Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

Hardwarenahe Softwareentwicklung

Timer, PWM, Watchdog

V5.1, ©2023 roger.weber@bfh.ch

Lernziele

Sie sind in der Lage:

- Einsatzgebiete von Timern zu nennen.
- Timer-Applikationen zu programmieren.

Inhaltsverzeichnis

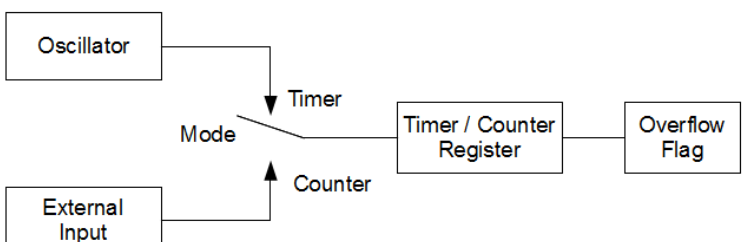
Grundlagen und Anwendungen

Anwendungen

- Auf einem Microcontroller hat es typischerweise mehrere Timer.
- Einsatzmöglichkeiten:
  - Zeitbasis (z.B. System-Clock für Betriebssystem)
  - Clock-Generation (z.B. Baudraten für serielle Kommunikation)
  - Counter von externen HW-Ereignissen
  - PWM-Generation
  - Zeitmessungen
  - Watchdog

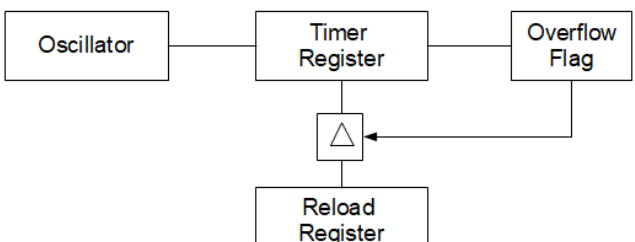
Timer / Counter

- Timer-Register: Je nach Microcontroller 8, 16 oder 32-Bit
- Im Timer-Betrieb wird der Inhalt des Timer-Registers durch einen Oszillator inkrementiert (up) oder dekrementiert (down).
- Im Counter-Betriebe wird das Timer-Register durch ein externes Signal inkrementiert.
- Beim Überlauf des Timers (bei 16-Bit Timer von 0xFFFF auf 0) wird ein Overflow-Flag gesetzt und je nach Konfiguration ein Inerrupt ausgelöst.



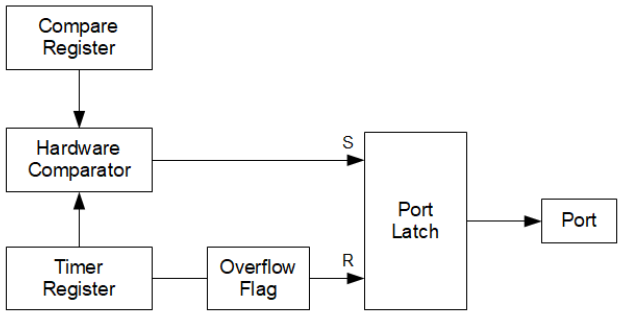
Timer Autoreload

- Erzeugung einer definierten Periodendauer.
- Overflow-Flag $\rightarrow$ Reload-Wert wird ins Timer-Register geladen.
- $t_{period} = \frac{(0xFFFF - Reload\_Value)}{f_{osc}}$



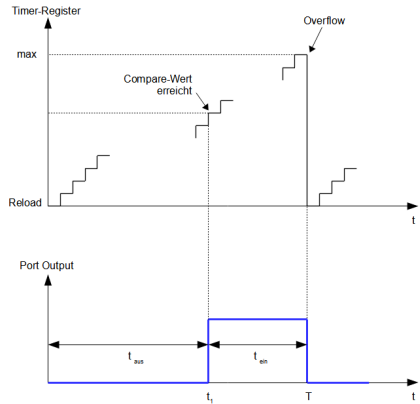
Compare-Einheit

- Hardware-Comparator vergleicht Timer-Register mit Compare-Register und setzt Flip-Flop am Ausgang.
- Overflow-Flag setzt das Flip-Flop zurück.



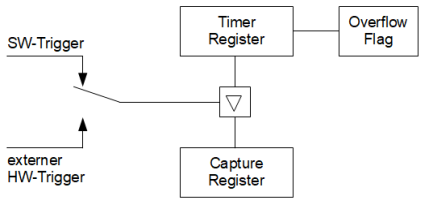
Compare-Einheit / PWM

- Erzeugung von PWM-Signalen (im Beispiel unten count-up).
- Hardwaremässig, SW nur zur Initialisierung.
- Duty-Cycle des PWM-Signals (Zeit, die der Ausgang hoch ist (t_{ein}) im Verhältnis zur Periodendauer ($T = t_{aus} + t_{ein}$), z.B. 30% Duty-Cycle.



