Seminário 6

Uso dos Timers do mC LP2129 para a realização de um "Phase Locked Loop" (PLL).

Sumário

- Descrição do problema.
- Projeto do algoritmo de controle.
- Configuração dos timers.
- Configuração das interrupções.
- Desenvolvimento e teste do programa usando simulador.

Referências

1. Datasheet do microcontrolador Philips LPC2129.
2. Manual de uso do microcontrolador Philips LPC2129.
3. Simulador do processador ARM7

Problema

O problema que devemos resolver consiste em usar o μC LP2129 para a construção de um sinal de onda quadrada cuja frequência seja 8 vezes a frequência de um sinal de referência (sempre a onda quadrada). A frequência de referência está entre 40 Hz e 60 Hz, com valor nominal de 50 Hz.

O tempo di sincronização deve ser o menor possível. Além disso, o sinal sintetizado deve permanecer em fase com o sinal de referência com o menor erro possível para o hardware considerado (f_{clk} = 12 MHz).

PLL

Um PLL permite regular a frequência e a fase de um sinal, colocando-o em uma relação conhecida e constante com um sinal de referência. A sua realização analógica é simplificada com o uso oportuno de circuitos integrados, como o CD4046.

A função do PLL é extremamente útil para diversas aplicações como multiplicador de frequência, síntese de frequências, circuitos de sincronização.

PLL

É importante ressaltar que obter a relação requerida entre as frequências não é, em si, difícil, mas a amarração de fase requer que a frequência do sinal gerado seja controlada em malha fechada.

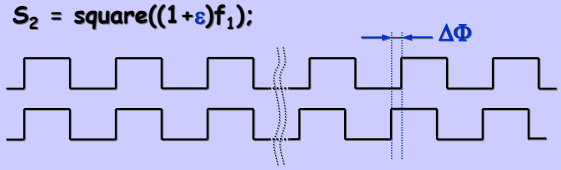
Uma realização em malha aberta, por causa das inevitáveis diferenças entre a frequência desejada e aquela efetiva, levaria a um defasamento continuamente variável entre os sinais de sincronismo e gerado. As duas formas de onda, na prática deslizam uma em relação à outra (ainda que muito lentamente).

PLL

Consideremos, por exemplo, os dois sinais:

S_1 = square(f_1);

S_2 = square($(1+\varepsilon)f_1$);



Por causa do erro ε na frequência, os dois sinais, que partem em fase, depois de um certo tempo aparecem defasados. O tempo integra a diferença entre as frequências!

PLL

A solução consiste em variar de modo controlado a frequência do sinal gerado de modo a impor o anulamento do erro de fase.

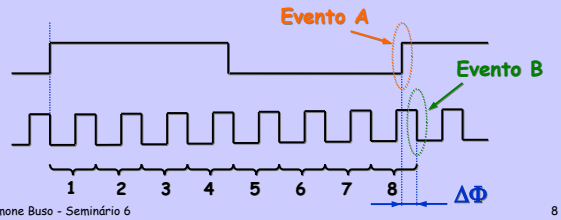
Um PLL realiza assim estas funções:

1. determinação do erro de fase;
2. regulação da frequência do sinal gerado para anular, com uma dinâmica controlada, o erro de fase.

Como todos os sistemas realimentados, também o PLL pode ser instável!

Algoritmo de controle do PLL

No nosso caso, devemos manter amarradas em fase duas ondas quadradas, uma externa (sincronismo) e uma construída por nós, com uma frequência de 8 vezes aquela original. Definamos em primeiro lugar o erro de fase:



Algoritmo de controle do PLL

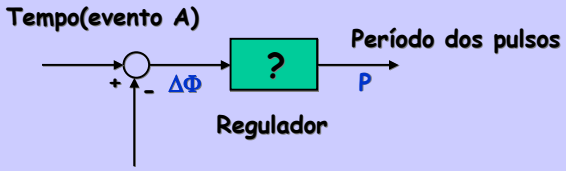
O erro de fase é o intervalo de tempo que transcorre entre os eventos A (fronte de subida do sinal de sincronismo) e B (final do oitavo impulso gerado).

Esta quantidade pode ser positiva ou negativa, porque a ordem entre os eventos não é conhecida a priori.

O objetivo do nosso sistema de controle é levar o erro de fase a zero em um certo tempo (o mais breve possível), modificando de modo oportuno o período dos impulsos gerados.

Algoritmo de controle do PLL

Configura-se assim uma estrutura de controle do tipo:



O problema é encontrar o regulador que obtenha o resultado (ΔΦ = 0) e garanta a estabilidade!

