

Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

23. November 2022

Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es>

1 Einführung

2 Einführung in Unix

3 TCP/IP in der Praxis

4 Bus-Systeme

4.1 Was sind Bus-Systeme?

4.2 RS-232

4.3 I²C (TWI)

4.4 SPI

4.5 PWM

4.6 Sonstiges

5 Echtzeit

5.1 Was ist Echtzeit?

5.2 Echtzeitprogrammierung

5.3 Multitasking

5.4 Ressourcen

5.5 Prioritäten



Ergänzungen
noch möglich

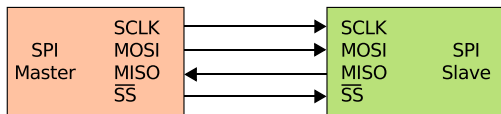
4 Bus-Systeme

4.4 SPI

Serial Peripheral Interface

seriell

- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät



synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt

4 Bus-Systeme

4.4 SPI

Serial Peripheral Interface

seriell

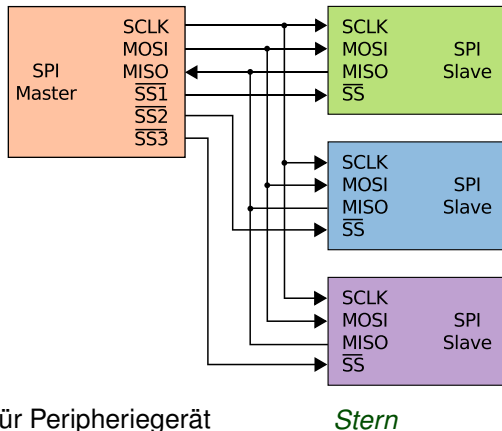
- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



4 Bus-Systeme

4.4 SPI

Serial Peripheral Interface

seriell

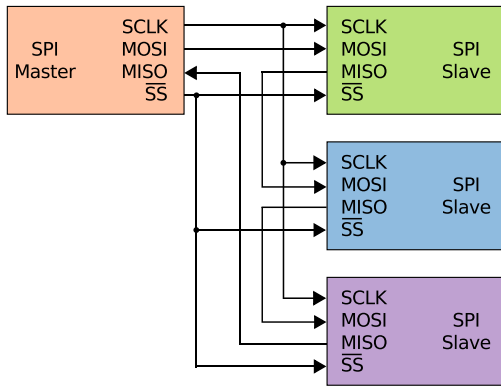
- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



*Kaskade
Daisy Chain*

4 Bus-Systeme

4.4 SPI

Serial Peripheral Interface

seriell

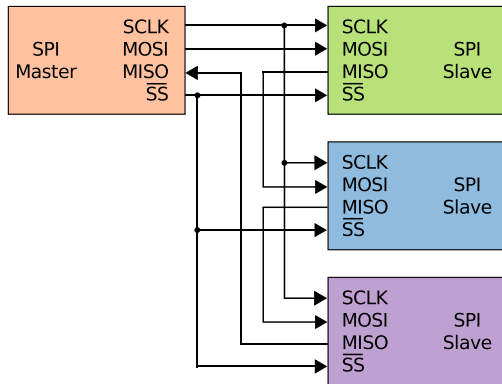
- *MOSI*: Master Out, Slave In
- *MISO*: Master In, Slave Out
- *SCLK*: Taktleitung (Clock)
- \overline{SS} : Slave Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Master* initiiert Kommunikation und steuert Taktleitung
- *Slave* wird über *Slave Select* ausgewählt



*Kaskade
Daisy Chain*

Slave gibt MOSI-Input um 1 Takt verzögert an MISO aus → Master setzt „im richtigen Moment“ \overline{SS}

4 Bus-Systeme

4.5 PWM

Pulsweitenmodulation – *pulse-width modulation*

- Steuerung von Motoren
- Nutzung als allgemeines Protokoll zur Übertragung analoger Werte

