Raritan.

# MSP430-Mikrocontroller mittels freier Software programmieren

Ingo van Lil, Software-Entwickler

21. März 2015
Chemnitzer Linux-Tage

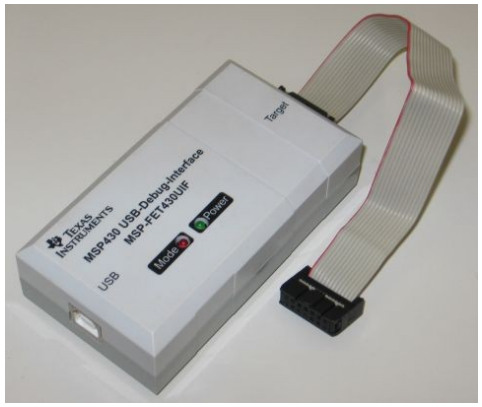
# Die Hardware

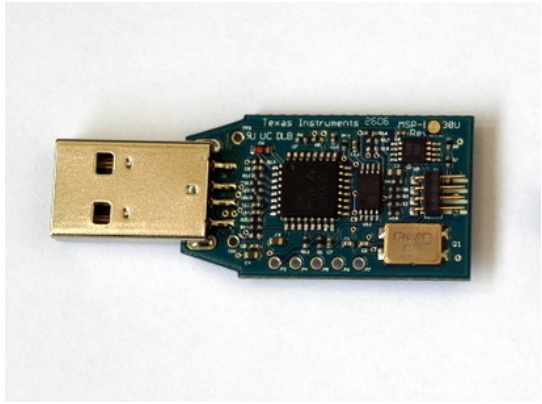

# MSP430-Familie


- 16-Bit RISC-Architektur, 8-25 MHz
- 8-128 Pins, kein externer Speicherbus
- RAM: 128 Byte – 66 KB
- Programmspeicher: 1-512 KB
- Peripherie: GPIO, Timer, USCI, ADC/DAC, etc.
- Spezielles: Funk (CC430), AES, LCD
- Stromspar-Modi

# Programmierschnittstellen

- JTAG
  - Vier Pins
  - Erlaubt Programmierung und Debugging
  - Z.B. MSP-FET430UIF
- Spy-Bi-Wire
  - Serielles JTAG über zwei Pins
  - langsamer
  - Z.B. eZ430-RF2500
- Bootstrap Loader (BSL)
  - Einfaches UART-Protokoll
  - Nur Flash-Zugriff, kein Debugging

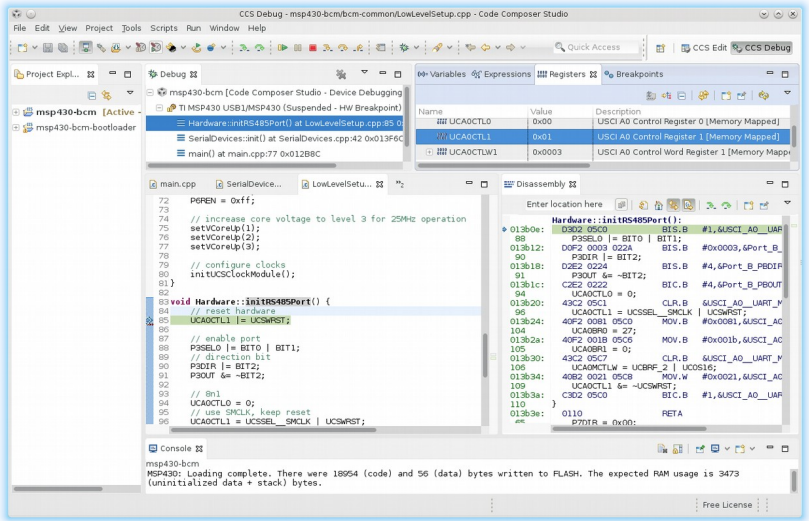


# Die Software

# Code Composer Studio (CCS)

- Offizielle IDE von TI
- Seit Version 5 für Linux verfügbar
- Editor, Compiler, Programmer, Debugger, Disassembler
- Basiert auf Eclipse
- Toolchains:
  - Proprietärer TI-Compiler
  - GCC (seit Version 6.1.0)
- Kostenlose Lizenz
  - TI-Compiler: 16kB-Limit für Programmgröße
  - GCC: Keine Beschränkung



