DEPARTAMENTO DE

ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

CONTROLADOR
PARA
SISTEMAS MULTIEIXO

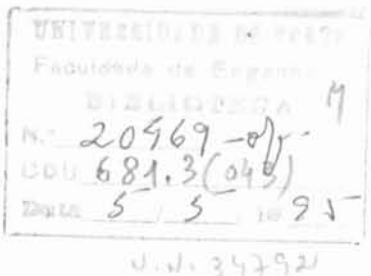
FACULDADE DE ENGENHARIA

UNIVERSIDADE DO PORTO

Rua dos Bragas, 4099 Porto Codex – PORTUGAL

José António Ferreira de Oliveira e Sá

CONTROLADOR
PARA
SISTEMAS MULTIEIXO



FACULADADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO
1991

Tese submetida para satisfação parcial dos requisitos do curso de Mestrado em Engenharia Electrotécnica e de Computadores (Perfil de Informática Industrial)

Tese realizada sob a supervisão do

Prof. Doutor José Carlos Diogo Marques dos Santos

Professor Catedrático do Departamento de Engenharia Electrotécnica e de Computadores

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

AGRADECIMENTOS

Desejo expressar os meus agradecimentos ao Prof. Doutor José Carlos Diogo Marques dos Santos pelo apoio prestado, tanto na fase de implementação do trabalho como na fase da escrita desta dissertação

Desejo agradecer ainda:

- ao Departamento de Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto, de um modo particular ao Prof. Doutor Francisco Freitas, pelo apoio prestado e meios postos à disposição, para a realização do trabalho.
- a todos quantos, directa ou indirectamente, me apoiaram e ajudaram.

Indice

1 INTRODUÇÃO

Nas máquinas com comando numérico, como nos robots, é comum a necessidade de controlar a posição e a velocidade de vários actuadores.

A utilização de um único computador, no controlo directo dos vários actuadores, conduz a soluções pouco modulares e com elevado número de ligações físicas. Pretende-se com este trabalho o desenvolvimento de unidades de controlo baseadas em processamento distribuído. O sistema desenvolvido será aplicado no controlo de um manipulador, em desenvolvimento na Faculdade de Engenharia da Universidade do Porto. A sua concepção deverá ser suficientemente modular de modo a permitir fácil utilização, não só no controlo do manipulador como em outras aplicações de controlo de posição e velocidade de eixos móveis.

1.1 DESCRIÇÃO DO MANIPULADOR

1.1.1 ESTRUTURA MECÂNICA

O manipulador tem seis graus de liberdade e destina-se a alimentar uma quinadora (fig 1.1). Três dos eixos são de movimento linear e três axiais.

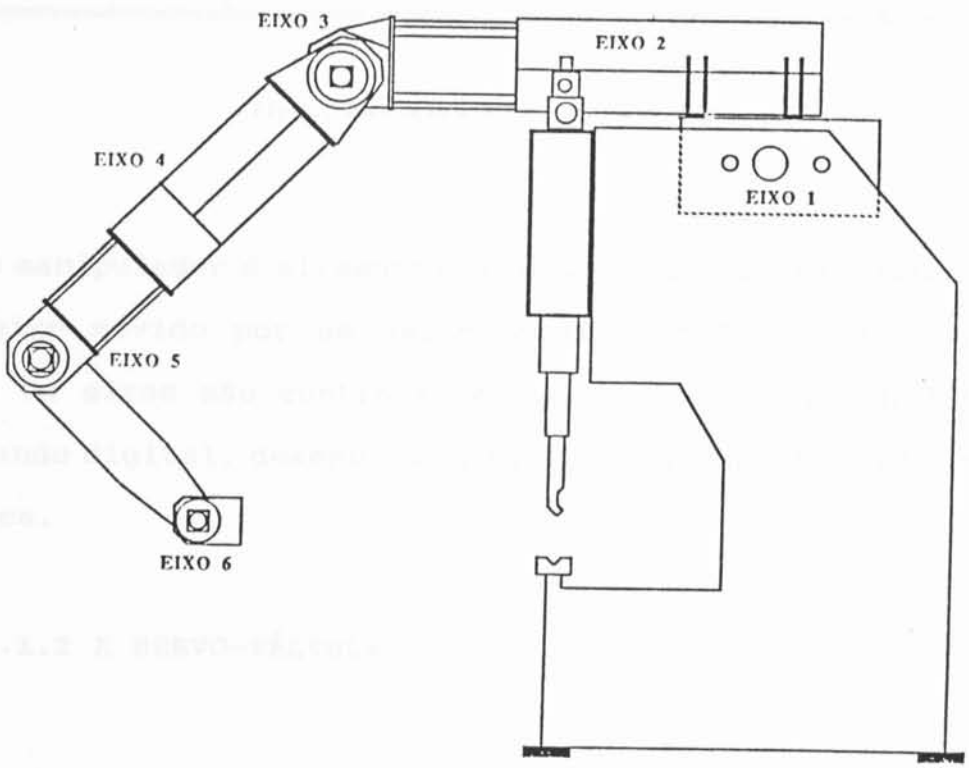


Fig 1.1 O MANIPULADOR

A resolução e alcance pretendidos para cada eixo são os especificados na fig. 1.2.

Eixo	Nº	Resolução	Alcance	Velocidade
Linear	1	1 mm	2000 mm	30 mm/s
Linear	2	0,02 mm	500 mm	30 mm/s
Rotativo	3	2'	38 º	38 º/s
Linear	4	0,02 mm	300 mm	30 mm/s
Rotativo	5	4'	76 º	76 º/s
Rotativo	6	20'	228 º	228 º/s

Fig 1.2 Resolução e alcance de cada eixo

O manipulador é alimentado por uma central hidraúlica, sendo cada eixo movido por um servo-actuador hidráulico, linear ou axial. Os eixos são controlados por servo-válvulas hidráulicas de comando digital, desenvolvidas pelo Departamento de Engenharia Mecânica.

1.1.2 A SERVO-VÁLVULA

Trata-se de uma válvula de comando digital. A variável de entrada é um número binário de oito bits, respresentado em sinal e grandeza. A figura 1.3 representa um corte da válvula, figurando na parte superior o motor passo a passo que acciona,

por meio de um excêntrico, as gavetas.

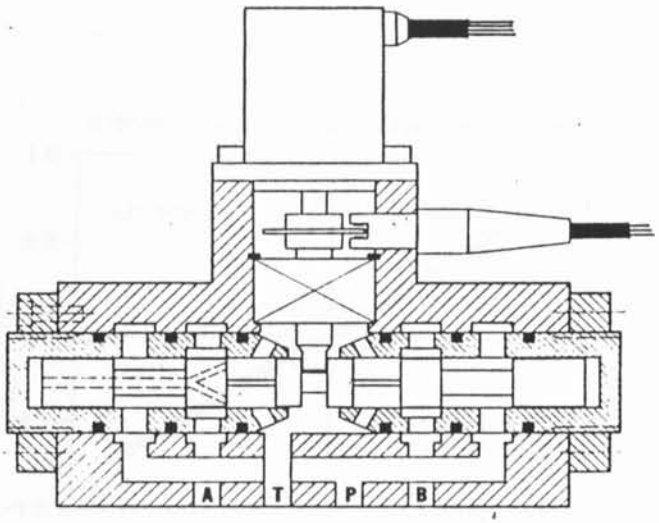


FIG. 1.3 VÁLVULA HIDRÁULICA

O valor da entrada representa o número de passos do motor, num ou no outro sentido, a partir da posição neutra. A válvula tem assim uma resolução consideravelmente limitada.

O caudal da válvula é função do deslocamento linear das suas gavetas accionadas pelo motor passo a passo [1]. De [1] foi retirado o quadro que, para a unidade utilizada, relaciona a posição do motor passo a passo com o caudal (Fig. 1.4).

As características mais marcantes da válvula são:

- Valores de entrada limitados ao intervalo -60 a +60 passos.
- Velocidade do motor passo a passo limitada a 600 passos por segundo.

- sempre que uma nova ordem implique inversão do sentido do motor passo a passo, o seu sistema de controlo faz uma paragem de oito milisegundos.

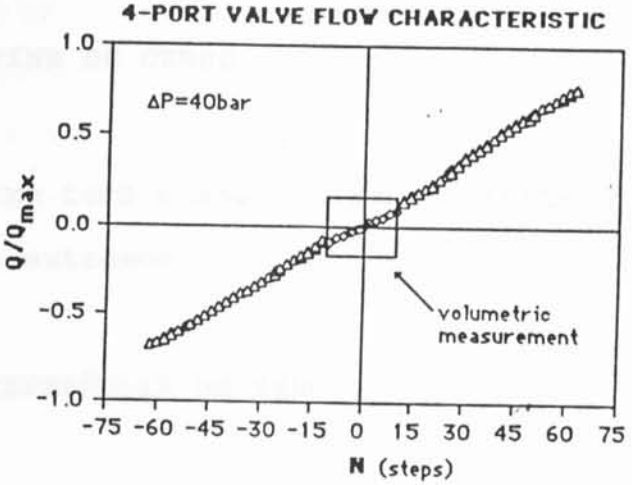


FIG. 1.4 CARACTERÍSTICAS DA VÁLVULA

A primeira destas características limita a resolução da velocidade do servo-actuador. A segunda é particularmente importante, por limitar a capacidade de aceleração do sistema.

1.1.3 CODIFICADORES

Os codificadores de posição adoptados são ópticos, do tipo incremental com dois canais em quadratura. Cada codificador dispõem de uma ou mais referências de zero. Estas referências geram um impulso com largura que pode variar entre T/4 e 3T/2,

sendo T o período correspondente à resolução do codificador.

Conforme o eixo os codificadores serão lineares ou axiais, de resolução adequada.

As saídas dos codificadores poderão ser em tensão ou do tipo colector aberto, dependendo do codificador disponível.

1.1.4 FINS DE CURSO

Cada eixo terá o seu movimento limitado por fins de curso, situados nos extremos.

1.1.5 REFERÊNCIA DE ZERO

Cada codificador dispõe de pelo menos uma referência, designada vulgarmente por referência de zero, que permite o alinhamento inicial. Nos casos em que a referência de zero não seja única, foi instalado um interruptor que é activado apenas em conjugação com uma delas. Por construção o espaço em que este interruptor está actuado é muito maior que o da referência de zero.

1.1.6 FREIO MECÂNICO

Os actuadores hidráulicos não dispoêm de capacidade de frenagem quando falha a alimentação. Este facto implica que o sistema de controlo, por motivos de segurança, disponha de meios para activar freios mecânicos.

1.1.7 CANAIS A/D

Num manipulador deste tipo cada eixo está sujeito a condições de carga particularmente diferenciadas dependentes, entre outros parâmetros, da posição dos outros eixo e da carga. A medida da pressão nos servoactuadores pode fornecer informação preciosa para o controlo de cada eixo.

A medida de velocidade será em principio calculada a partir da leitura de posição, poderá no entanto, pelo menos em alguns dos eixos, ser aplicado um taquímetro analógico.

O sistema de controlo deverá por isso dispor de dois canais analógicos de entrada.

1.2 DESCRIÇÃO DO TRABALHO

No capítulo 2 será feita uma análise de algumas opções quanto à arquitectura possível para o projecto e a justificação da opção feita.

A arquitectura escolhida para o sistema de controlo é baseada em microcontroladores ligados entre si por uma rede BITBUS. Razões económicas levaram á utilização do próprio controlador da rede, o processador 8044, no controlo do eixo. Esta opção cria algumas interdependências que condicionam opções tanto ao nível de software como hardware.

No capítulo 3 é feita uma abordagem de algumas características da rede BITBUS. A análise não pretende ser

exaustiva, mais informação pode ser encontrada em [6], [7], [8], [11], mas somente justificar opções tomadas ao longo do projecto.

No capítulo 4 é feita uma descrição do hardware desenvolvido para o projecto.

No capítulo 5 é descrito o software desenvolvido, tanto no computador central como nos processadores de eixo. A ênfase é dada ao software de comunicação, deixando a discussão do controlo dinâmico do eixo para o capítulo em que são analisados os resultados obtidos.

No capítulo 6 são analisados os resultados obtidos, sendo dada particular ênfase ao controlo dinâmico do eixo.

O capítulo 7 refere-se às conclusões do trabalho realizado.

2 ESCOLHA DE UMA ARQUITECTURA

2.1 INTRODUÇÃO

Os robots manipuladores são sistemas mecânicos constituídos por diversos elos interligados por articulações, lineares ou rotacionais. Estes sistemas exibem características, cinemáticas e dinâmicas, fortemente não lineares.

O controlo eficiente de um manipulador deve tomar em linha de conta estes fenómenos. Todavia uma análise mais detalhada revela que, em termos práticos, tal não é fácil de realizar. De facto, o elevado número de cálculos envolvidos levanta problemas de operação em tempo real, nomeadamente na manutenção de frequências de amostragem apropriadas. Por outras palavras, o elevado peso computacional dos algoritmos envolvidos requer um computador poderoso, a fim de permitir frequências de cálculo aceitáveis.

Este problema tem implicações económicas directas que explicam as soluções adoptadas ao nível industrial. Assim, é corrente a implementação de um algoritmo de controlo PID linear associado a cada eixo do robot [6]. Segundo esta arquitectura de controlo, as interligações entre os diversos elos são consideradas perturbações a eliminar pelo controlador de cada eixo. Apesar desta estratégia implicar desempenhos algo inferiores, tem todavia vantagens importantes :

- Soluções muito mais económicas.
- O ajuste dos controladores de acordo com a prática industrial já existente do controlo de processos.
- A construção de uma estrutura computacional distribuída, com um processador em cada eixo.
- Frequências de amostragem elevadas
- Elevada modularidade.

Atendendo a estas razões, entende-se ser do maior interesse o desenvolvimento de uma estrutura de controlo distribuído.

Num sistema deste tipo, competirá ao processador central a definição da trajectória e dos parâmetros a enviar a cada eixo. Se o manipulador, por sua vez, se insere num sistema de controlo mais complexo, compete igualmente a este processador a comunicação com o nível hierarquicamente superior.

O processador associado a cada eixo:

- Recebe comandos da unidade central.
- Informa o processador central sobre o seu estado.
- Faz a leitura dos sensores locais.
- Calcula os parâmetros a fornecer ao sistema de controlo de potência.
- Controla o estado dos actuadores locais.

2.2 ESCOLHA DO CONTROLADOR CENTRAL

O desenvolvimento da tecnologia de integração em larga escala, associada a uma feroz concorrência, tem conduzido à construção de computadores a preços impensáveis, ainda há poucos anos. No caso dos chamados IBM compatíveis, a divulgação tem sido

tão grande que estes computadores constituem hoje, de facto, um standard. Um projecto baseado em unidades deste tipo tem importantes vantangens como sejam:

- Relação custo capacidade, óptimo
- Aparecimento constante de novos computadores com capacidades mais desenvolvidas, mas mantendo a compatibilidade hardware e software.
- Fácil acesso a redes, nomeadamente as industriais.
- Grande quantidade de software já desenvolvido.

Tomando como unidade central da rede um destes computadores, é possível a construção de um sistema económico e que dificilmente ficará obsoleto. As razões expostas, aliadas à disponibilidade destes computadores, justificam a escolha de um computador do tipo "IBM PC", para unidade central deste projecto.

2.3 ESCOLHA DO PROCESSADOR DE EIXO

A escolha do processador a usar ao nível do eixo está dependente de três aspectos:

- Capacidade de cálculo.
- Preço da unidade de controlo de cada eixo.
- Existência de meios de desenvolvimento, em particular no que diz respeito ao trabalho envolvido nesta dimensão.
- Facilidade de interligação entre os vários processadores.

Da análise das exigências da aplicação, definidas no capítulo anterior, podemos concluir:

- Na representação de posição de qualquer dos eixos é suficiente o uso de um número binário de 16 bits.
- Para representar o parâmetro de saída, a grandeza a fornecer à válvula do comando hidráulico, é suficiente um número binário de 8 bit.
- O tempo de amostragem de cada eixo tem um limite inferior de 8 milisegundos, definido pelas características da válvula do comando hidráulico.

Nestas condições, escrevendo o software em "assembly", não é exigida uma grande capacidade de cálculo ao processador de eixo.

Vários fabricantes fornecem hoje processadores, e mesmo microcontroladores de oito bits, com capacidade para resolver o problema. O uso de microcontroladores permite uma considerável redução do número de componentes, com a consequente melhoria nos custos e fiabilidade [2].

O 8051, um microcontrolador de oito bits é hoje largamente utilizado na industria. Este microcontrolador na sua versão base tem as seguintes características [15]:

- Capacidade de endereçagem de memória de programa de 64 Kbytes.
- Capacidade de endereçagem de memória de dados de 64 Kbytes.
- 128 bytes de memória interna.
- 32 pinos programáveis como entrada ou saída.
- Cinco fontes de interrupção.
- Dois contadores de 16 bit programáveis como temporizadores.

- Processador booleano
- Porta série "full duplex".

Funciona nas versões mais correntes com relógio interno de 12 Mhz, executando nessas condições, a maior parte das instruções em 1 ou 2 microsegundos. Tem instruções de multiplicação e divisão, que executa em 4 microsegundos e um processador booleano muito útil em controlo industrial.

Vários fabricantes tem desenvolvido microcontroladores, baseados na arquitectura do 8051, vocacionados para aplicações específicas em controlo industrial. Todos eles mantêm compatibilidade com o 8051, no software e na arquitectura de base. Assim por exemplo:

- O 8044 e o PCB80C552, foram adaptados no sentido de melhorar a capacidade de comunicação por rede série.
- O UPI452, dispõe de meios altamente eficientes para a comunicação com outros processadores, usando bus paralelo. Dispõe ainda de entradas e saídas analógicas.

Atendendo às capacidades atrás referidas e a alguma disponibilidade de meios de desenvolvimento, foi decidido o uso de um processador da família 8051 para a implementação do controlador do eixo.

2.4 ESCOLHA DA REDE

No sistema que se pretende realizar, o controlador de cada eixo constitui por si um processador completo, dispondo de

memória, entradas e saídas. A maior parte da informação de que necessita para executar a sua função é obtida localmente. Do processador central são apenas recebidos os comandos, eventualmente com alguns parâmetros associados. Como resposta, o processador de eixo enviará informação sobre o seu estado actual. Em qualquer dos casos, a quantidade de informação envolvida é pequena, dando origem a mensagens muito curtas.

Como já foi referido as características da válvula utilizada implicam um período, entre actualizações dos seus parâmetros, superior a 8 milisegundos. Este é o limite inferior para o tempo entre comandos, a serem enviados do computador central a cada eixo. Em termos práticos, o controlador de cada eixo necessita de vários períodos de amostragem para fazer o controlo efectivo do eixo, assim o tempo entre cada novo comando será bastante superior ao limite mínimo referido.

O processador do eixo dispõe de capacidade suficiente para fazer a leitura dos sensores locais e tomar decisões. Nestas condições é possível dispensar a capacidade de ser este processador a iniciar comunicações.

Assim uma rede do tipo "master-slave" é uma solução aplicável a este projecto, sendo naturalmente a mais fácil de implementar.

2.4.1 REDE PARALELO

Uma solução correntemente utilizada para a interligação de processadores tem sido o uso de um bus paralelo. Existem mesmo alguns standards definidos para buses deste tipo.

No caso concreto dos microcontroladores, têm sido desenvolvidos alguns circuitos, especialmente vocacionados para este fim. Usando por exemplo o UPI452 [9] é possível, com recurso a muito pouco hardware, a implementação de uma eficiente rede de microcontroladores interligados por um bus paralelo.

O elevado número de linhas exigido conduz a soluções com graves inconvenientes, como sejam:

- Elevado custo.
- Baixa fiabilidade.
- Muito baixa capacidade de dispersão física dos vários elementos.
- Séria limitação ao número de elementos a associar ao bus.
- Baixa imunidade de ruído.

A grande vantagem deste tipo de bus tem a ver com a sua elevada capacidade de transferência de dados. Neste projecto esta capacidade não chega sequer a ser importante.

2.4.2 REDE SÉRIE

Os problemas apontados ao bus paralelo têm levado à sua progressiva substituição pelas redes série.

As ligações série baseadas no standard RS232, têm sido largamente utilizadas na ligação entre computadores e periféricos. A sua grande divulgação justifica o uso que por vezes é feito em aplicações industriais. Esta solução tem graves inconvenientes que a tornam pouco interessante. Assim:

- A expansão da rede exige modificação do hardware ao

nível do processador central.

- As velocidades de transmissão são baixas.
- As distâncias possíveis, entre o processador central e cada eixo, não são muito elevadas.
- A ligação é ponto a ponto, exigindo elevado número de linhas.
- A detecção de erros baseada apenas em bit de paridade, como é usal neste modo de transmissão, mostra-se muito pouco adequado ao uso em ambiente industrial.

Como alternativa à ligação ponto a ponto, baseada no standard RS232, têm sido ultimamente desenvolvidas redes do tipo bus multiponto ou anel.

Na última década, vários fabricantes ou associações de fabricantes, têm desenvolvido as suas próprias redes industriais de baixo nível as chamadas redes "Fieldbus". É o caso, na Europa, de um grupo de fabricantes alemães liderados pela Siemens com o "Profibus" e um grupo de fabricantes franceses com o "Fipbus". Na América, os mais conhecidos são o "Hart" (Rosemout) e o "I/A Field Bus" (Foxboro). Está neste momento a ser estudado pelo "Instrument Society of America" (ISA) uma normalização para uma rede industrial de baixo nível, baseada nas já referidas. Não se prevê uma conclusão breve dos trabalhos de normalização.

É corrente a implementação destas redes tendo como base circuitos específicos, é o caso por exemplo do "Programable HDLC/SDLC Protocol Controller" 8273 fabricado pela INTEL ou do "Advanced Data Link Controller" 68B54 fabricado pela Motorola. O uso destes controladores permite a construção de redes

eficientes, mas implica soluções hardware relativamente complexas.

A busca de uma solução económica leva-nos à exploração das capacidades incluídas nalguns microcontroladores para a constituição de redes série. Vários fabricantes têm instalado nos seus microcontroladores hardware, especialmente vocacionado para a implementação de redes série. Fazendo uso destas capacidades, é possível o desenvolvimento de redes de microcontroladores económicas e eficientes.

Restringindo a análise aos processadores compatíveis com o 8051 são três as opções:

- 8051 dispõe de uma porta série programável para a comunicação entre processadores.
- PCF80C532 com uma porta série especial para a constituição de uma rede designada por I^2C BUS.
- 8044 a associação num mesmo integrado de um processador, o 8051 e um processador dedicado a comunicações. Vocacionado para a implementação da rede BITBUS [7].

2.4.2.1 8051

A UART do próprio 8051 pode ser programada de modo a facilitar a comunicação em rede [15]. Com efeito esta UART, além da comunicação assíncrona usando uma palavra de oito bits, pode ser programada para transmitir palavras de nove bits. Na recepção, a UART pode ser programada de modo a receber ou não as palavras com o nono bit em 0. Usando esta facilidade, é possível

criar uma rede "master-slave" com o seguinte funcionamento:

Todos os "slave" são programados de modo a receberem apenas as tramas com o nono bit em 1. O "master" inicia a comunicação com uma palavra com o nono bit em 1 e os outros oito contendo o endereço de um dos "slave". Só o "slave" que reconhecer o endereço programa a sua UART para receber todas as palavras transmitidas. É assim possível, tendo embora vários processadores ligados a um mesmo meio físico, estabelecer ligação entre apenas dois deles.

Vários aspectos tornam pouco interessante o uso desta capacidade do 8051 para o projecto em causa. São eles:

- A máxima cadência de transmissão possível, com o 8051 a trabalhar a 12 Mhz, é 31,5 Kbits/s.
- A criação de uma rede minimamente fiável pressupõe a existência de um protocolo que permita a detecção e correcção de erros. Tal protocolo, por muito simples que seja, ocupa recursos do processador em detrimento do programa do utilizador [3].
- O programa de recepção terá de funcionar necessáriamente associado a uma interrupção de alta prioridade, de modo a não serem perdidos dados nem atrasar as comunicações. Este facto pode comprometer o uso do 8051 em aplicações em tempo real.

2.4.2.2 PCF 80C532

Processador baseado no 8051, com algumas características bastante melhoradas que o vocacionam para o controlo de processos [9]. A característica mais interessante, do ponto de vista desta análise, é sem dúvida a porta série de que o processador dispõe. Esta porta foi especialmente desenhada para a comunicação entre processadores, usando uma rede designada por I^2C BUS.

I^2C BUS é um bus série desenvolvido pela Philips, Signetics e Intel que, como o próprio nome sugere, se destina à ligação entre circuitos integrados. O bus é do tipo "multimaster", existindo circuitos capazes de funcionar como "master" ou como "slave", outros apenas como "slave". No segundo caso, o fabricante desenvolveu vários circuitos tais como, controladores de "display", memórias, relógios em tempo real, conversores A/D e D/A . Os circuitos capazes de funcionar como "master" ou "slave" são microcontroladores da família 8048 ou 8051.

Neste bus a comunicação é feita através de duas linhas, SCL que transporta o relógio e SDA os dados, ambas do tipo colector aberto. Sempre que o bus está livre ambas as linhas são matidas em nível lógico alto por resistências externas. A comunicação é iniciada por um "master" com uma transição de 1 para 0 na linha SDA, enquanto a linha SCL se mantem em nivel lógico 1. Isto provoca a activação de todos os "slave" presentes na rede que lêem o estado da linha de dados, nos próximos 8 ciclos de relógio. Os oito bits lidos são interpretados como sendo 7 bits de endereço e 1 de controlo, com a indicação leitura ou escrita

(Rd/Wr). O "slave" que reconhecer o endereço força SDA a 0, durante o nono ciclo de relógio gerado pelo "master". É este o modo de o "slave" demonstrar o reconhecimento do endereço e a disponibilidade para a comunicação.

Se o "slave" fez o reconhecimento do endereço, a ligação está estabelecida para o "master" ou o "slave", de acordo com o bit de Rd/Wr, transmitirem ou receberem dados. O relógio de sincronismo é, em qualquer dos casos, gerado pelo "master". Se nenhum "slave" reconhece o endereço, o "master" repete a tentativa. A ligação é assim estabelecida com um único "slave", ficando todos os outros desactivados.

Se o bit Rd/Wr estiver a zero, a transmissão é do "master" para o "slave". Neste caso o "master" tramsmite a informação e, ao nono ciclo de relógio, o "slave" deverá fazer o reconhecimento forçando a zero a linha SDA.

Se a linha Rd/Wr estiver a 1, a transmissão é do "slave" para o "master". Compete agora ao "slave" controlar o estado de SDA durante os oito primeiros ciclos de relógio, enviando assim a informação. O master confirma a recepção forçando SDA a 0 no nono ciclo de relógio.

A transmissão termina sempre por um sinal de fim de transmissão que é a coincidência de um degrau ascendente de SCL com SDA em 0. Este sinal reactiva todos os elementos ligados à rede que ficam a aguardar um novo sinal de início de transmissão.

As linhas SDA e SCL, do tipo colector aberto, possibilitam a arbitragem numa rede em que existam vários "master".

O facto de as linhas serem de colector aberto limita consideravelmente a sua imunidade ao ruído e comprimento.

A frequência do relógio, nesta rede, pode atingir os 114 Khertz permitindo assim velocidades máximas de transmissão abaixo dos 100 Kbit/s.

Neste bus as mensagens são ao byte e sem qualquer protocolo estabelecido ao nível "data link".

Pela análise feita pode concluir-se ser o I^2C BUS adequado à comunicação entre processador e periféricos ou vários processadores, apenas em ambiente muito protegido do ponto de vista de ruídos e de dispersão física muito limitada. Para a aplicação que se pretende não é a solução mais interessante.

2.4.2.3 8044

Este circuito resulta da associação num único integrado de um processador 8051 e um processador de comunicações (SIU) [7].

A comunicação entre os dois processadores é feita através da memória interna que é de duplo acesso.

A transferência de sinais de controlo, entre o processador e a SIU, é feita através de conjunto de registos especialmente dedicados (SFR) que foram acrescentados aos já existentes no 8051.

A memória interna do processador foi alargada (em relação ao 8051) para 192 bytes e a porta série assincrona deixou de existir. Em todos os outros aspectos, a arquitectura do 8051 foi mantida. A compatibilidade de software é completa.

