuição horizontal do sistema de controle.

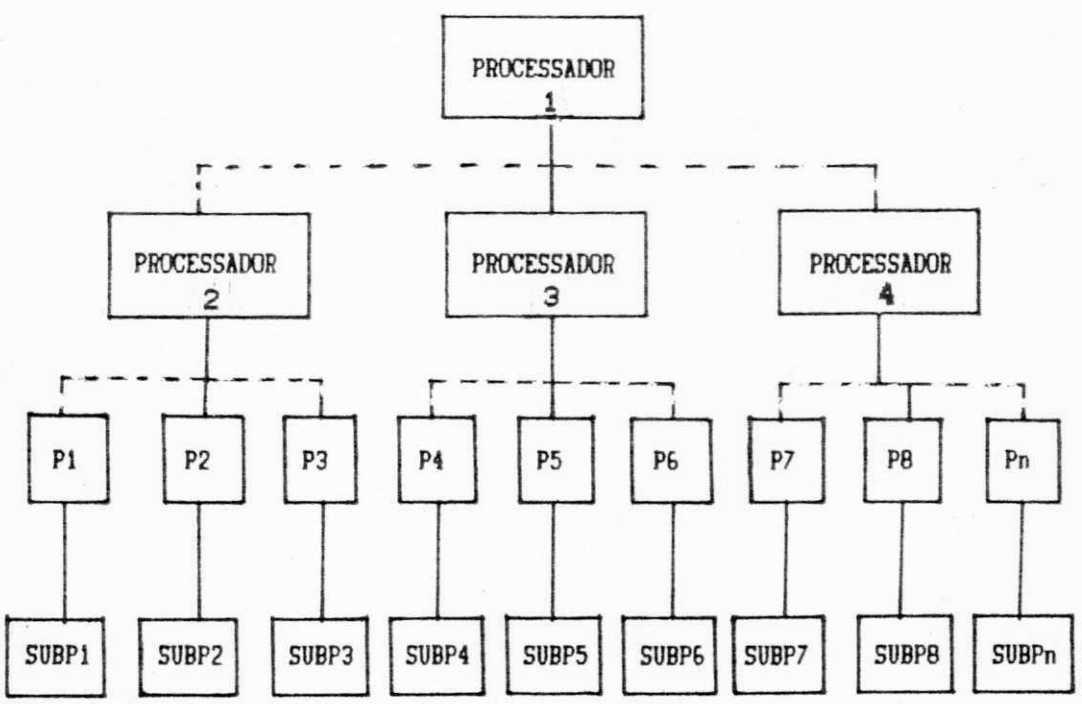


Figura 2.4. Sistema distribuído misto.

No caso de sistemas distribuídos verticalmente o número de níveis hieráquicos depende da complexidade da planta. Em princípio, utilizando-se uma estrutura hierárquica, é possível construir-se qualquer sistema de controle complexo, de maneira claramente organizada de acordo com a hierarquia desejada. Para realizar esta estrutura, a planta complexa deve ser dividida em processos parciais ou sub-processos semelhantes de tal forma que os módulos a serem implementados possam ser utilizados repetidamente no projeto do sistema.

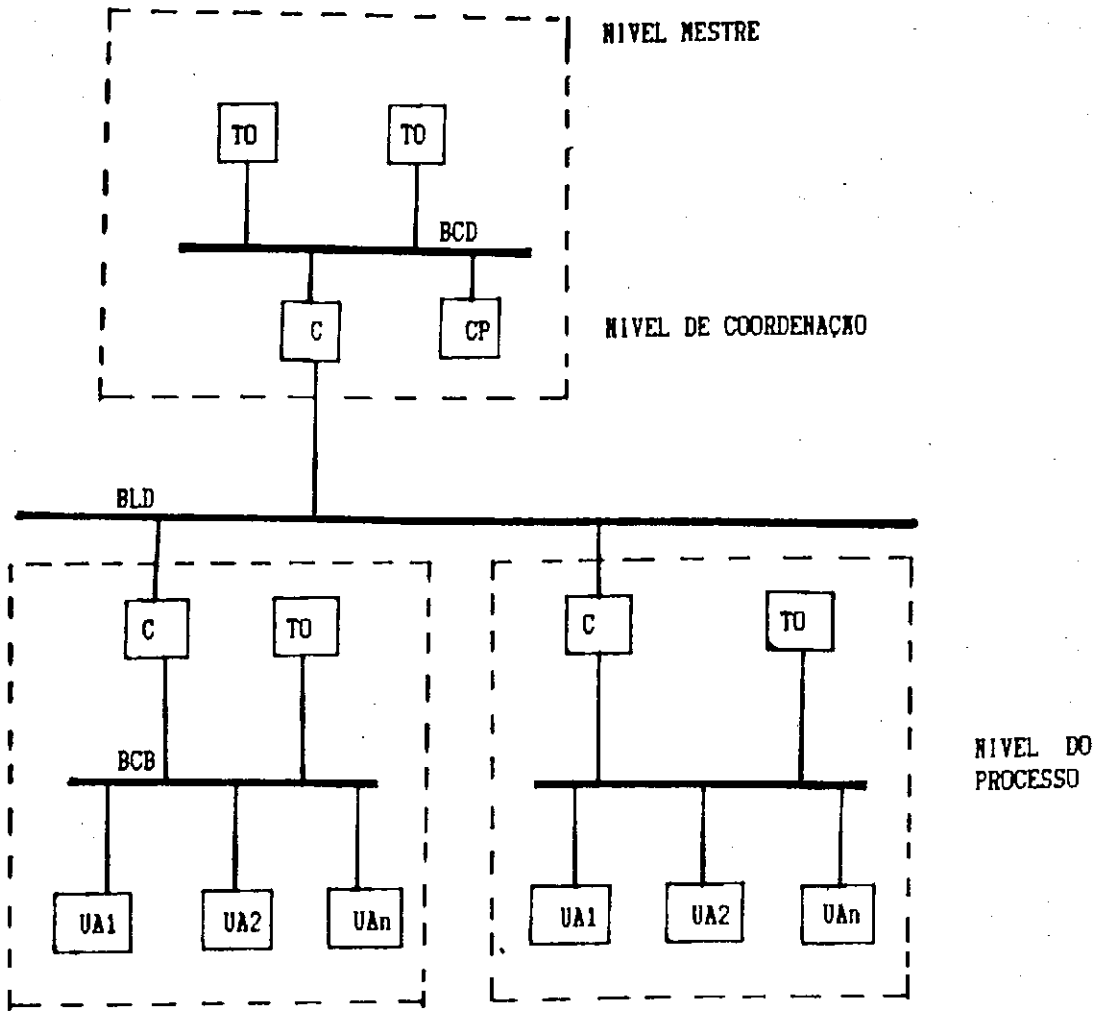
Os sistemas distribuídos verticalmente podem ser ainda classificados quanto a característica da distribuição. Viegen [8] apresenta duas formas de distribuição para um sistema vertical. Estas formas de distribuição são:

- Distribuição de funcões;
- distribuição de sistema.

No caso de distribuição de funções existe uma intensa cooperação entre os processadores, ou seja, há uma grande troca de informações entre os diversos níveis do sistema de tal modo que apenas algumas funções são distribuídas. Na distribuição de sistema, os processadores de um dado nível hierárquico processam suas próprias transações completamente, e ocasionalmente as passam aos níveis superiores.

2.4. Estrutura Ótima para Sistema Distribuídos de Controle

Uma estrutura ótima para sistemas distribuídos de controle é proposta por Fruh [21], Esta estrutura é mostrada



TO - Terminal para operador

CP - Computador de processo

C - Acoplador de barramentos

Figura 2.6. Estrutura ótima para sistema distribuído de controle de processos [21].

Esta estrutura otimizada possui as seguintes características:

i) As unidades de automação implementam as funções de aquisição de dados, controle e/ou controle de sequência. Estas unidades são multiplexadas e operam juntamente com o terminal de operador, ou um pequeno processo ou ainda um processo parcial da planta, com autonomia. Entretanto o aspecto estrutural da operação autônoma, precisa ser satisfeito somente por aquelas unidades de automação que provêem um número suficientemente grande das funções, bem como tem a velocidade e a capacidade suficiente de processamento;

ii) deve ser possível agrupar vários sub-sistemas (UA, OT, PC) dentro de uma pequena distância, entre 100 e 150 metros, e interliga-los através uma rede de comunicação de curta distância. Esta rede de comunicação deve proporcionar alta velocidade entre as transações, com o objetivo de minimizar o "overhead" causado pelo atrazo de comunicação;

iii) para interconectar várias faixas de curta distância, usa-se uma rede de comunicação de longa ou média distância, capaz de transmitir informações de um sub-sistema dentro de uma faixa de curta distância ao outro sub-sistema localizado em outra faixa.

iv) cada sub-sistema multiplexado é estruturado de tal forma que uma falha do processador central desta estrutura não influa em todas as funções associadas a este sub-sistema. Isto exige disponibilidade de redudância, e esta depende do processo e dos requisitos de confiabilidade. Essa consideração é válida tambem para as redes de comunicação de dados, as quais interligam os sub-sistemas de automação descentralizados.

E interressante observar que a estrutura de um sistema distribuído verticalmente hierárquico pode ser dividida em módulos funcionais, com características redundantes. Uma falha no sistema, em níveis superiores da hierarquia, não implica em parada total da planta. Os sub-sistemas na faixa de curta distância, continuam operando, com pequena degradação de desempenho.

2.5. O Sistema Proposto

A partir da análise anterior e utilizando os recursos disponíveis optou-se pelo projeto e a implementação de um Sistema Distribuído Verticalmente Hierárquico para Controle de Processos, o SDVC. A figura 2.6 apresenta um diagrama de blocos do SDVC. Este sistema é divido em três níveis hierárquicos com o objetivo de implemenetar uma estrutura de controle multinível [5]. Cada um destes níveis implementa funções bem definidas. No nível de processo, nível III, executam-se as funções de aquisição de dados e

controle direto sob o processo. No nível II, implementam-se as funções do controle, supervisão e coordenação. Por fim, o nível I é responsável pelas funções de otimização [22] e supervisão a nível de gerênciamento global do processo.

O sistema apresenta um alto grau de distribuição funcional, respeitando uma hierarquia vertical.

E interressante ressaltar que muitos dos conceitos aqui apresentados para sistemas distribuídos de controle são oriundos da aplicação de sistemas distribuídos desenvolvidos para ambientes de processamento de dados [23-26]. Em Harrison [27] discute-se a aplicação do modelo de Enslow [28], primariamente definido para sistemas distribuídos de processamento de dados, para sistemas distribuídos de controle. Nesta discussão conclui-se que devido a própria natureza da aplicação, que resulta em motivações diferentes, as similaridades entre as duas aplicações são superficiais, enquanto que as diferenças são bastante acentuadas.

Neste capítulo caracterizou-se os Sistemas Distribuídos de Controle de Processos. Apresentou-se ainda uma proposta para implementação de um Sistema Distribuído Verticalmente Hierárquico, o SDVC. No próximo capítulo abordam-se aspectos relacionados com a análise e implementação do software de Sistemas de Automação e Controle.

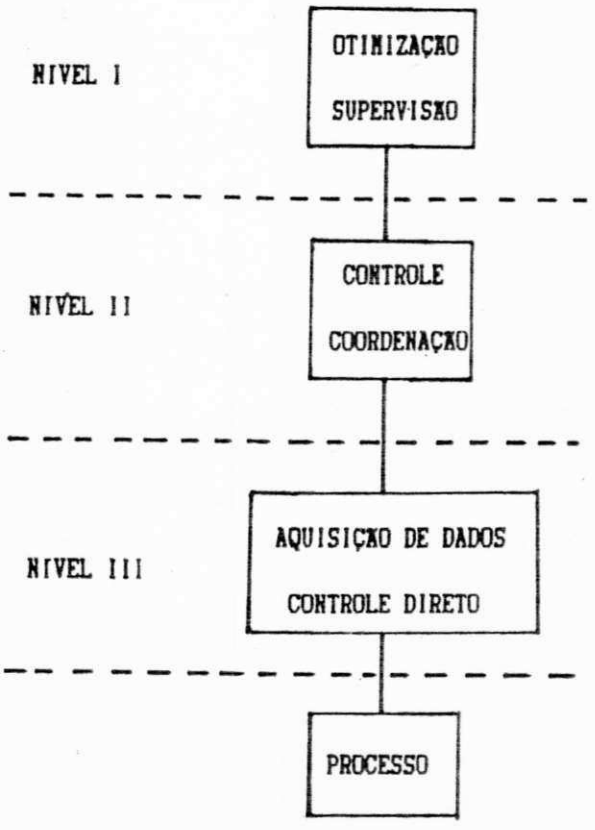


Figura 2.6. Diagrama de blocos do SDVC.

CAPITULO 3

PROJETO DE SISTEMAS PARA AUTOMAÇÃO E CONTROLE

3.1. Introdução

A crescente complexidade dos sistemas computacionais, tanto de processamento de dados como de automação e controle, levou ao surgimento de ferramentas para o desenvolvimento de modelos com o objetivo de analisar, especificar e representar a estrutura lógica destes sistemas. A partir da definição da estrutura lógica do sistema pode-se então implementar o software. Durante a década de 50 o procedimento utilizado para o desenvolvimento de programas consistia em agrupar um certo número de programadores experientes e então implementar um grande programa, sem documentação alguma ou com documentação insuficiente. O resultado obtido era um codificação extensa e inflexível. Este tipo de metodologia, ou falta de metodologia, porém tornou-se proibitiva no início da década de 60 devido ao crescente aumento dos custos de desenvolvimento de software. Na figura 3.1 [29]

pode-se acompanhar o crescente aumento dos custos de desenvolvimento e manutenção do software. Este aumento de custos levou a necessidade de desenvolver ferramentas e técnicas sistemáticas para a análise, documentação e manutenção mais eficiente e econômica de um sistema.

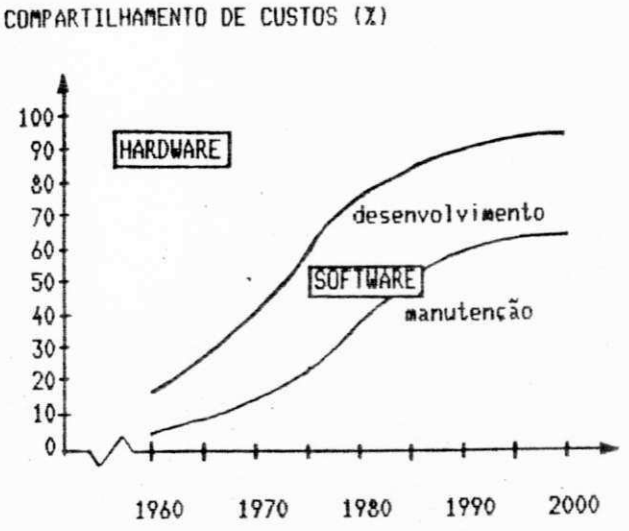


Figura 3.1. Relação custo compartilhado (desenvolvimento e manutenção) entre hardware e software.

