

Echtzeitsysteme

Übungen zur Vorlesung

Entwicklungsumgebung (Teil 2)

Tobias Klaus, Florian Schmaus, Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)
Lehrstuhl für Informatik 4 (Verteilte Systeme und Betriebssysteme)
<https://www4.cs.fau.de>

24.10.2016



Übersicht

- 1 Weiterführende Informationen zum Übungsbetrieb
- 2 Wahl geeigneter Datentypen
- 3 Handwerkszeug
 - Oszilloskop-Cursor
 - Zeit in eCos
 - Tiefpassfilter
 - Pulsweitenmodulation
- 4 Interruptbehandlung
 - ISR & DSR
 - eCos-Unterbrechungsbehandlung
- 5 Zusammenfassung



Informationen zum Übungsbetrieb

- Accounts im CIP-Pool?
 - cipan
 - Sprechstunde der CIP-Admins
- Kenntnisse im Umgang mit Terminals hilfreich
 - UNIX Vorkurs der FSI
 - <https://fsi.cs.fau.de/dw/informationen/ese/2015ws/vorkurs>
- Arbeiten im CIP
 - make edit &
- Arbeiten zu Hause
 - /proj/i4ezs/stm32 kopieren
 - ecosenv.sh anpassen
 - make debug ↪ arm-none-eabi-gdb mit Python-Bindings
 - Detaillierte Anleitung folgt ...



Wiederholungen?



- Einführung in GDB
 - EZS-Dashboard
 - https://www4.cs.fau.de/Lehre/WS16/V_EZS/Uebung/Folien/00_GDB-Primer.pdf
- Stack-Aufbau bei Funktionsaufrufen
 - Stack-Tiefe
 - Backtrace
- Komponenten der Toolchain
 - Präprozessor
 - Compiler
 - Flasher
 - ...



Beliebte Fehler

```
1 ezs_printf("Hallo");
```

- Flush der Zeichen erst bei `\n`



Übersicht

- 1 Weiterführende Informationen zum Übungsbetrieb
- 2 **Wahl geeigneter Datentypen**
- 3 Handwerkszeug
 - Oszilloskop-Cursor
 - Zeit in eCos
 - Tiefpassfilter
 - Pulsweitenmodulation
- 4 Interruptbehandlung
 - ISR & DSR
 - eCos-Unterbrechungsbehandlung
- 5 Zusammenfassung



Frage #1

Zu was wird `7/2` ausgewertet?

1. 3.5
2. 3
3. nicht definiert in C

Erklärung

- Standard-Typ für Ganzzahlen ist `int`
- Rest verschwindet bei Ganzzahl-Division



Frage #2

Zu was wird `2/7` ausgewertet?

1. 1
2. 0
3. nicht definiert in C

Erklärung

- Standard-Typ für Ganzzahlen ist `int`
- Rest verschwindet bei Ganzzahl-Division



Frage #3

Zu was wird `7/2.` ausgewertet?

1. immer noch 3
2. 0
3. 3.5

Erklärung

- 2. `==` `2.0` \leadsto `double` auf der rechten Seite
- 7 wird in diesem Ausdruck als `double` behandelt, auch linke Seite
- Division zweier `double` Werte



Frage #5

Zu was wird `1/2 + 1/2` ausgewertet?

1. nicht definiert
2. 0
3. 1 (dank Compileroptimierung)

Erklärung

- `int1 / (<größerer int2)` \leadsto `0 + 0 = 0`
- Compileroptimierung nicht C-Konform



Frage #6

Zu was wird `2 * M_PI` ausgewertet?

1. 6
2. ungefähr 6.28
3. 6.283185307179586476925286766559005768394338798750...

