Synchronous Serial Interface

TM4C123GH6PM Launchpad

Dr. Munesh Singh

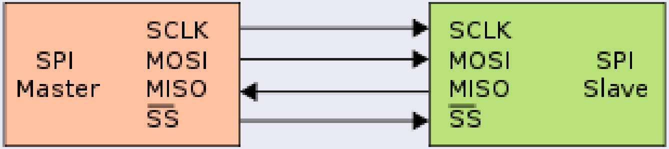
Indian Institute of Information Technology Design and Manufacturing
Kancheepuram, Chennai, Tamil Nadu

March 8, 2020

Synchronous Serial Interface

SSI

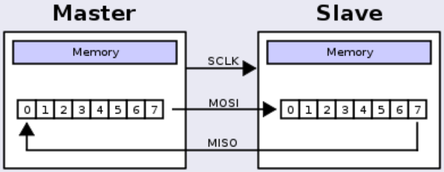
1. The Serial Peripheral Interface (SPI) is a synchronous serial communication interface
2. The interface was developed by Motorola in the mid 1980s
3. SPI devices communicate in full duplex mode using a master-slave architecture with a single master.
4. Multiple slave devices are supported through selection with individual slave select (SS) lines.



Synchronous Serial Interface

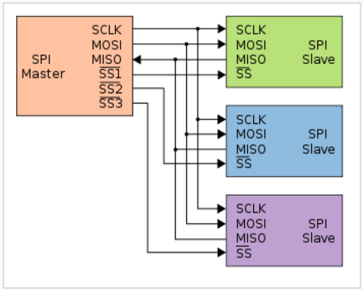
Data Transmission

1. The Bus master configures the clock using a frequency supported by the slave device
2. The master then selects the slave device with logic 0 on select line
3. Transmissions normally involve two shift registers of some given word size (8 bit)
4. These shift register in master and slave devices are virtual connected in ring topology

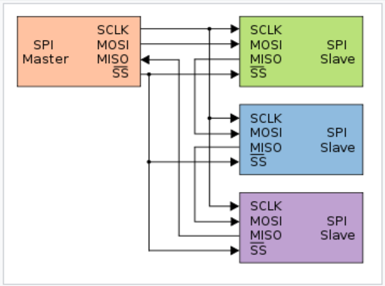


Synchronous Serial Interface

Connecting with Multiple SPI device



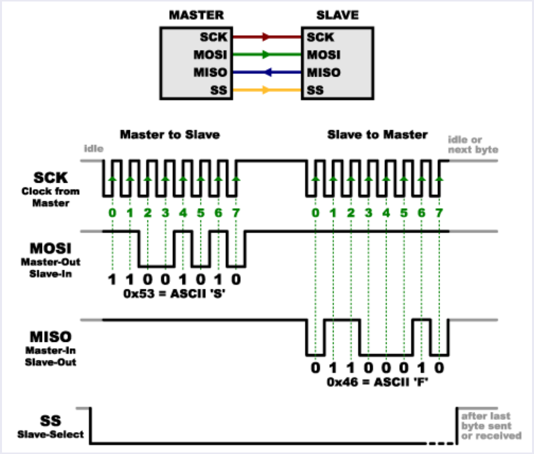
Typical SPI bus: master and three independent slaves



Daisy-chained SPI bus: master and cooperative slaves

Synchronous Serial Interface

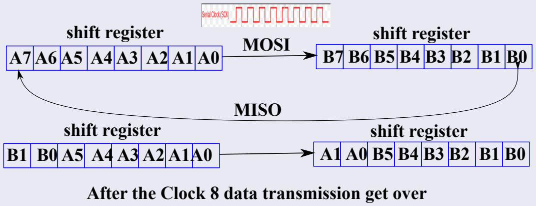
Operation



Synchronous Serial Interface

Configuration

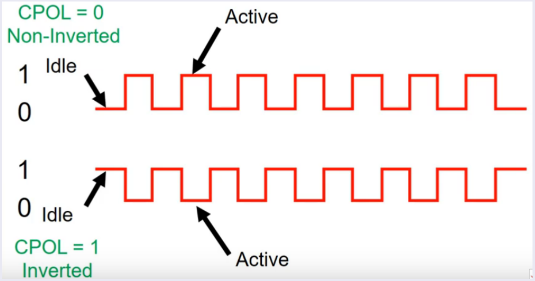
- SPI communication uses simple shift register



Synchronous Serial Interface

Configuration

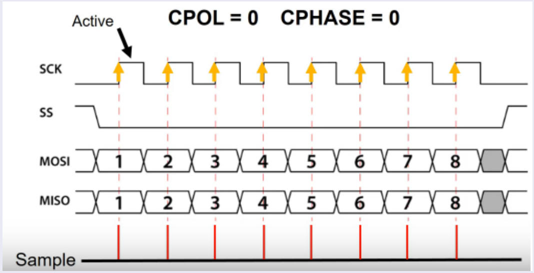
- SPI Mode Configuration required to set two register
  - **Clock Polarity (CPOL)**
  - **Clock Phase(CPHASE)**
    - When the data has to be toggled
    - When the data has to be sampled



Synchronous Serial Interface

Configuration

- **CPHASE=0** Data will be sampled at the leading edge of the clock
- **CPHASE=1** Data will be sampled at the trailing edge of the clock



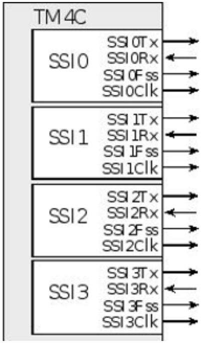
Synchronous Serial Interface

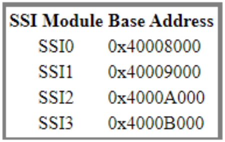
Modes

Mode	**CPOL**	**CPHASE**	**Comments**
Mode 0	0	0	Active state of of clock is 1 Data sampling at leading edge
Mode 1	0	1	Active state of of clock is 1 Data sampling at trailing edge
Mode 2	1	0	Active state of of clock is 0 Data sampling at leading edge
Mode 3	1	1	Active state of of clock is 0 Data sampling at trailing edge

