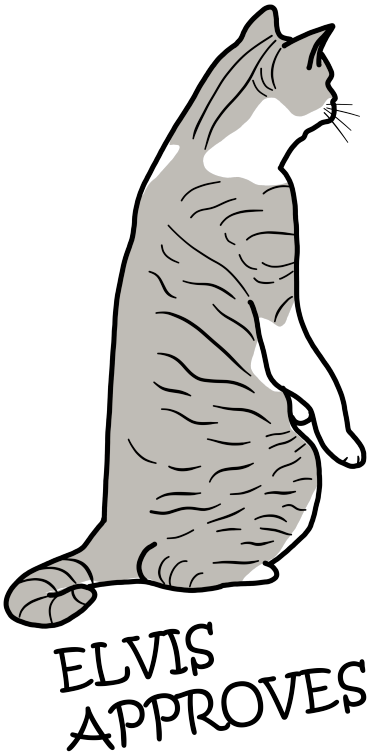
SigSys2

Formelsammlung

Sebastian Humbel

October 25, 2023



1 GPIO 8.1, 8.2

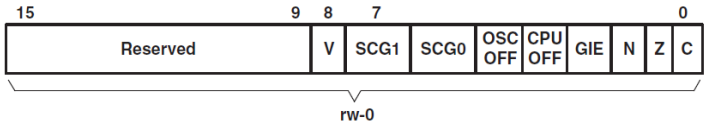
2 Interrupt Concepts 7.1, 7.2

A Interrupt is a signal indicating the occurrence of an event that needs immediate CPU attention. There exist different types of Interrupts:

- Hardware triggers
- Software triggers
- CPU exceptions

```
PxDIR &= ~0x01; // Set Px.0 as input
PxDIR |= 0x02;  // Set Px.1 as output
PxOUT &= ~0x02; // Clear LED on Px.1
PxIE |= 0x01; // IRQ when SW1 is pressed
...
__interrupt void portx_ISR(void)
{
  PxOUT ^= 0x02;
}
```

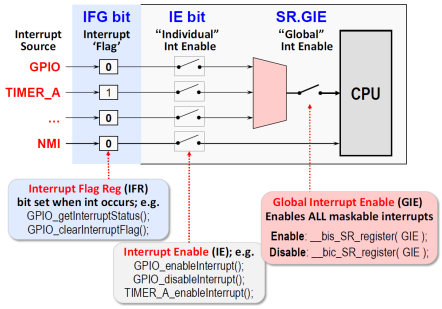
2.1 Maskabe vs. Nonmaskable 7.1.2



Maksable: Can be blocked (masked), trough flags.
Non-maskable interrupts (NMI):

- Cannot be masked, thus are always served
- Reserved for system critical events

2.1.1 Interrupt Flow



Interrupt kann über intrinsische Funktionen eingestellt werden. Alternativ mit enable_interrupt() funktion.

2.1.2 Interrupt Service Sequence 7.1.3

PC speicher den aktuellen ort im code bevor der Interrupt ausgeführ wird. SR wird auf dem stack gespeichert

2.1.3 Interrupt Identification Methods 7.1.4

Non-vectored systems (1) dont know, what triggerd the interrupt and have to identifa the interrupt sourth within the ISR.
Vectored interrupts (2) Have a ID for each interrupt type an can jump directly to the desired ISR.
Auto-vectored interrupts (3) have predefined addresses / fix vectors for the different interrupts.

2.1.4 Interrupt Priority Handling 7.1.5

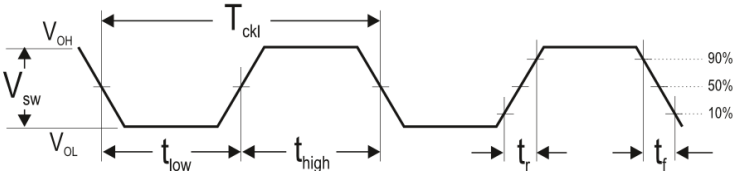
(1) *Polling order of SRQ flags* decides priority (if else if define the interrupt priority)
(2) *Daisy Chain-based Arbitration* defines the priority through hardwired logic. Used by the MSP430.
(3) *Interrupt controller-based Arbitration* allows priority configuration by the user.

2.2 Interrupt Software Design 7.3

3 Clock System

3.1 Clock Sources 6.3

Voltage Swing $V_{SW} = V_{OH} - V_{OL}$: Amplitude from low to high.
Frequency $f_{clk} = \frac{1}{T_{clk}}$: Number of cycles per second
Duty Cycle $DC = \frac{t_{high}}{T_{clk}} * 100\%$: Ratio of high time to period
Edge Speed t and t_f : Rising and falling times



3.2 Clock Stability 6.3.1

Statistical measure of the maximum allowable parameter fluctuations of a clock signal over a given time interval.

Possible factors:

- type of oscillator
- capacitive load
- ageing
- supply voltage or temperature

Short- and long-term effects:

- Clock Jitter
- Clock Drift

3.3 Source Selection 6.3.2

Choose the lowest frequency that allows for a reliable and correct system operation. Adjust the frequency to the speed of the fastest event.

