Serial Communications II

ENCE361 Embedded Systems 1

Course Coordinator: Ciaran Moore (ciaran.moore@Canterbury.ac.nz)

Lecturer: Le Yang (le.yang@canterbury.ac.nz)

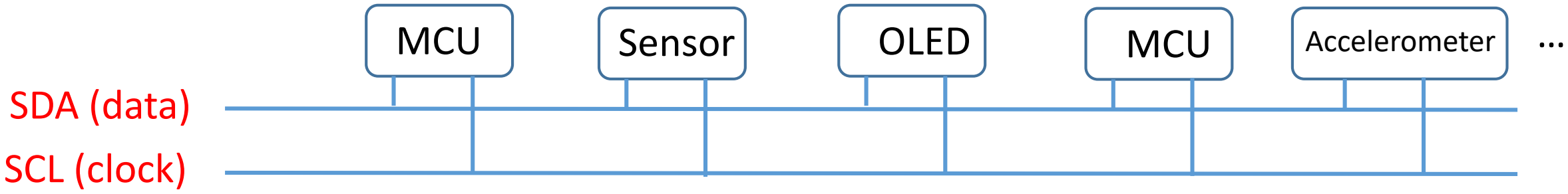
Department of Electrical and Computer Engineering

Where we’re going today

- **Overview of inter-integrated circuit (I2C) communications**
- I2C on Tiva C-series launchpad
- Example code
- Homework

I2C Basics (1)

- Inter-integrated circuit (I2C, I^2C or IIC) communications
  - Invented in 1982 by Philips Semiconductors, which has became NXP since 2006
  - Simple, robust, inexpensive and easy-to-use communications between a chain of peripheral devices and MCUs
    - Standard: 100 kbps, full-speed: 400 kbps, fast: 1Mbps, high-speed: 3.33 Mbps
    - Applications in smart phones, instruments, industrial equipment …
  - Two-wire bus structure with up to 127 devices connected in parallel



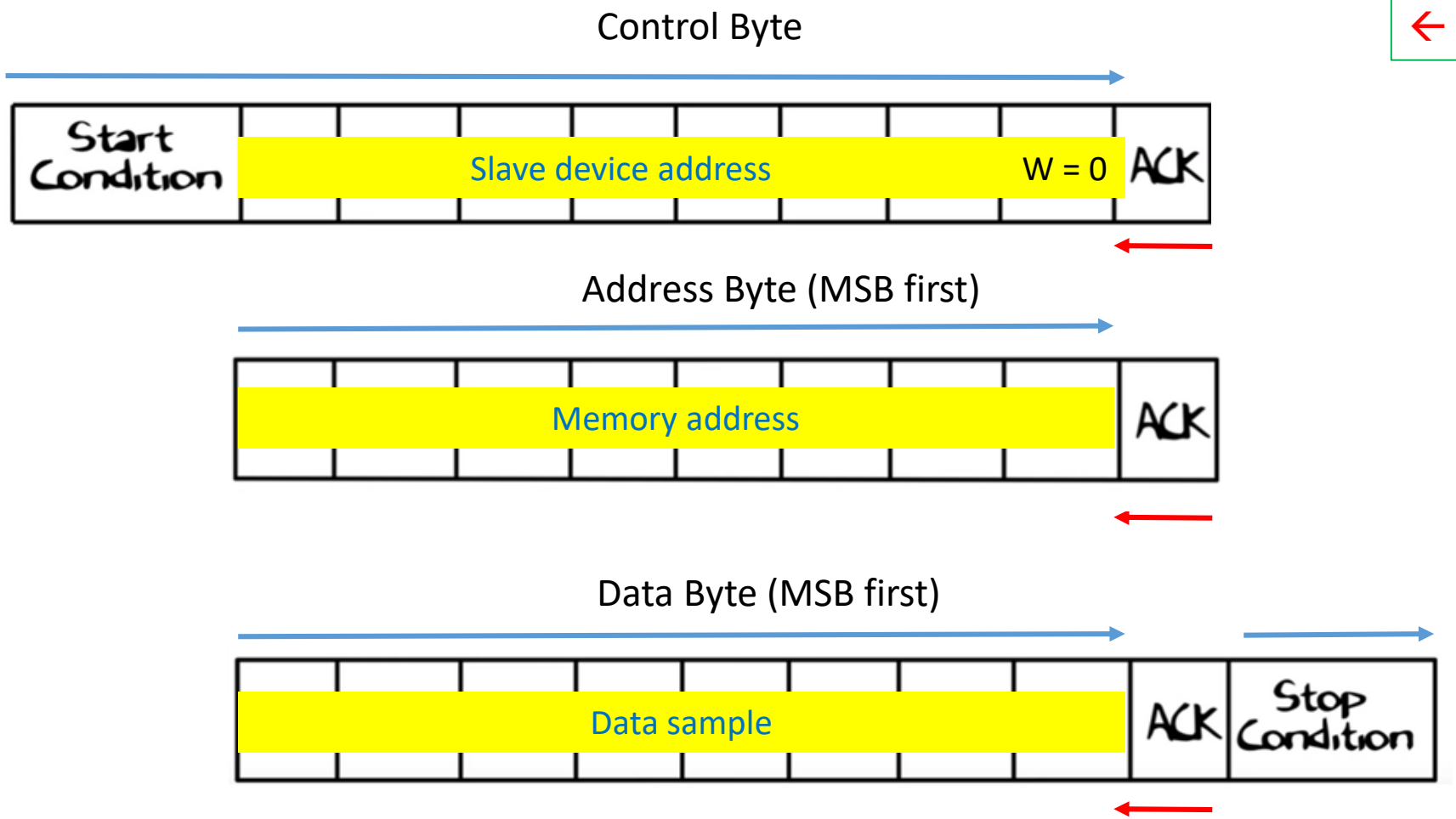
I2C Basics (2)

