Débora Freitas de Andrade

Relatório de Estágio Supervisionado Laboratório de Eletrônica Industrial e Acionamento de Máquinas

Campina Grande - Paraíba

Abril de 2016

Débora Freitas de Andrade

Relatório de Estágio Supervisionado
Laboratório de Eletrônica Industrial e Acionamento de Máquinas

Relatório de Estágio Supervisionado submetido à Coordenadoria do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Engenharia Elétrica.

Universidade Federal de Campina Grande – UFCG

Departamento de Engenharia Elétrica

Orientador: Dr. Antônio Marcus Nogueira Lima

Campina Grande - Paraíba

Abril de 2016

Débora Freitas de Andrade

Relatório de Estágio Supervisionado
Laboratório de Eletrônica Industrial e Acionamento de Máquinas

Relatório de Estágio Supervisionado submetido à Coordenadoria do Curso de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Engenharia Elétrica.

Trabalho aprovado em: 11/04/2016

Dr. Antônio Marcus Nogueira Lima
Orientador

Professor
Avaliador

Campina Grande - Paraíba
Abril de 2016

Agradecimentos

Agradeço à Deus por ter me dado saúde e força para superar as dificuldades.

Agradeço ao meu pais, Luiza e Lenimar, pelo seu esforço sobrenatural, por sempre ter incentivado meus sonhos, e que, com muito carinho e apoio, não mediram esforços para que eu chegasse até essa etapa de minha vida.

Aos meus irmãos Diana, Euler e Marina, que tanto amo.

A meu orientador, professor Dr. Antônio Marcus Nogueira Lima, pelo alto nível de orientação, contribuição e disponibilidade de tempo.

A professora Dr. Camila Gehrke e a Italo Pinheiro por toda paciência, auxílio e atenção.

Meus agradecimentos aos amigos que fizeram parte da minha formação e que me ajudaram durante essa caminhada.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

DSC	Controlador Digital de Sinal (*Digital Signal Controller*)
CCS	(*Code Composer Studio*)
APLC	Compensador Ativo de Potência (*Active Power Line Conditioner*)
ADC	Conversor Digital-Analógico (*Analog to Digital Converter*)
DAC	Conversor Analógico-Digital (*Digital to Analog Converter*)
PWM	Modulação por Largura de Pulso(*Pulse Width Modulation*)
DSP	Processador Digital de Sinal (*Digital Signal Processor*)
MCU	Microcontrolador (*Microcontroller*)
JTAG	(*Joint Test Action Group*)
SRAM	(*Static Random Access Memory*)
USB	(*Universal Serial Bus*)
CAN	(*Controller Area Network*)
GPIO	(*General Purpose Input/Output*)
LED	(*Light Emitting Diode*)
PLL	(*Phase Locked Loop*)

Sumário

1 Introdução

O objetivo principal deste relatório é apresentar, de forma sucinta, um guia documentando os procedimentos e subsistemas usados num sistema de filtro ativo, utilizando a plataforma de testes do LEIAM. No estudo de filtros ativos, toma-se como base os inversores, cuja ação de controle visa suprimir harmônicos de corrente ou tensão presentes no circuito onde se deseja instalá-los. São apresentadas algumas funções essenciais do Controlador Digital de Sinais (DSC, do Inglês, *Digital Signal Controller*) e da bancada de testes do LEIAM para aplicações em Eletrônica de Potência utilizando a placa de desenvolvimento DSC TMS320F28335.

O programa do Estágio Supervisionado contou com uma carga horária de 270 horas que foram integralizadas em 30 horas semanais, tendo seu início em 24 de novembro de 2015 e sendo finalizado em 27 de janeiro de 2016. O Estágio foi realizado na Universidade Federal de Campina Grande, no Laboratório de Eletrônica Industrial e Acionamento de Máquinas (LEIAM).

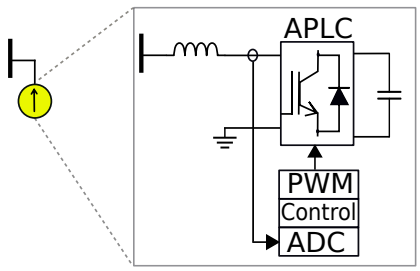
As atividades realizadas durante o programa da disciplina estágio como pré-requisito indispensável para obtenção do Grau de Bacharel em Engenharia Elétrica da Universidade Federal de Campina Grande, em síntese, foram: assimilação das funções básicas do Code Composer Studio (CCS) 3.3, conhecimento das configurações principais dos periféricos do DSC TMS320F28335 para aplicações em tempo real e dos componentes da plataforma de teste; montagem de um sistema teste do inversor controlado pelo DSC, criação de diagramas, esquemáticos e um roteiro para montagem de um inversor monofásico de ponte completa. As rotinas de inicialização do sistema, controle e interrupção foram desenvolvidas pela professora Dr. Camila Gehrke. A realização do teste do inversor foi supervisionada e auxiliada pelo aluno de doutorado Italo Pinheiro.

2 Filtro Ativo

Filtros ativos são compostos por conversores estáticos, cuja ação de controle visa suprimir harmônicos de corrente e/ou tensão presentes no circuito onde os mesmos estão instalados. Eles geralmente operam em malha fechada, auto regulando-se às condições dinâmicas observadas nas redes de distribuição. Os filtros ativos podem ser classificados em compensadores série, paralelo, híbrido e universal. O compensador utilizado na distribuição para minimização de harmônicos é denominado como compensador ativo de potência (APLC) e opera como filtro ativo paralelo atuando como fonte de corrente [Gehrke 2014].

Na Figura 1, é ilustrada a representação simplificada da implementação real de um APLC, representado como fonte de corrente controlada, onde a corrente medida é lida pelo ADC e as chaves do APLC são controladas pelo sinal PWM.

Figura 1 – Representação da estrutura simplificada de um APLC

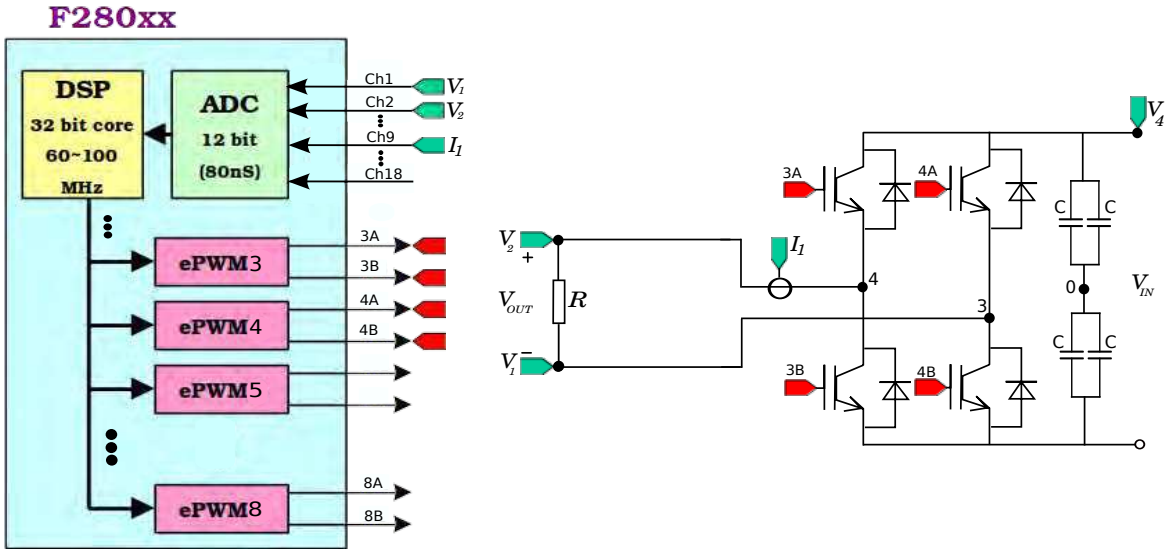


Fonte: [Gehrke 2014]

No que se refere aos filtros ativos, toma-se como base os conversores CC-CA(inversores) operando com PWM. A função dos inversores é fazer com que se produza uma corrente ou tensão que siga uma dada referência [Pomilio 2013]. A fim de entender melhor as placas e componentes da bancada de teste, um sistema de um inversor monofásico de ponte completa controlado pelo F23358 e alimentando uma carga linear resistiva foi montado.

Conforme a Figura 2, os valores de tensão e corrente são captados pelos sensores e convertidos pelo ADC no F28335 e as chaves do inversor são controladas através dos sinais PWM gerados pelo DSC. O sinail do sensor V1 está conectado ao "Ch1"do ADC, o V2 ao "Ch2", o V3 ao "Ch3"e o V4 ao "Ch4". O sinal do sensor I1 está conectado ao "Ch9", o I2 ao "Ch10", o I3 ao "Ch11", o I4 ao "Ch12", o I5 ao "Ch13"e o I6 ao "Ch14".

Figura 2 – Sistema inversor monofásico de ponte completa controlado pelo F23358



Fonte: [Texas Instruments 2008] adaptado pela autora

2.1 Descrição da bancada de testes

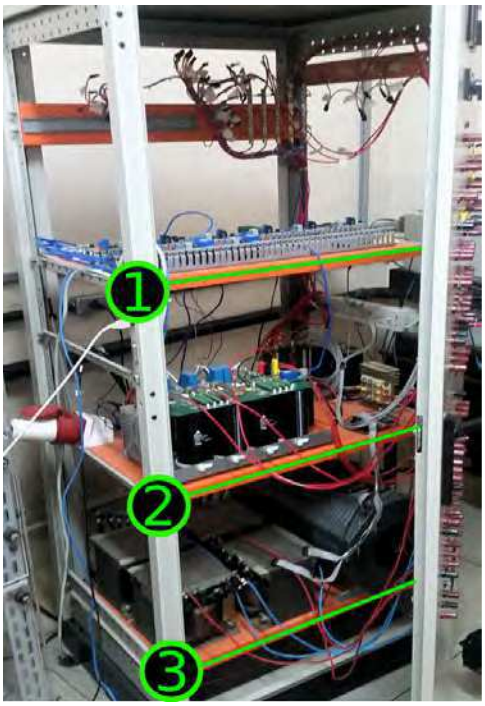
A bancada usada para testar o sistema da Figura 2, pode ser vista na Figura 3. No nível 1, encontram-se três sensores de tensão, seis sensores de corrente, condicionadores de sinais e o DSC. No nível 2, estão os drivers e o banco de capacitores do conversor, e resistores de potência. No nível 3, estão os indutores de acoplamento de valores ajustáveis(0.5mH, 1mH, 1.5mH, 2mH, 3mH, 4mH) e o varivolt que possui uma tensão máxima de saída de 380V.

Na Figura 4, os blocos em amarelo representam as placas que trabalham com sinais de tensão, em vermelho estão os que trabalham com os de corrente e os em azul com os sinais PWM.

