IST - 1995/1996
Sistemas Distribuídos para Ambientes Industriais
Trabalho de Laboratório – BITBUS

Controlo de uma Célula de Fabrico – II

1 Introdução

A célula de fabrico, apresentando funções mais ou menos específicas em termos do processo produtivo, é uma peça fundamental na organização de qualquer unidade fabril automatizada.

Em termos estritamente computacionais, o funcionamento duma célula de fabrico encontra-se habitualmente estruturado em torno de sistemas compactos, constituídos por uma base de processador inteligente complementada com elementos de *hardware* e *software* que lhe conferem características funcionais acrescidas: capacidade de entradas/saídas, interfaces com redes de comunicação, capacidade de execução concorrente de múltiplas tarefas. O último destes aspectos foi abordado, no laboratório, fundamentalmente através da utilização de um núcleo de sistema operativo multi-tarefa, o E286. Quando aos dois primeiros, serão tratados neste trabalho.

Como plataforma de comunicação iremos utilizar a rede BITBUS, uma infra-estrutura para interligação de elementos computacionais num sistema distribuído, muito utilizada em automatização industrial, para instrumentação e controlo. Esta infra-estrutura de comunicação suporta um número moderado (algumas dezenas) de nós, num espaço geográfico que poderá ascender a algumas centenas de metros.

A rede BITBUS utiliza uma interface eléctrica baseada na norma RS485. Os métodos de sincronização empregues podem ser síncronos, onde as taxas de transmissão podem variar de 500 kbps a 2.4 Mbps, ou assíncronos, onde as taxas de transmissão permitidas são 62.5 kbps ou 375 kbps. O valor escolhido para a taxa de transmissão, e o método de sincronização, condicionam a distância máxima coberta pela rede: ela pode ser de 30 metros para uma sinalização síncrona, enquanto que para sinalizações assíncronas as distâncias máximas a observar são de 1200 metros (a 62.5 kbps) ou 300 metros (a 375 kbps). Como protocolo de comunicação de dados é utilizado o SDLC (*Synchronous Data Link Control*).

Um nó duma rede BITBUS, embora dotado de algumas capacidades computacionais, terá como missão soberana assegurar a interligação da infra-estrutura computacional com o mundo exterior, através duma interface onde poderão ser ligados sensores, actuadores, etc. Teremos assim que um nó BITBUS será tipicamente constituído por uma plataforma computacional muito simples (usualmente construída em torno dum microcontrolador), complementada com interfaces genéricas (digitais ou analógicas) para outro equipamento existente na célula de fabrico, e naturalmente, uma interface com a rede BITBUS.

No presente trabalho iremos utilizar esta infra-estrutura de comunicação para estender, para o plano distribuído, o problema de controlo que abordámos a propósito da utilização do E286.

2 Descrição da Infra-estrutura Laboratorial

A infra-estrutura que iremos utilizar no laboratório é composta por um PC, capaz de comunicar com nós BITBUS situados em localizações remotas.

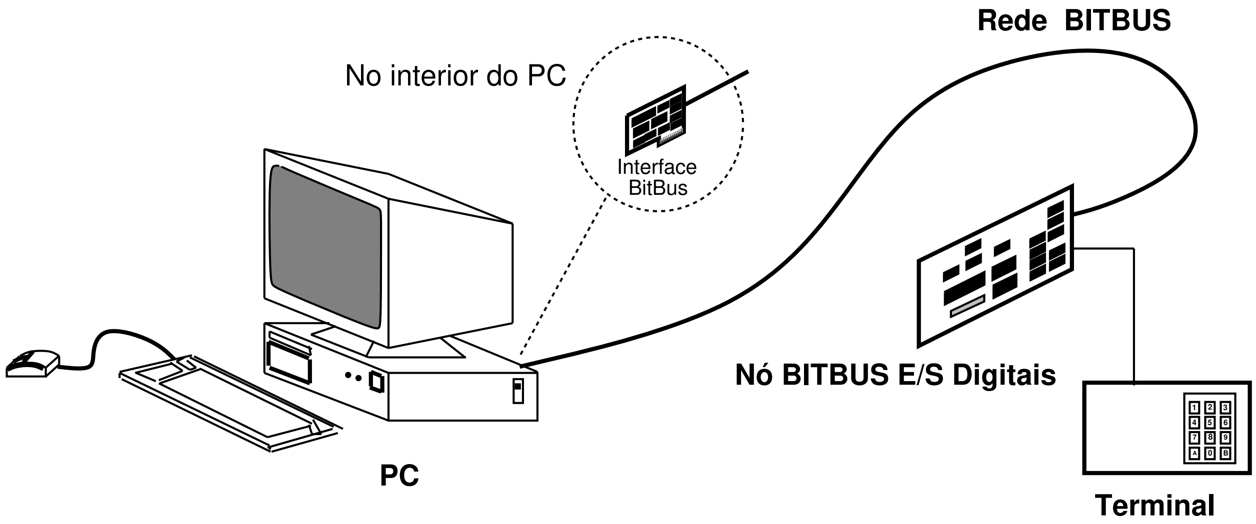

Figura 1: Infra-estrutura Laboratorial - Rede BITBUS

A ligação do PC à rede BITBUS é assegurada através duma placa de comunicações dedicada, montada no interior do PC e à qual as aplicações que aí são executadas podem aceder, via *Bus-ISA* (PC-XT). Por seu lado, ao nó remoto (suportando entradas/saídas digitais) encontra-se ligado um terminal de nível fábrica[1] rudimentar, com o qual o utilizador irá estabelecer diálogo. Utilizando este terminal iremos ainda simular a operação de alguns sensores, supostamente existentes na célula de fabrico.

