Este é um rascunho do primeiro artigo de uma série de três, publicado na revista Eletrônica Total, em parceria com Edmur Canzian, da CNZ. Para citar:

> VALIANTE FILHO, F. Desenvolvimento de Projetos Utilizando Microcontroladores. Revista Eletrônica Total, São Paulo, v. 108, p. 8 - 11, 01 jun. 2005.

DESENVOLVIMENTO DE PROJETOS UTILIZANDO MICROCONTROLADORES

Quantas vezes não perdemos tempo desenvolvendo um projeto, ou parte dele, e depois vemos que poderíamos ter feito aquilo de um jeito muito mais rápido ou mais fácil? Ou pior, quantas vezes nos lembramos daquele detalhe tão essencial quando o projeto já era dado quase que por encerrado? Ou descobrimos tarde demais que sub-dimensionamos o projeto? Faltou pino, espaço pro programa, etc.

Para evitar esses e outros problemas, é necessário adotar uma metodologia adequada para o desenvolvimento de projetos utilizando microcontroladores. Isto é, obedecer alguns passos básicos e fundamentais para otimizar o projeto em todos os sentidos: recursos utilizados, tempo de execução, confiabilidade, desempenho do hardware e do software, etc.

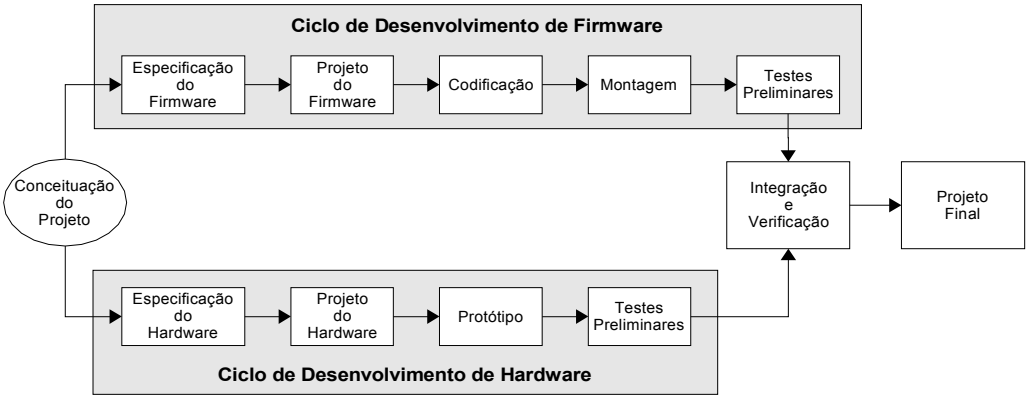
O primeiro passo é, obviamente, definir de maneira clara o problema a ser resolvido ou a idéia a ser implementada. O que é, o que faz e como faz. Com isto em mente partimos para o desenvolvimento. Os projetos utilizando microcontroladores dividem-se basicamente em hardware e software (firmware).

- **Hardware:** é o circuito eletrônico. Além do microcontrolador, abrange todas as entradas, saídas, alimentações e demais circuitos auxiliares.

- **Software:** é a programação (código) feita para que o projeto funcione. Diferentemente dos programas para computadores como o PC, o software aqui é "embarcado", isto é, será armazenado em uma memória dentro do próprio chip microcontrolador ou presente no mesmo circuito; e executado no próprio microcontrolador. Um termo mais adequados e utilizado para esse tipo de software é FIRMWARE.

Os desenvolvimentos do hardware e do firmware podem ocorrer em paralelo e até mesmo serem mantidos a cargo de pessoas ou equipes diferentes, desde que o projeto seja bem planejado e documentado. Para projetos menores é bem comum que um único projetista assuma a responsabilidade por ambas as "frentes". De qualquer forma, o projeto só estará concluído com hardware e firmware prontos, integrados e testados em conjunto.

A fig. 1 ilustra todos os passos do ciclo de desenvolvimento de projetos com microcontroladores (ou sistemas microprocessados):



Vamos nos ater primeiramente ao ciclo de desenvolvimento de hardware.

