Practical Training 1

MSP430 basics

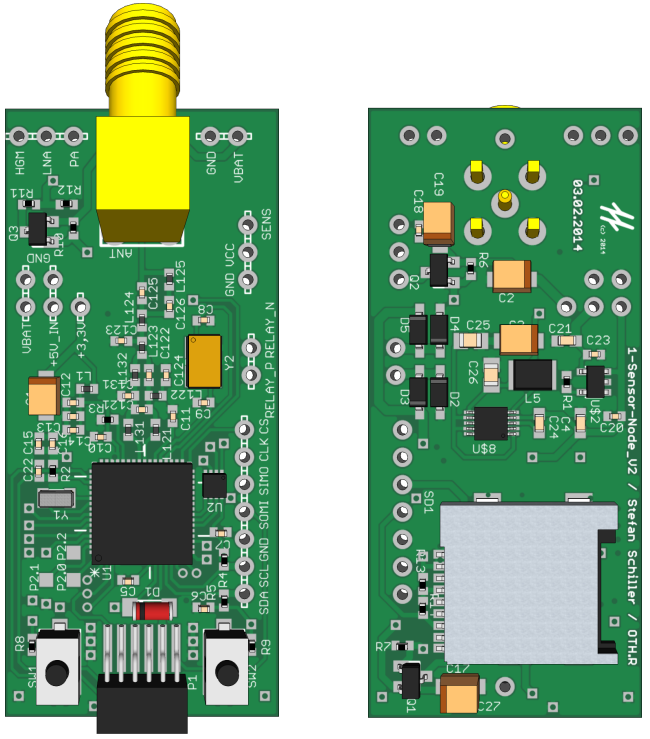
Practical training 1

- First steps / project setup
- Clocks
- GPIO
- ADC
- Timer
- Flash
- UART
- SPI / port expander

hardware

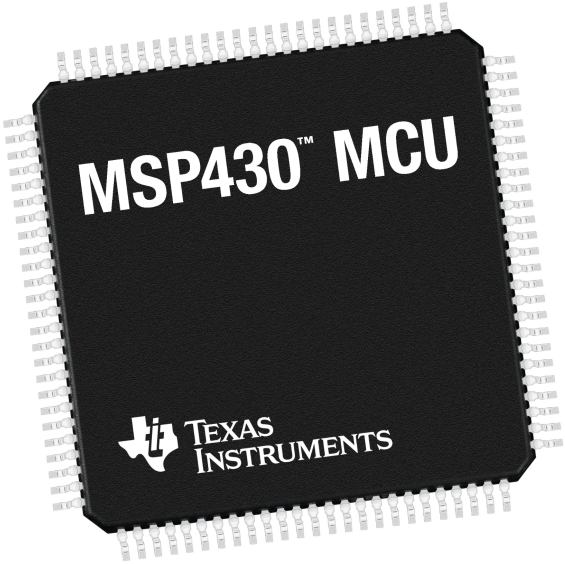
CC430 target board

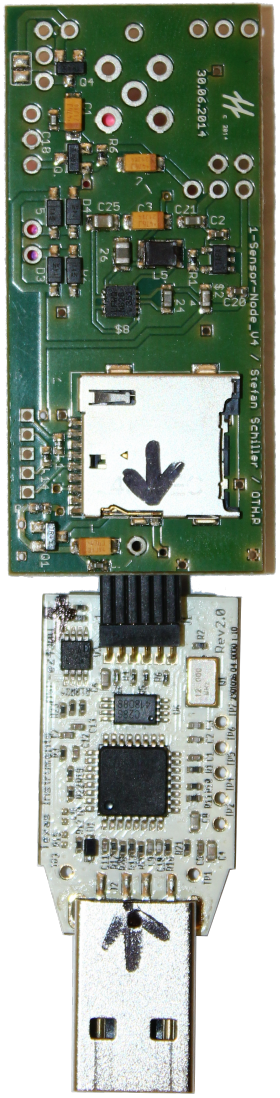
- CC430F6137
- microSD card
- UART
- SPI
- push button



CC430F6137

- 16 bit Microcontroller
- 20 MHz maximal frequency
- 32 kB of Flash memory
- 4 kB RAM
- 32 GPIOs
- 2x 16 bit Timers
- 10 and 12 bit ADC
- 128 Bit AES Security Encryption Coprocessor
- Communication via UART, SPI and I^2C

