Resumo ASE

-

Map Of Content

1. Type of Data Transfer
    1. Polling
    2. Interruptions
    3. DMA (Direct Memory Access)
2. Timers
    1. General Purpose Timer
    2. High Resolution Timer
    3. RTC timer
    4. Watchdogs
3. Peripheral
    1. General Purpose Input/Output (GPIO)
        - PWM
    2. ADC (Analog-to-Digital Converter)
    3. DAC (Digital-to-Analog Converter)
    4. Wireless Communication
        - Wi-Fi
        - Bluetooth
4. Interfaces
    1. I2C
    2. SPI
    3. UART
5. C
    1. Type Of Variabes
    2. Compilation
6. FreeRTOS
    1. FreeRTOS

2. Power Management
3. File System on Flash
4. OTA (Updates Over The Air)
5. Wireless Connectivity

7. Components used in class
    1. TC74 Sensor
    2. Display 7 segments
    3. SD card Reader
    4. Inversion of Logic Gate with MOSFETs

Type of Data Transfer

- Data Transfer is where you move data from internal components of the microcontroller itself, such as, moving data from registers of the CPU into or from the memory or peripherals.
- Besides you can transfer data between components without CPU interaction as long as the data bus is free.
- There are 3 type of data transfer essentially:
    1. Polling
    2. Interruptions
    3. DMA (Direct Access Memory)

Polling

- The CPU takes initiative, where it starts and controls the data transfer.
- In polling, the CPU actively check the status of a task or peripheral to see if the expecting data is ready to be transferred. However while it's waiting for the peripheral to be ready, it will steal clock cycles where it could be used for execution of instructions.
- **Advantages**:
    - Simple, to implement it, we use continuously loops, checking a flag or register in the peripheral to see if it has data available.
- **Disadvantages**:
    - Can be inefficient for slow peripherals or frequent data transfers

    - The processor wastes time constantly checking the peripheral, even if no data is ready. -> High Overhead

Interruptions

- In interruptions when the peripheral is ready to transfer data it will signal the CPU with a flag informing that the data in the peripheral is available.
- When the flag for interruptions in the CPU is signaled, it will abandon temporarily the execution of the program and execute the code that the interrupt handler is pointing to.
- The data transfer is made by the CPU but the busy waiting disappears, once it only occurs when the peripheral is ready.
- **Advantages:**
    - The CPU only spends time handling data transfer when necessary, improving overall performance.
- **Disadvantages**
    - Might introduce slight delays in handling the interrupt compared to polling continuously.

DMA (Direct Memory Access)

- Data is sent using DMA and DMA Controller, which is only supported by hardware and does not involve the CPU.
- During the data transfer, the DMA as the capability to control the address bus, the data bus and the control bus, because to transfer data it only needs to know:
    - source address where it will read the data
    - destination address where it will write the data
    - and the size in bytes of the data
- To transfer data, the most simple way, is to have a struct with content, # bytes/word that have been transferred and size of data to be transferred.
- **Data Transfer**
    1. DMAC ask for permission to be the bus master for the address,data and control bus
    2. wait for permission granted from the CPU
    3. while transfering:

        1. read a byte/word from the source address to a internal register of the DMA
        2. writes the data of the internal register into the destination address
        3. increments the # of bytes/word transferred
        4. loop while # bytes/words != size
    4. DMA remove the permission of the bus master and the buses and signal the CPU that they are free
- **Advantages:**
    - Reduces processor load significantly.
- **Disadvantages:**
    - Not suitable for all data transfer scenarios, particularly small data transfers.

Timers

-

References

1. https://docs.espressif.com/projects/esp-idf/en/stable/esp32c3/get-started/index.html

ATENCAO

- **Hardware timers** are the physical timer circuits on the ESP32C3 chip.
- **Software (ESP-IDF libraries)** allows you to interact with these hardware timers and define their behavior.
- **Counter:** This is indeed a core component. Both GPTs and the ESP Timer have a counter register. This register is what actually increments or decrements based on the timer configuration.
    - In the ESP32C3, GPTs are 54-bit counters, while the ESP Timer is a 64-bit counter.

- **Pre-scaler (GPTs only):** GPTs have an additional register called the pre-scaler. This allows you to divide the clock signal feeding the timer, effectively slowing down the counter's increment/decrement rate. The ESP Timer doesn't have a pre-scaler.
- **Compare Register:** This register is present in both GPTs and the ESP Timer. You can set a value in this register. When the counter reaches the value in the compare register, an event occurs (like an interrupt). This allows you to generate periodic events based on the timer.
- **Auto-reload:** Both GPTs and the ESP Timer offer auto-reload functionality. When enabled, upon reaching the compare register value, the counter automatically resets to zero and starts counting again. This is essential for creating periodic signals or tasks.