A primeira etapa do projeto de software de Sistemas de Automação e Controle, SAC, é definir as necessidades do usuário. Essas necessidades estabelecem as especificações e requisitos do SAC e os conceitos de software envolvidos. Baseado nos objetivos de projeto, necessidades de desempenho, documentação, especificação das necessidades de entrada e saída e interfaces, o projetista de software pode estabelecer a estrutura lógica do sistema. A partir da decomposição da estrutura lógica do sistema em módulos,

pode-se avaliar, refinar ou modificar a estrutura lógica do sistema. Após a obtenção de uma estrutura decomposta que satisfaça as necessidades primárias do projeto pode-se então, a partir desta estrutura decomposta, implementar os diversos módulos obtidos utilizando-se linguagens de programação.

Existem hoje disponíveis uma série de ferramentas que permitem a análise do software de sistemas [30,31], tais como análise funcional [32,33], projeto estruturado (metodologia de Jackson [34]), Diagrama de Fluxo de Dados [35], e Cálculo de Programas [30]. Uma destas ferramentas, o Diagrama de Fluxo de Dados (DFD), tem sido extensivamente utilizado para a análise de sistemas em ambientes de processamento de dados, e em algumas aplicações de engenharia de modo geral [36]. Entretanto as definições dos elementos notacionais para o DFD não permitem representar interações existentes entre as ações de controle, assim como temporizações existentes em Sistemas de Automação e Controle. Ward [37] e Booch [38] propõe a utilização do DFD, com a introdução de novos elementos notacionais, para a representação de eventos assíncronos e temporizações. No caso de Ward é utilizada a mesma notação gráfica introduzida por DeMarco [39], incluindo alguns novos elementos na notação, que permitem a aplicação desta ferramenta para a representação de eventos assíncronos e temporizações.

Neste capítulo introduz-se uma nova ferramenta na área de Sistemas de Automação e Controle, o Diagrama de Fluxo de Dados e Controle (DFDC) [40], que permite não só a representação, mas também a análise e documentação da estrutura lógicas destes sistemas. Neste capítulo apresenta-se também duas técnicas de decomposição para o DFDC, uma utilizando decomposição processo orientada [30,31] e outra utilizando decomposição dados orientado [38]. Apresenta-se ainda a aplicação desta ferramenta para a análise e representação do Sistema Distribuído Verticalmente Hierárquico para Controle de Processos (SDVC) proposto no Capítulo 2.

3.2. Diagrama de Fluxo de Dados e Controle: Elementos Notacionais e Aplicação

3.2.1. Elementos notacionais do DFDC

Para o DFDC utilizou-se a mesma representação gráfica adotada por Gane & Sarson para o Diagrama de Fluxo de Dados. Adotam-se para a nomenclatura do DFDC alguns dos elementos notacionais introduzidos por Ward, além da introdução de mais um elemento a notação, as entidades externas, elemento este não considerado na notação por Ward, e de grande importância para a decomposição do DFDC. Diferente de Ward, que adota a representação gráfica sugerida por DeMarco para o diagrama de fluxo de dados, optou-se por adotar uma representação baseada na sugerida por Gane & Sarson por dois motivos. O primeiro é devido a

própria legibilidade do diagrama, que é mais clara utilizando-se retângulos e não círculos ou elipses para representar as transformações. O segundo, a representação baseada em círculos para representar as transformações é muito semelhante com a utilizada para representar diagramas de estado, o que pode trazer problemas se o diagrama de fluxo de dados for decomposto para um diagrama de estados, levando a necessidade de alterar a notação utilizada na elaboração de diagramas de estado.

A figura 3.2 apresenta os elementos notacionais utilizados para a construção do DFDC, e a seguir descreve-se cada um destes elementos.

Figura 3.2. Elementos notacionais do DFDC.

a) Transformações

As transformações são representadas por retângulos com as bordas arredondadas. Um retângulo cheio representa uma transformação de dados. As transformações de dados modelam as partes do sistema responsáveis pelo processamento dos dados do sistema, tais como dados de entrada. As transformações de controle são representadas por um retângulo pontilhado. As transformações de controle agem sobre tranformações de dados e controle, ativando-as ou desativando-as. Para ambas as transformações a divisão superior no retângulo é utilizada para identificação da transformação, com números, no caso de transformações de dados e letras no caso de transformações de controle. A divisão inferior do retângulo é usada para nomear a transformação. Na divisão central descreve-se sumariamente a função da transformação.

b) Fluxos de dados e eventos

Os fluxos são representados por linhas. Os fluxos podem ser de dois tipos: dados e eventos. Os fluxos de dados podem ser de natureza discreta ou contínua. Os fluxos de dados contínuos são representados por uma linha cheia com duas setas na direção do fluxo. Os fluxos de dados discretos são representados por uma linha cheia com uma seta na direção do fluxo. Linhas pontilhadas representam fluxos de eventos. Os fluxos de eventos representam um comando específico em um ponto discreto no tempo.

Uma linha pontilhada com uma única seta na direção da transformação indica um sinal, por exemplo uma interrupção. Uma linha pontilhada com duas setas entrando na transformação indica uma ativação. Uma linha pontilhada com duas setas saindo da transformação indica uma desativação. Uma ativação representa a intenção de que a transformação ativada produza uma resposta. Uma desativação representa a inibição de uma saída por parte de uma transformação de dados ou controle.

c) Depósitos de dados e buffers

Os depósitos de dados em um sistema físico, são regiões de memória onde são armazenadas variáveis que são compartilhadas pelas transformações. O retângulo a esquerda identifica o depósito de dados, esta identificação é feita por um número precedido da letra D. Os buffers são representados por linhas pontilhadas e são identificados por um número precedido da letra B no retângulo a esquerda. No caso de depósitos de dados ou buffers serem repetidos na representação gráfica do sistema eles serão precedidos de mais um traço a esquerda.

d) Entidades externas

Entidades externas são representadas por um quadrado cheio. Dentro deste quadrado constará a denominação da entidade externa. No caso de entidades externas serem repetidas na representação do sistema, o retângulo receberá um traço inclinado no canto superior esquerdo.

3.2.2. Aplicação do DFDC

A aplicação do DFDC para a análise de um sistema deve seguir uma sequência de passos interativos, os quais podem ser resumidos como segue:

i) Identificar as entidades externas;

ii) indentificar os requisitos de entrada/saída;

iii) gerar a primeira representação do sistema, utilizando os elementos notacionais descritos na secção 3.2.1;

iv) aplicar refinamentos sucessivos à primeira representação conseguida no passo iii), com o objetivo de detalhar mais a representação e a análise, e identificar os requisitos de entrada/saída ainda não considerados.

Quando atinge-se o passo iii) o diagrama obtido será aqui chamado de DFDC de primeiro nível do sistema. Possívelmente esta representação não será ainda suficiente, havendo a necessidade de aplicar-se refinamentos à determinadas transformações. Através deste processo de refinamentos resumido no passo iv) atinge-se maiores níveis de detalhes. Este processo deve prosseguir até que seja atingido um nível de refinamento que possa representar de forma mais fiel possível a estrutura lógica do sistema em questão, estabelecendo desta forma as interações entre as várias transformações assim como as estruturas de dados necessárias para a implementação do software do sistema. Deve-se entretanto deixar muito claro que o objetivo de aplicar-se o DFDC para análise e/ou representação de um sistema é

conseguir uma visão global sobre a estrutura lógica do sistema e não detalhar-se todos os processos, interações e estruturas de dados presentes no software do sistema. Este nível de detalhamento será atingido quando aplicarmos técnicas de decomposição ao DFDC. Para o número de refinamentos a serem aplicados ao DFDC um valor típico é três [35].

Na figura 3.3 mostra-se o DFDC de um sistema simples de controle de temperatura. Para este sistema, como ve-se no DFDC de primeiro nível da figura 3.3.a consideram-se os seguintes parâmetros como entradas para o sistema: temperatura de referência ("Set Point"), a temperatura atual, e os comandos de início e fim (aborto) do processo de controle, e como saída a ação corretiva de controle. Duas entidades externas são identificadas: a interface com o processo e a interface com a fonte de referência de temperatura e os comandos de início e fim. A transformação 1, MUDRTEM, adquire a temperatura de referência proveniente da interface e a armazena no depósito de dados D1, REFTEMP. A transformação 3, LEVTEMP, executa a leitura da temperatura atual do processo atravéz da entidade externa INTERFACE COM O PROCESSO, efetua alterações nesta leitura, tais como filtragens digitais, a comparação com a temperatura de referência D1, e gera dois sinais indicando se a temperatura esta normal ou não, NORMAL e ANORMAL. A transformação 2, MANTEMP, mantém a temperatura do processo através da comparação entre a temperatura de referência , através de D1

e a temperatura atual D2. A transformação de controle A, CONTEMP, é responsável pela sincronização e ativação das ações de controle e aquisição de dados a serem executadas pelas transformaçoes de dados 1,2 e 3. Por exemplo, no caso de após ter ativado a transformação 3, esta enviaria um sinal indicando temperatura anormal, ANORMAL, a transformação 2 seria ativada a fim de efetuar a ação corretiva através de uma ação de controle, CONTROLE, enviada a interface com o processo. A interface com o processo básicamente poderia ser constituída de um conversor A/D que a partir dos sinal contínuo TEMPERATURA, produziria o sinal de temperatura discreto no tempo, TEMP DISC, e um conversor D/A que a partir do sinal discreto de controle CONTROLE, produziria o sinal contínuo CONT CONTI.

Na figura 3.3.b apresenta-se o refinamento da transformação 2. Esta decomposição se faz necessária a fim de melhor visualizar-se as funções da transformação 2, que executa duas tarefas distintas, que são a implementação do algoritmo de controle, transformação 2.1, ALCON e o acionamento da interface com oprocesso, transformação 2.2, ACINT.

Referindo-se ainda a figura 3.2.a note-se que quando uma transformação é refinada deve-se explicitar muito bem que o diagrama resultante é um refinamento. Isto é feito utilizando-se um esquema de identação de tal forma que o primeiro índice refira-se a transformação original. Por

exemplo, na figura 3.3.a, o índice 2 refere-se a transformação no primeiro nível de refinamento, e o índice seguinte refere-se ao nível de refinamento atual, por exemplo 2.1 e 2.2 na figura 3.3.b. Além disto deve-se introduzir o refinamento correspondente em um retângulo com o número da transformação original. Devem ficar fora deste retângulo os depósitos de dados, buffers e entidades externas compartilhadas por outras transformações. Este procedimento deve prosseguir até que cada transformação tenha sido refinada.

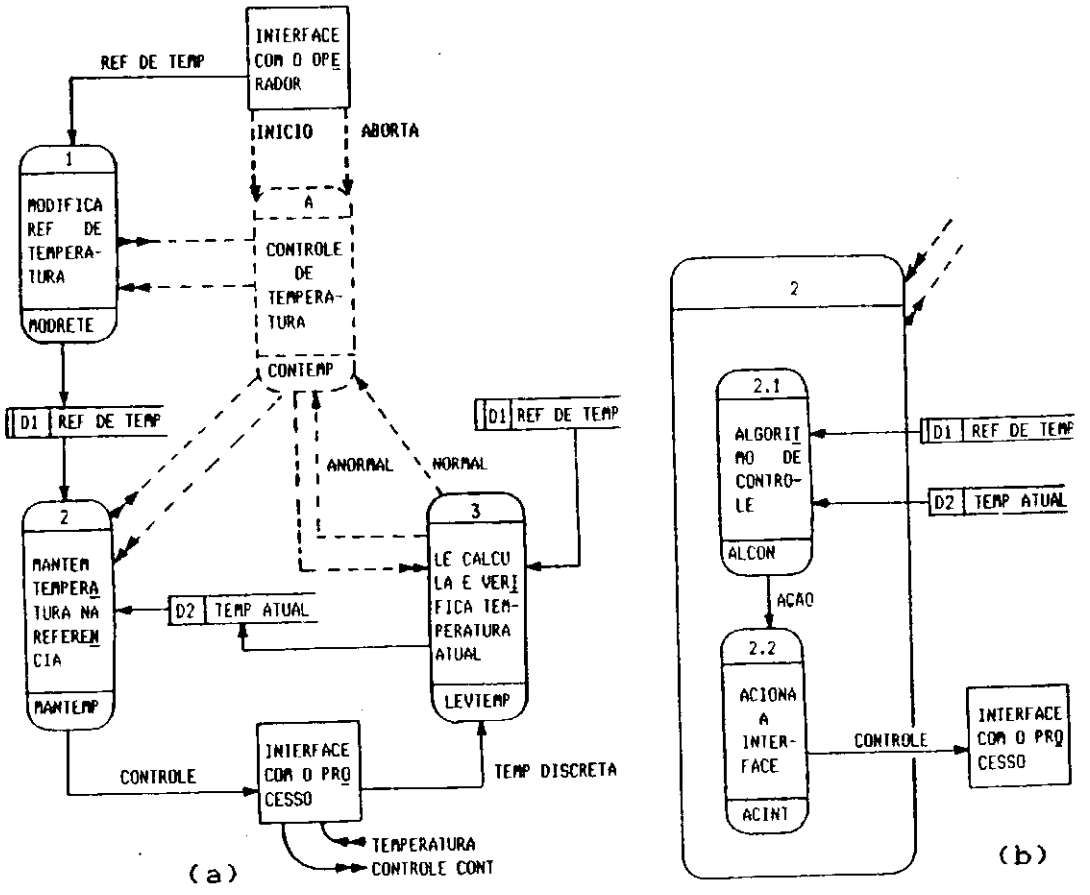


Figura 3.3. (a) DFDC de um sistema para controle de temperatura.

(b) Refinamento da transformação 2.

3.3. Decomposição do DFDC

Após a construção do DFDC do sistema deve-se decompor o DFDC para implementar-se o software do sistema. Para a implementação do software do sistema pode-se optar por linguagens tradicionais como Pascal, FORTRAN, C, etc. Outra opção é a utilização de linguagens modulares, objeto orientadas, como ADA, MODULA 2, C++, Smalltalk, etc. A escolha de uma ou outra classe de linguagem resultará na necessidade de uma decomposição processo orientada no caso do uso de linguagens tradicionais ou decomposição dados orientada no caso de linguagens modulares objeto orientadas.

3.3.1. Utilização de decomposição processo orientada

No caso de decomposição processo orientada a ênfase será dada à natureza do algoritmo a ser implementado através de procedimentos ou subprogramas, definindo desta forma as ações mais importantes que ocorrem no sistema. Diversas técnicas podem ser empregadas com esta finalidade, para obter uma estrutura mais detalhada a partir do DFDC, tais como: decomposição funcional (top-down ou bottom-up) [31,35], metodologia de Jackson [34,35], etc. Neste trabalho abordar-se somente a decomposição funcional, pois é a técnica de decomposição que será empregada em todo o trabalho.

A técnica de decomposição funcional baseia-se essencialmente em extrair-se a partir do DFDC módulos que implementem as funções requeridas pelas transformações bem como definir rígidamente as interfaces entre os diferentes

módulos. Note-se que a definição rígida das interfaces entre os diferentes módulos é de extrema importância na integração entre os módulos, resultando em uma diminuição significativa no tempo necessário para a integração entre os diferentes módulos.

