UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE ENGENHARIA ELÉTRICA

LUIS GUSTAVO GUEDES PEREIRA DE CASTRO

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

CAMPINA GRANDE-PB

FEVEREIRO-2009

LUIS GUSTAVO GUEDES PEREIRA DE CASTRO

RELATÓRIO DE ESTÁGIO SUPERVISONADO

Relatório de Estágio Supervisionado submetida à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção da graduação em Engenharia Elétrica.

Orientador
Prof. Eurico Bezerra de Souza Filho

CAMPINA GRANDE -PB

2009

LUIS GUSTAVO GUEDES PEREIRA DE CASTRO

ESTÁGIO SUPERVISIONADO

Estágio Supervisionado aprovado como requisito parcial necessário para a obtenção da graduação em Engenharia Elétrica pela Universidade Federal de Campina Grande.

Data de Aprovação: 13 de fevereiro de 2009

BANCA EXAMINADORA:

_____

Prof. Eurico Bezerra de Souza Filho
Universidade Federal de Campina Grande
Orientador

_____

Prof. Talvanes Meneses Oliveira
Universidade Federal de Campina Grande

Agradecimentos

A Alberto dos Anjos pela oportunidade de trabalho. A Alex, a Jean e a Jurandi pelo apoio durante o estágio.

Á Kamilla Montenegro pela ajuda na revisão deste relatório.

Aos amigos pelo incentivo e apoio, e a todos que de alguma forma colaboraram com este trabalho.

SUMÁRIO

Lista de Figuras

Lista de tabelas

1. Objetivo

A disciplina Estágio Supervisionado tem como objetivo conscientizar o aluno dos problemas tecnológicos, econômicos e humanos do setor industrial e de prestação de serviços, assim como, aperfeiçoar o processo de ensino-aprendizagem, permitindo aos alunos a aplicação dos conhecimentos teóricos na prática.

Neste relatório serão descritas as atividades exercidas durante o estágio referente a disciplina Estágio Supervisionado, realizado na empresa de equipamentos eletrônicos APEL.

2. A Empresa

A APEL – Aplicações Eletrônicas IND.COM.LTDA. foi fundada em 1975, com o objetivo de desenvolver e industrializar produtos na área de eletro-eletrônica, absorvendo tecnologia oriunda da Universidade Federal de Paraíba (atual UFCG).

Localizada no distrito industrial de Campina Grande, na Paraíba, a APEL possui cerca de cem funcionários distribuídos entre os setores: Desenvolvimento de Produtos, Mecânica, Desenho, Teste de Vídeo, Teste de Áudio, Serigrafia, Montagem de Equipamentos e setores Administrativos.

Os principais produtos desenvolvidos são equipamentos para sonorização e radiodifusão tais como: caixas acústicas, pré-amplificadores, sonofletores, matrizes de comutação, amplificadores, mesas de som, misturadores de áudio, processadores de áudio, etc. Desenvolve também equipamentos na área de cronometria, painéis eletrônicos e "softwares". Fornece equipamentos de outras empresas e presta serviços de instalação, testes e treinamentos.

3. Cronograma

O cronograma elaborado foi dividido em quatro etapas com a duração de doze semanas. As atividades foram iniciadas no dia 15/09/2008 e finalizadas no dia 15/12/2008.

A primeira etapa foi de familiarização com as atividades da empresa, na qual foram conhecidos os diversos setores e produtos. As demais etapas foram realizadas no setor de desenvolvimento de produtos.

A segunda etapa de trabalho consistiu no projeto de um medidor de impedâncias e durou cerca de sete semanas.

Na terceira etapa foi feito um estudo dos Microcontroladores da NXP da série LPC21XX, que durou cerca de dois meses, da quarta à décima - segunda semana.

Na última etapa foram aplicados os conhecimentos sobre os microcontroladores LPC21XX para utilização do processador de áudio DDX-8001.

ETAPAS DE TRABALHO	Semanas											
	1	2	3	4	5	6	7	8	9	10	11	12
1	X	X										
2		X	X	X	X	X	X	X				
3				X	X	X	X	X	X	X	X	X
4									X	X	X	X
Data de início: 15/09/2008 Data de termino: 15/12/2008												

Tabela 1: Cronograma proposto

4. Primeira Atividade

A primeira atividade foi o projeto de um medidor de impedâncias de baixo custo. O medidor será utilizado principalmente para testar a impedância de altofalantes, detectando produtos com mau funcionamento ao instalar sistemas de sonorização.

4.1 Conceito de impedância

Impedância é a medida de oposição à passagem de corrente elétrica em circuitos de corrente alternada (CA). No domínio do tempo, podemos escrever a tensão e corrente como:

$$v(t) = V\cos(wt + \theta) = \mathrm{Re}[Ve^{jwt}] \quad (4.1)$$

$$i(t) = I\cos(wt + \phi) = \mathrm{Re}[Ie^{jwt}] \quad (4.2)$$

onde "w" é a freqüência angular, em rad/s, e "w" = 2.π.f (f = freqüência em hertz).

As equações (4.1) e (4.2) podem ser escritas na forma fasorial como:

$$\hat{V} = V\angle\theta \quad (4.3)$$

$$\hat{I} = I\angle\phi \quad (4.4)$$

O conceito de impedância é decorrente da Lei de Ohm, que é a relação algébrica entre corrente e tensão em um resistor (equação 4.5). A razão entre o fasor tensão e o fasor corrente em um circuito de corrente alternada é definido como a impedância Z e expressa em ohm.

$$V = RI \quad (4.5)$$

$$Z = \frac{V}{I} \quad (4.6)$$

$$Z = Z\angle(\theta - \phi) \quad (4.7)$$

$$Z = R + jX \quad (4.8)$$

Quando escrito na forma cartesiana (equação 4.8), a impedância tem duas componentes. A parte real é a resistência (R) e a parte imaginária é chamada de reatância (X). A reatância pode ser capacitiva (parte imaginária negativa) ou indutiva (parte imaginária positiva). Portanto:

$$Z = R + j(X_L - Xc) \quad (4.9)$$

onde:

$$X_L = wL \quad (4.10)$$

$$Xc = \tfrac{1}{wC} \quad (4.11)$$

e L é a indutância elétrica em Henry e C é a capacitância elétrica em Farad.

O módulo da impedância pode ser escrito como:

$$Z = \sqrt{R^2 + X^2} \quad (4.12)$$

4.2 Características dos altofalantes

Altofalante é o termo genérico usado para descrever uma ampla variedade de transdutores que convertem energia elétrica em acústica.

Os altofalantes mais utilizados são os de bobina móvel. Nesses altofalantes, ao aplicar corrente elétrica nos terminais das bobinas, surgirá um campo magnético que vai interagir com o campo natural do imã permanente, criando uma reação de atração ou repulsão, promovendo o movimento do diafragma. Esse movimento criará uma turbulência ritmada no ar, consequentemente, ondas sonoras. A figura 1 ilustra as principais partes de um altofalante.

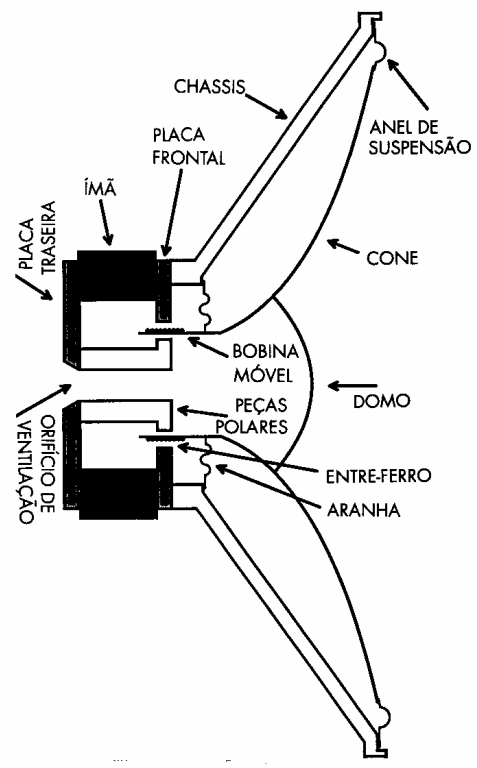


Figura 1: Partes de um altofalante de bobina móvel

As principais características dos altofalantes são: impedância, resposta em frequência, sensibilidade, potência e os parâmetros de Thiele-Small.

4.2.1 Impedância

A impedância de um altofalante depende da resistência do fio da bobina, da reatância indutiva e das correntes induzidas na bobina móvel. A impedância dos altofalantes varia com a frequência. Na figura 2, podemos destacar três impedâncias principais:

- Impedância máxima (Zmax): É a impedância na frequência de ressonância do altofalante. Na ressonância a impedância é puramente resistiva;
- Impedância mínima (Zmin): É a impedância mínima encontrada no gráfico após a frequência de ressonância;
- Impedância nominal (Znom): A impedância nominal é um valor de referência utilizado pelos fabricantes para descrever o valor da impedância do altofalante.

A impedância nominal é estabelecida como o valor de impedância a 1 kHz, e também pode ser calculada como:

$$Z_{nom} = 1.15 \cdot Z_{\min}$$

Figura 2: Curva de impedância de um altofalante de 4ohms

4.2.2 Resposta em frequência

A resposta em frequência é uma das mais importantes características dos altofalantes indicando a linearidade dos mesmos. A curva de resposta em frequência pode ser obtida de diversas formas. Pode-se submeter, para cada frequência, uma potência sempre igual a do altofalante; medir a potência sonora gerada e transformá-la novamente em energia elétrica, transformando-a, posteriormente, em energia mecânica para impulsionar um estilete que traça a curva.

No início da curva de resposta em frequência, encontra-se um máximo nas baixas frequências, o qual corresponde à frequência de ressonância do diafragma. A curva apresenta algumas oscilações até chegar à extremidade direita, na frequência de corte. A maioria dos altofalantes possui uma diferença de intensidade sonora entre cristas e depressões próximas das mesmas inferior a 10dB. Alguns altofalantes estão enquadrados na categoria de alta fidelidade, pois o máximo da frequência de ressonância não ultrapassa em 5 dB do valor mínimo seguinte. A figura 3 ilustra a curva de resposta em frequência para um altofalante de alta fidelidade.

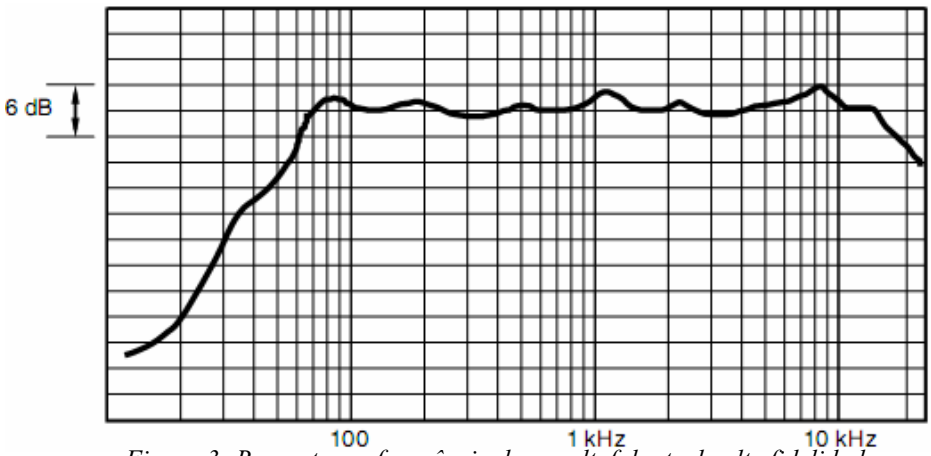


Figura 3: Resposta em frequência de um altofalante de alta fidelidade

4.2.3 Sensibilidade

A sensibilidade de um altofalante é a medida da eficiência com que ele converte a energia elétrica em acústica. Ela é medida à um metro de distância de um altofalante que está recebendo 1W de potência e usualmente expressa em dB. A sensibilidade indica o volume do som do altofalante. Se um altofalante possui uma sensibilidade maior que o outro, com uma tensão menor, consegui-se o mesmo volume de som.

4.2.4 Potência

A potência de um altofalante depende das suas dimensões e da sua forma de construção. Corresponde à máxima potência elétrica suportada pelo altofalante antes que a bobina seja danificada. Também chamada de Potência RMS ou eficaz, o teste para sua determinação é normalizado pela ABNT (NBR 10303).

4.2.5 Parâmetros de Thiele-Small

Os parâmetros Thiele-Small são comumente chamados de parâmetros eletromecânicos e são utilizados para o projeto de caixas acústicas. Foram desenvolvidos por A. N. Thiele da Australian Broadcasting Commission e aprimorado por Richard H. Small da Universidade de Sydney. Os parâmetros são:

- Fs: Freqüência de ressonância mecânica;
- Vas: Volume equivalente;
- Qts: Fator de qualidade total;
- Qes: Fator de qualidade elétrico;
- Qts: Fator de qualidade mecânico;
- η_o: Eficiência de referencia em meio espaço;
- Sd: Área efetiva do cone;
- $X_{MAX:}$ Deslocamento máximo p-p c/ 10% distorção;
- Vd: Volume deslocado = Sd x X_{MAX};
- X_{LIM}: Deslocamento máximo de pico antes do dano;

4.3 O Projeto

O medidor de impedâncias proposto é de baixo custo e portátil. Seu manuseio será restrito aos técnicos da APEL para auxiliar na instalação de sistemas de sonorização, medir a impedância de linhas e sonofletores, assim como, calibrar a impedância de linhas (taps). As características desejadas para o medidor foram:

- Faixa de medição: 2 a 1800 ohms
- Frequência de operação: 1 kHz
- Alimentação: 2 baterias de 9V
- Nível de sinal 2 volts

O medidor deve operar em 1 kHz a fim de emitir som no altofalante, garantindo ao instalador o correto funcionamento dos aparelhos. Serão utilizadas duas baterias de 9V como alimentação para o medidor, garantindo a portabilidade do mesmo.

4.3.1 Principio de funcionamento

O medidor possui uma escala associada a um potenciômetro linear e dois leds para indicar a impedância da carga.

Ao deslocar o indicador do medidor, quando houver uma mudança no estado dos leds, o valor onde houve a mudança corresponde ao valor da impedância da carga. Por exemplo, supondo que a carga a ser medida é de 24 ohms, se o indicador estiver abaixo do valor 24 Ω, o led esquerdo estará aceso e o direito apagado. Se ultrapassarmos a marca de 24 Ω, o led direito acenderá e o de esquerdo apagará. O botão liga deve ser mantido pressionado para que o circuito funcione.

O medidor possui três escalas: a de 24 Ω, 6 Ω e 600 Ω. As escalas podem ser selecionadas por meio de duas chaves seletoras. A figura 4 mostra o medidor projetado.

Uma onda senoidal de 2V de pico-a-pico é gerada pelo oscilador alimentado pelas baterias de 9V. O sinal é amplificado em corrente e aplicado ao circuito para medição. Esse é composto por um potenciômetro linear, uma resistência em série (Rs) e a carga. As resistências em série (Rs) possuem o mesmo valor das escalas associadas e as resistências Ru e Rd são utilizadas para calibrar a escala do medidor.

A medição da impedância é realizada indiretamente, baseada na tensão do potenciômetro e da carga. Os sinais no potenciômetro e na carga passam por um detector de pico e depois por um comparador, o qual modifica o estado dos dois leds de acordo com o valor de impedância correspondente no potenciômetro. O principio de funcionamento do medidor está ilustrado na figura 5.