Erklärung

- `M_PI` \leadsto `double`
- `float` Standard-Typ, außer zusätzliches Literal (`3.14f`)
- Begrenzter Wertebereich:
6.283185307179586000000000000000



Frage #7

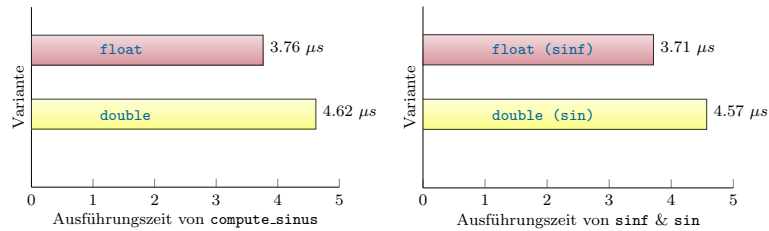
```
1 double a = 0.1;
2 double b = 0.2;
3 float aa = 0.1;
4 float bb = 0.2;
5 if (a+b == aa+bb){
6     ezs_printf("equal\n");
7 }else{
8     ezs_printf("unequal: %.30f != %.30f\n", (a+b), (aa+bb));
9 }
```

Was wird ausgegeben?

1. equal
2. unequal...



Vergleich der Laufzeiten



- Laufzeitzuwachs um **23 %** bei Wechsel **float** → **double**
- Soft Float? Hard Float? hier: Soft Float
- Noch mehr Optimierungspotential? Wo wird die Laufzeit verbraucht?
 - **99 %** der Gesamtlaufzeit für **sinf** und **sin**
- Wahl des Datentyps in Abhängigkeit der Wortbreite (32-Bit Cortex-M4, 8-Bit AVR)
- Spezialbibliothek für Signalverarbeitung mit Integer-Arithmetik
- Spezielle Hardware-Einheiten zur Signalverarbeitung

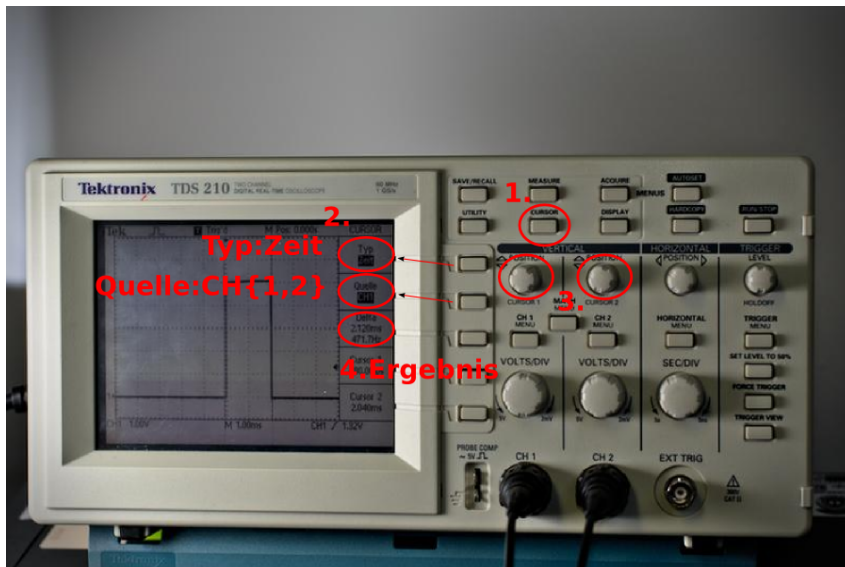


Übersicht

- 1 Weiterführende Informationen zum Übungsbetrieb
- 2 Wahl geeigneter Datentypen
- 3 Handwerkszeug
 - Oszilloskop-Cursor
 - Zeit in eCos
 - Tiefpassfilter
 - Pulsweitenmodulation
- 4 Interruptbehandlung
 - ISR & DSR
 - eCos-Unterbrechungsbehandlung
- 5 Zusammenfassung