SPI programming in TI ARM Tiva

- The TM4C123GH6PM microcontroller system includes four Synchronous Serial Interface modules (SSI0, SSI1, SSI2 and SSI3)
- Each SSI module provides a master or a slave interface for synchronous serial communication (SSI)
- The SSI modules are located at the following base addresses:





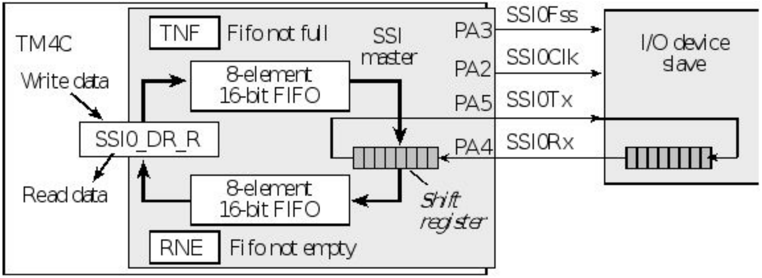
SSI Module	Base Address
SSI0	0x40008000
SSI1	0x40009000
SSI2	0x4000A000
SSI3	0x4000B000

The SSI protocol includes four I/O lines

- The slave select SSI0Fss is an optional negative logic control signal from master to slave signal signifying the channel is active.
- The second line, SSI0Clk, is a 50% duty cycle clock generated by the master.
- The SSI0Tx (master out slave in, MOSI) is a data line driven by the master and received by the slave.
- The SSI0Rx (master in slave out, MISO) is a data line driven by the slave and received by the master.

SSI

- On the TM4C the shift register can be configured from 4 to 16 bits.
- The shift register in the master and the shift register in the slave are linked to form a distributed register.
- The following figure illustrates communication between master and slave.



SSI FIFO

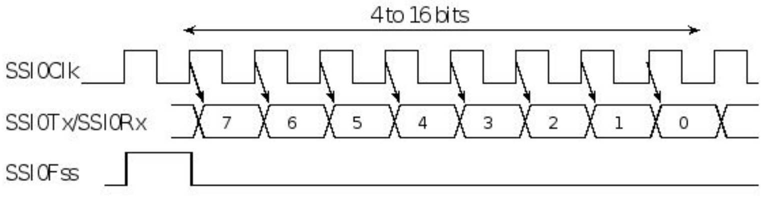
- The SSI on the TM4C employs two hardware FIFOs.
- Both FIFOs are 8 elements deep and 4 to 16 bits wide, depending on the selected data width.
- When performing I/O the software puts into the transmit FIFO by writing to the SSI0_DR_R register
- Similarly, gets from the receive FIFO by reading from the SSI0_DR_R register.
- If there is data in the transmit FIFO, the SSI module will transmit it.
- With SSI it transmits and receives bits at the same time.

SSI Communication Protocol

- There are three mode control bits (MS, SPO, SPH) that affect the transmission protocol.
- If the device is a master (MS=0) it generates the SCLK, and data is output on the SSI0Tx pin, and input on the SSI0Rx pin.
- The SPO control bit specifies the polarity of the SCLK
- The SPO bit specifies the logic level of the clock when data is not being transferred.
- The SPH bit affects the timing of the first bit transferred and received.

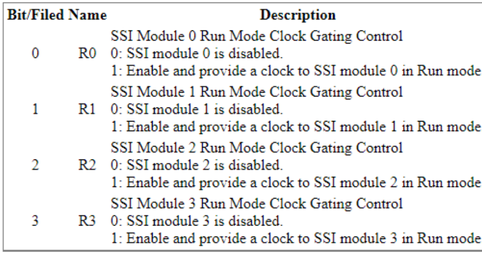
SSI Communication Protocol

- SPH is 0, then the device will shift data in on the first (and 3rd, 5th, 7th, etc.) clock edge.
- SPH is 1, then the device will shift data in on the first (and 4th, 6th, 8th, etc.) clock edge.
- The data is transmitted MSB first.



Enabling Clock to SSI

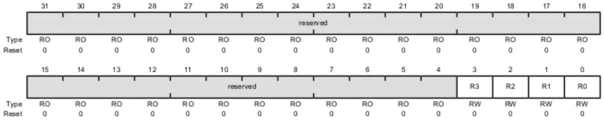
- To enable and use any of the peripheral modules in the Tiva chip, we must enable the clock to it.
- We use RCGCSSI register to enable the clock to SSI modules.
- We need RCGCSSI = 0x0F to enable the clock to all four SSI modules.



Bit/Filed	Name	Description
0	R0	SSI Module 0 Run Mode Clock Gating Control 0: SSI module 0 is disabled. 1: Enable and provide a clock to SSI module 0 in Run mode
1	R1	SSI Module 1 Run Mode Clock Gating Control 0: SSI module 1 is disabled. 1: Enable and provide a clock to SSI module 1 in Run mode
2	R2	SSI Module 2 Run Mode Clock Gating Control 0: SSI module 2 is disabled. 1: Enable and provide a clock to SSI module 2 in Run mode
3	R3	SSI Module 3 Run Mode Clock Gating Control 0: SSI module 3 is disabled. 1: Enable and provide a clock to SSI module 3 in Run mode

Synchronous Serial Interface Run Mode Clock Gating Control (RCGCSSI)

Base 0x400F.E000
Offset 0x61C
Type RW, reset 0x0000.0000

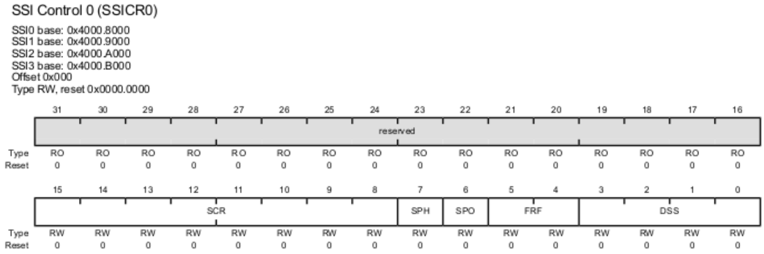


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Configuring the SSI Module

- The SSICR0 (SSI Control register 0) sets SSI configuration.
- Although the conventional SPI uses only 8-bit data
- The SSI modules allow transfer of data between 4 bits to 16 bits.



