UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

ENGENHARIA ELETRÔNICA

FELIPE DOS PASSOS CANTERI

MÉTODOS DE SINCRONISMO DE PROCESSADORES DIGITAIS DE SINAIS COM A REDE ELÉTRICA COMERCIAL, APLICADOS EM GERAÇÃO DE ENERGIA FOTOVOLTAICA

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2017

FELIPE DOS PASSOS CANTERI

MÉTODOS DE SINCRONISMO DE PROCESSADORES DIGITAIS DE SINAIS COM A REDE ELÉTRICA COMERCIAL, APLICADOS EM GERAÇÃO DE ENERGIA FOTOVOLTAICA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia Eletrônica, do Departamento Acadêmico de Eletrônica, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Marcio Mendes Casaro

PONTA GROSSA

2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa
DAELE – Departamento de Eletrônica

FOLHA DE APROVAÇÃO

MÉTODOS DE SINCRONISMO DE PROCESSADORES DIGITAIS DE SINAIS COM A REDE ELÉTRICA COMERCIAL, APLICADOS EM GERAÇÃO DE ENERGIA FOTOVOLTAICA

Desenvolvido por:

FELIPE DOS PASSOS CANTERI

Este trabalho de conclusão de curso foi apresentado em 05 de maio de 2017, como requisito parcial para obtenção do título de Bacharel em Engenharia Eletrônica. O candidato foi arguido pela banca examinadora composta pelos professores abaixo assinado. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Dr. Marcio Mendes Casaro
Professor Orientador

Dr. Eloi Agostini Junior
Membro titular

Dr. Helio Voltolini
Membro titular

- A Folha de Aprovação assinada encontra-se na Coordenação do Curso -

RESUMO

CANTERI, Felipe P. **Métodos de sincronismo de processadores digitais de sinais com a rede elétrica comercial, aplicados em geração de energia fotovoltaica**. 2017. 53 f. Trabalho de Conclusão de Curso de Bacharelado em Engenharia Eletrônica - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017.

A geração de energia fotovoltaica cresce no mundo e o Brasil tem imenso potencial. A geração fotovoltaica distribuída requer sincronismo entre o dispositivo controlador, usualmente um processador digital de sinais, e a rede elétrica comercial de distribuição. Foram estudados diversos métodos publicados na literatura, porém a grande maioria não apresenta todas as informações necessárias para sua utilização, exige muito processamento, circuitos complexos, volumosos e/ou custosos, sendo alguns patenteados. Logo, foram desenvolvidos dois métodos de sincronismo baseados em malhas de captura de fase (PLL) com foco em simplicidade e eficiência. O primeiro método apresentou faixa de atuação de 46Hz a 66Hz e tempo de estabilização mediante a distúrbio de até 4 ciclos da rede elétrica. O segundo método apresentou faixa de atuação de 8Hz a 116Hz, com estabilização em até 5 ciclos da rede. Considerando uma norma de geração distribuída estudada, a faixa de atuação de ambos os métodos foi adequada. Ressalta-se que os métodos de sincronismo podem ser considerados para outras situações nas quais se necessite de sincronismo com a rede.

Palavras-chave: Sincronismo. PLL. Energia Fotovoltaica. DSP.

ABSTRACT

CANTERI, Felipe P. **Synchronization methods of digital signal processors with the commercial electrical network, applied for distributed generation of photovoltaic energy**. 2017. 53 p. Course Conclusion Paper in Electronic Engineering, Bachelor's Degree - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2017.

The photovoltaic energy generation grows in the world and Brazil has a huge potential. The distributed photovoltaic generation requires synchronism between the controller device, usually a digital signal processor, and the commercial distribution electrical network. Several methods published in the literature were studied, though the vast majority doesn't present all the information necessary for its use, requires too much processing, complex, voluminous and/or costly circuits, with some being patented. Thus, two synchronization methods based on phase-locked loops (PLL) were developed, with focus on simplicity and efficiency. The first method presented a range of operation from 46Hz to 66Hz and stabilization time after disturbance of at most 4 cycles of the electrical network. The second method presented a range of operation from 8Hz to 116Hz, with stabilization in at most 5 cycles of the network. Considering one distributed generation regulation studied, the range of operation of both methods were adequate. It is emphasized that the synchronization methods can be considered for other situations in which synchronism with the electrical network is necessary.

Keywords: Synchronism. PLL. Photovoltaic Energy. DSP.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

CA	Corrente Alternada	
CC	Corrente Contínua	
CI	Circuito Integrado	
DSP	*Digital Signal Processor*	(Processador Digital de Sinais)
EPLL	*Enhanced Phase-Locked Loop*	
FPGA	*Field Programmable Gate Array*	(Arranjo de Portas Programáveis em Campo)
GPIO	*General Purpose Input-Output*	(Entrada-Saída de Propósito Geral)
NTC	Norma Técnica Copel	
PLL	*Phase-Locked Loop*	(Malha de Captura de Fase)
PPLL	*Predictive Phase-Locked Loop*	
PWM	*Pulse Width Modulation*	(Modulação por Largura de Pulso)
RC	(Filtro) Resistor-Capacitor	
VCO	*Voltage-Controlled Oscillator*	(Oscilador Controlado por Tensão)

LISTA DE ACRÔNIMOS

ABNT	Associação Brasileira de Normas Técnicas
ANEEL	Agência Nacional de Energia Elétrica
COPEL	Companhia Paranaense de Energia

LISTA DE SÍMBOLOS

C1	Capacitor do oscilador controlado por tensão
Cf	Capacitor do filtro passa-baixa
GW	Gigawatt – um bilhão de watts. Watt é a unidade de potência elétrica
Hz	Hertz – unidade de frequência
kHz	Kilohertz – mil hertz
kW	Kilowatt – mil watts
kWh	Kilowatt-hora – unidade de energia
kΩ	Kiloohm – mil ohms. Ohm é a unidade de resistência elétrica
m²	Metro-quadrado – unidade de área
ms	Milisegundo – um milésimo de segundo

mV	Milivolt – um milésimo de Volt
MW	Megawatt – um milhão de watts
nF	Nanofarad – um bilionésimo de farad. Farad é a unidade de capacitância elétrica
R1	Resistor do oscilador controlado por tensão
Rf	Resistor do filtro passa-baixa
Rin	Resistor de entrada
Rout	Resistor de saída
s	Segundo – unidade de tempo
V	Volt – unidade de tensão elétrica
µs	Microsegundo – um milionésimo de segundo

SUMÁRIO

1 INTRODUÇÃO

A geração de energia elétrica através da energia solar tem sido cada vez mais difundida e utilizada pela sociedade moderna. Fatores ambientais, econômicos e políticos têm alavancado as pesquisas nessa área e a tendência é que esse mercado continue crescendo. Uma das principais maneiras de aproveitar essa forma de energia é através de painéis fotovoltaicos, que transformam a radiação solar em energia elétrica diretamente.

Segundo IEA (2014), dos anos de 2010 a 2014, foram instalados sistemas fotovoltaicos com capacidade de geração superior à capacidade instalada nas últimas quatro décadas somadas. No começo de 2014, a capacidade de geração global passou dos 150GW e a previsão para 2050 é que sistemas fotovoltaicos gerem 16% da energia elétrica mundial.

O Brasil tem imenso potencial para geração de energia fotovoltaica, em especial a geração distribuída, a qual é a geração de pequeno porte no próprio consumidor ou próxima a este, conectada à rede de distribuição da geração centralizada de grande porte.

Conforme cálculos realizados por EPE (2014), considerando a área útil de telhados domiciliares para instalação de painéis fotovoltaicos, os estados brasileiros seriam capazes de gerar energia elétrica equivalente a 140% a 400% do seu consumo elétrico residencial se painéis fotovoltaicos fossem instalados em todos os domicílios. Certamente é uma hipótese não factível, mas que demonstra o potencial técnico de geração fotovoltaica distribuída no Brasil.

O sistema de distribuição elétrico brasileiro fornece tensão senoidal alternada em 60Hz, enquanto painéis fotovoltaicos produzem tensão contínua. Logo, são necessários conversores de frequência para a conversão CC-CA (corrente contínua para corrente alternada) e adequação da frequência e nível de tensão para que o sistema fotovoltaico seja conectado à rede de distribuição e a energia gerada seja utilizada concomitantemente com a energia da rede.

O conversor de frequência requer informação precisa da frequência e fase da rede elétrica, portanto métodos de sincronismo são necessários. Visto que, em geral, processadores digitais de sinais são utilizados para controlar os conversores, neste trabalho houve o desenvolvimento de métodos de sincronismo de

processadores digitais de sinais com a rede elétrica comercial, a serem aplicados em geração de energia fotovoltaica.

A seção 2 deste trabalho aborda com mais detalhes a geração de energia fotovoltaica distribuída no Brasil. A seção 3 apresenta alguns métodos de sincronismo existentes na literatura. A seção 4 explica os dois métodos desenvolvidos e na seção 5 são analisados seus resultados. A seção 6 apresenta as considerações finais.

1.1 TEMA

Desenvolvimento de dois métodos de sincronismo de processadores digitais de sinais com a rede elétrica de distribuição comercial.

1.2 DELIMITAÇÃO DO TEMA

Os dois métodos desenvolvidos utilizam o princípio de malhas de captura de fase (PLL). O primeiro método utiliza um circuito integrado PLL, enquanto o segundo faz a implementação virtual deste CI. O processador digital de sinais utilizado é TMS320F28069 da fabricante Texas Instruments.

1.3 PROBLEMA

É possível desenvolver métodos simples e eficazes de sincronismo que atendam normas para uso em geração fotovoltaica distribuída?

1.4 HIPÓTESE

Visto que existem diversos métodos de sincronismo com características, funcionalidades e complexidades diferentes, o desenvolvimento de um método simples e eficaz é possível.

1.5 OBJETIVOS

1.5.1 Objetivo Geral

Desenvolver dois métodos de sincronismo de processadores digitais de sinais com a rede elétrica comercial.

1.5.2 Objetivos Específicos

- Entender o funcionamento de malhas de captura de fase;
- Estudar métodos de sincronismo existentes;
- Desenvolver e testar dois métodos de sincronismo;
- Comparar os dois métodos e verificar sua aplicabilidade.

1.6 JUSTIFICATIVA

Apesar de existirem diversos métodos de sincronismo na literatura, é raro encontrar todas as informações necessárias para sua aplicação. Além disso, a grande maioria dos métodos exige elevado esforço de processamento no DSP ou circuitos externos complexos, volumosos e/ou custosos. Entretanto, métodos simples e funcionais já foram apresentados como por VENDRICHOSKI, DURSKI e CASARO (2010), demonstrando ser plausível o desenvolvimento dos dois métodos pretendidos.

2 GERAÇÃO DISTRIBUÍDA DE ENERGIA FOTOVOLTAICA NO BRASIL

A tabela 1 apresenta a composição da matriz de energia elétrica no Brasil, demonstrando a capacidade instalada referente às diferentes fontes de energia ou no caso de importação, a capacidade contratada.

Tabela 1 – Composição da matriz de energia elétrica no Brasil

Fonte	Capacidade Instalada	
	kW	% do total
Biomassa	14.195.172	8,8842
Eólica	10.393.742	6,5051
Fóssil	26.924.319	16,8509
Hídrica	98.080.466	61,385
Nuclear	1.990.000	1,2454
Solar	23.761	0,0148
Importação	8.170.000	5,1133
TOTAL	159.777.460	100

Fonte: ANEEL (2017) - Adaptado

É possível notar que aproximadamente 61% da capacidade instalada da energia elétrica brasileira provêm de origem hídrica. Apesar do impacto ambiental na construção de represas, a geração hidrelétrica utiliza uma fonte renovável e é considerada pouco poluente.

Entretanto, a geração de energia no território nacional é fortemente dependente do ciclo de chuvas e dos níveis dos reservatórios, além do fato da água desses reservatórios ser utilizada não só para geração de energia, como também para consumo humano.

Em situações de crise hídrica, além da falta de abastecimento de água nos grandes centros, acontece também a crise de produção energética. Nesses casos, conforme explica GOMES (2014), o governo coloca em funcionamento as termoelétricas instaladas no país, as quais geram energia a custos bem mais elevados e em sua maioria utilizam combustíveis fósseis, os quais são poluentes e não renováveis. Portanto, o Brasil precisa diversificar sua matriz energética.

A contribuição da energia fotovoltaica para a matriz energética brasileira é ainda quase insignificante, conforme visto na tabela 1. Contudo, há um enorme

potencial de crescimento nessa área. Segundo EPE (2012), a irradiação solar média anual varia entre 1200 e 2400kWh/m²/ano no Brasil, valores consideravelmente superiores a de países como a Alemanha (900 a 1250kWh/m²/ano) e França (900 a 1650kWh/m²/ano) que são grandes usuários de energia solar. Logo, levando-se em consideração estes dados e os dados apresentados na seção de introdução do presente trabalho, o potencial é imenso do ponto de vista técnico.

O crescimento da geração de energia fotovoltaica no Brasil se dá principalmente através de geração distribuída, definida por COPEL (2016) como “centrais geradoras de energia elétrica, de qualquer potência, com instalações conectadas diretamente no sistema elétrico de distribuição ou através de instalações de consumidores, podendo operar em paralelo ou de forma isolada”.

