5 - Controladores

5.1 - Estruturas de *hardware* de um controlador

Para melhor perceber o controlo efectuado com recurso aos sistemas microprocessados é necessário perceber o controlo realizado tradicionalmente no domínio analógico.

Controladores tradicionais de processos

Devemos recordar que os PLC usados na automatização de processos industriais apareceram como uma substituição natural dos bancos de relés existentes, sem qualquer tipo de interface com o utilizador que não fossem os botões de arranque/paragem para controlo das operações, e luzes de sinalização do estado de progresso de uma tarefa.

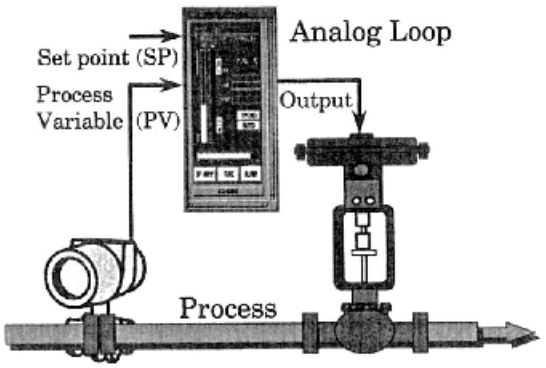

Fig. 5.1 – Malha fechada de controlo.

Os controladores de processos estavam fisicamente em contacto com o operador. Incluíam não só um indicador da variável do processo (PV), numa escala calibrada, mas também um indicador do ponto de funcionamento nessa escala (SP), e um indicador do sinal de saída para controlo. Este último indicador podia ser substituído por uma variável de saída (válvula, motor) obtida de um sinal de retorno do processo. O ciclo fechado, assim formado, designa-se por controlo em malha fechada (

Fig. 5.1): O sensor recolhe do processo a informação para enviar, por vezes através de uma cadeia de transmissão, ao controlador que aproxima a variável de saída do processo do valor de referência.

Uma forma mais generalizada de controlo distribuído pode ser observado na Fig. 5.2. Tal como anteriormente existem circuitos dedicados a cada sensor e a cada elemento controlado. No entanto, não existem ligações e circuitos dedicados directos. O que se pode começar a notar de diferente, entre o controlo tradicional e o controlo baseado em microprocessadores, é a oportunidade de interligar o domínio contínuo com o domínio discreto. Ambos os domínios devem ser convertidos para o domínio digital. Assim, alarmes detectados numa malha de controlo podem desencadear acções noutra malha. Também podemos observar que uma infinidade de circuitos de comando podem ser ligados a um mesmo porto sem que exista uma carga de tensão/corrente aplicada.

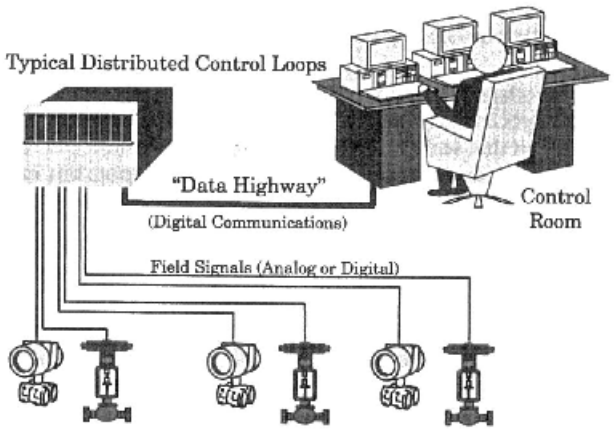

Fig. 5.2 – Diversas malhas de controlo partilhando um concentrador.

Arquitectura de controladores

Estes dispositivos electrónicos, chamados de controladores, muitas vezes possuíam cartas que realizavam funções especiais no seu estado inicial de desenvolvimento. Tradicionalmente podíamos encontrar uma carta especial para permitir a entrada/saída de sinais e outras especializadas na execução de algoritmos de controlo de processos. Outras, possivelmente, poderiam executar cálculos específicos, armazenar informação, ou gerir as comunicações com outros dispositivos através de uma rede digital. Duas arquitecturas emergiram e podem ser facilmente encontradas em qualquer sistema DCS.

