



Stellaris[®] One Day Workshop 2010

Student Guide



*Revision 1.4
July 2010*



Important Notice

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Copyright © 2005, 2010 Texas Instruments Incorporated

Revision History

Version 1.0	November 2009	TTO release of workshop
Version 1.1	January 2010	Errata
Version 1.2	January 2010	Errata
Version 1.3	February 2010	Lab step errata
Version 1.31	February 2010	Lab step errata
Version 1.4	July 2010	New roadmap and portfolio slides

Mailing Address

Texas Instruments
Training Technical Organization
7839 Churchill Way, M/S 3984
Dallas, Texas 75251-1903

Stellaris[®] One-Day Workshop

Introduction

Welcome to the Texas Instruments Stellaris one-day workshop. This workshop has been segmented in order for it to be presented in a variety of formats: lunch and learn style, half day and full day.

If you are attending the lunch and learn format, this introductory presentation will be presented.

If you are attending a half-day format, you will also have the chance to get hands-on with Code Composer Studio, and the LM3S3748 and LM3S8962 evaluation kits.

If you are attending the full-day presentation, your afternoon will be a series of in-depth presentations and labs covering the USB, CAN and Ethernet peripherals.

Whichever format that you're here for, welcome to the class ... let's get started!

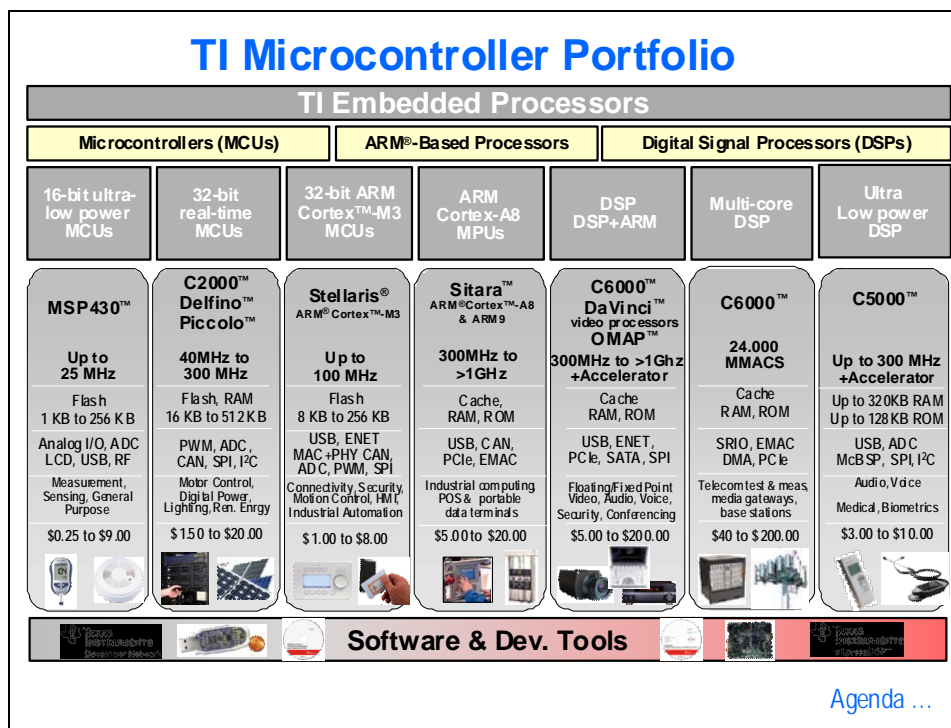
Objectives

- **Stellaris Microcontrollers Overview**
- **Stellaris Key Advantages**
- **Evaluation and Reference Design Kits**
- **Development Tools and Software Support**
- **Product Demonstrations**
- **Summary**

Module Topics

Stellaris® One-Day Workshop.....	1-1
<i>Module Topics.....</i>	<i>1-2</i>
<i>TI Microcontroller Portfolio.....</i>	<i>1-3</i>
<i>Stellaris ARM® Cortex™- M3 Overview</i>	<i>1-4</i>
ARM® Cortex™-M3 Benefits	1-4
No Assembly Required.....	1-5
First in ARM® Cortex™-M3 Microcontrollers	1-5
Roadmap.....	1-6
Family Technology.....	1-6
Internal Memories	1-7
Memory Protection Unit.....	1-7
Integrated MAC+PHY.....	1-8
NVIC	1-9
External Peripheral Interface	1-10
Battery-Backed Hibernation	1-10
Motor Control.....	1-11
<i>Evaluation and Reference Design Kits</i>	<i>1-12</i>
Motor Control Reference Design Kits.....	1-12
Reference Design Kits.....	1-13
Modules.....	1-13
Product Development Flexibility.....	1-14
<i>Development Tools and Support Software.....</i>	<i>1-15</i>
Stellaris Partners	1-15
StellarisWare	1-16
Available On-Line	1-16
Peripheral Driver Library	1-17
Graphics Library.....	1-17
USB Library	1-18
IEC 60730.....	1-18
In System Programming	1-19
Flash Programming GUI	1-19
ROM Enhancements.....	1-20
SAFERTOS	1-20
<i>Product Demonstrations</i>	<i>1-21</i>
CAN/Ethernet Demo	1-21
CNC Machine Demo	1-22
Autonomous Car.....	1-22
FIRST Robotics.....	1-23
<i>Summary</i>	<i>1-24</i>

TI Microcontroller Portfolio



Stellaris ARM® Cortex™- M3 Overview

What Is The ARM® Cortex™-M3?

The ARM Cortex family is comprised of three series

- **ARM Cortex-A Series**
 - Applications processors for complex OS and user applications.
 - Supports the ARM, Thumb and Thumb-2 instruction sets.
- **ARM Cortex-R Series**
 - Embedded processors for real-time systems.
 - Supports the ARM, Thumb, and Thumb-2 instruction sets
- **ARM Cortex-M Series**
 - Deeply embedded processors
 - Optimized for cost sensitive applications.
 - Supports the Thumb-2 instruction set only

Note:

- ARM Code 32-bit
- Thumb Code 16-bit
- Thumb-2 Code mostly 16-bit & some 32-bit (25% Faster, 26% Smaller)



Cortex™
Intelligent Processors by ARM®

For more information on the ARM Cortex-M3, see:

The Definitive Guide to the ARM Cortex-M3 by Joseph Yiu

ISBN: 978-0-7506-8534-4

Benefits ...

ARM® Cortex™-M3 Benefits

Cortex™-M3 Benefits

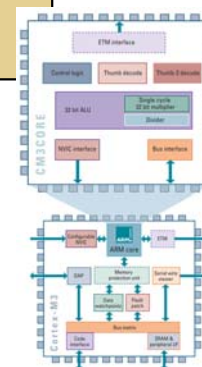
◆ **Capabilities beyond ARM7 for the MCU market:**

- **No Assembly Required**
- **Cortex-M3 requires approximately ½ the flash of ARM7 implementations**
- **2-4 times faster on MCU control applications**
 - Raw interrupt performance: we're 85% faster
 - PID (process control) main loop: we're 217% faster
 - Multiply-intensive code: we're 294% faster
 - Divide-intensive code: we're 726% faster



Features	ARM7TDMI	ARM Cortex-M3
Architecture	ARMv4T (von Neumann)	ARMv7-M (Harvard)
ISA Support	Thumb / ARM	Thumb / Thumb-2
Pipeline	3-stage	3-stage + branch speculation
Interrupts	FIQ / IRQ	NMI +1 to 240 physical interrupts
Interrupt Latency	24 - 42 cycles	12 cycles
Inter-Interrupt Latency	24 cycles	6 cycles
Sleep Modes	None	Integrated
Memory Protection	None	8 region MPU
Dhrystone	0.95 DMIPS/MHz (ARM) 0.74 DMIPS/MHz (Thumb)	1.25 DMIPS/MHz

Source: http://www.arm.com/products/CPUs/ARM_CortexM3.html


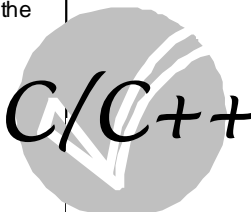


No ASM Required ...

No Assembly Required

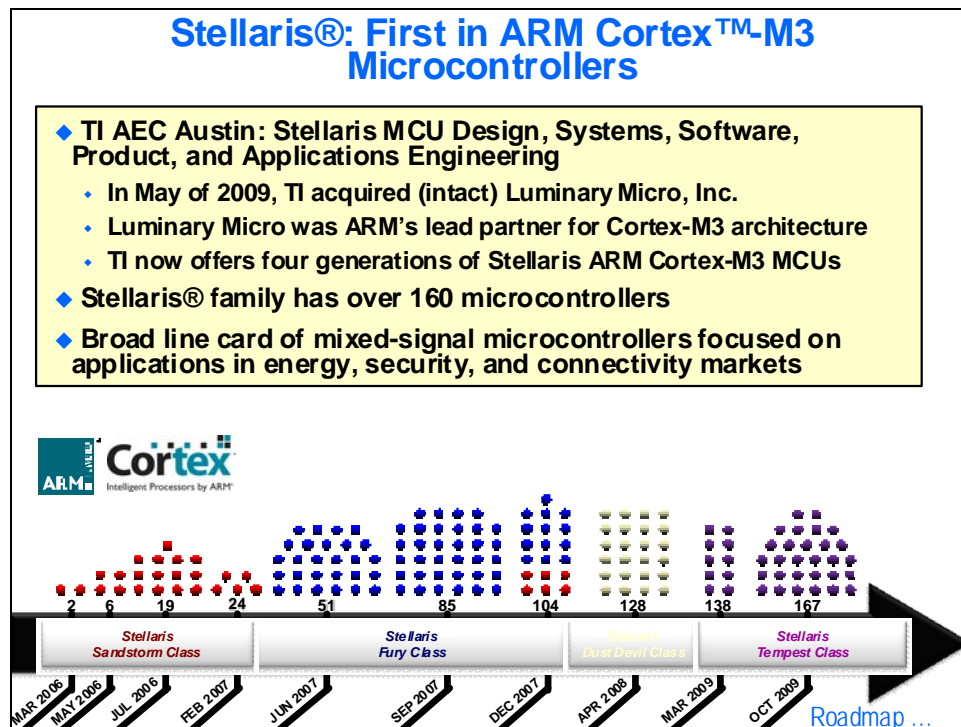
No Assembly Required

- ◆ **Cortex-M3 has complete hardware support for interrupts**
 - Interrupt Service Routines (ISRs) are written in C/C++
 - Interrupt setup is easily done in C/C++
 - C/C++ array which contains the vectors (pointers to the C/C++ functions)
 - Pointer to the stack (a C/C++ array)
- ◆ **No boot code ASM, no system configuration ASM**
 - ARM7 compilers normally come with an ASM boot routine (in object form) that does the setup.
 - For Cortex-M3, no boot routine is needed
 - Cortex-M3 hardware loads the stack pointer from memory and the initial PC from memory and enters as a normal C function.
 - User C/C++ code is all that is required.
- ◆ **Entire software code base can be written in C/C++**
 - ISRs
 - RTOS
 - Application code

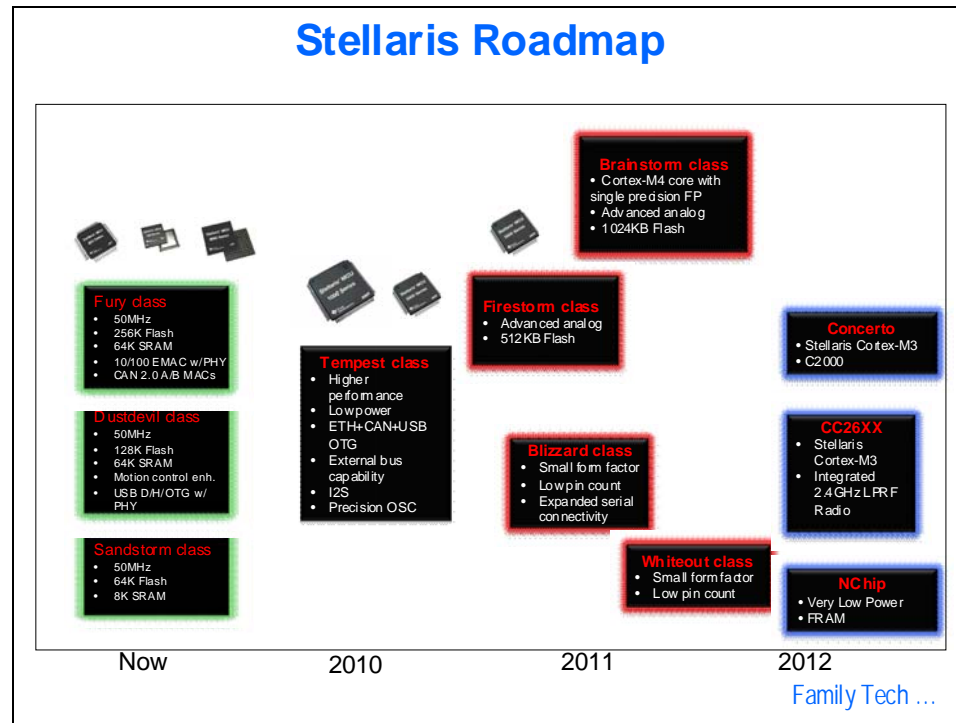



First in ARM ...

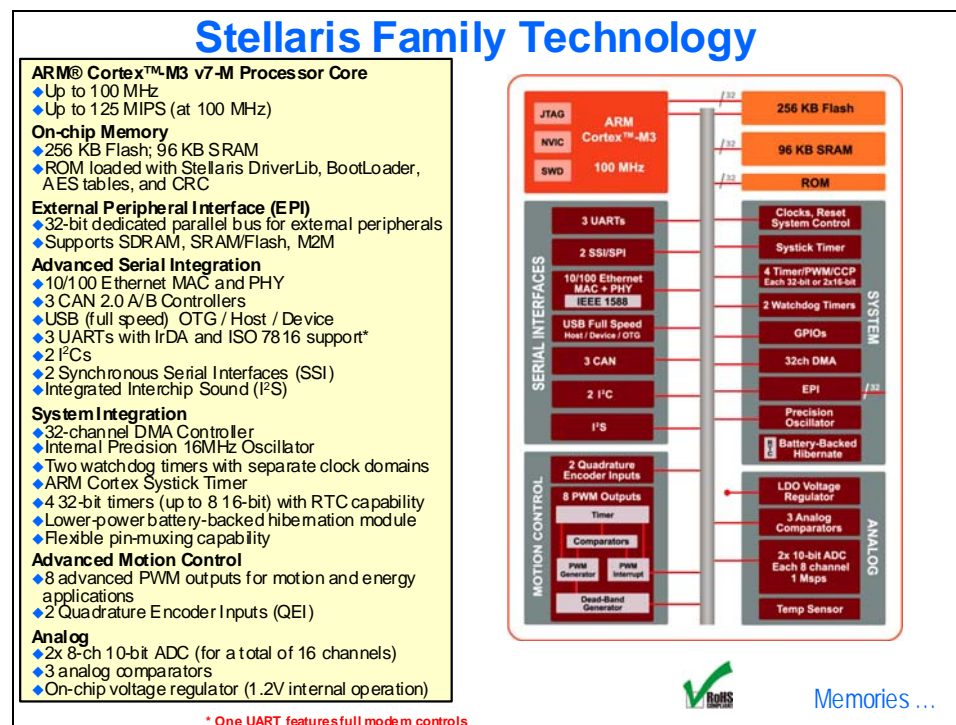
First in ARM® Cortex™-M3 Microcontrollers



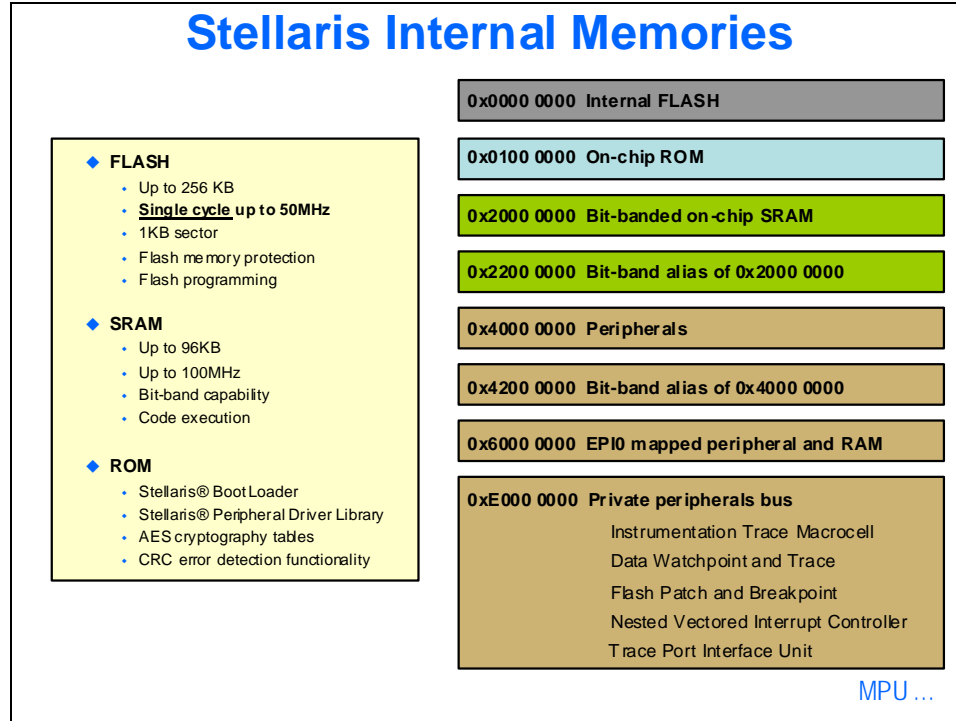
Roadmap



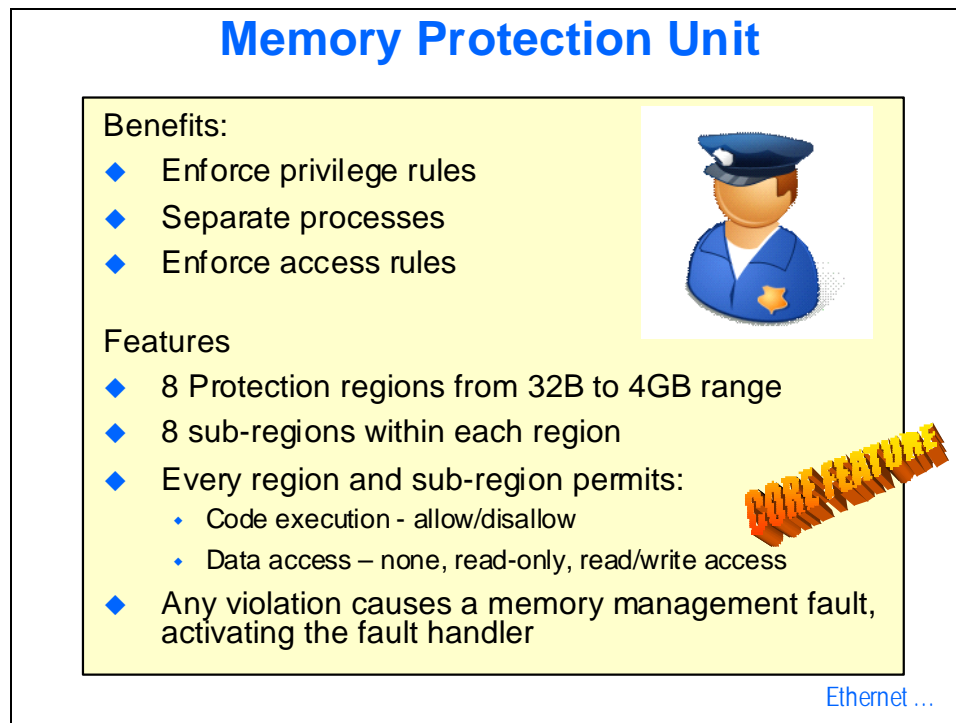
Family Technology



Internal Memories



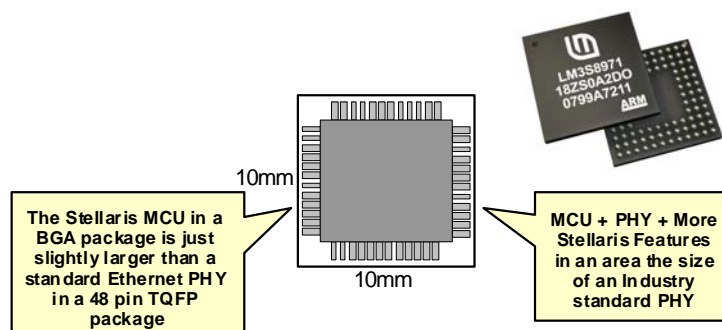
Memory Protection Unit



Integrated MAC+PHY

The Only ARM MCU w/ Integrated 10/100 Ethernet MAC+PHY

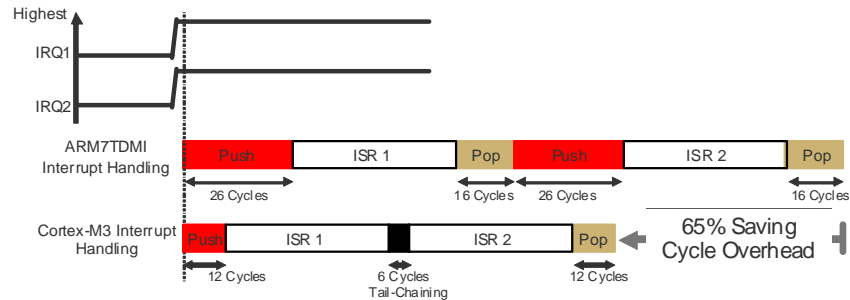
- ◆ Enables network connectivity and embedded web servers
- ◆ Lower external power budget requirements than solutions using an external PHY
- ◆ Savings in board space and system cost
- ◆ Hardware support for Precision Time Protocol (IEEE 1588 PTP)



NVIC ...

NVIC

Nested Vectored Interrupt Controller – Tail Chaining



ARM7TDMI

- 26 cycles from IRQ1 to ISR1 (up to 42 cycles if in LSM)
- 42 cycles from ISR1 exit to ISR2 entry
- 16 cycles to return from ISR2

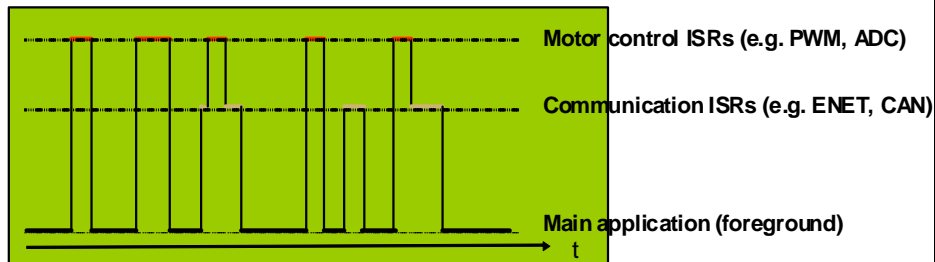
Cortex-M3

- 12 cycles from IRQ1 to ISR1 (Interruptible/Continual LSM)
- 6 cycles from ISR1 exit to ISR2 entry
- 12 cycles to return from ISR2

LSM = Load/Store Multiple instruction

CORE FEATURE
NVIC ...

NVIC Interrupt Priorities Example



- ◆ **Main application runs as foreground (base level)**
 - Easy to write since no “factoring” – normal application or RTOS based
 - Can use PLC style state-machine poll loop safely: ISRs keep data available
- ◆ **ISRs for Motor control are highest priority(ies)**
 - PWM, ADCs, Timer(s), Fault (may be highest), Temp sensor, etc
- ◆ **ISRs for communications below motor controls**
 - Ethernet, CAN, and/or serial
- ◆ **May use other priorities as needed**
 - Very fast interrupt response time, true nested interrupts, priority masking, easy ISR setup all contribute to making an easy solution
 - Application uses priority masking vs. interrupt-disable if needs critical region

CORE FEATURE

EPI ...

External Peripheral Interface

External Peripheral Interface (EPI)

◆ Multiple device types supported

- ◆ **SDRAM:** Supports x16 (Single Data Rate) at up to 50MHz
 - Supports low-cost SDRAMs up to 64 MB
 - Includes automatic refresh and access to all banks/rows.
 - Includes a sleep/standby mode to keep contents alive with minimal power draw.
- ◆ **Host-Bus Interface:** Traditional x8 MCU bus interface capabilities
 - Similar device compatibility options as PIC, ATmega, 8051, and others
 - Access to SRAM, NOR Flash, and other devices, with up to 24MB of addressing
 - Support of both muxed and de-muxed address and data
 - Access to a range of devices supporting the non-address FIFO x8 interface variant, with support for TXempty and RXfull
 - Speed controlled, with read and write data wait-state counters
 - Manual chip-enable (or use extra address pins)
- ◆ **Machine-to-Machine:** Wide parallel interfaces for fast communications
 - For instance, CPLDs and FPGAs
 - Data widths up to 32-bits, data rates up to 150 Mbytes/second
 - Optional "address" sizes from 4-bits to 16-bits
 - Optional clock output, read/write strobes, framing (with counter-based size), and clock-enable input

◆ Other features

- General parallel GPIO, FIFOed with speed control – for custom peripherals or digital controls
- Blocking and non-blocking reads
- FIFOed writes separate the processor from timing details
- Direct memory access (DMA)

Hibernation ...

Battery-Backed Hibernation

Battery-Backed Hibernation

◆ Battery-backed Hibernation Module (Standby current as low as 10µA*)

- ◆ **32-bit real-time counter (RTC)**
 - Programmable 32.768-kHz external oscillator or a 4.194304-MHz crystal
 - RTC software trim for making fine adjustments to the clock rate
- ◆ **256 bytes (sixty-four 32-bit words) of non-volatile battery-backed memory**
- ◆ **Power-switching logic to discrete external regulator (switch to battery)**
- ◆ **Low-battery detection, signaling, and interrupt generation**
- ◆ **Wake on RTC match and / or external pin**



- ◆ **On-chip Low Drop-Out (LDO) voltage regulator**
- ◆ **Low-power options on controller: Sleep and Deep-sleep modes**
- ◆ **Low-power options for peripherals: software controls shutdown of individual peripherals**
- ◆ **3.3-V supply brownout detection and reporting via interrupt or reset**

Operating Mode	Sandstorm Class	Fury Class	Dust Devil Class	Tempest Class*
Run	< 120 mA	160 mA (w/ETH)	120 mA	60 mA (w/o ETH) 80 mA (w/ETH)
Sleep	20 mA	20 mA (w/ETH)	20 mA	8 mA
Deep Sleep	700 µA	5 mA (w/ETH)	350 µA	600 µA
Hibernate	—	10 to 18 µA	10 to 18 µA	10 to 18 µA

* Preliminary

Motor Control ...

Motor Control

Motor Control

- ◆ Stellaris supports up to 8 general-purpose PWMs **and** up to 8 channels of motion control PWMs.

- ◆ **General-purpose PWMs**

- Stellaris 16-bit timer simple PWM mode with programmable output negation.

- ◆ **Motion-control PWM Module**

- Can generate simple PWM signals for a simple charge pump.
- Can generate paired PWM signals with dead-band delays for a half-H bridge driver.
- Can generate the full six channels of gate controls for a 3-Phase inverter bridge.
- Dead-band generator providing shoot-through protection.
- Synchronization of timers enables precise alignment of all edges.

- ◆ Up to 4 fault-condition handling inputs in hardware quickly provide low-latency shutdown.

- ◆ Up to 2 Quadrature Encoder Inputs provide accurate positioning for closed-feedback control.



Evaluation and Reference Design Kits

Evaluation Kits: “Zero-to-32bits” In 10 Minutes

- Everything a developer needs to get up and running in 10 minutes or less
 - Evaluation board(s)
 - All required cables
 - A choice of evaluation tools suites for popular development tools
 - Documentation
 - StellarisWare software
 - Applications notes
- Each kit functions both as an evaluation platform and as a serial in-circuit debug interface for any Stellaris microcontroller-based target board

							
E-K-LM3S811 Low pin count \$49	E-K-LM3S1968 High pin count \$59	E-K-LM3S2965 CAN Functionality \$79	E-K-LM3S3748 USB Host/Device \$109	E-K-LM3S6965 Ethernet MAC+PHY \$69	E-K-LM3S8962 Ethernet+CAN \$89	E-K-LM3S9B90 Ethernet+USB OTG \$99	E-K-LM3S9B92 Ethernet+OTG+MC \$99

Each kit comes in four versions:



And now ...


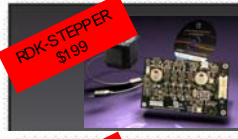




Order the kit you like, the other IDEs can be downloaded from
www.ti.com/Stellaris

Ref Des Kits ...

Motor Control Reference Design Kits

Open-Tool Motor Control Reference Design Kits

	AC Induction Motor Controller Design Example applications: <ul style="list-style-type: none"> • White goods • Residential and light commercial HVAC • 3-ph Industrial Motor Drives
	Stepper Motor Controller Design Example applications: <ul style="list-style-type: none"> • 2 and 3 axis CNC equipment • Sorting and grading equipment • Specialized printers and scanners
	Brushless DC Motor Controller with CAN/Ethernet Example applications: <ul style="list-style-type: none"> • Small appliances • Electric wheelchairs and mobility devices • Pumping and ventilation systems
	Brush DC Motor Controller with CAN Example applications: <ul style="list-style-type: none"> • Small appliances • Electric wheelchairs and mobility devices • Pumping and ventilation systems







Official FRST KoP Speed Controller – FRC 2009

Ref Des Kits ...

Reference Design Kits







Open-Tool Reference Design Kits

 <p>RDK-DM \$219</p>	<p>Touch-screen Intelligent Display Module with PoE</p> <p>Example applications:</p> <ul style="list-style-type: none"> • Security Systems & Building Access Controllers • White Goods and other Home Appliances • Factory Automation (System Status and Configuration)
 <p>RDK-DM-L35 \$219</p>	<p>Landscape-oriented Touch-screen Intelligent Display Module</p> <p>Example applications:</p> <ul style="list-style-type: none"> • Security Systems & Building Access Controllers • White Goods and other Home Appliances • Factory Automation (System Status and Configuration)
 <p>RDK-DM-SBC \$299</p>	<p>Stellaris 3.5" Landscape IDM Single Board Computer</p> <p>Example applications:</p> <ul style="list-style-type: none"> • Security Systems & Building Access Controllers • White Goods and other Home Appliances • Factory Automation (System Status and Configuration)
 <p>RDK-S2E \$139</p>	<p>Tiny Footprint Serial-to-Ethernet Module</p> <p>Example applications:</p> <ul style="list-style-type: none"> • SCADA Remote Terminal Units (RTUs) • Electronic Flow Meters (EFMs) • CCTV RS-232 Recorders

Modules ...

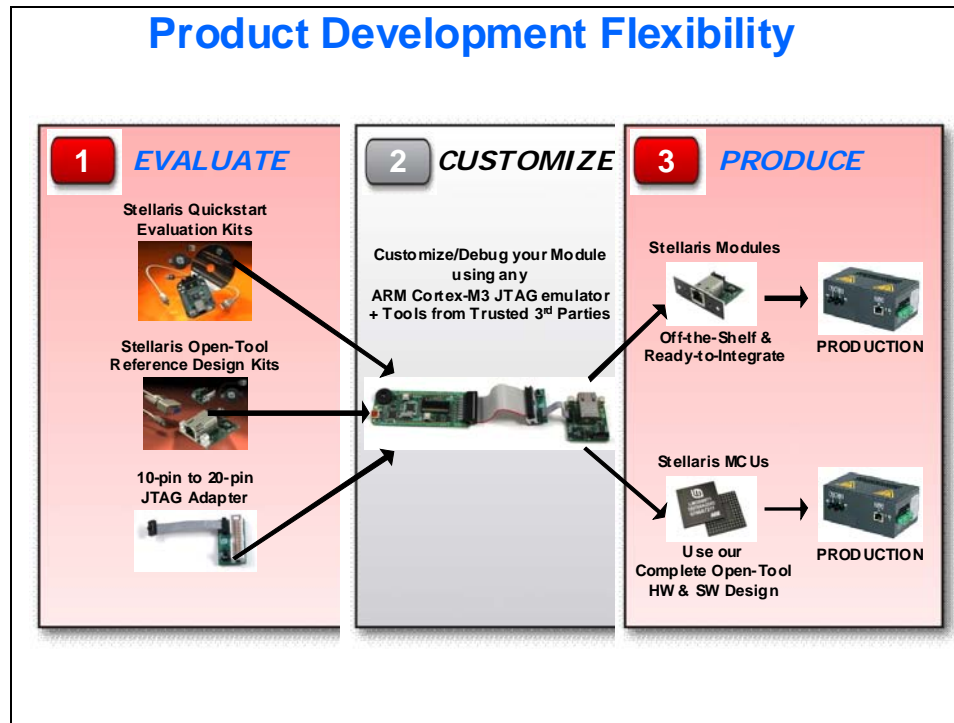
Modules

Open-Tool Modules Speed Time-to-Market

 <p>Brush DC Motor Control <i>MDL-BDC</i> Single unit: 109 USD</p>	 <p>Intelligent Display Module <i>MDL-IDM-L35</i> Single unit: 185 USD ...with Ethernet <i>MDL-IDM28</i> Single unit: 185 USD ...with PoE <i>MDL-IDM</i> Single unit: 199 USD</p>	 <p>Ethernet+CAN BLDC Motor Controller <i>MDL-BLDC</i> Single unit: 149 USD</p>
 <p>STEPPER Motor Control <i>MDL-STEPPER</i> Single unit: 169 USD</p>	 <p>Serial-to-Ethernet <i>MDL-S2E</i> Single unit: 49 USD</p>	 <p>AC Induction Motor Control <i>MDL-ACIM</i> Single unit: 239 USD</p>






Flexibility ...

Product Development Flexibility



Development Tools and Support Software

Development Tools for Stellaris MCUs

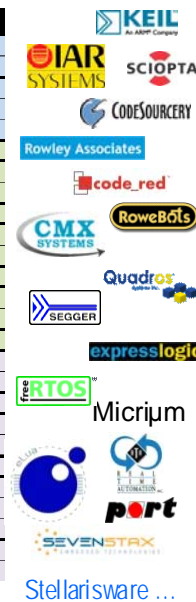
					
Eval Kit License	30-day full function. Upgradeable	32KB address-limited. Upgradeable	32KB address-limited. Upgradeable	90-day full function. Upgradeable	Full function. Onboard emulation limited
Compiler	GNU C/C++	IAR C/C++	RealView C/C++	GNU C/C++	TI C/C++
Debugger / IDE	gdb / Eclipse	C-SPY / Embedded Workbench	µVision	code_probe / Eclipse-based tool suite	CCS/Eclipse-based suite
Full Upgrade	199 USD personal edition / 3000 USD full support	2700 USD	MDK-Basic (256 KB) = €000 (2895 USD)	999 USD (upgrade to run on customer platform)	495 USD
JTAG Debugger		J-Link, ~299 USD	U-Link, ~199 USD	Red Probe, 150 USD	XDS510 / XDS560

Partners ...