O conjunto de módulos aplicacionais a desenvolver, irão permitir a um utilizador actuar remotamente com uma aplicação residente no PC, a partir do terminal nível fábrica.

3 Descrição do Ambiente de Desenvolvimento

Vimos anteriormente que a infra-estrutura a utilizar no laboratório era constituída por um conjunto diversificado de processadores, distribuídos pelos diferentes elementos computacionais. Assim:

- No PC, iremos ter o seu processador base (i80286, i80386 ou i80486) onde irão ser executadas determinadas aplicações, seja no sistema operativo MS-DOS ou no E286. Adiante enunciaremos as aplicações que os alunos deverão integrar neste ambiente computacional.
- Os nós remotos, **escravos** da rede BITBUS, são construídos em torno do microcontrolador i8044. Para além disso, todos estes nós têm já residente um executivo de tempo-real – **DCX-51** – e uma aplicação específica – o **RAC** (*Remote Access and Control*) – que suporta, no plano distribuído, um conjunto de serviços que os alunos irão utilizar na programação das aplicações destinadas a serem executadas nestes nós.

[1] *Shopfloor Terminal*, na literatura anglo-saxónica.

– Voltando ao PC, a placa de comunicação aí montada e que actua como **mestre** da rede de comunicação BITBUS, é igualmente construída em torno do microcontrolador i8044, possuindo também residentes o DCX-51 e os serviços do RAC.

Havendo diversos processadores envolvidos, na transferência de informação de pontos remotos da célula de fabrico até ao PC, é natural que seja necessário utilizar diferentes ambientes de desenvolvimento. Para cada uma das duas famílias de processadores, teremos:

– Aplicações BITBUS (nós remotos e comunicador instalado no PC). Estas aplicações irão ser desenvolvidas em PL/M-51 (uma linguagem de alto nível cuja sintaxe apresenta alguns pontos de contacto com o PASCAL e o C). O ambiente existente no laboratório permite gerar e agrupar imagens binárias, localizar a aplicação em endereços específicos e convertê-la para um formato especial (*Hex-Intel*) que possibilita o carregamento em nós da rede BITBUS. Está ainda disponível uma ferramenta interactiva (o *BitBus Monitor*) para auxílio ao desenvolvimento de aplicações em nós BITBUS. Esta ferramenta permite, por exemplo, carregar ficheiros em formato *Hex-Intel* nos diferentes nós, proceder ao lançamento dessas tarefas, aceder e modificar portos de interface com o exterior e imagens de memória. Em [6] abordamos de forma aprofundada o desenvolvimento de aplicações neste ambiente computacional. Os alunos são encorajados a desenvolverem as suas aplicações, a partir dos exemplos aí discutidos, também disponíveis em formato electrónico no directório `c:\bitbus\examples\`.

– Aplicações residentes no PC: de natureza marcadamente diferente, estas aplicações deverão ser pontualmente modificadas, de forma a tornar efectiva a sua integração com as aplicações BITBUS acima referidas. Estas aplicações compreendem:

  - uma aplicação MS-DOS, cujo perfil é fornecido em `c:\bitbus\examples\msdos\`, que irá ter como missão soberana proceder ao arranque de cada um dos nós que compõem a rede BITBUS. Entre as acções a realizar incluíu-se o carregamento dos ficheiros (em formato *Hex-Intel*) que contêm as aplicações, o seu lançamento pela criação das respectivas tarefas e eventualmente a materialização de outras acções de arranque relevantes. No apêndice B fornecem-se informações adicionais sobre esta aplicação.
  - uma aplicação E286, constituída pela aplicação de controlo da célula de fabrico, anteriormente desenvolvida pelos alunos. Adiante especificaremos a forma como esta aplicação deverá ser integrada com a operação da rede BITBUS.

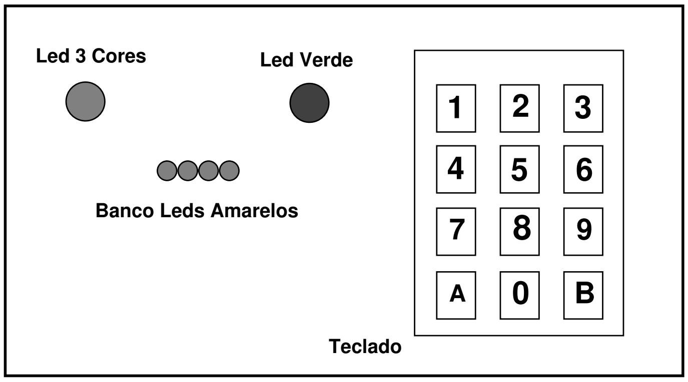

Figura 2: Terminal Nível Fabril

4 Definição do Problema

O sistema de comunicação que acabámos de descrever fornece o suporte adequado ao estabelecimento de interacções entre vários componentes, por hipótese situados em diferentes localizações físicas de uma dada célula de fabrico.

