User's Guide
SLAU597–March 2015

# MSP-EXP432P401R LaunchPad™ Evaluation Kit

The MSP-EXP432P401R LaunchPad™ is an easy-to-use Evaluation Module (EVM) for the MSP432P401R microcontroller. It contains everything needed to start developing on the MSP432 Low-Power + Performance ARM® 32-bit Cortex®-M4F microcontroller (MCU), including on-board emulation for programming, debugging, and energy measurements. The MSP432P401R device supports low-power applications requiring increased CPU speed, memory, analog, and 32-bit performance.

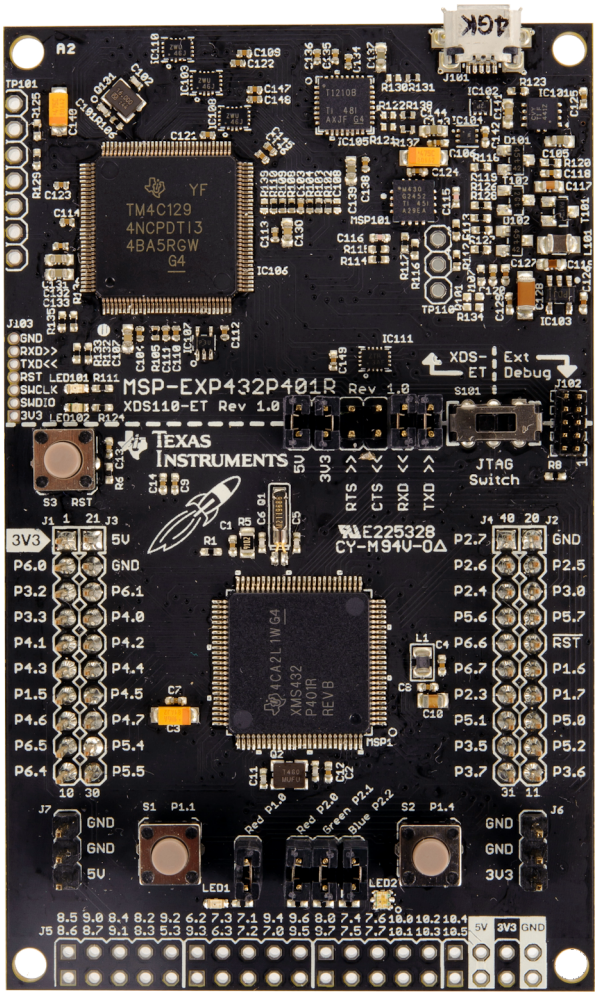


**Figure 1. MSP-EXP432P401R LaunchPad**

## Contents



## List of Figures



## List of Tables

# 1 Getting Started

## 1.1 Introduction

The MSP-EXP432P401R LaunchPad is an easy-to-use evaluation module (EVM) for the MSP432P401R microcontroller. It contains everything needed to start developing on the MSP432 Low-Power + Performance ARM 32-bit Cortex-M4F microcontroller (MCU), including on-board emulation for programming, debugging, and energy measurements. The MSP432P401R device supports low-power applications requiring increased CPU speed, memory, analog, and 32-bit performance.

Rapid prototyping is simplified by access to the 40-pin headers and a wide variety of BoosterPack™ plug-in modules that enable technologies such as wireless connectivity, graphical displays, environmental sensing, and many more. Free software development tools are also available such as TI's Eclipse-based Code Composer Studio™ (CCS) IDE, IAR Embedded Workbench™ IDE, and Keil µVision IDE. Code Composer (CCS) supports EnergyTrace™ technology when paired with the MSP432P401R LaunchPad. More information about the LaunchPad, the supported BoosterPacks, and the available resources can be found at TI's LaunchPad portal. To get started quickly, and find available resources in MSPWare, visit the TI Cloud Development Environment.

## 1.2 Key Features

- Low-power ARM Cortex-M4F MSP432P401R
- 40-pin LaunchPad standard that leverages the BoosterPack ecosystem
- XDS110-ET, an open-source onboard debugger featuring EnergyTrace+ technology and application UART
- Two buttons and two LEDs for user interaction
- Backchannel UART through USB to PC

## 1.3 What's Included

### 1.3.1 Kit Contents

- One MSP-EXP432P401R LaunchPad Evaluation Kit
- One Micro USB cable
- One Quick Start Guide

### 1.3.2 Software Examples (Section 3)

- Out-of-Box Software Example
- CC3100BOOST MQTT-Twitter LED Control Example
- BOOSTXL-K350QVG-S1 Graphics Library Example
- 430BOOST-SHARP96 Graphics Library Example

## 1.4 First Steps: Out-of-Box Experience

An easy way to get familiar with the EVM is by using its preprogrammed out-of-box code. It demonstrates some key features of the LaunchPad from a user level, showing how to use the pushbutton switches together with onboard LEDs and basic serial communication with a computer.

A more detailed explanation of the out-of-box demo can be found in Section 3.

### 1.5 Next Steps: Looking Into the Provided Code

It is now time to start exploring more features of the EVM!

www.ti.com/beginMSP432launchpad

To get started, you will need an integrated development environment (IDE) to explore and start editing the code examples. Refer to Section 4 for more information on IDEs and where to download them.

The out-of-box source code and more code examples are provided for download at http://www.ti.com/tool/msp-exp432p401r. Find what code examples are available and more details about each example in Section 3. All code is licensed under BSD, and TI encourages reuse and modifications to fit specific needs.

# 2 Hardware

Figure 2 shows an overview of the EVM hardware.

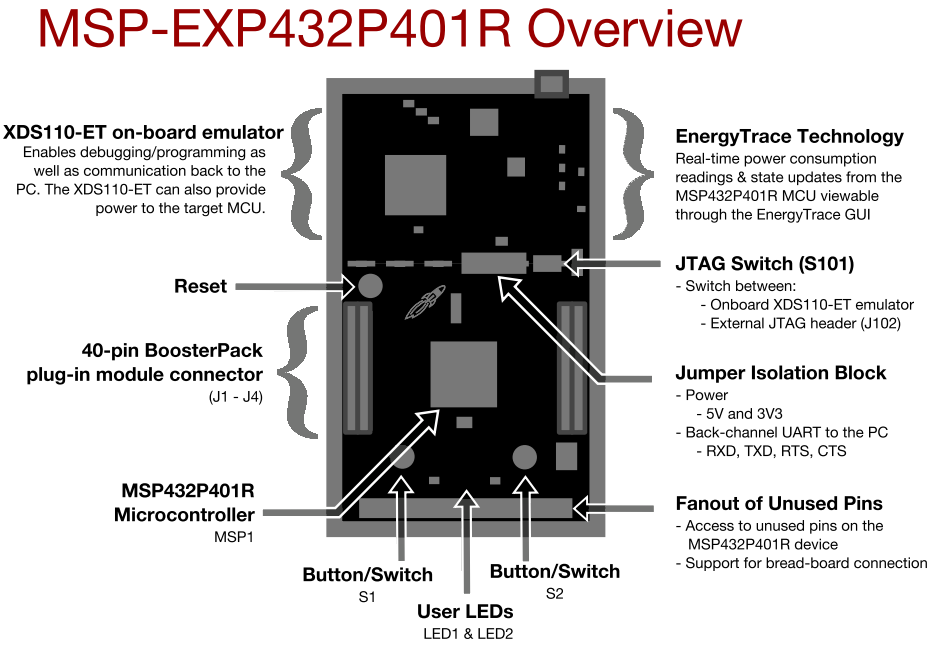


**Figure 2. EVM Overview**

## 2.1 Block Diagram

Figure 3 shows the block diagram.

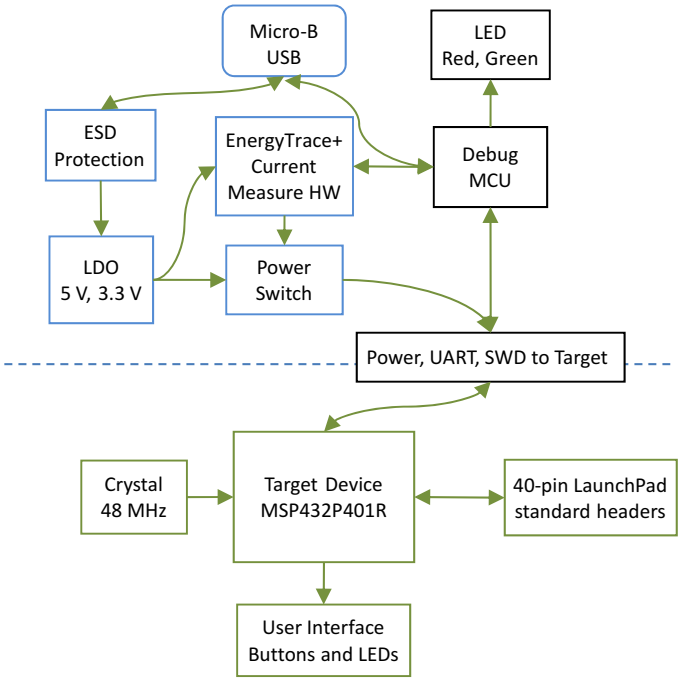


**Figure 3. Block Diagram**

## 2.2 MSP432P401R

The MSP432P401R is the first MSP432 family device featuring low-power performance with an ARM Cortex-M4F core. Device features include:

- Low-power ARM Cortex-M4F MSP432P401R
- Up to 48-MHz system clock
- 256KB flash memory, 64KB SRAM, and 32KB ROM with MSPWare libraries
- Four 16-bit timers with capture/compare/PWM, two 32-bit timers, and RTC
- Up to eight serial communication channels ($I^2C$, SPI, UART, and IrDA)
- Analog: 14-bit SAR ADC, capacitive touch, comparator
- Digital: AES256, CRC, uDMA

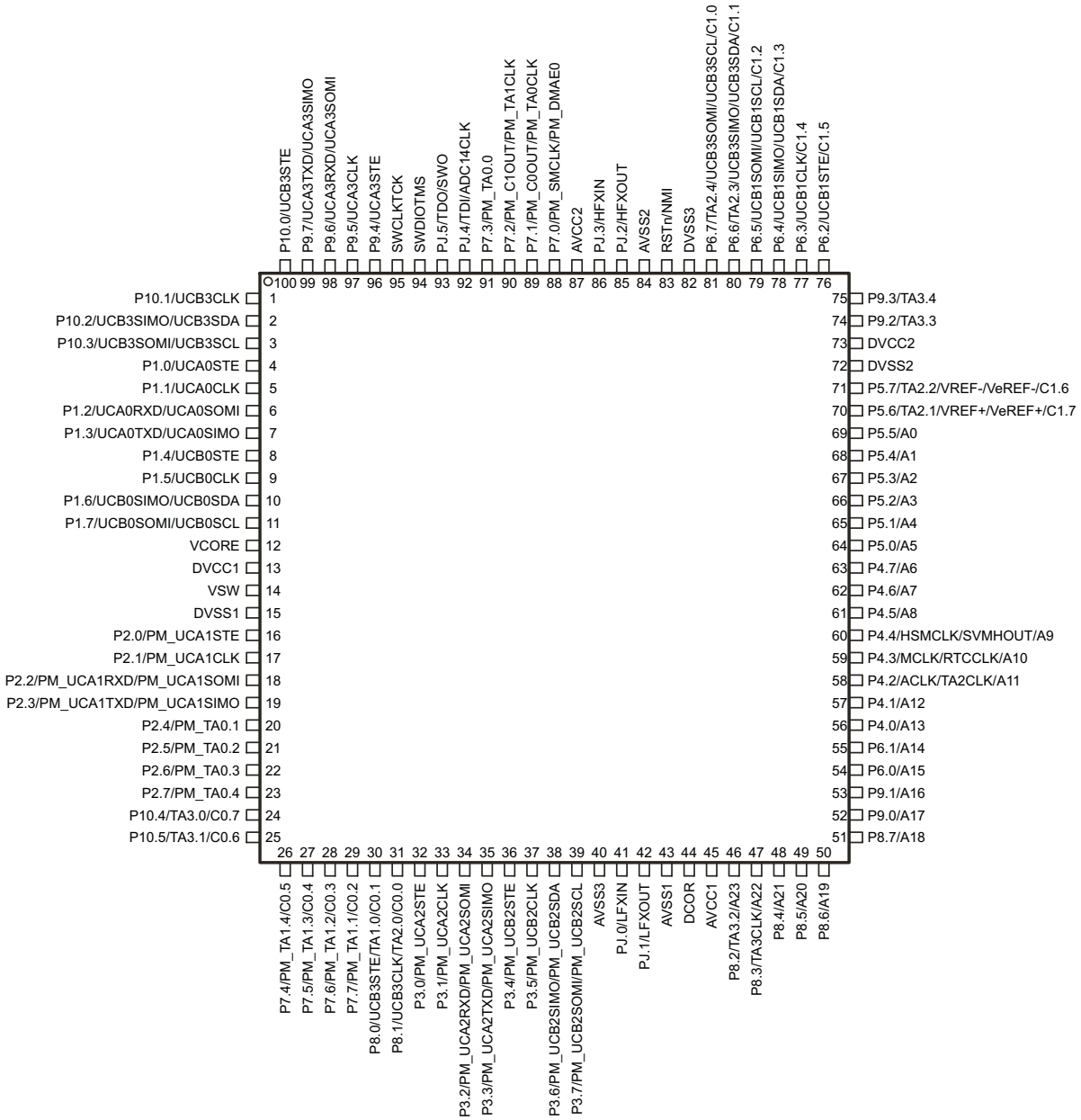


**Figure 4. MSP432P401RIPZ Pinout**

## 2.3 XDS110-ET Onboard Emulator

To keep development easy and cost effective, TI's LaunchPad evaluation kits integrate an onboard emulator, which eliminates the need for expensive programmers. The MSP-EXP432P401R has the XDS110-ET emulator, which is a simple low-cost debugger that supports nearly all TI ARM device derivatives.

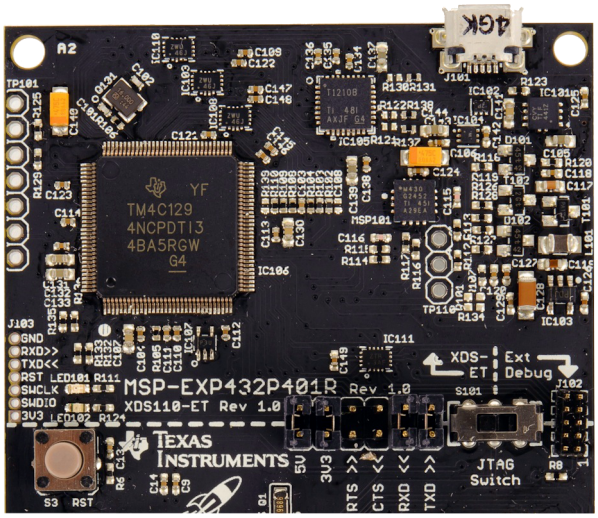


**Figure 5. XDS110-ET Emulator**

The XDS110-ET hardware can be found in the schematics in Section 6 and in the MSP-EXP432P401R Hardware Design Files zip folder.

### 2.3.1 XDS110-ET Isolation Block

The isolation block is comprised of switch S101 and the accompanying jumpers next the switch.

The isolation block allows the user to connect or disconnect signals that cross from the XDS110-ET domain into the MSP432P401R target domain. This crossing is shown by the dotted line across the LaunchPad. No other signals cross this domain, so the XDS110-ET can be decoupled from the MSP432P401R target side. This includes XDS110-ET Serial Wire Debug signals, application UART signals, and 3.3-V and 5-V power.

Table 1 lists the signals that are controlled at the isolation block.

