INTRUMENTAÇÃO
LAB 05
Determinação da curva de não-linearidade integral (INL) com a placa de aquisição de dados Keithley DAS-1600

(*http://diana.uceh.ualg.pt/Inst/lab05.pdf***)**

1. Introdução

Com este projecto pretende-se obter as especificações estáticas mais importantes de um conversor digital-analógico (DAC) e de um conversor analógico-digital (ADC): a especificação de não-linearidade integral (INL) e a especificação de não-linearidade diferencial (DNL). Simultaneamente pretende-se familiarizar o aluno com uma placa de aquisição de dados típica: a placa Keithley DAS-1600.

1.1 DAC

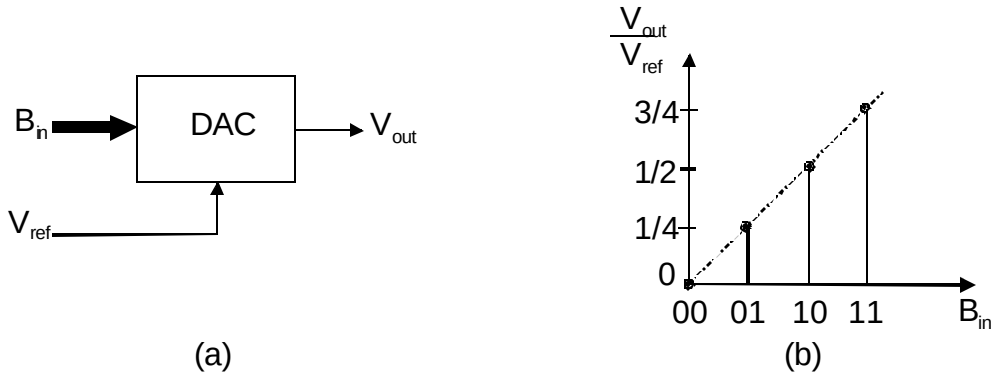


Figura 1: Conversor digital analógico (DAC). (a) Diagrama de bloco. (b) Curva de tranferência ideal para um DAC de 2 bits

A Figura 1(a) mostra um diagrama de blocos de um DAC. $\mathbf{B}_{in}$ é a palavra digital de entrada

$$B_{in} = b_0\, 2^0 + b_1\, 2^1 + \Lambda + b_{N-1}\, 2^{N-1}$$

e V_{out} é o valor analógico à saída

$$V_{out} = \frac{V_{ref}}{2^N} B_{in}$$
$$= V_{ref}\left(b_0 2^{-N} + b_1 2^{-N+1} + \Lambda + b_{N-1} 2^{-1}\right)$$

e V_{ref} é o valor da tensão de referência do DAC.

É habitual definir-se o valor do bit menos significativo (LSB):

$$1\text{LSB} = \frac{V_{ref}}{2^N}$$

A tensão de saída do DAC é assim o valor de 1 LSB multiplicado pelo número representado pela palavra digital de entrada. Notar **que a tensão de saída do DAC é sempre uma fracção da tensão de referência**.

A título de exemplo mostra-se na Figura 1(b) a curva **característica estática ideal** de um DAC de 2 bits.

A **curva característica estática real** de um DAC é bem mais complicada, como se pode ver na Figura 2.

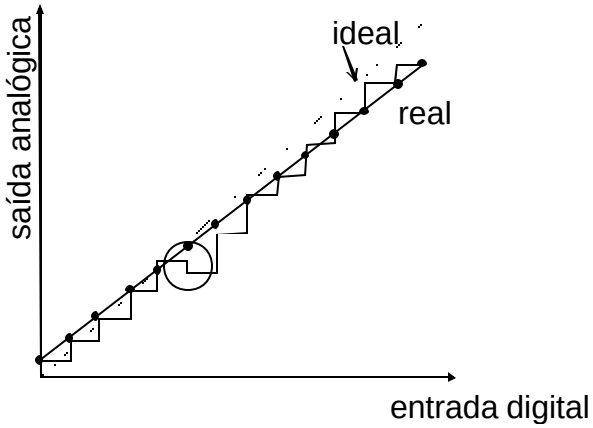


Figura 2: Característica estática real de um DAC

Para além do erro de offset (que traduz que a saída não é zero Volt quanto a entrada é o numero 0…00, e do erro de ganho (que traduz que o declive da recta ideal é diferente do da curva real), o DAC apresenta erros de não linearidade:

1. DNL (não linearidade diferencial). Dois códigos de entrada adjacentes devem produzir sinais analógicos que distam exactamente de 1 LSB. A especificação de DNL é a maior diferença entre dois valores medidos adjacentes e o valor ideal de 1LSB

$$\text{DNL(j)} = \left[V_{out}(j+1) - V_{out}(j)\right] - 1LSB$$
$$\text{DNL} = \max |\ \text{DNL(j)}\ |$$

Um conversor **não é monotónico** se o DNL é maior que 1LSB. Vê-se na Figura 2 um caso em que a entrada aumentou (de 1 bit) e a saída em vez de aumentar 1 LSB **baixou**. Conversores não monotónicos são potencialmente perigosos em circuitos com realimentação porque podem provocar instabilidade.

2. INL (não linearidade integral). É a maior diferença entre o valor real e o valor da recta definida pelos dois pontos extremos da curva real

$$INL(j) = V_{out\_real}(j) - V_{out\_recta}(j)$$
$$INL = \max | INL(j) |$$

Se o valor de | INL(j) | para qualquer j for sempre inferior a 1/2 LSB então podemos afirmar então podemos afirmar que um DAC com N bits de resolução também tem N bits de linearidade. Esta é a situação ideal.

1.2 ADC

No caso de um conversor analógico digital (ADC), o seu diagrama de bloco é um espelho do DAC: como se mostra na Figura 3 a entrada é um sinal analógico e a saída é uma palavra digital.

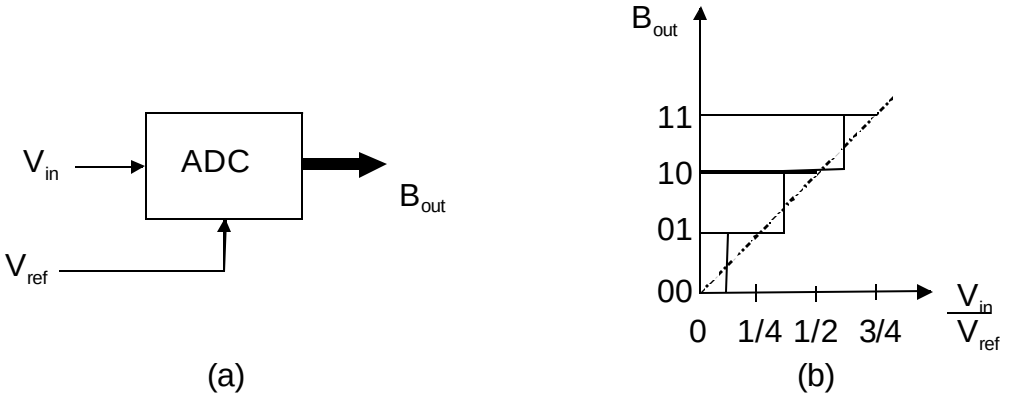


Figura 3: Conversor analógico digital (ADC). (a) Diagrama de blocos. (b) Curva característica entrada saída.

Agora a saída está relacionada com a entrada pela seguinte equação

$$\frac{V_{ref}}{2^N} B_{out} = V_{in} \pm V_x$$
$$V_{ref}\left(b_0 2^{-N} + b_1 2^{-N+1} + \Lambda + b_{N-1} 2^{-1}\right) = V_{in} \pm V_x$$
$$-\frac{1}{2}\mathrm{LSB} \le V_x \le +\frac{1}{2}\mathrm{LSB}$$

Notar que existe uma gama de valores de entrada $V_{in} \pm V_x$ à qual corresponde o mesmo código digital de saída. Esta ambiguidade designa-se por *erro de quantificação* que é, idealmente, no máximo igual a $\pm 1/2\mathrm{LSB}$.

Num conversor A/D real o erro de quantificação pode ser para alguns sinais de entrada mesmo superior a 1 LSB: neste caso alguns códigos digitais nunca aparecem à saída do conversor: diz que o conversor tem *missing codes*.

