Lecture 27: LM3S9B96 Microcontroller – Inter-Integrated Circuit (I^2C) Interface

Assistant Prof. Hongzi Zhu

hongzi@cs.sjtu.edu.cn

Stellaris® LM3S9B96 Microcontroller Data Sheet

Chapter 16

Inter-Integrated Circuit (I^2C) Interface

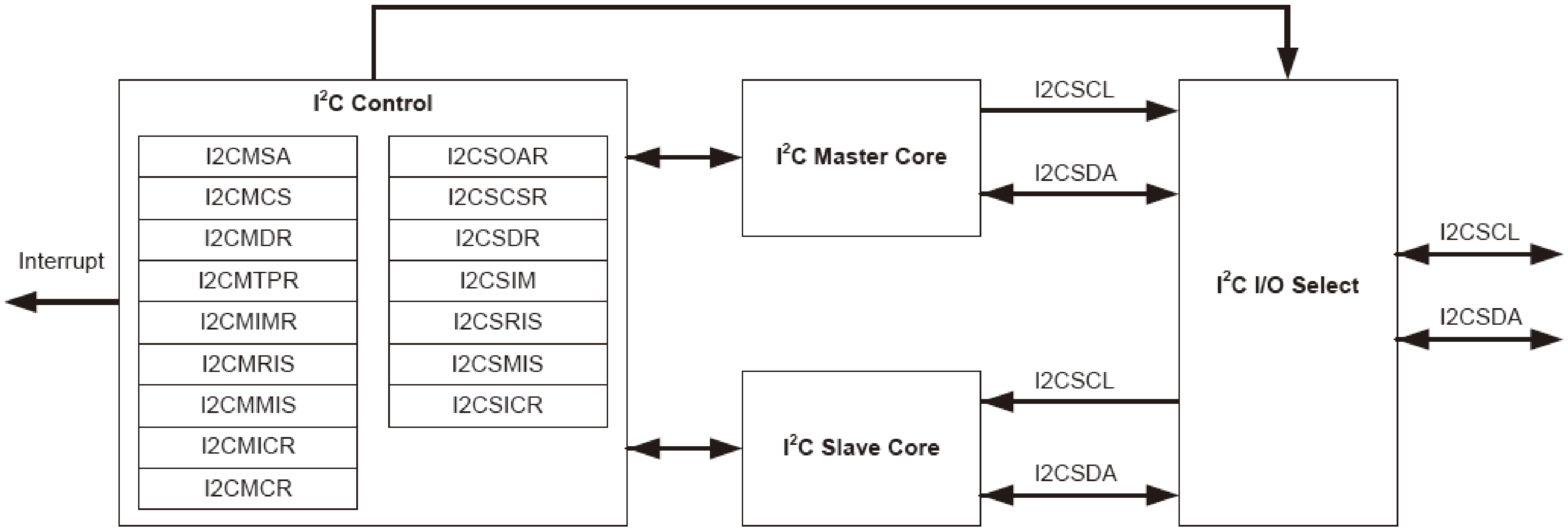
Inter-Integrated Circuit (I^2C) Interface

- The Inter-Integrated Circuit (I^2C) bus :
  - bi-directional data transfer via a two-wire design: a serial data line SDA and a serial clock line SCL
  - interfaces to external I^2C devices: serial memory, networking devices, LCDs, tone generators

Inter-Integrated Circuit (I^2C) Interface

- The LM3S9B96 microcontroller includes two I^2C modules:
  - Supports both transmitting and receiving data as either a master or a slave
  - Four I^2C modes: Master/Slave transmit/receive
  - Two transmission speeds: Standard (100 Kbps) and Fast (400 Kbps)
  - Master and slave interrupt generation
  - Master with arbitration and clock synchronization, multi-master support, and 7-bit addressing mode