Algoritmo de controle do PLL

Primeira solução: tento compensar o atraso em um único período de controle.

$$P(k+1) = P(k) + \frac{\Delta\Phi(k)}{4}$$

O novo período é igual ao antigo, ao qual se soma uma correção. A correção é igual ao erro de fase dividido por 4. Deste modo o novo período será tal que anula o erro no passo sucessivo.

Mas está será uma situação de equilíbrio?

Algoritmo de controle do PLL

Consideremos o seguinte exemplo: suponhamos que o sinal de sincronismo tenha um período P_S igual a 8 p.u. (unidade arbitrária).

$P(k) = 0.9$
$\Delta\Phi(k) = 0.8$
$P(k+1) = 1.1$

O valor de regime deveria ser 1!

Depois de 8 períodos o erro $\Delta\Phi(k+1)$ será:

$$\Delta\Phi(k+1) = 8 + 0.8 - 1.1 \cdot 8 = 0!$$

P_S $\Delta\Phi(k)$ 8 períodos $P(k+1)$

Algoritmo de controle do PLL

O exemplo precedente mostra que se tentamos compensar de modo imediato o erro de fase, o sistema começará a oscilar porque o anulamento do erro implica atribuir ao período dos pulsos um valor diferente daquele de regime!

Devemos seguramente modificar a nossa estratégia de controle.

Idéia: separemos a compensação do erro de fase da construção do período P em regime, de modo que, com o erro compensado, o período possa ser aquele "certo".

Algoritmo de controle do PLL

Solução "em duas fases":

Aqui construo o valor de regime

$$P_{Int}(k) = P_{Int}(k-1) + \frac{\Delta\Phi(k)}{8}$$

Aqui compenso o erro de fase

$$P(k+1) = P_{Int}(k) + \frac{\Delta\Phi(k)}{8}$$

A variável $P_{Int}(k)$ memoriza as correções que são feitas a cada vez.

Algoritmo de controle do PLL

Consideremos de novo o exemplo precedente:

$P_{Int}(k-1) = 0.9$

$\Delta\Phi(k) = 0.8$

Segundo o novo algoritmo: valor de regime!

$P_{Int}(k) = 0.9 + 0.1 = 1$

$P(k+1) = 1 + 0.1 = 1.1$

Ao final do período sucessivo teremos erro de fase nulo (porque P(k+1) é 1.1 como antes), mas agora o sistema está em regime porque em $P_{Int}(k)$ está memorizado o valor correto.

Algoritmo de controle do PLL

Observemos que o nosso algoritmo final tem a estrutura de um regulador PI discretizado, no qual as constantes proporcional e integral são iguais entre si e valem 1/8.

Esta particular escolha confere, todavia, ao algoritmo características muito vantajosas: este é capaz de anular o erro de regulação em apenas um passo!

Este tipo de regulador é chamado "dead-beat". Seu comportamento deriva do posicionamento do pólo do sistema em malha fechada na origem do plano complexo.

Realização do PLL

Para implementar o algoritmo devemos estabelecer uma estratégia para a medida do erro de fase, que, como visto, é um intervalo de tempo.

Naturalmente devemos usar os timers do μC.

Um deve operar no modo "capture" (atribuindo ao seu pino de entrada o sinal de sincronismo) e outro no modo "compare" (ou match), com a tarefa de medir o período dos pulsos e de gerar em seu pino de saída.

Realização do PLL

O problema seguinte é quando avaliar o erro de fase. Existem pelo menos duas possibilidades:

1. quando ocorre o evento A;
2. quando ocorre o evento B.

A escolha correta é a segunda, porque permite modificar o período dos impulsos sempre no início de uma nova sequência de 8, como previsto pelo algoritmo.

Se agíssemos no evento A, poderíamos modificar o período de uma sequência já iniciada, coisa que o algoritmo não prevê!

Realização do PLL

Tal escolha, ainda que correta, coloca outros dois problemas de ordem prática (categoria: temporização):

1. tempo de cálculo;
2. cálculo do erro de fase quando o evento A segue B.

O tempo para realizar os cálculos é limitado. Idealmente, devemos ser capazes de atualizar o período dos impulsos antes que se complete o primeiro semiperíodo do primeiro impulso da nova sequência.

Realização do PLL

Em regime este tempo é:

$P_{Smin}/16 = 1/(60 \cdot 16) = 1.04$ ms

Tendo em conta a reduzida complexidade do algoritmo de controle, estamos em condições de garantir o funcionamento em regime.

No transitório porém é possível que, para anular o erro de fase, o controle imponha uma frequência dos impulsos ainda maior que a máxima esperada.

