

Introduction

In this module we'll take a look at the MSP430 communications modules and the protocols that can be implemented over them.

Objectives

- USART
- USCI
- USI

*** I only insert blank pages when the voices tell me to***

Module Topics

Communication	5-1
<i>Module Topics.....</i>	<i>5-3</i>
MSP430 Communication Modules.....	5-5
USI.....	5-5
Data I/O via USI.....	5-6
SPI via USI.....	5-6
USART.....	5-7
Baudrate Generator.....	5-7
USCI.....	5-8
USCI Initialization Sequence	5-8
USCI Enhanced Features.....	5-9
USCI Baudrate Generator.....	5-9
<i>Optional Lab 6 – I2C Communications</i>	<i>5-11</i>
Hardware list:	5-12
Software list:.....	5-12
<i>IAR Kickstart Procedure.....</i>	<i>5-13</i>
Set up the Hardware	5-13
Load the MSP430F2013 Software.....	5-13
Set up for the FG4618/9	5-13
Complete the I2C Master Code	5-14
Build/Load/Run/Test	5-15
Shut Down	5-15
<i>Code Composer Studio Procedure.....</i>	<i>5-17</i>
Set up the Hardware	5-17
Load the MSP430F2013 Software.....	5-17
Set up for the FG4618/9	5-18
Complete the I2C Master Code	5-18
Build/Load/Run/Test	5-19
Shut Down	5-20
<i>Review Questions.....</i>	<i>5-21</i>

*** For security reasons this page must be left blank ***

MSP430 Communication Modules

MSP430 Communication Modules			
	USART	USCI	USI
	Universal Synch/Async Receiver/Transmitter	Universal Serial Communication Interface	Universal Serial Interface
U A R T	One modulator	Two modulators; supports n/16 timings - Auto baud rate detection - IrDA encoder & decoder - Simultaneous USCI_A and USCI_B (2 channels)	---
	One SPI channel - Master and slave modes - 3 and 4 wire modes	Two SPI (one each on USCI_A and USCI_B) - Master and slave modes - 3 and 4 wire modes	- One SPI available - Master and slave modes
S P I	(on '15x/16x only) - Master and slave modes - Up to 400kbps	- Simplified interrupt usage - Master and slave modes - Up to 400kbps	- SW state machine needed - Master and slave modes
I 2 C			

USI ...

USI

USI

- ◆ **MSP430x20xx devices**
- ◆ **Variable length shift register**
- ◆ **Supports I2C**
 - ◆ START/STOP detection
 - ◆ SCL held after START
 - ◆ SCL held after counter overflow
 - ◆ Arbitration lost detection
- ◆ **Supports SPI**
 - ◆ 8/16-bit Shift Register
 - ◆ MSB/LSB first
- ◆ **Flexible Clocking**
- ◆ **Interrupt Driven**

The diagram illustrates the internal structure of the USI module. It features an 8/16-bit Shift Register connected to SDO, SDA, and SDI pins. A Bit Counter is linked to the shift register and provides USIIFG and USISTTIFG signals. A START/STOP Detect block generates USISTP. An SCL Hold block is controlled by USIIFG and USISTTIFG. A Divider/HOLD block receives clock signals (SCLK, ACLK, SMCLK, SWCLK, TA0, TA1, TA2) and provides SCLK to the SCL Hold block. The USIIFG signal is also used to control the SCL Hold block.

Data I/O ...

Data I/O via USI

USI for Data I/O

- ◆ Data shift register: up to 16 bits supported
- ◆ Number of bits transmitted and received is controlled by a bit counter
- ◆ Transmit and Receive is simultaneous
- ◆ Data I/O is user-defined: MSB or LSB first
- ◆ Bit counter automatically stops clocking after last bit & sets flag
- ◆ No data buffering needed

SPI Implementation ...

