
**Universidade Federal de Campina Grande**

**Centro de Engenharia Elétrica e Informática**

Curso de Graduação em Engenharia Elétrica

ÍTALO RODRIGO MONTE SOARES

# TUTORIAL BÁSICO: DSP TMS320C6713

Campina Grande, Paraíba
Julho de 2011

ÍTALO RODRIGO MONTE SOARES

# TUTORIAL BÁSICO: DSP TMS320C6713

*Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.*

Área de Concentração: Processamento Digital de Sinais

Orientador:
Professor Edmar Candeia Gurjão, Dr. Sc.

Campina Grande, Paraíba
Julho de 2011

ÍTALO RODRIGO MONTE SOARES

# TUTORIAL BÁSICO: DSP TMS320C6713

Relatório de Estágio Supervisionado submetido à
Unidade Acadêmica de Engenharia Elétrica da
Universidade Federal de Campina Grande como parte dos
requisitos necessários para a obtenção do grau de
Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento Digital de Sinais

Aprovado em / /

**Professor Avaliador**
Universidade Federal de Campina Grande
Avaliador

**Professor Edmar Candeia Gurjão, Dr. Sc.**
Universidade Federal de Campina Grande
Orientador, UFCG

# SUMÁRIO

# LISTA DE ILUSTRAÇÕES

# 1 INTRODUÇÃO

O processamento digital de sinais é usado em várias aplicações, como em comunicações, controle, processamento de voz e vídeo, por isso é essencial que esta área seja bem abordada na formação de um engenheiro eletricista.

Os Kits de Desenvolvimento (DSK) são uma forma econômica de introduzir processamento digital de sinais em tempo real para estudantes. Os DSKs produzidos pela empresa *Spectrum Digital* e que utilizam processadores da *Texas Instruments* são um dos mais comuns no mercado.

A *Texas Instruments* possui uma grande variedade de modelos de Processadores digitais de sinais – *Digital Signal Processor* (DSP) – e os divide de acordo com suas características. Os C67xx pertencem à família de processadores de ponto flutuante, enquanto os C62xx e C64xx à família de processadores de ponto fixo. Processadores de ponto fixo são melhores para dispositivos que usam bateria, como celulares, pois utilizam menos energia que um processador equivalente de ponto flutuante (CHASSAING, 2005). A escolha de um DSP, além do consumo de energia já citado, leva em consideração fatores como custo e velocidade.

Neste relatório são apresentadas de forma geral as características do DSP TMS320C6713 e o seu ambiente de desenvolvimento. Na seção 2 é descrita a arquitetura do DSP C6713 e os componentes presentes na placa de desenvolvimento. A seção 3 contém uma breve descrição do *software* de desenvolvimento, o *Code Composer Studio*, e dos arquivos necessários para construir um projeto. A seção 3 mostra como proceder no primeiro contato com o DSK, desde a instalação ate a criação do primeiro projeto.

## 1.1 LABORATÓRIO DE AUTOMAÇÃO E PROCESSAMENTO DE SINAIS

Este trabalho de estágio foi realizado no Laboratório de Automação e Processamento de Sinais (LAPS) que é coordenado pelo professor Edmar Candeia

Gurjão. O LAPS é um laboratório de pesquisa do Departamento de Engenharia Elétrica da Universidade Federal de Campina Grande (UFCG).

As atuais linhas de pesquisa do LAPS se concentram nas áreas de Rádio Definido por Software e Processamento de Sinais, esta última com destaque para Amostragem Compressiva aplicada a áudio, vídeo e correção de erros.

O LAPS possui dois modelos de DSP, o TMS320C6711 e TMS320C6713, ambos da *Texas Instruments*. Os DSPs são utilizados nas pesquisas e nas aulas práticas da disciplina de Processamento Digital de Sinais (PDS) da graduação e da pós-graduação.

Os objetivos deste estágio foram produzir um tutorial para os alunos que iniciam o trabalho com o DSP, tanto nos projetos como na disciplina da área e verificar a viabilidade de substituir o modelo C6711 pelo C6713.

# 2 DESCRIÇÃO DO HARDWARE

Nesta seção serão apresentadas as características do processador digital de sinais TMS320C6713 e da placa de desenvolvimento que acompanha o DSK.

## 2.1 PROCESSADOR TMS320C6713

O processador TMS320C6713 (C6713) é baseado na arquitetura *Very Long Instruction Word* (VLIW) que permite várias instruções serem realizadas em paralelo, no mesmo ciclo de *clock*, no caso do C6713, 8 instruções podem ser realizadas a cada ciclo. Como ele opera a uma velocidade de 225 MHz, consegue realizar 8 instruções em 4,4 ns (1/225 MHz) ou 1800 milhões de instruções por segundo (MIPS).

O núcleo do C6713 possui 8 unidades funcionais independentes, divididas em dois lados, A e B, como mostrado na Figura 1. Cada lado tem uma unidade de multiplicação (.M), uma para operações lógicas e aritméticas (.L), uma para operações de manipulação de bits e aritméticas (.S) e outra para operações de carregamento, armazenamento e aritméticas (.D).

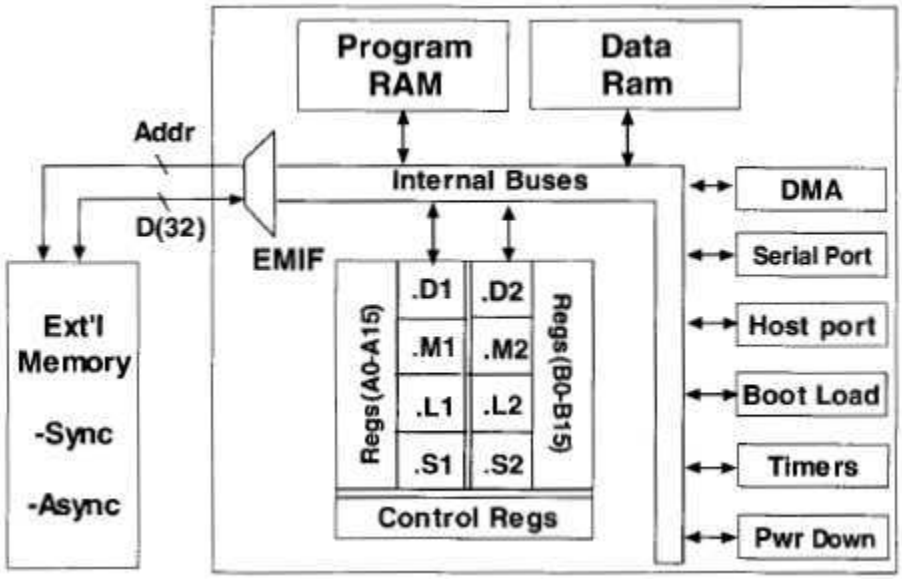


Figura 1. Arquitetura genérica de um processador C6x. Fonte: (KEHTARNAVAZ, 2005)

Cada unidade funcional pode ler ou escrever diretamente nos registradores de 32 bits. São um total de 32 registradores, 16 de cada lado. Qualquer interação com a CPU tem de ser feita através deles.