Stellaris Partners

Stellaris Partners in Excellence

Product	Third Party	Description
Compiler / Debugger	Code Red	Red Suite (GNU C/C++ Compiler, code_probe / Eclipse Debugger / IDE)
	CodeSourcery	CodeSourcery G++ (C/C++ Compiler), GDB / Eclipse Debugger / IDE
	IAR	IAR C/C++ Compiler, C-SPY / Embedded Workbench Debugger / IDE
	Keil	RealView C/C++ Compiler, µVision Debugger / IDE
	Rowley	CrossWorks for ARM (C/C++ Compiler, CrossStudio Debugger / IDE)
RTOS	CMX	CMX-RTX™ RTOS offering small footprint, fast context switch times
	ExpressLogic	ThreadX advanced RTOS designed specifically for deeply embedded applications
	FreeRTOS.org	FreeRTOS.org™ Open-Source mini real-time kernel
	IAR	PowerPac™ fully featured RTOS combined with a high performance file system
	Keil	RTX flexible royalty-free RTOS with source code
	Micrium	Portable, scalable, preemptive real-time, multitasking kernel (RTOS)
	Quadraxis	RTXC for embedded applications
	RoweBots	Unison Ultra Tiny Embedded Linux and POSIX Compatible RTOS
	SCIOPTA	SCIOPTA real-time operating system for safety-critical applications
	SEGGER	embOS RTOS for embedded applications designed
Stacks / Specialty	CMX	CMX-USB Device, CMX-CANopen™, CMX MicroNet, and TCP/IP protocol stacks
	eLua	Embedded Lua Programming Language for Stellaris
	ExpressLogic	NetX™ TCP/IP and USBX™ supporting USB Host and Device
	Interneiche	NicheLite and ARM Network Evaluation Kits
	Micrium	µC/USB Device, µC/USB Host, µC/TCP-IP, µC/Modbus, µC/CAN protocol stacks
	MicroDigital	smxUSB Device, smxUSB Host, and smxUSB On-The-Go (OTG) Stacks
	port GmbH	CANopen Library for Stellaris Microcontrollers
	Quadraxis	RTXCusb Host and Device stacks, CANopenRT CAN stack, and QuadNet TCP/IP
	RTA Automation	RTA Automation DeviceNet™ protocol stacks
	SEGGER	embOS/IP TCP/IP and emUSB Device Stack
	SEVENSTAX	SEVENSTAX TCP/IP-Stack and Embedded Web Server



Stellarisware ...

StellarisWare

StellarisWare®

License-free and Royalty-free source code for TI Cortex-M3 devices:

- Peripheral Driver Library
- Graphics Library
- USB Library
- Boot Loader
- IEC 60730 Library
- Flash Programming
- On-Chip ROM Enhancements

On-line ...

Available On-Line

StellarisWare®

Available On-Line

The screenshot shows the StellarisWare website interface. The main navigation bar includes links for Products, Applications, Design Support, and Sample & Buy. The 'StellarisWare & Code Examples' link is circled in red. Below the navigation bar, the 'Microcontrollers (MCU)' section is visible, with a 'Stellaris ARM Cortex-M3-based MCUs' link also circled in red. The 'Complete Listing of StellarisWare Software' section is shown, with a 'StellarisWare Software' link circled in red. The 'StellarisWare Complete (all boards, all components)' status is shown as 'ACTIVE'.

Driver Lib ...

Peripheral Driver Library

StellarisWare® Peripheral Driver Library

- ◆ High-level API interface to complete peripheral set
- ◆ Free license and royalty-free use
- ◆ Simplifies and speeds development of applications
- ◆ Can be used for application development or as programming example
- ◆ Available as object library and as source code
- ◆ Compiles on ARM/Keil, IAR, Code Red, CCS and GNU tools
- ◆ Peripheral driver library functions are preprogrammed in ROM on select Stellaris MCUs



GrLib ...

Graphics Library

StellarisWare® Graphics Library

- ◆ Set of graphics primitives and widgets for use on Stellaris MCUs.
- ◆ Three subsequent layers of functionality:
 - Display Driver Layer
 - Graphics Primitives Layer
 - Widget Layer
 - Each API in each layer is directly callable
- ◆ Written entirely in C (except where not possible), self-contained, easy-to-understand, efficient.
- ◆ Compiles on ARM/Keil, IAR, Code Red, CCS and GNU tools.
- ◆ Computations that can be performed at compile time whenever possible.
- ◆ Graphics Primitives:
 - Point, Line, Rectangle, Circle, Font, Image, Context, Buffer
 - 134 Computer Modern predefined fonts available
 - Up to 24-bit color (~150 common colors conveniently referenced in GraphicsLib)
- ◆ Widgets:
 - Canvas, Checkbox, Container, Push Button, Radio Button, Slider, ListBox
- ◆ Special Utilities
 - *frasterize*: render your own font to be recognized by GraphicsLib
 - *lmi-button*: predefined button shape with shadow and 3-D
 - *pnmto*: Convert a NetPBM image file into a format recognized by GraphicsLib



USLib ...

USB Library

StellarisWare® USB Library Stacks and Examples

◆ **USB-IF Compliance**

- Stellaris has passed USB Device and Embedded Host compliance testing

◆ **Device Examples:**

- HID Keyboard
- HID Mouse
- CDC Serial
- Generic Bulk
- Audio class
- Device Firmware Upgrade
- Oscilloscope

◆ **Host Examples:**

- Mass Storage
- HID Keyboard
- HID Mouse

◆ **Windows INF for supported classes**

- Points to base Windows drivers
- Sets config string
- Sets PID/VID
- Precompiled DLL saves development time

◆ **Device framework integrated into USBLib**

VID Request for embedded USB products

FREE
Vendor ID/
Product ID
sharing
program

IEC60730 ...

IEC 60730

StellarisWare® Safe At Home With IEC 60730

IEC The International Electrotechnical Commission (IEC)

- IEC: World's authority in international standards for household appliances
- StellarisWare extension provides support for IEC 60730 Class B safety requirements
- Class B covers most home appliances, such as washers/dryers, refrigerators, freezers, and cookers/stoves
- Free license and royalty-free use for use on Stellaris MCUs
- Library supports both startup and periodic testing requirements of IEC 60730

<http://www.iec.ch/index.html>

	Module	Description
StellarisWare™ Software	Reset Handler	Performs basic register and memory test out of reset.
	CPU Test	Performs stuck bit testing on the CPU PC and registers.
	SRAM Test	Performs stuck bit testing on the SRAM.
	Flash Test	Performs a CRC test on the Flash.
	ADC Test	Performs a conversion test on an ADC channel connected to a known voltage reference.
	GPIO Test	Performs ADC temperature sensor test.
Stellaris® Hardware	Clock/Interrupt Test	Performs GPIO input/output plausibility test.
	Nested Vector Interrupt Controller	Performs tests to check the clock frequency, interrupt handling, and execution.
	Automotive-grade Flash Memory	Deterministic, fast interrupt processing for execution certainty.
	Cyclical Redundancy Check in ROM	High reliability non-volatile memory for robust environments.
	2 Watchdog Timers	Especially useful in verifying the contents of memory in a Stellaris microcontroller.
	Precision Oscillator	Clocked with precision oscillator, a second WDT takes advantage of the non-maskable interrupt (NMI) handler safety feature of the ARM Cortex-M3 processor.
	Advanced Motion Control with Multiple Fault Conditioning Inputs	Supplies an accurate, independent time base when periodic safety tests are executed.
	Quadrature Encoder Inputs	Provides quick motor shutdown in low latency situations.
	Integrated Analog Comparators	Provides precise, closed loop control of motors.
	Internal Temperature Sensor	Used to trigger Stellaris' accurate ADC and to trigger an interrupt when needed, which is useful for infrequent out-of-range events such as a current or voltage spike.
	10/100 Ethernet MAC/PHY with IEEE 1588 PTP	Eliminates the performance-wasting requirement of constant CPU polling.
	Controller Area Network (CAN) 2.0 MACs	Used to monitor and shut down an appliance if the appliance overheats.

Note: Watchdog timers are completely independent hardware timers In System Programming ...

In System Programming

StellarisWare® In System Programming Options

Stellaris Serial Flash Loader

- ◆ Small piece of code that allows programming of the flash without the need for a debugger interface.
- ◆ All Stellaris MCUs ship with this pre-loaded in flash
- ◆ Interface options include UART or SSI
- ◆ TI supplies a Windows™ application (GUI or command line) that makes full use of all commands supported by the serial flash loader (LMflash.exe)
- ◆ See application note [AN01242](#)

Stellaris Boot Loader

- ◆ Small piece of code that can be programmed at the beginning of flash to act as an application loader
- ◆ Also used as an update mechanism for an application running on a Stellaris microcontroller.
- ◆ Interface options include UART (default), I²C, SSI, Ethernet, USB
- ◆ Included in the Stellaris Peripheral Driver Library with full applications examples
- ◆ Preloaded in ROM on select Stellaris Microcontrollers

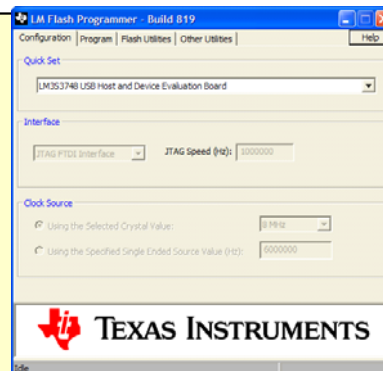
Flash GUI ...

Flash Programming GUI

StellarisWare® Flash Programming GUI

◆ LM Flash Programming GUI

- ◆ Simple graphical user interface
- ◆ Support for all Evaluation Kits
- ◆ Key features include:
 - Program
 - Verify
 - Erase
 - Read memory
- ◆ Available online
 - <http://focus.ti.com/mcu/docs/mcuorphan.tsp?contentId=87903>



ROM ...

ROM Enhancements


StellarisWare® On-Chip Software Enhancements (ROM)

StellarisWare® DriverLib

- ◆ High-level API interface to complete peripheral set.
- ◆ Simplifies and speeds development of applications.
- ◆ Saves user flash by storing peripheral setup and configuration code
- ◆ Allows programmer focus to be on the application—not setup

Other flash memory-saving options

- ◆ Advanced Encryption Standard (AES) cryptographic tables
 - Supported by the current AES example application
 - 128, 192 and 256-bits
- ◆ Cyclic Redundancy Check (CRC) functionality – for error detection



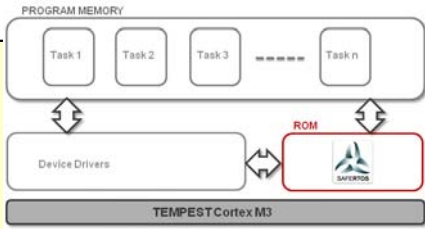
Stored in ROM on select Stellaris MCUs

SAFERTOS ...

SAFERTOS

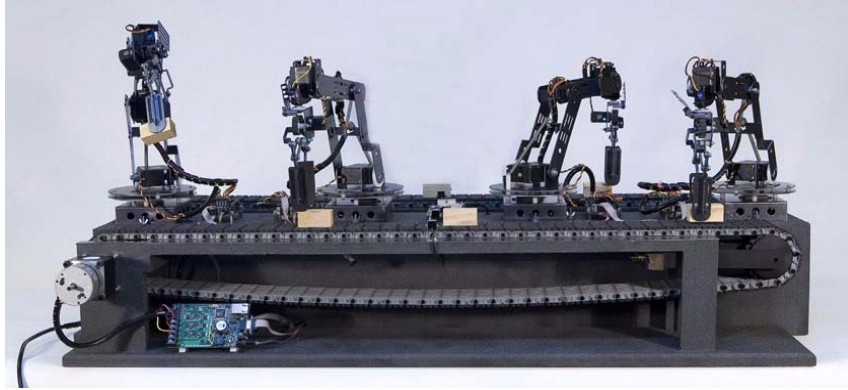
SAFERTOS Included On The LM3S9B96

- ◆ High-integrity RTOS in ROM
- ◆ Can be used as a standard operating system *OR* as part of a high integrity application which requires certification to **IEC61508** or **FDA510(k)**
- ◆ RTOS **value \$65k free** with Tempest LM3S9B96
- ◆ Integrated hardware/software solution shortens the time to market and significantly reduces cost for **Industrial** and **Medical** Applications
- ◆ Innovative *Design Assurance Pack* available separately from WITTENSTEIN provides **complete turnkey evidence** and process documentation



Product Demonstrations

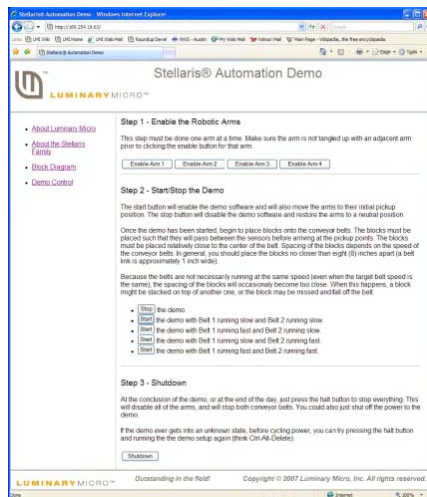
Stellaris CAN/Ethernet Automation Demo



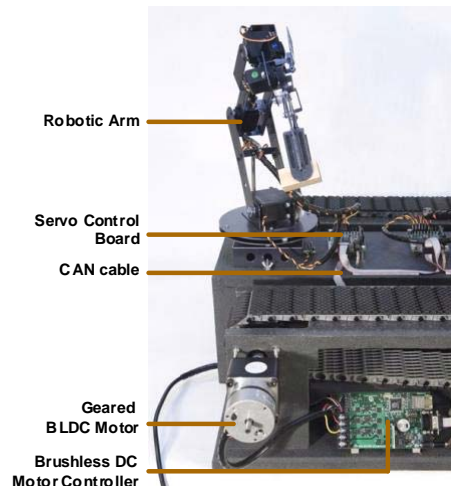
<http://www.youtube.com/watch?v=RyeUMx5cwSM>

CAN/Ethernet Demo

Stellaris CAN/Ethernet Automation Demo



Web Browser Console for Automation System Demo



<http://www.youtube.com/watch?v=RyeUMx5cwSM> CNC Demo ...

CNC Machine Demo

Stellaris 3-axis CNC Machine (AN01246)



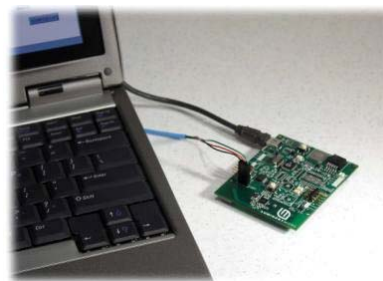
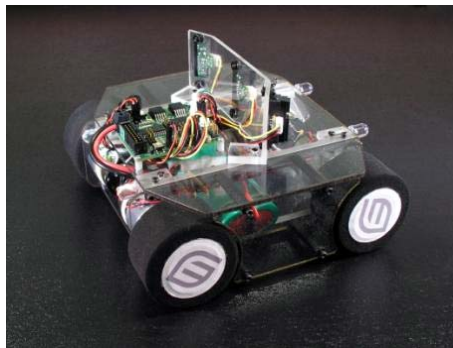
- ◆ **LM3S615 controls all three axes of stepper motion**
 - 6 advanced motion-control PWMs
 - Current sensing
 - Six limit switches
 - Active Indicator
 - Driver for tool control signals
 - Connectivity
 - CNC = Computer Numerical Control
- ◆ **LM3S316 controls QVGA LCD Touch Panel**

<http://www.youtube.com/watch?v=W8FpEJ5ZIY>

Car Demo ...

Autonomous Car

Stellaris Autonomous Car (AN01245)





- ◆ **Robot Uses one LM3S316 Stellaris MCU**
 - Four advanced motion-control PWMs drive four brushed motors
 - Four ADC channels for three infrared sensors and a bridge current monitor
 - Analog Comparator for photocell "nighttime" sensor
 - GPIOs for LED headlights
 - SPI for connection to 802.15.4 radio connection

<http://www.youtube.com/watch?v=M-7C7TIYJ8I>

FIRST ...

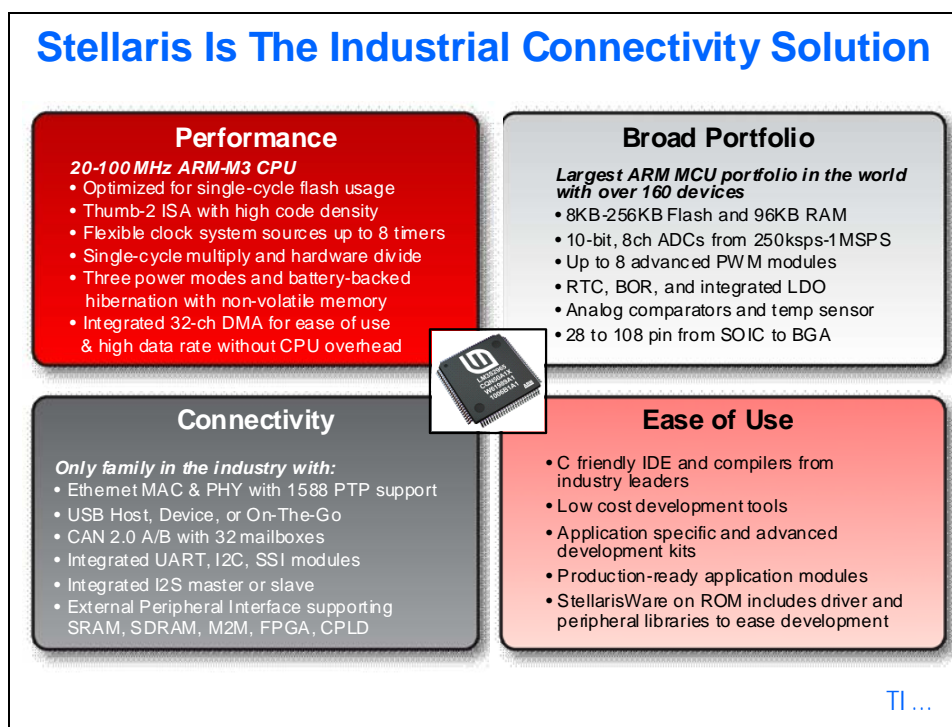
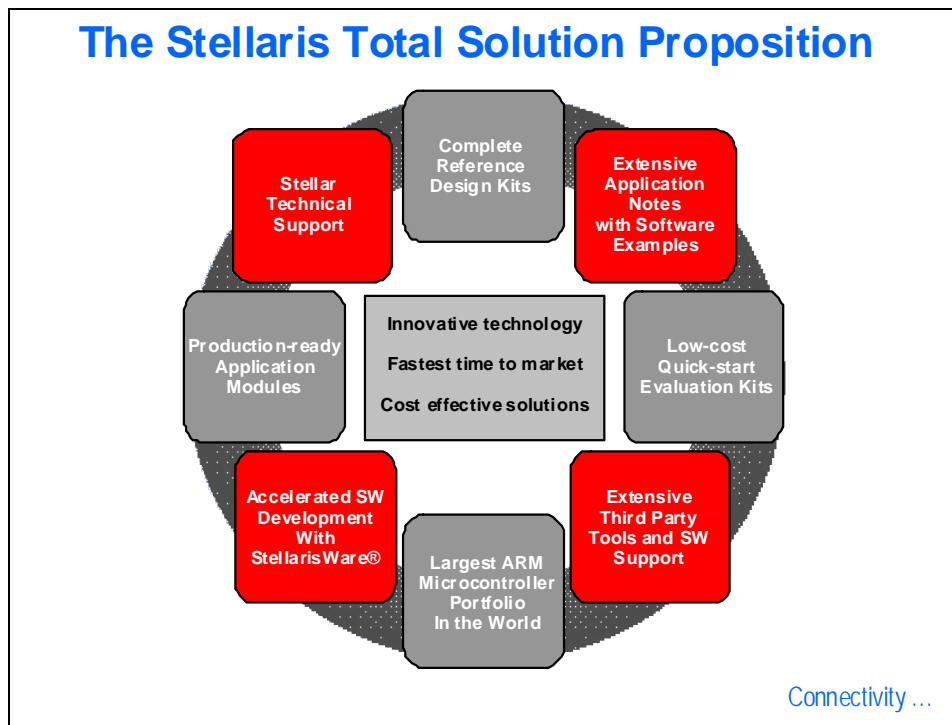
FIRST Robotics

FIRST Robotics Competition



- ◆ **FRC** is a worldwide robotics competition for high school students.
- ◆ FRC 2009 featured 1700 international teams.
- ◆ Stellaris MDL-BDC “Jaguar” selected as the Official Supplier of the speed controller in the FRC 2009 Kit-of-Parts delivered to each FRC 2009 team.

Summary



Introduction

In this section we'll have a chance to get hands-on with the LM3S3748 and LM3S8962 evaluation kits. So let's take a quick look at how these boards operate.

Objectives

- **Code Composer IDE QuickStart**
- **Flash Programmer**
- **LM3S8962 README FIRST**
- **LM3S3748 README FIRST**

Module Topics

Quickstart Labs	2-1
<i>Module Topics.....</i>	<i>2-2</i>
<i>LM3S3748 Evaluation Kit.....</i>	<i>2-3</i>
Oscilloscope Application.....	2-3
<i>LM3S8962 Evaluation Kit.....</i>	<i>2-4</i>
Web Server Maze Game Application	2-4
<i>Code Composer Studio 4.1.....</i>	<i>2-5</i>
Lab Procedure.....	2-5

LM3S3748 Evaluation Kit

LM3S3748 Evaluation Kit

- ◆ 50MHz LM3S3748 w/ 128K Flash & 64K SRAM
- ◆ Host and Device USB Connectors
- ◆ Bus or Self-powered USB Device support
- ◆ 128x128 Color LCD display
- ◆ microSD card slot
- ◆ Speaker with amplifier
- ◆ USB Flash memory stick
- ◆ USB Debugger interface
- ◆ Oscilloscope QuickStart application



[OS App ...](#)

Oscilloscope Application

LM3S3748 Evaluation Kit QuickStart App - Oscilloscope Demonstration -



Oscilloscope Wiring

Oscilloscope Demo

Oscilloscope Options



USB Host Mode
Data dump to the memory stick



USB Device Mode
Control the demo via a PC

[8962 ...](#)

LM3S8962 Evaluation Kit

LM3S8962 Ethernet+CAN Evaluation Kit

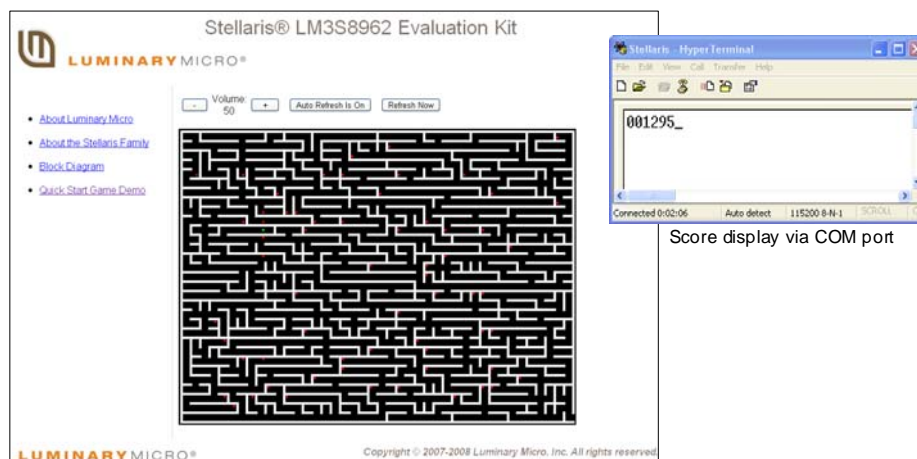
- ◆ LM3S8962 w/ Integrated 10/100 Ethernet controller and CAN MAC
- ◆ OLED graphics display
- ◆ Speaker
- ◆ microSD card slot
- ◆ USB Debugger interface
- ◆ Maze Game QuickStart application



OS App ...

Web Server Maze Game Application

QuickStart App for LM3S8962 Evaluation Kit – Web Server Maze Game -



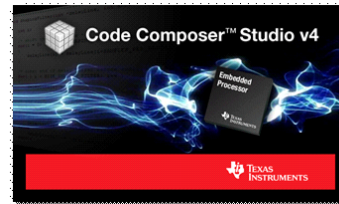
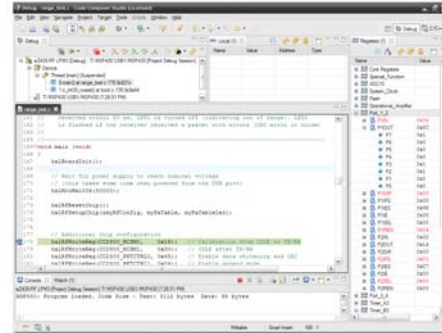
Game display on OLED and web browser

CCS ...

Code Composer Studio 4.1

Code Composer Studio v4.1

- ◆ **Code Composer Studio v4.1**
 - Platinum version supports Stellaris
 - Purchase MCU license for \$495
- ◆ **Can be used FREE with EVMs**
- ◆ **120 day full version eval period**
- ◆ **Eclipse based**
- ◆ **Includes**
 - Debugger
 - Profiler
 - Scripting
 - Image analysis and Visualization
 - C/C++ Compiler
 - Simulator



Procedure ...

Lab Procedure

Workshop Lab Procedure

- ◆ **If you have a 3748 board:**
 - Run the CCS QuickStart lab (chapter 3)
 - Reprogram the QuickStart App onto your board using the Flash Programmer procedure (chapter 4)
 - Run the 3748 README lab (chapter 5)
- ◆ **If you have a 8962 board:**
 - Run the CCS QuickStart lab (chapter 3)
 - Reprogram the QuickStart App onto your board using the Flash Programmer procedure (chapter 4)
 - Run the 8962 README lab (chapter 6)
- ◆ **Best case ... pair up with a partner who has the other board and see all the labs**

TI ...

*** Doodle Here ***

03 - QUICKSTART – CODE COMPOSER™ STUDIO

Stellaris® Development and Evaluation Kits for Code Composer™ Studio

The Stellaris Development and Evaluation Kits provide a low-cost way to start designing with Stellaris microcontrollers using Texas Instruments' Code Composer Studio development tools. The evaluation boards can function as either a complete evaluation target or as a debugger interface to any external Stellaris device.

Requirements

- You have a PC with a USB interface, running Microsoft® Windows XP (SP2 or greater) or Vista
- You have the Workshop Installation Software Flash Drive



CAUTION: There is a known electrical issue with the FT2232 device that is used in the on-board In Circuit Debug Interface (ICDI). Some USB hubs can cause the device to misbehave, with symptoms ranging from failed enumeration to corrupt data transfers. If you experience trouble when using the on-board ICDI, try connecting the USB cable directly to one of the USB ports on your PC or laptop.

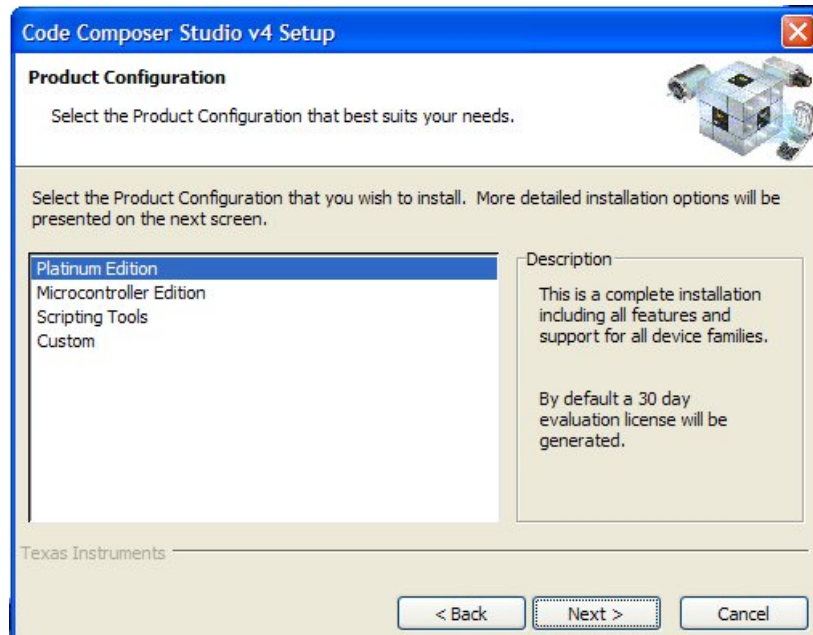
Code Composer Studio

This quickstart shows you how to install the Code Composer Studio development tool and how to use it to build and run an example application on your Stellaris Evaluation or Development Board.

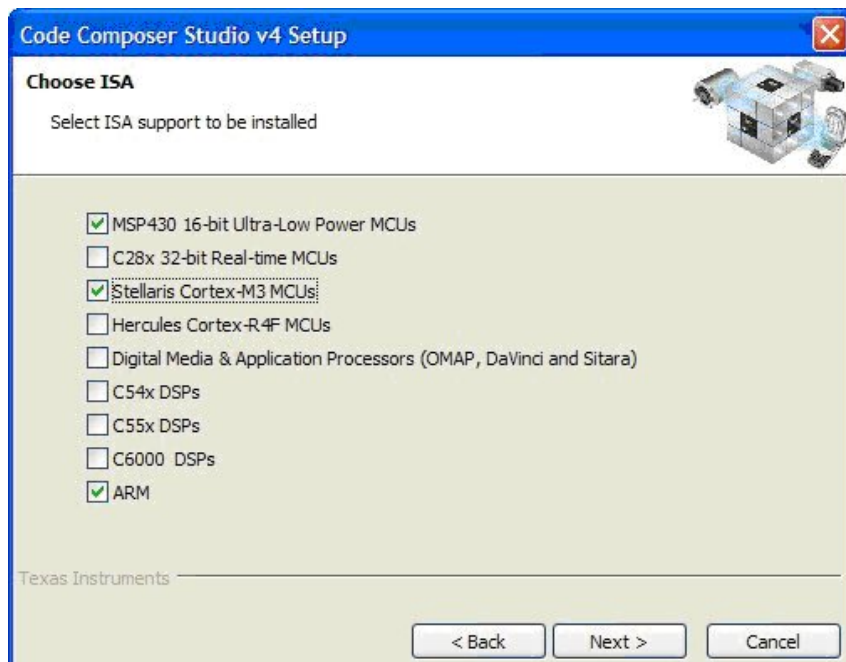
Step 1: Install Code Composer Studio

1. **Disconnect** any evaluation board that you have connected to your PC's USB port(s). **Insert** the Workshop Installation Flash Drive into a free USB port.
2. Using **Windows Explorer**, find the **setup_CCS_4.1** folder on the Flash drive and double-click on the file named **setup_CCS_n.n.n.n.exe**.
3. Follow the instructions in the Code Composer Studio installation program. Select the **Platinum Edition** for installation when the **Product Configuration** dialog window appears. Click **Next**.

QUICKSTART – CODE COMPOSER STUDIO

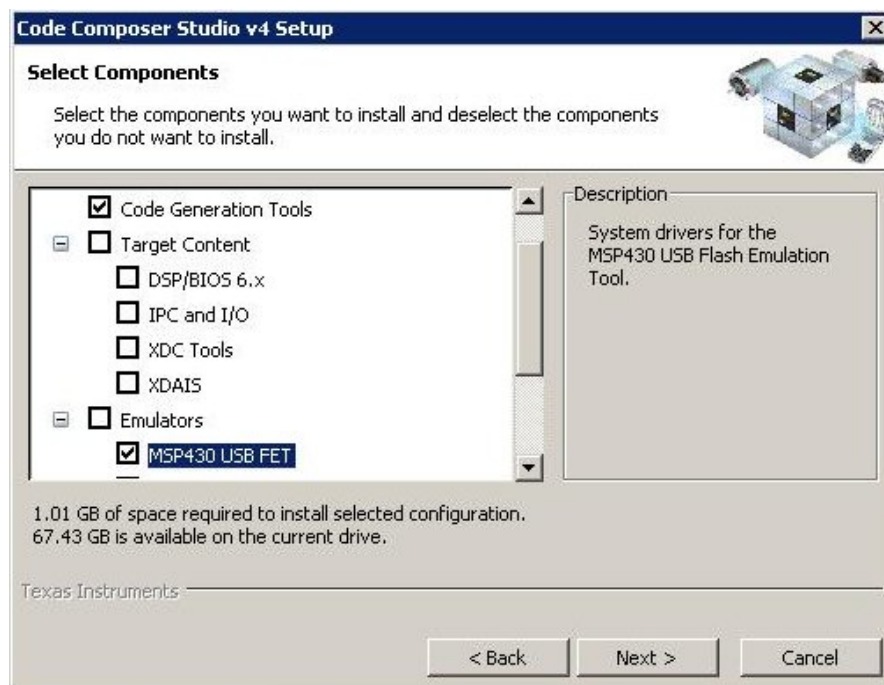


4. In the **Choose ISA** dialog, if you are attending a Stellaris only workshop, make sure that only the **Stellaris Cortex-M3 MCU** and **ARM** checkboxes are selected. If you are also attending an **MSP430** workshop, check that checkbox too. Click **Next**.



5. In the **Select Components** dialog, **uncheck** the **Target Content** and **Emulators** checkboxes. If you are attending a **Stellaris only** workshop, click **Next**. If you are attending a **MSP430** workshop too, check the **MSP430 USB FET** checkbox and click **Next**. The installation should take less than 10 minutes to complete.

QUICKSTART – CODE COMPOSER STUDIO



If you've been tasked with installing Code Composer only, please stop here and ask your instructor for further directions.

QUICKSTART – CODE COMPOSER STUDIO

Step 2: Install the StellarisWare® Package

A full set of C-based peripheral drivers is provided, covering all peripherals and functionality of the Stellaris devices. The StellarisWare package includes various example applications with project files for all major tool vendors that support Stellaris, including Code Composer Studio. To install StellarisWare components, follow these steps:

1. Make sure that the **Workshop Installation Flash Drive** is inserted into one of your PC's USB ports.
2. Using **Windows Explorer**, **open** the Flash drive and find the StellarisWare installation file that matches your board.
 - **LM3S3748 board** **SW-EK-LM3S3748-xxxx.exe**
 - **LM3S8962 board** **SW-EK-LM3S8962-xxxx.exe**

Double-click on the file for your board and select the **default** installation location when prompted. If you intend to run the labs for **both** boards, you will need the drivers and StellarisWare for **both** boards installed.

If you run a second (or more) StellarisWare installation, use the **default** installation directory. The board files will be installed in separate folders for each board. When you are warned about overwriting files, click **Yes to all**. All the overwritten files are the same.

NOTE: Check the www.ti.com/Stellaris web site for the latest software updates.

QUICKSTART – CODE COMPOSER STUDIO

Step 3: Start Code Composer Studio and Open a Workspace

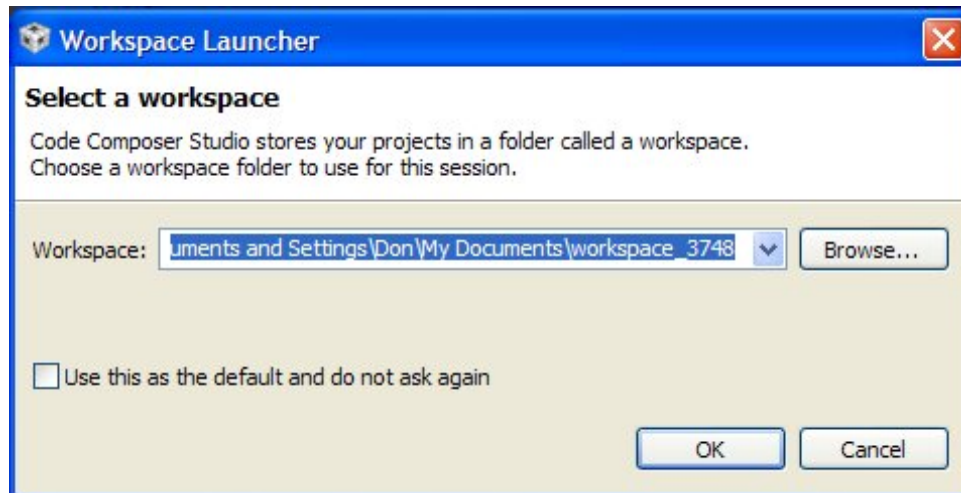
1. **Start** the **Code Composer Studio** IDE by selecting it from the Windows Start menu or double-clicking the icon installed on your desktop.