**Table 1. Isolation Block Connections**

| Signal | Isolation Type[(1)(2)] | Description |
|---|---|---|
| 5V | Jumper | 5-V power rail, VBUS from USB |
| 3V3 | Jumper | 3.3-V power rail, derived from VBUS by an LDO in the XDS110-ET domain |
| RTS >> | Jumper* | Backchannel UART: Ready-To-Send, for hardware flow control. The target can use this to indicate whether it is ready to receive data from the host PC. The arrows indicate the direction of the signal. |
| CTS << | Jumper* | Backchannel UART: Clear-To-Send, for hardware flow control. The host PC (through the emulator) uses this to indicate whether it is ready to receive data. The arrows indicate the direction of the signal. |
| RXD << | Jumper | Backchannel UART: The target MCU receives data through this signal. The arrows indicate the direction of the signal. |
| TXD >> | Jumper | Backchannel UART: The target MCU sends data through this signal. The arrows indicate the direction of the signal. |
| RST | Switch S101 | MCU RST signal (active low) |
| TCK_SWCLK | Switch S101 | Serial wire clock input (SWCLK) / JTAG clock input (TCK) |

(1) Jumper* corresponds to signals without a jumper placed by default.
(2) Switch* corresponds to signals that are controlled by S101, but not through IC111.

**Table 1. Isolation Block Connections (continued)**

| Signal | Isolation Type[(1)(2)] | Description |
|---|---|---|
| TMS_SWDIO | Switch S101 | Serial wire data input/output (SWDIO) / JTAG test mode select (TMS) |
| TDO_SWO | Switch S101 | Serial wire trace output (SWO) / JTAG trace output (TWO) (Also PJ.5) |
| TDI | Switch* S101 | JTAG test data input (Also PJ.4) |

Reasons to open these connections:

- To remove any and all influence from the XDS110-ET emulator for high accuracy target power measurements
- To control 3-V and 5-V power flow between the XDS110-ET and target domains
- To expose the target MCU pins for other use than onboard debugging and application UART communication
- To expose the UART interface of the XDS110-ET so that it can be used for devices other than the onboard MCU.

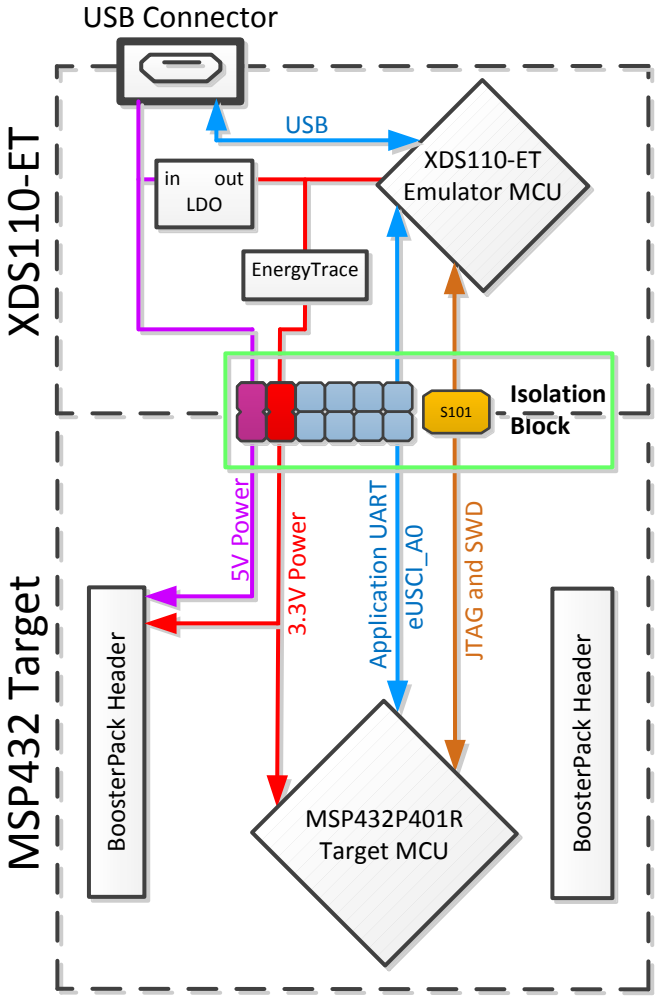


**Figure 6. XDS110-ET Isolation Block**

### 2.3.2 Application (or "Backchannel") UART

The XDS110-ET provides a "backchannel" UART-over-USB connection with the host, which can be very useful during debugging and for easy communication with a PC. The provided UART supports hardware flow control (RTS and CTS); although by default these signals are not connected to the target.

The backchannel UART allows communication with the USB host that is not part of the target application's main functionality. This is very useful during development, and also provides a communication channel to the PC host side. This can be used to create GUIs and other programs on the PC that communicate with the LaunchPad.

The pathway of the backchannel UART is shown in Figure 7. The backchannel UART eUSCI_A0 is independent of the UART on the 40-pin BoosterPack connector eUSCI_A2.

On the host side, a virtual COM port for the application backchannel UART is generated when the LaunchPad enumerates on the host. You can use any PC application that interfaces with COM ports, including terminal applications like Hyperterminal or Docklight, to open this port and communicate with the target application. You need to identify the COM port for the backchannel. On Windows PCs, Device Manager can assist.

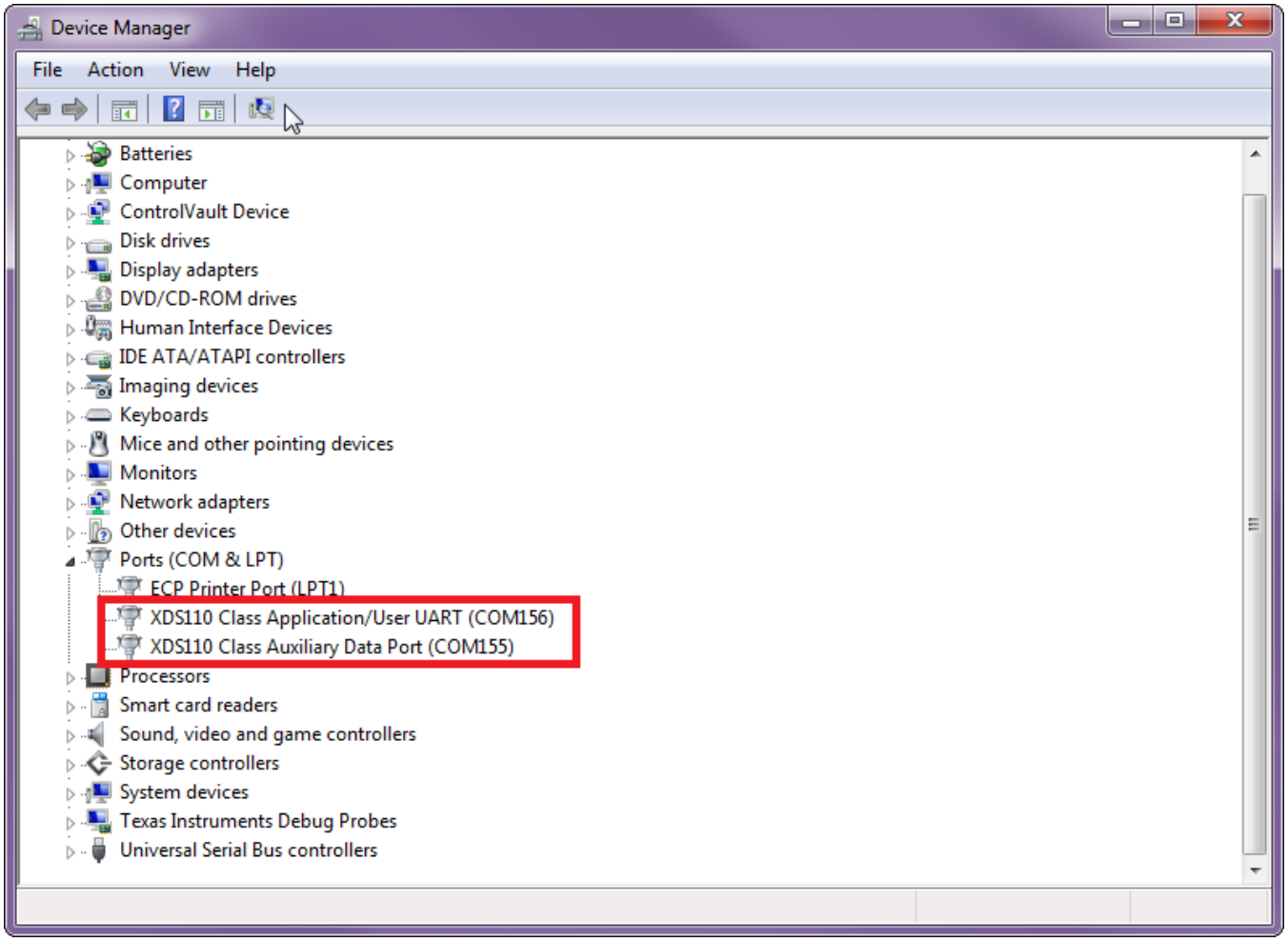


**Figure 7. Application Backchannel UART in Device Manager**

The backchannel UART is the XDS110 Class Application/User UART port. In this case, Figure 7 shows COM156, but this port can vary from one host PC to the next. After you identify the correct COM port, configure it in your host application according to its documentation. You can then open the port and begin communication to it from the host.

The XDS110-ET has a configurable baud rate; therefore, it is important that the PC application configures the baud rate to be the same as what is configured on the eUSCI_A0 backchannel UART.

The XDS110-ET also supports hardware flow control, if desired. Hardware flow control (CTS and RTS handshaking) allows the target MSP432P401R and the emulator to tell each other to wait before sending more data. At low baud rates and with simple target applications, flow control may not be necessary. Applications with higher baud rates and more interrupts to service have a higher likelihood that the will not be able to read the eUSCI_A0 buffer in time, before the next byte arrives. If this happens, the eUSCI_A0 will report an overrun error.

### 2.3.3 Using an External Debugger Instead of the Onboard XDS110-ET

Many users have a specific debugger that they prefer to use, and may want to bypass the XDS110-ET to program the target MCU. This is enabled by switch S101 and connector J102. Using an external debugger is simple, and full JTAG access is provided through J102.

1. Switch S101 to the external debug position (to the right)
2. Plug any ARM debugger into J102
   (a) J102 follows the ARM Cortex Debug Connector standard outlined here
   (b) Note that J102 is not keyed, ensure proper orientation of the debug cable, pin 1 of J102 is on the bottom right side
3. Plug USB power into the LaunchPad, or power it externally
   (a) Ensure that the jumper across 3V3 is connected if using USB power
   (b) External debuggers do not provide power, the Vcc pin is a power sense pin
   (c) More details on powering the LaunchPad can be found in Section 2.4

### 2.3.4 Using the XDS110-ET Emulator With a Different Target

The XDS110-ET emulator on the LaunchPad can interface to most ARM derivative devices, not just the on-board MSP432P401R target device.

This is not a common use case, but for users who want this functionality, there is a way to enable it. Connector J103 was added to expose all the necessary programming and power signals. J103 is a 50 mil spaced 7-pin header. By default it is not populated, so the user will have to populate a connector or directly solder in wires.

When using the XDS110-ET with a different target, the jumpers in the isolation block should be removed, and switch S101 moved to the external debug position. This will disconnect the XDS110-ET from the MSP432P401R target and enable debug of an external device. Because only the SWD signals are exposed, the user needs to set the debugger settings to SWD (without SWO) in the IDE. See the IDE specific MSP432 user's guides for more details on this setting.

To debug other external devices, there are many options in the ARM debugging ecosystem including the XDS100v2/3 and XDS200 from Texas Instruments. There are many other options including IAR I-jet, Keil ULINK, and Segger J-Link.

### 2.3.5 EnergyTrace+ Technology

EnergyTrace™ technology is an energy-based code analysis tool that measures and displays the application's energy profile and helps to optimize it for ultra-low power consumption.

MSP432 devices with built-in EnergyTrace+[CPU State] (or in short EnergyTrace+) technology allow real-time monitoring of internal device states while user program code executes.

EnergyTrace+ technology is supported on the LaunchPad MSP432P401R device + XDS110-ET debugger.

EnergyTrace+ technology is supported on the LaunchPad MSP432P401R device + XDS110-ET debugger. EnergyTrace technology is available as part of Texas Instrument's Code Composer Studio IDE. During application debug, additional windows are available for EnergyTrace technology. To enable EnergyTrace technology, go to:

- Window > Preferences > Code Composer Studio > Advanced Tools > EnergyTrace™ Technology
- Check the Enable Auto-Launch on target connect box

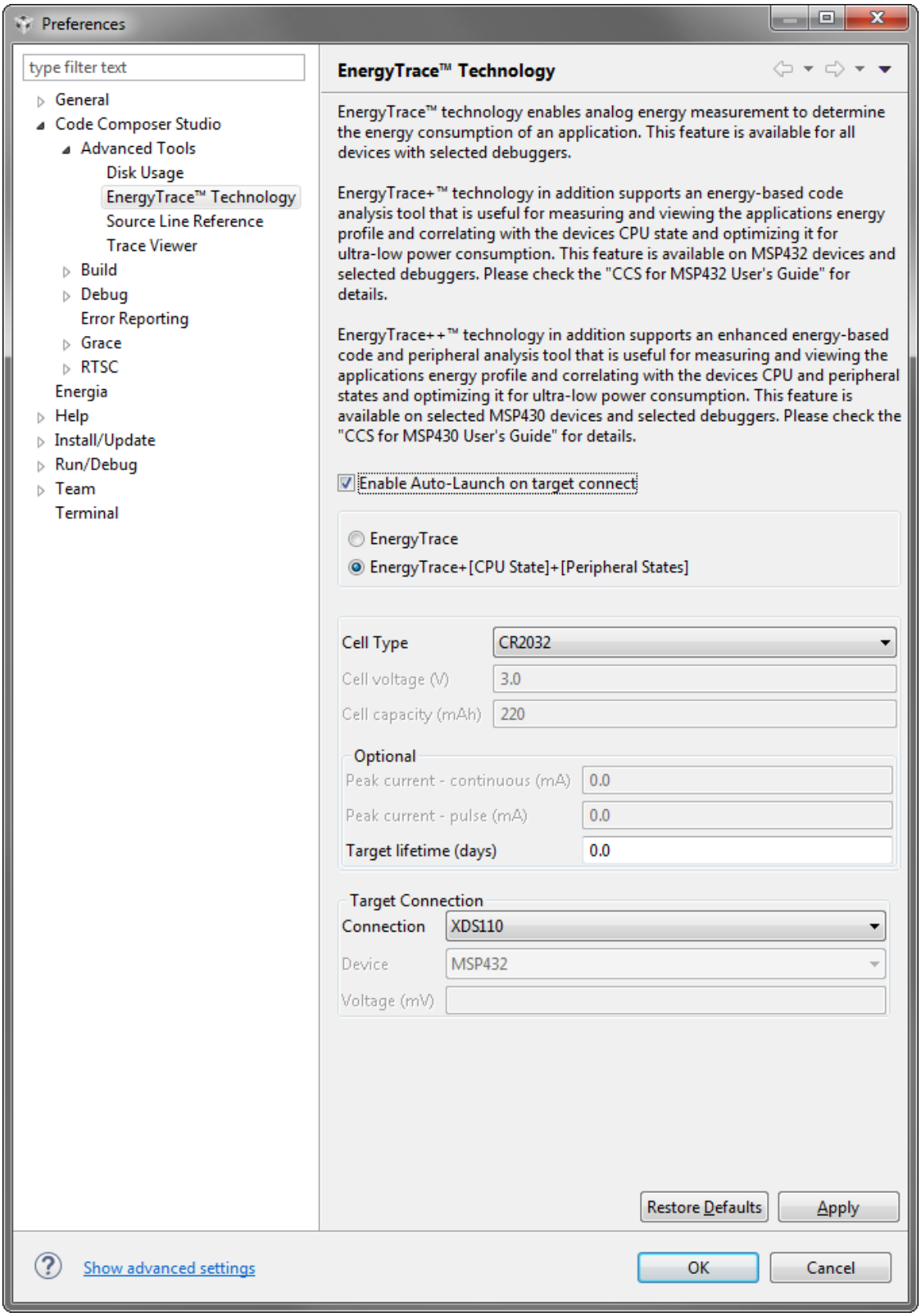


**Figure 8. EnergyTrace Technology Preferences**

Starting a debug session will now open EnergyTrace technology windows. These windows show energy, power, profile, and states to give the user a full view of the energy profile of their application.

