Reuben Palmer Rezende de Sousa

Relatório de TCC

Campina Grande, Brasil

11 de agosto de 2014

Reuben Palmer Rezende de Sousa

Relatório de TCC

Relatório de TCC submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia Elétrica.

Univesidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Enegenharia Elétrica - DEE

Orientador: Alexandre Cunha Oliveira

Campina Grande, Brasil

11 de agosto de 2014

Reuben Palmer Rezende de Sousa
Relatório de TCC/ Reuben Palmer Rezende de Sousa. – Campina Grande, Brasil, 11 de agosto de 2014-
35 p. : il. ; 30 cm.

Orientador: Alexandre Cunha Oliveira

Relatório de TCC – Univesidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Departamento de Enegenharia Elétrica - DEE , 11 de agosto de 2014.

Reuben Palmer Rezende de Sousa

Relatório de TCC

Relatório de TCC submetido à Coordenação de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande, Campus Campina Grande, como parte dos requisitos necessários para obtenção do título de Bacharel em Engenharia Elétrica.

Trabalho aprovado. Campina Grande, Brasil, 11 de agosto de 2014:

Alexandre Cunha Oliveira
Orientador

Professor
Convidado

Campina Grande, Brasil
11 de agosto de 2014

À minha presente e futura família.

Agradecimentos

Agradeço à todos que contribuíram ao longo desta intensa e árdua luta para chegar até aqui. Em especial, sou grato:

- a Deus, pelo infinito amor a todos Seus filhos, por este mundo incrível... por tudo;
- à minha família. Em especial ao meu pai e minha mãe, pelas madrugadas à minha espera para abrir o portão de casa, e por terem me dado forças para abrir um portão que leva a um dos tesouros mais nobres da humanidade: o conhecimento;
- ao professor Alexandre, pela paciência e pela oportunidade de me mostrar novos horizontes através da proposta deste trabalho;
- aos meus colegas, pelo incentivo, pelos momentos de descontração e companheirismo.

"Se quiser derrubar uma árvore na metade do tempo, passe o dobro do tempo amolando o machado."

Provérbio chinês

Lista de abreviaturas e siglas

ADC	Analog-to-Digital Converter
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processing Unit
DMA	Direct Memory Access
GPIO	General-purpose Input/Output
HRPWM	High Resolution Pulse Width Modulation
PIC	Programmable Interrupt Controller
PWM	Pulse Width Modulation

Sumário

1 Introdução

O DSP (Digital Signal Processor, ou Processador Digital de Sinais) é um microprocessador que possui arquitetura otimizada para as necessidades operacionais de processamento de sinais digitais. Um DSP pode processar dados em tempo real, fazendo-o ideal para aplicações em que não são tolerados atrasos; pode utilizar sinais de vídeo, voz, áudio, temperatura, posição que foram digitalizados (podendo digitalizá-los, caso contenha um módulo conversor) e manipulá-los matematicamente.

Desde os primeiros DSPs lançados comercialmente (o NEC μPD7720 e o AT&T DSP1, lançados no início da década de 1980), esse dispositivo, assim como os computadores pessoais, tornou-se cada vez mais popular e de menor custo. É utilizado em muitos produtos de mercado, tais quais tocadores de MP3, telefones celulares, modens, sistemas de controle em geral, etc.

Esses dispositivos são formados por diversos módulos, que formam um sistema consideravelmente complexo, a exemplo do DSP TMS320F28335 da Texas Instruments, cujo diagrama de blocos é mostrado na Figura 1.

Assim como nos microcontroladores PIC, a configuração do DSP é feita através da atribuição de valores a registradores. Devido à complexidade deste dispositivo, essa configuração normalmente exige do usuário consultas aos manuais técnicos do fabricante que possuem centenas de páginas, o que torna esse procedimento bastante penoso, exigindo um tempo considerável de projeto.

Então, surge a necessidade de que seja desenvolvida uma ferramenta que proporcione maior praticidade da configuração, assim como é feito para PICs no compilador da *Custom Computer Services, Inc.*.

O objetivo deste trabalho foi a elaboração de uma ferramenta que facilitasse o processo de configuração do TMS320F28335. Foram estudados o módulo conversor analógico-digital (ADC) e o módulo de Entrada e Saída de Propósito Geral (GPIO), e, utilizando o *framework* Qt, foi elaborado o assistente de configuração com uma interface gráfica simples que gera um arquivo de configuração de acordo com as opções definidas pelo usuário. O assistente permite a configuração apenas do GPIO.

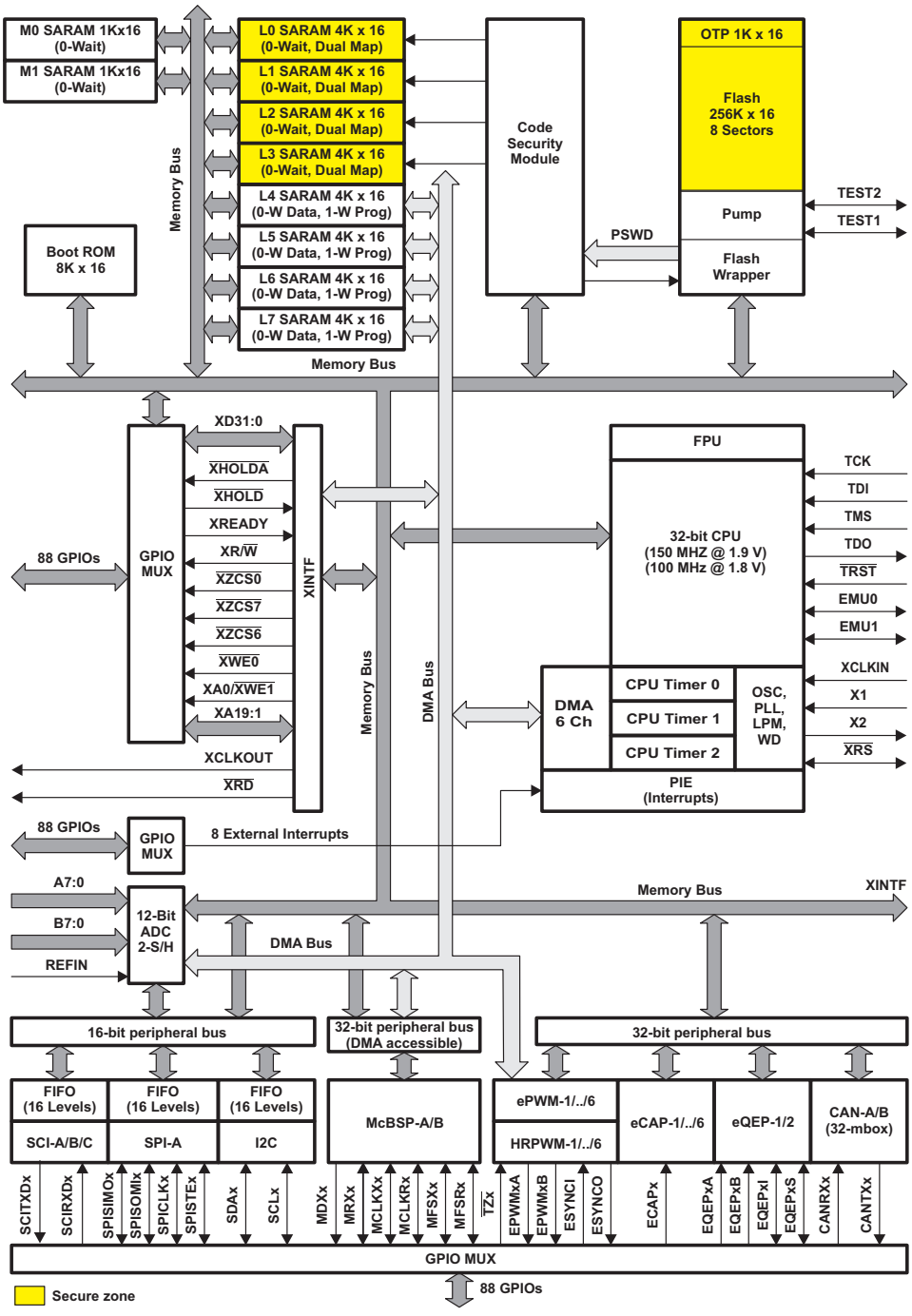


Figura 1 – Diagrama de blocos do DSP TMS320F28335.

2 Descrição do DSP TMS320F28335

Produzido pela Texas Instruments, o TMS320F28335 é um DSP baseado na arquitetura Delfino, e possui, dentre outras características:

- Fabricado com tecnologia CMOS, podendo operar em até 150 MHz; as tensões em seus pinos de Entrada/Saída (E/S) é 3,3 V;
- Barramento com arquitetura Harvard e CPU de 32 bits com precisão de ponto flutuante segundo o padrão IEEE-754;
- Controlador de DMA com seis canais;
- Até 18 saídas PWM, dentre as quais até 6 saídas podendo ser HRPWM;
- Módulo ADC de 12 bits com até 16 canais, e conversão em 80 ns;
- Até 88 pinos multiplexados e individualmente programáveis.