A indicação da numeração destacada na Figura 4 segue abaixo:

- 1: Sensores de tensão;
- 2: placa que tem como entrada os cabos dos sinais V1, V2, V3 e V4 e têm como saída o cabo V1-V4;
- 3: condicionamento dos sinais V1-V4;
- 4: sensores de corrente;
- 5: placas que tem como entrada os cabos dos sinais I1, I2, I3, I4, I5 e I6 e possuem como saídas os cabos I1-I4 e I5-I6;
- 6: condicionamento dos sinais I1-I6;

Figura 3 – Fotografia da plataforma de teste



Fonte: Autora

– 7: condicionamento dos sinais PWM;

– 8: placa que separa os sinais ePWM3A, ePWM3B, ePWM4A, ePWM4B e possui como saída os cabos 3A, 3B e 4A, 4B que serão conectados aos drivers;

– 9: alimentação das placas;

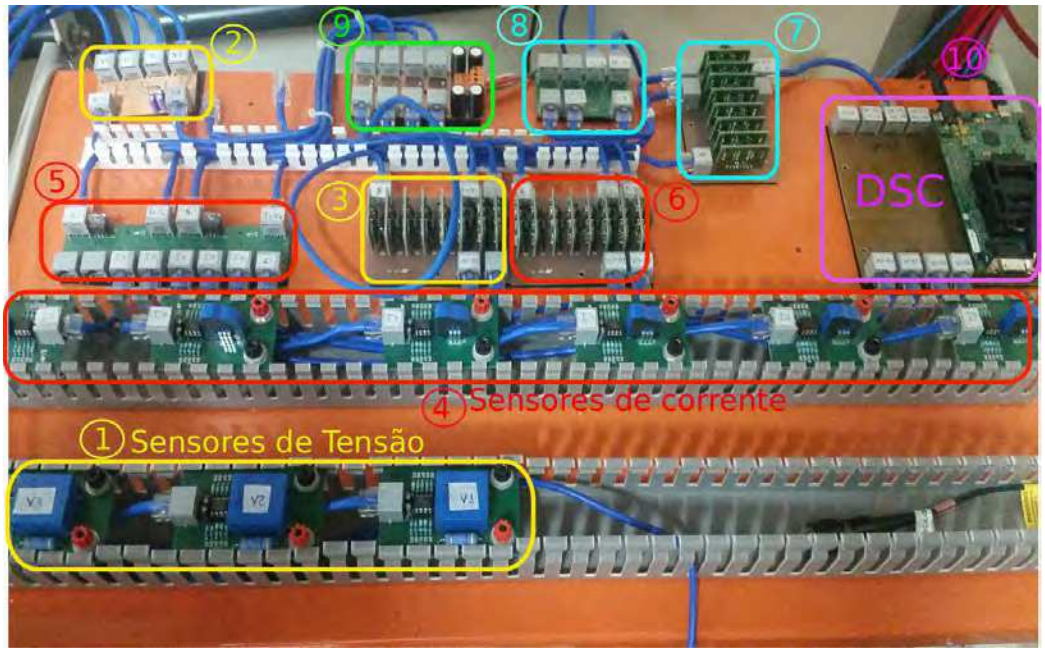
– 10: placa com o F28335 e os sinais do PWM e o ADC devidamente conectados

2.1.1 Conversor trifásico de energia e banco de capacitores

A bancada possui o Conversor SEMIKRON SKS 25 B6U + (B2CI) 2 10V 09, com 4 braços e cada par de interruptores possui um driver de acionamento também da SEMIKRON. A frequência máxima de chaveamento é de 15KHz e a tensão máxima do barramento é de 800V.

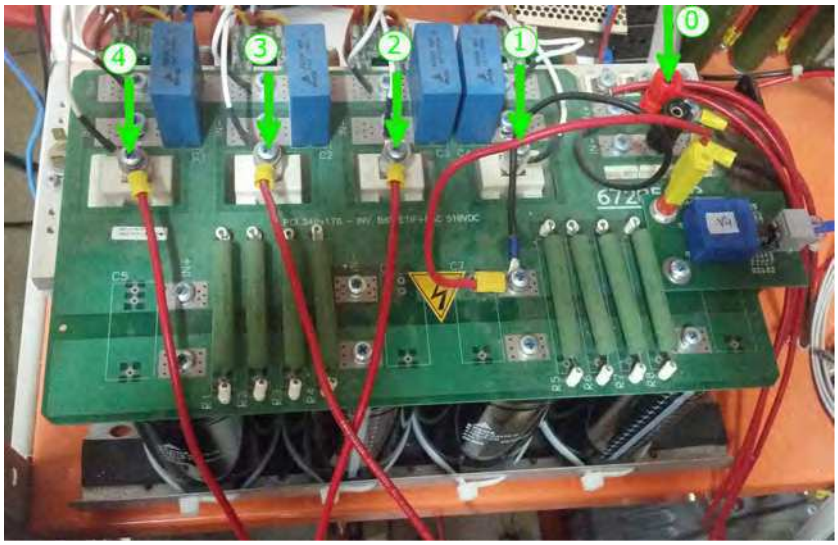
O conversor também possui um banco de capacitores que formarão o barramento CC. O ponto 0 marcado na Figura 5 representa o ponto 0 em comum da Figura 6. Os quatro capacitores da bancada são de C=4.7mF. Os pontos 1, 2, 3 e 4 são referentes aos quatro braços de chaves. Para um inversor monofásico, foram utilizados apenas os braços 3 e 4. Na Figura 6 é possível o esquemático do conversor, onde o mesmo possui um retificador trifásico não controlado ao qual foi ligado o varivolt em um dos braços. As entradas são representadas por uma fonte alternada de tensão em vermelho, e a saída está representada

Figura 4 – Fotografia dos componentes do primeiro nível da bancada



Fonte: Autora

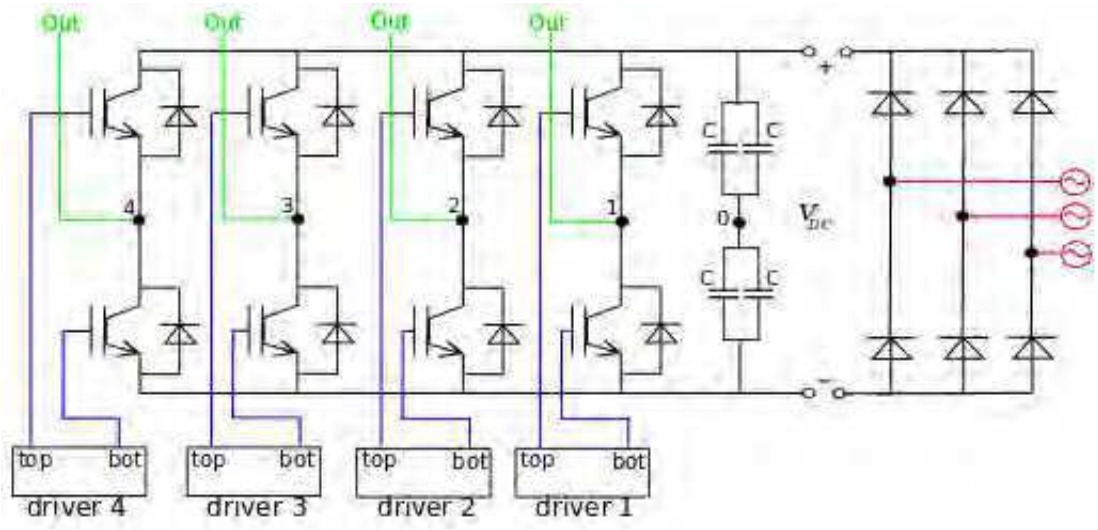
Figura 5 – Fotografia do Conversor da SEMIKRON



Fonte: Autora

em verde. Os drivers são conectados de tal forma que os pinos representados por *top* acionam as chaves superiores. Assim como os pinos *bot* acionam as chaves inferiores.

Figura 6 – Esquemático do conversor da SEMIKRON com quatro braços e retificador trifásico não controlado

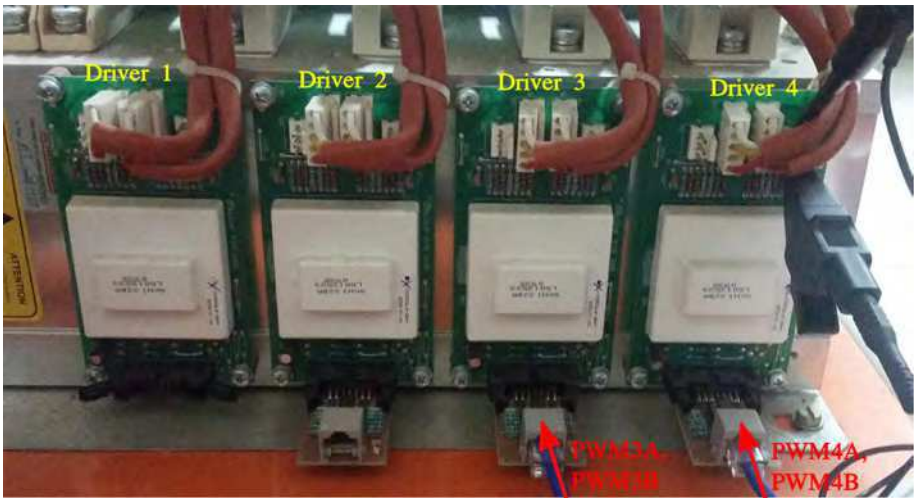


Fonte: Autora

2.1.2 Driver

O driver utilizado foi o SKHI22/2 da SEMIKRON. Cada driver possui uma chave superior e uma inferior que são controladas através dos sinais PWM conforme pode ser observado na Figura 7. Os drivers também possuem proteção de tempo morto(*Dead-Time*) e de alta corrente (*Over current*), como pode ser observado na Figura 8. No diagrama de blocos da Figura 8 é ilustrado as entradas do lado primário (*primary side*) do driver à esquerda, e as saídas, no lado secudário(*secondary side*) do driver à direita.

Figura 7 – Fotografia dos Drivers SKHI22/2 da SEMIKRON

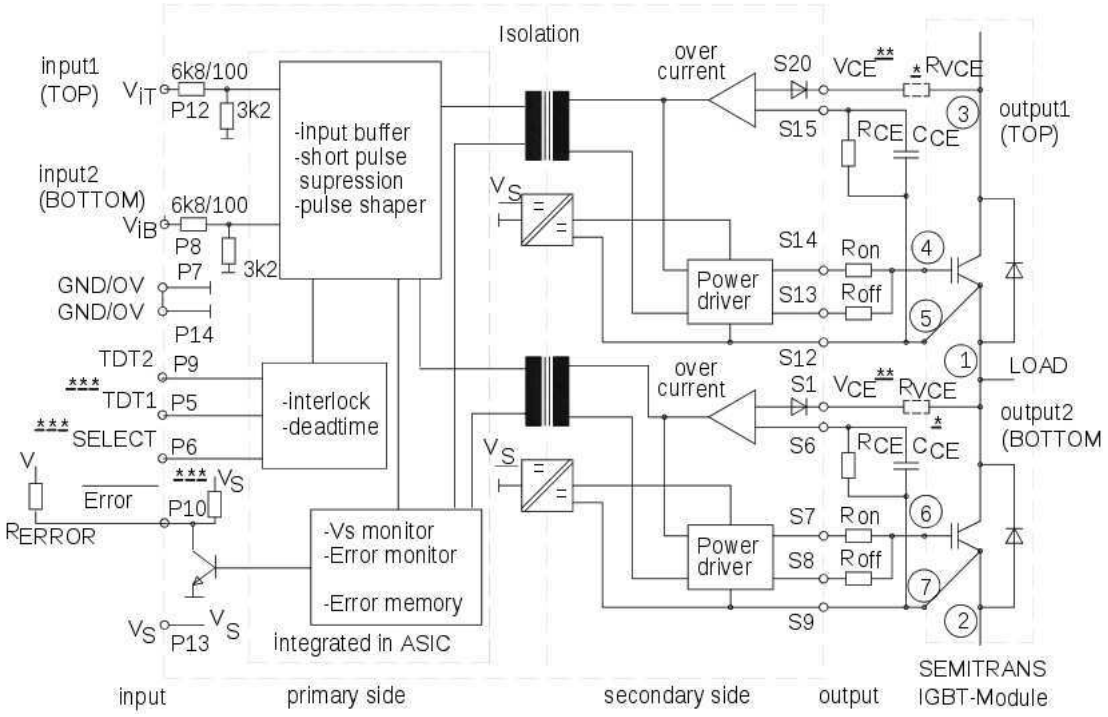


Fonte: Autora

Se um IGBT está ligado, o outro IGBT do mesmo braço não pode ser mudado. Além disso, um tempo de bloqueio digitalmente ajustável é gerado pelo driver, que tem que ser maior que o tempo necessário para desligar o IGBT. Isso é para evitar que um IGBT seja ligado antes do outro estar completamente desligado. Essa função pode ser neutralizada alterando a entrada *SELECT* (P6).

A memória de erro é definida em caso de baixa tensão ou curto-circuito dos IGBTs. A memória de erro irá bloquear todos os pulsos de comutação para os IGBTs e acionará a saída de erro (P10) do condutor. A saída do erro consiste em um transistor de coletor aberto, que dirige o sinal à terra em caso de erro. A memória de erro só pode ser resetada se não houver nenhum erro pendente, e ambas as entradas de sinal estiverem ajustadas para baixo ao mesmo tempo.

