

Intro Summary

Inhalt

Intro Summary	1
Systems & Real-time	2
Version Control Systems (VCS)	2
ANSI-C & Macros.....	3
Plattformunabhängiger LED Driver	3
Header- & Sourcefiles	3
DoxyGen & Graphviz	4
Synchronization.....	4
Interrupt & Reentrancy	5
Shell.....	6
Mealy	7
Events.....	8
Keys.....	9
Processor Expert	9
Clock / Timer	10
Trigger	10
Debounce	12
RTOS / FreeRTOS	13
FreeRTOS.....	14
Queue	16
Semaphoren.....	17
FreeRTOS Trace	18
Motor Signals / Trace	18
Quadrature Encoder.....	19
Tacho.....	20
Closed Loop Control (PID).....	21
Accelerometer	23
LCD	24
Radio Transceiver	27
Remote Motor Controller.....	29
Low Power	30
Hardware.....	32
IEEE (not MEP).....	35
ZigBee (not MEP).....	39
Laboratory Short Courses.....	41

Intro Summary

Systems & Real-time

-Systems

- Transforming: ein Input erzeugt einen Output (Video Encoder, Router, Switch)
 - Process quality, Datendurchsatz, Effiziente Hardware Ausnutzung nötig
- Reactive: Ein System reagiert auf einen Input (Regler, Anti-Blockier-System)
 - Gesteuert von externen Events, muss Antwortzeit garantieren
- Interactive: Interaktion mit User: Eingabe => Ausgabe... (HMI, Billetautomat)
 - Oft grosse Systemauslastung, kurze Antwortzeit
- Combination: oftmals sind Kombinationen aus den ersten drei Systemen

-Classification: Welche Systeme sind beispielsweise wie in einem Handy vorhanden?

Transforming: GSM-Data to SMS-Text, Reactive: Tippen auf Tastatur, erzeugt Text auf LCD, Interactive: HMI

-Realtime: Ein System reagiert auf Events aus der „wirklichen“ Welt.

- Antwortet immer korrekt
- Antwortet zur richtigen Zeit
- Antwortet unabhängig von der Systemlast (unter allen Bedingungen)
- Antwortet deterministisch (definiert und reproduzierbar) und vorhersehbar
- => Hard Realtime: Antwort ist akzeptiert, wenn das Resultat innerhalb der vorgegebenen Zeit korrekt ankommt
- => Soft Realtime: Resultat wird degradiert, je nach Zeit wenn es ankommt

-Timeliness: Faktor Zeit ist immer absolut (Wecker) oder relativ (Airbag) zu betrachten.

-Concurrency: Computersysteme sind sequentiell, Realität ist nebenläufig!

Für Systeme mit wenigen und langsamen Tasks ist Nebenläufigkeit realisierbar

-Reaction time: Echtzeitsysteme benötigen eine definierte Reaktionszeit

-Process quality: kein Informationsverlust, kurze Verzögerungszeiten

⇒ **Systeme einteilen und beschreiben können. Script S. 68.. /78..**

Version Control Systems (VCS)

-Concepts of Version Control Systems:

- Goal: Backup, go back if things break, share files / folders with lots of developers, synchronization, track changes, branch and merge files
- Funktion: Ein Benutzer verbindet sich mit einem Server, auf dem Server läuft ein Versionsverwaltungssystem (VCS) welches den Zugriff auf eine Datenbank (Repository) steuert. Der Benutzer holt sich eine Kopie des Projekts auf seinen Rechner (check-out) und arbeitet daran. Nach abgeschlossener Arbeit lädt er das Projekt auf den Server (commit). Jede Änderung erzeugt dabei eine neue Revision und stellt so sicher, dass alle Änderungen ersichtlich bleiben. Wird später weiter daran gearbeitet, muss der Benutzer das Projekt zuerst aktualisieren (update).
- Optimistic approach: Das System geht davon aus, dass selten Änderungen gemacht werden, der Benutzer muss Files mergen
- Pessimistic approach: Das System blockt das File für alle anderen Benutzer wenn einer eine Änderung vornimmt. (Achtung bei Files die abhängig sind voneinander!)
- Branching: Eine Kopie des Projektes auf dem Server erstellen, welche parallel zum Original erweitert wird. => sinnvoll bei grossen Anpassungen
- Merging: Kopie und Original auf dem Server zusammenfügen (heikel!)
- Conflict: Zwei Benutzer holen sich eine Kopie vom Server. Der eine macht eine Änderung und committed, der andere macht auch eine lokale Änderung auf derselben Zeile und will nach dem anderen commiten. Beide haben am selben was geändert => Konflikt
- **Script S. 247..**

ANSI-C & Macros

- Reasons: Namen anstatt Nummern, Konfiguration, Portabilität, Codeoptimierungen
⇒ Preprocessor macht einen textuellen Austausch => '\ ' über mehrere Zeilen
- configuration:

```
#define PL_HAS_LED (1) // ohne ;
#define PL_HAS_LED
LED_Init();
#endif
```
- portability:

```
#define ENABLE_INTERRUPTS __asm CLI; // Assembler immer in Makros
```
- Eigenschaften: + schnellerer Code + weniger Code (kein Funktionssprung nötig)
-Interface wird benötigt -schlechtere Kapslung -schwierigeres Debugging
⇒ Kompromiss zwischen Methoden und Makros
- traps and pitfalls: An Klammern denken (z.B. Punkt vor Strich), Vorsicht bei Deklarationen (a=0; ...), Reihenfolge von Befehlen beachten

Plattformunabhängiger LED Driver

-HW: immer mit Vorwiderstand, μ C können einfacher auf GND ziehen (=Kathode an μ C)

Implementation: LED.h

```
#if PL_IS_TOWER_BOARD
    #if PL_NOF_LED >= 1
        #include "LED1.h" // PE File
        #define LED1_On() (LED1_ClrVal()) // Cathode GND
        #define LED_ON(nr) (LED##nr##_SetVal()) //
    ...
    #if PL_NOF_LED >= 2
        #include "LED2.h"
    ...
    #endif
#endif
```

Alternative

! Beim SRB Board kann es sein, dass die LED mit SetVal() eingeschaltet wird (Anode GND).

Header- & Sourcefiles

- definition:
 - ⇒ Deklaration: Sichtbarmachen des Namens
 - ⇒ Definition: Speicher allozieren, Funktionalität festlegen
- ```
/* drv.c */
#include "drv.h"

int DRV_global = 7;
static int v;

void DRV_Init(void) {
 v = 3;
 DRV_global += v;
}
```

```
/* drv.h */
#ifndef __DRV_H__
#define __DRV_H__

extern int DRV_global;

void DRV_Init(void);

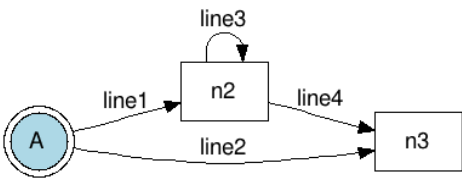
#endif /* __DRV_H__ */
```

```
/* main.c */
#include "drv.h"

void main(void) {
 DRV_Init();
}
```
- ⇒ #include: Textuelle Einbindung verschiedener Files
  - ⇒ Schutz vor rekursivem / mehrfachem Einbinden mit **#ifndef**, **#define**, **#endif**
  - ⇒ In Interfaces nur was nötig ist, nicht mehr und nicht weniger!
  - ⇒ Interfaces müssen geklappelt sein, sodass die Benutzer nur ein File einbinden müssen

-Reihenfolge des Einbindens: Wichtig ist, dass zuerst die Parameter initialisiert werden (**#define** TRUE 1), bevor sie verwendet werden (**#if** TRUE ...).

## DoxyGen & Graphviz

|                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -Documentation from Source Files: | + Compiler erkennt spezielle Kommentar-Tags und erstellt die Doku anhand der Source Files                                                                                                                                                                                                                                                                                                                                                      |
|                                   | + Automatische Aktualisierung der Dokumentation                                                                                                                                                                                                                                                                                                                                                                                                |
| -Dependencies:                    | mit Graphviz ist es möglich darzustellen, welche Files wie zusammenhängen                                                                                                                                                                                                                                                                                                                                                                      |
| -Settings:                        | Sind im myProject.doxyfile definiert (z.B. Input- und Output-Files)                                                                                                                                                                                                                                                                                                                                                                            |
| -Syntax:                          | <pre> /** text */ oder /*! text */ bei jedem DoxyGen Kommentar \file          bindet ohne weitere Angaben den Filenamen ein \author        [Name] \brief         kurz und bündig was die Methode macht \param         [Parameter] [Beschreibung] \todo          [tolle Beschreibung zu offene Tasks] =&gt; Todo Liste wird erzeugt \return        [Beschreibung] </pre>                                                                        |
| -Grafik zeichnen                  | <pre> \dot digraph myfsm {     rankdir=LR;     node [shape=doublecircle];     n1 [fillcolor=lightblue,style=filled,label="A" ];     node [shape=box]; n2 n3;     n1 -&gt; n2 [label="line1"];     n1 -&gt; n3 [label="line2"];     n2 -&gt; n2 [label="line3"];     n2 -&gt; n3 [label="line4"]; } </pre> <p>// L = left, R = right, T = top, D = down</p>  |
| -Grafik einbinden                 | <pre> \image html Led.jpg [format] [file] </pre>                                                                                                                                                                                                                                                                                                                                                                                               |

## Synchronization

|                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -Why:                                 | Werden vom einen System in ein anderes System Daten übertragen, so muss sichergestellt werden, dass der Sender nicht zu schnell und nicht zu langsam überträgt, sodass der Empfänger die Daten verstehen kann.<br>⇒ Timing ist entscheidend, also ist eine Synchronisation nötig                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| -computation speed(Abarbeitungszeit): | zu langsam => FAIL, zu schnell => Synchronisation möglich                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| -Different kinds of synchronisation:  | <ul style="list-style-type: none"> <li>• <b>Realtime:</b> es wird eine bestimmte Zeit gewartet (for loops) und dann wird gestartet<br/>=&gt; man hofft, dass alles in Ordnung ist nach dem Delay von z.B. 5ms ☺ <ul style="list-style-type: none"> <li>○ + keine Hardware nötig</li> <li>○ – ineffizient da der Controller warten muss</li> <li>○ – je nach Compiler anderes Delay =&gt; abhängig von Clockrate und Controller (Compiler darf nicht optimieren, inline assembler oder pragmas verwenden)</li> </ul> </li> <li>• <b>Gadfly:</b> Ein Flag wird aktiv, wenn ein Prozess bereit ist. Auf dieses Flag wird ständig gepollt und so gewartet, bis der andere Prozess auch bereit ist. <ul style="list-style-type: none"> <li>○ + einfach + geringe Latenzzeit</li> <li>○ – evtl. HW Flag nötig</li> <li>○ – benötigt viel Performance / Ständig Pollen = Volllast</li> </ul> </li> <li>• <b>Interrupt:</b> Der Prozess wird unterbrochen sobald der andere Prozess bereit ist. Aufpassen bei gemeinsamen Ressourcen (Code, Data)! <ul style="list-style-type: none"> <li>○ + effizient weil nicht gewartet werden muss</li> <li>○ + weniger Overhead im Code</li> <li>○ – muss HW-mässig unterstützt sein</li> <li>○ – Programmstatus muss vor dem Unterbruch gesichert werden</li> <li>○ – je nach dem benötigt die Abarbeitung einer ISR länger als ein paar NOPs</li> </ul> </li> </ul> |
| -Comparison:                          | Die Entscheidung muss von Fall zu Fall abgewogen werden                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

## Interrupt & Reentrancy

- Interrupt Source: Es gibt eine Vielzahl an Interrupt Quellen. Die meisten kommen von internen Modulen wie Timer oder ADC oder von externen IRQ-Pins. Neben den Hardware gibt es auch einen Softwareinterrupt (SWI).
- Interrupt Sequence: Nach der Initialisierung, der Aktivierung und der Maskierung der Interrupts wird diese Sequenz beim Auftritt eines Interrupt ausgeführt:

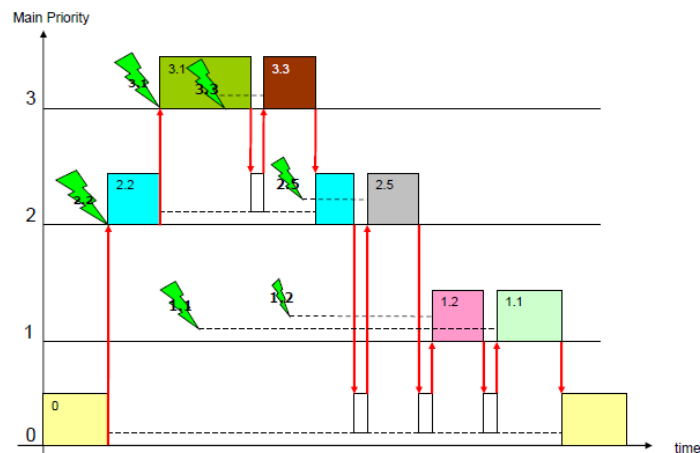
- Aktuelle Instruktion abarbeiten
- Nachschlagen der ISR-Adresse in der Interrupt Vektor Tabelle
- Rücksprungadresse und CPU Register auf dem Stack sichern
- SEI (Interrupts deaktivieren)
- Ausführen der ISR
- Register vom Stack wiederherstellen
- CLI (Interrupts aktivieren)
- Rücksprung zur nächsten Instruktion

-nice to know:

- ISR möglichst kurz halten, keine grösseren Funktionen in den ISRs. Besser ist es mit Steuersignalen (volatile flag) das Hauptprogramm zu beeinflussen => Latency

- **nested interrupts:**

Es gibt Interrupts mit verschiedenen Prioritäten. Interrupts mit höheren Prios (ob 0 oder 1 höher ist ist abhängig vom System) können die ISR von Interrupts mit tieferen Prios unterbrechen. Wird dies nicht unterstützt, wird die ISR des anderen Interrupts nach dem Rücksprung ausgeführt.



- Haben ISR und Hauptprogramm **gemeinsame Ressourcen**, so muss der Zugriff geschützt werden (atomar!), z.B. mit SEI/CLI, Enter-/ExitCritical oder mit Semaphoren
- Implementierung ist controllerabhängig (fixe/ variable ISR-Grösse, Adressierung, ..)
- Debugging ist oftmals schwierig, da gewisse Zustände selten Auftreten können.

-**Reentrancy:** Jede Methode die in einer ISR oder im Hauptprogramm aufgerufen werden kann muss reentrant (mehrfach ausführbar, ohne dass sie sich gegenseitig beeinflussen) sein. => reentrant != rekursiv

-Vorgehen ISR-Implementation:

- Allgemein
1. Interrupt Vektor initialisieren (Vector Table)
  2. Vor dem Aktivieren alle benötigten Interrupt-Flags löschen
  3. Interrupt-Quelle aktivieren
  4. Hauptapplikation läuft in einer Endlos-Schleife

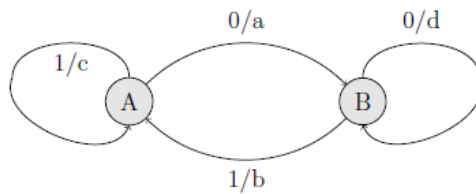
- ISR
1. Interrupt-Flag löschen
  2. Falls Interrupts mit höheren Prioritäten diesen Interrupt unterbrechen dürfen, müssen die Interrupts mit tieferen Prioritäten deaktiviert werden, falls nicht sollen alle Interrupts deaktiviert werden.
  3. ISR Funktion ausführen
  4. Aktivieren der inaktiven Interrupts (falls überhaupt deaktiviert)
  5. Von der ISR zurückspringen

Script S. 311



## Mealy

-state machines: Ereignisgesteuerte Aktionsfolgen



-description: Im Zustand 'A' wird der Input '0' empfangen. Die Statemaschine gibt ,a' aus und geht in den Zustand ,B':  $A \Rightarrow 0/a \Rightarrow B$

Zwei Zustände: A, B    Zwei Inputs: 0,1    Vier Outputs: a,b,c,d

-implementation:

```

Typedef enum { A = 0, B = 1 } StateT;
Static StateT state;

```

Conventional:    `if(state == A)`    *// einfach, für wenig states*

```

uint8_t r = Read();
if(r == 0)
 Write(a);
 state = B;
else if(r == 1)
 Write(c);
 state = A;
...

```

Switch:    `r = Read();`    *// kompakt, für wenig states*

```

switch(state)
case A:
 if(r==0)
 Write(a);
 State = B;
 else if(r==1)
 Write(c);
 state = B;
 break;
case B: ...

```

Table:    *// sehr kompakt, etwas schwieriger zu verstehen*

| State | 0   | 1   |
|-------|-----|-----|
| A     | B/a | A/c |
| B     | B/d | A/b |

Zuerst sollte die Tabelle gezeichnet werden.

```

Static uint8_t tbl[2][2][2] = {
 /* curr State r = 0 r = 1 */
 /* A */ {{B,a}, {A,c}},
 /* B */ {{B,d}, {A,b}}};
r = Read();
Write(tbl[state][r][1]);
state = tbl[state][r][0];

```

Remember 3D Array:

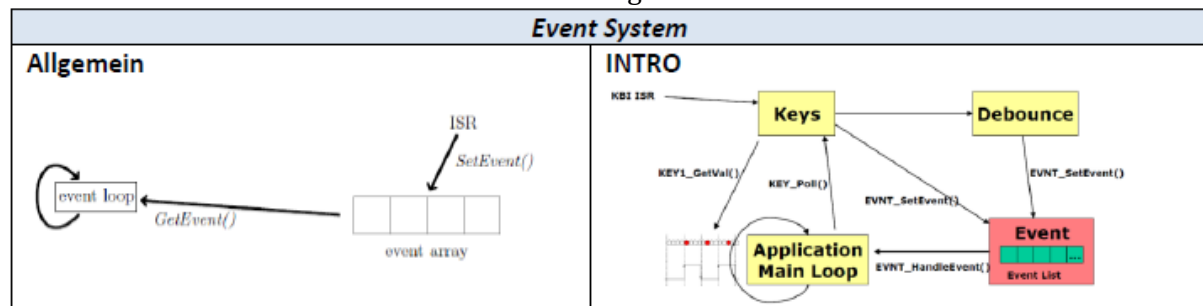
```

int numbers [2][3][4] = {{{1,2,3,4},{5,6,7,8},{9,10,11,12}}
,{{13,14,15,16},{17,18,19,20},{21,22,23,24}}};

```

## Events

- description: Eine Möglichkeit wichtige Dinge schnell abzuarbeiten und Dinge die warten können aufzuschieben. Nützlich im Zusammenhang mit Interrupts, da die ISR nur ein EventBit setzen muss. Den grossen Aufwand hat dann der Eventhandler.



