FTF | FREESCALE TECHNOLOGY FORUM POWERING INNOVATION

USB Debugging and Development

FTF-ENT-F0085

Tabitha Miller (Total Phase)
Rudan Bettelheim (Freescale)

Updated May 2012

Abstract

Your new MCU project now includes USB to connect to several devices and peripherals, and it looks a lot more complex than the SPI and UARTs you are used to. Come to this session to learn how to implement, analyze and debug USB systems on Freescale MCUs using the Beagle Protocol Analyzer.

Agenda

- Introduction to the Freescale MCUs, Kinetis, and Tower System
- Introduction to Total Phase Development Tools
- Review of I2C, SPI, and USB Protocols
- Lab
  - Lab Introduction
  - Programming the EEPROM
  - Bug 1: Enumeration of the CDC Device
  - Bug 2: SPI Programming
  - Feature: I2C LEDs
- Q&A

TOTAL PHASE

Facebook.com/Freescale
Tag yourself in photos
and upload your own!

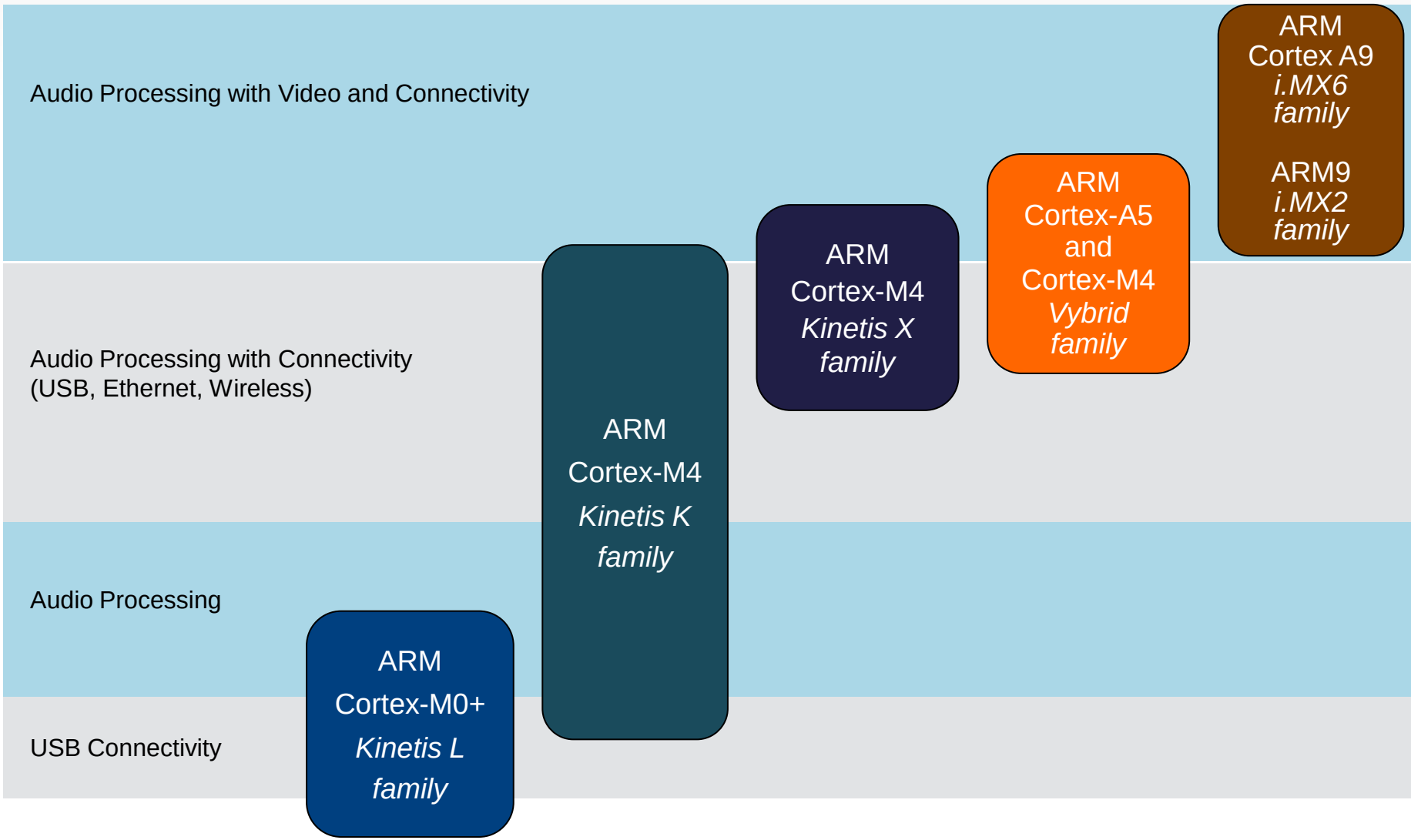
Tweeting?
Please use hashtag
#FTF2012

Session materials will be posted @ www.freescale.com/FTF
Look for announcements in the FTF Group on LinkedIn or follow Freescale on Twitter

freescale™

Freescale MCUs, Kinetis, and the Tower System

Accessories and Digital Audio



Kinetis: Freescale Enablement Bundle

Freescale Tower System

Kinetis MCU modules from $69

- Modular, expandable, open-source h/ware development platform for 8/16/32-bit MCUs/MPUs
- Rapid evaluation and prototyping with maximum h/ware reuse
- Supported by a growing range of peripheral plug-in boards (WiFi, Sensing, Graphics LCD, Audio,...)
- www.freescale.com/tower

Open source, reusable hardware platform

Freescale CodeWarrior IDE

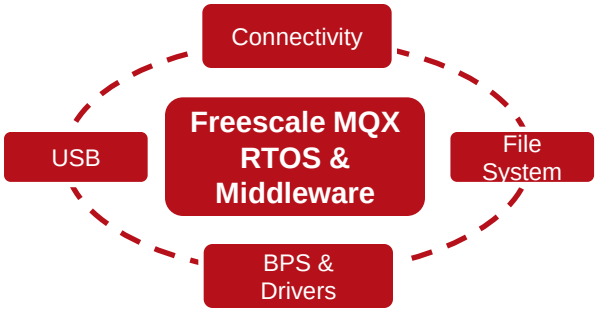
Free of charge up to 128KB

- Eclipse environment
- Includes **Processor Expert code generation wizard**
- Creates MQX-aware drivers
- Build, debug and flash tools
- Software analysis
- Kernel-aware debug
- Special Edition $0 up to 128KB
- www.freescale.com/codewarrior

Powerful IDE with code generation wizard for $0!

Freescale MQX RTOS

Free of charge ($95K est. value)



- Full-featured, scalable, proven RTOS with TCP/IP, USB, Graphics, Security and File Systems plug-ins
- Makes application code more stable, more maintainable and easier to upgrade – reduces time-to-market!
- Compatible with CodeWarrior, IAR, Keil & Green Hills IDEs
- www.freescale.com/mqx

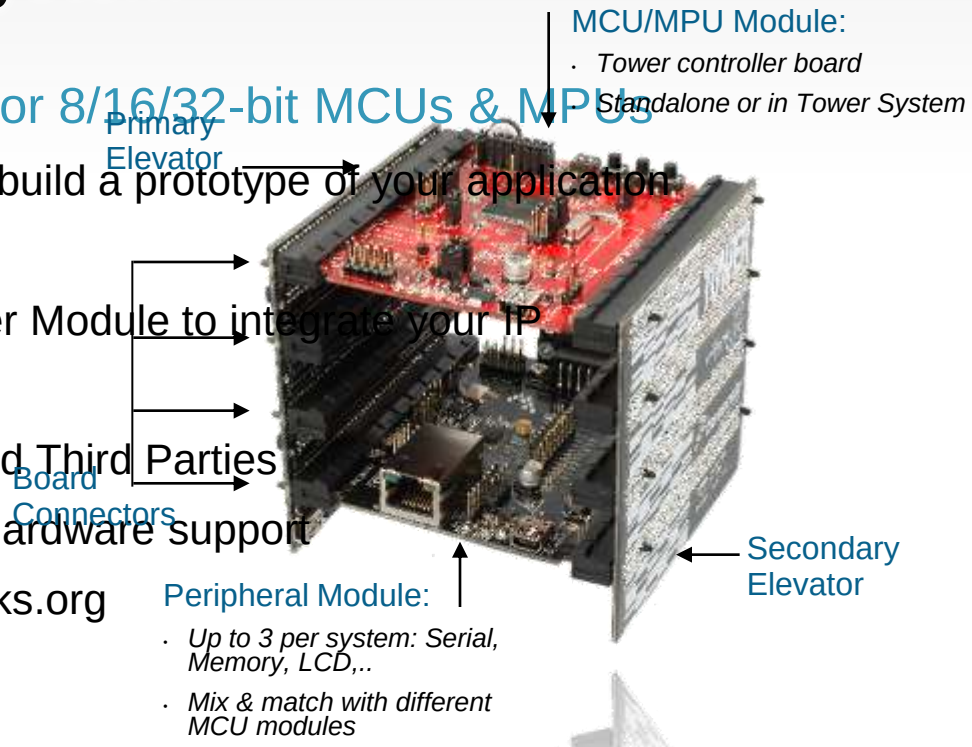
Bundled RTOS for $0!

One Stop Shop for Silicon, IDE & RTOS

The Freescale Tower System

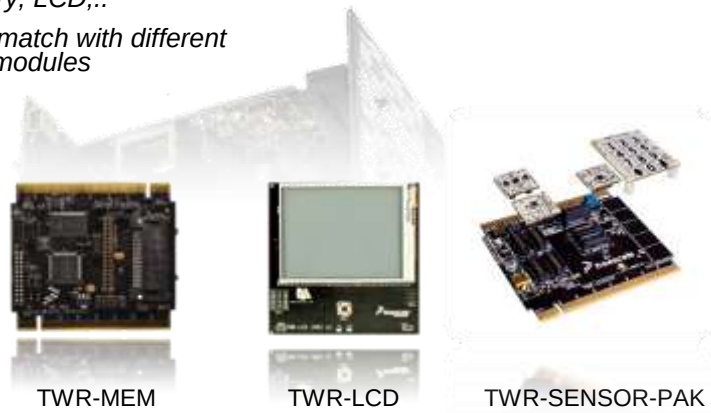
A modular development platform for 8/16/32-bit MCUs & MPUs

- Quickly combine Tower Modules to build a prototype of your application
- Modules sold individually or in kits
- Open Source: Build your own Tower Module to integrate your IP
- Cost-optimized hardware
- Software support from Freescale and Third Parties
- Growing community of Third Party hardware support
- On-line community: www.towergeeks.org



Rapidly build a prototype of your end application

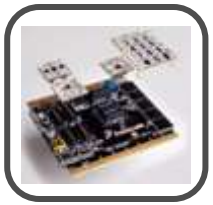
Support for all ColdFire+ and Kinetis MCUs!



netis Tower System: Reusable, modular development platform

www.freescale.com/tower www.towergeeks/org

MCU Families Supported	TWR Part Number	Contents	Price (SRP)
K30/40	**TWR-K40X256**	TWR-K40X256 (144MGA), TWRPI-SLCD	**$69**
	TWR-K40X256-KIT	TWR-K40X256 (144MBGA), TWRPI-SLCD TWR-SER, TWR-ELEV	**$139**
K50	**TWR-K53N512**	TWR-K53N512 (144MBGA), TWRPI-SLCD	**$109**
	TWR-K53N512-KIT	TWR-K53N512 (144MBGA), TWRPI-SLCD, TWR-SER, TWR-ELEV	**$179**
K10/20/60	**TWR-K60N512**	TWR-K60N512 (144MBGA)	**$69**
	TWR-K60N512-KIT	TWR-K60N512 (144MBGA), TWR-SER, TWR-ELEV	**$139**
	TWR-K60N512-IAR IAR SYSTEMS	TWR-K60N512-KIT (144MBGA), TWR-PROTO, Segger J-Link Lite Debug Probe, IAR EWARM IDE (eval. version)	**$239**
	TWR-K60N512-KEIL KEIL Tools by ARM	TWR-K60N512-KIT (144MBGA), UNLINK-ME Debug Probe, KEIL MDK IDE (eval. version)	**$199**



TWR-SENSOR-PAK

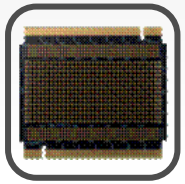
TWR-LCD

TWR-WIFI-RS2101

- IDEs: FSL CodeWarrior, IAR Embedded Workbench, Keil MDK, ...
- Freescale MQX RTOS
- OSJTAG Debug circuitry – program & debug with USB cable
- Low power touch sensing & plug-in socket for expansion: Sensors, Radio, etc…
- Fully compatible with all Tower peripheral modules

freescale™

Example Tower System Configurations


TWR-S08MM128 + 
TWR-PROTO + 
TWR-SER + 

TWR-ELEV = Medical Prototyping Solution


TWR-56F8257 + 
TWR-MC-LV3PH + 
TWR-SER + 

TWR-ELEV = Motor Control Solution


TWR-MCF5225X + 
TWR-SENSOR-PAK + 
TWR-SER + 

TWR-ELEV = Sensors Solution


TWR-K40X256 + 
TWR-LCD + 
TWR-AUDIO + 

TWR-ELEV = Multimedia Solution


TWRK60N512 + 
TWR-WIFI + 
TWR-MEM + 

TWR-ELEV = Wi-Fi Solution

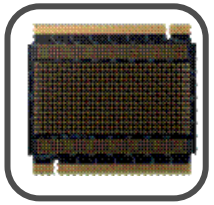
freescale™

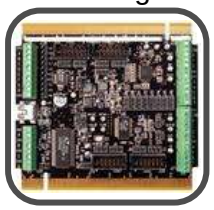
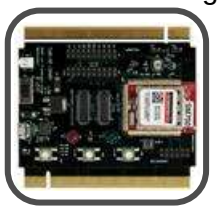
Available Tower System Modules www.freescale.com/tower

Processor Modules ($39-$119)

8bit	16bit	DSC	32bit - ColdFire	32bit – Power Arch	32bit Kinetis
					
TWR-S08LL64 TWR-S08LH64 TWR-S08JE128 TWR-S08MM128 TWR-S08GW64 TWR-S08UNIV	TWR-S12GN32 TWR-S12G128	TWR-56F8257	TWR-MCF51JE TWR-MCF51CN TWR-MCF51MM TWR-MCF51QM TWR-MCF5225X TWR-MCF5441X	TWR-MPC5125	TWR-K60N512 TWR-K40X256 TWR-K60N512-IAR TWR-K60N512-KEIL TWR-K53N512 KWIKSTIK-K40

Peripheral Modules ($15 – $149)

Serial	Prototyping	Wi-Fi	Memory	Sensors & Plug-Ins	Displays	Medical
						
TWR-SER TWR-SER2	TWR-PROTO	TWR-WIFI-RS2101 TWR-WIFI-G1011MI TWR-WIFI-AR4100	TWR-MEM	TWR-SENSOR-PAK TWR-SENSOR-PAK-AUTO TWRPI-MMA6900 TWRPI-MPL115A	TWR-LCD	MED-EKG

Analog	Audio	Mesh Networking	*MFi*
			
TWR-ADCDAC-LTC	TWR-AUDIO-SGTL	TWR-RF-SNAP	*TWR-DOCK*

Kinetis Tower System

- TWR-K60N512 Module
  - Features include:
    - 3-axis accelerometer
    - Capacitive touch pads
    - Tower connectivity to USB, CAN, SPI, and I2C
    - Part of the Freescale Tower system

Controller Module: TWR-K60D100M

K60 100MHz Tower Board

TWR-K60D100M

MSRP: $69
Launch: Jan 2012
Status: In Design

Hardware:

- K60DN512VMD10 MCU
- New Low Power Modes Enabled
- OSJTAG
- SD Card Socket
- Potentiometer
- Accelerometer
- Touch Sense Buttons
- Infrared
- TWRPI Socket

Software:

- MQX RTOS
- Code Warrior IDE
- IAR IDE

Related Modules:

- TWR-SER

Controller Module: TWR-K40D100M

K40 100MHz Tower Board

TWR-K40D100M

MSRP: $69
Launch: Feb 2012
Status: In Design

Hardware:

- K40DX256VMD10 MCU
- New Low Power Modes Enabled.
- OSJTAG
- SD Card Socket
- Potentiometer
- Accelerometer
- Touch Sense Buttons
- Infrared
- TWRPI Socket

Software:

- MQX RTOS
- Code Warrior IDE
- IAR IDE

Related Modules:

- TWR-SER
- TWRPI-SLCD

Controller | Peripheral | Plug-In

Tower Plug In: TWRPI-MMA845XQ

3-axis accelerometer

TWRPI-MMA845XQ

MSRP: $25
Launched: Sept 14, 2011

Hardware:

- MMA8451Q I2C 3-axis accelerometer
- Smart low-power, three-axis, capacitive micromachined accelerometer with 14 bits of resolution
- Programmable interrupts
- Freefall/Motion, Pulse, & Jolt Detection
- Compatible with the Freescale Tower System modules that utilize swappable (Tower) plug-ins (TWRPIs), including:
  - QWIKSTIK
  - TWR-SENSOR-PAK
  - TWR-K60N512
  - TWR-K40X256

Partner:

- If applicable, enter Partner name and website

Freescale's MFi Solution

Freescale's MFi solutions are based on the TWR-DOCK peripheral module

- TWR-DOCK supports development and rapid prototyping of electronic accessories for iPod, iPhone and iPad devices.
  - Access to the 30-pin connection
  - Analog audio and video signals with standard RCA and S-Video connectors
  - Digital audio streaming in both directions over USB
  - Control and communication with various devices
- TWR-DOCK concentrates all MFi controlled items on one Tower module, without including any processors or other Freescale products
- TWR-DOCK may be used with a wide range of Tower System MCU/MPU, peripheral, sensor and communication modules
- Kinetis-based demos are available