4 Bus-Systeme

4.6 Sonstiges

Matrix-Schaltung

- möglichst viele Aktoren/Sensoren
über möglichst wenige digital Inputs abfragen
bzw. über möglichst wenige digitale Outputs steuern
- Beispiele: LED-Felder, Tastaturen

R/2R-Netzwerk

- möglichst viele digitale Inputs
über einen einzigen analogen Input abfragen
- Beispiele: Tastaturen

Eingebettete Systeme

<https://gitlab.cvh-server.de/pgerwinski/es>

1 Einführung

2 Einführung in Unix

3 TCP/IP in der Praxis

4 Bus-Systeme

4.1 Was sind Bus-Systeme?

4.2 RS-232

4.3 I²C (TWI)

4.4 SPI

4.5 PWM

4.6 Sonstiges

5 Echtzeit

5.1 Was ist Echtzeit?

5.2 Echtzeitprogrammierung

5.3 Multitasking

5.4 Ressourcen

5.5 Prioritäten



Ergänzungen
noch möglich

5 Echtzeit

5.1 Was ist Echtzeit?

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung
- Fantasy-Rollenspiel in Echtzeit:
Der Zeitverlauf der Spielwelt entspricht dem der realen Welt.

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung
- Fantasy-Rollenspiel in Echtzeit:
Der Zeitverlauf der Spielwelt entspricht dem der realen Welt.
- Datenverarbeitung in Echtzeit:
Die Daten werden so schnell verarbeitet, wie sie anfallen.

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung
- Fantasy-Rollenspiel in Echtzeit:
Der Zeitverlauf der Spielwelt entspricht dem der realen Welt.
- Datenverarbeitung in Echtzeit:
Die Daten werden so schnell verarbeitet, wie sie anfallen.
- speziell: Echtzeit-Steuerung von Maschinen:
Die Berechnung kann mit den physikalischen Vorgängen schritthalten.

5 Echtzeit

5.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung
- Fantasy-Rollenspiel in Echtzeit:
Der Zeitverlauf der Spielwelt entspricht dem der realen Welt.
- Datenverarbeitung in Echtzeit:
Die Daten werden so schnell verarbeitet, wie sie anfallen.
- speziell: Echtzeit-Steuerung von Maschinen:
Die Berechnung kann mit den physikalischen Vorgängen schritthalten.

→ „Schnell genug.“

5.1 Was ist Echtzeit?

„Schnell genug.“

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

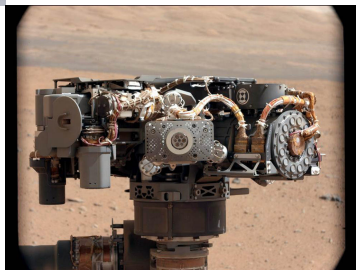
- „Ganz schlecht.“ → *harte Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*



5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*
verspätetes Ergebnis qualitätsmindernd
 - verwenden und Verzögerung in Kauf nehmen
 - verwerfen und Ausfall in Kauf nehmen

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*
verspätetes Ergebnis qualitätsmindernd
 - verwenden und Verzögerung in Kauf nehmen
 - verwerfen und Ausfall in Kauf nehmen
- „Es gibt keinen festen Termin. Möglichst schnell halt.“

5.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*
rechtzeitiges Ergebnis funktionsentscheidend
- „Unschön.“ → *weiche Echtzeit*
verspätetes Ergebnis qualitätsmindernd
 - verwenden und Verzögerung in Kauf nehmen
 - verwerfen und Ausfall in Kauf nehmen
- „Es gibt keinen festen Termin. Möglichst schnell halt.“
→ *keine Echtzeit*

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

Beispiel:

- Eine Motorsteuerung benötigt alle $2000\ \mu\text{s}$ einen Steuerimpuls, dessen Berechnung maximal $10\ \mu\text{s}$ dauert.