Figura 8 – Diagrama de blocos das funções, entradas e saídas do driver



Fonte: SEMIKRON

Na Figura 7 é possível observar a ponta de prova de um osciloscópio, usada para a medição do sinal de entrada de comutação da chave inferior do driver 4. No pino P8 encontra-se o sinal de entrada da chave inferior e o terra relativo ao sinal de entrada encontra-se no pino P7. Para a chave superior, o sinal de entrada e o terra relativo encontram-se nos pinos P12 e P14, respectivamente.

2.1.3 Sensor de corrente

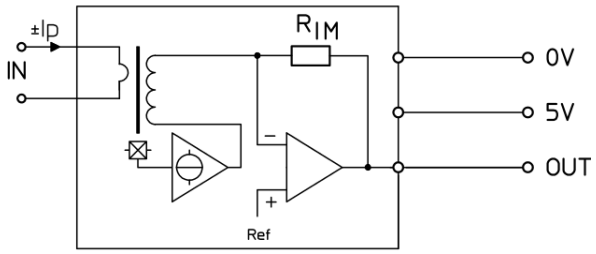
O LTS 25-NP da LEM é um transdutor de corrente que possui saída instantânea, excelente precisão, linearidade boa, variação de temperatura baixa, tempo de resposta otimizada, sem perdas de inserção e alta imunidade a interferências externas. Ele é utilizado para a medição eletrônica de correntes (DC, AC, pulsada) de até ±25A com separação galvânica entre o circuito primário e o circuito secundário. [LEM 2012]

O sensor de corrente utilizado é ilustrado na Figura 9. Na Figura 10, é ilustrada o esquema para a conexão dos pinos do transdutor de corrente da LEM. A corrente a ser medida deve ser conectada a entrada "IN", os pinos para alimentação estão representados por 0V e 5V e a saída por "OUT".

Figura 9 – Fotografia do sensor de corrente

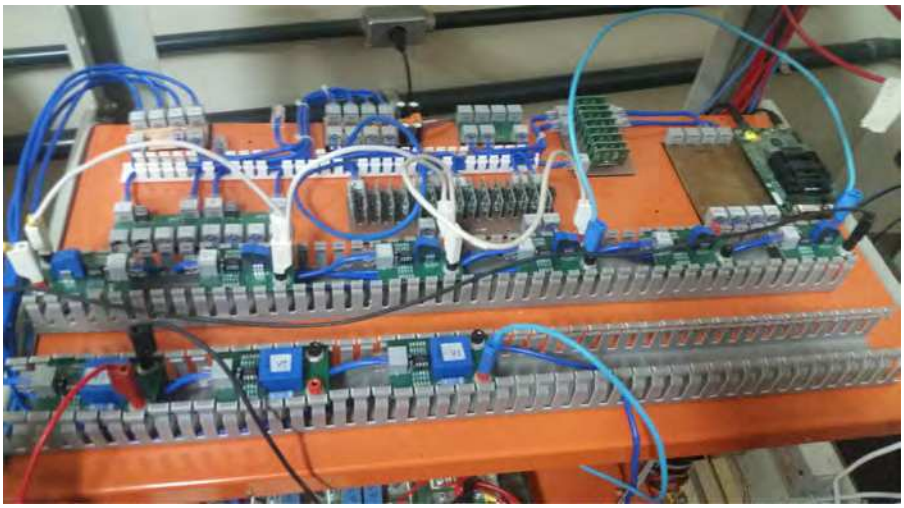
Fonte: Autora

Figura 10 – Circuito de conexão básico do LTS 25-NP



Fonte: [LEM 2012]

Figura 11 – Fotografia dos sensores conectados em série para calibração dos sensores de corrente



Fonte: Autora

Para entender os valores lidos pelo ADC, é preciso calibrar os sensores. Para calibrar os sensores de corrente, foi usada uma fonte de corrente ligada em série com os seis sensores,

conforme pode ser observado na Figura 11. O valor da corrente real DC foi comparado com o valor obtido pelo ADC e, após obter 10 pontos, para uma melhor precisão, a equação de reta para cada sensor foi incorporada ao código do ADC da seguinte forma:

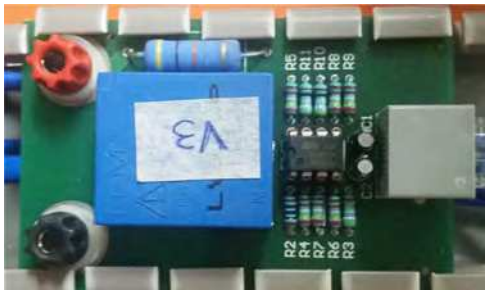
```
i1 = 0.0176*(AdcRegs.ADCRESULT8  >> 4)-38.5116;
i2 = AdcRegs.ADCRESULT9  >> 4;
i3 = 0.0175*(AdcRegs.ADCRESULT10 >> 4)-37.8571;
i4 = 0.0177*(AdcRegs.ADCRESULT11 >> 4)-38.1331;
i5 = 0.0177*(AdcRegs.ADCRESULT12 >> 4)-38.2890;
i6 = 0.0178*(AdcRegs.ADCRESULT13 >> 4)-38.5464;
```

2.1.4 Sensor de tensão

Para a medição da tensão foi usado o transdutor de tensão LV 25-P baseado no efeito de *Hall*. Ele possui tempo de resposta baixo, excelente precisão, linearidade boa, largura de banda alta e fornece isolação galvânica entre os circuitos primário e secundário. Para medições de tensão, uma corrente proporcional à tensão de medição deve passar através de uma resistência externa, selecionada pelo usuário, e instalada em série com o circuito primário do transdutor. Adequado para a medição eletrônica de tensões associada a DC, AC e circuitos de impulso de até 500V. [LEM 2012]

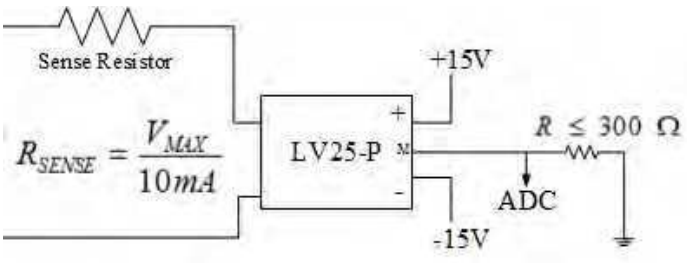
O sensor de tensão utilizado é ilustrado na Figura 12. Na Figura 13, é ilustrada o esquema para a conexão dos pinos do transdutor de tensão da LEM. Um resistor deve ser conectado de acordo com a fórmula $R_{sense} = V_{max}/10mA$ para ajustar a sensibilidade do sensor. A corrente a ser medida deve ser conectada a entrada onde encontra-se o resistor de sensibilidade, os pinos para alimentação estão representados por +15V e -15V e a saída por "ADC".

Figura 12 – Fotografia do sensor de tensão



Fonte: Autora

Figura 13 – Circuito de conexão básico do LV 25-P



Fonte: [LEM 2012]

Para calibrar os sensores de tensão, foi usado um varivolt ligado em paralelo com os quatro sensores, conforme pode ser observado na Figura 14. O valor DC do varivolt foi medido com um multimetro e comparado com o valor obtido pelo ADC. Foram obtidos

Figura 14 – Fotografia dos sensores conectados em paralelo para a calibração dos sensores de tensão

Fonte: Autora

10 pontos para uma melhor precisão. Por fim, a equação de reta para cada sensor foi incorporada ao código do ADC como segue abaixo:

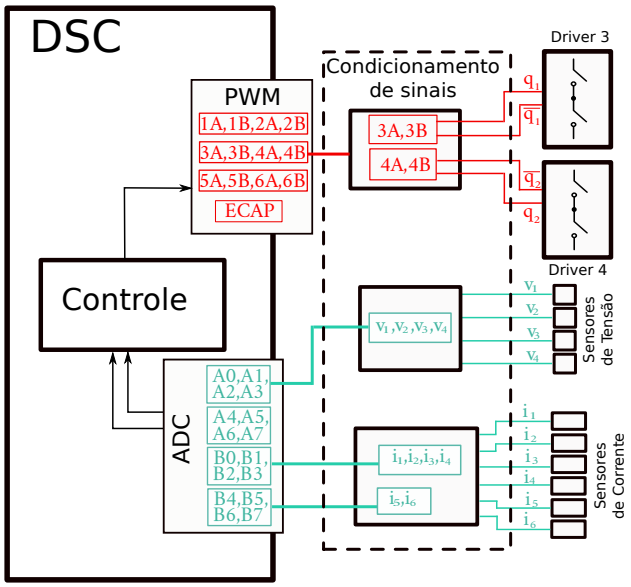
```
v1  = 0.2818*(AdcRegs.ADCRESULT0 >> 4)-607.633;
v2  = 0.2787*(AdcRegs.ADCRESULT1 >> 4)-595.5391;
v3  = 0.2754*(AdcRegs.ADCRESULT2 >> 4)-586.6239;
v4  = 0.2780*(AdcRegs.ADCRESULT3 >> 4)-589.8296;
```

2.2 Esquema de conexões

Para um inversor monofásico, o esquema de conexões pode ser visto a seguir na Figura 15. A placa do DSC está ligada a oito conectores RJ-45. Os 16 canais do conversor analógico digital utilizam 4 conectores e os sinais PWM utilizam os demais. Os conectores estão conectados a quatro sinais agrupados conforme a Figura 15.

Os sinais de tensão e corrente obtidos pelos sensores passam por um condicionamento

Figura 15 – Esquema das conexões do F28335 com componentes da bancada de teste



Fonte: Autora

e são conectados aos canais do ADC por meio de cabos UTP. Aqui, cada cabo UTP possui quatro pares de fios entrelaçados onde quatro deles transmitem os sinais e os demais são aterrados. Ou seja, para os sinais de tensão, entre o condicionador de sinal e o DSC só foi necessário um cabo. O sinal V1 está conectado ao pino A0, o V2 ao pino A1 e assim sucessivamente. Os sinais de corrente estão conectados à porta B. Do mesmo modo, o sinal de corrente I1 está conectado ao pino B0, o I2 ao pino B1, o I3 ao pino B2, o I4 ao pino B3, o I5 ao pino B4 e o I6 ao pino B5.

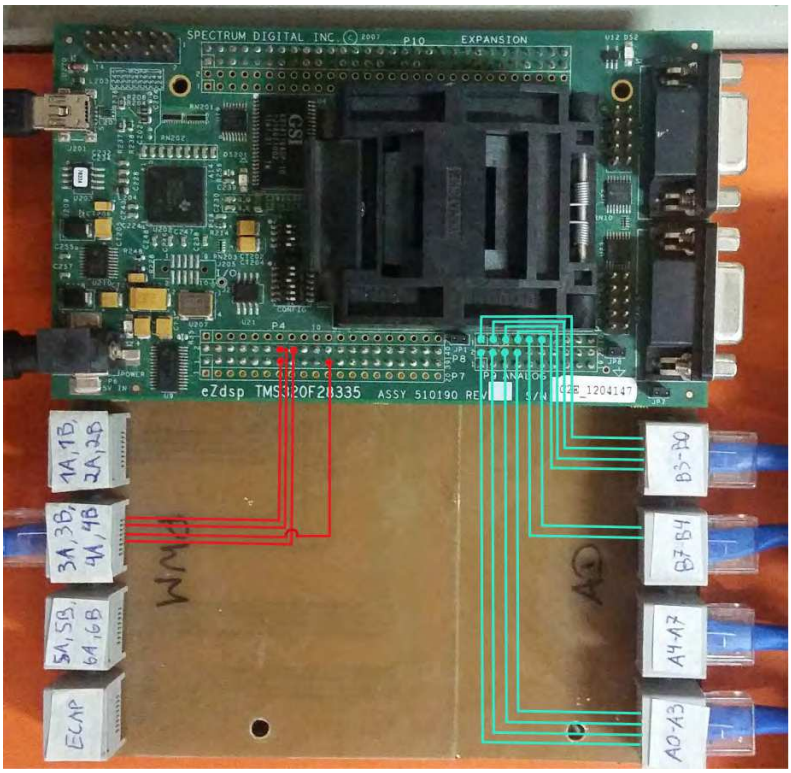
Os sinais ePWM3A, ePWM3B, ePWM4A, ePWM4B são separados em dois cabos, onde cada cabo transmite dois sinais PWM para os drivers. Ou seja, o cabo "3A,3B"está ligado ao "driver 3", onde o ePWM3A controla a chave superior e o ePWM3B é o seu complementar e controla a chave inferior.