Na figura 3.4 mostra-se a decomposição funcional para o DFDC do sistema de controle de temperatura mostrado na figura 3.3. Note-se que cada módulo derivado a partir do DFDC representa uma transformação e as interfaces entre os módulos são definidas pelos depósitos de dados D1, e D2 do DFDC, que definem as estruturas de dados necessárias a estas interfaces.

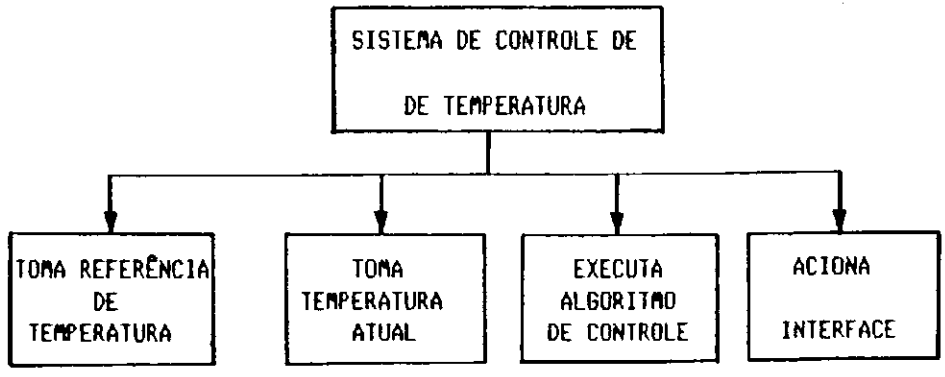


Figura 3.4. Decomposição funcional do DFDC da figura 3.3.

3.3.2. Decomposição dados orientada.

Nesta técnica de decomposição a ênfase recai sobre os dados e não sobre as ações (procedimentos). Duas importantes técnicas de decomposição dados orientada podem ser citadas: decomposição objeto orientada [38] e a metodologia conceitual de bancos de dados [31]. Enfase será

dada a decomposição objeto orientada pois dentro do escopo deste trabalho, automação de processos, é a técnica mais indicada, devido as linguagen modulares, objeto orientadas, possuirem embutidas sofisticadas estruturas e técnicas de comunicação inter-processos e inter-processadores [41], como exemplo o mecanismo de "rendevouz" em ADA [41]. Além de suportarem programação multi-tarefa e programação em tempo real.

Antes de abordar-se a metodologia de decomposição objeto orientada em si, será introduzido um conceito para objeto que não é único porém satizfaz às necessidades desta apresentação, maiores detalhes podem ser encontrados em [41]. Um objeto é uma entidade que possui um nome e um estado, sendo caracterizada pelas ações que executa e requer de outros objetos.

A decomposição objeto orientada a partir do DFD do sistema foi introduzida por Booch [38], e neste trabalho é estendida para o DFDC. Esta técnica pode ser resumida em uma série de passos, primariamente sugeridos por Abbott [42], que são descritos a seguir.

i) Identificar a partir do DFDC as operações executadas e requeridas pelos objetos;
ii) estabelecer a visibilidade de cada objeto em relação aos demais;
iii) estabelecer as interfaces entre os objetos;
iv) implementar os objetos.

Na figura 3.5 apresenta-se a decomposição objeto orientada para o DFDC do sistema de controle de temperatura mostrado na figura 3.3. Uma descrição detalhada desta técnica de decomposição pode ser encontrada em [38].

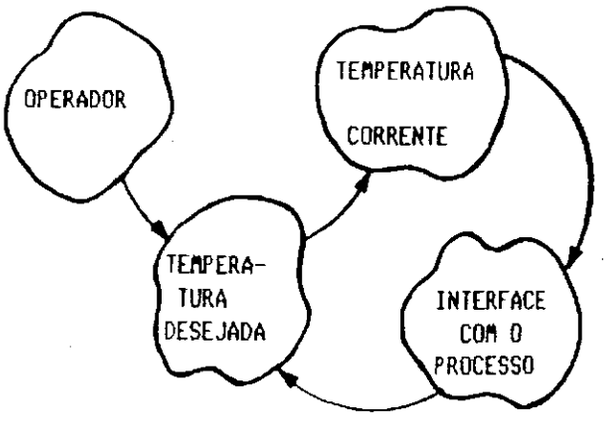


Figura 3.5. Decomposição objeto orientada do DFDC da figura 3.3.

3.4. Aplicação do DFDC ao SDVC

O DFDC foi aplicado para a análise e representação do SDVC, proposto no capítulo 2. Como mostrado no SDVC implementa-se uma estrutura de controle multinível, sendo dividido em três níveis hierárquicos, cada um desempenhando funções bem definidas. Os níveis são interligados por linhas de comunicação assíncronas padrão RS-232C.

No nível III do sistema, figura 3.6, executam-se as funções de aquisição de dados e controle direto sobre o processo. No nível II do sistema implementam-se e executam-

se os algoritmos de controle e a interação com o nível I. E o nível I é responsável pela execução de algoritmos de otimizacação, supervisão e interação com o usuário.

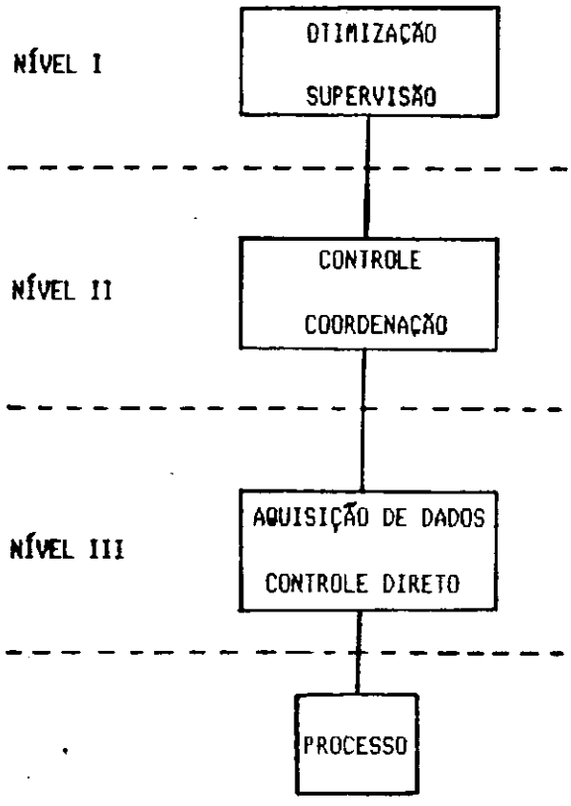


Figura 3.6. Diagrama de Blocos do SDVC.

3.4.1. DFDC de primeiro nível para o SDVC e refinamentos sucessivos

Seguindo os passos descritos na secção 3.1.2, chegou-se ao DFDC de primeiro nível para o SDVC. Na figura 3.7 mostra-se este DFDC.

Após a construção do DFDC de primeiro nível para SDVC, aplicam-se refinamentos às transformações e obtem-se a versão refinida. Para o refinamento optou-se por refinar-se independentemente cada nível.

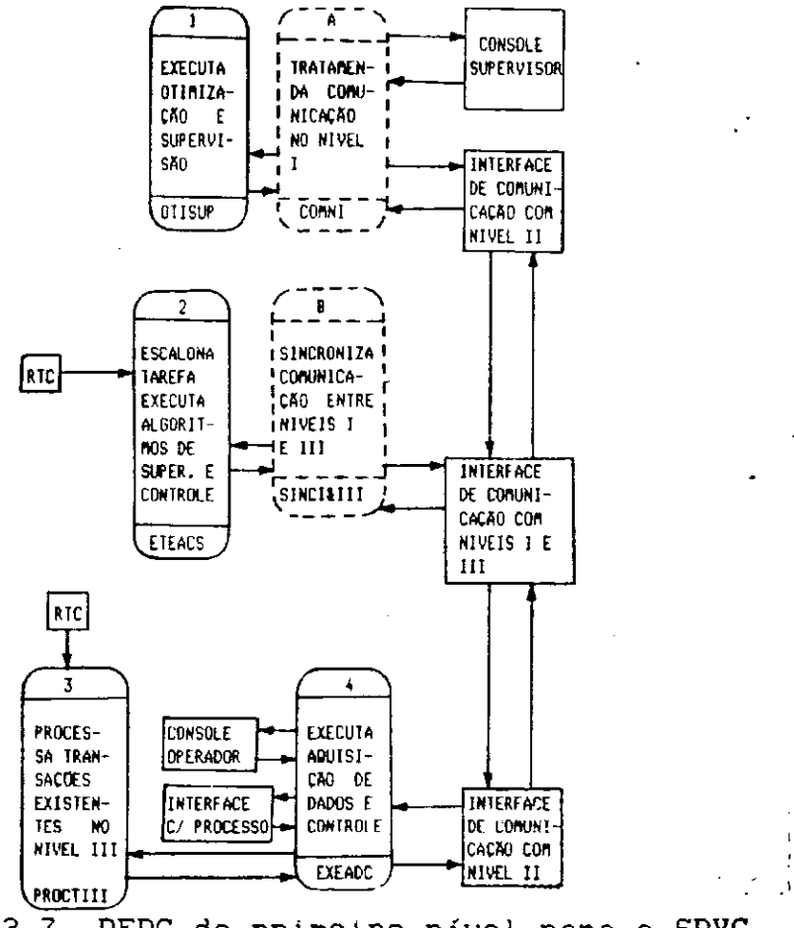


Figura 3.7. DFDC de primeiro nível para o SDVC.

O refinamento do nível III do DFDC de primeiro nível do SDVC resultou em dois diagramas, um correspondente à transformação 3, e outro à transformação 4 na figura 3.7.

Na figura 3.8 ve-se o refinamento da transformação 3 e na figura 3.9 o refinamento para a transformação 4.

Nas figuras 3.10 e 3.11 tem-se os refinamentos

correspondentes as transformações 2 e B da figura 3.7, respectivamente.

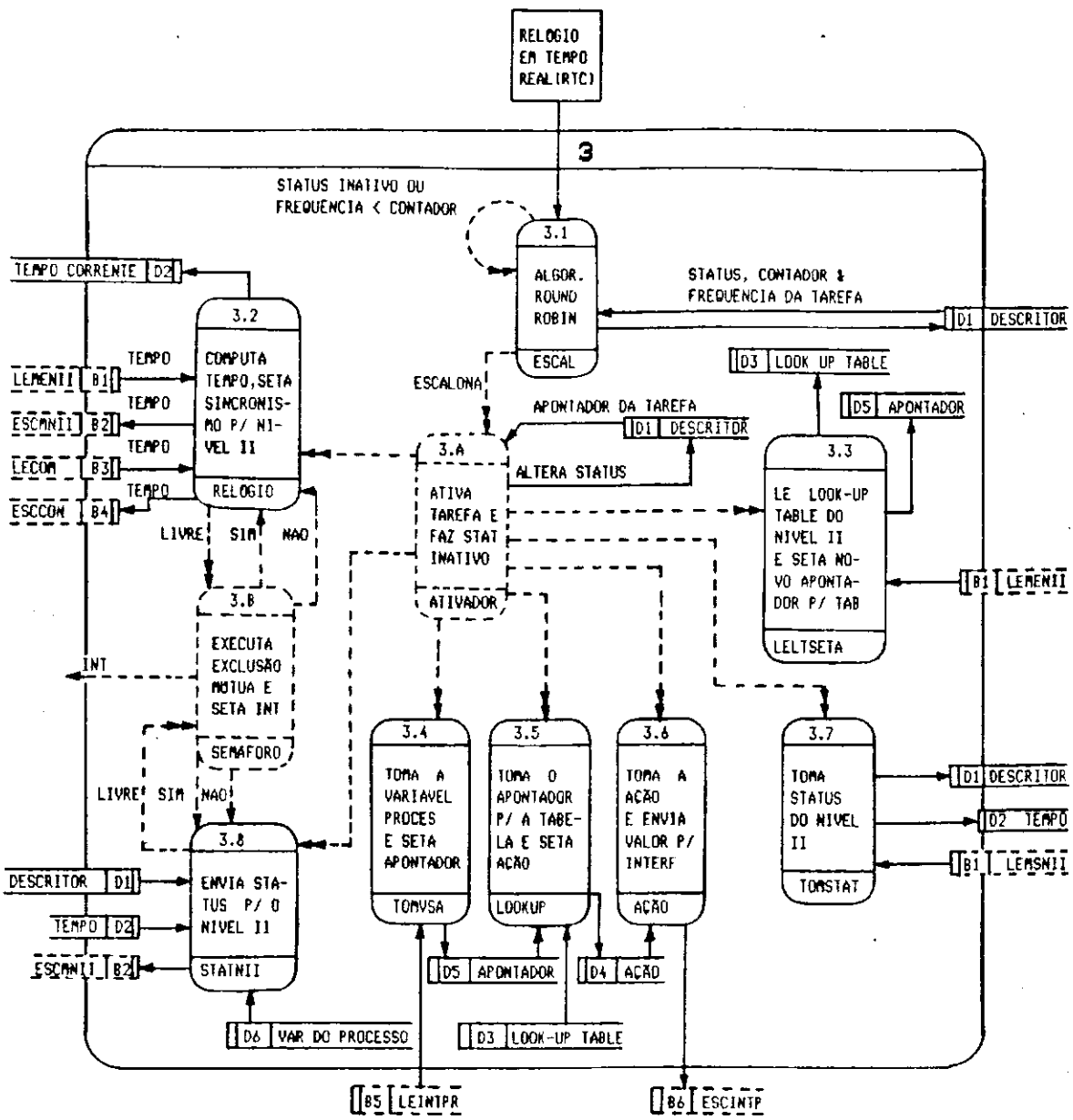


Figura 3.8. Refinamento da transformação 3 da figura 3.7.