When the IDE loads, it asks you where to open the workspace folder. To keep your projects separated, you should use a workspace for each board.

If you are using the **LM3S3748** board, name your workspace **workspace_3748**.

If you are using the **LM3S8962** board, name your workspace **workspace_8962**.

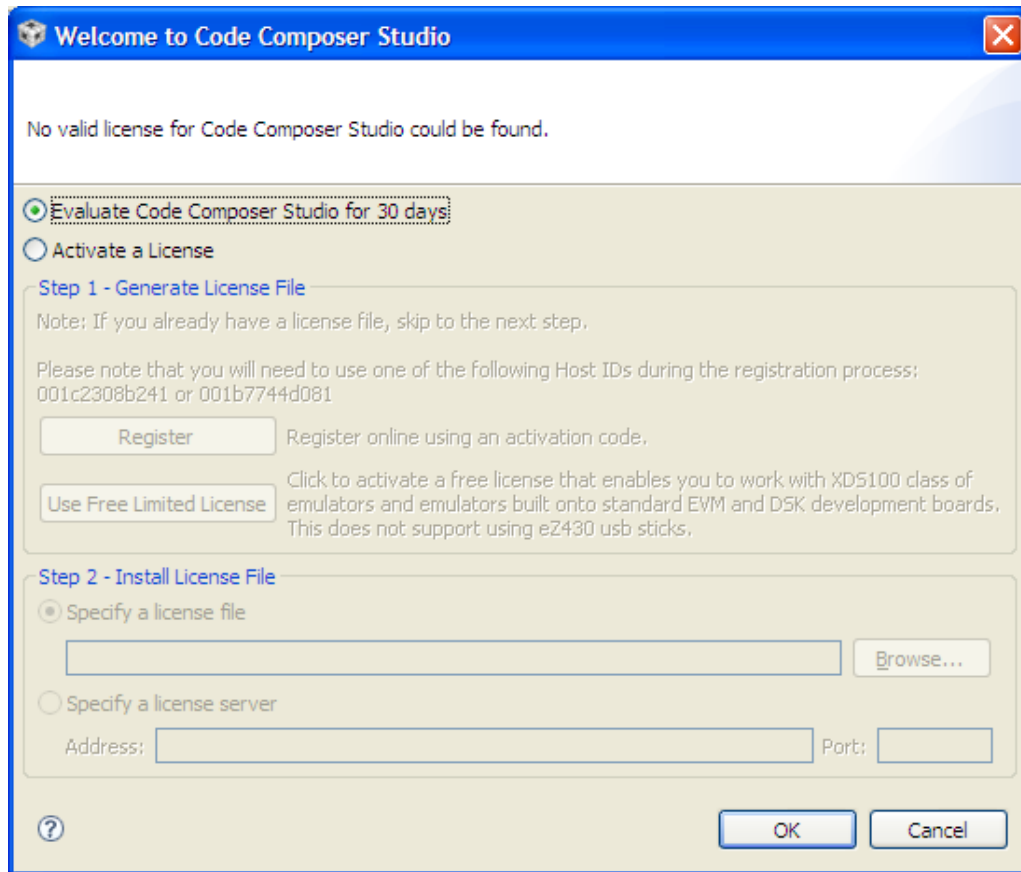
Do **not** check the **Use this as the default and do not ask again** checkbox. While it's possible to switch back and forth in the IDE, it's best to select your workspace at the start to prevent confusion. Click **OK**.



QUICKSTART – CODE COMPOSER STUDIO

2. If this is the first time you have run **Code Composer Studio IDE**, a dialog box may appear like the one shown below. If the dialog appears, select **Evaluate Code Composer Studio for 30 days** unless you already have a license that you wish to activate. Click **OK** to continue.

NOTE: If you have previously installed a Code Composer evaluation disk, you may not be able to move beyond this screen. Ask your instructor for assistance.

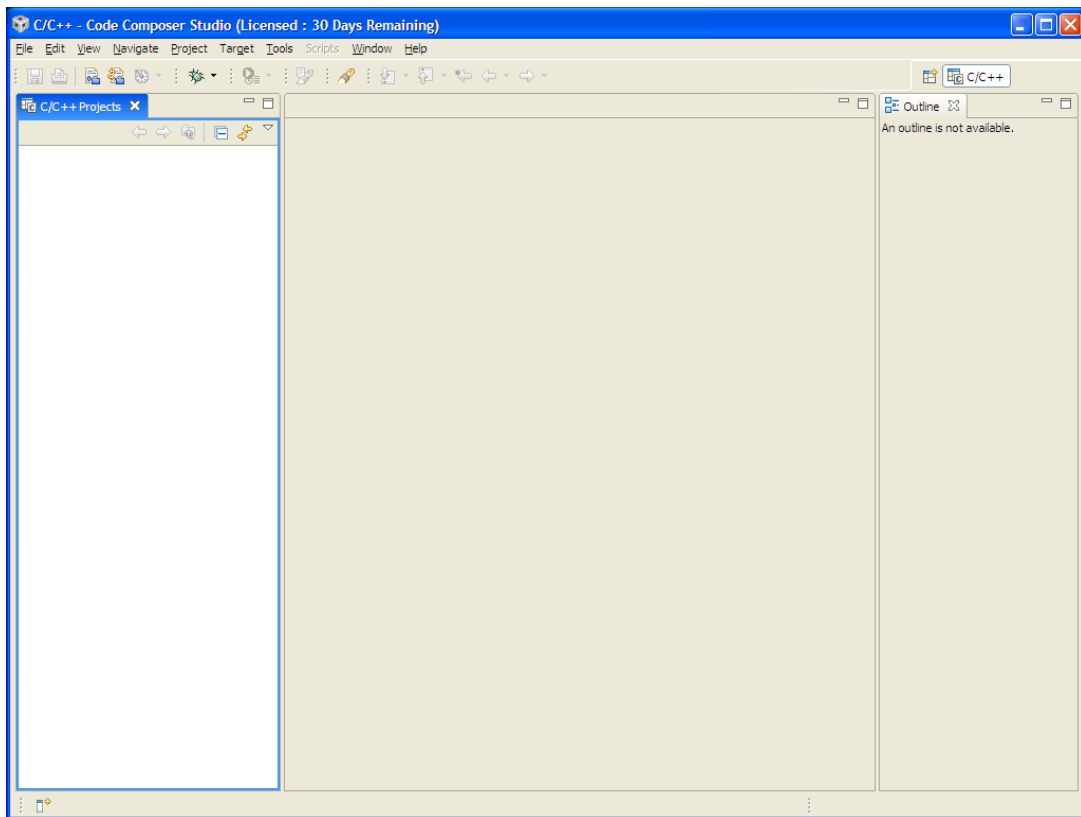


QUICKSTART – CODE COMPOSER STUDIO

3. **Code Composer Studio** may now open with the welcome page. If so, close out the welcome page by clicking the link in the upper right



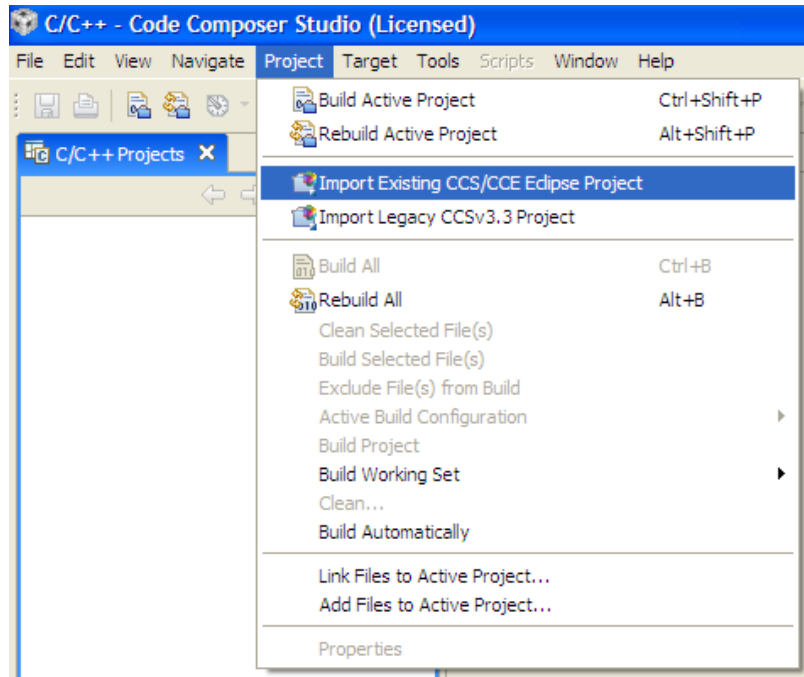
or by clicking the **X** on the **tab**. You should now have an empty workspace like that shown below. **Maximize** the window.



QUICKSTART – CODE COMPOSER STUDIO

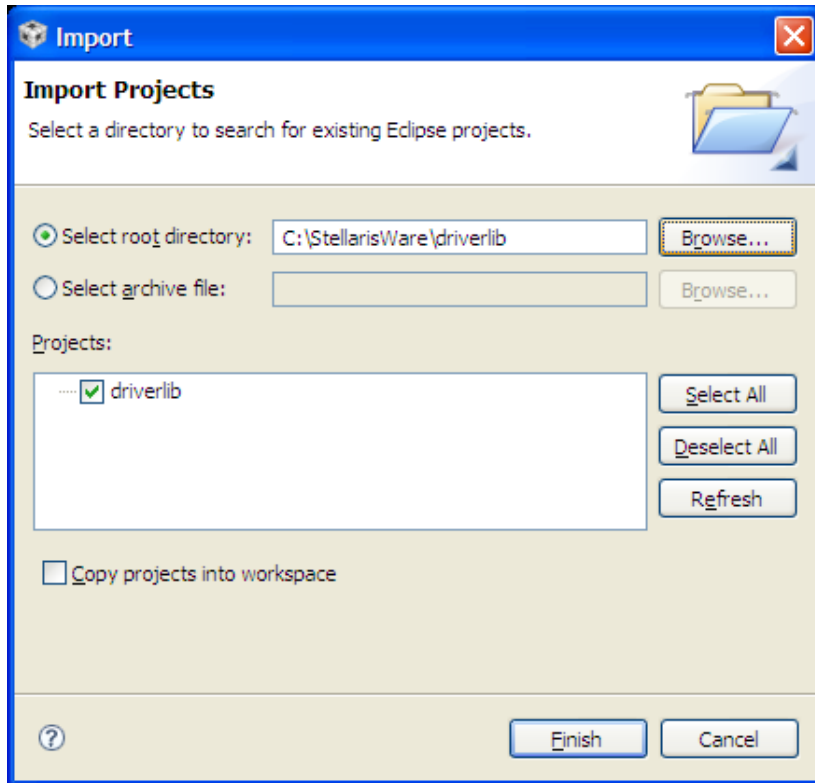
Step 4: Import Libraries

1. From the menu bar, click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**.



QUICKSTART – CODE COMPOSER STUDIO

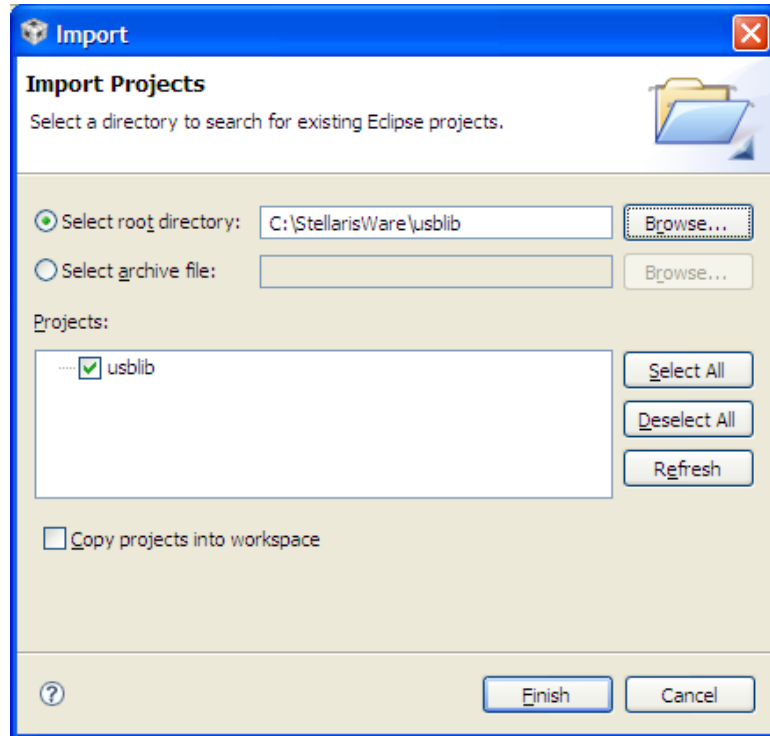
2. The **Import** dialog appears. Browse to the root directory of the driver library (C:\StellarisWare\driverlib) and click OK. Be sure that the checkbox next to **driverlib** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.



QUICKSTART – CODE COMPOSER STUDIO

3. Skip this step if you are using the LM3S8962 board.

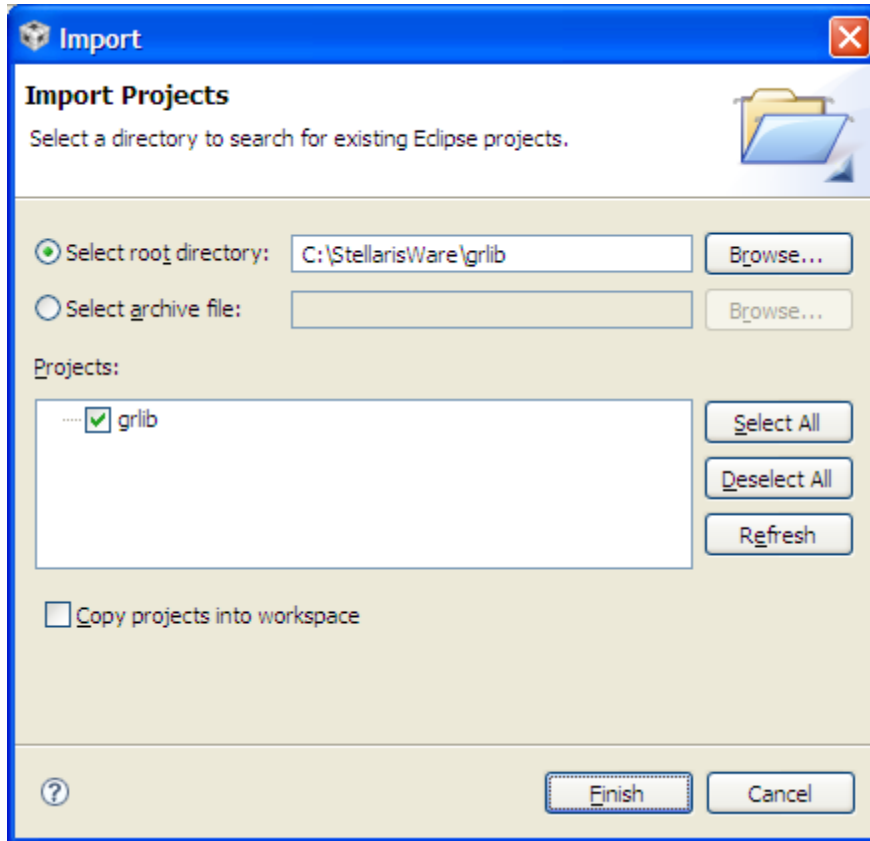
Select **Import Existing CCS/CCE Eclipse Project** from the **Project** menu again. Browse to the root directory of the USB library (**C:\StellarisWare\usblib**) and click **OK**. Be sure that the checkbox next to **usblib** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.



QUICKSTART – CODE COMPOSER STUDIO

4. Skip this step if you are using the LM3S8962 board.

Select **Import Existing CCS/CCE Eclipse Project** from the **Project** menu again. Browse to the root directory of the graphics library (C:\StellarisWare\gplib) and click **OK**. Be sure that the checkbox next to **gplib** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.



QUICKSTART – CODE COMPOSER STUDIO

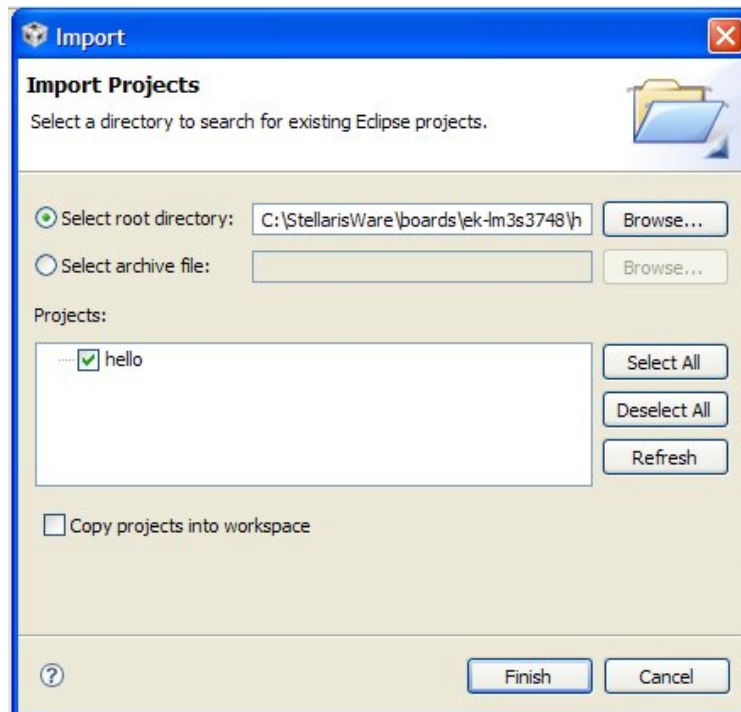
Step 4: Import Board Example

5. Select **Import Existing CCS/CCE Eclipse Project** from the **Project** menu again. Browse to the root directory for your chosen board ...

- **C:\StellarisWare\boards\ek-lm3s3748** for the **3748** board
- **C:\StellarisWare\boards\ek-lm3s8962** for the **8962** board

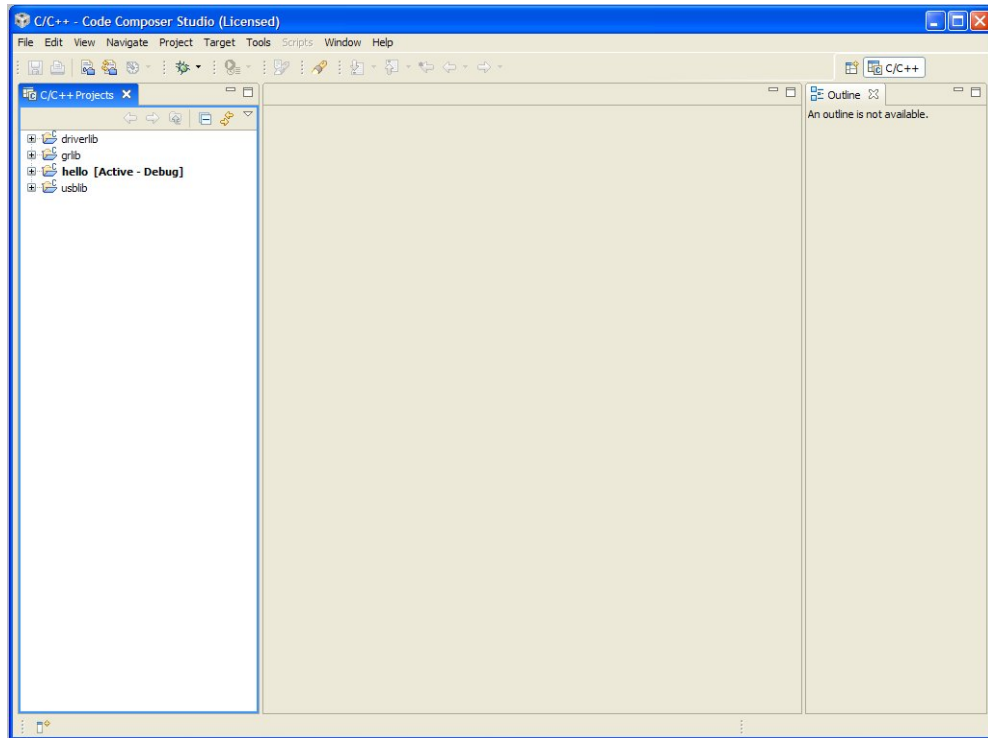
Browse to the root directory of the **hello** project and click **OK**.

The example screen shot below has the **EK-LM3S3748** board as the chosen board. (**C:\StellarisWare\boards\ek-lm3s3748**). Be sure that the checkbox next to **hello** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.



QUICKSTART – CODE COMPOSER STUDIO

6. All of the imported projects now appear in the **Projects Explorer** pane.

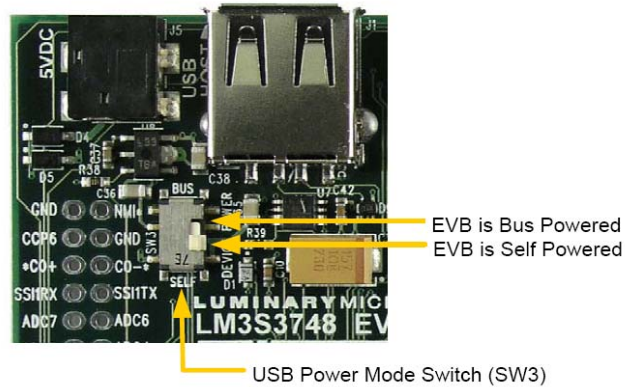


Since the **hello** project was the last one imported, it becomes the **Active Project**. Otherwise, you can right-click on it and set it as the **Active Project**.

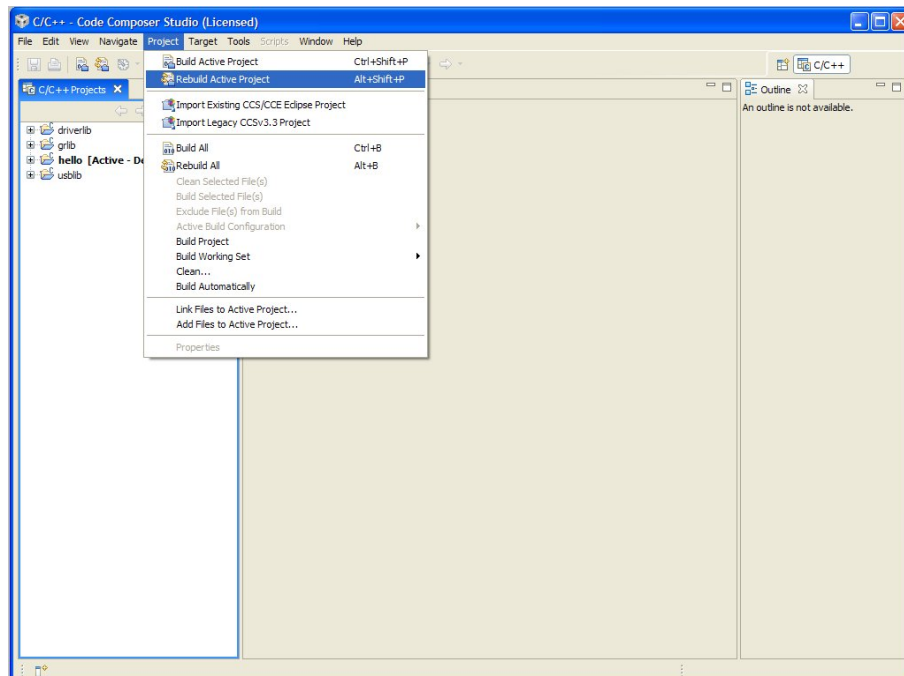
QUICKSTART – CODE COMPOSER STUDIO

Step 5: Building and Debugging a Project


1. Make sure that the **power switch** on your development board is in the **SELF** position, if your board has a switch. Then connect your development boards **DEBUG USB** port to your laptops USB port. Don't change the position of this switch while the board is powered as you may damage the switch's internal contacts. The **3748** power switch is shown below:

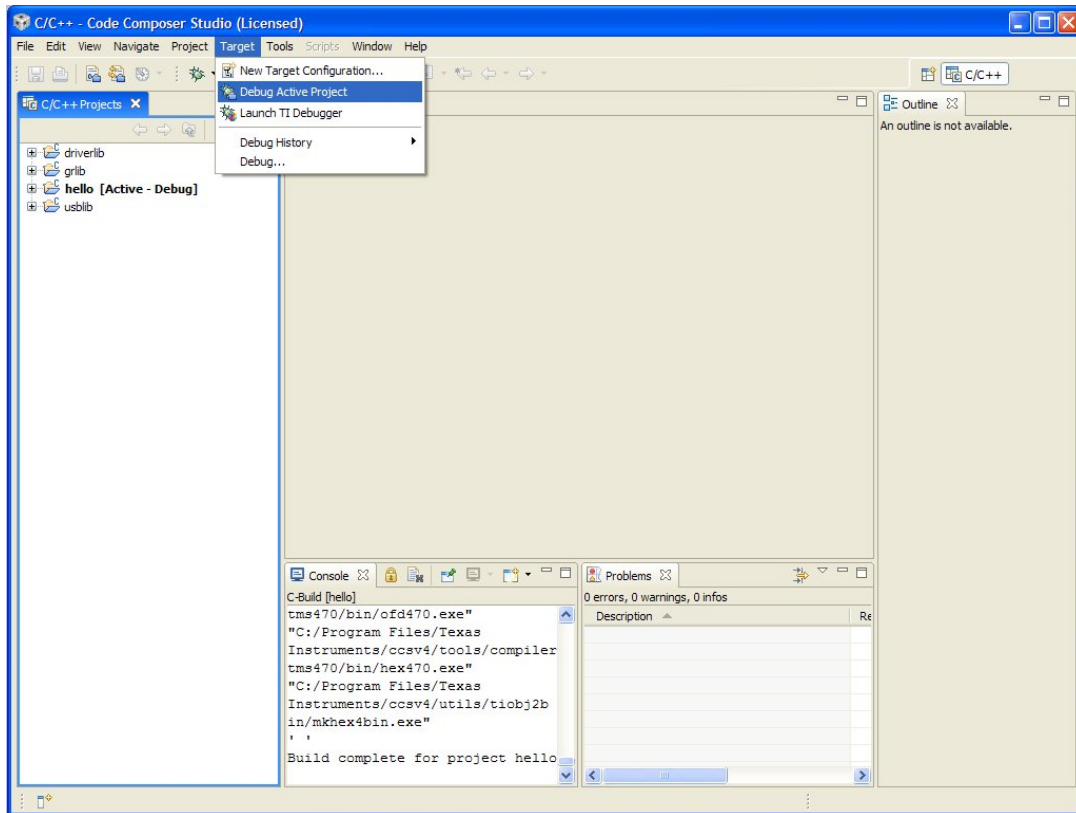



2. On the menu bar, click on **Project** and select **Rebuild Active Project**. The build may take a few moments. As the project builds, messages scroll by in the console window. When the build is complete, the words **Build complete for project hello** will appear in the console window.



QUICKSTART – CODE COMPOSER STUDIO

3. On the menu bar, click on **Target** and select **Debug Active Project**. Alternatively, you can simply click the **Debug Launch** icon  on the menu bar.




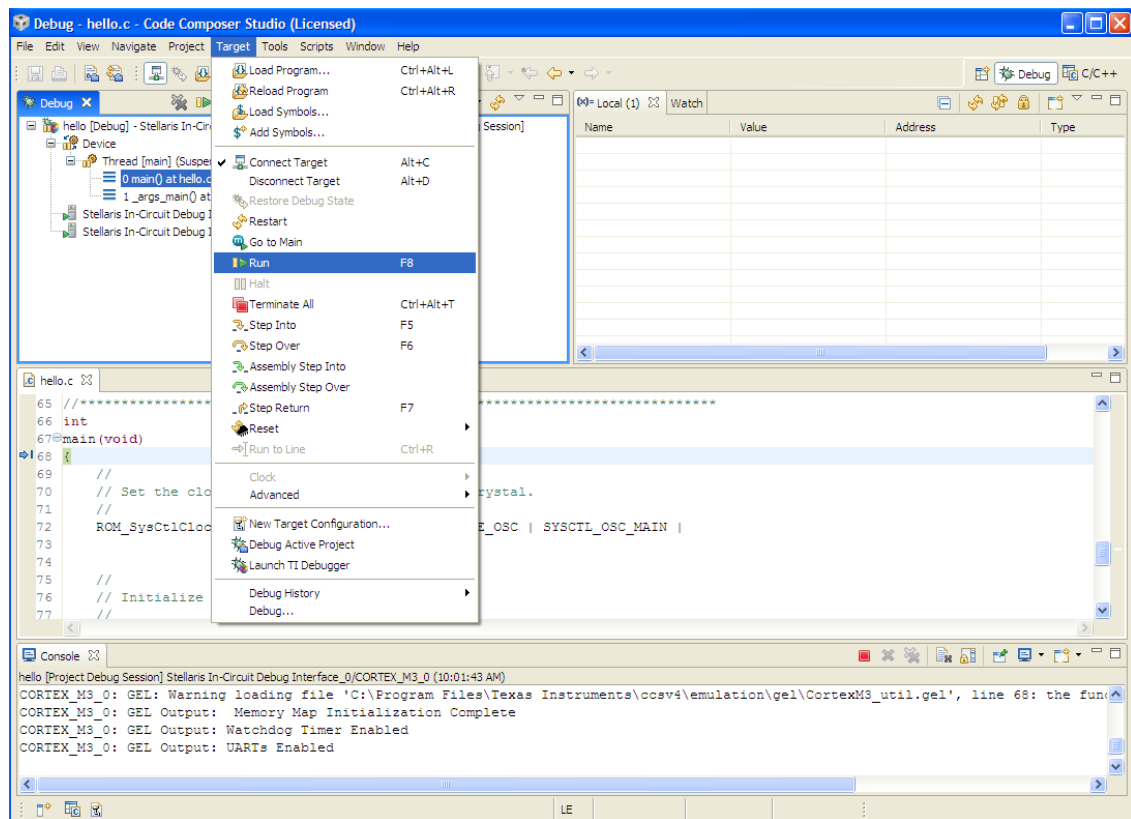
Code Composer Studio should switch to the **Debug Perspective**. If it does not, click the **Debug Perspective** button  in the upper right.

QUICKSTART – CODE COMPOSER STUDIO

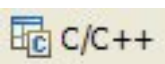
4. The **Code Composer Studio** debugger automatically connects to your evaluation board, programs the Flash memory, and runs to the beginning of the `main()` function. From here, you can examine and modify memory, program variables and processor registers, set breakpoints, step, and perform other typical debugging activities.

To run the program, select **Run** from the **Target** pull-down menu or click the

Run  button on the menu bar.



Note the display on your board. If everything has worked properly, you should see **Hello World** on the display. Don't worry about the preprogrammed application that came with the board, we'll re-flash it in the **Flash Programmer** section later.

Click the  **C/C++** **Perspective** button in the upper right of your display to return to the editor perspective.

Right-click on **hello** in the **Project** pane and select **Close Project** from the list.

If you are not going to run the next lab section, **Close** Code Composer Studio now.

QUICKSTART – CODE COMPOSER STUDIO

Optional!

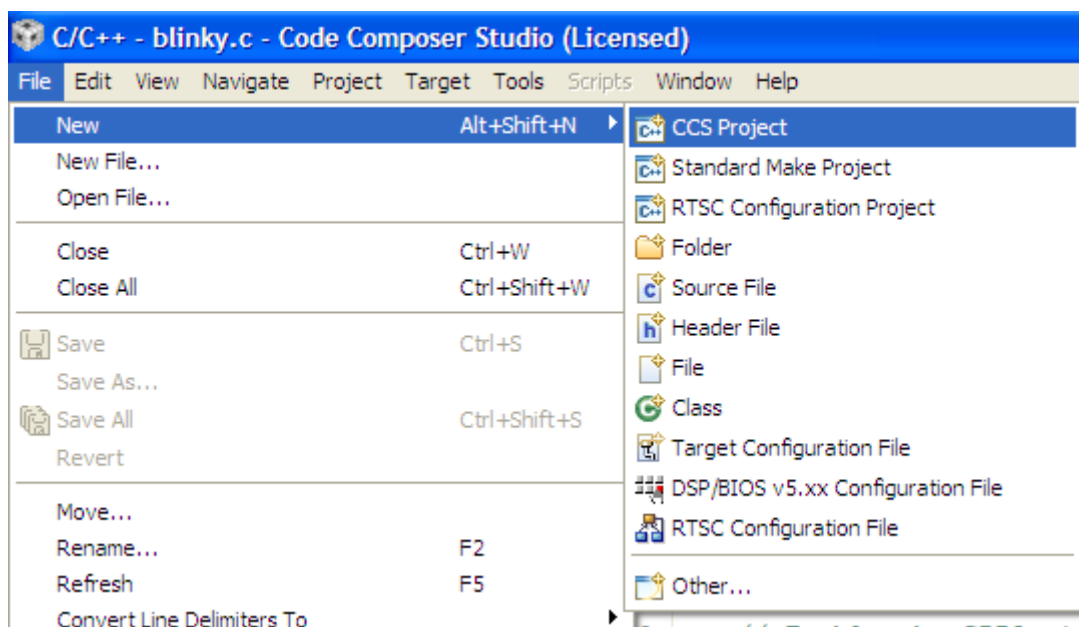
The following section should only be attempted if you have sufficient time to complete it. Let your instructor know that you're ready to start this section. Otherwise, you can complete it at home.

Creating a New Project

Once you have gone through the StellarisWare example applications, you may want to create your own project to begin development. While you can always start with an existing, simple project, sometimes you may want to start fresh.

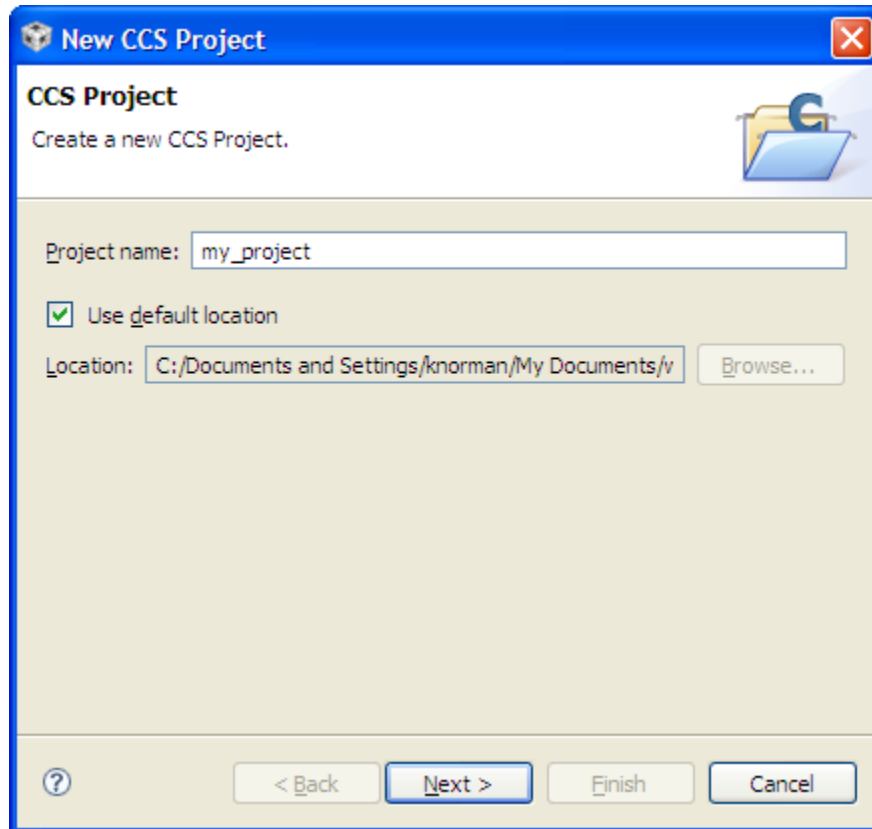
The example provided below creates a fresh project, copies code from an existing project, and builds the new project.

