[**ADSP-21160M**]

APRESENTAÇÃO SOBRE O DSP ADSP-21160M

RAFAEL ASTUTO AROUCHE NUNES

ORIENTADOR
Marcelo Portes de Albuquerque (CBPF)

LPS – Laboratório de Processamento de Sinais (UFRJ)
CAT - Coordenação de Atividades Técnicas (CBPF)

OUTUBRO DE 2005

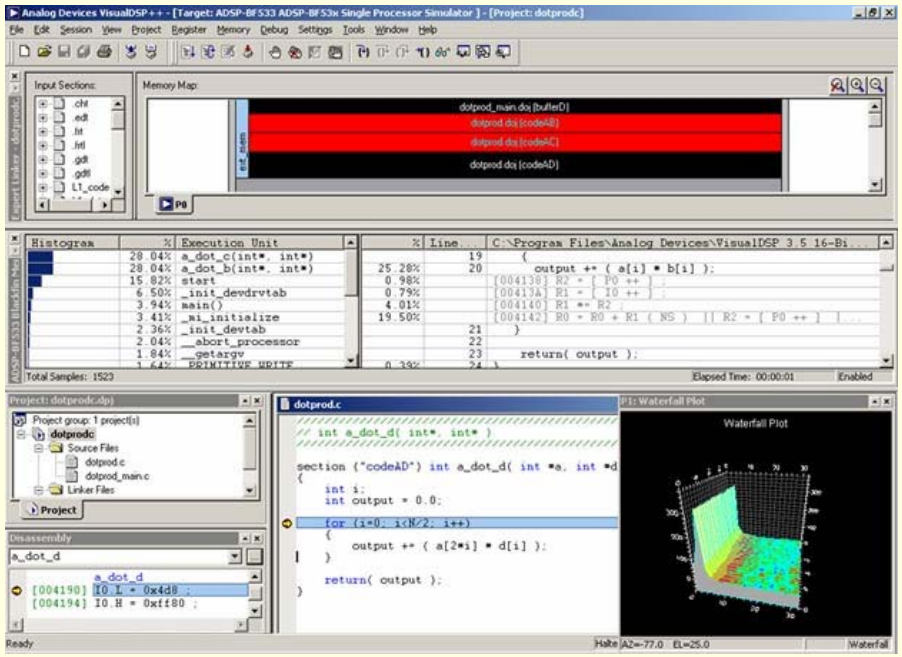
[ADSP-21160M] CARACTERÍSTICAS

- Fabricante: Analog Devices, Inc.
  http://www.analog.com/
- Segunda geração de processadores do fabricante
- Primeiro processador da nova família da Analog Devices' SHARC DSPs: ADSP-2116x
- Seu antecessor: ADSP-21065L (família ADSP-2106x)

[ADSP-21160M]

DESENVOLVIMENTO DE PROJETOS

- Kit de desenvolvimento EZ-KIT Lite:
  - Placa
  - IDE (VisualDSP++)
    - C/C++
    - Assembler
    - Interface integrada ao usuário (gerenciamento de projetos, debug, plot)
    - Simulação e emulação
    - Kernel (VDK)
    - Suporte TCP/IP e USB



[ADSP-21160M]

ESPECIFICAÇÕES & APLICAÇÕES

- Especificações técnicas
  - **Clock Speed:** 80 MHz
  - **MMACS:**160
  - **MFLOPS:** 480
  - **On-Chip SRAM (Mbits):** 4Mbit
  - **On-Chip ROM (Mbits):** 0Mbit
  - **Serial Ports:** 2
  - **Link Ports:** 6
  - **Core Voltage (V):** 2.5V
  - **DAI** – 0
  - **SPDIF/DTCP-Package:** PBGA
- Aplicações
  - Áudio
  - Instrumentação médica
  - Militar
  - Gráfica
  - Imagem
  - Comunicações

[ADSP-21160M]

ESPECIFICAÇÕES

- MMACS
  - MACs: Hardware de **M**ultiplicação e **AC**umulação.
  - Base para criação de filtros, FFTs e convoluções (multiplicação e acumulação sucessivas).
  - Realizam milhões de operações por segundo (daí o nome MMAC).
  - Quanto mais MMACS mais rápido e eficiente é o DSP.
- MFLOPS
  - Parecidos com os MMACS.
  - Realizam milhões de operações em ponto flutuante (**FLO**ating **P**oint).
  - Quanto mais MFLOPS mais rápido e eficiente é o DSP.