SPI via USI

USI Reduces CPU Load for SPI

MSP430

SCLK

SDO

SDIN

Peripheral

```

//Shift16_inout_Software
SR = DATA;
for (CNT=0x10;CNT>0;CNT--)
{
  P2OUT &= ~SDO;
  if (SR & 0x8000)
    P2OUT |= SDO;
  SR = SR << 1;
  if (P2IN & SDIN)
    SR |= 0x01;
  P2OUT |= SCLK;
  P2OUT &= ~SCLK;
}
                
```

425 Cycles

```

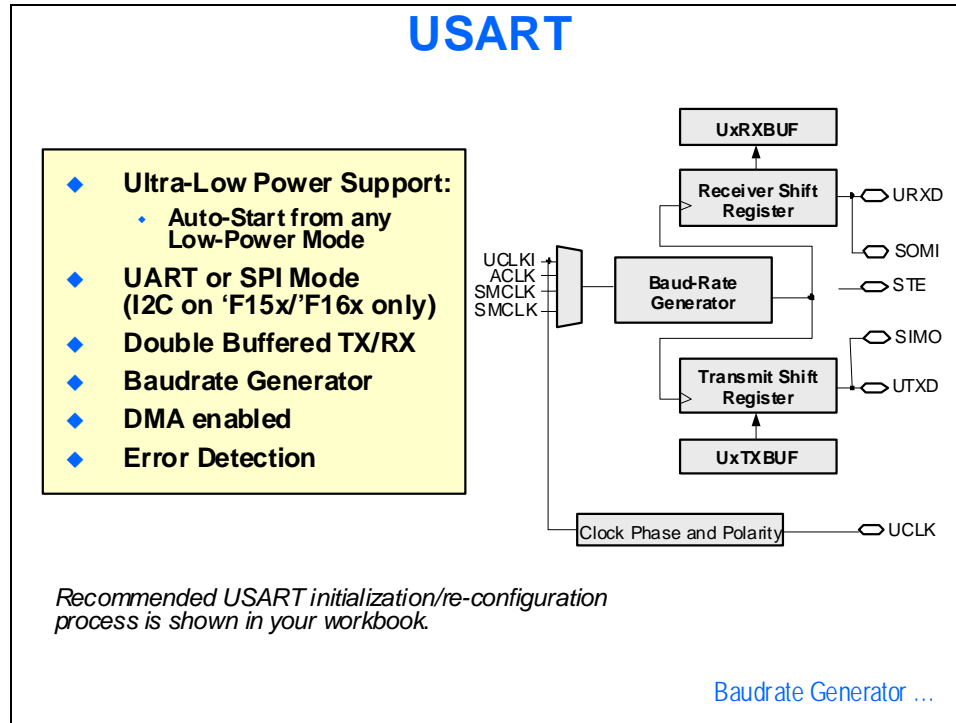
// Shift16_inout_USI
USISR |= DATA;
USICNT |= 0x10;
                
```

10 Cycles

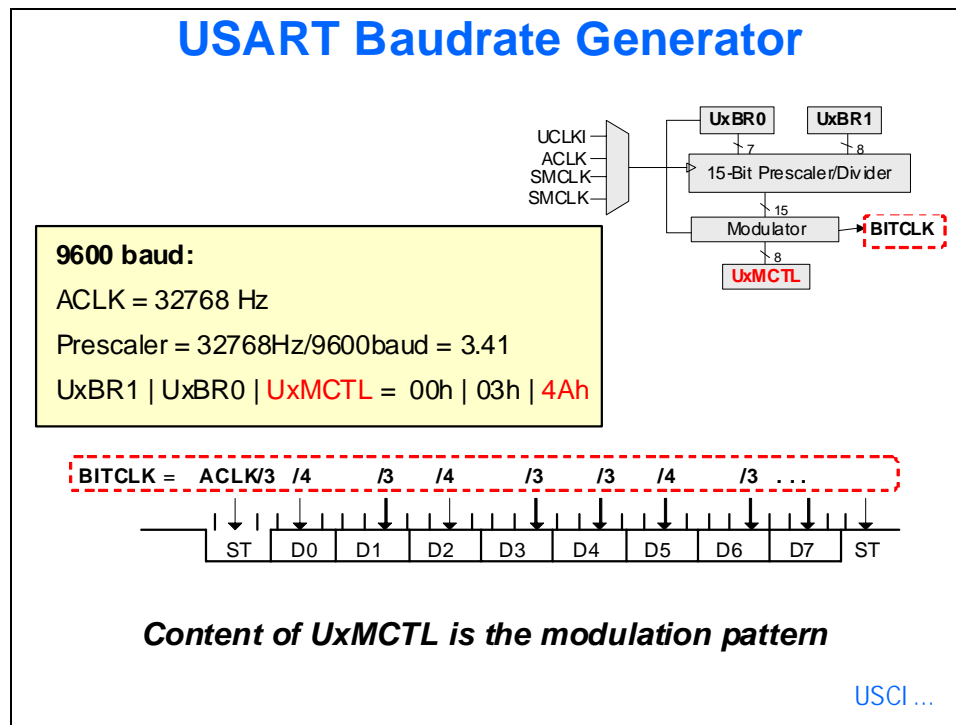
- ◆ I2C Slave has as little as 4us from clock edge to data
- ◆ Traditional software-only solution allows time for little else
- ◆ USI hardware enables practical and compliant I2C
- ◆ Code on MSP430 website

USART ...