As especificações de DNL e INL de um ADC são equivalentes às de um DAC. Um conversor A/D é garantido não ter *missing codes* se o erro DNL máximo é inferior a 1LSB ou alternativamente se o erro de INL máximo é inferior a 0.5 LSB. Neste caso pode também afirmar-se que um ADC de N bits de resolução também tem N bits de linearidade—é o caso ideal.

2. Montagem experimental

O objectivo deste trabalho é determinar as especificações de DNL e INL de um (dos dois) conversor D/A da placa Keithley DAS-1600.

A placa Keithley DAS-1600 é composta basicamente por
- um conversor analógico digital de 12 bits de resolução
- um multiplexer de 16 canais de entrada *single-ended* (8 canais diferenciais)
- dois conversores digitais analógicos de 12 bits de resolução
- um relógio de tempo real
- três portas digitais de entrada/saída (com 8 bits cada)

De toda esta funcionalidade apenas vamos precisar de trabalhar com o multiplexer (para seleccionar um canal), com o ADC, e com um dos DACs.

A placa Keithley DAS-1600 está préviamente configurada com os seguintes parâmetros:

Opção	*Configuração*
Board number	0
Board name	DAS 1602
Address	0x300
DMA channel	3
Interrupt	7
Clock timebase	10 MHz
Wait State	No
A/D mode	unipolar
multiplexer configuration	single-ended
DAC 0 mode	unipolar
DAC 1 mode	unipolar
DAC 0 ref	-5 V
DAC 1 ref	-5 V

Não é necessário configurar nenhum jumper ou interruptor na placa nem voltar a correr o programa de configuração (`CFG1600.exe`).

A placa Keithley DAS-1600 está ligada a uma *caixa de terminais* (STA -16).

Neste projecto vamos utilizar o DAC 0 e o canal 0 do multiplexer.

Uma vez que o DAC e o multiplxer têm linha de referência comum, basta ligar com um fio o terminal de saída do DAC 0 (`D/A 0 OUT`) com o terminal de entrada 0 do multiplexer (`CH 0 HI`)—ver Figura 4.

Figura 4: Terminais na caixa STA-16

3. Aquisição do sinal

O programa de aquisição de dados pode ser feito em QuickBasic (`c:\QB`) ou em Borland C (`c:\bc`)

3.1 Programa em Quick Basic

```
REM DAS1600 example program
10 INPUT "DAC input code [0, 4095]: ", code
12 hbyte = FIX(code / 16)
13 lbyte = (code AND 15) * 16
REM write low byte to DAC 0
14 OUT &H304, lbyte
REM write high byte to DAC
15 OUT &H305, hbyte
REM input channel is channel 0
20 N = 0
REM write in MUX scan register (base address+2) and select channel 0
30 OUT &H302, 0
REM write in base address anything and start conversion
40 OUT &H300, 0
REM read status register A (base address+8) and wait for end of
conversion
50  a = INP(&H308)
70 IF a < 128 THEN 80 ELSE 50
REM read low byte (base address) and high byte (base address+1)
80 lbyte = INP(&H300)
90 hbyte = INP(&H301)
```

```
REM convert to decimal
REM shift low byte 4 positions to the right
REM shift high byte 4 positions to the left
100 result = hbyte * 16 + lbyte / 16
110 PRINT "ADC result = "; result
```

Alguns comentários ajudam a compreender o programa

- a placa Keithley DAS-1600 está instalada no PC com o endereço 0x300 (em hexadecimal)
- O multiplexer é controlado por um registo de 8 bits que tem o endereço 0x302:

B7	B6	B5	B4	B3	B2	B1	B0
AH3	AH2	AH1	AH0	AL3	AL2	AL1	AL0

AL0 to AL3 = start of scan address
AH0 to AH3 = end of scan address

- O DAC 0 é controlado por dois registos de 8 bits com os endereços 0x304 e 0x305:

B7	B6	B5	B4	B3	B2	B1	B0
D3	D2	D1	D0	x	x	x	x

(a) low byte (0x304)

B7	B6	B5	B4	B3	B2	B1	B0
D11	D10	D9	D8	D7	D6	D5	D4

(a) high byte (0x305)

(logo a necessidade de operações de baixo nível nas linhas 12 e 13 do programa para converter de decimal em binário natural com este formato)

