Institut für Nachrichtenvermittlung und Datenverarbeitung

Universität Stuttgart

Prof. Dr.-Ing. A. Lotze

24. Bericht über verkehrstheoretische Arbeiten

Entwicklung der Steuersoftware für eine rechnergesteuerte Vermittlungsstelle

von

Helmut Weisschuh

Institute of Switching and Data Technics

University of Stuttgart

Prof. Dr.-Ing. A. Lotze

24th Report on Studies in Congestion Theory

Development of the Control Software for a Stored Program Controlled PCM Switching System

by

Helmut Weisschuh

1977

ABSTRACT

This research report comprises three topics, relating to a stored program controlled PCM-switching system, which is under development at the Institute of Switching and Data Technics, University of Stuttgart.

The first part of the report deals with the overall software structure of the central processor. Hereby, its hardware features are not taken into account. Then, for the implementation, this "ideal" software structure is adapted to a general purpose real-time computer. After a description of the complete software organization and the program operation sequences, a storage organization is developed, which grants an optimal protection for the different storage areas in the main memory of the central processor.

The second part of the report deals with simulation techniques, which can be used during the design phase, the development phase for the system and for the testing of the control software of the central processor, resp. In particular, an environment simulator is described, which has among other testing facilities, the facility to determine the program run times (for traffic investigations) of the switching programs. Furthermore some results of run time measurements are presented.

The third part deals with traffic investigations on the central processor of the considered switching system. A model of a queuing system is introduced and investigated analytically. In this model, batches of informations (requests) arrive at a storage (queue) in equidistantly distributed sampling (clock) instants. Waiting requests are served separately by a single server, which corresponds to the central processing unit. The distribution function of its service times is assumed to be of Erlangian type of k-th order. This distribution function was verified by program run time measurements with the environment simulator. At each sampling instant the server is occupied for a constant overhead phase. Some diagrams show the remarkable influence of various parameters, such as the parameter k and the overhead t_o, on the mean waiting times of requests.

In the following a brief review of the various chapters will be given.

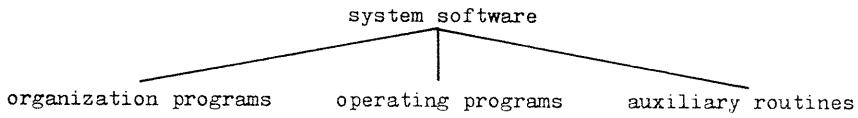
CHAPTER 1: Introduction

After some general aspects on stored program controlled switching systems, a brief description is given on the structure of the experimental PCM-switching system, which is under development at the Institute of Switching and Data Technics, University of Stuttgart. An outstanding characteristic feature of this system is the extensive preprocessing of control information by peripheral control units. This concept saves expensive central processor capacity and makes the SPC-system less sensitive with respect to overload.

A short survey of the report concludes Chapter 1.

CHAPTER 2: System Software Structure of the Central Processor

The system software of the central processor can be subdivided into three categories of programs:



The organization programs are responsible for scheduling and controlling of the operating programs, which perform the actual tasks in the central processor. The auxiliary routines execute frequently used functions for the organization and operating programs. In order to meet the real-time constraints, the system software is subdivided into seven priority levels (Fig. 2.2, page 22).

Each priority level has its own organization program. This organization program schedules and supervises the execution of the operating programs belonging to this priority level. The operating programs are assigned to the various priority levels and can be executed only in the assigned level under the supervision of the corresponding organization program. They perform the different tasks according to the information from the peripheral control units. All the organization and operating programs have access to a pool of auxiliary routines, which are common to all these programs.

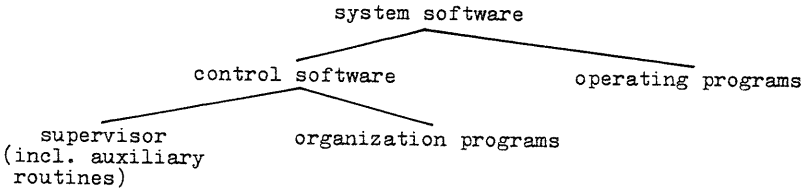
The organization programs are initiated by the interrupt circuitry. The interrupt sources (such as failure detection circuits or clock generator) generate signals to the interrupt circuitry. This unit transfers control to the organization program of the actual interrupt

level. After finishing work in this priority level, control is given back to the organization program of the interrupted priority class by the interrupt circuitry.

In order to implement this system software on a general purpose real-time computer, the structure had to be adapted to the given hardware facilities of this computer (Fig. 2.4, page 28).

Between the interrupt circuitry and the organization programs, a software level, called supervisor, was introduced. This supervisor schedules and supervises the execution of the organization programs. Besides these control functions, it also performs the tasks of the channel programs and of the input/output programs for the peripheral control units. The auxiliary routines were included into the supervisor, too, and they can be accessed by the different programs via so-called "Supervisor Calls".

By introducing the supervisor, the system software has the following arrangement:



In the following two chapters 3 and 4 the control software of the central processor is described.

CHAPTER 3: Supervisor

The supervisor corresponds to the nucleus in an operating system of a general purpose computer. It comprises all routines for the processing of interrupts, for the scheduling and controlling of user programs (tasks), for the different input/output operations and for other utility functions.

The different routines of the supervisor are activated by interrupt signals and they are running in the privileged mode or in the alarm mode of the central processing unit (CPU), whereas the user programs are running as tasks in the normal mode of the CPU.

After a short introduction (section 3.1) the different functions of the supervisor are described in detail.

EXECUTION of SUPERVISOR CALLS (section 3.2)

The functions of the supervisor (e.g. output via teleprinter) can be accessed by the different user programs via Supervisor Calls (SVC). Each SVC consists of an instruction sequence, which specifies the operations to be done by the supervisor. The first instruction is an instruction for changing the execution mode of the CPU. It is called system interrupt instruction. When executing this instruction a special routine for the preprocessing of Supervisor Calls is activated. This routine checks the parameters of the SVC and switches to the corresponding execution routine.

Two types of Supervisor Calls are distinguished, namely SVC's which can be executed immediately and SVC's which are served later via waiting lists (queues). These queues arise while waiting for events (e.g. acknowledgement signals from input/output units, signalling the end of operation).

SCHEDULING and CONTROLLING (section 3.3)

The supervisor provides time-shared processing of up to 14 independent programs. These programs are scheduled as tasks by the supervisor. To each task, a so-called task control block is assigned, which contains the necessary information about this task (e.g. contents of the program counter and of other CPU registers in case of an interrupt).

The scheduling of the tasks is governed by priorities. Each task is assigned a priority number. The task having the highest priority is always processed first. When a task requires input/output operations it has to wait until completion of these operations. In the meantime, other tasks may be continued.

A task can assume different states (e.g. it is waiting for input/output operations) and these states can be changed by special Supervisor Calls.

Furthermore special Supervisor Calls for the synchronization of tasks and for the task communication are implemented.

INPUT/OUTPUT for STANDARD DEVICES (section 3.4)

All input/output devices of the central processor, which do not interact with the switching periphery are called "standard devices" (e.g. teleprinter, printer unit, magnetic disc, etc.).

A task which must execute an input/output operation initializes this operation by a Supervisor Call. The supervisor has an input/output routine for each device (e.g. teleprinter, disc). This routine generates the corresponding external machine instructions required for the operation, and then after the execution of these instructions, it processes the acknowledgement signals from the peripheral device.

The input/output calls for the peripheral devices are executed in the sequence of their arrivals. If they can not be executed immediately, they are entered into a waiting list for each device.

The input/output routine of the magnetic disc also includes the file management.

INPUT/OUTPUT for PERIPHERAL SWITCHING UNITS (section 3.5)

The input/output to and from the peripheral switching units is clock driven. Every 10 ms a clock-signal from a timer activates a special input/output routine. This routine, running in the alarm mode of the CPU, transfers messages from the peripheral switching units into an internal queue within the supervisor and transfers instructions from an internal queue to the peripheral switching units. The transfer of information between the input/output queues and the programs is done via Supervisor Calls.

Besides this regular input/output functions, a facility is implemented which enables a program (task) to read and write information directly from and into the input/output queues of the supervisor. By this means the input/output operation to and from the peripheral switching units can be simulated by a program. (The simulation program is described in chapter 6.)

Furthermore the input/output routine cyclically activates an organization program which performs timing functions for the switching system.

PROTECTION (section 3.6)

Abnormal conditions in the execution of instructions generate interrupts. These conditions can be caused by hardware failures(e.g. parity error in the main memory) or by software failures (e.g. an illegal operation had to be carried out). The interrupt processing routine determines the reason for the failure and, in case of a software failure, the task causing this failure is released. In both cases a message is sent to the organization program which schedules the diagnostic programs for diagnosting this failure.

Furthermore some instructions which are not implemented in hardware have to be executed by this routine of the supervisor.

UTILITIES (section 3.7)

The utility routines provide special functions, which could not be assigned to the other function blocks of the supervisor. Some of these functions are e.g. testing conditions on the operator's panel, activating a buzzer, checking the state of a task etc.

SYSTEM INITIALIZATION (section 3.8)

After switching on the central processor, the system software has to be loaded into the main memory of the central processor. This function is carried out by the system initialization routine. The routine is activated by the bootstrap facility of the central processor.

INTERRUPT ENTRY ROUTINE (section 3.9)

The interrupt entry routine saves the contents of the registers of the CPU into the task control block of the interrupted task. Then it activates the corresponding routine, which is assigned to the interrupt cause. (This interrupt handling routines are contents of the sections 3.2 to 3.7.)

INTERRUPT EXIT ROUTINE (section 3.10)

After the work in the supervisor is finished, the interrupt exit routine loads the registers of the CPU with the contents of the task control block of that task having the highest priority and hands control over to it.

LOADER (section 3.11)

The function of the loader is to allocate main memory space for a program and to load this program into the main memory. When loading the program, the loader has to relocate the program into the allocated memory space. Whenever a program has finished, the loader has to release the memory space occupied by this program.

The implemented loader can load programs either from punched cards or from magnetic disc storage.

CONSOLE ROUTINE (section 3.12)

This routine enables communication between the operator and the supervisor. The operator may load and start programs via the console. Furthermore, for reason of software testing he may check the state of different tasks and request an output.

CHAPTER 4: Organization and Operating Programs

The organization programs are responsible for the scheduling and controlling of the operating programs, which perform the actual tasks in the central processor. The organization programs themselves are scheduled by the supervisor, which schedules the different programs as tasks.

For the program-to-task assignment, two solutions are discussed (section 4.2). One solution assigns different task numbers to an organization program and its operating programs. The other solution assigns the same task number to an organization program and its operating programs. The latter one was implemented.

Between organization programs an interchange of information is necessary (e.g. to activate an execution program which is running under the supervision of another organization program). This communication between organization programs is described in section 4.3. Section 4.4 deals with the control of operating programs by the organization programs and the flow of control information within organization programs.

The following sections explain some characteristic features of special organization programs.

CHAPTER 5: Main Memory Organization

The main memory of the central processor contains the programs of the system software and the required data. The data can be subdivided into fixed (rarely to be modified) and variable (often to be modified) ones. Fixed data are e.g. basic subscriber data such as information on his terminal location and data concerning the subscriber class. Variable data are e.g. data on the state of individual connections and data giving the momentary idle/busy state of the individual channels and links in the system for "path selection in the memory".

The aim of the main memory organization is to separate the variable data from the programs as well as from the fixed data and to store them in a hardware protected storage area. This is done in order to protect the programs and fixed data against falsification by erroneous programs.

The hardware facilities of the real-time computer used for system software implementation allow only restricted hardware protection. Therefore, a special main memory organization is developed (Fig. 5.3, page 131) which uses the facility of relative addressing by means of two different base address registers.

Program instructions as well as fixed data are addressed by the first base address register. The variable data are separated from the programs and from the fixed data and are addressed by the second base address register. In this way a quasi-protected memory area is created.

CHAPTER 6: Processor Environment Simulation

Simulation techniques are a tool often used in the design and test phase for stored program controlled switching systems. For the development of the experimental PCM-switching system, three simulation programs were written.

The first simulation program (section 6.2) simulates the control of a peripheral switching control unit. By means of this program the control function of this unit can be checked. Even a check with a faulty environment is possible because faulty messages can be used as input data for the simulation.

Furthermore, this program can be used to replace the hardware of the peripheral control unit and to perform the control functions within the central processor. In this case, however, the units controlled by the peripheral control units must be connected directly to the central processor.

The second program (section 6.3.2) represents a so-called interface simulator. By means of this interface simulator, messages originating in the peripheral control units of the real system can be generated by test staff and entered into the input queue of the supervisor. Vice versa, instructions can be read out of the output queue. Furthermore, the switching programs can be activated in order to process the messages waiting in the input queue.

This interface simulator can be used for system software testing. The input can be done via teleprinter or via a punched card reader. The test pattern is assembled by the test staff. The output of the switching programs (instructions in the output queue of the supervisor) can be checked.

The manual assembly of the test patterns is very cumbersome, therefore an automatic environment simulator was developed as a third program (section 6.3.3). This environment simulator does not only simulate the switching hardware but also subscriber traffic. The subscriber traffic takes into account various subscriber behavior such as calling subscriber goes on-hook without dialling, calling subscriber goes on-hook after dialling but before the called subscriber answers, etc.

The simulator generates messages for the switching software (e.g. calling subscriber goes off-hook) and feeds these messages into the input queue of the supervisor. (From there the messages are processed by the switching programs.) Instructions from the switching programs to the peripheral switching units are taken from the supervisor's output queue and checked whether they are legal, i.e. whether the instruction is a correct reaction of the switching software to the message sent before (e.g. a digit receiver can be only released if the corresponding subscriber was switched to this digit receiver).

The generation of messages is based upon events which describe the subscriber behavior. The time between two succeeding events is determined by means of random numbers and given probability distribution functions for the interarrival times (event-by-event simulation technique).

The simulator runs on the same central processing unit as the real switching software. A special facility of the simulator is the measurement of switching program run times. Some results of such run time measurements are shown on pages 155-159.

CHAPTER 6: Investigation on the Traffic Behavior of the Central Processor

In order to achieve an analytic description of the traffic behavior of the central processor a suitable queuing model was introduced. This model of the central processor is shown in Fig.1.

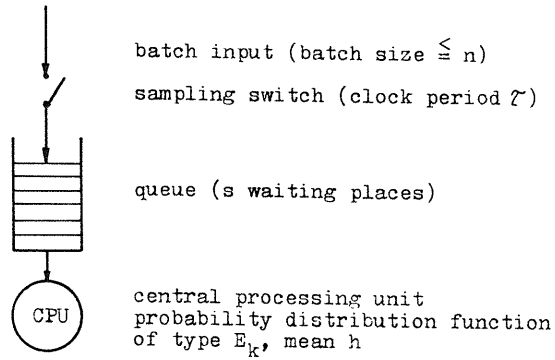


Fig.1: Queuing model of the central processor

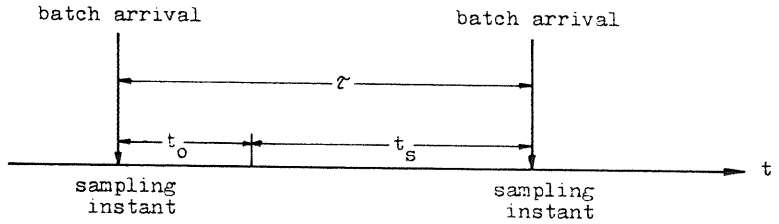


Fig.2: Timing diagram of the central processor

The model consists of a single server (CPU) in front of which there is a queue of finite capacity (s waiting places). The input process from the switching periphery is modeled by batch arrivals of informations (requests) at equidistant sampling instants (clock period τ, c.f. Fig.2). The batch size is an independent random variable G. The probabilities $P\{G=i\} = g(i)$ with $i=0,1,\ldots,n$ for the batch size are assumed to be independent of the sequence of sampling instants and of the service times of the requests.

The service times of the requests are distributed according to an Erlangian probability distribution function of k-th order (E_k). This probability distribution function was verified by program run time measurements (c.f. chapter 6). The queue discipline is service in order of arrival between batches, furthermore first-in/first-out within the members of a batch.

The service process and the arrival of requests are shown in Fig.2. At each sampling instant the server is occupied first for a constant overhead phase t_o and then for the above described service time t_s.

The analytical solution was done by means of the Imbedded Markov Chain method. The inspectation instants were chosen immediately before the sampling instants (Fig.2). For the case of statistical equilibrium, the set of equations for the probabilities of state was derived. This set of equations was solved iteratively by means of the relaxation method.

Based on this probabilities of state immediately before the sampling instants, the probabilities of state at an instant t_d after the sampling instant and the following characteristic traffic values were derived:

- probability of loss and mean loss rate of requests
- load of the central processor
- probability of waiting of requests
- mean queue length
- mean waiting time for requests

Numerical evaluations (Diagrams 7.14 - 7.18) in the report show the influence of various parameters on the mean waiting time of requests and an example for the probabilities of waiting and loss.

CHAPTER 7: Conclusion; c.f. Abstract Chapter 1

APPENDIX

Appendix A1: General Terms

This section includes some definitions of general terms, which are used throughout the report. After a short definition of the term "task", the organization of various tables and lists is described, where the data items are bits, words and blocks of words. Furthermore, the format of the control information which is interchanged between the central processor and the peripheral switching units is depicted.

Appendix A2: List of Implemented Supervisor Calls

In this section all implemented Supervisor Calls are listed.

Appendix A3: The General Purpose Real-Time Computer Siemens 306

The system software of the experimental PCM-switching system was implemented on a general purpose real-time computer, the computer Siemens 306. In this section the hardware configuration of this computer is shown and some characteristic features of this computer are described.

INHALTSVERZEICHNIS

Seite

Schrifttumsverzeichnis 6

Abkürzungen 11

1. Einleitung 16

1.1 Allgemeines über rechnergesteuerte Vermittlungsstellen 16

1.2 PCM-Projekt des Instituts für Nachrichtenvermittlung und Datenverarbeitung 17

1.3 Übersicht über die Arbeit 19

2. Struktur der Systemsoftware für den Steuerrechner 21

2.1 Prinzipielle Struktur der Systemsoftware 21

2.1.1 Allgemeines 21
2.1.2 Struktur und Prioritätsklassen 21
2.1.3 Arbeitsprogramme 23

2.2 Struktur des Unterbrechungssystems des Rechners Siemens 306 25

2.3 Ergänzung der Systemsoftware zur Anpassung an die Hardware des S306 27

3. Überwacher 30

3.1 Struktur des Überwachers 30

3.2 Bearbeitung von Überwacheraufrufen 33

3.2.1 Allgemeines 33
3.2.2 Struktur eines Überwacheraufrufes 34
3.2.3 Vorverarbeitung von Überwacheraufrufen 37
3.2.4 Ausführung von Überwacheraufrufen 39

3.2.4.1 Kurze Aufrufe 39
3.2.4.2 Lange Aufrufe 40

3.2.5 Aufruf der Überwacherfunktionen im Überwacher 42

3.3 Ablauforganisation 42

3.3.1 Allgemeines 42
3.3.2 Das Zustandsmodell 44
3.3.3 Taskverwaltung 46

3.3.3.1 Allgemeines 46
3.3.3.2 Taskwarteschlange und Taskkontrollblock 46
3.3.3.3 Funktionen der Taskverwaltung 48

3.3.4 Synchronisation 52

3.3.5 Kommunikation 55

3.4 Ein-Ausgabe Standardperipherie 58

3.4.1 Allgemeines 58
3.4.2 Ein-Ausgabebearbeitung 59
3.4.3 Unterbrechungsbearbeitung für Kanalanforderungen 61

3.4.3.1 Allgemeines 61
3.4.3.2 Unterbrechungsvorverarbeitung 62
3.4.3.3 Anzeigenbearbeitungsroutine 63

3.4.4 Warteschlangenorganisation 65

3.5 Ein-Ausgabe Vermittlungsperipherie 66

3.5.1 Allgemeines 66
3.5.2 Ein-Ausgabebearbeitung im Einzelnen 69

3.5.2.1 Taktbearbeitungsroutine 70
3.5.2.2 Aktivierung eines Organisationsprogrammes 72
3.5.2.3 Bearbeitungsroutinen für Überwacheraufrufe 76

3.5.3 Warteschlangenorganisation 78

3.5.3.1 Eingabewarteschlange 78
3.5.3.2 Ausgabewarteschlange 81

3.5.4 Funktionen zur Simulation der Ein-Ausgabe zur Vermittlungsperipherie 86

3.5.4.1 Allgemeines 86
3.5.4.2 Funktionen zur Simulation 86

3.6 Sicherungsfunktionen 89

3.6.1 Allgemeines 89
3.6.2 Unterbrechungsvorverarbeitung 89
3.6.3 Bearbeitungsroutinen zur Unterbrechungsbearbeitung 91

3.6.3.1 Befehlsausführung per Software 91
3.6.3.2 Softwarefehler 91
3.6.3.3 Unterbrechungstaste 92

3.6.4 Fehlerbearbeitungsroutine 93

3.7 Weitere Dienstfunktionen 94

3.8 Ureingabe 95

3.9 Unterbrechungseingangsroutine 96

3.10 Unterbrechungsausgangsroutine 97

3.10.1 Allgemeines 97
3.10.2 Aktivieren von Organisationsprogrammen 99

3.10.2.1 Allgemeines 100
3.10.2.2 Retten von Registerinhalten 100
3.10.2.3 Umschalten des Prozessors 100

3.11 Systemtask "Lader" 101

3.11.1 Allgemeines 101
3.11.2 Ladefunktion 101
3.11.3 Arbeitsspeicherverwaltung 103

3.12 Systemtask "Bedienteil" 105

4. Organisations- und Arbeitsprogramme 107

4.1 Allgemeines 107

4.2 Tasknummernzuteilung und Prioritäten 107

4.3 Informationsaustausch zwischen Organisationsprogrammen 110

4.4 Steuerungsablauf in einem Organisationsprogramm 117

4.5 Besonderheiten einzelner Organisationsprogramme 121

4.5.1 Organisationsprogramm ORG3 121
4.5.2 Organisationsprogramm ORG4 122
4.5.3 Organisationsprogramm ORG6 123
4.5.4 Organisationsprogramm ORG7 124

5. Speicherorganisation 127

5.1 Allgemeines 127

5.2 Randbedingungen durch den Rechner Siemens 306 128
5.3 Realisierte Speicherorganisation 130
6. Simulation der Vermittlungsperipherie 133
6.1 Allgemeines 133
6.2 Simulation einer Konzentratoranschlußschaltung 133
6.2.1 Möglichkeiten der Simulation 133
6.2.2 Simulationsprogramm 136
6.2.3 Betriebsmäßiger Ersatz der KAS durch das Simulationsprogramm 138
6.3 Testmöglichkeiten für die Systemsoftware durch Simulation 139
6.3.1 Allgemeines 139
6.3.2 Schnittstellensimulation 142
6.3.3 Umweltsimulation 147
6.4 Bestimmung der Bearbeitungszeiten von Meldungen durch Vermittlungsprogramme 152
6.4.1 Allgemeines 152
6.4.2 Meßmethode 152
6.4.3 Meßergebnisse 154
7. Verkehrstheoretische Untersuchung des Vermittlungsrechners 160
7.1 Allgemeines 160
7.2 Modell des Vermittlungsrechners 160
7.3 Herleitung des Gleichungssystems der Zustandswahrscheinlichkeiten 165
7.3.1 Allgemeines 165
7.3.2 Übergangswahrscheinlichkeiten der Markoffkette 168
7.3.3 Zustandswahrscheinlichkeiten 173
7.3.3.1 Beobachtungszeitpunkt kurz vor dem Taktzeitpunkt 173
7.3.3.2 Beobachtungszeitpunkt zu einer beliebigen Zeit t_d nach dem Taktzeitpunkt 174
7.4 Charakteristische Verkehrsgrößen 176
7.4.1 Allgemeines 176
7.4.2 Verlustwahrscheinlichkeit und mittlere Zahl von Verlustanforderungen (Verlustrate c_v) 177
7.4.3 Belastung 179
7.4.4 Wartewahrscheinlichkeit 179
7.4.5 Wartebelastung 181
7.4.6 Mittlere Wartezeit 182
7.5 Numerische Ergebnisse 185
8. Zusammenfassung 195

Anhang

A1 Allgemeine Begriffe 196
A1.1 Task 196
A1.2 Listen 197
A1.2.1 Bitweise organisierte Liste 197
A1.2.2 Wortweise organisierte Liste 198
A1.2.3 Blockweise organisierte Liste 198

A1.3 Warteschlangen 200
A1.4 Meldungen und Befehle für den Informationsaustausch zwischen dem Vermittlungsrechner und der vermittlungstechnischen Peripherie 200
A2 Zusammenstellung der realisierten Überwacheraufrufe 201
A3 Siemens Rechner 306 204

SCHRIFTTUMSVERZEICHNIS

/ 1/ Keister, W.: The evolution of Telephone Switching. Bell Laboratories Record 43(1965)6,197-203.

/ 2/ Lucas, P.: Les progrès de la commutation électronique dans le monde. Commutation & Electronique 44(1974) Janvier, 5-49.

/ 3/ Fuhrmann, J.J.: The World of Electronic Switching - Facts and Prospects. World Telecommunication Forum, 1975, Geneva, Conference Proceedings 1.5.2.1-1.5.2.5.

/ 4/ Katzschner, L.; Lörcher, W.: Neuere Entwicklungen in der Vermittlungstechnik. Bericht über das International Switching Symposium 1972. Nachrichtentechnische Zeitschrift 25(1972)10, 476-478.

/ 5/ Siemens (Hrsg.): Elektronisches Wählsystem EWS1. Einführung und Übersicht. 1972. Bestell-Nr.: A 30 795-X 185-X-1-18.

/ 6/ Jahrbuch des Elektrischen Fernmeldewesens 22 Verlag für Wissenschaft und Leben (1971). Georg Heidecker Bad Windsheim.

/ 7/ Rechnergesteuerte Vermittlungstechnik (EWS). Nachrichtentechnische Zeitschrift 27(1974)9, K208-K211.

/ 8/ Schulz, K.: Bedienungsrechner für EWS1 Ihre Anwendung im Netz von SPC - Vermittlungsstellen der Deutschen Bundespost. Nachrichtentechnische Zeitschrift 28(1975)5, K185-K188.

/ 9/ Katzschner, L.; Lörcher, W.; Weisschuh, H.: On an Experimental Local PCM - Switching Network.
a) Proceedings International Zürich Seminar on Integrated Systems for Speech, Video and Data Communications 1972, B6/1-5.
b) IEEE - Com 21(1973)10, 1144-1147.

/10/ Schumacher, K.: Das Modell DVA 306 - eine neue Datenverarbeitungsanlage des Systems 300. Elektronische Rechenanlagen 11(1969)2,105-109.

/11/ Siemens (Hrsg.): Zentraleinheit 306, Beschreibung und Befehlsliste. 1971. Bestell-Nr.: D13/3012.

/12/ Siemens (Hrsg.): Handbuch PR 306, Vorläufiges Exemplar. 1971.

/13/ Naumann, H.; Wiczorke, M.: PROSA 300 für Prozeßautomatisierung. Band 4: Das Modell 306. Siemens AG München 1973.

/14/ Lauber, R.: Prozeßautomatisierung I. Springer Verlag Berlin, Heidelberg, New York, 1976.

/15/ Siemens (Hrsg.): Organisationsprogramm, Beschreibung. 1969. Bestell-Nr.: 2-2600-457.

/16/ VDI/VDE - GMR Richtlinie 3554 (Entwurf): Funktionelle Beschreibung von Prozeßrechner-Betriebssystemen. Bericht des ad hoc - Arbeitskreises "Prozeßrechner-Betriebssysteme" des Unterausschusses 2 "Programmierung von Prozeßrechnern" der Sektion 4 "Prozeßrechner zum Messen, Steuern, Regeln" der VDI/VDE - Gesellschaft für Meß- und Regeltechnik.

/17/ Kleinert, H.; Marsing, M.: Betriebssysteme für reelle Adressierung. Siemens AG München 1973.

/18/ Caspers, P.G.: Aufbau von Betriebssystemen. Sammlung Göschen Band 7013. Walter de Gruyter Verlag Berlin 1974.

/19/ Otto, J.; Otto, V.; Hoffart, E.: PROSA 300 für Prozeßautomatisierung. Band 2: Einfache Anwendungen. Siemens AG München 1969.

/20/ Eichenauer, B.F.: Dynamische Prioritätsvergabe an Tasks in Prozeßrechensystemen. Dissertationsschrift, Universität Stuttgart 1975.

/21/ Ein einfaches Zustandsmodell für strikt eingriffsgesteuerten Echtzeitbetrieb. Arbeitsgruppe E2 "Programmierung von Prozeßrechnern" des SFB 49. Technische Universität München Bericht Nr.: 7306, 1973.

/22/ Dijkstra, E.W.: Co-operating Sequential Processes. Programming Languages. Academic Press London 1968, S.43-112.

/23/ Behandlung von Bit 16 (zur Adressierung des oberen Basisadressenregisters) bei den Adreßrechnungsbefehlen der DVA 306. Siemens Mitteilung PS-A 3/71.

/24/ Siemens (Hrsg.): PROSA 300, Beschreibung. 1971. Bestell-Nr.: D13/3014.

/25/ Coffmann, E.G.; Elphick, M.J.; Soshani, A.: System Deadlocks. Computing Surveys 3(1071)2, 67-78.

/26/ Ogata, H.; Nakajo, T.; Awaji, K.: Command Language for D-10 Electronic Switching System. Conference on Software Engineering for Telecommunication Switching Systems, Essex 1973. IEE Conference Publication No. 97, 80-89.

/27/ Langenbucher, G.: Implementierung von Besonderen Leistungsmerkmalen in einem PCM-Vermittlungssystem. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart, Studienarbeit Nr.: 443, 1974.

/28/ Keister, W.; Ketchledge, R.W.; Vaughan, H.E.: No. 1 ESS: System Organization and Objectives. The Bell System Technical Journal 43(1964)5, 1831-1844.

/29/ Clausert, H.; Kahlert, T.: Untersuchung digitaler Schaltungen mit dem Programmsystem DICAP, 1. Teil: Simulation. Elektronische Datenverarbeitung 11(1969) ,443-451.

/30/ Chappell, S.G.; Elmendorf, C.H.; Schmidt, L.D.: Logic - Circuit - Simulators. The Bell System Technical Journal 53(1974)8, 1451-1503.

/31/ Gärtner, E.: Programm zur Simulation der Konzentratoranschlußschaltung einer PCM - Vermittlungsstelle. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart, Studienarbeit Nr. 393, 1972.

/32/ Risak, V.: Simulation von Digitalrechnern. Computer Monographien Bd. 6. Carl Hanser Verlag München 1971.

/33/ Haugk, G.; Tsiang, S.H.; Zimmermann, L.: System Testing of the No. 1 Electronic Switching System. The Bell System Technical Journal 43(1964)5, 2575-2592.

/34/ Takamura, T.; Fujita, S.; Ito, Y.: Support Program System for DEX - 2 Electronic Switching System. Review of the Electrical Communication Laboratory 17(1969)11, 1361-1367.

/35/ Beck, D.S.: The Plessey system 250 - facilities for simulation of telecommunication equipment. Conference on Software Engineering for Telecommunication Switching Systems, Essex 1973. IEE Conference Publication No. 97, 31-38.

/36/ Miller, M.R.: Performance monitoring of SPC - systems under load. Conference on Software Engineering for Telecommunication Switching Systems, Essex 1973. IEE Conference Publication No. 97, 46-50.

/37/ Fontaine, B.: Echtzeit - Umweltsimulation. Elektrisches Nachrichtenwesen 46(1971)3,189-191.

/38/ Foucault, C.C.; Cerny, D.J.; Ponce De Leon, L.: TCS - Umweltsimulation. Elektrisches Nachrichtenwesen 48(1973)4,461-465.

/39/ Lacivita, J.; Merrit, R.; Robinson, R.: STS - A Software Test System for the ETS - 4 Stored Program Controlled Switching System. Proceedings of the International Switching Symposium, Cambridge 1972, 135-138.

/40/ Rozmarin,C.; Trelut, J.; Viellevoye, L.: The E11 Switching System. Proceedings of the International Switching Symposium, München, 1974, 134/1 - 134/8.

/41/ Schramel, F.J.; Rescoe, H.L.: System Support and Software Production of PRX 205. Proceedings of the International Switching Symposium, Cambridge 1972, 378-385.

/42/ Barton, M.E.; Haller, N.M.; Ricker, G.W.: No. 2Ess service Programs. The Bell System Technical Journal 48(1969)8, 2865-2891.

/43/ Mangold, G.: Testhilfeprogramme für das Vermittlungsprogramm einer PCM - Vermittlungsstelle. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart, Studienarbeit Nr. 468, 1975.

/44/ Neovius, G.: Artificial Traffic Trials Using Digital Computers. Ericson Technics 11(1955) 279-291.

/45/ Hutt, J.: Simulation der Vermittlungsperipherie durch ein ALGOL - Programm. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart, Studienarbeit Nr. 457, 1975.

/46/ Dietrich, H.: Erfassung und Auswertung der Laufzeit eines Programms zum Verbindungsaufbau und -abbau einer rechnergesteuerten Vermittlungsstelle. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart, Studienarbeit Nr. 479, 1975.

/47/ Dietrich, G.: Verkehrsmodell für zentralgesteuerte Vermittlungssysteme. Elektrisches Nachrichtenwesen 50(1975)1,30-36.

/48/ Langenbach-Belz,M.: Vergleich zweier Warteschlangenmodelle für Realzeit-Rechnersysteme mit Interrupt - bzw. Takt - gesteuerter Übernahme von Anforderungen aus der Peripherie. 3. GI Jahrestagung Hamburg 1973. In : Lecture Notes in Computer Science, Bd. 1, Springer Verlag Berlin, Heidelberg, New York, 1973, 304-313.

/49/ Wagner, D.: Wartesystem mit getakteter Gruppenankunft und beliebiger Bedienungsdauerverteilung. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart, Studienarbeit Nr. 445, 1974.

/50/ Langenbach-Belz,M.: Getaktete Wartesysteme bei Rechnern und zentralgesteuerten Nachrichtenvermittlungsanlagen. Dissertationsschrift, Universität Stuugart,1973.

/51/ Schwaertzel, H.G.: Serving Strategies of Batch Arrivals in Common Control Switching Systems. Nachrichtentechnische Zeitschrift 27(1974)9, 341-345.

/52/ Weisschuh,H.; Wizgall,M.: Investigations on the Traffic Behavior of the Common Control in SPC Switching Systems. Proceedings of the 8th International Teletraffic Congress, Melbourne 1976,612/1-8.

/53/ Burke, P.J.: Delays in Single - Server Queues with Batch Input. Operations Research 23(1975)4, 830-833.

/54/ Syski, R.: Introduction to Congestion Theory in Telephone Systems. Oliver and Boyd Edinburgh and London, 1960.

/55/ Kleinrock, L.: Queuing Systems. Volume I : Theory. John Wiley & Sons New York, London, Sidney, Toronto, 1975.

/56/ Takacs, L.: Introduction to the Theory of Queues. Oxford University Press, New York, 1962.

/57/ Lotze, A.: Einführung in die Nachrichtenverkehrstheorie. Vorlesung an der Universität Stuttgart.

/58/ Young, D.M. — Iterative solution of large linear systems. Academic Press New York, London, 1971.

/59/ Rieder, P.: — Prozeßzustände bei Echtzeitprogrammiersprachen. Lecture Notes in Computer Science, Bd. 12, Springer Verlag Berlin, Heidelberg, New York, 1974.

/60/ Eichenauer, B.; Haase, V.; Holleczek, P.; Kreuter, K.; Müller, G.: — PEARL, eine prozeß und experimentorientierte Programmiersprache. Angewandte Informatik 15(1973), 363-372.

/61/ Rieder, P.: — Das Task - und Timing Konzept von PEARL. Lecture Notes in Economics and Mathematical Systems, Bd. 78, Springer Verlag Berlin, Heidelberg, New York 1973.

/62/ Coffman, E.J. Jr.; Denning, P.J.: — Operating Systems Theory. Prentice - Hall Inc. Englewood Cliffs New Jersey 1973.

/63/ Hansen, P.B.: — Operating System Principles. Prentice Hall Inc. Englewood Cliffs New jersey 1973.

/64/ Hämmerle, W.: — Plattenspeicherverwaltung für Prozeßrechner Siemens Modell 306. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart, Studienarbeit Nr. 417, 1973.

/65/ Erne, H.: — Organisation der Ein- Ausgabewarteschlangen für den Vermittlungsrechner einer rechnergesteuerten PCM - Vermittlungsstelle. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuutgart, Studienarbeit Nr. 440, 1974.

/66/ Schwager, J.: — Realisierung von Ein - Ausgabefunktionen für das Betriebssystem einer rechnergesteuerten PCM - Vermittlungsstelle. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart, Studienarbeit Nr. 454, 1975.

/67/ Talpalaru, R.: — Realisierung wichtiger Grundbausteine des Betriebssystems einer rechnergesteuerten PCM - Vermittlungsstelle. Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart, Studienarbeit Nr. 441, 1974.

/68/ — Bericht über das System 300 - Beschreibung der wesentlichen Eigenschaften des Organisationsprogramms der DVA 306. Siemens Mitteilung D Lab 184/702b/K/men.

ABKÜRZUNGEN

Die Liste der verwendeten Abkürzungen ist eingeteilt in Abkürzungen, die in der gesamten Arbeit Verwendung finden, und in Abkürzungen, die nur im Kapitel 7 benutzt werden.

Eine Zusammenstellung der auftretenden Überwacheraufrufe mit den benutzten Abkürzungen befindet sich im Anhang A2.

Für die Flußdiagramme wurden die Sinnbilder nach DIN 66001 verwendet. Zusätzlich verwendete Sinnbilder sind am Ende dieses Abschnittes zusammengestellt.

