Application Report
SLAA763–August 2017

Interfacing With the MSP430™ JTAG Mailbox (JMB) System

Nima Eskandari *MSP430 Applications*

ABSTRACT

The embedded memory of MSP430™ microcontrollers (MCUs) can be programmed using the on-chip JTAG interface. The MSP430 microcontrollers support a 2-wire JTAG interface. This 2-wire JTAG interface is referred to as Spy-Bi-Wire (SBW). Using SBW, a host can access the nonvolatile FRAM memory on FRAM-based MSP430 MCUs. The System Control module provides the capability to exchange user data through the regular JTAG or SBW test and debug interface. The idea behind the JTAG mailbox (JMB) is to have a direct interface to the CPU during debugging, programming, and test that is identical for all MSP430 devices of this family and that uses only a few or no user application resources. The JTAG interface was chosen because it is available on all MSP430 devices and is a dedicated resource for debugging, programming, and test. The JMB system can be used for run-time data exchange.

This application report uses the SimpleLink™ MSP432P401R MCU to interface with the JTAG mailbox (JMB) system on FRAM-based MSP430 MCUs. The MSP430FR2311 LaunchPad™ development kit is used as the FRAM-based MSP430 MCU. Software examples are provided for the SimpleLink MSP432P401R MCU and the MSP430FR2311 FRAM-based MCU. The software example also makes use of the SimpleLink Software Development Kit (SDK), making it easy to port to other SimpleLink devices.

The source code and other files described in this application report can be downloaded from http://www.ti.com/lit/zip/slaa763. The example source code demonstrates how a SimpleLink MCU can access the JMB system on a target MSP430 MCU (FRAM-based) through the SBW communication.

Contents

List of Figures

List of Tables

Trademarks

1 Introduction

The JTAG mailbox (JMB) system can be used for run-time data exchange between the Spy-Bi-Wire (SBW) host, and the target MSP430 MCU.

Spy-Bi-Wire (SBW) provides a method for communication between a host and the target MSP430 MCUs. For the MSP430 MCU under SBW control, commands must be sent to the dedicated pins of the target MSP430 using the SBW protocol after the SBW entry sequence has been applied to the dedicated pins. A special entry sequence is used in this application report, which holds the Reset pin of the target MSP430 high, in order to not reset the device.

To enable the SBW connection, an entry sequence must be applied to the dedicated SBW pins. After the SBW entry sequence is applied, commands can be sent to the target MSP430.

Software example is provided for FRAM-based MSP430 devices. The software example use MSP432P401R as the SBW host to the target MSP430 MCU. The SBW protocol in these examples is implemented on top of the SimpleLink SDK, which makes it extremely easy to port to other SimpleLink devices. Using the SBW protocol, the host can access the JMB system of the target MSP430 MCU. Data is transferred to and received from the target MSP430 MCU using the JMB system. The system nonmaskable interrupt (NMI) can be used to receive the data in the JMB of the MSP430 MCU.

1.1 Supplementary Online Information

For more information, visit Replicator for MSP430 MCU. This page contains links to additional SBW and JTAG user's guides and source code for using an MSP430F5437 to program other MSP430 devices.

For more information on JTAG and SBW communication, see MSP430™ Programming With the JTAG Interface. This user's guide describes how the JTAG and SBW interface can be used to program MSP430 MCUs. It also describes the commands used by the SBW protocol.

For more information on how to use the JMB system on the target MSP430 MCU, see the MSP430FR4xx and MSP430FR2xx Family User's Guide and the MSP430FR58xx, MSP430FR59xx, and MSP430FR6xx Family User's Guide.

Additional support is provided by the TI E2E™ Community.

2 Software Example

Example software is available for SimpleLink devices to act as the SBW host to target MSP430 MCUs, in order to access the JMB.

The following LaunchPad development kits were used to develop the software examples:

- SimpleLink MSP432P401R LaunchPad Development Kit
- MSP430FR2311 LaunchPad Development Kit

2.1 Software Example File Descriptions

The example software can be downloaded from http://www.ti.com/lit/zip/slaa763.

Table 1 describes the content of the top-level folder.

Table 1. Top-Level Folder Description

Folder Name	Description
MSP430FR2311_JMB/msp430fr2311_JtagMailbox_01.c and MSP430FR2311_JMB/msp430fr2311_JtagMailbox_02.c	JMB examples for MSP430FR2311. The examples provide source code to access the JMB through interrupts and polling. msp430fr2311_JtagMailbox_01 uses the polling method. msp430fr2311_JtagMailbox_02 uses interrupts. Use this example with the MSP432Host_MSP430FR_JTAGMailbox
MSP432Host_MSP430FR_JTAGMailbox	Example SimpleLink SDK project for the MSP432™ host. The MSP432 MCU acts as a SBW host for MSP430 FRAM-based devices. The MSP432 host then communicates with the target MSP430 MCU through JMB. This example must be used with the MSP432P401R LaunchPad development kit. Note: This is a Code Composer Studio™ IDE project and must be imported into the IDE. This project is dependent on tirtos_builds_MSP_EXP432P401R_release_ccs, which also must be imported to the IDE.
tirtos_builds_MSP_EXP432P401R_release_ccs	TI RTOS project to be used with MSP432 example projects. This project is referenced by all MSP432 projects.

Table 2 describes the content of the MSP432Host_MSP430FR_JTAGMailbox project.