[ADSP-21160M]

ESPECIFICAÇÕES

- DAI
  - Digital Applications Interface.
  - Possui 20 pinos ligados ao processador que podem ser ligados à qualquer periférico ou a programas com input e output.
- SPDIF
  - Sony Phillips Digital Interface
  - Ligado ao codec de áudio do DSP (interface com CD player)
- DTCP
  - Encriptação e decriptação de informação de áudio

[**ADSP-21160M**]

O CHIP

- COMPOSTO POR VÁRIOS BARRAMENTOS DE MEMÓRIA
- GERADORES DE ENDEREÇO
- REGISTRADORES
- UNIDADES LOGICAS ARITMÉTICAS
- INTERRUPTORES
- TIMER

[ADSP-21160M]

A PLACA

- COMPOSTA BASICAMENTE POR 5 BLOCOS DOMINANTES:
  - **CORE PROCESSOR**
    - DAG
    - INTERRUPTS
    - TIMER
    - ALU
    - BUS CAPACITIES
  - **DUAL-PORTED SRAM**
  - **EXTERNAL PORT**
  - **I/O PROCESSOR**
    - DMA CONTROLER
  - **JTAG**

[ADSP-21160M]

A PLACA

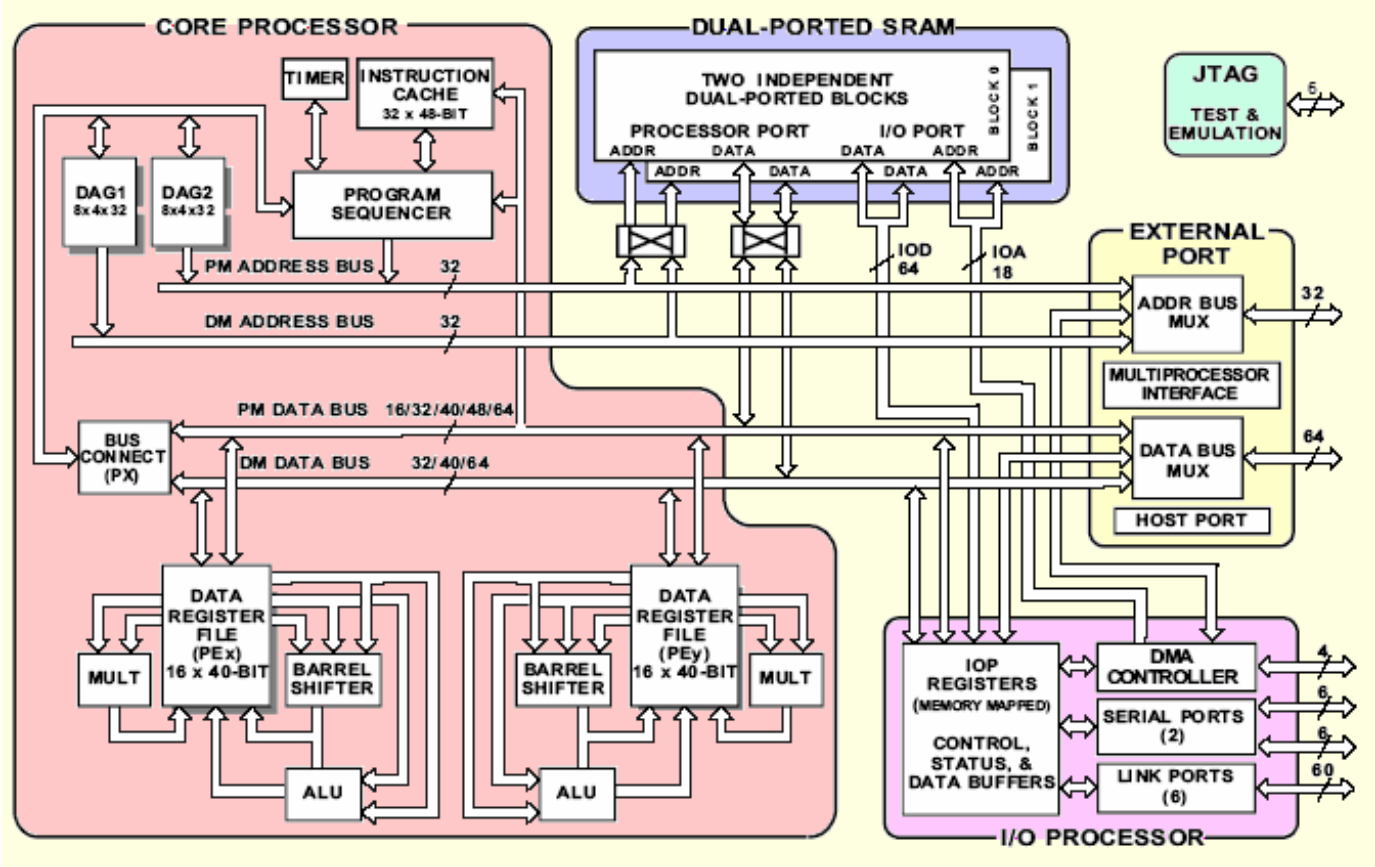


DIAGRAMA DE BLOCOS ADSP-21160M

CORE PROCESSOR

- DAG (DATA ADDRESS GENERATORS)
  - Existem dois geradores de endereço.
  - Fornecem endereçamento imediato ou indireto quando informação é transferida entre memória e registradores.
  - Permitem o processador a fornecer saídas de endereço simultâneas para duas operações de leitura ou escrita.
  - DAG1: suporta endereços de 32-bits para armazenamento em memória.
  - DAG2: suporta 32-bits para programar a memória para acesso de dados

CORE PROCESSOR

- BUS CAPACITIES
  - Composto por 6 barramentos principais:
    - PM address, PM data, DM address, DM data, IO address, IO data.
  - O PM address e o DM address bus transferem os endereços para instruções e dados.
  - PM e IO address contem 32-bits.
  - DM e IO data contem 64-bits.
  - DM address fornece um trajeto para o conteúdo de qualquer registrador no processador para ser transferido para qualquer outro registrador ou locação de memória.