A Companhia Paranaense de Energia (COPEL, 2016) estabelece ainda quatro categorias para geração distribuída:

1. Acesso de geração distribuída ao sistema da Copel com comercialização de energia;
2. Acesso de micro e minigeração distribuída ao sistema da Copel com compensação de energia;
3. Geração Própria - Operação em Paralelismo Momentâneo;
4. Geração Própria - Operação Isolada em Emergência.

Cada categoria obedece a uma norma técnica específica. Outras empresas de distribuição de energia possuem suas próprias normas, entretanto todas são obrigadas a seguir as prescrições vigentes nas resoluções normativas da Agência Nacional de Energia Elétrica (ANEEL) e nas normas técnicas da Associação Brasileira de Normas Técnicas (ABNT).

As categorias 3 e 4 referem-se a unidades consumidoras com geração própria destinada a operar em casos emergenciais ou a critério do consumidor, como, por exemplo, geradores a diesel para casos de falta de energia, podendo operar momentaneamente em paralelismo com a rede elétrica no caso da categoria 3. Já a categoria 1 refere-se a unidades geradoras que conectam-se à rede de distribuição de média e alta tensão, comercializando a energia fornecida.

A categoria 2 atende à Resolução Normativa ANEEL nº 482/2012 que estabelece que os consumidores podem instalar geradores de pequeno porte nas suas unidades consumidoras e injetar o excedente de energia gerado no sistema elétrico, o que gera créditos de energia válidos por 60 meses que podem ser

utilizados para abater do consumo da própria unidade ou outras unidades pertencentes ao mesmo proprietário. A potência instalada poderá ser de até 3MW para fontes hídricas e 5MW para demais fontes renováveis e a central geradora será classificada como microgeradora caso a potência instalada seja menor ou igual a 75kW e minigeradora para potências maiores.

Além de mais adequada ao escopo deste trabalho, a categoria 2 é ideal para a geração fotovoltaica distribuída de pequeno porte, especialmente em residências, visto que, durante o dia, em média o consumo nas residências é menor do que durante o período noturno. Logo, durante o dia, energia excedente é injetada na rede, produzindo créditos para abater o consumo que ocorrerá durante a noite (quando não há geração de energia fotovoltaica), eliminando custos com baterias.

A NTC (norma técnica Copel) 905200 estabelece os requisitos para o acesso de micro e minigeração distribuída ao sistema da Copel com compensação de energia. Um ponto a destacar desta norma é a necessidade de relés de sobre e subfrequência para proteção no ponto de conexão, proteção que deve ser também implantada nos conversores de frequência.

A tabela 2 mostra o ajuste dos tempos de atuação, ou seja, o tempo para o relé ou proteção do conversor desconectar o sistema de geração da rede elétrica dependendo da frequência medida da rede. Esses valores são importantes para validar os métodos de sincronismo desenvolvidos neste trabalho, visto que o processador digital de sinais deverá ser capaz de manter o sincronismo com a rede elétrica até esses limites para poder comandar a proteção do conversor de frequência.

Tabela 2 – Ajuste da atuação dos sistemas de proteção

Função	Estágio	Critério
Subfrequência	1º	58,5Hz a 10s
Subfrequência	2º	56,5Hz instantâneo
Sobrefrequência	1º	62Hz a 30s
Sobrefrequência	2º	66Hz instantâneo

Fonte: COPEL (2014) - Adaptado

3 MÉTODOS DE SINCRONISMO PUBLICADOS QUE UTILIZAM O PRINCÍPIO DE MALHAS DE CAPTURA DE FASE

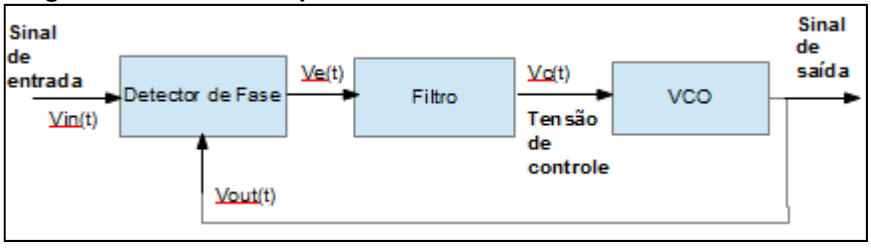
Conforme explicado nas seções anteriores, para a geração distribuída de energia fotovoltaica é necessário sincronizar o controlador do conversor de frequência, usualmente um processador digital de sinais (DSP), com a rede elétrica conectada ao conversor.

O método de sincronismo mais simples é a detecção da passagem por zero da rede, ou seja, determinação da frequência e fase da rede quando sua tensão passa de positiva para negativa e vice-versa. Segundo CHOI, KIM e KIM (2006), o defeito deste método é sua sensibilidade ao ruído. Caso haja ruído na tensão da rede, ocorrerá um erro na detecção da fase, porque o ponto de passagem por zero será cruzado várias vezes.

Malhas de captura de fase ou PLL, do inglês *phase-locked loop,* são uma solução para o problema de ruídos, sendo utilizados em diversos métodos de sincronismo. Conforme explica GARDNER (1966), duas importantes características dos PLLs são que sua largura de banda pode ser bastante pequena e que ele automaticamente segue a frequência do sinal de entrada. A estreita largura de banda garante alta rejeição de ruídos. A figura 1 apresenta um PLL básico.

O detector de fase compara o sinal de entrada com o sinal de vindo do VCO (oscilador controlado por tensão ou *voltage-controlled oscillator*, em inglês). Uma tensão dependente da diferença de fase entre os dois sinais é gerada, filtrada pelo filtro da malha e aplicada à entrada do VCO. A frequência de saída do VCO é, então, alterada por esta tensão de maneira a reduzir a defasagem entre os sinais.

Figura 1 - Malha de captura de fase básica



Fonte: Autoria própria

Assim que a malha estiver travada, ou seja, tiver adquirido o sincronismo de frequência, a tensão de controle no oscilador será a tensão necessária para manter

a frequência do VCO exatamente igual à frequência média do sinal de entrada. Entretanto, geralmente, porém não para todo PLL, é necessária uma tensão diferente de zero na entrada do filtro para manter o VCO em funcionamento, logo há uma defasagem entre o sinal de entrada e saída, a qual pode ser corrigida com o uso de um compensador ou ação integral.

GUO, WU e GU (2011) resumiram as características de diversos métodos de sincronismo que utilizam malhas de captura de fase. A tabela 3 lista esses métodos, permitindo uma comparação qualitativa entre eles. Na tabela, insensibilidade a distorções refere-se à imunidade a ruídos e distorções harmônicas. Insensibilidade a desbalanceamento refere-se à operação em sistemas trifásicos desbalanceados e métodos listados como "teórica" neste quesito foram primariamente projetados para operação monofásica, não testados em sistemas trifásicos desbalanceados.

Tabela 3 – Métodos de sincronismo que utilizam malhas de captura de fase

	Simplicidade no *design*	**Amplitude da faixa de frequências em que atua**	**Insensi-bilidade a distorções**	**Insensi-bilidade a desba-lanceamento**	**Aplicação monofásica**
Synchronous Frame PLL	Boa	Mediana	Mediana	Pobre	--
PQ-PLL	Mediana	Mediana	Mediana	Pobre	--
Double Synchronous Frame PLL	Mediana	Mediana	Boa	Boa	--
Sinusoidal Signal Integrator PLL	Mediana	Mediana	Boa	Boa	Boa
Double Second Order Generalized Integrator PLL	Mediana	Mediana	Boa	Boa	Boa
Enhanced PLL	Mediana	Mediana	Boa	Boa	Boa
Three-Phase Magnitude PLL	Mediana	Mediana	Boa	Boa	--
Quadrature PLL	Mediana	Mediana	Boa	Teórica	Boa
Robust Single-Phase PLL	Mediana	Mediana	Boa	Teórica	Boa
Predictive PLL	Mediana	Boa	Boa	Boa	Boa
Adaptive Linear Combiner PLL	Mediana	Mediana	Boa	Teórica	Boa
Multirate PLL	Mediana	Mediana	Boa	Teórica	Boa
Adaptive PLL	Mediana	Mediana	Boa	Boa	Boa

Fonte: GUO, WU e GU (2011) - Adaptado

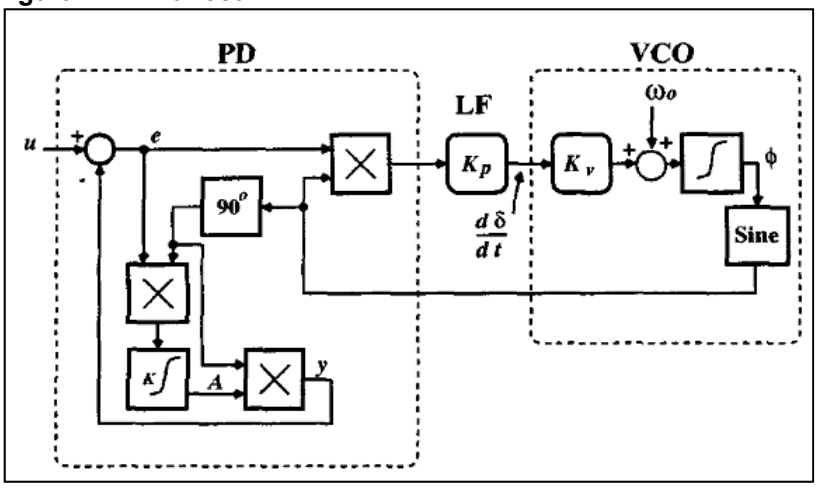
Foge do escopo deste trabalho explicar cada um destes métodos, entretanto para que haja uma base para comparação dos métodos desenvolvidos, foram escolhidos dois métodos, os quais serão explicados nas seções seguintes.

Enhanced PLL e *Predictive* PLL foram escolhidos, pois ambos são métodos projetados para aplicações monofásicas, assim como os métodos desenvolvidos, apesar de poderem ser adaptados para aplicações trifásicas. Outro motivo foi que *Enhanced* PLL aproxima-se bastante de um malha de captura de fase básica e que *Predictive* PLL apresenta-se como um dos métodos com melhores características de acordo com a tabela 3.

3.1 *ENHANCED PLL*

A figura 2 apresenta a malha de um *Enhanced* PLL que pode ser implementada tanto digitalmente quanto com componentes analógicos. u(t) é o sinal de entrada, e(t) o sinal de erro, y(t) o sinal de saída senoidal, Ø(t) o ângulo de fase e A(t) a amplitude instantânea.

Figura 2 - *Enhanced* PLL



Fonte: KARIMI-GHARTEMANI e IRAVANI (2001)

Conforme explicam KARIMI-GHARTEMANI e IRAVANI (2001), a principal diferença entre o EPLL (*enhanced* PLL) e o PLL convencional ocorre no detector de fase, pois onde haveria apenas uma multiplicação, há três multiplicações, uma subtração, uma integração e uma defasagem de 90 graus. Essas operações adicionais permitem que a saída do EPLL tenha exatamente a mesma amplitude,

frequência e fase da fundamental do sinal de entrada, enquanto malhas de captura de fase convencionais mantêm uma defasagem e apenas sincronizam em frequência.

O EPLL é altamente imune a ruídos. Visto que os valores de amplitude e fase do sinal de entrada são deixados disponíveis, o *enhanced* PLL pode ter mais aplicações do que o PLL convencional.

Segundo KARIMI-GHARTEMANI e IRAVANI (2001), pequenas variações nos parâmetros do *enhanced* PLL, tais como K, Kp, ω_0 (frequência central do oscilador) são tolerados sem efeito notável no funcionamento da malha, demonstrando sua robustez. Aumentar consideravelmente as constantes K, Kp e Kv, aumenta a velocidade de resposta da malha, ao custo de precisão.

Simulações para diversas aplicações do EPLL como detecção de pico, detecção de harmônicas, detecção de corrente reativa, extração de distúrbio, redução de ruído na detecção da passagem por zero e demodulação de amplitude e fase foram realizadas por KARIMI-GHARTEMANI e IRAVANI (2002).

Distúrbios como variação de 10% na amplitude da tensão, variação de 1Hz na frequência fundamental do sinal de entrada e ruídos como modulações de baixa frequência sobre o sinal com até 5% da amplitude do mesmo foram introduzidos nessas simulações, sendo valores bastante exagerados em relação à valores práticos segundo os autores.

