1. Periféricos

1.1 RTC (Real-Time Clock)

O periférico RTC é projetado para baixo consumo de energia. Para obter a funcionalidade ideal, o RTC requer um preciso clock externo de 32,768 kHz, que pode ser fornecido por um cristal oscilador.

Ele combina um relógio completo de hora do dia com alarme e um calendário Gregoriano ou Persa, complementado por um interrupção periódica programável. Os registros de alarme e calendário são acessados por um barramento de dados de 32 bits.

Os valores de tempo e de calendário são codificados em formato de código binário decimal (BCD). O formato de tempo pode ser de 24 horas ou o modo de 12 horas com um indicador AM / PM.

A atualização dos campos de hora e calendário e a configuração dos campos de alarme são realizadas por uma captura paralela no barramento de dados de 32-bit. Um controle de entrada é realizada para evitar registros de carregamento com dados de formato BCD incompatíveis ou com um dado incompatível de acordo com mês / ano / século atual.

Um circuito de calibração do clock divisor pode ser usado para compensar a imprecisão da frequência do cristal oscilador.

Uma saída RTC pode ser programado para gerar diversas formas de onda, incluindo um pré dimensionado clock derivado dos 32,768 kHz.

1.2 TC (Time-Counter)

Um módulo do TC inclui três canais idênticos. O número de módulos TC implementadas é específica do dispositivo.

Cada canal de TC pode ser programado de forma independente para executar uma vasta gama de funções incluindo a medição de frequência, a contagem do evento, intervalo de medição, a geração de pulsos, atraso de temporização e de modulação de largura de pulso.
Cada canal possui três entradas de clock externo, cinco entradas de clock interno e dois sinais de entrada / saída multi-propósito que pode ser configurado pelo usuário. Cada canal carrega um sinal de interrupção interno que pode ser programado para gerar interrupções do processador.

O TC incorpora um decodificador de quadratura (QDEC) ligados em frente dos temporizadores e acionado pelas entradas TIOA0, TIOB0 e TIOB1. Quando ativado, o QDEC executa as linhas de entrada de filtragem, decodificação de sinais de quadratura e liga-se aos temporizadores / contadores de forma a ler a posição e a velocidade do motor através da interface do utilizador. O bloco TC tem dois registradores globais que agem sobre todos os canais TC:

-Bloco de Controle Register (TC_BCR) - Permite canais para ser iniciado simultaneamente com a mesma instrução;

-Block Mode Register (TC_BMR) - Define as entradas de clock externo para cada canal, permitindo que eles sejam encadeados.

Endereços:

-PIOA→0x400E0E00 a 0x400E1000
-PIOB→0x400E1000 a 0x400E1200
-ACC→0x40040000 a 0x40044000
-UART1→0x400E0600 a 0x400E0740
-UART2→0x400E0800 a 0x400E0A00

2. PIO

-PA01→ PWMH1, TIOB0, A18 e WKUP1.
-PB12→ PWML1 e ERASE.
-PC12→ NCS3 e AD12.
-PB3→ UTXD1, PCK2 e AD7.
-PC20→ A2 e PWMH2.

2.1 Configurações

O debouncing representa uma técnica que fornece a certeza que quando um botão é apertado apenas uma vez seja interpretado corretamente como apenas 1 aperto de botão e se apertarmos o mesmo botão mais de uma vez seguida, o programa tem de conseguir ler corretamente o número de apertos de botão.

Isso deve acontecer devido ao fato de que quando apertamos um botão, o qual possui um contato mecânico, é gerada uma série de ruídos aleatórios, mesmo com essa duração sendo geralmente na ordem de milissegundos, o efeito prático é o de que durante esses poucos milissegundos, uma leitura efetuada nesse botão pode interpretar tanto que ele está ligado quando desligado. Porém, para um microprocessador, alguns milissegundos representam muito, por exemplo após detectar que o botão foi acionado, mediante a uma única leitura do botão, o programa pode executar as funções correspondentes e voltar ao mesmo ponto anterior da leitura do botão, assim, o efeito que temos é que em vez de o programa interpretar que apertamos o botão apenas 1 vez, ele pode acabar interpretando que apertamos o botão mais de 20 vezes em seguida.

Como exemplo podemos ter um algoritmo que utiliza a função delay, que detecta que o botão foi apertado, espera por 50 milissegundos, e verifica se o botão foi solto ou não. Caso o botão ainda esteja apertado, a rotina espera de novo 50 milissegundos, e novamente verifica se foi solto, até que finalmente o botão seja solto, sendo que apenas neste instante o programa considera que o botão foi apertado mesmo, e continua a execução.

```
if(botao = 1)
 {
delay_ms(100);
if(botão = 1)
       {
          // Botão acionado.
        }
elseif(botão = 1)
        {
          // nao esperou tempo suficiente
         }
  {
```

2.3 SET/Clear

A race condicions é uma expressão traduzida como condição de corrida e representa uma situação indesejável que ocorre quando um dispositivo tenta executar duas ou mais operações ao mesmo tempo e acaba não conseguindo saber a ordem correta que a tarefa deve ser executada. Uma condição de corrida pode ocorrer por exemplo se os comandos para ler e escrever uma grande quantidade de dados são recebidos quase no mesmo instante e as tentativas da máquinas para substituir alguns ou todos os dados antigos, enquanto os dados antigos ainda estão sendo lido pode resultar que o computador gere acidentalmente uma " operação ilegal ", notificação e encerramento do programa, erros de leitura os dados antigos ou erros de escrita os novos dados.A condição de corrida também pode ocorrer se as instruções são processadas na ordem incorreta.

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 uma vez que podemos selecionar qual operação se deseja que o sistema faça por vez, não tento o problema de o sistema querer fazer duas operações ao mesmo tempo, sem saber qual fazer primeiro.

2.4 Configurando um pino em modo de output

Quando se atribuí uma função periférica à linha de I / O, o que correspondente a setar o bit em PIO_PSR em zero, a linha de I / O é controlada pelo periférico A ou B ou C ou D e o pino passa a ser conduzido ou não dependendo do valor em PIO_ABCDSR1 PIO_ABCDSR2.

Quando a linha de I / O é controlada pelo PIO controlador, o pino pode ser configurado para ser conduzido, isso é feito por meio do Output EnableRegister (PIO_OER) e Output DisableRegister (PIO_ODR). Os resultados destas operações de gravação são detectados no estado de saída do registrador Output Status Register (PIO_OSR). Quando o bit está em zero, linha de I / O é usado como uma entrada única, já quando o bit está em um a linha de I / O é conduzida pelo PIO Controlador.

O nível conduzido sobre uma linha de I / O pode ser determinado setando a saída de dados do registrador (PIO_SODR) e dando um clear da saída de dados do registrador (PIO_CODR).

Estas operações, respectivamente, de setar e limpar a saída do Output Data Status Register (PIO_ODSR) se determinadas em PIO_OER e PIO_ODR o pino PIO_OSR é configurado para ser controlado pelo PIO controlador ou atribuído a uma função periférica. Isto permite configurar a linha I / O antes de ser controlada pelo PIO controlador. Da mesma forma, escrevendo no PIO_SODR e PIO_CODR afeta PIO_ODSR o que é importante, uma vez que se define o primeiro nível conduzido na linha de I / O.