2.4. Rechnersystem

2.4.1. Zeitverhalten

Anforderungen

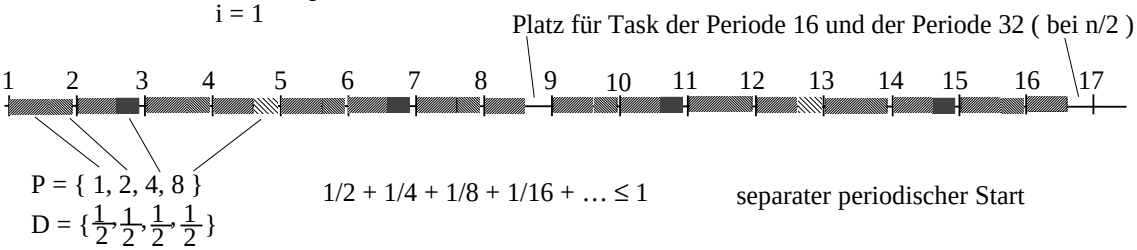
- Architektur für Echtzeitsysteme
- interne und externe Kommunikation
- Test- und Wartbarkeit des Systems

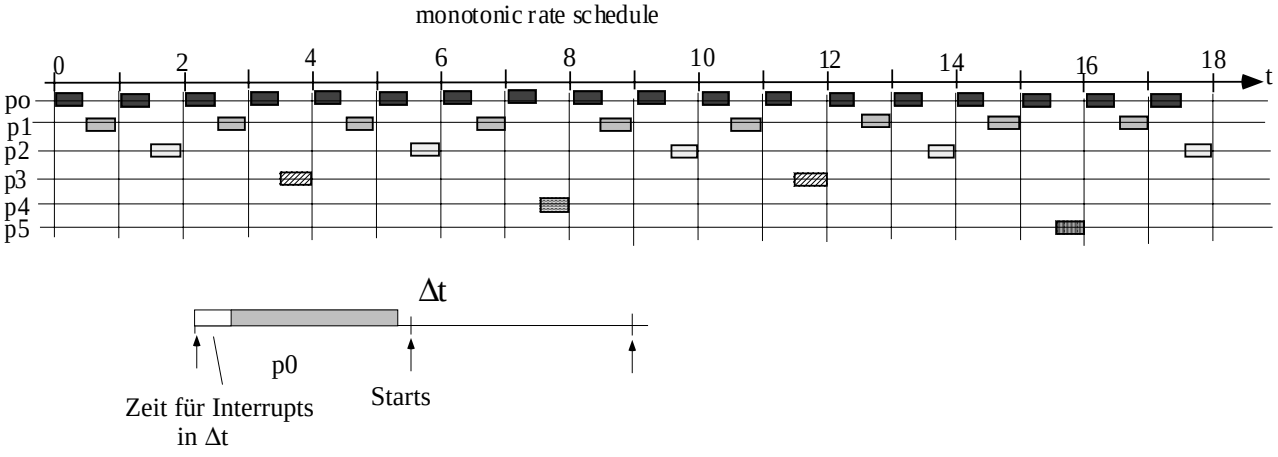
Verarbeitung in Echtzeit:

bei v = 1 m/s soll der AMR alle 10 cm reagieren können ==> Δt = 100 ms
==> Verteilung der Aufgaben in einem Zeitraster

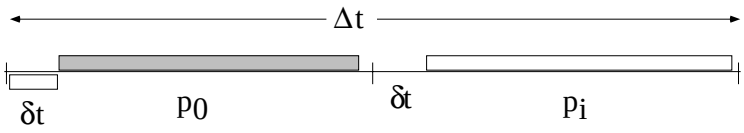
Echtzeitscheduling: n periodische Tasks der Perioden pi und Dauer ti

Bedingung $\sum_{i=1}^{n} \frac{t_i}{p_i} \leq 1$ (Liu & Leyland)





Zeitjitter



in Δt : ausgeführt p_0 , p_i und verteilt über Δt Interrupts der max. Gesamtdauer δt

tartzeiten;

p_0: n Δt … n Δt + δt; pi: $(m \cdot 2^i - 1) \cdot \Delta t$ … $(m \cdot 2^i - 1)\Delta t + \delta t)$

