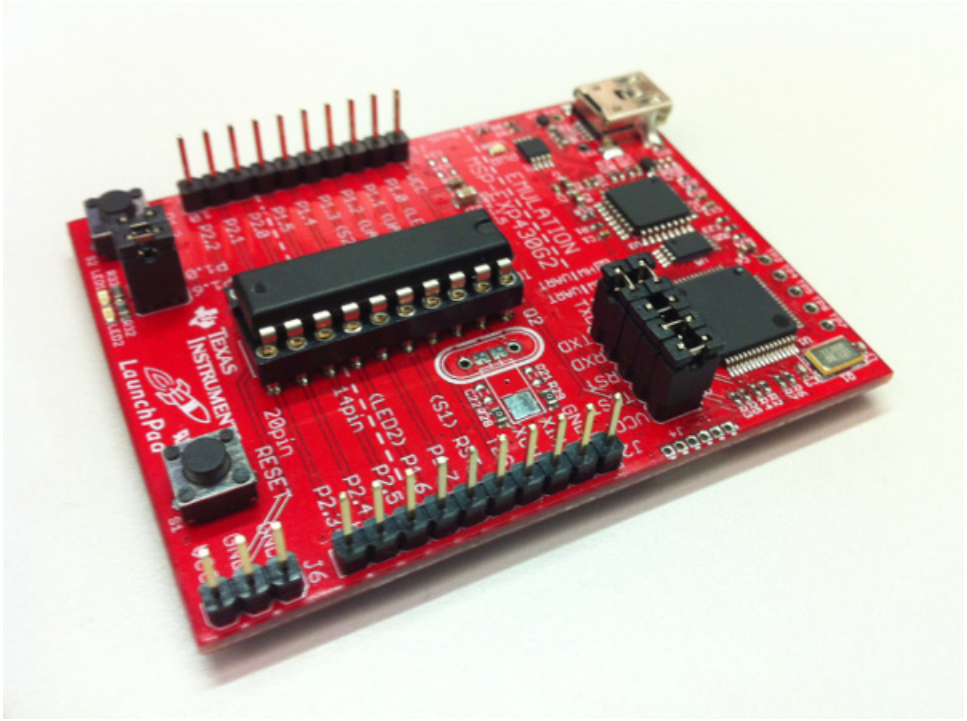
MSP430 - Übersicht

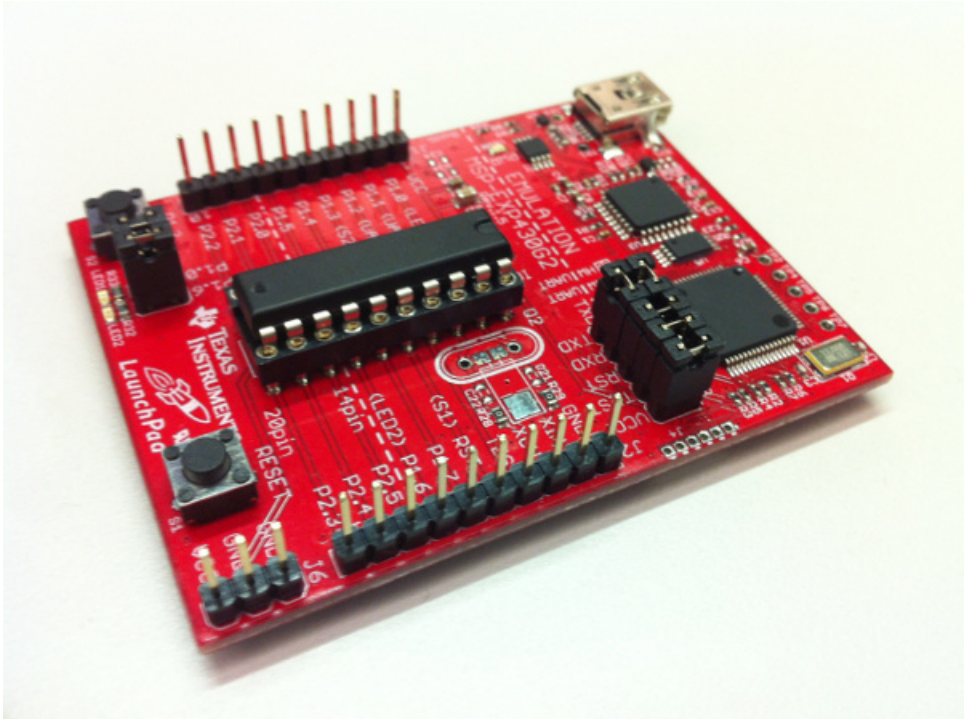


MSP430 - Übersicht

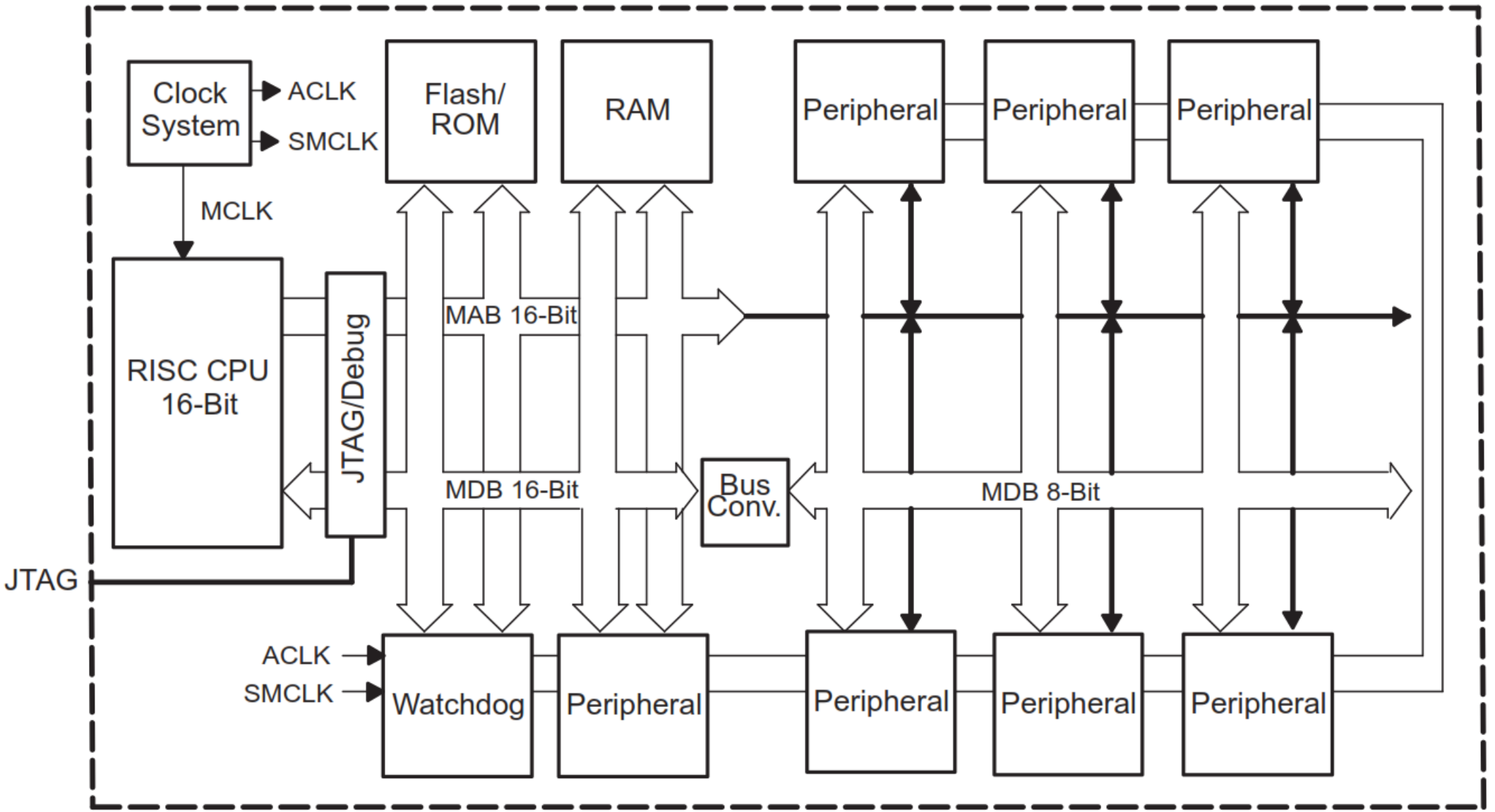
Finally – Materialsammlung (der Anfang):

https://github.com/dg6obo/msp430ckurs

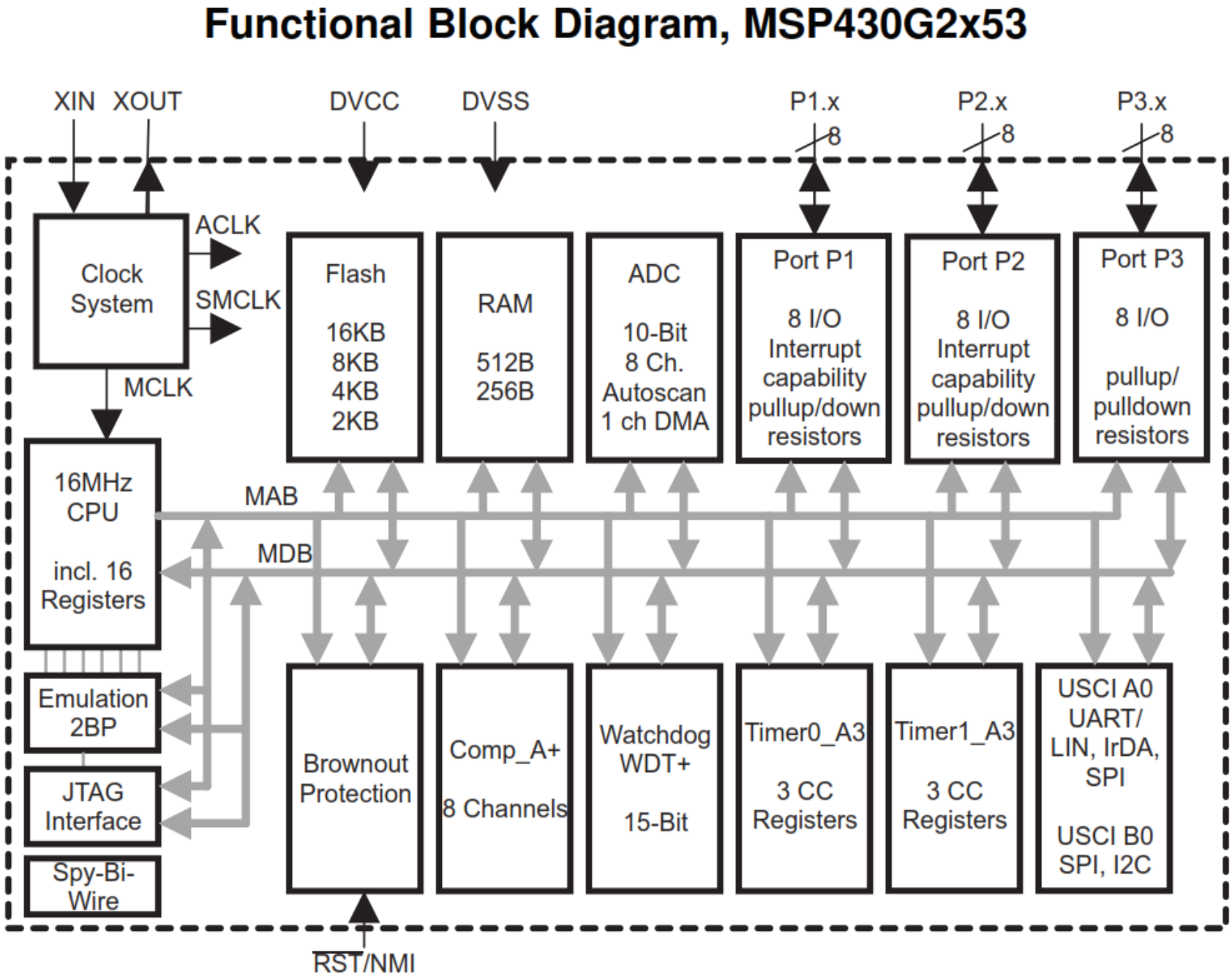
- heute etwas Theorie zum MSP430
- CPU / Clock / GPIO ...
- LED Blinken Teil 2
- mspdebug
- msp430-gcc