A unidade de comunicações série (SIU), um processador

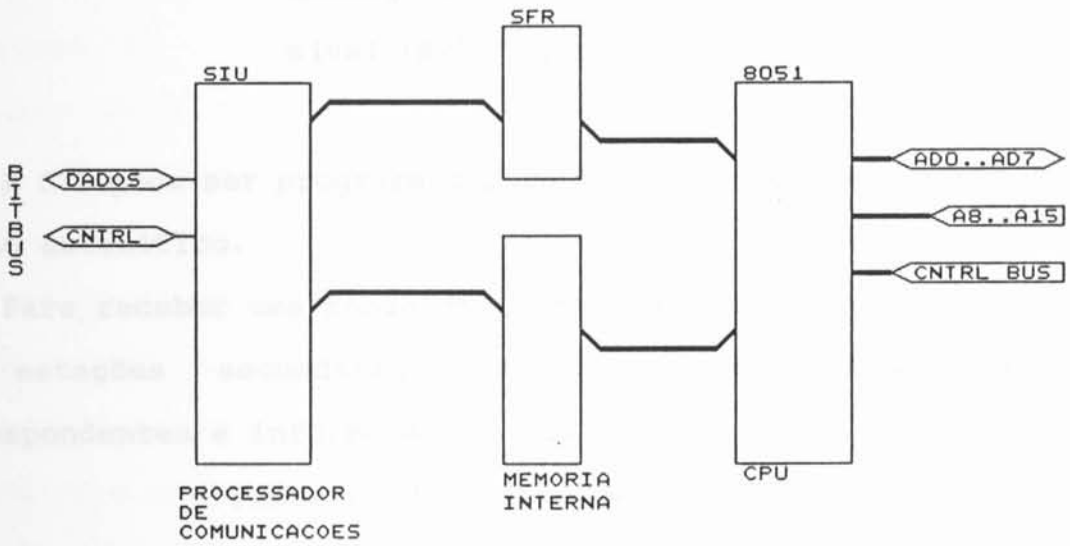


FIG. 2.1 MICROCONTROLADOR 8044

dedicado, funciona de um modo completamente autónomo em relação ao processador. Esta unidade de comunicações implementa um protocolo baseado no standard IBM Synchronous Data Link Control (SDLC).

A SIU, sem intervenção do processador, tem capacidade para:

- Inserir e reconhecer flags no início e fim de cada mensagem.
- Reconhecer, pelo endereço, as mensagens que lhe são dirigidas.
- Fazer a detecção de erros.
- Reconhecer e responder, a mensagens de, informação (I), receptor disponível (RR), receptor não disponível (RNR), rejeição (REJ) e mensagens de interrogação não numeradas (UP).
- A SIU pode enviar, sem intervenção do CPU,

mensagens de informação (I), receptor disponível (RR), e receptor não disponível (RNR).

A SIU pode ser programada para funcionar em modo automático ou não automático.

Para receber uma mensagem em modo automático, o modo usado nas estações secundárias, o CPU coloca nos registos correspondentes a informação:

- Ponteiro do buffer de recepção.
- Comprimento do buffer de recepção.
- Buffer de recepção vazio.

Se a estação for interrogada, e o buffer de transmissão estiver vazio, gera uma mensagem RR, caso contrário envia uma mensagem de informação.

Quando for recebida uma trama de informação, a SIU envia uma mensagem de reconhecimento ou pedido de repetição no caso de erro. Quando a mensagem é válida, interrompe o CPU e fornece-lhe o endereço do buffer de recepção, num registo de uso especial.

Para transmitir uma mensagem o CPU cria-a, num buffer em memória interna, e preenche os registos correspondentes com a informação:

- Ponteiro do buffer de transmissão.
- Comprimento da mensagem.
- Buffer de transmissão cheio.

A SIU, na próxima mensagem de interrogação que receba, faz a transmissão. Se, em resposta, receber um reconhecimento positivo fará a interrupção do CPU, se for negativo fará a retransmissão da mensagem, na próxima interrrogação.

Em modo não automático, o CPU deverá gerar e descodificar os campos de controlo das mensagens, e responder a todo o protocolo SDLC, é o modo utilizado na estação principal. Neste caso a SIU funciona em modo Assincronous Balanced MODE (ABM).

O processador de comunicações tem possibilidade para suportar duas topologias de rede, anel ou bus.

No modo em anel, os dados são recebidos no pino 11 e retransmitidos no 12, com o tempo de um bit de atraso. Neste modo é possível a transmissão com velocidades até 1 Mbit com relógio externo, ou até 375 Kbit com o relógio codificado nos dados.

No modo bus, é possível a transmissão em modo síncrono com velocidades até 2,4 Mbits/s, ou com o relógio codificado nos dados com velocidades de 62.5 ou 375 Kbits/s.

No modo de transmissão com o relógio codificado nos dados, ("self clocked") a codificação e descodificação do relógio é feita, em modo "Non Return to Zero Inverted" (NRZI), por hardware contido no próprio circuito.

A SIU, sem qualquer hardware exterior, faz na emissão a inserção de 0 a seguir a cada bloco de cinco 1 e, na recepção faz a sua eliminação.

É possivel usando o 8044 a construção de uma eficiente rede de comunicação entre microcontroladores. O facto de, nas estações secundárias da rede, o processador de comunicações funcionar em modo automático, limita ao mínimo a ocupação do outro processador nas comunicações .

2.5 ARQUITECTURA SELECCIONADA

Das várias hipóteses, usando processadores da família 8051, aquela que se baseia nas capacidades do 8044 é sem dúvida a que melhores condições oferece. Em relação às outras soluções série, pode ter um pequeno senão, o preço do próprio 8044, que de qualquer modo tem descido consideravelmente.

As capacidades deste processador podem ser utilizadas para a implementação de uma rede usando o protocolo BITBUS ou um definido pelo utilizador. As limitações causadas ao projecto pela adesão ao protocolo BITBUS não são significativas, como será visto no capítulo 3. As vantagens da sua utilização são consideráveis já que permitem a integração da unidade desenvolvida em outros projectos e eventualmente o uso neste projecto de unidades disponíveis comercialmente.

O sistema de controlo fica assim constituído por uma rede BITBUS, implementada com microcontroladores 8044. O nó "master" da rede estabelece as ligações com o computador central, um IBM compatível. Este nó é constituído por uma placa a instalar numa das fichas de expansão do PC. Cada um dos nós "slave" é o controlador de um dos eixos. A ligação entre o nó "master" da rede e cada um dos "slave" é feita por meio de um bus série do tipo multiponto. Estes nós baseados no 8044 constituem processadores completos, dispondo de memória, entradas e saídas. Todo o controlo do eixo é confiado às capacidades do processador 8044.

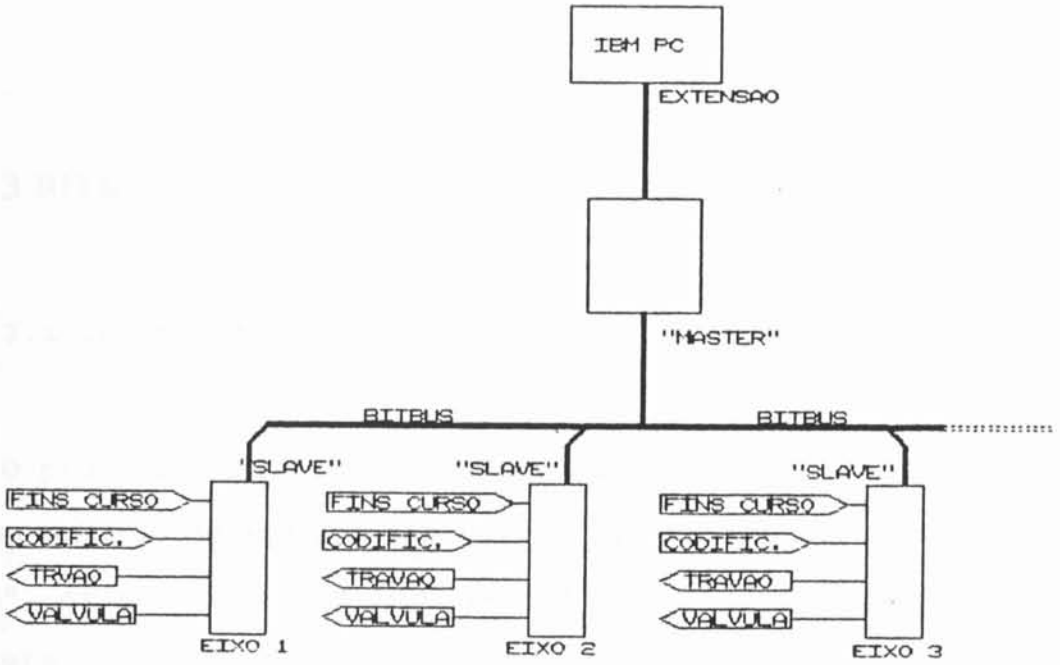


FIG. 2.2 ARQUITECTURA SELECCIONADA

A estrutura desenvolvida (Fig. 2.2) tem características modulares, permitindo a sua expansão. É igualmente possível a utilização nesta rede de módulos desenvolvidos por diversos fabricantes.

3 BITBUS

3.1 INTRODUÇÃO

O protocolo BITBUS define uma rede do tipo "master-slave", com capacidade de até 250 nós [8]. Sendo embora uma rede "master-slave", existem algumas implementações que permitem a troca de "masters". A arbitragem é neste caso garantida por uma linha "dasy-chain".

Esta rede é constituída por um processador, a desenpenhar as funções de controlador da rede, o nó "master" e até 250 nós "slave". Cada nó é construido, nas implementações mais correntes, à base de um processador da família 8044 e pode ter associado um outro processador designado por extensão do nó.

3.2 ESPECIFICAÇÕES

No protocolo BITBUS estão definidos alguns níveis de especificação [8].

- Ligação eléctrica
- Protocolo de transmissão de dados ("data link").
- Protocolo de mensagem.
- Comandos de alto nível (RAC).

3.2.1 LIGAÇÃO ELÉCTRICA

No nível de ligação eléctrica é seguido o standard RS485.

A transmissão pode ser feita em modo síncrono ou com o relógio codificado nos dados, o modo "self-clocked".

No modo síncrono são usados dois pares de fios torcidos um para o relógio de sincronismo e o outro para os dados. A frequência de relógio pode variar entre 0,5 e 2,4 Mhz. A codificação é feita de um modo muito simples, o valor a ser transmitido é posto em 0 ou 1 na descida do relógio e lido pelo receptor na subida. Neste modo o número de nós não pode ultrapassar os 28 e comprimento máximo do bus é de 30 metros.

No modo "self clocked" o relógio e os dados são codificados num mesmo sinal, usando o método NRZI. Para permitir um correcto sincronismo do hardware de descodificação do relógio, antes de cada trama, é transmitido um número mínimo de oito 0. Neste modo as velocidades de transmissão são de 62,5 ou 375 Kbits/s, podendo no primeiro caso o número de nós atingir os 250 e o comprimento do bus ultrapassar os 10 Km. O bus é constituído neste caso por um par de fios torcidos para a transmissão dos dados. Se o número de nós ultrapassar 30, há necessidade de utilizar repetidores e, para os controlar, é usado um segundo par de fios.

3.2.2 PROTOCOLO DE TRANSMISSÃO DE DADOS

O protocolo de transmissão de dados definido pelo BITBUS para a comunicação através da rede série é baseado no

"Syncrhonous Data Link Control" (SDLC) desenvolvido pela IBM.

A estação principal da rede, o "master", funciona com um subconjunto do protocolo SDLC designado "Assyncronous Balanced Mode" (ABM). É a estação principal que inicia e termina qualquer transmissão.

As estações secundárias, os nós "slave", após o reset ou a ocorrência de um erro irrecuperável ficam em "Normal Disconnect Mode" (NDM). É o nó "master", que através de uma mensagem de sincronismo, os muda para o estado "Normal Response Mode" (NRM).

As mensagens são transmitidas por uma trama com o seguinte formato.

Separador

A trama que constitui uma mensagem é sempre iniciada e terminada pela palavra de oito bits, 01111110. Este separador é único já que a SIU insere um 0 sempre que surge um conjunto de cinco 1 seguidos, excepto na transmissão do separador.

Endereço

Contém o endereço do nó destino, se a mensagem é de comando ou da origem nas mensagens de resposta. Isto é, o endereço não é alterado na mensagem de resposta.

Controlo

Este campo é usado para a troca de comandos e estado entre a estação principal e as secundárias. Define o tipo de mensagem (supervisão, sincronização e informação) e contém, nas mensagens em que tal é necessário, os números de sequência Nr e Ns para confirmação de recepção.

Informação

Este campo existe apenas nas mensagens de informação. Transporta a mensagem exactamente como é definida no protocolo de mensagem.

FSC

Este campo destina-se ao controlo de erros. É composto por 16 bits. O nó transmissor gera e transmite este campo que é testado pelo receptor.

No que respeita à comunicação com o computador associado a um nó, a extensão do nó, o BITBUS não estabelece qualquer protocolo de transmissão de dados.

3.2.3 PROTOCOLO DE MENSAGEM

O protocolo de mensagem em BITBUS é o definido pelo sistema

operativo IDCX51 (referido em 3.3.2), para a transmissão de mensagens entre tarefas.

Para a transmissão através da rede série, a mensagem é inserida no campo de informação da trama do BITBUS.

O protocolo de mensagem utilizado para a comunicação com o processador associado a um nó, a extensão do nó, é também o definido pelo IDCX51. Competirá a um programa especializado, dependente do hardware implementado, a transferência das mensagens entre o processador do nó e a extensão. No caso de ser usado o 8044 na implementação do nó, o programa de comunicação com a extensão está incluído na tarefa 0. Este programa usa para a comunicação uma porta paralelo de oito bits.

3.2.4 COMANDOS RAC

O protocolo BITBUS prevê um conjunto de comandos de alto nível que constituem, nesta rede, uma camada de aplicação.

A execução dos comandos (Remote Access and Control) RAC é desencadeada por mensagens. O protocolo de mensagem prevê para este fim uma palavra identificadora de comando no cabeçalho. Estes comandos permitem ao nó master, ou à sua extensão, executar acções em qualquer dos nós slave, sem que para isso seja necessário escrever código, no nó respectivo.

Os comandos RAC permitem:

- Fazer o reset de um nó
- Executar operações lógicas, ler ou escrever na área de memória externa reservada para entradas ou saídas.

- Criar ou apagar uma tarefa
- Ler ou escrever na memória interna ou externa.

3.3 SOFTWARE INSTALADO NO 8044

O modo mais simples de implementar a rede BITUS é com a utilização de processadores da família 8044.

O processador 8044 tem um programa já instalado nos primeiros 4 Kbytes da memória de programa, constituído por três blocos distintos [11]:

- Teste de hardware.
- IDCX51.
- Tarefa 0.

3.3.1 TESTE DE HARDWARE

O teste de hardware é executado cada vez que é feito o reset. Neste teste é feita a verificação do bom funcionamento do processador e da memória, interna e externa. Qualquer erro implica a paragem do teste e a mudança da maior parte das linhas de saída para alta impedância. Este teste evita que um processador, em más condições, possa provocar contenção do bus. O facto de a maioria das linhas do processador ficar em alta impedância, cria sérias dificuldades na caracterização da avaria.

3.3.2 IDCX51

O IDCX51 [12] é um sistema operativo elementar do tipo

multitarefa, está instalado nos processadores 8044 e constitui o ambiente de trabalho típico do BITBUS.

3.3.2.1 TAREFAS

O IDCX51 permite até oito tarefas simultâneas, sendo cada uma, um programa que funciona em ciclo como se fosse o único a usar o processador.

O sistema operativo funciona de modo a que cada tarefa disponha de:

- Uma área reservada de stack
- Um conjunto de registos
- Acumulador
- Registo de flags
- Ponteiro da stack
- Registo B
- Ponteiro de dados
- Um banco de registos
- Contador de programa

3.3.2.2 DESCRITORES DE TAREFA

Cada tarefa é descrita por um descritor, que contém informação sobre:

- Início do código da tarefa.
- Espaço de memória reservada para a stack privativa da tarefa.
- Identificador da tarefa.

- Banco de registos reservado.
- Prioridade (de 1 a 4).
- Interrupções atribuídas à tarefa.
- Ponteiro para o próximo descritor.

Os descritores de tarefa são organizados numa lista ligada que o sistema operativo percorre na operação de arranque. O 8044 tem já instalada a primeira tarefa, a tarefa 0. O ponteiro no descritor desta tarefa tem o valor FFF0H, o que permite ao utilizador instalar uma lista de até sete tarefas, com início nesse endereço.

3.3.2.3 IDD

A lista ligada inclui ainda um descritor especial designado por IDD que contém os seguintes parâmetros do sistema:

- Resolução do relógio.
- Prioridade do relógio.
- Comprimento do buffer de comunicações.
- Area de memória interna reservada pelo utilizador.

No 8044 existe já um IDD, no entanto se na lista ligada de descritores for incluído um novo, este último sobrepõe-se ao já existente, permitindo assim a reconfiguração do sistema.

3.3.2.4 RELÓGIO

O sistema operativo é regido em termos de tempo pelo "timer" 0 do processador. Este "timer" é carregado com um valor escolhido pelo utilizador, especificado no IDD, definindo assim a resolução

do relógio de sistema. Com base neste relógio, o sistema operativo fornece dois temporizadores por tarefa, que podem ser usados pelos programas do utilizador.

3.3.2.5 ESTADO DAS TAREFAS

Cada tarefa pode estar num de três estados :

- "Running" a tarefa em curso (pode não existir nenhuma).
- "Ready" as tarefas neste estado estão organizadas numa lista ligada, ordenada de acordo com a sua prioridade.
- "Asleep" estão neste estado todas as tarefa que, por invocação da primitiva do sistema operativo "Rqwait", estão a aguardar um acontecimento.

Uma tarefa passa do estado de running ao "asleep" pela execução de uma chamada ao sistema operativo (Rqwait), ficando à espera de um dos seguintes acontecimentos:

- Chegada de uma mensagem.
- Activação de uma das interrupções associadas à tarefa.
- Completar do tempo de um dos temporizadores associados à tarefa.

Cada um destes acontecimentos, provoca a passagem da tarefa do estado de "asleep" ao estado de "ready" ou "running". O estado para que passa a tarefa depende da sua prioridade ser menor ou

maior que a "running".

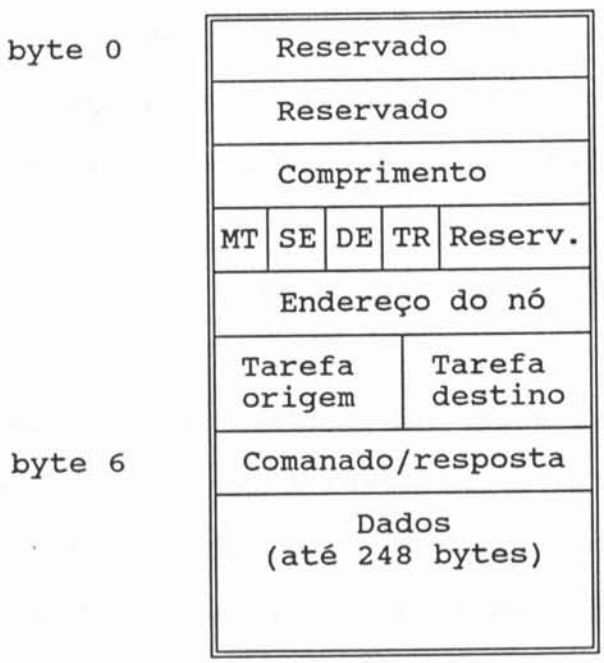
A tarefa "running" pode, por recurso a funções do sistema operativo, criar ou apagar uma tarefa. A tarefa criada fica no estado de "ready" ou "running" de acordo com a sua prioridade.

Sempre que a tarefa "running" seja forçada a ceder o lugar a outra, passa a ocupar o topo da lista das tarefas "ready".

3.3.2.6 MENSAGENS ENTRE TAREFAS

O IDCX51 possui um sistema de comunicações entre tarefas sob a forma de mensagens. As mensagens são do tipo, comando ou resposta. A comunicação é iniciada por uma qualquer tarefa, com uma mensagem de comando. A tarefa destino deverá estar no estado "asleep" a aguardar a mensagem. Compete à tarefa destino a resposta, com uma mensagem do tipo resposta.

As mensagens entre tarefas têm o seguinte protocolo:



Reservado

Espaço de dois bytes reservado pelo sistema operativo. Deve ser mantido a 0.

Comprimento

Comprimento da mensagem em bytes, incluindo o cabeçalho.

MT (1 bit)

Especifica se a mensagem é de comando (MT = 0) ou resposta (MT = 1).

SE (1 bit)

Nas mensagens comando SE = 1 define a mensagem como proveniente da extensão de um nó. Em mensagens resposta SE = 1 define o endereço da mensagem como sendo a extensão de um nó (i.é este bit não é alterado).

DE (1 bit)

Nas mensagens comando DE = 1 define a mensagem como destinada à extensão de um nó. Nas mensagens resposta DE = 1 indica que a resposta é proveniente da extensão de um nó (i.é este bit não é alterado).

TR (1 bit)

Não é utilizado devendo ser mantido em 0

Endereço

Endereço do nó destino nas mensagens de comando e origem nas mensagens resposta. O byte de endereço só tem justificação nas aplicações com múltiplos processadores, o caso da rede BITBUS.

Tarefa origem (4 bits)

Identifica a tarefa origem de uma mensagem de comando ou o destino numa mensagem resposta.

Tarefa destino (4 bits)

Identifica a tarefa destino nas mensagens de comando e origem nas de resposta.

Comando

Na mensagem de comando contém uma palavra identificadora do comando RAC. Na resposta contém indicações de erro.

As únicas diferenças entre o cabeçalho de uma mensagem de comando e a sua resposta são portanto, a palavra de comando e o bit MT.

Para enviar uma mensagem de comando, a tarefa terá de fazer, pelo recurso a uma função do sistema operativo, a reserva de um espaço de memória e preenchê-la com a mensagem a enviar (incluindo o cabeçalho). A mensagem de resposta ocupa o mesmo espaço de memória.

É corrente nos processadores a operarem na rede BITBUS o uso do sistema operativo IDCX51. O facto de o protocolo de mensagem ser o mesmo para as transmissões através de rede e entre tarefas do mesmo nó, torna a transmissão através da rede transparente. É o sistema operativo que ao verificar que o endereço, no cabeçalho da mensagem, não corresponde ao do próprio nó a dirige à tarefa 0. Esta tarefa encarrega-se da transmissão da mensagem através da rede série, recorrendo às capacidades da SIU.

As mensagens, a serem transmitidas através da rede serie, são obrigatoriamente criadas em memória interna. Este facto tem

implicações no comprimento da mensagem. Dada a exiguidade da memória interna e a conveniência de dispormos de mais que um "buffer" de comunicações a INTEL define o comprimento como sendo 20 bytes.

3.3.2.7 USO DA MEMÓRIA INTERNA

Nos processadores da família 8044 a memória interna de dados é escassa, 192 bytes. Esta memória é particularmente interessante para o programador já que os modos de endereçamento permitidos são vários e conduzem a tempos de acesso muito curtos (1 microsegundo).

A memória interna do 8044 é constituída por três grandes áreas:

- Espaço com endereços de 0 a 127, pode ser endereçada directa ou indirectamente. Contém quatro bancos de registos nos endereços de 0 a 31 e um espaço endereçável ao bit ocupando os endereços de 32 a 47. A restante é de uso geral.
- Espaço com endereços de 128 a 255, área de registos de uso especial, é uma área de memória endereçável apenas directamente. Não é possível outra utilização desta área de memória, além do uso dos registos especificados pelo fabricante (SFR).
- Espaço com endereços de 128 a 191, um espaço de memória de uso geral, apenas endereçável indirectamente.

A aparente sobreposição de endereços é resolvida pelos diferentes modos de endereçagem.

O IDCX51 reserva, na parte alta da memória interna (0 a 192), o espaço para a stack privativa de cada tarefa especificado no seu descritor. Na parte mais baixa, a seguir ao espaço endereçável ao bit (47), situa-se a área reservada pelo utilizador. A memória restante é dividida em buffers, com o comprimento especificado no IDD (divisão inteira), que são organizados numa lista ligada. A memória restante da divisão fica disponível para o utilizador.

A tarefa corrente tem a possibilidade de em qualquer instante, por recurso a primitivas do sistema, requerer mais memória ou libertar aquela de que já não necessite. O sistema operativo recalcula sempre o número de bufferes de comunicação disponíveis.

3.3.2.8 USO DA MEMÓRIA DE DADOS EXTERNA

A memória externa é obrigatoriamente mapeada a partir do endereço 0. O sistema operativo ocupa os primeiros 256 bytes, estando os restantes disponíveis para o utilizador.

3.3.2.9 INTERRUPÇÕES

As interrupções de que dispõe o processador são atribuídas a uma dada tarefa de acordo com o definido no seu descritor. A atribuição de uma interrupção a duas tarefas implica que apenas a referida em último lugar dispõe da interrupção.

A interrupção associada ao "timer 0" é utilizada pelo sistema operativo para a gestão de tempos, não podendo ser atribuída a uma tarefa.

A gestão de prioridades das interrupções pertence ao sistema operativo. São sempre definidas com baixa prioridade as interrupções associadas a uma tarefa de mais baixa prioridade que a "running" e com alta prioridade as outras. Este esquema permite de facto escalonar as prioridades em quatro níveis.

A alteração das prioridades das interrupções pelo programa do utilizador pode conduzir a mau funcionamento. O processo de gerir, de um modo seguro, as interrupções é a sua inibição e desinibição em bloco. A inibição global das interrupções, nos nós "slave", só pode ser feita por períodos curtos (<10ms), de outro modo a estação principal considera o nó em falta e passa a ignorá-lo.

Quando o processador é interrompido, a tarefa detentora da interrupção, se está em "wait", subtitui a tarefa "running" que passa a "ready".

Sempre que uma tarefa faz uso de funções do sistema operativo, este desactiva todas as interrupções durante a execução da função.

Usando o 8044 na construção da rede, as interrupções IE0 e IE1 são, à partida, atribuídas à tarefa 0. Se num dado nó não existe a porta paralelo, estas interrupções podem ser atribuídas a qualquer outra tarefa.

3.3.3 TAREFA 0

O 8044 tem já instalada a tarefa 0. Esta tarefa tem por função a gestão das comunicações externas ao nó, através da rede BITBUS ou da porta paralelo e a descodificação e execução de comandos designados por comandos RAC. Tem a prioridade 4 e usa o banco de registos 0.

4 HARDWARE

4.1 INTRODUÇÃO

No projecto foi adoptada uma estrutura modular para permitir a sua fácil adaptação às características específicas de possíveis aplicações diferentes.

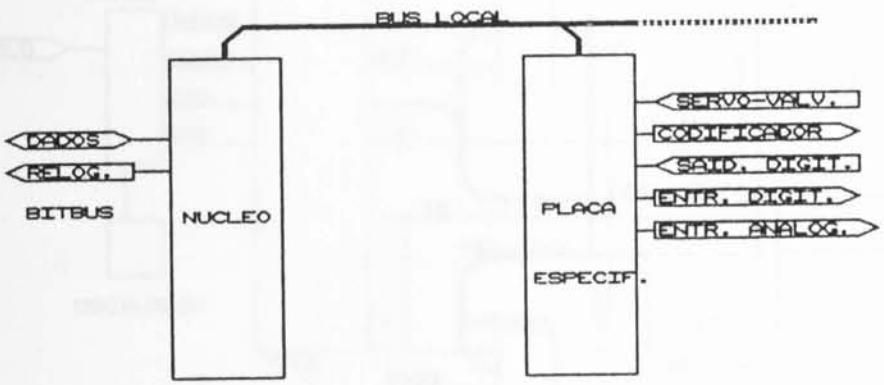


FIG. 4.1 DIAGRAMA DE BLOCOS DO EIXO

A base do projecto é um placa de 100X160 mm, designada por núcleo. Esta placa constitui por si um nó BITBUS completo com, processador, memórias, canal de comunicação série e portas de configuração. Para a sua ligação às placas de aplicação foi previsto um bus local paralelo, com capacidade de endereçamento

de 32 bytes.