Allgemeine Abkürzungen

ADR	Adresse als formaler Parameter in einem Überwacheraufruf
Adr	Adresse allgemein
Akk	Akkumulator
Ans	Anschlußnummer
ANZ	Anzeige
AP	Arbeitsprogramm
APU	Ausgabepuffer
AWS	Ausgabewarteschlange
AWSR	Ausgabewarteschlange für reguläre Befehle
AWSF	Ausgabewarteschlange für Befehle zur Fehlerbearbeitung
AZ	Alarmzustand
B_i	Bearbeitungsroutine i
BAR	Basisadressenregister
BAR x_i	Basisadressenregister x zur Task i gehörend
BR	Befehlsregister
Bz	Befehlszählerstand
CCITT	Comité Consultativ International Téléphonique et Télégraphique
CW	Codewort
DEST	Dezentrale Steuereinheit (KAS,VAS,ZKF)
DP	Dienstprogramme
DP_r	Dienstprogramme resident im Arbeitsspeicher
E/A	Ein/Ausgabe
EAP	Ein-Ausgabeprogramme
EPU	Eingabepuffer
EWS	Eingabewarteschlange
EWSF	Eingabewarteschlange für Fehlermeldungen
EWSR	Eingabewarteschlange für reguläre Meldungen
F	Fehlerbearbeitungsroutine
FP	Fehlerbearbeitungsprogramme

GER	Gerät
HP	Hilfsprogramme
HZ	Hilfszeiger
K	1 K = 1024
KAS	Konzentratoranschlußschaltung
Kt	Konzentrator
KP	Kanalprogramme
LZ	Lesezeiger allgemein
LZP	Lesezeiger für Eingabepuffer
M	Marke
MA	Makroaufruf
MR	Melderegister
MR_K	Melderegister zum Konzentrator
MR_V	Melderegister zum Vermittlungsrechner
MUX	Multiplexer
MZ	Merkzähler
n	Zahl der dezentralen Steuereinheiten bzw. Zahl der Speicherplätz im Eingabe- oder Ausgabepuffer
NOP	Nullbefehl
NZ	Normalzustand
NVDV	Institut für Nachrichtenvermittlung und Datenverarbeitung der Universität Stuttgart
ORG	Organisationsprogramm
ORGi	Organisationsprogramm der Prioritätsklasse i
ORG306	Organisationsprogramm für den Rechner Siemens 306
PCM	Puls-Code-Modulation
P(>t)	Bearbeitungszeitverteilung für Meldungen
P_{max}	letzte Zelle des Eingabepuffers
P_{min}	erste Zelle des Eingabepuffers
PZ	Privilegierter Zustand
RP	Routineprüfprogramme
Sem	Semaphor
SDA	Schnelldrucker
SDP	Sonderdienstprogramme
SPR	Sprungbefehl
SZ	Schreibzeiger
S306	Siemens Rechner 306 (handelsüblicher Prozeßrechner)
T_o-Takt	Takt zur Aktivierung des Organisationsprogrammes ORG6
TBR	Taktbearbeitungsroutine
Tln	Teilnehmer
A-Tln	rufender Teilnehmer
B-Tln	gerufener Teilnehmer
T_M	Multiplexertakt

T_U	Takt für Unterbrechungssignale (Alarmsignale) an den Vermittlungsrechner
TKB	Taskkontrollblock
TWS	Taskwarteschlange
U_i	Unterbrechungsbearbeitungsroutine i
UAR	Unterbrechungsausgangsroutine
UER	Unterbrechungseingangsroutine
UNT	Befehl "Unterprogrammsprung"
UP	Unterprogramm
VAS	Vermittlungsstellenanschlußschaltung
VP	Vermittlungsprogramme
VR	Vermittlungsrechner
VSt	Vermittlungsstelle
A-VSt	Vermittlungsstelle beim rufenden Teilnehmer
B-VSt	Vermittlungsstelle beim gerufenen Teilnehmer
WZE	Wählzeichenempfänger
Z	Arbeitsspeicherzelle Z
<Z>	Inhalt der Arbeitsspeicherzelle Z
Zk	Zähler k
ZkM	maximaler Zählerstand des Zählers k
Z_{max}	letzte Zelle einer Warteschlange
Z_{min}	erste Zelle einer Warteschlange
ZKF	Zentral-Koppelfeld
ZP	Zeitprogramme
τ	Taktperiodendauer (Taktzeit) für die Ein-Ausgabe zur Vermittlungsperipherie

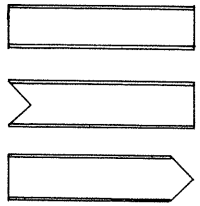
Abkürzungen, die nur im Kapitel 7 benutzt werden

A	Angebot
B	Verlustwahrscheinlichkeit
c_v	Verlustrate
$E[g]$	Erwartungswert für die pro Taktzeitpunkt eintreffenden Anforderungen
g	Größe einer zum Taktzeitpunkt eintreffenden Gruppe von Anforderungen
g'	Grenzwert für die Gruppengröße nach Gleichung (7-8)
g_{grenz}	maximale Zahl von Anforderungen einer ankommenden Gruppe, die vom System angenommen wird
g_m	mittlere Gruppengröße
g_{max}	maximale Gruppengröße
h	mittlere Bedienungszeit für eine Anforderung
h'	mittlere Bedienungszeit für eine Teilanforderung
k	Parameter der Erlang-k-Verteilungsfunktion

N	maximaler Wert der Zufallsvariablen $X(t)$
$P(>t)$	Verteilungsfunktion für die Bedienungszeit
$P(T_h>t)$	Verteilungsfunktion für die Bedienungszeit
$p(x')$	stationäre Zustandswahrscheinlichkeit für den Systemzustand kurz vor dem Taktzeitpunkt
$p(x,t_d)$	stationäre Zustandswahrscheinlichkeit für den Systemzustand zur Zeit t_d nach dem letzten Taktzeitpunkt
$P\{X(t_i)=x_i\}$	Wahrscheinlichkeit, daß zum Zeitpunkt t_i die Zufallsvariable $X(t_i)$ den Wert x_i hat
q_i	Wahrscheinlichkeit, daß für die Gesamtbedienungszeit T von x' Teilanforderungen gilt $i\cdot t_b < T \leq (i+1)\cdot t_b$
$\{R_i\}$	Teilzustandsraum
$r(z)$	Wahrscheinlichkeit, daß in der Zeit t_b insgesamt z Teilanforderungen abgefertigt werden
$r_{min}(z)$	Wahrscheinlichkeit, daß in der Zeit t_b mindestens z Teilanforderungen abgefertigt werden
s	Zahl der Warteplätze
t	Zeit
t_i	Taktzeitpunkt
t_i^*	Zeitpunkt kurz vor dem Taktzeitpunkt $t_i^* = t_i - 0$
t_b	Zeit, die zwischen zwei Taktzeitpunkten zur Bearbeitung von Anforderungen zur Verfügung steht
T_h	Zufallsvariable für die Bedienungszeit
t_d	Zeit nach dem letzten Taktzeitpunkt
t_v	Verwaltungszeit (Overhead) pro Taktzeitpunkt
$\ddot{u}(x_n,x_{n-1})$	zeitunabhängige Übergangswahrscheinlichkeit des Zustandsprozesses X
$\ddot{u}_{t_d}(x,y)$	Übergangswahrscheinlichkeit vom Zustand x kurz vor dem Taktzeitpunkt in den Zustand y zur Zeit t_d nach dem Taktzeitpunkt
v	mittlere Zahl abgewiesener Anforderungen
v_i	mittlere Zahl abgewiesener Anforderungen für den Teilzustandsraum R_i
W	Wartewahrscheinlichkeit
W'	Wartewahrscheinlichkeit nach der Verwaltungszeit
w	mittlere Wartezeit bezogen auf alle Anforderungen
$w(x')$	mittlere Wartezeit für eine Anforderung, die x' Teilforderungen im System antrifft
$w(g)$	mittlere Wartezeit für eine Gruppe von Anforderungen der Größe g
$w_1(x')$	mittlere Wartezeit einer Anforderung, die x' Teilanforderungen im System antrifft, bedingt durch die Verwaltungszeit t_v
$w_2(x')$	mittlere Wartezeit einer Anforderung, die x' Teilanforderungen im System antrifft, bedingt durch die Bedienungszeiten der bereits wartenden Teilanforderungen
x	Zahl der Anforderungen im System
$X(t)$	Zufallsvariable für die Zahl von Anforderungen im System
$\{X(t),t\geq 0)\}$	Zufallsprozeß der Systemzustände (Zustandsprozeß)

x'	Zahl von Teilanforderungen im System kurz vor dem Taktzeitpunkt
$X'(t_i^*)$	Zufallsvariable für die Zahl von Teilanforderungen im System kurz vor dem Taktzeitpunkt t_i
x^*	Zahl der Teilanforderungen im System kurz vor dem Taktzeitpunkt
$X^*(t_i)$	Zufallsvariable für die Zahl von Anforderungen im System kurz vor dem Taktzeitpunkt t_i
Y	Belastung
Y_b	Belastung durch Bedienungszeiten
Y_v	Belastung durch Verwaltungszeiten
z	Zahl der während der Zeit t_b abgefertigten Teilanforderungen
λ	mittlere Anrufrate
τ	Taktperiode (Taktzeit)
$\Omega(t_d)$	Wartebelastung durch Anforderungen zu einem beliebigen Zeitpunkt t_d nach dem letzten Taktzeitpunkt
σ	Varianz

Zusätzliche Sinnbilder für die Flußdiagramme



Überwacheraufruf allgemein

Überwacheraufruf ERWARTE NACHRICHT

Überwacheraufruf SENDE NACHRICHT

1. Einleitung

1.1 Allgemeines über rechnergesteuerte Vermittlungsstellen

Im Jahre 1965 wurde in Succasunna, New Jersey, USA, das erste kommerzielle rechnergesteuerte Vermittlungssystem der Welt eingeschaltet /1/. Seit dieser Zeit haben bereits einige Hundert weitere Systeme den Betrieb aufgenommen, und fast alle Postverwaltungen der Welt entwickeln und planen den Einsatz rechnergesteuerter Vermittlungssysteme /2,3,4/. Die Deutsche Bundespost hat dem allgemeinen Entwicklungstrend folgend 1974 die ersten drei Versuchsvermittlungsstellen in Betrieb genommen /5,6,7/.

Die rechnergesteuerten Vermittlungssysteme mit gespeichertem Steuerprogramm bieten gegenüber den herkömmlichen Systemen mit dezentraler Steuerung oder zentralen Steuerungen, die aber festverdrahtet sind, wesentliche Vorteile, sowohl für den Teilnehmer als auch für den Betreiber des Vermittlungssystems (Verwaltung).

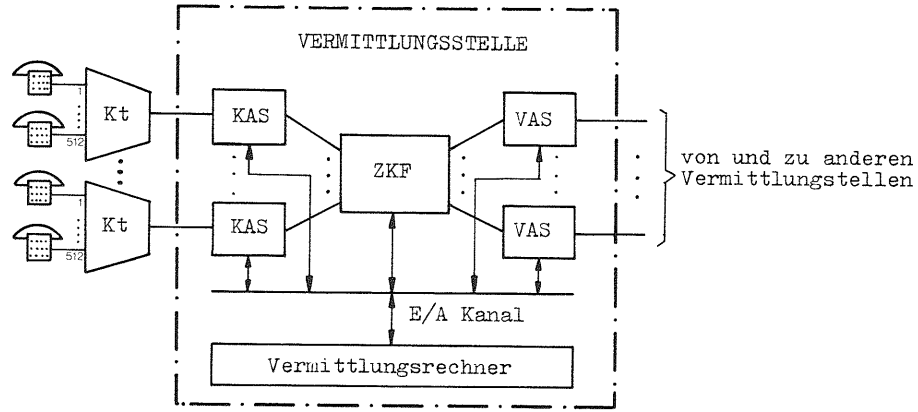
Dem Teilnehmer werden neue Dienste, die besonderen Leistungsmerkmale (Facilities), angeboten, welche bei den herkömmlichen Systemen mit tragbarem Aufwand nicht möglich waren; z.B. Kurzwahl, Fangen, Anklopfen, Anrufumleitung usw. Diese Dienste werden auf Antrag des Teilnehmers zur Verfügung gestellt. Sie können vom Bedienungspersonal der Vermittlungsstelle, zum Teil auch vom Teilnehmer selbst über dessen Teilnehmerapparat, aktiviert werden.

Entscheidend für die Einführung der rechnergesteuerten Vermittlungssysteme waren jedoch nicht die neuen Dienste für den Teilnehmer, sondern vielmehr die Betriebserleichterungen für die Verwaltung. Im Vordergrund dabei steht die Rationalisierung der Wartung und des Betriebes und die Vereinfachung der Störungsbeseitigung.

Die Einrichtungen der Vermittlungsstelle werden während ihres Funktionsablaufes entweder mitlaufend überwacht oder routinemäßig oder aufgrund von Störungen überprüft. Fehlerhafte Funktionseinheiten werden automatisch gesperrt oder es wird eine Ersatzschaltung vorgenommen. Die Erfassung von Störungsdaten, Verkehrsdaten, Prüfdaten, die bisher manuell durchgeführt wurde, kann ebenfalls automatisch durch den Rechner durchgeführt werden. Es ist sogar möglich, Fernmeldedienststellen für mehrere Vermittlungsstellen zu zentralisieren, und diesen über einen Bedienrechner /8/ Zugriff zu den einzelnen Vermittlungsrechnern zu geben. Der Bedienrechner ermöglicht weiterhin eine zentrale Gebührenerfassung und Rechnungserstellung, ferner eine Änderung der teilnehmerbezogenen Daten in den Vermittlungsrechnern von zentraler Stelle aus.

1.2 PCM-Projekt des Instituts für Nachrichtenvermittlung und Datenverarbeitung (NVDV)

Im Rahmen eines Forschungsprojektes über "Systemkonzepte und Software elektronischer Daten- und Fernsprechvermittlungen mit Steuerung durch Zentralrechner" wird am Institut NVDV ein Labormodell einer rechnergesteuerten PCM-Vermittlungsstelle aufgebaut /9/. Das Steuerungskonzept dieser Vermittlungsstelle (VSt) löst sich von der "totalen Zentralisierung", bei der alle Steuerungsaufgaben im Vermittlungsrechner (VR) ausgeführt werden, und überträgt möglichst viele einfachere Vermittlungsfunktionen dezentralen Steureinheiten. Die Übertragung und Vermittlung der Sprachsignale erfolgt digital über PCM-Systeme. Die Struktur der Vermittlungsstelle ist im Bild 1.1 angegeben.



Kt Konzentrator
KAS Konzentratoranschlußschaltung
ZKF Zentral-Koppelfeld
VAS Vermittlungsstellenanschlußschaltung

Bild 1.1: Struktur der PCM-Vermittlungsstelle

Die Teilnehmer sind über Konzentratoren (Kt) im Vorfeld an das Vermittlungssystem angeschlossen. Im Konzentrator wird der Verkehr von maximal 512 Teilnehmern (Tln) auf ein oder zwei 30/32 Kanal PCM-Systeme konzentriert und auch die für die Sprachübertragung notwendige Analog/Digital- bzw. Digital/Analogumsetzung durchgeführt. Weiterhin werden die vom Teilnehmer über Tastwahlapparate gewählten Ziffern über Wählzeichenempfänger (WZE) aufgenommen und der VSt übermittelt.

Jedem Kt ist in der VSt eine Konzentratoranschlußschaltung (KAS) zugeordnet, mit der er über den Signalisierkanal 16 des PCM-Übertragungssystems verbunden ist. Die KAS hat als wesentliche Aufgabe den Verbindungsaufbau und -abbau zwischen dem Tln und der VSt zu steuern. Die Aufgabe der KAS werden anhand eines einfachen Verbindungsaufbaues erklärt:

Wenn ein Tln am Kt abhebt, wird dies von der Kt-Steuerung erkannt und der KAS mitgeteilt. Die KAS sucht einen freien Kanal und einen freien WZE und veranlaßt die Durchschaltung im Kt auf diesen Kanal und die Anschaltung des WZE. Nach erfolgreicher Ausführung wird dem Tln in der KAS der Wählton angeschaltet. Ferner meldet die KAS dem VR, daß ein bestimmter rufender Tln abgehoben hat und mit welchem Kanal sowie mit welchem WZE er verbunden ist.

Die vom Tln gewählten Ziffern werden von dem WZE über die Kt-Steuerung zur KAS übertragen, welche sie in Form einer "Meldung" direkt an den VR weitergibt. Erst jetzt übernimmt der VR die weitere Steuerung des Verbindungsaufbaues.

Der Verbindungsaufbau von der VSt zu einem gerufenen Tln derselben VSt erfolgt durch einen Befehl des VR an die KAS, welche die Durchschaltung im Kt veranlaßt.

Ebenso wie der Verbindungsaufbau durch die KAS gesteuert wird, wird auch der Abbau einer Verbindung durch die KAS durchgeführt. Dabei wird beim Auflegen eines rufenden Tln automatisch die Verbindung zur VSt getrennt. Beim Auflegen eines gerufenen Tln erfolgt die Auftrennung der Verbindung auf Veranlassung des VR. Dem VR wurde zuvor durch die KAS auf der Seite des gerufenen Tln das Auflegen mitgeteilt.
Jedem abgehenden PCM-System zu einer anderen VSt ist eine Vermittlungsstellenanschlußschaltung (VAS) zugeordnet, die im wesentlichen die Aufgabe hat, die Signalisierung zwischen den VR der beiden VSt durchzuführen.

Das Zentral-Koppelfeld (ZKF) verbindet die verschiedenen ankommenden und abgehenden PCM-Systeme miteinander und führt die Durchschaltung zwischen den einzelnen Sprachkanälen aus. Die Wegesuche für das Koppelnetz erfolgt im VR, der die notwendigen Einstellbefehle zur Durchschaltung an das Koppelnetz ausgibt.

Der VR wertet die vom Tln gewählten Ziffern aus und führt nach dem Eintreffen der letzten Ziffer einer Rufnummer die Wegesuche für das ZKF durch und veranlasst die Durchschaltung. Weiterhin steuert er den Verbindungsaufbau zum gerufenen Tln über die betreffende KAS,

und veranlasst das Anschalten der verschiedenen Hörtöne durch die KAS. Nach Gesprächsende wird die Verbindung durch den VR getrennt und die Gebühren berechnet. Zusätzlich zu den allgemeinen Aufgaben für den Verbindungsaufbau und -abbau führt der VR alle Aufgaben durch, die im Zusammenhang mit besonderen Leistungsmerkmalen (z.B. Kurzwahl, Fangen) stehen.

Das Programmsystem des VR ermöglicht bei auftretenden Fehlern automatisch eine Fehlerdiagnose auszuführen und Maßnahmen zur Behebung des Fehlers durch das Bedienungspersonal zu veranlassen. Weiterhin werden durch Routineprüfprogramme alle Baugruppen der VSt laufend geprüft.

Der Informationsaustausch zwischen dem VR und den dezentralen Steuereinheiten (KAS, VAS, ZKF) erfolgt getaktet alle 10 ms. Zum Taktzeitpunkt kann pro Steuereinheit jeweils eine Meldung übernommen werden und ein Befehl ausgegeben werden.

Der Kt, die KAS und die VAS sind mikroprogrammierte Steuereinheiten. Als VR wird ein handelsüblicher Prozeßrechner - der Rechner Siemens 306 - verwendet /10/.

Im Gegensatz zur Hardware, die nur für eine kleine Ausbaustufe realisiert wird, ist die Software so ausgelegt, daß sie auch für einen Ausbau mit 8000 Teilnehmern eingesetzt werden könnte.

1.3 Übersicht über die Arbeit

In Kapitel 2 wird zunächst der prinzipielle Entwurf der Softwarestruktur des Vermittlungsrechners, die noch unabhängig vom verwendeten Rechnertyp ist, erläutert. Anschließend wird gezeigt, wie die Hardware des Rechners Siemens 306 durch zusätzliche Software ergänzt wird, um den Anforderungen des Vermittlungssystems zu genügen. Diese Ergänzungssoftware ist ein Bestandteil des Überwachers, der eine Softwareebene zwischen der eigentlichen Vermittlungssoftware und der Hardware des Rechners bildet. Der Überwacher wird in Kapitel 3 ausführlich beschrieben.

Die Aufgaben, die im VR auszuführen sind, werden durch sog. Arbeitsprogramme ausgeführt. Der Ablauf dieser Arbeitsprogramme wird durch Organisationsprogramme gesteuert und überwacht. In Kapitel 4 werden diese Organisationsprogramme und ihr Zusammenspiel mit den Arbeitsprogrammen behandelt.

In Kapitel 5 werden die Probleme des Speicherschutzes, der die Programme im Arbeitsspeicher vor Überschreiben durch fehlerhafte Programme schützen soll, behandelt,und es wird eine mögliche Realisierung für den Rechner Siemens 306 angegeben.

Die Software des VR ist ein sehr komplexes Programmsystem und es ist daher auch sehr schwierig die Programme auszutesten. Als Testhilfsmittel kann die digitale Simulation eingesetzt werden, die in Kapitel 6 behandelt wird. Es werden dabei alle Möglichkeiten aufgezeigt, die die digitale Simulation bietet, insbesondere wird gezeigt, wie mit Hilfe der Simulation auch die realen Programmlaufzeiten der Vermittlungsprogramme bestimmt werden können. Hierzu werden auch einige Ergebnisse mitgeteilt.

Im letzten Kapitel wird ein vereinfachtes mathematisches Modell für die Steuerungsabläufe im VR entwickelt und analytisch untersucht. Es wird dabei insbesondere die Auswirkung der Verwaltungszeit im VR beim Informationsaustausch mit der Vermittlungsperipherie untersucht und anhand von numerischen Ergebnissen diskutiert.

2. Struktur der Systemsoftware für den Steuerrechner

2.1 Prinzipielle Struktur der Systemsoftware

2.1.1 Allgemeines

Der zentrale Steuerrechner des Vermittlungssystems (Vermittlungsrechner) steuert und überwacht mit seinen Programmen alle Vorgänge im Vermittlungssystem. Die Gesamtheit der Programme dieses Steuerrechners wird als "Systemsoftware des Vermittlungsrechners" bezeichnet. (Im folgenden wird dafür nur kurz Systemsoftware geschrieben.) Die Systemsoftware wird in drei Klassen von Programmen aufgeteilt (Bild 2.1).

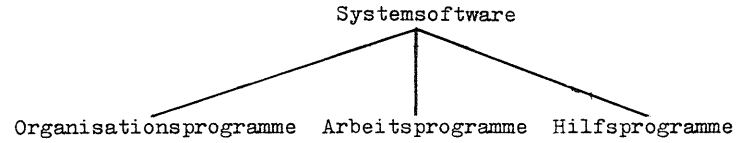


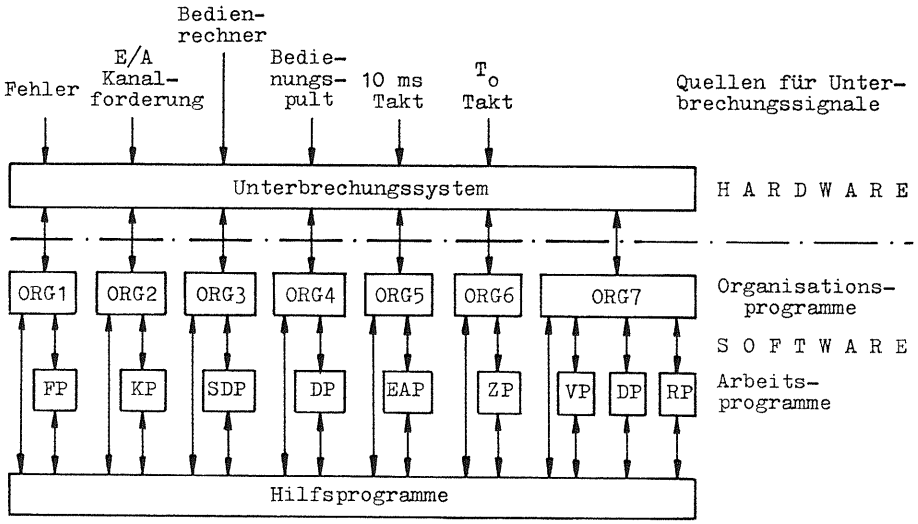
Bild 2.1: Systemsoftware des Steuerrechners (Vermittlungsrechner)

Die Organisationsprogramme steuern und überwachen den Ablauf der Arbeitsprogramme, welche die eigentlichen Aufgaben im Vermittlungsrechner durchführen. Häufig wiederkehrende Funktionen stehen den Organisations- und Arbeitsprogrammen als Hilfsprogramme zur Verfügung.

Die Aufgaben, die von den verschiedenen Arbeitsprogrammen durchgeführt werden, haben unterschiedliche Dringlichkeiten (Prioritäten). Deshalb sind die Programme der Systemsoftware in mehrere Prioritätsklassen eingeteilt.

2.1.2 Struktur und Prioritätsklassen

Die Systemsoftware ist in sieben Prioritätsklassen eingeteilt (vgl. Bild 2.2). Jede Prioritätsklasse hat ihr eigenes Organisationsprogramm (ORG). Dieses ORG steuert und überwacht den Ablauf der in seiner Prioritätsklasse ablaufenden Arbeitsprogramme (AP). Die Arbeitsprogramme sind auf die verschiedenen Prioritätsklassen entsprechend ihrer Aufgaben und Dringlichkeiten aufgeteilt. Sie können nur unter dem ORG ihrer Prioritätsklasse ablaufen. Allen Organisations- und Arbeitsprogrammen ist gemeinsam eine Anzahl von Hilfsprogrammen (HP) zugeordnet.



ORG	Organisationsprogramm (Nummer = Prioritätsklasse)	DP	Dienstprogramme
		EAP	Ein-Ausgabeprogramme
FP	Fehlerbehandlungsprogramme	VP	Vermittlungsprogramme
KP	Kanalprogramme	RP	Routineprüfprogramme
SDP	Sonderdienstprogramme		

Bild 2.2: Prinzipielle Struktur der Systemsoftware

Die Organisationsprogramme werden durch das Unterbrechungssystem des Rechners aktiviert. Anreize in Form von Unterbrechungssignalen an das Unterbrechungssystem können kommen von Fehlererkennungsschaltungen, von Ein-Ausgabe-Kanalwerken, von einem Datenkanal zu einem Bedienrechner, vom Bedienungspult des Rechners oder von verschiedenen Taktgebern, die zum Anstoßen von Real-Time Aufgaben vorgesehen sind. Die Eingänge für Unterbrechungssignale sind in Bild 2.2 als Pfeile über dem Unterbrechungssystem eingezeichnet. Den Unterbrechungssignalen sind Dringlichkeiten (Prioritäten) zugeordnet, die von links nach rechts abnehmen, d.h. Fehlersignale müssen vor E/A-Kanalanforderungen bearbeitet werden.

Die eintreffenden Unterbrechungssignale werden vom Unterbrechungssystem registriert und dieses vergleicht nun die Prioritätsklasse, die gerade in Bearbeitung ist, mit der Prioritätsklasse des neu eingetroffenen Unterbrechungssignals. Falls das neu eingetroffene Unterbrechungssignal eine höhere Priorität hat, wird das gerade laufende Programm

unterbrochen, und die Kontrolle wird an das ORG der neu zu bearbeitenden Prioritätsklasse übergeben. Wenn die Aufgaben in der höheren Prioritätsklasse beendet sind, gibt das ORG dieser höheren Prioritätsklasse die Kontrolle an das Unterbrechungssystem zurück, das seinerseits ein zuvor unterbrochenes Programm wieder zur Ausführung bringt.

Die Prioritätsklassen 1 bis 6 sind unterbrechende Prioritätsklassen, d.h. wenn eine Anforderung (Unterbrechungssignal) höherer Priorität eintrifft, wird das gerade laufende Programm niederer Priorität unterbrochen und das zur Anforderung gehörende Programm höherer Priorität aktiviert.

Die Grundprioritätsklasse 7 ist in drei weitere nichtunterbrechende Prioritätsklassen unterteilt. Die Programme dieser Prioritätsklassen laufen jeweils bis zu einer Unterbrechbarkeitsstelle und können dann erst durch dringendere Programme der Grundprioritätsklasse unterbrochen werden.

Die Arbeitsprogramme der Organisationsprogramme ORG1 bis ORG6 sind resident im Arbeitsspeicher, um schnelle Reaktionen auf Anforderungen zu garantieren. Von den Arbeitsprogrammen des ORG7 sind nur die Vermittlungsprogramme und ein Teil der Dienstprogramme resident im Arbeitsspeicher, während der andere Teil vor der Ausführung vom Magnetplattenspeicher in den Arbeitsspeicher geladen werden muss.

Um von einem Arbeitsprogramm aus Funktionen ausführen zu lassen, die in einem AP eines anderen ORG realisiert sind (z.B. Ein-Ausgabe mit der Peripherie), ist ein Informationsaustausch zwischen den Organisationsprogrammen über ein Hilfsprogramm möglich.

2.1.3 Arbeitsprogramme

Die Arbeitsprogramme haben entsprechend der Prioritätsklasse, zu der sie zugeordnet sind, unterschiedliche Aufgaben:

- FEHLERBEHANDLUNGSPROGRAMME (FP):

  Die FP dienen bei Auftreten eines Fehlers dazu, den Fehler zu lokalisieren, d.h. die defekte Baueinheit zu bestimmen, eine Ersatzschaltung vorzunehmen und dem Wartungspersonal Hinweise für die Reparatur zu geben.

- KANALPROGRAMME (KP):

  Die KP führen die Ein-Ausgabe von und zu jenen peripheren Geräten des Rechners durch, die nicht zur vermittlungstechnischen Peripherie gehören (z.B. Lochkartenleser).

- SONDERDIENSTPROGRAMME (SDP):

Die SDP führen betriebstechnische Aufgaben durch, die im Zusammenwirken mit dem Bedienrechner notwendig sind, und führen den Informationsaustausch mit diesem Rechner durch.(Z.B. Aufgaben, die im Zusammenhang mit dem Fernsprechentstörungsdienst oder dem Fernsprechauftragsdienst stehen.)

- DIENSTPROGRAMME (DP):

Die DP dienen dem Betrieb und der Verwaltung der Vermittlungsstelle. Sie steuern den Informationsaustausch mit dem Vermittlungspersonal und werden unter anderem verwendet zum Programmtest, zur Programmverwaltung, zu Statistik, zur Gebührenerfassung und zum Ändern von teilnehmerbezogenen Daten im Speicher des Rechners.

Die DP sind auf die Prioritätsklassen 4 und 7 entsprechend ihrer Dringlichkeit aufgeteilt. Der Informationsaustausch mit dem Vermittlungspersonal erfolgt unter der Priorität 4, damit Softwarefehler in den Prioritätsklassen 5 bis 7 den Zugriff nicht behindern. Der größte Teil der DP wie z.B. die Gebührenerfassung und die Anderung von teilehmerbezogenen Daten wird in der Prioritätsklasse 7 ausgeführt.

- EIN-AUSGABEPROGRAMME (EAP):

Die EAP steuern den Informationsaustausch mit der vermittlungstechnischen Peripherie. Sie übernehmen Meldungen von der Peripherie, tragen diese in Eingabewarteschlangen ein und geben in umgekehrter Richtung Befehle aus Ausgabewarteschlangen an die vermittlungstechnische Peripherie ab. Der Anstoß der EAP erfolgt durch einen zentalen 10 ms - Taktgeber.

- ZEITPROGRAMME (ZP):

Die ZP sind Programme, die in bestimmten zeitlichen Abständen aufgerufen werden müssen. Es sind teilweise Überwachungsprogramme, die zeitliche Abstände überwachen, und Programme für Verkehrsmessungen. Der Anstoß für die Zeitprogramme erfolgt durch einen zentralen Takt T_o.

- VERMITTLUNGSPROGRAMME (VP):

Die VP steuern alle vermittlungstechnischen Vorgänge in der Vermittlungsstelle. Sie verarbeiten dabei die Meldungen von der Peripherie und erzeugen Befehle an die Peripherie.

- ROUTINEPRÜFPROGRAMME (RP):

Die RP dienen zur routinemäßigen Prüfung der Systemeinrichtungen. Sie werden aktiviert wenn der Vermittlungsrechner gerade keine anderen Aufgaben durchzuführen hat.

2.2 Struktur des Unterbrechungssystems des Rechners Siemens 306

Bei dem Entwurf der Systemsoftware wurde davon ausgegangen, daß ein Steuerrechner vorhanden ist, welcher die gewünschte Zahl von sieben Unterbrechungsebenen und die dazugehörenden Funktionszustände (bestimmte Betriebszustände) besitzt. Weiterhin sollten die Hilfsprogramme "reentrant" geschrieben sein, so daß sie in allen Ebenen benutzt werden können.

Der für die Implementierung der Systemsoftware verwendete handelsübliche Prozeßrechner Siemens 306 (S306) hat jedoch eine andere Struktur mit nur drei Unterbrechungsebenen. Deshalb muß die Hardware des S306 durch zusätzliche Software so ergänzt werden, daß sie den Anforderungen der Software des Vermittlungssystems genügt.

Im folgenden wird deshalb kurz die Struktur des Unterbrechungssystems des Rechners S306 beschrieben. Im nächsten Abschnitt wird dann die Struktur der Systemsoftware unter Berücksichtigung der zusätzlichen Software für den Rechner S306 behandelt.

Die Struktur des Unterbrechungssystems des Rechners S306 /11,12,13/ ist im folgenden Bild 2.3 dargestellt.

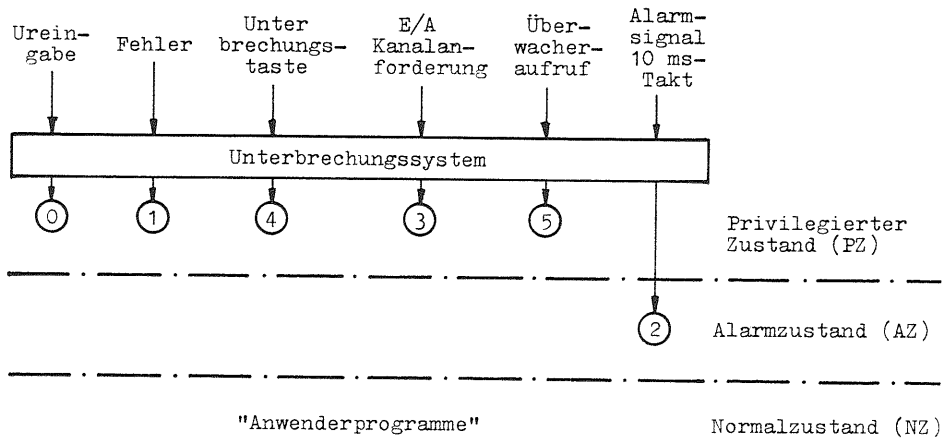


Bild 2.3: Unterbrechungssystem des Rechners Siemens 306

Der Rechner S306 hat insgesamt drei Funktionszustände. Alle Unterbrechungsursachen führen entweder in den Privilegierten Zustand (PZ) oder den Alarmzustand (AZ); der dritte Zustand, der Normalzustand (NZ) ist für die Ausführung der "Anwenderprogramme" vorgesehen, wobei unter "Anwenderprogrammen" alle Programme zu verstehen sind, die nicht in einem der durch eine Unterbrechung hervorgerufenen Zustände PZ oder AZ ablaufen.

Den verschiedenen Unterbrechungsursachen sind feste Einsprungadressen zugeordnet, auf die der Befehlszähler bei einer Unterbrechung eingestellt wird. Die Zahlen in den Kreisen im Bild 2.3 stellen diese Hardwareadressen dar. Entsprechend dieser Adressen können verschiedene Bearbeitungsroutinen aktiviert werden.

Im folgenden werden kurz die verschiedenen Unterbrechungsursachen angegeben:

- FEHLER:

  Bei Auftreten eines Hardwarefehlers (z.B. Parityfehler im Arbeitsspeicher) oder beim Auftreten eines nicht interpretierbaren Befehls erfolgt eine sofortige Unterbrechung .

- UNTERBRECHUNGSTASTE:

  Durch Betätigen einer Unterbrechungstaste am Bedienungsfeld des Rechners kann auch durch das Vermittlungspersonal eine sofortige Unterbrechung jedes laufenden Programms veranlaßt werden.

- E/A-KANALANFORDERUNG:

  Die Ein-Ausgabegeräte können parallel zur Zentraleinheit ihre Aufgaben durchführen. Damit nun diese Geräte das Ende einer Ein-Ausgabeoperation der Zentraleinheit mitteilen können, ist die Unterbrechung durch eine Kanalanforderung vorgesehen.

- ÜBERWACHERAUFRUF:

  Diese Unterbrechung wird durch jenen Umschaltbefehl verursacht, der von einem Programm im Normalzustand des Rechners abgegeben werden kann zur Aktivierung einer Bearbeitungsroutine, die im Privilegierten Zustand ablaufen muß.

- ALARMSIGNAL:

  Zur Durchführung von zeitkritischen Aufgaben kann ein laufendes Programm im Normalzustand oder eine Bearbeitungsroutine im Privilegierten Zustand durch ein sog. "Alarmsignal" unterbrochen werden. Damit wird eine bestimmte Bearbeitungsroutine aktiviert.

Alle Unterbrechungsursachen außer der E/A-Kanalanforderung führen sofort zu einer Unterbrechung der gerade augeführten Befehlsfolge. Die E/A-Kanalanforderung wird nur im Normalzustand des Rechners wirksam und hier auch nur, wenn eine Unterbrechung durch ein Kennzeichnungsbit im Befehlswort zugelassen ist.

Eine weitere fest zugeordnete Zelle im Arbeitsspeicher ist die Adresse Null für die UREINGABE:

Nach dem Einschalten des Rechners muß das Programmsystem eingelesen werden können. Zu diesem Zweck kann über ein Eingabegerät (z.B. Lochkartenleser) eine bestimmte Befehlsfolge ab der Adresse Null in den Arbeitsspeicher eingelesen werden und sofort gestartet werden. Die Ureingabe stellt während des normale Rechnerbetriebs keine Unterbrechungsursache dar; sie wurde jedoch hier zum Unterbrechungssystem hinzugenommen, da sich aus der Struktur des Unterbrechungssystems die Struktur der Software ergibt.

2.3 Ergänzung der Systemsoftware zur Anpassung an die Hardware des S306

Für die Ausführung bestimmter sog. privilegierter Befehle (z.B. Ein-Ausgabebefehle) muß sich der Rechner im Privilegierten Zustand befinden. Damit müssen zwangsläufig alle Programmteile, die diese Befehle benutzen wollen, im Privilegierten Zustand ablaufen.

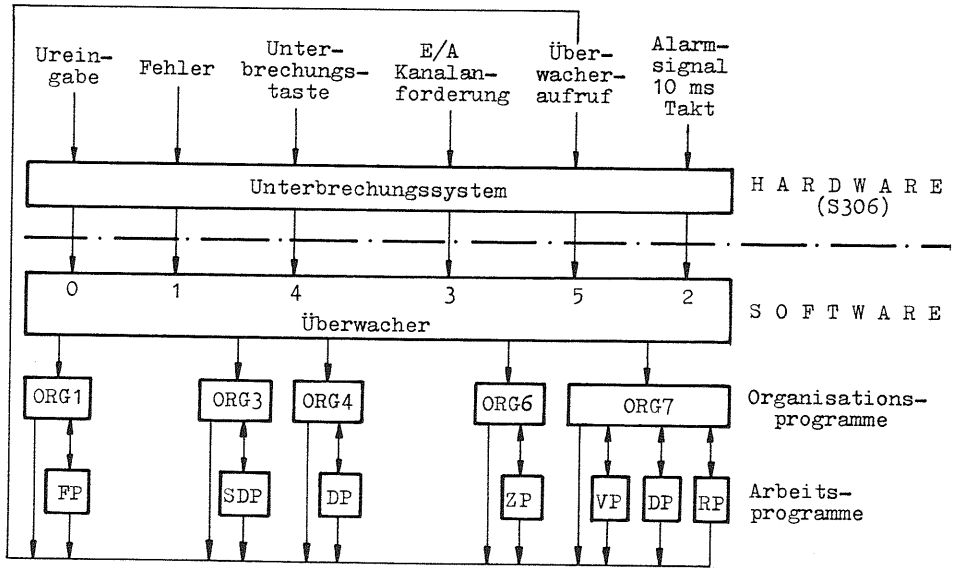
Damit Unterbrechungen durch Kanalanforderungen wirksam werden können, muß sich der Rechner im Normalzustand befinden. Daher sollen alle Programmteile, die keine privilegierten Befehle benutzen, im Normalzustand ablaufen. Ein weiterer Grund hierfür ist die relative Adressierung in Verbindung mit Basisadressenregistern, die einen gewissen Schutz der Programme vor gegenseitigem Überschreiben bietet (s. Abschnitt 5.3).

Es ergibt sich somit die im Bild 2.4 angegebene Struktur für die Systemsoftware wie sie auf dem Rechner S306 implementiert werden muß. Die in Abschnitt 2.2 bereits erwähnte notwendige Ergänzung der Unterbrechungshardware des S306 ist im Bild 2.4 Bestandteil des Überwachers.

Die Organisationsprogramme und Arbeitsprogramme sollen als sog. "Anwenderprogramme" im Normalzustand ablaufen. Alle Programmteile, die Unterbrechungen bearbeiten, sind im Überwacher zusammengefaßt, der im Privilegierten Zustand bzw. im Alarmzustand abläuft. Weiterhin wurden folgende Programme in den Überwacher mit integriert:

- ORG2 mit den dazugehörenden Kanalprogrammen

  Diese Programme sind für die Ein-Ausgabe über die Standardperipherie des Rechners zuständig. Da zur Durchführung der Ein-Ausgabe privilegierte Befehle notwendig sind, müssen diese im Zustand PZ ablaufen. Das Ende von Ein-Ausgabeoperationen wird von den Geräten über Kanalanforderungen signalisiert, was eine Unterbrechungsbearbeitung im Zustand PZ zur Folge hat.



ORG	Organisationsprogramm (Nummer = Prioritätsklasse)	DP	Dienstprogramme
FP	Fehlerbehandlungsprogramme	ZP	Zeitprogramme
SDP	Sonderdienstprogramme	VP	Vermittlungsprogramme
		RP	Routineprüfprogramme

Bild 2.4: Realisierte Struktur der Systemsoftware unter Berücksichtigung der Eigenschaften des Rechners Siemens 306

- ORG5 mit den dazugehörenden Ein-Ausgabeprogrammen

  Diese Programme sind für die Ein-Ausgabe von und zur vermittlungstechnischen Peripherie zuständig. Die Ein-Ausgabe erfolgt taktgesteuert alle 10 ms im Alarmzustand.

- Hilfsprogramme

  Die Hilfsprogramme stellen den Organisations- bzw. den Arbeitsprogrammen bestimmte Funktionen zur Verfügung. Sie sollten wegen ihrer Mehrfachbenutzung durch diese verschiedenen Programme unterschiedlicher Priorität reentrant sein, d.h. mehrfach parallel benutzbar. Dies ist aber bei der Hardwarestruktur des Rechners S306 nicht möglich.

  Deshalb wurden die Hilfsprogramme seriell wiederbenutzbar geschrieben und in den Überwacher verlagert. Der Aufruf der Hilfsprogramme erfolgt jetzt durch den Umschaltbefehl für den Überwacheraufruf, der eine Unterbrechung verursacht und in den Privilegierten Zustand umschaltet. Dadurch ist gewährleistet, daß während der Ausführung eines Hilfsprogrammes kein erneuter Aufruf durch Programme im Normalzustand erfolgen kann.

  Die Hilfsprogramme sollen auch bestimmte Privilegien haben, z.B. sollen sie zu Speicherbereichen zugreifen können, zu denen die "Anwenderprogramme" keinen Zugriff haben dürfen.

Die Bearbeitungsroutinen des Überwachers entsprechen jenen Programmmodulen eines Prozeßrechnerbetriebssystems, die für die Durchführung des Betriebsablaufes notwendig sind /14/. Das Organisationsprogramm ORG306 des Rechnerherstellers für den Rechner S306 hat eine ähnliche Struktur /15/. Es hätte als Überwacher für die Systemsoftware der rechnergesteuerten Vermittlungsstelle Verwendung finden können, jedoch hätten dazu mehrere Funktionen, die speziell für die Durchführung der vermittlungstechnischen Aufgaben notwendig sind, neu eingebaut werden müssen (z.B. die Funktionen für die Ein-Ausgabe von und zur Vermittlungsperipherie). Andere Funktionen hätten abgeändert werden müssen, wieder andere speicherplatzintensive Funktionen des ORG306 hätten nicht benutzt werden können (z.B. die Funktionen zur Verwaltung und Ablaufsteuerung nicht arbeitsspeicherresidenter Programme). Aus diesen Gründen wurde das ORG306 nicht verwendet und ein neuer Überwacher entwickelt, der im folgenden Kapitel 3 beschrieben wird.

