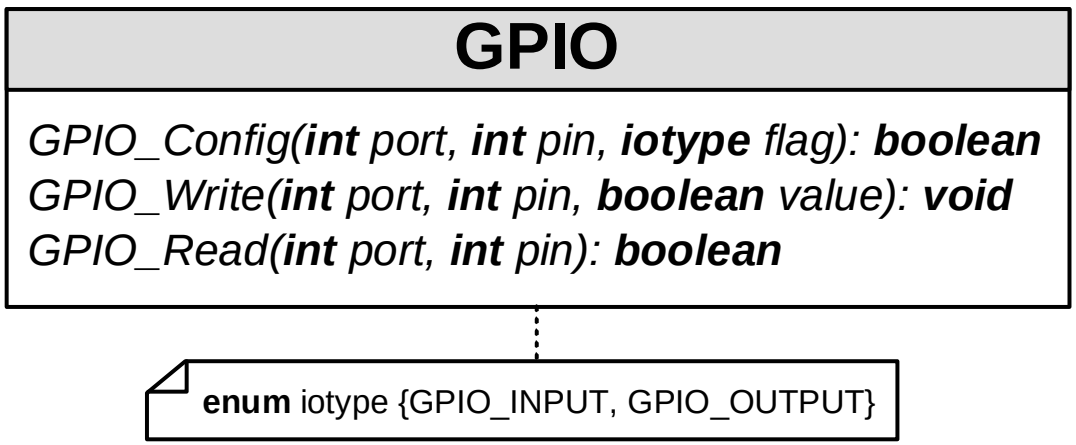
Hau den Lukas
Echtzeitsysteme II

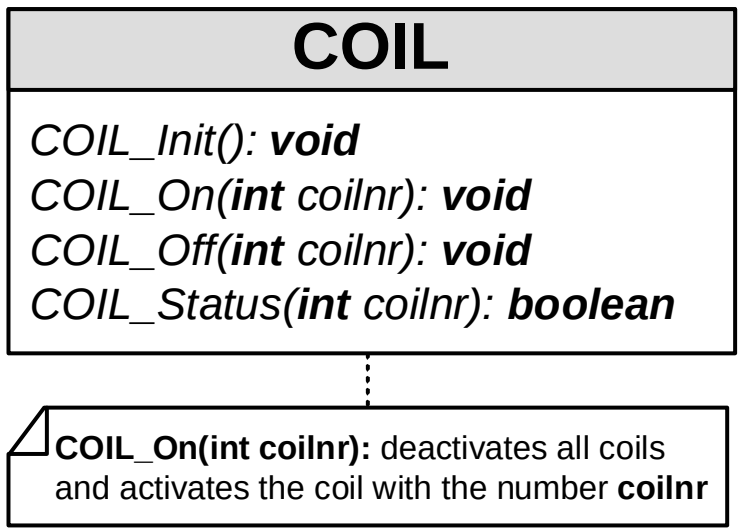
Wilhelm Haas
Wilhelm.Haas@informatik.stud.uni-erlangen.de

Friedrich-Alexander-Universität Erlangen-Nürnberg
Institut für Informatik
Lehrstuhl 4