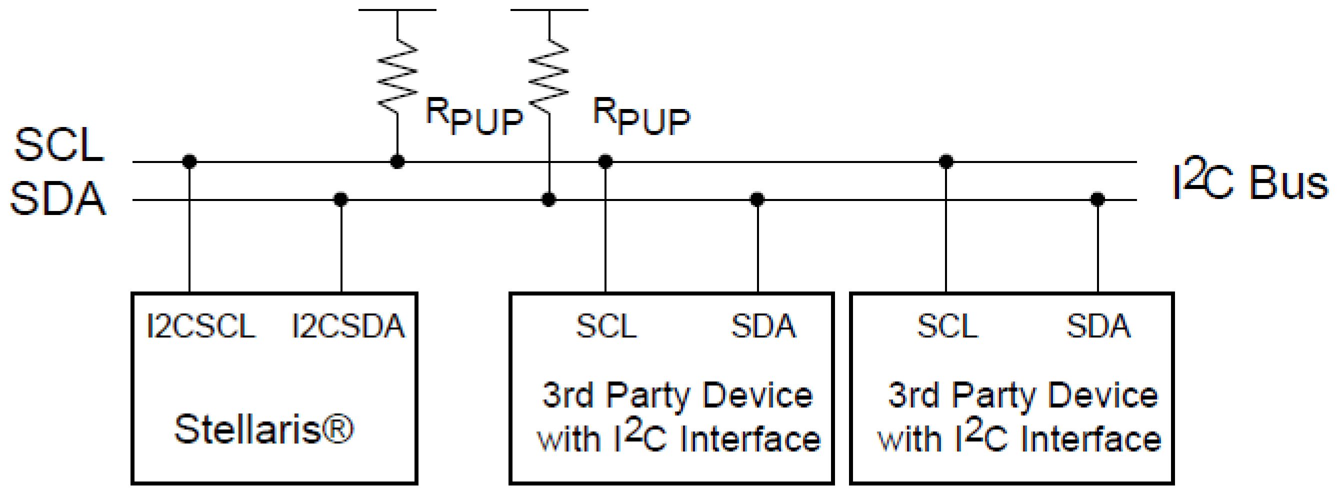
Block Diagram



Functional Description

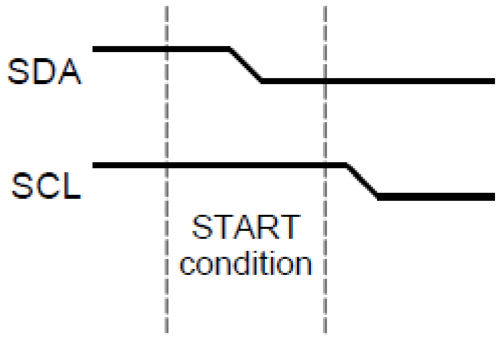
- The I^{2}C bus uses only two signals: SDA and SCL, named I2CSDA and I2CSCL on Stellaris microcontrollers
- The bus is considered idle when both lines are High
- Every transaction on the I2C bus is nine bits long, i.e., 8 data bits (MSB first) and 1 single acknowledge bit
- The number of bytes in one transfer is unrestricted
- When a receiver cannot receive another complete byte, it can hold the clock line SCL Low and force the transmitter into a wait state
- The data transfer continues when the receiver releases the clock SCL

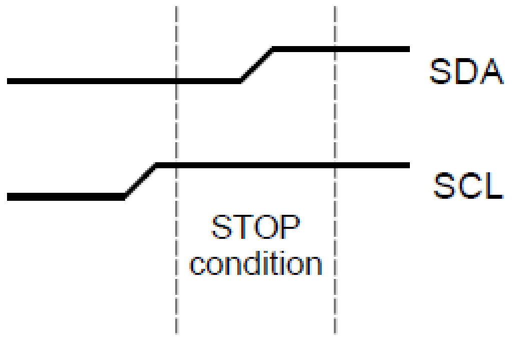
I^{2}C Bus Configuration


RPUP
RPUP
SCL
SDA
I2C Bus
I2CSCL
I2CSDA
Stellaris®
SCL
SDA
3rd Party Device with I2C Interface
SCL
SDA
3rd Party Device with I2C Interface

START and STOP Conditions

- The protocol of the I^{2}C bus defines two states to begin and end a transaction: START and STOP
  - A High-to-Low transition on the SDA line while the SCL is High is defined as a START condition
  - A Low-to-High transition on the SDA line while SCL is High is defined as a STOP condition