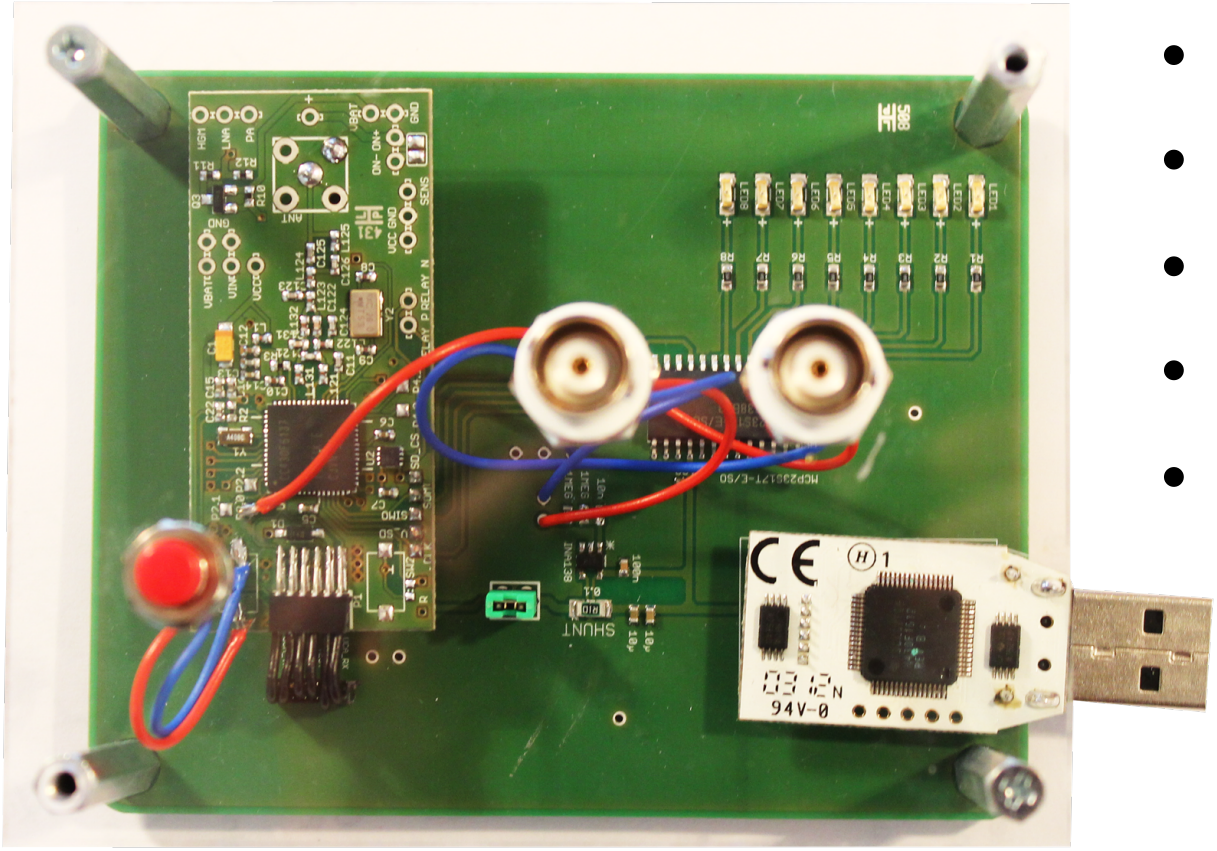




debugger

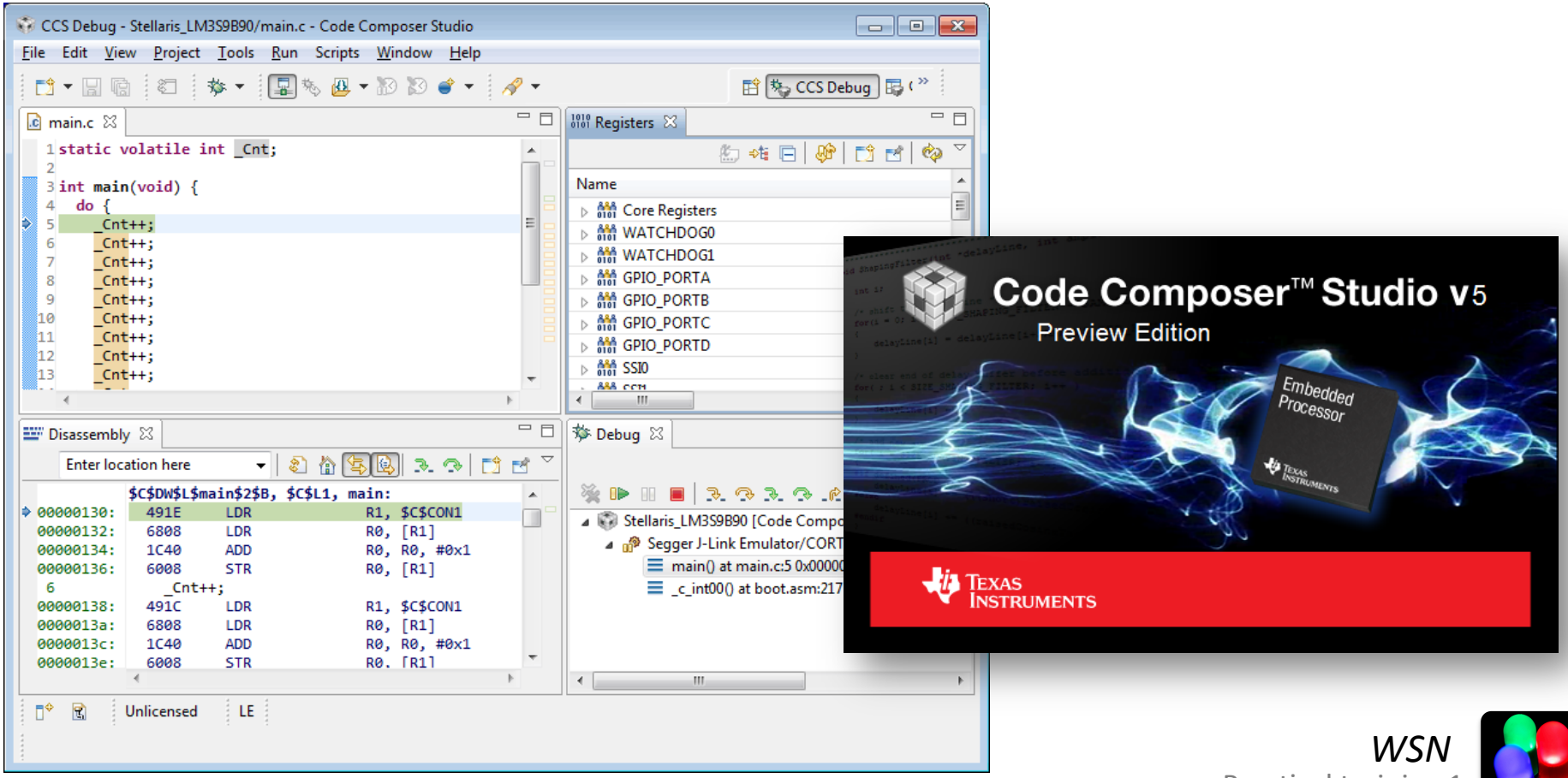
- Flash Emulation Tool (FET)
- Spy-Bi-Wire (2 connection JTAG)
- Set Breakpoints
- Watch registers and flash
- UART emulation

