Wdh. LPC 2148

- 45 Portpins → Port 0 und Port 1
- Funktionen umschaltbar, default → GPIO (general purpose input/output)
- Register:
    - IODIR0, IODIR1 → Eingang / Ausgang umschalten (z.B. IODIR0 = 0x3; Ausgang an Pins 0.0 & 0.1)
    - IOSET0, IOSET1 → HIGH-Pegel an Ports setzen (z.B. IOSET0 = 0x2; Port 0.1 auf HIGH)
    - IOCLR0, IOCLR1 → LOW-Pegel an Ports setzen (z.B. IOCLR0 = 0x2; Port 0.1 auf LOW)
    - IOPIN0, IOPIN1 → Stats der Pins auslesen
    - PINSEL0 – 2 → Pin-Funktion umschalten (PINSEL0 → 0.0 – 0.15, PINSEL1 → 0.16 – 0.31, PINSEL2 → 1.16 – 1.31)

Wdh. Externe Interrupts

- EINT0 – EINT3 → Portpins für ext. Interrupt
- EXTMODE → Pegel / Flanke umschalten (0 = Pegel, 1 = Flanke), letzte 4 Bit
- EXTPOLAR → EXTMODE prüfen, auslösenden Zustand einstellen
    - Wenn Flanke auslöst → 1 = steigende Flanke, 0 = fallende Flanke
    - Wenn Pegel auslöst → 1 = HIGH-Pegel, 0 = LOW-Pegel
- EXTINT:
    - bei 1 (letzte 4 Bit) → Interrupt wird ausgelöst
    - in der ISR → Interrupt quittieren (sonst verbleibt Prozessor in der ISR)
- Bsp.:

```
PINSEL1 = PINSEL1 | 0x20000000;     //P0.30 als EINT3
EXTMODE = EXTMODE | 0x0;            //EINT3 ist pegelabhängig (default)
EXTPOLAR = 0x8;                     //EINT3 reagiert auf HIGH-Pegel

in der ISR:
EXTINT = 0x8;                       //Interrupt an EINT3 quittieren
```

Einsatz von Interrupts

- ARM7 → 2 Arten von Interrupt:
    - IRQ – Interrupt Request
    - FIQ – Fast Interrupt Request (etwas schneller als IRQ)
- VIC – Vectored Interrupt Controller

Aufbau und Arbeitsweise des VIC

- Kontrollelement: VICVectCntl0 – VICVectCntl15
- Adresselement: VICVectAddr0 – VICVectAddr15
- Wenn eine Peripherieeinheit einen Interrupt auslösen möchte:
    - Anhand der Kanalnummer dieser Einheit wird geprüft, ob diese Peripherieeinheit einen Interrupt auslösen darf (VICIntEnable)
    - Priorität wird geprüft
    - Bei jedem aktiven Eingang wird geprüft, ob die Kanalnummer in der Tabelle mit der Peripherieeinheit übereinstimmt

- Ist dies der Fall, wird der zugehörige Adresseintrag in das Ausgaberegister kopiert (VICVectAddr)
- VIC gibt IRQ-Signal an den Prozessor → Prozessor springt in IRQ-Betriebsart

Die wesentlichen Kontrollregister des VIC

- VICIntEnable: Legt fest, ob Peripherieeinheit Interrupt auslösen darf (jedes Bit für eine Kanalnummer), z.B. VICIntEnable = 0x4000 (Kanal 14, EINT0)
- VICIntEnClear: Interrupt-Möglichkeit für Peripherieeinheiten entziehen
- VICVectCntlX (0 - 15): Aktivieren des VIC für Kanäle (6 Bit, 0-4 → Adresse, 5 → aktiv)
- VICVectAddrX (0 - 15): Adresse die ISR zuordnen
- VICDefVectAddr: Defaultadresse einer ISR, die in das Ausgaberegister übertragen wird
- VICVectAddr: Ausgaberegister des VIC → Übergabe der Sprungadresse an die ISR
- Beispiel:

```
PINSEL0 = 0xC0;                    //P0.3 als EINT1 → VIC-Kanal 15
VICVectCntl4 = 0x2F;               //0b101111 → aktiv auf Kanal 15
VICVectAddr4 = (unsigned long) my ISR;
VICIntEnable = 0x8000;             //Kanal 15 sind Interrupts erlaubt

in der ISR:
EXTINT = 0x2;                      //EINT1 quittieren
VICVectAddr = 0x0;                 //Interrupt quittieren
```

- in C:

```
void myExtInt(void) __irq{}        /__ als Hinweis für Compiler
```

Verwendung von Timern

- LPC2148 → 2 Timer: Timer0 und Timer1

Aufbau eines Timers

- Peripherial Clock des LPC2148 → 15 MHz
- Prescaler → Takt teilen, um langsamer zu zählen
- Match-Register → Zahlenwerte zum Vergleichen mit Zähler → TXMCR (reset, stop, interrupt)

Die Kontrollregister der Timer (X = 0/1)

- TXPR → Prescaler, teilt den Takt herunter
- TXTC → Timer Counter, Zählerstand
- TXTCR → Timer Control Register (nur 2 Bit):
  - Bit 0: 1 = Start, 0 = Stop
  - Bit 1: 1 = Reset
- TXMCR → Match Control Register, stellt die Match-Funktionalität für alle MRs ein
- TXMR0 – TXMR3 → Match Register (enthält Vergleichswerte)
- TXIR → quittieren des Interrupts in der ISR (richtet sich nach Match-Registern)

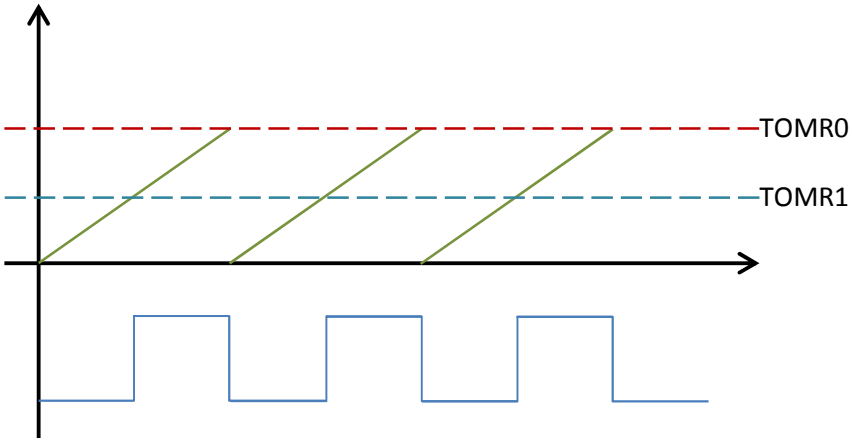
- Bsp.:

  T0MR2 = 0x2000; //Match Register setzen

  in der ISR:

  T0IR = 0x4; //0b0100 → MR2 Adresse, quittieren auf MR2

Beispiel-Programm



T0MR0 = 1000; //MR0 auf Timer0 auf 1000 einstellen
T0MR1 = 500; //MR1 auf Timer0 auf 500 einstellen
T0MCR = 0xB; //Interrupt bei MR1, Interrupt und Reset bei MR0 (001011 → 0xB)