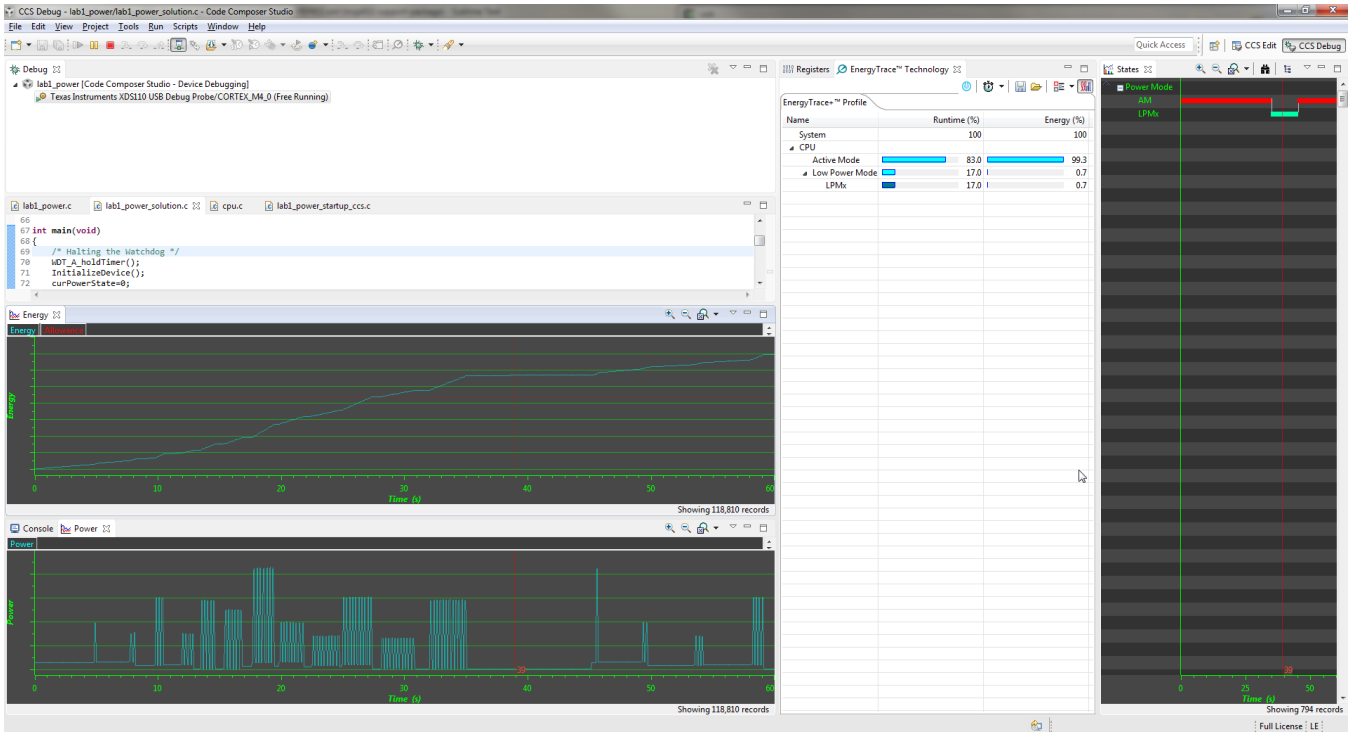


**Figure 9. EnergyTrace Windows**

This data allows the user to see exactly where and how energy is consumed in their application. Optimizations for energy can be quickly made for the lowest power application possible.

On the LaunchPad, EnergyTrace technology measures the current that enters the target side of the LaunchPad. This includes all BoosterPacks plugged in, and anything else connected to the 3V3 power rail. For more information about powering the LaunchPad, see Section 2.4.

For more information about EnergyTrace technology, see http://www.ti.com/tool/energytrace.

For more details and questions about setting up and using EnergyTrace technology with the MSP432P401R, see the Code Composer Studio 6 User's Guide for MSP432.

## 2.4 Power

The board was designed to accommodate various powering methods, including through the on-board XDS110-ET and from an external source or BoosterPack.

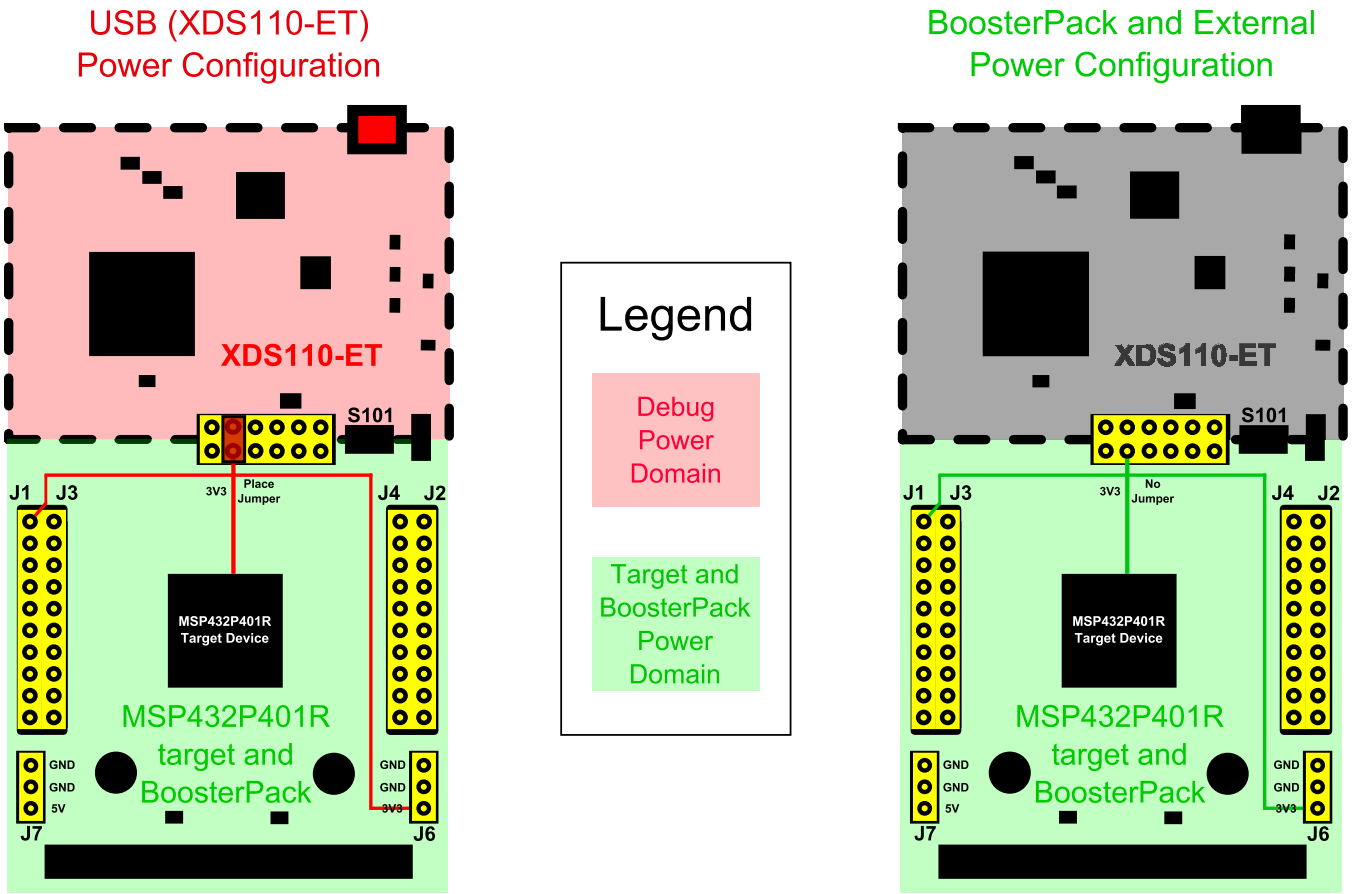


**Figure 10. MSP-EXP432P401R Power Block Diagram**

### 2.4.1 XDS110-ET USB Power

The most common power-supply scenario is from USB through the XDS110-ET debugger. This provides 5-V power from the USB and also regulates this power rail to 3.3 V for XDS110-ET operation and 3.3 V to the target side of the LaunchPad. Power from the XDS110-ET is controlled by the isolation block 3V3 jumper, ensure this jumper is connected for power to be provided to the target MCU side.

Under normal operation, the LDO on the XDS110-ET can supply up to 500 mA of current to the target side including any BoosterPacks plugged in. However, when debugging and using the EnergyTrace technology tool, this current is limited to 75 mA total. Be aware of this current limitation when using EnergyTrace technology.

### 2.4.2 BoosterPack and External Power Supply

Header J6 is present on the board to supply external power directly. It is important to comply with the device voltage operation specifications when supplying external power. The MSP432P401R has an operating range of 1.62 V to 3.7 V. More information can be found in the *MSP432P401xx Mixed-Signal Microcontroller data sheet* .

## 2.5 Measure MSP432 Current Draw

To measure the current draw of the MSP432P401R, use the 3V3 jumper on the jumper isolation block. The current measured includes the target device and any current drawn through the BoosterPack headers.

To measure ultra-low power, follow these steps:

1. Remove the 3V3 jumper in the isolation block, and attach an ammeter across this jumper.
2. Consider the effect that the backchannel UART and any circuitry attached to the MSP432P401R may have on current draw. Disconnect these at the isolation block if possible, or at least consider their current sinking and sourcing capability in the final measurement.
3. Make sure there are no floating input I/Os. These cause unnecessary extra current draw. Every I/O should either be driven out or, if it is an input, should be pulled or driven to a high or low level.
4. Begin target execution.
5. Measure the current. Keep in mind that if the current levels are fluctuating, it may be difficult to get a stable measurement. It is easier to measure quiescent states.

For a better look at the power consumed in the application, use EnergyTrace+ Technology. EnergyTrace+ Technology allows the user to see energy consumed as the application progresses. More details about EnergyTrace+ Technology can be found in Section 2.3.5.

## 2.6 Clocking

The MSP-EXP432P401R provides external clocks in addition to the internal clocks in the device.

- Q1: 32-kHz crystal (LFXTCLK)
- Q2: 48-MHz crystal (HFXTCLK)

The 32-kHz crystal allows for lower LPM3 sleep currents, and higher precision clock source than the default internal 32 kHz REFOCLK. Therefore, the presence of the crystal allows the full range of low-power modes to be used.

The 48-MHz crystal allows the device to run at its maximum operating speed for MCLK and HSMCLK.

The MSP432P401R device has several internal clocks that can be sourced from many clock sources. Most peripherals on the device can select which of the internal clocks to use to operate at the desired speed.

The internal clocks in the device default to the configuration listed in Table 2.

**Table 2. Default Clock Operation**

| Clock | Default Clock Source | Default Clock Frequency | Description |
|---|---|---|---|
| MCLK | DCO | 3 MHz | Master Clock<br>Sources CPU and peripherals |
| HSMCLK | DCO | 3 MHz | Subsystem Master Clock<br>Sources peripherals |
| SMCLK | DCO | 3 MHz | Low-speed subsystem master clock<br>Sources peripherals |
| ACLK | LFXT (or REFO if no crystal present) | 32.768 kHz | Auxiliary clock<br>Sources peripherals |
| BCLK | LFXT(or REFO if no crystal present) | 32.768 kHz | Low-speed backup domain clock<br>Sources LPM peripherals |

For more information about configuring internal clocks and using the external oscillators, see the *MSP432P4xx Family Technical Reference Manual* SLAU356.

## 2.7 BoosterPack Pinout

The MSP-EXP432P401R LaunchPad adheres to the 40-pin LaunchPad pinout standard. A standard was created to aid compatibility between LaunchPad and BoosterPack tools across the TI ecosystem.

The 40-pin standard is compatible with the 20-pin standard that is used by other LaunchPads like the MSP-EXP430FR4133. This allows some subset of functionality of 40-pin BoosterPacks to be used with 20-pin LaunchPads.

While most BoosterPacks are compliant with the standard, some are not. The MSP-EXP432P401R LaunchPad is compatible with all 20-pin and 40-pin BoosterPacks that are compliant with the standard. If the reseller or owner of the BoosterPack does not explicitly indicate compatibility with the MSP-EXP432P401R LaunchPad, compare the schematic of the candidate BoosterPack with the LaunchPad to ensure compatibility. Keep in mind that sometimes conflicts can be resolved by changing the MSP432P401R device pin function configuration in software. More information about compatibility can also be found at http://www.ti.com/launchpad.

Figure 11 shows the 40-pin pinout of the MSP-EXP432P401R LaunchPad.

Note that software configuration of the pin functions plays a role in compatibility. The MSP-EXP432P401R LaunchPad side of the dashed line in Figure 11 shows all of the functions for which the MSP432P401R device's pins can be configured. This can also be seen in the MSP432P401R data sheet. The BoosterPack side of the dashed line shows the standard. The MSP432P401R function whose color matches the BoosterPack function shows the specific software-configurable function by which the MSP-EXP432P401R LaunchPad adheres to the standard.