- O endereço da placa 0x300 coincide com o endereço do conversor A/D. Sempre que se escreve neste registo o conversor A/D inicia imediatamente uma conversão
- Para se saber quando a conversão acabou lê-se o Bit 7 do registo de status no endereço 0x308
- Para se saber o resultado da conversão lê-se o endereço 0x300 (low byte) e o endereço 0x301 (high byte): na primeiro obtem-se os bits menos significativos e na segundo os bits mais significativos no formato indicado na Figura 5. Logo é necessário fazer alguns "shifts" para a esquerda e para a direita para pôr os bits no sitio certo…

B7	B6	B5	B4	B3	B2	B1	B0
D3	D2	D1	D0	0	0	0	0

(a) registo 0x300 -low byte

B7	B6	B5	B4	B3	B2	B1	B0
D11	D10	D9	D8	D7	D6	D5	D4

(a) registo 0x301 - high byte

Figura 5: Formato dos bits retornados pelo ADC

3.2 Programa em C

O programa de aquisição dos dados feito na linguagem Borland C (`c:\bc`) serve-se de uma livraria (e de um device-driver) distribuidos com a placa, que fornece um conjunto de subrotinas de alto nível que escondem do programador todos os detalhes do hardware.

De um conjunto de subrotinas muito completo, é importante destacar as seguintes:

```
K_OpenDriver        inicializa a placa DAS-1600
K_GetDevHandle      estabelece comunicação com a placa DAS-1600
K_DAWrite           escreve no registo que controla o DAC
K_ADRead            le o  registo com o resultado da conversão do ADC
K_CloseDriver fecha a ligação com a placa
```

Há muito mais rotinas na livraria. Consulta o manual [2], ***mas não leves o manual para fora do laboratório!***

Mostra-se de seguida um programa exemplo:

```
/*****************************************************************************/
/*                   'C' - Single D/A & A/D Example                          */
/*                                                                           */
/* The following is an example program that demonstrates the steps required*/
/* to perform a single D/A conversion followed by a single A/D conversion  */
/* under program control.  To observe predictable results, connect the DAC */
/* output channel to the Analog Input channel used.                          */
/*                                                                           */
/* There are several factors to consider when converting the count value   */
/* that is passed to K_DAWrite() and returned from K_ADRead() functions:   */
/*      A/D Unipolar/Bipolar switch setting,                                 */
/*      DAC Unipolar/Bipolar switch setting,                                 */
/*      DAC Reference voltage (vs. Gain parameter set for K_ADRead).         */
/*         [Refer to the User's guide or Function Call Driver manual       */
/*          on converting volts->counts & counts->volts.]                   */
/*                                                                           */
/*                                                                           */
/* Refer to EXAMPLES.TXT for detail on the supported C languages.           */
/*                                                                           */
/*****************************************************************************/

/* C INCLUDE FILES */
#include "stdio.h"
#include "stdlib.h"

/* DAS-1600/1400/1200 DRIVER INCLUDE FILES */
#include "dasdecl.h"
#include "das1600.h"

/* LOCAL VARIABLES */
    DWORD       hDrv1600;           /* Driver Handle */
    DWORD       hDev1600;           /* Device Handle */
    short       nErr;               /* Function return error flag */
```

```
    WORD        wADval;             /* Strorage for A/D value */
    DWORD       dwDAval;            /* Strorage for D/A value */



/* BEGIN MAIN MODULE */

main()
{
/* PRINT SOME TEXT */
    printf ("                    'C' - Single D/A & A/D Example\n\n");
    printf ("The following is an example program that demonstrates the steps
required\n");
    printf ("to perform a single D/A conversion followed by a single A/D
conversion  \n");
    printf ("under program control.  To observe predictable results, connect the
DAC \n");
    printf ("output to the Analog Input channels used. \n");

/* FIRST STEP IS TO INITIALIZE THE HARDWARE/SOFTWARE */
    if (( nErr = K_OpenDriver("DAS1600", "das1600.cfg", &hDrv1600)) != 0)
    {
        printf( " Error %X during K_OpenDriver ", nErr ) ;
        exit(nErr);
    }

/* ESTABLISH COMMUNICATION WITH THE DRIVER THOUGH A DEVICE HANDLE */
    if ( ( nErr = K_GetDevHandle(hDrv1600, 0, &hDev1600) ) != 0)
    {
        printf( " Error %X during K_GetDevHandle ", nErr ) ;
        exit(1);
    }

/* THE DAC VALUE TO OUTPUT MUST BE CORRECTLY ALIGNED FOR OUTPUT; I.E. THE */
/* ACTUAL 12-BIT DATA MUST BE IN THE UPPER 12 BITS OF THE USER INTEGER */
/* The volt value of the DAC depends on the DAC settings */
    printf( "\nEnter DAC output value [0,4095]: ") ;
    scanf("%lu", &dwDAval) ;

    dwDAval = (dwDAval << 4) & 0xFFF0;

/* OUTPUT dwDAval TO DAC #0 */
    if ((nErr = K_DAWrite (hDev1600, 0, dwDAval)) != 0)
    {
        printf ("Error %X in K_DAWrite operation.\n", nErr);
        exit(1);
    }
    else
    {
        printf ("D/A value to DAC 0 is : %lu\n", (dwDAval >> 4) & 0xFFF);
    }

/* SAMPLE AND READ CHANNEL 0 AT GAIN 1 AND STORE SAMPLE IN wADval */
    if ((nErr = K_ADRead (hDev1600, 0, 0, &wADval)) != 0)
    {
        printf ("Error %X in K_ADRead operation.", nErr);
        exit(1);
```

```
    }

/* STRIP LEAST-SIGNIFICANT 4 BITS AND DISPLAY wADval */
    printf ("A/D value from channel 0 is : %u\n", (wADval >> 4) & 0xfff);

/* CLOSE THE DRIVER AND RELEASE ALL RESOURCES */
    K_CloseDriver(hDrv1600) ;

    return(0);

}
```

