

Gliederung zur Mikrocomputertechnik

1. Einleitung und Motivation
2. **Entwicklung mit Mikrocontrollern**
 - 2.1 Die Entwicklungsumgebung
 - 2.2 LaunchPad MSP-EXP430G2
 - 2.3 Code Composer Studio und erstes Programm
 - 2.4 Überblick über den Aufbau des MSP430**
 - 2.4.1 MSP430 Pinbelegung (Blick von außen)**
 - 2.4.2 MSP430 Blockdiagramm (Blick nach innen)**
 - 2.5 Programmierung und Debugging

Pinbelegung des MSP430G2231 (1)

Device	BSL	EEM	Flash (KB)	RAM (B)	Timer_A	USI	ADC10 Channel	Clock	I/O	Package Type ⁽²⁾
MSP430G2231IRSA16 MSP430G2231IPW14 MSP430G2231IN14	-	1	2	128	1x TA2	1	8	LF, DCO, VLO	10	16-QFN 14-TSSOP 14-PDIP

◆ G2231 Baustein, Package Optionen:

□ 14-Pin „Plastic Dual-In-Line package“ (PDIP)

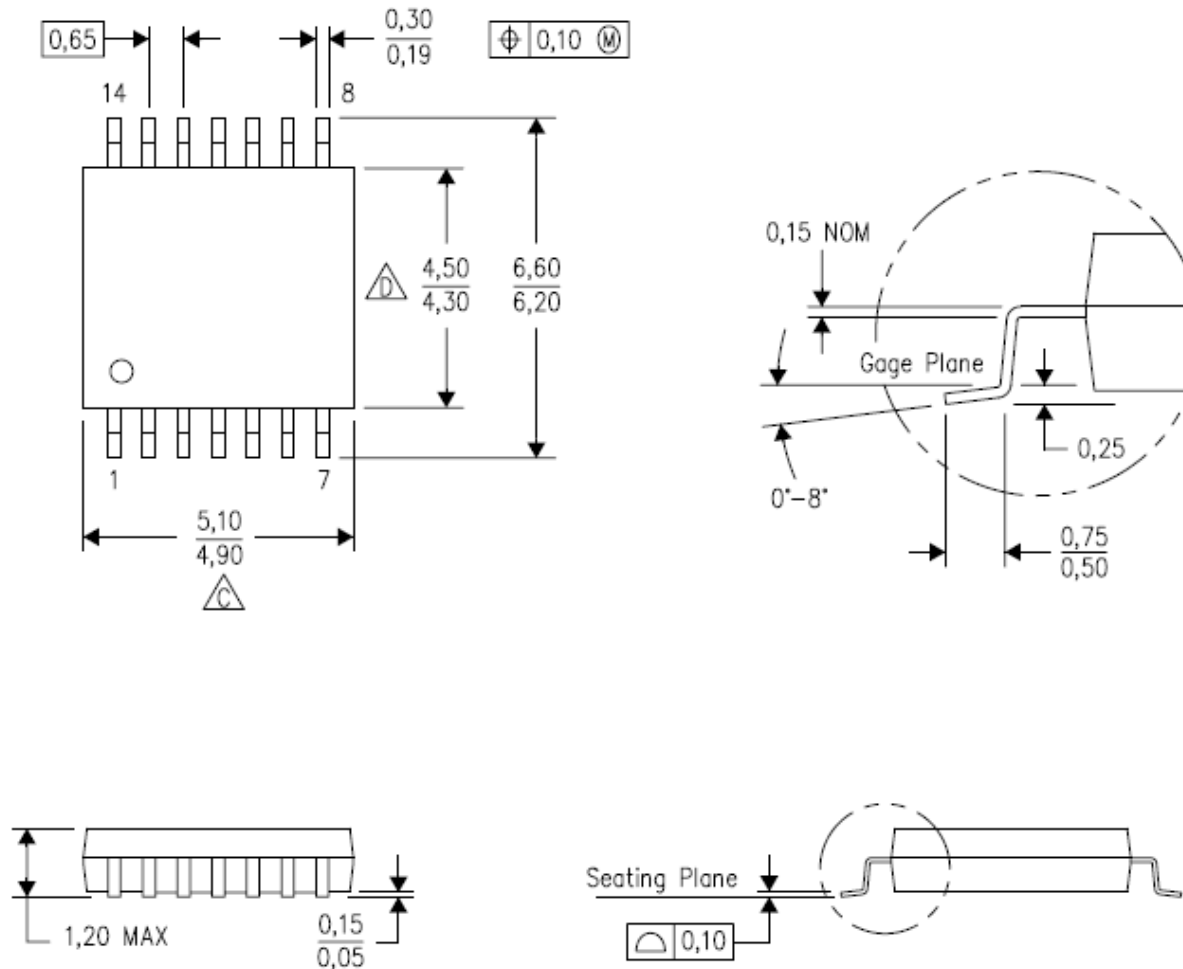
Abstand zwischen den Anschlüssen beträgt 0,1 inch \approx 2,5mm
– leicht zu löten und gut zum Basteln

□ TSSOP (Thin-Shrink Small Outline Package) verfügbar, (surface-mount device), aber 0,65mm Pin-Abstand (Rastermaß für die Pins)

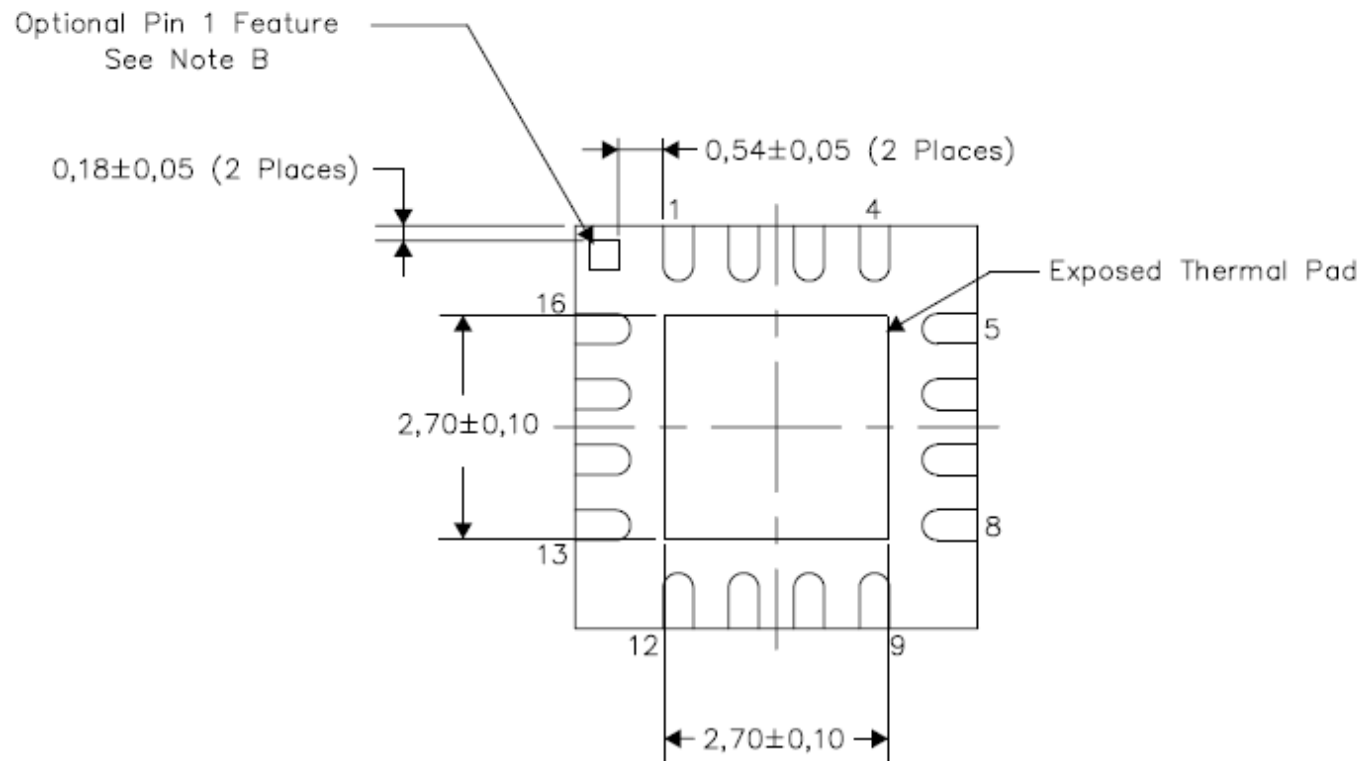
□ QFN (Quad Flatpack No-lead) , keine „Anschlussstifte“, „pads“ mit 0,65mm Abstand

Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

TSSOP-Package



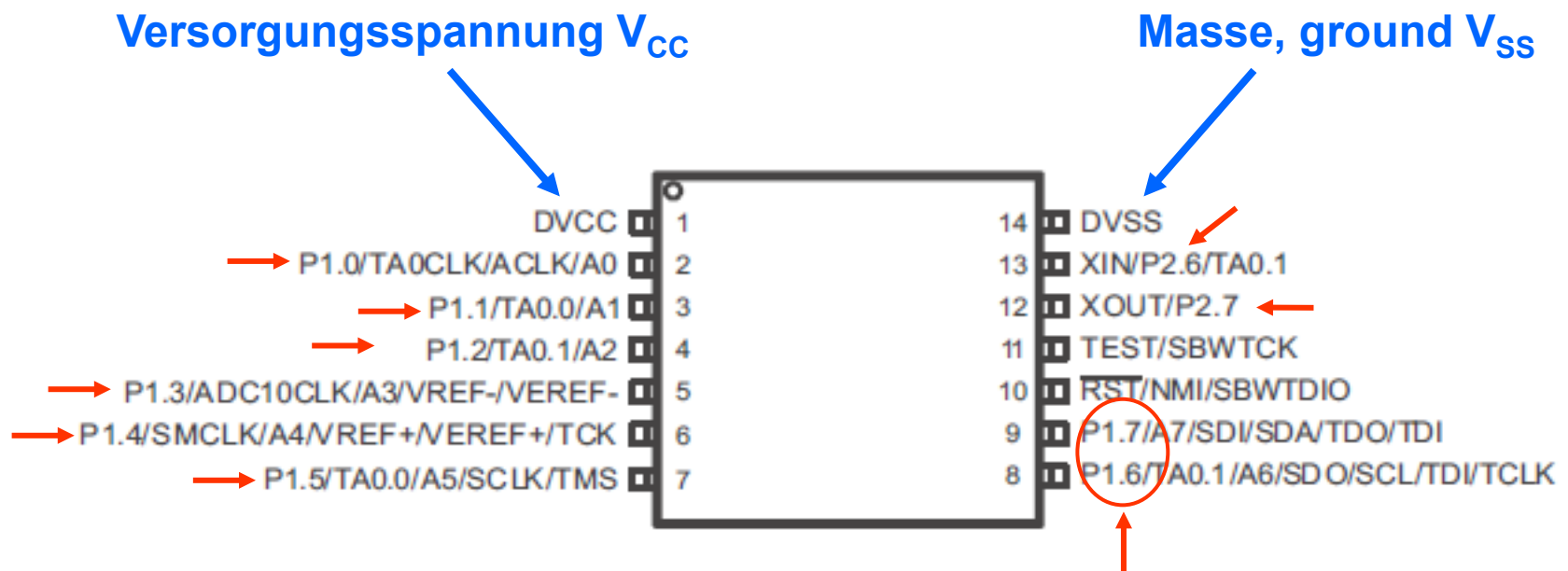
QFN-Package



Bottom View

Pinbelegung des MSP430G2231 (2)

- ◆ G2231 Baustein – hier 14-Pin „Plastic Dual-In-Line package“ (PDIP)
 - bemerkenswert: Anschlüsse haben verschiedene Funktionen

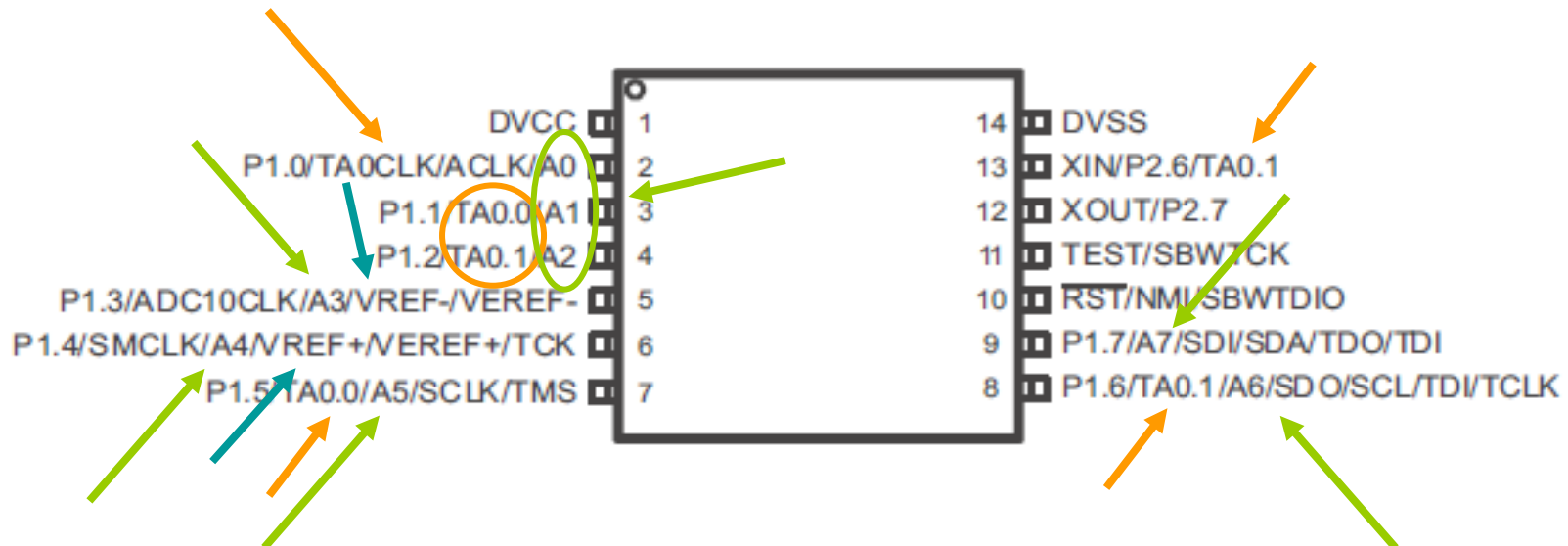


10 digitale Ein- und Ausgänge auf zwei Ports 1 und 2 verteilt: P1.0 bis 1.7 und P2.6 und P2.7, die unabhängig voneinander programmierbar sind.

Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

Pinbelegung des MSP430G2231 (3)

- ◆ **Timer Funktion:** **TA0CLK** Takteingang, **TA0.0** und **TA0.1** sind als Ausgänge verfügbar an verschiedenen Anschlüssen aufgrund der Bedeutung des Timers
- ◆ **8 analoge Eingänge A0 bis A7** für Analog-Digital-Umsetzer
 - **VREF+/-** sind wählbare, positive bzw. negative Anschlussmöglichkeiten für externe Referenzspannungen

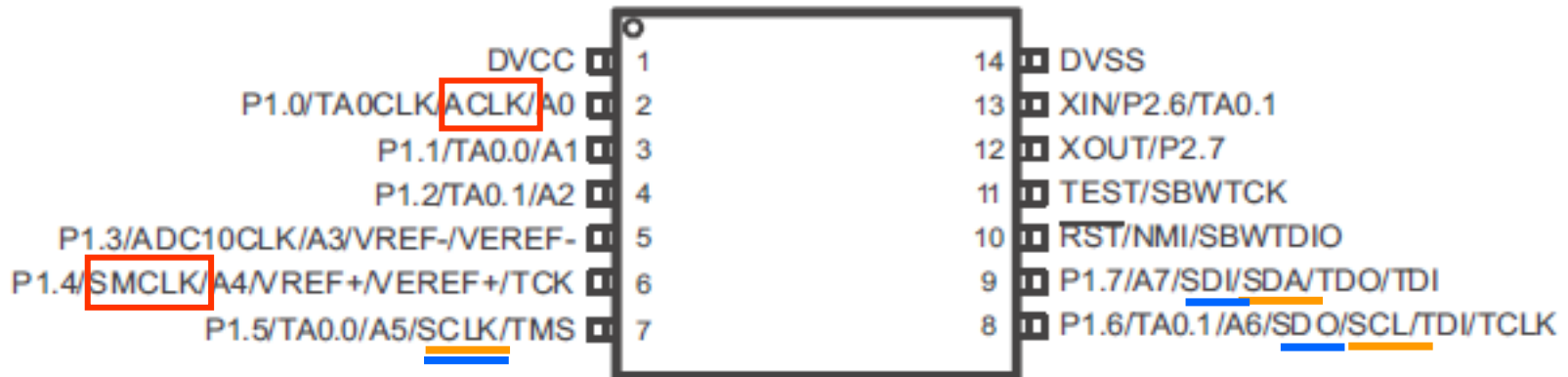


Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

Pinbelegung des MSP430G2231 (4)

◆ Takt und Kommunikationsschnittstellen

- **ACLK, SMCLK** – Taktausgänge des Mikrocontrollers
- I2C Modus: **SCLK** – clock input, **SCL** – I2C clock, **SDA** – I2C data
- SPI Modus: **SCLK** – SPI clock in-/output ,
SDO – SPI data output, **SDI** – SPI data input

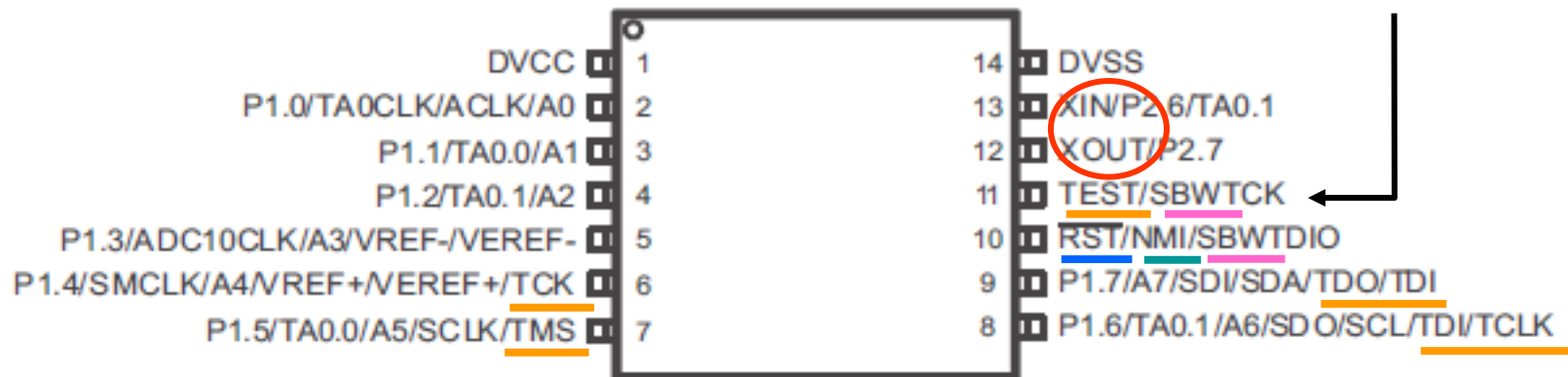


Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

Pinbelegung des MSP430G2231 (5)

- ◆ **XIN, XOUT** Anschlüsse für einen externen Quarz
- ◆ **RST** ist ein Reset-Signal mit negativer Logik (bRST, /RST, _RST)
- ◆ **NMI** – nicht maskierbarer Interrupt-Eingang
- ◆ **TCK, TMS, TCLK, TDI, TDO, TEST** sind Anschlüsse, die das JTAG-Interface bilden und mit denen der Mikrocontroller programmiert wird.
- ◆ **SBW** = Spy-Bi-Wire, JTAG-Interface mit nur zwei Anschlüssen (von TI)

TEST/SBWTCK ist neben V_{CC} und V_{SS} der einzige, nicht-programmierbare Anschluß

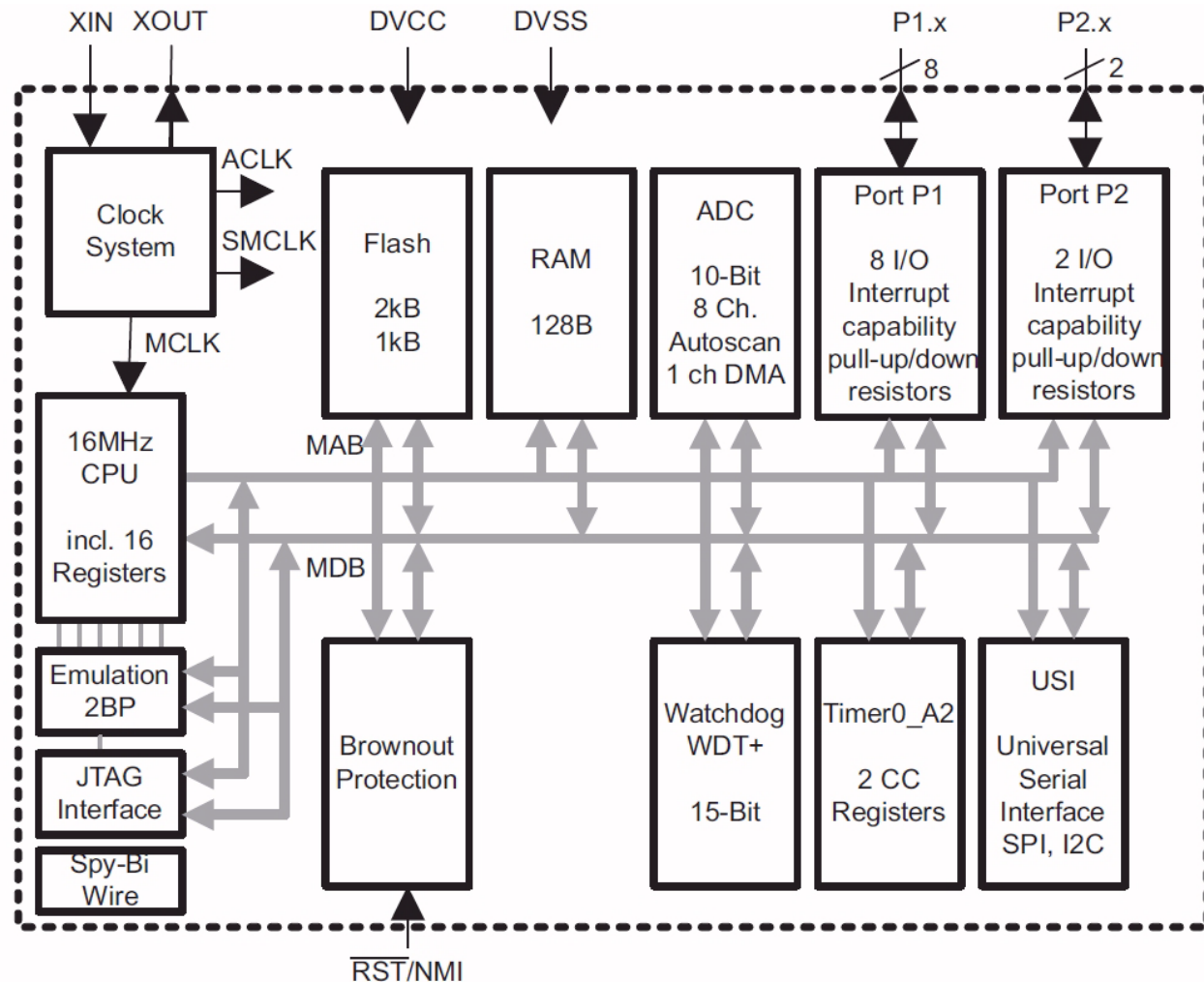


Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

Gliederung zur Mikrocomputertechnik

1. Einleitung und Motivation
2. **Entwicklung mit Mikrocontrollern**
 - 2.1 Die Entwicklungsumgebung
 - 2.2 LaunchPad MSP-EXP430G2
 - 2.3 Code Composer Studio und erstes Programm
 - 2.4 Überblick über den Aufbau des MSP430**
 - 2.4.1 MSP430 Pinbelegung (Blick von außen)
 - 2.4.2 MSP430 Blockdiagramm (Blick nach innen)**