-Create Eventhandle:

- `#define EVENT_INIT 0` // Initialization Event
- `#define EVENT_SW1 1` // SW1 pressed Event
- `#define EVENT_NOF_EVENTS 2` // Number of different Events
- `typedef enum EventHandle {`
  - `EVENT_INIT,` // Initialization Event
  - `EVENT_SW1,` // SW1 pressed Event
  - `...`
  - `EVENT_NOF_EVENTS` // always last one, number of diff. Events

-Event Data Structure:

```
static uint8_t Events[((EVENT_NOF_EVENTS-1)/8)+1];
// Array mit 1nem Bit pro Event zum Setzen und Löschen von Events
```

-Event setzen:

```
void SetEvent(EventHandle event)
{
 EnterCritical();
 Events[event/8] |= 0x80 >> (event % 8);
 ExitCritical();
}
```

-Event löschen:

```
void ClrEvent(EventHandle event)
{
 EnterCritical();
 Events[event/8] &= ~(0x80 >> (event % 8));
 ExitCritical();
}
```

-Event abfragen:

```
void GetEvent(EventHandle event)
{
 bool isSet;
 EnterCritical();
 isSet = Events[event/8] & (0x80 >> (event % 8));
 ExitCritical();
}
```

-Event Handling:

```
void HandleEvent (void (*callback) (EventHandle))
{
 uint8_t event; // iterates through the event bits
 EnterCritical();
 for(event = 0; event < EVENT_NOF_EVENTS; event++) {
 if(GetEvent(event) { // Event Bit set?
 ClrEvent(); // Clear Event Bit
 break; // Get out of loop
 }
 }
 ExitCritical();
 if (event != EVENT_NOF_EVENTS) {
 callback(event); // go to callback
 }
}
```



-Integration: 

```
void main (void)
{
 SetEvent(EVENT_INIT); // Init Event on startup
 for(;;) {
 HandleEvent (APP_HandleEvent); // callback übergeben
 }
}
```

-Auswerten: 

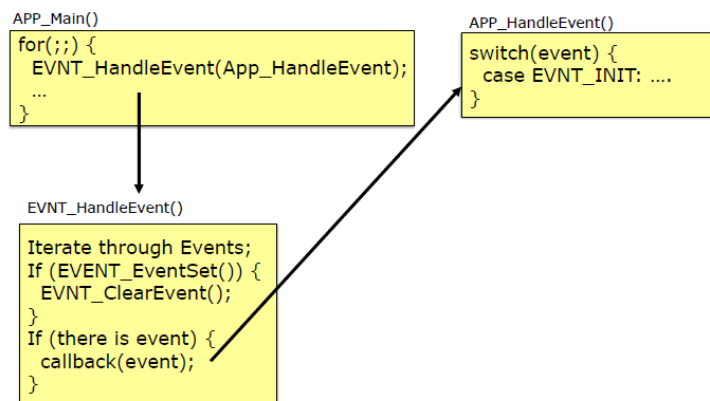
```
void APP_HandleEvent (EventHandle event)
{
 switch (event) {
 case EVENT_INIT:
 doSomething(); break;
 case EVENT_SW1:
 doSomething(); break;
 ..
 }
}
```

-Auslösen: 

```
void interrupt KeyISR(void)
{
 clearKeyISRFlag();
 if (Key1Pressed()) {
 SetEvent(EVENT_SW1);
 } elseif ..
}
```

-Priorities: Durch die Nummerierung ist es möglich eine Priorisierung einzuführen, je tiefer die Eventnummer, desto höher die Priorität.

-Reentrancy: Den Zugriff auf die gemeinsame Datenstruktur muss geschützt werden!



## Keys

- pullup/pulldown resistors: Zum Verhindern von undefinierten Zuständen (intern / extern)
- config: interne Register (pullup enable, input / output,
- synchronisation: Realtime, Gadget, Interrupt

Zum Beispiel Interrupt in Events.c:

```
void KBI1_OnInterrupt(void) {
 keyPressed = true; /* Key has been pressed */
 key = KBI1_GetVal(); /* Key code */
}
```

-Driver: Wie bei den LEDs müssen die richtigen Funktionen zugewiesen werden.

## Processor Expert

- Softwareabstraktion, welche die Hardware Anbindung erleichtert
- Einfacher plattformunabhängig zu programmieren.

## Clock / Timer

-clock: In einem Mikrocontrollersystem sind immer mehrere Clocks vorhanden:  
- Systemclock - CPU Clock - Bus Clock - ...  
Je nach System können diese Clocks gleich oder unterschiedlich sein. Die Clocks sind basierend auf dem internen Oszillator oder einem externen Quarz / Oszillator. Mit Hilfe von einem Frequency Locked Loop (FLL) können die Frequenzen skaliert werden.

-timer: Mit einem Timer kann ein periodischer Ablauf gesteuert werden. Für Real-Time Systeme ist dies sehr wichtig (bspw. periodische Ticks, die alle 10ms einen Interrupt auslösen).

Benötigt wird: -Zeitbasis, -Clock, -Synchronisierte Interrupts

-implementation:

```
#define TMR_TICK_INTERVALL 10 // Tick periodisch alle 10ms
void TMR_On10ms(void) { // wird von der ISR des Timers aufgerufen
 static uint8_t cnt = 0; // akzeptabel, da nur wenig Overhead!
 cnt++;
 if(cnt == 1000/TMR_TICK_INTERVALL) {
 SetEvent(EVENT_LEDBlink); // jede Sekunde wird ein
 } // Event ausgelöst.
 cnt = 0;
}
```

LEDBlink-

## Trigger

-description: Trigger basieren auf den periodischen Ticks von Timern. Das Ziel ist es, das bisherige System so zu erweitern, dass einfach Events gefeuert und kompliziertere Abläufe eingebunden werden können. Bspw. soll es möglich sein, 500ms nach dem Drücken eines Tasters eine LED einzuschalten. Zudem muss verhindert werden, dass die ISR des Timers überladen wird.

-goal: -universelles Interface -Verwendung eines einzigen Timers  
-minimaler Speicherbedarf

-Trigger zählen in ISR:

```
void TMR_On10ms(void) {
 TRG_IncTick(); // siehe Timer
}
```

-Triggers:

```
typedef enum {
 /*! \todo Extend the list of triggers as needed */
 TRG_HEARTBEAT=0,
 TRG_KEYPRESS,
 TRG_BTNSND_OFF,
 TRG_BTNSND_ON,
 TRG_NOF_TRIGGER // last one
} TRG_TriggerKind;
```

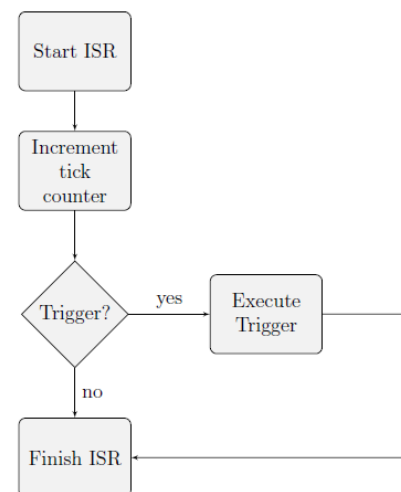
-Generische Struktur:

```
typedef void *TRG_CallbackDataPtr;
typedef void (*TRG_Callback)(TRG_CallbackDataPtr);
typedef uint16_t TRG_TriggerTime;

typedef struct {
 TRG_TriggerTime triggerTick;
 TRG_Callback callback;
 void *data;
} TRG_TriggerDesc;
```

-Trigger Array:

```
static TRG_TriggerDesc TriggerList[TRG_NOF_TRIGGER];
```



-Set Trigger:

```
uint8_t TRG_SetTrigger(TRG_TriggerKind trigger, TRG_TriggerTime ticks,
 TRG_Callback callback, TRG_CallbackDataPtr data) {
 EnterCritical();
 TriggerList[trigger].triggerTick = ticks;
 TriggerList[trigger].callback = callback;
 TriggerList[trigger].data = data;
 ExitCritical();
 return ERR_OK;
}
```

-Inc Ticks:

```
void TRG_IncTick(void) {
 TRG_TriggerKind trg = (TRG_TriggerKind) 0;
 EnterCritical();
 for (trg = (TRG_TriggerKind) 0; trg < TRG_NOF_TRIGGER; trg++) {
 if (TriggerList[trg].triggerTick > (TRG_TriggerTime) 0) {
 TriggerList[trg].triggerTick--;
 }
 }
 ExitCritical();
 while (checkCallback()) {} // Wird so oft ausgeführt bis keine Callbacks
 // mehr ausgeführt worden sind
}

static bool checkCallback(void) {
 TRG_TriggerKind trg = (TRG_TriggerKind) 0;
 TRG_Callback callback = (TRG_Callback) 0;
 TRG_CallbackDataPtr data = (TRG_CallbackDataPtr) 0;
 bool retVal = FALSE;

 for (trg = (TRG_TriggerKind) 0; trg < TRG_NOF_TRIGGER; trg++) {
 EnterCritical();
 if (TriggerList[trg].triggerTick == 0 &&
 TriggerList[trg].callback != (TRG_Callback) NULL) {
 callback = TriggerList[trg].callback; // backup
 data = TriggerList[trg].data; // backup
 TriggerList[trg].callback = (TRG_Callback) NULL;
 ExitCritical();
 callback(data);
 retVal = TRUE;
 } else {
 ExitCritical();
 }
 }
 return retVal;
}
```

-Aufruf:

```
void main(void) {
 TMR_Init();
 TRG_Init();
 EnableInterrupts();
 TRG_SetTrigger(TRG_HEARTBEAT, 0, LED_HeartBeat, NULL); // set now
 for(;;) {} // an der Stelle von NULL geht auch ein Parameter

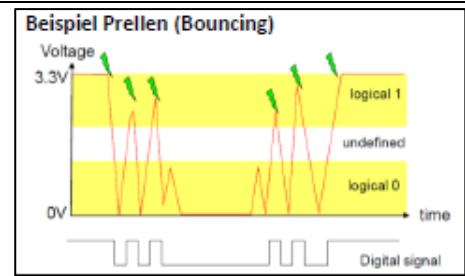
 Static void LED_HeartBeat(void *p) {
 LED1_Neg(); // toggles LED1
 TRG_SetTrigger(TRG_HEARTBEAT, 1000/TRG_TICKS_MS, LED_HeartBeat, null);
 } // call LED_HeartBeat again in 1sec
}
```

Callback Funktion

Pointer auf Parameter

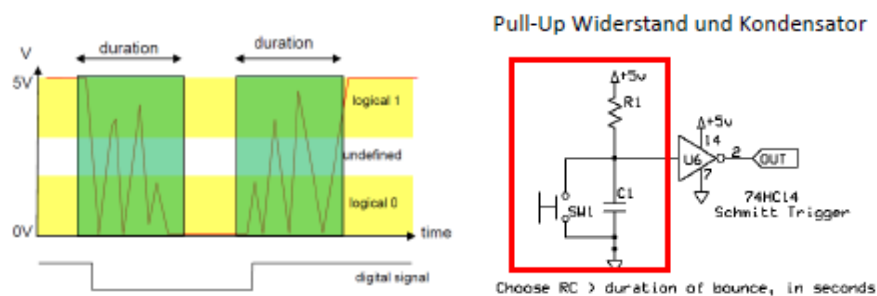
## Debounce

-description: Tasten können beim Drücken oder beim Loslassen prellen und so, anstatt eines einzigen Interrupts gleich mehrere Interrupts auslösen. Debouncing wirkt dieser Problematik entgegen.



-solution: Es wird während der Zeit in der der Schalter prellt der Input nicht mehr abgefragt (Software) oder die entstehenden Signalpeaks gefiltert (Hardware). Die zu überbrückende Zeit muss empirisch ermittelt werden.

-hardware:



-software:

### State Machine & Trigger



Da die ISR möglichst kurz sein muss, sind alle Zustandsübergänge mit Trigger implementiert. Der Interrupt löst lediglich den Startschuss aus.

- Alles muss reentrant sein
- Es vergehen einige Zyklen zwischen dem Tasten drücken und dem Setzen des Triggers => Verzögerungen / Timing!
- 

-implementation:

```
void KBI_OnInterrupt(void) {
 if(keyPressed() && DebounceFSM.state = IDLE) {
 DebounceProcess (DebounceData *data);
 // switch case Struktur (siehe Mealy) die den neuen Status setzt
 // (PRESSED) und einen Trigger mit der Callback DebounceProcess
 // nach der nötigen Debounce-Zeit setzt.
 }
}
```

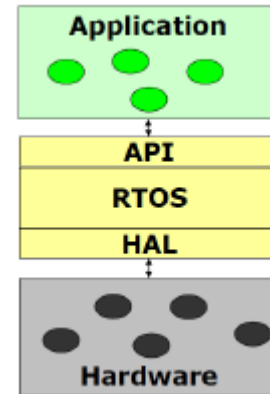
- Da in DebounceProcess nur ein paar Daten überschrieben und ein Trigger gesetzt wird ist der Code klein genug um in der ISR abgearbeitet zu werden.

## RTOS / FreeRTOS

- Anforderungen:
- Vorhersehbarkeit: maximale Ausführungszeit muss bekannt sein, deterministisch: alles sollte über das OS laufen
  - Präzises Timing: alles basiert auf einem genauer Tick Timer
  - Geschwindigkeit: OS setzt eine hohe Performance voraus

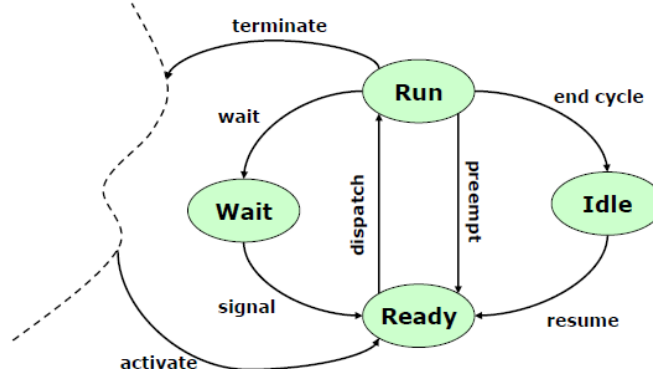
- Motivation: Stellt viele Services und Funktionen zur Verfügung (z.B. zur Synchronisation) welche komfortables und skalierbares Programmieren ermöglichen.

- Services:
- Bündelung von Ressourcen
  - Prozesse/Threads (Scheduler, Synchronisation, ...)
  - HW Ressourcen (Zugriffskontrollen, Real Time Clock)
  - Memory Management ("dynamische" Allokierung, Heap, ...)
  - Dateisystem (File I/O)
  - Kommunikation (GUI, Protocol stacks, Inter-Prozess-Kommunikation)



- nice-to-know: Um die Realtime Anforderungen erfüllen zu können, wird mit einem RTOS direkt auf die Hardware zugegriffen.

- OS State Diagramm:



| Scheduling & Context Switch (Prozesswechsel)                                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                                                                                                                                                                                                                                                                                                                                                                                      | <p><u>No-Preemptive (nicht unterbrechend)</u><br/>Sobald dem Prozess die CPU zugeteilt wurde, wird der Prozess so lange laufen, bis er von selbst oder wenn er blockiert die CPU wieder freigibt.</p> <p><u>Preemptive (unterbrechend)</u><br/>Immer den höchsten verfügbaren Task ausführen. Tasks mit identischer Priorität teilen CPU-Zeit.</p>                                                                         |
| Auswahlkriterien für ein spezifisches RTOS                                                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <ul style="list-style-type: none"> <li>• Funktionalität (API, Stacks, Standards, ...)</li> <li>• Programmiersprachen</li> <li>• Real-time oder nicht?</li> <li>• Benötigte Ressourcen</li> <li>• Skalierbarkeit</li> <li>• Verfügbarkeit</li> <li>• Offenheit</li> <li>• Tools &amp; Support</li> <li>• Kosten- &amp; Lizenzierungsmodell</li> <li>• Statisch oder dynamisch?</li> <li>• Eigene Vorlieben</li> </ul> | <p><u>Nichtfunktionale Faktoren</u></p> <ul style="list-style-type: none"> <li>• Lernkurve</li> <li>• Synergien innerhalb der Firma</li> <li>• Informationen &amp; Dokumentation</li> <li>• Qualität (Produkt, Dienstleister, ...)</li> <li>• Zertifizierung (SIL, DOI, OSEK, ...)</li> <li>• Lizenzierungsmodell</li> <li>• Kosten für Evaluation, Einführung und Unterhalt</li> </ul> <p>→ Werden oftmals vergessen!</p> |

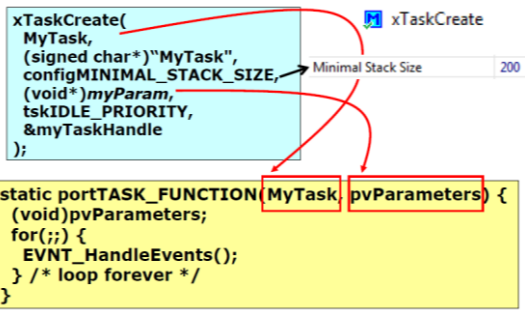
## FreeRTOS

|                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -Eigenschaften: | <ul style="list-style-type: none"> <li>- einfach, - portierbar, - lizenzfrei – kurzgefasst (nicht überladen)</li> <li>- enthält einen Micro Real-Time Kernel</li> <li>- Scheduler Grundsätze: <ul style="list-style-type: none"> <li>-Pre-emptive: höchste Prio wird ausgeführt</li> <li>-Cooperative: Prozesswechsel treten nur auf, wenn ein Task explizit die CPU abgibt.</li> </ul> </li> <li>- Services: Queues, Semaphoren, Timers, Trace, ...</li> <li>- Funktioniert mit verschiedensten Mikrocontrollern</li> </ul> |
| -RTOS kernel :  | verfügt über eine „mutiple priority list“ welches viele Freiheiten zulässt. Es können mehrere Tasks mit derselben Priorität laufen und es gibt zusätzliche Subprioritäten.                                                                                                                                                                                                                                                                                                                                                   |
| -Priorities:    | die kleinste Priorität (0) wird auch als tiefste Priorität gehandhabt.                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| -Configuration: | <ul style="list-style-type: none"> <li>- Software Interrupt auswählen (SWI)</li> <li>- Minimale Stackgrösse ( CPU-Bus*Einheiten = Bytes )</li> <li>- Heap Grösse ( in Byte)</li> <li>- Tickrate, im Normalfall 1Hz .. 1kHz</li> </ul>                                                                                                                                                                                                                                                                                        |