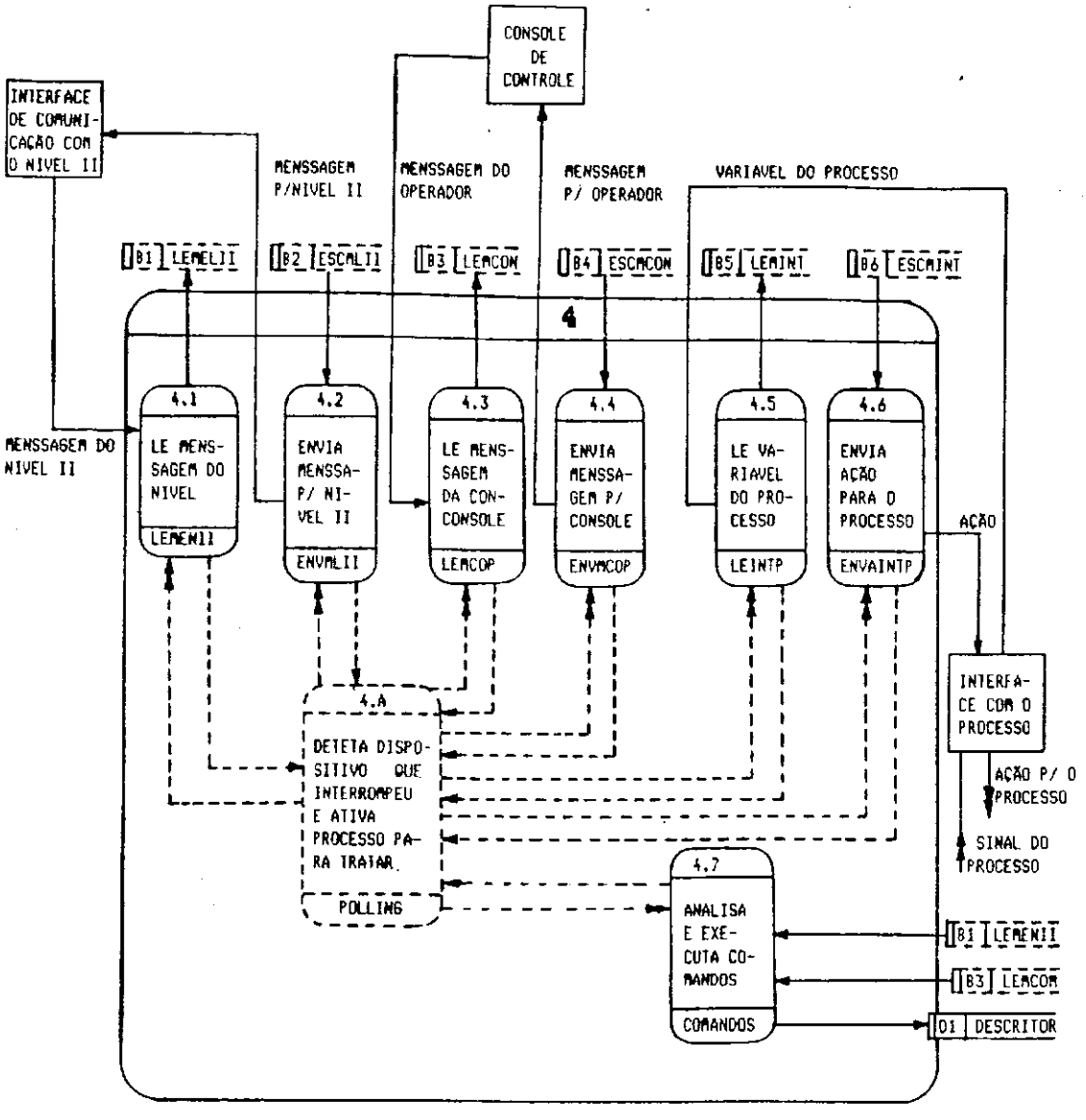


Figura 3.9. Refinamento da transformação 4 da figura 3.7.

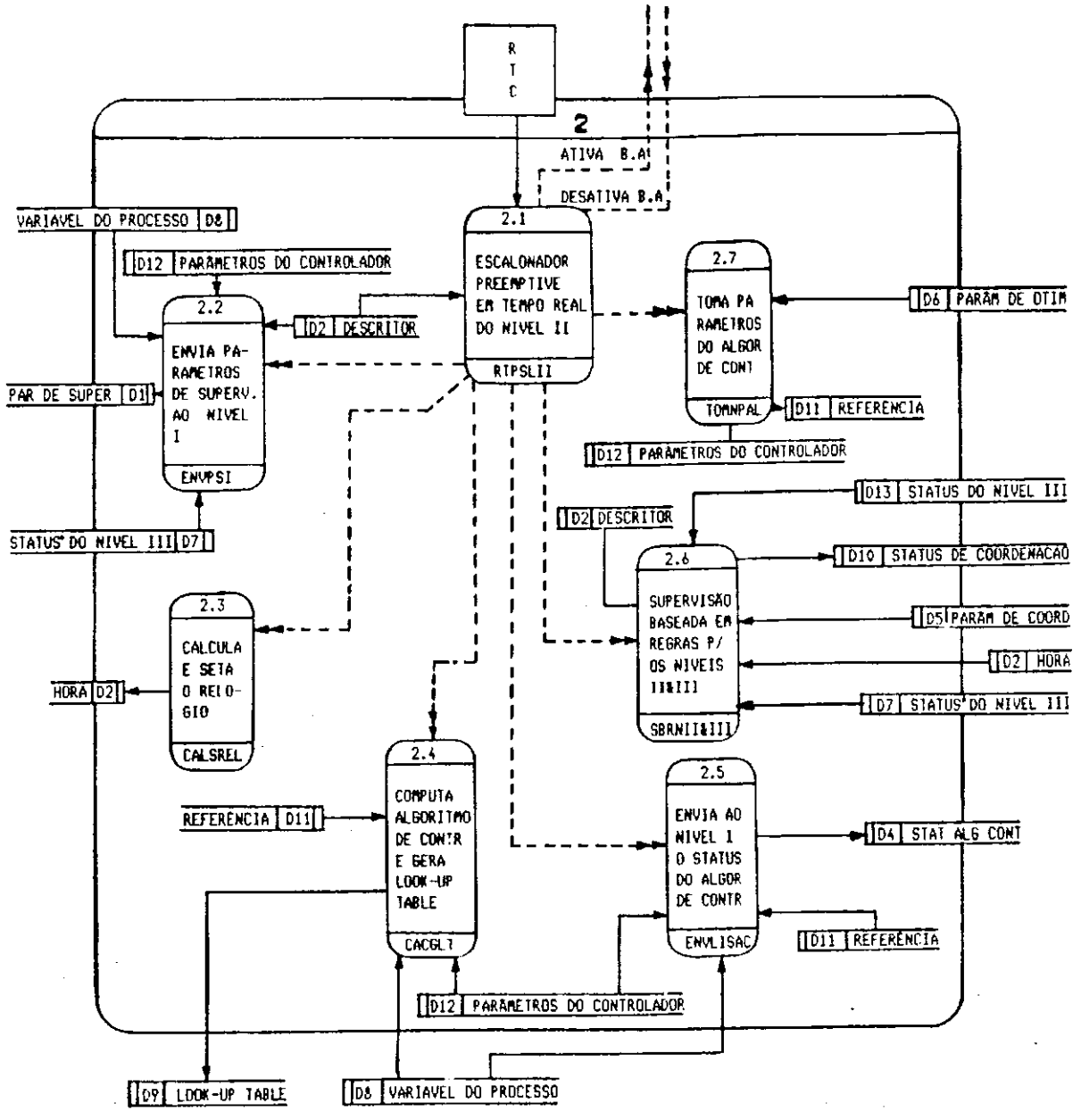


Figura 3.10. Refinamento da transformação 2 da figura 3.7.

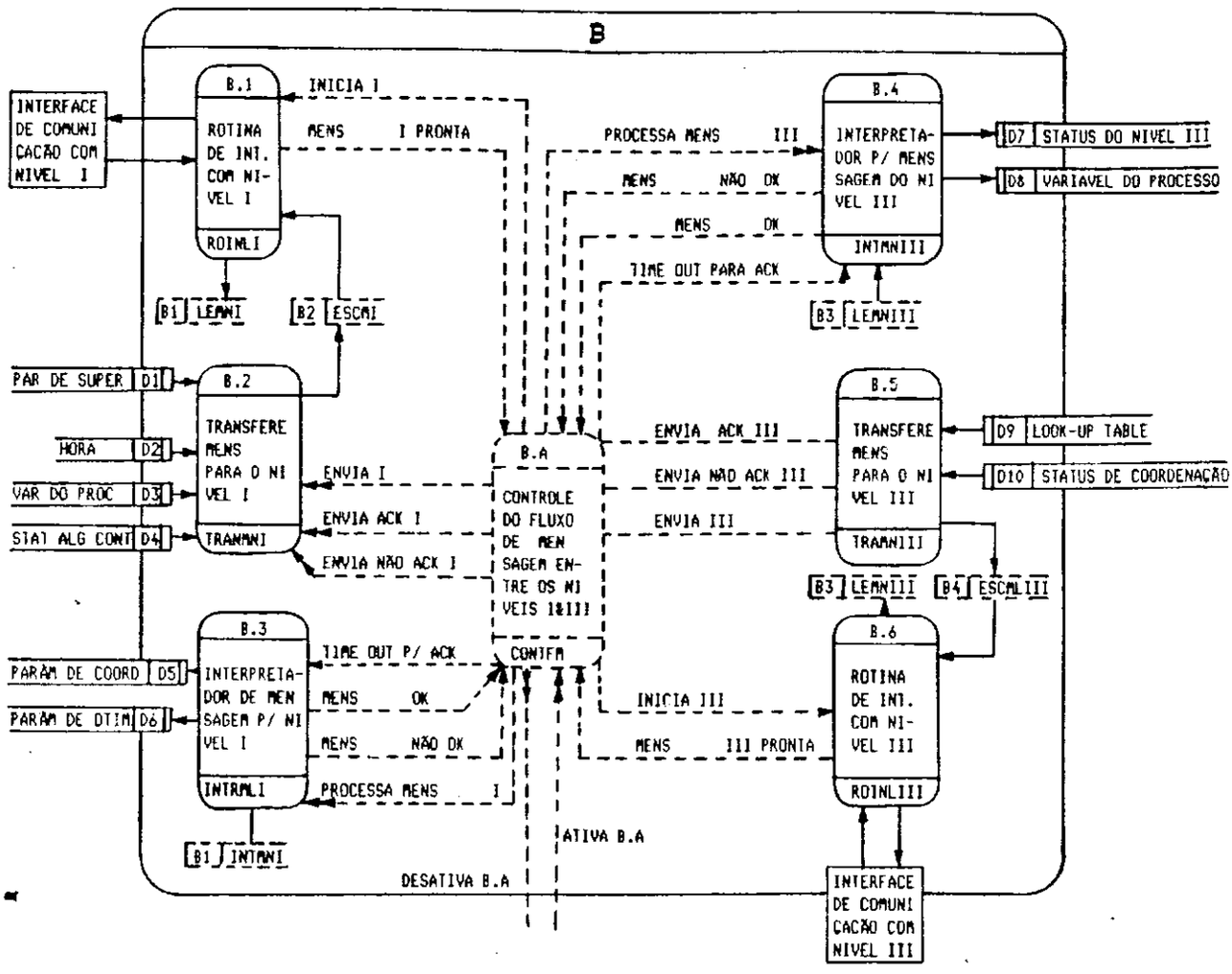


Figura 3.11. Refinamento da transformação B da figura 3.7.

3.4.2. Decomposição do DFDC do SDVC

Para a decomposição do DFDC do SDVC aplicou-se a técnica de decomposição funcional. Optou-se por esta técnica principalmente pela disponibilidade que havia com relação a linguagens processo orientadas e a total indisponibilidade de linguagens objeto orientadas além de grande parte da programação ter sido implementada em linguagem Assembly.

Na figura 3.12 tem-se o diagrama funcional correspondente ao nível III do SDVC.

Na figura 3.13 tem-se o diagrama funcional correspondente ao nível II do SDVC.

A decomposição para o nível I é apenas ilustrativa, uma vez que não foi implementada.

Resumindo, neste capítulo apresentou-se uma ferramenta que possibilita a otimização do processo de análise e projeto de sistemas para Sistema de Automação e Controle. Apresentou-se ainda técnicas de decomposição do DFDC utilizando tanto decomposição dados orientada como processo orientada. Deve-se ainda ressaltar a utilidade do DFDC como uma ferramenta bastante poderosa para a representação da estrutura lógica de Sistemas de Automação e Controle. No próximo capítulo descreve-se a contribuição dada a implementação do SDVC.

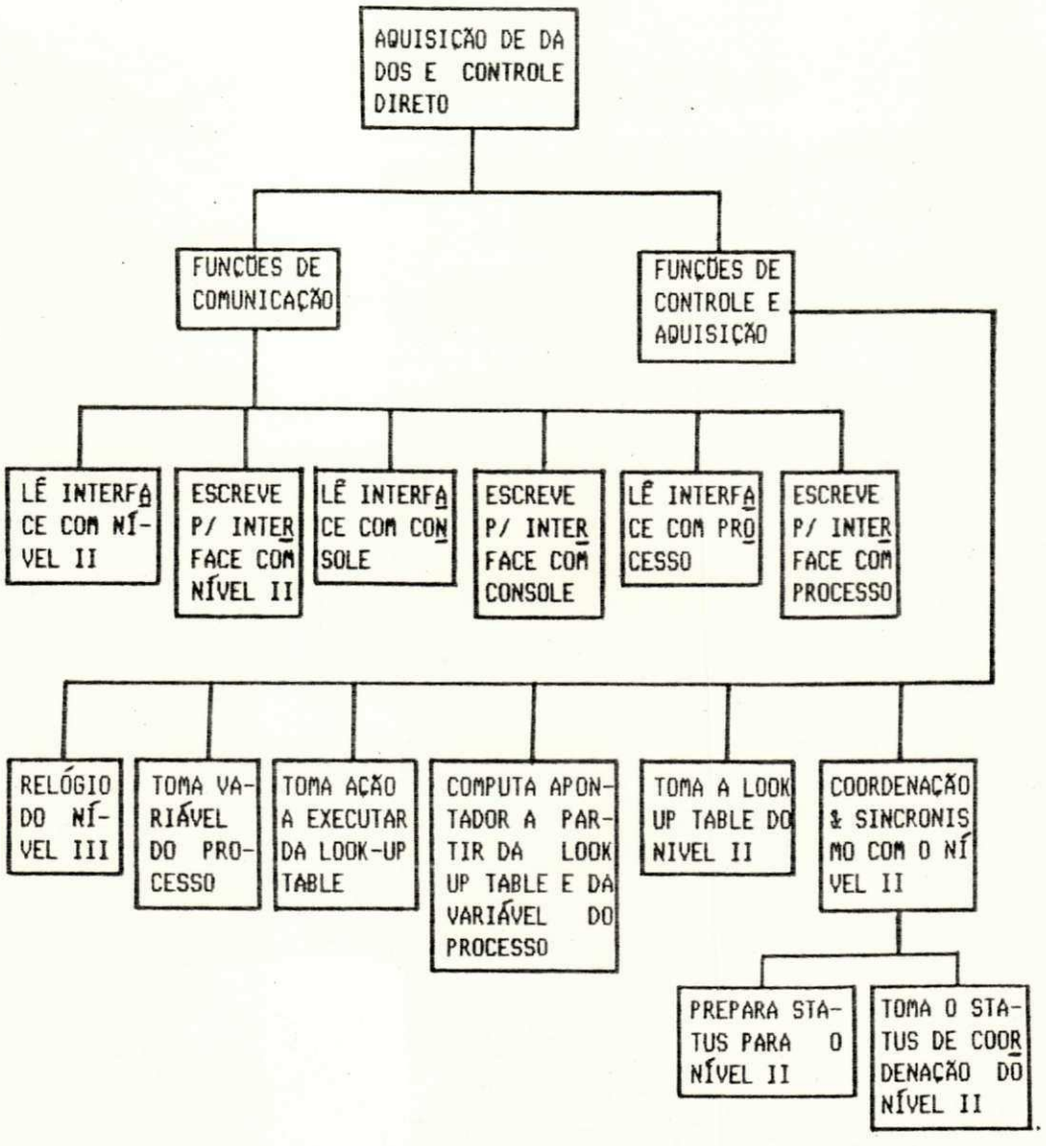


Figura 3.12. Diagrama funcional do Nível III do SDVC.

