# Interrupt/Event Steuerung

IOC Edge detected Interrupts = = Events (ioc.h Z: 457)

* Handler: Bearbeitung der Interrupts 26xx/startfiles/startup\_ccs.c  
  Vektrotabelle im Memory. Interrupts kommen von IOCIntRegister
* Interruptfunktionen vom Hersteller: 26xx/driverLIb/interrupt.c  
  Prioritäten, Freischalten, Löschen Interrupt, ….
* **Interrupt/event senden, warten, auslesen , …**  26xx/driverLib/cpu.c
* Event-Quelle in Fabrik Register eintragen, vom Hersteller 26xx/driverLib/event.c
* Event-WakeUP-Steuerung: Takt, Ausschalten MCU 26xx/driverLib/aon\_wuc.h
* Def./Adressen IOD, PORT (MCU, GPIO, AON, RF) Definitionen 26xx/dreiverLib/ioc.h

**Vebindet (map) Peripherals (GPIO, MCU, AON, RF) mit lDIOs**.Greift auf CFG zurück.IOCPortConfigureSet(DIOn, PORTID, PIN-CONFIG);

**Alle IO-Konstanten**: IOC\_NO\_WAKE\_UP für **Pinkonfiguration**. (Jeder Pin besitzt ein Register)  
**Interrupt Register** (Z: 674ff): Ruft die Funktion auf, wenn ein Interrupt sich ereignet, wird an Handler übergeben.

**AON**

* Event-Quelle für AON-Fabrik vom Hersteller 26xx/driverLib/aon\_event.h  
  AON Wakeup MCU-Funktionen für total 4 Events, Events werden dem WUC weitergegeben (Z: 132)
* RTC-Quelle: Kanalauswahl, delay, Compare/Capture, Zustand 26xx/driverLib/aon\_rtc.h
* Treiberstärke 26xx/driverLib/aon\_ioc.h

# System/Initialszustand

* Initfunktionen (& Powerenable ?!) inkl. RF-Funktionen ./system.h  
  Init RTC, LED, Interrupt, …

# PowerManagement

* Powerzustände vom Hersteller (Down, Standby, Refresh, CPU) 26xx/driverLib/sys\_ctrl.h
* PowerDomains 26xx/driverLib/prc.h

# Konstanten

* Alle IO-Konstanten (wie BOARD\_IO\_DP0) ./board.h
* Alle GPIO-Konstanten (wie GPIO\_PIN\_29) ./gpio.h
* Applikations-spezififische Konstanten (WAKE\_INTERVAL, ADVLEN) ./config.h

# Funktionen

* Radio-Funktionen (run, ..) ./radio.h  
  bus,start(), updateData() ./system.h
* RTC-Funktionen (calculation) ./rtc.h
* Sensoren