O C6713 possui 2 níveis de memória *cache*, o primeiro de 8 kB e o segundo de 256 kB, totalizando 264 kB de memória interna. Alguns periféricos do processador são descritos a seguir.

- Interface para acesso à memória externa (**EMIF**): fornece a temporização necessária para acesso à memória externa;
- **DMA**: para a transferência de dados entre partes da memória sem interferência na CPU;
- ***Boot Loader***: acelera o carregamento de código de uma memória externa ao chip ou de um *Host port* para a memória interna;
- ***Host port***: permite que um *host* acesse a memória interna;
- Temporizadores (***timers***): fornece contadores de 32 bits.

## 2.2 PLACA DE DESENVOLVIMENTO

A placa do DSK C6713 é uma plataforma de desenvolvimento autônoma que permite avaliar e desenvolver aplicações para o processador digital de sinais TMS320C6713. Ela inclui um CODEC (conversor analógico/digital digital/analógico),

memórias, um *Complex Programmable Logic Device* (CPLD), LEDs, chaves para configuração e controle, entre outros dispositivos que serão descritos a seguir. A placa e seu diagrama estão nas  Figura 2 e Figura 3.

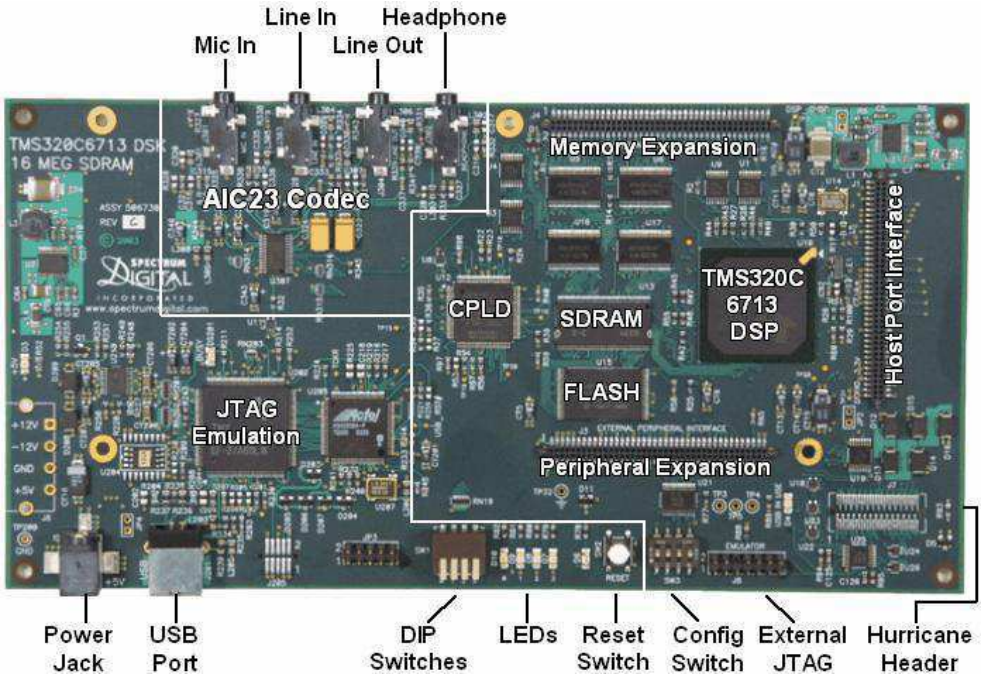


Figura 2. Placa de desenvolvimento do TMS320C6713. Fonte: (KEHTARNAVAZ, 2005)

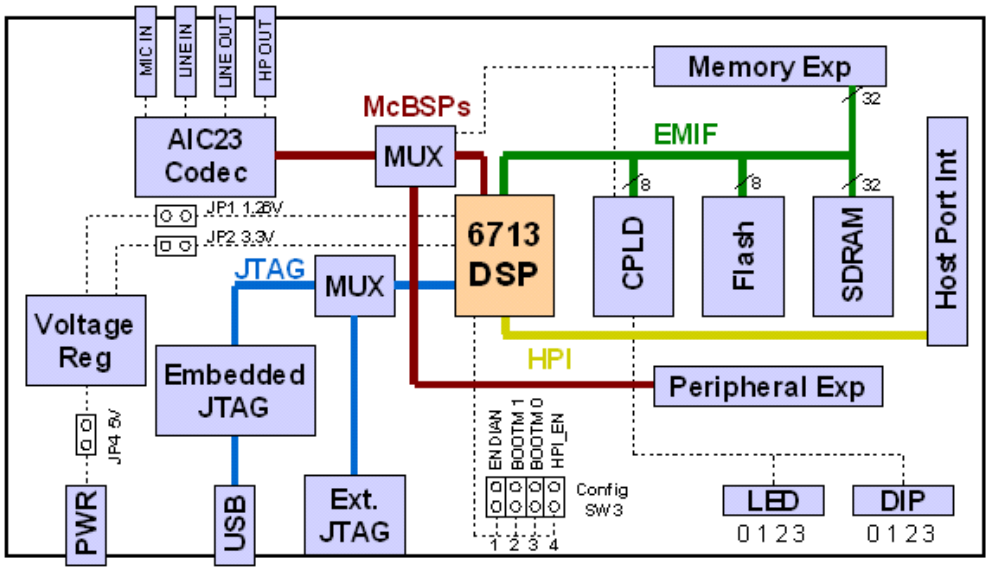


Figura 3. Diagrama da placa. Fonte: (SPECTRUM DIGITAL, 2003)

O CODEC (codificador/decodificador) é o dispositivo que transforma o sinal analógico em sinal digital para que ele possa ser processado pelo DSP. Depois que o DSP finaliza o trabalho, o sinal digital resultante retorna ao codec para ser convertido em sinal analógico.

O modelo presente na placa é o CODEC estéreo TLV320AIC23 (AIC23). Ele suporta taxas de amostragem de 8, 16, 24, 32, 44,1, 48 e 96 kHz. Essa taxa pode ser modificada por software. O tamanho da amostra é 32 bits (16 bits para cada canal caso use o sinal estéreo).

