Liftoff!

Hands-on Intro to the TI-MSP430

Introduction

Texas Instruments TI-MSP430 G2553 Microcontroller

Alternative to the AVR-based Arduino Platform

Energia Programming environment
    Forked from Arduino

Couple of rough edges, but basically a feature complete-port

Setting up Energia

https://github.com/energia/Energia/wiki/Getting-Started

You will need to have Java installed (you probably already do).

1 - Download and install the drivers for the Launchpad dev board (If you are using Linux, you should be able to skip this step)

2 - Download and extract Energia

energia-0101E0009-macosx.dmg - Mac OS X

energia-0101E0009-windows.zip - Windows

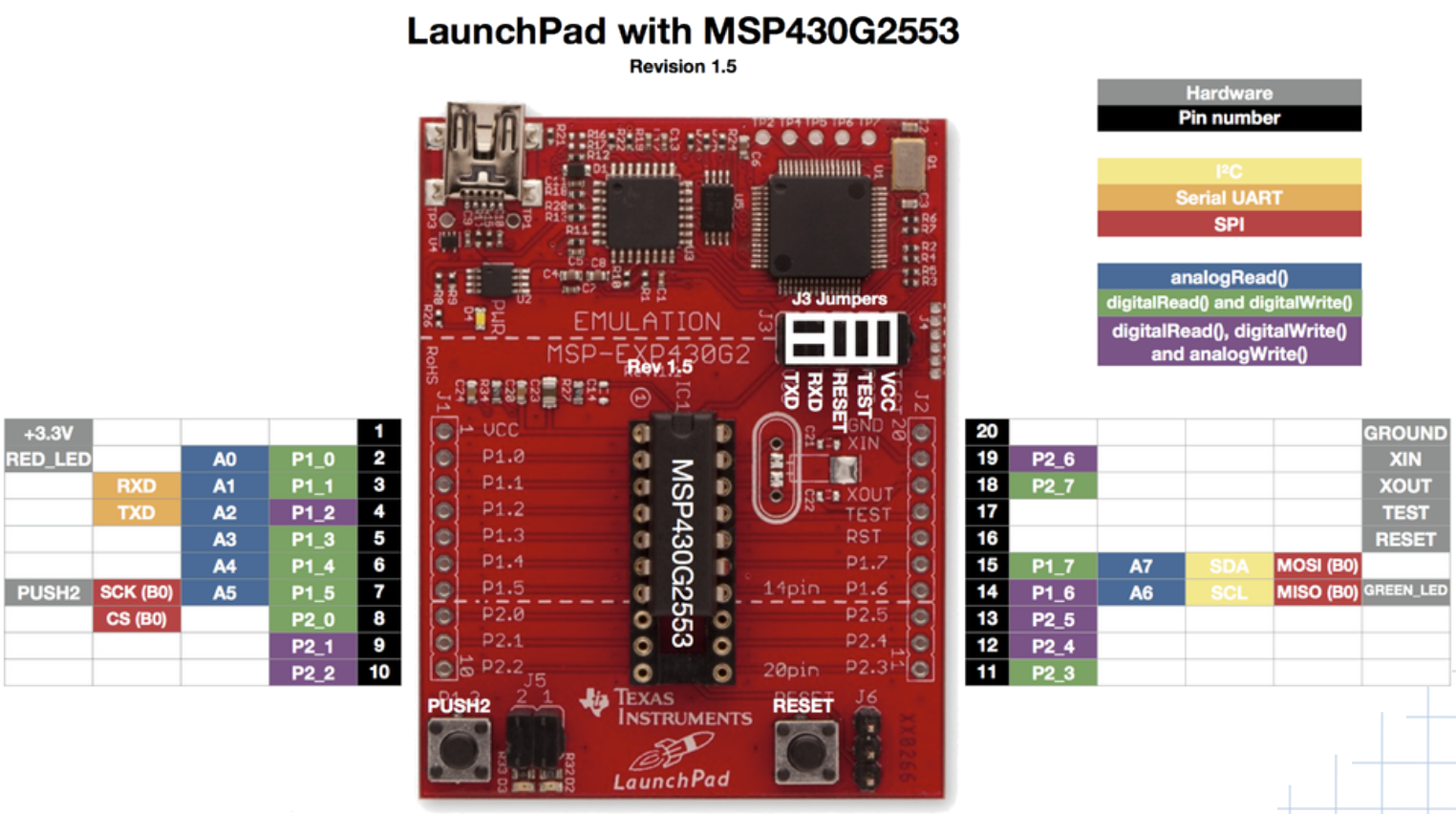
energia-0101E0009-linux.tgz - Linux

3 - To run Energia, simply run the executable in the folder you just extracted.

'Hello World'

A simple "blink the led" program.

Pinout


LaunchPad with MSP430G2553
Revision 1.5
Hardware
Pin number
I²C
Serial UART
SPI
analogRead()
digitalRead() and digitalWrite()
digitalRead(), digitalWrite() and analogWrite()
J3 Jumpers
EMULATION
MSP-EXP430G2
Rev 1.5
MSP430G2553
14pin
20pin
PUSH2
RESET
Texas Instruments
LaunchPad
+3.3V
RED_LED
PUSH2
RXD
TXD
SCK (B0)
CS (B0)
A0
A1
A2
A3
A4
A5
P1_0
P1_1
P1_2
P1_3
P1_4
P1_5
P2_0
P2_1
P2_2
P2_6
P2_7
P1_7
P1_6
P2_5
P2_4
P2_3
SDA
SCL
MOSI (B0)
MISO (B0)
GROUND
XIN
XOUT
TEST
RESET
GREEN_LED

Interrupts

Instead of running the processor all the time, we can put it (and various clocks) into a low power 'sleep' mode when we aren't using it. Interrupts will wake the MSP430 from a sleep mode.