Na primeira versão, todas as malhas de controlo partilham as funcionalidades de um controlador formado por diversas cartas com microprocessador (Fig. 5.3). Os microprocessadores encontram-se funcionalmente distribuídos em processamento das entradas/saídas, processamento do controlo, processamento das comunicações, etc. Nesta configuração em particular, o mesmo conjunto de cartas é grande parte das vezes usado em todas as versões de controlo. Como resultado todos os controladores são semelhantes de um ponto de vista de *hardware*, e torna-se relativamente fácil manter e alterar a estratégia de controlo. Podem ser reconhecidos por terem uma carta de saída, uma carta de entrada, uma carta de dados, uma carta de algoritmos, e um carta de comunicação. Regra geral, são acompanhados de cartas mais genéricas, tais como de alimentação de potência e cartas de redes.

A maior vantagem desta arquitectura está relacionada com o facto de todos os controladores apresentarem a mesma formação tornando a manutenção e alteração relativamente fáceis.

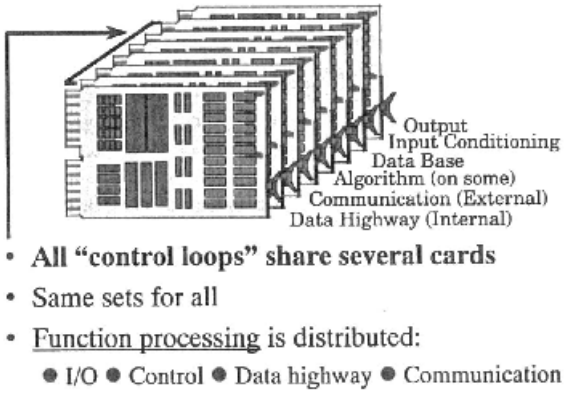

Fig. 5.3 – Controlador formado por cartas partilhadas pelos diferentes processos.

A segunda versão é formada por cartas microprocessadas individuais para cada malha de controlo (Fig. 5.4), onde algumas são usadas para malha de controlo, outras são usadas para realizar lógica de controlo. Tarefas como entrada/saída de dados e acondicionamento de sinais são executadas individualmente em cada carta microprocessada. Para além destas existe um conjunto separado de cartas para programação de algoritmos afim de expandir os fornecidos pelo fabricante, muitas vezes estas cartas são chamadas de multifunção ou multi-utilização. Diferentes combinações podem ser utilizadas, dependendo das necessidades. Para além das cartas já referidas o controlador pode possuir uma carta de rede de dados digital, uma carta de alimentação de potência, uma carta de diagnóstico, e uma carta que permita a todas as outras comunicarem entre si dentro do bastidor. A maior vantagem desta configuração é que a perda de uma carta reflecte-se apenas numa, ou num conjunto, de malhas de controlo.

Outras configurações foram aparecendo, mais difíceis de classificar, mas que retiravam partido das capacidades de processamento e de memória cada vez maiores.

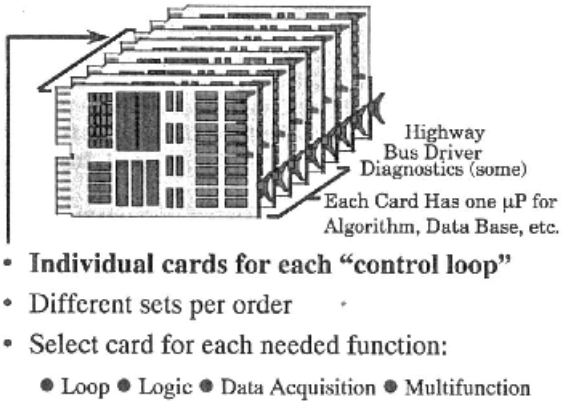

Fig. 5.4 – Controlador formado por cartas dedicadas a cada processo individualmente.

Da combinação das duas configurações anteriores resultou uma configuração que inclui todas as funcionalidades numa única carta. Muitas das arquitecturas usadas actualmente recorrem àquilo a que podemos chamar de controladores multifuncionais em vez de controladores de malha, controladores lógicos, e controladores de aplicações separadas. Esta aproximação também oferece a vantagem de integrar, num único conjunto, o *hardware*/*software*.