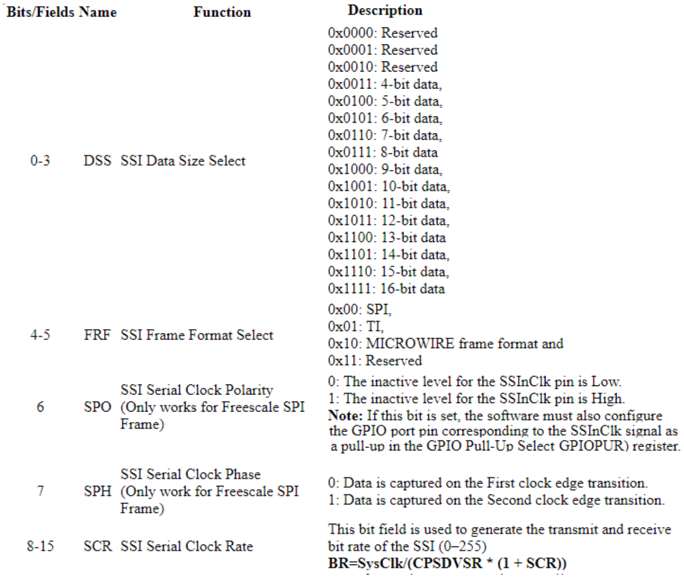
SSI Control 0 (SSICR0)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x000
Type RW, reset 0x0000.0000

	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
	SCR								SPH	SPO	FRF		DSS			
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

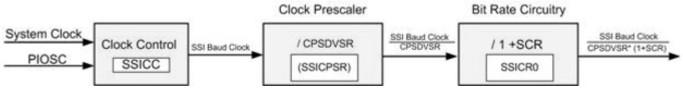
Configuring the SSI Module



Bits/Fields	Name	Function	Description
0-3	DSS	SSI Data Size Select	0x0000: Reserved 0x0001: Reserved 0x0010: Reserved 0x0011: 4-bit data, 0x0100: 5-bit data, 0x0101: 6-bit data, 0x0110: 7-bit data, 0x0111: 8-bit data 0x1000: 9-bit data, 0x1001: 10-bit data, 0x1010: 11-bit data, 0x1011: 12-bit data, 0x1100: 13-bit data 0x1101: 14-bit data, 0x1110: 15-bit data, 0x1111: 16-bit data
4-5	FRF	SSI Frame Format Select	0x00: SPI, 0x01: TI, 0x10: MICROWIRE frame format and 0x11: Reserved
6	SPO	SSI Serial Clock Polarity (Only works for Freescale SPI Frame)	0: The inactive level for the SSInClk pin is Low. 1: The inactive level for the SSInClk pin is High. **Note:** If this bit is set, the software must also configure the GPIO port pin corresponding to the SSInClk signal as a pull-up in the GPIO Pull-Up Select GPIOPUR) register.
7	SPH	SSI Serial Clock Phase (Only work for Freescale SPI Frame)	0: Data is captured on the First clock edge transition. 1: Data is captured on the Second clock edge transition.
8-15	SCR	SSI Serial Clock Rate	This bit field is used to generate the transmit and receive bit rate of the SSI (0–255) **BR=SysClk/(CPSDVSR * (1 + SCR))**

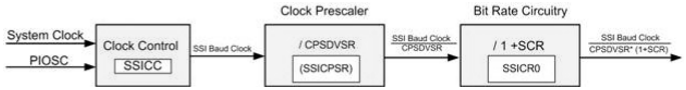
Setting a Bit Rate

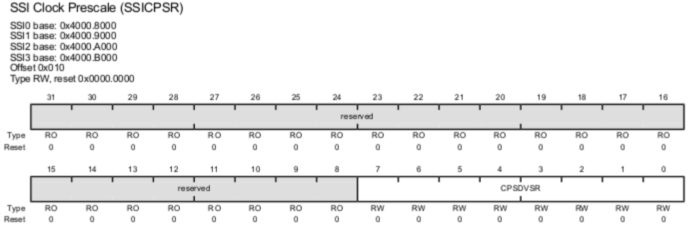
- SSI module clock source can be either of System Clock or PIOSC (Precision Internal Oscilator).
- The selected frequency is fed to prescaler before it is used by the Bit Rate circuitry
- The CPSDVSR (CPS Divisor) value comes from the prescaler divisor register



Setting a Bit Rate

- SSI module clock source can be either of System Clock or PIOSC (Precision Internal Oscilator).
- The selected frequency is fed to prescaler before it is used by the Bit Rate circuitry
- The CPSDVSR (CPS Divisor) value comes from the prescaler divisor register




SSI Clock Prescale (SSICPSR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x010
Type RW, reset 0x0000.0000

	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								CPSDVSR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Setting a Bit Rate

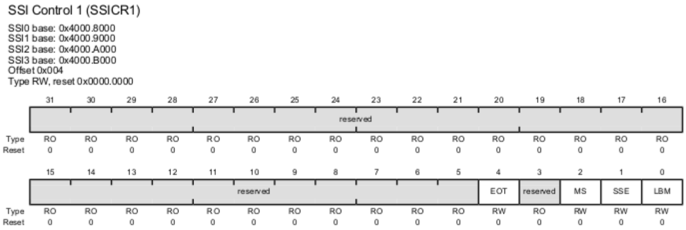
- The lower 8 bits of SSICPSR (SSI Clock Prescale) register are used to divide the CPU clock before it is fed to the Bit Rate circuitry..
- Only even values can be used for the prescaler since the D0 must be 0.
- For the prescaler register, the lowest value is 2 and the highest is 254.
- The SSICR0 (SSI Control register 0) allows the Bit Rate selection among other things.
- The output of clock prescaler circuitry is divided by $1 + \text{SCR}$ and then used as the SSI baud rate clock.
- The value of SCR can be from 0 to 255.

Setting a Bit Rate

- We can use the following formula to calculate the Bit Rate (BR):
  BR=SysClk/(CPSDVSR (1 + SCR))
- **Example 1:** Assume the prescaler register has SSICPSR=0x20 and system clock frequency is 16MHz. Find the values for the SCR part of the SSOCR0 register for the bit rate of (a) 100K, (b) 250K, and (c) 500K.
  Since the prescaler is 0x20 (32 in decimal), we have 16MHz / 32 = 500KHz.
- **Example 2:** In a given program, we have Bit Rate=50KHz and SCR=03 in SSICR0 register. Find the prescaler register value if system clock frequency is 16MHz.