Interrupt Sources:

- -Port Interrupts (Change in state on a digital input)
- -Timer Interrupts (Timer counter has reached a certain value)
- -ADC Interrupts (ADC has finished reading)
- -USI/UART (data has been received over serial)
- -Reset (Cannot be overridden)
- -Oscillator fault/flash memory access violation (Cannot be overridden)

Sleep Modes

LPM0 - The CPU is disabled.
LPM1 - The loop control for the fast clock (MCLK) is also disabled.
LPM2 - The fast clock (MCLK) is also disabled.
LPM3 - The DCO oscillator and its DC generator are also disabled.
LPM4 - The crystal oscillator is also disabled.

As sleep mode level increases, so does the time needed to wake up the chip!

Luckily, even the worst case wake up time from LPM4 is only a few microseconds.

Timers! (The non-arduino way)

Once enabled, timers trigger an interrupt when the CCR detects the counter is equal to the set target value, or the counter has overflowed. In order to make our own code run when the timer interrupt is triggered, we need to make an "Interrupt Service Routine", or ISR.

Clock Sources

ACLK: Auxiliary clock

The signal is sourced from LFXT1CLK (the external crystal) with a divider of 1, 2, 4, or 8. ACLK can be used as the clock signal for Timer A.

MCLK: Master clock

The signal can be sourced from LFXT1CLK, XT2CLK (if available), or DCOCLK with a divider of 1, 2, 4, or 8. MCLK is used by the CPU and system.

