

Systems

Transforming Systems	Transforming an inputstream into an outputstream -> Process quality, hardware must be efficiency	Video Encoder, Router, Switch
Reactive Systems	Reacting on an inputstream -> Controlled by external events, guarantee an output	Regler, Anti-Blockier-System
Interactive Systems	Interacting with humans -> Kurze Antwortzeiten, grosse Systemauslastungen	HMI, Billetautomat, Handy

Real-time

A Realtime system answers always at the right time (timeliness) and with a correct result (correctness).

- Antwortet unabhängig von der Systemauslast
- Antwortet deterministisch und vorhersehbar
- Hard-Realtime: Antwort nur akzeptiert, wenn das Resultat innerhalb der gewünschten Zeit ankommt
- Soft-Realtime: Resultat wird degradiert je nach Ankunftszeit der Antwort

Synchronization

Werden Daten von einem System ins andere übertragen, so muss sichergestellt werden, dass die Daten richtig übertragen werden. (computation speed = Abarbeitungszeit)

Realtime	Es wird eine bestimmte Zeit gewartet (loops) und hofft, dass die Abarbeitungszeit ausreicht + keine Hardware nötig - Ineffizient, da der Controller warten muss - Abhängig von der Clockrate und deshalb je nach Compiler anderes Delay
Gadfly	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 + einfach und geringe Latenzzeit - eventuell ist ein HW Flag notwendig - benötigt viel Performance weil ständig gepollt wird
Interrupt	Der Prozess wird unterbrochen, sobald der andere Prozess bereit ist + effizient, weil nicht gewartet wird + weniger overhead im Code - muss HW-mäßig unterstützt sein - Programmstatus muss vor Unterbruch gesichert werden (Reentrancy)

Interrupts & Reentrancy

Interrupt Sequence	<ul style="list-style-type: none"> - 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
Die ISR muss möglichst kurz gehalten werden. Es können Steuersignale (volatile flag) gesetzt werden für Mainprg.	
Reentrant wenn: → Nicht Selfmodifying →	
Prioritäten	Master Interrupt Flag (It enables/disables the interrupt globally) Non – nested Interrupts Simple Priority List Priorities and sub priorities

	<p>Interrupts mit den höheren Prioritäten können ISR mit kleineren Prioritäten unterbrechen. Wird die 2.2 unterbrochen, so wird sie automatisch weitergeführt, wenn 3.3 fertig ist. Die Subpriorität wird erst dann nötig, wenn zwei neue Routinen aufgerufen werden mit der gleichen Hauptpriorität.</p>
Reentrancy	<pre>#define ENTER_CRITICAL() { asm PSHA; asm TPA; asm SEI; asm STA CCR_reg; asm PULA; } #define EXIT_CRITICAL() { asm PSHA; asm LDA CCR_reg; asm TAP; asm PULA; } PSHA = Alles auf Stack legen TPA = Statusregister in Akku legen (Zu diesem Zeitpunkt sind Interrupts noch ON) SEI = Interrupts ausschalten STA CCR_reg = Akku in Adresse CCR_reg speichern ----- Subroutine programmieren ----- PSHA = Alles auf Stack legen LDA CCR_reg = Statusregister in Akku legen von der Adresse CCR_reg TAP = Statusregister von Akku laden (Interrupts sind nun wieder ON) PULA = Stack laden</pre>

Storage Class

```
static int myVar; /* this variable is visible only inside this compilation unit */
int myExternalVar; /* variable can be referenced from another compilation unit */
int myExternalVar; /* variable can be referenced from another compilation unit */
```

Declaration and Definition

```
extern int myVariable; /* a variable declaration */
extern int foo(void); /* a function declaration */

static int myVariable; /* Variable declaration and definition.
Memory for sizeof(int) is allocated */

extern int myVariable; /* Variable definition. Memory for sizeof(int) is not allocated */

int myVariable; /* a variable definition with external storage class */
```

Declaration
allocate no memory.

Definition
allocate memory for named object.

Makros / ANSI-C

<pre>#ifdef PL_BOARD_IS_SRB #define PL_NOF_LEDS 5 #elif defined(PL_BOARD_IS_FRDM) #define PL_NOF_LEDS 3 #else #error „Unknown board?” #endif</pre>	<pre>/* LED.h */ #include "platform.h" #if PL_HAS_LED void LED_Init(void); #else #define LED_Init() /* nothing */ #endif #define ENABLE_INTERRUPTS __asm CLI;</pre>	<pre>#define MY_DELAY 3 #define PREDELAY 5 #define POST_DELAY 2 #define DELAY [PREDELAY + POST_DELAY] return totalDelay(int noIterations) { return noIterations * DELAY; } void delay(void) { Wait(totalDelay(MY_DELAY)); }</pre> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">DELAY wird ersetzt PRE_DELAY + POST_DELAY</div>
--	---	---

Header- & Sourcefiles

```
/* 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);

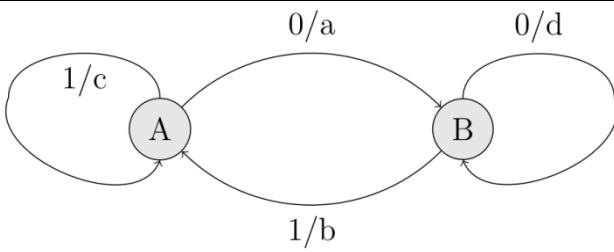
#endif /* __DRV_H__ */
```

```
/* main.c */

#include "drv.h"

void main(void) {
    DRV_Init();
}
```

