StellarisWare Examples

USER'S GUIDE

SW-EXAMPLES-UG-6594

Copyright

Texas Instruments
108 Wild Basin, Suite 350
Austin, TX 78746
Main: +1-512-279-8800
Fax: +1-512-279-8879
http://www.ti.com/stellaris

Revision Information

This is version 6594 of this document, last updated on October 13, 2010.

Table of Contents

1 Introduction

Texas Instruments® StellarisWare® software provides code examples in two different locations. The first type of code example is specific to a particular board and is found in the **boards** directory. The examples in this directory can be recomplied, downloaded and run on the specified board without modification. For more information on these examples, refer to the specific Board Firmware Development Package User's Guide.

The second type of example applies to all Stellaris® microcontrollers with a particular peripheral and can be found in the **examples** directory. These examples are small, single-purpose code segments that are meant to clearly and simply demonstrate a specific feature and must be customized to run on a particular board.

This document describes the examples available in the **examples** directory. Not every example can run on every Stellaris device; consult the device data sheet to determine if a particular feature is present. Furthermore please note: THESE EXAMPLES ARE NOT READY TO RUN PROJECTS. For ready-to-run projects please see the **boards** directory.

2 Peripheral Examples

Examples are organized by peripheral in the following sections. Each peripheral section contains a brief description of each example. They are located in your StellarisWare installation under the **examples** directory, where there is a separate sub-directory for each peripheral.

Note that these examples are different and separate from the board specific examples that you will find under the **boards** directory and which are documented elsewhere.

2.1 ADC Examples

2.1.1 Differential ADC (differential)

This example shows how to setup ADC0 as a differential input and take a single sample between AIN0 and AIN1. The value of the ADC is read and printed to the serial port.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- ADC0 peripheral
- GPIO Port E peripheral (for ADC0 pins)
- AIN0 - PE7
- AIN1 - PE6

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of the ADC.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.1.2 Single Ended ADC (single_ended)

This example shows how to setup ADC0 as a single ended input and take a single sample on AIN0/PE7.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- ADC0 peripheral
- GPIO Port E peripheral (for AIN0 pin)

- AIN0 - PE7

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of the ADC.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.1.3 ADC Temperature Sensor (temperature_sensor)

This example shows how to setup ADC0 to read the internal temperature sensor.

NOTE: The internal temperature sensor is not calibrated. This example just takes the raw temperature sensor sample and converts it using the equation found in the LM3S9B96 datasheet.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- ADC0 peripheral

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of the ADC.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.2 CAN Examples

2.2.1 Multiple CAN RX (multi_rx)

This example shows how to set up the CAN to receive multiple CAN messages using separate message objects for different messages, and using CAN ID filtering to control which messages are received. Three message objects are set up to receive 3 of the 4 CAN message IDs that are used by the multi_tx example. Filtering is used to demonstrate how to receive only specific messages,

and therefore not receiving all 4 messages from the multi_tx example. As messages are received the content of each are printed to the serial console.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- CAN0 peripheral
- GPIO port D peripheral (for CAN0 pins)
- CAN0RX - PD0
- CAN0TX - PD1

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of CAN.

- GPIO port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- INT_CAN0 - CANIntHandler

2.2.2 Multiple CAN TX (multi_tx)

This example shows how to set up the CAN to send multiple messages. The CAN peripheral is configured to send messages with 4 different CAN IDs. Two of the messages (with different CAN IDs) are sent using a shared message object. This shows how to reuse a message object for multiple messages. The other two messages are sent using their own message objects. All four messages are transmitted once per second. The content of each message is a test pattern. A CAN interrupt handler is used to confirm message transmission and count the number of messages that have been sent.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- CAN0 peripheral
- GPIO Port D peripheral (for CAN0 pins)
- CAN0RX - PD0
- CAN0TX - PD1

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of CAN.

- GPIO port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- INT_CAN0 - CANIntHandler

2.2.3 Simple CAN RX (simple_rx)

This example shows the basic setup of CAN in order to receive messages from the CAN bus. The CAN peripheral is configured to receive messages with any CAN ID and then print the message contents to the console.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- CAN0 peripheral
- GPIO port D peripheral (for CAN0 pins)
- CAN0RX - PD0
- CAN0TX - PD1

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of CAN.

- GPIO port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- INT_CAN0 - CANIntHandler

2.2.4 Simple CAN TX (simple_tx)

This example shows the basic setup of CAN in order to transmit messages on the CAN bus. The CAN peripheral is configured to transmit messages with a specific CAN ID. A message is then transmitted once per second, using a simple delay loop for timing. The message that is sent is a 4 byte message that contains an incrementing pattern. A CAN interrupt handler is used to confirm message transmission and count the number of messages that have been sent.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- CAN0 peripheral
- GPIO Port D peripheral (for CAN0 pins)
- CAN0RX - PD0
- CAN0TX - PD1

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of CAN.

