

Inhaltsverzeichnis

- 1. Motivation und Zielsetzung 1
- 2. Grundlagen 9
 - 2.1 Definition grundlegender Begriffe 9
 - 2.2 Klassifizierung von Rechensystemen 14
 - 2.2.1 Definition von Prozeß-, Rechner- und Software-Struktur 14
 - 2.2.2 Klassifizierung hinsichtlich der Prozessor-Struktur 15
 - 2.2.3 Klassifizierung hinsichtlich der Task-Struktur 17
 - 2.2.4 Klassifizierung hinsichtlich der Realzeitanforderungen 17
 - 2.3 Abstraktion und Information-Hiding 18
 - 2.3.1 Modularisierung von Programmen 18
 - 2.3.2 Strukturierung von Programmen durch Prozeduren 20
 - 2.3.3 Blockstruktur 25
 - 2.4 Interrupt-Verarbeitung in Prozeßrechnern 29
 - 2.4.1 Der Interrupt-Controller in PCs 31
 - 2.4.2 Aufbau einer „Interrupt Service Routine“ (ISR) 40
- 3. Realzeit-Programmierverfahren 43
 - 3.1 Merkmale von Realzeitsystemen 43
 - 3.2 Synchrone Programmierung 50
 - 3.2.1 Vorgehensweise 50
 - 3.2.2 Beispiel: Bahnsteuerung eines Regalförderzeugs 50
 - 3.2.3 Beispiel: Steuerung von zwei Läufers 58
 - 3.2.4 Vor- und Nachteile der synchronen Programmierung 66
 - 3.2.5 Übungen 67
 - 3.3 Asynchrone Programmierung 71
 - 3.3.1 Das Modell der „Quasiparallelität“ 71
 - 3.3.2 Das Modell der „Parallelität“ 72
 - 3.3.3 Beispiel: Bahnsteuerung eines Regalförderzeugs 75
 - 3.3.4 Beispiel: Steuerung von zwei Läufers 79
 - 3.3.5 „Task Control Block“ (TCB) 83
 - 3.3.6 Übung 85
 - 3.4 Gegenüberstellung der synchronen und asynchronen Programmierung87
- 4. Realzeit-Programmiersprachen 89
 - 4.1 Die Realzeit-Programmiersprache Modula-2 90
 - 4.1.1 Das Modulkonzept 90
 - 4.1.2 Konzepte der asynchronen Programmierung 96

VI Inhaltsverzeichnis

4.1.3	Co-Routinen	97
4.1.4	Konkurrierende Prozesse	102
4.1.5	Signal	102
4.1.6	Monitor	103
4.2	Die Realzeit-Programmiersprache Pearl	104
4.2.1	Sprachmittel zur Strukturierung eines Pearl-Programms	104
4.2.2	Die algorithmischen Sprachmittel	107
4.2.3	Ein-/Ausgabe	108
4.2.4	Tasking-Anweisungen	113
4.2.5	Bearbeitung von Interrupts	121
4.2.6	Synchronisierung von Tasks	122
4.2.7	Beispiel: Dosieranlage	123
4.2.8	Beispiel: Erfassung der Lufttemperatur	127
4.2.9	Beispiel: Bahnübergang	129
4.2.10	Beispiel: Automatisierung einer Eisenbahnstrecke	136
4.2.11	Beurteilung des Tasking in Pearl	144
4.2.12	Übung	145
4.3	Die Realzeit-Programmiersprache Ada	147
4.3.1	Das Konzept der Parallel-Programmierung in Ada	147
4.3.2	Synchronisierung mit Hilfe des Rendezvous-Konzepts	149
5.	Synchronisierung von Rechenprozessen	151
5.1	Problematik	151
5.2	Herleitung der elementaren Anforderungen für die Synchronisierung von Tasks	151
5.3	Herleitung der zeitlichen Zusammenhänge bei der Synchronisierung	161
5.4	Einfache Methoden zur Realisierung des wechselseitigen Ausschlusses	162
5.4.1	Interrupt-Sperre	162
5.4.2	Unteilbare „Exchange-Operation“	164
5.5	Praktische Regeln für die Realisierung von Synchronisierungen	165
6.	Petri-Netze zur Modellierung von Synchronisierungen	167
6.1	Netze aus Bedingungen und Ereignissen	167
6.2	Netze aus Stellen und Transitionen	169
6.3	Aufbau und Verhalten von Petri-Netzen	174
6.4	Konflikt und Kontakt	176
6.5	Beispiel für Systementwurf mit Petri-Netzen	178
7.	Semaphore	181
7.1	Die Entstehung von Semaphoren	181
7.2	Architektur von Semaphoren	182
7.3	Definition des Objekts „Semaphore“	183
7.4	Abbildung von Petri-Netzen auf Semaphoren	186