Die Programme des S306 zur Vorbereitung und Unterstützung des Betriebsablaufes (Testprogramme und Übersetzer) werden weiterhin verwendet, was einen Einfluß auf einige Bearbeitungsroutinen des neuen Überwachers hat.

Durch die Einführung des Überwachers, der die Schnittstelle zwischen der weiterhin benutzten Hardware des Unterbrechungssystems des S306 und der neuen Vermittlungssoftware bildet, ergibt sich folgende Einteilung für die Systemsoftware der rechnergesteuerten Vermittlungsstelle (Bild 2.5):

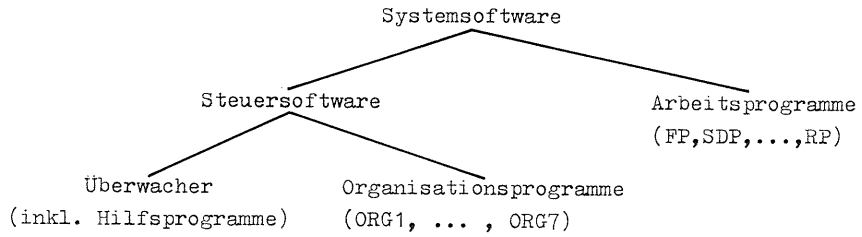


Bild 2.5: Systemsoftware für den Rechner S306 als Vermittlungsrechner

3. Überwacher

Der Überwacher umfaßt alle Bearbeitungsroutinen der Systemsoftware, die nicht im Normalzustand des Rechners ablaufen.

3.1 Struktur des Überwachers

Der Zugang zum Überwacher erfolgt grundsätzlich durch Unterbrechungssignale. Diese werden vom Unterbrechungssystem des Rechners erkannt, die Ausführung der gerade laufenden Befehlsfolge wird unterbrochen und die für die Bearbeitung der Unterbrechung zuständige Bearbeitungsroutine wird automatisch aktiviert.

Im Gegensatz zu jenen Bearbeitungsroutinen, die durch die Hardware des Rechners (Unterbrechungssystem) aktiviert werden, stehen die Tasks (s. Anhang A1). Aus der Sicht des Überwachers ist eine Task eine zusammengehörende Folge von Befehlen und Daten zur Durchführung einer bestimmten Aufgabe unter der Kontrolle des Überwachers. Der Überwacher teilt den Prozessor jeweils einer solchen Verwaltungseinheit zu.

Man unterscheidet zwei Arten von Tasks, die Systemtasks und die Anwendertasks. Die Systemtasks werden im Privilegierten Zustand des Rechners ausgeführt und die Anwendertasks im Normalzustand. Für die Behandlung der Tasks besteht aber kein Unterschied. Im folgenden wird, sofern keine Mißverständnisse entstehen, anstelle von Anwendertask nur Task geschrieben. In /16/ werden die Bearbeitungsroutinen des Überwachers als Rechenprozesse 1. Art bezeichnet und die Anwender- und Systemtasks zusammen als Rechenprozesse 2. Art.

Die Struktur des Überwachers ist im Bild 3.1 dargestellt.

Bei einer Unterbrechungsanforderung wird durch das Unterbrechungssystem die Unterbrechungseingangsroutine (UER) aktiviert. Diese rettet zuerst die für eine spätere Fortsetzung der unterbrochenen Task notwendigen Registerinhalte. Dann wird das Unterbrechungssignal analysiert und die für die Bearbeitung der Unterbrechung notwendige Bearbeitungsroutine aktiviert.

Die Bearbeitungsroutinen führen die entsprechenden Aufgaben durch und geben dann die Kontrolle an die Unterbrechungsausgangsroutine (UAR), welche eine evtl. unterbrochene Bearbeitungsroutine fortsetzt oder die Task mit der höchsten Priorität aktiviert.

Die Bearbeitungsroutinen sind entsprechend ihrer Funktionen zu Blöcken zusammengefaßt. Folgende Funktionsblöcke, die in den folgenden Abschnitten noch ausführlich beschrieben werden, können unterschieden werden:

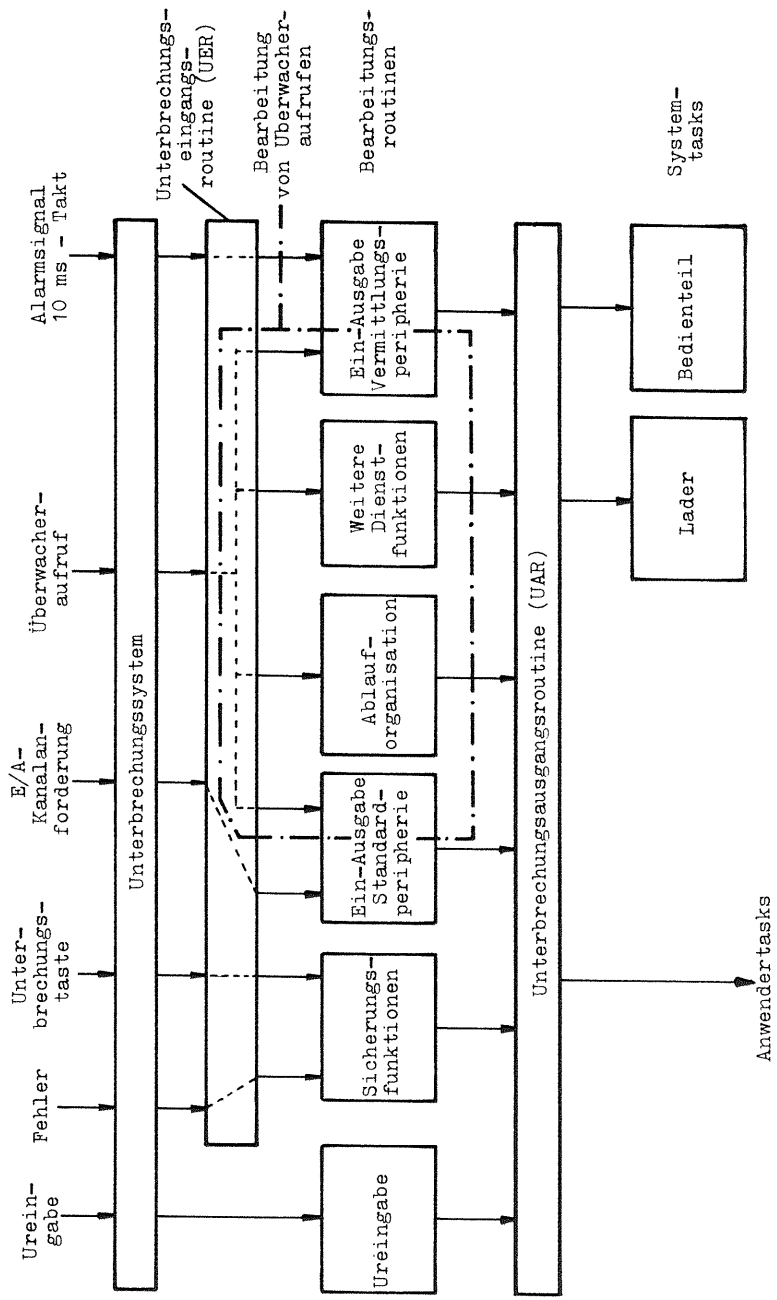


Bild 3.1: Struktur des Überwachers

SICHERUNGSFUNKTIONEN

Diese Bearbeitungsroutinen analysieren auftretende Soft- und Hardwarefehler und brechen bei Bedarf die betreffende Task ab, die den Fehler erzeugt hat.

ABLAUFORGANISATION

Die Bearbeitungsroutinen der Ablauforganisation verwalten und steuern die Tasks. Sie sorgen für die Prozessorzuteilung an die Tasks und steuern ferner die Synchronisation von Tasks und den Informationsaustausch (Nachrichtenaustausch) zwischen Tasks.

EIN-AUSGABE

Bei der Ein-Ausgabe wird unterschieden zwischen der Ein-Ausgabe von und zur Vermittlungsperipherie und der Ein-Ausgabe von und zur Standardperipherie, da diese beiden Gerätegruppen vollkommen unterschiedlich behandelt werden.

Die Vermittlungsperipherie umfaßt die dezentralen Steuereinheiten Konzentratoranschlußschaltung, Vermittlungsstellenanschlußschaltung und Zentral-Koppelfeld (s. Abschnitt 1.2), während die Standardperipherie alle anderen Ein-Ausgabegeräte des Rechners umfaßt (z.B. Lochkartenleser; s. Anhang A3).

Für die Ein-Ausgabe zur Standardperipherie ist für jedes Gerät eine Bearbeitungsroutine (Kanalprogramm) vorgesehen, die den Verkehr zwischen Zentraleinheit und Peripheriegeräten abwickelt. Das Kanalprogramm für den Plattenspeicher übernimmt ausserdem die Dateiverwaltung.

LADER

Der Lader hat die Aufgabe das Programm für eine Task von peripheren Speichermedien in den Arbeitsspeicher zu laden und den Arbeitsspeicher zu verwalten. Er ist als Systemtask realisiert.

BEDIENTEIL

Der Bedienteil hat die Aufgabe, den Informationsaustausch zwischen dem Bedienpersonal und dem Rechner zu unterstützen. Der Bedienteil ist ebenfalls als Systemtask realisiert.

UREINGABE

Der Bearbeitungsroutine Ureingabe obliegt die Aufgabe, den Überwacher in den Arbeitsspeicher zu laden und dessen Bedienteil zu aktivieren.

Im Anschluß daran wird in der Regel auch das gesamte Programmsystem zur Steuerung des Vermittlungssystems (ORG1 bis ORG7 und die Arbeitsprogramme) automatisch geladen.

WEITERE DIENSTFUNKTIONEN

Die Bearbeitungsroutinen für die "Weiteren Dienstfunktionen" bearbeiten alle Aufgaben, die nicht den oben behandelten Bausteinen zugeordnet werden können.

In üblichen Prozeßrechner-Betriebssystemen ist ausser diesen oben genannten Funktionsbausteinen noch ein Baustein für die Zeitverwaltung enthalten /16/. Dieser Baustein hat die Aufgabe Tasks in einem festen Zeitraster oder nach Ablauf einer bestimmten Wartezeit zu aktivieren. Die Funktion dieses Bausteines wurde in das ORG6 (Bild 2.4) verlagert. Im Überwacher ist lediglich in der Bearbeitungsroutine für die Ein-Ausgabe zur vermittlungstechnischen Peripherie ein Teil enthalten, der das ORG6 aktiviert.

In manchen Betriebssystembeschreibungen wird die Unterbrechungsausgangsroutine dem Funktionsblock Ablauforganisation zugeordnet. Wegen der übersichtlicheren Darstellung wird in der vorliegenden Arbeit in Anlehnung an /17/ diese Bearbeitungsroutine getrennt betrachtet.

Die Bearbeitungsroutinen für die Ein-Ausgabe sowie für die Ablauforganisation und die Hilfsfunktionen können von Anwendertasks über den UMSCHALTBEFEHL für einen Überwacheraufruf aktiviert werden. Diese Aktivierung und die Ausführung der Bearbeitungsroutinen wird in den folgenden Abschnitten 3.2 bis 3.5 beschrieben. Dabei werden alle jene Funktionen behandelt, die für das Verständnis von Kapitel 4 über die Organisations- und Arbeitsprogramme notwendig sind.

3.2 Bearbeitung von Überwacheraufrufen

3.2.1 Allgemeines

Um den Überwacher zum Ausführen von bestimmten Funktionen (z.B. die Durchführung einer Ein-Ausgabeoperation) zu veranlassen, muss von der Task, die die Funktion anfordert, der Überwacher aktiviert werden. Dies geschieht mit dem Umschaltbefehl. Zusätzlich zum Umschaltbefehl werden Parameter benötigt, die dem Überwacher angeben, welche Funktionen ausgeführt werden sollen. Der Umschaltbefehl zusammen mit den Parametern wird als Überwacheraufruf bezeichnet.

Ein Überwacheraufruf stellt eine Anforderung an den Überwacher zur Durchführung einer Funktion dar, wobei die Funktion durch Parameter im Aufruf charakterisiert ist.

Bei der Programmierung im Assembler wird ein Überwacheraufruf in der Regel als Makroaufruf geschrieben, für den der Übersetzer dann die entsprechende Befehlsfolge in das Objektprogramm einbaut. Der Überwacheraufruf wird daher auch als Makroaufruf bezeichnet, weitere Bezeichnungen in der Literatur sind Organisationsaufruf, Supervisor Call, Betriebssystemaufruf oder Aufruf an den Ablaufteil /17,18/.

3.2.2 Struktur eines Überwacheraufrufes

Für die Überwacheraufrufe wird die vom Rechnerhersteller verwendete Struktur /15,19/ beibehalten, damit die Übersetzer und Dienstprogramme des Rechnerherstellers weiter Verwendung finden können.

Ein Überwacheraufruf hat folgende Struktur:

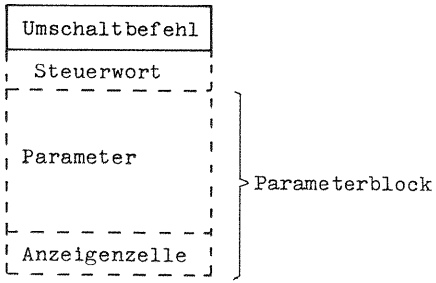


Bild 3.2: Struktur eines Überwacheraufrufes

Der Überwacheraufruf besteht aus einem Umschaltbefehl, zusätzlich eventuell aus einem Steuerwort und einem Parameterblock.

Der Umschaltbefehl bewirkt eine Unterbrechung, durch welche in den Privilegierten Zustand PZ umgeschaltet wird und die Unterbrechungseingangsroutine (UER) aktiviert wird. Es wird dabei der Teil der UER aktiviert, der für die Auswahl der Bearbeitungsroutine für die Aufrufbearbeitung zuständig ist. Der Umschaltbefehl enthält im Adreßteil eine Steuerinformation, anhand der zwei Klassen von Aufrufen unterschieden werden, nämlich Aufrufe mit und ohne Steuerwort (Bild 3.3).

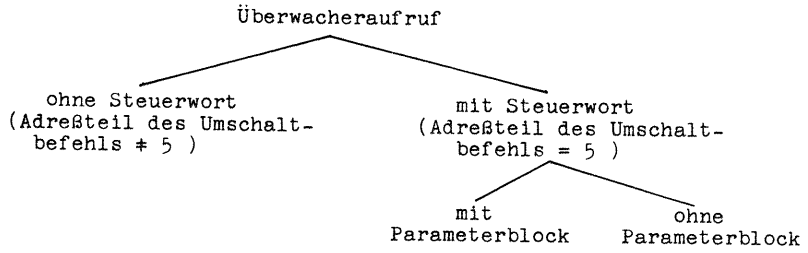


Bild 3.3: Aufteilung der Überwacheraufrufe

Ist der Adreßteil des Umschaltbefehls ≠ 5, so bestimmt er durch die im Adreßteil stehende Zahl direkt die Funktion und damit die Bearbeitungsroutine, die ausgeführt werden soll. Ist der Adreßteil dagegen = 5, so wird die auszuführende Funktion durch das Steuerwort bestimmt. Diese Art der Adreßteilauswertung ist durch die Software des S306 bedingt /15/ (Organisationsprogramm ORG306).

Sind für die Durchführung der Funktion im Überwacher Parameter notwendig, so werden diese im Parameterblock angegeben. Es ist allerdings auch möglich, Parameter über die Akkumulatoren zu übergeben.

Das Steuerwort steht unmittelbar hinter dem Umschaltbefehl. Der Parameterblock muß nicht unbedingt hinter dem Steuerwort stehen, er kann bei bestimmten Aufrufen auch an einer anderen Stelle des Programmes stehen; in diesem Fall steht dann im Steuerwort die Adresse des Parameterblockes.

Nach Ausführung der Funktion durch den Überwacher wird die Task mit jenem Befehl fortgesetzt, der auf den Aufruf folgt. Damit der Überwacher der Task, die den Aufruf abgesetzt hat, mitteilen kann, ob der Aufruf erfolgreich durchgeführt werden konnte, ist im Parameterblock eine Zelle, die sogenannte Anzeigenzelle, vorhanden, in die eine Quittung für die Ausführung als Bitmuster abgelegt werden kann.

Das Organisationsprogramm ORG306 des Herstellers ermöglicht nur bei den Aufrufen eine Quittungsgabe, bei denen ein Parameterblock mit Anzeigenzelle vorhanden ist. Durch den Überwacher können im Gegensatz zum ORG306 bei Aufrufen, die keine Anzeigenzelle besitzen, in den Akkumulatoren Quittungen übergeben werden.

Bei der Aufrufbearbeitung kann man bezüglich der Ausführung zwei Klassen unterscheiden. Zum einen Aufrufe, die sofort ausgeführt werden können (KURZE Überwacheraufrufe) und Aufrufe, die über Warteschlangen abgearbeitet werden (LANGE Überwacheraufrufe).

Aufrufe, die über Warteschlangen abgearbeitet werden, sind Aufrufe bei denen für die Ausführung der Funktion auf ein Ereignis gewartet werden muß; z.B. muß bei Ein-Ausgabeaufrufen auf das Ende der Operation im externen Gerät gewartet werden, das durch ein Unterbrechungssignal (Kanalanforderung) mitgeteilt wird. Wenn für ein Gerät bereits eine Operation angestoßen ist und ein neuer Aufruf für dieses Gerät eintrifft, so muß dieser Aufruf in eine Warteschlange eingereiht werden, da vor dessen Bearbeitung erst der bereits aktivierte Aufruf abgearbeitet werden muß.

Als Beispiel für einen Überwacheraufruf wird ein Aufruf für die Blattschreiberausgabe beschrieben. Es soll auf dem Blattschreiber mit der Nummer 0 ein Text ausgegeben werden, der ab der Zelle ADR im Arbeitsspeicher steht. (Der Text ist durch ein Endezeichen abgeschlossen, das die Ausgabe begrenzt).

Der Überwacheraufruf, der als Makroaufruf (MA) im Quellprogramm geschrieben ist, hat folgende Form:

```
MA   BSAU = 0 , ADR
```

Der Übersetzer baut dafür folgende Befehlsfolge in das Programm ein:

	Befehlsteil	Adreßteil		
	UNT'	(5)	Umschaltbefehl	
	003	19	Steuerwort	19 : Blattschreiberausg. 003 : Parameterblock umfaßt 3 Worte
Parameterblock	NOP	0	1. Parameter	Blattschreibernummer
	NOP	ADR	2. Parameter	Textadresse
	NOP	0	Anzeigenzelle	

Es handelt sich bei dem Aufruf um einen langen Überwacheraufruf.

Für die Überwacheraufrufe wird im folgenden nicht mehr die Darstellung des Quellprogrammes verwendet, sondern die Aufrufe werden als Prozeduraufrufe geschrieben. Der oben beschriebene Aufruf für die Blattschreiberausgabe hat dann die Form:

```
BSAU ( 0,ADR )
```

In der Klammer sind die Parameter des Aufrufs, nämlich die Blattschreibernummer und die Textadresse angegeben.

Die gesamte Bearbeitung von Überwacheraufrufen im Überwacher gliedert sich in zwei Teile, die Aufrufvorverarbeitung und die eigentliche Ausführung des Aufrufes. Diese beiden Teile werden in den folgenden Abschnitten 3.2.3 und 3.2.4 behandelt.

3.2.3 Vorverarbeitung von Überwacheraufrufen

Die Aufrufvorverarbeitung hat die Aufgabe die Inhalte der Akkumulatoren zu retten, die Fortsetzadresse der Task zu bestimmen und in einen der Task zugeordneten Speicherbereich (Taskkontrollblock) einzutragen (vgl. Abschnitt 3.3.3.2) und die entsprechende Bearbeitungsroutine für die Ausführung des Aufrufes zu aktivieren.

Die Bearbeitungsroutine wird bei Aufrufen ohne Steuerwort durch die Zahl im Adressteil des Umschaltbefehls bestimmt, und bei Aufrufen mit Steuerwort durch das Steuerwort.

Das Steuerwort hat folgende Struktur:

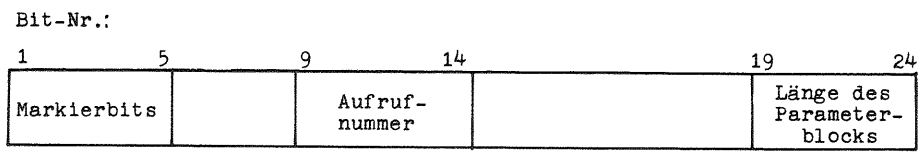


Bild 3.4: Aufbau des Steuerwortes eines Überwacheraufrufes

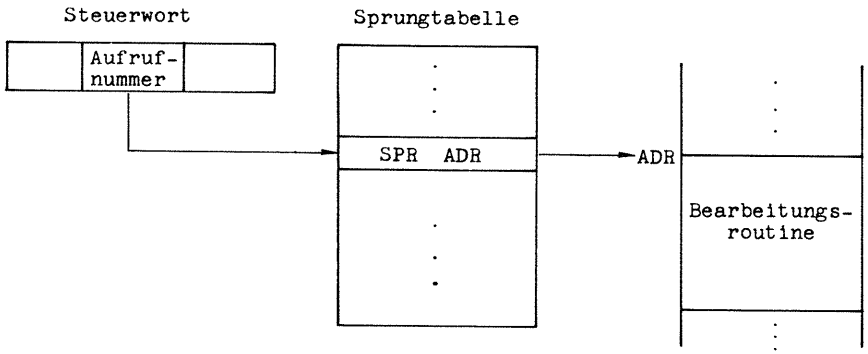
Die Aufrufnummer (Bit 9-14) bestimmt eine Menge von funktionell zusammengehörenden Aufrufen; (z.B. sind die Aufrufe für die Uhr zum Übernehmen des Datums oder der Uhrzeit durch die Aufrufnummer 28 gekennzeichnet). Die Unterscheidung der einzelnen Funktionen innerhalb der Menge wird durch die Markierbits (Bit 1-5) vorgenommen.

Die Bits 19-24 geben die Länge des Parameterblocks an. Mit dieser Längenangabe kann die Fortsetzadresse der Task bestimmt werden.

(Die Fortsetzadresse verweist auf den Befehl unmittelbar hinter dem Überwacheraufruf).

Bei Aufrufen ohne Steuerwort wird die Aufrufnummer aus dem Umschaltbefehl entnommen. Die Bestimmung der Bearbeitungsroutine für die durchzuführende Funktion erfolgt mittels einer Sprungtabelle(Bild 3.5).

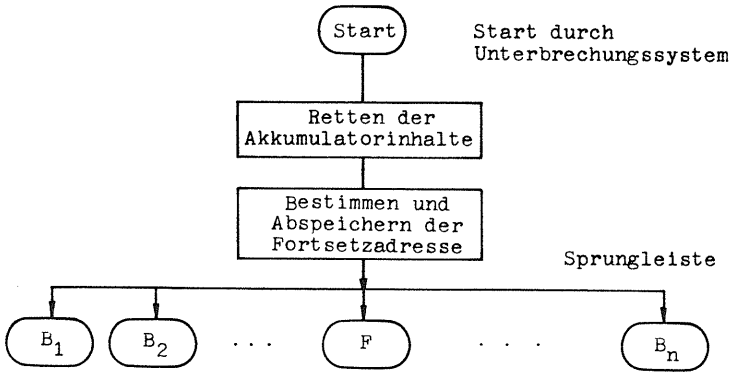
In der Sprungtabelle sind die Startadressen für die Bearbeitungsroutinen eingetragen. Der Zugang zur Sprungtabelle erfolgt mit der Aufrufnummer. Wenn ein bestimmter Aufruf nicht realisiert ist, so steht in der Sprungtabelle ein Sprung zu der Fehlerbearbeitungsroutine für Überwacheraufrufe (s. Abschnitt 3.6.4).



SPR ADR Sprungbefehl zur Bearbeitungsroutine mit der Anfangsadresse ADR

Bild 3.5: Bestimmung der Bearbeitungsroutine für einen Überwacheraufruf

Die Markierbits werden in der Bearbeitungsroutine ausgewertet. Die Abläufe bei der Aufrufvorverarbeitung sind in folgendem Flußdiagramm kurz zusammengefaßt.



B_i Bearbeitungsroutine i
F Fehlerbearbeitungsroutine

Bild 3.6: Flußdiagramm für die Vorverarbeitung von Überwacheraufrufen

Die Funktion der Fehlerbearbeitungsroutine wird bei den Sicherungsfunktionen (Abschnitt 3.6.4) beschrieben.

3.2.4 Ausführung von Überwacheraufrufen

Bei der Ausführung der Aufrufe müssen zwei Fälle unterschieden werden, nämlich Funktionen, die sofort ausgeführt werden können (kurze Aufrufe) und Funktionen, die über Warteschlangen abgearbeitet werden (lange Aufrufe) (s. Abschnitt 3.2.2).

Die Ausführung der verschiedenen Funktionen wird im folgenden nur im Prinzip beschrieben; die Funktionen selbst werden bei der Beschreibung der einzelnen Funktionsblöcke des Überwachers behandelt (Abschnitte 3.3 bis 3.5).

3.2.4.1 Kurze Aufrufe

Der Ablauf bei der Bearbeitung eines kurzen Aufrufes ist in folgendem Flußdiagramm angegeben:

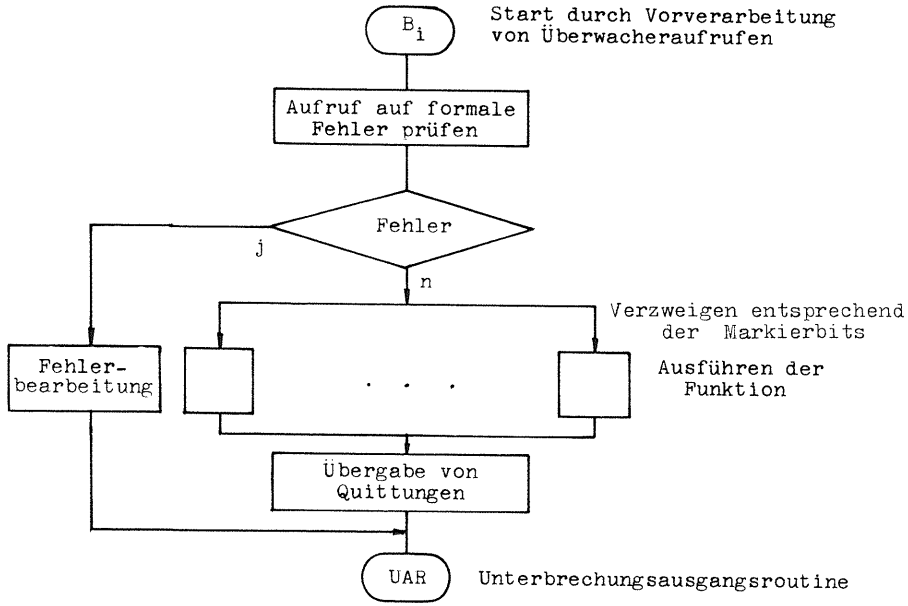


Bild 3.7: Bearbeitung kurzer Überwacheraufrufe

Zuerst wird der Aufruf auf formale Fehler geprüft, z.B. ob die Markierbits zugelassen sind und ob die angegebenen Parameter richtig sind. Falls der Aufruf fehlerhaft ist, wird die Fehlerbearbeitungsroutine aktiviert, und zur Unterbrechungsausgangsroutine verzweigt. Ansonsten wird entsprechend den Markierbits zu deren entsprechendem Bearbeitungsteil verzweigt.

Nach Ausführung der Funktion im Bearbeitungsteil kann der aufrufenden Task eine Ausführungsquittung übergeben werden. Diese Quittung wird bei Aufrufen mit Anzeigenzelle im Parameterblock in diese Zelle eingetragen und bei Aufrufen ohne Anzeigenzelle in den Speicherplatz für den Akkumulator im Taskkontrollblock.

Nach Beendigung der Bearbeitungsroutine wird die Unterbrechungsausgangsroutine angesprungen.

3.2.4.2 Lange Aufrufe

Bei der Bearbeitung langer Aufrufe kann die Funktion nicht sofort ausgeführt werden. Es muß zur Ausführung auf das Eintreffen eines Ereignisses gewartet werden.

Man unterscheidet Ereignisse, die direkt Folge einer Operation sind, und Ereignisse, die durch andere Vorgänge ausgelöst werden. Der erste Typ wird repräsentiert durch die Rückmeldungen externer Geräte, die einen Schreib-, Lese- oder Steuerauftrag (z.B. Positionieren der Schreib-Leseköpfe des Plattenspeichers) erhalten haben und nun den Abschluss der Operation melden. Zum zweiten Typ zählen alle Ereignisse, die durch andere Tasks ausgelöst werden können (z.B. Senden einer Nachricht an eine andere Task) oder durch Unterbrechungssignale eines Taktgebers. Der prinzipielle Ablauf für die Bearbeitung ist in folgenden Bild 3.8 dargestellt.

Die Bearbeitung langer Überwacheraufrufe gliedert sich in zwei Teile(im Bild 3.8 durch den linken und rechten Teil des Flußdiagrammes dargestellt). Der linke Teil kommt zur Ausführung, wenn der Aufruf von der Task abgegeben wird. Dabei wird der Aufruf in die Warteschlange für diesen Aufruftyp eingereiht und falls eine Operation eingeleitet werden kann, wird dies durch die Start-Routine durchgeführt. Danach wird auf das Eintreffen des für die Fortsetzung notwendigen Ereignisses gewartet. Z. B. wird bei Ein-Ausgabeoperationen, wenn noch kein Eintrag in der Warteschlange vorhanden ist, die Operation in dem betreffenden Gerät angestoßen.

Der zweite Teil der Bearbeitung wird durchgeführt, wenn das erwartete Ereignis eintrifft. Bei Ein-Ausgabeoperationen ist es das Unterbrechungssignal (Kanalanforderung), das das Ende der Operation signalisiert, und bei anderen Aufrufen kann es durch einen Überwacheraufruf von einer anderen Task signalisiert werden. Nach Eintreffen des Ereignisses wird der Aufruf fortgesetzt (z.B. bei einer blockweisen Ausgabe, bei der mit einem Aufruf mehrere Blöcke ausgegeben werden sollen, und das Gerät aber nur einen Block ausgeben kann) oder der Aufruf wird beendet und aus der Warteschlange ausgetragen. Falls in der Warteschlange noch wartende Aufrufe eingetragen sind, kann der nächste Aufruf mit der Start-Routine bearbeitet werden.

Wie aus dem Bild 3.8 ersichtlich ist, sind für die Ausführung eines Aufrufes zwei Eingänge (B_i und E_i) in die Bearbeitungsroutine erforderlich.

Eine ausführlichere Beschreibung eines langen Überwacheraufrufes erfolgt im Abschnitt 3.4 für die Ein-Ausgabe über Standardperipherie.

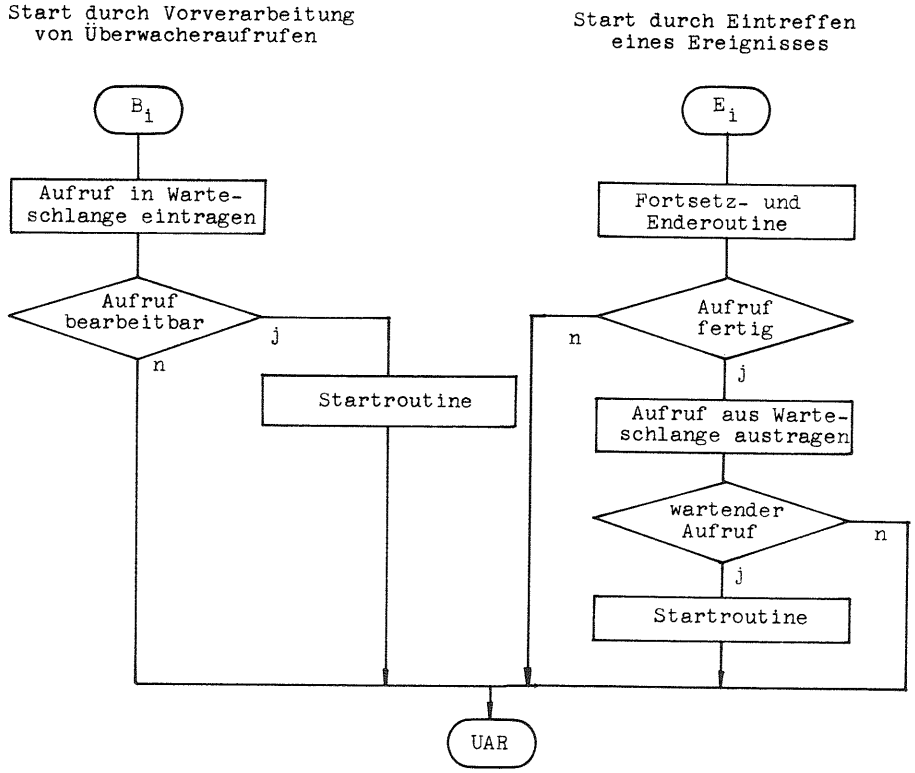


Bild 3.8: Bearbeitung langer Überwacheraufrufe

3.2.5 Aufruf der Überwacherfunktionen im Überwacher

Systemtasks werden wie Anwendertasks im Überwacher verwaltet. Der Unterschied zwischen den beiden Taskarten besteht darin, dass im Gegensatz zu den Anwendertasks, die im Normalzustand des Rechners ablaufen, die Systemtasks im Privilegierten Zustand ablaufen.

Der Zugang zum Überwacher von einer Anwendertask aus erfolgt über den Umschaltbefehl. Der Umschaltbefehl für die Überwacheraufrufe wird aber nur im Normalzustand wirksam. Damit nun auch von Systemtasks die Überwacherfunktionen aufgerufen werden können, wurde die Möglichkeit geschaffen, die Funktionen einfach über einen Unterprogrammsprung aufzurufen.

Der Überwacheraufruf in einer Systemtask besteht also aus einem Unterprogrammsprung. Das Steuerwort und der Parameterblock stehen wie bei den Anwendertasks hinter dem Unterprogrammsprung.

Außer durch Systemtasks ist es auch möglich, durch Bearbeitungsroutinen, die nicht zu den Bearbeitungsroutinen für Überwacheraufrufe zählen (z.B. die Sicherungsfunktionen, vgl. Bild 3.1), die Überwacherfunktionen aufzurufen. Der Aufruf erfolgt wie bei den Systemtasks über einen Unterprogrammsprung.

Beim Aufruf einer Funktion (z.B. zum Starten einer Anwendertask) durch eine Bearbeitungsroutine des Überwachers muß sich der Rechner stets im Privilegierten Zustand befinden, weil diese Bearbeitungsroutinen für die Adressierungsmodi des Privilegierten Zustandes geschrieben sind. Aus diesem Grund muß in der Taktbearbeitungsroutine, welche im "Alarmzustand" zur Ausführung kommt (s. Abschnitt 3.5.2.2), vor einem Überwacheraufruf der Rechner in den Privilegierten Zustand umgeschaltet werden (s. Abschnitt 3.10.2).

Die Aufrufe von Überwacherfunktionen in den Bearbeitungsroutinen des Überwachers werden im folgenden als "systeminterne" Überwacheraufrufe bezeichnet, im Gegensatz zu den "systemexternen" Überwacheraufrufen in den Anwendertasks.

3.3 Ablauforganisation

3.3.1 Allgemeines

Alle im Rechner zur Ausführung gelangenden Programme werden vom Überwacher als Tasks verwaltet. Jede Task kann sich in verschiedenen "Zuständen" befinden; z.B. sie belegt gerade den Prozessor oder sie wartet auf das Ende einer Ein-Ausgabeoperation.

Das Verweilen einer Task in einem bestimmten Zustand und die möglichen Übergänge zu anderen Zuständen können in einem sogenannten ZUSTANDSMODELL anschaulich beschrieben werden. In unserem Fall gelten für dieses - anschließend behandelte - Zustandsmodell folgende Randbedingungen:

a) Alle Tasks sind arbeitsspeicherresident, d.h. alle Programme der in der Ablauforganisation registrierten Tasks befinden sich im Arbeitsspeicher.

   Dies ist zweckmäßig weil die meisten Programme aus zeitlichen Gründen nicht erst vor ihrer Ausführung in den Arbeitsspeicher geladen werden können, d.h. ohnehin arbeitsspeicherresident sein müssen.Die Programme, die bei Bedarf in den Arbeitsspeicher geladen werden (z.B. Routineprüfprogramme),werden, wenn sie aktiviert sind, nur durch arbeitsspeicherresidente Programme unterbrochen. Sie brauchen daher nicht aus dem Arbeitsspeicher ausgelagert zu werden (vgl. Abschnitt 2.1.2).

   Damit entfallen in der Ablauforganisation die Probleme des Ein- und Auslagerns der zu den Tasks gehörenden Programme auf den Hintergrundspeicher.

b) Die Tasks besitzen statische Prioritäten für die Zuteilung des Prozessors, die sich während der Lebensdauer einer Task nicht mehr ändern können.

   Die statischen Prioritäten reichen aus, um die Aufgaben entsprechend ihrer Dringlichkeit ausführen zu können. Die Einführung dynamischer Prioritäten /20/ würde nur eine aufwendigere Ablauforganisation zur Folge haben.

c) Die Hardwarebetriebsmittel (Ein-Ausgabegeräte) werden den Tasks bei der Ausführung der Ein-Ausgabeoperation zugeordnet. Sie können nicht für bestimmte Tasks reserviert werden.

   Dies ist zweckmäßig, da der Rechner nur als Vermittlungsrechner benutzt wird, und das Vermittlungspersonal die Programme, die Ein-Ausgabegeräte benutzen, so zur Ausführung bringen kann, daß eine Belegung eines Gerätes für unterschiedliche Aufgaben ausgeschlossen wird. Die Betriebsmittelverwaltung wird damit dem Vermittlungspersonal übertragen.

3.3.2 Das Zustandsmodell

Eine Task kann verschiedene Zustände annehmen, welche durch Überwacheraufrufe verändert werden können /21/.

In der vorliegenden Arbeit kann eine Task nur die vier Zustände bekannt, bereit, aktiv und blockiert annehmen. In der Literatur /16/ wird oft eine Aufspaltung in weitere Zustände vorgenommen, die jedoch hier für die Beschreibung der Abläufe nicht notwendig ist.

Die vier Zustände sind folgendermaßen definiert:

Zustand bekannt:

Die Task ist angemeldet, d.h. das zur Task gehörende Programm befindet sich im Arbeitsspeicher, und die Task wird in den Listen der Ablauforganisation geführt.

Zustand bereit:

Die Task ist rechenfähig und wartet auf die Zuteilung des Prozessors.

Zustand aktiv:

Die Task belegt gerade den Prozessor, d. h. sie läuft gerade.

Zustand blockiert:

Die Task wartet auf das Eintreffen eines Ereignisses (z.B. auf das Ende einer Ein-Ausgabeoperation, das durch ein Unterbrechungssignal (E/A-Kanalanforderung) gemeldet wird).

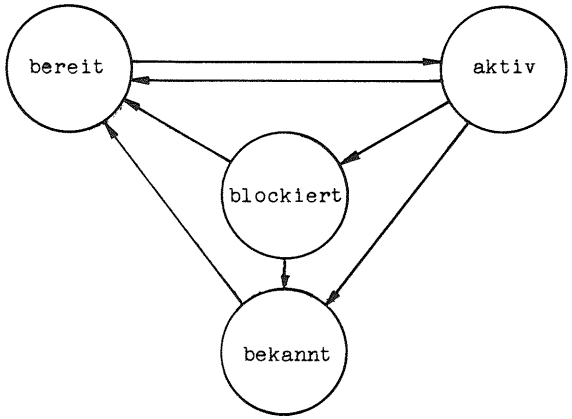


Bild 3.9: Zustandsmodell für Tasks

Der Übergang einer Task von einem Zustand in einen anderen Zustand (Bild 3.9) kann entweder durch Überwacheraufrufe von Tasks selbst oder durch Überwacherinterne Vorgänge ausgelöst werden.

Da zur Beschreibung der Zustandsübergänge die Kenntnis der sie auslösenden Überwacheraufrufe notwendig ist, werden die verschiedenen Übergänge an dieser Stelle nur im Prinzip erläutert. (Eine detailierte Beschreibung erfolgt in den Abschnitten 3.3.3 , 3.3.4 und 3.3.5 bei der Beschreibung der einzelnen Überwacheraufrufe.)

Der Übergang einer Task von einem Zustand in einen anderen Zustand (Bild 3.9) kann aus verschiedenen Anlässen erfolgen:

a) Die Task Nr.i ist "aktiv". Im verlauf der Task-eigenen Befehlsfolge ergeht ein Aufruf an den Überwacher. Dadurch wird der weitere Befehlsablauf der Task Nr.i unterbrochen und die Task Nr.i geht von "aktiv" über nach "blockiert" oder "bekannt".

