https://eventos.utfpr.edu.br//sicite/sicite2019

Plataforma digital para controle de conversores estáticos

Digital platform for static converter control

Thiago Fialho Guimarães
Autor
thiagofialho.vg@gmail.com
Universidade Tecnológica Federal do Paraná - Campus Curitiba, Curitiba, Paraná, Brasil

Alceu André Badim
handredh@gmail.com
Universidade Tecnológica Federal do Paraná - Campus Curitiba, Curitiba, Paraná, Brasil

RESUMO

Este trabalho apresenta o desenvolvimento de uma plataforma para controle digital para conversores estáticos utilizando microcontroladores. É apresentado hardware e software utilizado para produzir a plataforma assim como o método de controle e suas equações, além do circuito utilizado para testes, que inclui um conversor Boost, filtros e circuitos para aquisição de sinais baseados em amplificadores e resistores. Os resultados obtidos foram documentados, de forma que a dados encontrados possam ser facilmente utilizados e reproduzidos em diferentes estratégias de controle de conversores, citando dificuldades encontradas e métodos que trouxeram benefícios, ajudando a diminuir o tempo gasto na implementação do controle digital.

PALAVRAS-CHAVE: Conversor estático, Controle Digital, Tiva.

Recebido: 19 ago. 2019.
Aprovado: 01 out. 2019.

ABSTRACT

This paper presents the development of a digital control platform for static converters using microcontrollers. Hardware and software used to produce the platform is presented as well as the control method and its equations in addition to the circuit used for testing, which includes a Boost converter, filters and signal acquisition circuits based on amplifiers and resistors. The obtained results were documented, so that the data found can be easily used and reproduced in different converter control strategies, citing difficulties and methods that brought benefits, helping to reduce the time spent in the implementation of digital control.

KEYWORDS: Static converter, Digital control, Tiva.

INTRODUÇÃO

Em grande parte das aplicações com conversores de potência são utilizados circuitos de controle que mantem o sistema com as características desejadas, estes podem ser implementados de forma analógica, com circuitos eletrônicos compostos por amplificadores operacionais e filtros com um baixo custo, porém de difícil modificação. Outro meio de implementar um circuito de controle é de forma digital utilizando aproximações matemáticas para que equações contínuas sejam representadas adequadamente no domínio do tempo discreto podendo assim ser realizados com microcontroladores, que possibilitam uma mudança do modelo de controle com o mesmo hardware.

Para realizar controle digital de conversores com microprocessadores existem vários fatores que devem ser analisados e testados antes da sua implementação, devido a limitações da frequência de trabalho e memória interna ROM e RAM disponíveis e adequação das medidas do circuito de potência.

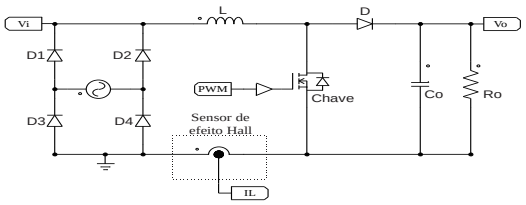
Visando diminuir dificuldades e o tempo gasto na implementação do controle utilizando microcontroladores nos conversores realizou-se este trabalho com o objetivo de criar uma plataforma digital para controle de conversores estáticos, abordando de forma genérica os estágios necessários, desde a adequação dos sinais a serem lidos dos circuitos de potência até a e implementação do modelo de controle.

CONTROLE DE CONVERSORES

Para verificar o funcionamento da plataforma de controle foi implementada a técnica utilizando o microcontrolador Tiva C series TM4C1294XL, com processador ARM Cortex-M4 de 32 bits com 120MHz, 1MB Flash, 256KB SRAM, 6KB e ADC de 12 bits de 3.3V suportando tensões até 5V. Utilizou também a biblioteca TivaWare disponibilizada pela Texas em conjunto com o ambiente de desenvolvimento integrado (IDE) Code Composer Studio, indicada pela fabricante do processador para programação dos seus microcontroladores.

O circuito utilizado como base para testes foi um conversor Boost, figura 1, onde L=11.34mH, Co=220uF e Ro=600 ohms.