## -Memory Management

| FreeRTOS Memory Schemes<br>Speicher allozieren für bspw. Tasks / Queues / Semaphoren                            |                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Scheme 1                                                                                                        | Alloziert nur Speicher. Kein vTaskDelete(), vQueueDelete(), ...                                                                                       |
| Scheme 2                                                                                                        | Speicherblock kann freigegeben werden. Verbindet freie Speicherblöcke nicht<br>→ Fragmentierung möglich! Ungünstig für zufällige alloc/free Sequenzen |
| Scheme 3                                                                                                        | Scheme für standard malloc() und free()                                                                                                               |
| Scheme 4                                                                                                        | Verbindet freie Speicherblöcke                                                                                                                        |
| Malloc(), Free() und FreeHeap()                                                                                 |                                                                                                                                                       |
| <pre>void *pvPortMalloc(size_t xWantedSize); void vPortFree(void *pv); size_t xPortGetFreeHeapSize(void);</pre> | <p><u>Beispiel</u></p> <pre>bufP = (char_t*)pvPortMalloc(sizeof("Hello")); vPortFree(bufP);</pre>                                                     |

## -Tasks

| Task Creation (xTaskCreate(), xTaskDelete())                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | Create Task and start FreeRTOS Scheduler                                                                                                                                                                                                                                                                                                                              |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  <pre>xTaskCreate(     MyTask,     (signed char*)"MyTask",     configMINIMAL_STACK_SIZE,     (void*)myParam,     tskIDLE_PRIORITY,     &amp;myTaskHandle );  static portTASK_FUNCTION(MyTask, pvParameters) {     (void)pvParameters;     for(;;) {         EVNT_HandleEvents();     } /* loop forever */ }</pre> <p><b>ACHTUNG:</b> Die Minimal Stak Size wird in Elementen gerechnet, NICHT in Bytes! D.h. bei 32-Bit CPU und Stack Size = 200<br/>→ 200*(4*Bytes) = 800 Bytes auf Stack</p> | <pre>void RTOS_Run(void) {     if (FRTOS1_xTaskCreate(         MainTask, // Name des Tasks         (signed portCHAR*)"Main",         configMINIMAL_STACK_SIZE+100,         (void*)NULL,         tskIDLE_PRIORITY,         (xTaskHandle*)NULL     ) != pdPASS) {         for(;;){}; /* error! probably out of memory*/     }     FRTOS1_vTaskStartScheduler(); }</pre> |

| Task Switches passieren auf...                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• Tick Interrupt</li> <li>• taskYIELD</li> <li>• RTOS API call</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| FreeRTOS API                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>vTaskDelay(#ofTicks)</b><br>Der Task wartet nach der Ausführung so viele Ticks bis zur nächsten Ausführung.<br>Tick = 10ms, vTaskDelay(1..1,9) => 0..10ms Verzögerung<br>Ein Tick Verzögerung heisst weiter beim nächsten!                                                                                                                                                                                                                                                                                                                | <pre>void vTaskFunction(void *pvParameters) {     for(;;) {         /* toggle the LED every 500ms */         LED0_Neg();         EVNT_HandleEvent(APP_HandleEvent);         vTaskDelay(500/portTICK_RATE_MS);     } /* for */ }</pre>                                                                                                                                                                                                                                                                                                                                  |
| <b>vTaskDelayUntil(&amp;TickDerLetztenAusführung, #ofTicks)</b><br>Ähnlich zu vorheriger Funktion, nur wird zusätzlich am Anfang des Tasks der aktuelle Tick gespeichert und dann der Delay-Funktion übergeben. Kann bei Tasks mit Ausführungszeit > Tick[ms] genauere Periodendauern zwischen einzelnen Ausführungen ergeben.                                                                                                                                                                                                               | <pre>xLastWakeTime = xTaskGetTickCount();  vTaskDelayUntil( &amp;xLastWakeTime, 10 );</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| <u>Weitere Funktionen (S. 5-14 Folien SW6e):</u> <ul style="list-style-type: none"> <li>• vTaskStartScheduler(), vTaskEndScheduler()</li> <li>• uxTaskPriorityGet(), vTaskPrioritySet()</li> <li>• vTaskSuspendAll(), vTaskSuspend()</li> <li>• vTaskResumeAll(), vTaskResume(), vTaskResumeFromISR()</li> <li>• vTaskYIELD()</li> <li>• void taskENABLE/DISABLE_INTERRUPTS(void)</li> <li>• void taskENTER/EXIT_CRITICAL(void)</li> </ul>                                                                                                   | <u>Erläuterungen:</u> <ul style="list-style-type: none"> <li>• Scheduler starten / stoppen</li> <li>• Task Priorität modifizieren</li> <li>• Task unterbrechen</li> <li>• Zu Task zurückkehren und ausführen</li> <li>-nur „FromISR“ sind in ISRs erlaubt!</li> <li>• Erzeugt einen Task-Switch</li> <li>• Interrupt-Handling ohne Nesting</li> <li>• Interrupt-Handling mit Nesting</li> </ul>                                                                                                                                                                        |
| Idle Task                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Task Transition Diagramm                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <p>-Wird erstellt mit dem ersten Aufruf von xTaskCreate().</p> <p><u>Pseudo Code für Idle Task</u></p> <pre>static void prvIdleTask(void *pvParameters) {     for(;;) {         RemoveDeletedTasksFromList();         if (!configUSE_PREEMPTION) {             taskYIELD();         } else if (configIDLE_SHOULD_YIELD) {             if (NofReadyTasks(tskIDLE_PRIORITY)&gt;1) {                 taskYIELD();             }         }         IdleHook(); /* Ergänzung mit eigenem Code*/     } /*möglich, Delays sind verboten! */ }</pre> | <pre> graph TD     Suspended -- "vTaskSuspend() called" --&gt; Ready     Ready -- "vTaskResume() called" --&gt; Suspended     Ready -- "vTaskSuspend() called" --&gt; Suspended     Suspended -- "vTaskResume() called" --&gt; Ready     Ready -- "Event" --&gt; Blocked     Blocked -- "vTaskSuspend() called" --&gt; Suspended     Blocked -- "Blocking API function called" --&gt; Blocked     Blocked -- "vTaskResume() called" --&gt; Ready     Ready --&gt; Running     Running --&gt; Ready     Running -- "Blocking API function called" --&gt; Blocked </pre> |
| Idle Should Yield Konfiguration (I = Idle Task)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <p>- configIDLE_SHOULD_YIELD set to 0</p> <p>Tasks mit derselben Prio werden „time-sliced“</p>                                                                                                                                                                                                                                                                                                                                                                                                                                               | <p>- configIDLE_SHOULD_YIELD set to 1</p> <p>nach Beenden abgeben =&gt; Task haben verschiedene Laufzeiten!</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

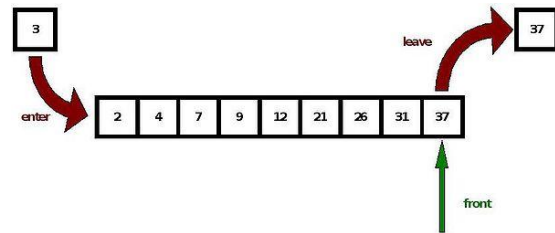


## Queue

Eine Queue ist eigentlich nichts anderes als eine Liste, welche meistens als FIFO konfiguriert wird.

### Eigenschaften

- Liste aus Elementen
- *Feste Elementgrösse/ItemSize (@ Erstellzeit)*
- Feste Queue Grösse (Queue Länge)
- Enqueue mittels kopieren (nicht Referenz!)
- Spezielle Routinen für ISR Benutzung



Enter = Enqueue

Leave = Dequeue

### FreeRTOS Queue-Funktionen (FreeRTOS Queue API)

#### Queue erstellen

```
xQueueHandle xQueueCreate(
 unsigned portBASE_TYPE uxQueueLength,
 unsigned portBASE_TYPE uxItemSize);
```

#### Queue löschen

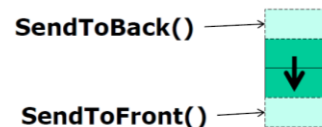
```
void vQueueDelete(xQueueHandle xQueue);
```



#### Daten Queue senden

```
portBASE_TYPE xQueueSendToBack(...);
portBASE_TYPE xQueueSendToFront(...);
```

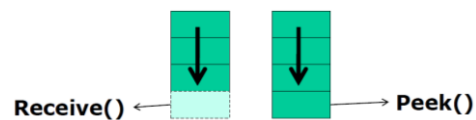
- Es kann ein Delay übergeben werden, falls Element noch kein Platz



#### Daten aus Queue auslesen

```
portBASE_TYPE xQueueReceive(...);
portBASE_TYPE xQueuePeek(...);
```

- Es kann ein Delay übergeben werden, falls noch kein Element vorhanden



### Eigene Queue implementieren

**WICHTIG:** C-File nicht *Queue.c* nennen, da das RTOS bereits ein C-File mit diesem Namen erzeugt!

#### Initialisierung

```
static xQueueHandle queueHandle;
queueHandle = FRTOS1_xQueueCreate(Queue_LENGTH, sizeof(char_t*)); //char pointer
if (queueHandle == NULL) {
 for (;;) {} /* out of memory? */
}
```

#### Daten in Queue speichern

```
void QUEUE_SendMessage(const char_t *msg) {
 char_t *ptr;
 size_t bufSize;

 bufSize = UTIL1_strlen(msg)+1;
 ptr = FRTOS1_pvPortMalloc(bufSize); // Speicher alloc.
 UTIL1_strcpy(ptr, bufSize, msg);
 if (FRTOS1_xQueueSendToBack(queueHandle, &ptr,
 portMAX_DELAY)!=pdPASS) {
 for(;;){} /* debug reason: queue access failed */
 }
}
```

#### Daten aus Queue auslesen

```
const char_t *QUEUE_ReceiveMessage(void) {
 const char_t *ptr;
 portBASE_TYPE res;

 res = FRTOS1_xQueueReceive(queueHandle,
 &ptr, 0);

 if (res==errQUEUE_EMPTY) {
 return NULL;
 } else {
 return ptr;
 }
}
```



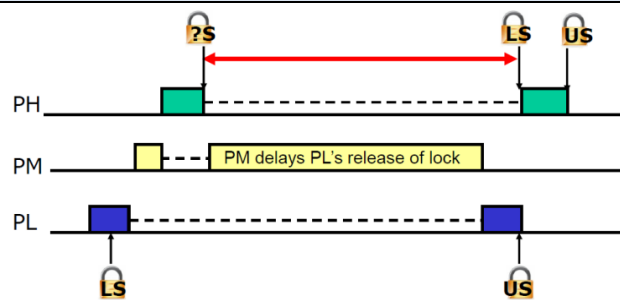
## Semaphoren

- Goal: - Ermöglichen einen sicheren Zugriff auf gemeinsame Ressourcen / Daten
  - Ermöglichen eine Synchronisation
- Description: Binär: ermöglicht einem Task den Zugriff (verfügbar / nicht verfügbar)
  - Zählend: eine definierte Anzahl Tasks kann zugreifen (~Queue)
  - Mutex: ähnlich wie binär, nur kann nur der Task der die Semaphore besitzt diese auch wieder freigeben.

?S = Anfrage für die Semaphore; LS = Lock Semaphore; US = Unlock Semaphore

### Problem: Priority Inversion

Task mit tiefer Prio sperrt eine Ressource.  
Ein Task mit hoher Prio unterbricht den  
Task mit tiefer Prio kann aber nicht laufen,  
da die gemeinsame Ressource belegt ist. Ein  
Task mit mittlerer Priorität, welcher nichts  
mit der gemeinsamen Ressource zu tun hat,  
unterbricht den Task mit der tiefen Prio und  
indirekt auch den mit der höheren Prio.



### Bedingung für ein Auftreten

- Fixed-priority preemptive scheduling
- Single Core Prozessor
- Binäre Semaphore

### Lösungsansätze:

- Priority Inheritance (Vererbung): der Task mit tiefer Prio erbt die Prio eines höheren Tasks falls dieser eine Anfrage startet. Nach dem Durchlauf erhält er wieder die alte Prio. Ein mittel priorisierter Task wird mit „push-through-Blocking“ unterbrochen. Bei mehreren Semaphoren können trotzdem noch Deadlocks entstehen!
- Priority Ceiling: Aufwändige Methode um Priority Inheritance so zu erweitern, dass keine Deadlocks mehr möglich sind. Ressourcen haben eine „Prioritätsschranke“ welche der Prio des höchsten Task entspricht der auf die Ressource zugreift.

- Rules:
1. Höchste Priorität gewinnt
  2. Ein Task wird blockiert, wenn seine Prio tiefer ist als die Schranke einer gerade gesperrten Ressource
  3. Ein Task erbt die Prioritätsschranke der Ressource (falls diese höher ist als die des Tasks), sobald ein anderer Task eine Anfrage auf die Ressource startet

### Typen von Blocking

- Direct: Falls bei einer Anfrage die Ressource bereits vergeben ist
- Push-through: Entsteht bei Priority Inheritance
- Transitive: Task 1 wird von Task 2 geblockt, wobei Task 2 von Task 3 geblockt wird

### -implementation

```
xSemaphoreHandle semaphore = NULL;
semaphore = FRTOS1_xSemaphoreCreateMutex(); // Semaphore erstellen
// ...CreateBinary(), ...CreateCounting(), ...CreateMutex()
if (semaphore != NULL) {
 if (FRTOS1_xSemaphoreTake(semaphore, portMAX_DELAY) == pdTRUE) {
 // locked Access: do something here...
 FRTOS1_xSemaphoreGive(semaphore); // free semaphore
 }
}
```

## FreeRTOS Trace

- Goal: Ermöglicht das Debuggen / Überwachen und Optimieren von FreeRTOS Applikationen
- functionality:
  - Laufzeit der Tasks
  - locken / unlocken von Semaphoren / Mutex
  - Queue In- und Outputs
- conditions:
  - der Performance Timer muss schneller sein als der Tick Timer. Dieser zählt so die Laufzeit der Tasks
- realisation: FreeRTOS platziert Makros an wichtigen Stellen (z.B. TaskCreate) und speichert Infos ins RAM (benötigt zusätzliches RAM!!). Diese Infos können später ausgelesen und übersichtlich dargestellt werden.
- percepio: Tool zum Darstellen von „Trace-Files“
- implementation:
 

```
// INIT TRACE @ RTOS.c
void RTOS_Run(void) {
 #if PL_HAS_RTOS_TRACE
 if (Ptrc1_uiTraceStart() == 0) {
 for (;;) { /* error starting trace */
 }
 }
 #endif
 ...
 // Log Queues @ MyQueue.c
 queueHandle = FRTOS1_xQueueCreate(Queue_LENGTH, Queue_ITEM_SIZE);
 #if PL_HAS_RTOS_TRACE
 Ptrc1_vTraceSetQueueName(queueHandle, "MsgQueue");
 #endif
}
```
- Get Data: Folgende Befehle in der CW Debugg-Shell ausführen:
 

```
> set start_addr [evaluate &RecorderData]
> set end_addr [evaluate ((char*)&RecorderData) + sizeof(RecorderData)-1]
> save -b $start_addr..$end_addr c:\\tmp\\trace.dump -o
```

## Motor Signals / Trace

- PWM :
  - low-active, output compare, 0...100% duty cycle (~Drehzahl)
  - die Auflösung muss kleiner als die kleinste Zeitkonstante des Prozesses sein und kleiner als die mechanische Motorenkonstante (0.63% Speed @Unominal)

- DIR: -GPIO welche die Drehrichtung angibt (zu definieren: 0= Vorwärts?)

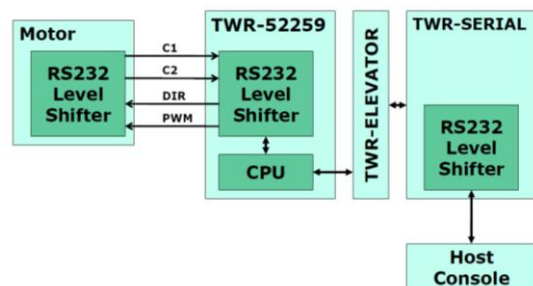
- C1 & C2: -Quadratur Encoder Signal

- Implementation:

```
#define MOT_SETDIRECTION(dir) \
 ((dir) == MOT_DIR_FORWARD ? DIR_ClrVal() : DIR_SetVal())
#define MOT_GETDIRECTION() \
 ((DIR_GetVal() == 0) ? MOT_DIR_FORWARD : MOT_DIR_BACKWARD)
```

Shell Parser:

```
} else if (UTIL1_strncmp(cmd, "motor duty", sizeof("motor duty")-1) == 0) {
 p = cmd + sizeof("motor duty");
 if (UTIL1_xatoi(&p, &val) == ERR_OK && val >= -100 && val <= 100) {
 PPG1_SetRatio16(percent * (0xFFFF / 100)); // Sets PWM duty
 // PPG ist die PE Komponente des PWM-Moduls- Vorzeichen nicht beachtet!
```



## Quadrature Encoder

-Goal : Motordrehzahl messen / überwachen können

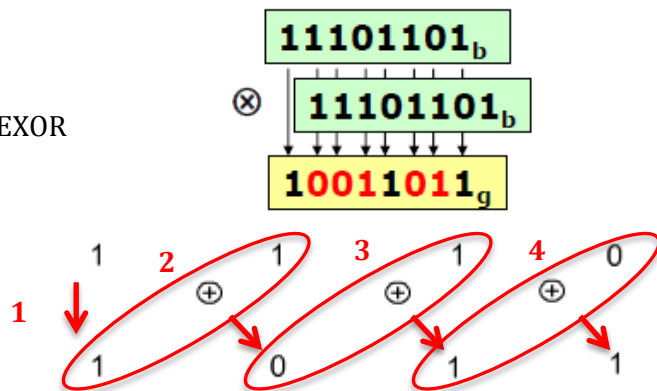
-Binary and Gray Code

Binary to Gray Code:

⇒ Um Eins verschieben und EXOR  
(Ungleich ergibt 1)

Gray Code to Binary:

⇒ Erste Zahl nach unten,  
ExOR mit der Zweiten



Gray Code hat einen Hammingabstand von eins, d.h. es ändert sich jeweils nur 1 Bit.