Na etapa de especificação devemos definir:

- Alimentação do circuito;
- Entradas e saídas – quantas e com quais características: isolação óptica, maior ou menor potência, adequação de nível elétrico, casamento de impedância, etc.;
- Outros circuitos auxiliares e de proteção;
- Microcontrolador a ser utilizado.

Neste último item, – a escolha do microcontrolador – , há alguns critérios a serem levados em consideração. Para um projeto feito como "hobby", fatalmente escolheremos um que já conheçamos, tenhamos facilidade de comprar, montar e programar. Porém, para um projeto mais "sério", é necessário avaliar também:

- Arquitetura de microcontrolador mais adequada ao projeto;
- Desempenho e velocidade;
- Recursos "embutidos": gerador de PWM, conversores A/D, acesso a memória externa, timer, etc.;
- Disponibilidade no mercado e preço;
- Ferramentas de programação disponíveis: custo, desempenho, recursos;
- Outras ferramentas de desenvolvimento como emuladores e kits.

Nos próximos artigos veremos um mesmo projeto sendo implementado com a utilização das três principais famílias de microcontroladores de 8 bits disponíveis no mercados hoje: Intel 8051 e compatíveis, PIC da Microchip e HC08 da Motorola. Ao final, será possível ter uma excelente idéia sobre qual é a melhor tecnologia a ser adotada.

A próxima etapa, após a especificação do hardware e com o microcontrolador já definido, é o projeto do circuito. É nesta etapa que aquelas entradas e saídas ganham "corpo". Uma entrada opto-acoplada recebe o CI opto-acoplador adequado. Uma saída de potência recebe seu transistor, tiristor ou mesmo relé. Uma entrada que precisa de

casamento de impedância pode ganhar um amplificador operacional com os resistores devidamente calculados, idem aos divisores de tensão necessários, etc. O resultado desta etapa é um esquema elétrico que pode ser feito em qualquer software CAD para circuitos eletrônicos.

Na confecção do protótipo há basicamente três caminhos a serem seguidos:

- Montagem de uma PCI (placa de circuito impresso) exclusivamente para este fim;
- Montagem do circuito em placa universal ou proto-board;
- Montagem do circuito em um kit de desenvolvimento.

A montagem em PCI é uma alternativa mais cara, principalmente à medida que se fizer necessária uma alteração de hardware. A montagem em um kit de desenvolvimento é a mais rápida e, em geral, mais flexível, especialmente quando utilizando um kit onde todas as conexões estão disponíveis para o usuário, sem pré-definições do fabricante. É o caso da **Plataforma Modular CNZ** que utilizaremos em nosso exemplo.

Feito o protótipo, é possível efetuar testes preliminares ainda que sem o firmware necessário para seu funcionamento. Deve-se testar as alimentações, sinais de clock e o correto funcionamento das entradas e saídas, valendo-se de intrumentos como multímetro, osciloscópio e gerador de funções. Pode ser necessário um firmware de teste, de poucas linhas, apenas para acionar entradas e saídas de modo a possibilitar essa verificação.

Hardware pronto, se isso fosse uma receita de bolo poderíamos dizer "reserve a massa que agora vamos preparar o recheio", isto é, o firmware.

A especificação do firmware segue a mesma idéia da do hardware, porém desta vez temos que definir:

- Os "problemas" a serem resolvidos;
- O algoritmo, isto é, o método a ser utilizado para a solução do problema;
- As variáveis a serem utilizadas.

O projeto do software pode ser encarado como a elaboração do programa usando uma linguagem "humana", seja ela gráfica na forma de fluxogramas e diagramas de estados, seja textual na forma de pseudo-códigos. Esta etapa é crucial para uma correta e precisa implementação do programa em código de máquina, ou linguagem de alto nível.

Com o projeto pronto passamos à sua tradução para a linguagem do microcontrolador, isto é, o processo de codificação. Para execução desta etapa é necessário apenas um editor de textos que possa salvar o **arquivo fonte** no formato texto puro (sem formatação).

Os microcontroladores só "entendem" linguagem de máquina. Para facilitar o trabalho dos projetistas é adotada uma série de mnemônicos que tornam os códigos de máquina mais acessíveis. Assim, para mover uma informação podemos usar o mnemônico "MOV" e para efetuar uma adição, o mnemônico "ADD". Esta linguagem de mnemônicos é chamada **assembly**. Cada fabricante define o assembly utilizado em seu microcontrolador, embora eles guardem grande semelhança entre si. Os arquivos assembly têm formato texto e possuem a extensão "asm".

