# Periféricos
# PIO

Rafael Corsi Ferrão corsiferrao@gmail.com

31 de março de 2016

## 1 Periféricos - Secção 11 datasheet

Periféricos são hardwares auxiliares encontrados no uC que fornecem funcionalidades extras tais como : gerenciador de energia (SUPC), comunicação serial UART (UART), comunicação a dois fios (TWI), controlador de saída e entrada paralela (PIO), dentre muitos outros.

**1 - Periféricos**

Liste a funcionalidade dos periféricos listados a seguir :
1. RTC - Real time clock / 2. TC - Timer/Counter

Os periféricos são configuráveis via escrita/leitura dos registradores, sendo que cada periférico possui um endereço. Os endereços reservados para os periféricos começa no valor 0x40000000 e termina no 0x5FFFFFFF como ilustrado na Fig. 1.

Um mapa mais detalhado que envolve os endereços dos periféricos pode ser encontrado na Pag. 38 do datasheet.

**1 - Periféricos**

Encontre os endereços referentes aos seguintes periféricos :
1. PIOA / 2. PIOB / 3. ACC / 4. UART1 / 5. UART2

## 2 PIO - Secção 31 datasheet

No ARM alguns pinos são gerenciados por um hardware chamado de Parallel Input/Output Controller (PIO), esse dispositivo é capaz de gerenciar até 32 pinos (I/O).

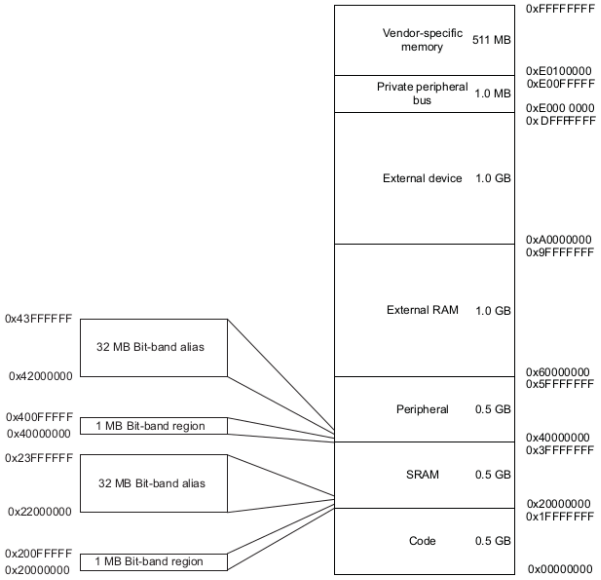


Figura 1: [Datasheet pg. 68] Memory Map

Cada I/O no ARM pode ser associado a uma função diferente (periférico), por exemplo: o I/O *PA20* pode ser controlador pelo periférico do PWM enquanto o *PA18* pode ser controlador pela UART.

Isso fornece flexibilidade ao desenvolvimento de uma aplicação, já que o I/O não possui uma funcionalidade fixa. Porém não possuímos tanta flexibilidade assim, existe uma relação de quais I/Os os periféricos podem controlar.

Os I/Os são classificados por sua vez em grandes grupos: A, B,C .... (exe: PA01, PB22, PC12) e cada grupo é controlado por um PIO (PIOA, PIOB, PIOC, ...).

A tabela na página 52, 53 e 54 do datasheet ilustra quais periféricos podem ser associados aos respectivos pinos, a Fig. 2 mostra algumas opções para o PIOA.

**Table 11-2. Multiplexing on PIO Controller A (PIOA)**

| I/O Line | Peripheral A | Peripheral B | Peripheral C | Peripheral D[1] | Extra Function | System Function | Comments |
|---|---|---|---|---|---|---|---|
| PA0 | PWMH0 | TIOA0 | A17 | | WKUP0[2] | | |
| PA1 | PWMH1 | TIOB0 | A18 | | WKUP1[2] | | |
| PA2 | PWMH2 | SCK0 | DATRG | | WKUP2[2] | | |
| PA3 | TWD0 | NPCS3 | | | | | |
| PA4 | TWCK0 | TCLK0 | | | WKUP3[2] | | |
| PA5 | RXD0 | NPCS3 | | | WKUP4[2] | | |
| PA6 | TXD0 | PCK0 | | | | | |
| PA7 | RTS0 | PWMH3 | | | | XIN32[3] | |
| PA8 | CTS0 | ADTRG | | | WKUP5[2] | XOUT32[3] | |
| PA9 | URXD0 | NPCS1 | PWMFI0 | | WKUP6[2] | | |
| PA10 | UTXD0 | NPCS2 | PWMFI1[1] | | | | |