-Drehscheiben:

Absolut Digital: Binäres hochzählen, pro Bit wird ein Sensor benötigt, ergibt  $2^N$  Positionen

Absolut Gray: weniger Fehleranfällig, da nur jeweils eine Änderung

Simple Regular: Scheibe mit regelmässigen Abständen, Richtung kann nicht ermittelt werden, benötigt aber nur 1 Signal

**Quadratur:** ähnlich wie Simple mit zwei versetzten Signalen  
(verschiedene Positionen erzeugen beim Ersten eine schnellere Flanke)

**Signalrate** muss bekannt sein:

Maximale Drehzahl \* Anzahl Positionen auf der Scheibe

-Sampling:

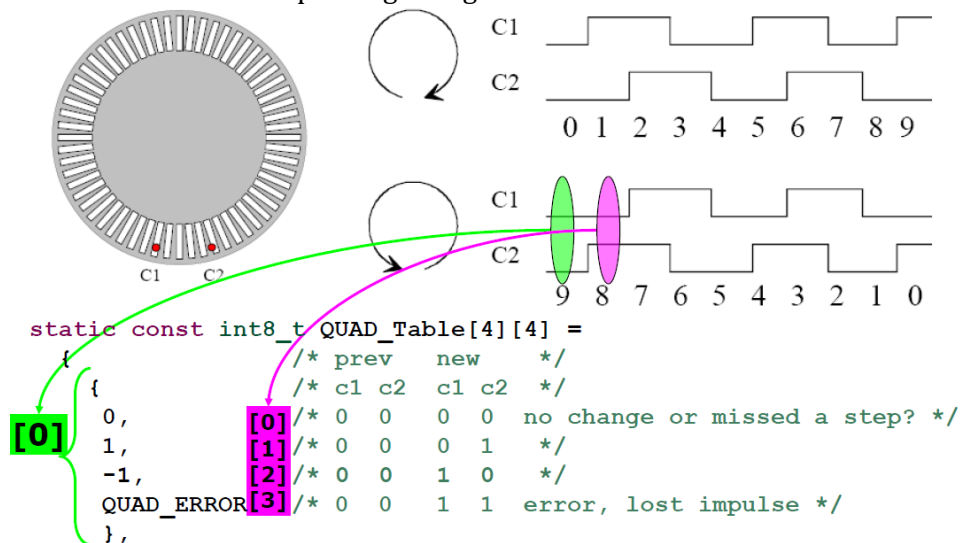
via Interrupts: abhängig von der Systemlast und die Latenz hat einen Einfluss.  
via Sampling: Abtasttheorem muss sicher eingehalten werden ( $f_s \geq 2 * f_{max}$ )

-Decoding:

Für das Abtasten der Signale wird ein Timer (inkl. Interrupt) im  $\mu s$ -Bereich benötigt. Dieser liest regelmässig die beiden BIT I/Os C1 und C2 ein.

**c12 = (C1\_GetVal() != 0 ? 2 : 0) | (C2\_GetVal() != 0 ? 1 : 0);**

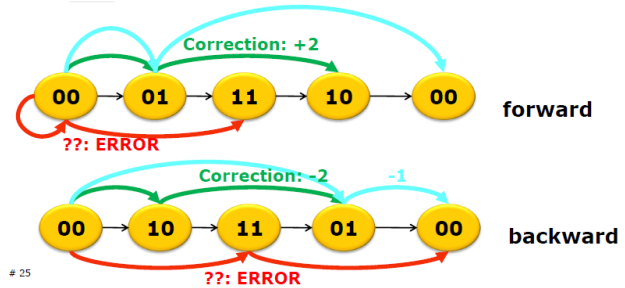
Die Zustandsüberprüfung erfolgt mittels Tabellen:



⇒ Diese Tabelle muss für alle 4 Zustände implementiert werden

-Tabelle mit Error Korrektur:

```
static const int8_t QUAD_Table[4][4][4] =
{
 /* pprev prev new */
 /* c12 c12 c12, */
 {
 0, /* 00 00 00 no change, miss
 1, /* 00 00 01 */
 -1, /* 00 00 10 */
 QUAD_ERROR, /* 00 00 11 error, lost imp
 },
```

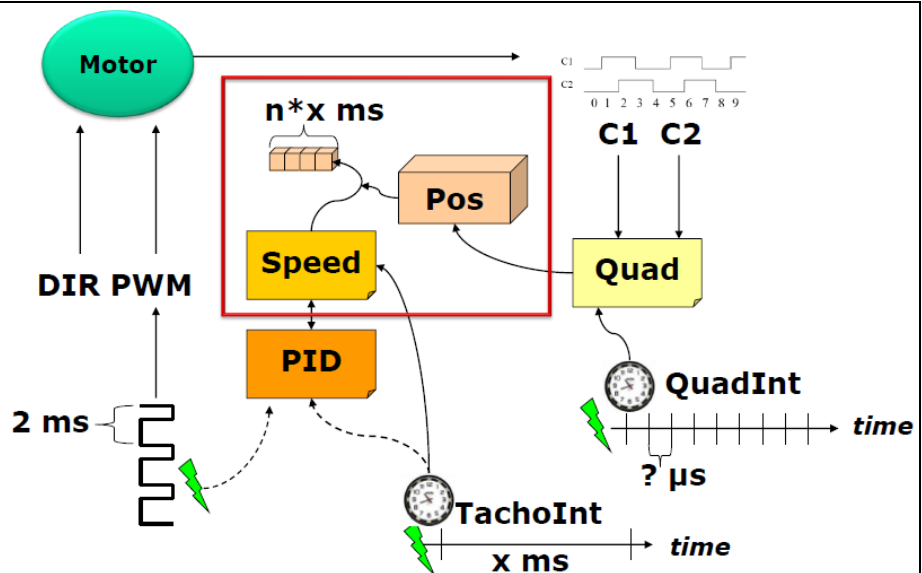


- ⇒ Tabelle muss für alle 16 Zustände implementiert werden.
- ⇒ Das Verlieren eines Impulses wird toleriert, solange die Richtung erkennbar ist

```
new_step = QUAD_Table[QUAD_prevlast_value][QUAD_last_value][c12];
QUAD_currPos += new_step; // Im Endeffekt wird ein Positionscouter hochgezählt
```

## Tacho

-System-Architektur:  
(Tacho : roter Bereich)



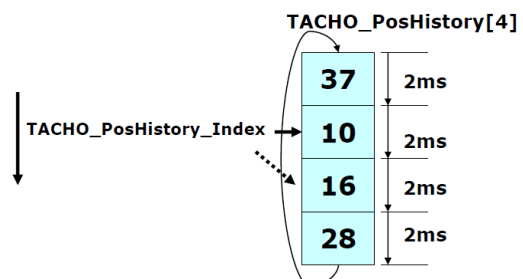
Der Tacho Timer ruft periodisch den Tacho Interrupt auf. Darin wird die aktuelle Position vom Quadratur Encoder ausgelesen und in ein Array gespeichert. Das Array beinhaltet n Werte über welche gemittelt werden kann. Das Array ist zudem als Ringbuffer implementiert.

Das Tacho Interrupt wird alle 2ms aufgerufen.

```
static volatile uint16_t TACHO_PosHistory[NOF_HISTORY];
static volatile uint8_t TACHO_PosIdx = 0;
```



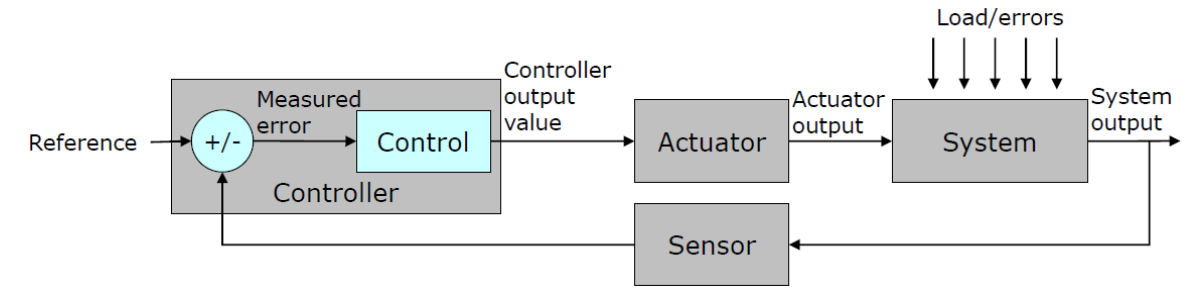
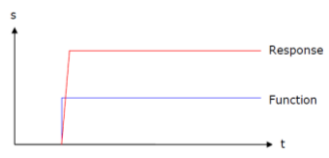
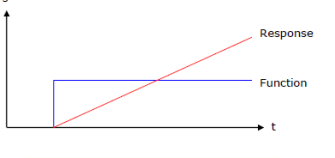
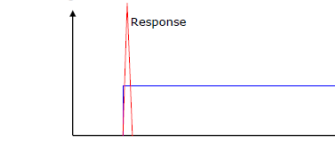
```
int32_t TACHO_CalcSpeed(void) {
 int32_t temp = 0;

 // (neuster Wert - ältester Wert) / Anzahl zu mittelnde Werte = Delta Speed
 temp = (int16_t) (TACHO_PosHistory [TACHO_PosIdx == 0 ? NOF_HISTORY - 1 : TACHO_PosIdx
 - 1]
 - TACHO_PosHistory[TACHO_PosIdx]);
 TACHO_currSpeed = (1000 * temp) / (SPEED_CALC_PERIOD_MS * (NOF_HISTORY - 1));
 return TACHO_currSpeed;
}
```



-Step Arithmetic: Es muss immer darauf geachtet werden, dass bei Subtraktionen um den Nullpunkt keine Auslöschung der Werte vorkommt (speed= 0?)

## Closed Loop Control (PID)

| Control vs. Closed Loop Control                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <u>Control (Steuerung)</u><br> <ul style="list-style-type: none"><li>Übertragung von Informationen</li><li>Bearbeitung von Informationen</li><li>Sensoren als Informationsproduzenten</li><li>Aktuatoren als Informationskonsumenten</li></ul> <u>Eigenschaften:</u> <ul style="list-style-type: none"><li>Open Loop (offener Regelkreis)</li><li>Kein Feedback</li></ul> | <u>Closed Loop Control (Regelung)</u><br> <ul style="list-style-type: none"><li>Geschlossener Regelkreis (closed loop)</li><li>Formelles Modell für<ul style="list-style-type: none"><li>Messen – Vergleichen – Steuern</li></ul></li><li>X = Sollwert (Input)</li><li>Z = Istwert / Messwert (Output)</li><li>e = Abweichung (X-Z), Regelfehler</li><li>Prozesskontroll Algorithmus ist im „Controller“ enthalten</li><li>Der Controller stellt einen neuen Stellwert „s“ für das System (Plant)</li></ul> |                                                                                                                                        |
| Systemarchitektur                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                        |
|  <p>Actuator output = PWM Signal<br/>System = Motor<br/>Sensor = Quadratur Encoder + Tacho</p> <p><u>Ziel:</u> Definiertes Verhalten des Systems, Berücksichtigung von Störeinflüssen</p>                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                        |
| Reglerauslegung                                                                                                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                        |
| Normalerweise wird der Regler anhand der Sprungantwort des System Outputs ausgelegt.                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                        |
| <u>Regler-Anteile</u><br>P = Proportional-Anteil → Einfache Verstärkung des Regelfehlers<br>I = Integral-Anteil → Summiert den Regelfehler auf, eliminiert stationäre Ungenauigkeit<br>D = Differential-Anteil → Steigung des Regelfehlers, schnelle Änderungen werden hoch verstärkt                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                        |
| <u>P-Anteil</u><br> $s(t) = K_p \cdot e(t)$                                                                                                                                                                                                                                                                                                                             | <u>I-Anteil</u><br> $s(t) = K_i \int_0^t e(t') \cdot dt'$                                                                                                                                                                                                                                                                                                                                                                                                                                                  | <u>D-Anteil</u><br> $s(t) = K_d \frac{d}{dt} e(t)$ |

### Reglerauslegung: PID-Regler

#### Beispiel-Code

```
esum = esum+e;
s = Kp*e + Ki*Ta*esum + Kd*(e-eprev)/Ta;
eprev = e;
```

#### Wofür welche Anteile?

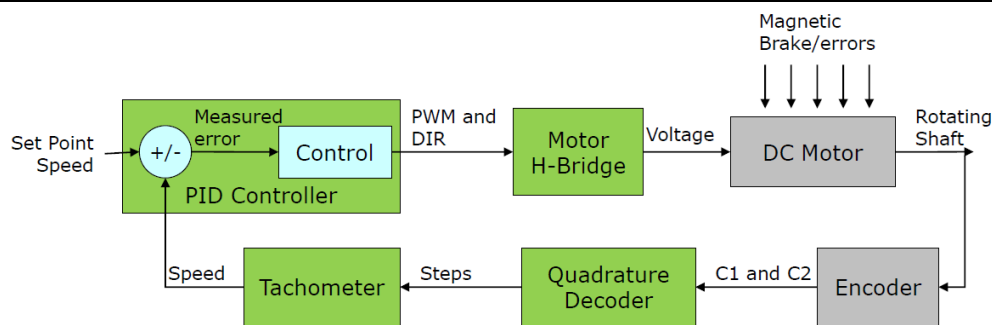
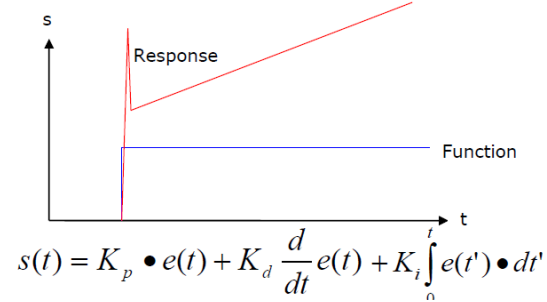
P+D Anteil: Geschwindigkeit des Reglers

I: Stationäre Genauigkeit

#### Sonstiges

- I-Anteil benötigt **Anti-Wind-Up**, damit der Wert begrenzt ist und ab einer gewissen Schwelle nicht mehr höher wird.
- **Totzeit** möglichst klein halten! (durch angemessenes Timing z.B. jede PWM-Periode einen neuen Stellwert generieren)

#### Sprungantwort



#### -implementation

```
void PID_Control(void) {
 /* Performs a PID calculation. The set point is specified by 'PID_SetPoint()' */
 static int32_t oldRegelDiff = 0; /*!< Remember error for next round (for D part) */
 int32_t regelDiff; /* actual error */
 int32_t setPoint; /* new set point */
 regelDiff = PID_setPoint - TACHO_CalcSpeed();
 if (PID_setPoint == 0) { MOT_SetVal(0); return; /* all done, this was easy :- */ }

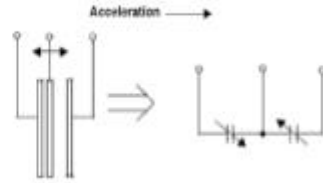
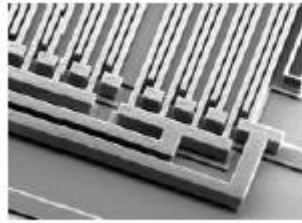
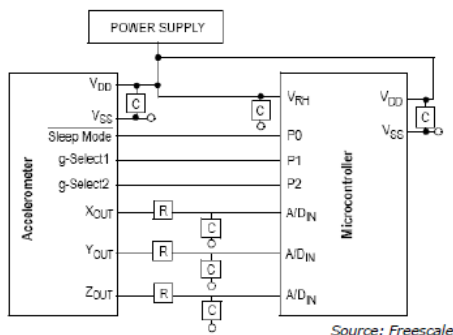
 setPoint = (PID_Kp100 * regelDiff / K_UNIT_VAL); /* P-Part */
 PID_RegIntegral += regelDiff * PID_SAMPLE_TIME_MS; /* I-Part */
 if (PID_RegIntegral > ANTI_WINDUP) { PID_RegIntegral = ANTI_WINDUP; }
 if (PID_RegIntegral < (-ANTI_WINDUP)) { PID_RegIntegral = -ANTI_WINDUP; }
 setPoint += (PID_Ki100 * PID_RegIntegral / K_UNIT_VAL);
 setPoint += (PID_Kd100 * (regelDiff - oldRegelDiff) / (K_UNIT_VAL*PID_SAMPLE_TIME_MS)); /* D-Part */
 oldRegelDiff = regelDiff;

 if(setPoint < 0){
 setPoint = -setPoint;
 MOT_SetDirection(MOT_DIR_BACKWARD);
 } else{ MOT_SetDirection(MOT_DIR_FORWARD); }

 if(setPoint > 65535){ setPoint = 65535; }
 MOT_SetVal((word) setPoint);
}
```

## Accelerometer

-Sensor: Accelerometer sind meistens MEMS (micro-electro-mechanical-systems). Durch eine auf den Chip wirkende Beschleunigung verändert sich die Kapazität welche gemessen und ausgewertet werden kann.



g-Select: Empfindlichkeit einstellbar  
R-C: Ausgangssignale sind gefiltert  
Die Ausgangsspannungen werden mit dem ADC gemessen (da Analog output).

-Calibration: Es wird für jeden Kanal (X, Y, Z) eine Kalibration benötigt:

- Gain Fehler
- Offset Fehler
- Wegen Temperatur- und Herstellungseinflüssen

Die Kalibrationsdaten (Offset + Wert bei 1g) können im Processor Expert eingegeben und so im Flash abgespeichert werden.



-implementation:

```
int16_t ACCEL1_GetXmg(void)
{
 int32_t L;

 L = ACCEL1_MeasureGetRawX(); /* reads Data from ADC */
 L -= CalNxOff; /* apply offset: defined from PE */
 L -= zeroGValue; /* get based to zero g: 0xFFFF / 2 */
 L *= 1000; /* scale to milli g */
 L /= (CalNx1g-CalNxOff); /* norm with 1g value: defined from PE */
 return (int16_t)L;
}

static void ACCEL_PrintStatus(const FSSH1_StdIOType *io) {
 FSSH1_SendStatusStr("Accel", "\r\n", io->stdOut);
 FSSH1_SendStatusStr(" milli-g", "", io->stdOut);
 FSSH1_SendNum16s(ACCEL1_GetXmg(), io->stdOut);
 FSSH1_SendStr(" ", io->stdOut);
 FSSH1_SendNum16s(ACCEL1_GetYmg(), io->stdOut);
 FSSH1_SendStr(" ", io->stdOut);
 FSSH1_SendNum16s(ACCEL1_GetZmg(), io->stdOut);
 FSSH1_SendStr("\r\n", io->stdOut);
}
```



## LCD

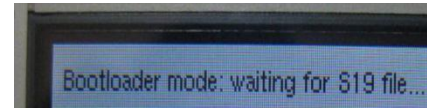
Resistive Touchscreens reagieren auf Druck, welcher zwei leitfähige Schichten teilweise miteinander verbindet. Diese Verbindung bildet einen Spannungsteiler welcher erlaubt die Position des Druckes zu ermitteln.

## -Bootloader

Mit einem Bootloader kann die Firmware eines Devices ohne Debug-Kabel geladen werden.