State Machines



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

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}}};
```

```
typedef enum {
    A, B
} StateT;

static StateT state;
static uint8_t tbl[2][2][2] =
{{{{B,a},{A,c}}},
 {{{B,d},{A,b}}}};
```

```
void Loop(void) {
    uint8_t r;
```

```
    for (;;) {
        r = Read();
        Output(tbl[state][r][1]);
        state = tbl[state][r][0];
    }
}
```

```
void Init(void) {
    state = A;
    Loop();
}
```

```
void Loop(void) {
    uint8_t r;

    for (;;) {
        r = Read();
        switch(state) {
            case A:
                if (r==1) {
                    Write(c);
                } else {
                    Write(a);
                    state = B;
                }
                break;
            case B:
                if (r==1) {
                    Write(b);
                    state = A;
                } else {
                    Write(d);
                }
                break;
        } /* switch */
    } /* for */
}

void Init(void) {
    state = A;
    Loop();
}
```

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;
```

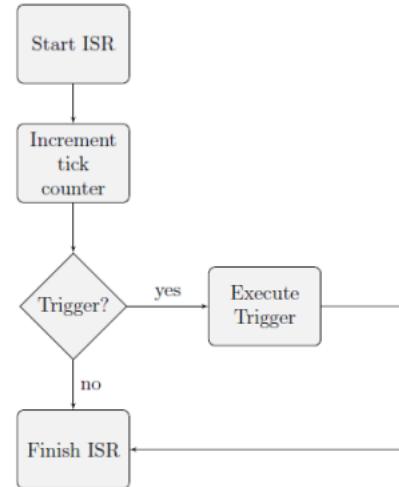
-Generische Struktur:

```
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
}

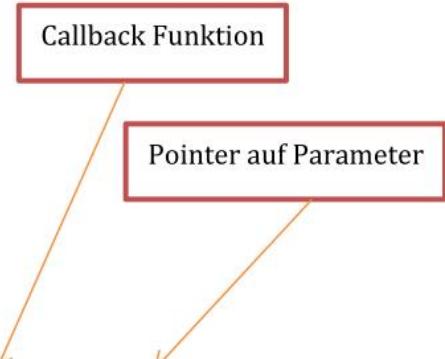
static bool checkCallback(void) {
    TRG_TriggerKind trg = (TRG_TriggerKind) 0;
    TRG_Callback callback = (TRG_Callback) 0;
    TRG_CallBackDataPtr data = (TRG_CallBackDataPtr) 0;
    bool retVal = FALSE;

    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:

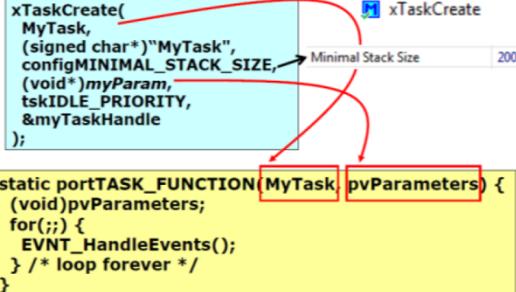
```
void main( void ) {
    TMR_Init();
    TRG_Init();
    EnableInterrupts();
    TRG_SetTrigger( TRG_HEARTBEAT, 0, LED_HeartBeat, NULL); // set now
    for(;;) {} // an der Stelle von NULL geht auch ein Parameter
}

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



The diagram illustrates the relationship between the 'Callback Funktion' and 'Pointer auf Parameter' (pointer to parameter). Two orange arrows point from these labels to the 'NULL' pointer in the line of code: 'TRG_SetTrigger(TRG_HEARTBEAT, 0, LED_HeartBeat, NULL);'. This visualizes how the function call provides a pointer to the callback function and a pointer to its parameters.

FreeRTOS

FreeRTOS Memory Schemes <i>Speicher allozieren für bspw. Tasks / Queues / Semaphoren</i>	
<i>Scheme 1</i>	Alloziert nur Speicher. Kein vTaskDelete(), vQueueDelete(), ...
<i>Scheme 2</i>	Speicherblock kann freigegeben werden. Verbindet freie Speicherblöcke nicht → Fragmentierung möglich! Ungünstig für zufällige alloc/free Sequenzen
<i>Scheme 3</i>	Scheme für standard malloc() und free()
<i>Scheme 4</i>	Verbindet freie Speicherblöcke
<i>Malloc()</i>, <i>Free()</i> und <i>FreeHeap()</i>	
<pre>void *pvPortMalloc(size_t xWantedSize); void vPortFree(void *pv); size_t xPortGetFreeHeapSize(void);</pre>	<u>Beispiel</u> <pre>bufP = (char_t*)pvPortMalloc(sizeof("Hello")); vPortFree(bufP);</pre>
Task Creation (<i>xTaskCreate()</i>, <i>xTaskDelete()</i>)	Create Task and start FreeRTOS Scheduler
 <p><u>ACHTUNG:</u> Die Minimal Stak Size wird in Elementen gerechnet, NICHT in Bytes! D.h. bei 32-Bit CPU und Stack Size = 200 → 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>
<pre>static portTASK_FUNCTION(Task1, pvParameters) { for(;;) { LED1_Neg(); } }</pre>	