O programa é tão simples que dispensa comentários…

Talvez mais importante é dar algumas dicas sobre como compilar correctamente o programa.

3.3 Compilação do programa

Primeiro cria uma pasta `c:\users\a999999` (onde a999999 é o teu numero) onde vais trabalhar. *Este pormenor é importante para não se instaurar a anarquia total no disco duro…*

- Copia para essa pasta o ficheiro de configuração `DAS1600.CFG` (que se encontra em `c:\DAS1600`).

- Abre um novo projecto (Project -> Open…) e dá-lhe um nome apropriado. Inclui no novo projecto o teu programa em C e as livrarias necessária, como se mostra na Figura 6.

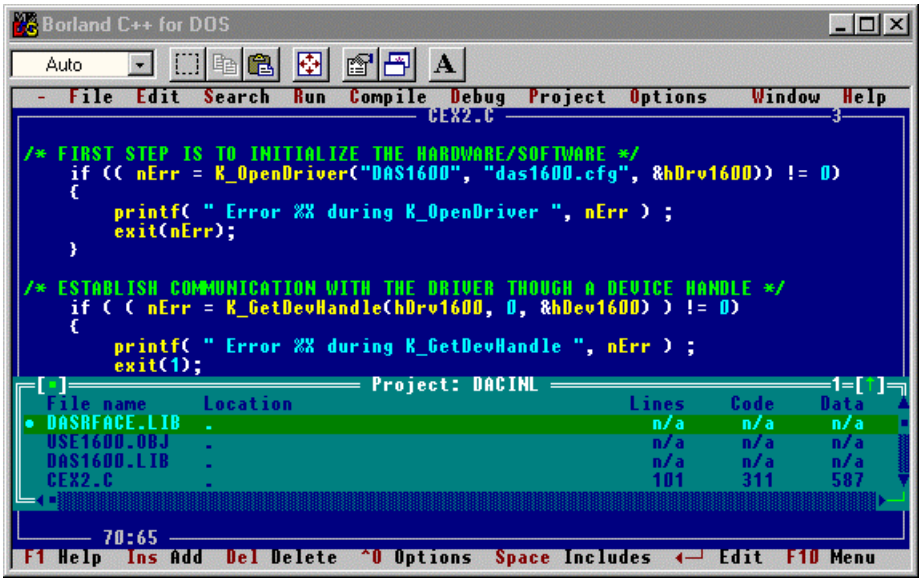


Figura 6: O ambiente de trabalho em Borland C (versão DOS)

Agora vai a Options->Directories e actualiza o conteúdo com o nome das pastas pertinentes, como se mostra na Figura 7.