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

Beispiel:

- Eine Motorsteuerung benötigt alle $2000\ \mu\text{s}$ einen Steuerimpuls, dessen Berechnung maximal $10\ \mu\text{s}$ dauert.
- Entweder: Der Steuer-Computer macht noch andere Dinge.
→ Risiko der Zeitüberschreitung

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

Beispiel:

- Eine Motorsteuerung benötigt alle $2000\ \mu\text{s}$ einen Steuerimpuls, dessen Berechnung maximal $10\ \mu\text{s}$ dauert.
- Entweder: Der Steuer-Computer macht noch andere Dinge.
→ Risiko der Zeitüberschreitung
- Oder: Der Steuer-Computer macht nichts anderes.
→ Verschwendung von Rechenzeit

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

Beispiel:

- Eine Motorsteuerung benötigt alle $2000\ \mu\text{s}$ einen Steuerimpuls, dessen Berechnung maximal $10\ \mu\text{s}$ dauert.
- Entweder: Der Steuer-Computer macht noch andere Dinge.
→ Risiko der Zeitüberschreitung
- Oder: Der Steuer-Computer macht nichts anderes.
→ Verschwendung von Rechenzeit
→ Na und?

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

Große Stückzahlen

- 138 000 Toyota Prius V von Mai 2011 bis April 2012

5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

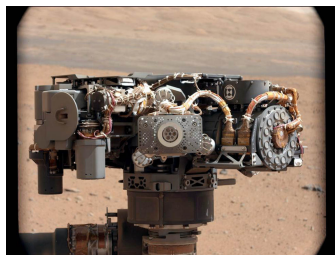
„Verschwendung von Rechenzeit – na und?“

Große Stückzahlen

- 138 000 Toyota Prius V von Mai 2011 bis April 2012

Wertvolle Ressourcen

- Fähigkeiten einer Raumsonde optimieren



5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

Große Stückzahlen

- 138 000 Toyota Prius V von Mai 2011 bis April 2012

Wertvolle Ressourcen

- Fähigkeiten einer Raumsonde optimieren
- Implantat: Platz- und Stromverbrauch minimieren



5.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

Große Stückzahlen

- 138 000 Toyota Prius V von Mai 2011 bis April 2012

Wertvolle Ressourcen

- Fähigkeiten einer Raumsonde optimieren
- Implantat: Platz- und Stromverbrauch minimieren

→ **Echtzeitprogrammierung**

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Aber wie?

5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

- ZigBee-Modul:
Funk- vs. UART-Protokoll
→ dedizierte Hardware

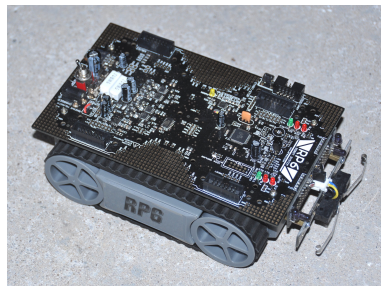


5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

- ZigBee-Modul:
Funk- vs. UART-Protokoll
→ dedizierte Hardware
- RP6:
Motorsteuerung vs. Anwender-Software
→ spezielle Software



5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

- ZigBee-Modul:
Funk- vs. UART-Protokoll
→ dedizierte Hardware
- RP6:
Motorsteuerung vs. Anwender-Software
→ spezielle Software
- Quadrocopter:
Motorsteuerung vs. Sensoren-Abfrage
vs. Funk-Fernsteuerung ...
→ spezielle Software



5.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

- ZigBee-Modul:
Funk- vs. UART-Protokoll
→ dedizierte Hardware
- RP6:
Motorsteuerung vs. Anwender-Software
→ spezielle Software
- Quadcopter:
Motorsteuerung vs. Sensoren-Abfrage
vs. Funk-Fernsteuerung ...
→ spezielle Software
- Flugzeugkabinensimulatortür:
Türsteuerung vs. Bedienung
→ Echtzeitbetriebssystem



