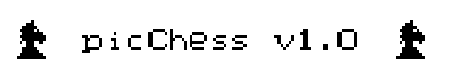
### Arthur Benemann

****

[**ENG04475 - Microprocessadores I**](http://moodleinstitucional.ufrgs.br/course/view.php?id=12413)

**Universidade Federal do Rio Grande do Sul**

**Engenharia Elétrica**

**Porto Alegre**

**26/06/2010**

Introdução

O meu objetivo para este projeto era de me familiarizar com uma familia de microntroladores da MICROCHIP. A familia em questão é a dos DSPIC33F. O projeto escolhido foi de realizar um jogo de xadres com uma interfaçe grafica e um oponente artificial. O microntrolador escolhido foi o 'DSPIC33F128MC804', que é destinado para controle de motores mas foi escolhido por causa de sua disponibilidade.

Como o objetivo era o aprendizado todo o codigo foi escrito apartir do zero, otimizado para este processador. Diversas rotinas como a de video utilizam algumas tecnicas interresantes como sera explicado adiante.

O codigo foi escrito de maneira a simplificar o seu entendimento. A separação do codigo em varios arquivos simplifica o debbugging e a reutilização de cada modulo. O codigo foi separado nos seguintes modulos:

* Audio
* Video
* Sensor de Temperatura
* Serial
* Memoria não volatil
* Comunicaão Serial
* Sensor de Temperatura
* Relogio (RTCC)
* Relogio Grafico
* Chess Engine
* Chess Interfaçe Humana

Audio

Para obter uma interface humana melhor audio foi adicionado, isso foi bastante simplificado pois o processador continha um DAC interno. Porem como a memoria interna do microprocessador era limitada (1 Mbit de Flash) uma memoria externa de 16Mbits foi adicionada ao projeto. A comunicação e gravação da memoria Flash serão explicadas adiante.

Os sons que podem ser tocados ficam gravados em sequencia ná memoria no formato 8,000 Hz 16 bit PCM, o que grante um fidelidade suficiente para tocar trechos de voz e não ocupa um espaço muito grante ( 128s de sons podem ser gravados nessa memoria). Para os tocar o DAC deve ser carregado com cada word no momento exato. Isso é feito com a ajuda de um pequeno buffer e um canal de DMA.Quando o canal é ativado o buffer é transferido word a word a cada interrupção do DAC. A interrupção do DMA é usada para re-encher o buffer de audio, com dados da memoria externa, o numero de vezes necessarias para tocar um som.

Desse modo é facil tocar um som,só deve-se carregar o endereço na memoria do som deve ser lido em um ponteiro e o numero de vezes que o buffer de audio deve ser enchido. Apartir desse pontoo mecanismo criado pelas interrupções se encadeia até o final dá musica.

O sinal gerado pelo DAC é amplificado por um circuito simples utilizando um LM380, o qual tem baixa distorção, é single supply e gera uma potencia razoavel.

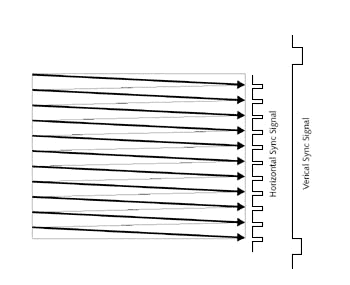
Video

O tipo de interfaçe de video escolhida foi a VGA, pois ela contem os sinais de sincronia horizontal e vertical separados dá imagem. Isso é importante para que a imagem fique corretamente enquadrada na tela e sem movimentos. Esse é um dos modulos mais pesados em termos computacionais caso fosse todo feito em software, porem tecnicas como DMA e o modulo de SPI ajudaram a reduzir tempo de processamento para menos de 10%.

Como o microntrolador tem memoria RAM limitada a resolução escolhida foi de 800x600 60Hz monocromatica. Internamente a resoluão é de 200x150 e cada pizel interno é representado como um bloco 4x4 externo, isso reduz o buffer de video a aproximadamente 4Kbytes. Dois buffers são necessarios para animações, e tambem facilita o desenho de imagens ná tela.

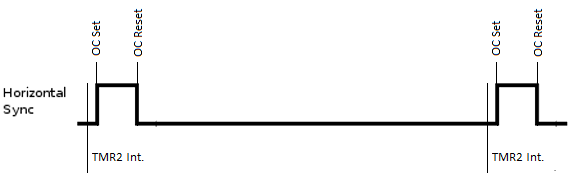
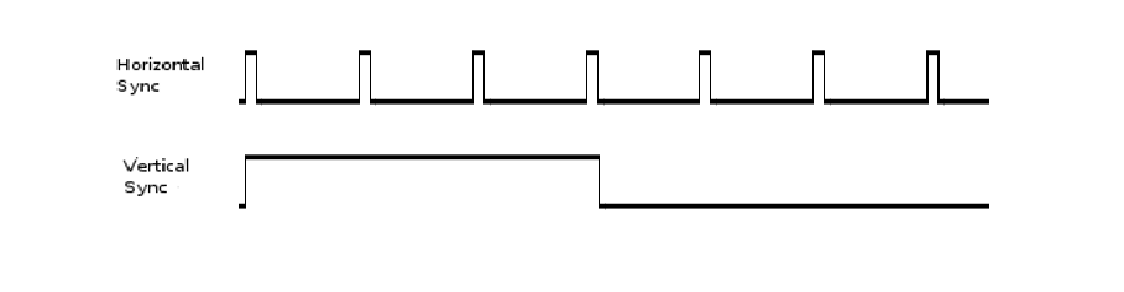
O sinal de video é composto de tres partes: o sinal de sincronia Horizontal( TTL 0v - 5v), Sincronia Vertical ( TTL 0v - 5v) e os tres sinais de cores (Red Green Blue, analogicos 0v - 0.7v). Como a imagem escolhida é monocromatia as tres linhas de cores estão ligadas juntas e só podem asumir dois niveis.

