

Eingebettete Systeme

Prof. Dr. rer. nat. Peter Gerwinski

22. Mai 2025

Eingebettete Systeme

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

- 1 Einführung
- 2 Hardwarenahe Programmierung
- 3 Einführung in Unix
- 4 TCP/IP in der Praxis
- 5 Bus-Systeme
 - 5.1 Was sind Bus-Systeme?
 - 5.2 RS-232
 - 5.3 I²C (TWI)
 - 5.4 SPI
 - 5.5 PWM
 - 5.6 Sonstiges
- 6 Echtzeit

...

2 Hardwarenahe Programmierung

- :) Übungsaufgaben 1, 17. 10. 2024:
Schaltjahr ermitteln, Multiplikationstabelle,
Fibonacci-Zahlen, fehlerhaftes Programm
- :) Übungsaufgaben 5, 21. 11. 2024:
Zahlensysteme, Mikrocontroller, LED-Blinkmuster
- Übungsaufgabe 8.1, 12. 12. 2024:
Trickprogrammierung
- Übungsaufgabe 10.1, 9. 1. 2025:
Personen-Datenbank

5 Bus-Systeme

5.2 RS-232

seriell

- *TX*: 1 Leitung für Daten
- *RX*: ggf. 1 Leitung für Daten in der anderen Richtung
- *GND*: gemeinsame *Masse*
- evtl. zusätzliche Steuerleitungen

asynchron

- *keine* Taktleitung für Abgleich, wann Daten anliegen
- Stattdessen: Abgleich über Zeitvereinbarungen

→ Jeder Teilnehmer braucht eine eigene Zeitbasis.

Punkt-zu-Punkt

- nur 2 Teilnehmer vorgesehen

5.2 RS-232

Synchronisation

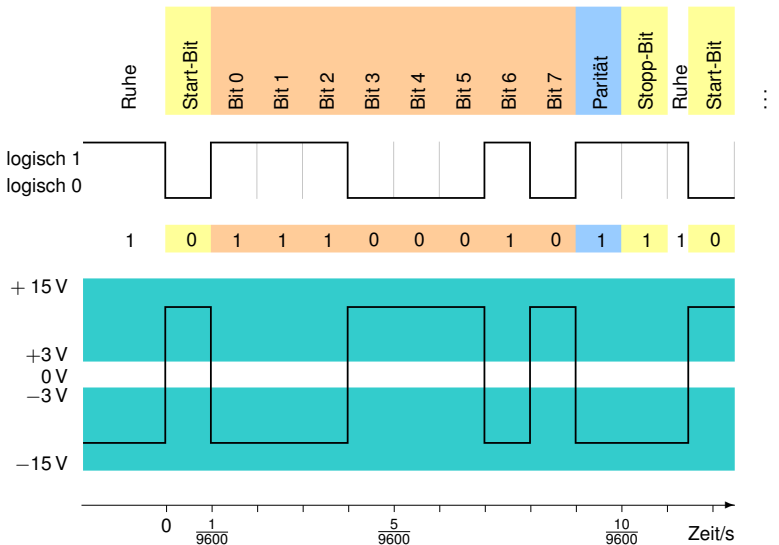
Daten

Check

9600 Baud, 8 Daten-Bits, ungerade Parität, 1 Stopp-Bit

Beispiel-Daten: ASCII „G“ = 71 = 0100 0111 binär

Übertragung der Daten von rechts (Bit 0) nach links (Bit 7)



5.3 I²C (TWI)

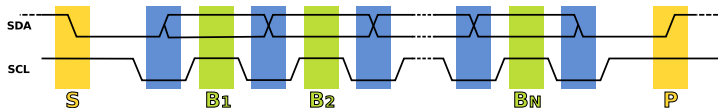
I²C = Inter-IC; TWI = Two-Wire-Interface

seriell

- *SDA*: 1 Leitung für Daten (in beiden Richtungen)
- *SCL*: Taktleitung (Clock)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung



busfähig

- *Controller* (früher: *Master*) initiiert Kommunikation und steuert Taktleitung
- *Target* (früher: *Slave*) liest Taktleitung, liest und schreibt Daten
- erstes gesendetes Byte: *Adresse* des Teilnehmers
- 2 Adressen pro Teilnehmer: Lesen/Schreiben

5 Bus-Systeme

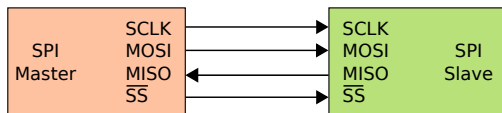
5.4 SPI

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

historische
Bezeichnungen



synchron

- Abgleich über Taktleitung

busfähig

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

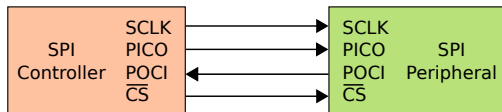
5 Bus-Systeme

5.4 SPI

Serial Peripheral Interface

seriell

- *PICO*: Peripheral In, Controller Out
- *POCI*: Peripheral Out, Controller In
- *SCLK*: Taktleitung (Clock)
- \overline{CS} : Chip Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät



synchron

- Abgleich über Taktleitung

busfähig

- *Controller* initiiert Kommunikation und steuert Taktleitung
- *Peripheral* wird über *Chip Select* ausgewählt