Os tempos do algoritmo devem ser avaliados no caso mais crítico.

Realização do PLL

Consideremos o caso em que a frequência da onda quadrada de sincronismo passe de 40 Hz a 60 Hz em degrau, imediatamente depois que o período dos impulsos foi calculado.

Suponhamos que o sistema estivesse em regime e portanto parta de um erro de fase nulo.

Ao final do novo período da onda quadrada o erro de fase será:

$1/60 - 1/40 = -8.333$ ms

O controle reduzirá o período dos impulsos de modo a compensar o atraso de fase.

Realização do PLL

O novo período dos impulsos será:

$P(k+1) = 1/320 - 8.333 \cdot 10^{-3}/4 = 1.041$ ms

que corresponde a uma frequência de aproximadamente 960 Hz.

O cálculo deve ser concluído em um tempo máximo de $P(k+1)/2 = 0.52$ ms.

Trata-se de um tempo compatível com a velocidade do nosso μC, como se pode verificar avaliando a parte do programa que corresponde ao algoritmo de controle.

Realização do PLL

```
PLL_routine:
        MOV     r2, #0                  /* reset do contador de ciclos          */
        LDR     r0, =CCR0               /* carrega endereço capture register    */
        LDR     r8, [r0]                /* lê valor CCR0                        */
        LDR     r0, =TOTC               /* carrega endereço TC0                 */
        LDR     r9, [r0]                /* lê valor TC0                         */
        SUB     r10, r8, r9             /* erro de sincronismo                  */
        LDR     r0, =PERIODO_0          /* uso o período medido                 */
        LDR     r6, [r0]                /* para evitar saturação                */
        MVN     r6, r6, ASR #1          /* - semiperíodo in r6                  */
        CMP     r10, r6                 /* r6 contém o semiperíodo (negado)     */
        SUBLE   r10, r10, r6, LSL #1    /* corrige medida erro                  */
        MOV     r10, r10, ASR #4        /* divido por 16                        */
        ADD     r4, r4, r10             /* integral das correções               */
```


Realização do PLL

As instruções que acessam a memória são 6, às quais se somam 3 instruções de deslocamento de dados entre registradores e 4 instruções aritméticas.

No caso pior, são realizadas um número de ciclos igual a:

$N = 6 \cdot 3 + 3 \cdot 2 + 4 \cdot 1 = 28$

que corresponde a um tempo estimado de execução de 2.33 μs.

Tem-se uma margem muito ampla no tempo de execução. Pode-se garantir o funcionamento até a frequências bem maiores!

Realização do PLL

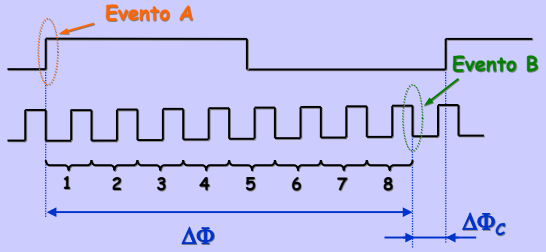
O segundo problema da realização da medida do erro de fase deve-se ao fato que nunca seremos capazes de obter valores > 0.

No caso de avanço de fase, o evento B precede o evento A. No momento dos cálculos, o tempo no evento A não pode, portanto, ser conhecido.

Na prática, o algoritmo utilizará o antigo valor do tempo do evento A.

Devemos, então, reconstruir o valor correto através de uma correção da medida.

Realização do PLL



O valor esperado para o erro de fase pode ser reconstruído somando ao erro encontrado (<0) o valor do período da onda de sincronismo.

Realização do PLL

Para garantir o correto funcionamento da correção, é necessário que:

1. esta venha aplicada apenas quando o erro de fase supera metade do período do sinal de sincronismo;
2. o período do sinal de sincronismo seja conhecido.

A segunda condição, dado que o sinal de sincronismo pode ter frequência variável, impõe medir o período, coisa que pode ser feita usando a interrupção associada à condição de "capture".

Realização do PLL

O comportamento transitório do regulador se ressentirá do fato que as variações do período de sincronismo podem ser obtidas apenas com um atraso de um período.

De fato, até que o período não seja completo não podemos medir-lhe a duração.

As eventuais correções dos erros de fase realizadas com valores de período não atualizados serão, portanto, erradas.

No pior caso, pode-se esperar um atraso de amarração de 2 sequências de 8 impulsos.

Problema

Consideremos a seguinte situação: com o PLL em regime, a frequência do sinal de sincronismo é reduzida em "degrau".