Table 2. MSP432 Host Example Project Content

Name	Description
SBW (Directory)	Contains the SBW implementation, JMB communication commands and peripheral drivers.
Board.h	Contains board specific macros.
main_tirtos.c	Entry point for TI RTOS.
MSP_EXP432P401R.h	MSP_EXP432P401R board specific APIs.
MSP_EXP432P401R.c	MSP432 specific for TI RTOS. Responsible for setting up the board specific items for the MSP_EXP432P401R board.
msp432host_sbw_msp430FR_jtagmailbox.c	Driver Initialization. Starts the SBW communication and interfaces with the JMB.

The SBW folder in each project contains the drivers for the peripherals used by the SimpleLink host. This folder also contains the SBW commands required for interfacing with the JMB on target MSP430 MCU.

Table 3 describes the content of the SBW folder.

Table 3. SBW Folder Content

Name	Description
SBW430FR.c	Implementation of the SBW commands. This file contains the commands used to interface with an MSP430 MCU through Spy-Bi-Wire. It also contains the commands for interfacing with the JMB
SBW430FR.h	Contains the function declarations for the SBW commands.
sbw_main.c	Starts the SBW communication and interfaces with the JMB. It does not reset the device and allows it to run while the SBW entry sequence is applied.
sbw_main.h	Contains the function declarations for the sbw_main.c
config.h	Contains configurable options such whether or not to output software progress to the serial terminal.
debug.c	Initializes the backchannel UART for communication with PC (Serial terminal programs such as Putty)
debug.h	Contains the function declarations for the debug.c
gpio_if.c	Device specific interface file to access the GPIOs. The GPIOs are used to control the dedicated SBW entry sequence pins and send SBW commands.
gpio_if.h	Contains the function declarations for the gpio_if.c

Table 3. SBW Folder Content (continued)

Name	Description
uart_if.c	Interfaces with the UART module on the device to send and receive data through UART protocol. (This file is not used in the example project.)
uart_if.h	Contains the function declarations for the uart_if.c
utils.c	Interfaces with the timer modules on the device to generate specified delays. Also contain other utilities for debugging.
utils.h	Contains the function declarations for the utils.c

3 Spy-Bi-Wire (SBW) Connections

The SimpleLink host and target MSP430 MCU must be connected to through the SBW pins and share the GND signal. The SBW pins include the two dedicated pins used for 2-wire JTAG interface.

3.1 Spy-Bi-Wire Connections for the Target MSP430

To access the MSP430 memory through the SBW interface, an entry sequence must be applied to the dedicated pins. In this report, the MSP430FR2311 is used as the target for SBW communication.

The SBW entry sequence must be applied to the TST and RST pins on the target MSP430 device, in order to invoke the SBW communication. For more information on the SBW entry sequence see MSP430™ Programming With the JTAG Interface and the other resources in Section 1.1.

Figure 1 shows the dedicated pins for SBW for the MSP430FR2311 LaunchPad development kit.

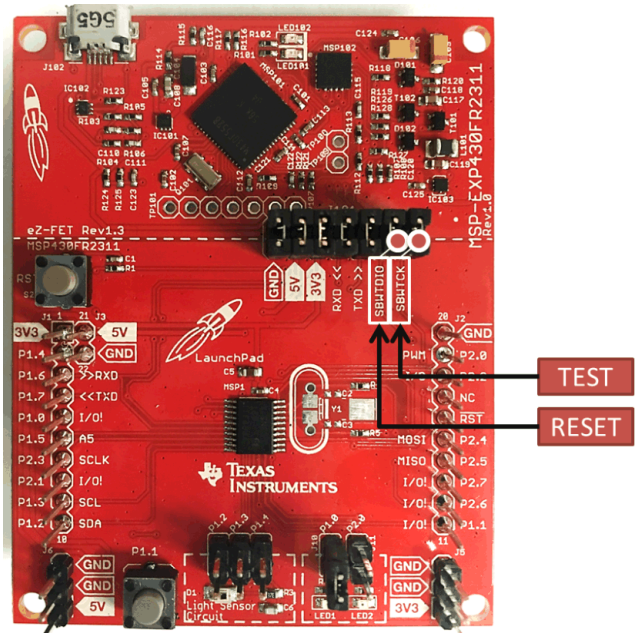

Figure 1. MSP430FR2311 LaunchPad Development Kit SBW Pins

These pins are connected to the SimpleLink host's GPIO. After the SBW entry sequence is completed, the host can use these same pins to send and receive SBW commands.

The pins for SBW interface are Reset and Test pins. Some LaunchPad development kits specify these two pins as SBWTDIO and SBWTCK.

- SBWTDIO: Spy-By-Wire Data Input/Output (RESET PIN)
- SBWTCK: Spy-By-Wire Clock (TEST PIN)

The final connections needed on the target MSP430 is the power connections. All power pins must be connected to the required voltages. In this case, the MSP430FR2311 is connected to a 3.3-V supply through VCC and GND pins.

3.2 Spy-Bi-Wire (SBW) Connections for the Host SimpleLink Device

The host device uses two GPIO pins to communicate through the SBW interface to the target MSP430. The host also uses one UART module. The UART module communicates the status of code execution to the PC through the backchannel serial port. These pins are defined in the board specific file (MSP_EXP432P401R.c) for the project.

The software example provided for the MSP432P401R LaunchPad development kit uses the following pins:

- Target reset pin: P6.0
- Target test pin: P6.1
- Backchannel UART to PC:
  - TX: P1.3
  - RX: P1.2

Figure 2 shows the pins used for the software example.

