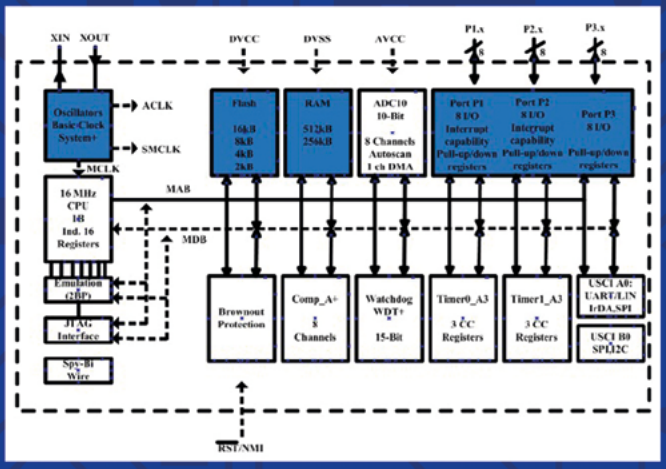
FRANCISCO EVERTON UCHÔA REIS

O MICROCONTROLADOR MSP430G2553

RECURSOS BÁSICOS: MEMÓRIA RAM E FLASH, SISTEMA DE CLOCK E RESET, PINOS E PORTAS COM EXEMPLOS DE UTILIZAÇÃO



MÓDULO I

edufpi

O Microcontrolador MSP430G2553

Francisco Everton Uchôa Reis

O Microcontrolador MSP430G2553

Recursos básicos: Memória RAM e Flash, sistema de clock e reset, pinos e portas com exemplos de utilização

Módulo I

edufpi

Projeto Gráfico. Capa. Diagramação.
Josafá Araújo Procópio

Revisão
Maria do Socorro Baptista Barbosa

FICHA CATALOGRÁFICA
Universidade Federal do Piauí
Biblioteca Comunitária Jornalista Carlos Castello Branco
Divisão de Representação da Informação

R375m Reis, Francisco Everton Uchôa.
O Microcontrolador MSP430G2553 : Recursos básicos : Memória RAM e Flash, sistema de clock e reset, pinos e portas com exemplos de utilização. Módulo I / Francisco Everton Uchôa Reis. - Teresina : EDUFPI, 2025.
92 f.

978-65-5904-436-8

1. Microcontrolador. 2. Memória RAM. 3. Memória Flash. 4. Memória Principal. 5. Registrador. I. Título.

CDD 004

Bibliotecário: Gésio dos Santos Barros - CRB3/1469

Editora da Universidade Federal do Piauí - EDUFPI
Campus Universitário Ministro Petrônio Portella
CEP: 64049-550 - Bairro Ininga - Teresina - PI - Brasil

Associação Brasileira das Editoras Universitárias

PREFÁCIO

O livro foi escrito para abordar, de forma simples e objetiva, o microcontrolador MSP430G2553, de encapsulamento QFN de 32 pinos.

Os recursos básicos, como memória, portas e pinos, são apresentados de forma organizada, e são inclusas algumas boas práticas de uso.

Sempre que possível, são apresentados exemplos de utilização dos recursos internos do microcontrolador. Para fins didáticos, os exemplos consideram a frequência de clock da CPU de 1 MHz ou um valor aproximado.

O software utilizado para os códigos é o Code Composer Studio versão 5.5.0.00077. Versões mais atualizadas, como a 12.3.0, também podem ser usadas, pois todas são disponibilizadas gratuitamente no site do fabricante, a Texas Instruments. Na versão 5.5.0.00077, é possível ainda instalar o Grace, uma ferramenta gráfica que simplifica a criação de código de configuração para os periféricos, tornando o desenvolvimento mais rápido.

Elaborado para estudantes, engenheiros e projetistas, este livro ensina, de forma didática, o funcionamento e a aplicação do MSP430G2553.

LISTA DE FIGURAS

Figura 1.1 – Organização da memória Flash, mostrando a divisão entre a memória principal (com segmentos de 512 bytes cada) e a memória de informação (segmentos A, B, C e D de 64 bytes cada).----------25
Figura 1.2 – Diagrama da memória Flash.----------31
Figura 2.1 – Circuito lógico resumido de geração de pulso de clock e seleção de fonte de clocks.---------- 38
Figura 2.2 – Circuito lógico completo do circuito de clock. Adaptado da fonte: MSP430F2xx, MSP430G2xx Family User's Guide, 2004, p. 277-----41
Figura 2.3 – Circuito lógico do Sistema de Reset do MSP430G2553.----47
Figura 3.1 – Distribuição dos pinos do MSP430G2553.----------54
Figura 3.2 – Layout 3D do MSP430G2553.----------55

LISTA DE TABELAS

Tabela 1.1 – Organização de toda a memória.----------12

Tabela 1.2 – Organização da memória RAM ----------15

Tabela 1.3 – Principais registradores de funções especiais.----------19

Tabela 1.4 – Porta 1 (P1) – Endereços de 0x0020 a 0x0027----------22

Tabela 1.5 – Timer_A0 – Endereços de 0x0160 a 0x0166----------23

Tabela 1.6 – ADC10 – Endereços de 0x01A0 a 0x01A6----------23

Tabela 1.7 – Tabela dos treze vetores de interrupção disponíveis.-----27

Tabela 1.8 – Fontes de interrupção de comunicação e seus respectivos nomes de vetores.----------29

Tabela 2.1 – Registrador DCOCTL----------43

Tabela 2.2 – Registrador BCSCTL1----------43

Tabela 2.3 – Registrador BCSCTL2----------45

Tabela 2.4 – Registrador BCSCTL3----------46

Tabela 3.1 – Tensão analógica convertida no valor digital correspondente.----------59

Tabela 3.2 – Descrição de cada pino em ordem crescente começando do pino 1 do MSP430G2553.----------63

Tabela 4.1 – Descrição das funcionalidades da porta 1.----- ----------74

Tabela 4.2 – Descrição das funcionalidades da porta 2.---------- 77

Tabela 4.3 – Descrição das funcionalidades da porta 3.----------78

Tabela 4.4 – Resumo dos registradores da porta 1.----------82

Tabela 4.5 – Registradores disponíveis em cada porta.---------- 83

Tabela 4.6 – Funções executadas pelo teclado matricial----------85

SUMÁRIO

CAPÍTULO 1 – MEMÓRIA RAM E FLASH ----------12

1.1 MEMÓRIA RAM do MSP430G2553 ----------13

1.1.1 Mapeamento da memória RAM ---------- 13

1.1.2 Organização e uso interno ----------14

1.1.3 Operação e comportamento elétrico ---------- 16

1.1.4 Estratégias de uso eficiente ----------16

1.1.5 Special Function Registers (SFRs) ----------18

1.1.6 Registradores de Periféricos ---------- 21

1.2 MEMÓRIA FLASH ----------24

1.2.1 Vetores de interrupção na Flash---------- ----------26

1.2.2 Registradores de Controle da Memória Flash---------- 30

1.3 Exemplos de utilização ----------33

1.3.1 MEMÓRIA RAM----------33

1.3.2 Memória Flash ----------34

CAPÍTULO 2 – SISTEMA DE CLOCK E DE RESET----------37

2.1 SISTEMA DE CLOCK ----------*37*

2.1.1 Sinais primários de clock ----------38

2.1.2 Fontes de clock ----------41

2.1.3 Registradores ----------42

2.2 SISTEMA DE RESET----------46

2.2.1 Sistema BOR ----------48

2.2.2 Sinal Power On Reset (POR) ----------48

2.2.3 Sinal Power Up Clear (PUC)----------49

2.3 EXEMPLOS DE UTILIZAÇÃO ----------49

CAPÍTULO 3 – PINOS ----------54

3.1 PINOS E SUAS FUNÇÕES ----------55

3.1.1 Pinos de entrada/saída digital (GPIOs) ----------56

3.1.2 Pinos de Entrada Analógica (ADC 10) ----------58

3.1.3 Pinos de Comunicação Serial ----------59

3.1.4 Pinos de Clock ----------60

3.1.5 Pinos de Temporizador (TIMER A) ----------60

3.1.6 Pinos de Alimentação e Controle ----------61

3.2 DETALHAMENTO DE TODOS OS PINOS ----------62

CAPÍTULO 4 – PORTAS ----------72

4.1 PORTAS 1 E 2. ----------72

4.1.1 Porta 1. ----------73

4.1.2 Porta 2. ----------76

4.2 PORTA 3. ----------77

4.3 REGISTRADORES. ----------79

4.4 EXEMPLOS DE UTILIZAÇÃO ---------- 83

4.5 BOAS PRÁTICAS DE USO DAS PORTAs. ----------89

REFERÊNCIAS BIBLIOGRÁFICAS ----------91

CAPÍTULO 1– MEMÓRIA RAM e FLASH

Este capítulo descreve os dois tipos principais de memória do MSP430G2553: a memória RAM, para armazenamento temporário de dados (dados voláteis) em execução, e a memória Flash, para armazenamento permanente do programa e dados essenciais (dados não-voláteis). A Tabela 1.1 apresenta como está organizada toda a memória do microcontrolador. A memória RAM está localizada nos endereços de 0x0200 a 0x03FF e a memória *FLASH* nos endereços de 0xC000 a 0xFFFF.

Tabela 1.1 – Organização de toda a memória.

Tipo	Localização (Hexadecimal)	Tamanho (bytes)
Vetores de interrupção	FFC0 a FFFF	64
Memória FLASH para o código do usuário e vetores de interrupção	C000 a FFFF	16384
Memória de informação	1000 a 10FF	256
Memória RAM	0200 a 03FF	512
Registradores de periféricos e de funções especiais	0000 a 01FF	512

1.1 Memória RAM do MSP430G2553

A memória RAM (*Random Access Memory*) é um dos elementos essenciais para o funcionamento correto de qualquer aplicação com este chip. Embora o seu tamanho reduzido de apenas 512 bytes possa parecer bastante limitado se comparado a outros microcontroladores modernos, essa capacidade é suficientemente adequada para grande parte das aplicações. Essas aplicações se encontram em controle, sensoriamento, automação de processos, sistemas embarcados alimentados por bateria, etc.

A ideia da utilização dessa memória é guardar dados temporários do programa para eventuais cálculos e processamento de dados. Por ser do tipo volátil, o conteúdo armazenado é perdido quando o microcontrolador é desligado de sua fonte de energia, que pode ser advinda de uma bateria ou de outra fonte CC. Nela, são alocadas variáveis globais e locais, a pilha de execução (stack), que é importante para chamadas de função, tratamento de interrupções e armazenamento de dados intermediários

1.1.1 Mapeamento da memória RAM

A RAM é acessada diretamente pela CPU com baixo tempo de

resposta e está mapeada entre os endereços de memória de 0x0200 até 0x03FF (valores em hexadecimal), formando um bloco contínuo de 512 (29) bytes. Esse bloco está dentro do espaço de endereçamento de 64 kb de 0x0000 até 0xFFFF que o MSP430 é capaz de acessar diretamente. Tudo isso é possível em virtude do barramento ser de 16 bits e ao modelo de memória Von Neumann que os MSP430 adotam, em que tantos dados quanto instruções compartilham o mesmo espaço de endereçamento, simplificando em muito a arquitetura do microcontrolador. Além disso, esse modelo permite que os dados sejam armazenados permanentemente na memória *FLASH* que será abordada mais adiante. Como se trata de uma memória do tipo SRAM (Static RAM), não há necessidade frequente de atualização como a DRAM. Isso resulta em tempo de acesso muito rápido com consumo reduzido de energia, dois fatores extremamente importantes em aplicações de baixo consumo.

1.1.2 Organização e uso interno

A memória RAM do MSP430G2553 é usada pela CPU do microcontrolador e seu uso é gerenciado inicialmente pelo compilador de forma bem organizada, com áreas definidas implicitamente para diferentes propósitos, a saber:

1. Pilha (*Stack)*: ocupa os endereços superiores da RAM,

começando em 0x03FF e crescendo em direção a 0x0200 (do fim para o começo da RAM). Essa região é controlada por meio do registrador SP (*Stack Pointe*r) sendo que, por meio dele, variáveis locais e parâmetros de função são empilhados dinamicamente;

2. Variáveis globais e vetores: normalmente alocados a partir do endereço 0x0200 e crescendo até o fim da RAM em 0x03FF e;

3. A região dinâmica livre da memória RAM:

a. É o espaço intermediário entre as variáveis globais, que são alocadas a partir do início da RAM (endereço 0x0200), e a pilha de execução (stack), que cresce a partir do final da RAM (endereço 0x03FF) em direção oposta;

b. Ela pode ser utilizada para armazenar vetores temporários ou estruturas de dados, desde que o programador tenha controle para evitar colisões com o crescimento da pilha ou com a expansão das variáveis globais.