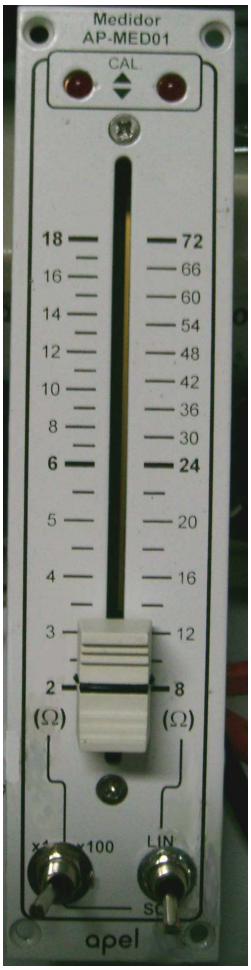


Figura 4: Medidor de Impedâncias

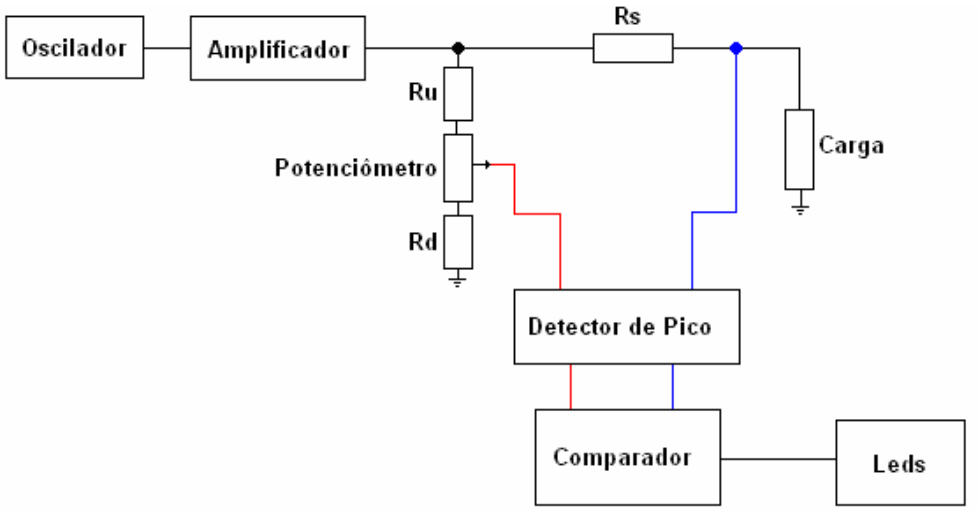


Figura 5: Principio de funcionamento do Medidor de Impedâncias

4.3.2 Circuito Elétrico

O circuito do medidor é composto pela alimentação, oscilador, amplificador, detector de pico e comparador.

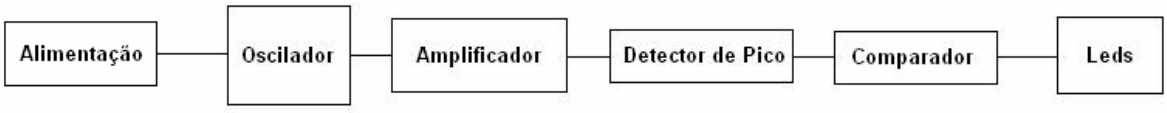


Figura 6: Circuito do medidor

4.3.2.1 Alimentação

O medidor é alimentado por duas baterias de 9V, proporcionando ao circuito uma tensão simétrica de ±9V. Para o funcionamento do circuito, o botão "liga" deve ser mantido pressionado. A figura 7 ilustra o esquema de alimentação do circuito.

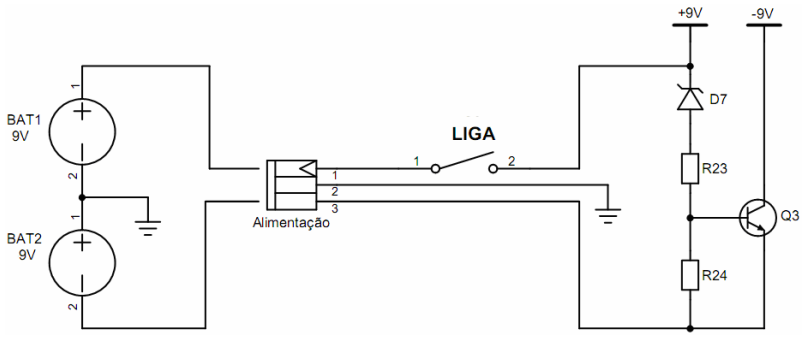


Figura 7: Alimentação do circuito

4.3.2.2 Oscilador

O oscilador senoidal deve gerar uma onda de 2V na frequência de 1kHz. O oscilador com ponte de Wien foi escolhido devido sua simplicidade de implementação. O CI utilizado foi o TL072 que possui dois amplificadores operacionais e será compartilhado com o circuito do amplificador.

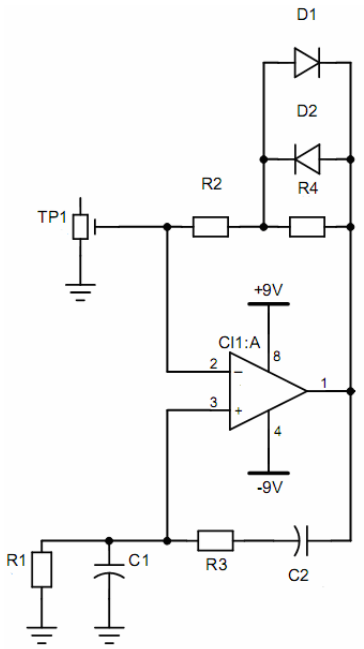


Figura 8: Oscilador

4.3.2.3 Amplificador

O circuito amplificador foi proposto pelo supervisor da empresa. É um amplificador de corrente com ganho de tensão unitário.

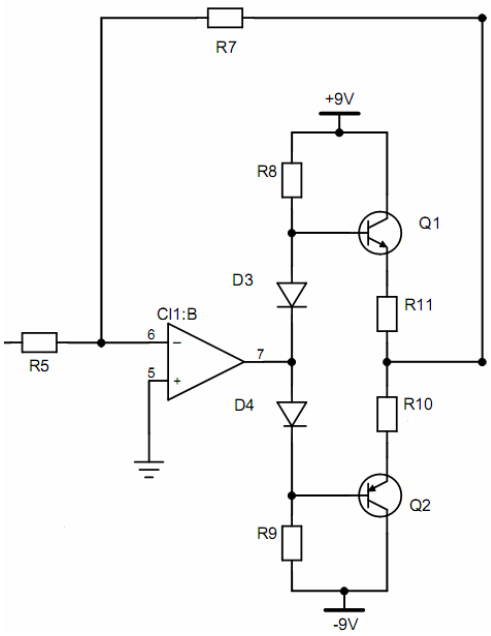


Figura 9: Amplificador

4.3.2.4 Detector de Pico

O circuito do detector de pico consiste em dois retificadores de pico de precisão (superdiodos) contendo capacitores na saída para manter a tensão de saída constante.

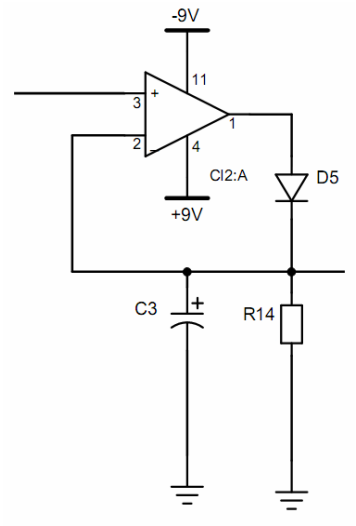


Figura 10: Detector de pico

4.3.2.5 Comparador

As tensões dos detectores de pico são comparadas utilizando amplificadores operacionais e definem o estado dos leds.

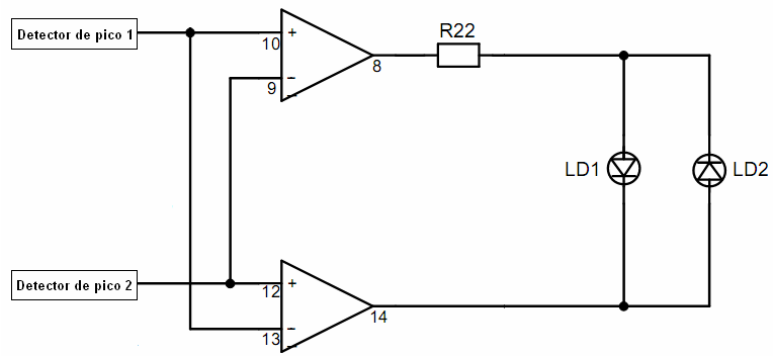


Figura 11: Comparador

4.3.3 Placa de circuito impresso

Após os testes em "protoboard", foi projetada a placa de circuito impresso utilizando o programa PCAD 2000. Sendo o equipamento portátil, o tamanho adequado da placa deve ser em torno de 15x4 cm. A placa construída possui 15,2cm de comprimento e 3,4cm de largura. Foi utilizado o plotter Protomat-S42 da LPKF para a confecção da placa. A soldagem foi realizada por técnicos da empresa.

Figura 12: Protomat-S42 da LPKF

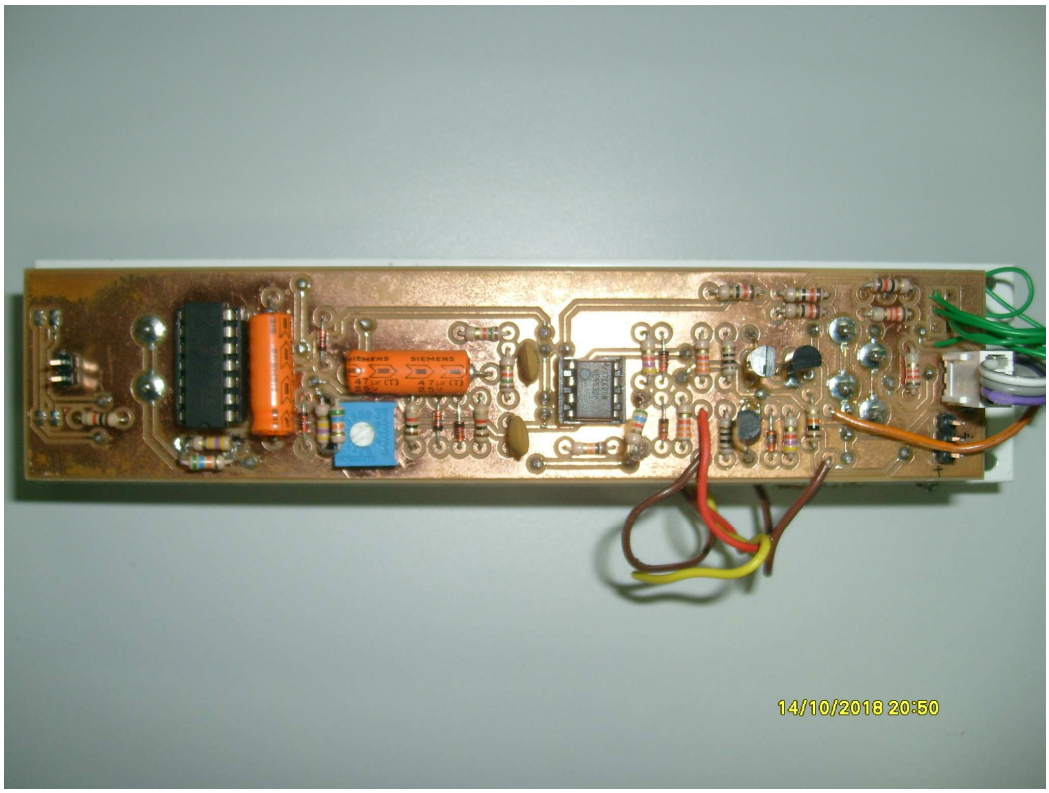


Figura 13: Placa do medidor após soldagem dos componentes

4.3.4 Testes

Após montar a placa, foram feitos testes com resistências conhecidas para calibrar o medidor. As resistências Rd e Ru foram substituídas por trimpots a fim de obter um ajuste perfeito da escala.

4.4 Considerações Finais

O projeto do compartimento externo do medidor, bem como sua confecção, foi elaborado por técnicos especializados da empresa. A maior dificuldade do projeto foi a calibração da escala, pois exigiu certo cuidado ao modificar o valor das resistências Rd e Ru.

Como o medidor opera na frequência de 1 kHz, é necessário conhecer o valor da impedância a ser medida nessa frequência. Nos altofalantes essa informação é conhecida.

5. Segunda atividade

A segunda atividade realizada foi o estudo dos Microcontroladores da série LPC21XX da NXP, que será utilizado na terceira atividade para uso do processador de áudio DDX8001.

O dispositivo utilizado foi o LPC2148 que são microcontroladores de 32/16 bits baseados na CPU ARM7TDMI-S. Devido ao seu pequeno tamanho, baixo consumo, alta velocidade e grande número de periféricos, esses microcontroladores tornaram-se essenciais, principalmente quando o tamanho e a alta capacidade de processamento são requisitos básicos. Suas principais aplicações são em controle industrial, sistemas médicos, controle de acesso, modem embarcado, gateways de comunicação e aplicações gerais. Neste capítulo serão introduzidos os conceitos básicos sobre os processadores ARM, seguido das principais características dos microcontroladores LPC2148.

5.1 O processador ARM

O primeiro processador ARM foi desenvolvido pela Arcorn Computers Limited em Cambridge, Inglaterra, entre Outubro de 1983 e abril de 1985. Naquele tempo, antes da formação da Advanced RISC Machine Limited (ARM Limited) em 1990, o ARM era conhecido como Arcorn RISC Machine. Atualmente o processador ARM é utilizado em diversos sistemas embarcados bem sucedidos de 32 bits como celulares, PDAS e videogames.

O processador ARM utiliza a arquitetura RISC (Reduced Instruction Set Computer), a qual é uma filosofia de design baseada em um conjunto simples e pequeno de instruções que levam aproximadamente a mesma quantidade de tempo para serem executadas. A arquitetura RISC possui as seguintes características:

- Conjunto reduzido e simples de instruções;
- Formatos simples e regulares de instruções;
- Operandos sempre em registros;
- Modos simples de endereçamento à memória;
- Uma operação elementar por ciclo máquina;
- Uso de pipeline;

5.1.1 O Processador ARM7TDMI-S

O processador ARM7TDMI-S é um membro da família ARM de microprocessadores para aplicações gerais. Sua arquitetura é baseada nos princípios RISC.

5.1.1.1 Pipeline

Pipeline é um mecanismo utilizado pelos processadores RISC para executar instruções. O ARM7TDMI-S utiliza um pipeline para aumentar a velocidade do fluxo de instruções para o processador. Com isso, diversas operações podem ocorrer simultaneamente. Esse processador utiliza um pipeline de três estágios. Os estágios de execução de uma instrução correspondem a:

- Busca: a instrução é carregada da memória;
- Decodificação: identifica a instrução a ser executada;

- Execução: processa a instrução e escreve o resultado em um registrador;

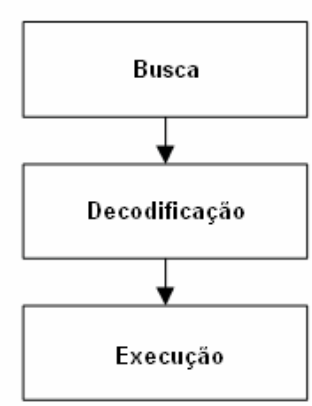


Figura 14: Pipeline em três estágios

Em operação normal, enquanto uma instrução está sendo executada, a próxima está sendo decodificada e uma terceira instrução está sendo procurada da memória.

