# Dual Processor Platform 2

# User Guide

Revised: September 2017

ETH Zurich
Computer Engineering Group, TIK
www.tec.ethz.ch

## Table of Contents

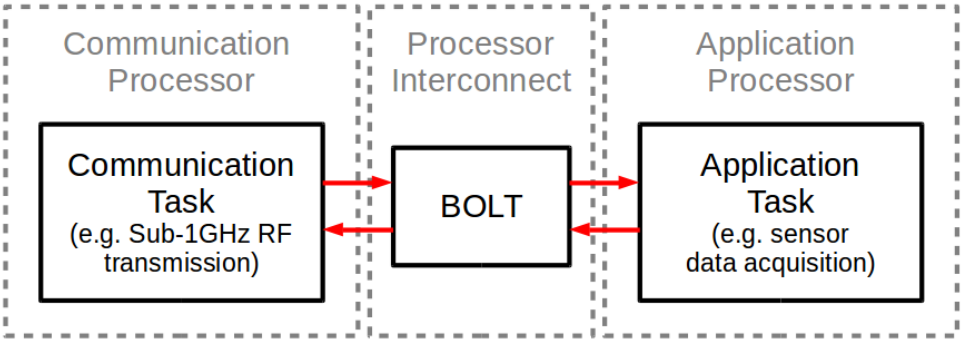
1 Introduction....2
2 DPP Architecture....3
3 Dual Processor Platform 2....4
3.1 Board-to-board connector....5
Pin layout....5
Mandatory pins....6
Pin levels....6
Programming and debugging interface....6
Bolt serial interface....6
3.2 Communication Board....8
Pin Mapping....8
Target programming and debugging....9
3.3 Dev Board....10
Overview....10
Power Supply....11
$V_{CC}$ Select....12
Serial Output....12
Debug Breakout....13
Reset Switch....13
Extension Header....13
$I^2C$ Header....14
MSP432 Pin Mapping....14
Power Measurements....16
3.4 Sample Code....17
CC430 Test Application....17
MSP432 Test Application....17
BOLT program....19
3.5 FAQ: Frequently asked questions....20
Which development environment (IDE) should I use to develop software for the DPP?....20
How can I program the targets?....20
Can the targets be programmed via USB (Bootstrap Loader)?....20
Can I use TI's EnergyTrace feature to measure the energy consumption?....21
Can I build my own DPP?....21
List of Abbreviations....21

# 1 Introduction

The Dual Processor Platform (DPP) is a novel architecture for embedded systems, primarily developed for wireless sensor nodes. It tries to mitigate resource interference by mapping different tasks onto separate processors. In the world of wireless sensor nodes, the two main tasks are typically sensing/actuation and communication (sending/receiving). We therefore map those two task onto an *application processor* and a *communication processor*. BOLT, a stateful processor interconnect, is leveraged to enable asynchronous message passing between the two tasks while strictly decoupling power, clock and time domains of the two processing elements.

# 2 DPP Architecture

A brief overview of the Dual Processor Platform architecture:




Each DPP consists of 3 components:

- an Application processor (APP)
- a Communication processor (COM)
- and the BOLT processor interconnect.

The two fundamentally different tasks – sending/actuation and communication – are mapped onto two different, physically separated processing elements (typically low-power microcontrollers). While the communication processor handles wireless packet transmission and reception, the application processor takes care of the sensor data acquisition.

The most notable advantages of this approach are:

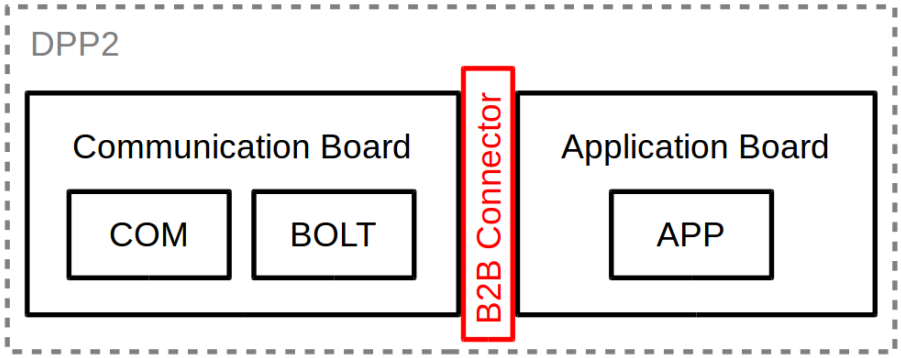
- modularity: exchanging one of the components is hassle free and does not require a change to the other components
- simpler software: by decoupling the two tasks and thus removing resource interference and time dependencies, the complexity for software development and verification is significantly reduced
- parallel development: since the two components are separate entities and only interact with each other via BOLT, they can be developed and maintained independently
- independent power management: each core can decide on its own when to enter or exit a low-power mode

To learn more about the DPP and BOLT, please refer to the following publications:

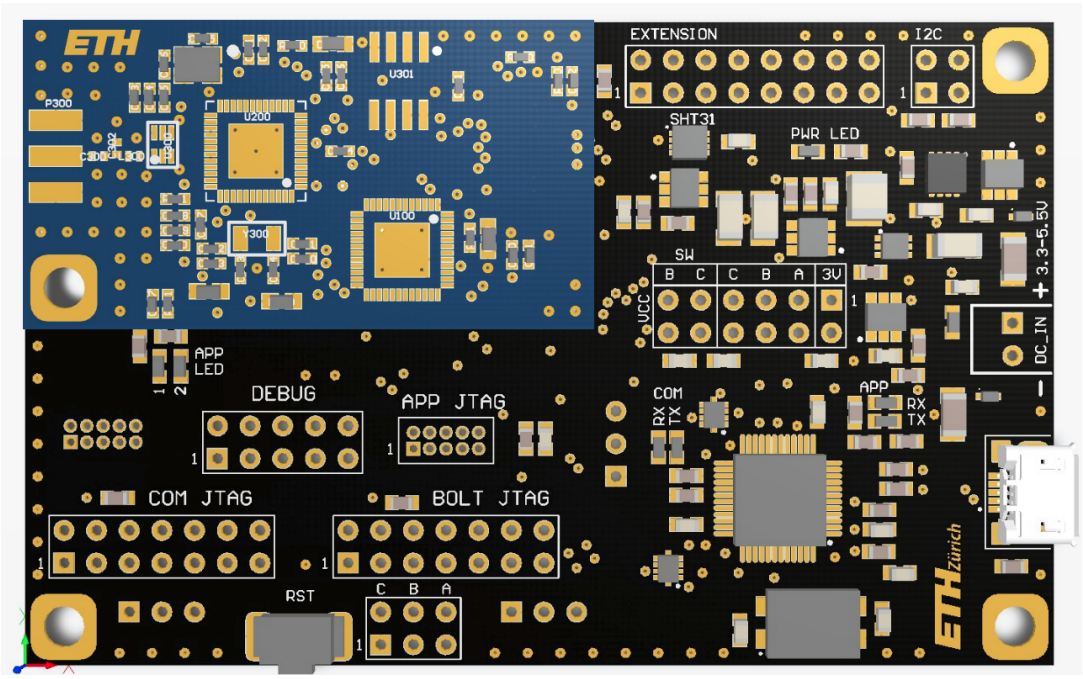
- Bolt: A Stateful Processor Interconnect. F. Sutton, M. Zimmerling, R. Da Forno, R. Lim, T. Gsell, G. Giannopoulou, F. Ferrari, J. Beutel, L. Thiele. 13th ACM Conference on Embedded Networked Sensor Systems (SenSys), November 2015
- The Design of a Responsive and Energy-efficient Event-triggered Wireless Sensing System. Felix Sutton, Reto Da Forno, David Gschwend, Tonio Gsell, Roman Lim, Jan Beutel, and Lothar Thiele. 14th International Conference on Embedded Wireless Systems and Networks (EWSN), February 2017

