I2C Read 3 Bytes

Start is High-Low on SDA when SCL High
Stop is Low-High on SDA when SCL High

I2C Write 3 Bytes

Start is High-Low on SDA when SCL High
Stop is Low-High on SDA when SCL High
### 17.4.1 UCBxCTL0, USCI_Bx Control Register 0

<table>
<thead>
<tr>
<th></th>
<th>UCA10</th>
<th>UCSLA10</th>
<th>UCMM</th>
<th>Unused</th>
<th>UCMST</th>
<th>UCMODEx=11</th>
<th>UCSYNC=1</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>r-1</td>
</tr>
</tbody>
</table>

- **UCA10** Bit 7  
  - Own addressing mode select  
  - 0: Own address is a 7-bit address  
  - 1: Own address is a 10-bit address

- **UCSLA10** Bit 6  
  - Slave addressing mode select  
  - 0: Address slave with 7-bit address  
  - 1: Address slave with 10-bit address

- **UCMM** Bit 5  
  - Multi-master environment select  
  - 0: Single master environment. There is no other master in the system. The address compare unit is disabled.  
  - 1: Multi-master environment

- **Unused** Bit 4  
  - Unused

- **UCMST** Bit 3  
  - Master mode select. When a master loses arbitration in a multi-master environment (UCMM = 1) the UCMST bit is automatically cleared and the module acts as slave.  
  - 0: Slave mode  
  - 1: Master mode

- **UCMODEx** Bits 2-1  
  - USCI Mode. The UCMODEx bits select the synchronous mode when UCSYNC = 1.  
  - 00: 3-pin SPI  
  - 01: 4-pin SPI (master/slave enabled if STE = 1)  
  - 10: 4-pin SPI (master/slave enabled if STE = 0)  
  - 11: I2C mode

- **UCSYNC** Bit 0  
  - Synchronous mode enable  
  - 0: Asynchronous mode  
  - 1: Synchronous mode
### 17.4.2 UCBxCTL1, USCI_Bx Control Register 1

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UCSSELx</strong></td>
<td><strong>Unused</strong></td>
<td><strong>UCTR</strong></td>
<td><strong>UCTXNACK</strong></td>
<td><strong>UCTXSTP</strong></td>
<td><strong>UCTXSTT</strong></td>
<td><strong>UCSWRST</strong></td>
<td></td>
</tr>
<tr>
<td><em>rw</em>-0</td>
<td><em>rw</em>-0</td>
<td><em>r0</em></td>
<td><em>rw</em>-0</td>
<td><em>rw</em>-0</td>
<td><em>rw</em>-0</td>
<td><em>rw</em>-0</td>
<td><em>rw</em>-1</td>
</tr>
</tbody>
</table>

**UCSSELx** Bits 7-6 USCI clock source select. These bits select the BRCLK source clock.

- 00: UCLKI
- 01: ACLK
- 10: SMCLK
- 11: SMCLK

**Unused** Bit 5 Unused

**UCTR** Bit 4 Transmitter/receiver

- 0: Receiver
- 1: Transmitter

**UCTXNACK** Bit 3 Transmit a NACK. UCTXNACK is automatically cleared after a NACK is transmitted.

- 0: Acknowledge normally
- 1: Generate NACK

**UCTXSTP** Bit 2 Transmit STOP condition in master mode. Ignored in slave mode. In master receiver mode the STOP condition is preceded by a NACK. UCTXSTP is automatically cleared after STOP is generated.

- 0: No STOP generated
- 1: Generate STOP

**UCTXSTT** Bit 1 Transmit START condition in master mode. Ignored in slave mode. In master receiver mode a repeated START condition is preceded by a NACK. UCTXSTT is automatically cleared after START condition and address information is transmitted. Ignored in slave mode.

- 0: Do not generate START condition
- 1: Generate START condition

**UCSWRST** Bit 0 Software reset enable

- 0: Disabled. USCI reset released for operation.
- 1: Enabled. USCI logic held in reset state.