   Sind weitere Tasks, welche den Prozessor wünschen, d.h. im Zustand "bereit" sind, vorhanden, so erhält jene Task Nr.j, welche die höchste Priorität hat, den Prozessor.(Übergang vom Zustand "bereit" nach "aktiv").

b) Die betrachtete Task Nr.i ist "bereit". Eine andere, derzeit aktive Task Nr.j veranlaßt durch einen Überwacheraufruf ihren eigenen Übergang nach "bekannt" oder "blockiert". Dann wird der Überwacher die Task Nr.i wieder aktivieren, sofern nicht andere Tasks höherer Priorität sich ebenfalls im Zustand "bereit" befinden.

c) Die Task Nr.i ist "blockiert", d.h. sie ging zu einem früheren Zeitpunkt von "aktiv" nach "blockiert" weil die Fortsetzung ihrer im Prozessor laufenden Befehlsfolge wegen fehlender Daten, die z.B. über den Lochkartenleser eingelesen werden sollen, nicht möglich war. Der Überwacher erkennt das Ende der Lochkarteneingabe (vgl. Abschnitt 3.4) und führt die Task Nr.i vom Zustand "blockiert" in den Zustand "bereit" über.

   Ist der Prozessor gerade von einer Task niederer Priorität belegt, so wird die Task Nr.i sofort aktiviert, ansonsten verläuft der weitere Zustandsübergang nach b).

d) Die Task Nr.i ist bekannt, d.h. das Programm befindet sich im Arbeitsspeicher und die Task ist in den Listen des Überwachers geführt. Die Task wird aber momentan nicht benötigt.

Jetzt werde die Task Nr.i (d.h. eine bestimmte Befehlsfolge, die zur Task Nr.i gehört) benötigt. Dann wird sie von einer Task Nr.j aus mit einem Überwacheraufruf nach dem Zustand "bereit" überführt. Der weitere Zustandsübergang verläuft analog zu c), wenn eine blockierte Task nach bereit übergeht.

3.3.3 Taskverwaltung

3.3.3.1 Allgemeines

Die Taskverwaltung wird durch den Überwacher durchgeführt.Sie hat die Aufgabe, den Ablauf der Tasks zu steuern und die für die Prozessorzuteilung notwendigen Informationen bereitzustellen.
Die Prozessorzuteilung, d.h. die Auswahl der Task mit der höchsten Priorität und das Laden der Register des Leitwerks mit der zur Task gehörenden Information, erfolgt in der Unterbrechungsausgangsroutine (s. Abschnitt 3.10).

Die Prioritäten der einzelnen Tasks werden in dem hier entwickelten Programmsystem durch Nummern geregelt. Jeder Task ist eine Nummer zugeordnet, die zugleich die Priorität der Task angibt. Dabei hat die Task mit der niedersten Nummer die höchste Priorität.

Für diese Taskverwaltung benötigt der Überwacher eine Anzahl von Listen, deren Zweck in den folgenden Abschnitten an geeigneter Stelle behandelt wird. Es sind dies:

Taskwarteschlange	1 Liste	
Taskkontrollblock	max. 14 Listen	
Semaphorwarteschlange	6 Listen	(auch mehr möglich)
Empfängerwarteschlange	1 Liste	
Nachrichtenwarteschlange	"	
Ein-Ausgabewarteschlange	"	je E/A-Gerät
Ladewarteschlange	"	

3.3.3.2 Taskwarteschlange und Taskkontrollblock

Alle auf die Prozessorzuteilung wartenden ("bereiten") Tasks sind in die Taskwarteschlange (TWS, Bild 3.10) eingetragen. Die TWS ist als bitweise organisierte Liste realisiert (vgl. Anhang A1.2). Der Eintrag einer Task in die TWS erfolgt durch Setzen des entsprechenden Bits und der Austrag einer Task durch Löschen dieses Bits.

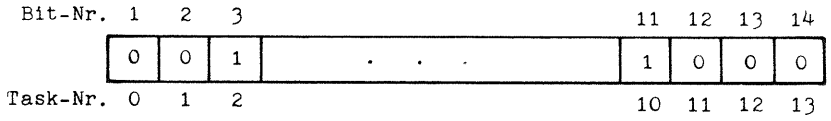


Bild 3.10: Taskwarteschlange (TWS)

Wenn das individuelle Bit einer Task in der TWS auf 1 gesetzt ist, bedeutet dies, daß diese Task auf die Prozessorzuteilung wartet oder bereits den Prozessor belegt hat. Das Bit mit der niedersten Nummer weist auf jene Task, die den Prozessor momentan belegt. Im Beispiel von Bild 3.10 belegt die Task Nummer 2 den Prozessor (Zustand aktiv) und die Task Nummer 10 wartet auf die Prozessorzuteilung (Zustand bereit).

Bei der Prozessorzuteilung an eine Task müssen die Register des Prozessors mit bestimmten Informationen von dieser Task geladen werden. Diese Informationen sind im TASKKONTROLLBLOCK gespeichert, der als Element einer blockweise organisierten (zusammenhängenden) Liste realisiert ist (vgl. Anhang A1.2). Der Taskkontrollblock (TKB) und seine Inhalte sind im Bild 3.11 dargestellt.

Zelle		
1	BAR 9 statisch	evtl. Taskverkettung
2	BAR 15 statisch	evtl. Taskverkettung
3	BAR 9 dynamisch	——
4	BAR 15 dynamisch	——
5	Befehlszählerstand	Kennbits der Akk.
6	Inhalt linker Akkumulator (Akk)	
7	Inhalt rechter Akkumulator	
8	Exponentenregister	Mantissenverlängerung
9	Zahl der Eintragungen in Warteschlangen	
10	Startadresse	

Bild 3.11 Taskkontrollblock (TKB)

Der individuelle Taskkontrollblock jeder Task umfaßt 10 Arbeitsspeicherzellen. Die ersten vier Zellen enthalten die statischen und dynamischen Inhalte der Basisadressenregister (BAR). (Die Nummern 9 und 15 der BAR sind nur durch die spezielle Organisation des verwendeten Rechners S306 bedingt,vgl. Anhang A3.) Die statischen Inhalte werden der Task von der Speicherverwaltung beim Anmelden zugeordnet. Sie sind gegeben durch die Lage jenes Programmes im Arbeitsspeicher, das zu der betreffenden Task gehört. Die dynamischen Inhalte werden beim Starten einer Task zunächst gleich den statischen gesetzt. Sie können jedoch,während die Task ausgeführt wird, von der Task verändert werden.

Falls eine unterbrochene Task wieder aktiviert wird, werden die zwei BAR des Leitwerks mit den Inhalten der Zellen 3 und 4 des TKB geladen. Die statischen Inhalte der Zellen 1 und 2 sind notwendig, damit, nachdem sich die Task beendet hat, ein späteres erneutes Starten mit den ursprünglichen Inhalten der BAR möglich ist. Die Zellen 1 und 2 können außerdem eine Verkettung zwischen den Tasks enthalten, wenn ein Programm oder Teile davon mehreren Tasks gemeinsam ist.

Die 5. Zelle enthält den Befehlszählerstand bei dem die Task unterbrochen wurde. Sie enthält weiterhin die zu den Akkumulatoren gehörenden Überlaufanzeigen.

Die Zellen 6 bis 8 enthalten die Inhalte des linken Akkumulators, des rechten Akkumulators mit Mantissenverlängerung sowie den Inhalt des Exponentenregisters.

Die 9. Zelle enthält einen Zähler, der angibt, wieviele Aufrufe an externe Geräte von dieser Task abgegeben wurden und noch unerledigt sind. (Die Zahl der unerledigten Aufrufe ist im Zustand"bekannt" einer Task wichtig, damit ein Abmelden der Task vor Erledigung aller Aufrufe verhindert werden kann.)

Beim Starten einer Task muß der Befehlszähler auf die Startadresse dieser Task gesetzt werden. Er steht in der 10. Zelle des TKB.

3.3.3.3 Funktionen der Taskverwaltung

Um eine Task in die Listen des Überwachers aufzunehmen oder dort zu streichen und um den Taskablauf zu steuern,stehen verschiedene Aufrufe an den Überwacher zur Verfügung, die nachstehend behandelt werden.Die durch die Überwacheraufrufe verursachten Zustandsübergänge der Tasks sind im Bild 3.12 dargestellt.

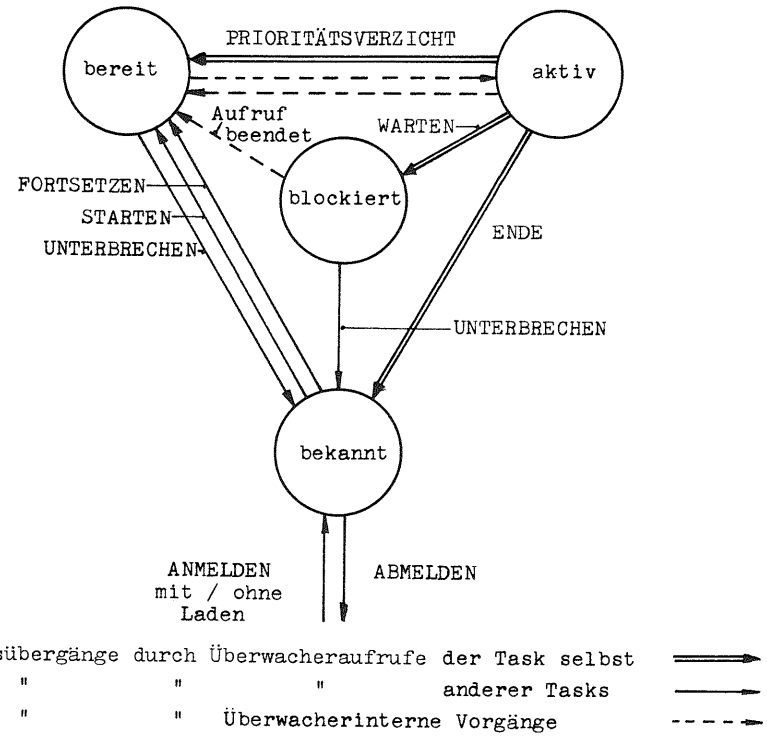


Bild 3.12: Zustandsübergänge von Tasks durch Funktionen der Taskverwaltung

Die Aufnahme einer Task in die Listen des Überwachers geschieht mit dem Aufruf ANMELDEN. Dabei kann ein Programmstück, das bereits im Arbeitsspeicher steht, als Task angemeldet werden, oder es kann auch das Laden des unter der Tasknummer zur Ausführung gelangenden Programmes von einem externen Speichermedium veranlaßt werden.

Das Streichen einer Task in den Listen, um danach ein anderes Programm unter dieser Tasknummer anmelden zu können, wird durch den Aufruf ABMELDEN veranlaßt.

Nachdem eine Task in den Listen des Überwachers registriert ist ("bekannt"), kann sie mit dem Aufruf STARTEN aktiviert werden. Sie wird damit in den Zustand "bereit" versetzt, und kann bei der Prozessorvergabe durch den Überwacher den Prozessor zugeteilt bekommen (Zustand "aktiv"). Eine Task meldet dem Überwacher mit dem Aufruf ENDE daß sie fertig ist. Sie geht damit in den Zustand "bekannt" über.

Mit dem Aufruf UNTERBRECHEN ist es möglich, eine Task vom Zustand "bereit" oder "blockiert" in den Zustand "bekannt" zu versetzen. Dadurch kann eine Task gezielt unterbrochen werden und zu einem späteren Zeitpunkt mit dem Aufruf FORTSETZEN wieder fortgesetzt werden (s. Abschnitt 4.2).

Bei der Bearbeitung "langer Überwacheraufrufe" (s. Abschnitt 3.2.4.2) kann eine Uberwacherfunktion nicht sofort ausgeführt werden. Z.B. wird bei Ein-Ausgabeoperationen diese Operation angestoßen und das Ein-Ausgabegerät meldet dann deren Abschluß durch ein Unterbrechungssignal (s. Abschnitt 3.4.3).

Wegen der im Verhältnis zur Zentraleinheit langsamen Operationsgeschwindigkeit der Ein-Ausgabegeräte ist die E/A-Operation mit Sicherheit noch nicht beendet, wenn die Task unmittelbar nach dem Anstoßen der E/A-Operation fortgesetzt wird. Der Aufruf WARTEN bietet nun die Möglichkeit , z.B. das Ende der Ein-Ausgabeoperation abzuwarten bevor diese Task weiterläuft. (Dies ist z.B. notwendig bei Daten, die über den Lochkartenleser eingelesen und bearbeitet werden sollen. Erst wenn diese Daten in den dafür reservierten Arbeitsspeicherzellen stehen, kann die Task diese Daten verarbeiten.)

Mit dem Aufruf PRIORITAETSVERZICHT kann eine Task bei der Prozessorvergabe durch den Überwacher (einmalig) auf die Zuteilung des Prozessors verzichten. Eine Task höherer Priorität kann dadurch - ohne ENDEmeldung an den Überwacher - einer Task niederer Priorität, die "bereit" ist, den Prozessor "ausleihen" solange sie selbst ihn (kurzzeitig) nicht benötigt (s. Abschnitt 4.5.4).

Die verschiedenen Überwacheraufrufe für die Funktionen der Taskverwaltung sind im folgenden zusammengestellt:

ANMELDEN mit Laden (Nummer, Speichermedium):

Der Taskkontrollblock, der zur Task "Nummer" gehört, wird belegt. An den Lader (s. Abschnitt 3.11) wird von der Taskverwaltung der Auftrag gegeben, von dem im Aufruf angegebenen externen Speichermedium das Programm in den Arbeitsspeicher zu laden. Der Lader ist als Systemtask realisiert. Er hat neben dem Laden des Programmes in den Arbeitsspeicher auch die Funktion der Arbeitsspeicherverwaltung. Nach erfolgreichem Laden trägt der Lader in den TKB die Werte der statischen Basisadressenregister ein (Lage des Programmes im Arbeitsspeicher). Die Startadresse im TKB wird auf Null gesetzt (vgl. Abschnitt 3.3.3.2).

Der Aufruf ANMELDEN mit Laden kann nicht sofort komplett ausgeführt werden, da der Lader über ein externes Gerät das Programm einlesen muß. Deshalb wird der Aufruf in eine Warteschlange für den Lader eingereiht. Falls es sich bei diesem Aufruf um den ersten Eintrag in der Warteschlange handelt, wird der Lader aktiviert. Die Aktivierung des Laders erfolgt durch Setzen seines Bits in der Taskwarteschlange.

Beim Laden eines Programmes vom Magnetplattenspeicher muß zusätzlich die Datei angegeben werden, in der das Programm abgelegt ist.

ANMELDEN ohne Laden (Nummer, Startadresse):

Der Überwacheraufruf ANMELDEN ohne Laden ermöglicht es, eine Befehlsfolge, die im Programm der aufrufenden Task steht, als eine neue Task im Uberwacher zu registrieren.

Der Taskkontrollblock, der zur Task "Nummer" gehört, wird belegt. Dabei werden von jener Task, welche die Funktion aufruft, die Inhalte der dynamischen BAR übernommen und für die neue Task als statische BAR in den TKB eingetragen. Als Startadresse wird im TKB die im Überwacheraufruf angegebene Startadresse eingetragen. Weiterhin wird eine Verkettung zur aufrufenden Task eingetragen, da diese den Arbeitsspeicher für das dazugehörende Programm belegt.

ABMELDEN (Nummer):

Der Taskkontrollblock der Task "Nummer" wird gelöcht und falls die Task Speicherplatz belegt hat wird dieser freigegeben. Die Task ist damit in der Verwaltung des Überwachers gestrichen.

STARTEN (Nummer):

Die Task "Nummer" , die im Zustand "bekannt" ist, wird in die Taskwarteschlange eingetragen (Setzen des entsprechenden Bits) und der Befehlszähler im TKB wird gleich der Startadresse gesetzt. Die Task ist damit im Zustand "bereit" und wartet auf die Prozessorzuteilung.

ENDE:

Die aktive Task wird aus der Taskwarteschlange ausgetragen (Löschen des entsprechenden Bits). Die Task ist damit im Zustand "bekannt".

UNTERBRECHEN (Nummer):

Die Task "Nummer" wird, falls sie im Zustand "bereit" ist, aus der Taskwarteschlange ausgetragen. Wenn sie sich dagegen im Zustand "blockiert" befindet, d.h. sie ist in einer Warteschlange eingetragen

und wartet auf ein Ereignis (s. Abschnitte 3.3.4, 3.3.5 und 3.4), dann wird sie aus dieser Warteschlange ausgetragen. (Wenn das Ereignis eintrifft, auf welches die betreffende Task gewartet hatte, so wird dieses entweder vom Überwacher selbst bearbeitet oder es wird in eine Warteschlange eingetragen, bis es von der betreffenden Task selbst, nach einem späteren Fortsetzen, bearbeitet werden kann.)

Nach Ausführung des Aufrufes befindet sich die Task "Nummer" im Zustand "bekannt".

FORTSETZEN (Nummer):

Die Task "Nummer" wird wieder in die Taskwarteschlange eingetragen. (Setzen des entsprechenden Bits). Dadurch ist die Task "Nummer" wieder im Zustand "bereit". Im Gegensatz zum Überwacheraufruf STARTEN wird bei dem Aufruf FORTSETZEN der Befehlszähler im TKB nicht verändert, d.h. die Task wird dort fortgesetzt, wo sie zuvor unterbrochen wurde.

WARTEN (Aufruf):

Die aktive Task wird aus der Taskwarteschlange ausgetragen und in der Warteschlange für den Überwacheraufruf "Aufruf" wird vermerkt, daß diese Task auf das Ende eines "langen Überwacheraufrufes" wartet (s.Abschnitt 3.4.4). Die Task ist damit im Zustand "blockiert" bis das Ereignis, auf das die Task wartet, eingetroffen ist.

PRIORITAETSVERZICHT:

Die aktive Task geht vom Zustand "aktiv" in den Zustand "bereit" über.

Neben den oben genannten Überwacherfunktionen gibt es noch weitere für die BAR-Verwaltung. Diese Funktionen, die im einzelnen nicht aufgeführt werden, erlauben die dynamischen Inhalte der BAR im Taskkontrollblock zu verändern. Dadurch ist es möglich, auch aus Programmen anderer Tasks Informationen zu lesen oder dorthin zu schreiben. Die Kopplung der Tasks erfolgt über die BAR.

3.3.4 Synchronisation

Verschiedene Tasks können gleichzeitig Zugriff zu einer gemeinsamen Datenzelle (z.B. einem Zähler) wünschen oder sie wollen gleichzeitig dieselbe Prozedur verwenden.

Um einen gleichzeitigen Zugriff zu der Datenzelle oder zu der Prozedur auszuschließen, muß der Zugriff "synchronisiert" werden. Dies kann mit Hilfe von sogenannten "Semaphoren" /22/ erreicht werden. Ein Semaphor stellt einen Zähler dar, der durch eine sogenannte P-Operation bzw. V-Operation verändert werden kann. Diese Veränderungen werden vom Überwacher ausgeführt, der hierzu eines speziellen Aufrufes durch die aktive Task bedarf.

Durch die P-Operation P(Sem) wird der Wert des Semaphors Sem um 1 erniedrigt, falls er danach noch positiv ist. Würde aber der Wert des Semaphors negativ, d.h. er hatte vorher bereits den Wert Null, so bedeutet dies, daß bereits eine andere Task den gleichen Zugriff ausgeführt hat. Deshalb wird in diesem Fall die betrachtete Task inaktiviert, d.h. sie wird aus der Taskwarteschlange ausgetragen und statt dessen in eine spezielle Warteschlange für diesen Semaphor eingetragen. Der Wert des Semaphors selbst wird dabei also nicht verändert.

Umgekehrt wird nach einem erledigten Zugriff einer Task mit der V-Operation der Wert des Semaphors vom Überwacher wieder auf 1 gesetzt. Außerdem wird vom Überwacher geprüft, ob weitere Tasks in der Warteschlange des Semaphors warten. Wenn ja, dann wird eine der wartenden Tasks aus der Warteschlange für den Semaphor ausgetragen und wieder in die Taskwarteschlange eingetragen. Sobald sie wieder fortgesetzt wird kommt dann die P-Operation, bei der sie zuvor abgewiesen wurde, erneut zur Ausführung.

Ein Semaphor nach obigem Prinzip kann nur die Werte 0 und 1 annehmen und wird als binärer Semaphor bezeichnet.

Die P-Operation wird in der vorliegenden Arbeit als SPERREN und die V-Operation als FREIGEBEN bezeichnet.

Die Warteschlange für einen binären Semaphor hat folgende Struktur:

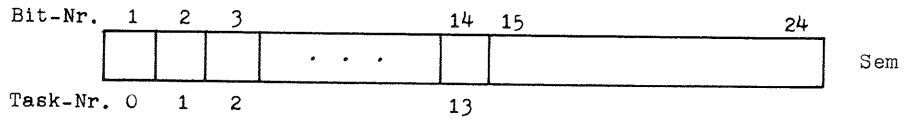
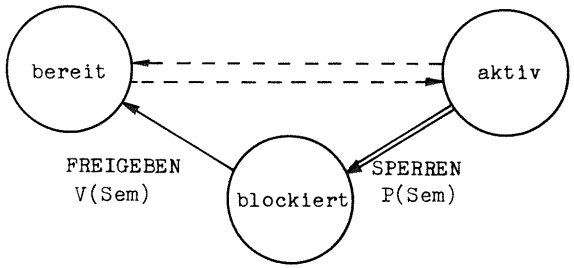


Bild 3.13: Warteschlange für einen binären Semaphor (Sem)

In den Bits 1 bis 14 sind die wartenden Tasks entsprechend ihrer Nummer gekennzeichnet und in Bit 15 bis 24 ist die Nummer jener Task eingetragen, die gerade den Semaphor belegt. Wenn der Semaphor Sem den Wert 1 hat, also "frei" anzeigt, so sind alle Bits in der Warteschlange auf Null.

Durch die beiden Semaphor-Operationen SPERREN(Sem) und FREIGEBEN(Sem), die der Synchronisation dienen, kann der Zustand einer Task verändert werden. Die möglichen Zustandswechsel sind in folgendem Bild 3.14 angegeben.



Zustandsübergänge durch Überwacheraufrufe der Task selbst ⟹
" " " anderer Tasks ⟶
" " Überwacherinterne Vorgänge - - - ⟶

Bild 3.14: Zustandsübergänge von Tasks durch Synchronisationsfunktionen

Falls eine Task einen bereits gesperrten Semaphor (Wert des Semaphors ist Null) zu ihren Gunsten sperren will, geht sie vom Zustand "aktiv" in den Zustand "blockiert" über. Sie kommt erst wieder in den Zustand "bereit" zurück, wenn der Semaphor durch jene andere Task, die gesperrt hatte, wieder freigegeben wurde.

3.3.5 Kommunikation

Für den Informationsaustausch zwischen verschiedenen Tasks muß die Möglichkeit bestehen, von einer Task aus eine Nachricht abzusenden und diese in einer anderen Task zu empfangen. (Z.B. wenn ein Routineprüfprogramm einen Fehler in einer Baueinheit feststellt und eine Fehlermeldung an das ORG1 senden will.)
Jede derartige Nachricht muß außer der Information ein Codewort enthalten, mit dessen Hilfe vom Überwacher jene andere Task ermittelt wird, für welche die Nachricht bestimmt ist. Zu diesem Zweck führt der Überwacher eine Codewortliste für jede Task, die eine Nachricht erhalten kann. Diese Liste enthält für jede dieser Tasks ein oder mehrere spezielle Codeworte.

Eine Nachricht hat folgende Struktur:

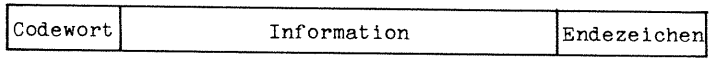


Bild 3.15: Struktur einer Nachricht

Die Nachricht beginnt mit einem Codewort aus vier Zeichen und endet mit einem Endezeichen. Dazwischen ist eine Information beliebiger Länge eingebettet.

Der Aufbau einer Coderwortliste ist im Bild 3.16 dargestellt:

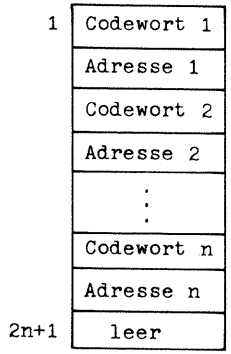


Bild 3.16: Struktur der Codewortliste einer Task, die Nachrichten empfangen kann

Jedes Codewort der Liste besteht wie das Codewort einer Nachricht aus vier Zeichen (≙1 Wort) und wird gefolgt von einer Adresse.

Erhält der Überwacher von einer Task den Aufruf SENDE NACHRICHT (Adr) so liest er in der angegebenen Adresse Adr der aufrufenden Task das Codewort der Nachricht. Nun vergleicht er dieses Codewort mit den Codeworten aller Codewortlisten auf Koinzidenz.

Stellt er eine Koinzidenz fest, so wird jene Task aktiviert, welche zu der betreffenden Liste gehört.(Die Task wird vom Zustand "blockiert" in den Zustand "bereit" überführt.) Die in der Liste (Bild 3.16) unmittelbar hinter dem Codewort stehende Adresse gibt den Befehlszählerstand an, bei dem die Task fortgesetzt werden soll. (Diese Adresse wird vom Überwacher im Taskkontrollblock jener Task in die Zelle für den Befehlszähler eingetragen.)

Nunmehr kann die empfangende Task die in der Nachricht stehende Information vom Überwacher zeichenweise anfordern. Dazu muß sie den Überwacheraufruf NACHRICHTENZEICHEN benutzen.

Ist aber jene empfangende Task, zu welcher das gesendete Codewort gehört, in diesem Augenblick andersweitig aktiv, so ist deren Codewortliste für den Überwacher nicht zugänglich, d.h. die Vergleichsoperation muß erfolglos bleiben. In diesem Fall wird die nicht bearbeitbare Nachricht vom Überwacher in die Nachrichtenwarteschlange eingereiht. Diese Warteschlange ist realisiert als blockweise organisierte (gekettete) Liste, wobei im Überwacher nur die beiden Zeiger für den Anfang und das Ende der Liste geführt werden und die einzelnen Listenelemente (Nachrichten) in den Programmen der Tasks stehen (vgl. Anhang A1.2).

Immer wenn eine aktive - deshalb bezüglich ihrer Coderwortliste nicht abfragbare Task - an den Überwacher den Aufruf ERWARTE NACHRICHT (Adr) schickt, bedeutet dies, daß sie momentan ihre Arbeit beendet hat und auf eine Nachricht warten will. Sie wird in den Zustand "blockiert" überführt. Dabei wird sie aus der Taskwarteschlange ausgetragen und in eine Liste der für eine Nachricht empfangsbereiten Tasks eingereiht (Bild 3.17, diese Liste wird auch als Empfängerwarteschlange EWS bezeichnet).

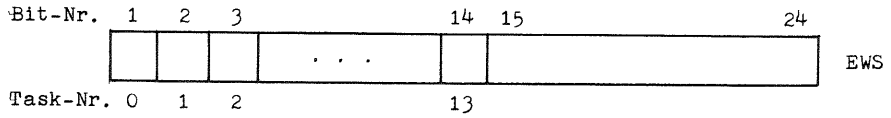


Bild 3.17: Liste der für Nachrichten empfangsbereiten Tasks (Empfängerwarteschlange EWS)

Jedes gesetzte Bit in der Liste entspricht einer (für Nachrichten) empfangsbereiten Task. Der Befehlszähler im Taskkontrollblock einer wartenden Task zeigt jeweils auf deren Codewortliste. Die Adresse Adr wurde dem Überwacher mit dem Aufruf ERWARTE NACHRICHT (Adr) übergeben.

Jedesmal wenn eine Task in diese Liste eingereiht wird, prüft der Überwacher ob in der Nachrichtenwarteschlange bereits eine Nachricht wartet, deren Codewort in der Codewortliste der neu eingetragenen Task steht. Wenn dies der Fall ist, kann die Task,wie bereits oben behandelt,wieder aktiviert werden.

In besonderen Fällen kann die sendende Task nach der Übergabe der Nachricht an den Überwacher, welches mit dem Aufruf SENDE NACHRICHT erfolgte, ihre Arbeit unterbrechen und mit dem Aufruf WARTEN AUF QUITTUNG auf eine Quittung der empfangenden Task warten.

Ist die Nachricht nach vorherigem Verweilen in der Nachrichtenwarteschlange über den Überwacher an die empfangende Task übergeben worden, so sendet die empfangende Task den Aufruf SENDE QUITTUNG an den Überwacher. Daraufhin aktiviert der Überwacher wieder die auf eine Quittung wartende Task.

Die Zustandsübergänge der Tasks durch die Uberwacheraufrufe zur Kommunikation (Nachrichtenübergabe) sind im folgenden Bild 3.18 dargestellt.

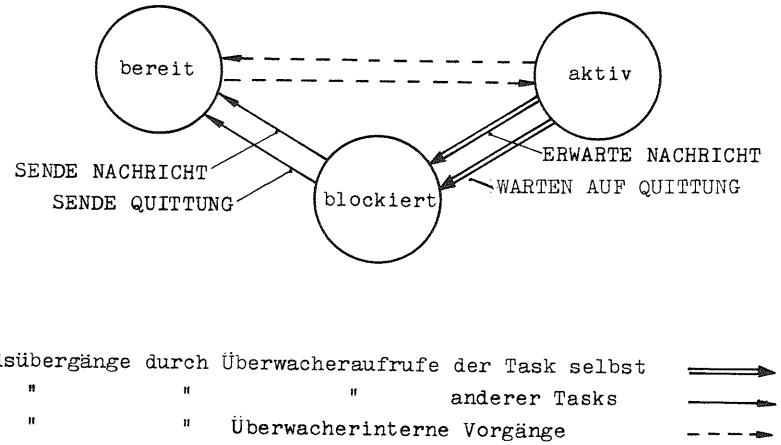


Bild 3.18: Zustandsübergänge von Tasks durch Kommunikationsfunktionen

Durch die Überwacheraufrufe ERWARTE NACHRICHT und WARTEN AUF QUITTUNG geht eine "aktive" Task in den Zustand "blockiert" über.

Eine auf eine Nachricht wartende Task ("blockiert" wegen Aufruf ERWARTE NACHRICHT) wird wieder "bereit", wenn eine andere Task eine Nachricht sendet, die das Codewort enthält, auf das die betreffende Task wartet.

Eine auf eine Quittung wartende Task ("blockiert" wegen Aufruf WARTEN AUF QUITTUNG) wird wieder "bereit", wenn die Task, für welche die Nachricht bestimmt war, den Aufruf SENDE QUITTUNG an den Überwacher abgibt.

3.4 Ein-Ausgabe Standardperipherie

3.4.1 Allgemeines

Die Ein-Ausgabegeräte des Rechners, die nicht unmittelbar zur vermittlungstechnischen Peripherie gehören, werden als "Standardperipherie" des Rechners bezeichnet. Es sind dies:

- Blattschreiber (Bedienungsblattschreiber)
- Lochkartenleser
- Lochkartenstanzer
- Lochstreifenleser
- Lochstreifenstanzer
- Schnelldrucker
- Magnetplatte

Für den Versuchsbetrieb der Vermittlungsstelle muß die gesamte Standardperipherie ebenfalls von der Systemsoftware der Vermittlungsstelle betrieben werden.

Sämtliche Funktionen, die diese Geräte ansprechen, werden als Ein-Ausgabefunktionen für die Standardperipherie bezeichnet. Bei der Magnetplatte gibt es neben den eigentlichen Funktionen für die Ein-Ausgabe noch die Funktionen für die Dateiverwaltung.

Alle Ein-Ausgabefunktionen wurden kompatibel zum Organisationsprogramm ORG306 /15/ realisiert. Es wird daher im folgenden nur der prinzipielle Ablauf der Funktionen beschrieben. Die einzelnen realisierten Funktionen sind im Anhang A2 aufgeführt.

3.4.2 Ein-Ausgabebearbeitung

Die Ein-Ausgabe über die Standardperipherie wird durch Ein-Ausgabebefehle an die Kanal- und Gerätesteuerungen angestoßen und läuft dann simultan zur Befehlsausführung in der Zentraleinheit ab. Das Ende einer Ein-Ausgabeoperation wird von den Geräten durch ein Unterbrechungssignal (Kanalanforderung, vgl. Abschnitt 2.2) gemeldet.

Die Bearbeitungsroutinen für die Ein-Ausgabe (Kanalprogramme) werden entweder durch einen Überwacheraufruf von einer Task aus oder durch ein Unterbrechungssignal von der Peripherie aus aktiviert. Der prinzipielle Ablauf ist im Bild 3.19 dargestellt.

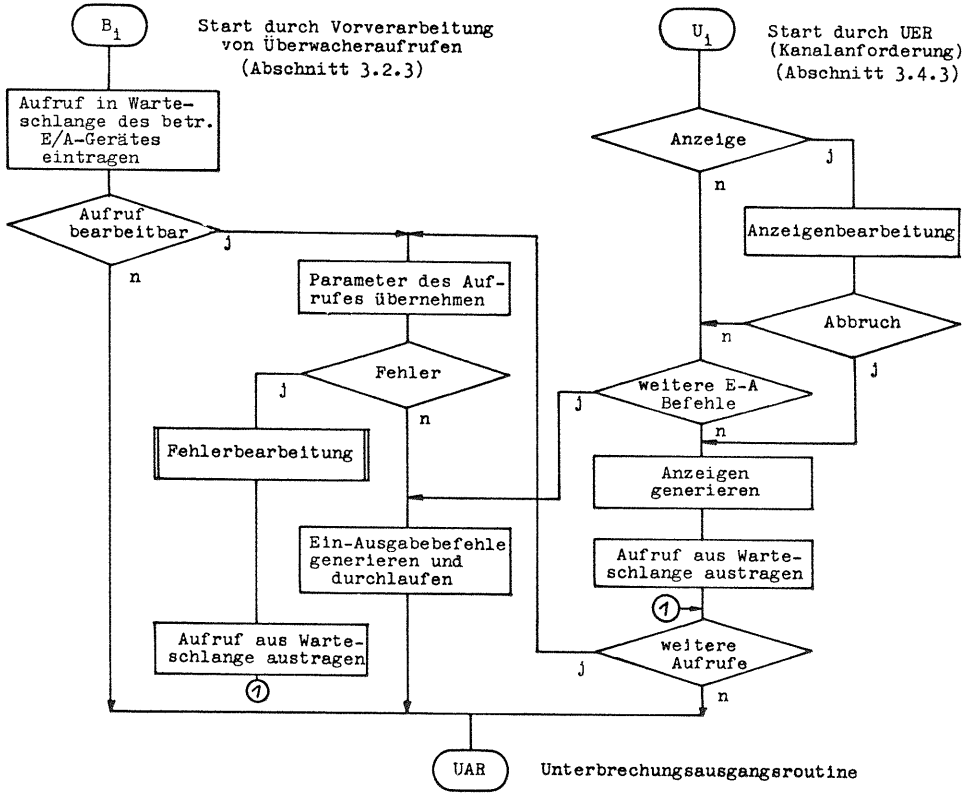


Bild 3.19: Bearbeitung eines Ein-Ausgabeaufrufes

Ein eintreffender Überwacheraufruf wird von der betreffenden Bearbeitungsroutine zuerst in die Warteschlange für das Gerät,über welches die Ein-Ausgabe erfolgen soll, eingereiht. (Linker Teil im Flußdiagramm Bild 3.19). Die Warteschlange ist so organisiert, daß der erste Eintrag (Überwacheraufruf) gerade in Bearbeitung ist. Nach dem Eintrag wird, falls schon andere Aufrufe in der Warteschlange stehen, in die Unterbrechungsausgangsroutine (UAR) gesprungen, da zuerst diese Aufrufe zu Ende bearbeitet werden müssen.

Falls es sich um den ersten Eintrag in der Warteschlange handelt, wird der Aufruf zuerst auf formale Fehler geprüft (z.B. ob die Markierbits im Steuerwort des Aufrufes oder die Zahl der Parameter zulässig sind, vgl. Abschnitt 3.2.3). Die einzelnen Parameter werden nicht beim Eintragen des Aufrufes in die Warteschlange geprüft, sondern erst bei der Ausführung der Funktion. Dies ist zweckmäßig, da die Parameter zur Überprüfung aus der Task, von welcher der Überwacheraufruf abgesetzt wurde, in den Überwacher übernommen werden müssen und damit auch schon für die Ausführung der Funktion zu Verfügung stehen.

Wenn die Parameter in dem Überwacheraufruf zugelassen sind, werden die betreffenden Ein-Ausgabebefehle an das externe Gerät generiert und durchlaufen. Danach wird in die UAR gesprungen.

Bei nicht zugelassenen Parametern in dem Überwacheraufruf wird zu einer Fehlerbearbeitungsroutine verzweigt. Diese bricht die Task ab, von welcher der Aufruf stammt.(Dabei wird die Task aus der Taskwarteschlange ausgetragen.) Außerdem gibt die Fehlerbearbeitungsroutine eine Fehlermeldung für das Vermittlungspersonal auf dem Bedienungsblattschreiber aus (s. Abschnitt 3.6.4). Nach dem Austragen des Überwacheraufrufes aus der Warteschlange wird geprüft,ob noch weitere Aufrufe in der Warteschlange eingetragen sind. Gegebenenfalls wird der nächste wartende Überwacheraufruf bearbeitet.

Nach dem Durchlaufen der Ein-Ausgabebefehle kann die weitere Aufrufbearbeitung durch die Bearbeitungsroutine des Überwachers erst durchgeführt werden, wenn das Ende der Ein-Ausgabeoperation durch ein Unterbrechungssignal (Kanalanforderung) dem Überwacher gemeldet wird. Das betreffende Gerät, welches die Ein-Ausgabeoperation ausführt, sendet nach Abschluß der Operation eine Kanalanforderung an die Zentraleinheit.

Diese Kanalanforderung muß von der Unterbrechungseingangsroutine (UER) verarbeitet werden. Die Fortsetzung der Aufrufbearbeitung beginnt anschließend mit der Abfrage, ob eine "Anzeige" bei der Durchführung der Ein-Ausgabeoperation aufgetreten ist. (Eine Anzeige tritt auf,

wenn eine Operation nicht richtig ausgeführt werden konnte. Die Beschreibung der Anzeigenbearbeitung erfolgt im nächsten Abschnitt 3.4.3)

Falls eine Anzeige aufgetreten ist ("die Operation wurde nicht richtig ausgeführt") , wird in die Anzeigenbearbeitungsroutine verzweigt. Wenn dort die Ursache für die Anzeige, meist durch einen Eingriff des Vermittlungspersonals, nicht behoben werden konnte, wird die weitere Ausführung des Aufrufes durch die Bearbeitungsroutine des Überwachers abgebrochen. (z.B. wenn bei einem Lesefehler während einer Lochkarteneingabe die fehlerhafte Lochkarte nicht korrigiert werden kann.) Dabei wird eine Anzeige in die "Anzeigenzelle" des Aufrufs (vgl. Abschnitt 3.2.2) eingetragen und der Aufruf aus der Warteschlange ausgetragen. Befinden sich weitere Aufrufe in der Warteschlange, so gelangt der nächste wartende Aufruf zur Bearbeitung.

Wenn die Ein-Ausgabeoperation vom betreffenden Gerät richtig ausgeführt wurde, so wird geprüft, ob der Aufruf komplett ausgeführt ist, oder ob noch weitere Ein-Ausgabebefehle notwendig sind.(Dies ist z.B. bei blockweiser Ausgabe von mehreren Blöcken notwendig.) Gegebenenfalls werden die nächsten Ein-Ausgabebefehle generiert und durchlaufen.

Bei einem Überwacheraufruf, der komplett ausgeführt ist, wird das Nullwort (keine Anzeige) in die Anzeigenzelle eingetragen. (Das Nullwort in der Anzeigenzelle zeigt der Task, welche den Überwacheraufruf abgesetzt hat, an, daß der Aufruf vom Überwacher ordungsgemäß ausgeführt wurde.) Daraufhin wird der Aufruf aus der Warteschlange ausgetragen.

3.4.3 Unterbrechungsbearbeitung für Kanalanforderungen

3.4.3.1 Allgemeines

Die externen Geräte müssen dem Überwacher den Abschluß einer Schreib-, Lese- oder Steueroperation durch ein Unterbrechungssignal (Kanalanforderung) mitteilen (vgl. Abschnitt 3.4.2). Damit der Überwacher das Gerät identifizieren kann, muß das betreffende Gerät in fest vorgegebenen Arbeitsspeicherzellen ein geräteindividuelles Bit setzen. Zusätzlich muß von dem Gerät in einer weiteren Arbeitsspeicherzelle, die jedem Gerät ebenfalls fest zugeordnet ist, eine Anzeige über das Ergebnis der Operation (erfolgreich oder nicht erfolgreich mit Ursache) abgelegt werden.

Die Abläufe in einem externen Gerät zur Abgabe einer Kanalanforderung und die Reaktion des Rechners auf diese Kanalanforderung können wie folgt beschrieben werden:

Anzeige in Geräteanzeigezelle eintragen
Bit für Geräteidentifikation setzen
Unterbrechungssignal an Prozessor senden
} externes Gerät

Fortsetzadresse in Zelle 0 eintragen
Bearbeitungsroutine für Unterbrechungssignal aktivieren
} Prozessor

Nach dem Abschluß einer Operation setzt das betreffende Gerät sein Identifikationsbit im Arbeitsspeicher. Weiterhin schreibt das Gerät die Anzeige in den Arbeitsspeicher und sendet das Signal "Kanalanforderung" an die Zentraleinheit. Wenn das Unterbrechungssignal Kanalanforderung wirksam wird, wird zu der Bearbeitungsroutine für die Unterbrechungsbearbeitung von Kanalanforderungen gesprungen. Die Adresse des nächsten Befehles der unterbrochenen Task (Fortsetzadresse) wird durch das Unterbrechungssystem des Rechners in der Arbeitsspeicherzelle Null abgelegt.