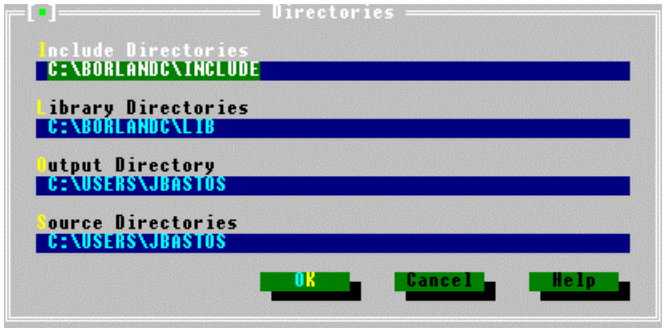


Figura 7: Actualização das pastas

De seguida actualiza as opções de compilação, como se mostra na Figura 8.

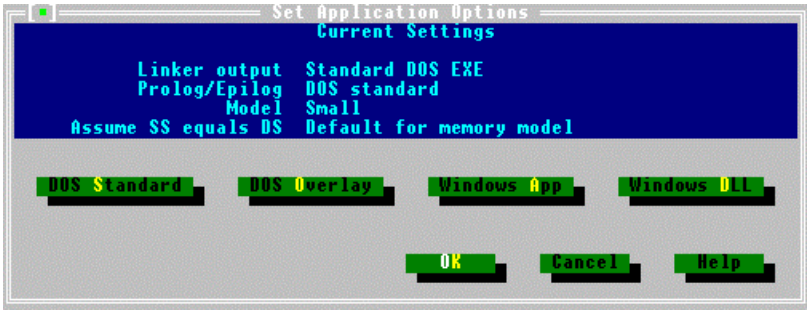


Figura 8: Opções de compilação

Não te esqueces de seleccionar a geração de código SMALL, como se mostra na Figura 9.

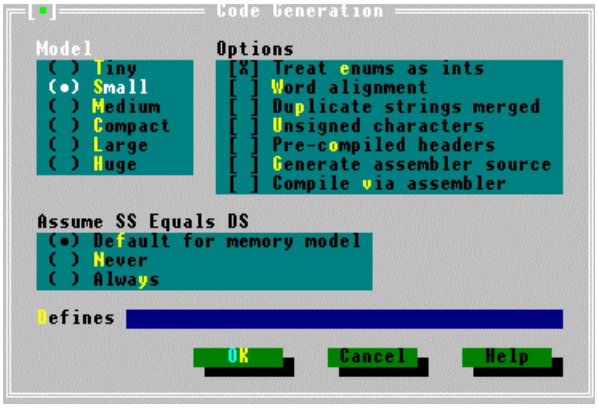


Figura 9: Escolha de código SMALL

e de informares o compilador que o computador tem co-processador numérico, como se mostra na Figura 10.

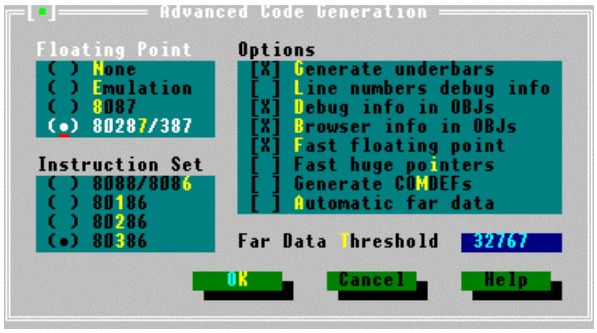


Figura 10: Opções de compilação

Não te esqueças também de definir que queres apenas ligar (*link*) o teu programa com livrarias estáticas, como se mostra na Figura 11.

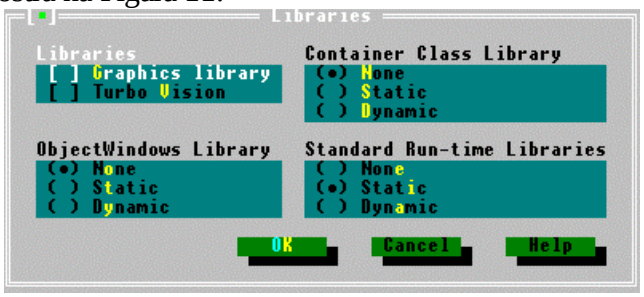


Figura 11: Opções para o linker

Por fim guarda (Options-> Save) o teu ambiente de trabalho. Se não te esqueceste de nenhum detalhe, o teu programa deve agora compilar sem problemas…

Corre o teu programa numa janela de DOS (File -> DOS Shell).