### 17.4.3 UCBxBR0, USCI_Bx Baud Rate Control Register 0

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UCBRx</strong> - low byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
</tr>
</tbody>
</table>

### 17.4.4 UCBxBR1, USCI_Bx Baud Rate Control Register 1

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UCBRx</strong> - high byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
<td><em>rw</em></td>
</tr>
</tbody>
</table>

**UCBRx** Bit clock prescaler setting. The 16-bit value of \((UCBxBR0 + UCBxBR1 \times 256)\) forms the prescaler value.
### 17.4.5 UCBxSTAT, USCI_Bx Status Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Unused</td>
<td>Unused.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>UCSCLLOW</td>
<td>SCL low</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>UCGC</td>
<td>General call address received. UCGC is automatically cleared when a START condition is received.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>UCBBUSY</td>
<td>Bus busy</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>UCNACKIFG</td>
<td>Not-acknowledge received interrupt flag. UCNACKIFG is automatically cleared when a START condition is received.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>UCSTPIFG</td>
<td>Stop condition interrupt flag. UCSTPIFG is automatically cleared when a START condition is received.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>UCSTTIFG</td>
<td>Start condition interrupt flag. UCSTTIFG is automatically cleared if a STOP condition is received.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>UCALIFG</td>
<td>Arbitration lost interrupt flag</td>
<td></td>
</tr>
</tbody>
</table>

### 17.4.6 UCBxRXBUF, USCI_Bx Receive Buffer Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>UCRXBUFx</td>
<td>The receive-data buffer is user accessible and contains the last received character from the receive shift register. Reading UCBxRXBUF resets UCBxRXIFG.</td>
<td></td>
</tr>
</tbody>
</table>

### 17.4.7 UCBxTXBUF, USCI_Bx Transmit Buffer Register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>UCTXBUFx</td>
<td>The transmit data buffer is user accessible and holds the data waiting to be moved into the transmit shift register and transmitted. Writing to the transmit data buffer clears UCBxTXIFG.</td>
<td></td>
</tr>
</tbody>
</table>
### 17.4.8 UCBxI2COA, USCI_Bx I^2^C Own Address Register

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCGCEN</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>I2COAx</td>
</tr>
</tbody>
</table>

- **UCGCEN** Bit 15: General call response enable
  - 0: Do not respond to a general call
  - 1: Respond to a general call

- **I2COAx** Bits 9-0: I^2^C own address. The I2COAx bits contain the local address of the USCI_Bx I^2^C controller. The address is right-justified. In 7-bit addressing mode, bit 6 is the MSB, and bits 9-7 are ignored. In 10-bit addressing mode, bit 9 is the MSB.

### 17.4.9 UCBxI2CSA, USCI_Bx I^2^C Slave Address Register

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>I2CSAx</td>
</tr>
</tbody>
</table>

- **I2CSAx** Bits 9-0: I^2^C slave address. The I2CSAx bits contain the slave address of the external device to be addressed by the USCI_Bx module. It is only used in master mode. The address is right-justified. In 7-bit slave addressing mode, bit 6 is the MSB, and bits 9-7 are ignored. In 10-bit slave addressing mode, bit 9 is the MSB.

### 17.4.10 UCBxI2CIE, USCI_Bx I^2^C Interrupt Enable Register

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>UCNACKIE</td>
<td>UCSTPIE</td>
<td>UCSTTIE</td>
<td>UCALIE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Reserved** Bits 7-4: Reserved
- **UCNACKIE** Bit 3: Not-acknowledge interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled
- **UCSTPIE** Bit 2: Stop condition interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled
- **UCSTTIE** Bit 1: Start condition interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled
- **UCALIE** Bit 0: Arbitration lost interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled
### 17.4.11 IE2, Interrupt Enable Register 2

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UCB0TXIE</td>
<td>UCB0RXIE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw-0</td>
<td>rw-0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 7-4 These bits may be used by other modules (see the device-specific data sheet).

- **UCB0TXIE** Bit 3 USCI_B0 transmit interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled

- **UCB0RXIE** Bit 2 USCI_B0 receive interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled

Bits 1-0 These bits may be used by other modules (see the device-specific data sheet).