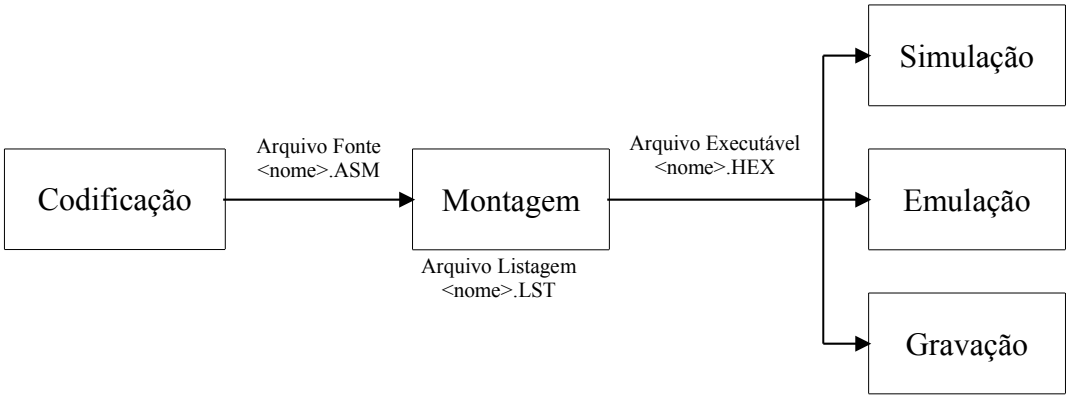
Essa linguagem de máquina mais "inteligível", que é o assembly, passa por um processo de montagem para gerar um código de máquina puro em formato hexadecimal. Isto é feito por um programa montador, ou **assembler**. Como resultado do processo de montagem temos também um arquivo de listagem, de extensão "lst", onde é possível ver o código em assembly ao lado do código em hexadecimal já montado, bem como o endereço de cada instrução. É também nesse arquivo que podemos verificar eventuais erros ocorridos na montagem. Caso não haja nenhum erro o arquivo hexadecimal é gerado e leva a extensão "hex".

Esse arquivo montado ("hex"), pode ser utilizado para:

- Simulação em software;
- Emulação em hardware;
- Gravação do microcontrolador em um programador (ou pelo fabricante);
- Gravação em kit de desenvolvimento.

Também é possível programar microcontroladores empregando linguagens de alto nível como a C. Para isto são necessários compiladores e bibliotecas adequados para cada microcontrolador, nem sempre gratuitos. A programação em alto nível facilita a execução de operações complexas, especialmente as matemáticas, assim como a migração entre diferentes microcontroladores. Como desvantagens temos um programa final maior (após compilação e montagem) e, em geral, a não obtenção do melhor desempenho possível na execução do programa.

A fig. 2 resume o processo de desenvolvimento do firmware da codificação à montagem:



Os testes preliminares no desenvolvimento do firmware podem ser feitos através de simulação no microcomputador. Esse recurso é bastante útil pela possibilidade de execução passo-a-passo, cálculos precisos do tempo de execução e garantia de não queimar nenhum componente caso haja algo errado...

Agora é possível integrar hardware e firmware efetuando a gravação do microcontrolador e a verificação do funcionamento. Aqui voltamos ao ainda não citado, mas velho conhecido, ciclo de projeto: desenvolver, testar, corrigir... Até obter hardware e firmware sem "bugs", funcionando corretamente, perfeitamente integrados, estáveis e de acordo com o especificado.