Master or Slave?

- The SSI Module inside the Tiva chip can be Master or Slave.
- We use MS bit in SSI control register 1 (SSICR1) to designate the Tiva chip as master or slave.
- Another important bit in the SSICR1 register is enable/ disable bit (SSE).
- We must disable SSIx module during the initialization process and enable it afterward.



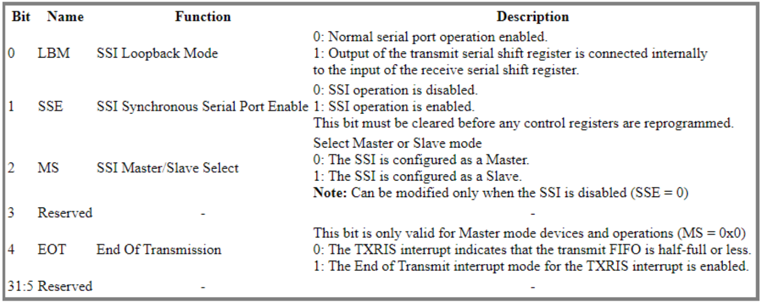
SSI Control 1 (SSICR1)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x004
Type RW, reset 0x0000.0000

	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											EOT	reserved	MS	SSE	LBM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RO	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Master or Slave?

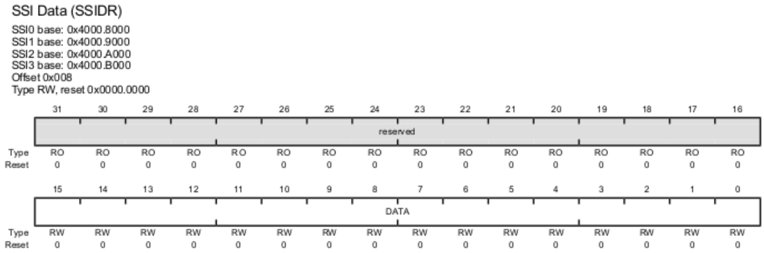


Bit	Name	Function	Description
0	LBM	SSI Loopback Mode	0: Normal serial port operation enabled. 1: Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.
1	SSE	SSI Synchronous Serial Port Enable	0: SSI operation is disabled. 1: SSI operation is enabled. This bit must be cleared before any control registers are reprogrammed.
2	MS	SSI Master/Slave Select	Select Master or Slave mode 0: The SSI is configured as a Master. 1: The SSI is configured as a Slave. **Note:** Can be modified only when the SSI is disabled (SSE = 0)
3	Reserved	-	-
4	EOT	End Of Transmission	This bit is only valid for Master mode devices and operations (MS = 0x0) 0: The TXRIS interrupt indicates that the transmit FIFO is half-full or less. 1: The End of Transmit interrupt mode for the TXRIS interrupt is enabled.
31:5	Reserved	-	-

Table: SSI Control 1 (SSICR1)

Data Register

- The data is placed in SSIDR (SSI Data register) for transmission.
- The SSIDR is also used for the received data buffer.
- In 8-bit data size selection, we must place the data into the lower 8-bits of the register and the rest of the register are unused.
- In the receive mode, data is right-justified meaning the lower 8-bit holds the received data.



SSI Data (SSIDR)

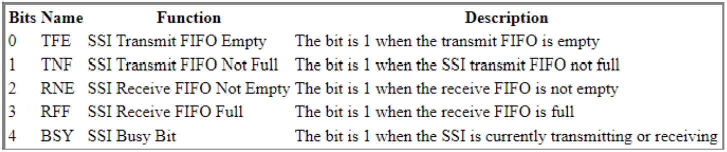
SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x008
Type RW, reset 0x0000.0000

	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

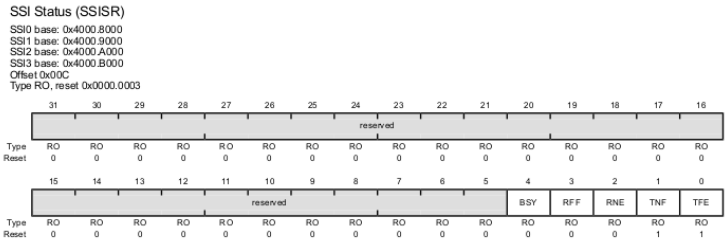
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DATA															
Type	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Status Flag Register

- We use the SSI Status flag register (SSISR) to monitor to see whether a byte has been received
- And if the transmission buffer is empty and ready for the next byte to be transmitted.



Bits	Name	Function	Description
0	TFE	SSI Transmit FIFO Empty	The bit is 1 when the transmit FIFO is empty
1	TNF	SSI Transmit FIFO Not Full	The bit is 1 when the SSI transmit FIFO not full
2	RNE	SSI Receive FIFO Not Empty	The bit is 1 when the receive FIFO is not empty
3	RFF	SSI Receive FIFO Full	The bit is 1 when the receive FIFO is full
4	BSY	SSI Busy Bit	The bit is 1 when the SSI is currently transmitting or receiving



SSI Status (SSISR)

SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x00C
Type RO, reset 0x0000.0003

	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

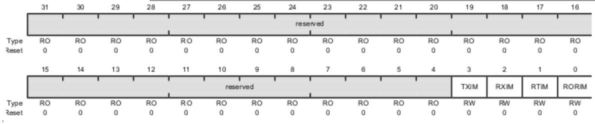
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											BSY	RFF	RNE	TNF	TFE
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Interrupt using NVIC

- We can use an interrupt handler to do the job of transmit and receive.
- By enabling the interrupt bits in SSIIM (SSI Interrupt Mask) register, we direct the interrupt to the NVIC interrupt controller and write an interrupt handler.
- Upon Reset, the interrupts are masked and raising of the flag will not direct it to NVIC.

