Leitura de ADC com DMA no ESP32

Descrição do Código

O buffer DMA no ESP32 funciona como um buffer circular ao utilizar o driver adc_continuous. Isso significa que o DMA escreve dados no buffer diretamente e, ao atingir o final, retorna ao início, sobrescrevendo os dados mais antigos. O driver gerencia a sincronização entre escrita e leitura para evitar conflitos.

Escrita no Buffer:

- O DMA escreve sequencialmente no buffer especificado.
- Quando o fim do buffer é atingido, ele retorna ao início.
- Este comportamento é gerenciado automaticamente pelo driver adc_continuous.

Leitura do Buffer:

- O código lê os dados utilizando a função adc_continuous_read.
- Os dados são copiados do buffer DMA para o buffer local do código.
- O driver garante que a leitura só ocorre em dados que já foram completamente escritos.

Sincronização e Consistência:

- O driver evita que dados em processo de escrita sejam lidos pelo código.
- Caso a leitura não ocorra a tempo, os dados mais antigos podem ser sobrescritos.

Código Proposto

```
#include <Arduino.h>

#include <driver/adc.h>              // Biblioteca para configurar o ADC
```

Leitura de ADC com DMA no ESP32

```
#include <driver/adc_continuous.h>    // Biblioteca para habilitar o modo contínuo com DMA

#define ADC_CHANNEL ADC1_CHANNEL_0   // Canal ADC a ser usado (GPIO36)
#define NUM_SAMPLES 1024             // Quantidade de amostras no buffer
#define SAMPLE_RATE 2000             // Taxa de amostragem em Hz (ex.: 2000 amostras por segundo)

// Handle para o driver ADC contínuo
adc_continuous_handle_t adc_handle = NULL;

// Buffer DMA para armazenar os dados coletados
uint16_t adc_buffer[NUM_SAMPLES];    // Buffer de 1024 amostras

// Função para configurar o ADC e o DMA
void setupADC_DMA() {
    adc_continuous_handle_cfg_t adc_config = {
        .max_store_buf_size = NUM_SAMPLES * sizeof(uint16_t),
        .conv_frame_size = NUM_SAMPLES
    };
    adc_continuous_new_handle(&adc_config, &adc_handle);

    adc_continuous_config_t adc_channel_config = {
        .sample_freq_hz = SAMPLE_RATE,
        .conv_mode = ADC_CONV_SINGLE_UNIT_1,
```

```
        .format = ADC_DIGI_OUTPUT_FORMAT_TYPE1
    };

    adc_digi_pattern_config_t channel_pattern = {
        .atten = ADC_ATTEN_DB_11,
        .channel = ADC_CHANNEL,
        .unit = ADC_UNIT_1,
        .bit_width = ADC_WIDTH_BIT_12
    };

    adc_channel_config.pattern_num = 1;
    adc_channel_config.adc_pattern = &channel_pattern;

    adc_continuous_config(adc_handle, &adc_channel_config);
    adc_continuous_start(adc_handle);
}

void setup() {
    Serial.begin(115200);
    delay(1000);
    Serial.println("Iniciando leitura ADC com DMA...");

    setupADC_DMA();
}
```

Leitura de ADC com DMA no ESP32

```
void loop() {
    size_t bytes_read = 0;

       esp_err_t ret = adc_continuous_read(adc_handle, adc_buffer, sizeof(adc_buffer),
&bytes_read, 1000);

    if (ret == ESP_OK) {
        Serial.println("Dados coletados:");
        for (size_t i = 0; i < bytes_read / sizeof(uint16_t); i++) {
            Serial.println(adc_buffer[i]);
        }
    } else if (ret == ESP_ERR_TIMEOUT) {
        Serial.println("Timeout ao coletar dados do ADC!");
    }
}
```