# 3 Dual Processor Platform 2

The Dual Processor Platform 2 is the 2nd iteration of our DPP hardware. In this new design, communication and application processor reside on a separate printed circuit board (PCB). It is a stacked design, i.e. the PCB with the application board acts as a motherboard for the smaller communication board. BOLT resides on the communication PCB.




The goal of this design is to keep the communication board as small and simple as possible by shifting the complexity and all I/O pins to the application board. This helps to keep the cost of the communication board down and reduces the design effort, thus enabling rapid prototyping for new radio chips and communication protocols.



## 3.1 Board-to-board connector

The main interface interconnecting the two boards has been carefully designed to cover a wide range of use-cases. The chosen connector and pin layout can be seen as the standard interface between the two boards and in general as the interface to communicate with BOLT. It is intended to be used for all future communication and application boards.

The connector is a 2-row 26-pin high density header with a pitch size of 0.8mm and a mated height of 4.5mm. Part numbers: Molex 52465-2671 (male) and 53307-2671 (female).
The male connector is part of the communication board.

### Pin layout

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 |
| 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 21 | 23 | 25 |

| *Pin#* | *Designator* | *Type*[1] | *Description* |
|---|---|---|---|
| 1 | COM_VCC | - | Power supply (2.2 – 3.3V, 100mA max.) |
| 2 | BOLT_VCC | - | Power supply (2.2 – 3.3V, 50mA max.) |
| 3 | BOLT_RXD | Output | UART RXD (serial input) / BSL data in |
| 4 | BOLT_RST | Output | Reset (active low) / nDTR / SBWTDIO |
| 5 | BOLT_TXD | Input | UART TXD (serial output) / BSL data out |
| 6 | BOLT_PROG | Output | TEST / nRTS / BSL entry / SBWTCK |
| 7 | BOLT_GPIO1 | I/O | General purpose I/O pin |
| 8 | BOLT_GPIO2 | I/O | General purpose I/O pin |
| 9 | APP_ACK | Input | Bolt acknowledgement line |
| 10 | APP_MISO | Input | Bolt SPI data input (master input, slave output) |
| 11 | APP_REQ | Output | Bolt request line |
| 12 | APP_MOSI | Output | Bolt SPI data output (master output, slave input) |
| 13 | APP_MODE | Output | Bolt mode select ($\overline{R}$/W) |
| 14 | APP_SCK | Output | Bolt SPI serial clock (max. 4MHz) |
| 15 | APP_IND | Input | Bolt indication line (for the Bolt queue towards APP) |
| 16 | COM_TREQ | Output | Timestamp request trigger |
| 17 | COM_IND | Input | Bolt indication line (for the Bolt queue towards APP) |
| 18 | COM_GPIO1 | I/O | General purpose I/O pin |
| 19 | COM_GPIO2 | I/O | General purpose I/O pin |
| 20 | COM_PROG2 | I/O | SWDIO / cJTAG_TMSC / General purpose I/O pin |
| 21 | COM_TXD | Input | UART TXD (serial output) / BSL data out |
| 22 | COM_PROG | Output | TEST / nRTS / BSL entry / SWDCLK / cJTAG_TCKC |
| 23 | COM_RXD | Output | UART RXD (serial input) / BSL data in |
| 24 | COM_RST | Output | Reset (active low) / nDTR / SBWTDIO |
| 25 | GND | - | Ground |
| 26 | GND | - | Ground |

1 From the perspective of the application processor.

## Mandatory pins

Only a subset of the pins specified above must be connected to the application processor, all other pins may be left floating (unconnected). The mandatory pins are:

- all 7 Bolt communication pins (APP_MISO, APP_MOSI, APP_SCK, APP_ACK, APP_REQ, APP_MODE, APP_IND)
- 2 power supply lines (BOLT_VCC and COM_VCC) and GND

## Pin levels

The signal level on all I/Os must not exceed the Vcc level. The application processor is allowed to completely turn off – i.e. set to ground level – one or both of the Vcc lines at any time. If both Vcc lines are supplied, they must be on the same voltage level.

Valid pin configurations are *input* (high impedance), *output low* and *output high*. Only pins specified as output may be driven low or high.

## Programming and debugging interface

The connector supports 3 different programming/debugging interfaces, whereof only one will be used (depending on the chosen communication processor):

- SBW (Spy-Bi-Wire) is the programming interface for TI's MSP430/CC430 microcontrollers
- cJTAG (compact 2-wire JTAG) is for TI's ARM SoCs
- SWD (serial wire debug) is for all other ARM MCUs

In addition, the bootloader (BSL) can be used to program/download the firmware to the microcontroller via a serial communication port (RS232).

## Bolt serial interface

All Bolt lines are active high. The SPI interface must be configured with polarity = 0 and phase = 0. The SPI master (APP or COM) is free to choose any serial clock speed up to 4 MHz. The steps to access the Bolt interface include:

- set the BOLT MODE line to read (low) or write (high)
- set the request line (BOLT REQ) high and wait for the acknowledgement line to rise (BOLT ACK)
- start the data transfer over the SPI interface (for a read operation, continue to read until the ACK line drops)
- set the REQ line low

If a write operation is requested and the ACK line does not rise upon request, the queue is full an no further data can be written. If a read operation is requested and the ACK line does not rise upon request, the queue is empty (there is no data to read). Note that during a read operation, the ACK line may drop before the last byte has been transmitted. Depending on the selected clock speed, this could encourage the SPI master to abort the transfer too early and loose one byte. On the opposite,