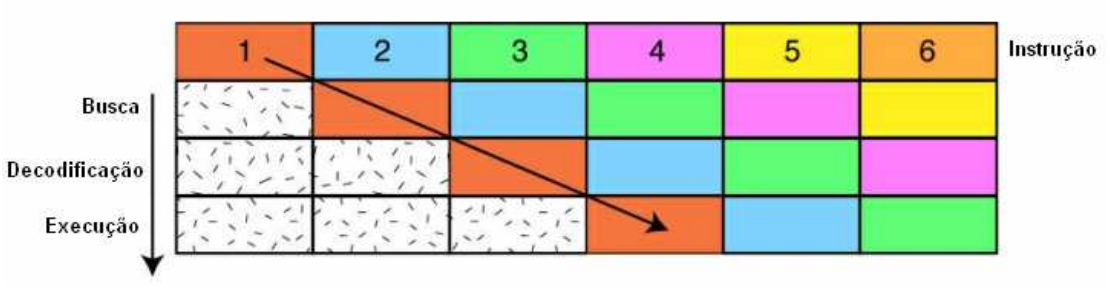


Figura 15: Seqüência de estágios do pipeline do ARM7

5.1.1.2 Acesso à memória

O processador ARM7TDMI-S possui arquitetura Von-Neumann, com um único barramento de 32 bits transportando instruções e dados, diferente da arquitetura Harvard que possui barramentos diferentes para dados e instruções. Apenas as instruções de carregamento, armazenamento e "swap" podem acessar dados da memória. Os dados podem ser de 8, 16 ou 32 bits.

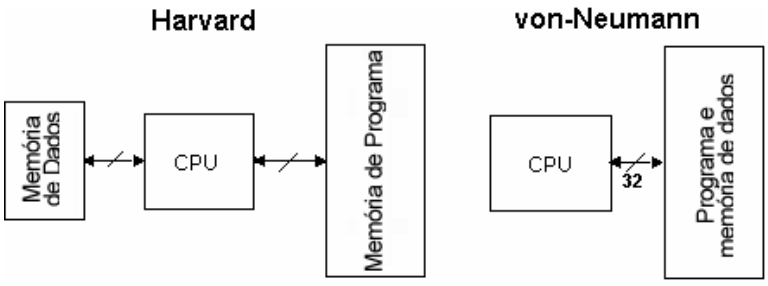


Figura 16: Arquiteturas Harvard e von-Neumann

5.1.1.4 Registradores

Os registradores de propósito geral podem armazenar dados ou endereços. Eles são identificados pelo prefixo "r" seguidos do número do registrador. O conjunto central de registradores possui um banco de 16 registradores (R0-R15) de 32 bits. Os

registradores R0-R12 não possuem função especial, já o R13, R14 e R15 desempenham funções especiais na CPU. O R13 é usado como apontador da pilha (“Stack Pointer” – SP). O R14 é chamando de “link register” (LR).

Quando uma chamada de uma função é feita o endereço de retorno é automaticamente salvo em R14 e está imediatamente disponível no retorno da função. Se a função faz parte de um ramo (branch), ou seja, se ela vai chamar outras funções, o valor de R14 deve ser preservado na pilha (R13). O R15 é o contador de programa (Program Counter – PC).

Figura 17: Registradores

5.1.1.5 Registrador de Status

O registrador de status (Current Program Status Register – CPSR) é um registrador adicional de 32 bits que contém os “flags” que monitoram e controlam as operações da CPU.

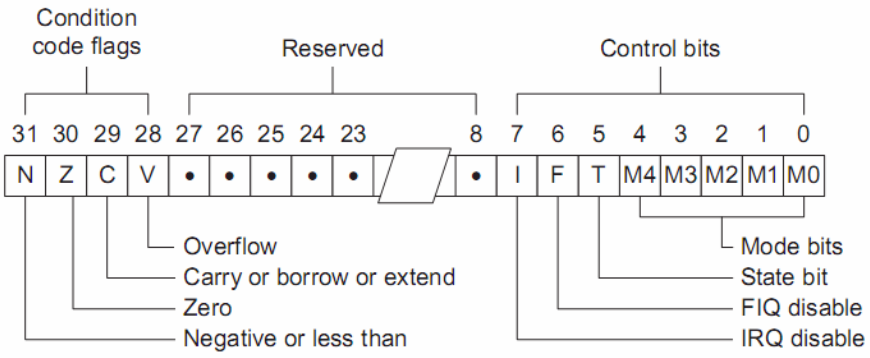


Figura 18: Registrador de status - CPSR

O CPSR é dividido em três partes de 8bits cada: flags de condição; espaço reservado; e bits de controle. Os flags de condição monitoram os resultados do processamento de dados, através deles é possível saber se o resultado de uma operação gerou um resultado negativo (N), zero (Z), vai - um (C) ou estouro (V). Os oito bits menos significativos de controle devem ser habilitados através da programação. Os bits 7 (I) e 8 (F) são usados para habilitar ou desabilitar modos de interrupção externa da CPU. Todos os periféricos do LPC2148 são conectados a essas duas linhas de

interrupção. Os bits 4 à 0 possuem o modo de operação do processador. O bit 5 é o bit THUMB. O ARM é capaz de executar dois conjuntos de instruções diferentes. O conjunto de instruções ARM de 32 bits e o THUMB de 16 bits. O bit 5 do CPSR indica qual conjunto de instruções está sendo executado.

5.1.1.6 Modos de Operação

O ARM7 possui sete modos de operação diferentes: *user*, *abort, fast interrupt request, interrupt request, supervisor, system e undefined*. Normalmente o controlador funciona no modo de usuário (user mode) com acesso ao banco de registradores R0-R15 e CPSR. Quando uma exceção ocorre, como uma interrupção ou um erro de memória, o processador mudará o modo de operação. Ocorrido uma exceção, os registradores R0-R12 e R15 continuarão os mesmos, mas R13 (LR) e R14 (SP) são substituídos por um novo par de registradores únicos para cada modo de operação.

O processador entra no modo *abort* quando há uma falha ao acessar a memória. Os modos *fast interrupt request* e *interrupt request* correspondem a dois níveis de interrupção disponíveis no processador. O modo *supervisor* é o modo que o processador se encontra após ser resetado, corresponde ao modo que geralmente um kernel de sistema operacional opera. No modo *system* podemos ler e escrever no CPSR, o qual corresponde a uma versão especial do modo *user*. Quando o processador encontra uma instrução indefinida ou não suportada ele entra no modo *undefined*. O modo *user* é usado por programas e aplicações. Todos os modos de operação, com exceção do modo user, possuem um registrador adicional, o SPSR (Saved Program Status Register), que armazena o valor do CPSR quando ocorre uma exceção. O modo FIQ (*fast interrup request*) possui os registradores R8-R12 duplicados, sendo possível entrar no modo de interrupção FIQ rapidamente, sem a necessidade de preservar os registradores na pilha.

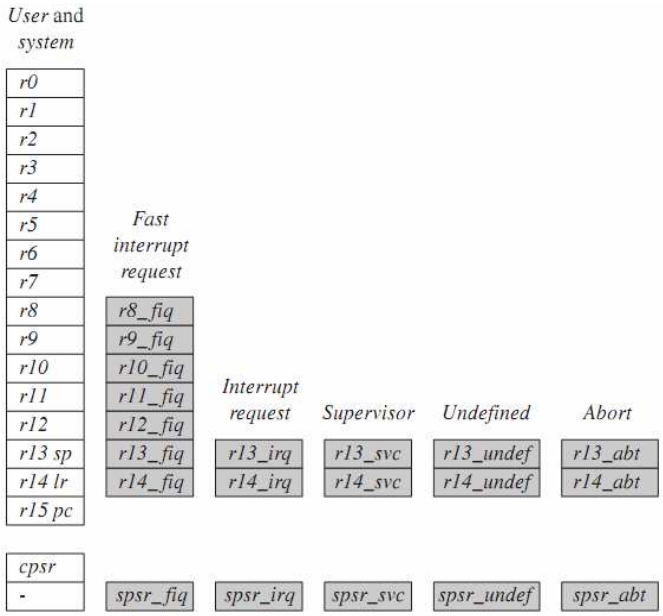


Figura 19: Banco de registradores ARM7

5.1.1.7 Instruções

A CPU ARM7 possui dois conjuntos diferentes de instruções: as instruções ARM de 32 bits e as instruções THUMB de 16 bits. O processador ARM7 foi projetado para operar como *big-endian* ou *litle-endian*. Os microcontroladores LPC2148 são *litle-endian*, ou seja, os bits mais significativos são os de maior ordem.

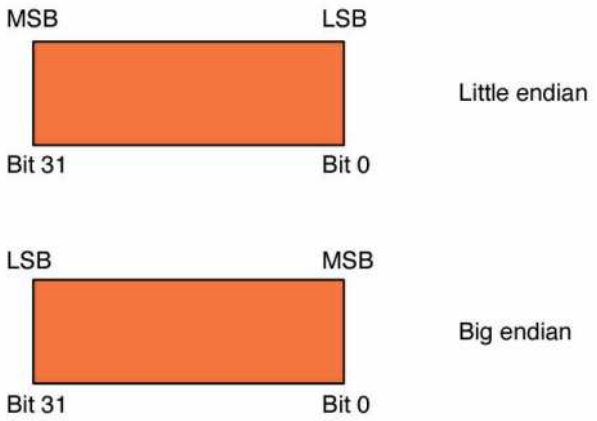


Figura 20: Litle-endian e big-endian

Cada instrução no conjunto ARM pode ser executada condicionalmente. Os quatro bits de maior ordem do operando são comparados com os flags do CPSR, se diferentes, a instrução não é executada e passa pelo pipeline como uma instrução NOP (No Operation). No modo THUMB apenas a instrução *branch* pode ser executada condicionalmente. O atributo de condição é pos-fixado ao mnemônico de instrução. Se nenhuma condição é especificada a condição "AL" é executada. A tabela abaixo lista os mnemônicos de execução condicional.

Mnemonic	Name	Condition flags
EQ	equal	Z
NE	not equal	z
CS HS	carry set/unsigned higher or same	C
CC LO	carry clear/unsigned lower	c
MI	minus/negative	N
PL	plus/positive or zero	n
VS	overflow	V
VC	no overflow	v
HI	unsigned higher	zC
LS	unsigned lower or same	Z or c
GE	signed greater than or equal	NV or nv
LT	signed less than	Nv or nV
GT	signed greater than	NzV or nzv
LE	signed less than or equal	Z or Nv or nV
AL	always (unconditional)	ignored

Figura 21: Mnemônicos de execução condicional

As instruções ARM podem ser divididas em seis categorias: *Branching*, Processamento de dados, Transferência de dados, transferência de blocos, interrupção por software e múltipla.

Apesar do ARM7 ser um processador de 32 bits, ele possui um conjunto de instruções de 16 bits chamado THUMB. O conjunto de instruções THUMB é uma forma comprimida do conjunto ARM. O THUMB possui maior densidade de código (espaço da memória utilizado por um programa executável) e sua implementação reduz em torno de 30% a memória que uma implementação ARM. Em compensação, a implementação em ARM é 40% mais rápida que a THUMB.

No modo THUMB não se tem acesso a todos os registradores. Apenas os conhecidos como "low registers" (R0-R7) podem ser completamente acessados. Os registradores R8-R12 só podem ser acessados com as instruções MOV, ADD e CMP. O CPSR só pode ser modificado indiretamente no modo THUMB.

5.1.1.8 Família de processadores ARM

A arquitetura ARM continua evoluindo desde que seu primeiro processador foi introduzido em 1985. Os processadores ARM estão divididos em famílias baseadas nos núcleos: ARM7, ARM9, ARM10 e ARM11. As tabelas abaixo trazem uma comparação das diferentes famílias de processadores ARM.

	ARM7	ARM9	ARM10	ARM11
Pipeline depth	three-stage	five-stage	six-stage	eight-stage
Typical MHz	80	150	260	335
mW/MHz[a]	0.06 mW/MHz	0.19 mW/MHz (+ cache)	0.5 mW/MHz (+ cache)	0.4 mW/MHz (+ cache)
MIPS[b]/MHz	0.97	1.1	1.3	1.2
Architecture	Von Neumann	Harvard	Harvard	Harvard
Multiplier	8 × 32	8 × 32	16 × 32	16 × 32

[a] Watts/MHz on the same 0.13 micron process.
[b] MIPS are Dhrystone VAX MIPS.

Figura 22: Comparação entre a família ARM

Esta seção visou introduzir os conceitos básicos sobre os processadores ARM, para maiores informações consultar a bibliografia.

5.2 O LPC2148

O LPC2148 é um microcontrolador da NXP Semiconductors de 32/16 bits baseado na CPU ARM7TDMI-S. Possui memória flash de 512kB, 40kB de memória RAM e o modo THUMB para aplicações nas quais o tamanho do código for crítico. Seu tamanho reduzido o torna ideal para aplicações onde o tamanho é importante. Suas principais características são:

- Encapsulamento LQFP64;
- 40kB de memória RAM (on-chip) e memória de programa flash de 512kB;
- Operação em até 60Mhz;
- Programação In-System/In-Aplication (ISP/IAP);
- Depuração em tempo real através das interfaces EmbeddedICE RT e Embedded Trace;
- Módulo USB 2.0 (Full Speed) com 2kb de RAM;
- Dois conversores A/D com um total de 14 entradas analógicas;
- Módulo Conversor D/A;
- Dois timer/contadores de 32 bits, com quatro canais de captura e comparação cada;
- Módulo PWM com seis saídas;
- Watchdog;
- RTC de baixo consumo com fonte independente e entrada de 32 khz dedicada;
- Múltiplas interfaces de comunicação serial, incluindo duas UARTs, dois Fast I2C (400kbits/s), SPI e SSP;
- Controlador de interrupções vetorado com prioridades e endereços configuráveis;
- 45 pinos de entrada e saída tolerantes a 5v;

- 21 pinos de interrupções externas disponíveis;
- Wake-up do processador via interrupção externa, USB, Brown-Out Detect ou RTC;
- Modo Power Saving incluindo Idle e Power-down;
- Operação entre 3.0V e 3.6V;

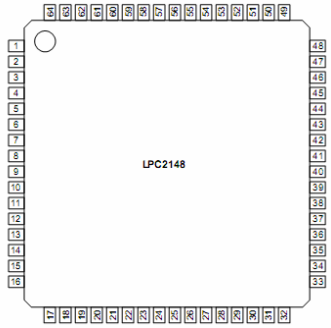


Figura 23: Encapsulamento LQFP 64

A figura 24 mostra o diagrama de blocos do LPC2148:

Devido ao tempo limitado, o estudo do LPC2148 foi direcionado para o uso do processador de áudio DDX8001. Os módulos estudados foram: Sistema de Controle, Interrupções, Entrada e saída, I^2C e Timers.