- Robust communications
  - Verify the correct reception of EVERY byte via ACK
  - Built-in arbitration to eliminate bus conflicts

- Easy-to-use communications
  - I2C Protocols consist of 8-bit words
  - Master-slave handshaking
    - Master generates clock signal and initiates communication
    - Address byte for selecting the device for data transmission
    - R/W bit for declaring reading or writing operation

Serial Packet Structure: Write

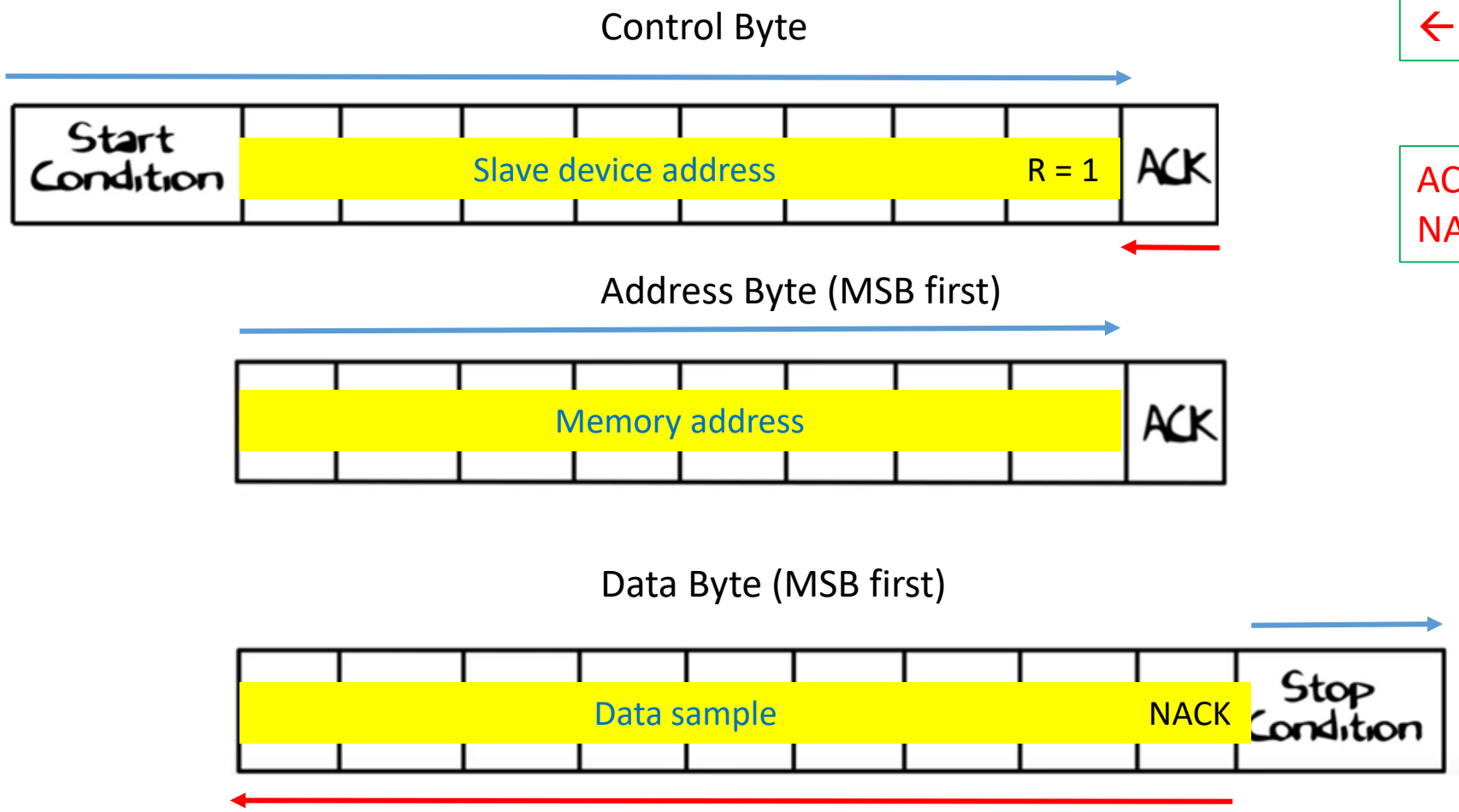
→ Master to Slave on SDA
← Slave to Master on SDA