Bits	Name	Function	Description
0	RORIM	SSI Receive Overrun Interrupt Mask	The receive FIFO overrun interrupt is masked when RORIM is zero and not masked when it is one
1	RTIM	SSI Receive Time-Out Interrupt Mask	The receive FIFO time-out interrupt is masked when RTIM is zero and not masked when it is one
2	RXIM	SSI Receive FIFO Interrupt Mask	The receive FIFO interrupt is masked when RXIM is zero and not masked when it is one
3	TXIM	SSI Transmit FIFO Interrupt Mask	The transmit FIFO interrupt is masked when TXIM is zero and not masked when it is one

SSI Interrupt Mask (SSIIM)
SSI0 base: 0x4000.8000
SSI1 base: 0x4000.9000
SSI2 base: 0x4000.A000
SSI3 base: 0x4000.B000
Offset 0x014
Type RW, reset 0x0000.0000

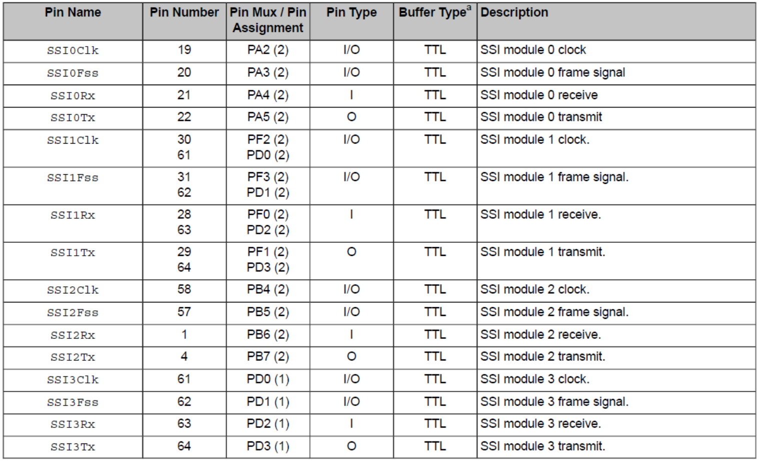


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												TXIM	RXIM	RTIM	RORIM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TM4C123GH SPI Communication Register Setting

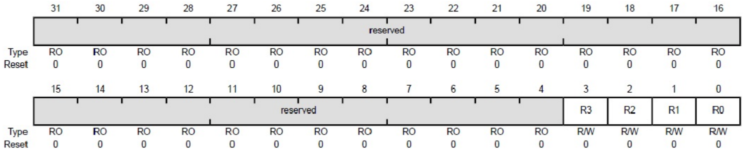
- In our microcontroller, the SPI or SSI channels and pins are altered with the following GPIOs



Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
SSI0Clk	19	PA2 (2)	I/O	TTL	SSI module 0 clock
SSI0Fss	20	PA3 (2)	I/O	TTL	SSI module 0 frame signal
SSI0Rx	21	PA4 (2)	I	TTL	SSI module 0 receive
SSI0Tx	22	PA5 (2)	O	TTL	SSI module 0 transmit
SSI1Clk	30 61	PF2 (2) PD0 (2)	I/O	TTL	SSI module 1 clock.
SSI1Fss	31 62	PF3 (2) PD1 (2)	I/O	TTL	SSI module 1 frame signal.
SSI1Rx	28 63	PF0 (2) PD2 (2)	I	TTL	SSI module 1 receive.
SSI1Tx	29 64	PF1 (2) PD3 (2)	O	TTL	SSI module 1 transmit.
SSI2Clk	58	PB4 (2)	I/O	TTL	SSI module 2 clock.
SSI2Fss	57	PB5 (2)	I/O	TTL	SSI module 2 frame signal.
SSI2Rx	1	PB6 (2)	I	TTL	SSI module 2 receive.
SSI2Tx	4	PB7 (2)	O	TTL	SSI module 2 transmit.
SSI3Clk	61	PD0 (1)	I/O	TTL	SSI module 3 clock.
SSI3Fss	62	PD1 (1)	I/O	TTL	SSI module 3 frame signal.
SSI3Rx	63	PD2 (1)	I	TTL	SSI module 3 receive.
SSI3Tx	64	PD3 (1)	O	TTL	SSI module 3 transmit.

TM4C123GH SPI Communication Register Setting

- RCGCSSI register : Since we have four modules for SSI communication we have this 4 bit wide register.
- Writing values from LSB decide which module has to be taken.
- Since we are going to use channel 2 writing (1¡¡2) will enable this module.

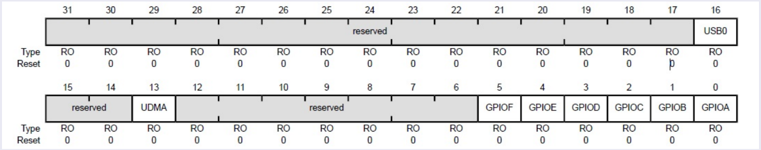


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TM4C123GH SPI Communication Register Setting

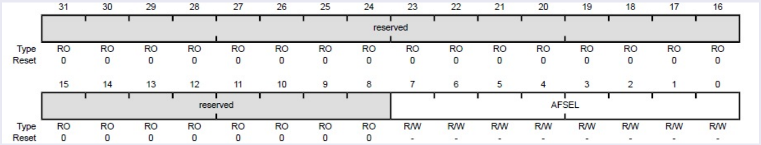
- RCGC2 regiser : we used this register before. SSI2 is present in the GPIOB



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								reserved								USB0
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		UDMA				reserved				GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
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

- AFSEL register : This register selects the PORTB for alternative function. We have discussed in GPIO programming

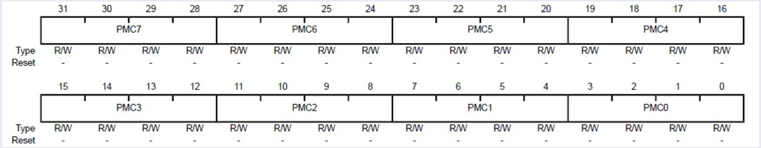


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
									reserved							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				reserved								AFSEL				
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