3.4.3.2 Unterbrechungsvorverarbeitung

Nachdem die Unterbrechung einer Task durch eine Kanalanforderung wirksam wurde, muß die Fortsetzadresse der unterbrochenen Task in deren Taskkontrollblock eingetragen werden. Danach muß die Bearbeitungsroutine für das Gerät, dessen Identifikationsbit gesetzt ist, aktiviert werden.

Da bis zum Zeitpunkt, bei dem die Unterbrechung für ein Gerät wirksam wird, auch Kanalanforderungen von anderen Geräten in der Zentraleinheit eingetroffen sein können, werden die Identifikationsbits aller Geräte in einer Arbeitsspeicherzelle zwischengespeichert. Die Identifikationsbits können dann nacheinander abgearbeitet werden.

Die Auswahl der Bearbeitungsroutine für ein Gerät, das eine Kanalanforderung an die Zentraleinheit geschickt hat, erfolgt über eine sogenannte "Sprungleiste" (Bild 3.20).

Der Zugang zu der Sprungleiste erfolgt über die Lage des Identifikationsbits in der Arbeitsspeicherzelle, in der alle Identifikationsbits zwischengespeichert werden. Es sind beim Rechner S306 maximal 24 Geräte und damit 24 Bearbeitungsroutinen möglich.

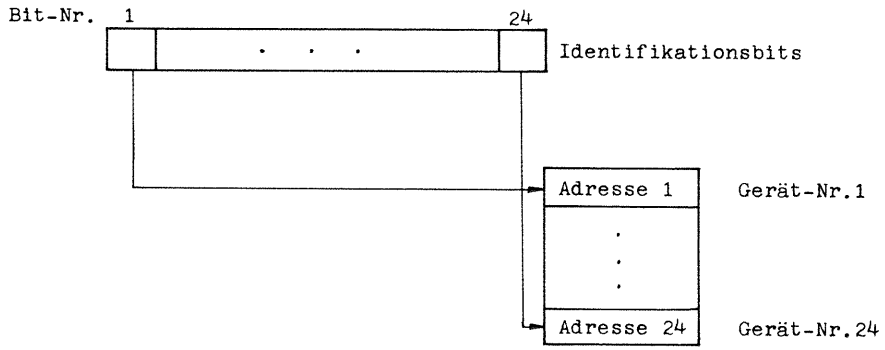


Bild 3.20: Sprungleiste zur Auswahl der Bearbeitungsroutine für Kanalanforderungen

Bei dem für die Versuchsvermittlungsstelle verwendeten Rechner sind nur 7 Geräte angeschlossen, d.h. es sind auch nur 7 unterschiedliche Identifikationsbits möglich. In der Sprungleiste steht bei den Identifikationsbits, die keinem Gerät zugeordnet werden können, ein Sprung in die Fehlerbearbeitungsroutine (s. Abschnitt 3.6.4).

Die Unterbrechungsvorverarbeitungsroutine ist ein Teil der Unterbrechungseingangsroutine (s. Abschnitt 3.9).

3.4.3.3 Anzeigenbearbeitungsroutine

Treten bei der Ausführung einer Schreib-, Lese oder Steueroperation Besonderheiten auf, so wird dies in der Anzeigenzelle des Gerätes gemeldet. Die Ursachen können unterschiedlichster Art sein; z. B. das Gerät ist nicht eingeschaltet oder es sollen nicht zugelassene Zeichen ausgegeben werden.

Man unterscheidet bezüglich ihrer Behandlung zwei Arten von Anzeigen, nämlich Anzeigen, die ein Eingreifen des Vermittlungspersonals erforderlich machen, und Anzeigen, die vom Überwacher selbständig bearbeitet werden können. In beiden Fällen wird eine sogenannte "Fehlermeldung" auf dem Bedienungsblattschreiber ausgegeben.

Wenn ein Eingriff des Bedienungspersonals erforderlich ist, muß dieses versuchen durch geeignete Maßnahmen den Fortgang der Ein-Ausgabebearbeitung zu ermöglichen.(Z.B. wenn beim Schnelldrucker das Tabellierpapier ausgeht,muß neues Papier eingelegt werden.) Durch Eingabe eines Quittungswortes über den Blattschreiber teilt das Vermittlungspersonal dann entweder mit, daß die Fehlerursache behoben werden

konnte (z.B. neues Tabellierpapier beim Schnelldrucker wurde eingelegt), oder daß die Ausführung des Überwacheraufrufes abgebrochen werden soll (z.B. das Drucken auf dem Schnelldrucker soll nicht fortgesetzt werden, da gerade kein neues Tabellierpapier vorrätig ist).

Für die Ausgabe von Fehlermeldungen wurden die beiden Unterprogramme GERMELD und GERFEHL realisiert. Der Aufruf der Unterprogramme erfolgt jeweils mit Angabe der Gerätenummer und eines Codewortes, bestehend aus vier Zeichen, das auf dem Bedienungsblattschreiber ausgegeben werden soll, und zusätzlich der Anzeige. Auf dem Bedienungsblattschreiber erfolgt dann eine Meldung, deren Format im Bild 3.21 dargestellt ist.

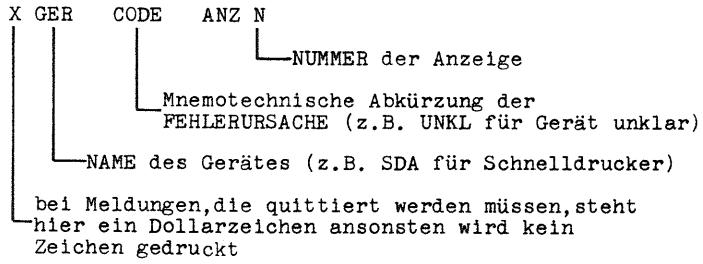


Bild 3.21: Aufbau einer Fehlermeldung

Bei Meldungen, die eine Quittung erforderlich machen, muss mit GER+ oder GER- quittiert werden. Dabei ist für GER die Bezeichnung der Fehlermeldung zu verwenden (z. B. bei Schnelldrucker SDA).

Der Überwacher führt eine Codewortliste für alle möglichen Quittungen. (Diese Codewortliste ist organisiert wie die Codewortlisten bei der Kommunikation zwischen Tasks; vgl. Abschnitt 3.3.5 .) Nach dem Empfang eines Quittungswortes über den Bedienungsblattschreiber, vergleicht der Überwacher dieses Quittungswort mit allen Codeworten seiner Liste auf Koinzidenz.

Falls er Koinzidenz feststellt, aktiviert er die entsprechende Bearbeitungsroutine. (Die Adresse der Bearbeitungsroutine steht in der Codewortliste hinter dem entsprechenden Codewort.) Weiterhin quittiert er das eingegebene Quittungswort durch Ausgabe eines Semikolons auf dem Bedienungsblattschreiber.

Falls der Überwacher keine Koinzidenz feststellt, so zeigt er dies durch Ausgabe eines Irrungszeichens auf dem Blattschreiber an. (In diesem Fall wurde ein falsches Quittungswort eingegeben.)

Durch diese Art von Fehlerbehandlung können mehrere Fehlermeldungen ausgegeben werden und auf deren Quittungen gewartet werden.

3.4.4 Warteschlangenorganisation

Die Warteschlangen für die peripheren Geräte sind analog zum ORG306 realisiert als blockweise organisierte (gekettete) Listen (s. Anhang A1.2). Im Organisationsprogramm stehen nur die beiden Zeiger für den Anfang und das Ende der Liste. Die Aufrufe in den Programmen sind die Blöcke, die verkettet werden. Zur Verkettung der Blöcke werden die Anzeigenzellen der Aufrufe verwendet, da diese erst nach Abschluß der Aufrufbearbeitung zum Eintragen der Quittung benötigt werden. Die Warteschlangen sind so organisiert, dass die Aufrufe in ihrer zeitlichen Eintreffreihenfolge nach abgearbeitet werden.

Ein Kettenzeiger hat folgendes Format:

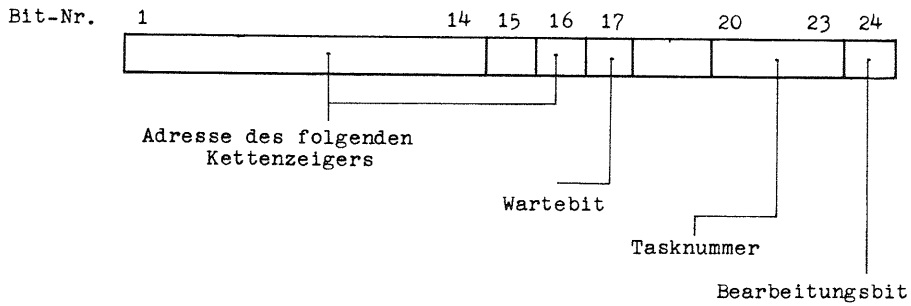


Bild 3.22: Format eines Kettenzeigers

Bit 1 bis 14 und 16 geben die Adresse des nächsten Kettenzeigers an. Bit 24 ist 1, wenn der Aufruf noch in der Warteschlange eingetragen ist, d.h. entweder gerade in Bearbeitung ist oder noch wartet. Das Wartebit Bit 17 zeigt an, wenn die Task im Zustand "blockiert" ist und dadurch auf das Ende der Ein-Ausgabeoperation wartet. Bit 20 bis 23 enthalten die Nummer der Task, zu der der Aufruf gehört, der durch den Kettenzeiger angegeben ist.

Es beziehen sich Bit 17 und 24 auf den Aufruf in dessen Anzeigenzelle der Zeiger steht und die restlichen Bits auf den in der Warteschlange folgenden Aufruf.

Durch diese Art der Warteschlangenorganisation können beliebig viele Aufrufe in eine Warteschlange eingereiht werden, da ein Aufruf, der an den Überwacher abgegeben wurde, im Programm jener Task steht, von welcher der Aufruf kommt, und daher keinen zusätzlichen Speicherplatz benötigt.

Diese Art der Warteschlangenorganisation hat allerding einen Nachteil, nämlich, wenn eine Task eine Anzeigenzelle (Kettenzeiger) in einem Aufruf überschreibt, der noch in einer Warteschlange eingetragen ist. Dadurch wird die Verkettung der Warteschlange zerstört. Nach einer Zerstörung einer Kette ist eine weitere Aufrufbearbeitung nicht mehr möglich.

Um diesen Nachteil zu umgehen, gibt es die Möglichkeit, die Warteschlange im Überwacher selbst zu führen, was einen Mehraufwand an Speicherplatz erfordert. Wegen des zusätzlichen Speicherplatzbedarfes wurde diese Möglichkeit nicht realisiert.

Eine weitere Möglichkeit besteht darin, den Aufruf für die entsprechende Funktion von der aufzurufenden Task so lange zu wiederholen, bis er ausgeführt werden kann. Dadurch würde der Prozessor praktisch "blind" belegt, denn er wiederholt nur noch den Aufruf und er wird für andere Tasks nicht frei. Diese Lösung ist also auch nicht durchführbar.

Für die Warteschlangenorganisation wurden spezielle Unterprogramme zum Eintragen und Austragen der Aufrufe aus den Warteschlangen entwickelt. Das Eintragen erfolgt durch den Aufruf WSEINTRAG (GERWSANF). Dabei gibt GERWSANF die Adresse des Kettenzeigers im Überwacher an, der auf den ersten Eintrag dieser Warteschlange verweist. Die Adresse des Aufrufes im Programm der aufrufenden Task ist dem Unterprogramm von der Aufrufbearbeitung her bekannt. Das Austragen eines Aufrufes erfolgt durch den Aufruf WSAUSTRAG (GERWSANF).

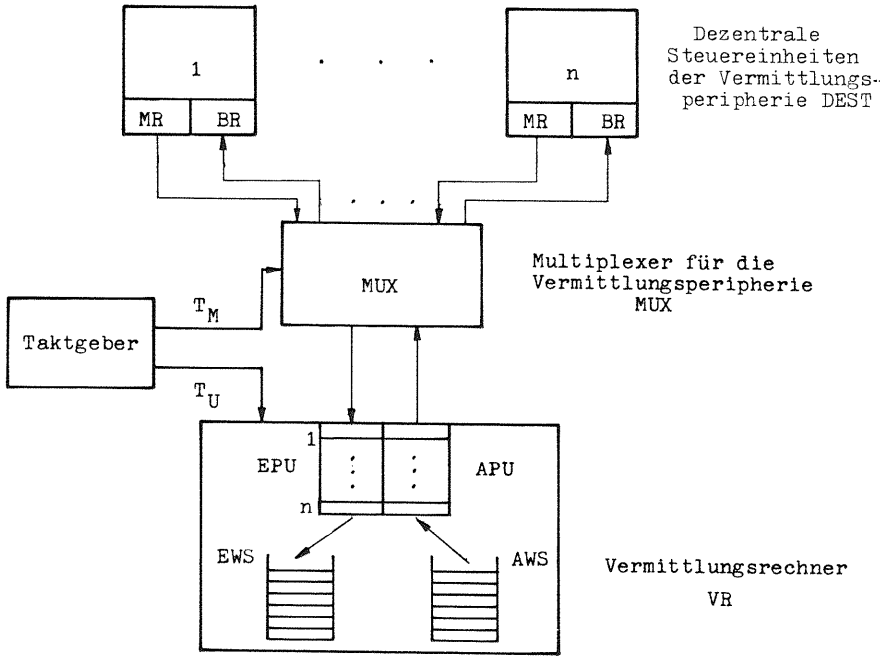
3.5 Ein-Ausgabe Vermittlungsperipherie

3.5.1 Allgemeines

Alle peripheren Baugruppen der Vermittlungsperipherie werden durch dezentrale Steuereinheiten (DEST) gesteuert (vgl. Abschnitt 1.2). Der Informationsaustausch zwischen dem zentralen Vermittlungsrechner (VR) und diesen DEST erfolgt über einen Multiplexer für die Vermittlungsperipherie.

Das Prinzip des Informationsaustausches zwischen der Vermittlungsperipherie und dem Vermittlungsrechner ist im Bild 3.23 dargestellt.

Die DEST besitzen jeweils ein Melderegister (MR) und ein Befehlsregister (BR) mit einer Speicherkapazität von 24 bit, die der Wortlänge des Rechners entspricht. Im Melderegister wird die Eingabeinformation (Meldung) für den Vermittlungsrechner bereitgestellt. Das Befehlsregister übernimmt die Ausgabeinformation (Befehl) vom Vermittlungsrechner. (Die Formate der Meldungen und Befehle sind im Anhang A1.4 angegeben.)



APU	Ausgabepuffer	EPU	Eingabepuffer
AWS	Ausgabewarteschlange	EWS	Eingabewarteschlange
BR	Befehlsregister	MR	Melderegister

Bild 3.23: Informationsaustausch zwischen Vermittlungsperipherie und dem Vermittlungsrechner

Im Rechner ist ein Eingabepuffer und ein Ausgabepuffer mit jeweils n Speicherplätzen vorgesehen. Jeder DEST ist ein Speicherplatz im Eingabepuffer und ein Speicherplatz im Ausgabepuffer durch die Nummer der DEST fest zugeordnet.

Der Multiplexer frägt angestoßen durch den Taktgeber zyklisch alle τ ms (10ms) die n MR der DEST ab, ob sie eine Meldung enthalten. Gegebenenfalls übernimmt er eine wartende Meldung für den VR aus dem MR und schreibt sie im Eingabepuffer in jene Speicherzelle, welche der DEST zugeordnet ist.

Wenn alle n MR abgearbeitet sind, beginnt die Ausgabephase. Hierbei werden zunächst die BR abgefragt, ob sie frei, d.h. empfangsbereit sind. Wenn ein BR frei ist, wird ein vorliegender Befehl aus dem Ausgabepuffer in das zugehörige BR übertragen und im APU gelöscht.

Die Meldungen im Eingabepuffer des VR müssen bis zum nächsten Ein-Ausgabezyklus wieder ausgelesen sein, damit sie nicht überschrieben werden. Da das Verarbeitungsprogramm für die Meldungen (Vermittlungsprogramm) u.U. nicht alle Meldungen innerhalb dieser Zeit (≙10ms) bearbeiten kann, werden die Meldungen grundsätzlich in einer Eingabewarteschlange zwischengespeichert. Analog dazu werden alle Ausgabebefehle in einer Ausgabewarteschlange bis zur Ausgabe zwischengespeichert.

Das Umspeichern der Meldungen vom Eingabepuffer in die Eingabewarteschlange und der Befehle von der Ausgabewarteschlange in den Ausgabepuffer wird von einer Bearbeitungsroutine (s. Abschnitt 3.5.2.1) ausgeführt, die ebenfalls zyklisch alle τ ms angestoßen wird. Der Anstoß erfolgt durch Unterbrechungssignale T_U von einem Taktgeber. Die Unterbrechungssignale sind um $\tau/2$ gegenüber dem Multiplexertakt versetzt. (Es stehen also $\tau/2$ (5ms) für die Ein-Ausgabe der Peripherie zu Verfügung. Dieselbe Zeit steht maximal für das Umspeichern der Meldungen und Befehle zwischen den Pufferspeichern und den Warteschlangen zur Verfügung.)

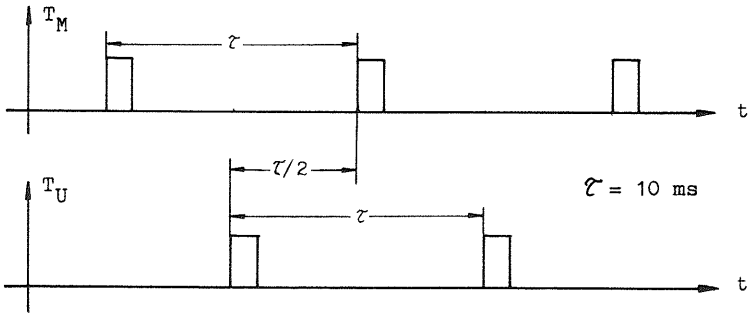


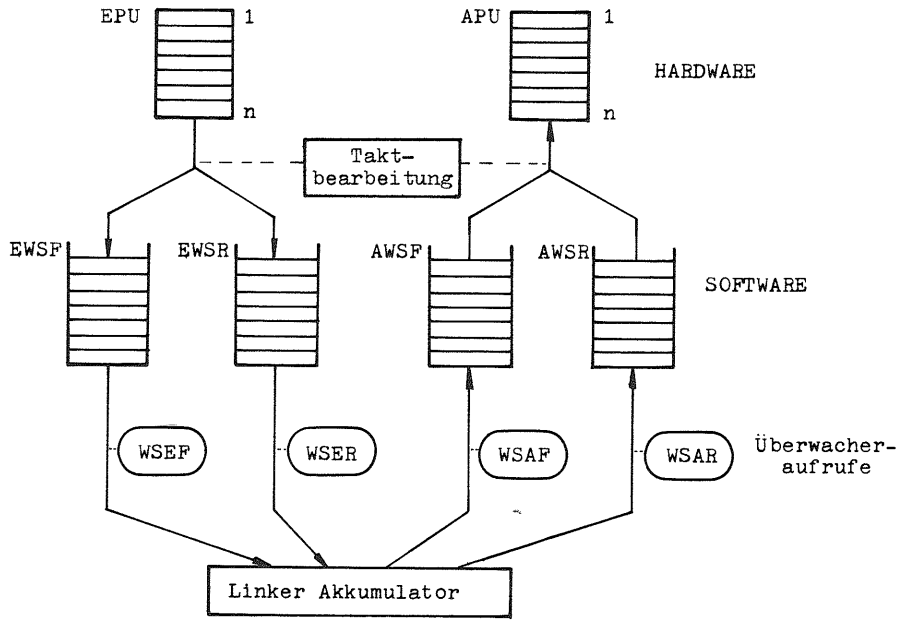
Bild 3.24: Taktfolgen zur Steuerung des Informationsaustausches zwischen der Vermittlungsperipherie und dem Vermittlungsrechner

Damit gewährleistet ist, daß innerhalb $\tau/2$ ms das Umspeichern der Meldungen vom EPU in die EWS erfolgt ist, muß das Unterbrechungssignal, das diese Bearbeitungsroutine aktiviert, möglichst sofort nach dem Eintreffen wirksam werden. Dies ist möglich, wenn das Unterbrechungssignal als sogenanntes "Alarmsignal" (vgl. Abschnitt 2.2) an den Prozessor gegeben wird.

Ein eintreffendes Alarmsignal verursacht im Rechner sofort eine Unterbrechung der gerade laufenden Befehlsfolge unabhängig davon, in welchem Zustand sich der Rechner befindet. (Im Gegensatz zu einer E/A-Kanalanforderung,die nur im Normalzustand des Rechners wirksam wird.)

3.5.2 Ein-Ausgabebearbeitung im Einzelnen

Die Ein-Ausgabe von und zur Vermittlungsperipherie erfolgt gemäß Abschnitt 3.5.1 gepuffert über sogenannte Warteschlangen. Dabei wird die zu übertragende Information in zwei Klassen eingeteilt, nämlich in Information zur Fehlerbehandlung (Fehlermeldungen) und in reguläre Meldungen. Die Fehlermeldungen müssen mit höherer Dringlichkeit bearbeitet werden; deshalb werden dafür getrennte Warteschlangen geführt (s. Bild 3.25).



APU	Ausgabepuffer	EPU	Eingabepuffer
AWSF	Ausgabewarteschlange für Befehle zur Fehlerbearb.	EWSF	Eingabewarteschlange für Fehlermeldungen
AWSR	Ausgabewarteschlange für reguläre Befehle	EWSR	Eingabewarteschlange für reguläre Meldungen

WSEF	Überwacheraufruf zur Eingabe aus der Warteschlange		EWSF
WSER	" " " " " "		EWSR
WSAF	" " Ausgabe in die		AWSF
WSAR	" " " " " "		AWSR

Bild 3.25: Ablauf der Ein-Ausgabe für die Vermittlungsperipherie

Die Übernahme der Meldungen (Fehlermeldungen und reguläre Meldungen) von der Peripherie in die Eingabewarteschlangen sowie die Ausgabe der Befehle aus den Ausgabewarteschlangen erfolgt durch die "Taktbearbeitungsroutine (TBR)".

Die Verarbeitung der eingetroffenen Meldungen erfolgt durch Arbeitsprogramme, die unter der Regie des ORG1 (Fehlermeldungen) und des ORG7 (reguläre Meldungen) zur Ausführung kommen (vgl. Abschnitt 2.1.2). Die Meldungen aus den Eingabewarteschlangen können mit Überwacheraufrufen übernommen werden. Ebenso können Befehle in die Ausgabewarteschlangen mit Überwacheraufrufen eingetragen werden (Bild 3.25).

Die gesamte Ein-Ausgabebearbeitung gliedert sich in einen Teil, der durch die Taktbearbeitungsroutine durchgeführt wird, und in einen Teil, der in einer Bearbeitungsroutine für Überwacheraufrufe zu Ausführung kommt (s. Bild 3.25).

3.5.2.1 Taktbearbeitungsroutine

Die Taktbearbeitungsroutine (TBR) wird alle τ ms (10ms) durch einen Taktgeber mittels eines "Alarmsignales" (vgl. Abschnitt 2.2) aktiviert. Die TBR kann neben dem Umspeichern von Meldungen und Befehlen weitere Aufgaben, die periodisch durchzuführen sind, übernehmen. Dazu zählt das Fortschalten einer Systemzeit, die für die vermittlungstechnischen Aufgaben (z.B. zur Bestimmung der Gesprächszeit für eine Fernsprechverbindung) benötigt wird, sowie das periodische Aktivieren des ORG6, das die Zeitprogramme für Zeitüberwachungsaufgaben steuert. Das ORG6 wird in T_o Zeitintervallen aktiviert, die von dem Aufruftakt des Taktgebers abgeleitet werden. Weiterhin muß das ORG1 bei eintreffenden Fehlermeldungen und das ORG7 bei eintreffenden regulären Meldungen aktiviert werden, wenn vorher keine Einträge mehr in den jeweiligen Eingabewarteschlangen vorhanden waren.(In diesem Fall sind die betreffenden Organisationsprogramme im Zustand "blockiert" und warten auf das Eintreffen von Meldungen.)

Der Ablauf bei der Taktbearbeitung durch die TBR wird im folgenden anhand des Flußdiagramms (Bild 3.26) beschrieben:

In der Unterbrechungseingangsroutine (vgl. Abschnitt 3.9) werden die Registerinhalte für die Fortsetzung der unterbrochenen Befehlsfolge gerettet und dann wird sofort in die Bearbeitungsroutine TBR gesprungen.

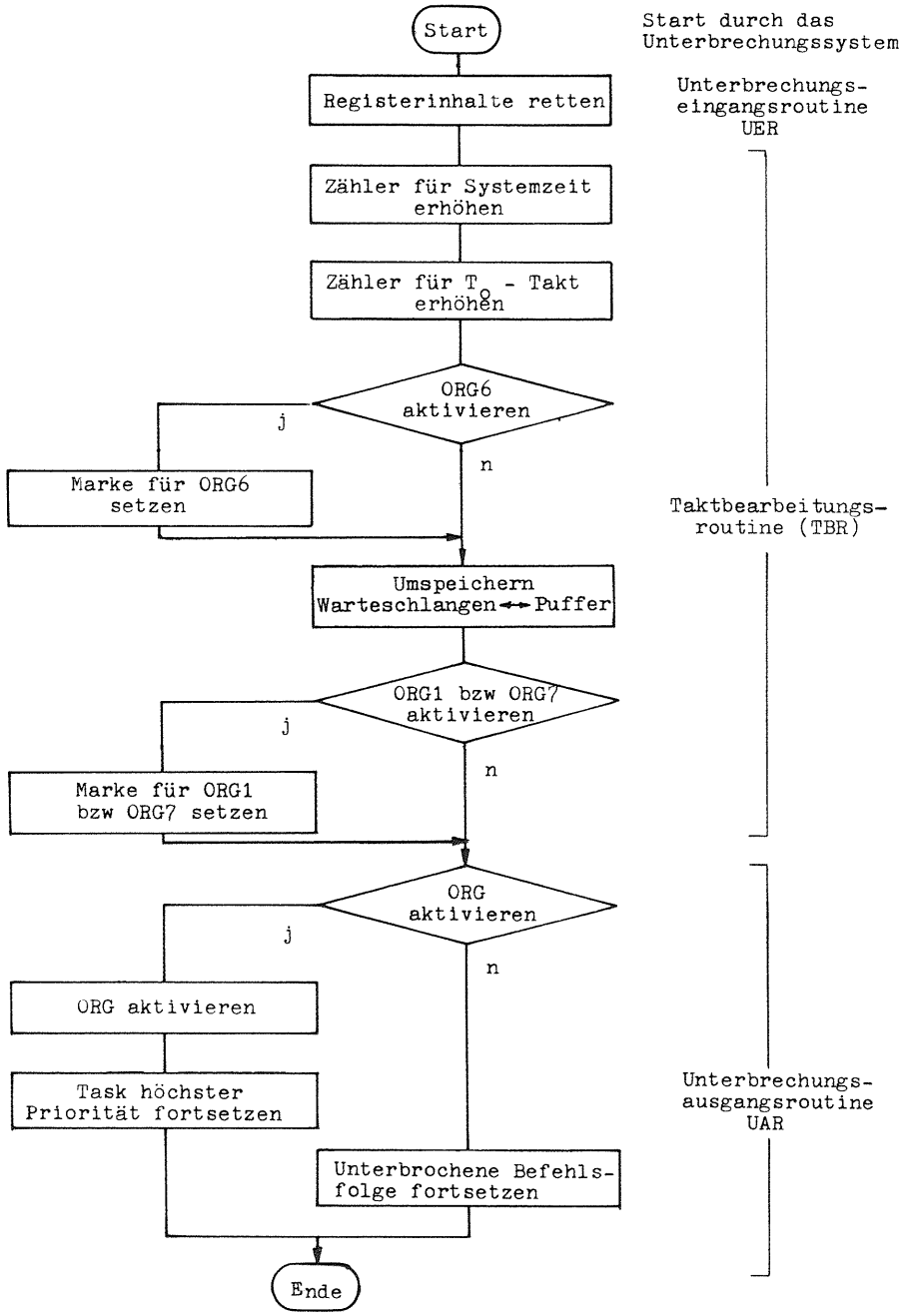


Bild 3.26: Ablauf bei der Taktbearbeitung

Für die Systemzeit wird ein Zähler geführt, der bei jedem Durchlaufen der TBR um eins erhöht wird. Die Einheit der Systemzeit beträgt dann τ ms (10ms) und der Zähler hat bedingt durch die Zahlendarstellung von ganzen Zahlen beim Rechner S306 (Vorzeichenbit + 23bit Dualzahl) eine Periode von $2^{23} \cdot 10$ ms $\approx$ 84900 sec, das etwa einem ganzen Tag entspricht.

Aus dem Takt T_U, der das Alarmsignal liefert, wird durch Taktuntersetzung mit einem Zähler ein Takt mit der Periode $T_o = k \cdot \tau$ abgeleitet. Der Zähler hat den Anfangswert k, und er wird bei jedem Durchlaufen der TBR um eins erniedrigt. Wenn er den Wert Null erreicht hat, wird er wieder auf k gesetzt und es wird eine Marke gesetzt, die anzeigt, daß das ORG6 aktiviert werden muß.

Nachdem die Zählerstände aktualisiert sind, werden die Meldungen und Befehle aus den Puffern und Warteschlangen umgespeichert. Dabei wird die entsprechende Marke für das ORG1 bzw. für das ORG7 gesetzt, wenn Meldungen eingetroffen sind und das betreffende ORG zu aktivieren ist.

In der Unterbrechungsausgangsroutine (s. Abschnitt 3.10) werden dann, falls Marken für zu aktivierende Organisationsprogramme gesetzt sind, diese aktiviert und dann die Task mit der höchsten Priorität fortgesetzt. Wenn keine ORG zu aktivieren sind , wird die durch das Taktsignal unterbrochene Befehlsfolge fortgesetzt.

Die Notwendigkeit der Aktivierung der Organisationsprogramme in der Unterbrechungsausgangsroutine UAR wird im nächsten Abschnitt behandelt.

3.5.2.2 Aktivierung eines Organisationsprogrammes

Die einfachste Art, ein Organisationsprogramm (ORG) zu aktivieren, besteht darin, die Task, welche dem betreffenden ORG zugeordnet ist, in die Taskwarteschlange (TWS) durch Setzen des entsprechenden Bits einzutragen. Zu diesem Zweck sind die folgenden drei Befehle notwendig:

Befehl 1 : Einlesen des Bitmusters "Taskwarteschlange (TWS)" in den Akkumulator (TWS→Akk)

Befehl 2 : ODER-Verknüpfung des Akkumulators mit dem neu zu setzenden Bit (ODER Bit i)

Befehl 3 : Abspeichern des Bitmusters aus dem Akkumulator in die Zelle "Taskwarteschlange" (Akk→TWS)

Da nun dieselbe Befehlsfolge z.B. auch in der Überwacherroutine zum Starten einer Task enthalten ist, und diese Befehlsfolge durch ein eintreffendes Taktsignal unterbrochen werden darf, ist das Ergebnis abhängig von der Stelle, an welcher die erste Befehlsfolge unterbrochen wird.

Betrachtet werde nun der Fall, daß die Überwacherroutine zum Starten einer Task zwischen dem 1. und dem 3. Befehl durch ein Taktsignal unterbrochen wird:

Nach der Unterbrechung wird die TBR vom Unterbrechungssystem aktiviert. Wird nun in der TBR auch die Befehlsfolge, welche die drei oben beschriebenen Befehle umfaßt, durchlaufen, so wird durch die TBR ein neues Bit in der TWS eingetragen. Dieses Bit wird allerdings wieder gelöscht, wenn die unterbrochene Befehlsfolge der Überwacherroutine zum Starten einer Task fortgesetzt wird, da das betreffende Bit im Bitmuster der TWS, mit dem die Überwacherroutine arbeitet, noch nicht enthalten war.
Der eben beschriebene Sachverhalt ist im Bild 3.27 an einem Beispiel dargestellt, wobei durch die Überwacherroutine zum Starten einer Task das Bit 4 in der TWS und durch die TBR das Bit 3 gesetzt werden soll. In diesem Fall ist das Bit 3, das durch die TBR gesetzt wurde, nach dem Abschluß der Befehlsfolge durch die Überwacherroutine zum Starten einer Task nicht mehr in der TWS vorhanden.

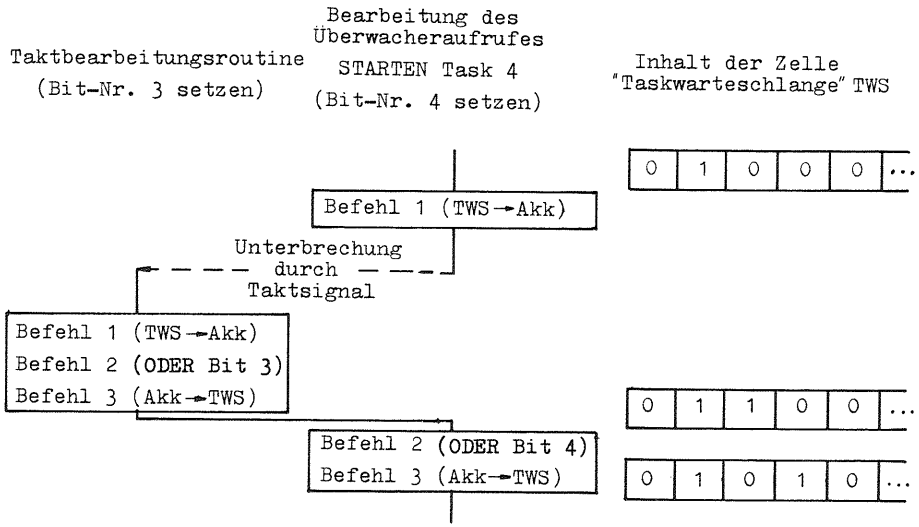


Bild 3.27: Konfliktsituation beim Setzen von Bits in der Taskwarteschlange TWS

Um nun zu verhindern, daß ein Eintrag in der TWS, wie eben beschrieben, "verlorengeht", darf die Befehlsfolge in der TBR erst durchlaufen werden, wenn die entsprechende Befehlsfolge in der Überwacherroutine zum Starten einer Task abgeschlossen ist.

Zu diesem Zweck müßte ein "Mechanismus zur gegenseitigen Verriegelung" programmiert werden, der im Bild 3.28 dargestellt ist. Dieses Lösungsprinzip ist bekannt unter dem Stichwort "critical regions".

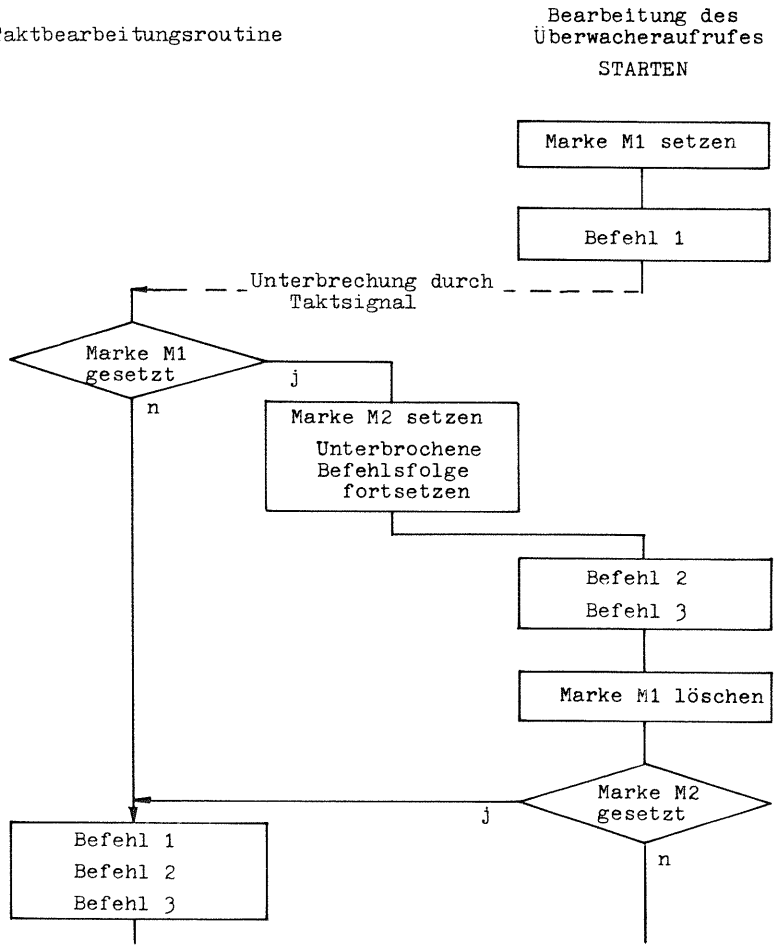


Bild 3.28: Synchronisationsmechanismus zur Vermeidung der Konfliktsituation nach Bild 3.27

Bevor in der Überwacherroutine zum Starten einer Task die betreffenden drei kritischen Befehle durchlaufen werden, wird eine Marke M1 gesetzt. Wenn nun eine Unterbrechung stattfindet, kann die TBR feststellen, daß sie die unterbrochene Befehlsfolge fortsetzen muß, bevor sie selbst weiterlaufen kann. Die TBR setzt daher eine Marke M2, die der anderen Überwacherroutine anzeigt, daß sie die Kontrolle an die TBR zurückgeben muß, wenn die drei kritischen Befehle ausgeführt sind.

Ein derartiger Programmentwurf ist möglich, wenn eine gegenseitige Beeinflussung mit nur einer anderen Bearbeitungsroutine im Überwacher vorliegt.

Die TBR soll aber die Möglichkeit bieten, Tasks nicht nur durch Setzen deren Bits in der Taskwarteschlange zu aktivieren, sondern auch durch Senden einer Nachricht von der TBR an die betreffenden Tasks (vgl. Abschnitt 4.4).

Weiterhin soll für eventuelle Erweiterungen die Möglichkeit bestehen, auch andere Funktionen des Überwachers aufzurufen. Zu diesem Zweck müßte in allen Bearbeitungsroutinen des Überwachers ein Verriegelungsmechanismus entsprechend Bild 3.28 einprogrammiert werden. Eine derartige Programmierung ist sehr komplex und sie birgt daher die Gefahr in sich, daß eine gegenseitige Beeinflussung nicht beachtet wird und dadurch zu Fehlern führen kann, die nur sehr schwer zu lokalisieren sind.

Eine wesentlich einfachere und übersichtlichere Art dieses Problem zu lösen, ist die Benutzung der Bearbeitungsroutinen des Überwachers. Der Aufruf der Überwacherfunktionen kann auf zwei Arten erfolgen:

- durch den Umschaltbefehl von einer Anwendertask aus (Normalzustand NZ des Prozessors, vgl. Abschnitt 3.2.2)
- durch einen Unterprogrammsprung bei systeminternen Aufrufen (Privilegierter Zustand PZ des Prozessors, vgl. Abschnitt 3.2.5)

Die Aufrufbearbeitung muß stets im Zustand PZ erfolgen, da u.U. Befehle ausgeführt werden, die nur in diesem Zustand erlaubt sind, und weil im Alarmzustand(AZ) die Adressen in den Befehlen anders interpretiert werden. Aus diesem Grund ist ein Überwacheraufruf in der TBR über einen einfachen Unterprogrammsprung, wie bei den systeminternen Aufrufen, nicht möglich.

Auch der Aufruf mit dem Umschaltbefehl ist nicht möglich, da bei der Ausführung dieses Befehls der Prozessor in jenen Zustand umschaltet, welcher vor der Unterbrechung durch das Taktsignal vorlag, also in den Zustand PZ oder NZ.

Eine mögliche Realisierung besteht darin, in einer Merkzelle, die stets von der Bearbeitungsroutine für Kanalanforderungen (vgl. Abschnitt 3.4.3) abgefragt wird, die Anforderung für die Aufrufbearbeitung einzutragen und eine Kanalanforderung durch einen speziellen Befehl hervorzurufen. Im Anschluß daran kann die durch das Taktsignal unterbrochene Befehlsfolge fortgesetzt werden.

Die von der TBR veranlaßte Unterbrechung durch eine solche Kanalanforderung wird aber erst wirksam, wenn in der gerade laufenden Anwendertask eine Unterbrechbarkeitsstelle auftritt. Das Eintreten der Unterbrechung kann dadurch sehr lange verzögert werden, was nicht zulässig ist, da bei eintreffenden Fehlermeldungen von der Peripherie das ORG1 möglichst sofort aktiviert werden muß. Falls z.B. in einem laufenden Arbeitsprogramm eine Endlosschleife auftreten würde, könnte kein ORG mehr aktiviert werden.

Aus diesen Gründen kann auch diese spezielle Art der Aktivierung von Bearbeitungsroutinen des Überwachers nicht realisiert werden.

