



MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE
COMPUTADORES

SISTEMAS ELECTRÓNICOS DE PROCESSAMENTO DE SINAL

BPSK Modem

Grupo n.º 2/3

André Filipe Barroso Cerqueira n.º 65144

Guilherme Branco Teixeira n.º 70214

João André Catarino Pereira n.º 73527

segunda-feira 15h30 - 18h30, LE1

Lisboa, 17 de Abril de 2015

Conteúdo

1	Introdução	1
2	Projecto	1
2.1	Projectos de Demonstração	1
2.1.1	sine8_buf	1
2.1.2	loop_intr	1
2.2	BPSK	2
2.2.1	P1. Oscilador controlado numericamente	2
2.2.2	P2. Transmissor BPSK	7
3	Conclusão	10
4	Anexos	10

¹As linhas de código apresentadas durante o relatório têm como objectivo demonstrar a maneira de raciocinar na resolução de problemas, não representando uma cópia exata do código usado em laboratório, podendo até, serem consideradas *pseudo-código*

1 Introdução

Este trabalho consiste na primeira parte do projecto de laboratório da cadeira: desenvolvimento de um modem de *Binary Phase Shift Keying* (BPSK).

Teve como objectivo a familiarização com o ambiente de desenvolvimento integrado de DSP que consiste nas placas de desenvolvimento DSK TMS320C6416 e DSK TMS320C6713 da *Texas Instruments* e no software de desenvolvimento *CodeComposerStudiov5.5*. Para tal correram-se dois projectos exemplo (sine8_buf e loop_intr) e, usando as ferramentas de debug, alteraram-se certos parâmetros de forma a observar os efeitos nos sinais resultantes. Também se consolidaram os conhecimentos adquiridos desenvolvendo dois mini projectos: um oscilador sinusoidal controlado numericamente e o modulador do modem BPSK.

2 Projeto

2.1 Projectos de Demonstração

- Resumo das funções e os seus objectivos com especial importancia ao loop
- Relaciona-las com as suas utilizações no projecto em si

2.1.1 sine8_buf

O objectivo deste projeto é representar a função sinusoidal, multiplicada por um determinado ganho, através de um conjunto de amostras que equivalem a um período da mesma, repetindo nos períodos seguintes esse mesmo conjunto. Este procedimento é realizado através da rotina de interrupção presente no programa.

Ao analisar o código deste projeto à primeira vista podemos concluir logo que este usa uma frequência de amostragem de 8 kHz , tem um ganho $G = 10$ predefinido e usa 8 amostras para representar a sinusoide. Depois de observar a sinusoide no osciloscópio variou-se o ganho a fim de perceber a sua influência e também o seu limite.

Para compreender o limite desta sinusoide é necessário ter em conta que se usa o formato de vírgula fixa Q15 para as amostras da sinusoide. Este formato tem como limite o valor $(2^{15} - 1) = 32767$. Considerando o valor máximo da sinusoide, se multiplicarmos a mesma por um ganho G=33 obtemos um valor superior ao permitido pelo formato Q15, fazendo com que nesses pontos o valor da sinusoide "caia".

2.1.2 loop_intr

Este projeto tem como objectivo fornecer-nos um template para os próximos projetos, em termos de comunicação com a placa e rotina de interrupção. Pode-se observar nas últimas linhas de código como se liga os sinais de entrada e saída aos canais da placa.

No projeto anterior observou-se os efeitos de overflow de uma variável. Neste observam-se os efeitos de aliasing (ou não, não me recordo se o DSP tem um filtro anti-aliasing à entrada) devido ao sinal de line-in ter a mesma frequência que a frequência de amostragem. Se havia anti-aliasing, a partir dos 4khz deixariamos de ver uma sinusoide com os 3.3V. Não fizemos a experiência de mudar para sinal quadrado e variar a frequência, mas provavelmente não íamos ver um sinal quadrado pois o espectro (infinito) deste teria que ser filtrado pelo anti-aliasing filter.

2.2 BPSK

Este projeto é constituído por duas partes, a primeira é um oscilador controlado numericamente e a segunda é um transmissor BPSK.

