UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS

MONITOR PARA CARACTERIZAÇÃO DE RUÍDOS EM REDES DE CORRENTE ALTERNADA.

Álvaro Macedo da Silva

[redacted]

Dissertação apresentada ao Instituto de Física e Química de São Carlos, para a obtenção do título de Mestre em Ciências (Física Aplicada).

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

Orientador:

Prof. Dr. Valentin Obac Roda

DEPARTAMENTO DE FÍSICA E CIÊNCIA DOS MATERIAIS

São Carlos - 1992

UNIVERSIDADE DE SÃO PAULO
Instituto de Física e Química de São Carlos

Fone (0162) 72-6222
Fax (0162) 72-2218

Av. Dr. Carlos Botelho, 1465
Caixa Postal 369
CEP 13560 - São Carlos - SP
Brasil

MEMBROS DA COMISSAO JULGADORA DA DISSERTAÇAO DE MESTRADO DE ALVARO MACEDO DA SILVA APRESENTADA AO INSTITUTO DE FISICA E QUIMICA DE SAO CARLOS, DA UNIVERSIDADE DE SAO PAULO, EM 13/07/1992

COMISSAO JULGADORA:

Prof.Dr.Valentim Obac Roda

Prof.Dr.Ruy Alberto Correa Altafim

Prof.Dr.Alvaro Garcia Neto

Dedico este trabalho

Aos meus pais Benício e Marina, que me ensinaram o amor e o respeito ao trabalho.

À minha esposa Iveti, pelo seu apoio moral e técnico.

Aos meus filhos Daniel e Bruno, por seu incentivo quando os problemas pareciam insolúveis.

AGRADECIMENTOS

À EMBRAPA/NPDIA, nas pessoas do Prof. Dr. Sérgio de Mascarenhas Oliveira, do Dr. Sílvio Crestana e do Dr. Ladislau Martin Neto, por seu incentivo e apoio de infra-estrutura.

À USP/IFQSC/DFCM, pela formação recebida.

Ao Prof. Dr. Valentin Obac Roda, meu orientador, por acreditar no meu trabalho e por me transmitir parte do seu conhecimento e da sua experiência.

À Engª Iveti A.P.Macedo da Silva, minha esposa, por seu apoio técnico, refazendo montagens de circuitos, e pela íntegra do trabalho de edição.

Ao Engº MSc. André Torre Neto, pelo apoio técnico em eletrônica e programação em linguagem C.

Ao Engº Carlos Alberto Corrêa, por sua inestimável orientação na racionalização do uso do tempo e na definição de metas.

Ao Físico MSc. Clóvis I. Biscegli, pelo apoio como amigo, pela colaboração na pesquisa bibliográfica e pela discussão sobre a estrutura da dissertação.

À Engª MSc. Lívia Cristina O.Villela Oliva, pela orientação na catalogação dos artigos da referência bibliográfica.

Aos professores Dr. Jarbas Caiado de Castro Neto, Drª Lourdes Onuchic e Dr. Délio P. Guerrini, por acreditarem no meu trabalho, avalizando o meu ingresso no programa de pós-graduação.

Ao Físico BS Rubens Bernardes Filho e à Bibliotecária BS Maria Angélica Dupas, pelo subsídio na normalização da apresentação do trabalho.

Às Bibliotecárias MSc. Vera L.C. Otaviano e BS. Janis Badovinot, pelo apoio na pesquisa bibliográfica e pela revisão do texto.

Aos técnicos do LIE, por seu apoio na montagem das placas.

Aos técnicos do NPDIA José Ferrazini, André Luís Brassolati, Luiz Aparecido de Godoy, Marcos Possato, e Wilson Risardi, por sua colaboração na confecção de peças e componentes não comerciais.

A todos aqueles que, por uma falha de minha memória, não tenham sido relacionados mas que igualmente colaboraram para que este trabalho tivesse êxito.

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

LISTA DE FIGURAS

Figura 2.1	Arquitetura básica do monitor.
Figura 3.1	Conversão A/D para medição de amplitudes em 60 Hz, níveis de baterias e picos de transientes.
Figura 3.2	Circuito para medição da tensão nominal da rede.
Figura 3.3	Diagrama de blocos do circuito de análise de freqüências.
Figura 3.4	Configuração do filtro passa-baixa.
Figura 3.5	Configuração do filtro passa-faixa.
Figura 3.6	Configuração inicial do detetor de zero, utilizando um comparador de voltagem.
Figura 3.7	Configuração final do detetor de zero, utilizando dois "Schmitt-triggers".
Figura 3.8	Registradores de pulsos dos analisadores de freqüência.
Figura 3.9	Formas de onda devidas a uma descarga atmosférica.
Figura 3.10	Diagrama de blocos do circuito de análise de pulsos.
Figura 3.11	Circuito detetor de pico em pulsos transientes.
Figura 3.12	Circuito de solicitação de interrupção para leitura de pulsos transientes.
Figura 3.13	Microprocessador, gerador de "clock" e "reset" do sistema.

Figura 3.14 Sistema de demultiplexação para ROM e RAM auxiliar.

Figura 3.15 Sistema de demultiplexação para memória de aquisição.

Figura 3.16 Sistema de controle de interrupções.

Figura 3.17 Memória de aquisição: interface com PC-XT e alimentação "stand-by".

Figura 3.18 Controlador de teclado.

Figura 3.19 Arranjo do teclado.

Figura 5.1 Gerador de palavras.

Figura 5.2 Circuito detetor de transiente e circuito contador de pulsos

Figura 5.3 Formas de onda observadas nos circuitos detetor de transiente e contador de pulsos.

LISTA DE TABELAS

Tabela 2.1	Estrutura dos dados armazenados na memória de aquisição.
Tabela 2.2	Estrutura dos dados do diretório.
Tabela 3.1	Comparação entre os parâmetros dos dois métodos de leitura de freqüência.
Tabela 3.2	Surtos oscilatórios mais severos.
Tabela 3.3	Surtos oscilatórios mais freqüentes.
Tabela 3.4	Surtos unidirecionais (maior pico).
Tabela 3.5	Funções das teclas.
Tabela 4.1	Vetores de interrupção do programa aquisição.
Tabela 5.1	Resultados dos testes dos detetores de zero da placa 1.
Tabela 5.2	Resultados dos testes de simulação de desvio da freqüência nominal da rede (60 Hz).
Tabela 5.3	Resultados dos testes de simulação de interferência de rádio-freqüência.
Tabela 5.4	Resultados dos testes de subtensão e sobretensão.
Tabela 5.5	Resultados dos testes de simulação de trasientes-valores lidos.
Tabela 5.6	Resultados da simulação de transientes.
Tabela 5.7	Resultados corrigidos da simulação de transiente.

SUMÁRIO

AGRADECIMENTOS..... i

LISTA DE FIGURAS..... ii

LISTA DE TABELAS..... iv

SUMÁRIO..... v

RESUMO.....viii

ABSTRACT.....viii

1. INTRODUÇÃO..... 1

2. ARQUITETURA E ESTRUTURA DE DADOS..... 4

2.1 Arquitetura..... 4

2.2 Estrutura de dados..... 8

3. DESCRIÇÃO DOS CIRCUITOS..... 12

3.1 Divisor de tensão..... 12

3.2 Análise de amplitudes..... 13

3.2.1 Tensão de pico transiente..... 13

3.2.2 Tensão nominal da rede..... 14

3.2.3 Tensões das baterias..... 15

3.3 Análise de freqüências..... 16

3.3.1 Filtros..... 16

3.3.2 Detetor de zero..... 19

3.3.3 Discriminador lógico..... 20

3.3.4 Registradores de pulsos..... 24

3.4 Análise de pulsos..... 26

3.4.1 Filtro passa-alta..... 30

3.4.2 Registradores de picos dos transientes..... 30

3.4.3 Detetor de pulsos..... 30

3.4.4 Medidores de duração e freqüência..... 32

3.5 Controle microprocessado..... 32

3.5.1 Microprocessador..... 33

3.5.2 Interface de demultiplexação..... 33

3.5.3 Banco de memória 35

3.5.4 Banco de memória ram auxiliar..... 36

3.5.5 Sistema de controle de interrupção..... 36

3.6 Memória de aquisição..... 37

3.7 Comunicação serial..... 39

3.8 Interface homem-máquina..... 39

3.8.1 Teclado..... 39

3.8.2 Visor..... 41

3.9 Relógio..... 42

4. DESCRIÇÃO FUNCIONAL DOS PROGRAMAS..... 43

4.1 Aquisição..... 43

4.1.1 Programa principal (MONITOR .C)..... 44

4.1.2 Módulo IOMON.C..... 47

4.1.3 Módulo CONVERT.C..... 48

4.1.4 Módulo RWMEM.C..... 49

4.1.5 Módulo AD.C..... 50

4.1.6 Módulo IRF.C..... 52

4.1.7 Módulo TECDISP.C..... 54

4.1.8 Módulo RELÓGIO.C..... 57

4.1.9 Módulo MAT.C..... 60

4.1.10 Módulo DEFINE.H..... 61

4.2 Transferência..... 61

4.2.1 Programa principal..... 62

4.2.2 Função leram ()..... 62

4.3 Relatório..... 63

4.3.1 Módulo RELAT.C..... 63

4.3.2 Módulo ROT_REL.C..... 63

5. RESULTADOS..... 67

5.1 Testes de desenvolvimento..... 67

5.1.1 Placa 1..... 67

5.1.2 Placa 2..... 69

5.1.3 Placa 3..... 70

5.1.4 Placa 4..... 71

5.2 Testes de simulação..... 73

5.2.1 Desvios da freqüência de 60 Hz..... 73

5.2.2 Interferências de rádio-freqüência..... 74

5.2.3 Variações na amplitude da rede..... 75

5.2.4 Transientes..... 76

6. CONCLUSÃO..... 81

7. REFERÊNCIAS BIBLIOGRÁFICAS..... 82

APÊNDICE A - FONTES DOS PROGRAMAS..... A-1

APÊNDICE B - CIRCUITOS ELETRÔNICOS..... B-1

APÊNDICE C - EQUIPAMENTOS UTILIZADOS NA DEPURAÇÃO..... C-1

APÊNDICE D - TABELAS DE ENDEREÇOS DE E/S..... D-1

APÊNDICE E - MAPA DE MEMÓRIA..... E-1

RESUMO

Distúrbios em redes elétricas podem causar danos materiais e/ou operacionais em equipamentos eletrônicos sensíveis, em particular, sistemas de microcomputadores. Com o objetivo de registrar estes distúrbios, um monitor para redes elétricas foi desenvolvido. O equipamento desenvolvido é controlado por um microprocessador 80C86-2. Os seguintes distúrbios podem ser monitorados: desvios da freqüência nominal da rede, interferências de rádio freqüência, subtensão, sobretensão e surtos de tensão. Para minimizar o gargalo de Von Neumann, os circuitos foram projetados de forma que a maior parte das leituras são solicitadas via interrupção. O equipamento foi testado a nível de laboratório e os resultados são apresentados.

ABSTRACT

Power line disturbances can adversely affect electronic sensitive equipment, particularly microcomputer based systems. In order to register these disturbances a power line monitor has been developed. The developed instrument is microprocessor based using a 80C86-2 as the main CPU. The following electrical disturbances can be monitored: nominal frequency deviations, radio-frequency interference, under and overvoltages, transient over voltages. Attempting to minimaze Von Neumann's bottle neck, the hardware was designed in such a way that most of the readings were requested through interrupts. The equipment was tested through a series of disturbance simulations, and the results are presented.

1 INTRODUÇÃO

Com a crescente utilização do computador e de equipamentos eletrônicos sofisticados em praticamente toda área de atividade humana, um aspecto passou a preocupar usuários e engenheiros em níveis cada vez mais rigorosos: a qualidade do suprimento de energia elétrica. Desde a constatação desta nova realidade, têm surgido, então, inúmeros equipamentos que, supostamente, protegem contra quaisquer distúrbios que venham a surgir na rede elétrica. Isso não é bem verdade, porque, de fato, a solução para as possíveis anomalias de modo algum é simples, considerando-se as várias formas de tais ocorrências e que a proteção contra todos eles torna-se economicamente proibitiva.

Torna-se necessário, então, o conhecimento dos distúrbios possíveis de estar presentes em uma rede. E para tal, deve-se, antes, ser definida uma linha de parâmetros segundo a qual, um fornecimento será considerado ou não como adequado. Desta forma, pode-se definir como distúrbio qualquer perturbação que tenha como consequência uma energia elétrica diferente da senoidal pura, constante em amplitude, freqüência e fase.

Pode-se adotar a classificação da qualidade do fornecimento de energia utilizada por T. S. Key (Key, 1979), a saber:

a) variação do nível de tensão;
b) desbalanceamento de fases;
c) distorção harmônica;
d) variação de freqüência.

Os ítens acima são considerados como fatores pertinentes à rede em regime. Devem ser levados em conta, ainda, os distúrbios transientes e aleatórios causados por curto-circuitos, chaveamento de cargas reativas, desligamentos de lâmpadas e outros.

Durante muitos anos, vários levantamentos foram feitos considerando-se os transientes em redes de alta tensão. Existe uma vasta literatura neste sentido. Entretanto a nível de baixa tensão, ou seja, no ponto de consumo, a informação encontra-se ainda na forma de artigos, sendo os primeiros trabalhos de vulto devidos a

Martzloff (1970) e a Allen (1974). Estas publicações fornecem uma excelente informação estatística sobre amplitude, duração e taxa de incidências de distúrbios em circuitos de baixa voltagem.

Com base em Martzloff (1970), Allen (1974) e em um levantamento da marinha, Key (1979) classificou as anomalias em três tipos:

- Transiente ou Sobretensão Oscilatória

São distúrbios com duração de 5μs a 200μs, com freqüências entre 200 Hz e 5 kHz. Seu valor eficaz limite é de 200% a 400% da tensão nominal ou maior, podendo seus picos serem positivos ou negativos. As causas mais comuns são descargas atmosféricas, chaveamento de cargas reativas e operação de outras cargas.

- Sobretensão ou Subtensão Momentâneas

Sua duração é de 64 ms a 1 s, dependendo do tipo de equipamento do sistema de distribuição. Situam-se, respectivamente, abaixo de 85% e acima de 110% da tensão eficaz nominal e devem-se principalmente a falhas do sistema, grandes variações em cargas elevadas e mal funcionamento do equipamento de energia.

- Interrupção de Energia

A duração das interrupções é de 2s a 60s, quando o restabelecimento é automático, ou de 15 min a 4 h, quando manual. Seu limiar superior de tensão é de 85% do valor nominal e deve-se a faltas do sistema, mudanças inaceitáveis de carga e mal funcionamento do equipamento de energia. Pode ocorrer também uma redução proposital no nível de tensão por parte da concessionária.

Tão importante quanto conhecer os distúrbios que em média podem ocorrer em uma instalação é a necessidade de se determinar a sensibilidade dos equipamentos a serem protegidos. Este aspecto foi abordado por Key (1979) e Trindade (1987).

Dependendo da confiabilidade exigida dos equipamentos ligados à rede elétrica, da sua susceptibilidade e da severidade dos

distúrbios presentes, vários métodos de condicionamento da energia podem ser adotados: supressores de surtos, transformadores de isolação, reguladores de tensão, conjuntos moto-geradores, fontes ininterruptas de energia e outros (Martzloff e Gruzs, 1988).

A adição desses equipamentos de proteção é onerosa, implicando em uma preponderância do fator econômico, quando da escolha da medida a ser aplicada. Entretanto, se os dados fornecidos como parâmetros para esta determinação forem incorretos, porque o levantamento foi inadequado, todo o processo é destituído de valor, ou o que é pior, pode conduzir a decisões erradas.

O objetivo deste trabalho foi o desenvolvimento de um monitor capaz de caracterizar os distúrbios mais freqüentes em uma rede C.A. de baixa tensão, sendo assim possível, para uma instalação existente, determinar-se qual a proteção mais adequada e, para as instalações novas, pode-se, adicionalmente, ser feita também a escolha do melhor local em uma planta, para a implantação de um sistema computacional.

2 ARQUITETURA E ESTRUTURA DE DADOS

2.1 ARQUITETURA

Todo sistema microprocessado possui uma distribuição e uma interconexão de elementos que são peculiares à atividade que por ele será servida. A este arranjo, levando-se em conta a forma pela qual o mesmo é "visto" por um programa em linguagem de máquina, dá-se o nome de arquitetura.

Na escolha de uma arquitetura, vários fatores devem ser considerados, havendo um compromisso entre eles para se determinar a estrutura final a ser adotada. Os parâmetros mais preponderantes são: tempo de processamento, tempo de aquisição, tempo de escrita e de leitura, tamanho dos dados, transferência dos dados, complexidade dos circuitos, complexidade e portabilidade do programa, e generalização/dedicação do sistema. Alguns fatores podem ser exclusivos. Por exemplo, em um sistema dedicado, em geral, não há necessidade de se preocupar com portabilidade do programa. Este é o caso do presente projeto.

Outro aspecto da atual arquitetura é que, devido à curta duração dos transientes, procurou-se deixar a maior parte das aquisições e decisões sobre as leituras ou não das mesmas a cargo do circuito. Desta forma, dois conhecidos causadores de atraso puderam ser minimizados:

- O gargalo de Von Neumann, onde o microprocessador executa a maior parte das ações, transformando o barramento de dados em uma via altamente congestionada;

- Programa complexo, onde, à semelhança do caso anterior, muitas decisões são tomadas durante o processamento, provocando, esperas igualmente indesejáveis.

Além dos fatores acima, a escolha da arquitetura, obviamente, deve ser orientada pelas necessidades do processo a ser controlado ou medido.

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

No caso do monitor de ruído, os requisitos necessários são :

- Medição das interferências de rádio frequência (IRF) mais comuns, ou seja, na faixa de 1 kHz a 1 MHz. Segundo Dang (1985) as IRF's podem ocorrer nas frequências de 10 kHz a 30 MHz, porém apenas os valores de 10 kHz a 1 MHz são prejudiciais aos sistemas, uma vez que os ruídos acima dessa faixa são de baixa intensidade.

- Medição dos transientes de curta duração causados por chaveamento de lâmpadas, aparelhos de ar condicionado, descargas atmosféricas e outros.

- Acompanhamento da tensão da rede em amplitude e frequência.

- Memória de programa.

- Memória para aquisição dos dados.

- Memória para as variáveis temporárias e pilha.

- Relógio para possibilitar registro do momento da ocorrência.

- Interface homem-máquina para interação entre o usuário e o equipamento.

Adicionalmente, foi instalada uma interface serial na placa de processamento para facilitar o desenvolvimento.

Chegou-se então, a uma arquitetura final, contendo os seguintes blocos, conforme apresentado na figura 2.1.

- Divisor de Tensão

- Análise de Amplitudes

- Análise de Frequências

- Análise de Pulsos

- Processador
- Memória de Programa
- Memória de Aquisição
- Memória Auxiliar
- Teclado e Display
- Relógio
- Comunicação Serial

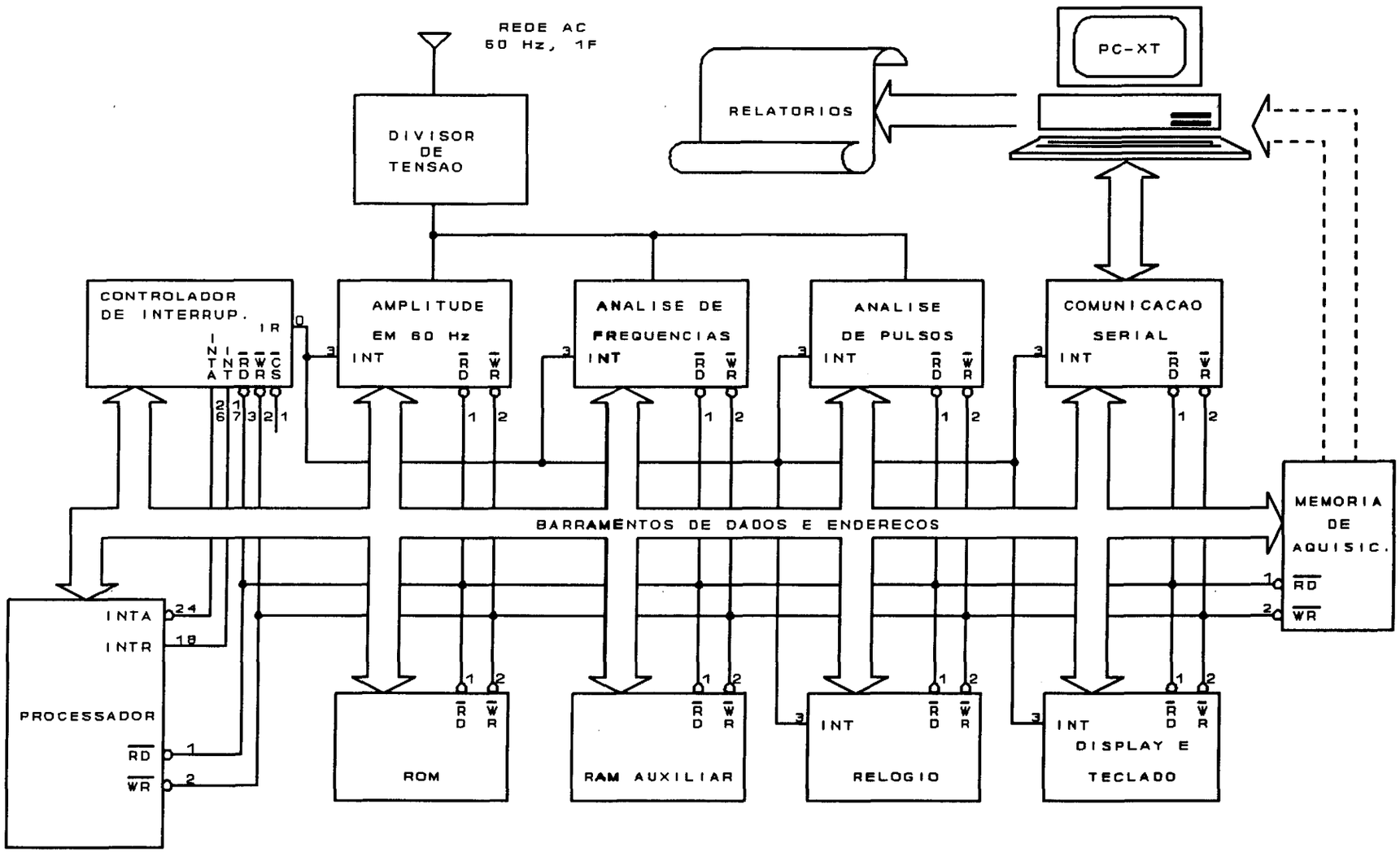


FIGURA 2.1 - ARQUITETURA BÁSICA DO MONITOR

2.2 ESTRUTURA DE DADOS

A escolha da estrutura dos dados de um sistema é um fator tão importante na determinação da velocidade do fluxo de informação quanto o são o circuito e o programa. Esta estrutura, se bem dimensionada, irá permitir uma otimização de ocupação da memória, bem como tornar possível o armazenamento de uma informação com um menor número de bytes, o que implica na redução de ciclos de máquina para escrita ou leitura.

No atual projeto, o maior dado possível para a aquisição foi projetado para 12 bits, ficando os bits de dados D13 a D15 reservados para armazenar o tipo de informação contida na palavra de 16 bits (ver tabela 2.1). Na Tabela 2.1 estão apresentadas as estruturas dos dados que representam as anomalias.

Os tamanhos dos dados representando data e horário foram determinados conforme segue:

segundo : máximo 59D = 3BH (6 bits)
minuto : máximo 59D = 3BH (6 bits)
hora : máximo 23D = 17H (5 bits)
dia : máximo 31D = 1FH (5 bits)
mês : máximo 12D = 0CH (4 bits)
ano : máximo 99D = 63H (7 bits)

TABELA 2.1 - ESTRUTURA DOS DADOS ARMAZENADOS NA MEMÓRIA DE AQUISIÇÃO

TIPO DE DADO	BIT			
	15 12	11 8	7 4	3 0
Distorção de 60 Hz	0 0 0 0	I I I I	I I I I	I I I I
IRF 0,6 a 10 kHz	0 0 0 1	I I I I	I I I I	I I I I
IRF 10 a 100 kHz	0 0 1 0	I I I I	I I I I	I I I I
IRF 100 a 1000 kHz	0 0 1 1	I I I I	I I I I	I I I I
Duração de transiente	0 1 0 0	I I I I	I I I I	I I I I
Frequência de transiente	0 1 0 1	I I I I	I I I I	I I I I
Pico de transiente	0 1 1 0	X X X X	I I I I	I I I I
Sobretensão e subtensão	1 0 0 0	X X X X	I I I I	I I I I
Nível baixo bateria pos.	1 0 0 1	X X X X	I I I I	I I I I
Nível baixo bateria neg.	1 0 1 0	X X X X	I I I I	I I I I
Segundos/minutos	1 0 1 1	S S S S	S S M M	M M M M
Hora	1 1 0 0	X X X X	X X X I	I I I I
Mes/dia	1 1 0 1	M M M M	X X X D	D D D D
Ano	1 1 1 0	X X X X	X I I I	I I I I

Onde:

I = bit de informação;

S = bit de segundos;

M = bit de minutos para informação de segundos/minutos ou de mes para mes/dia;

X = não importa.

Visando ainda um melhor aproveitamento de tempo e memória, foram adotadas as seguintes medidas:

- Escritas do ano apenas uma vez, sendo atualizado na mesma posição às 0:00h de 01.01 se não houver distúrbios até este momento, ou escrito em nova posição caso ocorram distúrbios.

- Idem para mês/dia, com atualização às 0:00h de todos os dias.

- Idem para hora a cada mudança de hora.

A disposição dos dados na memória, para cada tipo de ocorrência será a seguinte:

- Distorção de 60 Hz, IRF, sobretensão, subtensão, níveis de bateria

1ª palavra : dado
2ª palavra : segundo/minuto

- Transiente

1ª palavra : duração
2ª palavra : frequência
3ª palavra : valor de pico
4ª palavra : segundo/minuto

No início da memória de aquisição, um conjunto de dados, que para facilidade de indentificação será chamado de diretório, contém as seguintes informações sobre o arquivo de coleta dos distúrbios:

1ª palavra : dia/mes início aquis. - D0000H
2ª palavra : ano/hora início aquis. - D0002H
3ª palavra : seg/min início aquis. - D0004H
4ª palavra : dia/mes final aquis. - D0006H
5ª palavra : ano/hora final aquis. - D0008H
6ª palavra : seg/min final aquis. - D000AH
7ª palavra : bat. pos. final aquis. - D000CH
7ª palavra : bat. neg. final aquis. - D000DH
8ª palavra : end. do fim de arquivo - D000EH

Como pode ser visto na Tabela 2.2, a estrutura dos dados do diretório é diferente da estrutura de aquisição. Isto por não ser necessária, para esse caso, a utilização dos quatro bits mais significativos para controle. Desta forma, foi possível criar-se um diretório mais compacto.

TABELA 2.2 - ESTRUTURA DOS DADOS DO DIRETÓRIO

TIPO DE DADO	BIT			
	15 12	11 8	7 4	3 2
Dia/Mes	0 0 0 0	M M M M	0 0 0 D	D D D D
Ano/Hora	0 0 0 H	H H H H	0 A A A	A A A A
Seg/Min	0 0 S S	S S S S	0 0 M M	M M M M
Bateria	N N N N	N N N N	P P P P	P P P P
Endereço Final	E E E E	E E E E	E E E E	E E E E

Onde:

M = bit de mes em Dia/Mes ou minuto em Seg/Min;

D = bit de dia em Dia/Mes;

H = bit de hora em Ano/Hora;

A = bit de ano em Ano/Hora;

S = bit de segundo em Seg/Min;

N = bit de nível para bateria negativa em Bateria;

P = idem para bateria positiva;

E = bit de endereço em Endereço Final.

3 DESCRIÇÃO DOS CIRCUITOS

A primeira consideração feita ao se iniciar o projeto dos circuitos foi a imunidade dos mesmos aos ruídos da rede. O equipamento que iria registrar as anomalias da alimentação elétrica não poderia, obviamente, estar vulnerável às mesmas. Por outro lado, a proteção do equipamento também não seria simples, uma vez que antes de se caracterizar os distúrbios de cada local a ser monitorado não se poderia fazer nenhum tipo de prescrição quanto a tal proteção. A solução, então, seria a alimentação através de baterias.

A partir dessa determinação, a escolha de componentes eletrônicos de baixo consumo foi uma conseqüência natural. Definiu-se, assim, que seriam utilizados circuitos integrados fabricados com tecnologia CMOS. Para se ter uma idéia de grandeza, enquanto um controlador de interrupções 8259 NMOS consome 85 mA seu equivalente em CMOS, o 82C59, consome apenas 5 mA (Intel, 1987b). Uma noção mais aprofundada das diferenças entre as duas tecnologias, é apresentada no artigo de Pawloski, Moroyan e Altnether (1983). Adicionalmente, os componentes CMOS são mais tolerantes aos ruídos.

Os circuitos, que são descritos a seguir, foram distribuídos por seis placas:

Placa 1 - Análise de Freqüências;
Placa 2 - UCP, Comunicação Serial, Visor, Teclado e Relógio;
Placa 3 - Memória de Aquisição;
Placa 4 - Análise de Amplitudes e Análise de Pulsos;
Placa 5 - Fonte de Alimentação;
Placa 6 - Divisor de Tensão.

3.1 DIVISOR DE TENSÃO

O divisor de tensão não foi montado porque os testes foram efetuados com um gerador de funções, utilizando-se amplitudes de 5V compatíveis com os valores esperados pelo sistema após o divisor.

Convém lembrar, porém, que, para as freqüências em questão, ele não pode ser simplesmente resistivo, havendo necessidade de se utilizar um divisor auto-compensado.

3.2 ANÁLISE DE AMPLITUDES

O bloco de análise de amplitudes (Figura 3.1) é responsável pelo acompanhamento dos seguintes valores de tensão:

- Tensão de Pico de Transiente
- Tensão Nominal da Rede
- Tensão da Bateria Positiva
- Tensão da Bateria Negativa

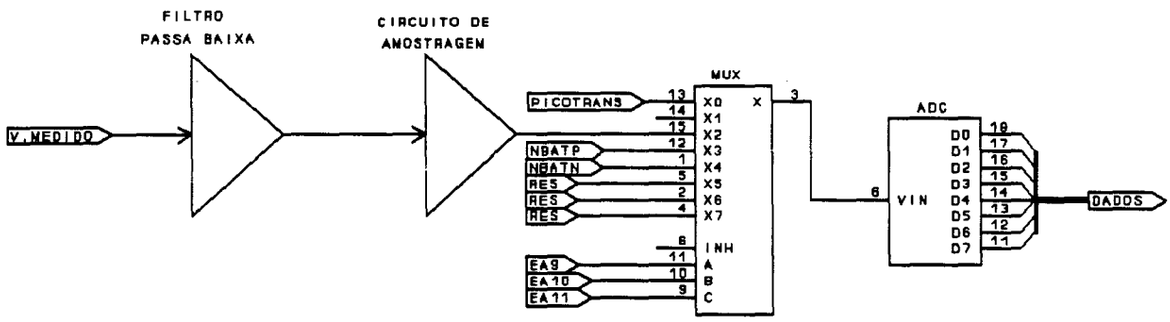


FIGURA 3.1 - CONVERSAO A/D PARA MEDIÇÃO DE AMPLITUDE EM 60 Hz, NÍVEIS DE BATERIAS E PICOS DE TRANSIENTES

3.2.1 TENSÃO DE PICO DE TRANSIENTE

A tensão de pico de transiente é registrada através de um detetor de pico e selecionada, como todos os quatro casos desse bloco, através de um multiplexador 4051 (RCA, 1983).

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

3.2.2 TENSÃO NOMINAL DA REDE

A tensão nominal da rede é acompanhada através de uma interrupção produzida cada vez que a onda da rede completa um ciclo. Neste ponto, o sistema lê o valor da amplitude armazenado por um circuito detetor de zero. Este valor é comparado com a faixa de 85% a 110% de 127 Vrms já citada no capítulo 2, para se verificarem as ocorrências de subtensão e sobretensão. O armazenamento ocorre apenas quando a tensão está fora da faixa.

Como para a proteção contra sobretensão o importante é saber se em uma determinada localidade, ou mesmo em um certo ponto do fornecimento, ocorrem sobretensões, não se considerou uma gama extensa acima de 110% de 127V, mas sim, foi estipulado um limite de 150%. Assim sendo, o máximo valor a ser medido em amplitude seria de 269V.

Um segundo divisor foi acrescentado e com uma configuração ligeiramente diferente dos divisores convencionais. Sua função é fazer uma conversão de faixa (Figura 3.2). Desta forma, a saída do primeiro divisor, de -15V a + 15V, passa a ser, na saída do segundo divisor, de 0 a 5V. Este sinal passa, então, por um filtro passa-baixa de 60 Hz e é selecionado pelo multiplexador e a seguir convertido para digital. Como o conversor análogo-digital escolhido é de oito dígitos, a resolução à entrada do primeiro divisor é de 2V.

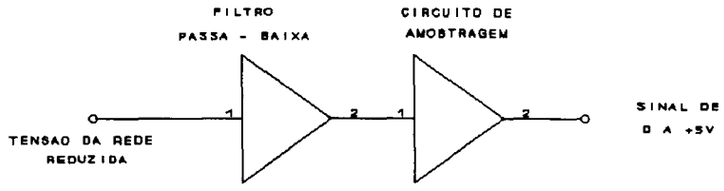


FIGURA 3.2 - CIRCUITO PARA MEDIÇÃO DA TENSÃO NOMINAL DA REDE

3.2.3 TENSÕES DAS BATERIAS

O sistema é alimentado por duas baterias conforme indicado no desenho MNT-ESQ-PL5, folha 01, no apêndice B.

Como os componentes possuem uma tolerância limitada em relação à tensão de alimentação, procurou-se monitorar as baterias de modo a não permitir que o sistema operasse fora dos limites inferiores de segurança.

O valor de tensão foi, então, determinado pelo microprocessador que é um dos menos tolerantes, exigindo que sua alimentação fique dentro da faixa de 4,5 V a 5,5 V. Através de uma fonte variável, diminuiu-se a tensão de entrada da fonte do sistema até se atingir o valor de 4,5 V. Este valor foi obtido para uma voltagem de entrada de 8,5 V.

Quando o valor da tensão de qualquer das baterias chega, então, nesse valor limite, três ações são tomadas pela central de processamento, a partir de um sinal de interrupção:

- seu valor e o horário da ocorrência são armazenados na memória de aquisição.

- a aquisição é interrompida para evitar que sejam armazenados dados não confiáveis devido à baixa tensão de alimentação dos circuitos.

- uma mensagem no visor de cristal líquido alerta sobre o estado de parada do equipamento.

A leitura das tensões das baterias pode, também, ser solicitada a qualquer momento via teclado. Uma terceira forma de leitura das baterias ocorre ao final do período de aquisição. Este término de aquisição é solicitado via teclado, após o que, é atualizado o diretório do arquivo, incluindo os valores das baterias.

3.3 ANÁLISE DE FREQÜÊNCIAS

Este bloco monitora as freqüências de eventos de longa duração desde 60 Hz até 1 MHz. Está dividido em quatro faixas:

- Faixa 1 - freqüências próximas a 60 Hz
- Faixa 2 - 600 Hz a 10 kHz
- Faixa 3 - 10 kHz a 100 kHz
- Faixa 4 - 100 kHz a 1 MHz

Na primeira faixa faz-se o acompanhamento da freqüência da rede, havendo um registro da mesma somente quando houver um desvio de 0,1% do seu valor nominal de 60 Hz.

Nas demais faixas, são verificadas interferências de radio-freqüência (IRF) de 600 Hz a 1 MHz.

Os circuitos para as quatro faixas são compostos, como pode ser visto na Figura 3.3, dos seguintes sub-blocos:

a) Filtro
b) Detetor de Zero
c) Discriminador Lógico
d) Registradores de Pulsos

3.3.1 FILTROS

Para cada um dos casos citados no ítem anterior, foi utilizado um filtro diferente, ou seja, para a primeira faixa um filtro passa-baixa, para as faixas 2 e 3 um filtro passa-faixa e para a última um filtro passa-alta. Em todos os casos foram utilizados filtros ativos com amplificadores operacionais (Stout, 1976).

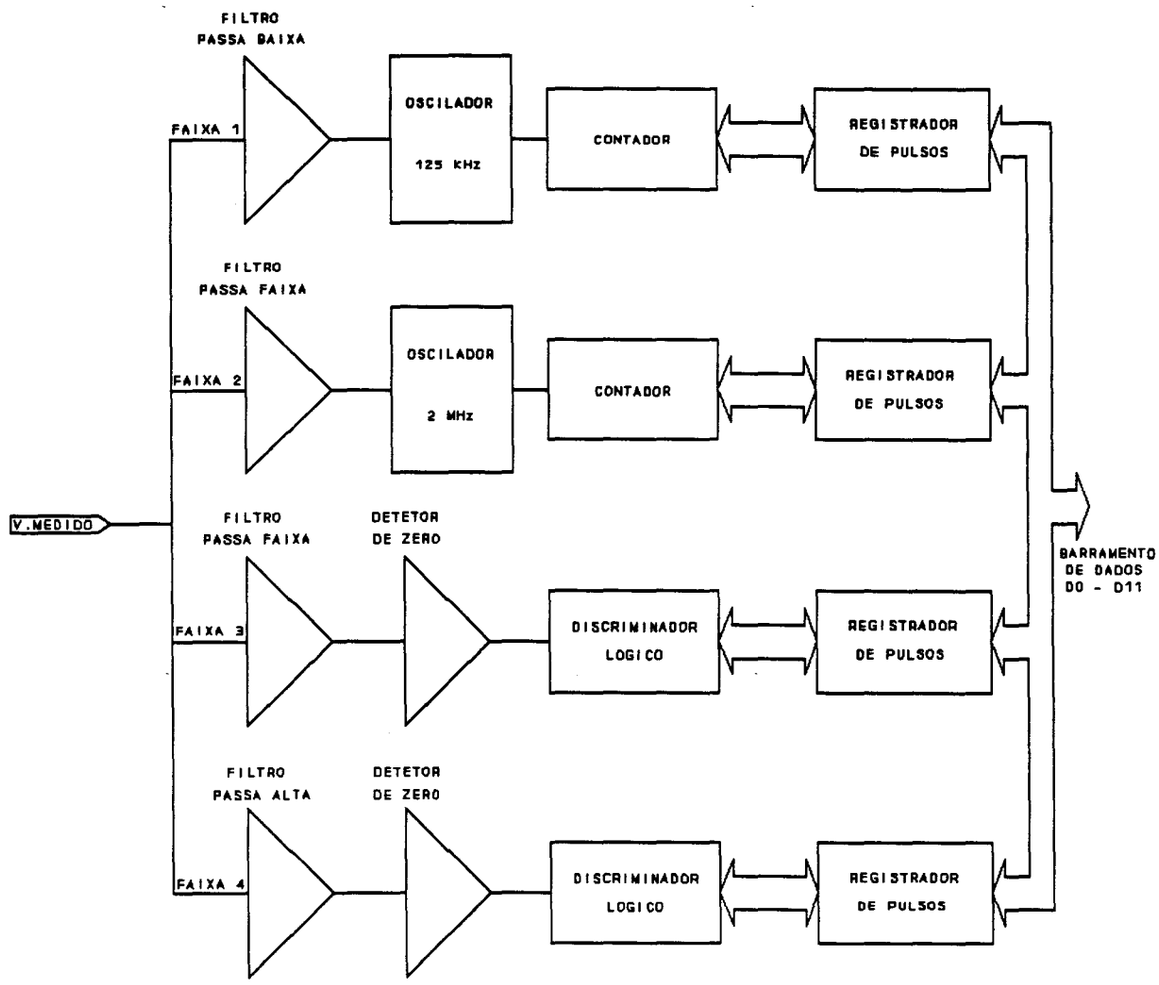


FIGURA 3.3 - DIAGRAMA DE BLOCOS DO CIRCUITO DE ANÁLISE DE FREQUÊNCIAS

3.3.1.1 FILTRO PASSA-BAIXA

Para a faixa de 60 Hz, a configuração do filtro foi a apresentada na Figura 3.4. Uma vez que a amplitude não é um fator preponderante nestes casos onde se pretende apenas determinar a freqüência do sinal, foi adotado o filtro do tipo Chebyshev de 2 polos com pico de 3 dB (Stout,1976).

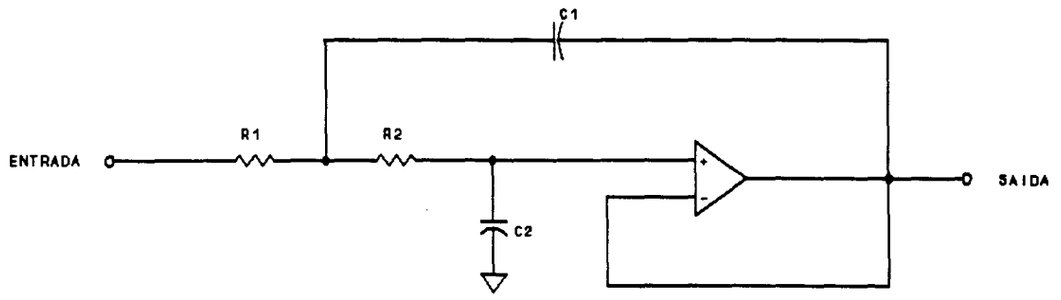


FIGURA 3.4 - CONFIGURAÇÃO DO FILTRO PASSA-BAIXA

Os valores dos componentes podem ser vistos no apêndice B, desenho MNT-ESQ-PL1, folha 1/13.

3.3.1.2 FILTROS PASSA-ALTA E PASSA-FAIXA

Para o filtro passa-alta a configuração utilizada é semelhante à do passa baixa havendo inversão dos capacitores com os resistores (Stout, 1976), (Figura 3.5).

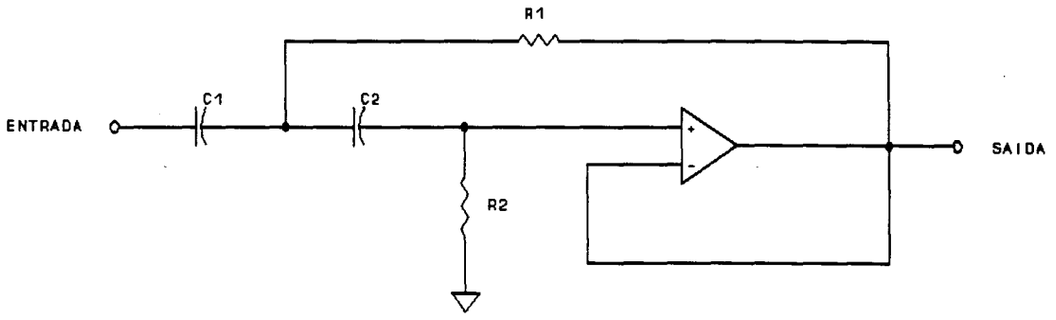


FIGURA 3.5 - CONFIGURAÇÃO DO FILTRO PASSA-ALTA

Para os filtros passa-faixa foi feita uma superposição de um passa-baixa com um passa-alta e utilizados os procedimentos de cálculo já citados (Stout, 1976).

Cada passa-faixa teve como valor de corte para o passa-baixa o valor superior da faixa correspondente e para corte do passa-alta o valor inferior.

3.3.2 DETETOR DE ZERO

O detetor de zero, inicialmente, era composto de um comparador simples de voltagem, sendo a onda de entrada comparada com o terra lógico (Figura 3.6).

Como alternativa final, foi adotado, por apresentar melhores resultados, um par de inversores "Schmitt-trigger" (Figura 3.7).

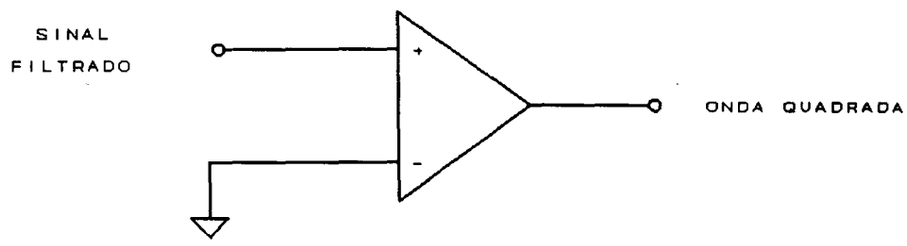


FIGURA 3.6 - CONFIGURAÇÃO INICIAL DO DETETOR DE ZERO, UTILIZANDO UM COMPARADOR DE VOLTAGEM

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

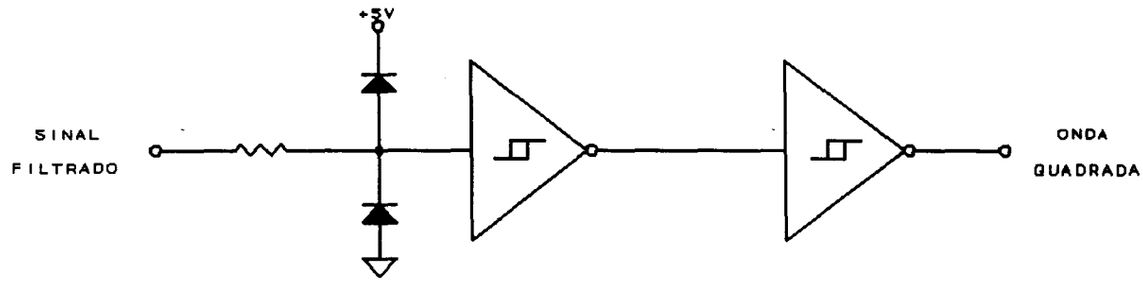


FIGURA 3.7 - CONFIGURAÇÃO FINAL DO DETETOR DE ZERO, UTILIZANDO DOIS SCHIMITT-TRIGGERS

3.3.3 DISCRIMINADOR LÓGICO

O discriminador lógico é composto para as faixas 2, 3 e 4, de um contador de pulsos e um comparador lógico. Para a faixa 1, constitui-se apenas do contador. Possui as seguintes funções:

- Armazenar a freqüência de um sinal de IRF caso este esteja dentro da faixa em questão.

- Solicitar, através de interrupção, a leitura dessa freqüência.

3.3.3.1 CONTADOR DE PULSOS

Entre os métodos disponíveis para medição de freqüência, dois foram analisados e comparados face às necessidades do presente trabalho. Ambos são discutidos em detalhe em Horowitz e Hill (1980).

No primeiro método, conta-se o número de pulsos (N_L) da freqüência a ser lida (f_L) durante um determinado intervalo que será utilizado como base de tempo (T_b). A freqüência será, então:

$$f_L = \frac{N_L}{T_b}$$

No segundo método, lê-se o inverso da freqüência, ou seja, seu período (T_L). Utiliza-se para isto um oscilador de referência com freqüência f_{osc} e, através de um detetor de zeros, totaliza-se o número de pulsos deste oscilador (N_{osc}) durante T_L. Tem-se, então:

$$T_L = N_{osc} x \frac{1}{f_{osc}} \quad e \quad f_L = \frac{1}{T_L}$$

$$Logo \quad f_L = \frac{f_{osc}}{N_{osc}}$$

Dois parâmetros são determinantes na escolha de um dos métodos: o erro e o tempo de totalização.

Em uma medição utilizando contadores, o erro é dado pelo desvio de ± 1 dígito na leitura. Assim sendo, para a determinação da freqüência pelo 1º método (LF) o erro, e_{LF} é dado, por ± 1 pulso da freqüência dentro do tempo base, e para o 2º método (LP) por ± 1 pulso do oscilador de referência dentro de T_L (erro e_{LP}). Portanto, para e_{LF}, tem-se:

$$e_{LF} = \frac{\Delta f_L}{f_L} = \frac{\frac{N_L \pm 1}{T_b} - \frac{N_L}{T_b}}{\frac{N_L}{T_b}}$$

$$Logo, \quad e_{LF} = \pm \frac{1}{N_L}$$

E, para e_{LP}, tem-se:

$$f_L = \frac{1}{T_L} \rightarrow \Delta f_L = -\frac{\Delta T_L}{T_L^2}$$

$$N_{osc} = \frac{f_L}{f_{osc}}$$

$$e_{LP} = \frac{\Delta f_L}{f_L} = -\frac{\Delta T_L}{T_L} = -\frac{\dfrac{N_{osc} \pm 1}{f_{osc}} - \dfrac{N_{osc}}{f_{osc}}}{\dfrac{1}{f_L}}$$

$$Logo, \quad e_{LP} = \pm \frac{f_L}{f_{osc}}$$

O tempo de aquisição para LF é a própria base de tempo T_b e para LP o período T_L.

Para o primeiro método foram escolhidas bases de tempo que produzissem erros dentro do primeiro dígito percentual. E para o segundo método foi escolhido um oscilador que permitisse manter a estrutura de dados do item 2, ou seja 12 dígitos para medição do evento.

A Tabela 3.1 mostra um quadro comparativo entre os dois métodos.

Estabelecendo-se um compromisso entre o erro e o tempo de totalização adotou-se o primeiro método para as faixas 3 e 4 e o segundo para as faixas 1 e 2. Desta forma o erro ficou entre 0,05% e 0,98% e o tempo entre 1,02 ms e 16,67 ms.

Como foi dito em 3.3.3 a leitura de freqüência é feita através de uma interrupção. A forma como isto ocorre, porém, difere da faixa 1 para as demais.

TABELA 3.1 - COMPARAÇÃO ENTRE OS PARÂMETROS DOS DOIS MÉTODOS DE LEITURA DE FREQUÊNCIA

FAIXA(Hz)	60	600 a 10k	10k a 100k	100k a 1M
N_L (pulsos)	102	61 a 1020	102 a 1020	102 a 1020
N_{osc}(pulsos)	2083	3334 a 200	800 a 80	80 a 8
f_{osc}(Hz)	125k	2M	8M	8M
T_b(s)	1,70	102,00m	9,60m	0,96m
T_L(s)	16,67m	1,67 a 100μ	100μ a 10μ	10μ a 1 μ
e_{LF}(%)	0,98	1,64 a 0,10	0,98 a 0,10	0,98 a 0,10
e_{LP}(%)	0,05	0,03 a 0,50	0,13 a 1,25	1,25 a 12,5

No primeiro caso, durante um período da tensão da rede, um oscilador de 125 KHz tem seus pulsos totalizados. Ao final desta contabilização é emitido um sinal de interrupção. O microprocessador lê o registrador de solicitação de leitura de freqüência para verificar qual faixa será servida. Em se tratando da faixa 1 serão lidas a amplitude e a freqüência da rede. O valor que não estiver dentro da faixa de ±0,1% em torno do nominal será registrado.

Para as demais faixas o procedimento inicial de verificação da faixa solicitante é o mesmo. Porém, a leitura somente será efetuada se houver um sinal de IRF entre 600 Hz e 1 MHz. O número de pulsos lidos será, então, armazenado.

3.3.3.2 COMPARADOR LÓGICO

Como foi dito em 3.3.3.1, a única condicionante para a leitura da freqüência da rede é a totalização dos pulsos do oscilador durante um período desta onda. Logo, para esta faixa não há necessidade de comparadores.

Para as demais faixas, entretanto, ao final de uma totalização torna-se necessária a comparação do número de pulsos com os valores da Tabela 3.1 (N_L e N_{osc}). A interrupção ocorrerá somente se o número de pulsos se enquadrar na sua respectiva faixa.

Na elaboração desses comparadores foi levado em conta o fato de que a contagem é progressiva. Assim sendo, ao entrar na faixa, os dígitos que fazem a transição de 0 para 1 elevam a saída Q de um "flip-flop" produzindo, através de uma lógica "E" com outro "flip-flop" uma saída em nível alto. Se o número de pulsos continuar crescendo e ultrapassar o segundo limite da faixa, um processo semelhante abaixará a saída Q do segundo "flip-flop", provocando uma saída 0 no comparador.

Os detalhes dessa lógica podem ser vistos no desenho MNT-ESQ-PL1, Folhas 8/13, 10/13 e 12/13.

3.3.4 REGISTRADORES DE PULSOS

Na hipótese de uma conclusão de totalização de pulsos na faixa 1 ou de uma ocorrência de IRF, é feita a emissão de um pedido de interrupção.

Após receber o sinal de interrupção solicitando leitura, o processador primeiramente lê a segunda metade do registrador mais significativo da faixa 3, identificando em quais faixas ocorreram IRF's (Figura 3.8).

A seguir o processador seleciona os registradores de pulsos da faixa em que ocorreu o distúrbio, lendo N_L ou N_{OSC}.

Por último é enviado um sinal que reinicializa os contadores de pulsos e o comparador lógico, se houver.

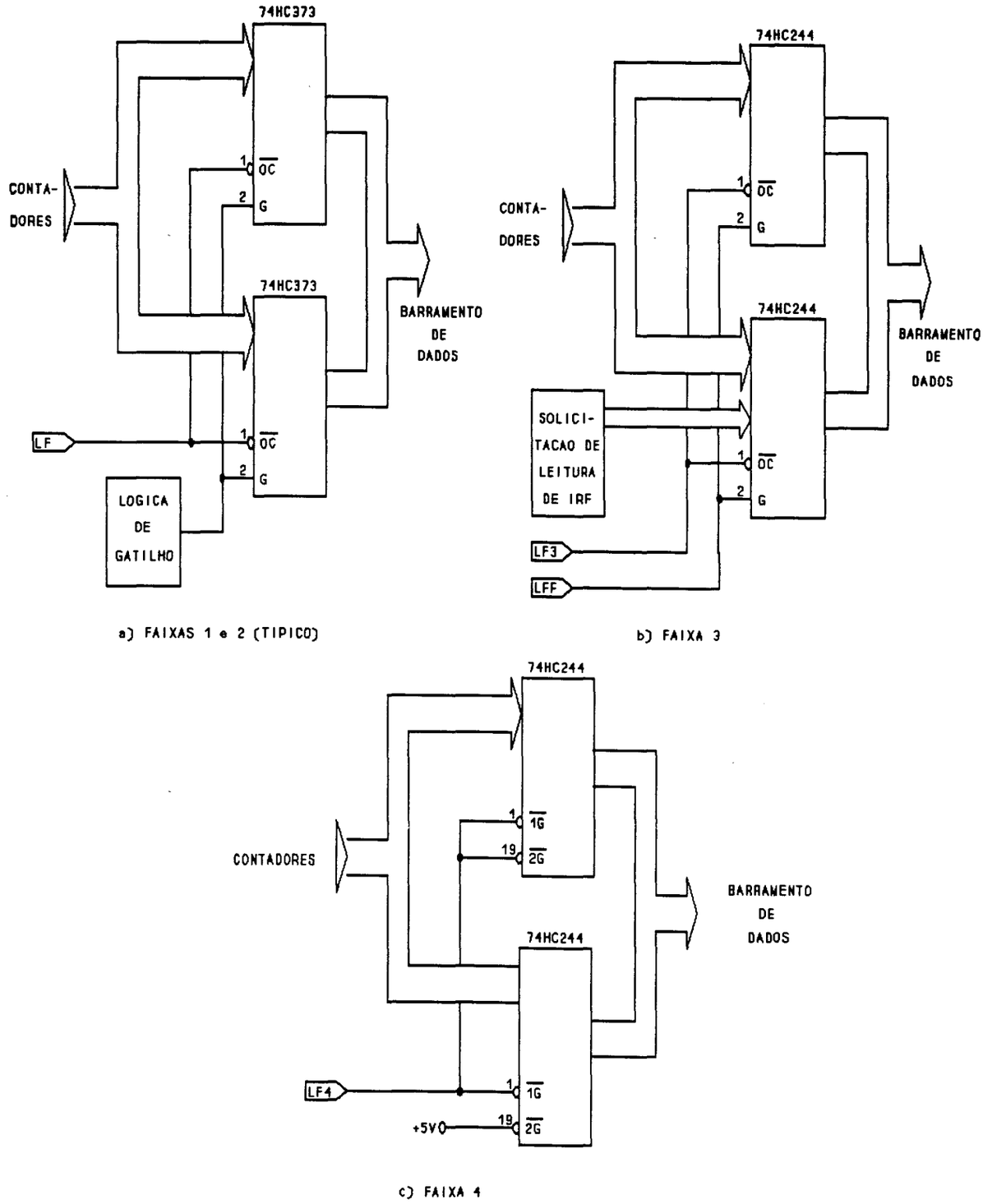


FIGURA 3.8 - REGISTRADORES DE PULSOS DOS ANALISADORES DE FREQUÊNCIA

3.4 ANÁLISE DE PULSOS

A forma de onda devida a uma descarga atmosférica depende das componentes ativa e reativa do circuito desde a entrada do prédio até o ponto de serviço. Assim diferentes pontos poderão receber diferentes formas para o mesmo transiente externo (McLoughlin, 1987). Para os pontos mais próximos da entrada, espera-se uma forma de onda conforme a apresentada na Figura 3.9(a), ao passo que para pontos mais distantes é mais provável que haja uma ocorrência do tipo mostrado na Figura 3.9(b).