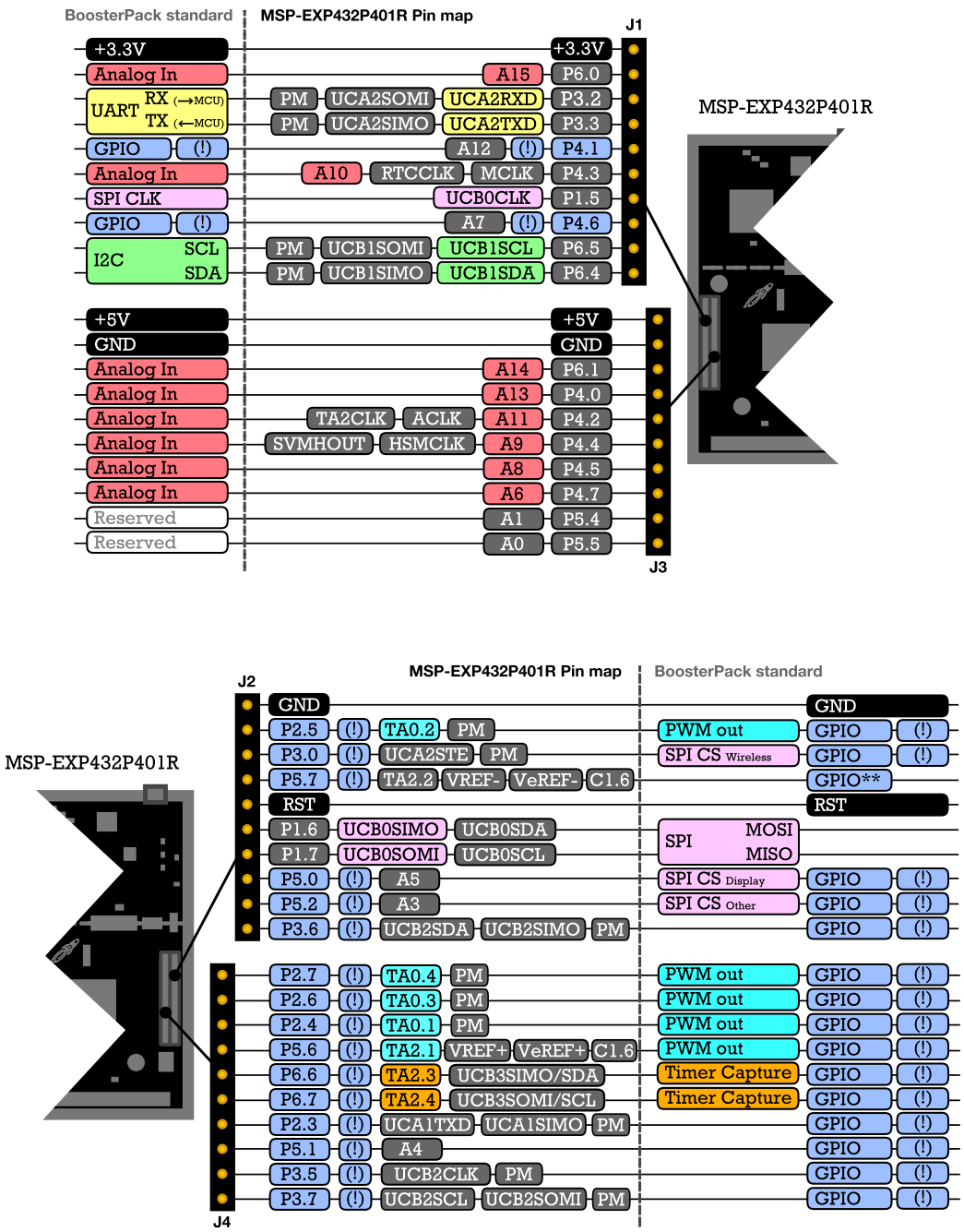


**Figure 11. LaunchPad to BoosterPack Connector Pinout**

## 2.8 Design Files

### 2.8.1 Hardware Design Files

Schematics can be found in Section 6. All design files including schematics, layout, bill of materials (BOM), Gerber files, and documentation are available in the MSP-EXP432P401R Hardware Design Files zip folder.

### 2.8.2 Software Examples

All design files including TI-TXT object-code firmware images, software example projects, and documentation are available in the MSP-EXP432P401R Software Examples zip folder.

## 2.9 Hardware Change Log

**Table 3. Hardware Change Log**

| PCB Revision | Date | Description |
|---|---|---|
| Rev 1.0 | March 2015 | Initial Release |

# 3 Software Examples

There are four software examples included with the MSP-EXP432P401R LaunchPad (see Table 4), which can be found in the MSP-EXP432P401R Software Examples zip folder.

**Table 4. Software Examples**

| Demo Name | BoosterPack Required | Description | More Details |
|---|---|---|---|
| Out-of-Box Software Example | N/A | The out-of-box demo pre-programmed on the LaunchPad from the factory. | Section 3.1 |
| CC3100BOOST MQTT-Twitter LED Control Example | CC3100BOOST | An IoT application demonstrating controlling an RGB LED using Twitter through Wi-Fi | Section 3.2 |
| BOOSTXL-K350QVG-S1 Graphics Library Example | BOOSTXL-K350SVG-S1 | A simple example showing how to use MSP Graphics Library (grlib) to display graphics primitives and images and implement touchscreen functionality | Section 3.3 |
| 430BOOST-SHARP96 Graphics Library Example | 430BOOST-SHARP96 | A simple example showing how to use MSP Graphics Library (grlib) to display graphics primitives and images | Section 3.4 |

To use any of the software examples with the LaunchPad, you must have an integrated development environment (IDE) that supports the MSP432P401R device.

**Table 5. IDE Minimum Requirements for MSP-EXP432P401R**

| Code Composer Studio™ IDE | IAR Embedded Workbench® IDE | Keil™ uVision® IDE |
|---|---|---|
| CCS v6.1 or later | IAR Embedded Workbench for ARM 7.10 or later | Keil uVision MDK-ARM v5 or later |

For more details on how to get started quickly, and where to download the latest CCS, IAR, and Keil IDEs, see Section 4.

## 3.1 Out-of-Box Software Example

This section describes the functionality and structure of the out-of-box software that is preloaded on the EVM. The source code can be found in the MSP-EXP432P401R Software Examples download, or more easily accessible through MSPWare (See 4.3).

The out-of-box software extends a basic blink LED example to allow users to control the blink rate and color of an RGB LED on the MSP432 LaunchPad.

### 3.1.1 Operation

Upon powering up the out-of-box demo, the RGB LED2 blinks red at 1 Hz. Switch S1 can be tapped repeatedly at a constant rate to set the blink frequency of LED2. Switch S2 cycles LED2 through four different color settings: Red, Green, Blue, and random RGB color. Each color setting retains its own blink frequency.

A PC GUI accompanies the out-of-box demo to allow user to set the color and blink rate of the RGB LED. If not already, connect the LaunchPad using the included USB cable to a computer. The out-of-box GUI can be opened from within CCS using the TI Resource Explorer: MSPWare > Development Tools > MSP-EXP432P401R > Examples > Out of Box Experience GUI. A copy can also be found in the MSP-EXP432P401R Software Examples zip download.

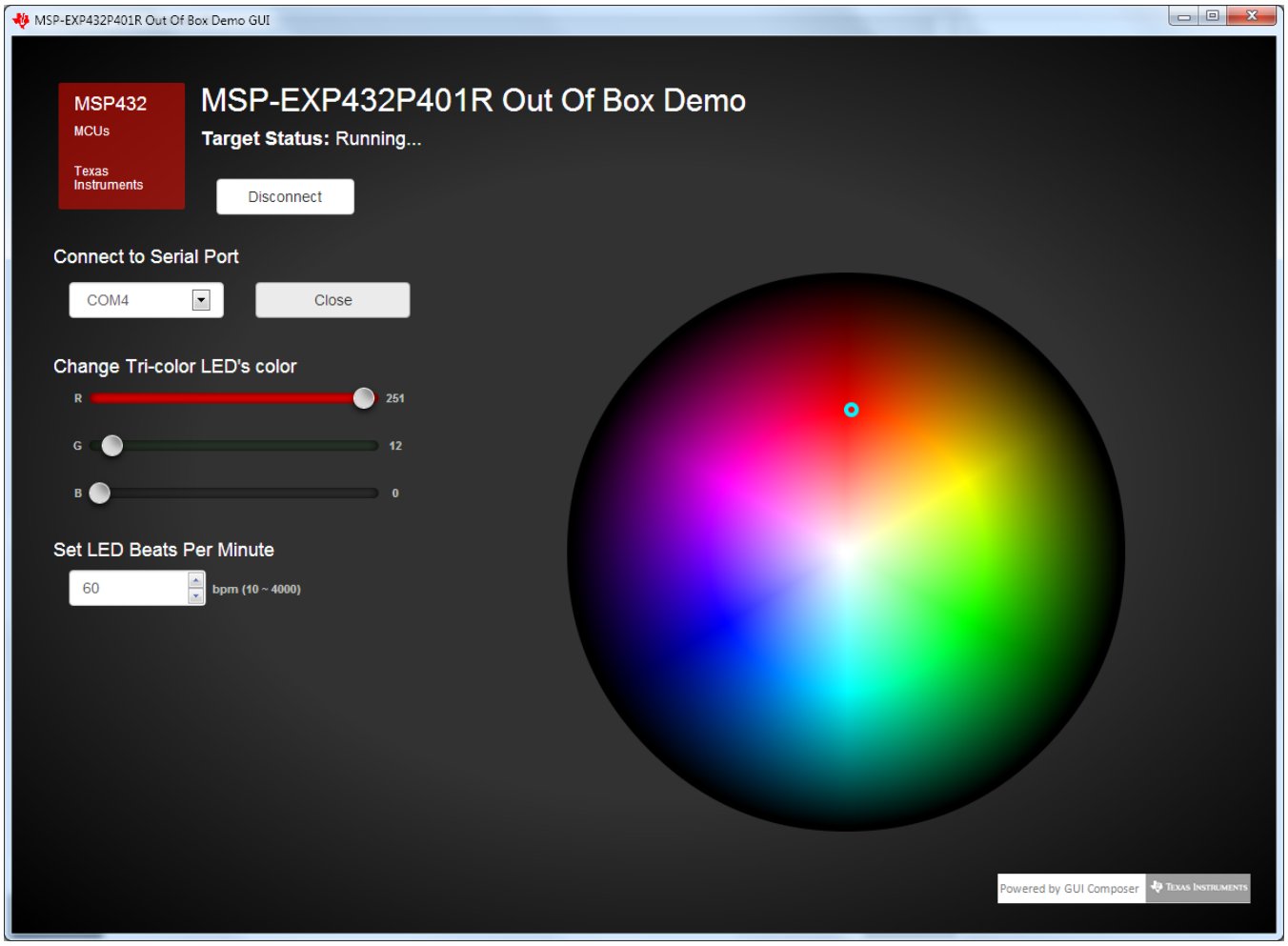


**Figure 12. Out-of-Box GUI Running Locally**

The GUI can also run directly from the TI Cloud Tools (see Section 4.1.1).

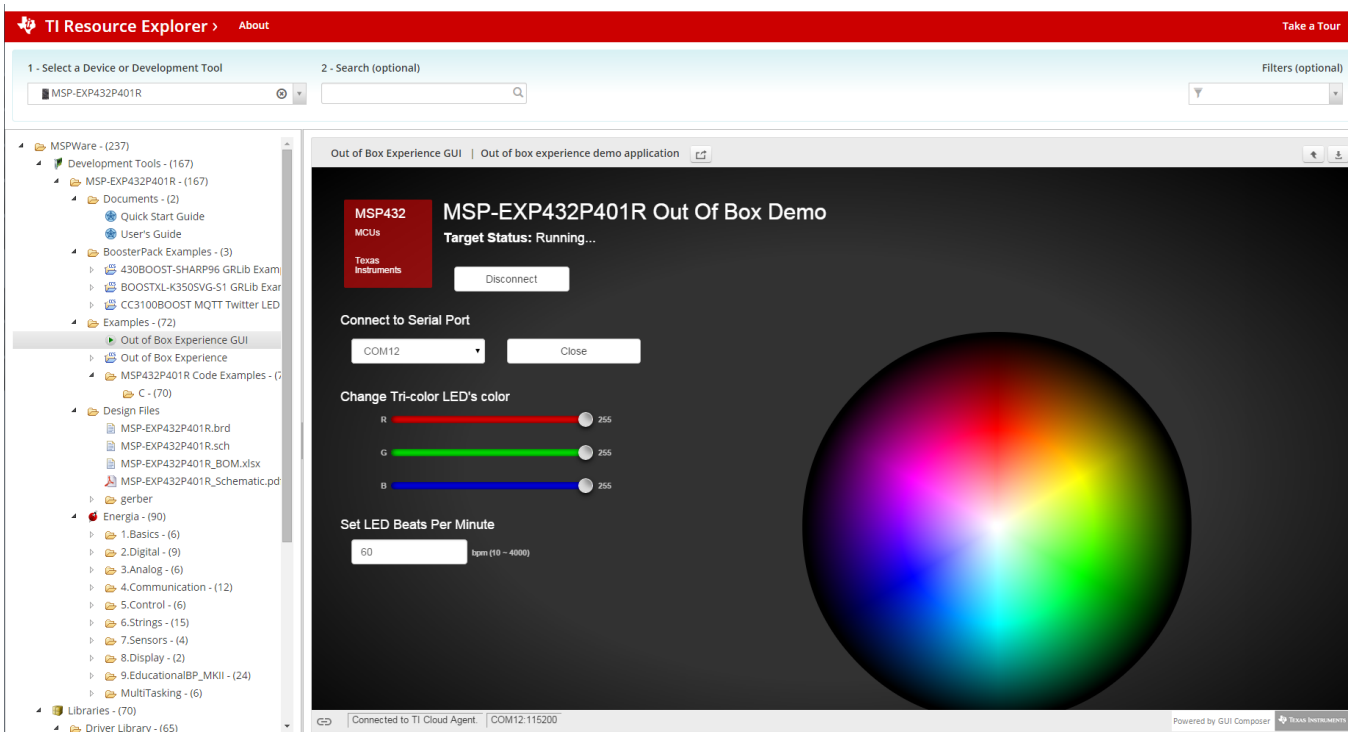


**Figure 13. Out-of-Box GUI Running From TI Cloud Tools**

Click on the Connect button to connect to the LaunchPad then open the serial COM port. Once the connection has been established and the GUI indicates, "Target Status: Running…," you can use the color wheel or the Red, Green, and Blue color sliders to set the color of the LaunchPad RGB LED. Changing the LED Beats Per Minute input box sets the RGB LED blink rate.

## 3.2 CC3100BOOST MQTT-Twitter LED Control Example

This section describes the functionality and structure of the CC3100BOOST MQTT-Twitter LED Control demo that is included in the MSP-EXP432P401R Software Examples download, or more easily accessible through MSPWare (See 4.3).

**Note:** This CC3100BOOST MQTT-Twitter LED Control demo requires the CC3100BOOST BoosterPack to function properly.

This demo uses the MQTT connectivity protocol to realize a simple Internet-of-Things application that allows user to control MSP432 LaunchPad RGB LED wirelessly via Twitter tweets.

### 3.2.1 Source File Structure

The project is split into multiple files. This makes it easier to navigate and reuse parts of it for other projects.

**Table 6. Source File and Folders**

| Name | Description |
|---|---|
| main.c | The demo's main function, shared ISRs, and so on |
| sl_common.h | Common SimpleLink™ technology definitions |
| Driver: board | Board specific driver including basic initializations |
| Driver: cli_uart | Command line interface for backchannel UART communication |
| Driver: spi_cc3100 | MSP432 SPI driver to interface CC3100 |
| Driver: uart_cc3100 | MSP432 UART driver to interface CC3100 |
| Library: mqtt | MQTT protocol library |
| Library: simplelink | Simplelink library containing Wi-Fi APIs |
| Library: driverlib | Device driver library (MSP432DRIVERLIB) |

### 3.2.2 Running the Demo

In order to connect the CC3100BOOST to a wireless access point, start by modifying *SSID_NAME* and *PASSKEY* in the #define section of main.c with your wireless access point's information. You may also need to change SEC_TYPE depending on your access point's setting.

Next, using TI Cloud tools or offline IDEs, build and download the project to the MSP432 LaunchPad. If not already, plug the CC3100BOOST BoosterPack onto the LaunchPad, and connect the LaunchPad to your computer. The CC3100 should automatically try to connect to the access point with the provided credentials. The LaunchPad outputs status messages through its Application/User UART COM port, which can be viewed by opening it using terminal applications (see Section 2.3.2).

Once the CC3100 has established internet connection and successfully subscribed to the public MQTT broker server, the LaunchPad RGB LED is ready to be controlled with Twitter. Any **public** tweets in the following format will change the RGB LED color on **all** MSP-EXP432P401R LaunchPad running this demo:

RGB(red_value, green_value, blue_value) #MSP432LaunchPad

The color parameters can be integers ranging from 0 to 255.

