COMUNICAÇÃO SPI ENTRE MICROCONTROLADORES ARM CORTEX-M4 TIVA C SERIES TM4C123GH6PM

Kaio Vítor Gonçalves de Freitas[1]; Jandilson Almeida Bandeira[1]; Melisse Pontes Cabral[2] ; José Lucas Damasceno Holanda[1]; Tamires dos Santos Pereira[3]

1 Universidade Federal de Campina Grande, Graduando em Engenharia Elétrica, kaio.freitas@ee.ufcg.edu.br; jandilson.bandeira@ee.ufcg.edu.br; lucasholanda95@gmail.com
1 Universidade Federal de Campina Grande, Graduando em Ciências da Computação, melisse.cabral@ccc.ufcg.edu.br
2 Escola Técnica Redentorista, Eixo Tecnológico de Controle e Processos Industriais, tsantosp16@gmail.com

Introdução

Progressivamente, a tecnologia se dissemina com facilidade em qualquer meio que é implantada e a ambição das pessoas que buscam seu entendimento e deixam suas obras representam a máxima expressão da dignidade humana, por fazer da vida uma luta conveniente a de um ser eterno em busca de um futuro melhor. Um exemplo prático de algo que abrange muitas áreas são os microcontroladores, que são usados em eletrodomésticos, dispositivos médicos implantados, sistema de controle de automóveis, sistemas de supervisões, entre outros dispositivos eletrônicos digitais que nos cercam.

De acordo com Fagner de Araújo Pereira (2014)

> A situação que nos encontramos hoje no campo de microcontroladores teve seu início a partir do desenvolvimento da tecnologia de circuitos integrados. Esse desenvolvimento permitiu que pudéssemos instalar centenas de milhares de transistores em um único chip, que foi a precondição para a possibilidade de fabricação de microprocessadores. Os primeiros computadores foram então desenvolvidos a partir dos microprocessadores, adicionando periféricos externos, tais como memórias, linhas de entrada e saída, temporizadores e outros circuitos. Aumentando ainda mais a densidade de elementos dentro dos chips resultou no desenvolvimento de circuitos que continham ambos, processador e periféricos. Isto mostra como o primeiro chip contendo um microcomputador chamado mais tarde de microcontrolador foi desenvolvido.

Os microcontroladores TM4C123GH6PM utilizam o núcleo Cortex-M4 fabricado pela Texas Instrument, sendo da série dos mais novos microcontroladores a possuírem esse núcleo que, conceitualmente, diferem do núcleo anterior (Cortex-M3) apenas por ter instruções DSP (*Digital Signal Processor*). A escolha desse microcontrolador foi dado pelo seu baixo custo, em torno de $13,00 no site de venda da Texas, pela conveniência de o ter e saber usá-lo, porém, os ensinamentos que serão dados poderão ser aplicados a qualquer microcontrolador, adaptando-os a sua plataforma. Dos periféricos disponíveis nele, iremos utilizar o GPIO (*General Purpose In/Out*), conhecidos como pinos de entrada e saída, SPI (*Serial Peripheral Interface*). A UART (*Universal Asynchronous Receiver/Transmitter*) é usada apenas para escrever na tela do computador os resultados obtidos na comunicação usando *software* Putty, para verificar a efetividade do projeto. O compilador usado é o Code Composer Studio v6 (CCS), que usa uma linguagem derivada de C.

A SPI é um protocolo que permite a comunicação serial síncrona do microcontrolador com outros dispositivos de maneira *full-duplex*, ou seja, a transmissão é simultânea e em ambos os sentidos, que são do Mestre para o Escravo e do Escravo para o Mestre, definidos

nesse tipo de comunicação. O mestre (*master*) é o gerador do sinal de sincronismo para que a transmissão só ocorra a partir de sua ordem, e o escravo (*slave*) faz o uso dele, sendo necessário haver apenas um mestre para qualquer quantidade de escravos. No nosso caso, a maior vantagem desse tipo de comunicação é o aumento de periféricos disponíveis para o usuário, já que tudo será duplicado pelo uso de duas placas idênticas.

Metodologia

Para a transferência de dados entre os microcontroladores é necessário, primeiramente, fazer a conexão dos fios de um para o outro. Foi usada a mesma configuração de pinos de entrada/saída para os dois: o pino digital A2 para o *clock*, o pino A3 como SS (*Slave Select*), o pino A4 como MISO (*Master Input Slave Ouput*) e o pino A5 como MOSI (*Master Output Slave Input*). É ligado o pino A2 dos dois para que compartilhem o mesmo *clock* assim como os pinos A3, para que haja um sincronismo de *frame*, e o pino do terra, explícito na placa, para que tenham a mesma referência. O pino A4 de um é ligado no A5 do outro (e vice versa), estando pronta toda a parte externa do projeto, restando agora o uso do *software*, cujos comandos de cada operação da biblioteca estão expostos em parênteses.