Total Phase Development Tools

Total Phase Solutions

- By using debugging tools manufactured by Total Phase you can:
  - Debug in real time
  - Quickly evaluate embedded systems
  - Program EEPROMs and flash memories
  - Easily collaborate with colleagues
  - Maximize productivity

Aardvark I2C/SPI Host Adapter

- General purpose I2C/SPI master or slave
  - Active communication on the SPI bus up to 8 MHz as an SPI master, 4 MHz as an SPI slave
  - Active communication on the I2C bus up to 800 kHz

Aardvark I2C/SPI
Host Adapter

Flash Center Software

- Designed to work with the host adapter line
  - Quickly program I2C or SPI EEPROMs or flash memories
  - In-system and stand-alone programming
  - Built-in XML part library
  - Gang programming support
  - Windows, Linux, Mac OS X

Control Center Software

- Designed to work with the Aardvark I2C/SPI Host Adapter
  - Read and write I2C/SPI messages
  - XML Batch Script support
  - Built-in Help System
  - Multiple adapter support
  - Windows, Linux, Mac OS X

Beagle I2C/SPI Protocol Analyzer

- Non-intrusively monitor an I2C or SPI bus
  - Interactive real-time display, filter, and search
  - Monitors I2C data up to 4 MHz
  - Monitors SPI data up to 24 MHz

Beagle I2C/SPI
Protocol Analyzer

Beagle USB 480 Protocol Analyzer

- Non-intrusively monitor high-, full-, low-speed USB 2.0
  - Interactive real-time display, filter, and search
  - Real-time class-level decoding
  - 64 MB on-board hardware buffer

Beagle I2C/SPI Protocol Analyzer

Data Center Software

- Designed to work with the Beagle Protocol Analyzers
  - LiveDisplay
  - LiveFilter
  - LiveSearch
  - 32-bit and 64-bit support
  - Tree View and Block view
  - Windows, Linux, Mac OS X

Komodo CAN Duo Interface

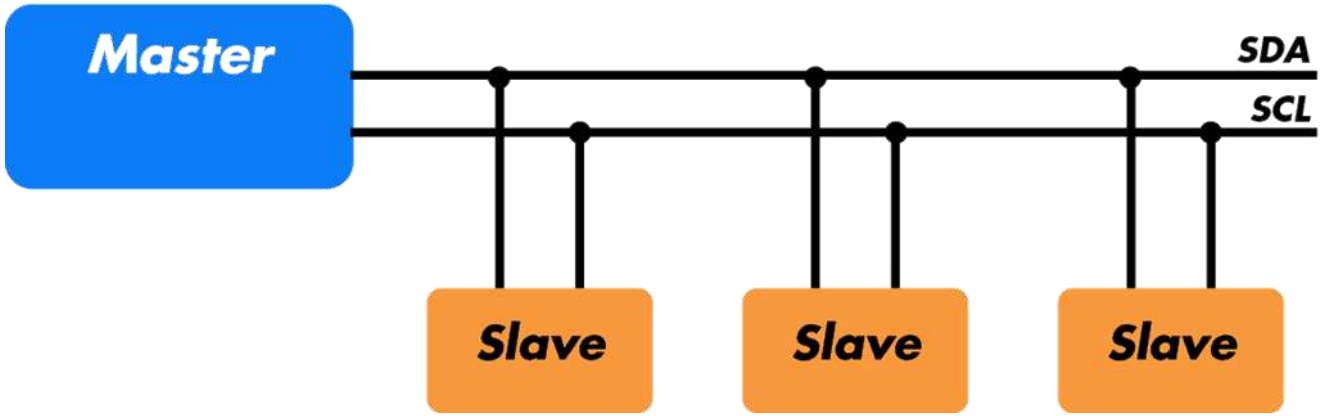
- Two channel USB-to-CAN adapter and analyzer
  - Actively transmit CAN data up to 1 Mbps
  - Non-intrusive CAN bus monitoring
  - Interactive real-time display, filter, and search

Komodo CAN Duo Interface

I2C Overview

I2C Overview

- Inter-IC Bus
  - A low bandwidth, short distance protocol for on-board communications
  - All devices are connected through two wires:
    - Serial Data (SDA)
    - Serial Clock (SCL)
  - All devices on a bus must have a unique address



I2C Theory of Operation

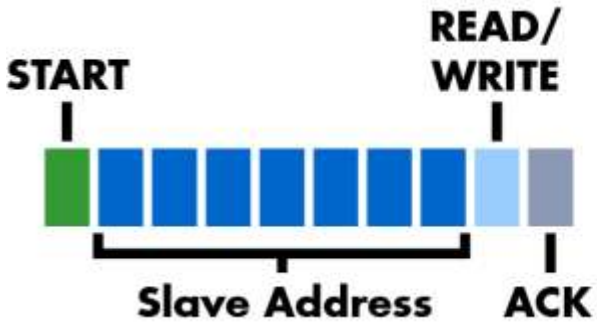
- Master device issues start condition informing slave devices to listen on the serial data line for their address

I2C Theory of Operation

- Master device issues start condition informing slave devices to listen on the serial data line for their address
- Master device sends address of target slave device and R/W flag

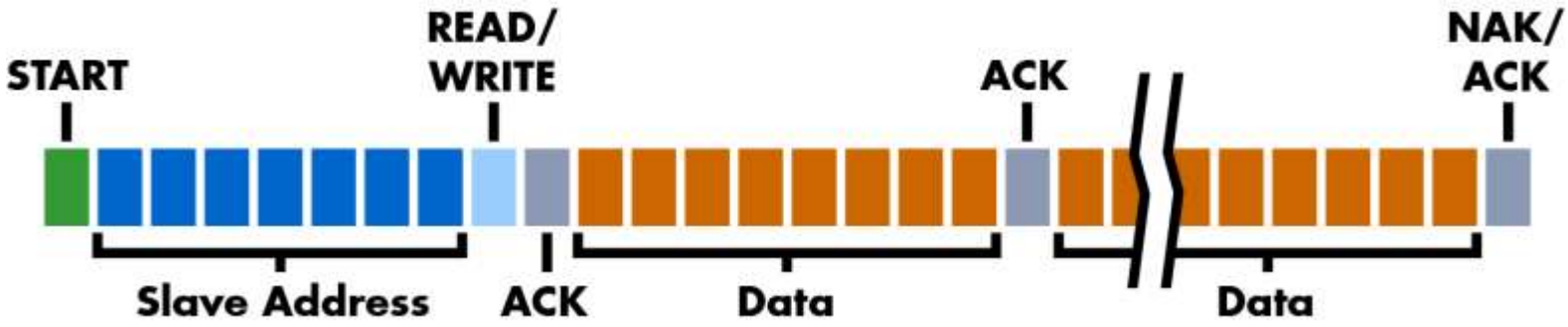
I2C Theory of Operation

- Master device issues start condition informing slave devices to listen on the serial data line for their address
- Master device sends address of target slave device and R/W flag
- Slave device with matching address responds with ACK



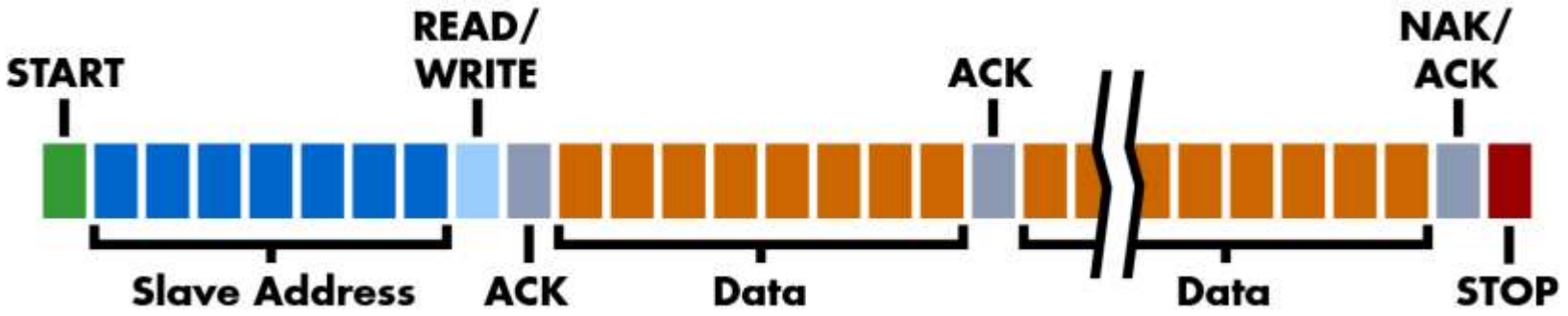
I2C Theory of Operation

- Master device issues start condition informing slave devices to listen on the serial data line for their address
- Master device sends address of target slave device and R/W flag
- Slave device with matching address responds with ACK
- Communication proceeds between master and slave
  - Either can receive of transmit: Transmitter sends 8 bits and receiver replies with a 1 bit ACK/NAK



I2C Theory of Operation

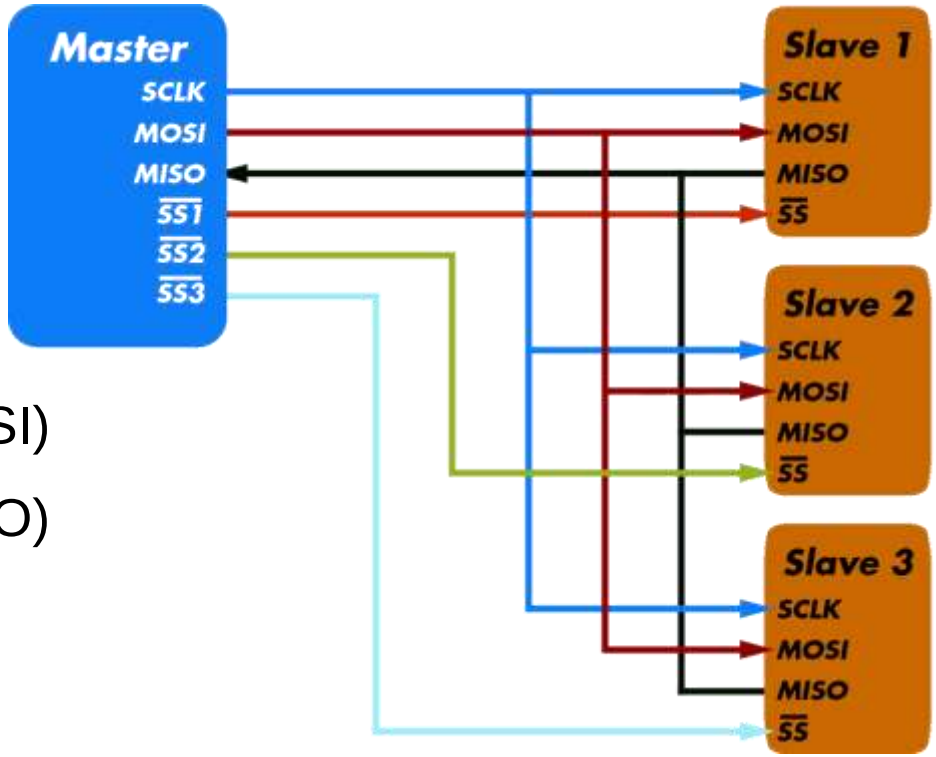
- Master device issues start condition informing slave devices to listen on the serial data line for their address
- Master device sends address of target slave device and R/W flag
- Slave device with matching address responds with ACK
- Communication proceeds between master and slave
  - Either can receive of transmit: Transmitter sends 8 bits and receiver replies with a 1 bit ACK/NAK
- Master issues STOP condition to terminate



SPI Overview

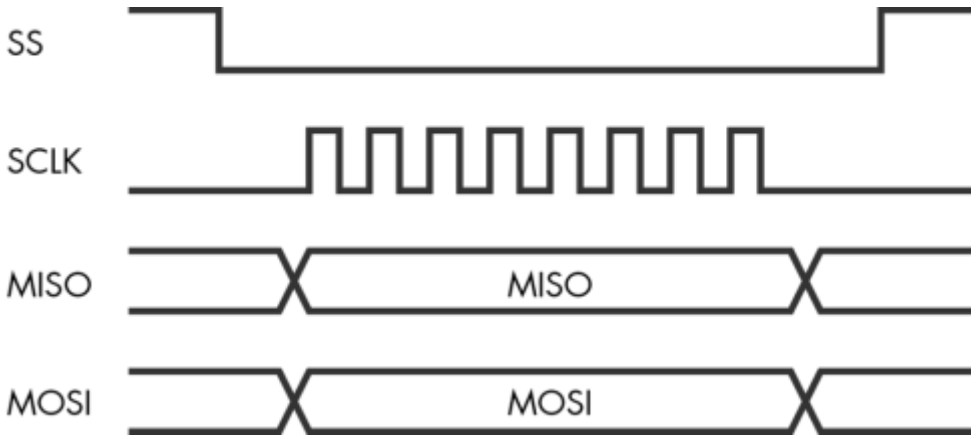
SPI Overview

- Serial Peripheral Interface bus
  - Full-duplex protocol
  - Ideally suited to data streaming applications
- Requires four signals:
  - Clock (SCLK)
  - Master out/Slave in (MOSI)
  - Master in/Slave out (MISO)
  - Slave select (SS)



SPI Theory of Operation

- All devices share SCLK, MOSI, MISO lines
  - SCLK is generated by master device for synchronization
  - Data is always transferred in both directions (MOSI/MISO)
- Each device on the SPI bus has its own SS line
  - The master pulls low on a slave's SS line to select a device for communication
- Exchange has no pre-defined protocol



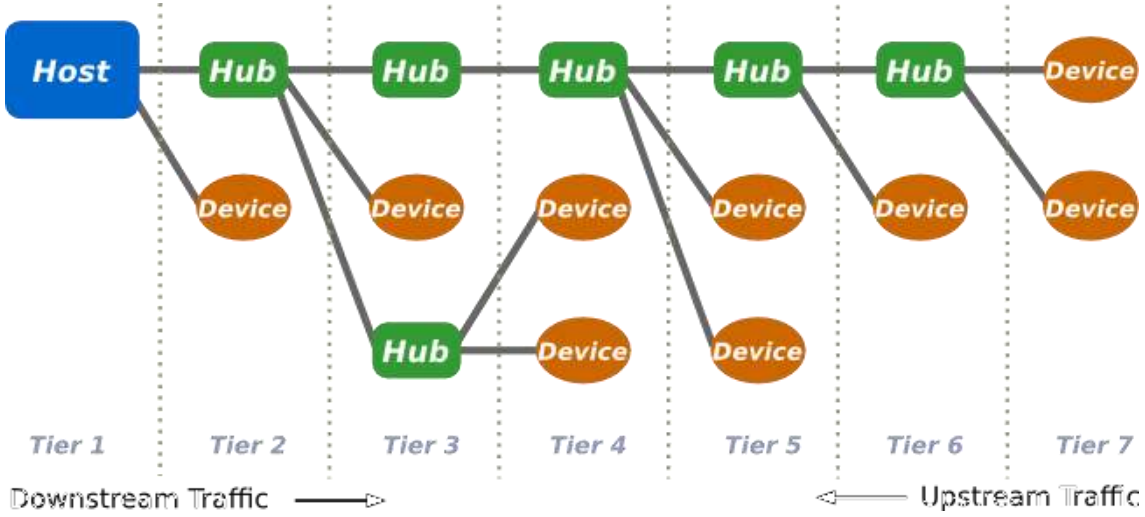
USB Overview

USB Overview

- Universal Serial Bus
  - A serial communications interface between devices
  - Standard interface for connecting peripheral devices to a host computer
  - Self-identifying devices and intelligent configuration

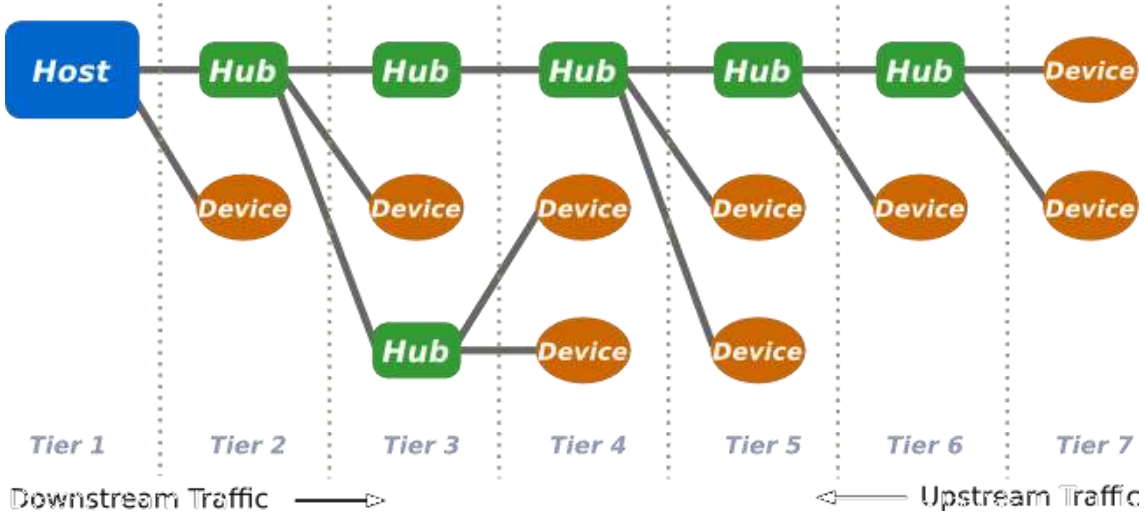
USB Architectural Overview

- Host-scheduled, token-based serial bus protocol
  - One Host
    - Usually a PC or embedded host
  - Up to 127 Devices
    - Respond to IN and OUT requests from the host