5.2 Echtzeitprogrammierung

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

5.2 Echtzeitprogrammierung

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
Durch Zähler im Interrupt-Handler: verschiedene Taktraten
1000mal pro Sekunde: Stopwatches
5mal pro Sekunde: Blinkende Power-On-LED
1000mal pro Sekunde: Bumper, ACS, PWM zur Motorsteuerung
Geschwindigkeitsmessung durch Zählen der Ticks in 0.2 s
Anpassung der Motorkraft in ± 1 -Schritten

5.2 Echtzeitprogrammierung

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
Durch Zähler im Interrupt-Handler: verschiedene Taktraten
1000mal pro Sekunde: Stopwatches
5mal pro Sekunde: Blinkende Power-On-LED
1000mal pro Sekunde: Bumper, ACS, PWM zur Motorsteuerung
Geschwindigkeitsmessung durch Zählen der Ticks in 0.2 s
Anpassung der Motorkraft in ± 1 -Schritten
- Nebenbei: Benutzerprogramm

5.2 Echtzeitprogrammierung

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
Durch Zähler im Interrupt-Handler: verschiedene Taktraten
1000mal pro Sekunde: Stopwatches
5mal pro Sekunde: Blinkende Power-On-LED
1000mal pro Sekunde: Bumper, ACS, PWM zur Motorsteuerung
Geschwindigkeitsmessung durch Zählen der Ticks in 0.2 s
Anpassung der Motorkraft in ± 1 -Schritten
- Nebenbei: 1 Benutzerprogramm

5.3 Multitasking

- *Kooperatives Multitasking*
Prozesse geben freiwillig Rechenzeit ab
- *Präemptives Multitasking*
Das Betriebssystem unterbricht laufende Prozesse
(englisch: *to pre-empt* – jemandem zuvorkommen)

5.3 Multitasking

- *Kooperatives Multitasking*
Prozesse geben freiwillig Rechenzeit ab
- *Präemptives Multitasking*
Das Betriebssystem unterbricht laufende Prozesse
(englisch: *to pre-empt* – jemandem zuvorkommen)
- *Scheduler*
Steuerprogramm, das Prozessen Rechenzeit zuteilt
- *Kontextwechsel*
Umschalten zwischen zwei Prozessen
- *Round-Robin-Verfahren (Rundlauf)*
Zuteilung von *Zeitschlitz*en auf einer *Zeitscheibe* an die Prozesse

5.3 Multitasking

- *Kooperatives Multitasking*
Prozesse geben freiwillig Rechenzeit ab
- *Präemptives Multitasking*
Das Betriebssystem unterbricht laufende Prozesse
(englisch: *to pre-empt* – jemandem zuvorkommen)
- *Scheduler*
Steuerprogramm, das Prozessen Rechenzeit zuteilt
- *Kontextwechsel*
Umschalten zwischen zwei Prozessen
- *Round-Robin-Verfahren (Rundlauf)*
Zuteilung von *Zeitschlitz*en auf einer *Zeitscheibe* an die Prozesse
- Ausblick: Zuteilung von Rechenzeit = wichtiger Spezialfall
allgemein: Zuteilung von Ressourcen

Beispiele für Multitasking

Quadrocopter-Steuerung *MultiWii*

- Konfiguration durch bedingte Compilierung (Präprozessor)
- In der Hauptschleife wird 50mal pro Sekunde der RC-Task aufgerufen, ansonsten zyklisch einer von bis zu 5 weiteren Tasks.

RP6-Steuerung

- Konfiguration durch bedingte Compilierung (Präprozessor)
- Lichtschranken an Encoder-Scheiben lösen bei Bewegung Interrupts aus. Die Interrupt-Handler zählen Variable hoch.
- 10000mal pro Sekunde: Timer-Interrupt
verschiedene Tasks werden unterschiedlich häufig aufgerufen
- Nebenbei: 1 Benutzerprogramm