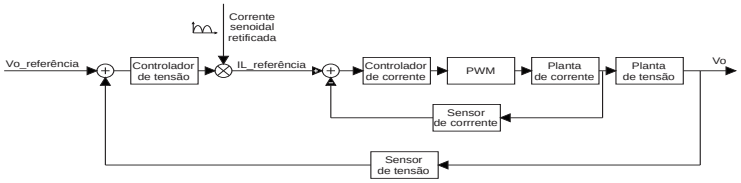
Figura 1 – Conversor Boost



Fonte: Próprio autor.

O modelo de controle utilizado segue o digrama de blocos da figura 2, composto por duas malhas. A malha externa é a de tensão, que tem o sinal de erro como a diferença da tensão de referência e de saída obtida no conversor, já a malha mais interna é a de corrente, que tem como valor de referência um seno retificado multiplicado pelo sinal de controle da malha de tensão e possui uma realimentação dada pela corrente do indutor.

Figura 2 – Diagrama para controle do conversor Boost



Fonte: Próprio autor.

Para realizar o controle desse conversor é necessário medir a tesão de entrada Vi, a corrente no indutor IL, e a tensão de saída Vo. Para medir a corrente no indutor IL utilizou o sensor de efeito Hall modelo HXS 20-NP/SP30, já para a tensão de entrada e saída, circuitos com divisor de tensão, *buffer* e filtros RC para adequar o sinal para leitura do processador.

Os circuitos de aquisição de dados produzem um ganho nas leituras das tensões de entrada e saída do conversor assim como da corrente do indutor que precisa ser utilizada nos sinais obtidos no ADC2, ADC3 ADC1 para que a equação de controle projetada funcione corretamente. Calculando o ganho para o ADC2, ADC3 e ADC1 temos respectivamente:

$$Gvi = 186.63 \quad (1)$$
$$Gvo = 333.30 \quad (2)$$
$$Gi = 26.4 \quad (3)$$

Para controle projetou-se as equações no tempo contínuo com o método de Tustin. Para conseguir a versão discretizada o tempo de amostragem adotado deve ser escolhido de forma que o controle seja realizado na mesma frequência que o chaveamento do conversor. A equações 5 e 6 são respectivamente as malhas de tensão e de corrente utilizadas no controle.

$$Cv = \frac{0.01\,z - 0.00995}{z-1} \quad (5)$$

$$Ci = \frac{0.7983\,z\text{^}2 + 0.04073\,z - 0.7576}{z\text{^}2 - 0.778\,z - 0.222} \quad (6)$$

É importante ressaltar que valores dos ganhos escolhidos para o controlador de tensão, assim como a equação da malha de corrente, estão diretamente ligados as características do conversor. Como a intenção do trabalho é abordar a utilização de microcontroladores para execução do controle, os cálculos e formulas encontradas dos controladores não serão abordados de forma detalhada, fazendo apenas o seu uso para facilitar o entendimento e exemplificar a plataforma de controle digital produzida.

Com todos os circuitos de aquisição dos sinais, proteções para o processador e equações de controle dimensionadas o código para programar o microcontrolador pode ser escrito, e assim o controle do conversor.

O controle é feito definindo o chaveamento dos interruptores utilizando uma onda PWM (*pulse width modulation*). O microcontrolador utilizado é capaz de produzir uma onda de pulsos com boa precisão, para isso são necessários uma série de comandos de ativação. A escolha do pino para utilização do sinal PWM não é arbitraria, as ligações já são pré-estabelecidas e bastam apenas ser habilitadas com os dados encontrados no *datasheet*.

Para controle do conversor Boost utilizou-se a frequência de chaveamento de 20k Hz e o PWM no pino PF1.

Assim como o PWM, o periférico de leitura analógico digital (ADC) conta com pinos pré-definidos para cada canal de aquisição, para verificar os canais e pinos disponíveis é necessário consultar o *datasheet.* Os canais de leitura utilizados foram o 0, 1 e 2 correspondem aos pinos PE3, PE2 e PE1 do processador.

