

ELF52 - Sistemas Microcontrolados

Interfaces Seriais: SPI e I2C

Professor:

Prof. Marcos Eduardo

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Interface SPI (SSI)

Interface SPI

- *Serial Peripheral Interface Bus:*
 - É um barramento padronizado de comunicação serial síncrona criado pela Motorola.
- Opera em *full-duplex*;
- Os dispositivos funcionam como *master/slave*:
 - Grande evolução tecnológica;
 - Altas velocidades de comunicação;
 - Padronização de protocolos de HW e SW;
 - Outras tecnologias utilizadas: SPI, I2C, 1-wire.
- O SPI utiliza 4 fios para conexão.

Vantagens

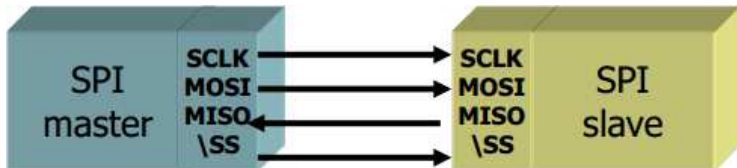
- Comunicação *full-duplex*;
- Alta velocidade (1 a 70MHz);
- Flexibilidade para a transferência de bits (qualquer tamanho de mensagens);
- Interfaceamento muito simples;
- O *master* gera o *clock*;
- Utiliza menos pinos de interface que barramentos paralelos.

Desvantagens

- Endereçamento feito por pino específico;
- Distâncias muito pequenas quando comparado com o RS232;
- Escravos não fazem *acknowledge* da mensagem.

Sinais Lógicos

- SCLK: *Serial Clock* (gerado pelo *master*);
- MOSI/SIMO: *Master Output, Slave Input* (gerado pelo *master*);
- MISO/SOMI: *Master Input, Slave Output* (gerado pelo *slave*);
- \SS : *Slave Select* (**Ativo em baixo**, gerado pelo *master*).



Sinais Lógicos

- Também são utilizados os seguintes nomes:
 - SCK (**SSIClk**): *Serial Clock* (gerado pelo master);
 - SDI, DI, SI (**SSIRx**): *Serial Data In*;
 - SDO, DO, SO (**SSITx**): *Serial Data Out*;
 - \nCS, \CS, \nSS, \STE (**SSIFss**): *Chip Select, Slave Transmit Enable*;
 - Observar que SDI/SDO (DI/DO, SI/SO) requer que o SDO do *master* seja conectado ao SDI do *slave* e vice-versa.

Transmissão de Dados

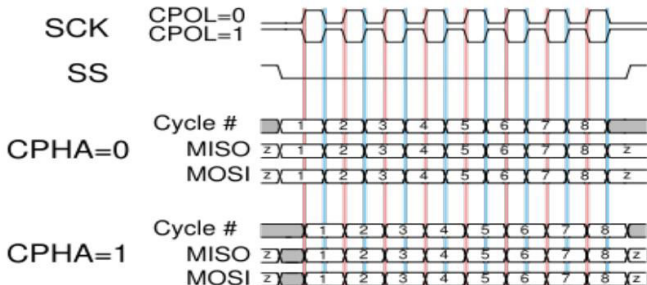
- O *master* inicia a comunicação selecionando o *slave*;
- A cada pulso de *clock* do *master*, um bit é enviado pelo MOSI e um bit é recebido pelo MISO;
- O MSB sempre vai primeiro;
- Qualquer quantidade de bits pode ser enviada/recebida;
- Dois parâmetros definem as bordas do sinal de *clock* onde os dados serão amostrados: *clock polarity* (CPOL) e *clock phase* (CPHA), dando 4 possibilidades, correspondendo aos modos de funcionamento da SPI;
- Os 4 modos são **incompatíveis** entre si.

Transmissão de Dados

- CPOL=0: *Clock* em **0** → *Idle*:
 - CPHA=0, dados são lidos na borda de **subida** e mudam na borda de **descida**;
 - CPHA=1, o inverso.
- CPOL=1: *Clock* em **1** → *Idle*:
 - CPHA=0, dados são lidos na borda de **descida** e mudam na borda de **subida**;
 - CPHA=1, o inverso.