A tensão de pico positivo poderá ser de até 6kV. Acima deste valor haveria arco entre as partes da rede (fase-fase, fase-neutro, fase-terra) (McLoughlin, 1987).

A tensão de pico negativo na forma de onda apresentada pela Figura 3.9(b) pode atingir valores de até 60% do pico positivo, ou seja, 3,6 kV.

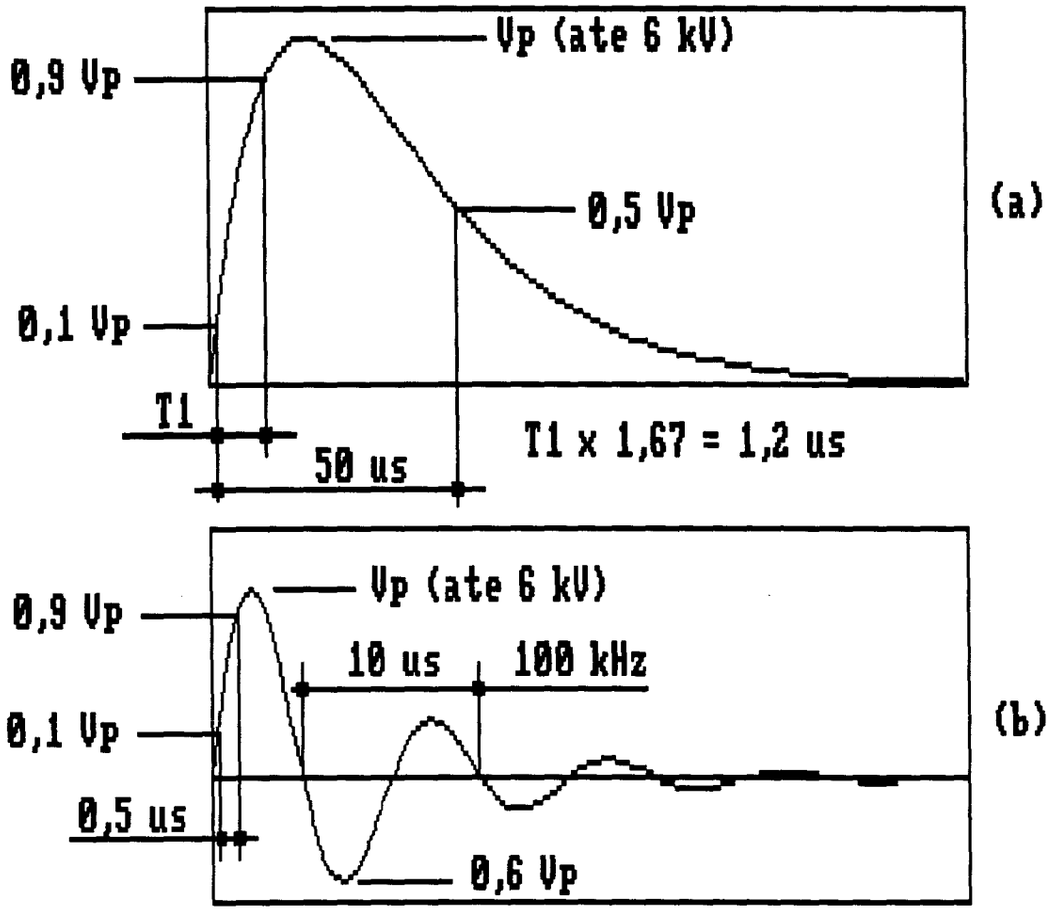


FIGURA 3.9 - FORMAS DE ONDA DEVIDAS A UMA DESCARGA ATMOSFÉRICA

Para dimensionamento do circuito de análise de pulsos, foram tomados como referência os dados levantados por Martzloff (1970). Os casos extremos por ele apresentados, estão nas Tabelas 3.2, 3.3 e 3.4.

TABELA 3.2 - SURTOS OSCILATÓRIOS MAIS SEVEROS

	CRISTA (V)	DURACAO (μ)	FREQUÊNCIA (kHz)	CAUSA
Maior pico	5600	64000	500	Descarga atmosférica
Maior Frequencia	750	20	2000	Partida de lâmpada fluorescente

TABELA 3.3 - SURTOS OSCILATÓRIOS MAIS FREQUENTES

	CRISTA (V)	DURACAO (μs)	FREQUÊNCIA (kHz)	CAUSA
Maior pico	2000	16000	250	Caldeira a óleo
Maior frequência	500	20	2000	Partida de lâmpada fluorescente

TABELA 3.4 - SURTOS UNIDIRECIONAIS (MAIOR PICO)

	CRISTA (V)	DURACAO (μs)	CAUSA
Mais severos	2700	9	Descargas atmosféricas
Mais frequêntes	900	5	Descargas atmosféricas

Baseando-se nesses dados, foram adotadas para análise de pulsos de curta duração as seguintes referências:

- Maior freqüência : 2 MHz
- Duração : 20 μs
- Valor de pico : 6 KV

O circuito de análise de pulsos compõe-se como pode ser visto na Figura 3.10, dos seguintes blocos:

a) Filtro Passa-alta
b) Registrador de Picos Positivos
c) Registrador de Picos Negativos
d) Detetor de Pulsos
e) Medidor de Duração
f) Medidor de Freqüência

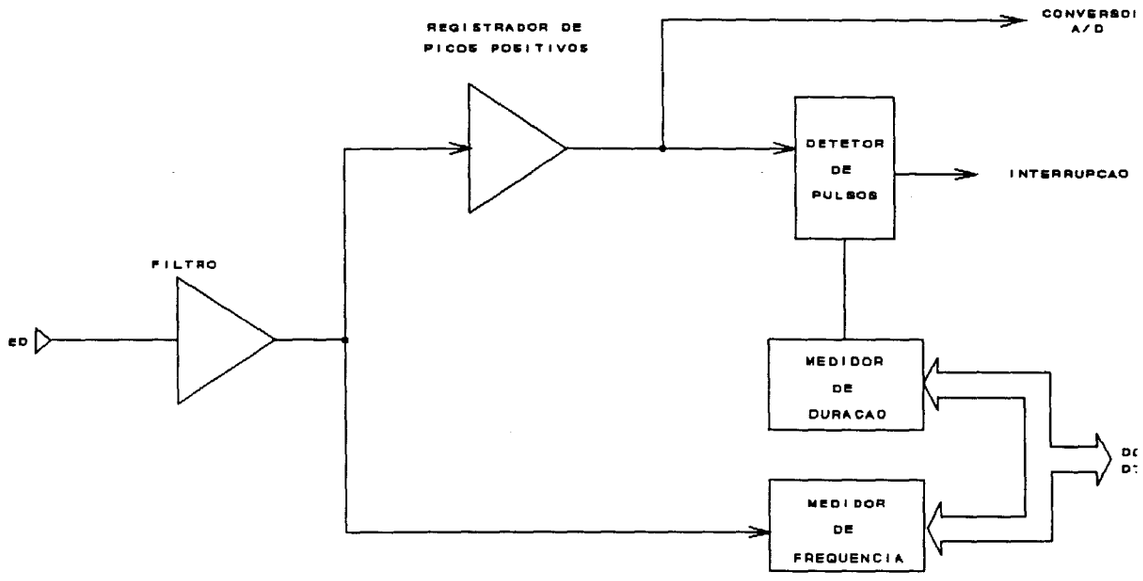


FIGURA 3.10 - DIAGRAMA DE BLOCOS DO CIRCUITO DE ANÁLISE DE PULSOS

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

3.4.1 FILTRO PASSA-ALTA

Devido ao valor mais elevado de freqüências esperadas para este circuito, podendo ir até 2 MHz ou mais, utilizou-se um amplificador com uma banda mais larga, o amplificador operacional LM 318 (National, 1989), com 12 MHz. Como no caso de descargas atmosféricas os surtos oscilatórios possuem em geral, uma freqüência de 100 KHz (Mc Loughlin, 1987), o filtro foi projetado com f_c = 90 KHz.

3.4.2 REGISTRADORES DE PICOS DOS TRANSIENTES

O registrador de picos dos transientes é basicamente, o conhecido circuito detetores de pico onde um capacitor é carregado por um seguidor de tensão, sendo sua carga mantida por meio da polarização reversa de um diodo e da alta impedância de entrada de um segundo seguidor de tensão (Figura 3.11).

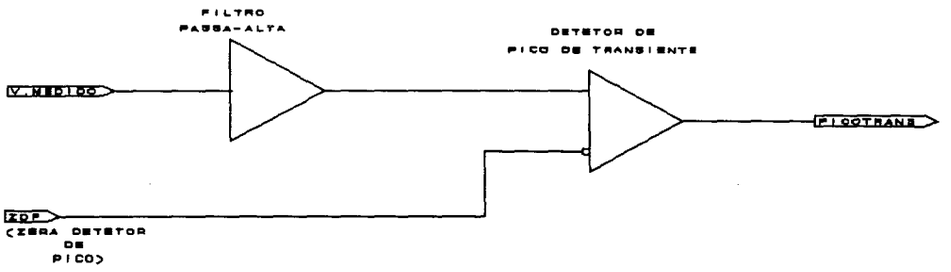


FIGURA 3.11 - CIRCUITO DETETOR DE PICO EM PULSOS TRANSIENTES

3.4.3 DETETOR DE PULSOS

O detetor de pulsos tem como função produzir uma interrupção na presença de um transiente. Como pode ser visto na Figura 3.12,

o elemento sensor do detetor é composto de dois comparadores que verificam se há algum pico armazenado nos registradores de pico.

Como as descargas atmosféricas podem produzir distúrbios oscilatórios de freqüência compatível com a faixa 4, a ocorrência de uma IRF daquela grandeza poderia disparar o sensor inadvertidamente.

Para evitar esses disparos inadequados foi acrescentado após o sensor um circuito temporizador composto de um multivibrador monoestável e um "flip-flop". Após a atuação dos comparadores este temporizador produz um atraso de aproximadamente 32 ms. Durante este tempo, caso o disparo tenha sido devido a um sinal de IRF, os registradores de pico serão zerados pela rotina que servir à interrupção correspondente (IR4).

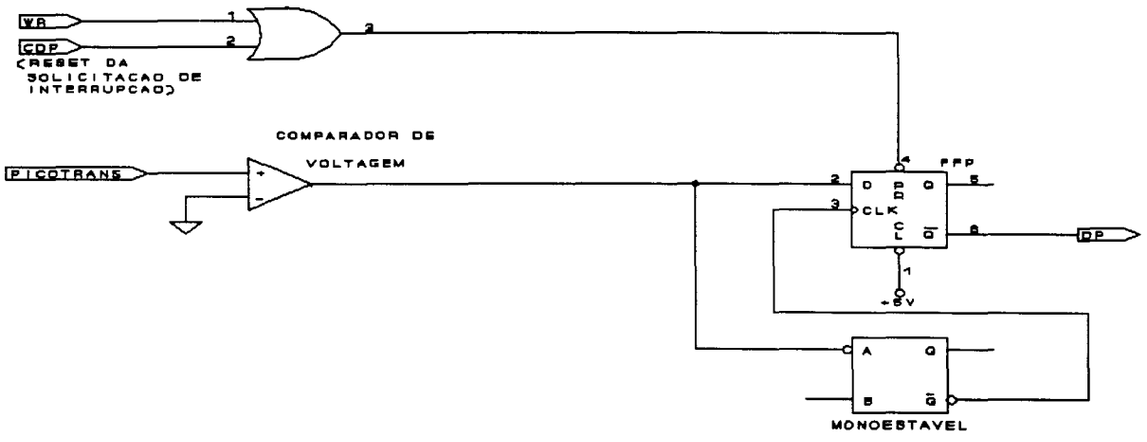


FIGURA 3.12 - CIRCUITO DE SOLICITAÇÃO DE INTERRUPÇÃO PARA LEITURA DE PULSOS TRANSIENTES

3.4.4 MEDIDORES DE DURAÇÃO E FREQÜÊNCIA

Os medidores de duração e freqüência utilizam os mesmos princípios dos contadores de pulsos para IRF, com pequenas alterações na parte de totalização e sem a necessidade da lógica envolvida naqueles casos, já que não há, aqui, necessidade de comparação com nenhum valor pré-determinado.

Adotou-se novamente o 82C54 por ser muito flexível e não exigir circuitos adicionais. A duração do pulso (DP) é contabilizada pelo "timer 1" e os pulsos pelo "timer 0".

Para medição da DP foi necessário abrir uma janela durante o intervalo em que o transiente estivesse presente. A lógica envolvida, a partir do sinal que dispara o monoestável do detetor de pulsos, habilita o gatilho G1 do 82C54. A medição da freqüência foi feita contando-se os pulsos do transiente dentro do tempo de duração, sendo o cálculo da mesma feito pelo programa de emissão de relatórios no PC.

A mesma lógica que habilita o contador da DP é responsável pela habilitação e bloqueio do contador para medição de freqüência, como pode ser visto no desenho MNT-ESQ-PL4, Folha 4/5.

3.5 CONTROLE MICROPROCESSADO

O controle microprocessado é composto de:

a) Microprocessador
b) Interface de Demultiplexação
c) Banco de Memória ROM
d) Banco de Memória RAM Auxiliar
e) Sistema de Controle de Interrupções

3.5.1 MICROPROCESSADOR

Para minimizar os tempos de fluxo de dados foi escolhido um processador de 16 bits e que pudesse trabalhar com um oscilador mais elevado: o 80C86-2 (Intel, 1987b).

Sendo o 80C86-2 um processador com barramento de endereços e dados multiplexado até o bit 15 (AD15), houve necessidade de criar uma interface de demultiplexação, explicada no ítem 3.5.2. A Figura 3.13 apresenta o oscilador e o microprocessador com suas entradas e saídas.

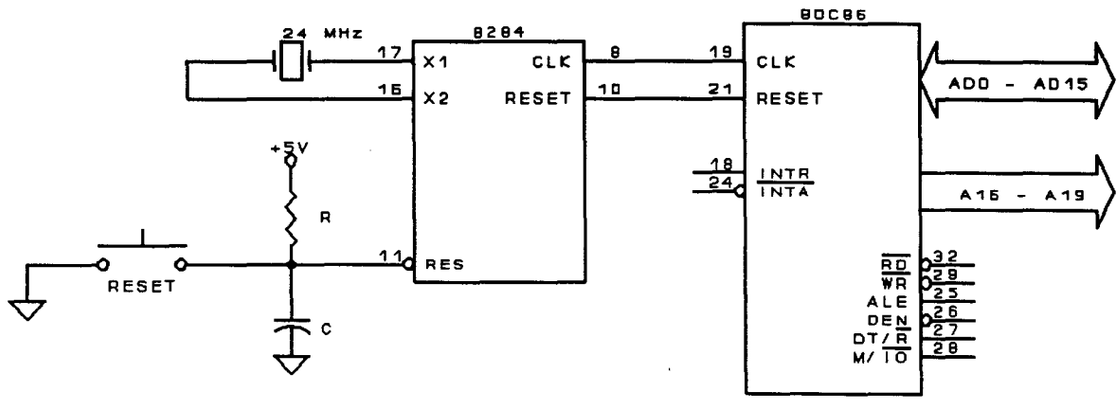


FIGURA 3.13 - MICROPROCESSADOR, GERADOR DE CLOCK E RESET DO SISTEMA

3.5.2 INTERFACE DE DEMULTIPLEXAÇÃO

Conforme já foi citado, no microprocessador 80C86-2, dados e endereços até o BIT 15 aparecem no mesmo pino, porém em momentos diferentes, sendo que o endereço ocorre primeiro. Para mantê-lo estável até o final da leitura ou escrita de um dado, deve-se armazená-lo em registradores.

Os dados por sua vez, devem ter a direção de seu fluxo controlada já que podem fazer parte de operações de escrita ou leitura. Quando ocorre uma leitura, para evitar que haja conflito entre dados e endereços nos pinos do microprocessador, primeiramente o endereço é armazenado nos registradores. A seguir, o barramento entra em um estado de alta impedância. Finalmente, os controladores de fluxo de dados ("transceivers") permitem que os mesmos possam fluir na direção determinada pelo processador.

Nas Figuras 3.14 e 3.15 pode-se ver que o sistema de demultiplexação foi duplicado. Isso se deve à presença de uma memória de aquisição.

Na Figura 3.15 pode-se ver ainda que a habilitação ou não depende de uma lógica aqui chamada de "Lógica do Tri-State". Ela verifica a presença da placa da memória de aquisição, ou melhor, verifica se a chave de habilitação da mesma está ligada ou não.

Nos desenhos do sistema , os barramentos e linhas de controle associados à memória de aquisição possuem o prefixo 3 (de "Tri-State").

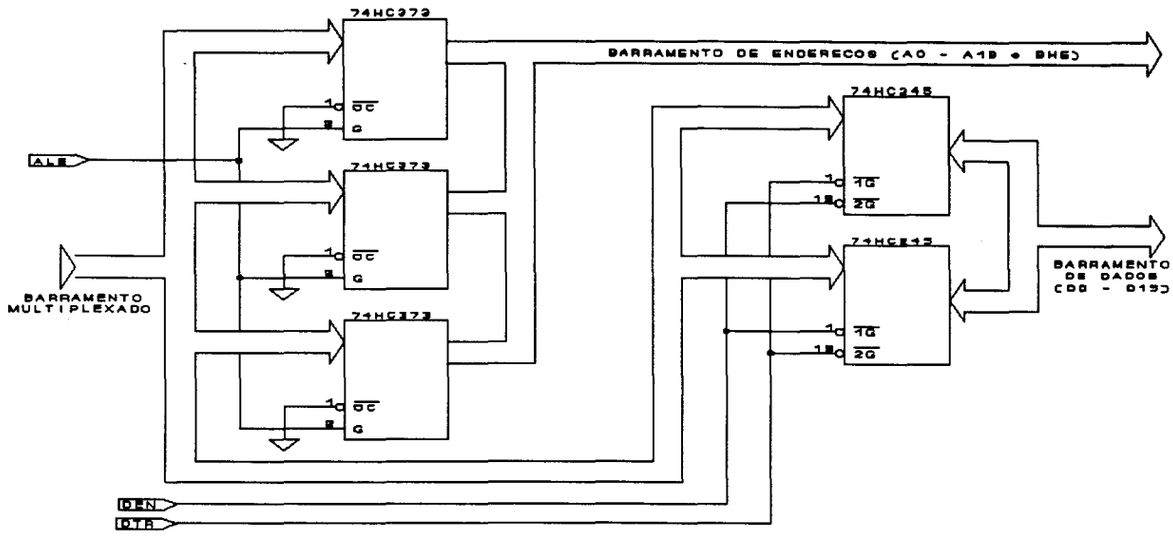


FIGURA 3.14 - SISTEMA DE DEMULTIPLEXAÇÃO PARA ROM E RAM AUXILIAR

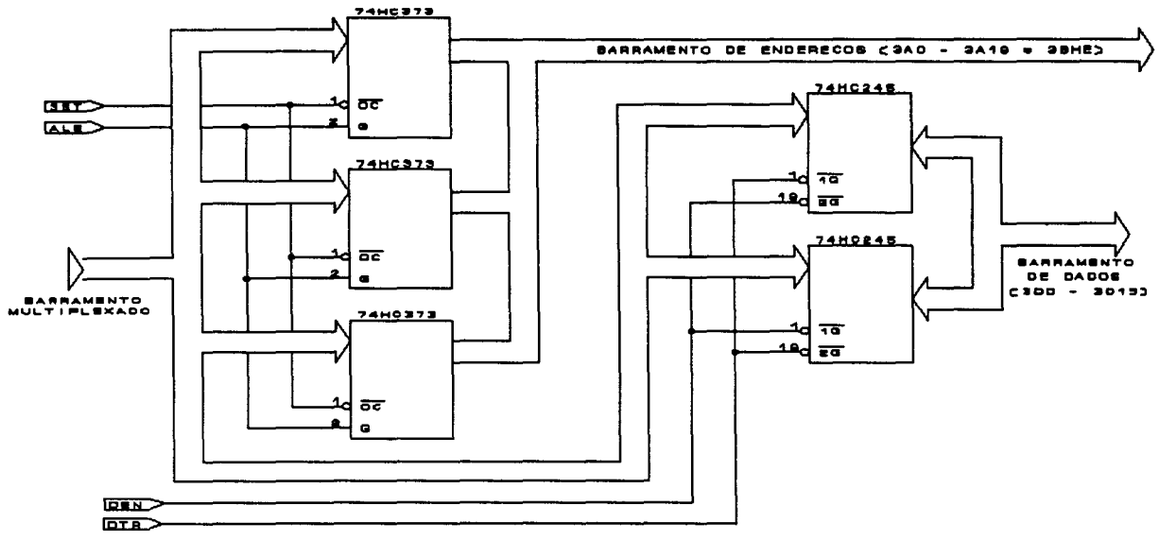


FIGURA 3.15 - SISTEMA DE DEMULTIPLEXAÇÃO PARA MEMÓRIA DE AQUISIÇÃO

3.5.3 BANCO DE MEMÓRIA

A memória permanente de programa está alocada nos endereços de F0000H a FFFFFH.

Dois programas foram armazenados nessa área:

- Monitor de Comunicação Serial
- Monitor de Ruídos

O primeiro não será discutido nesta dissertação por tratar-se simplesmente de ferramenta para desenvolvimento. O segundo programa é o responsável por toda a atividade do sistema.

A memória permanente é composta de circuitos integrados CMOS de 64 Kbytes, 27C64 (Intel, 1987d). Sua configuração pode ser vista no desenho MNT_ESQ_PL2, Folhas 7/17 e 8/17.

3.5.4 BANCO DE MEMÓRIA RAM AUXILIAR

O banco de memória auxiliar é responsável pelo armazenamento temporário de informações tais como atualização de calendário, horário, pilha, vetores de interrupção, variáveis, etc.. É composto de dois circuitos integrados 6416, de 8K x 8 bits, e está localizado, na parte mais baixa dos endereços. Ver mapa de memória no Apêndice E.

3.5.5 SISTEMA DE CONTROLE DE INTERRUPÇÕES

O sistema de controle de interrupções é composto de um gerenciador de interrupções 82C59 (Intel, 1987b), uma interface e um registrador de 8 bits (Figura 3.16).

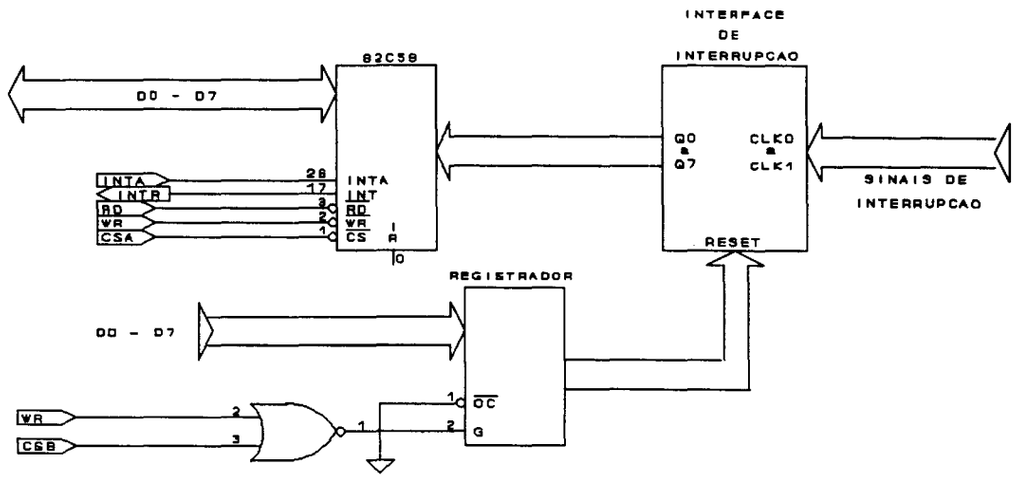


FIGURA 3.16 - SISTEMA DE CONTROLE DE INTERRUPÇÕES

O gerenciador de interrupções controla os seguintes eventos:

- Atualização da data (IR0)
- Atualização da hora (IR1)
- Atualização de minutos (IR2)
- Leitura de transientes (IR3)
- Leitura de IRF (IR4)
- Leitura de bateria baixa (IR5)
- Leitura do conversor AD (IR6)
- Leitura do teclado (IR7)

A interface é composta de oito "flip-flops" tipo D e sua função é manter o sinal de interrupção (IR0 - IR7) alto, de forma que o processador possa concluir a execução de uma instrução em andamento.

O latch tem como função abaixar a saída de cada "flip-flop" da interface ao final de um serviço de interrupção. Isto é obtido através da escrita de um dado onde o bit correspondente à rotina em andamento é zero. A seguir, o valor FFH é escrito recolocando o "flip-flop" em operação.

3.6 MEMÓRIA DE AQUISIÇÃO

Como a emissão dos relatórios das ocorrências de distúrbios é feita utilizando-se um microcomputador IBM PC-XT compatível, há necessidade de se fazer a transmissão dos dados do monitor para o micro.

Para evitar o transporte de um dos equipamentos foi necessária a criação de uma placa independente que contivesse uma memória não volátil, uma interface de comunicação e uma chave de habilitação. Por este motivo, a demultiplexacao de enderecos para esta memória é independente da adotada para a ROM e a RAM auxiliar.

Na figura 3.17 pode-se ver que a memória não volátil é composta de circuitos integrados tipo 6264, com tecnologia CMOS, e uma bateria de 3,6 V para manter os dados.

A interface é composta, basicamente, de dois controladores de fluxo de dados 74HC245 (Texas, 1988a) e um inversor. Ela possibilita a comunicação entre o PC-XT de 8 bits com o banco de memória de 16 bits.

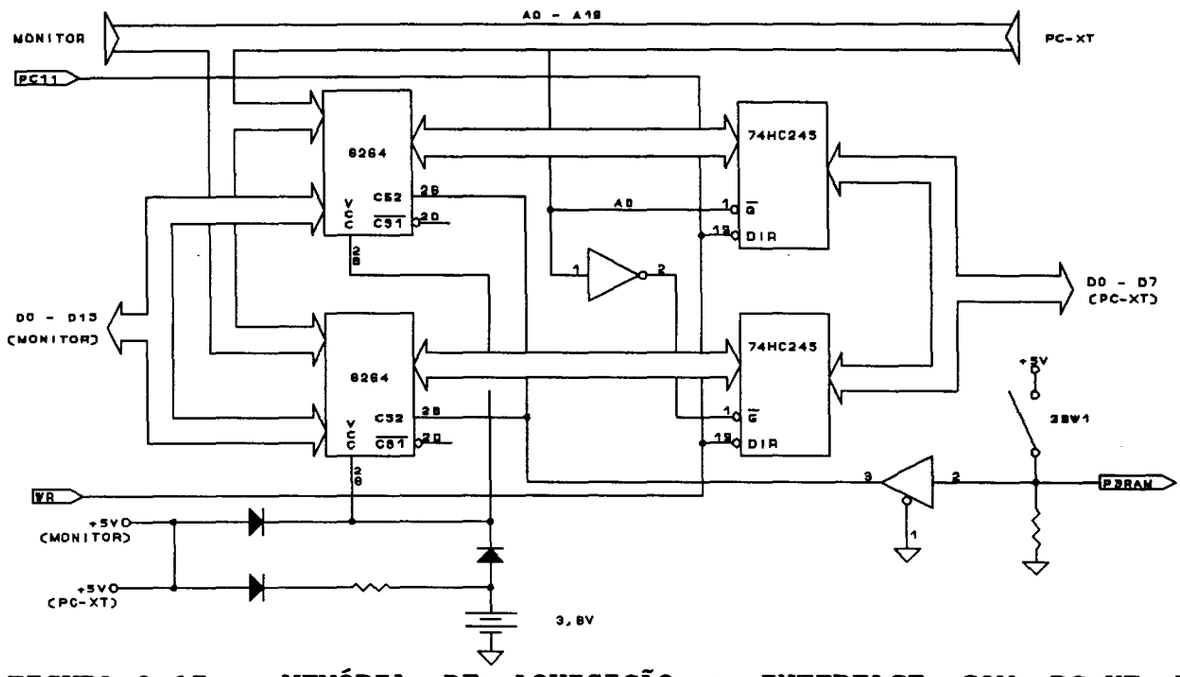


FIGURA 3.17 - MEMÓRIA DE AQUISIÇÃO : INTERFACE COM PC-XT E ALIMENTAÇÃO "STAND-BY"

A chave de habilitação 3SW1 é responsável pelo estado de repouso das memórias fazendo com que o consumo de energia seja reduzido. Assim, com a placa alimentada pela bateria, a informação poderá ser mantida durante dias.

A outra função de 3SW1 é providenciar um sinal ao circuito do processador para colocar em alta impedância os registradores de endereço e os controladores de fluxo de dados.

3.7 COMUNICAÇÃO SERIAL

Durante a fase de desenvolvimento, para dar suporte a este processo, foi criada uma porta de comunicação serial que, a princípio, foi idealizada apenas para este fim, mas que está disponível para comunicação do equipamento com um microcomputador.

Essa porta utiliza um cicuito integrado 82C54 (Intel, 1987c) como gerador de taxa de transmissão ("baud-rate"), um ICL232 (Intersil, 1987) que faz a conversão de nível TTL para o padrão EIA e uma controladora de comunicação serial 8251 (Intel, 1987a).

3.8 INTERFACE HOMEM-MÁQUINA

A interface homem-máquina do monitor de ruídos é realizada através do teclado e do visor de quatro dígitos.

3.8.1 TECLADO

O controle do teclado poderia ter sido efetuado com a aplicação de uma controladora de teclado e display como a 8279 (Intel, 1987c). Porém, devido ao seu alto consumo (120 mA) optou-se pela elaboração de um circuito empregando integrados SSI e MSI com tecnologia CMOS.

Como pode ser visto na Figura 3.18, este circuito é composto de :

- Oscilador de 100 kHz
- Contador Binário de 4 bits
- Decodificador 2 linhas para 4 linhas
- Dois registradores de 8 bits para armazenar código da tecla ("latches")
- Circuito supressor de falsos contatos

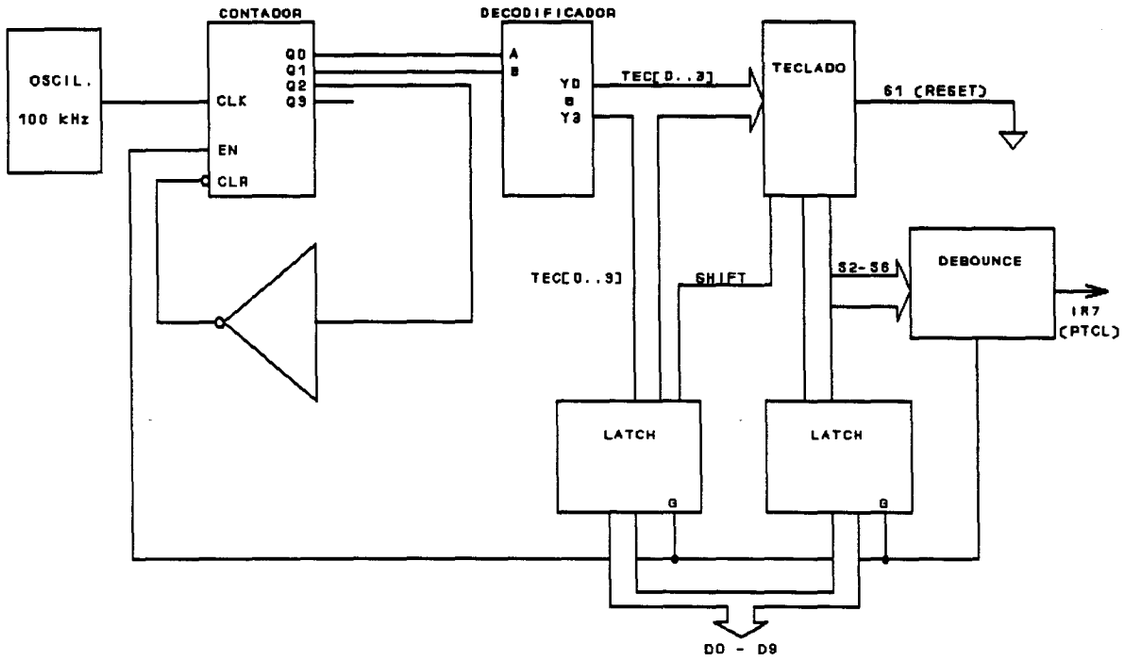


FIGURA 3.18 - CONTROLADOR DE TECLADO

O oscilador do teclado foi obtido dividindo-se a freqüência do sistema por 80, através de um contador programável, o circuito integrado 82C54 (Intel, 1987c).

A disposição das teclas pode ser vista na Figura 3.19.

SHIFT	RESET
F5	F6
F3	F4
F1	F2

7	8	9
4	5	6
1	2	3
ANULA	0	ENTRA

FIGURA 3.19 - ARRANJO DO TECLADO

As funções das teclas são apresentadas na Tabela 3.5.

TABELA 3.5 - FUNÇÕES DAS TECLAS

TECLA	FUNÇÃO
F1	Acerto de relógio
SHIFT F1	Disponível
F2	Pausa
SHIFT F2	Fim de pausa
F3	Acerto de data
SHIFT F3	Leitura de data
F4	Início de aquisição
SHIFT F4	Fim de aquisição
F5	Leitura de bateria positiva
SHIFT F5	Leitura de bateria negativa
F6	Leitura de memória disponível
SHIFT F6	Teste de visor
ANULA	Cancela comando
SHIFT ANULA	Cancela último caracter
ENTRA	Completa entrada de dado
0 - 9	Teclas numéricas

3.8.2 VISOR

O visor de quatro dígitos tem como função a apresentação dos seguintes eventos:

- Eco de Tecla
- Horário
- Data
- Memória Disponível
- Nível de Bateria

O visor adotado é composto de quatro dígitos de sete segmentos em cristal líquido, três pontos decimais e dois pontos centrais que permitem a apresentação de horário.

Seu acionamento é feito através do controlador de visor ICM7211 (Intersil, 1987). Este controlador possui 28 saídas para 4 dígitos de 7 segmentos, uma saída comum (BP - "back plane"), barramento para entrada de dados de 4 bits, barramento para seleção de dígitos (4 bits) e um pino de controle para aplicação com mais de 4 dígitos, ou seja, em cascata.

3.9 RELÓGIO

O relógio do sistema é composto praticamente de um único circuito integrado : o 82C54. Este componente possui três contadores programáveis.

Os contadores desse componente foram programados para dividir a freqüência de 1 Hz da entrada, respectivamente, por 60, 60 e 24, gerando, assim, os pulsos de segundos, minutos e horas.

A freqüência de 1 Hz, necessária ao relógio, foi obtida a partir da divisão da freqüência de varredura do teclado (100 KHz), utilizando-se dois "flip-flops" e um outro 82C54.

A atualização do horário apresentado no visor, bem como na memória, é feita através de sinais de interrupção da seguinte forma:

- Pulso de Minuto - atualiza visor e minutos na memória
- Pulso de Hora - atualiza hora na memória
- Pulso de Data (24 horas) - atualiza a data na memória

A razão de se manter informações sobre o horário na memória é a agilização quando da necessidade de registro de histórico de ocorrências de ruídos. Caso contrário, a cada ocorrência seria necessário ler todos os temporizasores do relógio. Da forma adotada, apenas a leitura de segundos é necessária.

4 DESCRIÇÃO FUNCIONAL DOS PROGRAMAS

O conjunto de programas do monitor de ruídos é composto de:

a) Aquisição;
b) Transferência;
c) Relatório.

O programa Aquisição é responsável pelo registro dos distúrbios e seus históricos na memória de aquisição.

Uma vez concluída essa aquisição, a placa onde esses dados estão armazenados é desconectada do monitor e conectada a um IBM/PC. As informações contidas nesta placa são lidas e armazenadas em disco. Esta operação é realizada pelo programa Transferência.

Por último, o programa Relatório lê os dados do disco, podendo imprimi-los um a um ou na forma de uma sinopse.

Para desenvolvimento dos programas foi adotada a linguagem C. A maior parte das tarefas foi resolvida utilizando-se essa linguagem de forma altamente satisfatória, uma vez que a mesma possui poderosas ferramentas para operações de entrada e saída e manipulação de bits, além de fazer uma interface extremamente simples com a linguagem Montadora.

Nos casos em que as funções da biblioteca C eram muito extensas, optou-se pela sua substituição por rotinas em linguagem montadora como pode ser visto, por exemplo, nas rotinas output() e input() (Ver Apêndice A). Suas fontes não foram geradas puramente em linguagem montadora, mas sim, suas instruções foram inseridas no próprio programa em linguagem C.

4.1 AQUISIÇÃO

O programa Aquisição, além de registrar os distúrbios, é responsável por todo o funcionamento do monitor. Pode-se dizer que o registro dos distúrbios é o produto final, mas que antes disso,

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

todo sistema precisa ser colocado em operação através das devidas inicializações, atualizações, cálculos, serviços de interrupções e leituras. Esta é a tarefa do programa Aquisição.

O programa Aquisição foi dividido nos seguintes arquivos:

a) MONITOR.C : programa principal
b) IOMON.C : rotinas de entrada e saída
c) CONVERT.C : rotinas de conversão
d) RWMEM.C : rotinas de escrita e leitura nas RAM's
e) AD.C : rotinas para tratamento de amplitudes
f) IRF.C : rotinas para tratamento de RFI
g) TECDISP.C : rotinas de teclado e display
h) RELOGIO.C : rotinas do relógio do sistema
i) MAT.C : rotinas matemáticas
j) DEFINE.H : arquivo de definição de macros

4.1.1 PROGRAMA PRINCIPAL (MONITOR.C)

O arquivo MONITOR.C compõe-se exclusivamente do programa principal "main()" que é responsável pelas seguintes tarefas:

a) Inicialização das Variáveis Globais do Sistema;
b) Habilitação de Interrupções do 80C86;
c) Teste do Visor;
d) Programação dos Temporizadores do Relógio;
e) Inicialização dos Vetores de Interrupção;
f) Inicialização da Controladora de Interrupções;
g) Habilitação dos "Flip-flops" de Interrupção, Teclado e Bateria;
h) Laço Eterno e Leitura de Tecla.

Estas tarefas são explicadas a seguir.

4.1.1.1 INICIALIZAÇÃO DAS VARIÁVEIS GLOBAIS DO SISTEMA

Todas as variáveis que necessitam manter seu valor dentro e fora das rotinas ou seja, variáveis globais, são inicializadas neste trecho do programa principal.

4.1.1.2 HABILITAÇÃO DE INTERRUPÇÕES DO 80C86

Para que o 80C86 possa reconhecer um pedido de interrupção, deve-se habilitar o seu "flip-flop" interno responsável pelo reconhecimento da informação contida no pino INTR ("interrupt request" - solicitação de interrupção). Isto é conseguido através da execução da função enable() da biblioteca do Turbo C.

4.1.1.3 TESTE DO VISOR

O visor é testado através da rotina pisca (38H), que faz piscar todos os segmentos dos quatro dígitos e os três pontos centrais. Este procedimento verifica, também, se o programa está sendo inicializado.

4.1.1.4 PROGRAMAÇÃO DOS TEMPORIZADORES DO RELÓGIO

Os temporizadores do relógio do sistema são programados com a chamada da rotina timer_rel () do arquivo relogio.c . Neste ponto ainda não é feito o acerto do relógio.

4.1.1.5 INICIALIZAÇÃO DOS VETORES DE INTERRUPÇÃO

O 80C86 possui um mecanismo peculiar de tratamento de interrupções. Ao ocorrer uma interrupção, a controladora 82C59 envia um sinal (INTR) ao microprocessador que responde com dois

sinais de reconhecimento da solicitação (INTA - "interrupt aknowledgement"). A controladora coloca então, no barramento de dados um número que identifica um vetor no primeiro Kbyte do mapa de memória. Este vetor é composto de quatro bytes que formam o endereço inicial da rotina que servirá a interrupção em questão.

Na inicialização dos vetores de interrupção, os endereços das rotinas correspondentes serão colocados nas posições indicadas na Tabela 4.1.

TABELA 4.1 - VETORES DE INTERRUPÇÃO DO PROGRAMA AQUISIÇÃO

NÚMERO DO VETOR	ENDEREÇO DO VETOR	ROTINA DE INTERRUPÇÃO
40H	100H	atual_data
41H	104H	atual_hora
42H	108H	atual_min
43H	10CH	dur_pulso
44H	110H	ler_irf
45H	114H	nivel_bat
46H	118H	ad_conv
47H	11CH	teclado

As funções das rotinas serão apresentadas juntamente com seus arquivos correspondentes.

O endereço a partir do qual os vetores foram colocados (100H) devem-se ao fato de que a área de 0 - FFH foi reservada à pilha e aos vetores de interrupção do programa de comunicação serial.

O número do vetor é o dado passado pela 82C59 ao microprocessador que o multiplica por quatro obtendo assim o endereço do vetor.

Para a inicialização dos vetores foi utilizada a rotina vetor() que será discutida juntamente com arquivo RWMEM.C

4.1.1.6 LAÇO ETERNO E LEITURA DE TECLADO

O laço eterno é composto de dois grupos de instruções. No primeiro, encontra-se um segundo laço que testa uma variável que sinaliza quando há tecla digitada. Quando isso ocorre, o laço é abandonado e o programa passa a executar o segundo grupo de instruções, ou seja, o reconhecimento da tecla que foi pressionada e o atendimento do seu respectivo serviço.

Após executar o serviço de teclado, o horário volta a ser exibido no VISOR e o laço eterno se reinicia.

4.1.2 MÓDULO IOMON.C

O arquivo IOMON.C contém as rotinas de entrada e saída do monitor, ou seja, de escrita e leitura dentro da área de 64 Kbytes referentes a periféricos (INTEL, 1987b). Compõe-se das seguintes rotinas ou funções:

a) output ()
b) input ()
c) input_int ()

4.1.2.1 FUNÇÃO OUTPUT ()

A função output () escreve um byte em um porto. As entradas para a mesma são o endereço do periférico e o dado a ser escrito no mesmo. Seu formato (Schildt, 1980) é apresentado abaixo:

*void output (unsigned *end_porto, char dado)*

O valor de saída (dado) foi declarado como char por ocupar apenas um byte.

4.1.2.2 FUNÇÃO INPUT ()

Esta função, cujo formato é *char input (unsigned *end_porto)*, de modo inverso a output (), lê um byte de um porto. Sua entrada é o endereço do porto e a saída é o dado contido no mesmo.

4.1.2.3 FUNÇÃO INPUT_INT ()

Semelhante a input (), porém seu retorno é um inteiro não sinalizado, ou seja, seu formato é *unsigned input_int (unsigned *end_porto)*.

4.1.3 MÓDULO CONVERT.C

Este módulo é responsável pelas conversões de parâmetros, geralmente alterando o seu formato real para um formato próprio para apresentação em display ou, em outros casos, o inverso. É composto das funções :

a) hexbcd ()
b) intbcd ()

4.1.3.1 FUNÇÃO HEXBCD ()

A função hexbcd () converte um inteiro compreendido entre 0 e FH, em sua representação BCD de dígitos entre 00 e 15. Seu formato é *char hexbcd (int hexa)*.

4.1.3.2 FUNÇÃO INTBCD ()

Semelhante à função anterior, porém sua entrada é um inteiro não sinalizado compreendido entre 0 e 9999D que é convertido para

uma cadeia de 4 dígitos. Seu formato é *void intbcd (unsigned entrada, char *saída)*.

4.1.4 MÓDULO RWMEM.C

O módulo RWMEM.C contém as rotinas que irão tratar dados da memória de aquisição e inicialização da tabela de vetores de interrupção. São elas :

a) vetor ()
b) grava_3ram ()
c) mostra_memória ()
d) diretório ()

4.1.4.1 FUNÇÃO VETOR ()

A função vetor () grava o endereço de uma rotina de interrupção. Seu formato é *void vetor (int intr, void interrupt (*isr) ())* onde, intr é o número do vetor de interrupção, ou seja, o número que ao ser multiplicado por quatro, conforme já citado na descrição de PCI2 no ítem 4.1.1.6, fornecerá o endereço do vetor na RAM auxiliar.

O parâmetro *isr é o ponteiro que indica o início da rotina.

4.1.4.2 FUNÇÃO GRAVA_3RAM ()

Esta função, cujo formato é *void grava_3RAM (unsigned *end, unsigned dado)*, escreve um inteiro não sinalizado (dado) na memória de aquisição na posição indicada pelo ponteiro *end.

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

4.1.4.3 FUNÇÃO MOSTRA_MEMÓRIA ()

A função mostra_memória () indica no display o número de Kbytes ainda disponível na memória de aquisição. Seu formato é *void mostra_memória (void)*. O número apresentado vai de 0 a 99.99 Kbytes, sendo a resolução de 0,01 Kbytes ou seja 10 bytes e um erro de 0,24/10,24 = 2,34%.

4.1.4.4 FUNÇÃO DIRETÓRIO ()

Esta função, cujo formato é *void diretório (unsigned momento)*, é responsável pela construção do diretório do arquivo de aquisição. A sua chamada ocorre apenas em duas ocasiões: quando solicitado o início da aquisição e ao se encerrar a mesma, seja por solicitação, ou por algum evento determinado pelo programa.

Na primeira hipótese *(momento = 0)*, são gravados na memória de aquisição a data e o horário do início do período de coleta. No segundo caso *(momento = 1)*, além de registrar o momento do término, são reportados ainda os níveis das baterias neste instante e o endereço do último dado do arquivo.

4.1.5 MÓDULO AD.C

O módulo AD.C contém todas as rotinas relativas à conversão análogo-digital, que são:

a) nível_bat ()
b) ler_bateria ()
c) mostra_bateria ()
d) ad_conv ()

4.1.5.1 FUNÇÃO NIVEL_BAT ()

A função nível_bat () é uma rotina de serviço de interrupção (IR5) e sua chamada ocorre quando o nível de qualquer das baterias cai abaixo de um valor pré-determinado. Seu formato é *void interrupt nivel_bat (void)*.

Ao ser chamada, nível_bat () registra no arquivo de aquisição o nível da bateria que provocou a interrupção e o momento da ocorrência.

4.1.5.2 FUNÇÃO LER_BATERIA ()

Esta função, cujo formato é *char ler_bateria (int tipo_bateria)*, pode ser chamada em quatro situações distintas:

- No fim da aquisição
- Na solicitação de leitura via teclado
- Na ocorrência de nível baixo de tensão

A entrada tipo_bateria, que é nula para bateria positiva e unitária para negativa, é um dos parâmetros do cálculo do endereço a ser fornecido ao multiplexador. Este endereço determina qual das duas será lida.

A saída da função é o nível da bateria em oito bits. O tratamento que será dado a este valor dependerá da forma que ocorreu a chamada de ler_bateria ().

No primeiro caso acima citado, os valores binários dos níveis das baterias positiva e negativa serão compostos e armazenados no endereço D000CH conforme descrito em 2.2.

No segundo caso é feita a conversão para decimal e apresentada no visor. Isto será detalhado em 4.1.5.3.

Na terceira e última situação, o valor binário é composto com a informação de tipo de distúrbio para ser armazenada na memória de aquisição.

4.1.5.3 FUNÇÃO MOSTRA_BATERIA ()

A função mostra_bateria () conforme citado em 4.1.5.2 atende à solicitação de leitura de bateria (teclas F5 e SHIFT F5 no programa principal).

Através da chamada da função div_flut () é feita uma simulação de ponto flutuante com duas casas decimais e apresentada no visor.

O seu formato é *void mostra_bateria (char binario)*, onde *binário* é o valor em oito bits do nível a ser apresentado.

4.1.5.4 FUNÇÃO AD_CONV ()

A conversão análogo-digital de um sinal, utilizando-se um circuito integrado ADC0804 (National, 1984), como é o caso deste projeto, compreende quatro etapas:

- Envio de um sinal de escrita ao conversor
- Desenvolvimento da conversão propriamente dita
- Emissão de um sinal de interrupção informando fim de conversão
- Leitura do byte representando o valor convertido

As rotinas que se utilizam desse serviço têm o seguinte procedimento básico: emitem o sinal de escrita; entram em um laço onde é testado um indicador de fim de conversão e ao se verificar a sua alteração, abandonam o laço e efetuam a leitura do ADC0804.

A tarefa de ad_conv () é exatamente mudar esse indicador, o ad_flag, para um. Seu formato é *void interrupt ad_conv (void)*.

4.1.6 MÓDULO IRF.C

O módulo IRF.C contém as rotinas relativas às interferências de radio-freqüência na faixa de 600 Hz a 2,5 MHz bem como ao acompanhamento da freqüência da rede. Estas rotinas são:

a) ler_irf ()
b) escr_irf ()
c) intervalo ()

4.1.6.1 FUNÇÃO LER_IRF ()

A função ler_irf () é uma rotina de interrupção que é chamada em uma das seguintes situações:

- Ao final de cada período da freqüência da rede para contabilização da mesma
- Na ocorrência de uma IRF

Ao ser solicitada, ler_irf () primeiramente faz a leitura do registrador que contém a(s) faixa(s) solicitante(s). A seguir, ler_irf(), através de uma chamada a escr_irf (), analisa a validade de registro do valor da freqüência e efetua o mesmo em caso positivo.

O formato dessa função é *void interrupt ler_irf (void)*.

4.1.6.2 FUNÇÃO ESCR_IRF ()

Conforme descrito no item anterior, a função escr_irf (), cujo formato é *void escr_irf (char faixa)*, analisa a validade do registro de uma freqüência, efetuando o mesmo em caso afirmativo.

São dois os critérios para essa análise:

- Verificar se a freqüência em questão já solicitou registro
- Caso a hipótese acima seja confirmada verificar há quanto tempo foi feita essa solicitação, para evitar que o mesmo ruído seja armazenado indevidamente.

A freqüência é registrada quando seu valor difere do anterior em dois pulsos na base de tempo ou dois pulsos do oscilador de referência e quando sua última ocorrência for anterior a um intervalo de dois minutos em relação à atual solicitação.

4.1.6.3 FUNÇÃO INTERVALO ()

A função intervalo () cujo formato é *int intervalo (unsigned ano_ant, unsigned mes_ant, unsigned dia_ant, unsigned hora_ant, unsigned min_ant)* é encarregada de calcular a decorrência de tempo entre dois momentos.

Devido ao critério de tempo citado em 4.1.6.2, não havia necessidade de que intervalo () possuísse uma saída capacitada a valores muito grandes. Assim, ficou estabelecido um limite próximo a um inteiro sinalizado, adotando-se uma resposta máxima de 31680 minutos (aproximadamente 22 dias).

Isso não limita os valores de entrada que podem ser quaisquer, sendo que, sempre que o intervalo for superior a 22 dias, o número retornado será 31680.

4.1.7 MÓDULO TECDISP.C

O módulo TECDISP.C contém todas as rotinas relativas ao teclado e ao visor. É composto das seguintes funções:

a) display ()
b) atraso ()
c) pisca ()
d) teclado ()

e) tecdis ()
f) tecen ()
g) dadoduplo ()

4.1.7.1 FUNÇÃO DISPLAY ()

A função display () mostra uma cadeia de quatro caracteres no visor de cristal líquido. Seu formato é *void display (char *dado)*.

Cada elemento de dado[i] corresponde a um dígito da direita para a esquerda e com i variando de 0 a 3.

Um laço que conta até 800 garante o intervalo exigido pelo visor entre a escrita de dois dígitos consecutivos (1200 μs mínimo).

4.1.7.2 FUNÇÃO ATRASO ()

A função atraso () foi criada inicialmente para em conjunto com o circuito supressor de falsos contatos garantir a atuação correta das teclas. No desenvolvimento do programa, no entanto, várias outras atividades de visor passaram a exigir um tempo de espera. Por exemplo a apresentação da data no visor. Primeiramente são apresentados o dia e o mês. Após um certo intervalo, produzido por atraso (), mostra-se o ano.

O formato de atraso é *void atraso (int mult)*.

4.1.7.3 FUNÇÃO PISCA ()

A função pisca () foi criada para que um dado seja apresentado no visor piscando.

Várias são suas aplicações:

- teste dos segmentos do visor
- verificar se a tecla de início de aquisição foi reconhecida
- idem para fim de aquisição
- verificar se o programa principal foi inicializado

Seu formato é *void pisca (char caracter)*.

4.1.7.4 FUNÇÃO TECLADO ()

O serviço de teclado é executado de forma análoga à conversão A/D : um laço testa um sinalizador até que, devido a uma interrupção produzida pela digitação de uma tecla, o mesmo seja alterado.

A função de teclado (), cujo formato é *void interrupt teclado (void)*, é exatamente esta: alterar tecla_flag, o sinalizador em questão.

4.1.7.5 FUNÇÃO DADODUPLO

Nos serviços de introdução de data e de horário foi necessária a criação de uma rotina que simultaneamente recebesse, via teclado, a entrada destes dados e os exibisse no visor.

Dadoduplo () desempenha essa tarefa, recebendo como entrada um caracter, que servirá de "prompt" para identificar a entrada solicitada, e o endereço da cadeia de dois caracteres que conterá as duas dezenas entradas.

Inicialmente, dadoduplo () preenche os quatro dígitos com o "prompt" fornecido. A seguir dadoduplo () recebe os caracteres via teclado da esquerda para a direita, estes caracteres vão sendo substituidos pelo dígito entrado e o mesmo vai sendo totalizado na respectiva dezena.

Na hipótese de ser pressionado SHIFT ANULA, o último dígito é substituído pelo "prompt" e subtraído da dezena.

Esses dois procedimentos (inserção e cancelamento) acabam por simular a existência de um cursor.