A comunicação com o AIC23 utiliza dois canais chamados de McBSP (*Multichannel buffered serial ports*). O McBSP0 é unidirecional e utilizado para enviar dados de controle ao CODEC, o McBSP1 é bidirecional e é utilizado na troca de dados entre o CODEC e o DSP. O CODEC é representado na Figura 4.

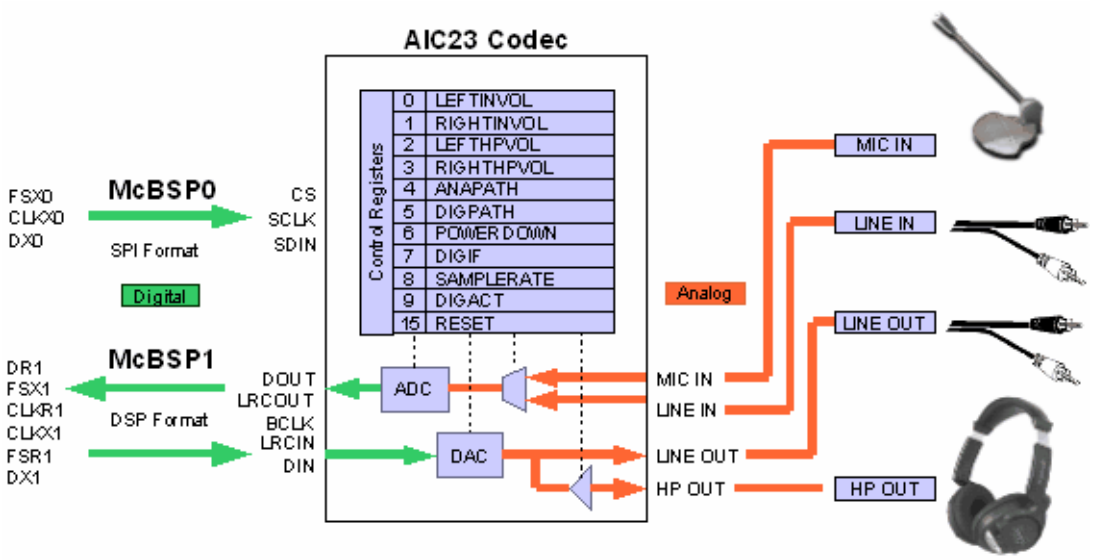


Figura 4. Codec AIC23. Fonte: (SPECTRUM DIGITAL, 2003)

A interface do sinal analógico com a placa é feita por quatro conectores de áudio do tipo P2 (3,5 mm) estéreo. O conector P2 macho possui três divisões, na ponta é enviado o sinal do canal esquerdo, no centro é enviado o canal direito e na base, o terra, que é comum para os dois canais. A Figura 5 mostra este conector.

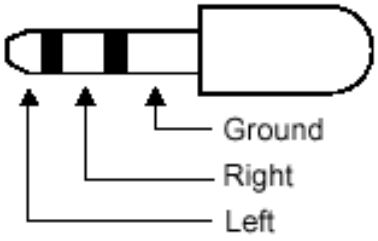


Figura 5. Conector de áudio P2 estéreo. Fonte: (SPECTRUM DIGITAL, 2003)

Os conectores MIC IN, para microfone, ou LINE IN, para outros sinais, são ativos um por vez. Já a saída é enviada para os dois conectores, LINE OUT e HP OUT, este último possui um amplificador ajustável e deve ser utilizado para ligar o sinal de

saída a um alto-falante ou fone de ouvido. Deve-se ter cuidado para que a entrada do sinal não exceda a tensão de 6V p-p.

Para auxiliar no armazenamento de dados, além da memória interna do processador, a placa contém dois tipos de memória externa: 16 MB SDRAM e 512 kB de memória Flash.

O DSK possui 8 LEDs, 4 que mostram o funcionamento da placa, como os que indicam que a fonte de alimentação e o cabo USB estão conectados, e outros 4 que podem ser definidos pelo usuário. Ao lado desses últimos se encontra um conjunto de 4 chaves (DIP *swiches*) que também podem ser programados pelo usuário, funcionando como uma interface de controle. A Figura 6 ilustra os LEDs e DIP switches programáveis. Outro conjunto de chaves, que fica ao lado do botão reset, é responsável pela configuração da inicialização (*boot*) da placa. O padrão é que todas devem estar na posição *off*.

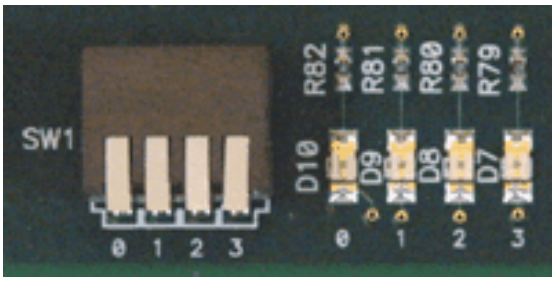


Figura 6. Dip swiches e LEDs.

Caso seja necessário aumentar a capacidade do DSK ou utilizar outras formas de entrada e saída, a placa disponibiliza três conectores de expansão. Os conectores são de 80 pinos e podem aumentar a memória, modificar *timer* e *clock* ou adicionar outros CODECs.

Outros dois dispositivos presentes na placa, mas que o programador tem pouco ou nenhum acesso, são o CPLD (*Complex Programmable Logic Device*) e o emulador JTAG. CPLD é um dispositivo lógico programável com capacidade inferior ao FPGA, ele realiza funções simples para eliminar a necessidade de mais dispositivos discretos na placa. O emulador JTAG permite que o usuário examine sinais e registros no DSP por meio de uma interface externa, é por intermédio dele que ocorre a comunicação com o computador pela porta USB.

A alimentação da placa é realizada por uma fonte de tensão de 5 V, que depois de passar por reguladores de tensão fornece 1,26 V para o núcleo do C6713 e 3,3 V para a memória e periféricos.

# 3 DESCRIÇÃO DO SOFTWARE

Nesta seção serão apresentados o software de desenvolvimento do DSK e os arquivos utilizados para programar o DSP.

## 3.1 CODE COMPOSE STUDIO

*Code Composer Studio* (CCS) é um ambiente de desenvolvimento integrado – *Integrated Development Evironment* (IDE) – que fornece ferramentas para construir, depurar e analisar em tempo real os programas.