Transmissão de Dados

- CPOL=0: *Clock em 0 → Idle:*
 - CPHA=0, dados são lidos na borda de **subida** e mudam na borda de **descida**;
 - CPHA=1, o inverso.
- CPOL=1: *Clock em 1 → Idle:*
 - CPHA=0, dados são lidos na borda de **descida** e mudam na borda de **subida**;
 - CPHA=1, o inverso.



Aplicações

- Leitor de Cartão SD
- *Display Oled SPI*



Interface SPI no Tiva

Características

- Possui 4 interfaces SPI (SSI)
- A tabela 17-1 mostra os pinos GPIO onde estão mapeados cada uma das interfaces SPI. (Cada SPI deve utilizar no mínimo 4 pinos):
 - SSInClk;
 - SSInFss;
 - SSInXDAT0 (TX);
 - SSInXDAT1 (RX).

Registradores

- Para ativar o SSI, o *clock* do respectivo SPI tem que ser ativado no registrador **RCGCSSI** (*SSI Run Mode Clock Gating Control*);
- Verificar o bit do SSI respectiva no registrador **PRSSI** (*SSI Peripheral Ready*) para saber se está pronto para o uso.

Registradores

- O registrador **SSICRO** (*Control*) (pag. 1245) controla várias funções:
 - **SCR**: Valor entre **0** e **255**. Controla o *Baud Rate* da SPI juntamente com o CPSDVSR obedecendo a equação:
$$BR = SysClk / [CPSDVSR * (1 + SCR)]$$
 - **SPH**: CPHA. **0** para dados capturados na primeira borda do *clock*.
1 para a segunda borda do *clock*;
 - **SPO**: CPOL (*polarity*);
 - **FRF**: Formato do quadro, possibilitando escolher o formato específico da Texas. Escolher o modo 0, tradicional;
 - **DSS**: Tamanho do dado a ser transmitido
0x3: 4 bits, ..., **0xF**: 16bits.

Registradores

- O registrador **SSICR1** controla outras funções:
 - **MODE**: Modo de operação para utilização de outros modos como **BI** e **QUAD SSI**. Usar 00 para modo clássico SPI;
 - **MS**: Modo *master* ou *slave*. **0** para *master*, **1** para *slave*;
 - **SSE**: Habilita o SSI.

Registradores

- O registrador **SSISR** fornece o estado de alguns flags, como o estado das FIFOs de transmissão e recepção e se o SSI está ocupado;
- O registrador **SSICPSR** (*prescaler*), contém o campo **CPSDVSR** que controla junto com o **SCR** o *baud-rate*. Qualquer número par entre 2 e 254:
$$BR = SysClk / [CPSDVSR * (1 + SCR)]$$
- O registrador **SSIDR** controla os dados a serem transmitidos ou recebidos da FIFO.

Passos Básicos (GPIO)

- Para configurar:

- 1 Habilite o *clock* no módulo GPIO no registrador **RCGGPIO** (cada bit representa uma GPIO) e espere até que a respectiva GPIO esteja pronta para ser acessada no registrador **PRGPIO** (cada bit representa uma GPIO) para os 4 pinos do SSI;
- 2 Desabilite a funcionalidade analógica dos pinos do GPIO no registrador **GPIOAMSEL**;

Passos Básicos (GPIO)

- 3 Preencha a função alternativa dos pinos do GPIO, no registrador **GPIOCTL** (verifique a tabela 10-2 no *datasheet* páginas 743-746);
- 4 Habilite os bits de função alternativa no registrador **GPIOAFSEL** para o pino do GPIO;
- 5 Habilite a função digital no pino do GPIO no registrador **PIODEN**;

Passos Básicos (GPIO)