4. Aquisição de dados

O teu programa em C (ou BASIC) deve ter varrer sequencialmente todos os códigos (de 0 a 4095) do DAC e guardar num ficheiro o código correspondente proveniente da conversão do ADC.

Entre cada código é ideal fazer um tempo de espera da ordem de centenas de mili-segundos, de forma a permitir ao DAC estabilizar perfeitamente o valor analógico à saída.

5. Processamento dos dados

Uma vez guardados os dados, o seu processamento pode ser feito em Matlab, Excel, etc. Apenas deves ter tido o cuidado de ter escolhido o caracter separador de colunas conveniente para o programa preferido (tab, virgula, espaço, etc.)

Essencialmente o que se pretende é, a partir de uma tabela de duas colunas com os dados, (código DAC, código ADC), construir uma terceira coluna com os valores dados por uma recta ideal y_{ideal}=mx+b definida pelos dois pontos extremos da curva adquirida, isto é

ponto1	código DAC xmin=0	código ADC	ymin
ponto 2	código DAC xmax=4095	código ADC	ymax

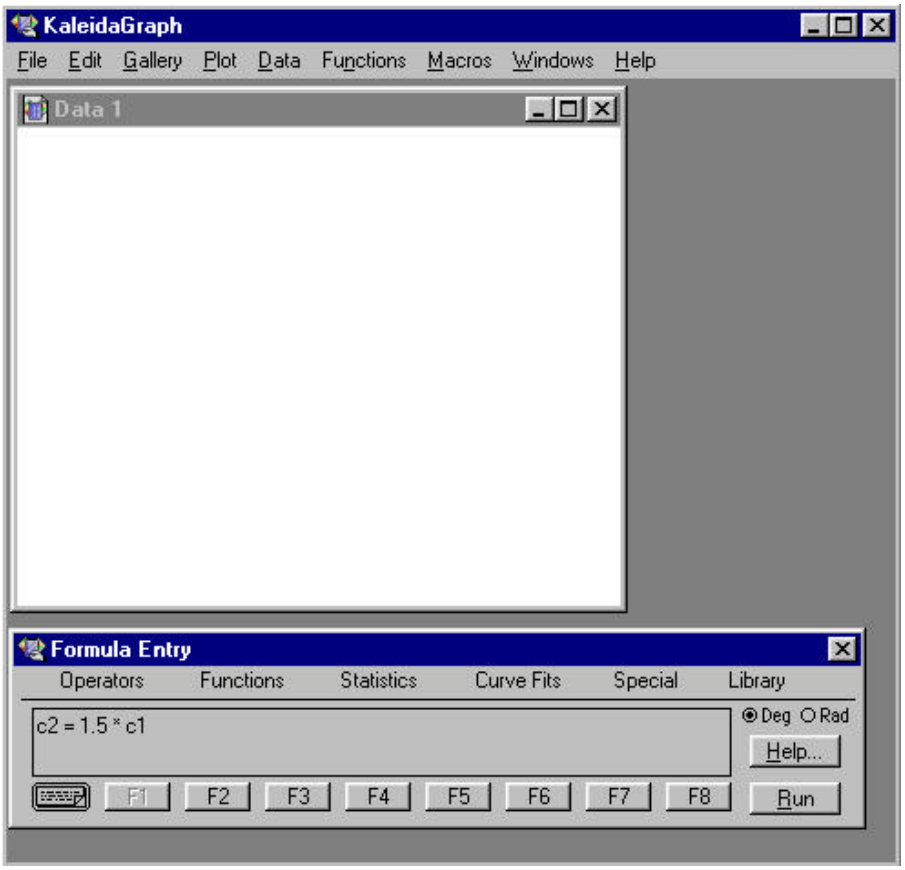
A ordenada na origem b é obviamente ymin e o declive é b=(ymax-ymin)/4095