9. Juni 2006

Überblick

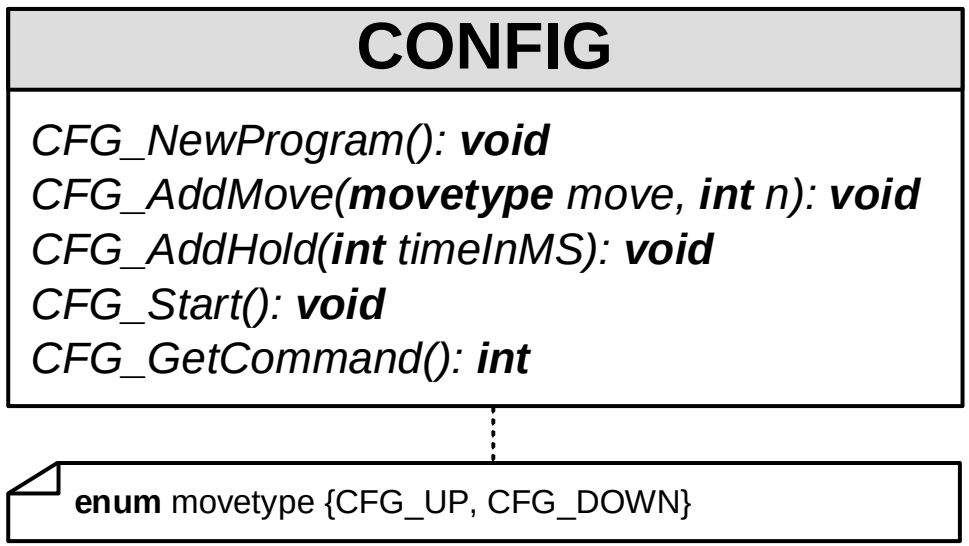

GPIO
GPIO_Config(int port, int pin, iotype flag): boolean
GPIO_Write(int port, int pin, boolean value): void
GPIO_Read(int port, int pin): boolean
enum iotype {GPIO_INPUT, GPIO_OUTPUT}

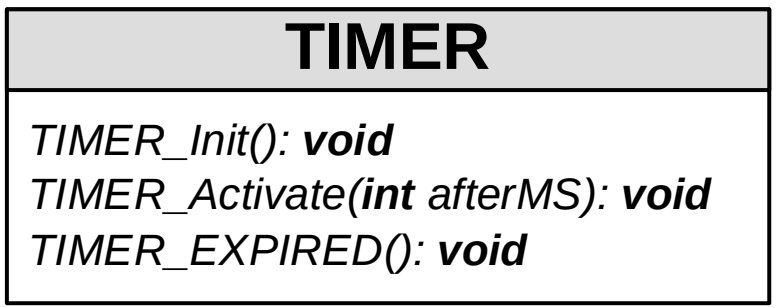

COIL
COIL_Init(): void
COIL_On(int coilnr): void
COIL_Off(int coilnr): void
COIL_Status(int coilnr): boolean
COIL_On(int coilnr): deactivates all coils and activates the coil with the number coilnr