From Easy I2C: Introduction to I2C on Youtube

Serial Packet Structure: Read

→ Master to Slave on SDA
← Slave to Master on SDA



ACK = 0
NACK = 1

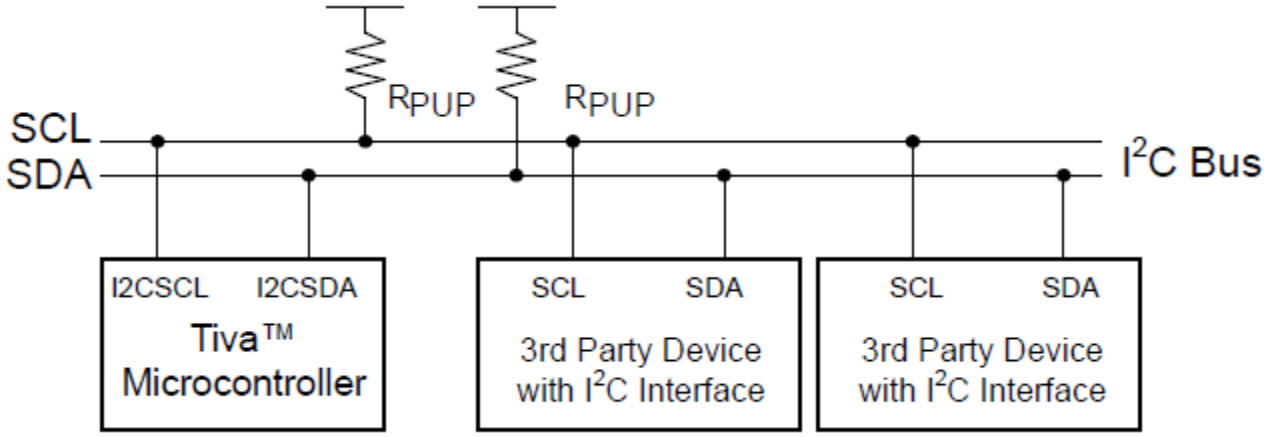
From Easy I2C: Introduction to I2C on Youtube

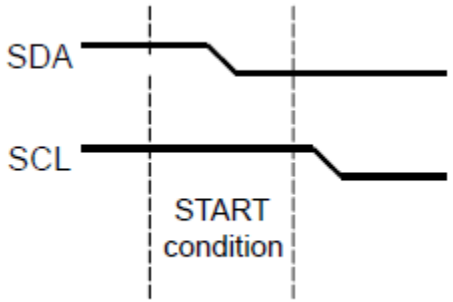
Where we're going today

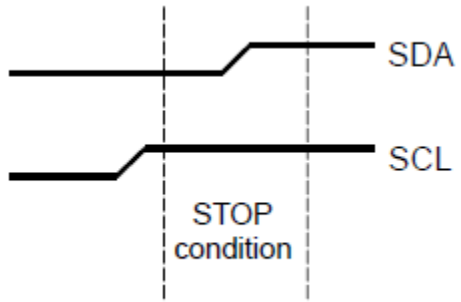
- Overview of inter-integrated circuit (I2C) communications
- **I2C on Tiva C-series launchpad**
- Example code
- Homework