FreeRTOS API	
vTaskDelay(#ofTicks) Der Task wartet nach der Ausführung so viele Ticks bis zur nächsten Ausführung. Tick = 10ms, vTaskDelay(1..1,9) => 0..10ms Verzögerung 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>
vTaskDelayUntil(&TickDerLetztenAusführung, #ofTicks) 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.	xLastWakeTime = xTaskGetTickCount(); vTaskDelayUntil(&xLastWakeTime, 10);
<u>Weitere Funktionen (S. 5-14 Folien SW6e):</u> <ul style="list-style-type: none"> • vTaskStartScheduler(), vTaskEndScheduler() • uxTaskPriorityGet(), vTaskPrioritySet() • vTaskSuspendAll(), vTaskSuspend() • vTaskResumeAll(), vTaskResume(), vTaskResumeFromISR() • vTaskYIELD() • void taskENABLE/DISABLE_INTERRUPTS(void) • void taskENTER/EXIT_CRITICAL(void) 	<u>Erläuterungen:</u> <ul style="list-style-type: none"> • Scheduler starten / stoppen • Task Priorität modifizieren • Task unterbrechen • Zu Task zurückkehren und ausführen -nur „FromISR“ sind in ISRs erlaubt! • Erzeugt einen Task-Switch • Interrupt-Handling ohne Nesting • Interrupt-Handling mit Nesting
Idle Task	Task Transition Diagramm
-Wird erstellt mit dem ersten Aufruf von <i>xTaskCreate()</i> . <u>Pseudo Code für Idle Task</u> <pre>static void prvIdleTask(void *pvParameters) { for(;;) { RemoveDeletedTasksFromList(); if (!configUSE_PREEMPTION) { taskYIELD(); } else if (configIDLE_SHOULD_YIELD) { if (NofReadyTasks(tskIDLE_PRIORITY)>1) { taskYIELD(); } } IdleHook(); /* Ergänzung mit eigenem Code*/ } /* möglich, Delays sind verboten! */ }</pre>	<pre> graph TD S[Suspended] -- "vTaskSuspend() called" --> R[Ready] R -- "vTaskResume() called" --> S R -- "vTaskSuspend() called" --> B[Blocked] B -- "Blocking API function called" --> R B -- "Event" --> R S -- "vTaskSuspend() called" --> B </pre>
Idle Should Yield Konfiguration (I = Idle Task)	
- configIDLE_SHOULD_YIELD set to 0 <p>Tasks mit derselben Prio werden „time-sliced“</p>	- configIDLE_SHOULD_YIELD set to 1 <p>nach Beenden abgeben => Task haben verschiedene Laufzeiten!</p>

Events

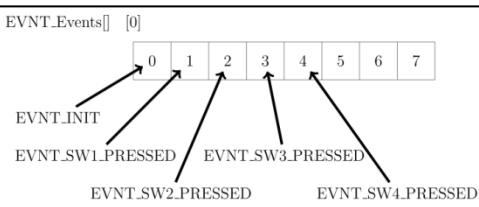
```
typedef uint8_t EVNT_Handle; Declaration a type for the event handle
/*!< We support up to 256 different events */

typedef enum {
    EVNT_INIT,           /*!< System Initialization */
    EVNT_SW1_PRESSED,    /*!< SW1 pressed */
    EVNT_SW2_PRESSED,    /*!< SW2 pressed */
    EVNT_SW3_PRESSED,    /*!< SW3 pressed */
    EVNT_SW4_PRESSED,    /*!< SW4 pressed */
    EVNT_NOF_EVENTS,     /*!< Must be last one */
} EVNT_Handle;
```

Another option is to define an enumeration to list of the different events being used. Be careful, the compiler might be more efficient with handling `uint8_t` than enumeration type

Event Data Structure

```
static uint8_t EVNT_Events[((EVNT_NOF_EVENTS-1)/8)+1];
/*!< Bit set of events */
```



Event auslösen

```
if (KEY2_Get()) { /* key pressed */
    EVNT_SetEvent(EVNT_SW2_PRESSED);
}
```

```
/*! \brief Event module initialization */
void EVNT_Init(void) {
    uint8_t i;

    i = 0;
    do {
        EVNT_Events[i] = 0; /* initialize data structure */
        i++;
    } while(i<sizeof(EVNT_Events)/sizeof(EVNT_Events[0]));
}
```

Events setzen/löschen

```
void EVNT_SetEvent(EVNT_Handle event) {
    EnterCriticalSection();
    SET_EVENT(event);
    ExitCriticalSection();
}

void EVNT_ClearEvent(EVNT_Handle event) {
    EnterCriticalSection();
    CLR_EVENT(event);
    ExitCriticalSection();
}

bool EVNT_EventIsSet(EVNT_Handle event) {
    bool res;

    EnterCriticalSection();
    res = GET_EVENT(event);
    ExitCriticalSection();
    return res;
}
```

Event Handling

```
void EVNT_HandleEvent(void (*callback)(EVNT_Handle)) {
    /* Handle the one with the highest priority. Zero is the event with the highest priority. */
    EVNT_Handle event;

    EnterCriticalSection();
    for (event=(EVNT_Handle)0; event<EVNT_NOF_EVENTS; event++) { /* does a test on every event */
        if (GET_EVENT(event)) { /* event present? */
            CLR_EVENT(event); /* clear event */
            break; /* get out of loop */
        }
    }
    ExitCriticalSection();
    if (event != EVNT_NOF_EVENTS) {
        callback(event);
        /* Note: if the callback sets the event, we will get out of the loop.
         * We will catch it by the next iteration.
         */
    }
}
```

Integration

```
void main(void) {
    EVNT_SetEvent(EVNT_INIT);
    for (;;) {
        EVNT_HandleEvent(APP_HandleEvent);
    }
}
```