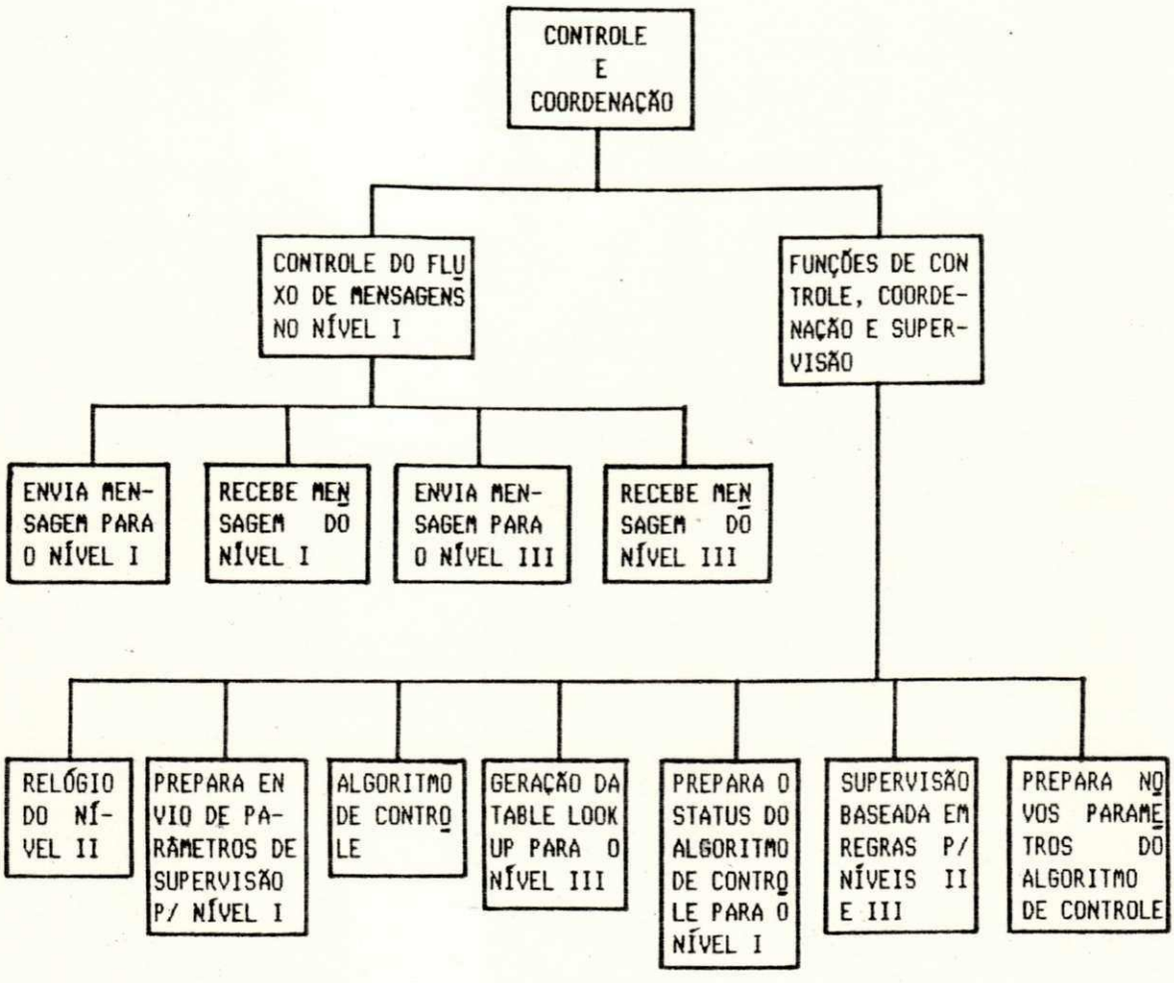


Figura 3.13. Diagrama funcional do Nível II do SDVC.

CAPITULO 4

CONTRIBUIÇÃO A IMPLEMENTAÇÃO DO SDVC

4.1. Introdução

Considerando o que foi exposto nos capítulos 2 e 3, neste capítulo descreve-se a implementação do hardware e do software dos níveis II e III do SDVC. A implementação do software baseia-se nos diagramas funcionais obtidos a partir da decomposição funcional do DFDC dos níveis II e III do SDVC, figuras 3.12 e 3.13.

4.2. Hardware do nível III

O hardware no nível III do sistema constitui-se do microcomputador MATER (Microcomputador para Aplicações em Tempo Real) [43] e de uma placa de interface com um micro motor DC.

4.2.1. Microcomputador do nível III

O microcomputador utilizado no nível III é o MATER. O MATER é constituído de um microprocessador M6800 [44],

relógio de 1 MHz, 8 Kbytes de EPROM, 8 Kbytes de RAM, 2 portas de entrada/saída serial (ACIA) [44], interface RS 232 C, 20 portas de saída paralela (PIA) [44], e relógio em tempo real (RTC) com base de tempo de 3 ms. O RTC é conectado ao pino NMI do microprocessador M6800.

4.2.2. Interface com um micro motor DC

A interface com o micro motor DC é dividida em duas partes, a primeira correspondendo ao circuito de acionamento e a segunda ao circuito para detecção e aquisição da velocidade. Na figura 4.1 ve-se um diagrama em blocos desta interface.

4.2.2.1. Descrição do circuito de acionamento

A regulagem da velocidade do micro motor DC baseia-se em comandos por impulsos [45]. O princípio de comandos por impulsos consiste em regular a velocidade do rotor proporcionalmente ao tempo que a tensão nominal é aplicada ao motor. O comando do micro motor DC por impulsos baseia-se então na aplicação de impulsos de tensão de comando com amplitude constante à tensão nominal Unom. Deste modo seu funcionamento baseia-se na alternância de períodos de aceleração e frenagem. Se estes períodos são curtos em relação a duração total do tempo de aceleração e frenagem do rotor , a velocidade do rotor W não aumentará, desta forma estabilizando-se em um valor médio Wmédio. O valor de Wmédio com relação a uma carga constante e tensão de

excitação constante, é determinado pela duração relativa da tensão aplicada, ou seja é proporcional ao tempo de duração do impulso ti e o período dos impulsos Ti

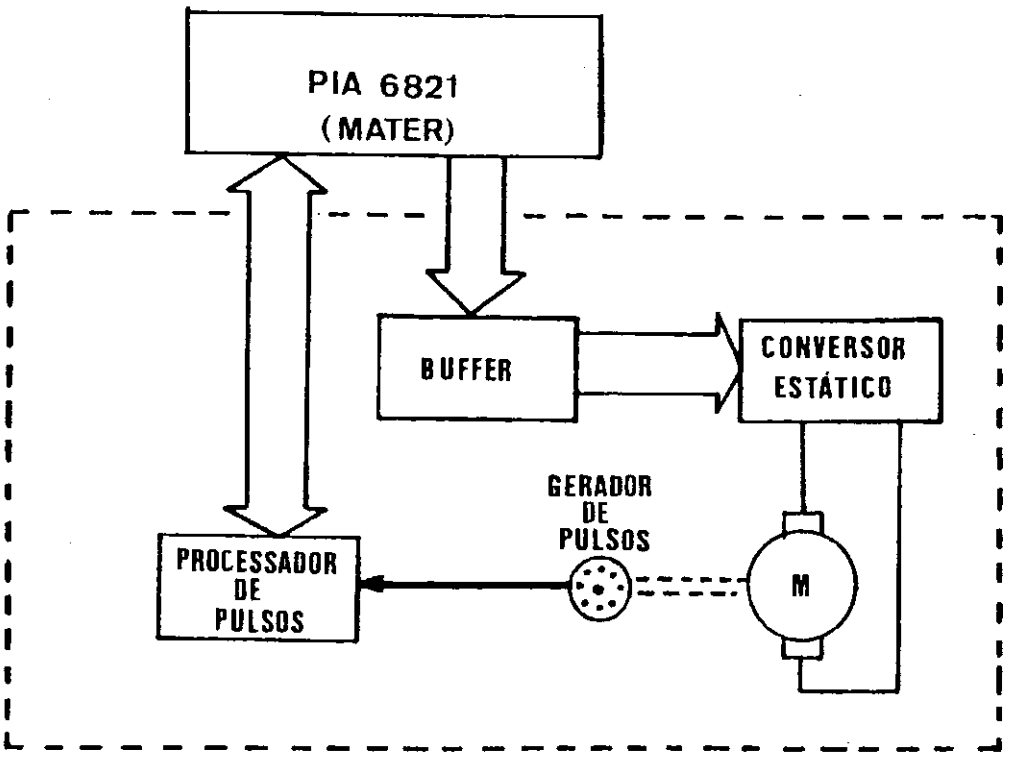


Figura 4.1. Diagrama em blocos da interface com o micro motor DC.

O comando por impulsos faz com que o valor instantâneo da velocidade do rotor oscile de forma contínua em torno de um valor determinado. A amplitude das oscilações deve ser muito pequena em relação a constante eletromecânica do motor e o período de repetição dos impulsos deve ser muito grande. Com o aumento da freqência dos impulsos de comando e o aumento da constante da constante de tempo eletromecânica, a amplitude das oscilações de velocidade

diminuem. Dentro de determinadas condições o valor da velocidade média Wmédio do motor matem-se constante.

Em [45] são sugeridos diversos esquemas para montagem do acionador do motor. Neste trabalho optou-se pela implementação de um conversor estático utilizando uma ponte de transistores funcionando como fonte de corrente. Na figura 4.2 ve-se um diagrama do circuito do conversor estático utilizado para o acionamento do micro motor DC. O circuito possibilita o controle da rotação do motor em ambas as direções, além de possibilitar desativa-lo. O sentido de rotação do motor é horário quando a corrente é de A para B e anti-horário quando a corrente é de B para A.

Para a corrente no motor ser de A para B, rotação horária, o transistor T1 deve estar na região ativa e T2 saturado, enquanto que T3 e T4 devem estar cortados. Para rotação no sentido anti-horário, corrente no motor de B para A, T3 deve estar na região ativa e T4 saturado, T1 e T2 dever estar cortados.

O objetivo consiste em chavear estes transistores através de níveis lógicos , 0 e 1, através de uma interface com o microcomputador, no caso utilizou-se uma PIA M6820 do microcomputador MATER. Porém os pinos da PIA não são capazes de absorver e fornecer as correntes de base necessárias ao corte e saturação dos transistores T1 e T3, assim como de T2 e T4. A solução é a utilização de buffers entre a base dos transistores e a PIA. A solução adotada é

mostrada na figura 4.3.

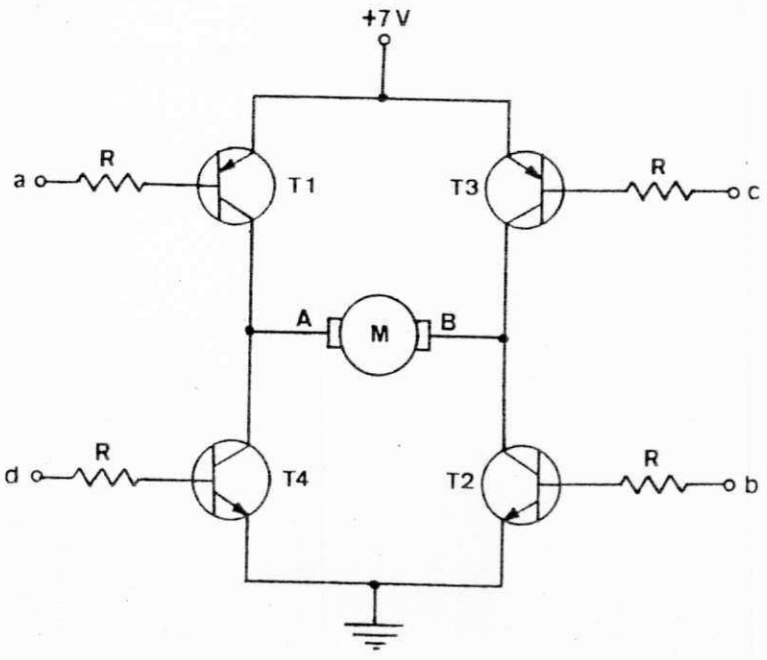


Figura 4.2. Diagrama do conversor estático.

Os amplificadores conectados aos pontos a, b, c e d na figura 4.3 são projetados de forma a absorver e prover as correntes necessárias ao corte e a saturação dos transistores. Um pequeno ganho é introduzido aos amplificadores conectados aos pontos a e c de modo a garantir a polarização reversa das junções base-emissor de T1 e T3 quando um nível lógico '1', ou em torno de 5 volts, é aplicado a entrada destes amplificadores, obtendo assim uma saída em torno de 10V, a qual garante a polarização reversa das junções base-emissor de T1 e T3.

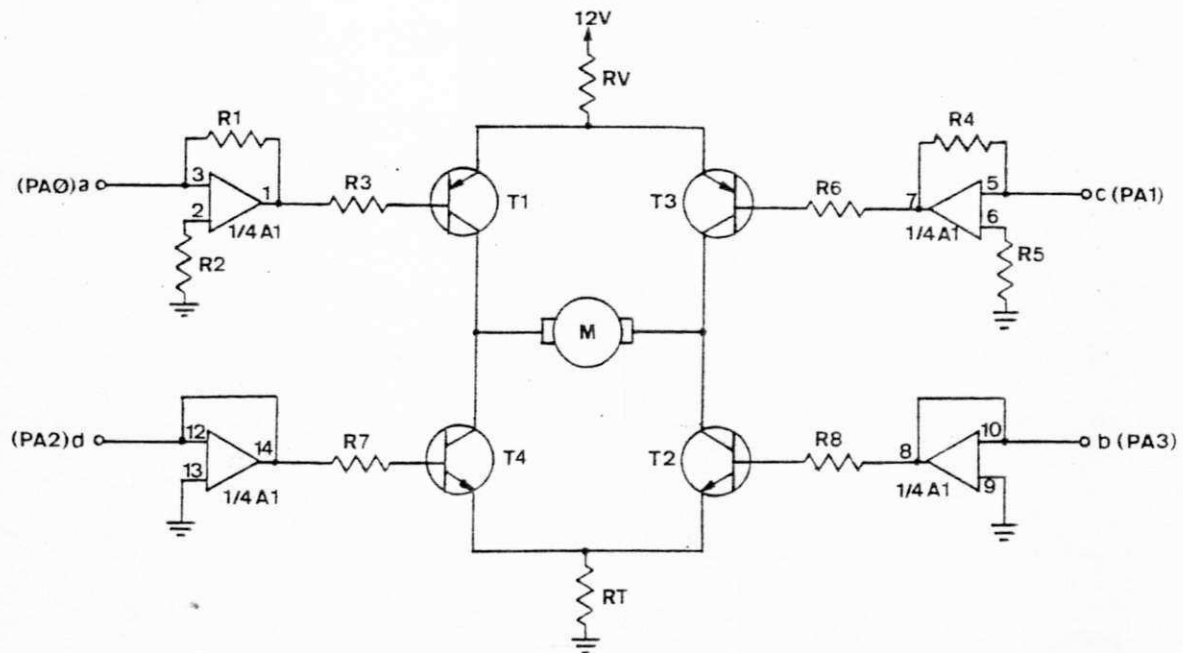


Figura 4.3. Diagrama final do conversor estático com os buffers.