1. To add a new project to your workspace, go to **File → New → CCS Project**.



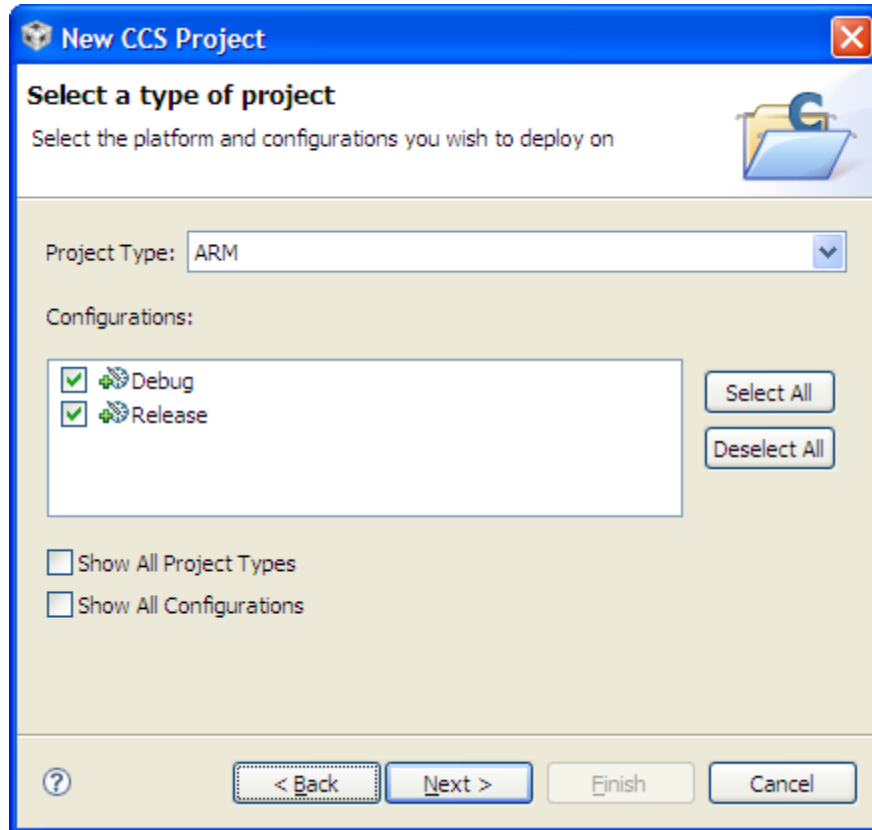
QUICKSTART – CODE COMPOSER STUDIO

2. **Code Composer Studio** prompts you with a dialog asking you to name the project (how about **my_project**) and specify the location of the project. Name your project, specify the location (your default workspace will do), and click **Next**.



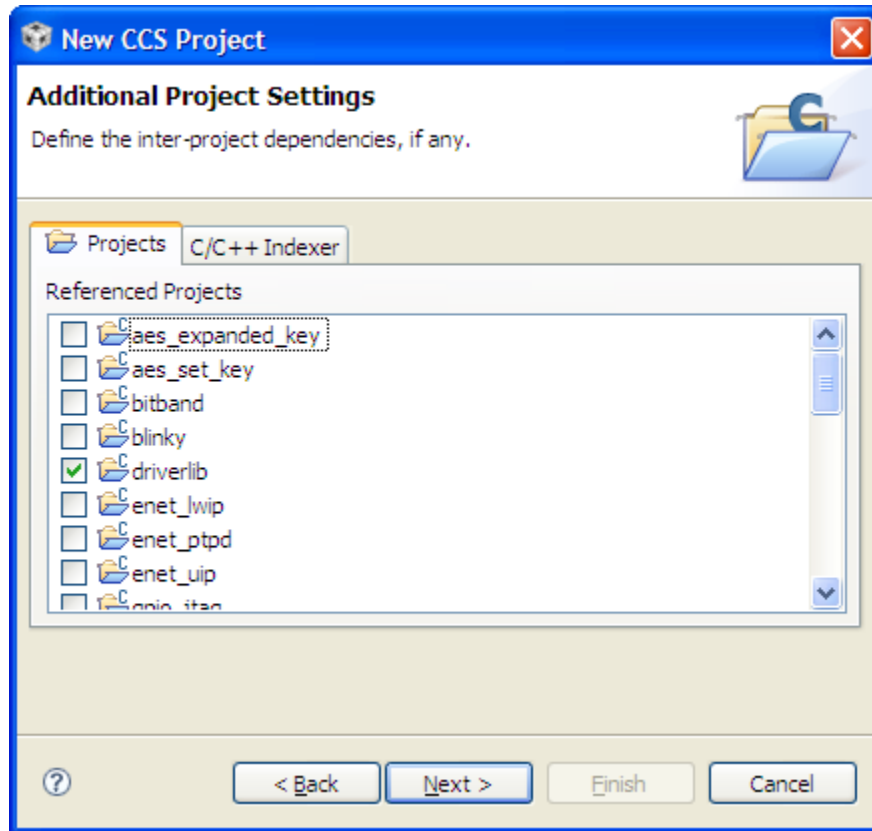
QUICKSTART – CODE COMPOSER STUDIO

3. The next dialog asks for the project type and configurations. Select **ARM** as the project type, check both the **Debug** and **Release** configurations, and click **Next**.



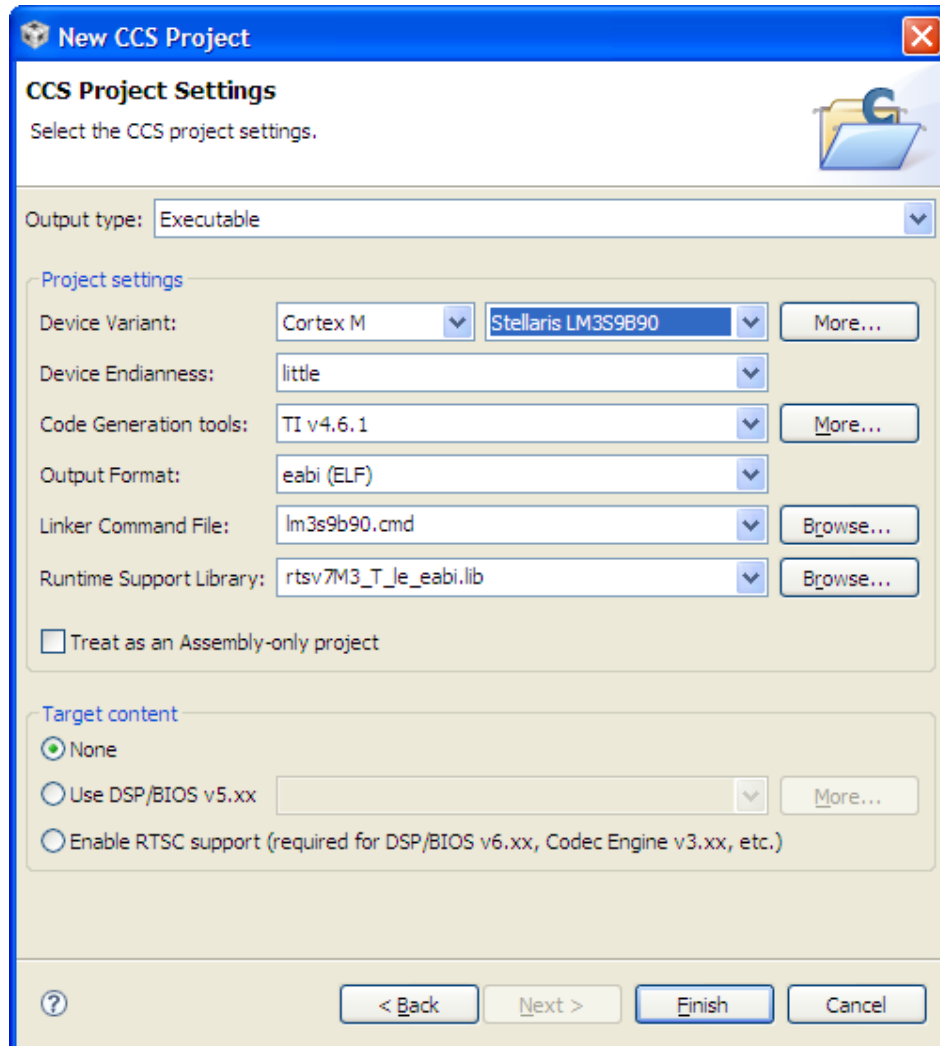
QUICKSTART – CODE COMPOSER STUDIO

4. The next dialog allows you to define any inter-project dependencies. If your project will be using **driverlib**, **usblib**, or **gplib**, now is a good time to define that dependency. **Select** those libraries and click **Next**.



QUICKSTART – CODE COMPOSER STUDIO

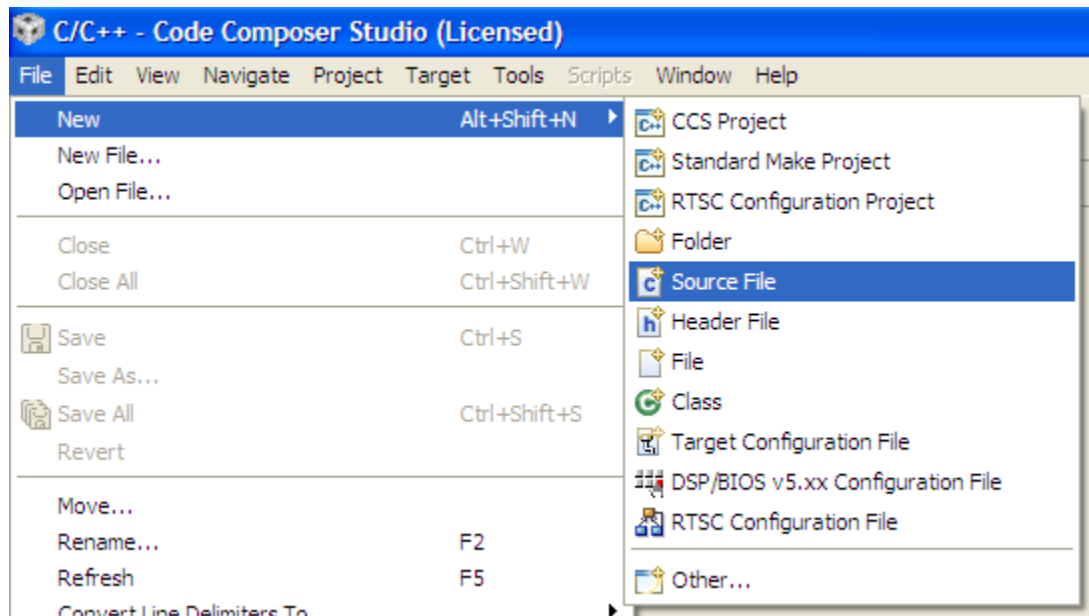
5. On the next dialog, select the appropriate **Stellaris device**, **little endian**, the **TI code generation tool**, **ELF output format**, and the **rtsv7M3_T_le_eabi.lib** runtime library. Click **Finish**.



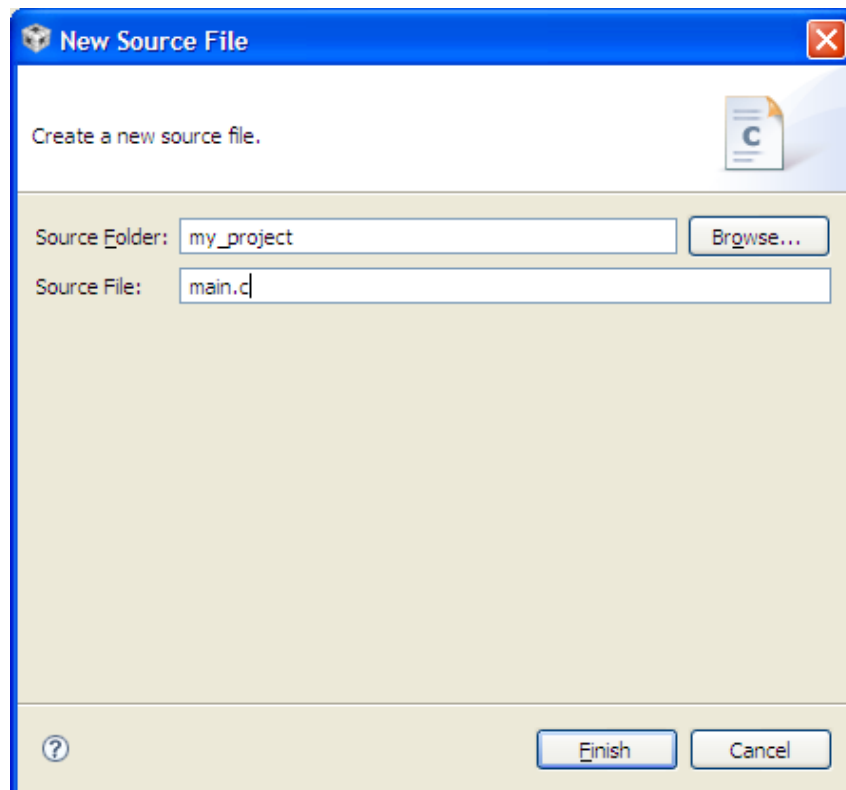
6. A new project has now been created, but the project does not include any source code. The next step is to add some startup code to the project. Using Windows Explorer, copy the **startup_ccs.c** file from an existing example directory in StellarisWare (such as **C:\StellarisWare\boards\<board>\blinky**) to your new project directory created in step 2 above (in your **My Documents** folder).

QUICKSTART – CODE COMPOSER STUDIO


7. Create a new C source file by going to **File → New → Source File**.

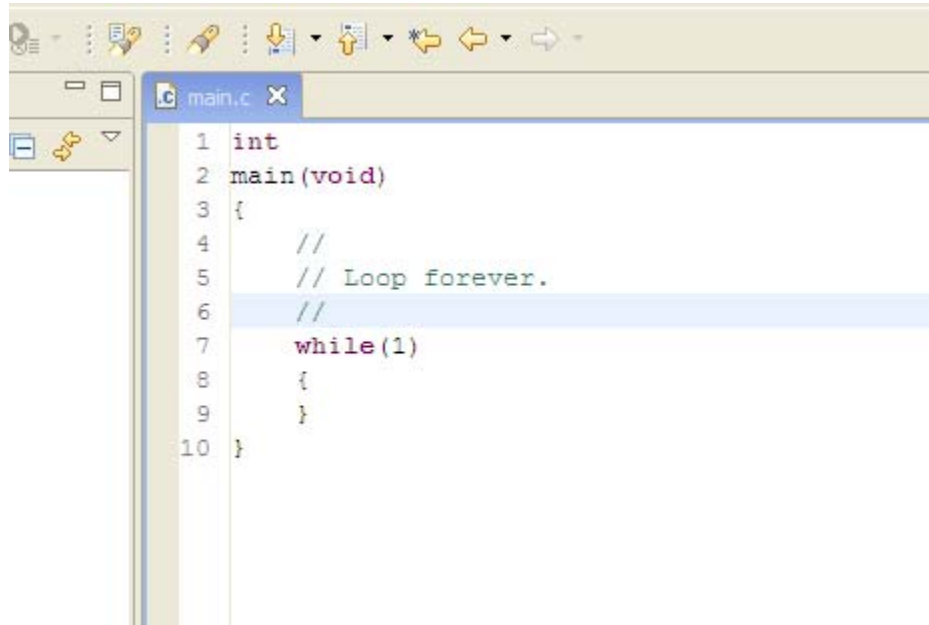


8. In the dialog box that pops up, name the file **main.c** and click **Finish**.



QUICKSTART – CODE COMPOSER STUDIO

9. **Code Composer Studio** will open the **main.c** file for editing. Add a **main()** function to **main.c** as shown below. **Save** the file by clicking the  **Save** button.



10. Both files, **main.c** and **startup_ccs.c**, should have automatically been added to the project. If not, **right-click** on **my_project** in the Project pane, select **Add Files to Project** and browse to the files.

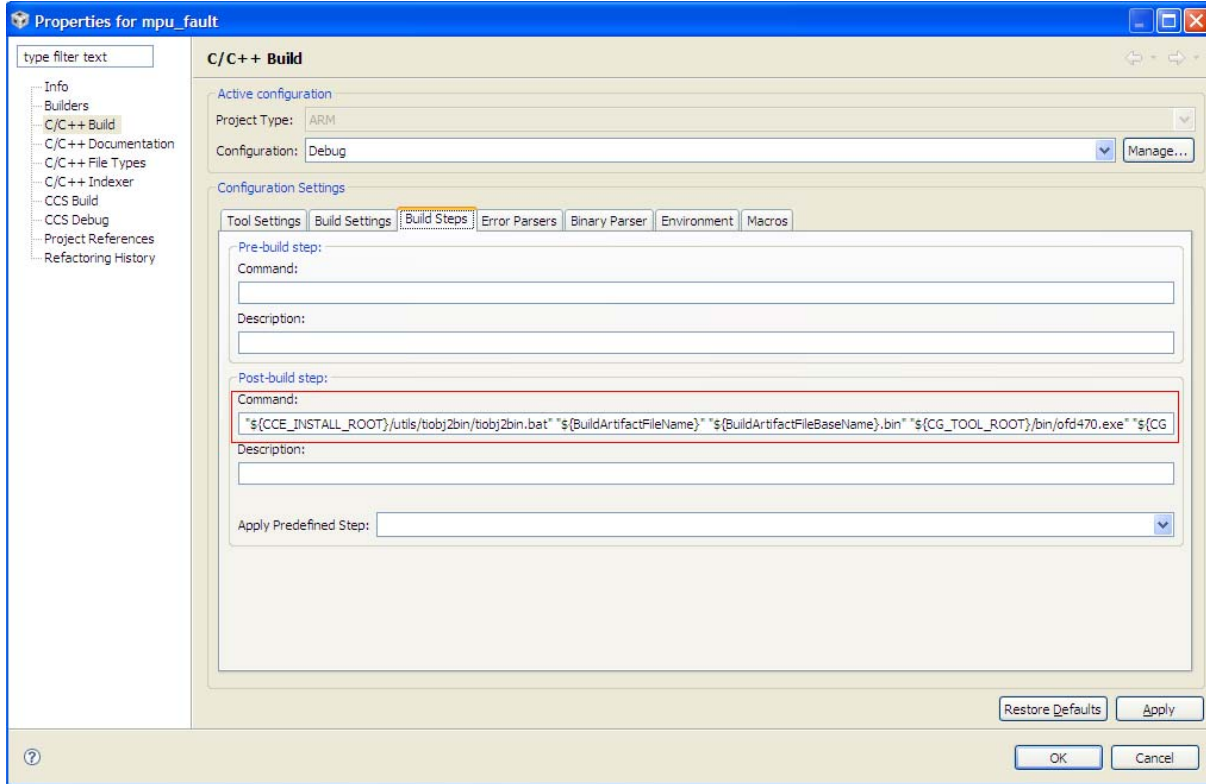


11. **Build** the new project by selecting **Project → Rebuild Active Project**.

With your project created, all you really need to do is add your own code. Use the existing StellarisWare board examples as a reference.

QUICKSTART – CODE COMPOSER STUDIO

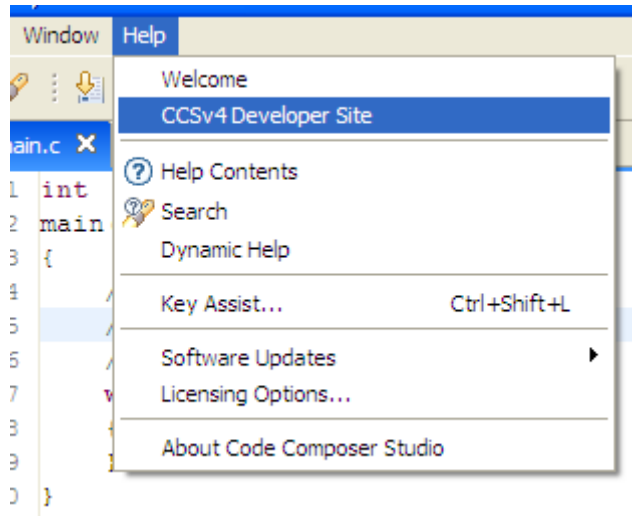
NOTE: To set up your project to output a binary file (**.bin**) to be used with tools such as **LM Flash Programmer**, copy the post-build step from an existing StellarisWare example project.



QUICKSTART – CODE COMPOSER STUDIO

Conclusion

You have now installed the Code Composer Studio development tools and used them to build and load an example application on your Stellaris Evaluation Board. From here, you can experiment with the debugger or start creating your own application using the example projects as a reference. For further information on Code Composer Studio, go to the CCS Developer Site.



Close Code Composer Studio now.

References

The following references are included on the Stellaris Evaluation Kit Documentation and Software DVD and are also available for download at www.ti.com/Stellaris:

- *Stellaris Evaluation Kit User's Manual*
- *StellarisWare Software*, Order Number SW-LM3S
- *StellarisWare Peripheral Driver Library User's Guide*, Order Number SW-DRL-UG

In addition, the following website may be useful:

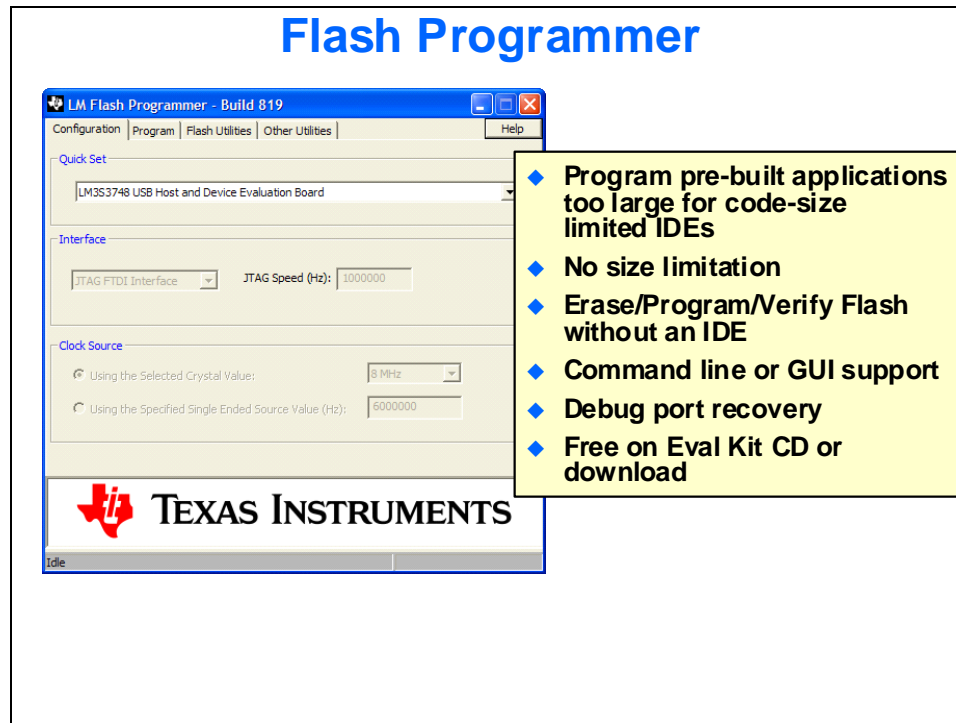
- Code Composer Studio website at <http://www.ti.com/ccstudio>

QUICKSTART – CODE COMPOSER STUDIO

LM Flash Programmer – Installation and Use

Before we run the next lab(s), you'll need to restore the quickstart application to your board.

This lab will step through the installation and use of the LM Flash Programmer. Steps are included for the QuickStart applications on both the LM3S3748 and LM3S8962 evaluation kits.



Hardware list:

- PC
- USB cable
- LM3S3748 or LM3S8962 evaluation kit

Software list:

- Workshop Installation Flash Drive

LM Flash Programmer Procedure

Drivers

1. StellarisWare Package

If you have not done so already, follow the steps in the **QUICKSTART** procedure to install the StellarisWare package for your board(s). This package includes the **bin** files required to program the quickstart application into the Flash memory of the microcontroller on your evaluation board.

2. Install the LM Flash Programmer onto your PC

Insert the **Workshop Installation Flash Drive** into a free USB port on your computer. Using Windows Explorer, open the Flash drive and double-click on **LMFlashProgrammer.msi**.

Follow the steps in the **Setup Wizard**, agreeing to the license and selecting the default installation folder. When the installation is complete, click **Close**.

3. Run the Flash Programmer

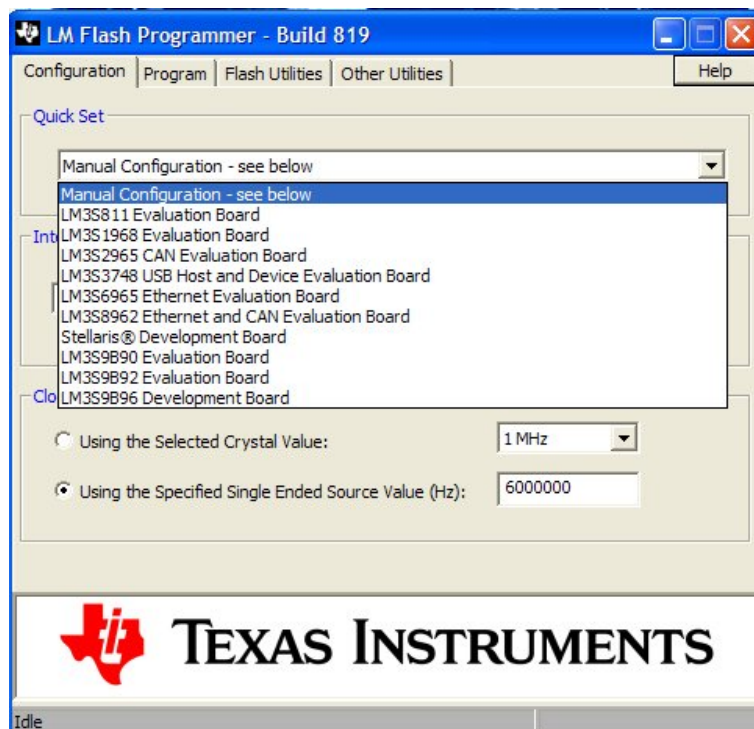
Make sure that Code Composer Studio is **closed** ... otherwise CCS and the Flash Programmer will “fight” for control of the board.



There should be a shortcut to the **LM Flash Programmer** on your desktop **double-click** it to open the tool. If the shortcut does not appear, go to **Start → All Programs → Texas Instruments → LM Flash Programmer** and **click** on **LM Flash Programmer**.

4. Select Your Board

Select your evaluation board, either the **LM3S3748 USB Host and Device Evaluation Board** or the **LM3S8962 Ethernet and CAN Evaluation Board** from the **Quick Set** pull-down menu under the **Configuration** tab.



5. Programming Setup

Click on the **Program** tab. Click the **Browse** button and navigate to the **bin** file for your board:

LM3S3748 QuickStart:

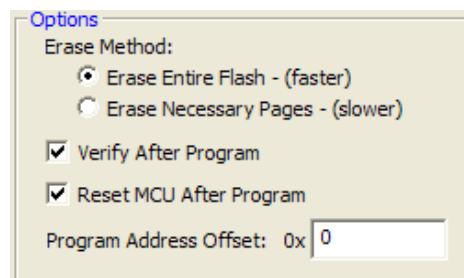
C:\StellarisWare\boards\ek-lm3s3748\qs-scope\ccs\Debug\qs-scope.bin

LM3S8962 QuickStart:

C:\StellarisWare\boards\ek-lm3s8962\qs_ek-lm3s8962\ccs\Debug\qs_ek-lm3s8962.bin

FYI: There are folders here containing applications which have been built with each supported IDE.

Make sure that the following checkboxes are selected:



6. Program the Flash

Assure that your board is properly connected to your computer's USB port and the evaluation boards **Debug** port. **Click** the **Program button**. You should see the programming and verification status at the bottom of the window. After these steps are complete, the application should be running on your evaluation kit.

Close the LM Flash Programmer.



You're done.

*** eee's uv yous ***

Stellaris[®] LM3S3748 Evaluation Kit

README FIRST

The Stellaris LM3S3748 Evaluation Kit provides a low-cost way to start designing applications with Stellaris microcontrollers on a compact and versatile evaluation platform. The evaluation kit design highlights the LM3S3748 microcontroller's key features including USB 2.0 full-speed (12 Mbps) controller, Analog-to-Digital Converter (ADC), and serial interfaces. The LM3S3748 Evaluation Board (EVB) includes connectors for both embedded USB Host and USB Device operation.

The LM3S3748 EVB can be used either as an evaluation platform or as a low-cost in-circuit debug interface (ICDI). In ICDI mode, the on-board microcontroller is bypassed, allowing programming or debugging of an external target.

Power to the EVB can be supplied through the DC jack, the USB Device connector, or the USB debug interface connector. A small switch controls whether the board is bus-powered using the USB Device connector or self-powered using the DC jack or USB debug interface connector.

WARNING: Do not change the power mode switch while power is applied. Doing so may damage the switch contacts.

Hardware list:

- PC with two USB ports, running Microsoft® Windows 2000, XP, or Vista
- USB cable
- LM3S3748 evaluation kit

Software list:

- Workshop Installation Flash Drive

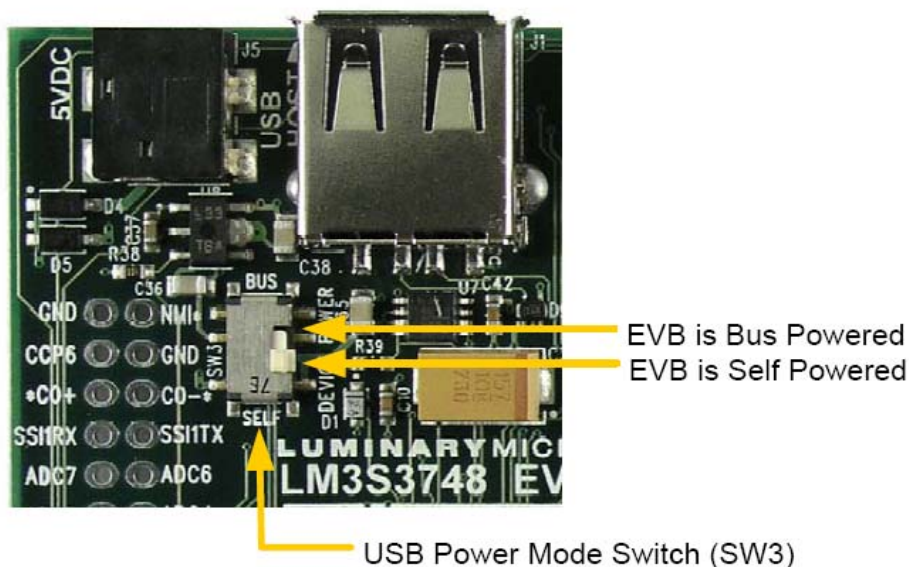
Procedure

Initial Board Set-Up

1. Power the EVB

Move the USB power mode switch (**SW3**) to the **SELF** position as shown below.

Note: The switch must be placed in the **SELF** position for the initial board setup to work correctly.



2. Connect the USB cable to the board

Using one of the USB cables provided in the kit, connect the mini-b (smaller) end of the USB cable to the USB debug interface connector labeled **DEBUG USB** on the EVB.

Note: If the USB cable is not plugged into the **DEBUG USB** connector, the board will not be powered and you will not be able to install the FTDI drivers in the next section.

3. Connect the USB cable to your PC

Connect the other end (Type A) of the USB cable to a free USB port on your host PC.

The PC's USB port is capable of sourcing up to 500 mA for each attached device, which is sufficient for the evaluation board. If connecting the board through a USB hub, it **must** be a powered hub.

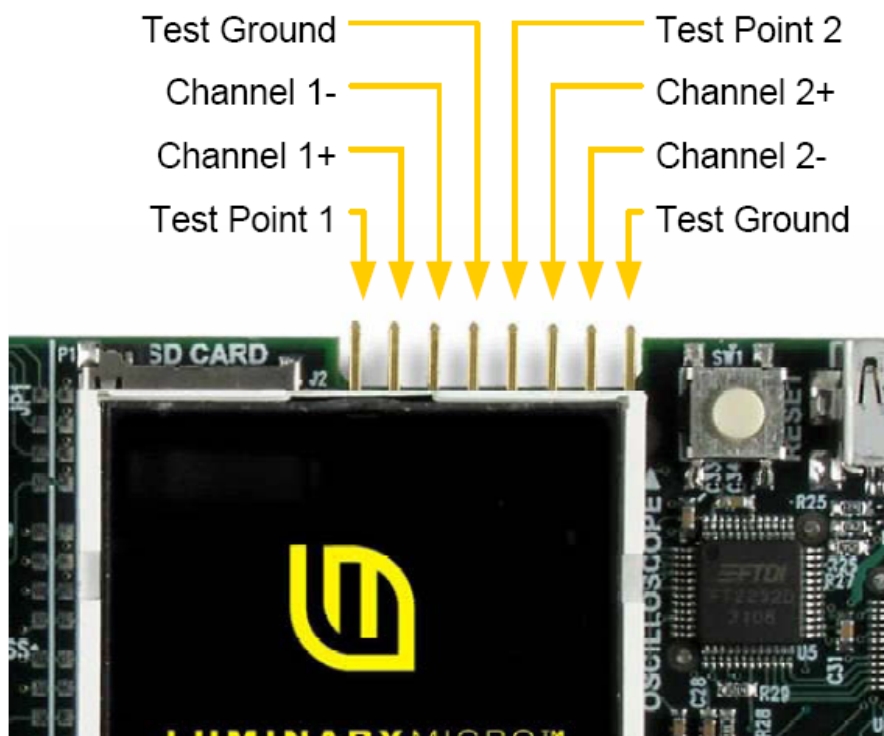
Quickstart Application

The LM3S3748 Evaluation Board comes preprogrammed with a quickstart application. Once you have powered the board, this application runs automatically. You have probably already noticed this running as you installed the drivers. A splash screen appears on the LCD for a few seconds before the application begins.

The quickstart application provides a simple two channel oscilloscope sampling at up to 1M samples per second. The two oscilloscope channels are differential measurement channels which provide waveform acquisition using the LM3S3748 microcontroller's Analog-to-Digital Converter (ADC). The evaluation board includes an oscilloscope header that contains the two channel differential inputs, two test point pins, and two test ground pins.

Test Point 1 is connected to the speaker input on the EVB and allows the signal for the keyboard click to be viewed. Note that waveform capture is typically not taking place while the keyboard is being serviced so the click may not be seen on the waveform display for every keypress.

Test Point 2 is connected to the output of a PWM generator set to drive a 1KHz square wave.



The EVB has a four-way navigation switch with press-to-select functionality that is used to configure the oscilloscope. The navigation switch is labeled **NAVIGATE** on the board. Rocking the control in the desired direction sends **up**, **down**, **left**, or **right** messages to the application and pressing on the center sends the **select** message.