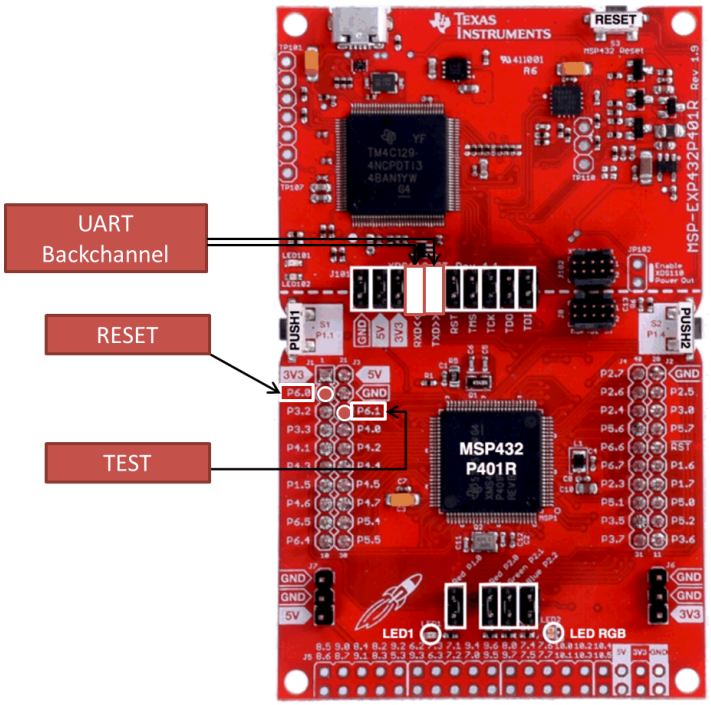

Figure 2. MSP432P401R LaunchPad Development Kit SBW Host Pins

The SimpleLink Host connects to the MSP430 MCU through the Reset (SBW data input/output pin) and Test (SBW clock pin) pins. The SimpleLink host and MSP430 MCU must share the ground signal. Also, the MSP430 MCU must be powered up. Finally, through the backchannel UART, a PC can be used to view the status of the firmware update on the MSP430 MCU.

4 How to Use the Software Examples

The software example in this application report uses SimpleLink MSP432 SDK v1.40.01.00. Code Composer Studio (CCS) Version 7.1.0.00016 was used to compile and debug the projects. Download and extract the zip file containing the software examples.

4.1 Import the Projects to CCS

To import the projects:

1. Click Project > Import CCS Projects.
2. Select Browse and navigate to the extracted file from the software example zip file (see Figure 3).

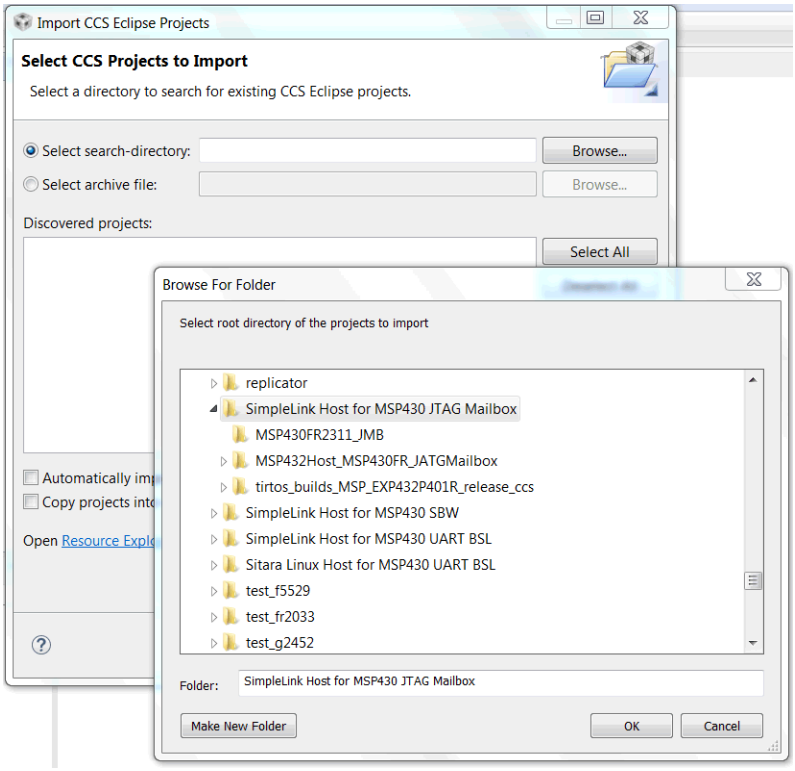

Figure 3. Import Software Example Projects to CCS

3. CCS automatically finds all of the projects inside the folder. Select all projects including the TI RTOS build projects and import them to your workspace (see Figure 4).

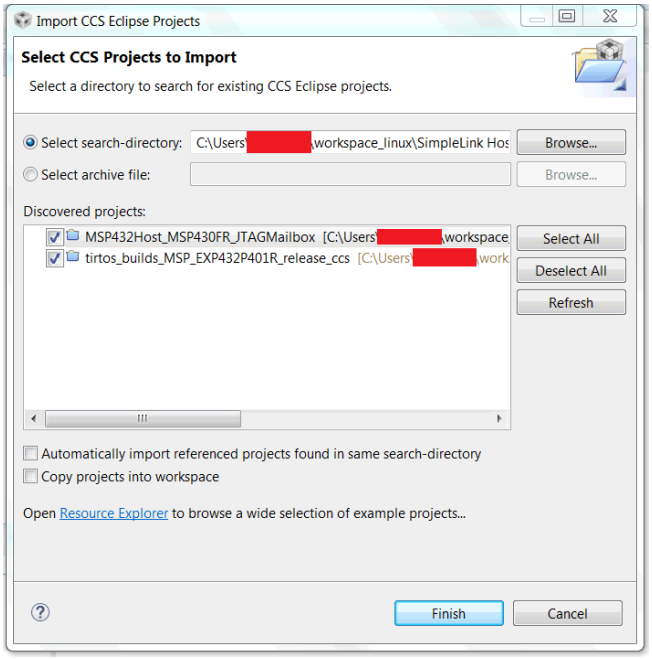

Figure 4. Import all Projects to CCS

4. After importing, the projects must be built. Select all of the imported projects, except for the TI RTOS build projects. Right click and select Rebuild Project (see Figure 5).