Os níveis lógicos necessários ao controle da direção de rotação do motor são mostrados na tabela 4.1.

Deve-se ressaltar que o circuito amplificador conectado aos pontos a e c deve ser capaz de absorver uma certa corrente quando um nível lógico '0' é aplicado a estes pontos, esta corrente é dada por (I_M/β). O circuito amplificador deve ainda ser capaz de prover as correntes de fuga ("leakage current") dos transistore T1 e T3 quando um nível lógico '1' é aplicado a estes pontos.

No caso dos transistores T2 e T4 os circuitos

conectados aos pontos b e d, devem prover a corrente de base necessária a saturação dos transistores T2 e T4 quando um nível lógico '1' é aplicado a estes pontos. O circuito deve ser ainda capaz de absorver a corrente de fuga ("leakage current") quando um nível lógico '0' é aplicado a estes pontos.

Direção de rotação	T1	T2	T3	T4	níveis lógicos			
					a	b	c	d
Horário	C	S	CO	CO	0	1	1	0
Anti Horário	CO	CO	C	S	1	0	0	1
Parado	CO	CO	CO	CO	0	0	0	0

Onde: C - Conduzindo

S - Saturado

CO - Cortado

Tabela 4.1. Níveis lógicos para o controle da direção de rotação do motor.

Utilizando-se transistores com um ganho alto, T2 e T4, e valores apropriados de resistores de base, é possível garantir o funcionamento correto do circuito.

E interressante notar que o motor é alimentado com uma fonte de corrente e o torque do motor é diretamente

proporcional a corrente, uma vez que a densidade do fluxo de campo magnético é constante:

$$Te = TL + BN + J(dN/dt)$$

$$kFila = TL + BN + J(dN/dt)$$

$$(kFila - TL) = BN + J(dN/dt)$$

$$N = (kFi - TL/B) (1 - e \exp ((-B/J)t))$$

4.2.2.2. Circuito para detecção e aquisição da velocidade

Para detecção da velocidade do motor optou-se pelo uso de um disco perfurado juntamente com uma foto-emissor/detetor. Apesar das limitações, este tipo de estratégia é suficiente e atinge os objetivos, que são apenas para a validação do software desenvolvido.

Na figura 4.4 ve-se o diagrama do circuito de interface com o conjunto motor/sensor.

O circuito de aquisição de velocidade é constitído do sensor, que está montado no conjunto motor/sensor, mais um circuito de acondicionamento do sinal. O circuito de acondicionamento do sinal é constituído de um contador módulo 16. O contador módulo 16 é utilizado para evitar que sejam geradas muitas interrupções para o microcomputador MATER, quando são gerados os pulsos pelo detector. Deste modo é feita uma contagem de 16 pulsos mais o resíduo contido no contador módulo 16. Após a leitura do resíduo, o contador é resetado, retomando-se assim um novo ciclo de

contagem.

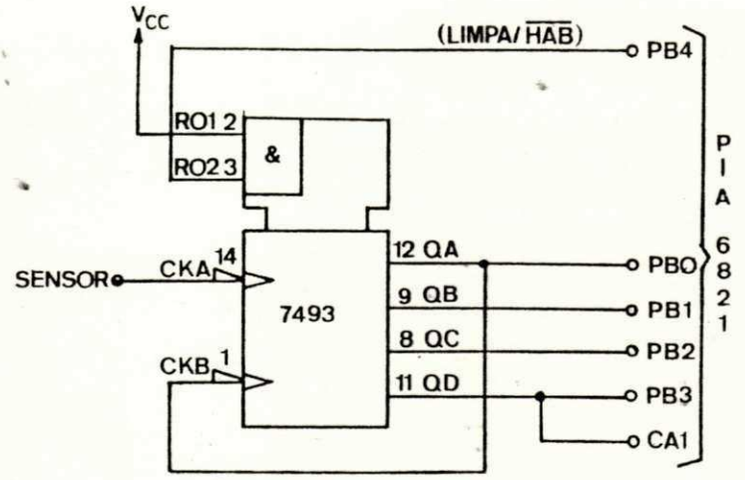


Figura 4.4. Diagrama elétrico da interface para aquisição da velocidade do motor.

Há que se ressaltar a simplicidade desta estratégia para a deteção de velocidade assim como as suas limitações. A precisão do conjunto detetor é bastante pobre em baixas velocidades pois o contador módulo dezesseis mascarará pulsos de baixa frequência. Em velocidade altas haverá perda de precisão, pois o foto emissor/detetor utilizado (CNY 47) possui um tempo de resposta muito elevado para esta aplicação. Um esquema ideal para deteção de velocidade pode ser encontrado em [46].

4.3. Hardware do nível II

O hardware do nível II constitui-se de um microcomputador CAMAÇARI. Este microcomputador baseia-se no microprocessador Z80 [47], implementado em cartões padrão "EUROCARD", com barramento "ECB" ("Euro Card Bus"). A versão utilizada do CAMAÇARI constitui-se de três cartoẽs. Um cartão de UCP (Unidade Central de Processamento), contendo: um microprocessador Z80A (relógio de 4 MHz), 16 Kbytes de memória EPROM, duas portas de saída seriais (Z80A DART) [47], interface padrão RS 232C, 16 linhas de saída paralelas (Z80A PIO) [47], interface paralela padrão CENTRONICS e quatro contadores/temporizadores programáveis (Z80A CTC) [47]. Um cartão de memória com 64 Kbytes de RAM. E, um cartão controlador de disco contendo: um controlador de disco (I8272) [48], um controlador de Acesso Direto a Memória (Z80A DMA) [47], e interfaces para unidades de disco de 5 1/4 e 8 polegadas.

4.4. Software do nível III

O software do nível III divide-se em duas partes: o programa de inicialização do MATER e os programa do nível III do SDVC. A partir dos DFDC do nível III do SDVC no capítulo 3 e da aplicação específica, o micro motor DC, obtem-se os DFDC das figuras 4.5 e 4.6, e a respectiva decomposição funcional na figura 4.7.

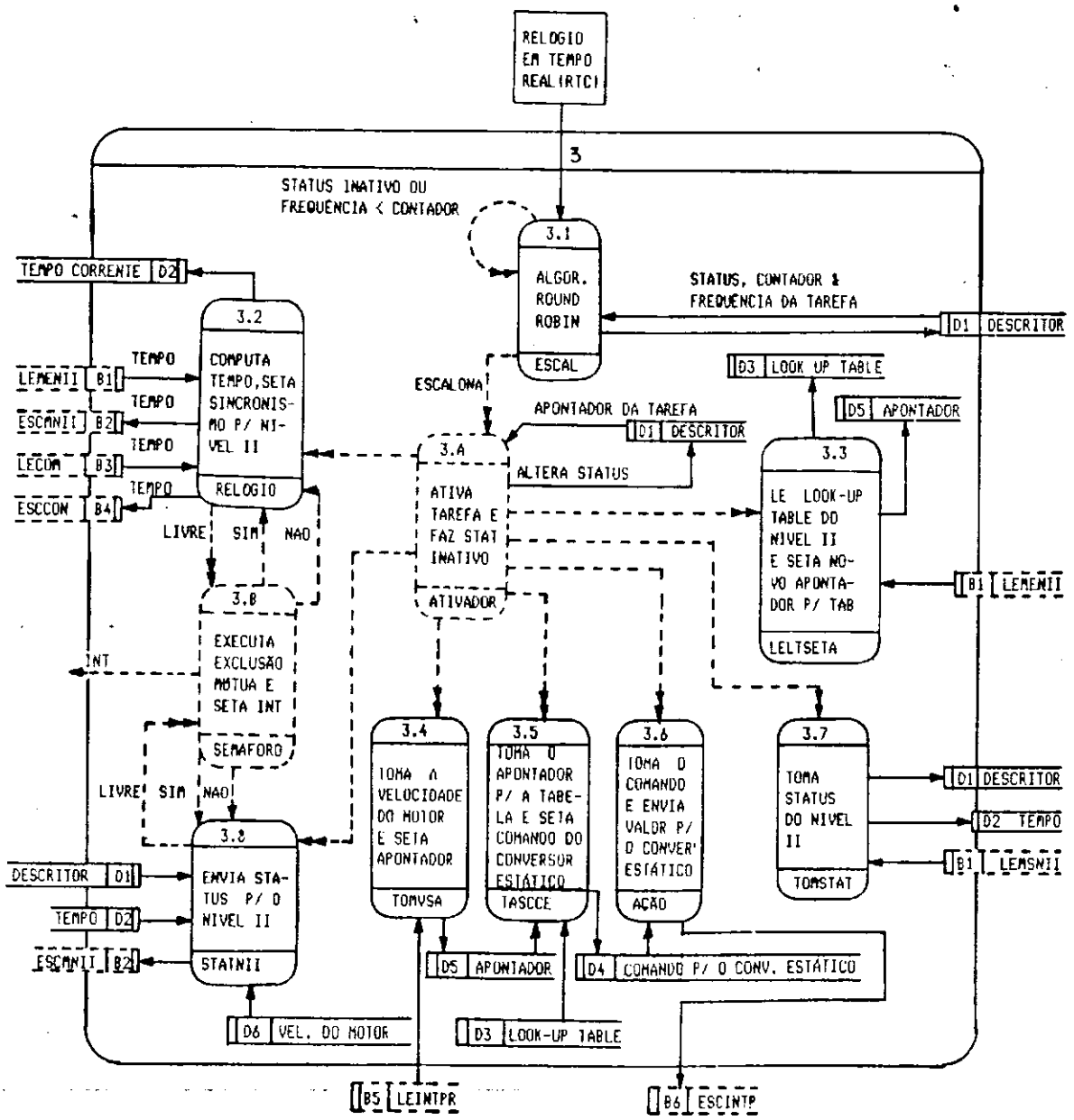


Figura 4.5. DFDC decomposto da transformação 3, figura 3.7, para a aplicação do SDVC ao micro motor DC.

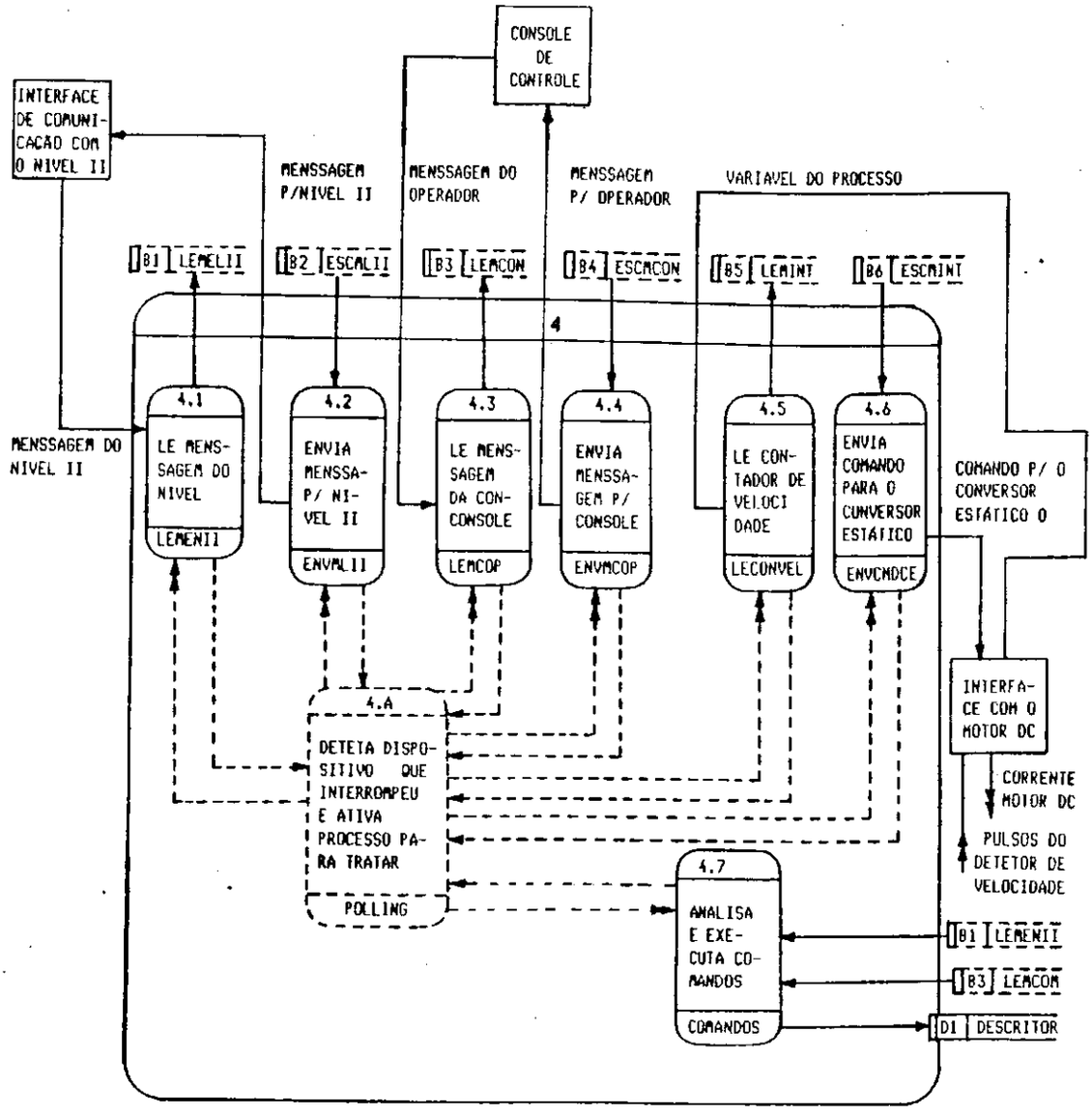


Figura 4.6. DFDC decomposto da transformação 4, figura 3.7, para a aplicação do SDVC ao micro motor DC.