Pressing the left push button S1 on the LaunchPad publishes a 32-bit unique ID from the LaunchPad to the MQTT broker, which then gets tweeted by the twitter account, @MSPLaunchPad. You may then use this 32-bit Unique ID in your tweet message to control the RGB LED on the specific LaunchPad+CC3100BOOST combination tied to that unique ID:

<32-bit unique ID> RGB(red_value, green_value, blue_value) #MSP432LaunchPad

### 3.2.3 Overview of Backend Servers

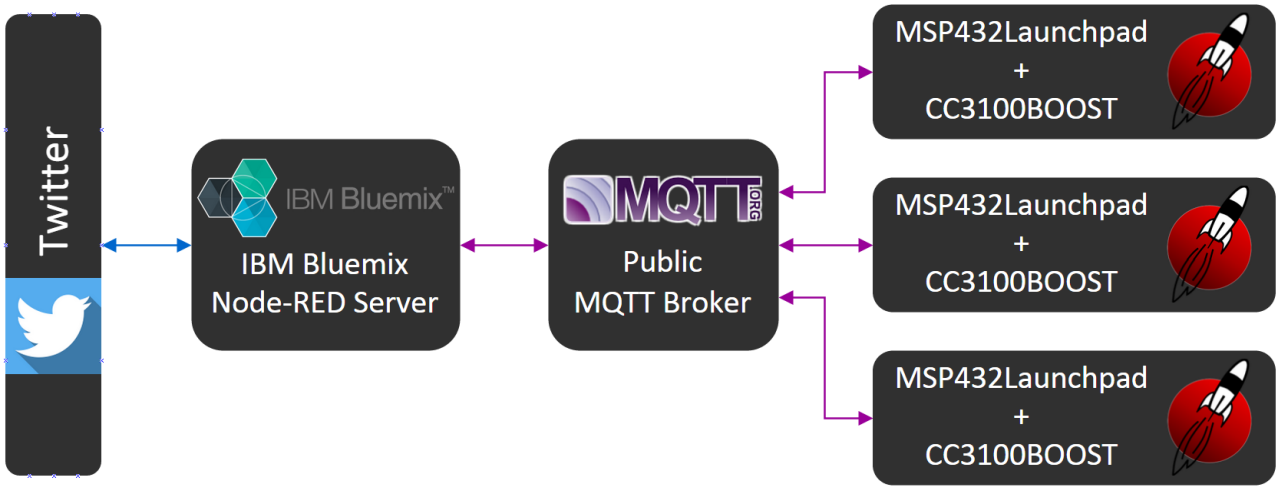


**Figure 14. Backend Block Diagram of CC3100BOOST MQTT-Twitter LED Control Demo**

As shown in the above Figure 14, inputs from either the MSP432 LaunchPad or Twitter travel through a couple of intermediary servers before reaching the output on the opposite end. Instead of interacting with Twitter server directly through the more resource intensive HTTP, the MSP432 LaunchPad communicates with the cloud solely through MQTT protocol. MQTT is a publish-subscribe messaging protocol designed for lightweight M2M communications. Multiple clients sends message to one another through a server known as a broker, and each client can publish messages to different topics and subscribe to multiple topics. While a dedicated MQTT broker can be setup for an application, this demo uses one of the several MQTT brokers that are freely available to the public, http://iot.eclipse.org/sandbox.html.

Every LaunchPad running the CC3100BOOST MQTT-Twitter LED Control demo subscribes to the MQTT topic, *"/msp/cc3100/demo"*. This is why any RGB data published to this topic will change the LED color on all LaunchPads running this demo. However, each LaunchPad also subscribes to an *"<uniqueID>"* topic that can be used to control LaunchPads individually.

A cloud server is also setup/maintained by the MSP Team using the IBM Bluemix cloud platform service. This server runs a couple of Node-RED applications that interface with Twitter directly through HTTP. After processing public tweets containing #MSP432LaunchPad, the Node-RED server also act as a MQTT client, publishing color information to either the *"/msp/cc3100/demo"* or *"<uniqueID>"* topic, which then gets received on subscribed LaunchPads. Conversely, unique id data published by the LaunchPads to the *"/msp/cc3100/demo/fromLP"* topic gets received by the Node-RED server, which then tweets a time stamped message on the Twitter account @MSPLaunchPad.

Check out IBM Bluemix to see how you can also build your own cloud application.

### 3.2.4 Developing With CC3100BOOST BoosterPack

A SimpleLink Wi-Fi CC3100 Software Development Kit (SDK) can be downloaded at http://www.ti.com/tool/cc3100sdk. It contains drivers, many sample applications for Wi-Fi features and internet, and documentation needed to use the CC3100 Internet-on-a-chip™ solution.

The CC3100BOOST MQTT-Twitter LED Control Demo was developed on CC3100SDK_1.0.0. Service pack update may be required on the CC3100BOOST with newer SDK release. Refer to the CC3100 SimpleLink Wi-Fi and IoT Solution Getting Started Guide for more information.

### 3.3 BOOSTXL-K350QVG-S1 Graphics Library Example

This software is available in the MSP-EXP432P401R Software Examples zip download, or more easily accessible through MSPWare (see Section 4.3).

The demo shows how to use the *MSP Graphics Library* http://www.ti.com/tool/msp-grlib or "grlib," in a project with the Kentec display. This demo shows the user how to enable the touch screen, create buttons, and use graphics primitives including colors and images.

The program begins by calibrating the touch screen. There is a routine that detects the four corner coordinates to determine if an eligible rectangle boundary is formed. If the calibration was incorrect, a message will display on the screen indicating the calibration failed. When successful, the calibration provides a reference for all button presses throughout the rest of the program.

The next step is to select the mode of the program- display primitives or images. Each mode simply cycles through without user interaction to show off features of the display. In the graphics primitives mode, the following primitives are shown:

- Pixels
- Lines
- Circles
- Rectangles
- Text

The application is heavily commented to ensure it is very clear how to use the grlib APIs. The above primitives are shown as well as the underlying concepts of grlib including background and foreground colors, context, fonts, opacity, and more.

The images mode shows the drawing of a few different images both compressed and uncompressed. Image compression can have a big impact to drawing speeds for simple images. To draw images with the MSP Graphics Library, they must first be converted into the right file format. These files can be generated by the Image Reformer tool that comes packaged with grlib. Launch this tool from the grlib folder or direct from TI Resource Explorer.

- File Path: <grlib root>\utils\image-reformer\imagereformer.exe
- TI Resource Explorer > MSPWare > Libraries > Graphics Library > MSP430 Image Reformer

The Image Reformer tool allows you to import images and output into grlib specific files to add to your grlib project. Image Reformer does not manipulate any images (such as color modifications, rotation, or cropping), any image manipulation must be done before importing into the Image Reformer tool. More information about MSP grlib and the Image Reformer tool can be found in the Design Considerations when Using MSP430 Graphics Library application note.

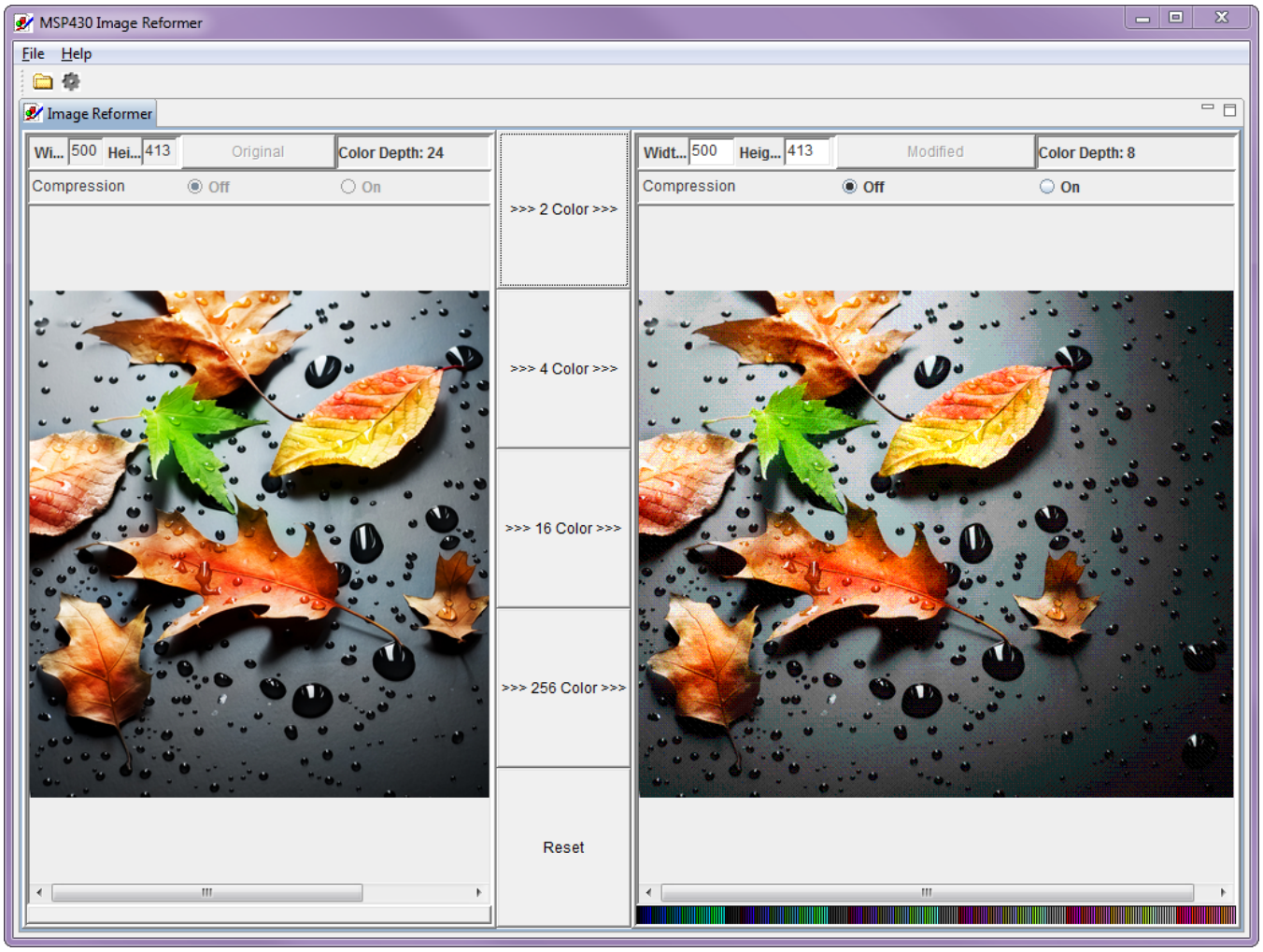


**Figure 15. Importing and Converting an Image With MSP Image Reformer**

### 3.4 430BOOST-SHARP96 Graphics Library Example

This software example is similar to the BOOSTXL-K350QVG-S1 Graphics library example. It shows how to use the *MSP Graphics Library* http://www.ti.com/tool/msp-grlib or "grlib," in a project with the Sharp 96×96 display. The Sharp 96×96 display BoosterPack does not support touch or color, it is a simple monochrome LCD. It is a great LCD for ultra-low power display applications and has a unique mirrored pixel display.

This demo cycles screens without user interaction to show simple graphics primitives.

- Pixels
- Lines
- Circles
- Rectangles
- Text
- Images

This demo introduces the functions to configure grlib such as initialization, color inversion, and using foreground and background colors properly.

# 4 Resources

## 4.1 Integrated Development Environments

Although the source files can be viewed with any text editor, more can be done with the projects if they're opened with a development environment like Code Composer Studio™ (CCS), Keil™ uVision®, IAR Embedded Workbench®, or Energia.

### 4.1.1 TI Cloud Development Tools

TI's Cloud-based software development tools provide instant access to MSPWare content and a web-based IDE.

#### 4.1.1.1 TI Resource Explorer Cloud

TI Resource Explorer Cloud provides a web interface for browsing examples, libraries and documentation found in MSPWare without having to download files to your local drive.

Go check out TI Resource Explorer Cloud now at dev.ti.com.

#### 4.1.1.2 Code Composer Studio Cloud

Code Composer Studio Cloud is a web-based IDE that allows code edit, compile and download to devices right from your web browser. It also integrates seamlessly with TI Resource Explorer Cloud with the ability to import projects directly on the cloud.

A full comparison between CCS Cloud and CCS Desktop is available here.

See Code Composer Studio Cloud now at dev.ti.com.

### 4.1.2 Code Composer Studio

Code Composer Studio Desktop is a professional integrated development environment that supports TI's Microcontroller and Embedded Processors portfolio. Code Composer Studio comprises a suite of tools used to develop and debug embedded applications. It includes an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler, and many other features.

**Note**: MSP432 LaunchPad requires CCS Version 6.1.0 or later. Refer to the *Code Composer Studio 6.1 for MSP432 User's Guide* (SLAU575) for detailed instructions of using the IDE with MSP432.

You can learn more about CCS and download it at http://www.ti.com/tool/ccstudio.

### 4.1.3 Keil uVision

uVision IDE is an embedded project development environment included in Keil's Microcontroller Development Kit Version 5, that provides an source code editor, project manager, and make utility tool. µVision supports all the Keil tools including C/C++ Compiler, Macro Assembler, Linker, Library Manager, and Object-HEX Converter.

**Note**: MSP432 LaunchPad requires uVision IDE/MDK Version 5 or later. Refer to the *ARM Keil MDK 5 IDE for MSP432 User's Guide* (SLAU590) for detailed instructions of using the IDE with MSP432.

You can learn more about Keil uVision and download it at http://www.keil.com/arm/mdk.asp.

### 4.1.4 IAR Embedded Workbench for ARM

IAR Embedded Workbench for ARM is another very powerful integrated development environment that allows you to develop and manage complete embedded application projects. It integrates the IAR C/C++ Compiler, IAR Assembler, IAR ILINK Linker, editor, project manager, command line build utility, and IAR C-SPY Debugger.

**Note**: MSP432 LaunchPad requires IAR Embedded Workbench for ARM Version 7.10 or later. Refer to the *IAR Embedded Workbench for ARM 7.40.2 for MSP432 User's Guide* (SLAU574) for detailed instructions of using the IDE with MSP432.

You can learn more about IAR Embedded Workbench and download it at https://www.iar.com/iar-embedded-workbench/arm.

### 4.1.5 Energia

Energia is a simple open-source community-driven code editor that is based on the Wiring and Arduino framework. Energia provides unmatched ease of use through very high level APIs that can be used across hardware platforms. Energia is a light-weight IDE that doesn't have the full feature set of CCS, Keil, or IAR. However, Energia is great for anyone who wants to get started very quickly or who doesn't have significant coding experience.

You can learn more about Energia and download it at www.energia.nu.

## *4.2 LaunchPad Websites*

More information about the LaunchPad, supported BoosterPacks, and available resources can be found at:

- MSP-EXP432P401R Tool Folder: resources specific to this particular LaunchPad kit
- TI's LaunchPad portal: information about all LaunchPad kits from TI

## *4.3 MSPWare and TI Resource Explorer*

TI Resource Explorer is a tool integrated into CCS that allows you to browse through available design resources. TI Resource Explorer will help you quickly find what you need inside packages including MSPWare, ControlSuite, TivaWare and more. TI Resource Explorer is well organized to find everything that you need quickly, and you can import software projects into your workspace in one click!

TI Resource Explorer Cloud is one of the TI Cloud Development tools, and is tightly integrated with CCS Cloud. See Section 4.1.1 for more information.

MSPWare is a collection of code examples, software libraries, data sheets, and other design resources for all MSP devices delivered in a convenient package – essentially everything developers need to become MSP experts!

In addition to providing a complete collection of existing MSP design resources, MSPWare also includes a high level API called MSP Driver Library. This library makes it easy to talk to MSP hardware. More information can be found at http://www.ti.com/tool/mspware.