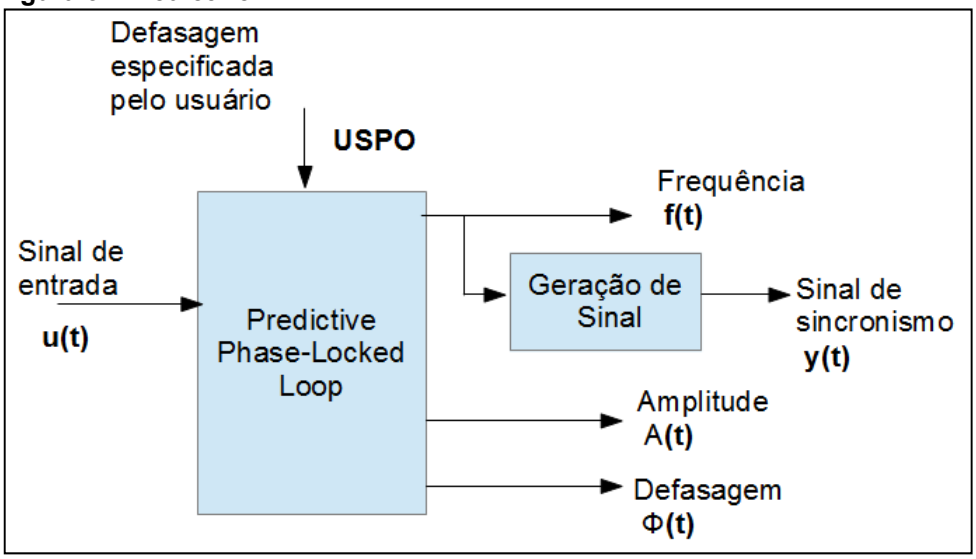
Para todas as aplicações, em no máximo três ciclos do sinal de entrada a malha conseguiu estabilizar, adquirindo sincronismo e permitindo leitura precisa das variáveis desejadas, mesmo após a inserção dos distúrbios. Foi demonstrado funcionamento tanto para 50Hz quanto 60Hz. TIMORABADI e DAWSON (2006) consideraram a faixa de operação do EPLL como sendo de 20Hz a 120Hz. Logo, *Enhanced* PLL pode ser considerado um método eficiente e robusto para sincronismo de processadores digitais de sinais com a rede elétrica.

3.2 *PREDICTIVE PLL*

A figura 3 apresenta o diagrama de um *Predictive PLL.* u(t) é o sinal de entrada, y(t) é o sinal de saída, f(t) e A(t) são a frequência e amplitude da

componente fundamental do sinal de entrada e Ø(t), a defasagem entre a fundamental do sinal de entrada e um sinal de rastreamento gerado pelo PPLL (*Predictive* PLL).

Figura 3 - *Predictive* PLL



Fonte: Autoria própria

O PPLL é composto por um preditor e um oscilador/deslocador de fase. O *Predictive* PLL gera um sinal senoidal internamente chamado de sinal de rastreamento, o qual é uma cópia exata da componente fundamental do sinal de entrada quando o sistema está sincronizado.

O preditor utiliza o sinal de rastreamento para obter três amostras do sinal de entrada, sendo a primeira no pico positivo do sinal de rastreamento a as demais, 120 e 240 graus depois. O valor de tensão dessas amostras é utilizado para calcular a amplitude e frequência da componente fundamental do sinal de entrada, além da defasagem entre a componente fundamental e o sinal de rastreamento. Essas três variáveis são apresentadas nas saídas A(t), f(t) e Ø(t) respectivamente.

O oscilador/deslocador de fase amostra o sinal de entrada várias vezes em intervalos regulares de 0 a 240° do sinal de rastreamento. Após 240°, nenhuma amostragem é realizada, então o valor da defasagem entre a componente fundamental do sinal de entrada e o sinal de rastreamento é utilizado para ajustar o período do segundo.

Quando ocorre o sincronismo, não há defasagem entre os sinais, exceto se a entrada USPO for utilizada. Esta variável força uma defasagem entre a

componente fundamental do sinal de entrada e o sinal de rastreamento, útil para quando o sinal de entrada é defasado em relação à rede elétrica do qual foi amostrado, devido a circuitos de condicionamento de sinal como transformadores.

O sinal de saída ou de sincronismo y(t) é uma onda quadrada de três estágios. Quando ocorre o pico positivo do sinal de rastreamento, a saída vai para um nível de tensão positivo e permanece neste nível até 120° do sinal de rastreamento. De 120° a 240° do sinal de rastreamento, a saída tem nível de tensão negativo e de 240° até 0, ela possui nível 0.

Em relação ao PLL convencional, o PPLL apresenta a vantagem de possuir os valores de frequência, amplitude e fase disponíveis para uso nas suas saídas. Além disso, TIMORABADI e DAWSON (2006) realizaram testes com um PPLL programado em um FPGA (*field programmable gate array* ou arranjo de portas programáveis em campo) e demonstraram que ele é capaz de operar em uma faixa de frequência de aproximadamente 0 a 2kHz, apresenta alta resistência a ruídos e distorções harmônicas e continua funcional com variações de frequência de até 120Hz por segundo. Somando-se a isto sua estabilização em até dois ciclos do sinal entrada, o método demonstra-se eficiente e robusto para sincronismo de processadores digitais de sinais com a rede elétrica.

4 MÉTODOS DE SINCRONISMO DESENVOLVIDOS

Conforme citado anteriormente, foram desenvolvidos dois métodos de sincronismo de processadores digitais de sinais com uma fase da rede elétrica comercial. Estes estão explicados em seções posteriores. Ambos os métodos baseiam-se no funcionamento de circuitos integrados PLL, portanto a seção seguinte explica o funcionamento destes.

4.1 CIRCUITOS INTEGRADOS DE MALHA DE CAPTURA DE FASE

Existem diversos circuitos integrados que realizam a função de uma malha de captura de fase ou PLL (*phase-locked loop)*, entre eles o 4046B, fornecido por diversos fabricantes diferentes (Texas Instruments, NXP Semiconductors, STMicroeletronics, etc).

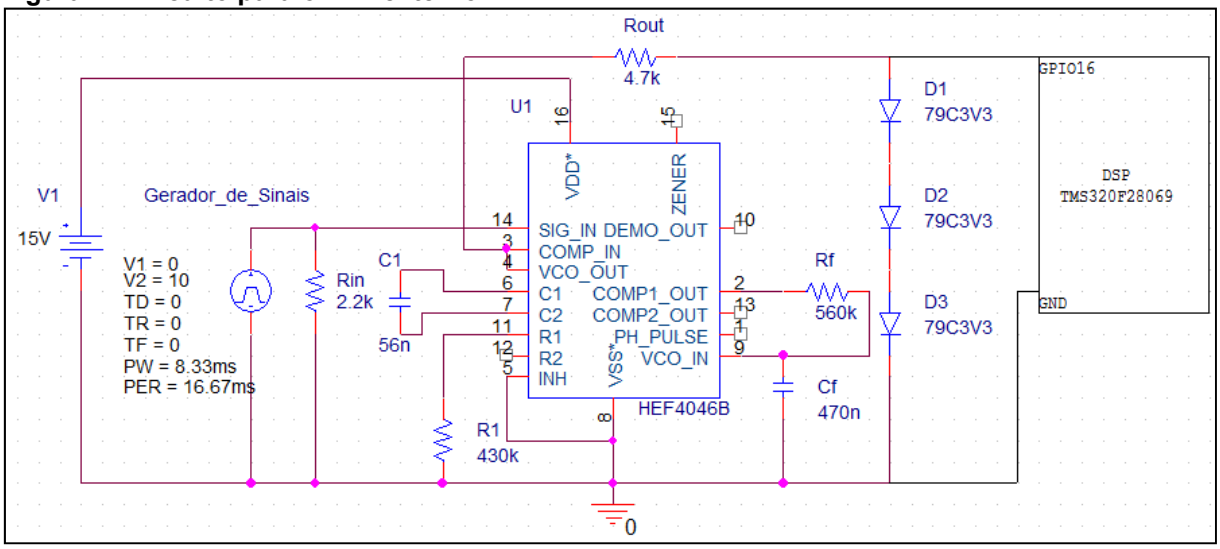
Quando conectados aos componentes adequados, esses circuitos integrados são capazes de manter a sua saída sincronizada em frequência com a frequência da componente fundamental da rede na sua entrada.

Dependendo da configuração utilizada, a saída pode ser praticamente livre de quaisquer ruídos da entrada, entretanto ser defasada em relação à entrada, ou não apresentar defasagem, porém apresentar baixa imunidade a ruídos da entrada.

Essa defasagem tem um valor fixo de tempo, cujo valor é ¼ do período da frequência central do PLL, equivalente a π/2 radianos quando a frequência da entrada e a frequência central do circuito são a mesma. A frequência central é definida por componentes conectados externamente e é a frequência que aparece na saída do PLL quando não há sinal na entrada.

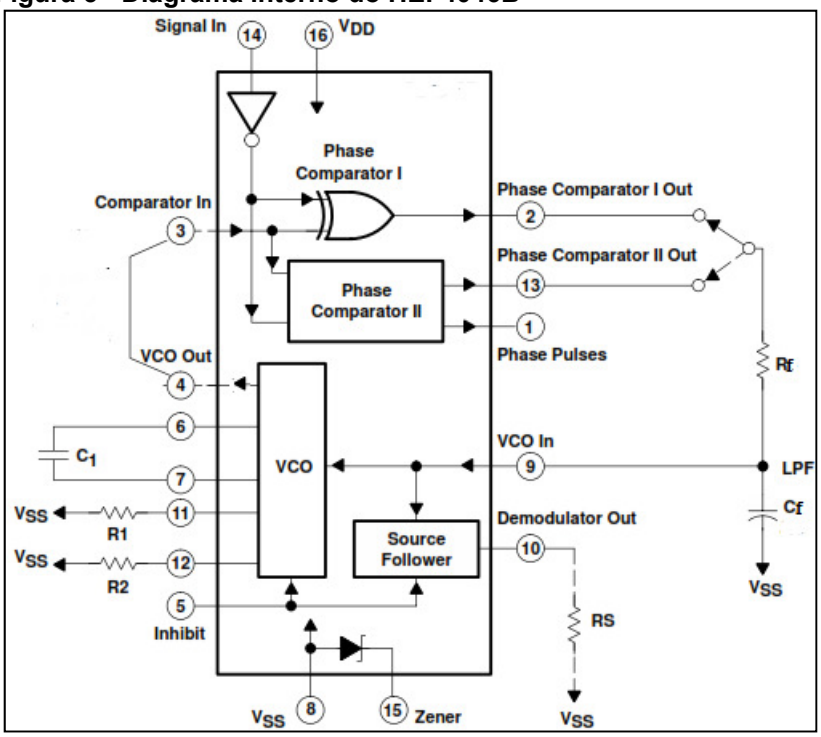
Um circuito integrado PLL é basicamente dividido em três partes: comparador de fase, filtro passa-baixa e oscilador controlado por tensão. Para o método de sincronismo desenvolvido utilizando um PLL externo, foi usado o circuito integrado digital HEF4046B, com tecnologia CMOS. O circuito da figura 4 foi montado em laboratório e a figura 5 apresenta o diagrama interno do circuito integrado.

Figura 4 - Circuito para o PLL externo



Fonte: Autoria própria

Figura 5 - Diagrama interno do HEF4046B



Fonte: MORGAN (2003) - Adaptado

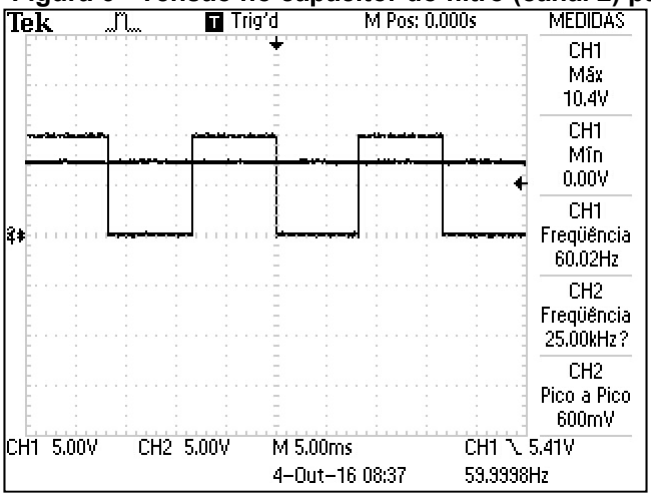
Existem dois comparadores de fase no circuito integrado utilizado que compartilham as mesmas entradas (pinos 14 e 3). O comparador 1 gera uma defasagem entre o sinal de entrada no pino 14 e o sinal de saída (pino 4) do circuito integrado, porém foi escolhido devido a sua alta rejeição de ruídos, característica que o comparador 2 não possui. O comparador 1 é composto por uma porta Ou-

Exclusivo precedida por alguns amplificadores, os quais não alteram a forma dos sinais de entrada, apenas sua amplitude.

A primeira entrada (pino 14) recebe o sinal vindo da rede elétrica, enquanto a segunda recebe a saída do oscilador controlado por tensão. Foi utilizado um gerador de sinais ao invés da rede elétrica devido à sua praticidade para testar diferentes frequências, entretanto para aplicações práticas qualquer circuito que converta a rede para uma onda quadrada com 10,5 a 15V no semiciclo positivo e 0 a 4,5V no semiciclo negativo será adequado.

A saída do comparador é uma onda quadrada que, quando o PLL está sincronizado, possui o dobro da frequência da rede. Essa saída alimenta o filtro passa-baixa (componentes Rf e Cf da figura 4). A saída do filtro é uma tensão positiva de baixa oscilação, como pode ser visto na figura 6, aplicada à entrada do oscilador controlado por tensão (pino 9).