A Tabela 1.2 ilustra essa organização da memória RAM.

Tabela 1.2 – Organização da memória RAM

0x03FF: Pilha (STACK) que cresce regressivamente em direção ao endereço 0x0200
Região dinâmica livre (vetores, variáveis locais, etc.)
0x0200: Variáveis globais/estáticas

Dentre as regiões apresentadas, destaca-se a importância particular da pilha, pois nela ficam guardados os dados de interrupções, as variáveis locais temporárias e os endereços de retorno de funções. Um cuidado especial deve ser tomado ao utilizá-la no intuito de evitar o que é conhecido como estouro de pilha (stack overflow). Esse estouro ocorre quando a região da pilha cresce demais e acaba atingindo as outras regiões, causando efeitos colaterais como corrupção de dados, imprevisibilidade e travamentos do programa.

1.1.3 Operação e comportamento elétrico

Essa memória é otimizada para sistemas embarcados. O consumo de energia é extremamente baixo para manter os dados armazenados. Mesmo nos modos de mais baixo consumo, como o LPM3 e o LPM4, os dados ainda são preservados integralmente com consumo de corrente que pode chegar a menos de 0,1 μA! Isso permite que dados sensíveis sejam preservados em longos períodos de inatividade, ao mesmo tempo em que aumenta a duração de uma bateria numa aplicação embarcada.

1.1.4 Estratégias de uso eficiente

Devida à memória ser bastante limitada, algumas recomendações incluem:

1. Utilizar variáveis locais sempre que possíveis no lugar

de variáveis globais;

2. Evitar utilizar estruturas de dados grandes, como vetores longos ou estruturas (structs) complexas, dentro de funções;

3. Monitorar o uso da pilha, especialmente em sistemas com muitas chamadas de função aninhadas, recursões ou interrupções frequentes;

4. Usar diretiva como __no_init, ou mapeamento manual para posicionar vetores e variáveis em áreas específicas da RAM, dependendo do compilador, e reduzir o número de variáveis globais desnecessárias;

5. Utilizar constantes armazenadas na memória Flash sempre que possível, evitando consumir RAM com dados imutáveis;

6. Utilizar alocação dinâmica sempre que possível para alocar variáveis em tempo de execução. No lugar de reservar um grande bloco de memória que eventualmente não será totalmente utilizado, alocar apenas a quantidade de memória necessária para a aplicação corrente em um dado ponto do programa e, quando não for mais necessária, realizar a liberação de memória ocupada pelas variáveis.

Para facilitar, o ambiente de desenvolvimento Code Composer Studio entrega um relatório de uso de memória após a compilação do

programa. Nesse relatório, é apresentado o quanto da memória RAM está sendo ocupada, permitindo analisar a estabilidade do programa e prevenir falhas por excesso de alocação.

1.1.5 Special Function Registers (SFRs)

Os Special Function Registers (SFRs) ou registradores de funções especiais são mapeados na memória e exercem controle direto sobre diversas funcionalidades internas do microcontrolador e de seus periféricos. Cada registrador configura uma função específica de um periférico do microcontrolador, como as portas de I/O, o sistema de clock, os temporizadores, o conversor analógico-digital (ADC), etc. Esses registradores não fazem parte da RAM, mas estão localizados em um intervalo específico de endereços de memória, entre 0x0100 e 0x01FF para os registradores de 16 bits e entre 0x0000 e 0x00FF para os registradores de 8 bits, e são acessados via leitura e escrita em endereços fixos da mesma forma que outras regiões da memória.

Esses registradores são essenciais para:

· Configurar o sistema de clock;

· Controlar o Watchdog Timer;

· Habilitar e desabilitar interrupções;

· Monitorar flags de interrupção;

· Acionar modos de baixo consumo de energia.

Ao trabalhar com o MSP430, é impossível configurar adequadamente o funcionamento do sistema sem manipular os SFRs, pois eles atuam como a interface de controle da CPU com o restante da arquitetura interna. Na Tabela 1.3 são apresentados os principais registradores de funções especiais, com o endereço de cada um deles.

Tabela 1.3 – Principais registradores de funções especiais.

Endereço	Nome do registrador	Função principal
0x0100	IE1	*Interrupt Enable Register* 1 – Habilita interrupções do Watchdog, NMI, etc.
0x0102	IFG1	Interrupt Flag Register 1 – Indica ocorrência de interrupções
0x0120	WDTCTL	Watchdog Timer Control – Configura e desabilita o watchdog

0x0128	BCSCTL1	Basic Clock System Control 1 – Ajusta fonte de clock e divisores
0x0129	BCSCTL2	Basic Clock System Control 2 – Controla o DCO, LFXT1, e divisores adicionais
0x012D	DCOCTL	DCO Control Register – Define a frequência do Digitally Controlled Oscillator

É muito importante que esses e outros registradores sejam configurados corretamente para garantir o funcionamento adequado do microcontrolador para uma dada aplicação. Como no exemplo imediato, cita-se o Watchdog Timer (WDT), que vem ativado automaticamente e, caso o seu registrador WDTCTL não seja ajustado no início do código, o sistema poderá reiniciar de forma periódica, provocando falha na operação normal do dispositivo.

Citam-se também outros registradores muito importantes, como o

IE1, IE2, IFG1 e IFG2, que controlam quais fontes de interrupção estão ativas e quais foram acionadas, lembrando que as fontes de interrupção são muito úteis para otimização do código e redução de consumo do microcontrolador.

Todos os demais registradores podem ser consultados em detalhes no datasheet da família do microcontrolador.

1.1.6 Registradores de Periféricos

Logo abaixo dos SFRs, no espaço de endereçamento de 0x0010 até aproximadamente 0x01FF, estão localizados os registradores dos periféricos. Esses registradores controlam diretamente as funcionalidades integradas do MSP430G2553, como:

· Portas digitais (P1 e P2);

· Temporizadores (Timer_A);

· Conversor Analógico-Digital (ADC10);

· Interfaces de comunicação serial (USCI A – UART/SPI e USCI B – I²C/SPI);

· Comparadores analógicos;

· Sistemas de clock.

Cada periférico possui um conjunto de registradores de controle, entrada e saída. Assim como os SFRs, esses registradores não são ar-

mazenados fisicamente na RAM, mas são acessíveis por meio de instruções de leitura/escrita em endereços fixos da memória.

Alguns exemplos de registradores de periféricos são apresentados nas tabelas a seguir:

Tabela 1.4 – Porta 1 (P1) – Endereços de 0x0020 a 0x0027

Endereço	Nome	Função
0x0020	P1IN	Entrada digital (leitura dos pinos)
0x0021	P1OUT	Saída digital (escrita nos pinos)
0x0022	P1DIR	Direção dos pinos (entrada ou saída)
0x0025	P1IE	Habilita interrupções por pino
0x0026	P1IES	Define borda de interrupção (subida/descida)
0x0027	P1IFG	Flags de interrupção por pino

Tabela 1.5 – Timer_A0 – Endereços de 0x0160 a 0x0166

Endereço	Nome	Função
0x0160	TACTL	Controle do Timer_A (modo, clock, reset)
0x0162	TAR	Contador principal do Timer_A
0x0164	TACCTL0	Controle de comparação/ captura canal 0
0x0166	TACCR0	Valor de comparação canal 0

Tabela 1.6 – ADC10 – Endereços de 0x01A0 a 0x01A6

Endereço	Nome	Função
0x01A0	ADC10CTL0	Controle do ADC: ligar/desligar, interrupção
0x01A2	ADC10CTL1	Seleção do canal, clock, modo de amostragem
0x01A6	ADC10MEM	Resultado da conversão analógica-digital

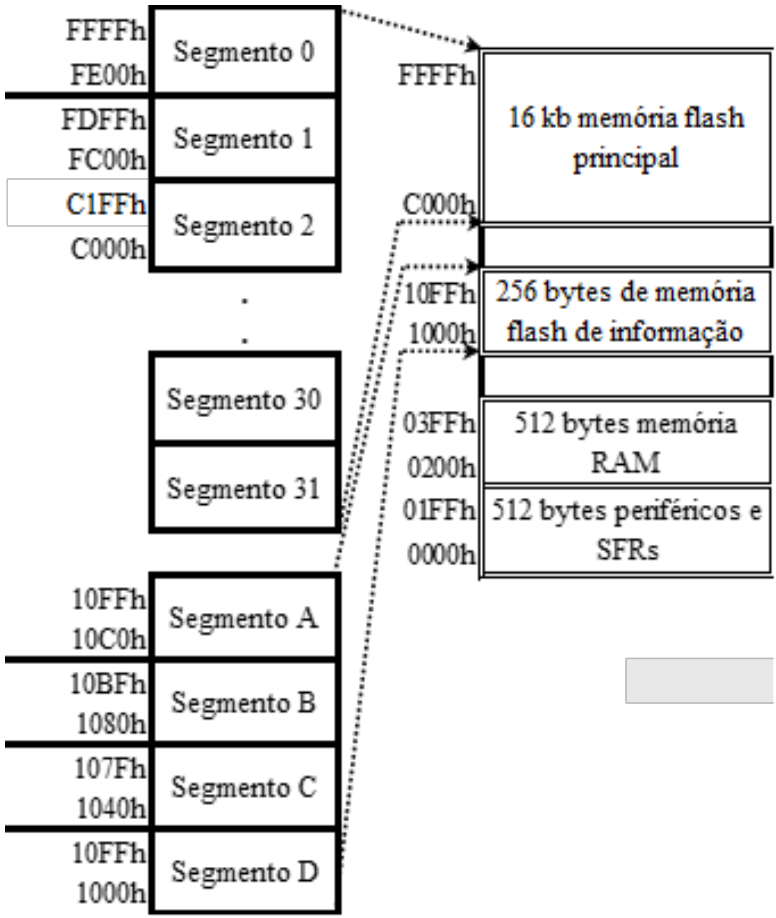
Apesar da sua capacidade limitada, a memória RAM do MSP430G2553 é extremamente eficiente e suficiente para a maioria das aplicações embarcadas de pequeno porte. Sua rapidez, baixo consumo e integração com os modos de economia de energia fazem dela uma peça-chave na arquitetura enxuta, mas poderosa, desse microcontrolador.

1.2 Memória Flash

A memória Flash desempenha um papel muito importante como memória do tipo não-volátil. Isso quer dizer que ela armazena os dados mesmo quando a alimentação é desligada. Todo o código do usuário representado por uma série de instruções é armazenado e executado, além de dados que podem ser guardados permanentemente, a saber parâmetros de calibração ou valores constantes.Além da memória principal, existe uma área separada especial chamada de memória de informação, dividida em segmentos de 64 bytes, usada para armazenar dados permanentes.

A organização da Flash é feita em segmentos. A memória principal é dividida em segmentos de 512 bytes cada, e a memória de informação é organizada em quatro segmentos menores, nomeados A, B, C e D, cada um com 64 bytes. A Figura 1.1 apresenta a organização da memória Flash por segmentos.

Figura 1.1 – Organização da memória Flash, mostrando a divisão entre a memória principal (com segmentos de 512 bytes cada) e a memória de informação (segmentos A, B, C e D de 64 bytes cada).



Fonte: Adaptado do documento "MSP430 Flash Memory Characteristics", Figura 1.

A CPU pode escrever dados na Flash em unidades de byte ou palavra. No entanto, a operação de escrita possui algumas particularidades. Uma característica importante da Flash é que a escrita de dados para programar os bits para o valor lógico 0 pode ser feita individualmente, seja bit a bit, byte a byte ou palavra a palavra. O tamanho do segmento da Flash não afeta essa escrita de 0. Na memória Flash, a gravação de um valor lógico 1 é um processo não tão simples. Em vez de simplesmente alterar um bit, é necessário executar uma operação de apagamento de um segmento inteiro. Dessa forma, é impossível mudar um único bit para 1 sem apagar completamente o segmento no qaul ele está localizado.

Essa peculiaridade na escrita da Flash está relacionada ao seu funcionamento físico, que exige tensões mais altas para apagar e programar. O microcontrolador possue um circuito interno chamado "charge pump" para gerar essa tensão elevada necessária para as operações de programação e apagamento da Flash.