2.2.1 P1. Oscilador controlado numericamente

Usa-se o projeto descrito na secção 2.1.2 como base para a construção de um oscilador controlado numericamente (*NCO*). Um *NCO* é um gerador de sinal digital que cria uma forma de onda discretamente representada no tempo e na amplitude. O *NCO* terá as seguintes características:

Tabela 1: Características do *NCO* a construir.

Parâmetro	Símbolo	Valor
Frequência de amostragem	f_s	16kHz
Frequência mínima	f_{min}	2kHz
Frequência máxima	f_{max}	6kHz

O projeto de construir o *NCO* seguiu os seguintes passos que estão posteriormente desmitigados:

1. Oscilador de Relaxamento (Integrador de Rampa).
2. Look-up-table (LUT).
3. Obtenção do sinal sinusoidal através da LUT.
4. Criação de duas variáveis para controlar a frequência e amplitude do sinal sinusoidal.
5. Utilização do sinal de entrada para controlar a frequência do sinal.
6. Teste do oscilador com uma onda quadrada à entrada para observar o sinal modulado.
7. Melhoria da qualidade do oscilador com interpolação linear.
8. Comparaçao dos espectros com e sem interpolação.

P1-1

Visto que a placa usada no laboratório (DSK TMS320C6713) tem 1V de amplitude máxima na saída, foi usado o formato Q15 para representar o sinal de saída, pois um sinal representado em Q15 tem amplitude máxima de $2^{15} - 1 = 32767$.

Para construir um Oscilador de Relaxamento foi criado um programa que aumenta por a cada ciclo o sinal à saída de modo a que este tenha o comportamento de uma rampa. Este incremento tem de tomar um valor, que por agora se vai considerar constante, tal que 32767 seja seu múltiplo. Ao verificar a tabela 1, podemos chegar aos seguintes valores de incremento com as suas frequências correspondentes:

Tabela 2: Valores a atribuir ao incremento do sinal na saída.

Frequência de saída(f_0)	Incremento(Δ)
$2kHz$ (f_{min})	8192
$4kHz$	16384
$6kHz$ (f_{max})	24576

Os valores calculados na tabela 2 foram calculados através da seguinte fórmula:

$$f_0 = \frac{\Delta}{2A} f_s \Leftrightarrow \Delta = 2A \frac{f_0}{f_s} , A = 32767 \quad (1)$$

Foi então possível obter as seguintes rampas com diferentes frequências:

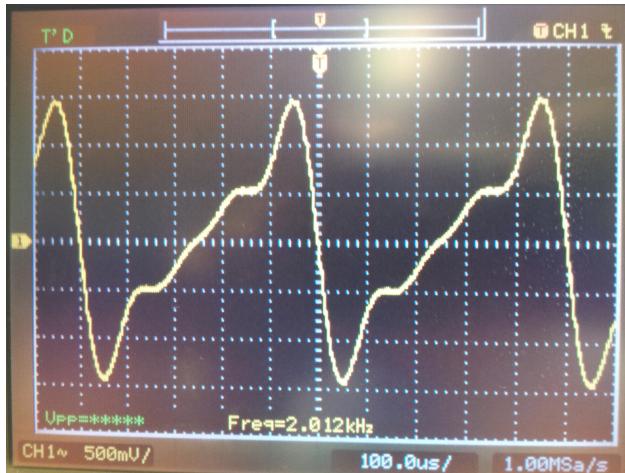


Figura 1: Rampa com frequência de $2kHz$

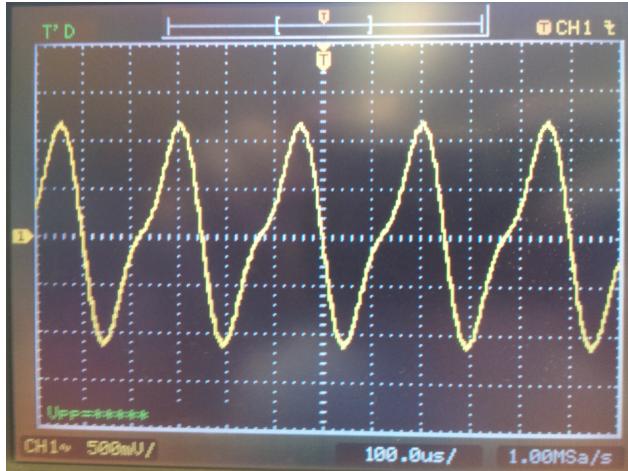


Figura 2: Rampa com frequência de $4kHz$

Ao observar as duas rampas pode-se constatar que para uma frequência mais baixa, ou seja, um incremento menor, a rampa apresenta mais "degraus" e um menor declive. Um incremento menor na rampa significa um maior número de amostras da mesma dentro do intervalo possível, o que reduz o declive da rampa.

P1-2

Foi criada uma tabela (LUT) com os valores em Q15 de meio ciclo de um seno de modo a que seja possível ir retirar os seus valores para criar um sinal sinusoidal. Esta tabela tem 32 valores e foi construída com a seguinte equação:

$$32767 * \sin \frac{n\pi}{32}, \quad n = 0, 1, 2, \dots, 31 \quad (2)$$

Para calcular apenas meio ciclo do seno, é necessário ter 32 valores pertencentes ao intervalo $[0, \pi]$ na fase. Assim, é necessário restringir a fase a esse intervalo através da divisão observada na equação, $\frac{n\pi}{32}$. O produto do seno pelo valor 32767 é a conversão do seno para Q15. Não se multiplica por $2^{15} = 32768$ pois os sinais estão em complemento para dois, o que significa que para o máximo do seno, esta LUT ultrapassaria o limite do formato, o que causaria o corte dessa amostra no sinal.

P1-3

Usando como base a rampa criada na secção P1-1 para indexação da LUT criada na secção P1-2, foi possível criar um sinal de uma sinusoide.

De acordo com o projeto, a rampa é representada num formato Q10, com 1 bit sinal, 5 bits inteiros e 10 bits fraccionários. Para a indexação são usados os 5 bits inteiros, ou seja, os 5 mais significativos (excluindo o bit de sinal) do sinal da rampa, esses 5 bits irão indexar e escolher qual o valor da tabela de LUT a usar para criar a sinusoide.

Este processo foi efetuado com o seguinte pedaço de código dentro do ciclo:

```
rampa=rampa+incremento; // Criar a rampa
index=rampa>>10;
```

```

index=31 & index;           // Usar apenas 5 bits
sinusoida=LUT[ index ];

```

Como se pode observar, para isolar os 5 bits inteiros fez-se dez *shifts* direitos de maneira a ter um sinal apenas com zeros, o bit sinal e esses 5 bits encostados à direita. A seguir aplicou-se uma máscara constituída por uma AND com 31, de maneira a retirar o bit sinal e finalmente isolar num sinal o índice pretendido da LUT. Com o índice é uma questão de aceder à tabela e obter y_1 , neste caso representado pelo sinal *sinusoida* (Figura 3).

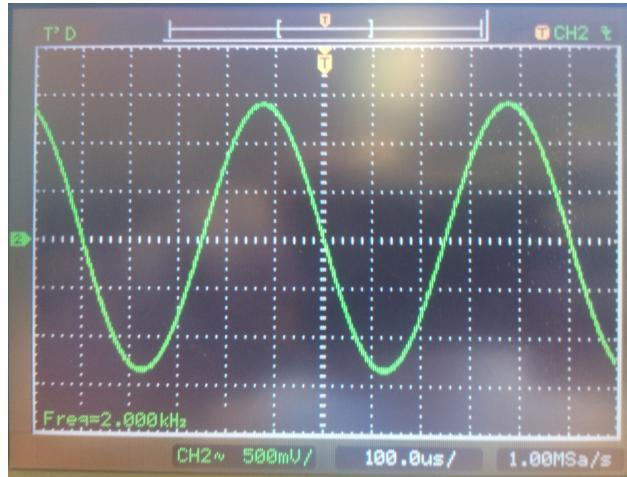


Figura 3: Sinal sinusoidal criado de 2kHz.