Devido à considerável complexidade dos módulos que constituem o DSP em estudo, juntamente com o breve espaço de tempo para a execução deste trabalho, o aplicativo desenvolvido contempla apenas o módulo de GPIO, descrito na seção seguinte.

2.1 Módulo de Entrada e Saída de Propósito Geral - GPIO

O GPIO é responsável pela configuração de 88 pinos, denominados GPIO0-GPIO87. Os registradores de multiplexação desse módulo permitem selecionar a função de cada pino, que pode ser entrada digital, saída digital, ou conectado a um de até três sinais de E/S de periféricos. Os 88 pinos constituem três portas, sendo duas com 32 bits e uma com 24 bits. A porta A compreende os pinos GPIO0-GPIO31, enquanto que as portas B e C são formadas pelos pinos GPIO32-GPIO63 e GPIO64-GPIO87, respectivamente.

Os seis registradores GPxMUXn (com x = {A, B, C} e n = {1, 2}) são responsáveis por definir a funcionalidade dos pinos individualmente. Cada pino possui 2 bits de configuração nesses registradores. Nas Figuras 12, 13 e 14 (ver anexo) estão especificadas as possíveis funções atribuíveis a cada pino e os valores correspondentes para as portas A, B e C, respectivamente.

Caso o pino seja escolhido como E/S digital, é necessário definir a direção dos dados. Isso é feito através dos três registradores GPxDIR (x = {A, B, C}). Cada bit

desses registradores está associado a um pino, e pode assumir dois valores possíveis: 0, tornando-o entrada, ou 1, que o seleciona como saída.

Pinos selecionados como entrada (digital ou de algum periférico) podem ser submetidos à qualificação, que pode filtrar ruído. Aqueles conectados à entrada de um periférico podem ser assíncronos (sem sincronização), ou serem sincronizados com o relógio da CPU (SYSCLKOUT). Entradas digitais possuem uma possibilidade adicional de qualificação, através de janelas de amostragem. A seguir são listados cada um desses tipos de operação:

- **Sem sincronização (entrada assíncrona):** esse modo é utilizado por periféricos em que a sincronização não é necessária, ou em que o próprio periférico realiza a sincronização.
- **Sincronização apenas com SYSCLKOUT:** é o modo de qualificação de todos os pinos em *reset*. Devido ao sinal de entrada ser assíncrono, pode levar até um ciclo de tempo do relógio (SYSCLKOUT) para que seja detectada a mudança de estado no pino.
- **Qualificação através de janela de amostragem:** o sinal é sincronizado com o relógio do sistema (SYSCLKOUT) e então é qualificado por um número de ciclos especificado. Dois parâmetros são especificados pelo usuário nesse tipo de qualificação: o período de amostragem e o número mínimo de amostras obtidas para identificar uma possível mudança no pino de entrada.

Na Figura 2 é mostrado o diagrama de blocos para o modo de janela de amostragem.

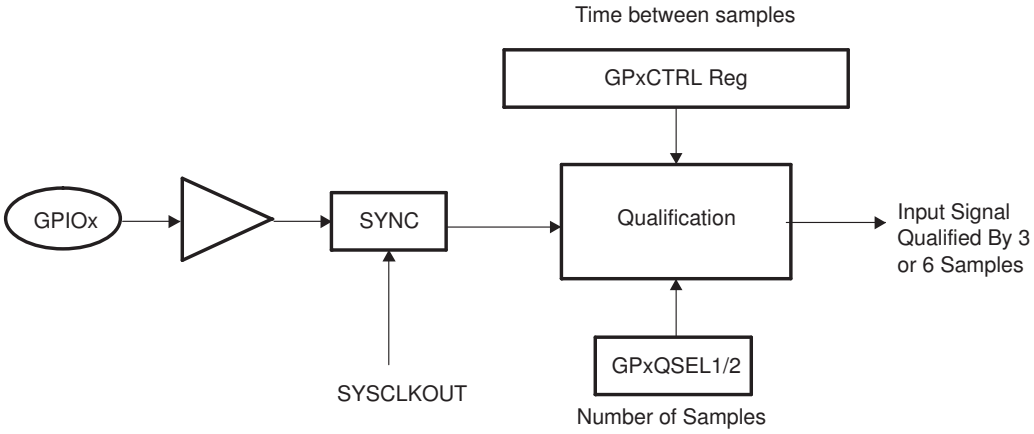


Figura 2 – Qualificação de entrada através da janela de amostragem.

Os três modos listados são configurados para cada pino através dos quatro registradores GPxQSELn (x = {A, B} e n = {1, 2}). A porta C não possui qualificação. Um pino

é configurado por 2 bits correspondentes de um desses registradores, cuja configuração é mostrada na Tabela 1.

Tabela 1 – Modos de sincronização

GPxQSELn	modo de operação
[0,0]	síncrono com o relógio do sistema
[0,1]	qualificação com 3 amostras
[1,0]	qualificação com 6 amostras
[1,1]	assíncrono

O período de amostragem é definido para grupos de oito pinos através dos dois registradores GPxCTRL (x = {A, B}). Cada registrador possui quatro campos de 8 bits cada, denominados QUALPRDn (n = {0, 1, 2, 3}). Mais detalhes desse registrador são mostrados na Figura 21 (ver anexo).

A configuração do período de amostragem é a seguinte:

$$\text{Período de amostragem} = \begin{cases} T_{\text{SYSCLKOUT}}, & \text{se QUALPRDn} = 0 \\ 2 \cdot \text{QUALPRDn} \cdot T_{\text{SYSCLKOUT}}, & \text{se QUALPRDn} > 0 \end{cases}$$

em que $T_{\text{SYSCLKOUT}}$ é o período de tempo de SYSCLKOUT.

A cada pino do GPIO está associado um resistor de *pull-up*, cujo objetivo é evitar problemas de flutuação. A configuração dessa funcionalidade é feita através do registrador GPxPUD (x = {A, B, C}). Cada bit desse registrador ativa (quando igual a 0) ou desativa (quando igual a 1) o *pull-up*. Além disso, nos pinos que podem funcionar como saída PWM (GPIO0-GPIO11) essa funcionalidade é desabilitada por padrão.

2.2 Módulo ADC

O módulo ADC possui 16 canais, e pode ser configurado como dois módulos de 8 canais independentes, ou de 16 canais, quando esses dois módulos são cascateados. Apesar dos múltiplos canais, há apenas um conversor analógico-digital, de 12 bits, conforme o diagrama de blocos da Figura 3.

É importante ressaltar que a tensão analógica deve estar no intervalo de 0 à 3 V.