Auswerten

```
void APP_HandleEvent(EVNT_Handle event) {
    switch(event) {
        case EVNT_INIT:
            /* write welcome message */
            LCD_WriteString("System startup...");

        case EVNT_SW1_PRESSED:
            SND_Beep(300); /* beep for 300 ms */
            /* changes desired temperature */
            ChangeTemperature(1); /* increase temperature */
            SendTemperature(); /* use transceiver */
            break;

        case EVNT_SW2_PRESSED:
            SND_Beep(300); /* beep for 300 ms */
            /* changes desired temperature */
            ChangeTemperature(-1); /* decrease temperature */
            SendTemperature(); /* use transceiver */
            break;
    } /* switch */
}
```

Doxxygen

<pre>/*! * \file Bindet ohne Angaben den Namen ein * \author Name * \brief Implements a test routine * \param [in] val Input value * \return Error code */ uint8_t MyTest(uint16_t val) { /*! \todo Implement function */ Return 0 ; }</pre>	<p>uint8_t MyTest (uint16_t val)</p> <p>Implements a test routine.</p> <p>Parameters: [in] val Input value</p> <p>Returns: Error code</p> <p>Todo: Implement function</p> <pre>return 0;</pre>
<pre>\dot digraph myfsm { rankdir=LR; node [shape=doublecircle]; n1 [fillcolor=lightblue,style=filled,label="A"]; node [shape=box]; n2 n3; n1 -> n2 [label="line1"]; n1 -> n3 [label="line2"]; n2 -> n2 [label="line3"]; n2 -> n3 [label="line4"]; } Grafik: \image html Led.jpg [format] [file]</pre>	<pre> graph LR A((A)) -- "line1" --> n2[n2] A -- "line2" --> n3[n3] n2 -- "line3" --> n2 n2 -- "line4" --> n3 </pre> <p>MSC Sequence Diagram:</p> <pre> sequenceDiagram participant Client participant Server Client->>Server: get accel activate Client Note over Client: arcgradient = 8; a [label="Client"], b [label="Server"]; a->b: [label="get accel"] a=>b: [label="get accel"] a>b: [label="ack"] a<=b: [label="accel data"] deactivate Client </pre>

Graycode

Binary to Gray <ul style="list-style-type: none"> Binary um eins nach rechts verschieben EXOR verrechnen 	<p>Binary: 11101101_b</p> <p>Gray: 10011011_g</p>
Gray to Binary <ul style="list-style-type: none"> Erste Zahl nach unten Exor mit der zweiten Zahl 	<p>Gray:</p> <p>Binary:</p> <p>Gray: 10011011_g</p> <p>Binary: 11101101_b</p>

Semaphoren

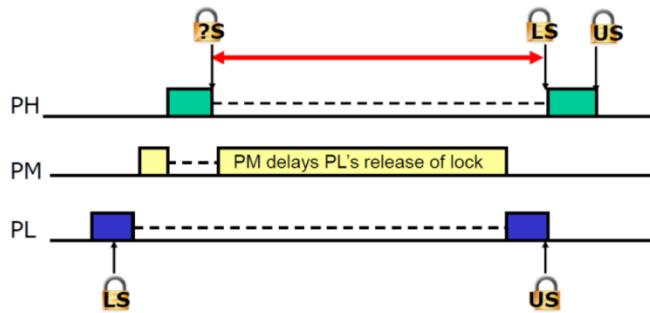
Ermöglichen einen sicheren Zugriff auf gemeinsame Ressourcen / Daten und ermöglichen eine Synchronisation

- 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

Lösungansätze:

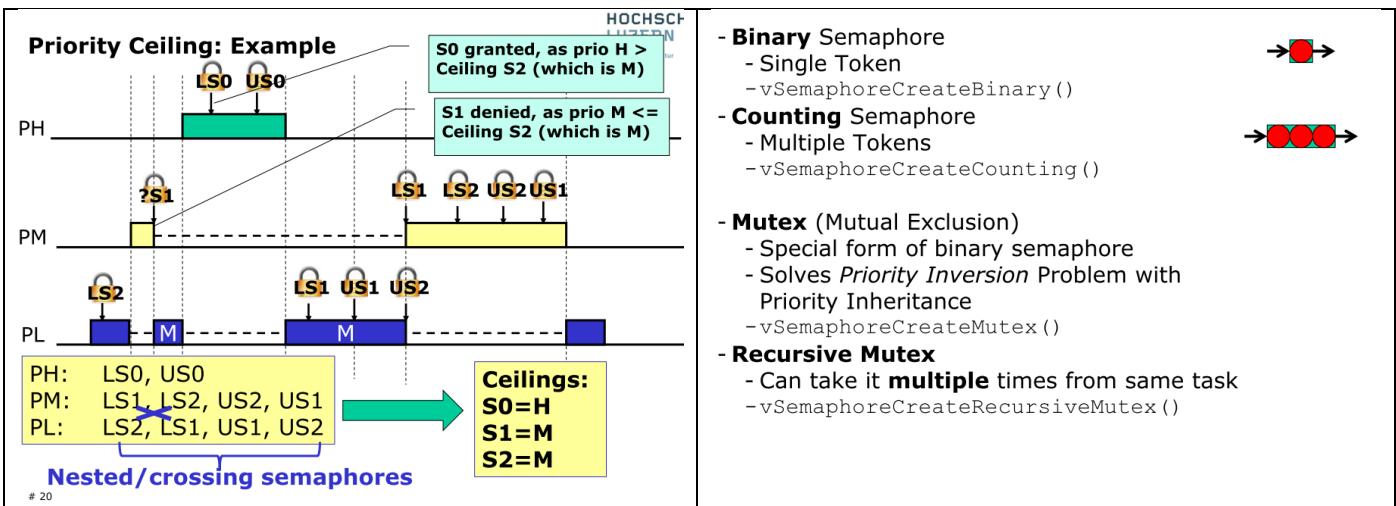
- Priority Inheritance (Vererbung): der Task mit tiefer Prio erbt die Prio eines höheren Tasks falls dieser eine Anfrage startet. Nach dem Durchlauf erhält er wieder die alte Prio. Ein mittel priorisierte Task wird mit „push-trough-Blocking“ unterbrochen. Bei mehreren Semaphoren können trotzdem noch 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