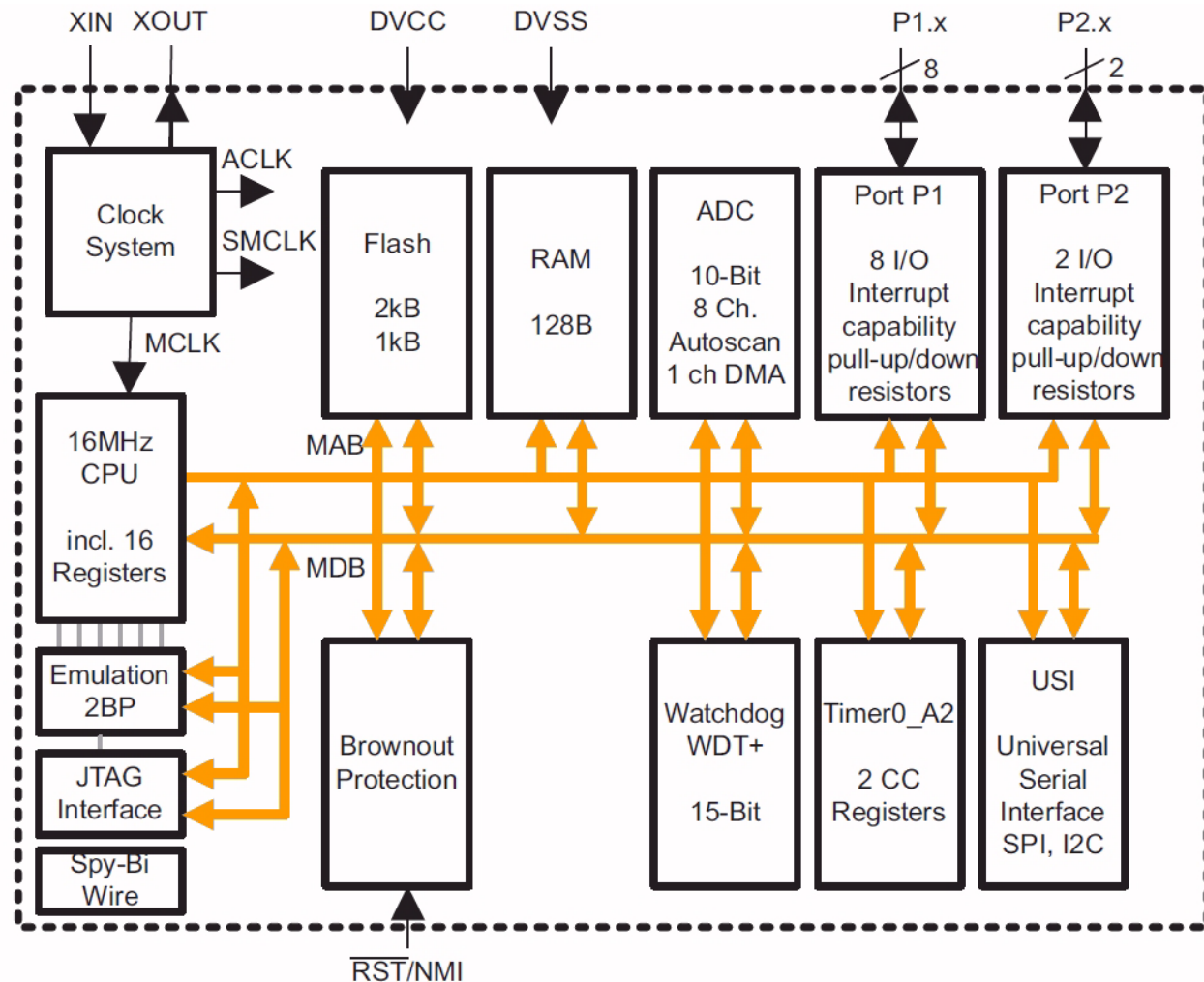
Block Diagramm des MSP430G2231



Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

Block Diagramm des MSP430G2231

- ◆ **Schaltungsblöcke sind durch**
MAB
(memory address bus)
und **MDB**
(memory data bus)
verbunden