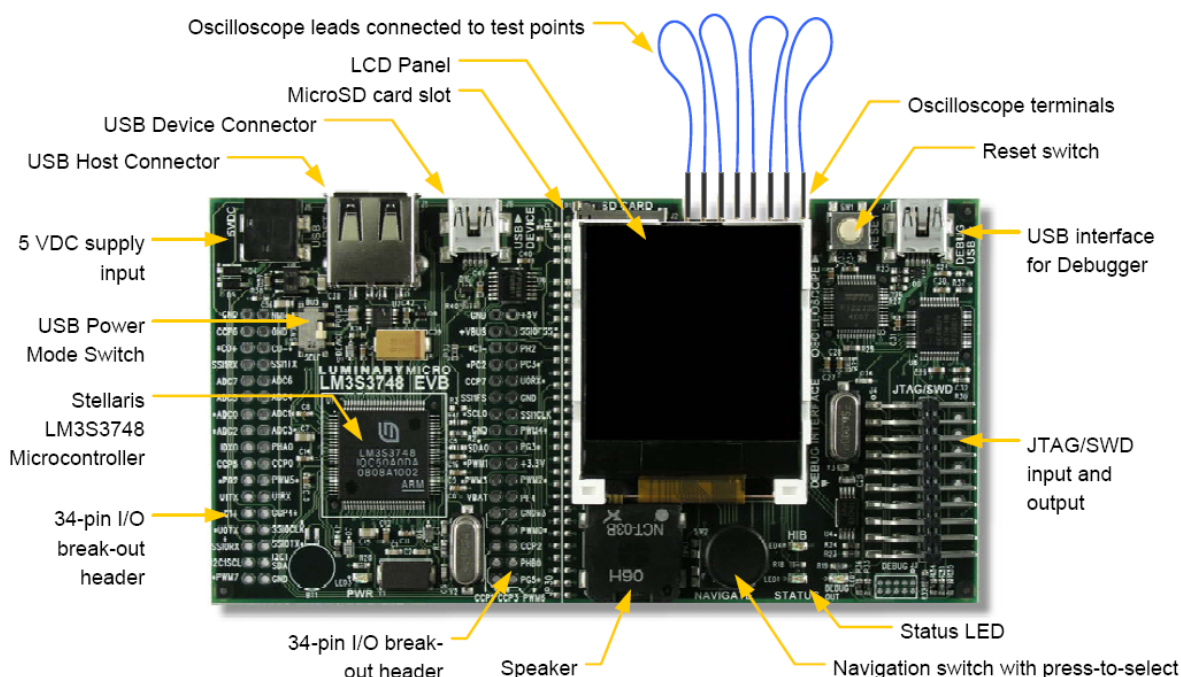
Controls and settings are arranged into groups by function such as display settings, trigger settings, file operations, and setup choices. These groups are accessed by pressing **select** to display the main menu. With the menu displayed, use **up** and **down** to navigate between the available groups. When the desired group is highlighted, press **select** once again to dismiss the menu.

Controls from the currently selected group are shown in the bottom portion of the LCD. Use **up** and **down** to cycle through the controls in the group and **left** and **right** to change the value of, or select the action associated with, the control which is currently displayed.

Using the oscilloscope to view Test Point 1 and Test Point 2, make the following connections using the included jumpers:

1. Connect Test Point 1 to Channel 1+.
2. Connect Test Ground to Channel 1-.
3. Connect Test Point 2 to Channel 2+.
4. Connect Test Ground to Channel 2-.

The connections should look like the graphic below.



5. The test point signals should now be visible on the LCD. To modify the volts per division and time per division options, follow these additional steps. Press **select** to bring up the main menu, navigate to highlight the **Display** group, and press **select** again. The **Display** group will be displayed at the bottom of the LCD.

Note: The **Display** group is the default active group after power-up.

6. Press **up** or **down** until the **Timebase** option is selected and then press **left** or **right** to modify the amount of time per division.

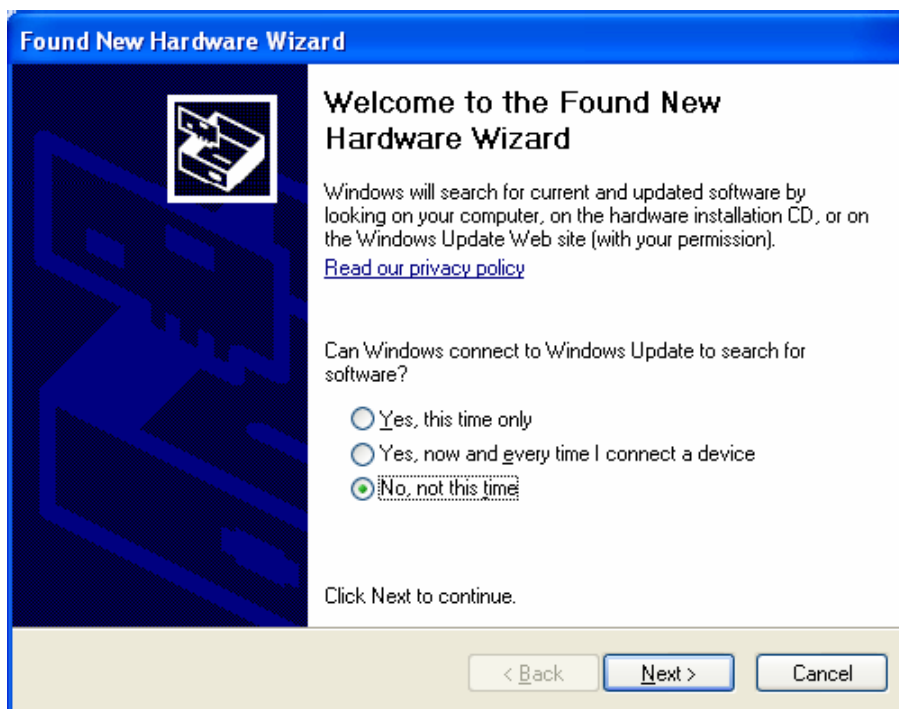
7. Press **up** or **down** until the **Ch1 Scale** or **Ch2 Scale** option is selected and then press **left** or **right** to modify the number of volts per division for that channel.

USB Device Mode

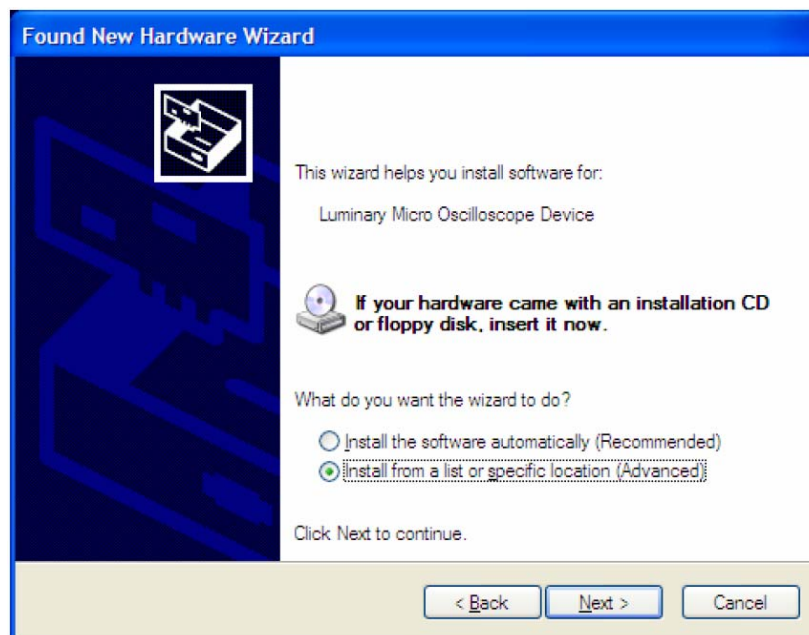
The quickstart oscilloscope application can connect to a Windows host machine via USB where a companion application running on the PC can be used to control the oscilloscope and display and save the waveforms. In this mode, the LM3S3748 EVB will be operating in USB device mode.

The first step is to install the necessary USB drivers on the PC.

1. Disconnect the USB cable from the USB connector labeled **DEBUG USB** if connected.
2. Move the USB power mode switch (SW3) to the **BUS** position.
3. Connect the mini-b (smaller) end of the USB cable to the USB device connector labeled **USB DEVICE** on the EVB. Connect the other end (Type A) to a free USB port on your host PC.
4. Windows starts the **Found New Hardware Wizard** and asks if it can connect to Windows Update to search for software. Select **No, not this time** and then click **Next**.



- Next, the **Found New Hardware Wizard** asks you from where to find the installation software. Select **Install from a list or specific location (Advanced)** and click **Next**.



- Make sure that the **Workshop Installation Flash Drive** is installed and ready in one of your laptops USB ports. Select **Search for the best driver in these locations**, and check **Include this location in the search**. Browse on the Flash Drive to the folder named **F:\windows_drivers** (the drive letter may be different) and click **Next**



- When the driver installation is finished, click **Finish** to close the dialog box.

8. The next step is to install the Windows Oscilloscope application from the **Workshop Installation Flash Drive**.

Note: The Windows oscilloscope application only supports WindowsXP and Vista.

Make sure that the **Workshop Installation Flash Drive** is installed and ready in one of your laptops USB ports.

9. Using **Windows Explorer**, open the **Workshop Installation Flash Drive**, find the file named **SW-USB-win-xxxx.msi** and **double-click** on it.

When the **Setup Wizard** appears, click **Next** and **Next** again to select the **default installation folder**. **Agree** to the license and click **Next**. Finally click **Next** to start the installation. When the installation completes, click **Close**.

10. After the USB examples have been installed, you can run the Oscilloscope application by clicking **Start → All Programs → Texas Instruments → Stellaris → USB Example → LM Oscilloscope**.

You should be able to see and control the application on the board using the Windows application. If you do not see the waveform in the display, **disconnect** and **reconnect** the USB cable on the evaluation board.

Close the Windows application when you are finished.

USB Host Mode

The quickstart oscilloscope application can also run in a USB Host mode. To run the application in USB Host mode, do the following:

1. Disconnect the USB cable from the evaluation board.
2. Move the USB power mode switch (SW3) to the **SELF** position.
3. Connect the mini-b (smaller) end of the USB cable to the USB debug interface connector labeled **DEBUG USB** on the EVB. Connect the other end (Type A) to a free USB port on your host PC.
4. Press **select** to bring up the main menu, navigate to highlight the **Setup** group, and press **select** again. The **Setup** group will be displayed at the bottom of the LCD. Press **up** or **down** until the **USB Mode** option is selected and then press **left** or **right** to switch to **Host** mode.

Note: We have had some issues with a recent revision of the quickstart application when switching to host mode. If the application appears to freeze at this point, you will need to re-flash the application. The flash programmer lab in chapter 6 will step you through this process.

In USB Host mode, the quickstart application can save waveforms to a USB flash memory stick or a Micro-SD card. The waveform files can be saved in bitmap (.bmp) or as comma separated value (.csv) formats. The LM3S3748 Evaluation Kit includes a USB flash memory stick, but a Micro-SD card is not included.

To save waveforms to the USB flash memory stick, do the following:

5. Plug the USB memory stick into the **USB HOST** connector. You should see a **USB drive detected** message on the LCD.
6. Press **select** to bring up the main menu, navigate to highlight the **File** group, and press **select** again. The **File** group will be displayed at the bottom of the LCD.
7. Press **up** or **down** until the **BMP on USB** option is selected, and then press **left** or **right** to save the bitmap waveform file to the USB memory stick.
8. Press **up** or **down** until the **CSV on USB** option is selected, and then press **left** or **right** to save the comma separated value waveform file to the USB memory stick.

Where to Find More Information

For more information on the LM3S3748 Evaluation Kit, see the *Stellaris LM3S3748 Evaluation Kit User's Manual*.

For more information on the LM3S3748 Evaluation Kit Quickstart Oscilloscope Application, see the *StellarisWare® Driver Library User's Guide* in the LM3S3748 Evaluation Kit's Example Applications section.

The above mentioned documents can be found on the Stellaris LM3S3748 Evaluation Kit CD or at the www.ti.com/Stellaris web site.

Software Development Tools

The next step is to install and run the software development tools included in the evaluation kit. For more information, see the quickstart guides included on the Stellaris LM3S3748 Evaluation Kit CD. Additional tools may be available through the www.ti.com/Stellaris web site.

References

The following references are included on the Stellaris LM3S3748 Evaluation Kit Documentation and Software CD and are also available for download at www.ti.com/Stellaris:

- *Stellaris LM3S3748 Evaluation Kit User's Manual*
- StellarisWare Driver Library
- *StellarisWare Driver Library User's Guide*
- *Stellaris LM3S3748 Microcontroller Data Sheet*



You're done.

*** one more blank page ***

Stellaris® LM3S8962 Evaluation Kit

README FIRST

The Stellaris LM3S8962 Evaluation Kit for Ethernet and CAN provides a low-cost way to start designing Ethernet and controller area network (CAN) applications with Stellaris microcontrollers. The LM3S8962 Evaluation Board (EVB) can function as either a complete evaluation target, or as a debugger interface to any external Stellaris device. The included USB cable is all that is needed to provide power and communication to the host PC.

Hardware list:

- PC with a USB port, running Microsoft® Windows 2000, XP, or Vista
- USB cable
- LM3S8962 evaluation kit

Software list:

- Workshop Installation Flash Drive

Procedure

Initial Board Set-Up

1. Connect and power the EVB

Connect the LM3S8962 Evaluation Board to the LM3S2110 CAN Device Board using the 10-way ribbon cable included in the kit. The cable inserts into the header labeled CAN on each board.

Note: Do not connect the larger 20-pin ribbon cable between the two boards. This will enable JTAG debugging of the LM3S2110. You could inadvertently overwrite the code in the devices flash memory.

To power the EVB, use the USB cable supplied in the kit. Connect the mini-b (smaller) end of the USB cable to the connector labeled **P4** on the EVB. Connect the other end (Type A) to a free USB port on your host PC. A PC's USB is capable of sourcing up to 500 mA for each attached device, which is sufficient for the evaluation board. If connecting the board through a USB hub, it must be a powered hub.

Quickstart Application

The LM3S8962 Evaluation Board comes preprogrammed with a quickstart application. Once you have powered the board, this application runs automatically. You have probably already noticed this running as you installed the drivers. A splash screen appears on the OLED display for a few seconds before the application begins.

The quickstart application is a game in which you navigate a character through a maze. Use the directional push buttons to move the character, and the user pushbutton (**SELECT**) to fire a missile to destroy the monsters. Score accumulates for maze progress and the number of monsters destroyed. The game lasts for only one character “life,” the score displays at the end of the game.

Since the OLED display on the evaluation board has burn-in characteristics similar to a CRT, the application also contains a screen saver. The screen saver only becomes active if two minutes have passed without the user pushbutton being pressed while waiting to start the game (that is, the screen saver never appears during game play).

After two minutes of running the screen saver, the display turns off and the user LED blinks. Exit either mode of screen saver by pressing the user pushbutton (**SELECT**). Press the button again to start the game.

The LM3S8962 Evaluation Board uses the CAN module on the LM3S8962 to communicate with the included CAN device board. During game play, the volume is adjusted by using the **up** and **down** buttons on the CAN device board. The Status LED on the CAN device board will blink during various portions of the quick start application.

The LM3S8962 microcontroller contains an integrated Ethernet controller, which is also used by the game. During the game, a user can attach the board to a LAN. If there is a DHCP server present on the LAN, the board attempts to obtain an IP address from the DHCP server. After about a minute, if no DHCP server is found, the board defaults to an IP address displayed at the bottom of the main game screen. To view the web page you must configure your host machine to be on the same subnet as the board. If connecting to a LAN, your PC's configuration should match the board's, except for the last 3 digits of the address. In most cases, if you connect the board directly to your PC, your machine automatically detects the correct IP address and subnet settings after several seconds.

1. PC Networking Changes

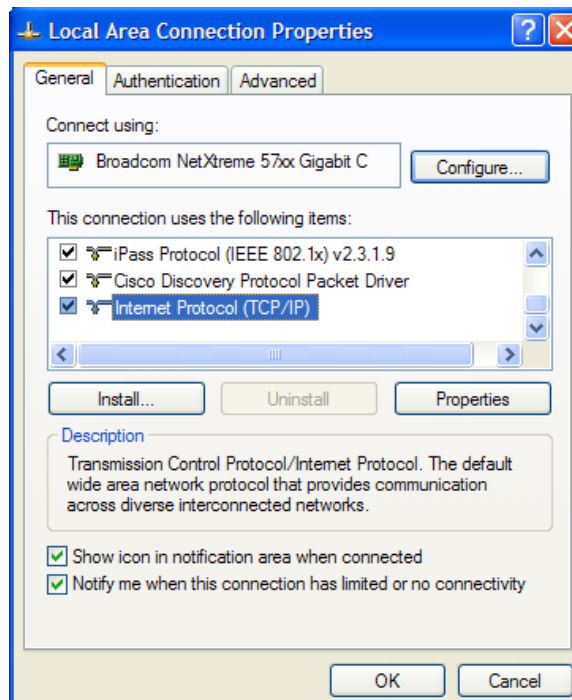
After 30 seconds or so, you should see an IP address at the bottom of the OLED display. This is the IP address of the LM3S8962 Ethernet interface.

If you have a 802.11, AirCard™, 3G or other wireless Internet connection, **disable** it at this time. We want to guarantee that your browser makes its connection attempt through the Ethernet cable. **Connect** the provided Ethernet cable to your PC and the 8962 board.

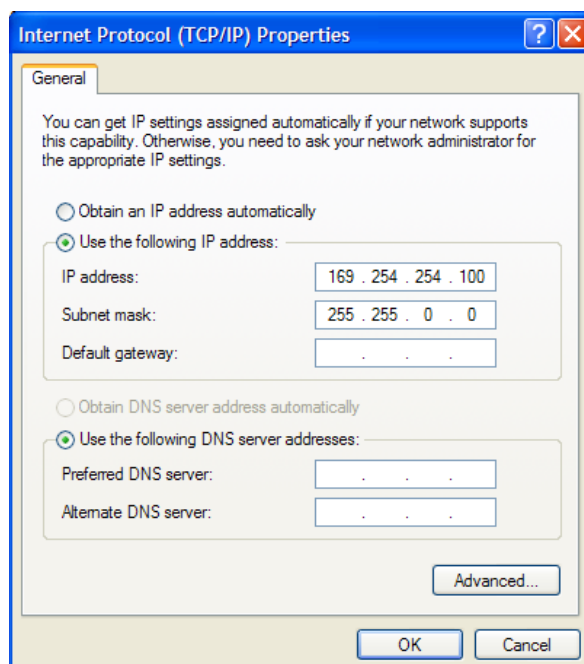
If you have problems connecting to the board in later steps, you may need to disable your firewall software.

Since there is no DHCP server present to allocate an address to the Ethernet port on the LM3S8962 board, the **lwIP** stack has defaulted to a preprogrammed IP address. In order to communicate, we need to provide the PCs Ethernet port a suitable address. Depending on the OS that you're using, the following procedure may be slightly different.

Go to **Start → Control Panel → Network Connections → Local Area Connection**. Click the **Properties** button. When the **Local Area Connection Properties** window appears, scroll down until you see **Internet Protocol (TCP/IP)** like shown below:



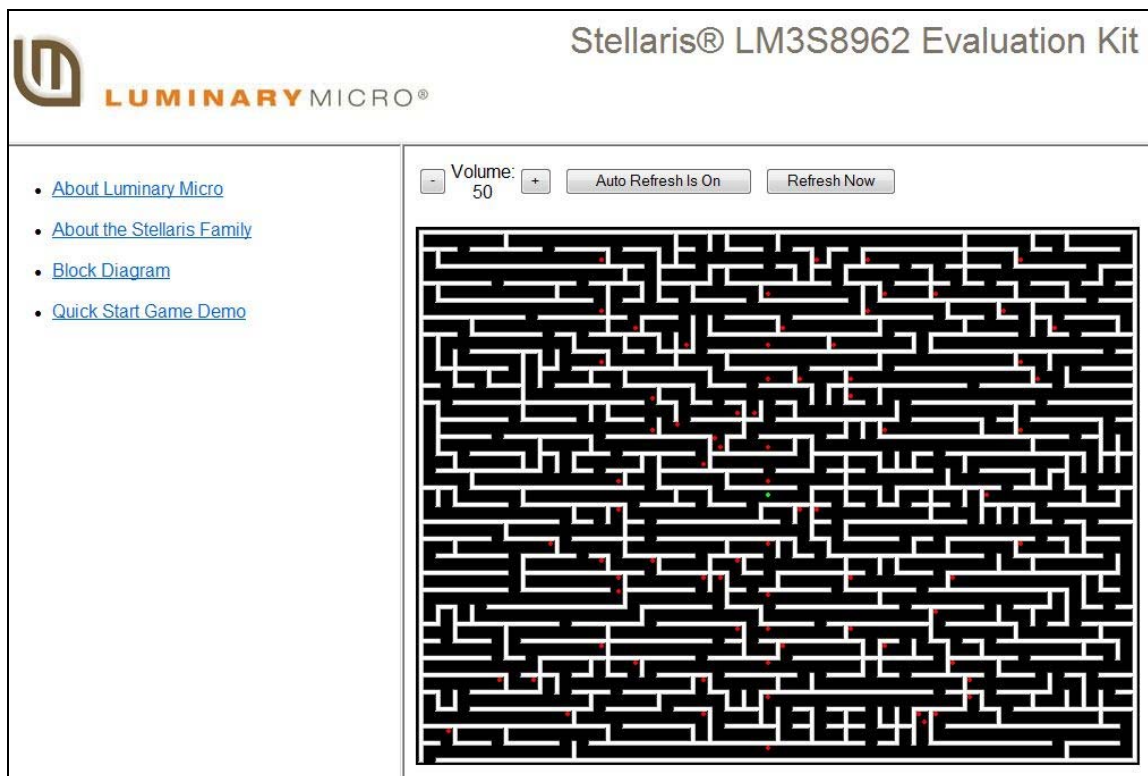
Click the **Properties** button. When the **Internet Protocol (TCP/IP) Properties** window appears, make a mental (or other) note of the settings you see, then make the selections shown below (if your boards' IP address was different, simply make sure that the first three fields are the same as shown on the OLED, and the fourth is different):



Click **OK**. Close the **Local Area Connection Properties** and **Local Area Connection Status** windows.

2. Start your browser

Start your web browser, and **enter** the address shown on your OLED display and press **Enter**. A web page, like the one below, served from the LM3S8962 should appear.



If the maze image above appears as a gray box on your browser, you likely do not have Java installed on your PC. Look on the flash installation drive and run the **jre-6u17-windows-i586-s.exe** Java offline installation file.

3. UART Connection

While the game is being played, a running tally of the score is output through UART0 of the LM3S8962 microcontroller. UART0 is connected to the FTDI's second serial channel. This serial channel is available to Windows as a Virtual COM port. To determine which COM# Windows has assigned to the Virtual COM port on the LM3S8962 board, follow these steps:

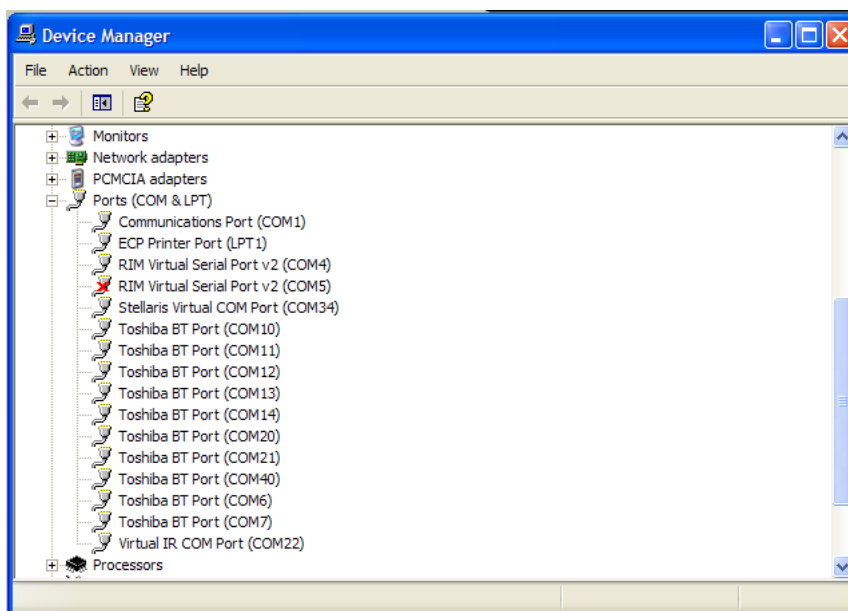
From the **Start Menu**, select **Control Panel**, then double-click the **System** icon.

Select the **Hardware** tab.

Click on the **Device Manager** button.

Click on the + symbol to expand the **Ports (COM & LPT)** group.

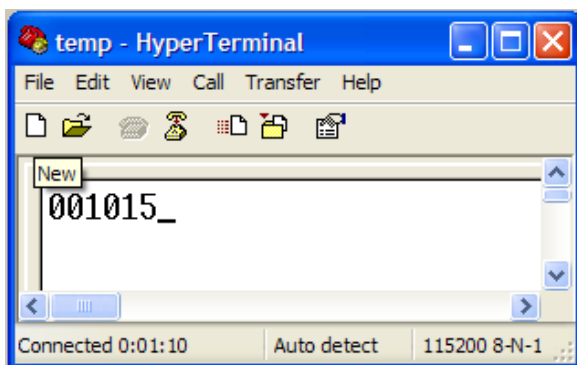
Stellaris Virtual COM Port (COM#) is listed as shown in the figure below. This COM# is the device you connect to using your terminal application. In this example, the COM port is **COM34**.




To view the score, open up a terminal application on your PC such as **HyperTerminal** (usually found under **Start → All Programs → Accessories → Communications**).

Give the connection a name like **Temp**, then connect using COM#, where # is the number Windows has assigned the Virtual COM port. Set the serial connection to a baud rate of **115200**, **8** data bits, **no** parity, **1** stop bit, and **no** flow control.

Your HyperTerminal window should look something like this:



If you have problems running HyperTerminal, open **Code Composer Studio** and, from the Menu bar, click **Window → Show view → Other...**. Click the + next to the **Terminal** folder and select **Terminal**. Click **OK**. At the top of the Terminal pane, click **Settings** . Change the **Connection Type** to **Serial** and use the settings shown above.

References

The following references are included on the Stellaris LM3S8962 Evaluation Kit Documentation and Software CD and are also available for download from www.ti.com/Stellaris :

- *Stellaris LM3S8962 Evaluation Kit User's Manual*
- StellarisWare® Driver Library
- *StellarisWare® Driver Library User's Manual*
- *Stellaris LM3S8962 Microcontroller Data Sheet*



You're done.

*** another blank page ***

CAN Peripheral Module

Introduction

This module covers the CAN peripheral, library and examples provided with the LM3S8962 evaluation kit.

Learning Objectives

- CAN Basics
- Ethernet vs. CAN
- CAN Drivers
- CAN Stacks
- Lab

Module Topics

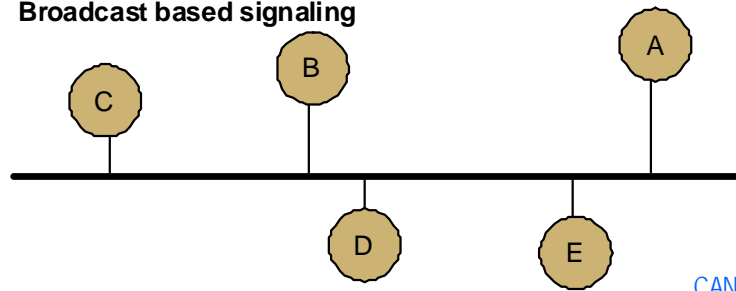
CAN Peripheral Module	1
<i>Module Topics</i>	2
<i>CAN Basics</i>	3
CAN Bus	3
CAN Node	4
CAN Principles	4
Message Format.....	5
Arbitration	5
Message Types	6
Message Objects	6
FIFO Mode	7
Ethernet vs. CAN.....	7
Stellaris CAN.....	8
<i>Kits and Software</i>	9
Drivers	10
APIs	10
Code Example	11
Stacks.....	14
<i>CAN Lab</i>	15
Description:	15
Hardware list:	16
Software list:.....	16
Procedure.....	17

CAN Basics

Controller Area Network (CAN)

A Multi-Master Serial Bus System

- ◆ Bosch CAN 2.0 A/B Standard
- ◆ High speed (up to 1 Mbps)
- ◆ Add a node without disturbing the bus (number of nodes not limited by protocol)
- ◆ Fewer wires (lower cost, less maintenance, and more reliable)
- ◆ Redundant error checking (high reliability)
- ◆ No node addressing (message identifiers)
- ◆ Broadcast based signaling

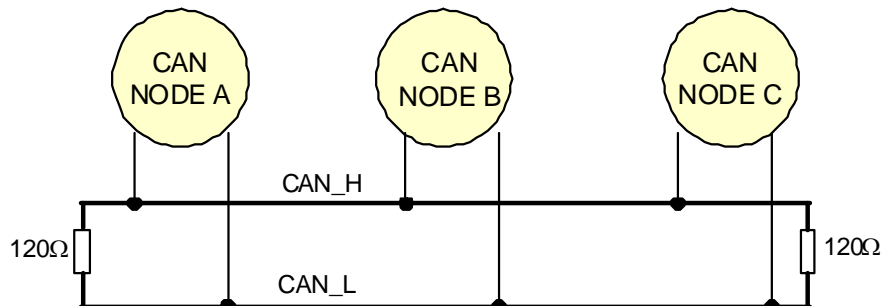


CAN Bus ...

CAN Bus

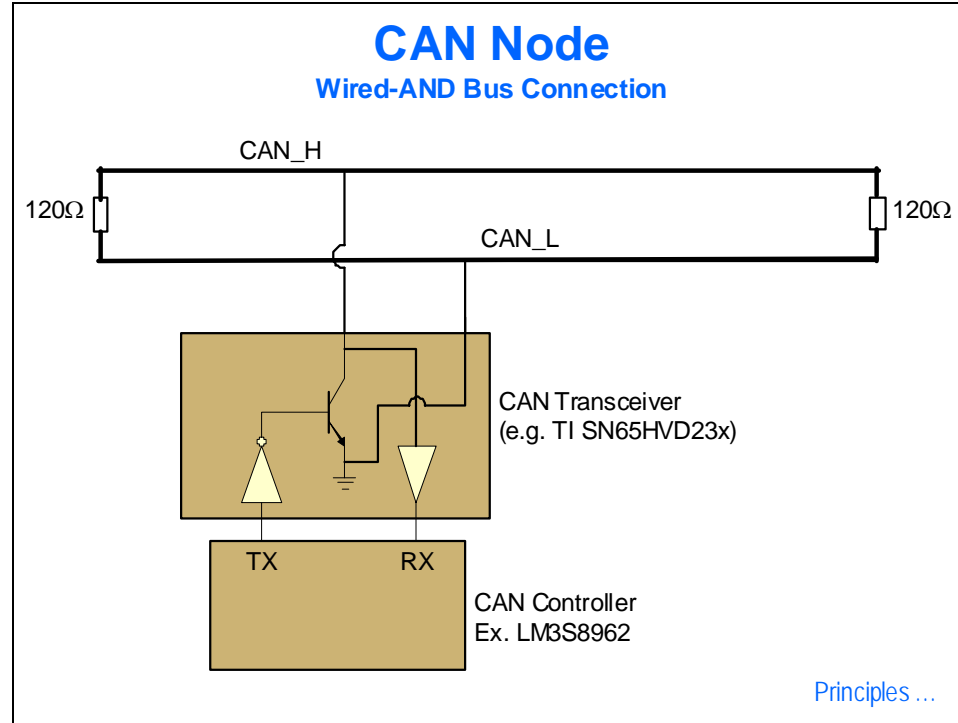
CAN Bus

- ◆ Two wire differential bus (usually twisted pair)
- ◆ Max. bus length depend on transmission rate
 - ◆ 40 meters @ 1 Mbps



CAN Node ...

CAN Node



CAN Principles

Principles of Operation

- ◆ Data messages transmitted are identifier based, not address based
- ◆ Content of message is labeled by an identifier that is unique throughout the network
 - ◆ (e.g. rpm, temperature, position, pressure, etc.)
- ◆ All nodes on network receive the message and each performs an acceptance test on the identifier
- ◆ If message is relevant, it is processed (received); otherwise it is ignored
- ◆ Unique identifier also determines the priority of the message
 - ◆ (lower the numerical value of the identifier, the higher the priority)
- ◆ When two or more nodes attempt to transmit at the same time, a non-destructive arbitration technique guarantees messages are sent in order of priority and no messages are lost

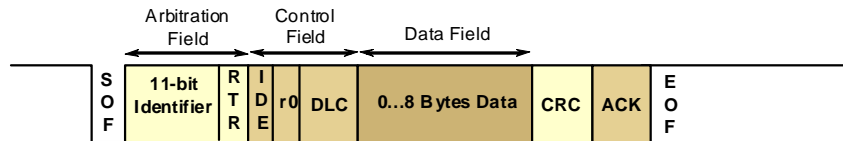
MSG Format ...

Message Format

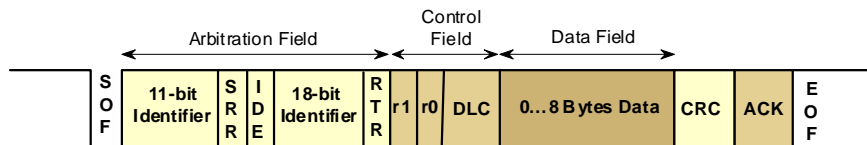
CAN Message Format

- ◆ Data is transmitted and received using Message Frames
- ◆ 8 byte data payload per message
- ◆ Standard and Extended identifier formats

- ◆ **Standard Frame: 11-bit Identifier (CAN v2.0A)**



- ◆ **Extended Frame: 29-bit Identifier (CAN v2.0B)**

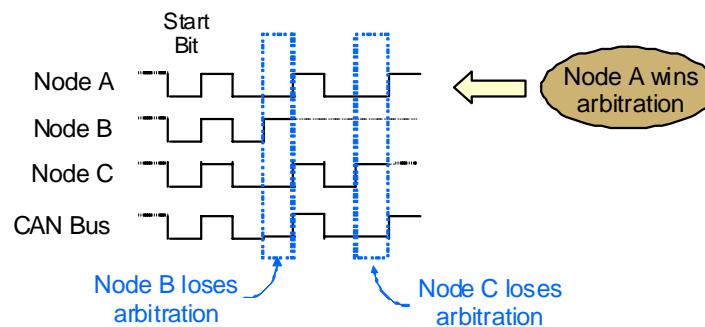


Arbitration ...

Arbitration

Non-Destructive Bitwise Arbitration

- ◆ Bus arbitration resolved via arbitration with wired-AND bus connections
 - ◆ Dominate state (logic 0, bus is high)
 - ◆ Recessive state (logic 1, bus is low)



MSG Types ...

Message Types

CAN Message Types

Data Frame

- *"Hello everyone, here's DATA ALPHA, hope you like it!"*

Remote Frame

- *"Hello everyone, can somebody please produce DATA ALPHA?"*

Error Frame

- *[Everyone out loud] "Uh-oh! Whoever sent that, let's try again."*

Overload Frame

- *"I heard you, but I'm bigger than you – and I am busy. Can you please wait?"*

CAN networks hear everything... when a node sends something, everyone listens and they either REACT or IGNORE

- ◆ There are two types of transmissions; event-triggered and time-triggered.
 - In event-triggered, events (such as a temperature threshold) cause transmission.
 - In time-triggered, each CAN-entity is only allowed to transmit during a given time allotment.
 - Stellaris features the ability to **disable auto-retransmission** for the benefit of **TTCAN** (so our device doesn't speak during someone else's turn)

MSG Objects ...

Message Objects

Message Objects (Mailboxes)

- ◆ **Message main parts: message identifier and data**
 - Identifier "names" the content of the message (11 bits or 29 bits)
 - Data can be from 0 to 8 bytes (0=remote frame)
- ◆ **Messages are:**
 - Received by protocol controller
 - Passed to message handler
 - Loaded into appropriate message object
 - Messages can be filtered (ignored) using a mask
- ◆ **Message object RAM**
 - 32 identical blocks
 - Ordered by priority (1 is highest, 32 lowest)
 - Application reads/writes Message RAM via the message object registers, not accessible directly

FIFO Mode ...