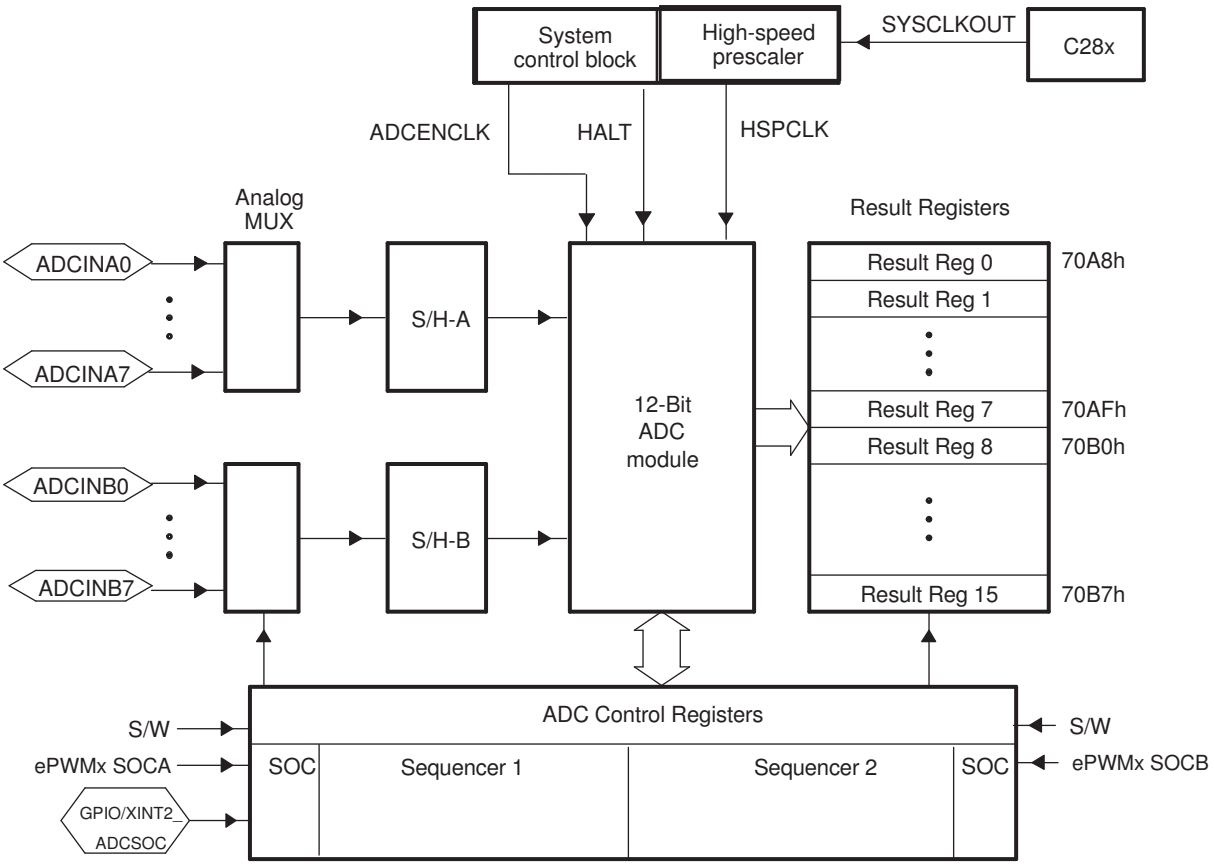


Figura 3 – Registradores do módulo ADC.

O valor digital resultante da conversão é calculado da seguinte forma:

$$\text{Valor Digital} = \begin{cases} 0 & \text{se } v_a \leq 0\text{V} \\ 4096(v_a - \text{ADCLO})/3 & \text{se } 0\text{V} < v_a < 3\text{V} \\ 4095 & \text{se } v_a \geq 3\text{V} \end{cases}$$

em que v_a é a tensão analógica. Valores fracionários são truncados.

O sequenciador do ADC consiste em dois sequenciadores de 8 estados independentes (SEQ1 e SEQ2) que podem ser cascateados, formando um sequenciador de 16 estados (SEQ). O termo "estado" refere-se ao número de auto conversões que podem ser realizadas pelo sequenciador. Em ambos os casos, o ADC pode realizar séries de auto conversão, ou seja, uma vez recebido um pedido de SOC (*Start-of-Conversion*), o ADC pode realizar múltiplas conversões automaticamente.

Em cada conversão, qualquer um dos 16 canais pode ser selecionado através do MUX analógico. Após a conversão, o valor digital obtido é armazenado no registrador de resultados apropriado (ADCRESULTn, n = {0, 1, 2, ..., 15}). O primeiro resultado é armazenado em ADCRESULT0, o segundo, em ADCRESULT1, e assim por diante.

O ADC pode operar em dois modos de amostragem: sequencial ou simultâneo. Em

ambos os casos, o campo de bits CONVxx (x = {00, 01, 02, ..., 15}) define o pino (ou pares de pinos, para o segundo caso) a serem amostrados e convertidos.

No modo sequencial, todos os quatro bits de CONVxx definem o pino de entrada. O Bit Mais Significativo (MSB) define qual dos dois blocos *Sample and Hold* (S/H) será associado ao pino, e os três Bits Menos Significativos (LSB) definem o *offset* (e.g., o valor 0111b seleciona o pino de entrada ADCINA7, e 1101b seleciona o pino ADCINB5).

No modo de amostragem simultânea, o MSB é descartado e a amostragem é realizada simultaneamente para os dois pinos de mesmo *offset* (e.g., se CONVxx contém o valor 0011b, os pinos ADCINA3 e ADCINB3 são amostrados por S/H-A e S/H-B, respectivamente). A conversão é realizada inicialmente para a tensão em S/H-A, e em seguida, para S/H-B. O resultado da primeira conversão é armazenado no registrador ADCRESULTn atual, e o da segunda, no próximo registrador ADCRESULTn, e o ponteiro de registrador de resultado é então incrementado em 2 (uma vez que ocorreram duas conversões).

No modo de amostragem sequencial, caso ocorram pedidos de conversão (SOC) de um dos sequenciadores enquanto o conversor A/D estiver ocupado, estes entrarão em fila, e o conversor atenderá prioritariamente SEQ1.

2.2.1 Modo Auto Sequencial Ininterrupto

O modo descrito nesta seção, cujo fluxograma encontra-se na Figura 4, se aplica aos dois sequenciadores, SEQ1 e SEQ2. Nesse modo de operação, cada um dos sequenciadores pode realizar até 8 conversões em uma única sessão, e até 16 caso sejam cascateados. O resultado de cada conversão é armazenado em um dos oito registradores ADCRESULTn (n = {0, 1, 2, ..., 7} para SEQ1 e n = {8, 9, 10,..., 15} para SEQ2), que são preenchidos em ordem crescente de endereço.

O número de conversões em uma única sequência é controlado por MAX_CONVn (n = {1, 2}), que é um campo de bits do registrador ADCMAXCONV. Esse campo possui 3 bits (ou 4 bits caso haja cascateamento), e, no início de uma sessão de auto sequência, seu valor é carregado automaticamente no contador de sequenciamento (SEQ_CNTR[3:0]), no registrador de status de auto sequência (ADCASEQSR).

A partir do valor inicial, SEQ_CNTR tem seu valor decrementado enquanto que o sequenciador inicia do estado em CONV00 e continua sequencialmente (CONV01, CONV02, e assim sucessivamente) até que SEQ_CNTR alcance o valor nulo. O número de conversões concluídas durante uma sessão de auto sequencia é igual a um mais o valor de MAX_CONVn.

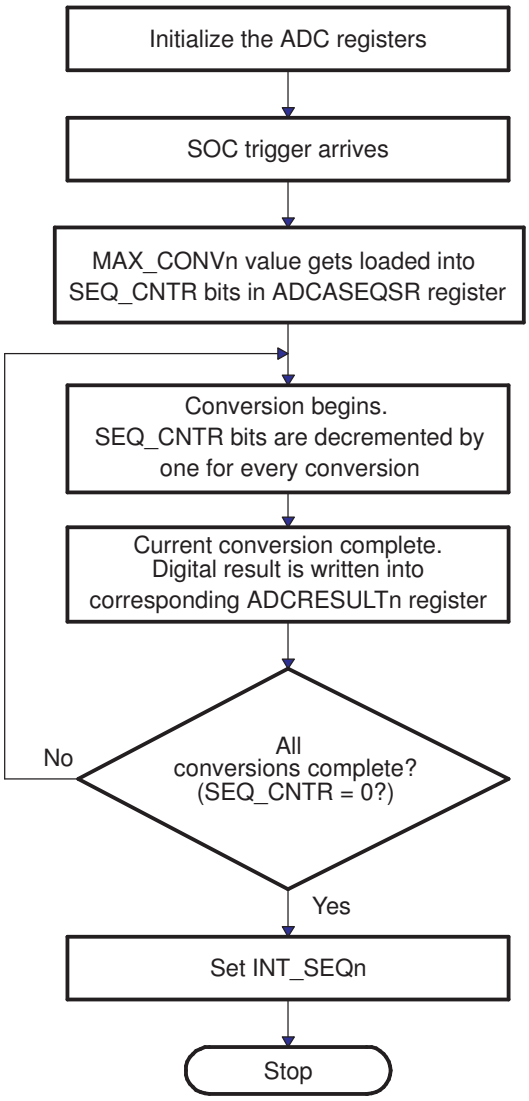


Figura 4 – Fluxograma para o modo auto sequencial ininterrupto.

2.2.2 Gatilhamento

Cada sequenciador possui entradas de gatilho, que podem ser habilitadas ou desabilitadas. Na Tabela 2, estão listados os tipos de gatilho para SEQ1, SEQ2 e SEQ (quando há cascateamento de SEQ1 e SEQ2). Observa-se que o sequenciador 1 pode ser ativado no programa, ou pelo módulo PWM, ou através da ocorrência de uma interrupção. Para o sequenciador 2, o gatilho ocorre apenas via software ou via PWM. No caso do cascateamento, ocorre a união dessas possibilidades.

É importante destacar que, uma vez gatilhado, o sequenciador não pode ser parado ou desligado. O programa deve esperar até o fim da sequência, ou iniciar um *reset* do sequenciador, que o traz ao estado inicial.

Tabela 2 – Possibilidades de gatilhamento.