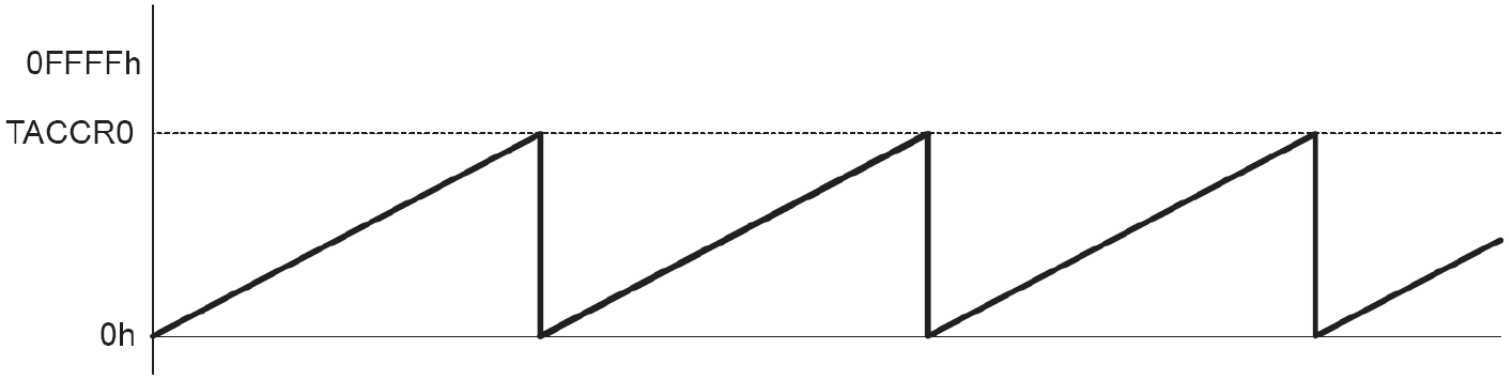
SMCLK: Sub-main clock

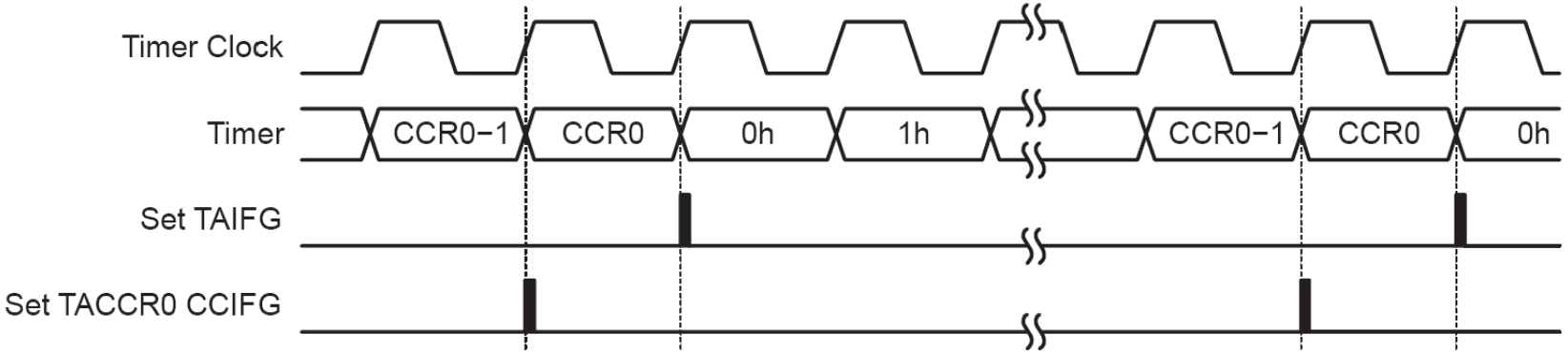
The signal is sourced from either XT2CLK (if available), or DCOCLK with a divider of 1, 2, 4, or 8. SMCLK can be used as the clock signal for Timer A.