MSP430 – Übersicht allgemein



MSP430 – MSP430G2x53

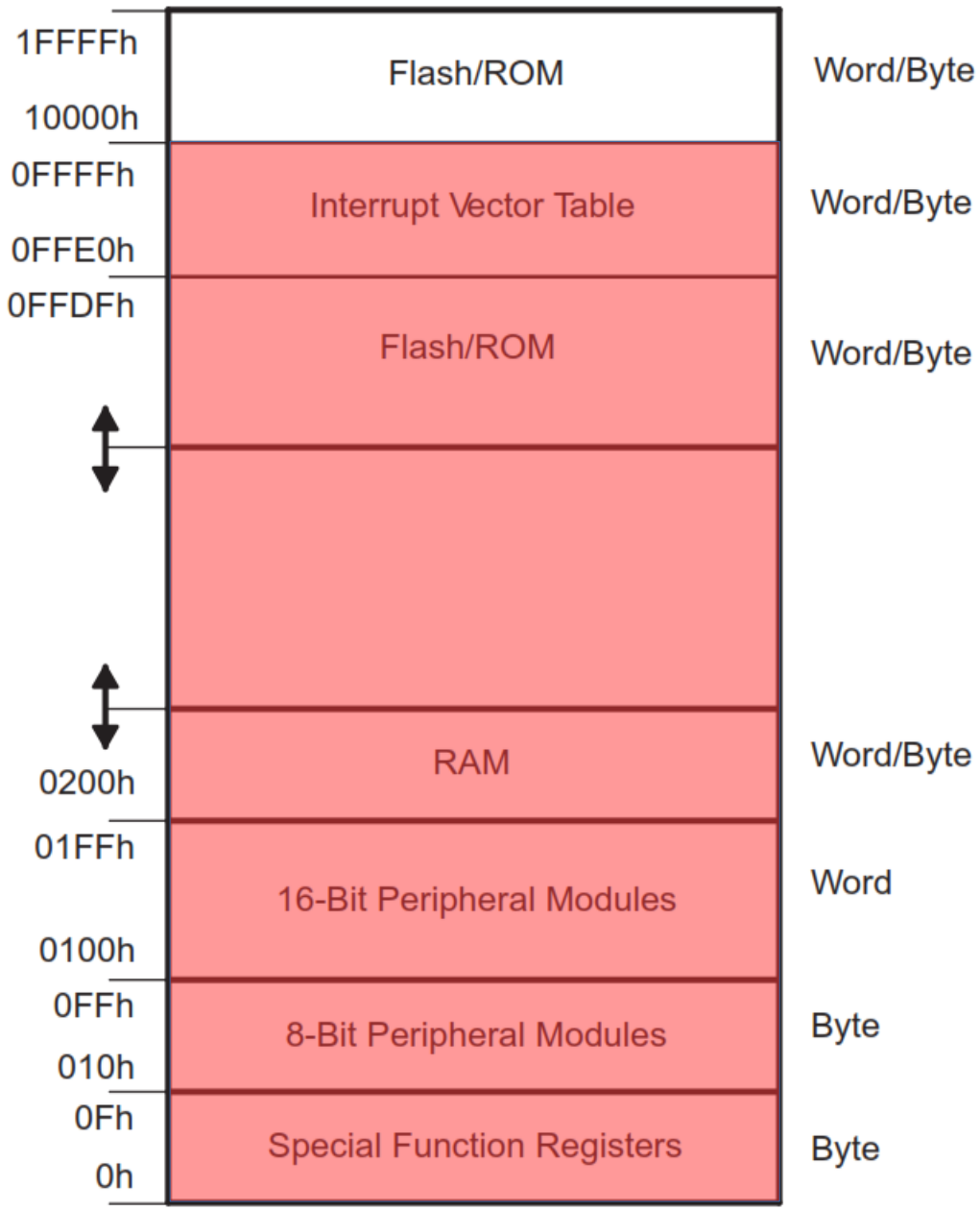


MSP430 – Adressraum

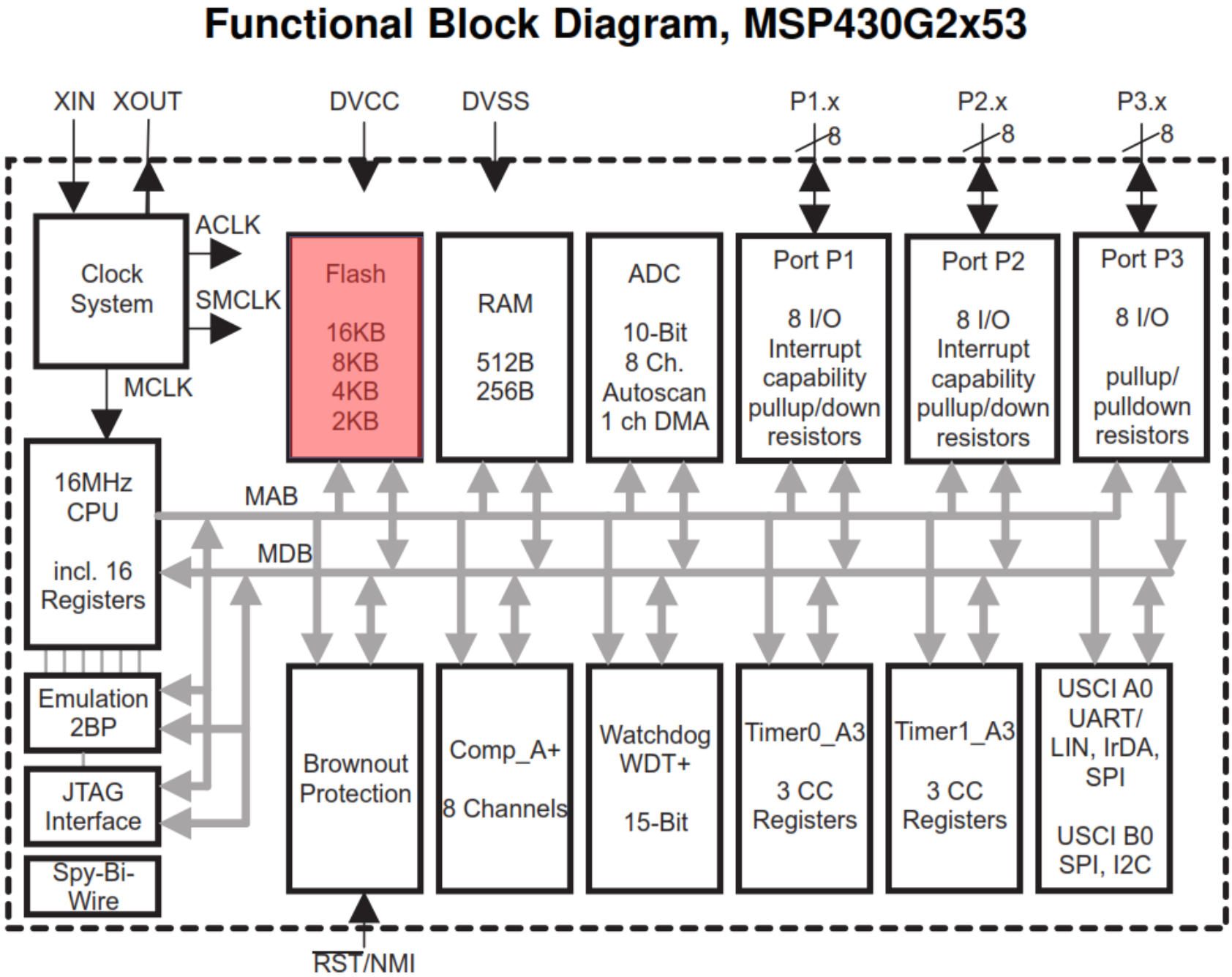
64KByte

- Flash / ROM
- Interrupt Vector Table
- RAM
- 16 Bit Module
- 8 Bit Module
- SFR