USB Architectural Overview

- Host-scheduled, token-based serial bus protocol
  - Zero or more hubs
    - Allows connection of multiple downstream devices to an upstream host or hub
  - Seven Tiers
    - Upper limit of 7 tiers, max number of hubs at 5 with the root host at tier 1



USB Terminology

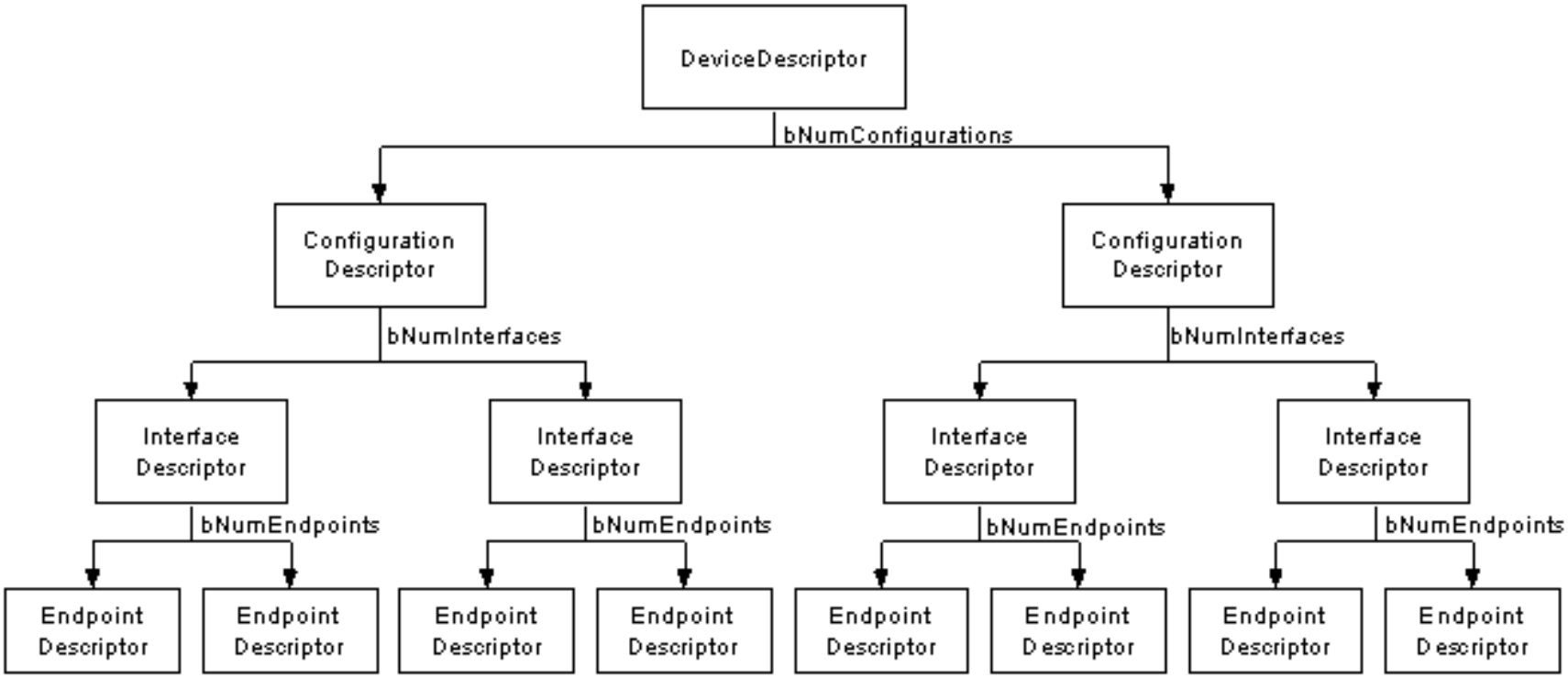
- Packet
  - The smallest element of data transmitted on the bus
- Endpoint and Pipe
  - **Endpoint** is the fundamental unit of communication in USB
  - **Pipes** represent a data pathway between the host and the device's endpoint

USB Terminology

- Four types of USB transfers
  - **Control**: Non-periodic transfers
    - Typically used for configuration
  - **Interrupt**: Guaranteed to occur in a certain time interval
  - **Isochronous**: Periodic, continuous transfer
    - No error checking
    - Good for video, audio
  - **Bulk**: General transfer scheme for large amounts of data
    - Error checking

USB Terminology

- Descriptors



USB Terminology

- Descriptor
  - Lists device information and capabilities
  - Device

Device Descriptor	Radix: auto
bLength	18 (0x12)
bDescriptorType	DEVICE (0x01)
bcdUSB	2.0 (0x0200)
bDeviceClass	Defined in Interface (0x00)
bDeviceSubClass	0x00
bDeviceProtocol	0x00
bMaxPacketSize0	8 (0x08)
idVendor	1121 (0x0461)
idProduct	19733 (0x4d15)
bcdDevice	2.0 (0x0200)
iManufacturer	(0x00)
iProduct	USB Optical Mouse (0x02)
iSerialNumber	(0x00)
bNumConfigurations	1 (0x01)

USB Terminology

- Descriptor
  - Lists device information and capabilities
  - Device, Configuration

Configuration Descriptor	Radix: auto
bLength	9 (0x09)
bDescriptorType	CONFIGURATION (0x02)
wTotalLength	34 (0x22)
bNumInterfaces	1 (0x01)
bConfigurationValue	1 (0x01)
iConfiguration	(0x00)
bmAttributes.Reserved1	0x1
bmAttributes.SelfPowered	Bus Powered (0x0)
bmAttributes.RemoteWakeup	Supported (0x1)
bmAttributes.Reserved0	0x00
bMaxPower	100mA (0x32)

USB Terminology

- Descriptor
  - Lists device information and capabilities
  - Device, Configuration, Interface

Interface Descriptor	Radix: auto
bLength	9 (0x09)
bDescriptorType	INTERFACE (0x04)
bInterfaceNumber	0x00
bAlternateSetting	0x00
bNumEndpoints	1 (0x01)
bInterfaceClass	Human Interface Device (0x03)
bInterfaceSubClass	Boot Interface (0x01)
bInterfaceProtocol	Mouse (0x02)
iInterface	(0x00)

USB Terminology

- Descriptor
  - Lists device information and capabilities
  - Device, Configuration, Interface, Endpoint, and String

Endpoint Descriptor	Radix: auto
bLength	7 (0x07)
bDescriptorType	ENDPOINT (0x05)
bEndpointAddress	1 IN (0x81)
bmAttributes.TransferType	Interrupt (0x3)
bmAttributes.Reserved0	0x00
wMaxPacketSize	4 bytes (1 transaction per microframe if HS) (0x0004)
bInterval	LS/FS:10ms HS:64ms (0x0a)

String Descriptor	Radix: auto
bLength	36
bDescriptorType	STRING (0x03)
bString	USB Optical Mouse

USB Terminology

- Enumeration
  - Process where the host detects the devices on the bus, assigns an address, and reads their descriptors

Statistics | Enumeration

Device Details			
Product	USB Optical Mouse		
Serial Number	<none>		
Manufacturer	<none>		
Class	Defined in Interface		
VID	PID	Rev	USB
0x0461	0x4d15	2.0	2.0
Configurations			
Config 1	Bus Powered, 100mA, RemoteWakeup		
OTG	none / corrupted		
IF 0 (alt 0)	HID, Boot Interface, Mouse		
EP 1 IN	Intr, 4B, LS/FS:10ms HS:64ms		
BOS			
BOS descriptor not detected or corrupted.			

USB Terminology

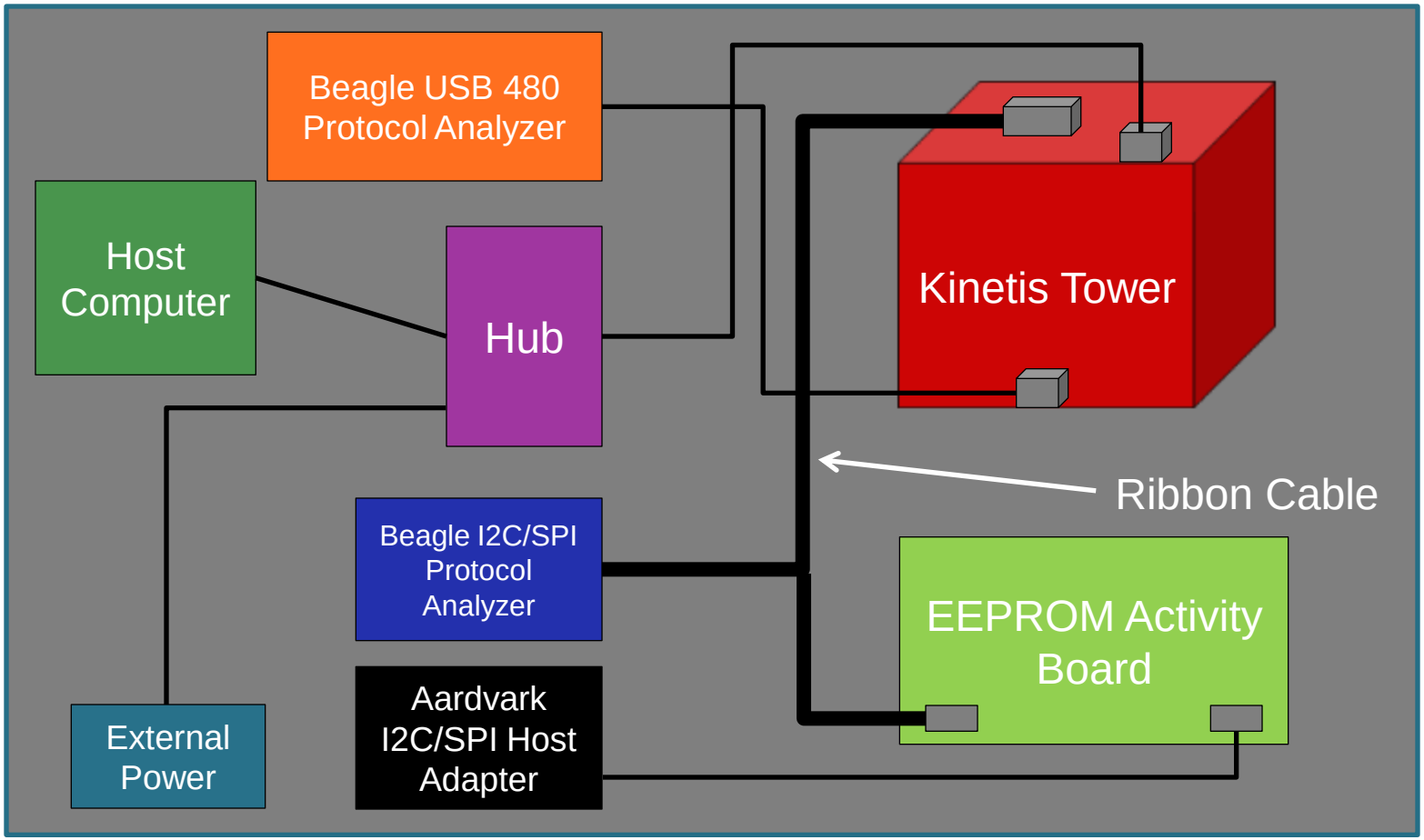
- Class
  - Pre-defined protocols to simplify drivers from common types of devices
  - Popular USB classes

Class	Usage	Description
01h	Interface	Audio
02h	Device & Interface	Communications and CDC Control
08h	Interface	Mass Storage
09h	Device	USB hub

Lab Introduction

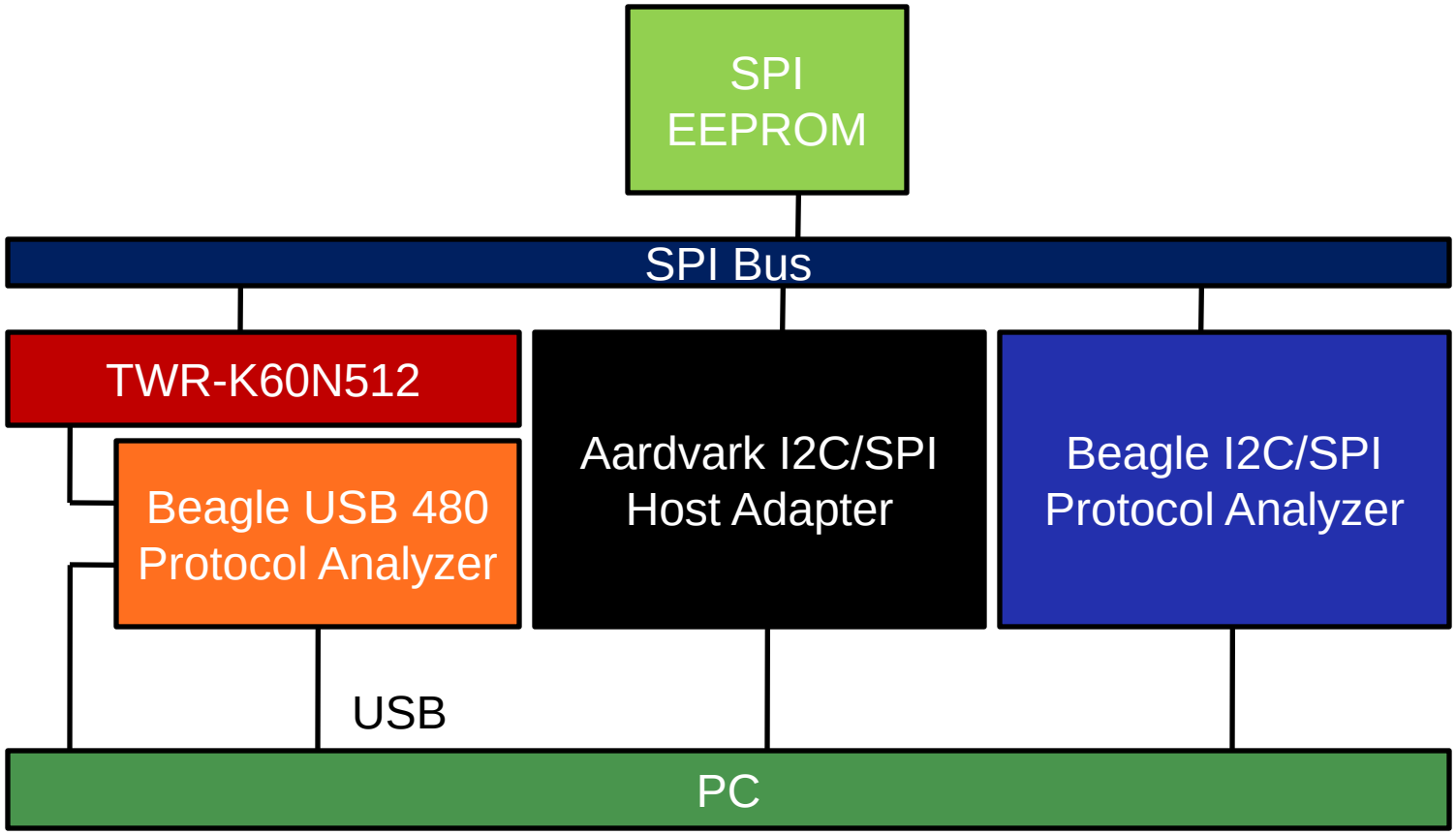
Lab Set Up

Physical Set Up



Lab Set Up

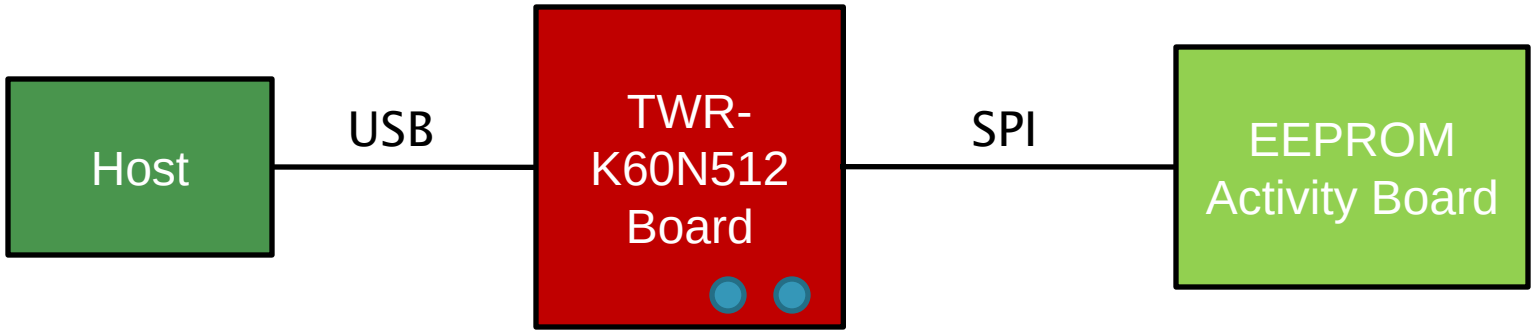
Logical Set Up



Lab Set Up

Final Product Design

- TWR-K60N512 fetches a record from the SPI EEPROM on button press
- TWR-K60N512 sends record over CDC USB and displays on host