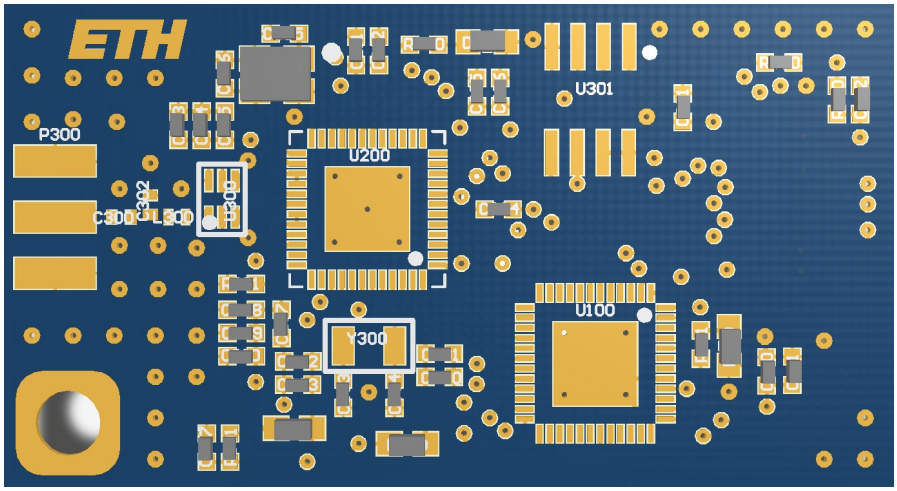
if a high clock speed is chosen, too many bytes may be received. It is therefore necessary to include a length field in each message on the application layer.

## 3.2 Communication Board

Our DPP communication board consists of a TI CC430 SoC and BOLT.

Specs:

- TI CC430F5147, 32k ROM, 4k SRAM with Sub-1-GHz RF Transceiver Core (CC1101)
- BOLT, implemented on a TI MSP430FR5969 with 64k FRAM
- Cypress FM25V10 1MBit FRAM chip (optional)
- 32 kHz crystal oscillator (Epson FC135)
- 26 MHz crystal oscillator (Epson TSX3225)
- PCB dimensions: 44 x 24mm




### Pin Mapping

The pin mapping on the CC430 is listed in the table below.

| *Pin* | *Designator* | *Signal type*[2] | *Recomm. config*[3] | *Function mapping (PxSEL.y)* | *Description* |
|---|---|---|---|---|---|
| P1.0 | - | X | OL | - | Unused (NC) |
| P1.1 | APP_IND | I | Z | - | BOLT indication line (output queue, APP side) |
| P1.2 | COM_MISO | X | Z | UCB0SOMI (1) | BOLT SPI master input, slave output |
| P1.3 | COM_MOSI | X | Z | UCB0SIMO (1) | BOLT SPI master output, slave input |
| P1.4 | COM_SCK | X | Z | UCB0CLK (1) | BOLT SPI serial clock |
| P1.5 | COM_RXD | X | Z | UCA0RXD (1) | UART / BSL receive |
| P1.6 | COM_TXD | X | Z | UCA0TXD (1) | UART / BSL transmit |
| P1.7 | - | X | OL | - | Unused (NC) |
| P2.0 | COM_FRAM_EN | O | OL | - | FRAM enable pin (high = enable) |
| P2.1 | COM_TREQ | I | Z | - | Timestamp request line |
| P2.2 | COM_IND | I | Z | - | BOLT indication line (high = data available) |

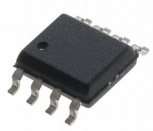
2 I = digital input, O = digital output, A = analog, X = don't care
3 Z = high impedance (input), OL = output low, OH = output high, IL = input with pulldown, IH = input with pullup

| | | | | | |
|---|---|---|---|---|---|
| P2.3 | COM_MODE | O | OL | - | BOLT mode select line (low = read, high = write) |
| P2.4 | COM_REQ | O | OL | - | BOLT request line |
| P2.5 | COM_ACK | I | Z | - | BOLT acknowledgement line |
| P2.6 | COM_FUT_USE | I/O | OL | - | Reserved for future use, connected to BOLT |
| P2.7 | - | X | OL | - | Unused (NC) |
| P3.0 | COM_LED | O | OL | - | Debug LED (output high = on, low = off) |
| P3.1 | COM_GPIO1 | I/O | OL | - | Debug pin |
| P3.2 | COM_GPIO2 | I/O | OL | - | Debug pin |
| P3.3 | COM_PROG2 | I/O | OL | - | Debug pin |
| P3.4 | - | X | OL | - | Unused (NC) |
| P3.5 | - | X | OL | - | Unused (NC) |
| P3.6 | - | X | OL | - | Unused (NC) |
| P3.7 | - | X | OL | - | Unused (NC) |
| P5.0 | XIN | X | Z | Crystal mode (1) | 32kHz crystal oscillator |
| P5.1 | XOUT | X | Z | Crystal mode (1) | 32kHz crystal oscillator |
| PJ.0 | TDO | X | 0L | - | Unused (NC) |
| PJ.1 | TDI | X | 0L | - | Unused (NC) |
| PJ.2 | TMS | X | 0L | - | Unused (NC) |
| PJ.3 | TCK | X | 0L | - | Unused (NC) |

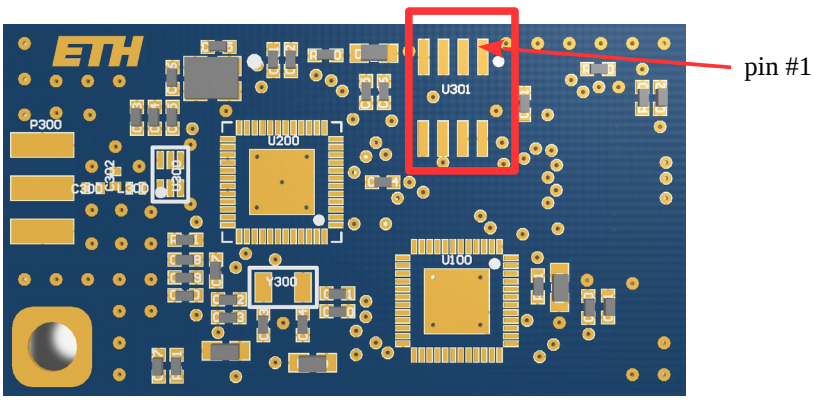
## Target programming and debugging

The CC430 can be accessed by Spy-Bi-Wire (SBW) only. SBW is a 2-wire version of the TI JTAG interface with the same capabilities.

## Installing the optional FRAM chip



To increase the amount of memory available to the CC430 SoC, an FRAM chip can be installed on the PCB. FRAM is highly reliable, non-volatile memory with fast access times. The Cypress chip supports an SPI clock speed of up to 40 MHz and claims an endurance of several trillion write cycles and over 100 years of data retention. The power dissipation is ~300uA when active (at 1 MHz) and less than 5uA in sleep mode.