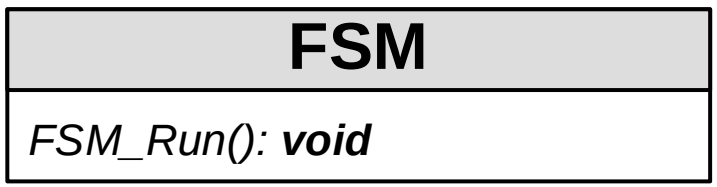
PhotoSensor

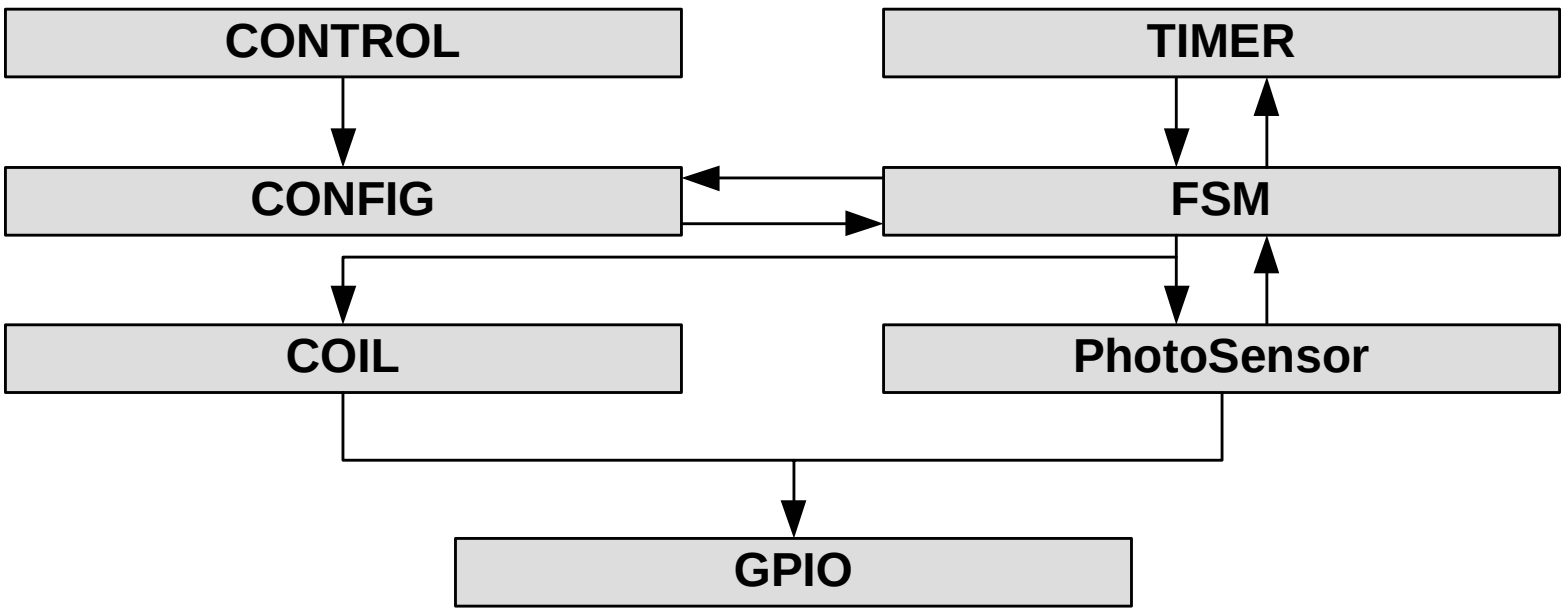
*PS_Init(): **void***
*PS_GetStatus(**int** psnr): **boolean***
*PS_SetEnableDisable(**int** psnr, **boolean** state): **void***
*PS_BREAK(): **void***

CONTROL
*CTRL_UpContinuous(**int** n): **void*** *CTRL_DownContinuous(**int** n): **void*** *CTRL_UpStepwise(**int** n): **void*** *CTRL_DownStepwise(**int** n): **void*** *CTRL_OscContinuous(**int** m, **int** n, **int** rounds): **void*** *CTRL_OscStepwise(**int** m, **int** n, **int** rounds): **void*** *CTRL_NewProgram(): **void*** *CTRL_Start(): **void***