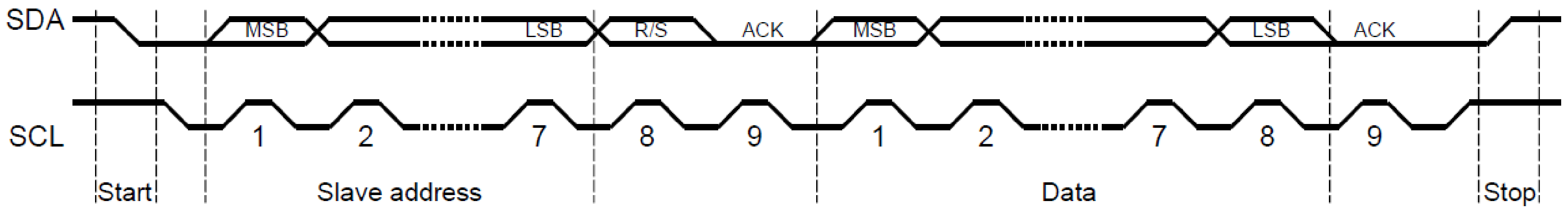


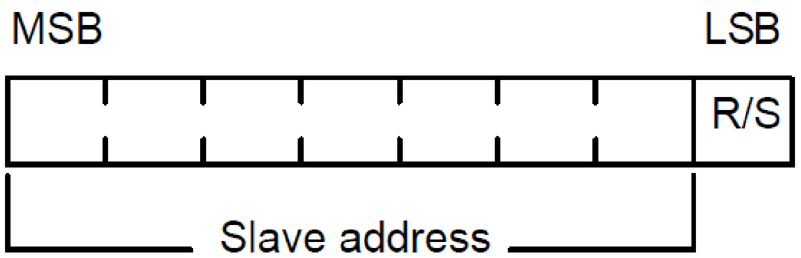


Data Format with 7-Bit Address

- After the START condition, a slave address is transmitted
- The address is 7 bits long and followed by a direction bit (R/S bit in the **I2C Master Slave Address (I2CMSA)** register)
  - R/S is cleared, a transmit operation (send)
  - R/S is set, a request for data (receive)
- A data transfer is always terminated by a STOP condition generated by the master
- A master can initiate communications with another device on the bus by generating a repeated START condition and addressing another slave without first generating a STOP condition.

Data Format with 7-Bit Address

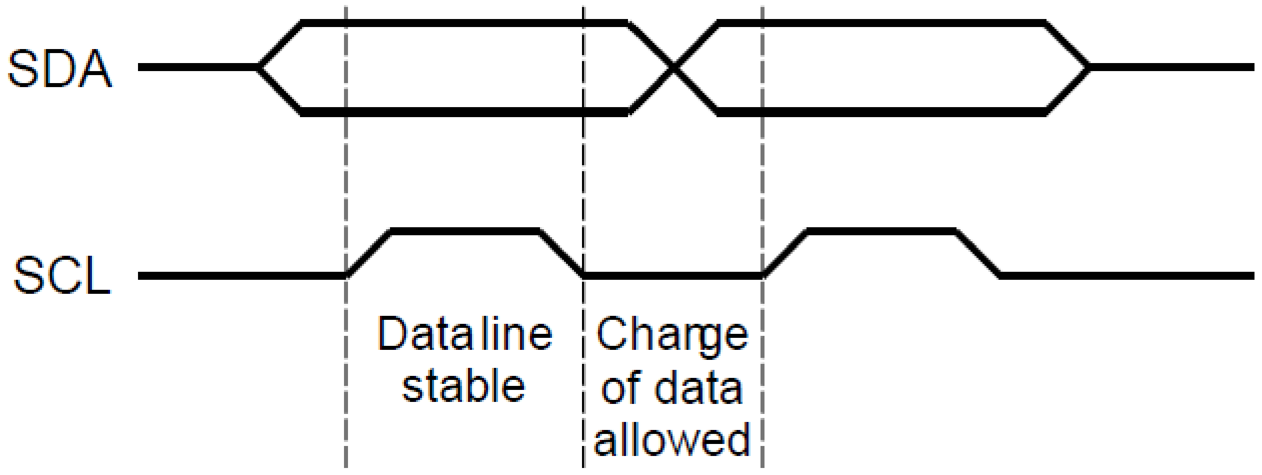




⌘ A zero in the R/S position of the first byte means that the master transmits (sends) data to the selected slave, and a one in this position means that the master receives data from the slave

Data Validity

- The data on the SDA line must be stable during the high period of the clock
- The data line can only change when SCL is Low