  - O IO address e IO data acessam internamente a memória via DMA sem prejudicar o andamento do processador.

CORE PROCESSOR

- TIMER
  - Intervalo programável.
  - Provem geração de interrupções periódicas.
  - Quando ativo o timer decrementa um contador de 32-bits em cada ciclo.
  - Quando seu contador chega a zero o ADSP-21160 gera uma interrupção. O contador é então automaticamente recarregado.
- ALU (UNIDADE LOGICA E ARITMÉTICA)
  - Realiza as operações aritméticas e lógicas
  - Ponto fixo ou flutuante

CORE PROCESSOR

- INTERRUPTS
  - 4 interruptores
    - IRQ0-2
    - Reset
  - Podem controlar:
    - Timer
    - Operações no DMA
    - Buffers circulares
    - Operações aritméticas
    - Multi-processamento de vetores
    - Definições do usuário
  - Interface com LEDs
  - Realização de tarefas pré-definidas em software
  - Interação com o timer
    - SIG_TMZ0

CÓDIGO DE EXEMPLO: TIMER E INTERRUPTS

```
// RAFAEL ASTUTO – 28/03/05

#include <stdio.h>
#include <21160.h>
#include <macros.h>
#include <signal.h>
asm("#include <def21160.h>");

unsigned n=0;        // indice correspondente ao numero de
   interrupcoes

void timer_func (int);

void timer_func (int y)
{
   if (n==4 || n==5 || n==6 || n==7)
        set_flag (SET_FLAG0, SET_FLAG);
   if (n==2 || n==3 || n==6 || n==7)
        set_flag (SET_FLAG1, SET_FLAG);
```

CÓDIGO DE EXEMPLO: TIMER E INTERRUPTS

```
   if (n==1 || n==3 || n==5 || n==7)
        set_flag (SET_FLAG2, SET_FLAG);
   if (n==0 || n==1 || n==2 || n==3)
        set_flag (SET_FLAG0, CLR_FLAG);
   if (n==0 || n==1 || n==4 || n==5)
        set_flag (SET_FLAG1, CLR_FLAG);
   if (n==0 || n==2 || n==4 || n==6)
        set_flag (SET_FLAG2, CLR_FLAG);

   n = circindex (n,1,8);      // buffer circular
}

unsigned time = 40E6;          // configurando o timer

void main (void)
{
   asm("bit set mode2 IRQ0E;");
```