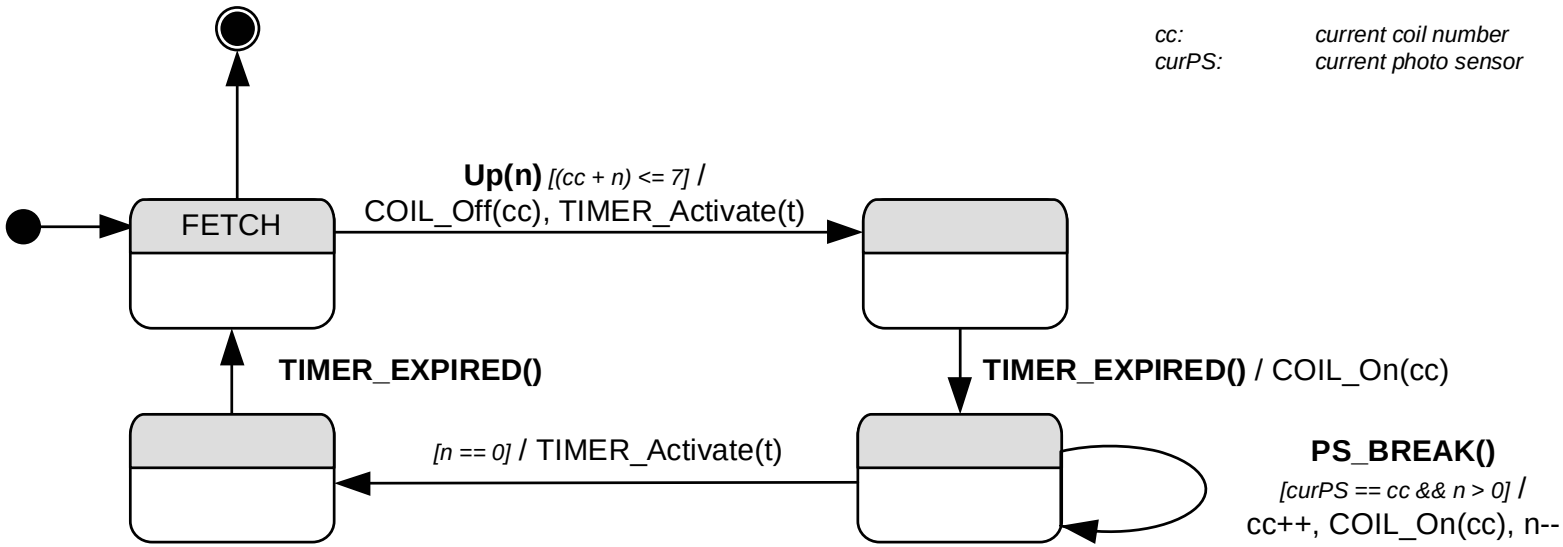

CONFIG
CFG_NewProgram(): void
CFG_AddMove(movetype move, int n): void
CFG_AddHold(int timeInMS): void
CFG_Start(): void
CFG_GetCommand(): int
enum movetype {CFG_UP, CFG_DOWN}