- 6 Habilite o *clock* no módulo SSI no registrador **RCGCSSI** (cada bit representa uma SSI) e espere até que a respectiva SSI esteja pronta para ser acessada no registrador **PRSSI** (cada bit representa uma SSI);
- 7 Desabilite o SSI no registrador **SSICR1** escrevendo 0 no bit **SSE**;
- 8 Configure o SSI como *master* no registrador **SSICR1** escrevendo 0 no bit **MS**;

Passos Básicos (GPIO)

- 9 Configure o registrador de *prescaler* **SSICPSR** como divisor de *clock*, e o campo SCR no registrador **SSICR0** segundo a equação de *baud rate*;
- 10 Configure o registrador **SSICR0**, os bits **SPH**, **SPO** para 0 (se o SPI estiver sendo executado no modo normal) e o campo **FRF** para 0;
- 11 Configure o tamanho do número de bits na FIFO no campo **DSS** do registrador **SSICR0**;
- 12 Habilite o SSI no registrador **SSICR1** escrevendo 1 no bit **SSE**.

Passos Básicos (Transmissão/Recepção)

- Para transmitir e receber (sem interrupção):
 - 1 Pode-se fazer uma função que transmita e uma que receba ou ambas juntas;
 - 2 Para transmitir, espere até que bit de FIFO não cheia (**TNF**), no registrador **SSISR** esteja setado. Quando estiver em 1, copie o dado a ser transmitido para o registrador **SSIDR**;
 - 3 Para receber, espere até que o bit de FIFO não vazia (**RNE**), no registrador **SSISR** esteja setado. Quando estiver em 1, copie o dado do registrador **SSIDR** para uma variável.

Interface I2C

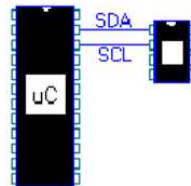
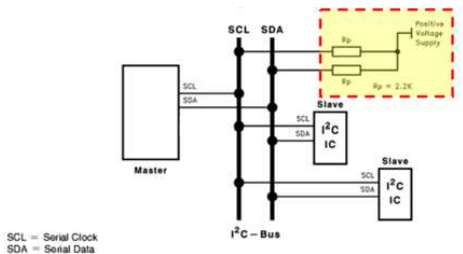
Aplicações

- I2C (Inter-Integrated Circuit) é um barramento de comunicação que foi desenvolvido pela Philips em 1980;
- Velocidades de operação:
 - *Low-speed mode*: DC-10 kbps;
 - *Standard*: 100 kbps;
 - *Fast mode*: 400 kbps;
 - *Fast mode plus*: 1 Mbps;
 - *High-speed mode*: 3,4 Mbps.
- O I2C utiliza apenas dois fios (saídas com coletor aberto): *half-duplex*;
- Dados úteis sempre de 8 em 8 bits (1 byte).



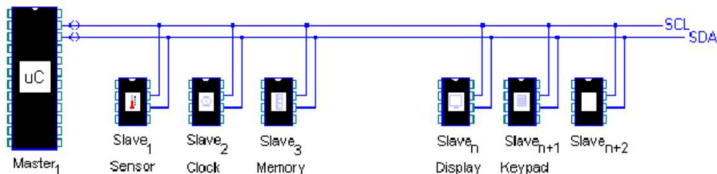
Comunicação

- Pinos de comunicação:
 - *Serial Data (SDA)* e *Serial Clock (SCL)*;
 - **Ambos devem ter pull-up (2K2)**
- Há dois tipos de nós: *Master* e *Slave*:

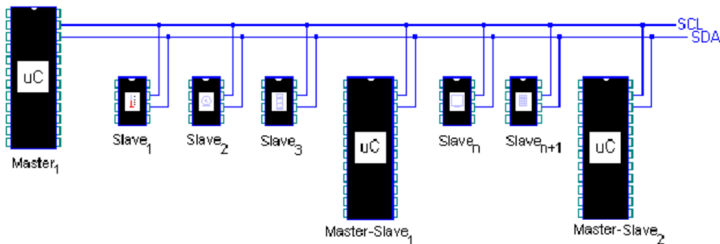


Topologias de Conexão

- *Master-slave:*



- *Multi-master Multi-slave:*



Protocolo de Comunicação

- O *master* começa a comunicação enviando:
- Um *start bit* (transição descendente de SDA com SCL=1);
- O endereço de 7-bits do *slave* (MSB primeiro):
 - O oitavo bit corresponde ao modo de escrita ou leitura;
 - Se for **1** corresponde a uma **leitura**, se for **0** corresponde a uma **escrita**.