1.2.1 Vetores de interrupção na Flash

A região final da memória Flash, entre os endereços 0xFFE0 e

0xFFFF, é reservada para os vetores de interrupção. Cada vetor ocupa 2 bytes (16 bits) e apontam para o endereço inicial da rotina de tratamento de uma interrupção específica.

Essa área é gravada automaticamente pelo linker do compilador durante a geração do firmware, com base nas declarações #pragma vector= ou __attribute__((interrupt)) no código C. O programador não precisa gravar esses endereços manualmente, mas deve garantir que as rotinas estão corretamente nomeadas e posicionadas.

Durante uma interrupção, a CPU realiza o seguinte fluxo:

1. Armazena o endereço de retorno (PC) e o registrador SR na pilha
2. Lê o endereço do vetor de interrupção na Flash
3. Salta para a rotina associada

Após a execução, usa a instrução RETI para retornar, restaurando PC e SR.

Tabela 1.7 apresenta todos os vetores de interrupção disponíveis.

Tabela 1.7 – Tabela dos treze vetores de interrupção disponíveis.

Endereço (hexadecimal)	Interrupção associada	#pragma vector=
FFE0	-	-

FFE2	-	-
FFE4	Porta 1	PORT1_VEC-TOR
FFE6	Porta 2	PORT2_VEC-TOR
FFE8	-	-
FFEA	ADC10	ADC10_VEC-TOR
FFEC	UCA0TX(*) e UCB0TX Em modo UART ou SPI: UCB0TXIFG transmissão de dados. Em modo I2C: UCB0TXIFG e UCB0RXIFG.	USCIAB0TX_ VECTOR
FFEE	UCA0RX e UCB0RX Em modo SPI: UCB0RXIFG recepção de dados SPI. Em modo I2C: flags UCALIFG, UCNACKI-FG, ICSTTIFG, UCSTPIFG.	USCIAB0RX_ VECTOR
FFF0	Canais 1, 2 do timerA0 e estouro do timerA0	TIMER0_A1_ VECTOR
FFF2	Canal 0 do timerA0	TIMER0_A0_ VECTOR
FFF4	Watchdog Timer	WDT_VECTOR
FFF6	Comparador A+	COMPARATO-RA_VECTOR
FFF8	Canais 1, 2 do timerA1 e estouro do timerA1	TIMER1_A1_ VECTOR
FFFA	Canal 0 do timerA1	TIMER1_A0_ VECTOR
FFFC	Não-mascarável, falha de oscila-dor, e violação de acesso	NMI_VECTOR

FFFE	RESET – Endereço de entrada do programa principal	RESET_VECTOR

Essa tabela está disponível no datasheet oficial do MSP430G2553, e indica que o último endereço da Flash (0xFFFE) contém o vetor de reset, ou seja, o ponto onde o programa começa a ser executado após a inicialização do microcontrolador. Esse vetor aponta para a função main() indiretamente, por meio do código de inicialização.

Observação: na USCI_A em qualquer modo (UART ou SPI), o nome do vetor será sempre relacionado ao sentido da comunicação. A Tabela 1.8 ilustra bem isso e mostra, para os demais vetores relacionados, todas as possibilidades da comunicação da USCI_A e USCI_B.

Tabela 1.8 – Fontes de interrupção de comunicação e seus respectivos nomes de vetores.

USCI_A	
UART TX	USCIAB0TX_VECTOR
UART RX	USCIAB0RX_VECTOR
SPI TX	USCIAB0TX_VECTOR
SPI RX	USCIAB0RX_VECTOR
USCI_B	
I2C TX ou	
I2C RX	USCIAB0TX_VECTOR

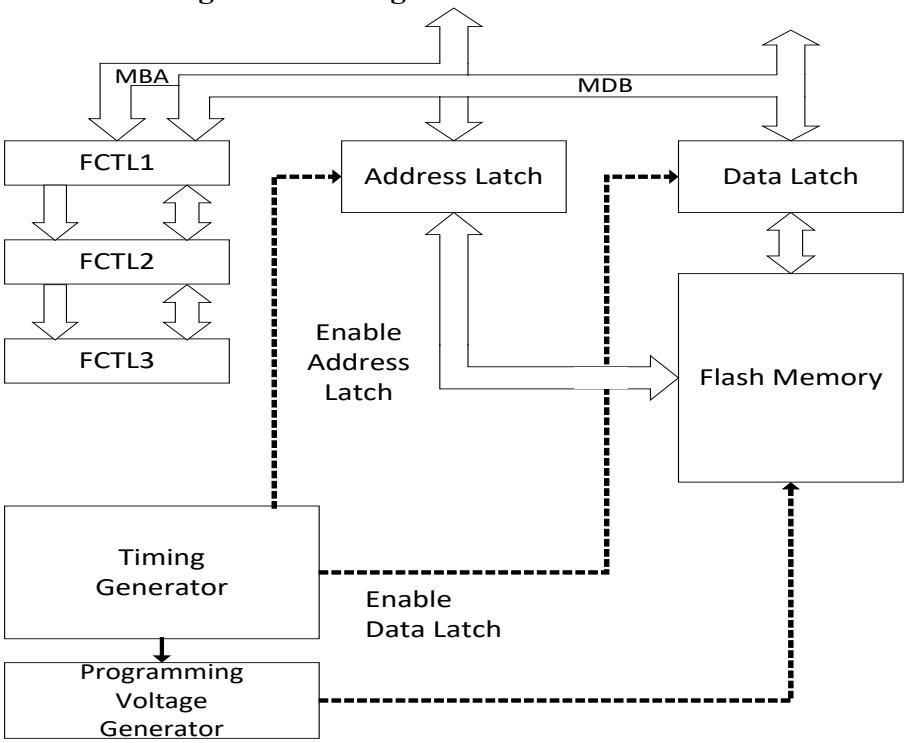
I2C Status: UCALIFG (perda de arbitragem), UCNACKIFG (não reconhecimento), ICSTTIFG (início), ICSTTIFG (término).	USCIAB0RX_VECTOR
SPI TX	USCIAB0TX_VECTOR
SPI RX	USCIAB0RX_VECTOR

1.*2.2 Registradores de Controle da Memória Flash*

Para realmente dominar a interação com a memória Flash, seja para gravar um novo programa ou atualizar dados persistentes, o programador precisa interagir diretamente com um conjunto específico de registradores: os Registradores de Controle da Flash. Embora a escrita e o apagamento sejam operações complexas, esses quatro registradores: FCTL1, FCTL2, FCTL3 e FCTL4 encapsulam a lógica necessária para tal operação.

Na Figura 1.2 é apresentado o diagrama de blocos de controle e acesso da memória Flash do MSP430 da família 2. O registrador FCTL4 não é implementado para o MSP430G2553. O diagrama ilustra como um processador interage com a memória Flash para operações de leitura, escrita e apagamento.

Figura 1.2 – Diagrama da memória Flash.



Os registradores de Controle FCTL1 a FCTL3 são a interface de software para o hardware de controle da Flash. Eles contêm bits de controle que definem a operação a ser realizada (ex: escrita, apagamento de segmento), o estado de segurança e os sinalizadores de status (ex: operação completa, erro). Esses são mais do que simples endereços de memória; eles são a interface primordial para garantir que as operações na Flash aconteçam de forma segura e controlada. Um aspecto crucial em

sua manipulação é a necessidade de prover uma "chave" de segurança, um mecanismo robusto que impede modificações acidentais e protege a integridade do firmware e dos dados críticos armazenados.

Vejamos os papéis específicos de cada um:

· **FCTL1 (Flash Control 1):** Este registrador é o maestro das operações de escrita e apagamento. É nele que especificamos se desejamos escrever um único byte ou uma palavra, e também o bit que desencadeia as operações de apagamento, seja de um segmento da memória ou de todo o conteúdo da Flash.

· **FCTL2 (Flash Control 2):** Sua função principal é ajustar a temporização das operações da Flash. Nele configuramos o gerador de clock interno que dita a velocidade dos processos de programação e apagamento, além de selecionar qual fonte de clock será utilizada para essas tarefas sensíveis.

· **FCTL3 (Flash Control 3):** Este registrador funciona como um painel de controle e segurança da Flash. Contém os flags que indicam o estado atual da memória, como se está ocupada (BUSY) com uma operação ou se algum erro ocorreu. Mais importante ainda, é aqui que se gerencia a proteção contra escrita (LOCK) e se habilita o mecanismo de chave de segurança, essencial para permitir qualquer modificação.

Compreender as funções gerais desses registradores é um passo inicial. No entanto, o controle preciso da Flash exige um mergulho mais profundo na definição de cada um de seus bits. Cada bit nesses registradores possui um papel específico, desde a seleção de fontes de clock até o controle de flags de status e a ativação de modos de proteção. Para uma análise detalhada bit a bit, com todos os seus comportamentos e interdependências, a referência é o "MSP430F2xx, MSP430G2xx Family User's Guide", que oferece a documentação completa de todos os periféricos do microcontrolador.

1.3 Exemplos de utilização

A seguir são apresentados dois exemplos de utilização sendo um para cada tipo de memória.

1.3.1 Memória RAM

Nesse programa, foi testado a utilização da memória RAM, em que uma variável do tipo int cujo nome é contador é iniciada e incrementada em uma unidade a cada 1000000 ciclos. Depois essa variável é mostrada no LCD. Ao reiniciar o microcontrolador, percebemos que a variável reinicia seu valor para 0, mostrando assim a volatilidade da memória RAM e como ela guarda valores apenas enquanto o MSP está funcionando.

```
#include <msp430.h>
#include <i2c_lcd.h>
#include <i2c.h>
#include <stdio.h>

#include <ti/mcu/msp430/Grace.h>
int main(void) {
      Grace_init();
      int contador = 0;
      LCD_Init(0x27, 2, 16);
      LCD_SetCursor(0, 0);
      clearArea();
      while (1) {
            lcd_put_num(contador, 0, 0);
            contador++;
            __delay_cycles(1000000);
            LCD_Clear();
      }
}
```

1.4.2 Memória Flash

Nesse programa, observamos o uso da memória FLASH, em que iniciamos um contador que é incrementado em uma unidadade a cada 1000000 de ciclos e mostra o valor no LCD. A cada 20 unidades adicionadas, o valor é salvo na memória FLASH. Ou seja, ao reiniciar o MSP, ele irá começar a contagem do ultimo múltiplo de 20 que passou pelo LCD. Assim, mostrando o caráter não volátil da memória FLASH,

que, diferente da memória RAM, mesmo ao desligar a alimentação de energia elétrica, guarda seu valor.

```
#include <msp430.h>
#include <i2c_lcd.h>
#include <i2c.h>
#include <stdio.h>
#include <ti/mcu/msp430/Grace.h>

#define SEGMENT_D 0x1040
unsigned int *flash_ptr = (unsigned int *) SEGMENT_D;

void write_to_flash(unsigned int value) {
      FCTL3 = FWKEY;
      FCTL1 = FWKEY + ERASE;
      *flash_ptr = 0;
      FCTL1 = FWKEY + WRT;
      *flash_ptr = value;
      FCTL1 = FWKEY;
      FCTL3 = FWKEY + LOCK;
      __delay_cycles(10000);
}

unsigned int read_from_flash() {
      return *flash_ptr;
}

int main(void) {
      Grace_init();
      // Inicializar o LCD ANTES de fazer qualquer
leitura
      LCD_Init(0x27, 2, 16);
      LCD_SetCursor(0, 0);
```

```
	LCD_Print(“Iniciando...”);
	__delay_cycles(1000000);
	clearArea();

	unsigned int contador = 0;
	unsigned int valor_lido = read_from_flash();

	// Proteção contra valores inválidos
	if (valor_lido == 0xFFFF || valor_lido > 50000)

	{	contador = 0;
	} else {
		contador = valor_lido;
	}

	while (1) {
		lcd_put_num(contador, 0, 0);
		contador++;

		if (contador % 20 == 0) {
			write_to_flash(contador);
		}

		__delay_cycles(1000000);
		clearArea();
	}

}
```

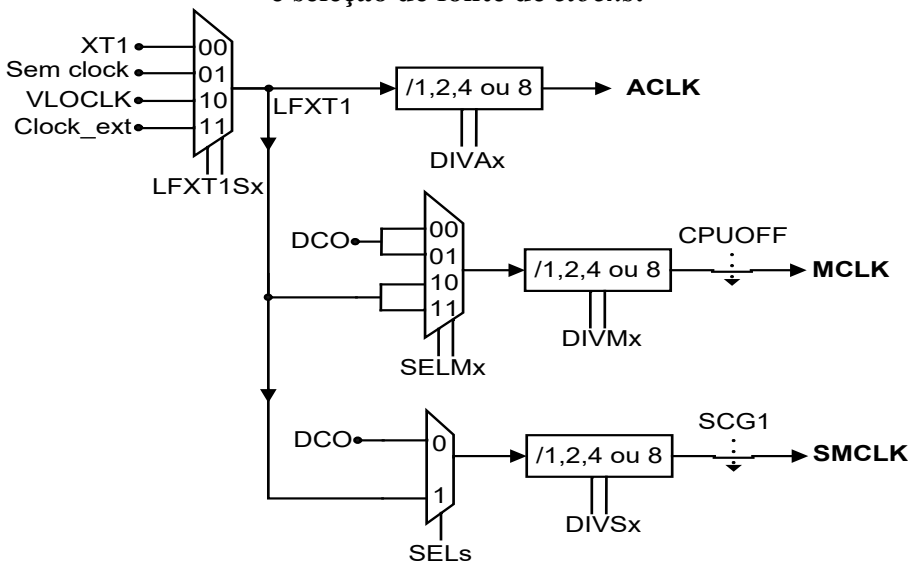
CAPÍTULO 2 -- SISTEMA DE CLOCK E DE RESET

Muitas funcionalidades de um microcontrolador são dependentes de um sinal de *clock*, como *timers*, conversor A/D, verificação de falhas, etc. O sinal de *reset* é muito importante para inicializar o microcontrolador e seus periféricos a um estado inicial, possibilitando tirá-lo de uma eventual condição de travamento. Será visto em detalhes o sistema de *clock* e de *reset*.