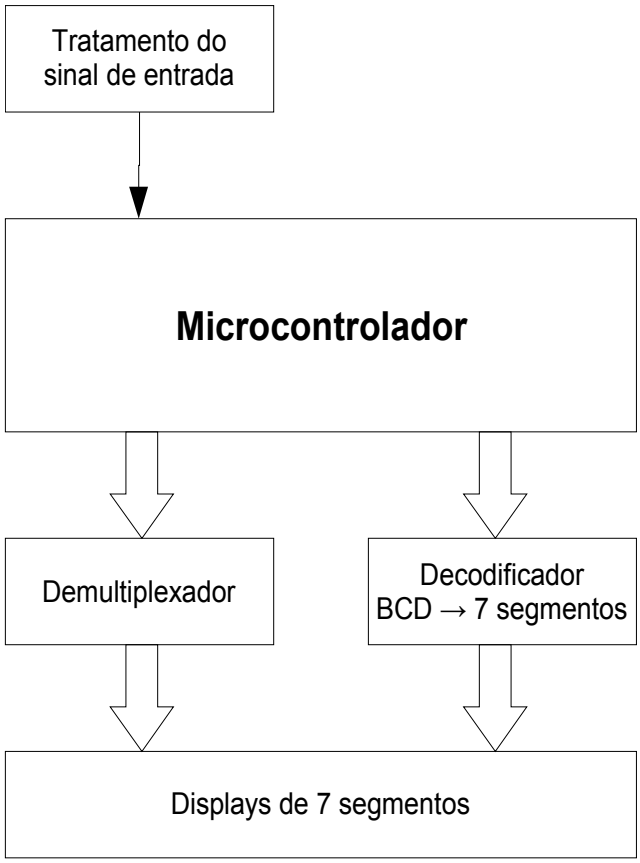
Este nosso protótipo final pode, enfim, ganhar uma PCI definitiva, instalação em caixa adequada, etc.

Para exemplificar a metodologia, vamos iniciar o projeto de um **freqüencímetro digital**, cujas implementações finais para as famílias 8051, PIC e HC08 serão vistas nos próximos artigos.

Especificação do projeto "Freqüencímetro Digital":

- Instrumento capaz de medir a freqüência de sinais digitais periódicos de nível elétrico compatível com a entrada do microcontrolador. A faixa de freqüência a ser medida deve partir de 0,1 Hz até a máxima possível em cada família de microcontrolador. O instrumento deve ser capaz de apresentar décimos de hertz em sua leitura, que deve ser feita através de um display.

Seguindo o ciclo de desenvolvimento de hardware, o próximo passo é a especificação do hardware. A única interface com o usuário é o display para a exibição do resultado, o qual optaremos por um bloco de 8 displays de 7 segmentos operando de forma multiplexada. Isto implica na presença de um circuito de acionamento adequado para o display, além do circuito de multiplexação. O freqüencímetro precisa de uma entrada de sinal com as adequações de nível elétrico e proteções necessárias. Há também o coração do projeto que é o microcontrolador. Na figura abaixo podemos ver o diagrama de blocos do circuito do nosso projeto. Em tempo, apesar de não exibir a alimentação, sabemos que é necessário uma fonte com a tensão correta e estável, que poderia até ser externa.



Os requisitos do microcontrolador neste projeto são um timer interno de 16 bits e uma interrupção externa sensível à borda; além de possuir memória e pinos de E/S suficientes.

O uso de multiplexação para o acionamento do display diminui significativamente o consumo de energia e poderíamos dizer que é uma necessidade. Já o emprego de CIs dicretos para o demultiplexador e o decodificador de endereços possibilitam a economia de pinos de E/S do microcontrolador, bem como uma simplificação do firmware, resultando na possibilidade de se empregar um modelo com menos pinos de E/S e memória e, por conseqüência, mais barato. Para uma produção em pequena quantidade, mesmo considerando-se uma placa maior e os dois CIs adicionais, a economia compensa. Porém, para uma produção maior, ou um projeto mais "delicado", isso não costuma ser verdade. Ou seja, há um compromisso entre o resultado necessário e o investimento para obtê-lo tanto no aspecto técnico como no financeiro. Cabe ao projetista pesar os prós e contras de cada opção.

A partir deste ponto, o projeto do firmware passa à elaboração do circuito, o que será visto nos próximos artigos, respeitando-se as particularidades de cada implementação.

Passemos agora ao ciclo de projeto do firmware.

O problema a ser resolvido é a medição da freqüencia do sinal de entrada. Em linhas gerais, a solução deste problema, ou seja, o algoritmo a ser utilizado, constitui-se em

medir o período do sinal através de um contador acionado através de uma base de tempo conhecida. Medido o período é feita a conversão para freqüência. O display deve ser atualizado periodicamente. Esse ciclo é contínuo.