Se durante a inserção de um dado for pressionada a tecla ANULA, dadoduplo () é abandonada sem a totalização do dado, ou melhor, a cadeia conterá uma informação sem sentido. Portanto, à continuação do programa é feito um teste desta condição da tecla para que o dado seja considerado ou desprezado.

O formato de dadoduplo () é *void dadoduplo (char prompt, char dezena [2])*.

4.1.8 MÓDULO RELÓGIO.C

O módulo RELÓGIO.C contém as rotinas referentes ao relógio do sistema, que são:

a) div_freq ()
b) timer_rel ()
c) ler_timer ()
d) disp_rel ()
e) acerto ()
f) atual_min ()
g) atual_hora ()
h) novo_horário ()

4.1.8.1 FUNÇÃO DIV_FREQ ()

O contador programável 82C54 para ser utilizado precisa antes ser inicializado, uma vez que pode trabalhar em vários modos (INTEL, 1987c).

A tarefa da função div_freq () é programar esses contadores. Seu formato é *void div_freq (unsigned *end_contr, unsigned *end_dado, char modo, unsigned divisor)*, onde:

- end_contr : endereço do contador para escrita do modo
- end_dado : endereço do contador para escrita do divisor
- modo : modo de operação do contador (contador, gerador de taxa de transmissão, divisor de freqüência, gerador de pulso único, gerador de gatilho)
- divisor : valor pelo qual o contador irá dividir o "clock" de entrada

Os dois endereços diferentes para o mesmo contador (end_contr e end_dado) devem-se ao fato de que o 82C54, além de possuir um pino de CS ("chip-select" - seleção do integrado), possui também as entradas A0 e A1 que definem se o dado sendo escrito é palavra de controle ou um divisor.

4.1.8.2 FUNÇÃO TIMER_REL ()

A função timer_rel () inicializa os três contadores do 82C54 do relógio e um contador que fornece a base de tempo de 1s para o contador do relógio.

O formato de timer_rel () é *void timer_rel (void)*.

4.1.8.3 FUNÇÃO LER_TIMER ()

A função ler_timer () efetua a leitura de qualquer dos três contadores de um 82C54. Inicialmente é lido o "status" do 82C54 para verificar qual o modo em que o mesmo está operando. Determina-se, assim, se será lido apenas um byte ou uma palavra.

Seu formato é *int ler_timer (unsigned *end_timer)*.

4.1.8.4 FUNÇÃO DISP_REL ()

A função disp_rel (), cujo formato é *void disp_rel (char opcao)*, exibe o horário no visor.

O parâmetro "opção" determina a forma da apresentação:

- 0 : são apresentados horas e minutos
- 1 : são apresentados minutos e segundos

4.1.8.5 FUNÇÃO ACERTO ()

A função acerto () atualiza os contadores do relógio.

Como os contadores trabalham com contagem regressiva, não são entrados os valores diretamente mas sim seus complementos. Em outras palavras, embora o formato de acerto () seja *void acerto (char horas, char minutos)*, os parâmetros realmente fornecidos são "24-horas" e "60-minutos".

Para efetuar o acerto do relógio, a base de tempo é acelerada, sendo o divisor do contador de base alterado para 1. Assim, a base muda de 1s para 26 μs. Com isso, os tempos máximos dispendidos para este acerto serão:

- $60 \times 60 \times 26 \times 10^{-6} = 93{,}6$ ms para minutos
- $60 \times 60 \times 24 \times 10^{-6} = 2{,}2$ s para horas

A esses tempos deve-se somar, ainda, a duração de execução da rotina atraso (200) ao final de dadoduplo () quando da entrada do novo horário. Os tempos dos demais passos do programa são desprezíveis quando comparados com o valor para atraso () que é de 1,85 s. Tem-se, assim, um total de 4,14 s.

Ao se igualarem os contadores aos valores relativos ao novo horário, a base de tempo reassume seu valor inicial.

4.1.8.6 FUNÇÕES ATUAL_MIN () E ATUAL_HORA ()

As funções atual_min () e atual_hora () são duas rotinas de interrupção que atualizam as variáveis que contém respectivamente o minuto e a hora.

Além da variável hora, atual_hora () atualiza também a memória de aquisição no mesmo endereço em que estava armazenado o horário anterior, caso não tenha ocorrido distúrbios, ou em endereço novo, caso contrário.

O formato dessas duas funções são:

- *void interrupt atual_min (void)*
- *void interrupt atual_hora (void)*

4.1.8.7 FUNÇÃO NOVO_HORÁRIO ()

A função novo_horário () é responsável pela entrada de um novo horário para acerto do relógio.

Conforme já citado anteriormente, o horário é inserido através da rotina dadoduplo ().

Caso durante a inserção seja digitada a tecla "ANULA", nenhuma alteração é feita e o programa retorna ao bloco principal.

Seu formato é *void novo_horario (void)*.

4.1.9 MÓDULO MAT.C

Embora a biblioteca da linguagem C possua todas as rotinas matemáticas que esse projeto necessitou, elas se utilizam de outros procedimentos da própria linguagem C e/ou do sistema, sendo gerados, assim códigos extensos e lentos.

Por esse motivo foram criadas algumas funções específicas contidas no módulo MAT.C. São elas:

a) pot10 ()
b) div_flut ()

4.1.9.1 FUNÇÃO POT10 ()

A função pot10 (), cujo formato é *long pot10 (int n)* eleva o número 10 à potência indicada por n. O número devolvido é um inteiro do tipo long, ou seja, a entrada máxima será 9, acarretando em uma saída máxima igual a 1 000 000 000.

4.1.9.2 FUNÇÃO DIV_FLUT ()

A função div_flut simula uma divisão com saída em ponto flutuante em formato adequado ao visor, ou seja, XX.XX.

O resultado da divisão é colocado em uma cadeia de caracteres.

O formato de div_flut é *void div_flut (unsigned dividendo, unsigned divisor, char *saida)*.

4.1.10 MÓDULO DEFINE.H

Este módulo contém apenas as definições dos macro-identificadores do programa AQUISIÇÃO, sendo incluído como "header" nos demais módulos (Shildt,1990).

4.2 TRANSFERÊNCIA

O programa Transferência, que permanecerá no PC, é responsável pela leitura dos dados da placa 3, ou memória de aquisição,

armazenando-as em disco flexível ou em disco rígido. Este programa é composto de um único módulo contendo duas rotinas.

- main () - programa principal
- le_ram () - rotina de transferência dos dados

4.2.1 PROGRAMA PRINCIPAL

O programa executa apenas três tarefas:

- Solicita a entrada do nome do local do levantamento
- Define o endereço inicial do arquivo de aquisição, ou seja D000:0000
- Chama a rotina de leitura da placa 3.

4.2.2 FUNÇÃO LERAM ()

A função le_ram () é responsável pela leitura propriamente dita.

Inicialmente, le_ram () solicita os nomes do local do levantamento e do arquivo de saída, podendo constar do segundo a definição da unidade de disco.

A seguir le_ram () lê o diretório do arquivo de aquisição e já obtém o endereço de fim deste arquivo.

Finalmente é feita a leitura do restante do arquivo e inserido o nome do local.

O formato de le_ram () é *void le_ram (unsigned *memoria)*.

4.3 RELATÓRIO

Uma vez estando os dados gravados em discos, pode ser efetuada a emissão de relatórios sobre os mesmos.

O programa RELATÓRIO desempenha essa tarefa, imprimindo uma sinopse dos distúrbios ou uma lista detalhada dos mesmos. No primeiro caso é emitida uma tabela contendo apenas tipo e quantidade e, no segundo, são impressos o tipo, a data, o horário, os níveis e/ou freqüência das anomalias.

Esse programa é composto de dois módulos:

a) RELAT.C
B) ROT_REL.C

4.3.1 MÓDULO RELAT.C

O módulo RELAT.C contém apenas o programa principal do relatório, main ().

As principais tarefas desempenhadas por main () são:

- Alocação de memória para importar dados do disco
- Criação do canal de comunicação com a impressora
- Definição da forma de impressão
- Impressão do cabeçalho do relatório
- Contabilização dos dados
- Fechamento do canal de comunicação com a impressora
- Liberação da área de memória alocada para os dados

4.3.2 MÓDULO ROT_REL.C

O módulo ROT_REL.C contém as rotinas do programa Relatório que são:

a) le_disco ()
b) dist60 ()

c) rf1 ()
d) rf2 ()
e) rf3 ()
f) trans ()
g) ampl60 ()
h) pbat ()
i) nbat ()
j) hora ()
l) ano ()
m) dir_arq ()
n) aj_pag ()
o) aj_pag2 ()

4.3.2.1 FUNÇÃO LE_DISCO ()

A função le_disco () é responsável pela transferência dos dados do disco para a memória do PC-XT alocada no início do programa principal.

Seu formato é *int le_disco (void)* e o valor retornado é o número de dados lidos.

4.3.2.2 FUNÇÕES DIST60 (), RF1 (), RF2 () E RF3 ()

Estas funções fazem a contabilização dos distúrbios registrados pela placa de análise de freqüência, a saber :

- *void dist60 (void)* - desvios na freqüência de 60 Hz
- *void rf1 (void)* - interferência de radio-freqüência na faixa de 600 Hz a 10 KHz
- *void rf2 (void)* - interferência de radio_freqüência na faixa de 10 KHz a 100 KHz
- *void rf3 (void)* - interferência de radio_freqüência na faixa de 100 KHz a 2,5 MHz

4.3.2.3 FUNÇÃO TRANS ()

A função trans () é responsável pela contabilização dos distúrbios transientes.

Seu formato é *void trans (void)*.

4.3.2.4 AMPL60 (), PBAT () E NBAT ()

As funções acima são responsáveis pela contabilização das ocorrências relativas a amplitude, a saber:

- *void ampl60 (void)* - sub ou sobretensão na tensão nominal da rede
- *void pbat (void)* - nível baixo da bateria positiva
- *void nbat (void)* - nível baixo da bateria negativa

4.3.2.5 FUNÇÕES HORA (), MES_DIA (), ANO ()

Estas funções são responsáveis pela atualização, durante a elaboração do relatório, das variáveis que contêm os parâmetros que lhes deram nome. Seus formatos são:

- *void hora (void)*
- *void mes_dia (void)*
- *void ano (void)*

4.3.2.6 FUNÇÃO DIR_ARQ ()

A função dir_arq () é responsável pelo reconhecimento e impressão dos dados contidos no diretório do arquivo de aquisição.

Seu formato é *void dir_arq (void)*.

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

4.3.2.7 FUNÇÕES AJ_PAG () E AJ_PAG2 ()

Estas funções contabilizam o número de linhas já impressas em uma página para verificar se o próximo dado ainda poderá ser impresso antes que seja feita a mudança de página.

A diferença entre as duas é que aj_pag () faz essa verificação entre distúrbios e aj_pag2 () entre o último distúrbio e a sinopse.

Seus formatos são *void aj_pag (void)* e *void aj-pag2 (void)*.

5 RESULTADOS

Dois tipos de testes foram efetuados sobre o sistema:

a) Testes de Desenvolvimento
b) Testes de Simulação de Anomalias

5.1 TESTES DE DESENVOLVIMENTO

Os testes de desenvolvimento foram efetuados ao longo de todo o projeto com o objetivo de verificar o funcionamento parte por parte dos circuitos e do programa.

Os procedimentos e os resultados para esses testes são apresentados a seguir.

5.1.1 PLACA 1

5.1.1.1 FILTROS PARA INTERFERÊNCIAS DE RÁDIO-FREQÜÊNCIA

Para efetuar os testes dos filtros para interferência de rádio-freqüência foram injetados sinais senoidais de 5V de amplitude utilizando-se gerador de funções SGF 2000B descrito no apêndice C. Esses filtros foram montados em circuito impresso para evitar-se fugas ao longo da placa.

5.1.1.2 DETETORES DE ZERO

Os detetores de zero foram testados em conjunto com os filtros. Conforme citado anteriormente esses detetores foram primeiramente montados utilizando-se comparadores de voltagem com saída em coletor aberto ligado a resistores "pull-up".

Esses comparadores apresentaram problemas nos dois tipos de montagem, provocando instabilidade nos circuitos lógicos a eles ligados, provavelmente devido a sinais espúrios.

A solução para esse problema foi substituir os comparadores por circuitos utilizando dois inversores "Schmitt-Trigger". Os testes dos detetores de zero conduziram aos resultados apresentados na Tabela 5.1.

TABELA 5.1 - RESULTADOS DOS TESTES DOS DETETORES DE ZERO DA PLACA 1

Faixa Nominal (Hz)	Faixa com Presença de Pulsos (Hz)
60	até 145
600 - 10k	520 - 14,5k
10k - 100k	8k - 117k
100k - 1M	até 850k

O fato das faixas reais extrapolarem as nominais não causa problemas porque os comparadores lógicos só permitem que haja interrupção dentro dos valores estipulados.

A faixa 4, apesar de utilizar um filtro passa-alta, cortou num limite superior de 850 kHz. Este fato pode ter sido causado pelas características do operacional, que possui uma banda de 4,5 MHz para laço fechado, mas que deve ser reduzida quando realimentado na configuração desse filtro.

5.1.1.3 CIRCUITOS LÓGICOS DA PLACA 1

Os testes dos circuitos lógicos foram efetuados através do próprio programa de comunicação entre o PC e o sistema e também utilizando-se um gerador de palavras de 20 bits,construído com 20 resistores "pull-up" e duas chaves do tipo "DIP" com dez contatos, conforme indicado na Figura 5.1.

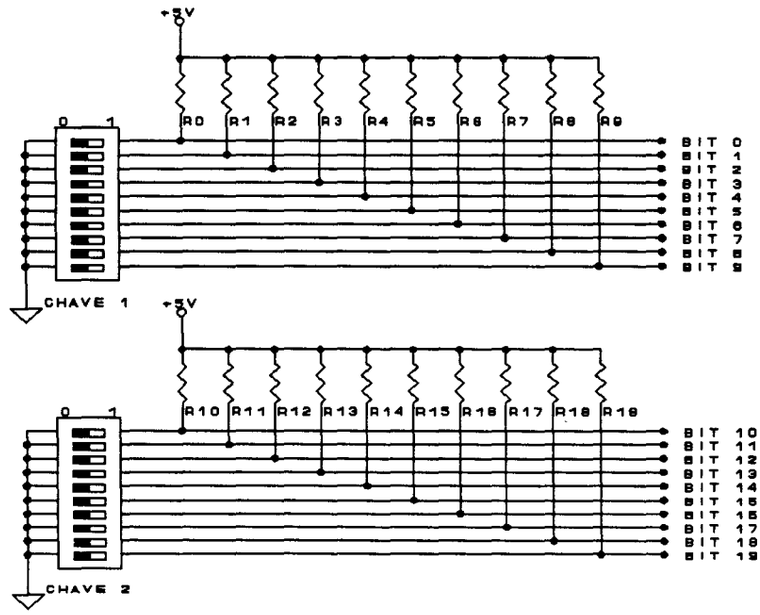


Figura 5.1 - Gerador de Palavras

5.1.2 PLACA 2

5.1.2.1 DECODIFICADORES

Os decodificadores da placa 2 foram testados, inicialmente, ou seja, antes de ser possível a comunicação com o PC, utilizando somente o gerador de palavras.

Após se estabelecer a comunicação, os testes foram confirmados via comandos entrados pelo teclado e análise dos pulsos de saída por um analisador digital.

5.1.2.2 MICROPROCESSADOR E MEMÓRIAS

Os testes do microprocessador e das memórias foram efetuados através da criação de pequenas rotinas de programação e verificação dos sinais de controle pelo analisador digital.

5.1.2.3 SISTEMA DE INTERRUPÇÕES

Para testar o sistema de interrupções foram criadas rotinas simples e injetados sinais de interrupção produzidos pelo gerador de palavras. Tais rotinas, para confirmar o funcionamento, apresentavam alguma informação no visor de quatro dígitos. Por exemplo: 1 1 1 1.

5.1.2.4 TECLADO E VISOR

O visor foi testado diretamente com o programa de comunicação, através de comandos entrados pelo teclado do PC. Para testar o teclado do sistema, foi criada uma rotina que reconhecia a tecla digitada, mostrando um código correspondente no visor.

5.1.2.5 RELÓGIO

O relógio do sistema já foi testado diretamente com a rotina para produzir horário. Inicialmente o horário de acerto já estava contido em uma variável do programa. Após o desenvolvimento de novas rotinas de teclado, foi possível testar o relógio fazendo acertos via teclado.

5.1.3 PLACA 3

Os testes da placa 3 foram similares aos da placa 2, sendo primeiramente utilizado o gerador de palavras para avaliar os decodificadores e, a seguir, os endereços foram injetados com os comandos do programa de comunicação.

Novamente utilizando o programa de comunicação, foram efetuadas operações de escrita e leitura para verificação das memórias.

Adicionalmente, foram efetuados testes de escrita e leitura conectando-se a placa diretamente ao barramento do PC.

5.1.4 PLACA 4

5.1.4.1 DETETOR DE PRESENÇA DE TRANSIENTES

Para testar o detetor de presença de transientes foram injetados sinais de freqüências a partir de 100 kHz. A configuração final do detetor é um resultado desses testes.

5.1.4.2 CIRCUITO DE INTERRUPÇÃO DE TRANSIENTES

Primeiramente foi testado o circuito comparador desse bloco, utilizando-se o sinal de saída do circuito anterior. O conjunto completo foi testado com subrotinas simples e um analisador digital.

5.1.4.3 CIRCUITO DE CONVERSÃO ANÁLOGO-DIGITAL E DECODIFICADORES

Através dos comandos do programa de comunicação, foram testados os decodificadores, o multiplexador e o seu registrador de endereços.

Esse procedimento também foi suficiente para testar o conversor análogo-digital (conversor A/D). Para tal, foram injetados sinais de níveis diferentes nas várias entradas do multiplexador, efetuando-se a leitura do conversor A/D.

5.1.4.4 CIRCUITO DE MEDIÇÃO DE TRANSIENTE

Somente a parte lógica foi testada a nível de desenvolvimento. Para isto, utilizou-se o programa de comunicação.

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

Os contadores programáveis foram testados através das rotinas do programa de aquisição, na fase de simulação.

Esses últimos testes levaram a alterações no circuito original, chegando-se à versão final apresentada no desenho MNT-ESQ-PL4, folha 4/8.

5.1.4.5 CIRCUITO DE LEITURA DE NÍVEL DE BATERIA

Para o circuito de leitura de nível de bateria, primeiramente foi feito o simples teste de variar a tensão de entrada e acompanhá-la na saída dos amplificadores.

A seguir foi desenvolvida a rotina de leitura das baterias, possibilitando o teste final desse circuito.

5.1.4.6 CIRCUITOS DETETORES DE PICO

Os circuitos detetores de pico, tanto para transiente como para 60 Hz, foram testados injetando-se os sinais nas faixas correspondentes. Para transientes, sinais acima de 100 kHz até 2,5 MHz.

A descarga dos capacitores dos circuitos de retenção era feita através de um comando do programa de comunicação.

Esses testes levaram a alterações na escolha dos amplificadores operacionais, chegando-se à configuração final com o LM318 na entrada e o AD515 na saída, que nos testes de desenvolvimento apresentaram um bom resultado. Porém, como poderá ser visto na apresentação dos resultados dos testes de simulação, o amplificador de entrada ainda não é suficientemente rápido. Uma opção poderia ser o OPA-111 (Burr Brown, 1989), porém seu custo é muito elevado.

5.1.4.7 CIRCUITO GERADOR DE TRANSIENTES

Esse circuito, na realidade, foi desenvolvido para atuar como ferramenta de testes, gerando os transientes.

Através de multivibrador monoestável 74121 (Texas, 1988b) e um trimpot, ajusta-se o tempo de duração do transiente. Este tempo pode ser verificado por um analisador digital.

A saída do circuito de controle de tempo, com o auxílio de um comparador de voltagem, aciona uma chave analógica 4066 (RCA, 1983) que habilita ou desabilita a passagem da onda.

Como gatilho dos trens de pulso, foi utilizada a saída de segundos do relógio do sistema.

5.2 TESTES DE SIMULAÇÃO

Os testes de simulação de anomalias tiveram como objetivo avaliar o funcionamento do conjunto circuitos/programas. Foram desenvolvidos separadamente, para cada tipo de distúrbio, de modo a possibilitar um maior controle dos parâmetros. Seus resultados são apresentados a seguir. Cabe salientar que apenas os valores relativos aos sinais serão expostos. Os dados de horário e diretório foram observados, representando fielmente o momento.

5.2.1 DESVIOS DA FREQÜÊNCIA DE 60 Hz

Os desvios da freqüência de 60 Hz foram simulados injetando-se sinais com freqüência de 50 Hz, 55 Hz, 60 Hz, 65 Hz e 70 Hz. O número de pulsos do oscilador de 125 kHz lidos durante esses testes e suas freqüências correspondentes são apresentados na Tabela 5.2.

TABELA 5.2 - RESULTADOS DOS TESTES DE SIMULAÇÃO DE DESVIO DA FREQUÊNCIA NOMINAL DA REDE (60 Hz)

FREQUÊNCIA INJETADA (Hz)	NÚMERO DE PULSOS	FREQUÊNCIA CORRESPONDENTE (Hz)	ERRO(%)
50	2516	49,68	-0,68
55	2277	54,90	-0,18
60	2074	60,27	+0,45
65	1937	64,53	-0,72
70	1790	69,83	-0,24

Os resultados são satisfatórios considerando-se que os erros estão dentro da casa decimal e que a freqüência injetada foi medida utilizando-se osciloscópio, o que pode introduzir erros devido a indefinição de espessura de traço e de escala.

5.2.2 INTERFERÊNCIAS DE RÁDIO-FREQÜÊNCIA

As interferências de rádio-freqüência foram simuladas injetando-se sinais na faixa de 1 kHz a 500 kHz. Os resultados são apresentados na Tabela 5.3.

TABELA 5.3 - RESULTADOS DOS TESTES DE SIMULAÇÃO DE INTERFERÊNCIA DE RÁDIO-FREQUÊNCIA

FREQUÊNCIA INJETADA (Hz)	PULSOS LIDOS	FREQUÊNCIA CORRESPONDENTE (Hz)	ERRO(%)
1740	1153	1735	-0,29
2500	802	2494	-0,24
4000	502	3984	-0,40
17390	167	17396	+0,03
17390	166	17292	-0,56
50000	475	49479	-1,04
147000	139	144790	-1,50
200000	190	197917	-1,04
500000	478	497917	-0,42

Os resultados para as interferências de rádio-freqüência apresentaram um grau de aceitabilidade semelhante ao obtido para desvio de 60 Hz. Foram notáveis, porém, os erros para as freqüências de 17390 Hz (2ª leitura) e para 50 kHz, 147 kHz e 200 kHz. Além do fator leitura com osciloscópio, importante também é o fato de que estas leituras, com exceção de 50 kHz, ocorrem próximas aos inícios das faixas 3 e 4. Isso causou obtenção de números pequenos de pulsos, influindo considerávelmente no erro. Este fato é bem ilustrado pelas duas leituras de 17390 Hz, onde a variação de um dígito no número de pulsos alterou o erro de +0,03% para -0,56%. Para minimizar este erro, a solução seria aumentar o número de faixas para leitura de IRF.

5.2.3 VARIAÇÕES NA AMPLITUDE DA REDE

Conforme mencionado no ítem 3.2.2 a sobretensão máxima adotada foi de ±269 V em amplitude (150% de 127x√2). Após passar pelos dois divisores idealizados, esta faixa corresponde à faixa de 0 a 5 Vcc na entrada do conversor A/D. Assim sendo, aos valores limites de sub e sobretensão (85% e 110% de 127 V_{RMS}, respectivamente) correspondem às tensões de 2,83 Vcc e 3,66 Vcc.

O circuito foi então submetido aos valores apresentados na Tabela 5.4.

TABELA 5.4 - RESULTADOS DOS TESTES DE SUB E SOBRETENSÃO

TENSÃO INJETADA (Vcc)	DADO LIDO NO CONVERSOR A/D	TENSÃO LIDA CORRESPONDENTE (Vcc)	ERRO(%)
2,8	143	2,80 V	0
4,0	203	3,98 V	-0,49

Os erros, como podem ser vistos, são aceitáveis, uma vez que seus valores estão abaixo de 1% e deve-se considerar, novamente, os desvios causados pela leitura da amplitude no osciloscópio.

5.2.4 TRANSIENTES

Os transientes conforme citado no ítem 5.1.4.7 foram simulados através de uma chave analógica e um multivibrador monoestável, permitindo que um trem de pulsos com duração e freqüência controladas fosse injetado no sistema. Os resultados são apresentados na Tabela 5.5, sendo a amplitude de entrada para todas as ondas de 5 Vcc e a duração de 58 μs.

TABELA 5.5 - RESULTADOS DA SIMULAÇÃO DE TRANSIENTES - VALORES LIDOS

ENTRADA	LEITURA		
FREQUÊNCIA (kHz)	PULSOS P/ DURAÇÃO	PULSOS P/ FREQUÊNCIA	LEITURA DO A/D
250	501	11	152
333	507	15	153
500	504	25	151
666	502	33	146

Conforme mencionado, anteriormente:

$$Td = N_{osc} * T_{osc}$$

$$f_L = \frac{N_L}{T_d}$$

Onde, Td : tempo de duração calculado

N_{osc} : nº de pulsos do oscilador de referência (8 MHz) lidos durante Td

T_{osc} : período do oscilador de referência

f_L : freqüência calculada

N_L : nº de pulsos da freqüência lidos durante Td

Para a amplitude tem-se:

$$V_p = \frac{N_{AD}}{255} * 5$$

Onde, V_p : valor de pico do transiente na escala de 0 a 5 Vcc

N_{AD} : valor lido no barramento de dados do conversor A/D

Aplicando-se esses cálculos sobre a Tabela 5.5 chegou-se à Tabela 5.6.

TABELA 5.6 - RESULTADO DA SIMULAÇÃO DE TRANSIENTES

ENTRADA	SAÍDA					
FREQUENCIA (kHz)	DURAÇÃO (μs)	FREQUÊN-CIA (kHz)	AMPLI-TUDE (Vcc)	e_D (%)	e_F (%)	e_A (%)
250	62,7	183	3,0	8,1	-27	-40
333	63,4	246	3,0	9,3	-26	-40
500	63,0	397	3,0	8,6	-21	-40
666	62,8	532	3,0	8,3	-20	-42

Onde, e_D :erro de duração
e_F :erro de freqüência
e_A :erro de amplitude

Embora os resultados estejam bem distantes da realidade, algumas correções podem ser feitas para minimizar este problema.

Primeiramente, atente-se para os sinais que ocorrem no circuito detetor de presença de transiente e no circuito contador de pulsos apresentados na Figura 5.2.

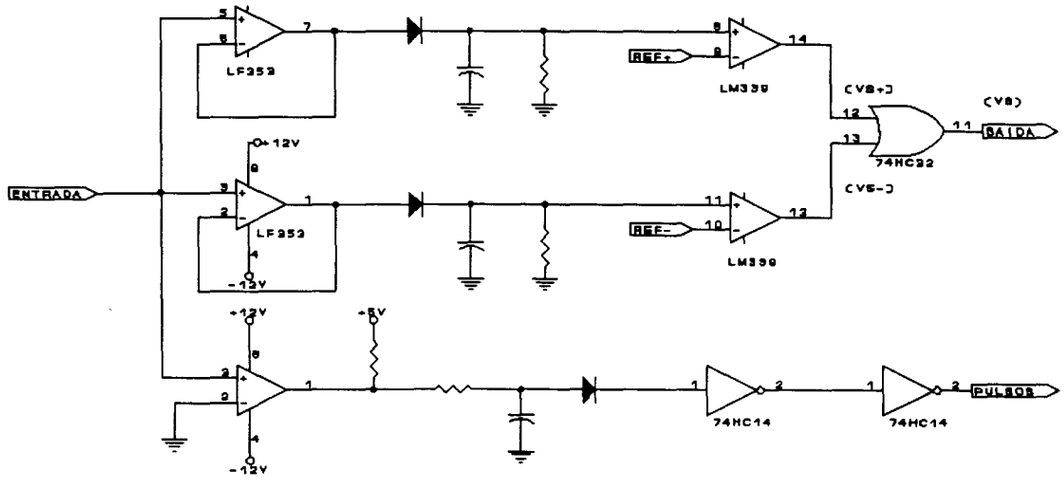


Figura 5.2 - Circuito detetor de transiente e circuito contador de pulsos

As formas de onda desses sinais são vistas na Figura 5.3.

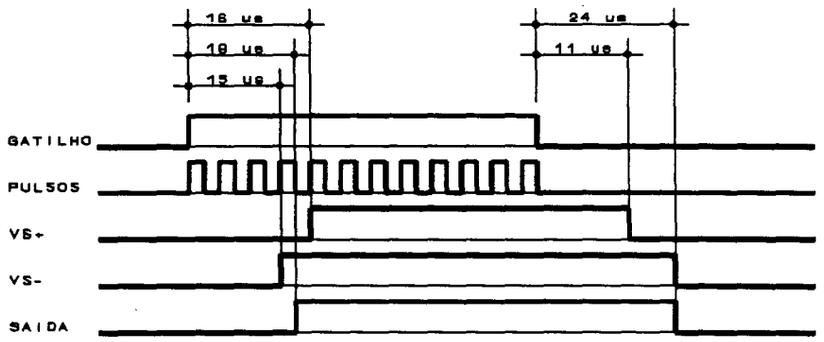


Figura 5.3 - Formas de onda observadas nos circuitos detetor de transiente e contador de pulsos

As ondas Vs+ e Vs- à saída dos comparadores representam as presenças das partes positiva e negativa do sinal de entrada, respectivamente.

Embora a onda de entrada fosse senoidal e os circuitos fossem semelhantes, as respostas não são iguais, havendo uma variação considerável nas durações desses dois sinais (Vs+ e Vs-). Em outras palavras, a confiabilidade nas características dos componentes é questionável. Esse fato causa a discrepância na medição da duração do transiente. Pior ainda, faz com que parte dos pulsos seja perdida, uma vez que o sinal "SAIDA" é quem determina o intervalo de aquisição. Todos os pulsos anteriores ao início desse sinal são perdidos, provocando o erro de medida da freqência.

A situação mais grave é a amplitude do transiente. A causa deste problema é facilmente localizável: a velocidade de resposta do amplificador de entrada do detetor de pico. A sua solução seria

investir em amplificadores de alta velocidade com custos mais elevados.

Além da correção desses desvios através da alteração dos circuitos ou da seleção de componentes, outra forma seria ajustar os dados através do programa de emissão de relatórios.

Para corrigir o tempo de duração com uma precisão aceitável seria necessários um levantamento estatístico. Com os dados apresentados na Tabela 5.6, considerando o valor médio para a duração de 63 μs, a correção seria de 0,92.

Nos levantamentos efetuados notou-se uma perda média de 4 pulsos na determinação da freqüência. Basta, portanto, adicionar este número à coluna "Pulsos p/ Freqüência" da Tabela 5.5.

A correção mais grave é a da amplitude. Embora, aparentemente, baste multiplicar o valor lido pela correção média de 1,8, seria necessária uma estatística bastante extensa para se chegar a um fator confiável.

Efetuando-se as correções por programa, acima sugeridas, chegou-se aos valores da Tabela 5.7.

TABELA 5.7 - RESULTADOS CORRIGIDOS DA SIMULAÇÃO DE TRANSIENTE

ENTRADA	SAIDA CORRIGIDA					
FREQ. (kHz)	DURAÇÃO (μs)	FREQ. (kHz)	AMPL. (Vcc)	e_D (%)	e_F (%)	e_A (%)
250	57,7	268	5,04	-0,5	+7,2	+0,8
333	58,3	326	5,04	+0,5	-2,1	+0,8
500	58,0	500	5,04	0	0	+0,8
666	57,8	640	4,86	-0,3	-3,9	-2,8

Obviamente, esses resultados ainda não são o ideal, havendo necessidade, conforme citado anteriormente, de um levantamento mais extenso para se obter uma boa estatística.

6 CONCLUSÃO

O capítulo 5 mostrou que o monitor para caracterização de ruídos apresenta resultados satisfatórios para o acompanhamento da frequência da rede, da amplitude da rede e das interferências de rádio-frequência na faixa de 1 kHz a 850 kHz.

Para a análise de interferências de curta duração, algumas alterações são necessárias, podendo, por exemplo ser utilizado o detetor de pico desenvolvido por Altafin, Janiszewski e Giacometi (1990). Podem associar-se, ainda, estas alterações, correções de leitura através do programa de emissão de relatórios.

Em suma, o monitor para caracterização de ruídos desenvolvido é capaz de determinar parâmetros que possibilitam a adoção de proteções para redes que alimentem equipamentos eletrônicos sensíveis.

Para continuidade do trabalho, sugere-se que seja desenvolvido um equipamento capaz de monitorar redes trifásicas. Para tal, pode-se adotar um sistema de acompanhamento das três fases simultaneamente, sendo a leitura dos diversos eventos multiplexada.

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

7 REFERÊNCIAS BIBLIOGRÁFICAS

ALTAFIN, R.A.C; JANISZEWSKI, J.M.; GIACOMETTI, J.A. Detetor de picos de correntes e tensões impulsivas usando eletreto. In: Seminário de Materiais no Setor Elétrico, 2, Curitiba, 1990, Curitiba, COPEL, 1990, p.14-19.

ANALOG, Integrated Circuits. Analog Devices Inc., 1984.

BURR-BROWN, Integrated Circuits Data Book. Tucson, AZ, Burr-Brown Co., 1989.

DANG, G. Ultra-isolation devices yield better protection from electrical noise. Computer Tecnology Review, p.187-194, prim. 1985.

HOROWITZ, P. & HILL, W. The art of eletronics. Cambridge, MA, E.U.A. Cambridge University Press, 1980, 716p.

INTEL, Microcommunications Handbook. Santa Clara, CA, Intel Co, 1987a.

INTEL, Microprocessor and Peripheral Handbook. Santa Clara, CA, Intel Co., v.1, 1987b.

INTEL, Microprocessor and Peripheral Handbook. Santa Clara, CA, Intel Co., v.2, 1987c.

INTEL, Memory Components Handbook. Santa Clara, CA, Intel Co., 1987d.

INTERSIL, Component Data Catalog. Cupertino, CA, Intersil Inc., 1987.

KEY, T.S. Diagnosing power quality-related computer problems. IEEE Transactions on Industry Application, v.IA-15, n.4, p. 381-393, ago. 1979.

MARTZLOFF, F.D. & GRUZS, T.M. Power quality site surveys: facts, fiction and fallacies. IEEE Transactions on Industry Applications, v. IA-24, n.6, p.1005-1018, nov./dez. 1988.

MARTZLOFF, F.D. & HAHN, G.J. Surge voltages in residential and industrial power circuits. IEEE Transactions on Power Apparatus, v. PAS-89, n.6, p.1049-1055, jul./ago. 1970.

McLOUGHLIN, R. & REDDING, R.J. Protection of eletronics from lightning and other large AC power pulses. EMC Technology, p.33-38, set./out. 1987.

NASH JR., H.O. & WELLS, F.M. Power systems disturbances and considerations for power conditioning. IEEE Transactions on Industry Applications, v. IA-21, n.6, p.1472-1481, nov./dez. 1985.

NATIONAL, Data Conversion/Acquisition Databook. Santa Clara, CA, National Semicondutor Co., 1984.

NATIONAL, General Purpose Linear Devices Data Book. Santa Clara, CA, National Semicondutor Co., 1989.

PAWLOSKI, M.B.; MOROYAN, T.; ALNETHER, J. Inside CMOS technology. Byte Publications Inc., p.94-107, set. 1983.

RCA, CMOS Integrated Circuits. RCA Solid State, 1983.

SCHILDT, H. Turbo C/C^{++} the Complete Reference. Berkeley, CA, Mc Graw Hill, 1990.

STOUT, D. Handbook of operational amplifier circuit design. New York, Mc Graw Hill, 1976. Capitulo 10 e 11.

TEXAS, High-speed, CMOS Logic Data Book. Dallas, TE, Texas Instruments Inc., 1988a.

TEXAS, TTL Logic Data Book. Dallas, TE, Texas Instruments Inc., 1988b.

TRINDADE, O. Uma contribuição ao estudo do efeito dos distúrbios da rede elétrica sobre a operação confiável de microprocessadores. São Carlos, IFQSC, USP, 1987.(Dissertação).

APÊNDICE A

FONTES DOS PROGRAMAS

```
/*****************************************************
*              MONITOR.C                             *
*            PROGRAMA PRINCIPAL                      *
*****************************************************/

#include <dos.h>
#include "define.h"

char   tecla_flag,   adc_flag,   mascara_reset,   mascara_int,
mascara_isr;
char    flag_dist_hora,    flag_dist_data,    flag_dist_ano,
flag_inicio_volt, flag_inicio_faixa2;
char digito [4], cisr, faixa;
unsigned jisr, kisr;
unsigned minuto, hora, ano, data;
unsigned tecla;
unsigned    *ram_aquis,    *ram_ant_hora,    *ram_ant_ano,
*ram_ant_data;
unsigned NP_ant [4], *sil [3];
unsigned ano_ant [4], mes_ant [4], dia_ant [4];
unsigned hora_ant [4], minuto_ant [4];
unsigned ano_ant_volt, mes_ant_volt, dia_ant_volt;
unsigned hora_ant_volt, minuto_ant_volt;
unsigned volt_var, nivel;
unsigned volt_var_max, volt_var_min;

void      output (unsigned *, char);
void      vetor (int ,void interrupt (*)());

void      interrupt atual_data (void);
void      interrupt atual_hora (void);
void      interrupt atual_min (void);
void      interrupt pulso (void);
void      interrupt ler_irf (void);
void      interrupt nivel_bat (void);
void      interrupt ad_conv (void);
void      interrupt teclado (void);

void           display (char *);
char           input (unsigned *);
void           timer_rel (void);
void           pisca (char);
void      novo_horario (void);
void           nova_data (void);
void      mostra_data (void);
char           ler_ad (int, int);
void      mostra_bateria (char);
void           acerto (char, char);
void      mostra_memoria (void);
void      diretorio (unsigned);
void      atraso (int);
void      grava_3RAM (unsigned *, unsigned);
void      div_freq (unsigned *, unsigned *, char, unsigned);
unsigned  potxy (unsigned, unsigned);

void main ()
{
     char  nivel_bateria,  flag_rel  =  0,  flag_dir1  =  0,
mascara_aux, flag_pausa = 0;
     int i;
     unsigned nivel, nivel_max, segundo, seg_aux, seg_min;
```

```
/*      INICIALIZACAO DAS VARIAVEIS GLOBAIS  */

     tecla = 0;
     tecla_flag = 0;
     adc_flag = 0;
     minuto = 0;
     hora = 0;
     ano = 0;
     flag_dist_hora = 0;
     flag_dist_data = 0;
     flag_dist_ano = 0;
     flag_inicio_volt = 0;
     volt_var = 0;
     ano_ant_volt = 0;
     mes_ant_volt = 0;
     dia_ant_volt = 0;
     hora_ant_volt = 0;
     minuto_ant_volt = 0;
     for (i = 0; i <= 3; i++)
     {    NP_ant [i] = 0;
          ano_ant [i] = 0;
          mes_ant [i] = 0;
          dia_ant [i] = 0;
          hora_ant [i] = 0;
          minuto_ant [i] = 0;

        }
     sil [0] = (unsigned *) SIL1;
     sil [1] = (unsigned *) SIL2;
     sil [2] = (unsigned *) SIL3;
     sil [3] = (unsigned *) SIL4;
     ram_aquis = (unsigned *) 0x10;
     mascara_reset = 0xFF;

/*   HABILITACAO DE INTERRUPCOES DO 80C86          */

     enable ();

/*   TESTE DO DISPLAY                    */

     pisca (0x38);

/*   PROGRAMACAO DO TIMER DO RELOGIO DO SISTEMA    */

     timer_rel ();

/*   PROGRAMACAO DO TIMER DO TECLADO               */

     div_freq   ((unsigned   *)   CTIMERTEC,   (unsigned   *)
DADOTIMERTEC, MODOTIMERTEC, (unsigned) DIVTIMERTEC);

/*   INICIALIZACAO DOS VETORES DE INTERRUPCAO      */

     vetor (INTR0, atual_data);
     vetor (INTR1, atual_hora);
     vetor (INTR2, atual_min);
     vetor (INTR3, pulso);
     vetor (INTR4, ler_irf);
     vetor (INTR5, nivel_bat);
     vetor (INTR6, ad_conv);
     vetor (INTR7, teclado);
```

```
/*   INICIALIZACAO DA CONTROLADORA DE INTERRUPCOES     */

     mascara_int = MASC7;

     output    ((unsigned *) EICW1, ICW1);
     output    ((unsigned *) EICW2, ICW2);
     output    ((unsigned *) EICW4, ICW4);
     output    ((unsigned *) EOCW1, mascara_int);

/*   PRESET DOS FLIP-FLOP'S DE INTERFACE COM A 82C59   */

     output  ((unsigned *) INTRES,0);
     output    ((unsigned *) INTRES, 0xFF);

/*   PRESET DO FLIP-FLOP DO TECLADO                    */

     output  ((unsigned *) FFPTEC, 0);

/*   PRESET DOS FLIP-FLOP'S DO NIVEL DE BATERIA   */

     output ((unsigned *) BUFBAT, 0);


/*                    LOOP ETERNO            */

     for (;;)
     {    while (tecla_flag == 0)
               output ((unsigned *) FFPTEC, 0);

          output ((unsigned *) EOCW1, 0xFF);
          output ((unsigned *) TECBLOQ, 0);
          output ((unsigned *) INTRES, INTRES7);
          output ((unsigned *) FFPTEC, 0);
          tecla_flag = 0;

          switch (tecla)
          {
               /*  ACERTO DE RELOGIO    */
               case   F1:
                    if (flag_rel == 0)
                    {    flag_rel = 1;
                         mascara_int = mascara_int & (MASC0 &
MASC1 & MASC2 & MASC7);
                    }
                    novo_horario ();
                    break;

               /*      DISPONIVEL  */
               case SHIFTF1:
                    break;

                         /*        PAUSA      */
               case   F2:
                    if (flag_dir1 == 1)
                    {    flag_pausa = 1;
                         mascara_aux = mascara_int;
                         mascara_int = MASC7;
                         for (i = 0; i <= 3; i++)
                              digito [i] = (0xE);
                         display (digito);
                    }
```

```
                    break;

                        /*     FIM DE PAUSA  */
                case SHIFTF2:
                    if (flag_dir1 == 1)
                    {    for (i = 0; i <= 3; i++)
                              output (sil [i], 0);
                         mascara_int = mascara_aux;
                         flag_pausa = 0;
                         mascara_reset = 0xBF;
                         output ((unsigned *) ZDP, 0);
                         div_freq  ((unsigned  *)  TRANSCONTR,
(unsigned *) TRANST0, MODOTRANSFREQ, (unsigned) DIVTRANS);
                         div_freq  ((unsigned  *)  TRANSCONTR,
(unsigned *) TRANST1, MODOTRANSDUR, (unsigned) DIVTRANS);
                         output ((unsigned *) CDP, 0);
                         output ((unsigned *) INTRES, 0);
                         output ((unsigned *) INTRES, 0x7F);
                         output ((unsigned *) DFEN, 0);
                    }
                    break;

                        /*    ACERTO DE DATA */
                case   F3:
                    if (flag_rel == 0)
                         pisca (DATAPROMPT);
                    else
                    {
                         nova_data ();
                    }
                    break;

                        /*   LEITURA DA DATA */
                case SHIFTF3:
                         mostra_data ();
                    break;

                /* INICIO DE AQUISICAO   */
                case   F4:
                    if (flag_dir1 == 0)
                    {    diretorio (0);
                         pisca (0xB);
                         mascara_reset = 0xBF;
                         ram_ant_ano = ram_aquis;
                         grava_3RAM (ram_aquis++, ano);
                         ram_ant_data = ram_aquis;
                         grava_3RAM (ram_aquis++, data);
                         ram_ant_hora = ram_aquis;
                         grava_3RAM (ram_aquis++, hora);
                                        flag_dir1 = 1;
                         mascara_int = mascara_int & MASC3 &
MASC4 & MASC5;

                         for (i = 0; i <= 3; i++)
                              output (sil [i], 0);
                         output ((unsigned *) ZDP, 0);
                         div_freq  ((unsigned  *)  TRANSCONTR,
(unsigned *) TRANST0, MODOTRANSFREQ, (unsigned) DIVTRANS);
                         div_freq  ((unsigned  *)  TRANSCONTR,
(unsigned *) TRANST1, MODOTRANSDUR, (unsigned) DIVTRANS);
                         output ((unsigned *) CDP, 0);
```

```
                    output ((unsigned *) INTRES, 0);
                    output ((unsigned *) INTRES, 0x7F);
                    output ((unsigned *) DFEN, 0);
                    output ((unsigned *) EOCW2, EOINE);
               }
               break;

                    /*   FIM DE AQUISICAO     */
          case SHIFTF4:
               if (flag_dir1 == 1)
               {    diretorio (1);
                    disable ();
                    for (i = 0; i <= 3; i++)
                         digito [i] = (0x5);
                    display (digito);
                    for (;;);
               }
               break;

          /* LEIT. BAT. POSITIVA   */
          case   F5:
               nivel_bateria = ler_ad (0, 128);
               mostra_bateria (nivel_bateria);
               break;

                    /* LEIT. BAT. NEGATIVA    */
          case SHIFTF5:
               nivel_bateria = ler_ad (1, 128);
               mostra_bateria (nivel_bateria);
               break;

          /*  LEITURA MEM. DISP.   */
          case   F6:
               mostra_memoria ();
               break;

          /*   TESTE DO DISPLAY    */
          case SHIFTF6:
               pisca (0x38);
     }
     tecla = 0;
     output ((unsigned *) EOCW1, mascara_int);
     output ((unsigned *) INTRES, 0xFF);
     output (sil [0], 0);
     output (sil [1], 0);
     output (sil [2], 0);
     output (sil [3], 0);
     if ((flag_rel == 1) && (flag_pausa == 0))
               disp_rel (0);
}
}
```

```
/*****************************************************
*                IOMON.C                            *
*            ROTINAS DESTE ARQUIVO                  *
* TITULO  : OUTPUT                                  *
* FUNCAO  : escrever um byte em um porto            *
* TITULO  : INPUT                                   *
* FUNCAO  : ler um byte de um porto                 *
* TITULO  : INPUT_INT                               *
* FUNCAO  : ler um inteiro de um porto              *
*****************************************************/

/*****************************************************
* TITULO  :output                                   *
* FUNCAO  :escrever um byte em um porto             *
* ENTRADAS:endereco do porto, byte                  *
* SAIDAS  :nao ha                                   *
* TESTADA : SIM [X] NAO [ ]                         *
*****************************************************/

void output (unsigned *end_porto, char dado)
{
     asm  {
          push dx
          mov  al,dado
          mov  dx,end_porto
          out  dx,al
          pop  dx};
}
/*****************************************************
* TITULO  : input                                   *
* FUNCAO  : ler um byte de um porto                 *
* ENTRADAS: endereco do porto (unsigned *)          *
* SAIDAS  : byte contido no porto (char)            *
* TESTADA : SIM [X] NAO [ ]                         *
*****************************************************/

char input (unsigned *end_porto)
{
     char      dado;
     asm  {
          push dx
          mov  dx,end_porto
          in   al,dx
          mov  dado,al
          pop  dx};
        return dado;
}
/*****************************************************
* TITULO  : input_int                               *
* FUNCAO  : ler um inteiro de um porto              *
* ENTRADAS: endereco do porto (unsigned *)          *
* SAIDAS  : inteiro contido no porto                *
* TESTADA : SIM [X] NAO [ ]                         *
*****************************************************/

unsigned input_int (unsigned *end_porto)
{
     int  dado;
     asm  {
          push dx
          mov  dx,end_porto
```

```
        in    ax,dx
        mov   dado,ax
        pop   dx};
      return dado;
}
```

```
/**************************************************
*                CONVERT.C                        *
*             ROTINAS DESTE ARQUIVO               *
* TITULO   : HEXBCD                               *
* FUNCAO   : converte um HEXADECIMAL em BCD       *
* TITULO   : INTBCD                               *
* FUNCAO   : converter um inteiro nao sinalizado em *
*         um bcd de quatro digitos                *
**************************************************/

/**************************************************
* TITULO   : hexbcd                               *
* FUNCAO   : converte um HEXADECIMAL em BCD       *
* ENTRADAS: hexadecimal a ser convertido (int)    *
* SAIDAS   : BCD (char) de dois digitos           *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

char hexbcd (int hexa)
{
     int dezena = 0;

     if (hexa < 10)
          return (char) hexa;
     do
     {    hexa -= 10;
          dezena++;
     }
     while (hexa >= 10);
     return (char) ((dezena << 4) + hexa);
}
/**************************************************
* TITULO   : intbcd                               *
* FUNCAO   : converter um inteiro nao sinalizado em *
*         um bcd de quatro digitos                *
* ENTRADAS: inteiro a ser convertido, endereco da *
*         cadeia de saida                         *
* SAIDAS   :                                      *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void intbcd (unsigned entrada, char *saida)
{
     unsigned auxiliar;
     int i;

     auxiliar = entrada;

     for (i = 3; i >= 0; i--)
     {    auxiliar = entrada / pot10 (i);
          saida [i] = (char) auxiliar;
          entrada -= auxiliar * pot10 (i);
     }
}
```

```
/**************************************************
*                RWMEM.C                          *
*            ROTINAS DESTE ARQUIVO                *
* TITULO  : VETOR                                 *
* FUNCAO  : inicializar um vetor de interrupcao   *
* TITULO  : GRAVA_3RAM                            *
* FUNCAO  : gravar dados na RAM auxiliar          *
* TITULO  : MOSTRA_MEMORIA                        *
* TITULO  : DIRETORIO                             *
* FUNCAO  : inicializar o diretorio da aquisicao  *
* FUNCAO  : mostrar memoria de aquisicao disponivel*
**************************************************/

#include "define.h"

extern    unsigned *ram_aquis, data, ano, hora, minuto;

/**************************************************
* TITULO  : vetor                                 *
* FUNCAO  : inicializar um vetor de interrupcao   *
* ENTRADAS: nome da rotina de interrupcao         *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void vetor (int intr, void interrupt (*isr)())
{
     asm {
          push      ds
          push cx
          push si
          mov  ax,  0
          mov  ds,  ax
          mov  ax,  intr
          mov  cl,  2
          sal  ax,  cl
          mov  si,  ax
          mov  ax,  isr
          mov  [si],   ax
          inc  si
          inc  si
               mov ax,  isr.2
          mov  [si],     ax
          pop  si
          pop  cx
          pop  ds};
}
/**************************************************
* TITULO  : grava_3RAM                            *
* FUNCAO  : gravar dados na RAM auxiliar          *
* ENTRADAS: endereco (unsigned), dado (int)       *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void grava_3RAM (unsigned *end, unsigned dado)
{
     asm {
          push      ds
          push si
          mov  ax,  SEG_DADOS
```

```
          mov  ds,  ax
          mov  si,  end
          mov  ax,  dado
          mov  [si],   ax
          pop  si
          pop  ds};
}
/**************************************************
* TITULO  : mostra_memoria                        *
* FUNCAO  : mostrar memoria de aquisicao disponivel*
*        em blocos de 1024 bytes                  *
* ENTRADAS:                                       *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void mostra_memoria (void)
{
     unsigned k_bytes;
        char disponivel [4];

     k_bytes =  ((unsigned *) MAXMEM - ram_aquis) / 1024;

     intbcd (k_bytes, disponivel);

     display (disponivel);

     atraso ((int) MULT);
}
/**************************************************
* TITULO  : diretorio                             *
* FUNCAO  : inicializar o diretorio da aquisicao  *
* ENTRADAS: indicador de inicio (0) ou fim (1)    *
*        de aquisicao (unsigned)                  *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void diretorio (unsigned momento)
{
     unsigned data_dir, ano_hora_dir, seg_min_dir, nivel_batn,
nivel_batp, nivel_bat_dir, seg_aux;

     output ((unsigned *) 0xC2, COUNTERLATCH0);

     seg_aux = input ((unsigned *) DADOTIMER0REL);
     seg_aux = (unsigned) (60 - seg_aux);
     seg_aux = seg_aux & 0xFF;
     seg_aux = seg_aux << 8;

     data_dir = data & 0xFFF;
     ano_hora_dir = (ano & 0x7F) | (hora << 8);
     seg_min_dir = seg_aux | (minuto & 0x3F);
     nivel_batn = ((unsigned) ler_ad (1,128) & 0xFF) << 8;
     nivel_batp = (unsigned) ler_ad (0,128) & 0xFF;
     nivel_bat_dir = nivel_batn | nivel_batp;

     grava_3RAM  (((unsigned  *)  (DATA  +  6  *  momento)),
data_dir);
     grava_3RAM  (((unsigned  *)  (ANOHORA  +  6  *  momento)),
ano_hora_dir);
```

```
    grava_3RAM  (((unsigned  *)  (SEGMIN  +  6  *  momento)),
seg_min_dir);

    if (momento == 1)
    {    grava_3RAM ((unsigned *) BAT, nivel_bat_dir);
         grava_3RAM   ((unsigned   *)   FINAL,   (unsigned)
--ram_aquis);
         ram_aquis++;
    }
}
```

