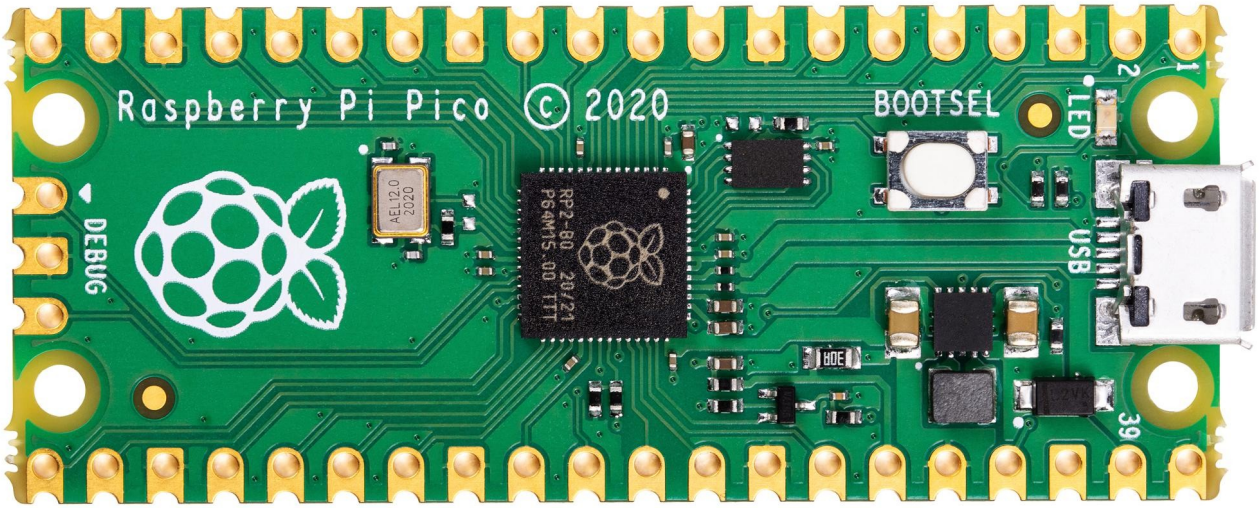
PIO (Programmable Input/Output)



A PIO é um dos periféricos mais interessantes e único do rppico (Raspberry pi Pico) porque ele levou a comunicação pelos pinos GPIOs do microcontrolador a outro nível, e um nivel muito elevado de sofisticação e desempenho.

A PIO tem quatro máquinas de estado, eu acredito que todos que tenham interesse por esse artigo deve saber oque é uma máquina de estado, por isso não vou explicar em profundidade esse quesito.

A máquina de estado da PIO é totalmente independente da cpu e essa característica é muito importante para muitas aplicações, porque você põe dados e ou retira dados dos pinos independentemente da cpu isso significa que o tempo entre coletar o dado no pino e disponibiliza-lo na memoria interna e ou executar alguma outra ação é muito menor do que se fosse a cpu lendo o dado no pino, esse é um dos motivos pelo qual pipocou na internet muitos projetos usando o rppico gerando sinais de vídeo.

As máquinas de estado são pequenos “processadores” dedicados a ler e ou escrever nos pinos, e com isso houve a necessidade de ter um programa para elas, assim uma linguagem para isso. A linguagem usada para programar a PIO é uma **linguagem assembly específica** chamada **PIO Assembly**.

O que é PIO Assembly?

A PIO Assembly é uma linguagem de baixo nível usada para escrever programas que rodam diretamente nas máquinas de estado (state machines) da PIO. Cada máquina de estado da PIO pode executar um

programa independente, permitindo criar protocolos de comunicação personalizados, controlar GPIOs com precisão de tempo e muito mais.

Esse é um exemplo de programa em assembly para fazer um led piscar.

```
.program blink
    set pindirs, 1    ; Define o pino como saída

again:
    set pins, 1 [31]      ; Liga o pino (HIGH)
    set pins, 0 [31]      ; Desliga o pino (LOW)
    jmp again         ; Repete o loop
```

Observem que o programa não tem um delay entre as linhas que ligam e desligam o led, essa é uma das características muito interessantes do assembler da PIO, pois como ele é totalmente independente da cpu o que é feito nessa situação é configurar a máquina de estado para um clock bem baixo.