Arbitration

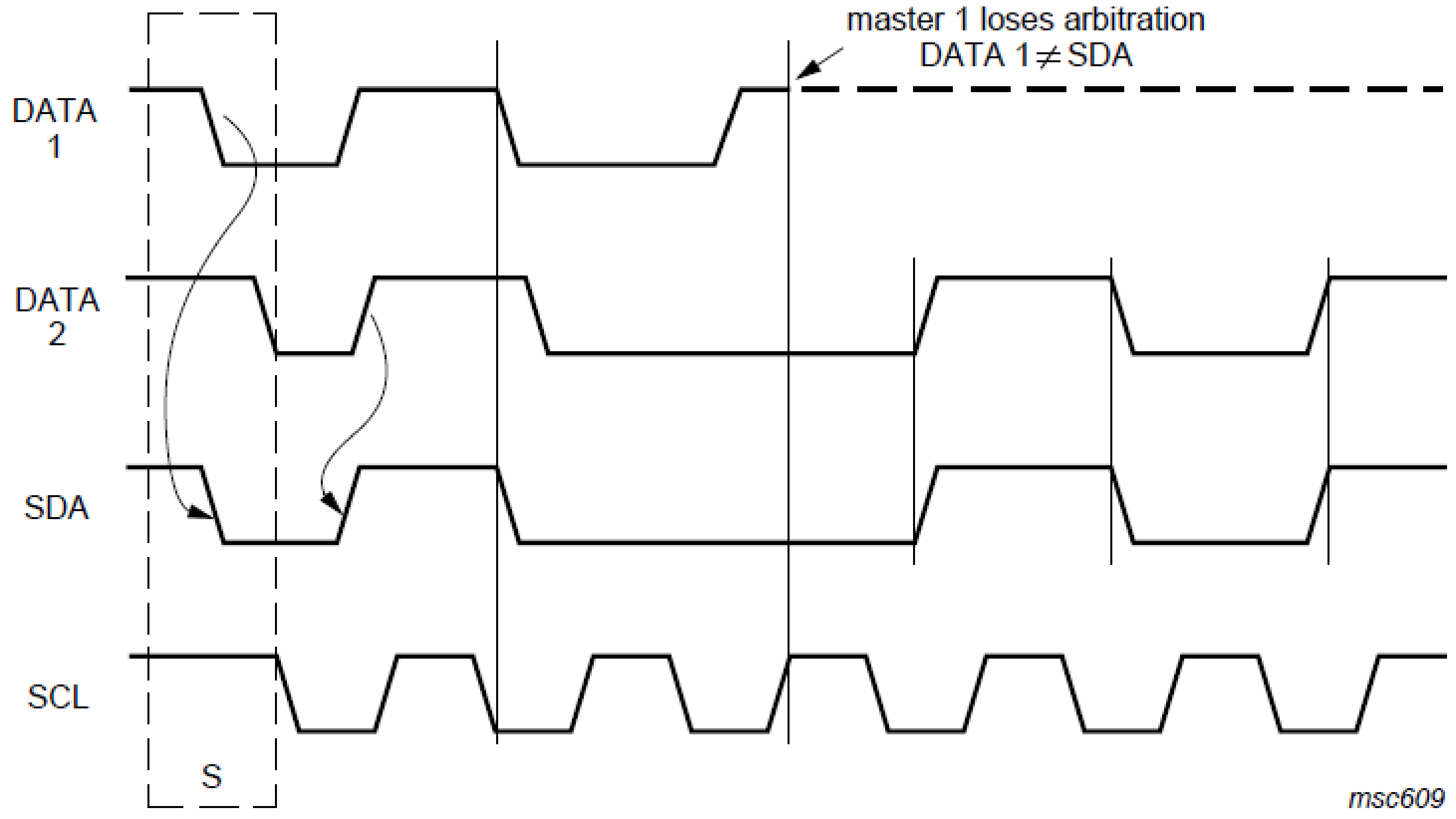


Fig 8. Arbitration procedure of two masters

Acknowledge

- All bus transactions have a required acknowledge clock cycle
- During the acknowledge cycle, the transmitter (master or slave) releases the SDA line
- To acknowledge the transaction, the receiver must pull down SDA during the acknowledge clock cycle
  - When a slave receiver does not ack the slave address, the master can generate a STOP condition and abort the current transfer.
  - When the master acting as a receiver, it is responsible for acknowledging each transfer made by the slave
  - The master receiver controls the number of bytes in the transfer by not generating an acknowledge on the last data byte; slave transmitter will give up the data line