2.1 Sistema de clock

O sistema de *clock* é responsável por fornecer os sinais de temporização que coordenam o funcionamento da CPU, dos periféricos e dos demais sistemas internos do microcontrolador. Esse sistema é projetado para oferecer flexibilidade e baixo consumo de energia, permitindo ao desenvolvedor selecionar diferentes fontes de *clock* conforme a necessidade da aplicação, sem consumir muito do processamento do *chip*. A Figura 2.1 mostra o circuito lógico resumido do sistema de *clock* sendo possível visualizar três fontes de *clock* que podem ser utilizados para a CPU e os periféricos, a saber: ACLK, MCLK e SMCLK.

Figura 2.1 – Circuito lógico resumido de geração de pulso de *clock* e seleção de fonte de *clocks*.



2.1.1 - Sinais primários de clock

O microcontrolador em estudo oferece duas opções de sinais primários de clock: o LFXT1 e o DCO. O LFXT1 é o de baixa frequência (umas dezenas de kHz) e o DCO é de alta frequência (na ordem de MHz).

· LFXT1 de baixa frequência

Oferece a opção do Oscilador Interno de Baixa Frequência (VLO, sigla para “Very Low-frequency Oscillator”), que fornece em média 12 kHz (no mínimo de 4 e no máximo 20 kHz) que é selecionado quando

LFXT1Sx = 10 e XTS = 0 (XTS = 1 não é suportado, somente para outros modelos da família 2).

Ademais, fornece suporte para cristal externo, sendo o módulo Oscilador de Baixa Frequência com Cristal (LFXT, sigla para “Low-Frequency Crystal Oscillator”) deve sempre operar para o modo de baixa frequência (LF – Low Frequency). O funcionamento deste depende de um cristal de quartzo que se conecta nas portas XIN (pino P2.6) e XOUT (pino P2.7). Os bits XCAPx configuram a capacitância para a carga fornecida internamente entre 1 pF, 6 pF, 10 pF ou 12,5 pF (capacitores externos podem ser adicionados), para o cristal no modo LF. Como, por exemplo, um famigerado cristal de relógio de 32.768 Hz pode ser utilizado para implementar um relógio digital de baixo consumo. Vale ressaltar que este oscilador não permanece funcionando enquanto o VLO está funcionando, assim como o VLO não funciona enquanto o LFXT está funcionando. Essa característica proporciona uma redução racional de consumo de energia para esse módulo.

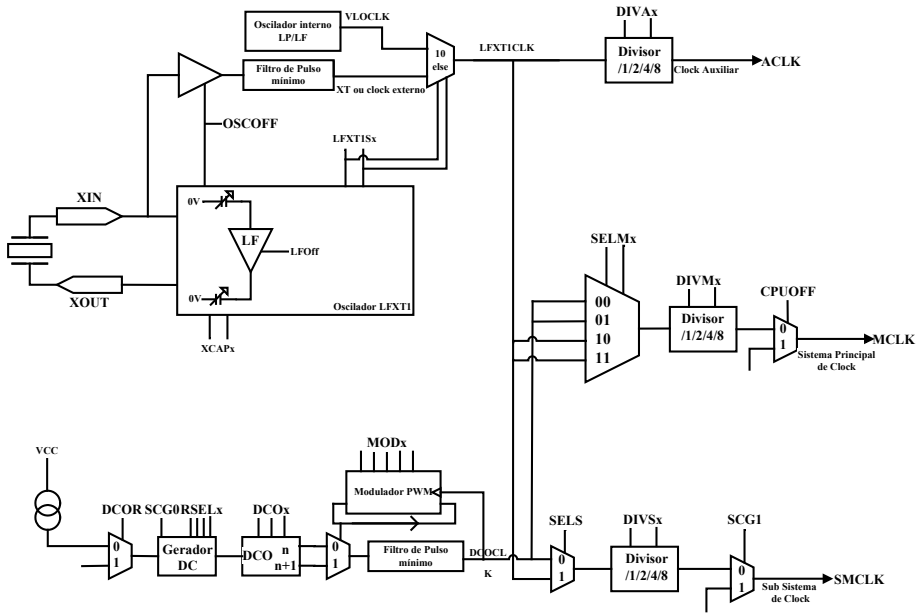
Por fim, é possível o LFXT1 receber um sinal de clock digital externo de baixa frequência na porta XIN, desde que os bits LFXT1s estejam configurados em 11 e a porta P2.6 configurada para a função XIN.

· DCO de alta frequência

Além do LFXT1, existe um oscilador interno com frequência ajustável por software, o Oscilador Controlado Digitalmente (DCO, sigla para “Digitally Controlled Oscillator”). O sinal que sai do gerador DC do DCO (veja Figura 2.2) é selecionado pelo RSELx (do registrador BCSCTL1, com 4 bits, para definir entre as faixas de frecuencia de ~0,1 MHz e ~16MHz). Note que com 4 bits para RSELx, são possíveis dezesseis valores de frequência diferentes. Esse sinal selecionado ainda é aplicado a um divisor programável, que por meio dos bits DCOx (registrador DCOCTL) escolhe uma frequência mais específica dentro da faixa escolhida. O sinal de frequência selecionado e o próximo (DCOx e DCOx+1) são aplicados a um modulador de frequência, que mistura periodicamente essas duas frequências, deixando a frequência de saída com seu valor entre as frequências fornecidas. Por meio do bit MODx (do registrador DCOCTL com 5 bits) é configurado quantos dos 32 ciclos do modulador usarão o DCOx+1. Com MODx = 4, por exemplo, 4 ciclos com DCOx+1 e 28 com DCOx serão utilizados pelo modulador. Vale ressaltar que essa fonte de clock de DCO é afetada por temperatura e tensão de alimentação, daí a necessidade de se utilizar as constantes de calibração específicas para o chip (que é baseado em testes realizado pelo fabricante). As constantes de calibração são para frequências de

operação de 1, 8, 12 e 16 MHz.

Figura 2.2 – Circuito lógico completo do circuito de clock. Adaptado da fonte: MSP430F2xx, MSP430G2xx Family User's Guide, 2004, p. 277.



2.1.2 - Fontes de clock

Os dois sinais primários de clock podem ser selecionados para as duas fontes de clock MCLK, SMCLK. No caso do ACLK, por se tratar de uma fonte de clock de baixa velocidade, apenas o sinal LFXT1 está disponível.

· **Main Clock (MCLK)**

Essa fonte fornece o clock somente para a CPU executar as ins-

truções, podendo ser essa fonte desligada pelo bit CPUOFF. No caso do watchdog timer, essa fonte é apenas monitorada pela unidade lógica de verificação de falha. O sinal para essa fonte pode ser qualquer uma das disponíveis (Figura 2.1), selecionados pelo bit SELMx do registrador BSCCTL2 (00 e 01, fonte DCO; 10 e 11 LFXT1), podendo ser dividido por 1, 2, 4 ou 8 (DIVMx).

· **Sub-Main Clock (SMCLK)**

O sinal para essa fonte pode ser qualquer uma das disponíveis (Figura 2.1), selecionadas pelo bit SELSx do BSCCTL2 (0 sinal DCO; 1 LFXT1), podendo ser dividido por 1, 2, 4 ou 8 (DIVSx), usado primariamente para periféricos como timers ou unidades de comunicação USCI_A e USCI_B, nos modos UART/SPI e SPI/ I2C respectivamente.

· **Auxiliary Clock (ACLK)**

Vindo sempre do LFXT1, a frequência pode ser dividida por 1, 2, 4 ou 8 (DIVAx). Usado primariamente para periféricos que requerem baixo consumo, por ser um clock lento, mas estável.

2.1.3 - Registradores

Os registradores que configuram o sistema de clock são: DCOCTL, BCSCTL1, BCSCTL2 e BCSCTL3.

2.1.3.1 - DCOCTL (DCO clock frequency control)

Esse registrador é responsável por ajustes finos na frequência do oscilador interno DCO, possibilitando obter uma frequência mais precisa. A Tabela 2.1 apresenta os registrador DCOCTL.

Tabela 2.1 – Registrador DCOCTL

Bit	Campo	Tipo	Descrição
0-4	MODx	R/W	Seleciona em quantos ciclos, de 32 ciclos, a frequência DCOx + 1 será usada, exceto quando o bit DCOx não é 7
5-7	DCOx	R/W	Seleciona uma frequência específica da faixa escolhida pelo RSELx

2.1.3.2 BCSCTL1 (Basic Clock System Control 1)

Esse registrador é responsável por ajustes por passos maiores na frequência do oscilador interno DCO, possibilitando obter uma alteração de frequência maior. Tabela 2.2 apresenta os registrador BCSCTL1.

Tabela 2.2 – Registrador BCSCTL1

Bit	Campo	Tipo	Descrição
0-3	RSELx	R/W	Seleciona uma das frequências disponíveis para serem usadas para o DCO clock

4-5	DIVAx	R/W	Divisor de clock Auxiliar (ACLK), sendo que a frequência será divida por: 00 -> /1 01 -> /2 10 -> /4 11 -> /8
6	XTS	R/W	Em outros microcontroladores, seleciona em que modo a fonte de clock LFTX irá funcionar 0 -> Baixa Frequência 1 -> Alta Frequência Todavia o MSP430G2553 só suporta modo de baixa frequência (LF mode)
7	XT2OFF	R/W	Em outros microcontroladores, liga ou desliga a fonte de clock XT2 irá funcionar 0 -> Ligado 1 -> Desligado Todavia o MSP430G2553 não possui o módulo oscilador XT2

2.1.3.3 BCSCTL2 (Basic Clock System Control 2)

Seleciona o sinal primário de clock para as fontes MCLK e SMCLK, além do fator de divisão. A Tabela 2.3 apresenta os registrador BCSCTL2.

Tabela 2.3 – Registrador BCSCTL2

Bit	Campo	Tipo	Descrição
0	LFXT1OF	R	Indica se o oscilador LFXT1 : 0 -> Não falhou (padrão) 1 -> Falhou
1	XT2OF	R	Em chips com o módulo XT2 (que não é o caso do chip em estudo), indica falha no funcionamento deste módulo
2-3	XCAPx	R/W	Seleciona a capacitância para ser usada no LFTX1: 00 -> 1 pF 01 -> 6 pF 10 -> 10 pF 11 -> 12,5 pF
4-5	LFXT1Sx	R/W	Seleciona se o LFTXCLK 00 -> cristal de 32 kHz do LFTX1 01 -> Reservado 10 -> VLO 11 -> Fonte de clock digital externa
6-7	XT2Sx	R/W	Não está presente no MS-P430G2553, todavia em outros chips seleciona a frequência do módulo XT2

2.1.3.4 BCSCTL3 (Basic Clock System Control 3)

Seleciona o sinal para a fonte de clock ACLK e sinaliza a operação dos osciladores externos de baixa frequência. A Tabela 2.4 apresenta os registrador BCSCTL3.