Timer A

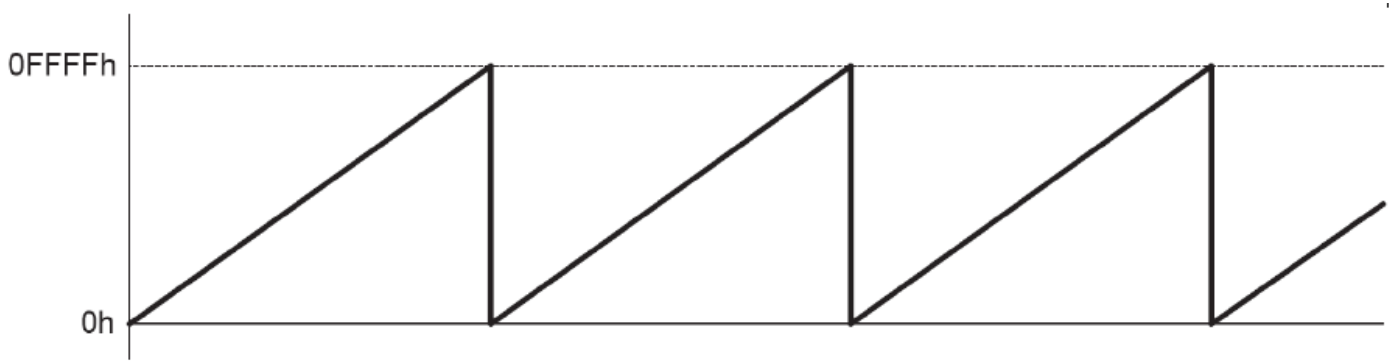
16-bit counter
4 modes of operation – Stop, Up, Continuous, Up/Down
3 capture/compare registers (CCRx)
2 interrupt vectors – TACCR0 and TAIV