von-Neumann Architektur



MSP430 – MSP430G2x53



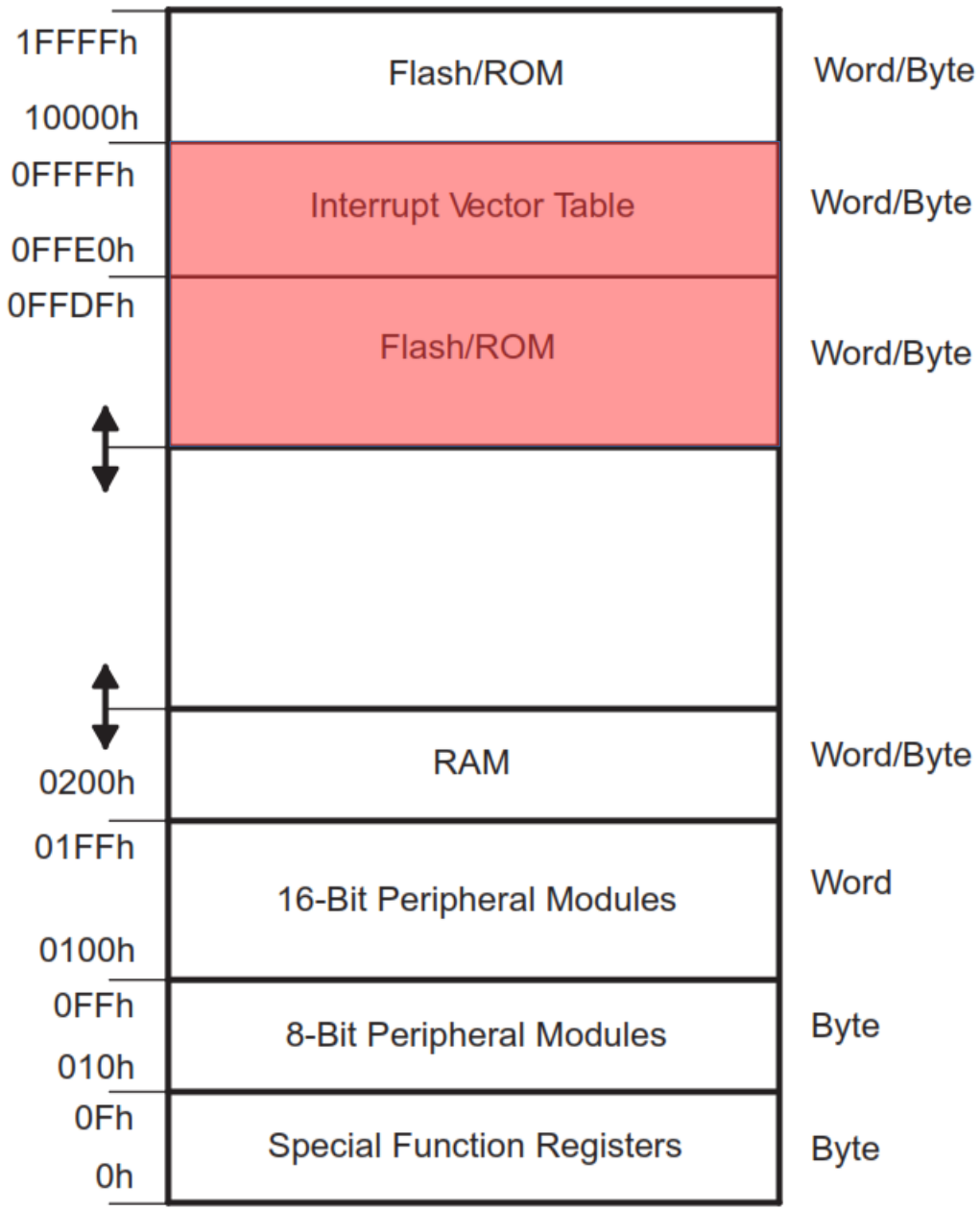
MSP430 – Adressraum Flash

Flash / ROM

nicht flüchtig,
für Code und Daten

Größe ist Typ-abhängig

Flash > 0x10000 nur
bei Typen mit >60k Flash



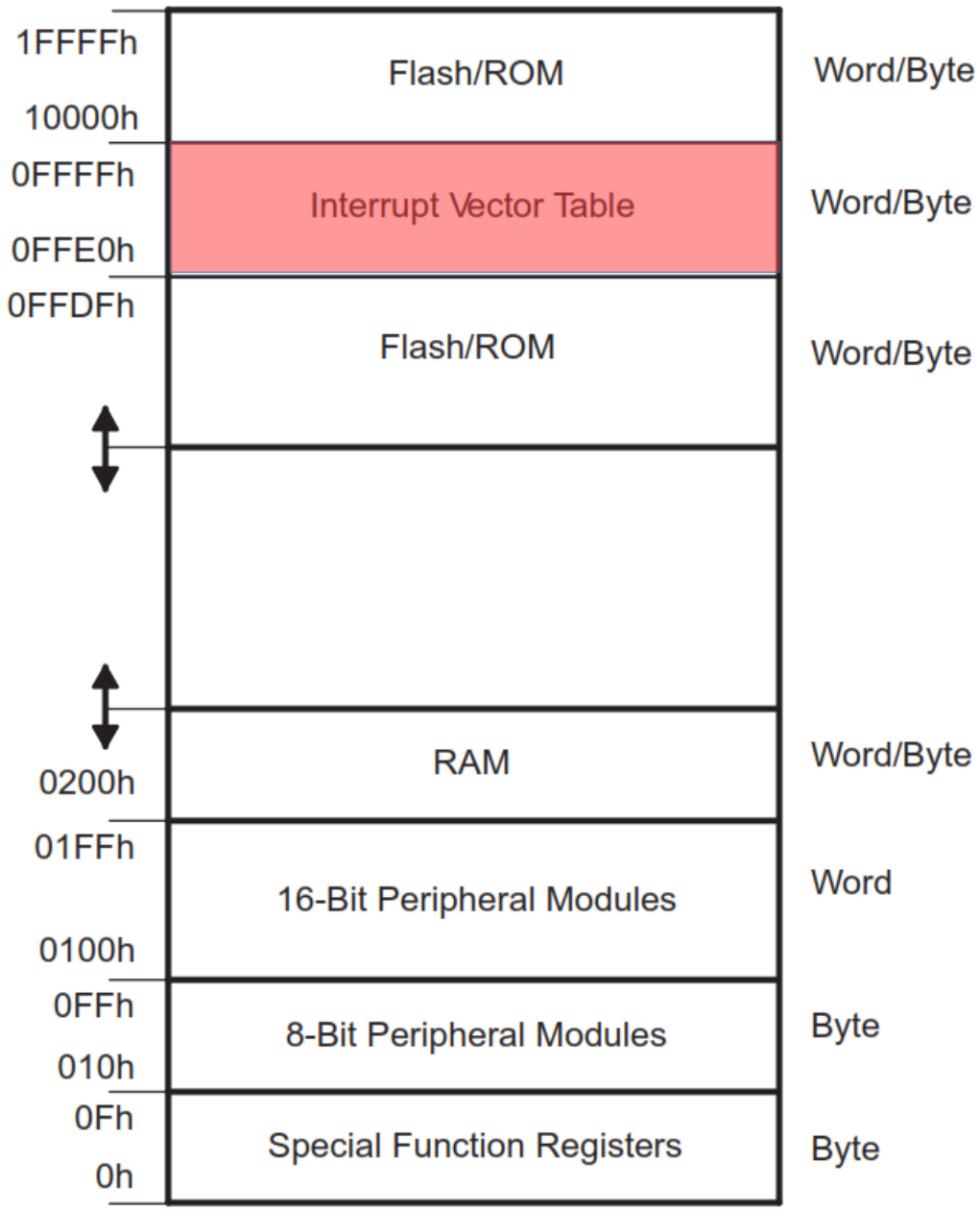
MSP430 – Adressraum ISR Vektoren

Flash / ROM:

endet bei 0xFFFF

erster Bereich Interrupt-Vektoren

G2553: 0xFFC0..FFFF



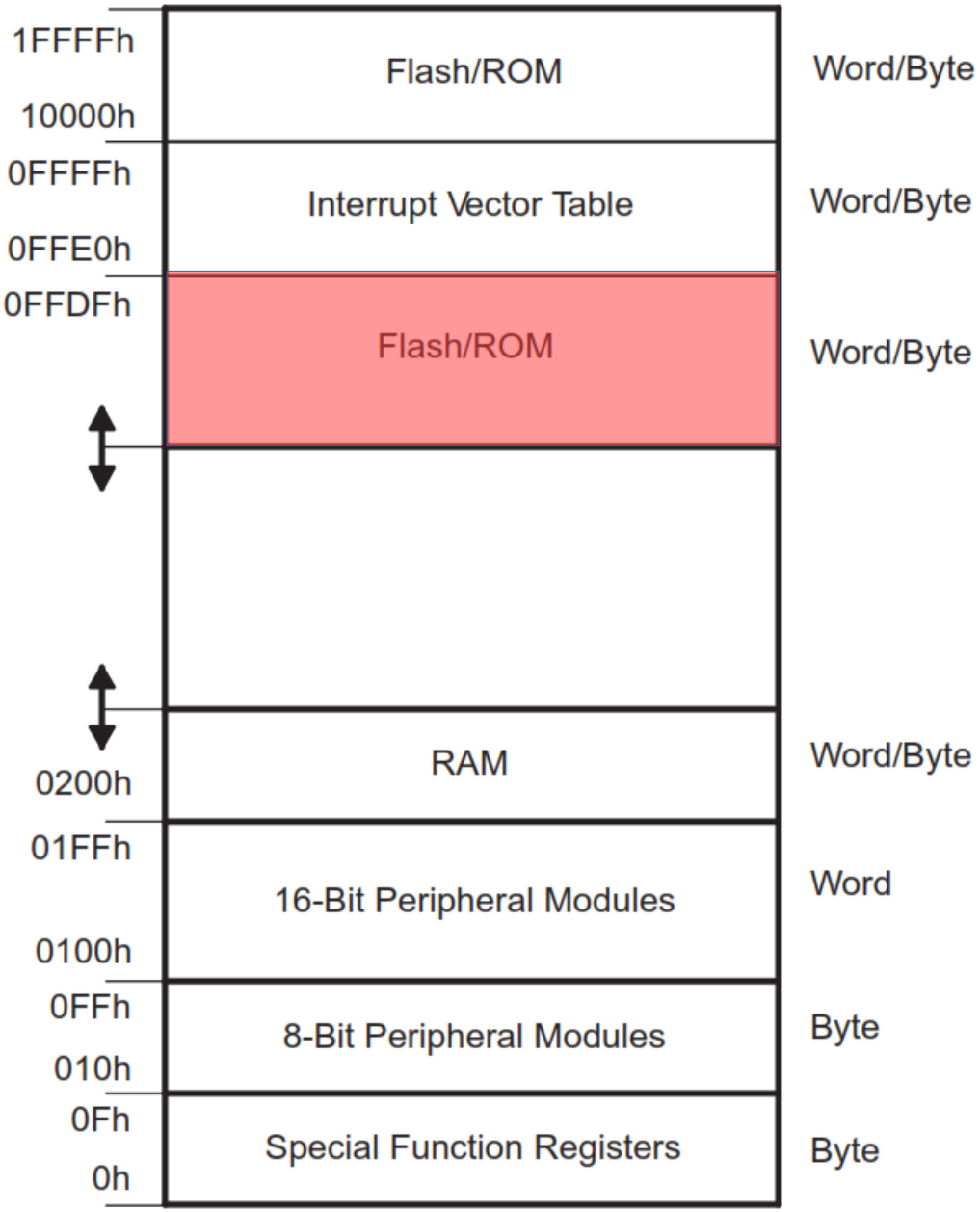
MSP430 – Adressraum Code/Daten

Flash / ROM:

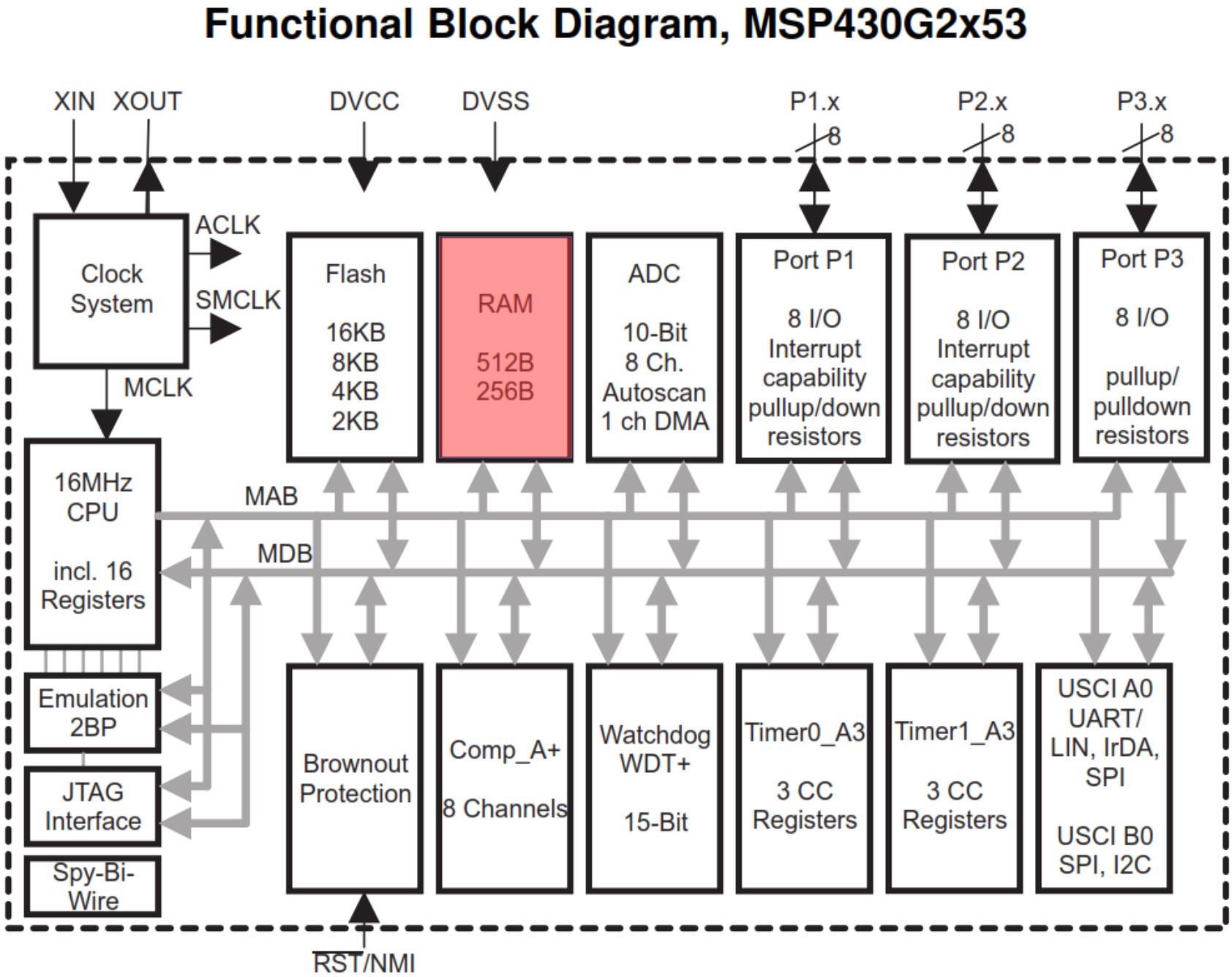
Startadresse/Beginn
nach Typ unterschiedlich
je nach Flashausstattung

G2553:

16kB Flash, daher
Start bei 0xC000



MSP430 – MSP430G2x53



MSP430 – Adressraum RAM

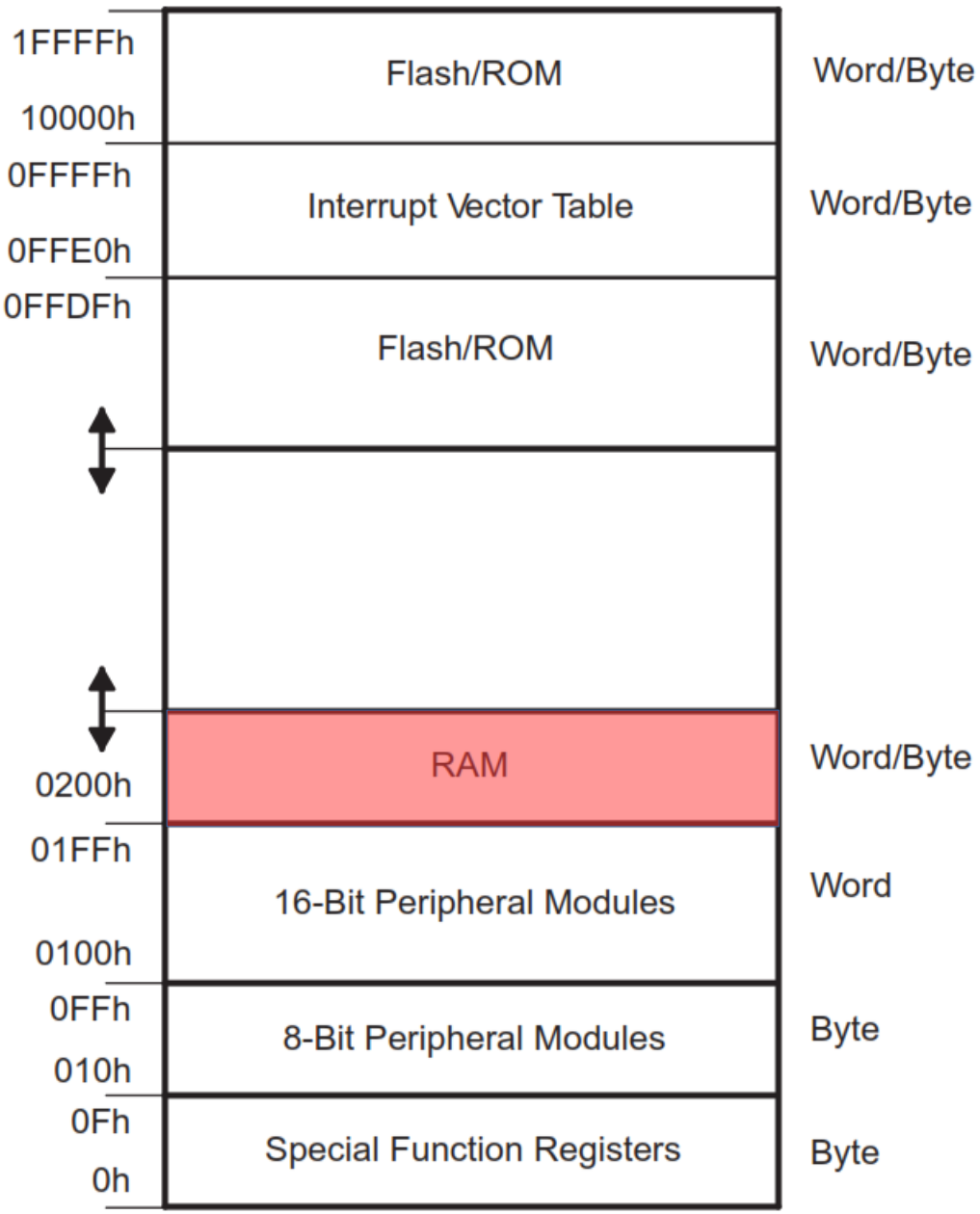
RAM:

Startadresse/Beginn
Immer 0x0200, Ende
je nach RAM Ausstattung

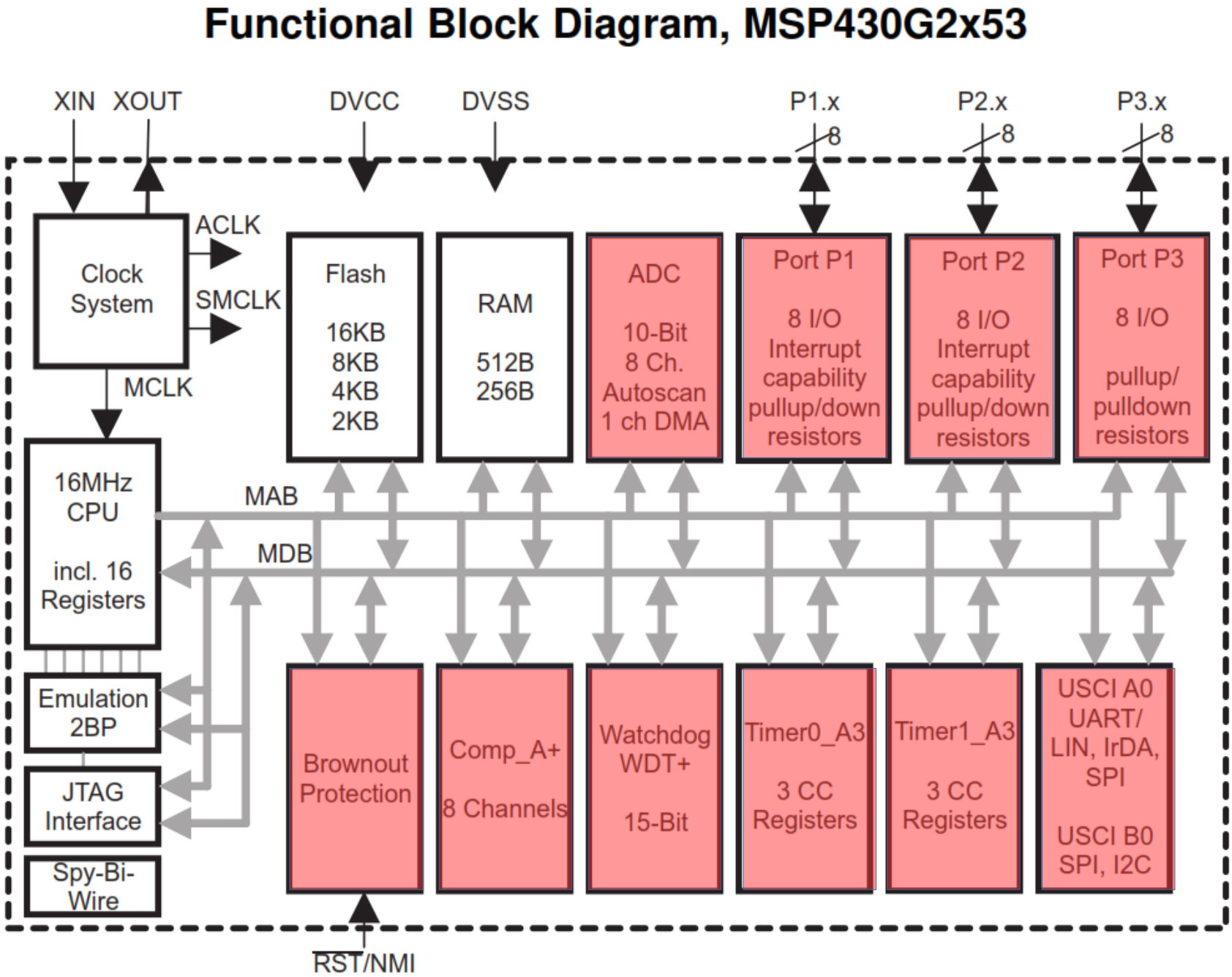
G2553:

512 Byte RAM, daher
Ende bei 0x03FF

→ flüchtiger Speicher!