5.2.1 Sistema de controle

O bloco do sistema de controle inclui diversas funções e registradores de controle para funções que não estão relacionadas especificamente com os periféricos, são elas:

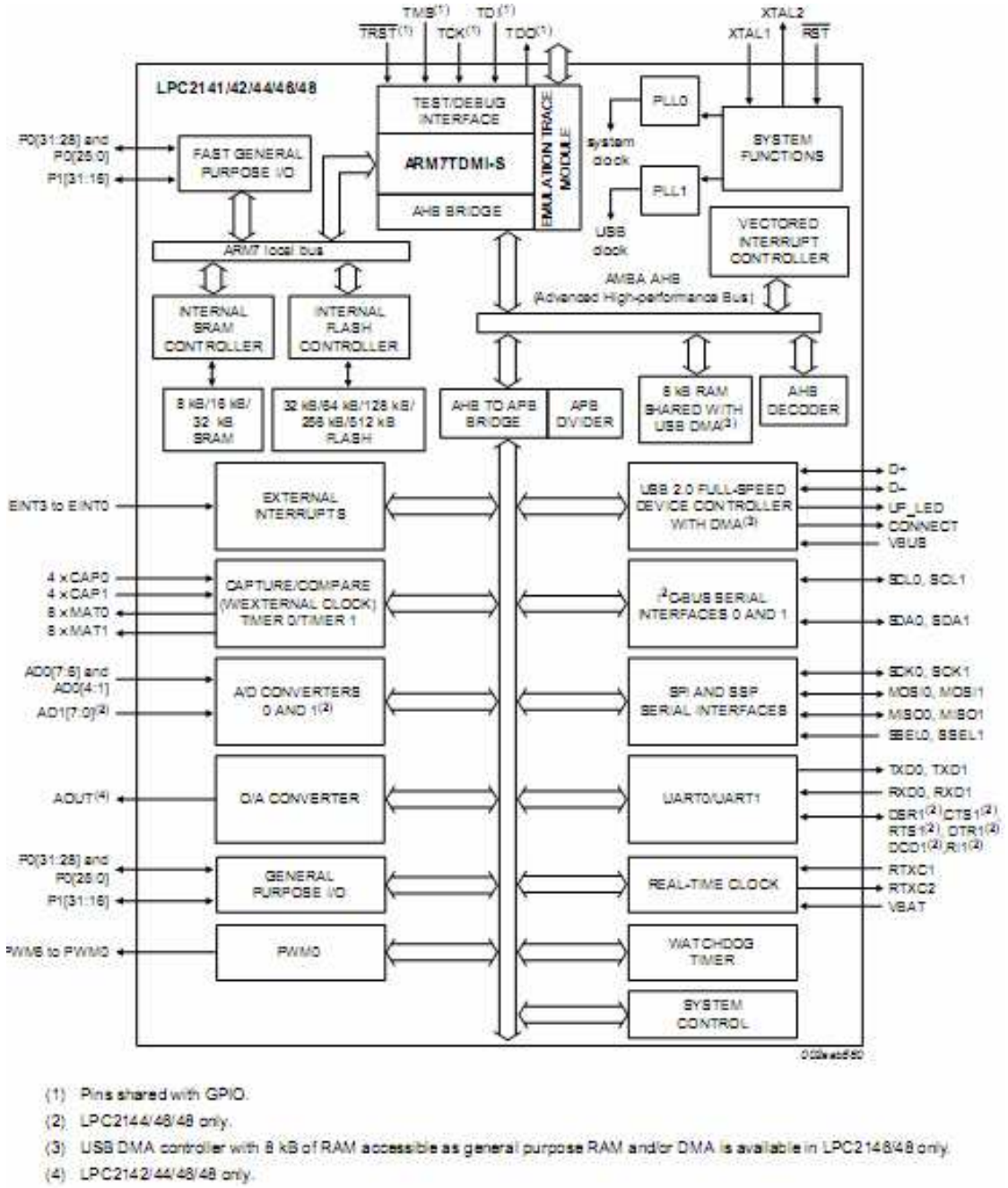
- Oscilador;
- Entradas de Interrupção externa;
- PLL;
- Power Control;
- Reset;
- APB Divider;
- Wake-up Timer;
- Sistemas de controle e status;
- Memory Mapping Control;

5.2.1.1 Oscilador

O oscilador integrado (on-chip) do LPC2148 opera com um cristal externo de 10 MHz a 25 MHz. Caso o módulo PLL não esteja conectado, a frequência do oscilador (fosc) é igual à frequência de clock do processador (CCLK).

5.2.1.2 PLL

Existem dois módulo PLL (Phase Locked Loop) no LPC2148. O PLL0 é usado para gerar o clock do sistema (CCLK) enquanto o PLL1 é usado para suprir um clock de 48 MHz ao módulo USB. Estruturalmente os PLL0 e PLL1 são idênticos, porém o PLL0 é capaz de gerar interrupções.



(1) Pins shared with GPIO.
(2) LPC2144/48/49 only.
(3) USB DMA controller with 8 kB of RAM accessible as general purpose RAM and/or DMA is available in LPC2148/48 only.
(4) LPC2142/44/48/48 only.

Figura 24: Diagrama de blocos do LPC2148

O PLL aceita um clock de entrada na frequência de 10 MHz a 25 MHz. A frequência de entrada é multiplicada até uma faixa entre 10 MHz e 60 MHz para o CCLK e até 48 MHz para o módulo USB, utilizando o CCO (Current Controlled Oscillators). O multiplicador pode ser um inteiro na faixa entre 1 e 32 (para o LPC2148 o multiplicador não pode ser maior que 6 devido ao limite de frequência da CPU). O CCO opera na faixa de 156 MHz a 320 MHz, portanto existe um divisor adicional a fim de manter o CCO na sua faixa de frequência enquanto o PLL está estabelecendo sua frequência requerida. O divisor de saída pode ser escolhido entre os valores 2, 4, 8 ou 16. O diagrama de blocos do PLL é mostrado abaixo:

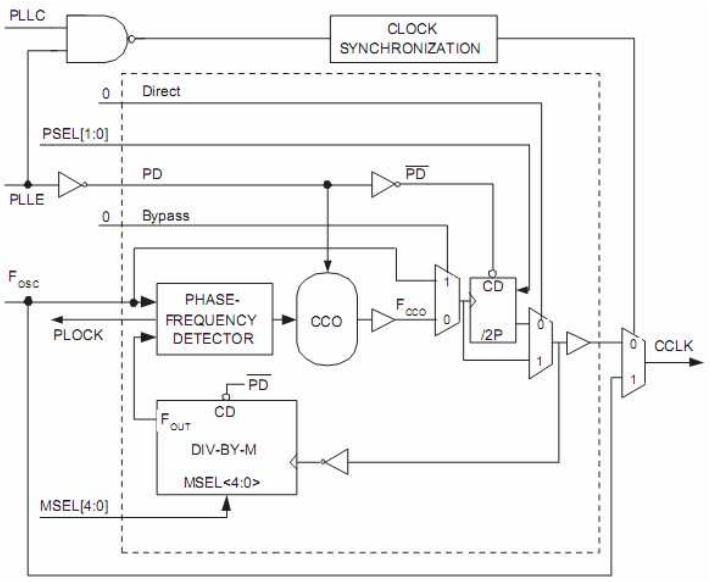


Figura 25: Diagrama de blocos do PLL

A frequência do PLL pode ser obtida pelas seguintes equações:

$$CCLK = MxFosc \quad (5.1)$$

$$CCLK = Fcco / (2xP) \quad (5.2)$$

$$Fcco = CCLKx2xP \quad (5.3)$$

$$Fcco = FoscxMx2xP \quad (5.4)$$

onde:

- Fosc: frequência do cristal externo;
- Fcco: frequência do CCO;
- CCLK: frequência de saída do PLL (clock do sistema);
- M: multiplicador do PLL;
- P: Divisor do PLL;

5.2.1.3 Reset e Wake-up timer

O LPC2148 possui duas fontes de reset, o pino RESET e o reset do watchdog. Caso ocorra um reset do chip por qualquer uma das fontes, o Wake-up Timer é ativado até que a fonte de reset externo seja desabilitada, o oscilador esteja funcionando e um número fixo de clocks tenha passado, em seguida o controlador completa sua inicialização.

O Wake-up Timer assegura que o oscilador e outras funções essenciais para o funcionamento do chip estejam habilitadas antes do processador poder executar instruções.

5.2.1.4 Power Control

Dois modos são suportados pelo LPC2148: Idle mode e Power-Down mode.

No modo Idle a execução de instruções é suspensa até que ocorra um reset ou uma interrupção. As funções dos periféricos continuam em operação e devem gerar interrupções para que o processador retorne ao seu estado normal. O modo Idle elimina energia gasta pelo processador, por sistemas de memória e controladores relacionados, e pelos barramentos internos.

No modo Power-Down o oscilador é desligado e o chip não recebe clocks internos. O estado do processador, dos registradores e da RAM interna são preservados, assim como os estados lógicos dos pinos de saída continuam estáticos. A operação em modo normal pode ser recuperada por reset ou interrupções específicas. O modo Power-Down reduz o consumo do chip para próximo de zero.

5.2.1.5 Entradas de interrupção externa

O LPC2148 possui entradas de interrupção externa de até nove bordas ou sensível ao nível. Quando os pinos são combinados podem ser processados como quatro sinais de interrupção independentes. As entradas de interrupção externa podem ser usadas para retirar o processador do modo de operação Power-down.

5.2.1.6 APB divider

O APB divider determina a relação entre o clock do processador (CCLK) e o clock usado pelos dispositivos periféricos (PCLK). Ele possui duas funções.

A primeira função é prover o clock desejado aos periféricos através do barramento APB,a fim de permitir a operaração na frequência escolhida pelo processador. A segunda função é prover economia de energia quando a aplicação não necessita de nenhum periférico. A conexão do APB divider é mostrada na figura abaixo.

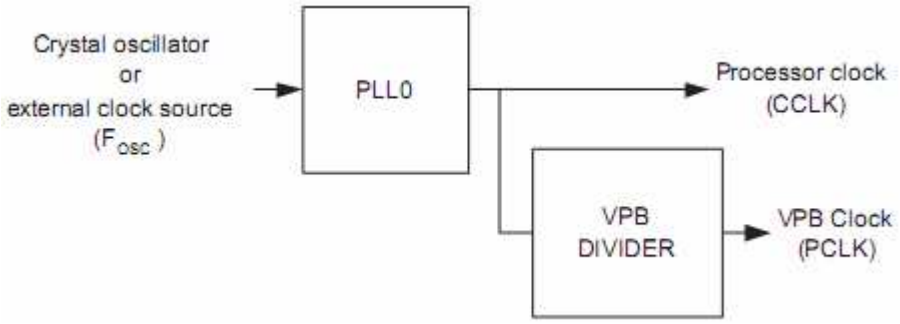


Figura 26: Conexões do APB divider

5.2.1.7 Memory Mapping Control (MMC)

O MMC altera o mapeamento do vetor de interrupções que aparece iniciando no endereço 0x0000 0000. Isso habilita a execução de código em diferentes espaços de memória para ter controle das interrupções.

5.2.2 Entrada e Saída

O LPC2148 possui duas portas de 32 bits cada para funções de entrada e saída (GPIO). Um total de 30 entradas/saídas e um pino de saída simples estão disponíveis nos 32 pinos da PORT0 (os pinos do microcontrolador são divididos em dois grupos,

PORT0 e PORT1, cada um com 32 pinos). A PORT1 possui 16 pinos disponíveis para GPIO. A PORT0 e a PORT1 são controlados por quatro grupos de registradores:

- IODIR (GPIO Port Direction control Register): Esse registrador controla individualmente a direção de cada pino da porta;
- IOSET (GPIO Port Output Set Register): Esse registrador controla o estado dos pinos de saída em conjunto com o registrador IOCLR. Escrever 1 nesse registrador produz nível lógico zero no pino de saída correspondente. Escrever zero não há efeito;
- IOCLR (GPIO Port Output Clear Register): Escrever 1 nesse registrador produz nível lógico zero nos pinos de saída correspondentes. Escrever zero não há efeito;
- IOPIN (GPIO Port Pin Value Register): Esse registrador retorna ao estado dos pinos das portas, independente da direção (entra ou saída).

Para diferenciar entre as PORTs 0 e 1, o número correspondente aparece logo após o prefixo "IO", ou seja, para os pinos da PORT0 temos: IO0DIR, IO0SET, IO0CLR e IO0PIN. Abaixo temos um exemplo de uso desses registradores:

```
IO0DIR = 0X0000 0080;     pino P0.7 é configurado como saída;
IO0CLR = 0X0000 0080;     pino P0.7 em nível lógico 0;
IO0SET = 0X0000 0080;     pino P0.7 em nível lógico 1;
IO0CLR = 0X0000 0080;     pino P0.7 em nível lógico 0;
```

Para maiores informações sobre a descrição dos pinos do LPC2148 consultar o datasheet.

5.2.3 I^2C

O I^2C é um protocolo de comunicação serial desenvolvido pela Philips que é usado para conectar periféricos de baixa velocidade a: dispositivos como placa mãe, a um sistema embarcado ou a um telefone celular. Existem três tipos de protocolo I^2C de acordo com a velocidade de transmissão:

- 100 kbps (Standard mode);
- 400 kbps (fast mode);
- 3.4 Mbps (high-speed mode).

O protocolo I2C utiliza apenas dois fios para comunicação: o de clock (SCL) e o de dados (SDA). Cada dispositivo no barramento é reconhecido por um único endereço e pode atuar como transmissor e/ou receptor, mestre ou escravo. O I^2C é multimaster, ou seja, o barramento pode ser controlado por mais de um mestre.

O LPC2148 possui duas interfaces para comunicação via I^2C: I^2C0 e I^2C1. Uma configuração típica do barramento I^2C é mostrada na figura abaixo:

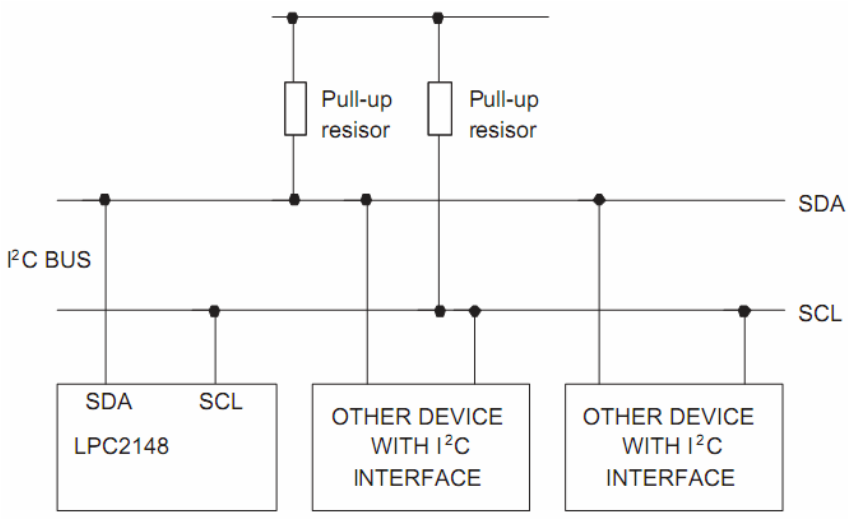


Figura 27: Barramento I^2C

O módulo I^2C no LPC possui as seguintes características:

- Compatível com a interface I^2C padrão;
- Transmissão em até 400kbits/s (Fast I^2C);
- Fácil configuração dos modos máster, slave ou máster/slave;
- Clocks programáveis;
- Multi-master (não existe um único mestre);
- Transferência de dados bidirecional entre mestres e escravos;
- A sincronização serial do clock permite a comunicação de dispositivos com diferentes taxas de transmissão por um único barramento;

A interface I2C no LPC2148 é *byte oriented* e possui quatro modos de operação:

- Master Transmitter Mode: Nesse modo os dados são transmitidos de um mestre para um escravo. O primeiro byte transferido contém o endereço do escravo que irá receber os dados e um bit de direção de dados no valor 0 (escrita). Os dados são transmitidos em 8 bits por vez. Após cada byte ser transmitidos um bit *acknowledge* é recebido. As condições START e STOP indicam respectivamente o início e o fim da transmissão.