Como ADC é de 12 bits, o valor lido será um número de 0 a 4095, assim possuí um ganho do periférico de leitura dado por:

$$Gadc = \frac{1}{4095} \quad (7)$$

Para o controle há necessidade ainda de realizar uma interrupção periódica, nesse caso em especifico o PWM pode ser utilizado como *trigger* para uma interrupção feita pelo próprio ADC do microcontrolador, já que cada ciclo deverá ser relizada a leitura dos sinais e cálculo das malhas de controle. Além disso, fazendo a interrupção dessa forma é possível dar um atraso curto na leitura dos sinais com a função ADCPhaseDelaySet, que recebe a leitura modulo ADC utilizado com um atraso dado em graus múltiplos de 22.5. Esse atraso evita a aquisição de sinais no mesmo instante da comutação das chaves, o que melhora o desempenho tanto do ADC quanto do controle.

A função que é executada em cada interrupção deve conter a leitura das amostras e cálculo das malhas de controle. Nesta função foram adicionadas estratégias importantes. Uma delas é colocar o pino 4 da porta B em nível alto no início, e no fim da interrupção o mesmo pino é colocado em nível logico baixo. Dessa forma analisando o sinal com um osciloscópio é possível vizualizar o esforço do processador pelo tempo em nível logico alto da onda, sabendo se há tempo hábil para execução da malha de controle, quanto menos tempo em nível logico alto mais rápido o processador executa o código da interrupção.

A saturação das malhas de controle também é necessária, pois em uma inicialização do microcontrolador sem o funcionamento do conversor os sinais de controle podem assumir valores muito grandes, o que pode gerar problemas ao tentar controlar o circuito de potência.

As funções descritas anteriormente, inicialização do PWM e ADC, são funções externas e precisam ser chamadas na função principal, além disso é necessário definir a frequência de clock do processador. Como a aplicação exige uma grande velocidade do processador utiliza-se a máxima possível que é de 120 MHz.

O código de programação utilizado no microcontrolador é mostrado na tabela 1, e com pequenas alterações podem ser utilizados em outras estratégias de controle.

Tabela 1 – Código do controle para o microcontrolador

```
//Incluindo bibliotecas de funções utilizadas
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/pwm.h"
#include "driverlib/adc.h"
#include "driverlib/interrupt.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "inc/hw_types.h"
#include "inc/hw_ints.h"
// The error routine that is called if the driver library encounters an error
#ifdef DEBUG
void
__error__(char *pcFilename, uint32_t ui32Line)
{ while(1); }
#endif
```