## 3.3 Dev Board

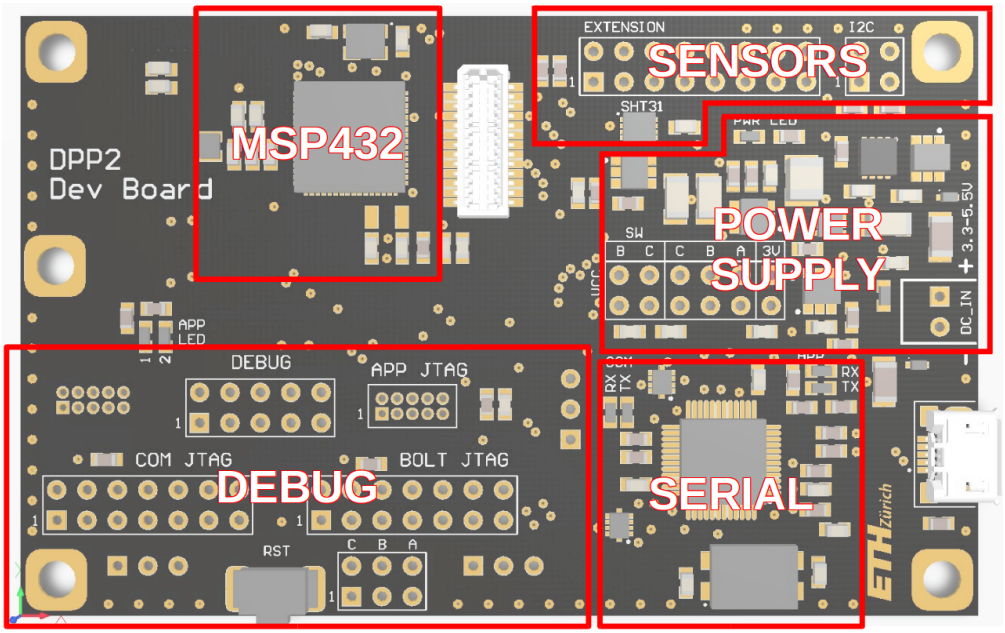
Our DPP Dev Board contains an application processor (APP) and acts as a motherboard for the B-C board at the same time. The APP is a TI MSP432 Arm Cortex M4F microcontroller with a max. clock speed of 48MHz, 256k ROM and 64k SRAM.

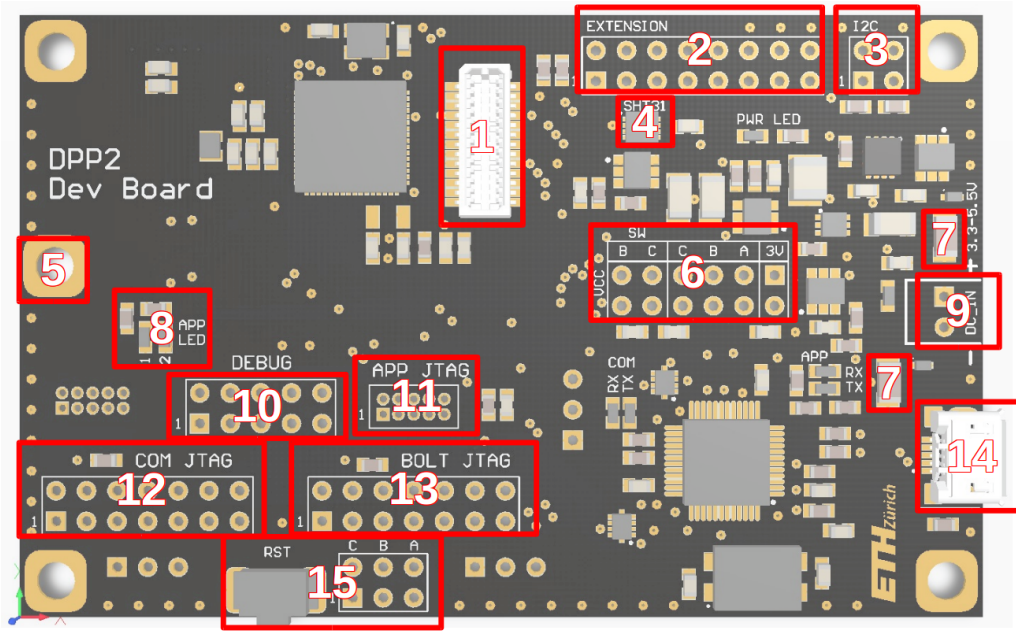
As its name suggests, the Dev Board is a generic application board to develop, debug and test software for the DPP.

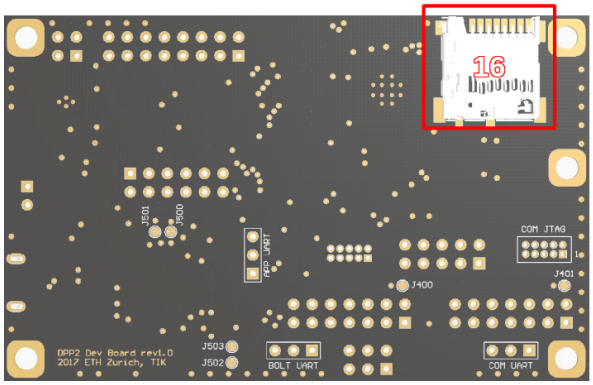
### Overview

The Dev Board can be roughly divided into 5 main sections:

- MSP432 (APP)
- sensors + extension
- power supply
- debug
- serial (FTDI chip, acts as a serial forwarder between APP / COM and the USB)



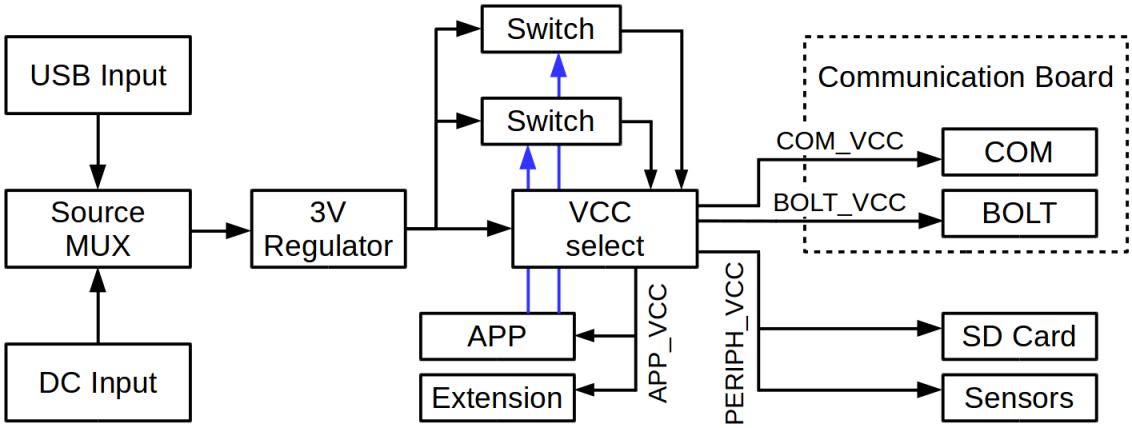



| | |
|---|---|
| 1 | Board-to-board connector |
| 2 | 16-pin Extension header |
| 3 | 4-pin I²C header |
| 4 | Temperature + humidity sensor |
| 5 | Mounting hole for the communication board |
| 6 | 12-pin VCC select header |
| 7 | 500mA resetting fuse |
| 8 | APP debug LEDs |
| 9 | DC power supply input |
| 10 | 10-pin Debug breakout |
| 11 | 10-pin APP JTAG header (1.27mm pitch) |
| 12 | 14-pin COM JTAG header |
| 13 | 14-pin Bolt JTAG header |
| 14 | Micro USB connector |
| 15 | Reset switch and selector |
| 16 | MicroSD Card slot |

## Power Supply

The Dev Board can be powered via USB or an external DC power supply (DC input). Both inputs are fused and ESD protected. In addition, the DC input has reverse current protection to avoid damage in case the terminals are accidentally swapped. The voltage range for the DC input is 3.3 - 5.5V.
It is safe to leave the USB connected for serial logging while an external DC power supply is used. In this case, the source multiplexer will automatically select the DC input.