- Button Behavior
  - SW1: Displays next record
  - SW2: Displays previous record

Programming the EEPROM

Programming the EEPROM

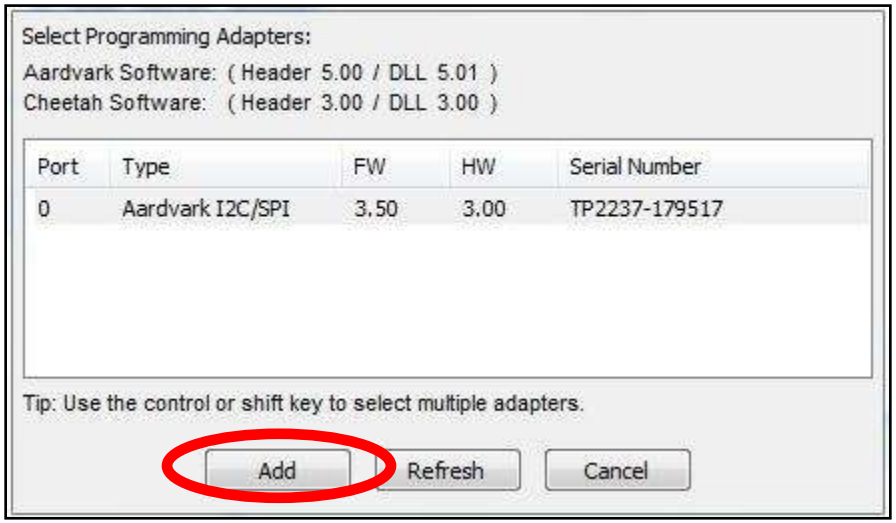
- **Objective**: Introduce the Aardvark I2C/SPI Host Adapter and Flash Center Software

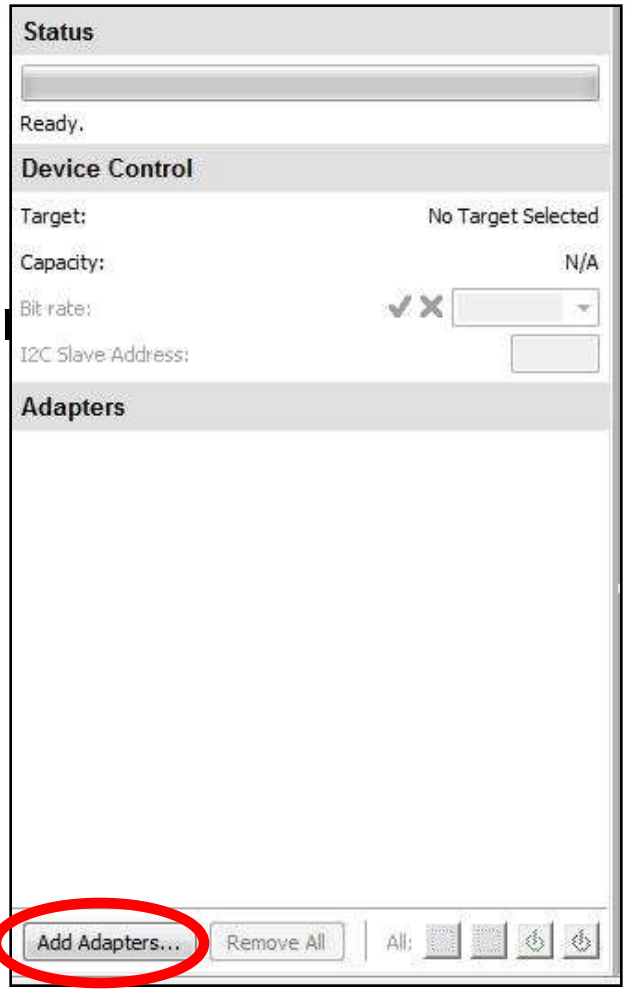
- **Task**: Use the Aardvark I2C/SPI Host Adapter and Flash Center Software to read and program the SPI EEPROM

Programming the EEPROM

Configuring the Aardvark I2C/SPI Host Adapter for use

1. Open the **Flash Center Software**
2. Click on **Add Adapters**
3. Select Aardvark I2C/SPI Host Adapter
4. Click **Add**

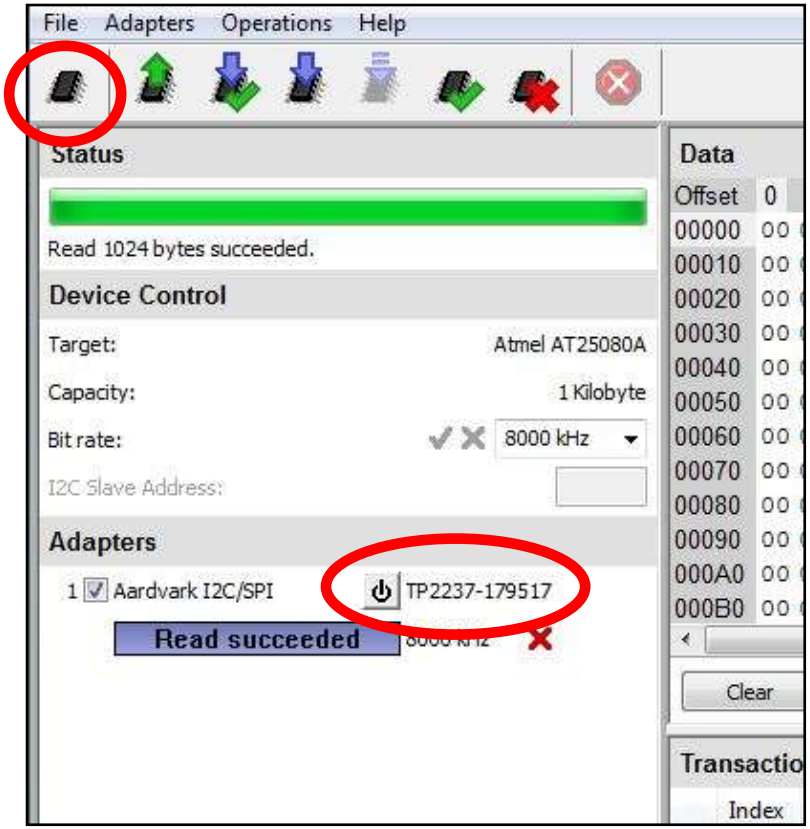




Programming the EEPROM

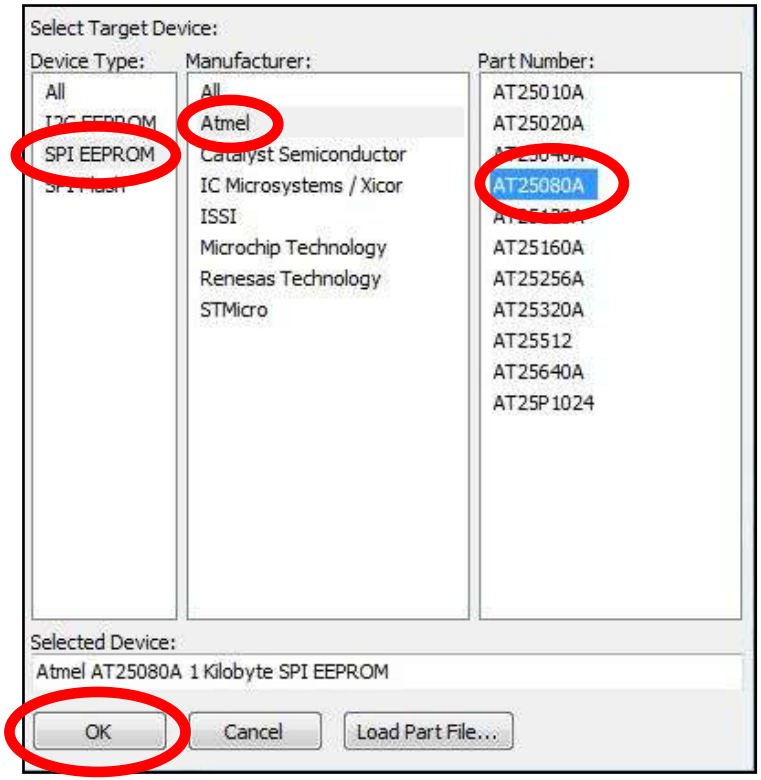
5. Turn on the **Target Power** button

6. Click on **Choose Target** to specify which part you will be using



Programming the EEPROM

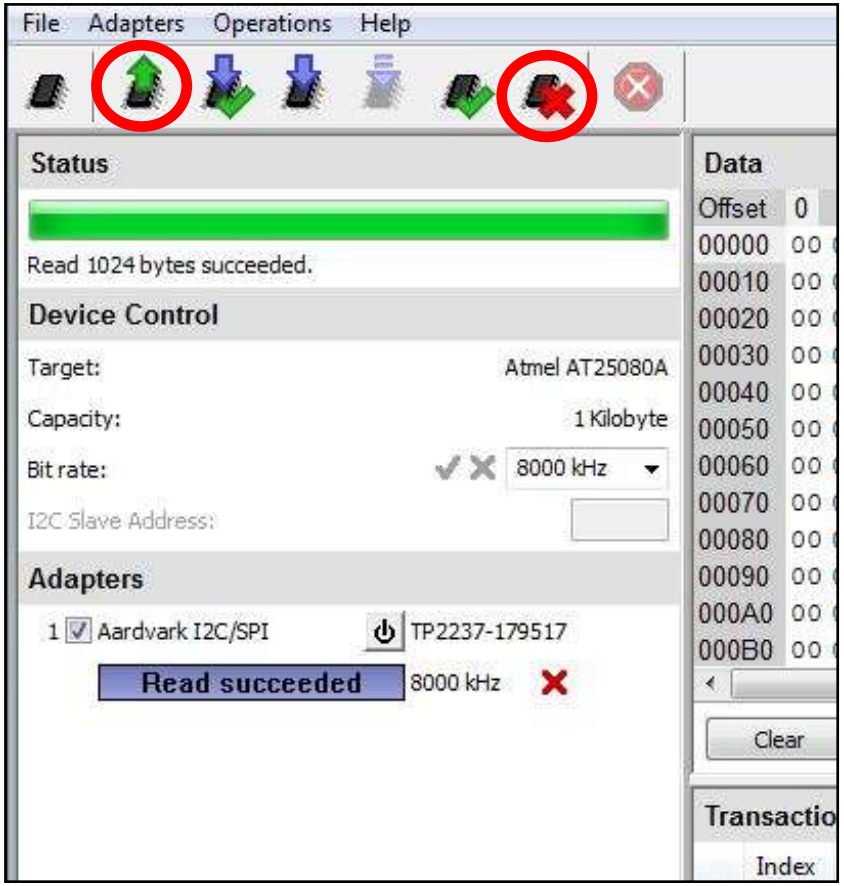
7. Select **SPI EEPROM** under Device Type
8. Under Manufacturer, select **Atmel**
9. Under Part Number, choose **AT25080A**
10. Click **OK**



Programming the EEPROM

Reading and Erasing the SPI EEPROM Contents

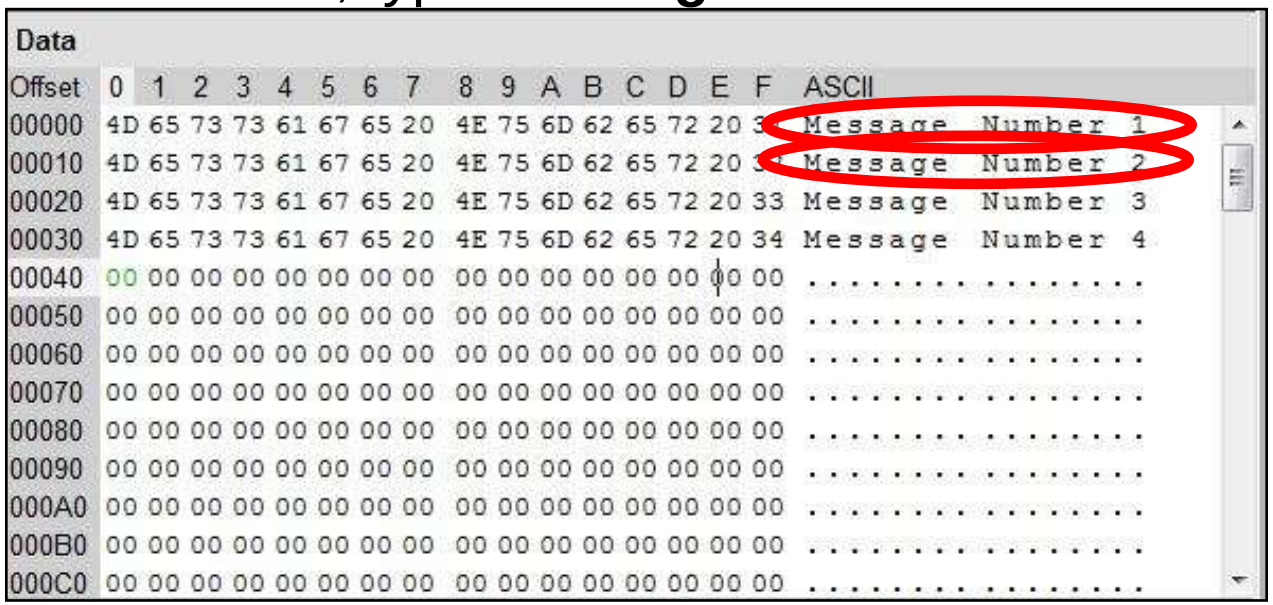
11. Click the **Read Target** icon
12. Click **Erase**
13. Select **OK**
14. Click **Read Target** again



Programming the EEPROM

Writing Data to the SPI EEPROM

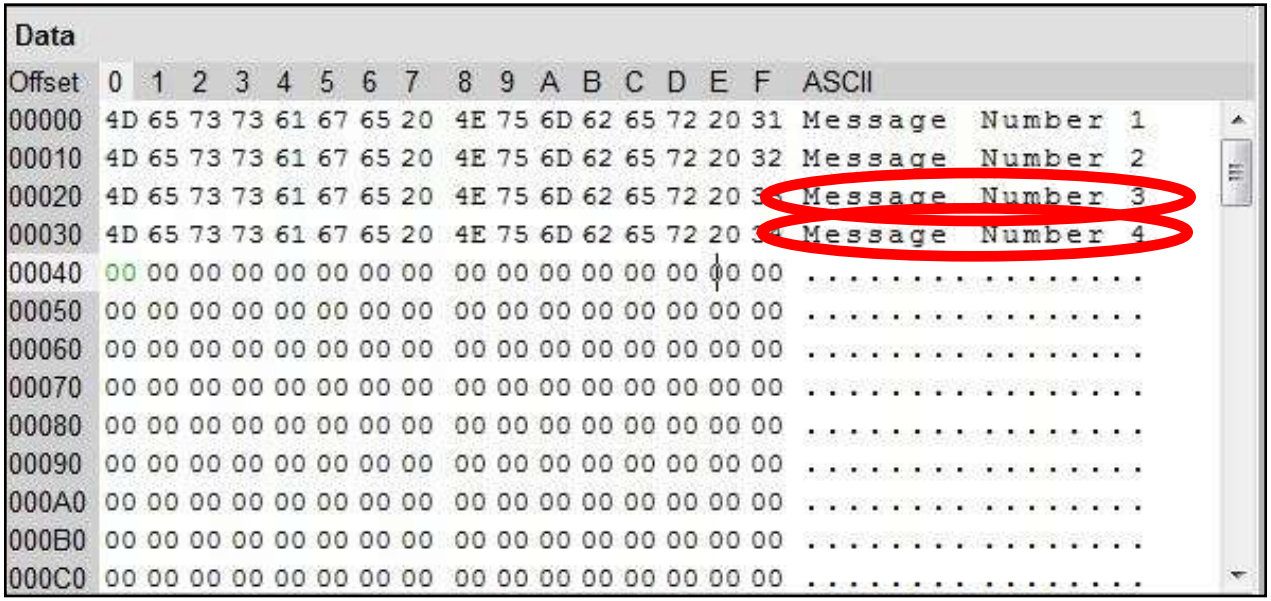
15. Click into the ASCII editor
16. On the first line, type **Message Number 1**
17. On the second line, type **Message Number 2**



Programming the EEPROM

18. On the third line, type **Message Number 3**

19. On the fourth line, type **Message Number 4**

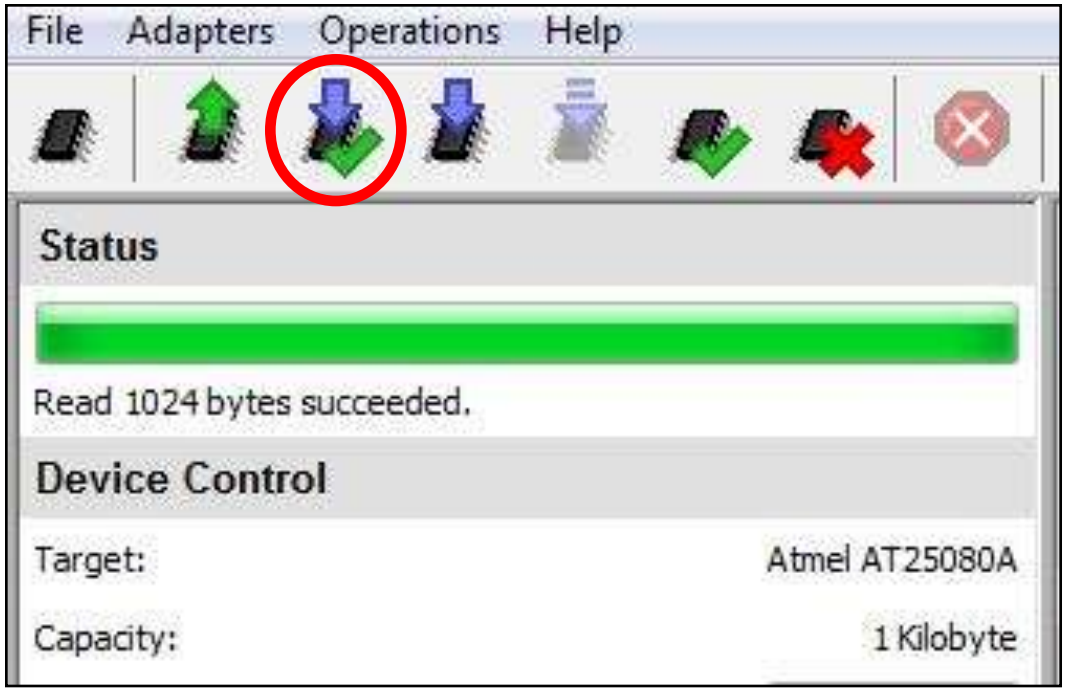


Programming the EEPROM

18. On the third line, type **Message Number 3**

19. On the fourth line, type **Message Number 4**

20. Select the **Program and Verify** icon



Bug 1: Enumeration of CDC Device

Bug 1: Enumeration of CDC Device

- **Objective:** Introduce the Beagle USB 480 Protocol Analyzer and Data Center Software to view enumeration details.