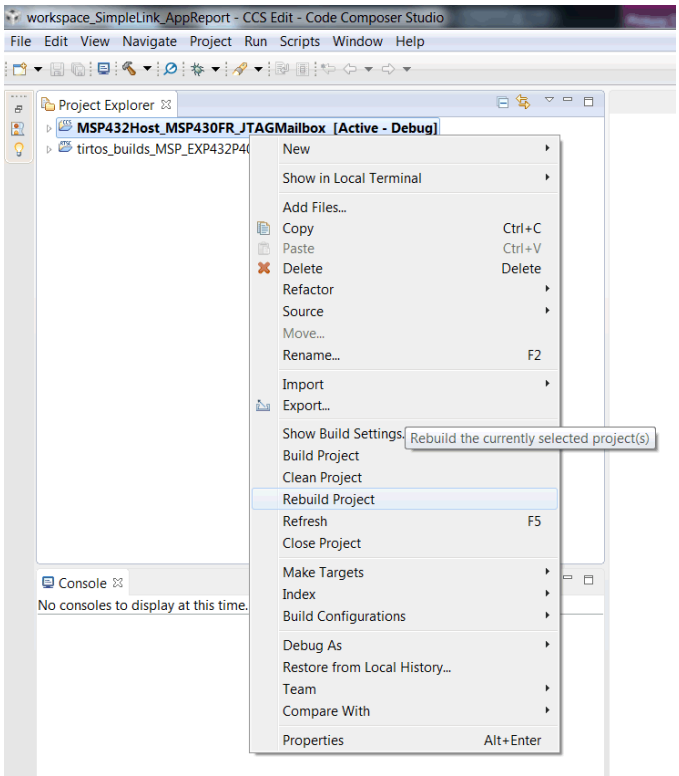

Figure 5. Building Software Example Projects

5. The tirtos_builds_MSP_EXP432P401R_release_ccs is rebuilt automatically. This project is referenced by the MSP432Host_MSP430FR_JTAGMailbox project (see Figure 6).

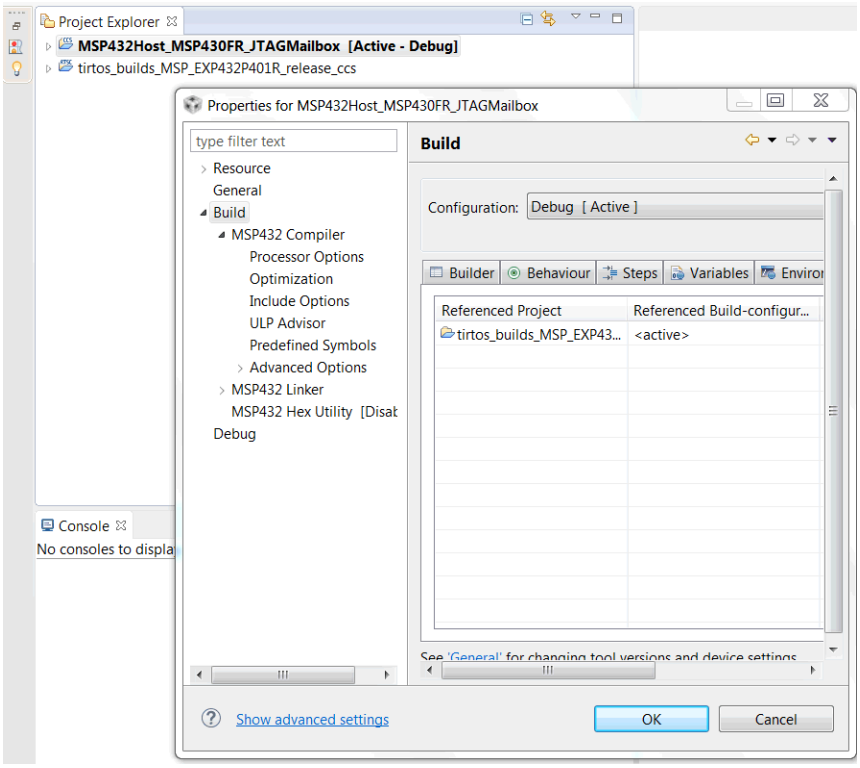

Figure 6. TI RTOS Build Project Dependency

4.2 Run the Software Examples

Connect the SimpleLink host to the MSP430 target. For this demo, the MSP432P401R LaunchPad development kit and MSP430FR2311 LaunchPad development kit are used.

4.2.1 MSP430FR2311 JTAG Mailbox Access Through SBW Example

The first step for showcasing the JMB is to program the MSP430FR2311 with the example code provided in the MSP430FR2311_JMB folder. Both examples provided in this folder access the JMB by reading available data and sending new data when the JMB is empty. The first example uses the polling method, while the second example uses interrupts.