Na Figura 16, estão destacados os pinos usados do F28335. Para um inversor monofásico, foi utilizado apenas quatro sinais PWM.

2.3 Etapas de preparação para o uso da bancada

Primeiramente deve-se conectar aos pontos 3 e 4 da Figura 5 a carga. Para esse teste foi usado um resistor de 40 ohms. Nos terminais da carga será feita a medição dos sinais de tensão. Logo, será necessário conectar os sensores de tensão nos terminais da carga, como visto na Figura 2. Outra alternativa para observar o sinal de saída seria usar um osciloscópio.

Figura 16 – Pinos do PWM e do ADC usados do F28335



Fonte: Autora

A tensão de barramento Vin é gerada pelo varivolt, o sensor de tensão V4 encontra-se conectado ao banco de capacitores e pode ser usado para capturar os valores de Vin. Os sensores de corrente devem ser conectados a critério do usuário.

Após realizadas as conexões, deve-se ligar a fonte de alimentação do DSC, a fonte chaveada de ±15V que é distribuída aos sensores de tensão e corrente. Recomenda-se deixar o varivolt em zero volts antes de ligá-lo, de forma a evitar uma variação de tensão muito grande. Por fim, liga-se o varivolt, variando-o para a tensão desejada.

No CCS, após abrir o projeto, deve-se conectar o *target*, construir e carregar o programa. Para observar os dados capturados pelo ADC em tempo real, deve-se acessar o menu *Debug -> Real-time Mode* e também adicionar as variáveis que se deseja observar na janela *Watch*. Finalmente para rodar o programa, deve-se clicar em *Run*.

É importante saber que alguns componentes não foram utilizados pois não funcionaram corretamente, são eles: os pinos PWM1A,PWM1B,PWM2A,PWM2B; o varivolt da plataforma de teste; o sensor de corrente I2 e os drivers 1 e 2.

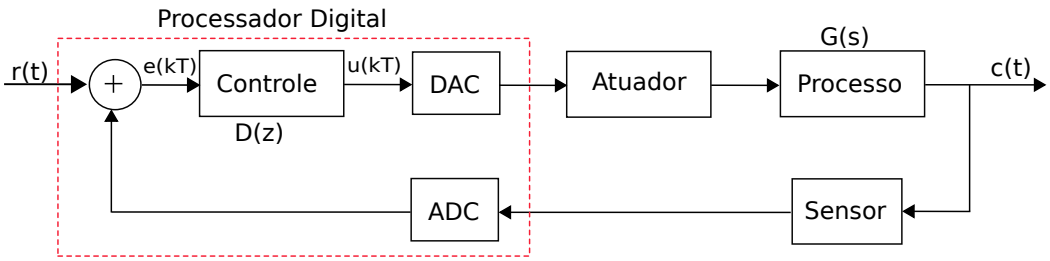
3 Controlador Digital de Sinais

Nesse trabalho foi utilizado o módulo de desenvolvimento eZdsp starter kit para o Controlador Digital de Sinais (DSC) TMS320F28335, fabricada pela Spectrum Digital.

DSC são dispositivos muito versáteis e poderosos, eles possuem as principais características dos microcontroladores (MCU) mais importantes funções de Processadores Digitais de Sinais (DSP). Os microcontroladores são geralmente utilizados em aplicações embarcadas, onde se deseja monitorar, controlar e atuar num sistema autônomo. Para isso, eles possuem unidade de processamento, unidade de memória e unidade de acesso para periféricos. Já os processadores digitais de sinais são utilizados quando os sinais precisam ser convertidos de analógico para digital, para então serem manipulados digitalmente e finalmente serem convertidos em sinais analógicos novamente. Eles possuem uma arquitetura especializada para reduzir o número de instruções e operações necessárias de um algoritmo de processamento de sinais.

O TMS320F28335 pode ser utilizado na indústria para diferentes soluções de monitoramento e de controle em tempo real. Como no controle de motores, no controle e monitoramento de conversores CA/CC e inversores utilizados em energias renováveis (solar, eólica), etc. Na Figura 17 é possível observar um diagrama de blocos do controlador digital em um processo genérico G(s).

Figura 17 – Sistema de Controle Digital



Fonte: Autora

Os controladores digitais possuem diversas vantagens, entre elas, a de serem menos sensíveis a efeitos externos do ambiente, possibilitarem estratégias de controle avançadas, por exemplo, não-linear e multi-variável, poderem executar vários loops e multiplas funções e serem capazes de modificar e armazenar os parâmetros de controle.

O TMS320F28335 possui uma alta capacidade de processamento além de rápida resposta a interrupções com uma CPU de 32 Bits, 150MHz, três temporizadores de 32 bits, 18 saídas PWM, 16 canais ADC de 12 bits com taxa de conversão de 80 ns. Ele é capaz

de ler e transferir dois operandos da memória para a central de processamento em um único ciclo de clock. É possível acessar tanto as memórias internas quanto uma memória flash externa. Também permite aos usuários desenvolverem o seu software de controle do sistema em uma linguagem de alto nível, sendo eficiente para códigos em C/C++ e Assembly.

3.1 A Placa eZdsp F28335

O kit F28335 eZdsp é uma plataforma de desenvolvimento de software completo para a série TMS320F2833x de controladores digitais de sinais de ponto flutuante. Ele fornece as ferramentas de hardware e software necessárias para o desenvolvimento inicial em diversas aplicações. O kit eZdsp inclui uma placa para emulação JTAG, interface USB ao PC, 128Kx16 de SRAM assíncrona integrada, interfaces CAN 2.0 e RS-232. Também está incluído no kit o CD contendo o ambiente de desenvolvimento Code Composer e os drivers necessários, a fonte de alimentação universal +5V e o cabo USB, como pode ser visto na Figura 18.

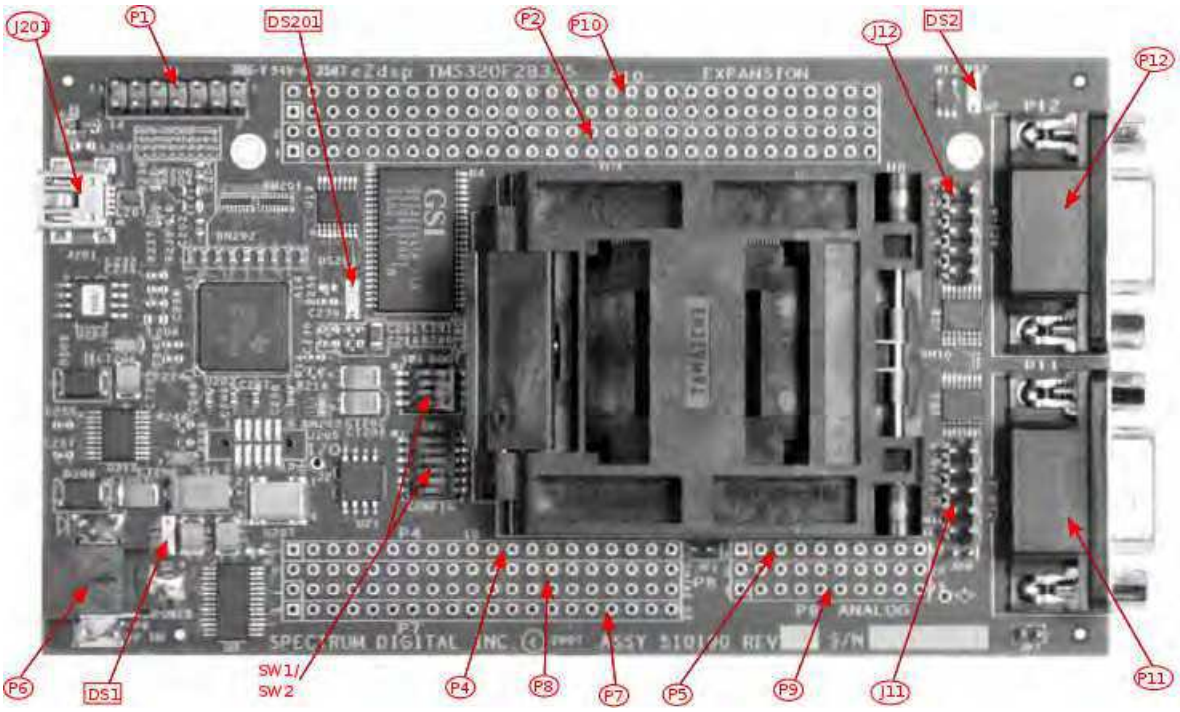
Figura 18 – eZdsp starter kit

Fonte: [Vieira 2013]

A F28335 tem quatorze conectores, conforme Figura 19. A função de cada conector é mostrada na Tabela 1.

É possível observar na Figura 19 que a placa tem três diodos emissores de luz verde. DS1 indica a presença de +5 volts e está normalmente acesa quando a energia é aplicada à placa.O LED DS2 está sob o controle do GPIO32. DS201 mostra o estado da ligação

Figura 19 – Placa eZdsp F28335



Fonte: [Spectrum Digital, Inc 2007]

Tabela 1 – F28335 Connectors

Conector	**Função**
P1	JTAG Interface
P2	Expansion
P4/P8/P7	I/O Interface
P5/P9	Analog Interface
P6	Power Connector
P10	Expansion
P11	CAN-A
P12	SCI-A
J11	CAN-B
J12	SCI-B
J201	USB Connector

Fonte: eZdsp F28335 Technical Reference

de emulação USB. Estas são mostradas na tabela abaixo. A placa possui também dois *switches*. SW1 é usado para selecionar a opção de carregamento na inicialização. SW2 é utilizado para selecionar a configuração do processador.

4 Ambiente de Desenvolvimento

O Code Composer Studio (CCS) é um ambiente de desenvolvimento integrado para a família C2000 de processadores digitais de sinais da Texas Instruments para o desenvolvimento de aplicações no F28335. O presente capítulo fornece os principais procedimentos para a instalação do CCS versão 3.3, criação, configuração e simulação de um projeto utilizando o F28335.

4.1 Instalação do CCS 3.3

O CD para instalação do *software* CCS v3.3 pode ser encontrado no starter kit eZdsp. Siga os procedimentos presentes na caixa do produto e instale o Code Composer e os drivers requeridos. É importante salientar que o CCS v3.3 foi instalado e funcionou corretamente somente no sistema operacional Windows XP.