Figura 6 - Tensão no capacitor do filtro (canal 2) para entrada de 60Hz (canal 1)



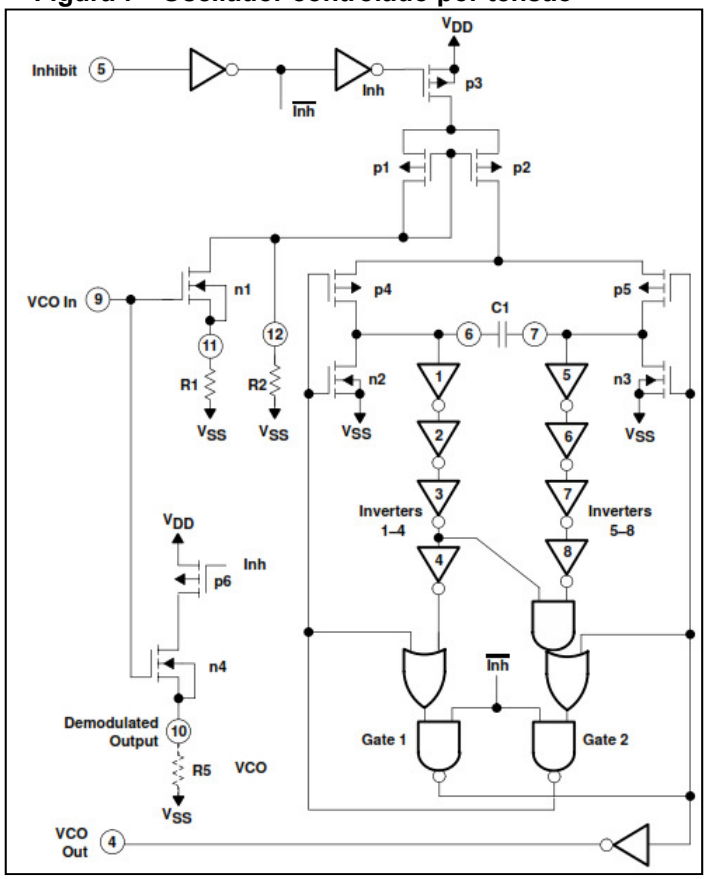
Fonte: Autoria própria

O circuito do oscilador é apresentado na figura 7. Como o resistor R2 não foi utilizado, a corrente que circula pelo capacitor C1 possui a mesma amplitude da corrente em R1, a qual é linearmente dependente da tensão de entrada VCO In no pino 9, a qual é a tensão do capacitor do filtro passa-baixa. Essa corrente circula no capacitor através dos transistores p4 e n3 ou p5 e n2.

De maneira simplificada, supondo que a corrente esteja circulando entre p4 e n3, o pino 7 estará efetivamente conectado ao VSS (negativo da fonte de alimentação). C1 será carregado pela corrente que circula por ele e assim que

atingir cerca de 7,8V, o braço de inversores 1 a 4 irá conduzir, o que causará o bloqueio de p4 e n3, fazendo o capacitor se descarregar rapidamente através de n2 e começar a ser carregado através de p5 e n2. Assim que o capacitor novamente atingir cerca de 7,8V, os inversores 5 a 8 irão conduzir, bloqueando p5 e n2 e habilitando p4 e n3, recomeçando o ciclo. Isso gerará uma onda quadrada com razão cíclica de 50% na saída VCO Out (pino 4), cuja frequência será a frequência da rede quando o PLL atingir o sincronismo ou a frequência central na ausência de sinal da rede.

Figura 7 - Oscilador controlado por tensão



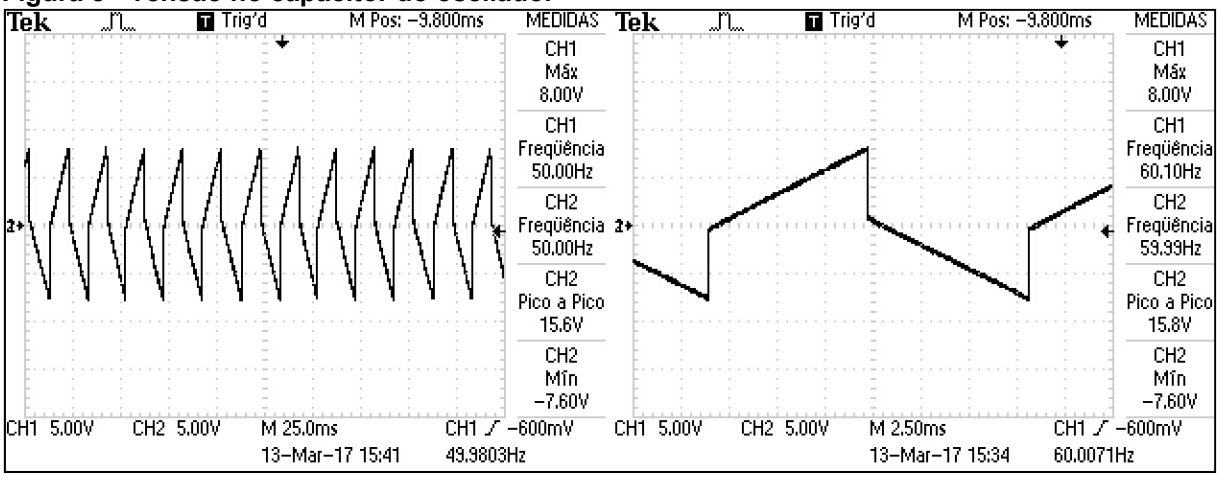
Fonte: MORGAN (2003)

A figura 8 mostra a tensão no capacitor C1 para frequências da rede de 50Hz (esquerda) e 60Hz (direita), em diferentes bases de tempo, permitindo observar o comportamento descrito. O resistor RS pode ser conectado do pino 10 para impedir a saturação do filtro passa-baixa, porém este não foi necessário. Portanto, o pino foi deixado em aberto. O pino 5 é utilizado para desabilitar o

oscilador quando seu nível lógico for 1, sendo este mantido em 0 no circuito montado.

A saída do oscilador é conectada à segunda entrada do comparador (pino 3), fechando a malha e permitindo que o PLL trave na frequência da componente fundamental do sinal de entrada vindo da rede.

Figura 8 - Tensão no capacitor do oscilador



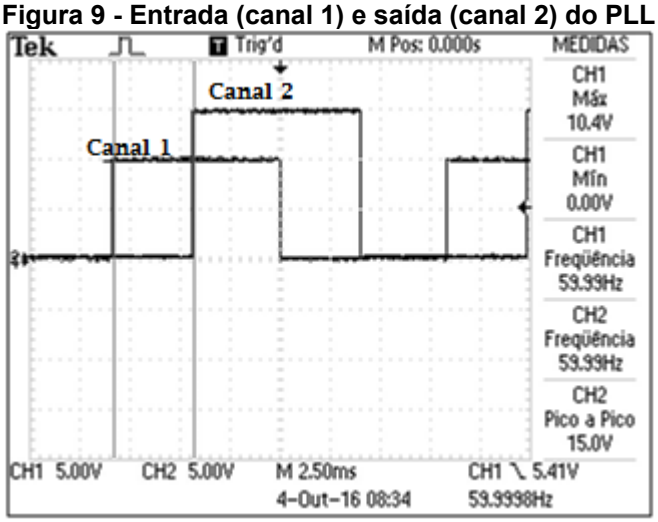
Fonte: Autoria própria

Quando o comparador 1 é utilizado, o PLL possui dois intervalos de frequência, chamados intervalo de travamento e intervalo de captura. O intervalo de captura é o intervalo de frequências de entrada para as quais o PLL consegue adquirir sincronismo, enquanto o intervalo de travamento é o intervalo de frequências de entrada para as quais o PLL consegue manter o sincronismo caso já o tenha adquirido. O intervalo de travamento é sempre maior ou igual ao intervalo de captura.

O circuito apresentado na figura 4 é capaz de adquirir sincronismo para entradas de 50 e 70Hz, pois são valores dentro do intervalo de captura, e é capaz de manter esse sincronismo caso a entrada varie para 40 ou 80Hz, pois são valores dentro do intervalo de travamento, porém é incapaz de adquirir sincronismo diretamente em 40 ou 80Hz.

A figura 9 mostra as formas de onda na entrada e saída do PLL para uma entrada de 60Hz. É possível notar uma defasagem de 4ms. Conforme já explicado, essa defasagem, dependente da frequência central do PLL, é mantida em 4ms não importando a frequência de entrada, desde que o PLL esteja sincronizado.

Figura 9 - Entrada (canal 1) e saída (canal 2) do PLL



Fonte: Autoria própria

4.2 MÉTODO DE SINCRONISMO UTILIZANDO UM PLL EXTERNO

O primeiro método desenvolvido de sincronismo do DSP com a rede elétrica utilizou o circuito da figura 4, ou seja, é um método que utiliza um circuito integrado PLL entre o sinal da rede e o processador digital de sinais. A principal vantagem do uso do PLL externo é que o sinal entregue ao DSP tem alta imunidade aos ruídos da rede elétrica, contudo há a defasagem causada pelo PLL que deve ser corrigida pelo DSP.

Os componentes Rf e Cf do filtro passa-baixa e R1 e C1 do oscilador controlado por tensão foram determinados através de equações e curvas encontrados na folha de dados do circuito integrado, baseando-se em uma frequência central próxima de 60Hz. O valor de Rout foi determinado de maneira a manter a corrente na entrada do DSP em níveis adequados e Rin foi arbitrado por ser um valor comum para resistores de *pull-down*. Como a montagem foi feita em *protoboard* e não havia preocupação com espaço, foram usados três diodos para o grampeamento da tensão na entrada do DSP, entretanto um diodo zener pode substituí-los para uma aplicação em placa de circuito impresso.

O programa completo implementado no DSP TMS320F28069 da fabricante Texas Instruments, programado em linguagem C, pode ser encontrado no apêndice A. Simplificadamente, o funcionamento do programa baseia-se no recebimento do sinal vindo do PLL pelo DSP em um pino de GPIO (*general purporse input-output)*,

ou seja, em um pino de entrada/saída digital, e sincronizar uma saída PWM (*pulse width modulation)* em frequência e fase com a rede. O pino de entrada escolhido foi o GPIO16 e a saída PWM escolhida foi EPWM1A que utiliza o pino denominado GPIO00.

De maneira geral, a função de módulos PWM, ou modulação por largura de pulso em português, é gerar formas de onda retangulares com razão cíclica variável e frequência e amplitude constantes. Para esta aplicação não é necessário alterar a razão cíclica, apenas a frequência, pois a rede elétrica comercial possui simetria entre os tempos de duração dos semiciclos positivo e negativo. Porém o que motivou a escolha de uma saída PWM como saída sincronizada com a rede elétrica foi a vantagem de o sincronismo poder ser verificado facilmente através de um osciloscópio e por não ser necessário um temporizador, visto que o DSP utilizado possui apenas 3 temporizadores gerais, porém possui 16 canais de PWM divididos em 8 módulos com um temporizador por módulo.

O sinal vindo do oscilador do PLL que entra no pino GPIO16 do DSP gera uma interrupção por borda de descida, ou seja, toda vez que o sinal no pino passa de nível lógico 1 para 0, o código em execução no DSP é interrompido e a função que trata a interrupção é executada, retornando ao término desta para onde o programa estava. Utilizar uma interrupção é essencial, pois garante que o sincronismo seja realizado toda vez que a saída do PLL completar um período, com atraso mínimo.

A função que trata a interrupção é a função que realiza o sincronismo do DSP com a rede elétrica, portanto ela está apresentada na figura 10. A função começa armazenando o valor atual do temporizador do PWM. Em seguida, é feita a correção de fase ou de frequência.

A correção de fase é realizada apenas uma vez a cada 100 interrupções, enquanto a correção de frequência é feita nas demais. Isso ocorre porque tentar forçar a correção de fase enquanto a frequência não está sincronizada impede que ocorra o sincronismo da frequência. Sincronizar a fase a cada 100 ciclos garante que a frequência já esteja sincronizada quando ocorrer o sincronismo de fase. Além disso, conforme já explicado, a defasagem da saída do PLL em relação à rede elétrica é fixa, mesmo com alteração da frequência, desde que o PLL não perca o sincronismo, portanto não é necessário corrigir a defasagem do DSP tão frequentemente.

Figura 10 - Função que trata a interrupção gerada no pino GPIO16

```
161 /**
162  * Trata a interrupção XINT1, configurada neste projeto para
163  * ser gerada pela GPIO16 (borda de descida)
164  */
165 __interrupt void  XINT1_ISR(void)
166 {
167     perSincAux = EPwm1Regs.TBCTR; //valor atual do contador do PWM
168
169     //Sincronismo de fase (1 a cada 100 ciclos)
170     if(pulsosSincCont == 100) {
171         EPwm1Regs.TBCTR = 10125; //a rede está 4ms adiantada em relação ao PLL e 10125 = 4ms
172         pulsosSincCont = 0;
173         //estourosPWM -= 1; //esse ciclo foi ignorado pelo ajuste de frequencia
174     }
175
176     //Sincronismo de frequência
177     else {
178         //adiciona os períodos que já resetaram e subtrai a contagem anterior
179         perSinc = perSincAux + perPWMOld*estourosPWM - perSincOld;
180         estourosPWM = 0;
181
182         // Guarda as variáveis para o próximo ciclo
183         perPWMOld = EPwm1Regs.TBPRD; //o valor do período atual
184         perSincOld = perSincAux; //o valor do TBCTR quando ocorreu a interrupção
185
186         // Atualiza período da saída ePWM1
187         EPwm1Regs.TBPRD = perSinc;
188         EPwm1Regs.CMPA.half.CMPA = (perSinc+1)/2; // razão cíclica do EPWM1A
189
190         //Saídas
191         freqSinc = 50624*50/(perSinc); //frequência medida no último período em Hz
192         //50624 = 20ms, 42153 ~= 16.67ms(60Hz)
193     }
194     pulsosSincCont++;
195     GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; //LED troca de estado
196     PieCtrlRegs.PIEIFR1.bit.INTx4 = 0;  //limpa a flag
197     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //reconhece a interrupção, para poder receber mais
198 }
```

Fonte: Autoria própria

Com os componentes utilizados, a defasagem do PLL em relação à rede elétrica é de 4ms. O sincronismo de fase é feito, então, forçando o valor de 10125 como valor atual do temporizador do PWM, pois na base de tempo configurada para o módulo, 10125 equivale a 4ms. Isso adiantará o PWM em relação ao PLL, sincronizando sua fase com a da rede.