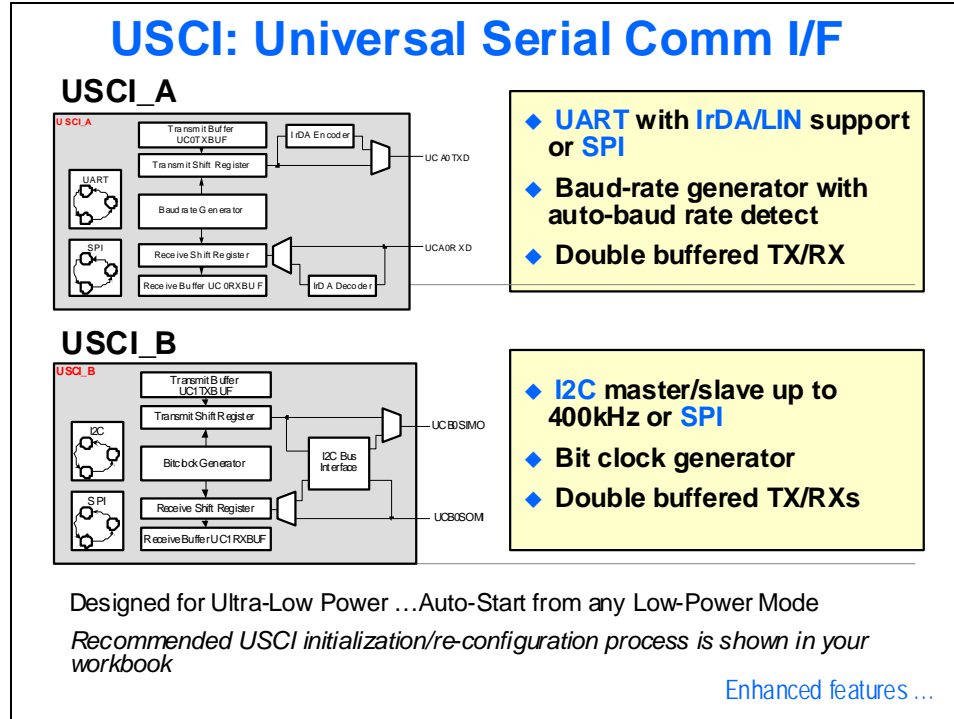
USART



Baudrate Generator



USCI



USCI Initialization Sequence

Note: Initializing or Re-Configuring the USCI Module

The recommended USCI initialization/re-configuration process is:

- 1) Set UCSWRST (BIS.B #UCSWRST, &UCAxCTL1)
- 2) Initialize all USCI registers with UCSWRST = 1 (including UCAxCTL1)
- 3) Configure ports.
- 4) Clear UCSWRST via software (BIC.B #UCSWRST, &UCAxCTL1)
- 5) Enable interrupts (optional) via UCAxRXIE and/or UCAxTXIE

USCI Enhanced Features

USCI Enhanced Features

- ◆ New standard MSP430 serial interface
- ◆ Auto clock start from any LPMx
- ◆ Two independent communication blocks
- ◆ Asynchronous communication modes
 - UART standard and multiprocessor protocols
 - UART with automatic Baud rate detection (LIN support)
 - Two modulators support n/16 bit timing
 - IrDA bit shaping encoder and decoder
- ◆ Synchronous communication modes
 - SPI (Master & Slave modes, 3 & 4 wire)
 - I2C (Master & Slave modes)

Baudrate Generator ...

