Table of Contents

Precautions 2
Pins, pinout, pin map and pin file 3
All about pins 3
Pinout and mapping for the MSP430-G2-553 (Revision 1.5) 3
Pin map 4
Extract from header file “pins_energia.h” 5
Use of Energia functions with MSP430 pins 6
Important Tips 6
Troubleshooting 6
Reset instructions 6
Common errors 6
Explanation of the code (sketch in Energia) “Blink.ino” 7

Precautions

1. **DO NOT** connect wires or components while the board is connected to the computer!
2. Once in Energia, click on Help -> About Energia. Ensure you are running version 17 as shown in the screenshot below:

Pins, pinout, pin map and pin file

All about pins

- **Physical pins**: In this workshop, when referring to physical pins in the script, you **DO NOT** have to physically do anything with the pins. By physical pins, we are referring to the pin numbering of the MSP430G2553 microcontroller. For example, if you look at the pin map of the MSP430, the physical pins refer to the pin numbers in black boxes starting from pin 1 all the way through to pin 20.
- **Pins and pin aliases**: When referring to pin aliases in documentation, we usually write them using a dot. For example, physical pin 2, which corresponds to the red LED, is also referred to by the pin alias "P1.0". However, as shown in the header file "pins_energia.h", when wishing to use this pin alias in our code, we need to use an underscore instead of a dot. In this case, for example, we would write "P1_0" instead of "P1.0" in our code.
- **Commands and functions**: The reason that we use underscores for our aliases is primarily due to the fact that dots are reserved for commands and functions. For example, the command "Serial" allows us to operate the hardware UART on the MSP430. This command has a number of functions, for example, "begin()", "end()", "print()" and many more. To execute one of this functions, we would type "Serial.begin(9600)", for example, to set the data rate at 9600 bits per second for serial data transmission. Notice the use of the dot to access the "begin()" function with the "Serial" command.

  ADVICE: The best source to identify and understand the different commands used in Energia, is the Energia website: http://energia.nu/Reference_Index.html

Pinout and mapping for the MSP430-G2-553 (Revision 1.5)