Tabela 2.4 – Registrador BCSCTL3

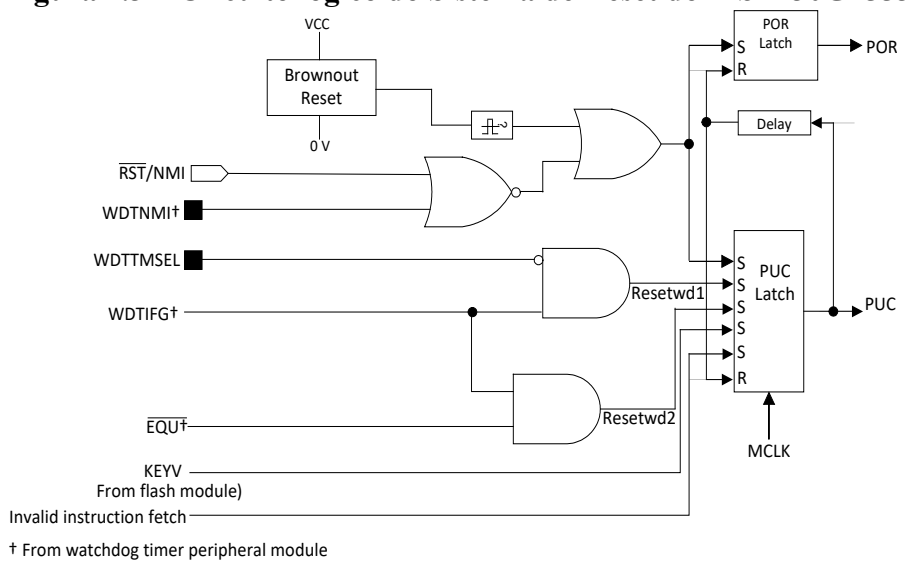
Bit	**Campo**	**Tipo**	**Descrição**
0	LFXT1OF	R	Indica se o oscilador LFXT1 : 0 -> Não falhou (padrão) 1 -> Falhou
1	XT2OF	R	Em *chips* com o módulo XT2 (que não é o caso do *chip* em estudo), indica falha no funcionamento deste módulo
2-3	XCAPx	R/W	Seleciona a capacitância para ser usada no LFTX1: 00 -> 1 pF 01 -> 6 pF 10 -> 10 pF 11 -> 12,5 pF
4-5	LFXT1Sx	R/W	Seleciona se o LFTXCLK 00 -> cristal de 32 kHz do LFTX1 01 -> Reservado 10 -> VLO 11 -> Fonte de *clock* digital externa
6-7	XT2Sx	R/W	Não está presente no MSP430G2553, todavia em outros *chips* seleciona a frequência do módulo XT2

2.2 Sistema de Reset

O reset coloca o MCU o seu estado inicial em uma condição padrão já conhecida, ao produzir a partir de determinados eventos externos e internos ao microcontrolador. O reset coloca no estado padrão a CPU,

os periféricos, o sistema de clock, a memória RAM, os registradores de configuração dos pinos, os registradores SFR e especialmente o watchdog do MSP430 no modo de reset ativado! A Figura 2.3 apresenta a arquitetura do sistema de reset, na qual logo encontramos a presença de um bloco chamado de Brownout Reset (BOR) e de dois sinais de saída desse sistema: POR e PUC, que serão abordados nesta seção.

Figura 2.3 – Circuito lógico do Sistema de Reset do MSP430G2553



Fonte: Adaptado de MSP430F2xx, MSP430G2xx Family User's Guide, 2004, p. 30.

2.2.1 Sistema BOR

O sistema BOR é uma maneira de garantir que o microcontrolador irá funcionar sobre uma tensão segura. Isso previne uma inicialização com tensões indevidas que podem gerar um funcionamento inesperado ou uma obstrução de memória flash.

Quando o sistema detecta um crescimento da tensão de alimentação, é mantido um sinal para o POR Latch, ativando um sinal POR para o sistema, que o coloca em um estado padrão de reset, mantendo esse estado até 2000 microsegundos (tBOR) após a tensão de alimentação ultrapassar a margem superior V(B_IT+) (em 1,49 V), liberando o microcontrolador para funcionar normalmente. Em casos de queda da tensão, abaixo de V(B_IT–) (em 1,35 V), o sistema BOR volta a mandar um sinal para o POR Latch, garantindo uma faixa segura em que o microcontrolador pode funcionar sem risco, pois fora dessa faixa, o microcontrolador fica em um constante estado de reset.

2.2.2 Sinal Power On Reset (POR)

Um sinal de POR é gerado por um destes dois eventos: quando o microcontrolador é energizado (via sistema BOR), que é o evento mais comum, e de maneira manual, colocando o pino RST/NMI em um nível lógico baixo. Esse sinal redefine a maioria dos registradores de peri-

féricos para seus estados padrão de inicialização como timers, ADC, comparador, etc.

2.2.3. Sinal Power Up Clear (PUC)

Enquanto que o sinal de POR supervisa a parte de alimentação, o PUC supervisiona condições internas, podendo lidar com elas sem mudar todo o estado do chip.

Um sinal de PUC sempre é provocado por uma emissão do sinal POR, além de ser gerado por um estouro na contagem do watchdog (quando este está no modo watchdog), uma violação da chave de acesso do registrador de controle do watchdog ou da memória flash. Além disso, esse sinal é provocado quando a CPU tenta buscar uma instrução em um endereço de memória reservado para periféricos (de 0x0000 até 0x01FF).

O sinal PUC não altera o conteúdo da RAM, mas altera alguns registradores do microcontrolador. No datasheeet da família 2 é possível verificar quais são todos os registradores alterados pelo PUC e quais não são.

2.3 Exemplos de utilização

No código a seguir é demonstrado o funcionamento dos sinais de clock fornecidos pelo MSP430G2553. A ideia é ligar leds diferen-

tes com base em frequência de oscilação de clocks diferentes. O Led conectado no pino P2.1 comuta de estado a cada segundo com base na frequência de ACLK que é selecionado ao timerA0, enquanto que o led conectado no pino P2.2 comuta de estado a cada meio segundo com base na frequência da fonte do SMCLK que é selecionado ao timerA1. O Led no pino P2.0 comuta com base em um atraso proporcionado por um comando de laço for. Esse atraso depende da velocidade da fonte MCLK. No código, essa velocidade foi calibrada para 1 MHz.

```
#include <msp430.h>

void main(void) {
      WDTCTL = WDTPW | WDTHOLD;  //desativa o watchdog
      // Configurar os pinos de saída
      P2DIR |= BIT0 | BIT1 | BIT2;//LEDs em P2.0,
    P2.1, P2.2
      P2OUT &= ~(BIT0 | BIT1 | BIT2);//Nível lógico
      baixo     //  //Selecionar o sinal VLO para a
      fonte de clock ACLK
      BCSCTL3 |= LFXT1S_2; // ACLK = VLO (~12kHz)

      // Timer0_A com clock ACLK para comutar o estado
      do LED1 na   //P2.1
      TA0CCTL0 = CCIE;// Habilita interrupção
      TA0CCR0 = 12000;            // Aproximadamente
      1 segundo      //com VLO
      TA0CTL = TASSEL_1 | MC_1 | ID_0;  // ACLK, modo
up, divisão /8
```

```
    //Timer1_A: SMCLK -> LED2/P 2.2
    TA1CCTL0 = CCIE;                //    Habilita
interrupção
    TA1CCR0 = 62500;            // Aproximadamente
    0.5s (se //SMCLK = 1MHz / 8)
    TA1CTL = TASSEL_2 | MC_1 | ID_3;  // SMCLK, modo
up, divisão/8

    //  Timer1_A1:  MCLK  ->  LED3  (simulando  com
polling simples)
    BCSCTL1 = CALBC1_1MHZ;      // DCO ~1MHz
    DCOCTL = CALDCO_1MHZ;       //    Valor    pré
    calibrado,armazenado na //memória
    __enable_interrupt();             //  Habilita
interrupções

    //Loop principal: piscar LED3 /p 2.0 com MCLK
(delay manual)
    while (1) {
       volatile unsigned int i;
       P2OUT ^= BIT0;
        for (i = 0; i < 50000; i++)   ;// Delay
simples com MCLK
    }
}

// Interrupção Timer0_A0 (ACLK -> LED1)
#pragma vector = TIMER0_A0_VECTOR
__interrupt void Timer0_A(void) {
    P2OUT ^= BIT1;

}
```

```
// Interrupção Timer1_A0 (SMCLK -> LED2)
#pragma vector = TIMER1_A0_VECTOR
__interrupt void Timer1_A(void) {
P2OUT ^= BIT2;
}
```

No código a seguir, é demonstrado o efeito dos sinais POR e PUC. O funcionamento esperado é um dos leds piscando (na P2.1) durante a inicialização do chip por um reset POR. Após certo tempo, o watchdog timer é colocado no modo watchdog e o programa fica sendo mantido em um laço infinito vazio. Como o reset do watchdog não é realizado nesse laço, o timer acaba estourando a sua contagem após algum tempo, gerando o sinal PUC. Então, um outro led na P2.2 pisca, sinalizando um reset PUC devido ao estouro de contagem do watchdog.

```
#include <msp430.h>
// Variavel que fará a verificação de reset e nao sera
apagada da RAM apos o reset por POR ou WatchDog
#include NOINIT(resetFlag)
unsigned int resetFlag;

void delay(void) {
      volatile unsigned int i;
      for (i = 0; i < 50000; i++) ;
}
```

```
void main(void) {
      WDTCTL = WDTPW | WDTHOLD;   // Parar o watchdog
      int i, j;

      // Configurar pinos de saída
      P2DIR |= BIT1 | BIT2; // LEDs em P2.1, P2.2
      P2OUT &= ~(BIT1 | BIT2); //Nível lógico baixo
      nas portas de //saída, para garantir que os
      LEDS iniciem desligados
      if (resetFlag == 0xA5A5) {
      // Identificado um reset por WatchDog
      for (i = 0; i < 5; i++) {
            P2OUT ^= BIT2;
            delay();
      }
      } else {
            // Identificado a inicialização do chip,um
      reset POR
      for (j = 0; j < 5; j++) {
            P2OUT ^= BIT1;
            delay();
      }
      }
      // Apos o Reset POR, atribuimos um valor a
      variavel //persistente ao reset
      resetFlag = 0xA5A5;
      // Definimos o WatchDog para o modo Reset
      WDTCTL = WDTPW | WDTCNTCL | WDTIS0;
      // Força WatchDog com um loop infinito vazio
      while (1) {
      }
}
```

CAPÍTULO 3– PINOS

Neste capítulo, será estudado sobre os pinos com encapsulamento QFN (Quad Flat No-lead) de 32 pinos que estão distribuídos conforme apresentado nas figuras 3.1 e 3.2, retiradas do datasheet oficial disponibilizado no site oficial do fabricante, Texas Instruments. Possui pinos nos quatro lados (pois tem o formato quadrado e plano) e os pinos não são saltados para fora do corpo do chip, mas sinda é possível o acesso direto aos 32 pinos.

Figura 3.1 – Distribuição dos pinos do MSP430G2553.