SEQ1	SEQ2	SEQ
SOC via software	SOC via software	SOC via software
ePWMx SOCA	ePWMx SOCB	ePWMx SOCA
XINT2_ADCSOC		ePWMx SOCB
		XINT2_ADCSOC

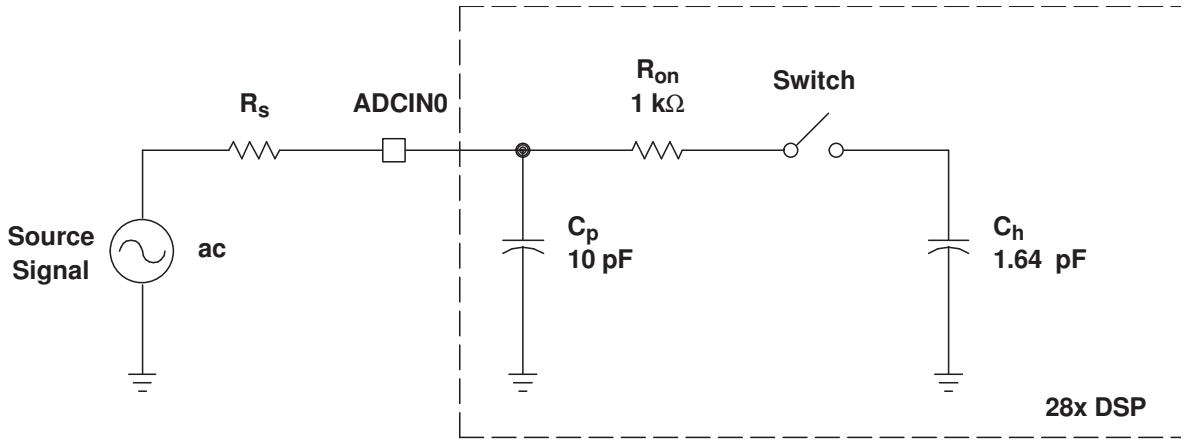
2.2.3 Temporização

O módulo ADC possui vários estágios de escalonamento, que permitem várias opções de frequência de operação. Essas opções envolvem tanto a operação de todo o módulo quanto especificamente o bloco S/H. Esse último caso é importante quando a fonte possui impedância relativamente alta, necessitando de um tempo maior de amostragem para que os valores lidos sejam consistentes.

Na Figura 5, é mostrado o modelo de impedâncias para uma entrada analógica do ADC. Ao ser acionada, a chave (*Switch*) é fechada, e o carregamento do capacitor C_h é iniciado. Para que os valores lidos pelo ADC sejam consistentes, é preciso que a tensão nesse capacitor se estabilize. Quanto maior a resistência da fonte, R_s, maior a constante de tempo do circuito RL formado, e maior o tempo necessário para que o regime permanente seja atingido.

Na Figura 6 é mostrado o escalonador correspondente aos blocos S/H. O relógio dos periféricos, HSPCLK, é dividido pelo valor contido no campo de bits ADCCLKPS[3:0] do registrador ADCTRL3. Além disso, ainda é possível uma divisão por 2, configurável através do bit CPS do registrador ADCTRL1. Assim, é obtida a frequência desejada para o módulo ADC, denominada ADCCLK.

Ainda na Figura 6, é mostrado o bloco responsável por gerar os pulsos que controlam o tempo de amostragem (*SOC pulse generator*). A duração dos pulsos é determinada pelo valor do campo de bits ACQ_PS[3:0], do registrador ADCTRL1, e pode ser modificada de forma a corrigir influências da impedância da fonte a ser amostrada.



Typical Values of the Input Circuit Components:

Switch Resistance (R_{on}):	**1 kΩ**
Sampling Capacitor (C_h):	**1.64 pF**
Parasitic Capacitance (C_p):	**10 pF**
Source Resistance (R_s):	**50 Ω**

Figura 5 – Modelo de impedância da entrada analógica do ADC.

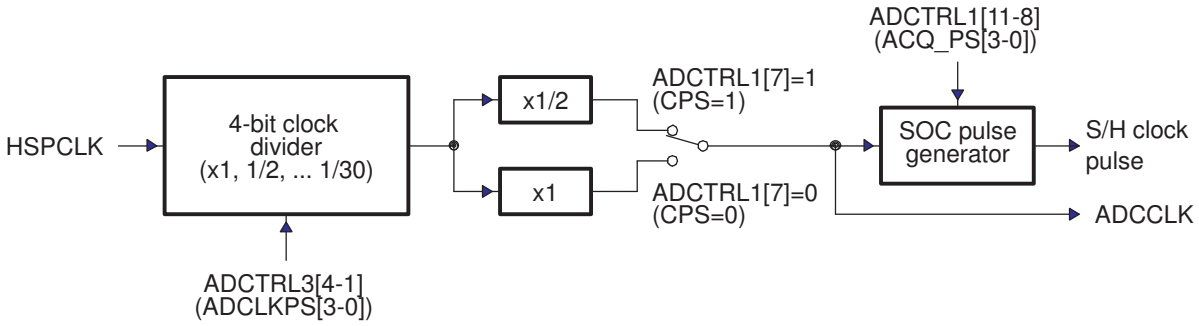


Figura 6 – Temporizador do módulo ADC e do S/H.

3 O framework Qt

3.1 O que é Qt

Basicamente, Qt é um *framework*. Para melhor entendimento sobre Qt, é importante, antes de tudo, entender o que significa o termo *framework*. Diferentemente de uma biblioteca de classes, onde cada classe é única e independente uma das outras, um *framework* utiliza um conjunto de classes flexível e extensível para permitir a construção de várias aplicações com pouco esforço, sendo apenas necessário especificar as particularidades de cada aplicação.

Em outras palavras, em um *framework*, as classes cooperam de forma a resolver um conjunto de problemas semelhantes.

O Qt suporta várias plataformas: Windows, WinCE, iOS, Mac OS X, Linux/X11, dentre outras. Além disso, é constituído por várias ferramentas, a exemplo do *Qt designer*, que permite a elaboração da interface gráfica do usuário (GUI). A ferramenta *qmake* automatiza o processo de *build* do projeto. O principal componente do Qt é o conjunto de bibliotecas, escritas nativamente em C++. Apesar disso, o Qt pode ser usado com outras linguagens de programação, uma vez que possui bibliotecas para esse fim.

De acordo com a funcionalidade, as classes do Qt são agrupadas em módulos dentre os quais o principal é o *QtCore*. Como o próprio nome diz, esse módulo é o cerne, e nele estão funcionalidades não gráficas. Todos os outros módulos dependem desse. O *QWidget* provê um conjunto de elementos para criação de interfaces gráficas clássicas estilo *desktop*, tais quais *combo boxes*, *check boxes*, *line edits*, dentre outros.

Um importante recurso que o diferencia dos demais *frameworks* é o de Sinais e *Slots*, descrito na próxima seção. Essa funcionalidade desempenhou um papel fundamental na elaboração do aplicativo desenvolvido neste trabalho.

3.2 Sinais e Slots

Sinais e Slots são utilizados para comunicação entre objetos. Na programação de GUIs, quando o usuário modifica um *Widget*, normalmente outro *Widget* é notificado. Isso ocorre, por exemplo, quando o usuário clica em um botão para fechar o aplicativo, e então é chamada a função adequada para que o programa seja finalizado.

Frameworks mais antigos realizam esse tipo de comunicação através de *callbacks*. Um *callback* é um ponteiro para uma função. Assim, quando se deseja a notificação sobre

algum evento, é passado à função de processamento um ponteiro para outra função. A função de processamento então chama o *callback* quando for apropriado. Essa abordagem possui dois problemas:

1. *callbacks* são de tipo não seguro. Não há certeza de que a função de processamento irá chamar o *callback* com os argumentos corretos, e
2. o *callback* é fortemente acoplado à função de processamento, uma vez que a esta deve "saber" qual *callback* chamar.

No mecanismo de Sinais e *Slots*, um sinal é emitido quando um evento particular ocorre. Os *widgets* do Qt possuem muitos sinais pré-definidos, mas o programador pode criar sinais personalizados. O *slot* é a função chamada em resposta a um sinal particular. Assim como no caso dos sinais, podem ser definidos novos *slots* personalizados.

Esse recurso é de tipo seguro: a assinatura de um sinal deve coincidir com a assinatura do *slot* receptor. Além disso, há fraco acoplamento: o *slot* não "sabe" a qual ou quais sinais está conectado. Adicionalmente, um sinal pode ser conectado a vários *slots*, enquanto que este pode ser conectado a vários sinais.

4 O Aplicativo Desenvolvido

A partir da descrição do GPIO, é possível elaborar um fluxograma de configuração dos pinos, que é ilustrado na Figura 7. O detalhamento da configuração da qualificação (quando aplicável) é mostrado na Figura 8.

O procedimento adotado para a elaboração do fluxograma consistiu em:

1. Leitura do Manual de Dados (5), que contém de forma sucinta a descrição do funcionamento de várias partes do DSP;
2. Leitura do Guia de Referência correspondente ao módulo em estudo (para o caso do GPIO, (1));
3. Identificação dos registradores envolvidos no processo de configuração, bem como de suas funções;
4. Análise da relação entre a configuração de cada registrador e a configuração dos demais, de forma a encontrar uma ordem coerente de configuração dos mesmos (e.g., não faz sentido configurar qualificação em um pino definido como saída).