CÓDIGO DE EXEMPLO: TIMER E INTERRUPTS

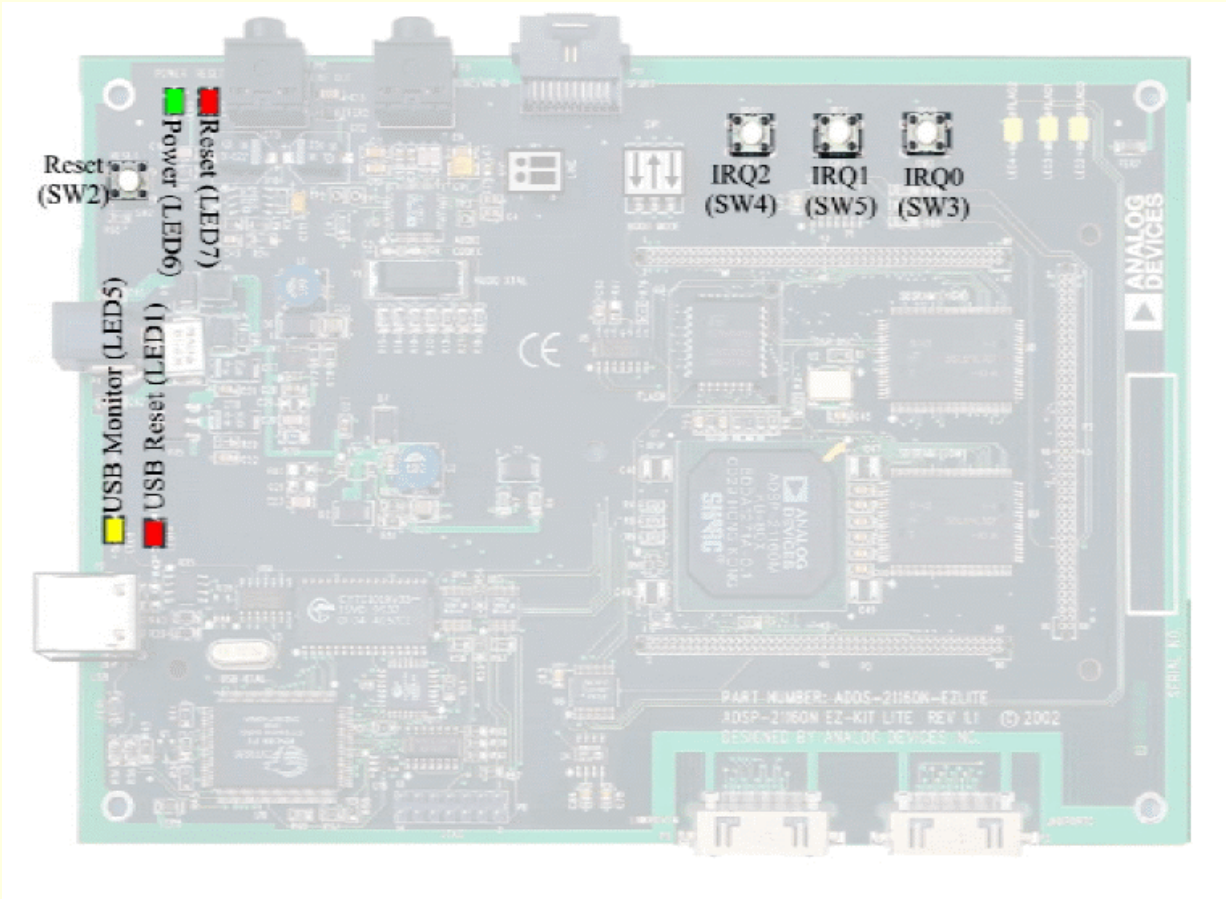
```
   timer_set (time, time);      // iniciando o timer
   timer_on();                        // ligando o timer

   interrupt (SIG_TMZ0, timer_func);      // gerando a
   interrupcao

   do
   {
   idle ();
   } while (1);
}
```

[**ADSP-21160M**]

CÓDIGO DE EXEMPLO: TIMER E INTERRUPTS



[ADSP-21160M]

DUAL PORTED SRAM

- Contêm quatro mega de memória, organizada em dois blocos com dois megabits cada.
  - podem ser configuradas de diferentes formas para armazenamento de dados
- Possuem acesso independente do core processor, IO processor ou do controlador DMA.
- Toda memória pode ser acessada através de palavras de 16, 32, 48 ou 64 bits.
- A memória pode ser configurada no máximo de 128k de palavras de 32-bits cada, ou 256k de palavras com 16-bits cada ou qualquer combinação que venha a completar os quatro mega disponíveis.