5 MSP-EXP430G2 Hardware

5.1 Device Pinout

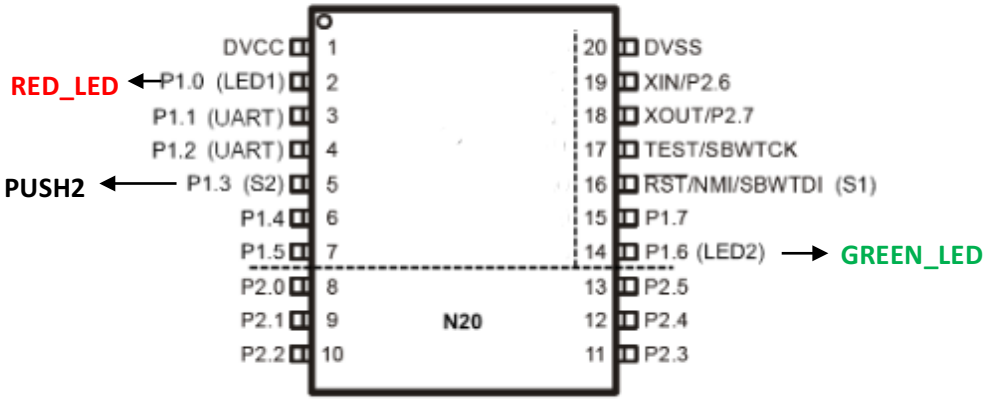


Figure 5. Device Pinout

Figure 1: Device pinout[1] for MSP430G2

[1] See user guide: http://www.ti.com/lit/ug/slau318g/slau318g.pdf

Pin map

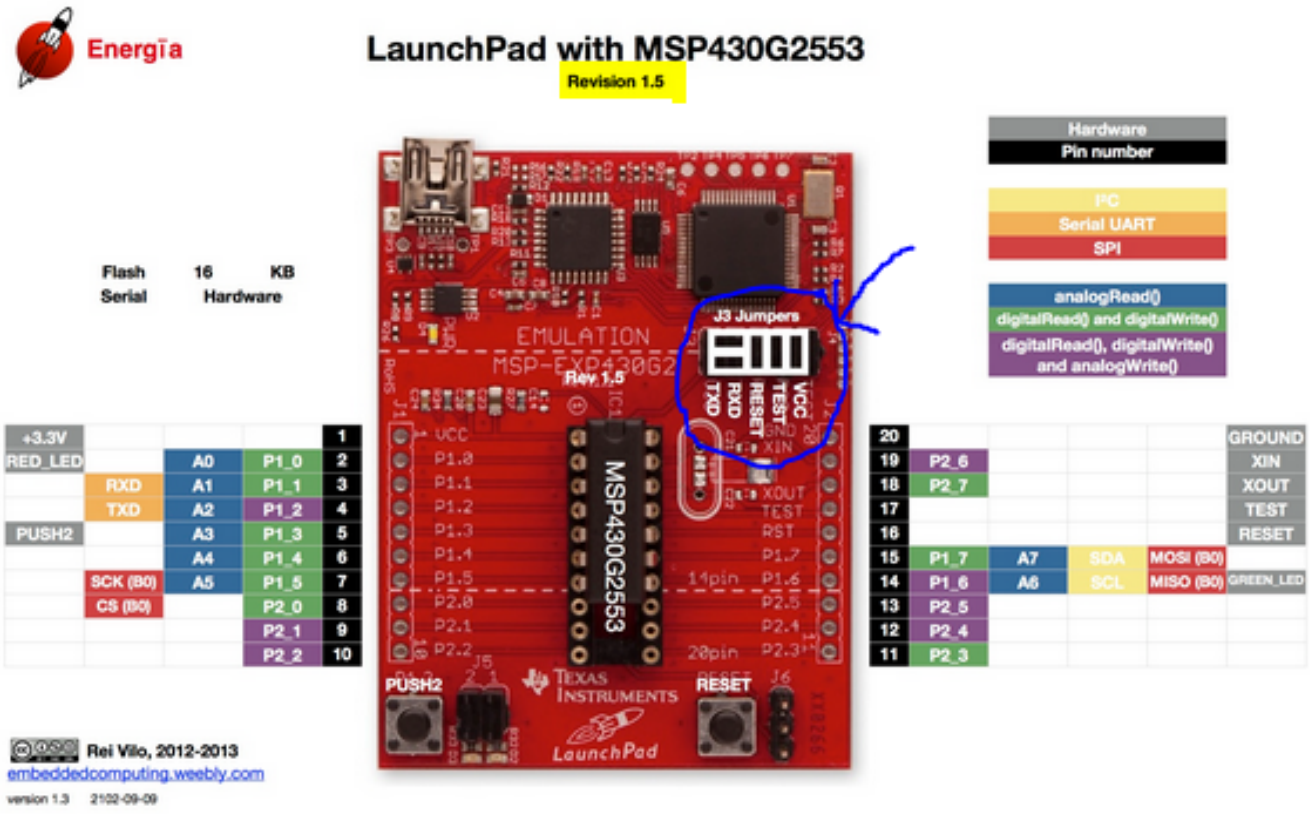


Figure 2: Pin map[2] for Rev 1.5 of MSP430G2553

WARNING: Pay attention to J3 Jumpers for enabling Hardware (HW) UART.

- All pins operate at 3.3V.
- The **black** boxes correspond to the actual physical pins on the board.
- The **grey** boxes correspond to the peripherals already present on the board, such as push buttons and LEDs.
- The **blue** boxes show which pins can be used for analogue inputs, for example, from a pot.
- The **green** boxes show which pins can be used for digital inputs, such as from a pushbutton, or digital outputs, such as to an LED.
- The **purple** boxes show additional pins which can be used for digital inputs or outputs, but in addition, may also serve to write an analogue value to the pin using PWM.
- The remaining boxes, being yellow, orange and red refer to different types of communications supported by the board, however, these do not need to be considered for the purpose of this workshop.

[2] Courtesy of http://energia.nu/wordpress/wp-content/uploads/2014/01/LaunchPads-MSP430G2-%E2%80%94-Pins-Maps-13-42.jpeg