Available Speed Modes

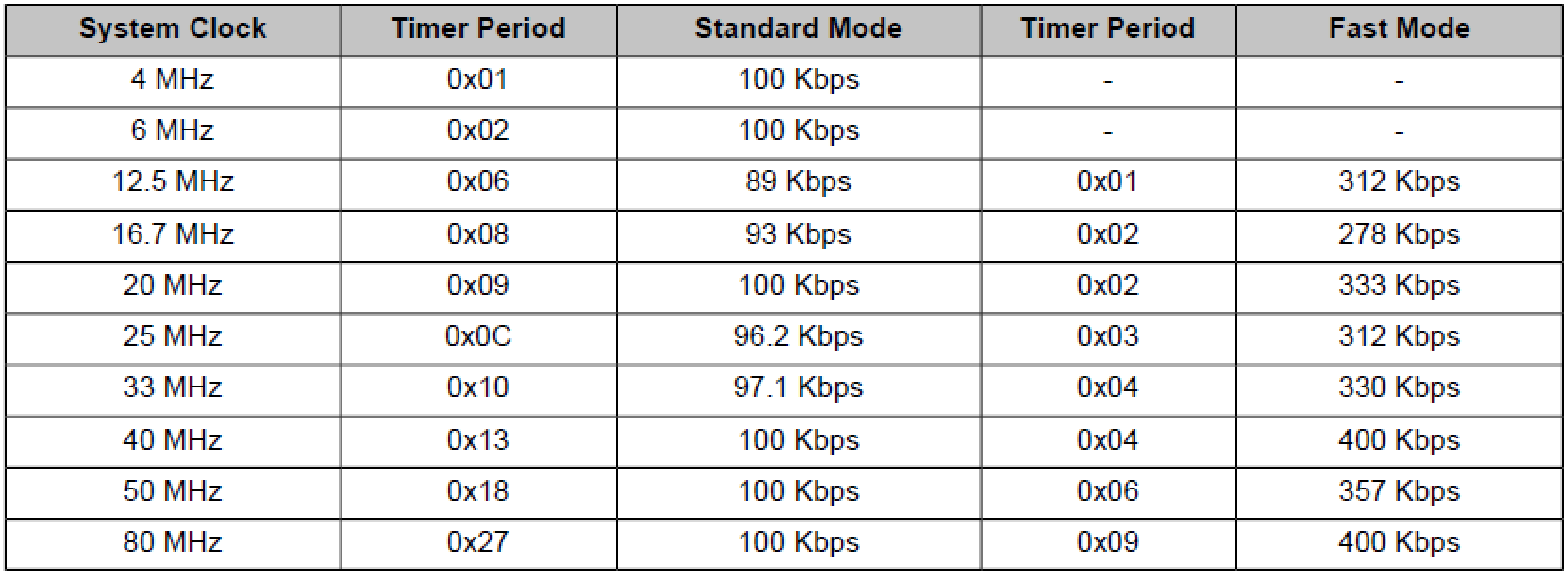
- The I^2C bus can run in either Standard mode (100 kbps) or Fast mode (400 kbps)
- The selected mode should match the speed of the other I^2C devices on the bus
- The mode is selected by using a value in the **I2C Master Timer Period (I2CMTPR)** register
- The I2C clock rate is determined by the parameters:
  - CLK_PRD is the system clock period
  - SCL_LP is the low phase of SCL (fixed at 6)
  - SCL_HP is the high phase of SCL (fixed at 4)
  - TIMER_PRD is the programmed value in the **I2CMTPR** register

Available Speed Modes

$\mathit{SCL\_PERIOD} = 2 \times (1 + \mathit{TIMER\_PRD}) \times (\mathit{SCL\_LP} + \mathit{SCL\_HP}) \times \mathit{CLK\_PRD}$

- For example: $\mathit{CLK\_PRD} = 50\ \mathit{ns}$, $\mathit{TIMER\_PRD} = 2$, $\mathit{SCL\_LP}=6$, $\mathit{SCL\_HP}=4$

  Therefore, the SCL frequency is : $1/\mathit{SCL\_PERIOD} = 333\ \mathit{Khz}$



System Clock	Timer Period	Standard Mode	Timer Period	Fast Mode
4 MHz	0x01	100 Kbps	-	-
6 MHz	0x02	100 Kbps	-	-
12.5 MHz	0x06	89 Kbps	0x01	312 Kbps
16.7 MHz	0x08	93 Kbps	0x02	278 Kbps
20 MHz	0x09	100 Kbps	0x02	333 Kbps
25 MHz	0x0C	96.2 Kbps	0x03	312 Kbps
33 MHz	0x10	97.1 Kbps	0x04	330 Kbps
40 MHz	0x13	100 Kbps	0x04	400 Kbps
50 MHz	0x18	100 Kbps	0x06	357 Kbps
80 MHz	0x27	100 Kbps	0x09	400 Kbps

Interrupts

- The I2C can generate interrupts when the following conditions are observed:
  - Master transaction completed
  - Master transaction error
  - Slave transaction received
  - Slave transaction requested
  - Stop condition on bus detected
  - Start condition on bus detected
- The I2C master and I2C slave modules have separate interrupt signals
- A module can generate interrupts for multiple conditions but only one single interrupt signal is sent to the interrupt controller

I^{2}C Master Interrupts

- ⌘ When a transaction completes or when an error occurs during a transaction
- ⌘ To enable the I^{2}C master interrupt, the IM bit in the **I2C Master Interrupt Mask (I2CMIMR)** register must be set
- ⌘ When an interrupt condition is met, software must check the ERROR bit in the **I2C Master Control/Status (I2CMCS)** register to verify that an error didn't occur
- ⌘ The interrupt is cleared by writing a **1** to the IC bit in the **I2C Master Interrupt Clear (I2CMICR)** register
- ⌘ If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I2C Master Raw Interrupt Status (I2CMRIS)** register