TIMER
TIMER_Init(): void
TIMER_Activate(int afterMS): void
TIMER_EXPIRED(): void


FSM
FSM_Run(): void


CONTROL
TIMER
CONFIG
FSM
COIL
PhotoSensor
GPIO

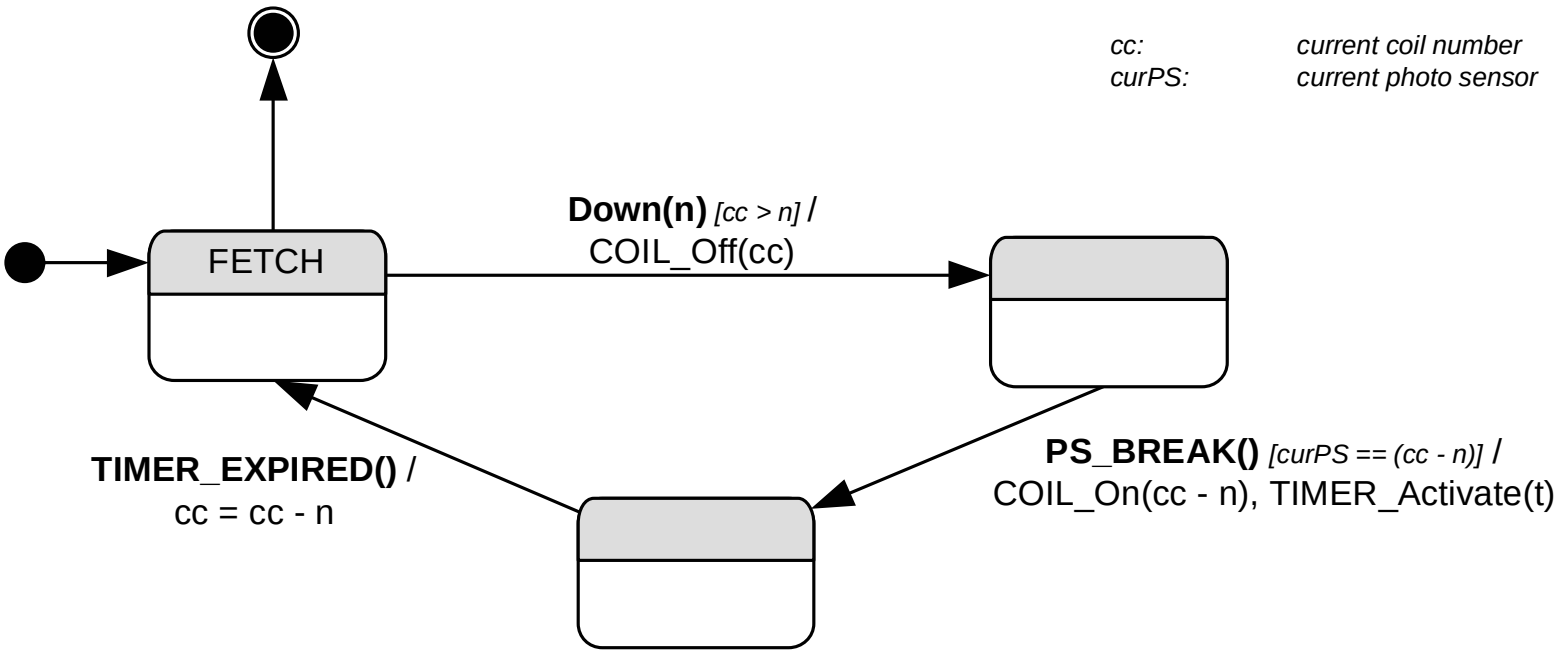
Überblick

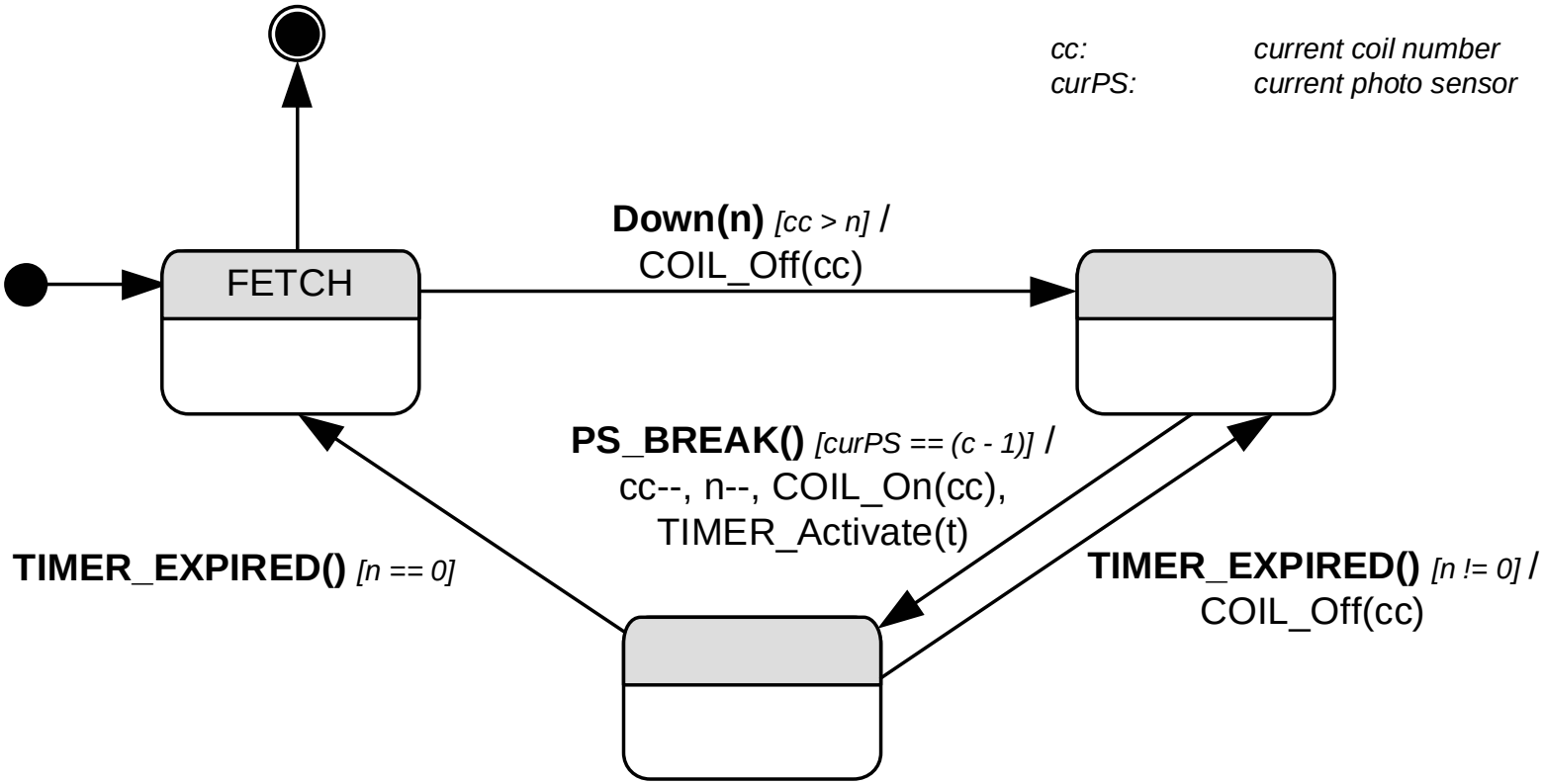

cc: current coil number
curPS: current photo sensor
FETCH
Up(n) [(cc + n) <= 7] /
COIL_Off(cc), TIMER_Activate(t)
TIMER_EXPIRED()
TIMER_EXPIRED() / COIL_On(cc)
[n == 0] / TIMER_Activate(t)
PS_BREAK()
[curPS == cc && n > 0] /
cc++, COIL_On(cc), n--

- ist eigentlich kein Spezialfall
- kann durch einzelne UP-Continous aufgebaut werden

```
function up_stepwise(n)
    for i = 1 to n
        up_continous(1)
    end for
end function
```

DOWN-Continuous




cc: current coil number
curPS: current photo sensor
FETCH
Down(n) [cc > n] /
COIL_Off(cc)
PS_BREAK() [curPS == (c - 1)] /
cc--, n--, COIL_On(cc),
TIMER_Activate(t)
TIMER_EXPIRED() [n == 0]
TIMER_EXPIRED() [n != 0] /
COIL_Off(cc)

Überblick

- ist Besitzer von zwei Ereignisobjekten
  - **TASK-EVENT**