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

```
/**************************************************
*                      AD.C                       *
*            ROTINAS DESTE ARQUIVO                *
* TITULO  : NIVEL_BAT                             *
* FUNCAO  : registrar valor de tensao de bateria  *
*        em nivel baixo                           *
* TITULO  : LER_AD                                *
* FUNCAO  : ler uma das entradas do MUX atraves   *
*        do conversor AD                          *
* TITULO  : MOSTRA_BATERIA                        *
* FUNCAO  : mostrar no display nivel de bateria   *
* TITULO  : AD_CONV                               *
* FUNCAO  : sinalizar final de conversao          *
* TITULO  : PULSO                                 *
* FUNCAO  : reconhecer uma interrupcao causada    *
*        por um transiente                        *
* TITULO  : TRANS                                 *
* FUNCAO  : registrar duracao, frequencia e  pico *
*        de um  transiente                        *
**************************************************/

#include <dos.h>
#include "define.h"

extern  unsigned tecla, minuto, *ram_aquis;
extern  char adc_flag, tecla_flag, mascara_reset, mascara_int;

void      pisca (char);
void      divflut (unsigned, unsigned, char *);
void atraso (int);
char input (unsigned *);
char      ler_ad (int, int);
void      grava_3RAM (unsigned *, unsigned);

/**************************************************
* TITULO  : trans                                 *
* FUNCAO  : registrar duracao, frequencia e pico  *
*        de um  transiente                        *
* ENTRADAS:                                       *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void trans (void)
{
     unsigned td, freq, Vp, seg_min;

        output ((unsigned *) 0xC2, COUNTERLATCH0);
     seg_min  =  ((60  -  (unsigned)  (input  ((unsigned  *)
DADOTIMER0REL))) << 6) | minuto;

     output ((unsigned *) TRANSCONTR, TRANSRDBAK);

     td = (input_int ((unsigned *) TRANST1) & 0xFF);
     td = td | (input_int ((unsigned *) TRANST1) << 8);
     td = (((unsigned) DIVTRANS - td) & 0xFFF) | 0x4000;

     freq = (input_int ((unsigned *) TRANST0) & 0xFF);
     freq = freq | (input_int ((unsigned *) TRANST0) << 8);
     freq = (((unsigned) DIVTRANS - freq) & 0xFFF) | 0x5000;
```

```
     Vp = (ler_ad (2,1) & 0xFF) | 0x6000;

     grava_3RAM (ram_aquis++, td);
     grava_3RAM (ram_aquis++, freq);
     grava_3RAM (ram_aquis++, Vp);
     grava_3RAM (ram_aquis++, seg_min);

     div_freq ((unsigned *) TRANSCONTR, (unsigned *) TRANST0,
MODOTRANSFREQ, (unsigned) DIVTRANS);
     div_freq ((unsigned *) TRANSCONTR, (unsigned *) TRANST1,
MODOTRANSDUR, (unsigned) DIVTRANS);

     if ((unsigned) ram_aquis > ((unsigned) MAXMEM - (unsigned)
10))
     {    tecla = SHIFTF4;
          tecla_flag = 1;
     }
}
/*****************************************************
* TITULO  : pulso                                    *
* FUNCAO  : reconhecer uma interrupcao causada       *
*         por um transiente                          *
* ENTRADAS: pedido de interrupcao do circuito        *
*         detetor de transientes                     *
* SAIDAS  :                                          *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

void interrupt pulso (void)
{
     asm {
          mov ax,DS
          mov ds,ax};
     output ((unsigned *) INTRES, 0);
     output ((unsigned *) EOCW2, EOINE);
     output ((unsigned *) DFEN, 0);

     trans ();

     enable ();
     output ((unsigned *) INTRES, 0);
     output ((unsigned *) INTRES, 0xFF);

     div_freq ((unsigned *) TRANSCONTR, (unsigned *) TRANST0,
MODOTRANSFREQ, (unsigned) DIVTRANS);
     div_freq ((unsigned *) TRANSCONTR, (unsigned *) TRANST1,
MODOTRANSDUR, (unsigned) DIVTRANS);
     output ((unsigned *) ZDP, 0);
     output ((unsigned *) CDP, 0);
     output ((unsigned *) DFEN, 0);
}
```

```
/**************************************************
* TITULO  : nivel_bat                             *
* FUNCAO  : registrar valor de tensao de bateria  *
*        em nivel baixo                           *
* ENTRADAS: pedido de interrupcao de bateria      *
*        em nivel baixo                           *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void interrupt nivel_bat (void)
{
     asm{
          mov  ax,DS
          mov  ds,ax     };

     tecla_flag =1;
     tecla = (unsigned) SHIFTF4;

     enable ();
     output ((unsigned *) INTRES, INTRES5);
     output ((unsigned *) INTRES, 0xFF);
     output ((unsigned *) EOCW2, EOINE);
}
/**************************************************
* TITULO  : LER_AD                                *
* FUNCAO  : ler uma das entradas do MUX atraves   *
*        do conversor AD                          *
* ENTRADAS: tipo de entrada (0 - bat. positiva,   *
*        1 - bat. negativa, 2 - Tensao de pico,   *
*        3 - tensao instantanea da rede) e nu-    *
*        mero de leituras por ponto               *
* SAIDAS  : valor convertido sem ajuste de faixa  *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

char ler_ad (int tipo_entrada, int cont_media)
{
     char nivel_bin = 0, mascara_aux;
     unsigned auxiliar;
     int i;
     enable ();
     mascara_aux = mascara_int;
     mascara_int = MASC6;
     output ((unsigned *) EOCW1, mascara_int);
     output ((unsigned *) INTRES, 0);
     output ((unsigned *) INTRES, ~INTRES6);
     switch (tipo_entrada)
     {    case 0:
               auxiliar = (unsigned) NBATP;
               break;
          case 1:
               auxiliar = (unsigned) NBATN;
               break;
          case 2:
               auxiliar = (unsigned) VP;
               break;
          case 3:
               auxiliar = (unsigned) AMPL60;
      }
```

```
     output ((unsigned *) auxiliar, 0);

     auxiliar = 0;

     for (i = 0; i < cont_media; i++)
     {    output ((unsigned *) ADC, 0);
          while (adc_flag == 0);
          adc_flag = 0;
          nivel_bin = input ((unsigned *) ADC);
          auxiliar += ((unsigned) nivel_bin) & 0xFF;
     }
     nivel_bin = (char) (auxiliar / cont_media);

     mascara_int = mascara_aux;
     output ((unsigned *) FFPTEC, 0);
     tecla_flag = 0;
     output ((unsigned *) EOCW1, mascara_int);
     output ((unsigned *) INTRES, INTRES6);
     output ((unsigned *) INTRES, mascara_reset);

     return nivel_bin;
}
/*****************************************************
* TITULO   : mostra_bateria                          *
* FUNCAO   : mostrar no display nivel de bateria     *
* ENTRADAS: valor da tensao em binario               *
* SAIDAS   :                                         *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

void mostra_bateria (char binario)
{
     char resultado [4];
     unsigned bin_aux;

     bin_aux = (((unsigned) binario) & 0xFF) * 15;

     divflut (bin_aux, 255, resultado);

     resultado [3] = resultado [3] | 0x10;

     display (resultado);

     atraso ((int) MULTBAT);
}
/*****************************************************
* TITULO   : ad_conv                                 *
* FUNCAO   : sinalizar final de conversao            *
* ENTRADAS: pedido de interrupcao do conversor AD    *
* SAIDAS   :                                         *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

void interrupt ad_conv (void)
{
     asm{
          mov  ax,DS
          mov  ds,ax};

     adc_flag = 1;
```

```
        enable ();
        output ((unsigned *) INTRES, 0);
        output ((unsigned *) INTRES, ~INTRES6);
        output ((unsigned *) EOCW2, EOINE);
}
```

```
/**************************************************
*              IRF.C                              *
*            ROTINAS DESTE ARQUIVO                *
* TITULO  : LER_IRF                               *
* FUNCAO  : identificar e ler irf                 *
* TITULO  : ESCR_IRF                              *
* FUNCAO  : ler e registrar historico da irfn     *
* TITULO  : ESCR_60                               *
* FUNCAO  : monitorar frequencia nominal da rede, *
*        registrando quando houver desvios        *
* TITULO  : INTERVALO                             *
* FUNCAO  : calcular o intervalo entre o horario  *
*        atual e o horario anterior               *
**************************************************/

#include <dos.h>
#include "define.h"

extern char flag_dist_hora, flag_dist_data, flag_dist_ano;
extern char tecla_flag, flag_inicio_volt, flag_inicio_faixa2;
extern unsigned tecla, *ram_aquis, minuto, ano, hora, data;
extern unsigned NP_ant [4], *sil [3];
extern unsigned ano_ant [4], mes_ant [4], dia_ant [4];
extern unsigned hora_ant [4], minuto_ant [4];
extern char faixa, mascara_int, mascara_reset;
extern unsigned ano_ant_volt, mes_ant_volt, dia_ant_volt;
extern unsigned hora_ant_volt, minuto_ant_volt;
extern unsigned volt_var;

void      escr_irf (char);
void      escr_60 (void);
void      grava_3RAM (unsigned *, int);
void      pisca (char);

/**************************************************
* TITULO  : ler_irf                               *
* FUNCAO  : identificar e ler irf                 *
* ENTRADAS: pedido de interrupcao de irf          *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void      interrupt ler_irf (void)
{
     int i;

     asm  {
          mov  ax,DS
          mov  ds,ax };

     output ((unsigned *) INTRES, INTRES4);
     enable ();
     output ((unsigned *) EOCW2, EOINE);
          flag_dist_hora = 1;
          flag_dist_data = 1;
          flag_dist_ano  = 1;

          faixa = (input ((unsigned *) LFF)) & 0xF;

          if ((faixa & 1) == 1)
          {    if (flag_inicio_volt == 0)
```

```
                {    flag_inicio_volt = 1;
                     output ((unsigned *) Z60, 0);
                }
                else
                {    escr_60 ();
                     flag_inicio_volt = 0;
                }
                output (sil [0], 0);
           }
           if ((faixa & 2) == 2)
           {    if (flag_inicio_faixa2 == 0)
                {    flag_inicio_faixa2 = 1;
                }
                else
                {    escr_irf (2);
                     flag_inicio_faixa2 = 0;
                }
                output (sil [1], 0);
           }
           if ((faixa & 4) == 4)
           {    escr_irf (3);
                div_freq ((unsigned *) TRANSCONTR, (unsigned *)
TRANST0, MODOTRANSFREQ, (unsigned) DIVTRANS);
                div_freq ((unsigned *) TRANSCONTR, (unsigned *)
TRANST1, MODOTRANSDUR, (unsigned) DIVTRANS);
                output ((unsigned *) ZDP, 0);
                output ((unsigned *) CDP, 0);
                output (sil [2], 0);
           }
           if ((faixa & 8) == 8)
           {    escr_irf (4);
                div_freq ((unsigned *) TRANSCONTR, (unsigned *)
TRANST0, MODOTRANSFREQ, (unsigned) DIVTRANS);
                div_freq ((unsigned *) TRANSCONTR, (unsigned *)
TRANST1, MODOTRANSDUR, (unsigned) DIVTRANS);
                output ((unsigned *) ZDP, 0);
                output ((unsigned *) CDP, 0);
                output (sil [3], 0);
           }
           if  ((unsigned)  ram_aquis  >  ((unsigned)  MAXMEM  -
(unsigned) 14))
           {    tecla = SHIFTF4;
                tecla_flag = 1;
           }
      enable ();
      output ((unsigned *) INTRES, 0xFF);
}
/*****************************************************
* TITULO   : escr_irf                                *
* FUNCAO   : ler e registrar historico da irfn       *
* ENTRADAS: faixa n da irf                           *
* SAIDAS   :                                         *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

void escr_irf (char faixa)
{
     unsigned segundo, seg_aux, NP, seg_min, *end_registrador,
limpa_faixa;
     int dif_tempo;
     char ajusta_faixa;
```

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

```
     ajusta_faixa = faixa - 1;

     switch (faixa)
     {    case 2:
               limpa_faixa = 0xFFF;
               break;
          case 3:
               limpa_faixa = 0x3FF;
               break;
          case 4:
               limpa_faixa = 0xFFF;
     }
     output ((unsigned *) 0xC2, COUNTERLATCH0);
     seg_aux  =  60  -  (unsigned)  (input  ((unsigned  *)
DADOTIMER0REL));

     segundo = seg_aux << 6;
     seg_min = segundo | minuto;
     end_registrador  =  (unsigned  *)  ((unsigned)  LFF  +
(unsigned) faixa * 0x80);
     NP  =  ((input_int  (end_registrador))  &  limpa_faixa)  |
((ajusta_faixa) * 0x1000);

     dif_tempo = intervalo (ano_ant [ajusta_faixa], mes_ant
[ajusta_faixa],    dia_ant    [ajusta_faixa],    hora_ant
[ajusta_faixa], minuto_ant [ajusta_faixa]);
     if ((NP < NP_ant [ajusta_faixa] - 20) || (NP > NP_ant
[ajusta_faixa] + 20) || dif_tempo > 2)
     {    grava_3RAM (ram_aquis++, NP);
          grava_3RAM (ram_aquis++, seg_min);
          ano_ant [ajusta_faixa] = ano & 0x7F;
          mes_ant [ajusta_faixa] = (data >> 8) & 0xF;
          dia_ant [ajusta_faixa] = data & 0x1F;
          hora_ant [ajusta_faixa] = hora & 0x1F;
          minuto_ant [ajusta_faixa] = seg_min & 0x3F;
          NP_ant [ajusta_faixa] = NP;
     }

}
/*****************************************************
* TITULO  : escr_60                                  *
* FUNCAO  : monitorar frequencia nominal da rede,    *
*         registrando quando houver desvios          *
* ENTRADAS: faixa n da irf                           *
* SAIDAS  :                                          *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

void escr_60 (void)
{
     unsigned segundo, seg_aux, NP, seg_min,  nivel;
     int dif_tempo;
     nivel = (int) (ler_ad (3, 1) & 0xFF);

     output ((unsigned *) 0xC2, COUNTERLATCH0);
     seg_aux  =  60  -  (unsigned)  (input  ((unsigned  *)
DADOTIMER0REL));

     segundo = seg_aux << 6;
     seg_min = segundo | minuto;
```

```
     dif_tempo  =  intervalo  (ano_ant_volt,  mes_ant_volt,
dia_ant_volt, hora_ant_volt, minuto_ant_volt);
     if ((nivel >= (unsigned) (SOBRETENSAO)) || (nivel <=
(unsigned) SUBTENSAO))
          if (((nivel > (volt_var + 2)) || (nivel < (volt_var
- 2))) || dif_tempo > 2)
               {    volt_var = nivel;
                    nivel = nivel | 0x8000;
                    grava_3RAM (ram_aquis++, nivel);
                    grava_3RAM (ram_aquis++, seg_min);
                    ano_ant_volt = ano & 0x7F;
                    mes_ant_volt = (data >> 8) & 0xF;
                    dia_ant_volt = data & 0x1F;
                    hora_ant_volt = hora & 0x1F;
                    minuto_ant_volt = seg_min & 0x3F;
               }
     output ((unsigned *) Z60, 0);
     NP = ((input_int ((unsigned *) LF1)) & 0xFFF);
     dif_tempo = intervalo (ano_ant [0], mes_ant [0], dia_ant
[0], hora_ant [0], minuto_ant [0]);
     if ((NP <= 0x821 || NP >= 0x825) && ((NP <= NP_ant [0] -
20) || (NP >= NP_ant [0] + 20) || dif_tempo > 2))
     {    grava_3RAM (ram_aquis++, NP);
          grava_3RAM (ram_aquis++, seg_min);
          ano_ant [0] = ano & 0x7F;
          mes_ant [0] = (data >> 8) & 0xF;
          dia_ant [0] = data & 0x1F;
          hora_ant [0] = hora & 0x1F;
          minuto_ant [0] = seg_min & 0x3F;
          NP_ant [0] = NP;
     }
}
/*****************************************************
* TITULO  : intervalo                                *
* FUNCAO  : calcular o intervalo entre o horario     *
*         atual e o horario anterior                 *
* ENTRADAS: horario anterior, constando de ano,      *
*         mes, dia, hora, minuto e segundo           *
* SAIDAS  : diferenca em minutos                     *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

int intervalo (unsigned ano_ant, unsigned mes_ant, unsigned
dia_ant, unsigned hora_ant, unsigned min_ant)
{
     unsigned ano_aux, mes_aux, dia_aux, hora_aux, min_aux;
     int bissexto, dif_min;
        char ultimo_dia, dif_dia;
     char tabdia [2][13] =
     {    {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
          {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
     };

     ano_aux = ano & 0x7F;
     mes_aux = (data >> 8) & 0xF;
     dia_aux = data & 0x1F;
     hora_aux = hora & 0x1F;
     min_aux = minuto & 0x3F;

     if (ano_ant == ano_aux)
          if ((mes_aux - mes_ant) > 1)
```

```
                return (int) MAXMIN;
            else
                if ((mes_aux - mes_ant) == 1)
                {    bissexto = ano_aux % 4 == 0 && ano_ant %
100 != 0 || ano_ant % 400 == 0;
                     ultimo_dia = tabdia [bissexto][mes_ant];
                     dif_dia = ultimo_dia - dia_ant + dia_aux;
                }
                else
                     dif_dia = dia_aux - dia_ant;

        else
            if (mes_ant != 12 || mes_aux != 1)
                return (int) MAXMIN;
            else
                dif_dia = 31 - dia_ant + dia_aux;

        if (dif_dia >= 22)
            return (int) MAXMIN;
        else
            if (dif_dia == 0)
                if (hora_aux == hora_ant)
                     dif_min = min_aux - min_ant;
                else
                     dif_min = (hora_aux - hora_ant - 1) * 60 +
(60 - min_ant) + min_aux;
            else
                dif_min = ((dif_dia - 1) * 24 + (24 - hora_ant
- 1) + hora_aux) * 60 + (60 - min_ant) + min_aux;
        return dif_min;
}
```

```
/**************************************************
*                TECDISP.C                        *
*             ROTINAS DESTE ARQUIVO               *
* TITULO  : DISPLAY                               *
* FUNCAO  : escrever nos quatro digitos do display *
* TITULO  : ATRASO                                *
* FUNCAO  : causar um atraso de compensacao para  *
*         perifericos                             *
* TITULO  : PISCA                                 *
* FUNCAO  : piscar todos os segmentos do display  *
* TITULO  : TECLADO                               *
* FUNCAO  : reconhecer tecla e chamar servico cor- *
*         respondente                             *
* TITULO  : DADODUPLO                             *
* FUNCAO  : ler duas dezenas via teclado          *
**************************************************/

#include  <dos.h>
#include  "define.h"

extern unsigned tecla;
extern char tecla_flag, mascara_int, mascara_reset;
extern char digito [4];

void            output (unsigned *, char);
void            display (char *);
void       atraso (int);
unsigned   input_int (unsigned *);
char       input (unsigned *);

/**************************************************
* TITULO  : display                               *
* FUNCAO  : escrever nos quatro digitos do display *
* ENTRADAS: string (char *) com os quatro digitos *
* SAIDAS  :                                       *
* TESTADA : SIM [X] NAO [ ]                       *
**************************************************/


void display (char *dado)
{
     int i,j;

     for (i = 0; i <= 3; i++)
     {    output ((unsigned *)(0x1054 + i * 0x2000), dado [i]);
          for (j = 0; j <= TEMPO; j++);}
}
/**************************************************
* TITULO  : atraso                                *
* FUNCAO  : causar um atraso de compensacao para  *
*         perifericos                             *
* ENTRADAS: multiplicar da base de atraso (int)   *
* SAIDAS  :                                       *
* TESTADA : SIM [X] NAO [ ]                       *
**************************************************/


void atraso (int mult)
{
     int i, j;
```

```
     for (i = 0; i <= mult; i++)
          for (j = 0; j <= TEMPO; j++);
}
/*****************************************************
* TITULO  : pisca                                    *
* FUNCAO  : piscar todos os segmentos do display     *
* ENTRADAS:                                          *
* SAIDAS  :                                          *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

void pisca (char caracter)
{
     int i;
     unsigned j;
     char branco [4], total [4];

     for (i = 0; i <= 3; i++)
     {    branco [i] = 0xF;
          total  [i] = caracter;
     }

     for (i = 0; i <= 2; i++)
     {    display (branco);
          for (j = 0; j <= DELAYDISP; j++);
          display (total);
          for (j = 0; j <= DELAYDISP; j++);
     }
}
/*****************************************************
* TITULO  : teclado                                  *
* FUNCAO  : reconhecer tecla e chamar servico cor-   *
*        respondente                                 *
* ENTRADAS: pedido de interrupcao do teclado         *
* SAIDAS  :                                          *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

void interrupt teclado (void)
{
     asm  {
          mov  ax,DS
          mov  ds,ax     };

     tecla = input_int ((unsigned *) TECLATCH);
     tecla_flag = 1;

     enable ();
     output ((unsigned *) FFPTEC, 0);
     output ((unsigned *) INTRES, INTRES7);
     output ((unsigned *) INTRES, 0xFF);

     output ((unsigned *) EOCW2, EOI7);
     output ((unsigned *) EOCW2, EOINE);
}
```

```
/**************************************************
* TITULO  : dadoduplo                             *
* FUNCAO  : ler duas dezenas via teclado          *
* ENTRADAS: prompt e string para devolucao do dado *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void dadoduplo (char prompt, char dezena [2])
{
          int i;
     char   digito   [4],   ult_dezena   [2],   mascara_aux,
mascara_reset_aux;

     mascara_reset_aux = mascara_reset;
     mascara_aux = mascara_int;

     tecla = 0;
     dezena [1] = 0;
     dezena [0] = 0;

     for (i = 0; i <= 3; i++)
          digito [i] = prompt;

     display (digito);

     i = 3;

     atraso (MULTEC);
     mascara_reset = ~INTRES7;
     tecla_flag = 0;
     mascara_int = MASC7;
     output ((unsigned *) EOCW1, mascara_int);
     output ((unsigned *) INTRES, INTRES7);
     output ((unsigned *) INTRES, mascara_reset);
     output ((unsigned *) FFPTEC, 0);

     do
     {    while (tecla_flag == 0);
          tecla_flag = 0;
          output ((unsigned *) EOCW1, 0xFF);
          output ((unsigned *) TECBLOQ, 0);
          switch (tecla)
          {    case TEC0:
                    digito [i--] = 0x30;
                    display (digito);
                    break;
               case TEC1:
                    digito [i] = 0x31;
                    if (i > 1)
                    {    ult_dezena [1] = (char) pot10 (i - 2);
                         dezena [1] += ult_dezena [1];
                              }
                    else
                    {    ult_dezena [0] = (char) pot10 (i);
                         dezena [0] += ult_dezena [0];
                    }
                    i--;
                    display (digito);
                    break;
               case TEC2:
```

SERVIÇO DE BIBLIOTECA E DOCUMENTAÇÃO
FÍSICA

```
                    digito [i] = 0x32;
                    if (i > 1)
                    {    ult_dezena [1] = 2 * (char) pot10 (i
- 2);
                         dezena [1] += ult_dezena [1];
                    }
                    else
                    {    ult_dezena [0] = 2 * (char) pot10 (i);
                         dezena [0] += ult_dezena [0];
                    }
                    i--;
                    display (digito);
                    break;
               case TEC3:
                    digito [i] = 0x33;
                    if (i > 1)

                    {    ult_dezena [1] = 3 * (char) pot10 (i
- 2);
                         dezena [1] += ult_dezena [1];
                    }
                    else
                    {    ult_dezena [0] = 3 * (char) pot10 (i);
                         dezena [0] += ult_dezena [0];
                    }
                    i--;
                    display (digito);
                    break;
               case TEC4:
                    digito [i] = 0x34;
                    if (i > 1)
                    {    ult_dezena [1] = 4 * (char) pot10 (i
- 2);
                         dezena [1] += ult_dezena [1];
                    }
                    else
                    {    ult_dezena [0] = 4 * (char) pot10 (i);
                         dezena [0] += ult_dezena [0];
                    }
                    i--;
                    display (digito);
                    break;
               case TEC5:
                    digito [i] = 0x35;
                    if (i > 1)
                    {    ult_dezena [1] = 5 * (char) pot10 (i
- 2);
                         dezena [1] += ult_dezena [1];
                    }
                    else
                    {    ult_dezena [0] = 5 * (char) pot10 (i);
                         dezena [0] += ult_dezena [0];
                    }
                    i--;
                    display (digito);
                    break;
               case TEC6:
                    digito [i] = 0x36;
                    if (i > 1)
                    {    ult_dezena [1] = 6 * (char) pot10 (i
- 2);
```

```
                    dezena [1] += ult_dezena [1];
                }
                else
                {   ult_dezena [0] = 6 * (char) pot10 (i);
                    dezena [0] += ult_dezena [0];
                }
                i--;
                display (digito);
                break;
            case TEC7:
                digito [i] = 0x37;
                if (i > 1)
                {   ult_dezena [1] = 7 * (char) pot10 (i
- 2);
                    dezena [1] += ult_dezena [1];
                }
                else
                {   ult_dezena [0] = 7 * (char) pot10 (i);
                    dezena [0] += ult_dezena [0];
                }
                i--;
                display (digito);
                break;
            case TEC8:
                digito [i] = 0x38;
                if (i > 1)
                {   ult_dezena [1] = 8 * (char) pot10 (i
- 2);
                    dezena [1] += ult_dezena [1];
                }
                else
                {   ult_dezena [0] = 8 * (char) pot10 (i);
                    dezena [0] += ult_dezena [0];
                }
                i--;
                display (digito);
                break;
            case TEC9:
                digito [i] = 0x39;
                if (i > 1)
                {   ult_dezena [1] = 9 * (char) pot10 (i
- 2);
                    dezena [1] += ult_dezena [1];
                }
                else
                {   ult_dezena [0] = 9 * (char) pot10 (i);
                    dezena [0] += ult_dezena [0];
                }
                i--;
                display (digito);
                break;
            case ANULA:
                tecla_flag = 0;
                output ((unsigned *) FFPTEC, 0);
                output ((unsigned *) INTRES, INTRES7);
                output    ((unsigned    *)    INTRES,
mascara_reset);
                mascara_int = mascara_aux;
                mascara_reset = mascara_reset_aux;
                output ((unsigned *) EOCW1, mascara_int);
                return;
```

```
                case SHIFTANULA:
                     digito [++i] = prompt;
                     if (i > 1)
                          dezena [1] -= ult_dezena [1];
                     else
                          dezena [0] -= ult_dezena [0];
                     display (digito);
           }
           atraso (MULTEC);
           output ((unsigned *) EOCW1, 0xFF);
           tecla_flag = 0;
           output ((unsigned *) FFPTEC, 0);
           output ((unsigned *) INTRES, INTRES7);
           output ((unsigned *) INTRES, mascara_reset);
           output ((unsigned *) EOCW1, mascara_int);
      }
      while (tecla != (unsigned) ENTRA);
      tecla_flag = 0;
      output ((unsigned *) FFPTEC, 0);
      output ((unsigned *) INTRES, INTRES7);
      output ((unsigned *) INTRES, mascara_reset);
      mascara_int = mascara_aux;
      mascara_reset = mascara_reset_aux;
      output ((unsigned *) EOCW1, mascara_int);
}
```

```
/**************************************************
*                RELOGIO.C                        *
*             ROTINAS DESTE ARQUIVO               *
* TITULO  : DIV_FREQ                              *
* FUNCAO  : inicializar um timer de um 8254       *
* TITULO  : TIMER_REL                             *
* FUNCAO  : inicializar o timer de base de tempo e *
*         o timer do relogio                      *
* TITULO  : LER_TIMER                             *
* FUNCAO  : ler latches de dados de um timer      *
* TITULO  : DISP_REL                              *
* FUNCAO  : mostrar horario no LCD                *
* TITULO  : ACERTO                                *
* FUNCAO  : acertar  horas e minutos do relogio   *
* TITULO  : ATUAL_MIN                             *
* FUNCAO  : atualizar minutos na RAM auxiliar e   *
*         horario no display                      *
* TITULO  : ATUAL_HORA                            *
* FUNCAO  : atualizar horas na RAM auxiliar       *
*         e na memoria de aquisicao               *
* TITULO  : NOVO_HORARIO                          *
* FUNCAO  : receber do teclado o horario para     *
*         atualizar o relogio                     *
**************************************************/

#include <dos.h>
#include "define.h"

extern unsigned mes_dia, ano, data;
extern char ei10, eoi1, eoi2, eoi3, eoi4, eoi5, eoi6, eoi7;
extern unsigned *ram_aquis, *ram_ant_hora;
extern char digito [4];
extern unsigned tecla, minuto, hora;
extern char flag_dist_hora, mascara_int;

void      display (char *);
void      div_freq (unsigned *, unsigned *, char, unsigned);
char      hexbcd (int);
void      grava_3RAM (unsigned *, int);
void atraso (int);
void      dadoduplo (char, char []);
char input (unsigned *);

/**************************************************
* TITULO  : div_freq                              *
* FUNCAO  : inicializar um timer de um 8254       *
* ENTRADAS: endereco de controle, endereco de dado,*
*         modo do timer, divisor                  *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void div_freq (unsigned *end_contr, unsigned *end_dado, char
modo, unsigned divisor)
{
     char lsdiv, msdiv;

     lsdiv = (char) (divisor & 0xFF);
     output (end_contr, modo);
     output (end_dado, lsdiv);
     if ((modo & RW_WORD) == RW_WORD)
```

```
     {    msdiv = (char) (divisor >> 8);
          output (end_dado, msdiv);
     }
}
/*****************************************************
* TITULO  : timer_rel                                *
* FUNCAO  : inicializar o timer de base de tempo e *
*         o timer do relogio                         *
* ENTRADAS:                                          *
* SAIDAS  :                                          *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/
void timer_rel (void)
{
     div_freq   ((unsigned   *)   CTIMER1BASE,   (unsigned   *)
DADOTIMER1BASE, MODOTIMER1BASE, (unsigned) BASE_RELOGIO);
     div_freq   ((unsigned   *)   CTIMEREL,   (unsigned   *)
DADOTIMER0REL, CTIMER0REL, (unsigned) DIVTIMER0REL);
     div_freq   ((unsigned   *)   CTIMEREL,   (unsigned   *)
DADOTIMER1REL, CTIMER1REL, (unsigned) DIVTIMER1REL);
     div_freq   ((unsigned   *)   CTIMEREL,   (unsigned   *)
DADOTIMER2REL, CTIMER2REL, (unsigned) DIVTIMER2REL);
}
/*****************************************************
* TITULO  : ler_timer                                *
* FUNCAO  : ler latches de dados de um timer         *
* ENTRADAS: endereco do timer                        *
* SAIDAS  : valor da contagem armazenada no latch    *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

int ler_timer (unsigned *end_timer)
{
     char contagem, mscontagem, modo;

     modo = input (end_timer);

     contagem = input (end_timer);

     if ((modo & RW_WORD) == RW_WORD)
     {    mscontagem = (input (end_timer)) << 8;
          contagem += mscontagem;
     }
        return contagem;
}
/*****************************************************
* TITULO  : disp_rel                                 *
* FUNCAO  : mostrar horario no LCD                   *
* ENTRADAS: 0 - horas:minutos, 1 - minutos:segundos*
* SAIDAS  :                                          *
* TESTADA : SIM [X]   NAO [ ]                        *
*****************************************************/

void disp_rel (char opcao)
{
     char horario[4];
     unsigned *dig_0_1, *dig_2_3, *contr_timer_rel;

     char dois_digitos;

     if (opcao == 0)
```

```
     {    dig_0_1 = (unsigned *) DADOTIMER1REL;
          dig_2_3 = (unsigned *) DADOTIMER2REL;
          output ((unsigned *) CTIMEREL, RBSTCOUNTREL12);
     }
     else
     {    dig_0_1 = (unsigned *) DADOTIMER0REL;
          dig_2_3 = (unsigned *) DADOTIMER1REL;
          output ((unsigned *) CTIMEREL, RBSTCOUNTREL01);
     }
     dois_digitos = hexbcd (60 - ler_timer (dig_0_1));
     horario[0] = dois_digitos & 0x0F;
     horario[1] = (dois_digitos & 0xF0) >> 4;
     if (opcao == 0)
          dois_digitos = hexbcd (24 - ler_timer (dig_2_3));
     else
          dois_digitos = hexbcd (60 - ler_timer (dig_2_3));
     horario[2] = dois_digitos & 0x0F;
     horario[3] = ((dois_digitos & 0xF0) >> 4) + 0x20;
     display (horario);
}
/**************************************************
* TITULO  : acerto                                *
* FUNCAO  : acertar  horas e minutos do relogio   *
*         e zerar segundos                        *
* ENTRADAS: horas e minutos (char)                *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void acerto (char horas, char minutos)
{
     char leitura_contador;
     char lsbase, msbase;
     int i;

     if ((minutos == 0) && (horas != 0))
     {    minutos = 60;
          horas--;
     }

     if ((horas == 0) && (minutos != 0))
          horas = 24;

     if ((horas == 0) && (minutos == 0))
     {    horas = 23;
          minutos = 60;
     }

     div_freq   ((unsigned   *)   CTIMER1BASE,   (unsigned   *)
DADOTIMER1BASE, MODOTIMER1BASE, (unsigned) BASE_ACERTO);

     for (i = 0; i <= CONF_HORAS; i++)
     {    do
          {    output ((unsigned *) CTIMEREL, COUNTERLATCH2);
               leitura_contador   =   input   ((unsigned   *)
DADOTIMER2REL);
          }
          while (leitura_contador != horas);
     }
     for (i = 0; i <= CONF_MINUTOS; i++)
     {    do
```

```
          {    output ((unsigned *) CTIMEREL, COUNTERLATCH1);
               leitura_contador   =   input   ((unsigned   *)
DADOTIMER1REL);
          }
          while (leitura_contador != minutos);
     }

     minuto = (60 - minutos) | 0xB000;

     div_freq   ((unsigned   *)   CTIMER1BASE,   (unsigned   *)
DADOTIMER1BASE, MODOTIMER1BASE, (unsigned) BASE_RELOGIO);
     disp_rel (0);
}
/**************************************************
* TITULO  : atual_min                             *
* FUNCAO  : atualizar minutos na RAM auxiliar e   *
*        horario no display                       *
* ENTRADAS: pedido de interrupcao do timer de seg. *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void interrupt atual_min (void)
{
     asm  {
          mov  ax,DS
          mov  ds,ax };


     output ((unsigned *) 0xC2, COUNTERLATCH1);
     minuto = (60 - (input ((unsigned *) DADOTIMER1REL))) |
0xB000;

     disp_rel (0);

     enable ();
     output ((unsigned *) INTRES, INTRES2);
     output ((unsigned *) INTRES, 0xFF);

     output ((unsigned *) EOCW2, EOINE);
}
/**************************************************
* TITULO  : atual_hora                            *
* FUNCAO  : atualizar horas na RAM auxiliar       *
*        e na memoria de aquisicao                *
* ENTRADAS: pedido de interrupcao do timer de min. *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void interrupt atual_hora (void)
{
     asm  {
          mov  ax,DS
          mov  ds,ax};


     output ((unsigned *) 0xC2, COUNTERLATCH2);
     hora = (24 - (input ((unsigned *) DADOTIMER2REL))) |
0xC000;
```

```
     if (flag_dist_hora == 1)
     {    ram_ant_hora = ram_aquis;
          grava_3RAM (ram_aquis++, hora);
          flag_dist_hora = 0;
        }
     else
          grava_3RAM (ram_ant_hora, hora);

     enable ();
     output ((unsigned *) INTRES, INTRES1);
     output ((unsigned *) INTRES, 0xFF);

     output ((unsigned *) EOCW2, EOINE);
}
/**************************************************
* TITULO  : novo_horario                          *
* FUNCAO  : receber do teclado o horario para     *
*         atualizar o relogio                     *
* ENTRADAS:                                       *
* SAIDAS  :                                       *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void novo_horario (void)
{
     int i;
     char acerto_horas, acerto_minutos, horario [2];

     dadoduplo (HORAPROMPT, horario);

     if (tecla != (unsigned) ANULA)
     {    acerto_horas = 24 - horario [1];
          acerto_minutos = 60 - horario [0];
          acerto (acerto_horas, acerto_minutos);
          minuto = horario [0] | 0xB000;
          hora = horario [1] | 0xC000;
     }
     else
     {    minuto = (60 - (input ((unsigned *) DADOTIMER1REL)))
| 0xB000;
          hora = (24 - (input ((unsigned *) DADOTIMER2REL))) |
0xC000;
     }
     enable ();
     output ((unsigned *) INTRES, INTRES0 & INTRES1 & INTRES2
& INTRES7);
     output ((unsigned *) INTRES, 0xFF);
     output ((unsigned *) FFPTEC, 0);
}
```

```
/**************************************************
*              MAT.C                              *
*            ROTINAS DESTE ARQUIVO                *
* TITULO  : POT10                                 *
* FUNCAO  : elevar 10 a n-esima potencia positiva *
* TITULO  : DIVFLUT                               *
* FUNCAO  : simular uma divisao em ponto flutuante *
**************************************************/

char hexbcd (int);

/**************************************************
* TITULO  : pot10                                 *
* FUNCAO  : elevar 10 a n-esima potencia positiva *
* ENTRADAS: n (unsigned)                          *
* SAIDAS  : 10^n (long)                           *
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

long pot10 (int n)
{
     long resultado = 1;

     while (n > 0)
     {    resultado *= 10;
          n--;
     }
     return resultado;
}
/**************************************************
* TITULO  : divflut                               *
* FUNCAO  : simular uma divisao em ponto flutuante *
* ENTRADAS: dividendo (unsigned), divisor (idem)  *
*        e ponteiro da cadeia de saida das par-   *
*           tes inteira e fracionaria  (2 casas)  *
* SAIDAS  : quociente (unsigned) e resto (unsigned)*
* TESTADA : SIM [X]   NAO [ ]                     *
**************************************************/

void divflut (unsigned dividendo, unsigned divisor, char
*saida)
{
     unsigned quociente, resto;
     char auxconv;

     quociente = dividendo / divisor;

     resto = (100 * (dividendo - quociente * divisor)) /
divisor;

     auxconv = hexbcd (quociente);

     saida [3] = auxconv >> 4;
     saida [2] = auxconv & 0xF;

     auxconv = hexbcd (resto);

     saida [1] = auxconv >> 4;
     saida [0] = auxconv & 0xF;
}
```

```
/**************************************************
* TITULO   : potxy                                 *
* FUNCAO   : elevar um numero x a um numero y      *
* ENTRADAS: x (unsigned) e y (unsigned)            *
* SAIDAS   : resultado (unsigned)                  *
* TESTADA : SIM [X]   NAO [ ]                      *
**************************************************/

unsigned potxy (unsigned base, unsigned potencia)
{
     unsigned resultado = 1, contador;
          for (contador = 1; contador <= potencia; contador++)
               resultado *= base;
     return resultado;
}
```

```
/**************************************************
*                                                 *
*                    RELAT.C                      *
*      PROGRAMA DE EMISSÃO DE RELATÓRIOS          *
*                                                 *
**************************************************/


#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>


#define  MAXLIN          54
#define  MAXDADOS        1000
#define    TAM_LOCAL     50

struct inic_arq {
              unsigned int byte_menor      : 8;
              unsigned int byte_maior      : 8; };

struct dados {
              unsigned int medida_disturbio :12;
              unsigned int tipo_disturbio   : 4; };

struct rf2 {
              unsigned int freq_rf2   : 10;
              unsigned int nao_ocup   :  6; };

struct ampl {
              unsigned int nivel      : 8;
              unsigned int nao_ocup   : 8; };

struct hora {
              unsigned int hora       :  5;
              unsigned int nao_ocup   : 11; };

struct ano  {
              unsigned int ano        : 7;
              unsigned int nao_ocup   : 9; };

struct segmin {
              unsigned int minutos    : 6;
              unsigned int segundos   : 6;
              unsigned int nao_ocup   : 4; };

struct mes_dia {
              unsigned int dia        : 5;
              unsigned int nao_ocup1  : 3;
              unsigned int mes        : 4;
              unsigned int nao_ocup2  : 4; };
/*
struct freq60  {
              unsigned int pulsos     : 7;
              unsigned int nao_ocup   : 9; };
*/

unsigned *tab_dados;
char flag_impr = 'N', flag_pbat = '0', flag_nbat = '0';
int  cont_dist60,  cont_rf1,  cont_rf2,  cont_rf3,  cont_trans,
```

```
cont_sobre, cont_sub;
int d_dia, d_mes, d_ano, h_hora, h_min, h_seg, cont_pag,
cont_lin;
int d_dia1, d_mes1, d_ano1, h_hora1, h_min1, h_seg1,
cont_dados;
int d_dia2, d_mes2, d_ano2, h_hora2, h_min2, h_seg2,final;
float Vbp, Vbn, Vbpfinal, Vbnfinal;;
FILE *impressora;
c h a r i g u a l _ l i n [ ] =
"==========================================================
================\n";
c h a r l i n _ s e p [ ] =
"----------------------------------------------------------
----------------";
char local [TAM_LOCAL];

#include "rot_rel.c"

/* anlprint: le dados no disco, armazena na area alocada, ana-
*/
/*          lisa o tipo de informacao que cada dado contem
*/
/*        e apresenta o resultado na tela.                  */
main()
{
     struct dados *lote;
     int i, cont_lin_delta = 0;
     unsigned *topo_arq;
     c h a r a s t e r _ l i n [ ] =
"**********************************************************
****************\n";

     cont_dist60 = 0;
     cont_rf1 = 0;
     cont_rf2 = 0;
     cont_rf3 = 0;
     cont_trans = 0;
     cont_sobre = 0;
     cont_sub = 0;


     fprintf (impressora,"%d\n", MAXLIN);

     tab_dados = (unsigned *)malloc(MAXDADOS * sizeof(int));

        if (!tab_dados)
       { printf ("ALOCACAO DE MEMORIA MAL SUCEDIDA\n");
     exit(0); };

     topo_arq = tab_dados;

    clrscr();
     cont_pag = 1;
     impressora = fopen("LPT1","w");

          le_disco();
    tab_dados = topo_arq;
     printf ("TOTAL DE BYTES ARMAZENADOS = %d\n", final * 2);
     do
       { clrscr();
```

```
          printf ("DESEJA IMPRESSAO DOS DADOS? (S/N)");
          flag_impr = getchar ();
          if (flag_impr == 's')
            flag_impr = 'S';
          if (flag_impr == 'n')
            flag_impr = 'N'; }
     while (flag_impr != 's' && flag_impr != 'S' && flag_impr
!= 'n'
             && flag_impr != 'N');
     fprintf (impressora,"%s", aster_lin);
     fprintf (impressora,"*   MONITOR PARA CARACTERIZACAO DE
DISTURBIOS EM REDES A.C.                    *\n");
     fprintf (impressora,"*   ORIENTADOR : PROF. DR. VALENTIN
OBAC RODA                                  *\n");
     fprintf (impressora,"*   ALUNO        : ALVARO MACEDO DA
SILVA       PROGRAMA: MESTRADO             *\n");
     fprintf (impressora,"%s\n", aster_lin);
     dir_arq();
     if (flag_impr == 'S')
       { fprintf (impressora,"\n\n
DESCRICAO DOS DISTURBIOS\n");
          fprintf (impressora,"%s\n\n", lin_sep);
          fprintf (impressora,"      DATA     HORARIO
TIPO DE DISTURBIO                VALOR  UNIDADE\n\n");
          fprintf (impressora,"%s\n", lin_sep); };
     if (flag_impr == 'S')
       cont_lin = 18;
     else
       cont_lin = 9;
     cont_dados = 8;
     do
       { lote = (struct dados *)++tab_dados;
       switch(lote->tipo_disturbio)
        { case 0:
                dist60();
                break;
          case 1:
                rf1();
            break;
          case 2:
                rf2();
            break;
          case 3:
                rf3();
            break;
          case 4:
                trans();
                break;
          case 8:
                ampl60();
                break;
          case 9:
                pbat();
                break;
            case 10:
                nbat();
            break;
          case 12:
                hora();
            break;
          case 13:
```

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FISICA

```
            mes_dia();
          break;
        case 14:
       ano();
      }
       }
    while (cont_dados++ != final);

    if (cont_lin != 3 && flag_impr == 'S')
      { for (i = cont_lin; i <= (MAXLIN - 1); i++)
           { fprintf (impressora,"\n");
              cont_lin++; }
         aj_pag2 ();
         fprintf (impressora,"%s\n",lin_sep); }

    if (flag_impr == 'N')
    {    fprintf (impressora,"\n%s\n", lin_sep);
         cont_lin += 3;
    }

   fprintf (impressora,"
SINOPSE\n%s\n", lin_sep);
    if (flag_pbat == '1')
    {    fprintf (impressora,"  BATERIA POSITIVA ABAIXO DE 8V
AS %d:%d'%d\" H DE %d/%d/%d\n\n", h_hora1, h_min1, h_seg1,
d_dia1, d_mes1, d_ano1);
         cont_lin_delta += 2;
    }
    if (flag_nbat == '1')
    {    fprintf (impressora,"  BATERIA NEGATIVA ABAIXO DE 8V
AS %d:%d'%d\" H DE %d/%d/%d\n\n", h_hora2, h_min2, h_seg2,
d_dia2, d_mes2, d_ano2);
         cont_lin_delta += 2;
    }
    fprintf (impressora,"    BATERIA POSITIVA AO FINAL DO
LEVANTAMENTO:  %3.1f V\n\n", Vbpfinal);
    fprintf (impressora,"    BATERIA NEGATIVA AO FINAL DO
LEVANTAMENTO: -%3.1f V\n\n", Vbnfinal);
    fprintf (impressora,"      MEMORIA OCUPADA TOTAL:  %d
BYTES\n\n", final * 2);
    fprintf (impressora,"    DISTORCAO NA FREQUENCIA DE 60 Hz
          : %d\n\n", cont_dist60);
    fprintf (impressora,"    INTERFERENCIA DE R.F. NA FAIXA
0,6 A 10   kHz: %d\n\n", cont_rf1);
    fprintf (impressora,"    INTERFERENCIA DE R.F. NA FAIXA 10
A 100  kHz: %d\n\n", cont_rf2);
    fprintf (impressora,"    INTERFERENCIA DE R.F. NA FAIXA
100 A 1000 kHz: %d\n\n", cont_rf3);
    fprintf (impressora,"    TRANSIENTES DE CURTA DURACAO
          : %d\n\n", cont_trans);
    fprintf (impressora,"    SOBRETENSOES (ACIMA DE 140 Vrms)
         : %d\n\n", cont_sobre);
    fprintf (impressora,"    SUBTENSOES  (ABAIXO DE 108 Vrms)
         : %d\n", cont_sub);
    cont_lin += (21 + cont_lin_delta);
    if (cont_lin < MAXLIN)
      for (i = cont_lin; i <= MAXLIN; i++)
       fprintf (impressora,"\n");
   fprintf (impressora,"\n\n
     %d\n\n\n\n\n\n\n\n\n\n\n", cont_pag++);
    fclose (impressora);
```

```
    free (tab_dados);
    return 0;
}
```

```
/*****************************************************
*                                                    *
*                    ROT_REL.C                       *
*          ROTINAS DO PROGRAMA RELATÓRIO             *
*                                                    *
*****************************************************/


int le_disco (void);
void dist60 (void);      /* distorcao em 60 Hz */
void rf1 (void);         /* rfi 0,6 a 10 kHz   */
void rf2 (void);         /* rfi 10 a 100 kHz   */
void rf3 (void);         /* rfi 100 a 1000 kHz */
void trans (void);       /* transiente rapido  */
void ampl60 (void);      /* distorcao amplitude em 60 Hz  */
void pbat (void);        /* bateria positiva descarregada */
void nbat (void);        /* bateria nebativa descarregada */
void min_seg (void);     /* horario - minutos e segundos  */
void hora (void);        /* horario - hora                */
void mes_dia (void);     /* calendario - mes e dia        */
void ano (void);         /* calendario - ano              */
void dir_arq (void);    /* imprime topo do arquivo       */
void  aj_pag  (void);        /*  muda  pagina  de  impressao  se
necessario */
void aj_pag2 (void);


/*le_disco: le os dados da placa 3RAM armazenados no disco a:
*/

int le_disco(void)
{
     int unsigned i, j=0;
    FILE *entrada;

    entrada = fopen("dados.mon", "r");
     for (i=0; i < 8; i++)
      { *tab_dados = (unsigned) getw(entrada);
     tab_dados++;
          j++; }
     --tab_dados;
     final = (unsigned) (*tab_dados)/2;
    tab_dados++;
    do
      { if (i > MAXDADOS)
       { fprintf (impressora,"TENTATIVA DE LER ALEM DO FIM DO
ARQUIVO\n");
          break; };
     *tab_dados = getw(entrada);
     tab_dados++;
        j++; }
     while (i++ != final);

    fclose(entrada);
     return --j;
}

/* dist60: case 0 */

void dist60 (void)
```

```
{
    struct dados *tab1;
    struct segmin *tab2;
    float clock = 125e+3, freq;

     tab1 = (struct dados*)tab_dados++;
     cont_dados++;
     freq = clock / (float)tab1->medida_disturbio;

    tab2 = (struct segmin *)tab_dados;
    h_min = (int)tab2->minutos;
    h_seg = (int)tab2->segundos;

     if (flag_impr == 'S')
       { fprintf (impressora,"    %2d/%2d/%2d   %2d:%2d'%2d\"
DISTORCAO DE FREQUENCIA EM 60 Hz        %5.2f   Hz\n", d_dia,
d_mes, d_ano, h_hora, h_min, h_seg, freq);
          aj_pag ();
          if ((cont_lin < MAXLIN) && (cont_lin != 3))
            fprintf (impressora,"%s\n", lin_sep);
          aj_pag (); };
     cont_dist60++;
}

/* rf1: case 1 */

void rf1 (void)
{
    struct dados  *tab1;
    struct segmin *tab2;
    float clock = 2e+6, freq;

     tab1 = (struct dados *)tab_dados++;
    cont_dados++;
    freq = clock / (float)tab1->medida_disturbio;

     tab2 = (struct segmin *)tab_dados;
    h_min = (int)tab2->minutos;
    h_seg = (int)tab2->segundos;

     if (flag_impr == 'S')
       { fprintf (impressora,"    %2d/%2d/%2d   %2d:%2d'%2d\"
INTERFERENCIA DE RADIO - FREQUENCIA    %6.2f    Hz\n", d_dia,
d_mes, d_ano, h_hora, h_min, h_seg, freq);
          aj_pag ();
          if ((cont_lin < MAXLIN) && (cont_lin != 3))
            fprintf (impressora,"%s\n", lin_sep);
          aj_pag (); };
     cont_rf1++;
}

/* rf2: case 2 */

void rf2 (void)
{
    struct rf2 *tab1;
    struct segmin *tab2;
    float clock = 25000, freq;

     tab1 = (struct rf2 *)tab_dados++;
```

```
    cont_dados++;
     freq = (float)tab1->freq_rf2 * clock / 255e+3;

     tab2 = (struct segmin *)tab_dados;
    h_min = (int)tab2->minutos;
    h_seg = (int)tab2->segundos;

     if (flag_impr == 'S')
       { fprintf (impressora,"    %2d/%2d/%2d   %2d:%2d'%2d\"
INTERFERENCIA DE RADIO - FREQUENCIA    %6.2f   kHz\n", d_dia,
d_mes, d_ano, h_hora, h_min, h_seg, freq);
          aj_pag ();
          if ((cont_lin < MAXLIN) && (cont_lin != 3))
            fprintf (impressora,"%s\n", lin_sep);
          aj_pag (); };
     cont_rf2++;
}

/* rf3: case 3 */

void rf3 (void)
{
    struct rf2 *tab1;
    struct segmin *tab2;
    float clock = 250e+3, freq;

     tab1 = (struct rf2 *)tab_dados++;
    cont_dados++;
     freq = (float)tab1->freq_rf2 * clock / 255e+3;

    tab2 = (struct segmin *)tab_dados;
    h_min = (int)tab2->minutos;
    h_seg = (int)tab2->segundos;

     if (flag_impr == 'S')
       {  fprintf  (impressora,"       %d/%d/%d     %d:%d:%d
INTERFERENCIA DE RADIO - FREQUENCIA    %6.2f   kHz\n", d_dia,
d_mes, d_ano, h_hora, h_min, h_seg, freq);
          aj_pag ();
          if ((cont_lin < MAXLIN) && (cont_lin != 3))
            fprintf (impressora,"%s\n", lin_sep);
          aj_pag (); };
     cont_rf3++;
}

/* trans: cases 4, 5, 6 e 7 */

void trans (void)
{
    struct dados  *tab1;
    struct ampl *tab2;
    struct segmin *tab3;
    float td, freq, Vp;
    int i;

     tab1 = (struct dados *)tab_dados++;
    cont_dados++;
    td = (float)tab1->medida_disturbio / 8;

     tab1 = (struct dados *)tab_dados++;
```

```
    cont_dados++;
     if (td != 0)
          freq = (float)tab1->medida_disturbio * 1e+3 / td;

     tab2 = (struct ampl *)tab_dados++;
    cont_dados++;
    Vp = 23.63 * (float)tab2->nivel;

     tab3 = (struct segmin *)tab_dados;
    h_min = (int)tab3->minutos;
    h_seg = (int)tab3->segundos;

     if (cont_lin > (MAXLIN - 5))
       { for (i = cont_lin; i <= (MAXLIN - 1); i++)
            {  fprintf (impressora,"\n");
               cont_lin++; };
          aj_pag (); };

     if ((flag_impr == 'S') && (td !=0))
     {    fprintf (impressora,"    %2d/%2d/%2d   %2d:%2d'%2d\"
TRANSIENTE  RAPIDO\n",  d_dia,  d_mes,  d_ano,  h_hora,  h_min,
h_seg);
          aj_pag ();
          fprintf (impressora,"
DURACAO                          %6.2f   us\n", td);
          aj_pag ();
          fprintf (impressora,"
FREQUENCIA                       %7.2f   kHz\n", freq);
          aj_pag ();
          fprintf (impressora,"
MAXIMA AMPLITUDE                 %6.2f   V\n", Vp);
          aj_pag ();
          if ((cont_lin < MAXLIN) && (cont_lin != 3))
            fprintf (impressora,"%s\n", lin_sep);
          aj_pag ();
     }

     if (td != 0)
          cont_trans++;
}

/* ampl60: case 8 */

void ampl60 (void)
{
    struct ampl *tab1;
    struct segmin *tab2;
    float V60;

     tab1 = (struct ampl *)tab_dados++;
    cont_dados++;
    V60 = 2.11 * (float)tab1->nivel - 269.41;

    tab2 = (struct segmin *)tab_dados;
    h_min = (int)tab2->minutos;
     h_seg = (int)tab2->segundos;

     if (flag_impr == 'S')
       { fprintf (impressora,"    %2d/%2d/%2d   %2d:%2d'%2d\"
DISTORCAO DE AMPLITUDE EM 60 Hz          %6.2f    V\n", d_dia,
```

```
d_mes, d_ano, h_hora, h_min, h_seg, V60);
          aj_pag ();
          if ((cont_lin < MAXLIN) && (cont_lin != 3))
            fprintf (impressora,"%s\n", lin_sep);
          aj_pag (); };
     if (V60 > 197.6)
       cont_sobre++;
     else
       if (V60 < 152.7)
          cont_sub++;
       else
          printf ("ATENCAO: AMPLITUDE NORMAL EM 60 Hz REPORTADA
COMO DISTURBIO\n");
}

/* pbat: case 9 */

void pbat (void)
{
    struct ampl *tab1;
    struct segmin *tab2;
     tab1 = (struct ampl *)tab_dados++;
    cont_dados++;
    Vbp = 0.0471 * (float)tab1->nivel;
    d_dia1 = d_dia;
    d_mes1 = d_mes;
    d_ano1 = d_ano;
    h_hora1 = h_hora;

    tab2 = (struct segmin *)tab_dados;
    h_min1 = (int)tab2->minutos;
    h_seg1 = (int)tab2->segundos;
    flag_pbat = '1';
}

/* nbat: case 10 */

void nbat (void)
{
    struct ampl *tab1;
    struct segmin *tab2;
     tab1 = (struct ampl *)tab_dados++;
    cont_dados++;
    Vbn = -0.0471 * (float)tab1->nivel;
    d_dia2 = d_dia;
    d_mes2 = d_mes;
    d_ano2 = d_ano;
    h_hora2 = h_hora;

    tab2 = (struct segmin *)tab_dados;
    h_min2 = (int)tab2->minutos;
    h_seg2 = (int)tab2->segundos;
    flag_nbat = '1';
}

/* hora: case 12 */

void hora (void)
{
    struct hora *tab;
```

```
    tab = (struct hora *)tab_dados;
    h_hora = (int)tab->hora;
}

/* mes_dia: case 13 */

void mes_dia (void)
{
    struct mes_dia *tab;

    tab = (struct mes_dia *)tab_dados;
    d_dia = (int)tab->dia;
    d_mes = (int)tab->mes;
}

/* ano: case 14 */

void ano (void)
{
    struct ano *tab;

    tab = (struct ano *)tab_dados;
    d_ano = (int)tab->ano;
}

/* dir_arq: imprime topo do arquivo */

void dir_arq (void)
{
    struct inic_arq *historico;

     historico = (struct inic_arq *)tab_dados++;
     fprintf (impressora,"    INICIO DA   AQUISICAO == DIA:
%2d/%2d/", historico->byte_menor, historico->byte_maior); /*
DIA/MES */

     historico = (struct inic_arq *)tab_dados++;
     fprintf    (impressora,"%2d           HORARIO:    %2d:",
historico->byte_menor, historico->byte_maior);     /* ANO
HORA*/

     historico = (struct inic_arq *)tab_dados++;
     fprintf    (impressora,"%2d'%2d\"    H\n",
historico->byte_menor,    historico->byte_maior);       /*
MINUTOS:SEGUNDOS */

     historico = (struct inic_arq *)tab_dados++;
     fprintf (impressora,"    FINAL   DA   AQUISICAO == DIA:
%2d/%2d/", historico->byte_menor, historico->byte_maior); /*
DIA/MES */

     historico = (struct inic_arq *)tab_dados++;
     fprintf    (impressora,"%2d           HORARIO:    %2d:",
historico->byte_menor, historico->byte_maior);     /* ANO
HORA*/

     historico = (struct inic_arq *)tab_dados++;
     fprintf    (impressora,"%2d'%2d\"    H\n",
historico->byte_menor,    historico->byte_maior);       /*
MINUTOS:SEGUNDOS */
```