FIFO Mode

FIFO Mode

- ◆ You can “concatenate” multiple message objects (saves interrupt overhead for messages larger than 8 bytes)
- ◆ Set the identifier and mask to be the same value
- ◆ Data is filled starting with the highest priority message object (1 to 32)
- ◆ EOB bit (End of Buffer) set to 1 in last object (0 indicates this object belongs to a FIFO)

Enet vs CAN ...

Ethernet vs. CAN

Ethernet vs CAN

	PRO	CON
CAN	Robust, deterministic, real-time	Slower, not as widely adopted on HMI*-side of industrial control
Ethernet	Standardization in traditional office equipment, web-interface capability, high throughput	Non-real-time, non-deterministic

- ◆ Both CAN and Ethernet (together and independently) have viable benefits in industrial applications
- ◆ Gateways and floor terminals tend to be the most common places for CAN+Ethernet solutions
- ◆ An ARM Cortex-based Ethernet hub means code compatibility throughout the entire system

* Human-Machine Interface

Stellaris CAN ...

Stellaris CAN

Stellaris Integrated Controller Area Network (CAN)

Main Features



- ◆ Up to 3 Bosch-licensed CAN controllers
- ◆ Each supports CAN protocol version 2.0 part A/B
- ◆ Bit rates up to 1Mb/s
- ◆ 32 message objects, each with its own identifier mask
- ◆ Maskable interrupt
- ◆ Disable automatic retransmission mode for Time Triggered CAN (TTCAN)
- ◆ Programmable loop-back mode for self test operation

TI also provides

- ◆ Over 50 CAN-enabled Stellaris® ARM® Cortex™-M3 microcontrollers
- ◆ The EK-LM3S2965 CAN-network-in-a-CAN evaluation kit
- ◆ The EK-LM3S8962 CAN-network-plus-Ethernet evaluation kit
- ◆ Access to CAN quickstart applications and software examples from renowned CAN stack providers.

MCUs ...

Kits and Software

Ethernet+CAN Connected MCUs

	MCUs in Series	Memory and Speed		Core	General Purpose Timer Modules				Motion Control		Serial Interfaces					Analog				Digital		Package Options									
		Flash (KB)	SRAM (KB)		Max Speed (MHz)	Internal Precision Oscillator	32-bit Timer	16-bit Timer	Various Frequency OCP	RTC	Outputs	PWM	Fault Inputs	10/100 Ethernet MAC+PHY	IEEE 1588	CAN MAC	USB Full Speed	UART	I ² C	SS/SPi	fS		ADC (10-bit)		Internal Temp Sensor	LDO Voltage Regulator	Analog Comparators	Digital Comparators	GPIOs (5V)	Hibernates	
																							ADC Channels	ADC Speed (Ksps)							
LM3S8000s	12	256	64	-	50	-	4	8	1	6	?	6	1	2	?	?	3	-	3	2	2	-	8	1000	?	?	3	-	46	?	100-LQFP 108-BGA
LM3S9000s	6	256	96	?	100	?	4	8	2	8	?	8	4	2	?	?	2	OH/D	3	2	2	?	16	1000	?	?	3	7	65	?	100-LQFP

- ◆ First MCUs featuring fully integrated 10/100 Ethernet MAC+PHY and up to 3 Bosch CAN 2.0 A/B MACs
- ◆ IEEE 1588 Precision Time Protocol hardware assist

Stellaris LM3S8962 Evaluation Kit



- Stellaris LM3S8962 MCU with fully-integrated CAN module
- OLED graphics display with 128 x 64 pixel resolution
- User LED, navigation switches, and select pushbuttons
- Magnetic speaker
- LM3S8962 I/O available on labeled break-out pads
- Standard ARM® 20-pin JTAG debug connector with input and output modes
- ◆ Standalone CAN device board using Stellaris LM3S2110 microcontroller
- ◆ Ethernet cable, CAN ribbon cable, USB and JTAG cables

CAN Connected MCUs

		MCUs in Series		Memory and Speed		Core	General Purpose Timer Modules				Motion Control		Serial Interfaces					Analog				Digital		Package Options							
				Flash (KB)	SRAM (KB)		Max Speed (MHz)	Precision Oscillator	32-bit Timer	16-bit Timer	Watchdog	Output Comp.	PWM	Fault Inputs	Q1	Q2	10/100 Ethernet MAC+PHY	IEEE 1588	CAN MAC	USB Full Speed	UART	IC	SS/SP		I2S	ADC Channels	ADC Speed (Ksps)	Internal Temp Sensor	LDO Voltage Regulator	Analog Comparators	Digital Comparators
LM3S2000s	26	256	96	?	80	?	4	8	2	8	?	8	4	2	-	-	2	-	3	2	2	?	16	1000	?	?	3	7	60	?	64-LQFP 100-LQFP 108-BGA
LM3S5000s	12	256	96	?	80	?	4	8	2	8	?	8	4	2	-	2	OH/D	3	2	2	?	16	1000	?	?	3	7	71	?	64-LQFP 100-LQFP	
LM3S8000s	12	256	64	-	50	-	4	8	1	6	?	6	1	2	?	?	3	-	3	2	2	-	8	1000	?	?	3	-	46	?	100-LQFP 108-BGA
LM3S9000s	6	256	96	?	100	?	4	8	2	8	?	8	4	2	?	?	2	OH/D	3	2	2	?	16	1000	?	?	3	7	65	?	100-LQFP

- ◆ Featuring up to 3 Bosch CAN 2.0 A/B CAN MACs
- ◆ Independent CAN buffer allows simultaneous CAN usage with all other peripherals

Stellaris LM3S2965 Evaluation Kit



- OLED graphics display with 128 x 64 pixel resolution
- User LED, navigation switches, and select pushbuttons
- Magnetic speaker
- LM3S2965 I/O available on labeled break-out pads
- Standard ARM® 20-pin JTAG debug connector with input and output modes
- ◆ Standalone CAN device board using Stellaris LM3S2110 microcontroller
- ◆ CAN ribbon cable, USB and JTAG cables

Drivers ...

Drivers

StellarisWare CAN Drivers

- ◆ StellarisWare has a full set of drivers for the CAN modules
- ◆ Bit timing calculations are often complex and difficult to understand. StellarisWare has an easy to use API called **CANBitTimingSet** that automatically configures the CAN bit timing based on a passed set of parameters
 - There are also a set of pre-configured bit timing settings for each speed that you can use
- ◆ The clocking on the Fury devices is different than the Dust Devil and Tempest devices
 - Fury uses an 8MHz dedicated clock when the PLL is enabled, and the crystal clock otherwise
 - Dust Devil and Tempest use the system clock, whether that is the main oscillator, internal oscillator, or PLL

APIs ...

APIs

Stellaris CAN APIs: Initialization and Interrupts

- ◆ **Initialization**
 - CANInit()
 - CANBitTimingSet(), CANBitTimingGet()
 - CANBitRateSet()
 - CANEnable(), CANDisable()
 - CANRetrySet(), CANRetryGet()
- ◆ **Interrupts**
 - CANIntRegister(), CANIntUnregister()
 - CANIntEnable(), CANIntDisable()
 - CANIntStatus()
 - CANIntClear()

Stellaris CAN APIs: Message Objects and Status

- ◆ **Message objects**
 - ◆ CANMessageSet()
 - ◆ CANMessageClear()
 - ◆ CANMessageGet()
- ◆ **Status**
 - ◆ CANStatusGet()
 - ◆ CANErrCntrGet()

[Code Example ...](#)

Code Example

Code Example – CAN Initialization

- ◆ **CAN interfaces must be initialized in both applications**
 - ◆ LM3S8962 board – qs_ek-lm3s8962 project
 - ◆ LM3S2110 board – can_device-qs project

```
// Configure CAN pins
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
GPIOPinTypeCAN(GPIO_PORTD_BASE, GPIO_PIN_0 | GPIO_PIN_1);
// Enable the CAN controllers
SysCtlPeripheralEnable(SYSCTL_PERIPH_CAN0);
CANInit(CAN0_BASE);
// Configure the clock rate 8MHz, bit rate set to 250K bits/sec
CANBitRateSet(CAN0_BASE, 8000000, 250000);
CANEnable(CAN0_BASE);
// Enable interrupts from CAN controller
CANIntEnable(CAN0_BASE, CAN_INT_MASTER | CAN_INT_ERROR);
// Set up the message objects that will receive all messages
CANConfigureNetwork();
// Enable interrupts for the CAN in the NVIC
IntEnable(INT_CAN0);
```

Message Objects Used in QS Application

- ◆ Four CAN message objects are used in `qs_ek-lm3s8962` app (`can_net.c` file)
 - `g_MsgObjectButton` – receives button messages (Rx interrupt enabled) note: there are two buttons, and two events (press & release)
 - `g_MsgObjectLED` – sends LED brightness (Tx interrupt enabled)
 - `g_MsgObjectTx` – sends commands to 2110 (Tx interrupt enabled)
 - `g_MsgObjectRx` – receives commands from 2110 (Rx interrupt enabled)
- ◆ The “Message Object #” selects one of 32 locations in message object memory, also determines priority (lower # is higher)
- ◆ The “Message ID” is the 11-bit or 29-bit message identifier, in this application, no masking is needed or enabled

Message Object #	symbol
1	MSGOBJ_NUM_BUTTON
2	MSGOBJ_NUM_LED
3	MSGOBJ_DATA_TX
4	MSGOBJ_DATA_RX

Message ID	symbol
0x10	MSGOBJ_ID_BUTTON
0x12	MSGOBJ_ID_LED
0x20	MSGOBJ_ID_DATA_TX
0x21	MSGOBJ_ID_DATA_RX

Code Example – Configure Receive Object

- ◆ LM3S8962 board receives “button” messages from LM3S2110 CAN device board
- ◆ `g_MsgObjectButton` structure is initialized in `can_net.c` as follows:

```
// Set the identifier and mask for the button object
g_MsgObjectButton.ulMsgID = MSGOBJ_ID_BUTTON;
g_MsgObjectButton.ulMsgIDMask = 0;

// This enables interrupts for received messages
g_MsgObjectButton.ulFlags = MSG_OBJ_RX_INT_ENABLE;

// Set the size of the message and the data buffer used
g_MsgObjectButton.ulMsgLen = 2;
g_MsgObjectButton.pucMsgData = g_pucButtonMsg;

// Configure the Button receive message object
CANMessageSet(CAN0_BASE, MSGOBJ_NUM_BUTTON,
               &g_MsgObjectButton, MSG_OBJ_TYPE_RX);
```

Code Example – Configure Transmit Object

- ◆ LM3S2110 board sends “button” messages to the LM3S8962 board
- ◆ `g_MsgObjectButton` structure is initialized in `can_device_qs.c` as follows:

```
// This is the message object used to send button updates. This
// message object will not be “set” using CANMessageSet right now
// as that would trigger a transmission
g_MsgObjectButton.ulMsgID = MSGOBJ_ID_BUTTON;
g_MsgObjectButton.ulMsgIDMask = 0;
// This enables interrupts for transmitted messages.
g_MsgObjectButton.ulFlags = MSG_OBJ_TX_INT_ENABLE;
// Set message length, which should only be two bytes and the
// data is always whatever is in g_pucButtonMsg
g_MsgObjectButton.ulMsgLen = 2;
g_MsgObjectButton.pucMsgData = g_pucButtonMsg;
```

Code Example – Transmit Button Object

- ◆ LM3S2110 board transmits a button message to the 8962 using `g_MsgObjectButton`
- ◆ `g_MsgObjectButton` has already been initialized except for the two bytes of data containing the event (press or release) and specific button (up or down)
- ◆ Once data is set, `CANMessageSet()` is called to transmit the message

```
// ucEvent can be EVENT_BUTTON_PRESS (0x10)
// or EVENT_BUTTON_RELEASED (0x11)
g_MsgObjectButton.pucMsgData[0] = ucEvent;
// ucButton can be TARGET_BUTTON_DN (1) or TARGET_BUTTON_UP (0)
g_MsgObjectButton.pucMsgData[1] = ucButton;
// send the message
CANMessageSet(CAN0_BASE, MSGOBJ_NUM_BUTTON, &g_MsgObjectButton,
MSG_OBJ_TYPE_TX);
```

Code Example – Receive Object

- ◆ 8962 board receives a button message from the 2110 using `g_MsgObjectButton` in the interrupt handler `CANHandler()`

```
CANHandler(void)
{
    unsigned long ulStatus;
    // Find the cause of the interrupt, if it is a status interrupt then just
    // acknowledge the interrupt by reading the status register.
    ulStatus = CANIntStatus(CAN0_BASE, CAN_INT_STS_CAUSE);
    switch(ulStatus)
    {
        case MSGOBJ_NUM_BUTTON:
        {
            // Read the Button Message.
            CANMessageGet(CAN0_BASE, MSGOBJ_NUM_BUTTON, &g_MsgObjectButton, 1);
            // . . . Rest of the code to handle the button message
            break;
        }
        // all other cases: MSGOBJ_NUM_LED, MSGOBJ_NUM_DATA_TX, MSGOBJ_NUM_DATA_RX
    }
    CANIntClear(CAN0_BASE, ulStatus);
}
```

[Stacks ...](#)

Stacks

CAN Stacks

- ◆ The main two protocols used in CAN-based designs are CANopen and DeviceNet
 - ◆ Other protocols include POWERLINK, EtherCAT, Profinet, EtherNet/IP and also TCP/IP



<http://www.micrium.com>



<http://www.port.de>



<http://www.rtaautomation.com/>

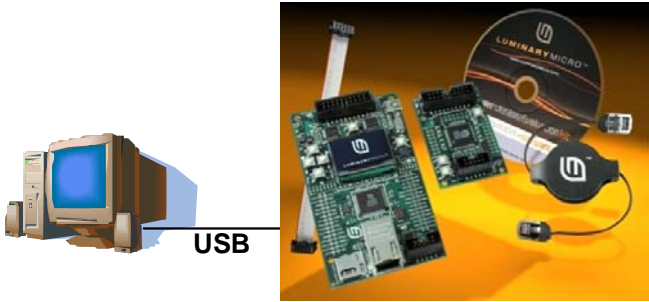
[Lab ...](#)

CAN Lab

Description:

We'll use the LM3S8962 eval board and the LM3S2110 CAN device board to investigate CAN communication.

CAN Lab



- ◆ LM3S8962 EVM
- ◆ LM3S2110 CAN Device board
- ◆ 10-pin CAN cable (not the 20-pin JTAG cable)
- ◆ USB cable
- ◆ Buttons on device board send CAN volume messages
- ◆ Select button on LM3S8962 board lights device LED

Hardware list:

- ✓ LM3S8962 Evaluation Board
- ✓ LM3S2110 CAN device board
- ✓ 10-pin CAN ribbon cable
- ✓ 3 meter, 4 pin, USB Type A-M to mini-USB Type B-M

Software list:

- ✓ Code Composer Studio 4.1
- ✓ LM3S8962 software examples

Procedure

1. Open the IDE

Open **Code Composer Studio** and create a **workspace_8962CAN** workspace (we're using a new workspace for each lab to prevent issues and learn the steps by repetition). **Close** the Welcome screen if it appears. Click the **C/C++ Perspective** button to make sure that you're in the editing perspective.

2. Import driverlib

From the menu bar, click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**. When the **Import** dialog appears, browse to the root directory of the driver library (**C:\StellarisWare\driverlib**). Click **OK**. Be sure that the checkbox next to **driverlib** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.

3. Import the CAN project

From the menu bar, click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**. When the **Import** dialog appears, browse to the root directory of the quickstart project (**C:\StellarisWare\boards\ek-lm3s8962\qs_ek-lm3s8962**). Click **OK**. Be sure that the checkbox next to **qs_ek-lm3s8962** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.

4. Open can_net.c for editing

When the project has finished importing, **close** any open editor windows. **Maximize** the IDE window if you haven't already and **expand** the **qs_ek-lm3s8962** folder in the **Projects** pane.

Double-click on **can_net.c** to open it for editing.

can_net.c is made up of several modules:

CANMain()	Handles connection to the other device and incoming messages
CANGetTargetVersion()	Retrieves the target board's firmware version
CANUpdateTargetLED()	Sends message to set brightness of the LED on the target board
CANConfigureNetwork()	Configures message objects used
CANHandler()	CAN interrupt handler
CANConfigure()	Configures the CAN h/w and interrupts

As long as you're poking around in the CAN code, here are some of the important objects to look for:

Message Object Structure

```
typedef struct
{
    unsigned long ulMsgID;
    unsigned long ulMsgIDMask;
    unsigned long ulFlags;
    unsigned long ulMsgLen;
    unsigned char *pucMsgData;
}
tCANMsgObject
```

Members:

ulMsgID The CAN message identifier used for 11 or 29 bit identifiers.

ulMsgIDMask The message identifier mask used when identifier filtering is enabled.

ulFlags This value holds various status flags and settings specified by tCANObjFlags.

ulMsgLen This value is the number of bytes of data in the message object.

pucMsgData This is a pointer to the message object's data.

Description:

The structure used for encapsulating all the items associated with a CAN message object in the CAN controller.

tCANObjFlags – these are the flag bits in the CAN message object

MSG_OBJ_TX_INT_ENABLE This indicates that transmit interrupts should be enabled, or are enabled.

MSG_OBJ_RX_INT_ENABLE This indicates that receive interrupts should be enabled, or are enabled.

MSG_OBJ_EXTENDED_ID This indicates that a message object will use or is using an extended identifier.

MSG_OBJ_USE_ID_FILTER This indicates that a message object will use or is using filtering based on the object's message identifier.

MSG_OBJ_NEW_DATA This indicates that new data was available in the message object.

MSG_OBJ_DATA_LOST This indicates that data was lost since this message object was last read.

MSG_OBJ_USE_DIR_FILTER This indicates that a message object will use or is using filtering based on the direction of the transfer. If the direction filtering is used, then ID filtering must also be enabled.

MSG_OBJ_USE_EXT_FILTER This indicates that a message object will use or is using message identifier filtering based on the extended identifier. If the extended identifier filtering is used, then ID filtering must also be enabled.


MSG_OBJ_REMOTE_FRAME This indicates that a message object is a remote frame.


MSG_OBJ_NO_FLAGS This indicates that a message object has no flags set.

5. Test the current software

Before we change anything, let's make sure that the project works as it is. **Connect** your 8962 board to your laptop. From the menu bar, click **Project** and then select **Build Active Project**. Sometimes it's handy to build the project and check for errors without attempting to load it to Flash memory.

6. Debug and Run

When the build completes, we need to load the program to the board and start the debugger. Click **Target** on the menu bar, then click **Debug Active Project** or you can simply click the  **Debug** button.

The program will **load** to the 8962 Flash memory and the perspective will switch to **Debug**. Click the  **Run** button to start program execution. The maze game on the OLED display should appear and the speaker should emit the game sounds.

7. Back to the Editor

Click the **C/C++ Perspective** button (in the upper right) to return to the editor. Page down and find the **CANHandler()** module in **can_net.c**. This module handles the interrupt caused by received CAN messages. When you have more time, you can look at where those messages were sent by exploring the **can_device_qs** project that runs on the LM3S2110.

8. Look at CANUpdateTargetLED()

We'd like to add some code that will turn the LM3S2110 board status LED **on** when the **up** button is released and **off** when the **down** button is released.

If you haven't looked closely at the **CANUpdateTargetLED()** module, now would be a good time to do so. Note that there are two passed parameters:

ucLevel	0 is off and 1 is on
bFlash	true flashes the LED, false does not

Calling **CANUpdateTargetLED()** with the correct parameters sends a CAN message from the LM3S8962 board to the LM3S2110 board.

9. Add some code

Find the **CANHandler()** module and note how **AudioVolume()** is called to alter the game volume. Add the calls to **CANUpdateTargetLED()** as shown below:

```
        if(g_MsgObjectButton.pucMsgData[1] == TARGET_BUTTON_UP)
        {
            //
            // Adjust the volume up by 10.
            //
            AudioVolumeUp(10);
            CANUpdateTargetLED(1, false);    //LED ON
        }

        //
        // Check if the down button was released.
        //
        if(g_MsgObjectButton.pucMsgData[1] == TARGET_BUTTON_DN)
        {
            //
            // Adjust the volume down by 10.
            //
            AudioVolumeDown(10);
            CANUpdateTargetLED(0, false);    //LED OFF
        }
    }


    break;
```


10. Change Preferences

It would be nice not to have to Build and Debug in two separate steps. We can change a selection so that clicking on **Debug** will rebuild the project if needed.

On the menu bar, click **Window**, then **Preferences**. Click on the + next to **CCS**, and then click on **Debug**. Check the **Rebuild the program (if required) before loading** checkbox. Click OK

11. Build/Load/Run

Click the  **Debug** button. Save the changes to **can_net.c** and reload the **.out** file when prompted. The project will automatically build and load to the 8962 Flash memory.

Change the perspective to **Debug**. Click the  **Run** button to start program execution. The maze game on the OLED display should appear and the speaker should emit the game sounds.

12. Testing our changes

After you click the **resume** button, **don't** touch the **up/down** buttons until **Press Select To Play** appears on the OLED. At that point, pressing the **up** button on the LM3S2110 sends a **VolumeUp CAN** message to the **LM3S8962 board**, which then responds by sending a CAN message to turn on the status LED on the **LM3S2110 board**. The down button works in a similar fashion to turn the LED off.

If you wait too long and the screen saver appears, press the **Select** button on the **LM3S8962 board** to exit the screen saver mode.

13. Remove changes

Go back to the **C/C++ perspective** and edit the **can_net.c** file. Comment out the two lines that you added to the code. **Build/Load/Run** again to make sure you've removed the changes.

Remember that the QuickStart application has been replaced by this application. You can reprogram it at any time with either Code Composer Studio or the Flash Programmer.



You're done

*** A mind is the only thing that can really be blank ***

Ethernet Peripheral Module

Introduction

This module covers the Ethernet peripheral, library and examples provided with the LM3S8962 evaluation kit.

Learning Objectives

- Ethernet MAC and PHY
- Stellaris implementations
- Choosing a stack
- Open source stacks
- Lab

Module Topics

Ethernet Peripheral Module.....	1
<i>Module Topics.....</i>	<i>2</i>
Common Protocols	3
MAC and PHY	3
Hardware Design	5
Ethernet MCUs	6
Serial to Ethernet Kit	7
IEEE1588	7
Choosing a Stack	9
Some Handy Definitions	9
Stellaris APIs	12
<i>Ethernet Lab</i>	<i>13</i>
Description:	13
Hardware list:	14
Software list:.....	14
Procedure.....	15
<i>Additional Web Server Information</i>	<i>22</i>

Common Protocols

Five-Layer TCP/IP Model – Common Protocols

Application Layer: DHCP, DNS, FTP, HTTP, IMAP4, IRC, NNTP, XMPP, POP3, RTP, SIP, SMTP, SNMP, SSH, TELNET, RPC, RTCP, RTSP, TLS (and SSL), SDP, SOAP, GTP, STUN, NTP, etc...

Transport Layer: TCP, UDP, DCCP, SCTP, RSVP, ECN, etc...

Network/Internet Layer: IP (IPv4, IPv6), OSPF, IS-IS, BGP, IPsec, ARP, RARP, RIP, ICMP, ICMPv6, IGMP, etc...

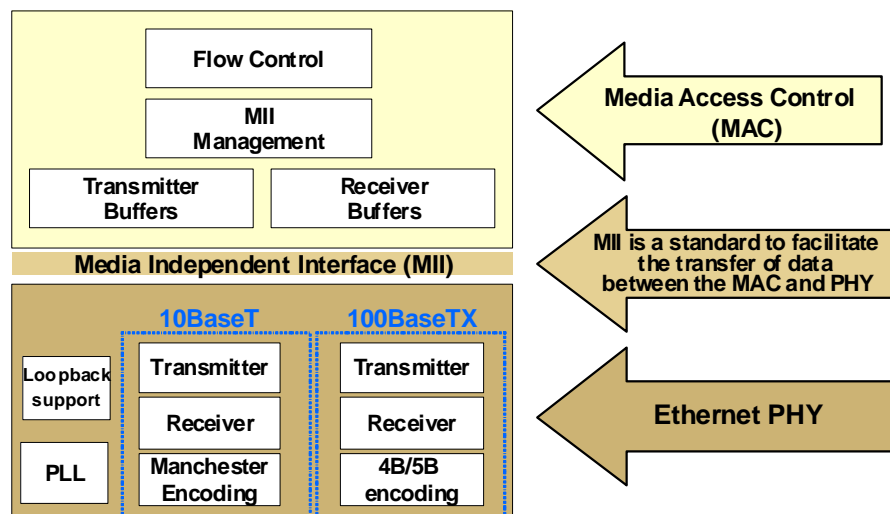
Data Link Layer: Ethernet, 802.11 (WLAN), 802.16, Wi-Fi, WiMAX, ATM, DTM, Token ring, FDDI, Frame Relay, GPRS, EVDO, HSPA, HDLC, PPP, PPTP, L2TP, ISDN, ARCnet, LLTD, etc...

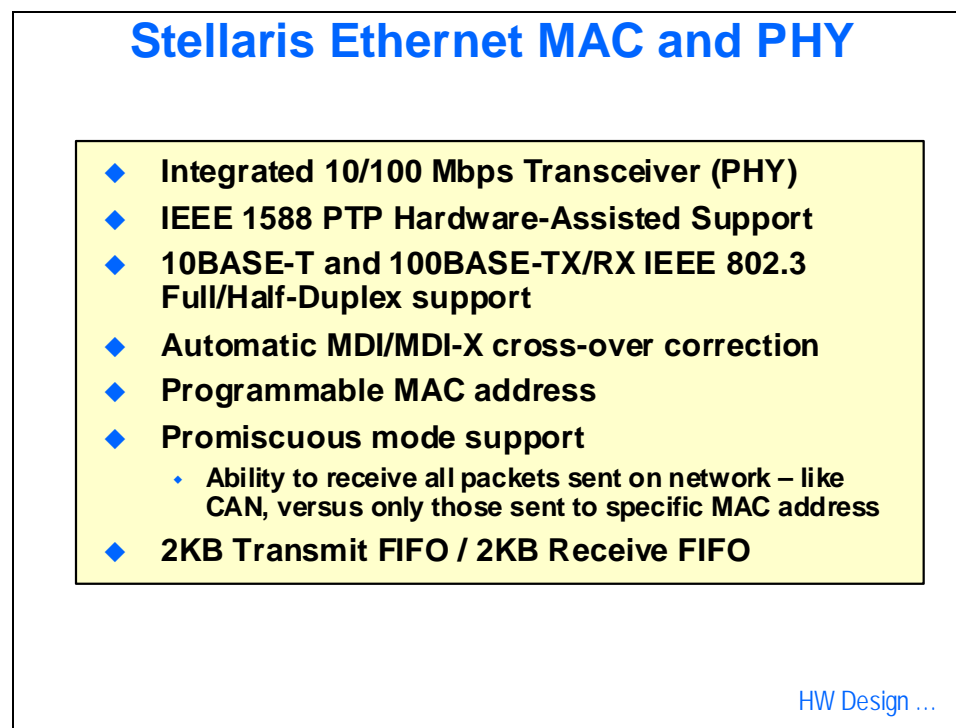
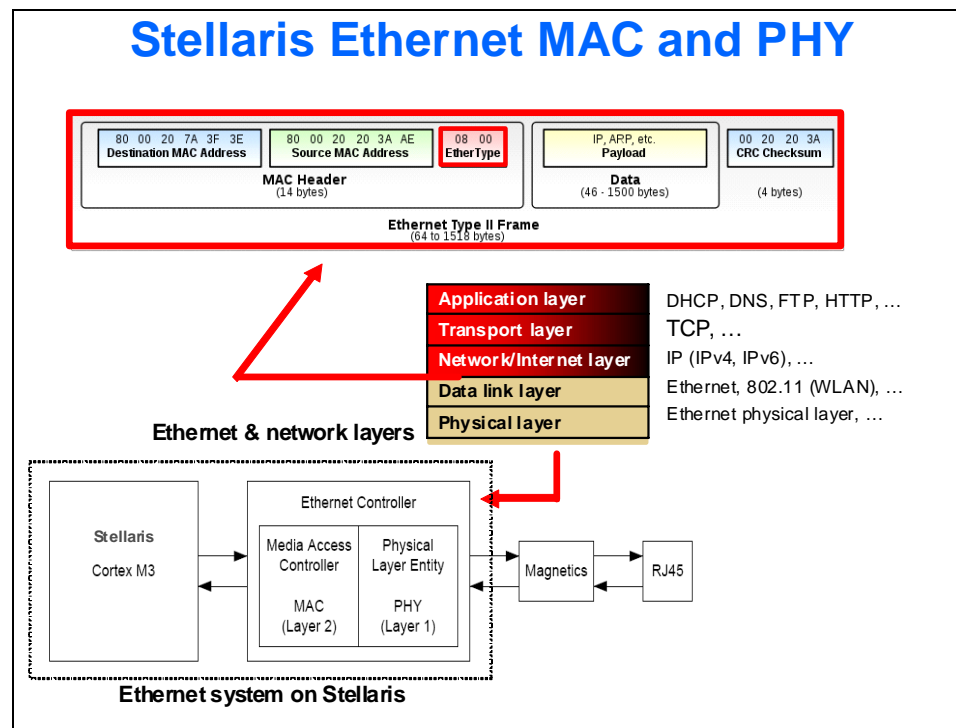
Physical Layer: Ethernet physical layer, Twisted pair, Modems, PLC, SONET/SDH, G.709, Optical fiber, Coaxial cable, etc...

MAC + PHY ...

MAC and PHY

Ethernet MAC and PHY





13 - 5

MCUs ...

Ethernet MCUs

Ethernet Connected MCUs

	MCUs in Series	Memory and Speed		Core	General Purpose Timer Modules				Motion Control		Serial Interfaces						Analog				Digital		Package Options								
		Flash (KB)	SRAM (KB)		Max Speed (MHz)	Internal Precision Oscillator	32-bit Timer	16-bit Timer	Watchdog	CCP	RTC	Outputs	Inputs	QEI	10/100 Ethernet MAC+PHY	IEEE 1588	CAN MAC	USB Full Speed	UART	I ² C	SSI/SP	FS		ADC (10-bit)		Internal Temp Sensor	LDO Voltage Regulator	Analog Comparators	Digital Comparators	GPIOs 5-V	Hibernate
																								ADC Channels	ADC Speed (Ksp/s)						
LM 3S 6000s	19	256	64	-	50	-	4	8	1	6	?	6	1	2	?	?	-	3	2	2	-	8	1000	?	?	3	-	46	?	100-LQFP 108-BGA	
LM 3S 8000s	12	256	64	-	50	-	4	8	1	6	?	6	1	2	?	?	-	3	2	2	-	8	1000	?	?	3	-	46	?	100-LQFP 108-BGA	
LM 3S 9000s	6	256	96	?	100	?	4	8	2	8	?	8	4	2	?	?	2	OH/D	3	2	2	?	16	1000	?	?	3	7	65	?	100-LQFP

Stellaris LM3S6965 Evaluation Kit



- OLED graphics display with 128 x 64 pixel resolution
- User LED, navigation switches, and select pushbuttons
- Magnetic speaker
- LM3S6965 I/O available on labeled break-out pads
- Standard ARM® 20-pin JTAG debug connector with input and output modes
- MicroSD card slot
- Included µIP Web Server (from FreeRTOS.org™)

◆ Ethernet, USB, and JTAG Cables

Ethernet+CAN Connected MCUs

	MCUs in Series	Memory and Speed		Core	General Purpose Timer Modules				Motion Control		Serial Interfaces						Analog				Digital		Package Options										
		Flash (KB)	SRAM (KB)		ROM/Flash	Max Speed (MHz)	Internal Precision Oscillator	32-bit Timer	16-bit Timer	Watchdog Timing	CCP	RTC	Outputs	Inputs	Fault Inputs	QEI	10/100 Ethernet MAC+PHY	IEEE 658	CAN MAC	USB Full Speed	UART	I ² C		SS/HS	I ² S	ADC (10-bit)		Internal Temp Sensor	LDO Voltage Regulator	Analog Comparators	Digital Comparators	GPIOs (5V)	Hibernate
																										ADC Channels	ADC Speed (Ksp/s)						
LM3S8000s	12	256	64	-	50	-	4	8	1	6	?	6	1	2	?	?	?	3	-	3	2	2	-	8	1000	?	?	3	-	46	?	100-LQFP 108-BGA	
LM3S9000s	6	256	96	?	100	?	4	8	2	8	?	8	4	2	?	?	?	2	OH/D	3	2	2	?	16	1000	?	?	3	7	65	?	100-LQFP	

Stellaris LM3S8962 Evaluation Kit



- Stellaris LM3S8962 MCU with fully-integrated CAN module
- OLED graphics display with 128 x 64 pixel resolution
- User LED, navigation switches, and select pushbuttons
- Magnetic speaker
- LM3S8962 I/O available on labeled break-out pads
- Standard ARM® 20-pin JTAG debug connector with input and output modes

◆ Standalone CAN device board using Stellaris LM3S2110 microcontroller

◆ Ethernet cable, CAN ribbon cable, USB and JTAG cables

Kit ...

Serial to Ethernet Kit

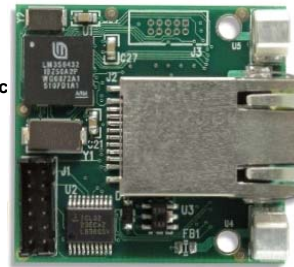
Stellaris Serial-to-Ethernet Kit



Example applications:

- SCADA Remote Terminal Units (RTUs)
- Electronic Flow Meters (EFMs)
- Medical Point-of-Care and Retail Point-of-Sales Machines
- CCTV RS-232 Recorders
- RS-232 Stepper Motor Controller Systems

- **LM3S6432** in a 10 x 10 mm BGA package for reduced board size
- **10/100 Mbit Ethernet port**
 - Auto MDI/MDIX cross-over correction
 - Traffic and link indicators
- **Serial ports**
 - UART0 has RS232 levels, transceiver runs at up to 250 Kbits/sec
 - UART1 has CMOS/TTL levels, can run at 1.5 Mbits/sec
 - UART ports include RTS/CTS for flow control
- **Software**
 - IP configuration with static IP address or DHCP
 - Telnet server for access to serial port
 - Web server for module configuration
 - UDP responder for device discovery
 - Telnet client for Ethernet-based serial port extender
- **Module supports 5 V and 3.3 V supplies**
- **JTAG port pads for factory programming**

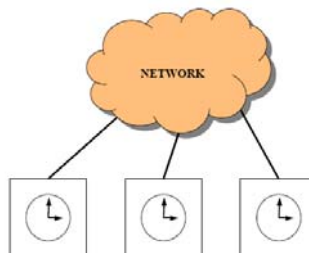


PTP ...

IEEE1588

Ethernet - IEEE1588 PTP

- ◆ IEEE 1588 is “Precision Clock Synchronization Protocol for Network and Control Systems” or **Precision Time Protocol (PTP)**
- ◆ IEEE 1588 is a protocol designed to synchronize real-time clocks in the nodes of a distributed system that communicate using a network (Ethernet) at a high degree of accuracy
- ◆ Microsecond accuracy is easily achievable using low cost, small footprint implementations such as Stellaris



Visualizing IEEE 1588

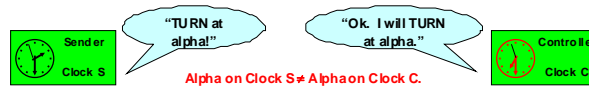
- ◆ Before IEEE 1588, Ethernet communication in control applications occurred without absolute determinism:

- Assume *Sender* sends a control instruction *Turn* to *Controller*
- Assume also that *Clock S* and *Clock C* are not synchronized

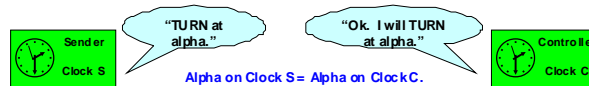
- If *Sender* asks *Controller* to *Turn* upon receipt of the instruction, then there is no telling when *Controller* will receive *Turn*.