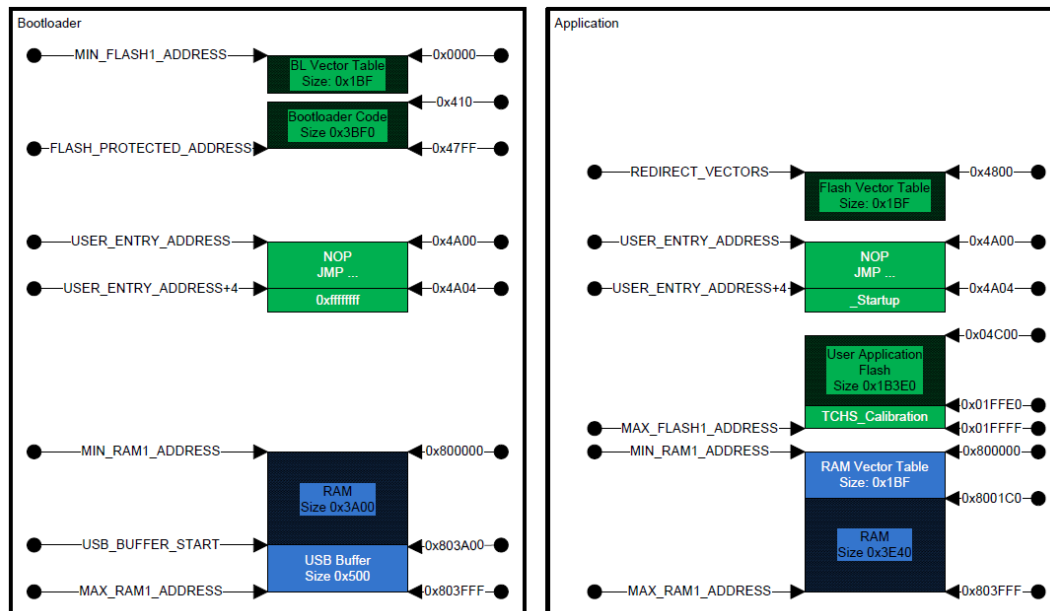
Folgendes wird für den Bootloader benötigt:

- S19-File (S-RECORD: ASCII basierende Textcodierung)
- Device muss sich im „Bootloader mode“ befinden



Das Device wird z.B. mittels USB am PC angeschlossen. Dabei erscheint das Device als Wechselmedium (FAT16) im Arbeitsplatz. Nun kann das S19-File mittels Drag&Drop oder kopieren auf das Wechselmedium verschoben werden. Der Bootloader lädt das File, parst es und flasht die darin enthaltene Applikation auf den Device.

Der Applikationsflash muss überhalb der Adresse „**FLASH\_PROTECTED\_ADDRESS**“ sein um den Bootloader nicht zu überschreiben.

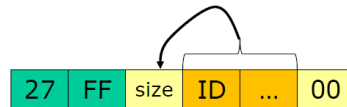


## -I2C

**Message via I2C an LCD-Board senden um Werte wie die Motorengeschwindigkeit anzuzeigen.**

```
void I2C_SendTachoSpeed(void) {
 int32_t speed;
 char buf[9];
 uint16_t snt;

 speed = TACHO_GetSpeed();
 buf[0] = I2C_PRE_BYTE0;
 buf[1] = I2C_PRE_BYTE1;
 buf[2] = 5; /* data length */
 buf[3] = (uint8_t) I2C_MSG_MOTOR_SPEED;
 buf[4] = (uint8_t) (speed>>24);
 buf[5] = (uint8_t) ((speed&0xff0000)>>16);
 buf[6] = (uint8_t) ((speed&0xff00)>>8);
 buf[7] = (uint8_t) (speed&0xff);
 buf[8] = 0; /* termination byte */
 I2C2_SelectSlave(I2C_LCD_ADDR);
 I2C2_SendBlock(&buf[0], sizeof(buf), &snt);
}
```



- Auf der Seite des Towers:
1. Aufbau der I2C Message
  2. Slave Select
  3. Senden der Message



### Auf der Seite des LCD-Boards (JM128) wird die Message empfangen (PSEUDOCODE)

```
Interrupt: I2C_OnReceiveData(
 res= I2C1_RecvBlock(&data, 1, &nofBytesReceived);
 // reads a block from input buffer
 buf[index] = data;
 index++;
 // Interrupt trifft so oft auf, bis der Input-Buffer leer ist und alle Daten empfangen sind
 // Trifft dies zu wird die Message interpretiert
...
} else if (buf[3]==I2C_MSG_MOTOR_SPEED &&
 buf[2]==I2C_MOTOR_MSG_LEN&&index>I2C_MOTOR_MSG_MAX_IDX) {
 motorSpeed = (int32_t)((buf[4]<<24)|(buf[5]<<16)|(buf[6]<<8)|buf[7]);
 index = 0; /* message completed */
 // mittels getter Methode wird der Motorspeed von der LCD UI gelesen
```

#### -I2C Dataflow

- Der Tower sendet Daten an den LCD (siehe oben) und löst dort ein Interrupt aus und werden vom Parser interpretiert
  - Neue Daten: der Tower updatet die Werte des LCD (bspw. aktueller Motorspeed)
  - Datenanfrage: der Tower verlangt die Werte der Slider auf dem LCD
- Die Shell liest auch von der I2C Schnittstelle und ermöglicht die Verwendung derselben Parser-Methoden.
- I2C ist unidirektional

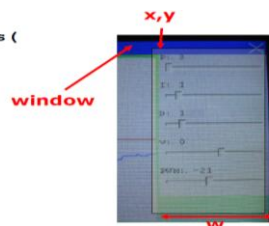
### -C++ und Slider-Widget

#### Slider Creation

```
static SliderWidget sliderP, sliderI, sliderD, sliderV, sliderPWM;

void SLIDER_CreateSliders(UI1_Window *window, UI1_PixelDim x, UI1_PixelDim y, UI1_PixelDim w, UI1_PixelDim h) {
 sliderPWM.Init(window,x,y,w,h/5,(SliderValT)PWM_SLIDER_MIN,(SliderValT)PWM_SLIDER_MAX,(SliderStepT)PWM_SLIDER_STEP, (const unsigned char*) "PWM",0);
 sliderP.Init(window,x,y+h/5,w,h/5,(SliderValT)PID_MIN,(SliderValT)PID_MAX,(SliderStepT)PID_STEP, (const unsigned char*) "P Param",0);
 sliderI.Init(window,x,y+2*h/5,w,h/5,(SliderValT)PID_MIN,(SliderValT)PID_MAX,(SliderStepT)PID_STEP, (const unsigned char*) "I Param",0);
 sliderD.Init(window,x,y+3*h/5,w,h/5,(SliderValT)PID_MIN,(SliderValT)PID_MAX,(SliderStepT)PID_STEP, (const unsigned char*) "D Param",0);
 sliderV.Init(window,x,y+4*h/5,w,h/5,(SliderValT)PID_MIN,(SliderValT)PID_MAX,(SliderStepT)PID_STEP, (const unsigned char*) "Velocity",0);
}

void SLIDER_CreateSliders(
 UI1_Window *window,
 UI1_PixelDim x,
 UI1_PixelDim y,
 UI1_PixelDim w,
 UI1_PixelDim h)
{
 ...
}
```

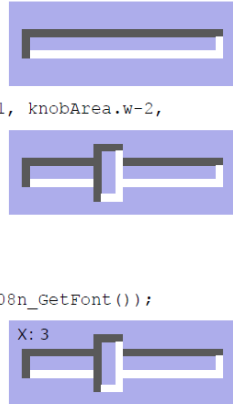
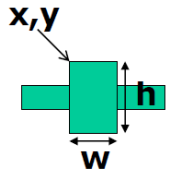


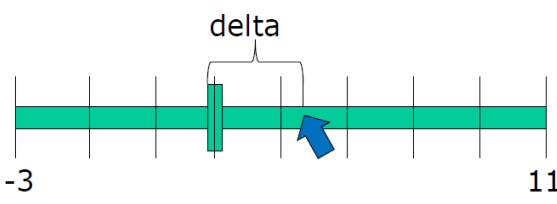
#### Window Callback

Hier wird nur das Handling von „sliderP“ dargestellt. Je nach EventCallbackKind wird ein Event ausgelöst.

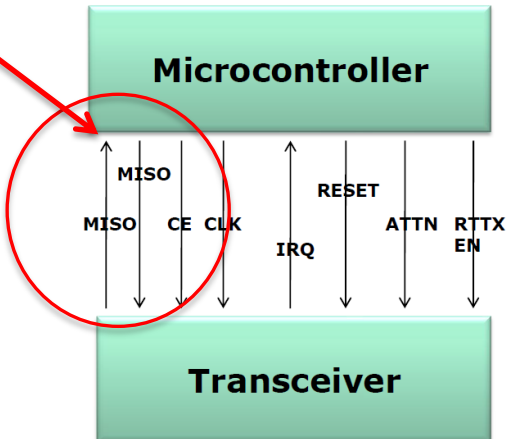
Bspw. wird beim Klicken auf den Slider ein onClickMove()-Event ausgelöst. Dieser setzt den neuen wert im Slider und zeichnet ihn mit „paint()“ neu.

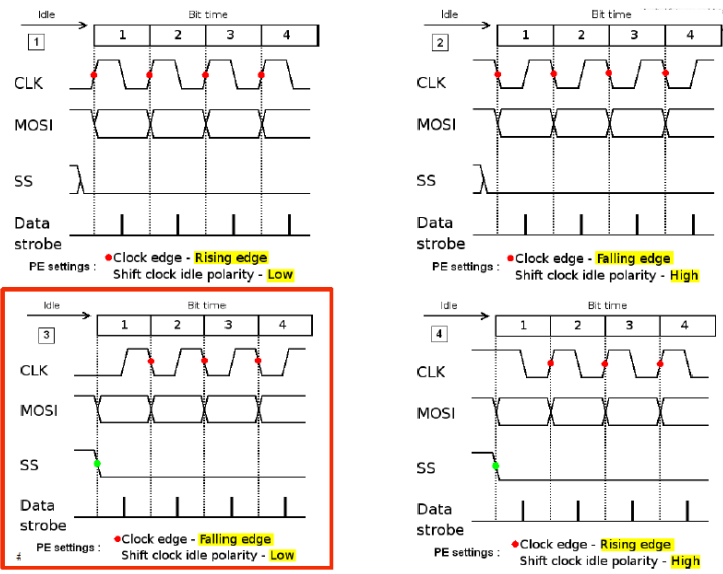
```
void SLIDER_SliderW_WindowCallback(
 UI1_Window *window, UI1_Element *element,
 UI1_EventCallbackKind kind,
 UI1_Pvoid data)
{
 if (kind==UI1_EVENT_CLICK) {
 sliderP.OnClick((UI1_Coordinate*) data);
 /* other objects? */
 } else if (kind==UI1_EVENT_CLICK_MOVE) {
 sliderP.OnClickMove((UI1_Coordinate*) data);
 } else if (kind==UI1_EVENT_PAINT) {
 sliderP.Paint();
 }
}
```

| Command Getter                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Liest den Wert des Sliders aus und schreibt ihn in Form eines Strings in den übergebenen Buffer.</p> <p>Wird bspw. von I2C benötigt.</p> | <pre> void SLIDER_GetCmdString(unsigned char *buf, size_t bufSize) {     SliderValT val;     static SliderValT oldPWM=-1, oldKp=-1, oldKi=-1, oldKd=-1, oldV=-1;      for(;;) { // breaks         val = sliderPWM.GetValue();         if (val!=oldPWM) {             UTIL1_strcpy(buf, bufSize, (const unsigned char*)"motor duty ");             UTIL1_strcatNum32s(buf, bufSize, val);             oldPWM = val;             break;         }     }     /* \todo Add other command providers as needed */     buf[0] = '\0'; // empty response     break; } </pre>                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Paint()                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <p>Stellt die Slider mit dem Balken, den Knöpfen (Knobs) sowie Stellung des Knobs auf dem LCD dar.</p>                                      | <pre> void SliderWidget::Paint(void) {     // background     UI1_DrawFilledBox(m_window, m_area.x, m_area.y, m_area.w, m_area.h,         UI1_COLOR_BRIGHT_BLUE);     // slider horizontal line     UI1_DrawHLine(m_window, m_area.x+SLIDER_H_BORDER, m_area.y+(m_area.h/2)-2,         m_area.w-(SLIDER_H_BORDER*2), UI1_COLOR_BRIGHT_GREY);     ...      // draw slider knob     knobArea = GetKnobPosition();     UI1_DrawFilledBox(m_window, knobArea.x+1, knobArea.y+1, knobArea.w-2,         knobArea.h-2, UI1_COLOR_BRIGHT_BLUE);     ...      // write label and value     buf[0] = '\0';     UTIL1_strcpy(buf, sizeof(buf), m_label);     ...      x = (UI1_PixelDim)(m_window-&gt;prop.x + m_area.x);     y = (UI1_PixelDim)(m_window-&gt;prop.y + m_area.y);     FDispl_WriteString(buf, UI1_COLOR_BLACK, &amp;x, &amp;y, Cour08n_GetFont()); } </pre> <p># 14</p>                                 |
| GetKnobPosition()                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| <p>Liest die aktuelle Position des Knopfes aus und gibt sie zurück.</p>                                                                     | <pre> WidgetArea SliderWidget::GetKnobPosition(void) {     WidgetArea area;     int i;      area.w = KNOB_WIDTH;     area.h = m_area.h-(KNOB_H_BORDER*2);     if (area.h&gt;KNOB_HEIGHT) {         area.h = KNOB_HEIGHT; // max height     }     i = (int)((GetVal()-GetMin())/((GetMax()-GetMin())/m_steps)); // i is in     range 0..m_steps     area.x = SLIDER_H_BORDER+(UI1_PixelDim)(m_area.x+(m_area.w*i/m_steps));     area.x -= area.w/2; // center knob at pixel position     if (area.x&lt;m_area.x+SLIDER_H_BORDER) { // completely to the left: start at     left side         area.x = m_area.x+SLIDER_H_BORDER;     } else if (area.x&gt;m_area.x+m_area.w-SLIDER_H_BORDER-area.w) { // completely     to the right: adjust position         area.x = m_area.x+m_area.w-SLIDER_H_BORDER-area.w;     }     area.y = m_area.y + (m_area.h/2) - area.h/2; // center     return area; } </pre>  |

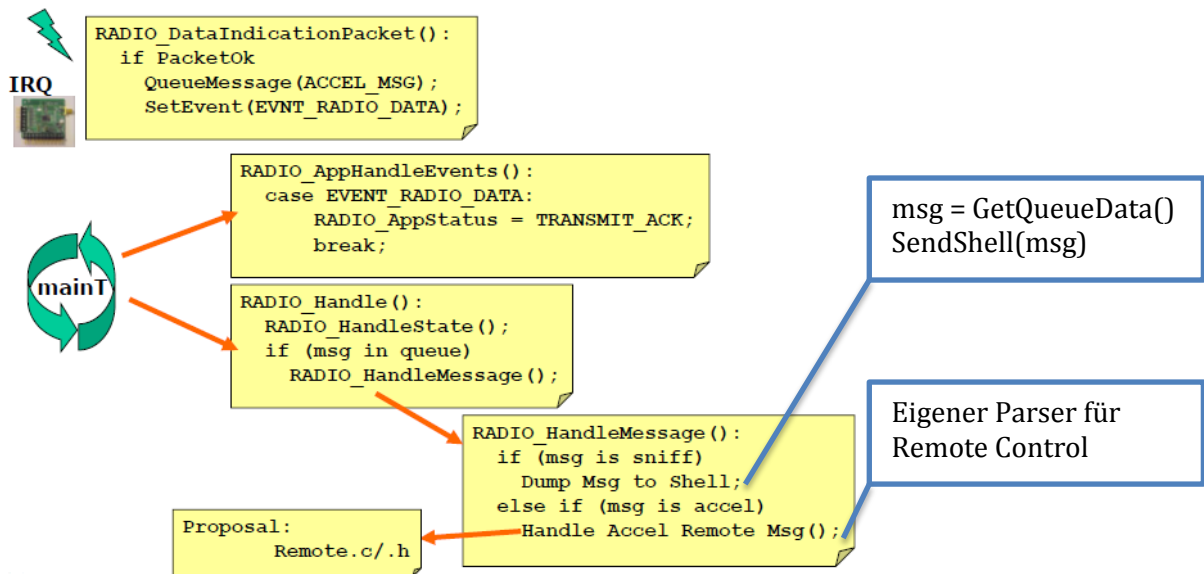
| SetKnobPosition()                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                              |
|------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Setzt die Position des Knopfes. Wird bei der Initialisierung sowie beim Klicken auf das LCD benutzt. | <pre> void SliderWidget::SetKnobPosition(WidgetArea *clickPos) {     SliderValT delta;     WidgetArea currKnob;      currKnob = GetKnobPosition();     delta = clickPos-&gt;x - currKnob.x; // get difference in pixels     delta *= (GetMax()-GetMin())/m_area.w; // scale to value     SetVal(GetVal()+delta); } </pre>  |

## Radio Transceiver

| SPI Interface                                                                                                                                                                                                                                                                                                                                                |                                                                                     |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| <p><b>SPI ist schneller aber auch komplizierter als I2C</b></p> <p>MISO = Master In – Slave out<br/>MOSI = Master Out – Slave In<br/>CE = Chip Enable<br/>CLK = Clock</p> <p><b>Steuersignale</b><br/>IRQ = Interrupt Request (Wakeup uC, <b>benötigt Pullup!</b>)<br/>RESET = Reset Transceiver<br/>ATTN = Wakeup transceiver<br/>RTTX = Antenna switch</p> |  |

| SPI Protocol                                                                                                                                                                                                                                                                                                                        |                                                                                      |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| <p>Die SPI Schnittstelle kann auf <b>grundsätzlich vier verschiedene Arten konfiguriert</b> werden. Die Abbildung rechts zeigt was die Einstellungen der „Clock edge“ und „Shift clock idle polarity“ bewirken. Im INTRO wird die <b>rot umrahmte Version</b> benutzt und ist im Allgemeinen die am <b>häufigsten genutzte</b>.</p> |  |

### Radio: Packet handling (Receive Packet)



### RADIO\_DataIndicationPacket()

```

void RADIO_DataIndicationPacket(tRxPacket *sRxPacket) {
 if (sRxPacket->u8Status==SMAC1_TIMEOUT) {
 EVNT_SetEvent(EVNT_RADIO_TIMEOUT);
 } else if (sRxPacket->u8Status == SMAC1_SUCCESS) {
 if (RADIO_isSniffing) {
 QueueMessage(RADIO_QUEUE_MSG_SNIFF,
 sRxPacket->pu8Data, sRxPacket->u8DataLength);
 }
 if (RADIO_AppStatus==RADIO_WAITING_FOR_ACK && isACK()) {
 EVNT_SetEvent(EVNT_RADIO_ACK);
 } else if (isMessageForMe()) {
 /* queue it? */
 EVNT_SetEvent(EVNT_RADIO_DATA);
 } else { /* unknown packet? */
 EVNT_SetEvent(EVNT_RADIO_UNKNOWN);
 }
 } else if (sRxPacket->u8Status==SMAC1_OVERFLOW) {
 EVNT_SetEvent(EVNT_RADIO_OVERFLOW);
 }
}