TM4C123GH SPI Communication Register Setting

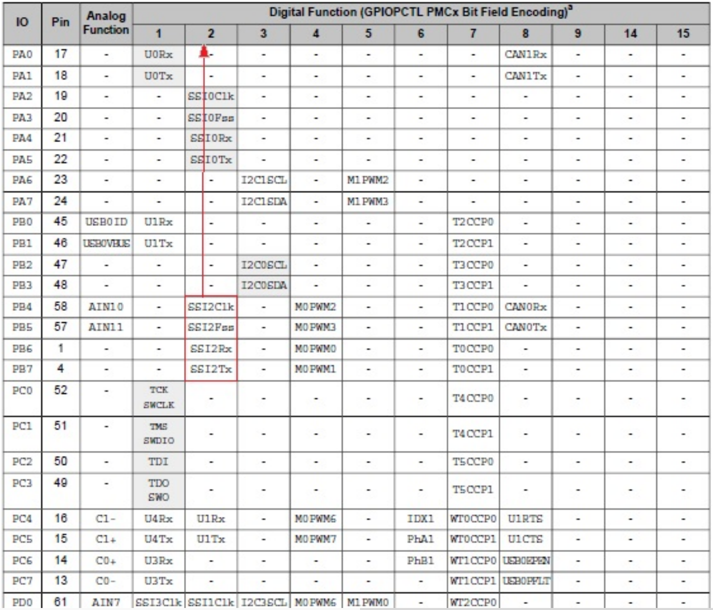
- PCTL register : We have set the functionality of GPIOB to alternative but each single pin of this microcontroller consist more one single functions.
- So we need to tell the microcontroller that we want SSI functionality on this GPIO.
- For selecting the functionality of each PIN of the GPIO we have the following 8 field in PCTL, each 4 bit long The value for each pin can be found by this table



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	PMC7				PMC6				PMC5				PMC4			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PMC3				PMC2				PMC1				PMC0			
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

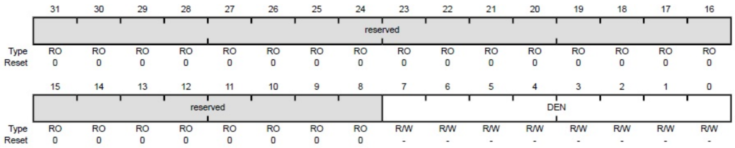
TM4C123GH SPI Communication Register Setting



IO	Pin	Analog Function	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[a]										
			1	2	3	4	5	6	7	8	9	14	15
PA0	17	-	U0Rx	-	-	-	-	-	-	CAN1Rx	-	-	-
PA1	18	-	U0Tx	-	-	-	-	-	-	CAN1Tx	-	-	-
PA2	19	-	-	SSI0Clk	-	-	-	-	-	-	-	-	-
PA3	20	-	-	SSI0Fss	-	-	-	-	-	-	-	-	-
PA4	21	-	-	SSI0Rx	-	-	-	-	-	-	-	-	-
PA5	22	-	-	SSI0Tx	-	-	-	-	-	-	-	-	-
PA6	23	-	-	-	I2C1SCL	-	M1PWM2	-	-	-	-	-	-
PA7	24	-	-	-	I2C1SDA	-	M1PWM3	-	-	-	-	-	-
PB0	45	USB0ID	U1Rx	-	-	-	-	-	T2CCP0	-	-	-	-
PB1	46	USB0VBUS	U1Tx	-	-	-	-	-	T2CCP1	-	-	-	-
PB2	47	-	-	-	I2C0SCL	-	-	-	T3CCP0	-	-	-	-
PB3	48	-	-	-	I2C0SDA	-	-	-	T3CCP1	-	-	-	-
PB4	58	AIN10	-	SSI2Clk	-	M0PWM2	-	-	T1CCP0	CAN0Rx	-	-	-
PB5	57	AIN11	-	SSI2Fss	-	M0PWM3	-	-	T1CCP1	CAN0Tx	-	-	-
PB6	1	-	-	SSI2Rx	-	M0PWM0	-	-	T0CCP0	-	-	-	-
PB7	4	-	-	SSI2Tx	-	M0PWM1	-	-	T0CCP1	-	-	-	-
PC0	52	-	TCK SWCLK	-	-	-	-	-	T4CCP0	-	-	-	-
PC1	51	-	TMS SWDIO	-	-	-	-	-	T4CCP1	-	-	-	-
PC2	50	-	TDI	-	-	-	-	-	T5CCP0	-	-	-	-
PC3	49	-	TDO SWO	-	-	-	-	-	T5CCP1	-	-	-	-
PC4	16	C1-	U4Rx	U1Rx	-	M0PWM6	-	IDX1	WT0CCP0	U1RTS	-	-	-
PC5	15	C1+	U4Tx	U1Tx	-	M0PWM7	-	PhA1	WT0CCP1	U1CTS	-	-	-
PC6	14	C0+	U3Rx	-	-	-	-	PhB1	WT1CCP0	USB0EPEN	-	-	-
PC7	13	C0-	U3Tx	-	-	-	-	-	WT1CCP1	USB0PFLT	-	-	-
PD0	61	AIN7	SSI3Clk	SSI1Clk	I2C3SCL	M0PWM6	M1PWM0	-	WT2CCP0	-	-	-	-

TM4C123GH SPI Communication Register Setting

- we want SSI on GPIOB hence writting 2 at the 4th , 5th ,6th and 7th bit in the PCTL will set the pin at desire functionality.
- So the value of 0x22220000 will be written in the PCTL.
- DEN register : SSI is a digital signal interface hence each GPIO pin should be enabled for digital signal.

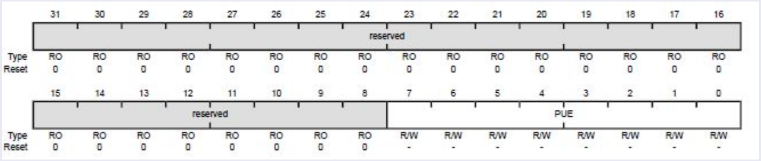


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DEN							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

TM4C123GH SPI Communication Register Setting

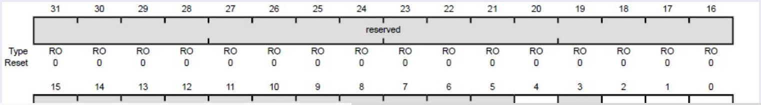
- PUR register : SSI is connected at 4,5,6 and 7th bit. Hence pull up would require for each.
- value for this register will be 1 for every used bit. Above registers are also discussed in GPIO programming section



	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								PUE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

- SSI Control 1 (SSICR1) : The SSICR1 register contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

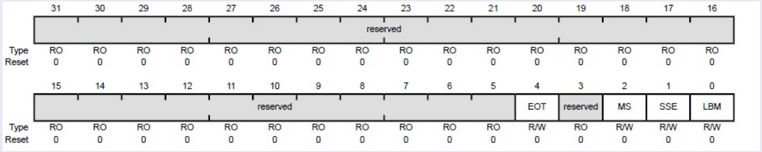


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TM4C123GH SPI Communication Register Setting

- SSI Control 1 (SSICR1) : The SSICR1 register contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.