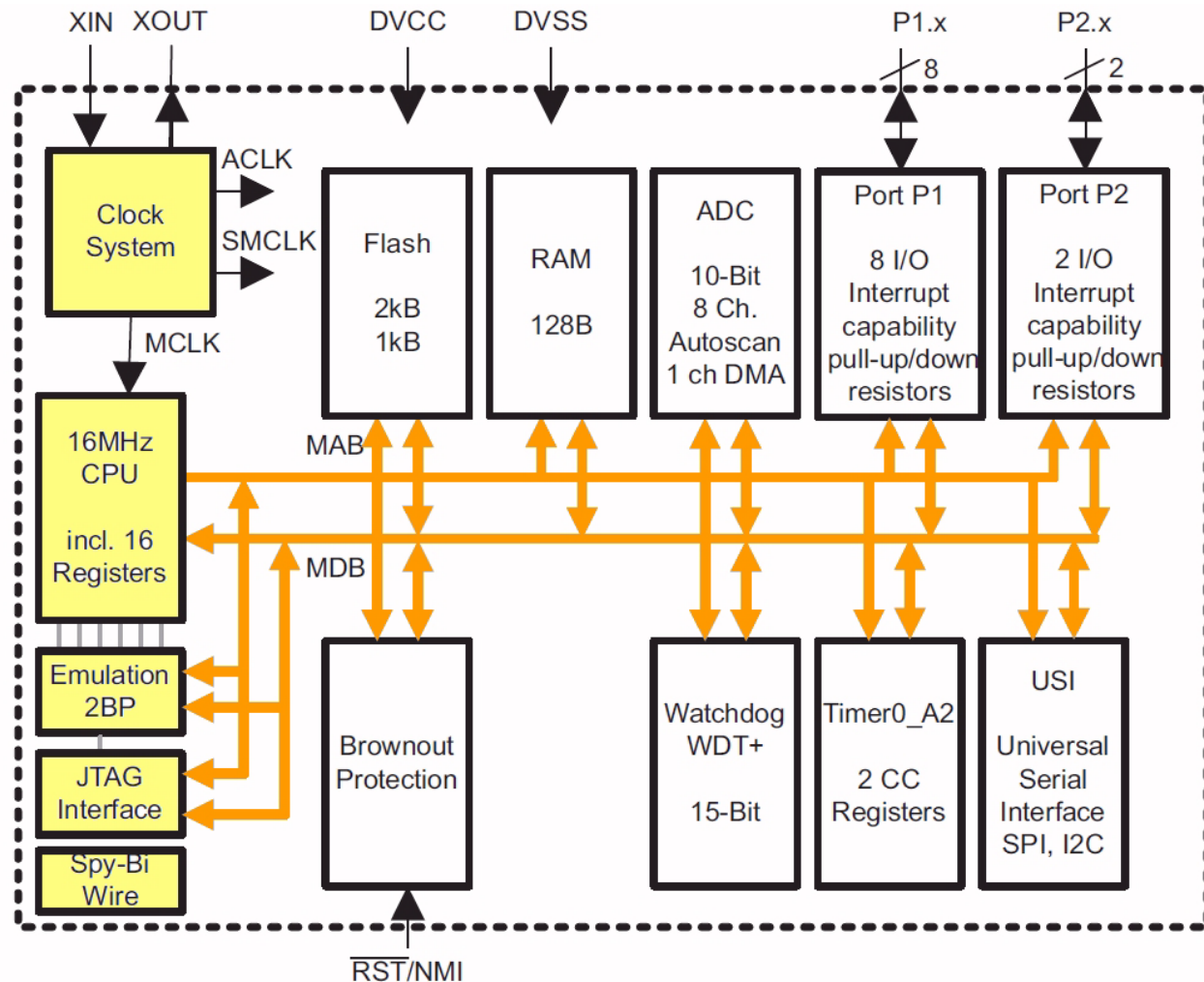


Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

Block Diagramm des MSP430G2231

◆ CPU und „Support-Blöcke“:

- Clock-Generator
- JTAG-Interface
- Spy-Bi-Wire
- Emulation (TI-Debugger)



Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

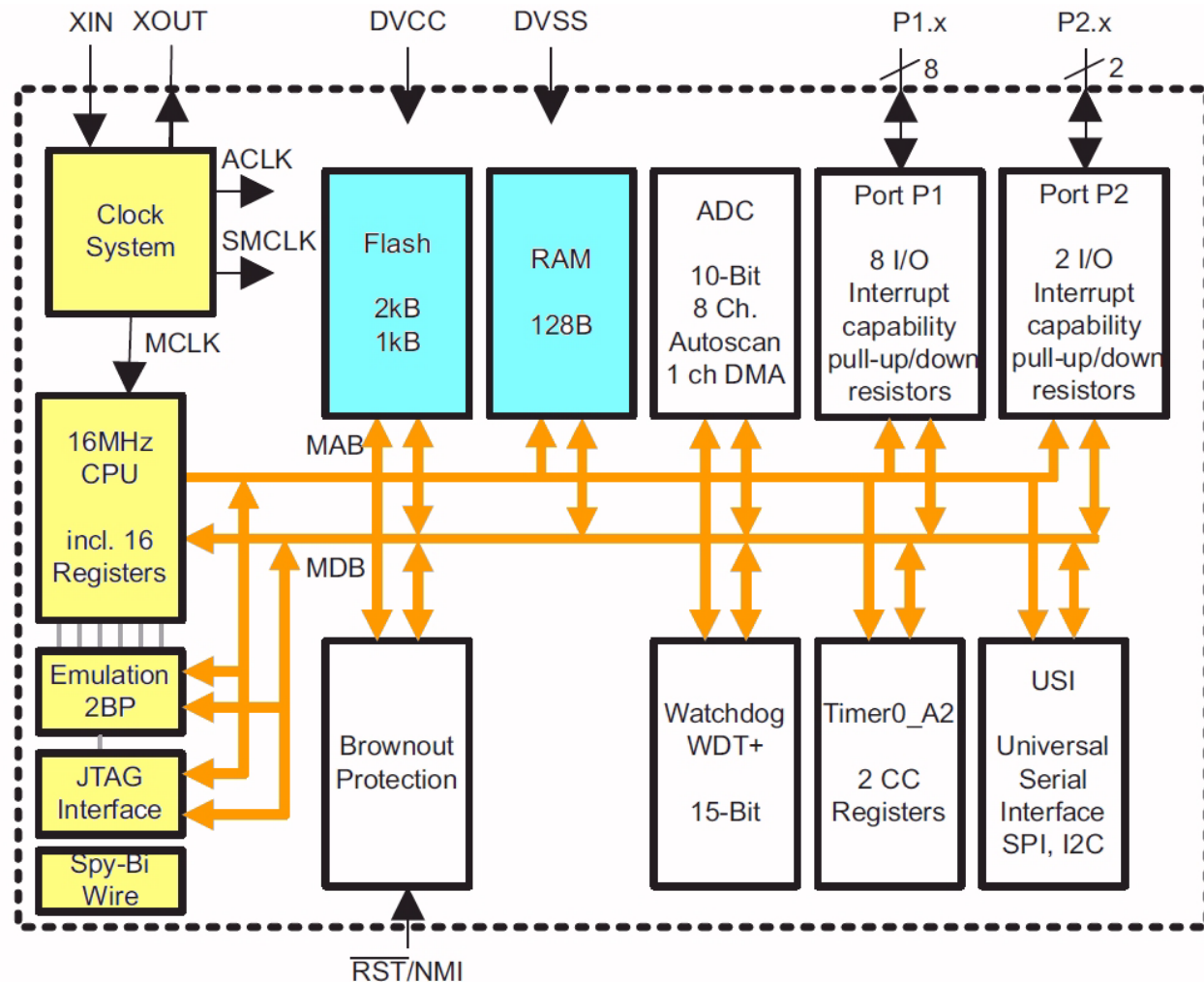
Block Diagramm des MSP430G2x31

◆ Speicher

□ 128 (S)RAM

□ MSP430G2x31:

„x“ codiert x kB
des Flash-
Speichers, z.B.:
G2231 – 2kB
G2131 – 1kB

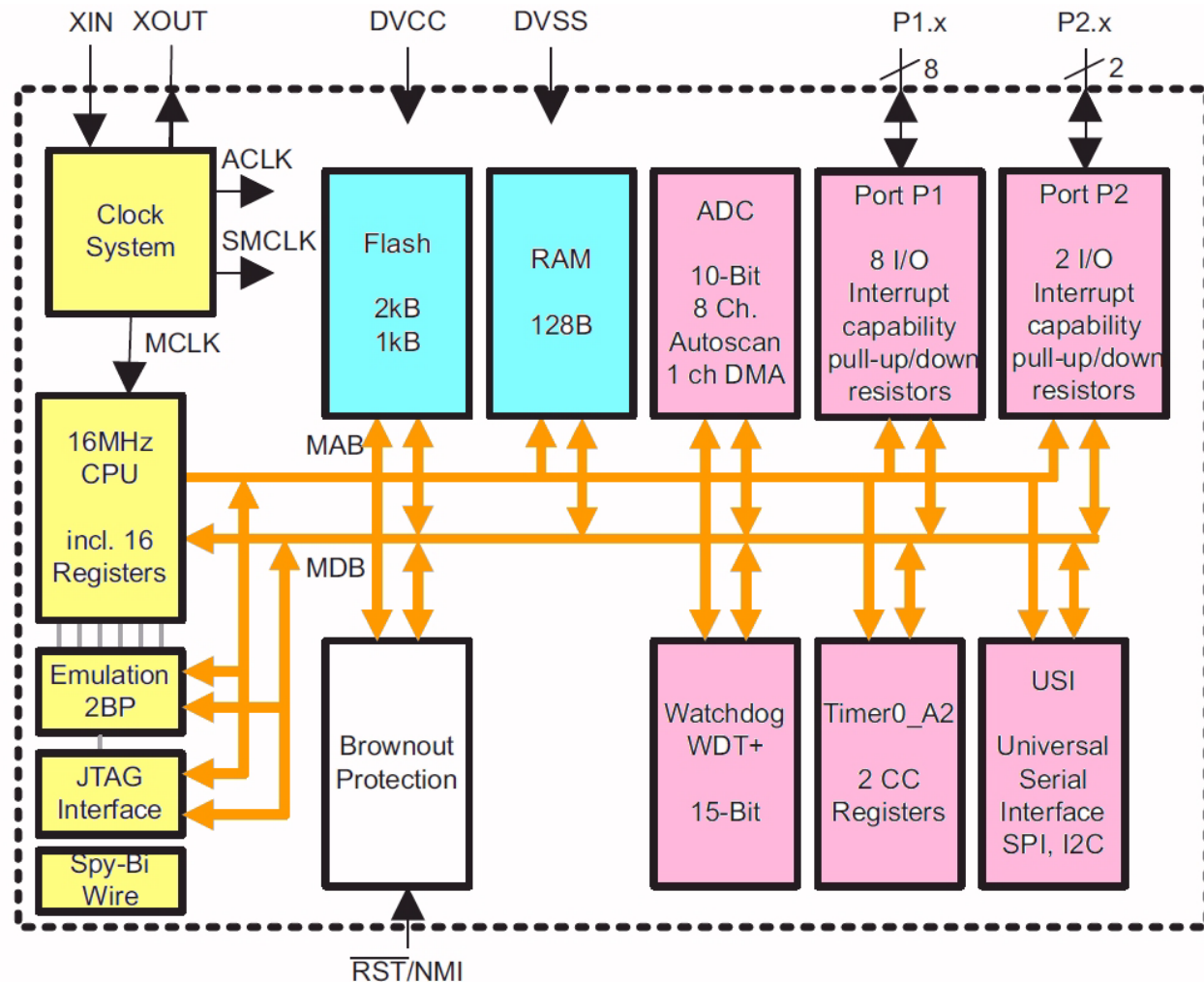


Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

Block Diagramm des MSP430G2231

◆ 6 Blöcke für Peripheriefunktionen:

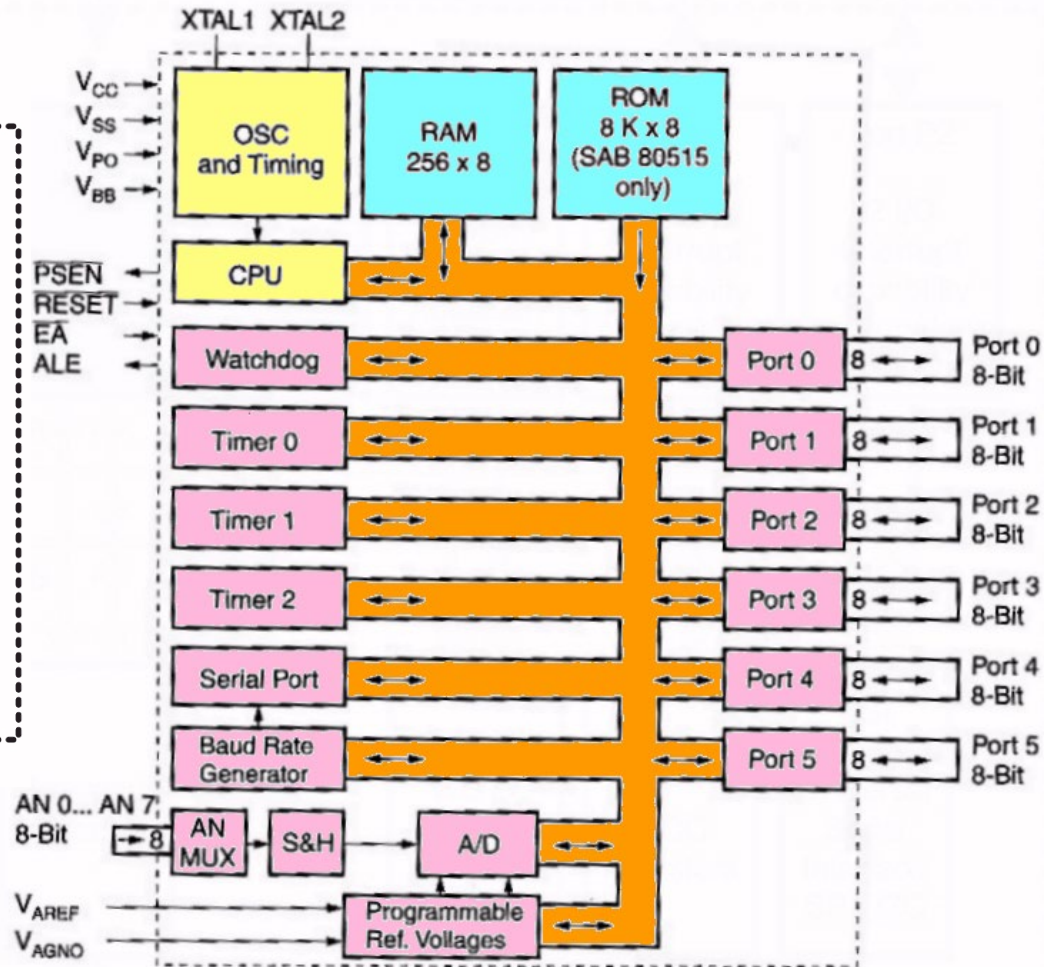
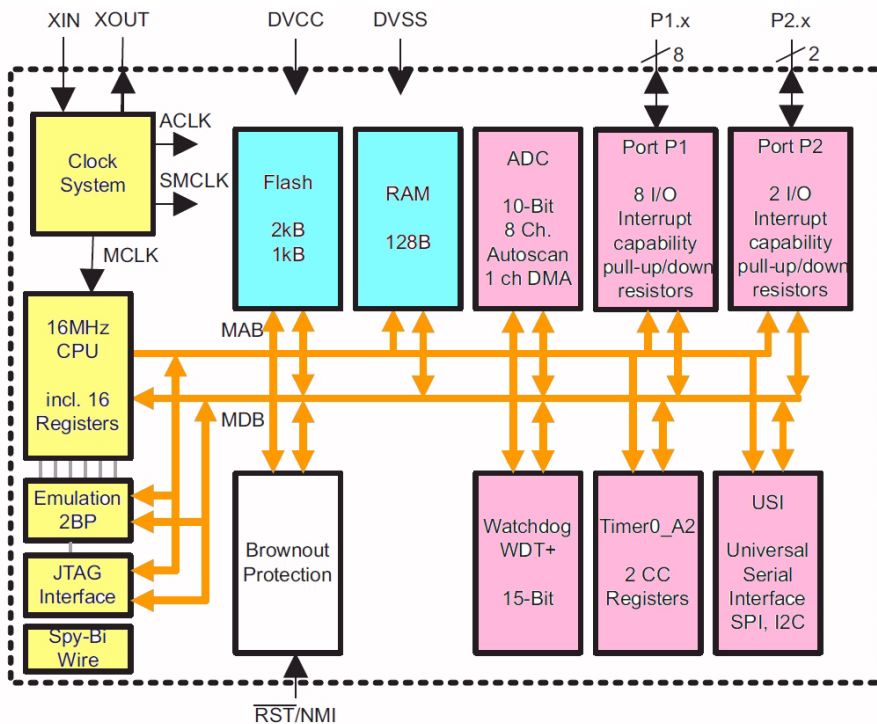
- Ein-/Ausgangs-Ports
- Timer
- Watchdog-Timer
- Universal-Serial-Interface
- 10bit-ADC



(Mehr Peripherieblöcke
bei größeren Bausteinen)

Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

Vergleich des Block Diagramme von MSP430G2231 und 80515



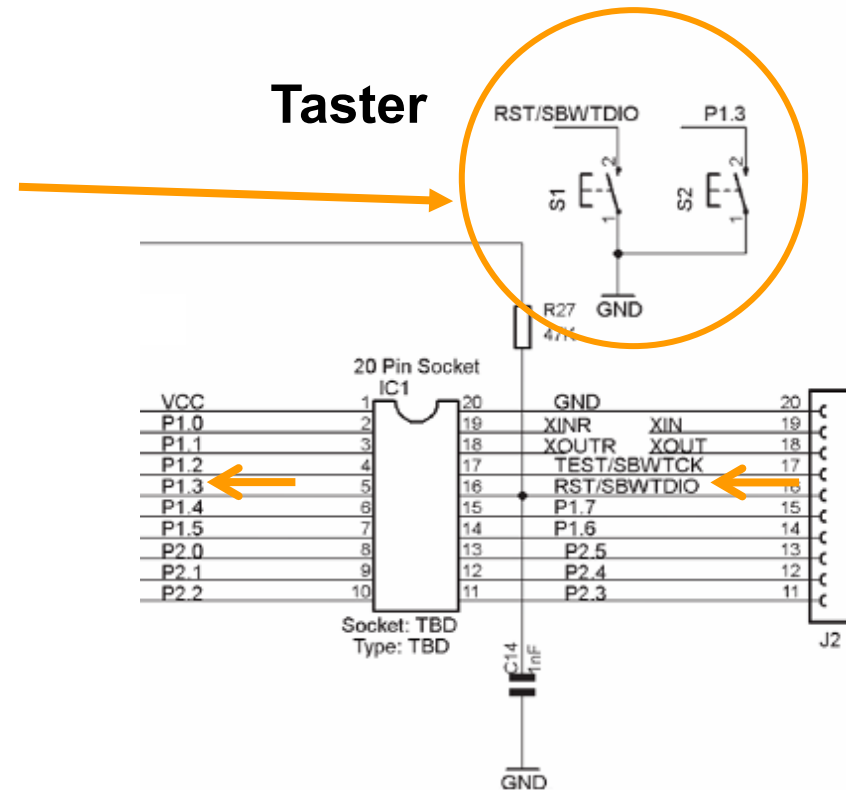
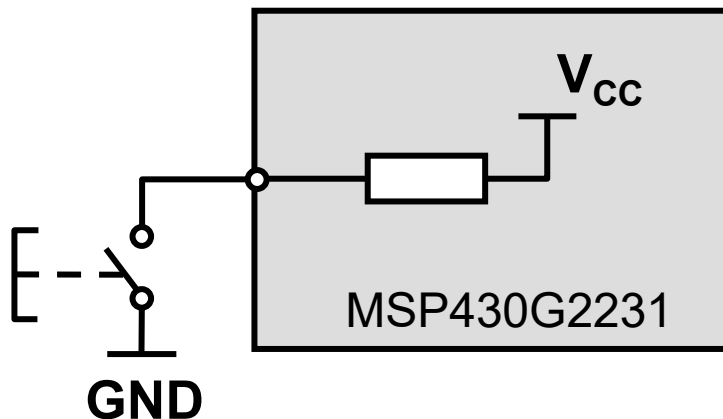
Quellen: Datenblatt SLAS694E, Texas Instruments, Feb. 2010; B. Schaaf, Mikrocomputertechnik, Hanser 2010.

Gliederung zur Mikrocomputertechnik

1. Einleitung und Motivation
2. **Aufbau und Entwicklung mit Mikrocontrollern**
 - 2.1 Die Entwicklungsumgebung
 - 2.2 LaunchPad MSP-EXP430G2
 - 2.3 Code Composer Studio und erstes Programm
 - 2.4 Überblick über den Aufbau des MSP430
 - 2.5 Die Programmiersprache C → Praktikum
 - 2.6 Beispiel –Abfrage der digitalen Eingabe in C

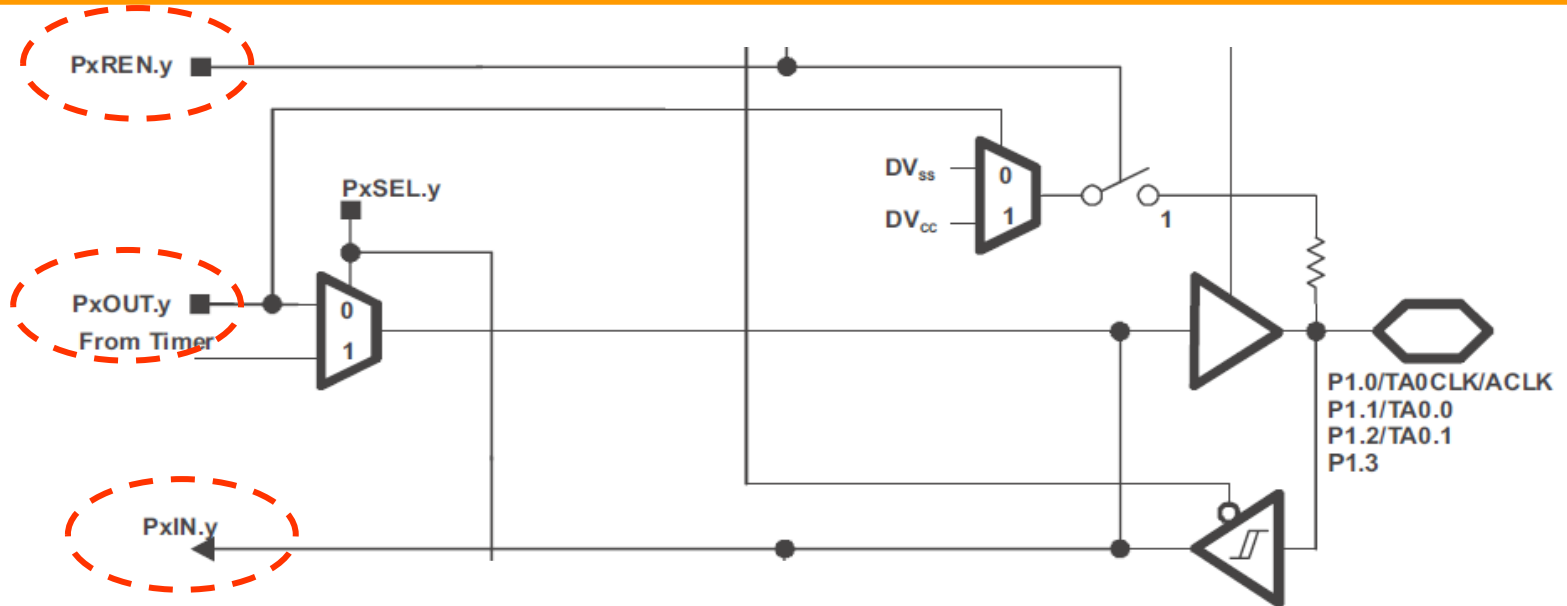
Schaltplan mit Peripherie

- ◆ Die Tastschalter sind „active low“, da bei Betätigung eine Verbindung zu GND besteht.
- ◆ Jeder I/O-Pin hat einen (im C-Programm) konfigurierbaren Pull-Up- bzw. Pull-Down-Widerstand:



Quelle: Texas Instruments, SLAU318, Juli 2010.