Agora pode-se construir uma 4a coluna que é a curva de não linearidade integral (INL) desejada

$$INL(j) = y_{real}(j) - y_{recta}(j)$$

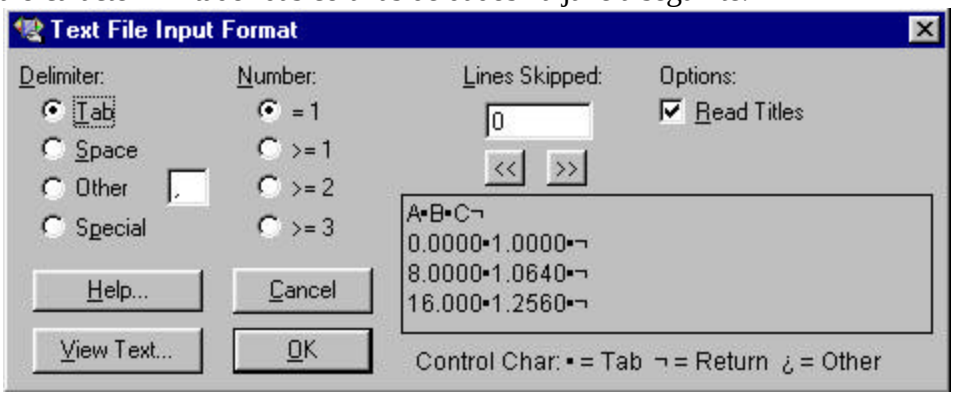
Utilizando a 1a coluna como variavel independente (x) e a 4a coluna como variavel dependente (y) pode-se ver gráficamente a curva INL do conversor D/A.

A titulo de exemplo vai-se utilizar um programa muito simples: **kaleidagraph.** É semelhante ao Excel mas mais simples e mais *user friendly*.



Seleccione **File > Import > Text** para escolher o ficheiro onde guardou os resultados

e escolha o caracter limitador das colunas de dados na janela seguinte:



Faça um gráfico **Gallery > Linear > Line** escolhendo para eixo dos x a 1a coluna e para eixo dos y a 4a coluna (a 3a e 4a coluna são geradas fácilmente a partir da 2a coluna utilizando as funções disponiveis na janela Formula Entry)

Mostra-se de seguida na **Figura 12** um gráfico a título de exemplo…

O teu gráfico será com certeza diferente porque este foi feito com outro DAC!

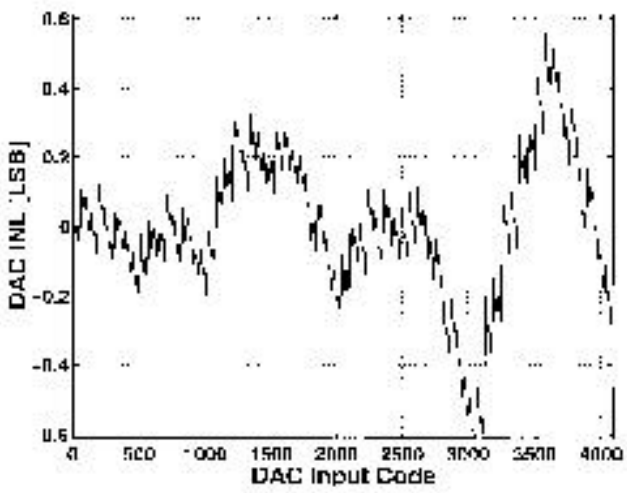


Figura 12: Curva INL de um DAC de 12 bits de resolução

A especificação de INL é o maior valor (absoluto) deste gráfico. Idealmente, para se poder afirmar que o DAC tem 12 bits de linearidade, a especificação de INL é INL<1/2 LSB.

Podes agora fácilmente traçar a curva de não linearidade diferencial (DNL) utilizando a fórmula

$$\mathrm{DNL(j)} = \left[y_{real}(j+1) - y_{real}(j) \right] - 1$$

Mostra este gráficos e os valores das especificações de INL e DNL obtidos ao instrutor.

6. Bibliografia

[1] Keithley DAS-1600/1400 Series USER'S GUIDE
[2] Keithley DAS-1600/1400 Series FUNCTION CALL DRIVER