O nó fica completo com uma placa específica interligada com núcleo através do bus local, na qual reside o hardware particular da aplicação. A placa usada neste projecto contém, contador de posição, entradas analógicas, entradas e saídas digitais. O bus local permite a aplicação de várias placas.

4.2 NÚCLEO

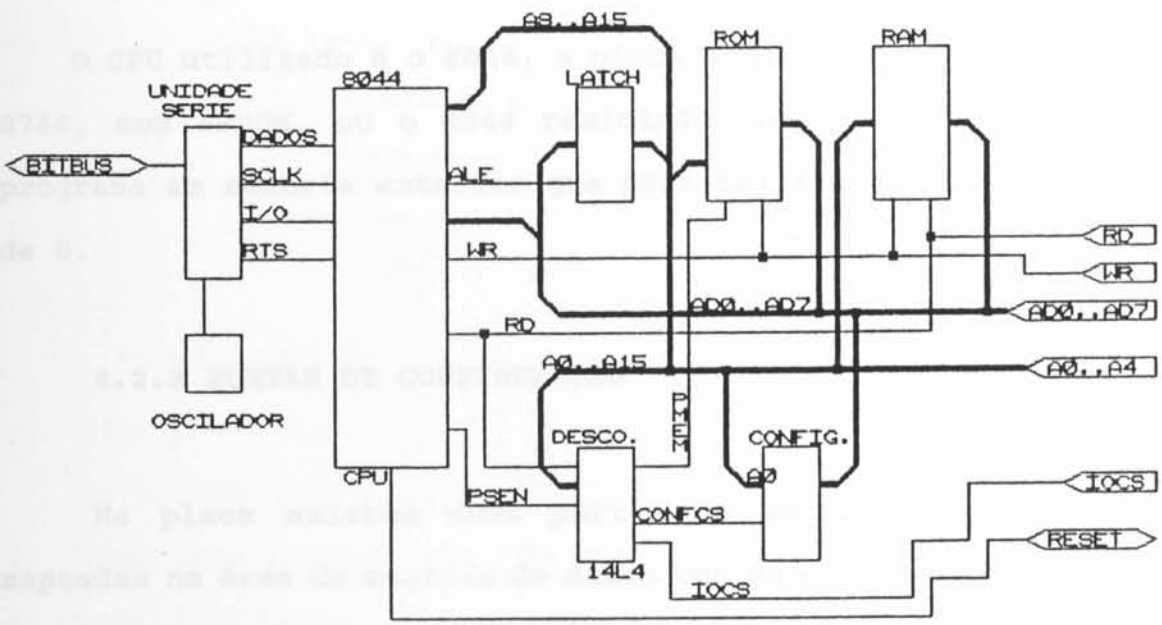


FIG. 4.2 DIAGRAMA DE BLOCOS DO NÚCLEO

O cartão que constitui o núcleo contém (Fig. 4.2):

- O processador 8044.
- Portas de configuração e endereço.

- Espaço para memória de dados, até 32 Kbytes.
- Espaço para memória de programa, até 32 Kbytes.
- Unidade de comunicação com a rede BITBUS.
- Unidade de descodificação.
- Circuito de reset.
- Saídas de teste.
- Bus paralelo

Esta placa foi projectada de modo a manter a compatibilidade com a rede BITBUS, pode ser configurada como master ou slave.

4.2.1 CPU

O CPU utilizado é o 8044, a placa admite no entanto o uso do 8744, com EPROM, ou o 8344 residindo, nesta hipótese, todo o programa em memória exterior que para tal é endereçada a partir de 0.

4.2.2 PORTAS DE CONFIGURAÇÃO

Na placa existem duas portas de entrada de oito bits, mapeadas na área de memória de dados com endereços FFFEH e FFFFH, destinadas à configuração do sistema.

Em qualquer das entradas de configuração a ausência do shunt é lida como 0.

Os shunts de 30 a 37 seleccionam o endereço do nó. Os nós "slave", que constituem cada um dos eixos, podem ter endereços de 1 a 250.

Os shunts de 20 a 27 têm a seguinte utilização:

- J20 e J21 seleccionam o modo de funcionamento com a seguinte configuração.

  00 modo síncrono

  10 "self-clocked" a 375 Kbit/s

  11 "self-clocked" a 62.5 Kbit/s

- J22 reservado
- J23 configuração da memória de programa.

  0 usa memória interna nos endereços 0 a 1000H

  1 o programa é lido em memória externa qualquer que seja o endereço.

- J24 selecciona o modo de usar a memória de programa.
- J25 reservado
- J26 Utilização da porta paralelo de comunicação com a extensão
- J27 reservado

4.2.3 MEMÓRIAS

No cartão principal foram aplicados dois suportes JEDEC de 28 pinos para as memórias. Um destes suportes destina-se ao uso de memória de dados (RAM) com capacidades 2 a 32 Kbytes, o outro, destinado à memória de programa, admite o uso de memórias de 8 a 32 Kbytes que podem ser dos tipos, RAM, PROM, UV-EPROM ou EEPROM.

A memória de dados está mapeada no espaço de endereços 0 a 7FFFH. A descodificação é incompleta para memórias de capacidade inferior a 32 Kbytes.

No caso de ser usado o processador 8044, o sistema operativo utiliza o espaço de 0 a 00FFH, sendo todo o restante disponível para o utilizador. O espaço de endereçamento entre 8000H a FFFFH nunca é usado para memória de dados.

Existem na placa dois shunts de três posições cada um, que permitem a configuração do suporte da memória de dados de modo a permitir a instalação memórias RAM de 2 a 32 Kbytes.

Na placa estão previstos dois modos de utilizar o espaço de memória de programa do processador, configuráveis pelo bit 4 da porta de configuração designado MMCFG (J24).

Se MMCFG = 1 (shunt inserido), está disponível para memória de código todo o espaço de 0 a FFFFH. A descodificação é incompleta para memórias de capacidade inferior a 64 Kbytes. Deste espaço se o bit 3 da porta de configuração EA = 0, o espaço de memória de código de 0 a 1000H é ocupado por memória interna e o processador só acede a memória exterior para endereços superiores.

Se MMCFG = 0, shunt retirado, estão disponíveis para memória de programa os espaços:

- 0 a 1000H, ocupado ou não por memória interna de acordo com o shunt J23.
- 8000H a FEFFH disponível para o utilizador.
- FFF0H a FFFDH usado para o descritor da tarefa 1.

Nos espaços de endereçagem 8000H a FEFFH e FFF0H a FFFDH, existe apenas memória de programa. A descodificação é feita de modo que, com estes endereços, a memória seja seleccionada sempre que seja activada a linha de Wr. É possível, deste modo, escrever neste espaço de memória, usando as instruções de escrita em memória de

dados. Um programa residente no processador pode assim alterar a memória de programa, o que não é corrente nos processadores da família 8051.

Os comandos RAC permitem inclusivé a carga de programas num qualquer nó da rede a partir do computador central.

4.2.4 UNIDADE DE COMUNICAÇÃO SÉRIE

A comunicação série no BITBUS usa no nivel físico o standard RS 485. É usado um par de fios para dados DADO e *DADO e um segundo par, dependente da configuração, para relógio ou controlo de possíveis repetidores.

Neste projecto a conversão dos sinais de níveis TTL do processador para as características específicas do standard RS 485 está confiada ao par de circuitos 75174 e 75175 .

Foram instaladas duas fichas DB9, uma macho e outra fêmea, de modo a facilitar a interligação dos vários nós sem quaisquer elementos de derivação.

Para evitar reflexões nas linhas de transmissão é aconselhável o uso de resistências de carga nos extremos. Nas placas não foi previsto qualquer espaço para a sua instalação. As placas que se situem nos extremos do bus ficam cada uma com uma ficha DB9 disponível, onde devem ser instaladas essas resistências.

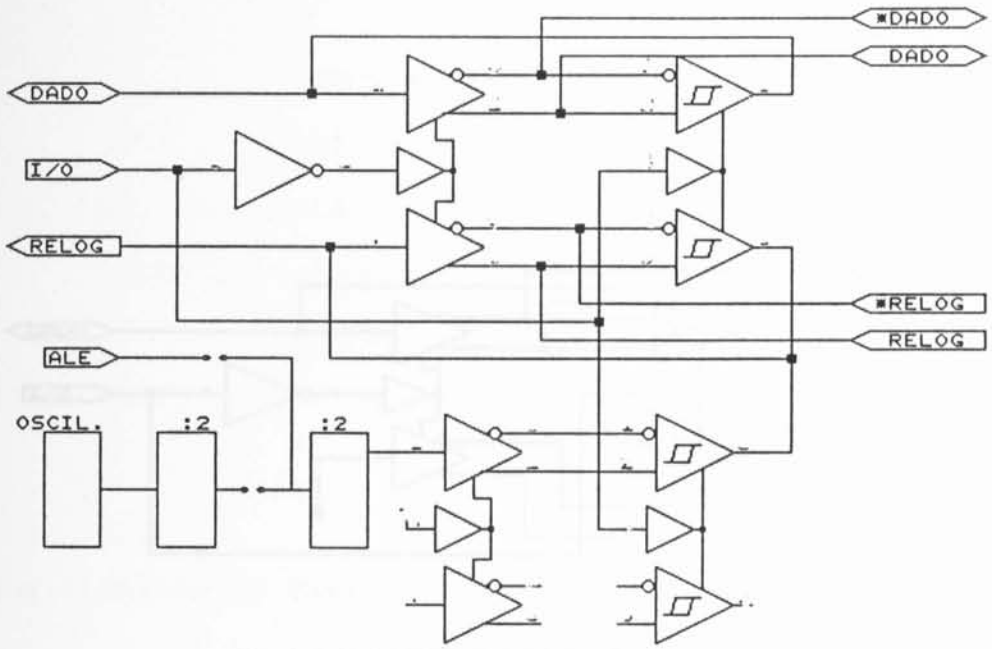


FIG. 4.3 COMUNICAÇÃO EM MODO SINCRONO

É possivel a configuração da unidade de comunicação série para funcionar em modo sincrono ou "self clocked".

Em modo síncrono (Fig 4.3) o hardware exterior fornece ao processador um sinal de relógio que pode ser derivado do ALE depois de dividido por 2, e assim permitir transmissões com cadências próximas de 1 Mbit/s ou através de um oscilador, a instalar na placa, cadências de 0,5 a 2,4 Mbit/s.

No modo "self clocked" a codificação é feita em NRZI pela SIU. As cadências de transmissão são 62,5 Kbits/s ou 375 Kbits/s, de acordo com estado dos bit 0 e 1 (J20 J21) da porta de configuração, supondo a frequência de relógio do processador igual a 12 Mhz.

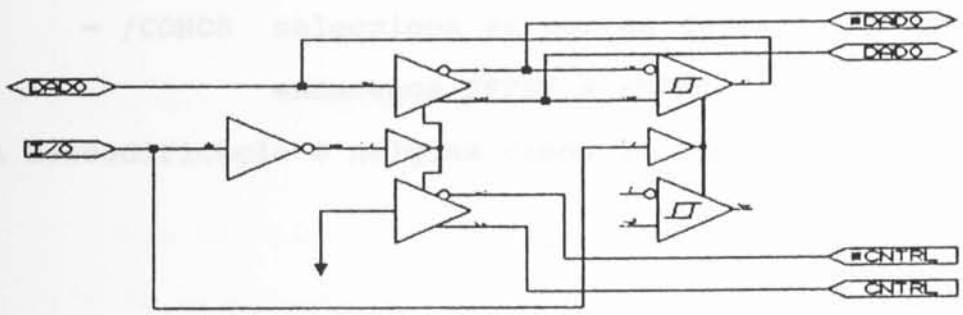


FIG. 4.4 COMUNICAÇÃO EM MODO "SELF CLOCKED"

4.2.5 RESET

A linha de reset presente no bus paralelo é do tipo colector aberto. Esta solução permite que a placa específica da aplicação possa fazer o reset de todo o sistema.

4.2.6 DESCODIFICAÇÃO

A descodificação é feita com recurso ao uso da PAL 14L4, tendo as saídas:

- /IOCS linha de selecção da área disponível como entrada/saída. Esta linha é activa (nível lógico 0) no espaço de endereçamento FF00H a FF10H
- /ROMS selecciona a memória de programa. Se CMEM = 1 então /ROMCS = PSEN. Se CMEM = 0 então

/ROMCS é activa nos espaços de endereçamento 8000H a FF00H e de FFF0H a FFFDH, apenas se estão activas as linhas de Wr ou PSEN.

- /CONCS selecciona as portas de configuração nos endereços FFFEH e FFFFH.

A descodificação é nalguns casos incompleta.

4.2.7 SAÍDAS DE TESTE

A placa está dotada de dois leds ligados às portas do processador P10 (verde) e P11 (vermelho) (endereços 90 e 91). Estes sinalizadores são usados pelo sistema para informação na fase de teste. Terminada esta fase, podem ser usados por programas do utilizador.

A sequência de testes é a seguinte:

- Teste do conjunto de instruções do processador. O bom resultado deste teste é indicado pelos dois leds apagados.
- Teste da ROM interna, o seu sucesso é indicado com o acender do led vermelho
- Teste da RAM interna, o seu sucesso é indicado pelo acender de ambos os leds.
- teste da RAM externa, o seu sucesso é indicado pelo acender do led verde, ficando o vermelho apagado.

Se em qualquer das fases de teste for detectado algum erro o teste é interrompido e o processador fica bloqueado com as

saídas em alta impedância.

4.2.8 BUS LOCAL

Na placa está instalada uma ficha de vinte e seis pinos, contendo:

- Bus de dados
- As linhas de A0 a A4 do bus de endereços
- Reset
- Duas linhas de interrupção IE0 e IE1
- Linhas de alimentação 0 e 5 volts.
- /IOCS da PAL
- RD
- WR
- ALE
- P12

Neste projecto houve a preocupação de evitar que erros na placa específica de aplicação possam danificar o processador. No bus de dados e nas linhas de RD, WR e ALE foram aplicados buffers, de modo a garantir um "fan-out" 10, considerando níveis TTL. Nas linhas de endereço e saídas da PAL não houve a mesma preocupação, já que qualquer delas é do tipo TTL, com pouca utilização na placa de núcleo.

Este conjunto de linhas constitui um bus local elementar, com as características estáticas e dinâmicas típicas do processador 8051.

4.3 PLACA ESPECÍFICA

Uma segunda placa, com formato 100X160 mm, contém o hardware específico da aplicação. Pode ser ligada ao núcleo, directamente por ficha ou por cabo paralelo de 26 condutores.

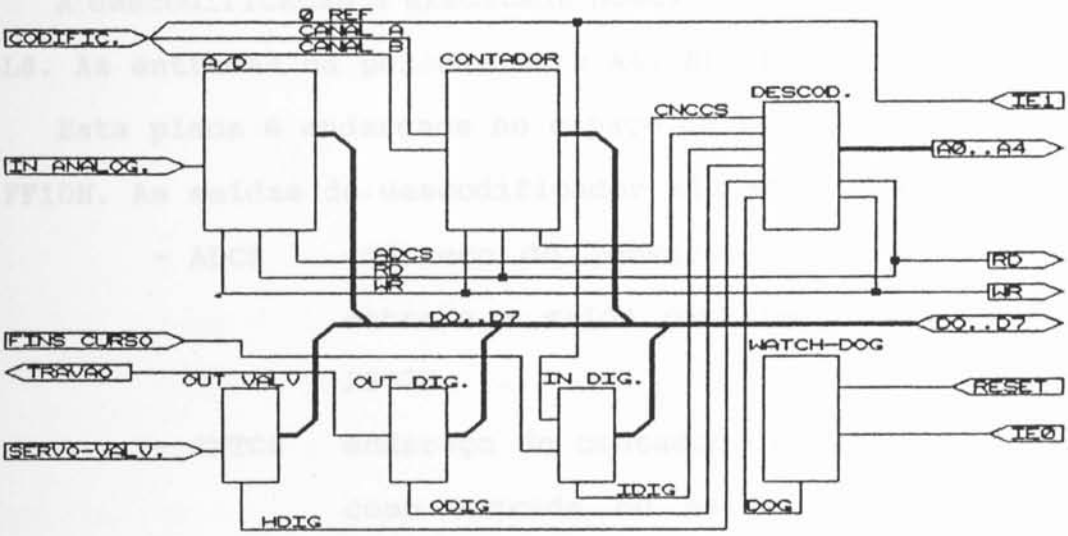


FIG. 4.6 DIAGRAMA DE BLOCOS DA PLACA ESPECÍFICA

Esta placa contém:

- Descodificação.
- Dois canais de entrada para a ligação do codificador de posição.
- Um canal de entrada para a referência de zero do codificador de posição.

- Contador de posição.
- Conversor A/D com dois canais de entrada.
- "Watch-dog".
- Saída digital de 8 bits para a válvula hidráulica.
- Saída digital de 4 bits.
- Entrada digital de 4 bits.

4.3.1 DESCODIFICAÇÃO

A descodificação é executada nesta placa com o recurso à pal 16L8. As entradas da pal são A0 a A4, RD, WR, e IOCS.

Esta placa é enderçada no espaço de memória de dados FF00H a FF10H. As saídas do descodificador são as seguintes:

- ADCS endereço do conversor A/D, é activo como entrada e saída para os endereços FF04H a FF07H.
- CNTCS endereço do contador de posição, é activo como entrada ou saída para os endereços FF08H a FF0BH.
- DOG porta de escrita para actuação sobre o watch-dog com endereço FF0CH.
- PRST porta de saída para contolo do contador de posição, é activa com endereço FF03H.
- IDIG porta digital de entrada de dados (4 bit), é activa com endereço FF02H.
- ODIG porta de saída digital, (4 bit) é activa com endereço FF01H.
- -HDIG porta de saída para a servo-válvula, é

activa com endereço FF00H.

4.3.2 CONTADOR DE POSIÇÃO

Por razões económicas os codificadores seleccionados para o projecto são do tipo incremental, com dois canais de saída desfasados 90 graus.

O sistema de detecção de sentido e contagem foi desenvolvido com base no circuito THCT2000. Trata-se de um integrado contendo:

- Um contador de 16 bit.
- Duas latch de 8 bit.
- Um circuito de filtragem digital dos sinais provenientes do codificador.
- Um circuito de descriminação de sentido.
- Porta paralelo de comunicação com um bus.

O circuito necessita de um sinal de relógio para o seu controlo, com frequência pelo menos quatro vezes superior à máxima frequência do sinal proveniente do codificador.

A comunicação entre o processador e este circuito é feita por um bus de oito bits. O processador pode ler e escrever no contador como sendo duas portas paralelo.

A leitura do valor actual do contador tem de ser feita como uma palavra de dezasseis bits. O circuito dispõe de duas latch que permitem a memorização global do contador, para posterior leitura. Para permitir a memorização e leitura das latch em tempo compatível com o 8044 é necessário que o relógio de controlo do contador tenha uma frequência superior a 4 Mhz.

Na placa o sinal de relógio disponível é de 1 MHZ, por isso foi utilizado um método alternativo para a leitura do contador. O TCHT2000 dispõe de uma linha do tipo colector aberto que forçada a 0 provoca a memorização do valor global do contador. Esta linha é activada antes de cada leitura.

A placa dispõe de três shunts que permitem configurar o TCHT2000 de modo a que a contagem seja feita, no degrau ascendente de um dos canais de entrada, no ascendente e no descendente de um só canal, ou ascendente e no descendente de ambos os canais. É assim possível configurar o sistema para funcionar com resolução 1, 2 ou 4 vezes a resolução base do codificador.

4.3.3 ENTRADAS DOS CODIFICADORES

Os sinais provenientes do codificador são ligados a esta placa através de acopladores ópticos. É possivel assim o uso de codificadores com saídas em tensão de 5 a 12 volts ou colector aberto. Os sinais do optoacoplador são filtrados por meio de um "schmitt-trigger" e sincronizados com o relógio de controlo do THCT200.

Foi prevista uma entrada para a referência de zero,que alguns codificadores possuem, esta entrada tem isolamento óptico e filtragem igual às restantes provenientes do codificador. A referência de zero pode ser lida pelo processador através de uma porta ou activar uma interrupção.

4.3.4 CONVERSOR A/D

Na placa foram ainda criadas entradas analógicas, destinadas à leitura de um taquímetro e de um sensor de pressão. Para a implementação destas entradas foi adoptado o conversor NEC 7002, trata-se de um conversor de oito ou dez bits com tempo de conversão 5 milisegundos (a dez bits). O conversor tem quatro canais analógicos dos quais apenas dois são usados.

Os sinais de entrada são condicionados por intermédio de amplificadores operacionais que igualmente fazem uma filtragem elementar do tipo passa baixo.

A entrada está prevista para tensões de 0 a 5 volts, ou com a aplicação de uma resistência na placa, para a entrada em fonte de corrente de 0 a 20 miliamperes. Para não aumentar o número de componentes, a gama de 4 a 20 miliamperes será obtida, a partir da de 0 a 20 miliamperes, por correcção por software.

4.3.5 WATCH-DOG

Foi implementado nesta placa um circuito de watch-dog constituído por um oscilador e contador binário. O oscilador, de 1 Mhz, serve ainda para o controlo do contador de posição e para o conversor A/D.

O microcontrolador deverá fazer o reset do "watch-dog" com intervalos de tempo regulares. O "watch_dog" tem dois níveis de actuação. Numa primeira fase com tempo limite ajustável entre 100

e 400 milisegundos é actuada a interrupção IE0. No caso de não ser feito entretanto o reset do "watch-dog" este entrará numa segunda fase com tempo limite regulável entre 800 milisegundos e 3,2 segundos, findo o qual é feito o reset do processador.

4.3.6 ENTRADAS DIGITAIS

A placa dispõe de quatro entradas digitais com isolamento óptico.

4.3.7 SAÍDAS DIGITAIS

Foram previstas quatro saídas digitais, com isolamento óptico.

4.3.8 SAÍDA PARA A SERVO-VÁLVULA

Para a ligação da servo-válvula foi criada uma saída digital de oito bits. As saídas têm isolamento óptico.

5 SOFTWARE

5.1 INTRODUÇÃO

De acordo com a arquitectura definida no capítulo 2 o sistema de controlo do robot é constituído por uma rede de microcontroladores. A rede é do tipo "master-slave", tendo associado ao nó "master" um computador do tipo IBM compatível.

Ao computador associado ao nó "master" compete a definição do próximo ponto a ser atingido por cada eixo e o tempo disponível para o fazer. A comunicação deste computador com os dos eixos é feita por meio de mensagens através da rede BITBUS.

Ao processador de cada eixo foram atribuídas as seguintes tarefas:

- Execução dos comandos enviados pelo processador central.
- Resposta com informação sobre o seu estado actual.
- Leitura dos sensores associados ao eixo.
- Cálculo dos parâmetros a fornecer à válvula hidráulica.
- Controlo dos actuadores locais.

5.2 COMANDOS

Para esta rede foi desenvolvido um conjunto de comandos que constituem uma aplicação específica. Estes comandos são enviados pelo processador central a cada eixo que os interpreta e executa. A cada comando, o processador do eixo responde com uma mensagem, contendo informação sobre a execução do comando e o seu estado actual.

Os comandos criados são os seguintes:

- **Leitura**

  o processador de eixo não executa qualquer acção com este comando, destina-se a permitir ao computador central interrogar o eixo.

- **Posição**

  este comando tem como parâmetros, próxima posição e velocidade. O processador do eixo executará as acções necessárias para ser feito o avanço para a próxima posição. A velocidade é neste comando interpretada como um limite, em módulo.

- **Paragem**

  este comando força a paragem do eixo.

- **Avanço**

  este comando tem como parâmetro a velocidade, provoca o avanço do eixo à velocidade especificada, até novo comando. O sentido do avanço é determinado pelo sinal da velocidade.

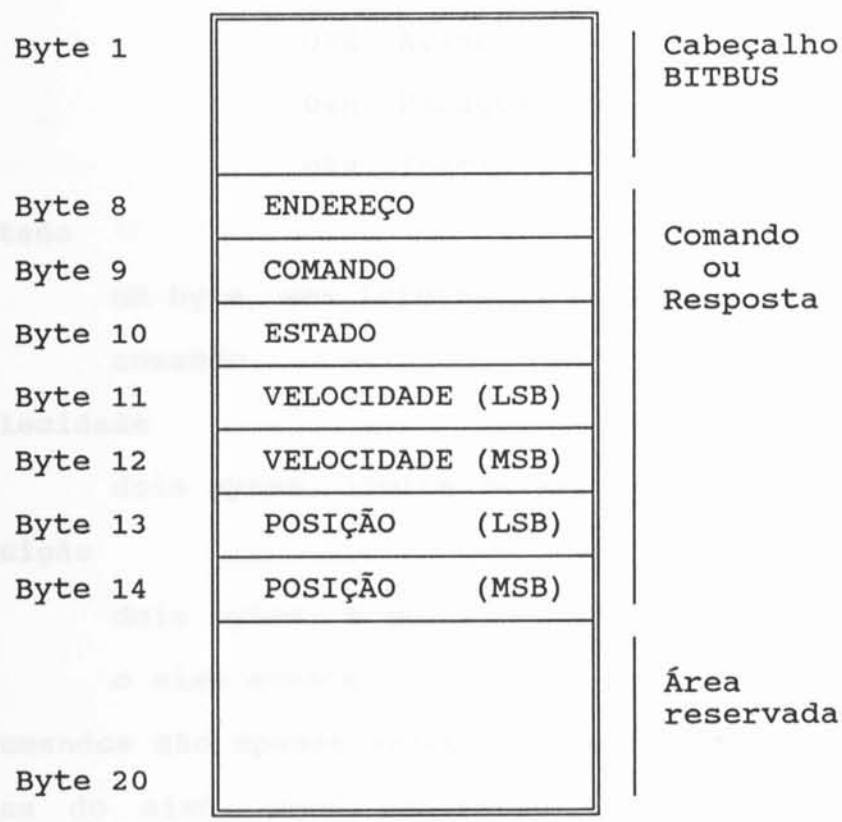
- **Início**

  trata-se de um comando, sem parâmetros, que movimenta o eixo para a sua posição inicial. Durante a execução deste comando o valor inicial dos contadores é estabelecido de acordo com a referência existente no codificador.

Os comandos são enviados pelo processador central como mensagens, através da rede, dirigidos ao processador do eixo.

A mensagem de comando segue protocolo de mensagens do BITBUS. Toda a informação referente ao comando está contida no espaço de dados. Deste modo não são afectadas as capacidades da rede BITBUS, nomeadamente os comandos RAC. .

A mensagem fica asim constituida:



- **Cabeçalho**

  composto por sete bytes de acordo com o protocolo BITBUS. Na palavra de comando do cabeçalho da mensagem é escrito 80H de modo a não ser interpretado pela tarefa 0

- **Endereço**

  um byte contém o endereço do eixo a que se destina o comando. Os valores possíveis são, de acordo com o definido no BITBUS, de 1 a 250.

- **Comando**

  um byte com o comando de acordo com o já referido. Os comandos criados são:

  01H Leitura

  02H Posição

  03H Avanço

  04H Paragem

  05H Início

- **Estado**

  um byte, sem informação relevante na mensagem de comando.

- **Velocidade**

  dois bytes, limite de velocidade.

- **Posição**

  dois bytes, a posição para onde se pretende que o eixo avance.

Os comandos são apenas executados se não colidirem com as capacidades do eixo, caso contrário é gerada a respectiva

mensagem de erro.

Velocidade e posição são números binários de dezasseis bits em complemento para 2. O byte menos sigificativo vem em primeiro lugar. A unidade em que é expressa a posição é o passo da régua, neste caso 0,005 mm. A unidade em que é expressa a velocidade é passos da régua por período de amostragem, sendo o período de amostragem 10 milisegundos.

O processador de eixo recebe a mensagem, interpreta o comando e gera uma mensagem resposta.

A mensagem resposta tem uma estrutura semelhante à de comando.

- **Cabeçalho**

  sete bytes, de acordo com protocolo de mensagens próprio do BITBUS.

- **Endereço**

  um byte, retorna o endereço do nó apenas para confirmação.