Vamos admitir que a base de tempo para o temporizador do microcontrolador seja de 1μs. Ele será programado para gerar uma interrupção a cada 1ms, que será utilizada para atualizar o display e completar a contagem do período. A detecção do sinal será feita através da interrupção externa, gerada pela presença de uma borda de subida, de forma a atualizar a letura a cada período.

As principais variáveis do firmware serão:

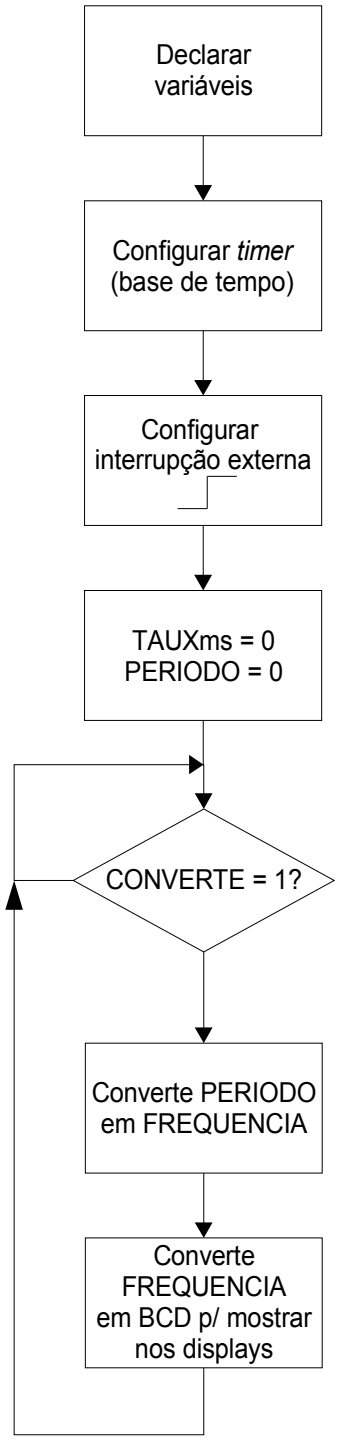
- TAUXms – variável auxiliar para a medida do período (n * 1ms);
- TAUXus – variável auxiliar para a medida do período (n * 1μs);
- PERIODO – variável que armazena o período formada pela soma "coerente" das variáveis auxiliares;
- FREQ – variável da freqüência, calculada a partir de PERIODO;
- DIG[1:8] – variáveis dos dígitos em BCD, obtidas a partir de FREQ.

Para que possamos desenvolver o código corretamente, é necessário colocar esse princípio de funcionamento no papel, de maneira organizada. Isto pode ser feito através de um fluxograma. O fluxograma é a estrutura do programa que será implementado; se o firmware "não funcionar" no fluxograma, significa que não vai funcionar na prática. Quanto mais detalhado o fluxograma, mais fácil será o processo de codificação. Devem ser representado, ao menos, os blocos principais.

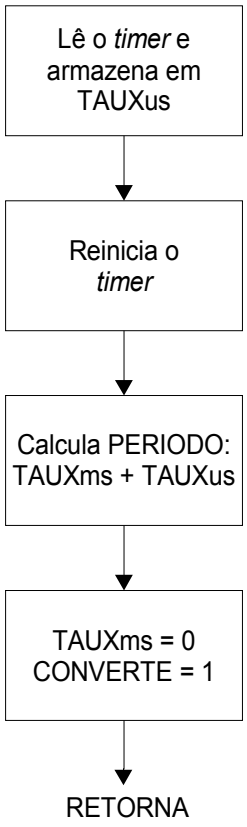
A figura seguinte mostra o fluxograma do nosso freqüencímetro. Tanto do programa principal como das interrupções externa e de temporizador.

A próxima etapa do ciclo de desenvolvimento do firmware é a codificação, bastante diferente em cada microcontrolador. Isto será visto nos próximos artigos, juntamente com os circuitos, integração e testes finais do projeto com cada família de microcontrolador: 8051, PIC (14 bits) e Motorola (HC08).

PROGRAMA PRINCIPAL



ROTINA DE INTERRUPÇÃO EXTERNA



ROTINA DE INTERRUPÇÃO DE TEMPO