Insper

Computação Embarcada - IOs (Input/Output)

Rafael Corsi - rafael.corsi@insper.edu.br

Fevereiro 2018

Entrega até o começo da aula do dia ..

Visão Geral

(*Utilize o manual encontrado em :* Manuais/**Cortex-M7-SAM-E70*** para resolução dessa secção.)*

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

Liste a funcionalidade dos periféricos listados a seguir :

- RTC - Real time clock
- TC - Timer/Counter

Os periféricos são configuráveis via escrita/leitura nos registradores do microcontrolador, cada periférico possui um endereço único mapeado em memória. A Fig. [fig:same70pg41-memmappdf] define os endereços de memórias reservado para cada funcionalidade do uC.

2

- Qual endereço de memória reservado para os periféricos ?
- Qual o tamanho (em endereço) dessa secção ?

O diagrama completo do mapeamento de memória pode ser encontrado na página 41.

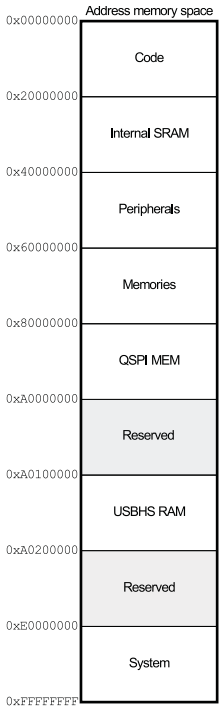

Figure 1: Mapa de memória SAME70 *(pg. 41)*

3

Encontre os endereços de memória referentes aos seguintes periféricos :

1. PIOA
2. PIOB
3. ACC
4. UART1
5. UART2

PMC - Gerenciador de energia

O Power Management Controller (PMC) é um periférico responsável por gerenciar a energia e clock dos demais periféricos. Para utilizarmos um periférico é necessário primeiramente ativarmos o mesmo no PMC.

Cada periférico é referenciado no PMC via um número único (ID), esse ID também será utilizado para o gerenciamento de interrupções. Os IDs estão listados na Tabela : 13.1 do datasheet Cortex-M7-SAM-E70.pdf.

4

Qual ID do PIOC ?

Parallel Input Output (PIO)

Secção 32 datasheet

Leitura obrigatória, Secção 32 do datasheet.

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

Além do controle direito do pino pelo PIO, cada I/O no ARM-Atmel 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 os I/Os não possuem uma funcionalidade fixa. Porém não possuímos tanta flexibilidade assim, existe uma relação de quais I/Os os periféricos podem controlar.

Podemos interpretar a tabela como : o pino **102** do microcontrolador identificado como **PA0** (PIOA_0) pode ser utilizado como **WKUP0** (wakeup) ou mapeado para um dos tres perifericos :

- Periferico A: PWM (Pulse width modulation)
- Periferico B: TIOA0 (Timer 0)
- Periferico C: I2C_MCL (I2C master clear)

A tabela na página 16 do datasheet (Table 5-1) ilustra quais periféricos podem ser associados aos respectivos pinos, a Fig. [fig:PIOA_mux] mostra as opções para o PIOA0 até PIOA9.

