Resumo de protocolos de comunicação

1)Visão geral sobre protocolos

Existem dois formatos que os protocolos podem seguir, serial e paralelo.

.**Serial**: envia e recebe toda a informação sequencialmente, o que permite que o número de fios seja menor.

.**Paralela**: é capaz de enviar e receber dados em mais de uma via de comunicação, ou seja, é capaz de enviar vários bits simultaneamente, resultando em maior rapidez na transmissão da informação.

1.1)Características:

.Taxa de comunicação: representa a velocidade da comunicação, cuja unidade é bits por segundo (bps). Em comunicações síncronas é chamada de *clock* e em assíncronas é nomeada como *Baud Rate*.

.Métodos:

-Síncrono: utiliza um sinal de *clock*, ou seja, para o envio de bits, uma vantagem é sua velocidade de transmissão de dados e uma desvantagem é a necessidade de um fio extra para o *clock*.

-Assíncrono: não utiliza o *clock,* porém necessita que os dispositivos que se comunicam possuam a mesma *Baude Rate* ou taxa de comunicação.

.Sentido de Transmissão:

-**Full-duplex:** dispositivo pode enviar e receber dados ao mesmo tempo.

-Half-duplex: não pode receber e enviar dados simultaneamente.

-Simplex: dispositivo que pode apenas receber ou enviar dados.

.Tensão do Protocolo: tensão que os protocolos identificam os níveis lógicos alto e baixo.

.Terminologias:

-RX/TX: RX é o termo usado para representar o pino receptor de uma comunicação serial e TX representa o transmissor.

-Mestre e escravo: método de comando centralizado onde apenas o dispositivo mestre pode iniciar uma comunicação.

2)Protocolos

.UART: UART é um acrônimo para *Universal Asynchronous Receiver / Transmitter,* é um protocolo assíncrono e *full-duplex.*

-Funcionamento: o pino de transmissão (Tx) do protocolo envia um pacote de bits que será interpretado bit a bit pelo pino receptor (Rx). Cada pacote enviado contém 1 start bit que indica o início da mensagem, 1 ou 2 stop bits para indicar o final da mensagem, 5 a 9 bits de informação e 1 bit de paridade para evitar a recepção de erros.

-Ligação: é realizada por dois pinos Rx/Tx que dependem do *Baud Rate* como referência.

-Aplicações: receptores GPS, Arduino, *Bluetooth Modules*, GSM and GPRS *Modems*, *Wireless Communication Systems*, dentre outros.

.I2C: é *half-duplex*, pois contém apenas um pino para envio de dados, e síncrono, pois usa um pino de clock.

-Funcionamento: o dispositivo mestre deve informar aos dispositivos escravos o início da comunicação, ou "Start condition", para tal, o pino SCL deve estar em nível lógico alto e o pino SDA em nível lógico baixo. Quando isso ocorrer, todos os escravos estarão prontos para receber a primeira informação que é o endereço do escravo que comunicará com o mestre, junto com a operação que este escravo desempenhará. Em situações em que houver mais de um mestre na comunicação, terá preferência o mestre que sinalizar mais rápido o início de uma transmissão. Depois que o endereço é enviado, o escravo que tiver o endereço correspondente realizará a operação de leitura ou escrita da informação até que o dispositivo mestre envie uma "stop condition" para interromper a comunicação.

-Ligação: este protocolo utiliza apenas dois pinos, SDA que é o sinal de dados e SCL o clock.

-Aplicações: comunicação entre circuitos integrados, sejam eles microcontroladores, sensores (acelerômetros, GPS, etc…).

.SPI: comunicação *full-duplex* e síncrona.

-Funcionamento: primeiramente o mestre gera um clock e seleciona através do pino SS com qual dispositivo será efetuada a comunicação. Em

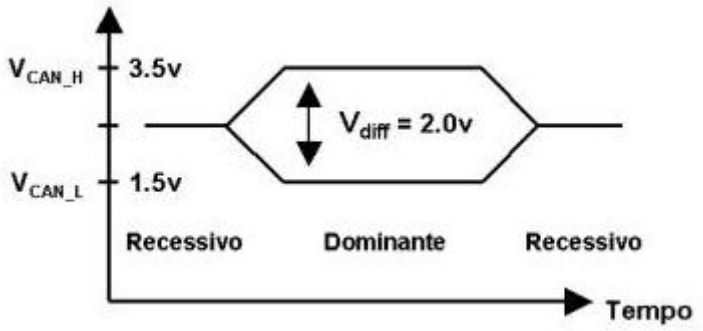
seguida os dados são enviados para o dispositivo de destino pelo pino MOSI e então o dispositivo escravo envia uma resposta (se necessário) ao mestre pelo pino MISO.