Die einzige realisierbare Möglichkeit, in der TBR Überwacheraufrufe abzusetzen, besteht darin, den Prozessor bei Bedarf in den Zustand PZ umzuschalten und dann wie bei systeminternen Aufrufen die Aufrufbearbeitung über einen Unterprogrammsprung anzuspringen. Diese Umschaltung des Prozessors wird erst im Abschnitt 3.10.2 behandelt, da hierzu weitere Kenntnisse aus späteren Abschnitten notwendig sind.

3.5.2.3 Bearbeitungsroutinen für Überwacheraufrufe

Zur Übernahme von Meldungen aus den Eingabewarteschlangen in eine Task (vgl. Bild 3.25) und zur Übergabe von Befehlen aus einer Task in die Ausgabewarteschlangen stehen folgende Überwacheraufrufe zur Verfügung:

WSEF Eingabe einer Meldung aus der Warteschlange EWSF

WSER Eingabe einer Meldung aus der Warteschlange EWSR

WSAF Ausgabe eines Befehls in die Warteschlange AWSF

WSAR Ausgabe eines Befehls in die Warteschlange AWSR

Die Informationsübergabe erfolgt für alle Aufrufe im linken Akkumulator.

Die Eingabe einer Meldung wird am Beispiel des Aufrufes WSER beschrieben; sie läuft für den Aufruf WSEF in derselben Weise ab. Wenn z.B. die Task ORG7 (vgl. Abschnitt 4.5.4) eine Meldung aus der Eingabewarteschlange EWSR zur Bearbeitung übernehmen will, durchläuft sie

den Überwacheraufruf WSER. Damit wird der Überwacher aktiviert und dessen Bearbeitungsroutine (für den Aufruf WSER) trägt in den Speicherplatz für den linken Akkumulator im Taskkontrollblock (TKB) (s. Abschnitt 3.3.3.2) der Task ORG7 eine wartende Meldung aus der Eingabewarteschlange EWSR ein. (Nach der Ausführung des Aufrufes durch den Überwacher wird die aufrufende Task mit dem Inhalt des TKB fortgesetzt.) Wenn die Eingabewarteschlange zu dem Zeitpunkt leer ist, wird die Zelle für den linken Akkumulator im TKB gelöscht und dadurch der Task angezeigt, daß die Warteschlange keine Meldungen mehr enthält. (Das Nullwort ist als Meldung ausgeschlossen.)

Bei dem Eingabeaufruf WSER wird von der Bearbeitungsroutine des Überwachers gleichzeitig im rechten Akkumulator die Systemzeit übergeben. Diese Systemzeit wird in den Vermittlungsprogrammen benötigt zur Bestimmung der Gesprächsdauer für ein Fernsprechgespräch. (Die Gesprächsdauer wird als Zeitdifferenz zwischen dem Gesprächsbeginn und dem Gesprächsende bestimmt.) Die Übergabe der Systemzeit bei dem Überwacheraufruf WSER erwies sich als zweckmäßig, da die Systemzeit in der Regel nur bei der Bearbeitung von Meldungen benötigt wird und damit schon zusammen mit der Meldung vorliegt; sie muß nicht erst bei Bedarf durch einen zusätzlichen Überwacheraufruf übernommen werden. Die zusätzlichen Überwacheraufrufe zur Übernahme der Systemzeit bei Bedarf würden insgesamt mehr Rechenzeit beanspruchen als die jeweilige Übergabe bei dem Aufruf WSER, selbst wenn die Systemzeit zur Bearbeitung der gerade übergebenen Meldung nicht benötigt wird. Für die Zeitbestimmung in den Vermittlungsprogrammen besteht, wegen der Zeiteinheit von 10ms (vgl. Abschnitt 3.5.2.1), kein Unterschied ob die Systemzeit gleich mit der Meldung übernommen wird oder erst, wenn sie im Programm benötigt wird.

Die Ausgabe eines Befehls in eine Ausgabewarteschlange wird am Beispiel des Aufrufes WSAR beschrieben; sie läuft für den Aufruf WSAF in derselben Weise ab. Wenn z.B. die Task ORG7 einen Befehl an eine DEST ausgeben will, trägt sie den Befehl in den linken Akkumulator ein und durchläuft den Überwacheraufruf WSAR. Die Bearbeitungsroutine für den Aufruf WSAR übernimmt den Befehl und trägt ihn in die Ausgabewarteschlange AWSR ein, wenn diese noch über freie Warteplätze verfügt. Der Speicherplatz für den linken Akkumulator im TKB wird gelöscht, um der aufrufenden Task anzuzeigen, daß der Befehl vom Überwacher übernommen wurde. Wenn dagegen die Ausgabewarteschlange zu dem Zeitpunkt bereits voll ist, bleibt der Befehl im linken Akkumulator erhalten, um der Task den Warteschlangenüberlauf anzuzeigen. (Der Befehl geht damit nicht verloren.)

3.5.3 Warteschlangenorganisation

Die Meldungen, die von der Vermittlungsperipherie kommen, müssen gemäß Abschnitt 3.5.1 bis zu ihrer Verarbeitung in Eingabewarteschlangen zwischengespeichert werden. Entsprechendes gilt für die Befehle an die Peripherie, die vom Zeitpunkt ihrer Generierung bis zur eigentlichen Ausgabe zum Taktzeitpunkt in Ausgabewarteschlangen zwischengespeichert werden.

Die Meldungen von der Peripherie müssen in der Reihenfolge ihres Eintreffens bearbeitet werden, damit die "logische" Reihenfolge der Meldungen erhalten bleibt. Z.B. wenn ein Teilnehmer abhebt und sofort mit der Wahl beginnt, so kann es vorkommen, daß die beiden Meldungen "Tln hat abgehoben" und "Tln wählt Ziffer" unmittelbar hintereinander in die Warteschlange EWSR eingetragen werden. Würde nun von den beiden Meldungen die Meldung "Tln wählt Ziffer" vom Vermittlungsprogramm **zuerst bearbeitet, so müßte dieses mit einer Fehlermeldung reagieren, denn bevor ein Teilnehmer mit der Wahl beginnt, muß er auch abgehoben haben (logischer Ablauf einer Verbindung).**

Aus diesem Grund werden die Eingabewarteschlangen nach der Disziplin FIFO (first-in, first-out) abgearbeitet. Entsprechendes gilt für die Ausgabewarteschlangen.

3.5.3.1 Eingabewarteschlange

Das Prinzip der Eingabewarteschlangenbearbeitung wird am Beispiel einer Warteschlange aufgezeigt; sie ist für beide Eingabewarteschlangen (Fehlermeldungen, reguläre Meldungen) identisch. Es wird eine zyklische Organisationsform zugrunde gelegt, wobei ein Schreibzeiger (SZ) für das Einschreiben und ein Lesezeiger (LZ) für das Auslesen verwendet wird. Wenn einer der beiden Zeiger LZ oder SZ das Warteschlangenende erreicht, wird er wieder auf den Anfang gesetzt. Zusätzlich zeigt ein Merkzähler an, wieviel Eintragungen in der Warteschlange sind.

Die Struktur der Warteschlange ist im Bild 3.29 angegeben; dabei seien gerade 2 Meldungen in der Warteschlange.

Die Flußdiagramme für das Ein- und Austragen von Meldungen in bzw. aus den Warteschlangen sind in den Bildern 3.30 und 3.31 angegeben. Das Eintragen der Meldungen in die Eingabewarteschlangen (Bild 3.31) wird durch die Taktbearbeitungsroutine durchgeführt und das Austragen der Meldungen (Bild 3.30) durch die Bearbeitungsroutine für die Überwacheraufrufe WSER bzw. WSEF.

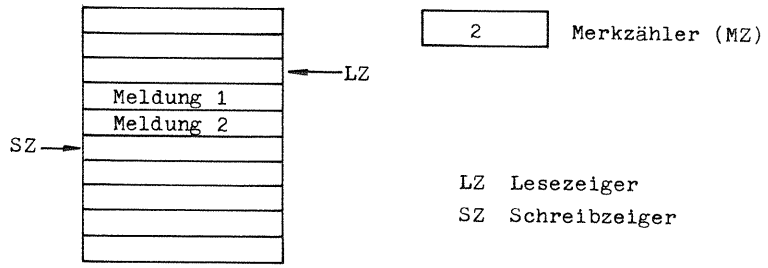
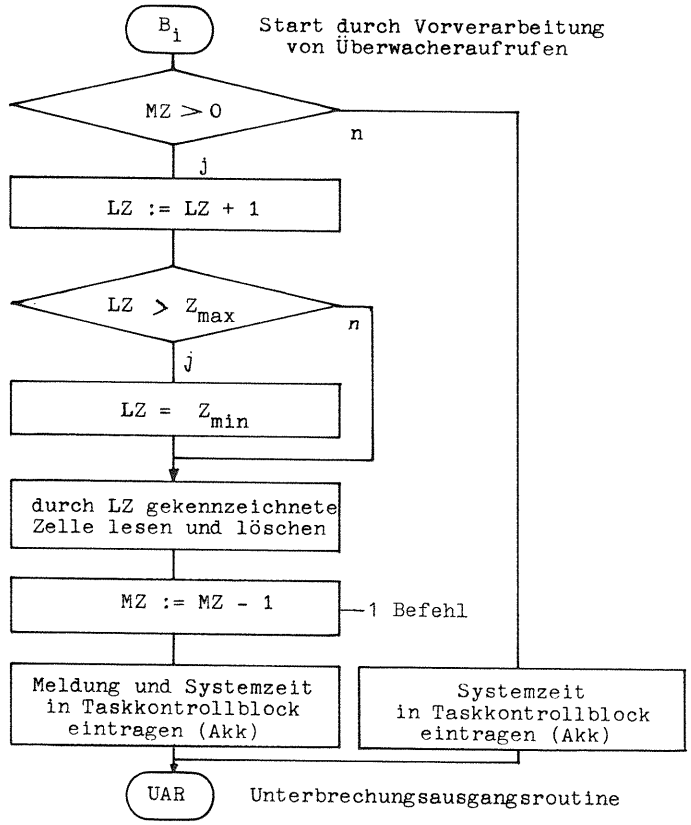
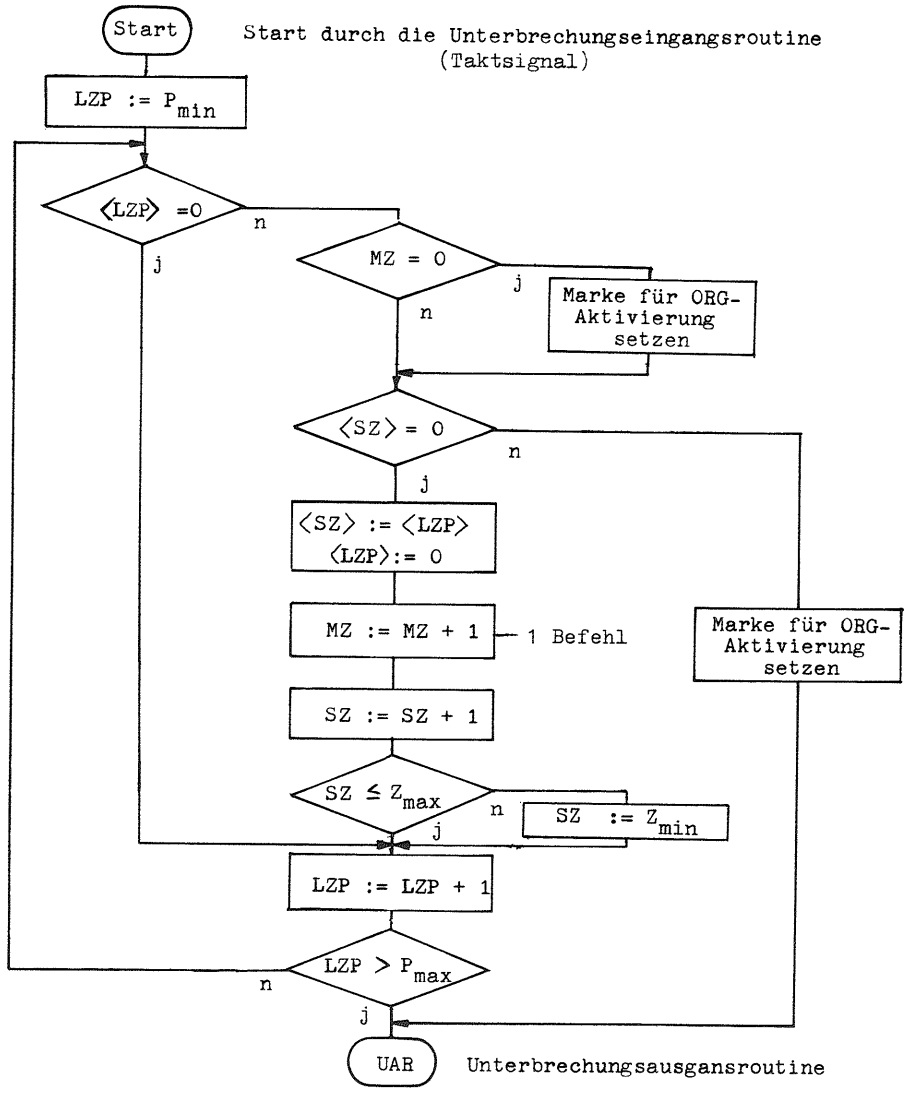


Bild 3.29: Struktur der Eingabewarteschlange



MZ Merkzähler Z_{max} letzte Zelle der Warteschlange
LZ Lesezeiger Z_{min} erste Zelle der Warteschlange

Bild 3.30: Flußdiagramm für das Auslesen einer Meldung aus einer Eingabewarteschlange EWS



LZP	Lesezeiger für Eingabepuffer	⟨Z⟩	Inhalt der durch den Zeiger Z gekennzeichneten Zelle
MZ	Merkzähler für Zahl der Eintragungen in der Warteschlange	SZ	Schreibzeiger
P_{min}	erste Zelle des Eingabepuffers	Z_{max}	letzte Zelle der Warteschlange
P_{max}	letzte Zelle des Eingabepuffers	Z_{min}	erste Zelle der Warteschlange

Bild 3.31: Flußdiagramm für das Umspeichern von Meldungen aus dem Eingabepuffer in die Eingabewarteschlange

Die beiden Bearbeitungsroutinen benutzen eine Datenzelle, den Merkzähler MZ gemeinsam. Da aber zur Änderung dieses Merkzählers(vgl. Bild 3.30 und 3.31) nur ein Befehl notwendig ist, kann keine gegenseitige Beeinflussung auftreten, wenn die Bearbeitungsroutine für den Überwacheraufruf WSER bzw. WSEF durch die TBR unterbrochen wird. Die TBR kann die andere Bearbeitungsroutine des Überwachers unterbrechen, da sie durch das Taktsignal ("Alarmsignal") aktiviert wird.

3.5.3.2 Ausgabewarteschlange

Das Prinzip der Ausgabewarteschlange wird ebenfalls am Beispiel einer Warteschlange aufgezeigt. Die Organisation ist wieder für beide Warteschlangen (AWSF und AWSR) identisch. Die Organisationsform ist in folgendem Bild dargestellt.

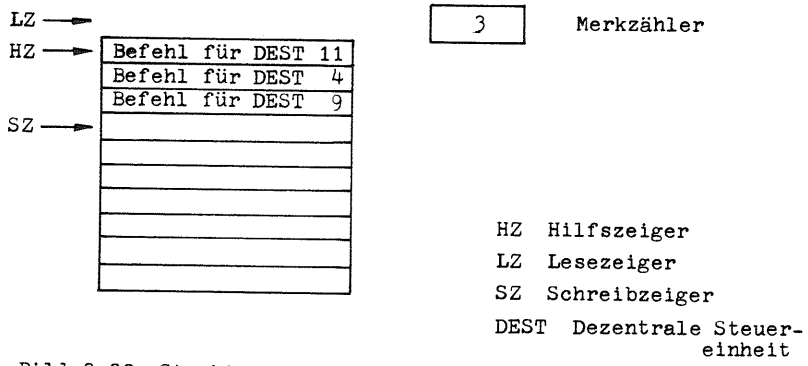


Bild 3.32: Struktur der Ausgabewarteschlange

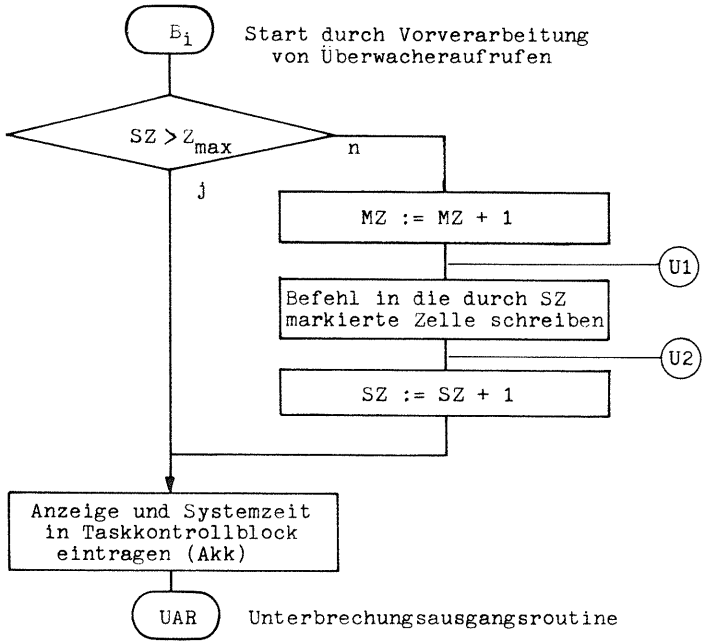
Die Warteschlange ist linear organisiert, d.h. es wird immer von vorne beginnend eingeschrieben und ausgelesen. Das Einschreiben erfolgt ab der durch den Schreibzeiger gekennzeichneten Stelle. Ein Austrag aus der Warteschlange kann nur erfolgen, wenn im Ausgabepuffer die zum Befehl gehörende Speicherzelle frei ist.

Im Beispiel kann der erste Befehl an die DEST Nummer 11 nur ausgetragen werden, wenn im Ausgabepuffer der 11. Speicherplatz frei ist.

Die Ausgabewarteschlange wird zum Taktzeitpunkt von der ersten belegten Zelle bis zur ersten freien Zelle abgesucht. Da unter Umständen nicht alle Befehle ausgetragen werden können, können in der Warteschlange "Löcher" (leere Zellen) entstehen. Würde nun ein neu eintreffender Befehl auf irgendeinen freien Platz eingetragen, so könnte dieser Befehl zeitlich vor einem "älteren" Befehl an die gleiche DEST, der noch weiter hinten in der Warteschlange steht, ausgetragen werden.

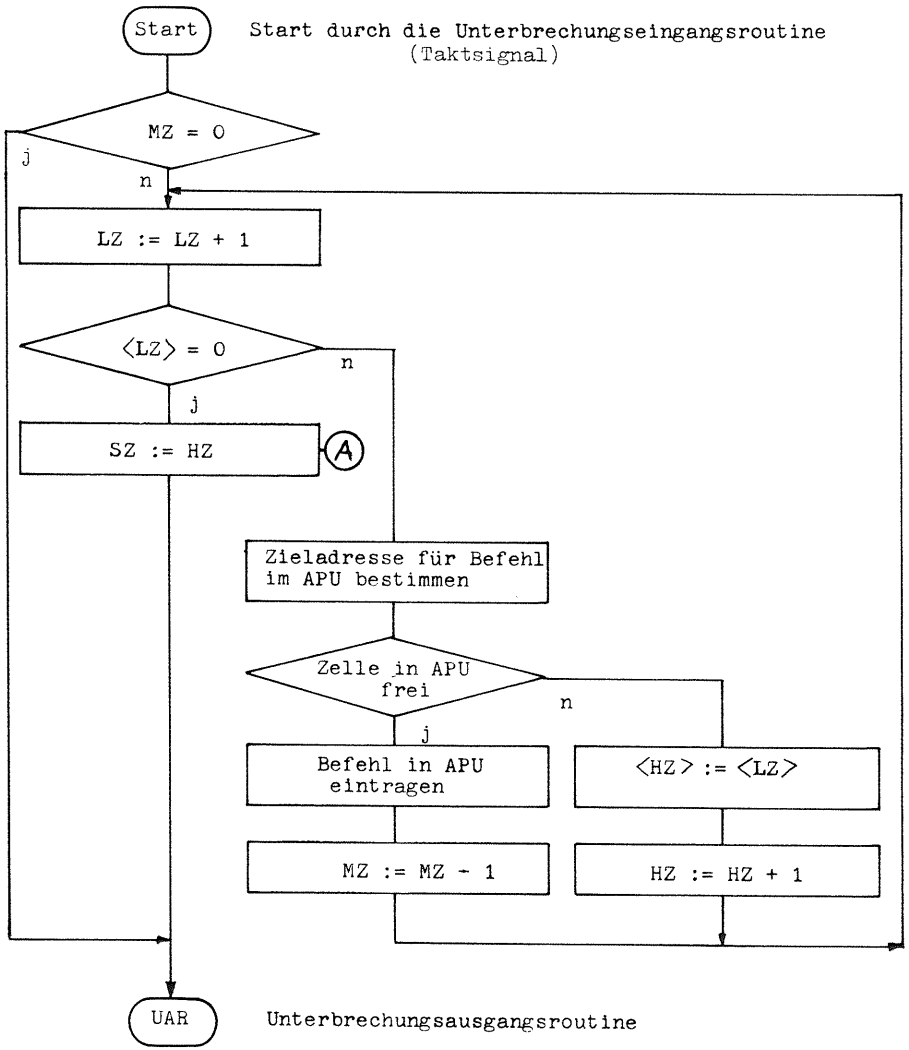
Um dies zu vermeiden, wird die Warteschlange stets wieder dichtgepackt, d.h. nicht austragbare Befehle werden mit Hilfe des Hilfszeigers in der Warteschlange nach vorne geschoben.

Die Flußdiagramme für das Ein- und Austragen von Befehlen sind in den beiden nächsten Bildern 3.33 und 3.34 angegeben. Das Eintragen von Befehlen (Bild 3.33) erfolgt durch die Bearbeitungsroutine für den Überwacheraufruf WSAR bzw. WSAF. Das Austragen der Befehle (Bild 3.34) wird durch die Taktbearbeitungsroutine (TBR) vorgenommen.



MZ Merkzähler für Zahl der Eintragungen in der Warteschlange
LZ Lesezeiger
SZ Schreibzeiger
Z_{max} letzte Zelle der Warteschlange

Bild 3.33: Flußdiagramm für das Eintragen eines Befehls in die Ausgabewarteschlange



APU Ausgabepuffer
HZ Hilfszeiger
MZ Merkzähler
LZ Lesezeiger
SZ Schreibzeiger
<Z> Inhalt der Zelle Z

Anfangswerte der Zeiger s. Bild 3.32

Bild 3.34: Flußdiagramm für das Umspeichern von Befehlen aus der Ausgabewarteschlange in den Ausgabepuffer

Da nun die Bearbeitungsroutine für den Überwacheraufruf WSAR (bzw. WSAF) beim Eintragen eines Befehles in die Ausgabewarteschlange durch die TBR unterbrochen werden kann, und die Zeiger, die dort verwendet werden, auch in der TBR zum Umspeichern der Befehle verändert werden, liegt eine gegenseitigen Beeinflussung vor, die synchronisiert werden muß.

Das im Abschnitt 3.5.2.2 beschriebene und implementierte Verfahren zum systeminternen Aufruf von Überwacherfunktionen kann hier nicht angewendet werden, weil dazu jedesmal zu viel Zeit für die Aufrufvorverarbeitung benötigt würde (Overhead!).

Eine sehr einfache Art der Synchronisation ist im Flußdiagramm (Bild 3.35) dargestellt:

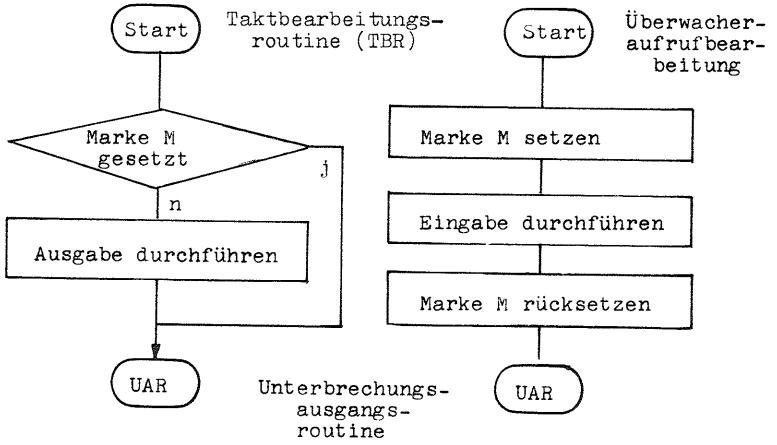


Bild 3.35: Synchronisation bei der Behandlung von Befehlen

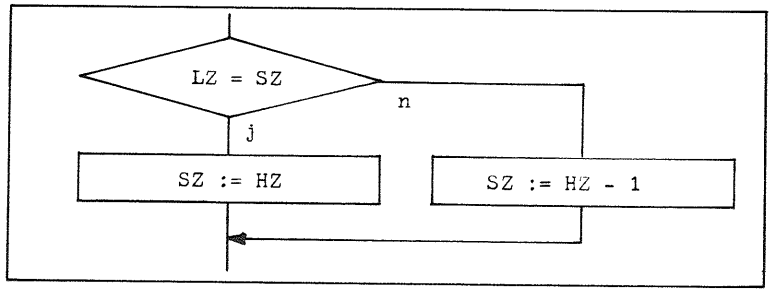
Bei der Aufrufbearbeitung für die Überwacheraufrufe WSAR bzw. WSAF wird in einer hierfür reservierten Zelle eine Marke M gesetzt, wenn die Eingabe eines Befehles in die entsprechende Ausgabewarteschlange (AWS) durchgeführt wird. Da nun diese Befehlsfolge durch ein Taktsignal unterbrochen werden kann, wird in der TBR diese Marke M abgefragt. Falls sie gesetzt ist, werden keine Befehle aus der AWS in den APU umgespeichert. Die Ausgabe des gesamten Inhalts der AWS wird also durch diesen Synchronisationsmechanismus jedesmal um eine volle Taktperiode (10ms) verzögert, wenn zum Taktzeitpunkt gerade ein Befehl in die AWS eingetragen wird. Damit die Zeit, in der eine Ausgabe durch die TBR blockiert wird, möglichst kurz ist, müssen für die beiden AWS getrennte Marken verwendet werden.

Dieser einfache Synchronisationsmechanismus hat aber den Nachteil, daß eine Blockierung der Ausgabe stattfinden kann und dadurch die AWS mehr Einträge aufnehmen muß, was wegen deren begrenzter Länge zu einem Warteschlangenüberlauf führen kann.

Dieser Nachteil der eventuellen um 10ms verzögerten Übergabe des Inhaltes der AWS an die Peripherie kann durch folgende Lösung vermieden werden:

In die TBR (Bild 3.34) wird eine zusätzliche Abfrage eingebaut, an der erkannt wird, daß die Bearbeitungsroutine für den Überwacheraufruf WSAR bzw. WSAF beim Einschreiben eines Befehles in die AWS durch das Taktsignal unterbrochen wurde. Kritische Stellen für die Unterbrechung der Einschreiboperation sind die durch die Marken U1 und U2 im Flußdiagramm Bild 3.33 markierten Stellen, da an diesen Stellen Zeiger verändert werden und die Operationen, die eine Veränderung notwendig machen (z.B. Eintragen des Befehles in die entsprechende Zelle der AWS) noch nicht ausgeführt sind.

Die Erkennung einer Unterbrechung der Befehlsfolge für das Eischreiben eines Befehles in eine AWS im Flußdiagramm (Bild 3.33) an der Stelle U2 kann durch eine zusätzliche Abfrage im Flußdiagramm (Bild 3.34) der TBR durchgeführt werden.Diese Abfrage muß vor der Ausführung der Befehlsfolge, die durch die Marke A gekennzeichnet ist, durchgeführt werden. Zu diesem Zweck wird die durch die Marke A gekennzeichnete Befehlsfolge durch die Befehlsfolge nach Bild 3.36 ersetzt.



HZ Hilfszeiger
LZ Lesezeiger
SZ Schreibzeiger

Bild 3.36: Erkennung von Unterbrechungen und Maßnahmen zur Synchronisation

Im Normalfall hat, wenn das erste Nullwort in der AWS erkannt wird, der Lesezeiger (LZ) denselben Stand wie der Schreibzeiger (SZ). Bei einer Unterbrechung an der Stelle U2 (Bild 3.33) wurde der Befehl bereits in die AWS eingetragen, der Schreibzeiger aber noch nicht erhöht, d.h. LZ > SZ (Ausgang nein bei der Abfrage, Bild 3.36). In diesem Fall muß der SZ um eins kleiner gesetzt werden als der Hilfszeiger (HZ). Wenn die Bearbeitungsroutine für den Überwacheraufruf WASR bzw. WSAF fortgesetzt wird, wird der SZ dann automatisch an die richtige Stelle gerückt (durch Erhöhung um eins, Bild 3.33).

Eine Unterbrechung an der Stelle U1 (Bild 3.33) wird nicht erkannt; sie führt auch zu keinen falschen Ergebnissen. Sie kann lediglich dazu führen, daß bei leerer AWS, wenn der Merkzähler (MZ) aber schon erhöht ist, einmal der Ausgabeteil der TBR durchlaufen wird, aber kein Befehl gefunden wird, obwohl der MZ den Wert eins hat.

Von den beiden Vorschlägen zur Erkennung einer Unterbrechung der Bearbeitungsroutine für den Aufruf WSAR bzw. WSAF durch die TBR wurde der Vorschlag mit der zusätzlichen Abfrage nach Bild 3.36 realisiert.

3.5.4 Funktionen zur Simulation der Ein-Ausgabe zur Vermittlungsperipherie

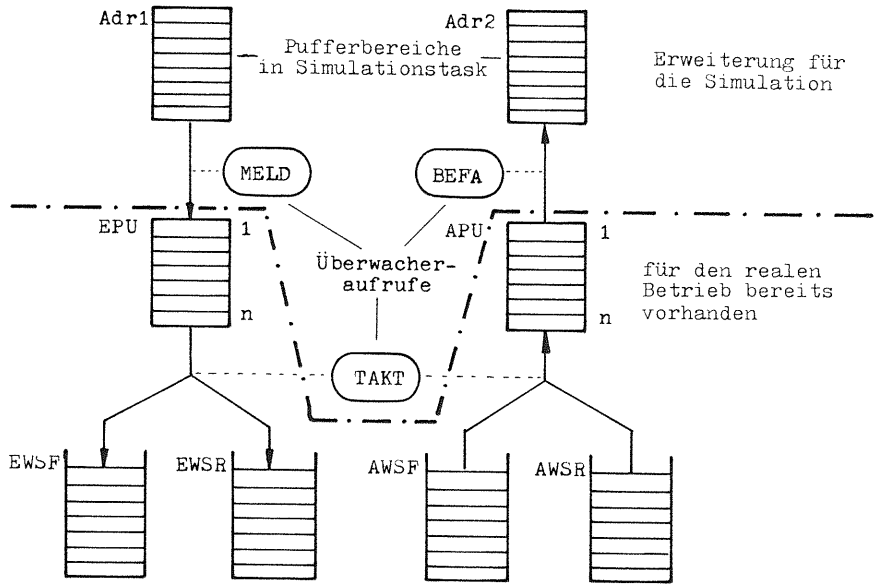
3.5.4.1 Allgemeines

Die Entwicklung der Software bei der Realisierung komplexer Systeme läuft meist parallel zur Hardwareentwicklung. Es ist daher wünschenswert Mittel zu besitzen, mit deren Hilfe während der Softwareentwicklung die Hardware simuliert werden kann.

In dem vorliegenden Vermittlungssystem ist die vermittlungstechnische Peripherie über die beiden Ein-Ausgabepuffer (EPU,APU) mit dem Steuerrechner gekoppelt. Wenn man nun von einer speziellen "Simulationstask" Meldungen "quasi von außen" in den Eingabepuffer des Rechners einschreiben kann und wie der E/A-Multiplexer die Befehle aus dem Ausgabepuffer lesen kann, ist es möglich, durch diese Task die "Hardware-Peripherie" zu simulieren.

3.5.4.2 Funktionen zur Simulation

Die für die Ein-Ausgabesimulation notwendigen Funktionen sind im Bild 3.37 dargestellt.



APU	Ausgabepuffer	EPU	Eingabepuffer
AWSF	Ausgabewarteschlange für Befehle zur Fehlerbearbeitung	EWSF	Eingabewarteschlange für Fehlermeldungen
AWSR	Ausgabewarteschlange für reguläre Befehle	EWSR	Eingabewarteschlange für reguläre Meldungen

TAKT Überwacheraufruf zum Umspeichern von Meldungen / Befehlen
MELD " zur Eingabe von Meldungen in EPU
BEFA " " Ausgabe von Befehlen aus APU

Bild 3.37: Funktionen zur Simulation der Ein-Ausgabe zur Vermittlungsperipherie

Zur Eingabe von Meldungen aus der Simulationstask in den Eingabepuffer des Überwachers ist der Überwacheraufruf MELD vorgesehen. Zur Ausgabe der Befehle aus dem Ausgabepuffer des Überwachers in die Simulationstask dient der Aufruf BEFA. Das Umspeichern der Meldungen von dem Eingabepuffer in die Eingabewarteschlangen und der Befehle von den Ausgabewarteschlangen in den Ausgabepuffer, das im realen Betrieb von der Taktbearbeitungsroutine (vgl. Abschnitt 3.5.2.1) vorgenommen wird, übernimmt der Aufruf TAKT.

Die einzelnen Überwacheraufrufe werden im folgenden ausführlich beschrieben:

MELD(Adr1,n):

Aus dem Pufferspeicher der Simulationstask sind n Meldungen ab der Adresse Adr1 zu entnehmen. Die Meldungen werden entsprechend der DEST-Nummer, die in der Meldung enthalten ist, in den Eingabepuffer (EPU) des Überwachers übernommen. Die Speicherplätze der übernommenen Meldungen im Pufferspeicher der Simulationstask werden gelöscht. Wenn der zur DEST-Nummer gehörende Speicherplatz im EPU bereits belegt ist, wird die Meldung nicht übernommen, um ein Überschreiben von Meldungen im EPU zu verhindern.

BEFA(Adr2):

Alle n Zellen des (realen) Ausgabepuffers(APU) werden in den Pufferspeicher der Simulationstask, der im Programm der Task ab der Adresse Adr2 steht, kopiert. Dabei wird gleichzeitig der APU gelöscht.

TAKT:

Die Meldungen des realen Eingabepuffers EPU werden in die realen Eingabewarteschlangen EWSR bzw. EWSF übernommen. Befehle aus den Ausgabewarteschlangen werden in den Ausgabepuffer übertragen.Ein Überlauf einer Eingabewarteschlange wird der Simulationstask durch Übergabe einer Anzeige im linken Akkumulator angezeigt.

Im Gegensatz zur realen Taktbearbeitungsroutine (TBR) wird ein Organisationsprogramm (ORG) noch nicht aktiviert, wenn Meldungen dafür in der betreffenden Eingabewarteschlange eintreffen und das ORG noch nicht aktiv ist.Es ist dadurch möglich erst mehrere Einträge in die Warteschlangen EWSR bzw. EWSF durchzuführen und dann erst das betreffende ORG durch die Simulationstask zu aktivieren. Am logischen Ablauf des Vermittlungsgeschehens ändert sich dadurch nichts.

Einzelheiten dieser Simulation der Vermittlungsperipherie werden im Kapitel 6 ausführlich behandelt.

3.6 Sicherungsfunktionen

3.6.1 Allgemeines

Bei irregulären Abläufen im Prozessor, die durch Hard- oder Softwarefehler verursacht werden, wird eine Unterbrechung der laufenden Befehlsfolge hervorgerufen. Nach dieser Unterbrechung muß deren Ursache erkannt werden,und von den Sicherungsfunktionen des Überwachers müssen entsprechende Maßnahmen ergriffen werden.

Eine Unterbrechung kann außer bei Fehlern auch auftreten, wenn die Unterbrechungstaste am Bedienungsfeld des Rechners betätigt wird, oder wenn ein Befehl ausgeführt werden soll, der nicht durch Hardware realisiert ist und durch den Überwacher per Software ausgeführt (simuliert) werden soll.

3.6.2 Unterbrechungsvorverarbeitung

Nachdem eine Unterbrechung wirksam wurde, müssen durch die Unterbrechungseingangsroutine (UER, s Abschnitt 3.9) die Akkumulatorinhalte zwischengespeichert werden (s. Bild 3.38). Dann kann die Analyse der Unterbrechung vorgenommen werden.

Es können folgende Unterbrechungsursachen auftreten:

Unterbrechungsursache 1:- Fehlersignal von einem externen Gerät

Unterbrechungsursache 2:- Ein nicht zugelassener Befehl sollte zur Ausführung kommen

- Es wurde versucht, den Inhalt einer Speicherzelle im geschützten Bereich des Speichers anzusprechen
- Es wurde versucht, eine Speicherzelle anzusprechen, die außerhalb des Speichers liegt
- Es wurde versucht, bei einem Befehl mehr als viermal zu substituieren
- Ein nicht realisierter Befehl muß vom Überwacher simuliert werden
- Die Unterbrechungstaste am Bedienungsfeld des Rechners wurde betätigt

Bei Eintreffen eines Fehlersignale von einem Gerät der Standardperipherie (Unterbrechungsursache 1) wurde dieses Gerät mit falschen Befehlen angesprochen. Es handelt sich um keinen technischen Gerätefehler sondern um einen Programmierfehler in einer Ein-Ausgabebearbeitungsroutine. Aus diesem Grund wird keine Fehlerbearbeitungsroutine aktiviert, sondern nur eine Fehlermeldung der Form

System-Fehler Fi

über den Bedienungsblattschreiber ausgegeben, wobei i die Nummer des Gerätes angibt. (s. Bild 3.38) Ferner wird der Prozessor mit dem Stop-Befehl gestoppt. (Der Fehler sollte , wenn überhaupt, nur während der Testphase für den Überwacher auftreten!)

Für die Unterbrechungsursachen 2 werden geeignete Bearbeitungsroutinen aktiviert (s. Bild 3.38).Mit Ausnahme der Unterbrechung durch die Unterbrechungstaste und durch Befehle, die simuliert werden müssen, handelt es sich bei diesen Unterbrechungen um sogenannte "Softwarefehler" (Programmierfehler) in Anwendertasks.

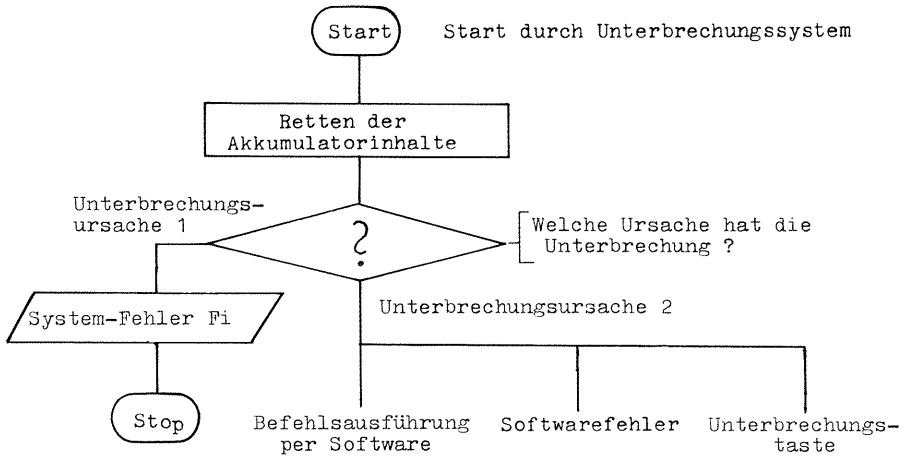


Bild 3.38: Ablauf bei der Unterbrechungsvorverarbeitung

3.6.3 Bearbeitungsroutinen zur Unterbrechungsbearbeitung

3.6.3.1 Befehlsausführung per Software

Bei dem Rechner S306 sind adressarithmetische Befehle, die sich auf Adressen mit gesetztem Kennbit zur Adressierung mit dem Basisadressenregister BAR 15 beziehen, nicht in Hardware realisiert. Diese Befehle müssen durch eine Bearbeitungsroutine des Überwachers entsprechend der in /23/ vorgegebenen Beschreibung per Software ausgeführt (simuliert) werden.

Nach ausgeführter Befehlssimulation werden die Akkumulatoren für die unterbrochene Task wieder geladen und die Task kann über die Unterbrechungsausgangsroutine fortgesetzt werden.

3.6.3.2 Softwarefehler

Bei auftretenden Softwarefehlern wird durch das Unterbrechungssystem in der Zelle Null des Arbeitsspeichers neben dem Befehlszählerstand der unterbrochenen Befehlsfolge auch die Unterbrechungsursache in codierter Form abgespeichert. Aus diesem Code wird eine Fehlernummer abgeleitet, die auf dem Bedienungsblattschreiber als Fehlermeldung ausgegeben wird. Die Fehlermeldung hat folgende Form:

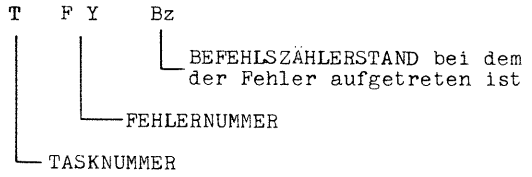


Bild 3.39: Aufbau einer Fehlermeldung

Neben der Fehlernummer wird auch die Nummer der gerade "aktiven" Task und deren Befehlszählerstand ausgegeben.