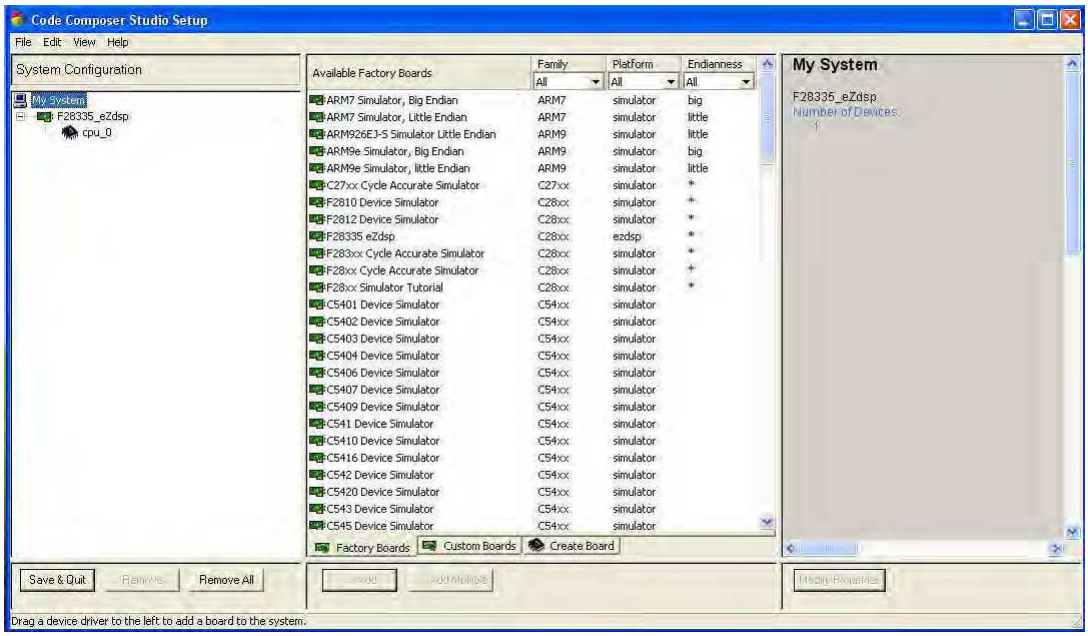
Depois de concluída a instalação, será criado um ícone na área de trabalho designado por Code Composer SETUP. Deve-se selecionar e executar o programa para designar o *"Target"*. Uma janela para a configuração do sistema será aberta, conforme é possível observar na Figura 20. Selecionar na lista *"Available Factory Board"* a opção compatível com o sistema utilizado. A lista de placas(*"Board"*) disponíveis mudará dependendo da instalação. Em *"Family"*, deve-se selecionar C28xx. Em *"Platform"* deve-se selecionar *"F28335 XDS100 USB Emulator"*. E necessário clicar no botão *"Add"* para importar a seleção para *"System Configuration"*. A configuração selecionada aparecerá abaixo do ícone *"My system"*. Em seguida, deve-se conectar na entrada USB do PC na placa de desenvolvimento do DSC. Após clicar no botão *"Save and Quit"* e deve-se clicar em *"Yes"* para abrir o CCS com a configuração selecionada anteriormente.

4.2 Criação de um Novo Projeto

a) Após executar o CCS, deve-se selecionar a aba *"Project"* e em seguida clicar na opção *"New"*. Uma nova janela será exibida, como na Figura 21. Deve-se configurar da seguinte forma:

   - Crar um nome para o projeto em *"Project Name"*;
   - Escolher a pasta onde os arquivos do projeto serão salvos em *"Location"*;
   - Selecionar o tipo de projeto, que no caso será um arquivo executável(.out), em *"Project Type"*;

Figura 20 – Code Composer Studio Setup



– Por fim, selecionar o tipo de DSC que será utilizado, que para a placa de desenvolvimento deve ser o TMS320C28XX, em *"Target"*.

Figura 21 – Janela Project Creation

b) Para criar um novo código em C ou C++, é necessário acessar o menu *FileNew -> Source -> File* e salvar o arquivo com o nome desejado;

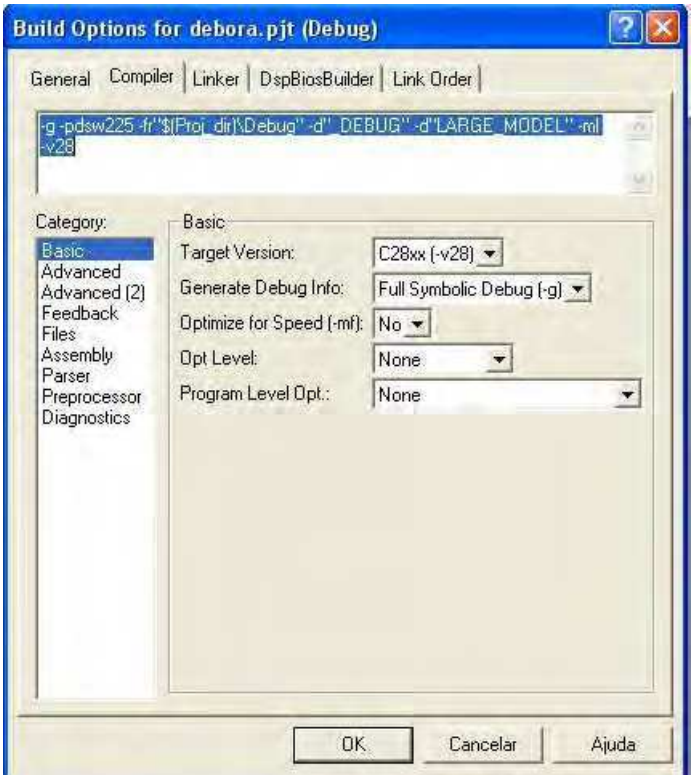
c) Deve-se adicionar arquivos ao projeto através do menu *Project -> Add Files to Project*;

– Em seguida, adicionar os arquivos com extensão .c ou .cpp desejados;

– Para compilar o programa, é necessário incluir algumas bibliotecas. Para um código experimental simples, seŕá necessário adicionar apenas o arquivo *rts2800_ml.lib*;

– Para a transferência correta das informações para o hardware, é necessário adicionar um arquivo pré-definido pela Texas Instruments chamado *28335_RAM_lnk.cmd*.

Figura 22 – Janela Build Options - Compiler



d) É necessaŕio acessar o menu *Project -> Build Options*, conforme a Figura 22 na aba *"Compiler"*, selecionar *"Preprocessor"* e em *"Include Search Path(-i)"*, colar o caminho onde os arquivos da pasta *include* estão disponíveis. Se os arquivos estiverem no diretório *Downloads*, o caminho será parecido com: $user\backslash Downloads\backslash DSP288x - headers\backslash include$.

e) Conforme Figura 23, ainda no menu *"Build Options"*, na aba *"Linker"* deve-se preencher o campo *"Stack Size"* com o valor de 0x400, para designar o tamanho da pilha de dados.

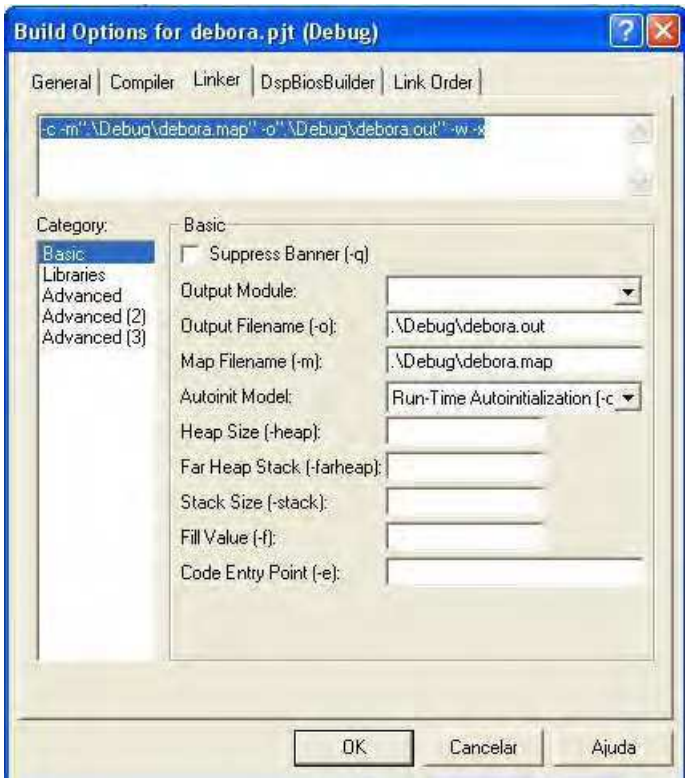
4.3 Construir e carregar o programa

a) Para construir o programa, é necessário acessar o menu *Project -> Build*;

b) Após o programa ter sido construído com sucesso, conectar o target acessando o menu *Debug -> Connect*;

c) Transferir o código para a placa acessando o menu *File -> Load Program*. O Code Composer criará uma pasta dentro do diretório do projeto chamada *Debug*,

Figura 23 – Janela Build Options - Linker



como pode ser visto na Figura 24. Após selecionar o arquivo executável criado pelo comando *Build*, que possui a terminação **.out*, deve-se clicar em *"Open"* para finalmente carregar o programa no *Hardware*.

Figura 24 – Janela Load Program

É importante lembrar que após qualquer mudança no código, será necessário reconstruir(*Rebuild*) e recarregar (*Reload*) o programa.

5 Principais Configurações

Este capítulo aborda de forma simples a configuração e uso dos periféricos de GPIO, do *clock*,do sistema de interrupção no F28335, do PWM e do ADC. Ao longo do capítulo, serão apresentados alguns exemplos utilizando as unidades de entrada e saída, interrupcão, inicializando o PWM e o ADC para facilitar e reduzir o tempo de aprendizado. Ao final do capítulo, dois exemplos desenvolvidos por [Vieira 2013] serão comentados.

5.1 Unidade I/O

No F28335, as entradas e saídas digitais estão agrupadas em três portas digitais: A,B e C. Ele possui 88 pinos I/O multiplexados chamados de GPIO0-GPIO87,que podem ser configurados para uma determinada função de acordo com a necessidade do usuário. Normalmente, cada pino possui mais de uma função, isto é, o pino fornece acesso a diversos periféricos do microcontrolador. O acesso a estas funções se dá através da multiplexação no pino. A porta A é composta pelos pinos GPIO0 até GPIO31, A porta B do GPIO32 até o GPIO63 e a porta C do GPIO64 até GPIO87.

Várias funções são atribuídas aos pinos GPIO. Eles podem ser configurados para entradas e saídas periféricas ePWM, SCI, SPI, eCAN, eCAP, iniciar a conversão do ADC, acordar o dispositivo dos modos de baixa energia HALT e STANDBY, entre outras.

O registrador GPxDIR define a direção dos dados. Se o valor for igual a 1, o pino GPIO será uma saída. Se o valor for igual 0, o pino será uma entrada. O registrador GPxCLEAR é uma máscara que força o pino GPIO para o nível lógico 0 e o GPxSET força o pino GPIO para o nível lógico 1.

Os registrador de dados GPxDAT reflete o estado atual do pino. Escrever para os registradores GPxDAT possibilita limpar ou setar a saída correspondente e, se o pino está habilitado como uma saída, o pino também será levado para o nível lógico 0 ou 1.

Abaixo segue algumas formas de como os registradores de dados GPIO podem ser acessados:

– GpioDataRegs.GPASET.all: seta todos os pinos da porta A.

– GpioDataRegs.GPBDAT.bit.GPIO32 = 1: configura o registrador de dados do LED DS2 para o nível lógico 1;

– GpioDataRegs.GPBSET.bit.GPIO32 = 1: configura o registrador de dados do LED DS2 para o nível lógico 1;

– GpioDataRegs.GPBDAT.bit.GPIO32 = 0: configura o registrador de dados do LED DS2 para o nível lógico 0;

– GpioDataRegs.GPBCLEAR.bit.GPIO32 = 1: configura o registrador de dados do LED DS2 para o nível lógico 0;

– GpioDataRegs.GPBTOGGLE.bit.GPIO32 = 1: Inverte o nível lógico do LED DS2;

Abaixo segue uma função demonstrando a configuração de um pino GPIO como entrada ou saída:

```
void seleciona_GPIO(void)
{
     EALLOW;
// GPIO15-GPIO0 = General Purpose I/O
     GpioCtrlRegs.GPAMUX1.all = 0:

// GPIO31-GPIO16 = General Purpose I/O
     GpioCtrlRegs.GPAMUX2.all = 0;

// Configura todos os pinos da porta A(GPIO0-GPIO31) como entradas
     GpioCtrlRegs.GPADIR.all = 0;

// Configura o pino GPIO9 como saída
     GpioCtrlRegs.GPADIR.bit.GPIO9 = 1;

// Pino configurado para ePWM1A
     GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 1;
EDIS;
}
```

A Macro em C “EALLOW” destrava o acesso para alguns registradores especiais, e a “EDIS” desabilita esse efeito. Lembrando que em C++, deve-se escrever da seguinte forma: “EALLOW()” e “EDIS()”.