É importante resumir os valores dos registradores após um *reset*, ou seja, os valores correspondentes à configuração padrão:

- *pull-up* desabilitado nos pinos que poder ser conectados ao PWM, e habilitado nos demais;
- pinos configurados como de propósito geral (GPIO);
- pinos configurados como entrada digital;
- sincronização com o relógio do sistema (SYSCLKOUT).

A partir dos diagramas das Figuras 7 e 8, foi elaborada a interface gráfica do usuário, que é mostrada na Figura 9.

O aplicativo possui centenas de *widgets*, e alguns desses elementos precisam ser notificados quando determinado *widget* é modificado. Isso é necessário, uma vez que certas opções de configuração só podem ser disponibilizadas sob certas circunstâncias (e.g., não faz sentido que a opção de sincronização esteja disponível quando um pino for selecionado como saída).

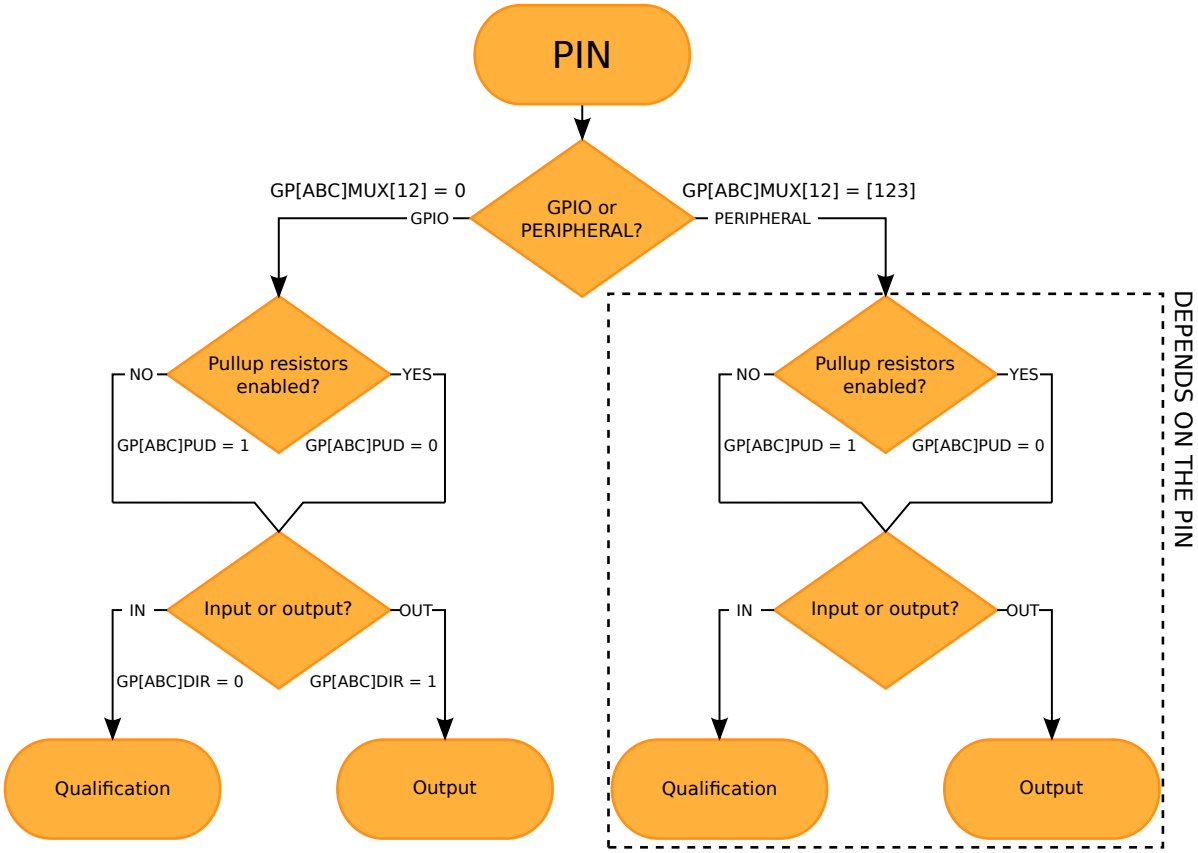


Figura 7 – Fluxograma geral de configuração do GPIO.

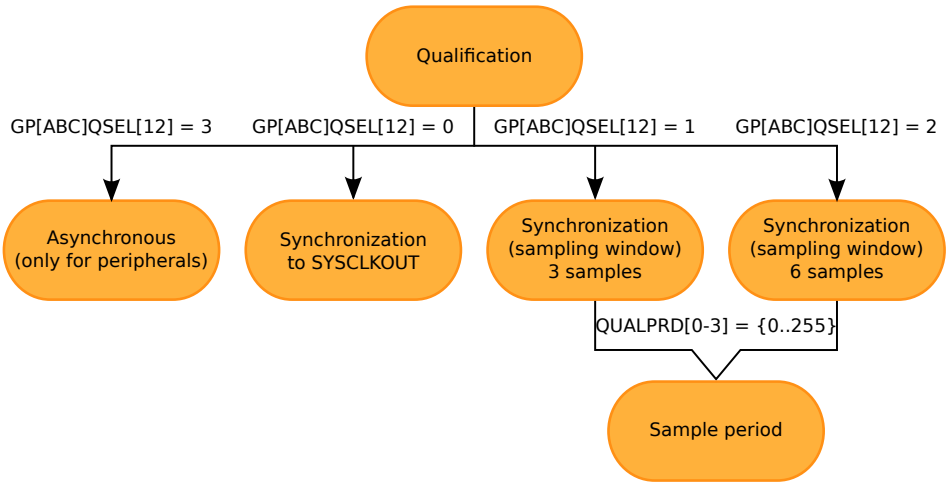


Figura 8 – Fluxograma de configuração para a qualificação, quando esta for aplicável.

Dessa forma, vários sinais podem ser emitidos, e cada um tem efeito em um elemento específico. Uma forma de resolver esse problema seria interconectar os sinais emitidos pelos mais de 300 *widgets* a centenas de *slots*. Uma solução mais prática se dá através do uso da classe *QSignalMapper*, que permite o agrupamento de sinais, cada um diferen-

Figura 9 – Interface do aplicativo desenvolvido.

ciado através de um objeto do tipo *QObject*, que pode ser uma *QString*, um índice (tipo *int*), dentre outros. Assim, é possível conectar todo o grupo de sinais a apenas uma função *slot*.

Na Figura 10 é ilustrado um exemplo de aplicação do *QSignalMapper*, em que há dez botões, numerados de 0 a 9, agrupados em um array, denominado *buttons*. Um clique em um botão (sinal) é conectado ao seu respectivo *slot* no *QSignalMapper*. Se houver mapeamento, o sinal mapeado é emitido com o índice correspondente ao botão clicado.

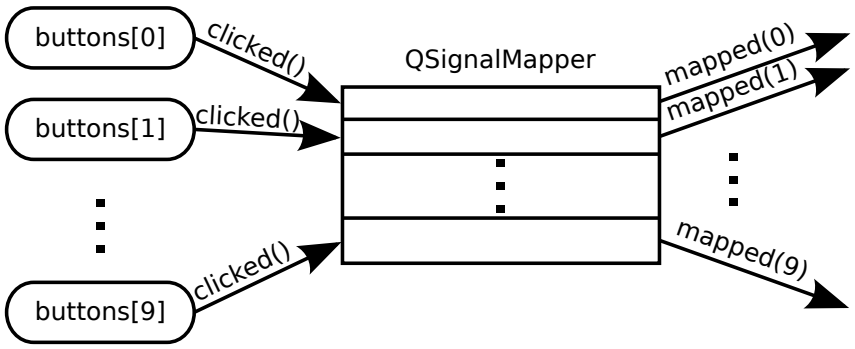


Figura 10 – Exemplo de utilização do mecanismo de mapeamento de sinais.

Uma vez realizada a configuração dos pinos, o usuário clica na aba *Save All*, clica no botão *Save*, e então é gerado um arquivo com extensão *.c* que contém a definição de uma função de configuração. Para a configuração mostrada na Figura 9, é gerado um arquivo com o conteúdo mostrado no Anexo B.

Esse arquivo deve ser incluído em um projeto compilável do pacote de exemplos fornecido pela Texas Instruments, que pode ser baixado no próprio site da empresa. Na

Figura 11, é indicado o arquivo dentro do qual a função `void Gpio_setup()` deve ser chamada.