A coexistência de múltiplas malhas de controlo, residentes numa única carta, torna possível a criação de algumas estratégias de controlo multi-malha. Esta capacidade derrota a intenção de integridade das estratégias de controlo baseadas numa única malha de controlo, impossível de obter com qualquer tipo de inter-ligação da estratégia de controlo. A única protecção para as estratégias de controlo actuais são os controladores redundantes, que são agora mais práticos e baratos que antigamente.

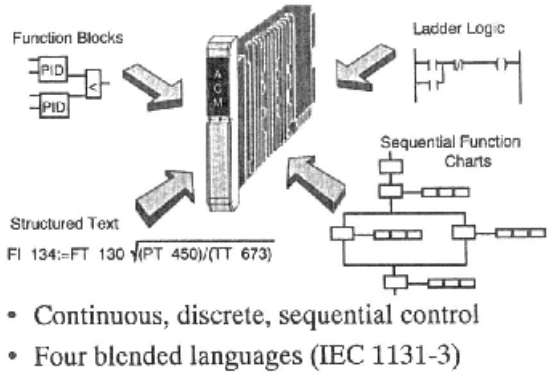

Fig. 5.5 – Diferentes linguagens de programação/configuração.

Será de esperar um vasto uso de linguagens de configuração dentro de um mesmo módulo (Fig. 5.5). A norma IEC 61131-3 da Comissão Electrotécnica Internacional define 5 linguagens de programação, algumas destas linguagens são blocos funcionais para controlo contínuo, lógica de relé para aplicações discretas, e linguagens estruturadas para cálculos complexos. A quinta linguagem diz respeito à linguagem *assembly*, que possui uma utilização pouco amigável.

5.2 - Estruturas de software de um controlador

Tal como os controladores analógicos influenciaram a criação do *hardware*, também influenciaram o desenvolvimento do *software*. O *software* deve desempenhar a função do controlador, à medida que as capacidades vão aumentando, novas funções devem aparecer. Ou seja, novas combinações de funções são agora possíveis, o que aumenta, por si, ainda mais as capacidades, a um custo reduzido. Esta é a área que tem alterado e expandido todo o campo de aplicabilidade do controlo de processos.

Programação

Quando se fala de computadores, encontram-se os termos *hardware*, *software* e *firmware*. O *hardware* é toda a parte da máquina que podemos ver e tocar. O *software*, por outro lado, é o conjunto de instruções dentro do computador que lhe diz como funcionar. *Firmware*, no entanto, é *software* gravado numa ROM e que se mantém inalterável, de forma a que algumas rotinas sejam executadas repetitivamente, tais como algoritmos para a realização de operações matemáticas.

A programação do *firmware* deve ser completa e muito especifica. Um processador num controlador de processos é programado para executar um número específico de rotinas básicas, definidas por comandos. Estas rotinas estão armazenadas num ROM, à qual, em princípio, o operador não tem acesso. Temos como exemplo uma rotina que faz com que uma CPU carregue uma instrução num determinado registo; execute o comando, que diz respeito a essa instrução; e que de seguida carregue outra instrução. Se não houver nenhuma instrução, o processador deve esperar, verificando, no entanto, periodicamente pela sua chegada, que a ocorrer desencadeia novo processo. Este funcionamento já foi visto, com algum pormenor anteriormente.

Para generalizar, as instruções dirigem a informação armazenada num endereço de memória específico, que se destina ao processamento e que deverá ser colocado num registo de dados. Dali é deslocada para uma unidade de aritmética lógica (ALU –*Arithmetic Logic Unit*), onde as operações lógicas ou aritméticas são realizadas. A informação pode temporariamente ser armazenada no acumulador, de forma a ser cruzada com outros dados recolhidos pela mesma ou outra instrução posterior. Logo que possível, os dados devem ser deslocados da CPU para a memória para aí serem armazenados.

A memória onde os dados se encontram armazenados são do tipo RAM, para que assim o programador tenha a ela acesso e a possa utilizar e alterar. O programador pode combinar os comandos que o processador pode executar, em arranjos específicos consecutivos que realizarão o seu objectivo.