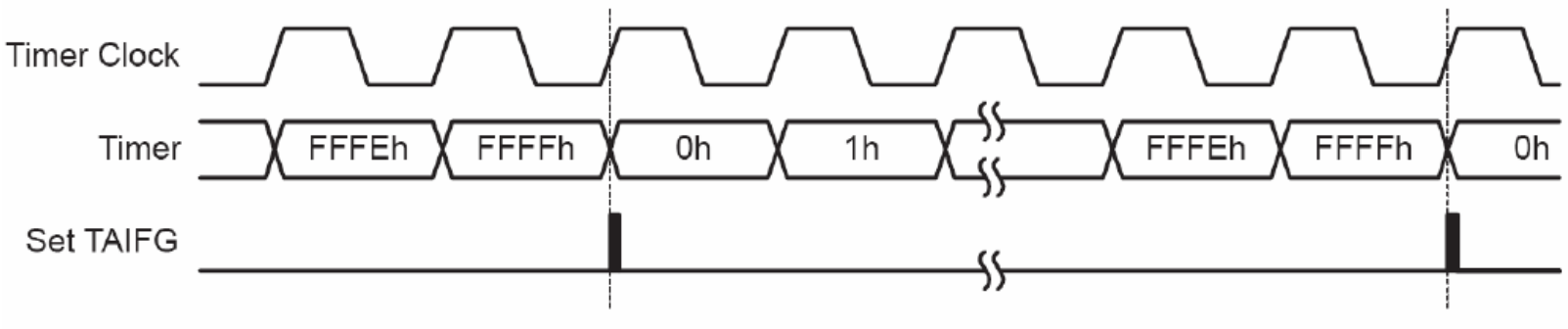
Up Mode


0FFFFh
TACCR0
0h

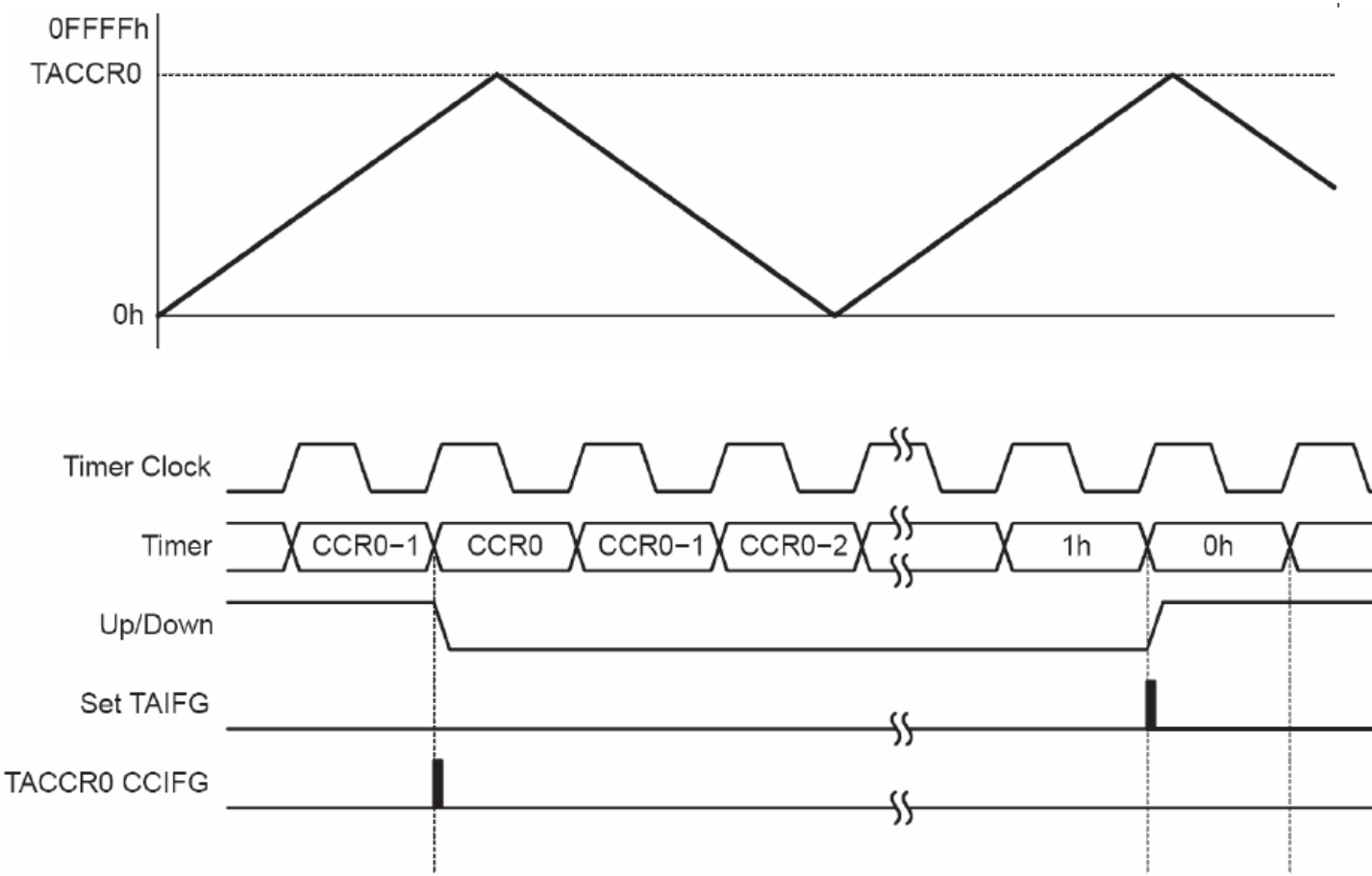

Timer Clock
Timer
CCR0−1
CCR0
0h
1h
CCR0−1
CCR0
0h
Set TAIFG
Set TACCR0 CCIFG

Continuous Mode

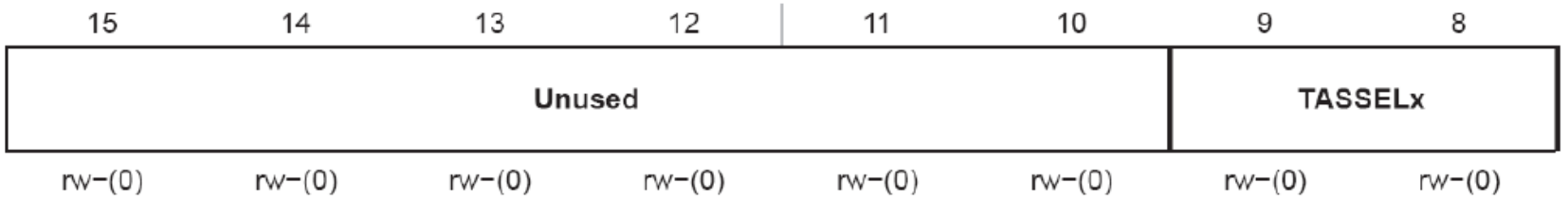




Up/Down Mode


0FFFFh
TACCR0
0h
Timer Clock
Timer
CCR0−1
CCR0
CCR0−1
CCR0−2
1h
0h
Up/Down
Set TAIFG
Set TACCR0 CCIFG

TACTL (part 1)



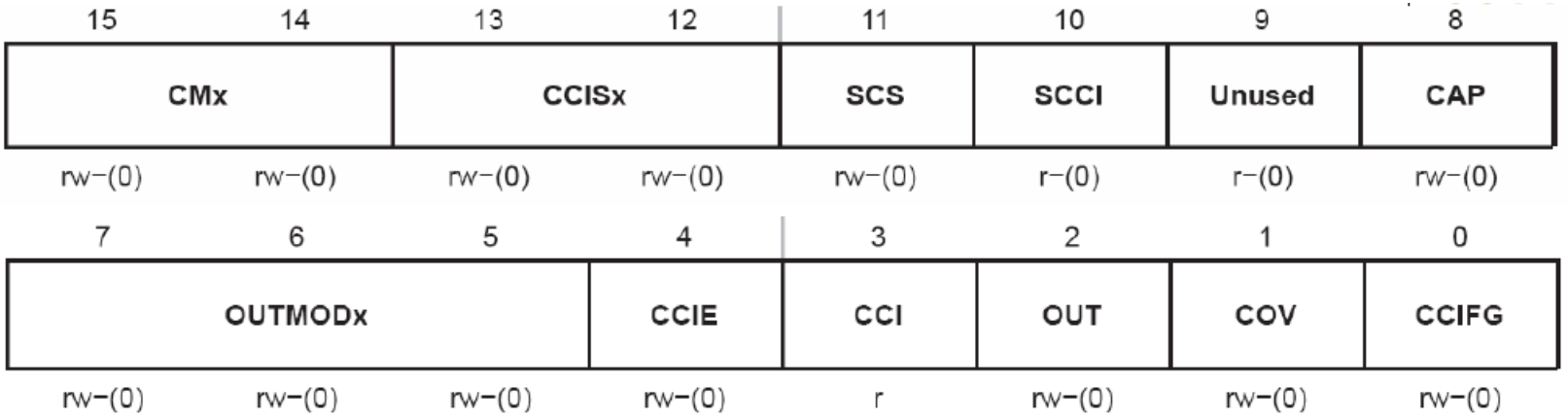
Unused	Bits 15-10	Unused
TASSELx	Bits 9-8	Timer_A clock source select 00 TACLK 01 ACLK 10 SMCLK 11 INCLK

TACTL (part 2)

7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG
rw-(0)	rw-(0)	rw-(0)	rw-(0)	rw-(0)	w-(0)	rw-(0)	rw-(0)

IDx	Bits 7-6	Input divider. These bits select the divider for the input clock. 00 /1 01 /2 10 /4 11 /8
MCx	Bits 5-4	Mode control. Setting MCx = 00h when Timer_A is not in use conserves power. 00 Stop mode: the timer is halted 01 Up mode: the timer counts up to TACCR0 10 Continuous mode: the timer counts up to 0FFFFh 11 Up/down mode: the timer counts up to TACCR0 then down to 0000h
Unused	Bit 3	Unused
TACLR	Bit 2	Timer_A clear. Setting this bit resets TAR, the TACLK divider, and the count direction. The TACLR bit is automatically reset and is always read as zero.
TAIE	Bit 1	Timer_A interrupt enable. This bit enables the TAIFG interrupt request. 0 Interrupt disabled 1 Interrupt enabled
TAIFG	Bit 0	Timer_A interrupt flag 0 No interrupt pending 1 Interrupt pending