MSP430 – MSP430G2x53

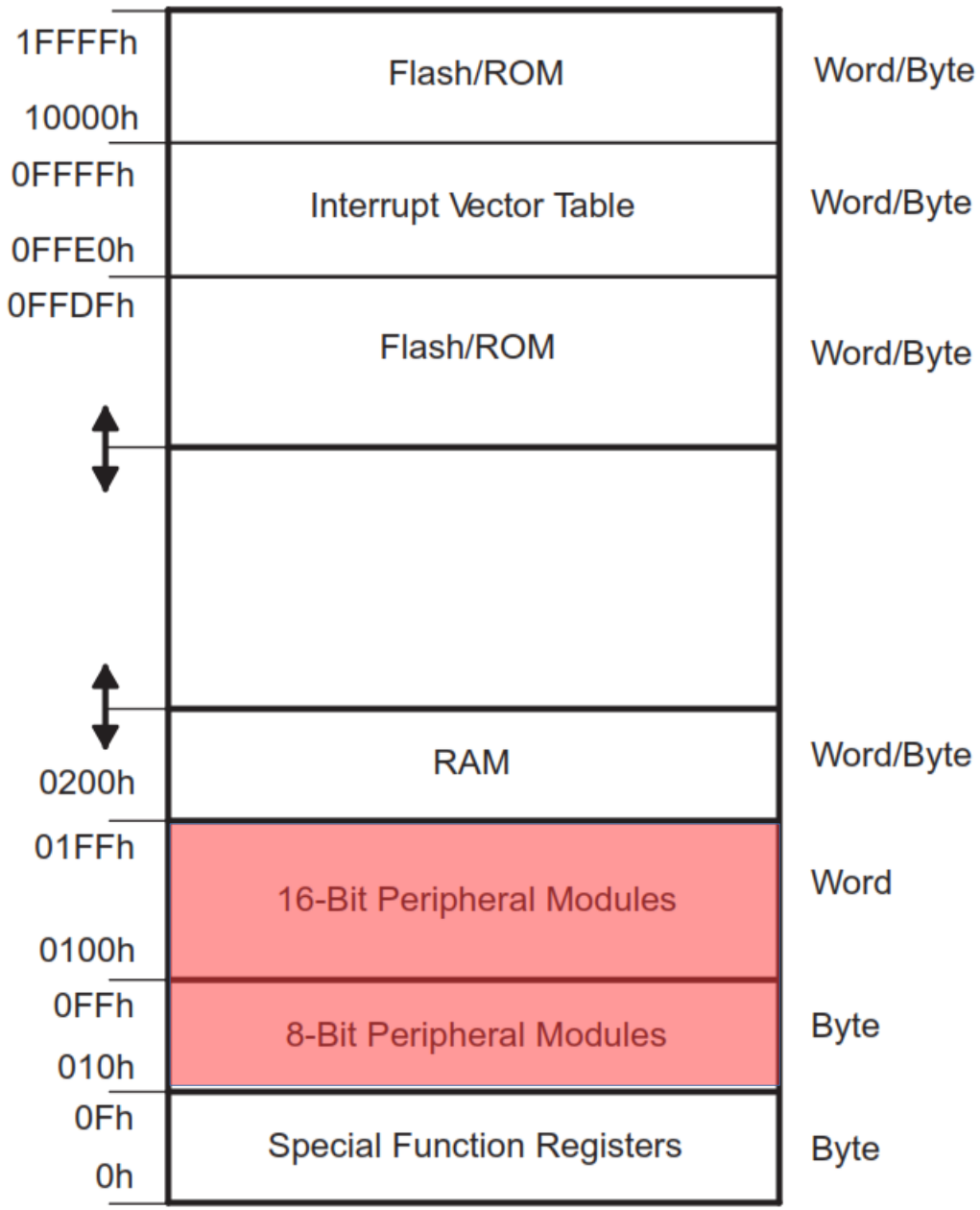


MSP430 – Adressraum Module

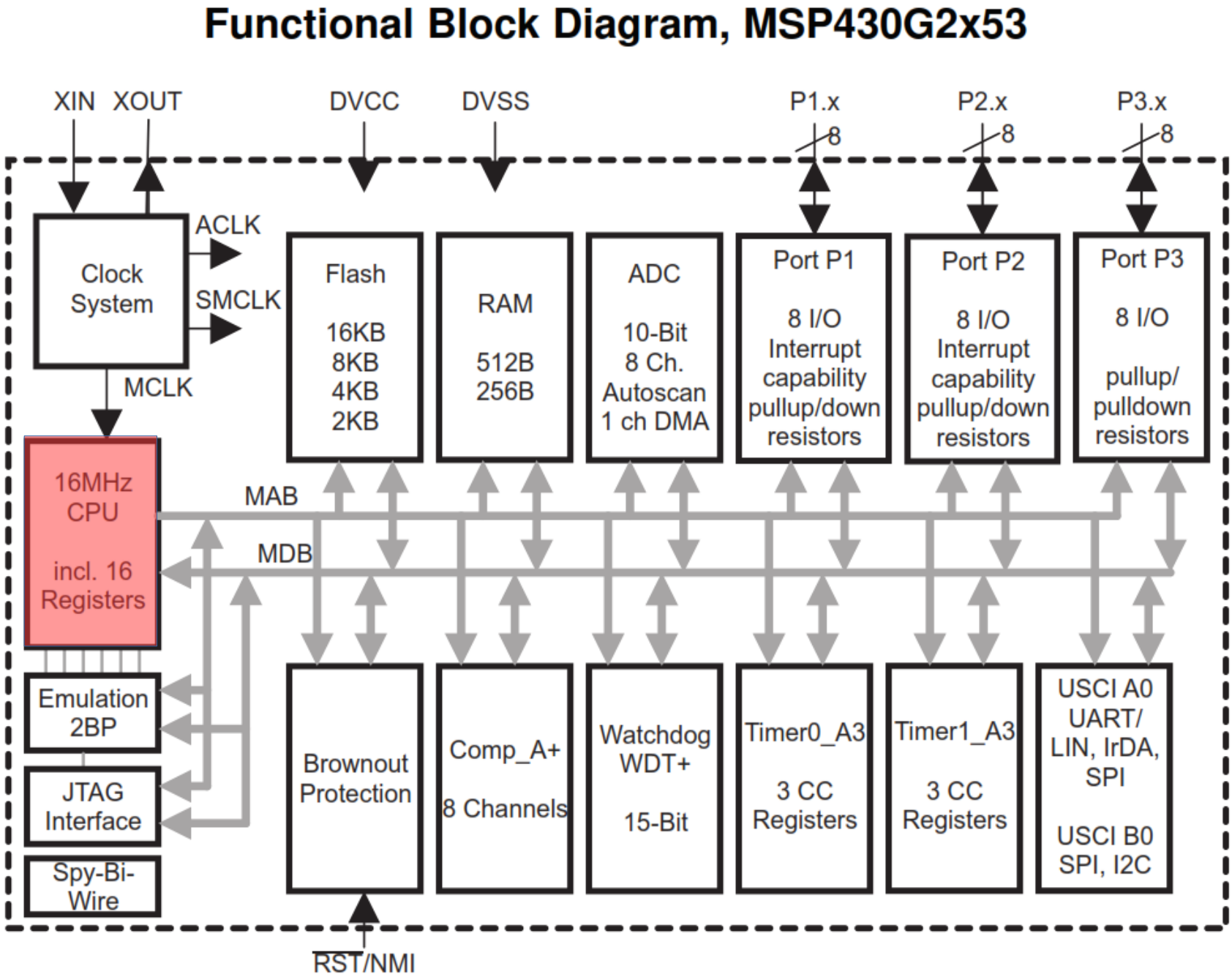
Peripheral Modules:

0x0010..01FF

- Ports
- Timer
- UART
- ADC
- Watchdog / Brownout
- Capture/Compare
- Analog



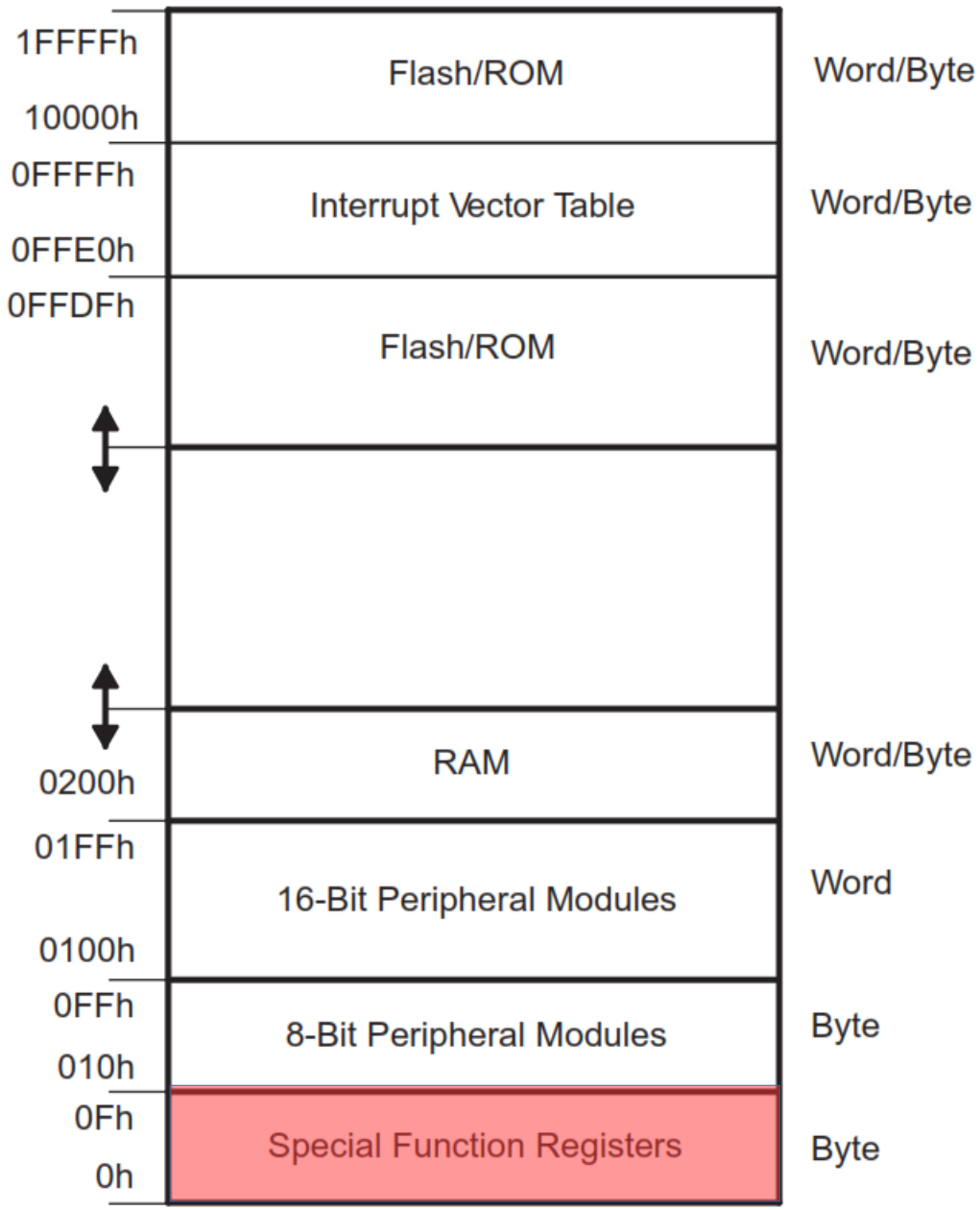
MSP430 – MSP430G2x53



MSP430 – Adressraum SFR/CPU

Special Function Register

0x0010..01FF

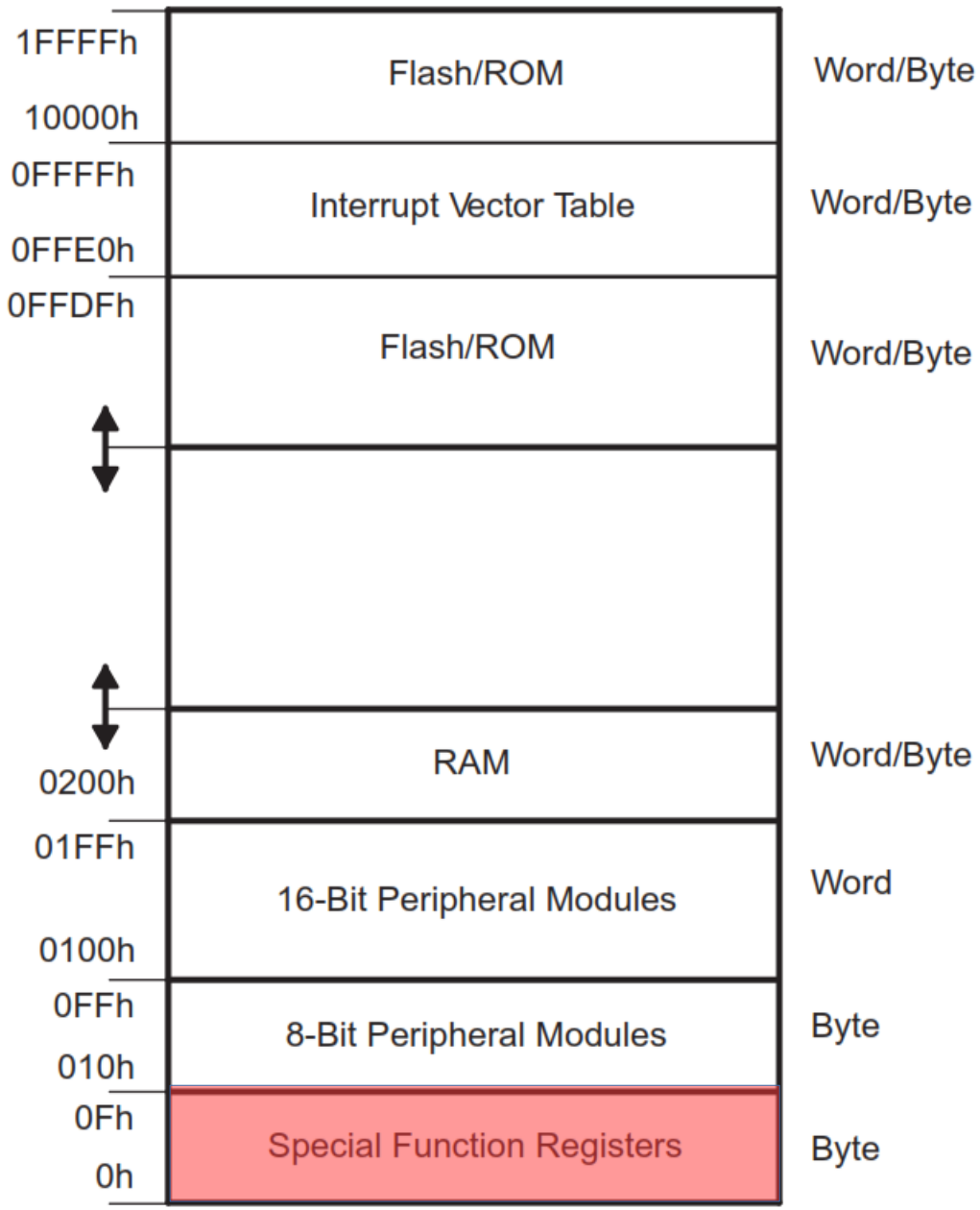


MSP430 – Adressraum SFR/CPU

Special Function Register

0x0010..01FF

Sind einen Blick mehr wert :)