### 17.4.12 IFG2, Interrupt Flag Register 2

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UCB0TXIFG</td>
<td>UCB0RXIFG</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw-1</td>
<td>rw-0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 7-4 These bits may be used by other modules (see the device-specific data sheet).

- **UCB0TXIFG** Bit 3 USCI_B0 transmit interrupt flag. UCB0TXIFG is set when UCB0TXBUF is empty.
  - 0: No interrupt pending
  - 1: Interrupt pending

- **UCB0RXIFG** Bit 2 USCI_B0 receive interrupt flag. UCB0RXIFG is set when UCB0RXBUF has received a complete character.
  - 0: No interrupt pending
  - 1: Interrupt pending

Bits 1-0 These bits may be used by other modules (see the device-specific data sheet).

### 17.4.13 UC1IE, USCI_B1 Interrupt Enable Register

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unused</td>
<td>UCB1TXIE</td>
<td>UCB1RXIE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
<td>rw-0</td>
</tr>
</tbody>
</table>

- **Unused** Bits 7-4 Unused

- **UCB1TXIE** Bit 3 USCI_B1 transmit interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled

- **UCB1RXIE** Bit 2 USCI_B1 receive interrupt enable
  - 0: Interrupt disabled
  - 1: Interrupt enabled

Bits 1-0 These bits may be used by other USCI modules (see the device-specific data sheet).
#pragma vector=USCIAB0TX_VECTOR
__interrupt void USCI0TX_ISR(void) {

    if(IFG2&UCA0TXIFG) && (IE2&UCA0TXIE) { // USCI_A0 requested TX interrupt
        if(print_flag) {
            if(currentindex == txcount) {
                senddone = 1;
                print_flag = 0;
                IFG2 &= ~UCA0TXIFG;
            } else {
                UCA0TXBUF = printbuff[currentindex];
                currentindex++;
            }
        } else if(UART_flag) {
            if(!donesending) {
                UCA0TXBUF = txbuff[txindex];
                if(txbuff[txindex] == 255) {
                    donesending = 1;
                    txindex = 0;
                } else {
                    txindex++;
                }
            }
        }
        IFG2 &= ~UCA0TXIFG;
    }
    if((IFG2&UCB0RXIFG) && (IE2&UCB0RXIE)) { // USCI_B0 I2C RX interrupt occurs here
    } else if (((IFG2&UCB0TXIFG) && (IE2&UCB0TXIE)) { // I2C TX interrupt
    }
}

#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void) {

    if((IFG2&UCA0RXIFG) && (IE2&UCA0RXIE)) {  // USCI_A0 requested RX interrupt
        if(!started) { // Haven't started a message yet
            if(UCA0RXBUF == 253) {
                started = 1;
                newmsg = 0;
            }
        } else { // In process of receiving a message
            if(UCA0RXBUF != 255) && (msgindex < (MAX_NUM_FLOATS*5)) {
                rxbuff[msgindex] = UCA0RXBUF;
                msgindex++;
            } else { // Stop char received or too much data received
                if(UCA0RXBUF == 255) { // Message completed
                    newmsg = 1;
                    rxbuff[msgindex] = 255; // "Null"-terminate the array
                    started = 0;
                    msgindex = 0;
                }
            }
        }
        IFG2 &= ~UCA0RXIFG;
    }

    if((UCB0I2CIE&UCNACKIE) && (UCB0STAT&UCNACKIFG)) {
        UCB0STAT &= ~UCNACKIFG;
    }
    if((UCB0I2CIE&UCSTPIE) && (UCB0STAT&UCSTPIFG)) {
        UCB0STAT &= ~UCSTPIFG;
    }
    if((UCB0I2CIE&UCSTTIE) && (UCB0STAT&UCSTTIFG)) {
        UCB0STAT &= ~UCSTTIFG;
    }
    if((UCB0I2CIE&UCALIE) && (UCB0STAT&UCALIFG)) {
        UCB0STAT &= ~UCALIFG;
    }
}