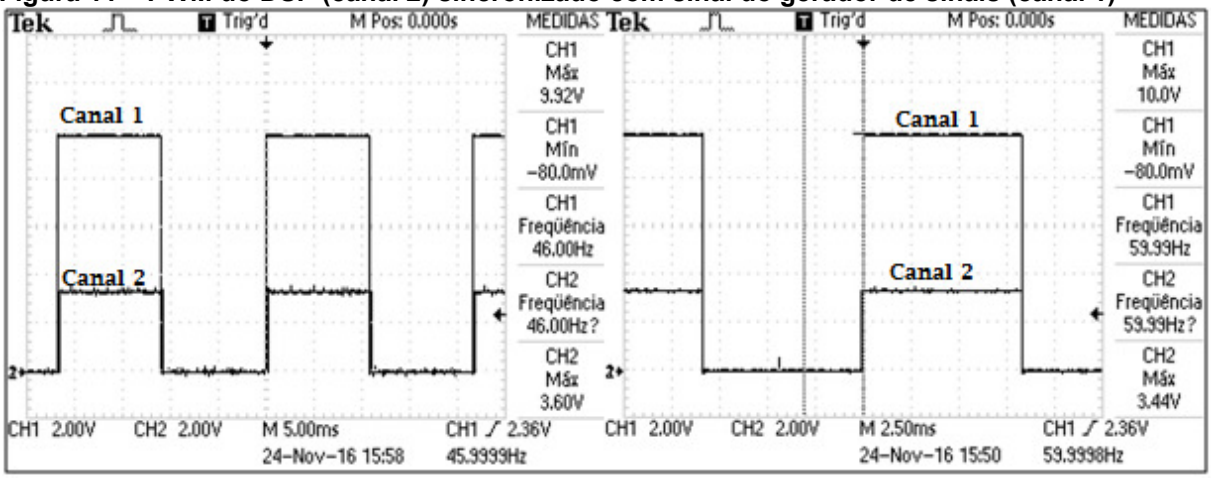
Para o sincronismo de frequência, primeiramente é calculado o intervalo de tempo desde a última interrupção. Isto é feito somando-se o valor atual do temporizador do PWM com o produto do número de vezes que o PWM atingiu seu valor máximo pelo seu período e subtraindo do valor do temporizador do PWM na interrupção anterior. O resultado da operação anterior é o número de contagens do timer do PWM desde a última interrupção e este valor é definido como o novo

período do PWM, o que faz o sincronismo de frequência do PWM com a rede elétrica.

Uma variável “freqSinc” recebe o valor calculado da frequência da rede e pode ser utilizada em projetos futuros para funções que necessitem do valor da frequência de sincronismo.

Este método de sincronismo funcionou corretamente para frequências de entrada de 46Hz a 66Hz, o que é suficiente para atender aos requisitos da NTC 905200, a norma técnica da Copel para acesso de micro e minigeração distribuída ao sistema da Copel, a qual foi explicada na seção 2. Fora desse intervalo, a correção de fase não funciona corretamente. Assume-se que seja possível aumentar o intervalo de frequências para o qual o método de sincronismo funcionaria corretamente, caso seja necessário para outras aplicações. A figura 11 apresenta o sinal vindo do gerador de sinais juntamente com a saída PWM do DSP para 46Hz (esquerda), limite inferior de correção de fase, e 60Hz (direita), demonstrando o sincronismo atingido.

Figura 11 – PWM do DSP (canal 2) sincronizado com sinal do gerador de sinais (canal 1)



Fonte: Autoria própria

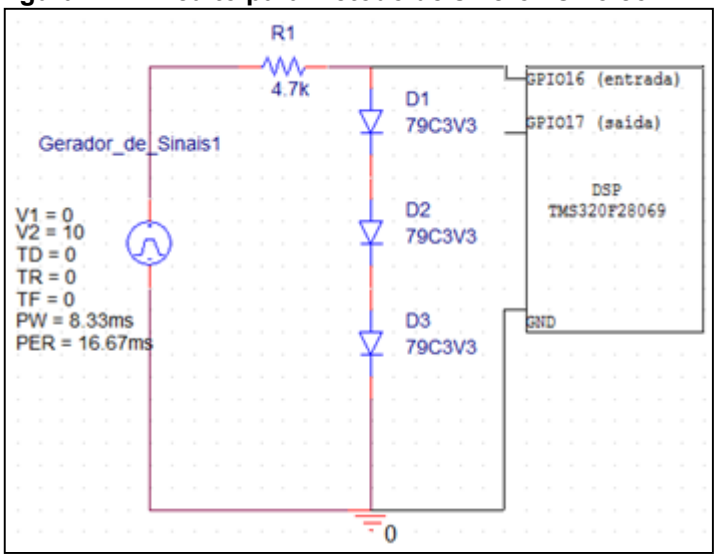
4.3 MÉTODO DE SINCRONISMO UTILIZANDO UM PLL VIRTUAL

O segundo método de sincronismo desenvolvido baseou-se em criar uma versão virtual de um circuito integrado de malha de captura de fase, ou seja, no desenvolvimento de um algoritmo capaz de imitar de maneira bastante próxima o

comportamento de um circuito integrado PLL, assim mantendo suas vantagens, como a rejeição de ruídos e estabilização rápida, e diminuindo custos e volume pela eliminação do circuito integrado e dos componentes conectados.

A figura 12 apresenta o circuito montado para este método de sincronismo. Assim como para o método anterior, o gerador de sinais representa o sinal vindo da rede elétrica já transformado em uma onda quadrada de 10V durante o semiciclo positivo e 0V durante o negativo. Esses valores especificamente não são necessários, apenas deve-se garantir que o sinal que chega ao DSP seja uma onda quadrada que possua entre 2V a 3V no semiciclo positivo da rede e entre 0V e 0,8V no semiciclo negativo. Como a montagem foi feita em *protoboard* e não havia preocupação com espaço, foram usados três diodos para o grampeamento da tensão, entretanto um diodo zener pode substituí-los para uma aplicação em placa de circuito impresso. O valor do resistor de entrada foi determinado de maneira a garantir uma tensão de polarização de aproximadamente 0,75V em cada diodo.

Figura 12 - Circuito para método de sincronismo com PLL virtual



Fonte: Autoria própria

O programa desenvolvido para este método de sincronismo imita especificamente o funcionamento do circuito da figura 4. Ele foi implementado no DSP TMS320F28069 da fabricante Texas Instruments e programado em linguagem C. O código-fonte completo pode ser encontrado no apêndice B.

Foram necessários dois pinos de entrada/saída digital, GPIO16 e GPIO17 (*general purpose input-output*), e um temporizador. GPIO16, configurado como

entrada, equivale ao pino 14 do circuito integrado PLL HEF4046B, portanto é a entrada do comparador de fase que recebe o sinal vindo da rede elétrica. GPIO17 é um pino de saída que, quando o sincronismo é atingido, é comutado na mesma frequência da rede elétrica e em fase com a mesma.

Para realização do sincronismo, o pino GPIO16 é amostrado a cada 50µs. O intervalo de amostragem é garantido, pois o temporizador gera uma interrupção a cada 50µs, garantindo que o programa interrompa o que estiver executando e execute a função que trata a interrupção. Visto que essa função realiza o sincronismo, ela será explicada em detalhes.

A primeira parte da função que trata a interrupção está apresentada na figura 13. Este segmento de código realiza a função do comparador de fase e do filtro RC passa-baixa do circuito integrado PLL. A relação entre a tensão de saída (VFOut) e a tensão de entrada (VFIn) do filtro no domínio S é apresentada na equação (1).

Figura 13 - Parte inicial da função que trata a interrupção gerada pelo temporizador

```
172  /**
173   * Trata interrupções geradas pelo timer0 a cada 50us.
174   * É utilizado para amostrar o pino GPIO16 e sincronizar o GPIO17 com a rede.
175   * Mais código poderá ser adicionado (projetos futuros) para realizar mais
176   * ações a cada 50us ou múltiplos inteiros de 50us.
177   */
178  __interrupt void  TINT0_ISR(void)
179  {
180      //Virtualização do comparador de fase e do filtro RC do PLL real
181      VFOut = 1.9e-4*VFInM1 + 0.9998*VFOutM1; //tensão no filtro virtual
182      if(GpioDataRegs.GPADAT.bit.GPIO16 != VCOOut){ //ou exclusivo
183          VFInM1 = 15; //equivalente aos 15V na saída do comparador do CD4046
184      }
185      else {
186          VFInM1 = 0;
187      }
188      VFOutM1 = VFOut;
189
```

Fonte: Autoria própria

$$VF_{out} = \frac{1}{RCs+1} \times VF_{in} \qquad (1)$$

Conforme pode ser visto na figura 4, o resistor utilizado foi de 560kΩ e o capacitor de 470nF. Substituindo-se na equação, tem-se:

$$VF_{out} = \frac{1}{0{,}2632 \times s+1} \times VF_{in} \quad (2)$$

Através do método *zero-order hold* (retentor de amostras de ordem zero), com intervalo de amostragem de 50μs, chegou-se à função de transferência no domínio discreto Z, conforme apresentado na equação (3).

$$VF_{out} = \frac{1{,}9 \times 10^{-4}}{z - 0{,}9998} \times VF_{in} \quad (3)$$

Ou na forma expandida, que equivale à função na linha 181 do código da figura 12, sendo “k” a amostra atual e “k-1” a amostra anterior:

$$VF_{out}[k] = 1{,}9 \times 10^{-4} \times VF_{in}[k-1] + 0{,}9998 \times VF_{out}[k-1] \quad (4)$$

A tensão de saída do filtro RC virtual é calculada a cada amostragem, enquanto a tensão de entrada, assim como no circuito integrado PLL, depende do comparador de fase, o qual é uma operação ou-exclusivo entre o sinal amostrado da rede elétrica (GPIO16) e o sinal de saída do oscilador controlado por tensão (VCOOut), o qual, neste caso, é uma variável de valor 1 ou 0. A tensão de entrada é 15V para sinais diferentes e 0V para sinais iguais.

A figura 14 apresenta o restante da função de sincronismo, composta pelo oscilador controlado por tensão virtual e a comutação da saída GPIO17 com correção de defasagem. Conforme explicado na seção 4.1, o oscilador controlado por tensão do circuito da figura 4 possui um capacitor que é carregado por uma corrente linearmente dependente da tensão de saída do filtro passa-baixa e toda vez que esse capacitor atinge 7,8V, ele é descarregado e a saída do oscilador é comutada.

Assim, foi definida a variável VCapacitorVirtual responsável por representar a tensão no capacitor do oscilador virtual. Sabendo-se da relação linear entre a corrente de carga e a tensão no capacitor de filtro e admitindo-se que essa corrente não varia durante o período de amostragem de 50μs, chegou-se à equação (5) para calcular a tensão no capacitor do oscilador. VFout é a tensão de saída do filtro RC passa-baixa, “n” representa o valor da amostragem atual e “n-1” da amostragem anterior.

$$V_{capacitorvirtual}[n] = K \times VF_{out}[n] + V_{capacitorvirtual}[n-1] \quad (5)$$

Figura 14 - Parte final da função que trata a interrupção gerada pelo temporizador

```
189
190     amostras += 1; //para calcular o período da rede
191
192     //Virtualização do oscilador controlado por tensão (VCO)
193     VCapacitorVirtual = (VCapacitorVirtual + 0.006592*VFOut); //vai carregando o capacitor C1
194     if (VCapacitorVirtual > 7.80) { //carregou ao maximo, comutar
195         VCOOut = !VCOOut;
196         VCapacitorVirtual = 0;
197
198         perSinc = 0.1*amostras; //período da rede em ms
199         freqSinc = 1000/perSinc; //frequência da rede em Hz
200         amostras = 0;
201         sinalParaComutar = 1; //GPIO17 deverá comutar após intervalo de correção de fase
202     }
203
204     //Comutação do GPIO17 com correção da defasagem
205     if(amostras*0.05 >= (0.5*perSinc - correcaoFase)){
206         if(sinalParaComutar == 1) {
207             sinalParaComutar = 0;
208             if(VCOOut == 1) {
209                 GpioDataRegs.GPACLEAR.bit.GPIO17 = 1; //rede em 0
210             }
211             else {
212                 GpioDataRegs.GPASET.bit.GPIO17 = 1; //rede em 1
213             }
214         }
215     }
216
217     CpuTimer0Regs.TCR.bit.TIF = 1;  //limpa a flag
218     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1; //reconhece a interrupção, para poder receber mais
219 }
```

Fonte: Autoria própria

Medições revelaram que a tensão média no capacitor de filtro quando o PLL do circuito da figura 4 está sincronizado em 60Hz é de 7,10V, com oscilações em torno de ±100mV. O capacitor do oscilador deve carregar de 0 a 7,8V em meio período da rede, neste caso, 8,333ms, e são realizadas 166,67 amostragens de 50µs nesse período. Logo, K foi calculado pela equação (6) e o valor da tensão do capacitor do oscilador pode ser calculado a cada amostragem, conforme demonstrado na linha 193 do código da figura 14.