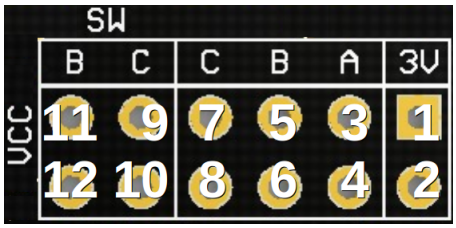



Application processor (APP) can control the power switch for the peripherals (PERIPH_VCC) and the communication board (blue arrows).

## $V_{CC}$ Select

The $V_{cc}$ Select header offers several options for power measurements and the possibility to select a switchable power supply for COM and BOLT. In the default configuration, jumpers are set across pins 1-2, 3-4, 5-6 and 7-8. In order to be able to turn off the power supply for COM and BOLT from APP, jumpers may be set across 9-10 and 11-12 instead of 5-6 and 7-8.
Note that pins 2, 3, 5 and 7 are internally connected.

| *Pin* | *Description* |
|---|---|
| 1 | Regulator output, 3V |
| 2 | Jumpered regulator output, 3V |
| 3 | Jumpered regulator output, 3V |
| 4 | Supply for APP (APP_VCC) |
| 5 | Jumpered regulator output, 3V |
| 6 | Supply for Bolt (BOLT_VCC) |
| 7 | Jumpered regulator output, 3V |
| 8 | Supply for COM (COM_VCC) |
| 9 | Switchable supply, 3V |
| 10 | Supply for COM (COM_VCC) |
| 11 | Switchable supply, 3V |
| 12 | Supply for Bolt (BOLT_VCC) |




## Serial Output

A dual USB UART IC from FTDI is utilized to convert and forward the serial output from the application and communication processor to USB. Once the Dev Board is connected to a computer, two serial ports will show up. If for some reason this method doesn't work for you, you can also use the dedicated UART headers on the Dev Board to connect a serial cable. The TXD/RXD pins are also available on the 14-pin TI JTAG connectors (pin 12 = TXD, pin 14 = RXD).

As a last option, one could also use the UART backchannel of the MSP-FET debugger to access the serial output of the communication processor and BOLT.

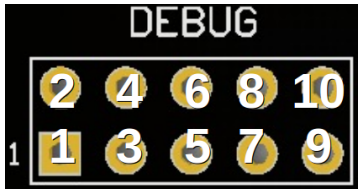
Note that the FTDI chip is only powered when a USB cable is plugged in.

| *Pin* | *Description* |
|---|---|
| 1 | Ground |
| 2 | UART TXD |
| 3 | UART RXD |


## Debug Breakout

The debug breakout provides easy access to a couple of debug GPIO pins for each processor. Note that COM_PROG2 may be used as a GPIO pin, but doubles up as a programming pin.

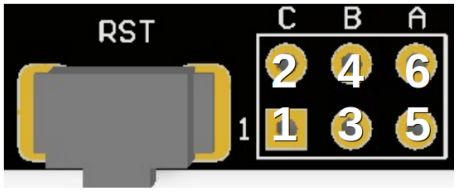
| *Pin* | *Designator* |
|---|---|
| 1 | APP_VCC |
| 2 | COM_VCC |
| 3 | APP_GPIO1 |
| 4 | APP_GPIO2 |
| 5 | BOLT_GPIO1 |
| 6 | BOLT_GPIO2 |
| 7 | COM_GPIO1 |
| 8 | COM_GPIO2 |
| 9 | COM_PROG2 |
| 10 | GND |




## Reset Switch

There is one reset switch on the Dev Board, which can be configured to reset one, two or all three cores. As an example: Set a jumper across 1-2 to connect COM to the reset switch.

| *Pin* | *Designator* | *Description* |
|---|---|---|
| 1 | COM_RST | COM reset line |
| 2 | RST | Reset switch |
| 3 | BOLT_RST | Bolt reset line |
| 4 | RST | Reset switch |
| 5 | APP_RST | APP reset line |
| 6 | RST | Reset switch |

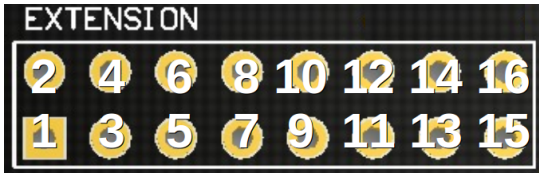



## Extension Header

The extension header adds a lot of flexibility to the development board. Basically any kind of circuit can be interfaced with the application processor through this header. All of the 12 data lines can be configured as digital pins, whereof 8 are fully mappable digital I/Os and 4 are also configurable as analog pins. As a consequence, we have up to 4 single-ended ADC channels and up

to 3 digital interfaces (UART, I²C, SPI) available.
Note that the current drain through the $V_{cc}$ line (pin #1) should not exceed 50mA.

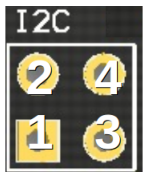
| *Pin* | *Designator* | *Description* |
|---|---|---|
| 1 | APP_VCC | 3V |
| 2 | APP_EXT13 | GPIO |
| 3 | APP_EXT0 | GPIO, mappable |
| 4 | APP_EXT12 | GPIO |
| 5 | APP_EXT1 | GPIO, mappable |
| 6 | APP_EXT11 | GPIO |
| 7 | APP_EXT2 | GPIO, mappable |
| 8 | APP_EXT10 | GPIO |
| 9 | APP_EXT3 | GPIO, mappable |
| 10 | APP_EXT9 | GPIO |
| 11 | APP_EXT4 | GPIO, mappable |
| 12 | APP_EXT8 | GPIO |
| 13 | APP_EXT5 | GPIO, mappable |
| 14 | APP_EXT7 | GPIO, mappable |
| 15 | APP_EXT6 | GPIO, mappable |
| 16 | GND | Ground |




## I²C Header

The I²C header can be used to attach additional sensors. Note that the signal lines are shared with the Sensirion SHT30 sensor and have a 10k pullup resistor. To avoid conflicts, all attached sensors must have an address other than 0x44.
The current drain through the $V_{cc}$ line (pin #1) should not exceed 50mA.