3.4 Internal vs. External Clock 6.3.3

Internal
Pro:

- Reduce external component count
- Less expensive
- Induce lower power consumption

Con:

- Reduced clock stability
- Less flexibility
- Narrower bandwidth

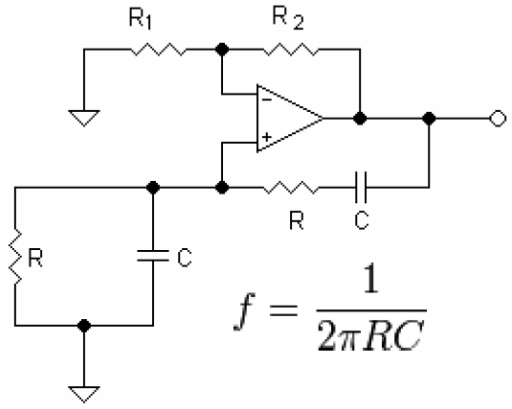
External
Pro:

- Wider choice of frequencies
- More design flexibility

Con:

- Increment the external component count
- Higher clock speeds induce higher power consumption

3.5 RC-based Oscillator



3.6 Quartz Crystal Oscillators 6.3.4

Based on mechanical resonance of a vibrating piezoelectric crystal.

3.7 Pierce Crystal Oscillator 6.3.4

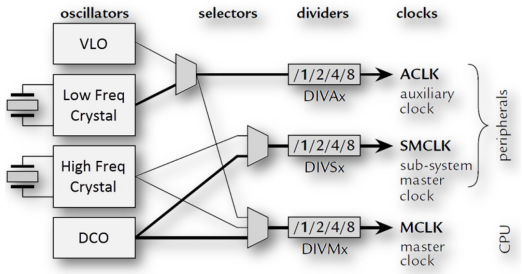
Series Resonant Oscillator

3.8 Colpitts Crystal Oscillator 6.3.4

Parallel Resonant Oscillator

3.9 Oscillator Startup Time 6.3.4

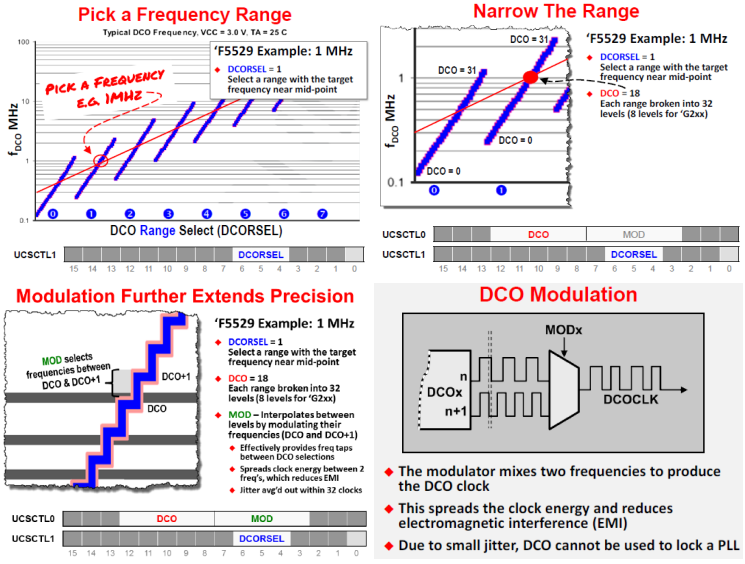
4 The MSP430 System Clock 6.4



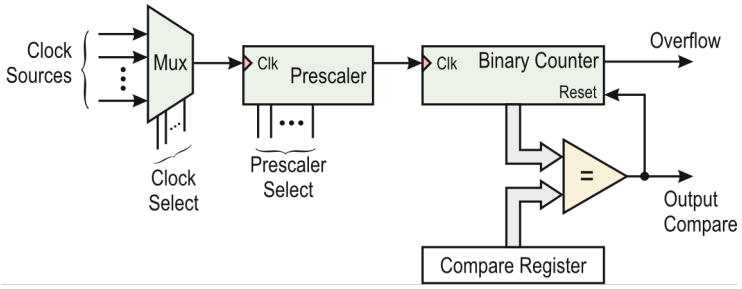
4.1 DCO Clock

Digitally controlled oscillator

1. Select a frequency range (DCORSEL)
2. Select the frequency within that range (DCO)
3. Setup Modulation (MOD)



5 Timer & Event Counters 7.4



5.1 Signature Timer Applications 7.4.3

Watchdog timer is secured through a password in the *WDTCTL* register. *Real-Time Clocks* provides absolute time (second, minute, hour, day, month, week). *Baud Rate Generation* can provide clock base for baud rate.

$$baudrate = \frac{f_{clk}}{PS \cdot TopCount}$$

PS : Prescale Factor [1]
f_{clk} : Clock frequency [Hz]
$TopCount$: Compare value [1]

5.1.1 MSP430 Timer Support 7.4.4

Operating Modes
Input Capture Operation
Measuring a Signal Width
Output Compare Operation