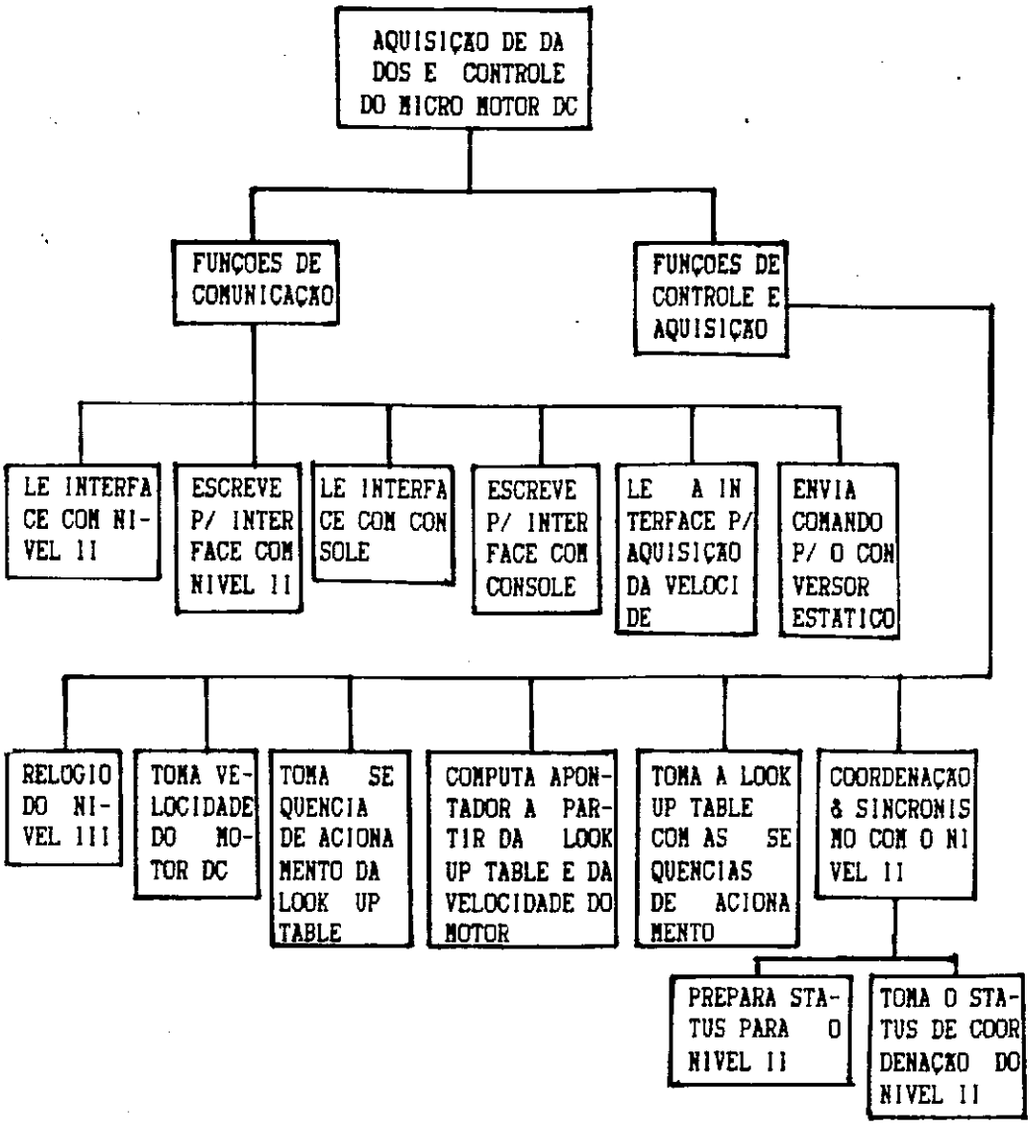


Figura 4.7. Decomposição funcional do DFDC das figuras 4.5 e 4.6.

4.4.1. Programa de inicialização do MATER

O programa de inicialização do MATER executa as funções de inicialização das interfaces de comunicação com o terminal do operador e com o nível II do SDVC. Após a execução da inicialização destas interfaces envia-se um comando de carregamento do programa do nível III para o nível II. O programa aplicativo do nível III é enviado através da interface RS-232 C pelo nível II. A codificação deste programa segue o formato MIKBUG da Motorola [49]. Terminada a transferência o programa de inicialização executa o programa aplicativo do nível III.

4.4.2. Programa do nível III do SDVC

O programa do nível III do SDVC foi desenvolvido baseado em um executivo em tempo real escrito em linguagem Assembly para o microprocessador M6800, o SOATER (Software para Aplicações em Tempo Real) [50]. O SOATER é um executivo em tempo real baseado em um algoritmo de escalonamento estático, onde a prioridade é definida pela posição da tarefa em uma estrutura de dados definida por uma lista encadeada circular, de tal forma que a última tarefa aponta para primeira. No SOATER uma tarefa é definida por um campo identificador, um conjunto de códigos e se necessário um conjunto de dados. O identificador da tarefa é constituído por um descritor constituído dos seguintes campos:

- FREQ: indica a frequência de ocorrência da tarefa, somente para tarefas síncronas, tarefas assíncronas possuem este campo igual a zero, o valor deste campo é um múltiplo inteiro do RTC.
- CONT: contador da tarefa, é definido do mesmo modo que o campo de freqência em múltiplos inteiros do RTC, a cada ocorrência do RTC este campo é decrementado até ser igual a zero. Assim que o contador atinge o valor zero o status da tarefa é modificado para pronto para executar, setando o campo de status, STAT, e o contador é reinicializado novamente com valor de frequência contido no campo FREQ.
- STAT: este campo define se uma tarefa esta pronta para executar, quando seu valor é 1, ou bloqueada, quando seu valor é 0.
- LINK: apontador para a próxima tarefa na lista encadeada circular.

Uma tarefa no SOATER pode assumir três possíveis estados: executanto, pronta para executar, e bloqueada. O acionamento das tarefas é feito segunda uma prioridade estática que é definida pela posição da tarefa na lista encadeada circular. A cada interrupção do RTC, executa-se o programa NUCLEO, constituído do escalonador e do acionador. O escalonador atualiza os campo do contador, CONT, da tarefa, caso seja zero reinicializa-o para o valor definido no campo de frequência, FREQ, e seta o campo de status,

STAT, de forma a deixar a tarefa pronta para executar. Este processo é repetido para todas as tarefas síncronas. O acionador a partir da prioridade definida pela posição da tarefa na lista encadeada circular dos descritores, acionará as tarefas cujos campos de status, STAT, forem iguais a 1, isto é prontas para executar, e reinicializa este campo para 0 de modo a bloquear a tarefa.

As operações de entrada e saída são tratadas por programas dedicados para leitura/escrita da/para a interface com o processo, no caso um micro motor DC, leitura/escrita da/para a interface de comunicação com o nível II, e opcionalmente leitura/escrita da/para uma console de operador. As rotinas para tratamento dos dados de/para estas interfaces são acionadas por interrupção atravéz de um programa de "polling", que é executado assíncronamente, com relação ao RTC, quando da ocorrência de uma interrupção no pino INT do processaor M6800. A exclusão mútua no SOATER é implementada utilizando-se semaforos [51] para o controle ao acesso de regiões críticas de memória, sincronização de processos concorrentes.

Para o SOATER desenvolveu-se um conjunto de tarefas para acionamento da interface com um pequeno motor DC, aquisição de dados, comunição com a console do operador e comunicação com o nível II. Desenvolveu-se também o programa de polling para gerenciamento das interrupções. A seguir descreve-se a implementação das tarefas necessárias ao acionamento da interface com um micro motor DC e aquisição

de dados (velocidade do micro motor DC), tarefas de entrada e saída e os comandos do nível III.

4.4.2.1. Acionamento da interface e regulação de um micro motor DC

O algoritmo de acionamento do motor contitui-se básicamente de um algoritmo para controle do conversor estático que controla o valor médio da potência aplicado ao motor, Apêndice A. Quatro tarefas implementam este algoritmo. Três destas tarefas, ditas tarefas básicas de comando, acionam a interface de modo que o motor rode no sentido horário, sentido anti-horario ou fique desativado, estas tarefas são denominadas respectivamente HORAR, ANTIH, e DESAT. A quarta tarefa, denominada PARTE, ativa sequencialmente no tempo as três tarefas básicas de comando, bem como determina através de sua frequência o período total de tempo no qual as três tarefas básicas de comando devem estar restritas, definindo portanto o período do algoritmo de comando. Deste modo a tarefa PARTE realiza a atualização dos campos de frequência das três tarefas básicas de comando nos seus próprios decritores periódicamente no tempo segundo sua proópria frequência.

Os comandos são enviados ao conversor estático através da interface paralela do MATER.

A regulação da velocidade do motor é feita aplicando-se diferentes sequências de controle ao conversor

estático. As sequências estão aranjadas em uma tabela, "look up table". A partir de um apontador que indica a sequência corrente de controle aplicada ao conversor estático e da comparação entre o valor de referência incrementa-se ou decrementa-se o valor do apontador para a tabela, de forma a buscar uma nova sequência de controle para aumentar ou diminuir a velocide do motor, mantendo assim a velocidade do motor no valor de referência. São utilizadas duas tabelas, uma delas é a tabela corrente, e a outra é a tabela que esta sendo atualizada pelo nível II.

4.4.2.2. Tarefas de aquisição de dados

As tarefas de aquisição de dados são responsáveis pela obtenção do valor corrente da variável observada do processo. No caso do micro motor DC, é feito o cálculo da velocidade do motor em RPM, apartir dos valores obtidos da do circuito para deteção e aquisição de velocidade, Apêndice A.

4.4.2.3. Tarefas de comunicação com a console do operador e comunicação com o nível II

Diversas tarefas executam as função de interface de comunicação no nível III. Estas tarefas estão divididas em dois grupos: as tarefas de comunicação com o nível II e opcionalmente as tarefas de comunicação com a console do operador. A tarefas de comunicação com nível II executaram as função de: atualização do valor da variável do processo

no nível II, sincronização de relógio, atualização das tabelas de controle, "look up table", envio dos parâmetros de supervisão para o nível II, recebimento dos parâmetros de coordenação provenientes do nível II. Os parâmetros de supervisão são constituídos dos descritores das tarefas, relógio, posicção do apontador na tabela de controle e o valor atual da variável do processo. Os parâmeros de coordenação são contituídos de alterações nos descritores das tarefas, como por exemplo alteração da frequência de ocorrência de tarefas, e relógio para sincronização.

4.5. Software do nível II

O software do nível III do SDVC baseia-se no ETREC (Executivo em Tempo Real para Controle de Processos) [52]. O ETREC é um executivo em tempo real preemptivo, com prioridade mista. O conceito de tarefa no ETERC possui as mesma características decritas para o SOATER. Os decritores no ETREC possuem os seguintes campos.

- FREQ: indica a frequência de ocorrência da tarefa, somente para tarefas síncronas, tarefas assíncronas possuem este campo igual a zero. O valor deste campo é um múltiplo inteiro do RTC.
- CONT: contador da tarefa, é dado como a freqência em múltiplos inteiros do RTC, a cada ocorrência do RTC este campo é decrementado até ser igual a zero. Assim que o contador atinge o valor zero o status da tarefa é modificado para pronto para executar,

setando o campo de status, STAT, e o contador é reinicializado novamente com valor de frequência contido no campo FREQ.

- STAT: define se uma tarefa esta pronta para executar, quando setado para 1 ou bloqueada, quando setado para 0.
- STAK: apontador de pilha da tarefa.
- PRIO: valor inicial da prioridade da tarefa.
- DECR: decremento do status da tarefa.
- LINK: apontador para a próxima tarefa na lista encadeada circular.

Uma descrição detalhada do ETREC pode ser encontrada em [51].

Para o ETREC foram reescritas as rotinas de entrada em saída, ou seja comunicação com o operador e com o nível III, bem como introduzidas mais primitivas, semáforos, para sincronização de processos concorrentes.

4.5.1. Tarefas do nível II

Desenvolveu-se um conjunto de tarefas para o nível II do SDVC baseando-se no diagrama funcional obtido da decomposição do DFDC, figura 3.13. Estas tarefas são divididas em dois grupos: as tarefas para comunicação com o nível III e com a console do operador e as tarefas de supervisão e controle.

4.5.1.1 Tarefas de comunicação

As tarefas de comunicação são responsáveis pela comunicação com o nível I, console do operador e nível III. Neste trabalho apresentam-se as tarefas para comunicação com a console do operador e com o nível III. Foram também desenvolvidas as rotinas de comunicação para as interfaces seriais do CAMAÇARI. As tarefas de comunicação são responsáveis pela formatação, transmissão e recepção de mensagens, bem como a verificação da consistência das mensagens recebidas. No caso da comunicação com o nível III a consistência é verificada por retransmissão. Estas tarefas e rotinas de comunicação correspondem ao refinamento da transformação do DFDC do SDVC, figura 3.11.

4.5.1.2. Tarefas de controle e supervisão

No nível II do sistemas foram desenvolvidas tarefas que permitem a execução de algoritmos de controle e supervisão. Desenvolveram-se ainda tarefas com o objetivo de testar o software implementado, sendo assim desenvolveu-se um algoritmo com características bastante simples para o controle da velocidade do micro motor DC, bem como algumas tarefas de supervisão.

A regulação da velocidade do micro motor DC é conseguida através do envio de tabelas com as sequências de controle do conversor estático para o nível III. Estas tabelas foram conseguidas de forma heurística. Basicamente construiram-se, por inspeção, algumas tabelas para a regu-

lação da velocidade do micro motor DC em uma faixa de velocidades específica. No nível II o algoritmo de controle implementado compara o valor de referência com a variável do processo, no caso a velocidade corrente do motor. A partir do erro obtido escolhe-se uma determinada tabela de controle que contenha as sequência de controle do conversor estático que reduza o erro para zero. O algoritmo utilizado é bastante simples, sendo constituído de um conjunto de regras. Na figura 4.8 apresenta-se este algoritmo.

```
se erro = 0
então tabela corrente = tabela anterior
senão
   se erro < 0
   então
      se tabela corrente não é última tabela
         então tabela corrente = tabela anterior - 1
         senão tabela corrente = tabela anterior
      fimse
   senão
      se tabela corrente não é primeira tabela
         então tabela corrente = tabela anterior - 1
         senão tabela corrente = tabela anterior
      fimse
   fimse
fimse
```

Figura 4.8. Algoritmo básico de controle.

As funções de supervisão básicamente implementam a interface com o operador. Desenvolveram-se tarefas para mostrar a velocidade do motor, alterar parâmetros nos descritores das tarefas do nível II e III, ajuste de relógio dos níveis II e III e atuação direta sobre o conversor estático no nível III.

Neste capítulo apresentou-se a contribuição a implementação do SDVC. Apresentou-se ainda um algoritmo de controle para avaliar o software desenvolvido. No capítulo seguinte apresentam-se conclusões e perspectivas futuras do trabalho.

CAPITULO 5

CONCLUSÃO

O estudo e a contribuição para a implementação do SDVC proporcionaram resultados bastante concretos. Entre eles detacam se dois. O primeiro referente a pesquisa bibliográfica sobre sistemas distribuídos para controle de processos. Nesta pesquisa procurou-se compilar as informações enfatizando os elementos básicos destes sistemas. O segundo e mais importante referesse a experiência adquirida com o estudo e implementação dos níveis II e III do SDVC. Esta esperiência é principalmente caracterizada pela observação da necessidade de uma ferramenta eficiente para o projeto de sistemas de automação e controle.

Desta forma, introduziu-se uma ferramenta eficiente para a análise e documentação de Sistemas de Automação e Controle, o DFDC. Esta ferramenta possibilita a definição clara da estrutura lógica destes sistemas, além de proporcionar a redução do tempo necessário a implementação

do software, resultanto em diminuição dos custos de programação. A aplicação do DFDC produz uma documentação clara e eficiente, o que possibilita a otimização do processo de manutenção de Sistemas de Automação e Controle.