5.2 Clock

O oscilador da placa e PLL proporcionam os sinais de clock para o dispositivo. A configuração do clock principal(SYSCLKOUT) é uma tarefa importante, pois as atividades temporizadas das unidades periféricas são dimensionadas de acordo com o valor do clock do sistema para funcionar corretamente. O circuito PLL possibilita a escolha de diferentes clocks de acordo com a frequência específica exigida para cada aplicação.

O registrador PCLKCRx habilita/desabilita o clock para as unidades periféricas.

Abaixo, segue uma função demonstrando a configuração do clock interno (SYSCLK) para 150MHz, supondo um oscilador de 30 MHz.

```
void Inicializa_sistema(void)
{
     EALLOW;
// Configura os bits do clock
     SysCtrlRegs.PLLSTS.bit.DIVSEL = 2;

// 30MHz * 10 / 2 = 150 MHz SYSCLK
     SysCtrlRegs.PLLCR.bit.DIV = 10;

// SYSCLK/2 Pré-escalona o clock para os dispositivos periféricos.
     SysCtrlRegs.HISPCP.all = 0x000

// SYSCLK / 4
     SysCtrlRegs.LOSPCP.all = 0x0002;
     SysCtrlRegs.PCLKCR3.all = 0x0000;

// Habilita o clock para unidade periférica
     SysCtrlRegs.PCLKCR3.bit.GPIOINENCLK=1;

EDIS;
}
```

5.3 Interrupção

Uma interrupção é um evento que causa o processador parar a execução do programa e desviar para um bloco de código chamado rotina de interrupção. Apoś terminar o tratamento da interrupção o controle retorna ao programa interrompido, ou seja, para o endereço de memória imediatamente posterior ao que estava quando ocorreu a interrupção.

O F28335 possui 16 canais de interrupção, onde em 14 canais o programador pode habilitar ou desabilitar e os outros dois são o RESET e NMI. Os dezesseis canais estão conectados a uma tabela de vetores de interrupção que são endereços de memória de 32bits. É de responsabilidade do programador preencher esta tabela com o endereço inicial de cada rotina de interrupção utilizada. A Texas endereça estes canais na memória ROM do F28335 da seguinte forma: o RESET (RS) aponta

para o endereço 0x00 0040, o NMI aponta para o endereço 0x00 0042 e assim de forma consecutiva. Todos estes endereços estão na RAM, logo o programador deve preencher uma instrução de 32 bits nestes endereços de memória.

Como o F28335 tem 96 fontes de interrupção e apenas 16 canais para entradas de interrupção, a Texas utiliza o chamado PIE(Peripheral Interrupt Expansion) que remapeia a memória criando um efeito de multiplexação das fontes de interrupção. [Vieira 2013]

5.4 Timers

O F28335 possui três temporizadores independentes de 32 bits. O timer 1 e o timer 2 são geralmente utilizados para operações em tempo real, e o timer 0 é utilizado para uso geral.

Uma vez habilitado (TCR-bit $4 = 0$),o clock passa por um pré-escalonador (pre-scaler) de 16 bits (PSCH: PSC) para reduzir a frequência de trabalho. Caso ocorra um underflow ele faz com que o contador de 32 bits(TIMH: TIM) comece a contar em contagem regressiva. Quando este também sofrer um underflow ele irá gerar uma interrupção para a CPU. [Vieira 2013]

5.5 PWM

O F28335 possui 6 canais capazes de gerar sinais PWM. Esta sessão apresenta os principais registradores necessários para configurar e utilizar sinais PWM do DSC. O clock base dos ePWM é o SYSCLKOUT, mesmo do sistema. O ePWM do F28335 é capaz de iniciar uma conversão analógica-digital sem interação de software. O periférico PWM, além de suportar a geração de PWM independente e complementar, é capaz de operar nos modos de contagem crescente, contagem decrescente e contagem crescente-decrescente. Ele também possui pinos que suportam funcionalidades de alta resolução (HRPWM).

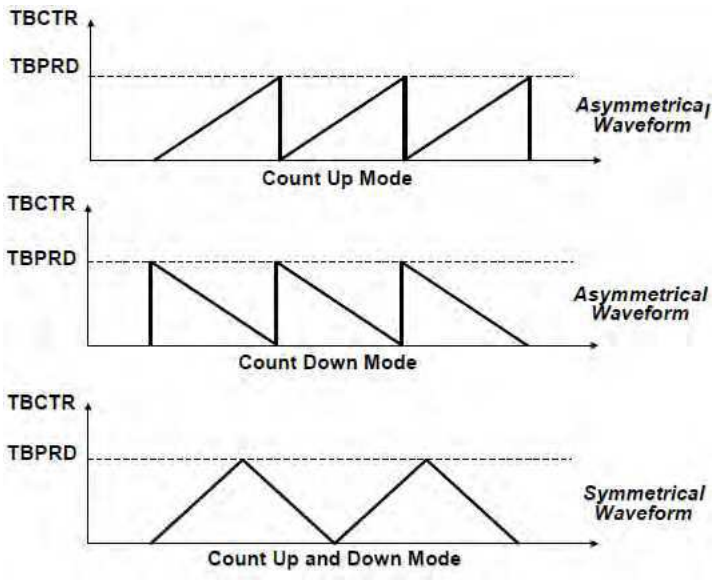
Na Figura 25, é possível observar os três modos de operação dos contadores do PWM. Eles são de contagem crescente, contagem decrescente e contagem crescente-decrescente.

Os registradores CMPA e CMPB pertencem à unidade comparadora do F28335 e são utilizados na determinação da frequência de trabalho. Dependendo da configuração, a saída do ePWM ao ser comparada com os pontos CA e CB podem ir para 1, para zero, mudar de valor lógico ou não fazer nada. Na figura Figura 26, o sinal EPWMA ao ser comparado com o ponto CA é setado para valores maiores que CA e vai para zero para valores menores que CA. Já o sinal EPWMB possui a mesma frequência do EPWMA, porém o seu ciclo de trabalho é menor.

Os principais registradores que precisam ser configurados são:

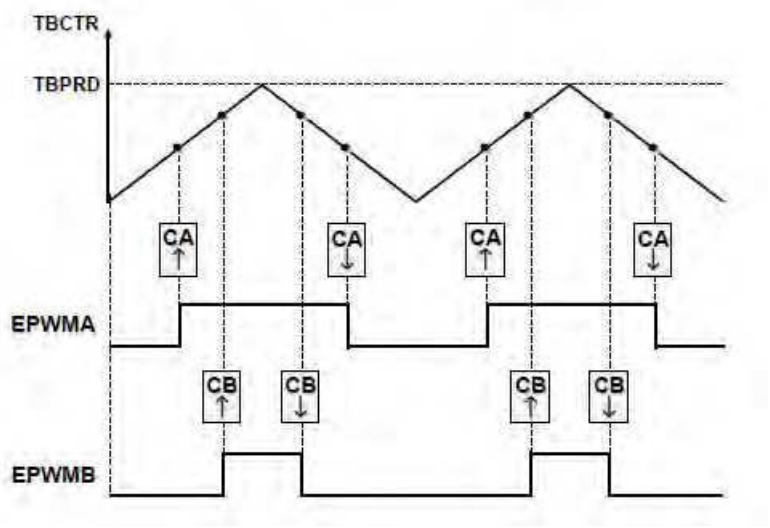
– TBCTL: registrador de 16 bits responsável pelas configurações de controle. É utilizado para escalonar a frequência de entrada entre 1 e 1792. Especifica se começa contando

Figura 25 – Modos de operação dos contadores ePWM



Fonte: TI C28xdps Guide modificado

Figura 26 – Saída ePWM para uma contagem Up-Down



Fonte: TI C28xdps Guide modificado

pra cima ou para baixo do pulso de clock, modo de operação do contador, entre outras coisas.

– TBPRD: define o tamanho do período do sinal de saída, em múltiplos do período do sinal de entrada. De acordo com a fórmula TBPRD=Tpwm/(2*Tsysclout*CLKDIV*HSPCLKDIV)
– CMPA ou CMPB: Define a largura do pulso do ePWM1A ou ePWM1B
– AQCTLA: Configura o formato do sinal de saída do ePWM1A

A função abaixo exemplifica uma inicialização do EPWM1A e EPWM1B:

```
void Init_PWM()
{

// Configura o periodo do contador PWM. TBPRD=900 TBCLK counts
   EPwm1Regs.TBPRD = 3750;
   EPwm1Regs.TBPHS.half.TBPHS = 0x0000;   // fase 0
   EPwm1Regs.TBCTL.bit.CTRMODE = 2; // Modo up-down

// Desabilita sicronização
   EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;

   EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
   EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
   EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
   EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // carrega em CTR=Zero
   EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // carrega em CTR=Zero


// Leva a saída ePWM1A para um na contagem crescente
   EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
// Leva a saída ePWM1A para zero na contagem decrescente
   EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
// Leva a saída ePWM1B para zero na contagem crescente
   EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;
// Leva a saída ePWM1B para um na contagem decrescente
   EPwm1Regs.AQCTLB.bit.CBD = AQ_SET;

 // Seta o DSC para começar com 100% de ciclo de trabalho.
   EPwm1Regs.CMPA.half.CMPA = 0;
}
```

5.6 Conversor Analógico Digital

O bloco ADC é um conversor de 12-bit e 16 canais. Ele contém duas unidades de sample-and-hold para amostragem simultânea. Ele tem uma taxa de conversão rápida de 80ns. A Equação 5.1 representa a relação entre o número de bits e o número que será utilizado na representação digital do sinal analógico convertido. Onde Vin corresponde ao

valor lido na entrada do conversor, Vref+ e Vref- são as tensões utilizadas para limitar os sinais analógicos, n é o número de bits utilizados para representar o sinal digital e D é o valor digitalizado da entrada convertido pelo ADC. O F28335 é limitado em 0 V a 3V, e possui n=12 bits de resolução.

$$V_{in} = \frac{D * (V_{ref+} - V_{ref-})}{2^n - 1} + V_{ref-} = \frac{D * (3)}{4095} \tag{5.1}$$

A conversão pode ser inicializada por software(por exemplo, inserindo 1 no bit de start), por um sinal externo no pino *GPIO/XINT2_ADCSOC*, ou por um evento do PWM(comparação, underflow). Ele pode operar em vários modos de operação, como por exemplo, sequencial em cascata ou sequencia dual, amostragem sequencial individual ou simultânea e no modo start em sequência única ou contínua.

Os principais registradores envolvidos em um processo de conversão analógica digital são:

– ADCTRL1: registrador de 16 bits responsável por configurar parâmetros na conversão a-d, resetar o ADC para as configurações iniciais, dividir a frequência de entrada por 2 ou por 1, entre outros;

– ADCTRL2: permite e habilita a conversão ser iniciada por um evento do causado pelo ePWM SOCB, via software ou através de um sinal no pino GPIO31-0 GPIO Port A;

– ADCTRL3: configura a taxa de conversão do ADC, dentre outros parâmetros;

– MAXCONV: define o número de canais a serem convertidos. O número colocado no registrador é igual ao número de conversões menos 1. Por exemplo, se MAXCONV = 4, o registrador realizará cinco conversões.

5.7 Exemplo 1

Recomenda-se aos usuários baixar e usar os arquivos de cabeçalho (.h) e exemplos fornecidos pela Texas Instruments para facilitar a programação em C/C++ para os dispositivos C28x. Os arquivos de cabeçalho implementam um método de camada de abstração de *hardware* para permitir fácil acesso ao código C/C++ aos registradoress periféricos de memória mapeada. O pacote de arquivos está disponível no site da Texas Instruments.

Após o download, os arquivos devem ser extraídos para um diretório de escolha do usuário. É importante lembrar de indicar o caminho de busca do compilador C para incluir os arquivos de cabeçalho do diretório *include* (conforme descrito na seção 4.2).

O objetivo do exemplo a seguir é implementar um contador para acender o LED DS2 da placa (pino GPIO32) quando o nível lógico do segundo bit menos significativo

do contador for igual a um. Os LEDS serão comandados via interrupção utilizando o temporizador CpuTimer0.