- **Comando**

  um byte, é aproveitado na resposta para transmitir informação sobre possíveis erros na execução do comando.

  00H Comando interpretado e executado sem erro

  01H Comando desconhecido

  02H Posicão pedida excede os limites do eixo.

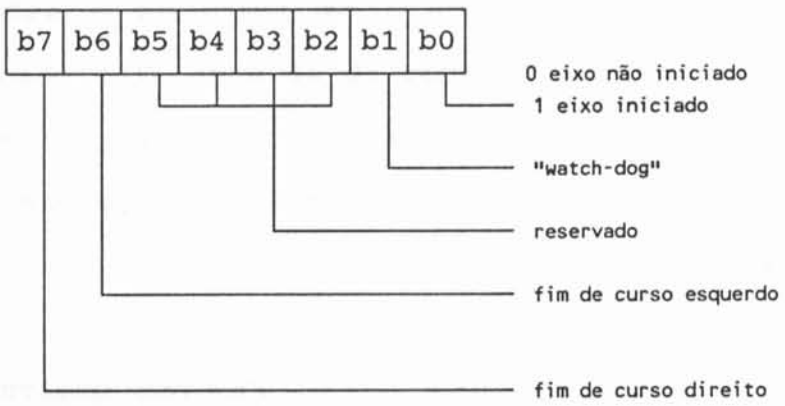
  04H Velocidade pedida excede as capacidades do eixo.

08H Comando não executado por estar activado o fim de curso que impede o avanço no sentido pedido.

10H Eixo ainda não inicializado. Esta mensagem é gerada sempre que seja dada ordem de avanço ou posição a um eixo ainda não inicializado depois de um reset

- Estado

um byte, com informação sobre o estado actual do eixo. O conteúdo deste byte tem o seguinte significado.



Nos fins de curso, 0 significa que não foi atingido.

- Velocidade

dois bytes, contém informação sobre a velocidade actual do eixo.

- Posição

dois bytes, contém informação sobre a posição actual do eixo.

As mensagens têm um comprimento útil sempre igual a 14 bytes. Nesta aplicação a definição do comprimento no IDD foi mantida em 20 bytes de acordo com o usual no BITBUS, deixando assim um espaço disponível de 6 bytes para possível expansão do campo de parâmetros.

5.3 PROGRAMAS DESENVOLVIDOS

Para esta aplicação foram desenvolvidos dois programas distintos.

Um, instalado no computador central, destina-se à interligação entre o programa de cálculo da trajectória e o processador de cada eixo através da rede BITBUS.

O outro a instalar em cada eixo tem por função a interpretação e execução dos comandos.

5.3.1 COMPUTADOR CENTRAL

Para a comunicação entre o programa de cálculo de trajectórias e o programa de controlo de cada eixo foi escrita uma função em C.

A função tem como parâmetro um vector de cinco inteiros com a seguinte informação:

- **V_0 Endereço**

  o endereço do eixo a que se destina o comando.

- **V_1 Comando**

  comando a ser executado pelo eixo, de acordo com o conjunto de comandos implementados.

- **V_2 Estado**

  sem qualquer aplicação na mensagem de comando, é aqui usado apenas para que a mensagem de comando e resposta tenham a mesma estrutura.

- **V_3 Velocidade**

  a velocidade limite pretendida para o eixo.

- **V_4 Posição**

  a próxima posição para a qual o eixo deve avançar.

A informação referente a velocidade e posição é irrelevante em alguns dos comandos.

Esta função gera uma mensagem de comando, de acordo com as especificações do protocolo BITBUS, dirigida à tarefa 1 do nó cujo endereço está defenido em V_0. Esta mensagem é transmitida através da porta paralelo à estação principal que se encarrega de a dirigir à estação secundária endereçada.

Uma vez transmitida a mensagem o processador testa as linhas de controlo da porta paralelo à espera da resposta.

Da mensagem recebida, é extraida a informação com a qual é composto o vector de resposta. O vector de cinco inteiros, da resposta, contém informação com uma estrutura muito semelhante à do comando, assim:

- **V_0 Endereço**

  apenas para possível controlo.

- **V_1 Erro**

  contém os erros respeitantes à transmissão da mensagem e execução do comando

- **V_2 Estado**

  com informação sobre o estado do eixo, nomeadamente fins de curso.

- **V_3 Velocidade**

  o valor da velocidade actual do eixo.

- **V_4 Posição**

  o valor da posição actual do eixo.

É assim possível ao programa do utilizador uma comunicação transparente, através da rede BITBUS com o programa instalado em cada um dos eixos.

5.3.2 SOFTWARE DO EIXO

Para o processador instalado no eixo foi escrito um conjunto de programas com a finalidade de:

- Receber as comunicações do processador central.
- Construir as respostas.
- Interpretar e executar os comandos.
- Ler os sensores locais.
- Actuar as saídas

Todo o software foi desenvolvido tendo em conta as características especiais do processador e muito particularmente o software já instalado no processador 8044.

O programa foi dividido em tarefas de modo a aproveitar as

capacidades do IDCX51. As tarefas desenvolvidas são as seguintes:

- **Tarefa 1**

  responsável pelas comunicações com o "master" e a descodificação dos comandos.

- **Tarefa 2**

  a tarefa base do sistema que se encarrega do controlo dinâmico do eixo. Esta tarefa é repetida com um período de 10 milisegundos. Compete ainda a esta tarefa, a actuação das saídas, leitura das entradas e a activação do "watch-dog".

- **Tarefa 3**

  tarefa criada exclusivamente para o posicionamento inicial do eixo, extinguindo-se de seguida.

- **Tarefa 4**

  tarefa destinada à recuperação de interrupção provocada pelo "watch-dog".

A ocupação de memória interna feita pelo sistema operativo presente no 8044 obriga a um particular cuidado na sua utilização. Para não provocar um crescimento excessivo da "stack" foi evitado neste programa o recurso ao uso de rotinas. A necessária estruturação do programa foi conseguida pelo aproveitamento das capacidades do assemblador utilizado, o ASM51, no uso de macros. Esta opção conduz a um programa mais longo mas no caso concreto o espaço de memória de programa não constitui problema.

As capacidades do IDCX51 para a comunicação entre tarefas de um mesmo eixo não foi aqui utilizado, por ser um processo moroso. Em vez disso a comunicação entre as várias tarefas é feita utilizando variáveis em memória interna.

As variáveis utilizadas para a comunicação são as seguintes:

- **YREF** pedido de posição.
- **VREF** limite de velocidade.
- **YACT** posição actual
- **VLACT** velocidade actual.
- **STATUS** estado actual.

Todas as variáveis ocupam dois bytes excepto status que é de um byte.

5.3.2.1 TAREFA 1

Faz a recepção das mensagens dirigidas ao nó, interpreta o comando contido na mensagem e gera a resposta.

Foi atribuída a esta tarefa a prioridade 3 de modo a não demorar na resposta ao master, nem interferir com a tarefa 0. A tarefa uma vez iniciada é colocada de imediato no estado "asleep", a aguardar a chegada de uma mensagem. Uma vez recebida a mensagem, é feita a descodificação e execução do comando do seguinte modo:

- **Leitura**

  não é executada qualquer acção por este comando.

- **Posição**

  actualiza os valores das variáveis YREF e VREF.

Se o pedido de velocidade ou posição ultrapassa os limites do eixo, os valores de YREF e VREF mantêm-se inalterados e tal facto é assinalado, como erro, na mensagem de resposta.

- **Avanço**

  actualiza VREF de acordo com o valor contido na mensagem. Se VREF for positivo faz YREF igual ao valor do extremo superior do eixo, se for negativo faz YREF igual ao ao valor do extremo inferior do eixo.

- **Inicio**

  por recurso a uma primitiva do sistema cria a tarefa 3 que se encarrega de controlar o eixo até ser atingida a posição de referência.

- **Paragem**

  o valor de VREF é igualado a zero forçando assim a paragem do eixo.

Qualquer que tenha sido o comando executado, a tarefa usa o mesmo buffer para criar a mensagem de resposta, para tal altera o cabeçalho e os dados:

- Altera para resposta o bit MT no cabeçalho da mensagem, mantendo inalterado todo o restante.
- Altera o byte 9 (comando) para 0 se a operação foi bem executada ou escreve o código do erro.
- Actualiza o byte 10 com o valor da variável estado.
- Actualiza os valores dos bytes 11 e 12 com o valor da variável VLACT (velocidade).

- Actualiza os valores dos bytes 13 e 14 com o valor da variável YACT (posição).

Por recurso a uma função do sistema operativo transmite a mensagem resposta e volta ao estado "asleep", aguardando nova mensagem.

5.3.2.2 TAREFA 2

A tarefa 2 faz o controlo dinâmico de posição e velocidade. De acordo com o tempo de amostragem definido, esta tarefa é repetida, com um período de 10 milisegundos. O controlo do tempo é feito pelo temporizador próprio da tarefa. A sua prioridade foi definida em 2, para que esta tarefa possa ser interrompida pela de comunicações e assim, manter elevada rapidez na resposta às mensagens.

Esta tarefa lê as entradas do eixo, actualiza as respectivas variáveis, faz o reset do "watch-dog" e o controlo dinâmico do eixo.

No caso da leitura do contador de posição, ao valor lido é adicionado um desvio, de modo a definir o zero do eixo no ponto que seja mais conveniente para a aplicação.

A velocidade é calculada e actualizado o valor da variável VLACT. A velocidade é expressa em passos da régua (no caso 0,005 mm) por período de amostragem (10 milisegundos).

De acordo com a leitura das entradas digitais é actualizado o valor da variável status.

De acordo com os valores lidos e o estado de VREF e YREF,

esta tarefa calcula o valor a fornecer à válvula do comando hidráulico. O algoritmo utilizado para o controlo dinâmico do eixo e os resultados obtidos serão discutidos no capítulo 6, referente aos testes realizados.

5.3.2.3 TAREFA 3

A tarefa 3 será criada pelo comando INICIO e destina-se a colocar o eixo na posição inicial.

Esta tarefa torna-se necessária pelo facto de os codificadores de posição serem do tipo incremental e não estar previsto nas placas alimentação de "backup".

Os sensores de posição dispõem além dos canais A e B de uma referência de zero que pode não ser única. Nos sensores com mais que uma referência de zero será instalado um interruptor que só estará activo em coincidência com uma delas, tornado-a assim única.

A esta tarefa foi atribuída a prioridade 1 de modo a nunca interromper a tarefa 2. De notar que o tempo de execução da tarefa 2, mesmo se interrompida pela tarefa 1, será previsivelmente muito inferior ao período de repetição. Assim entre cada repetição da tarefa 2 será executada a tarefa 3. A actuação da tarefa 3 sobre o sistema é levada a cabo pela conveniente actualização das variáveis YREF e VREF.

A tarefa 3 executa as seguintes acções:

- Inicia os contadores com o valor correspondente ao extremo superior.
- Faz YREF (posição pedida) igual ao valor do extremo

inferior.

- Atribui a VREF (velocidade pedida) a um valor seleccionado de modo a que o eixo se mova a velocidade moderada (escolhida de acordo com as características do eixo).
- Enquanto o fim de curso inferior não for atingido esta tarefa mantem-se a ler a palavra de estado. De notar que dada a baixa prioridade da tarefa 3 ela será entretanto interrompida pelas restantes que controlam o eixo.
- Quando atingido o fim de curso inferior, YREF é alterado para o valor correspondente ao extremo superior do eixo.
- A tarefa 3 volta a ficar em ciclo a ler a palavra de status e a aguardar a activação do interruptor associado à referência de zero.
- Atingido este interruptor a VREF é alterada de modo a que a velocidade seja agora muito baixa. Como a referência da régua activa uma interrupção, esta tarefa, por recurso a uma primitiva do sistema operativo, ficará no estado de "asleep" a aguardar a interrupção.
- Atingida a referência, os contadores são carregados com o valor correspondente à posição da referência.
- Um bit da palavra de status é activado de modo a especificar que o eixo já está calibrado.
- Em YREF é agora colocado o valor da posição de

repouso do eixo e em VREF um valor conveniente para a velocidade de avanço.

- Tendo cumprido a sua função a tarefa extingue-se por recurso a uma primitiva do sistema operativo.

5.3.2.4 TAREFA 4

Esta tarefa é definida com prioridade 4 de modo a poder interromper outra qualquer que porventura fique em ciclo. A primeira vez que esta tarefa for activada (no reset) ficará em estado "asleep" à espera da interrupção que lhe está atribuída, a interrupção IE0. Se esta interrupção for activada por falta de actualização do "watch-dog", pela tarefa 2, fará o seguinte:

- Força a 0 a variável VREF provocando assim a paragem do eixo.
- Força a 1 o bit 1 da palavra de status.

6 RESULTADOS EXPERIMENTAIS

Para o teste da unidade desenvolvida foi instalado no laboratório um sistema constituido por:

- Central hidráulica
- Actuador linear hidráulico com um curso útil de 300 mm
- Codificador linear com uma resolução de 0,02 mm
- Servo-válvula de controlo digital.

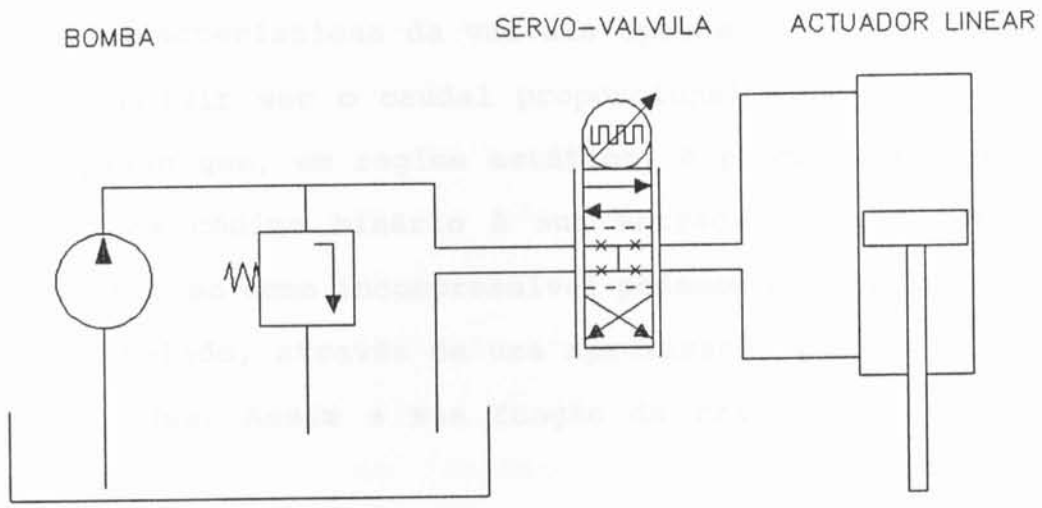


Fig. 6.1 Sistema Hidráulico

Nos testes realizados a unidade de leitura de posição foi programada para a leitura com resolução multiplicada por 4, resultando assim uma resolução efectiva de 0,005 milimetros. As características da válvula, referidas no capítulo 1, implicam um

tempo de amostragem maior que 8 milisegundos pelo que foi definido, como tempo de amostragem 10 milisegundos. Nos testes realizados, este valor é usado como unidade de tempo.

Os limites de velocidade definidos no capítulo 1 são, para os eixos lineares, de 30 milímetros por segundos ou seja 60 unidades da régua por período de amostragem.

6.1 FUNÇÃO DE TRANSFERÊNCIA DO SISTEMA

No sistema referido, a variável a ser controlada (i. é. a variável de saída) é a posição do actuador hidráulico e a variável de controlo (i. é. a variável de entrada), um número binário de oito bits, a fornecer à entrada da servo-válvula.

Das características da válvula apresentadas no capítulo 1 podemos concluir ser o caudal proporcional à posição do motor passo a passo que, em regime estático, é proporcional ao valor do número em código binário à sua entrada. Nestas condições, admitindo o óleo como incompressível podemos modelizar o sistema a ser controlado, através de uma aproximação por um sistema de primeira ordem. Assim a sua função de transferência pode ser expressa por:

$$G = \frac{K_q}{s} \qquad (6.1)$$

sendo a constante K_q um ganho a ser determinado.

6.2 DETERMINAÇÃO DE K_q

A constante K_q representa a relação entre o valor fornecido à válvula e a velocidade do actuador.

Para a sua determinação foram utilizadas as capacidades do sistema construído. O computador central envia, através da rede BITBUS, um comando que tem como parâmetro o valor da abertura da válvula. O processador de eixo transfere para a válvula o parâmetro recebido do comando. Nestas condições o valor à entrada da válvula é constante. O processador central interroga o eixo, durante o tempo suficiente para que a velocidade atinja um valor constante. O gráfico da fig. 6.2 exibe os valores da velocidade de actuador, depois de atingido o regime estável, para várias aberturas da válvula.

Do gráfico podemos concluir que a associação da válvula e do actuador disponíveis não são de modo nenhum adequados à execução dos testes, usando os limites de velocidade definidos anteriormente (i. é. 30mm/s). De facto o uso desta gama de velocidades implicaria que a válvula fosse usada num intervalo muito estreito da sua característica. Para evitar esta dificuldade os testes forão realizados, usando como limite de velocidade 200 unidades da régua por período de amostragem ou seja, 100 mm/s.

No gráfico da fig. 6.3 é relacionado o ganho do sistema Kq com a abertura da válvula. Mais uma vez é aqui notória a incapaciadade da válvula em responder segundo o gráfico da fig. 1.4.

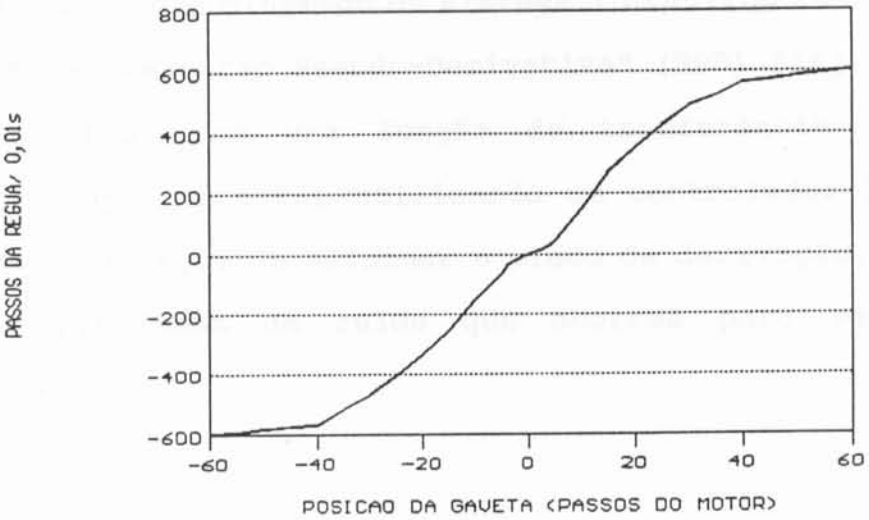


FIG. 6.2 VELOCIDADE DO ACTUADOR - ABERTURA DA VÁLVULA

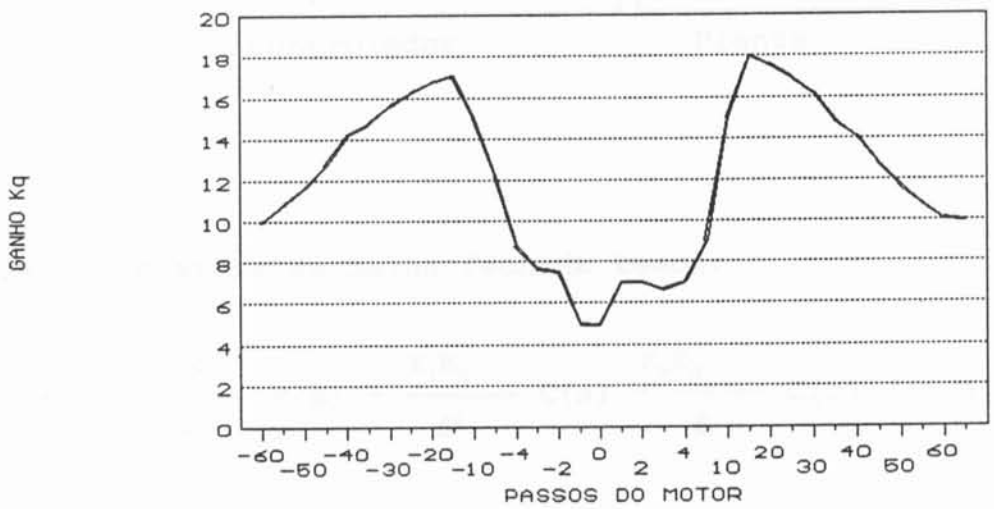
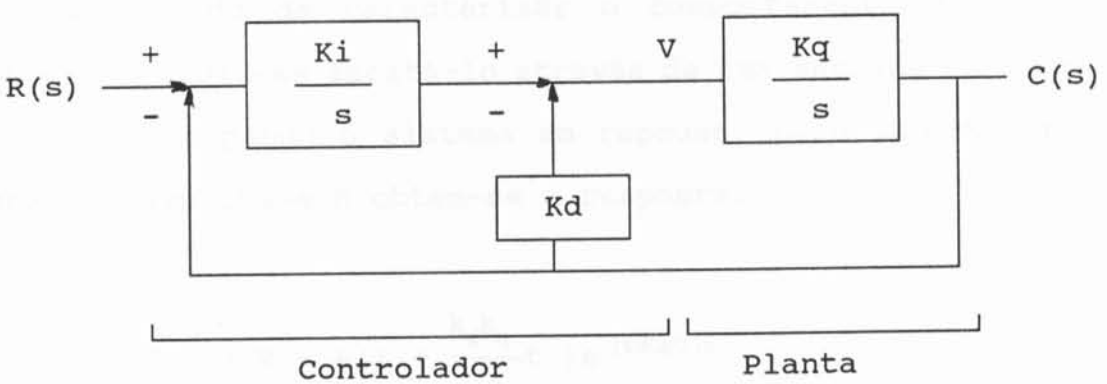


Fig 6.3 Valor de Kq para cada abertura da válvula

6.3 CONTROLO DINÂMICO

Para o controlo dinâmico do sistema hidráulico foi utilizado o método "Realimentação Pseudo-Derivativa" (RPD) [14]. Trata-se de um algorítmo com uma função de tranferência C(s)/R(s) semelhante à que se obtém utilizando um controlador PID. Este método tem a vantagem de eliminar o bloco de derivação, evitando assim, os problemas de ruído que ocorrem para velocidades próximas de zero.



Para o sistema em malha fechada temos:

$$C(s) = \frac{K_i K_q}{S^2} R(s) - \frac{K_i K_q}{S^2} C(s) - \frac{K_q K_d}{S} C(s) \quad (6.2)$$

$$\frac{C(s)}{R(s)} = \frac{K_i K_q}{s^2 + K_q K_d s + K_i K_q} \quad (6.2)$$

A equação característica do sistema é assim:

$$s^2 + K_q K_d s + K_i K_q = 0 \qquad (6.3)$$

Em robótica é corrente forçar o sistema de tal modo que este exiba uma resposta rápida, mas nunca subamortecida. Nesta ordem de ideias, o sistema deverá ter uma equação característica com o amortecimento crítico, resultando a relação:

$$\frac{K_q K_d^2}{4} = K_i \qquad (6.4)$$

6.3.1 RESPOSTA AO DEGRAU

No sentido de caracterizar o comportamento dinâmico do sistema, decidiu-se excitá-lo através de uma entrada em degrau.

Assim, supondo o sistema em repouso, para uma entrada em degrau de amplitude R obtem-se a resposta:

$$C(t) = R - R(1 + \frac{k_d k_q}{2} t\ ) e^{-(KdKq/2)t} \qquad (6.5)$$

O sistema apenas será linear se as limitações impostas pela servo-válvula não forem ultrapassadas, ou seja se o sistema não for forçado à saturação.

As características da válvula indicam que a sua gama de variação está compreendida entre -60 e +60 passos do motor passo a passo.

Sendo a variável V, a referência numérica que o controlador fornece à válvula:

$$V(s) = \frac{sC}{K_q} \qquad (6.6)$$

ou seja

$$V(t) = R \frac{K_q K_d^2}{4} te^{-(KqKd/2)/t} \qquad (6.7)$$

esta variável tem um máximo

$$V_{máx} = R \frac{K_d}{5,436} \qquad (6.8)$$

que ocorre para $t = \frac{2}{K_q K_d}$ (unidade de tempo 0,01s)

A segunda limitação do sistema tem a ver com o gradiente de variação de V, já que o motor passo a passo, que controla a gaveta da válvula, tem a velocidade limitada a 600 passos por segundo, ou seja 6 passos por período de amostragem.

Representando por D este gradiente, então o seu valor é máximo no instante inicial e é dado por:

$$D_{máx} = R \frac{K_q K_d^2}{4} \qquad (6.9)$$

Fazendo Kq = 4, o valor obtido do gráfico da fig. 6.3, para zonas à volta de zero, os valores de Ki e Kd são:

- Ki = 0,0004
- Kd = 0,02

O funcionamento do sistema será linear, com estas constantes, se o degrau de excitação não ultrapassar os 15000

passos da régua.

Foram realizados testes utilizando como excitação degraus com valores de 2000 (1), 5000 (2), 10000 (3), 20000 (4) e 25000 (5) passos da régua ou seja 10, 25, 50, 100 e 125 mm. Dos resultados obtidos foi feito o gráfico da fig. 6.3. Nele se referenciam os valores dos pedidos do sistema à válvula nos instantes iniciais. É bem visível a capacidade do sistema responder ao degrau de 10000 passos da régua e a sua nítida saturação para o pedido de 20000.

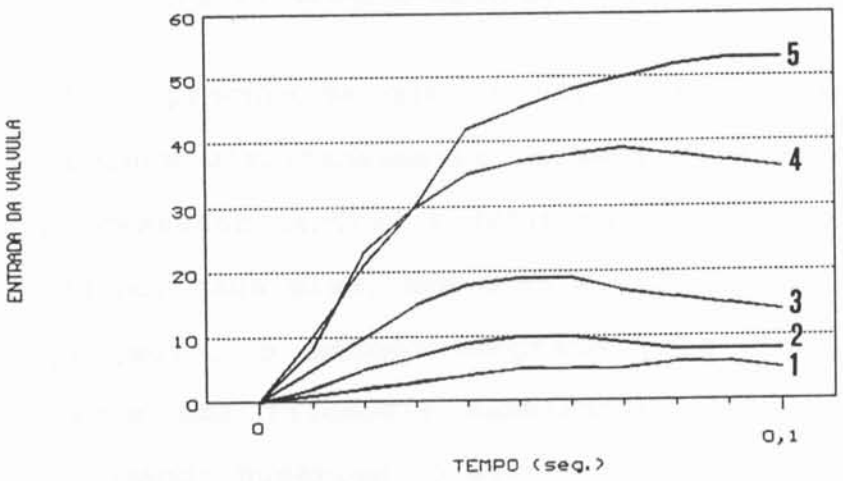


Fig. 6.4 Pedidos à válvula para valores próximos da origem

O gráfico da fig. 6.4 foi realizado com valores de degrau iguais ao caso anterior. Nele são visíveis os valores máximos dos pedidos feitos à válvula que estão ainda longe da saturação.

6.3.2 CONTROLO DE VELOCIDADE

Nesta secção pretende-se que o eixo seja controlado eficientemente, não só em posição, mas também em velocidade.

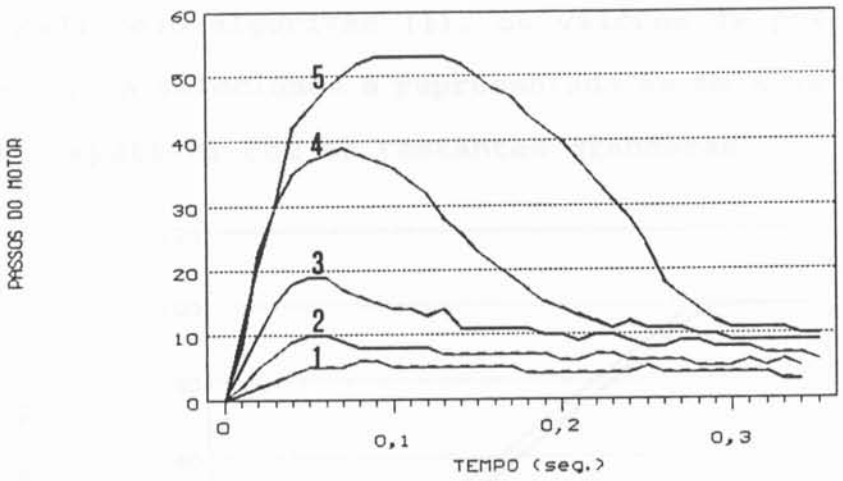


Fig. 6.5 Valores máximos pedidos à válvula

Em robótica procura-se que os vários eixos de um braço iniciem e terminem simultaneamente os seus movimentos. Assim, compete ao processador central a definição da próxima posição, a ser atingida por cada eixo, bem como a velocidade.