```

### QueueMessage()

```

static void QueueMessage(uint8_t kind, const char *msg, uint8_t msgSize) {
 /* format is: kind(8bit) dataSize(8bit) data */
 uint8_t i, buf[RADIO_QUEUE_ITEM_SIZE];
 signed portBASE_TYPE pxHigherPriorityTaskWoken;

 buf[0] = kind;
 buf[1] = msgSize;

 i = 2;
 while(msgSize>0 && i<sizeof(buf)) {
 buf[i++] = *msg;
 msg++;
 msgSize--;
 }
 if (FRTOS1_xQueueSendToBackFromISR(RADIO_MsgQueue, &buf[0], &pxHigherPriorityTaskWoken)!=pdTRUE) {
 /* was not able to send to the queue. Well, not much we can do here... */
 }
}

```

Läuft in einer ISR!

| <b>RADIO_Handle()</b>                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>• Ruft die State Machine auf: <b>HandleState()</b>,</li> <li>• Liest von der Queue: <b>xQueueReceive()</b></li> <li>• Stellt die Daten richtig zusammen und sendet diese an die Shell: <b>HandleMessage()</b></li> </ul> | <pre> void RADIO_Handle(void) {     uint8_t buf[RADIO_QUEUE_ITEM_SIZE];      if (RADIO_isOn) {         RADIO_HandleState();     }     /* poll radio message queue */     if (FRTOS1_xQueueReceive(         RADIO_MsgQueue, buf, 0) == pdPASS)     {         /* received message from queue */         RADIO_HandleMessage(buf);     } } </pre> |
| <b>RADIO_HandleState() State Machine</b>                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                |
| <p>Rechts ist die State Machine des Radio Transceivers abgebildet.</p>                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                |

## Remote Motor Controller

-Goal: Der eine sagt "hello" und der andere bestätigt den korrekten Empfang mit "ack".

| Mögliche Probleme, Störung der Übertragung                                                                                                                                                                                                                                                                                                     |         |                                                                                                                                                                      |     |         |     |       |       |       |         |  |  |       |      |      |     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|---------|-----|-------|-------|-------|---------|--|--|-------|------|------|-----|
| <p>Wie merke ich, dass die Daten meine eigenen sind?</p> <ul style="list-style-type: none"><li>• Gleicher Kanal? → Anderen Kanal wählen</li><li>• Oder eine Art Protokoll einführen. Mit einer individuellen PANID (Previous Access Network Identifier) kann. Zusätzlich kann das Protokoll z.B. mit einer CRC-Summe ergänzt werden.</li></ul> |         | <table><tr><td colspan="4">message</td></tr><tr><td>PANID</td><td colspan="3">payload</td></tr><tr><td>PANID</td><td>kind</td><td>data</td><td>CRC</td></tr></table> |     | message |     |       |       | PANID | payload |  |  | PANID | kind | data | CRC |
| message                                                                                                                                                                                                                                                                                                                                        |         |                                                                                                                                                                      |     |         |     |       |       |       |         |  |  |       |      |      |     |
| PANID                                                                                                                                                                                                                                                                                                                                          | payload |                                                                                                                                                                      |     |         |     |       |       |       |         |  |  |       |      |      |     |
| PANID                                                                                                                                                                                                                                                                                                                                          | kind    | data                                                                                                                                                                 | CRC |         |     |       |       |       |         |  |  |       |      |      |     |
| <table><tr><td>"ABC"</td><td>'x'</td><td>value</td></tr><tr><td>PANID</td><td>kind</td><td>payload</td></tr></table> <p>SendString("x37");</p> <p>↓</p> <p>"ABCx37"</p> <p>↓</p> <p>DataIndicationPacket()</p> <p>↓</p> <p>rxPacket == "ABCx...?"</p> <p>↓ Yes!</p> <p>Extract payload</p>                                                     |         |                                                                                                                                                                      |     | "ABC"   | 'x' | value | PANID | kind  | payload |  |  |       |      |      |     |
| "ABC"                                                                                                                                                                                                                                                                                                                                          | 'x'     | value                                                                                                                                                                |     |         |     |       |       |       |         |  |  |       |      |      |     |
| PANID                                                                                                                                                                                                                                                                                                                                          | kind    | payload                                                                                                                                                              |     |         |     |       |       |       |         |  |  |       |      |      |     |

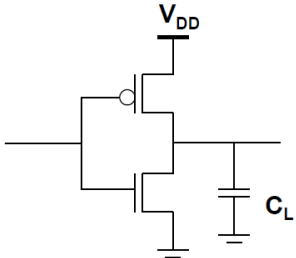
## Low Power

Leistung (Power) =  $U \cdot I$

Energie = Integral der Leistung über die Zeit

Zeitfaktor:

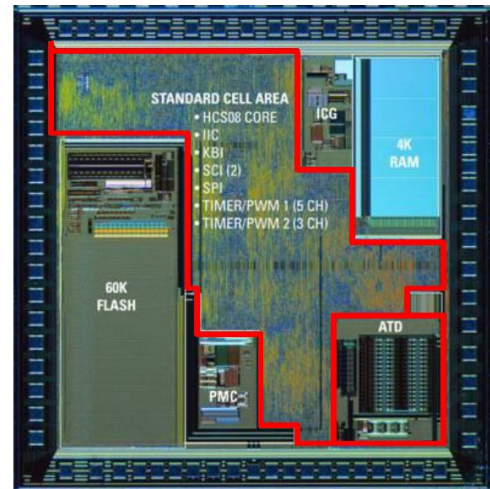
- Schneller erledigen: weniger Energie wird benötigt
- Aber es benötigt mehr Energie um etwas schneller zu erledigen

| <i>Leistung eines Transistors</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|--------------|----------|-----------------|-------|---------------------|----------|----------------|-----|-----------------|----------|----------------------|-------------------|------------------|-------------------|-----------------|
|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | <div style="background-color: yellow; padding: 10px; margin-bottom: 10px; border: 1px solid black;"> <math display="block">P_{\text{total}} = \alpha C_L V_{DD}^2 f + t_{sc} V_{DD} I_{\text{peak}} + V_{DD} I_{\text{leak}}</math> </div> <div style="background-color: #d3d3d3; padding: 10px; border: 1px solid black;"> <table style="width: 100%; border-collapse: collapse;"> <tr><td style="width: 20px;"><b>P</b></td><td><b>Power</b></td></tr> <tr><td><math>\alpha</math></td><td>Activity factor</td></tr> <tr><td><math>C_L</math></td><td>Transistor capacity</td></tr> <tr><td><math>V_{DD}</math></td><td>Supply voltage</td></tr> <tr><td><math>f</math></td><td>Clock frequency</td></tr> <tr><td><math>t_{sc}</math></td><td>Shortcut time factor</td></tr> <tr><td><math>I_{\text{peak}}</math></td><td>Shortcut current</td></tr> <tr><td><math>I_{\text{leak}}</math></td><td>Leakage current</td></tr> </table> </div> | <b>P</b> | <b>Power</b> | $\alpha$ | Activity factor | $C_L$ | Transistor capacity | $V_{DD}$ | Supply voltage | $f$ | Clock frequency | $t_{sc}$ | Shortcut time factor | $I_{\text{peak}}$ | Shortcut current | $I_{\text{leak}}$ | Leakage current |
| <b>P</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | <b>Power</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
| $\alpha$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Activity factor                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
| $C_L$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | Transistor capacity                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
| $V_{DD}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Supply voltage                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
| $f$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Clock frequency                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
| $t_{sc}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Shortcut time factor                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
| $I_{\text{peak}}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Shortcut current                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
| $I_{\text{leak}}$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Leakage current                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
| <i>Wie kann die benötigte Leistung gesenkt werden?</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |
| <p><b>External Power</b></p> <ul style="list-style-type: none"> <li>• Low Power Peripherie verwenden <ul style="list-style-type: none"> <li>○ z.B. 3.3V anstatt 5V</li> <li>○ Geräte/Bausteine mit internen Low Power Optimierungen/Shutdown</li> <li>○ High-End Power Supplies (weniger Verluste in der Speisung)</li> <li>○ Geräte, welche keine Kühlung benötigen</li> </ul> </li> <li>• Displays <ul style="list-style-type: none"> <li>○ Low Power LCD, LED, etc.</li> <li>○ Displaygrösse verringern</li> <li>○ Aktivität verringern</li> <li>○ Helligkeit und Farbe an Umgebungslicht anpassen</li> <li>○ Zero-Power (b-stabile) Displays</li> </ul> </li> <li>• Schutz vor Umgebungswärme, da der Leakage Current bei Transistoren bei steigender Temperatur auch steigt</li> </ul> <p>Das Ganze ist natürlich auch eine Preisfrage, da qualitativ höherwertige Bausteine meistens auch teurer sind.</p> <p><b>Hardware Pins</b></p> <ul style="list-style-type: none"> <li>• Debug Interface <ul style="list-style-type: none"> <li>○ Deaktivieren falls nicht benutzt</li> </ul> </li> <li>• Unbenutzte Pins <ul style="list-style-type: none"> <li>○ Als Output: Low</li> <li>○ Als Input: Pull-Ups (internal)</li> <li>○ Internal Unwired Pins (SiP)</li> </ul> </li> </ul> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |          |              |          |                 |       |                     |          |                |     |                 |          |                      |                   |                  |                   |                 |

## Einflussfaktoren auf den Stromverbrauch bei einem Mikrocontroller

### Die Size: Memory

- RAM (& Cache)
  - Grosse Fläche
  - Benötigt viel Leistung
- FLASH/EEPROM
  - Geringere Fläche
  - Benötigt weniger Leistung
  - Spezielle Programmierspannung?
- So viel wie benötigt
- ABER: Möglicherweise wird mehr benötigt als man denkt:
  - Stack!
  - Neue Funktionen
  - Produkt Lebenszyklus
  - Entwicklungsunterstützung (trace)



### Die Size: Core & Peripherie

- CPU Core
  - Register
  - Befehlssätze (Instruction set)
  - Pipelines
- Peripherie
  - Timer, PWM, ...
  - I2C, USB, SPI, SCI, ...
  - So wenig wie möglich
- ATD (Analog-to-Digital Konverter)
  - Relativ gross
  - Auf das Minimum reduzieren

### Low Power Modes

- Um so weniger Funktionen benötigt werden, desto tiefer kann der Low Power Mode sein

#### Wait Mode

- CPU gestoppt (Clock gating)
- Bus Clock bleibt aktiv aber wird gesenkt (ca. auf 1MHz)
- Interrupt: Wait mode wird beendet
- Vorteile
  - Einfach zu integrieren
  - $I_{dd}$  Reduktion im Vergleich zum Run-Mode
  - Keine Stop Erholungszeit (Interrupts bleiben aktiv)
  - Noise Reduktion für A/D-Wandlung
- Nachteile
  - Spannungswandler bleiben aktiv
  - Benötigt mehr Energie als in den Stop Modes

Increased Functionality  
More power needed



Stop1  
Stop2  
Stop3  
Wait  
Run

Lowest Power Consumption

| Mode  | PDC | PPDC          | CPU, Digital<br>Peripherals,<br>Flash | RAM     | ICG              | ATD                   | Regulator | I/O Pins    | RTI           |
|-------|-----|---------------|---------------------------------------|---------|------------------|-----------------------|-----------|-------------|---------------|
| Stop1 | 1   | 0             | Off                                   | Off     | Off              | Disabled <sup>1</sup> | Off       | Reset       | Off           |
| Stop2 | 1   | 1             | Off                                   | Standby | Off              | Disabled              | Standby   | States held | Optionally on |
| Stop3 | 0   | Don't<br>care | Standby                               | Standby | Off <sup>2</sup> | Disabled              | Standby   | States held | Optionally on |

<sup>1</sup> Either ATD stop mode or power-down mode depending on the state of ATDPU.

<sup>2</sup> Crystal oscillator can be configured to run in stop3. Please see the ICG registers.



### FreeRTOS and Low Power

Die `vApplicationIdleHook()` Funktion kann die CPU in Low Power Modes setzen.

```
70 #define configUSE_PREEMPTION 1
71 #define configUSE_IDLE_HOOK 1
72 #define configUSE_TICK_HOOK 1
73 #define configCPU_CLOCK_HZ 1000000
```

### Processor Expert

Bei den CPU-Komponenten sind Funktionen verfügbar, um die Low Power Modes zu setzen oder rückzusetzen.

Cpu:MC13213

- ☒ SetHighSpeed
- ☒ SetLowSpeed
- ☒ SetSlowSpeed
- ☒ GetSpeedMode
- ☒ SetIntVect
- ☒ GetIntVect

Cpu:MCF52259CAG80

- ☒ SetHighSpeed
- ☒ SetLowSpeed
- ☒ SetSlowSpeed
- ☒ GetSpeedMode
- ☒ GetResetSource
- ☒ GetBusFreqHz

Eventuell müssen auch Timer- und Schnittstellen-Frequenzen angepasst werden (z.B. UART/SCI auf 4800 baud o.ä.).

## Hardware

### Wieso werden Pull-Up und Pull-Down Widerstände eingesetzt?

- Mikroprozessoreingänge auf ein definiertes Signal ziehen (vor allem wichtig bei Einschalten/Reset des Mikrocontrollers)
- Werden zum Teil zur Konfiguration von Devices verwendet (Logisch High / Low)
- Open-Drain/Open-Kollektor-Eingänge benötigen meistens einen Pull-Up Widerstand, um ihre Funktion erfüllen zu können
- Open-Source/Open-Emitter-Eingänge (selten vorhanden) benötigen meistens einen Pull-Down Widerstand, um ihre Funktion erfüllen zu können

### LED und Vorwiderstand

Bei LED's lieber die Kathode an den Mikrocontroller anschliessen und die Anode mit einem Pull-Up an der Speisung (Bild 1).

**Grund:** Mikrocontroller erträgt Sink Current meistens besser als Source Current (ich glaube ca. 1/2 Silizium Die Area für selben Strom).

Bild 2 = Lösung mit praktisch gar keiner Strombelastung für µP

Bild 1

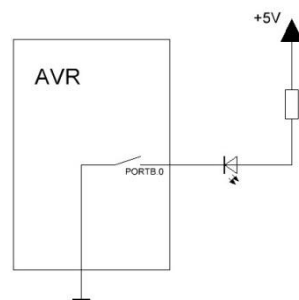
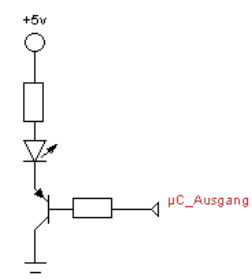
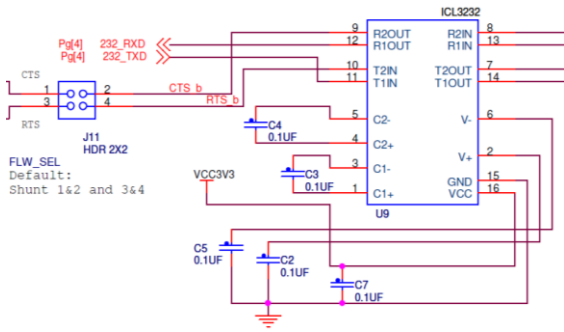
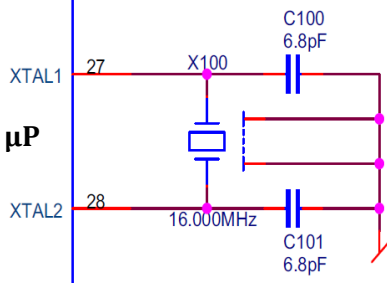
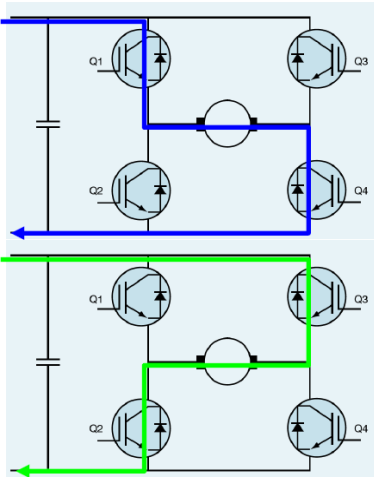
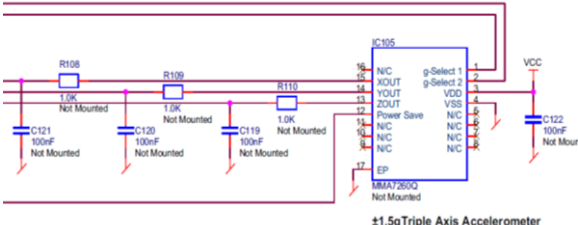
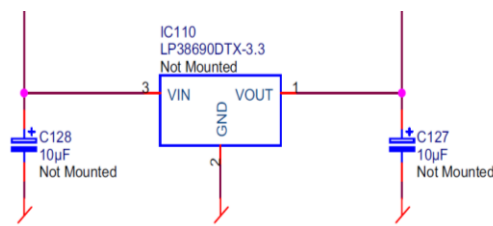
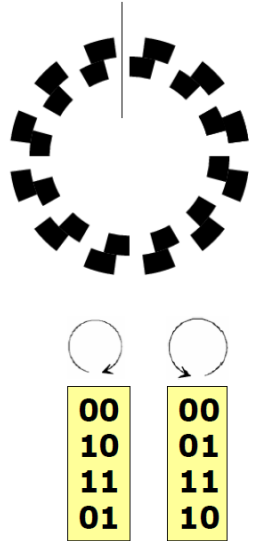
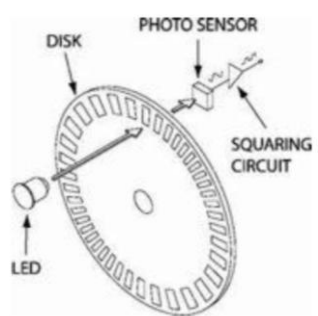