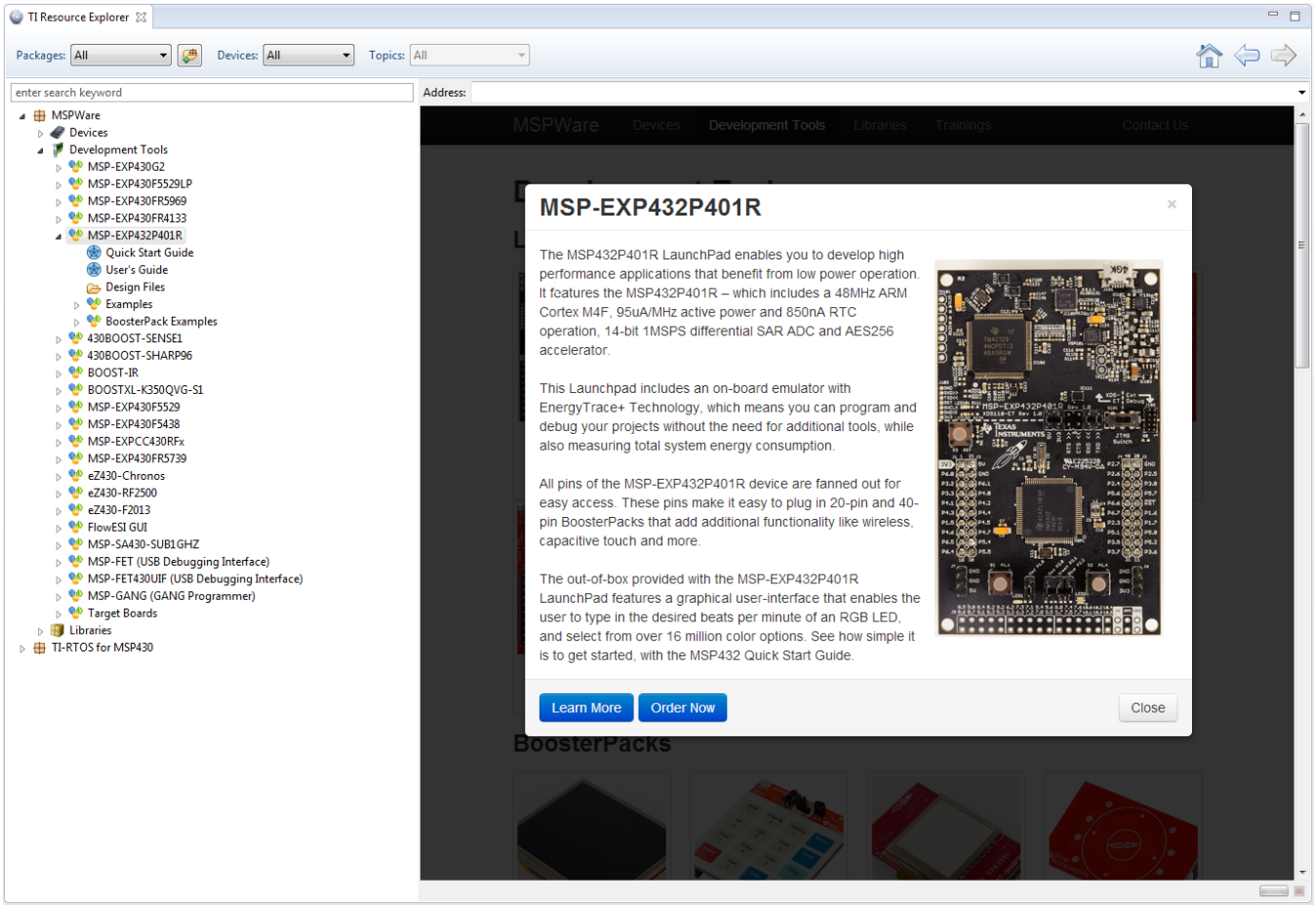


**Figure 16. Using TI Resource Explorer to Browse MSP-EXP432P401R in MSPWare**

Inside TI Resource Explorer, these examples and many more can be found, and easily imported into CCS with one click.

## 4.4 MSP432P401R

### 4.4.1 Device Documentation

At some point, you will probably want more information about the MSP432P401R device. For every MSP device, the documentation is organized as shown in Table 7.

**Table 7. How MSP Device Documentation is Organized**

| Document | For MSP432P401R | Description |
|---|---|---|
| Device family user's guide | *MSP432P4xx Family Technical Reference Manual* | Architectural information about the device, including all modules and peripherals such as clocks, timers, ADC, and so on. |
| Device-specific data sheet | *MSP432P401xx Mixed-Signal Microcontroller data sheet* | Device-specific information and all parametric information for this device |

### 4.4.2 MSP432P401R Code Examples

This is a set of very simple MSP432P401xx code examples that demonstrate how to use the entire set of MSP432 peripherals: serial communication, ADC14, Timer_A, Timer_B, and so on. These examples show both the direct register access and driver library methods.

Every MSP derivative has a set of these code examples. When starting a new project or adding a new peripheral, these examples serve as a great starting point.

### 4.4.3 MSP432 Application Notes and TI Designs

There are many application notes that can be found at www.ti.com/msp432, as well as TI Designs with practical design examples and topics.

## 4.5 Community Resources

### 4.5.1 TI E2E Community

Search the E2E™ forums at e2e.ti.com. If you cannot find your answer, post your question to the community!

### 4.5.2 Community at Large

Many online communities focus on the LaunchPad; for example, http://www.43oh.com. You can find additional tools, resources, and support from these communities.

# 5 FAQ

*Q: I can't program my LaunchPad; the IDE can't connect to target. What's wrong?*

A: Check the following:

- Is the JTAG switch (S101) in the correct orientation?
  - Switch to left for XDS110-ET onboard debugger
  - Switch to the right for external debugger connection
    - If using an external debugger, is USB power provided?
- Check the debugger settings: change to Serial Wire Debug (SWD) without SWO
  - Under targetconfigs double-click the *.ccxml file
  - Click the Advanced tab at the bottom
  - Click on Texas Instruments XDS110 USB Debug Probe
  - Under Connection Properties, change SWD Mode Settings to Use SWD Mode with SWO Trace Disabled

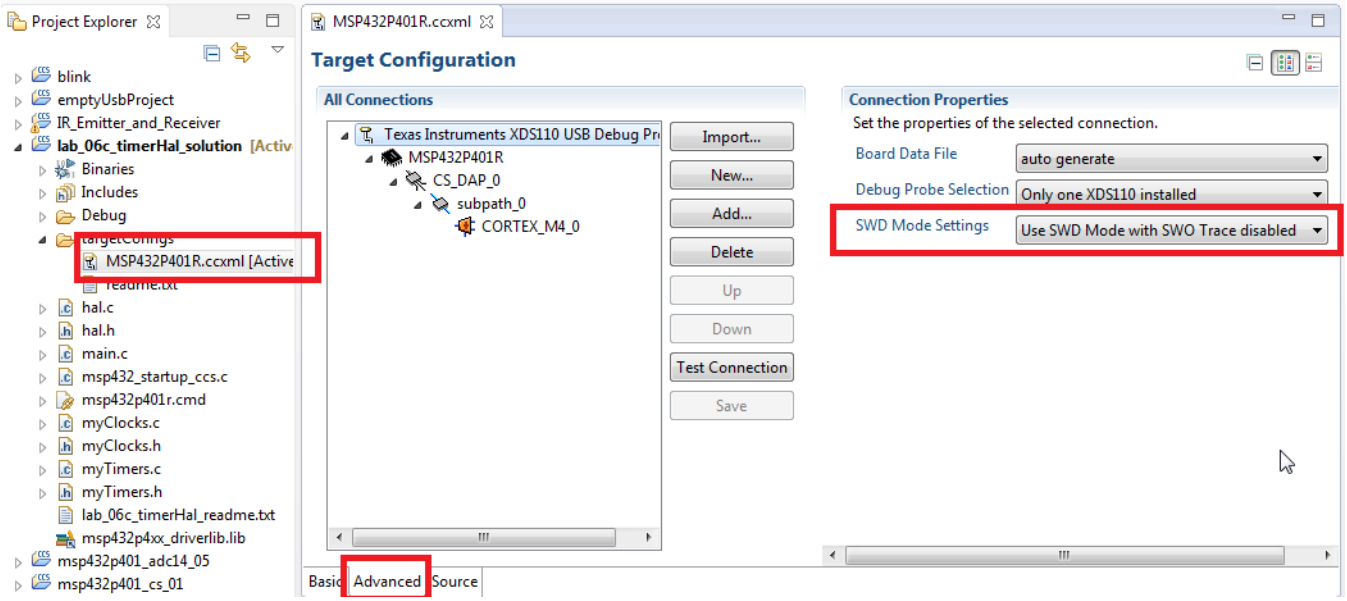


**Figure 17. SWD Mode Settings**

  - When the settings of Port J (PJSEL0 and PJSEL1 bits) are changed, full JTAG access is prevented on these pins. Changing to use SWD allows access through the dedicated debug pins only.
- If even this can't connect, reset the device to factory settings
  - Click **View** → **Target Configurations**. CCS shows the target configuration.

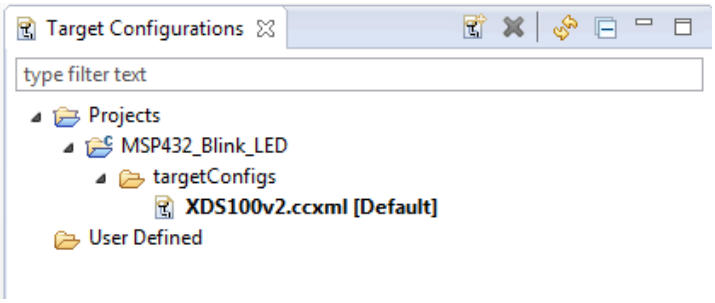


**Figure 18. Target Configurations**

If using the onboard emulator, XDS110-ET is shown.

– Right click **Launch Selected Configuration**.

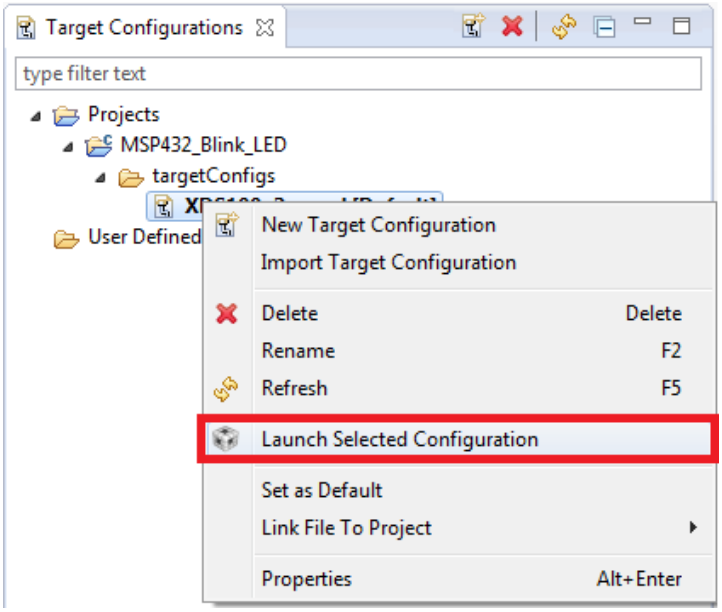


**Figure 19. Launch Selected Configuration**

– The debugger will now connect to your device (which is still possible), but does not try to halt the CPU, write to registers or even download code (which would not be possible). The Debug view that is spawned shows the CPU core, but marks it as disconnected.

– Right click **Show all cores**.

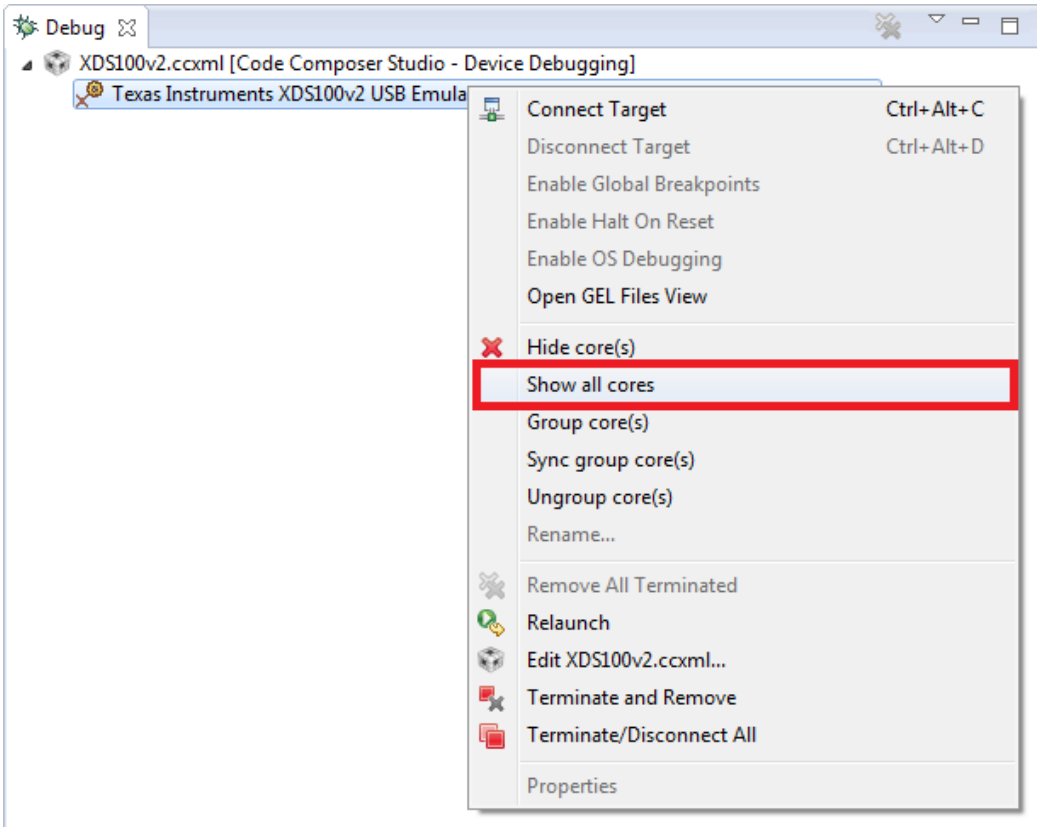


**Figure 20. Show All Cores**

The MSP432 Debug Access Port, or DAP, is shown under Non Debuggable devices.