Organização do tempo de execução das acções de controlo

Assim como o *hardware* pode apresentar diferentes configurações, também o *software* pode ser construído de diferentes formas para a mesma função. Num programa de *software*, o

processador irá ler todas as linhas de código escritas para uma função. Ao tempo consumido a execução de todo este código é dada a designação de *scan-time*. Existem diferentes metodologias para construir o *software*.

Num computador tradicional, independentemente da dimensão do programa, quer demore segundos ou horas a ser executado, o computador deve executar todas as instruções sequencialmente antes de chegar ao fim, voltando ao início do programa logo de seguida.

São poucos os programas de computador que podem parar a execução de código num determinado instante, embora isso seja possível, nomeadamente, recorrendo aos modos de adormecimento, hoje normais em qualquer processador.

Para que o processador possa desempenhar as suas tarefas, cada uma delas é executada por uma rotina especifica. Estas rotinas podem ser reunidas para executarem as configurações de controlo, cada um destas pequenas instruções juntas asseguram o funcionamento adequado de uma malha de controlo. Afim de obter uma execução de tempo real duas abordagens podem ser utilizadas.

Uma configuração, que chamaremos de duração fixa, todas as funções dispõem do mesmo tempo de execução. Nos controladores com tempos de execução fixa existem, por exemplo, 32 divisões ao longo das quais o controlador varre a rotina afim de realizar a estratégia de controlo. Cada um dos pedaços da estratégia de controlo deve caber num desses tempo de varrimento ou *slots* de tempo. Haverá então 32 funções disponíveis no controlador de uma biblioteca de funções. Durante o funcionamento do controlador, cada uma destas *slots* de tempo será varrida na mesma ordem sempre que o processador execute essa rotina. A primeira *slot* poderá albergar um algoritmos de PID, a segunda um bloco de soma, etc.

De notar que o tempo despendido a realizar um algoritmo PID poderá ser superior ao necessário para realizar um algoritmo de multiplicação. No entanto, usando esta estratégia, o mesmo tempo é atribuído para as duas. Haverá então algum tempo gasto na *slot* de multiplicação, que o processador não usa para realizar qualquer tarefa útil. Isto é altamente ineficiente para o desempenho do controlador, independentemente das tarefas e de quem as programa. Para além disso, quando as estratégias de controlo são alteradas, qualquer função cabe numa qualquer *slot*, assim são possíveis alterações *online* com bastante facilidade.

Os algoritmos nestes controladores estão colocados em cartas de algoritmos, e executam basicamente as seguintes tarefas: Lê o ponto de funcionamento desejado, lê o estado de funcionamento actual; executa a estratégia de controlo; coloca os resultados na carta de saída. Todos os cálculos usam os mesmos algoritmos, por isso, as rotinas não necessitam de ser duplicadas na memória para as diversas malhas de controlo. Após ser retida a alimentação, o controlador irá desempenhar exactamente a mesma função, mas não terá acesso aos dados anteriores das variáveis de controlo do sistema de processo.

A outra estratégia de controlo, a que chamaremos de tempo variável, as três tarefas anteriores podem demorar tempos de execução diferentes. Todas estas tarefas, com tempos de execução diferentes, são reunidas. O programador deve assim ter em atenção o tempo global de execução, de forma que o tempo de varrimento seja conhecido. Assim é possível ter tempos de varrimento diferentes para sistemas a executarem as mesmas tarefas. Esta vantagem, proporcionada pelo facto de cada algoritmo usar apenas do tempo de execução necessário, abre caminho à realização de muitas mais tarefas no mesmo intervalo de tempo. De qualquer forma é sempre bom relembrar que o programador deve ter sempre presente o tempo global de varrimento para que este não se torne demasiadamente elevado. Este aspecto torna a configuração e a introdução de alterações muito difíceis de concretizar.

Os avanços tecnológicos reflectem hoje, de alguma forma, estas duas metodologias. A velocidade de processamento disponível e as capacidades de memória existentes permitem diminuir em muito os efeitos destas limitações.