| *Pin* | *Designator* | *Description* |
|---|---|---|
| 1 | PERIPH_VCC | 3V, switchable |
| 2 | APP_SENSOR_SDA | I²C data |
| 3 | APP_SENSOR_SCL | I²C clock |
| 4 | GND | Ground |




## MSP432 Pin Mapping

The pin mapping on the MSP432 is listed in the table below.

| *Pin* | *Designator* | *Signal type*[4] | *Recomm. config*[5] | *Function mapping (PxSEL0.y, PxSEL1.y)* | *Description* |
|---|---|---|---|---|---|
| P1.0 | APP_BSL | X | OL | - | Bootstrap loader entry pin |
| P1.1 | - | X | OL | - | Unused (NC) |
| P1.2 | APP_RXD | X | Z | UCA0RXD (1, 0) | Debug/BSL UART receive |

4 I = digital input, O = digital output, A = analog, X = don't care
5 Z = high impedance (input), OL = output low, OH = output high, IL = input with pulldown, IH = input with pullup

| | | | | | |
|---|---|---|---|---|---|
| P1.3 | APP_TXD | X | Z | UCA0TXD (1, 0) | Debug/BSL UART transmit |
| P1.4 | - | X | OL | - | Unused (NC) |
| P1.5 | APP_SCK | X | Z | UCB0CLK (1, 0) | BOLT SPI serial clock |
| P1.6 | APP_MOSI | X | Z | UCB0SIMO (1, 0) | BOLT SPI master output, slave input |
| P1.7 | APP_MISO | X | Z | UCB0SOMI (1, 0) | BOLT SPI master input, slave output |
| P2.0 | APP_LED1 | O | OL | - | Debug LED (low = off, high = on) |
| P2.1 | APP_LED2 | O | OL | - | Debug LED (low = off, high = on) |
| P2.2 | APP_GPIO1 | O | OL | - | Debug pin |
| P2.3 | APP_GPIO2 | O | OL | - | Debug pin |
| P3.0 | APP_EXT0 | I/O | OL | - | Extension header pin 3, digital (mappable) |
| P3.1 | APP_EXT1 | I/O | OL | - | Extension header pin 5, digital (mappable) |
| P3.2 | APP_EXT2 | I/O | OL | - | Extension header pin 7, digital (mappable) |
| P3.3 | APP_EXT3 | I/O | OL | - | Extension header pin 9, digital (mappable) |
| P3.4 | APP_EXT4 | I/O | OL | - | Extension header pin 11, digital (mappable) |
| P3.5 | APP_EXT5 | I/O | OL | - | Extension header pin 13, digital (mappable) |
| P3.6 | APP_EXT6 | I/O | OL | - | Extension header pin 15, digital (mappable) |
| P3.7 | APP_EXT7 | I/O | OL | - | Extension header pin 14, digital (mappable) |
| P4.2 | COM_TREQ | O | OL | | COM time request pin |
| P4.3 | COM_IND | I | Z | | BOLT indication line (output queue, COM side) |
| P4.4 | APP_IND | I | Z | | BOLT indication line |
| P4.5 | APP_MODE | O | OL | | BOLT mode select line |
| P4.6 | APP_REQ | O | OL | | BOLT request line |
| P4.7 | APP_ACK | I | Z | | BOLT acknowledgement line |
| P5.0 | APP_EXT13 | I/O/A | OL | - | Extension header pin 2, analog or digital |
| P5.1 | APP_EXT12 | I/O/A | OL | - | Extension header pin 4, analog or digital |
| P5.2 | APP_EXT11 | I/O/A | OL | - | Extension header pin 6, analog or digital |
| P5.3 | APP_EXT10 | I/O/A | OL | - | Extension header pin 8, analog or digital |
| P5.4 | - | X | OL | - | Unused (NC) |
| P5.5 | APP_VBAT | A | Z | *Analog (1, 1)* | Supply voltage sense (voltage divider) |
| P5.6 | APP_EXT9 | I/O/A | OL | - | Extension header pin 10, analog or digital |
| P5.7 | APP_EXT8 | I/O/A | OL | - | Extension header pin 12, analog or digital |
| P6.6 | APP_SENSOR_SDA | X | Z | UCB3SDA (0, 1) | $I^2C$ signal data |
| P6.7 | APP_SENSOR_SCL | X | Z | UCB3SCL (0, 1) | $I^2C$ signal clock |
| P7.0 | APP_SD0 | X | Z | UCA1SOMI (1, 0) | SD Card SPI MISO |
| P7.1 | APP_SD1 | X | Z | UCA1CLK (1, 0) | SD Card SPI serial clock |
| P7.2 | APP_SD2 | X | Z | UCA1SIMO (1, 0) | SD Card SPI MOSI |
| P7.3 | APP_SD3 | X | OL | - | SD Card SPI slave enable |
| P8.0 | APP_COM_$\overline{EN}$ | I/O | Z | - | Power switch (COM + BOLT enable) |
| P8.1 | APP_PERIPH_$\overline{EN}$ | I/O | Z | - | Power switch (peripherals enable) |
| PJ.0 | LFXIN | X | Z | Crystal mode (1, 0) | Low-frequency crystal (32kHz) |
| PJ.1 | LFXOUT | X | Z | Crystal mode (1, 0) | Low-frequency crystal (32kHz) |

| | | | | | |
|---|---|---|---|---|---|
| PJ.2 | HFXOUT | X | Z | Crystal mode (1, 0) | High-frequency crystal (48MHz) |
| PJ.3 | HFXIN | X | Z | Crystal mode (1, 0) | High-frequency crystal (48MHz) |
| PJ.4 | APP_TDI | X | Z | - | JTAG TDI |
| PJ.5 | APP_TDOSWO | X | Z | - | JTAG TDO |

Note: The secondary digital functions on Ports P2, P3, and P7 are fully mappable (see p.125f in the datasheet).

## Power Measurements

The current drain of each sub-circuit can be measured at the $V_{CC}$ select header. To measure the consumption of the whole board, simply use a source meter at the DC input. Note that this also includes the whole power supply circuit and the power LED, which alone drains several milliamps. Alternatively, one can perform high-precision measurements at the output of the LDO (measure across pins 1-2 of $V_{CC}$ select).