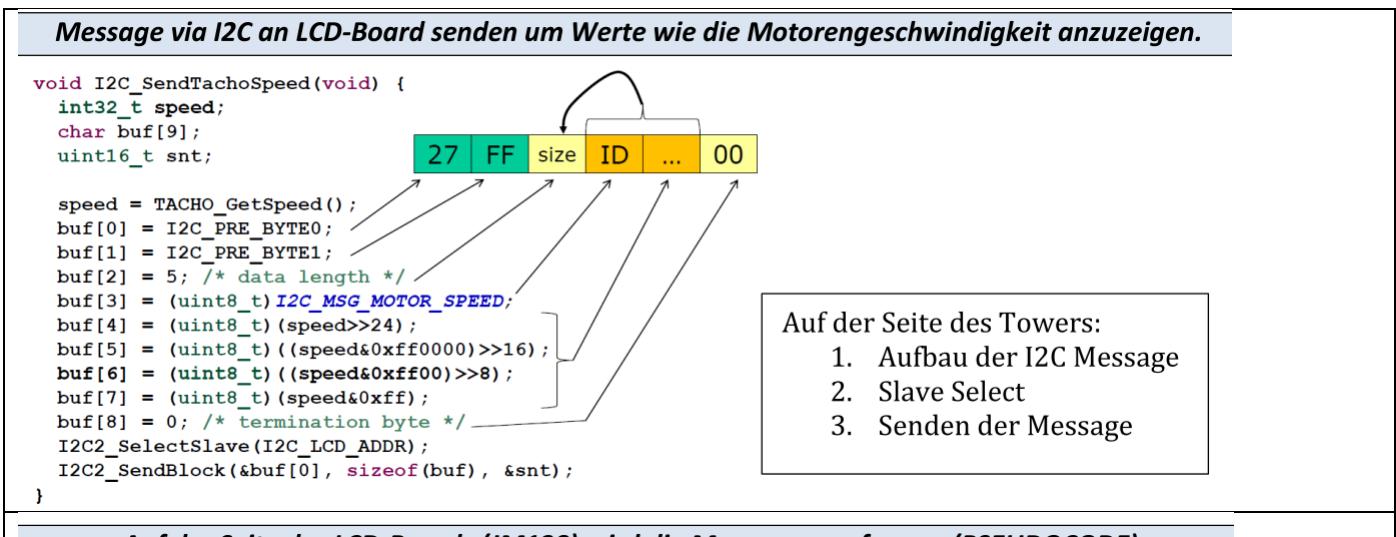
- Direct: Falls bei einer Anfrage die Ressource bereits vergeben ist
- Push-through: Entsteht bei Priority Inheritance
- Transitive: Task 1 wird von Task 2 geblockt, wobei Task 2 von Task 3 geblockt wird

-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
    }
}
```



LCD-Board



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

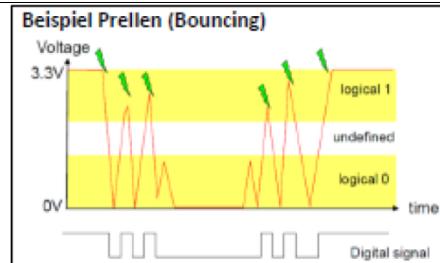
```
Interrupt: I2C_OnReceiveData()
    res= I2C1_RecvBlock(&data, 1, &nofBytesReceived);
    // reads a block from input buffer
    buf[index] = data;
    index++;
    // Interrupt trifft so oft auf, bis der Input-Buffer leer ist und alle Daten empfagen sind
    // Trifft dies zu wird die Message interpretiert
    ...
} 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 update 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

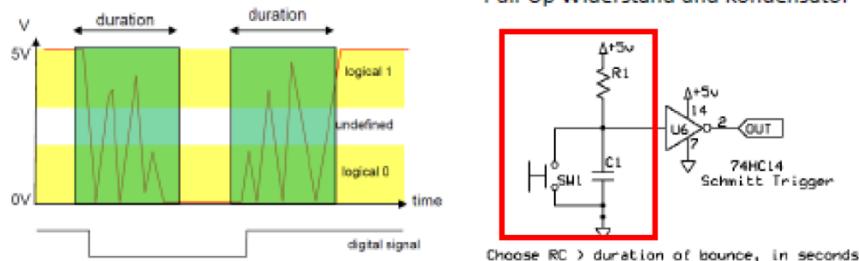
Debouncing

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

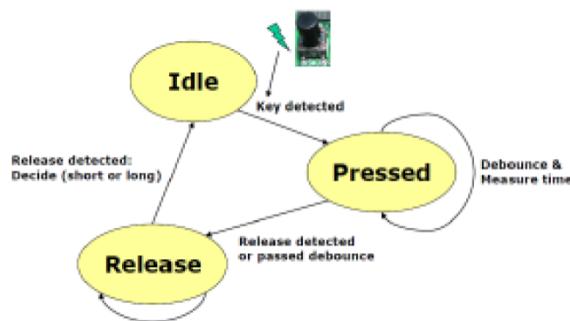


-solution: Es wird während der Zeit in der der Schalter prellt der Input nicht mehr abgefragt (Software) oder die entstehenden Signalpeaks gefiltert (Hardware). Die zu überbrückende Zeit muss empirisch ermittelt werden.