Para transformar um programa em C em um arquivo executável que possa ser carregado no DSP, o CCS possui um compilador, um *assembler* e um *linker*. O compilador gera um arquivo fonte em *assembly*, com extensão .asm, a partir do arquivo fonte em C. O *assembly* converte o arquivo fonte .asm em linguagem de máquina .obj. Então, o *linker* combina o arquivo .obj com os objetos da biblioteca para gerar um arquivo executável com extensão .out. O arquivo .out é carregado diretamente no processador C6713. Esse processo está indicado na Figura 7.

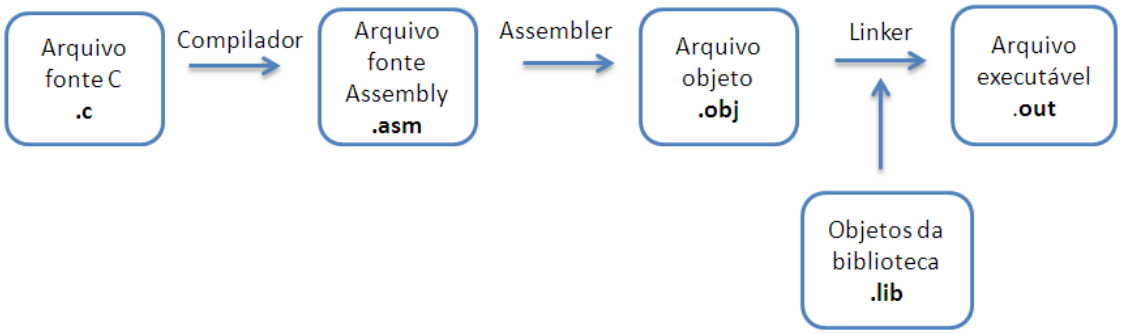


Figura 7. Esquema da construção do aquivo executável.

Uma série de ferramentas para depuração está disponível, como, por exemplo, adicionar pontos de interrupção, monitorar variáveis, observar a memória e os registradores, monitorar o tempo de execução ou realiza-la passo a passo. Neste tutorial não serão apresentadas essas ferramentas. Para um aprofundamento nestas partes consulte o TI CCS Tutorial (TEXAS INSTRUMENTS, 2000).

## 3.2 TIPOS DE ARQUIVOS UTILIZADOS

Durante o desenvolvimento com o DSK será necessário trabalhar com diversos tipos de arquivos, sendo eles:

- arquivo.pjt: contém o projeto;
- arquivo.c: programa fonte em C;
- arquivo.asm: programa fonte em *assembly*, criado pelo usuário, pelo compilador C ou pelo otimizador;
- aquivo.sa: programa fonte *assembly* linear. O otimizador linear usa o arquivo.sa como entrada para produzir um programa *assembly* arquivo.asm;
- arquivo.h: arquivo de cabeçalho;
- arquivo.lib: arquivo de biblioteca;
- arquivo.cmd: arquivo de comandos que mapeia seções da memória;
- arquivo.obj: arquivo objeto criado pelo *assembly*;
- arquivo.out: arquivo executável criado pelo *linker* para ser carregado no processador.

## 3.3 ARQUIVOS SUPORTE

Alguns arquivos suporte são necessários para gerar o executável que será carregado no DSP. Esses arquivos servem para determinar algumas configurações da memória, do CODEC e do processador. Para iniciantes, não é necessário saber detalhes destes arquivos. Os cinco primeiros citados abaixo estão na pasta *Support* nos exemplos do livro do Chassaing (CHASSAING, 2005), disponível no LAPS.

- **C6713dskinit.c**: contém as funções que iniciam o DSP, o codec, a porta serial e as entradas e saídas.
- **C6713dskinit.h**: arquivo cabeçalho com os protótipos de funções. As configurações do ganho do sinal e a seleção entre a entrada pelo microfone ou a padrão são modificadas nesse arquivo.

- **C6713dsk.cmd**: arquivo de comando do *linker*. É ele que determina se a memória usada será a externa ou a interna.
- **Vectors_intr.asm**: arquivo de vetores para programa que utiliza interrupção.
- **Vectors_poll.asm**: arquivo de vetores para programa que utiliza *polling*.
- **rts6700.lib**, **dsk6713bsl.lib** e **csl6713.lib**: bibliotecas de suporte para o tempo de execução, a placa e o chip, respectivamente. Ficam localizadas nas pastas C6000\gtools\lib, C6000\dsk6713\lib e C6000\csl\lib.

Deve-se alterar o arquivo original C6713dsk.cmd, substitua a linha **.vectors :> IVECS** para **.vecs :> IVECS**, como indicado na Figura 8. Essa alteração corrige um erro que aparece na construção do projeto.

```
/*C6713dsk.cmd  Linker command file*/

MEMORY
{
  IVECS:    org=0h,          len=0x220
  IRAM:     org=0x00000220, len=0x0002FDE0
  SDRAM:    org=0x80000000, len=0x00100000
  FLASH:    org=0x90000000, len=0x00020000
}

SECTIONS
{
  .EXT_RAM :> SDRAM
  .vecs :> IVECS /*modificar esta linha (.vecs->.vectors)*/
  .text     :> IRAM
  .bss      :> IRAM
  .cinit    :> IRAM
  .stack    :> IRAM
  .sysmem   :> IRAM
  .const    :> IRAM
  .switch   :> IRAM
  .far      :> IRAM
  .cio      :> IRAM
  .csldata  :> IRAM
}
```

Figura 8. Modificação no arquivo suporte C6713dsk.cmd.

# 4 PROGRAMANDO O DSK

Nesta seção será mostrado detalhadamente como proceder no primeiro contato com o DSK, desde sua instalação, teste e utilização do *Code Composer Studio*.

## 4.1 INSTALAÇÃO

A instalação do *Code Composer Studio* e dos drivers da placa é simples. Insira o CD de instalação, um menu de opções aparecerá automaticamente. Execute o arquivo Launch.exe caso o menu não apareça. Selecione a opção *Install Project*. Desative o antivírus antes de começar a instalação.

Instale primeiro o C6000 CODE COMPOSER STUDIO, em seguida o DSK6713 *Drivers and Target Content* e o *Flash Burn*. Dois ícones aparecerão na área de trabalho após a instalação, DSK CCStudio v3.1 e 6713 DSK Diagnostic Utility v3.1., indicados na Figura 9.

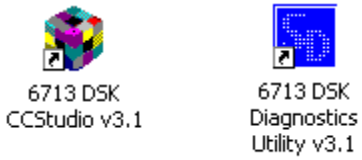


Figura 9. Ícones que aparecerão depois da instalação.

