****

**Engenharia Eletrônica**

**EEN241 – Microcontroladores e Sistemas Embarcados**

**noturno**

**Prof. Rafael Corsi**

**Henrique Pereira Rosa 11.02741-0**

**Eduardo Galinskas Karwoski 13.01129-4**

**31/Março/2016**

*1 Periféricos*

*1.1 RTC – Real Time Clock*

O RTC é um periférico responsável por fornecer um CLOCK (oscilação) confiável, ou seja, que varie pouco com a temperatura ou outros fatores que comprometem a conduções de semicondutores. Por possuir uma relativa baixa variação no período de oscilação, pode ser aplicado em como fonte para aplicações que necessitam de precisão como relógios que ao utilizá-lo terão contagens confiáveis de tempo e levarão mais tempo até terem que ser reconfigurados ou sincronizados.

*1.2 TC – Timer/Counter*

O TC é um periférico responsável por fornecer um desvio no firmware (interrupção) quanto este atinge o valor configurado. Basicamente, há um incremento a cada ciclo de clock, porém este incremento pode ser configurado por outros registradores fazendo com que o tempo para o TC atingir o valor configurado varie conforme nossa necessidade.

*1.3 Endereços de PIOA, PIOB, ACC, UART1 e 2*

PIOA: 0x400E0E00

PIOB: 0x400E1000

ACC: 0x40040000

UART1: 0x400E0800

USART0: 0x4002400

*2 PIO*

*2.1 Multiplexação de alguns pinos uC*

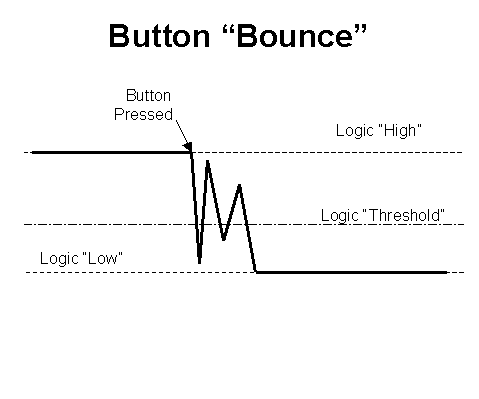
PA1: PWMH1, TIOB0, A18 e WKUP1.

PB3: UTXD1, PCK2 e AD7.

PB12: PWML1 e ERASE.

PC12: NCS3 e AD12.

PC20: A2 e PWMH2.

*2.2.0 Debouncing*

Bouncing são sinais indesejáveis nos circuitos normalmente causados pela natureza mecânica das chaves. Esses sinais nada mais são que oscilações durante a transição pressionamento e soltura do dispositivo mecânico.

*2.2.1 Algoritmo de Debouncing*

Uma técnica simples para evitar o boucing é verificar o pino de leitura do uC durante algum tempo e validar o pressionamento somente quando o sinal for o mesmo em todo o intervalo de tempo.

*2.3.0 Race Conditions*

É uma condição que ocorre quando 2 ou mais “agentes” podem usar o mesmo “bit”. Então somente o último a alterar esse bit saberá estado real dele.

*2.3.1 Como a arquitetura do Hardware evita Race Conditions*

Para efetuar a modificação é necessário manipular registadores diferentes, além do que tanto para habilitar quanto para limpar o pino é necessário escrever 1 no registrador.

*2.4 Síntese do Funcionamento do PIO*

Quando a linha de I/O é atribuída a uma função de periférico o bit correspondente no PIO\_PSR vale zero e o drive que controla a linha de I/O é controlado pelo periférico. Os periféricos A, B, C ou D dependendo do valor encontrado em PIO\_ABCDSR1 e em PIO\_ABCDSR2 determinam se o pino é ou não controlado.

Quando a linha de I/O é controlada pelo controlador do PIO, o pino pode ser configurado para ser controlado. Isto é feito escrevendo o Output Enable Register (PIO\_OER) e Output Disable Register (PIO\_ODR). O resultado dessas operações são detectadas no Output Status Register (PIO\_OSR). Quando um bit no registrador está em zero, a correspondente linha de I/O é usada como apenas um input. Quando o bit está em um, a correspondente linha de I/O é dirigida pelo controlador do PIO*.*

*O* nível do controlador em uma linha de I/O pode ser determinado pelo *writing* no Set Output Data Register (PIO\_SODR) e no Clear Output Data Register (PIO\_CODR). Essas operações de escrita, respectivamente, “seta” e limpa o Output Data Status Register (PIO\_ODSR), o qual representa THE DATA DRIVEN nas linhas de I /O. Escrevendo no PIO\_OER e no PIO\_ODR administra o PIO\_OSR se o pino é configurado para ser controlado pelo controlador do PIO ou atribuído para uma função de periférico. Isto permite a configuração das linhas dos I /O antes de configurá-lo para ser administrado pelo controlador do PIO.

Similarmente, escrevendo PIO\_SODR e PIO\_CODR afeta o PIO\_ODSR. Isto é importante como define a primeiro nível do controlador nas linhas dos I /O.