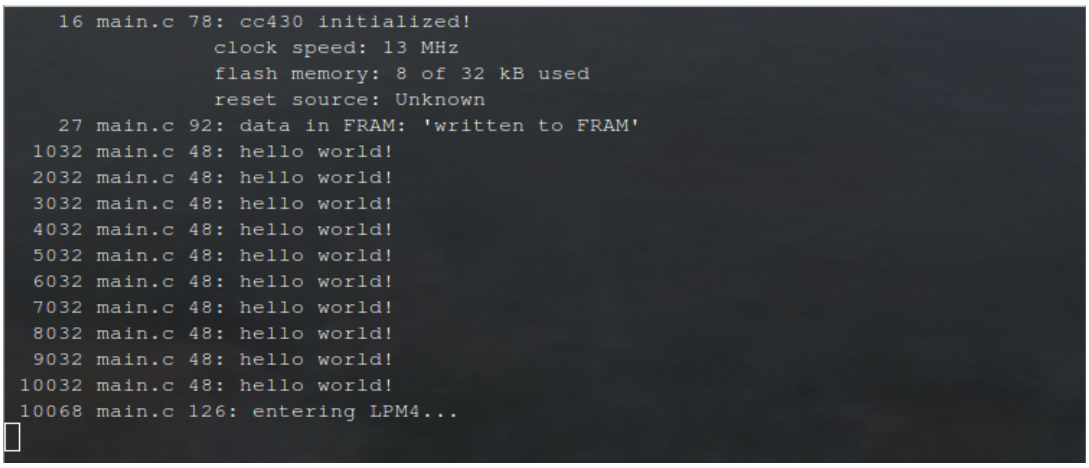
Note that the Dev Board was not designed to perform low-side measurements.

## 3.4 Sample Code

The sample code is compatible with TI Code Composer Studio (CCS) 7.2.

### CC430 Test Application

The sample code for the CC430 shows how to configure the core and communicate via BOLT. It is basically a simple ‘blink LED’ test application and does not include the radio core. If you would like to try out the radio with Glossy or LWB, check out our repository on Github for sample code.




After a few blinks, the MCU goes into deepsleep mode (LPM4). Please read through the comments in the code for more information about the program flow.

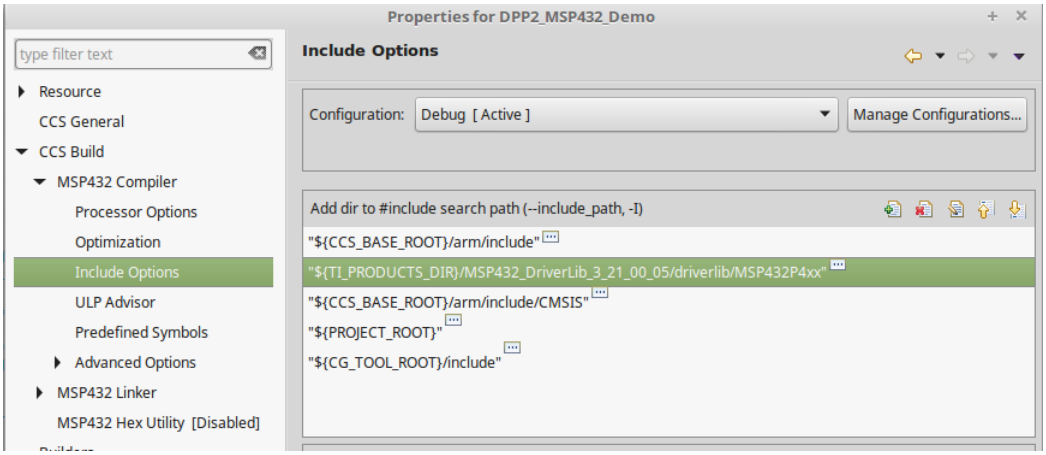
In order to compile the code, you will need the MSP430 driver library (copy the subfolder ‘driverlib’ from the zip file into the CC430 demo code folder).

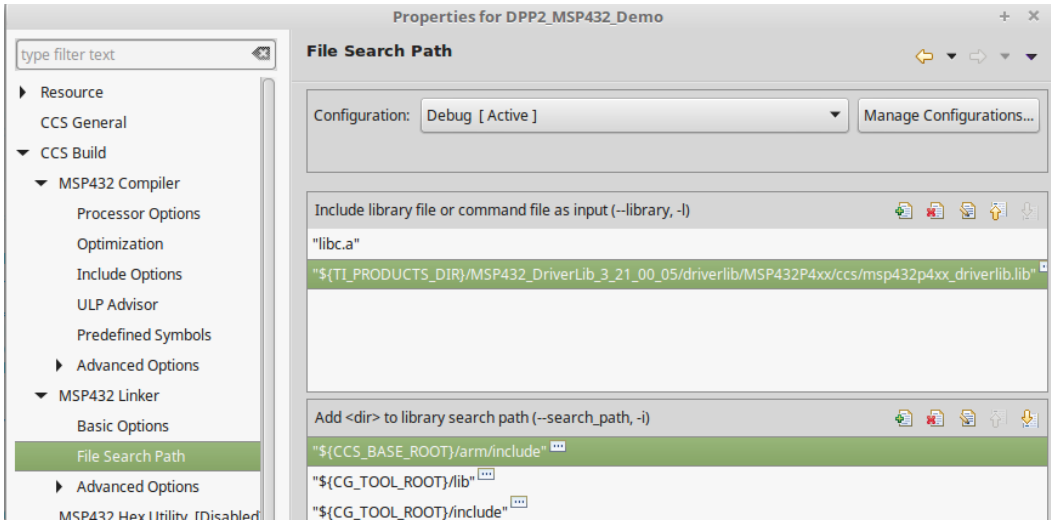
### MSP432 Test Application

The sample code for the MSP432 shows how to configure the core and communicate via BOLT. In the main loop, the sensor values (internal temperature, input voltage and Sensirion SHT30) are sampled and printed out via UART. After a few loop passes, the MCU goes into deepsleep mode (LPM4.5). Please read through the comments in the code for more information about the program flow.

```
   12 main.c 81: msp432 initialized!
                 device ID: 0xa005 (rev 0x44)
                 clock speed: 48 MHz
                 flash memory: 18 of 128 kB used, 2 wait states
                 reset source: LPMx5
   29 main.c 115: RTC 2017-07-01 10:00:00
   33 main.c 244: voltage monitor: 11427 (3486mV)
   38 main.c 246: internal temperature sensor: 4775 (22°C)
   46 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.3%
 1052 main.c 115: RTC 2017-07-01 10:00:01
 1056 main.c 244: voltage monitor: 11448 (3492mV)
 1061 main.c 246: internal temperature sensor: 4782 (23°C)
 1069 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.3%
 2075 main.c 115: RTC 2017-07-01 10:00:02
 2080 main.c 244: voltage monitor: 11415 (3482mV)
 2085 main.c 246: internal temperature sensor: 4778 (23°C)
 2093 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.3%
 3099 main.c 115: RTC 2017-07-01 10:00:03
 3103 main.c 244: voltage monitor: 11446 (3492mV)
 3108 main.c 246: internal temperature sensor: 4780 (23°C)
 3116 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.3%
 4122 main.c 115: RTC 2017-07-01 10:00:04
 4127 main.c 244: voltage monitor: 11440 (3490mV)
 4131 main.c 246: internal temperature sensor: 4772 (22°C)
 4140 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.2%
 5146 main.c 115: RTC 2017-07-01 10:00:05
 5150 main.c 244: voltage monitor: 11443 (3492mV)
 5155 main.c 246: internal temperature sensor: 4776 (22°C)
 5163 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.3%
 6169 main.c 115: RTC 2017-07-01 10:00:06
 6174 main.c 244: voltage monitor: 11412 (3482mV)
 6178 main.c 246: internal temperature sensor: 4775 (22°C)
 6186 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.2%
 7193 main.c 115: RTC 2017-07-01 10:00:07
 7197 main.c 244: voltage monitor: 11431 (3488mV)
 7202 main.c 246: internal temperature sensor: 4772 (22°C)
 7210 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.2%
 8216 main.c 115: RTC 2017-07-01 10:00:08
 8220 main.c 244: voltage monitor: 11403 (3480mV)
 8225 main.c 246: internal temperature sensor: 4777 (22°C)
 8233 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.2%
 9239 main.c 115: RTC 2017-07-01 10:00:09
 9244 main.c 244: voltage monitor: 11435 (3488mV)
 9249 main.c 246: internal temperature sensor: 4777 (22°C)
 9257 main.c 132: SHT31 temperature: 25.9°C, humidity: 45.2%
10263 main.c 158: entering LPM...
```