I2C on Tiva C-Series Launchpad (1)

- 4 I2C modules (I2C0 – I2C3) on Tiva board
  - Half duplex
  - Can be configured as a master or slave or simultaneous master and slave
  - Master/slave interrupt generation
    - Master interrupt when a transmit or receive operation is completed or aborted
    - Slave interrupt after data transmission or START/STOP bit is detected



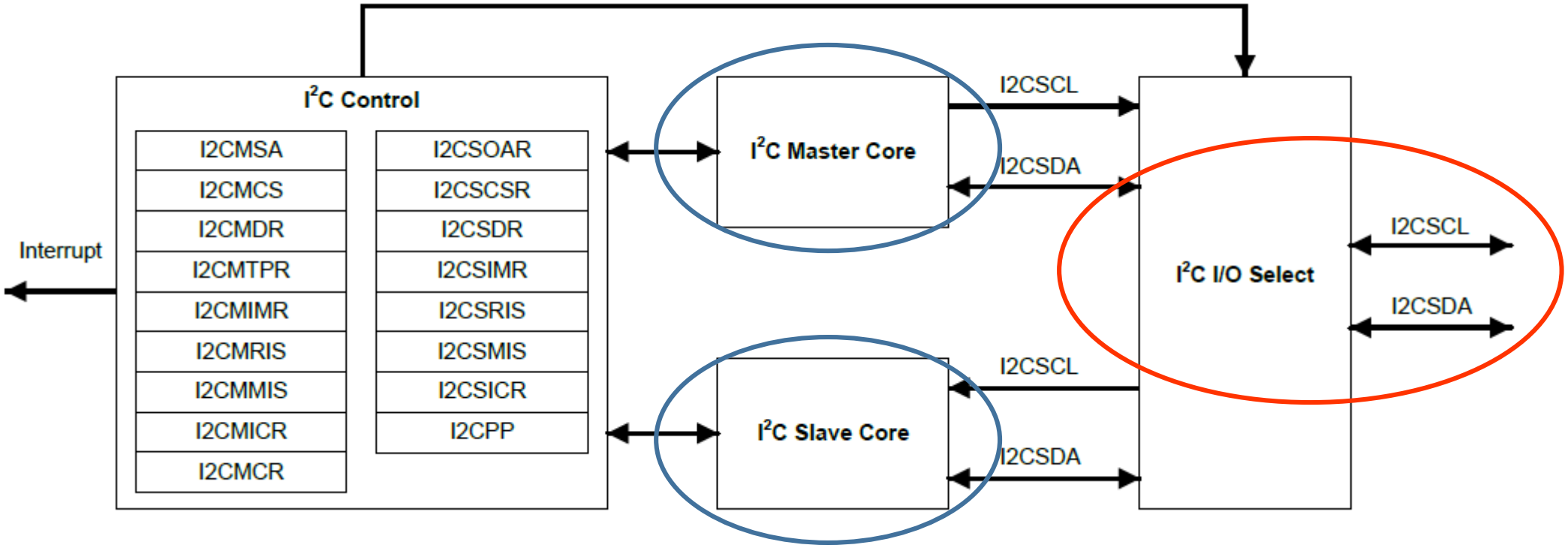




From Page 999, TM4C123GH6PM Data Sheet.pdf

I2C on Tiva C-Series Launchpad (2)

I2C Module Diagram



From Page 998, TM4C123GH6PM Data Sheet.pdf

I2C on Tiva C-Series Launchpad (3)

Default to be I2C function

Alternate functions

Pin Name	Pin Number	Pin Mux / Pin Assignment	Pin Type	Buffer Type[a]	Description
I2C0SCL	47	PB2 (3)	I/O	OD	I^2C module 0 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C0SDA	48	PB3 (3)	I/O	OD	I^2C module 0 data.
I2C1SCL	23	PA6 (3)	I/O	OD	I^2C module 1 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C1SDA	24	PA7 (3)	I/O	OD	I^2C module 1 data.
I2C2SCL	59	PE4 (3)	I/O	OD	I^2C module 2 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C2SDA	60	PE5 (3)	I/O	OD	I^2C module 2 data.
I2C3SCL	61	PD0 (3)	I/O	OD	I^2C module 3 clock. Note that this signal has an active pull-up. The corresponding port pin should not be configured as open drain.
I2C3SDA	62	PD1 (3)	I/O	OD	I^2C module 3 data.