Neste projecto, o método adoptado para o controlo de velocidade segue uma filosofia semelhante à do controlo de máquinas com comando numérico. O eixo é controlado em posição, todavia a referência integra a informação de velocidade através da definição de uma rampa, cujo gradiente é a velocidade desejada. Para tal, por cada período de amostragem, à última referência de posição é somado o valor da velocidade, obtendo-se assim a nova referência. O processo repete-se, até a referência de posição igualar a posição final desejada.

Os resultados obtidos por este método podem ser observados nos gráficos das figuras seguintes. Os gráficos foram construídos a partir de valores obtidos para um deslocamento de 100 mm, com diferentes limites de velocidade.

Em cada um dos gráficos são representadas, a rampa de posição gerada pelo algoritmo (1), os valores da posição (2) e velocidade (3). A velocidade é representada em mm/s de modo a ter uma escala compatível com as restantes grandezas.

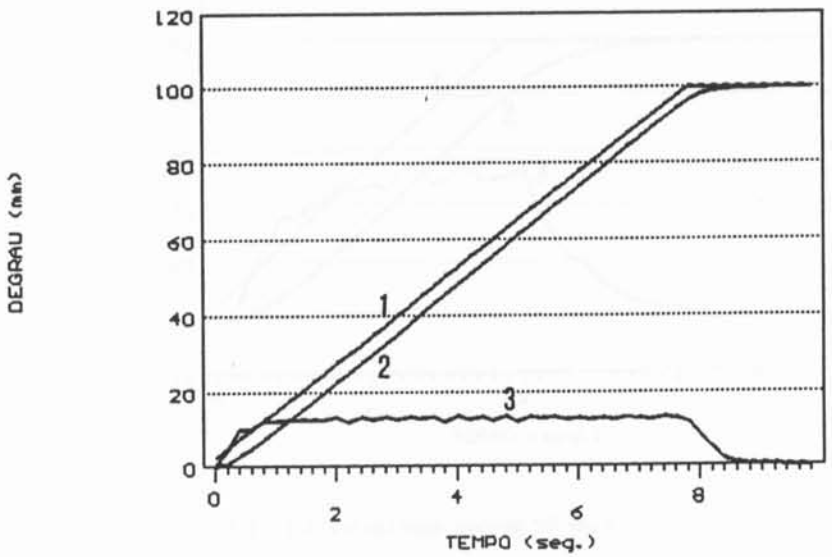


Fig. 6.6 Velocidade pedida 12,5 mm/s

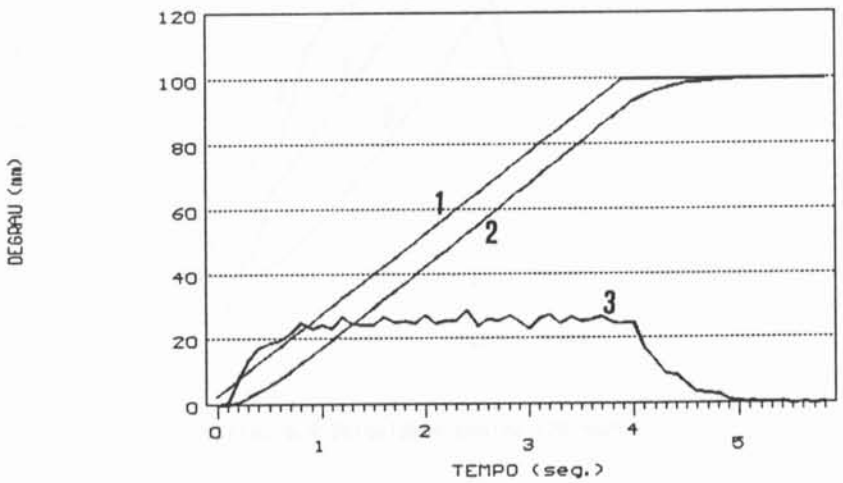


Fig. 6.7 Velocidade pedida 25 mm/s

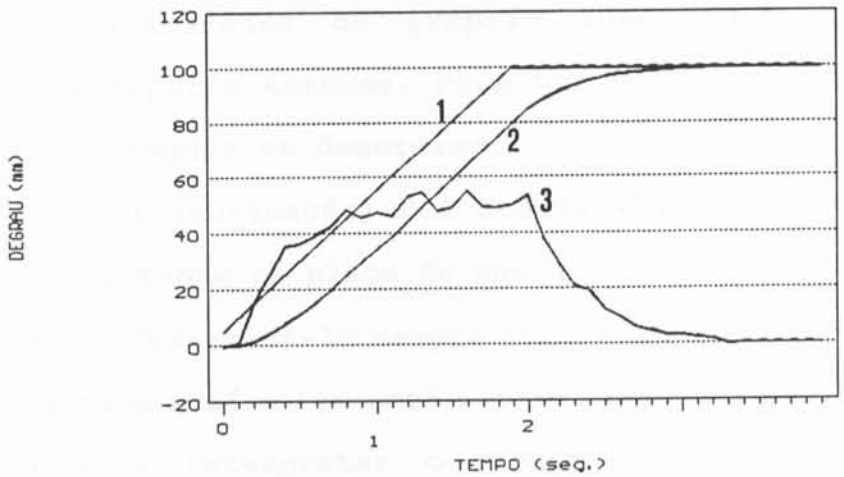


Fig. 6.8 Velocidade pedida 50 mm/s

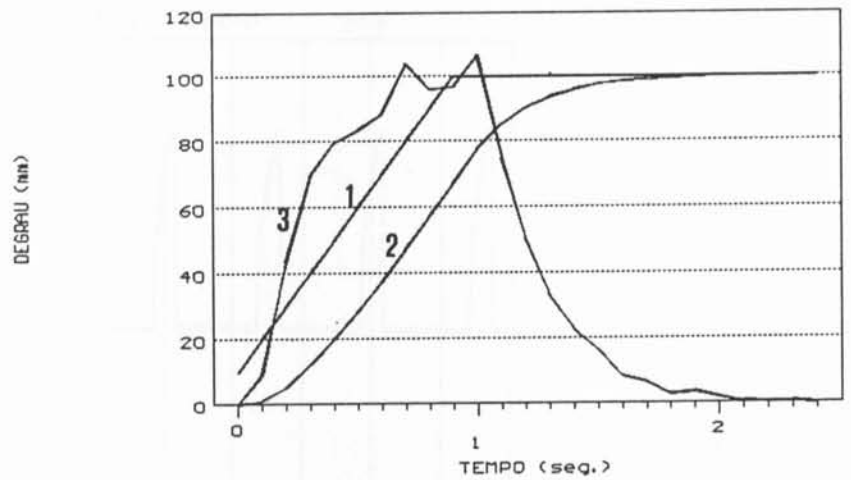


Fig. 6.9 Velocidade pedida 100 mm/s

6.4 OCUPAÇÃO DO PROCESSADOR

A implementação da rede BITBUS, aproveitando para o controlo do eixo as capacidades do próprio 8044, implica algumas limitações que importa estudar. Para tal, foi feito o conjunto de testes que a seguir se descrevem.

A tarefa 1 (comunicação) foi modificada de modo a activar o led verde, existente na placa de núcleo, sempre que é recebida uma mensagem e a desactivá-lo sempre que seja enviada a resposta. O tempo, durante o qual o led está activo, corresponde à ocupação do processador a interpretar o comando recebido e gerar a mensagem de resposta. Foi possível, deste modo, obter o gráfico da figura 6.10, onde pode ser verificado que o tempo gasto na interpretação e resposta a uma mensagem é de cerca de 0,85 milisegundos.

FIG. 6.10 Tempo de resposta a uma mensagem

Os valores apresentados dizem respeito à descodificação do comando "informação".

Usando um método semelhante com a tarefa 2 (controlo dinâmico), actuando agora sobre o led vermelho, permitiu obter o gráfico da figura 6.11. Neste gráfico é visível o tempo de amostragem de 10 milisegundos e o tempo em que o processador executa o cálculo respeitante ao controlo do eixo, cerca de 1,7 ms.

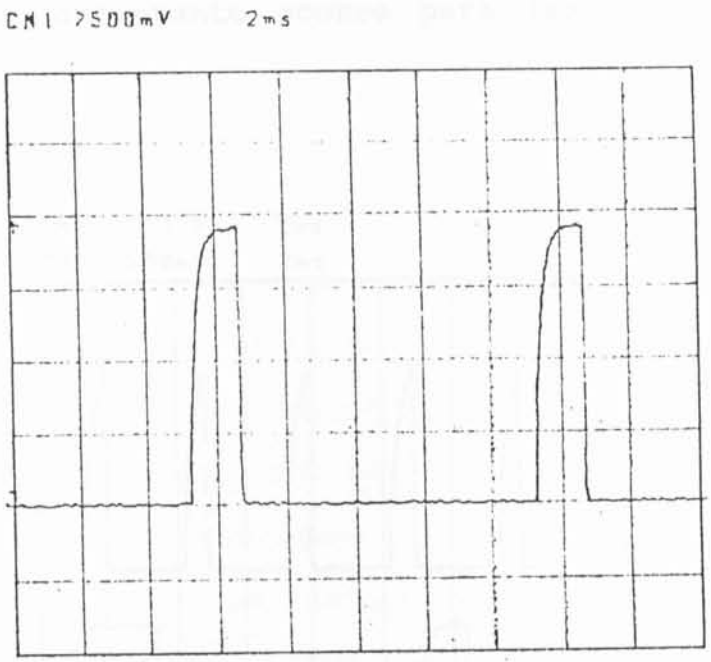


Fig 6.11 Tempo de execução da tarefa de controlo

Como foi já referido à tarefa 1 foi atribuída uma prioridade mais alta que a tarefa 2, para evitar atrasos nas comunicações. Para testar as capacidades da rede de comunicação e a sua

possível interferência no controlo dinâmico do eixo foi feito o seguinte teste. O processador central envia a um eixo mensagens de comando com uma frequência próxima de 400 hertz enquanto o processador de eixo continua a sua tarefa de controlo dinâmico. Os resultados obtidos podem ser observados no gráfico da figura 6.12. O tempo que decorre entre o instante em que a tarefa 2 é iniciada o seu fim, no caso de ser interrompida, é de 3 milisegundos. O tempo de duração da tarefa 2 é neste caso superior em cerca de 0,5 milisegundos à soma dos tempos da tarefa 1 e 2 actuando individualmente. Esta diferença justifica-se, quer pelo tempo gasto pelo sistema operativo, quer pela execução da tarefa 0 que entretanto ocorre para fazer a transmissão da mensagem.

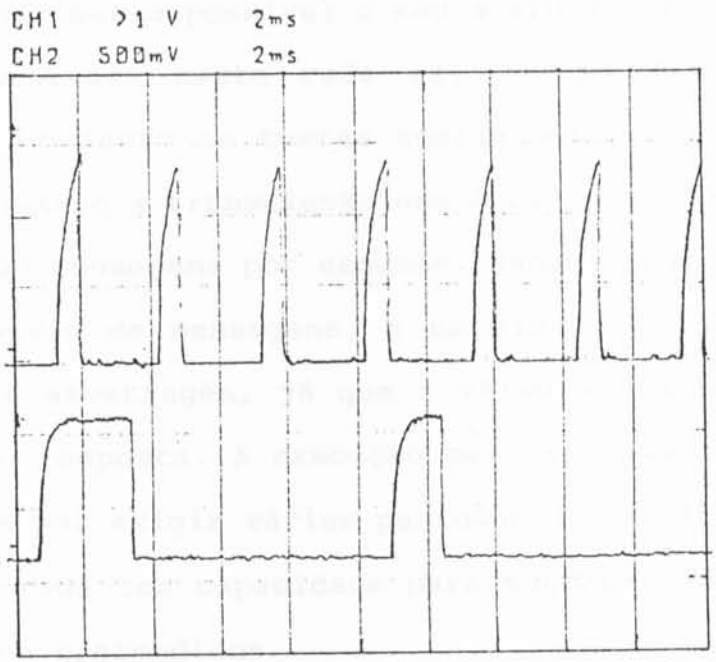


Fig. 6.12 Interferência das comunicações na tarefa de controlo

7 CONCLUSÕES

A análise dos resultados obtidos no capítulo 6 permite concluir que a arquitectura seleccionada satisfaz as necessidades do projecto.

Do ponto de vista da rede, a escolha do protocolo BITBUS, implementado com os processadores 8044, revela-se uma solução económica e com capacidade para responder às exigências da aplicação. O comprimento da mensagem permitida é perfeitamente adequado, sendo mesmo possível o seu alargamento. A cadência de mensagens possíveis nesta rede está acima das necessidades previsíveis. De facto os testes realizados sobre um dos eixos permitiram, usando a transmissão com a cadência de 375 Kbits/s, o envio de 400 mensagens por segundo. Não há qualquer interesse prático no envio de mensagens, a um eixo, com uma frequência superior à de amostragem, já que a válvula hidráulica não tem capacidade de resposta. A execução pelo sistema de um qualquer comando acaba por exigir vários períodos de amostragem, assim a rede implementada tem capacidade para suportar as comunicações dos seis eixos pretendidos.

Do ponto de vista da capacidade do processador para fazer o controlo do eixo, o tempo gasto no cálculo é muito inferior ao

período de amostragem pretendido. O frequência de amostragem utilizada nos testes foi de 100 hertz, no entanto a unidade construída revela capacidades para frequências mais elevadas.

A arquitectura seleccionada é perfeitamento modular permitindo a sua utilização não só no controlo de manipuladores mas igualmente em outros tipos de máquinas em que seja importante o controlo de posição e velocidade. É o caso das máquinas com comando numérico.

A opção de contruir, em placas diferentes, o núcleo do BITBUS e a placa de aplicação, permite que o trabalho desenvolvido possa ser facilmente configurado para outras aplicações eventualmente diferentes do comando de posição.

BIBLIOGRAFIA

[1] STEPPER MOTOR DRIVEN HYDRAULIC VALVES

FREITAS, F.J.T.

RESTIVO, M.T.

ALMEIDA, A.A.R.

Departamento de Engenharia Mecanica da F.E.U.P.

8th International Symposium on FLUID POWER

Birminghan, 19-21 Abril de 1988

[2] MICROCONTROLLES OFFER REAL TIME ROBOTICS CONTROL

IRA HORDEN

Computer Design (Outubro, 1985) 98-101.

[3] A COMMUNICATION PROTOCOL FOR SINGLE CHIP PROCESSORS

N. C. BURD

Journal of Microcomputer Applications (1986) 9, 105-111

[4] INFORMATIC INDUSTRIELLE IV

HENRY NUSSBAUMER

Editora Presses Polytechniques Romandes

[5] INTRODUCING DATA COMMUNICATIONS PROTOCOLS

PETER HARDY

[6] ROBOTICS

JOHN J. CRAIG

Editora Addsison Wesley

[7] 8044AH/8344AH HIGH PERFOMMANCE 8-BIT MICROCONTROLLER WITH ON-CHIP SERIAL COMMUNICATION CONTROLLER

Intel

[8] THE BITBUS INTERCONNECT SERIAL CONTROL BUS SPECIFICATION
Intel

[9] DEVELOPMENT DATA PCB80C552/PCB83C552
Philips

[10] 8044 BITBUS ENHANCED MICROPROCESSOR PROGRAMAR'S SUMMARY
Intel

[11] GUIDE TO USING THE DISTRIBUTED CONTROL MODULES
Intel

[12] iDCX51 DISTRIBUTED CONTROL EXECUTIVE USER'S GUIDE
Intel

[13] UPI-452 CHMOS PROGRAMABLE I/O PROCESSOR
Intel

[14] AUTOMATIC CONTROL SISTEMS
RICHARD M. PHELAN
Editora Cornell University Press

[15] 8051 ARCHITECTURAL SPECIFICATION AND FUNCTIONAL DESCRIPTION
Intel

ANEXO A

PROGRAMAS

```
        /*  ****************************************** */
        /*                                             */
        /*        COMUNICAÇÃO DO PC A UM EIXO           */
        /*                                             */
        /*  ****************************************** */


#include<dos.h>
#include<stdio.h>

int   p_out(int port,unsigned char valor)
;
int  trx ( unsigned char *pbuf )
;
void   cria_buf_tx( unsigned char *pbuf , unsigned char *ptrx )
;
int    bitbus ( int *trx_eixo )
;
void  cria_buf_rx (unsigned char *pl , unsigned char  *ptrx)
;
int   rcx_bit(unsigned char *buf_rmx)
;
int
      status_port = 0x20c,
      data_port   = 0x208,
      comand_port = 0x20a;
unsigned char
      status,
      tfnf = 0x01,
      rcmd = 0x20,
      rfne = 0x10;
```

```
int     bitbus ( int *trx_eixo )

/*      tem como parâmetro um vector de inteiros, promove
        a sua transmissão através da rede BITBUS e faz a
        recepção da resposta.          */

        {
          unsigned char  *pl, *ptrx;
          int            cntr_erro ;

          ptrx = (unsigned char *)malloc(12);
          pl = (unsigned char *)trx_eixo;
          cria_buf_tx( pl,ptrx);
          cntr_erro = trx(ptrx);
          if (!cntr_erro)
          {   cntr_erro = p_out(comand_port,0x00);
              if (!cntr_erro)
                  {
                     cntr_erro = rcx_bit(ptrx);
                     cria_buf_rx ( ptrx , pl) ;
                  }
           }
                    else
              trx_eixo[1] = cntr_erro ; /* indicação de erro */
          return (cntr_erro) ;

        }
```

```
int   p_out(int port,unsigned char valor)

/* testa o estado da porta de comunicação e se disponível
   envia um caracter. A funçao dispõem de detecção de erro
  por excesso de tempo. */

     {

    int  tempo = 30000, erro = 0, erro7 = 7   ;
    unsigned char status;

       do
         {
           --tempo;
           status = inportb(status_port);
        }

     while ((status & tfnf) && tempo);

     if (tempo != 0)
         {
         outportb(port,valor);
         }
         else
           erro = erro7;
     return (erro);

    }
```

```
int  trx ( unsigned char pbuf[] )

       /* por recurso à função P_OUT() faz a transmissão dos
          bytes presentes em pbuf[]. Testa o comprimento de
          pbuf[] presente em pbuf[0] */

  {
  int  i,erro = 10 ;

    for (i =0; i != (pbuf[0]-2) ; i++)
       erro =  p_out (data_port,pbuf[i])  ;

    return(erro);
  }
```

```
void   cria_buf_tx( unsigned char *pbuf , unsigned char *ptrx )

/*   Esta função acrescenta à informação a ser transmitida o
cabeçalho  de acordo com as especificações BITBUS        */

     {

       ptrx [0]  = 0x0E    ;  /*comprimento da mensagem  */
       ptrx [1]  = 0x40    ;
       ptrx [2]  = 0x80    ;  /* pbuf [0]              */
       ptrx [3]  = 0x01    ;  /*tarefa destino         */
       ptrx [4]  = 0x80    ;  /*comando                */
       ptrx [5]  = pbuf [0];  /* endereco do eixo      */
       ptrx [6]  = pbuf [2];  /* comando ao eixo       */
       ptrx [7]  = pbuf [4];  /* byte de flags         */
       ptrx [8]  = pbuf [6];  /* velocidade            */
       ptrx [9]  = pbuf [7];  /* velocidade            */
       ptrx [10] = pbuf [8];  /* próxima posição       */
       ptrx [11] = pbuf [9];  /* próxima posição       */
       ptrx [12] = pbuf [10];
       ptrx [13] = pbuf [11];
     }
```

```
void  cria_buf_rx (unsigned char *pl , unsigned char *ptr)

 /*  separa, do cabeçalho, a informação contida na mensagem   */

      {
       int    i;

        ptr[0]  = pl[5] ;          /* endereço */
        ptr[1]  = 0x0   ;
        ptr[2]  = pl[6] ;          /* comando */
        ptr[3]  = 0x0   ;
        ptr[4]  = pl[7] ;          /* flag    */
        ptr[5]  = 0x0   ;
        ptr[6]  = pl[8] ;          /* velocidade */
        ptr[7]  = pl[9] ;          /* velocidade */
        ptr[8]  = pl[10];         /* posição    */
        ptr[9]  = pl[11];         /* posição     */
        ptr[10] = pl[12];        /* reservado   */
        ptr[11] = pl[13];        /* reservado  */

      }
```

```
int   rcx_bit(unsigned char *buf_rmx)

/*      faz a recepção da mensagem de resposta     */
     {

    int   i,
          tempo = 30000;
    unsigned char
          status,
          comando = 0xff,
          fim_msg = 0;
    int  erro = 0,
          erro4 = 4;

    i = 0;
       do
    {
      tempo = 30000;
         do
      {
              /* lê a porta de status até à chegada de uma
                   mensagem de resposta  */
          -- tempo;
         status = inportb(status_port);
      }
      while((status & rfne) && tempo  );
     /* lê toda a mensagem, verificando se o bute recebido é o
              último  */
       if (tempo != 0)
        {
         status = inportb(status_port);
          if (status & rcmd)
          {
            buf_rmx[i] = inportb(data_port);
            i++;
          }
```

```
            else
            {
             comando = inportb(comand_port);
              fim_msg = 1;
            }
          }
        }
      while ( (fim_msg == 0) & (tempo != 0));
      if ((comando != 0) & (erro == 0))
            erro = erro4;
            if (erro == 0)
                erro = buf_rmx[4];
      return ( erro );
  }


   main ()
```