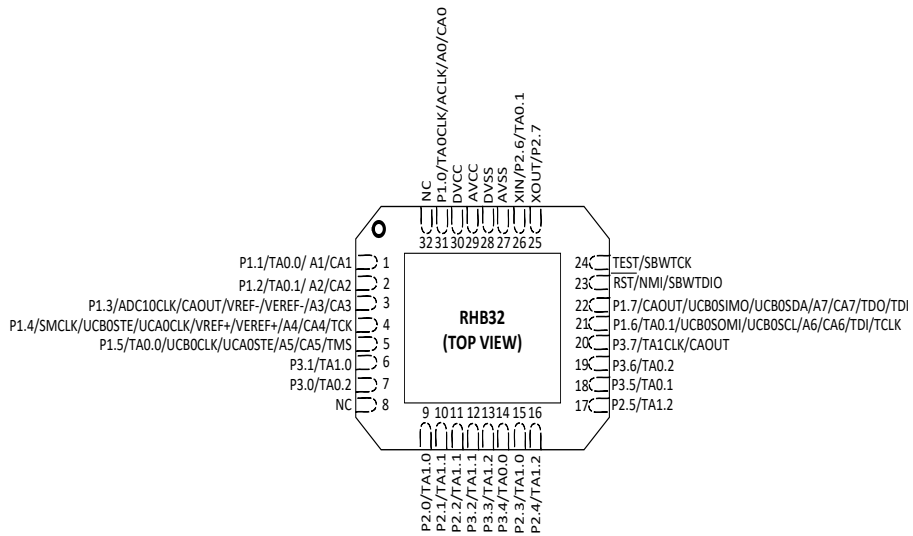
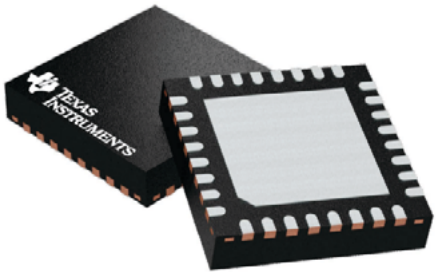


Figura 3.2 – Layout 3D do MSP430G2553.



Ele possui o tamanho de 5,0 mm x 5,0mm e altura de aproximadamente 1mm com os pinos distribuídos em 8 de cada lado na parte inferior do chip. Sendo assim, esse pinos tem aproximadamente 0,3 mm x 0,5 mm e são distribuídos com aproximadamente 0,5 mm de distância um do outro. Esse formato de encapsulamento é muito utilizado quando não existe muita disponibilidade de espaço ou deseja-se utilizar uma pequena área do projeto, porém, seu tamanho reduzido dificulta a solda manual, sendo necessária a utilização de fornos de refluxo ou estações de retrabalho para que haja a conexão adequada dos pinos.

3.1 Pinos e suas funções

Os 32 pinos podem ser utilizados como entrada/saída digital, co-

municação serial, temporizadores, entradas analógicas, alimentação ou controle. Esses pinos são distribuídos nos corpo do chip em ordem crescente, partindo de 1 a 32, no sentido anti-horário, partindo do pino mais superior da lateral esquerda. O pino 1 pode ser ainda marcado por um ponto ou um chanfro na parte de cima do chip, para facilitar e orientar a montagem. O pino 8 e o 32 não são conectados ao circuito interno do chip e não possuem função.

3.1.1 Pinos de entrada/saída digital (GPIOs)

Os GPIOs (General Purpose Input/Output) são pinos do microcontrolador que podemos utilizar livremente como entradas digitais, geralmente utilizadas para ler estado de botões, sensores e etc.; ou ainda como saídas digitais utilizadas para acender LEDs, acionar relés, enviar sinais e etc. Tais pinos são chamado de pinos de propósito geral porque, por meio do software, podemos decidir qual função ele irá assumir. A seguir, são apresentados dois exemplos práticos implementados em linguagem C da utilização dos pinos, o primeiro como saída e em seguida como entrada digital.

3.1.1.1 Pino 1.0 funcionando como saída digital

Através dos registradores P1SEL e P1SEL2, selecionamos a função do pino P1.0 para I/O por meio destes dois comandos: P1SEL &= ~0x01; P1SEL2 &= ~0x01. Em seguida, configuramos para saída digital por meio do comando: P1DIR |= 0x01.

Se desejarmos nível alto no pino, basta fazer: P1OUT |= 0x01. Caso nível baixo, o comando é P1OUT &= ~0x01.

Essa é uma forma simples e prática de como configurar por meio de software o pino P1.0 como saída digital. Para configurar o pino do microcontrolador, utilizamos registradores, que são endereços de memória. Cada bit de um registrador controla uma função específica. O registrador P1DIR, por exemplo, define a direção do pino: se um bit for 0, ele configura o pino como entrada; se for 1, como saída. Na programação, a operação |= (OR bit a bit) é a forma de ligar o bit 0, o que, nesse caso, configura o pino P1.0 para funcionar como saída. Em seguida, utiliza-se outro registrador, o P1OUT, fazendo a mesma operação anterior e colocando o pino em nível lógico alto.

3.1.1.2 Pino 1.0 funcionando como entrada digital

Aproveitando a configuração da seção anterior, para a função de entrada digital, basta alterar a configuração do P1DIR para P1DIR &=

~0x01. Para verificar se o pino está em nível alto, podemos utilizar o bloco de comando de controle: if (P1IN & 0x01) { }. Se o pino está nível alto, executa o que está dentro do comando de controle if; caso contrário, não executa.

Assim, como no exemplo apresentado anteriormente, utilizaram-se registradores para configurar o pino como entrada e depois para verificar se o dado da entrada está em nível lógico alto. Com isso, pode-se implementar diversos projetos que necessitam de análise de sinais externos.

3.1.2 Pinos de Entrada Analógica (ADC 10)

Para sinais analógicos, o microcontrolador conta com a funcionalidade de Conversor Analógico Digital, que de forma bem simples serve para transformar esses sinais analógicos em valores digitais que ele pode entender. O ADC10 converte tensões analógicas em valores digitais de 10 bits, ou seja, isso significa que ele divide a faixa de tensão em 1024 níveis ($2^{10} = 1024$).

Como exemplo, uma variação de um sensor de temperatura que emite um sinal de tensão que varia de 0 V a 3,3 V, o valor correspondente em digital é apresentado na Tabela 3.1.

Tabela 3.1 – Tensão analógica convertida no valor digital correspondente.

Tensão emitida pelo sensor	Valor entendido pelo MSP
0 V	0
1,65 V	511
3,33 V	1023
Va	Vd = (1023 x Va) / 3,3 (Fórmula)

3.1.3 Pinos de Comunicação Serial

Os pinos de comunicação serial são muito importantes no microcontrolador, pois são usados para enviar e receber dados de forma sequencial (bit a bit) por meio de protocolos de comunicação serial, ou seja, ao invés de enviar todos os bits de uma vez (como ocorre na comunicação paralela), é enviado um bit por vez.

Entre os protocolos seriais utilizados temos UART, SPI e I2C. O UART (Universal Asynchronous Receiver/Transmitter) funciona utilizando dois pinos: o TX para transmissão e o RX utilizado para recepção, o que o torna muito simples, sendo amplamente utilizado para conectar o microcontrolador ao computador, módulo bluetooth, GPS entre outros.

Já o SPI (Serial Peripheral Interface) faz uma comunicação mais rápida e utiliza pelo menos três pinos MOSI (dados de um controlador principal para um secundário), MISO (dados do secundário para o principal) e SCLK (sinal de clock). Esse método é muito utilizado quando existe a necessidade de alta velocidade, como na utilização de memórias e displays.

Por fim, temos o sistema I2C (Inter-Integrated Circuit) que permite a interligação de vários dispositivos no mesmo barramento e utiliza dois pinos SCL (clock) e SDA (dados). Uma utilização bem comum para esse tipo de comunicação é implementação de displays e relógios.

3.1.4 Pinos de Clock

Os pinos de clock são aqueles utilizados para receber ou distribuir sinais de clock, sinais esses que serão utilizados internamente ou externamente. Uma aplicação muito interessante desses pinos é que por meio deles pode-se utilizar um cristal externo para gerar um clock estável que pode ser utilizado em projetos que exigem muita precisão de tempo, como um relógio de tempo real.

3.1.5 Pinos de Temporizador (TIMER A)

Os pinos podem atuar também como entrada ou como saída de temporizadores, como o timerA0 e o timerA1, que são temporizadores internos do microcontrolador.

Esses temporizadores podem atuar contando o tempo, gerando sinais PWM ou medindo períodos de sinais por meio de capturas de seus três canais de número 0 ao 2. Eles permitem diversas aplicações, mas principalmente automação de tarefas com precisão de tempo, como piscar LEDs, gerar sons ou controlar equipamentos.

Daí a extrema importância desses pinos, sendo que, dependendo da aplicação, um pino para o timer poderá ser configurado como entrada ou como saída. Por exemplo, para possibilitar a realização da captura de bordas de subida de um sinal digital, o pino 1 pode ser configurado para ser a entrada de captura do canal 0 do timerA0. Em outra aplicação, esse mesmo pino pode ser configurado como saída de PWM desse mesmo canal desse timer.

3.1.6 Pinos de Alimentação e Controle

Os pinos de alimentação e de controle são fundamentais para o funcionamento do chip. Esses pinos não controlam periféricos diretamente, mas são essenciais para alimentação de todos eles, além de gra-

vação de programas e controle geral do microcontrolador.

O pino VCC recebe a tensão (normalmente de 3,3V), enquanto o GND é o referencial de terra (0 V). Já o pino ~RST/NMI permite reiniciar o microcontrolador manualmente através da aplicação de um nível baixo nesse pino. Por fim, apenas os dois pinos TEST e SBWTDIO são usados para gravação e depuração do código, através da interface Spy-Bi-Wire, que é eficiente.

3.2 Detalhamento de Todos os Pinos

No tópico anterior foi apresentada uma visão geral sobre as funções dos principais pinos do MSP430G2553. Abaixo, na Tabela 3.2, temos a apresentação dos pinos divididos de acordo com suas numerações, funções e nomenclaturas.

Tabela 3.2 – Descrição de cada pino em ordem crescente começando do pino 1 do MSP430G2553.

Número do Pino	Nomenclatura	Descrição
1	P1.1 / TA0.0/ UCA0RXD/ A1/ CA1	I/O - Função de entrada/ saída
		Timer0_A – Captura, comparação, PWM
		USCI_A0 UART – RX (Recebimento de dados)
		USCI_A0 SPI – SPI (Dados do escravo para o mestre)
		A1 - Entrada analógica 1 do conversor A/D
		CA1 - Entrada analógica 1 do comparador analógico

2	P1.2/ TA0.1/ UCA0TXD/ UCA0SIMO/ A2/ CA2	I/O - Função de entrada/ saída
		Timer0_A – Captura, Comparação, PWM
		USCI_A0 UART – TX (Transmissão de dados)
		USCI_A0 SPI – SPI (Dados do mestre para o escravo)
		Entrada analógica 2 do conversor A/D
		CA2 - Entrada analógica 2 do comparador analógico
3	P1.3/ ADC10CLK/ A3/ VREF-/ VEREF-/ CA3/ CAOUT	I/O - Função de entrada/ saída
		ADC10 - Saída do clock interno
		Entrada analógica 3 do conversor A/D
		CA3 - Entrada analógica 3 do comparador analógico
		CAOUT - Saída digital do comparador

4	P1.4/ SMCLK/ UCB0STE/ UCA0CLK/ A4/ VREF+/ VEREF+/ CA4/ TCK	I/O - Função de entrada/ saída
		SMCLK - Clock secundário externo
		USCI_B0 - Habilitação de transmissão do escravo
		USCI_A0 - Clock da comunicação serial (SPI)
		Entrada analógica 4 do conversor A/D
		ADC10 - Referência positiva de tensão
		CA4 - Entrada analógica 4 do comparador analógico
		JTAG - Programação e testes do dispositivo

5	P1.5/ TA0.0/ UCB0CLK/ UCA0STE/ A5/ CA5/ TMS	I/O - Função de entrada/ saída
		Timer0_A – Comparação, PWM
		USCI_B0 - Clock entre mestre e escravo.
		USCI_A0 - Habilitar a transmissão
		Entrada analógica 5 do conversor A/D
		CA5 - Entrada analógica 5 do comparador analógico
		JTAG - Programação e testes do dispositivo
6	P3.1/ TA1.0	I/O - Função de entrada/ saída
		Timer1_A - Comparação, PWM
7	P3.0/ TA0.2	I/O - Função de entrada/ saída
		Timer1_A - Captura, comparação, PWM
8	NC	Não conectado
9	P2.0/ TA1.0	I/O - Função de entrada/ saída
		Timer1_A - Captura, comparação, PWM

10	P2.1/ TA1.1	I/O - Função de entrada/ saída
		Timer1_A - Comparação, PWM
11	P2.2/ TA1.1	I/O - Captura, comparação, PWM
		Timer1_A - Geração, captura e comparação
12	P3.2/ TA1.1	I/O - Função de entrada/ saída
		Timer1_A - Comparação, PWM
13	P3.3/ TA1.2	I/O - Função de entrada/ saída
		Timer1_A - Comparação, PWM
14	P3.4/ TA0.0	I/O - Função de entrada/ saída
		Timer1_A - Comparação, PWM
15	P2.3/ TA1.0	I/O - Função de entrada/ saída
		Timer1_A - Comparação, PWM
16	P2.4/ TA1.2	I/O - Função de entrada/ saída
		Timer1_A - Comparação, PWM