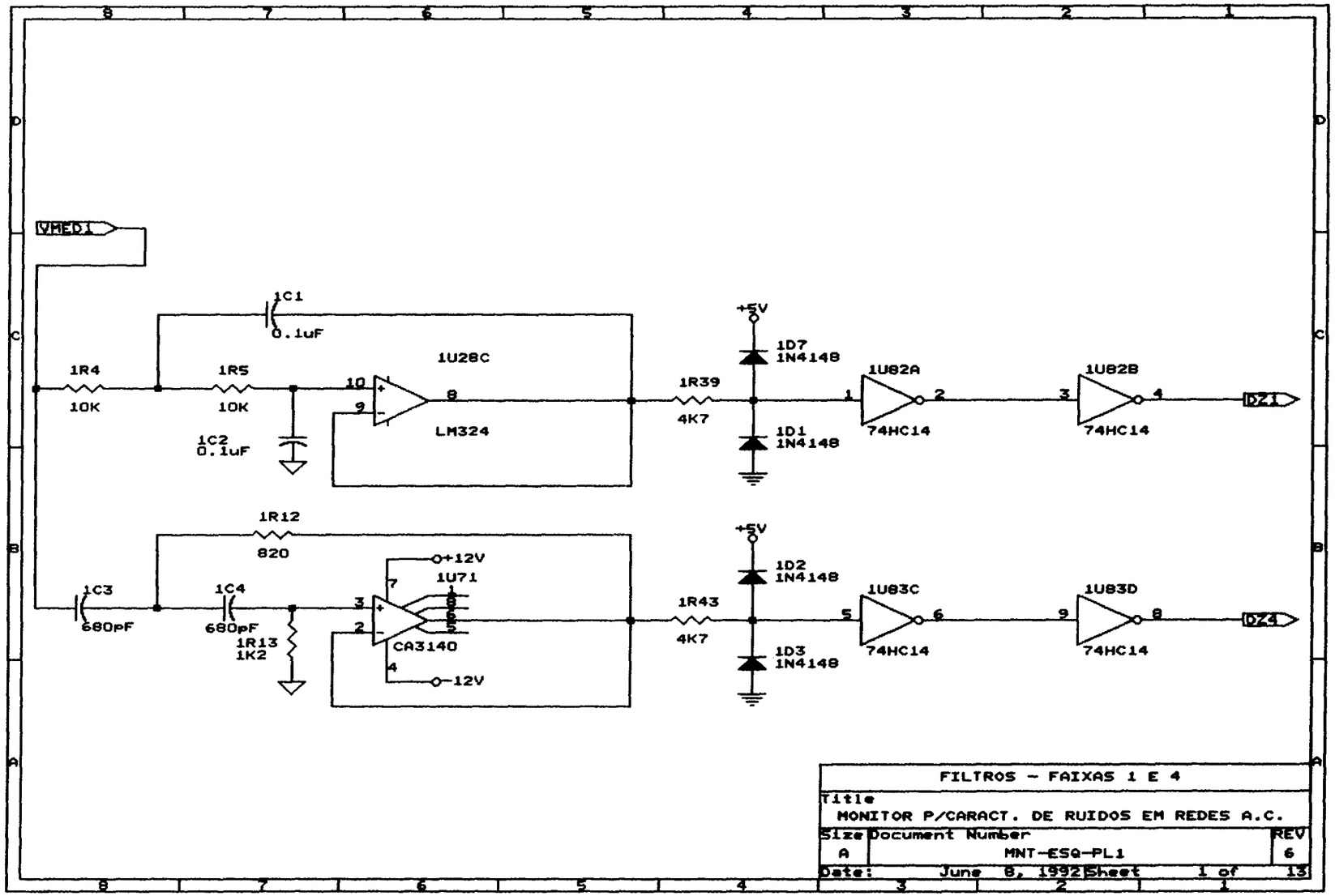
```
     historico = (struct inic_arq *)tab_dados++;
    Vbpfinal = ((float)historico->byte_menor) * 15/255;
    Vbnfinal = ((float)historico->byte_maior) * 15/255;
}

/* aj_pag: ajusta pagina de impressao se necessario */

void aj_pag (void)
{
    cont_lin++;
     if (cont_lin == (MAXLIN + 1))
       { fprintf (impressora,"%s\n\n
            %d\n\n\n\n\n\n\n\n\n", igual_lin, cont_pag++);
          fprintf (impressora,"      DATA     HORARIO
TIPO DE DISTURBIO                  VALOR    UNIDADE\n\n%s\n",
lin_sep);
          cont_lin = 3; };
}

/* aj_pag2: ajusta pagina de impressao se necessario */

void aj_pag2 (void)
{
    cont_lin++;
     if (cont_lin == (MAXLIN + 1))
       {  fprintf (impressora,"%s\n\n
            %d\n\n\n\n\n\n\n\n\n", igual_lin, cont_pag++);
          cont_lin = 1; };
}
```

```
/*****************************************************
*                                                    *
*                    LERAM.C                         *
*    PROGRAMA DE LEITURA DA PLACA DE AQUISICAO       *
*                                                    *
*****************************************************/


#include <stdio.h>
#include <conio.h>
#include <alloc.h>
#include <process.h>

#define   DS            0xD000    /*  SEGMENTO  DE  DADOS  DE
AQUISICAO */
#define   MAXDADOS      0x7FFF
#define        TAM_LOCAL      50

void le_ram(unsigned, unsigned *);

/*   char local [TAM_LOCAL];  */

/* leram: le dados na placa 3RAM e armazena em disco flexivel
*/

void main()
{
     unsigned *end_inic;
/*
     printf ("Local do levantamento ");
     lelinha (local, TAM_LOCAL);
*/
     end_inic = (unsigned *)0 ; /* define endereco inicial da
memoria 3RAM */

     le_ram((unsigned) DS, end_inic);
}


/*ler_int: le um inteiro nao sinalizado de uma posicao da   */
/*      memoria de aquisicao                              */

unsigned ler_int (unsigned seg, unsigned *end)
{
     unsigned dado;
     asm {
          push      ds
          push si
          mov  ax,  seg
          mov  ds,  ax
          mov  si,  end
          mov  ax,  [si]
          mov  dado,     ax
          pop  si
          pop  ds};
     return dado;
}

/*le_ram: le os dados da placa 3RAM e armazena em disco     */
```

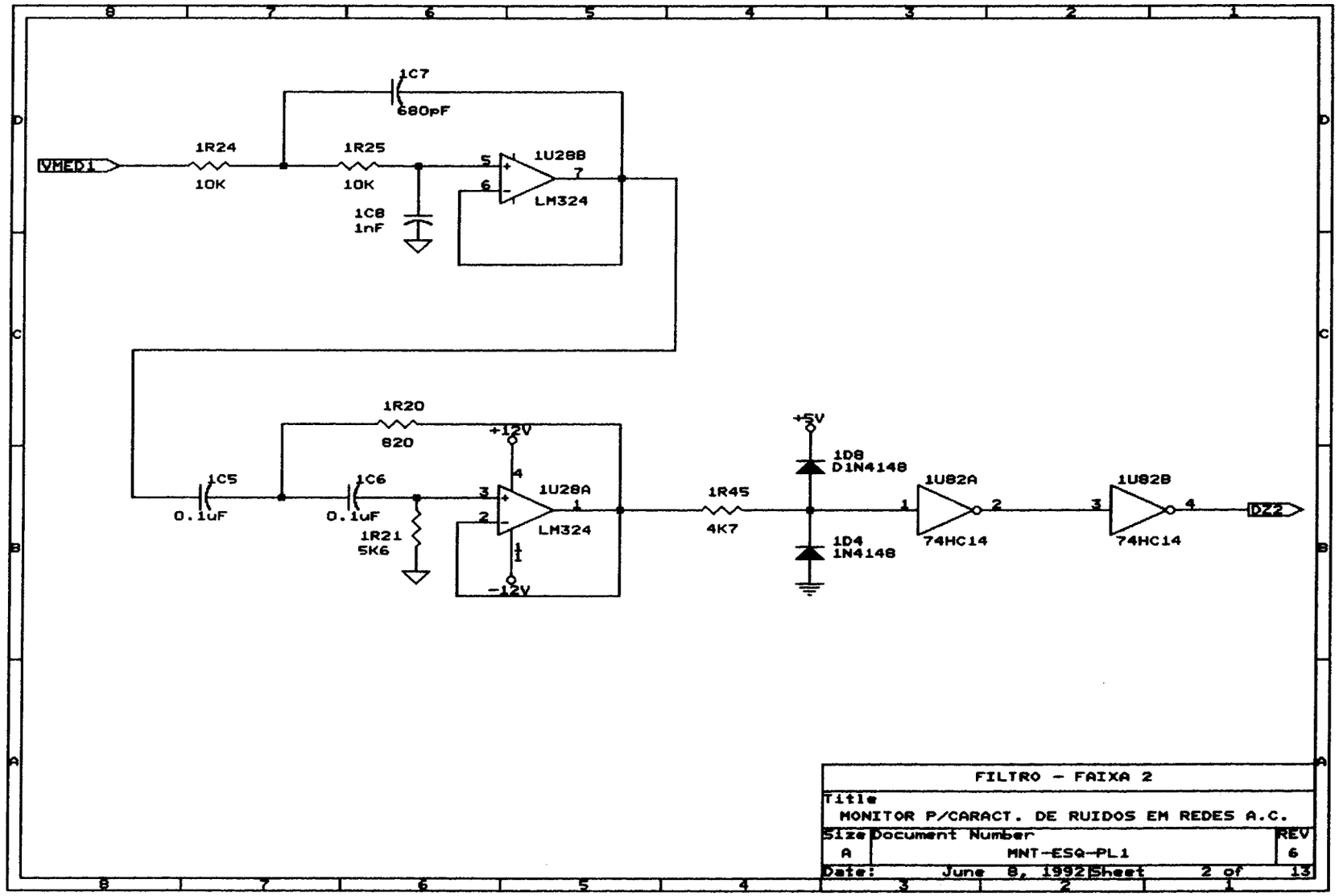
```
void le_ram(unsigned segmento, unsigned *memoria)
{
     int i, j=0, c;
     unsigned dado, *final;
     FILE *saida;
     char nomearq[] = "dados.mon";
/*
     printf ("Nome do arquivo ");
     gets (nomearq);
*/

     saida = fopen(nomearq, "w");

    for (i=0; i <= 7; i++)
       {     dado = ler_int (segmento, memoria++);
          putw(dado, saida);
          j++;
       }
     final = (unsigned *) dado;

    do
       { if (j > MAXDADOS)
          { printf("ENDERECO ALEM DO LIMITE DE MEMORIA\n");
            break; }
          dado = ler_int (segmento, memoria);
          putw(dado, saida);
          j++; }
     while (memoria++ < final);
/*   i = 0;
     while ((c = local [i++]) != EOF && c != '\n')
       fputc (c, saida);
     fputc ('\n', saida);
*/
       fclose (saida);
}
```

APÊNDICE B

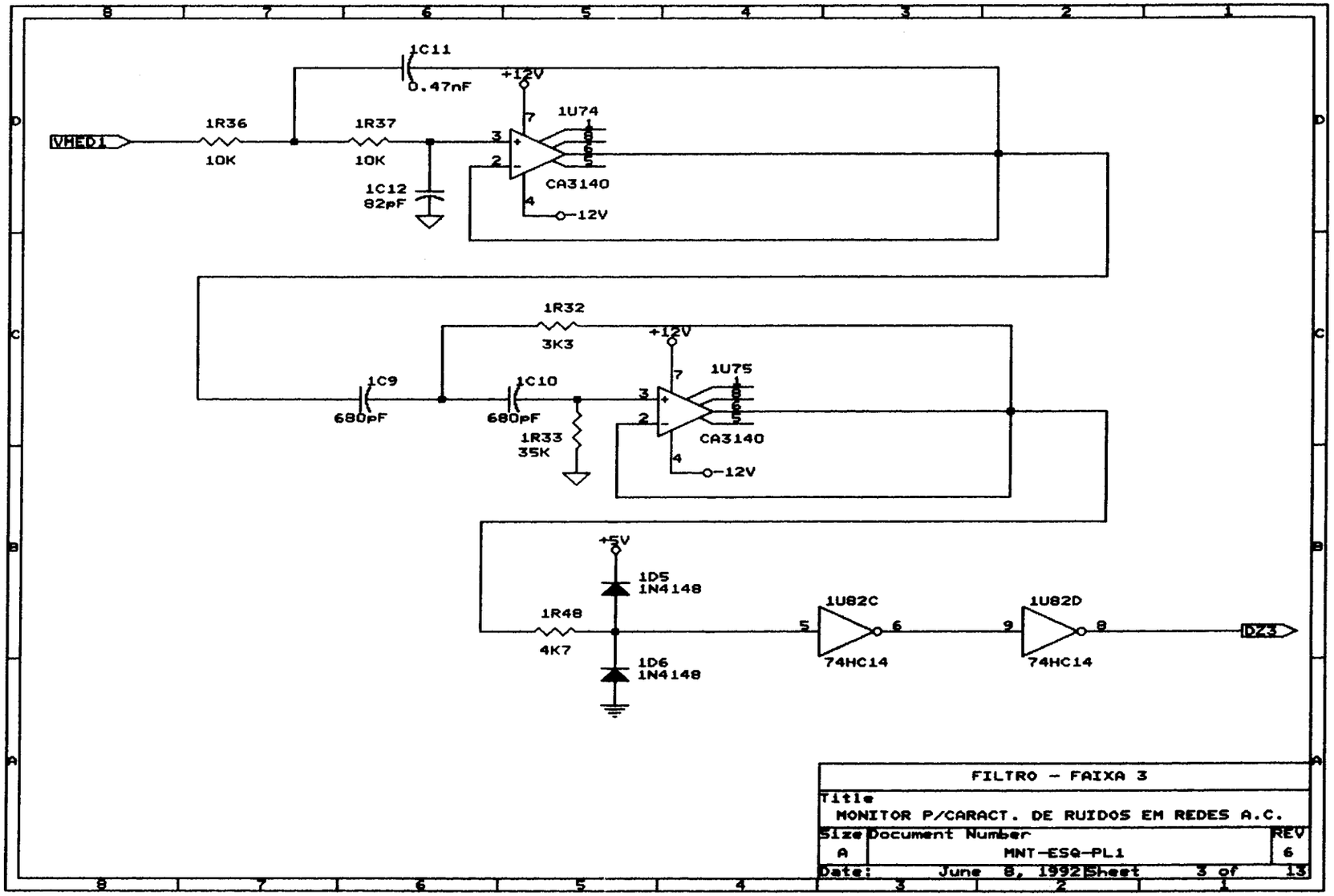
CIRCUITOS ELETRÔNICOS



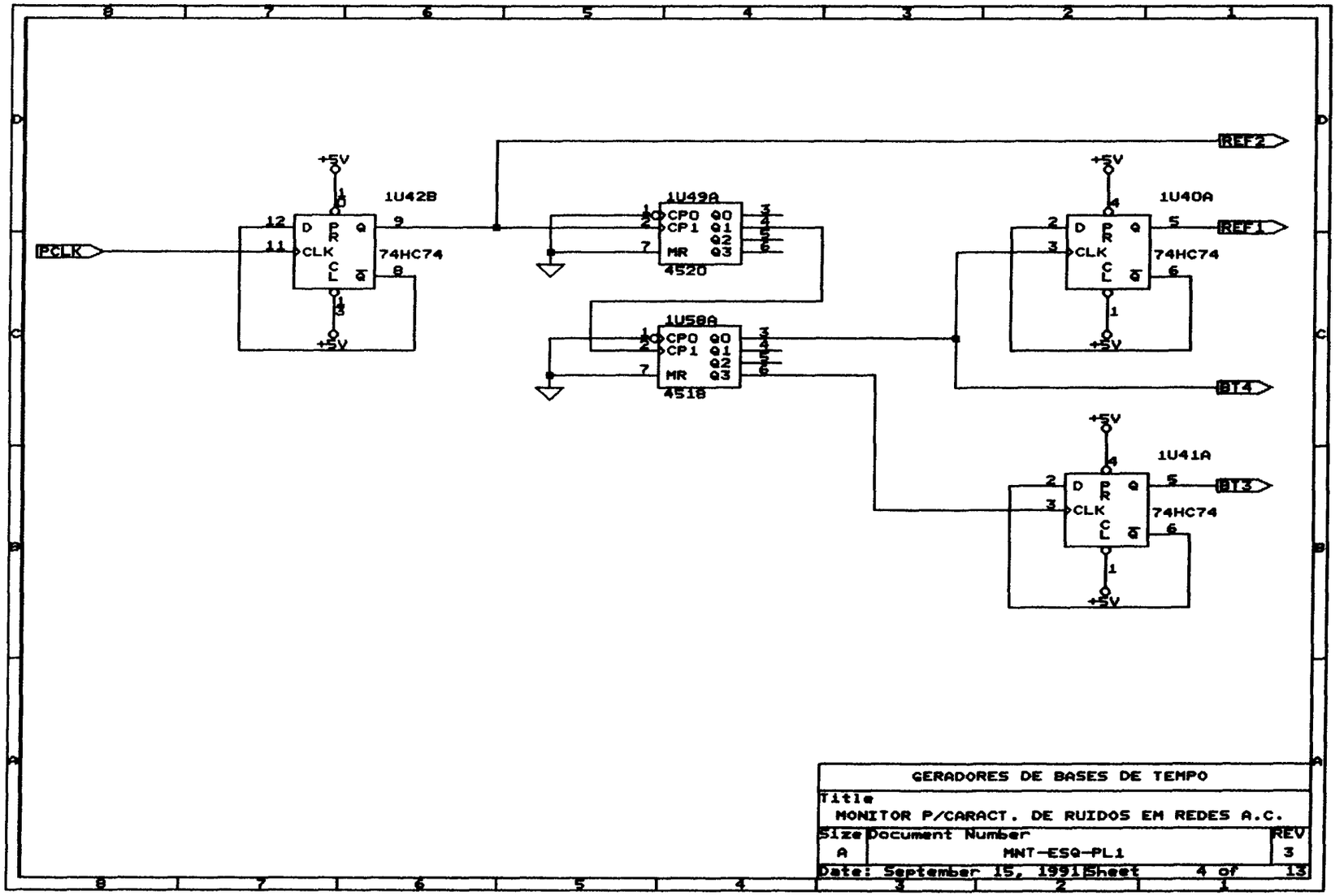
FILTROS - FAIXAS 1 E 4	
Title MONITOR P/CARACT. DE RUIDOS EM REDES A.C.	
Size A	Document Number MNT-ESQ-PL1 REV 6
Date: June 8, 1992	Sheet 1 of 13

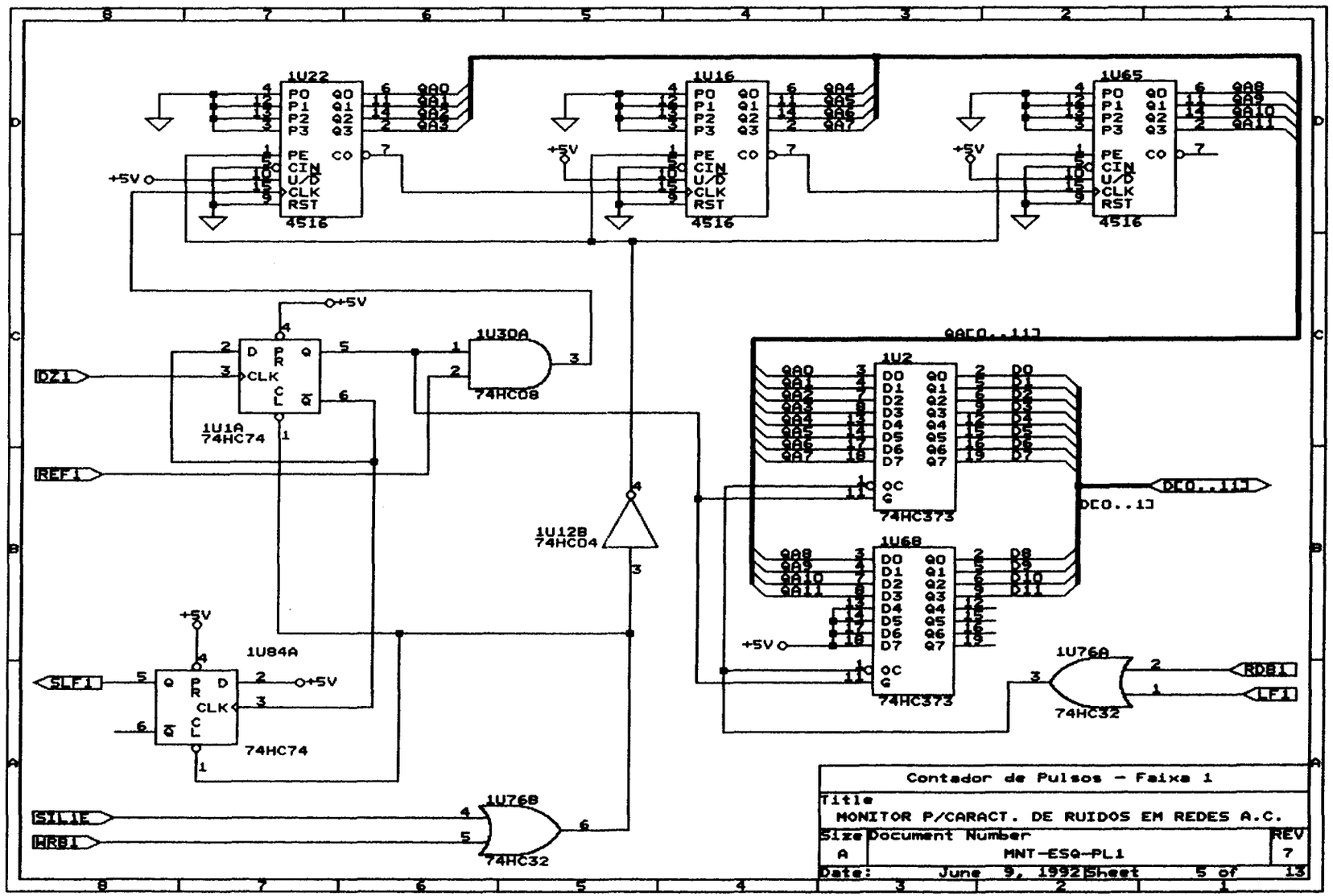


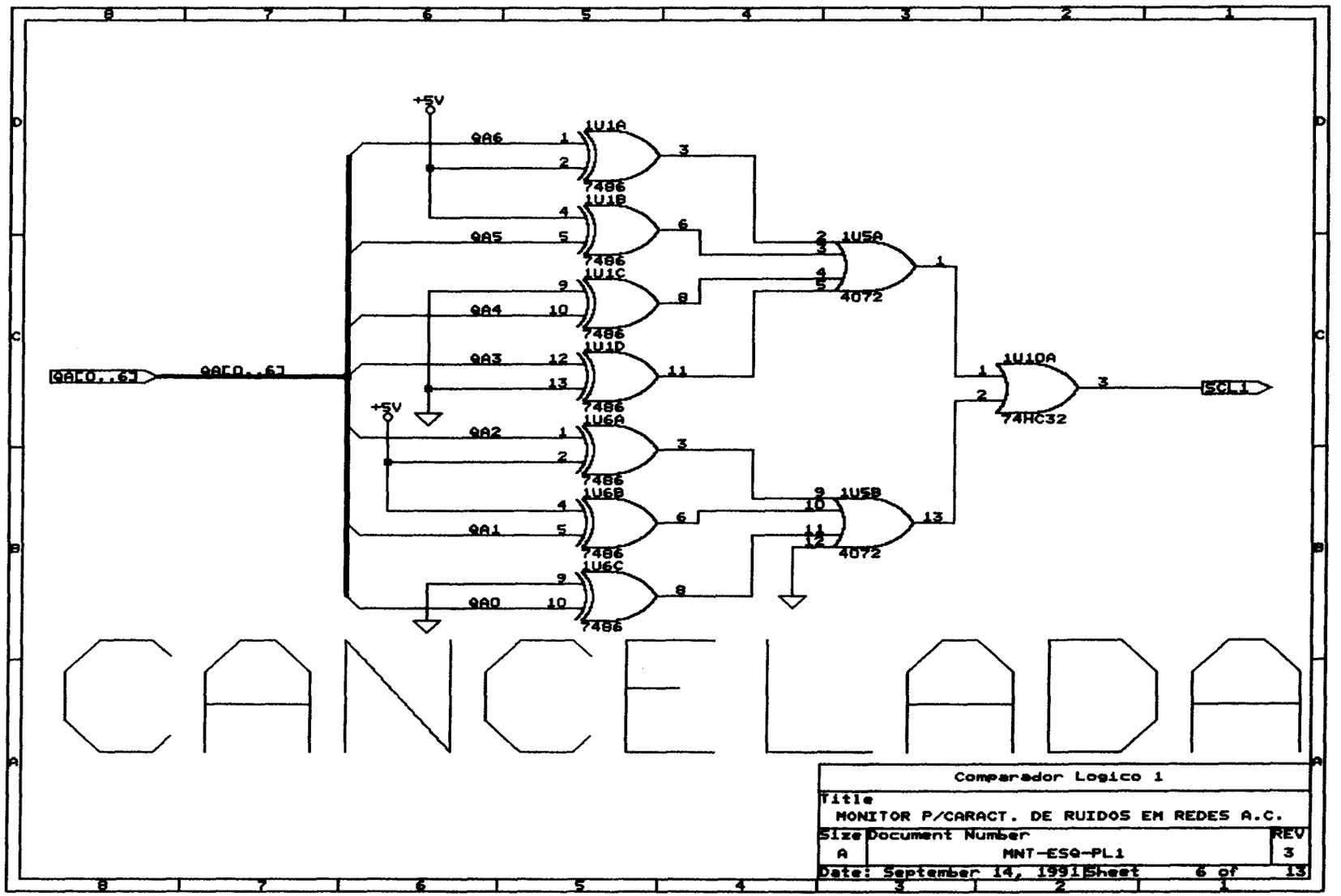
SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA



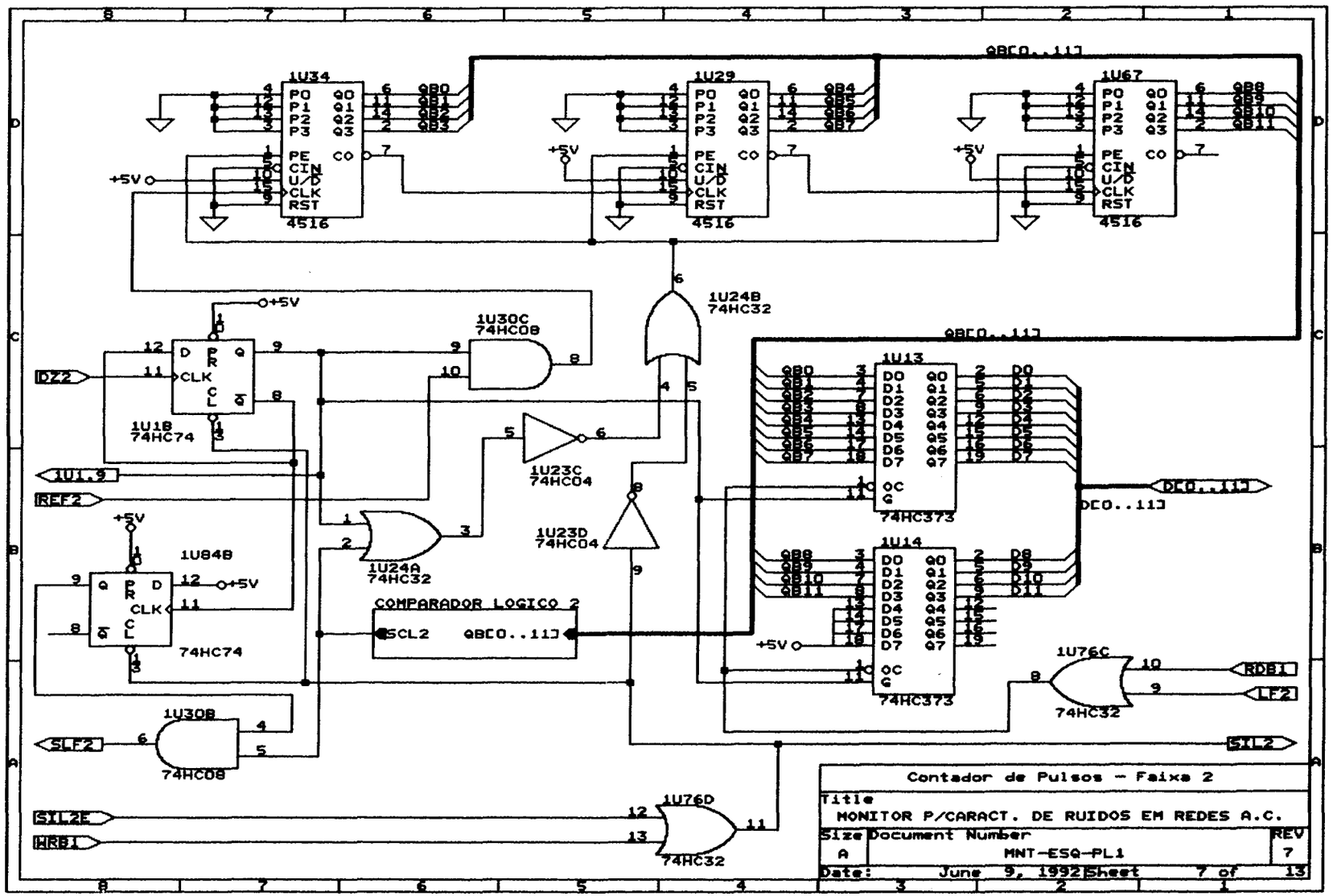
FILTRO - FAIXA 3	
Title MONITOR P/CARACT. DE RUIDOS EM REDES A.C.	
Size A	Document Number MNT-ESQ-PL1 REV 6
Date: June 8, 1992	Sheet 3 of 13

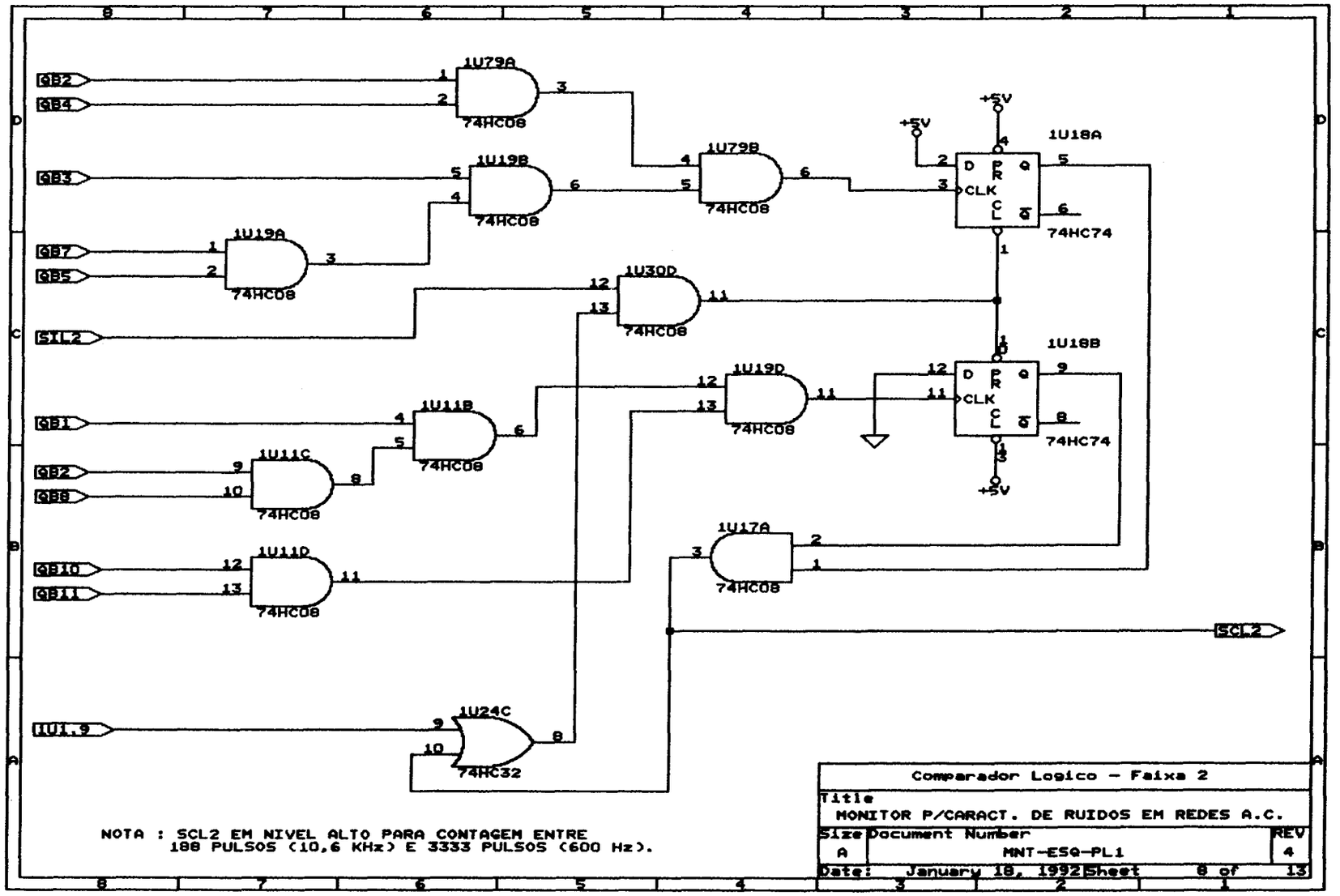

PCLK
+5V
1U42B
D
PR
Q
CLK
CL
Q̄
74HC74
1U49A
CP0
CP1
MR
Q0
Q1
Q2
Q3
4520
1U58A
4518
1U40A
1U41A
REF2
REF1
BT4
BT3
GERADORES DE BASES DE TEMPO
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL1
REV
3
Date: September 15, 1991
Sheet 4 of 13


1U22
4516
1U16
4516
1U65
4516
+5V
QA[0..11]
1U2
74HC373
1U68
74HC373
D[0..11]
D[0..1]
1U30A
74HC08
1U1A
74HC74
DZ1
REF1
1U12B
74HC04
1U84A
74HC74
SLF1
1U76A
74HC32
RDB1
LF1
1U76B
74HC32
SIL1E
WRB1
Contador de Pulsos - Faixa 1
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL1
REV
7
Date: June 9, 1992
Sheet 5 of 13



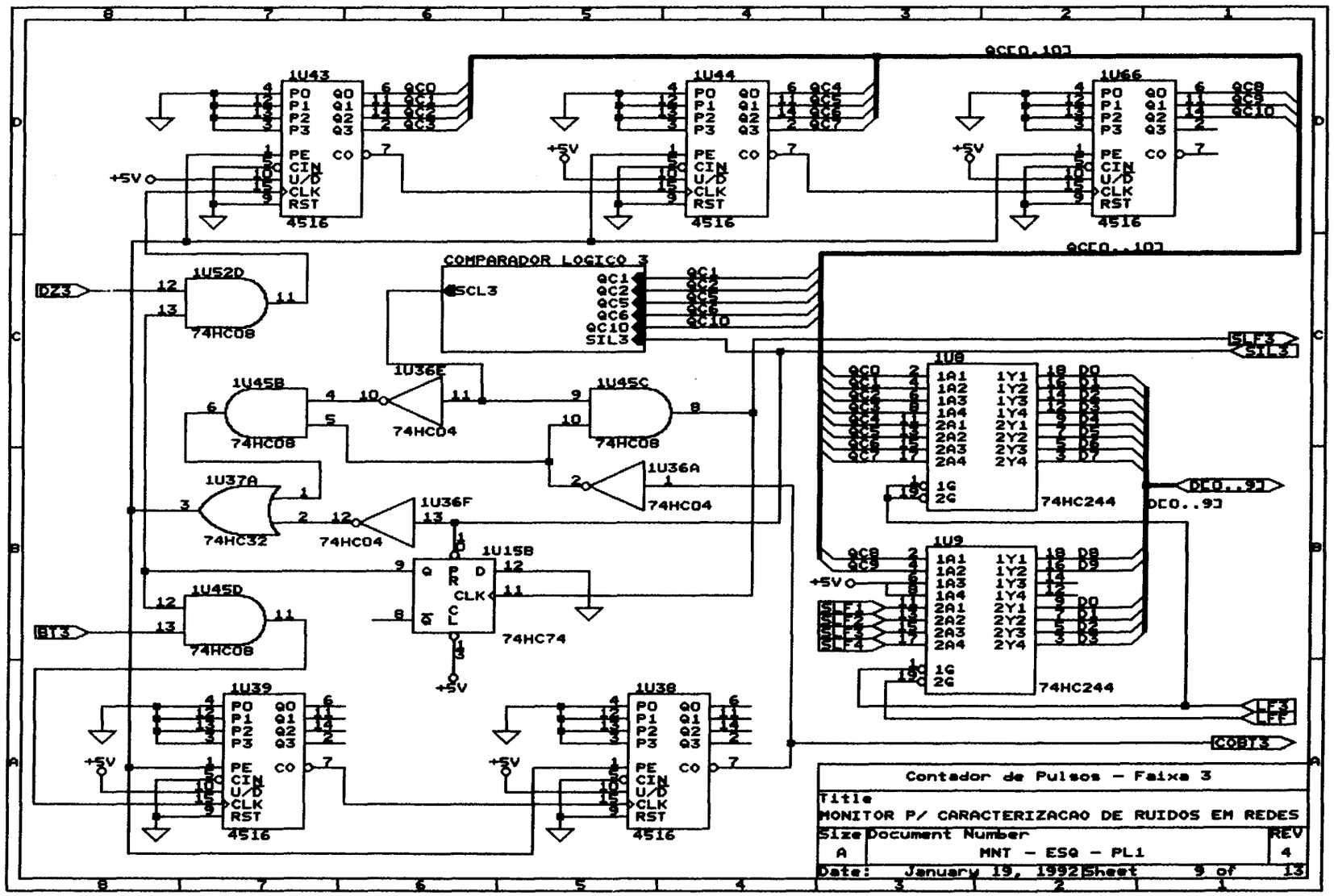
Comparador Logico 1		
Title MONITOR P/CARACT. DE RUIDOS EM REDES A.C.		
Size A	Document Number MNT-ESQ-PL1	REV 3
Date: September 14, 1991	Sheet 6 of 13	

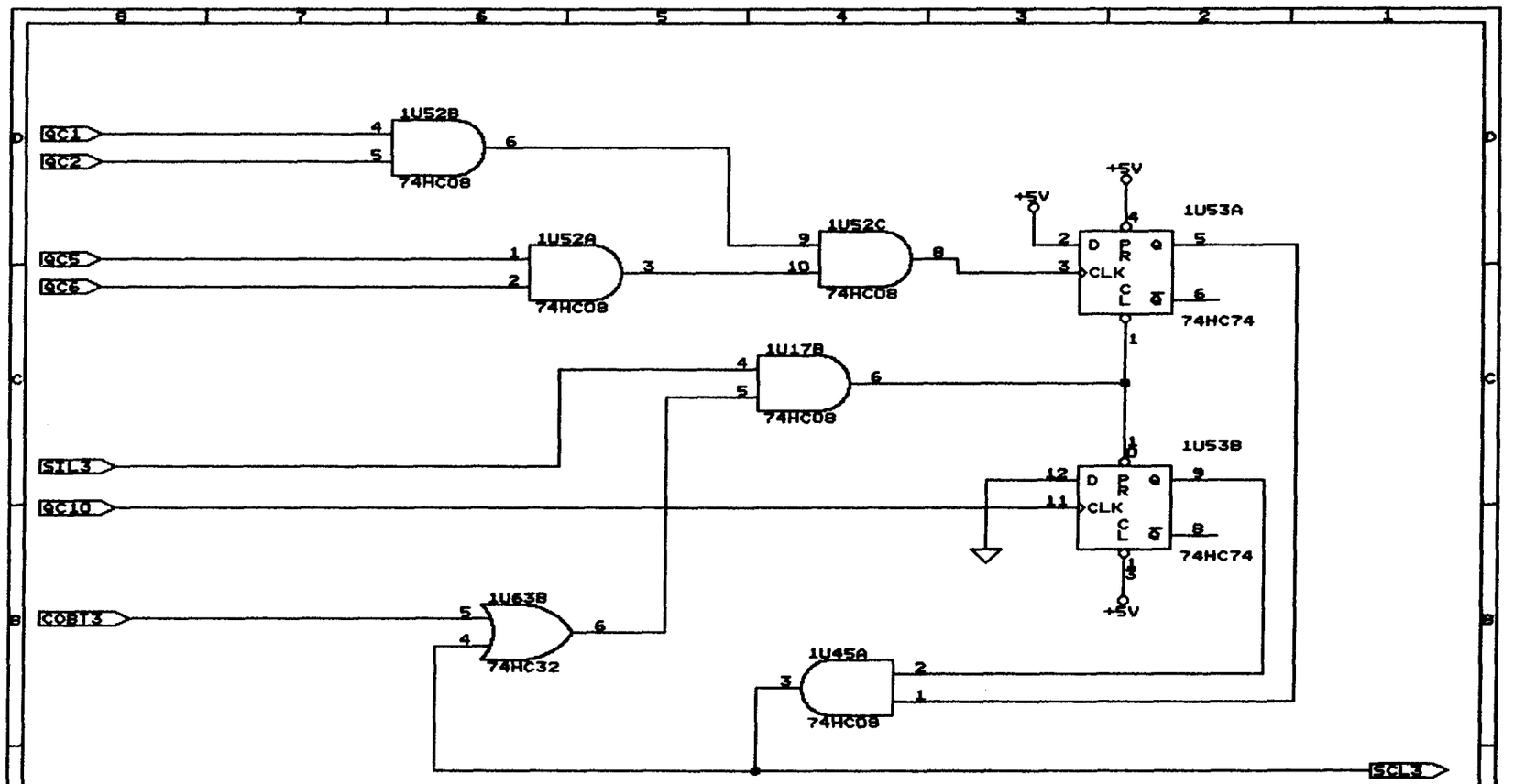

1U34
4516
1U29
4516
1U67
4516
QB[0..11]
+5V
1U24B
74HC32
1U30C
74HC08
1U18
74HC74
DZ2
1U1.9
REF2
1U23C
74HC04
1U23D
74HC04
1U24A
74HC32
1U84B
74HC74
COMPARADOR LOGICO 2
SCL2
QB[0..11]
1U13
74HC373
1U14
74HC373
D[0..11]
1U76C
74HC32
RDB1
LF2
SIL2
1U30B
74HC08
SLF2
1U76D
74HC32
SIL2E
HRB1
Contador de Pulsos - Faixa 2
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL1
REV
7
Date: June 9, 1992
Sheet 7 of 13



NOTA : SCL2 EM NIVEL ALTO PARA CONTAGEM ENTRE
188 PULSOS (10,6 KHz) E 3333 PULSOS (600 Hz).

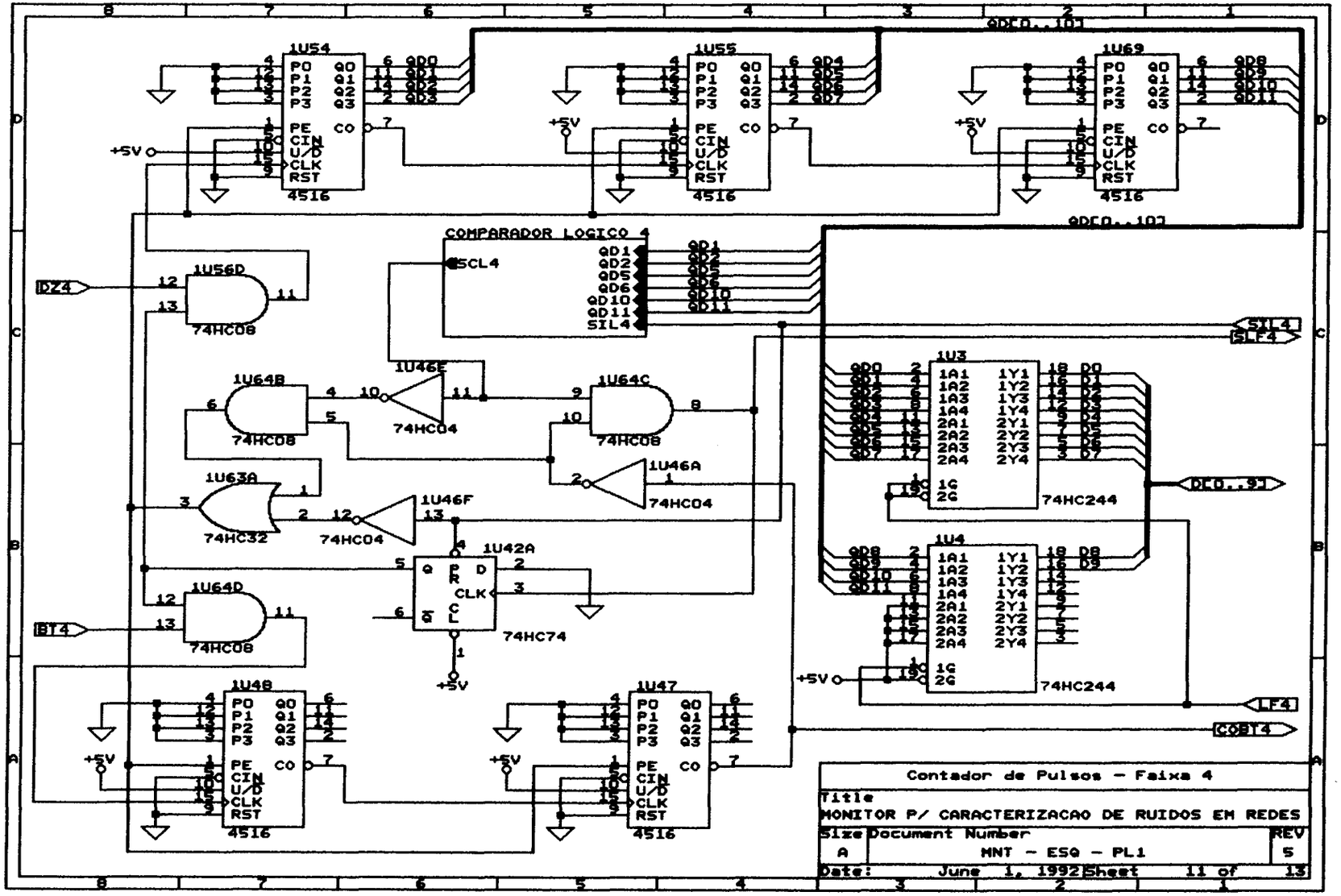
Comparador Logico - Faixa 2		
Title MONITOR P/CARACT. DE RUIDOS EM REDES A.C.		
Size A	Document Number MNT-ESQ-PL1	REV 4
Date: January 18, 1992	Sheet 8 of 13	


1U43
1U44
1U66
4516
QC[0..10]
COMPARADOR LOGICO 3
SCL3
SIL3
SLF3
1U52D
74HC08
DZ3
1U45B
1U45C
1U45D
BI3
1U36E
1U36A
1U36F
74HC04
1U37A
74HC32
1U15B
74HC74
1U8
1U9
74HC244
D[0..9]
LF3
COBI3
1U39
1U38
+5V
Contador de Pulsos - Faixa 3
Title
MONITOR P/ CARACTERIZACAO DE RUIDOS EM REDES
Size
A
Document Number
MNT - ESQ - PL1
REV
4
Date: January 19, 1992
Sheet 9 of 13

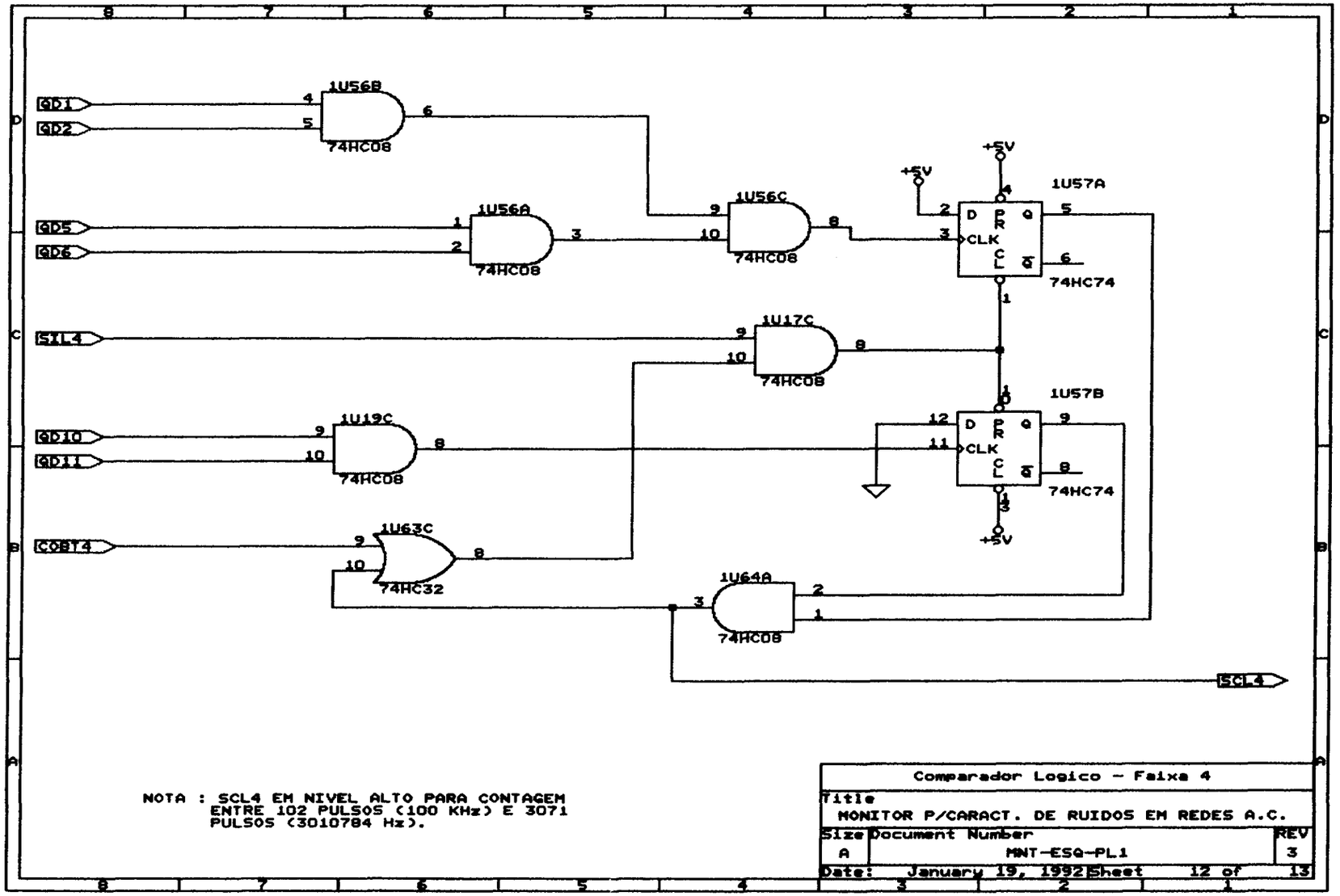


NOTA : SCL3 EM NIVEL ALTO PARA CONTAGEM ENTRE 102 PULSOS (10 KHz) E 1023 PULSOS (100294 Hz).