## 4.2 TESTE

Um programa de teste está armazenado na memória Flash e é executado automaticamente sempre que o DSK é ligado. O *Power On Self Test* (POST) leva de 10 a 15 segundos para finalizar, durante esse tempo todos os subsistemas do DSK são automaticamente testados e uma senóide de 1kHz é colocada na saída do CODEC por 1 segundo. Se o teste for executado com sucesso, os 4 LEDs irão piscar 3 vezes e depois permanecerão acesos.

Para realizar um teste mais detalhado, pode-se utilizar o software DSK *Diagnostic Utility*, instalado junto com o CCS. Clique no botão *Start* e espere alguns segundos, caso todos os componentes estejam funcionando aparecerá a palavra PASS, como indicado na Figura 10.

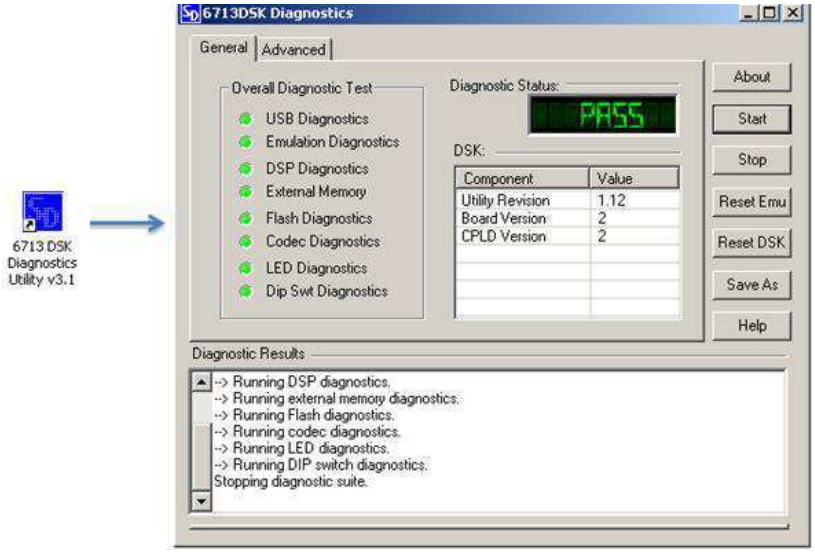


Figura 10. DSK Diagnostic Utility.

## 4.3 INICIAR O DSK

Ao abrir o programa CCS uma mensagem no canto inferior esquerdo da janela mostrará que o DSK está desconectado. Para conectá-lo, vá ao menu *Debug* → *Connect* ou use o atalho Alt+C. Se os drivers da placa foram instalados corretamente, uma nova mensagem indicará que o DSK está conectado. O DSK deve estar ligado antes de iniciar o CCS.

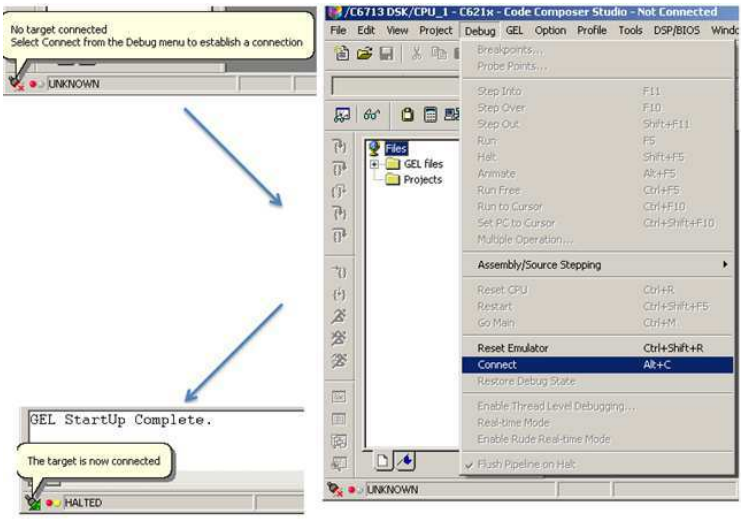


Figura 11. Conexão do DSK ao CCS.

## 4.4 CRIAR UM PROJETO

A melhor forma de iniciar o contato com o DSP é utilizar projetos já existentes, e modificá-los de acordo com sua necessidade. Com a prática se torna mais fácil criar projetos do zero.

Para criar um novo projeto vá ao menu *Project* → *New*. A janela apresentada na Figura 12 irá aparecer, nela coloque o nome do projeto e modifique o campo Target para a opção TMS320C67XX.

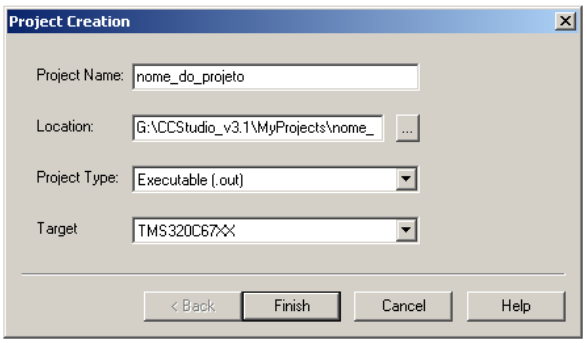


Figura 12. Janela para criar um novo projeto.

Agora, crie um novo arquivo, *File* → *New* → *Source File*. Neste primeiro exemplo copie o código da Figura 13, salve-o dentro do diretório do projeto e lembre-se de colocar a extensão .c.

```
#include "DSK6713_AIC23.h"
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;  //determina a taxa de amostragem

void main()
{
  short sample_data;

  comm_poll();                  //inicia o DSk, codec e McBSP
  while(1)                    //loop infinito
  {
    sample_data = input_sample(); //entrada da amostra
    output_sample(sample_data);   //saída da amostra
  }
}
```

Figura 13. Código básico para o DSP.

O exemplo da Figura 13 cria uma variável para salvar as amostras da entrada e envia esse valor para a saída do CODEC, sem realizar nenhum processamento. Ele serve como base para outros projetos. A função **input_sample()** é utilizada para capturar a amostra e a função **output_sample()** para enviá-la.

O próximo passo é adicionar os arquivos ao projeto, o arquivo fonte recém-criado e os de suporte já citados, em *Project* → *Add Files to Project*. Os arquivos que

devem ser adicionados a esse projeto são: c6713dskinit.c, C6713dsk.cmd, Vectors_poll.asm, csl6713.lib, dsk6713bsl.lib, rts6700.leb e o novo arquivo criado. Depois deste procedimento a janela *Project view* do CCS ficará como na Figura 14.