Die Task wird in den Zustand "bekannt" versetzt, d.h. aus der Taskwarteschlange ausgetragen und die Registerinhalte in den Taskkontrollblock eingeschrieben. Danach erfolgt die Prozessorneuvergabe durch die Unterbrechungsausgangsroutine.

Der Teil der Bearbeitungsroutine für die Ausgabe der Fehlermeldung und für das Abbrechen der betreffenden Task ist als Unterprogramm realisiert (Taskabbruchroutine), so daß er von allen Routinen des Überwachers benutzt werden kann (s. Abschnitt 3.6.4). Die Parameter für den Aufruf des Unterprogrammes sind die Fehler- und die Tasknummer.

Um eine weitere Fehlerdiagnose durchzuführen wird dem ORG1 eine Nachricht übersandt (mittels des Aufrufes SENDE NACHRICHT, vgl. Abschnitt 3.3.5), aus der hervorgeht, daß die Task Nummer i abgebrochen wurde. Falls der Fehler durch die Fehlerbehandlungsprogramme des ORG1 behoben werden kann, kann die unterbrochene Task durch den Überwacheraufruf FORTSETZEN vom ORG1 aus wieder aktiviert werden.

3.6.3.3 Unterbrechungstaste

Anwendertasks, die durch Programmierfehler u.U. eine durch eine Kanalanforderung nicht unterbrechbare Endlosschleife durchlaufen, machen eine Bedienung des Rechners unmöglich . Sie hindern außerdem andere Tasks höherer Priorität, die auf das Ende von Ein-Ausgabeoperationen warten, daran weiterzulaufen.(Eine derartige Endlosschleife wird daran erkannt, daß das Bedienungspersonal keinen Zugriff mehr zum Rechner über den Bedienungsblattschreiber erhält.) Um solche Tasks abzubrechen, kann die Unterbrechungstaste am Bedienungsfeld betätigt werden.

Durch das Betätigen der Unterbrechungstaste wird die UER aktiviert und diese startet die Bearbeitungsroutine für die Unterbrechungstaste. Diese hat die Aufgabe, alle Tasks unterhalb einer bestimmten Priorität P, die schon bei der Generierung des Überwachers vorgegeben wird, abzubrechen (d.h. in den Zustand "bekannt" zu versetzen).Es wird für alle bereiten und die gerade aktive Task, die dadurch abgebrochen werden, eine Meldung auf dem Blattschreiber ausgegeben, wobei durch die Fehlernummer angegeben wird, daß diese Tasks lediglich durch das Betätigen der Unterbrechungstaste abgebrochen wurden. Anschließend erfolgt der Sprung des Überwachers in seine Unterbrechungsausgangsroutine.

Bei dieser Bearbeitung der Unterbrechung muß allerdings beachtet werden, daß die Unterbrechung nach Ausführung des gerade in Bearbeitung befindlichen Befehles wirksam wird, unabhängig davon, in welchem Zustand sich der Prozessor befindet. Es kann vorkommen, daß die UER durch einen Softwarefehler aktiviert wird und bereits beim Retten der Registerinhalte zufällig auch eine Unterbrechung durch Betätigen der Unterbrechungstaste wirksam wird. Es wird dabei die Anzeige der alten Unterbrechungsursache zusammen mit der Fortsetzadresse in der Zelle Null des Arbeitsspeichers überschrieben. Damit ist es unmöglich, den Programmablauf definiert fortzusetzen.

Um diesen Fall auszuschließen, wird eine Bearbeitung des Unterbrechungssignals nur durchgeführt, wenn der Prozessor vor der Unterbrechung im Normalzustand war.Es wird deshalb bei der Bearbeitung (Bild 3.40) zuerst geprüft, in welchem Zustand sich der Prozessor vor der Unterbrechung befand. (Der Zustand des Prozessors vor der Unterbrechung wird ebenfalls in der Zelle Null durch das Unterbrechungssystem gekennzeichnet.) Befand sich der Prozessor im Alarmzustand, so werden sofort die Registerinhalte wieder geladen und die unterbrochene Befehlsfolge fortgesetzt. Falls der Privilegierte Zustand vorlag, wird ebenfalls die unterbrochene Befehlsfolge fortgesetzt, sofern dies möglich ist. Es wird dazu der Stand des Befehlszählers geprüft, ob er auf die UER verweist. Ist dies der Fall, so wird eine Fehlermeldung nach Bild 3.39 auf dem Bedienungsblattschreiber ausgegeben und der Prozessor mit dem Stop-Befehl gestoppt. Die Fehlernummer in der Meldung gibt an, daß die Unterbrechungstaste in einem unzulässigen Zeitpunkt betätigt wurde.

Um diesen Fehler auszuschließen, muß vor dem Betätigen der Unterbrechungstaste der Prozessor gestoppt werden (durch eine Stop-Taste am Bedienungsfeld) und geprüft werden, in welchem Zustand er sich befindet. Die Unterbrechungstaste darf dann nur im Normalzustand betätigt werden.

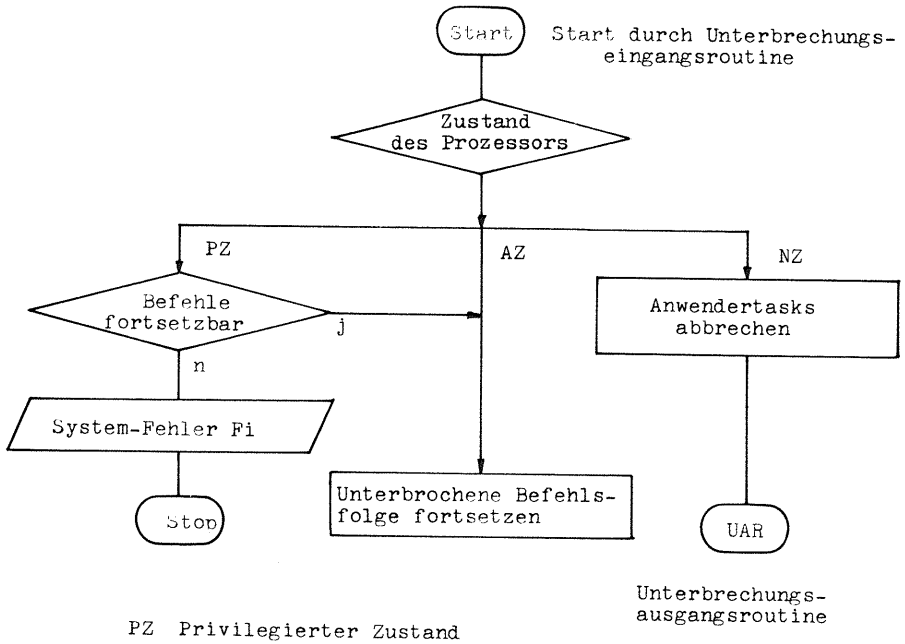
Die Abläufe bei der Bearbeitung des Unterbrechungssignals "Unterbrechungstaste" sind im Bild 3.40 kurz zusammengefaßt.

3.6.4 Fehlerbearbeitungsroutine

Die Fehlerbearbeitungsroutine dient dazu, bei falschen Überwacheraufrufen die aufrufende Task mit Hilfe der Taskabbruchroutine (s. Abschnitt 3.6.3.2) abzubrechen, d.h. aus der Taskwarteschlange auszutragen und dadurch in den Zustand "bekannt" zu versetzen.
Die Fehlerbearbeitungsroutine selbst übernimmt lediglich die Parameterversorgung für den Aufruf der Taskabbruchroutine. (Übergabe der Fehlernummer und der Tasknummer, von der aufrufenden Task).
Die Ausgabe einer Fehlermeldung wird in der Taskabbruchroutine vorgenommen.

Die Fehlerbearbeitungsroutine kann auf zwei Arten aktiviert werden:

- durch direkten Ansprung über die Sprungleiste bei der Aufrufvorverarbeitung (vgl. Abschnitt 3.2.3)
- durch einen Unterprogrammaufruf bei der Ausführung einer Bearbeitungsroutine für Überwacheraufrufe (vgl. Abschnitt 3.2.4)



PZ Privilegierter Zustand
AZ Alarmzustand
NZ Normalzustand

Bild 3.40: Bearbeitung des Unterbrechungssignals
Unterbrechungstaste

Beim direkten Ansprung wird die falsche Aufrufnummer als Fehlernummer ausgegeben und nach dem Durchlaufen der Taskabbruchroutine die UAR angesprungen.

Bei einem Unterprogrammaufruf wurde die Fehlernummer der Fehlerbearbeitungsroutine als Parameter übergeben. Nach dem Durchlaufen der Taskabbruchroutine muß ein Rücksprung zum aufrufenden Bearbeitungsteil erfolgen, da der fehlerhafte Überwacheraufruf u.U. noch aus einer Warteschlange ausgetragen werden muß und die Aufrufbearbeitung für eventuell weitere wartende Aufrufe fortgesetzt werden muß.

3.7 Weitere Dienstfunktionen

Die "Weiteren Dienstfunktionen" stellen Funktionen dar, die von Bearbeitungsroutinen des Überwachers bearbeitet werden, die keinem der anderen Funktionsblöcke des Überwachers nach Bild 3.1 zugeordnet

werden können. Diese Funktionen werden nicht alle für die Systemsoftware der rechnergesteuerten Vermittlungsstelle benötigt. Sie mußten jedoch realisiert werden, damit die Hilfsprogramme und Compiler des Rechnerherstellers weiter Verwendung finden konnten.

Die Funktionen werden in diesem Abschnitt nur pauschal beschrieben. Sie sind jedoch im Anhang A2 einzeln aufgeführt.

Man kann grob zwei Klassen dieser "Weiteren Dienstfunktionen" unterscheiden, nämlich taskbezogene und gerätebezogene.

Zur ersten Klasse gehören die Funktionen zum direkten Lesen von Information aus einer Task und zum Schreiben von Information in eine Task von der aufrufenden Task aus. Ferner gehören dazu Funktionen zur Ermittlung der eigenen Tasknummer und die Abfrage von Zuständen anderer Tasks.

Zur zweiten Klasse gehören die Funktionen zur Abfrage von Schaltern am Bedienungsfeld des Rechners sowie zum Setzen von Anzeigen am Bedienungsfeld. Weiterhin gehören dazu Funktionen zur Abfrage einer Uhr, die an den Rechner angeschlossen ist.

3.8 Ureingabe

Die Ureingaberoutine hat die Aufgabe beim Einschalten des Rechners den Überwacher in den Arbeitsspeicher zu laden und in einen ablauffähigen Zustand zu bringen.

Über den Lochkartenleser wird durch Betätigen der Ureingabetaste automatisch der Informationsinhalt einer Lochkarte in den Arbeitsspeicher ab der Speicherzelle Null eingelesen. Nach Abschluß der Eingabe erfolgt unmittelbar ein Sprung zur Adresse Null. Damit werden die ersten Befehle der Ureingaberoutine aktiviert. Mit diesen Befehlen werden nun die restlichen Befehle der Ureingaberoutine von dem Plattenspeicher eingelesen, die ihrerseits den Überwacher vom Plattenspeicher einlesen und in einen definierten Ausgangszustand bringen.

Dabei wird die Startadresse des Bedienteils in den Taskkontrollblock eingetragen und das betreffende Bit für den Bedienteil in der Taskwarteschlange auf eins gesetzt. Damit ist die Systemtask "Bedienteil" im Zustand bereit und die UAR wird aktiviert. (Der Bedienteil kann dann eventuell das gesamte Vermittlungsprogrammsystem in den Arbeitsspeicher laden.)

Die Befehle der Ureingaberoutine, die vom Plattenspeicher gelesen werden, stehen im Arbeitsspeicher hinter dem Überwacher. Sie werden nachdem der Bedienteil aktiviert ist, nicht mehr benötigt und können deshalb beim Laden von Programmen überschrieben werden.

3.9 Unterbrechungseingangsroutine

Die Unterbrechungseingangsroutine (UER) hat die Aufgabe, die für die Fortsetzung einer unterbrochenen Task notwendigen Registerinhalte des Prozessors zu retten, eine Identifikation des Unterbrechungssignals vorzunehmen und die für die Unterbrechungsursache notwendige Bearbeitungsroutine zu aktivieren.

Die UER besteht aus mehreren Teilen, die durch das Unterbrechungssystem des Prozessors automatisch bei der Unterbrechung aktiviert werden. Die Aufteilung der UER ist im Bild 3.41 angegeben, wobei ihren einzelnen Teilen individuelle Einsprungadressen durch das Unterbrechungssystem des Prozessors zugeordnet sind (vgl. Abschnitt 2.2)

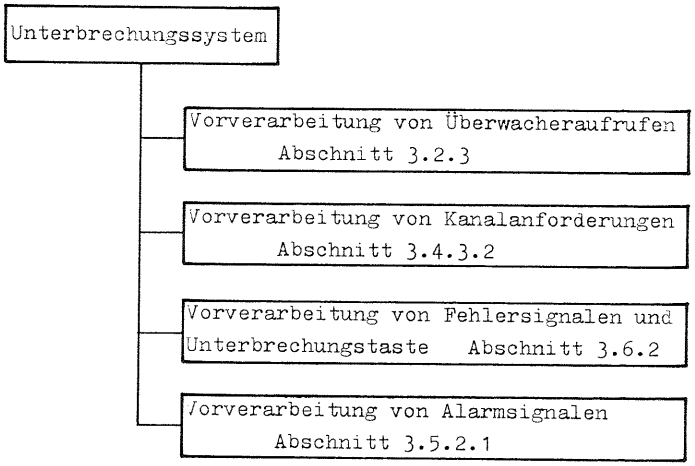


Bild 3.41: Aufteilung der Unterbrechungseingangsroutine

Die verschiedenen Vorverarbeitungsroutinen (Bild 3.41) bestimmen aus dem Unterbrechungssignal und Zusatzinformation die entsprechende Bearbeitungsroutine. Eine Ausnahme bildet die Unterbrechungsvorverarbeitung für Taktsignale, in der nur Registerinhalte gerettet werden. Die einzelnen Teile der UER wurden bereits in vorhergehenden Abschnitten behandelt (s. Bild 3.41).

3.10 Unterbrechungsausgangsroutine

3.10.1 Allgemeines

Die Unterbrechungsausgangsroutine (UAR) hat die Aufgabe, nach dem Ende einer Bearbeitungsroutine des Überwachers der unterbrochenen Task den Prozessor wieder zuzuteilen oder, falls eine Task höherer Priorität "bereit" geworden ist, dieser den Prozessor zuzuteilen. Der Ablauf in der UAR ist im Bild 3.42 dargestellt.

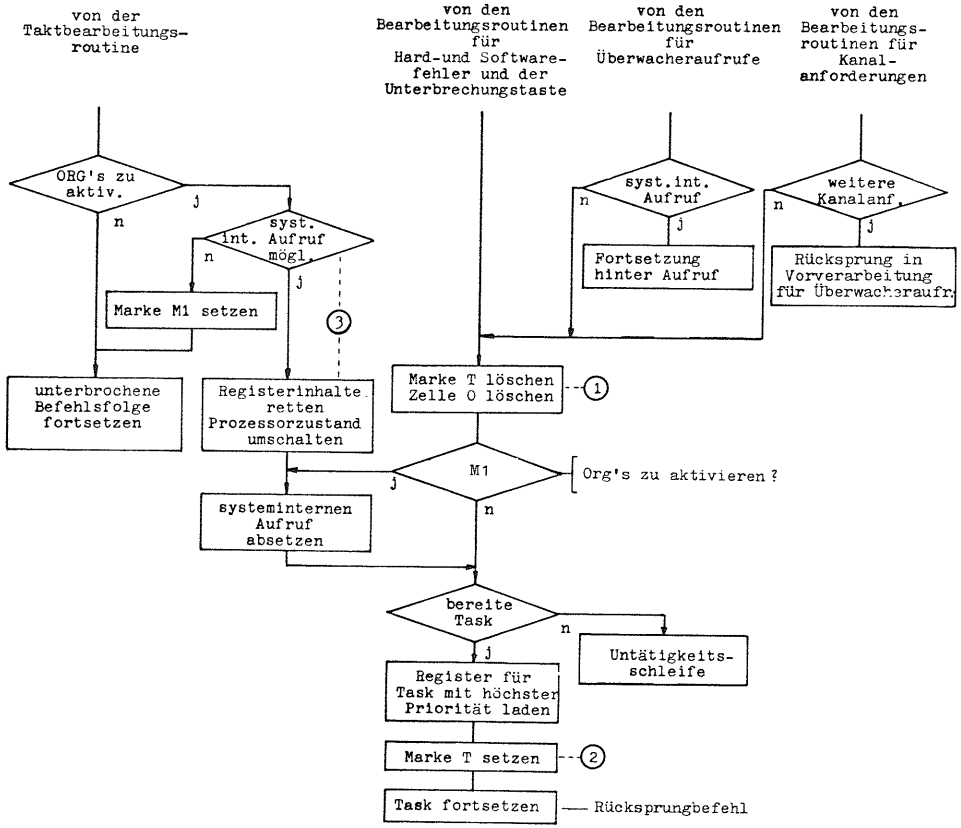


Bild 3.42: Abläufe in der Unterbrechungsausgangsroutine

Der Ablauf zerfällt in zwei Teile, in einen Teil, der den Ausgang aus der Taktbearbeitungsroutine darstellt, und in einen Teil für die Ausgänge aus den restlichen Unterbrechungsbearbeitungsroutinen.

Nach der Bearbeitung eines Taktsignales wird geprüft, ob ein Organisationsprogramm (ORG) mittels eines systeminternen Aufrufes zu aktivieren ist. Falls die Abfrage negativ ausfällt, werden die Register des Prozessors für die unterbrochene Task wieder geladen, und es erfolgt das Umschalten des Prozessors in den Zustand vor der Unterbrechung und die Fortsetzung der unterbrochenen Befehlsfolge.

Falls ORG's zu aktivieren sind (vgl. Abschnitt 3.5.2), wird geprüft ob der Prozessor zum Absetzen von systeminternen Aufrufen in den Zustand PZ umgeschaltet werden darf. (Zum Absetzen von systeminternen Aufrufen muß der Prozessor im Zustand PZ sein, s. Abschnitt 3.2.5.) Die dazu notwendigen Kriterien und das Umschalten selbst werden im nächsten Abschnitt behandelt. Wenn eine Umschaltung des Prozessors nach PZ nicht möglich ist, wird eine Marke M1 gesetzt und wie oben die unterbrochene Befehlsfolge fortgesetzt.

Für die anderen Unterbrechungsbearbeitungsroutinen gibt es einen gemeinsamen Teil. Darin wird nach dem Löschen der Marke T und der Zelle Null anhand der Marke M1 abgefragt, ob ORG's zu aktivieren sind. Wenn dies der Fall ist, werden sie mit dem Aufruf SENDE NACHRICHT (vgl. Abschnitt 4.4) aktiviert, d.h. in den Zustand bereit versetzt. Danach wird geprüft, ob eine bereite Task in der Taskwarteschlange vorhanden ist. Wenn eine oder mehrere bereite Tasks vorhanden sind , wird die Task mit der höchsten Priorität ausgewählt, und es werden die Register des Prozessors für diese Task mit den Inhalten aus dem Taskkontrollblock geladen. Wenn keine bereite Task vorhanden ist, wird in eine Untätigkeitsschleife im Normalzustand des Prozessors gesprungen.

Nach einer Bearbeitungsroutine für Überwacheraufrufe wird jeweils geprüft, ob dieser Aufruf vom Überwacher selbst kam. Wenn dies der Fall ist, wird die Befehlsfolge des Überwachers hinter dem Aufruf fortgesetzt, ansonsten wird die Task höchster Priorität aktiviert.

Nach den Bearbeitungsroutinen für Kanalanforderungen wird geprüft, ob noch weitere Kanalanforderungen vorliegen. Ist dies der Fall, so wird in die Unterbrechungsvorverarbeitungsroutine zurückgesprungen.

Die im Bild 3.42 durch (1) und (2) gekennzeichneten Befehle dienen dazu, der Taktbearbeitungsroutine anzuzeigen, welche Befehlsfolge bei der Unterbrechung durch ein Taktsignal gerade bearbeitet wurde. Die Beschreibung dieser Befehle erfolgt zusammen mit der der durch (3) gekennzeichneten Befehlsfolge im nächsten Abschnitt.

3.10.2 Aktivieren von Organisationsprogrammen

3.10.2.1 Allgemeines

In der Taktbearbeitungsroutine (TBR) kann es erforderlich sein, ein Organisationsprogramm (ORG) zu aktivieren (z.B. das ORG1, weil von der Vermittlungsperipherie eine Fehlermeldung eingetroffen ist, vgl. Abschnitt 3.5.2.1). Die Aktivierung des betreffenden ORG erfolgt über den Überwacheraufruf SENDE NACHRICHT (s. Abschnitt 4.4). Damit nun in der TBR ein systeminterner Aufruf abgesetzt werden kann, muß sich der Prozessor im Privilegierten Zustand PZ befinden, da die Durchführung dieser Aufrufe nur im Zustand PZ möglich ist (vgl. Abschnitt 3.5.2.2).

Als weitere Voraussetzung für das Absetzen eines systeminternen Aufrufes muß vor der Unterbrechung durch das Taktsignal eine Task aktiv oder der Überwacher in der Untätigkeitsschleife gewesen sein. Dies ist notwendig, da für eine spätere Fortsetzung der eventuell durch das Taktsignal unterbrochenen Befehlsfolge die Registerinhalte abgespeichert werden müssen. Für die Bearbeitungsroutinen des Überwachers (außer den Systemtasks) gibt es keinen Taskkontrollblock. Deshalb können diese Befehlsfolgen nicht unterbrochen werden. Außerdem könnte vor der Unterbrechung gerade die betreffende Bearbeitungsroutine desselben Aufrufes aktiv gewesen sein. Ein erneuter Aufruf würde zwangsläufig zu falschen Ergebnissen führen.

Während der Ausführung der TBR befindet sich der Prozessor im Zustand AZ. Im AZ gibt es durch die Hardware des Rechners keine Möglichkeit den Zustand des Prozessors vor der Unterbrechung durch das Taktsignal abzufragen. Ein Kriterium für den Zustand vor der Unterbrechung liefert aber der Inhalt der Zelle Null des Arbeitsspeichers. In dieser Speicherzelle wird bei allen Unterbrechungen, außer jener durch das Taktsignal, der Befehlszählerstand bei der Unterbrechung und Kennbits für die Unterbrechungsursache eingetragen. Da diese Speicherzelle erst in der UAR gelöscht wird, (s. Bild 3.42 Befehl (1)) gibt ihr Inhalt, wenn er nicht das Nullwort enthält, an, daß der Überwacher vom Taktsignal in einer Bearbeitungsroutine unterbrochen wurde. In diesem Fall darf kein systeminterner Überwacheraufruf erfolgen.

3.10.2.2 Retten von Registerinhalten

Wenn der Prozessor vor der Unterbrechung durch ein Taktsignal einer Task zugeteilt war, müssen die Registerinhalte dieser Task in den Taskkontrollblock eingetragen werden. Zur Kennzeichnung, daß der Prozessor einer Task zugeteilt ist, wurde die Marke T eingeführt. Die Marke T wird auf eins gesetzt, bevor der Sprung in die Befehlsfolge der Task erfolgt (s. Bild 3.42 Befehl ②).

Die Marke T allein genügt nicht als Kriterium, daß die Registerinhalte gerettet werden müssen. Es ist nämlich möglich, daß die UAR durch das Taktsignal zwischen dem Befehl ② und dem Rücksprungbefehl (Bild 3.42) unterbrochen wurde. Es muß also zusätzlich der Befehlszählerstand abgefragt werden. Zeigt dieser auf den Rücksprungbefehl, so wurde dieser noch nicht ausgeführt und die Registerinhalte dürfen auch nicht abgespeichert werden. Damit für eine Task, die genau bei diesem Befehlszählerstand (Adresse des Rücksprungbefehles) durch das Taktsignal unterbrochen wurde, aber trotzdem die Registerinhalte gerettet werden, wird zu Kontrollzwecken der Inhalt des Taskkontrollblockes der unterbrochenen Task mit den aktuellen Registerinhalten verglichen. Wenn diese nicht übereinstimmen, müssen sie gerettet werden, da dann diese Task bei dem betreffenden Befehlszählerstand unterbrochen wurde.

Wurde die UAR nach den Befehlen ① und vor dem Befehl ② durch das Taktsignal unterbrochen, so brauchen keine Registerinhalte gerettet zu werden, da nach dem Durchlaufen der Bearbeitungsroutine für einen Überwacheraufruf diese Befehle wieder durchlaufen werden. (Prozessorneuvergabe durch die UAR.)

Nach obigen Ausführungen ist ein Aufruf einer Funktion des Überwachers durch einen systeminternen Aufruf nach dem Durchlaufen der TBR nur möglich, wenn die Arbeitsspeicherzelle Null das Nullwort enthält. Die Registerinhalte für eine Task, die durch das Taktsignal unterbrochen wurde, brauchen nur gerettet zu werden, wenn die Marke T gesetzt ist.

3.10.2.3 Umschalten des Prozessors

Bevor eine Funktion des Überwachers durch einen systeminternen Aufruf aufgerufen werden kann, muß der Prozessor in den Privilegierten Zustand PZ umgeschaltet werden.

Die Umschaltung des Prozessors vom Zustand AZ in den Zustand PZ mit dem Umschaltbefehl ist beim Rechner S306 nur möglich, wenn der Prozessor vor der Unterbrechung durch das Taktsignal bereits im Zustand PZ war. Der Prozessor wird jedoch vorwiegend bei der

Unterbrechung durch Taktsignale im Normalzustand sein. In diesem Fall muß eine Umschaltung auf andere Weise "erzwungen" werden. Dies geschieht mit Hilfe eines nicht interpretierbaren (nicht zugelassenen) Befehls, der automatisch eine Unterbrechung und damit die Umschaltung in den Zustand PZ hervorruft. Vor dem Durchlaufen dieses Befehls muß jedoch die Einsprungadresse für die Unterbrechungsvorverarbeitungsroutine verändert werden, damit nicht regulär diese durchlaufen wird, sondern die UAR fortgesetzt wird. Nach der auf diese Weise erzwungenen Prozessorumschaltung muß jedoch die Änderung der Einsprungadresse wieder rückgängig gemacht werden.

Diese Art der Prozessorumschaltung wird nun grundsätzlich benutzt, da eine Abfrage des Prozessorzustandes, der vor der Unterbrechung durch das Taktsignal vorlag, aufgrund der Kriterien für das Retten von Registerinhalten mehr Befehle notwendig machen würde.

3.11 Systemtask "Lader"

3.11.1 Allgemeines

Der Lader hat die Aufgabe, zum einen das Programm für eine Task von einem peripheren Speichermedium in den Arbeitsspeicher zu laden und zum anderen muß er den Arbeitsspeicher verwalten.

Programme werden durch die Übersetzer des Rechnerherstellers (Assembler, Algol- und Fortran-Compiler) als sogenannte "Grundspracheprogramme" /24/ (Maschinencode) entweder auf Lochkarten oder Lochstreifen ausgestanzt oder in einer Datei auf dem Plattenspeicher abgelegt. Der Überwacher soll allerdings nur Grundspracheprogramme von Lochkarten oder aus einer Datei vom Plattenspeicher laden können. Diese Einschränkung wurde eingeführt, da praktisch keine Grundspracheprogramme auf Lochstreifen vorliegen und um den Überwacher nicht zu umfangreich werden zu lassen (Speicherplatzbedarf).
Der Lader muß zum Arbeitsspeicher mit absoluten Adressen zugreifen können. Deshalb muß er im Zustand PZ des Prozessors ablaufen. Dementsprechend wurde er als Systemtask realisiert. Dies bietet auch die Möglichkeit durch den Lader nötigenfalls alle Überwacherfunktionen des Überwachers aufzurufen.

3.11.2 Ladefunktion

Der Lader wird von einer Task aus durch den Überwacheraufruf "ANMELDEN mit Laden" aufgerufen (vgl. Abschnitt 3.3.3.2). Die Abläufe im Überwacher bei der Bearbeitung dieses Überwacheraufrufes sind im Flußdiagramm (Bild 3.43) dargestellt.

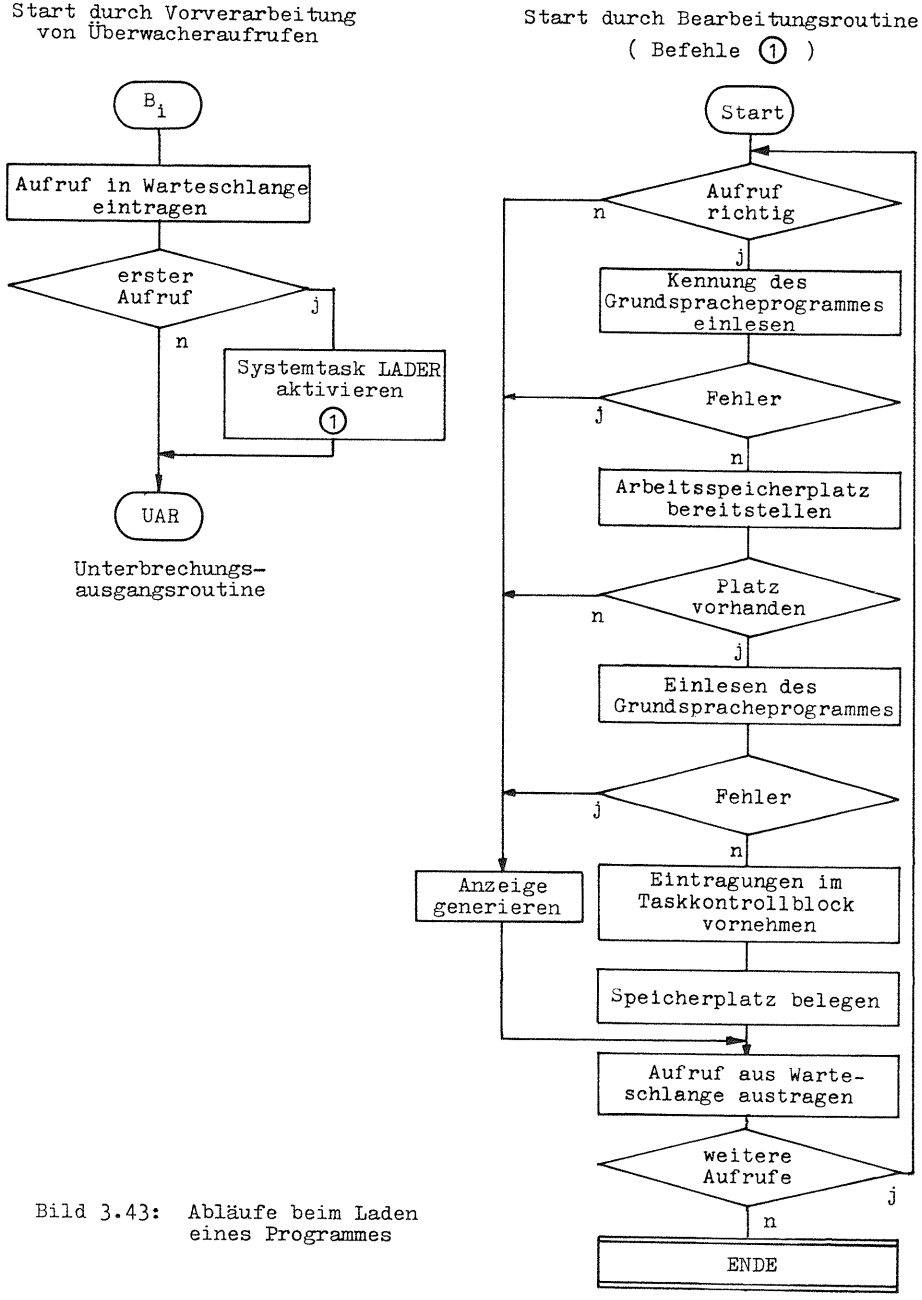


Bild 3.43: Abläufe beim Laden eines Programmes

Durch die Bearbeitungsroutine für den Überwacheraufruf "ANMELDEN mit Laden" wird der Aufruf in die Warteschlange für diesen Aufruf eingetragen (linker Teil des Flußdiagrammes). Falls es sich um den ersten Eintrag in der Warteschlange handelt wird die Systemtask "Lader" aktiviert. Dieses Aktivieren (Befehlsfolge 1) geschieht durch Setzen ihres Bits in der Taskwarteschlange. Damit wird bei der Prozessorvergabe durch die UAR der Systemtask "Lader" der Prozessor zugeteilt. (Der Lader hat die Tasknummer 0.)

In der Systemtask "Lader" (rechter Teil das Flußdiagrammes) wird zuerst geprüft, ob der Überwacheraufruf formal richtig ist. Enthält er Fehler, so wird eine Anzeige generiert, die in die Anzeigenzelle des Aufrufes (vgl. Abschnitt 3.2.2) eingetragen wird, und der Aufruf wird aus der Warteschlange ausgetragen. Wenn noch weitere Aufrufe in der Warteschlange eingetragen sind, kommt der nächste Aufruf zur Ausführung, ansonsten beendet sich die Systemtask durch den Überwacheraufruf ENDE (vgl. Abschnitt 3.3.3.3).

Bei einem formal richtigen Aufruf wird die Kennung des im Aufruf angegebenen Grundspracheprogrammes eingelesen. Sie enthält die Angaben über die Länge des Programmes. Danach wird der notwendige Speicherplatz für das Programm bereitgestellt. (Die Arbeitsspeicherverwaltung wird im nächsten Abschnitt 3.11.3 behandelt.)

Nach erfolgter Speicherplatzzuteilung wird das Programm in den Arbeitsspeicher eingelesen. Danach werden die Eintragungen für die Basisadressenregister im Taskkontrollblock, welche die Lage des Programmes im Arbeitsspeicher angeben, vorgenommen und die Anzeige für den Aufruf gelöscht. (Der Aufruf war erfolgreich.) Zum Schluß wird der Speicherplatz in den Listen für die Arbeitsspeicherverwaltung belegt.

Wenn an irgendeiner Stelle des Ablaufes ein Fehler aufgetreten ist, wird eine Anzeige generiert und wie bei falschen Aufrufen verfahren. Diese Anzeige wird von der aufrufenden Task gelesen und auf eine eventuelle Fehleranzeige hin ausgewertet.

3.11.3 Arbeitsspeicherverwaltung

Im Überwacher können maximal 14 Tasks verwaltet werden (vgl. Abschnitt 3.3.3). Dazu zählen auch die beiden Systemtasks "Lader" und "Bedienteil", die arbeitsspeicherresident sind. Es verbleiben also insgesamt 12 Tasks, für die Speicherplatz zur Verfügung gestellt werden muß.

Im Bild 3.44 ist der ungünsigste (allgemeine) Fall für eine Speicherplatzbelegung angegeben:

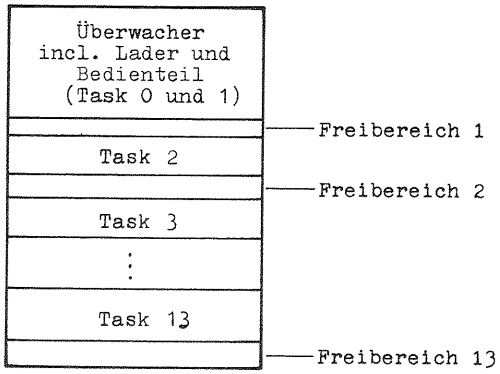


Bild 3.44: Beispiel für eine Arbeitsspeicherbelegung

Der Überwacher mit seinen beiden Systemtasks "Lader" und "Bedienteil" steht vorne im Arbeitsspeicher und die "Anwendertasks" (alle Tasks, die im Zustand NZ des Prozessors zur Ausführung kommen) sind über den gesamten Arbeitsspeicher verteilt. Durch wiederholte Belegungen und Freigaben von Speicherbereichen für Programme können Lücken (Freibereiche) in der Speicherbelegung entstehen. Es sind bei 12 Anwendertasks insgesamt 13 Freibereiche möglich.

Um nun den Speicherplatz verwalten zu können, wird eine Liste der Freibereiche durch den Überwacher geführt. Diese Liste gehört zum Arbeitsspeicherbereich des Überwachers. Dadurch wird verhindert, daß diese Liste durch eine Anwandertask zerstört werden kann. Die Liste der Freibereiche hat folgendes Format:

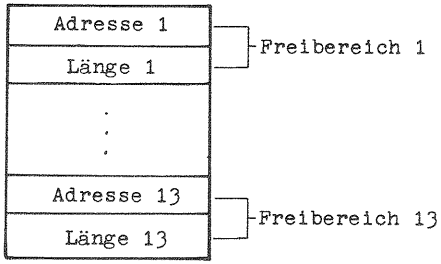


Bild 3.45: Liste der maximal 13 Freibereiche im Arbeitsspeicher

Für jeden Freibereich enthält die Liste (Bild 3.45) zwei Eintragungen. Ein Eintrag enthält die Adresse der ersten Arbeitsspeicherzelle des Freibereiches und der andere Eintrag enthält die Länge dieses Freibereiches.

Beim Suchen eines freien Arbeitsspeicherbereiches für das Programm einer Task werden die Listeneinträge der Reihe nach abgesucht, und es wird der kleinste Freibereich, in dem das Programm Platz hat, belegt. Anschließend wird der betreffende Freibereich der neuen Belegung angepaßt. Bei einer Arbeitsspeicherplatzfreigabe (ausgelöst von einer Task durch den Überwacheraufruf "ABMELDEN", vgl. Abschnitt 3.3.3.3) werden aufeinanderfolgende Freibereiche zu einem Freibereich zusammengefaßt. Die Länge des von einer Task belegten Arbeitsspeicherbereiches wird in einer Liste geführt, die für jede Task (insgesamt14) einen Listenplatz enthält.

Im Normalfall werden die Freibereiche von vorne (ab Freibereich Nr.1) abgesucht und belegt. Durch einen Parameter im Aufruf ist es jedoch möglich, auch Speicherplatzz gezielt hinten im Arbeitsspeicher zu belegen. Dadurch wird verhindert, daß durch ein nichtresidentes Programm, das nur für Testzwecke oder zum Laden der gesamten residenten Vermittlungssoftware benötigt wird, nach dessen Abmeldung vorne im Arbeitsspeicher ein Freibereich entsteht.

Die Möglichkeit, immer alle Freibereiche durch Verschieben der Programme im Arbeitsspeicher zu einem Freibereich zusammenzufassen, scheitert daran, daß bei Ein-Ausgabeoperationen den Geräten die betreffenden Adressen beim Anstoßen der Operation übergeben werden und solange die Operation noch andauert nicht geändert werden können. Beim Verschieben eines Programmes, für das in diesem Augenblick (im cycle-stealing) eine Ein-Ausgabeoperation läuft, würden dessen Adressen bei der Verschiebeoperation geändert und dadurch zu Fehlern führen. Das Zusammenschieben der Freibereiche wäre jedoch möglich, wenn gerade keine Ein-Ausgabeoperation läuft. Aus Aufwandsgründen wurde von dieser Realisierungsmöglichkeit abgesehen.

3.12 Systemtask "Bedienteil"

Die Bedienung des Rechners für vermittlungstechnische Aufgaben erfolgt über das ORG4 und die dazugehörenden Dienstprogramme (vgl. Abschnitt 2.1.3). Während der Testphase für die Systemsoftware ist es jedoch wünscheswert, vom Bedienungspersonal aus Anforderungen direkt an den Überwacher zu geben (z.B. zum Laden eines Programmes in den Arbeitsspeicher, oder zur Ausgabe des Zustandes einer Task

auf dem Bedinungsblattschreiber). Zu diesem Zweck wurde die Systemtask "Bedienteil" eingeführt. Sie ist als Systemtask realisiert, da sie Zugriff zu den Listen des Überwachers haben muß. Sie hat die Tasknummer 1.

Der Bedienteil wird von der Ureingaberoutine (durch Setzen des zur Tasknummer 1 gehörenden Bits in der Taskwarteschlange) aktiviert. Nach der Prozessorzuteilung durchläuft er einen Überwacheraufruf für die "Eingabe Bedienungsblattschreiber" und einen weiteren Aufruf für das Warten dieser Task bis zum Abschluß der Eingabe. Damit ist der Bedienteil solange im Zustand "blockiert" bis die Eingabe über den Bedienungsblattschreiber erfolgt ist.

Nach erfolgter Blattschreibereingabe wird der Bedienteil (durch die Bearbeitungsroutine für "Ein-Ausgabe Standardperipherie", vgl. Abschnitt 3.4) wieder aktiviert.