USCI Baudrate Generator

USCI Baudrate Generator

- ◆ Oversampling Baud Rate Generation
- ◆ Two Modulators:
 - UCBR_{Sx} and UCBR_{Fx} select modulation pattern
- ◆ RX sampled using BITCLK16

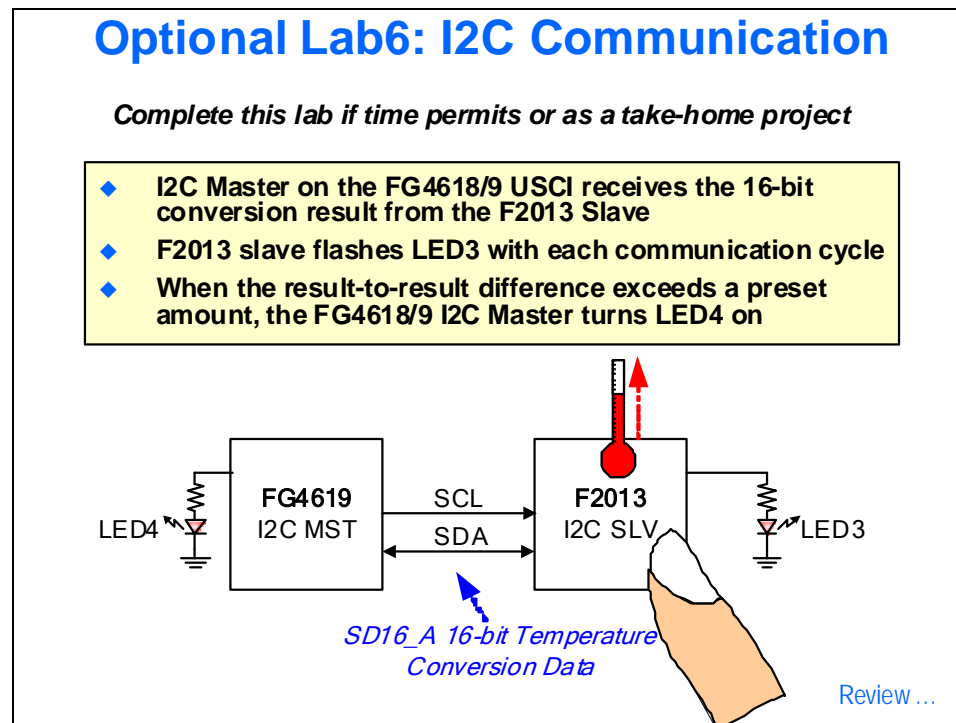
Optional Lab6 ...

*** War and Peace started this way ***

Optional Lab 6 – I2C Communications

This lab should be attempted if time permits during the class or as a take-home project for the student.

The MSP430F2013 is used to measure the temperature. It then transmits the result to the MSP430FG4618/9 via the I2C connection on the USCI port. The MSP430FG4618/9 will then determine if a preset difference has been reached, at which point it will light LED4. The MSP430F2013 will also flash LED3 each communication cycle. In this I2C implementation, the MSP430F2013 will be the slave and the MSP430FG4618/9 will be the master.



Hardware list:

- WinXP PC
- MSP-FET430UIF
- USB cable
- JTAG ribbon cable
- MSP430FG461x/F28xx Experimenter's Board
- Jumpers

Software list:

- IAR Kickstart for MSP430 version 4.21B
- Code Composer Studio 4.1
- Labs
- Additional pdf documentation
- Adobe™ Reader

IAR Kickstart Procedure

In this lab, you will complete an I2C data link between the two MSP430s on the Experimenter's Board. Our tasks will be to:

- Load ready-to-use USI I2C slave code on the MSP430F2013 (slave address = 0x48)
- Complete partial MSP430FG4618/9 USCI_B I2C Master Receiver code

Set up the Hardware

1. JTAG

The first thing we're going to do is to load the ready-to-use I2C slave code into the **MSP430F2013**. **Remove** the JTAG ribbon cable from the **MSP430FG4618/9** debug port and place it in the **MSP430F2013** debug port.

Load the MSP430F2013 Software

Note: Please do not load the MSP430F2013 code into the MSP430FG4618/9!

2. Load the I2C Software into the MSP430F2013

Open *IAR Kickstart*, create a new workspace and project called **Lab6** in the **IAR Labs\Lab6** folder. Don't forget to set the project options with the target device being the **MSP430F2013**.

Add **Lab6_2013_solution.c** to the project and **build/load** it to the **MSP430F2013**. Feel free to open the code in the editor and take a look at it.