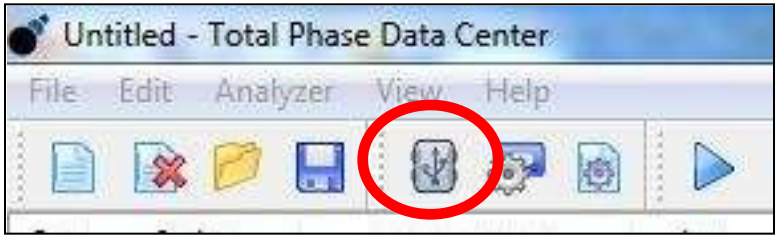
- **Task:** Set up and start a live capture using the Data Center Software to assist in fixing the error in the device descriptors.

Beagle USB 480
Protocol Analyzer

Bug 1: Enumeration of CDC Device

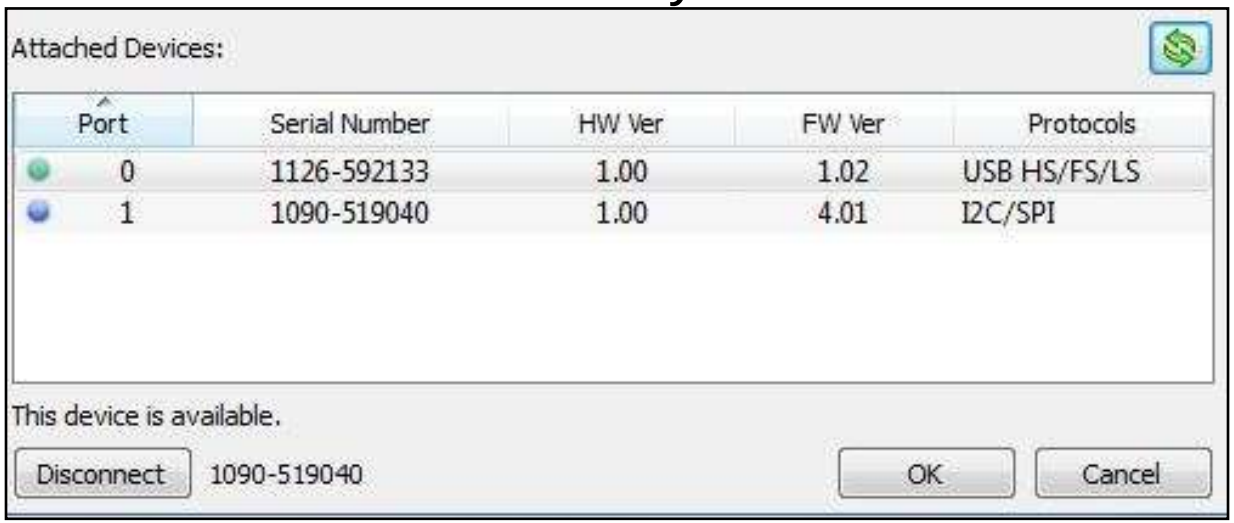
Configuring the Beagle USB 480 Protocol Analyzer for use

1. Open the Data Center Software



2. Click **Connect to Analyzer**

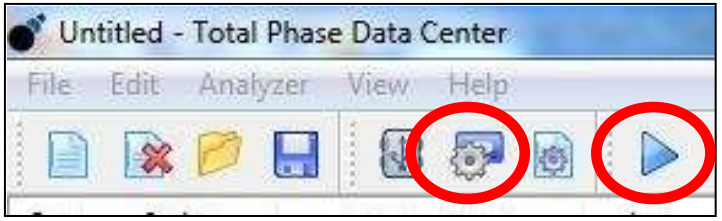
3. Select the USB HS/FS/LS analyzer and click **OK**



Bug 1: Enumeration of CDC Device

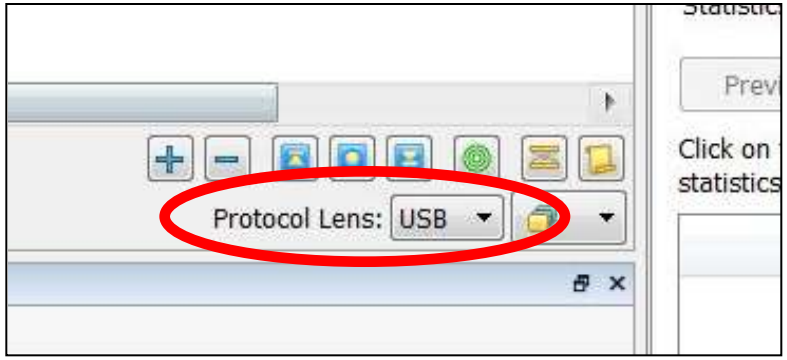
4. Click **Device Settings**.

5. Change the capture protocol to **USB**



6. Set the protocol lens to **USB**

7. **Start** the capture



Bug 1: Enumeration of CDC Device

Launching the Project and the Kinetis Tower System

8. Launch the **IAR Embedded Workbench**

9. Click **File** → **Open** → **Workspace**

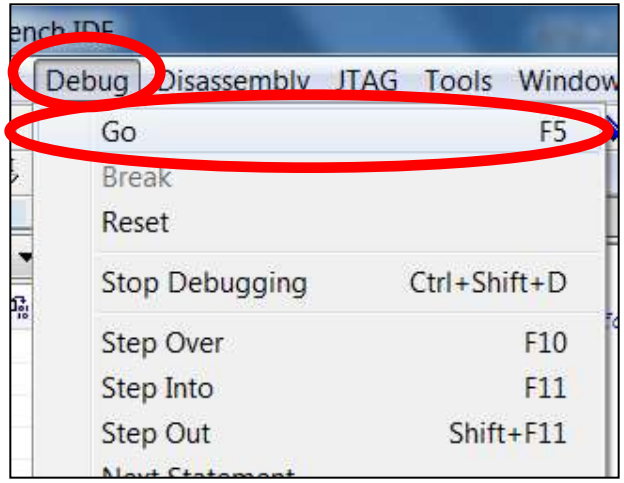
10. Open the **TotalPhaseKinetisTower.work** folder in Desktop\TotalPhase\materials\kinetis_projects

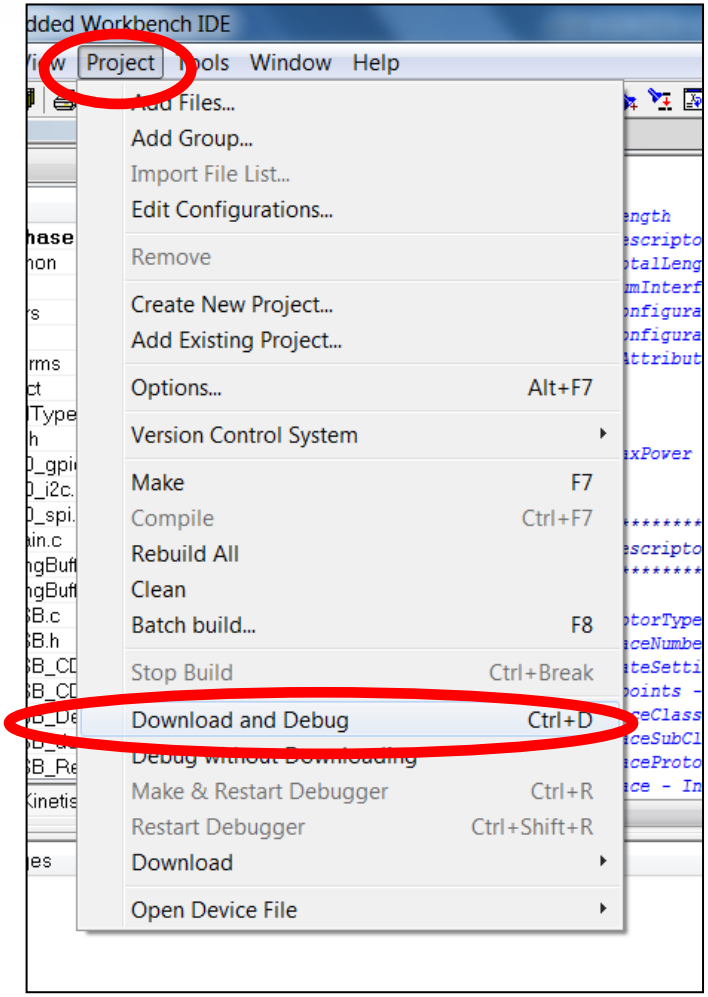
11. Select the **TotalPhaseKinetisTower.eww** project

Bug 1: Enumeration of CDC Device

13. Select **Project → Download and Debug**

14. Select **Debug → Go**





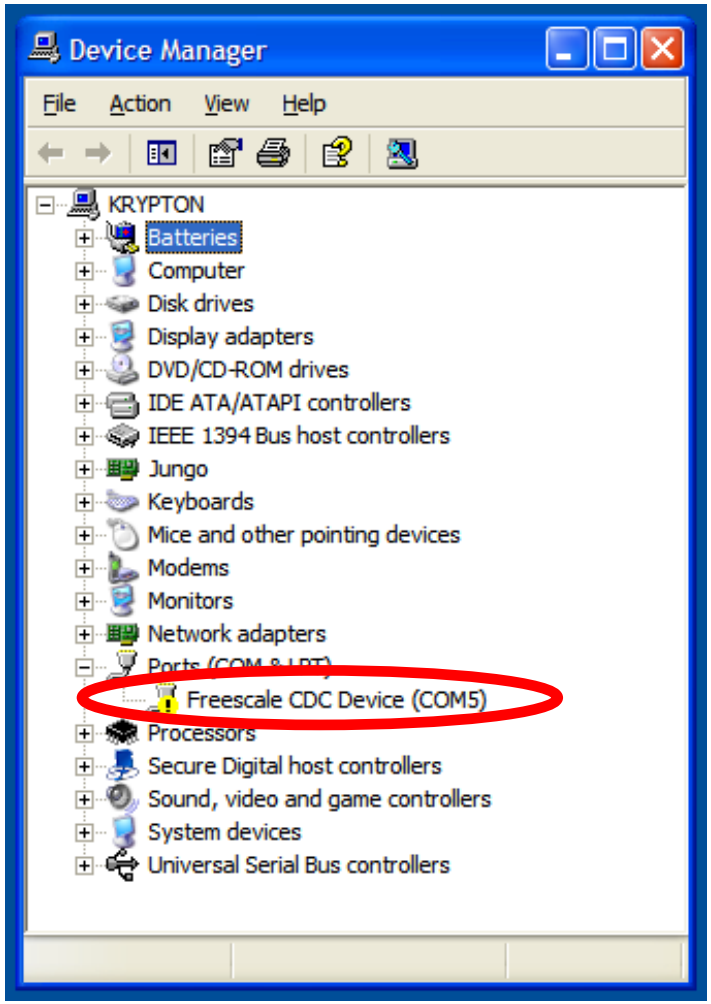
Bug 1: Enumeration of CDC Device

- Go back to the Data Center Software. You should notice the device has gone into Suspend.

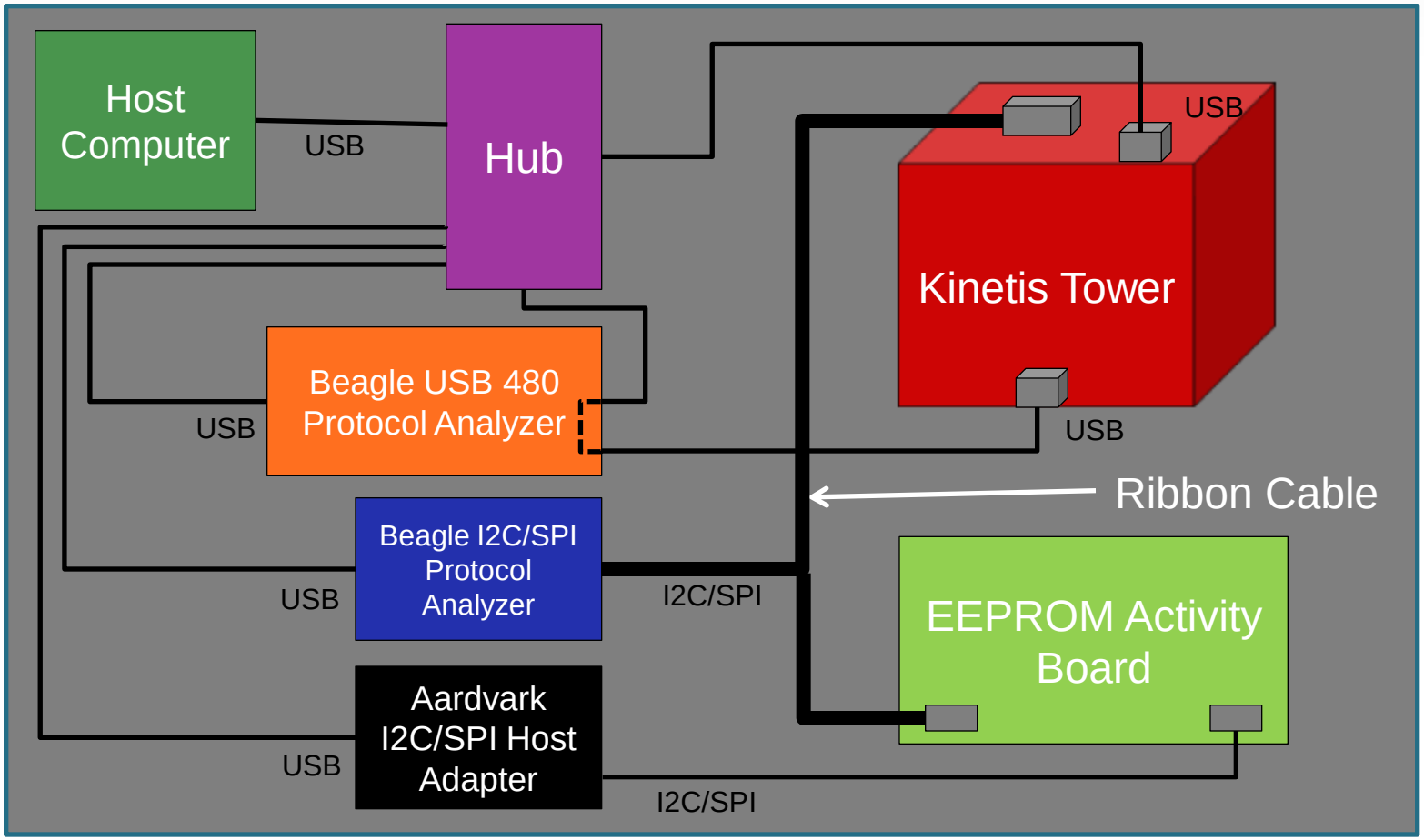
Sp	Index	m:s.ms.us	Len	Err	Dev	Ep	Record	Summary
FS	31	2:05.005.412	11.0 ms				<Reset> / <Chirp J> / <Tiny J>	
FS	32	2:05.016.472					<Full-speed>	
FS	34	2:05.085.548	10.9 ms				<Reset> / <Chirp J> / <Tiny J>	
FS	35	2:05.096.474					<Full-speed>	
FS	47	2:05.167.061	8 B		06	00	Get Device Descriptor	Index=0 Length=8
FS	61	2:05.168.811	18 B		06	00	Get Device Descriptor	Index=0 Length=18
FS	75	2:05.169.686	67 B		06	00	Get Configuration Descriptor	Index=0 Length=255
FS	97	2:05.171.939	4 B		06	00	Get String Descriptor	Index=0 Length=255
FS	111	2:05.172.814	18 B		06	00	Get String Descriptor	Index=2 Length=255
FS	125	2:05.173.941	18 B		06	00	Get String Descriptor	Index=3 Length=255
FS	139	2:05.176.815	18 B		06	00	Get Device Descriptor	Index=0 Length=18
FS	153	2:05.178.438	67 B		06	00	Get Configuration Descriptor	Index=0 Length=265
FS	175	2:05.185.440	0 B		06	00	Set Configuration	Configuration=1
FS	185	2:05.217.321	0 B		06	00	Set Configuration	Configuration=0
FS	195	2:05.217.819	0 B	I	06	00	IN txn	
FS	200	2:05.246.817	246 ms	T			<Suspend>	