Pretende-se assim estender para o plano distribuído a aplicação de controlo de uma célula de fabrico anteriormente desenvolvida, numa arquitectura em que as funções responsáveis pelo controlo da célula estão residentes no PC, mas em que a sua operação é estimulada a partir de informações recolhidas de pontos remotos da célula de fabrico.

Em concreto, a ordem para aprovação de uma *pallete* deverá passar a ser introduzida a partir de terminais nível fábrica, remotos ao PC. Por outro lado, a transição da zona de ESCALONAMENTO para a zona PLATAFORMA (veja Figura 3) que simboliza a chegada efectiva de um filoguiado à plataforma de carga deverá resultar da mudança de estado (simulada) dum sensor, hipoteticamente ligado ao nó **escravo** da rede BITBUS.

De igual forma, a saída efectiva do filoguiado da plataforma de carga (i.e. o abandono da zona SAIDA do sinóptico) deverá igualmente resultar da mudança de estado do (hipotético) sensor destinado a fornecer esta indicação.

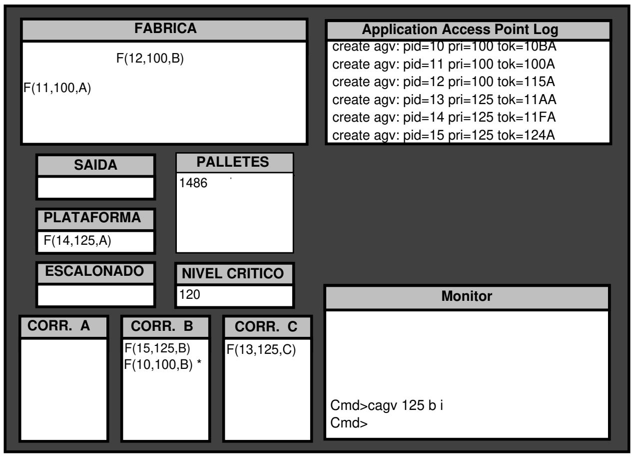

Figura 3: Sinóptico da Aplicação E286.

A aplicação distribuída que pretendemos construir será então constituída por duas vertentes nitidamente distintas:

- a tarefa executada no nó **escravo** da rede BITBUS, que tem como função recolher informação, a partir do terminal nível fábrica e de sensores supostamente existentes na célula de fabrico, disponibilizando-a através de uma interface de troca de mensagens.
- a aplicação de controlo, programada no sistema operativo E286 e residente no PC. Note-se que a aplicação MS-DOS atrás referida é igualmente executada no PC, mas unicamente durante a fase de arranque da aplicação distribuída a desenvolver.

A tarefa residente no nó agregado ao terminal de nível fábrica será responsável pelo recolha de ordens de aprovação de *palletes*, introduzidas no terminal nível fábrica por um dado inspector[2]. Será

[2] A informação a recolher, tendo em conta o programa já desenvolvido pelos alunos, consistirá unicamente no identificador desse inspector.

igualmente responsável pela leitura dos sensores supostamente existentes na célula de fabrico, a pedido da aplicação E286 residente no PC.

A aplicação E286 materializa a sua interface com a placa BITBUS montada no PC, através duma tarefa dedicada, que deverá ser desenvolvida pelos alunos e convenientemente integrada na aplicação já desenvolvida anteriormente. Adiante especificaremos a forma como deve ser construída esta tarefa, bem como as linhas mestras que devem nortear o trabalho de integração.

5 Especificação do Trabalho a Desenvolver

Esta secção contém uma especificação pormenorizada de cada uma das tarefas, que constituem a aplicação a desenvolver pelos alunos.

5.1 Tarefa Destinada ao Nó Terminal

Esta tarefa destina-se a ser executada pelo nó onde o terminal de nível fábrica está ligado. Tratando-se de um nó remoto, irá actuar como **escravo** da rede BITBUS. A tarefa a programar, que deverá ser a **única** a instalar neste nó, irá ser desenvolvida em PL/M-51, pelo que a utilização dos serviços do DCX-51 ou do RAC é directa. As funções a concretizar por esta tarefa são as seguintes:

- Estabelecer o estado inicial de todos os recursos relevantes (por exemplo, apagar todos os led's).
- Proceder à leitura do teclado, através do algoritmo apresentado no apêndice A. Este componente será utilizado para efectuar a aprovação de *palletes*, pela introdução dos **quatro** dígitos correspondentes à **identificação** do inspector[3].
  - Por cada tecla premida correspondente aos quatro dígitos de identificação deverá acender-se, a começar da esquerda para a direita, um *led* amarelo, no banco de quatro *led's* dessa cor. Ao completar-se a recolha dos quatro dígitos de identificação, todos os *led's* amarelos deverão ficar acesos.
- Tendo-se efectuado a recolha da **identificação** do inspector deve compor-se uma mensagem e enviá-la à tarefa residente no nó da rede BITBUS instalado no PC.
- Após o envio da mensagem o *led* destinado a reportar a aceitação da *pallete* pela aplicação E286 (*led* de três cores) deve passar de apagado à cor amarelo.
- O estado deste *led* de três cores poderá posteriormente transitar para vermelho ou verde[4].
  - Se o *led* transitar para a cor vermelha, tal significa que a capacidade da plataforma de carga (**quatro** *palletes*) foi excedida, devendo-se aguardar que uma *pallete* seja retirada. O terminal nível fábrica não poderá aceitar uma nova ordem para aprovação duma *pallete* enquanto este *led* não transitar de vermelho para verde.

[3]Preve-se que o identificador 0000 possa ser considerado um identificador inválido, devendo ser ignorado pelo sistema.

[4]Esta transição irá ser da responsabilidade da aplicação desenvolvida sobre o E286. Contudo, é possível gerar estes acontecimentos, para auxílio ao desenvolvimento do programa, por operação interactiva a partir do *BitBus Monitor*.

- Ao apresentar a cor verde este *led* indica que a *pallete* foi aceite para depósito na plataforma de carga. Este *led* deve assim permanecer até ser carregada a tecla "0", que o inspector utiliza para efectuar uma comprovação local, antes de poder iniciar a aprovação de uma nova *pallete*. Após a introdução da tecla "0" o *led* de três cores deve transitar para o estado apagado.

- Sempre que seja recebido um pedido para efectuar a leitura de um sensor[5], esta tarefa deve de imediato desencadear o seu processamento. Recordamos que supostamente existem na célula dois sensores: um que indica a chegada do filoguiado à plataforma de carga; outro que sinaliza a sua saída efectiva. As alterações nos estados destes sensores deverão ser simuladas pelo premir das teclas "A" e "B", respectivamente. A alteração do estado do sensor é sinalizada à aplicação E286 através duma resposta que emparelha com a ordem recebida anteriormente.

- Por seu lado, o *led* verde, indicativo de que o terminal se encontra operacional deverá encontrar-se, durante todas estas operações (leitura do teclado e processamento dos pedidos para leitura dos sensores), a piscar com um período aproximadamente igual a 1 segundo.

Note-se que a especificação desta tarefa encerra em si própria a definição dum conjunto de actividades de natureza concorrente (manipulação de ordens para aprovação de *palletes*, serviço de pedidos para leitura de sensores, piscar do *led* de actividade), pelo que a sua programação deverá considerar este requisito [6].

5.2 Aplicação E286

A integração da aplicação de controlo de uma célula de fabrico anteriormente desenvolvida com a operação da rede BITBUS deve observar os seguintes requisitos:

- Deve ser criada uma nova tarefa E286, que passaremos a designar por *servidor da rede BITBUS*, tendo como função realizar todas as operações de interface à carta BITBUS, instalada no PC. A especificação pormenorizada desta tarefa é dada abaixo.

- Cada ordem para aprovação de *palletes* passa a chegar à aplicação já existente através do *servidor da rede BITBUS*. Admite-se que os alunos definam uma metodologia que permita a esta tarefa uma operação optimizada, eventualmente contornando a interacção que anteriormente passava obrigatoriamente pelo servidor de pedidos.

- A função `agvMove` que modelava a movimentação dos filoguiados na célula de inspecção, deve ser substituída por pedidos para leitura dos sensores correspondentes, canalizados para o nó remoto da rede BITBUS através do correspondente servidor. À semelhança do caso anterior admite-se igualmente a definição de uma metodologia que permita uma interacção optimizada entre o *servidor da rede BITBUS* e os filoguiados.

O *servidor da rede BITBUS* será integralmente escrito na linguagem C. O acesso aos serviços do RAC faz-se através de funções, já parcialmente definidas no directório `c:\bitbus\e286ubi\`, que traduzem uma emulação daqueles serviços, tradicionalmente conhecida por *Universal BITBUS Interface* (UBI), aqui materializada ao nível do E286. As funções a realizar pela tarefa que temos vindo a discutir são as seguintes:

---

[5] Este pedido consiste numa ordem enviada da aplicação E286, directamente para esta tarefa.

[6] A sua concretização como uma máquina de estados poderá simplificar o código.

- Aguardar por pedidos para leitura de sensores. Sempre que estes ocorram esta tarefa deverá traduzi-los para `ordens` a enviar ao nó BITBUS instalado remotamente, passando a aguardar a chegada da correspondente `resposta`. Após a sua recepção e correspondente notificação dos filoguiados[7] o *servidor da rede BITBUS* deverá passar a um estado onde aguarda por novos pedidos para leitura de sensores.