A aplicação do DFDC para o projeto do SDVC foi limitada a disponibilidade de hardware. Deste modo aplicou-se o DFDC para a análise, documentação e implementação dos níveis II e III do SDVC. Estes dois níveis foram implementados e testados para a regulação da velocidade de um micro motor DC. Os resultados obtidos se avaliados do ponto de vista qualitativo foram amplamente atingidos, uma vez que se pode mostrar a execelência da aplicação do DFDC como uma ferramenta eficiente para a análise e documentação de Sistemas de Automação e Controle. Do ponto de vista quantitativo os resultados obtidos foram aquém dos desejados porém não dos esperados. Isto pois, tinha-se pleno conhecimento das limitações do conjunto motor/sensor utilizado para os testes do software implementado. Independente dos resultados obtidos com o micro motor DC não terem sido satisfatórios do ponto de vista quantitativo, pelas limitações citadas, o sistema implementado poderia facilmente ser aplicado para o controle de processos com constante de tempo maior, como por exemplo controle de temperatura ou processos químicos lentos.

Outro ponto a destacar é a filosofia de programação empregada. Procurou-se desenvolver o software do sistema

totalmente orientado para tarefas, ou seja estruturou-se a análise e decomposição do sistema de modo a definirem-se as tarefas necessárias a implementação do sistema proposto. Deste modo procurou-se definir neste trabalho, uma tarefa como sendo uma unidade básica de programação ao invés de subrotinas ou procedimentos. A vantagem advinda desta abordagem resulta em uma programação mais simples, pois as estruturas de controle necessária ao software são definidas basicamente por duas entidades, o executivo em tempo real e as macros para sincronização de processos concorrentes.

Como trabalhos futuros sugerem-se:

- Uso de técnicas de inteligência artificial para executar as funções de supervisão do sistema, bem como detecção e correção de erros ou falhas;
- Alocar as funções de escalonamento dos níveis inferiores em processadores dedicados a esta função, com o objetivo de diminuir o "overhead" causado pelo algoritmo de escalonamento;
- Projetar o software do nível I de modo a utilizar-se linguagens objeto orientadas em sua implementação;

Finalmente deve-se ressaltar o aspecto conceitual deste trabalho. A introdução do DFDC no meio de Controle e Automação tem por objetivo minimizar o tempo investido na implementação destes sistemas, além de produzir uma documentação clara e consistente que possa ser concretamente utilizada para a manutenção do sistema implementado.

BIBLIOGRAFIA

[1] S. Bennett & D. A. Linkens: Editors, Computer Control of Industrial Processes, Capítulos 2 e 3, IEE Control Engineering Series, London: Peter Peregrinus, 1982.

[2] S. Bennett & D. A. Linkens: Editors, Real-Time Computer Control, IEE Control Engineering Series, London: Peter Peregrinus, 1984.

[3] S. Yoshii e S. Kuroda, "A distributed Control System Concept and Architeture", Proceedings of IFAC Workshop on Distributed Control Systems, Tampa, Flórida, 2-4 Outubro, 1979, pp. 53-63.

[4] W. Hofmann, "Process Automation with Decentralized Control Systems", Process Automation, 1980, pp. 3-6.

[5] R.T. Tenney e N. R. Sandell, "Structures for Distributed Decision Making", IEEE Trans. Systems Man and Cybernetics, Vol. SMC-11, Num. 8, 1981, pp. 517-527.

[6] R.T. Tenney e N. R. Sandell, "Strategies for Distributed Decision Making", IEEE Trans. Systems Man and Cybernetics, Vol. SMC-11, Num. 8, 1981, pp. 527-538.

[7] N.R. Sandell et al, "Survey of Decentralized Control Methods for Large Scale Systems", IEEE Trans. on Automatic Control, Vol. C-33, Num. 12, 1976, pp. 108-128.

[8] M. van Viegen, "A Distributed Data Acquisition and Control System", Microprocessing and Micro-programing, Num. 12, 1983, pp. 211-216.

[9] C. W. Rose e J.D. Schoeffler, "Distribution of Intelligence and Input/Output in Data Acquisition Systems", Prooceedings International Telemetering Conference, Los Angeles, California, 1976, pp. 705-720.

[10] J.D. Schoeffler e C.W. Rose, "Distributed Computer Intelligence for Data Acquisition and Control", IEEE Trans. on Nuclear Science, Vol. NS-23, Fevereiro, 1976.

[11] D.J. Fraade e S. Gast, "A Survey of Computer Networks and Distributed Control", Digital Computer Applications to Process Control, Van Nauta Lenke ed., IFAC and North Holland Publishing Company, 1977, pp. 13-29.

[12] Th. L. d´Epinay, "Structure of an Ideal Distributed Computer Control System", Proceedings of IFAC Workshop on Distributed Control Systems, Tampa, Flórida, 2-4 Outubro, 1979, pp. 65-73.

[13] L. Borsi e K. Pavlik, "The Concepts and Structures of Distributed Process Automation Systems", Process Automation, 1980, pp. 63-70.

[14] M. Kesunović, "Distribution of Architecture and Allocation of Functions in an Integrated Microprocessor-Based Substation Control and Protection System", Proceedings IFAC Real Time Digital Control Applications, Guadalajara, México, 1983, pp. 85-92.

[15] J.A. Stankovic, "A Perspective on Distributed Computer Systems", IEEE Trans. on Computer, Vol. C-33, Num. 12, pp. 1984, 11-18.

[16] J.M.S. Nogueira, "Sistemas Distribuídos da Automação Industrial", 2o. CONAI, São Paulo, 1985, pp. 283-294.

[17] A. Perkusich, J. H. F. Cavalcanti e G. S. Deep, "Sistema Distribuído Verticalmente Hierárquico para Controle de Processos", 6o. Congresso da Sociedade Brasileira de Automática, Belo Horizonte, Novembro, 1986, pp. 668-673.

[18] S. Bennett & D. A. Linkens: Editors, Computer Control of Industrial Processes, Capítulo 6, IEE Control Engineering Series, London: Peter Peregrinus, 1982.

[19] J.D. Schoeffler, "Distributed Computer Systems for Industrial Process Control", Computer, Vol. 17, Num. 2, pp. 11-18, Fevereiro, 1984, pp. 11-18.

[20] W. Friedewald e H.J. Charwat, "Design of Grafic Displays for CRT's in Control Rooms", Process Automation, 1980, pp. 7-12.

[21] K.F. Fruh, "Optimal Structure of Distributed Process Automation Systems", Process Automation, 1980, Editorial.

[22] R. Isermann, Digital Control Systems, Berlin, Springer Verlag, 1981.

[23] H. Lorin, Aspects of Distributed Computer Systems, New York, John Willey & Sons, 1980.

[24] G.A. Champine, R.D. Coop e R.C. Heinselman, Distributed Computer Systems: Impact on Managment, Design and Analysis, Amsterdam, North Holland, 1980

[25] G.M. Booth, The Distributed System Environment: Some Practical Approach, MacGraw Hill Book Company, 1981.

[26] G. van Bochmann, Concepts for Distributed Systems Design, Berlin, Springer Verlag, 1983.

[27] T. J. Harrison, "Distributed Data Processing Systems- Distributed Control Systems: Similarities an Differences. A Panel Discussion", ed. T. J. Harrison, New York, Pergamon Press, 1979, pp. 75-82.

[28] P. H. Enslow Jr., "What is a "Distributed" Data Processing system?",Computer, Janeiro, 1978, pp. 13-26.

[29] K.W. Plessman and H. Zoller, "High-Level-Language Programing In The Field of Process Control", IECON´86, anais do congresso, Vol.2, pp. 665-670.

[30] G.D. Bergland, "A Guided Tour of Program Design Methodologies", Computer, Vol.14, Num.10, Outubro 1981, pp. 13-37.

[31] Stephen S. Yau and Jeffery J.-P Tsai, "A survey of Software Design Techniques", IEEE Trans. on Software Engineering, Vol. SE-12, Num. 6, Junho 1986, pp. 713-721.

[32] E. Yourdon and L.L. Constantine, Structured Design, Englewood Cliffs, N.J.: Prentice-Hall, 1979.

[33] E. Yourdon, Techniques Of Program Structure and Design, Englewood Cliffs, N.J.: Prentice-Hall, 1975.

[34] M. A. Jackson, Principles of Program Design, New York: Academic Press, 1975.

[35] C. Gane and T. Sarson, Structured System Analysis: Tools and Techniques, Englewood Cliffs NJ: Prentice-Hall, 1979.

[36] A. Shenolikar, "Structured Data Analysis for Eletronic Design to Manufacturing Automation in a Multi-Vendor Environment", IECON'86, anais do congresso, Vol 2, pp. 687-692.

[37] P. T. Ward, "The Transformation Schema: An Extension of the Data Flow Diagram to Represent Control and Timing", IEEE Trans. on Software Engineering, Fevereiro 1986, Vol. SE-12, No. 2, pp. 198-210.

[38] G. Booch, "Object-Oriented Development", IEEE Trans. on Software Engineering, Fevereiro 1986, Vol. SE-12, No. 2, pp. 211-221.

[39] T. DeMarco, "Structured Analalysis and System Specification", in Guide 47 Proc., Guide Int., Inc., 1978.

[40] A. Perkusich, G.S. Deep e J.H.F. Cavalcanti, "Data and Control Flow Diagram and Process Automation", a ser publicado, IECON'87, anais do Congresso, Cambridge, Mass., 2-6 Novembro 1987.

[41] Object-Oriented Programming Workshop, in Sigplan Notices, Vol. 21 , Num. 10, Outubro 1986.

[42] R. Abbott, "Program Design by Informal English Descriptions", Commun ACM, Vol. 26, Num. 11, Novembro 1983, pp. 884-888.

[43] J.H.F. Cavalcanti, G.S. Deep, J.S. Rocha Neto, I.S.S. Silva, "Microcomputador para Aplicações em Tempo Real (MATER)", Anais do 1o. Simpósio em Controle de Processos por Computador, 1o. SICOP, Rio de Janeiro, 1981, pp. 207-209.

[44] Motorola Microprocessor Data Catalog, 1982.

[45] Zilog Componentes Data Book, 1984.

[46] Intel Components Data Catalog, 1981.

[47] M6800 - Microprocessor Application Manual, 1975.

[48] J.H.F. Cavalcanti, Software para Aplicações em Tempo Real - SOATER, comunicação pessoal.

[49] Alex V. Pinto, G.S. Deep e J.H.F. Cavalcanti, "Exclusão Mútua de Processos Concorrentes em Sistema de Controle por Microprocessador", 1o. Congresso Latino-Americano de Automática/5o. Congresso Brasileiro de Automática, UFPb, Campina Grande Pb, 3 a 6 de Setembro, 1984.

[50] Robert R. Brandt, ETREC - Executivo em Tempo Real para Controle de Processos, Tese de Mestrado, CCPgEE/CCT/UFPb, Campina Grande, Paraíba, 1985.

[51] E. Aemenski e G.Falk, Micromachines Electriques, Editions MIR-Moscou, 1977.

[52] R.R. Sule, B.J. Vasanth, T. Krishnan & M. Kumar, "Microprocessor-Based Speed Control System for High Accuracy Drives", IEEE Trans. on Industrial Eletronics, Vol. IE-32, Num. 3, August 1985, pp. 209-214.

[53] M.J. Gonzalez, "Deterministic Processor Scheduling", ACM Comp. Surveys, Vol. 9, 1977, pp. 173-204.

[54] S.S. Yau & M.U. Caglayan, "Distributed Software System Design Representation Using Modified Petri Nets", IEEE Trans. on Software Engineering, Vol. SE-9, Num. 6, November 1983, pp. 733-745.

APENDICE A

IMPLEMENTAÇÃO DA INTERFACE COM UM MICRO MOTOR DC

Como mostrado no capítulo 4 a interface com o micro motor DC é dividida em duas partes, a primeira correspondendo ao circuito de acionamento e a segunda ao circuito para detecção e aquisição da velocidade. Os valores dos componentes para o conversor estático e os buffers, figura A.1 são:

- A1 LM324
- T1,T3 BD140
- T2,T4 BD139
- R1,R4 10 KΩ, 1/4 W
- R2,R5 22 KΩ, 1/4 W
- R3,R6 1,5 KΩ, 1/4 W
- R7,R8 3,3 KΩ, 1/4 W
- RV 33Ω , 1 W
- RT 5,6 KΩ, 1 W

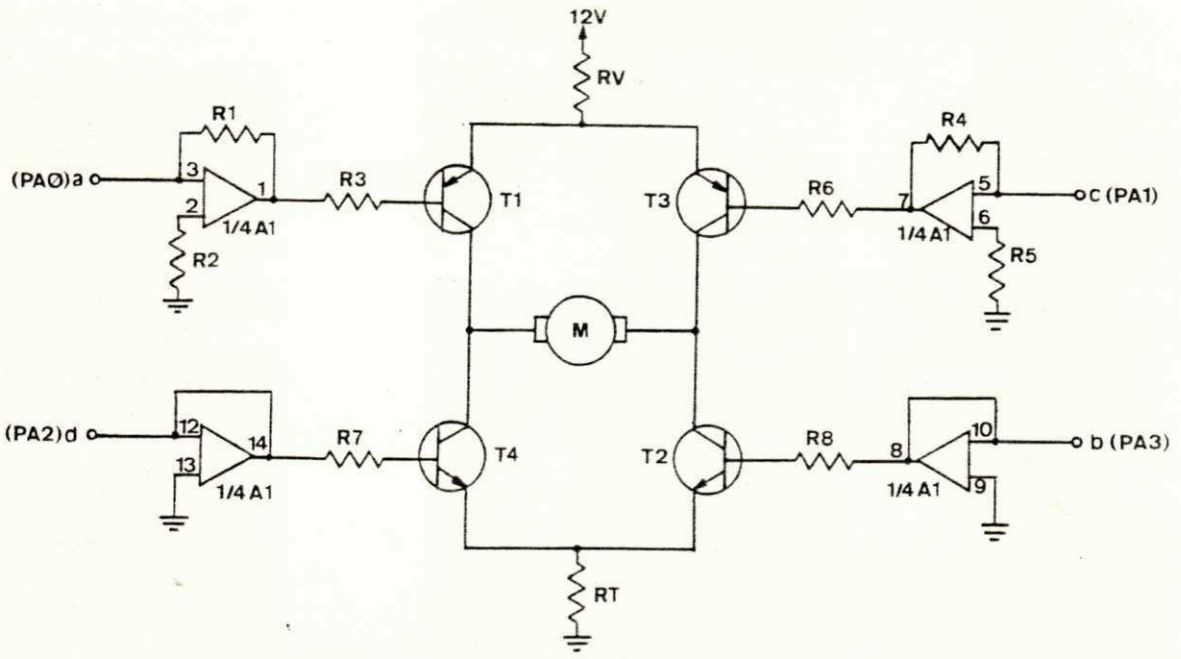


Figura A.1. Diagrama final do conversor estático com os buffers.

Como dito no capítulo 4, utilizou-se para detecção da velocidade do motor um disco perfurado juntamente com uma foto-emissor/detetor. O disco perfurado é construído com um total de 40 furos, o que proporciona uma precisão de 8 graus. O disco perfurado é montado em uma base metálica juntamente com o micro motor DC, e conectado a este por uma polia de borracha. Todos os sinais, e tensões de alimentação são conectados ao conjunto motor/sensor através de um conector DB 25.

O circuito para interface do conjunto motor/sensor com o microcomputador MATER foi implementado em um cartão para wire-wrap no mesmo padão do cartão utilizado para montagem do microcomputador MATER.