    Wird verwendet um Signale vom Timer an den TASK zu schicken. Timer-Interrupt (*TIMER_EXPIRED()*) setzt das Ereignis wenn eine bestimmte Zeit abgelaufen ist.
  - **PS-EVENT**
    Wird verwendet um Signale vom PhotoSensor an den TASK zu schicken. PS-Interrupt (*PS_BREAK()*) setzt das Ereignis wenn eine Lichtschranke unterbrochen wird.
- Führt *FSM_Run()* aus.

- Dekrementiert einen Counter nach einer vorher statisch festgelegten Zeit.
- Erreicht der Counter den Wert 0, dann wird *TIMER_EXPIRED()* aufgerufen.
- *TIMER_EXPIRED()* setzt das *TIMER-EVENT*.

- Wird aktiviert wenn eine Lichtschranke unterbrochen wird.
- Ruft *PS_BREAK()* auf.
- *PS_BREAK()* setzt das *PS-EVENT*.

Überblick

- *Wartezeit*
  - Ist die Wartezeit zu gross, dann wirken die Bewegungen abgehakt.
  - Ist die Wartezeit zu klein, dann kann das bei der Bewegung nach unten zur Beschleunigung des Projektils, statt zur Abbremsung, führen.
- *PS_BREAK()* sollte nur aktiviert werden, wenn eine bestimmte, vorher festgelegte, Lichtschranke unterbrochen wird.