- É também da responsabilidade desta tarefa receber a informação relativa à aprovação das *palletes*. Isso pode ser feito de forma cíclica[8]. Estando disponível, esta informação deverá ser processada:

  - Caso a plataforma de carga não possa albergar a *pallete* sob aprovação[9] a tarefa do *servidor da rede BITBUS* deve actuar, através dos serviços de manipulação de portos do RAC, directamente sobre o *led* de três cores, para que passe a apresentar a cor vermelha.
  - Caso contrário, ou logo que a *pallete* sob aprovação possa ser aceite, o *led* referido deverá passar a apresentar a cor verde. Após esta acção, deve regressar-se à actividade cíclica de aguardar por uma nova ordem de aprovação.

Note-se que o diálogo com a carta BITBUS por parte do correspondente servidor E286 pressupõe dois tipos de interação:

- Uma interação do tipo assíncrono, resultante da transacção de mensagens com entidades que podem não responder de imediato (indicação da mudança de estado dum sensor).
- Uma interacção do tipo síncrono, onde a chamada à UBI não retorna até que o pedido seja satisfeito (funções do RAC para manipulação de memória e portos).

Por esta razão torna-se necessário proceder ao emparelhamento das `respostas` recebidas a partir da interface com a carta BITBUS, com as `ordens` correspondentes, colocando-se numa fila de espera qualquer `resposta` de carácter assíncrono recebida no decurso de uma operação síncrona. Uma tentativa para recepção de uma `resposta` de natureza assíncrona deverá por isso começar por consultar esta fila de espera.

Dado que as funções fornecidas em `c:\bitbus\e286ubi\` não suportam esta funcionalidade, elas deverão ser obrigatoriamente modificadas pelos alunos, para passarem a operar como requerido [10]. É também da responsabilidade dos alunos a construção da rotina de envio de mensagem.

5.3 Aplicação MS-DOS

As funções associadas à aplicação MS-DOS, a desenvolver em PL/M-86 e a executar no PC, são as indicadas abaixo. Esta aplicação apenas se executa durante a fase de arranque. O acesso aos serviços do RAC faz-se através da **UBI** (*Universal BITBUS Interface*), uma emulação daqueles serviços ao nível do MS-DOS.

- Estabelecer o estado inicial dos recursos computacionais relevantes.

[7] Através duma metodologia que mais uma vez será da inteira responsabilidade dos alunos.
[8] Uma interface mais optimizada à carta da rede BITBUS passaria pela utilização de interrupções. Contudo, neste trabalho iremos seguir uma metodologia mais simples que passa pelo emprego de uma técnica de *polling*.
[9] Por se encontrar com a lotação de **quatro** *palletes* preenchida.
[10] Em [4] encontra-se uma especificação pormenorizada da **UBI**.

- Carregar o código relativo à aplicação BITBUS no nó **escravo** e lançar a respectiva tarefa.
- Terminar a sua execução, dando lugar ao carregamento e posterior entrada em execução da aplicação E286.

6 Organização do Trabalho no Laboratório

Os PC's existentes no laboratório possuem instalado, debaixo do directório `c:\bitbus` todo o *software* necessário para a realização do trabalho, estando já devidamente configurados.

Tratando-se de *software* sujeito a licença de utilização não está prevista a sua distribuição pelos alunos. De qualquer forma, a necessidade de uma infra-estrutura laboratorial específica para teste das aplicações, invalida a possibilidade de trabalhar noutro local que não seja o laboratório.

Os alunos deverão desenvolver as suas aplicações debaixo do directório `c:\bitbus\work\` podendo, se desejarem criar sub-directórios adicionais. Neste caso, deverão copiar para o novo directório de trabalho o ficheiro `help.bbm` sempre que aí desejarem utilizar o *BitBus Monitor*.

Debaixo do directório `c:\bitbus\examples\` os alunos encontram um conjunto diversificado e abrangente de aplicações tipo, que deverão estudar cuidadosamente. Este directório poderá ser copiado livremente.

O conhecimento aprofundado dos temas abordados na referência [6] é fundamental para a execução deste trabalho. No laboratório encontra-se disponível uma ampla bibliografia que engloba todos os manuais referentes ao equipamento e pacotes de *software* a utilizar.

7 Entrega do Trabalho

Este trabalho deve ser entregue em data e local $\boxed{\text{A INDICAR}}$. O material a entregar consiste numa *diskette* com o programa desenvolvido (fontes, objectos e makefiles) e numa listagem do mesmo. Admite-se que a listagem da aplicação E286 contemple apenas os aspectos de integração, devendo no entanto o conteúdo da *diskette* abranger toda a aplicação. Todos estes elementos devem vir identificados com o número do grupo de laboratório e identificação dos alunos que o compõem, devidamente acondicionados numa capa apropriada. A visualização posterior do trabalho será efectuada com base no trabalho entregue.

8 Perguntas e Respostas

P 1 *Devo simplificar sempre que possível?*

Sim! Simplifique o mais que puder, respeitando os requisitos do enunciado. Opte sempre pelo que der menos trabalho.