Comparador Logico - Faixa 3		
Title MONITOR P/CARACT. DE RUIDOS EM REDES A.C.		
Size A	Document Number MNT-ESQ-PL1	REV 3
Date: January 19, 1992	Sheet 10 of 13	

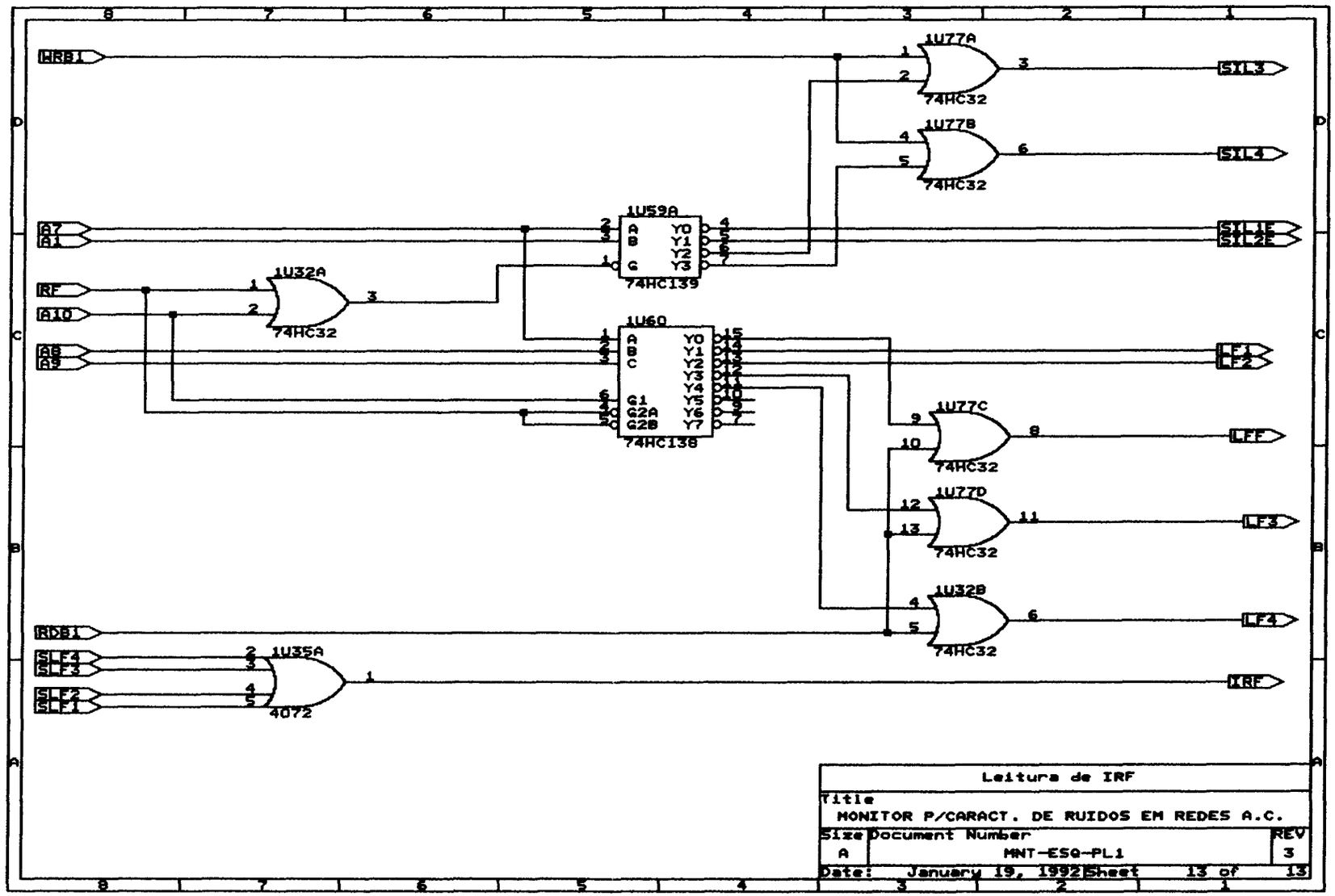


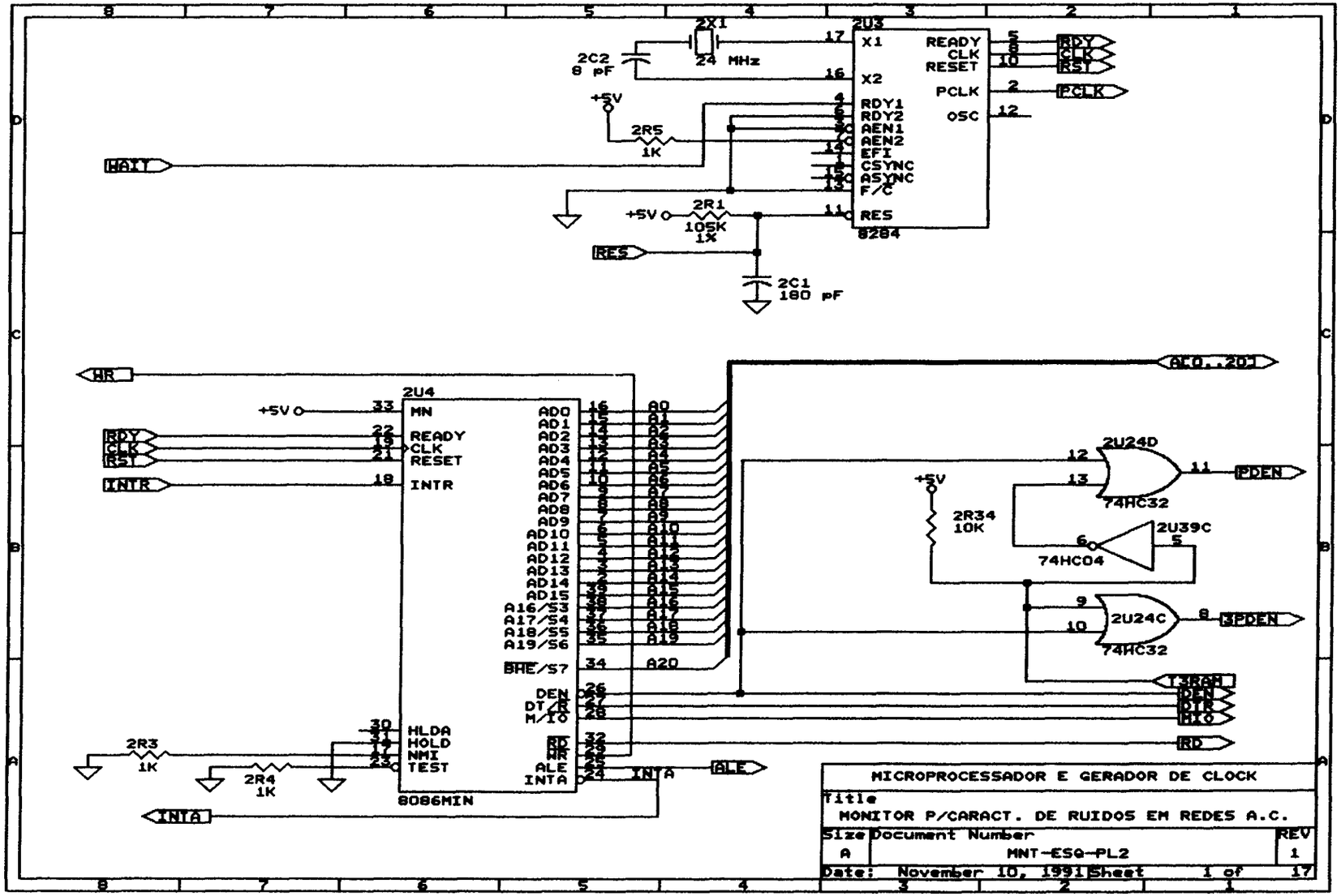
1U54
1U55
1U69
4516
COMPARADOR LOGICO 4
SCL4
QD1
QD2
QD5
QD6
QD10
QD11
SIL4
1U56D
74HC08
DZ4
1U64B
1U46E
74HC04
1U64C
1U46A
1U63A
74HC32
1U46F
1U42A
74HC74
1U64D
BT4
1U48
1U47
1U3
1U4
74HC244
D[0..9]
SIL4
SLF4
LF4
COBT4
+5V
QD[0..10]
Contador de Pulsos - Faixa 4
Title
MONITOR P/ CARACTERIZACAO DE RUIDOS EM REDES
Size
A
Document Number
MNT - ESQ - PL1
REV
5
Date: June 1, 1992
Sheet 11 of 13

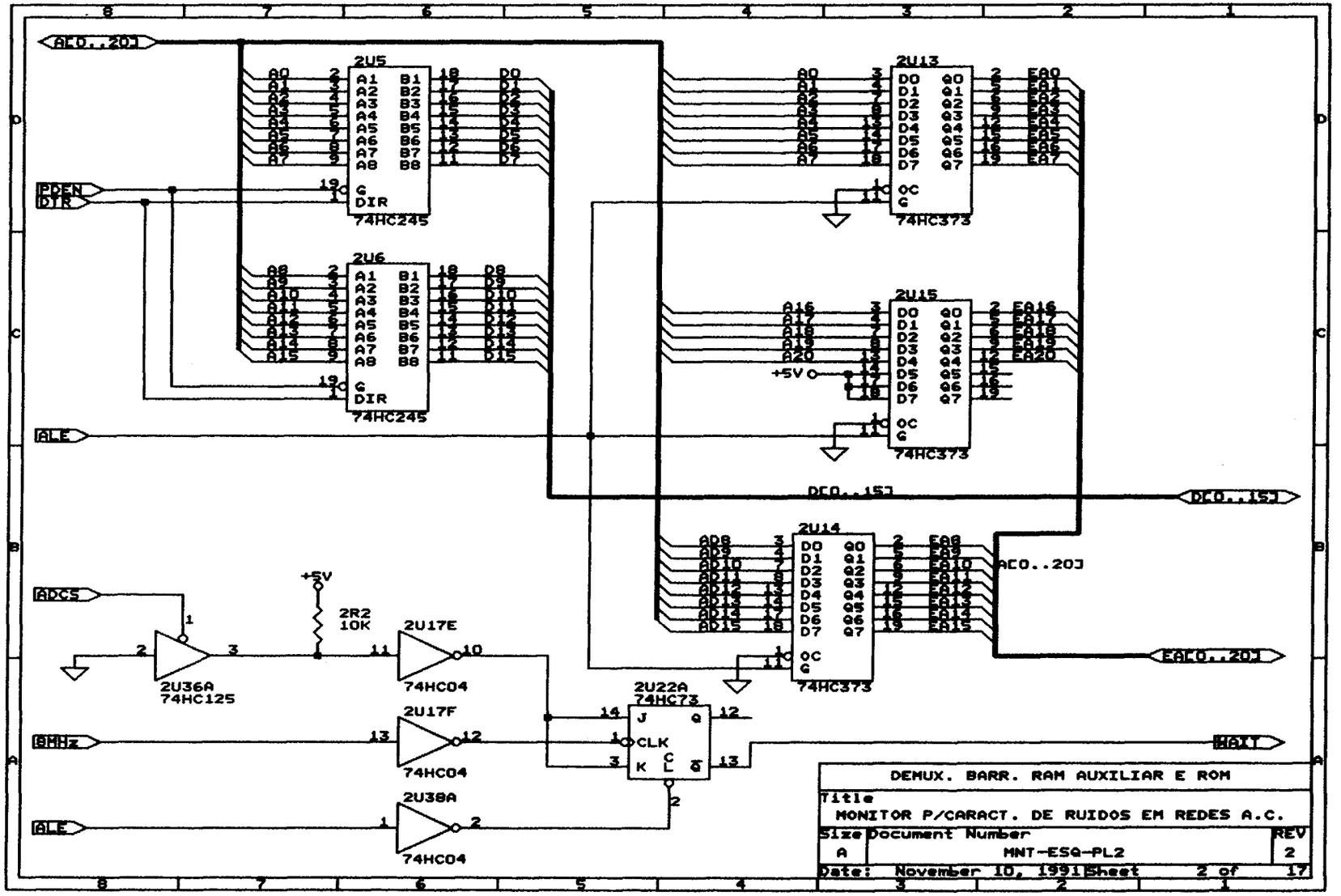


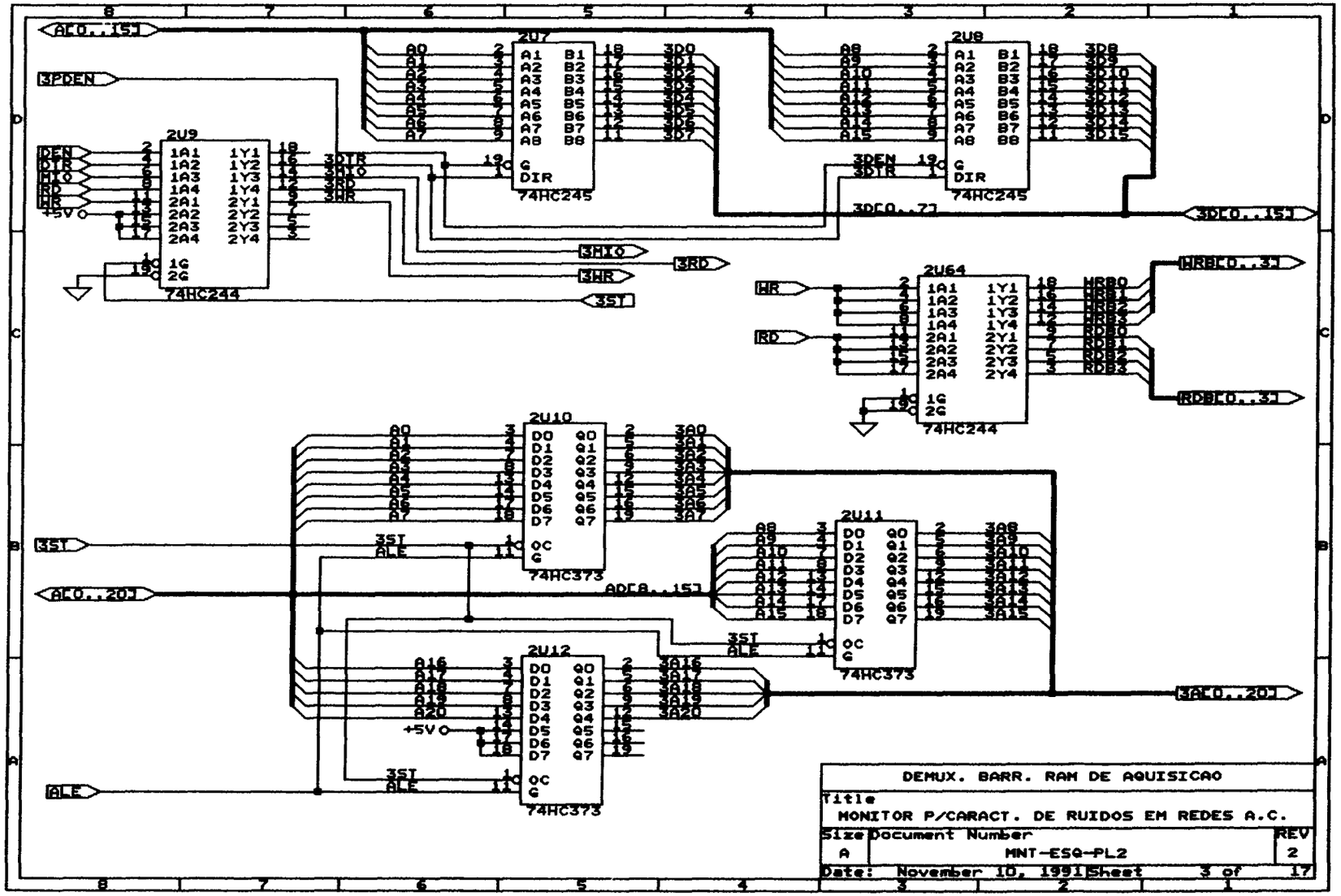
NOTA : SCL4 EM NIVEL ALTO PARA CONTAGEM
ENTRE 102 PULSOS (100 KHz) E 3071
PULSOS (3010784 Hz).

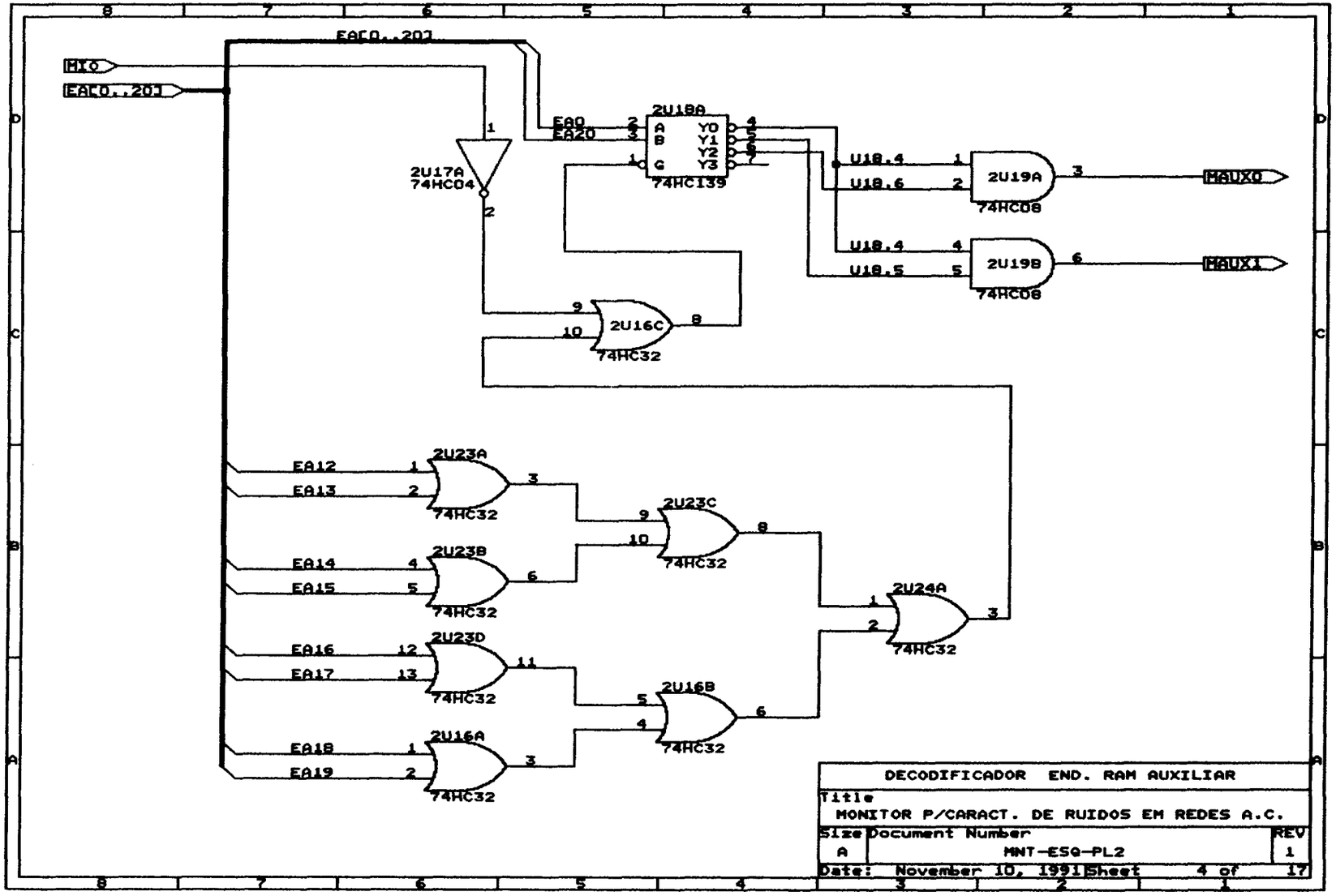
Comparador Logico - Faixa 4		
Title MONITOR P/CARACT. DE RUIDOS EM REDES A.C.		
Size A	Document Number MNT-ESQ-PL1	REV 3
Date: January 19, 1992	Sheet 12 of 13	

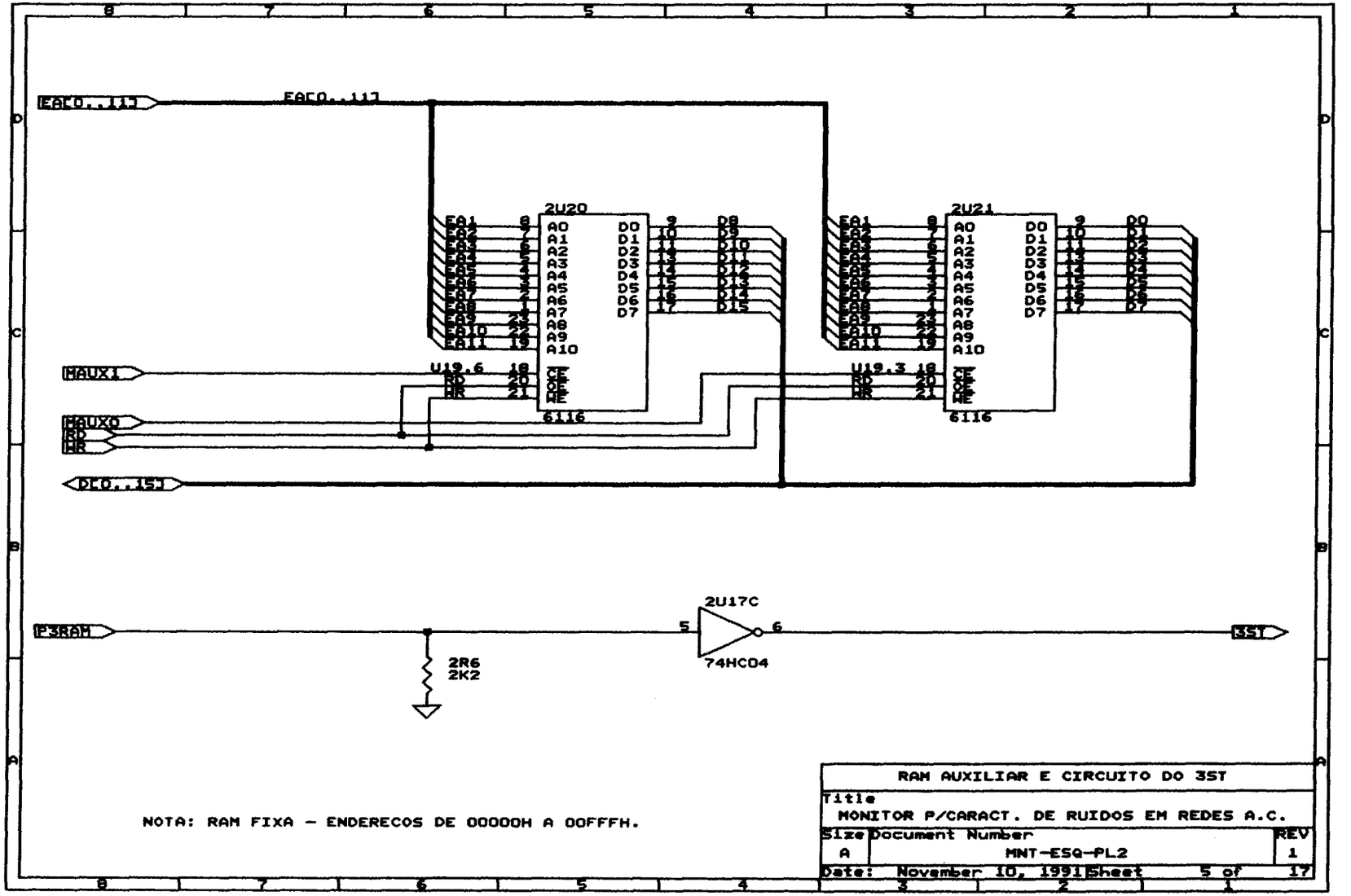

WRB1
A7
A1
RF
A10
A8
A9
RDB1
SLF4
SLF3
SLF2
SLF1
1U32A
74HC32
1U59A
74HC139
1U60
74HC138
1U77A
74HC32
1U77B
74HC32
1U77C
74HC32
1U77D
74HC32
1U32B
74HC32
1U35A
4072
SIL3
SIL4
SIL1E
SIL2E
LF1
LF2
LFF
LF3
LF4
IRF
Leitura de IRF
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL1
REV
3
Date: January 19, 1992
Sheet 13 of 13


2U3
8284
X1
X2
RDY1
RDY2
AEN1
AEN2
EFI
CSYNC
ASYNC
F/C
RES
READY
CLK
RESET
PCLK
OSC
2X1
24 MHz
2C2
8 pF
2R5
1K
2R1
105K
1%
2C1
180 pF
+5V
WAIT
RES
RDY
CLK
RST
PCLK
2U4
8086MIN
MN
READY
CLK
RESET
INTR
HLDA
HOLD
NMI
TEST
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
AD8
AD9
AD10
AD11
AD12
AD13
AD14
AD15
A16/S3
A17/S4
A18/S5
A19/S6
BHE/S7
DEN
DT/R
M/IO
RD
WR
ALE
INTA
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
2R3
1K
2R4
1K
HR
INTR
INTA
ALE
A[0..20]
2U24D
74HC32
2U39C
74HC04
2U24C
74HC32
2R34
10K
PDEN
3PDEN
T3RAM
DEN
DTR
MIO
RD
MICROPROCESSADOR E GERADOR DE CLOCK
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL2
REV
1
Date: November 10, 1991
Sheet 1 of 17


AC0..20]
PDEN
DIR
ALE
2U5
74HC245
2U6
74HC245
2U13
74HC373
2U15
74HC373
2U14
74HC373
+5V
DC0..15]
EAC0..20]
ADCS
2U36A
74HC125
2R2
10K
2U17E
74HC04
2U17F
74HC04
2U38A
74HC04
BMHz
2U22A
74HC73
WAIT
DEMUX. BARR. RAM AUXILIAR E ROM
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL2
REV
2
Date: November 10, 1991
Sheet 2 of 17

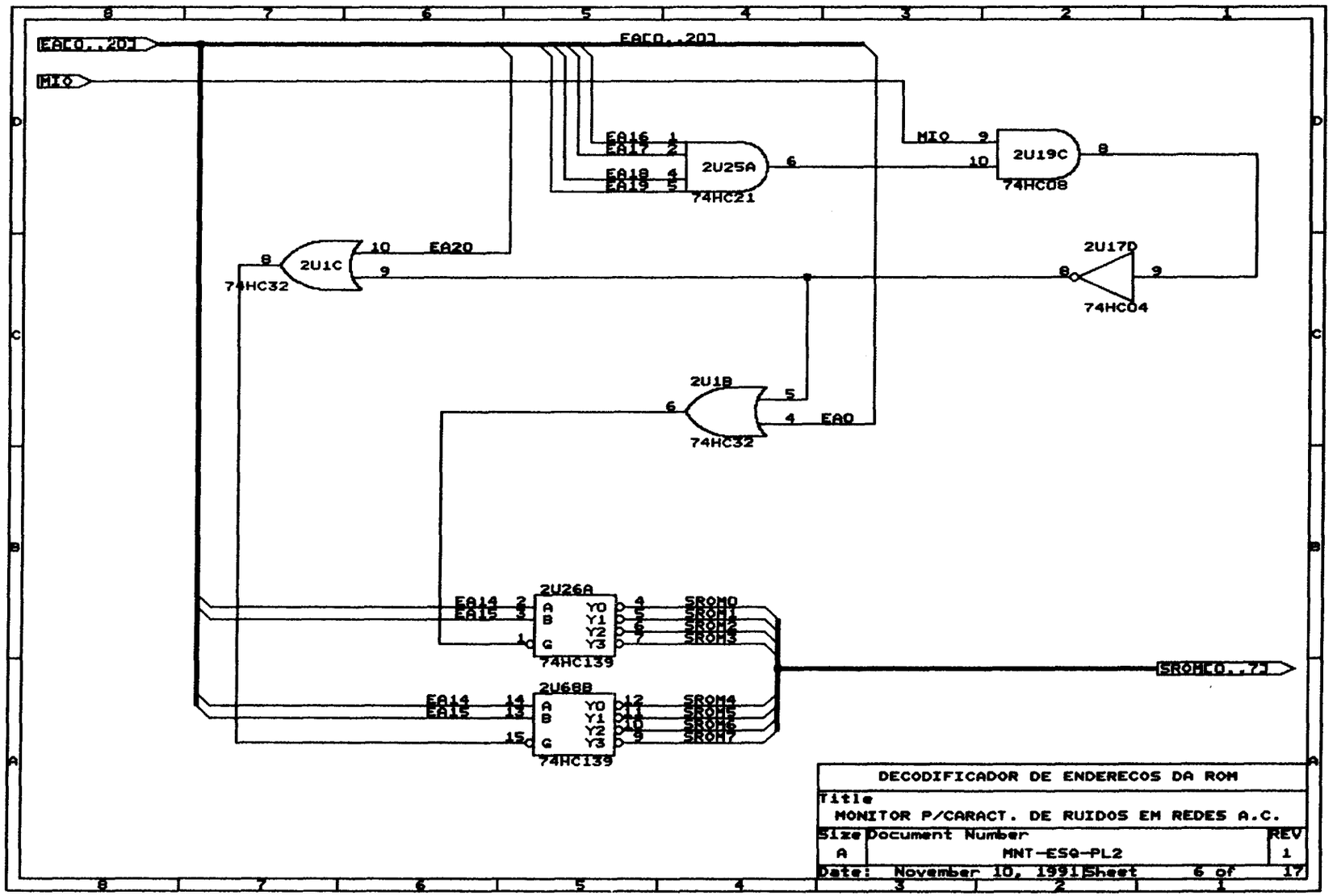

AD[0..15]
3PDEN
2U7
74HC245
2U8
74HC245
3D[0..7]
3D[0..15]
2U9
74HC244
DEN
DTR
MIO
RD
WR
+5V
3DTR
3MIO
3RD
3WR
3ST
2U64
74HC244
WR
RD
WRB[0..3]
RDB[0..3]
2U10
74HC373
2U11
74HC373
2U12
74HC373
3ST
ALE
AD[8..15]
AD[0..20]
3A[0..20]
ALE
DEMUX. BARR. RAM DE AQUISICAO
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL2
REV
2
Date: November 10, 1991
Sheet 3 of 17


EA[0..20]
MIO
EA[0..20]
2U17A
74HC04
2U18A
74HC139
EA0
EA20
U18.4
U18.6
2U19A
74HC08
MAUX0
U18.4
U18.5
2U19B
74HC08
MAUX1
2U16C
74HC32
EA12
EA13
2U23A
74HC32
EA14
EA15
2U23B
74HC32
EA16
EA17
2U23D
74HC32
EA18
EA19
2U16A
74HC32
2U23C
74HC32
2U16B
74HC32
2U24A
74HC32
DECODIFICADOR END. RAM AUXILIAR
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL2
REV
1
Date: November 10, 1991
Sheet 4 of 17

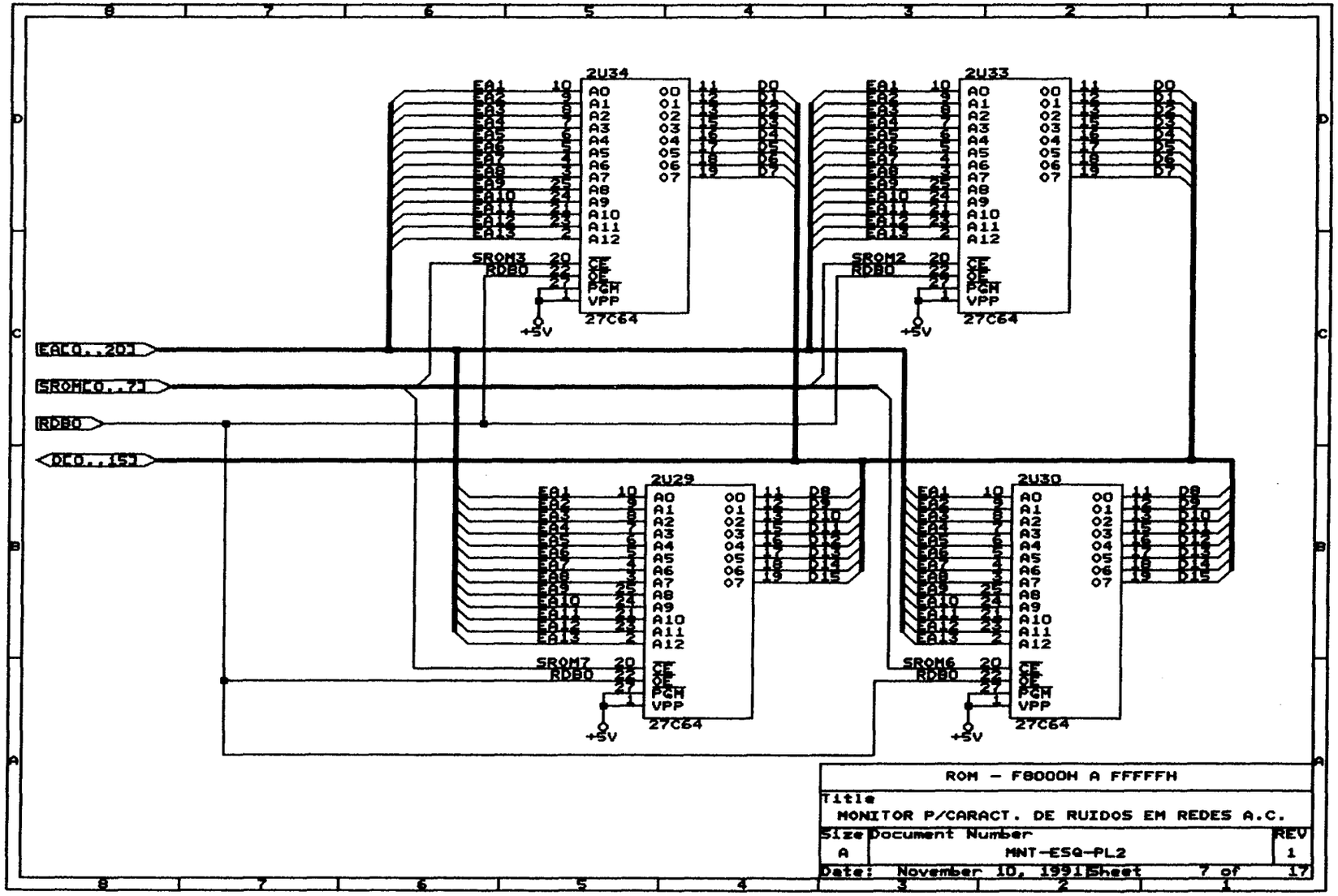


NOTA: RAM FIXA - ENDERECOS DE 00000H A 00FFFH.

RAM AUXILIAR E CIRCUITO DO 3ST		
Title MONITOR P/CARACT. DE RUIDOS EM REDES A.C.		
Size A	Document Number MNT-ESQ-PL2	REV 1
Date: November 10, 1991	Sheet 5 of 17	



SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

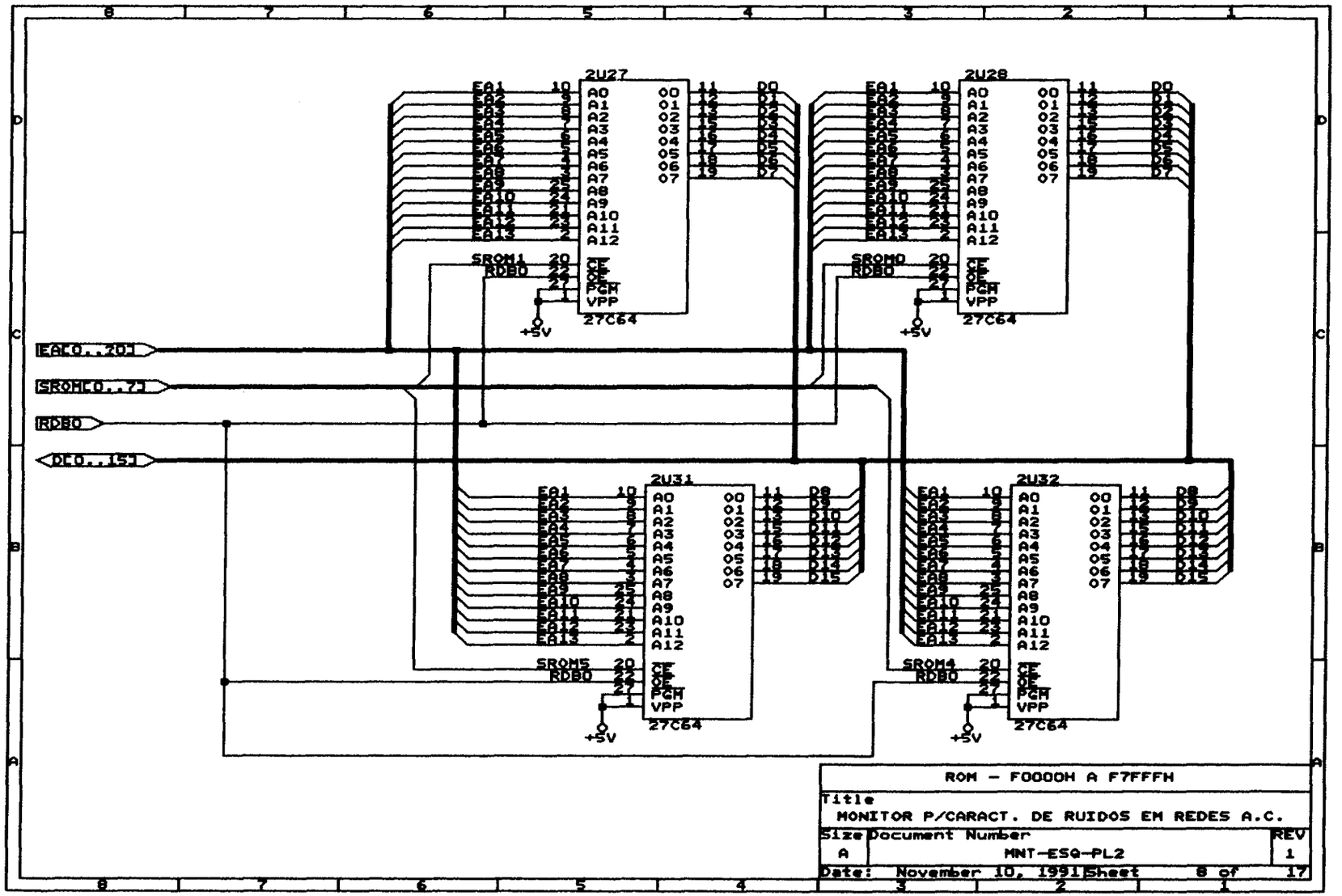


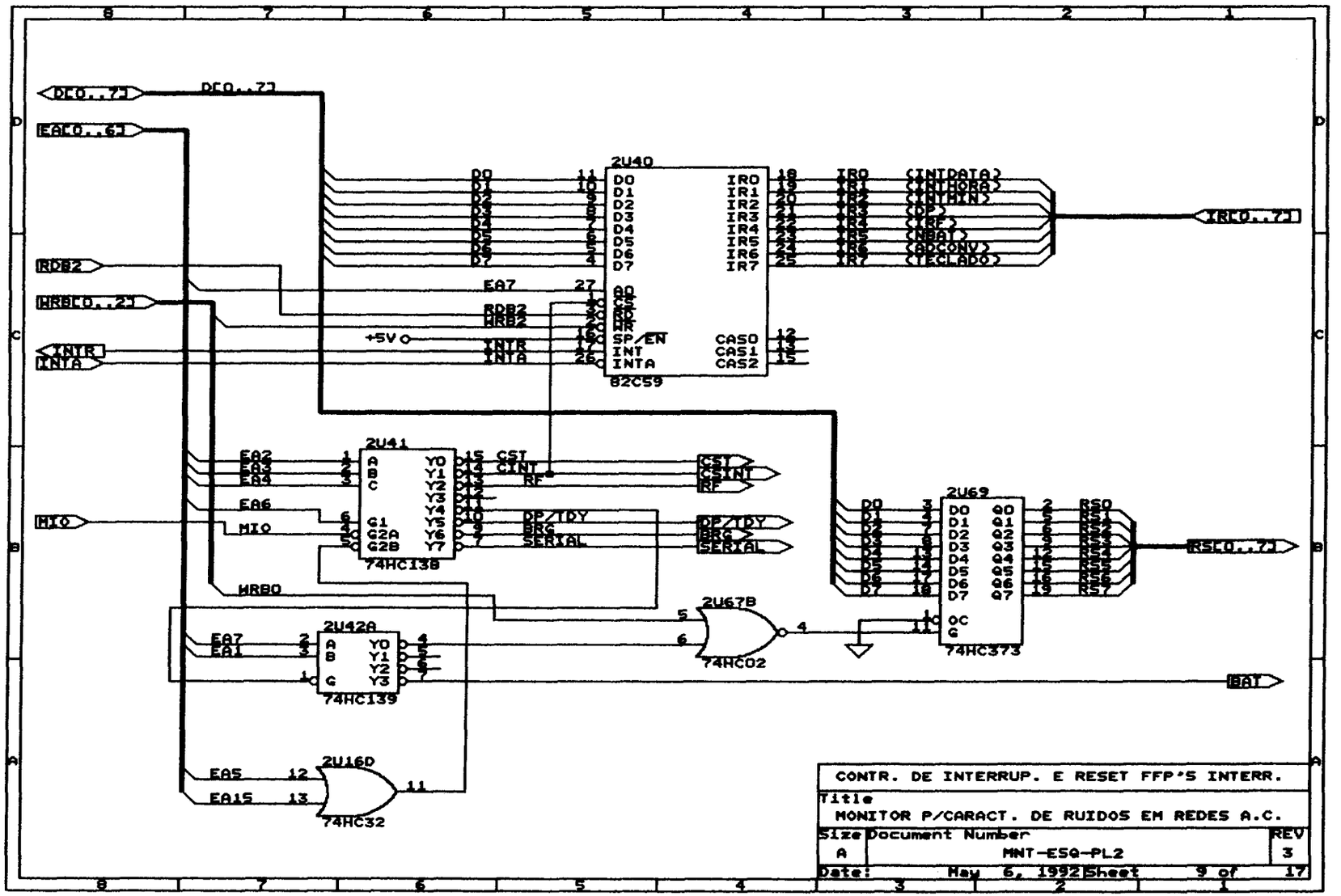
ROM - F8000H A FFFFFH

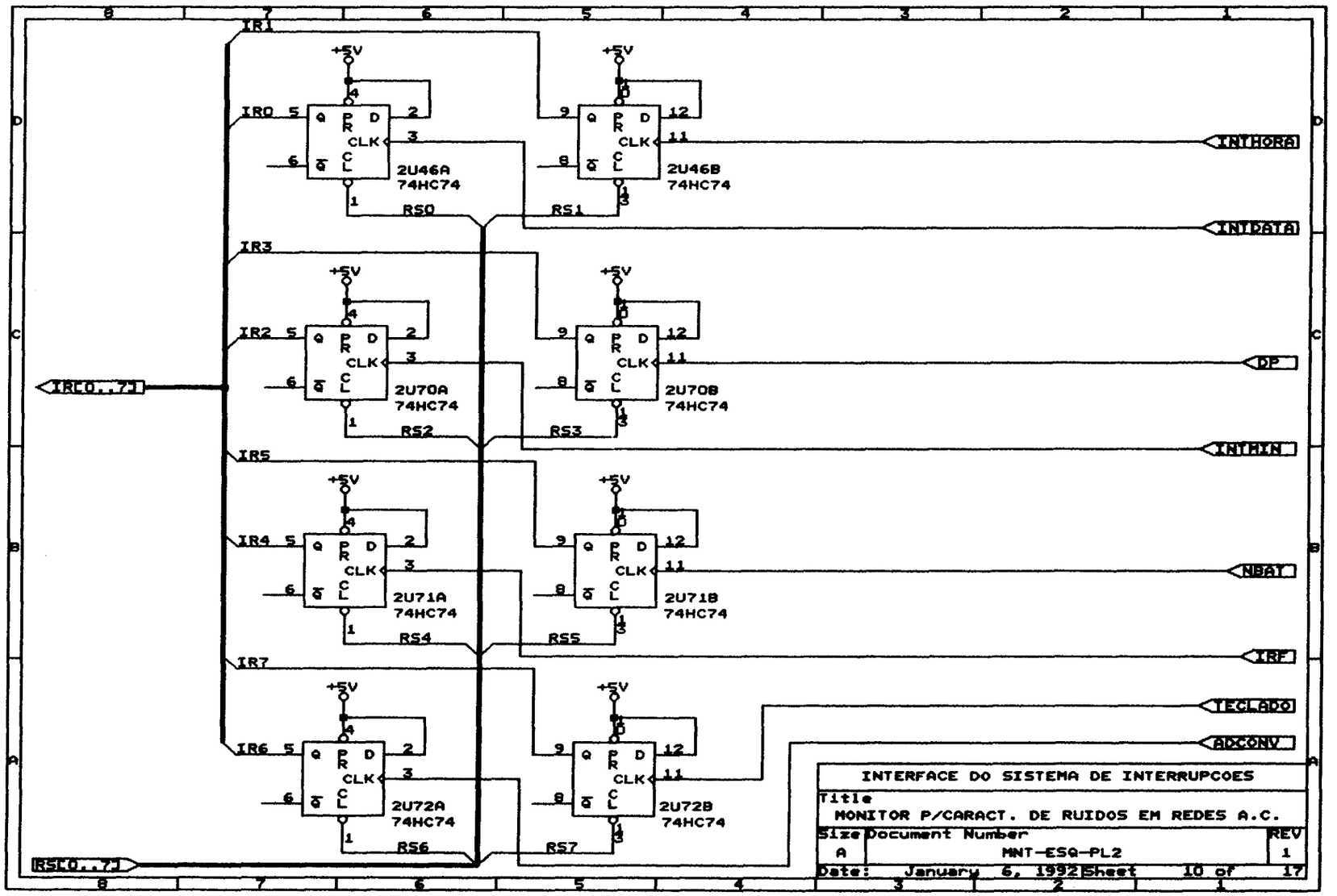
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.

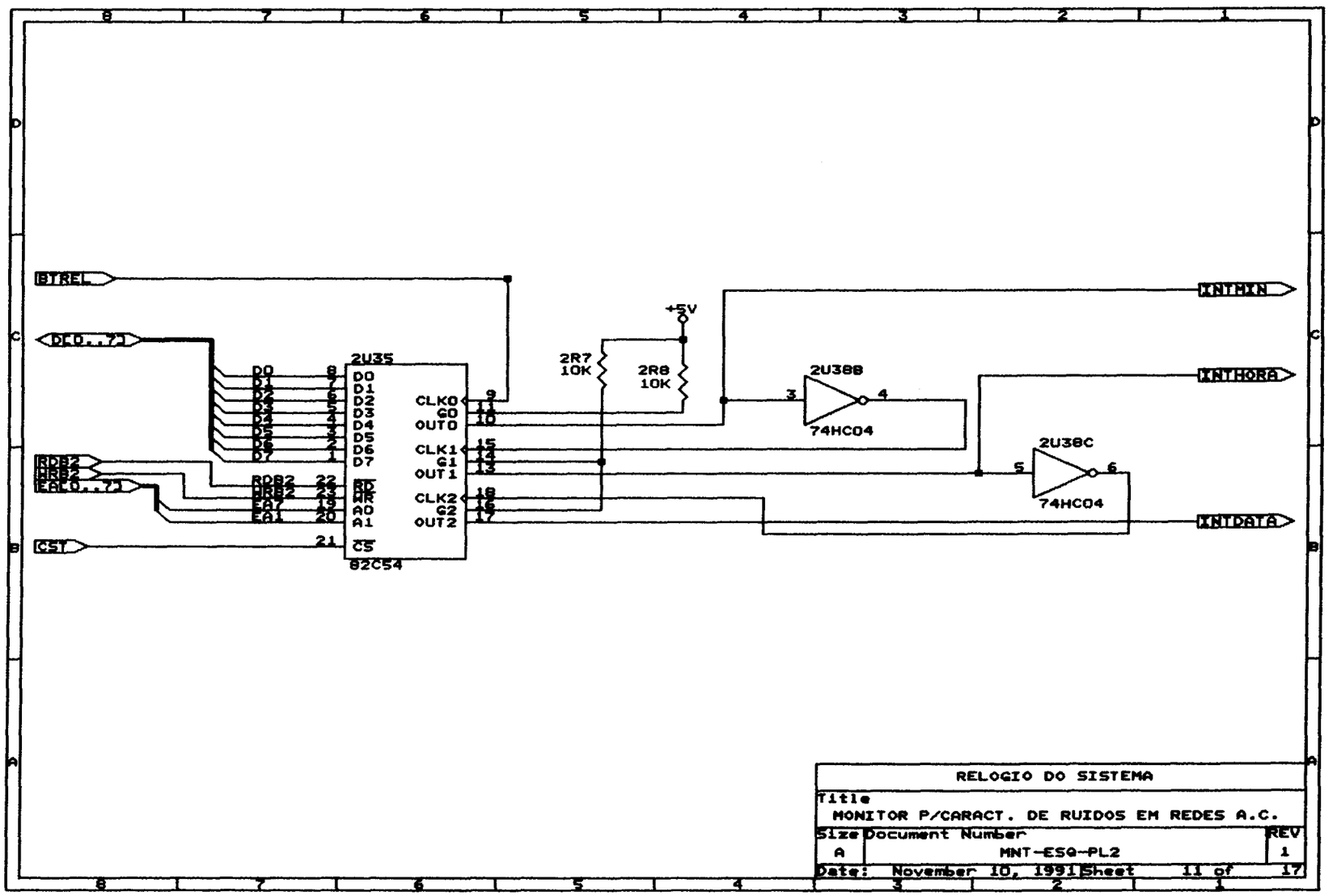
Size	Document Number	REV
A	MNT-ESQ-PL2	1

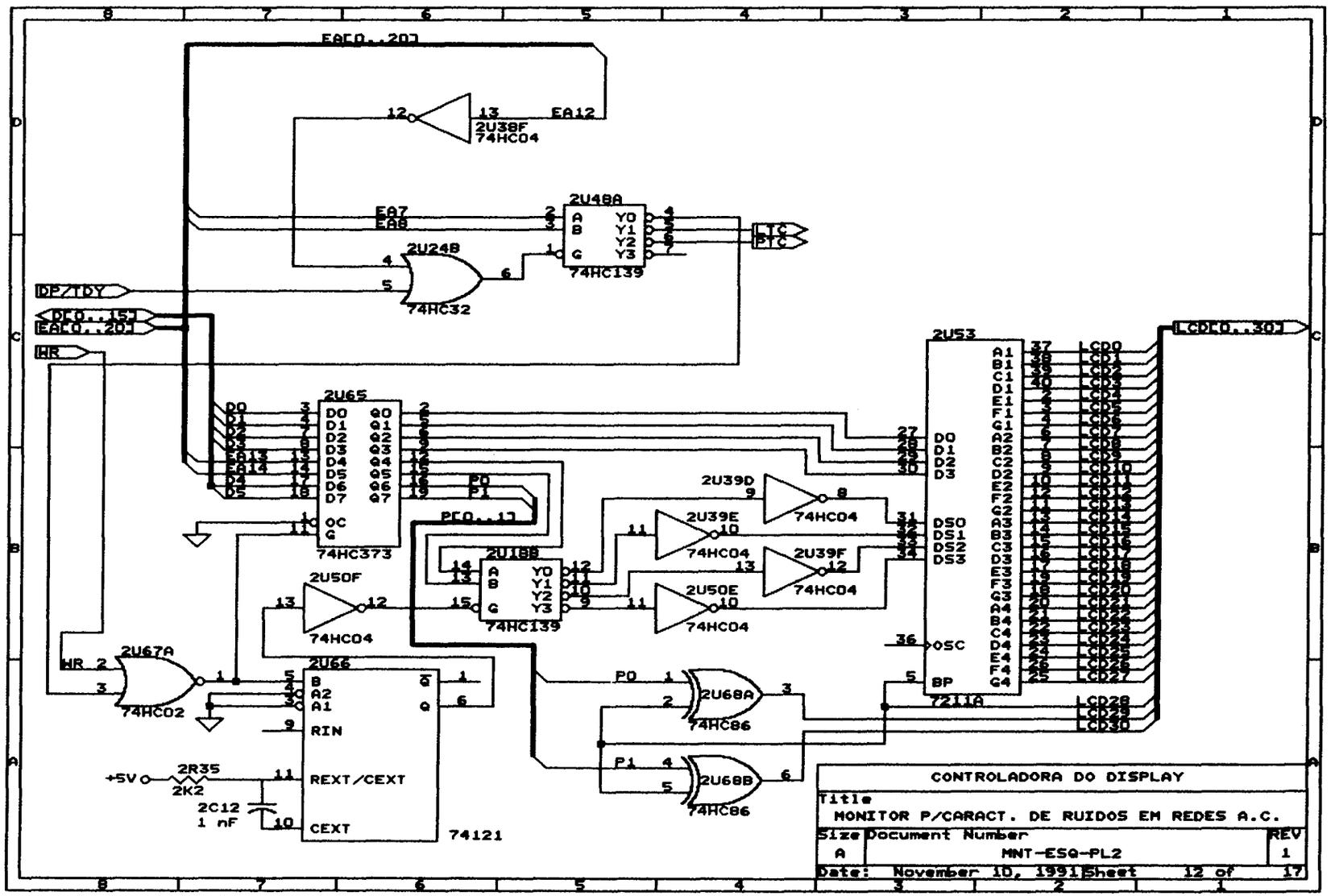
Date: November 10, 1991 Sheet 7 of 17

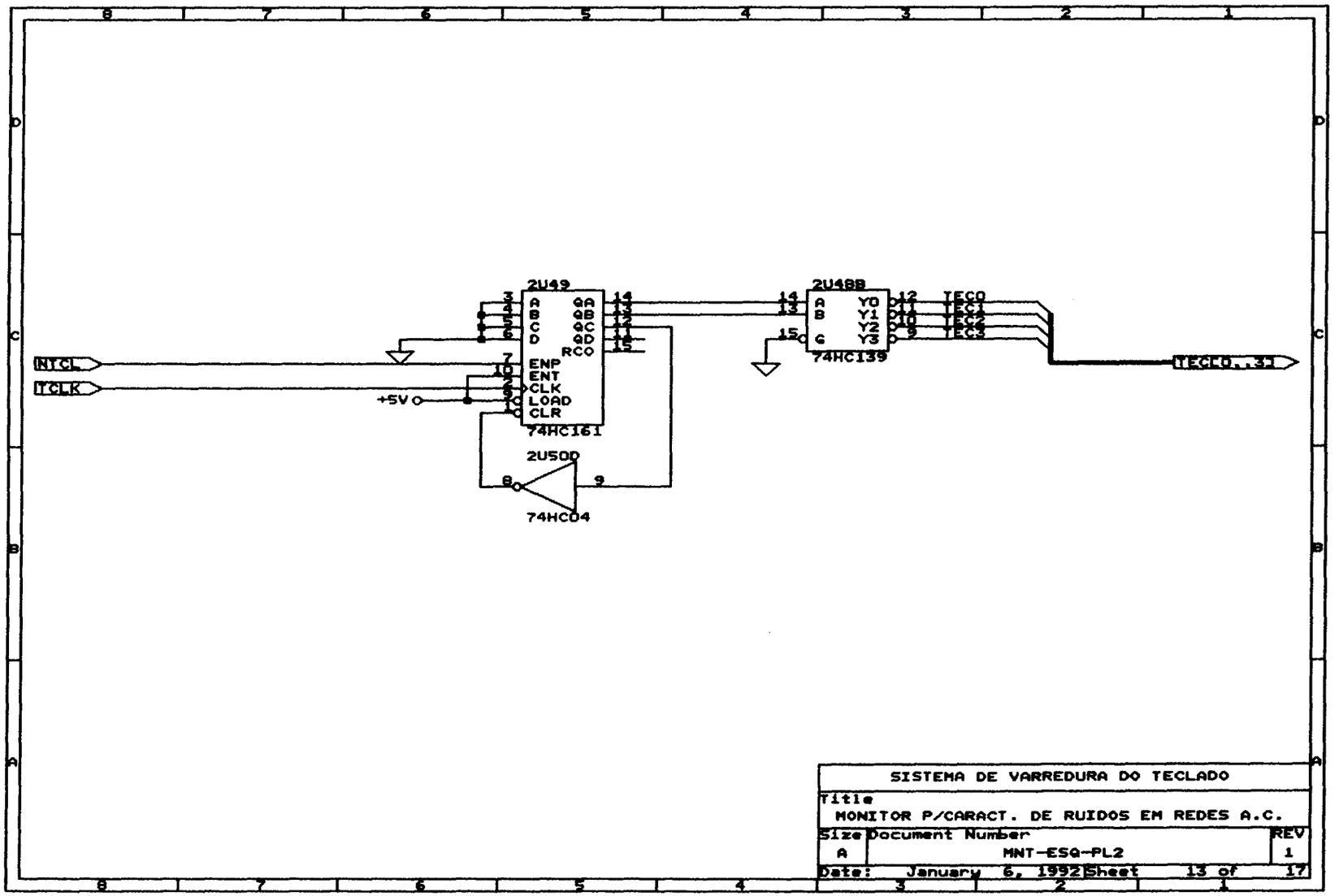

2U27
2U28
2U31
2U32
27C64
SROM1
SROM0
SROM5
SROM4
RDB0
+5V
EA[0..20]
SROM[0..7]
RDB0
D[0..15]
ROM - F0000H A F7FFFH
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL2
REV
1
Date: November 10, 1991
Sheet 8 of 17


DC[0..7]
EA[0..6]
RDB2
WRB[0..2]
INTR
INTA
MIO
+5V
2U40
82C59
D0 D1 D2 D3 D4 D5 D6 D7
A0 CS RD WR
SP/EN INT INTA
IR0 IR1 IR2 IR3 IR4 IR5 IR6 IR7
CAS0 CAS1 CAS2
EA7
RDB2
WRB2
INTR
INTA
IR[0..7]
{INTDATA}
{INTHORA}
{INTMIN}
{DP}
{RF}
{NBAT}
{ADCONV}
{TECLADO}
2U41
74HC138
A B C G1 G2A G2B
Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
EA2 EA3 EA4 EA6
CST CINT RF
DP/TDY BRG SERIAL
2U69
74HC373
D0 D1 D2 D3 D4 D5 D6 D7
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7
OC G
RS[0..7]
RS0 RS1 RS2 RS3 RS4 RS5 RS6 RS7
WRB0
2U67B
74HC02
2U42A
74HC139
A B G
Y0 Y1 Y2 Y3
EA7 EA1
BAT
2U16D
74HC32
EA5 EA15
CONTR. DE INTERRUP. E RESET FFP'S INTERR.
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size A
Document Number
MNT-ESQ-PL2
REV 3
Date: May 6, 1992
Sheet 9 of 17


IR1
IR0
IR3
IR2
IR5
IR4
IR7
IR6
IRC0..7]
RSC0..7]
+5V
2U46A
74HC74
2U46B
74HC74
2U70A
74HC74
2U70B
74HC74
2U71A
74HC74
2U71B
74HC74
2U72A
74HC74
2U72B
74HC74
RS0
RS1
RS2
RS3
RS4
RS5
RS6
RS7
INTHORA
INTDATA
DP
INTHIN
NBAT
IRF
TECLADO
ADCONV
INTERFACE DO SISTEMA DE INTERRUPCOES
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL2
REV
1
Date: January 6, 1992
Sheet 10 of 17


BTREL
D[0..7]
RDB2
WRB2
EA[0..7]
CST
2U35
82C54
D0
D1
D2
D3
D4
D5
D6
D7
RDB2
WRB2
EA7
EA1
RD
WR
A0
A1
CS
CLK0
G0
OUT0
CLK1
G1
OUT1
CLK2
G2
OUT2
2R7
10K
2R8
10K
+5V
2U38B
74HC04
2U38C
74HC04
INTMIN
INTHORA
INTDATA
RELOGIO DO SISTEMA
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL2
REV
1
Date: November 10, 1991
Sheet 11 of 17


EA[0..20]
2U38F
74HC04
EA12
EA7
EA8
2U48A
74HC139
LTC
PTC
2U24B
74HC32
DP/TDY
D[0..15]
EA[0..20]
WR
2U65
74HC373
PD
PI
P[0..1]
2U18B
74HC139
2U50F
74HC04
2U67A
74HC02
2U66
74121
RIN
REXT/CEXT
CEXT
+5V
2R35
2K2
2C12
1 nF
2U39D
2U39E
2U39F
2U50E
2U68A
2U68B
74HC86
2U53
7211A
LCD[0..30]
CONTROLADORA DO DISPLAY
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL2
REV
1
Date: November 10, 1991
Sheet 12 of 17

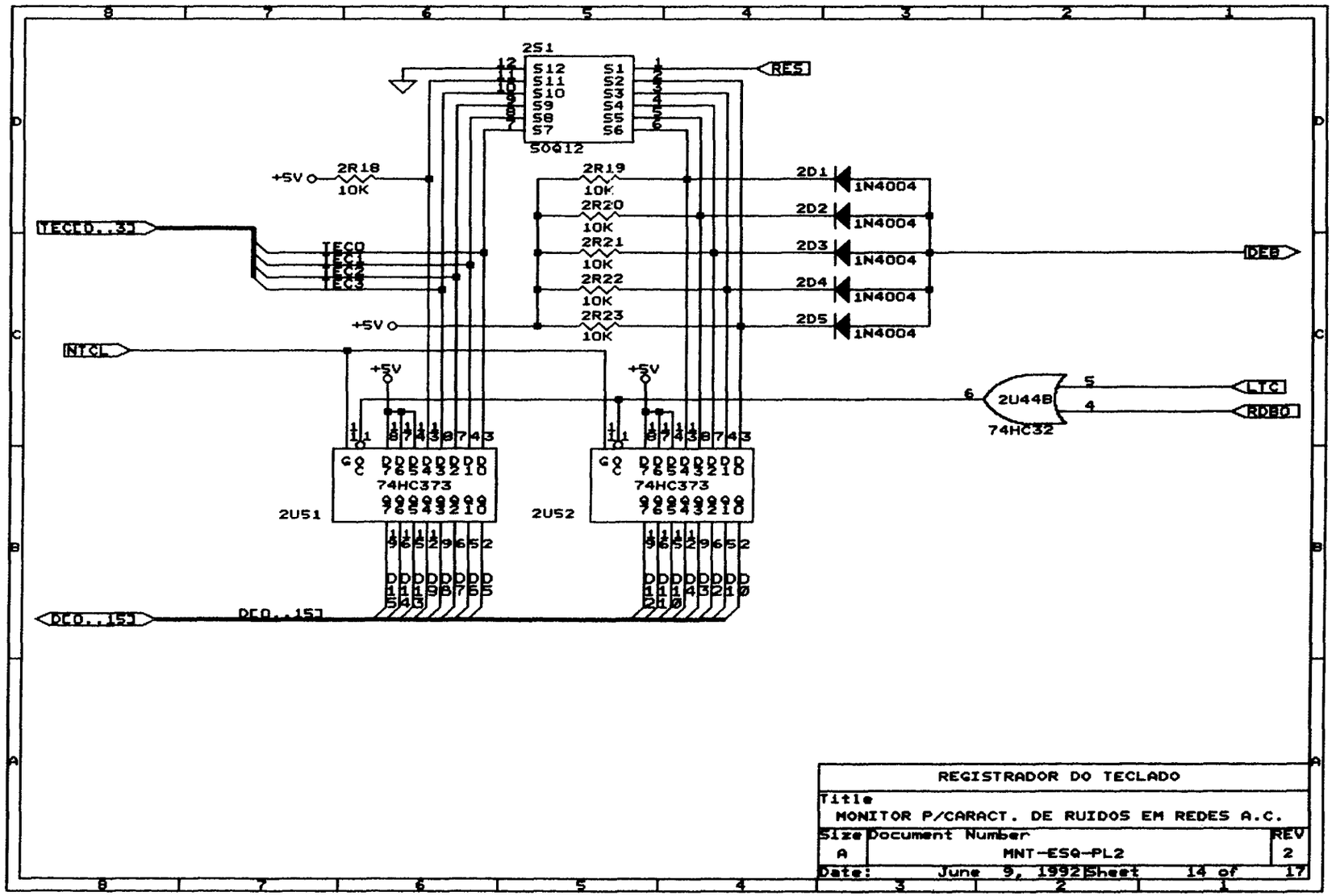


SISTEMA DE VARREDURA DO TECLADO

Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.