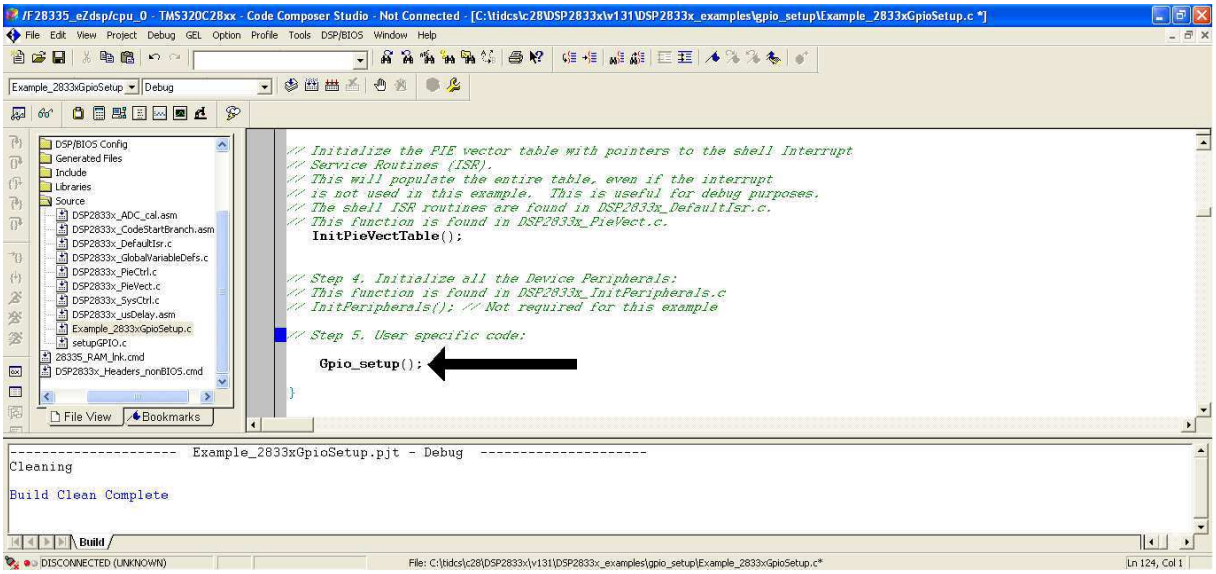


Figura 11 – Função de configuração gerada pelo assistente em um exemplo de projeto fornecido pela Texas Instruments.

5 Considerações Finais

Neste trabalho foi apresentado o desenvolvimento de um assistente de configuração para o DSP TMS320F28335, facilitando consideravelmente a configuração dos pinos desse dispositivo. Essa é normalmente uma atividade penosa, sujeita a erros do usuário, e que exige um tempo considerável.

Os conhecimentos adquiridos durante o curso, principalmente nas disciplinas que tratam sobre programação e arquitetura de computadores, foram imprescindíveis para a realização deste trabalho.

Além disso, o conhecimento adquirido na elaboração desse aplicativo resultou em um grande aprendizado sobre o próprio DSP, bem como sobre Qt, que mostrou-se uma ferramenta bastante poderosa e prática para a elaboração da interface de usuário.

Sugestões para Trabalhos Futuros

Como trabalho futuro, serão implementadas as configurações para os demais módulos do DSP, com maior prioridade aos módulos ADC e PWM.

Pretende-se também desenvolver um *plug-in*, utilizando a plataforma Eclipse, que permita a configuração dos módulos mencionados, uma vez que a versão mais atual do compilador *Code Composer Studio* (da Texas Instruments) é baseada nessa plataforma.

Referências

1 TEXAS INSTRUMENTS. *TMS320x2833x,2823x System Control and Interrupts - Reference Guide*. [S.l.], 2007. Citado na página 13.

2 BLANCHETTE, J. *Mapping Many Signals to One*. <http://doc.qt.digia.com/qq/qq10-signalmapper.html>. Nenhuma citação no texto.

3 SIGNALS & Slots. <http://qt-project.org/doc/qt-5/signalsandslots.html>. Nenhuma citação no texto.

4 TEXAS INSTRUMENTS. *TMS320x2833x Analog-to-Digital Converter Module - Reference Guide*. [S.l.], 2007. Nenhuma citação no texto.

5 TEXAS INSTRUMENTS. *Data Manual*. [S.l.], 2007. Citado na página 13.

6 SAUVé, J. P. *Frameworks*. <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>. Nenhuma citação no texto.

Anexos

ANEXO A – Detalhamento dos bits de registradores.

REGISTER BITS			PERIPHERAL SELECTION			
GPADIR GPADAT GPASET GPACLR GPATOGGLE		**GPAMUX1 GPAQSEL1**	**GPIOx GPAMUX1 = 0,0**	**PER1 GPAMUX1 = 0, 1**	**PER2 GPAMUX1 = 1, 0**	**PER3 GPAMUX1 = 1, 1**
QUALPRD0	0	1, 0	GPIO0 (I/O)	EPWM1A (O)	Reserved	Reserved
	1	3, 2	GPIO1 (I/O)	EPWM1B (O)	ECAP6 (I/O)	MFSRB (I/O)
	2	5, 4	GPIO2 (I/O)	EPWM2A (O)	Reserved	Reserved
	3	7, 6	GPIO3 (I/O)	EPWM2B (O)	ECAP5 (I/O)	MCLKRB (I/O)
	4	9, 8	GPIO4 (I/O)	EPWM3A (O)	Reserved	Reserved
	5	11, 10	GPIO5 (I/O)	EPWM3B (O)	MFSRA (I/O)	ECAP1 (I/O)
	6	13, 12	GPIO6 (I/O)	EPWM4A (O)	EPWMSYNCI (I)	EPWMSYNCO (O)
	7	15, 14	GPIO7 (I/O)	EPWM4B (O)	MCLKRA (I/O)	ECAP2 (I/O)
QUALPRD1	8	17, 16	GPIO8 (I/O)	EPWM5A (O)	CANTXB (O)	$\overline{\text{ADCSOCAO}}$ (O)
	9	19, 18	GPIO9 (I/O)	EPWM5B (O)	SCITXDB (O)	ECAP3 (I/O)
	10	21, 20	GPIO10 (I/O)	EPWM6A (O)	CANRXB (I)	$\overline{\text{ADCSOCBO}}$ (O)
	11	23, 22	GPIO11 (I/O)	EPWM6B (O)	SCIRXDB (I)	ECAP4 (I/O)
	12	25, 24	GPIO12 (I/O)	$\overline{\text{TZ1}}$ (I)	CANTXB (O)	MDXB (O)
	13	27, 26	GPIO13 (I/O)	$\overline{\text{TZ2}}$ (I)	CANRXB (I)	MDRB (I)
	14	29, 28	GPIO14 (I/O)	$\overline{\text{TZ3}}$ (I)/$\overline{\text{XHOLD}}$ (I)	SCITXDB (O)	MCLKXB (I/O)
	15	31, 30	GPIO15 (I/O)	$\overline{\text{TZ4}}$ (I)/$\overline{\text{XHOLDA}}$ (O)	SCIRXDB (I)	MFSXB (I/O)
		GPAMUX2 GPAQSEL2	**GPAMUX2 = 0, 0**	**GPAMUX2 = 0, 1**	**GPAMUX2 = 1, 0**	**GPAMUX2 = 1, 1**
QUALPRD2	16	1, 0	GPIO16 (I/O)	SPISIMOA (I/O)	CANTXB (O)	$\overline{\text{TZ5}}$ (I)
	17	3, 2	GPIO17 (I/O)	SPISOMIA (I/O)	CANRXB (I)	$\overline{\text{TZ6}}$ (I)
	18	5, 4	GPIO18 (I/O)	SPICLKA (I/O)	SCITXDB (O)	CANRXA (I)
	19	7, 6	GPIO19 (I/O)	$\overline{\text{SPISTEA}}$ (I/O)	SCIRXDB (I)	CANTXA (O)
	20	9, 8	GPIO20 (I/O)	EQEP1A (I)	MDXA (O)	CANTXB (O)
	21	11, 10	GPIO21 (I/O)	EQEP1B (I)	MDRA (I)	CANRXB (I)
	22	13, 12	GPIO22 (I/O)	EQEP1S (I/O)	MCLKXA (I/O)	SCITXDB (O)
	23	15, 14	GPIO23 (I/O)	EQEP1I (I/O)	MFSXA (I/O)	SCIRXDB (I)
QUALPRD3	24	17, 16	GPIO24 (I/O)	ECAP1 (I/O)	EQEP2A (I)	MDXB (O)
	25	19, 18	GPIO25 (I/O)	ECAP2 (I/O)	EQEP2B (I)	MDRB (I)
	26	21, 20	GPIO26 (I/O)	ECAP3 (I/O)	EQEP2I (I/O)	MCLKXB (I/O)
	27	23, 22	GPIO27 (I/O)	ECAP4 (I/O)	EQEP2S (I/O)	MFSXB (I/O)
	28	25, 24	GPIO28 (I/O)	SCIRXDA (I)	$\overline{\text{XZCS6}}$ (O)	
	29	27, 26	GPIO29 (I/O)	SCITXDA (O)	XA19 (O)	
	30	29, 28	GPIO30 (I/O)	CANRXA (I)	XA18 (O)	
	31	31, 30	GPIO31 (I/O)	CANTXA (O)	XA17 (O)	