Extract from header file "pins_energia.h"

pins_energia.h

```
103  //
104
105  // Pin names based on the silkscreen
106  //
107  static const uint8_t P1_0 = 2;
108  static const uint8_t P1_1 = 3;
109  static const uint8_t P1_2 = 4;
110  static const uint8_t P1_3 = 5;
111  static const uint8_t P1_4 = 6;
112  static const uint8_t P1_5 = 7;
113  static const uint8_t P2_0 = 8;
114  static const uint8_t P2_1 = 9;
115  static const uint8_t P2_2 = 10;
116  static const uint8_t P2_3 = 11;
117  static const uint8_t P2_4 = 12;
118  static const uint8_t P2_5 = 13;
119  static const uint8_t P1_6 = 14;
120  static const uint8_t P1_7 = 15;
121  static const uint8_t P2_7 = 18;
122  static const uint8_t P2_6 = 19;
123
124  static const uint8_t RED_LED = 2;
125  static const uint8_t GREEN_LED = 14;
126  static const uint8_t PUSH2 = 5;
127  static const uint8_t TEMPSENSOR = 128 + 10; // depends on chip
```

```
81   static const uint8_t A0  = 2;
82   static const uint8_t A1  = 3;
83   static const uint8_t A2  = 4;
84   static const uint8_t A3  = 5;
85   static const uint8_t A4  = 6;
86   static const uint8_t A5  = 7;
87   static const uint8_t A6  = 14;
88   static const uint8_t A7  = 15;
89   static const uint8_t A10 = 128 + 10; // special. This is the internal temp sensor
90
91   //                     +-\/-+
92   //               VCC   1|    |20  GND
93   //         (A0)  P1.0  2|    |19  XIN
94   //         (A1)  P1.1  3|    |18  XOUT
95   //         (A2)  P1.2  4|    |17  TEST
96   //         (A3)  P1.3  5|    |16  RST#
97   //         (A4)  P1.4  6|    |15  P1.7  (A7) (SCL) (MISO) depends on chip
98   //         (A5)  P1.5  7|    |14  P1.6  (A6) (SDA) (MOSI)
99   //               P2.0  8|    |13  P2.5
100  //               P2.1  9|    |12  P2.4
101  //               P2.2 10|    |11  P2.3
102  //                      +----+
103  //
```

Figure 3: Extract from the header file[3] "pins_energia.h" for the MSP430

[3] https://github.com/energia/Energia/blob/master/hardware/msp430/variants/launchpad/pins_energia.h

Use of Energia functions with MSP430 pins

- It is recommended that for external inputs and outputs, different pins are used rather than the ones which have been assigned functionality already.
    - For example, pin 1 is the power supply voltage of the board rated at 3.3V (1.8V – 3.6V) while pin 20 is the ground (0V) for the microcontroller.
    - Pins 2, 3, 4, 5 and 14 may be used for analogue inputs.
        - However, pins 2, 5 and 14 are already connected to the on-board push buttons and LEDs while pins 3 and 4 are used for the UART (RXD, TXD). Hence, it is recommended that these pins (2, 3, 4, 5 and 14) are not used to connect to analogue inputs.
    - Pin 9 is recommended for writing an analogue value to it using pulse width modulation (PWM) since no other peripherals are connected to this pin.
- A few examples of functionality in Energia include the following:
    - Using the digitalRead() function to read the value from push button labelled PUSH2.
    - Using the digitalWrite() function to toggle the RED and GREEN LEDs on or off.
    - Using the analogWrite() function to write an analogue value to pin 9.

Important Tips

- For a detailed description of the icons in the Energia IDE, navigate to the link below: http://energia.nu/Guide_Environment.html
- Only make **ONE CHANGE at a time** to your code and test.
- **DO NOT** leave spaces in filenames.
- Try and use English characters in file paths wherever possible.

Troubleshooting

Reset instructions

1. First press the RESET push button and hold it for 5 seconds.
2. Shut down Energia, unplug the MSP430 board, re-plug the board back in, restart Energia.
3. Press and hold the RESET button again for 5 seconds.
4. Run the built-in Blink example to ensure everything is back to normal.
5. Then return to the task under consideration.

Common errors

A common error that occurs is the following:
tilib: MSP430_Initialize: Could not find MSP-FET430UIF on specified COM port (error = 57)
tilib: device initialization failed

If this error occurs, try disconnecting the MSP430 launchpad from the computer and then reconnecting it. If this does not solve the issue, then follow the reset instructions above.