Size	Document Number	REV
A	MNT-ESQ-PL2	1

Date: January 6, 1992 Sheet 13 of 17

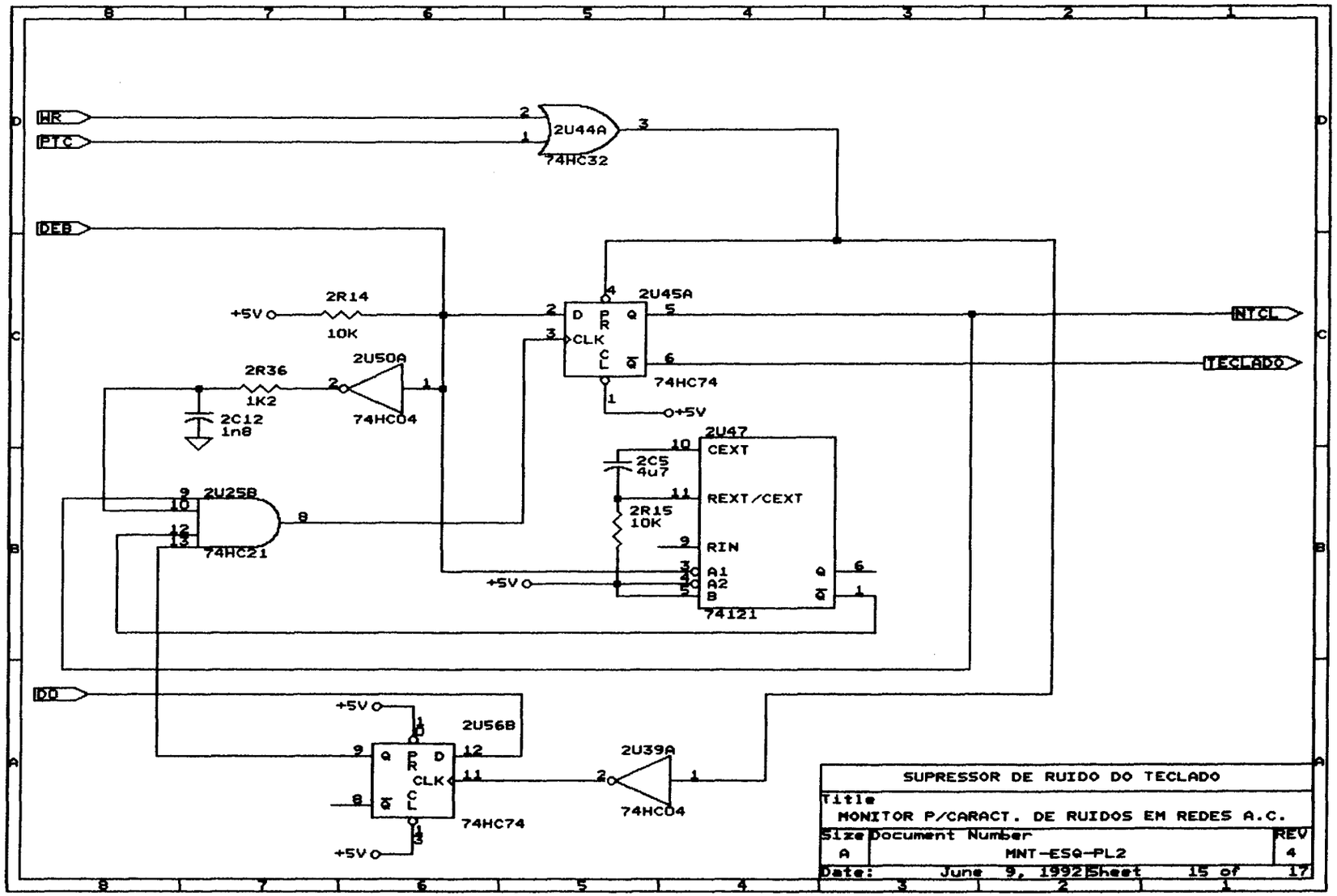


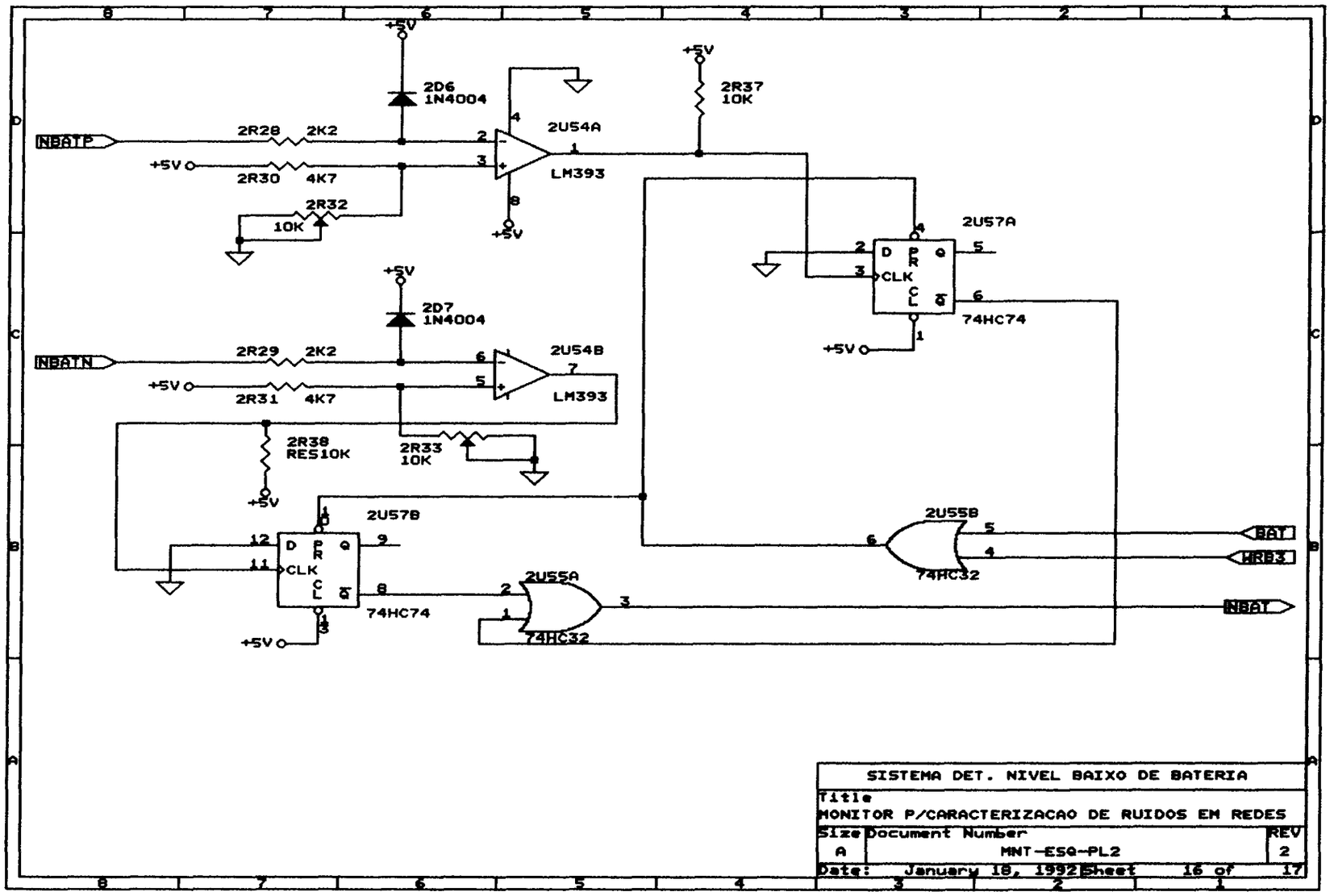
REGISTRADOR DO TECLADO

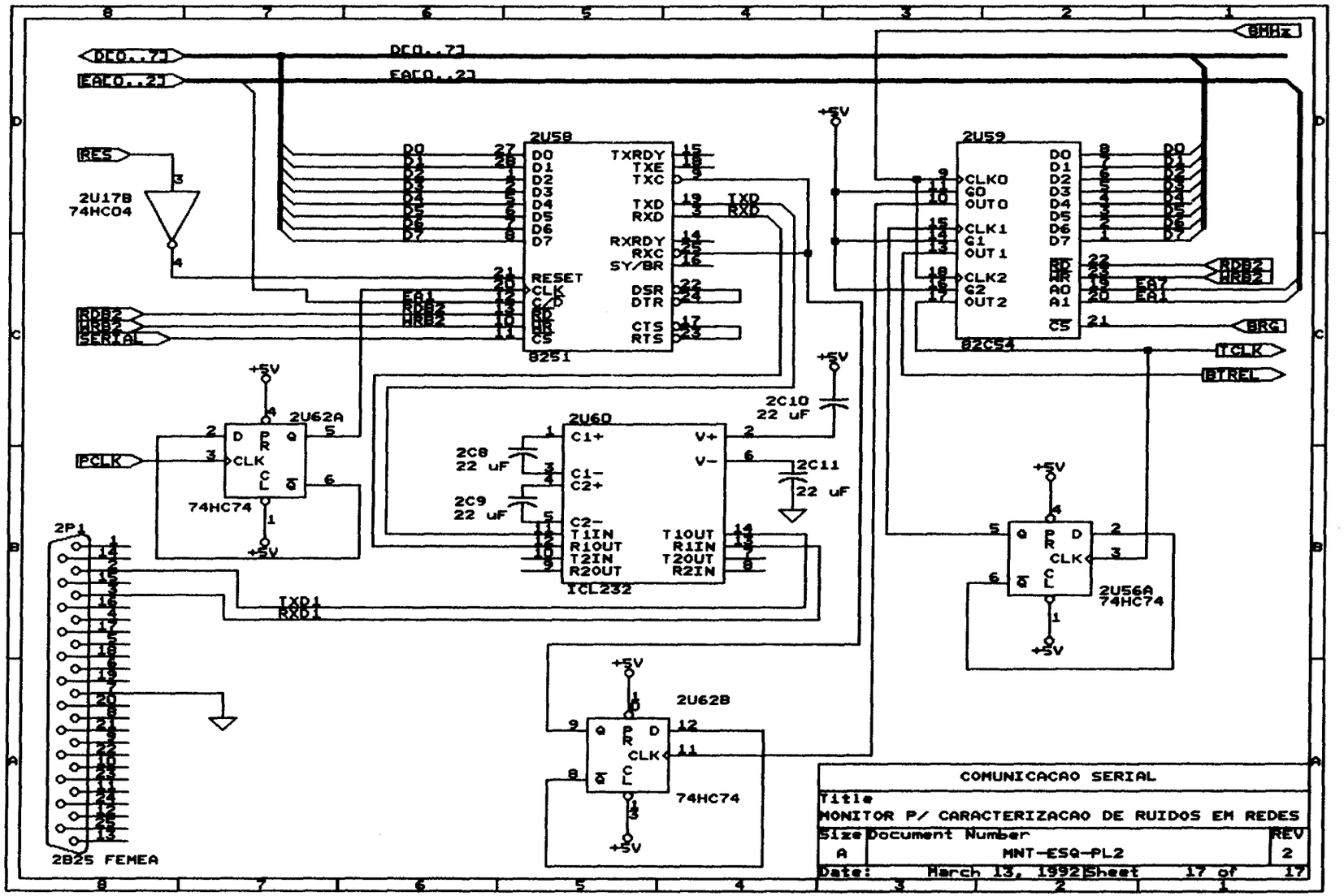
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.

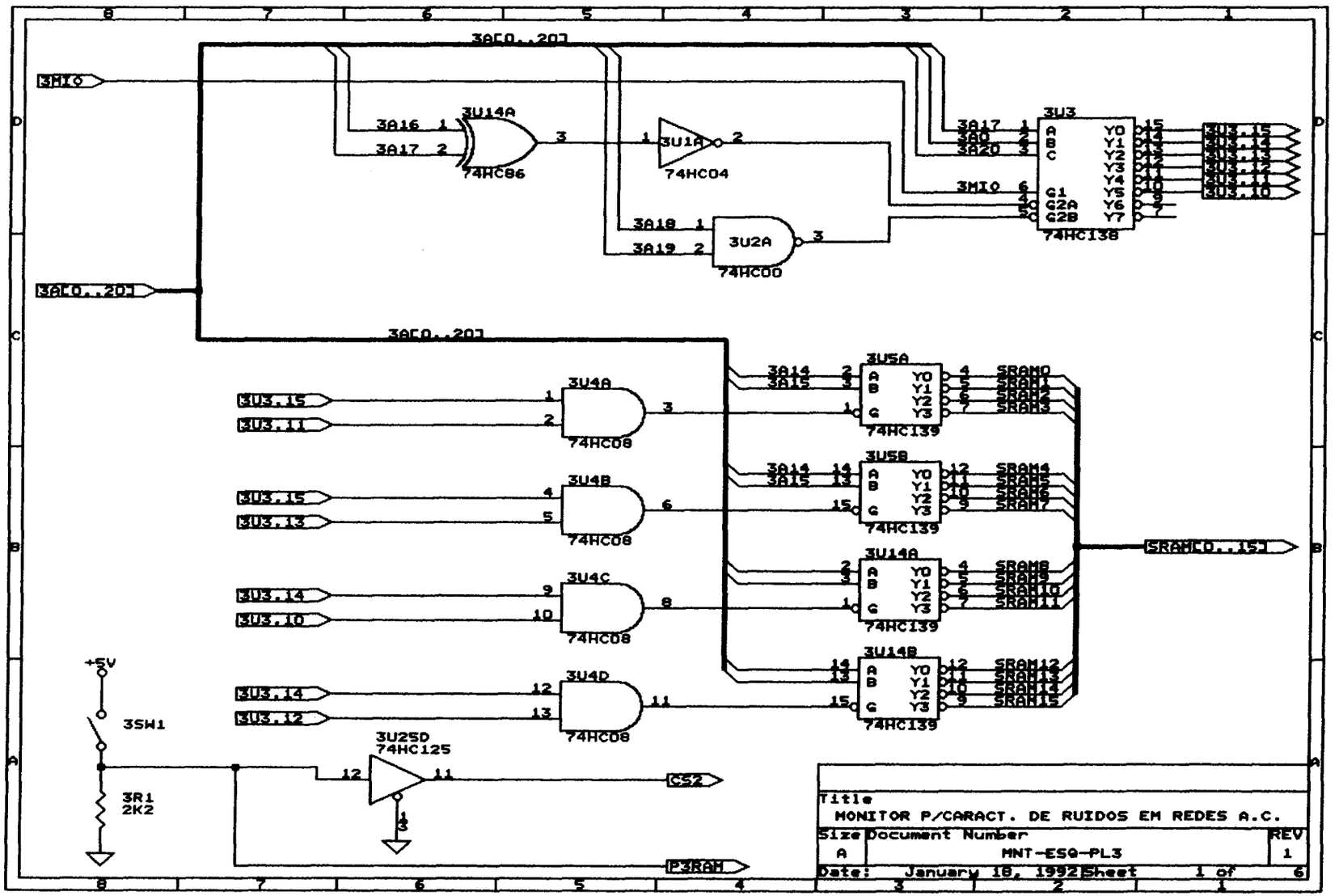
Size	Document Number	REV
A	MNT-ESQ-PL2	2

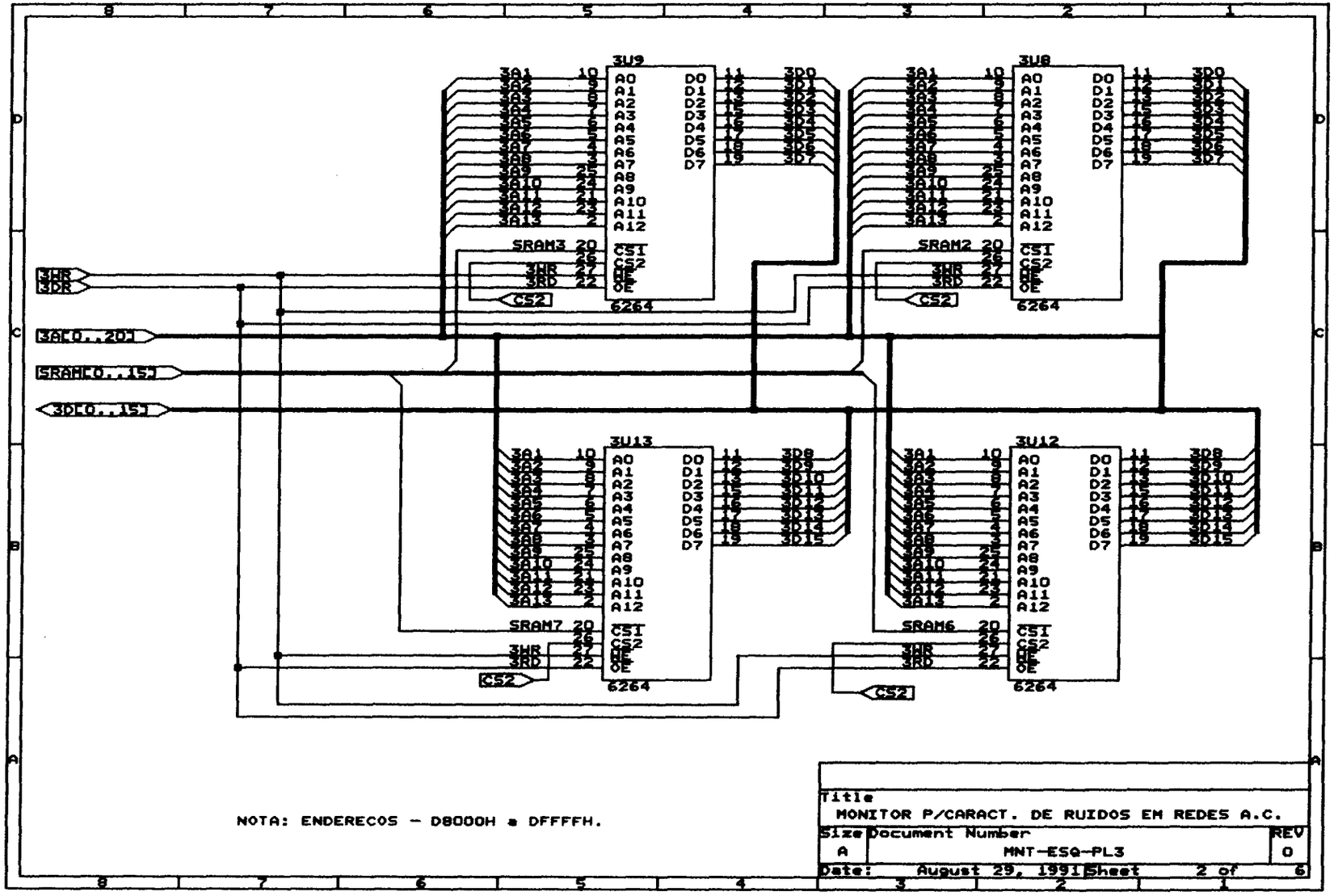
Date: June 9, 1992 Sheet 14 of 17


WR
PTC
2U44A
74HC32
DEB
+5V
2R14
10K
2U45A
D
PR
Q
CLK
CL
Q̄
74HC74
NTCL
TECLADO
2R36
1K2
2U50A
74HC04
2C12
1n8
2U47
CEXT
2C5
4u7
REXT/CEXT
2R15
10K
RIN
A1
A2
B
74121
2U25B
74HC21
DD
2U56B
74HC74
2U39A
74HC04
SUPRESSOR DE RUIDO DO TECLADO
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL2
REV
4
Date: June 9, 1992
Sheet 15 of 17


+5V
2D6
1N4004
NBATP
2R28
2K2
+5V
2R30
4K7
2R32
10K
2U54A
LM393
+5V
2R37
10K
2U57A
D
PR
Q
CLK
CL
Q̄
74HC74
+5V
+5V
2D7
1N4004
NBATN
2R29
2K2
+5V
2R31
4K7
2U54B
LM393
2R38
RES10K
2R33
10K
+5V
2U57B
D
PR
Q
CLK
CL
Q̄
74HC74
+5V
2U55A
74HC32
2U55B
74HC32
BAT
HRB3
NBAT
SISTEMA DET. NIVEL BAIXO DE BATERIA
Title
MONITOR P/CARACTERIZACAO DE RUIDOS EM REDES
Size
A
Document Number
MNT-ESQ-PL2
REV
2
Date: January 18, 1992
Sheet 16 of 17

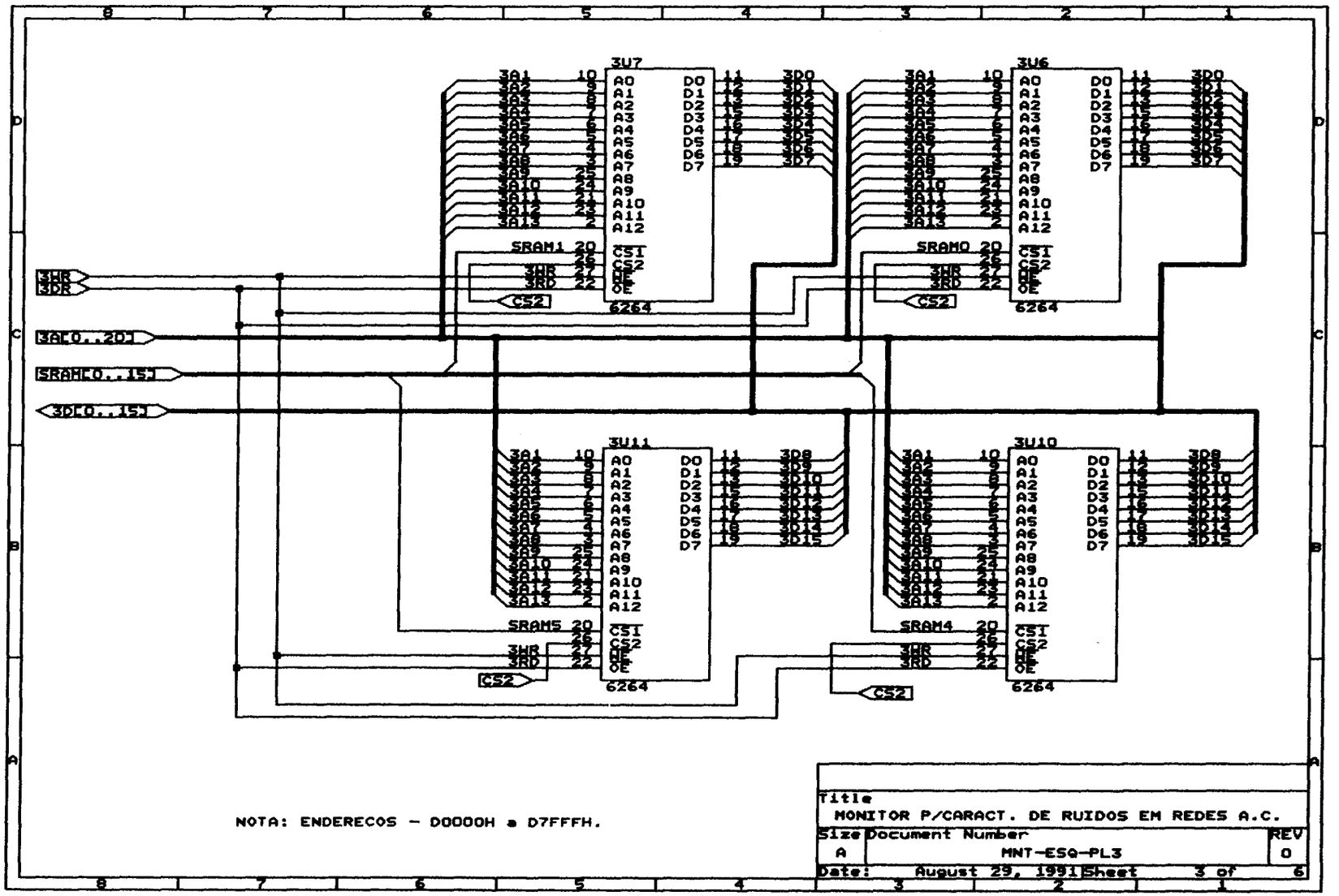

DE[0..7]
EA[0..2]
RES
2U17B
74HC04
2U58
8251
RESET
CLK
C/D
RD
WR
CS
TXRDY
TXE
TXC
TXD
RXD
RXRDY
RXC
SY/BR
DSR
DTR
CTS
RTS
RDB2
HRB2
SERIAL
2U59
82C54
CLK0
G0
OUT0
CLK1
G1
OUT1
CLK2
G2
OUT2
A0
A1
8MHz
BRG
TCLK
BTREL
+5V
2U62A
74HC74
PCLK
2U60
ICL232
C1+
C1-
C2+
C2-
V+
V-
T1IN
R1OUT
T2IN
R2OUT
T1OUT
R1IN
T2OUT
R2IN
2C8
22 uF
2C9
22 uF
2C10
22 uF
2C11
22 uF
2P1
2B25 FEMEA
TXD1
RXD1
2U62B
74HC74
2U56A
74HC74
COMUNICACAO SERIAL
Title
MONITOR P/ CARACTERIZACAO DE RUIDOS EM REDES
Size
A
Document Number
MNT-ESQ-PL2
REV
2
Date: March 13, 1992
Sheet 17 of 17


3MIO
3A[0..20]
3U14A
74HC86
3U1A
74HC04
3U2A
74HC00
3U3
74HC138
3U4A
74HC08
3U4B
74HC08
3U4C
74HC08
3U4D
74HC08
3U5A
74HC139
3U5B
74HC139
3U14A
74HC139
3U14B
74HC139
SRAM[0..15]
+5V
3SW1
3R1
2K2
3U25D
74HC125
CS2
P3RAM
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL3
REV
1
Date: January 18, 1992
Sheet 1 of 6



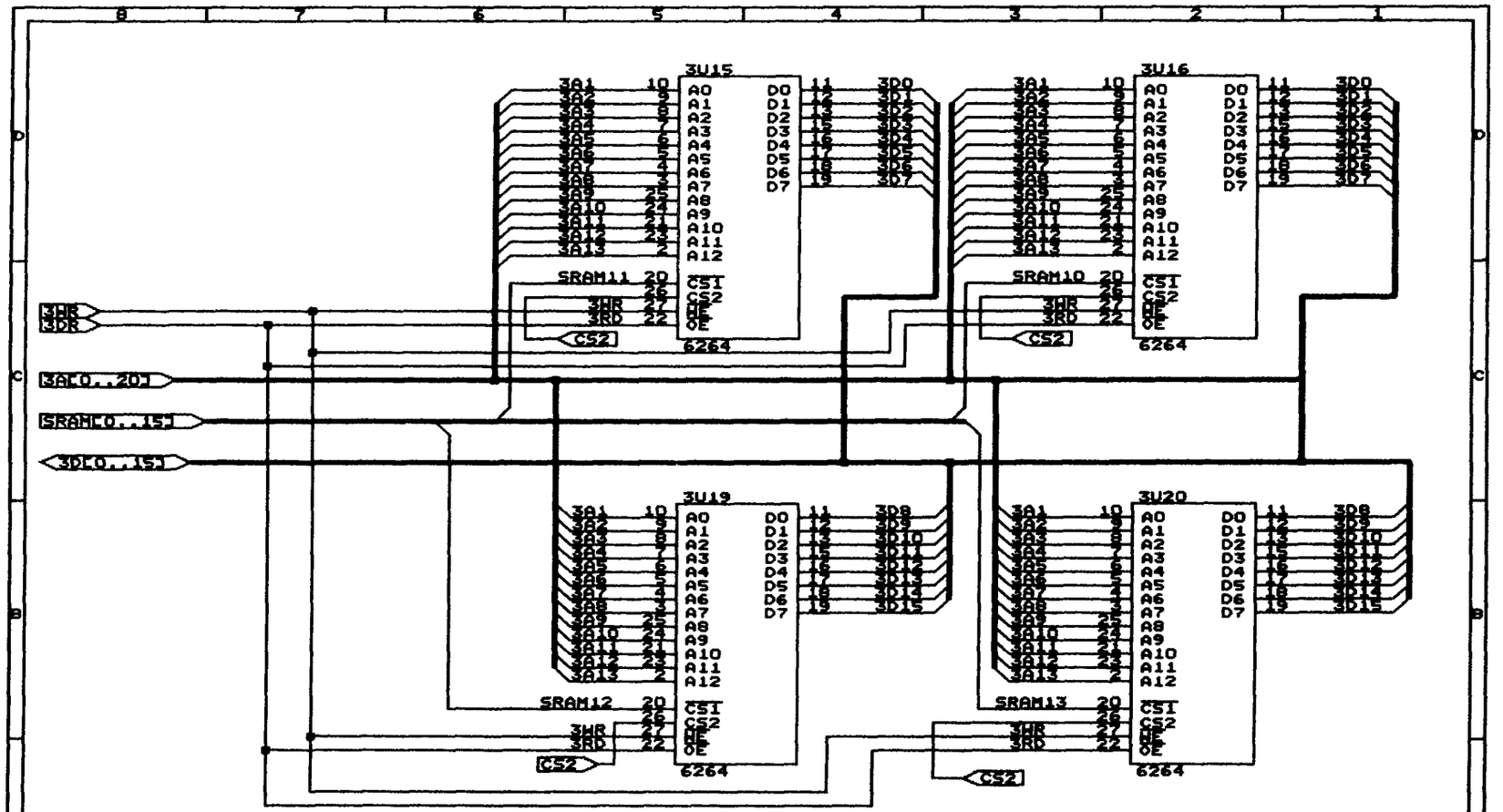
NOTA: ENDERECOS - D8000H a DFFFFH.

Title	MONITOR P/CARACT. DE RUIDOS EM REDES A.C.	
Size A	Document Number MNT-ESQ-PL3	REV 0
Date: August 29, 1991	Sheet 2 of 6	



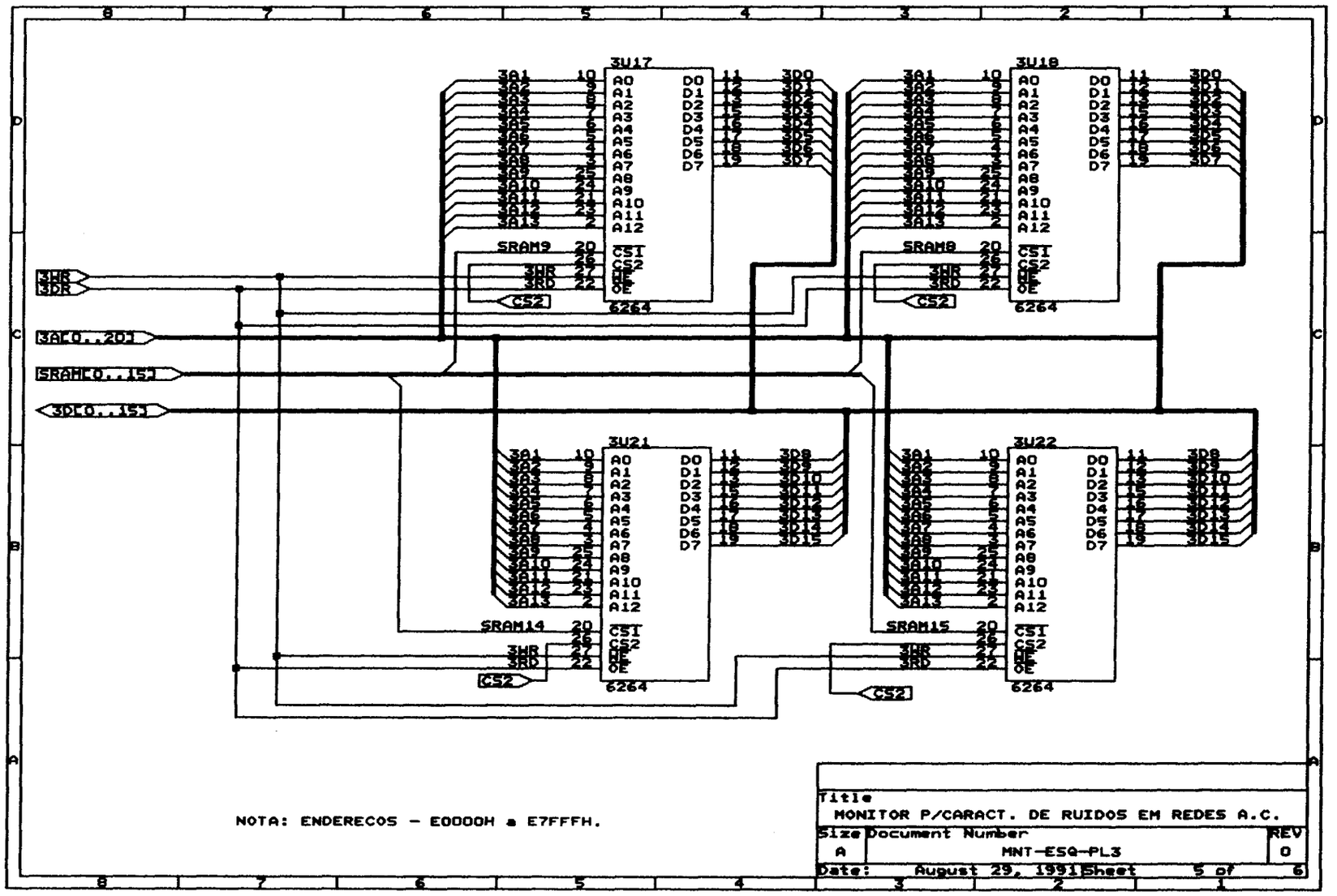
NOTA: ENDERECOS - D0000H a D7FFFH.

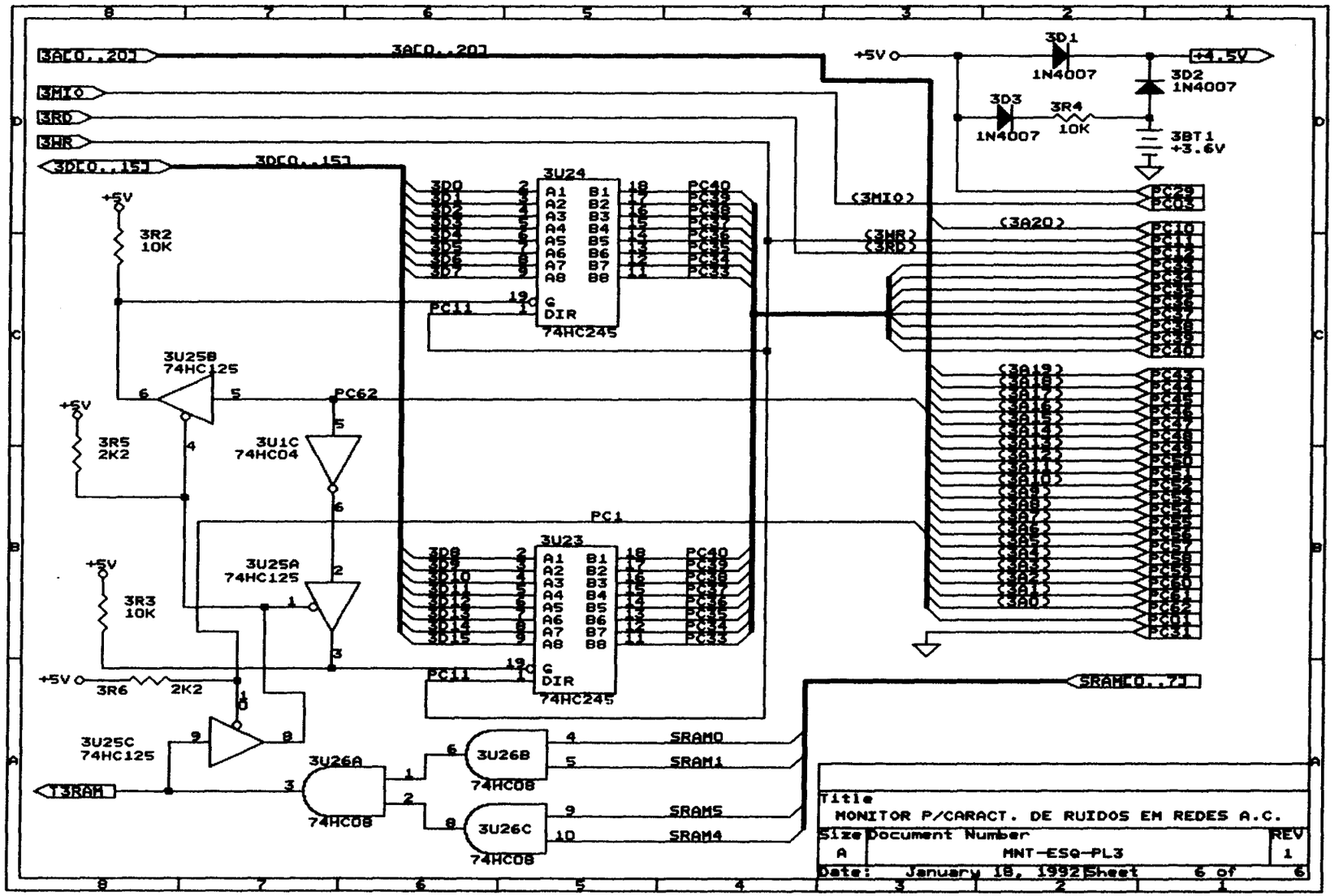
Title	MONITOR P/CARACT. DE RUIDOS EM REDES A.C.	
Size A	Document Number MNT-ESQ-PL3	REV 0
Date: August 29, 1991	Sheet 3 of 6	

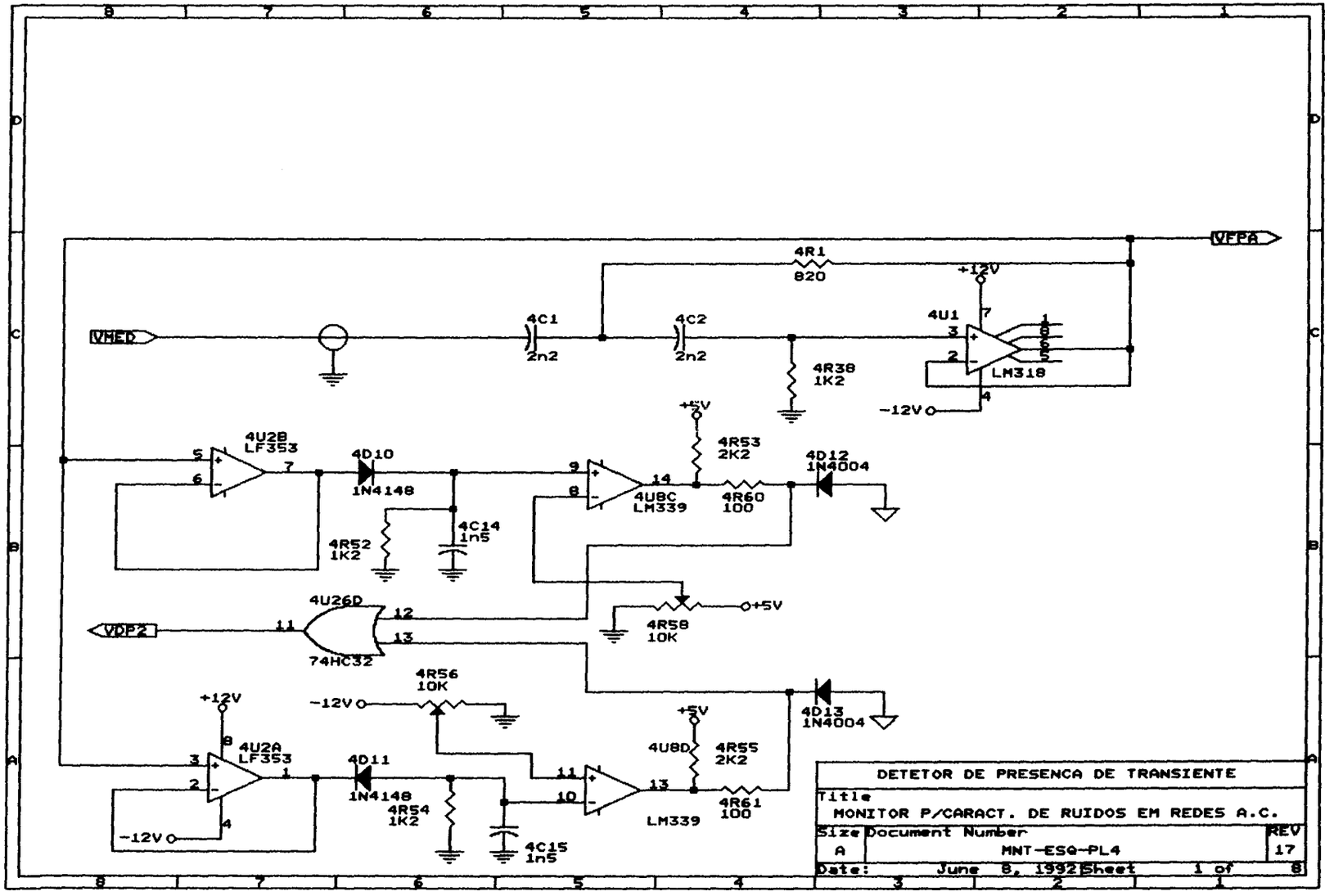


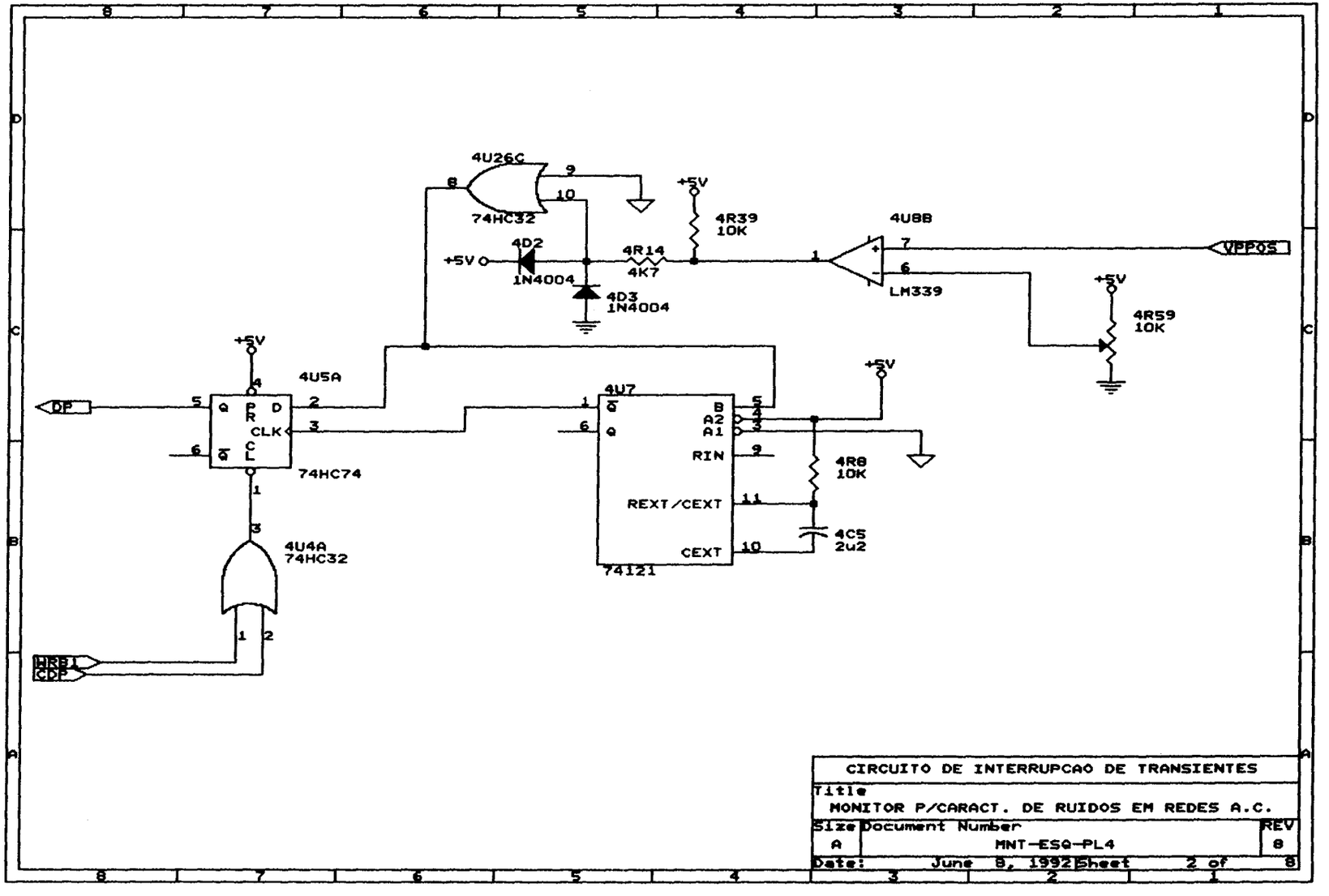
NOTA: ENDERECOS - E8000H a EFFFFH.