# GCC-Toolchain für MSP430 (alt)

- http://mspgcc.sourceforge.net
- Basiert auf GCC 4.6 (März 2012)
- Stabil und erprobt
- Fedora-Paket: msp430-gcc
  - nur C-Compiler, kein C++
  - In Fedora 21 momentan kaputt (Bugzilla: #1175942)
- Obsolet, nicht mehr gewartet
- Kein MSP430X-Support (20-Bit-Register), nur 64 kB addressierbar

# GCC-Toolchain für MSP430 (neu)

- http://www.ti.com/tool/msp430-gcc-opensource
- Gemeinschaftsprojekt von TI und Red Hat
- Seit einigen Wochen nicht mehr Beta
- Basiert auf GCC 4.9
- Einzeln oder in CCS 6 enthalten, noch keine Fedora-Pakete
- MSP430X-Support (20-Bit-Register)

# MSPDebug

- http://mspdebug.sourceforge.net
- Freies Tool zum Programmieren und Debuggen
- Linux, Windows, BSD, OS/X
- Unterstützt zahlreiche Programmiergeräte und -schnittstellen
- Liest Hex-Files und ELF-Binaries inkl. Symboltabellen
- Disassembler
- GDB-Server
- MSP430-Emulator

# Toolchains im Vergleich

- Aussage von TI: Durchschnittlich ca. 15% größerer und langsamerer Code mit GCC
- Eigener Vergleich (nicht repräsentativ):

| | Gesamt-Programm | Eigene Funktionen | Taktzyklen |
|---|---|---|---|
| TI-Compiler | 1670 Bytes | 1042 Bytes | 803 |
| GCC (alt) | | | |
| -O0 | 1962 Bytes | 1760 Bytes | 610 |
| -O2 | 1456 Bytes | 1254 Bytes | 351 |
| -Os | 1330 Bytes | 1128 Bytes | 370 |
| GCC (neu) | | | |
| -O0 | 2480 Bytes | 1646 Bytes | 641 |
| -O2 | 1902 Bytes | 1054 Bytes | 394 |
| -Os | 1776 Bytes | 926 Bytes | 392 |

# Unterschiede: Interrupt-Handler

TI-Compiler:

```
#pragma vector = TIMER0_A0_VECTOR
__interrupt void timer0_a0_isr(void)
{
    // ...
}
```

MSP430-GCC:

```
__attribute__((interrupt(TIMER0_A0_VECTOR)))
void timer0_a0_isr(void)
{
    // ...
}
```

# Unterschiede: Assembler

TI-Compiler:

```
  .cdecls C,LIST,"msp430.h"

  .global multiply_u16

multiply_u16:
  .asmfunc
    push sr          ; save status register
    dint             ; disable interrupts
    nop

    mov r12, &MPY    ; first operand
    mov r13, &OP2    ; second operand

    mov &RESLO, r12 ; result low word
    mov &RESHI, r13 ; result high word

    pop sr           ; restore status register
    ret
  .endasmfunc
```

MSP430-GCC:

```
#include "msp430.h"

.globl multiply_u16
.type multiply_u16,function

multiply_u16:
    push sr          ; save status register
    dint             ; disable interrupts
    nop

    mov r12, &MPY    ; first operand
    mov r13, &OP2    ; second operand

    mov &RESLO, r12 ; result low word
    mov &RESHI, r13 ; result high word

    pop sr           ; restore status register
    ret
```

# Unterschiede: ABI-Konventionen

Quellcode:

```
int sum(int a, int b);
extern int c;

c = sum(42, 99);
```

MSP430-GCC (alt):

```
  6:   3e 40 63 00     mov #99, r14

  a:   3f 40 2a 00     mov #42, r15

  e:   b0 12 00 00     call #0x0000

 12:   82 4f 00 00     mov r15, &0x0000
```

TI-Compiler, MSP430-GCC (neu):

```
  0:   3c 40 2a 00     mov #42, r12

  4:   3d 40 63 00     mov #99, r13

  8:   b0 12 00 00     call #0x0000

  c:   82 4c 00 00     mov r12, &0x0000
```

# Live-Demonstration