Insper | Instituto de Ensino e Pesquisa

08 - Projeto
PIO Input

Rafael Corsi
rafael.corsi@insper.edu.br

8 de março de 2017

Entregar até o final da aula

1. Criar softwares para microcontroladores utilizando suas especificidades (periféricos/ low power);
2. Avaliar e melhorar soluções embarcadas integrando hardware/software levando em conta adequação a uma aplicação;
3. Integrar em um protótipo hardware, software básico, sistema operacional de tempo real e módulos de interfaceamento com usuários, de comunicação e de alimentação;
4. Compreender as limitações de microcontroladores e seus periféricos;
5. Buscar e analisar documentação (datasheet) e extrair informações

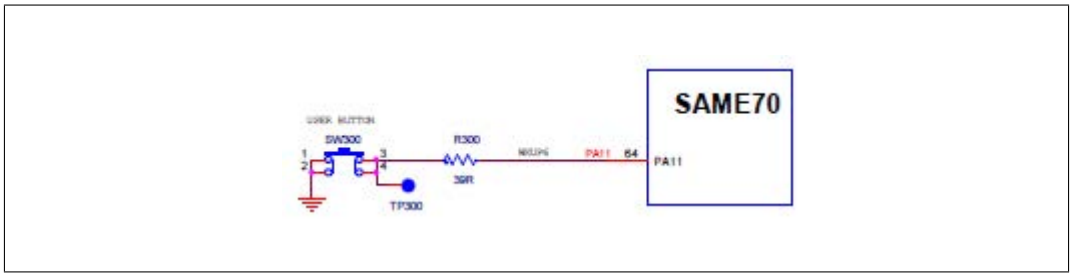
1 Entendimento

Passos necessários para o entendimento correto de como um pino digital, deve ser configurado para ser controlado pelo PIO.

1. Identifique no kit de desenvolvimento (SAME-70-XPLD) qual botão é reservado para o usuário :

   SW300

2. Via o manual do kit de desenvolvimento, redesenhe o esquema elétrico referente ao botão do usuário:



3. Identifique o pino, o PIO e o bit correspondente ao botão.

Pino	PIO	Bit
64	A	11

4. Preencha a tabela a seguir com o valor lido nos casos em que o botão não está pressionado e quando está.

Botão Pressionado (SW0)	Valor digital lido no PIO
Sim	0
Não	1

5. Identifique os registradores que considera necessário modificar (leia a secção do datasheet referente a configurar o PIO em modo de entrada).
   PIO_PER / PIO_IER / PIO_ISR

2 Programação parte 1 - SW0

Com base no projeto passado, utilize o botão reservado ao usuário para ativar o LED (Botão não pressionado = LED acesso/ Botão pressionado = Led apagado). Importe o projeto base localizado em : /codigos/08-PIO-INPUT para o seu repositório e modifique-o de acordo com os passos a seguir.

1. Importe as configurações do LED do projeto 07-PIO-INPUT. Coloque essas configurações dentro da função : ledConfig().

2. Ative o clock do PIO que controla o botão no PMC:

Registrador:	PMC_PCER0

3. Ative o PIO para controlar o pino do botão

Registrador:	PIO_PER

4. **Desative o buffer de saída**

Registrador:	PIO_ODR

5. Ative o pull-up

Registrador:	PIO_PUER

Explique porque isso é necessário :

Se quisermos alterar comportamento do botão, devemos ativar o pullup. Quando o pullup está ativado, a leitura será Vcc se o botão não estiver apertado, e 0 quando estiver apertado.

6. Decida se irá ativar ou não o deboucing
   - não ativado : deve tratar o problema por sw

Registrador:	

   - ativado : deve configurar os registradores (mais indicado)

Registrador:	PIO_IFSCER
Registrador:	PIO_SCDR
Registrador:	

7. faça a leitura periódica no while(1) para checar se o botão foi ou não pressionado alterando, altere o status do led dependendo do botão.

Registrador:	PIO_SODR

6 - While(1)

Qual a alternativa para evitar que o status do botão seja (ou precise ser) verificado continuamente?

3 Programação parte 2 - Blink

Utilize o botão para controlar se o LED irá piscar ou não.

4 Programação parte 3 - Botões extras

Utilize a placa : OLED1XPlained pro que possui mais 3 botões para, Utilize os mesmos passos anteriores para descobrir os pinos e o PIO referente a cada botão. Implemente as seguintes funcionalidades:

1. Button 1 : Diminuir a frequência do piscar do led
2. Button 3 : Aumentar a frequência do piscar do led

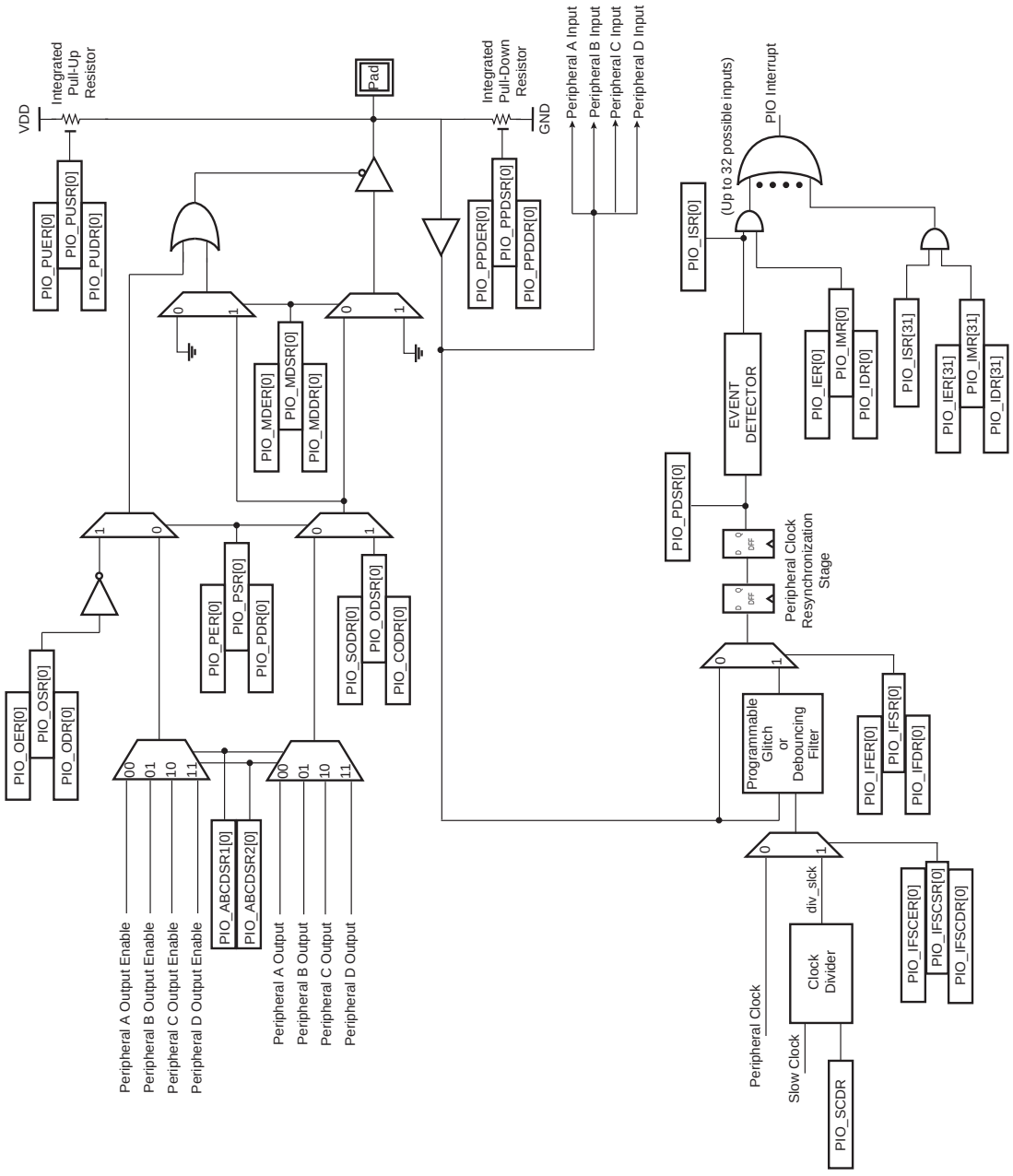
4.1 Questões do firmware em avaliação

Estaremos trabalhando nessa etapa os seguintes itens dos objetivos de

1. Faz uso correto de define a fim de melhorar o entendimento/ manipulação do firmware
2. Compreende como as informações extraídas do manual se traduzem para o código
3. Correlaciona as diversas informações contidas em diferentes documentos.
4. Faz uso de comentários
5. Sabe usar corretamente as ferramentas de gravação e depuração
6. Tem claro o fluxo de desenvolvimento

Não esqueça de enviar o código para o repositório do git.

5 Diagrama PIO


VDD
Integrated Pull-Up Resistor
Pad
Integrated Pull-Down Resistor
GND
Peripheral A Input
Peripheral B Input
Peripheral C Input
Peripheral D Input
PIO_PUER[0]
PIO_PUSR[0]
PIO_PUDR[0]
PIO_PPDER[0]
PIO_PPDSR[0]
PIO_PPDDR[0]
PIO_MDER[0]
PIO_MDSR[0]
PIO_MDDR[0]
PIO_OER[0]
PIO_OSR[0]
PIO_ODR[0]
PIO_PER[0]
PIO_PSR[0]
PIO_PDR[0]
PIO_SODR[0]
PIO_ODSR[0]
PIO_CODR[0]
PIO_ABCDSR1[0]
PIO_ABCDSR2[0]
Peripheral A Output Enable
Peripheral B Output Enable
Peripheral C Output Enable
Peripheral D Output Enable
Peripheral A Output
Peripheral B Output
Peripheral C Output
Peripheral D Output
PIO_ISR[0]
(Up to 32 possible inputs)
PIO Interrupt
EVENT DETECTOR
PIO_IER[0]
PIO_IMR[0]
PIO_IDR[0]
PIO_ISR[31]
PIO_IER[31]
PIO_IMR[31]
PIO_IDR[31]
PIO_PDSR[0]
Peripheral Clock Resynchronization Stage
Programmable Glitch or Debouncing Filter
PIO_IFER[0]
PIO_IFSR[0]
PIO_IFDR[0]
PIO_IFSCER[0]
PIO_IFSCSR[0]
PIO_IFSCDR[0]
div_slck
Clock Divider
Peripheral Clock
Slow Clock
PIO_SCDR