/* O programa "MAIN" aqui desenvolvido tem como função essencial a teste da unidade desenvolvida. Começa por interrogar através do monitor qual o comando desejado. De acordo com o comando, pede os necessários parâmetros. Por recurso à funçao BITBUS() faz a sua transmissão ao nó respectivo e responde através do monitor. A resposta é, um código de erro no caso de problemas na transmissão ou interpretação do comando ou o estado actual do processador.

```
   {
   int      i,
          output[6],
          erro,
          num,
          cmd;
```

```
output[1] = output[2] = output[3] = 0;
output[4] = output[5] = 0;
output[0] = 128;

cmd = 1;
while  (cmd != 0)
{
 printf(" COMANDO ?? ");
 scanf("%d",&num);
 cmd = num;
 if (cmd != 0)
  {
    if (cmd == 1)
     {
       output[1] = 1;
       output[2] = output[3] = output[4] = 0;
       output[0] = 128;
     } ;
     if (cmd == 2)
      {
        output[0] = 128;
        output[1] = 2;
        printf(" VELOCIDADE ? ");
        scanf("%d",&num);
        output[3] = num;
        printf(" POSICAO ?    ");
        scanf("%d",&num);
        output[4] = num ;
      } ;
     if (cmd == 3)
      {
        output[0] = 128;
        output[1] = 3  ;
        printf(" VELOCIDADE ? ");
        scanf("%d",&num) ;
        output[3] = num ;
```

```
        };
         erro = bitbus(output);
         if (erro)
            printf("ERRO NUMERO:   %d\n",erro);
         else
            {
             printf(" ENDERECO      %d\n",output[0]);
             printf(" A POSIÇÃO     %d\n",output[4]);
             printf(" VELOCIDADE    %d\n",output[3]);

            }
          }
        }
}
```

```
;
;
;                    ************************************
;                    *                                  *
;                    *          DECLARAÇÖES             *
;                    *                                  *
;                    ************************************
;
;
;
;       CONSTANTES
;
MSC_FC_L        EQU     0       ;máscara de fim de curso baixo
MSC_FC_H        EQU     0       ;máscara de fimde curso alto
MSC_REF0        EQU     0       ;máscara de referenciade 0
FAZER_NADA      EQU     1       ;código da instrução leitura
AVNC_NRML       EQU     2       ;código da instrucao posição
POS_INIC        EQU     4       ;código da instruçãoao início
VELOCI          EQU     3       ;avançar em velocidade
ERR_TNC         EQU    33     ;código de erro de tarefa não criada
MSTR_ADRS       EQU     0       ;endereço do master
FLG_RSP         EQU     0       ;byte de flags de resposta
ADDR_MSTR       EQU     0       ;endereço do master
SRC_DEST        EQU     0       ;
RESP_OK         EQU     0       ;resposta sem erro
OFFSETL         EQU     0      ;para deslocar o zero dos contadores
OFFSETH         EQU     80H     ;
LIMVL           EQU     0FFH    ;limite de velocidade
LIMVH           EQU     0       ;
TMPR2           EQU     10      ;tempo entre chamadas da task2
VEL_BAIXA_L     EQU     10      ;velocidade de aproximação
VEL_BAIXA_H     EQU     0       ;para a referência
VEL_MEDIA_L     EQU     80      ;velocidade de avanço
VEL_MEDIA_H     EQU     0       ;na tarefa de inicializacao
PRT_COUNT       EQU     0       ;porta de preset do contador
VALINICL        EQU     0       ;posição do indice de referência
VALINICH        EQU     0
```

```
EX_ALTL         EQU    30H   ;limite superior do eixo
EX_ALTH         EQU    75H
EX_INFL         EQU    0D0H     ;limite inferior do eixo
EX_INFH         EQU    8AH
PORT_DIG        EQU    0FF40H ;endereço da porta digital
PRT_POS         EQU    0FF48H ;endereço da porta de posição
PRT_RES         EQU    0FF43H ;porta de reset e latch
ENDERECO        EQU    80H    ;endereço do eixo
LIMVDIGL        EQU    60     ;limite imposto pela válvula digital
LIMVDIGH        EQU    0
GREEN_LED       EQU    91H    ;led verde para testes
RED_LED         EQU    90H
YINICL          EQU    0      ;posição inicial a ser carregada
YINICH          EQU    0      ;nos contadores
MOVEL           EQU    3
HOMEL           EQU    0
HOMEH           EQU    0
FACESCL         EQU    2AH
FACESCH         EQU    0
CONSTKI         EQU    30     ;valor inicial da constante
derivativa
CONSTKD         EQU    2      ;valor da constante de integracao
LIMDIGL         EQU    60     ;limite imposto pela válvula digital
LIMDIGH         EQU    0
;
;
VARS    SEGMENT    DATA
          RSEG       VARS
YREF:       DS       2      ;pedido de posição
YREFT:      DS       2
VREF:       DS       2      ;pedido de velocidade
VREFT:      DS       2
YACT:       DS       2      ;posição actual
REFER:      DS       2      ;
BUFFERX:    DS       4      ;
BUFFERY:    DS       4
BUFAUX:     DS       5
```

```
INTGR:      DS      4       ;variável integral
VLACT:      DS      2       ;cópia da velocidade actual
LIMIT1:     DS      2
LIMIT2:     DS      2
STATUS:     DS      1       ;fins de curso etc.
TEMP:       DS      2       ;
CMND_RSP:   DS      1       ;resposta no corpo da mensagem
MSG_PTR:    DS      1       ;guarda ponteiro do buffer de mensagem
TEMP1:      DS      2
KI:         DS      1
KD:         DS      1
YP:         DS      4
TMP1:       DS      2
;
;
BOOLEAN     SEGMENT     BIT
            RSEG    BOOLEAN     ;espaço de variáveis booleanas
FCMP:       DBIT        1       ;resultado da operação compare
BITCX:      DBIT        1       ;
BITCY:      DBIT        1       ;
FLAG_DIV:   DBIT        1       ;flag de sinal para a divisão
FIM_CURSOL: DBIT        1       ;activo no extremo inferior
FIM_CURSOH: DBIT        1       ;activo no extremo superior
CARRY:      DBIT        1
SINAL:      DBIT        1
;
;
;TABELA     SEGMENT     XDATA
;           RSEG        TABELA      ;
```

```
%' '
%'                 **********************************    '
%'                 *                                *    '
%'                 *  MACROS EM COMPLEMENTO PARA 2  *    '
%'                 *                                *    '
%'                 **********************************    '
%' '
%' '
%'          macros usando apenas dois bytes  '
%' '
%'        faz (MEM1) = (MEM1) + (MEM2) '
%'             MEM1 e MEM2 são endereços de memória interna'
%' '
%*DEFINE(SOMA(MEM1,MEM2))
%'aw
        (
        MOV     A,%MEM1
        ADD     A,%MEM2
        MOV     %MEM2,A
        MOV     A,%MEM1+1
        ADDC    A,%MEM2+1
        MOV     %MEM1+1,A
        )
%' '
%*DEFINE(SUBTRAI(MEM1,MEM2))
%' '
%' '
%'      (MEM1) = (MEM1) - (MEM2)  '
%'      MEM1 e MEM2 são endereços de memória interna  '
%' '
        (
        %COMPLEMENTA(%MEM2)
        %SOMA(%MEM1,%MEM2)
        %COMPLEMENTA(%MEM2)
        )
```

```
%' '
%*DEFINE(COMPLEMENTA(MEM))
%' '
%'  faz o complemento para 2 do número (MEM) '
%' '
        (
         MOV    A,%MEM
         CPL    A
         MOV    %MEM,A
         MOV    A,%MEM+1
         CPL    A
         MOV    %MEM+1,A    %' faz complemento para 1 '
         CLR    A
         SETB   C
         ADDC   A,%MEM
         MOV    %MEM,A
         MOV    A,#0
         ADDC   A,%MEM+1
         MOV    %MEM+1,A
         )
%' '
%' '
%*DEFINE(COPIA(MEM1,MEM2))
%' '
%' '
%'    (MEM1) = (MEM2)
%' '
        (
        MOV     %MEM1,%MEM2
        MOV     %MEM1+1,%MEM2+1
        )
%' '
%' '
%' '
```

```
%*DEFINE(COMPARA(MEM1,MEM2)) LOCAL CMP1 CMP2 CMP3 CMP4 CMP5 CMP6
%' '
%' '
%'  faz a comparação em módulo dos números Bx em R3 e R4 '
%'  e By em R5 e R6 activando a variável booleana FCMP se '
%'   Bx > By ,não altera Bx nem By
%' '
%' '
        (
        CLR     BITCX       %'flag de Bx complementado '
        CLR     BITCY       %'flag de By complementado '
        CLR     FCMP        %'flag de saída            '
        MOV     A,%MEM1+1
        JNB     ACC.7,%CMP1       %'o número já é positivo '
        SETB    BITCX
        %COMPLEMENTA(%MEM1)
%CMP1:
        MOV     A,%MEM2
        JNB     ACC.7,%CMP2       %'o número já é positivo '
        SETB    BITCY
        %COMPLEMENTA(%MEM2)
%CMP2:
        CLR     C
        MOV     A,%MEM2+1
        SUBB    A,%MEM1+1
        JC      %CMP3       %' Bx > By   '
        JNZ     %CMP4       %' Bx < By   '
        MOV     A,%MEM2
        SUBB    A,%MEM1
        JNC     %CMP4       %'Bx <= By    '
%CMP3:
        SETB    FCMP        %' sai com indicação de Bx > By '
%CMP4:
        JNB     BITCX,%CMP5    %' Bx não foi complementado  '
```

```
        %COMPLEMENTA(%MEM1)
%CMP5:
        JNB     BITCY,%CMP6     %' By não foi complementado  '
        %COMPLEMENTA(%MEM2)
%CMP6:
        )
%' '
%' '
%' '
%' '
%'      macros utilizando quatro bytes  '
%' '
%' '
%*DEFINE(SOMAR(BUFX,BUFY))  LOCAL SST1
%' '
%' soma (BUFX) = (BUFX) + (BUFY) '
%' '
        (
        MOV     R2,#4
        CLR     C
        MOV     R0,#%BUFX
        MOV     R1,#%BUFY
%SST1:
        MOV     A,@R0
        ADDC    A,@R1
        MOV     @R0,A
        INC     R0
        INC     R1
        DJNZ    R2,%SST1
        )
%' '
%' '
```

```
%' '
%*DEFINE(COMPLEMENTAR(BUFX)) LOCAL SCT2
%' '
%'          complementa para dois um número em 4 bytes '
%' '
        (
        MOV     R2,#4
        SETB    C
        MOV     R0,#%BUFX
%SCT2:
        MOV     A,@R0
        CPL     A
        ADDC    A,#0
        INC     R0
        DJNZ     R2,%SCT2
        )
%' '
%' '
%*DEFINE(SUBTRAIR(BUFX,BUFY))
%' '
%' '
%'      (BUFX) = (BUFX) - (BUFY)   '
%'                                 '
        (
        %COMPLEMENTAR(%BUFY)
        %SOMAR(%BUFX,%BUFY)
        )
%' '
%' '
%*DEFINE(COMPARAR(BUFX,BUFY)) LOCAL ST3 ST4 ST5 ST6 ST7 ST8 ST9
%' '
%' '
%'  compara (BUFX) com (BUFY) activa FCMP se (BUFX) > (BUFY) '
%' '
%' '
```

```
%' '
        (
        CLR     BITCX
        CLR     BITCY
        CLR     FCMP
        MOV     A,%BUFX+3
        JNB     ACC.7,%ST3
        SETB    BITCX
        %COMPLEMENTAR(%BUFX)
%ST3:
        MOV     A,%BUFY+3
        JNB     ACC.7,%ST4
        SETB    BITCY
        %COMPLEMENTAR(%BUFY)
%' '
%ST4:
        MOV     R2,#4
        MOV     R0,#%BUFX+3
        MOV     R1,#%BUFY+3
        CLR     C
%ST5:
        MOV     A,@R0
        SUBB    A,@R1
        JNC     %ST6
        JNZ     %ST7
        DEC     R0
        DEC     R1
        DJNZ    R2,%ST5
        JMP     %ST7
%ST6:
        SETB    FCMP    %' em 1 se (BUFX) > (BUFY) '
%ST7:
        JNB     BITCX,%ST8
        %COMPLEMENTAR(%BUFX)
%ST8:
```

```
        JNB     BITCY,%ST9
        %COMPLEMENTAR(%BUFY)
%ST9:
        )
%' '
%' '
%*DEFINE(LIMPAR(BUFX))  LOCAL SLP1
%' '
%'  faz o buffer = 0 '
%' '
        (
        MOV     R2,#4
        MOV     R0,#%BUFX
%SLP1:
        MOV     @R0,#0
        INC     R0
        DJNZ    R2,%SLP1
        )
%' '
%' '
%*DEFINE(MULTIPLICAR(BUFX,KM)) LOCAL SLMT1 SLMT2 SLMT3
%' '
%' '
%' multiplica um número de 4 bytes (BUFX) por (KM)   '
%'  KM e um número de oito bits '
%' '
%' '
        (
        CLR     BITCX
        MOV     A,#%BUFX+3
        JNB     ACC.7,%SLMT1
        SETB     BITCX
        %COMPLEMENTAR(%BUFX)
%SLMT1:
        %LIMPAR(BUFAUX)
```

```
        MOV     R2,#4
        CLR     C
        MOV     CARRY,C
        MOV     R0,#%BUFX
        MOV     R1,#BUFAUX
%SLMT2:
        MOV     A,%KM
        MOV     B,A
        MOV     A,@R0
        MUL     AB
        MOV     C,CARRY
        ADDC    A,@R1
        MOV     @R1,A
        MOV     A,B
        INC     R1
        ADDC    A,@R1
        MOV     CARRY,C
        MOV     @R1,A
        INC     R0
        DJNZ    R2,%SLMT2
%' '
        %COPIAR(%BUFX,BUFAUX)
        JNB     BITCX,%SLMT3
        %COMPLEMENTAR(%BUFX)
%SLMT3:
        )
%' '
%' '
%*DEFINE(COPIAR(BUFX,BUFY)) LOCAL SCP1
%' '
%' '
        (
        MOV     R2,#4
        MOV     R0,#%BUFX
        MOV     R1,#%BUFY
```

```
%SCP1:
        MOV     A,@R1
        MOV     @R0,A
        INC     R0
        INC     R1
        DJNZ    R2,%SCP1
        )
%' '
%' '
%' '
%*DEFINE(TRDIV(BUFX,VAR)) LOCAL TD1
%' '
%' '
%'    copia para (BUFX) (VAR) executando uma  '
%'    multiplicação por 256   '
%' '
        (
        %LIMPAR(%BUFX)
        MOV     %BUFX+1,%VAR
        MOV     %BUFX+2,%VAR+1
        MOV     A,%BUFX+2
        JNB     ACC.7,%TD1
        MOV     %BUFX+3,#0FFH
%TD1:
        )
%' '
%' '
%*DEFINE(TRDDIV(BUFX,VAR)) LOCAL TDD1
%' '
%' '
%'       cópia para (BUFX) (VAR) executando uma   '
%'       multiplicação 256 * 256     '
%' '
       (
       %LIMPAR(%BUFX)
```

```
        MOV     %BUFX,%VAR
        MOV     %BUFX+1,%VAR+1
        MOV     A,%BUFX+1
        JNB     ACC.7,%TDD1
        MOV     %BUFX+2,#0FFH
        MOV     %BUFX+3,#0FFH
%TDD1:
        )
%' '
%' '
%*DEFINE(SAIDA(BUFX)) LOCAL SLS1 SLS2
%' '
        (
        MOV     A,%BUFX+3
        MOV     C,ACC.7
        MOV     A,%BUFX+2
        JNC     %SLS1
        CPL     A
        SETB    C
        ADDC    A,#0
        CPL     A
        ORL     A,#80H
        JMP     %SLS2
%SLS1:
        CPL     A
        ANL     A,#7FH
%SLS2:
        MOV     DPTR,#PORT_DIG
        MOVX    @DPTR,A
        )
%' '  ;
```

```
;
;
;                    ************************************
;                    *                                  *
;                    *          POSIÇÃO                 *
;                    *            COM                   *
;                    *     CONTROLO DE VELOCIDADE       *
;                    *                                  *
;                    ************************************
;
;
;
$INCLUDE(DCX51A.EXT)
$INCLUDE(DCX51A.LIT)
$INCLUDE(ROBOT2.DCL)
$INCLUDE(CPLM32.MAC)
$INCLUDE(DCX51A.MAC)
;
      CSEG   AT  0FFF0H
;
;
ITD1:
      %ITD(INC_TASK1,8,81H,0111B,03H,00000B,ITD2)
                  ;
                  ;INC_TASK1 tarefa de comunicação
                  ;stack 4 bytes
                  ;identificador da tarefa
                  ;banco de registos 3
                  ;prioridade 3
                  ;não atribuída qualquer interrupção
                  ;IDD
;
;
;
;
```

```
;
        CSEG    AT    0FE00H
;
;
ITD2:
          %ITD(INC_TASK2,6,82H,0110B,02H,00000B,ITD3)
                    ;
                    ;INC_TASK2 endereço do inicio da tarefa 1
                    ;reserva de espaço para stack, 6 bytes
                    ;indentificador da tarefa
                    ;escolhido o banco de registos 2
                    ;escolhida a prioridade 2
                    ;não atribuída qualquer interrupção
;
;
;
ITD3:
          %ITD(INC_TASK3,4,82H,0101B,01H,00100B,ITD4)
                    ;
                    ;INC_TASK3 endereço do início da tarefa 3
                    ;reserva de espaço para stack, 4 bytes
                    ;indentificador da tarefa
                    ;escolhido o banco de registos 1
                    ;escolhida a prioridade 1
                    ;atribuída a interrupção IE0
;
;
ITD4:
          %ITD(INC_TASK4,4,84H,0101B,01H,00001B,IDD)
                    ;
                    ;INC_TASK4 endereço do início da tarefa 4
                    ;reserva de espaço para stack, 4 bytes
                    ;indentificador da tarefa
                    ;escolhido o banco de registos 1
                    ;escolhida a prioridade 1
```

```
                    ;atribuída a interrupção IE1
;
;
IDD:
        %IDD(-1000,0,0,30H,0F800H)
                    ;
                    ;mantém o período do relógio do sistema
                    ;não muda a prioridade do relógio de sistema
                    ;mantém buffer de de comunicações em 20 bytes
                    ;reserva 40 bytes de memoria para o utilizador
                    ;termina a cadeia ITD/IDD
;
;
;
```

```
;
;                 ************************************
;                 *                                  *
;                 *      TAREFA DE COMUNICAÇÃO       *
;                 *                                  *
;                 ************************************
;
;
      CSEG      AT 0C000H
;
;
INC_TASK1:
                        ;tarefa de comunicação
                        ;aguarda comunicação do master,
                        ;executa e responde
                        ;
          MOV       B,#WAIT_FOREVER     ;timeout infinito
          MOV       A,#MESSAGE_EVENT    ;aguarda mensagem
          SETB      TESTE1
          CALL      RQWAIT
          MOV       CMND_RSP,#0         ;resposta OK
          MOV       MSG_PTR,R7
          MOV       A,R7
          ADD       A,#8                ;avança o cabeçalho
          MOV       R0,A
          MOV       A,@R0               ;comando
 ;
 ;                    descodificação do comando
 ;
 ;
          CJNE      A,#AVNC_NRML,S7T53
          JMP       S7T52
 S7T53:
          JMP       S7T2
 ;          avança para a posição pedida
```

```
;
S7T52:
        MOV     CMND_RSP,#0     ;comando reconhecido
        INC     R0              ;salta flags
        INC     R0
        MOV     VREFT,@R0
        INC     R0
        MOV     VREFT+1,@R0     ;actualiza velocidade
        INC     R0
        MOV     YREFT,@R0
        INC     R0
        MOV     YREFT+1,@R0     ;actualiza a próxima posição
;
;       verifica limites
;
        MOV     A,YREFT+1
        JB      ACC.7,S7T47     ;não ultrapassa limite superior
        MOV     LIMIT1,#EX_ALTL ;limite superior
        MOV     LIMIT1+1,#EX_ALTH
        %COMPARA(YREFT,LIMIT1)
        JNB     FCMP,S7T47
        JMP     S7T41           ;excede limite superior
S7T47:
        MOV     A,YREFT+1
        JNB     ACC.7,S7T42
        MOV     LIMIT1,#EX_INFL ;limite inferior
        MOV     LIMIT1+1,#EX_INFH
S7T48:
        %COMPARA(YREFT,LIMIT1)
        JNB     FCMP,S7T42
        JMP     S7T41           ;excede limite
S7T42:
        MOV     A,VREFT+1
        JB      ACC.7,S7T43     ;velocidade em módulo
        %COMPLEMENTA(VREFT)
```

```
S7T43:
        MOV     LIMIT1,#LIMVL
        MOV     LIMIT1+1,#LIMVH
        %COMPARA(VREFT,LIMIT1)
        JB      FCMP,S7T44      ;velocidade excede limite
;
;                   confirma os valores da velocidade
;                   e posição se estes não ultrapassam os
;                   limites estabelecidos
      %COPIA(VREF,VREFT)
        %COPIA(YREF,YREFT)
        JMP     S7T21
;
S7T41:                          ; erro de posição
        MOV     A,CMND_RSP
        ORL     A,#00000010B    ;excede limite de posição
        MOV     CMND_RSP,A
        JMP     S7T21
;
S7T44:
                                ;erro de velocidade
        MOV     A,CMND_RSP
        ORL     A,#00000100B    ;excede limite de velocidade
        MOV     CMND_RSP,A

S7T21:
        JMP     S7T1
;
;         comando início, cria uma tarefa que faz
;         o eixo avançar até encontrar a referência,
;         inicializa os contadores e  faz
;         avanço até atingir a posição de repouso
S7T2:
        MOV     A,@R0
        CJNE    A,#POS_INIC,S7T31
```

```
        MOV     CMND_RSP,#0             ;comando reconhecido
        MOV     DPTR,#ID_TSK5
        CALL    RQCREATETASK        ;tarefa de inicialização
        JMP     S7T1
;
;
;       comando de avanço a velocidade constante
;       recebe como parâmetro a velocidade.
;       Se a velocidade pedida é negativa
;       YREF = limite inferior, se a velocidade pedida
;       é positiva YREF = limite superior.
;       A velocidade muda para módulo e actualiza VREF
;
S7T31:
        CJNE    A,#VELOCI,S7T38     ;comando de avanço
        JMP     S7T39
S7T38:
        JMP     S7T3
S7T39:
        MOV     CMND_RSP,#0         ;comando reconhecido
        INC     R0
        INC     R0
        MOV     VREFT,@R0       ;byte mais baixo da velocidade
        INC     R0
        MOV     VREFT+1,@R0
;
        MOV     LIMIT1,#LIMVL
        MOV     LIMIT1+1,#LIMVH
        %COMPARA(VREFT,LIMIT1)
        JNB     FCMP,S7T32
        %COPIA(VREF,VREFT)
        MOV     YREF,#EX_ALTL
        MOV     YREF+1,#EX_ALTH ;extremo superior do eixo
        MOV     A,VREF+1
        JNB     ACC.7,S7T32
```

```
        MOV     YREF,#EX_INFL   ;se velocidade negativa
        MOV     YREF+1,#EX_INFH ;posição = extremo inferior
        JMP     S7T33
;
S7T32:
        MOV     A,CMND_RSP
        ORL     A,#0000100B     ; erro
        MOV     CMND_RSP,A      ;excesso de velocidade pedida
S7T33:
        JMP     S7T1
;
;
;
;
S7T3:
        MOV     A,@R0
        CJNE    A,#FAZER_NADA,S7T81
;                       este comando destina-se a permitir
;                       a leitura do estado actual do eixo
        MOV     CMND_RSP,#0
        JMP     S7T1
;
ST781
        MOV     A,@R0
        CJNE    A,#PARA,S7T51
        MOV     VREF,#0
        MOV     VREF+1,#0
        JMP     S7T1
;
;                   erro, comando desconhecido
S7T51:
        MOV     CMND_RSP,#1
;
S7T1:
        MOV     A,MSG_PTR
```

```
        ADD     A,#3            ;
        MOV     R0,A
        MOV     A,#80H
        ORL     A,@R0
        MOV     @R0,A           ;muda para resposta          V[3]
;
        INC     R0              ;enderço de nó inalterado V[4]
        INC     R0              ;endereço  inalterado        V[5]
        INC     R0
        MOV     @R0,#RESP_OK    ;resposta no cabeçalho       V[6]
        INC     R0                   ;endereço
        INC     R0
        MOV     @R0,CMND_RSP    ;sucesso ou não da tarefa V[7]
        INC     R0
        MOV     @R0,#STATUS     ;estado do eixo              V[8]
        INC     R0
        MOV     @R0,VLACT  ;velocidade actual                V[9]
        INC     R0
        MOV     @R0,VLACT+1  ;                               V[10]
        INC     R0
        MOV     @R0,YACT     ;posição actual                 V[11]
        INC     R0
        MOV     @R0,YACT+1   ;                               V[12]
        MOV     DPH,#0
        MOV     DPL,MSG_PTR     ;buffer de resposta
        CALL    RQSENDMESSAGE   ;reenvia a mensagem ao master
;
;
      JMP       INC_TASK1       ;tarefa terminada
;
;
```

```
;
;
;           ****************************************
;           *                                      *
;           *      LEITURA DE ENTRADAS             *
;           *           E                          *
;           *      CONTROLO DE POSIÇÃO             *
;           *                                      *
;           ****************************************
;
INC_TASK2:
;                   bloco de inicialização
;
;
        MOV     DPTR,#PORT_DIG      ;endereço da saída
        MOV     A,#0
        MOVX    @DPTR,A             ;bloqueia motor
        MOV     DPTR,#PRT_RES       ;porta de reset
        MOV     A,#01H
        MOVX    @DPTR,A             ;faz reset dos contadores
        MOV     A,#0FFH
        MOVX    @DPTR,A
        MOV     A,#02H
        MOVX    @DPTR,A
        MOV     A,#0FFH
        MOVX    @DPTR,A             ;em modo slave
        MOV     DPTR,#PRT_POS
        MOV     A,#YINICL           ;valor inicial dos contadores
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#YINICH
        MOVX    @DPTR,A           ;inicializa contadores, teste
        MOV     A,#0H
        MOV     YREF,A
        MOV     YREF+1,A
```

```
        MOV     VREF,A
        MOV     VREF+1,A
        MOV     VLACT,A
        MOV     VLACT+1,A
        MOV     YACT,A
        MOV     YACT+1,A
        MOV     TMP1,#0FH
        MOV     INTGR,A
        MOV     INTGR+1,A
        MOV     YP,#0
        MOV     YP+1,#0
        MOV     KD,#CONSTKD       ;valor inicial para KD
        MOV     KI,#CONSTKI       ;valor inicial para KI
        MOV     A,#TMPR2          ;período de repetição
        CALL    RQSETINTERVAL     ;primitiva do sistema
;
SLTASK11:
;
        MOV     A,#INTERVAL_EVENT
        MOV     B,#WAIT_FOREVER
        CALL    RQWAIT            ;espera o tempo definido
;
;                   leitura dos contadores
;
        MOV     DPTR,#PRT_RES  ;porta de controlo dos contadores

        MOV     A,#02          ;bloqueia latch
        MOVX    @DPTR,A
        MOV     A,#0FFH
        MOVX    @DPTR,A        ;latch na posição normal
        MOV     DPTR,#PRT_POS  ;para ler porta de posição
        MOVX    A,@DPTR        ;byte mais signf. de posição
        MOV     TEMP1+1,A
        INC     DPTR
        MOVX    A,@DPTR
```

```
        MOV         TEMP1,A         ;byte menos signf. de posição
        MOV         LIMIT2,#OFFSETL     ;offset dos contadores
        MOV         LIMIT2+1,#OFFSETH  ;para deslocar o zero
        %SOMA(TEMP1,LIMIT2)
        MOV         LIMIT2,YACT         ;neste momento com
        MOV         LIMIT2+1,YACT+1     ;a posição anterior
        MOV         YACT,TEMP1
        MOV         YACT+1,TEMP1+1     ;posição actual
;               velocidade = posição acual - posição anterior
;
        %SUBTRAI(TEMP1,LIMIT2)
        MOV         VLACT,TEMP1
        MOV         VLACT+1,TEMP1+1     ;velocidade actual
;
;
;
        MOV         DPTR,#PORT_DIG  ; contém informação
        MOVX        A,@DPTR         ;sobre fins de curso
        ORL         A,STATUS        ; e outras entradas
        MOV         STATUS,A
        CLR         FIM_CURSOL
        CLR         FIM_CURSOH
        MOV         A,#MSC_FC_L
        ANL         A,STATUS
        JZ          SLTASK13
        SETB        FIM_CURSOL
        JZ          SLTASK12
SLTASK13:
        MOV         A,#MSC_FC_H
        ANL         A,STATUS
        JZ          SLTASK12
        SETB        FIM_CURSOH
;
SLTASK12:
;
```

```
;
;           ************************************
;           *                                  *
;           *       MOVIMENTO DO EIXO          *
;           *                                  *
;           ************************************
;
;
;
;       copia o valor dos pedidos de velocidade e posição
;       por a tarefa de comunicação ter maior prioridade
;       que a de posição
;
        CLR     IE.7        ;desactiva todas as interrupções
        MOV     REFER,YREF
        MOV     REFER+1,YREF+1 ;copia próxima posição
        MOV     TMP1,VREF
        MOV     TMP1+1,VREF+1  ;copia pedido de velocidade
        SETB    IE.7        ;enable geral de interrupt
;
;
;        controlo de posição de um cilindro hidráulico,
;        o valor menos significativo de cada variável
;        ocupa a posição mais baixa de memoria. As variáveis
;        são representadas em complemento para 2
;
;        YREF  posição pedida dois bytes yref e yref+1
;        VREF  pedido de velocidade, dois bytes
;        YACT  posição actual dois bytes
;        VLACT  saida para a valvula um byte sinal e grandeza
;
;
;             a partir da referência de velocidade
;            é gerada uma rampa de pedidos de posição.
;             Este módulo adiciona ao último valor pedido,
```

```
;               a velocidade, com um período igual ao período
;               de amostragem. Assim é gerada uma rampa de
;               pedidos de posição com a inclinação dada pela
;               velocidade pedida. O próximo valor pedido
;               é feito igual a YREF logo que o iguale ou exceda
;
        %COPIAR(BUFAUX,YP)     ;yp contem o último valor pedido
        %TRDDIV(BUFFERX,VREF)
        %TRDDIV(BUFFERY,YREF)
        %SUBTRAIR(BUFAUX,BUFFERY) ;a distância para YREF
        %COMPARAR(BUFAUX,BUFFERX) ;se menor que VREF, objectivo
        JB      FCMP,SALTO0
        %COPIAR(BUFFERX,BUFFERY)  ;atingido
        JMP     SALTO2
SALTO0:
        MOV     A,BUFAUX+3
        JB     ACC.7,SALTO1
        %COPIAR(BUFAUX,YP)         ;se erro negativo
        %SUBTRAIR(BUFAUX,BUFFERX) ;subtrai a velocidade
        %COPIAR(BUFFERX,BUFAUX)   ;ao último pedido
        JMP    SALTO2
SALTO1:
        %COPIAR(BUFAUX,YP)         ;se negativa, soma
        %SOMAR(BUFAUX,BUFFERX)
        %COPIAR(BUFFERX,BUFAUX)
salto2:
        %COPIAR(YP,BUFFERX)      ;guarda o último valor pedido
;
;               módulo de controlo de posiçãg
;               tem como entradas a nova posição pedida
;               no BUFFERX e a posição actual em YACT
;
        %TRDDIV(BUFFERY,YACT)
        %SUBTRAIR(BUFFERX,BUFFERY)  ;erro de posição
        %MULTIPLICAR(BUFFERX,KI)    ;KI, constante integradora
```

```
        %SOMAR(INTGR,BUFFERX)        ;integra
        %COPIAR(BUFFERNB    BITCX,SCONTR2
        %COMPLEMENTAR(BUFFERX)       ;correcção do sinal
SCONTR2:
        %COPIAR(BUFFERY,BUFFERX)
        %SOMAR(BUFFERX,BUFAUX)
        %COPIAR(INTGR,BUFFERX)       ;restaura o integrador
        %COPIAR(BUFFERX,BUFFERY)
;
SCONTR3:
        %SAIDA(BUFFERX)              ;saída em sinal e grandesa
;
        JMP     SLTASK11
;
;
;
;
```

```
;               ****************************************
;               *                                      *
;               *           POSICIONAMENTO             *
;               *                                      *
;               *              INICIAL                 *
;               *                                      *
;               ****************************************
;
;
;
INC_TASK3:
;
;
          MOV     YREF,#EX_INFL
          MOV     YREF+1,#EX_INFH
          MOV     VREF,#VEL_MEDIA_L
          MOV     VREF+1,#VEL_MEDIA_H
TSK41:
          JNB     FIM_CURSOL,TSK41  ;avanca até extremo inf.
          MOV     YREF,#EX_ALTL
          MOV     YREF+1,#EX_ALTH
TSK42:
;              avança até atingir o fim de curso
;              da referência de zero.
          JNB     REF0,TSK42
          MOV     VREF,#VEL_BAIXA_L
          MOV     VREF+1,#VEL_BAIXA_H
          MOV     B,#WAT_FOREVER
          MOV     A,#INTERRUPT_EVENT
          CALL    RQWAIT
;     aguarda a interrupção associada à referência
;     de zero da régua
          CLR     IE.7
          MOV     DPTR,#PRT_POS
```

```
        MOV     A,#YINICL
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,#YINICH
        MOVX    @DPTR,A             ;valor dos contadores
        MOV     YREF,#HOMEL         ;posição de repouso
        MOV     YREF1,#HOMEH
        MOV     VREF,#VEL_MEDIA_L
        MOV     VREF+1,#VEL_MEDIA_H
        SET     IE.7
        MOV     A,#3
        CALL    RQDELETETASK
;
;
```