 from master to slave


 from slave to master

Protocolo de Comunicação

- O *slave* do respectivo endereço com um bit ACK (*acknowledge*);
 - Para **escrita**:
 - O *master* envia bytes de dados (escreve no *slave*) intercalados por bits ACK do *slave*.
 - Para **leitura**:
 - O *master* recebe bytes de dados (lê do *slave*) intercalando bits ACK do *master*, exceto após o último byte recebido.




 from master to slave

 from slave to master

Protocolo de Comunicação

- Para finalizar a comunicação, o *master* pode enviar:
 - Um *stop* bit (transição ascendente de SDA com SCL=1);
 - Um novo *start* bit (para recomeçar outra transferência).

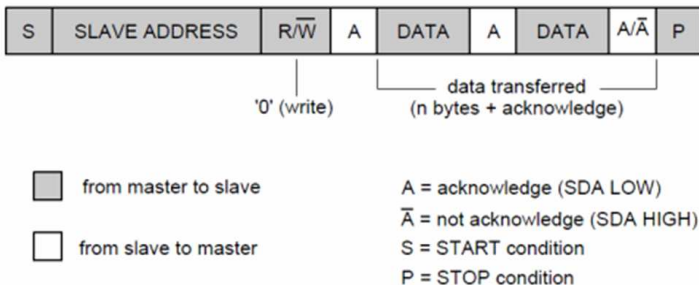


 from master to slave

 from slave to master

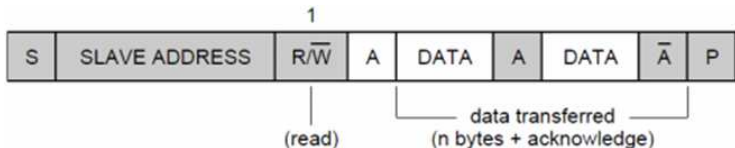
Operação de Escrita

- Se o *master* precisa apenas escrever no *slave*:



Operação de Leitura

- Se o *master* precisa apenas ler do *slave*:



 from master to slave

 from slave to master

A = acknowledge (SDA LOW)

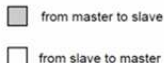
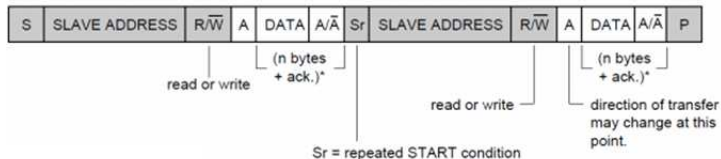
\bar{A} = not acknowledge (SDA HIGH)

S = START condition

P = STOP condition

Operação Mista

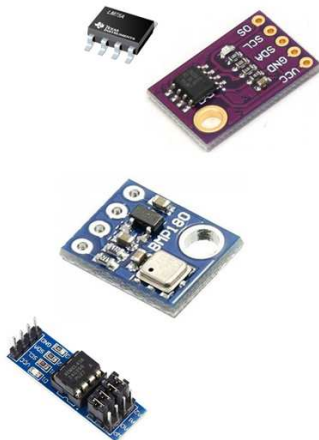
- Às vezes o *master* precisa escrever alguns dados e depois ler do *slave*;
- Por exemplo, enviar o endereço do escravo com bit de *write* e depois enviar um tipo de dados adicional como o endereço de um registrador. É necessário um bit de *repeated start condition*:



A = acknowledge (SDA LOW)
Ā = not acknowledge (SDA HIGH)
S = START condition
P = STOP condition

Aplicações

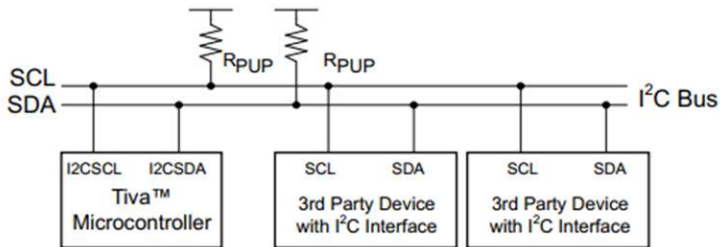
- Sensor de temperatura LM75A;
- Sensor digital de pressão BMP180;
- EEPROM serial.