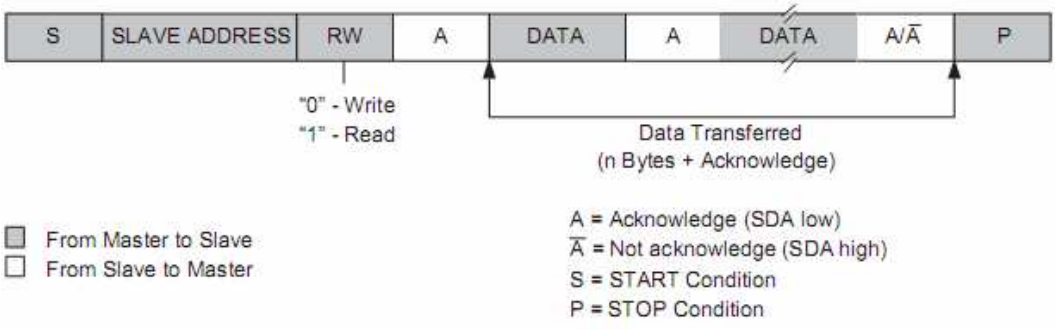


Figura 28: Formato da mensagem para o modo Master Transmitter

- Master Receiver Mode: Nesse modo os dados enviados por um escravo são recebidos pelo mestre. A transferência é iniciada da mesma forma da anterior, a

diferença é que o bit de direção tem o valor 1 (leitura). Após a transferência, sua repetição da condição START altera o modo para o Master Transmitter.

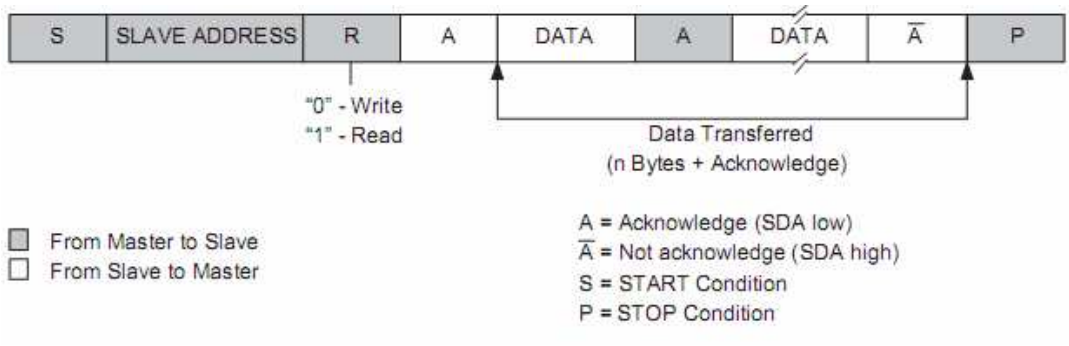


Figura 29: Formato da mensagem para o modo Master Receiver

- Slave Receiver Mode: Nesse modo os dados recebidos são enviados por um mestre. Após a inicialização do módulo, o escravo espera pelo seu endereço seguido do bit de direção. Se o bit de direção foi 0, ele entra no modo Slave Receiver, se 1 entra no modo Slave Trasmitter.

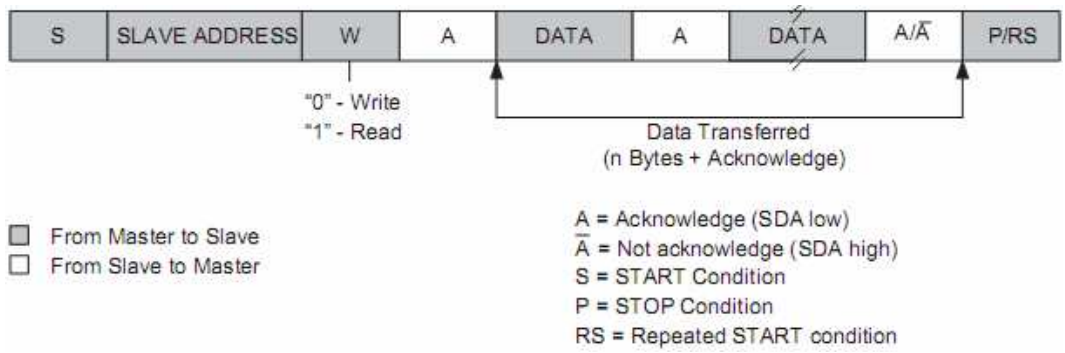


Figura 30: Formato da mensagem para o modo Slave Transmitter

- Slave Transmitter Mode: No modo Slave, quando o primeiro byte é recebido o dispositivo está operando no modo Slave Receive. Quando o bit de direção no valor 1 é recebido uma interrupção é requisitada e o microcontrolador tende a tornar-se o mestre do barramento. Após a transferência, o microcontrolador retorna ao modo escravo imediatamente.

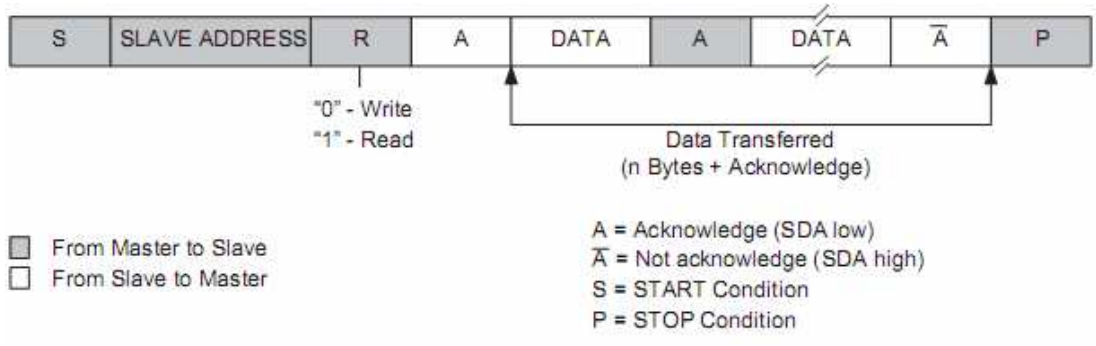


Figura 31: Formato da mensagem para o modo Slave Receiver

Para utilizar a interface I^2C0, os pinos P0.2 (SCL0) e P0.3 (SDA0) devem ser configurados através do registrador PINSEL0 (Pin Select 0) que configura a função que cada pino deve desempenhar. Para a interface I2C1 os pinos P0.11 (SCL1) e P0.15

(SDA1) devem ser configurados através do registrador PINSEL1. Se o valor do registrador PINSEL não for alterado, os pinos estão configurados como entrada/saída.

5.2.3.1 Registradores

Os principais registradores utilizados na interface I^2C são:

- I2CONSET: É o registrador de controle para escrita;
- I2STAT: É o registrador de status, durante a operação fornece as condições da transmissão possibilitando ao software determinar a próxima ação;
- I2DAT: É o registrador de dados. Os dados recebidos ou transmitidos são armazenados nesse registrador;
- I2ADR: É o registrador de endereços do escravo. Contem os sete bits de endereço do escravo para operação no modo Slave. No modo de operação Master, não é utilizado;
- I2SCLH: Armazena os bits mais significativos para determinação do clock do barramento;
- I2SCLL: Armazena os bits menos significativos para determinação do clock do barramento;
- I2CONCLR: É o registrador de controle para limpeza;

O valor dos registradores I2SCLH e I2SLL determina a velocidade do barramento. O valor desses registradores pode ser determinado pela fórmula:

$$I^2C_{bitfrequency} = \frac{PCLK}{I2CSCLH + I2CSCLL}$$

Figura 32: Fórmula para determinação da frequência do I2C.

Para maiores informações sobre a interface I^2C, registradores e modos de operação, consultar o datasheet do LPC2148.

5.2.4 Interrupções

Para tratar as interrupções, o LPC2148 possui um controlador dedicado chamado Controlador de Interrupção Vetorada (Vectored Interrupt Controller-VIC). O VIC aceita todos os pedidos de interrupção (32) e os categoriza como FIQ (Fast Interrupt Request), Vectored IRQ (Vectored Interrupt Request) ou non-Vectored IRQ. Com isso, podemos ajustar dinamicamente as prioridades das interrupções.

As interrupções FIQ são as de maior prioridade. A maior velocidade é obtida quando apenas uma interrupção é classificada como FIQ, se mais de uma for classificada como FIQ, ao ocorrer um pedido de interrupção, a rotina deverá identificar qual fonte a gerou.

As interrupções classificadas como Vectored IRQ possuem prioridade média, mas apenas 16 das 32 interrupções podem ser classificadas nesta categoria. Dessas, qualquer uma pode ser designada para qualquer uma das 16 posições do vetor IRQ. A posição 0 do vetor possui maior prioridade enquanto a 15 possui menor.

As interrupções IRQ não vetoradas possuem menor prioridade. Um pedido de interrupção FIQ pode ser atendido enquanto o microcontrolador estiver executando a rotina de interrupção IRQ, vetoradas ou não.

5.2.4.1 Registradores

Os registradores utilizados pelo VIC para tratamento das interrupções são:

- VICIRQStatus: É o registrador de status IRQ. Esse registrador lê a condição dos pedidos de interrupção habilitados e classificados como IRQ;
- VICFIQStatus: É o registrador de status FIQ. Esse registrador lê a condição dos pedidos de interrupção habilitados e classificados como FIQ;
- VICRawIntr: Lê o status dos 32 pedidos de interrupção/interrupção por software, independente do estado ou classificação;
- VICIntSelect: Classifica as interrupções como FIQ ou IRQ;
- VICIntEnable: Habilita as interrupções FIQ e/ou IRQ;
- VICIntEnClr: Limpa um ou mais bits do VICIntEnable;
- VICSoftInt: Com conteúdo desse registrador é realizada uma operação OR com os 32 pedidos de interrupção de diversos periféricos;
- VICSoftIntClear: Limpa um ou mais bits do VICSoftInt;
- VICProtection: Possibilita o limite de acesso aos registradres VIC;
- VICVectAddr: Quando uma interupção IRQ, a rotina de serviço IRQ lê o valor desse registrador e salta para o valor lido. Contém o endereço do vetor IRQ;
- VICDefVectAddr: Armazena o endereço da rotina de interrupção para interrupções não vetoradas;
- VICVectAddr0-15: Armazena o endereço da rotina de serviços da interrupção para as 16 posições do vetor de interrupção IRQ. Ou seja, VICVectAddr1 contém o endereço da posição 1, enquanto VICVectAddr15 o endereço da posição 15;
- VICVectCntl0-15: São os registradores de controle para cada uma das posições no vetor IRQ. A posição 0 tem maior prioridade e a 15 a menor.

Para maiores detalhes sobre os registradores VIC, consultar o datasheet do LPC2148.

5.2.5 Timers

Os dois timers/counters do LPC2148 (Timer0 e Timer1) foram projetados para contar ciclos do PCLK ou de um clock externo, podem gerar interrupções e desempenhar outras ações em intervalos de tempo específicos baseado nos registradores Match Registers. Eles também incluem quatro entradas de captura (Capture). Suas principais características são:

- Timer/counter de 32 bits com prescaler programável de 32bits;
- Operação como Timer ou Counter;
- Até quatro canais de captura de 32 bits por Timer.
- Quatro Match Registers de 32 bits que permitem reset, parada ou operação contínua, com geração opcional de interrupção quando há igualdade;
- Quatro saídas externas correspondendo aos Match Registers capazes de modificar o nível lógico quando há igualdade;

5.2.5.1 Registradores

Os registradores relacionados aos Timers/Counters são:

- IR: É o registrador de interrupções. Pode ser escrito para limpar interrupções e lido para identificar qual das 8 possíveis fontes gerou a interrupção;
- TCR: É o registrador de controle dos timers. É usado para controlar as funções dos Timers/Counters;
- TC: É o contador de tempo. É um registrador de 32 bits incrementado a cada PR+1 ciclos do PCLK. É controlador pelo TCR;
- PR: É o registrador de Prescale. Quando o PC (Prescale Counter) é igual ao PR, o próximo clock incremente o TC e limpa o valor de PC;
- PC: É o contador do Prescale. É um contador de 32 bits que é incrementado até o valor armazenado em PR. Ao atingir o valor em PR, TC é incrementado e PC reinicia a contagem. O PC é incrementado a cada PCLK;
- MCR: É usado para controlar se uma interrupção é gerada e se TC é resetado quando uma igualdade (Match) ocorre;
- MR0-3: São os Match Registers. Podem ser habilitados através do MCR para resetar o TC, parar o TC e o PC e/ou gerar interrupções quando o valor nesses registradores foram igual ao valor em TC;
- CCR: É o registrador de controle de captura. O CCR controla quais bordas das entradas serão usadas para carregar os registradores de captura e/ou quando uma interrupção é gerada;
- CR0-3: São os registradores de captura;
- EMR: Controle os pinos externos de Match;
- CTCR: Seleciona entre o modo Counter ou Timer. No modo Counter seleciona o sinal e as bordas para contagem.

Para diferenciar entre os registradores do Timer 0 e do Timer 1, deve-se prefixar os registradores com T0 para o Timer 0 e T1 para o Timer1, ou seja, o registrador TC do Timer 0 é o T0TC e o do Timer 1 é o T1TC. Para maiores detalhes sobre o módulo Timer/Counter consultar o datasheet do LPC2148.

5.3 O compilador

O compilador utilizado foi o µVision versão 3.40 da Keil. O compilador suporta microcontroladores ARM de diversos fabricantes e utiliza a linguagem de programação C. Possui funções de simulação e depuração em tempo real, suporte a um grande número de placas de desenvolvimento e aos principais dispositivos programadores do mercado.

Para programação do LPC2148 foi utilizado o N-Link da Micros4you. Esse dispositivo é compatível com o µLink da Keil que oferece depuração em tempo real via JTAG e suporte aos principais microcontroladores ARM. Abaixo temos uma foto do N-Link.

Figura 33: Programador N-Link

5.4 A placa de desenvolvimento

Para realizar os testes com o microcontrolador foi desenvolvida, utilizando o software PCAD2000, uma placa com as principais funções a serem utilizadas no controle do processador de áudio DDX8001. A placa foi projetada para ser utilizada com uma placa base desenvolvida na APEL para os projetos com os microcontroladores LPC2148. A placa base possui acesso aos pinos do microcontrolador, conector para programação e o conector de reset do sistema. A figura 34 mostra a placa base utilizada.

A placa projetada possui conectores de acesso aos pinos do microcontrolador, um conector para o LCD, conector de alimentação, conector para a interface I^2C e UART. As figuras 35 e 36 mostram a placa projetada e a placa de desenvolvimento pronta para uso. O layout da placa de desenvolvimentos encontra-se em anexo.