5.3 - Controladores de redundância

Nos dias de controlo em malha fechada por hardware, era muito raro uma malha de controlo ser backup por outra. O mais comum seria trocar para manual nos processos mais críticos, no entanto, os custos eram suportados para malhas de controlo singulares. Se o processo de controlo possui-se alguma forma de controlo que interliga várias malhas de comando, até uma simples cascata de controlo, realizaria um processo de backup nestas condições, a redundância fica muito complicada devido às componentes adicionais, que podem fazer com que a estratégia de controlo seja mais fraca.

O conceito da fiabilidade de uma malha fechada de controlo

Não existem duvidas de que há oportunidades de incluir redundância nos processos controlados em malha fechada. Muitos sistemas de controlo baseados em controladores utilizam a informação fornecida por diversos tipos de sensores, e respectiva análise, assim como outros dispositivos. Desta forma a malha de controlo deixa de ser independente, e quando falha todo o processo de pára. Para prevenir estas situações, todo o processo deve ser redundante, ou a malha fechada deve estar "entwined" com relés e circuitos de corte, que como já foi referido, são muito mais vulneráveis que o próprio controlador.

A disputa para a integridade de cada uma das malhas de controlo comecou no início do controlo distribuído, onde as malhas de controlo partilhadas eram a única forma, do ponto de vista económico, de usar microprocessadores. Os vendedores de sistemas distribuídos passaram a ser, deste ponto de vista, menos competitivos. Os vendedores de sistemas distribuídos desenvolveram então uma forma de contornar o problema. Um grupo de malhas de controlo partilhavam um único controlador de segurança. Devido ao seus custos, este esquema apenas era viável quando quatro, ou mais, controladores partilhavam o sistema. por seu lado, os vendedores de sistemas de controlo, que não o distribuído, passaram a reclamar que apenas a redundância de uma malha era adequada. Esta postura convenceu muitos clientes a requererem aos vendedores de sistemas distribuídos esta funcionalidade, fazendo com que estes, que usavam o sistema partilhado, tivesse que multiplicar os equipamentos de segurança e de uma forma proporcional os custos.

Computadores centrais redundantes

quando os computadores centrais ficaram disponíveis tornou-se muito difícil torna-los redundantes, as era pelo menos possível de executar como se esquematiza na Fig. 5.6. Para criar redundância, u, segundo computador deveria ser programado exactamente como o primeiro, de relembrar que a cópia electrónica de código ainda não era muito fácil. Não existindo uma forma automática de reflectir no segundo as alterações realizadas no primeiro computado. Um mecanismo de comutação, realizado na forma de uma rotina,deveria ser criada de forma a que

fosse possível verificar o alinhamento de todas as entradas e saídas. Quando uma falha ocorresse no primeiro, se notar que se escreve quando, e não se) a rotina de comutação deveria verificar cada linha de código antes que fosse autorizada a comutação. Como resultado, esta tarefa podiria demorar muito tempo, por vezes horas. Esta situação pode tornar-se completamente inaceitável em grande parte dos processos a controlar. Devemos ter em mente que a evolução dos grandes sistemas de controlo provem da área comercial, onde uma perda de funcionamento não é, regra geral, tão importante como numa linha de produção.

Fig. 5.6 – FIGURA 6.2.

Redundância nos microprocessadores com malhas de controlo partilhadas

Com o aparecimento dos primeiros controladores microprocessados, os primeiros dispositivos utilizavam a metodologia de malha de controlo partilhadas, devido principalmente aos custos. O facto de existir uma partilha das malhas de controlo possibilita a transferência de dados entre as diferentes malhas de controlo, o que permite introduzir estratégias complexas de controlo. Desta forma foi observado que era preferivel salvaguardar toda a estratégia de controlo. Mas também passou a ser necessário executar procedimentos de diagnóstico nestes novos processadores e respectivos circuitos. Um cirrcuito de monitorização é necessário para permitir visualizar o resultado do diagnóstico e determinar se é necessário mudar para o backup (*sistema de suporte*). Nesses dias de início, esta tarefa era muito cara. Optava-se então por partilhar diversos processos de backup de forma a poder manter os custos reduzidos (Fig. 5.7).

Fig. 5.7 – FIGURA 6.3.