Como se pode observar na figura foi possível criar um sinal aproximadamente sinusoidal através de uma LUT indexada por uma rampa. Nesta figura não é possível notar mas quando se aproxima mais o sinal no osciloscópio verifica-se pequenos degraus ao longo do sinal, que simbolizam a aproximação obtida através das amostras.

P1-4

Foram criadas duas variáveis para fosse possível controlar a frequência e a amplitude do sinal à saída, *incremento* e *amplitude*, respectivamente.

A primeira variável foi já antes referida, nas secções 2.2.1 e 2.2.1. Esta variável, caso alterada iria alterar a frequência da rampa, e por consequência, a frequência do sinal de saída. Como se pode observar na tabela 2, esta variável terá como limite máximo o valor 24576 e como limite mínimo 8192. Podemos também concluir que quanto maior for esta variável, maior a frequência do sinal de saída, tal como se o seu valor diminuir, a frequência de saída irá diminuir também.

A segunda variável (*amplitude*) foi criada com o propósito de modelar a amplitude do sinal de saída, esta causou mudanças mais significativas no código, tal como se pode verificar:

```

rampa=rampa+incremento;
index=rampa>>10;
index=31 & index;
aux=amplitude*LUT[ index ];

```

```

aux=aux<<1;           // Retirar bit de sinal extra
sinusoide=-aux>>16;   // Colocar a variavel com 16 bits em Q15

```

Esta variável tem como valor máximo 32767, pois os valores afixados na tabela LUT já apresentam um valor com a amplitude máxima de Q15, não tendo assim problemas em relação ao formato de representação nem à amplitude do sinal transmitido à placa.

Deste modo foi possível ter duas variáveis, *incremento* e *amplitude*, que modelavam a frequência e a amplitude do sinal de saída respectivamente.

P1-5

Para que a frequência do sinal de saída seja modelado através da amplitude do sinal de entrada, o sinal de entrada terá de controlar a variável *incremento*.

Um exemplo de garantir que isto era possível era receber um novo valor para a variável *incremento*, sendo que este valor vinha diretamente da entrada:

```

incremento=-inbuf ;
rampa=rampa+incremento ;
( . . . )

```

Deste modo, quando se aumentava a amplitude do sinal de entrada, a variável *incremento* aumentava, e por consequência também aumentava a frequência de entrada.

P1-6

P1-7

Com o objectivo de melhorar a qualidade do sinal criado vai ser usado o método de interpolação linear descrito na seguinte equação:

$$y = y_1 + (y_2 - y_1) * \delta_x \quad (3)$$

Sendo que δ_x refere-se aos 10 bits menos significativos ignorados na altura em que se retira o valor *index* com o propósito de indexação. O processo de recolher o valor de δ_x e do cálculo da interpolação estão descrito nas seguintes linhas de código:

```

rampa=rampa+delta ;
deltaX=1023 & rampa;    // Retirar os 10 bits menos significativos
deltaX=deltaX<<5;      // Representar deltax em Q15
index=rampa>>10;
index=31 & index ;
aux=amplitude*LUT[index] ;
aux=aux<<1;
sinusoide=-aux>>16;
Y1=sinusoide ;
aux=amplitude*LUT[index+1];
aux=aux<<1;

```

```

Y2=aux>>16;
aux=(Y2-Y1)*deltaX ;
aux=aux<<1;
Y=Y1+(aux>>16);

```

Sendo que o sinal Y corresponde ao sinal com maior qualidade, ou seja, interpolado, e que o sinal *sinusoide* corresponde ao sinal com menor qualidade.

Ambos os sinais podem ser observados na imagem ??, embora a sua comparação não poderá ser efetuada na sua observação.