I^{2}C Slave Interrupts

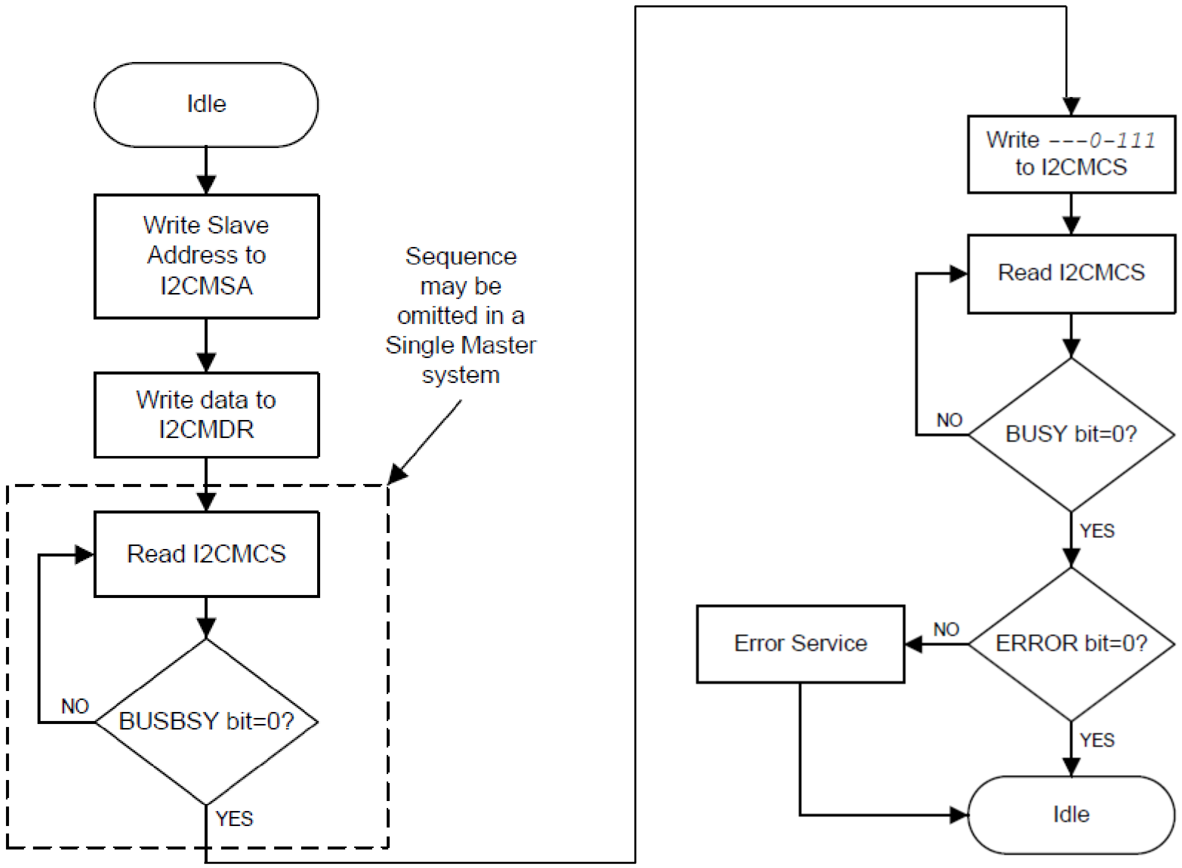
- When data has been received or requested
- This interrupt is enabled by setting the DATAIM bit in the in the **I2C Slave Interrupt Mask (I2CSIMR)** register
- Software determines whether the module should write (transmit) or read (receive) data from the **I2C Slave Data (I2CSDR)** register, by checking the RREQ and TREQ bits of the **I2C Slave Control/Status (I2CSCSR)** register
- The interrupt is cleared by setting the DATAIC bit in the **I2C Slave Interrupt Clear (I2CSICR)** register
- In addition, the slave module can generate an interrupt when a start and stop condition is detected (STARTIM and STOPIM bits in **I2CSIMR**)
- If the application doesn't require the use of interrupts, the raw interrupt status is always visible via the **I2C Slave Raw Interrupt Status (I2CMRIS)** register

Loopback Operation

- The I^{2}C modules can be placed into an internal loopback mode for diagnostic or debug work
  - Set the LPBK bit in the **I2C Master Configuration (I2CMCR)** register
  - SDA and SCL signals from the master and slave modules are tied together