Zeitjitter δt = Unschärfe im Startzeitpunkt der Prozesse

für jeden Interrupt: retten des PSW, IRF := 0, DI := 1

Sprung in Interruptroutine über Trapvektor } (Hardware)

Retten der Register, EI := 1

Aufnehmen und Abspeichern von Daten

(Eventflag setzen)

Restaurieren der Register & RTI

Architektur :

hohe Parallelität natürlich gegeben:

Sensordatenvorverarbeitung --- Lokalisation --- Steuerung des Fahrzeugs --- Repräsentation der Umwelt --- Planung --- Navigation

==> Mehrrechnerarchitektur oder parallele Prozesse in einem Rechner

2.4.2. Einzelrechner

Alle Peripheriegeräte (Sensoren und Aktuatoren) hängen an dem Rechner
==> nur für kleine AMR's sinnvoll

2.4.2.1. Betriebssystem

Echtzeitbetriebssystem nötig

- hartes Echtzeitsystem: "Rechne mit dem Schlimmsten!; worst case is normal case!"
- Interrupts (wieviele in Δt?, pathologische Fälle, Garantien?)
- Einplanen aller Prozesse in Raster Δt (statische vs. dynamische Schedule)
- parallele Prozesse
  Monitore auf gemeinsam genutzte Betriebsmittel (Speicherbereiche, …) zur Kommunikation zwischen Prozessen
  Starten und Beenden eines Prozesses (durch Scheduler oder beenden selbst?)

2.4.2.2. Einzelrechner mit Peripherie an Bord

Rechner im Streichholzschachtelformat 51 x 36 mm

z. B.

- **Phytec Micro-Modul - 8051**
  Rechner OKI 80C154S, 12 MHZ Takt, 32 kByte SRAM, 256 Byte Flash
  RS232-Schnittstelle; 11-Kanal 12 Bit A/D-Wandler m. 7,5 µs Wandlungszeit
  3 V, 35 mA; Real-Time-Clock 3, 636 MHz, 2 Chip-Select-Signale
- **Phytec Micro-Modul CAN-Adapter**
  Aufsteckplatine auf Micro-Modul 8051
- **Phytec Micro-Modul -165**
  Siemens 16-Bit Controller SAB C165; 1 MByte SRAM, ! MByte Flash
  RS232-Schnittstelle, Real-Time-Clock PFC8583 mit I^2C-Bus, Chip-Select-Signale

Rechner im Scheckkartenformat 85 x 55 mm

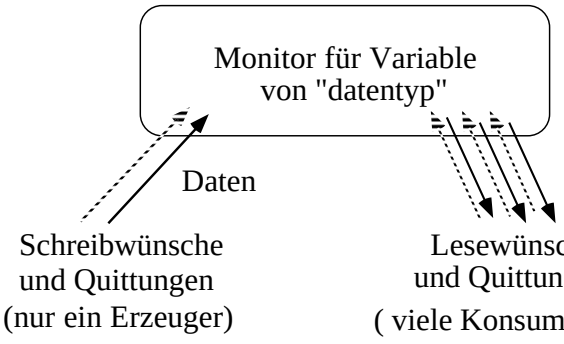
z. B.

- **Phytec Minimodul PPC505**
  Rechner 32-Bit Motorola Power-PC MPC505, 64-Bit FPU, 33 MHz, Risc-Arch.
  2x4 kByte Daten- und Programmcache, 2 MByte SRAM, 2 MByte Flash
  24-Bit Watchdog, 64-Bit Zeitbasis, 32-Bit Dekrementer, 16-Bit Interrupttimer
  interner Bus nach außen geführt; 24 Multifunktionspins für I/O; 3,3 V, 365 mA
- **Phytec Minimodul DSP-C5x**
  TMS320 C50 DSP, 10k x 16 Bit internes SRAM ; max 256 x16 Bit SRAM
  16-Bit Timer/Counter, 5 Interrupt-Eingänge, Watchdog-Timer
- **Phytec CAN-Modul- 592**
  Philips P80C 592 Can-Controller, 16 MHz, 8-Kanal A/D-Wandler 10 Bit,
  bis 160 kByte SRAM, 32 kByte EPROM, 2 PWM-Ausgänge 8 Bit, Watchdog

2.4.2.3. Blackboard

Die Synchronisation der Daten auf dem Rechner kann durch eine Blackboard erfolgen.

