Insper

07 – Pesquisa:

PIO Output

Rafael Corsi

rafael.corsi@insper.edu.br

16 de fevereiro de 2017

Matheus Marotzke

Engenharia da Computação – INSPER – 2017

1 Periféricos

Questão. 1.1: Periféricos

- RTC - Real time clock /

  O real time clock é um circuito integrado de hardware que acompanha o tempo atual.

- TC - Timer/Counter

  Dentre as funções do TC estão:

  - Timing.
  - Periodic interrupt/event generation.
  - Pulse Width Modulation.
  - Event time stamping.
  - Event counting.
  - Signal parameter measurements (Period, duty cycle, etc.).

Questão. 1.2: Mapa de memória

Qual endereço de memória reservado para os periféricos ?

0x400000000

qual o tamanho (em endereço) dessa secção ?

0x200000000 endereços.

Questão. 1.3: Periféricos

Encontre os endereços de memória referentes aos seguintes periféricos :

1. PIOA - 0x400E0E00
2. PIOB - 0x400E1000
3. ACC - 0x40044000
4. UART1 - 0x400E0A00
5. UART2 - 0x400E1A00

2 PMC - Gerenciador de energia

Questão. 2.1: PIO D ID

Qual ID do PIOC ?

12

3 Parallel Input Output (PIO)

Questão. 3.1: PIO periféricos

Verifique quais periféricos podem ser configuráveis nos I/Os :

1. PC1 – Somente PWMC0_PWML1 (Pulse Width Modulation)

2. PB6 – Somente SWDIO/TMS

3.1 Configurações

Questão. 3.2: Debouncing

O que é deboucing ?

Fazer o que é chamado de *switch debounce* é implementar um input manual para um sistema digital, isso exige que o sistema filtre o ruído promovido pelo clique do botão, ou seja, um botão não entrega um input limpo, então fazer um *debounce* .

Descreva um algorítimo que implemente o deboucing.

Define variável de contagem count = 0

Define um evento de aquisição com uma amostragem relativamente alta.

Define variável bool de estado (PRESSIONADO)

evento de aquisição de dados:

se o sinal do botão estiver HIGH:

count = 0;

mudar qual o ESTADO = !PRESSIONADO

senão:

aumenta count++; (Até um limite, não muito alto)

se count = limite:

mudar qual o ESTADO = PRESSIONADO

3.3 SET/Clear

Questão. 3.3: Race conditions

O que é race condicions ?

Na computação, uma *race condition* ocorre quando, quase ao mesmo tempo, uma função de *write e read* são chamadas. Isso leva o computador a tentar ler uma memória enquanto está sendo escrita.

Como que essa forma de configurar os registradores evita isso?

Esse tipo de falha pode ser evitado através da serialização do acesso à memória. Isso faz com que quando dois comandos, um de leitura e um de escrita são chamados ao mesmo tempo o computador sempre executa o de leitura antes de começar a escrita.

3.4 Configurando um pino em modo de output

Questão. 3.4: Pino em modo output

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

Existem duas formas de fazer *assignment* para um I/O Line. Ou a Line será controlada por um periférico ou por um controller. No caso do periférico seu PIO_PSR (Peripheral Status Register) é zero, os valores de A,B,C,D registrados em PIO_ABCDSR1 e PIO_ABCDSR2 determinam se a *line* é ou não *driven*. Já no caso do *Controller* controlar a *line* os pinos podem ser configurados escrevendo nos Outputs de Enable e Disable (PIO_OER & PIO_ODR), ao escreve-las são detectadas pelo que contém o status PIO_OSR. Se for 0, o pino só funciona pra input, caso contrário, funciona sendo *driven* pelo I/O controllers.

A determinação de *driven* pode ser alterada mudando os valores através de gravações em PIO_ set e de clear que escrevem esses comandos no PIO_ODSR que representa os arquivos *driven* nas *lines* .