- Even if *Sender* asks *Controller* to *Turn* at a given time *alpha*, there is still the problem of unsynchronized clocks.



- But if *Sender* asks *Controller* to *Turn* at a given time *alpha*, and the clocks are synchronized to a master, then determinism is achieved



PTP in Industrial Applications

- ◆ Industry synchronization requirements for PTP Applications

Application area	Required synchronization accuracy
Low speed sensors (e.g. pressure, temperature)	Milliseconds
Common electro-mechanical devices (e.g. relays, breakers, solenoids, valves)	Milliseconds
General automation (e.g. materials handling, chemical processing)	Milliseconds
Precise motion control (e.g. high speed packaging, printing, robotics)	A few microseconds
High speed electrical devices (e.g. synchrophasor measurements)	Microseconds
Electronic ranging (e.g. fault detection, triangulation)	Sub microsecond

- ◆ PTP and motion control

- Variable frequency drives require few 10s of microseconds
 - Software generally 5μs
 - 44% of applications are networked, 63% use Ethernet TCP/IPB
- Servo-controlled systems require 100s of nanoseconds
 - Requires significant hardware assist
 - 36% of applications are networked, 56% use Ethernet TCP/IPC

- ◆ Stellaris implementation

- Open source lwIP + PTPd : within 500nS of master clock, jitter +/- 500nS
- This represents a greater than ten fold improvement over typical SW-only implementations

Choosing a Stack ...

Choosing a Stack

Choosing a Stack – Application Examples

- ◆ ***“I want to add Ethernet connectivity to my factory floor.”***
 - Stellaris Ethernet-enabled MCUs
- ◆ ***“Transmission/Reception of information needs to be guaranteed.”***
 - TCP over UDP
- ◆ ***“I need workers at each machine workstation to be able to monitor and control the workstation from the workstation’s HMI computer.”***
 - HTTP
- ◆ ***“I need the ability to send system update files to each workstation from our main control room.”***
 - FTP
- ◆ ***“I want the system to automatically send an email if the connection to a workstation is bad.”***
 - ICMP
- ◆ ***“I need for my system to operate behind a firewall so that no one can hack in from the outside.”***
 - NAT
- ◆ ***“I need to be able to add more workstations to the network with ease.”***
 - DHCP

Some Handy Definitions

Common TCP/IP Protocols

Acronym	Translation	Purpose
TCP	Transmission Control Protocol	guarantee delivery
IP	Internet Protocol	data oriented
UDP	User Datagram Protocol	fire-and-forget
ARP	Address Resolution Protocol	finding a address
RARP	Reverse Address Resolution Protocol	finding a address
BOOTP	Bootstrap Protocol	finding a address
DHCP	Dynamic Host Configuration Protocol	adding devices to a network
BSD	Berkeley Socket	connecting to the internet
ICMP	Internet Control Message Protocol	error message generation
IGMP	Internet Group Management Protocol	manage IP multicast groups
PPP	Point-To-Point Protocol	direct point-to-point connection
SLIP	Serial Line Internet Protocol	direct point-to-point connection
DNS	Domain Name System	translate host name to address
FTP	File Transfer Protocol	transfer files point-to-point
TFTP	Trivial File Transfer Protocol	FTP, but for smaller files
RIP	Routing Information Protocol	routing internal networks
RTP/RTCP	Real-time Transport (Control) Protocol	send audio/video over internet
Telnet	Terminal Emulation	remote access
HTTP	Hypertext transfer Protocol Server	publish/retrieve web pages
SNMP	Simple Network Management Protocol	manage/monitor client status
SMTP	Simple Mail Transport Protocol	send email over internet
POP3	Post Office Protocol-3	retrieve email over internet
SNTP	Synchronized Network Time Protocol	network clock synchronization
PTP*	Precision Time Protocol (aka IEEE1588)	deterministic synchronization
NAT	Network Address Translation	network privacy
SSLs	Secure Sockets Layer	secure communication
IPSec	Internet Protocol Security	virtual private network
IKE	Internet Key Exchange	security key/certificate sharing

This slide is not in the presentation.

Choosing a Stack – Size and Cost Considerations

- ◆ Not every stack combination fits in every microcontroller
- ◆ Need to consider the RAM/ROM size of the microcontroller

- ◆ How much memory is on the specified Stellaris MCU?
 - ◆ What else is the Stellaris MCU expected to manage besides Ethernet?
- ◆ Most third-party partners post accurate stack size info
- ◆ Some 3P code size claims are not always accurate
- ◆ Free options - Open source uIP and lwIP ported to Stellaris
- ◆ Professional options - Optimized for size, for speed, and for functionality better than open source

Stacks ...

Communications Stacks for Stellaris®

TPV	Product	Stack	ARP	AutoIP	BOOTP	BSD	DHCP	DNS	FTP	HTTP	ICMP	IGMP	IKE	IP	IPSec	NAT	POP3	PPP	PTP	RARP	RIP	RTSP	SLIP	SMTP	SNMP	SNTP	SSL	Telnet	TFTP	UDP	802.11
CMX Systems	CMX-MicroNET	TCP/IP	•	•							•	•	•					•					•					•		•	
CMX Systems	CMX Add Ons	Networking SW Options					•	•	•	•							•	•					•	•	•					•	•
Express Logic	NetX	TCP/IP	•								•	•	•							•								•		•	
Express Logic	NetX Add Ons	Networking SW Options		•		•	•	•	•	•						•	•	•					•	•	•			•	•		
Interneiche	NicheLITE	TCP/IP	•	•		•	•				•		•															•	•	•	
Interneiche	NicheStack	TCP/IP	•	•		•	•	•	•		•	•	•															•	•	•	
Interneiche	Interneiche Add Ons	Networking SW Options					•	•	•	•			•		•	•	•	•			•	•	•	•	•	•	•	•			
Micrium	µC/UDP-IP	UDP/IP	•		•						•		•																		•
Micrium	µC/TCP-IP	TCP/IP	•		•						•		•																•		•
Micrium	Micrium Add Ons	Networking SW Options					•	•	•	•						•							•	•	•			•	•		
SEVENSTAX	SEVENSTAX TCP/IP	TCP/IP									•															•	•			•	
SEVENSTAX	SEVENSTAX Add Ons	Networking SW Options	•	•		•	•	•	•				•		•	•	•						•								
SEGGER	embOS/IP	TCP/IP	•	•		•	•	•	•	•	•	•	•	•			•			•			•			•	•	•	•	•	
uIP	open source	TCP/IP	•								•	•	•															•		•	
lwIP	open source	TCP/IP	•	•			•	•			•	•	•				•						•					•		•	

Open Source TCP/IP Stacks

uip – Micro IP

- **Protocols supported**
 - Transmission Control Protocol (TCP)
 - User Datagram Protocol (UDP)
 - Internet Protocol (IP)
 - Internet Control Message Protocol (ICMP)
 - Address Resolution Protocol (ARP)
- **Memory requirements**
 - Typical code size on the order of a few kilobytes
 - RAM usage can be as low as a few hundred bytes.
 - Memory conserved by limiting to one outstanding transmit packet
- **Website**
 - <http://www.sics.se/~adam/uip>

uip and lwip licenses

- No restriction in shipping in real products
- Redistribution of stack source or binaries (such as in our kit) must carry copyright

lwip – Light-weight IP

- **Protocols supported**
 - Internet Protocol (IP) including packet forwarding over multiple network interfaces
 - Internet Control Message Protocol (ICMP) for network maintenance and debugging
 - User Datagram Protocol (UDP) including experimental UDP-lite extensions
 - Transmission Control Protocol (TCP) with congestion control, RTT estimations, and fast recovery/transmit
 - Dynamic Host Configuration Protocol (DHCP)
 - Point-to-Point Protocol (PPP)
 - Address Resolution Protocol (ARP) for Ethernet
 - Specialized raw API for enhanced performance
 - Optional Berkeley-like socket API
- **Memory Requirements**
 - Typical code size is on the order of 25 to 40 kilobytes
 - RAM requirements are approximately 15 to a few tens of kilobytes
- **Website**
 - <http://www.sics.se/~adam/lwip>
 - <http://savannah.nongnu.org/projects/lwip>

Stellaris APIs

Stellaris Ethernet APIs: Initialization and Interrupts

◆ Initialization

- EthernetInitExpClk()
- EthernetConfigSet(), EthernetConfigGet()
- EthernetMACAddrSet(), EthernetMACAddrGet()
- EthernetPHYWrite(), EthernetPHYRead()
- EthernetEnable(), EthernetDisable()

◆ Interrupts

- EthernetIntRegister(), EthernetIntUnregister()
- EthernetIntEnable(), EthernetIntDisable()
- EthernetIntStatus()
- EthernetIntClear()

Stellaris Ethernet APIs: Receive and Transmit

◆ Receive

- EthernetPacketAvail()
- EthernetPacketGet()
- EthernetPacketGetNonblocking()

◆ Transmit

- EthernetSpaceAvail()
- EthernetPacketPut()
- EthernetPacketPutNonBlocking()

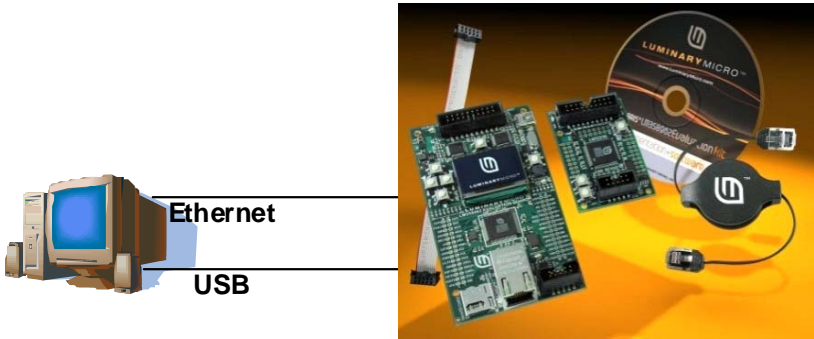
Lab ...

Ethernet Lab

Description:

We'll use the LM3S8962 eval board to connect to our PC via Ethernet. We'll take a look at the lwip stack and the I/O web server.

Ethernet Lab



- ◆ LM3S8962 EVM
- ◆ Ethernet Cable
- ◆ USB cable
- ◆ Control LM3S8962 via served web page

Hardware list:

- ✓ LM3S8962 Evaluation Board
- ✓ Retractable Ethernet cable
- ✓ 3 meter, 4 pin, USB Type A-M to mini-USB Type B-M

Software list:

- ✓ Code Composer Studio 4.1
- ✓ LM3S8962 software examples

Procedure

1. Open the IDE

Open **Code Composer Studio** and create a **workspace_8962ENET** workspace (we're using a new workspace for each lab to prevent issues and learn the steps by repetition). **Close** the Welcome screen if it appears. Click the **C/C++ Perspective** button to make sure that you're in the editing perspective. Close any open editor windows.

2. Import driverlib

Click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**. When the **Import** dialog appears, browse to the root directory of the driver library (**C:\StellarisWare\driverlib**) and click **OK**. Be sure that the checkbox next to **driverlib** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.

3. Import the Ethernet project

Click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**. When the **Import** dialog appears, browse to the root directory of the Ethernet project (**C:\StellarisWare\boards\ek-lm3s8962\enet_io**) and click **OK**. Be sure that the checkbox next to **enet_io** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.

4. Open file for editing

Maximize the IDE window if you haven't already. In the **Project Explorer** pane, **double-click** on **enet_io.c** to open it for editing.

Read the example description starting around line number 50. This code handles both IO control demos.

enet_io.c is made up of several modules:


ControlCGIHandler()	Called when the web browser requests LED or PWM control
SetTextCGIHandler()	Called when the web browser requests to write test to OLED
SSIHandler()	Called by the HTTP server when it encounters an SSI tag
DisplayIPAddress()	Displays the lwIP type IP address
SysTickIntHandler()	Handles the SysTick interrupt
lwIPHostTimerHandler()	Supports host timer functions
main()	Sets up the clock, ethernet and IO ports, inits the OLED, configures SysTick and enables interrupts

5. Test existing project

Before we change anything, let's make sure that the project works as it is.


Connect the LM3S8962 board to your PC via the USB cable, as usual. Then, using the Ethernet cable included in the kit, **connect** the LM3S8962 board to the Ethernet port on the back of your PC. If your PC's wireless connection is enabled, you should **disable** it now.

If you have problems connecting to the board in later steps, you may need to disable your firewall software.

Click the  **Debug** button now to build and load the project to your board.

6. Run

When the build and load completes, make sure that you are in the **Debug perspective**.

Click the  **Run** button to run the code on your board. You should see “**Web-Based I/O Control**” on the OLED display on the LM3S8962 board.

Minimize Code Composer Studio.

After 30 seconds or so, you should see the following on the OLED display (your IP address may be different):

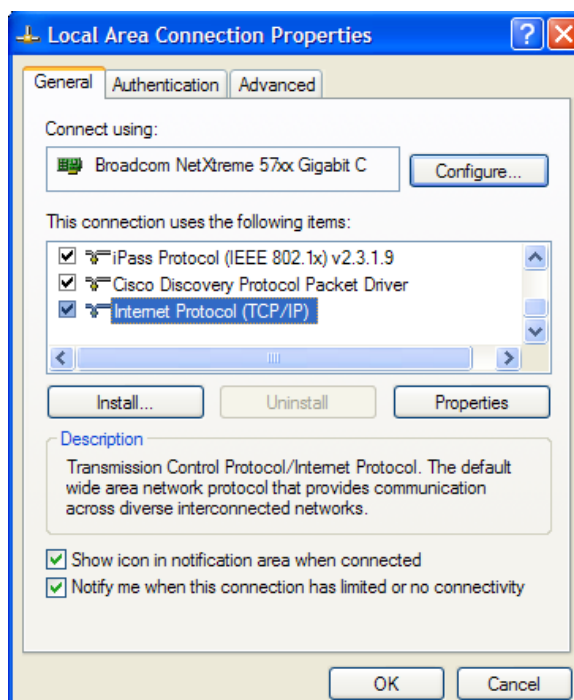
```
Web-Based I/O Control
IP:          169.254.254.108
MASK:        255.255.0.0
GW:          0.0.0.0
```

7. PC Network changes

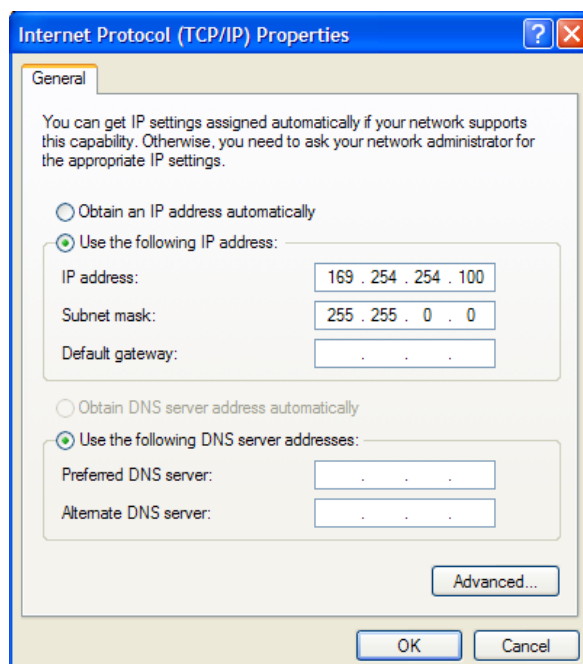
There are two possibilities here: either your PC and the board have compatible addresses, or not. To find out, click **Start → Control Panel → Network Connections**. **Right-click** on **Local Area Connection** and then select **Status**. Click on the **Support** tab to see your IP address. If the first **three** parts of your IP address are the **same** as the address shown on the LM3S8962 OLED display, you got lucky! The addresses are compatible and you can skip to **step 8**. If not, follow these steps:

The **lwIP** stack has defaulted to a preprogrammed IP address. In order to communicate with that address, we need to give the PC's Ethernet port a suitable address. Depending on the OS that you're using, this procedure may be slightly different.

Go to **Start → Control Panel → Network Connections → Local Area Connection**. Click the **Properties** button. When the **Local Area Connection Properties** window appears, scroll down until you see **Internet Protocol (TCP/IP)** like shown below:



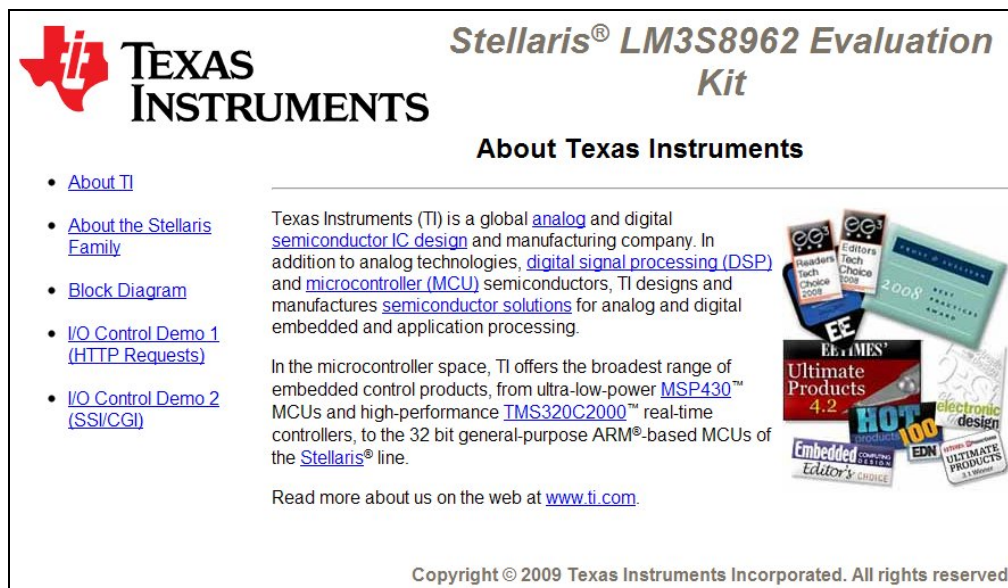
Click the **Properties** button. When the **Internet Protocol (TCP/IP) Properties** window appears, make a mental (or other) note of the settings you see, then make the selections shown below (if your boards' IP address was different, simply make sure that the first three fields are the same, and the fourth is different):



Click **OK**. Close the **Local Area Connection Properties** and **Local Area Connection Status** windows.

8. Start your browser

Start your web browser, and enter the address shown on your OLED display and press **Enter**. A web page served from the LM3S8962 should appear.



TEXAS INSTRUMENTS

Stellaris® LM3S8962 Evaluation Kit

About Texas Instruments

- [About TI](#)
- [About the Stellaris Family](#)
- [Block Diagram](#)
- [I/O Control Demo 1 \(HTTP Requests\)](#)
- [I/O Control Demo 2 \(SSI/CGI\)](#)

Texas Instruments (TI) is a global [analog](#) and digital [semiconductor IC design](#) and manufacturing company. In addition to analog technologies, [digital signal processing \(DSP\)](#) and [microcontroller \(MCU\)](#) semiconductors, TI designs and manufactures [semiconductor solutions](#) for analog and digital embedded and application processing.

In the microcontroller space, TI offers the broadest range of embedded control products, from ultra-low-power [MSP430™](#) MCUs and high-performance [TMS320C2000™](#) real-time controllers, to the 32 bit general-purpose ARM®-based MCUs of the [Stellaris®](#) line.

Read more about us on the web at www.ti.com.

Copyright © 2009 Texas Instruments Incorporated. All rights reserved.

If you are having issues seeing the web page on your browser, you likely do not have Java installed. Look on the flash installation drive and run the **jre-6u17-windows-i586-s.exe** Java offline installation file.

9. Trying out the controls

On the web page, **click** the link to **I/O Control Demo 1** and then press the **Toggle LED** button a few times, noting the LED on the LM3S8962 board. You may also note the green activity light on the LM3S8962 Ethernet connector as you press the button.

On the web page, **click** the link to **I/O Control Demo 2**. Type the text of your choice (keep it clean) in the **Display this text on the screen:** box and click **Send Text**. The text that you typed should appear on the OLED screen.

10. Make some changes in the embedded web-server

The embedded web server used in the **enet_io** example uses the **lwIP TCP/IP** stack. When you first start the application, the **index.html** file is displayed in your web browser.

In this part of the lab, we will modify this web page using notepad as our editor. The easiest way to provide the html files to the application is by putting them on a microSD card. The application is coded to look for the file system there first. In the workshop, we're not using a microSD card, so instead, we will create a new file system image to embed in the application itself. There is a command line tool in the **\StellarisWare\tools\bin** folder that will generate a header file with an array for each file in the **\fs** folder.

Exit your browser, then, using **Windows Explorer**, find **index.html** file in the **C:\StellarisWare\boards\ek-lm3s8962\enet_io\fs** folder (This is the original location of the files ... had we allowed Code Composer to copy the files to our workspace folder, we'd be editing those copied files instead). Right click on **index.html** and select **Open with**, then click **Notepad** to open the file for editing using Notepad.

11. Find/Replace

Find the lines near the bottom of the file that look like this:

```
</p>
<p>Read more about us on the web at <a href="http://www.ti.com"
target="_top">www.ti.com</a>.
<p></p>
<br>
```

Add some code so that it looks like this:

```
</p>
<p>Read more about us on the web at <a href="http://www.ti.com"
target="_top">www.ti.com</a>.
<p></p>
</p>
<p>WooHoo! YourName was here!
<p></p>
<br>
```

Save the file and **close** your editor.

12. Convert the HTML files to a Header File

In Code Composer Studio, examine the file **lmi_fs.c** in the **enet_io** project. You will find the command line and options that are needed to run the **makefsfile** utility in the comments towards the top of this file. The command line you will need is:

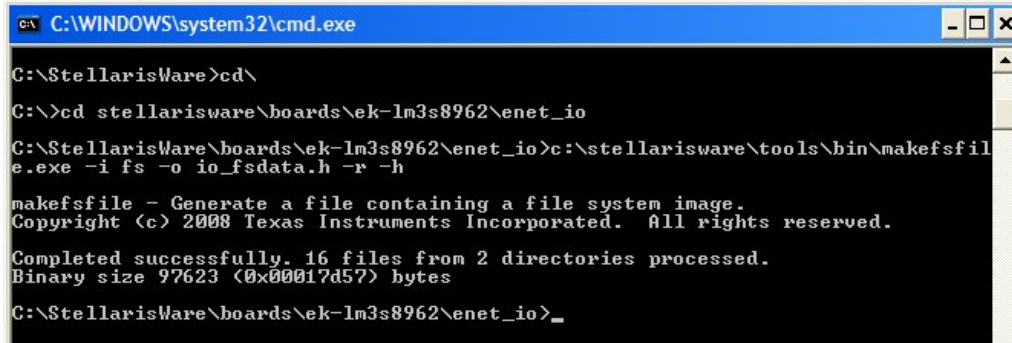
```
C:\StellarisWare\tools\bin\makefsfile.exe -i fs -o io_fsdata.h -r -h
```

Open a command prompt window using **Start → Run**, type **cmd** and click **OK**.

Type **cd** and press **Enter** to return to the root directory **C:**

Then type **cd stellarisware\boards\ek-lm3s8962\enet_io** and press **Enter**.


In the command window, type the command shown above to call the **makefsfile** utility. This will create a new **io_fsdata.h** header file which is included by **io_fs.c**. Close the command window when you are finished.

A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the following commands and output:

```
C:\StellarisWare>cd\  
C:\>cd stellarisware\boards\ek-lm3s8962\enet_io  
C:\StellarisWare\boards\ek-lm3s8962\enet_io>c:\stellarisware\tools\bin\makefsfile.exe -i fs -o io_fsdata.h -r -h  
makefsfile - Generate a file containing a file system image.  
Copyright (c) 2008 Texas Instruments Incorporated. All rights reserved.  
Completed successfully. 16 files from 2 directories processed.  
Binary size 97623 (0x00017d57) bytes  
C:\StellarisWare\boards\ek-lm3s8962\enet_io>_
```

13. Rebuild the enet_io Example Application

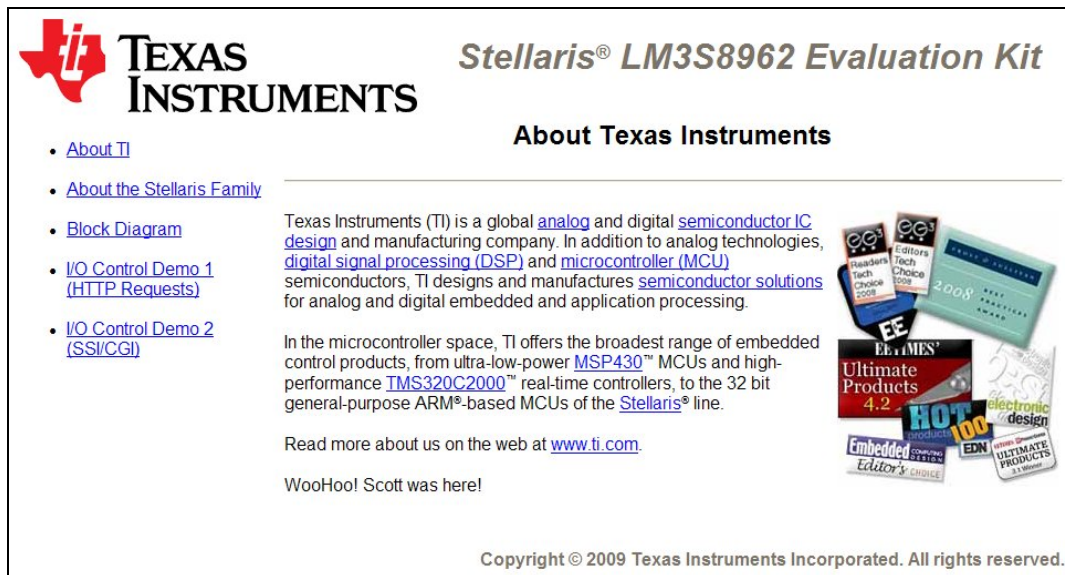
Maximize Code Composer Studio. We just modified one of the files in the project without the IDE knowing it, so we need to perform a clean build.

Return to the **C/C++ perspective**. Right-click on **enet_io** in the **Project Explorer** window and select **Clean Project**. Click the  **Debug** button to build/load the project.

Make sure your LM3S8962 evaluation board is connected and assure that you're in the **Debug** perspective.

14. Load the Modified Website in your Browser

Make sure your Ethernet cable is connected and **reset** the LM3S8962 evaluation board (**disconnect/reconnect** the USB cable). Wait for IP address to show up on your OLED and type the IP address into address bar:



15. Restore your network settings

Remember your original network settings on your PC? **Restore** those and re-enable your wireless connection (if necessary). **Close** any open windows on your desktop.

Remember that the QuickStart application has been replaced by this application. You can reprogram it at any time with either Code Composer Studio or the Flash Programmer.



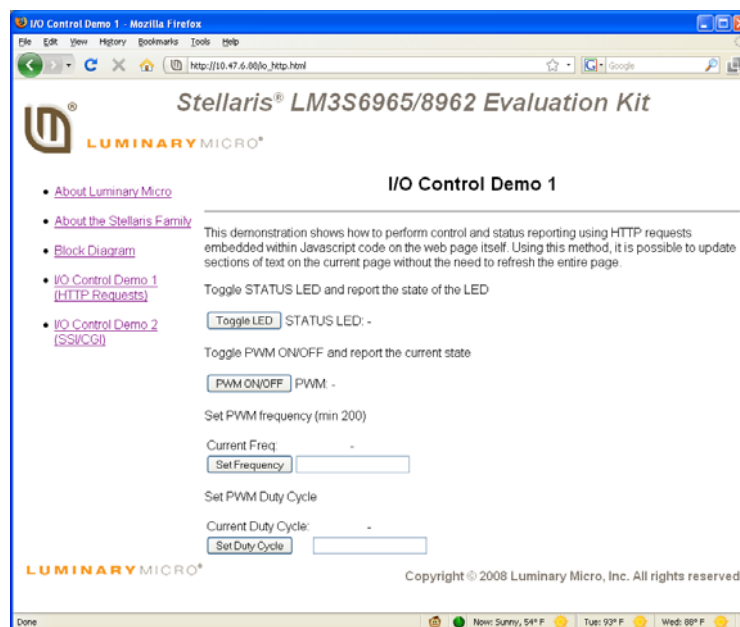
You're done

Additional Web Server Information

Hands On: I/O Control Demonstrations

- ◆ The enet_io application illustrates two methods of controlling board operations from the web browser:
 - I/O Control Demo 1 shows direct HTTP requests generated via JavaScript code in the web page (io_http.html).
 - I/O Control Demo 2 shows the use of Server Side Includes (SSI) and Common Gateway Interface (CGI) to perform the same operations (io_cgi.shtml)
- ◆ All web site files are stored as a file system image (io_fsdata.c) linked into the application image in flash.

Hands On: I/O Control Demo 1



Hands On: Demo 1– HTTP Requests

- ◆ JavaScript in the web page generates HTTP GET requests for specific filenames.
- ◆ lwIP web server passes these filenames to the file system (`lmi_fs.c`) which recognizes them as “special” and performs whichever task the filename is intended to trigger.
 - Toggle LED state (`/cgi-bin/toggle_led`)
 - Set and report PWM frequency (`/pwm_freq?value=<frequency>`)
- ◆ JavaScript reads the response from the web server and uses `<div>` tags to insert the relevant text into the displayed page.

Hands On: Demo 1– HTTP Requests

- ◆ **Advantages:**
 - Updates can be made without having to reload the whole web page.
 - Quicker user feedback.
 - Lower network traffic.
- ◆ **Disadvantages:**
 - Client browser must support JavaScript.
 - HTML files are more complex to develop and less easy to understand.

Hands On: Embedded Control

- ◆ Below is an example of the html code used for the Toggle LED button from "io_http.html"

```
<table>
<tr>
<td>
<input id="toggle" value="Toggle LED"
onclick="toggle_led()" type="button">
</td>
<td>
STATUS LED:
</td>
<td>
<div id="ledstate" align="center"> - </div>
</td>
</tr>
</table>
```

Hands On: Embedded Control

- ◆ Below is an example of the JavaScript code used for the Toggle LED button from "io_http.html"

```
function toggle_led()
{
var req = false;
var led = false;

function ledComplete() {
if(led.readyState == 4) {
if(led.status == 200) {
document.getElementById("ledstate").innerHTML = "<div>" + led.responseText +
"</div>";
}
}
}

if(window.XMLHttpRequest){
req = new XMLHttpRequest();
led = new XMLHttpRequest();
}
else if(window.ActiveXObject) {
req = new ActiveXObject("Microsoft.XMLHTTP");
led = new ActiveXObject("Microsoft.XMLHTTP");
}

if(req){
req.open("GET", "/cgi-bin/toggle_led?id" + Math.random(), true);
req.send(null);
}

if(led){
led.open("GET", "/ledstate?id=" + Math.random(), true);
led.onreadystatechange = ledComplete; led.send(null);
}
}
```

Hands On: Embedded Control

- ◆ Below is an example of the C source code from `lmi-fs.c` that handles the request for toggling the LED

```
//
// Process request to toggle STATUS LED
//
if(strncmp(name, "/cgi-bin/toggle_led", 19) == 0)
{
    //
    // Toggle the STATUS LED
    //
    io_set_led(!io_is_led_on());

    //
    // Setup the file structure to return whatever.
    //
    ptFile->data = NULL;
    ptFile->len = 0;
    ptFile->index = 0;
    ptFile->pextension = NULL;

    //
    // Return the file system pointer.
    //
    return(ptFile);
}
```

Hands On: Demo 2 – SSI/CGI

The screenshot shows a web browser window titled "I/O Control Demo 2 - Mozilla Firefox" displaying the "Stellaris® LM3S6965/8962 Evaluation Kit" interface. The page has a sidebar with links: "About Luminary Micro", "About the Stellaris Family", "Block Diagram", "I/O Control Demo 1 (HTTP Requests)", and "I/O Control Demo 2 (SSI/CGI)". The main content area is titled "I/O Control Demo 2" and contains a paragraph explaining the demo. Below the text is a table for controlling the LED and PWM settings.

Control	Current	New
LED State	OFF	<input checked="" type="checkbox"/>
PWM State	OFF	<input type="checkbox"/>
PWM Frequency (Hz)	440	<input type="text" value="1200"/>
PWM Duty Cycle (%)	50	<input type="text" value="25"/>

Below the table is an "Update Settings" button. Further down is a section titled "Display this text on the screen:" with a text input field containing "Hello World!" and a "Send Text" button. The footer of the page includes the Luminary Micro logo and copyright information: "Copyright © 2008 Luminary Micro, Inc. All rights reserved."

Hands On: Demo 2 – SSI/CGI

- ◆ HTML pages include “Server Side Include” tags indicating values to be inserted in the page data as it is served to the browser.
- ◆ The application registers SSI and CGI handlers with the HTTP server during initialization.
- ◆ The HTTP server calls the SSI handler when a tag from the registered list is detected and handler returns the text to insert after the tag.
- ◆ The HTTP server calls the registered CGI handler if a URL matching the registered CGI name is requested.
- ◆ The HTML contains standard forms to gather user input.

Hands On: Demo 2 – SSI/CGI

- ◆ **Advantages**
 - Client browser need not support JavaScript.
 - HTML is extremely simple and uses only standard forms and some “comment-like” SSI tags (`<!--#<tag>-->`).
 - Offloads work to the common HTTP server module (URL checking, parameter parsing).
 - File system driver is independent of the application data that it is managing.
- ◆ **Disadvantages**
 - Page reload each time a form is submitted.

Hands On: Demo 2 – SSI/CGI

- ◆ Below is an example of the html code used to send a text string to the browser in “io_cgi.shtml”

```
<form method="get" action="settxt.cgi" name="settxt">
<table>
<tr>
<td>
<b>Display this text on the screen:<br></b>
<input maxlength="20" size="20" name="DispText">
<input name="Display" value="Send Text" type="submit">
</td>
</tr>
</table>
</form>
```

Hands On: Demo 2 – SSI/CGI

- ◆ Below is an example of the html code from “io_cgi.shtml” which uses SSI tags to insert status information.

```
<tr>
<td>LED State</td>
<td>
<!--#LEDtxt-->
</td>
<td>
<input name="LEDon" value="1" type="checkbox">
</td>
</tr>
<tr>
<td>PWM State</td>
<td>
<!--#PWMtxt-->
</td>
<td>
<input name="PWMon" value="1" type="checkbox">
</td>
</tr>
```

The HTTP server parses the SSI tags from the page as it is being sent and passes them to the application SSI handler.

The application returns an appropriate block of text which is inserted into the page following the comment tag.

Note: The tag is *not* replaced. The SSI text is inserted after the closing “-->” This is different from typical SSI implementations but allows operation without additional buffering.