Através do director de controlo e a ligação de comunicação digital existentes entre ele e o controlador, cada controlador pode manter as mesmas entradas e saídas sem necessidade física de as comutar, em vez de usar um controlador de reserva que mantenha o software de configuração da estratégia de controlo.

Quando o director de controlo detecta que o diagnóstico de um controlador deu negativo, comuta o mais rapidamente possível a base de dados do que está com problemas para o controlador de reserva, redireccionaa as entradas e as saídas, e continua a operar do controlador de reserva tal como se trata-se do controlador principal ou original. Esta tarefa tem que ser executada enquanto o controlador em rotura tem possibilidade de reposta.

Numa arquitectura onde todo o hardware/software é idêntico, não existe necessidade de executar um backup de todas as estratégias de controlo. Uma estratégia de backup 1 para N é possível porque toda a estratégia de controlo é salvaguardada em conjunto. A redundância é transparente quer para o próprio processo quer para o utilizador, com excepção de haver necessidade de tomar medidas para corrigir a falha. A base desta estratégia de redundância está na baixa probabilidade de um outro controlador poder falhar sem que o primeiro tenha sido devidamente reparado. Este métoo de salvaguarda é relativamente seguro, mas claro que, o facto

de se estar a retirar uma base de dados de um controlador com problemas deve mecerecer alguma atenção.

Redundância microprocessada em bastidores com controladores de uma carta

Na arquitectura de controladores que usavam diferentes conjuntos de cartas para cada controlador, a redundância é rara (Fig. 5.8).

Fig. 5.8 – FIGURA 6.4.

Quando os vendedores dessas mesmas estruturas ofereceram uma versão multifuncional, então uma grande variedade de sistemas de backup apareceu. Eles são sumariados por dois tipos muito generalistas, partilhado e um-para-um. Embora apenas só algumas versões se encontrem disponíveis no mercado, ainda se podem encontrar algumas por ai espalhadas.

Nestes bastidores com apenas cartas de controlo individuais que são feitas por muitos vendedores, apenas as cartas multifuncionais podem dar backup às suas semelhantes. A maioria automaticamente irá refrescar qualquer alteração que ocorra no controlador principal. Alguns fabriacantes permitem que o backup seja carregado com uma configuração diferente e chamam a isso benefício nomeadamente, para permitir a reconfiguração em funcionamento normal.

Uma forma de backup partilhado, 1 para N, está disponível em alguns fabricantes nos casos onde mais do que uma carta multifunção identicas é backup através de um controlador director para uma reserva. Neste caso, tal como nos bastidores que partilham os controladores, cada estratégia e configuração pode ser única. Existindo um interruptor de entrada saída que liga a carta de reserva áquela que está a falhar. Em algumas das versões parilhadas, uma carta de memória separada retém cada uma das configurações (mas não os valores correntes) de cada uma das cartas do bastido e fornece essa informação à medida das necessidades.

Falar do JTAG

Ambos os tipos (partilhados e 1 para 1) possuem cartas redundantes no mesmo backplane, algumas usam circuitos redundantes no backplane afim de obter segurança, afirmandoque o backplane é passivo e por isso susceptivél a algumas falhas. Alguns fabricantes não colocam cartas de redundância nem fontes de alimentação separadas. Se poder haver algum probema com a fonte de alimentação pode pensar-se em colocar uma alternativa. Alguns sistemas permitem um backup de bastido para bastidor usando diferentes cabines, com fontes de alimentação separadas, podendo existir diferentes circuitos de alimentação.

Em muitos controladores multifunção, a sua configuração goi realizada como se de um grande programa se tratasse. Se o controlador se encontrar a realizar uma operação de scan do programa, e ocorrer algum tipo de problema, a operação de controlo deve recomeçar do início no controlador de backup, muitoas vezes recomencando as tarefas sequências. Isto acontece sem que a configuração tenha sido transferida para o controlador de reserva.

No controlo de um processo térmico, farmaceutico, ou quimico esta situação pode tornar-se problemática. Alguns vendedores desenvolveram controladores redundantes que continuamente acompanhan o desenrolar do processo, as alterações realizadas em tempo real, ou noutras instantes no controlador principal. Isto garante que o processo de comutação seja muito mais suave e o processo se continue a desenrolar normalmente.