Cursor



Umgang mit Zeit in eCos

- Aktuelle Aufgabe: Ausführung soll um feste Zeit *verzögert* werden
~> `cyg_thread_delay()`
- Erwartet Parameter der Einheit *Clock-Ticks* – *Wieso?*
- Zeitmessung nur per Timer möglich ~> Timer-Takt kleinste Einheit

`cyg_clock_get_resolution(cyg_real_time_clock())`
liefert Auflösung der Echtzeituhr:

```
1 typedef struct {  
2     cyg_uint32 dividend;  
3     cyg_uint32 divisor;  
4 } cyg_resolution_t;
```

- $\frac{\text{dividend}}{\text{divisor}}$ ~> Zeit in ns, die ein Tick dauert
- Umrechnung sollte zur Übersetzungszeit erfolgen
Wieso?



Tiefpassfilter

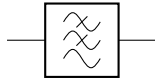


Abbildung: Schaltbild RC-Glied

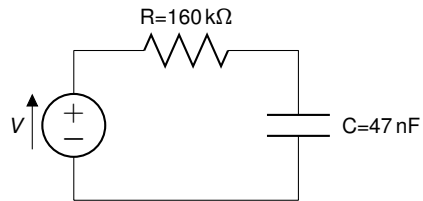
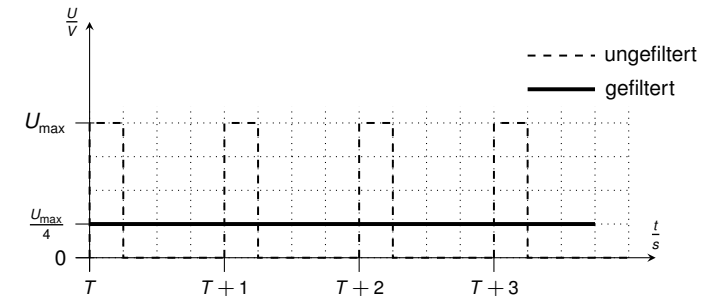


Abbildung: Schaltung RC-Filter auf EZS-Board

- Filterung hochfrequenter Schwingungen
- Zeitkonstante $\tau = R \cdot C = 7,52\text{ms}$
- Grenzfrequenz (Dämpfung um 3dB $\approx 71\%$): $f_c = \frac{1}{2 \cdot \pi \cdot \tau} = 21\text{Hz}$



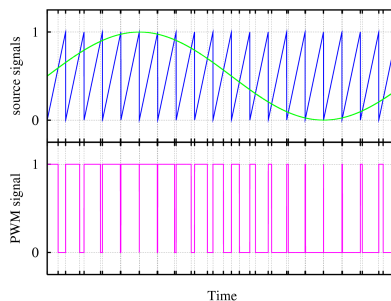
Digital-Analog Umwandlung & Pulsweitenmodulation I



- Einschaltdauer proportional zum Mittelwert des Ausgangssignals
- Variation der Pulsweite oder Einschaltdauer (engl. duty cycle)
- Pulsweitenmodulation (engl. pulse-width modulation, PWM)
- „Pseudo“ Digital-Analog-Wandler (engl. digital-analog converter, DAC)



Digital-Analog Umwandlung & Pulsweitenmodulation II



Verfahren zur Signalerzeugung

- Hardware: Vergleich periodischer Zähler und Wert der Einschaltdauer
- Weit verbreitet: Motorsteuerung, Class-D-Verstärker, Schaltnetzteile, Nachrichtenübertragung, ...
- Mittels Tiefpass \leadsto Digital-Analog-Wandlung
- libEZS: `void ezs_dac_write(uint8_t)` (zusätzlich *echter* DAC auf Board)



Übersicht

- 1 Weiterführende Informationen zum Übungsbetrieb
- 2 Wahl geeigneter Datentypen
- 3 Handwerkszeug
 - Oszilloskop-Cursor
 - Zeit in eCos
 - Tiefpassfilter
 - Pulsweitenmodulation
- 4 Interruptbehandlung
 - ISR & DSR
 - eCos-Unterbrechungsbehandlung
- 5 Zusammenfassung



Wie behandle ich einen Interrupt?