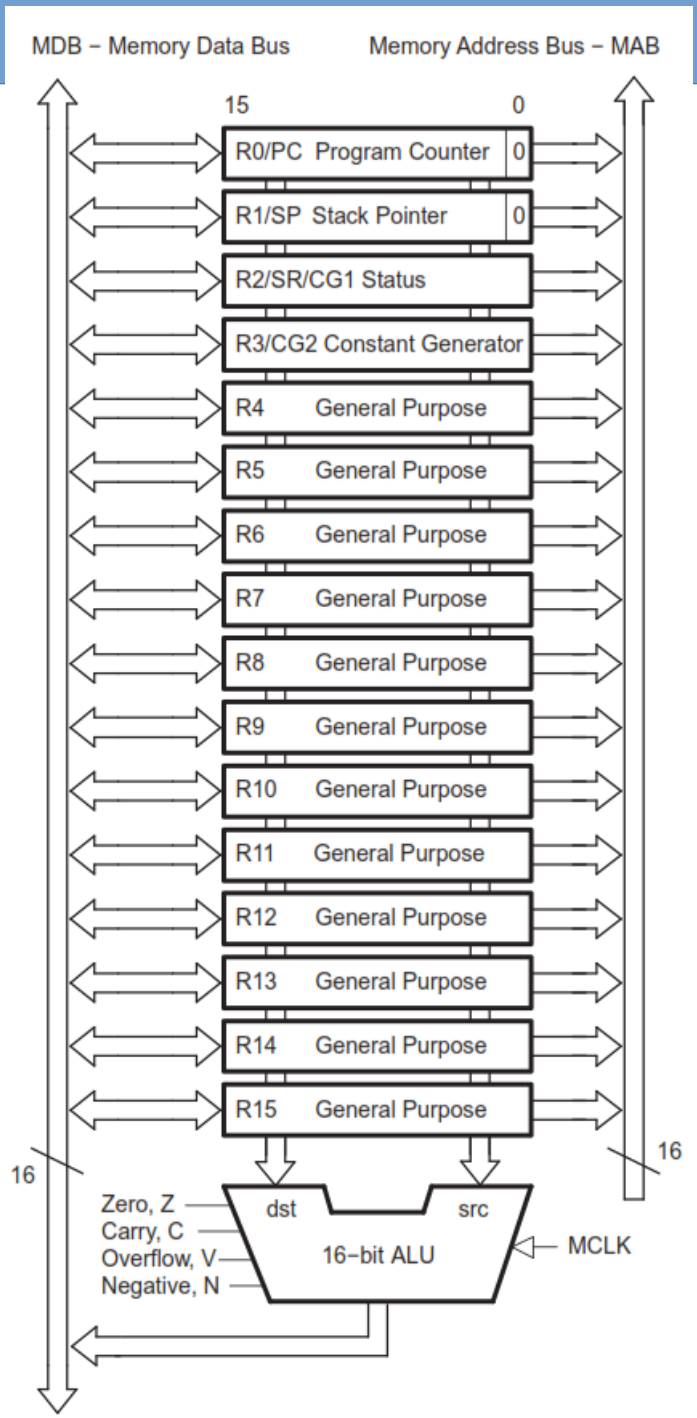


MSP430 - CPU

Die CPU besteht aus

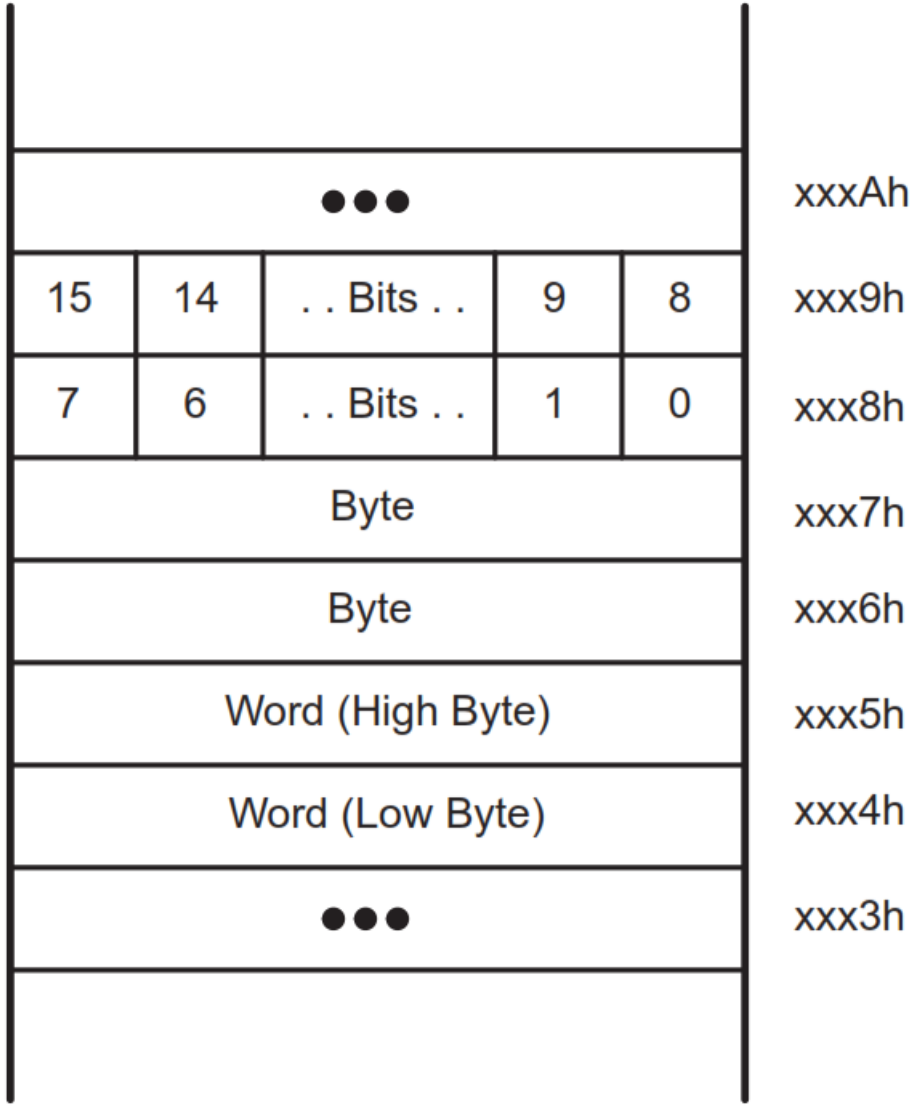
- 16 bit ALU
- Program Counter (R0)
- Stack Pointer (R1)
- Flags (R2)
- Constant Generator (R3)
- 12 Registern