Figura 34: Placa base utilizada nos projetos com o LPC2148

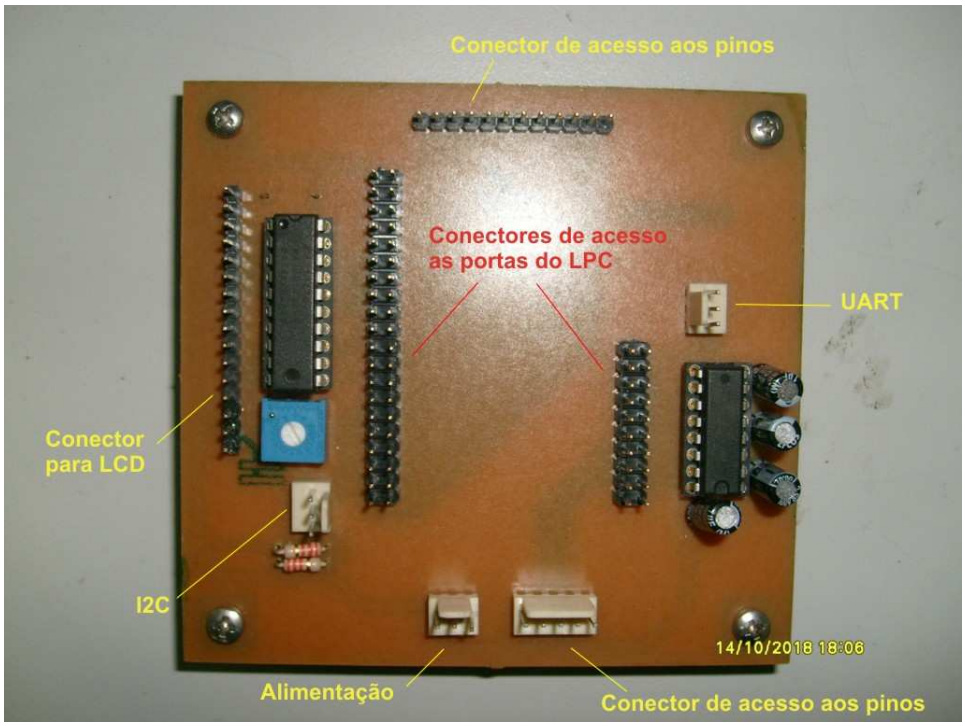


Figura 35: Funções da placa de desenvolvimento

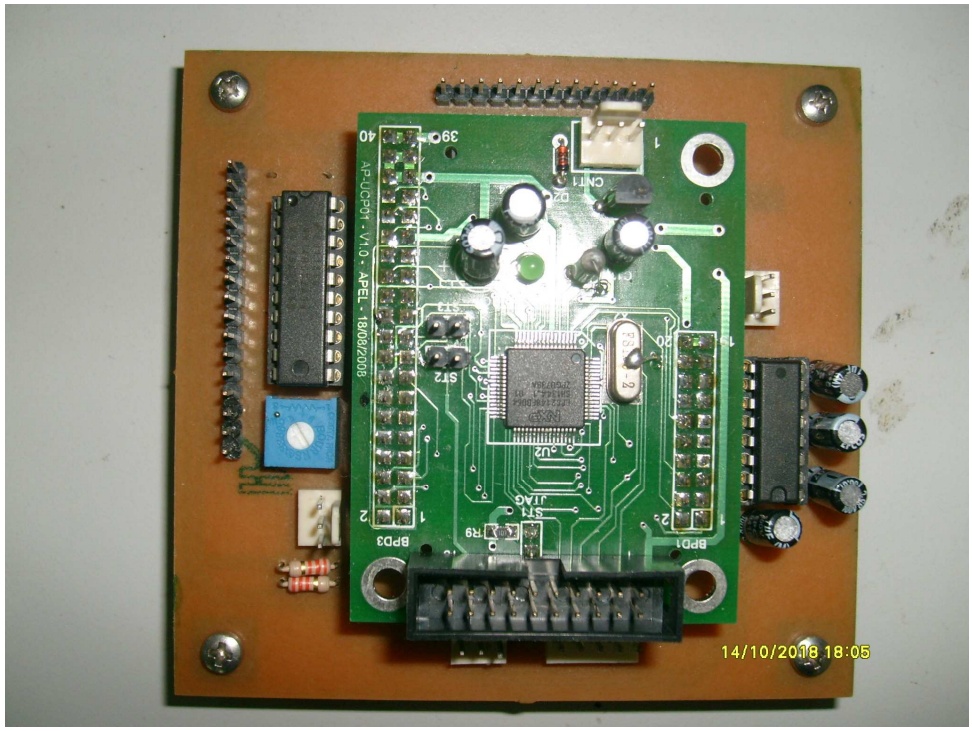


Figura 36: Placa de desenvolvimento

5.5 Considerações Finais

Testes de todos os módulos descritos foram realizados visando ganhar maior intimidade com o microcontrolador e o compilador antes de iniciar a terceira atividade.

6. Terceira atividade

Após o estudo das funções básicas do LPC2148, o microcontrolador foi utilizado para o controle do processador de áudio DDX-8001.

Entre as diversas funções do DDX, foram enfatizadas as de controle de volume, de equalização e de filtros. Após o seu estudo, uma rotina foi desenvolvida para implementar as funções de controle de volume e de equalização.

6.1 O DDX-8001

O DDX-8001 é um processador de áudio digital que possui diversas funções de controle como: controle de volume e mudo, filtros, equalizadores predefinidos, funções para roteamento de canais, etc. O DDX pode ser controlado por qualquer dispositivo que possua interface I^2C. O controle consiste em modificar o valor dos registradores específicos para cada função. As principais características do DDX-8001 são:

- 8 canais de entradas e saídas seriais;
- 6 canais de entrada DSD/SACD;
- 8 canais de saída DDX (PWM);
- Controle via I^2C;
- 10 equalizadores de 32 bits programáveis por canal;
- Controle de volume e mute;
- Entrada e saída serial de dados via I^2S;
- Operação em 3.3 V;
- Processamento interno de ganho/atenuação;
- Filtros definidos pelo usuário;

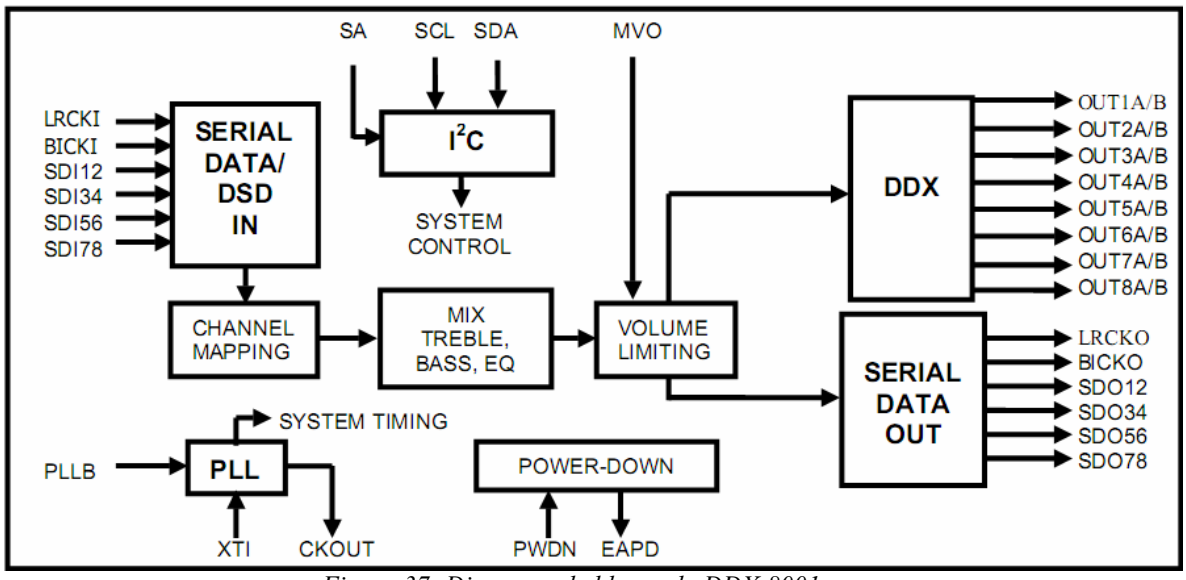


Figura 37: Diagrama de blocos do DDX-8001

6.1.1 Registradores

O DDX possui um conjunto de 84 registradores endereçados entre 0x00 e 0x83. A configuração desses registradores modifica a função desempenhada por cada um.

Para a inicialização do DDX, os Registradores de Configuração devem ser configurados corretamente de acordo com a taxa de amostragem do conversor A/D de entrada. Em nosso caso, a placa de desenvolvimento opera na taxa de amostragem de 96 kHz. A seqüência de inicialização consiste em carregar no DDX o valor nos registradores segundo a seqüência mostrada na tabela baixo:

REGISTRADOR	ENDEREÇO	VALOR
Configuration Register A	0x00	0x89
Configuration Register B	0x01	0x01
Configuration Register C	0x02	0x00
Configuration Register D	0x03	0xFE
Configuration Register E	0x04	0xFF
Configuration Register F	0x05	0x41
Configuration Register G	0x06	0x00
Configuration Register H	0x07	0x7E
Configuration Register I	0x08	0x00
Master Mute	0x09	0X01
Master Volume	0x0A	0x3F
Volume do Canal 0	0x0B	0x10
Volume do Canal 1	0x0C	0x10
Volume do Canal 2	0x0D	0x10
Volume do Canal 3	0x0E	0x10
Volume do Canal 4	0x0F	0x10
Volume do Canal 5	0x10	0x10
Volume do Canal 6	0x11	0x10
Volume do Canal 7	0x12	0x10
Mute do Canal 0	0x13	0x10
Mute do Canal 1	0x14	0x10
Mute do Canal 2	0x15	0x10
Mute do Canal 3	0x16	0x10
Mute do Canal 4	0x17	0x10
Mute do Canal 5	0x18	0x10
Mute do Canal 6	0x19	0x10
Mute do Canal 7	0x1A	0x10
Origem do Canal 0 e 1	0x1B	0x00
Origem do Canal 2 e 3	0x1C	0x00
Origem do Canal 4 e 5	0x1D	0x11
Origem do Canal 6 e 7	0x1E	0x11
Master Mute	0x09	0x00

Tabela 2: Seqüência de inicialização do DDX

A inicialização consiste em configurar os Registradores de Configuração de A a I, assim como, modificar o volume e as origens dos 8 canais antes do início o uso. No princípio da inicialização o Mute Geral estava habilitado e ao final ele foi desabilitado, caso haja sinal nas entradas, haverá sinais nas 8 saídas. Os canais 0, 1, 2 e 3 foram configurados para receber o sinal do canal esquerdo e os canais 4, 5, 6 e 7 recebem o sinal do canal direito.

6.1.1.1 Controle de volume e mute

Os registradores para controle de volume estão endereçados entre 0x09 e 0x1A. Os registradores Master Mute (0x09) e Master Volume Control (0x0A), controlam o mute e o volume de todos os canais simultaneamente. Para o controle de volume individual dos canais, devemos utilizar os registradores de 0x0B (para o Canal 0) ao 0x12 (para o canal 7). O controle de mute individual é realizado pelos registradores endereçados entre 0x13 e 0x1A.

Os registradores de controle de volume são de 8 bits. Para obtermos o máximo volume no canal, devemos carregar o registrador com 0x00. Consequentemente o valor 0xFF corresponde ao volume mínimo.

6.1.1.2 Equalizadores

O registrador de 8 bits endereçado em 0x22 é o registrador de equalizadores pré-definidos. Os cinco bits menos significativos deste registrador definem o tipo de equalizador utilizado. A tabela 3 traz as funções de equalização e o valor dos bits associados.

6.1.1.3 Filtros

Cada canal do DDX-8001 possui 10 filtros. Cada filtro pode ser configurado como passa-baixas, passa-alta ou paramétrico. O projeto dos filtros consiste no cálculo dos coeficientes, que depende do tipo de filtro escolhido. O DDX utiliza a seguinte equação para implementação dos filtros:

$$y[n] = 2 \cdot \{(b_0/a_0)/2\} \cdot x[n] + 2 \cdot \{(b_1/a_0)/2\} \cdot x[n-1] + \{(b_2/a_0) \cdot x[n-2]\}$$
$$+ 2 \cdot \{(-a_1/a_0)/2\} \cdot y[n-1] + \{(-a_2/a_0) \cdot y[n-2]\}$$

Figura 38: Equação dos filtros

Na equação acima, y[n] representa a saída e x[n] a entrada. Após o cálculo, para serem carregados, os coeficientes devem estar na forma de números inteiros de ponto fixo de 24 bits com sinal; devem estar na faixa [800000h, 7FFFFFh] ou [-1,1] em decimal.

O endereçamento dos registradores que irão receber os valores dos coeficientes dos filtros é diferente do endereçamento dos outros registradores do DDX. No DDX existem dois registradores que são os ponteiros para endereçar os coeficientes de cada filtro. Esses registradores estão endereçados em 0x3B e 0x3C e formam uma palavra de 10 bits. Nesses registradores deve ser carregado o endereço do filtro que se deseja carregar os cinco coeficientes de 24 bits. Em seguida deve-se carregar os registradores endereçados entre 0x3D e 0x4B com os valores dos coeficientes. Para finalizar, deve-se escrever 0x02 no registrador 0x4C, que é o registrador de Controle de Escrita de Coeficientes (Coefficient Write Control Register). A escrita nesse registrador atualiza o valor dos coeficientes em uma memória RAM (User Defined Coefficient RAM) responsável por armazenar os coeficientes de todos os filtros. Na RAM os filtros estão endereçados em espaços de cinco em cinco, iniciando pelo filtro 0 do canal 0 que possui o endereço 0x00. O endereço do filtro 1 do canal 0 é 0x05 e assim consecutivamente até

o décimo filtro do canal 7. A tabela 4 traz o endereço de todos os coeficientes para os filtros do DDX.

PEQ (3..0)	Setting
00000	Flat
00001	Rock
00010	Soft Rock
00011	Jazz
00100	Classical
00101	Dance
00110	Pop
00111	Soft
01000	Hard
01001	Party
01010	Vocal
01011	Hip-Hop
01100	Dialog
01101	Bass-Boost #1
01110	Bass-Boost #2
01111	Bass-Boost #3
10000	Loudness 1 (least boost)
10001	Loudness 2
10010	Loudness 3
10011	Loudness 4
10100	Loudness 5
10101	Loudness 6
10110	Loudness 7
10111	Loudness 8
11000	Loudness 9
11001	Loudness 10
11010	Loudness 11
11011	Loudness 12
11100	Loudness 13
11101	Loudness 14
11110	Loudness 15
11111	Loudness 16 (most boost)

Tabela 3: Equalizadores

Index (Decimal)	Index (Hex)		Coefficient	Default
0	00h	Channel 1 – Biquad 1	C1H10 (b1/2)	000000h
1	01h		C1H11 (b2)	000000h
2	02h		C1H12 (a1/2)	000000h
3	03h		C1H13 (a2)	000000h
4	04h		C1H14 (b0/2)	400000h
5	05h	Channel 1 – Biquad 2	C1H20	000000h
…	…	…	…	…
49	31h	Channel 1 – Biquad 10	C1HA4	400000h
50	32h	Channel 2 – Biquad 1	C2H10	000000h
51	33h		C2H11	000000h
…	…	…	…	…
99	63h	Channel 2 – Biquad 10	C2HA4	400000h
100	64h	Channel 3 – Biquad 1	C3H10	000000h
…	…	…	…	…
399	18Fh	Channel 8 – Biquad 10	C8HA4	400000h

Tabela 4: Endereçamento dos filtros

6.2 A placa de desenvolvimento

A placa de desenvolvimento utilizada foi a Proex v1.0 que possui dois canais de entrada (esquerdo e direito) e oito canais de saída. Para a aplicação em questão, apenas duas saídas foram utilizadas. As saídas utilizadas pelo DDX são do tipo PWM e passam por um amplificador operacional antes de estarem disponíveis nos conectores de saída. A placa ainda possui conectores para alimentação e para a interface I^2C.