P 2 *Tenho liberdade para acrescentar ao trabalho funcionalidades adicionais ?*

Não! Uma eventual alteração à especificação contida no enunciado deve ser previamente discutida com o respectivo assistente.

O corpo docente reserva o direito de penalizar (até um máximo de **3 valores**) a introdução de aspectos funcionais totalmente ortogonais à natureza do trabalho. Exemplos: écrans sofisticados para indicação da composição do grupo, anúncios à *Coca-Cola*, etc.

P 3 *Tentei utilizar a ajuda no* BitBus Monitor *e apenas obtive a mensagem* `E$FNEXIST`.

O ficheiro `help.bbm` tem de se encontrar no directório onde o `BBM` é evocado, para que a ajuda possa ser prestada. Proceda à cópia daquele ficheiro, para o seu directório de trabalho, a partir de `c:\bitbus\examples\`.

P 4 *Como se processa a geração das aplicações MS-DOS, escritas em PL/M-86?*

Consulte o apêndice B.

P 5 *Como se processa a geração das aplicações, escritas em PL/M-51, destinadas a serem executadas em nós da rede BITBUS?*

Este tema é abordado de forma exaustiva em [6].

P 6 *Numa aplicação destinada a um nó BITBUS posso usar livremente a RAM interna do 8044? E a RAM externa ?*

A resposta é negativa em ambos os casos! A RAM interna é um recurso escasso, uma vez que é usada extensivamente pelo sistema. Sugerimos que coloque na RAM interna apenas as variáveis que aí tem de residir obrigatoriamente (por exemplo: o ponteiro para uma mensagem). As restantes variáveis deverão ser projectadas sobre a RAM externa, fora do espaço ocupado pelo sistema.

Os ficheiros `c:\bitbus\lib\mklibd\dcm44sp.a51` e `dcm44nm.a51` fornecem, respectivamente, o mapa de memória ocupado pelo sistema e inexistente nas cartas BITBUS.

P 7 *Existe alguma restrição na utilização da memória destinada ao armazenamento de constantes (variáveis que só podem ser acedidas em leitura, armazenadas no segmento de código) e código ?*

Positivo! As restrições são análogas às enunciadas na questão anterior.

P 8 *Como projecto uma determinada variável na RAM interna ou na RAM externa ?*

Consulte o manual da linguagem PL/M-51 para saber como deve fazer a sua declaração. Consulte igualmente os programas de exemplo apresentados em [6].

P 9 *Como posso saber o segmento e endereço de projecção dessa variável?*

O editor/localizador `RL51` gera um ficheiro (extensão `.m51`) com esta informação.

P 10 *Introduzi uma nova variável no meu programa e este misteriosamente deixou de funcionar ?*

Verifique a correcção do programa. Se o problema subsistir, a sua origem pode residir numa incorrecta projecção dos segmentos de dados na memória física. A conjugação com os módulos `c:\bitbus\lib\dcx51i.lib(dcm44_space)` e `(dcm44_nomem)` deverá evitar todo e qualquer problema.

Em último caso a análise do mapa da localização gerado pelo `RL51`, poderá fornecer indicações úteis. Estenda a sua análise a segmentos de dados, constantes e código, procurando eventuais incoerências.

P 11 *É necessário ter algum cuidado especial com a compilação separada de ficheiros e posterior combinação ?*

Na edição do programa há que assegurar a consistência de declarações *públicas* e *externas*. Na localização das imagens binárias, é necessário especificar a ordem pela qual os diferentes segmentos (código, constantes, dados) são projectados na memória.

Um comportamento irregular por parte do *BitBus Monitor* (por exemplo: indicação do erro `E$MAX$TASKS` sabendo-se de antemão que aquele limite não foi excedido) indicia uma incorrecta projecção de segmentos na memória física. Não deixe de ler a pergunta 10.

Dada a reduzida dimensão do programa a realizar, os alunos poderão optar por evitar a utilização de compilação separada minimizando deste modo os problemas acima referidos.

P 12 *No exemplo* `c:\bitbus\examples\distri\*.p51`*, onde está efectuada a declaração* pública *da variável* `NODE_ADD` *?*

As referências a entidades específicas dos nós BITBUS (portos, endereço, configuração) são obtidas a partir da biblioteca `c:\bitbus\lib\dcx51i.lib`, durante a fase de agrupamento de imagens binárias, realizada pelo `RL51`. As aplicações do utilizador apenas necessitam de as referenciar *externamente*. A título informativo fornece-se o ficheiro `c:\bitbus\lib\mklibd\dcm44io.a51`, onde são definidas aquelas referências.

P 13 *Tentei enviar uma* ordem *a partir de um nó escravo e não consegui. Qual é o problema ?*

No protocolo de transacção de mensagens do BITBUS somente o mestre do *bus* tem capacidade de enviar ordens. Um nó que seja escravo apenas pode emitir respostas, emparelhadas com as respectivas ordens. O protocolo de transacção de mensagens da rede BITBUS é discutido exaustivamente em [6].

Referências

[1] Intel Corporation. *PL/M-86 Programming Manual*, 1978.