17	P2.5/ TA1.2	I/O - Função de entrada/ saída
		Timer1_A - Captura, comparação, PWM
18	P3.5/ TA0.1	I/O - Função de entrada/ saída
		Timer1_A - Compara-ção, PWM
19	P3.6/ TA0.2	I/O - Função de entrada/ saída
		Timer1_A - Compara-ção, PWM
20	P3.7/ TA1CLK/ CAOUT	I/O - Função de entrada/ saída
		Timer1_A - entrada de clock externo
		CAOUT - saída do com-parador analógico

21	P1.6/ TA0.1/ A6/ CA6/ UCB0SOMI/ UCB0SCL/ TDI/TCLK	I/O - Função de entrada/ saída
		Timer0_A - Comparação, PWM
		Entrada analógica 6 do conversor A/D
		CA6 - Entrada analógica 6 do comparador analógico
		USCI_B0 SPI- Recebe dados do escravo
		USCI_B0 I2C - Linha de clock SCL
		JTAG – Entrada de clock ou dados durante testes
22	P1.7/ A7/ CA7/ CAOUT/ UCB0SIMO/ UCB0SDA/ TDO/TDI	I/O - Função de entrada/ saída
		Entrada analógica 7 do conversor A/D
		CA7 - Entrada positiva do comparador analógico
		CA7 - Entrada analógica 7 do comparador analógico
		USCI_B0 SPI - Envia dados do mestre pro escravo
		USCI_B0 I2C - Linha de dados SDA

23	RST/ NMI/ SBWTDIO	Reset - Reinicialização do microcontrolador
		NMI – Interrupção não-mascarável com alta prioridade
		SBWTDIO - Programação e depuração
24	TEST/ SBWTCK	TEST - Modo de teste para os pinos JTAG
		SBWTCK - Entrada de clock para a interface Spy-Bi-Wire
25	P2.7/ XOUT	I/O - Função de entrada/ saída
		XOUT - Terminal de saída do oscilador de cristal,
26	P2.6/ XIN/ TA0.1	I/O - Função de entrada/ saída
		XIN - Entrada do oscilador de cristal
		Timer0_A - Comparação, PWM
27	DVSS	Referência de Terra
28	DVSS	Referência de Terra
29	AVCC	Fonte de alimentação analógica
30	DVCC	Pino de alimentação digital

		I/O - Função de entrada/ saída
		Timer0_A - Entrada de clock para o temporiza-dor
		ACLK - Saída de sinal de clock auxiliar
		Entrada analógica 8 do conversor A/D
31	P1.0/ TA0CLK/ ACLK/ A0 CA0	CA0 - Entrada analógi-ca 0 do comparador analógico
32	NC	Não conectado

CAPÍTULO 4 – PORTAS

Nesta seção, teremos uma abordagem mais específica apresentando as portas do microcontrolador, que são grupos de pinos organizados para facilitar a programação e o controle dos sinais. Nesse modelo de MSP, temos 3 portas (P1.X, P2.X, P3.X), sendo que cada porta tem 8 pinos numerados de 0 a 7, que podem ser configurados de acordo com as funções abordadas no capítulo anterior. As portas 1 e 2 se diferenciam da porta 3, pois a P3.X não possue a função de interrupção.

Cada um desses pinos dessas portas podem ser configurados individualmente, de acordo com a função desejada, pois alguns desses pinos possuem várias funções multiplexadas, que podem ser ativadas ou desativadas pelo programação do projeto. Cada porta possui um conjunto de registradores básicos que ajudam a configurar as funções para cada pino da porta, e assim é possível acessar todas ferramentas disponibilizadas pelo microcontrolador de forma multiplexada.

4.1 Portas 1 e 2.

As portas P1.X e P2.X, possuem a capacidade de interrupção que

pode ser habilitada individuamente e configurada para fornecer uma interrupção em borda de subida ou descida de um sinal de entrada. Todas as linhas de entrada e saída (E/S) de P1 fornecem um único vetor de interrupção, e todas as linhas de E/S de P2 também fornecem um único vetor de interrupção, porém diferente do vetor de P1.

4.1.1 Porta 1.

A porta P1 é a porta que possue mais funções disponíveis no MPS430G2553, além da configuração de E/S digital de uso geral. Ela permite a utilização de periféricos importantes, como a comunicação UART, SPI e I2C, conversores analógico-digitais (ADC10), temporizadores (Timer_A) e comparadores analógicos. Portanto, essa porta se trata de uma porta bastante completa e com uma maior concentração de funções, dando maiores possibilidades ao desenvolvedor de projetos. Além disso, como já mencionado, a porta P1 também tem a capacidade de gerar interrupções configuradas, com os registradores PxIFG, PxIE e PxIES, que seram discutidos nos tópicos seguintes sobre registradores.

A Tabela 4.1 nos mostra as funções relacionadas entre seus respectivos pinos da porta P1 e a numeração da pinagem, no qual é encontrado cada terminal dessa porta.

Tabela 4.1 – Descrição das funcionalidades da porta 1.

TERMINAL P1.X		
Nome	Nº	Descrição
P1.0/ TA0CLK/ ACLK/ A0/ CA0	31	Pinos de E/S digital de uso geral Timer0_A, entrada do sinal de Clock TACLK ACLK saída do sinal ADC10 entrada analogica A0 Comparador_A+, entrada CA0
P1.1/ TA0.0/ UCA0RXD/ UCA0SOMI/ A1/ CA1	1	Pinos de E/S digital de uso geral Timer0_A, captura: entrada CCI0A, comparação: saída Out0 / transmissão BLS USCI_A0 UART modo: receber entrada de dados USCI_A0 SPI modo: sáida de dados escravo/ entrada de dados mestre ADC10 entrada analogica A1 Comparador_A+, entrada CA1
P1.2/ TA0.1/ UCA0TXD/ UCA0SIMO/ A2/ CA2	2	Pino de E/S digital de uso geral Timer0_A, captura: entrada CCI1A, comparação: saída Out1 USCI_A0 UART modo: transmissão de saída de dados USCI_A0 SPI modo: entradas de dados escravo/ saída de dados mestre ADC10 entrada analogica A2 Comparador_A+, entrada CA2

P1.3/ ADC10CLK/ A3/ VREF-/VEREF-/ CA3/ CAOUT	3	Pino de E/S digital de uso geral ADC10 saída de clock do conversor ADC10 entrada analogica A3 ADC10 tensão de referência negativa Comparador_A+, entrada CA3 Comparador_A+, saída
P1.4/ SMCLK/ UCB0STE/ UCA0CLK/ A4/ VREF+/ VEREF+/ CA4/ TCK	4	Pino de E/S digital de uso geral SMCLK, saída de sinal USCI_B0 habilitação da transmissão escrava USCI_A0, clock entrada/saída ADC10 entrada analogica A4 ADC10 tensão de referência positiva Comparador_A+, entrada CA4 JTAG clock de teste, terminal de entrada para programação e teste do dispositivo
P1.5/ TA0.0/ UCB0CLK/ UCA0STE/ A5/ CA5/ TMS	5	Pino de E/S digital de uso geral Timer0_A, comparação: saída Out0/ receber BLS USCI_B0, clock entrada/saída USCI_A0 habilitação da transmissão escrava ADC10 entrada analogica A5 Comparador_A+, entrada CA5 JTAG, seleção do modo de teste
P1.6/ TA0.1/ A6/ CA6/ UCB0SOMI/ UCB0SCL/ TDI/TCLK	21	Pino de E/S digital de uso geral Timer0_A, comparação: saída Out1 ADC10 entrada analogica A6 Comparador_A+, entrada CA6 USCI_B0 SPI, modo escravo OUT e mestre IN USCI_B0 I2C, modo clock SCL I2C JTAG, entrada de dados de teste ou entrada de clock teste

		Pino de E/S digital de uso geral
P1.7/		ADC10 entrada analogica A7
A7/		Comparador_A+, entrada CA7
CA7/		Comparador_A+, saída
CAOUT/		USCI_B0 SPI, modo: escravo IN e mestre OUT
UCB0SIMO/		USCI_B0 I2C, modo: dados SDA I2C
UCB0SDA/		JTAG terminal de saída de dados de teste ou
TDO/TDI	22	entrada de dados de teste

4.1.2 Porta 2.

A porta P2 se assemelha à porta P1, porém com menos disponibilidades de funções multiplexadas. Ela é mais utilizada para temporizadores e para entradas e saídas digitais. Podemos entender a porta P2 como uma porta auxiliar que vem para adicionar com a porta P1 como se fosse uma ampliação dos pinos de P1. Assim, podemos utilizar os pinos da porta P2 para funções especificas de temporização que utiliza os timers e que também conta com as interrupções, e assim deixar a portar P1 reservada para outras funções especificas diferentes de P2.

Na Tabela 4.2, temos a relação entre os pinos e suas respectivas funções da porta P2.

Tabela 4.2 – Descrição das funcionalidades da porta 2.

TERMINAL P2.X		
Nome	Nº	Descrição
P2.0/ TA1.0	9	Pino de E/S digital de uso geral Timer1_A, captura: entrada CCI0A, comparação: saída Out0
P2.1/ TA1.1	10	Pino de E/S digital de uso geral Timer1_A, captura: entrada CCI1A, comparação: saída Out1
P2.2/ TA1.1	11	Pino de E/S digital de uso geral Timer1_A, captura: entrada CCI1B, comparação: saída Out1
P2.3/ TA1.0	15	Pino de E/S digital de uso geral Timer1_A, captura: entrada CCI0B, comparação: saída Out0
P2.4/ TA1.2	16	Pino de E/S digital de uso geral Timer1_A, captura: entrada CCI2A, comparação: saída Out2
P2.5/ TA1.2	17	Pino de E/S digital de uso geral Timer1_A, captura: entrada CCI2B, comparação: saída Out2
XIN/ P2.6/ TA0.1	26	Terminal de entrada do cristal oscilador Pino de E/S digital de uso geral Timer0_A, comparação: saída Out1
XOUT/ P2.7	25	Terminal de saída do cristal oscilador Pino de E/S digital de uso geral

4.2 PORTA 3.

A porta 3 já se assemelha à porta 2, pois também é mais utilizada para E/S digitais de uso geral e para timer, porém diferentemente da

porta P2, a P3 não possui interrupções, sendo uma porta mais simples. Apesar disso, a P3 não deixa de ser importante, pois ela permite mais possibilidades de utilização dos periféricos de canais dos timers A0 e A1, deixando livre as outras portas para funções mais específicas.

Na Tabela 4.3, temos a relação entre os pinos e suas respectivas funções da porta P3

Tabela 4.3 – Descrição das funcionalidades da porta 3.

TERMINAL P3.X		
Nome	Nº	Descrição
P3.0/ TA0.2	7	Pino de E/S digital de uso geral Timer0_A, captura: entrada CCI2A, comparação: saída Out2
P3.1/ TA1.0	6	Pino de E/S digital de uso geral Timer1_A, comparação: saída Out0
P3.2/ TA1.1	12	Pino de E/S digital de uso geral Timer1_A, comparação: saída Out1
P3.3/ TA1.2	13	Pino de E/S digital de uso geral Timer1_A, comparação: saída Out2
P3.4/ TA0.0	14	Pino de E/S digital de uso geral Timer0_A, comparação: saída Out0
P3.5/ TA0.1	18	Pino de E/S digital de uso geral Timer0_A, comparação: saída Out1
P3.6/ TA0.2	19	Pino de E/S digital de uso geral Timer1_A, comparação: saída Out2
P3.7/ TA1CLK/ CAOUT	20	Pino de E/S digital de uso geral Timer1_A,entrada do sinal de clock TACLK Comparador _A+, saída

4.3 Registradores.

Cada porta possui um conjunto básico de registradores que controlam sua operação. Como foi visto nos tópicos anteriores, todas as portas têm a função de entrada e saída digitais de uso geral e ao menos uma função alternativa multiplexada em um mesmo pino. Assim, os registradores nos auxiliam na programação do microcontrolador, pois com ele podemos configurar o MSP430 de acordo com a necessidade do projeto. Todas a portas possuem esse conjunto básico de registradores:

· Um registrador de entrada (PxIN) para a leitura do nível dos pinos da porta;

· Um registrador de saída (PxOUT) para escrita de nível nos pinos da porta;

· Um registrador de controle individual da direção dos pinos da porta, para definir se o pino será de entrada ou saída (PxDIR);

· Um registrador para selecionar a função do pino (PxSEL e PxSEL2), que define entre a função normal de E/S e uma função multiplexada ao pino;

· E o registrador PxREN para habilitar/desabilitar o resistor de

pull-up ou de pull-down de cada pino.