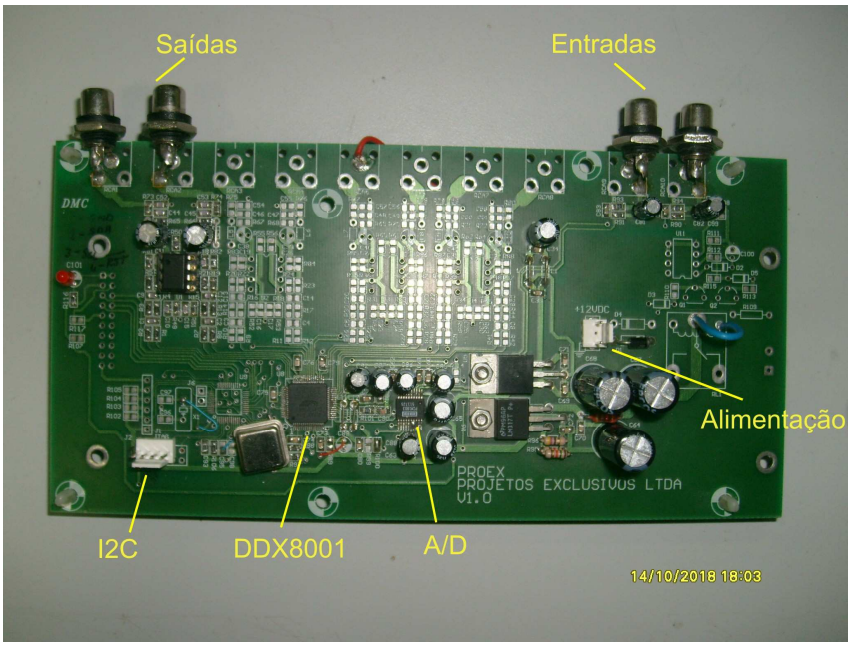


Figura 39: Proex V1.0

6.3 Sistema de Controle

O sistema de controle do DDX8001 é composto por: CD Player, altofalante, placa de desenvolvimento Proex, LPC2148, teclado e LCD (20x4). O CD Player é a fonte do sinal de áudio que será processado no DDX. O altofalante é a saída do sistema, responsável por reproduzir o áudio processado. O LPC2148 é responsável pelo controle das funções do DDX que serão modificadas via teclado e expostas ao usuário através do LCD. Abaixo temos o diagrama de blocos para o sistema de controle e a figura do sistema real.

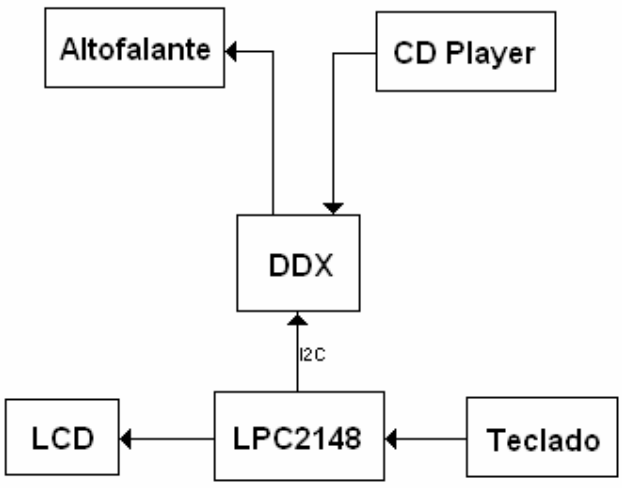


Figura 40: Diagrama de blocos do sistema do controle do DDX

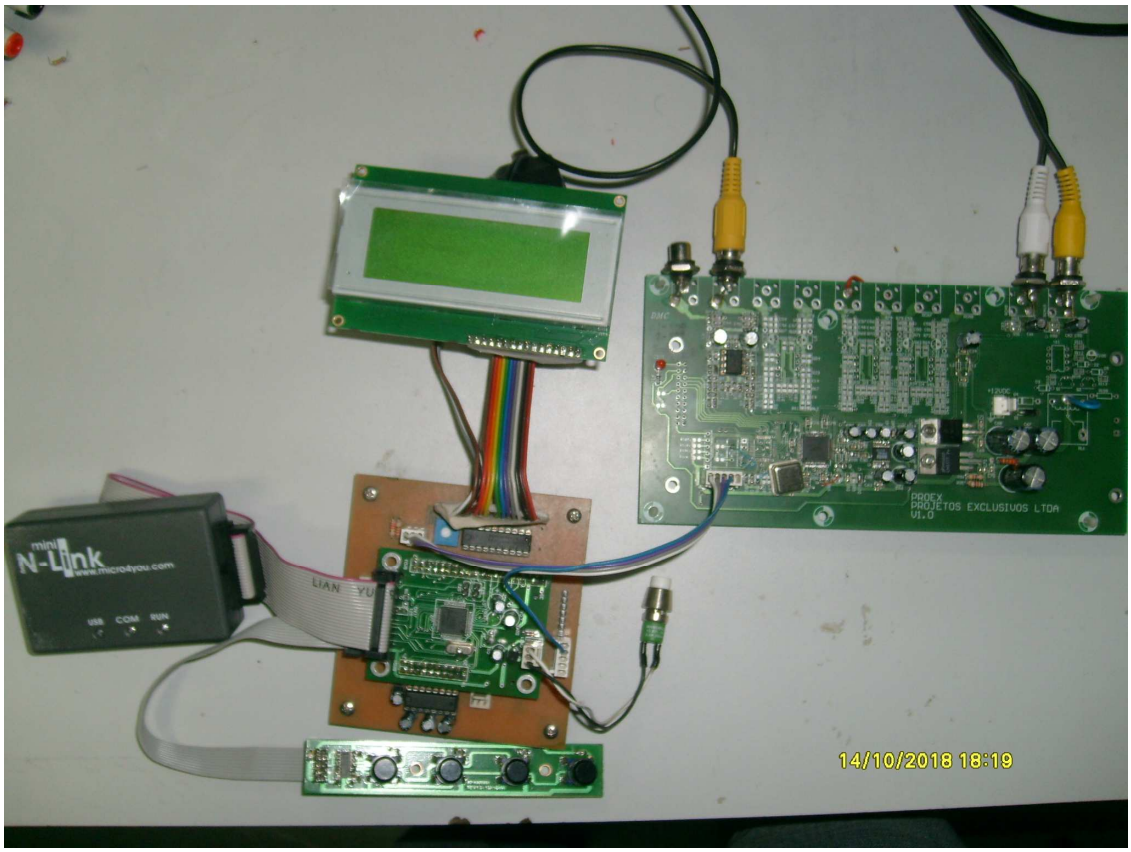


Figura 41: Sistema de controle real

6.4 Rotina de controle

A rotina desenvolvida para o LPC2148 visa controlar a inicialização, as funções de volume e a equalização do DDX-8001. O compilador utilizado possui funções para o uso do LCD e da interface I^2C. A biblioteca para uso do teclado foi desenvolvida por técnicos da APEL. As teclas são lidas de forma serial pelo microcontrolador através do CI 74165.

No início da rotina, o Timer0 é configurado, o LCD, a interface I^2C e o teclado são inicializados. Como o compilador não possui função de delay, o Timer 0 é configurado para interromper a cada 1 milessegundo, e uma função de delay é implementada. O módulo I^2C é inicializado com a taxa de 200 kbits/s.

Após a inicialização dos periféricos utilizados pelo sistema, o DDX é inicializado conforme a seqüência exposta na Tabela 2. Concluída a inicialização, o sistema exibe os valores do volume e equalizador correntes no sistema. A partir daí, inicia-se a rotina de seleção de menus. O sistema possui quatro submenus: Volume, Equalizador, Mute e Filtro, os quais podem ser acessados através do teclado utilizando os três primeiros botões, chamados de ENTER, UP e DOWN.

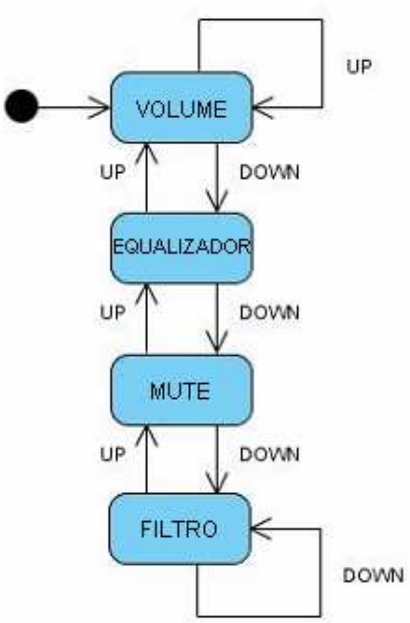


Figura 42: Submenus

O submenu Volume modifica o volume do áudio de saída ao pressionar as teclas UP ou DOWN. A tecla ENTER abandona o submenu Volume e retorna ao menu. O submenu Equalizador funciona da mesma forma, sendo as teclas UP e DOWN utilizadas para selecionar o equalizador desejado. No submenu Mute, a tecla UP habilita o mute e a tecla DOWN desabilita. O submenu Filtro foi utilizado para o cálculo dos parâmetros de um filtro passa-baixas na frequência de 1kHz.

A rotina completa com as bibliotecas do LCD, I^2C , teclado e DDX, encontra-se anexada no final do relatório.

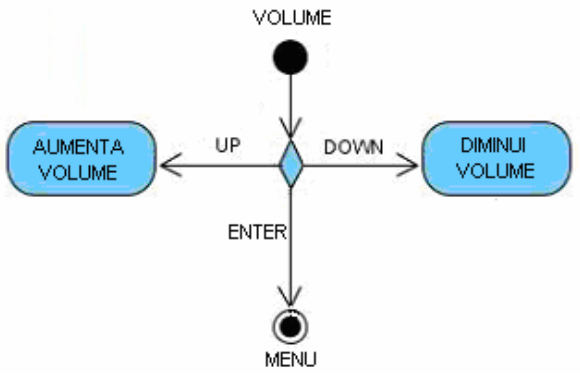


Figura 43: Submenu Volume

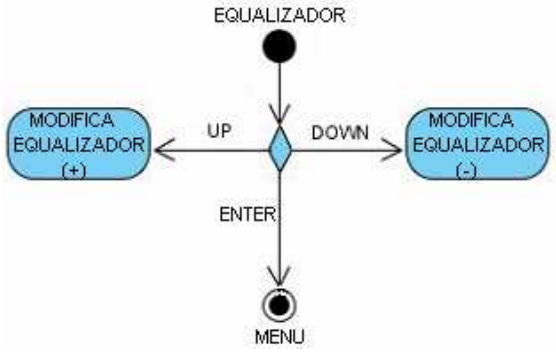


Figura 44: Submenu Equalizador

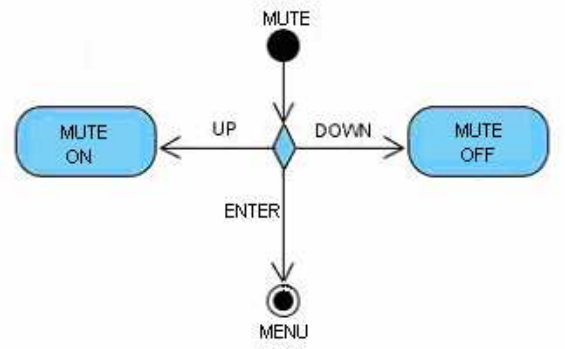


Figura 45: Submenu Mute

6.5 Considerações Finais

O controle do DDX-8001 pode ser realizado através de qualquer microcontrolador que possua a interface de comunicação I^2C. O LPC2148 foi escolhido com fins de aprendizado. A implementação dos filtros não pode ser realizada devido ao término do estágio, apenas o cálculo dos parâmetros foi realizado.

7. Considerações Finais

Neste relatório foram apresentadas as atividades realizadas durante o estágio supervisionado na empresa de equipamentos eletrônicos APEL.

As atividades propostas no cronograma foram executadas dentro do prazo estabelecido para término do estágio.

O estágio foi de grande importância para familiarização com novas tecnologias e com os problemas presentes em uma empresa de equipamentos eletrônicos.

Referências Bibliográficas

[1]. VASSALLO, Francisco. **Manual de Caixas Acústicas e Alto-Falantes**. Recife, 2005.

[2].CORREIRA, Carlos; SETTE, Homero. **Alto-falantes e Caixas Acústicas, Características e utilização**. Nova Santa Rita, RS: Eletrônica Selenium SA.

[3]. RUMSEY, Francis; McCORMIC, Tim. **Sound and Recording: An Introduction**. ELSEVIER, 2006.

[4]. VASSALLO, Francisco. **Manual de Caixas Acústicas e Alto-Falantes**. Recife, 2005.

[5]. ARM LIMITED. **ARM7TDMI-S Technical Reference Manual**. Março, 2004. Disponível em: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0234b/DDI0234.pdf

[6]. ARM LIMITED. **ARM Architecture Reference Manual**. Julho, 2005. Disponível em: www.arm.com/miscPDFs/14128.pdf

[7]. SLOSS, Andrew; SYMES, Dominic; WRIGHT, Cris. **ARM System Developer's Guide-Designing and Optimizing System Software**. ELSEVIER, 2004.

[8]. PHILIPS SEMICONDUCTORS. **LPC214X User Manual**. Agosto, 2005. Disponível em: http://www.nxp.com/acrobat_download/datasheets/LPC2141_42_44_46_48_4.pdf

[9] RODRIGUES, Daniel. **Microcontroladores ARM7 (Philips - família LPC213x) - O poder dos 32 Bits - Teoria e Prática.** Editora Érica, 2004.

[10] SEDRA, Adel; SMITH, Kenneth; **Microeletrônica.** Makron Books, São Paulo, 2005.

[11] NILSSON, James; RIEDEL, Susan. **Circuitos Elétricos**. LTC, Rio de Janeiro, 2003.