[2] Intel Corporation. *MCS-51 Utilities User's Guide*, 1986.

[3] Intel Corporation. *DCX-51 Distributed Control Executive User's Guide*, 1987.

[4] Intel Corporation. *Universal BitBus Interface (UBI) User's Guide*, 1987.

[5] Intel Corporation. *PL/M-51 User's Guide*, 1988.

[6] José Rufino. Introdução ao Desenvolvimento de Aplicações sobre o DCX-51. Instituto Superior Técnico, Lisboa, Março 1995.

A Informações Técnicas sobre a Infra-estrutura Laboratorial

Na actual rede BITBUS existem dois nós, o instalado no PC e o nó remoto onde se encontra ligado o terminal de nível fábrica. Os endereços destes nós são `0FFH` e `02H`, respectivamente.

Este apêndice resume algumas informações técnicas sobre a memória disponível nos nós BITBUS, respectivo mapa de endereçamento e quando aplicável (nó remoto) os endereços dos respectivos portos de saída.

Adicionalmente são prestadas algumas informações adicionais acerca da ligação do terminal nível fábrica ao nó da rede BITBUS. Na primeira aula de laboratório pretende-se que os alunos completem a informação fornecida, através duma pequena investigação que deverão levar a cabo através da utilização do *BitBus Monitor*.

O *BitBus Monitor* é uma aplicação que sendo executada no MS-DOS permite estabelecer um diálogo interactivo com os diferentes nós de uma rede BITBUS, quer estes estejam localizados no PC, quer remotamente. O *BitBus Monitor* é evocado através do comando:

```
DOS> bbm
BBM>
```

O *BitBus Monitor* possuí a possibilidade de recorrer a ajuda em tempo-real, pela evocação do comando:

```
BBM> help
```

A partir daqui, o próprio utilitário fornece as indicações de como obter ajuda adicional. Entre a funcionalidade oferecida por este utilitário encontra-se a possibilidade de:

- Aceder a portos (leitura e escrita).
- Realizar operações lógicas (E, OU e OU-exclusivo) sobre portos.
- Aceder a posições de memória na RAM interna (leitura e escrita).
- Aceder a posições de memória na RAM externa (leitura e escrita).
- Aceder a posições na memória reservada a código (leitura e escrita).
- Carregar e salvaguardar programas.
- Criar e eliminar tarefas.
- Enviar e receber mensagens.
- Visualizar o conteúdo de mensagens.
- Obter informações sobre o sistema e nós da rede.

Informações complementares podem ser obtidas recorrendo ao respectivo manual ou à facilidade de ajuda em tempo-real.

A.1 Terminal Nível Fábrica

O esquema de ligações do Terminal Nível Fábrica ao nó da rede BITBUS encontra-se representado na Figura 4.

Pede-se que os alunos, por recurso directo ao *BitBus Monitor* preencham a Tabela 1, onde se encontra omissa a informação necessária ao controlo dos leds. Uma investigação semelhante deverá ser conduzida para o teclado.

	Endereço do Porto	Bit	Valor
Led verde aceso Led verde apagado			
Led 3 cores verde			
Led 3 cores vermelho			
Led 3 cores laranja			
Led Amarelo-A aceso Led Amarelo-A apagado			
Led Amarelo-B aceso Led Amarelo-B apagado			
Led Amarelo-C aceso Led Amarelo-C apagado			
Led Amarelo-D aceso Led Amarelo-D apagado			