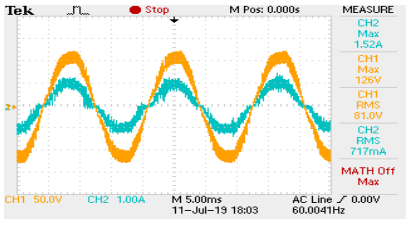
```
// Função para inicializar o modulo PWM
int32_t PWM_FREQ = 20000;       //Frequência do PWM
int32_t CPU_FREQ = 120e6;       //Frequência do clock
int32_t pwm_word;               //pwm_word = (CPU_FREQ/PWM_FREQ);
void Init_PWM (void)
{   ROM_GPIOPinConfigure(GPIO_PF1_M0PWM1);       //Configura o pino PF1 como PWM
    ROM_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0); //Habilita o modulo PWM 0
    //Configura tipo de PWM e informa utilização do gerador PWM 0
    PWMGenConfigure(PWM0_BASE,PWM_GEN_0,PWM_GEN_MODE_DOWN|PWM_GEN_MODE_NO_SYNC|PWM_GEN_MODE_DBG_STOP);
   PWMGenPeriodSet(PWM0_BASE, PWM_GEN_0, pwm_word); //Configura a frequência do PWM através de um período que depende da frequência de clock
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1, 1); //Configura a largura do pulso inicial (0%)
    PWMGenEnable(PWM0_BASE, PWM_GEN_0); //Habilita o PWM de modulo 0 gerador 0 no pino PF1
    PWMGenIntTrigEnable (PWM0_BASE, PWM_GEN_0,PWM_TR_CNT_ZERO); //Habilita trig do PWM
    PWMOutputState(PWM0_BASE,PWM_OUT_1_BIT, true); //Habilita as saida 1 do PWM}
uint32_t ui32Value[8];          //buffer para gravar as variáveis
float Ui[3]={0.0f}, Uv[3]={0.0f};   //Variaveis da malha de corrente e malha de tensão
float Ei[3]={0.0f}, Ev[3]={0.0f};   //Erro de corrente, Erro de tensão
float Vin=0.0f, Vout=0.0f, IL=0.0f, ST = 200.0f; //Tensão de entrada, Tensão de saida, Corrente no indutor, Setpoint de tensão
void ADCInterrupcao()
{   ADCIntClear(ADC0_BASE, 0); // Limpa a flag de interrupção
    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, 0xFF);
    ADCSequenceDataGet(ADC0_BASE, 0, ui32Value); // Faz a leitura do ADC 0
    // Normaliza os valores do ADC para valores reais
    Vin = (ui32Value[1])*(0.04557509); //PE2
    Vout = (ui32Value[2])*(0.08139186); //PE1
    IL = (ui32Value[0])*(0.00644688); //PE3
    Ev[0] = ST - Vout;
    Uv[0] = Uv[1]+0.01*Ev[0]-0.00995*Ev[1];
    if(Uv[0] > 1.1) { Uv[0] = 1.1; } //Saturação da malha de tensão
    else if(Uv[0] < -1.5) { Uv[0] = -1.5; }
    //Atualiza valores da malha de tensão
    Uv[2] = Uv[1];
    Uv[1] = Uv[0];
    Ev[2] = Ev[1];
    Ev[1] = Ev[0];
    Ei[0] = (0.009259221)*(Vin)*(Uv[0]) - IL; //Erro da corrente
    Ui[0] = ((0.778*Ui[2])-((-0.222)*Ui[1])+((0.798)*Ei[0])+((0.041)*Ei[1])+((-0.758)*Ei[2]));
    if(Ui[0] > 0.95) { Ui[0] = 0.95; }
    else if (Ui[0]<0.1) { Ui[0] = 0.1; }
    //Atualiza valores da malha de corrente
    Ei[2] = Ei[1];
    Ei[1] = Ei[0];
    Ui[2] = Ui[1];
    Ui[1] = Ui[0];
    PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1, pwm_word*Ui[0]); //Atualização do valor do PWM
    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_4, 0x00);
}
void ADCInit(void)
{    ROM_SysCtlPeripheralEnable (SYSCTL_PERIPH_ADC0); // Habilita o ADC0
   //Configura a primeira sequência de amostras para capturar o valor do canal 0 quando o PWM iniciar
    ADCSequenceConfigure(ADC0_BASE, 0, ADC_TRIGGER_PWM0, 0);
    ADCPhaseDelaySet(ADC0_BASE,ADC_PHASE_22_5); //Habilita um delay entre a leitura do AD e do PWM
    //Seta as sequencias de amostras
    ADCSequenceStepConfigure(ADC0_BASE, 0, 0, ADC_CTL_CH0);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 1, ADC_CTL_CH1);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 2, ADC_CTL_CH2);
    //Informa quantos canais ADC foram utilizados
    ADCSequenceStepConfigure(ADC0_BASE, 0, 3, ADC_CTL_IE | ADC_CTL_CH2|ADC_CTL_END);
    ADCIntDisable(ADC0_BASE, 0);
    ADCIntClear(ADC0_BASE, 0);
    ADCIntRegister(ADC0_BASE, 0, &ADCInterupcao); //configura função a ser executada na interrupção
    ADCIntEnable(ADC0_BASE, 0);
    ADCSequenceEnable(ADC0_BASE, 0);
}
int main(void)
{ // Configura clock do Tiva
 SysCtlMOSCConfigSet(SYSCTL_MOSC_SESRC);                         SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ|SYSCTL_OSC_MAIN|SYSCTL_USE_PLL|SYSCTL_CFG_VCO_480),CPU_FREQ);
    pwm_word = (CPU_FREQ/PWM_FREQ); // Periodo para PWM com frequência igual a PWM_FREQ
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); // Habilita o port F
    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOF)) {}
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE); // Habilita o port E
    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOE)) {}
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB); // Habilita o port B
    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOB)) {}
    GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4); // Habilita o pino PB4 como saída
    Init_PWM ();  // Inicializa o PWM
    ADCInit();// Inicializa o ADC e interupção do PWM
    while(1){}}
```

Fonte: Próprio autor.