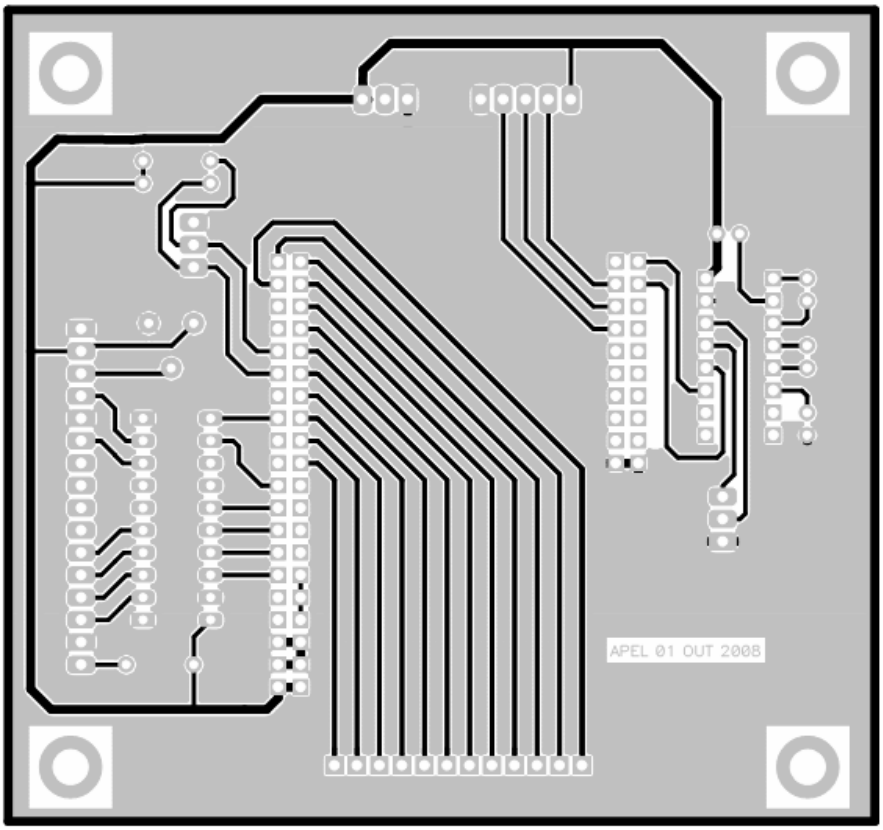
[12]APOGEE TECHNOLOGY, **DDX-8001 Datasheet,** Disponível em: www.apogeemems.com/ddx/PDFs/DDX-8001.pdf

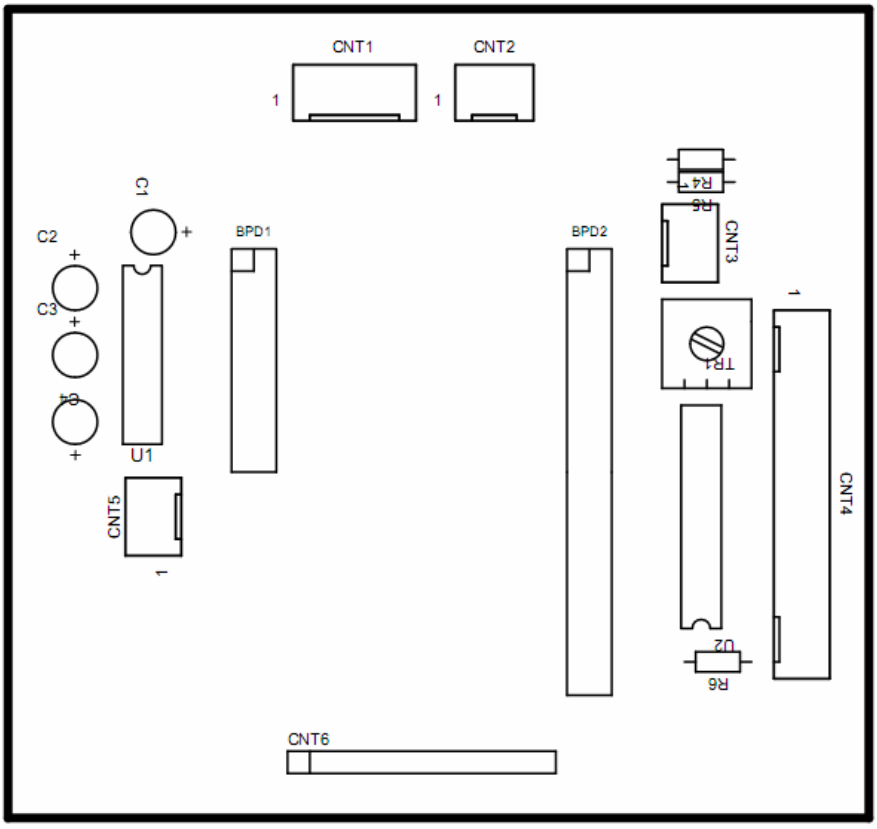
[13]APOGEE TECHNOLOGY, **AN-06: Filter Design Equations,** Disponível em: http://www.apogeemems.com/ddx/PDFs/AN-06.pdf

[14]APOGEE TECHNOLOGY, **AN-11: I2C Procedure - 5.1 Channels using the DDX-8000,** Disponível em: httphttp://www.apogeemems.com/ddx/PDFs/AN-11.pdf

Anexos

Layout da Placa de Desenvolvimento do LPC2148





Rotina do Sistema de Controle do DDX-8001

1. Programa Principal

```
#define _MAIN_
#include <stdio.h>
#include <GERAL.H>
#include <math.h>

#if CPU_PROCESSOR == CPU_PROCESSOR_LPC213x
  #include <LPC213X.H>
#endif

#if CPU_PROCESSOR == CPU_PROCESSOR_LPC214x
  #include <LPC214X.H>
#endif

#include <MAIN.H>

int main(void) {
    u08 key;
    tU8 buffer[2];
    app_boot();

  lcd_clear();
   lcd_posic(0);
   lcd_print("Master Sound = ");
   lcd_gotoxy(1,15);
   lcd_putbyte(ddx_volume_master(1));
   lcd_gotoxy(2,0);
   lcd_print("Preset = ");
   lcd_gotoxy(2,8);
   ddx_set_preset(1);
   buffer[0] = 0x00;
   i2cWrite(0x40, 1, 0x1B, buffer, 1);
   i2cStop();
   buffer[0]=0x41;
   i2cWrite(0x40, 1, 0x1F, buffer, 1);
   i2cStop();
   lcd_clear();
   do
   {
      key = kbd_leitura();
      if (key == 0x01) submenu_enter();
      else if (key == 0x02)   ++submenu;
      else if (key == 0x03) --submenu;
      adjust_menu();


   }while(1);
```

```
//   app_calcular();
 }

void app_boot(void) {


   // Timer 0
   app_timer0_init();


    // Uart 1
   uart1_init();

   // Teclado
   kbd_setup();


   //LCD
   lcd_setup();
   lcd_init();

   // I2C 0
   i2cInit(200000);

 //Inicializa o DDX
   ddx_setup();
   ddx_init();

   ddx_boot_teste();


}

void app_timer0_init(void)
{
   /*
      Configuracao dos tempos
      MSEL = 5
      PSEL = 2
      PCLK = 12*MSEL/f(VPBDIV) = 60 MHZ

      Tick do Timer Requerido = 100 us

   */

  T0MR0 = 59999;         // 1 mSec = 60000 - 1 contagens
  T0MCR = 3;             // Interrupt and Reset on MR0
  T0TCR = 1;             // Habilita o Timer 0
```

```
  /* Configura a interrupcao no vetor 0 */
  VICVectAddr2 = (unsigned long) app_timer0_irq;
 VICVectCntl2 = 0x20 | 4;      // use it for Timer 0 Interrupt
 VICIntEnable |= 0x00000010;   // Habilita a interrupcao do Timer 0
}

void app_timer0_irq (void) __irq
{

  if (app_delay_hab) {
    if (!--app_delay_conta) app_delay_hab = OFF;
  }

  T0IR            = 1;   // Limpa o flag de interrupção
 VICVectAddr = 0;   // Reconhece a interrupção

}

void app_delayms(u16 tempo)
{

  app_delay_conta = tempo;
  app_delay_hab      = ON;
  while(app_delay_hab);

}
void menu_tela_inicial(void)
{

  /* Dados do programa e versão de compilação */
  lcd_cprint("DDX-8001 - Driver",1);
  lcd_cprint("www.apel.com.br",2);
  lcd_cprint("DDX - Rev. 1.0",3);
  lcd_cprint("2007.11.16.19.00",4);
  app_delayms(1000);

}
void submenu_enter (void)
{
  if (submenu==0x01)
    enter_master();
  else if (submenu==0x02)
    enter_preset();
  else if (submenu==0x03)
    enter_mute_all();
  else if (submenu==0x04)
     passa_baixas();
}
```

```
void enter_master (void)
{

  lcd_posic(0);
  lcd_print("Master Sound");
  while (kbd_leitura() != 0x01)
  {
   if (kbd_leitura() == 0x02) {
      lcd_gotoxy(2,1);
      lcd_putbyte(ddx_volume_master(1));
    }
    if (kbd_leitura() == 0x03) {
      lcd_gotoxy(2,1);
      lcd_putbyte(ddx_volume_master(0));
    }
  }
lcd_clear();
submenu=1;
}

void enter_preset (void)
{
  while (kbd_leitura() != 0x01)
  {
   if (kbd_leitura() == 0x02) {
      lcd_posic(0);
      lcd_print("Preset");
      lcd_gotoxy(2,1);
      ddx_set_preset(1);

    }

    if (kbd_leitura() == 0x03) {
      lcd_posic(0);
      lcd_print("Preset");
      lcd_gotoxy(2,1);
      ddx_set_preset(0);
      }
    }
lcd_clear();
submenu=1;
}

void enter_mute_all (void)
{
    tU8  buffer[2];
  while (kbd_leitura() != 0x01)
  {
   if (kbd_leitura() == 0x02)
   {
```

```
        lcd_posic(0);
        lcd_print(" Mute all");
        lcd_gotoxy(2,1);
        lcd_print("ON");
        buffer[0] = 0x01;
        i2cWrite(0x40, 1, 0x09, buffer, 1);
        i2cStop();
    }

     if (kbd_leitura() == 0x03)
     {
        lcd_posic(0);
        lcd_print(" Mutel all");
        lcd_gotoxy(2,1);
        lcd_print("OFF");
        buffer[0] = 0x00;
        i2cWrite(0x40, 1, 0x09, buffer, 1);
        i2cStop();
     }
  }
lcd_clear();
submenu=1;
}
void adjust_menu(void)
{
  if (submenu >menu_tam) submenu = menu_tam;
  if (submenu <=0) submenu = 0;

  if (submenu==1)
  {
    lcd_clear();
    lcd_posic(0);
    lcd_print("Master Sound");
    app_delayms(200);
  }

  else if (submenu==2)
  {
    lcd_clear();
    lcd_posic(0);
    lcd_print("Preset");
    app_delayms(200);
  }
  else if (submenu==3)
  {
    lcd_clear();
    lcd_posic(0);
    lcd_print("Mute all");
    app_delayms(200);
  }
```

```
  else if (submenu==4)
  {
    lcd_clear();
    lcd_posic(0);
    lcd_print("Filtro");
    app_delayms(200);
  }
}

void passa_baixas (void)
{
  s32 b0, b1, b2, a0, a1, a2;
  s32 b2_a0, b0_a0_2, a2_a0, a1_a0_2,b1_a0_2;
  s32 wc,wco, ws, alfa, q,fs;

  while (kbd_leitura() != 0x01)
  {
   if (kbd_leitura() == 0x02)
   {
       lcd_posic(0);
       lcd_print("Filtro");
       lcd_gotoxy(2,1);
       lcd_print("ON");
       buffer[0] = 0x01;
       i2cWrite(0x40, 1, 0x09, buffer, 1);
       i2cStop();
   }

    if (kbd_leitura() == 0x03)
    {
      lcd_posic(0);
      lcd_print("Filtro");
      lcd_gotoxy(2,1);
      lcd_print("OFF");
      buffer[0] = 0x00;
      i2cWrite(0x40, 1, 0x09, buffer, 1);
      i2cStop();
    }

    if (kbd_leitura() == 0x04)
    {
    fs = 19200;
    wc = 2*pi*1000/fs;
    ws = sin(wc);
    wco = cos(wc);
    alfa = ws/(2*q);
    b0 = (1-wc)/2;
    b1 = 1-wc;
    b2 = (1-wc)/2;
    a0 = 1+alfa;
```

```
    a1 = -2*wc;
    a2 = 1-alfa;

    b2_a0 = b2/a0;
    b0_a0_2 = (b0/a0)/2;
    a2_a0 = -a2/a0;
    a1_a0_2 = -(a1/a0)/2;
    b1_a0_2 = (b1/a0)/2;

    //converter
    //i2c
    }

  }
lcd_clear();
submenu=1;
}
```

2. Arquivo de Cabeçalho: Main.h

```
  /* UART1.C */
  extern   word   strtoint                (byte *buffer, byte digitos);
  extern   void   trim                    (byte *buffer, byte tamanho);
  extern   void    uart1_init              (void);

  extern   void    uart1_putchar            (u08 dado);
  extern   void    uart1_putbuffer          (u08 *buffer, u16 tam);
  extern   void    uart1_putabuffer         (u08 *buffer, u16 tam);
  extern   void    uart1_putstring          (u08 *buffer);
  extern   void  uart1_modo                 (u08 modo);
  extern   void    uart1_puthexa            (u08 valor);

  /* UTIL.C */
  extern   word   strtoint                (byte *buffer, byte digitos);
  extern   void   trim                    (byte *buffer, byte tamanho);

  /* LCD.C */
  extern    void   lcd_setup               (void);
  extern   void    lcd_init               (void);
  extern    void    lcd_print              (byte *string);
  extern    void    lcd_posic              (byte posicao);
  extern    void   lcd_lposic              (byte linha);
  extern   void    lcd_gotoxy             (byte linha, byte coluna);
  extern   void    lcd_clear              (void);
  extern    void    lcd_lprint             (byte *string);
  extern   void    lcd_cprint             (byte * string, byte linha);
  extern   void    lcd_wprint             (word valor, byte digitos);
  extern   void    lcd_bprint             (byte dado);
  extern   void    lcd_putword            (word dado);
```

```
  extern   void    lcd_putbyte                    (byte dado);

  extern   void    lcd_data_out                   (byte dado);
  extern   void    lcd_lclear                     (byte linha);
  extern    void  lcd_blink                      (byte estado);
  extern    void  lcd_shift_screen               (byte estado);
  extern    void  lcd_shift_cursor               (byte estado);
  extern    void  lcd_cursor                     (byte estado);
  /* KBD.C */
  extern   void   kbd_setup                      (void);
  extern    byte   kbd_leitura                   (void);
  extern   byte    kbd_externo                    (void);

  /* I2C0.C */
  extern    void   i2cInit                       (tU32 i2cFrequency);
  extern   tS8    i2cRepeatStart                 (void);
  extern   tS8     i2cRead                        (tU8  addr, tU8* pBuf, tU16 len);
  extern    tS8    i2cWrite                       (tU8 addr, tU8 extraCmd, tU16 extra,
                                                  tU8* pData, tU16 len);
  extern   tS8    i2cStop                        (void);


  /* DDX8001.C */
  extern   void   ddx_setup                      (void);
  extern   void   ddx_init                       (void);
  extern   void   ddx_boot_teste                 (void);
  extern   byte    ddx_volume_master                  (byte tipo);
  extern    byte   ddx_volume_chanel                  (byte canal, byte tipo);
  extern  void   ddx_set_preset                  (byte tipo);
  extern void submenu_enter                      (void);
  extern void enter_master                       (void);
  extern void enter_preset                       (void);
  extern void enter_mute_all                     (void);
  extern void adjust_menu                        (void);

  extern void passa_baixas                       (void);
#ifdef _MAIN_
  void   app_boot                          (void);
  void    app_timer0_init                   (void);
  void    app_timer0_irq                    (void) __irq;
  void    app_delayms                        (u16 tempo);
  #endif
u08 submenu = 0;
#ifdef _MAIN_
 u16   app_delay_conta                     = 1000;
  u08   app_delay_hab                        = OFF;
  u08 app_int_ext                           = OFF;

#endif
```