ANEXO B

PROGRAMAÇÃO DA PAL

PROGRAMAÇÃO DA PAL DA PLACA DE NÚCLEO

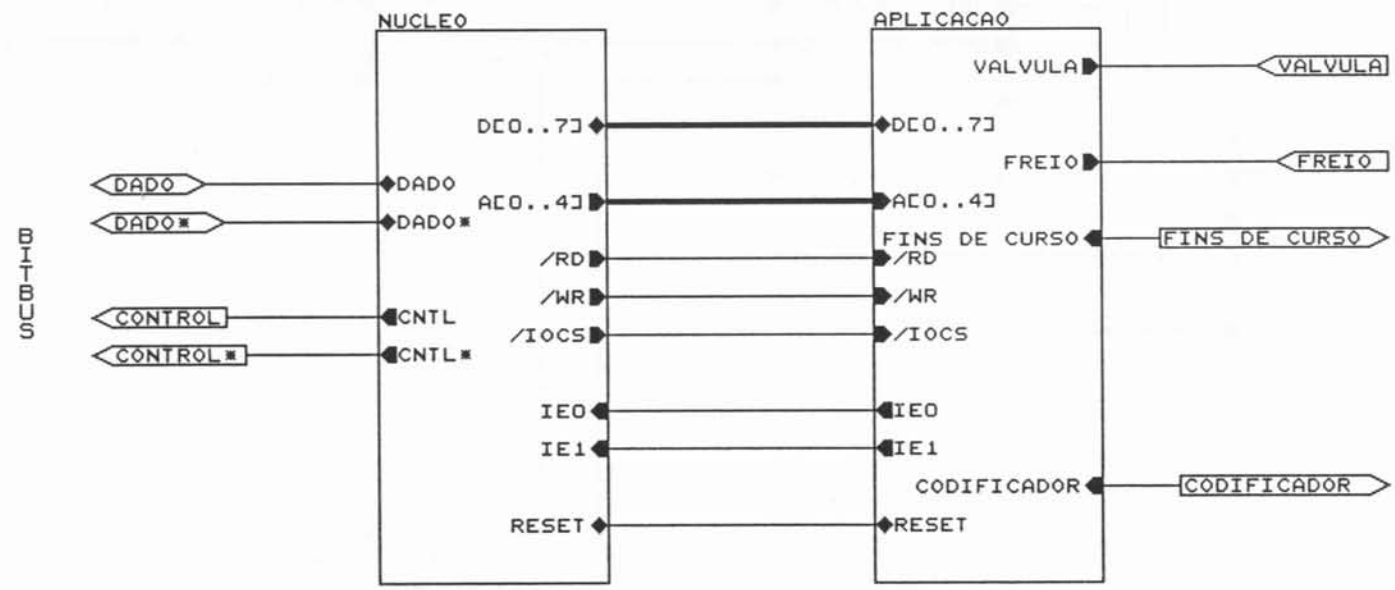
PAL DESIGN SPECIFICATION

PAL14L4

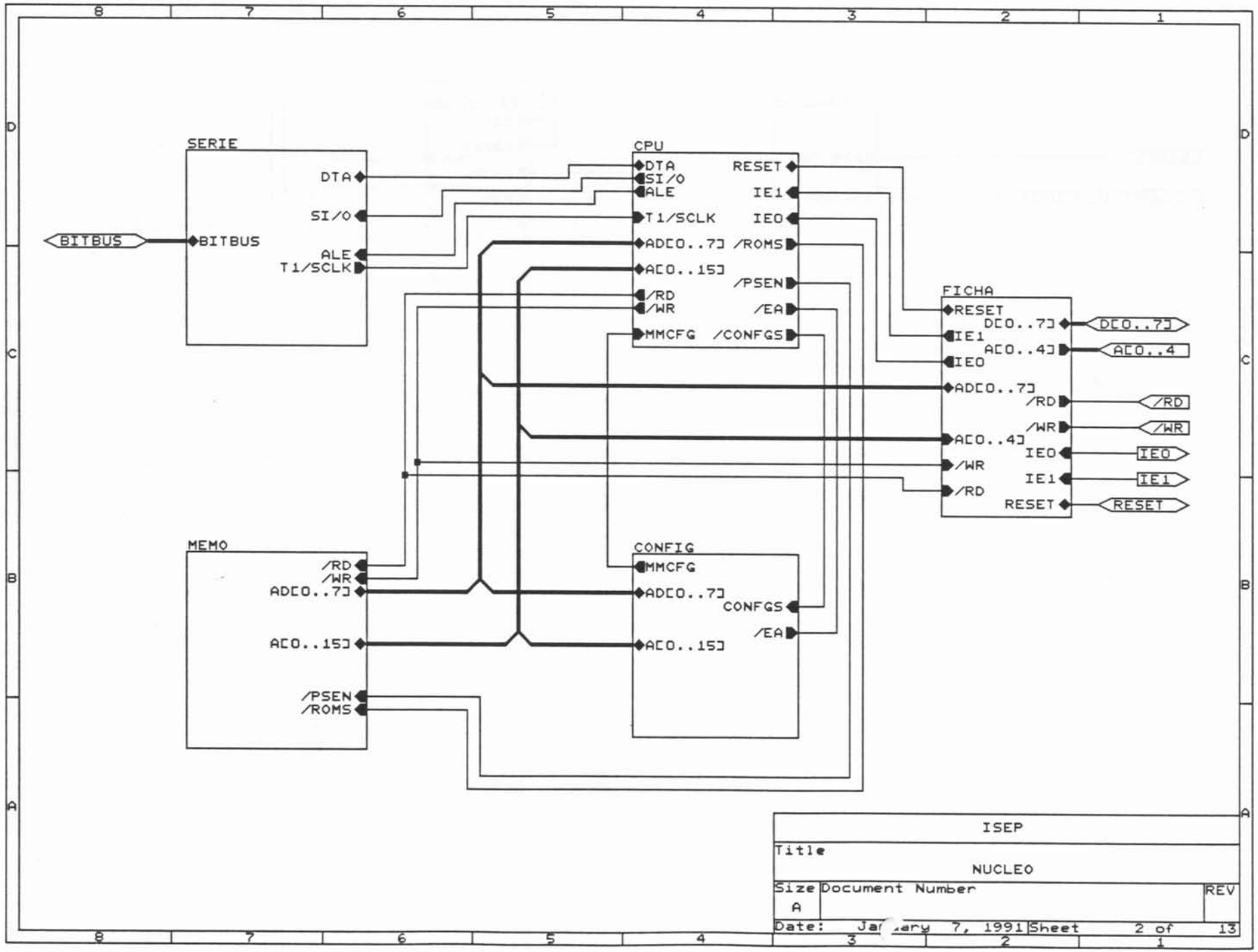
```
af ae ad ac ab aa a9 a8 a6 GND
a7 mmcfg io confgs roms iocs fifo a5 rd VCC

/fifo   =  af* ae* ad* ac* ab* aa* a9* a8* /a7* /a6* /a5
/iocs   =  af* ae* ad* ac* ab* aa* a9* a8* a7* a6* /a5
       +  af* ae* ad* ac* ab* aa* a9* a8* a7* /a6
       +  af* ae* ad* ac* ab* aa* a9* a8* /a7* a6
/roms   =  mmcfg
       +  af* /mmcfg* io
/confgs =  af* ae* ad* ac* ab* aa* a9* a8* a7* a6* a5* /rd

FUNCTION TABLE

af ae ad ac ab aa a9 a8 a6 a7 mmcfg io a5 rd ; INPUT
confs roms iocs fifo                         ; OUTPUT
```

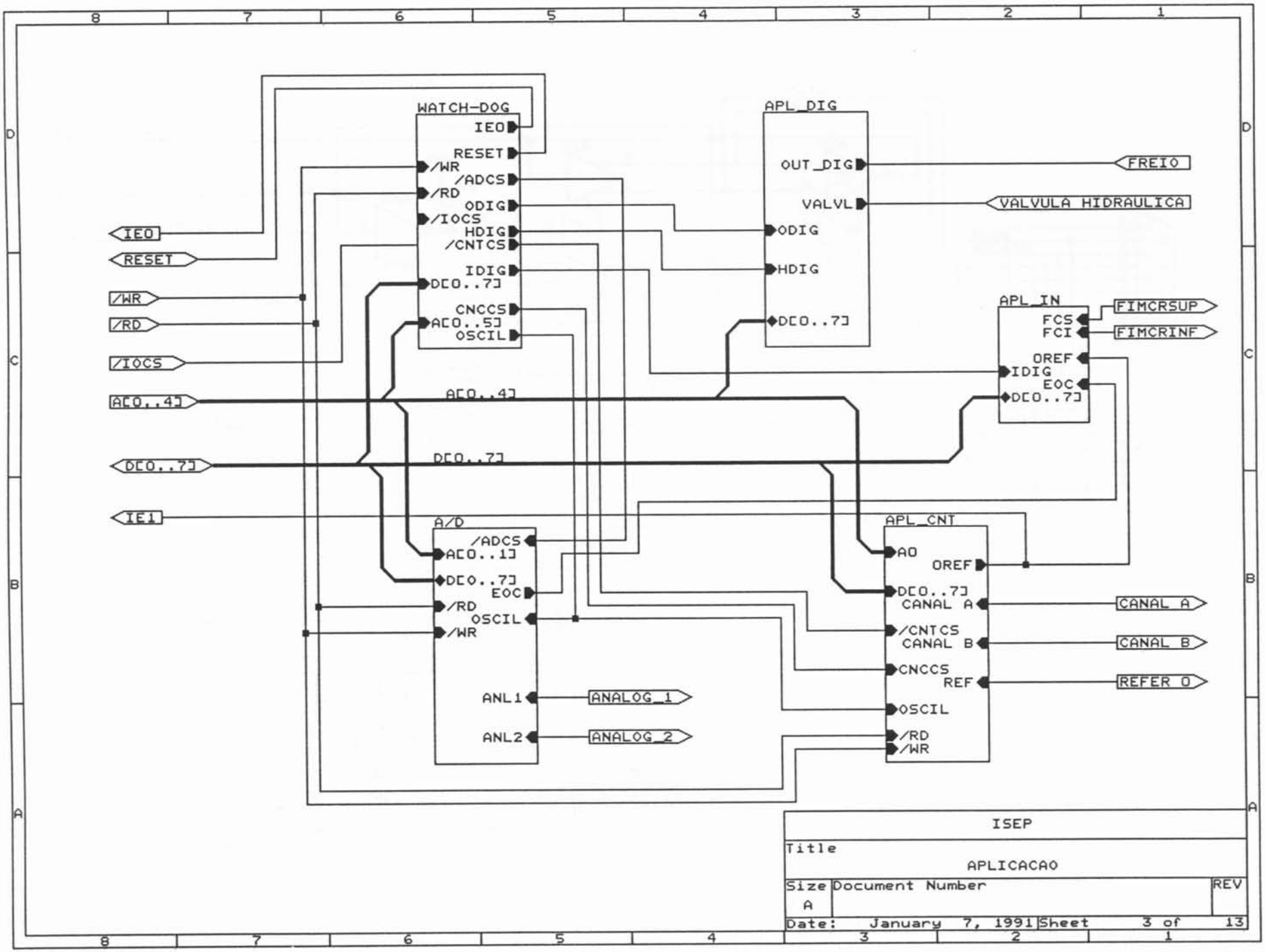
ANEXO C

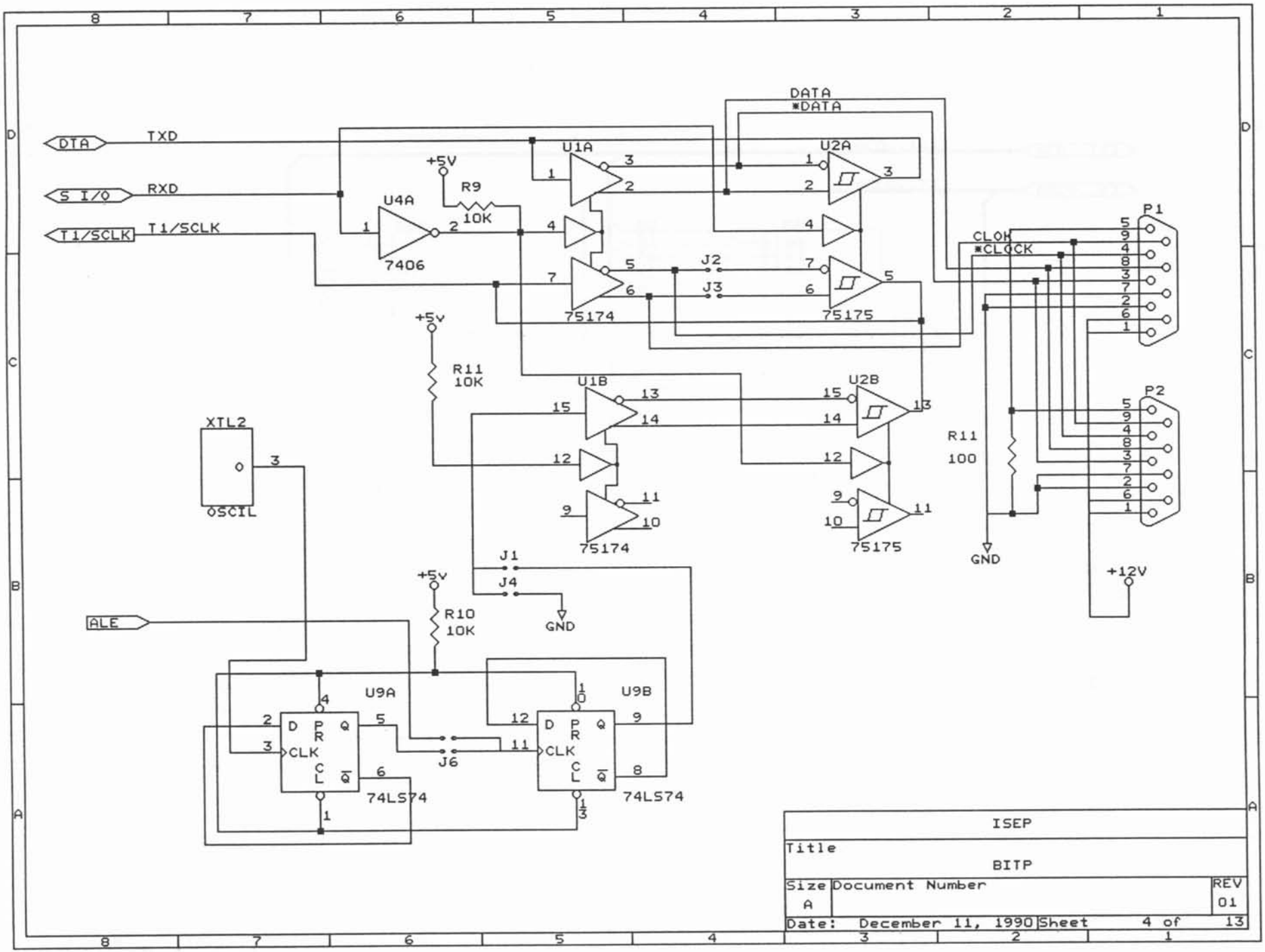
ESQUEMAS ELÉCTRICOS

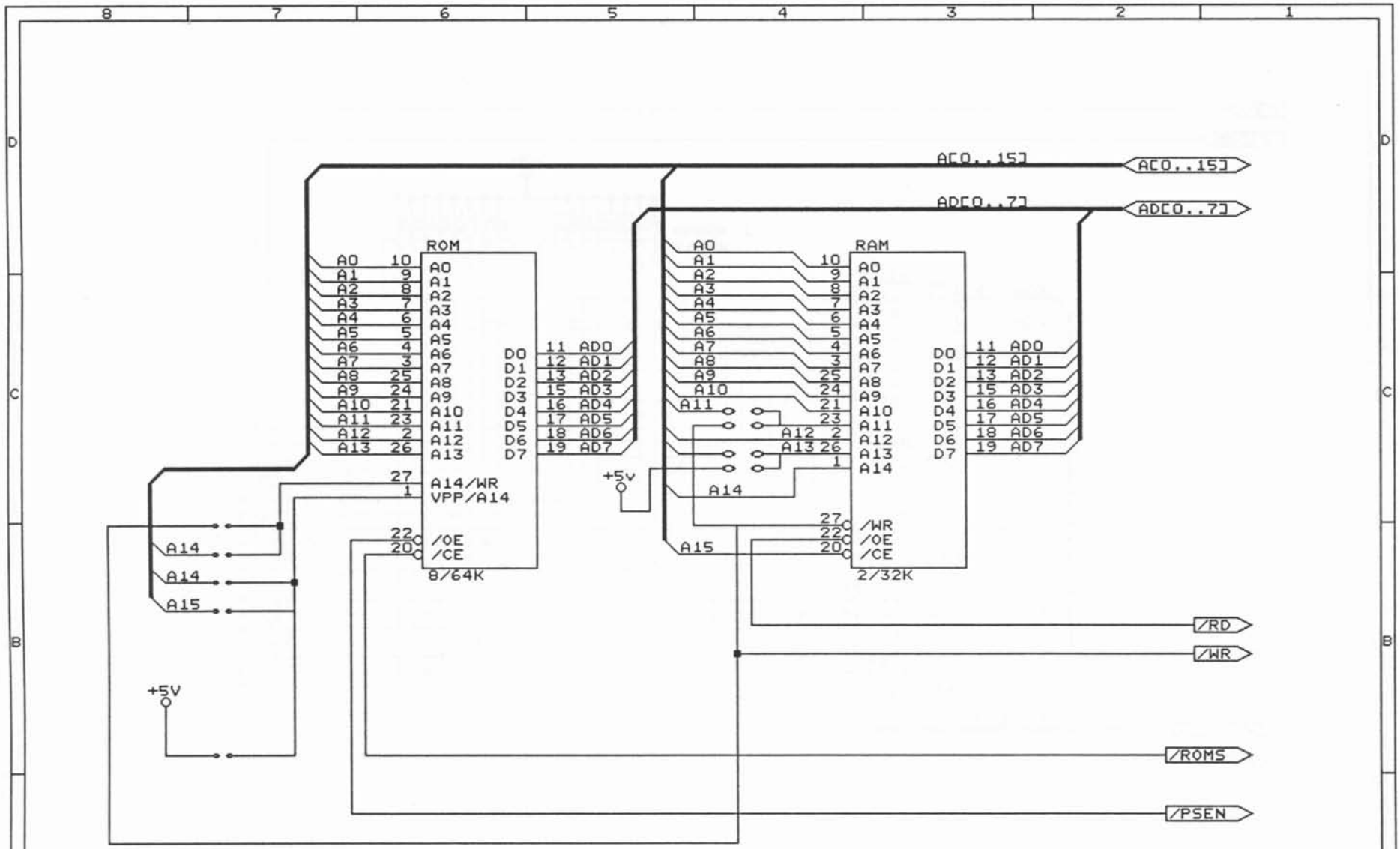


ISEP		
Title: RAIZ		
Size: A	Document Number	REV
Date: December 11, 1990	Sheet 1 of 13	

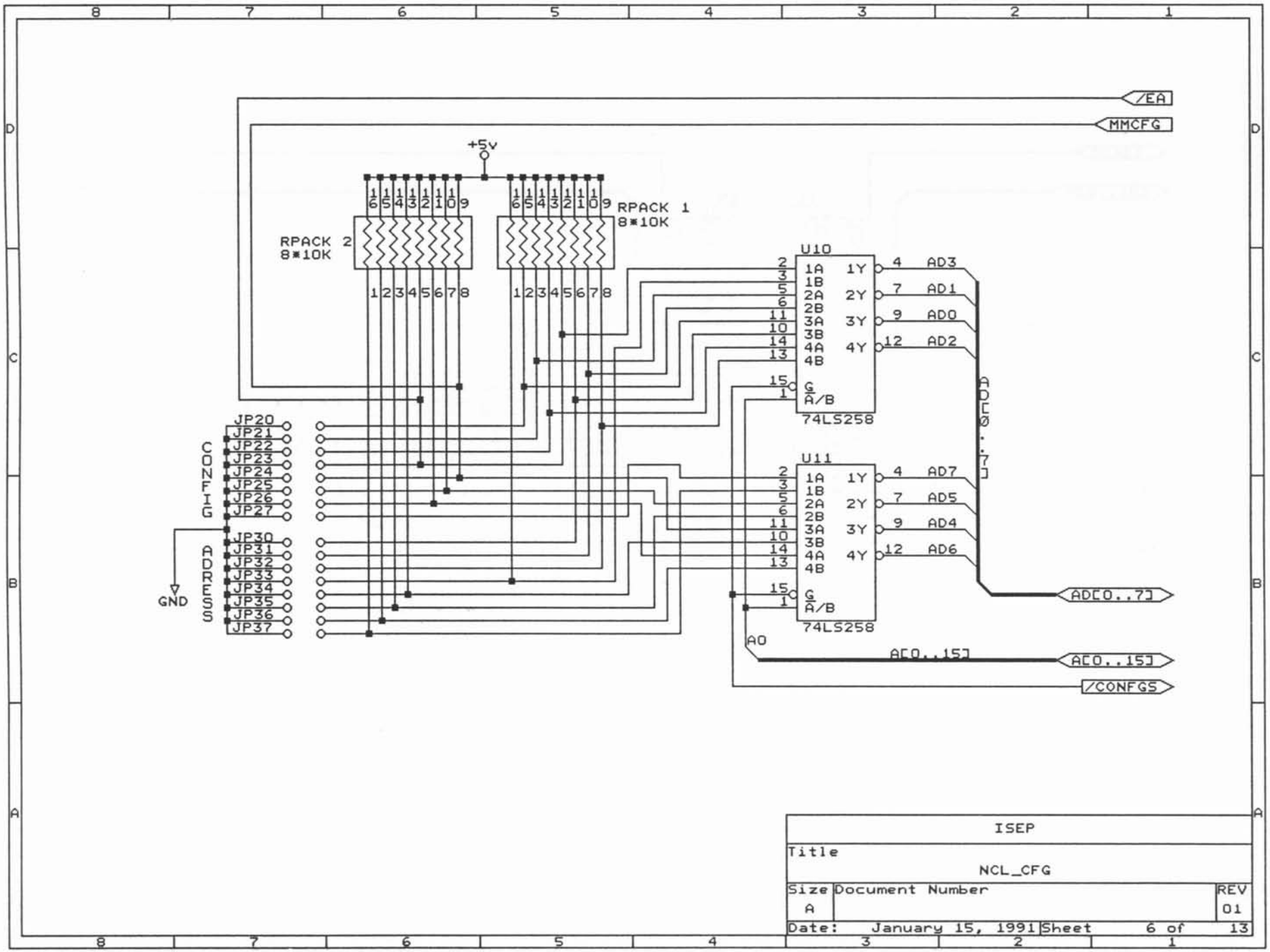

SERIE
DTA
SI/O
BITBUS
BITBUS
ALE
T1/SCLK
CPU
DTA
SI/O
ALE
T1/SCLK
AD[0..7]
A[0..15]
/RD
/WR
MMCFG
RESET
IE1
IEO
/ROMS
/PSEN
/EA
/CONFGS
FICHA
RESET
IE1
IEO
AD[0..7]
A[0..4]
/WR
/RD
D[0..7]
A[0..4]
/RD
/WR
IEO
IE1
RESET
MEMO
/RD
/WR
AD[0..7]
A[0..15]
/PSEN
/ROMS
CONFIG
MMCFG
AD[0..7]
A[0..15]
CONFGS
/EA
ISEP
Title
NUCLEO
Size
A
Document Number
REV
Date: January 7, 1991
Sheet 2 of 13


WATCH-DOG
IEO
RESET
/WR
/ADCS
/RD
ODIG
/IOCS
HDIG
/CNTCS
IDIG
DC0..7]
CNCCS
AC0..5]
OSCIL
APL_DIG
OUT_DIG
VALVL
ODIG
HDIG
DC0..7]
FREIO
VALVULA HIDRAULICA
APL_IN
FCS
FCI
OREF
IDIG
EOC
DC0..7]
FIMCRSUP
FIMCRINF
IEO
RESET
/WR
/RD
/IOCS
AC0..4]
DC0..7]
IE1
A/D
/ADCS
AC0..1]
DC0..7]
EOC
/RD
OSCIL
/WR
ANL1
ANL2
ANALOG_1
ANALOG_2
APL_CNT
A0
OREF
DC0..7]
CANAL A
/CNTCS
CANAL B
CNCCS
REF
OSCIL
/RD
/WR
CANAL A
CANAL B
REFER 0
ISEP
Title
APLICACAO
Size
A
Document Number
REV
Date: January 7, 1991
Sheet 3 of 13

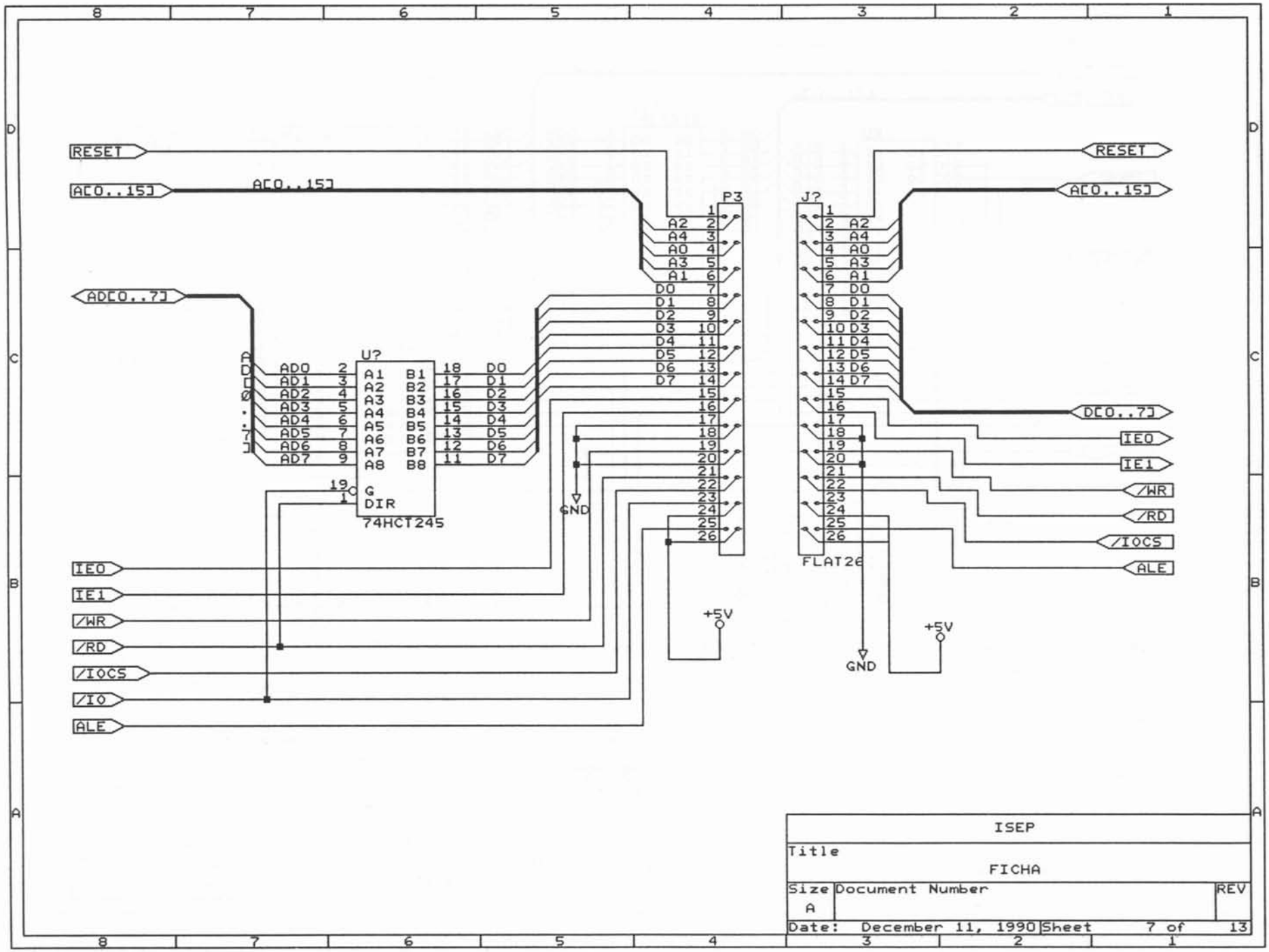

DTA
TXD
S I/O
RXD
T1/SCLK
T1/SCLK
+5V
R9
10K
U4A
7406
U1A
U2A
DATA
*DATA
J2
J3
75174
75175
CLOK
*CLOCK
P1
P2
+5V
R11
10K
U1B
U2B
75174
75175
R11
100
GND
+12V
XTL2
OSCIL
J1
J4
GND
+5V
R10
10K
ALE
U9A
U9B
J6
74LS74
74LS74
ISEP
Title
BITP
Size
A
Document Number
REV
01
Date: December 11, 1990
Sheet 4 of 13