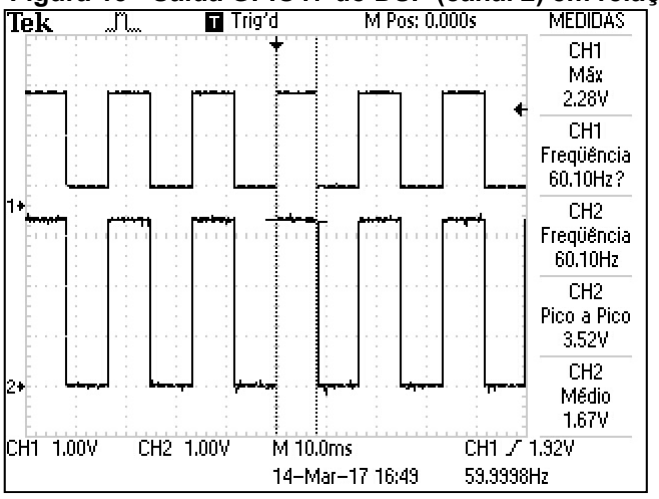
$$K = \frac{7{,}8}{7{,}1 \times 166{,}67} = 6{,}592 \times 10^{-3} \quad (6)$$

Assim que o valor da tensão no capacitor do oscilador virtual atinge ao menos 7,8V, a variável VCOOut é comutada, o valor desta tensão é zerado e o período e frequência da rede são calculados, pois é contado o número de amostras entre cada carga completa do capacitor e o intervalo entre as amostras é de 50µs.

VCOOut equivale ao pino 4 do circuito integrado PLL da figura 4, ou seja, a saída do oscilador controlado por tensão. Conforme ocorre no circuito integrado, VCOOut sincroniza em frequência com a rede elétrica, porém há uma defasagem fixa entre ambos. Essa defasagem foi medida como 4,16ms e independe da frequência, desde que a mesma esteja sincronizada.

Como esta variável defasada é necessária para o funcionamento, a correção de fase é feita sobre a saída no pino GPIO17 do DSP. Assim que o capacitor virtual do oscilador é zerado, é dado um sinal indicando que GPIO17 pode comutar. Sabendo-se o valor da defasagem fixa e do período da rede, o tempo necessário para a próxima passagem por zero da rede será a diferença entre meio período da rede e o valor da defasagem. Quando este tempo passar, calculado a partir do número de amostragens, GPIO17 é comutada, o que garante que esta saída fique sincronizada tanto em fase quanto em frequência com a rede. A figura 15 mostra a saída sincronizada do DSP em relação a uma entrada de 60Hz, demonstrando que o método de sincronismo funcionou conforme esperado.

Figura 15 - Saída GPIO17 do DSP (canal 2) em relação à entrada de 60Hz



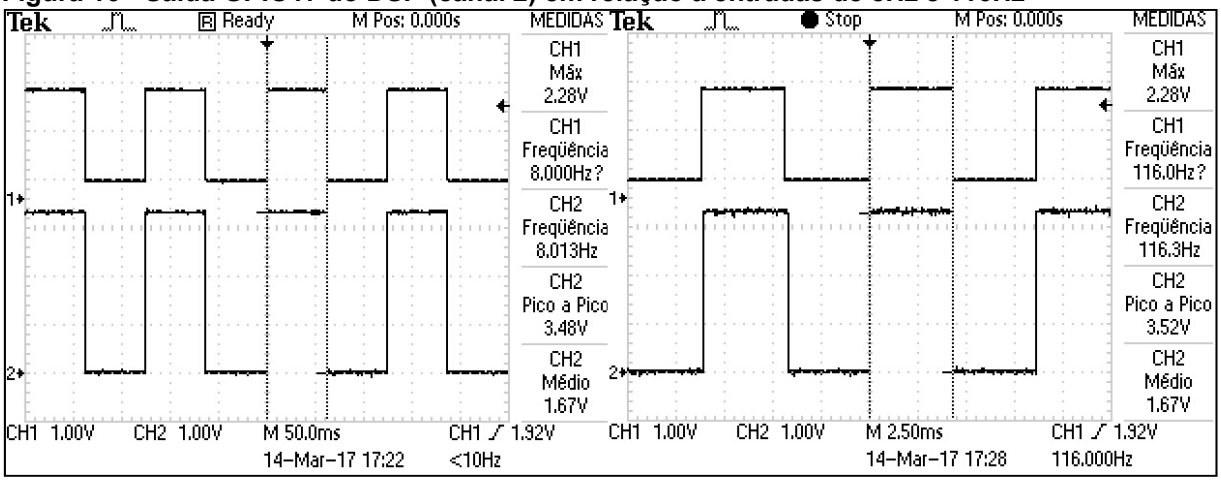
Fonte: Autoria própria

Para aplicações que utilizem este método de sincronismo, caso seja causada uma defasagem entre a rede elétrica e o sinal que efetivamente chega ao DSP, devido a transformadores, filtros, entre outros, a variável "correcaoFase" poderá ter seu valor alterado de 4,16ms para valores maiores ou menores de maneira a corrigir essa defasagem. Lembrando que, conforme explicado, 4,16ms é o valor da defasagem ente o sinal de entrada no pino GPIO16 e a variável VCOOut.

Conforme ocorre com o circuito integrado, o PLL virtual possui uma frequência central na qual sua saída é mantida na ausência de sinal da rede. O valor medido foi de 59,95Hz.

Semelhantemente a um circuito integrado PLL, a malha de captura de fase virtual possui um intervalo de frequências de travamento e de captura. O intervalo de captura, ou seja, o intervalo no qual o PLL consegue adquirir sincronismo é de 53Hz a 67Hz, enquanto o intervalo de travamento, ou seja, o intervalo no qual o PLL consegue manter o sincronismo desde que já o tenha adquirido dentro do intervalo de captura, é de 8Hz a 116Hz. A figura 16 demonstra o funcionamento nesses dois limites.

Figura 16 - Saída GPIO17 do DSP (canal 2) em relação à entradas de 8Hz e 116Hz



Fonte: Autoria própria

5 ANÁLISE DOS RESULTADOS

Conforme apresentado na seção 4, os dois métodos de sincronismo desenvolvidos, denominados PLL Externo e PLL Virtual, foram capazes de sincronizar com o sinal vindo da rede elétrica tanto em fase quanto frequência.

No quesito de faixa de frequências de atuação, a tabela 4 permite realizar uma comparação entre os métodos. Conforme explicado anteriormente, o intervalo de captura é o intervalo de frequências no qual o PLL consegue adquirir sincronismo e o intervalo de travamento é o intervalo de frequências no qual o sincronismo é mantido caso já tenha sido adquirido dentro do intervalo de captura.

Tabela 4 – Faixa de frequências de atuação dos métodos desenvolvidos

Intervalo	PLL Externo	PLL Virtual
Captura	46Hz a 66Hz	53Hz a 67Hz
Travamento	46Hz a 66Hz	8Hz a 116Hz

Fonte: Autoria própria

Na seção 2 foi apresentada a NTC (norma técnica Copel) 905200 que estabelece os requisitos para acesso de micro e minigeração distribuída ao sistema da Copel com compensação de energia. Conforme visto, são necessários relés de sobre e subfrequência para proteção no ponto de conexão, proteção que deve ser também implantada nos conversores de frequência. Para uma subfrequência de 56,5Hz ou uma sobrefrequência de 66Hz, a proteção deve atuar instantaneamente desconectando o conversor da rede. Ambos os métodos desenvolvidos são capazes de manter o sincronismo até esses limites, logo poderiam ser utilizados com conversores de frequência para geração fotovoltaica distribuída de pequeno porte considerando este quesito.

O intervalo de travamento do método PLL Virtual foi muito maior que do PLL Externo, sendo superior, inclusive, do que o *Enhanced* PLL apresentado na seção 3.1 que possui faixa de atuação de 20Hz a 120Hz.

No quesito de tempo de estabilização, do momento em que o sinal da rede é conectado ao processador digital de sinais até o sincronismo de frequência e fase, podem ser necessários até 100 ciclos da rede elétrica para o método PLL Externo e 14 ciclos para o PLL Virtual no pior caso possível. Entretanto, caso já haja

sincronismo e a frequência da rede for alterada, a nova estabilização levará no máximo 4 ciclos para o método PLL Externo e 5 ciclos para o PLL Virtual.

O número elevado de ciclos para a estabilização inicial do PLL Externo deve-se à maneira como é feito seu sincronismo de fase. Caso necessário, acredita-se poder reduzir bastante esse número com pequenas alterações no código-fonte do método. Contudo, a estabilização inicial será necessária raras vezes, enquanto a estabilização após sincronismo inicial é suficientemente rápida.

No quesito de imunidade a ruídos e distorções não foram realizados testes extensivos sobre os métodos desenvolvidos. Entretanto, a função de um circuito integrado de malha de captura de fase é exatamente fornecer na sua saída um sinal sincronizado em frequência com a frequência da componente fundamental do sinal da rede, eliminando ruídos e distorções harmônicas. Visto que o método PLL Externo utiliza este circuito integrado e o método PLL Virtual imita o funcionamento dele, pode-se assumir que os dois métodos são altamente imunes a ruídos e distorções harmônicas.

O método PLL Virtual possui vantagens sobre o PLL Externo na faixa de frequências de atuação e no tempo de estabilização inicial, além de eliminar a necessidade de diversos componentes como 2 capacitores, 3 resistores, um circuito integrado PLL, além de possivelmente uma fonte. As desvantagens do PLL Virtual em relação ao PLL Externo estão no tempo de estabilização quando já há sincronismo e a frequência se altera e no esforço de processamento no DSP, o qual é consideravelmente maior no PLL Virtual, apesar deste método não exigir tanto do processamento.

6 CONCLUSÃO

Ao longo deste trabalho, estudou-se o funcionamento de malhas de captura de fase (PLL) e métodos de sincronismo da rede elétrica comercial com processadores digitais de sinais. Além disso, foram desenvolvidos e testados dois métodos de sincronismo, os quais foram comparados entre eles e com métodos encontrados na literatura e verificada sua aplicabilidade. Logo, todos os objetivos deste projeto foram cumpridos.

A problemática apresentada foi solucionada e a hipótese confirmada, pois os métodos de sincronismo desenvolvidos são simples, eficazes e capazes de operar dentro dos parâmetros de uma das normas para geração fotovoltaica distribuída estudada.

Geração distribuída de energia fotovoltaica requer métodos de sincronismo de processadores digitais de sinais com a rede elétrica. Conforme demonstrado, existem diversos métodos de sincronismo na literatura, entretanto estes são raramente apresentados com todas as informações necessárias para sua implementação e, em sua maioria, necessitam de elevados esforços de processamento ou circuitos externos complexos, volumosos e/ou custosos, além de alguns serem patenteados. Considerando a tendência de crescimento da geração de energia fotovoltaica no mundo e o potencial brasileiro demonstrados, este trabalho pôde contribuir com esta área, apresentando dois novos métodos que obtiveram bons resultados, em especial o método de sincronismo utilizando um PLL virtual, que podem ser utilizados nesta aplicação e, possivelmente, em outras.

REFERÊNCIAS

IEA (International Energy Agency*)*. **Technology Roadmap:** Solar Photovoltaic Energy. Paris, França, 2014.

EPE (Empresa de Pesquisa Energética). **Inserção da Geração Fotovoltaica Distribuída no Brasil – Condicionantes e Impactos.** Rio de Janeiro, 2014.

VENDRICHOSKI, J. C.; DURSKI, Y.; CASARO, M. M. Método preciso de sincronismo do DSP com a rede elétrica. In: SEMANA DE ELETRÔNICA E AUTOMAÇÃO. 1, 2010, Ponta Grossa. **Anais do I Seminário de Eletrônica e Automação.** Ponta Grossa: Universidade Tecnológica Federal do Paraná, 2010.

ANEEL (Agência Nacional de Energia Elétrica). **Banco de Informações de Geração.** Abr. 2017. Disponível em: <http://www2.aneel.gov.br/aplicacoes/capacidadebrasil/capacidadebrasil.cfm>. Acesso em: 06 abr. 2017.

GOMES, K. **Para evitar crise, Brasil precisa diversificar matriz energética.** Fev. 2014. Disponível em: <http://p.dw.com/p/1B3Fq>. Acesso em: 05 abr. 2017.

EPE (Empresa de Pesquisa Energética). **Análise da Inserção da Geração Solar na Matriz Elétrica Brasileira.** Rio de Janeiro, 2012.

