

Hallo Welt!

Teil Zwei

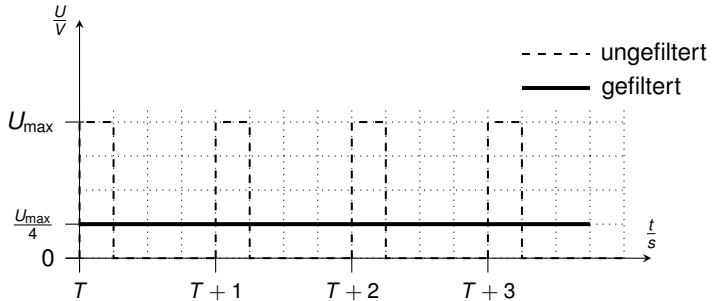
Florian Franzmann Tobias Klaus Peter Wägemann

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

22.10.2015



Pulsweitenmodulation (PWM)

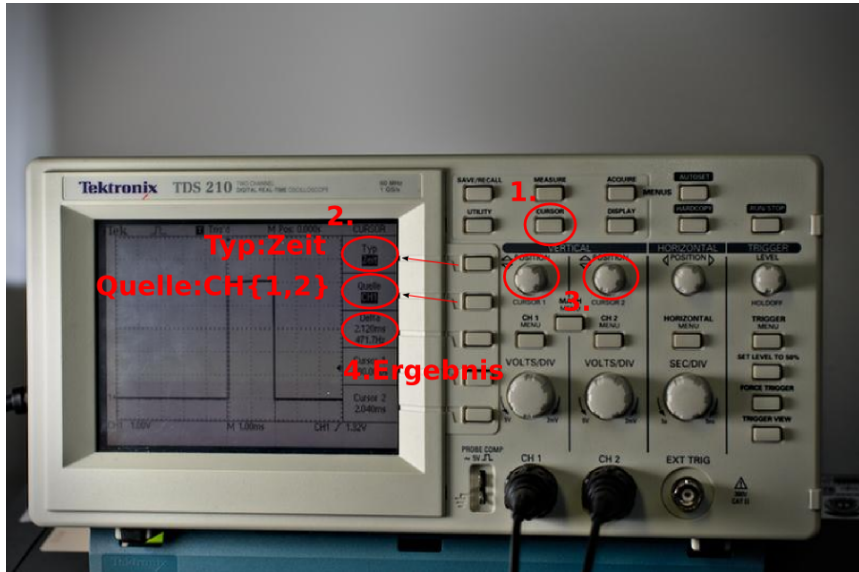


Verfahren zur Signalerzeugung

- Tiefpass $\rightsquigarrow$ Digital-Analog-Wandlung
- Weit verbreitet: Motorsteuerung, Class-D-Verstärker, Schaltnetzteile, Nachrichtenübertragung,...
- libEZX: `void ezs_dac_write(uint8_t)`



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;
```

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



- 1 Handwerkszeug
 - Pulsweitenmodulation
 - Oszilloskop-Cursor
 - Zeit in eCos

- 2 Ausblick
 - eCos-Unterbrechungsbehandlung



Wie behandle ich einen Interrupt?

Interrupt-Service-Routinen-Ausführung

- Unverzüglich, *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: *Unverzüglich* nach ISR
- sonst: Beim *Verlassen* des Kerns

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



Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR¹

```
#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

¹ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html



Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR¹

```
#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)

¹ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html



Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR¹

```
#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

¹ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html



Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR¹

```
#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)

¹ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html



Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR¹

```
#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)

¹ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html



Wie behandle ich einen Interrupt?

Anmeldung von ISR und DSR¹

```
#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

¹ecos.sourceware.org/docs-latest/ref/kernel-interrupts.html



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 | CYG_ISR_HANDLED;  
    } 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



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 | CYG_ISR_HANDLED;  
    } 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



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 | CYG_ISR_HANDLED;  
    } 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



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 | CYG_ISR_HANDLED;
    } 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



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
 $\leadsto$ 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
 $\leadsto$ normalerweise 1
2. Ausführung *synchron* zum Scheduler
 Was bedeutet das?



Fragen?



¹ <https://commons.wikimedia.org/wiki/User:Pensiero~commonswiki>