5 Bus-Systeme

5.4 SPI

Serial Peripheral Interface

seriell

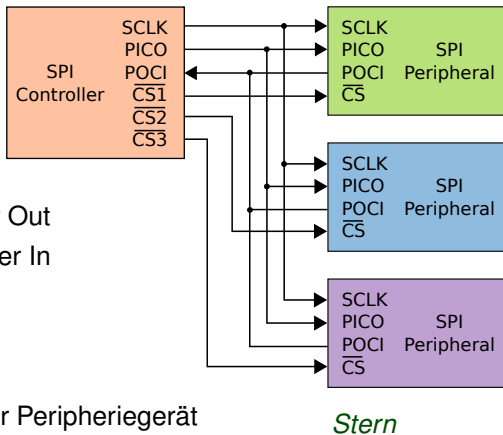
- *PICO*: Peripheral In, Controller Out
- *POCI*: Peripheral Out, Controller In
- *SCLK*: Taktleitung (Clock)
- \overline{CS} : Chip Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Controller* initiiert Kommunikation und steuert Taktleitung
- *Peripheral* wird über *Chip Select* ausgewählt



5 Bus-Systeme

5.4 SPI

Serial Peripheral Interface

seriell

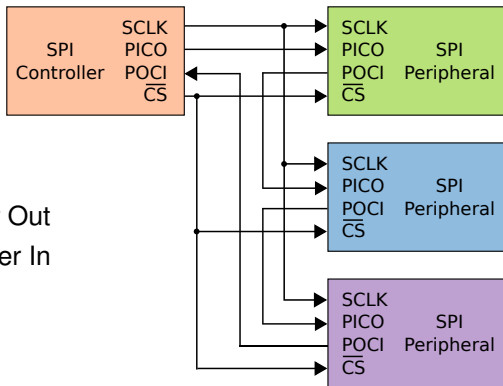
- *PICO*: Peripheral In, Controller Out
- *POCI*: Peripheral Out, Controller In
- *SCLK*: Taktleitung (Clock)
- \overline{CS} : Chip Select (invertiert)
- *GND*: gemeinsame Masse
- evtl. *VCC*: Stromversorgung für Peripheriegerät

synchron

- Abgleich über Taktleitung

busfähig

- *Controller* initiiert Kommunikation und steuert Taktleitung
- *Peripheral* wird über *Chip Select* ausgewählt



Kaskade
Daisy Chain

Peripheral gibt PICO-Input um 1 Takt verzögert an POCI aus → Controller setzt „im richtigen Moment“ \overline{CS}

5 Bus-Systeme

5.5 PWM

Pulsweitenmodulation – *pulse-width modulation*

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

5 Bus-Systeme

5.6 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

6 Echtzeit

6.1 Was ist Echtzeit?

6 Echtzeit

6.1 Was ist Echtzeit?

- Animation in Echtzeit:
schnelle Berechnung anstatt Wiedergabe einer Aufzeichnung

6 Echtzeit

6.1 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.

6 Echtzeit

6.1 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.

6 Echtzeit

6.1 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.

6 Echtzeit

6.1 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.“

6.1 Was ist Echtzeit?

„Schnell genug.“

6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“

6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*

6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*



6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*



6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

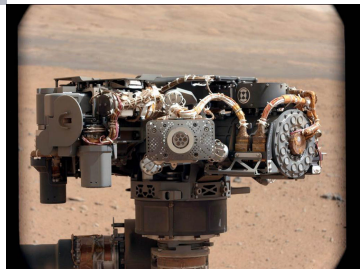
- „Ganz schlecht.“ → *harte Echtzeit*



6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

- „Ganz schlecht.“ → *harte Echtzeit*



6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

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

6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

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

6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

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



6.1 Was ist Echtzeit?

„Schnell genug.“ – „Und wenn nicht?“

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



6.1 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

6.1 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.“

6.1 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*

6.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

6.1 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.

6.1 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

6.1 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

6.1 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?

6.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

„Verschwendung von Rechenzeit – na und?“

6.1 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

6.1 Was ist Echtzeit?

Das Problem:

Harte Echtzeitanforderungen



Ressourcen optimal nutzen

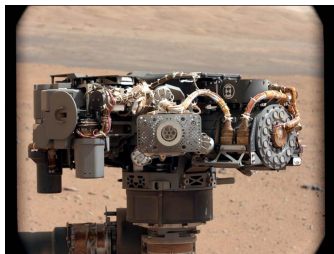
„Verschwendung von Rechenzeit – na und?“

Große Stückzahlen

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

Wertvolle Ressourcen

- Fähigkeiten einer Raumsonde optimieren



6.1 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



6.1 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

6.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Aber wie?

6.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

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

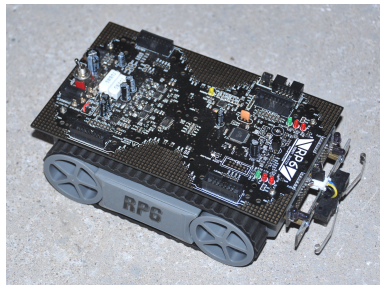


6.2 Echtzeitprogrammierung

Echtzeitanforderungen erfüllen, ohne Ressourcen zu verschwenden

Beispiele für Lösungen:

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



6.2 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



6.2 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
- Flugzeugkabinensimulatortür:
Türsteuerung vs. Bedienung
→ Echtzeitbetriebssystem



6.2 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.