TACCTLx



15	14	13	12	11	10	9	8
CMx		CCISx		SCS	SCCI	Unused	CAP
rw−(0)	rw−(0)	rw−(0)	rw−(0)	rw−(0)	r−(0)	r−(0)	rw−(0)

7	6	5	4	3	2	1	0
OUTMODx			CCIE	CCI	OUT	COV	CCIFG
rw−(0)	rw−(0)	rw−(0)	rw−(0)	r	rw−(0)	rw−(0)	rw−(0)

CAP Bit 8 Capture mode
- 0 Compare mode
- 1 Capture mode

CCIE Bit 4 Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag.
- 0 Interrupt disabled
- 1 Interrupt enabled

CCIFG Bit 0 Capture/compare interrupt flag
- 0 No interrupt pending
- 1 Interrupt pending

Example: 'Hello World v2.0'

```
void setup() {
          P1DIR |= BIT6;                  // P1.6 to output
        CCR0 = 65536-1;                   // PWM Period
        CCTL1 = OUTMOD_7;                 // CCR1 reset/set
        CCR1 = 100;                       // CCR1 PWM duty cycle
        TACTL = TASSEL_2 + MC_1 + ID_3;   // SMCLK, up mode, divide clock by 8
        CCTL0 = CCIE;                     //Enable timer interrupts
        _BIS_SR(LPM0_bits);               // Enter LPM0
}

void loop() {
      //Don't do anything here...we are waiting for interrupts!
}

#pragma vector=TIMER0_A0_VECTOR    //These two lines say this is an ISR
__interrupt void Timer_A(void) {                //cont.
      P1OUT ^= BIT6;                           //Toggle pin 1.0
}
```

Reading a Digital Value

digitalRead(pin);

where pin is a digital pin; returns either HIGH or LOW

https://github.com/cantwt/TVCOG_MSP430_POD

Writing a Digital Value

digitalWrite(pin, value);

where pin is a pin identifier and value is HIGH or LOW

Reading an Analog Value

analogRead(pin);

This will return a number in the range 0-1023 that maps to the input voltage (from 0 to Vref)

To set the reference voltage, use:

analogReference(type);

where type can be:

DEFAULT: the default analog reference of input voltage (VCC) ~3.3V-3.6V
INTERNAL1V5: internal analog reference voltage of 1.5V
INTERNAL2V5: internal analog reference voltage of 2.5V
EXTERNAL: the voltage applied to the VREF pin is used as the reference.

Let's build a THAT

(Temperature and Humidity Analysis Thing)

-Uses a humidity/temperature sensor
-Uses a CdS cell to sense light level
-Outputs data via morse code using a LED

