Documentação da Biblioteca `gpio_irq_manager`

Documentação da Biblioteca `gpio_irq_manager`

Introdução

A biblioteca **gpio_irq_manager** foi desenvolvida com o objetivo de **encapsular e modularizar o tratamento de interrupções de GPIO** no Raspberry Pi Pico. Ela foi criada para facilitar o gerenciamento de interrupções em projetos que envolvem múltiplos periféricos, como **joysticks, botões, sensores digitais** e outros dispositivos que dependem de eventos em tempo real. Ao abstrair a complexidade de configurar e gerenciar interrupções, a biblioteca permite que o desenvolvedor se concentre na lógica de alto nível do projeto, sem se preocupar com detalhes de hardware.

A biblioteca é especialmente útil em sistemas embarcados, onde a eficiência e a organização do código são críticas. Ela oferece uma interface simples para registrar e remover callbacks, além de implementar um mecanismo de **debounce** para evitar leituras falsas causadas por ruídos ou vibrações mecânicas.

Utilidade da Biblioteca

A biblioteca **gpio_irq_manager** foi projetada para ser utilizada em cenários onde é necessário tratar interrupções de GPIO de forma modular e organizada. Alguns exemplos de aplicações incluem:

1. **Controle de Joysticks:**
   - Gerenciar interrupções geradas por movimentos ou pressionamentos de botões em joysticks.
2. **Botões e Interruptores:**
   - Tratar eventos de pressionamento de botões ou mudanças de estado em interruptores, com suporte a debounce para evitar leituras falsas.
3. **Sensores Digitais:**
   - Detectar mudanças de estado em sensores digitais, como sensores de proximidade ou sensores de toque.
4. **Sistemas de Automação:**
   - Monitorar eventos em sistemas de automação residencial ou industrial, como acionamento de relés ou detecção de falhas.
5. **Interfaces de Usuário:**
   - Implementar interfaces simples com botões ou outros dispositivos de entrada que dependem de interrupções para respostas rápidas.

Funcionalidades Principais

A biblioteca **gpio_irq_manager** oferece as seguintes funcionalidades:

1. **Registro de Callbacks:**
   - Permite registrar uma função de callback para ser executada quando ocorre uma interrupção em um pino GPIO específico.
2. **Remoção de Callbacks:**
   - Permite remover um callback previamente registrado, desabilitando a interrupção para o pino GPIO correspondente.
3. **Debounce Integrado:**
   - Implementa um mecanismo de debounce para evitar leituras falsas causadas por ruídos ou vibrações mecânicas.
4. **Inicialização Simplificada:**
   - Configura automaticamente o sistema de interrupções do Raspberry Pi Pico, permitindo que o desenvolvedor comece a usar a biblioteca rapidamente.
5. **Modularidade:**
   - A biblioteca foi projetada para ser modular, permitindo que seja facilmente integrada em projetos maiores ou combinada com outras bibliotecas.

Documentação das Funções

Função: `void gpio_irq_handler(uint gpio, uint32_t events)`

Propósito:
Função de tratamento de interrupção que é chamada automaticamente pelo sistema de interrupções do Raspberry Pi Pico quando ocorre uma interrupção em um pino GPIO.

Parâmetros:

- `uint gpio`: O número do pino GPIO que gerou a interrupção.
- `uint32_t events`: O tipo de evento que causou a interrupção (por exemplo, borda de subida ou descida).

Comportamento:

1. Verifica se o pino GPIO é válido (0 a 29) e se há um callback registrado para ele.
2. Obtém o tempo atual usando `get_absolute_time()`.
3. Verifica se o tempo desde a última interrupção é maior que o tempo de debounce (200 ms).

4. Se o tempo de debounce for respeitado, atualiza o tempo da última interrupção e chama a função de callback correspondente ao pino.

Função: `void register_gpio_callback(uint gpio, void (*callback)(void), uint32_t event_mask)`

Propósito:
Registra uma função de callback para ser chamada quando ocorre uma interrupção em um pino GPIO específico.

Parâmetros:

- `uint gpio`: O número do pino GPIO para o qual o callback será registrado.
- `void (*callback)(void)`: A função de callback que será executada quando a interrupção ocorrer.
- `uint32_t event_mask`: O tipo de evento que deve gerar a interrupção (por exemplo, `GPIO_IRQ_EDGE_FALL` para borda de descida).

Comportamento:

1. Verifica se o pino GPIO é válido (0 a 29).
2. Armazena a função de callback no vetor `callbacks`.
3. Habilita a interrupção para o evento especificado usando `gpio_set_irq_enabled()`.

Função: `void remove_gpio_callback(uint gpio, uint32_t event_mask)`

Propósito:
Remove um callback previamente registrado para um pino GPIO, desabilitando a interrupção para o evento especificado.

Parâmetros:

- `uint gpio`: O número do pino GPIO para o qual o callback será removido.
- `uint32_t event_mask`: O tipo de evento para o qual a interrupção será desabilitada.

Comportamento:

1. Verifica se o pino GPIO é válido (0 a 29).
2. Remove a função de callback do vetor `callbacks`, definindo-a como `NULL`.
3. Desabilita a interrupção para o evento especificado usando `gpio_set_irq_enabled()`.

Função: `void gpio_irq_manager_init()`

Propósito:
Inicializa o gerenciador de interrupções de GPIO, configurando a função de tratamento de interrupções

e habilitando interrupções no banco de GPIOs.

Comportamento:

1. Configura a função `gpio_irq_handler` como o callback global para interrupções de GPIO.
2. Habilita interrupções no banco de GPIOs usando `irq_set_enabled()`.

Conclusão

A biblioteca **gpio_irq_manager** foi criada para **encapsular e modularizar o tratamento de interrupções de GPIO**, tornando-a uma ferramenta essencial para projetos que envolvem múltiplos periféricos, como joysticks, botões e sensores digitais. Ao abstrair a complexidade de configurar e gerenciar interrupções, a biblioteca permite que o desenvolvedor se concentre na lógica de alto nível do projeto, sem se preocupar com detalhes de hardware.

Com essa documentação detalhada, você está pronto para integrar a biblioteca **gpio_irq_manager** em seus projetos e aproveitar ao máximo suas funcionalidades. Para mais detalhes, consulte o arquivo `gpio_irq_manager.c` e `gpio_irq_manager.h`.