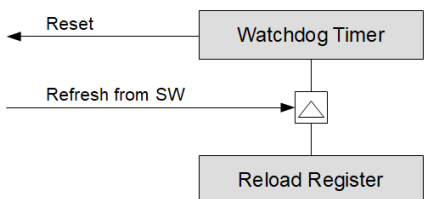
Capture-Einheit

- Auslesen des aktuellen Wertes des Timer-Registers.
- Auslesen wird durch Trigger-Signal (SW oder HW) ausgelöst.
- Anwendungen: Zeitmessungen, Stoppuhr.



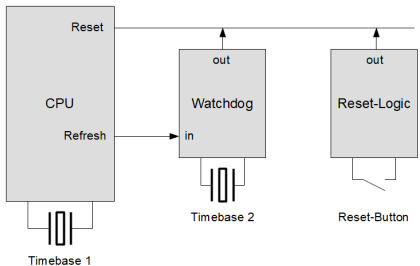
Watchdog

- Sind bei vielen Microcontrollern schon integriert.
- Aufgabe des Watchdog: Reset des Systems bei Problemen.
- Auslöser: Programmierfehler, Endlosschleifen, undefinierter Zustand.
- Die Software muss den Watchdog periodisch zurücksetzen (Refresh-Zyklus). Sonst spricht dieser an und löst einen Reset aus.
- Refresh des Watchdog am Ende der Hauptschleife in main().



Zusatzfunktionen Watchdog

- Watchdogs überwachen oft auch die Speisung und führen im Fehlerfall einen Reset durch.
- Mögliche Probleme: Glitches (kurze Spannungeinbrüche) oder Brownout (zu tiefe Spannung).
- Watchdogs können auch eine eigene Zeitbasis haben und so die Zeitbasis der CPU prüfen.



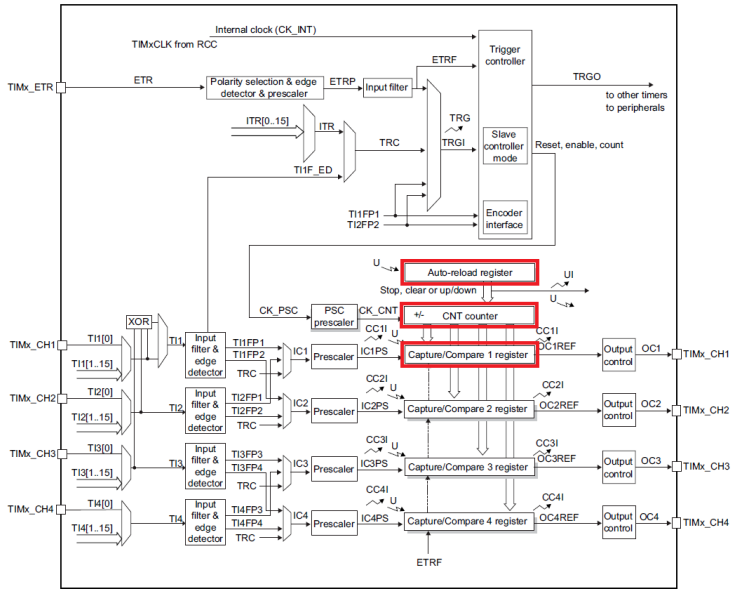
Timer auf dem STM32H7xx

Timer auf dem STM32H7xx

Der STM32H7xx stellt diverse Timer zur Verfügung:

- 1 High-Resolution Timer (HRTIM)
- 2 Advanced Control Timer (TIM1 und TIM8)
- 10 General Purpose Timer (TIM2 bis TIM5 und T12 bis T17)
- 2 Basic Timer (TIM6 und TIM7)
- 1 Low-Power Timer (LPTIM)
- 2 Watchdog
- 1 RTC (Real Time Clock)

Blockdiagramm Timer 2 bis 5 des STM32H7xx



Programmierung der Timer

Timer-Register

- Programmierung mit Hilfe der HAL-Library oder direkt auf die HW-Register.
- Die wichtigsten HW-Register der Timer sind:

Register	**CMSIS**	**Funktion**
TIM control register x	TIMx→CR1	Aktiviert verschiedene Funktionen
TIM status register	TIMx→SR	Timer Status
TIM prescale register	TIMx→PSC	Prescale für Clock
TIM capture/compare mode register	TIMx→CCMRx	Definition compare / capture mode
TIM counter register	TIMx→CNT	Counter-Wert
TIM auto reload register	TIMx→ARR	Autoreload-Wert
TIM DMA/Interrupt enable register	TIMx→DIER	DMA und Interrupts enablen

Programmierung des Timers

```
/* Enable Clock for Timer 2 */
__HAL_RCC_TIM2_CLK_ENABLE();

/* TIM2 Control Reg1, RM0433 p 1694; CKD = 0 and  DIR = 1; */
TIM2->CR1 = 0x0010;

/* set the Autoreload value */
TIM2->ARR = 10000;

/* set the prescaler value */
TIM2->PSC = 24000;

/* TIM2 DMA / Interrupt Enable Register RM0433 p 1700; UIE = 1 (enable
    interrupt) */
TIM2->DIER |= 0x01;

/* TIM2 Control Reg1, RM0433 p 1694; CEN = 1 (counter enable) */
TIM2->CR1 |= 0x01;

/* enable TIM2 interrupt in NVIC, ISER, vector number 28 */
NVIC->ISER[0] = 1 << 28;
```