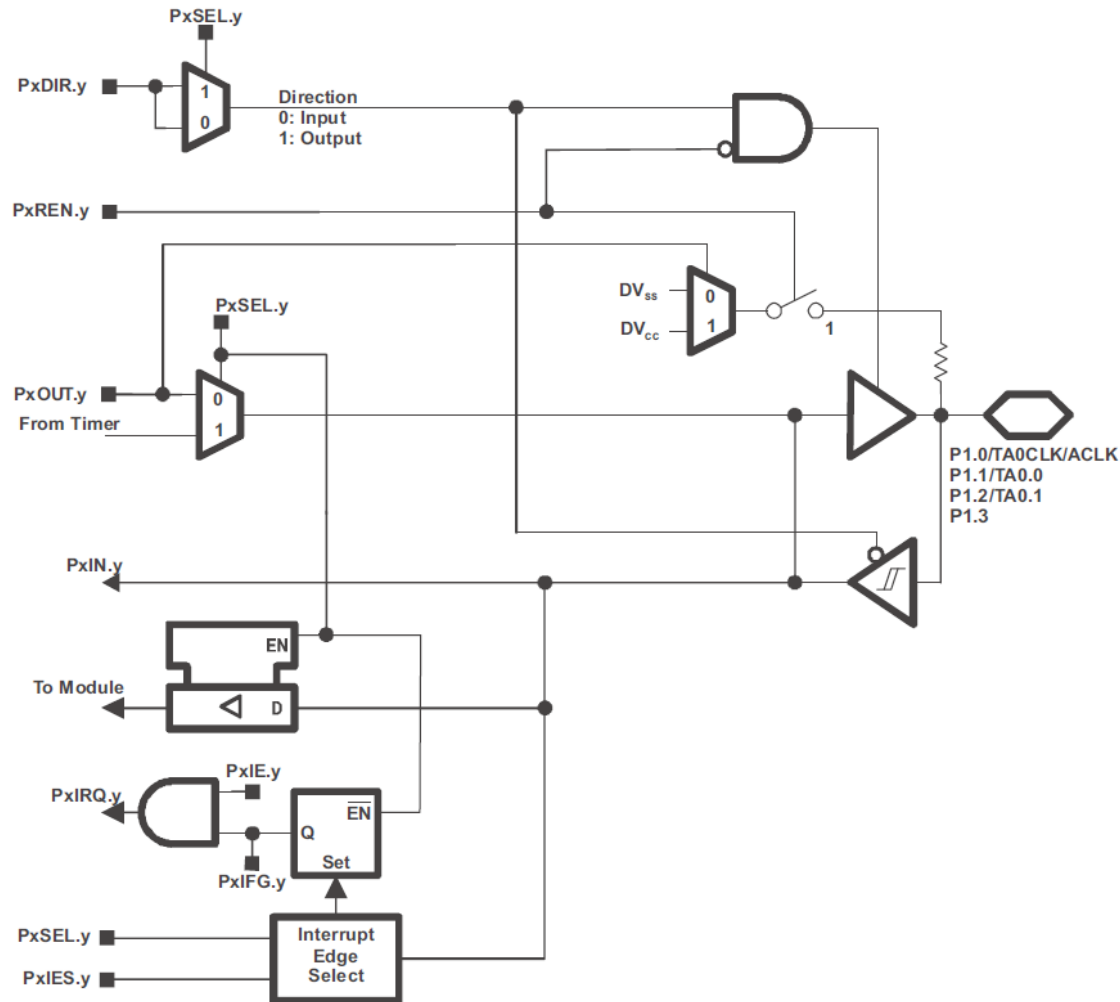
Schaltplan mit Peripherie



- ◆ Jeder I/O-Pin hat einen programmierbaren Pull-Up/Pull-Down Widerstand, falls der I/O-Pin als Eingang definiert ist:
 - Register P1REN.y (REN = Resistor ENable) für Port P1.y,
 - Register P1OUT.y für das Pull-Up bzw- Pull-Down Potential (Es wird entweder VCC oder VSS an den Widerstand angelegt.)
 - P1IN ist das Eingangsregister

Quelle: Texas Instruments, SLAS694E, Feb. 2010.

Schaltplan mit Peripherie



Quelle: Texas Instruments, SLAS694E, Feb. 2010.

Speicherzuordnung der Ein- und Ausgänge

- ◆ **Digitale Ein- und Ausgänge sind P1 und P2 mit den Anschlüssen P1.0 bis P1.7 und P2.6 und P2.8**
- ◆ **Speicherzuordnung durch die Peripherieregister**


Port	Register	Short Form	Address
P1	Input	P1IN	020h
	Output	P1OUT	021h
	Direction	P1DIR	022h
	Interrupt Flag	P1IFG	023h
	Interrupt Edge Select	P1IES	024h
	Interrupt Enable	P1IE	025h
	Port Select	P1SEL	026h
	Port Select 2	P1SEL2	041h
	Resistor Enable	P1REN	027h

Quelle: Datenblatt SLAU144J, Texas Instruments, July 2013.

Peripherieregister der Ein- und Ausgänge

◆ Konfigurationsbeispiel

- $P1DIR.x = 1$ – Output
- $P1DIR.x = 0$ – Input



PIN NAME (P1.x)	x	FUNCTION	CONTROL BITS / SIGNALS ⁽¹⁾		
			P1DIR.x	P1SEL.x	ADC10AE.x (INCH.y = 1)
P1.0/ TA0CLK/ ACLK/ A0	0	P1.x (I/O)	I: 0; O: 1	0	0
		TA0.TACLK	0	1	0
		ACLK	1	1	0
		A0	X	X	1 (y = 0)

- ◆ $P1OUT.x = 0$ oder $1 \rightarrow$ Ausgangspin P1.x wird auf V_{SS} oder V_{CC} gelegt.
- ◆ $P1IN.x$ – das Auslesen dieses Registers gibt eine logische 0 oder 1 je nachdem, ob am P1.x-Anschluss 0V oder V_{CC} anliegt.

Quelle: Datenblatt SLAS694E, Texas Instruments, Feb. 2010.

Grundlegende Abfrage eines digitalen Eingangs im Programm

- ◆ **Polling – eine wiederholte Abfrage des Eingangs**
 - Einfach
 - Benötigt umso mehr Prozessor-Zeit, je häufiger abgefragt wird.
 - (jetzt)

- ◆ **Interrupt**
 - benötigt eine bestimmte Hardware
 - Einbindung ins Programm schwieriger, da ein Unterprogramm aufgerufen und beendet werden muss.
 - (ISR: Interrupt Service Routine – später)

Test des digitalen Eingangs (1)

◆ C-Code für die rote LED

```
#include <msp430g2231.h>
```

```
#define LED1 BIT0
```

```
#define S2 BIT3
```

```
void main(void)
```

```
{
```

```
    WDTCTL = WDTPW + WDTHOLD;
```

```
    P1DIR |= LED1;
```

```
    P1OUT &= ~LED1;
```

```
    P1REN |= S2;
```

```
    P1OUT |= S2;
```

**für LaunchPad
Rev 1.5 ohne
Bestückung**

```
    for (;;) {
```

```
        if ((P1IN & S2)==0) {
```

```
            P1OUT |= LED1; }
```

```
        else {
```

```
            P1OUT &= ~LED1; }
```

```
    }
```

```
}
```

```
// oder <msp430g2553.h>
```

```
// LED1 - P1.0
```

```
// Schalter - P1.3
```

```
// Watchdog-Timer anhalten
```

```
// P1.0 in Ausgangsrichtung
```

```
// LED1 ausgeschaltet
```

```
// Resistor Enable an S2 ein
```

```
// Resistor an VCC (Pull-Up)
```

```
// infinite loop
```

```
// falls Schalter gedrückt
```

```
// LED1 einschalten
```

```
// LED1 ausschalten
```

Beispiel im CCS

Setzen und Löschen von Bits in C

Test des digitalen Eingangs (2)

◆ C-Code für die rote LED

```
#include <msp430g2231.h>
```

```
#define LED1 BIT0  
#define S2 BIT3
```

BIT0 und BIT3 im Header-File definiert

```
void main(void)
```

```
{
```

```
    WDTCTL = WDTPW + WDTHOLD;
```

```
    P1DIR |= LED1;
```

```
    P1OUT &= ~LED1;
```

| bitweises „ODER“

```
    for (;;) {
```

```
        if ((P1IN & S2) == 0) {
```

```
            P1OUT |= LED1; }
```

```
        else {
```

```
            P1OUT &= ~LED1; }
```

& bitweises „UND“

Achtung Klammer, weil ==
Vorrang vor dem & hat

```
    }
```

```
}
```

Übung 2a und 2b

- ◆ Bei gedrücktem Schalter S2 soll LED1 ein- und LED2 ausgeschaltet sein.
- ◆ Bei geöffnetem Schalter S2 soll LED2 ein- und LED1 ausgeschaltet sein.

- ◆ Implementierung der Abfrage mit zwei Schleifen

Übung 3

- ◆ Bei jeder Betätigung von Schalter S2 sollen LED1 und LED2 umgeschaltet werden.
- ◆ LED1 und LED2 sollen während der Betätigung ein- bzw. ausgeschaltet sein.