measure interface



- Current shunt
- SPI port expander
- 8 LEDs
- Push button P1.0
- BNC on P2.0

software


CCS Debug - Stellaris_LM3S9B90/main.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
CCS Debug
main.c
static volatile int _Cnt;
int main(void) {
do {
_Cnt++;
Registers
Name
Core Registers
WATCHDOG0
WATCHDOG1
GPIO_PORTA
GPIO_PORTB
GPIO_PORTC
GPIO_PORTD
SSI0
Disassembly
Enter location here
CDWLmain2B, CL1, main:
00000130: 491E LDR R1, CCON1
00000132: 6808 LDR R0, [R1]
00000134: 1C40 ADD R0, R0, #0x1
00000136: 6008 STR R0, [R1]
6 _Cnt++;
00000138: 491C LDR R1, CCON1
0000013a: 6808 LDR R0, [R1]
0000013c: 1C40 ADD R0, R0, #0x1
0000013e: 6008 STR R0, [R1]
Debug
Stellaris_LM3S9B90 [Code Compo
Segger J-Link Emulator/CORT
main() at main.c:5 0x0000
_c_int00() at boot.asm:217
Unlicensed
LE
Code Composer™ Studio v5
Preview Edition
Embedded Processor
Texas Instruments

operators

```
 A    =   0b01101001
~A    =   0b10010110
 A   |=   0b00000010   ➔   A=0b01101011
 A   &=  ~0b00001000   ➔   A=0b01100001
 A   ^=   0b10001000   ➔   A=0b11100001
 A   <<             2   ➔   A=0b10100100
 A   >>             2   ➔   A=0b00011010
```


registers

PxDIR	set the direction of Port x as Input or Output
PxOUT	set the state of Port x when set as Output
PxIN	reads the state of Port x when set as Input
PxIE	enables interrupts on Port x

interrupts

```
// Port1 interrupt service routine ISR
#pragma vector = PORT1_VECTOR
__interrupt void Port1(void){

    P4OUT ^= BIT1;                  // P4.1 toggle
    P1IFG &= ~BIT0;                 // P1.0 IFG Flag clear
}
```


watch dog timer

```
// Stop watchdog timer

 WDTCTL = WDTPW | WDTHOLD;
```


reset with watch dog timer

```
#pragma vector = PORT1_VECTOR
__interrupt void Port1(void){
    P1IFG &= ~BIT0;                 // P1.0 IFG Flag cleared

     if (!(BIT0 & P1IN)){
        WDTCTL &= ~WDTHOLD;         // Start watchdog timer
        while(1);                   // will reset the program
     }

}
```


low power modes

```
//enter LPM4 w/interrupts
__bis_SR_register(LPM4_bits + GIE);

// Exit active CPU
__bic_SR_register_on_exit(LPM4_bits)
```


delay

```
//basic delay function
_delay_cycles(12000);


//specific for 12 MHz delay
void delay_ms(u_int time_ms) {
    u_int c = 0;
    while (c++ < time_ms) {
        _delay_cycles(12000);
    }
}
```