From Page 998, TM4C123GH6PM Data Sheet.pdf

Where we're going today

- Overview of inter-integrated circuit (I2C) communications
- I2C on Tiva C-series launchpad
- **Example code**
- Homework

Example Code (1)

From i2c_driver.h

```
/* I2C Control  */
#define I2CPort          GPIO_PORTB_BASE
#define I2CSDAPort       GPIO_PORTB_BASE
#define I2CSCLPort       GPIO_PORTB_BASE

#define I2CSDA_PIN       GPIO_PIN_3
#define I2CSCL_PIN       GPIO_PIN_2

#define I2CSCL           GPIO_PB2_I2C0SCL
#define I2CSDA           GPIO_PB3_I2C0SDA

#define READ             1
#define WRITE            0

void Delay_us(void);
char I2CGenTransmit(char *pbData, int32_t cSize, bool fRW, char bAddr);
bool I2CGenIsNotIdle();
```

Pin Name	Pin Number	Pin Mux / Pin Assignment
I2C0SCL	47	PB2 (3)
I2C0SDA	48	PB3 (3)

Example Code (2)

From i2c_driver.c

```
char I2CGnTransmit(char *pbData, int32_t cSize, bool fRW, char bAddr) {

    /* Send Address High Byte */
    I2CMasterSlaveAddrSet(I2C0_BASE, bAddr, WRITE);

    I2CMasterDataPut(I2C0_BASE, *pbData);     // Push one byte to master data register

    I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);

    Delay_us();

    /* Idle wait */
    while(I2CGenIsNotIdle());

    /* Increment data pointer */
    pbData ++;
```

```
bool I2CGenIsNotIdle() {

    return !I2CMasterBusBusy(I2C0_BASE);

}
```

Example Code (3)

From i2c_driver.c

```
/* Loop data bytes */
for(i = 1; i < cSize; i++) {
    I2CMasterDataPut(I2C0_BASE, *pbData);          // Push one byte to master data register

    while(I2CMasterBusy(I2C0_BASE));               // Wait until I2C bus becomes idle
    if (i  == cSize - 1) {
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
        Delay_us();
        while(I2CMasterBusy(I2C0_BASE));
    }
    else {
          I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_CONT);
          Delay_us();
          while(I2CMasterBusy(I2C0_BASE));
          }
    pbData ++;
 }
```

Example Code (4)

From i2c_driver.c

```
I2CMasterSlaveAddrSet(I2C0_BASE, bAddr, READ);
while(I2CMasterBusy(I2C0_BASE));
for (i = 0; i < cSize; i++) {
    if (cSize == i + 1 && cSize == 1) {
       I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE);
    }
    else if (cSize == i + 1 && cSize > 1) {
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_FINISH);
    }
    else if (i == 0) {
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_START);
        Delay_us();
        while(I2CMasterBusy(I2C0_BASE));
    }
    else {
        I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_CONT);
        Delay_us();
        while(I2CMasterBusy(I2C0_BASE));
    }
   *pbData = (char)I2CMasterDataGet(I2C0_BASE);
    pbData ++;
}
```

Example Code (5)

From readAcc.c

```
void initAccl (void)
{
   .
   .
   .
   /* Enable I2C Peripheral */
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);
   SysCtlPeripheralReset(SYSCTL_PERIPH_I2C0);

   /* Set I2C GPIO pins */
   GPIOPinTypeI2C(I2CSDAPort, I2CSDA_PIN);                 // SDA, PB3
   GPIOPinTypeI2CSCL(I2CSCLPort, I2CSCL_PIN);              // SCL,  PB2
   GPIOPinConfigure(I2CSCL);
   GPIOPinConfigure(I2CSDA);

   /* Setup I2C  */
   I2CMasterInitExpClk(I2C0_BASE, SysCtlClockGet(), true);  // I2C master clock initialization and enable
                                                            // true = 400 kbps, false = 10kbps
```