Explanation of the code (sketch in Energia) “Blink.ino”

- The programme starts with a description of what the code does:

```
Blink §
/*
  Blink
  The basic Energia example.
  Turns on an LED on for one second, then off for one second, repeatedly.
  Change the LED define to blink other LEDs.

  Hardware Required:
  * LaunchPad with an LED

  This example code is in the public domain.
*/
```

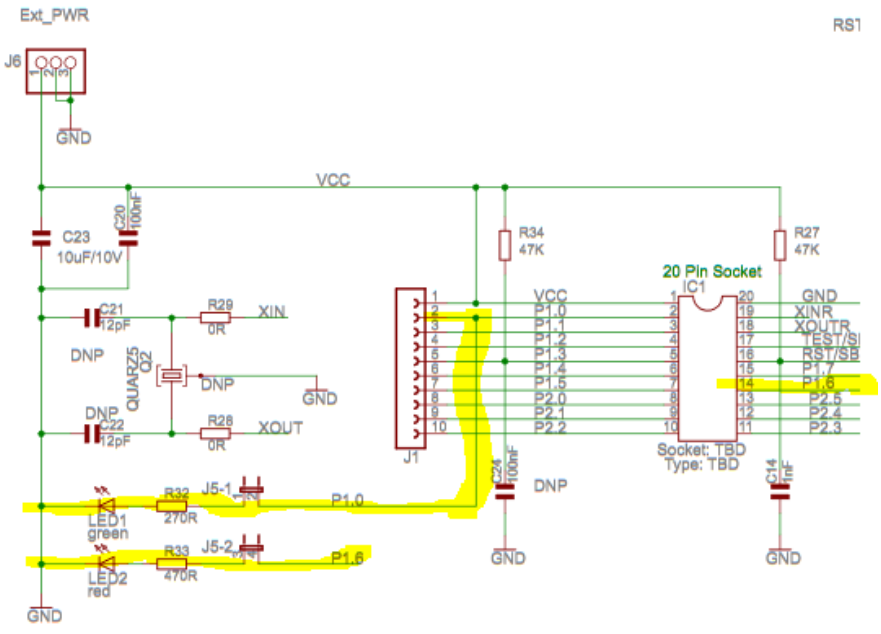
- Subsequently, it defines constants using the #define directive. For example, in the following code snippet, the red LED on the Launchpad can be addressed using the name “LED”.

```
// most launchpads have a red LED
#define LED RED_LED

//see pins_energia.h for more LED definitions
//#define LED GREEN_LED
```

As explained in the lecture notes, the name “RED_LED” is actually an alias for the physical pin to which the actual red LED on the Launchpad is connected to, as shown on page 20 of the user guide, an extract of which is presented below. This alias is defined in the header file “pins_energia.h”.

```
124  static const uint8_t RED_LED = 2;
```



Looking at the pinout and mapping for the MSP430-G2-553, presented earlier in this *Read Me* document, as well as the extract from the user guide above, we can see that the alias name "RED_LED" **should** correspond to physical pin 2 on the Launchpad.

WARNING: The user guide has a mistake in that the two LEDs have been swapped. Physical pin 2, also referred to by the alias P1.0, actually corresponds to the red LED not the green LED (LED1) shown in the user guide extract. This can be verified as follows; change the "`#define LED RED_LED`" to "`define LED 2`", which means that the name "LED" corresponds to physical pin 2. Downloading the programme to the MSP430 and running it will show that **it is the red LED** which toggles ON and OFF and not the green LED.

- There are two routines in an Energia sketch; a setup routine and a loop routine.
  - The command "void" simply means that the routine does not return any values.
  - The setup routine is responsible for initialising variables, pin modes, serial communication and more. In this example, the setup routine sets the pin defined by LED (in this case, physical pin 2), to be an output pin.

```
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(LED, OUTPUT);
}
```

As stated on the Energia website: "*The setup() function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc. The setup function will **only run once**, after each powerup or reset of the LaunchPad board.*" [http://energia.nu/Setup.html]

- The loop routine runs the code within it continuously allowing various aspects of the Launchpad to be controlled using different commands.
  - In this example, the loop routine turns the LED connected to physical pin 2 (defined by LED and also addressed by the alias RED_LED) ON and OFF continuously every second.
  - The commands HIGH and LOW are reserved words in Energia and correspond to a high state (+Vcc or '1') and a low state (Ground, or '0'), respectively.
  - The function "digitalWrite", writes a value to a digital pin, as explained in the lecture notes.
  - The function "delay", pauses the routine for the amount specified in brackets (defined in milliseconds).

```
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(LED, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(LED, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

As stated on the Energia website: "*After creating a setup() function, which initializes and sets the initial values, the loop() function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the LaunchPad board.*" [http://energia.nu/Loop.html]