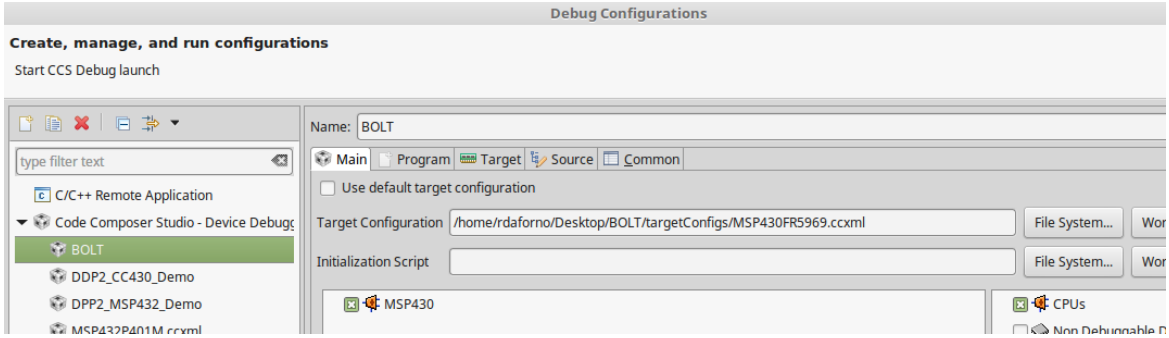
In order to compile the code, you will need to download the [MSP432 driver library]. Extract the zip file into your TI installation folder and create a link to the library (header files and compiled lib file) in your project settings:






## BOLT program

You can flash the provided Intel hex file with CCS or any other software capable to access an MSP430 core. To flash it with CCS, go to ‘Run’ > ‘Debug Configurations’ and create a new configuration. Click ‘File System’ next to the blank field ‘Target Configuration’ to select the file MSP430FR5969.ccxml in the folder BOLT of the demo code. In the next tab ‘Program’, select ‘File System’ next to the field ‘Program’ to browse for the BOLT hex file. As a last step, go to the tab ‘Target’ > ‘Auto Run and Launch Options’ and make sure ‘Connect to the target on debugger startup’ is enabled. Click ‘Apply’ to save the settings and ‘Debug’ to flash the program onto the target.




Note: There are 4 precompiled BOLT images for different message sizes (64, 128, 256 bytes) and with different startup behavior (with or without queue reset).

| *Filename* | *Max. message size* | *Max. # messages per queue* | *Queue reset at startup* |
|---|---|---|---|
| BOLT_64b_reset.hex | 64 bytes | 292 | yes |
| BOLT_128b_reset.hex | 128 bytes | 148 | yes |
| BOLT_256b_reset.hex | 256 bytes | 74 | yes |
| BOLT_64b_noreset.hex | 64 bytes | 292 | no |
| BOLT_128b_noreset.hex | 128 bytes | 148 | no |
| BOLT_256b_noreset.hex | 256 bytes | 74 | no |

## 3.5 FAQ: Frequently asked questions

### Which development environment (IDE) should I use to develop software for the DPP?

Since all 3 MCUs on the DPP are from Texas Instruments, TI Code Composer Studio (CCS) is the natural choice. CCS is available free of charge and runs on Windows, MacOS and Linux.

### How can I program the targets?

Use the JTAG connectors on the Dev Board to program COM, BOLT and APP. Each connector on the board is labeled and pin#1 is always marked. To flash the programs, we recommend to use the TI MSP-FET debugger for BOLT and COM. For the APP MCU, you can either use TI's adapter for the MSP-FET (part no. MSP-FET-432ADPTR) or any other compatible debugger such as the Blackhawk XDS100v2.

### Can the targets be programmed via USB (Bootstrap Loader)?

Yes, the program can be downloaded onto the targets (COM/APP) directly via a USB cable without any additional hardware. Prerequisites:

- The solder jumpers J500, J501, J502 and J503 on the backside of the Dev Board need to be closed. This will directly connect the DTR and RTS lines of the FTDI chip to the reset and TEST (BSL entry) pin of the CC430 and MSP432 MCU.
- In order to use the provided scripts to access the BSL, you need to have Python 2.7 and pyserial installed on your computer.
- On the MSP432, the BSL needs to be unlocked with a debugger first. Open the project `BSL/MSP432_BSL_Unlock` and flash it onto the MCU. If the process was successful, the LED should blinkt continuously. The BSL is now permanently unlocked.

How to use the BSL scripts on a Linux machine (will also work on a Windows or Mac computer with some adjustments):

- First, you need to install the MSP430 Python modules. To do so, open the folder `BSL/python-msp430-tools-custom` and run the following command:
  `sudo python setup.py install`
  Note that this software is a modified version of the python-msp430-tools 0.8.1.
- To flash a program onto an MCU, use the provided scripts `flash_cc430.sh` and `flash_msp432.sh`.

Remark: Depending on your operating system, the RTS line may be set high after the serial port is closed, which means the TEST line of the MCU will be pulled high. As a consequence, the MCU can no longer be reset with the reset switch. Use the provided script `reset.py` to reset the MCU.

## Can I use TI's EnergyTrace feature to measure the energy consumption?


Yes, the EnergyTrace feature can be used for BOLT and COM with the MSP-FET. To do so, desolder the 0 Ohm resistor next to the JTAG connector and close the solder jumper J400 (BOLT) or J401 (COM). Also, make sure to remove the jumper for the respective MCU on the Vcc header.

## Can I build my own DPP?

Yes, of course. If you are interested in building your own DPP, we are happy to provide you with schematics for Bolt.

## List of Abbreviations

| | |
|---|---|
| DPP | Dual Processor Platform |
| APP | application processor |
| COM | communication processor |
| MCU | microcontroller unit |
| CCS | Code Composer Studio |

Reto Da Forno
reto.daforno@tik.ee.ethz.ch