COPEL (Companhia Paranaense de Energia). **Geração Distribuída.** Jun. 2016. Disponível em: <http://www.copel.com/hpcopel/root/nivel2.jsp?endereco=%2Fhpcopel%2Froot%2Fpagcopel2.nsf%2Fdocs%2FA3C753509472FD030325781000637369>. Acesso em: 08 abr. 2017.

COPEL (Companhia Paranaense de Energia). **NTC 905200.** Fev. 2014. Disponível em: <http://www.copel.com/hpcopel/root/ntcarquivos.nsf/E00A539C1F08DF2003257F69004DF8BC/$FILE/NTC%20905200%20Acesso%20de%20Micro%20e%20Minigera%C3%A7%C3%A3o%20Distribu%C3%ADda.pdf>. Acesso em: 08 abr. 2017.

CHOI, J.W.; KIM, Y.K.; KIM, H.G. Digital PLL control for single-phase photovoltaic system. **IEE Proceedings Electric Power Applications,** v. 153, n. 1, p. 40-46, fev. 2006.

GARDNER, F. M. **Phase-Lock Techniques.** New York: John Wiley and Sons, 1966.

GUO, X.Q.; WU, W.Y.; GU, H.R. Phase locked loop and synchronization methods for grid-interfaced converters: a review. **Przeglad Elektrotechniczny,** v. 2011, n. 4, p.182-187, abr. 2011.

KARIMI-GHARTEMANI, M.; IRAVANI, M.R. A New Phase-Locked Loop (PLL) System. In: MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS. 44, 2001, Dayton, Estados Unidos. **Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems.** Dayton, Estados Unidos: Promatch Printing, 2001. p. 421-424.

KARIMI-GHARTEMANI, M.; IRAVANI, M. R. A nonlinear adaptive filter for online signal analysis in power systems: applications. **IEEE Transactions on Power Delivery,** v. 17, n. 2, p.617-622, abr. 2002.

TIMORABADI, H. S.; DAWSON, F.P. A Wide-Range Synchronization System for AC Power Systems. In: IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS. 4, 2006, Montréal, Canadá. **2006 IEEE International Symposium on Industrial Electronics (ISIE'06).** Montréal, Canada: The Institute of Electrical and Electronics Engineers, Inc., 2006. p. 1667-1672.

MORGAN, D. K. **CD4046B Phase-Locked Loop: A Versatile Building Block for Micropower Digital and Analog Applications.** Fev. 2003. Disponível em: <http://www.ti.com/lit/an/scha002a/scha002a.pdf>. Acesso em: 05 abr. 2017.

APÊNDICE A - Código-fonte do programa do método de sincronismo utilizando um PLL externo

```c
//Projeto: PLL externo
//Versão: 24 de Novembro de 2016
//Autor  : Felipe dos Passos Canteri

/**
 * Este projeto realiza o sincronismo do DSP com a rede elétrica,
 * utilizando um PLL externo.
 * A GPIO 16 está configurada como interrupção externa. Ao receber uma borda de
 * descida, a interrupção é gerada, fazendo o LED da placa alternar seu estado.
 * O LED está inicialmente aceso e a GPIO 16 em aberto mantém nível lógico 1.
 * O pino GPIO00 está configurado como PWM de razão cíclica 50%. Inicialmente com
 * período 20ms, mas que atualiza quando a interrupção externa é gerada.
 * Quando a GPIO16 foi conectada ao PLL sintonizado em 60Hz, a saída do PWM ficou
 * com 60Hz.
 * O PWM ficou em fase com a rede, adiantado em relação ao PLL.
 */

#include "F2806x_Device.h"
#include "F2806x_GlobalPrototypes.h"
#include "math.h"

// Declaração de funções
void inicializar(void);
void configurarPortasIO(void);
void habilitarInterrupcao(void);
__interrupt void  XINT1_ISR(void);
__interrupt void EPWM1_INT_ISR(void);
void habilitarPWM(void);

// Declaração de variáveis
float perTimer0;    //período do timer0
float contTimer0;   //contagem atual do timer0 (crescente). A real é decrescente
//variáveis para sincronizar o PWM com a rede
float perSinc = 0;
float perSincOld = 50624;
float perSincAux = 0;
float freqSinc = 0; //frequência medida vindo do PLL no pino GPIO 16
float perPWMOld = 50624;
float estourosPWM = 0;
int pulsosSincCont = 0; //usado para corrigir a defasagem
//
int i = 0;  //índices de loops

void main(void)
{
    inicializar();
    ConfigCpuTimer(&CpuTimer0, 81, 20000); //81MHz(clk do SYSCLKOUT), 1000 = 1ms
    configurarPortasIO();
    habilitarPWM();
    habilitarInterrupcao();

    //***************fim das configurações*****************//

    perTimer0 = CpuTimer0Regs.PRD.all; //periodo do timer
    CpuTimer0Regs.TCR.bit.TSS = 0;  //timer começa a operar
    // Loop
    while(1)
    {
        //periodo-valor atual do timer
        contTimer0 = perTimer0-CpuTimer0Regs.TIM.all;
    }
}

/**
 * Inicialização geral do DSP
 * Reseta GPIOs para o estado padrão, desabilita e limpa interrupções
 */
void inicializar(void)
```

```
69  {
70      // Initialize System Control:
71      // PLL, WatchDog, enable Peripheral Clocks
72      // This function is found in the DSP2802x_SysCtrl.c file.
73         InitSysCtrl();
74
75      // Disable CPU interrupts and clear all CPU interrupt flags:
76         IER = 0x0000;    // Disable CPU interrupts
77         IFR = 0x0000;    // Clear all CPU interrupt flags
78
79      // Initialize the PIE control registers to their default state.
80      // The default state is all PIE interrupts disabled and flags
81      // are cleared.
82      // This function is found in the DSP2802x_PieCtrl.c file.
83         InitPieCtrl();
84
85      // Initialize the PIE vector table with pointers to the shell Interrupt
86      // Service Routines (ISR).
87      // This will populate the entire table, even if the interrupt
88      // is not used in this program.  This is useful for debug purposes.
89      // The shell ISR routines are found in DSP2802x_DefaultIsr.c.
90      // This function is found in DSP2802x_PieVect.c.
91         InitPieVectTable();
92
93      // Initialize GPIO:
94      // This function is found in the DSP2802x_Gpio.c file and
95      // illustrates how to set the GPIO to it's default state.
96         InitGpio();
97
98      // Only used if running from FLASH (MemCopy and InitFlash)
99      // Copy time critical code and Flash setup code to RAM
100     // The  RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
101     // symbols are created by the linker. Refer to the linker files.
102        MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
103     // Call Flash Initialization to setup flash waitstates
104     // This function must reside in RAM
105        InitFlash(); // Call the flash wrapper init function
106
107     // Initialize the Device Peripheral. This function can be
108     // found in DSP2802x_CpuTimers.c
109        InitCpuTimers();
110 }
111
112 /**
113  * Configura direção e valor inicial das portas GPIOs que serão usadas
114  */
115 void configurarPortasIO(void)
116 {
117     // Configura pinos I/O das portas A e B
118     // Porta A consiste em GPIO0-GPIO31, port B consiste em GPIO32-GPIO58.
119     // Portas analógicas são AIO0-AIO15.
120     EALLOW; //permite mexer em registradores protejidos
121
122     // Esses registradores definem a função dos pinos. 00 = pino é uma General
123     // Purpose I/O. 01 a 11 permitem que o pino tenha outras funções. Exemplo
124     // GPIO34 = 01 transforma o GPIO34 na saída do COMP2OUT (O). O padrão é 00.
125     GpioCtrlRegs.GPAMUX2.bit.GPIO17 = 0x00; //GPIO
126     GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0x00; //GPIO conectada ao led
127
128     // Esses registradores definem a direção dos pinos.
129     // 0 = input (padrão), 1 = output.
130     // Só tem efeito se o pino estiver definido como GPIO
131     GpioCtrlRegs.GPADIR.bit.GPIO16 = 0;
132     GpioCtrlRegs.GPADIR.bit.GPIO17 = 1;
133     GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1; //LED
134
135     // Registradores gpa(b)dat, set, clear e toggle lêem e escrevem valores
136     // nos pinos se estes estiverem configurados como GPIO. dat serve para tudo.
```

```
137     // Já set, clear e toogle executam ao receber 1.
138     GpioDataRegs.GPASET.bit.GPIO17 = 1;     //coloca 1 no pino 17
139     GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;   //coloca 0 no pino 34 (LED acende)
140
141     EDIS; //desabilita mexer em registradores protegidos
142 }
143
144 /**
145  * Habilita interrupção por borda de descida na GPIO 16
146  * Também habilita interrupção gerada pelo ePWM1
147  */
148 void habilitarInterrupcao(void)
149 {
150     // Função do F2806x_PieCtrl.c
151     //Habilita módulo PIE e habilita envio de interrupções ao CPU, mas
152     //não habilita grupos INT1...INT12 especificamente.
153     EnableInterrupts();
154
155     EALLOW;
156     //Habilitando a interrupção na GPIO 16
157     GpioIntRegs.GPIOXINT1SEL.all = 16; //XINT1 recebe sinais da GPIO 16
158     //XINT1 gera interrupção na borda de descida e habilitada para nível PIE
159     XIntruptRegs.XINT1CR.all = 1;
160     PieCtrlRegs.PIEIER1.bit.INTx4 = 1; //habilita o envio do XINT1 ao nível CPU
161     //EPWM1
162     PieCtrlRegs.PIEIER3.bit.INTx1 = 1; //habilita o envio do EPWM1_INT ao CPU
163     //habilita grupos INT1 e INT3 a nível CPU, eles contém o XINT1 e EPWM1_INT
164     IER = 5;
165     EDIS;
166 }
167
168 /**
169  * Trata a interrupção XINT1, configurada neste projeto para
170  * ser gerada pela GPIO16 (borda de descida)
171  */
172 __interrupt void  XINT1_ISR(void)
173 {
174     perSincAux = EPwm1Regs.TBCTR; //valor atual do contador do PWM
175
176     //Sincronismo de fase (1 a cada 100 ciclos)
177     if(pulsosSincCont == 100) {
178         //a rede está 4ms adiantada em relação ao PLL e 10125 = 4ms
179         EPwm1Regs.TBCTR = 10125;
180         pulsosSincCont = 0;
181         //estourosPWM -= 1; //esse ciclo foi ignorado pelo ajuste de frequencia
182     }
183
184     //Sincronismo de frequência
185     else {
186         //adiciona os períodos que já resetaram e subtrai a contagem anterior
187         perSinc = perSincAux + perPWMOld*estourosPWM - perSincOld;
188         estourosPWM = 0;
189
190         // Guarda as variáveis para o próximo ciclo
191         perPWMOld = EPwm1Regs.TBPRD; //o valor do período atual
192         perSincOld = perSincAux; //o valor do TBCTR quando ocorreu a interrupção
193
194         // Atualiza período da saída ePWM1
195         EPwm1Regs.TBPRD = perSinc;
196         EPwm1Regs.CMPA.half.CMPA = (perSinc+1)/2; // razão cíclica do EPWM1A
197
198         //Saídas
199         freqSinc = 50624*50/(perSinc); //frequência medida no último período em Hz
200         //50624 = 20ms, 42153 ~= 16.67ms(60Hz)
201     }
202     pulsosSincCont++;
203     GpioDataRegs.GPBTOGGLE.bit.GPIO34 = 1; //LED troca de estado
204     PieCtrlRegs.PIEIFR1.bit.INTx4 = 0;  //limpa a flag
```

```
205         //reconhece a interrupção, para poder receber mais desse grupo PIE
206         PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
207     }
208
209     /**
210      * Trata a interrupção gerada pelo ePWM1 quando TBCTR = TBPRD
211      */
212     __interrupt void EPWM1_INT_ISR(void)
213     {
214         estourosPWM++;
215         EPwm1Regs.ETCLR.bit.INT = 1; //limpa a flag
216         PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;
217     }
218
219     /**
220      * Habilita o módulo ePWM1 para ser usado. Ele é a saída sincronizada com
221      * o gerador de sinais ou a rede elétrica.
222      */
223     void habilitarPWM(void)
224     {
225         float periodo = 50624; //o período irá incluir o 0, sendo 50625
226
227         //configurando os pinos
228         EALLOW;
229         GpioCtrlRegs.GPAPUD.bit.GPIO0 = 1;  // Desabilita pull-up na GPIO0 (EPWM1A)
230         GpioCtrlRegs.GPAPUD.bit.GPIO1 = 1;  // Desabilita pull-up na GPIO1 (EPWM1B)
231         GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1; // Configura GPIO0 como EPWM1A
232         GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1; // Configura GPIO1 como EPWM1B
233         EDIS;
234
235         EALLOW;
236         //O período dos PWMs é de 16 bits. Com as configurações abaixo,
237         //o módulo começa com PWM de 50Hz sem defasagem.
238         EPwm1Regs.TBPRD = periodo;  //o período de contagens de TBCLK(time-base clock)
239
240         //Desativa registradores de alta resolução
241         EPwm1Regs.HRPCTL.bit.HRPE = 0;
242         EPwm1Regs.HRPCTL.bit.TBPHSHRLOADE = 0;
243
244         //Registrador com valor da defasagem
245         EPwm1Regs.TBPHS.all = 0.75*periodo; //90 graus de defasagem
246
247         //Registrador Time-Base Control
248         EPwm1Regs.TBCTL.bit.CTRMODE = 0; //contagem de 0 a TBPRD
249         EPwm1Regs.TBCTL.bit.PRDLD = 0; //o período é lido do shadow register
250         EPwm1Regs.TBCTL.bit.SYNCOSEL = 0; //esse módulo repassa pulsos de sinc. que recebe
251         EPwm1Regs.TBCTL.bit.CLKDIV = 5; //divide o clock por 32
252         EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0; //divide o clock por 1. Total será 32.
253         //o registrador de defasagem é carregado no contador quando ocorre ev. de sinc.
254         EPwm1Regs.TBCTL.bit.PHSEN = 1;
255         //EPwm1Regs.TBCTL.bit.SWFSYNC = 1; //aplica a defasagem uma vez
256
257         //Registrador Counter-Compare Control Register
258         EPwm1Regs.CMPCTL.bit.SHDWAMODE = 0; //registrador shadow é usado ao invés do que
259         EPwm1Regs.CMPCTL.bit.SHDWBMODE = 0; // controla o pwm diretamente
260         EPwm1Regs.CMPCTL.bit.LOADAMODE = 0; //Período e outras configs são carregadas quando o
261         EPwm1Regs.CMPCTL.bit.LOADBMODE = 0; // contador do PWM zera
262
263         //Registrador Action-Qualifier Submodule
264         EPwm1Regs.AQCTLA.bit.PRD = 1; //Ação quando TBCTR = período. (clear)
265         EPwm1Regs.AQCTLA.bit.CAU = 2; //Ação quando TBCTR = CPMA e CTR está aumentando.(set)
266
267         EPwm1Regs.CMPA.half.CMPA = (periodo+1)/2; // razão cíclica do EPWM1A
268
269         //Event-Trigger Submodule, esses registradores permiter gerar uma flag a todo ciclo
270         //do PWM, entre outros. A flag será útil no ajuste do período
271         EPwm1Regs.ETSEL.bit.INTSEL = 2; //flag gerada quando TBCTR = TBPRD
272         EPwm1Regs.ETSEL.bit.INTEN = 1; //gera interrupção
```

```
273         EPwm1Regs.ETPS.bit.INTPRD = 1; //evento só precisa ocorrer uma vez p/gerar interrupção
274         EDIS;
275     }
276
277     //===========================================================================
278     // Fim do código.
279     //===========================================================================
280
```