Figura 2: Mux PIOA - periféricos vs pinos

**1 - I/O pinos**

Verifique no datasheet do uC os pinos físicos do uC associados aos I/O: PA01, PB22 e PC12.

**1 - PIO periféricos**

Verifique quais periféricos podem ser configuráveis nos I/Os :
1. PC20 / 2. PB3.

## 2.1 Configurações

O PIO suporta as seguintes configurações :

- Interrupção em nível ou borda em qualquer I/O
- Filtragem de "glitch"
- Deboucing
- Open-Drain
- Pull-up/Pull-down
- Capacidade de trabalhar de forma paralela

**Deboucing**

- O que é deboucing ?
- Descreva um algorítimo que implemente o deboucing.

## 2.2 Funcionalidade

O diagrama de blocos do PIO é ilustrado na Fig. 3.

Podemos a partir do diagrama de blocos tirar as seguintes conclusões :

1. Peripheral DMA (direct memmory access) controller (PDC) : O P/IO pode receber dados via DMA.
2. Interrup Controller : Já que o PIO suporta interrupções nos I/Os o mesmo deve se comunicar com o controlador de interrupções para informar a CPU (NVIC) que uma interrupção ocorreu.
3. PMC : A energia e clock desse periférico é controlado pelo PMC (Power management controller).

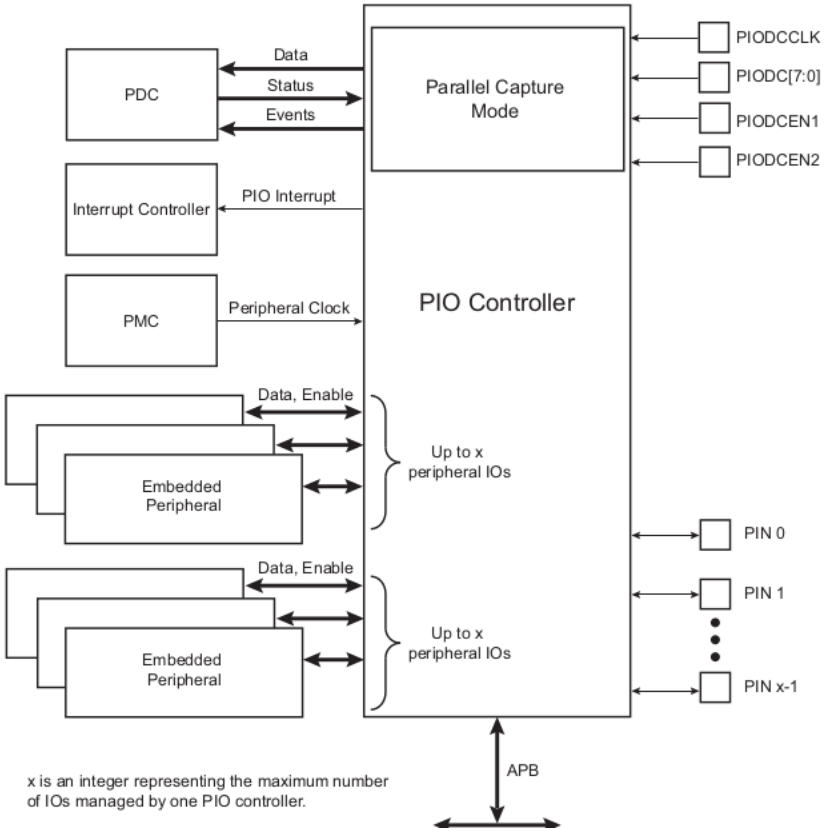


Figura 3: [Datasheet pg. 569] Block Diagram

4. Embedded peripheral : O acesso aos pinos dos periféricos é realizado via PIO.

Jm diagrama lógico mais detalhado é encontrado no datasheet (Fig. 4), esse diagrama mostra as funções dos registradores e seu impacto no PIO.