5.7.1 Criar o projeto e adicionar as bibliotecas necessárias

Deve-se criar o projeto conforme descrito por seção 4.2, adicionando-se os seguintes arquivos ao projeto:

- *"DSP2833x_GlobalVariableDefs.c"*: define todas as variáveis globais;
- *"DSP2833x_CodeStartBranch.asm"*: conjunto de instruções em assembly requerida;
- *"28335_RAM_lnk.cmd"*: conecta as saídas programadas em C para a memória física do dispositivo;
- *"DSP2833x_Headers_nonBIOS.cmd"*: conecta os registradores globais aos correspondentes endereços físicos;
- *"rts2800_fpu32.lib"*: biblioteca de suporte a linguagem C e que precisa estar em todo projeto;
- *"DSP2833x_SysCtrl.c"*: inicialização do sistema de controle e funções suporte;
- *"DSP2833x_ADC_cal.asm"*: copia os dados de calibração do dispositivo para registradores;
- *"DSP2833x_usDelay.asm"*: função de atraso simples usada para inserir um atraso especificado em código;
- *"DSP2833x_PieVect.c"*: funções de inicialização da tabela PIE Vector;
- *"DSP2833x_DefaultIsr.c"*: rotinas para a utilização interrupção;
- *"DSP2833x_CpuTimers.c"*: inicialização dos temporizadores e funções suporte

5.7.2 Código e comentários

```
#include "DSP2833x_Device.h"


extern void InitSysCtrl(void);
extern void InitPieCtrl(void);
extern void InitPieVectTable(void);
extern void InitCpuTimers(void);
extern void ConfigCpuTimer(struct CPUTIMER_VARS *, float, float);


//Protótipo das funções
void Gpio_select(void);
```

```
interrupt void interrup_timer0(void);

void main(void)
{
// contador utilizado para acender o LED
     int counter=0;

// Chamada da função que inicializa o sistema
     InitSysCtrl();

     EALLOW;
// Rehabilitando o watchdog (a função InitSysCtrl() desabilita o WD)
     SysCtrlRegs.WDCR= 0x00AF;

     EDIS;
     DINT; // Desabilita as interrupções

     Gpio_select(); // Função que define o pino GPIO32 como saída

// chamada da função que desabilita todos os canais de interrupção
//e limpa as interrupções pendentes.
     InitPieCtrl();

// leva a memória dos periféricos a um valor inicial.
     InitPieVectTable();

     EALLOW;

// Remapeamento de memória para as interrupções
     PieVectTable.TINT0 = &interrup_timer0;

     EDIS;
//função que limpa os timers e os leva para um estado conhecido
     InitCpuTimers();

// Configura o timer0, a velocidade interna SYSCLKOUT e
//período para overflow de 100 ms.
     ConfigCpuTimer(&CpuTimer0,150,100000);
```

```
// As cinco linhas abaixo habilitam os caminhos necessarios para o
//dispositivo reconhecer uma interrupção.
     PieCtrlRegs.PIEIER1.bit.INTx7 = 1;
     IER |=1;
     EINT; // Macro que habilita as interrupções de controle
     ERTM; // Macro que habilita as interrupções de debug
     CpuTimer0Regs.TCR.bit.TSS = 0; // Inicializa o Timer0start timer0


     while(1)
     {
          while(CpuTimer0.InterruptCount == 0);
          CpuTimer0.InterruptCount = 0;
          EALLOW;
          SysCtrlRegs.WDKEY = 0x55; // GoodKeypara 0 WatchDog
          EDIS;
          counter++;
          //mascara para acender o LED da placa quando o nível lógico
          //do segundo bit menos significativo do contador for igual a 1.
          if(counter&2) GpioDataRegs.GPBDAT.bit.GPIO32= 1;
               else GpioDataRegs.GPBDAT.bit.GPIO32 = 0;
     }
}


void Gpio_select(void)
{
     EALLOW;
     GpioCtrlRegs.GPAMUX1.all = 0;
     GpioCtrlRegs.GPAMUX2.all = 0;
     GpioCtrlRegs.GPBMUX1.all = 0;
     GpioCtrlRegs.GPBMUX2.all = 0;
     GpioCtrlRegs.GPCMUX1.all = 0;
     GpioCtrlRegs.GPCMUX2.all = 0;
     GpioCtrlRegs.GPADIR.all = 0;
     GpioCtrlRegs.GPBDIR.all = 0;
     GpioCtrlRegs.GPCDIR.all = 0;
     GpioCtrlRegs.GPBDIR.bit.GPIO32 = 1;
```

```
     EDIS;
}

interrupt void interrup_timer0(void)
{
     CpuTimer0.InterruptCount++;
     EALLOW;
// Reset do WatchDog caso o código falhar
     SysCtrlRegs.WDKEY = 0xAA;
     EDIS;
// Para reconhecer a última linha da interrupção e voltar
//para o programa principal
     PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
```

5.8 Exemplo 2

Neste exemplo será realizada a conversão analógica digital das entradas ADCIN_A0 e ADCIN_A1. Nestas entradas estão dois resistores variáveis VR1 e VR2. A conversão dos valores de tensão do resistores iniciará automaticamente quando o sinal SOCA do ePWM2 atingir o tempo especificado e os valores convertidos serão mostrados nos LEDs (forma binária), onde a interrupção Timer0 será utilizada para permitir que a mudança dos LEDS seja notada.

Primeiramente é preciso adicionar todos os arquivos do exemplo anterior ao projeto e o arquivo *DSP2833x_Adc.c*

```
#include "DSP2833x_Device.h"

// Protótipo de funções externas a este código
extern void InitAdc(void);
extern void InitSysCtrl(void);
extern void InitPieCtrl(void);
extern void InitPieVectTable(void);
extern void InitCpuTimers(void);
extern void ConfigCpuTimer(struct CPUTIMER_VARS *, float, float);
extern void display_ADC(unsigned int);

//Protótipo de funções dentro deste código
void Gpio_select(void);
interrupt void cpu_timer0_isr(void);
```

```
interrupt void adc_isr(void);

//Variavéis Globais
unsigned int Voltage_VR1;
unsigned int Voltage_VR2;

void main(void)
{
InitSysCtrl(); // Inicialização Básica

EALLOW;
SysCtrlRegs.WDCR= 0x00AF; // Reabilitando o WatchDog
EDIS;

DINT; //Desabilita todas interrupções

// Função que define os pinos GPIO9, GPIO11, GPIO34 e GPIO49 como saídas
Gpio_select();

InitPieCtrl();
InitPieVectTable();

// Setando o conversor analógico digital
InitAdc();
AdcRegs.ADCTRL1.all = 0;

//Definindo a Janela de amostragem
AdcRegs.ADCTRL1.bit.ACQ_PS = 7;

// Escolhendo o modo cascata
AdcRegs.ADCTRL1.bit.SEQ_CASC =1;
AdcRegs.ADCTRL1.bit.CPS = 0; // dividindo por 1
AdcRegs.ADCTRL1.bit.CONT_RUN = 0; // modo não contínuo
AdcRegs.ADCTRL2.all = 0;

//Habilitando a interrupção do SEQ1
AdcRegs.ADCTRL2.bit.INT_ENA_SEQ1 = 1;

//trigger dado pelo sinal ePWM_SOC(do ePWM2) para começar conversão
```

```
AdcRegs.ADCTRL2.bit.EPWM_SOCA_SEQ1 =1;

// 0=interrupção após o fim de cada sequência
AdcRegs.ADCTRL2.bit.INT_MOD_SEQ1 = 0;

// ADC clock: FCLK=HSPCLK/2 * ADCCLKPS (HSPCLK=75MHz e FCLK=12.5MHz)
AdcRegs.ADCTRL3.bit.ADCCLKPS = 3;

AdcRegs.ADCMAXCONV.all = 0x0001;// Conversão de dois canais
AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0;// ADCINA0 como 1ª conversão
AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 1; // ADCINA1 como 2ª conversão
EPwm2Regs.TBCTL.all = 0xC030; // Setando o ewp2
/*
bit 15-14 11: FREE/SOFT, 11 = ignore emulation suspend
bit 13 0: PHSDIR, 0 = count down after sync event
bit 12-10 000: CLKDIV, 000 => TBCLK = HSPCLK/1
bit 9-7 000: HSPCLKDIV, 000 => HSPCLK = SYSCLKOUT/1
bit 6 0: SWFSYNC, 0 = no software sync produced
bit 5-4 11: SYNCOSEL, 11 = sync-out disabled
bit 3 0: PRDLD, 0 = reload PRD on counter=0
bit 2 0: PHSEN, 0 = phase control disabled
bit 1-0 00: CTRMODE, 00 = count up mode
*/

// TPPRD +1 = TPWM/(HSPCLKDIV * CLKDIV * TSYSCLK)=20 us/6.667ns
EPwm2Regs.TBPRD = 2999;

// Configurando para que a conversão seja iniciada pelo ePWM2
EPwm2Regs.ETPS.all = 0x0100;
/*
bit 15-14 00: EPWMxSOCB, read-only
bit 13-12 00: SOCBPRD, don't care
bit 11-10 00: EPWMxSOCA, read-only
bit 9-8 01: SOCAPRD, 01 = generate SOCA on first event
bit 7-4 0000: reserved
bit 3-2 00: INTCNT, don't care
bit 1-0 00: INTPRD, don'tcare
*/
```

```
//Fazendo que o sinal SOCA acione a conversão
EPwm2Regs.ETSEL.all = 0x0A00;
/*
bit 15 0: SOCBEN, 0 = disable SOCB
bit 14-12 000: SOCBSEL, don't care
bit 11 1: SOCAEN, 1 = enable SOCA
bit 10-8 010: SOCASEL, 010 = SOCA on PRD event
bit 7-4 0000: reserved
bit 3 0: INTEN, 0 = disable interrupt
bit 2-0 000: INTSEL, don't care
*/
EALLOW;
PieVectTable.TINT0 = &cpu_timer0_isr;
PieVectTable.ADCINT = &adc_isr;
EDIS;

InitCpuTimers(); //Configuração dos CPUTimers
ConfigCpuTimer(&CpuTimer0,150,100000);
PieCtrlRegs.PIEIER1.bit.INTx7 = 1; // CPU Timer 0
PieCtrlRegs.PIEIER1.bit.INTx6 = 1; // ADC

IER |=1;

EINT;
ERTM;

CpuTimer0Regs.TCR.bit.TSS = 0; //Inicializando o timer0

while(1)
{
while(CpuTimer0.InterruptCount <5)
{
// delay de 500 ms
EALLOW;
SysCtrlRegs.WDKEY = 0x55;
EDIS;
}
//Mostrar a Voltagem do resistor1 nosLEDs
display_ADC(Voltage_VR1);
```

```
while(CpuTimer0.InterruptCount <10) // delay de 1000 ms
{
EALLOW;
SysCtrlRegs.WDKEY = 0x55;
EDIS;
}

//Mostrar a Voltagem do resistor2 no LEDs
display_ADC(Voltage_VR2);
CpuTimer0.InterruptCount = 0;
}
}

void Gpio_select(void)
{
EALLOW;
GpioCtrlRegs.GPAMUX1.all = 0;
GpioCtrlRegs.GPAMUX2.all = 0;
GpioCtrlRegs.GPBMUX1.all = 0;
GpioCtrlRegs.GPBMUX2.all = 0;
GpioCtrlRegs.GPCMUX1.all = 0;
GpioCtrlRegs.GPCMUX2.all = 0;
GpioCtrlRegs.GPADIR.all = 0;
GpioCtrlRegs.GPADIR.bit.GPIO9 = 1;
GpioCtrlRegs.GPADIR.bit.GPIO11 = 1;
GpioCtrlRegs.GPBDIR.all = 0;
GpioCtrlRegs.GPBDIR.bit.GPIO34 = 1;
GpioCtrlRegs.GPBDIR.bit.GPIO49 = 1;
GpioCtrlRegs.GPCDIR.all = 0;
EDIS;
}

interrupt void cpu_timer0_isr(void)
{
CpuTimer0.InterruptCount++;
EALLOW;
SysCtrlRegs.WDKEY = 0xAA; // service WD #2
EDIS;
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
```