Para todos os registradores podemos definir um bit em nível lógico igual a “1” ou “0”. Se PxSEL = 0, o registrador irá selecionar a função normal de E/S daquele pino, e se PxSEL = 1, seleciona a função alternativa. Para PxDIR = 1, o pino é configurado como saída, e o nível lógico presente nesse pino será aquele configurado pelo bit do registrador PxOUT. Por exemplo, se tivermos um LED com terminal positivo no pino P3.0 e o negativo no GND, para acender esse LED precisamos de um nível lógico igual a 1 no pino P3.0. Para isso, colocamos P3DIR = 1 (saída) e P3OUT = 1 (define o nível lógico 1 para a saída do respectivo pino).

Se o registrador PxDIR = 0, o pino será configurado como entrada e o nível lógico externo presente no pino pode ser lido pelo registrador PxIN. Para ler o estado de um botão que está conectado ao pino P3.2 com um resistor de pull-down, você precisa configurar o pino como uma entrada. Para isso, defina P3DIR = 0. Quando o botão for pressionado, o P3IN terá o valor lógico 1 (alto). Quando o botão estiver solto, o P3IN terá o valor 0 (baixo). Desse modo, o microcontrolador consegue interpretar o estado do botão com base no valor lógico do registro PxIN.

Além desse grupo básico de registradores, há resistores de pull-down e pull-up internos que nos ajudam a reduzir componentes externos ao microcontrolador e a reduzir falhas de conexões devido a má conexão. Portanto, esse modelo de MSP também possui registradores para configurar esses resistores internos. O registrador PxREN controla a ativação e desativação dos resistores. Quando PxREN = 1, o resistor interno é ativado e o registrador PxOUT controla se o resistor é de pull--up (PxOUR =1) ou pull-down (PxOUT = 0).

As portas P1 e P2, além dos registadores mencionados acima, incluem os registradores específicos para a função de interrupção. Um registrador sinalizador de interrupção do pino (PxIFG), um registrador de seleção de borda de sensibilidade de cada pino (PxIES) e um registrador de habilitação individual de interrupção para cada pino da porta (PxIE). A função de interrupção disponível nas portas PI e P2 permite que uma transição de "0" para "1" ou de "1" para "0" em um dos pinos dessas portas provoque a interrupção do programa em execução, desde que o respectivo bit do registrador PxIE esteja setado, assim como o controle global de interrupções (GIE).

As interrupções de P1 e P2 são desabilitadas quando PxSEL = 1, ou seja, quando qualquer bit de P1SELx ou P2SELx é definido, a

função de interrupção do pino correspondente é desabilitada e os sinais nesses pinos não gerarão interrupções em P1 ou P2, independentemente do estado do bit P1IE ou P2IE correspondente.

Tabela 4.4 – Resumo dos registradores da porta 1.

REGISTRADOR	DESCRIÇÃO
P1DIR	Determina a direção de cada pino (0 = entrada, 1 = saída)
P1OUT	Define o valor lógico nos pinos configurados como saída
P1IN	Lê o valor lógico presente nos pinos configurados como entrada
P1REN	Habilita os resistores internos (pull-up ou pull-down)
P1SEL	Ativa funções alternativas primárias (bit a bit)
P1SEL2	Ativa funções alternativas secundárias
P1IE	Habilita interrupções nos pinos selecionados
P1IES	Seleciona a borda de ativação da interrupção (subida ou descida)
P1IFG	Indica qual pino gerou uma interrupção

Tabela 4.5 – Registradores disponíveis em cada porta.

REGISTRADOR		
P1	P2	P3
P1DIR	P2DIR	P3DIR
P1OUT	P2OUT	P3OUT
P1IN	P2IN	P3IN
P1REN	P2REN	P3REN
P1SEL	P2SEL	P3SEL
P1SEL2	P2SEL2	P3SEL2
P1IE	P2IE	-
P1IES	P2IES	-
P1IFG	P2IFG	-

4.4 Exemplos de utilização

No código do exemplo 1, todos os pinos das três portas são colocados como saída e em um laço inifinito o estado das portas são comutados.

```
#include <msp430.h>

int main(void) {
      WDTCTL = WDTPW | WDTHOLD;   // Desabilita o
Watchdog Timer
      // Configura todas as portas GPIO como saídas e
      definir nível  //baixo
```

```
    P1DIR = 0xFF;
    P1OUT = 0x00;
    P2DIR = 0xFF;
    P2OUT = 0x00;
    P3DIR = 0xFF;
    P3OUT = 0x00;

    // Desabilita Timer_A0 e seus canais
    TA0CTL = MC_0;         // Parar Timer_A0
    TA0CCTL0 = 0;          // Desabilitar Canal 0
    TA0CCTL1 = 0;          // Desabilitar Canal 1
    TA0CCTL2 = 0;          // Desabilitar Canal 2

    // Desabilita o comparador analógico
    CACTL1 = 0;           // Desabilitar Comparador
Analógico
    CACTL2 = 0;           // Desabilitar Comparador
Analógico

    // Desabilita comunicações (UART, SPI, I2C) do
    USCI_A0 e USCI_B0
    UCA0CTL1 = UCSWRST;    // Colocar o USCI_A0 em
reset

    // Desabilita o conversor A/D (ADC10)
    ADC10CTL0 = 0;         // Desabilitar ADC10
    ADC10CTL1 = 0;
    ADC10AE0 = 0;

    //Reconfigura os pinos de função especial para
modo I/O padrão
    P1SEL =  0x00;
    P1SEL2 = 0x00;
```

```
      P2SEL =  0x00;
      P2SEL2 = 0x00;
      P3SEL =  0x00;
      P3SEL2 = 0x00;
      P1REN =  0x00;
      P2REN =  0x00;
      P3REN =  0x00;
      while(1){
            P1OUT ^= 0xFF; //comuta o estado dos pinos
da porta 1
            P2OUT ^= 0xFF; //comuta o estado dos pinos
      da porta 2
            P3OUT ^= 0xFF; //comuta o estado dos pinos
da porta 3
            __delay_cycles(200000);//aguarda 200 ms
      }
}
```

No exemplo dois a seguir, o código é desenvolvido para controlar 3 LEDs através de um teclado matricial 4x4 e cada tecla possui uma função a ser executada. Dessa forma, temos listado abaixo a ação que corresponde à tecla do botão na Tabela 4.6.

Tabela 4.6 – Funções executadas pelo teclado matricial

TECLA	FUNÇÃO
1	Acender o LED 1
2	Acender o LED 2
3	Acender o LED 3
4	Comutar o LED 1

5	Comutar o LED 2
6	Comutar o LED 3
7	Apagar o LED 1
8	Apagar o LED 2
9	Apagar o LED 3
A	Apagar todos os LED's
B	-
C	Comuta todos os LED's
D	Deslocamento de bit à esquerda para os LED's
*	-
0	-
#	-

```
#include <msp430.h>
#define ROWS 4
#define COLS 4

unsigned char teclado[ROWS][COLS] =
{ { '1', '2', '3', 'A' }, { '4', '5', '6', 'B' }, {
'7', '8', '9', 'C' }, { '*', '0', '#', 'D' } };

void main(void) {

      WDTCTL = WDTPW | WDTHOLD;   // Desabilita o
Watchdog Timer
      P2DIR |= 0xFF;   // Configura P2.0-P2.7 como
saídas para os LEDs
      P2OUT = 0x00;   // Desliga todos os LEDs

      P3DIR &= ~(BIT3 + BIT4 + BIT5 + BIT6); //
      Configura P3.3-P3.6 //como entradas para as
```

```
    linhas do teclado
    P3DIR |= (BIT0 + BIT1 + BIT2 + BIT3);     //
    Define as colunas //como saídas
    P2REN |= (BIT0 | BIT1 | BIT2 | BIT3); //Habilita
    resistor de //Pull-Down
    unsigned int tecla_pressionada, row, col;

    while (1) {
        tecla_pressionada = 0;//em zero indica
que não há tecla
        //pressionada
        for (col = 0; col < COLS; col++) {

            P3OUT |= (BIT0 + BIT1 + BIT2 + BIT3);
            // Define as //saidas com  valores
            altos
            P3OUT &= ~(BIT0 << col); // Define
            valor baixo para //a coluna atual

            for (row = 0; row < ROWS; row++) {
                if (!(P3IN & (BIT3 << row))) {
                // Verifica se a //linha atual
                está baixa (botão pressionado)
                    tecla_pressionada =
teclado[row][col];
                    while (!(P3IN & (BIT3
                    << row))); /Aguarda //a
                    liberação do botão
                }
            }
        }

        P2OUT &= 0; // Desliga todos os LEDs
```

```
            switch (tecla_pressionada) {
            case '1':
            case '2':
            case '3':
                  P2OUT |= BIT0 << (tecla_pressionada
- '1');
                  break;
            case '4':
            case '5':
            case '6':
                  P2OUT ^= BIT0 << (tecla_pressionada
- '4');
                  break;
            case '7':
            case '8':
            case '9':
                  P2OUT &= ~(BIT0 << (tecla_pressionada
- '7'));
                  break;
            case '*':
                  break;
            case '0':
                  break;
            case '#':
                  break;
            case 'A':
                  P2OUT &= ~(BIT0 | BIT1 | BIT2);
                  break;
            case 'B':
                  break;
            case 'C':
                  P2OUT ^= BIT0 + BIT1 + BIT2;
                  break;
```

```
            case 'D':
                  P2OUT = (P2OUT << 1);
                  break;
            default:
                  break;
            }
      }
}
```

4.5 Boas práticas de uso das portas.

O MSP430G2553 é um microcontrolador que nos disponibiliza funções importantes para um projeto robusto, e o encapsulamento de 32 pinos nos disponibiliza um maior número de portas nos proporcionando maiores possibilidades de utilizar periféricos externos. Assim, com o conhecimento de todas suas funções em relação a cada pino, é possível fazer um projeto de forma a utilizar os pinos de forma eficiente, porém sempre respeitando os limites do microcontrolador e tomando alguns cuidados de utilização para evitar o consumo elevado do chip e um funcionamento indesejado.

Com isso, podemos citar alguns cuidados importantes:

· Para proteções para pinos de entradas, adicione resistores de pull-up ou pull-down de 10 kΩ;

· Para saídas como LED, usar resistores limitadores de corrente;

· Evitar flutuações;

· Para alimentação, não ultrapassar o limite de tensão Vcc de 3,6 V, pois do contrário pode causar problemas com o microcontrolador;

· Correntes em GPIO de no máximo 6 mA por pino, porém o ideal seria manter uma corrente em torno dos 2 mA para garantir a segurança do MSP;

· Em entradas ADC, evitar ruídos usando filtros capacitivos se necessário.

REFERÊNCIAS BIBLIOGRÁFICAS

NEWTON C. Braga. Artigos sobre eletrônica e microcontroladores. Disponível em: https://newtoncbraga.com.br/. Acesso em: 12 jun. 2025.

PEREIRA, Fábio. Microcontroladores MSP430: teoria e prática. 1. ed. São Paulo: Érica, 2005.

TEXAS INSTRUMENTS. MSP430F2xx, MSP430G2xx family user's guide. Dallas, dez. 2004. Disponível em: https://www.ti.com/lit/ug/slau144j/slau144j.pdf. Acesso em: 12 jun. 2025.

TEXAS INSTRUMENTS. MSP430F2xx, MSP430G2xx family user's guide (Rev. K). Dallas, [s.d.]. Disponível em: https://www.ti.com/lit/ug/slau144k/slau144k.pdf. Acesso em: 12 jun. 2025.

TEXAS INSTRUMENTS. MSP430G2x53, MSP430G2x13 mixed signal microcontroller: datasheet. Dallas, abr. 2011. Disponível em: https://www.ti.com/lit/gpn/MSP430G2553. Acesso em: 12 jun. 2025.

TEXAS INSTRUMENTS. MSP430 flash memory characteristics (Rev. B). Dallas, [s.d.]. Disponível em: https://www.ti.com/lit/an/slaa334b/slaa334b.pdf. Acesso em: 12 jun. 2025.