4.2.1.1 MSP430FR2311 JMB Example

For this demo, use the interrupt based example (msp430fr2311_JtagMailbox_02.c).

```
/* --COPYRIGHT--,BSD_EX
 * Copyright (c) 2012, Texas Instruments Incorporated
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * *  Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
```

```
 * *  Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 *
 * *  Neither the name of Texas Instruments Incorporated nor the names of
 *    its contributors may be used to endorse or promote products derived
 *    from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
 * THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
 * PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
 * CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
 * EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
 * PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
 * OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
 * WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
 * OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
 * EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 ******************************************************************************
 *
 *                       MSP430 CODE EXAMPLE DISCLAIMER
 *
 * MSP430 code examples are self-contained low-level programs that typically
 * demonstrate a single peripheral function or device feature in a highly
 * concise manner. For this the code may rely on the device's power-on default
 * register values and settings such as the clock configuration and care must
 * be taken when combining code from several examples to avoid potential side
 * effects. Also see www.ti.com/grace for a GUI- and www.ti.com/msp430ware
 * for an API functional library-approach to peripheral configuration.
 *
 * --/COPYRIGHT--*/
//******************************************************************************
//  MSP430FR2311 Demo - JTAG mailbox receive and send data through interrupt.
//  Nima Eskandari
//  Texas Instruments Inc.
//  June 27, 2017
//  Built with Code Composer Studio v7.2
//******************************************************************************

#include <msp430.h>
#include <stdint.h>

uint32_t SendToJMB = 0;
uint32_t ReceivedFromJMB = 200000;
/*
 * main.c
 */
int main(void) {

    uint32_t loopIndex = 0;
    WDTCTL = WDTPW | WDTHOLD;   // Stop watchdog timer
    P1OUT &= ~BIT0;                         // Clear P1.0 output latch for a defined
power-on state
    P1DIR |= BIT0;                          // Set P1.0 to output direction

    PM5CTL0 &= ~LOCKLPM5;                   // Disable the GPIO power-
on default high-impedance mode
                                            // to activate previously configured
```

```
port settings

    SYSJMBC = JMBMODE;
    SFRIE1 |= JMBINIE;


    while(1)
    {
        P1OUT ^= BIT0;                          // Toggle P1.0 using exclusive-OR
        loopIndex = ReceivedFromJMB;
        while(loopIndex --)
        {
            __delay_cycles(10);
        }

        SendToJMB++;
        if (JMBOUT0FG & SYSJMBC)
        {
            SYSJMBO0 = (uint16_t)(SendToJMB & 0x0000FFFF);
            SYSJMBO1 = (uint16_t)((SendToJMB>>16) & 0x0000FFFF);
        }

    }
}

#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=SYSNMI_VECTOR
__interrupt void SYSMI_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(UNMI_VECTOR))) UNMI_ISR (void)
#else
#error Compiler not supported!
#endif
{
   switch(__even_in_range(SYSSNIV, SYSSNIV__CBDIFG))
   {
        case SYSSNIV__NONE: break;
        case SYSSNIV__SVSLIFG: break;
        case SYSSNIV__UBDIFG: break;
        case SYSSNIV__ACCTEIFG: break;
        case SYSSNIV__VMAIFG: break;
        case SYSSNIV__JMBINIFG:
            ReceivedFromJMB = SYSJMBI1;
            ReceivedFromJMB = ReceivedFromJMB << 16;
            ReceivedFromJMB |= SYSJMBI0;
            if (ReceivedFromJMB > 200000)
            {
                ReceivedFromJMB = 200000;
            }
            break;
        case SYSSNIV__JMBOUTIFG:
            break;
        case SYSSNIV__CBDIFG: break;
        default: break;
   }

}
```

The code performs the following operations:

- Disable the watchdog timer.
- Set P1.0 as output and set it high.
- Enable the JMB interrupt, and set the mode of the JMB to 32-bit mode.
- Start the main loop in the code.
- This loop toggles the P1.0 pin. The toggle rate is based on the value of ReceivedFromJMB variable.
- The variable SendToJMB is incremented every time P1.0 is toggled. If the JMB outgoing mailbox has been previously read, the new data is written to the JMB outgoing mailbox.
- While the main loop is running, if a new value is available in the incoming mailbox, the interrupt service routine updates the value of the ReceivedFromJMB variable.

After the MSP430FR2311 is programmed with code above, the MSP432P401R must be programmed using the MSP432Host_MSP430FR_JTAGMailbox project.

4.2.1.2 MSP432P401R Host For JMB Example

Connect the two devices. The pin-to-pin connection is also available in the README file.

Figure 7, Figure 8, and Figure 9 show the connections for the MSP432P401R LaunchPad development kit and MSP430FR2311 LaunchPad development kit.