Bug 1: Enumeration of CDC Device

- Device manager with flagged device



Please spend the next few minutes debugging this problem

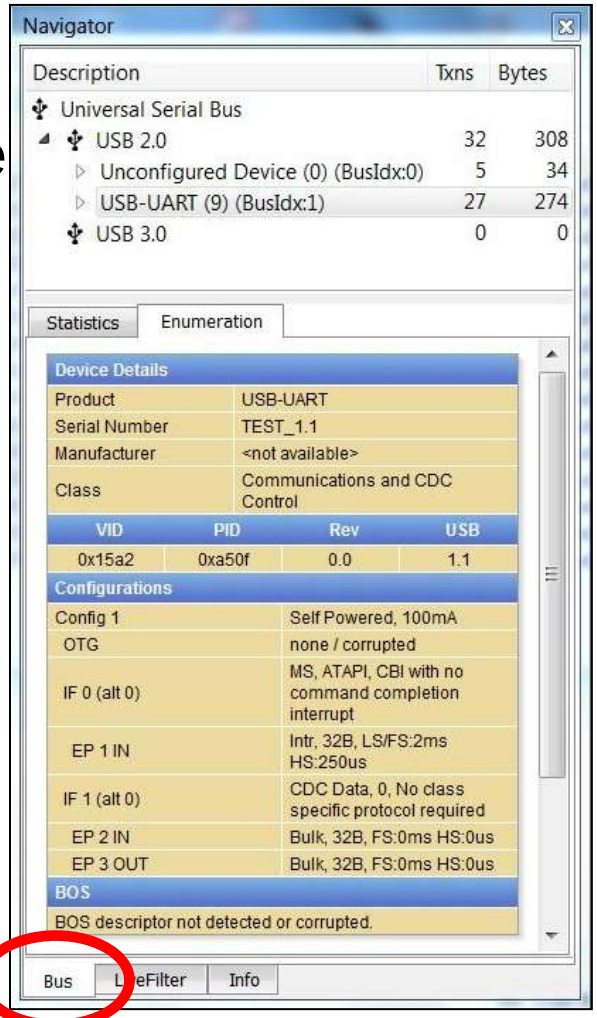


Bug 1: Enumeration of CDC Device

Using the Data Center Software to Find the Problem

14. Go back to the Data Center Software

15. Go to the **Bus** pane

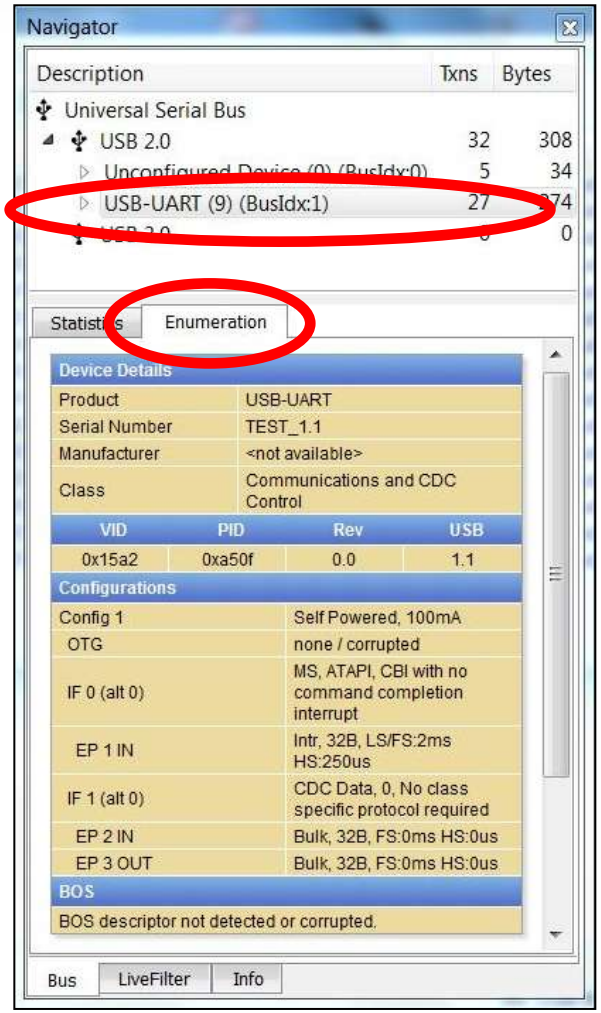


Bug 1: Enumeration of CDC Device

Using the Data Center Software to Find the Problem

16. Select the **USB-UART** device

17. Click on the **Enumeration** tab

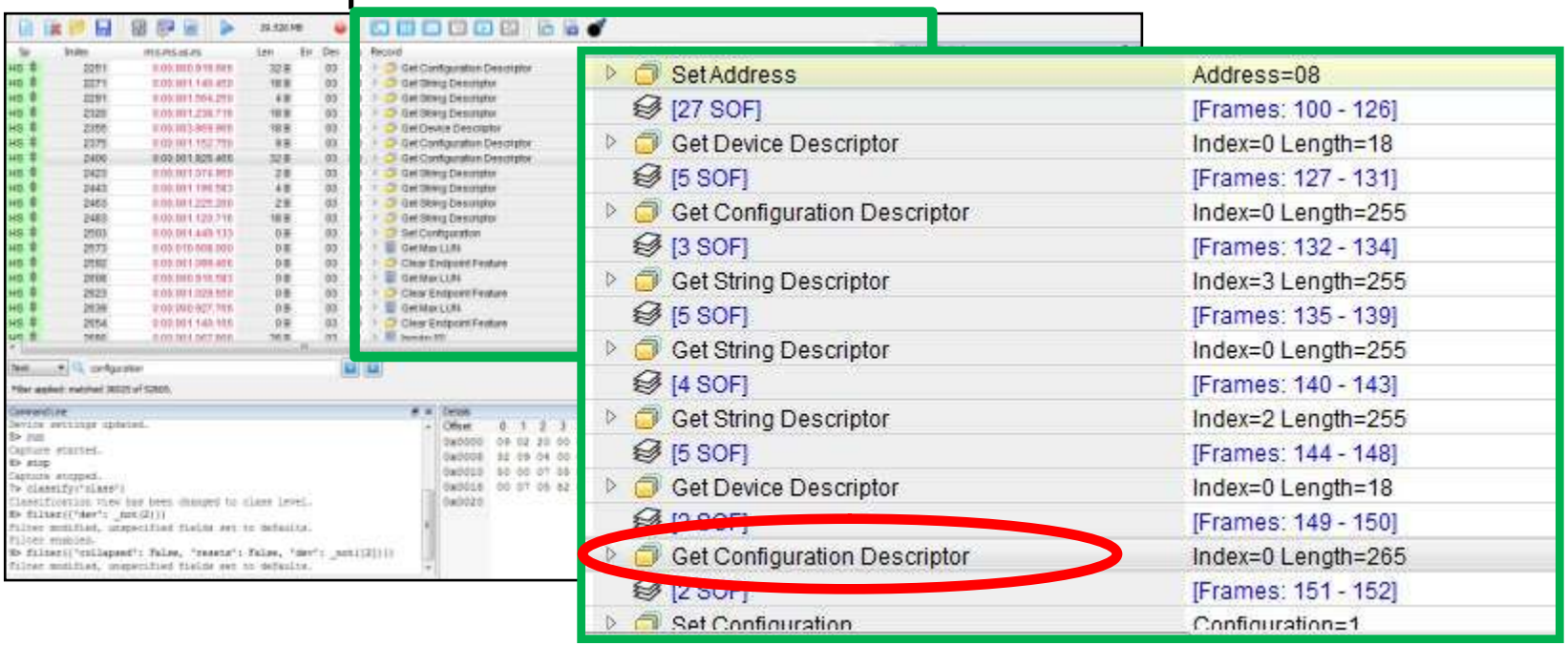


Bug 1: Enumeration of CDC Device

18. Go to the Data Transaction Window

19. Scroll to find the second **Get Configuration Descriptor** packet

20. Click on the packet

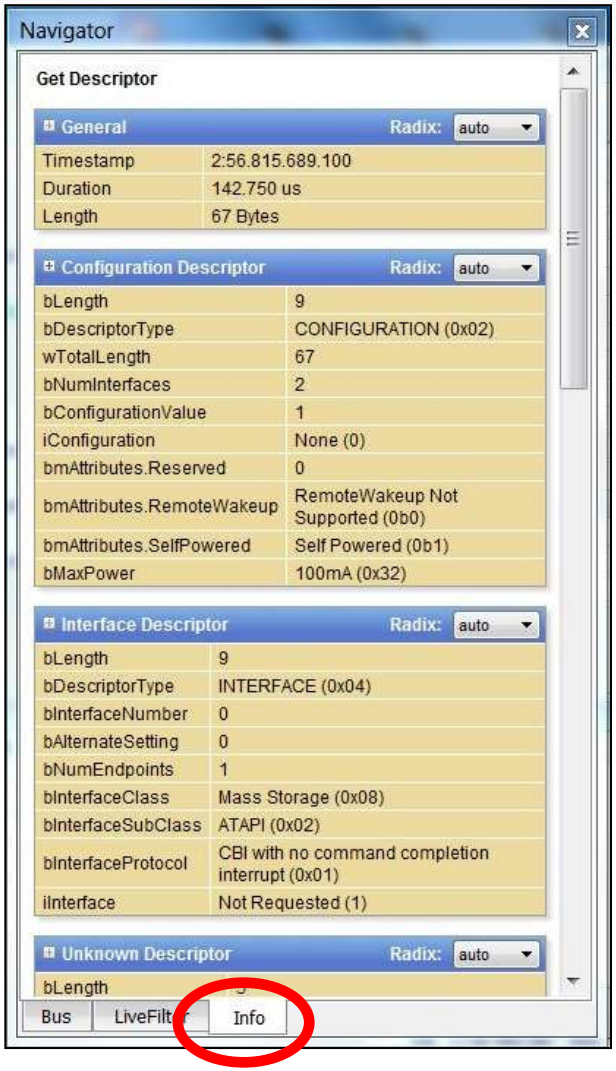


Bug 1: Enumeration of CDC Device

21. Go to the **Info** tab

22. Note the **bInterfaceClass** description

Interface Descriptor	Radix: auto
bLength	9
bDescriptorType	INTERFACE (0x04)
bInterfaceNumber	0
bAlternateSetting	0
bNumEndpoints	1
bInterfaceClass	Mass Storage (0x08)
bInterfaceSubClass	ATAPI (0x02)
bInterfaceProtocol	CBI with no command completion interrupt (0x01)
iInterface	Not Requested (1)

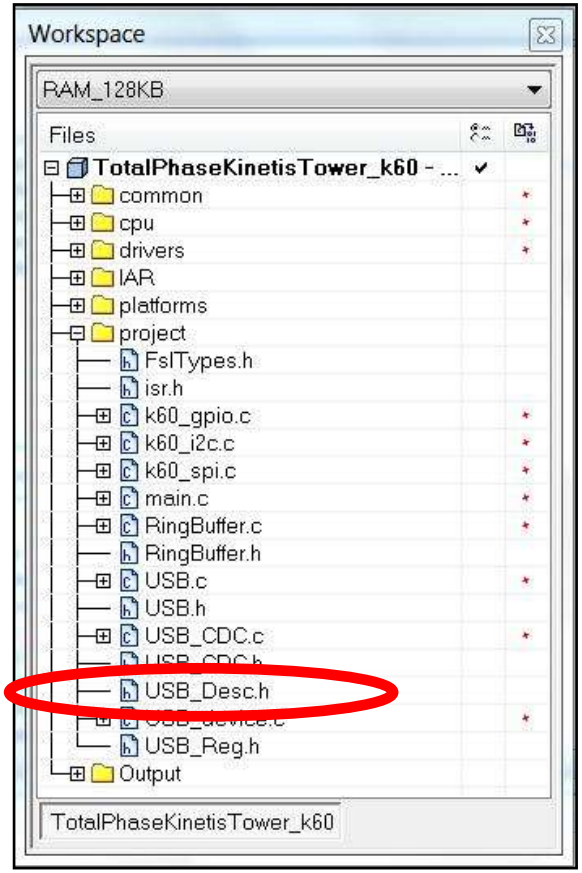


Bug 1: Enumeration of CDC Device

Fixing the USB Descriptor Bug

23. Go back to the IAR Embedded Workbench program

24. Open **USB_Desc.h** from the directory



Bug 1: Enumeration of CDC Device

25. Scroll and find the first Interface Descriptor section

26. Change the bInterface class from 0x08 to **0x02** in Line 123

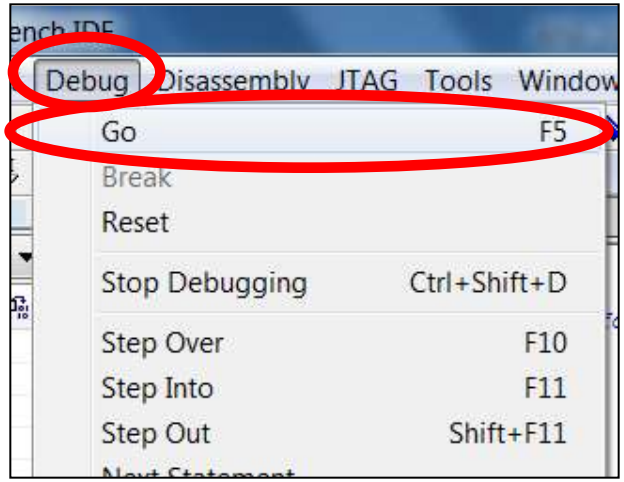
```
/*****************************************************************************
*       Interface Descriptor
*****************************************************************************/
   0x09, // blength
   0x04, // bDescriptorType - Interface descriptor
   0x00, // bInterfaceNumber - Index of the iface (zero based)
   0x00, // bAlternateSetting;
   0x01, // bNumEndpoints - 1 endpoint
   0x08, // bInterfaceClass - CDC Control
   0x02, // bInterfaceSubClass - Abstract Control Model
   0x01, // bInterfaceProtocol - AT Commands: V.250
   0x01, // iInterface - Index to String descriptor
```