Segue-se um erro de fase negativo, porque o trem de 8 pulsos se conclui antes do evento A correspondente.

O mecanismo de correção descrito reconstrói o erro de fase, mas o algoritmo não produz qualquer correção do período dos pulsos.

A que se deve este comportamento?

Problema

No momento da atualização do período dos pulsos, o valor memorizado para o período do sinal de sincronismo ainda é aquele antigo!

Assim a reconstrução do erro produz um erro de fase final praticamente igual a 0, o que não leva a qualquer correção do período dos pulsos.

Apenas sucessivamente o período do sinal de sincronismo será atualizado, e então apenas no ciclo sucessivo se poderá ter qualquer efeito.

Problema

Felizmente, no entanto, o erro de fase resultante ao sucessivo instante de controle, será compensado como se fosse obtido todo no ciclo de controle imediatamente precedente.

Isto, no entanto, é errado, porque na realidade isto deriva de dois ciclos de controle nos quais a compensação não ocorreu.

A correção que será realizada será portanto errada. Isto gerará um transitório que pedirá ainda outros ciclos para se estinguir.

Isto reduz significativamente a velocidade de resposta.

Solução

A solução consiste em identificar os aumentos de período do sinal de sincronismo e, caso ocorra, modificar a lei de controle de modo a evitar o surgimento do transitório.

É suficiente somar ao termo integral uma quantidade igual ao incremento do período medido dividida por 8. Além disso, no momento da correção seguinte, a atualização da parte integral não deve ser feita.

Isto permite levar o sistema ao regime com um atraso máximo igual a três sequências de 8 pulsos.

Realização do PLL

Assim estruturado, o PLL garante o menor tempo de amarração possível (3 ciclos) e a faixa de captura desejada (de 40 Hz a 60 Hz).

A máxima resolução do erro de fase pode ser diminuída até o limite de um período de contagem do timer utilizado como "base dos tempos", isto é, aquele em modalidade capture. Levando este valor a T_{clk}, teremos também a máxima precisão possível com o nosso μC (83.3 ns).

Trata-se agora de configurar os timers no modo adequado.

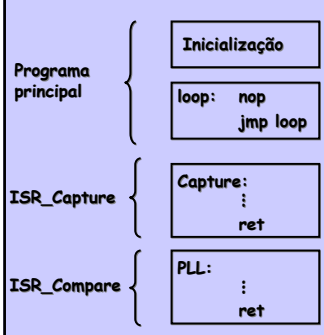
Realização do PLL

A organização mais conveniente para a aplicação requer o uso de interrupções.

O primeiro timer (TIMER0) pode servir de base de tempo, operar em modalidade capture e medir o período do sinal de sincronismo. A sua atividade pode ter menor prioridade em relação àquela do TIMER1.

Este é usado na modalidade match (compare) de modo a produzir os pulsos. A sua ISR deve ter a máxima prioridade. Cada oitava IR devemos executar o algoritmo do PLL, de modo a atualizar o período.

Diagrama de blocos do PLL



O programa, depois de uma fase di inicialização de variáveis, registros, periféricos, etc., entra em um ciclo infinito de espera.

rotina que atualiza a medida do período do sinal de sincronismo.

rotina que executa o algoritmo do PLL.

Organização do programa PLL

Realização do PLL

Neste ponto deve-se verificar para cada ISR a condição de intervalo.

Para este exemplo, o tempo disponível é muito maior que a duração de cada ISR. O problema não é crítico.

Assim, no pior caso, a ISR com máxima prioridade poderá ocasionalmente ser atrasada por aquela associada ao TIMER0, mas como esta é muito breve, o atraso é desprezível. Não há violação da condição de intervalo.

Configuração das interrupções

Escrevendo no registro VICIntEnable a sequência de bit:

```
.equ  VICInt_CFG,          0x00000030
```

Habilitam-se os dois timer a produzir uma IR quando se verificar o evento de capture e a condição de match respectivamente.

Escrevendo no registro VICIntSelect a sequência:

```
.equ  VICSel_CFG,          0x00000020
```

Qualifica-se TIMER0 como IRQ e TIMER1 como FIQ.

Configuração dos timer

Ocorre como em exemplos já tratados (geração de onda quadrada, medida de período).

Dadas as especificações é oportuno inicialmente fixar pclk=cclk atuando sobre o controlador do bus dos periféricos.

TIMER0 é configurado para identificar bordas positivas no pino C0.0.

TIMER1 é configurado para gerar uma onda quadrada no pino M1.0.

Ambos os pinos devem ser preliminarmente conectados a um dos pinos de I/O do μC.