## 2.3 SET/Clear

Algumas funcionalidades no PIO são configuráveis via dois registradores distintos (set/clear), o primeiro apenas altera o estado do bit específico de 0 para 1, enquanto que o segundo (clear) altera o estado do registrador de 1 para 0.

Isso é utilizado para evitar uma condição de corrida ("race condicions"*).

Um exemplo desse caso é o registrador Output Data Status Register (PIO_ODSR) que configura se um pino será estado "1"ou "0". Para configurarmos por exemplo o bit 2 (referente ao pino PIOA02) precismos "setar"o bit 2 via o registrador Set Output Data Register (IO_SODR):

```
PIOA->SODR = 1 << 1;
```

*http://stackoverflow.com/questions/34510/what-is-a-race-condition/34745#34745)

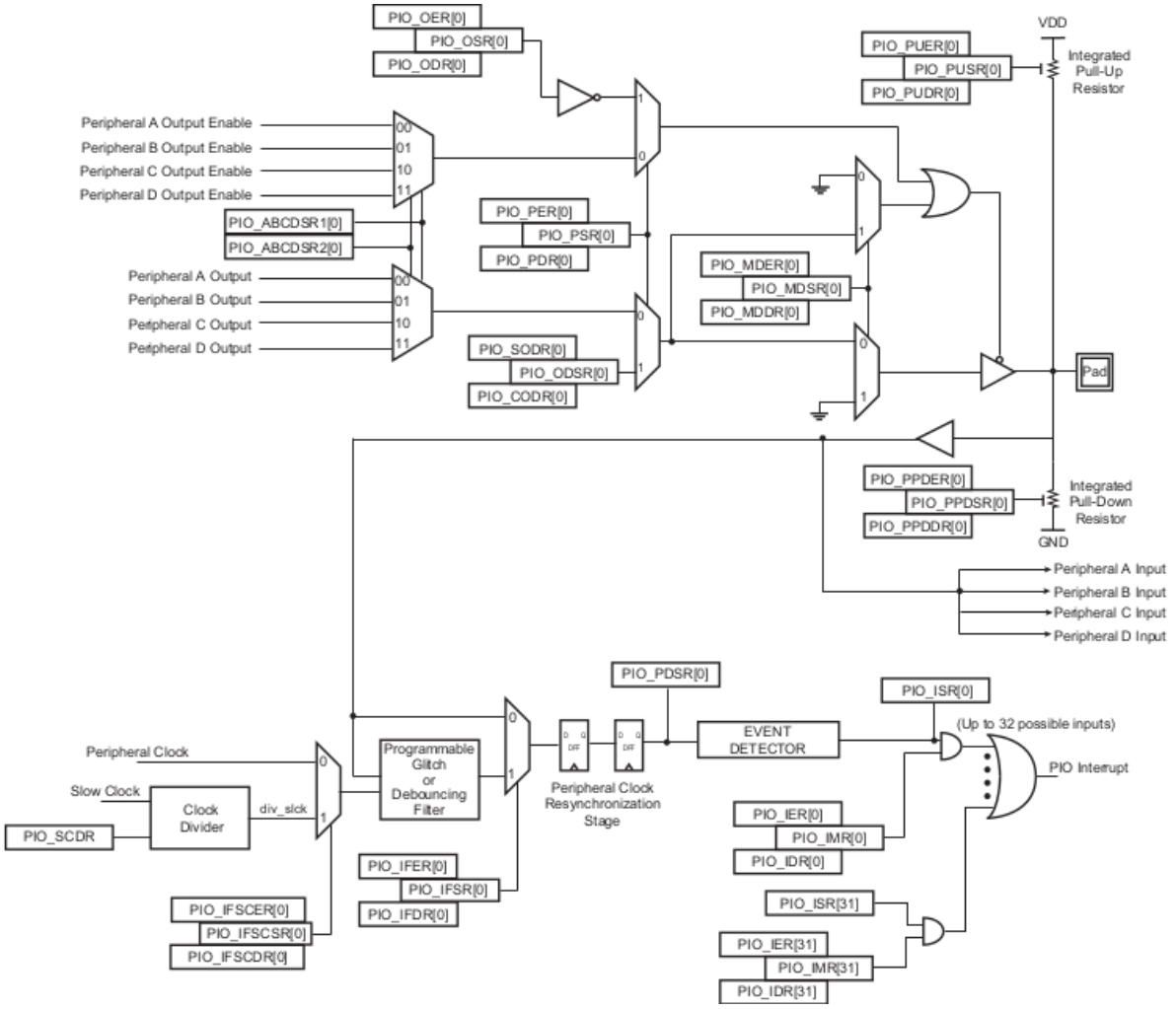


Figura 4: [Datasheet pg. 571] I/O Line Control Logic

Não podemos por exemplo zerar esse registrador :

```
PIOA->SODR &= ~(1 << 1);
```

Para tanto é necessário utilizarmos o registrador Clear Output Data Register (CODR)

```
PIOA->CODR = 1 << 1;
```

**race condicions**

- O que é race condicions ?
- Como que essa forma de configurar os registradores evita isso?

## 2.4 Configurando um pino em modo de output

31.5.4 - Output Control (datasheet pg. 573)

When the I/O line is assigned to a peripheral function, i.e., the corresponding bit in PIO_PSR is at zero, the drive of the I/O line is controlled by the peripheral. Peripheral A or B or C or D depending on the value in PIO_ABCDSR1 and PIO_ABCDSR2 determines whether the pin is driven or not.

When the I/O line is controlled by the PIO Controller, the pin can be configured to be driven. This is done by writing the Output Enable Register (PIO_OER) and Output Disable Register (PIO_ODR). The results of these write operations are detected in the Output Status Register (PIO_OSR). When a bit in this register is at zero, the corresponding I/O line is used as an input only. When the bit is at one, the corresponding I/O line is driven by the PIO Controller.