- GPIO port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- INT_CAN0 - CANIntHandler

2.3 EPI Examples

2.3.1 EPI SDRAM Mode (sdram)

This example shows how to configure the EPI bus in SDRAM mode. This example has been written to be compatible with the Texas Instruments 8MB SDRAM expansion card for the DK-LM3S9B96.

For the EPI SDRAM mode, the pinout is as follows: Address11:0 - EPI0S11:0 Bank1:0 - EPI0S14:13 Data15:0 - EPI0S15:0 DQML - EPI0S16 DQMH - EPI0S17 /CAS - EPI0S18 /RAS - EPI0S19 /WE - EPI0S28 /CS - EPI0S29 SDCKE - EPI0S30 SDCLK - EPI0S31

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- EPI0 peripheral
- GPIO Port C peripheral (for EPI0 pins)
- GPIO Port E peripheral (for EPI0 pins)
- GPIO Port F peripheral (for EPI0 pins)
- GPIO Port G peripheral (for EPI0 pins)
- GPIO Port H peripheral (for EPI0 pins)
- GPIO Port J peripheral (for EPI0 pins)
- EPI0S0 - PH3
- EPI0S1 - PH2
- EPI0S2 - PC4
- EPI0S3 - PC5
- EPI0S4 - PC6
- EPI0S5 - PC7
- EPI0S6 - PH0
- EPI0S7 - PH1
- EPI0S8 - PE0
- EPI0S9 - PE1
- EPI0S10 - PH4
- EPI0S11 - PH5
- EPI0S12 - PF4
- EPI0S13 - PG0
- EPI0S14 - PG1
- EPI0S15 - PF5
- EPI0S16 - PJ0
- EPI0S17 - PJ1
- EPI0S18 - PJ2
- EPI0S19 - PJ3

- EPI0S28 - PJ4
- EPI0S29 - PJ5
- EPI0S30 - PJ6
- EPI0S31 - PG7

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of EPI0.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.4 I2C Examples

2.4.1 I2C Master Loopback (i2c_master_slave_loopback)

This example shows how to configure the I2C0 module for loopback mode. This includes setting up the master and slave module. Loopback mode internally connects the master and slave data and clock lines together. The address of the slave module is set in order to read data from the master. Then the data is checked to make sure the received data matches the data that was transmitted. This example uses a polling method for sending and receiving data.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- I2C0 peripheral
- GPIO Port B peripheral (for I2C0 pins)
- I2C0SCL - PB2
- I2C0SDA - PB3

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of I2C.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.4.2 Slave Receive Interrupt (slave_receive_int)

This example shows how to configure a receive interrupt on the slave module. This includes setting up the I2C0 module for loopback mode as well as configuring the master and slave modules. Loopback mode internally connects the master and slave data and clock lines together. The address of the slave module is set to a value so it can receive data from the master.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- I2C0 peripheral
- GPIO Port B peripheral (for I2C0 pins)
- I2C0SCL - PB2
- I2C0SDA - PB3

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of I2C.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- INT_I2C0 - I2C0SlaveIntHandler

2.4.3 SoftI2C AT24C08A EEPROM (soft_i2c_atmel)

This example shows how to configure the SoftI2C module to read and write an Atmel AT24C08A EEPROM. A pattern is written into the first 16 bytes of the EEPROM and then read back.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- Timer0 peripheral (for the SoftI2C timer)
- GPIO Port B peripheral (for SoftI2C pins)
- PB2 (for SCL)
- PB3 (for SDA)

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of I2C.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application, you must add these interrupt handlers to your vector table.

- INT_TIMER0A - Timer0AIntHandler

2.5 PWM Examples

2.5.1 PWM dead-band (dead_band)

This example shows how to setup the PWM0 block with a dead-band generation.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- GPIO Port D peripheral (for PWM pins)
- PWM0 - PD0
- PWM1 - PD1

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of the PWM.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.5.2 PWM Invert (invert)

This example shows how to setup PWM0 using the inverted output function. This feature allows you to invert the polarity of the PWM output. This example is setup to invert a 25% duty cycle to get a 75% duty cycle every 5 seconds.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- GPIO Port D peripheral (for PWM0 pin)
- PWM0 - PD0

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of the PWM.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)

- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.5.3 PWM Reload Interrupt (reload_interrupt)

This example shows how to setup an interrupt on PWM0. This example demonstrates how to setup an interrupt on the PWM when the PWM timer is equal to the configurable PWM0LOAD register.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- GPIO Port D peripheral (for PWM0 pin)
- PWM0 - PD0

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of the PWM.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- INT_PWM0 - PWM0IntHandler

2.6 ROM Examples

2.6.1 Direct ROM Function Calls (rom_direct)

This example shows how to directly call a ROM based driver library function using the **ROM_** prefix on the driver library function name. When you call a ROM function in this way, it will only work on a part with ROM, and you will have to change it to work with a non-ROM part.