Click the **Go** button to start the MSP430F2013 I2C slave code running. You'll probably have no visual indication that the code is running. **Exit** the debugger by clicking the **Stop Debugging** button. Now you should be looking at the editor window in *IAR Kickstart*.

Set up for the FG4618/9

3. Set up for the MSP430FG4618/9 I2C Master Code

Swap the JTAG connector to the MSP430FG4618/9 debug port.

Close the **Lab6_2013_solution.c** code in the editor window (if you still have it open). **Right-click** on the file in the Workspace window and select **Remove**, then click **Yes**.

Add **Lab6_4618_exercise.c** to the project. **Change** the project option target device to the **MSP430FG4618** or **MSP430FG4619**.

Open the source file in the editor and feel free to look around in it.

Complete the I2C Master Code

Let's fill in the blanks one at the time in the code extract below. Lazy folks can reference the solutions ...

```

P3SEL |= 0x06;                // Assign I2C pins to USCI_B0
UCB0CTL1 |= _____;        // Enable SW reset (why?)
UCB0CTL0 = _____;        // I2C Master, synchronous mode
UCB0CTL1 = _____;        // Use SMCLK, keep UCSWRST set
UCB0BR0 = 11;                // fSCL = SMCLK/11 = 95.3kHz
UCB0BR1 = 0;
UCB0I2CSA = 0x48;            // Set slave address
UCB0CTL1 &= ~_____;        // Clear SW reset, resume operation
UCB0I2CIE |= UCNACKIE;      // Interrupt on slave Nack
IE2 |= UCB0RXIE;            // Enable RX interrupt

```

4. **UCB0CTL1 |= _____;**

It's pretty easy to find the USCI software reset in the UCB0CTL1 section of the header file. Why do you think the USCI should be in reset while you're programming its bits? Gee, that's a tough one ...

5. **UCB0CTL0 = _____;**

I2C Master

Look for the master mode select in the UCB0CTL0 section.

Synchronous mode

A quick look at the Initialization and Reset chapter of the USCI/I2C section will tell you that the UCMODEx bits must be set properly to be in I2C mode. In addition, you must select the synchronous mode.

6. **UCB0CTL1 = _____;**

Clock source

You must select the appropriate USCI clock source to use SMCLK. In this case, that's source 2. Verify that in the User's Guide.

Keep UCSWRST set

Make sure the USCI software reset stays set,

7. UCB0CTL1 &= ~_____

This one's easy. Now that everything is all set up, you can clear the USCI software reset.

Build/Load/Run/Test

8. Build/Load/Run

Compile the code, download it to the MSP430FG4618/9 and run it. LED3 (next to the MSP430F2013 debug port) should be blinking about once every 2 seconds.

9. Test the Code

With the code running, place your fingertip on the MSP430F2013 device (the little one next to the MSP430F2013 debug port). After a few seconds, LED4 (underneath the LCD display) should light. Remove your finger and the LED will quickly go off. Look around in the MSP430FG4618/9 code to see what the threshold is to light the LED. Change it if you like.

Shut Down

10. Shut down

Shut down *IAR Kickstart*. Disconnect the JTAG debug interface from both the Experimenter's Board and the PC.



IAR Users ... you're done. Proceed to the review questions on page 5-21.

*** It's about time for a refreshment, I think ***

Code Composer Studio 4.1 Procedure

In this lab, you will complete an I2C data link between the two MSP430s on the Experimenter's Board. Our tasks will be to:

- Load ready-to-use USI I2C slave code on the MSP430F2013 (slave address = 0x48)
- Complete partial MSP430FG4618/9 USCI_B I2C Master Receiver code

Set up the Hardware

1. JTAG

The first thing we're going to do is to load the ready-to-use I2C slave code into the **MSP430F2013**. Remove the JTAG ribbon cable from the **MSP430FG4618/9** debug port and place it in the **MSP430F2013** debug port.

Load the MSP430F2013 Software

Note: Please do not load the MSP430F2013 code into the MSP430FG4618/9!

2. Load the I2C Software into the MSP430F2013

Open *CCS*, create a new workspace in the **CCS Labs\Lab6** folder. Create a new project in that workspace folder called **Lab6_2013**. Don't forget to set the project options with the target device being the **MSP430F2013**.