Bild 2





| RS-232 Treiber / Level-Shifter                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Quarze und Oszillatoren                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  <p><b>Funktionsweise</b><br/>Ein RS-232 Level Shifter ändert wie der Name schon sagt die Signalpegel. Zwischen <math>\mu P</math> und Level Shifter beträgt die Spannung TTL Niveau (3.3V/5V o.ä.) und nach dem Level Shifter normalerweise <math>\pm 9 \dots 15V</math>. Dadurch erhält man eine bessere Störfestigkeit (SNR wird grösser).</p>                                                                                                          |  <p><b>Funktionsweise</b><br/>Ein Oszillator oder Quarz erzeugt einen Clock. Von diesem externen Referenzclock ausgehend können nun die <math>\mu C</math>-Clocks (CPU, Bus etc.) generiert werden.<br/>Es kann bei vielen <math>\mu P</math>'s auch ein interner Oszillator verwendet werden, jedoch ist seine Frequenz meistens ungenauer als die eines externen Oszillators.</p>                                                                                                                                                                                |
| H-Brücke                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|                                                                                                                                                                                                                                                                                                                                                                                                                                                           | <p>Eine H-Brücke kann zur Steuerung eines Motors verwendet werden. Dabei kann der Motor in beide Drehrichtungen betrieben werden (blau = z.B. vorwärts / grün = rückwärts)</p> <p><u>Idee:</u></p> <ul style="list-style-type: none"> <li>• 4 Schalter (Transistoren)</li> <li>• Jeder individuell ansteuerbar</li> </ul> <p><u>Anforderungen:</u></p> <ul style="list-style-type: none"> <li>• Exaktes Timing</li> <li>• Schalter müssen synchron schalten</li> </ul> <p><u>Motortreiber (Signale):</u></p> <ul style="list-style-type: none"> <li>• Drehrichtung</li> <li>• PWM (für Geschwindigkeit)</li> <li>• Andere (Nothalt, etc.)</li> </ul> |
| A/D Wandler (ADC)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 3-Axis Accelerometer (Beschleunigungssensor)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <p>Ein A/D-Wandler oder ADC wandelt einen analogen Wert in einen digitalen Wert um. Folgend ein paar Eigenschaften eines ADC:</p> <ul style="list-style-type: none"> <li>• Resolution in Bit, 12 Bit für <math>\mu P</math> sehr häufig</li> <li>• Das Wandlungsverfahren in einem <math>\mu P</math> ist meistens „sukzessive Approximation“</li> <li>• Full Scale Wert wird von einer Referenzspannung vorgegeben (z.B. 3.3V)</li> <li>• Eine kürzere Wandlungszeit (in Samples/s) bedeutet meist auch eine ungenauere Messung</li> </ul> |  <p><b>Funktionsweise</b><br/>Generiert aus der Beschleunigung in allen 3 Dimensionen (X, Y, Z) drei Spannungen. Diese können vom <math>\mu C</math> mittels A/D-Wandler eingelesen werden.</p>                                                                                                                                                                                                                                                                                                                                                                  |

| Spannungsregler                                                                                                                                                                                                                                                                       | Quadratur Encoder                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  <p><b>Funktionsweise</b><br/>Generiert aus seiner höheren Spannung bei „VIN“ eine tiefere Spannung bei „VOUT“ (z.B. 3.3V). Die Beiden Kondensatoren dienen als Filterung/Stützung der Speisung.</p> | <p><b>Funktionsweise</b><br/>Wandelt einen Winkel in ein digitales Signal um (z.B. bei Motoren). Dabei sind die 2 Signale (innerer und äusserer Ring) im Gray-Code codiert (nur ein einziger Bitwechsel beim nächsten Zustand).</p>  <p><b>Informationen:</b></p> <ul style="list-style-type: none"> <li>• Geschwindigkeit</li> <li>• Drehrichtung</li> </ul> <p><b>Ausführungen:</b></p> <ul style="list-style-type: none"> <li>• Mechanisch <ul style="list-style-type: none"> <li>○ prellen, schlecht für high speed</li> </ul> </li> <li>• Mangetisch <ul style="list-style-type: none"> <li>○ Hall Sensoren, für raue Umgebungen</li> </ul> </li> <li>• Optisch <ul style="list-style-type: none"> <li>○ Keine mechanische Trägheit, High speed</li> </ul> </li> </ul>  |

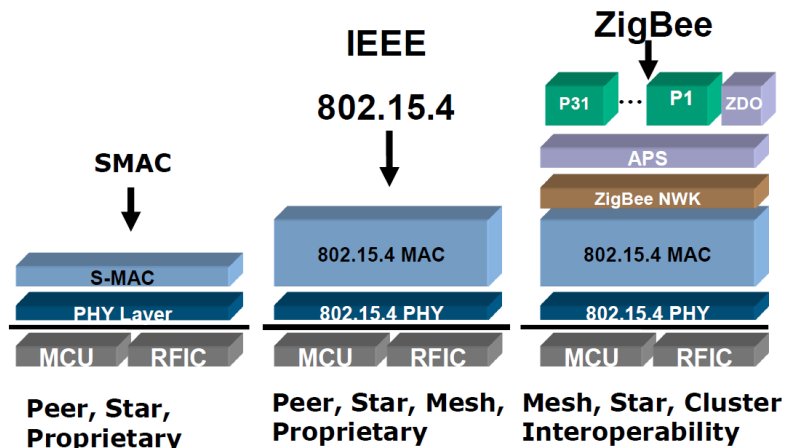
## IEEE (not MEP)

| Was ist IEEE 802.15.4?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |           |         |               |        |               |          |               |                      |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|---------|---------------|--------|---------------|----------|---------------|----------------------|
| <p>IEEE 802.15.4 ist ein Standard, welcher die Physical (PHY) und Medium Access Control (MAC) Layer beinhaltet.</p> <ul style="list-style-type: none"><li>Für Wireless Personal Area network (WPAN)</li><li>3 Frequenzbänder</li><li>3dBm minimale Übertragungsleistung (500µW)</li><li>Fokus<ul style="list-style-type: none"><li>Low Cost: &lt;5\$</li><li>Low Speed: 250kbit/s</li><li>Low Power: Batterie tauglich</li></ul></li><li>Realtime: Garantierter Zeit Slot</li><li>Integrierte Sicherheit</li></ul>                                                                                                                                                                                                   | <div><div><div>Application</div><div>Application Layer (AL)<br/>Application Framework (AF)<br/>ZigBee Device Objects (ZDO)<br/>Application Support Sublayer (ASP)</div><div>Network (NWK)<br/>Star / Mesh / Cluster - Tree</div><div>Media Access Control (MAC)<br/>Device Types, Channel Access</div><div>Physical Interface (PHY)<br/>868 MHz / 915 MHz / 2.4 GHz</div></div><div><div>IEEE 802.15.4</div><div>ZigBee Alliance</div><div>Customer specific</div></div></div> |           |         |               |        |               |          |               |                      |
| Physical (PHY) Layer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |           |         |               |        |               |          |               |                      |
| <p><b>Tasks</b></p> <ul style="list-style-type: none"><li>Aktivieren/deaktivieren der Transceiver<ul style="list-style-type: none"><li>send, receive, sleep</li></ul></li><li>Receiver Energy Detect (ED)<ul style="list-style-type: none"><li>Ermittelt Signalstärke</li></ul></li><li>Link Quality Indication (LQI)<ul style="list-style-type: none"><li>Qualität des empfangenen Signals</li></ul></li><li>Clear Channel Assessment (CCA)<ul style="list-style-type: none"><li>Bestimmt Medium Aktivität: busy oder idle</li></ul></li><li>Auswahl der Kanalfrequenz<ul style="list-style-type: none"><li>27 channels möglich</li></ul></li></ul>                                                                 | <p><b>Frequenzen</b></p> <p>Frequenzen sind für diverse Dienste reserviert (Mobiltelefone, Fernsehen etc.). Für uns interessant: ISM Band ("Industrial, Scientific, Medical"), da lizenzfrei</p> <table><tr><th>Frequency</th><th>Comment</th></tr><tr><td>433 – 464 MHz</td><td>Europe</td></tr><tr><td>900 – 928 MHz</td><td>Americas</td></tr><tr><td>2.4 – 2.5 GHz</td><td>WLAN/WPAN, worldwide</td></tr></table>                                                          | Frequency | Comment | 433 – 464 MHz | Europe | 900 – 928 MHz | Americas | 2.4 – 2.5 GHz | WLAN/WPAN, worldwide |
| Frequency                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Comment                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |           |         |               |        |               |          |               |                      |
| 433 – 464 MHz                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Europe                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |           |         |               |        |               |          |               |                      |
| 900 – 928 MHz                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | Americas                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |           |         |               |        |               |          |               |                      |
| 2.4 – 2.5 GHz                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | WLAN/WPAN, worldwide                                                                                                                                                                                                                                                                                                                                                                                                                                                           |           |         |               |        |               |          |               |                      |
| <p><b>Problem:</b> Knoten will senden, jedoch ist bereits ein anderer Knoten am senden</p> <p><b>Lösung:</b></p> <ul style="list-style-type: none"><li>CSMA (Carrier Sense Multiple Access) → Wartet, falls jemand schon am Senden ist und versucht es nach einer zufälligen Zeit wieder</li><li>CD (Collision Detection) → „Hört zu“, ob eine Kollision stattfindet. Falls ja: Kollisionssignal senden und später neu versuchen (Beispiel: Ethernet)</li><li>CA (Collision Avoidance) → Für Wireless Protokolle, wenn nicht gleichzeitig gesendet und empfangen werden kann. Falls Carrier frei: Broadcast zu den anderen Stationen, dass sie nicht senden dürfen. Variante: RTS senden und CTS empfangen</li></ul> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |           |         |               |        |               |          |               |                      |
| <p><b>2.4GHz PHY Frame Format:</b></p> <p>Max. Übertragungszeit:<br/>4.25ms bei 250 Kbps</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <div><div>Bytes:51</div><div><div>Preamble + Seq. Nr</div><div>Frame Size</div><div>Max 127 Bytes</div><div>Payload</div></div><div>PHY protocol data unit</div></div>                                                                                                                                                                                                                                                                                                         |           |         |               |        |               |          |               |                      |

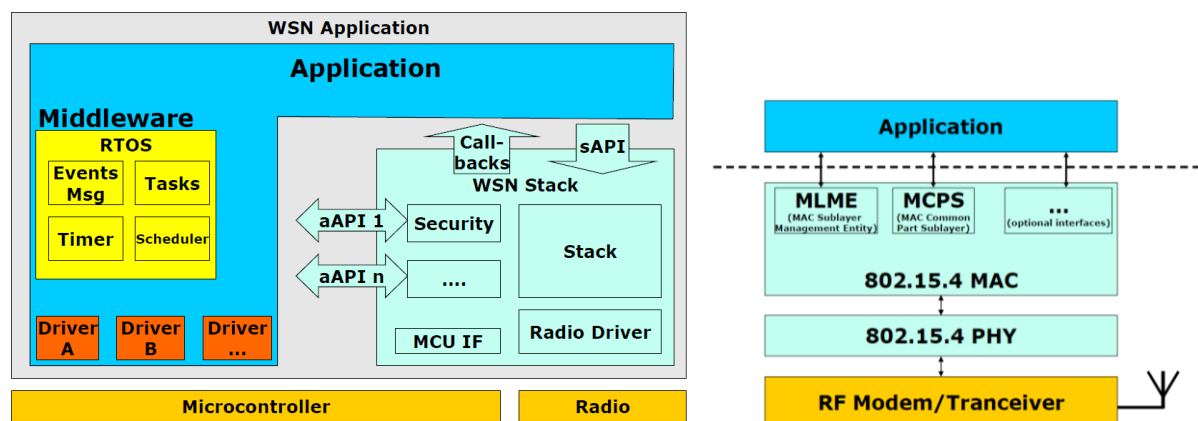
## Media Access Control (MAC) Layer

### SMAC

- + Kleine Grösse (~5Kbyte)
- + Einfach (wenige API calls, einfache State Machine)
- + 2.4GHz Band (lizenzfrei)
- Routing? Addressing?
- Realtime (keine Garantien)
- Keine Verschlüsselung
- Channels: Welchen wählen?
- Low Power?
- Verlorene Pakete (neu senden?)



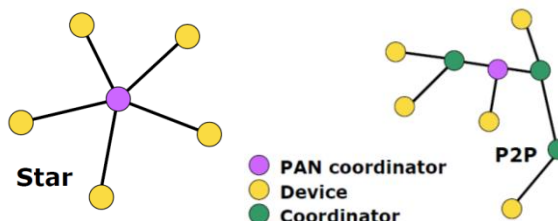
### INTRO Systemübersicht



### Netzwerkstrukturen

- Peer-to-peer (P2P)
- Star

PAN = Public Area Network



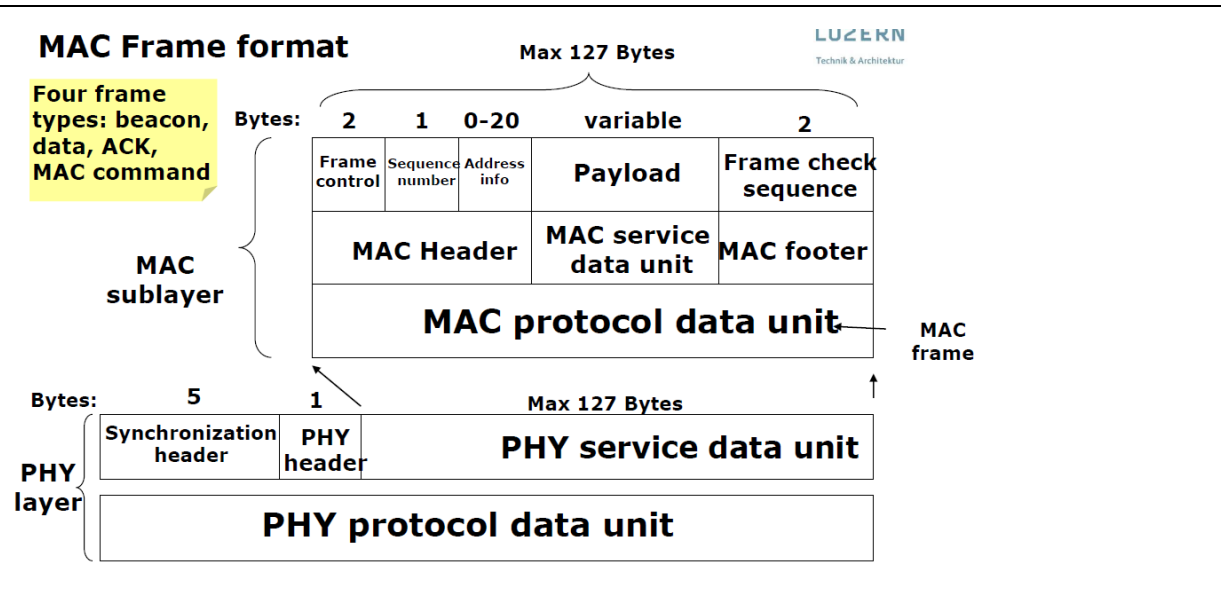
### Device Types

- Reduced Function Device (RFD)
  - Einfach und billig, nur Stern-Topologie → z.B. Netzwerk Ecken, Lichtschalter
- Full Function Device (FFD)
  - Volle 802.15.4 Unterstützung, kann auch Abschlussknoten sein → z.B. Netzwerk Knoten, Router
- Personal Area Network (PAN) Coordinator
  - Spezialfall eines FFD, kennt das gesamte Netzwerk, erstellt Netzwerk ID und Adress-Blöcke

### Coordinator

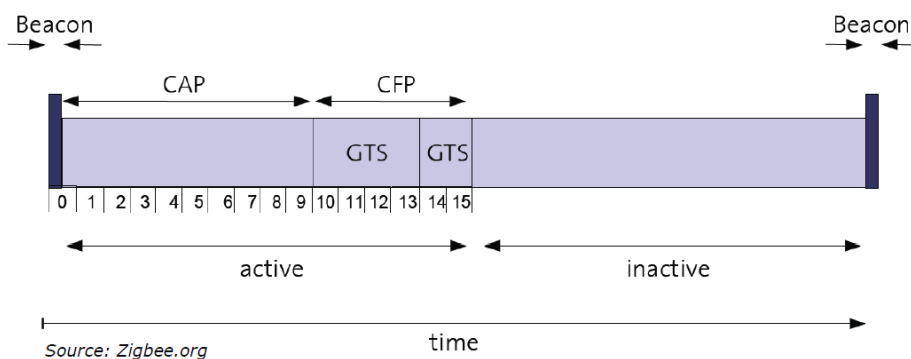
Jedes Device muss eine Verbindung zu einem anderen Device besitzen. Device mit mehreren Verbindungen = Coordinator.

Coordinator: Stellt Synchronisation Services (Beacon/Non-Beacon), Routing etc. zur Verfügung

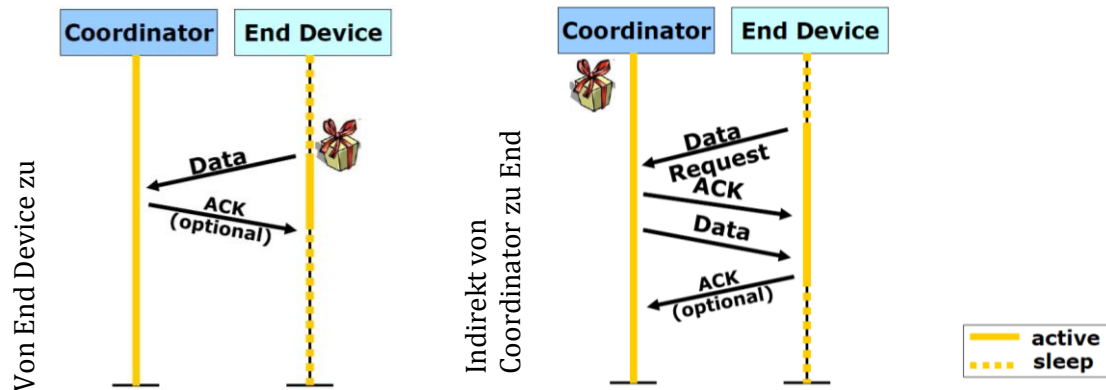


### Channel Access

- Non-Beacon-Mode
  - CSMA/CA Methode: Carrier Sense Multiple Access mit Collision Avoidance
- Beacon Mode: Synchronisation der Knoten über eine gemeinsame Zeitbasis
  - Superframe Struktur:
    - Einrichtung über PANC Konten (Personal Area Network Coordinator)
    - CAP = Contention Access Period: Jeder Knoten kann senden (Kollisionen)
    - CFP = Contention Free Period: Jeder Knoten soll nur in seinem reserv. Time Slot senden
    - GTS = Guaranteed Time Slot: Reserviert für Knoten mit real-time Anforderungen/Bandbreite



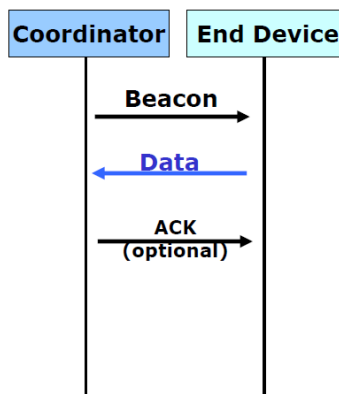
### Data Transfer (Non-Beacon Mode)



### Data Transfer (Beacon Mode)

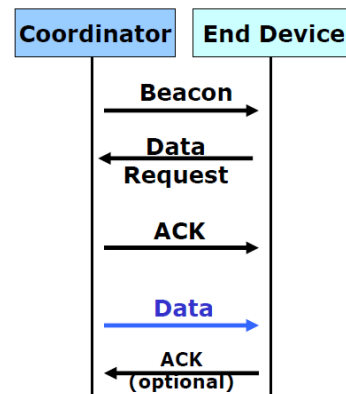
#### End Device (ED) zu Coordinator (CORD)