Interface I2C no Tiva

Características

- Possui 10 interfaces I2C;
- A tabela 18-1 mostra os pinos GPIO onde estão mapeados cada uma das interfaces I2C. (Cada I2C deve utilizar 2 pinos);
 - I2CnSCL;
 - I2CnSDA.

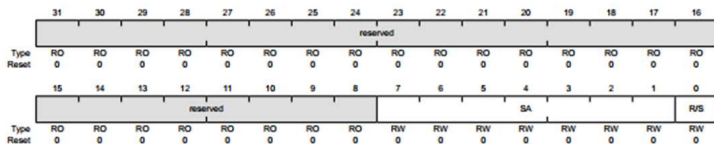


Registradores

- Para ativar o I2C, o *clock* do respectivo I2C tem que ser ativado no registrador **RCGI2C** (*Run Mode Clock Gating Control*);
- Verificar o bit do I2C respectiva no registrador **PRI2C** (*Peripheral Ready*) para saber se está pronto para o uso.

Registadores

- O registrador **I2CMSA** (*Master Slave Address*) controla o endereço do *slave* e se é operação de leitura ou escrita.



- R/S**: Especifica se a próxima operação de *master* será de recepção (1) ou escrita (0);
- SA**: Especifica os 7 bits do escravo.

Registadores

- O registrador **I2CMCS** (*Control/Status*) tem duas possibilidades. Ele acessa os bits de status quando lido e os bits de controle quando escrito;
- Quando lido:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	ACTDMARD		ACTDMATX		reserved											
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CLKTO	BUSBSY	IDLE	ARELST	DATAACK	ADRACK	ERROR	BUSY
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

- **CLKTO**: Erro de *timeout* de *clock*;
- **BUSBSY**: Barramento I2C está ocupado (usado para múltiplos masters). Muda com o START ou STOP bits;
- **IDLE**: Controlador I2C está livre;
- **DATAACK**: Os dados transmitidos receberam ACK;
- **ADRACK**: O endereço transmitido recebeu ACK;
- **ERROR**: Foi detectado um erro na última operação;
- **BUSY**: Controlador I2C está ocupado.

Registadores

- O registrador **I2CMCS** tem duas possibilidades. Ele acessa os bits de status quando lido e os bits de controle quando escrito;
- Quando escrito:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved									BURST	OCMD	HS	ACK	STOP	START	RUN
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

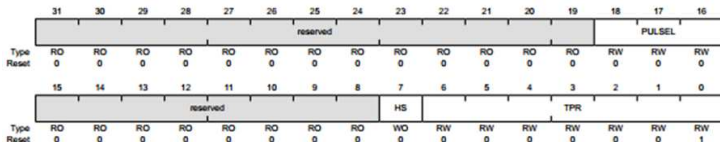
- **BURST**: Modo *burst* (várias escritas ou várias leituras, de acordo com o tamanho (*bytes*) configurado em **I2CMBLEN**);
- **HS**: Habilita modo *high-speed* até 3.34 Mbps;
- **ACK**: Se setado, o *master* envia ACK automaticamente na recepção;
- **STOP**: O controlador gera a condição de STOP quando setado;
- **START**: O controlador gera um START ou *repeated* START;
- **RUN**: O *master* é habilitado.

Registradores

- O registrador **I2CMDBR** (*Master Data*) contém os dados a serem transmitidos se estiver no **modo de transmissão do master** e os dados recebidos se estiver no **modo de recepção do master**;
- Se o bit **BURST** estiver ativo, o registrador **I2CFIFODATA** armazena os respectivos dados.