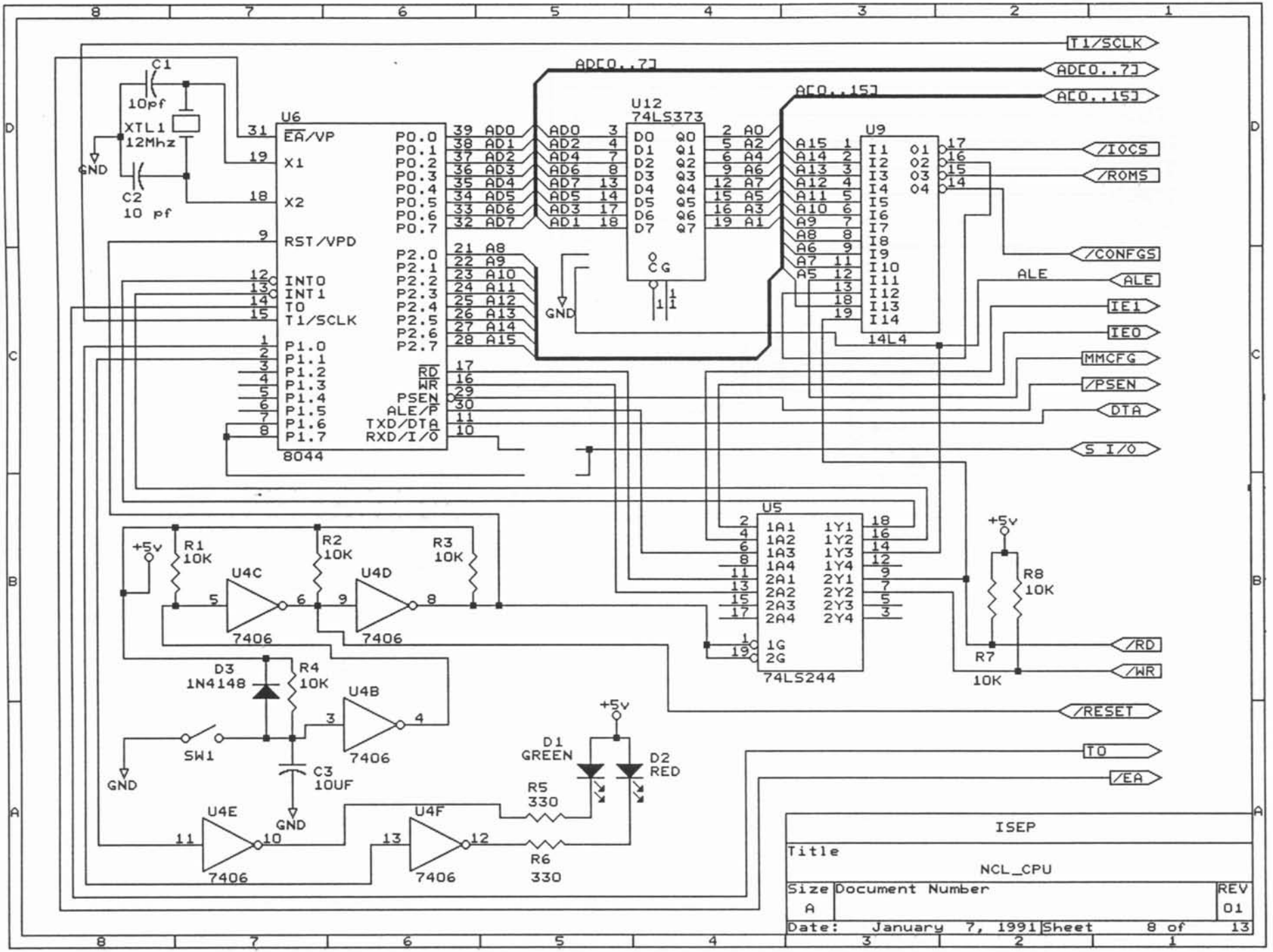


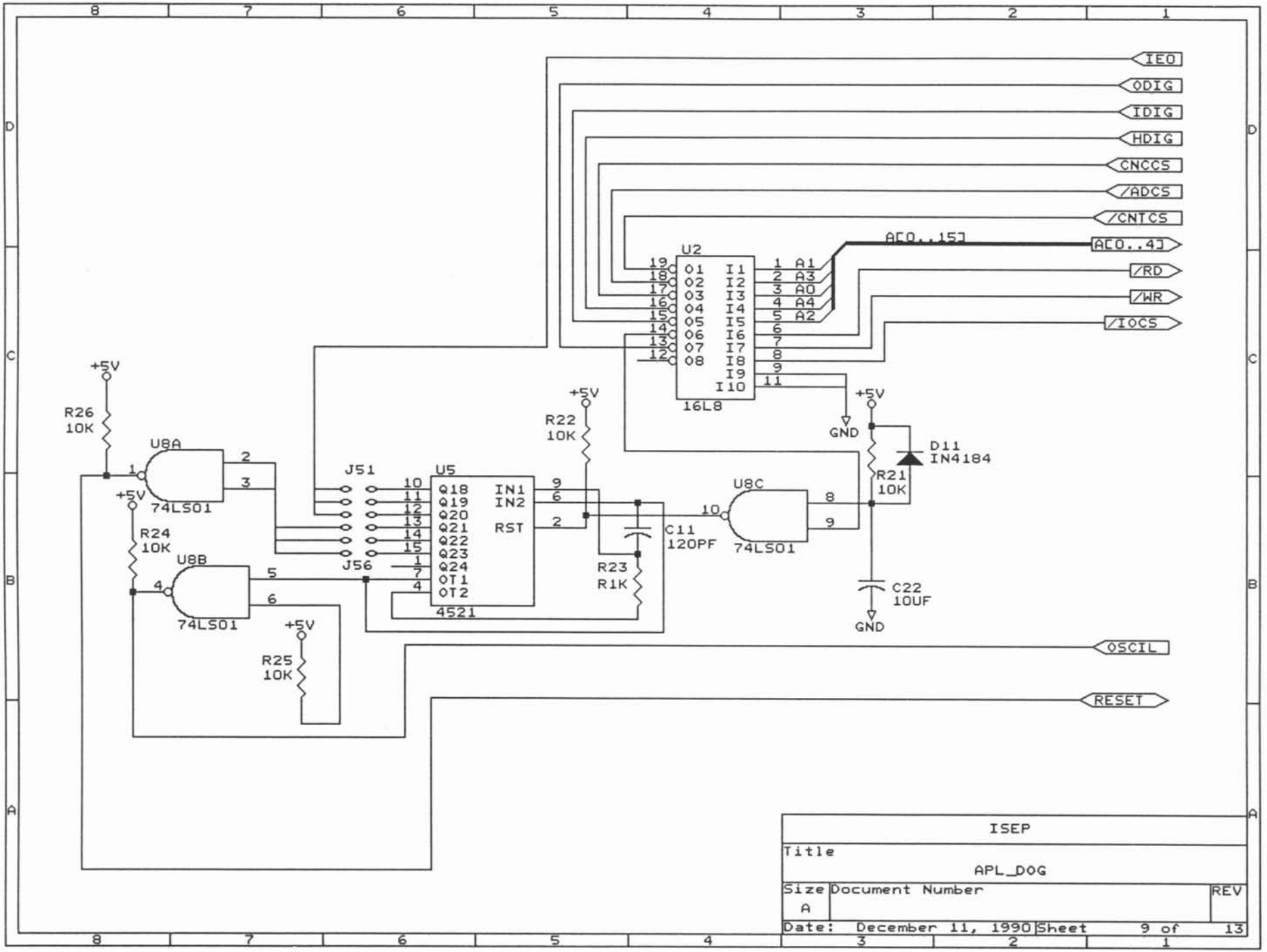
ISEP		
Title: MEMO		
Size: A	Document Number	REV
Date: December 16, 1990	Sheet 5 of 13	

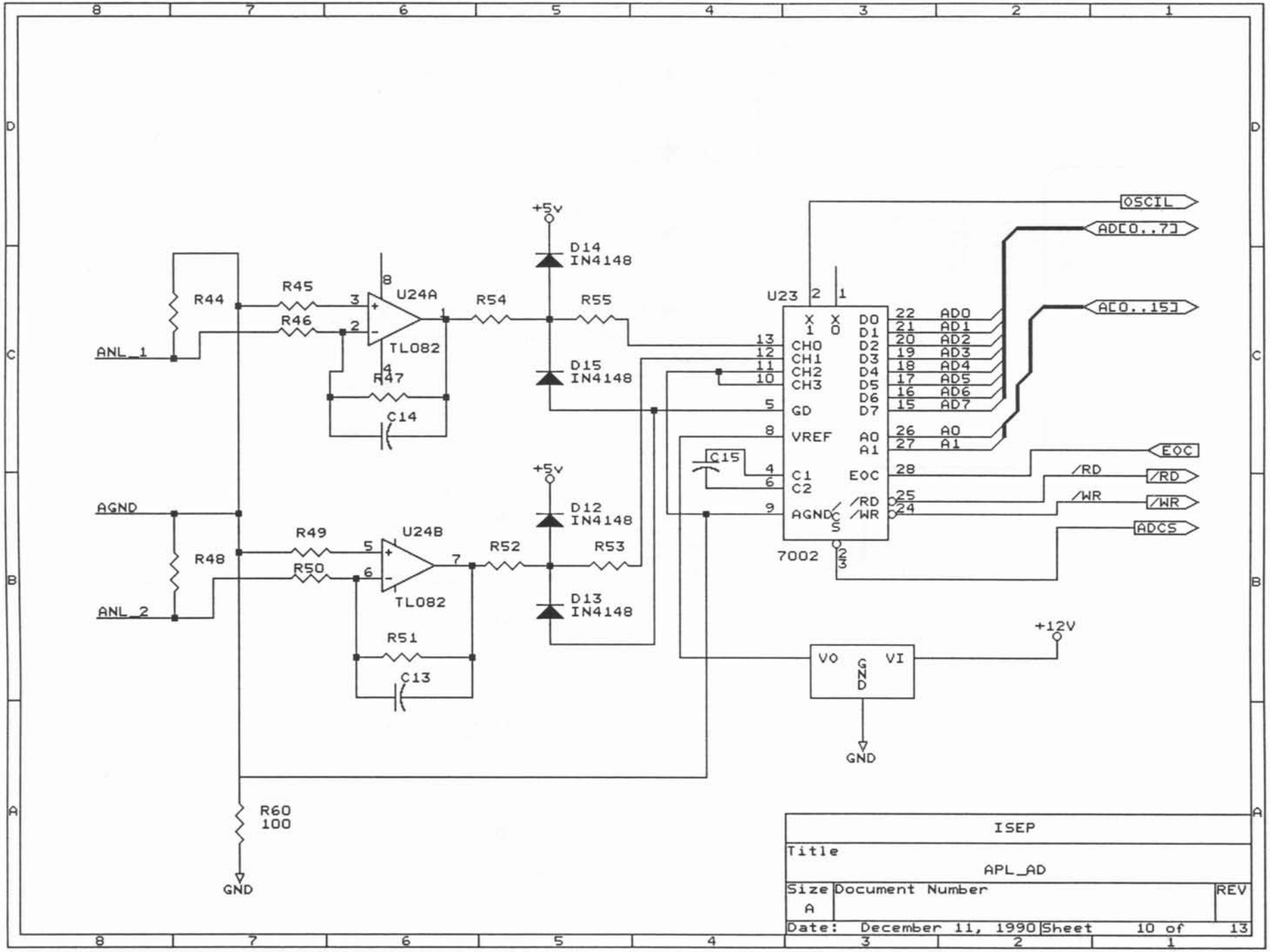


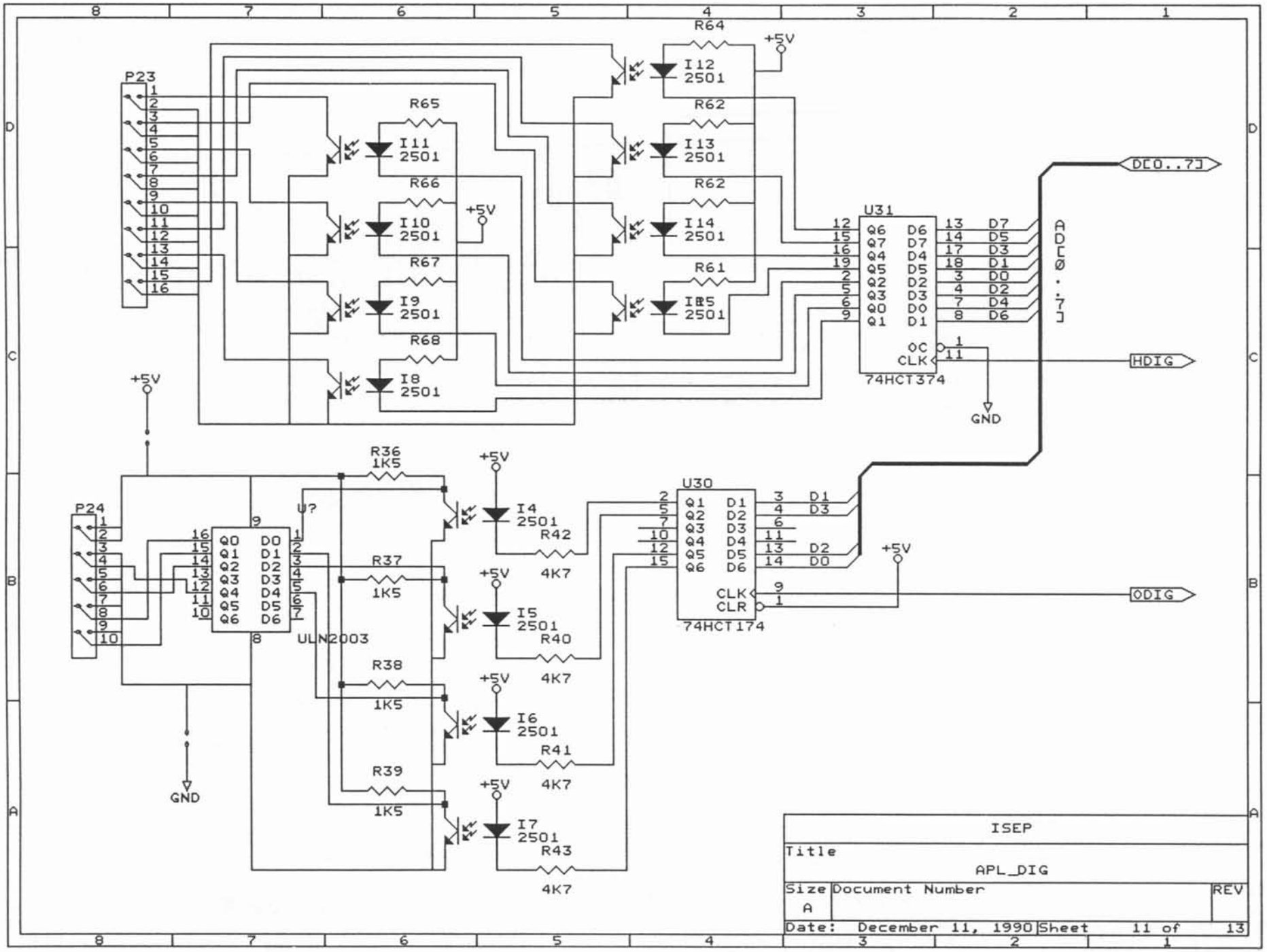
ISEP		
Title: NCL_CFG		
Size: A	Document Number:	REV: 01
Date: January 15, 1991	Sheet 6 of 13	

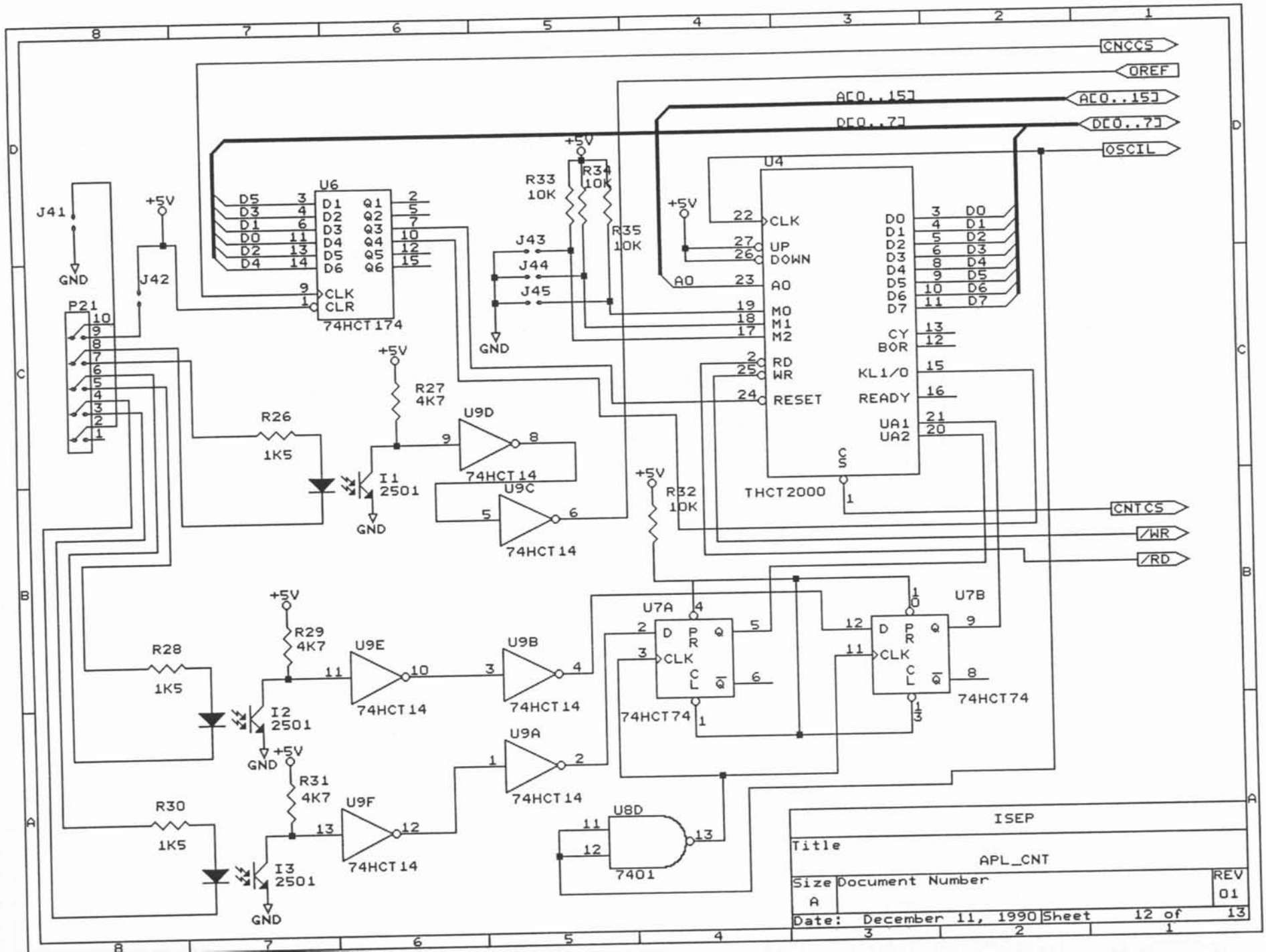

RESET
A[0..15]
AD[0..7]
U?
74HCT245
P3
J?
FLAT26
D[0..7]
IE0
IE1
/WR
/RD
/IOCS
/IO
ALE
GND
+5V
ISEP
FICHA
Size A
Document Number
REV
Date: December 11, 1990
Sheet 7 of 13

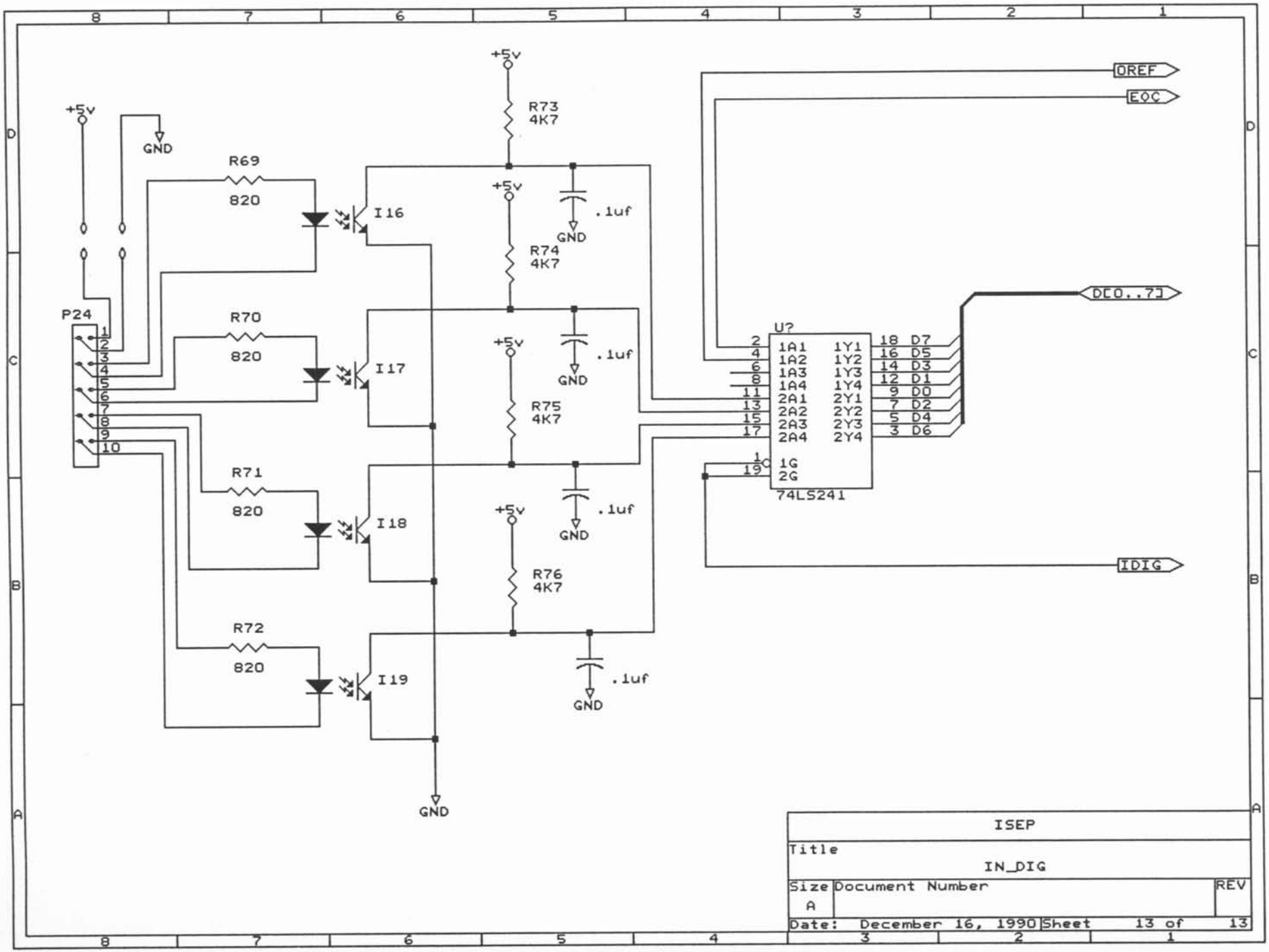

T1/SCLK
AD[0..7]
A[0..15]
/IOCS
/ROMS
/CONFGS
ALE
IE1
IE0
MMCFG
/PSEN
DTA
S I/O
/RD
/WR
/RESET
T0
/EA
C1
10pf
XTL1
12Mhz
C2
10 pf
GND
U6
8044
EA/VP
X1
X2
RST/VPD
INT0
INT1
T0
T1/SCLK
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
RD
WR
PSEN
ALE/P
TXD/DTA
RXD/I/O
U12
74LS373
U9
14L4
U5
74LS244
+5v
R1
10K
R2
10K
R3
10K
R4
10K
R5
330
R6
330
R7
10K
R8
10K
U4C
U4D
U4B
U4E
U4F
7406
D3
1N4148
SW1
C3
10UF
D1
GREEN
D2
RED
ISEP
Title
NCL_CPU
Size
A
Document Number
REV
01
Date: January 7, 1991
Sheet 8 of 13


IEO
QDIG
IDIG
HDIG
CNCCS
/ADCS
/CNTCS
A[0..15]
A[0..4]
/RD
/WR
/IOCS
U2
16L8
I1 I2 I3 I4 I5 I6 I7 I8 I9 I10
A1 A3 A0 A4 A2
GND
+5V
D11
IN4184
R21
10K
R22
10K
R26
10K
R24
10K
R25
10K
R23
R1K
C11
120PF
C22
10UF
U8A
U8B
U8C
74LS01
J51
J56
U5
Q18 Q19 Q20 Q21 Q22 Q23 Q24 OT1 OT2
IN1 IN2 RST
4521
OSCIL
RESET
ISEP
Title
APL_DOG
Size
A
Document Number
REV
Date: December 11, 1990
Sheet 9 of 13


ANL_1
ANL_2
AGND
R44
R45
R46
R47
R48
R49
R50
R51
R52
R53
R54
R55
R60
100
C13
C14
C15
U24A
U24B
TL082
TL082
D14
IN4148
D15
IN4148
D12
IN4148
D13
IN4148
+5V
+5V
GND
U23
7002
CH0
CH1
CH2
CH3
GD
VREF
C1
C2
AGND
DO
D1
D2
D3
D4
D5
D6
D7
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
A0
A1
EOC
/RD
/WR
OSCIL
AD[0..7]
A[0..15]
EOC
/RD
/WR
ADCS
+12V
VO
VI
GND
ISEP
Title
APL_AD
Size
A
Document Number
REV
Date: December 11, 1990
Sheet 10 of 13


P23
P24
U31
74HCT374
U30
74HCT174
U?
ULN2003
R64
R65
R66
R67
R68
R61
R62
R62
R36
1K5
R37
1K5
R38
1K5
R39
1K5
R42
4K7
R40
4K7
R41
4K7
R43
4K7
I12 2501
I11 2501
I10 2501
I9 2501
I8 2501
I13 2501
I14 2501
I5 2501
I4 2501
I6 2501
I7 2501
+5V
GND
D[0..7]
ADC[0..7]
HDIG
ODIG
D7
D5
D3
D1
D0
D2
D4
D6
OC
CLK
CLR
ISEP
Title
APL_DIG
Size
A
Document Number
REV
Date: December 11, 1990
Sheet 11 of 13


CNCCS
OREF
A[0..15]
D[0..7]
OSCIL
U6
74HCT174
U4
THCT2000
J41
J42
J43
J44
J45
P21
R26
1K5
R27
4K7
R28
1K5
R29
4K7
R30
1K5
R31
4K7
R32
10K
R33
10K
R34
10K
R35
10K
I1
2501
I2
2501
I3
2501
U9A
U9B
U9C
U9D
U9E
U9F
74HCT14
U7A
U7B
74HCT74
U8D
7401
GND
+5V
CNTCS
/WR
/RD
ISEP
APL_CNT
Size A
Document Number
REV 01
Date: December 11, 1990
Sheet 12 of 13


+5v
GND
R69
820
I16
R73
4K7
R74
4K7
.1uf
P24
R70
820
I17
R75
4K7
R71
820
I18
R76
4K7
R72
820
I19
OREF
EOC
D[0..7]
IDIG
U?
1A1
1A2
1A3
1A4
2A1
2A2
2A3
2A4
1G
2G
1Y1
1Y2
1Y3
1Y4
2Y1
2Y2
2Y3
2Y4
D7
D5
D3
D1
D0
D2
D4
D6
74LS241
ISEP
Title
IN_DIG
Size
A
Document Number
REV
Date: December 16, 1990
Sheet 13 of 13