Todos os procedimentos são semelhantes para o *master* e o *slave*. Deve-se definir o *clock* do sistema no mestre (SysCtlClockSet), lembrando que a frequência de operação do núcleo é de 80MHz, não podendo ultrapassar esse valor. Logo, habilitou-se os periféricos SSI0 e a porta A do microcontrolador (SysCtlPeripheralEnable), liberando todos os pinos dessa porta para uso, e os pinos foram configurados conforme o primeiro parágrafo da metodologia. As últimas configurações do protocolo SPI a ser feita (SSIConfigSetExpClk) é o tipo de *frame*, selecionado como Freescale SPI, o modo, que deve ser selecionado *MASTER* no mestre e *SLAVE* no escravo, o *Baud Rate*, que é a taxa de transferência de dados e precisa ser o mesmo nos dois microcontroladores para que haja sincronismo, e o tamanho dos dados, escolhidos de 8 bits. Havendo-se configurado todos os parâmetros do SPI, chegou a hora de ativá-lo para iniciar a transmissão (SSIEnable).

Para evitar que a comunicação comece enquanto ainda há dados salvos nos *buffers* de recepção, é usado um comando *while* com uma função que retorna 1 enquanto a recepção ainda estiver sendo realizada (SSIDataGetNonBlocking), ficando preso no *loop* até que essa condição seja quebrada, usando, em seguida, um comando para limpar qualquer lixo restante (SSIIntClear). Então, os dados que serão enviados precisam ser salvos em variáveis e enviados ao *buffers* de transmissão (SSIDataPut), espera-se o final dessa operação com um *while* e uma condição dentro (SSIBusy) para então salvar em outra variável (SSIDataGet) o valor vindo do receptor, finalizando a transmissão.

Resultados e discussão

Notou-se bastante eficácia na comunicação entre os dois microcontroladores ao usar a UART para escrever os valores das variáveis no Putty em cada etapa (transmissão, recepção), sendo necessário, quando compilado, sempre dar *reset* primeiramente no escravo, para que os dados a ser transmitidos por ele sejam colocados no *buffer* de transmissão antes do mestre dar início à transferência, ao receber o *reset* em seguida. Caso contrário, o mestre envia seus dados mas não receber nada.

Apesar disso, um fator relevante é a viabilidade do uso da biblioteca existente para os comandos desse periférico, uma vez que, sem ela, teríamos que alterar o valor de cada

registrador de 32 bits associado a essa comunicação, com a ajuda do *datasheet* para diferenciar os bits que devem ser mudados dos bits reservados, para que ela fosse concluída,

aumentando o esforço para chegar em um objetivo comum. No entanto, o estudo sem encapsulamento resulta em um melhor entendimento sobre cada etapa da comunicação, logo, deve ser estudado com atenção assim que o estudante entender bem todo o protocolo e quiser ter mais versatilidade para suas aplicações, o que não é possível quando se tem funções muito encapsuladas.

Conclusões

Conforme as expectativas concretizadas nos procedimentos, na importância e na flexibilidade dos microcontroladores na atualidade, executamos o planejamento de expandir horizontes de modo a definir conceitos sólidos, escolher o objetivo a ser alcançado e explicar, passo a passo, como realizá-lo, tornando o estudo mais completo e seguro.

Vimos, também, a importância de buscar as características disponíveis no seu microcontrolador e separar as que atendem suas necessidades para que, junto com um estudo do objeto a ser controlado por ele, todas suas necessidades possam ser supridas.

Palavras-Chave: Comunicação; Serial; Microcontrolador; Duplex; Mestre-Escravo.

Referências

FLOYD, THOMAS L. **Sistemas Digitais**. Editora Bookman. 9ª edição. 2007.
FRENZEL JR. L. E. **Eletrônica Moderna**. Editora Bookman. 1º Edição, 2015.
MANO, M. M. **Digital Design.** Editora Pearson/Prentice-Hall. 5ª edição, 2013.
PEREIRA, F. A**. Microcontroladores PIC e linguagem C** (Apostila). 2014, 85p.
SOUZA, S. A. **Introdução aos microcontroladores ARM Cortex-M4 Tiva C Series da Texas Instrument** (Apostila). Universidade Federal de Juiz de Fora. 2015, 61p.