Table 5-1. 144-lead Package Pinout

LQFP Pin	LFBGA Ball	UFBGA Ball	Power Rail	I/O Type	Primary		Alternate		PIO Peripheral A		PIO Peripheral B		PIO Peripheral C		PIO Peripheral D		Reset State
					Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal	Dir	Signal, Dir, PU, PD, HIZ, ST
102	C11	E11	VDDIO	GPIO_AD	PA0	I/O	WKUP0[1]	I	PWMC0_PWMH0	O	TIOA0	I/O	A17/BA1	O	I2SC0_MCK	–	PIO, I, PU, ST
99	D12	F11	VDDIO	GPIO_AD	PA1	I/O	WKUP1[1]	I	PWMC0_PWML0	O	TIOB0	I/O	A18	O	I2SC0_CK	–	PIO, I, PU, ST
93	E12	G12	VDDIO	GPIO	PA2	I/O	WKUP2[1]	I	PWMC0_PWMH1	O	–	–	DATRG	I	–	–	PIO, I, PU, ST
91	F12	G11	VDDIO	GPIO_AD	PA3	I/O	PIODCO[2]	I	TWD0	I/O	LONCOL1	I	PCK2	O	–	–	PIO, I, PU, ST
77	K12	L12	VDDIO	GPIO	PA4	I/O	WKUP3/PIODC1[2]	I	TWCK0	O	TCLK0	I	UTXD1	O	–	–	PIO, I, PU, ST
73	M11	N13	VDDIO	GPIO_AD	PA5	I/O	WKUP4/PIODC2[2]	I	PWMC1_PWML3	O	ISI_D4	I	URXD1	I	–	–	PIO, I, PU, ST
114	B9	B11	VDDIO	GPIO_AD	PA6	I/O	–	–	–	–	PCK0	O	UTXD1	O	–	–	PIO, I, PU, ST
35	L2	N1	VDDIO	CLOCK	PA7	I/O	XIN32[4]	I	–	–	PWMC0_PWMH3	O	–	–	–	–	PIO, HIZ
36	M2	N2	VDDIO	CLOCK	PA8	I/O	XOUT32[4]	O	PWMC1_PWMH3	O	AFE0_ADTRG	I	–	–	–	–	PIO, HIZ
75	M12	L11	VDDIO	GPIO_AD	PA9	I/O	WKUP5/PIODC3[2]	I	URXD0	I	ISI_D3	I	PWMC0_PWMFI0	I	–	–	PIO, I, PU, ST

Figure 2: Mux PIOA - periféricos vs pinos pg. 16

5

Verifique quais periféricos podem ser configuráveis nos I/Os :

1. PC1
2. PB6

O SAME70 possui internamente 5 PIOs: PIO**A**, PIO**B**, PIO**C**,PIO**D** e PIO**E**. Cada um é responsável por gerenciar até 32 pinos.

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, …).

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

6

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

Funcionalidade

O diagrama de blocos do PIO é ilustrado na Fig. [fig:PIO_geral], onde :

1. Peripheral DMA (direct memmory access) controller (PDC) : O PIO 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).
4. Embedded peripheral : O acesso aos pinos dos periféricos é realizado via PIO.

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

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” [1]).

Um exemplo desse caso é o registrador Output Data Status Register (PIO_ODSR) que configura o valor de saída de um pino, ou seja, se será “1” ou “0”. Para configurarmos por exemplo o bit 2 (referente ao pino PIOA02) precismos “setar” o segundo bit via o registrador Set Output Data Register (IO_SODR):