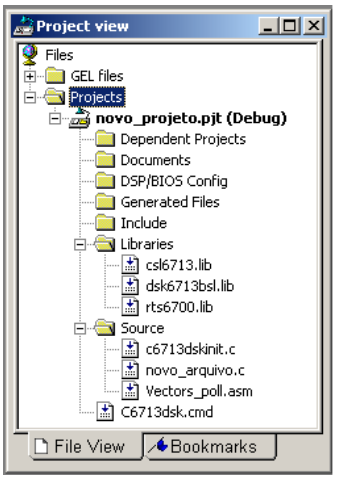


Figura 14. Janela Project view depois de acrescentar os asquivos ao projeto.

Para finalizar, ajuste as opções do compilador C e do *linker*. Selecione *Project → Build Options*. Na categoria *Basic* da aba *Compiler*, modifique o campo *Target Version* para a opção C6713x (-mv6713). Na categoria *Advanced* modifique o campo *Memory Models* para opção Far (--men_model data=far), na categoria *Preprocessor* indique o endereço dos arquivos cabeçalho no campo *Include Search Path* (-i), normalmente é C6000\dsk6713\include, na pasta do CCS. No Campo *Pre-Define Symbol* (-u) coloque CHIP_6713. Essas modificações estão indicadas nas Figura 15,Figura 16 eFigura 17.

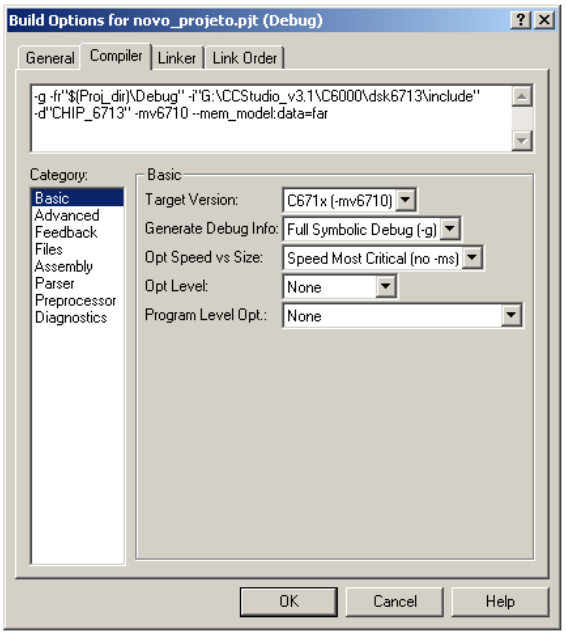


Figura 15. Opções básicas de compilação.

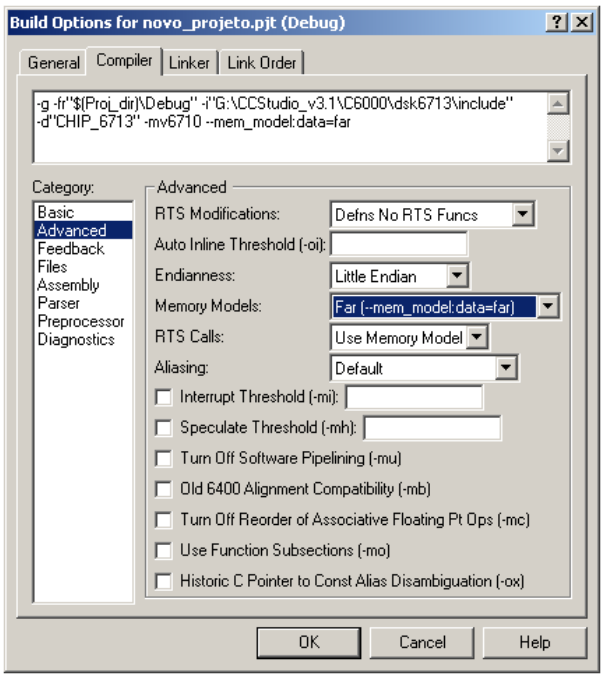


Figura 16. Opções avançadas de compilação.

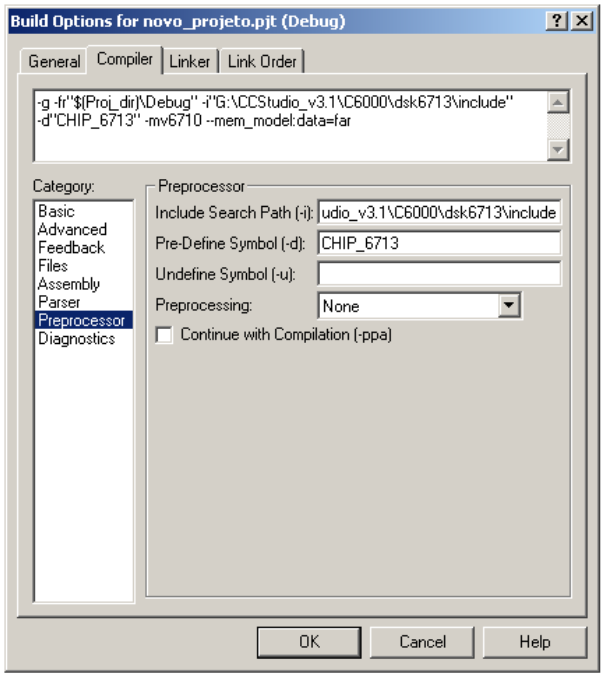


Figura 17. Opções no preprocessamento da compilação.

Na aba *Linker*, fixe os valores de *Heap Size* e *Stack Size* em 0x400 (1024 em decimal), como na Figura 18. Essa alteração é apenas para não aparecer um *warning* quando o projeto for construído. *Stack* e *Heap* são locais da memória onde as variáveis são armazenadas. O valor 0x400 indica que o tamanho destinados a essas memórias é 1 kB.

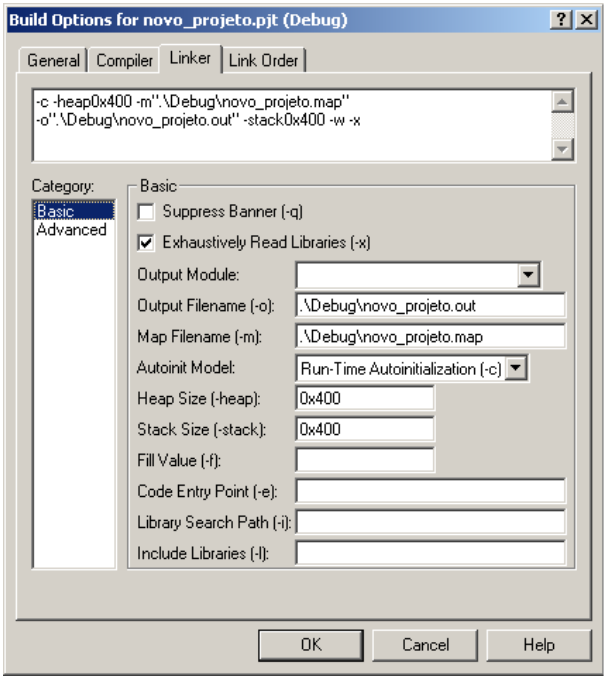


Figura 18. Opções básicas do Linker.