Registadores

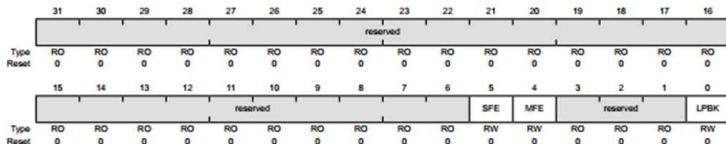
- O registrador **I2CMTPR** controla o período do *clock* do SCL:



- TPR**: Campo a ser preenchido, segundo a seguinte equação:
SCL_PERIOD = 2*(1+TPR)*10*CLK_PRD
 - Para 80MHz $\rightarrow CLK_PRD = 12,5$ ns
 - SCL_Freq : com $HS = 0$, temos as opções *Standard mode*: 100kbps, *Fast-mode*: 400kbps, *Fast-mode plus*: 1Mbps
 - $SCL_Period = 1/SCL_Freq$

Registadores

- O registrador **I2CMCR** configura o modo de operação ou *master* ou *slave*;



- SFE:** 0 para modo escravo desabilitado e 1 para modo escravo habilitado;
- MFE:** 0 para modo *master* desabilitado e 1 para modo *master* habilitado.

Registradores

- Se utilizar interrupções há outros registradores como: **I2CMIMR**, **I2CMRIS**, **I2CMICR**.

Passos Básicos (GPIO)

- Para configurar:
 - 1 Habilite o *clock* no módulo GPIO no registrador **RCGGPIO** (cada bit representa uma GPIO) e esperar até que a respectiva GPIO esteja pronta para ser acessada no registrador **PRGPIO** (cada bit representa uma GPIO) para os 2 pinos do I2C. (Por exemplo para a I2C0 são os pinos **PB2** e **PB3**);
 - 2 Desabilite a funcionalidade analógica dos pinos do GPIO no registrador **GPIOAMSEL**;

Passos Básicos (GPIO)

- 3 Preenche a função alternativa dos pinos do GPIO, para o SCL e SDA, no registrador **GPIOCTL** (verifique a tabela 10-2 no datasheet páginas 743-746);
- 4 Habilite os bits de função alternativa no registrador **GPIOAFSEL** para o pino do GPIO;
- 5 Habilite a função digital no pino do GPIO no registrador **GPIODEN**;
- 6 Sete o pino que será I2CSDA para dreno aberto no registrador **GPIOODR**;

Passos Básicos (I2C)

- 7 Habilitar o *clock* no módulo I2C no registrador **RCGI2C** (cada bit representa uma I2C) e esperar até que a respectiva I2C esteja pronta para ser acessada no registrador **PRI2C** (cada bit representa uma I2C);
- 8 Habilitar a função de *master* no registrador **I2CMCR** escrevendo 1 no bit **MFE**;
- 9 Configurar o *clock* no campo **TPR** registrador **I2CMTPR**. Por exemplo, para 100kbps:
$$2*(TPR+1)*10*12,5ns = 10us \rightarrow TPR = 39$$

Passos Básicos

- Para transmitir e receber (sem interrupção):
 - Devem ser seguidas as máquinas de estados de transmissão, recepção ou *repeated start* segundo as figuras 18-8, 18-9, 18-10, 18-11, 18-12 e 18-13 a partir da página 1290 do *datasheet*.

Comparação

<i>Comm.</i>	<i>Description</i>	<i>I/O</i>	<i>Max. Speed</i>	<i>Max. Distance</i>	<i>Max. Possible Number of Devices in a Bus</i>
UART	Asynchronous serial point-to-point communication	2	115.2kbps	15m	2 (Point-to-Point)
I2C	Short-range synchronous master-slave serial communication using one data and one clock line	2	3.4 Mbps	0.5m	127*
SPI	Short-range synchronous master-slave serial communication	3/4	70 Mbps	0.1m	Virtually unlimited

*Depende do número de bits do endereço.

Dúvidas?