- *alle haben 16 Bit*
- *Data und Adress-Bus*



MSP430 - Speicherzugriff

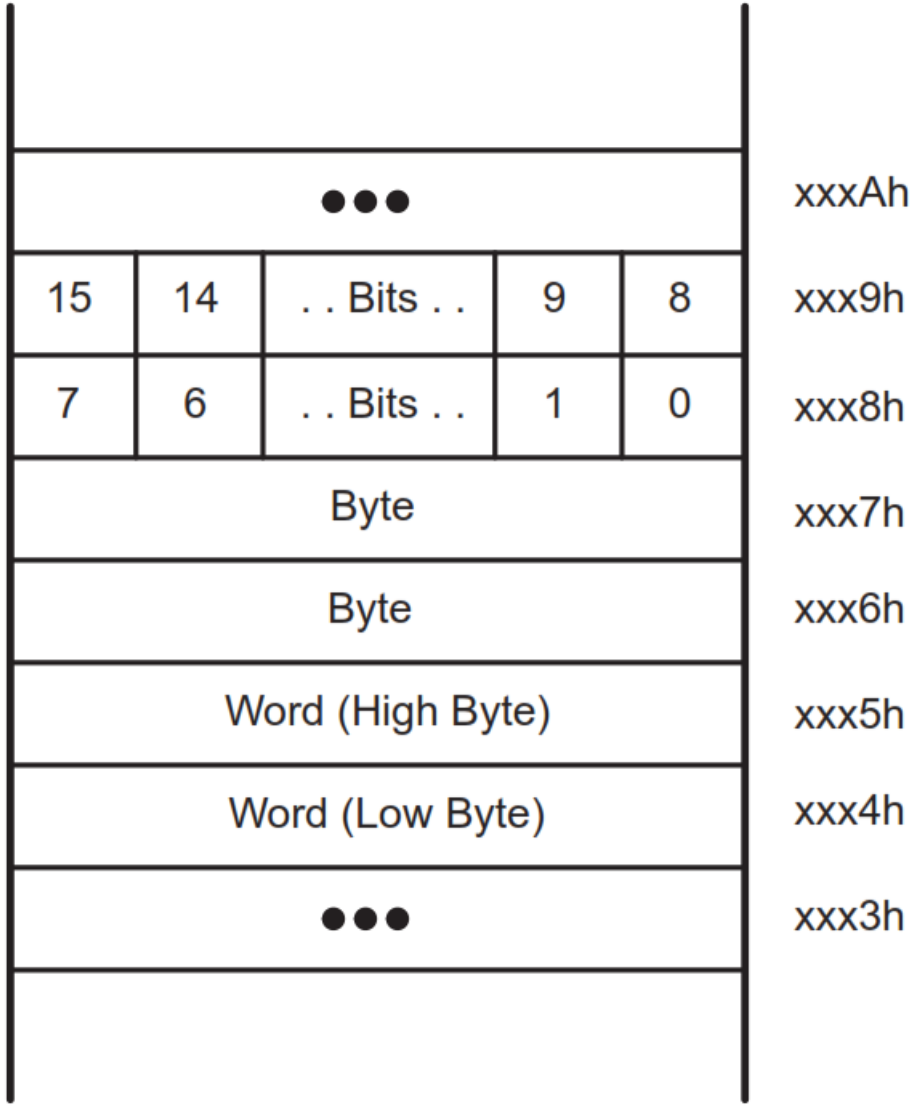
MSP430:
16bit Architektur



MSP430 - Speicherzugriff

MSP430:
16bit Architektur

Speicher 8bit = 1 Byte
Worte = 2 Byte = 16bit

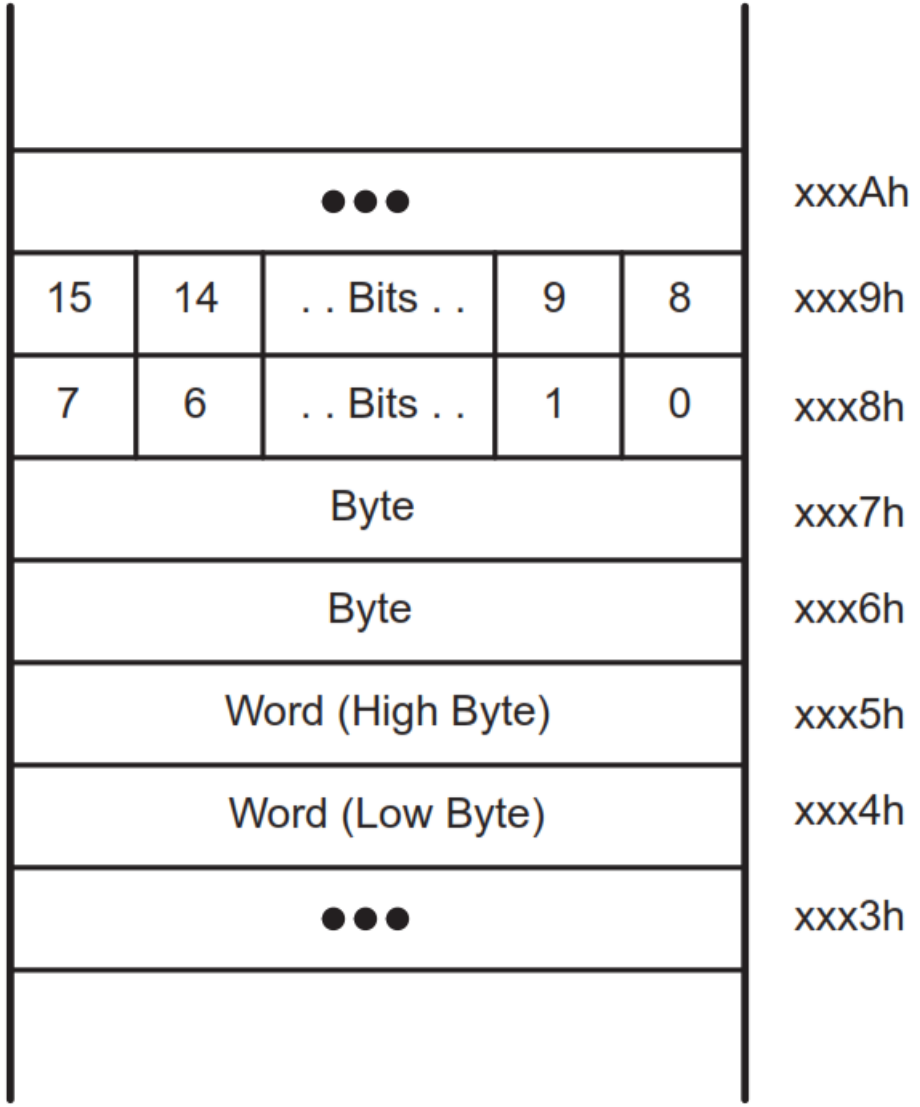


MSP430 - Speicherzugriff

MSP430:
16bit Architektur

Speicher 8bit = 1 Byte
Worte = 2 Byte = 16bit

Worte beginnen immer
Auf geraden Adressen



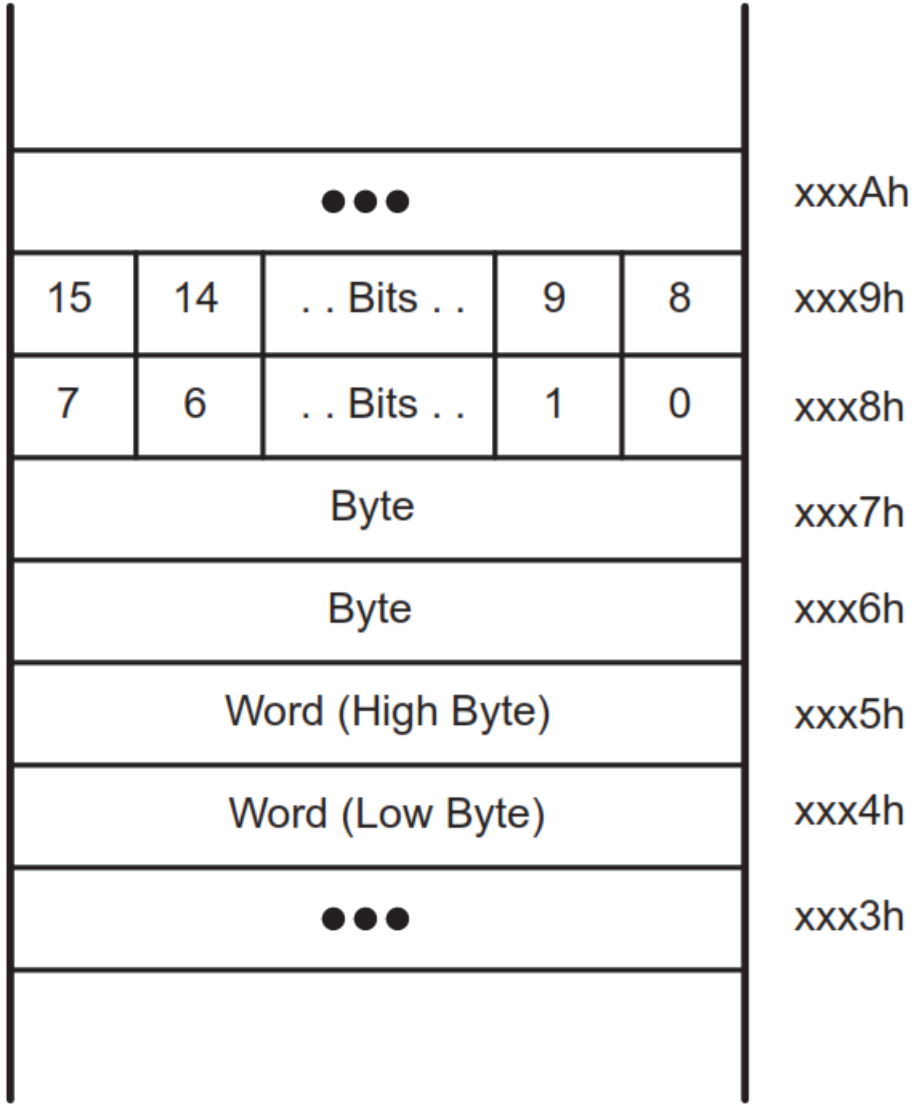
MSP430 - Speicherzugriff

MSP430:
16bit Architektur

Speicher 8bit = 1 Byte
Worte = 2 Byte = 16bit

Worte beginnen immer
Auf geraden Adressen

Zugriff auch als Byte



MSP430 - Speicherzugriff

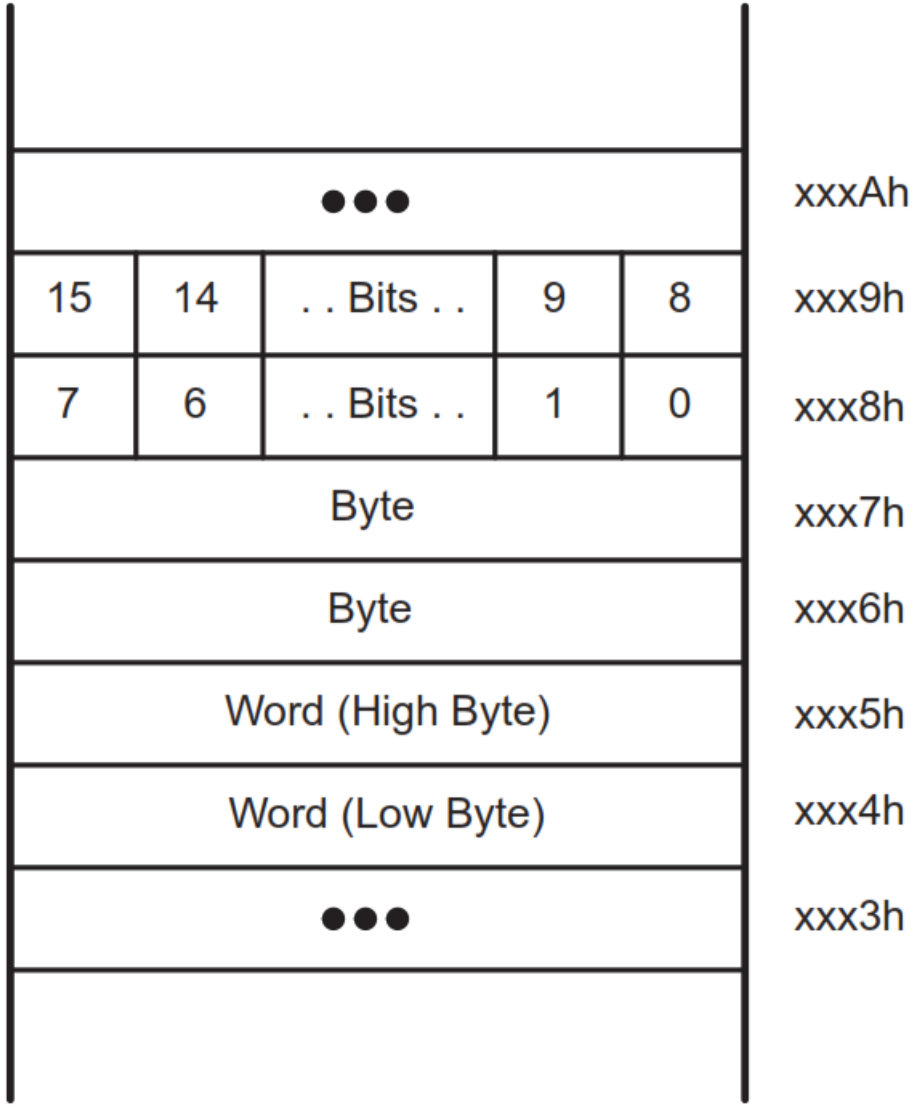
MSP430:
16bit Architektur

Speicher 8bit = 1 Byte
Worte = 2 Byte = 16bit

Worte beginnen immer
Auf geraden Adressen

Zugriff auch als Byte

Teils auch als Bit



MSP430 – Blinkenlights

Lesson 1:

- delay über for-Schleife und counter
- einfach und gut nachvollziehbar
- kann aber einem Opimizer zum Opfer fallen
- CPU ist busy, kein genaues Timing

MSP430 – Blinkenlights

Lesson 2:

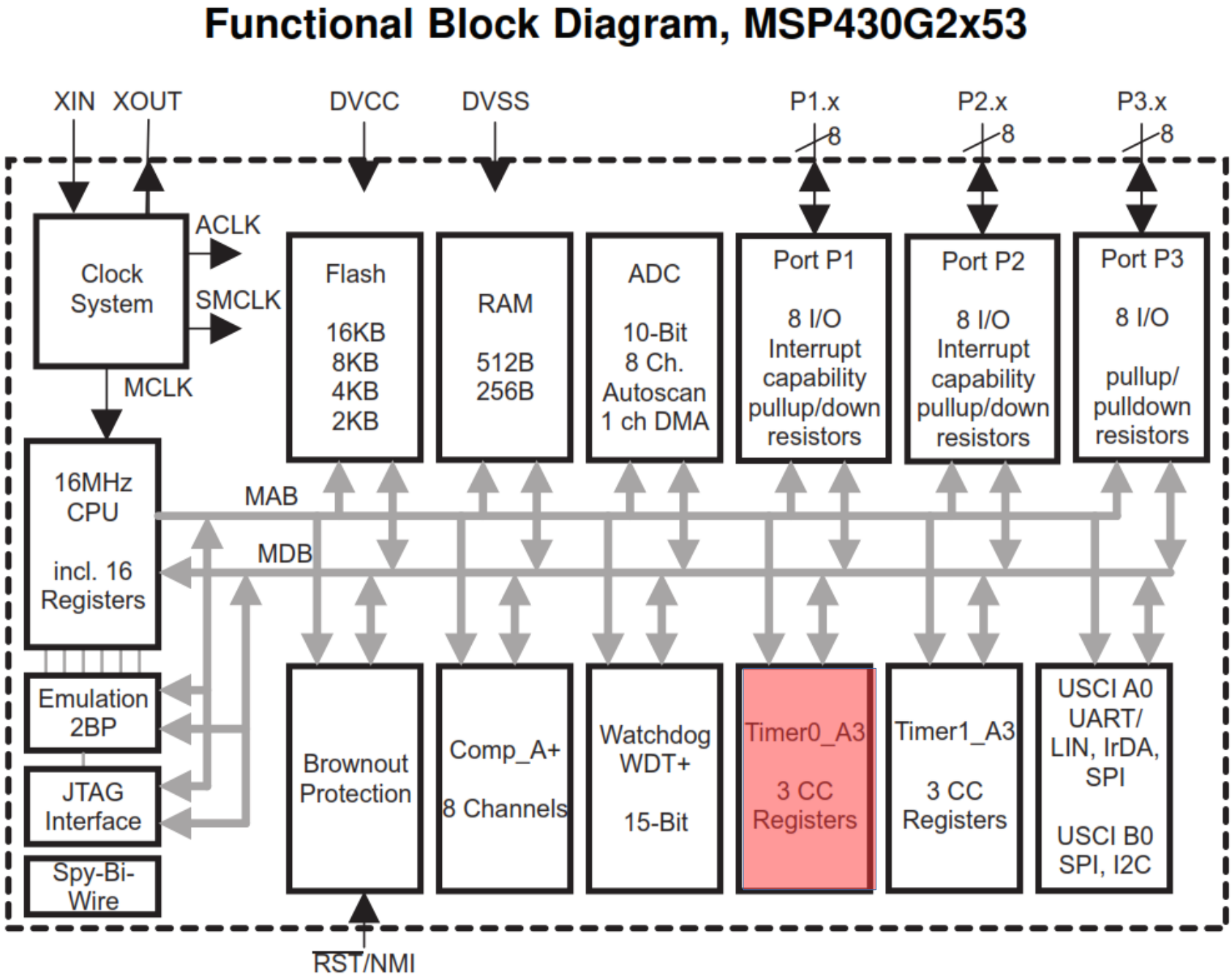
- delay über „intrinsic“
- klarer was passieren soll
- schnelleres Blinken – warum?
- Optimizer-safe
- CPU ist busy, aber immerhin „genaues“ Timing

MSP430 – Blinkenlights

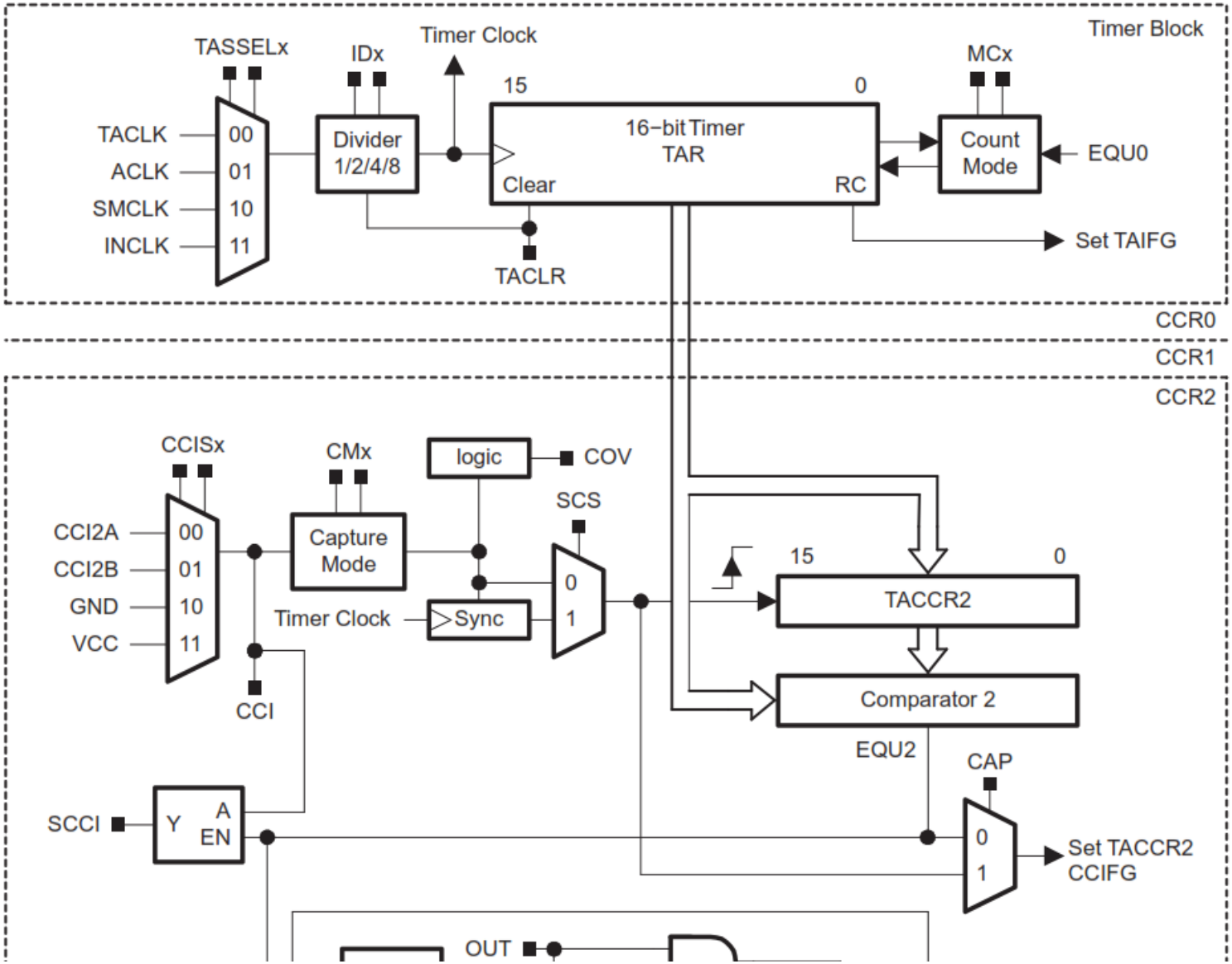
Lesson 3:

- delay über TIMER_A
- Warum Software wenn es dafür Hardware gibt?
- Optimizer-safe
- CPU ist für anderes frei
- sehr genaues Timing

MSP430 – TIMER_A



MSP430 – TIMER_A



MSP430 – TIMER_A

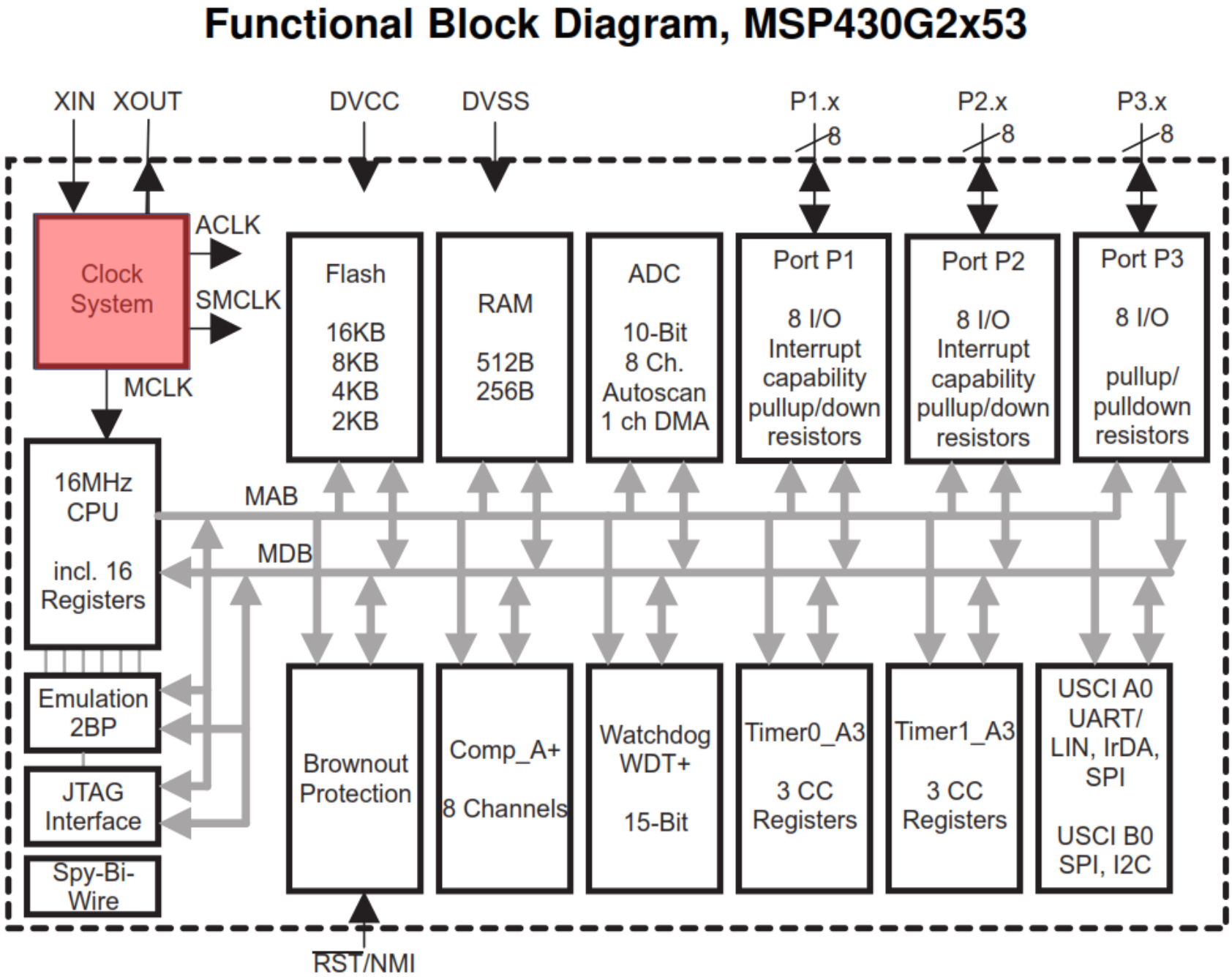
Das sieht kompliziert aus!

Viele Register und Flags im Spiel, viel zu konfigurieren…

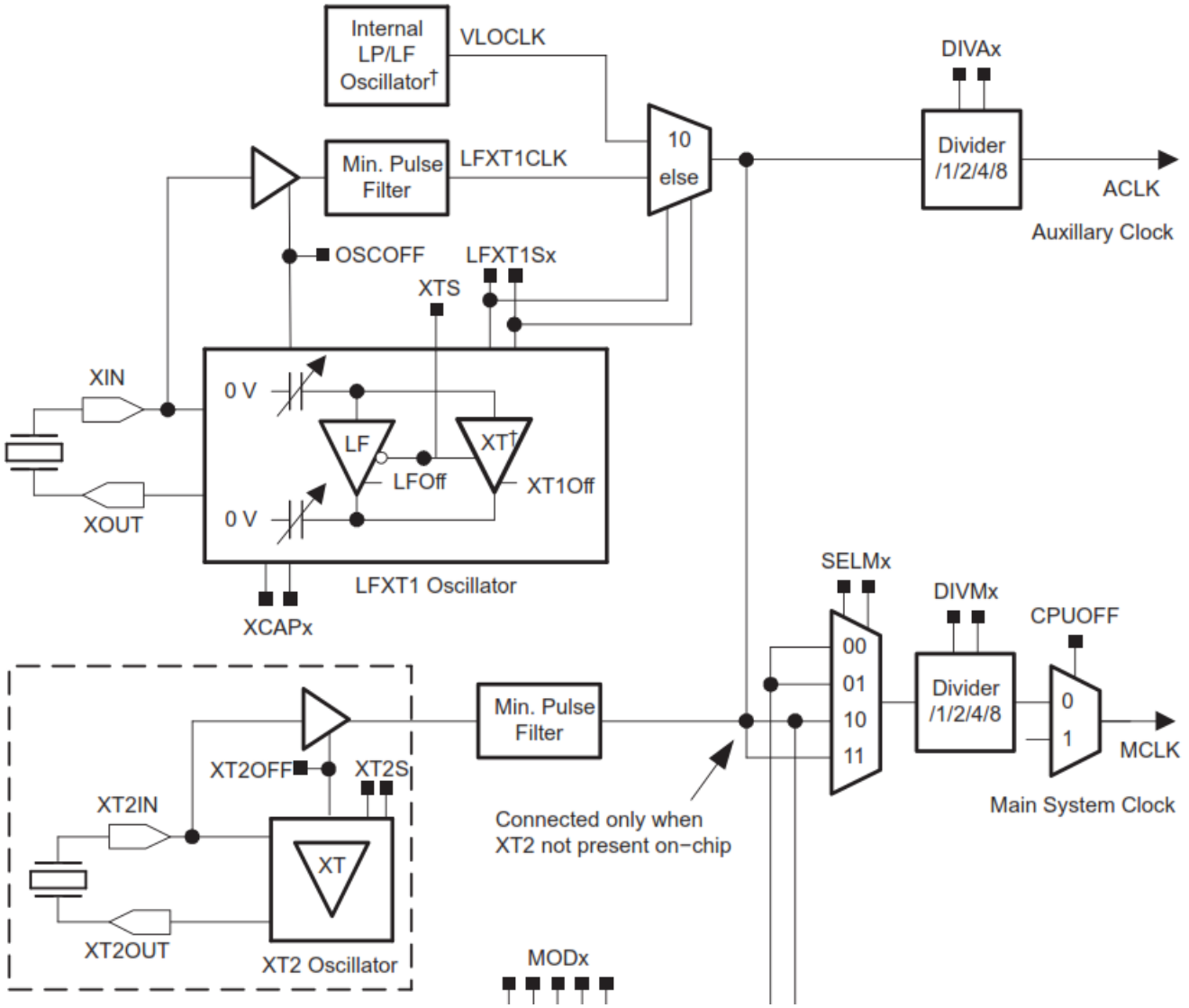
… für einen Timer!

Und das ist nicht alles...

MSP430 – Basic Clock Module+



MSP430 – Basic Clock Module+



MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
```

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
DCOCTL = 0x00;
```

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
DCOCTL = 0x00;
BCSCTL1 = CALBC1_1MHZ;
```

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
DCOCTL = 0x00;
BCSCTL1 = CALBC1_1MHZ;
DCOCTL = CALDCO_1MHZ;
```

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
DCOCTL = 0x00;
BCSCTL1 = CALBC1_1MHZ;
DCOCTL = CALDCO_1MHZ;

BCSCTL1 |= XT2OFF | DIVA_0;
```

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
DCOCTL = 0x00;
BCSCTL1 = CALBC1_1MHZ;
DCOCTL = CALDCO_1MHZ;

BCSCTL1 |= XT2OFF | DIVA_0;
BCSCTL3 = XT2S_0 | LFXT1S_0 | XCAP_1;
```

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
DCOCTL = 0x00;
BCSCTL1 = CALBC1_1MHZ;
DCOCTL = CALDCO_1MHZ;

BCSCTL1 |= XT2OFF | DIVA_0;
BCSCTL3 = XT2S_0 | LFXT1S_0 | XCAP_1;

TACCTL0 = CM_0 | CCIS_0 | OUTMOD_0 | CCIE;
```

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
DCOCTL = 0x00;
BCSCTL1 = CALBC1_1MHZ;
DCOCTL = CALDCO_1MHZ;

BCSCTL1 |= XT2OFF | DIVA_0;
BCSCTL3 = XT2S_0 | LFXT1S_0 | XCAP_1;

TACCTL0 = CM_0 | CCIS_0 | OUTMOD_0 | CCIE;
TACCR0 = 32767;
```

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
BCSCTL2 = SELM_0 | DIVM_0 | DIVS_0;
DCOCTL = 0x00;
BCSCTL1 = CALBC1_1MHZ;
DCOCTL = CALDCO_1MHZ;

BCSCTL1 |= XT2OFF | DIVA_0;
BCSCTL3 = XT2S_0 | LFXT1S_0 | XCAP_1;

TACCTL0 = CM_0 | CCIS_0 | OUTMOD_0 | CCIE;
TACCR0 = 32767;
TACTL = TASSEL_1 | ID_0 | MC_1;
```

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
CCIE = Capture Compare Interrupt Enable
```

Vorgriff auf Interrupts

– kommt nochmal genauer :)

MSP430 – Basic Clock Module+

… und noch mehr Register und Flags!

```
CCIE = Capture Compare Interrupt Enable
```

TACCR0 Interruptvektor für TACCR0 CCIFG

TAIV Interruptvektor für alle anderen Flags

MSP430 – Basic Clock Module+

… Code needed!

MSP430 – Basic Clock Module+

Thanks!

MSP430 – Übersicht allgemein

Code needed..