Com esse procedimento feito, vá ao menu *Project* → *Rebuild All* para construir o projeto e obter o arquivo executável, o resultado esperado é mostrado na Figura 19. O arquivo com a extensão .out é criado e se encontra dentro da pasta Debug no diretório do projeto. Esse arquivo é carregado no DSP pelo menu *File* → *Load Program*. Depois de carregado, o C6713 executa o programa pelo comando *Debug* → *Run* e para a execução com em *Debug* → *Halt*.

```
-------------------------  novo_projeto.pjt - Debug  -------------------------
[c6713dskinit.c] "G:\CCStudio_v3.1\C6000\cgtools\bin\cl6x" -g -fr"G:/CCStudio_v3

[Vectors_poll.asm] "G:\CCStudio_v3.1\C6000\cgtools\bin\cl6x" -g -fr"G:/CCStudio_v

[novo_arquivo.c] "G:\CCStudio_v3.1\C6000\cgtools\bin\cl6x" -g -fr"G:/CCStudio_v3

[Linking...] "G:\CCStudio_v3.1\C6000\cgtools\bin\cl6x" -@"Debug.lkf"
<Linking>

Build Complete,
  0 Errors, 0 Warnings, 0 Remarks.
```

Figura 19. Resultado da construção do projeto.

Para verificar se o DSP está executando o programa corretamente, ligue um gerador de sinais à entrada LINE IN e um osciloscópio à saída LINE OUT. Ligue o terra do osciloscópio à base do conector P2 e a ponta de prova na sua extremidade, o canal esquerdo. Este canal é a saída padrão, será apresentado ainda neste tutorial como alterar o canal de saída ou utilizar os dois canais.

## 4.5 INTERRUPÇÃO E POLLING

Os programas escritos para o DSP podem ser baseados em interrupção ou *polling*. Um programa baseado em *polling* fica constantemente testando se tem sinal para ser transmitido ou recebido. Já um programa baseado em interrupção, o processador para de fazer o que fazendo para executar o código da interrupção.

O exemplo anterior utiliza *polling*, a função **comm_poll()** é utilizada para iniciar o DSP, CODEC e McBSP e o programa é escrito dentro de um laço infinito, *while*(1), na função *main*. Para transformá-lo em um programa que utiliza interrupção, modifique a função de inicialização para **comm_intr()** e escreva o código dentro da função **interrup void c_int11()**, como indicado na Figura 20. Outra modificação é adicionar o arquivo suporte Vectors_intr.asm no lugar do Vectors_poll.asm. É utilizada a rotina de serviço de interrupção *c_int11* porque é o seu endereço que está indica no arquivo Vectors_intr.asm. O DSP possui outras interrupções que são detalhadas no documento TMS320C6000 Peripherals Reference Guide. (TEXAS INSTRUMENTS, 2001).

```
#include "dsk6713_aic23.h"
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;

interrupt void c_int11()          //rotina da interrupção
{
   short sample_data;

   sample_data = input_sample(); //entrada da amostra
   output_sample(sample_data);   //saída da amostra
   return;
}

void main()
{
  comm_intr();                     //inicia DSK, codec, McBSP
  while(1);                      //loop infinito
}
```

Figura 20. Exemplo de programa com interrupção.

## 4.6 USAR A ENTRADA DO MICROFONE E MODIFICAR O GANHO

O ganho do sinal de entrada em LINE IN é modificado no arquivo cabeçalho de suporte c6713dskinit.h. É aconselhável copiar este arquivo e o c6713dskinit.c da pasta *Support* para a pasta do seu projeto com o objetivo de deixar os arquivos originais sem modificação.

Os valores de alguns registradores são o que determinam as configurações do CODEC. O registrador 0 controla o volume do canal esquerdo e o registrador 1, do canal direito. Eles inicialmente contêm o valor 0x0017, aumentando este valor até 0x001c a amplitude do sinal será aumentada.

Para selecionar a entrada a partir do microfone (MIC IN), é necessário modificar o registrador 4 de 0x0011 para 0x0015. A entrada padrão e o microfone são multiplexados, apenas um dos dois pode ser ativo por vez. A Figura 21 mostra a parte do arquivo c6713dskinit.h onde se encontram os valores desses registradores.

```
DSK6713_AIC23_Config config = { \
    0x0017, /* Set-Up Reg 0       Left line input channel volume control */
            /* LRS      0         simultaneous left/right volume: disabled */
            /* LIM      0         left line input mute: disabled */
            /* XX       00        reserved */
            /* LIV      10111     left line input volume: 0 dB */

    0x0017, /* Set-Up Reg 1       Right line input channel volume control */
            /* RLS      0         simultaneous right/left volume: disabled */
            /* RIM      0         right line input mute: disabled */
            /* XX       00        reserved */
            /* RIV      10111     right line input volume: 0 dB */

    0x01f9, /* Set-Up Reg 2       Left channel headphone volume control */
            /* LRS      1         simultaneous left/right volume: enabled */
            /* LZC      1         left channel zero-cross detect: enabled */
            /* LHV      1111001   left headphone volume: 0 dB */

    0x01f9, /* Set-Up Reg 3       Right channel headphone volume control */
            /* RLS      1         simultaneous right/left volume: enabled */
            /* RZC      1         right channel zero-cross detect: enabled */
            /* RHV      1111001   right headphone volume: 0 dB */

    0x0011, /* Set-Up Reg 4       Analog audio path control */
            /* X        0         reserved */
            /* STA      00        sidetone attenuation: -6 dB */
            /* STE      0         sidetone: disabled */
            /* DAC      1         DAC: selected */
            /* BYP      0         bypass: off */
            /* INSEL    0         input select for ADC: line */
            /* MICM     0         microphone mute: disabled */
            /* MICB     1         microphone boost: enabled */
```

Figura 21Configuração dos registradores no arquivo c6713dskinit.h

## 4.7 USAR OS DOIS CANAIS DO ESTÉREO

O CODEC AIC23 tem capacidade de trabalhar usando dois canais, tanto na entrada como na saída. Um adaptador P2 macho/RCA fêmea, como o apresentado na Figura 22, é utilizado para separar os canais do conector P2.