Ein Sensor fordert bei einem Monitor für eine von ihm gemessene Variable - z. B. eine Radaraufnahme - einen Puffer an, in den er seine Daten einschreibt; erst wenn er fertig ist, werden die Daten zum Lesen für andere Prozesse freigegeben.



Rückmeldung " Schreiben zu Ende"
setzt in den anderen Puffern "leer"

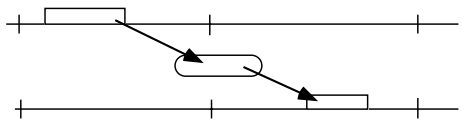
==> Lesewünsche werden aus einem Puffer befriedigt, bis ein aktuelleres zur Verfügung steht

Erzeuger muß ggf. warten bis ein Puffer leer und frei

Probleme:

große Datenblöcke; blockieren die Puffer lange Zeit

Zeitverhalten

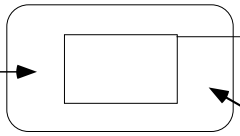


was passiert, wenn Quittung ausbleibt?

Anforderung:

will lesen/schreiben

Variable von "datentyp"

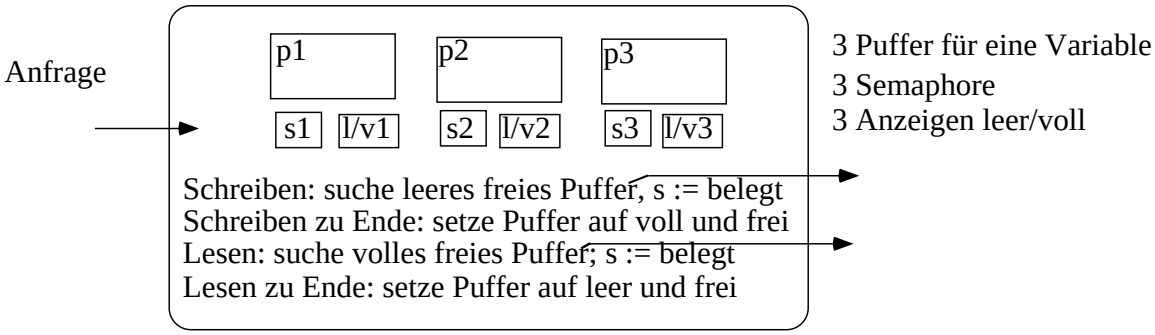


Antwort:

hier Puffer zum Lesen/Schreiben
(Referenz auf Variable von "datentyp")

Lesen/Schreiben zu Ende

Binnensicht: Verwaltung von 3 Puffern durch einen Monitor



Ein Erzeuger, ein Verbraucher, Verbraucher sieht das neueste Datum

2.4.2.4. PC auf einem AMR

normaler PC **nicht unter Windows**

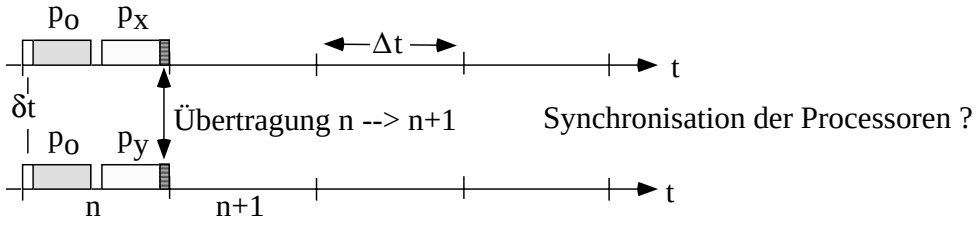
Betriebssystem:	Echtzeitbetriebssystem wie Real-time Linux oder QNX
Stromversorgung:	Wechselrichter 24 V= --> 230 V, 50 Hz, 200 W
Anschluß von Peripherie:	über USB oder anderen Bus, Ethernet o. ä.

2.4.3. Mehrrechnersystem

Die Datenverarbeitung an Bord des AMR wird durch mehrere parallel arbeitende Rechner erledigt.