-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 Tastendrücken 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.

Shell

- Goal: Einfaches Userinterface als Schnittstelle zwischen µC und Computer
- SCI : Rx (SDA), Tx (SCL) ⇔ RS232 Transceiver / Level Shifter ⇔ Serial COM Port
 - + einfach, günstig, wenig HW Ressourcen, keine Treiber - RS232 Port nötig, kein Power
- USB : DP, DM ⇔ USB Bridge (falls nicht integriert) ⇔ Virtual COM Port
- Functiondescription: Shell over USB:

```

USB_Init();           // Ringbuffer (Rx, Tx), FSL_USB_Stack
Shell_Init();         // Mutex erstellen, ShellTask erstellen
ShellTask:           char cmd_buf[32], usb_tx_buf[64], *msg;
                     // USB1_App_Task(usb_tx_buf);
                     // regelmäßig aufrufen: übermittelt TX Buffer
FSSH1_ReadAndParseCMD (cmd_buf, stdio, ParseCommand);
                     // Liest Zeilenweise vom Input, schreibt in cmd_buf und ruft via
                     // callback ParseCommand() auf, falls neue Daten vorhanden sind.
ParseCommand(cmd, stdio)
                     if (UTIL1_strcmp(cmd_buf, „help“) == 0)           // msg = „help“?
                     FSSH1_SendStr (“help text”, stdio);          // Antworttext
                     if (MyOwnParser (cmd_buf, stdio)!= 0) {return Err ;}
MyOwnParser (cmd_buf, stdio)
                     if (UTIL1_strncmp(cmd_buf, “channel”, sizeof(“channel”)-1 == 0)
                         const char *p; long val;
                         p = cmd + sizeof („channel“);
                         (void) UTIL1_xatoi (&p, &val);           // string to num
                         doSomethingwithReadValue (val);          // set num

```

strcmp:

vergleicht zwei Strings

strncmp:

vergleicht die ersten n Zeichen zweier Strings

xatoi:

ascii to integer

strcat(a, size, b):

hängt String b String a an und speichert es in String a

num32sToStr(a, size, num): signed 32Bit number to String A

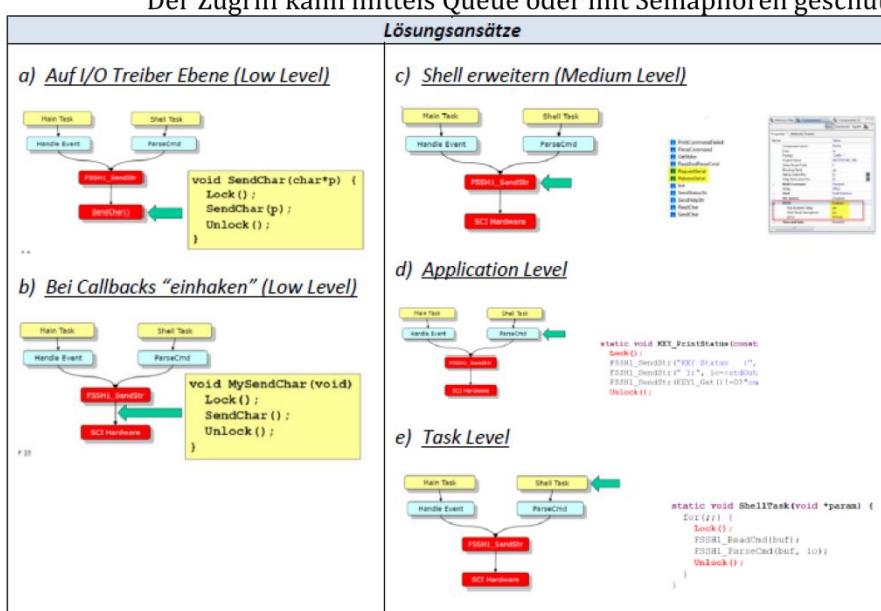
sizeof(“Hallo”): Gibt 6 zurück. ‘\0’ gehört bei Strings dazu

-protected access:
die

Es muss sichergestellt werden, dass nicht zwei Quellen gleichzeitig auf Shell schreiben wollen. Sonst kann es vorkommen, dass die beiden Ausgaben wirr durchmischt werden.

Quelle 1: „Hallo“, Quelle 2: „Welt“, Resultat: „H W a e l l t o“

Der Zugriff kann mittels Queue oder mit Semaphoren geschützt werden.

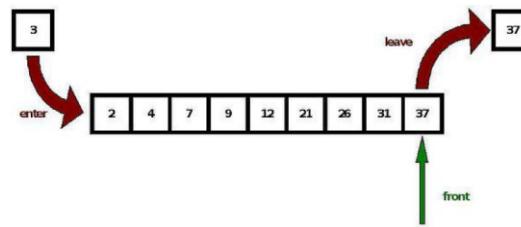


Queue

Eine Queue ist eigentlich nichts anderes als seine 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);
```



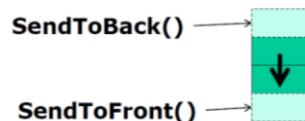
Queue löschen

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

Daten Queue senden

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

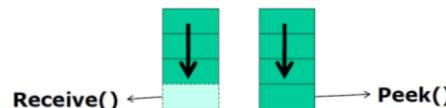
- Es kann ein Delay übergeben werden, falls Element noch kein Platz



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;
    }
}
```