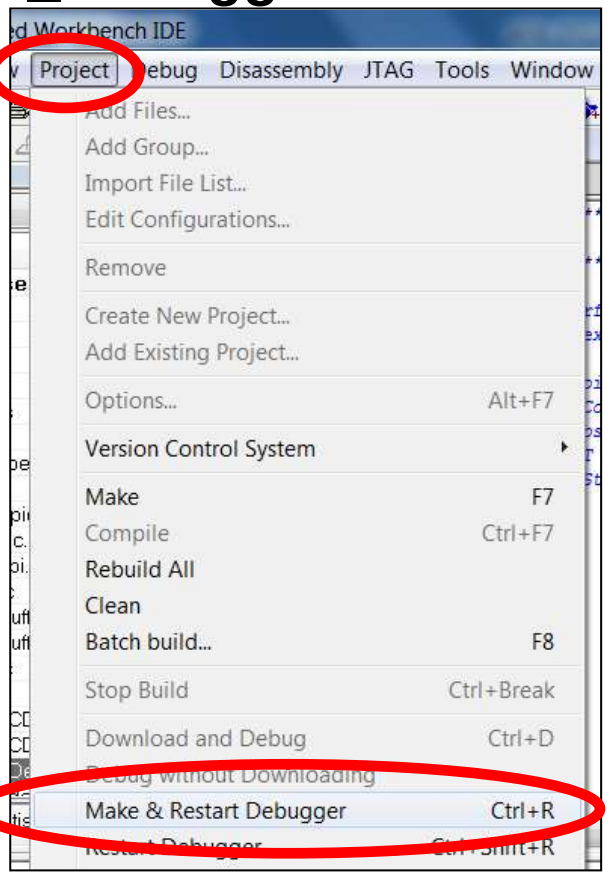
Bug 1: Enumeration of CDC Device

Verifying the Solution

27. Select **Project → Make_Restart_Debugger**

28. Select **Debug → Go**

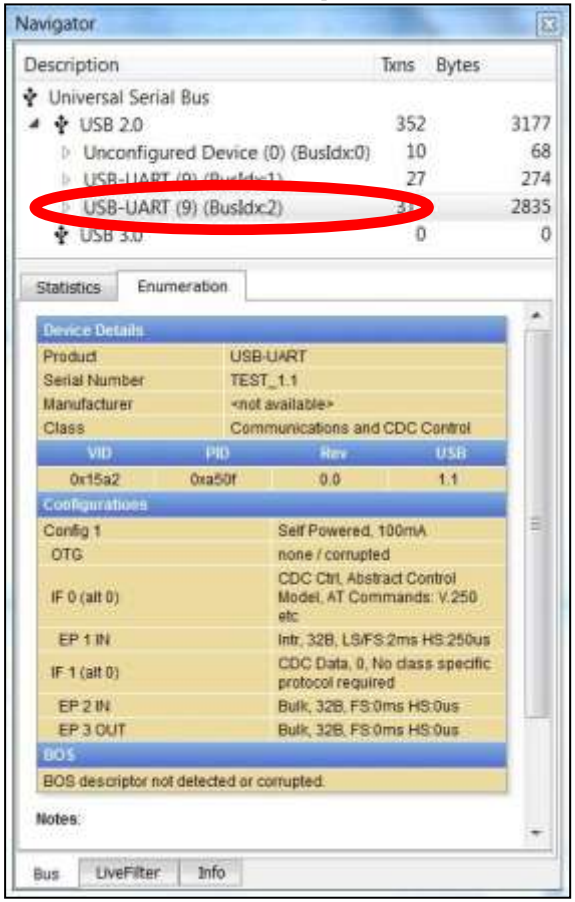




Bug 1: Enumeration of CDC Device

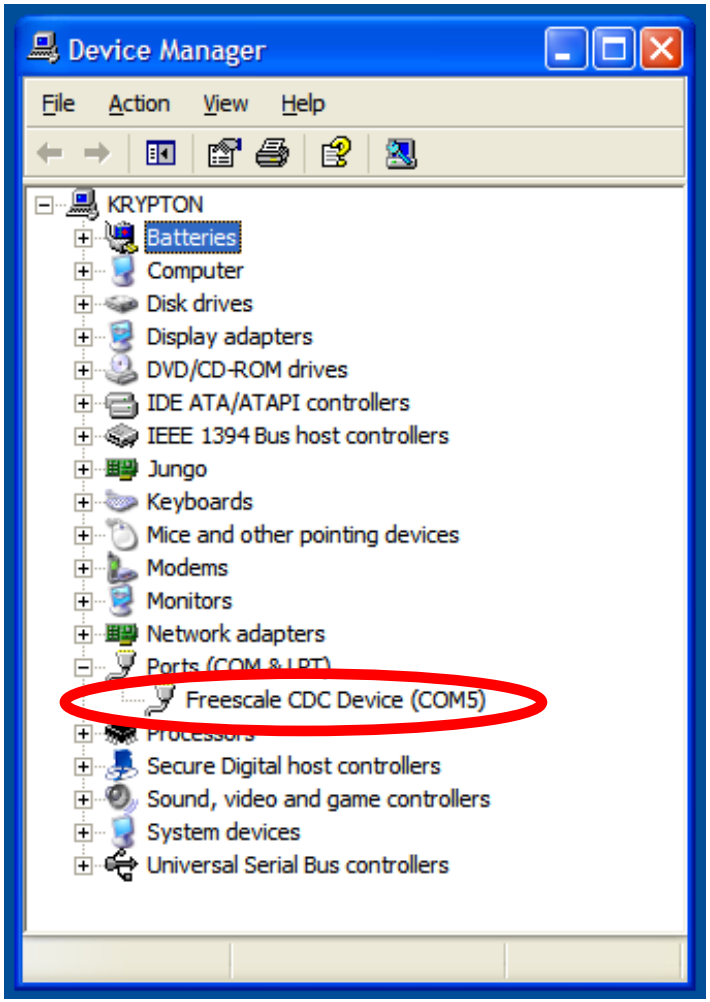
29. Go back to the Data Center Software

30. Go back to the **Bus** pane and select the second **USB-UART** device



Bug 1: Enumeration of CDC Device

31. Go back to the **Device Manager**



Bug 2: SPI Programming

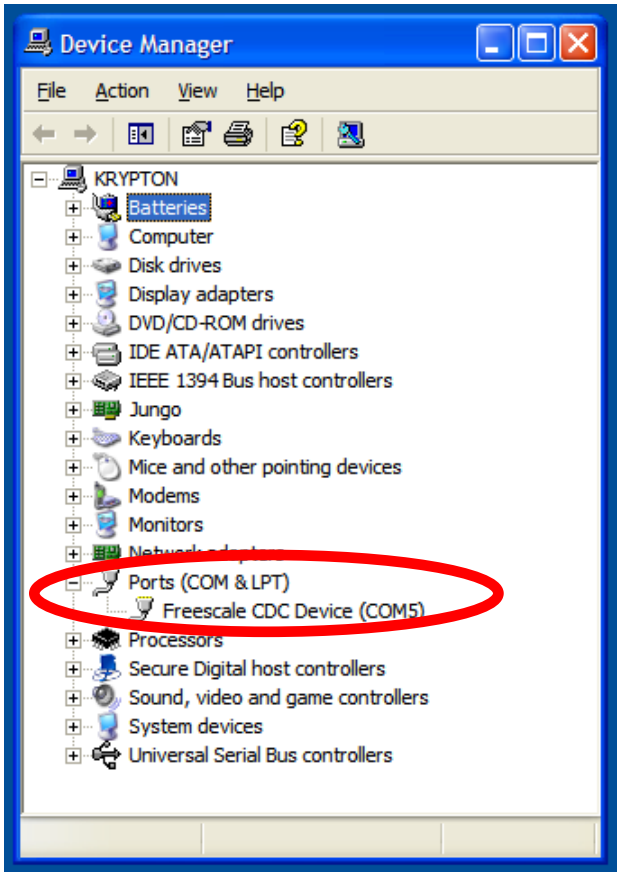
Bug 2: SPI Programming

- **Objective**: Introduce the Beagle I2C/SPI Protocol Analyzer and establish communication with the Kinetis Tower from the host PC

- **Task**: Use the Terminal Window application to connect to the COM port exposed by the CDC USB device

Bug 2: SPI Programming

1. Go back to the **Device Manager** and select **Ports**
2. Please note your number after the **“COM”**
3. Write this number down



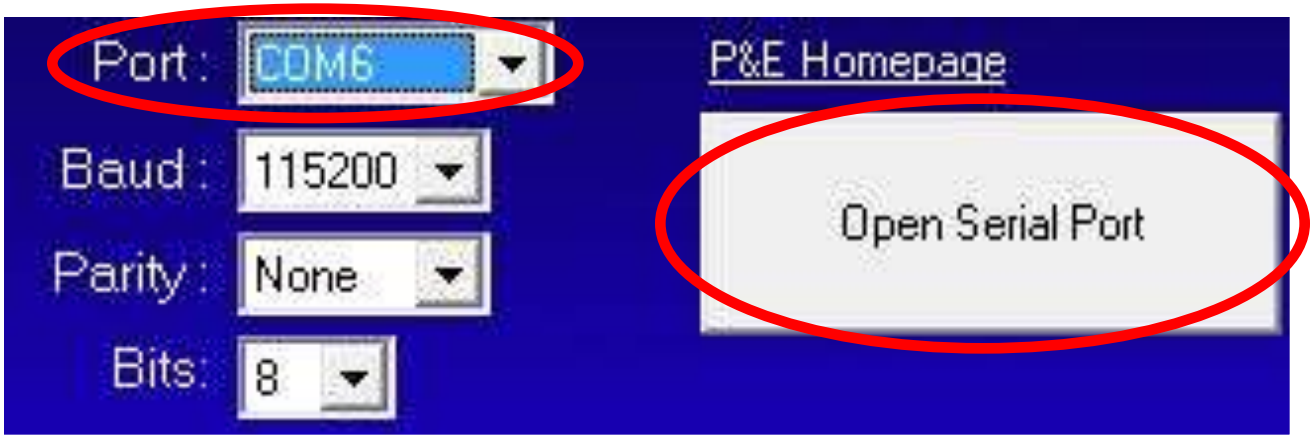
Bug 2: SPI Programming

Communicating using the CDC Device

4. Open the Kinetis Toolkit by clicking Start > Programs > P&E Toolkit > Utilities > TerminalWindow

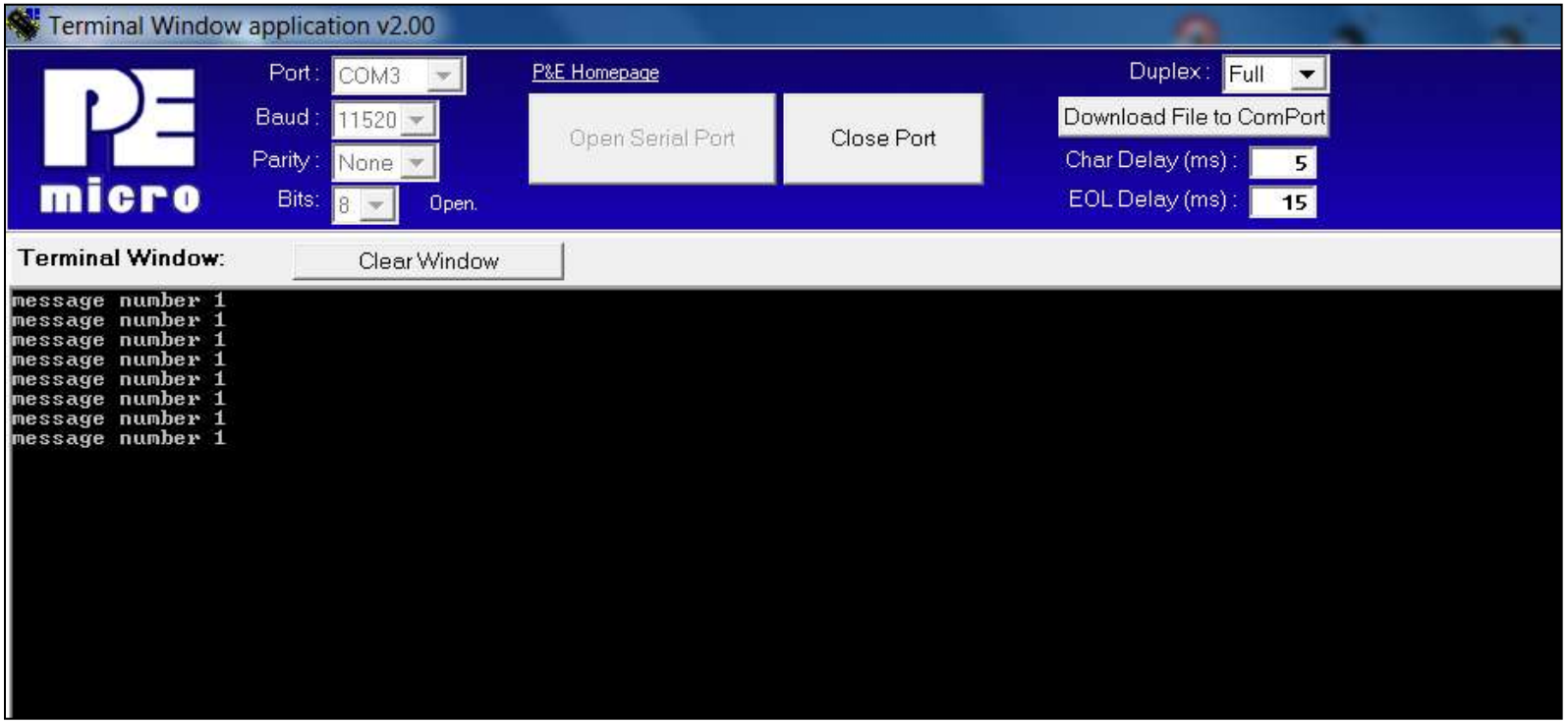
5. Select COM#, where # represents the number written down previously

6. Click Open Serial Port



Bug 2: SPI Programming

7. Click on one of the two blue buttons located on the board
8. Click the buttons a few times. You may receive a message like this:

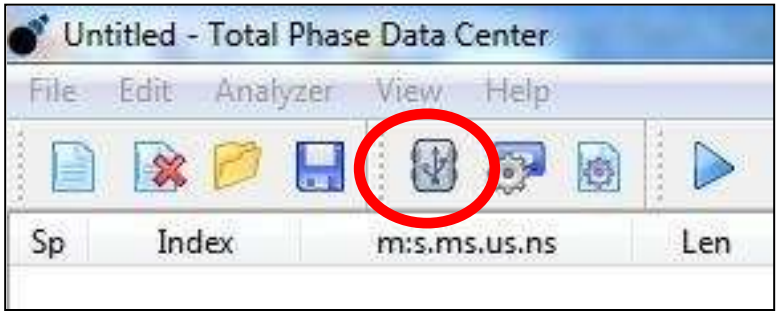


Bug 2: SPI Programming

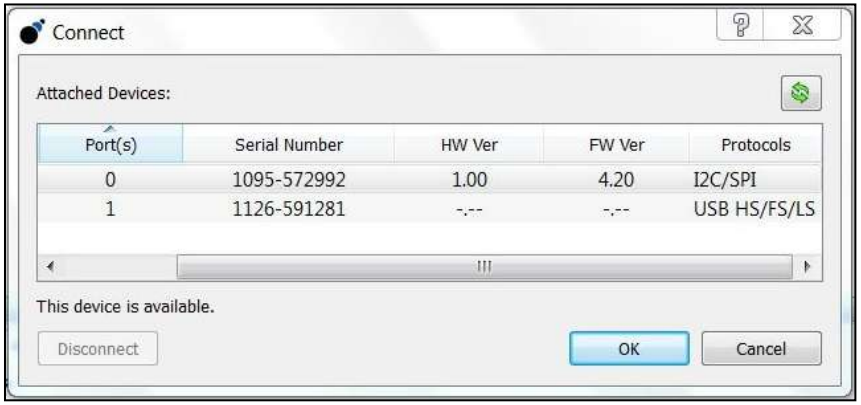
Configuring the Beagle I2C/SPI Protocol Analyzer for use

9. Open another instance of the **Data Center Software**

10. Click **Connect**

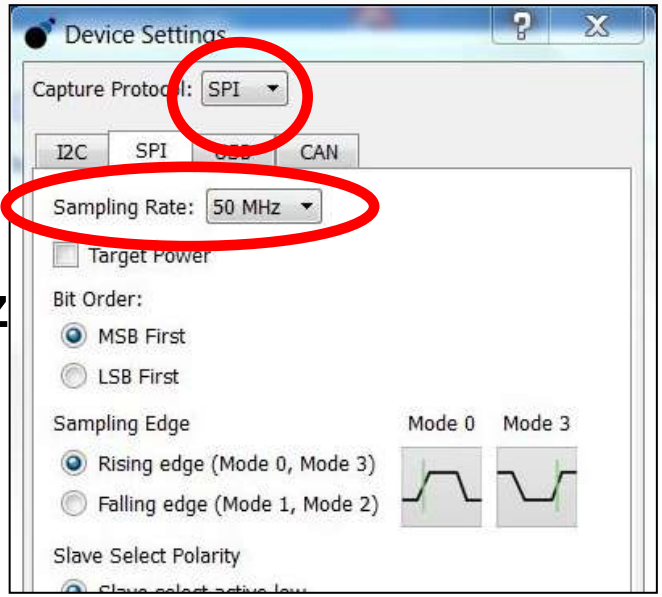


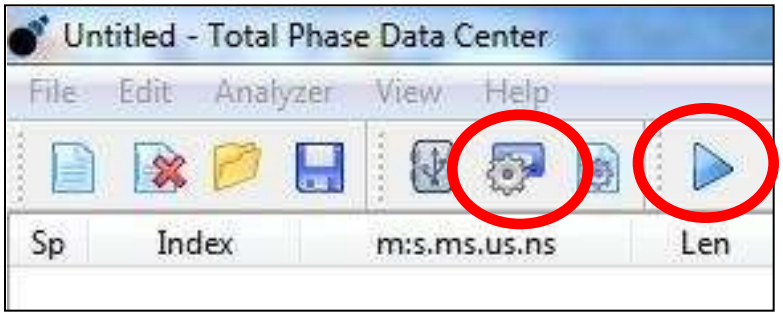
11. Connect to the **Beagle I2C/SPI Protocol Analyzer**

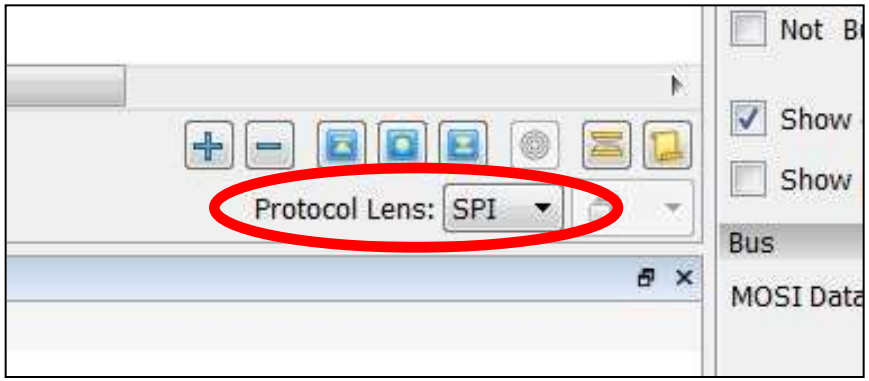


Bug 2: SPI Programming

12. Go **to Device Settings**
13. Select **SPI** in the pull down menu
14. Change the sampling rate to **50 MHz**
15. Change the protocol lens to **SPI**
16. **Start** the capture