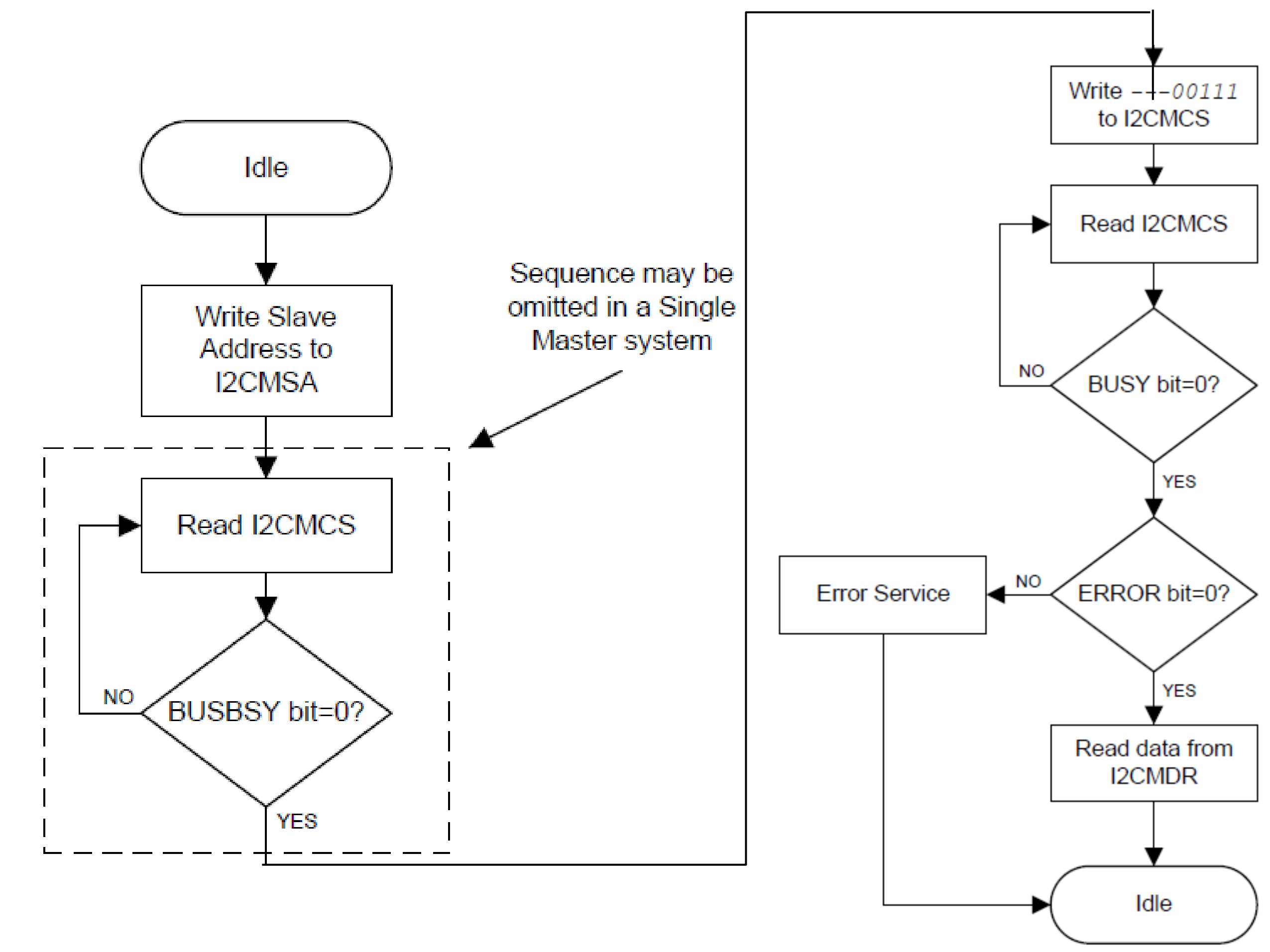
I^{2}C Master Command Sequences

Master Single TRANSMIT



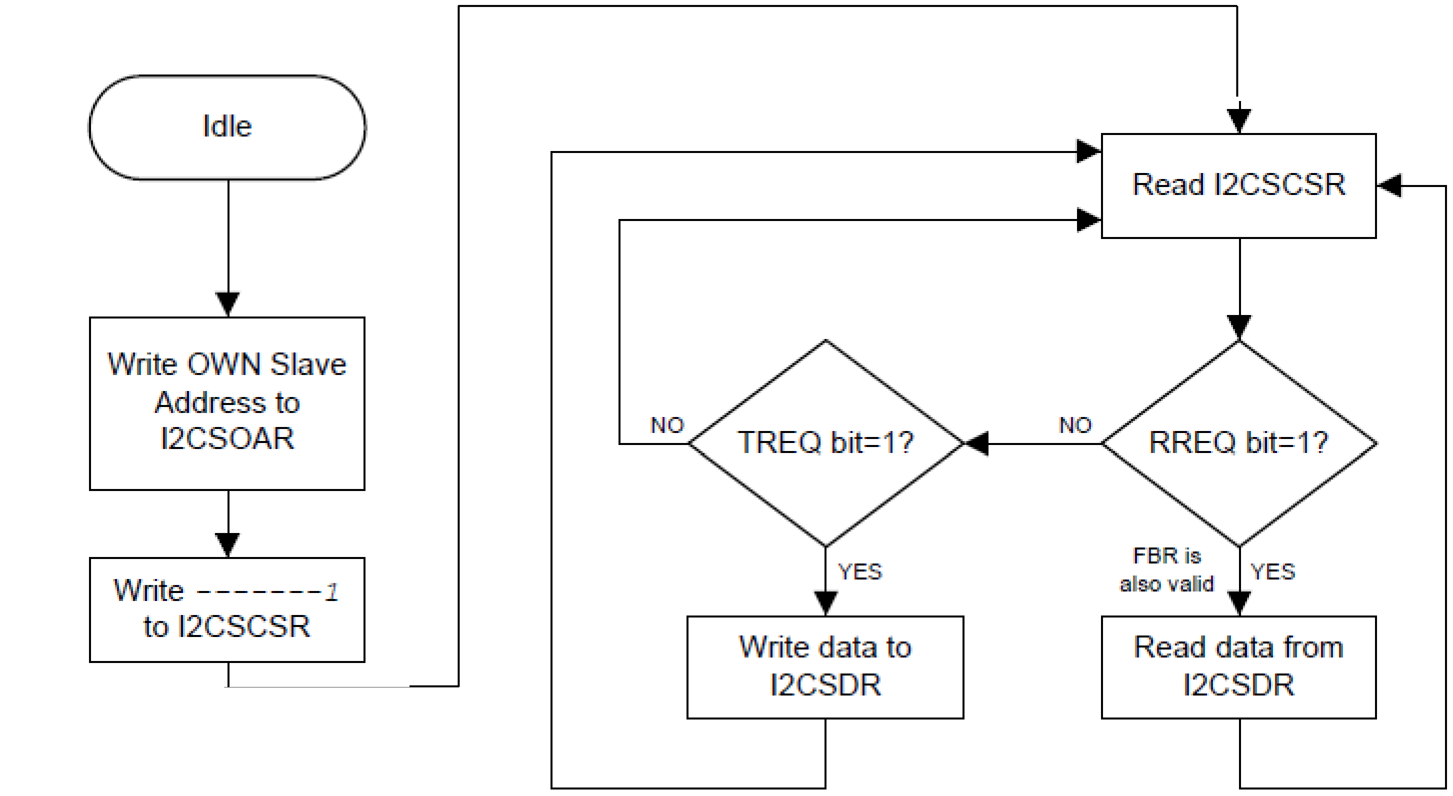
I^{2}C Master Command Sequences

Master Single RECEIVE



I^{2}C Slave Command Sequences

Slave Command Sequence



Initialization and Configuration

- The following example shows how to configure the I^{2}C module to transmit a single byte as a master:
  - 1. Enable the I^{2}C clock by writing a value of 0x00001000 to the **RCGC1** register in the System Control module
  - 2. Enable the clock to the appropriate GPIO module via the **RCGC2** register in the System Control module
  - 3. In the GPIO module, enable the appropriate pins for their alternate function using the **GPIOAFSEL** register
  - 4. Enable the I^{2}C pins for Open Drain operation
  - 5. Configure the PMCn fields in the **GPIOPCTL** register to assign the I2C signals to the appropriate
  - 6. Initialize the I2C Master by writing the **I2CMCR** register with a value of 0x00000010

Initialization and Configuration

- 7. Set the desired SCL clock speed of 100 Kbps by writing the **I2CMTPR** register with the correct value:

  (SYSTEM CLOCK/(2*(SCL_LP + SCL_HP)*SCL_CLK))-1;

  = (20MHZ/(2*(6+4)*100000))-1;

  = 9
- 8. Specify the slave address of the master and that the next operation is a Transmit by writing the **I2CMSA** register with a value of 0x00000076, which sets the slave address to 0x3B
- 9. Place data (byte) to be transmitted in the data register by writing the **I2CMDR** register with the desired data
- 10. Initiate a single byte transmit of the data from Master to Slave by writing the **I2CMCS** register with a value of 0x00000007 (STOP, START, RUN)
- 11. Wait until the transmission completes by polling the **I2CMCS** register's BUSBSY bit until it has been cleared.

Register Map

- The I^2C’s base address:
  - I^2C Master 0: 0x4002.0000
  - I^2C Slave 0: 0x4002.0800
  - I^2C Master 1: 0x4002.1000
  - I^2C Slave 1: 0x4002.1800
- Table 16-4 on page 701 lists the I^2C interface registers.
- For detailed register descriptions, refer to Chapter 16.6