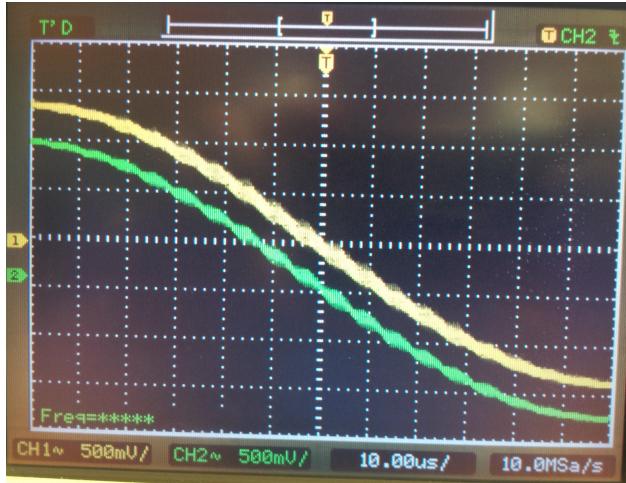


Figura 4: Sinal sinusoide com interpolação e sem interpolação.

P1-8

2.2.2 P2. Transmissor BPSK

O objectivo deste projeto é criar um transmissor BPSK com recurso a três elementos principais, uma fonte de bits, um codificador diferencial e mapeador, e um modulador. Neste projeto foi utilizada uma frequência de amostragem $f_s = 16$ kHz e uma frequência portadora $f_0 = 4$ kHz.

Para ter uma fonte de bits no transmissor usa-se um "bit-rate clock" cuja função vai ser criar uma sequência de bits b_n com $f_b = 1$ kbps. Isto significa que, considerando f_s , a cada 16 ciclos é gerado um novo bit, alternado em relação ao anterior. Assim, implementou-se um contador que é incrementado em cada ciclo e que tem uma condição para verificar quando chegar ao valor 16. Ao entrar nessa condição é implementada a lógica para cálculo do novo bit da sequência e o contador é reiniciado.

Para calcular o novo bit, basta negar o bit anteriormente obtido para obter uma sequência de bits alternada, tendo sido concretizado através de uma simples XOR:

$$b_n = b_{n-1} \oplus 1 \quad (4)$$

Com esta operação, o bit resultante será sempre alternado em relação ao anterior. Assim, obtém-se uma onda quadrada, observada no osciloscópio, que varia entre "0"e "1"e representa b_n com uma frequência de 500 Hz(ver figura com bn). Esta frequência ocorre devido a dividir-se f_b por dois pois cada meio ciclo da onda quadrada corresponde a um bit.

Após obter a fonte de bits passou-se ao segundo elemento do transmissor: o codificador diferencial e mapeador. Começando pelo codificador diferencial, este serve para evitar ambiguidades na fase de maneira a poder sempre recuperar uma sequência de bits num canal que tenha sofrido uma variação na fase. A codificação de b_n resulta também de uma operação lógica XOR, como se pode observar:

$$c_n = c_{n-1} \oplus b_n , c_0 = 0 \quad (5)$$

Como um bit novo só quando o contador chega ao valor 16, o mesmo também só é codificado nessa condição, ou seja, a cada 16 ciclos é codificado um bit.

Tal como em b_n também c_n é representado por uma onda quadrada que varia entre "0"e "1"só que com o dobro do período, devido a só ter dois resultados para os quatro casos possíveis da XOR que realiza a codificação(figura X).

IMAGEM bn com cn

Depois de obter c_n passa-se ao mapeamento do mesmo. O mapeamento baseia-se em duas condições:

$$c_n = '1' \rightarrow d_n = +1 \quad c_n = '0' \rightarrow d_n = -1 \quad (6)$$

Esta lógica podia ser facilmente implementada com recurso a duas condições "if", mas optou-se por evitar essa lógica para tornar o programa mais eficiente. Assim recorreu-se a um shift e a uma subtração. Atenção que esta não é a maneira mais eficiente pois a subtração faz com que o programa tenha de passar pela ALU. Pode-se observar então o mapeamento efetuado através da seguinte expressão:

$$d_n = 32767 * ((c_n << 1) - 1) \quad (7)$$

Antes de mais, o ganho que está a ser multiplicado é utilizado devido ao elemento modulador do transmissor, que vai ser falado a seguir. Considerando a expressão sem esse ganho, vê-se que para $c_n = 0$, como o shift não influencia o resultado, o mesmo só depende da subtração e é sempre o pretendido, $d_n = -1$. Se $c_n = 1$, o shift duplica esse valor, e depois ao subtrair obtém-se $d_n = 1$.