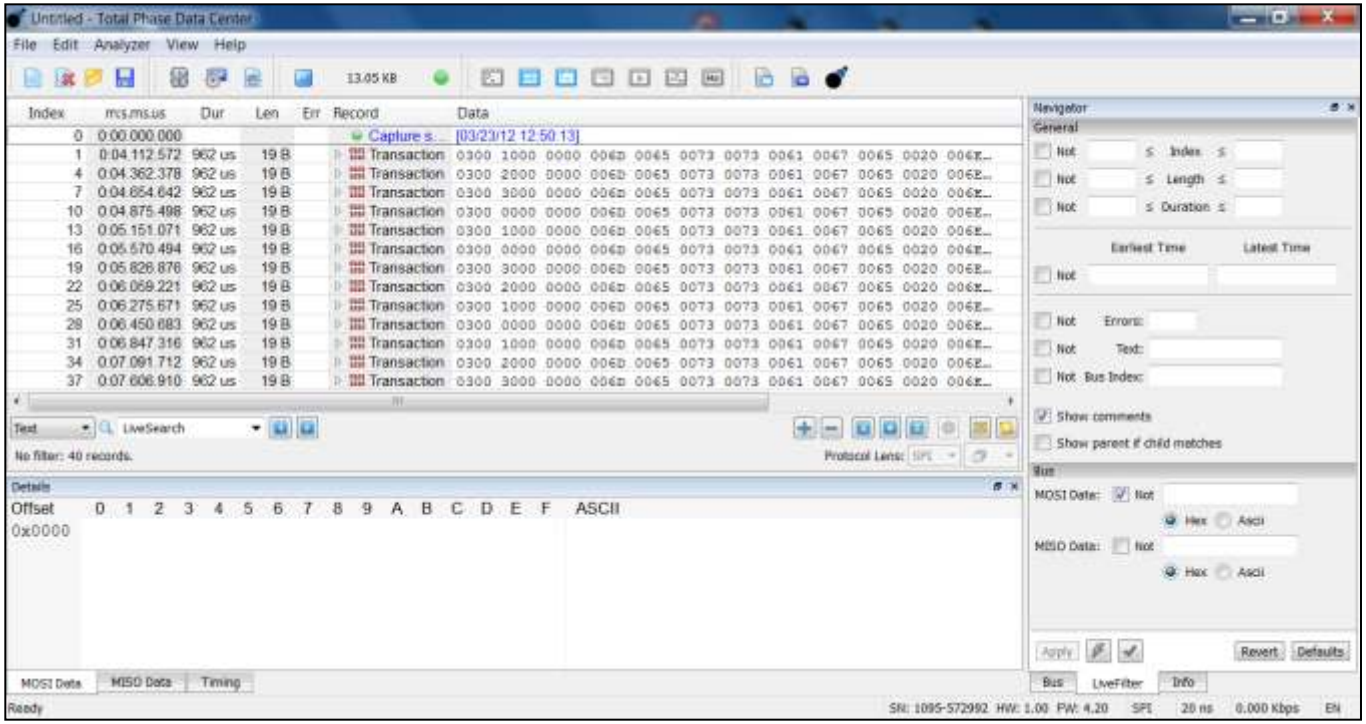


Bug 2: SPI Programming

Finding the SPI Bug

17. Click the buttons on the Kinetis Tower

18. Observe the behavior in the Data Center Software



Bug 2: SPI Programming

Debugging the SPI EEPROM

- SPI EEPROM size = 1 kilobyte
- Understanding read format

- Use the Data Center Software to debug

Bug 2: SPI Programming

Solution

- Address has the wrong endianness

13	0:05.151.071	962 us	19 B	Transaction	0300 1000 0000 006D 0065 0073 0073 0061 0067 0065 0020 006E...
14	0:05.151.071	962 us	19 B	MOSI	03 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
15	0:05.151.071	962 us	19 B	MISO	00 00 00 6D 65 73 73 61 67 65 20 6E 75 6D 62 65 72 20 31

Bug 2: SPI Programming

Solution

- Original code (in read_eeprom) in Line 84

```
// Message address
uint16_t addr = 0x10 * msg_index;
* (uint16_t) (&out_data[1]) = addr;
```

- Modified code

```
// Message address
uint16_t addr = 0x10 * msg_index;
out_data[1] = addr >> 8;
out_data[2] = addr & 0xff;
```

Feature: I2C LEDs

Feature: I2C LEDs

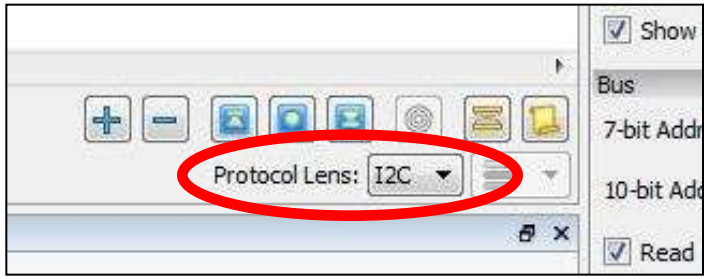
- **Objective**: Use the buttons on the Kinetis Tower to control the LEDs

- **Task**: Use the Total Phase tools to prototype the advanced feature for the I2C LEDs and then implement on the Kinetis Tower

Feature: I2C LEDs

Configuring the Data Center Software to run an I2C capture

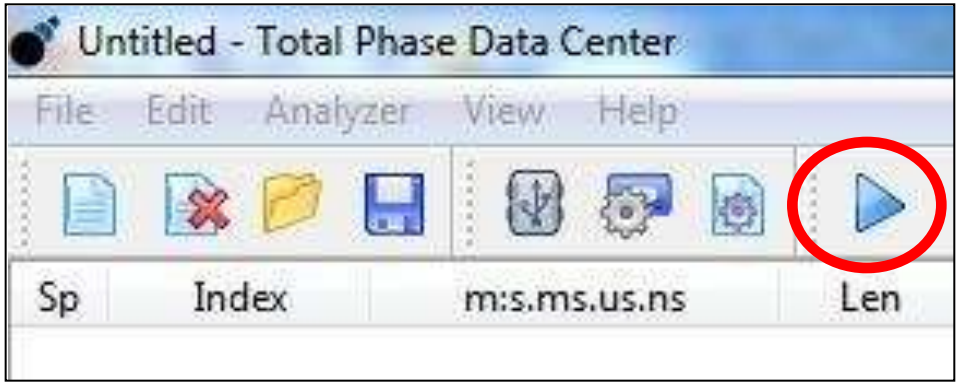
1. Go back to the Data Center Software running SPI
2. **Stop** the capture
3. **Clear** the transaction window
4. Change the **Device Settings** and **Protocol Lens** to **I2C**



Feature: I2C LEDs

Configuring the Data Center Software to run an I2C capture

1. Go back to the Data Center Software running SPI
2. **Stop** the capture
3. **Clear** the transaction window
4. Change the **Device Settings** and **Protocol Lens** to **I2C**
5. **Start** the capture



Feature: I2C LEDs

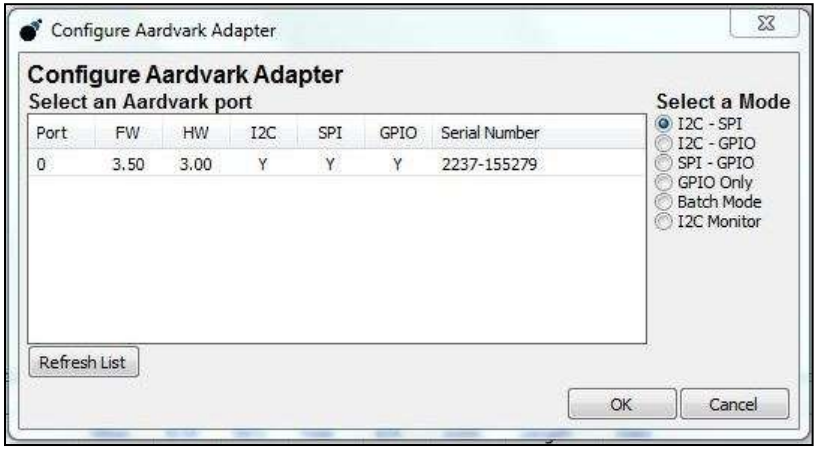
Configuring the Control Center Software for use

6. Close the **Flash Center Software**

7. Launch the **Aardvark_GUI.exe** to open the Control Center Software

8. Click on **Configure Aardvark Adapter** 

9. Select the device and click **OK**



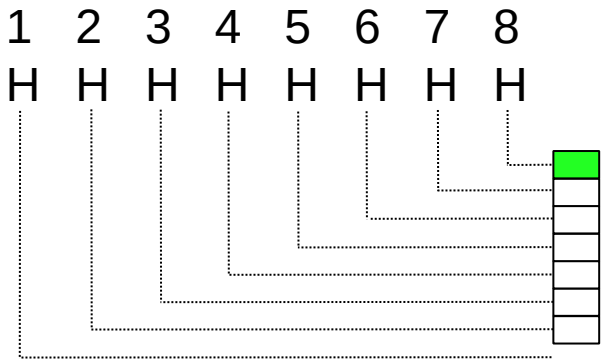
Feature: I2C LEDs

Understanding the I2C Lights

- Command structure for communicating with I2C LEDs

	Addr	Cmd	Data
Init:	0x38	0x03	0x00

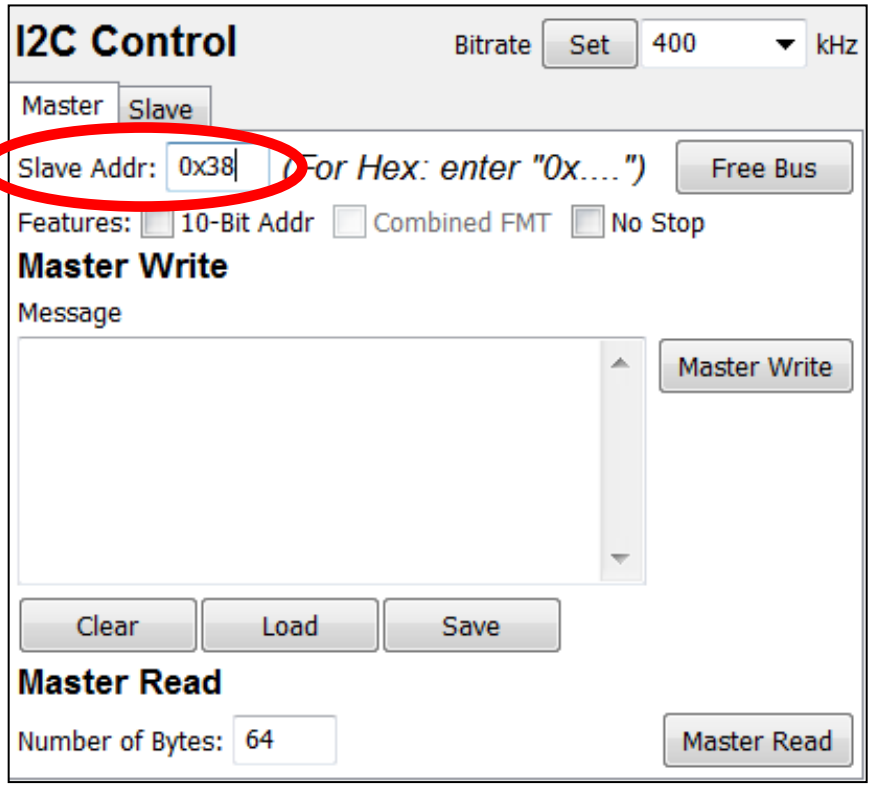
	Addr	Cmd	Data
Updating:	0x38	0x01	0xHH



Feature: I2C LEDs

Configuring the Control Center Software for use

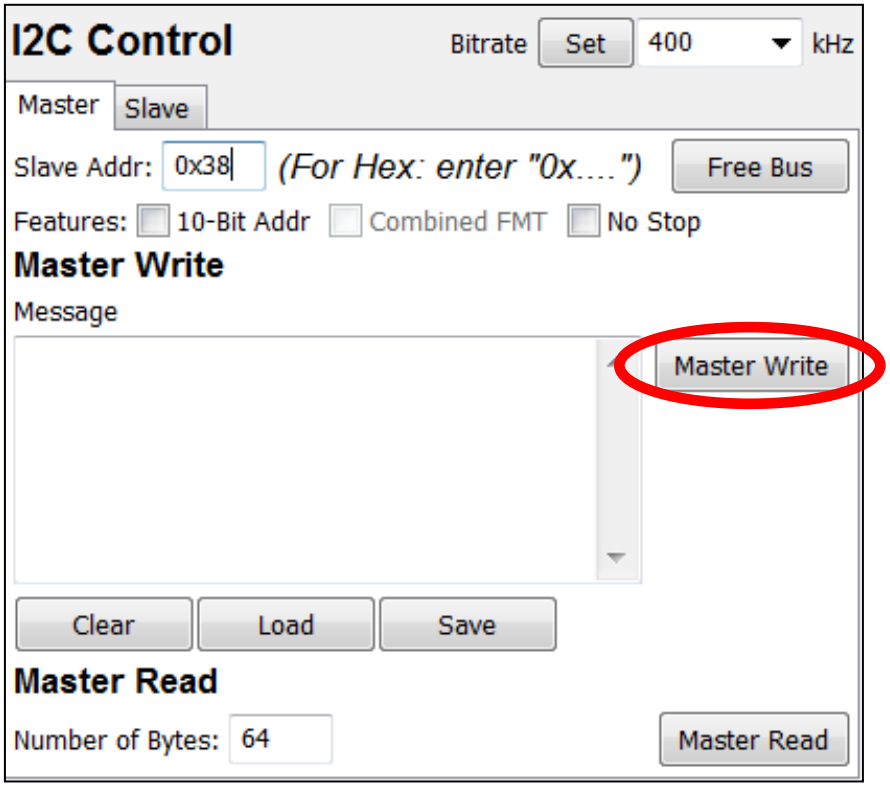
10. Type in the slave address 0x38



Feature: I2C LEDs

Initializing the LEDs

11. Type **03 00** in the message box.



12. Click **Master Write**

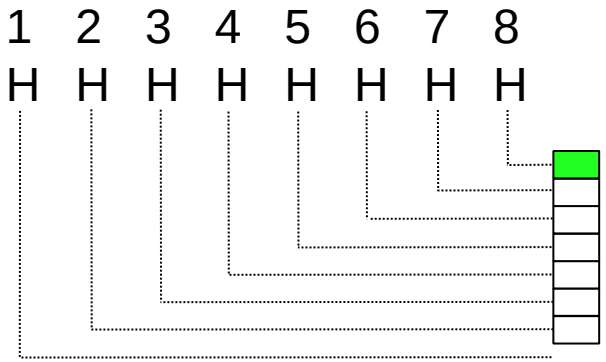
Feature: I2C LEDs

Understanding the I2C Lights

- Command structure for communicating with I2C LEDs

	Addr	Cmd	Data
Init:	0x38	0x03	0x00

	Addr	Cmd	Data
Updating:	0x38	0x01	0xHH



Feature: I2C LEDs

- Function to communicate with I²C slaves:
  - Call i2c_write (uint8_t slave_addr, uint8_t *data, int len);
- Implement:
  - static void init_leds()
  - static void set_leds(u08 index)
- Observe implementation with Beagle Data Center software

Feature: I2C LEDs

Solution

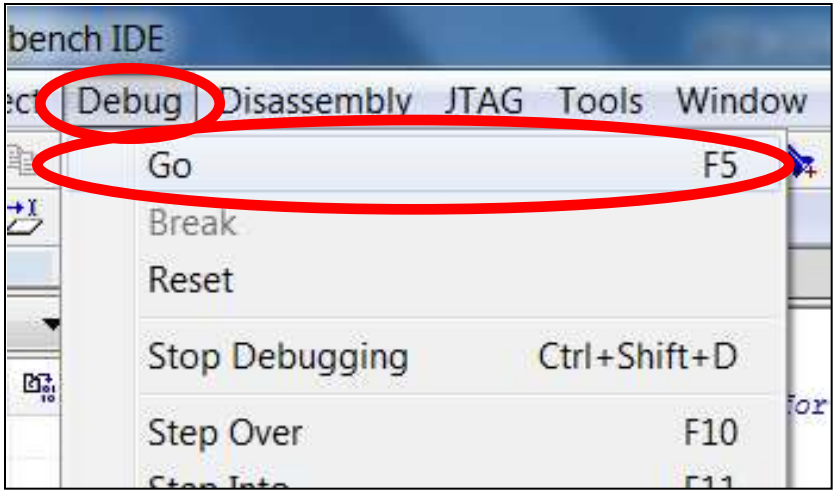
13. Implement init_leds and set_leds in main.c

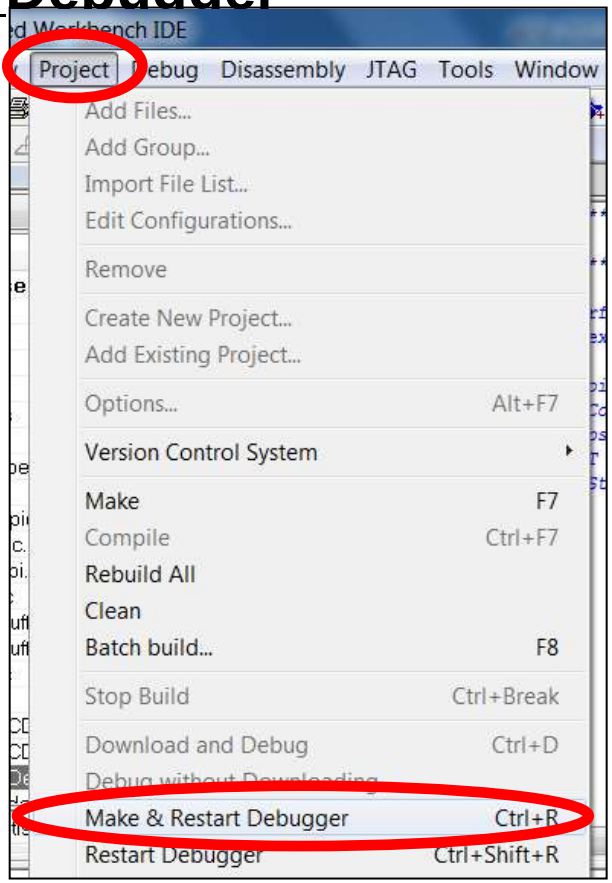
```
void init_leds () {
  uint8_t data[2];
  data[0] = 0x03;
  data[1] = 0x00;
  i2c_write(0x38, data, 2);
}

void set_leds (uint8_t val) {
  uint8_t data[2];
  data[0] = 0x01;
  data[1] = ~(0x01 << val);
  i2c_write(0x38, data, 2);
}
```

Feature: I2C LEDs

Verifying the Solution

14. Select **Project → Make_Restart_Debugger**
15. Select **Debug → Go**
16. Press Switch 3





Questions

Thank You

Don’t forget to fill out the evaluation form.


freescale™