Tabela 1: Controlo dos Led's

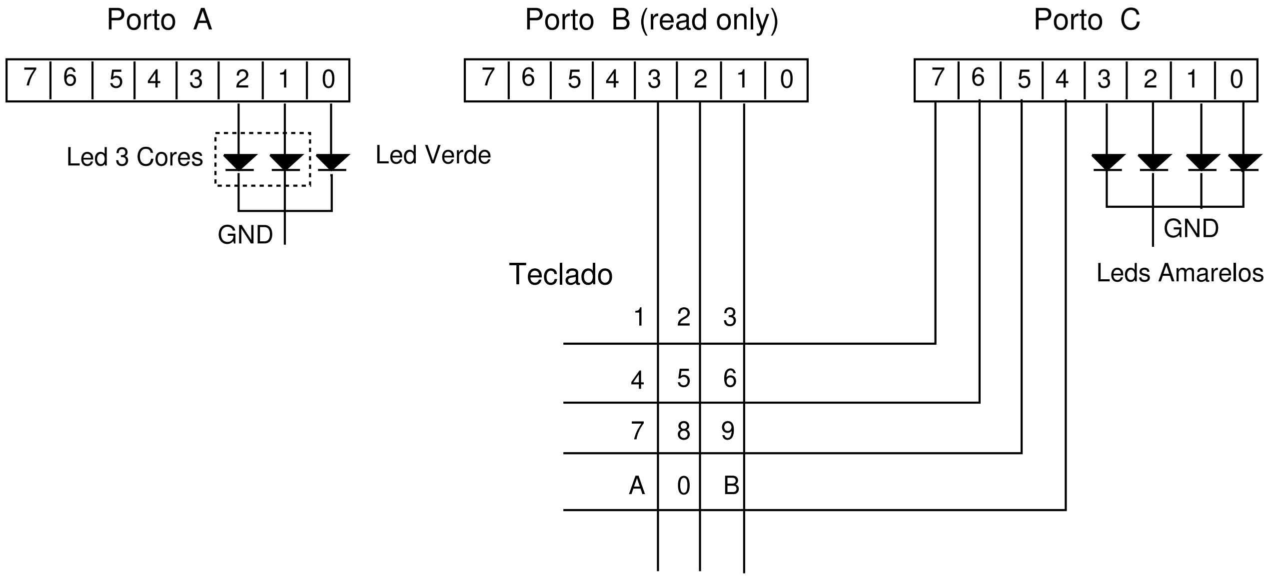

Figura 4: Esquema Eléctrico do Terminal Nível Fábrica

O algoritmo que deverá ser usado no teclado, para descodificação da tecla premida resume-se no seguinte: no porto de saída correspondente às linhas da matriz associada ao teclado deverá, ciclicamente, ser imposto um padrão onde apenas uma linha apresenta uma tensão correspondente ao valor lógico "X" (suponhamos a linha do topo). Se alguma tecla nesta linha (1, 2 ou 3) estiver premida, tal condição passará à coluna respectiva (por hipótese, se a tecla 1 for premida a coluna mais à esquerda terá o valor lógico "Y"); caso contrário apresentará um valor lógico contrário.

Memória	Endereço Início	Endereço Fim	Dimensão
Código (externo)	8000H	9FFFH	8K
Dados (externos)	000H	7FFH	2K

Tabela 2: Mapa das Memórias existentes nas placas

Função	Endereço	Leitura	Escrita	*Bit*
PORT A	0FFC0H	•	•	
PORT B	0FFC1H	•		
PORT C	0FFC2H	•	•	
MCS0	0FF80H-0FF87H	•	•	
	0FF00H,0FF01H	•	•	
MCS1	0FF88H-0FF8FH	•	•	
LED1	90H	•	•	•
LED2	91H	•	•	•
RDY/NE	B4H	•	•	•
NODE ADDRESS	0FFFFH	•		
CONFIGURATION	0FFFEH	•		
OPT0	92H	•	•	•
OPT1	93H	•	•	•
INT0	B2H	•		•
INT1	B3H	•		•

Tabela 3: Mapa dos Portos Entrada/Saída

B Embrião da Aplicação MS-DOS

O directório `c:\bitbus\examples\msdos` contém o embrião da aplicação MS-DOS a desenvolver. O código fonte, escrito em PL/M-86, está contido no ficheiro `bbload.p86`. As funções já concretizadas incluem as seguintes acções: efectuar uma operação de *reset* a nós específicos da rede BITBUS, carregar aí o ficheiro `bltask.hex` que contém uma das aplicações de exemplo discutidas em [6] e desencadear a criação das tarefas correspondentes.

O respectivo ficheiro executável é gerado a partir da *makefile* existente no mesmo directório, fornecendo o comando:

```
DOS> nmake
```

O ambiente para geração de aplicações é semelhante ao discutido em [6], para a linguagem de programação PL/M-51. Exceptua-se o filtro `udi2dos` que permite obter um executável MS-DOS, a partir do resultado do agrupamento de imagens binárias.

Esta aplicação deverá ser modificada pelos alunos na medida do necessário, de forma a integrar-se correctamente com o conjunto de todos os programas a desenvolver/integrar.

Seguidamente apresenta-se a listagem desta aplicação, tal como é fornecida nos exemplos distribuídos em formato electrónico.

C Emulação dos Serviços do RAC em S.O. Alienígenas

Os serviços do RAC (*Remote Access and Control*) podem ser evocados a partir de aplicações desenvolvidas em sistemas operativos diferentes do DCX-51, residentes em extensões aos nós da rede BITBUS, como por exemplo a infra-estrutura computacional do PC.

Exige-se que esse sistema operativo possua uma biblioteca para emulação dos serviços do RAC, destinada a esse sistema operativo. O diálogo com a carta BITBUS realizar-se-á naturalmente pela interface definida para a extensão.

No caso do sistema operativo MS-DOS esta biblioteca é fornecida pela Intel, para aplicações escritas na linguagem PL/M-86. Para o núcleo multi-tarefa E286 foi desenvolvida, para aplicações em linguagem C, uma biblioteca rudimentar para emulação de alguns dos serviços acima citados.

Os ficheiros referentes ao embrião desta biblioteca, bem como uma aplicação de teste dos mesmos para o ambiente multi-tarefa do E286 pode ser encontrado em `c:\bitbus\e286ubi\`. Os alunos deverão modificar, na medida do necessário, as funções fornecidas de forma a cumprir as especificações contidas neste enunciado (veja secção 5.2). Em [4] encontra-se uma especificação pormenorizada da **UBI**.

Seguidamente apresentamos as listagens destes ficheiros, tal como são fornecidos na distribuição em formato electrónico.