2.4.3.1. Zeitverhalten

Zeitverteilung



Übergabe von Daten

wann:	am Ende einer Periode Δt mit Zeitstempel der Erzeugungszeit
wie:	vereinbartes Datenformat (Blocklänge, Datentyp, Sicherungsbits, etc.)
wo:	in Puffer; Erzeuger schreibt, Verbraucher können lesen
Ort des Puffers:	zentral: (Blackboard), Sender und Empfänger schreiben/lesen verteilt: Verteiler liest und schreibt

2.4.3.2. Verteiler für Daten (Kommunikationscontroller)

Rechner mit Masterfunktion für das Transportmedium adressiert, liest und schreibt Daten über das Transportmedium.

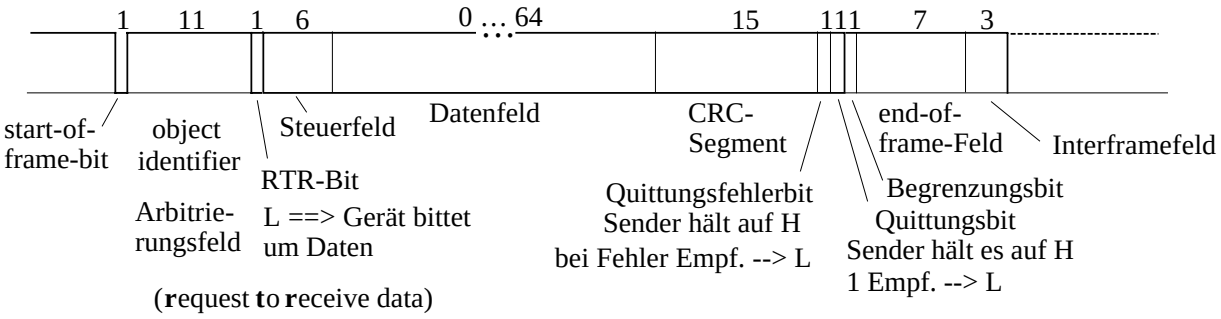
Für eine Variable stellt jedes interessierte Gerät einen Monitor zur Verfügung.

Der Kommunikationscontroller kennt den Erzeuger und die Verbraucher, liest Daten vom Erzeuger und schreibt sie in die Puffer aller Verbraucher.

Vorteil:	starre Synchronisation der Datentransporte Controller kennt alle Erzeuger und Verbraucher passgenauer Datentransfer
Nachteil:	komplexe Verwaltung
Transportmedium:	Bus (Ethernet, CAN-Bus, I^2C-Bus,VME, Firewire)

2.4.3.2.1. CAN-Bus

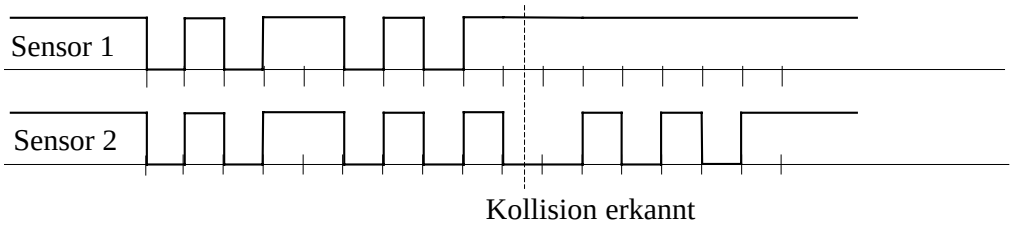
Topologie:	Bus oder Ring
max. # Teilnehmer:	32 (RS 485)
max. Leitungslänge:	1000 m
Busmedium:	2-Draht (verdrillt & geschirmt) oder LWL
Schnittstelle:	RS 485 oder ISO-DIS 11898
Übertragungsrate:	20 kBit/s - 1 MBit/s
Updatezeit:	ca. 2 ms (8 Teilnehmer, 4 Byte Nutzdaten)
Teilnehmerhierarchie:	dezentral, nachrichtenorientiert
Zugriffsverfahren:	bitweise Arbitration, CSMA/CA
Telegrammlänge:	0 - 8 Byte Nutzdaten
Fehlerabstand:	HD = 6



Adressierung: objektorientiert durch Namen (identifier)

Zugriffsverfahren: modifiziertes CSMA:

Teilnehmer darf auf freien Bus zugreifen
alle Teilnehmer hören mit
Kollision wird erkannt; Low (=0) ist dominant
Sender mit dominantem Identifier setzt sich durch