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

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;
```

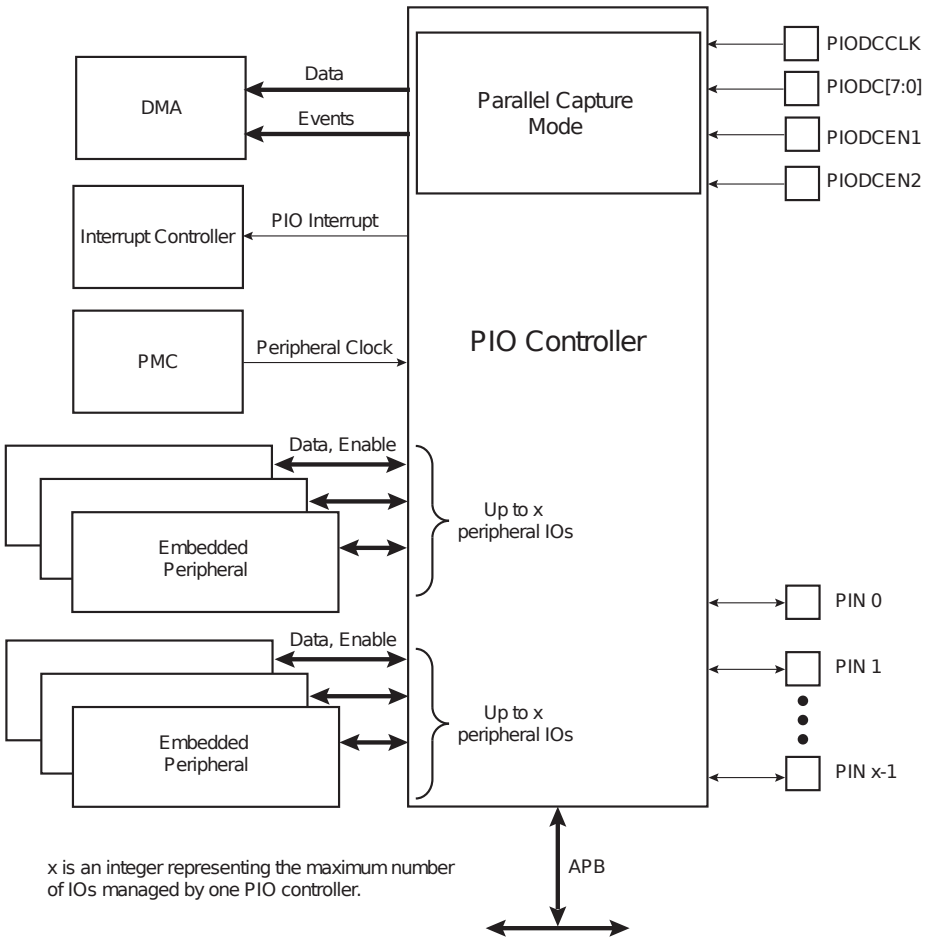

Figure 3: Block Diagram - pg. 346

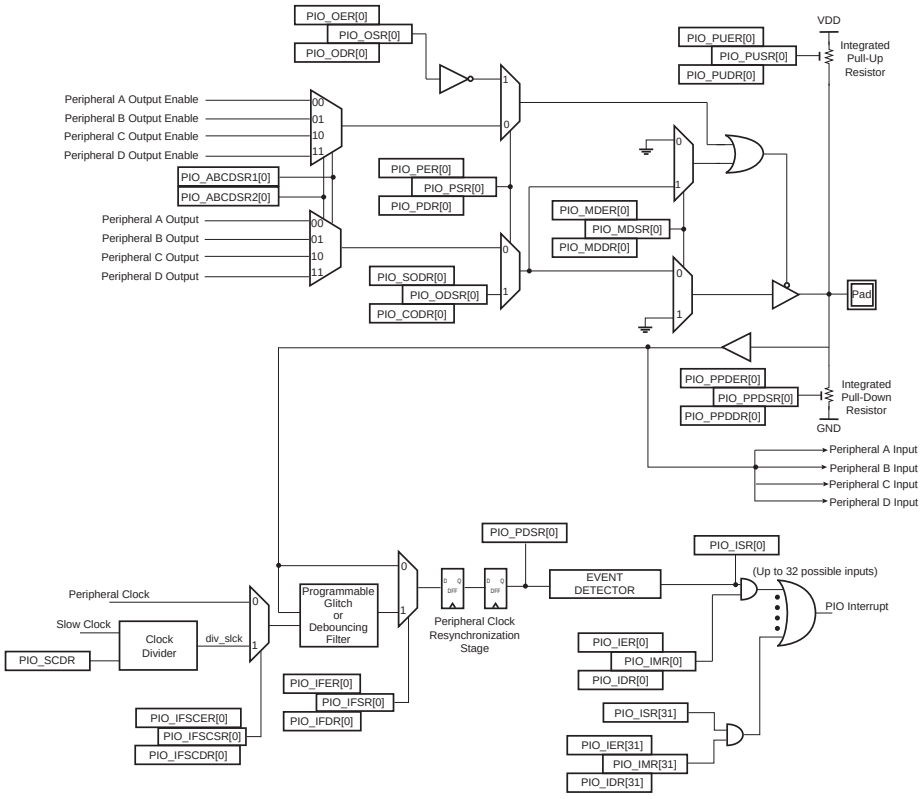

Figure 4: [Datasheet pg. 571] I/O Line Control Logic

Esse tipo de controle evita que tenhamos que fazer uma operação de leitura no registrador antes de alterar o bit específico, aumentado assim a eficiência desse periférico.

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

Configurando um pino em modo de output

```
31.5.4 - Output Control (. 550)

When the I/O line is , i.e., the corresponding bit in PIO\_PSR is at
zero, . 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.

. the Output Enable Register () and Output Disable Register (). are
detected in the Output Status Register (). When a bit in this register
is at , the corresponding I/O line is used as an . When the bit is at ,
the corresponding I/O line is .

line can be determined by in the Set Output Data Register () and the
Clear Output Data Register (). the Output Data Status Register (), .
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.
```

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