Figura 12 – Matriz de seleção de periféricos para o MUX GPIO-A

REGISTER BITS			PERIPHERAL SELECTION			
GPBDIR GPBDAT GPBSET GPBCLR GPBTOGGLE		**GPBMUX1 GPBQSEL1**	**GPIOx GPBMUX1 = 0, 0**	**PER1 GPBMUX1 = 0, 1**	**PER2 GPBMUX1 = 1, 0**	**PER3 GPBMUX1 = 1, 1**
QUALPRD0	0	1, 0	GPIO32 (I/O)	SDAA (I/OC)(1)	EPWMSYNCI (I)	$\overline{\text{ADCSOCAO}}$ (O)
	1	3, 2	GPIO33 (I/O)	SCLA (I/OC)(1)	EPWMSYNCO (O)	$\overline{\text{ADCSOCBO}}$ (O)
	2	5, 4	GPIO34 (I/O)	ECAP1 (I/O)	XREADY (I)	
	3	7, 6	GPIO35 (I/O)	SCITXDA (O)	XR/$\overline{\text{W}}$ (O)	
	4	9, 8	GPIO36 (I/O)	SCIRXDA (I)	$\overline{\text{XZCS0}}$ (O)	
	5	11, 10	GPIO37 (I/O)	ECAP2 (I/O)	$\overline{\text{XZCS7}}$ (O)	
	6	13, 12	GPIO38 (I/O)	Reserved	$\overline{\text{XWE0}}$ (O)	
	7	15, 14	GPIO39 (I/O)		XA16 (O)	
QUALPRD1	8	17, 16	GPIO40 (I/O)		XA0/$\overline{\text{XWE1}}$ (O)	
	9	19, 18	GPIO41 (I/O)		XA1 (O)	
	10	21, 20	GPIO42 (I/O)		XA2 (O)	
	11	23, 22	GPIO43 (I/O)		XA3 (O)	
	12	25, 24	GPIO44 (I/O)		XA4 (O)	
	13	27, 26	GPIO45 (I/O)		XA5 (O)	
	14	29, 28	GPIO46 (I/O)		XA6 (O)	
	15	31, 30	GPIO47 (I/O)		XA7 (O)	
		GPBMUX2 GPBQSEL2	**GPBMUX2 = 0, 0**	**GPBMUX2 = 0, 1**	**GPBMUX2 = 1, 0**	**GPBMUX2 = 1, 1**
QUALPRD2	16	1, 0	GPIO48 (I/O)	ECAP5 (I/O)	XD31 (I/O)	
	17	3, 2	GPIO49 (I/O)	ECAP6 (I/O)	XD30 (I/O)	
	18	5, 4	GPIO50 (I/O)	EQEP1A (I)	XD29 (I/O)	
	19	7, 6	GPIO51 (I/O)	EQEP1B (I)	XD28 (I/O)	
	20	9, 8	GPIO52 (I/O)	EQEP1S (I/O)	XD27 (I/O)	
	21	11, 10	GPIO53 (I/O)	EQEP1I (I/O)	XD26 (I/O)	
	22	13, 12	GPIO54 (I/O)	SPISIMOA (I/O)	XD25 (I/O)	
	23	15, 14	GPIO55 (I/O)	SPISOMIA (I/O)	XD24 (I/O)	
QUALPRD3	24	17, 16	GPIO56 (I/O)	SPICLKA (I/O)	XD23 (I/O)	
	25	19, 18	GPIO57 (I/O)	$\overline{\text{SPISTEA}}$ (I/O)	XD22 (I/O)	
	26	21, 20	GPIO58 (I/O)	MCLKRA (I/O)	XD21 (I/O)	
	27	23, 22	GPIO59 (I/O)	MFSRA (I/O)	XD20 (I/O)	
	28	25, 24	GPIO60 (I/O)	MCLKRB (I/O)	XD19 (I/O)	
	29	27, 26	GPIO61 (I/O)	MFSRB (I/O)	XD18 (I/O)	
	30	29, 28	GPIO62 (I/O)	SCIRXDC (I)	XD17 (I/O)	
	31	31, 30	GPIO63 (I/O)	SCITXDC (O)	XD16 (I/O)	

(1) Open drain

Figura 13 – Matriz de seleção de periféricos para o MUX GPIO-B

REGISTER BITS			PERIPHERAL SELECTION	
GPCDIR GPCDAT GPCSET GPCCLR GPCTOGGLE		**GPCMUX1**	**GPIOx or PER1 GPCMUX1 = 0, 0 or 0, 1**	**PER2 or PER3 GPCMUX1 = 1, 0 or 1, 1**
no qual	0	1, 0	GPIO64 (I/O)	XD15 (I/O)
	1	3, 2	GPIO65 (I/O)	XD14 (I/O)
	2	5, 4	GPIO66 (I/O)	XD13 (I/O)
	3	7, 6	GPIO67 (I/O)	XD12 (I/O)
	4	9, 8	GPIO68 (I/O)	XD11 (I/O)
	5	11, 10	GPIO69 (I/O)	XD10 (I/O)
	6	13, 12	GPIO70 (I/O)	XD9 (I/O)
	7	15, 14	GPIO71 (I/O)	XD8 (I/O)
no qual	8	17, 16	GPIO72 (I/O)	XD7 (I/O)
	9	19, 18	GPIO73 (I/O)	XD6 (I/O)
	10	21, 20	GPIO74 (I/O)	XD5 (I/O)
	11	23, 22	GPIO75 (I/O)	XD4 (I/O)
	12	25, 24	GPIO76 (I/O)	XD3 (I/O)
	13	27, 26	GPIO77 (I/O)	XD2 (I/O)
	14	29, 28	GPIO78 (I/O)	XD1 (I/O)
	15	31, 30	GPIO79 (I/O)	XD0 (I/O)
		GPCMUX2	**GPCMUX2 = 0, 0 or 0, 1**	**GPCMUX2 = 1, 0 or 1, 1**
no qual	16	1, 0	GPIO80 (I/O)	XA8 (O)
	17	3, 2	GPIO81 (I/O)	XA9 (O)
	18	5, 4	GPIO82 (I/O)	XA10 (O)
	19	7, 6	GPIO83 (I/O)	XA11 (O)
	20	9, 8	GPIO84 (I/O)	XA12 (O)
	21	11, 10	GPIO85 (I/O)	XA13 (O)
	22	13, 12	GPIO86 (I/O)	XA14 (O)
	23	15, 14	GPIO87 (I/O)	XA15 (O)

Figura 14 – Matriz de seleção de periféricos para o MUX GPIO-C

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND- R/W = Read/Write; R = Read only; -*n* = value after reset

Figura 15 – Registrador do MUX 1 do GPIO-A (GPAMUX1).

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	**13 12**	**11 10**	**9 8**	**7 6**	**5 4**	**3 2**	**1 0**
GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figura 16 – Registrador do MUX 2 do GPIO-A (GPAMUX2).

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO47	GPIO46	GPIO45	GPIO44	GPIO43	GPIO42	GPIO41	GPIO40
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO39	GPIO38	GPIO37	GPIO36	GPIO35	GPIO34	GPIO33	GPIO32
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figura 17 – Registrador do MUX 1 do GPIO-B (GPBMUX1).

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO63	GPIO62	GPIO61	GPIO60	GPIO59	GPIO58	GPIO57	GPIO56
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO55	GPIO54	GPIO53	GPIO52	GPIO51	GPIO50	GPIO49	GPIO48
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figura 18 – Registrador do MUX 2 do GPIO-B (GPBMUX2).

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
GPIO79	GPIO78	GPIO77	GPIO76	GPIO75	GPIO74	GPIO73	GPIO72
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO71	GPIO70	GPIO69	GPIO68	GPIO67	GPIO66	GPIO65	GPIO64
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figura 19 – Registrador do MUX 1 do GPIO-C (GPCMUX1).

31 16
Reserved
R-0

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
GPIO87	GPIO86	GPIO85	GPIO84	GPIO83	GPIO82	GPIO81	GPIO80
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figura 20 – Registrador do MUX 2 do GPIO-C (GPCMUX2).

31 24	23 16
QUALPRD3	QUALPRD2
R/W-0	R/W-0
15 8	7 0
QUALPRD1	QUALPRD0
R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figura 21 – Registrador de controle de qualificação GPACTRL. Idem para GPBCTRL.