“<!--#LEDtxt-->” in the source is expanded to, for example “<!--#LEDtxt-->OFF”

Hands On: Registering SSI Handlers

```
// SSI tag indices for each entry in the g_pcSSITags array.
#define SSI_INDEX_PWMDDUTY 0
#define SSI_INDEX_PWMSTATE 1

// This array holds all the strings that are to be recognized as SSI tag
// names by the HTTPD server. The server will call SSIHandler to request a
// replacement string whenever the pattern <!--#tagname--> (where tagname
// appears in the following array) is found in ".ssi", ".shtml" or ".shtm"
// files that it serves.
static const char *g_pcConfigSSITags[] =
{
    "PWMDDuty",      // SSI_INDEX_PWMDDUTY
    "PWMState",      // SSI_INDEX_PWMSTATE
};

// The number of individual SSI tags that the HTTPD server can expect to
// find in our configuration pages.
#define NUM_CONFIG_SSI_TAGS (sizeof(g_pcConfigSSITags) / sizeof(char *))

// Prototype for the main handler used to process server-side-includes for the
// application's web-based configuration screens.
static int SSIHandler(int iIndex, char *pcInsert, int iInsertLen);
```

After initializing the HTTPD server, register the handler by calling...

```
// Register our SSI tags and handler with the HTTP server.
http_set_ssi_handler(SSIHandler, g_pcConfigSSITags, NUM_CONFIG_SSI_TAGS);
```

Hands On: The SSI Handler Function

```
// This function is called by the HTTP server whenever it encounters an SSI
// tag in a web page. The iIndex parameter provides the index of the tag in
// the g_pcConfigSSITags array. This function writes the substitution text
// into the pcInsert array, writing no more than iInsertLen characters.

static int SSIHandler(int iIndex, char *pcInsert, int iInsertLen)
{
    unsigned long ulVal;

    // Which SSI tag have we been passed?
    switch(iIndex)
    {
        case SSI_INDEX_PWMSTATE:
            // Write the PWM state (ON or OFF) into the supplied insert buffer.
            io_get_pwmstate(pcInsert, iInsertLen);
            break;

        case SSI_INDEX_PWMDDUTY:
            // Get the current PWM duty cycle.
            ulVal = io_get_pwmddutycycle();

            // Write it as an ASCII string into the supplied insert buffer.
            usprintf(pcInsert, iInsertLen, "%d", ulVal);
            break;
    }

    // Tell the server how many characters our insert string contains.
    return(strlen(pcInsert));
}
```


Hands On: Registering CGI Handlers

```
// Prototypes for the various CGI handler functions.
static char *ControlCGIHandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[]);
static char *SetTextCGIHandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[]);

// CGI URI indices for each entry in the g_psConfigCGIURIs array.
#define CGI_INDEX_CONTROL 0
#define CGI_INDEX_TEXT 1

// This array is passed to the HTTPD server to inform it of special URIs
// that are treated as common gateway interface (CGI) scripts. Each URI name
// is defined along with a pointer to the function which is to be called to
// process it.
static const tCGI g_psConfigCGIURIs[] =
{
    { "/iocontrol.cgi", ControlCGIHandler }, // CGI_INDEX_CONTROL
    { "/settxt.cgi", SetTextCGIHandler } // CGI_INDEX_TEXT
};

// The number of individual CGI URIs that are configured for this system.
#define NUM_CONFIG_CGI_URIS (sizeof(g_psConfigCGIURIs) / sizeof(tCGI))
```

After initializing the HTTPD server, register the handlers by calling...

```
// Register our CGI handlers with the HTTP server.
http_set_cgi_handlers(g_psConfigCGIURIs, NUM_CONFIG_CGI_TAGS);
```

Hands On: A CGI Handler Function

```
// This CGI handler is called whenever the web browser requests URI /iocontrol.cgi.
static char *ControlCGIHandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[])
{
    tBoolean bParamError;
    long lPWMState, lPWMFrequency;

    // We have not encountered any parameter errors yet.
    bParamError = false;

    // Get each of the expected parameters.
    lPWMState = FindCGIParameter("PWMOn", pcParam, iNumParams);
    lPWMFrequency = GetCGIParam("PWMFrequency", pcParam, pcValue, iNumParams,
                                &bParamError);

    // Was there any error reported by the parameter parser?
    if(bParamError || (lPWMFrequency < 200) || (lPWMFrequency > 20000))
    {
        // Return the URI of the parameter error page.
        return("/perror.html");
    }

    // We got all the parameters and the values were within the expected ranges
    // so go ahead and make the changes.
    io_pwm_freq((unsigned long)lPWMFrequency);
    io_set_pwm((lPWMState == -1) ? false : true);

    // Send back the default response page.
    return("/io_cgi.shtml");
}
```

*** It's kind of a "visual time-out" ***

USB Peripheral Module

Introduction

This module covers the USB peripheral, library and examples provided with the LM3S3748 evaluation kit.

Learning Objectives

- USB Basics
- Stellaris Implementation
- USB Library
- USB Examples
- Two labs

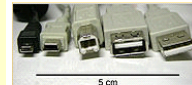
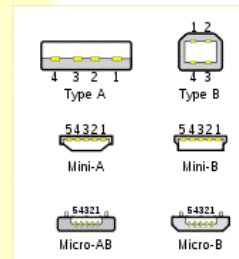
Module Topics

USB Peripheral Module.....	1
<i>Module Topics.....</i>	<i>2</i>
<i>USB Basics.....</i>	<i>3</i>
Stellaris USB	4
USB Stacks	5
Stellaris MCU's with USB	6
LM3S3748 USB Schematic.....	7
USB Library	7
USB Code Examples	8
Labs	9
<i>USB Mass Storage Host Lab.....</i>	<i>11</i>
Description:	11
Hardware list:	11
Software list:.....	11
Procedure.....	12
<i>USB HID Mouse Device Lab</i>	<i>16</i>
Description:	16
Hardware list:	16
Software list:.....	16
Procedure.....	17

USB Basics

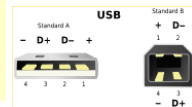
USB Basics

- ◆ **Multiple connector sizes**
- ◆ **4 pins – power, ground and 2 data lines**
(5th pin ID for USB 2.0 connectors)
- ◆ **Configuration connects power 1st, then data**
- ◆ **Standards:**
 - ◆ **USB 1.1**
 - Defines **Host** (master) and **Device** (slave)
 - Speeds to 12Mbps/sec
 - Devices can consume 500mA (100mA for startup)
 - ◆ **USB 2.0**
 - Speeds to 480Mbps/sec
 - OTG addendum
 - ◆ **USB 3.0**
 - Speeds to 4.8Gbps/sec
 - New connector(s)
 - Separate transmit/receive data lines



Different types of USB connectors from left to right

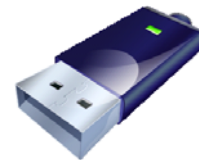
- 8-pin AGO[citation needed]
- Mini-B plug
- Type B plug
- Type A receptacle
- Type A plug



USB Tutorial: http://www.computer-solutions.co.uk/info/Embedded_tutorials/usb_tutorial.htm

USB Basics

- ◆ **USB Device ... most USB products are slaves**
- ◆ **USB Host ... usually a PC, but can be embedded**
- ◆ **USB OTG ... On-The-Go**
 - Allows a single port to be either a Device (a printer connected to a PC) or a Host (a printer connected to a camera)
 - Two connected OTG ports undergo host negotiation
- ◆ **Host polls each Device at power up. Info from Device includes:**
 - Device Descriptor (Manufacturer & Product ID so Host can find driver)
 - Configuration Descriptor (Power consumption and Interface descriptors)
 - Endpoint Descriptors (Transfer type, speed, etc)
- ◆ **Process is called *Enumeration* ... allows Plug-and-Play**



Stellaris USB ...

Stellaris USB

Stellaris USB

- ◆ USB 2.0 Full Speed (12 Mbps) operation
- ◆ Transfer types: Control, Interrupt, Bulk and Isochronous

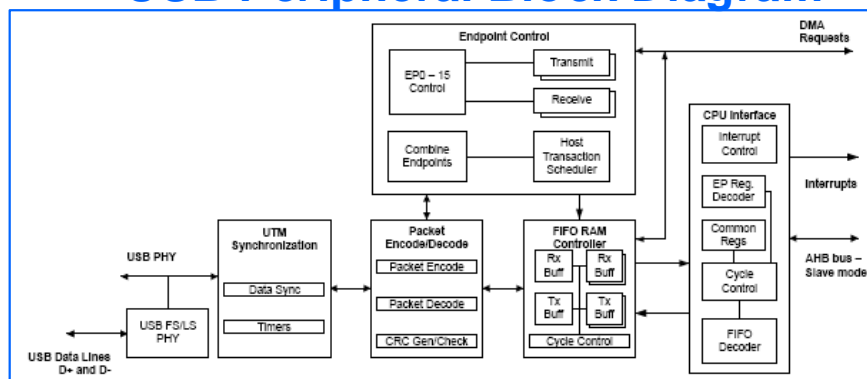
Stellaris collaterals

- ◆ Luminary Micro is a member of the USB Implementers Forum.
- ◆ Stellaris is approved to use the USB logo
- ◆ Vendor/Product ID sharing



Block Diagram ...

USB Peripheral Block Diagram



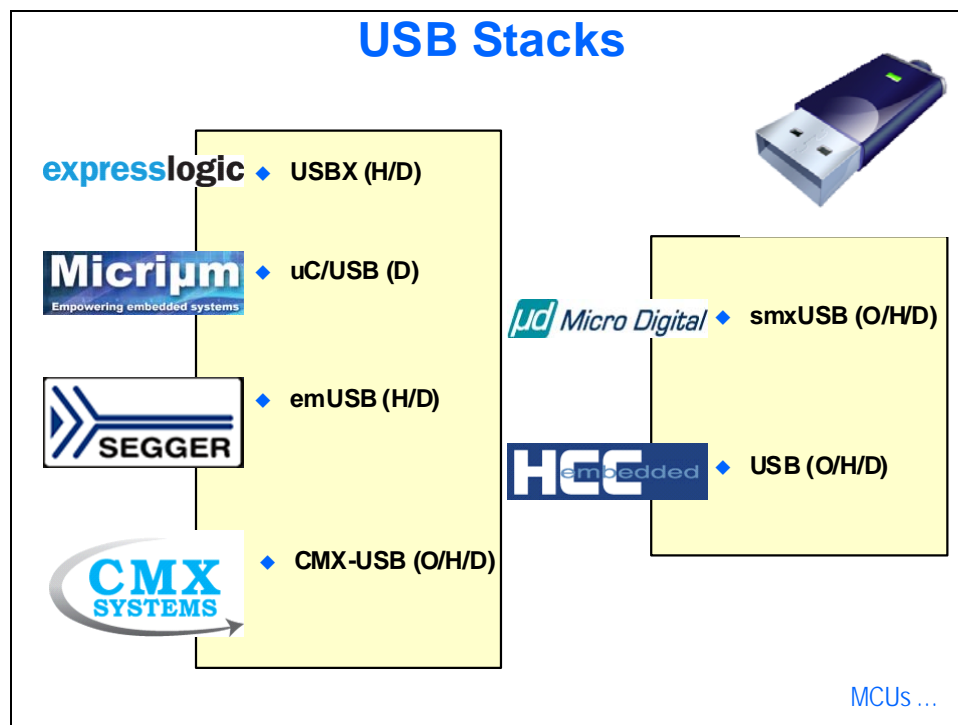
◆ Integrated USB Controller and PHY

◆ Up to 32 Endpoints

- ◆ 1 dedicated control IN endpoint and 1 dedicated control OUT endpoint
- ◆ Up to 15 configurable IN endpoints and 15 configurable OUT endpoints
- ◆ 4 KB Dedicated Endpoint Memory
- ◆ DMA capability (up to three IN Endpoints and three OUT Endpoints)
- ◆ 1 endpoint may be defined for double-buffered 1023-bytes isochronous packet size

Stacks ...

USB Stacks



Stellaris MCU's with USB

USB Connected MCUs

	MCU Part Series	Memory and Speed			Core	General Purpose Timer Modules					Motion Control		Serial Interfaces					Analog					Digital		Package Options						
		Flash (KB)	SRAM (KB)	ROM (KB)		Max Speed (MHz)	Precision Oscillator	32-bit Timer	16-bit Timer	Watchdog	CCP	RTC	PWM	10/100 Ethernet MAC+PHY	IEEE 1588	CAN/MAC	USB Full Speed	UART	FC	SS/SP	I2S	ADC Channels	ADC Speed (Ksps)	Internal Temp Sensor		LD Voltage Regulator	Analog Comparators	Digital Comparators	GPIOs (5-V)	Hibernate	
LM3S3700s	4	128	64	?	50	-	4	8	1	8	?	8	4	1	-	-	OH/D	3	2	2	-	8	1000	?	?	3	-	61	?	64-LQFP 100-LQFP	
LM3S5000s	12	256	96	?	80	?	4	8	2	8	?	8	4	2	-	-	2	OH/D	3	2	2	?	16	1000	?	?	3	7	71	?	64-LQFP 100-LQFP
LM3S9000s	6	256	96	?	100	?	4	8	2	8	?	8	4	2	?	2	OH/D	3	2	2	?	16	1000	?	?	3	7	65	?	100-LQFP	

- ◆ USB 2.0 full speed (12 Mbps) support (MAC+PHY) : Host/Device/OTG – with DMA
- ◆ Compact 64-pin LQFP or feature-rich 100-pin LQFP options

Stellaris LM3S3748 Evaluation Kit



- CSTN graphics display (128x128 resolution, 16-bit color)
- User LED, navigation switch and pushbutton
- Magnetic speaker
- LM3S3748 I/O available on labeled break-out pads
- Standard ARM® 20-pin JTAG debug connector with input and output modes
- MicroSD card slot

- ◆ USB and JTAG cables, Jumper wires
- ◆ USB Flash Drive (128MB)

Ethernet+USB OTG Connected MCUs

	MCU Part Series	Memory and Speed		Core	General Purpose		Motion Control		Serial Interfaces					Analog			Digital		Package Options												
		Flash (KB)	SRAM (KB)		Time (μs)	Time (μs)	PWM	Encoder	UART	I2C	FS	ADC Channels	ADC Speed (kSps)	Internal Temp Sensor	LD/Voltage Regulator	Analog Comparators	Digital Comparators														
		Flash (KB)	SRAM (KB)	ROM (KB)	Max Speed (MHz)	Internal Precision Oscillator	32-bit Timer	16-bit Timer	Watchdog Time (s)	CCP	RTC	Outputs	Fault Inputs	CEI	10/100 Ethernet MAC+PHY	IEEE 1588	CAN MAC	USB Full Speed	UART	FS	SS/SP	I2S	ADC Channels	ADC Speed (kSps)	Internal Temp Sensor	LD/Voltage Regulator	Analog Comparators	Digital Comparators	GPIOs (5-V)	Hibernate	Package Options
LM3S3700s	4	128	64	?	50	?	4	8	1	8	?	8	4	1	?	-	0/H/D	3	2	2	?	8	100.0	?	?	3	-	61	?	64-LQFP 100-LQFP	
LM3S5000s	12	256	96	?	80	?	4	8	2	8	?	8	4	2	?	2	0/H/D	3	2	2	?	16	100.0	?	?	3	7	71	?	64-LQFP 100-LQFP	
LM3S9000s	6	256	96	?	100	?	4	8	2	8	?	8	4	2	?	2	0/H/D	3	2	2	?	16	100.0	?	?	3	7	65	?	100-LQFP	

- ◆ First MCUs featuring fully integrated 10/100 Ethernet MAC+PHY, USB OTG MAC+PHY, and up to 2 Bosch CAN 2.0 A/B MACs
- ◆ LM3S9B96 features SAFERTOS in ROM



EK-LM3S9B90



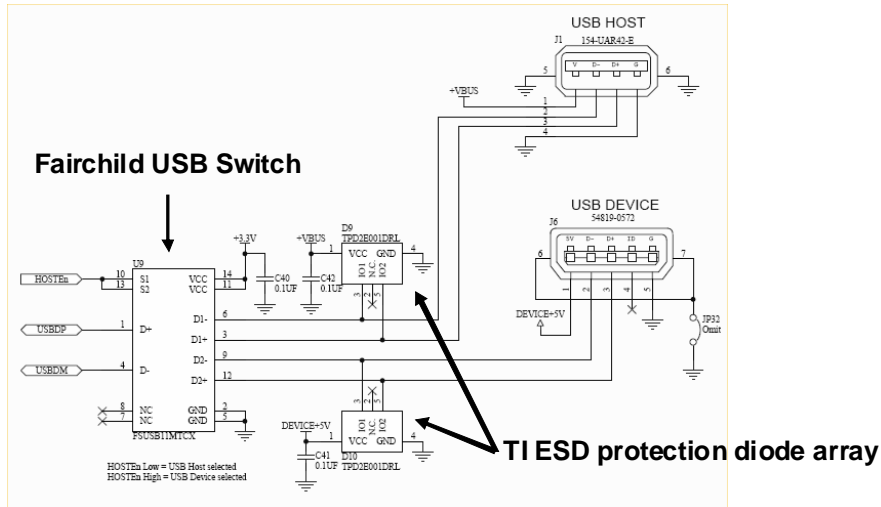
EK-LM3S9B92

- ◆ Evaluation board with LM3S9B90 (hibernate) or LM3S9B92 (max GPIOs)
- ◆ In-Circuit Debug Interface (BD-ICDI) board
 - Connects to USB port on PC and to 10-pin ARM JTAG connector on the evaluation board
 - 8-pin Power/UART connector provides power and virtual comm-port to the evaluation board
- ◆ Cables
- ◆ CD with evaluation software tools, documentation, source code, schematics

Schematic ...

LM3S3748 USB Schematic

LM3S3748 Eval Board USB Schematic



USBLib ...

USB Library

Stellaris USB Library

- ◆ Supports Host, Device and On-The-Go ports
- ◆ Written entirely in C
- ◆ User Guide: SW-UG-4781.pdf
- ◆ Supports the following tool chains:
 - ◆ Keil
 - ◆ CodeSourcery
 - ◆ IAR Embedded Workbench
 - ◆ Code Red Technologies
 - ◆ Code Composer Studio



Code Examples ...

USB Code Examples

USB Code Examples

USB Boot Loader Demo 1
USB Boot Loader Demo 2
USB Generic Bulk Device
USB HID Keyboard Device
[USB HID Mouse Device](#)
USB MSC Device
USB Serial Device
USB HID Keyboard Host
USB HID Mouse Host
[USB Mass Storage Class Host](#)

Location: C:\StellarisWare\boards\lek-lm3s3748

USB Code Examples

- ◆ Stellaris examples implement a custom USB “micro-stack”
- ◆ Stack footprint is about 10K in most configurations
- ◆ Application code sizes
 - usb_dev_mouse (HID) – 17kB
 - usb_dev_msc (MSC) – 16kB
 - usb_dev_serial (CDC) – 18kB
 - usb_host_mouse – 14kB
 - usb_host_msc – 21kB
- ◆ USB Bootloader (DFU) - <6k
- ◆ USB Flash stick updater – 10kB

[Lab ...](#)

Labs

USB Mass Storage Class Host Lab

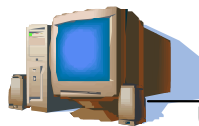
- ◆ LM3S3748 EVM
- ◆ USB Flash Drive
- ◆ USB cable



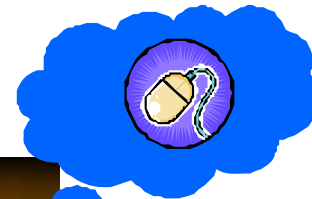
- ◆ LM3S3748 accesses Flash Drive through USB port as a Host
- ◆ Uses FatFs: An open-source file system for embedded designs
 - Supports FAT12, FAT16 (with 8.3) and FAT32
 - Multiple volumes
 - FAT32 & Long File Name (LFN) support requires license from Microsoft
 - http://elm-chan.org/fsw/ff/00index_e.html

Lab ...

USB HID Mouse Device Lab



USB



- ◆ LM3S3748 EVM
- ◆ USB Flash Drive
- ◆ USB cable

- ◆ Code sends Descriptor structures and enumerates as a mouse
- ◆ Code sends Mouse movements and button press to the PC Host

TI ...

*** Left on its own, this page would have been blank ***

USB Mass Storage Host Lab

Description:

We'll use the LM3S3748 MCU to access a Flash drive connected to the USB Host port of the evaluation board.

Hardware list:

- ✓ LM3S3748 Evaluation Board
- ✓ USB Flash Drive with some files onboard (anything will do)
- ✓ 3 meter, 4 pin, USB Type A-M to mini-USB Type B-M

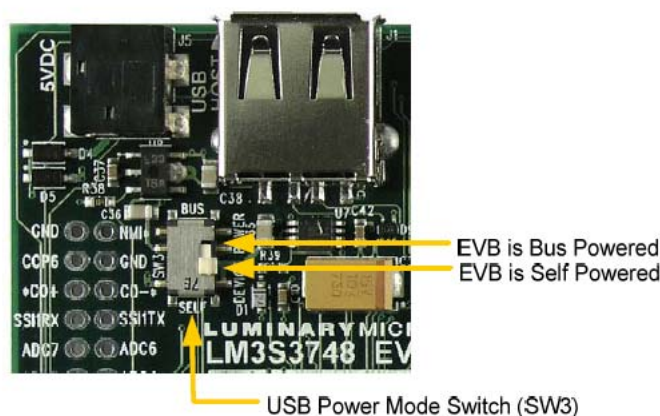
Software list:

- ✓ Code Composer Studio
- ✓ LM3S3748 software examples

Procedure

1. LM3S3748 EVM

Check the **USB power mode switch** on the board. We want to power the board from the **DEBUG USB** port. The switch should be in the “**SELF**” position as shown below. Don’t reposition the switch while power is applied as you may damage the switch contacts.



2. Connect the ICDI USB Cable

Connect the small end of the USB cable to the connector on the **LM3S3748** board marked “**DEBUG USB**”. You’ll find it on the corner of the board furthest away from the micro-processor. **Make sure that you’re using the LM3S3748 board.**

Connect the other end of the USB cable to an open port on your PC. Since you already have the drivers loaded, your PC should recognize the board quickly.

3. Open the IDE

Open **Code Composer Studio** and create a **workspace_3748USB** workspace (we’re using a new workspace for each lab to prevent issues and learn the steps by repetition). **Close** the Welcome screen if it appears. Click the **C/C++ Perspective** button to make sure that you’re in the editing perspective.

Maximize Code Composer Studio.

4. Import driverlib

Click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**. When the **Import** dialog appears, browse to the root directory of the driver library (**C:\StellarisWare\driverlib**) and click **OK**. Be sure that the checkbox next to **driverlib** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.

5. Import grlib

Click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**. When the **Import** dialog appears, browse to the root directory of the graphics library (C:\StellarisWare\grlib) and click **OK**. Be sure that the checkbox next to **grlib** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.

6. Import usblib

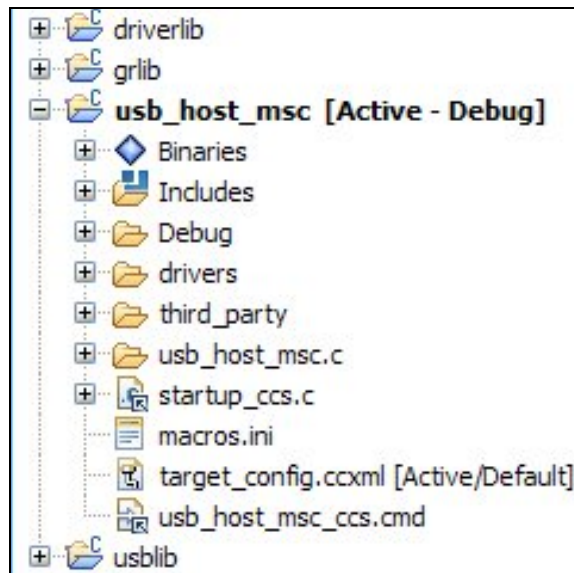
Click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**. When the **Import** dialog appears, browse to the root directory of the USB library (C:\StellarisWare\usblib) and click **OK**. Be sure that the checkbox next to **usblib** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.

7. Import the usb_host_msc project

Click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**. When the **Import** dialog appears, browse to the root directory of the project library (C:\StellarisWare\boards\ek-lm3s3748\usb_host_msc) and click **OK**. Be sure that the checkbox next to **usb_host_msc** in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.

8. Project Contents

Click on the “+” next to **usb_host_msc** and your Project Explorer should look like this:



The included files provide the following functionality:

startup_ccs.c – Startup code to set Heap, Stack and basic interrupts

Expand the **usb_host_msc.c** folder:

usb_host_msc.c – Main example code

Expand the **drivers** folder:

buttons.c – Functions for handling the on-board push buttons

formike128x128x16.c – LCD display driver code

Expand the **third_party**, then **fatfs**, then the **port** and **src** folders:

fat_usbmsc.c – Functions to access FAT file system via Luminary USB stack

ff.c – Underlying FAT (File Allocation Table) file system code from FatFs

9. Open usb_host_msc.c for editing

Double-click on **usb_host.msc.c** in the **Project Explorer** pane to open it for editing. **Maximize** the editor window if it hasn't opened that way already.

Click anywhere inside the code. On the menu bar, click **Edit, Find/Replace** and enter **main(** in the **Find:** box. Click **Find**, then **Close**. You should see where the **main()** code begins.

The next 150 lines or so performs the overall initialization. In that code, the initial state of **g_eState** (a global variable for the enumeration state) is set to **STATE_NO_DEVICE** . Later in the initialization code, **GrStringDraw()** (a graphics library call) is used to write “**No Device**” to the LCD display. Also note that a **GPIOPinWrite()** call sets the USB switch shown earlier in the schematic to the **Host** port.

Use the steps above to find **while(1)**.

A **switch()** statement switches between three states depending on **g_eState** (defined to have three states; **STATE_NO_DEVICE**, **STATE_DEVICE_ENUM** and **STATE_DEVICE_READY**). The earlier initialization code set **g_eState = STATE_NO_DEVICE**. The **MSCCallback()** code above **main()** (a callback from the lower level USB driver) sets the state of **g_eState** during runtime to either **STATE_NO_DEVICE** or **STATE_DEVICE_ENUM**. This callback function was registered with the driver in the line containing **USBMSCDriveOpen()**.


The first case in the **while()** loop, run during USB enumeration, detects whether enumeration has been successful and sets **g_eState** to **STATE_DEVICE_READY** (or it errors out). It also reads the directory (**DirUpdate()**) and writes the result to the LCD display (**UpdateWindow()**).

The code in the **STATE_DEVICE_READY** case occurs after enumeration is complete and the USB device is ready. Note how the buttons move through the directory.

10. Build and Load

If the Console pane is not visible at the bottom of your display, click **View → Console** from the menu bar.

Make sure that your USB Flash drive is **NOT** plugged into the EVM at this time.


Click the  **Debug** button on the menu bar to build/load the project to the LM3S3748 board. Monitor the progress of the build in the Console pane.




The program will **load** to the 3748 Flash memory and the perspective will switch to **Debug**. (if it doesn't, click the **Debug perspective** button in the upper right of your display) Once the load process completes, **click** the  **Run** button to start program execution.

11. Test the Code

The LCD display should have the top banner with the example code name on it, along with “**No Device**”. Plug in your USB Flash drive to the Host connector on the board and the directory should be displayed. (if you don't see any files, maybe there are no files on the flash drive ... copy some onto it from your laptop and try again). Due to licensing issues, FatFS only displays 8.3 filenames. Use the Navigation button below the LCD display to move up and down through the directory. If you have any sub-directories on the drive, you can highlight them and press the navigation button straight down to move into them.

12. Close the Project

Stop debugging by pressing the  **Terminate All** button on the menu bar. If it hasn't already done so, switch the IDE back to the editing perspective by clicking the **C/C++**

button   **Debug**  **C/C++** on the far right of the menu bar. **Collapse** the view of the **usb_host_msc** project in the **Project Explorer**, close any editor windows and **disconnect** your USB Flash drive from the evaluation board.

USB HID Mouse Device Lab

Description:

We'll emulate a USB mouse device with the LM3S3748EVM through the boards USB Device port to our PC host.

Hardware list:

- ✓ LM3S3748 evaluation board
- ✓ 3 meter, 4 pin, USB Type A-M to mini-USB Type B-M

Software list:

- ✓ Code Composer Studio
- ✓ LM3S3748 software examples

Procedure

1. Delete the `usb_host_msc` project

Since we already have all the supporting files imported, we can just delete the previous code example and import the next, rather than reload everything from zero. **Right-click** on `usb_host_msc` in the **Projects** pane and select **Delete**. Make sure that **Do not delete contents** is checked (if not, the original example files would be deleted) and click **Yes**.

2. Import the `usb_dev_mouse` project

Click on **Project**, and then select **Import Existing CCS/CCE Eclipse Project**. When the **Import** dialog appears, browse to the root directory of the project library (`C:\StellarisWare\boards\ek-lm3s3748\usb_dev_mouse`) and click **OK**. Be sure that the checkbox next to `usb_dev_mouse` in the Project pane is **checked** and that **Copy projects into workspace** is **unchecked**. Click **Finish**.

3. Open `usb_mouse_structs.c` for editing

Expand the `usb_dev_mouse` project and the `usb_mouse_structs.c` folder. Remember those descriptors that a USB Device provides to the host during enumeration? Open the `usb_mouse_structs.c` file for editing and check out the manufacturer, product, serial number, interface and configuration structures.

4. Open `usb_dev_mouse.c` for editing

Expand the `usb_dev_mouse.c` folder and open the `usb_dev_mouse.c` file for editing. Look around in the code at the initialization, graphics, etc, but most of the action occurs in the `ButtonHandler()` function about halfway through the code. This code is run from the `main()` loop each time the `BUTTON_TICK_EVENT` occurs. Look a little further down to see where the code formats the mouse movements and button presses for transfer to the host.

5. Build/Download the code

Similar to the previous procedure, **build** and **download** the project to the evaluation board. Don't run the code yet.

6. Connect the Board

Disconnect the ICDI USB cable from the **DEBUG** port of the evaluation board.

Check the **USB power mode switch** on the board (refer to the earlier illustration). Switch it to the "**BUS**" position so that power for the evaluation board will be provided through the USB Device port.

Connect the USB cable to the **USB DEVICE** port on the EVM (just left of the LCD display).

7. Test the Code

The board should power up, and you should see “**Waiting for Host ...**” on the LCD display. As soon as your PC recognizes the boards as a mouse, “**Host connected ...**” should appear on the display.

Use your PC’s mouse to position the cursor in the middle of your computer screen, and then use the **Navigation** button on the LM3S3748 board to move it around the screen. You can press the button **down** to **select**. Note that the LCD display shows the status of the Navigation button.

8. Turn it Off

Disconnect the LM3S3748 evaluation board from your PC and switch the **USB power mode switch** back to “**SELF**”. Switch Code Composer Studio back to **C/C++** perspective and **exit** the IDE.

Remember that the QuickStart application has been replaced by this application. You can reprogram it at any time with either Code Composer Studio or the Flash Programmer.



You're done

Introduction

Thanks for attending the workshop. Make sure to take all your belongings and handouts with you. Good luck with your project!

Don't Forget!

- ◆ Take your workbook and purchased board(s) home with you
- ◆ Please leave the installation flash drives on the table
- ◆ Fill out the evaluation form on line if possible:
www.tiworkshop.com and click Online feedback form
(use paper forms otherwise)
- ◆ Visit the TI Wiki for this workshop:
http://wiki.davincidsp.com/index.php/Stellaris_One_Day_Workshop

Thank you for attending

Have a safe trip home

2



Addendum

Additional materials of interest to Stellaris users

USB Examples for EKx-LM3S3748 Evaluation Kit

USB Boot Loader Demos 1 and 2.....	3
USB Generic Bulk Device	4
USB HID Keyboard Device.....	4
USB HID Mouse Device	5
USB MSC Device	5
USB Serial Device	5
USB HID Keyboard Host	5
USB HID Mouse Host	6
USB Mass Storage Class Host.....	6

USB Boot Loader Demos 1 and 2

Demonstrate the use of the flash-based USB boot loader

At startup, the application displays a message then branches to the USB boot loader to await the start of an update. The boot loader presents a Device Firmware Upgrade interface to the host allowing new applications to be downloaded to flash via USB.

The **usb_boot_demoX** application can be used along with this application to easily demonstrate that the boot loader is actually updating the on-chip flash.

The application **dfuwrap**, found in the **boards** directory, can be used to prepare binary images for download to a particular position in device flash. This application adds a Luminary-specific prefix and a DFU standard suffix to the binary. A sample Windows command line application, **dfuprog**, is also provided which allows either binary images or DFU-wrapped files to be downloaded to the board or uploaded from it.

The **usb_boot_demo1** and **usb_boot_demo2** applications are essentially identical to **boot_demo1** and **boot_demo2** with the exception that they are linked to run at address 0x1800 rather than 0x0. This is due to the fact that the USB boot loader is not currently included in the Stellaris ROM and therefore has to be stored in the bottom few KB of flash with the main application stored above it.

USB Generic Bulk Device

Generic USB device offering simple bulk data transfer to and from the host

The device uses a vendor-specific class ID and supports a single bulk IN endpoint and a single bulk OUT endpoint. Data received from the host is assumed to be ASCII text and it is echoed back with the case of all alphabetic characters swapped.

A Windows INF file for the device is provided on the installation CD. This INF contains information required to install the WinUSB subsystem on WindowsXP and Vista PCs. WinUSB is a Windows subsystem allowing user mode applications to access the USB device without the need for a vendor-specific kernel mode driver. A sample Windows command-line application, **usb_bulk_example**, illustrating how to connect to and communicate with the bulk device is also provided as part of the Windows examples package on the installation CD or via download from http://www.luminarymicro.com/products/software_updates.html. Project files are included to allow the examples to be built using Microsoft VisualStudio 2005.

USB HID Keyboard Device

Turns the evaluation board into a USB keyboard supporting the Human Interface Device class

The color STN display shows a virtual keyboard which can be navigated using the direction control button on the board. Pressing down on the button presses the highlighted key, sending its usage code and, if necessary, a shift modifier, to the USB host. The board status LED is used to indicate the current Caps Lock state and is updated in response to pressing the ``Caps" key on the virtual keyboard or any other keyboard attached to the same USB host system.

The device implemented by this application also supports USB remote wakeup allowing it to request the host to reactivate a suspended bus. If the bus is suspended (as indicated on the application display), pressing the Select key will request a remote wakeup assuming the host has not specifically disabled such requests.

USB HID Mouse Device

Turns the evaluation board into a USB mouse supporting the Human Interface Device class

Presses on the navigation control on the evaluation board are translated into mouse movement and button press messages in HID reports sent to the USB host allowing the evaluation board to control the mouse pointer on the host system.

USB MSC Device

Turns the evaluation board into a USB mass storage class device

The application will use the **microSD** card for the storage media for the mass storage device. The screen will display the current action occurring on the device ranging from disconnected, no media, reading, writing and idle.

USB Serial Device

Turns the evaluation kit into a virtual serial port when connected to the USB host system

The application supports the USB Communication Device Class, Abstract Control Model to redirect UART0 traffic to and from the USB host system. File `usb_dev_serial_win2k.inf` may be used to install the example as a virtual COM port on a Windows2000 system. For WindowsXP or Vista, **`usb_dev_serial.inf`** should be used.

USB HID Keyboard Host

Demonstrates how to support a USB keyboard attached to the evaluation kit board

The display will show if a keyboard is currently connected and the current state of the Caps Lock key on the keyboard that is connected on the bottom status area of the screen. Pressing any keys on the keyboard will cause them to be printed on the screen and to be sent out the UART at 115200 baud with no parity, 8 bits and 1 stop bit. Any keyboard that supports the USB HID bios protocol should work with this demo application.

USB HID Mouse Host

Demonstrates how to support a USB mouse attached to the evaluation kit board

The display will show if a mouse is currently connected and the current state of the buttons on the bottom status area of the screen. The main drawing area will show a mouse cursor that can be moved around in the main area of the screen. If the left mouse button is held while moving the mouse, the cursor will draw on the screen. A side effect of the application not being able to read the current state of the screen is that the cursor will erase anything it moves over while the left mouse button is not pressed.

USB Mass Storage Class Host

Demonstrates how to connect a USB mass storage class device to the evaluation kit

When a device is detected, the application displays the contents of the file system and allows browsing using the buttons.