6 Low-Power Modes 7.3.6

Operating Mode	CPU (MCLK)	SMCLK	ACLK	RAM Retention	BOR	Self Wakeup	Interrupt Sources
Active	☒	☒	☒	☒	☒		Timers, ADC, DMA, WDT, I/0, External Interrupt, COMP, Serial, RTC, other...
LPM0		☒	☒	☒	☒	☒	
LPM1		☒	☒	☒	☒	☒	
LPM2			☒	☒	☒	☒	
LPM3			☒	☒	☒	☒	
LPM3.5					☒	☒	External Interrupt, RTC
LPM4				☒	☒		External Interrupt
LPM4.5					☒		External Interrupt

6.1 Activity Profiles

6.2 Power Consumption in CMOS Technology

$$P \sim \alpha C_L V_{dd}^2 f$$

$$E \sim \alpha C_L V_{dd}^2 f t$$

$$E \sim \alpha C_L V_{dd}^2 (\#cycles)$$

$$\tau \sim C_L \frac{V_d d}{(V_d d - V_T)^2}$$

$$f \sim \frac{1}{\tau} \sim V_{dd}$$

V_{dd} : supply voltage [V]
V_T : threshold voltage [V] α : switching activity [1]
C_L : load capacity [F]
f : clock frequency [Hz]
τ : Gate delay [ms]

6.3 Interrupts and Low-Power Modes

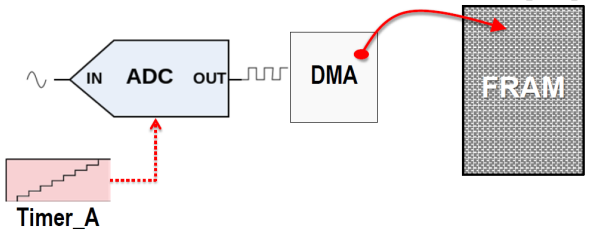
Low power modes are configured with the bits *CPUOFF*, *OSCOFF*, *SCG0* and *SCG1* in the **Status Register** (SR, also called R2).

6.4 CCS6 Intrinsic Functions for MSP430

```
#include <intrinsics.h>
```

6.5 Low-Power Optimization

Replace software with hardware peripherals.



7 Real-Time Clock (RTC) 7.4.3

7.1 RTC Interrupts

1. Alarm
2. Interval Timer
3. Prescaler 0
4. Prescaler 1
5. Ready for Register operation

Example 1: Time Event

```
int main (void)
{
  WDTCTL = WDTPW | WDTHOLD; //stop watchdog
  //disable high-impedance ports
  PM5CTL0 &= ~LOCKLPM5;
  //just for testing (P1.0 / LED1)
  P1DIR |= 0x01;
  P1OUT &= ~0x01;

  PJSEL0 = BIT4 | BIT5; //Init. LFXT pins

  //Configure LFXT 32kHz crystal
  CSCTL0_H = CSKEY >> 8;   //Unlock CS reg.
  CSCTL4 &= ~LFXTOFF;      //Enable LFXT
  do
```

```
  {
    //Clear LFXT fault flag
    CSCTL5 &= ~LFXTOFFG;
    SFRIFG1 &= ~OFIFG;
    //Test oscillator fault flag
  } while (SFRIFG1 & OFIFG);
  CSCTL0_H = 0; //Lock CS registers

  //write RTCKEY to unlock RTC reg. for write
  RTCCTL0_H = 0xA5;
  //hold RTC, clock from output of RT1PS,
  // RTCTEVIFG on 8-bit overflow
  RTCCTL1 = 0x48;

  //Enable time event interrupt
  RTCCTL0_L = 0x40;
  //prescale timer 0: divided by 64
  RTCPS0CTL = 0x2800;
  //prescale timer 1: out from RT0PS, div. by 2
  RTCPS1CTL = 0x8000;
  //Enable RTC
  RTCCTL1 &= ~0x40;
  //Enter LPM3 with interrupt enabled
  __bis_SR_register(LPM3_bits + GIE);

  while (1)
  {}
}
```

Example 2: RTC Calendar

```
//BCD, hold RTC, RTCMode, one minute overflow
RTCCTL1 = 0xE0;

RTCSEC = 0x00;  //Set Seconds
RTCMIN = 0x00;  //Set Minutes
RTCHOUR = 0x10; //Set Hours
RTCDOW = 0x03;  //Set DOW (sunday = 0)
RTCDAY = 0x18;  //Set Day
RTCMON = 0x10;  //Set Month
RTCYEAR = 0x2023; //Set Year

//setup alarm
//enable minutes alarm, minutes = 1 and others are
    don't care
// --> so an alarm is generated on 00:01:00,
    01:01:00, ...
RTCAMIN = 0x81;
RTCAHOUR = 0x00;
RTCADOW = 0x00;
RTCADAY = 0x01;

RTCCTL0_L = 0x60;
//Enable time event & alarm interrupt
RTCCTL1 &= ~0x40;
//Enable RTC
__bis_SR_register(LPM3_bits + GIE); //Enter LPM3 w/
    interrupt
while (1)
{}
```