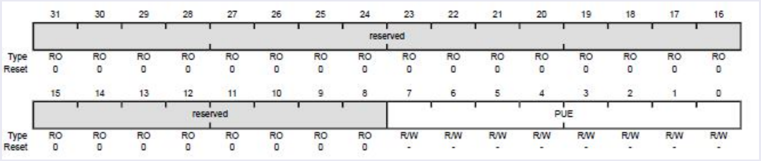
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved											EOT	reserved	MS	SSE	LBM
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	RO	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- EOT End of Transmission bit is only valid for Master mode devices and operations for interrupt method.
- MS bit decides the role of device 1 means slave and 0 means the master mode,SSE starts the SSI module.
- LBM enables the loop-back. When LBM gets 1 means Output of the transmit serial shift register is connected internally to the input of the receive serial shift register.

TM4C123GH SPI Communication Register Setting

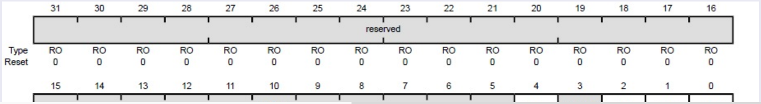
- PUR register : SSI is connected at 4,5,6 and 7th bit. Hence pull up would require for each.
- value for this register will be 1 for every used bit. Above registers are also discussed in GPIO programming section



	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								PUE							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	-	-	-	-	-	-	-	-

- SSI Control 1 (SSICR1) : The SSICR1 register contains bit fields that control various functions within the SSI module. Master and slave mode functionality is controlled by this register.

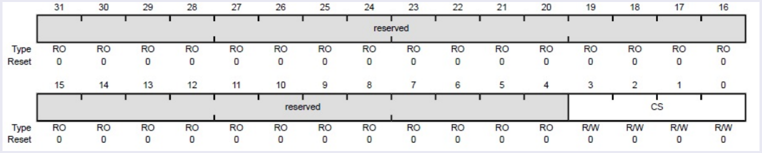


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

TM4C123GH SPI Communication Register Setting

- SSI Clock Configuration (SSICC) : This register selects the clock source for the SSI module.



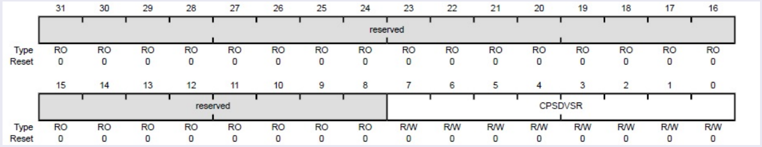
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved												CS			
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- Writing this register to 0x5 will select the internal precision oscillator of 16 Mhz while making it zero means the cpu clock will be used depends up-to PLL and other settings.

TM4C123GH SPI Communication Register Setting

- SSI Clock Prescale (SSICPSR) : This is 8 bit wide register shown below.

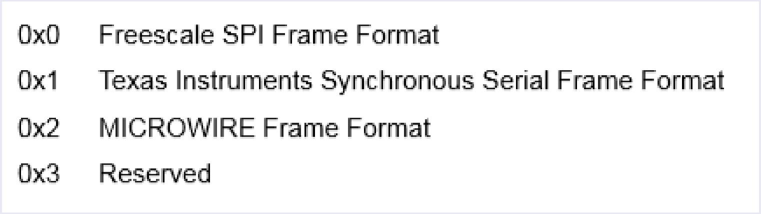


	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								CPSDVSR							
Type	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- The SSICPSR register specifies the division factor which is used to derive the SSInClk from the system clock.
- The clock is further divided by a value from 1 to 256, which is 1 + SCR.
- SCR is programmed in the SSICR0 register. The frequency of the SSInClk is defined by:
  SSInClk = SysClk / (CPSDVSR * (1 + SCR)) where the SCR value is calculated by the SSICR0 register.

TM4C123GH SPI Communication Register Setting

- SSI Control 0 (SSICR0) Register : This is 16 bit wide register as shown below

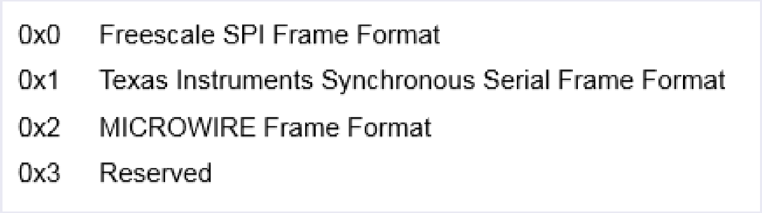


0x0	Freescale SPI Frame Format
0x1	Texas Instruments Synchronous Serial Frame Format
0x2	MICROWIRE Frame Format
0x3	Reserved

- This register selects the type of SSI communication.
- There are more settings in terms of phase and polarity.
- The phase is selected by SPH bit and the polarity of clock is determined by th SPO bit for most of the cases they are going to be zero
- which means the data will be sampled on first clock edge and the steady state clock will be at low.

TM4C123GH SPI Communication Register Setting

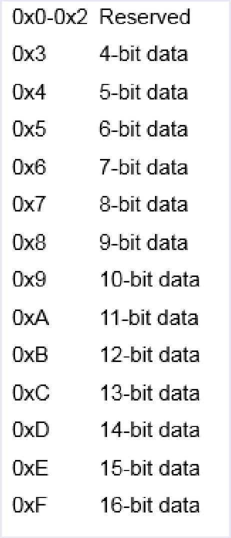
- The FRF field decides the communication format



0x0	Freescale SPI Frame Format
0x1	Texas Instruments Synchronous Serial Frame Format
0x2	MICROWIRE Frame Format
0x3	Reserved

- The format we are going to use is Freescale format hence this field will be zero
- DSS field selects the data bit format that has to be sent in one single burst and they can be
- We are going with 8 bit data format in this example. We are using SCR =0 hence SSInClk = 16 Mhz / 64 = 250000 Hz.