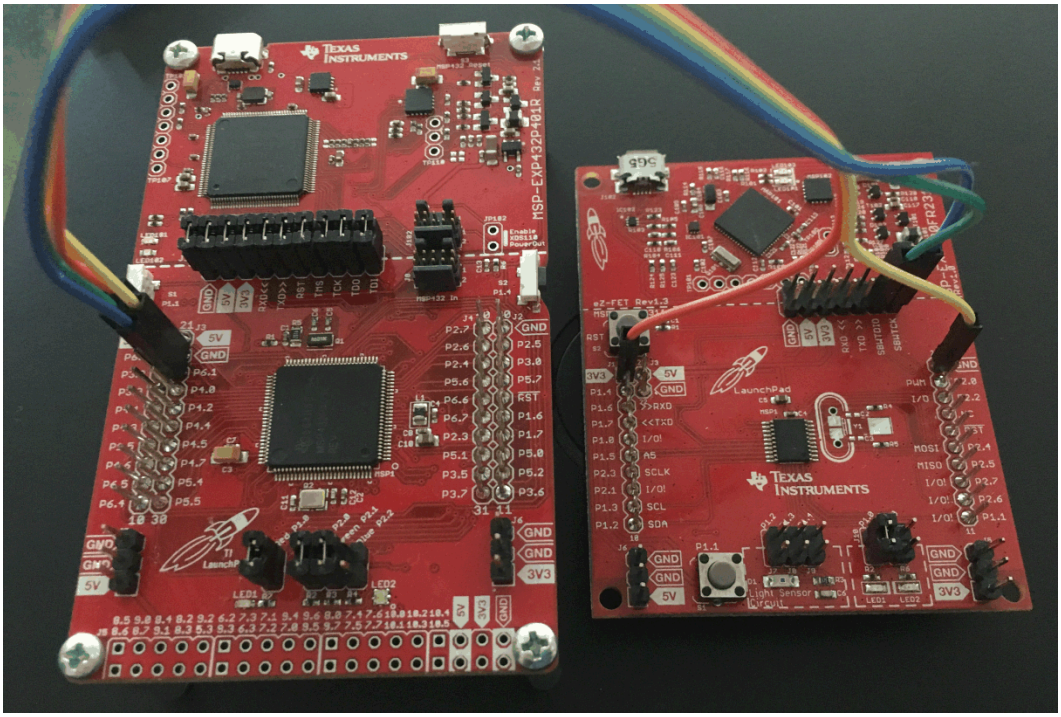

Figure 7. MSP432P401R LaunchPad Development Kit Connected to MSP430FR2311

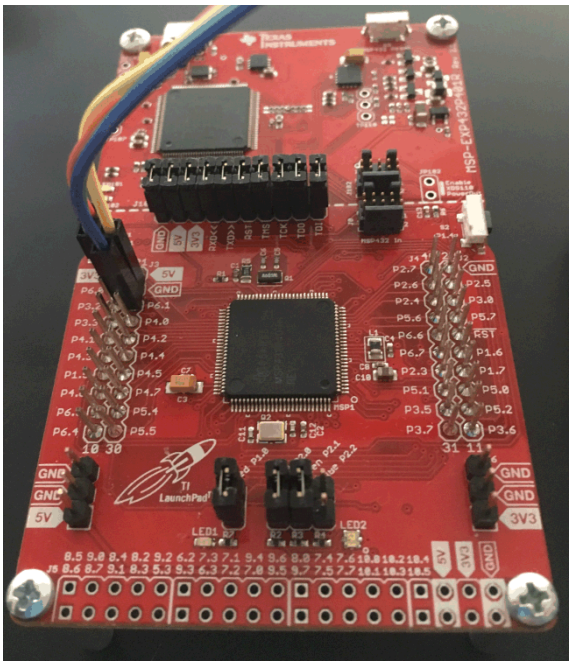

Figure 8. MSP432P401R LaunchPad Development Kit SBW Host Connections

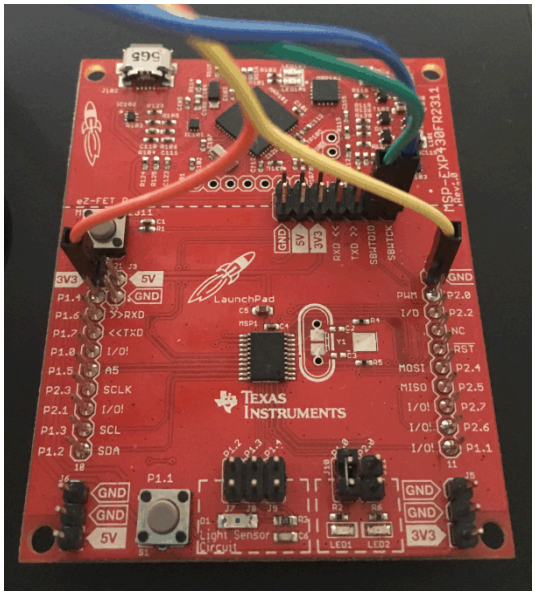

Figure 9. MSP430FR2311 LaunchPad Development Kit SBW Connections

In this setup, the MSP430FR2311 is powered from the MSP432P401R LaunchPad development kit. The MSP432P401R LaunchPad development kit is powered through the USB connection to PC.

After connecting the MSP432 LaunchPad development kit to the PC, the COM port number for serial communication to the board can be found in Windows Device Manager.