The level driven on an I/O line can be determined by writing in the Set Output Data Register (PIO_SODR) and the Clear Output Data Register (PIO_CODR). These write operations, respectively, set and clear the Output Data Status Register (PIO_ODSR), which represents the data driven on the I/O lines. Writing in PIO_OER and PIO_ODR manages PIO_OSR whether the pin is configured to be controlled by the PIO Controller or assigned to a peripheral function. This enables configuration of the I/O line prior to setting it to be managed by the PIO Controller.

Similarly, writing in PIO_SODR and PIO_CODR affects PIO_ODSR. This is important as it defines the first level driven on the I/O line.

**configurando um pino em modo saída**

- Explique com suas palavras o trecho anterior extraído do datasheet do uC, se possível referencie com o diagrama "I/O Line Control Logic".

# 3 Programação

## 3.1 Led Blue

O projeto localizado em : EEN251/Codigos/GPIO_REGISTRADORES_LED é um exemplo de como inicializar um pino (no caso o PIOA_19 - LED Azul) em modo de output e aciona lógica alta nesse pino.

1. importe o projeto para o seu repositório e rode o programa.

2. modifique o programa (utilize o registrador PIO_CODR) e faça a saída ser 0.

## 3.2 Blink

Utilizando a função da biblioteca da Atmel (ASF) *delay_ms()*, que provoca um delay no microcontrolador, implemente o LED Azul piscando a uma taxa de 1 segundo.

## 3.3 Led Azul + Verde + Vermelho - Blink

Modifique o projeto e adicione agora a configuração para o acionarmos o LED verde e vermelho.

Passos sugeridos :

1. qual os pinos que controlam os leds ?
2. quais os PIOs que controlam os leds ?
3. trabalhe com #defines para deixar o código mais flexível
4. implemente a configuração do LED Verde
    - uma vez configurado faça os dois leds piscarem
5. implemente a configuração do LED Vermelho
    - uma vez configurado faça os três leds piscarem