APÊNDICE B - Código-fonte do programa do método de sincronismo utilizando um PLL virtual

```c
//Projeto: PLL Virtual
//Versão: 14 de março de 2017
//Autor  : Felipe dos Passos Canteri

/**
 * Este projeto realiza o sincronismo do DSP com a rede elétrica, utilizando um PLL
 *  virtual equivalente ao CD4046.
 *
 * O sinal vindo da rede deve ser tratado conforme necessário com amplificadores,
 * transformadores, etc, de maneira a formar uma onda quadrada que possua entre 2V a 3V
 * no semiciclo positivo da rede e entre 0V e 0,8V no semiciclo negativo.
 *
 * Esse sinal deve ser conectado ao pino GPIO16 e é amostrado a cada 50us. Esse sinal é
 * comparado com uma variável digital, a qual, quando ocorre o sincronismo possui a mesma
 * frequência da rede, mas é defasada em aproximadamente 4,16ms. Essa defasagem é fixa
 * quando a frequência é sincronizada, independente da frequência.
 *
 * O pino GPIO17 está configurado como saída e fica sincronizado tanto em frequência
 * quanto em fase com a rede. A variável "correcaoFase" é usada para corrigir a defasagem.
 * Caso a defasagem em relação à rede seja diferente da padrão (4,16ms), devido à
 * inserção de um filtro, entre outros, o usuário poderá modificar essa variável.
 *
 * O DSP consegue manter o sincronismo de fase e frequência desde 8Hz a 116Hz (intervalo
 * de travamento), mas só consegue garantidamente adquirir sincronismo entre 53Hz e 67Hz
 * (intervalo de captura). Na falta de sinal de entrada, o GPIO17 tem frequência 59,95Hz
 * (frequência central).
 */

#include "F2806x_Device.h"
#include "F2806x_GlobalPrototypes.h"
#include "math.h"

// Declaração de funções
void inicializar(void);
void configurarPortasIO(void);
void habilitarInterrupcao(void);
__interrupt void  TINT0_ISR(void);

// Declaração de variáveis

//--Timer--//
float perTimer0;    //período do timer0
float contTimer0;   //contagem atual do timer0 (crescente). A real é decrescente

//--Variáveis para sincronizar com a rede--//
float freqSinc = 0; //frequência da rede (Hz)
float perSinc = 0;  //período da rede (ms)
float VCapacitorVirtual = 0;    //tensão no capacitor C1 virtual (pinos 6 e 7 do PLL real)
float VFInM1 = 15;      //tensão na entrada do filtro RC virtual (amostra anterior)
float VFOut = 0;        //tensão na saída do filtro RC virtual (valor atual)
float VFOutM1 = 0;      //tensão na saída do filtro RC virtual (amostra anterior)
float amostras = 0;     //estouros do timer0, usado para calcular o período da rede
float VCOOut = 0;       //variável que é comutada para comparar com a entrada.Pino 4 do CI
float correcaoFase = 4.16; //defasagem em ms entre borda de subida da rede e do VCOOut
int sinalParaComutar = 0; //indica para GPIO17 comutar após intervalo para corrigir fase

//--Outras variáveis--//
int i = 0;  //índices de loops

void main(void)
{
    inicializar();
    ConfigCpuTimer(&CpuTimer0, 81, 50); //81MHz(clock do SYSCLKOUT), 50 = 50us
    configurarPortasIO();
    habilitarInterrupcao();
    //***************fim das configurações*****************//
```

```
67
68      perTimer0 = CpuTimer0Regs.PRD.all; //periodo do timer
69      CpuTimer0Regs.TCR.bit.TSS = 0;  //timer começa a operar
70      // Loop
71      while(1)
72      {
73          //a linha seguinte não tem função alguma neste projeto, apenas ocupa o while
74          contTimer0 = perTimer0-CpuTimer0Regs.TIM.all;   //período - valor atual do timer
75      }
76  }
77
78  /**
79   * Inicialização geral do DSP
80   * Reseta GPIOs para o estado padrão, desabilita e limpa interrupções
81   */
82  void inicializar(void)
83  {
84      // Initialize System Control:
85      // PLL, WatchDog, enable Peripheral Clocks
86      // This function is found in the DSP2802x_SysCtrl.c file.
87         InitSysCtrl();
88
89      // Disable CPU interrupts and clear all CPU interrupt flags:
90         IER = 0x0000;    // Disable CPU interrupts
91         IFR = 0x0000;    // Clear all CPU interrupt flags
92
93      // Initialize the PIE control registers to their default state.
94      // The default state is all PIE interrupts disabled and flags
95      // are cleared.
96      // This function is found in the DSP2802x_PieCtrl.c file.
97         InitPieCtrl();
98
99      // Initialize the PIE vector table with pointers to the shell Interrupt
100     // Service Routines (ISR).
101     // This will populate the entire table, even if the interrupt
102     // is not used in this program.  This is useful for debug purposes.
103     // The shell ISR routines are found in DSP2802x_DefaultIsr.c.
104     // This function is found in DSP2802x_PieVect.c.
105        InitPieVectTable();
106
107     // Initialize GPIO:
108     // This function is found in the DSP2802x_Gpio.c file and
109     // illustrates how to set the GPIO to it's default state.
110        InitGpio();
111
112     // Only used if running from FLASH (MemCopy and InitFlash)
113     // Copy time critical code and Flash setup code to RAM
114     // The  RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
115     // symbols are created by the linker. Refer to the linker files.
116        MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
117     // Call Flash Initialization to setup flash waitstates
118     // This function must reside in RAM
119        InitFlash(); // Call the flash wrapper init function
120
121     // Initialize the Device Peripheral. This function can be
122     // found in DSP2802x_CpuTimers.c
123        InitCpuTimers();
124 }
125
126 /**
127  * Configura direção e valor inicial de portas GPIOs
128  */
129 void configurarPortasIO(void)
130 {
131     // Configura pinos I/O das portas A e B
132     // Porta A consiste em GPIO0-GPIO31, port B consiste em GPIO32-GPIO58.
```

```
133     // Portas analógicas são AIO0-AIO15.
134     EALLOW; //permite mexer em registradores protejidos
135
136     // Esses registradores definem a função dos pinos. 00 = pino é uma General Purpose
137     // I/O. 01 a 11 permitem que o pino tenha outras funções. Exemplo GPIO34 = 01
138     // transforma o GPIO34 na saída do COMP2OUT (O). O padrão é 00 (GPIO).
139     GpioCtrlRegs.GPAMUX2.bit.GPIO17 = 0x00; //GPIO
140     GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0x00; //GPIO conectada ao led
141
142     // Esses registradores definem a direção dos pinos. 0 = input (padrão), 1 = output.
143     // Só tem efeito se o pino estiver definido como GPIO
144     GpioCtrlRegs.GPADIR.bit.GPIO16 = 0;
145     GpioCtrlRegs.GPADIR.bit.GPIO17 = 1;
146     GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1; //LED
147
148     //Registradores gpa(b)dat, set, clear e toggle lêem e escrevem valores nos pinos se
149     //estes estiverem configurados como GPIO. dat serve para tudo. Já set, clear e toogle
150     //executam ao receber 1.
151     GpioDataRegs.GPASET.bit.GPIO17 = 1;     //coloca 1 no pino 17
152     GpioDataRegs.GPBCLEAR.bit.GPIO34 = 1;   //coloca 0 no pino 34 (LED acende)
153
154     //Escolhe número de amostragens para modificar valor do registrador associado ao pino
155     //Útil para não detectar ruídos
156     //GpioCtrlRegs.GPAQSEL1.bit.GPIO0 = 1; //3 amostragens em 3 SYSCLKOUTs
157     //GpioCtrlRegs.GPAQSEL2.bit.GPIO16 = 1;
158
159     EDIS; //desabilita mexer em registradores protegidos
160 }
161
162 /**
163  * Habilita geração de interrupção pelo timer0
164  */
165 void habilitarInterrupcao(void)
166 {
167     // Função do F2806x_PieCtrl.c
168     //Habilita módulo PIE e habilita envio de interrupções ao CPU, mas
169     //não habilita grupos INT1...INT12 especificamente.
170     EnableInterrupts();
171
172     EALLOW;
173     CpuTimer0Regs.TCR.bit.TIE = 1; //timer0 gera interrupção
174     PieCtrlRegs.PIEIER1.bit.INTx7 = 1; //habilita o envio do TINT0 ao nível CPU
175     IER = 1; //habilita grupo INT1 a nível CPU
176     EDIS;
177 }
178
179 /**
180  * Trata interrupções geradas pelo timer0 a cada 50us.
181  * É utilizado para amostrar o pino GPIO16 e sincronizar o GPIO17 com a rede.
182  * Mais código poderá ser adicionado (projetos futuros) para realizar mais
183  * ações a cada 50us ou múltiplos inteiros de 50us.
184  */
185 __interrupt void  TINT0_ISR(void)
186 {
187     //Virtualização do comparador de fase e do filtro RC do PLL real
188     VFOut = 1.9e-4*VFInM1 + 0.9998*VFOutM1; //tensão no filtro virtual
189     if(GpioDataRegs.GPADAT.bit.GPIO16 != VCOOut){ //ou exclusivo
190         VFInM1 = 15; //equivalente aos 15V na saída do comparador do CD4046
191     }
192     else {
193         VFInM1 = 0;
194     }
195     VFOutM1 = VFOut;
196
197     amostras += 1; //para calcular o período da rede
198
```

```
199     //Virtualização do oscilador controlado por tensão (VCO)
200     VCapacitorVirtual = (VCapacitorVirtual + 0.006592*VFOut); //vai carregando o capacitor
201     if (VCapacitorVirtual > 7.80) { //carregou ao maximo, comutar
202         VCOOut = !VCOOut;
203         VCapacitorVirtual = 0;
204
205         perSinc = 0.1*amostras; //período da rede em ms
206         freqSinc = 1000/perSinc; //frequência da rede em Hz
207         amostras = 0;
208         sinalParaComutar = 1; //GPIO17 deverá comutar após intervalo de correção de fase
209     }
210
211     //Comutação do GPIO17 com correção da defasagem
212     if(amostras*0.05 >= (0.5*perSinc - correcaoFase)){
213         if(sinalParaComutar == 1) {
214             sinalParaComutar = 0;
215             if(VCOOut == 1) {
216                 GpioDataRegs.GPACLEAR.bit.GPIO17 = 1; //rede em 0
217             }
218             else {
219                 GpioDataRegs.GPASET.bit.GPIO17 = 1; //rede em 1
220             }
221         }
222     }
223
224     CpuTimer0Regs.TCR.bit.TIF = 1;  //limpa a flag
225     //reconhece a interrupção, para poder receber mais desse grupo PIE
226     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
227 }
228
229 //===========================================================================
230 // Fim do código.
231 //===========================================================================
```