– Right click **Connect Target**

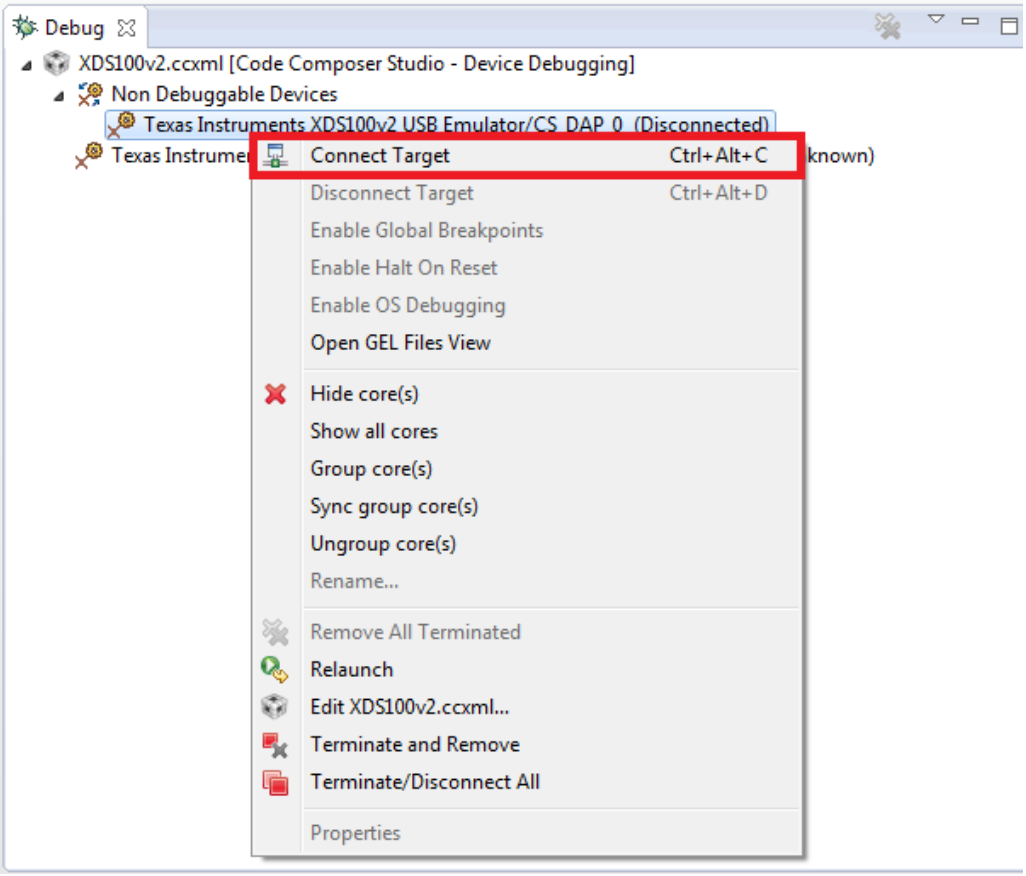


**Figure 21. Connect Target**

Now a script needs to be run to return the device back to factory settings

Scripts > default > MSP432_Factory_Reset

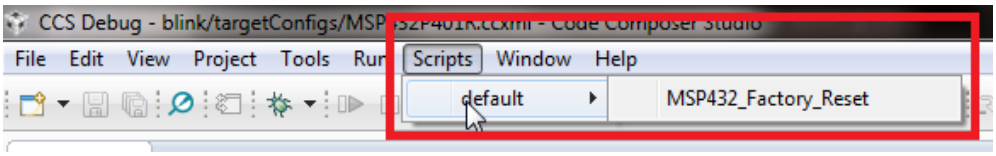


**Figure 22. MSP432_Factory_Reset Script**

- These instructions are generally the same for all IDEs, but the exact steps may vary slightly by IDE. See the MSP432 IDE User guides below for additional details:
  – Code Composer Studio 6 User's Guide for MSP432
  – Keil uVision IDE Version 5 for MSP432 User's Guide
  – IAR Embedded Workbench for ARM 7.10 User's Guide for MSP432

*Q: How do I use the LaunchPad and my Segger J-Link to debug the target externally? It won't connect to the onboard connector.*

A: The Segger J-Link is the only major ARM debugger that doesn't come with an adapter for the 10-pin small pitch ARM connector. The adapter cable is found here, and can be purchased from Digi-Key here

*Q: Why doesn't the back-channel UART on the MSP432 LaunchPad work with my serial terminal program at speeds faster than 56000 baud?*

A: Certain serial terminal programs such as HTerm or the CCS built-in terminal might not work with the MSP432 LaunchPad at specific baudrates, resulting in the software not being able to open the virtual COM port or in the baud rate getting configured incorrectly. An issue with the LaunchPad emulator firmware has been identified and will be fixed in the next release. Until the update is available, use Tera Term, ClearConnex, or HyperTerminal instead or reduce the baud rate to speeds of 38,400 baud or lower.

*Q: Problems plugging the MSP432 LaunchPad into a USB3.0 Port*

A: It has been observed that when the MSP432 LaunchPad is connected to USB3.0 ports provided by a certain combination of USB3.0 host controller hardware and associated device drivers that the IDE is unable to establish a debug session with the LaunchPad, resulting in an error message like “CS_DAP_0: Error connecting to the target: (Error -260 @ 0x0) An attempt to connect to the XDS110 failed.” in the case of Code Composer Studio. In this case the CCS-provided low-level command line utility ‘xdsdfu' will also not be able to establish a connection with the LaunchPad.

Specifically, this issue was observed on PCs running Windows 7 that show the “Renesas Electronics USB 3.0 Host Controller” and the associated “Renesas Electronics USB 3.0 Root Hub” in the device manager. After updating the associated Windows USB drivers to more recent versions obtained from the hardware vendor the issue went away. There might be other USB3.0 hardware and device driver combinations that will lead to the same issue. If you think you might be affected try contacting your PC vendor or try locating and installing more recent versions of the USB3.0 device drivers. Alternatively, connect the LaunchPad to an USB2.0 port on your PC if available.

*Q: I can't get the backchannel UART to connect. What's wrong?*

A: Check the following:

- Do the baud rate in the host's terminal application and the eUSCI settings match?
- Are the appropriate jumpers in place, on the isolation jumper block?
- Probe on RXD and send data from the host. If you don't see data, it might be a problem on the host side.
- Probe on TXD while sending data from the MSP432. If you don't see data, it might be a configuration problem with the eUSCI module.
- Consider the use of the hardware flow control lines (especially for higher baud rates).