RECAP – Fragen

Radio Transceiver	
What does a Radio Transceiver do?	Establish communication over the air
Name 3 components of the Radio Transceiver	SMAC, PHY, TRSVR
How is the payload packaging done	Each Layer adds the whole previous Layer as its payload. It adds its own data at the start and end of the previous layers data
Name the 4 Layers of IEEE802.15.4	Application, Network, Media Access Control, Physical Interface
Tacho	
Wieso braucht die Delta-Time Implementation ein Timeout?	Da es bei einem Stillstand (oder sehr langsam Bewegungen) kein Schrittwechsel gibt und somit die Zeit (delta T) ins Unendliche laufen würde.
Welche Methode ist bei schnellen Bewegungen geeigneter?	Die Variante der Berechnung mittels Positionsvergleich (Delta Pos-Approach).
Welche Methode ist für langsame (wenige Steps) geeigneter?	Der Delta T-Approach (Messung der Zeit für einen Schritt).
Wieso handelt es sich bei der Tacho-Implementation um eine Schätzung und nicht eine Messung? (Was sind die Faktoren?)	Folgende Faktoren haben Einfluss auf die Ungenauigkeit der geschätzten Geschwindigkeit: <ul style="list-style-type: none"> - Gewisse Ungenauigkeit bereits beim Quadratur Encoder - Quantisierungseffekt (Digitale Encoder) - Backward Looking (Steps der Vergangenheit werden zur Berechnung verwendet)
Nennen Sie eine Datenstruktur für eine effiziente Implementierung der Positionsvergleich-Variante?	Mit einem Ringbuffer. Der zu überschreibende Wert wird vor dem Überschreiben für die Berechnung verwendet und kann danach überschrieben werden.
Beschleunigungssensoren	
Wie gross ist der kleinste Schritt bei einem Left justified Wert von 13bit auf uint16?	216/213 = 23 = 8
Was ist der Vorteil von Industriellen SD Karten?	Die billigen SD Karten haben oft verschiedene Geschwindigkeiten, Zugriffszeiten und verschiedene Schreibmodi. Die Industriellen sind zuverlässiger und die Ansteuerung ist viel einfacher
Welche Werte müssen bei den Beschleunigungssensoren kalibriert werden?	Offset & Gain bei 1g
Für welche Daten aus dem Accelerometer Teil könnte der Interne Flash verwendet werden?	Justierdaten „Kalibrationsdaten“
Wie ist der Speicher des Flash's aufgebaut und vorauf muss beim Beschreiben von diesem geachtet werden?	Der Flash ist in Blöcken strukturiert und es muss beim Beschreiben beachtet werden das nicht über einen Block mit Code geschrieben wird.
Reflectance	
Was ist der Vorteil der digitalen Messung des Reflectance Sensors?	Es braucht keine analogen Eingänge. Digitale Pins sind genug vorhanden.
Was sind Fehlerquellen der digitalen Messung?	Kondensatoren und Widerstände haben Toleranzen -> ungleiche Entladung
Wozu ist die Diode D1 im Schema auf Seite 5?	Zur visuellen Anzeige für den Benutzer ob die Infrarot Leds in Betrieb sind.
Nenne mir eine kleine Änderung im Schema für Low Power Anwendungen?	R11 und D1 entfernen
Was sind mögliche Fehlerquellen die zu einer höheren Beleuchtung der Phototransistoren führen?	Sonneneinstrahlung, Beleuchtungen, Benachbarte IR-Leds strahlen ab
Wie können mögliche Fehlerquellen minimiert werden?	Abdeckung nach aussen oder zwischen den IR-LEDs (schwierig wegen Platzmangel),
ARM Cortex und Keys	
Sind alle Mikrocontroller mit einem ARM-Cortex M0 gleich?	Nein, nur der Kern ist derselbe, die Peripherie, sowie der Speicher sind unterschiedlich.
Wie viele Stack Pointer hat der ARM Cortex M0?	2, Main SP und Process SP
Welcher Architektur entspricht der ARM Kern? Von Neumann oder Harvard?	Harvard, der Code und SRAM sind am selben Bus
Sind alle 16 Register frei verwendbar?	Nein, nur R0 - R12, R13 => SP, R14 => LR, R15 => PC R16 => xPSR
Wie wird ein kritischer Abschnitt im Code geschützt?	EnterCritical(), ExitCritical()
RTOS	
Welche Methoden für das Task controlling kennst du?	uxTaskPriorityGet/Set(), vTaskSuspend/SuspendAll(), vTaskResume/ResumeAll/ResumeFromISR(), taskYIELD(), taskENABLE/DISABLE_INTERRUPTS(), taskENTER/EXIT_CRITICAL(), TaskDelay/DelayUntil()
Welcher Task wird in priority-based preemptive Scheduling ausgeführt?	Der Task mit der höchsten Priorität
In welcher Situation macht der Aufruf von vTaskEndScheduler() sinn?	Beispielsweise für Ultra-low Power Anwendungen, wenn Energie gespart werden soll
Welche Schemas existieren, um Memory für Tasks zu allozieren?	Schema 1: Nur allozieren, Schema 2: Blöcke können wieder freigegeben werden (Fragmentierung möglich), Schema 3: Wrapper für malloc und free, Schema 4 (neu): Freie Blöcke werden gemerget
Wofür haben wir die Shell-Queue implementiert?	Für den geregelten Zugriff von Informationen zwischen mehreren Prozessen
Regler	
Wozu braucht es überhaupt ein Regler?	Um interne und externe Einflüsse zu korrigieren.
Welche Regelung wird für eine Positionsregelung benötigt?	PD-Regler, weil Motor bereits ein integrales Verhalten besitzt und dieses ausgeglichen werden muss
Für was wird Anti-Windup benötigt?	Über diese Variable wird der maximale I- Anteil festgelegt, der auf das System wirken kann.
Remote Control	
In which order has the RNet-Application to be initialized?	<ol style="list-style-type: none"> 1. Initialize Stack 2. Assign all message handlers 3. Assign the own node address 4. Process Radio StateMachine --> keep doing e.g. in own task!
Which are the different levels of our RNet -	APP : Application Layer