Vamos entender tudo isso:

set pins, 1 [31]

Essa linha significa o seguinte, set pins, 1 é fácil de saber oque faz, esse comando assembly diz para por o pino apontado por pins com o valor 1 ou seja 3,3V deve aparecer no pino em questão, agora a diretiva [31] é que é muito interessante, pois ela diz que o programa ficará parado por 31 ciclos do clock, UAUUUUUU… que maravilha e a cpu não ficou parada por isso. Então imagina que você tenha configurado a maquina de estado para um clock com um ciclo de 5ms(mili segundos) 5 x 31 = 155ms esse é o tempo que o led ficará apagado e depois aceso, ou seja o led vai piscar a uma taxa de aproximadamente 6 vezes por segundo, mas sua cpu está rodando a 133MHz lembre-se.
E tem mais sobre isso ainda, cada linha de código no assembly da PIO é totalmente executada em apenas 1 ciclo de clock, com isso fica muito fácil de fazer um program muito deterministico em relação ao tempo de execução das tarefas.

Características da PIO no Raspberry Pi Pico

1. **Blocos de PIO**:
    - O RP2040 possui **2 blocos de PIO independentes**, cada um com 4 máquinas de estado (state machines), totalizando **8 máquinas de estado programáveis**.
2. **Máquinas de Estado (State Machines)**:
    - Cada máquina de estado é um processador independente que executa um programa PIO.
    - As máquinas de estado podem ser programadas para implementar protocolos de comunicação personalizados ou manipular sinais de GPIO de forma altamente eficiente.
3. **Instruções PIO**:
    - A PIO possui um conjunto de **9 instruções básicas** que podem ser usadas para criar programas personalizados.
    - As instruções incluem operações como leitura/escrita de GPIO, manipulação de registradores, controle de fluxo (loops e branches), e sincronização com clocks externos.
    - O conjunto de instruções é simples, mas poderoso, permitindo a criação de protocolos complexos.
4. **Flexibilidade de GPIO**:
    - As máquinas de estado da PIO podem controlar **qualquer pino GPIO** do RP2040.
    - Isso permite que você use a PIO para implementar protocolos personalizados em praticamente qualquer pino do Pico.
5. **Velocidade e Precisão**:
    - A PIO opera de forma independente da CPU principal, permitindo operações de alta velocidade e baixa latência.
    - Ela pode manipular sinais com precisão de **1 ciclo de clock**, o que é ideal para protocolos de comunicação rápidos ou controle de dispositivos sensíveis ao tempo.

6. **Protocolos Suportados**:
    - A PIO pode ser usada para implementar uma variedade de protocolos de comunicação, como:
        - **Protocolos seriais**: UART, I2C, SPI, 1-Wire, etc.
        - **Protocolos de controle**: PWM, controle de LEDs (WS2812/NeoPixel), controle de motores, etc.
        - **Protocolos personalizados**: Você pode criar seu próprio protocolo para atender às necessidades específicas do seu projeto.
7. **Memória de Programa**:
    - Cada bloco de PIO possui uma **memória de programa compartilhada** de 32 instruções.
    - Como há 4 máquinas de estado por bloco, é possível compartilhar o mesmo código entre várias máquinas ou programar cada máquina de forma independente.
8. **Clock Divisor**:
    - Cada máquina de estado pode ser configurada com um **divisor de clock** para ajustar a velocidade de operação.
    - Isso permite que a PIO opere em velocidades mais baixas, se necessário, para compatibilidade com dispositivos externos.
9. **Interrupções**:
    - A PIO pode gerar interrupções para a CPU principal, permitindo sincronização entre operações de hardware e software.
10. **Exemplos de Uso**:
    - Implementação de protocolos não suportados nativamente pelo RP2040 (ex: DPI, VGA, MIDI).
    - Controle de LEDs endereçáveis (ex: WS2812/NeoPixel).
    - Geração de sinais PWM de alta precisão.
    - Leitura de sensores com protocolos personalizados.

Esse é o que eu queria dizer nesse artigo, não vou entrar em detalhes de como configurar o SDK do rppico no Visual Studio Code porque tem uma tonelada de tutoriais na internet explicando como fazê-lo e acredite-me é simples.

No próximo artigo entrarei em detalhes da programação da PIO.

Até lá.

Paulo da Silva (pgordão).

Técnico em Eletrônica.

Analista de Sistemas.

31/JAN/2025