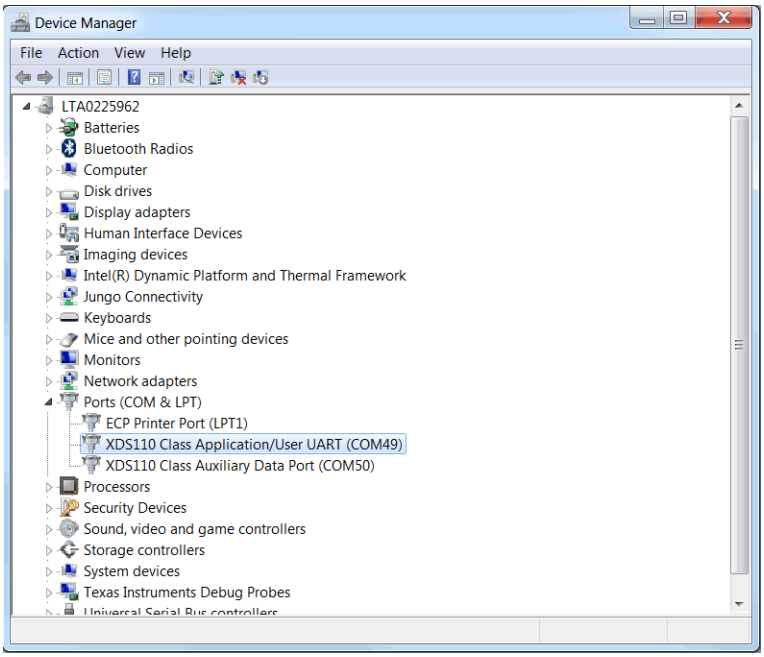

Figure 10. Application UART COM Port Number in Windows Device Manager

A serial communication terminal program such as PuTTY can be used to view the execution of the example software.

Using PuTTY, open a serial communication with the following specification to the COM port found in the Windows Device Manager.

- Baud-rate: 9600
- Data bits: 8
- Stop bits: 1
- Parity: None

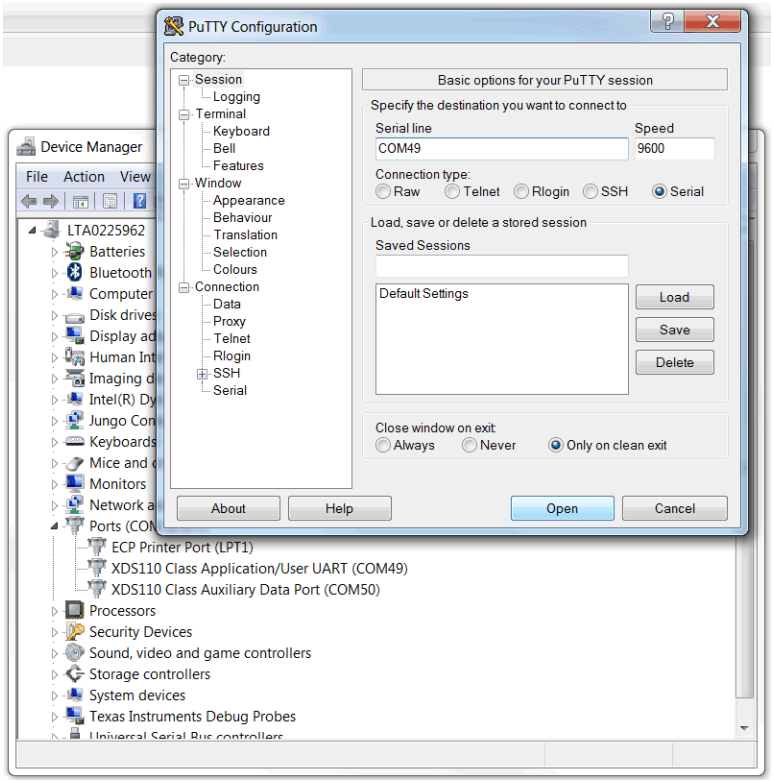

Figure 11. PuTTY Serial Communication Settings

Next, debug the project MSP432Host_MSP430FR_JTAGMailbox by Run > Debug.

After the binary firmware image is uploaded to MSP432P401R, run the example by clicking the green play button.

