

## Passos básicos

### 1. Definir a direção do pino e propriedades

- Entrada/ Saída
- Pullup ? pullDown ? ...

### 2. Identificar o pino e seu PIO

- ex: PC18

### 3. Extrair para o firmware esses dados

• defines

### 4. Ativar PMC para controlar pino

- `pmc_enable_periph_clk()`

### 5. Configurar PIO para gerenciar pino no modo correto

- `pio_output()`
- `pio_input()`

### 6. Agir/ler o pino

- `pio_set()` / `pio_clear()`
- `pio_get()`

## Não consigo ler uma entrada

Está com problema em ler uma entrada? Siga os seguintes passos de debug:

### 1. Verifique se o pino que está tentando ler é o correto.

- Muitas vezes decidimos por ler um pino mas acabamos por ligar o sinal que desejamos ler no lugar errado.

### 2. Verifique se passou as informações corretas para o código (PIO/ PIO\_ID/ Máscara) estão corretas?

3. Você está executando em um código que funciona? Pegue um código bem simples apenas para testar o pino, só quando funcionar incorpore o mesmo no seu projeto.
4. Pode ser algum problema na conexão. Remova qualquer conexão do pino e ligue um jumper a ele. Conecte esse jumper ao `gnd` da placa. Execute o firmware e você deve ler 0, agora conecte o pino ao `3,3` da placa e você deve ler `1`.
5. O sinal que você pretende ler precisa de `pull-up` / `pull-down` ?
6. O pino pode estar queimado (essas coisas acontecem), mude de pino e teste novamente.
7. Você está executando em um código que funciona? Pegue um código bem simples apenas para testar o pino, só quando funcionar incorpore o mesmo no seu projeto (está repetido para garantir que você leu).

## Estrutura de código

A seguinte estrutura é utilizado para acionarmos um pino desse microcontrolador:

### Biblioteca ASF

Todas as funções que controlam o PIO estão documentadas em:

- [http://asf.atmel.com/docs/latest/same70/html/group\\_\\_sam\\_drivers\\_\\_pio\\_group.html](http://asf.atmel.com/docs/latest/same70/html/group__sam_drivers__pio_group.html)

### Periférico ID (`ul_id`)

O ID do PIO (ou de qualquer outro periférico) é um número inteiro único que identifica o periférico. Esse valor pode ser extraído do manual do microcontrolador (secção 13), ou utilizando o valor já definido no arquivo `.h` de configuração do uC. Exemplo:

```
#define LED_PIO_ID 12 // PIOC possui ID 12
```

Podemos usar o ID que já foi definido no arquivo `.h`:

```
#define LED_PIO_ID ID_PIOC
```

Sendo esse segundo método mais aconselhável pois possibilita maior portabilidade do código.

### Máscara (ul\_mask)

A máscara é utilizada para configurarmos apenas alguns bits específicos. Vamos considerar o exemplo do LED do kit SAME70-XPLD:

```
#define LED_PIO_IDX      8u                // ID do LED no PIO
#define LED_PIO_IDX_MASK (1u << LED_PIO_IDX) // Mascara para
CONTROLAMOS o LED
```

Podemos ler a linha que define o `LED_PIO_IDX_MASK` como: pegue o valor em binário **1** ( `0000 0000 0000 0001` ) e desloque (todos os bits) oito casas para direita:

```
1u = 0000 0000 0000 0001
                        <--- LED_PIO_IDX vezes
(8x)
resultando:

LED_PIO_IDX_MASK = 0000 0001 0000 0000
                    ^
                    |
                    | Apenas esse bit está 'ativado'
```

nosso uC é de 32 bits, nesse exemplo estamos exibindo apenas 16 bits.

### p\_pio

Explicar o `p_pio`, ponteiro organizado em forma de struct que ordena um endereço de memória....