Quittung: mindestens ein Teilnehmer quittiert
durch Quittungsbit --> Low

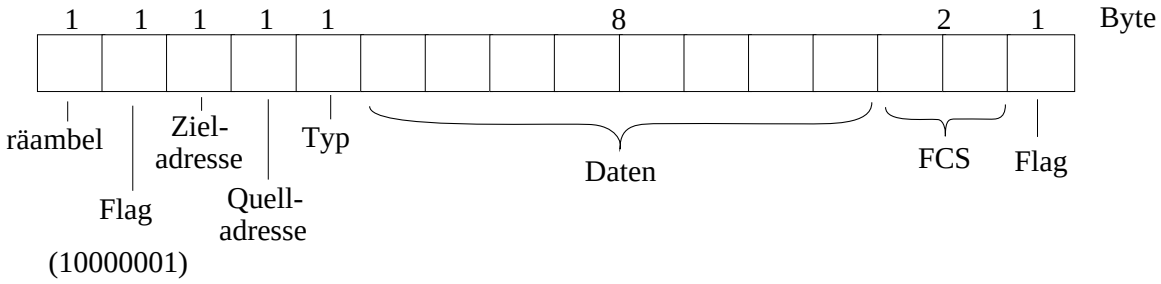
Verbindungsart: passive Teilnehmer empfangen alle Telegramme
sondern die heraus, deren Identifier bei ihnen vermerkt ist

==> Multi- und Broadcast-Verbindungen möglich

2.4.3.2.2. Ethernet

Darstellung von 0 und 1: Manchester-Code ==> Pakete ohne Takte übertragbar

Paketaufbau: vereinfachte Pakete gegenüber HTLC
Telegramme von 16 Byte Länge

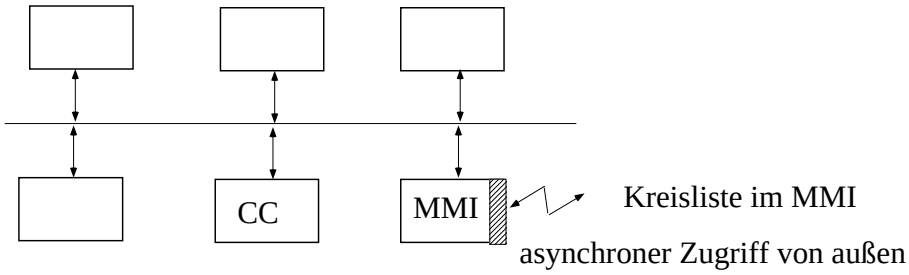


idle: 11…11
mehr als 15 Einsen

Zugriffsverfahren: CSMA/CD Kollisionserkennung in der Präambel

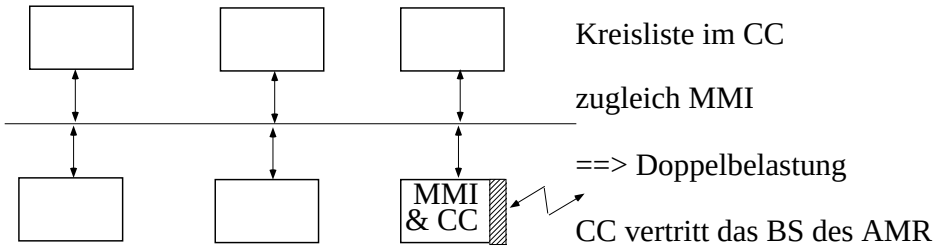
Übertragungsrate: 10 MBit/s (12,8 µs pro Paket), 625.000 Nutzdatenbyte/s

2.4.3.2.3. Kommunikation über Bus mit Controller



Zugriff auf den gesamten Datenverkehr von außen ohne Störung des Zeitverhaltens

Nachteil: zusätzlicher Transport aller Daten zur MMI durch den CC



2.4.3.2.4. Testbarkeit

Zu testen sind Funktion und Laufzeitverhalten von

- Kommunikations-Bus
- Kommunikationssoftware
- Rechner-Hardware
- Sensoren und Sensordatenverarbeitung
- Programme auf den Rechnern

Test von Kommunikationsbus und -Software offline möglich

Test von Rechnerhardware offline möglich

Sensoren, Sensordatenverarbeitung und Programme auf dem Rechner müssen online testbar sein.