Title	MONITOR P/CARACT. DE RUIDOS EM REDES A.C.	
Size A	Document Number MNT-ESQ-PL3	REV 0
Date: August 29, 1991	Sheet 4 of 6	


3U17
3U18
3U21
3U22
6264
SRAM9
SRAM8
SRAM14
SRAM15
3HR
3DR
3RD
CS2
3A[0..20]
SRAM[0..15]
3D[0..15]
NOTA: ENDERECOS - E0000H a E7FFFH.
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL3
REV
0
Date: August 29, 1991
Sheet 5 of 6


3A[0..20]
3MIO
3RD
3WR
3D[0..15]
+5V
3R2
10K
3U25B
74HC125
PC62
3R5
2K2
3U1C
74HC04
3U25A
74HC125
3R3
10K
3R6
2K2
3U25C
74HC125
T3RAM
3U26A
74HC08
3U26B
74HC08
3U26C
74HC08
SRAM0
SRAM1
SRAM5
SRAM4
3U24
74HC245
3U23
74HC245
PC11
PC1
3D1
1N4007
3D2
1N4007
3D3
1N4007
3R4
10K
3BT1
+3.6V
+4.5V
SRAH[0..7]
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL3
REV
1
Date: January 18, 1992
Sheet 6 of 6


VFPA
4R1
820
+12V
4U1
LM318
VMED
4C1
2n2
4C2
2n2
4R38
1K2
-12V
+5V
4R53
2K2
4U2B
LF353
4D10
1N4148
4U8C
LM339
4R60
100
4D12
1N4004
4R52
1K2
4C14
1n5
4U26D
VDP2
74HC32
4R58
10K
+5V
4R56
10K
-12V
+12V
4D13
1N4004
4U2A
LF353
4D11
1N4148
4R54
1K2
4C15
1n5
4U8D
4R55
2K2
4R61
100
LM339
-12V
DETETOR DE PRESENCA DE TRANSIENTE
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL4
REV
17
Date: June 8, 1992
Sheet 1 of 8

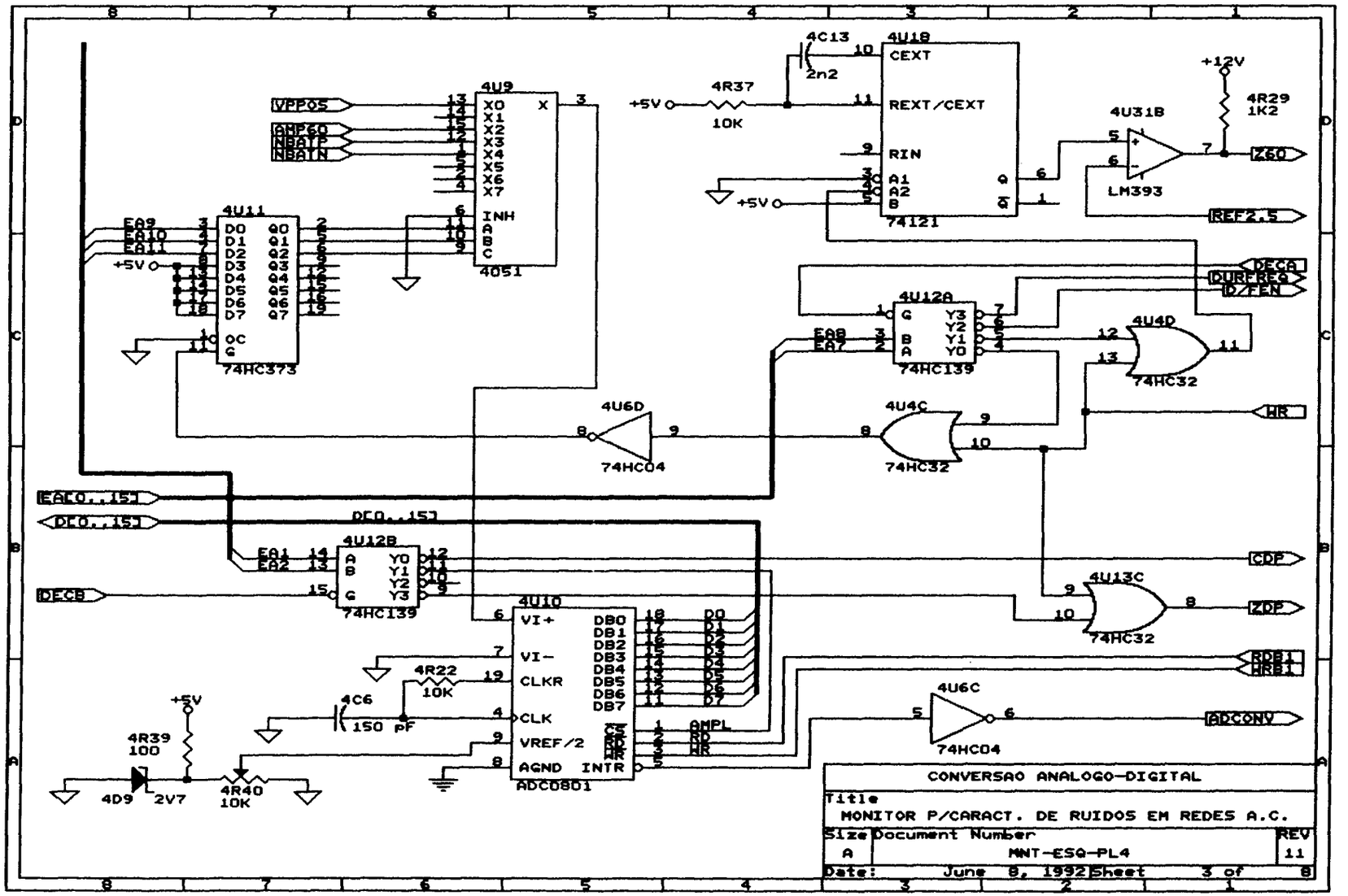


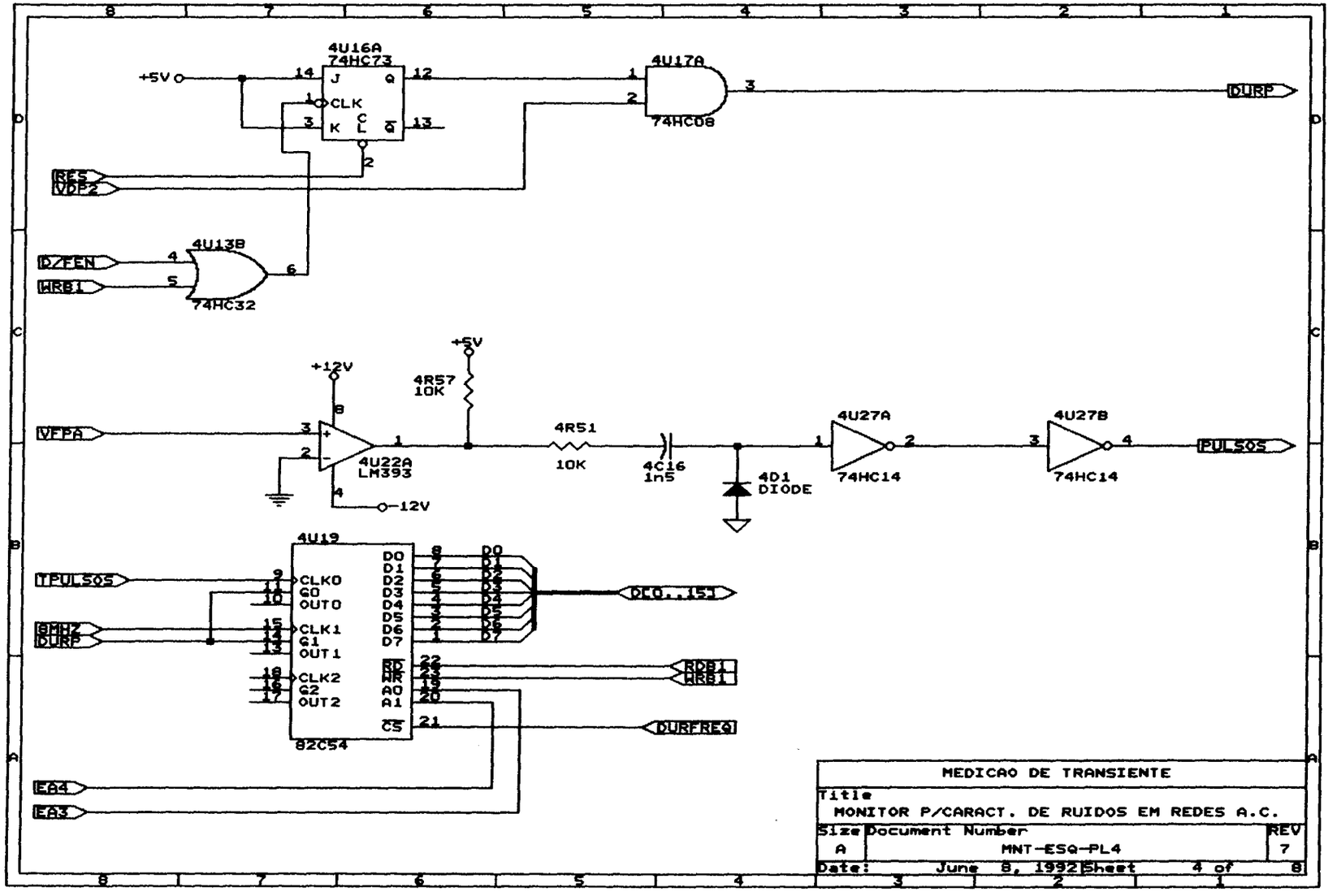
CIRCUITO DE INTERRUPCAO DE TRANSIENTES

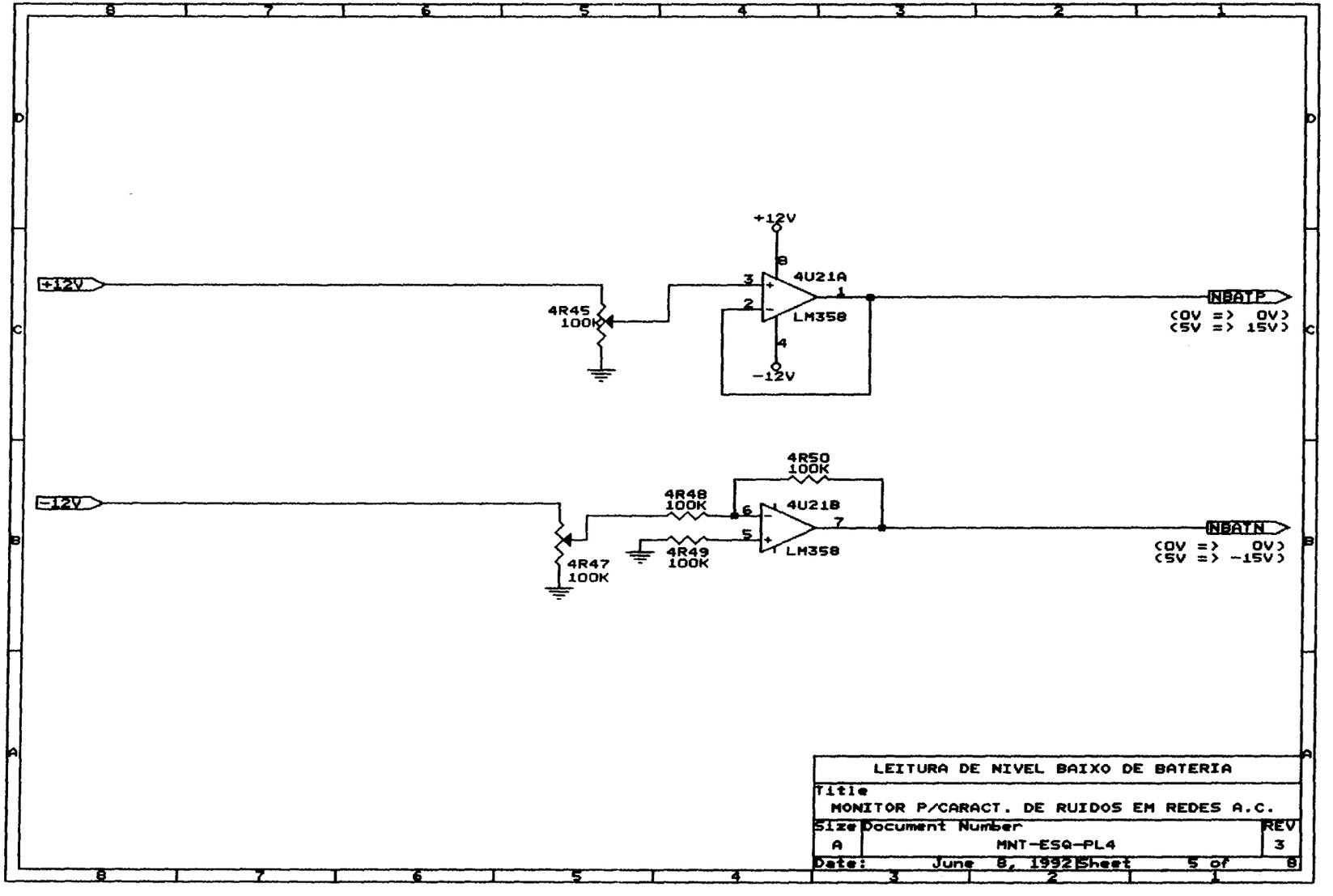
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.

Size	Document Number	REV
A	MNT-ESQ-PL4	8

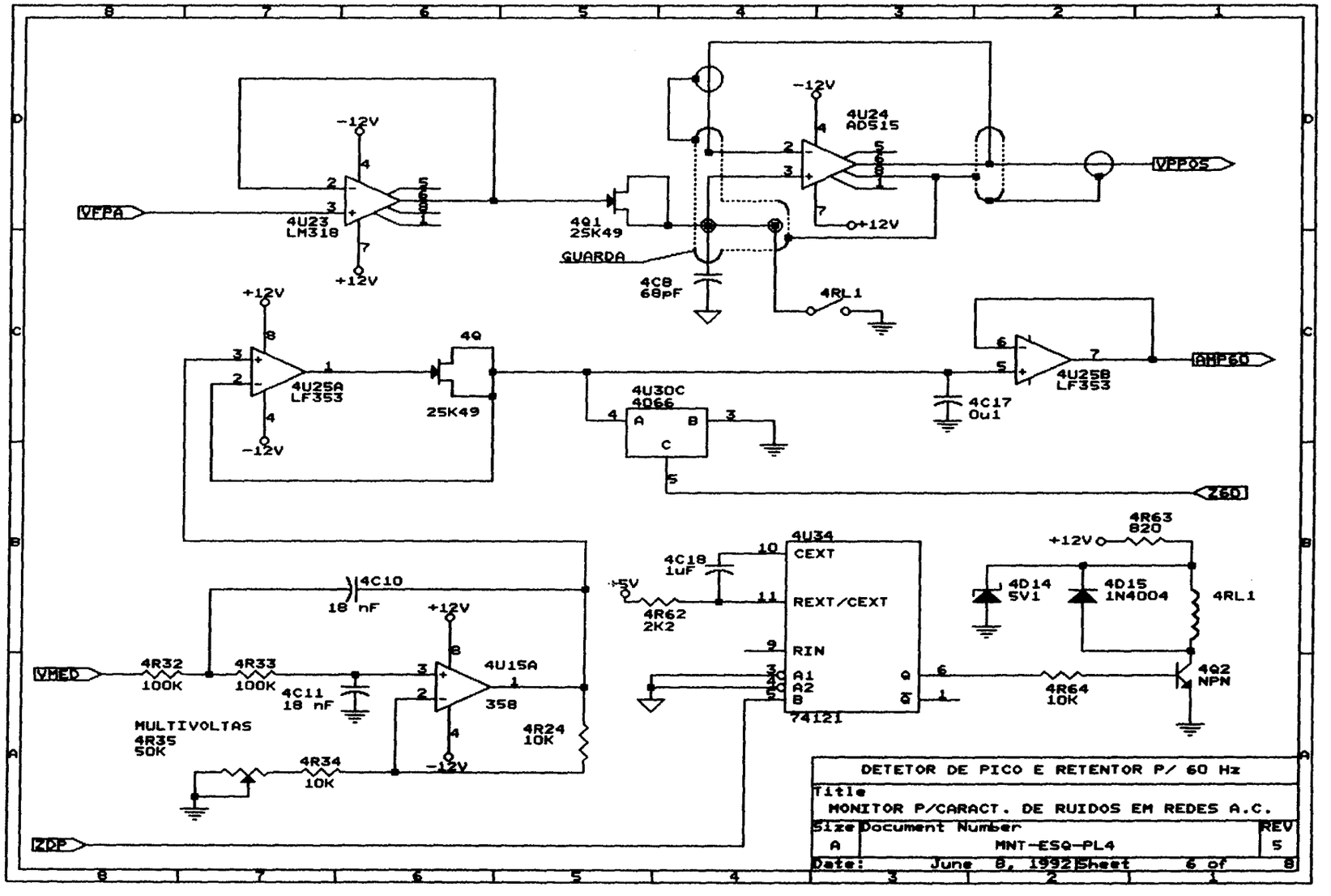
Date: June 8, 1992 Sheet 2 of 8

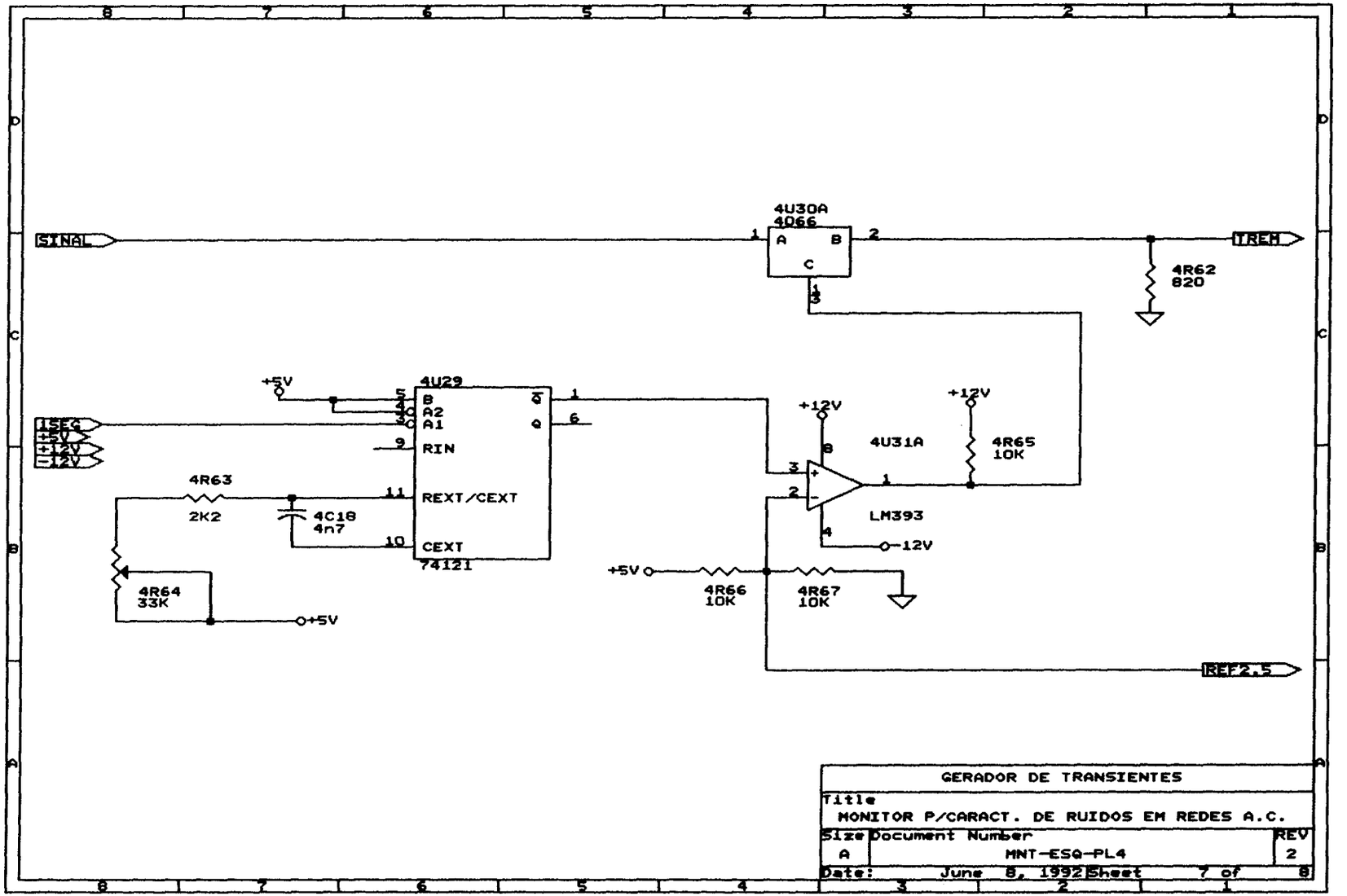

4U9
4051
VPP05
AMP60
NBATP
NBATN
X0
X1
X2
X3
X4
X5
X6
X7
X
INH
A
B
C
4U11
74HC373
EA9
EA10
EA11
+5V
D0
D1
D2
D3
D4
D5
D6
D7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
OC
G
4C13
2n2
4U18
74121
CEXT
REXT/CEXT
RIN
A1
A2
B
Q
4R37
10K
+12V
4R29
1K2
4U31B
LM393
Z60
REF2.5
DECA
DURREQ
D/FEN
4U12A
74HC139
EA8
EA7
4U4D
74HC32
4U6D
74HC04
4U4C
74HC32
HR
EA[0..15]
DF[0..15]
4U12B
74HC139
EA1
EA2
DECB
CDF
4U13C
74HC32
ZDF
4U10
ADC0801
VI+
VI-
CLKR
CLK
VREF/2
AGND
INTR
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
D0
D1
D2
D3
D4
D5
D6
D7
AMPL
RD
HR
RDB1
HRB1
4R22
10K
4C6
150 pF
4R39
100
4D9
2V7
4R40
10K
4U6C
74HC04
ADCONV
CONVERSAO ANALOGO-DIGITAL
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL4
REV
11
Date: June 8, 1992
Sheet 3 of 8

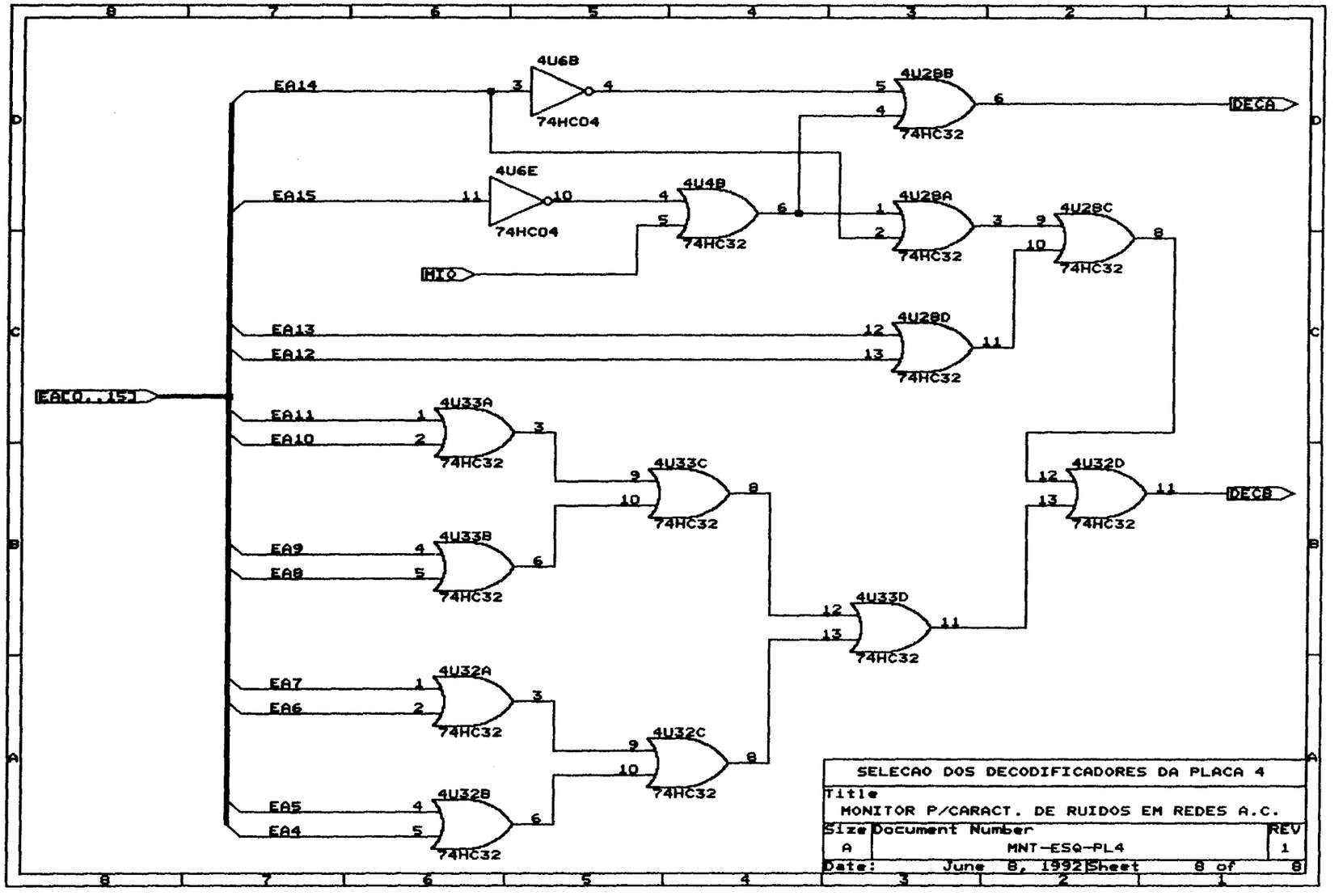

4U16A
74HC73
+5V
J
Q
CLK
K
CL
4U17A
74HC08
DURP
RES
VDP2
4U13B
74HC32
DZFEN
WRB1
+12V
+5V
4R57
10K
VFPA
4U22A
LM393
-12V
4R51
10K
4C16
1n5
4D1
DIODE
4U27A
74HC14
4U27B
74HC14
PULSOS
4U19
82C54
TPULSOS
8MHZ
DURP
CLK0
G0
OUT0
CLK1
G1
OUT1
CLK2
G2
OUT2
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
CS
DC0..15]
RDB1
WRB1
DURFREQ
EA4
EA3
MEDICAO DE TRANSIENTE
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL4
REV
7
Date: June 8, 1992
Sheet 4 of 8

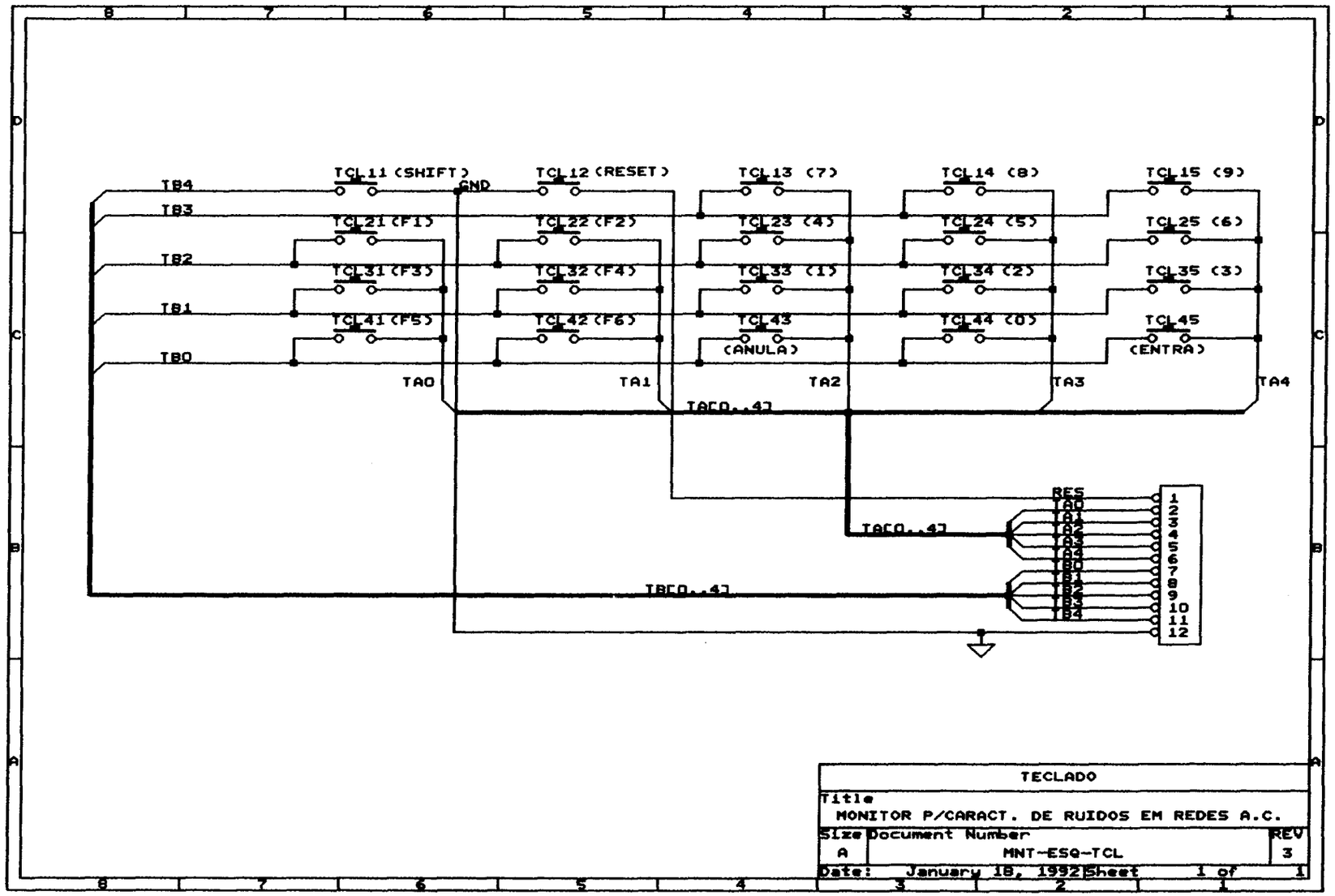


SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FISICA

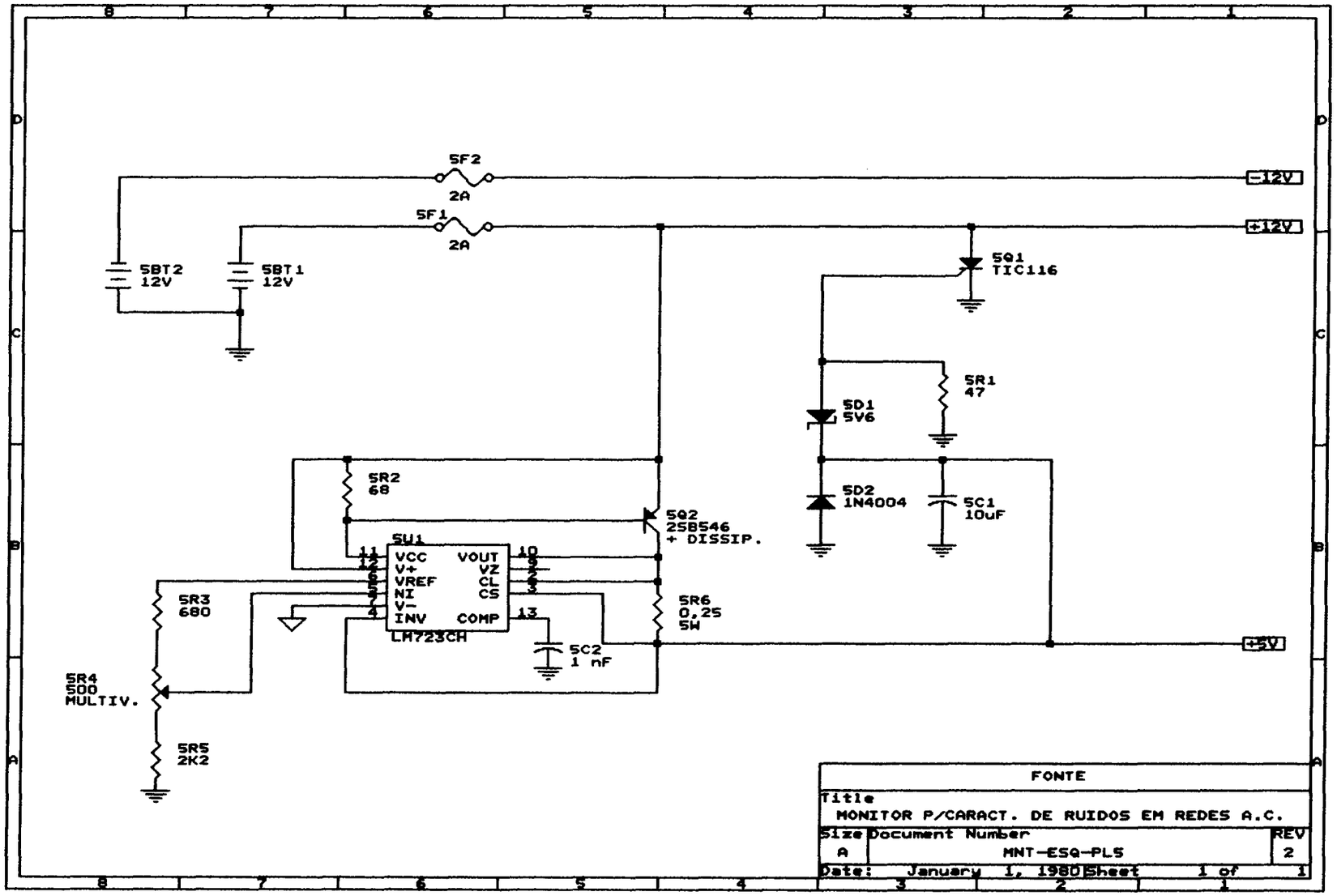

VFPA
4U23
LM318
-12V
+12V
4Q1
25K49
GUARDA
4C8
68pF
4U24
AD515
VPPOS
4RL1
4U25A
LF353
4Q
25K49
4U30C
4066
4C17
0u1
4U25B
LF353
AMP60
Z60
4C10
18 nF
VMED
4R32
100K
4R33
100K
4C11
18 nF
4U15A
358
MULTIVOLTAS
4R35
50K
4R34
10K
4R24
10K
ZDP
4U34
CEXT
REXT/CEXT
RIN
A1
A2
B
Q
74121
4C18
1uF
+5V
4R62
2K2
4R63
820
4D14
5V1
4D15
1N4004
4RL1
4R64
10K
4Q2
NPN
DETETOR DE PICO E RETENTOR P/ 60 Hz
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL4
REV
5
Date: June 8, 1992
Sheet 6 of 8

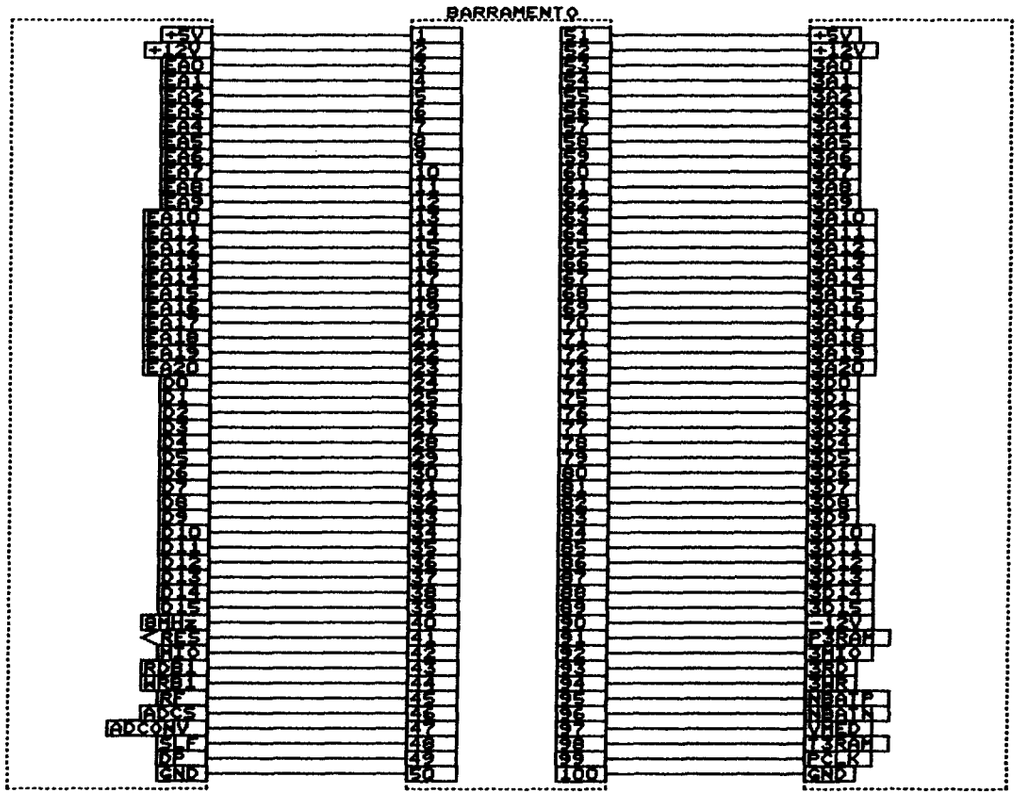

SINAL
4U30A
4066
A
B
C
TREM
4R62
820
+5V
4U29
B
A2
A1
RIN
REXT/CEXT
CEXT
74121
15EG
+5V
+12V
-12V
4R63
2K2
4C18
4n7
4R64
33K
+5V
+12V
4U31A
LM393
-12V
4R65
10K
+5V
4R66
10K
4R67
10K
REF2.5
GERADOR DE TRANSIENTES
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL4
REV
2
Date: June 8, 1992
Sheet 7 of 8


EA14
EA15
EA13
EA12
EA11
EA10
EA9
EA8
EA7
EA6
EA5
EA4
EAC0..15]
MIO
DECA
DECB
4U6B
4U6E
4U4B
4U28A
4U28B
4U28C
4U28D
4U33A
4U33B
4U33C
4U33D
4U32A
4U32B
4U32C
4U32D
74HC04
74HC32
SELECAO DOS DECODIFICADORES DA PLACA 4
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PL4
REV
1
Date: June 8, 1992
Sheet 8 of 8



TECLADO		
Title MONITOR P/CARACT. DE RUIDOS EM REDES A.C.		
Size A	Document Number MNT-ESQ-TCL	REV 3
Date: January 18, 1992	Sheet 1 of 1	


SF2
2A
SF1
2A
-12V
+12V
SBT2
12V
SBT1
12V
SQ1
TIC116
SR1
47
SD1
5V6
SD2
1N4004
SC1
10uF
SR2
68
SQ2
2SB546
+ DISSIP.
SU1
VCC
V+
VREF
NI
V-
INV
VOUT
VZ
CL
CS
COMP
LM723CH
SR3
680
SR4
500
MULTIV.
SR5
2K2
SR6
0,25
5W
SC2
1 nF
+5V
FONTE
Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.
Size
A
Document Number
MNT-ESQ-PLS
REV
2
Date: January 1, 1980
Sheet 1 of 1



Title
MONITOR P/CARACT. DE RUIDOS EM REDES A.C.

Size	Document Number	REV
A	MNT-ESQ-BUS	3

Date: January 6, 1992 Sheet 1 of 1

APÊNDICE C

EQUIPAMENTOS UTILIZADOS NA DEPURAÇÃO DE CIRCUITOS E PROGRAMAS

1 ANALISADOR LÓGICO PM3632

1.1 Fabricante : Philips , comercializado por John Fluke Mfg Co., Inc.

1.2 Entrada de dados

1.2.1 Número de canais : 4, 8, 16 ou 32.
1.2.2 Limite de transição : TTL ou variável
variável = -9V a +9V.
1.2.3 Impedância de entrada : 100 KΩ / 6 pF.

1.3 Clock

1.3.1 Interno : 5 Hz a 100 MHz.
1.3.2 Externo :

- tempo mínimo de "set-up" : 5 ns
- tempo máximo de "hold" : 1 ns
- período de clock : 14 ns
- largura mínima do pulso : 7 ns

1.3.3 Transição de clock : positiva ou negativa.
1.3.4 Taxa máxima de clock:

Número de canais	interno (MHz)	externo (MHz)
4	100,0	70,0
8	50,0	50,0
16	25,0	25,0
32	12,5	12,5

1.4 Trigger

1.4.1 Modo : sequencial ou combinacional.
1.4.2 Nº de palavras ou eventos : 4.
1.4.3 Largura máxima de palavra : 32 bits (respeitado o número de canais).
1.4.4 Sequências disponíveis : 19 e mais uma a ser definida pelo usuário, sendo limitadas ao seguinte:

Número de canais	Sequências disponíveis
4	0, 1, 2, 3
8	0, 1, 2, 3
16	0, 1 e 4
32	Todas

1.4.5 Atraso no trigger : 0 a 50 000 amostras.

1.5 Capacidade de memória:

Número de canais	Número de amostras
4	8000
8	4000
16	2000
32	1000

1.6 Monitor de vídeo : fósforo verde, 5".

1.7 Características Físicas:

- altura : 13 cm
- largura : 30 cm
- profundidade : 38 cm

1.8 Teclado : 39 teclas.

2 ANALISADOR LÓGICO PM 3655

2.1 Fabricante : Philips, comercializado por John Fluke Mfg Co., Inc.

2.2 Entrada de dados

2.2.1 Número de canais : 48.

2.2.2 Limite de Transição:

- TTL;
- ECL;
- variável = -4,8 Va + 7,5 V em degraus de 100 mV.

2.2.3 Impedância de entrada : 1 MΩ, 8 pF.

2.3 Clock

2.3.1 Interno : 50 Hz a 100 MHz.

2.3.2 Externo:

- frequência : DC a 100MHz
- tempo de "set-up" mínimo : 2,5 ns
- tempo de "hold" máximo : 2,5 ns

2.4 Trigger

2.4.1 Modo : sequencial ou combinacional, podendo ser interno ou externo.

2.4.2 Número de palavras ou níveis :4 níveis com três palavras "ou" cada , ou 2 palavras internas mais 1 externa.

2.4.3 Largura máxima de palavra : 48 bits.

2.4.4 Atraso no trigger : até 65534 clocks ou eventos.

2.5 Capacidade de memória : 2048 palavras de 48 bits.

2.6 Monitor de vídeo : fósforo âmbar, 9".

2.7 Características Físicas :

- altura : 18 cm
- largura : 46 cm
- profundidade : 53 cm

2.8 Teclado : 85 teclas.

3 OSCILOSCÓPIO 2213A

3.1 Fabricante : Tektronix.

3.2 Fator de deflexão : 2 mV/div. na escala de 5V/div.

3.3 Escala X : 2 mV/div a 5 V/div (prova 1x).
20 mV/div a 50 V/div (prova 10x).

3.4 Escala Y : 0,05 µs/div a 0,5 s/div.

3.5 Resposta a pulso : 5,8 ns.

3.6 Banda de frequência : DC a 60 Mhz.

4 MULTÍMETRO SK 6135

4.1 Fabricante : KAISE.

4.2 Display : cristal líquido de 4 dígitos.

4.3 Escalas

4.3.1 Tensão : 40 mV, 4 V, 40 V, 400 V, 1000 V (AC/DC).
4.3.2 Corrente : 400 μA, 4 mA, 40 mA, 400 mA, 10 A (DC).
4.3.3 Resistência: 400 Ω, 4 KΩ, 40 KΩ, 400 KΩ, 4 MΩ.

4.4 Resolução

4.4.1 Tensão : 100 μV, 1 mV, 10 mV, 1 V.
4.4.2 Corrente : 100 nA, 1 μA, 10 μA, 100 μA, 10 mA.
4.4.3 Resistência: 100 mΩ, 1 Ω, 10 Ω, 100 Ω, 1 KΩ, 10 KΩ.
Obs: Valores em correspondência às respectivas faixas de cada grandeza medida.

5 GERADOR DE FUNÇÕES SGF 2000B

5.1 Fabricante : Servus.

5.2 Níveis de saída :

- TTL;
- LO - ondas com amplitude máxima de 1V;
- HI - ondas com amplitude máxima de 9V.

5.3 Superposição de nível DC : 0 a ± 6V.

5.4 Faixas de frequência : 0,2/2/20/200/2K/200K/2M (Hz).

5.5 Formas de onda : triangular, senoidal, quadrada, pulso e

dente de serra.

5.6 Display : 4 dígitos, segmentos de LED's.

APÊNDICE D

MAPA DE ENDEREÇOS DE ENTRADA E SAÍDA

MAPA DE ENDEREÇOS DE ENTRADA E SAÍDA

DECOD.	FUNÇÃO	15 12	11 8	7 4	3 0	END.
2U41	RELÓGIO	0 X X X	X X X X	0 1 0 0	0 0 0 0	40H
		0 X X X	X X X X	0 1 0 0	0 0 1 0	42H
		0 X X X	X X X X	1 1 0 0	0 0 0 0	C0H
		0 X X X	X X X X	1 1 0 0	0 0 1 0	C2H
2U41	INTERRUP.	0 X X X	X X X X	0 1 0 0	0 1 0 0	44H
		0 X X X	X X X X	1 1 0 0	0 1 0 0	C4H
2U41	BRG	0 X X X	X X X X	0 1 0 1	1 0 0 0	58H
		0 X X X	X X X X	0 1 0 1	1 0 1 0	5AH
		0 X X X	X X X X	1 1 0 1	1 0 0 0	D8H
		0 X X X	X X X X	1 1 0 1	1 0 1 0	DAH
2U41	SERIAL	0 X X X	X X X X	0 1 0 1	1 1 0 0	5CH
		0 X X X	X X X X	0 1 0 1	1 1 1 0	5EH
2U41	RES(Y3)	0 X X X	X X X X	X 1 0 0	1 1 0 0	4CH
2U41	SIL1	0 X X X	X 0 X X	0 1 0 0	1 0 0 0	48H
E	SIL2	0 X X X	X 0 X X	1 1 0 0	1 0 0 0	C8H
1U1	SIL3	0 X X X	X 0 X X	0 1 0 0	1 0 1 0	4AH
	SIL4	0 X X X	X 0 X X	1 1 0 0	1 0 1 0	CAH
	LFF	0 X X X	X 1 0 0	0 1 0 0	1 0 X X	448H
	LF1	0 X X X	X 1 0 0	1 1 0 0	1 0 X X	4C8H
2U41	LF2	0 X X X	X 1 0 1	0 1 0 0	1 0 X X	548H
E	LF3	0 X X X	X 1 0 1	1 1 0 0	1 0 X X	5C8H
1U60	LF4	0 X X X	X 1 1 0	0 1 0 0	1 0 X X	648H
	RESERVA	0 X X X	X 1 1 0	1 1 0 0	1 0 X X	6C8H
	RESERVA	0 X X X	X 1 1 1	0 1 0 0	1 0 X X	748H
	RESERVA	0 X X X	X 1 1 1	1 1 0 0	1 0 X X	7C8H
2U41	INTRES	0 X X X	X X X X	0 1 0 1	0 0 0 X	50H
E	TECBLOQ	0 X X X	X X X X	1 1 0 1	0 0 0 X	D0H
2U42A	RESERVA	0 X X X	X X X X	0 1 0 1	0 0 1 X	52H
	BAT.BAIXA	0 X X X	X X X X	1 1 0 1	0 0 1 X	D2H
2U41	LTC	0 X X 1	X X X 0	1 1 0 1	0 1 X X	10D4H
E	PTC	0 X X 1	X X X 1	0 1 0 1	0 1 X X	1154H
2U48	RESERVA	0 X X 1	X X X 1	1 1 0 1	0 1 X X	11D4H
2U41,	DISP(DS0)	0 0 0 1	X X X 0	0 1 0 1	0 1 X X	1054H
2U48A	DISP(DS1)	0 0 1 1	X X X 0	0 1 0 1	0 1 X X	3054H
E	DISP(DS2)	0 1 0 1	X X X 0	0 1 0 1	0 1 X X	5054H
2U18B	DISP(DS3)	0 1 1 1	X X X 0	0 1 0 1	0 1 X X	7054H

DECOD.	FUNÇÃO	15 12	11 8	7 4	3 0	END.
2U12A	Z60	1 1 X X	X X X 0	1 X X X	X X X X	C086H
	D/FEN	1 1 X X	X X X 1	0 X X X	X 1 1 X	C106H
	TRANST0	1 1 X X	X X X 1	1 X X 0	0 1 1 X	C186H
	TRANST1	1 1 X X	X X X 1	1 X X 0	1 1 1 X	C18EH
	TRANST2	1 1 X X	X X X 1	1 X X 1	0 1 1 X	C196H
	TRANSCONT	1 1 X X	X X X 1	1 X X 1	1 1 1 X	C19EH
	VP	1 1 X X	0 0 0 0	0 X X X	X 1 1 X	C006H
	RESERVA	1 1 X X	0 0 1 0	0 X X X	X 1 1 X	C286H
	AMPL60	1 1 X X	0 1 0 0	0 X X X	X 1 1 X	C406H
	NBATP	1 1 X X	0 1 1 0	0 X X X	X 1 1 X	C606H
	NBATN	1 1 X X	1 0 0 0	0 X X X	X 1 1 X	C806H
	RESERVA	1 1 X X	1 0 1 0	0 X X X	X 1 1 X	CA00H
	RESERVA	1 1 X X	1 1 0 0	0 X X X	X 1 1 X	CC00H
	RESERVA	1 1 X X	1 1 1 0	0 X X X	X 1 1 X	CE00H
2U12B	CDP	1 0 0 0	0 0 0 0	0 0 0 0	0 0 0 X	8000H
	ADC	1 0 0 0	0 0 0 0	0 0 0 0	0 0 1 X	8002H
	RESERVA	1 0 0 0	0 0 0 0	0 0 0 0	0 1 0 X	8004H
	ZDP	1 0 0 0	0 0 0 0	0 0 0 0	0 1 1 X	8006H

Onde,

RELÓGIO : Endereços dos contadores do relógio do sistema.

INTERRUP.: Endereços da controladora de interrupção.

BRG : Endereços contadores do gerador de taxa de transmissão.

SERIAL : Endereços da controladora de comunicação serial.

SIL1 : Endereço para zerar contadores da frequência de 60 Hz.

SIL2 : Endereço para zerar contadores de IRF faixa 2.

SIL3 : Endereço para zerar contadores de IRF faixa 3.

SIL4 : Endereço para zerar contadores de IRF faixa 4.

LFF : Endereço para leitura do identificador da(s) faixa(s) de IRF solicitando registro.

LF1 : Endereço para leitura dos contadores da frequência de 60 Hz.

LF2 : Endereço para leitura dos contadores de IRF da faixa 2.

LF3 : Endereço para leitura dos contadores de IRF da faixa 3.

LF4 : Endereço para leitura dos contadores de IRF da faixa 4.

INTRES : Endereço do registrador para controle de estado dos "flip-flop's" da interface de interrupção.

TECBLOQ : Endereço do "flip-flop" de bloqueio do teclado.

BAT.BAIXA: Endereço para reabilitar "flip-flops" de interrupção de bateria em baixo nível.

LTC : Endereço para leitura de código de tecla digitada.

PTC : Endereço para habilitação do "flip-flop" de teclado.

DISP(DS0):	Endereço do dígito 0 do visor.
DISP(DS1):	Endereço do dígito 1 do visor.
DISP(DS2):	Endereço do dígito 2 do visor.
DISP(DS3):	Endereço do dígito 3 do visor.
Z60 :	Endereço para zerar detetor de pico da amplitude da rede.
D/FEN :	Endereço para habilitação/desabilitação da medição de transientes.
TRANST0 :	Endereço do contador 0 de medição de transiente.
TRANST1 :	Endereço do contador 1 e medição de transiente.
TRANST2 :	Endereço do contador 2 de medição de transiente.
TRANSCONT:	Endereço para controle do contador de transiente.
VP :	Endereço do multiplexador para seleção de leitura de pico de transiente.
AMPL60 :	Endereço do multiplexador para seleção de leitura de tensão da rede.
NBATP :	Endereço do multiplexador para seleção de leitura da bateria positiva.
NBATN :	Endereço do multiplexador para seleção de leitura da bateria negativa.
CDP :	Endereço para habilitação do flip-flop do detetor de transiente.
ADC :	Endereço do conversor A/D.
ZDP :	Endereço para zerar detetor de pico de transiente.

SERVIÇO DE BIBLIOTECA E INFORMAÇÃO - IFQSC
FÍSICA

APÊNDICE E

MAPA DOS ENDEREÇOS DAS MEMÓRIAS

MAPA DOS ENDEREÇOS DAS MEMÓRIAS

TIPO DE MEMÓRIA	BANCO DE MEMÓRIA	ENDEREÇOS	
		INICIAL (H)	FINAL (H)
RAM AUXILIAR	BRA0+BRA1	00000	00FFE
RAM AUXILIAR	BRA1	00001	00FFF
RAM AUXILIAR	BRA0	00000	00FFE
ROM	BR0+BR4	F0000	F3FFE
ROM	BR1+BR5	F4000	F7FFE
ROM	BR2+BR6	F8000	FBFFE
ROM	BR3+BR7	FC000	FFFFE
ROM	BR4	F0001	F3FFF
ROM	BR5	F4001	F7FFF
ROM	BR6	F8001	FBFFF
ROM	BR7	FC001	FFFFF
ROM	BR0	F0000	F3FFE
ROM	BR1	F4000	F7FFE
ROM	BR2	F8000	FBFFE
ROM	BR3	FC000	FFFFE
3RAM	B3R0+B3R4	D0000	D3FFE
3RAM	B3R1+B3R5	D4000	D7FFE
3RAM	B3R2+B3R6	D8000	DBFFE
3RAM	B3R3+B3R7	DC000	DCFFE
3RAM	B3R8+B3R12	E0000	E3FFE
3RAM	B3R9+B3R13	E4000	E7FFE
3RAM	B3R10+B3R14	E8000	EBFFE
3RAM	B3R11+B3R15	EC000	ECFFE
3RAM	B3R4	D0001	D3FFF
3RAM	B3R5	D4000	D7FFF
3RAM	B3R6	D8000	DBFFF
3RAM	B3R7	DC000	DCFFF
3RAM	B3R12	E0001	E3FFF
3RAM	B3R13	E4001	E7FFF
3RAM	B3R14	E8001	EBFFF
3RAM	B3R15	EC001	ECFFF
3RAM	B3R0	D0000	D3FFE

TIPO DE MEMÓRIA	BANCO DE MEMÓRIA	ENDEREÇOS	
		INICIAL (H)	FINAL (H)
3RAM	B3R1	D4000	D7FFE
3RAM	B3R2	D8000	D8FFE
3RAM	B3R3	DC000	DCFFE
3RAM	B3R8	E0000	E3FFE
3RAM	B3R9	E4000	E7FFE
3RAM	B3R10	E8000	EBFFE
3RAM	B3R11	EC000	ECFFE

Onde, RAM AUXILIAR: memória auxiliar.

ROM : memória de programa.

3RAM : memória de aquisição.

BRAn : Banco número 'n' da memória auxiliar n = 0 a 1.

BRn : Banco número 'n' da memória de programa n = 0 a 7.

B3R : Banco número 'n' da memória de aquisição n = 0 a 15.

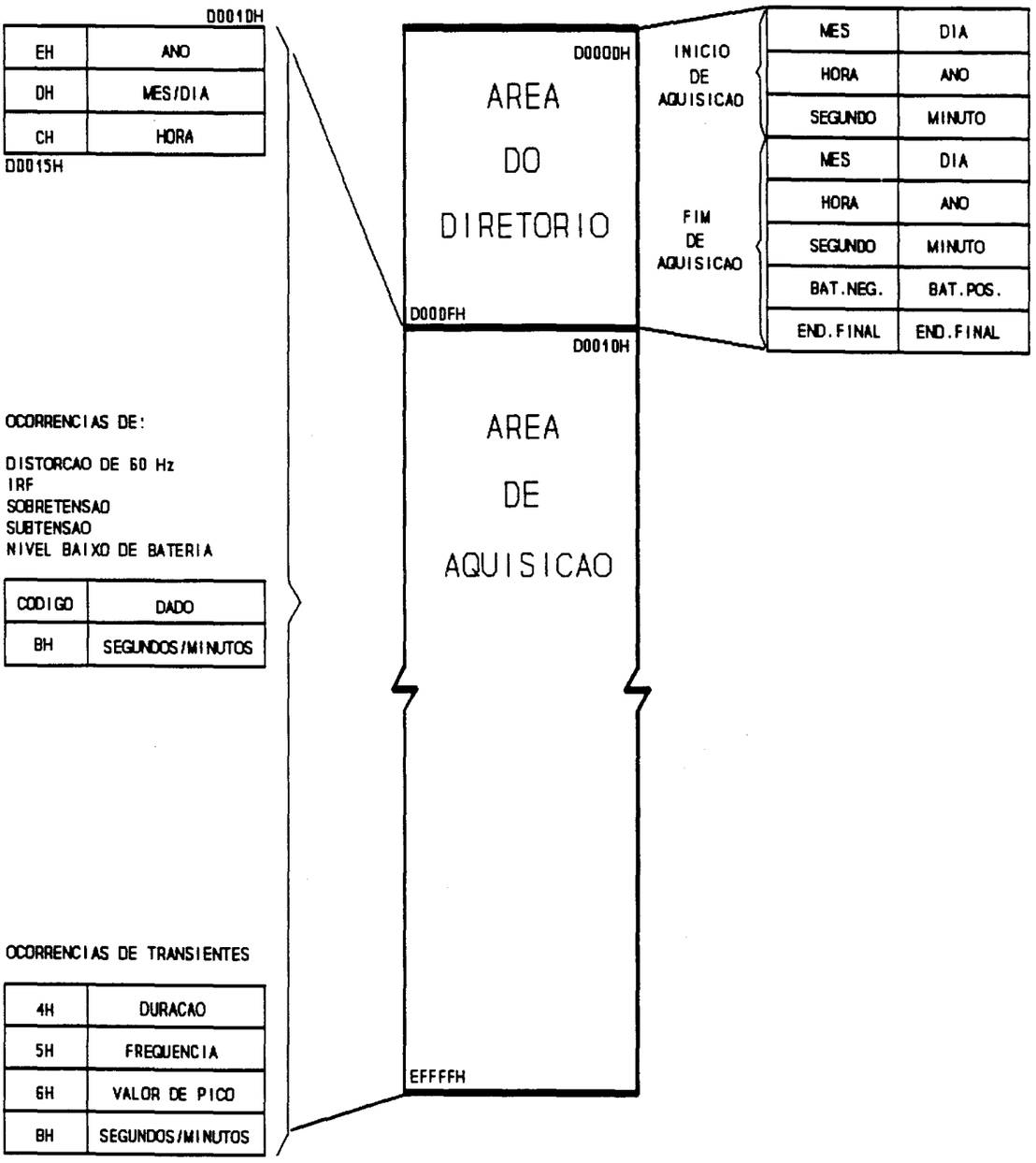


FIGURA E.1 - Mapa da Memória de Aquisição