- Beacon wird periodisch gesendet
- Koordinator könnte immer aktiv sein
- End Device kann schlafen (sleep)
- Geringster Energieverbrauch
- Benötigt präzises Timing
- Beacon-Periode im ms bis min Bereich
- Knoten sendet Daten an Coordinator
- **Direct** Data Exchange



#### Coordinator (CORD) zu End Device (ED)

- Daten an End Device senden
  - End Device ist am schlafen (sleep)
- Coordinator
  - Speichert die Daten
- End Device
  - Wacht periodisch auf
  - Fragt, ob Daten verfügbar sind
  - Wechselt die Polling Rate (Daten Frequenz)



## ZigBee (not MEP)

### Wieso ZigBee?

IEEE 802.15.4 ist schön und gut, jedoch benötigen wir noch folgendes:

- + Alles oberhalb des MAC Layer
- + Routing
- + Zusätzliche Netzwerktopologien: Mesh (Masche, Geflecht), Cluster-Tree („Büschel-Baum“)
- + Gemeinsam genutzte Netzwerk Infrastruktur
- + Interoperabilität

### Was ist ZigBee (Alliance)?

- + (Ist NICHT Open Source. Industrie Konsortium aus End Usern, OEM's, Chip/Software Entwickler)
- + Für Ultra Low Power Wireless Personal Area Network (WPAN)
- + Was macht es anders?
  - Für unterschiedliche Applikationen und Märkte
  - Durchdachtes Netzwerk Management
  - Standardisierung
  - Interoperabilität
  - Low Cost, Low Power
  - Fokus auf niedrige Datenraten
  - Fokus auf Verbindungen mit niedrigem Duty Cycle
- + Architektur
  - Basiert auf IEEE 802.15.4
  - Beinhaltet AL, NWK, MAC und PHY Layer

#### Application

#### Application Layer (AL)

Application Framework (AF)  
ZigBee Device Objects (ZDO)  
Application Support Sublayer (ASP)

#### Network (NWK)

Star / Mesh / Cluster - Tree

#### Media Access Control (MAC)

Device Types, Channel Access

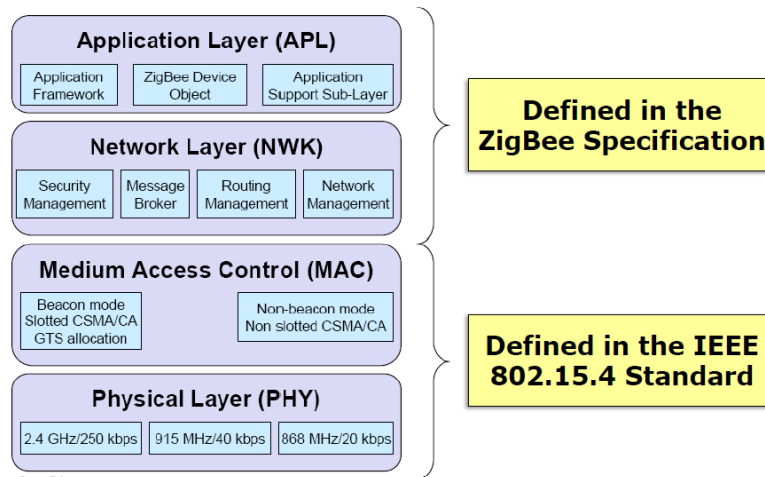
#### Physical Interface (PHY)

868 MHz / 915 MHz / 2.4 GHz

### Wofür ist ZigBee geeignet?

- + Low Energy Netzwerke (beacon, ...)
- + Monitoring und Systemkontrolle mit niedrigen Datenraten
- + Sporadische Daten, geringer Datenumfang, kleine Pakete
- + Abdeckung von grösseren Gebieten (Mesh Netzwerke)
- Abdeckung von grösseren Gebieten ohne Router
- Grosse Daten Pakete / Datenumfang
- Mobile Applikationen mit häufigen verbinden/trennen – zuordnen/trennen
- Daten Streaming (low quality Audio möglich)

### Protokoll Stack: Architektur

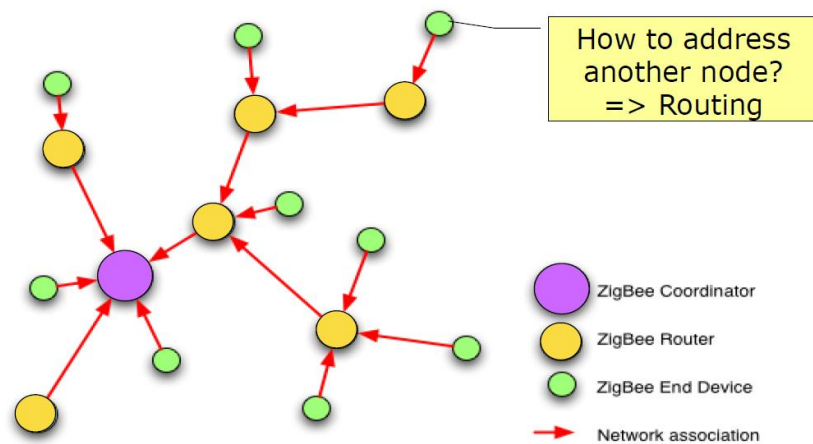


Zwischen den einzelnen Layern sind verschiedene SAP's (Service Access Point) vorhanden.

### ZigBee Device Types

- + ZigBee Coordinator (ZC) ≈ PAN Coordinator (FFD), arbeitet optional als Router
  - Nur ein einziger pro ZigBee Netzwerk
  - Baut das Netzwerk auf und startet es
- + ZigBee Router (ZR) ≈ Coordinator (FFD)
  - Optimale Netzwerk-Komponente, ordnet sich mit ZC oder existierendem ZR zu
- + ZigBee End Device (ZED) ≈ End Device (RFD)
- + Optionale Netzwerk-Komponente
- + Erlaubt kein Zuordnungen und Routing

### Netzwerk Struktur



### Routing Methoden



Node sends packet to all recipients in range. If node is not the final recipient, then it will forward the packet.

**AODV  
Mesh**

Ad Hoc on-Demand Distance Vector Routing): Node sends ROUTE REQUEST (RREQ) to neighbor. The recipient sends a ROUTE REPLY (RREP) in case it is the final destination of the packet, or in case he knows the destination. Otherwise he sends a RREQ.

**Tree**

Node sends packet to parent node. Parent node delivers packet to its own parent node or to his own child node (in case the final destination is a child node).

### Profiles

- IPM (Industrial Plant Monitoring) = Prozessüberwachung industrieller Systeme
  - Druck, Temperatur, IR, ...
- HA (Home Automation) = Haus/Gebäude oder Büroräume
  - Licht, Schalter, HVAC (Klima etc.), Wasserpumpen, Storen, Einbrecher Alarmsysteme. ...



## Laboratory Short Courses

### Exploring Embedded C

- Find the memory map of your microcontroller and list the address ranges of RAM and ROM
  - Siehe "Memory Map" des Datenblattes
- What data types does your compiler support and how many bits of storage are required for each?
  - char                8 Bit
  - short int        16 Bit
  - long int            32 Bit
  - float             32 Bit
  - double            64 Bit
  - "int" ist generell abhängig vom Zielsystem (vgl. 8-Bit oder 32-Bit CPU)

### Variables

- What does "scope" of a variable mean?
  - Bereich, wo die Variable gültig ist, d.h. wo man sie „sieht“
- A variable is defined inside a function and the compiler does not allocate a specific memory location in RAM for its storage. a) Is this an automatic or static variable? b) Where is it stored? c) What is its scope? d) Is data stored in this variable during one function call available to the function in the next call?
  - a) automatic
  - b) stack
  - c) nur innerhalb der Funktion
  - d) nein
- A static variable is defined inside a function. a) What is its scope? b) Where is storage space allocated for it? c) Is data stored in this variable during one function call available to the function in the next call?
  - a) nur innerhalb der Funktion
  - b) RAM
  - c) ja
- A variable is defined outside a function. a) Is this a static or automatic variable? b) If another separately compiled function wishes to use this variable, what must be done in this function to do this?
  - a) static
  - b) sie muss als „volatile“ deklariert werden

### Minimal Startup

- With minimal startup code for the S08, you should get an error or warnings when you make the project? What does it mean?
  - Warning message: "Initialization data lost." → Initialisierungswerte wurden überschrieben oder gar nicht erst beschrieben
- What changes must you make to your program to compensate for the problem?
  - Da der minimale Startup Code nur den Stackpointer initialisiert, müssen Sie sich um allfällige Initialisierung von globalen Variablen selber kümmern; z.B. in einer Init() Routine.

### Plain Char

- For your HCS08 compiler, is 'char' a signed or unsigned type?
  - CodeWarrior 10.3: signed

### Using Individual Bits in a Memory Location

- Does your compiler produce bit addressing instructions such as BSET and BCLR?
  - Yes ???

### CodeWarrior I/O Port and Bit Addressing

- What does the following line of code accomplish?  
extern volatile PTFDSTR PTFD @0x00000040;
  - Port D @ Adresse 0x00000040 wird als „flüchtig“ deklariert
- Explain how the structure PTFDSTR works to allow you to access PTFD as a byte or as individual bits in the port.
  - **Annahme:** Es wird immer das ganze Byte beschrieben. Mit Masken könne jedoch auch nur einzelne Bits beschrieben werden.

### Using Interrupts

- What makes an interrupt handler or service routine different than an "ordinary" function?
  - Eine ISR wird nur auf externen Abruf ausgeführt. D.h. sie wird nicht vom Programmcode selber direkt aufgerufen, sondern von einem externen Event wie z.B. einer steigenden Flanke am Pin XY.

### Connecting a C Program to an Assembly Language Program

- Does your compiler allow you to insert assembly language instructions "in-line" with your C code? If so, how do you do this?
  - Ja. Mit dem Keyword "asm {xy\_code};" kann Code eingefügt werden.
- How does your C program transfer arguments to and from the assembly language program?
  - Es wird so oft es geht über die Register der CPU gearbeitet und möglichst wenig über den Stack

## Serial I/O Interfaces – RS-232-C

### Explore 2

- What part would you chose to include in an embedded system design to convert from the CMOS logic levels of the SCI to RS-232-C and provides at least two input and two output signals?
  - MAX3232CSE+ (3...5.5V, 2 Treiber)

### Stimulate 2

- Write a programming algorithm (a design) for a DCE device that allows it to transmit data only when another device (DTE or DCE) is ready for it.
  - Set CTS
  - Read RTS
  - If RTS ok send, else repeat loop
- Write a programming algorithm (a design) for a DTE device that allows it to transmit data only when another device (DTE or DCE) is ready for it.
  - Set RTS
  - Read CTS
  - If CTS ok send, else repeat loop

## Interrupts using C

- Can you give three examples of I/O devices for which interrupts could be used to synchronize the microcontroller with the I/O?
  - Push Button; Handshaking bei Kommunikationsschnittstellen (z.B. SPI); externe Events wie z.B. fallende Flanken oder Brown Out
- Assume you are designing microcontroller systems to be used in an automotive application. Give three examples of important events that would lend themselves to being implemented in a microcontroller using an interrupt system.
  - Temperatur überschritten; Drehzahlmessung eines Motors (Fächerscheibe / Lochscheibe); Prozesskontrolle wie z.B. Schritt 3 von 5 erreicht

### Explore 1

- Where in your documentation do you find the vector locations (or vector addresses) for interrupts?
  - Im Datenblatt unter MCU Resets, Interrupts etc. (Kapitel 12.5)
- How does your microcontroller allow for asynchronous events to occur and be recognized?
  - Mittels Prioritäten kann die Reihenfolge der ISR's eingestellt werden. Tritt ein Interrupt auf, wird das Interrupt Flag gesetzt.
- How does your microcontroller branch to the correct interrupt service routine?
  - Beim Interrupt wird die Adresse der ISR (Vektor-Adresse) in den Program Counter (PC) geladen und somit als nächstes diese Adresse angesprungen.
- How does your microcontroller return to the interrupted program at the point it was interrupted?
  - Vor dem Sprung in die ISR wird die nächste Adresse im Program Counter (PC) auf den Stack geladen und am Schluss der ISR wieder in den Program Counter zurückgeladen.
- How does your microcontroller allow the programmer to globally enable and disable all interrupts?
  - In einem Register wird das allgemeine Interrupt Enable Flag gesetzt.
- How does your microcontroller allow the programmer to enable and disable selected interrupts?
  - Mit der "Interrupt Mask" können einzelne Interrupts enabled oder disabled werden (in einem Register werden Bits gesetzt)
- How does your microcontroller disable further interrupts so the first can be serviced without being interrupted?
  - Mittels Prioritäten oder "Enter Critical"
- How does your microcontroller deal with multiple, simultaneous interrupts?
  - Führt die Interrupts je nach Priorität aus.
- What can cause a pending interrupt?
  - Wenn schon ein interrupt ausgeführt wird, wenn der IRQ kommt
- How do you reset a flag that caused an interrupt?
  - Am besten am Anfang der ISR unter Berücksichtigung der Prozesseigenschaften.
- What happens if you do not reset the flag in the interrupt service routine?
  - Der Interrupt wird immer wieder in einer Endlosschleife ausgeführt.

### Stimulate 2

- There is always a delay between the interrupt request and when the CPU starts to execute the interrupt service routine. This is called the interrupt latency. Give three components that cause interrupt latency.
  - Interrupt asynchron zu clock; CPU führt aktuellen Befehl zu Ende aus; evtl. Register sichern; evtl. ist schon ein anderer Interrupt mit selber oder höherer Priorität am ausführen
- Give an advantage and a disadvantage of automatically pushing registers onto the stack?
  - Pro: Definierter Zustand nach Zurückspringen aus der ISR; muss nicht selbergemacht werden
  - Contra: Nicht unbedingt bei jedem Interrupt nötig (benötigt Zeit und Speicher)

### Explore 4

- How do you signify to the compiler that a function should be treated as an interrupt handler or service routine?
  - **ISR**(XY\_Interrupt) {Put ISR Code here}

## Analog Input Sampling

### Stimulate 1

- For an A/D with 8-bits and a 5 volt maximum (full scale) input, what is the smallest change in the input signal that can be detected?
  - 19.6mV  $\rightarrow 5V * [1/(2^n - 1)]$
- For an A/D with 10-bits and a 5 volt maximum (full scale) input, what is the smallest change in the input signal that can be detected?
  - 4.88mV

### Stimulate 2

- What elements affect the conversion time for various A/D converters?
  - Resolution, Wandlungsverfahren, System clock frequency
- Your microcontroller's A/D may have its own clock derived from the system clock. How is the A/D clock frequency determined?
  - System clock frequency und über ein Register (z.B. Clock divider)

### Stimulate 3

- What does bandwidth limited mean?
  - Frequenzbereich ist begrenzt (keine unendlichen Frequenzen!)
- What kind of electronic filter must you use to limit the bandwidth of a signal?
  - Tiefpass
- You wish to digitize a signal whose maximum frequency is 1 kHz with your A/D.
  - What is the minimum sample rate (in samples/second) that can be used?
    - 125 (Sukzessive Approximation)  $\rightarrow 1\text{kHz} / n$
  - What is the maximum conversion time?
    - 8ms  $\rightarrow 1\text{s} / (\text{samples/second})$

### Stimulate 4

- $\text{Dynamic Range} = \frac{V_{MAX}}{V_{NOISE}}$
- What is the dynamic range of a signal whose maximum is 26 volts and whose noise is 26 mV?

- 1000
- Dynamic range is often expressed in decibels. What is the dynamic range of the signal above in decibels?
  - $60\text{dB} \rightarrow 10^{\left(\frac{xx\text{ dB}}{20}\right)} = 1000$

#### Stimulate 5

- Consider a sinusoidal signal  $v(t)$  where  $V_{\text{MAX}}$  is 5 volts and  $f_{\text{MAX}}$  is 1 kHz. The aperture time of the A/D is  $1\mu\text{s}$ . What is the worst-case change in voltage, as a percent of full scale that can occur?  $\rightarrow$  *Analog Input (Sinus) =  $v(t) = V_{\text{MAX}} * \sin(2\pi * f_{\text{MAX}} * t)$* 
  - $\Delta v(t) = v(1\mu\text{s}) - v(0) = 0.55\text{mV}$
  - $\Delta v(t) \text{ in } \% = \frac{0.55\text{mV}}{5\text{V}} * 100 = \underline{\underline{0.011\%}}$

#### Explore 3

- What influences the aperture time in your microcontroller?
  - Wie schnell der „Sample and Hold“-Block ist; Wandlungszeit

#### Auswahlkriterien für A/D-Wandler

1.  $2^n \leq \frac{V_{\text{MAX}}}{V_{\text{NOISE}}}$
2.  $2^n \geq \frac{V_{\text{MAX}}}{V_{\text{Smallest\_Signal}}}$
3. *Conversion Time*  $< \frac{1}{2 * f_{\text{MAX}}}$
4. *Sampling Frequency*  $> 2 * f_{\text{MAX}}$
5. *Aperture Time*  $t_{\text{AP}} \leq \frac{1}{2\pi * f_{\text{MAX}} * 2^n}$

#### Stimulate 6

- A temperature transducer produces an analog voltage from 0 to 5 volts over a range of temperatures from 0 to  $100^\circ\text{C}$ . When an oscilloscope is used to look at the output of the transducer, we see a noise level of about 1 mV peak-to-peak. How many bits are needed in the A/D so that the electronic noise is less than the quantization level?
  - 1 bis 11 Bits (Formel 1)
- The same temperature transducer is being used to provide a temperature display of 0 to  $100^\circ\text{C}$  with a resolution of  $1^\circ\text{C}$ . How many bits are needed in the A/D for this case?
  - 7 Bit ( $2^7 = 128 > 100$ )
- You wish to digitize a 100 kHz signal.
  - What is the minimum sample frequency?
    - $200\text{kHz} = 2 * f_{\text{Signal}}$
  - What is the maximum conversion time?
    - $5\mu\text{s} = 1 / f_{\text{Sample\_Minimum}}$
- You look up the specifications for an A/D and find its conversion time is  $16\mu\text{s}$ . What is the maximum frequency that can be converted?
  - $31.25\text{kHz} \rightarrow 16\mu\text{s} = \frac{1}{2 * f_{\text{MAX}}} \rightarrow f_{\text{MAX}} = \frac{1}{2 * 16\mu\text{s}} = 31.25\text{kHz}$

#### Stimulate 9

- Propose a circuit using two diodes that will limit the voltage at the input to the A/D converter to at most one diode drop above the maximum input and one diode drop below the minimum.
  - Clamping Dioden (1 Diode mit Kathode gegen Speisung; 1 Diode mit Anode gegen GND)