RESULTADOS

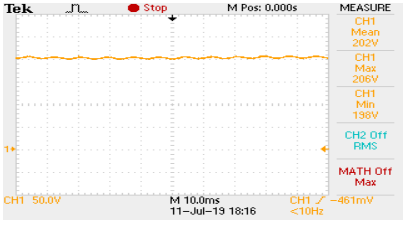
O controle do conversor Boost tem como obtivo aumentar o fator de potencia fazendo com que a corrente e tensão tenham o mesmo formato e ainda manter a tensão de saída constante em um valor definido. Com a melhora das medidas o controle do conversor funcionou corretamente, assim em testes obteve-se tensão e corrente de entrada com mesmo formato chegando a um fator de potencia de 0,958 e distorção harmonica de 35%, como mostra a figura 3 e tensão de saída constante, conforme a figura 4.

Figura 3 – Tensão e corrente de entrada



Fonte: Próprio autor.

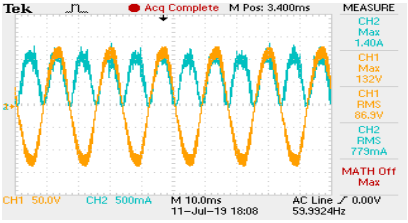
Figura 4 – Tensão de saída



Fonte: Próprio autor.

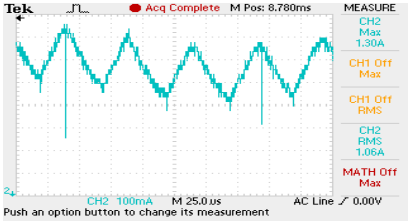
A corrente controlada no conversor é a que passa pelo indutor, assim realizando a medida com um osciloscópio a tensão de entrada e corrente no indutor, mostrada na figura 5 e ainda a corrente em alta frequência no indutor, figura 6.

Figura 5 – Corrente no indutor



Fonte: Próprio autor.

Figura 6 – Corrente em alta frequência



Fonte: Próprio autor.

CONCLUSÕES

Depois dos testes realizados verificou o funcionamento do protótipo utilizando a estratégia de controle proposta, mostrando a funcionalidade da plataforma montada e a viabilidade da sua implementação que pode ser utilizada em novos tipos de controles.

Algumas dificuldades foram encontradas, a principal foi na leitura dos sinais de entrada do microcontrolador, principalmente o de corrente, devido à grande quantidade de ruído que atrapalha a leitura. Os ruídos foram amenizados implementando filtros e evitando que as leituras fossem realizadas no instante da comutação da chave, que é o período que apresenta mais ruídos no sinal. Outra dificuldade fica por conta da impossibilidade de analisar variáveis do código em tempo real, o que poderia ajudar na análise do comportamento do circuito e identificação de erros.

AGRADECIMENTOS

A Deus por minha vida, família e amigos.

Ao Prof. Dr. Alceu André Badim pela oportunidade e apoio deste trabalho.

A Universidade tecnológica Federal do Paraná, pelo conhecimento e oportunidade de fazer o curso.

A fundação Araucária pelo apoio durante a realização deste trabalho

REFERÊNCIAS

Texas Instruments Inc. Datasheet: EVAL KIT TIVA C SERIES LAUNCHPAD, EK-TM4C1294XL. Electronic Publication, 2002.

STMicroeletronics. Datasheet: LM324. Electronic Publication, 1999.

Texas Instruments Inc. Datasheet: LF347. Electronic Publication, 2013.

Life Energy Motion. Datasheet: Current Transducer HXS 20-NP/SP30. Electronic Publication, 2014.