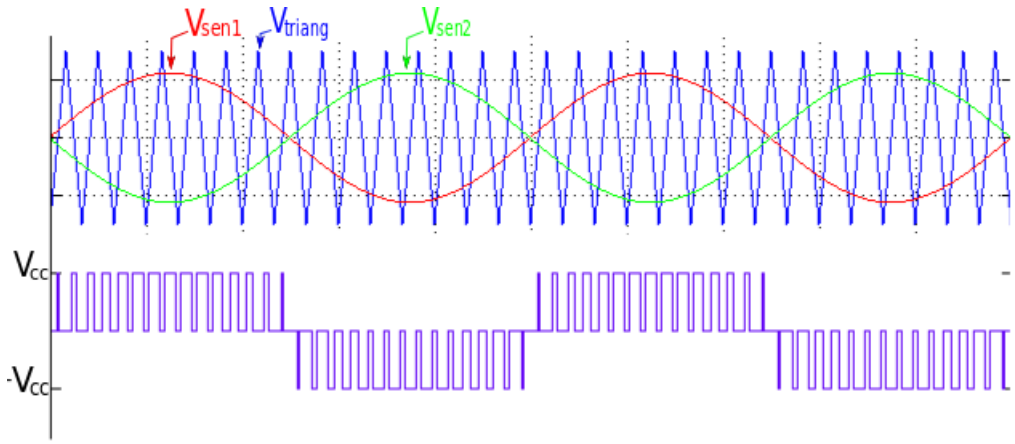
```
}

interrupt void adc_isr(void)
{
Voltage_VR1 = AdcMirror.ADCRESULT0; //Armazenamento dos resultados
Voltage_VR2 = AdcMirror.ADCRESULT1;//Armazenamento dos resultados

// Reinicialização para a próxima sequencia do ADC
AdcRegs.ADCTRL2.bit.RST_SEQ1 = 1; // Reseta SEQ1
AdcRegs.ADCST.bit.INT_SEQ1_CLR = 1; //Força um clear INT SEQ1 bit
// Reconhecimento da última linha de código da interrupção
PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
}
```

6 Resultados

Neste capítulo serão discutidos os resultados obtidos da montagem.

Modulação da largura de pulsos com referência senoidal (PWMS) baseia-se na comparação de uma onda de referência senoidal de baixa freqüência com uma onda triangular de alta freqüência. O cruzamento dessas duas formas de onda estabelece a duração dos sinais de comando das chaves estáticas controladas. Para a montagem de um inversor monofásico de ponte completa foi utilizado dois sinais de referência V_{sen1} e V_{sen2}. Onde os ePWM3A e ePWM3B foram configurados para operar no modo simétrico (onda triangular), de formar complementar e tendo como referência o sinal V_{sen1}. Os sinais ePWM4A e ePWM4B foram configurados de forma semelhante, mas tendo como referência o sinal V_{sen2}.

Figura 27 – Modulação da largura de pulsos com referência senoidal (PWMS) três níveis



Fonte: Autora

O código da função de controle abaixo foi implementado para o controle em malha aberta do conversor de onda completa por modulação da largura de pulsos com referência senoidal(PWMS).

```
void Fx_Control(float v1,float v2,float v3,float v4,
                          float i1,float i2,float i3,
                          float i4,float i5,float i6)
{
////// SINE //////
   dt=dt+h1;
```

```
//sinais de referência senoidais
   v1_ref=  0.9*sin(2*pi*60*dt);
   v2_ref= -0.9*sin(2*pi*60*dt);
   if (dt>=2*pi) dt=0;

// OPEN LOOP
//sinais de referência senoidais com amplitude ajustada de acordo com
//a triangular do PWM
   v1_pwm = PWM_PRD*(0.5*v1_ref+0.5);
   v2_pwm = PWM_PRD*(0.5*v2_ref+0.5);

   if(v1_pwm>3740) v1_pwm=3740;
   if(v1_pwm<10)   v1_pwm=10;

   if(v2_pwm>3740) v2_pwm=3740;
   if(v2_pwm<10)   v2_pwm=10;
////// PWM  //////
//unidade comparadora do PWM
   EPwm3Regs.CMPA.half.CMPA = v1_pwm;
   EPwm3Regs.CMPB = v1_pwm;
   EPwm4Regs.CMPA.half.CMPA = v2_pwm;
   EPwm4Regs.CMPB = v2_pwm;

}
```

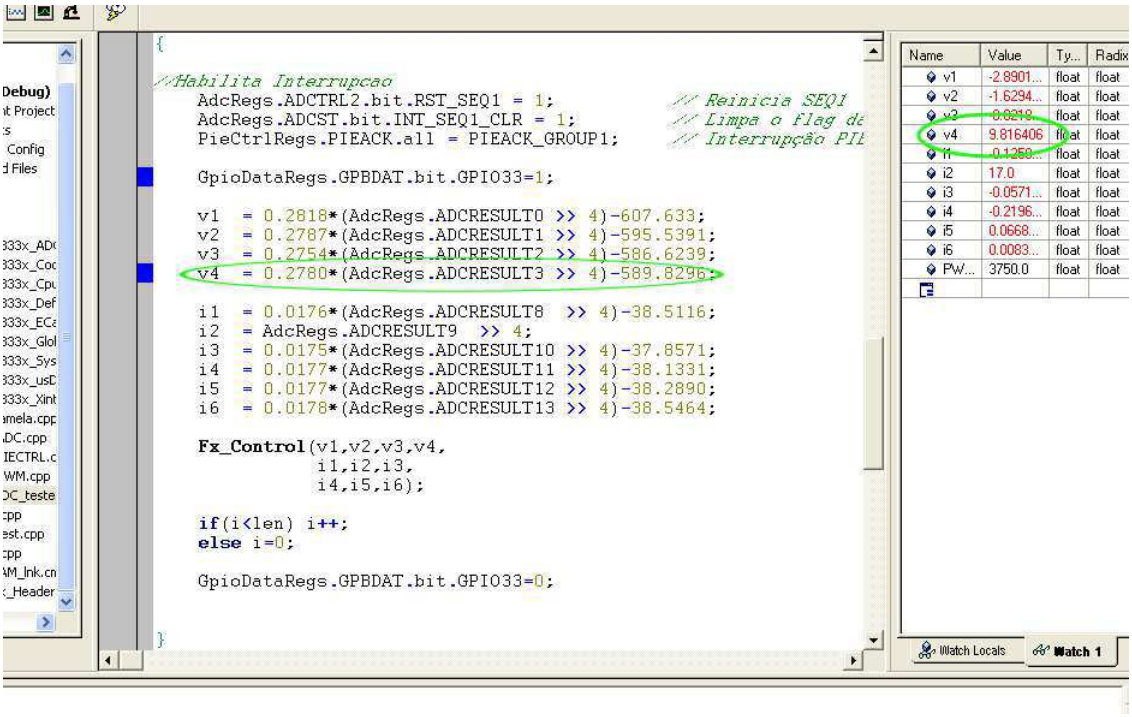
Os sinais de referência senoidais comparados com uma onda portadora triangular e a saída PWM de três níveis são ilustrados na Figura 27.

Na Figura 29, é ilustrada os sinais de comando das saídas ePWM3A e ePWM3B para comutar as chaves presentes no driver 3 e 4. Também é possível observar o tempo morto configurado pelo drivers, para proteger o sistema de curto-circuito decorrentes de chaves fechadas num mesmo instante.

Na Figura 28, é possível observar o funcionamento do ADC, que ocorreu conforme esperado. É ilustrada também parte da configuração da rotina de interrupção do ADC e a calibração dos sensores de tensão e corrente, destacando o sensor V4. Ao lado, os valores de V4 em DC lidos são mostrados em tempo-real na janela Watch.

Na Figura 30, é possível observar o sinal de saída medido por um osciloscópio conectado as saídas 3 e 4 como visto na Figura 2 do inversor de três níveis com as características ilustradas na Figura 27, onde a tensão de barramento é Vcc=10V. Além

Figura 28 – Leitura do sensor V4 pelo ADC devidamente calibrado



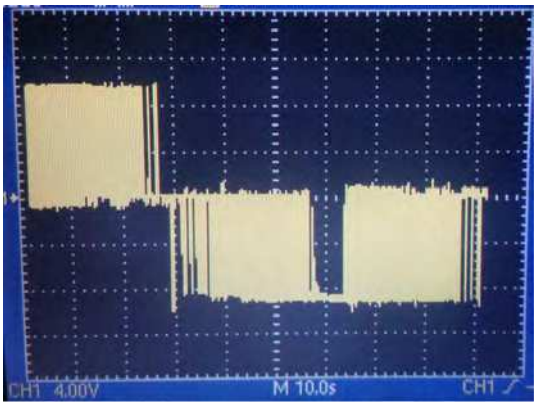
Fonte: Autora

Figura 29 – Sinais de comando para comutar as chaves

Fonte: Autora

da forma de onda ter sido coerente com a teoria, a amplitude do sinal de saída também apresentou o valor Vcc esperado, conforme ilustrada na Figura 28 de 10V.

Figura 30 – Tensão de saída do inversor com três níveis (+10V,0 e -10V)



Fonte: Autora

7 Conclusão

O presente guia teve como objetivo apresentar a bancada de testes do LEIAM, assim como uma breve descrição de seus componentes, e também apresentar de forma clara e abreviada as principais configurações necessárias para que um usuário iniciante possa começar a desenvolver sua aplicação desejada rapidamente usando o DSC F28335. O resultado do teste de um inversor monofásico ponte completa foi satisfatório. O controle em malha aberta do conversor por modulação da largura de pulsos com referência senoidal funcionou conforme esperado.

Como trabalho futuro, é sugerido um estudo mais aprofundado do código implementado pela Dr. Camila Gerke, onde as principais configurações do DSC vistas nesse guia poderão ser utilizadas na implementação de um sistema mais robusto, somado a uma estratégia de controle mais elaborada e um estudo dos harmônicos gerados.

Referências

BORMANN, F. *Introduction to TMS320F28335*. [S.l.], 2011.

GEHRKE, C. S. *Localizacao, dimensionamento e controle de compensadores ativos em redes de distribuicao*. Tese (Doutorado) — Universidade Federal de Campina Grande, Campina Grande, Brasil, 2014.

LEM. *Current Transducer LTS 25-NP*. [S.l.], 2012.

LEM. *Voltage Transducer LV 25-P*. [S.l.], 2012.

LEMOS, J. W. *Filtros Ativos Cooperativos em Redes de Baixa Tensão*. Dissertação (Mestrado) — Universidade Federal de Campina Grande, 2009.

PANTECH SOLUTIONS. *Getting Started with Code Composer Studio V3.3*. [S.l.], 2009.

POMILIO, J. A. *Eletrônica de Potência para Geração, Transmissão e Distribuição de Energia Elétrica*. Dissertação (Apostila) — Universidade Estadual de Campinas, 2013.

SEMIKRON. *PCB Drivers*. [S.l.], 2008.

SPECTRUM DIGITAL, INC. *eZdsp F28335 Technical Reference*. [S.l.], 2007.

TEXAS INSTRUMENTS. *Data Manual: TMS320F28335 Digital Signal Controllers (DSCs)*. [S.l.], 2007.

TEXAS INSTRUMENTS. *C28xTM Digital Power Supply Workshop*. [S.l.], 2008.

TEXAS INSTRUMENTS. *Reference Guide: Enhanced Pulse Width Modulator (ePWM) Module*. [S.l.], 2009.

TEXAS INSTRUMENTS. *Reference Guide: Analog-to-Digital Converter (ADC)*. [S.l.], 2010.

TEXAS INSTRUMENTS. *Reference Guide: System Control and Interrupts*. [S.l.], 2012.

VIEIRA, L. P. *Guia de Utilização do Controlador Digital de Sinal TMS320F28335*. Dissertação (Trabalho Final de Curso) — Federal de Juiz de Fora, 2013.