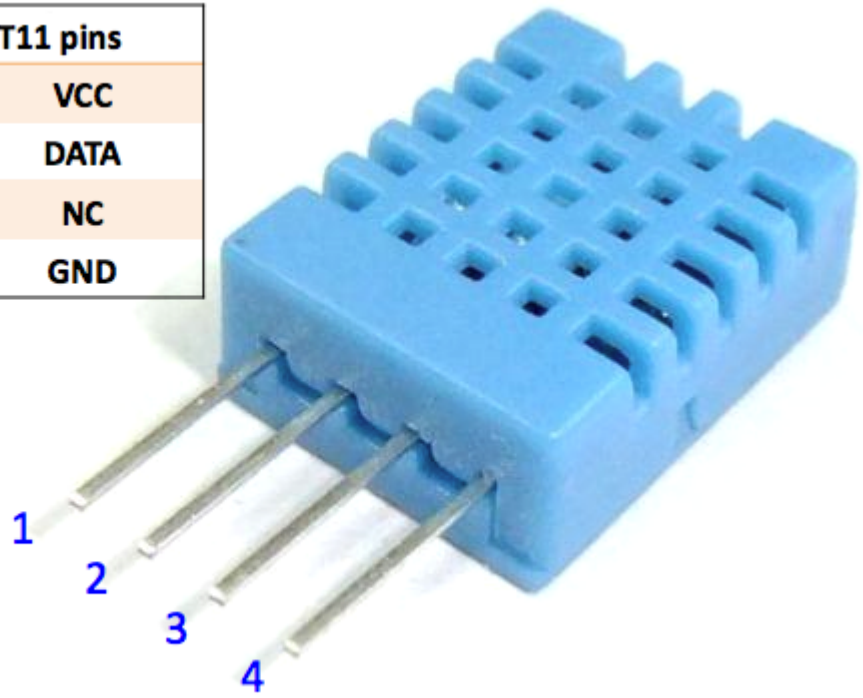
DHT11 Temp/Humidity Sensor

Communicates over a proprietary 1 wire protocol...

We'll just use a library to deal with it. (Originally written for Arduino)

To Connect to the Launchpad:

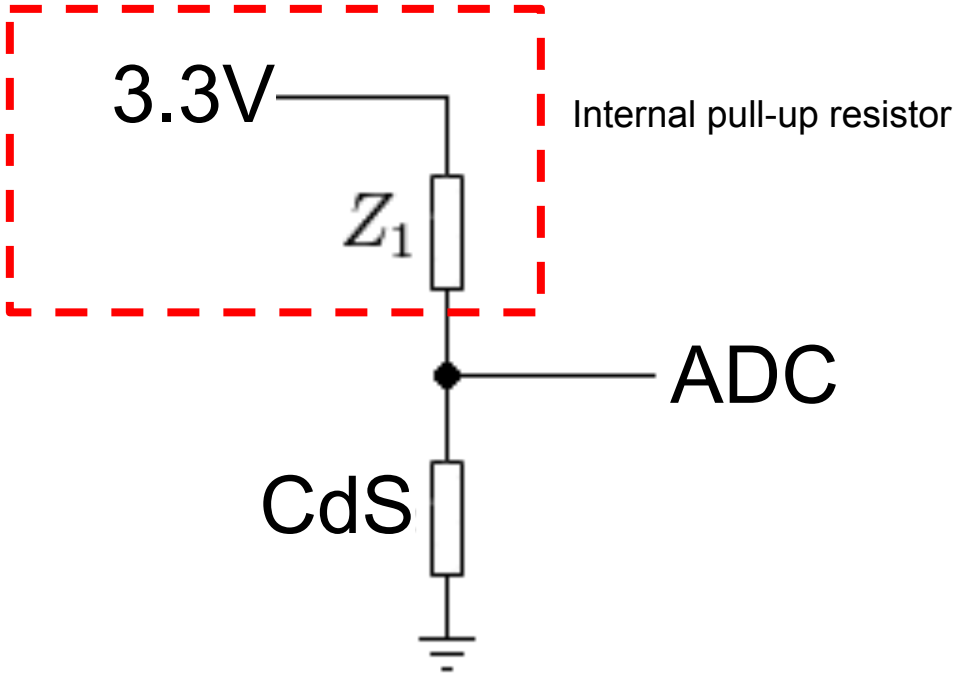
DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



CdS PhotoResistor

Cadmium Sulfide cell is a resistor whose resistance decreases with increasing incident light intensity. This is due to photons (of sufficient frequency) causing bound electrons to move into the conduction band (decreasing the resistance).

We will use these in a voltage divider, using the internal pull-up resistor.



Morse Code

Example to make the MSP430 talk in Morse Code

The table for morse code implemented in a large switch statement

Uses a modified delay to use LPM1 instead of LPM0