[ADSP-21160M]

I/O PROCESSOR

- Contem duas portas seriais, seis link ports e o controlador DMA.
- Serial Port:
  - Comunicação com dispositivos periféricos;
  - Recebem e transmitem de forma independente;
- Link Port:
  - Contem 6 link ports de 8-bits cada;
  - Proporcionam capacidades adicionais de input e output;
  - Podem operar independente e/ou simultameamente;
  - Pode transferir dados via DMA para a memória on-chip;

I/O PROCESSOR

- DMA Controller:
  - **D**irect **M**emory **A**ccessing
  - Pode operar independente e de forma “invisível” ao processador principal;
  - Pode ser usado na comunicação entre a memória interna e as portas seriais ou link ports;
  - Possui 14 canais disponíveis;
  - Pode ser usado em programas onde é necessária a liberação do processador principal para uso de outras tarefas;

[**ADSP-21160M**]

AD1881A SOUDMAX Codec

- 16-bits de resolução
  - Tanto para conversões A/D quanto para D/A
- Saídas com canais estéreo.
- Taxa de amostragem: 48 KHz
  - Resolução de 1 Hz
- Conectada ao DSP através da porta SPORT0.

[**ADSP-21160M**]

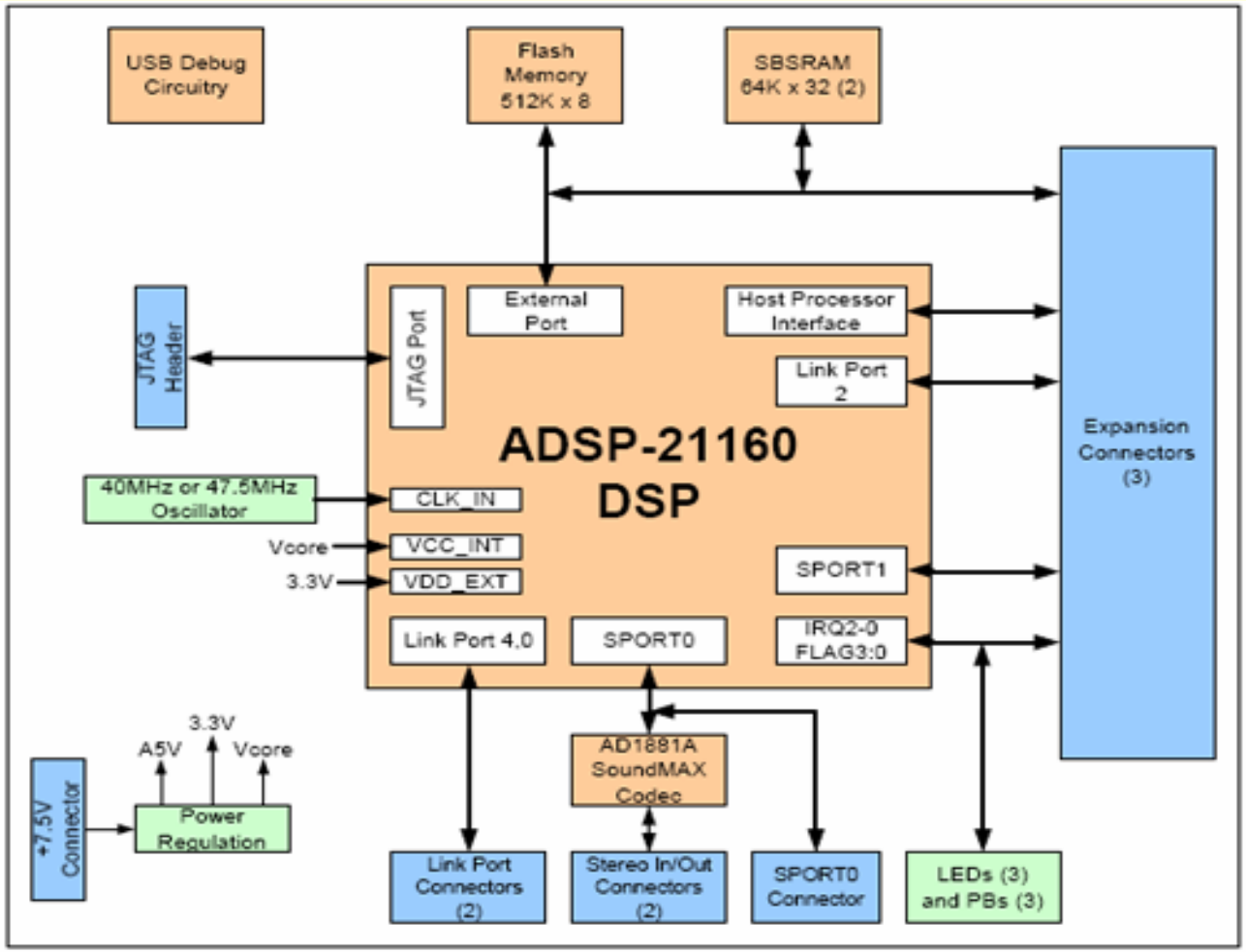
AD1881A SOUDMAX Codec

- Conversor Digital–Analógico

RESOLUÇÃO	16 bits
EXCURÇÃO	1 V_{rms} = 2,83 V_{p-p}
RESOLUÇÃO	2,83 / 65536 = 4,3182 × 10^{-5} V