-Ligação: existem dois tipos de ligação para o protocolo SPI, ligação paralela e ligação em cascata. Na ligação paralela, o protocolo utiliza 1 pino de MOSI, MISO e clock para todos dispositivos e 1 pino SS para cada escravo ligado à comunicação. Na ligacão em cascata, é utilizado apenas 1 pino SS para todos os dispositivos ligados na comunicação, porém causa perda de velocidade na comunicação.

-Aplicações: utilizado em LCD e LED, memórias flash e EEPROM, para a comunicação entre diferentes periféricos como conversor analógico digital (ADC) e conversor digital analógico (DAC).

.CAN: CAN é um acrônimo para *Controller Area Network* e também é um protocolo *half-duplex* e síncrono.

-Funcionamento: é baseado no conceito multi-mestre, no qual todos os módulos podem se tornar mestre em determinado momento e escravo em outro e suas mensagens são enviadas em regime multicast, caracterizado pelo envio de toda e qualquer mensagem para todos os módulos existentes na rede. Além disso o CAN também exercita o conceito CSMA/CD with NDA (*Carrier Sense Multiple Access / Collision Detection with Non-Destructive Arbitration*), isto é: todos os módulos verificam o estado do barramento, analisando se outro módulo está ou não enviando mensagens com maior prioridade, para que o módulo de maior prioridade continue enviando sua mensagem deste ponto, sem ter que reiniciá-la. No CAN, os dados não são representados por bits em nível “0” ou nível “1”, ao invés disso, são representados por bits Dominantes e bits Recessivos, criados em função da condição presente nos fios CAN_H e CAN_L.



-Ligação: considerando-se fios elétricos como o meio de transmissão dos dados, existem três formas de se constituir um barramento CAN, dependentes diretamente da quantidade de fios utilizada. Existem redes baseadas em 1, 2 e 4 fios. As redes com 2 e 4 fios trabalham com os sinais de dados CAN_H

(CAN *High*) e CAN_L (CAN *Low*). No caso dos barramentos com 4 fios, além dos sinais de dados, um fio com o VCC (alimentação) e outro com o GND (referência) fazem parte do barramento, levando a alimentação às duas terminações ativas da rede. As redes com apenas 1 fio têm este, o fio de dados, chamado exclusivamente de linha CAN.

-Aplicações: metrôs, ferrovias, máquinas raio X, além de diversas aplicações em veículos, tendo em vista que o CAN foi inicialmente criado com esse propósito.

Referências

Uma visão geral sobre os protocolos UART, I2C e SPI:
https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html

1. UART
   https://paginas.fe.up.pt/~hsm/docencia/comp/uart/
   https://www.newtoncbraga.com.br/index.php/telecom-artigos/1709-
   vídeo explicativo: https://www.youtube.com/watch?v=3PfZ65lnCHw

2. I2C
   http://www.univasf.edu.br/~romulo.camara/novo/wp-content/uploads/2013/11/Barramento-e-Protocolo-I2C.pdf
   http://mundoprojetado.com.br/i2c/
   vídeo explicativo: https://www.youtube.com/watch?v=3yRQcGxIilQ

3. SPI
   https://www.embarcados.com.br/spi-parte-1/
   http://mundoprojetado.com.br/spi/
   vídeo explicativo: https://www.youtube.com/watch?v=zjY-oGmltgs

4. CAN
   https://www.youtube.com/watch?v=ky4eVDTICUQ
   https://web.fe.up.pt/~ee99058/projecto/pdf/Can.pdf
   vídeo mais ou menos explicativo: https://www.youtube.com/watch?v=2ApyndAOI0s
   texto explicativo: http://www.alexag.com.br/CAN_Bus_Parte_2.html