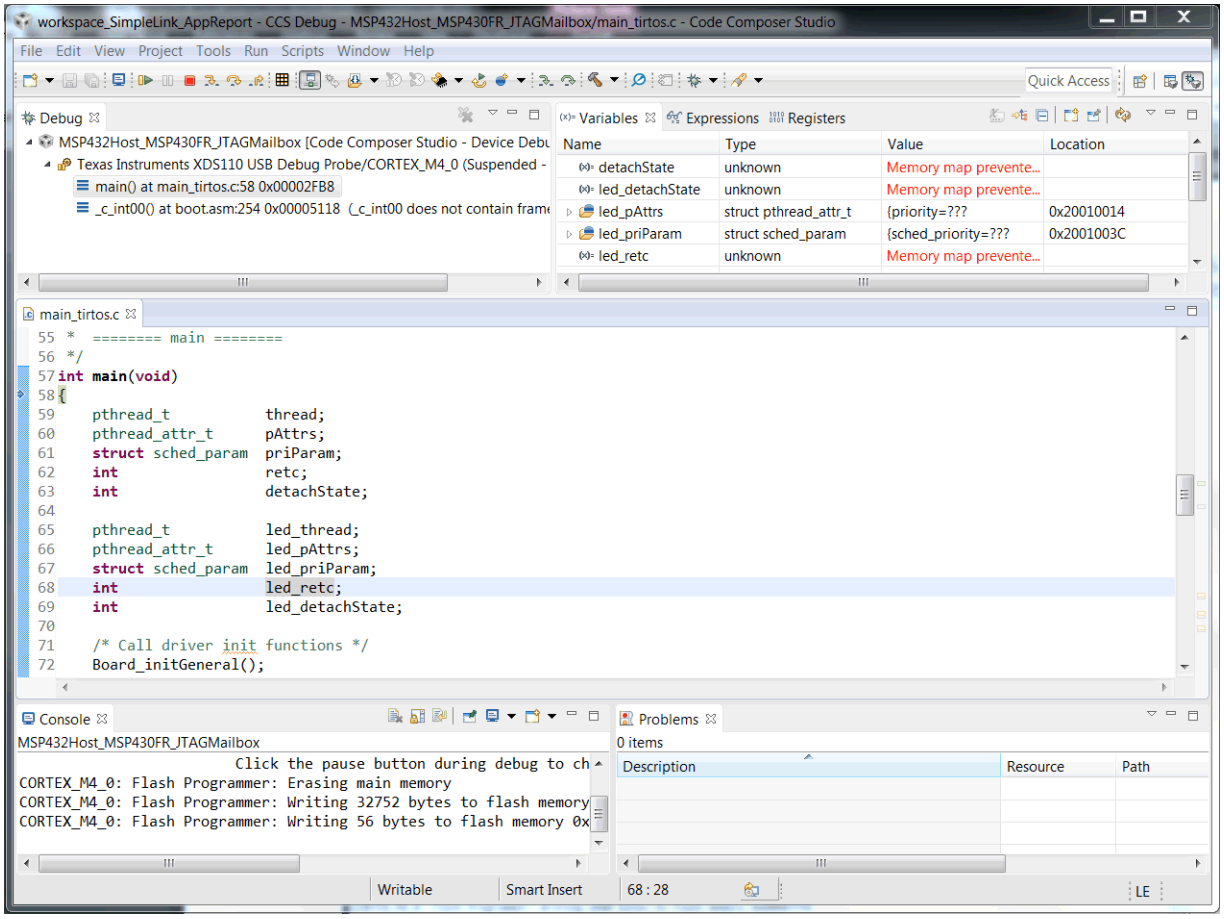

Figure 12. MSP432Host_MSP430FR_JTAGMailbox Project Debug Mode

The next step is to view the serial output on PuTTY. At this point, the LED on P1.0 should be blinking. The example communicates to the MSP430FR2311 through the JMB to change the toggle rate of this LED. Also to showcase the outgoing mailbox, the MSP430FR2311 sends the number of times the LED has toggled. The serial terminal presents a menu to the user.

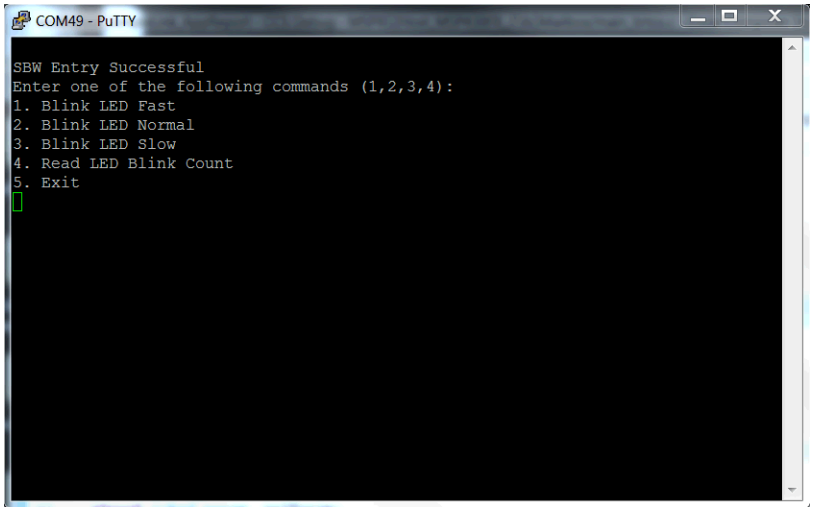

Figure 13. MSP432Host_MSP430FR_JTAGMailbox Project Serial Output

The following options are available to the user:

1. Send a value to the incoming mailbox of the target MSP430FR2311. This value is small and causes a faster toggle rate for P1.0 of the MSP430FR2311.
2. Same as the previous command with a larger value.
3. Same as the previous command with the largest value. This causes the slowest toggle rate.
4. Display the value in the outgoing mailbox of the target MSP430FR2311. This value is the number of times the LED had been toggled since the last time the mailbox was emptied.
5. Exit SBW and end the communication.

5 Error Messages

If the target-to-host connection setup is incorrect, Figure 14 shows the error that can be seen in the serial terminal.

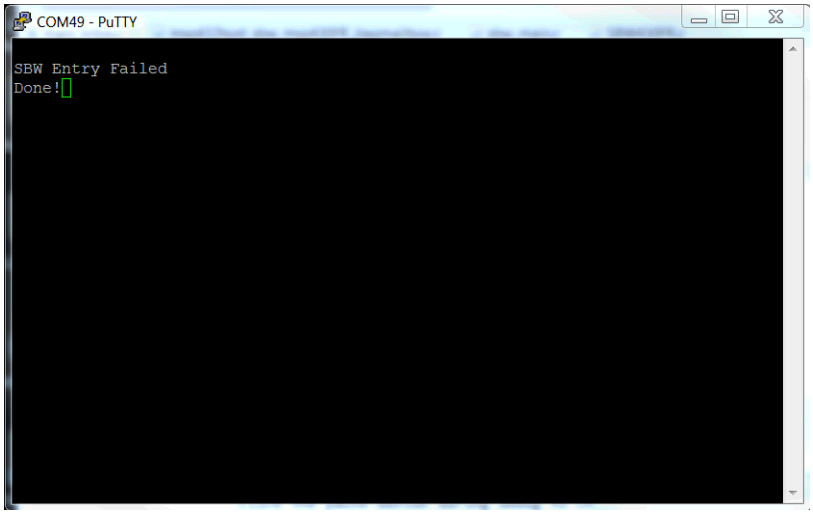

Figure 14. SBW Error

To correct this error, double check the connections between the host and the target. Then power cycle the target MSP430 MCU and run the host again.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

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