Application?	NWK : Network Layer MAC : Media Access Layer PHY : Physical Layer	
What is a CRC?	CRC is the check sum of a packet. It is used to avoid incorrect packets.	
Whats's an Ad-hoc-Network?	A network with two or more nodes. It connects devices without a fix infrastructure like a Wireless Access Point.	
List one advantage and one disadvantage of an Ad - hoc-Network.	+ safe (no central Routing point needed) - each device works like a router and has to be active a long time (high power consumption)	
Mutual Access		
Worauf muss man achten bei der Wahl der PWM Frequenz im Zusammenhang mit einer H-Brücke?	Man muss auf die maximal zulässige PWM Frequenz der H Brücke achten. Sie steht normalerweise im Datenblatt	
Wie kann Verlustleistung eines PWM gesteuerten Geräts verringert werden?	Bei Höherer Frequenz wird die H Brücke kürzer belastet und verursacht so weniger Wärme.	
Was sollte beachtet werden falls man mit dem Regler ein PWM Signal regeln will.	Das der Regler Aufruf immer Synchron mit dem PWM aufgerufen wird, das das PWM Signal nicht mitten in der High Periode ein Höheren Cycle bekommt.	
Wieso sollten gemeinsam genutzte Daten mit einer Queue oder einem Semaphore geschützt werden?	Um zu verhindern dass die Daten gegenseitig überschrieben werden, oder zweimal das selbe geschieht (Beispiel Suppe)	
Welche Funktionen werden bei Semaphore verwendet?	Lock() und Unlock()	
Ultraschallsensor		
Zähle die vier Anschlüsse des HC-SR04 Ultraschallsensor auf und beschreibe sie kurz.	VCC: 5V Speisung GND: GND ist halt GND... Trigger: Aktiviert die Messung Echo: Zeit des Echos hin und zurück entspricht Distanz zu Objekt.	
Was passiert wenn beim Messen kein Objekt detektiert wird?	Der Timer zählt bis zum Overflow. Dieser wird als „kein Objekt entdeckt“ interpretiert.	
Warum wird das Ergebnis bei einer Umrechnung der Echozeit in die Distanz nicht stimmen, wenn mit Schallgeschwindigkeit von 58 cm/us gerechnet wird? Gibt es weitere Faktoren die das Ergebnis beeinflussen?	Die Distanz des Schalls geht zum Objekt und zurück -> Dies erfordert eine Division durch zwei. Weitere aber weniger beeinflussende Faktoren sind Temperatur, Luftfeuchtigkeit und Luftdichte	
Welcher Variabtentyp kann mittels FreeMaster betrachtet werden?	Globale Variablen	
Über welches Verbindungsprotokoll kommuniziert FreeMaster zu FRDM-Board? Worauf ist bei dieser Konfiguration zu achten?	Mittels UART. Da bereits die Shell über UART kommuniziert, muss diese in der ProcessorExpert Komponente disabled werden. Es empfiehlt sich die Shell Komponente Standartmässig auf die Bluetooth Verbindung zu konfigurieren, damit die Shell trotzdem noch betrieben werden kann.	
Makros		
Was sind die Vor- und Nachteile von Macros?	Vorteile: <ul style="list-style-type: none">• Schnellerer Code• Kleinerer Code• Fixe Einträge im Code können zentral verwaltet werden	Nachteile: <ul style="list-style-type: none">• Interfaces sind Low-Level, es gibt keine Typensicherheit• Verschachtelung des Codes• Erschwertes Debugging
Wie sieht die genaue Definition des led.h-Files aus, damit es nicht rekursiv inkludiert werden kann?	#ifndef __LED_H__ #define __LED_H__ // Inhalt des *.h-Files #endif	
Reentrancy		
3. Was ist eine atomare Anweisung?	Eine atomare Anweisung kann nicht durch einen Interrupt unterbrochen werden.	
Warum sollten die Anweisungen EnterCritical() und ExitCritical() nicht verschachtelt verwendet werden?	Bei der Anweisung EnterCritical(); wird das aktuelle CCR-Register abgespeichert und anschließend das Interruptbit gelöscht. Wird die Funktion nochmals aufgerufen wird nun das bereits geänderte CCR-Register abgespeichert.	
Semaphore und Mutex		
Gegeben sind die Semaphore S1 und S2 sowie die Prozesse PL, PM und PH. PL hat die Priorität 1, PM die 2 und PH die 3 (größere Zahlen bedeuten höhere Priorität). Die Semaphore S1 wird von PL und PM verwendet. Die Semaphore S2 wird von PL und PH verwendet. Wie hoch ist das Priority Ceiling der Semaphore S1 und S2?	Es braucht mindestens drei Prozesse mit drei unterschiedlichen Prioritäten: ein Prozess mit niedriger Priorität der die Semaphore hält, ein Prozess mit mittlerer Priorität der die Ausführung des niedrigpriorisierten Prozesses verhindert und ein Prozess mit hoher Priorität der auf die Semaphore wartet.	
Was gilt es bei der Verwendung von FreeRTOS Semaphore aus Interrupt Service Routinen zu beachten?	Statt der Funktionen xSemaphoreTake bzw. xSemaphoreGive müssen die zugehörigen FromISR-Funktionen verwendet werden: xSemaphoreTakeFromISR bzw. xSemaphoreGiveFromISR.	