Interrupt-Service-Routinen-Ausführung

- Unverzögerlich, *asynchron*
~ auch innerhalb von Kernelfunktionen!
- Innerhalb ISR *keine Systemaufrufe* erlaubt!
⇒ Anmelden einer Deferrable Service Routine (DSR)

Deferrable-Service-Routinen-Ausführung

- *Synchron* zum Scheduler
- Falls Scheduler nicht verriegelt: *Unverzögerlich* nach ISR
- sonst: Beim *Verlassen* des Kerns

Synonym: *Prolog-Epilog-Schema* bzw. *top/bottom half*



Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR²

```
#include <cyg/kernel/kapi.h>
void cyg_interrupt_create
(
  cyg_vector_t vector,
  cyg_priority_t priority,
  cyg_addrword_t data,
  cyg_ISR_t* isr,
  cyg_DSR_t* dsr,
  cyg_handle_t* handle,
  cyg_interrupt* intr
);
```

- Interruptvektornummer
~ Hardwarehandbuch



²<http://ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html>

Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR²

```
#include <cyg/kernel/kapi.h>
void cyg_interrupt_create
(
  cyg_vector_t vector,
  cyg_priority_t priority,
  cyg_addrword_t data,
  cyg_ISR_t* isr,
  cyg_DSR_t* dsr,
  cyg_handle_t* handle,
  cyg_interrupt* intr
);
```

- Interruptpriorität
- für unterbrechbare Unterbrechungen
(hardwareabhängig)



²<http://ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html>

Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR²

```
#include <cyg/kernel/kapi.h>
void cyg_interrupt_create
(
  cyg_vector_t vector,
  cyg_priority_t priority,
  cyg_addrword_t data,
  cyg_ISR_t* isr,
  cyg_DSR_t* dsr,
  cyg_handle_t* handle,
  cyg_interrupt* intr
);
```

- Beliebiger Übergabeparameter für
ISR/DSR



²<http://ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html>

Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR²

```
#include <cyg/kernel/kapi.h>
void cyg_interrupt_create
(
  cyg_vector_t vector,
  cyg_priority_t priority,
  cyg_addrword_t data,
  cyg_ISR_t* isr,
  cyg_DSR_t* dsr,
  cyg_handle_t* handle,
  cyg_interrupt* intr
);
```

■ Funktionszeiger auf
ISR-Implementierung

Signatur:

cyg_uint32 (*)(cyg_vector_t, cyg_addrword_t)

²<http://ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html>

Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR²

```
#include <cyg/kernel/kapi.h>
void cyg_interrupt_create
(
  cyg_vector_t vector,
  cyg_priority_t priority,
  cyg_addrword_t data,
  cyg_ISR_t* isr,
  cyg_DSR_t* dsr,
  cyg_handle_t* handle,
  cyg_interrupt* intr
);
```

■ Funktionszeiger auf
DSR-Implementierung

Signatur:

cyg_uint32 (*)(cyg_vector_t, cyg_ucount32 count, cyg_addrword_t)

²<http://ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html>

Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR²

```
#include <cyg/kernel/kapi.h>
void cyg_interrupt_create
(
  cyg_vector_t vector,
  cyg_priority_t priority,
  cyg_addrword_t data,
  cyg_ISR_t* isr,
  cyg_DSR_t* dsr,
  cyg_handle_t* handle,
  cyg_interrupt* intr
);
```

■ Handle und Speicher für
Interruptobjekt

²<http://ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html>

ISR-Implementierungsskelett

Beispiel einer minimalen ISR

```
cyg_uint32 isr(cyg_vector_t vector, cyg_addrword_t data) {
  cyg_bool_t dsr_required = 0;
  ...
  cyg_acknowledge_isr(vector);
  if (dsr_required) {
    return CYG_ISR_CALL_DSR;
  } else {
    return CYG_ISR_HANDLED;
  }
}
```