TM4C123GH SPI Communication Register Setting



0x0-0x2	Reserved
0x3	4-bit data
0x4	5-bit data
0x5	6-bit data
0x6	7-bit data
0x7	8-bit data
0x8	9-bit data
0x9	10-bit data
0xA	11-bit data
0xB	12-bit data
0xC	13-bit data
0xD	14-bit data
0xE	15-bit data
0xF	16-bit data

TM4C123GH SPI Communication Program

```
#include <stdint.h>
#include <stdlib.h>
#include "TM4C123GH6PM.h"
void printChar(char c);
void printString(char * string);
void send_byte(char data);
```

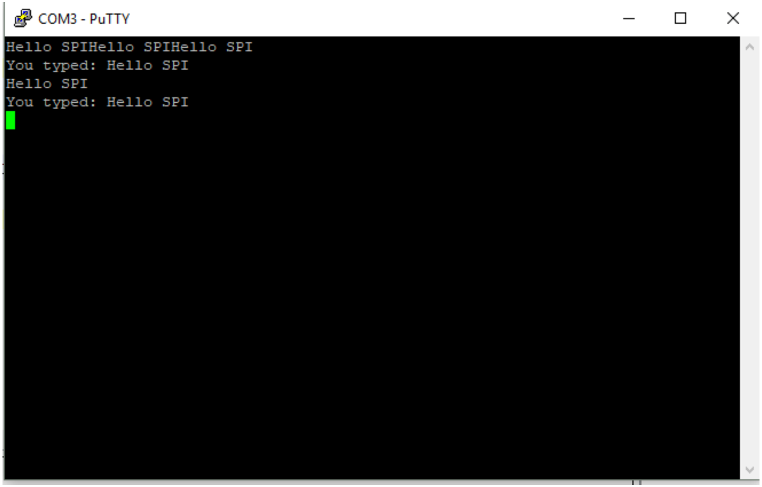
```
void uart_ini(void){
 SYSCTL_RCGCUART_R|= (1<<0); //Enable the UART0 module
 SYSCTL_RCGCGPIO_R |= (1<<0);   //enable clock to PORTA
 GPIO_PORTA_AFSEL_R=(1<<1)|(1<<0); //Enable PA0 and PA1 As Alternate Function PIN
 GPIO_PORTA_PCTL_R|=0x00000001|0x00000010; // make PA0 PA1 UART output pin
 GPIO_PORTA_DEN_R|=(1<<0)|(1<<1); //Enable digital on PA0 PA1
 UART0_CTL_R&= ~(1<<0); //Disable the UART by clearing the UARTEN
 UART0_IBRD_R=104;  // integer portion of the BRD
 UART0_FBRD_R=11;   //fractional portion of the BRD
 UART0_LCRH_R=(0x3<<5)|(1<<4); // 8-bit, no parity, 1-stop bit
 UART0_CTL_R = (1<<0)|(1<<8)|(1<<9); //Enable the UART by setting the UARTEN bit
}
```

```
void spi_master_ini(void){
 SYSCTL_RCGCSSI_R|=(1<<2); //selecting SSI2 module
 //SYSCTL_RCGC2_R|=(1<<1);   //providing clock to PORTB
     SYSCTL_RCGCGPIO_R |= (1<<1);   //enable clock to PORTA
 GPIO_PORTB_AFSEL_R|=(1<<4)|(1<<5)|(1<<6)|(1<<7);//selecting alternative fuctions
 GPIO_PORTB_PCTL_R=0x22220000;//selecting SSI as alternative function
 GPIO_PORTB_DEN_R|=(1<<4)|(1<<5)|(1<<6)|(1<<7);//enabling digital mode for PORTB 4,5,6,7
 GPIO_PORTB_PUR_R|=(1<<4)|(1<<5)|(1<<6)|(1<<7);//selecting pull ups for 4,5,6,7
 SSI2_CR1_R=0;        //disabling SSI module for settings
 SSI2_CC_R=0;         //using main system clock
 SSI2_CPSR_R=64;      //selecting divisor 64 for SSI clock
 SSI2_CR0_R=0x7;      //freescale mode, 8 bit data, steady clock low
 SSI2_CR1_R|=(1<<1)|(1<<0);   //enabling SSI/LoopBack (Internal SPI Receiver )
}
```

```
void printString(char * string) {
 while(*string) {
  printChar(*(string++));
 }
}
//function for sending each byte of string one by one
void send_str(char *buffer){
 while(*buffer!=0){
 send_byte(*buffer);
  buffer++;
 }
}
void send_byte(char data){
 while((SSI2_SR_R&(1<<0))==0); //waiting for transmission to be done
     SSI2_DR_R=data; //putting the byte to send from SSI
}
char rec_char(void) {
 char c;
 while((SSI2_SR_R&(1<<2))==0); //waiting for transmission to be done
 c=SSI2_DR_R; //putting the byte to send from SSI
 return c;
}
char* rec_str(char delimiter) {
 int stringSize = 0;
 char* string = (char*)calloc(15,sizeof(char));
 char c = rec_char();
 printChar(c);
 while(c!=delimiter) {
  *(string+stringSize) = c; // put the new character at the end of the array
  stringSize++;
  c = rec_char();
  printChar(c); // display the character the user typed
 }
 return string;
}
void printChar(char c)  {
  while((UART0_FR_R & (1<<5)) != 0);
  UART0_DR_R = c;
}
```

```
int main(){
     spi_master_ini(); //initializing controller as master
     uart_ini();
     send_str("Hello SPI\r");//sending string
     while(1){
     char* string = rec_str('\r');
     printString("\n\r");
     printString("You typed: ");
     printString(string);
     printString("\n\r");
     free(string);
     }
     return 0;
}
```

SPI Communication Output



```
Hello SPIHello SPIHello SPI
You typed: Hello SPI
Hello SPI
You typed: Hello SPI
```

Thank You