Add **Lab6_2013_solution.c** to the project and **build/load** it to the **MSP430F2013**. Feel free to open the code in the editor and take a look at it.

Click the **Run** button to start the MSP430F2013 I2C slave code running. You'll probably have no visual indication that the code is running. **Exit** the debugger by clicking the **Terminate All** button. Now you should be looking at the editor window in *Code Composer Studio*.

Set up for the FG4618/9

3. Set up for the MSP430FG4618/9 I2C Master Code

Swap the JTAG connector to the MSP430FG4618/9 debug port.

Close the `Lab6_2013_solution.c` code in the editor window (if you still have it open). We could delete the source file from the project, but ...

CAUTION: Eclipse (the editor used here) actually deletes the source file from the workspace folder. In our case, that's not an issue. When we added our source file, Eclipse made a copy of our source file in the workspace folder. But if you store your original source files in the workspace folder, they will be deleted in this process. Consider yourself warned.

Instead, let's do something a little more interesting. Create a new project in this workspace called **Lab6_4618** (I know, that's not a very imaginative name). Don't forget to set the project options with the target device being the **MSP430FG4618 (or 19)**. Check this out ... now our workspace has two projects in it. Imagine the possibilities.

Lab6_4618 is now the **Active Project** (notice the project pane). Add `Lab6_4618_exercise.c` to the project. We can easily switch between projects by right-clicking on the project and selecting *Set as Active Project*. But, leave *Lab6_4618* as the active project now.

Open the `Lab6_4618_exercise.c` source file in the editor and feel free to look around in it.

Complete the I2C Master Code

Let's fill in the blanks one at the time in the code extract below. Lazy folks can reference the solutions ...

```

P3SEL |= 0x06;           // Assign I2C pins to USCI_B0
UCB0CTL1 |= _____; // Enable SW reset (why?)
UCB0CTL0 = _____; // I2C Master, synchronous mode
UCB0CTL1 = _____; // Use SMCLK, keep UCSWRST set
UCB0BR0 = 11;           // fSCL = SMCLK/11 = 95.3kHz
UCB0BR1 = 0;
UCB0I2CSA = 0x48;       // Set slave address
UCB0CTL1 &= ~_____;     // Clear SW reset, resume operation
UCB0I2CIE |= UCNACKIE; // Interrupt on slave Nack
IE2 |= UCB0RXIE;        // Enable RX interrupt

```

4. UCB0CTL1 |= _____;

It's pretty easy to find the USCI software reset in the UCB0CTL1 section of the header file. Why do you think the USCI should be in reset while you're programming its bits? Gee, that's a tough one ...

5. UCB0CTL0 = _____;**I2C Master**

Look for the master mode select in the UCB0CTL0 section.

Synchronous mode

A quick look at the Initialization and Reset chapter of the USCI/I2C section will tell you that the UCMODEx bits must be set properly to be in I2C mode. In addition, you must select the synchronous mode.

6. UCB0CTL1 = _____;**Clock source**

You must select the appropriate USCI clock source to use SMCLK. In this case, that's source 2. Verify that in the User's Guide.

Keep UCSWRST set

Make sure the USCI software reset stays set.

7. UCB0CTL1 &= ~_____

This one's easy. Now that everything is all set up, you can clear the USCI software reset.

Build/Load/Run/Test**8. Build/Load/Run**

Compile the code, download it to the MSP430FG4618/9 and run it. LED3 (next to the MSP430F2013 debug port) should be blinking about once every 2 seconds.

9. Test the Code

With the code running, place your fingertip on the MSP430F2013 device (the little one next to the MSP430F2013 debug port). After a few seconds, LED4 (underneath the LCD display) should light. Remove your finger and the LED will quickly go off. Look around in the MSP430FG4618/9 code to see what the threshold is to light the LED. Change it if you like.

Shut Down

10. Shut down

Shut down *Code Composer Studio*. Disconnect the JTAG debug interface from both the Experimenter's Board and the PC.



CCS Users ... you're done.

Review Questions

Review

- ◆ The new, standard MSP430 serial comm. module is:
- ◆ Implementing SPI on the USI or USCI provides a _____ and _____ solution.
- ◆ The best place to look for code examples is:
- ◆ The best place to find technical documentation is:

You can find the answers to these questions in the Addendum section at the end of this workbook.

*** Relax, it's almost over ***