2.6.2 Mapped ROM Function Calls (rom_mapped)

This example shows how to map ROM function calls at compile time to use a ROM function if available on the part, or a library call if the function is not available in ROM. This allows you to write code that can be used on either a part with ROM or without ROM without needing to change the code. The mapping is performed at compile time and there is no performance penalty for using the mapped method instead of the direct method. Mapped ROM functions are called with a **MAP_** prefix on the driver library function name.

2.7 SSI/SPI Examples

2.7.1 SoftSSI Master (soft_spi_master)

This example shows how to configure the SoftSSI module. The code will send three characters on the master Tx then polls the receive FIFO until 3 characters are received on the master Rx.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- GPIO Port A peripheral (for SoftSSI pins)
- SoftSSICLK - PA2
- SoftSSIFss - PA3
- SoftSSIRx - PA4
- SoftSSITx - PA5

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of SoftSSI.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- SysTickIntHandler

Note:
This example provide the same functionality using the same pins as the spi_master example. As such, it can be used as a guide for how to convert code which uses hardware SSI to the SoftSSI module.

2.7.2 SPI Master (spi_master)

This example shows how to configure the SSI0 as SPI Master. The code will send three characters on the master Tx then polls the receive FIFO until 3 characters are received on the master Rx.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- SSI0 peripheral
- GPIO Port A peripheral (for SSI0 pins)
- SSI0CLK - PA2
- SSI0Fss - PA3
- SSI0Rx - PA4
- SSI0Tx - PA5

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of SSI0.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.7.3 TI Master (ti_master)

This example shows how to configure the SSI0 as TI Master. The code will send three characters on the master Tx then poll the receive FIFO until 3 characters are received on the master Rx.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- SSI0 peripheral
- GPIO Port A peripheral (for SSI0 pins)
- SSI0CLK - PA2
- SSI0Fss - PA3
- SSI0Rx - PA4
- SSI0Tx - PA5

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of I2C0.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.8 System Control Examples

2.8.1 System Clock Configuration with PLL (system_clock_pll)

This example shows how to set up the system clock to use the PLL.

2.9 System Tick Timer (SysTick) Examples

2.9.1 Systick Interrupt (systick_int)

This example shows how to configure the SysTick and the SysTick interrupt.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- NONE

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of Systick.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- SysTickIntHandler

2.10 General Purpose Timer Examples

2.10.1 16-Bit One-Shot Timer (oneshot_16bit)

This example shows how to configure Timer0B as a one-shot timer with a single interrupt triggering after 1ms.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- TIMER0 peripheral

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of Timer0.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- INT_TIMER0B - Timer0BIntHandler

2.10.2 16-Bit Periodic Timer (periodic_16bit)

This example shows how to configure Timer0B as a periodic timer with an interrupt triggering every 1ms. After a certain number of interrupts, the Timer0B interrupt will be disabled.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- TIMER0 peripheral

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of Timer0.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- INT_TIMER0B - Timer0BIntHandler

2.10.3 PWM using Timer (pwm)

This example shows how to configure Timer1B to generate a PWM signal on the timer's CCP pin.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- TIMER1 peripheral
- GPIO Port E peripheral (for CCP3 pin)
- CCP3 - PE4

The following UART signals are configured only for displaying console messages for this example. These are not required for operation of Timer0.

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

This example uses the following interrupt handlers. To use this example in your own application you must add these interrupt handlers to your vector table.

- None.

2.11 UART Examples

2.11.1 UART Polled I/O (uart_polled)

This example shows how to set up the UART and use polled I/O methods for transmitting and receiving UART data. The example receives characters from UART0 and retransmits the same character using UART0. It can be tested by using a serial terminal program on a host computer. This example will echo every character that is type until the return/enter key is pressed.

This example uses the following peripherals and I/O signals. You must review these and change as needed for your own board:

- UART0 peripheral
- GPIO Port A peripheral (for UART0 pins)
- UART0RX - PA0
- UART0TX - PA1

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265