7.5	Beispiele	190
7.5.1	Wechselseitiger Ausschluß	190
7.5.2	Automatisierung einer Eisenbahnstrecke	193
7.5.3	Erzeuger-Verbraucher-Problem	193
7.5.4	Kanal mit beschränkter Kapazität (Bounded Buffer)	193
7.6	Anfordern mehrerer Betriebsmittel (Konjunktion)	194
7.7	Anfordern eines Betriebsmittels aus mehreren (Disjunktion)	210
7.8	Beurteilung der Eigenschaften von Semaphoren	213
7.9	Übungen	214
7.9.1	Gemeinsamer Meßwert (Sema 1)	214
7.9.2	Einfache Folge (Sema 2)	215
7.9.3	Kreisverkehr (Sema 3)	216
7.9.4	Dijkstra's Dining Philosophers (Sema 4)	218
7.9.5	Komplexe Folge (Sema 5)	219
7.9.6	Erzeuger-Verbraucher-Problem für reale Verhältnisse (Sema 6)	220
8.	Monitor	221
8.1	Definition des Objektes Monitor	221
8.2	Ergänzung von „Monitor“ mit „Signal“	223
8.2.1	Gründe für die Einführung von Objekten vom Typ „Signal“ in einen Monitor	223
8.2.2	Semantik bei der Zusammensetzung von Monitor und Signal	224
8.2.3	Die Architektur für die Zusammensetzung von Monitor und Signalen	225
8.3	Beispiele	227
8.3.1	Realisierung von Semaphoren mit Hilfe von Monitor und Signal	227
8.3.2	Optimierung des Bounded-Buffer-Problems (Dijkstra's sleeping barber)	228
8.4	Anfordern mehrerer Betriebsmittel (Konjunktion)	231
8.5	Anfordern eines Betriebsmittels aus mehreren (Disjunktion)	233
8.6	Zusammenfassende Beurteilung des Monitor-Konzepts	235
8.7	Übungen	235
9.	Rendezvous	237
9.1	Gründe für die Einführung des Rendezvous-Konzepts	237
9.2	Das einfache Rendezvous	238
9.3	„Select“-Anweisung	245
9.3.1	Aufbau und Semantik der Select-Anweisung	245
9.3.2	Beispiel: Geschützter Meßwert	246
9.4	„Guarded-Select“-Anweisung	248
9.4.1	Aufbau und Semantik von „Guarded-Select“	248
9.4.2	Beispiel: Bounded Buffer	249
9.5	Anfordern mehrerer Betriebsmittel (Konjunktion)	250
9.6	Anfordern eines Betriebsmittels aus mehreren (Disjunktion)	251
9.7	Zusammenfassende Beurteilung von „Rendezvous“	252
9.8	Übungen	252

10. Typische Synchronisierungsaufgaben	253
10.1 Synchronisierung eines Betriebsmittels (Wechselseitiger Ausschluß)	253
10.1.1 Lösung mit Semaphoren	253
10.1.2 Lösung mit Monitor	256
10.1.3 Lösung mit Rendezvous	259
10.2 Synchronisierung eines n-fach vorhandenen Betriebsmittels	261
10.2.1 Lösung mit Semaphoren	261
10.2.2 Lösung mit Monitor	264
10.2.3 Lösung mit Rendezvous	266
10.3 Einräumen eines Vorrangs an n-ter Stelle für eine Task bei der Zuteilung eines Betriebsmittels	268
10.3.1 Lösung mit Semaphoren	268
10.3.2 Lösung mit Monitor	273
10.3.3 Lösung mit Rendezvous	275
10.4 Höchste Priorisierung der Betriebsmittel-Anforderung einer Task (Readers Writers Problem)	276
10.4.1 Lösung mit Semaphoren	277
10.4.2 Lösung mit Monitor	278
10.4.3 Lösung mit Rendezvous	279
10.5 Feste Reihenfolge der Tasks (Erzeuger-Verbraucher-Problem)	281
10.5.1 Lösung mit Semaphoren	281
10.5.2 Lösung mit Monitor	285
10.5.3 Lösung mit Rendezvous	286
10.6 Überholvorgänge (Bounded Buffer)	287
10.6.1 Lösung mit Semaphoren	287
10.6.2 Lösung mit Monitor	290
10.6.3 Lösung mit Rendezvous	290
 11. Problemorientierte Modellierung der Software für Realzeitsysteme	 291
11.1 Stellung des Software-Entwurfs im Entstehungsgang eines Prozeß- automatisierungssystems	292
11.2 Modularisierung des Automatisierungssystems als Grundlage des Software-Entwurfs	293
11.3 Festlegung der sequentiellen Vorgänge in einem Modul	297
11.4 Modellierung eines sequentiellen Vorgangs mit Hilfe von endlichen Automaten	297
11.4.1 Definition und Implementierung des endlichen Automaten	299
11.4.2 Zusammenfassung einer Kette zu einem Zustand	302
11.4.3 Beispiel: Geldautomat	308
11.5 Modellierung der Synchronisierungen zwischen Tasks mit Hilfe von Petri-Netzen und Rendezvous	314

11.6 Ein umfassendes Beispiel: Produktionsstraße	314
11.6.1 Modularisierung des Systems	316
11.6.2 Anzahl der sequentiellen Vorgänge in jedem Modul	318
11.6.3 Modellierung der sequentiellen Vorgänge mit Hilfe von endlichen Automaten	319
11.6.4 Modellierung eines Teils der Synchronisierung mittels Petri-Netzen	320
11.6.5 Modellierung eines Teils der Synchronisierung mittels Rendezvous	330
11.6.6 Schematische Vorgehensweise bei der Umsetzung des Software- Entwurfs in ein synchrones Programm (für SPS nach DIN 61131-3)	331
11.6.7 Schematische Vorgehensweise bei der Umsetzung des Software- Entwurfs in ein asynchrones Programm	336
Anhang 1: Begriffsdefinitionen laut DIN	341
Anhang 2: Verwendete DIN-Vorschriften	346
Literaturverzeichnis	347
Sachverzeichnis	351