1. Beliebiger ISR-Code
2. Bestätigung der Interruptbehandlung
Wozu ist das gut?
3. Anforderung einer DSR
oder
4. Rückkehr ohne DSR

ISR-Implementierungsskelett

Beispiel einer minimalen ISR

```
cyg_uint32 isr(cyg_vector_t vector, cyg_addrword_t data) {
    cyg_bool_t dsr_required = 0;
    ...
    cyg_acknowledge_isr(vector);
    if (dsr_required) {
        return CYG_ISR_CALL_DSR;
    } else {
        return CYG_ISR_HANDLED;
    }
}
```

1. Beliebiger ISR-Code
2. Bestätigung der Interruptbehandlung
Wozu ist das gut?
3. Anforderung einer DSR
oder
4. Rückkehr ohne DSR



ISR-Implementierungsskelett

Beispiel einer minimalen ISR

```
cyg_uint32 isr(cyg_vector_t vector, cyg_addrword_t data) {
    cyg_bool_t dsr_required = 0;
    ...
    cyg_acknowledge_isr(vector);
    if (dsr_required) {
        return CYG_ISR_CALL_DSR;
    } else {
        return CYG_ISR_HANDLED;
    }
}
```

1. Beliebiger ISR-Code
2. Bestätigung der Interruptbehandlung
Wozu ist das gut?
3. Anforderung einer DSR
oder
4. Rückkehr ohne DSR



ISR-Implementierungsskelett

Beispiel einer minimalen ISR

```
cyg_uint32 isr(cyg_vector_t vector, cyg_addrword_t data) {
    cyg_bool_t dsr_required = 0;
    ...
    cyg_acknowledge_isr(vector);
    if (dsr_required) {
        return CYG_ISR_CALL_DSR;
    } else {
        return CYG_ISR_HANDLED;
    }
}
```

1. Beliebiger ISR-Code
2. Bestätigung der Interruptbehandlung
Wozu ist das gut?
3. Anforderung einer DSR
oder
4. Rückkehr ohne DSR



DSR-Implementierungsskelett

Beispiel einer minimalen DSR

```
void dsr_function(
    cyg_vector_t vector,
    cyg_ucount32 count,
    cyg_addrword_t data)
{
    ...
}
```

1. Anzahl der ISRs, die diese DSR anforderten
~> normalerweise 1
2. Ausführung *synchron* zum Scheduler
Was bedeutet das?



Beispiel einer minimalen DSR

```
void dsr_function(  
    cyg_vector_t vector,  
    cyg_ucount32 count,  
    cyg_addrword_t data)  
{  
    ...  
}
```

1. Anzahl der ISRs, die diese DSR anforderten
→ normalerweise 1
2. Ausführung *synchron* zum Scheduler
Was bedeutet das?



- 1 Weiterführende Informationen zum Übungsbetrieb
- 2 Wahl geeigneter Datentypen
- 3 Handwerkszeug
 - Oszilloskop-Cursor
 - Zeit in eCos
 - Tiefpassfilter
 - Pulsweitenmodulation
- 4 Interruptbehandlung
 - ISR & DSR
 - eCos-Unterbrechungsbehandlung
- 5 Zusammenfassung



- Vorsicht bei der **Wahl des Datentypen**
 - Abhängig von Zielarchitektur
 - Datentypen **beeinflussen die Laufzeit**
 - Performance *nicht notwendig, aber hilfreich* für Echtzeitsysteme
- Handwerkszeug für EZS
 - Arbeiten mit Oszilloskopen
 - Zeiten in eCos
 - DAC, PWM, Filter
- Interruptbehandlungen
 - Möglichst kurze ISR
 - Synchrone Abarbeitungen DSR



- [1] David Goldberg.
What every computer scientist should know about floating-point arithmetic.
ACM Computing Surveys (CSUR), 23(1):5–48, 1991.