Tal como em b_n e c_n esta operação só é executada a cada 16 ciclos pois depende diretamente de c_n e só se mapeia um novo bit depois de ele ser codificado. Concluído o mapeamento obtém-se mais uma vez uma onda quadrada mas desta vez varia entre -1"e "1"(figura X).

IMAGEM cn com dn

P2-2 Falta agora gerar a onda portadora a modular. Esta foi implementada de forma mais simples face ao projeto anterior uma vez que a frequência é estática (4 kHz) e este valor consiste numa fração inteira da frequência de amostragem (16kHz). Em primeiro lugar criou-se uma tabela com quatro valores dum período da sinusóide, sendo esta:

Tabela 3: Amostras da onda portadora

contador	seno
0	0
1	32767
2	0
3	-32767

Escolheram-se estes valores uma vez que o período de amostragem coincide com os instantes de máximo, mínimo e zero-crossing da portadora. Para gerar a portadora (figura X) recorre-se a um contador que, em cada interrupção (ocorrendo em cada instante de amostragem, como explicado no enunciado), aponta para cada entrada da tabela e põe a amostra numa variável que, após se incrementar o contador, irá ser multiplicada por d_n . Esta tabela não gera uma onda triangular pois das harmónicas destas, abaixo da frequência de amostragem ($f_0=4\text{kHz}$ e $3f_0=12\text{kHz}$) apenas a primeira harmónica se encontra na banda de passagem do filtro de reconstrução do DAC, que terá frequência de corte $F_s/2$.

IMAGEM com portadora

A onda modulada é então representada pela seguinte expressão:

$$s_n = d_n \sin(2\pi f_0 T_s n) \quad (8)$$

Neste caso os valores do seno são os valores das amostras discutidas anteriormente. Tem-se dois sinais Q15, ou seja o bit mais significativo para o sinal e 15 bits para a parte fracionária. Ao multiplicar-se dois sinais Q15 sabe-se que o resultado será sempre Q15, contudo como o multiplicador retorna um valor em Q30, para este ser armazenado tem que se fazer shift left uma vez para eliminar o sinal repetido e shift right 16 vezes para transportar os bits mais significativos encostar os bits do resultado nos bits menos significativos para se truncar

P2-3 Após implementar o modulador, tem-se o transmissor BPSK completo e para testá-lo, pôs-se nos dois canais do osciloscópio a onda portadora e a onda modulada (figura Y).

IMAGEM portadora com modulada

Como se pode observar pela imagem e considerando a expressão (8), a onda portadora começa em oposição de fase com a onda modulada. Isto ocorre quando $d_n = -1$, o que significa que quando $d_n = 1$ a onda modulada é igual à portadora. É possível observar quando d_n muda de valor pois é exatamente quando a onda modulada inverte a sua direção (figura Z).

("a sua direção"é uma boa maneira de explicar?)

IMAGEM com dn e onda modulada

Este fenómeno acontece a cada 8 ciclos pois como já foi explicado anteriormente tem-se um bit-rate dividido por dois que acaba por influenciar d_n e que...

Espectro

Modulação=> tempo: é multiplicação da onda quadrada com a sinusoide. Na frequência: as harmónicas ímpares (espectro quadrada) convoluídas com dirac a 4kHz(espectro portadora). Na imagem o espectro da quadrada centrada em 4khz como se esperava. As harmónicas estão espaçadas 500Hz pois o sinal d_n consiste numa onda quadrada de frequência fundamental 250 Hz.

3 Conclusão

-Principais resultados e conclusões sobre eles, erros a corrigir (se houverem), o que melhorar

4 Anexos

-Código?

-possivelmente pode-se aqui algumas das imagens