ANEXO B – Código gerado pelo aplicativo.

```
#include "DSP28x_Project.h"

 void Gpio_setup()
{
    EALLOW;

    GpioCtrlRegs.GPAPUD.bit.GPIO0 = 1;       // pullup disabled
    GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0;      // general-purpose
    GpioCtrlRegs.GPADIR.bit.GPIO0 = 0;       // digital input
    GpioCtrlRegs.GPAQSEL1.bit.GPIO0 = 0;
    GpioCtrlRegs.GPACTRL.bit.QUALPRD0 = 6;

    GpioCtrlRegs.GPAPUD.bit.GPIO1 = 1;       // pullup disabled
    GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 1;
    GpioCtrlRegs.GPAQSEL1.bit.GPIO1 = 3;     // async

    GpioCtrlRegs.GPAPUD.bit.GPIO2 = 1;       // pullup disabled
    GpioCtrlRegs.GPAMUX1.bit.GPIO2 = 1;
    GpioCtrlRegs.GPAQSEL1.bit.GPIO2 = 3;     // async

    GpioCtrlRegs.GPAPUD.bit.GPIO3 = 1;       // pullup disabled
    GpioCtrlRegs.GPAMUX1.bit.GPIO3 = 2;
    GpioCtrlRegs.GPAQSEL1.bit.GPIO3 = 3;     // async

    GpioCtrlRegs.GPAPUD.bit.GPIO4 = 1;       // pullup disabled
    GpioCtrlRegs.GPAMUX1.bit.GPIO4 = 0;      // general-purpose
    GpioCtrlRegs.GPADIR.bit.GPIO4 = 0;       // digital input
    GpioCtrlRegs.GPAQSEL1.bit.GPIO4 = 1;

    GpioCtrlRegs.GPAPUD.bit.GPIO5 = 1;       // pullup disabled
    GpioCtrlRegs.GPAMUX1.bit.GPIO5 = 0;      // general-purpose
    GpioCtrlRegs.GPADIR.bit.GPIO5 = 0;       // digital input
    GpioCtrlRegs.GPAQSEL1.bit.GPIO5 = 0;

    GpioCtrlRegs.GPAPUD.bit.GPIO6 = 1;       // pullup disabled
```

```
GpioCtrlRegs.GPAMUX1.bit.GPIO6 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO6 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO6 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO7 = 1;       // pullup disabled
GpioCtrlRegs.GPAMUX1.bit.GPIO7 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO7 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO7 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO8 = 1;       // pullup disabled
GpioCtrlRegs.GPAMUX1.bit.GPIO8 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO8 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO8 = 1;
GpioCtrlRegs.GPACTRL.bit.QUALPRD1 = 9;

GpioCtrlRegs.GPAPUD.bit.GPIO9 = 1;       // pullup disabled
GpioCtrlRegs.GPAMUX1.bit.GPIO9 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO9 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO9 = 1;

GpioCtrlRegs.GPAPUD.bit.GPIO10 = 1;       // pullup disabled
GpioCtrlRegs.GPAMUX1.bit.GPIO10 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO10 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO10 = 2;

GpioCtrlRegs.GPAPUD.bit.GPIO11 = 1;       // pullup disabled
GpioCtrlRegs.GPAMUX1.bit.GPIO11 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO11 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO11 = 2;

GpioCtrlRegs.GPAPUD.bit.GPIO12 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX1.bit.GPIO12 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO12 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO12 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO13 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX1.bit.GPIO13 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO13 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO13 = 0;
```

```
GpioCtrlRegs.GPAPUD.bit.GPIO14 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX1.bit.GPIO14 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO14 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO14 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO15 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX1.bit.GPIO15 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO15 = 0;       // digital input
GpioCtrlRegs.GPAQSEL1.bit.GPIO15 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO16 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO16 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO16 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO16 = 0;
GpioCtrlRegs.GPACTRL.bit.QUALPRD2 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO17 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO17 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO17 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO17 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO18 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO18 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO18 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO18 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO19 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO19 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO19 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO19 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO20 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO20 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO20 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO20 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO21 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO21 = 0;      // general-purpose
```

```
GpioCtrlRegs.GPADIR.bit.GPIO21 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO21 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO22 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO22 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO22 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO22 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO23 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO23 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO23 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO23 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO24 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO24 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO24 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO24 = 0;
GpioCtrlRegs.GPACTRL.bit.QUALPRD3 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO25 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO25 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO25 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO25 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO26 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO26 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO26 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO26 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO27 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO27 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO27 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO27 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO28 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO28 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO28 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO28 = 0;
```

```
GpioCtrlRegs.GPAPUD.bit.GPIO29 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO29 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO29 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO29 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO30 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO30 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO30 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO30 = 0;

GpioCtrlRegs.GPAPUD.bit.GPIO31 = 0;       // pullup enabled
GpioCtrlRegs.GPAMUX2.bit.GPIO31 = 0;      // general-purpose
GpioCtrlRegs.GPADIR.bit.GPIO31 = 0;       // digital input
GpioCtrlRegs.GPAQSEL2.bit.GPIO31 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO32 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO32 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO32 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO32 = 0;
GpioCtrlRegs.GPBCTRL.bit.QUALPRD0 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO33 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO33 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO33 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO33 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO34 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO34 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO34 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO34 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO35 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO35 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO35 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO35 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO36 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO36 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO36 = 0;       // digital input
```

```
GpioCtrlRegs.GPBQSEL1.bit.GPIO36 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO37 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO37 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO37 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO37 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO38 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO38 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO38 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO38 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO39 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO39 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO39 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO39 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO40 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO40 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO40 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO40 = 0;
GpioCtrlRegs.GPBCTRL.bit.QUALPRD1 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO41 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO41 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO41 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO41 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO42 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO42 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO42 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO42 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO43 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO43 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO43 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO43 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO44 = 0;       // pullup enabled
```

```
GpioCtrlRegs.GPBMUX1.bit.GPIO44 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO44 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO44 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO45 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO45 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO45 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO45 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO46 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO46 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO46 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO46 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO47 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX1.bit.GPIO47 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO47 = 0;       // digital input
GpioCtrlRegs.GPBQSEL1.bit.GPIO47 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO48 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO48 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO48 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO48 = 0;
GpioCtrlRegs.GPBCTRL.bit.QUALPRD2 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO49 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO49 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO49 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO49 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO50 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO50 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO50 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO50 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO51 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO51 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO51 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO51 = 0;
```

```
GpioCtrlRegs.GPBPUD.bit.GPIO52 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO52 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO52 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO52 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO53 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO53 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO53 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO53 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO54 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO54 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO54 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO54 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO55 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO55 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO55 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO55 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO56 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO56 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO56 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO56 = 0;
GpioCtrlRegs.GPBCTRL.bit.QUALPRD3 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO57 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO57 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO57 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO57 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO58 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO58 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO58 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO58 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO59 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO59 = 0;      // general-purpose
```

```
GpioCtrlRegs.GPBDIR.bit.GPIO59 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO59 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO60 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO60 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO60 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO60 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO61 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO61 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO61 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO61 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO62 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO62 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO62 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO62 = 0;

GpioCtrlRegs.GPBPUD.bit.GPIO63 = 0;       // pullup enabled
GpioCtrlRegs.GPBMUX2.bit.GPIO63 = 0;      // general-purpose
GpioCtrlRegs.GPBDIR.bit.GPIO63 = 0;       // digital input
GpioCtrlRegs.GPBQSEL2.bit.GPIO63 = 0;

GpioCtrlRegs.GPCPUD.bit.GPIO64 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO64 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO64 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO65 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO65 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO65 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO66 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO66 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO66 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO67 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO67 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO67 = 0;       // digital input
```

```
GpioCtrlRegs.GPCPUD.bit.GPIO68 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO68 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO68 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO69 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO69 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO69 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO70 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO70 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO70 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO71 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO71 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO71 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO72 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO72 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO72 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO73 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO73 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO73 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO74 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO74 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO74 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO75 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO75 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO75 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO76 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO76 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO76 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO77 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO77 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO77 = 0;       // digital input
```

```
GpioCtrlRegs.GPCPUD.bit.GPIO78 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO78 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO78 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO79 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX1.bit.GPIO79 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO79 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO80 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX2.bit.GPIO80 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO80 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO81 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX2.bit.GPIO81 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO81 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO82 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX2.bit.GPIO82 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO82 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO83 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX2.bit.GPIO83 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO83 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO84 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX2.bit.GPIO84 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO84 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO85 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX2.bit.GPIO85 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO85 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO86 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX2.bit.GPIO86 = 0;      // general-purpose
GpioCtrlRegs.GPCDIR.bit.GPIO86 = 0;       // digital input

GpioCtrlRegs.GPCPUD.bit.GPIO87 = 0;       // pullup enabled
GpioCtrlRegs.GPCMUX2.bit.GPIO87 = 0;      // general-purpose
```

```
    GpioCtrlRegs.GPCDIR.bit.GPIO87 = 0;       // digital input

    EDIS;
}
```