A ultima imagem recebida é salva internamente no monitor e aparece ná tela, cada uma dessas imagens é chamada de frame. A nossa missão é gerar esses frames na frequencia que o monitor os consome. As imagens são 'pintadas' ná tela conforme a figura abaixo:



Para indicar quando se está enviando um frame novo existe um sinal chamado de sincronia vertical que deve ser um pulso no começo de cada frame. Outro sinal chamado de sincronia horizontal gera pulsos antes de cada linha. Em um certo tempo entre as borda do sinal de sincronia horizontal os valores de cada pixel na tela naquela posião deve ser manadado ( 0v = preto, 0.7v = branco). A escolha de um clock de 40 MHz facilita a geração do sinal de video pois a frequencia entre cada um desses pixels é de 40MHz para uma resolução de 800x600.

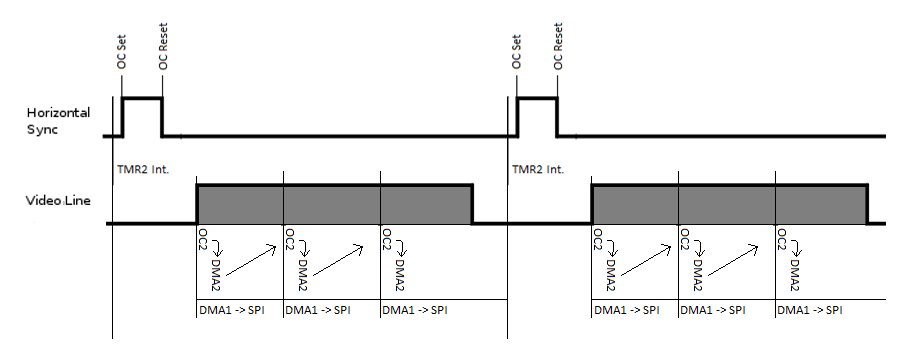
O modulo de video é controlado pelo Timer 2 o qual esta programado para estourar exatamente ná frequencia do sinal de sincronia horizontal. Para gerar esse sinal o modulo de Output Compare é usado para ligar e desligar um pino do micro automaticamente. A interupção do timer 2 é usada para decidir se esse é o inicio de um novo frame, caso sim o sinal de sincronia vertial é ativado. Como a interupão leva um certo tempo para ser executada o sinal de sincronia horizontal tambem tem esse delay adicionado a ele deixando os dois sinais sincronizados.

 O tempo no qual o sinal se sincronia vertical fica ativo é exatamente o tempo de um certo numero de pulsos de sincronia horizontal.

Tendo esses dois sinais corretamente sincronizados e na frequencia correta já temos um frame estavel no monitor.Falta agora gerar os pixels. O modulo SPI é usado para serializar uma word (16bits) bit a bit atraves de um pino do micro. Mas como a resolução interna do buffer de video é de 200 pixels por linha e não 800, o clock utilizado nessa serrializaão é de 10 MHz. Carregar o modulo SPI diretamente atreves de uma interrupção não seria viavel pis isso usaria quase todo o tempo do processador para fazer este carregamento. Um canal de DMA é utilizado então para transferir um pequeno buffer de 'linha' a cada linha a ser serializada ( leia-se 'pulso de sincronia horizontal'). O buffer de linha deve ser carregado apartir do buffer de video antes de cada linha ná rotina de interrupção do TMR2, que occore antes de cada linha.E o DMA deve ser ativado no momento certo do pulso de sincronia horizontal.

Ao utilizar a propria interupão do modulo SPI para requisitar novos dados via DMA acorre um problema, o modulo SPI fica trancado por um clock o que gera um pixel largo a cada 16 pixels enquanto o DMA esta transferindo o dado. Um modo utilizado para resolver isto e o problema de ativar o DMA no momento certo é usar um modulo de Output Compare ligado ao timer dois apenas pela sua interrupção. Essa interupão gera um request para o DMA que transfere uma word por vez para o modulo SPI.

Isso resolve o problema da primeira transferencia, para transmitir as proximas words nessa linha ( uma linha = 200 pixels = 12.5 words ) um segundo canal de DMA é usado para comandar o momento de interrupção do OC que gera a requisição para o DMA do SPI. Esse DMA utiliza um buffer estatico para transferir os momentos de interrupção exatos para o primeiro DMA cada vez que uma interrupção é gerada.

 Desse para transmitir uma linha de video (200 pixels + sinal de sincronia horizontal) o processador só tem que carregar o buffer de linha antes de cada linha. O buffer de linha poderia ser completamente eliminado caso o DMA deste micro. cobrisse toda a memoria RAM ( ele está limitado a apenas 2K) lendo diretamente do buffer de video.