# 6 Schematics

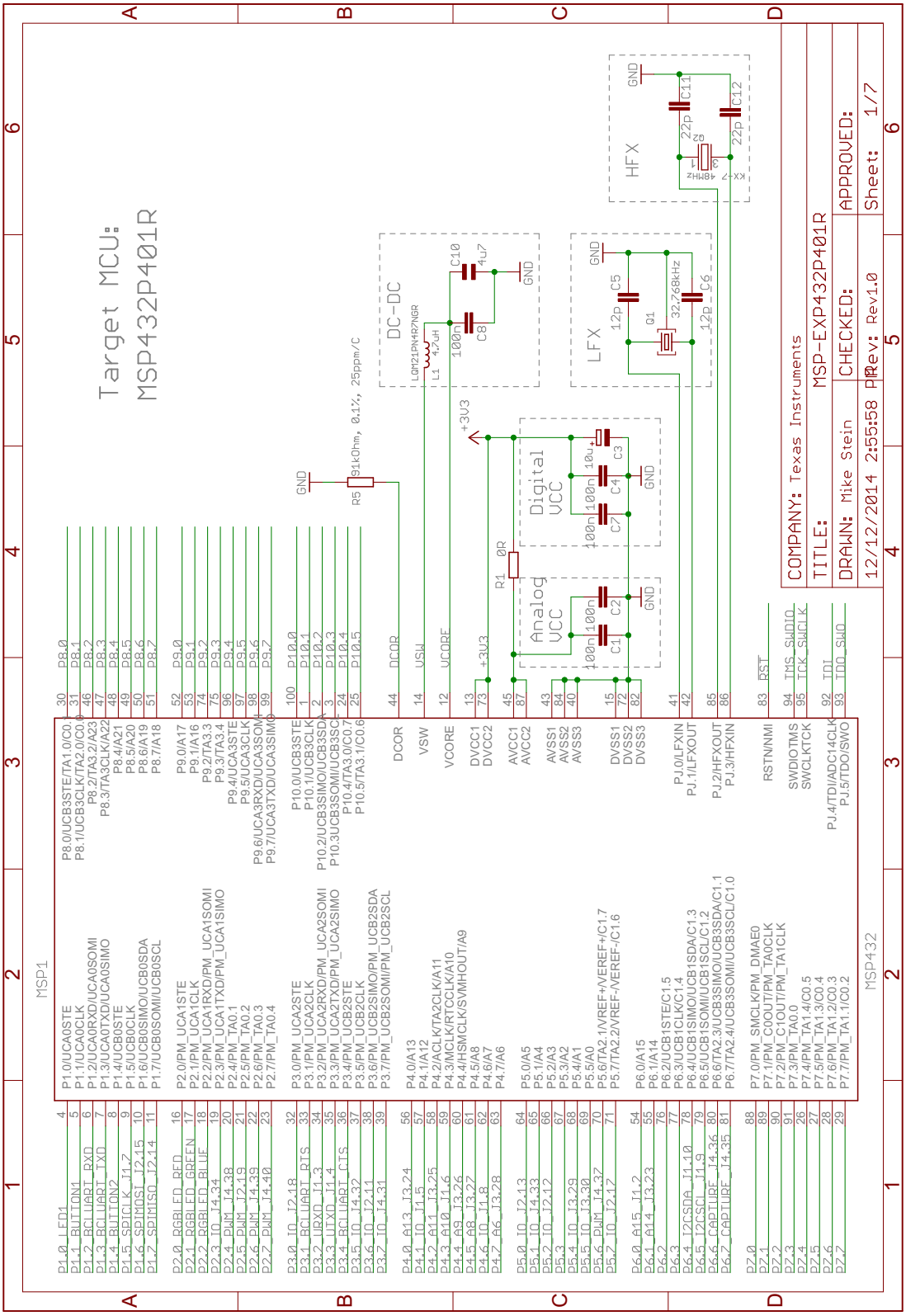


**Figure 23. Schematics (1 of 7)**

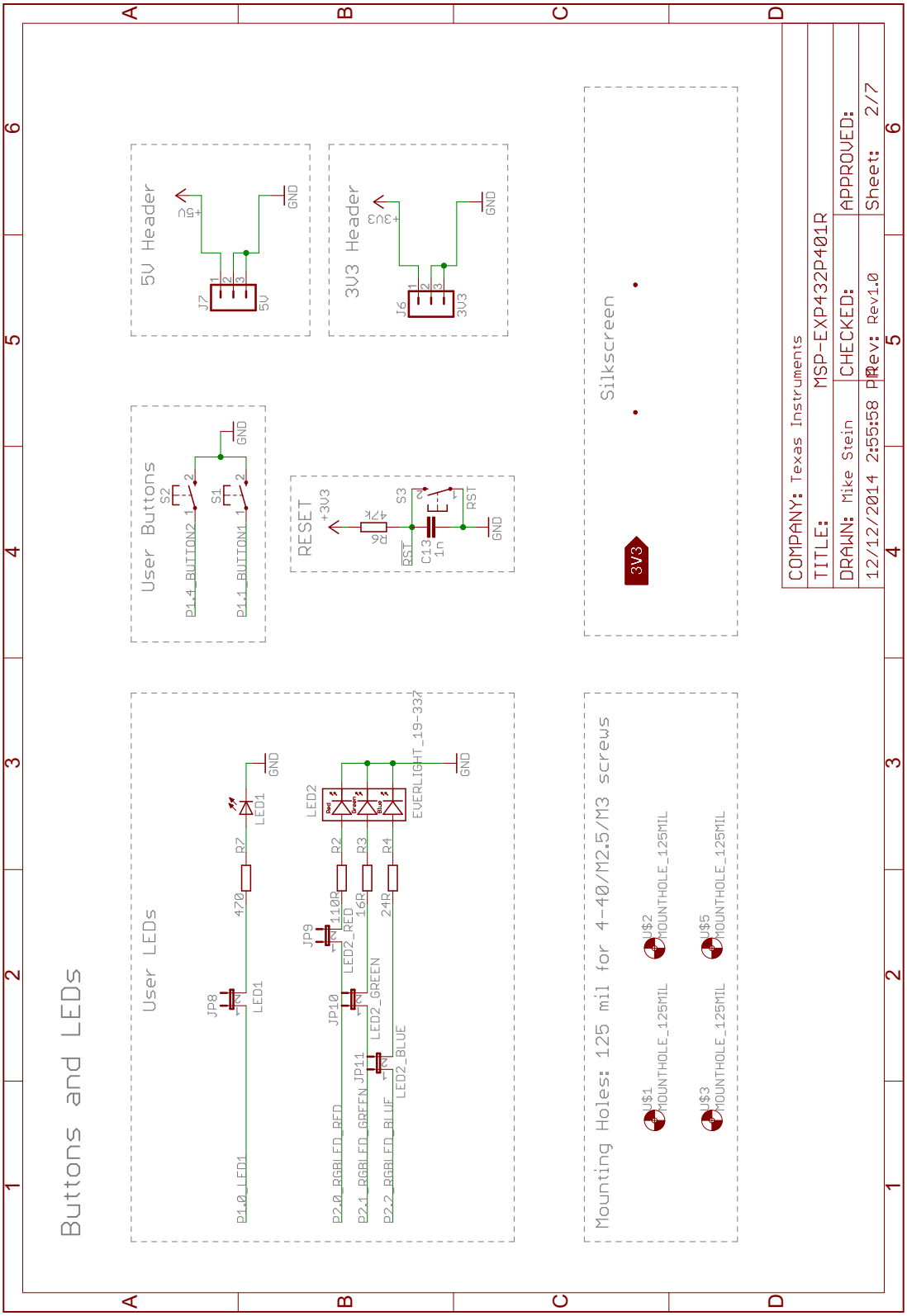


**Figure 24. Schematics (2 of 7)**

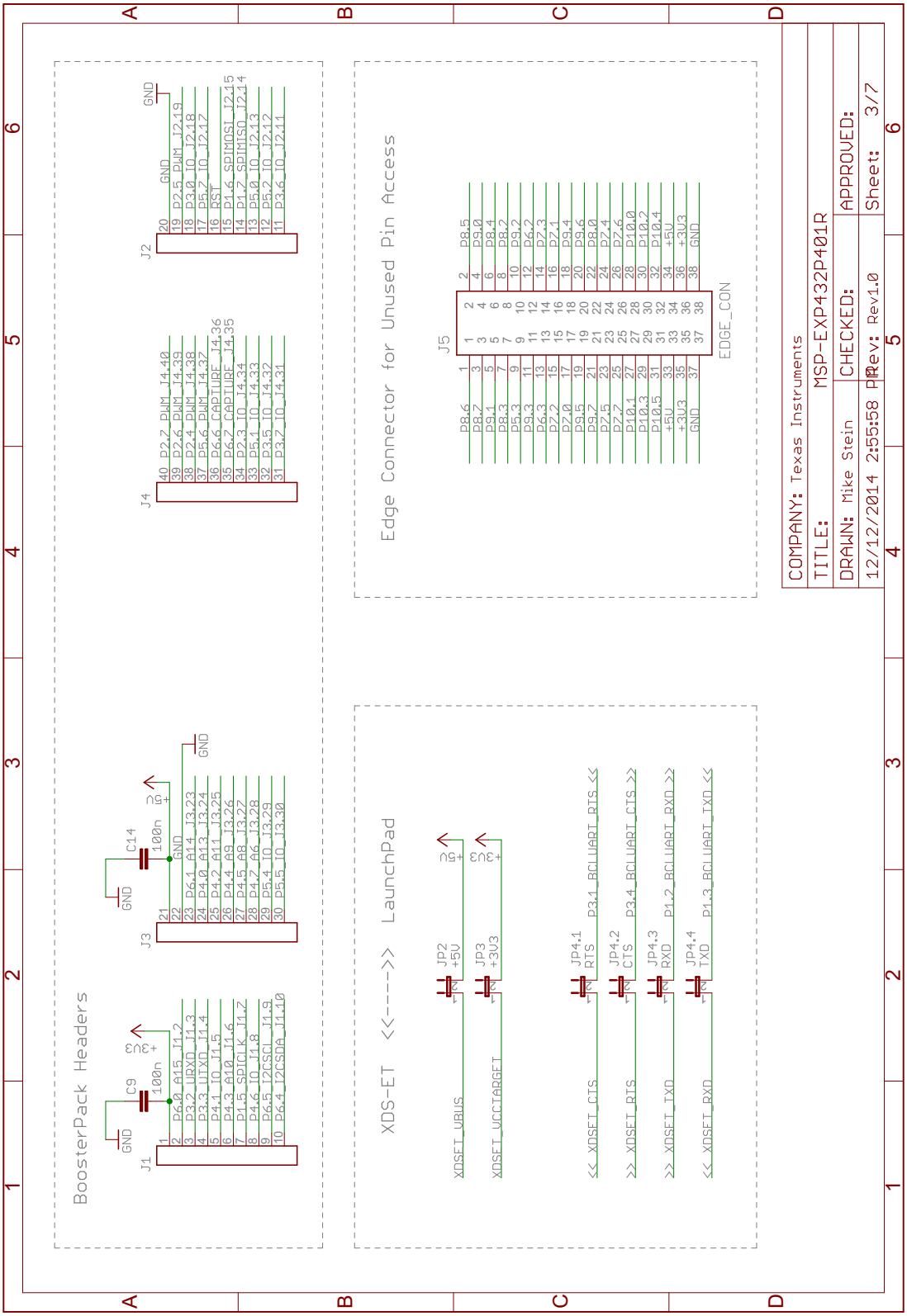


**Figure 25. Schematics (3 of 7)**

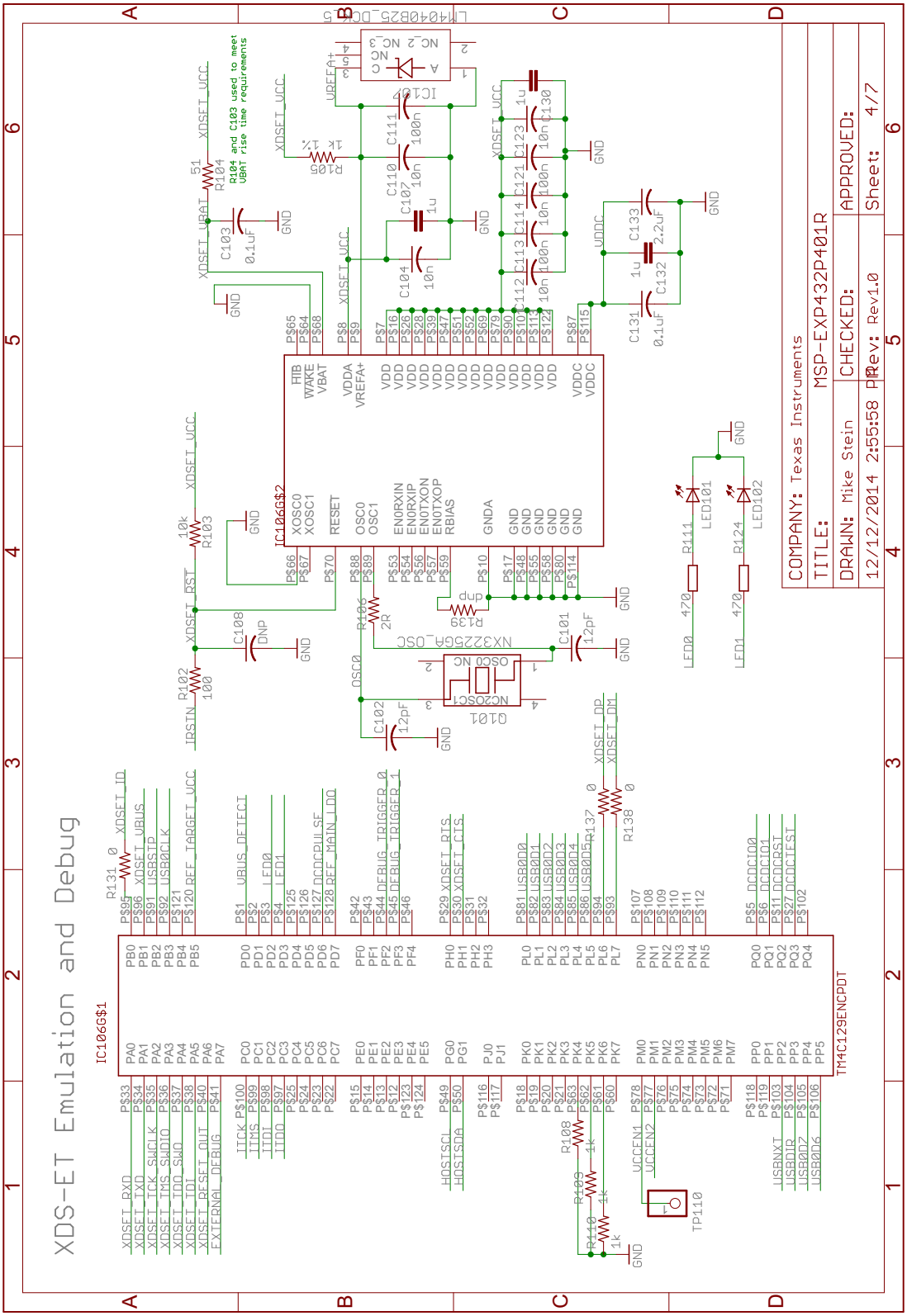


**Figure 26. Schematics (4 of 7)**

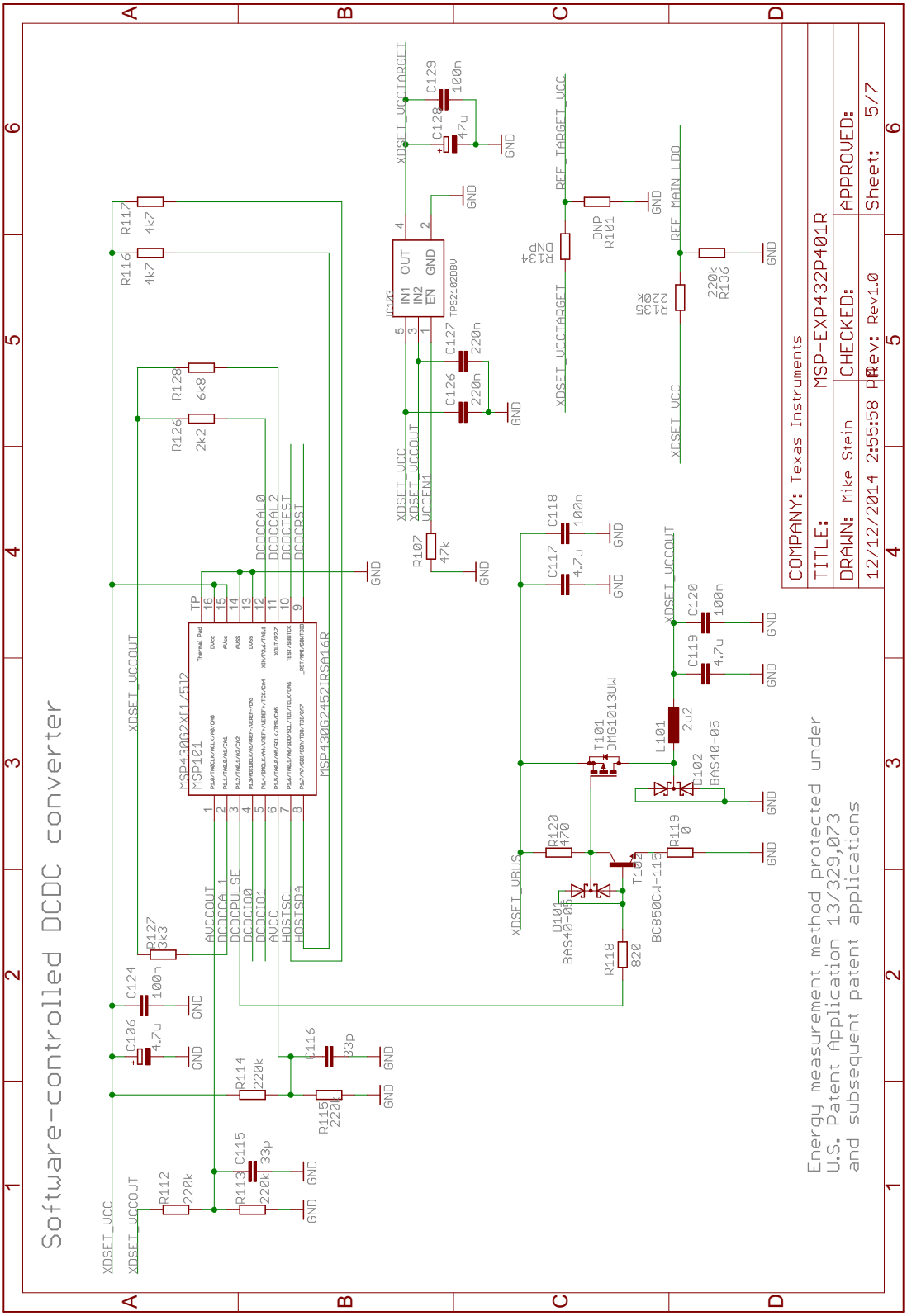


**Figure 27. Schematics (5 of 7)**

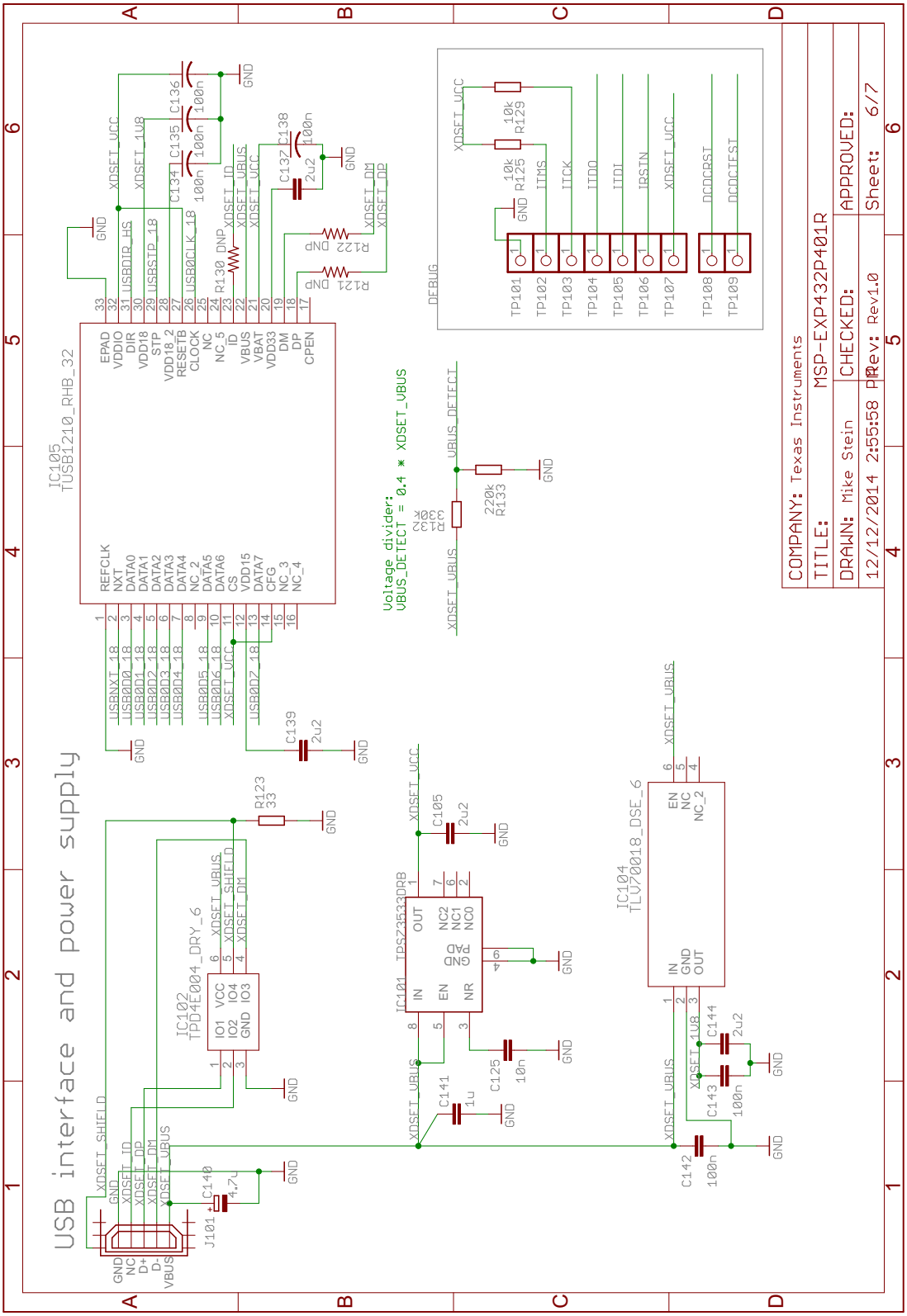


**Figure 28. Schematics (6 of 7)**

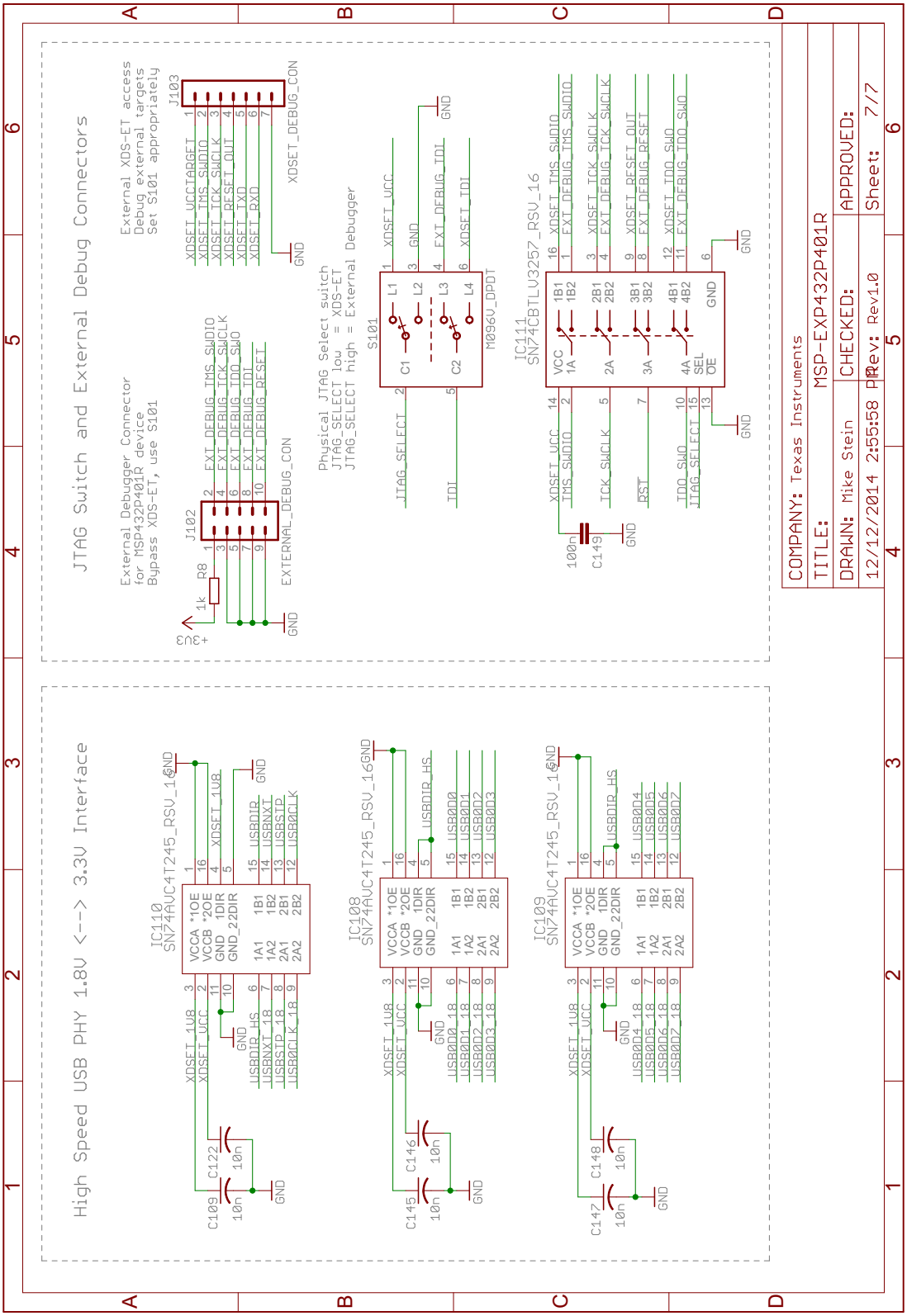


**Figure 29. Schematics (7 of 7)**