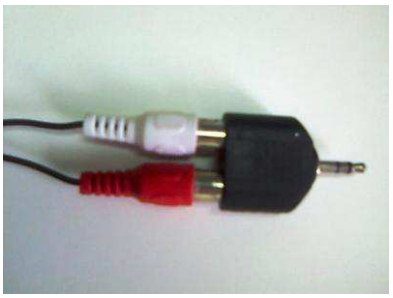

Figura 22. Adaptador P2 macho/RCA fêmea

Para usar esse recurso, crie uma variável tipo *union*. Uma *union* (união) determina que uma única localização de memória possa ter representações diferentes. A *union* criada deve ter uma representação tipo Uint32, para guardar o valor amostrado, e a outra deve ser um vetor de duas posições do tipo *short*, para determinar qual canal será usado. Um exemplo para esta variável é:

Union {Uint32 combo; short channel[2];} AIC23_data;

Neste exemplo, o nome da *union* é AIC23_data e ela possui uma variável denominada combo para receber o sinal de entrada e outra denominada *channel* para indicar o canal.

As funções para determinar o canal de saída são: output_left_sample() e output_right_sample(), para saída pelo canal esquerdo e direito respectivamente.

O código da Figura 23 mostra um exemplo em que o sinal de entrada é recebido pelo canal esquerdo e a saída é enviada também pelo canal esquerdo.

```
#include "dsk6713_aic23.h"
Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;

#define LEFT    0
#define RIGHT   1
union {Uint32 combo; short channel[2];} AIC23_data;

interrupt void c_int11()                          //rotina de interrupção
{
    AIC23_data.combo = input_sample();            //amostra recebida

    output_left_sample(AIC23_data.channel[LEFT]); //entrada e saída pelo
                                                  //canal esquerdo
    return;
}

void main()
{
    comm_intr();
    while(1);
```

Figura 23. Código com entrada e saída estéreo.

## 4.8 CONTROLAR OS LEDS E DIP SWITCHES

Os LEDs e chaves (DIP switches) do DSK que podem ser programados são de grande utilidade para acompanhar e controlar os programas. Para utiliza-los, acrescente os arquivos cabeçalhos dsk6713_led.h e dsk6713_dip.h.

A função DSK6713_LED_init() inicia o módulo dos LEDs e deve ser utilizada antes das outras funções do LED, ela não precisa de parâmetros. As outras funções disponíveis são:

- **DSK6713_LED_on()**: Liga o LED e o parâmetro é o número correspondente ao LED (0, 1, 2 ou 3). Exemplo: DSK6713_LED_on(1), deixa o LED 1 acesso.
- **DSK6713_LED_off()**: Desliga o LED e o parâmetro é o número correspondente ao LED. Exemplo: DSK6713_LED_off(3), apaga o LED 3.
- **DSK6713_LED_toggle()**: Muda o estado do LED entre ligado e desligado, o parâmetro também é o número correspondente ao LED.

A função DSK6713_DIP_init() inicia o DIP switch, ela não possui parâmetros e deve ser utilizada antes de chamar a outra função que é DSK6713_DIP_get(). Esta recebe como parâmetro o número correspondente à chave (0, 1, 2 ou 3) e retorna 0 caso a chave esteja pressionada ou 1 caso ela esteja na posição normal.

O exemplo da Figura 24 mostra uma utilização dos LEDs e DIP switches.

```
if (DSK6713_DIP_get(0) == 0){ // se ch #0 for pressionada
     DSK6713_LED_on(3);       // liga LED #3
     DSK6713_LED_off(2);      // disliga LED #2
     DSK6713_LED_toggle(1);   // muda o estado do LED #1
}
```

Figura 24. Funções para os LEDs e as chaves.

## 4.9 MODIFICAR A FREQUÊNCIA DE AMOSTRAGEM

O CODEC AIC23 é ideal para trabalhar com áudio. Para englobar todos os formatos de áudio digital, a sua taxa de amostragem pode ser modificada facilmente. No início do programa fonte principal existe a seguinte linha:

Uint32 fs = DSK6713_AIC23_FREQ_8KHZ.

Neste caso, a frequência de amostragem é 8 kHz. Modifique o valor da variável fs de acordo com a Tabela 1. Essa variável é utilizada como parâmetro na função DSK6713_AIC23_setFreq(), no arquivo C6713dskinit.c.

*Tabela 1. Frequências de amostragem disponíveis.*

| **Variável fs** | **Frequência** |
|---|---|
| DSK6713_AIC23_FREQ_8KHZ | 8,0 kHz |
| DSK6713_AIC23_FREQ_16KHZ | 16,0 kHz |
| DSK6713_AIC23_FREQ_24KHZ | 24,0 kHz |
| DSK6713_AIC23_FREQ_32KHZ | 32,0 kHz |
| DSK6713_AIC23_FREQ_44KHZ | 44,1 kHz |
| DSK6713_AIC23_FREQ_48KHZ | 48,0 kHz |
| DSK6713_AIC23_FREQ_96KHZ | 96,0 kHz |

# 5 CONCLUSÃO

O estágio supervisionado realizado no Laboratório de Automação e Processamento de Sinais foi satisfatório. Iniciou-se o estudo do DSP TMS320C6713, que deve substituir o antigo modelo utilizado no laboratório, o TMS320C6711.

O DSK TMS320C6713 mostrou-se mais eficiente e adequado para ser utilizado nas aulas práticas da disciplina de Processamento Digital de Sinais, substituindo o DSK TMS320C6711, principalmente por não apresentar falhas na comunicação, comum no modelo antigo, e por possibilitar a mudança na taxa de amostragem.

O C6713 possui uma boa documentação. Os manuais disponibilizados pela *Texas Instruments* e pela *Spectrum Digital* facilitaram a elaboração deste tutorial.

# BIBLIOGRAFIA

CHASSAING, R. **Digital Signal Processing and Applications with the C6713 and C6416 DSK**. New Jersey: Wiley, 2005.

KEHTARNAVAZ, N. **Real-Time Digital Signal Processing**. Oxford: Elsevier, 2005.

MEDEIROS, R. J. V. D. **Introdução ao Processador Digital de Sinais TMS320C6711**. Universidade Federal de Campina Grande. Campina Grande. 2007.

SPECTRUM DIGITAL. **TMS320C6713 DSK Technical Reference**. Stanfford. 2003.

TEXAS INSTRUMENTS. **TMS320C6000 Code Composer Studio Tutorial**. Dallas. 2000.

TEXAS INSTRUMENTS. **TMS320C6000 Peripherals Reference Guide**. Dallas. 2001.

TEXAS INSTRUMENTS. **TMS320C6000 Programmer's Guide**. Dallas. 2010.