SEM OFFSET	- 32.768 = - 1,415 V 0= 0 V +32.768= + 1,415 V

[**ADSP-21160M**]

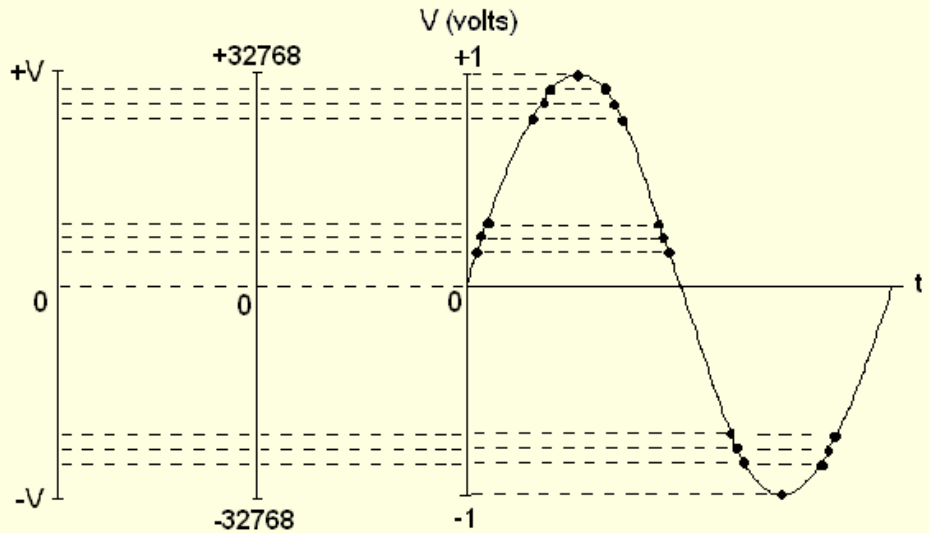
AD1881A SOUDMAX Codec



[**ADSP-21160M**]

AD1881A SOUDMAX Codec

- ESQUEMA DA CONVERSÃO



[**ADSP-21160M**]

CÓDIGO DE EXEMPLO: CONVERSÃO A/D

```
//Rodrigo Torres e Rafael Astuto

#include <cmath>
#include <21160.h>
#include <signal.h>
#include <cstdlib>
#include <stdio.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "defines.h"
#include "csound.h"
#include "bin2dec.h"
asm("#include <def21160.h>");
```

[**ADSP-21160M**]

CÓDIGO DE EXEMPLO: CONVERSÃO A/D

```
float out;

int main()
{
   CSound codec;
   char binOut[80];
   char binIn[80];
   int lenBit;

   printf("Entre com o valor binario(0x0000-0xffff): ");
   fgets(binOut, sizeof(binOut), stdin);

   while (true)
   {
        asm("idle;");
        codec.setSamples (binOut, binOut);    // conversor D/A
        codec.getSamples (binIn, binIn);      // conversor A/D
   }
   return (0);
}
```

[**ADSP-21160M**]

APRESENTAÇÃO SOBRE O DSP ADSP-21160M

RAFAEL ASTUTO AROUCHE NUNES

ORIENTADOR
Marcelo Portes de Albuquerque (CBPF)

LPS – Laboratório de Processamento de Sinais (UFRJ)
CAT - Coordenação de Atividades Técnicas (CBPF)

OUTUBRO DE 2005