Einige der möglichen Aufgaben, die mit Hilfe der Systemtask "Bedienteil" durchführbar sind, werden kurz angegeben:

- Aufruf der Funktionen der Ablauforganisation (vgl. Abschnitt 3.3) (z.B. zum Laden und Ausführen eines Programmes als Task)
- Ausgabe von Taskzuständen über den Bedienungsblattschreiber
- Ausgabe der Freibereichsliste der Arbeitsspeicherverwaltung über den Bedienungsblattschreiber
- Setzen der Freibereiche des Arbeitsspeichers in einen definierten Zustand (Einschreiben von vorgegebenen Bitmustern)

4. ORGANISATIONS- UND ARBEITSPROGRAMME

4.1 Allgemeines

Zur Durchführung aller Vermittlungsaufgaben, die im Steuerrechner ausgeführt werden müssen, sind die ARBEITSPROGRAMME (AP) zuständig. Der Ablauf der einzelnen AP wird durch die Organisationsprogramme ORG1 bis ORG7 gesteuert und überwacht. Die AP sind jeweils entsprechend ihrem Aufgabentyp einem bestimmten Organisationsprogramm (ORG) zugeteilt. Sie können nur unter diesem ORG zur Ausführung kommen. (Die Aufgaben der einzelnen AP wurden bereits im Kapitel 2 behandelt.) Die Organisations- und Arbeitsprogramme fordern zur Ausführung bestimmter Aufgaben die Funktionen des Überwachers durch Überwacheraufrufe (vgl. Abschnitt 3.2) an.

Der Ablauf der Organisations- und Arbeitsprogramme wird durch den Überwacher gesteuert; dabei werden diese Programme im Überwacher als Tasks verwaltet. Die Zuordnung der verschiedenen Programme zu einzelnen Tasks und die sich damit ergebenden Prioritäten für die Tasks werden im folgenden Abschnitt behandelt.

4.2 Tasknummernzuteilung und Prioritäten

Die verschiedenen Organisationsprogramme (eischließlich ihrer zugehörigen Arbeitsprogramme) haben unterschiedliche Prioritäten (vgl. Kapitel 2); dabei gibt die Nummer des ORG direkt dessen Priorität an. (Das ORG1 hat die höchste Priorität und das ORG7 die niederste Priorität.)

Jedes Organisationsprogramm muß (für den Überwacher) noch einer eigenen Tasknummer zugeordnet werden.Damit kommen die ORG mit unterschiedlicher Priorität als Tasks zur Ausführung.

Für die Zuteilung von Tasknummern an die Arbeitsprogramme gibt es zwei Lösungen:

I) Alle Arbeitsprogramme, die von demselben ORG verwaltet werden, erhalten im Falle ihrer Aktivierung eine Tasknummer, die um 1 größer ist als die Tasknummer des betreffenden ORG. (ORG als Task i, AP als Task i+1)

II) Alle Arbeitsprogramme, die von demselben ORG verwaltet werden, kommen unter derselben Tasknummer wie das betreffende ORG zur Ausführung.

Entsprechend der Tasknummernzuteilung gemäß I bzw. II müssen die Arbeitsprogramme unterschiedlich aktiviert werden. Die dazu notwendigen Abläufe sind in folgendem Bild 4.1 dargestellt.

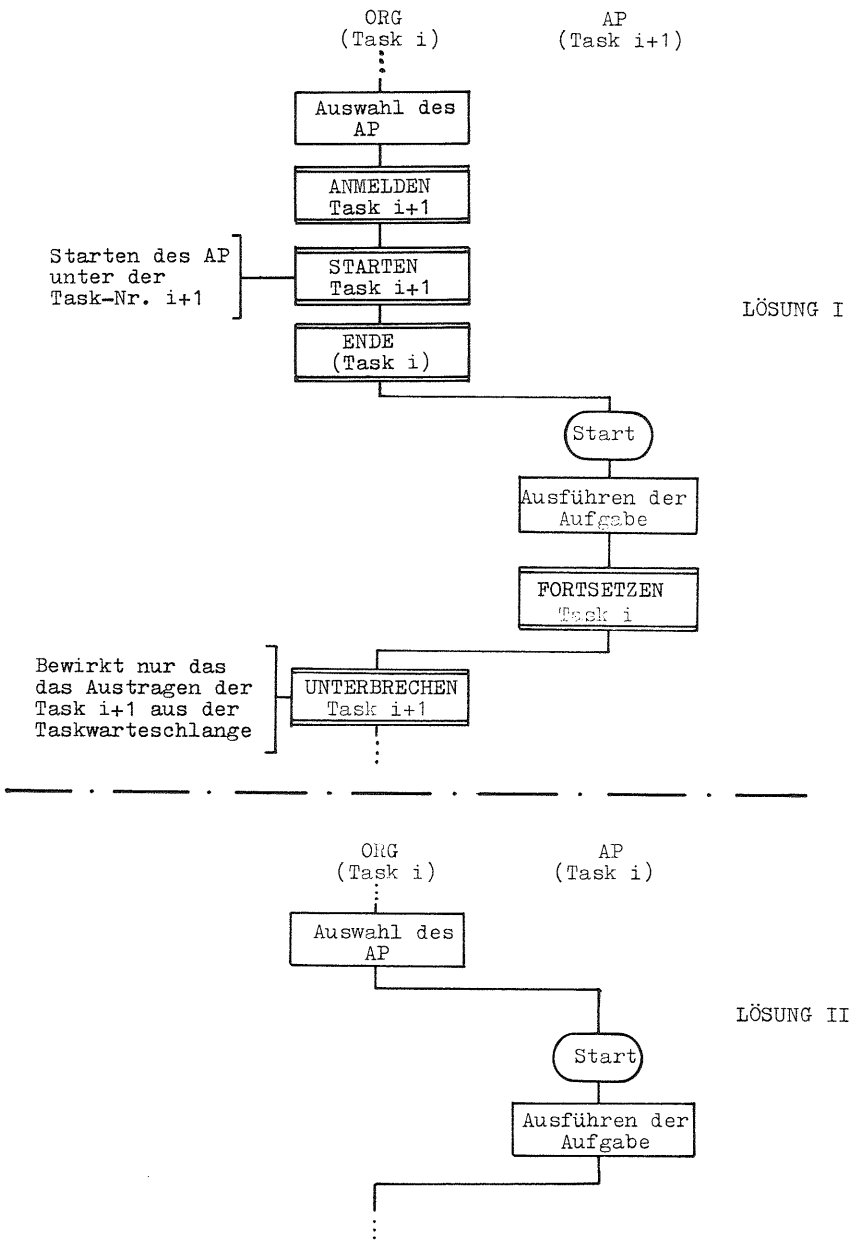


Bild 4.1: Aktivierung eines Arbeitsprogrammes durch ein Organisationsprogramm

Bei der Lösung I (Bild 4.1) muß das Arbeitsprogramm vom Organisationsprogramm mit den beiden Überwacheraufrufen ANMELDEN und STARTEN aktiviert werden. Das ORG kann sich danach selbst beenden (Aufruf ENDE), da unter einem ORG gleichzeitig nur ein AP ablaufen darf. Wenn das AP seine Aufgaben erledigt hat, aktiviert es wieder sein ORG mit dem Überwacheraufruf FORTSETZEN, das dann seinerseits das AP mit dem Überwacheraufruf UNTERBRECHEN formal beendet.

Für die Ausführung eines AP sind bei der Lösung I demnach 5 Überwacheraufrufe notwendig, die natürlich auch eine gewisse Bearbeitungszeit durch den Überwacher beanspruchen (Overhead).

Die Lösung II vermeidet diese Überwacheraufrufe und damit auch den dafür notwendigen Overhead. Das AP wird direkt von seinem ORG angesprungen. Wenn es seine Aufgaben erledigt hat, erfolgt vom AP wieder direkt der Rücksprung zum ORG.

Für die Organisationsprogramme ORG1 bis ORG6 wurde die Lösung I realisiert und für das Organisationsprogramm ORG7 eine Mischform aus den beiden Lösungen I und II.

Die Mischform für das ORG7 ergibt sich aus der Notwendigkeit, während des Ladens eines Arbeitsprogrammes vom Plattenspeicher in den Arbeitsspeicher, andere Arbeitsprogramme, die resident im Arbeitsspeicher sind, zur Ausführung zu bringen. Diese Mischform wird noch im Abschnitt 4.5.4 ausführlich behandelt. (Für die Organisationsprogramme ORG1 bis ORG6 konnte die Lösung I realisiert werden, da deren Arbeitsprogramme alle resident im Arbeitsspeicher sind.)

Die sich damit ergebende Tasknummernzuteilung ist im Bild 4.2 angegeben. Die Tasknummern beginnen mit der Nummer 2, da die Nummern 0 und 1 bereits für die Systemtasks LADER und BEDIENTEIL vergeben sind (vgl. Abschnitte 3.11 und 3.12).

Zwischen den Organisationsprogrammen ist ein Austausch von Informationen zum Aktivieren ihrer Arbeitsprogramme notwendig. (Z.B. muß, wenn in einem Routineprüfprogramm des ORG7 ein Fehler erkannt wurde, ein Fehlerbehandlungsprogramm unter dem ORG1 aktiviert werden.) Die Durchführung des Informationsaustausches zwischen Organisationsprogrammen wird in folgendem Abschnitt behandelt.

Programm	Task-nummer (Priorität)	resident im Arbeitsspeicher
Organisationsprogramm ORG1	2	ja
Fehlerbehandlunsprogramme (FP)	2	ja
Organisationsprogramm ORG3	3	ja
Sonderdienstprogramme (SDP)	3	ja
Organisationsprogramm ORG4	4	ja
Dienstprogramme (I) (DP)	4	ja
Organisationsprogramm ORG6	5	ja
Zeitprogramme (ZP)	5	ja
Organisationsprogramm ORG7	6	ja
Vermittlungsprogramme (VP)	6	ja
Dienstprogramme (II) (DP)	6	ja
Dienstprogramme (III) (DP)	7	nein
Routineprüfprogramme (RP)	7	nein

Bild 4.2: Zuteilung von Tasknummern zu Programmen

4.3 Informationsaustausch zwischen Organisationsprogrammen

Zum Informationsaustausch zwischen Organisationsprogrammen werden die Überwacheraufrufe SENDE NACHRICHT und ERWARTE NACHRICHT benutzt (vgl. Abschnitt 3.3.5). Ein Organisationsprogramm, das alle Aufgaben erledigt hat, wartet mit dem Überwacheraufruf ERWARTE NACHRICHT auf eine Nachricht, die ihm angibt, welche neuen Aufgaben durchzuführen sind. Wenn ein ORG die Durchfürung einer Aufgabe durch ein anderes ORG benötigt, so wird dies durch den Überwacheraufruf SENDE NACHRICHT dem benötigten ORG über den Überwacher mitgeteilt.

Eine Nachricht hat folgendes Format:

Codewort 1	Codewort 2	Codewort 3	Parameter	Endezeichen

Bild 4.3: Struktur einer Nachricht für den Informationsaustausch zwischen Organisationsprogrammen.

Das Codewort 1 (CW1) der Nachricht nach Bild 4.3 besteht aus 4 Zeichen und ist für den Überwacher bestimmt. Er ermittelt mit Hilfe dieses Codewortes jenes Organisationsprogramm, für das die Nachricht bestimmt ist. Die 4 Zeichen des CW1 sind so gewählt, daß sie der Abkürzung des ORG entsprechen, für das die Nachricht bestimmt ist. (Z.B. wenn die Nachricht für das ORG1 bestimmt ist, so besteht das CW1 genau aus den 4 Zeichen "ORG1". Natürlich muß das benötigte ORG, das diese Nachricht empfangen soll, auch dieses Codewort in seiner Liste für den Überwacheraufruf ERWARTE NACHRICHT enthalten.)

Die Restinformation der Nachricht nach Bild 4.3 ist für jenes benötigte ORG bestimmt, das die Nachricht (zeichenweise auf dem Weg über den Überwacher mit Hilfe des Überwacheraufrufes NACHRICHTENZEICHEN) empfängt. Dabei gibt das Codewort 2 an, von welchem ORG die Nachricht kommt, und das Codewort 3 gibt an, welches Arbeitsprogramm des betreffenden ORG zu aktivieren ist. Die Parameter sind dann für dieses Arbeitsprogramm bestimmt. Die gesamte Nachricht wird durch ein Endezeichen abgeschlossen. Dadurch sind Nachrichten unterschiedlicher Länge (unterschiedliche Zahl von Parametern) möglich.

Die Organisationsprogramme ORG1, ORG6 und ORG7 können nicht nur von anderen ORG's aus sondern auch durch den Überwacher im Rahmen der Taktbearbeitungsroutine (TBR, vgl. Abschnitt 3.5.2.1) über den Überwacheraufruf SENDE NACHRICHT aktiviert werden:

Das ORG1 bzw. das ORG7 wird aktiviert, wenn eine Fehlermeldung bzw. eine reguläre Meldung von einer dezentralen Steuereinheit eintrifft und die betreffende Eingabewarteschlange zu diesem Zeitpunkt leer war.

Das ORG6 (Zeitprogramme) wird aktiviert, wenn der Zähler für dessen T_o-Takt den Wert Null erreicht.

Bei diesen Nachrichten, die von der Taktbearbeitungsroutine abgesetzt werden, besteht die Nachricht nur aus dem Codewort 1 und dem Endezeichen. Die Information selbst, die zu verarbeiten ist, steht für das ORG1 bzw. das ORG7 in den beiden Eingabewarteschlangen. Für das ORG6 wird außer der impliziten Nachricht "Zeitpunkt T_o" keine weitere Information benötigt.

Während der Ausführung eines Arbeitsprogrammes unter seinem ORG kann es u.a. erforderlich sein, mehrere Nachrichten an ein anderes ORG oder sogar an mehrere unterschiedliche ORG's abzusenden.

Eine durch den Überwacheraufruf SENDE NACHRICHT abgeschickte Nachricht wird bis zur vollständigen Übernahme durch die empfangende Task in der Nachrichtenwarteschlange des Überwachers geführt. Solange die Nachricht in der Warteschlange steht, ist auch das Bit 24 der Anzeigenzelle dieses Aufrufes gesetzt. (Die Nachrichtenwarteschlange ist genauso organisiert wie die Warteschlangen für die Ein- Ausgabegeräte, vgl. Abschnitt 3.4.4.) Während dieser Zeit darf dieser Aufruf nicht erneut durchlaufen werden. Dadurch kann durch den Überwacheraufruf SENDE NACHRICHT nur eine Nachricht gesendet werden. Weitere Nachrichten müssen beim sendenden ORG so lange gespeichert werden bis die vorangehende (abgeschickte) Nachricht vom empfangenden ORG angenommen wurde. Es müssen allerdings nur solche Nachrichten vom sendenden ORG gespeichert werden, die für ein ORG niederer Priorität (größere Nummer) bestimmt sind. Eine Nachricht an ein ORG höherer Priorität (kleinere Nummer) wird von diesem sofort angenommen und bearbeitet, wobei das ORG niederer Priorität (sendendes ORG) dann warten muß, bis die Aufgaben in der höheren Priorität erledigt sind.

Damit nun die Nachrichten entsprechend der Priorität des ORG, für das sie bestimmt sind, abgeschickt werden und nicht nach der Reihenfolge ihrer Generierung, ist es notwendig, so viele unterschiedliche Überwacheraufrufe SENDE NACHRICHT im betreffenden sendenden ORG zu haben, wie ORG's niederer Priorität vorhanden sind. Es ist nämlich nicht möglich, eine Nachricht, die in der Nachrichtenwarteschlange des Überwachers steht, durch die sendende Task (ORG) wieder aus der Warteschlange auszutragen. Bei nur einem Überwacheraufruf SENDE NACHRICHT müßte z.B. im ORG1 eine Nachricht an das ORG4 warten, wenn zuvor vom ORG1 an das ORG7 eine Nachricht gesendet wurde und diese vom ORG7 noch nicht vollständig übernommen worden ist.

Der Nachrichtenaustausch zwischen ORG's kann auf zwei Arten erfolgen, die im Bild 4.4 dargestellt sind.

Den beiden Abläufen liegt die Annahme zugrunde, daß die Programmausführung unter dem ORGk unterbrochen wurde und das ORGi (i<k) aktiviert wurde. Z.B. wurde ein Routineprüfprogramm unter dem ORG7 durch eine eintreffende Fehlermeldung von der Peripherie unterbrochen und das ORG1 durch die Taktbearbeitungsroutine aktiviert. Das ORG1 hat nun selbst eine Nachricht an das ORG7, z.B. um ein Routineprüfprogramm zu aktivieren.

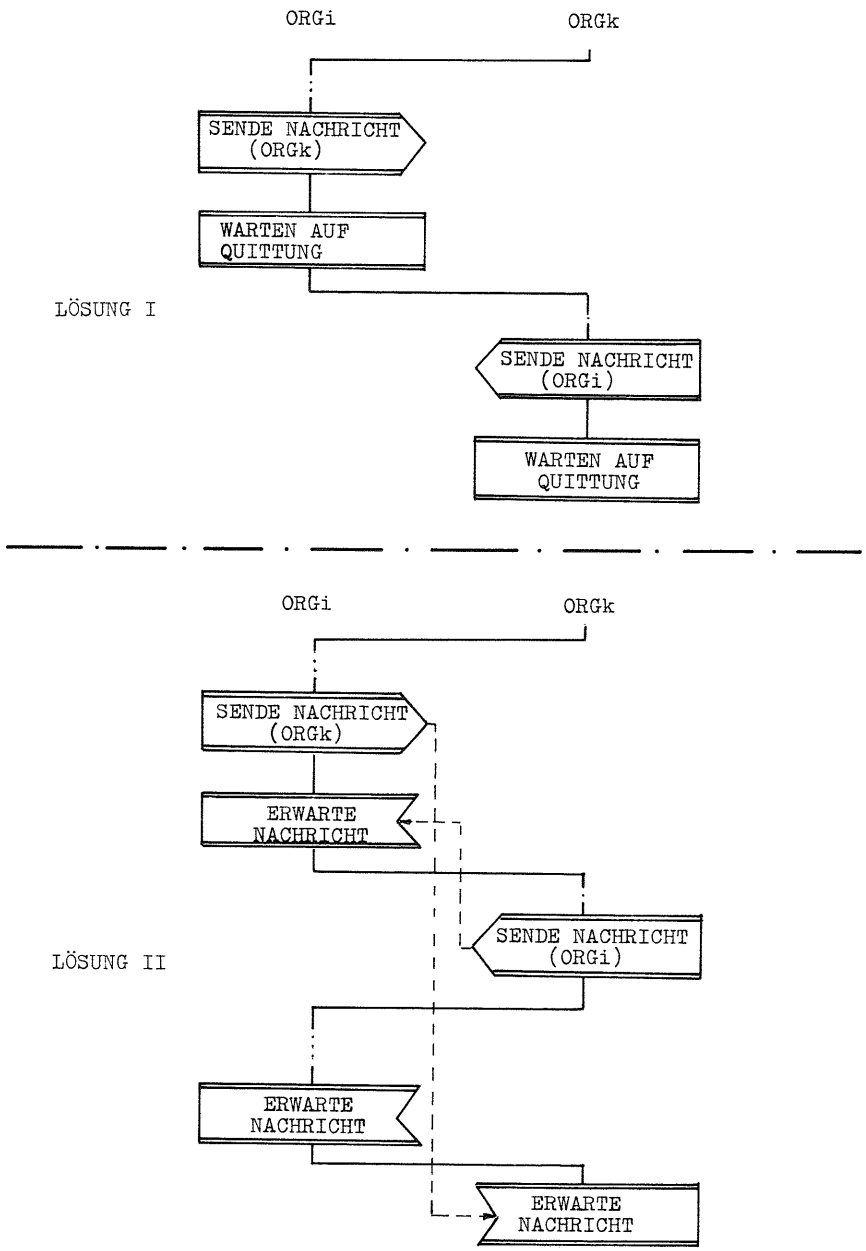


Bild 4.4: Prinzip des Nachrichtenaustausches zwischen Organisationsprogrammen (ORGi hat eine höhere Priorität als ORGk)

Nachdem nun die Aufgaben in der höheren Priorität (ORGi) erledigt sind, wird das unterbrochene Programm niederer Priorität fortgesetzt. Dieses habe nun seinerseits eine Nachricht an das ORGi.Z.B. könnte durch ein Routineprüfprogramm ein Fehler entdeckt werden, und es soll ein Fehlerbehandlungsprogramm aktiviert werden.

Bei der Lösung I (Bild 4.4) wird, nachdem die Nachricht durch den Überwacheraufruf SENDE NACHRICHT dem Überwacher übergeben ist, mit dem Aufruf WARTEN AUF QUITTUNG darauf gewartet, daß die Nachricht von dem empfangenden ORG übernommen wurde. Nach Bild 4.4 kann aber die Nachricht des ORGi vom ORGk nicht übernommen werden, da es selbst eine Nachricht abgeschickt hat und auf eine Quittung wartet. Es entsteht ein sogenannter "Deadlock" /25/ , d.h. die beiden Tasks (Programme) warten gegenseitig aufeinander und können dadurch nicht weiterlaufen.

Um einen Deadlock zu vermeiden muß deshalb die Lösung II (Bild 4.4) realisiert werden. Bei dieser Lösung wird nach dem Senden einer Nachricht nicht auf die Übernahme der Nachricht durch das empfangende ORG gewartet. Statt dessen ist dieses ORG empfangsbereit für eine eventuell ankommende Nachricht. (Überwacheraufruf ERWARTE NACHRICHT) Dadurch können beide Programme weiterlaufen. Allerdings muß die Tatsache, daß eine Nachricht vom empfangenden ORG übernommen wurde, von diesem dem sendenden ORG mit einer Nachricht mitgeteilt werden.

Die Abläufe für das Senden und Empfangen von Nachrichten in einem Organisationsprogramm gemäß Lösung II sind im folgenden Bild 4.5 ausführlich dargestellt.

Ein Organisationsprogramm "wartet" grundsätzlich, nachdem es alle Aufgaben erledigt hat, mit dem Überwacheraufruf ERWARTE NACHRICHT auf das Eintreffen einer Nachricht.(Mit anderen Worten , es ist für Nachrichten empfangsbereit.)

Wenn im Überwacher eine Nachricht für dieses ORG eintrifft, so wird es durch den Überwacher aktiviert, d.h. in der Taskwarteschlange wird "sein" Bit gesetzt.

Bei der Nachricht kann es sich handeln um:

a) eine "echte" Nachricht (keine Ausführungsquittung)

b) eine Ausführungsquittung

Im Fall a) wird die Nachricht übernommen und das entsprechende Arbeitsprogramm (oder falls notwendig mehrere Arbeitsprogramme) ausgeführt. Danach wird erneut auf eine Nachricht "gewartet".

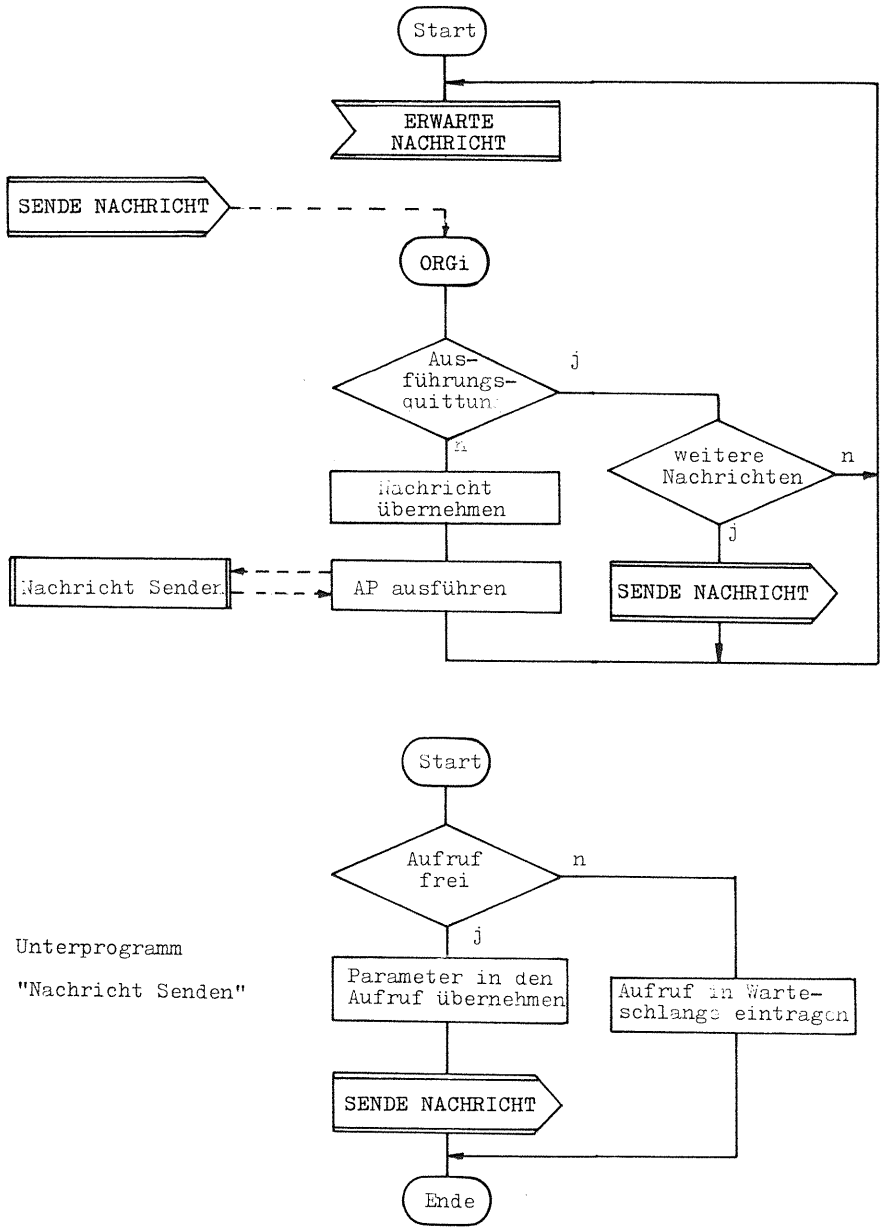


Bild 4.5: Abläufe beim Senden und Empfangen von Nachrichten

Im Fall b) wird geprüft, ob in der ORG-eigenen Warteschlange noch Nachrichten an andere ORG's zwischengespeichert sind. Wenn dies der Fall ist, wird eine Nachricht aus der Warteschlange mit dem Überwacheraufruf SENDE NACHRICHT abgeschickt. (Beschreibung des Falles b) siehe unten.)

Während der Ausführung eines Arbeitsprogrammes kann es erforderlich werden, eine Nachricht an ein anderes ORG abzusenden. Zu diesem Zweck wird ein Unterprogramm des ORG zum Senden dieser Nachricht aufgerufen. Im Unterprogramm (unterer Teil des Bildes 4.5) wird zuerst geprüft, ob der Überwacheraufruf zum Abschicken einer Nachricht noch (für eine andere Nachricht) in der Nachrichtenwarteschlange des Überwachers eingetragen ist. (Bit 24 der Anzeigenzelle des Überwacheraufrufes ist gesetzt.) Wenn dies der Fall ist, muß die neue Nachricht in eine Warteschlange des sendenden ORG eingetragen werden. Diese Warteschlange ist als blockweise organisierte Liste realisiert (vgl. Anhang A1.2). Falls der Überwacheraufruf erlaubt ist, kann die Nachricht sofort abgeschickt werden.

Damit solche Nachrichten, die in der Warteschlange des ORG eingetragen werden, später ebenfalls abgesendet werden können, wird jede in einem (anderen) empfangenden ORG verarbeitete Nachricht mittels der Nachricht "Ausführungsquittung" quittiert. Trifft eine derartige Nachricht ein (Fall b),so wird nur eine eventuell wartende Nachricht aus der Warteschlange abgeschickt (Bild 4.5 oben rechts) und dann wieder in den Wartezustand des sendenden ORG verzweigt.

Die beschriebene Art des Informationsaustausches zwischen Organisationsprogrammen erlaubt es, die Organisationsprogramme und die dazugehörenden Arbeitsprogramme getrennt voneinander zu programmieren. Eine Kopplung zwischen den Organisationsprogrammen besteht nur durch das Nachrichtenformat.

4.4 Steuerungsablauf in einem Organisationsprogramm

Die Organisationsprogramme erhalten ihre Aufgaben durch "Nachrichten" mitgeteilt. Diese kommen entweder von anderen Organisationsprogrammen oder vom Überwacher selbst (Taktbearbeitungsroutine, vgl. Abschnitt 3.5.2.1). Aus der Nachricht bestimmen die ORG's dann das betreffende Arbeitsprogramm und bringen es zur Ausführung. Der Steuerfluß in einem ORG ist im Bild 4.6 dargestellt.

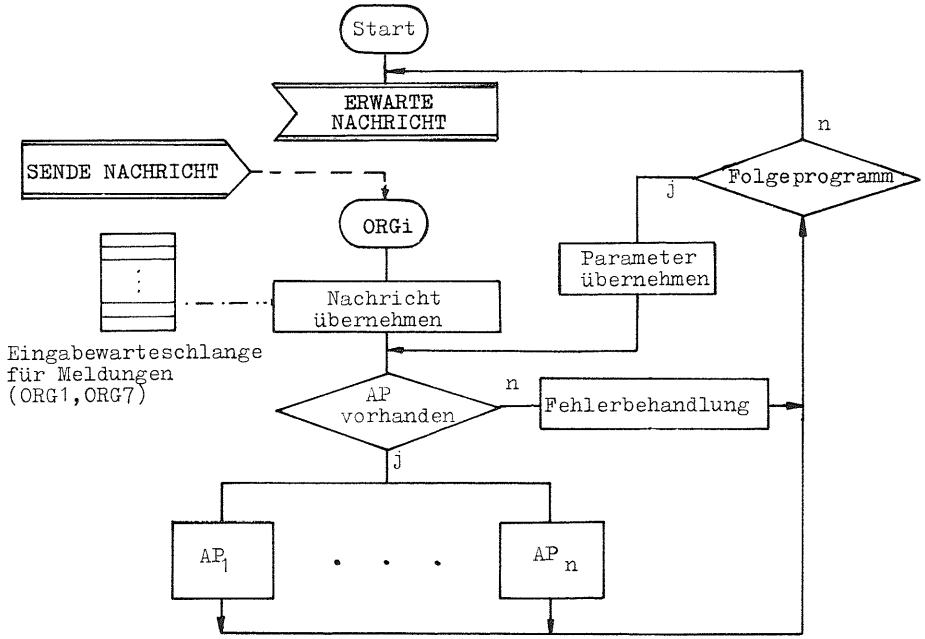


Bild 4.6: Steuerfluß in einem Organisationsprogramm

Wie bereits im Abschnitt 4.3 beschrieben wurde, wartet ein ORG auf das Eintreffen einer Nachricht. In der Nachricht ist angegeben, welches Arbeitsprogramm ausgeführt werden soll und zusätzlich sind noch die Parameter angegeben, welche das Arbeitsprogramm benötigt. Wenn die Nachricht vom Überwacher selbst kommt (Taktbearbeitungsroutine), so steht die Information in der Eingabewarteschlange für Meldungen von der Peripherie (für ORG1 und ORG7) oder es wird keine Information benötigt (ORG6).

Nach Übernahme der Nachricht durch den Überwacheraufruf NACHRICHTENZEICHEN, oder durch die Überwacheraufrufe für die Eingabe einer Meldung aus einer Eingabewarteschlange (WSER, WSEP, vgl. Abschnitt 3.5.2.3) kann das betreffende Arbeitsprogramm ausgewählt werden. Zu diesem Zweck ist jedem Arbeitsprogramm eine Nummer zugeordnet, die dann entweder in einer Meldung von der Peripherie oder in einer Nachricht (Codewort 3, vgl. Abschnitt 4.3) enthalten sein muß.

Die Aktivierung des Arbeitsprogrammes erfolgt über eine Sprungtabelle (Bild 4.7).(Es wird an dieser Stelle nur die Aktivierung von Arbeitsprogrammen behandelt, die resident im Arbeitsspeicher sind. Die Aktivierung nichtresidenter Arbeitsprogramme wird im Abschnitt 4.5.4 behandelt.)

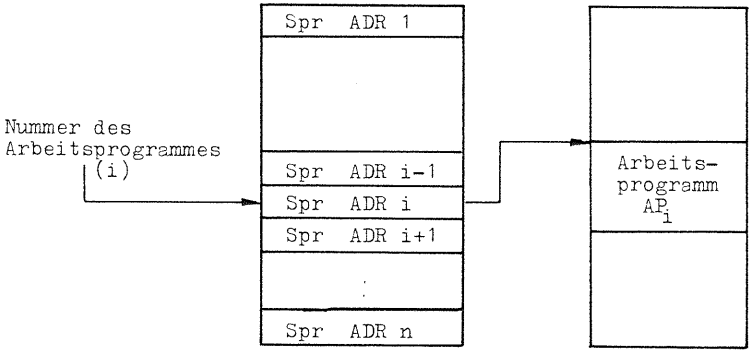


Bild 4.7: Bestimmung eines Arbeitsprogrammes

Die Sprungtabelle (Bild 4.7) enthält die Startadressen der Arbeitsprogramme. Der Zugang zur Tabelle erfolgt mit der Nummer des Programmes, das aktiviert werden soll.

Wenn die Nummer des Arbeitsprogrammes nicht existiert, wird zu einer Fehlerbehandlung verzweigt. In dieser Fehlerbehandlung wird eine Nachricht an das ORG1 gesendet, daß ein nicht vorhandenes Arbeitsprogramm aktiviert werden sollte. Tritt eine falsche AP-Nummer im ORG1 selbst auf, so wird eine Ausgabe über den Bedienungsblattschreiber an das Vermittlungspersonal durchgeführt.

Nach Durchlaufen jedes Arbeitsprogrammes wird geprüft, ob die Ausführung eines weiteren Arbeitsprogrammes notwendig ist (Bild 4.6). Wenn keine AP mehr auszuführen sind, wird wieder auf das Eintreffen einer Nachricht gewartet. Falls ein weiteres AP zu aktivieren ist, werden die Parameter dafür übernommen und das neue AP wird wieder über die Sprungtabelle aktiviert.

Als Kriterium für die Ausführung weiterer Arbeitsprogramme dienen Einträge in Warteschlangen, die im folgenden behandelt werden.

Nach der Beschreibung des Steuerungsablaufes in einem Organisationsprogramm wird nun der Datenfluß beschrieben, der angibt, wie die Parameter vom ORG an ein AP übergeben werden bzw. durch das ORG von einem AP übernommen werden(Bild 4.8).

Bei der Übernahme einer Nachricht durch ein ORG aus der Nachrichtenwarteschlange des Überwachers, oder einer Meldung aus einer Eingabewarteschlange werden die Parameter, welche das AP dieses ORG benötigt, in einen Speicherbereich PAREIN (Parametereingabe) eingetragen. Die Parameter werden dabei eventuell umcodiert und in eine Form gebracht, die den AP einen einfachen Zugriff ermöglichen (z.B. wird pro Speicherplatz des PAREIN nur ein Parameter abgelegt).

Wenn von einem AP ein Befehl an die Vermittlungsperipherie oder eine Nachricht an ein anderes ORG abgegeben werden soll, so werden dafür die Parameter in einem Speicherbereich PARAUS (Parameterausgabe) abgelegt. Das ORG entnimmt die Parameter aus diesem Speicherbereich und stellt damit eine Nachricht an ein anderes ORG oder einen Befehl an die Vermittlungsperipherie zusammen.

Um ein ORG von einem AP aus zur Durchführung eines weiteren AP zu veranlassen, werden die dafür notwendigen Parameter ebenfalls in PARAUS eingetragen. Das ORG übernimmt die Parameter bis zur Aktivierung dieses weiteren AP in eine Warteschlange (sog. Internwarteschlange, die als blockweise organisierte zusammenhängende Liste realisiert ist, vgl. Anhang A1.2). Durch die Internwarteschlange ist es möglich von einem AP aus auch nacheinander mehrere AP's zur Ausführung zu bringen.

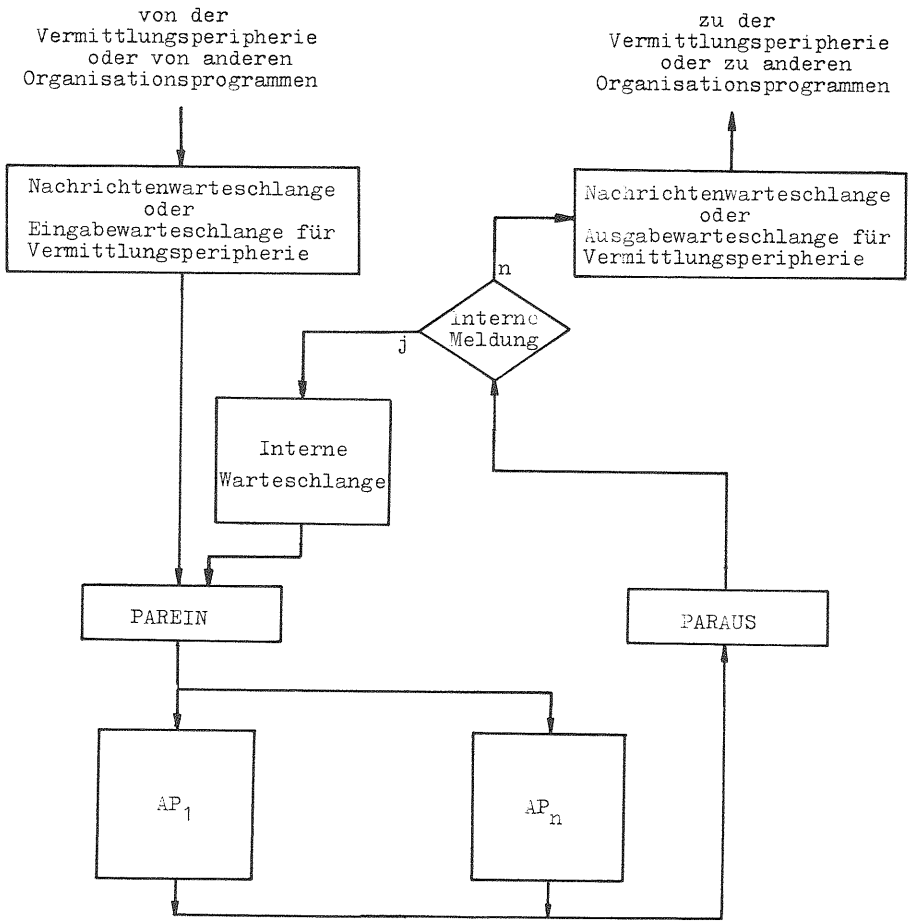


Bild 4.8: Datenfluß in einem Organisationsprogramm

Die Unterscheidung, ob es sich bei den Einträgen in PARAUS um Informationen handelt, die nach außen (Vermittlungsperipherie) gegeben werden oder die intern verarbeitet werden müssen, wird anhand eines Codes vorgenommen, der in der ersten Zelle von PARAUS steht. (Wenn das Bit 1 gesetzt ist, handelt es sich um Informationen, die intern verarbeitet werden müssen.)

Wenn ein AP die Kontrolle an sein ORG zurückgibt (durch Rücksprung in sein ORG), prüft das ORG, ob noch Einträge in der Internwarteschlange vorliegen (siehe auch Bild 4.6).Wenn dies der Fall ist, wird das nächste AP aktiviert, wobei die Parameter wieder in PAREIN bereitgestellt werden; andernfalls wird eine, noch in einer Eingabewarteschlange stehende Meldung verarbeitet, oder es wird wieder auf eine Nachricht "gewartet".

Die oben beschriebene Art der Aktivierung der Arbeitsprogramme macht eine übersichtliche Programmierung möglich. Der Programmierer braucht nur die Schnittstelle PAREIN bzw. PARAUS zu kennen und kann dadurch auf einfache Art andere Programme aktivieren.

Die Schnittstelle PARAUS bietet darüberhinaus die Möglichkeit, Aufgaben, die durch Arbeitsprogramme ausgeführt werden, eventuell durch dezentrale Steuereinheiten ausführen zu lassen. Z.B. könnte die Wegesuche für das Zentral-Koppelfeld, die jetzt durch ein Arbeitsprogramm durchgeführt wird, stattdessen auch per Hardware (z.B. Mikroprozessor) ausgeführt werden, wenn diese Steuereinheit realisiert würde. Damit nun in diesem Fall anstelle der Aktivierung des Arbeitsprogrammes "Wegesuche" ein Befehl an die Vermittlungsperipherie gegeben wird, braucht nur der Code in PARAUS geändert zu werden.

Nach der allgemeinen Beschreibung der Organisationsprogramme wird im folgenden Abschnitt 4.5 auf Besonderheiten einzelner Organisationsprogramme eingegangen.

4.5 Besonderheiten einzelner Organisationsprogramme

Die Organisationsprogramme weisen entsprechend dem Aufgabenspektrum der Arbeitsprogramme, die sie zur Ausführung bringen, einige Besonderheiten auf.

4.5.1 Organisationsprogramm ORG3

Das Organisationsprogramm ORG3 steuert und überwacht den Ablauf der Sonderdienstprogramme. Diese wiederum führen Aufgaben durch, die im Zusammenhang mit einem übergeordneten Steuerrechner stehen. Im realisierten Labormodell (vgl. Abschnitt 1.2) ist nur ein Steuerrechner vorhanden und somit können keine Sonderdienstprogramme von einem anderen Rechner angefordert werden. Es besteht jedoch die Möglichkeit, Nachrichten über den Bedienteil (vgl. Abschnitt 3.12) einzugeben und dort mit

dem Aufruf SENDE NACHRICHT an den Überwacher zu geben, der seinerseits dann das ORG3 aktiviert.

Im Falle einer Rechnerkopplung muß in den Überwacher eine Bearbeitungsroutine eingebaut werden, die die Unterbrechungssignale bearbeitet und das ORG3 aktiviert.

4.5.2 Organisationsprogramm ORG4

Das Organisationsprogramm ORG4 dient im wesentlichen dazu, die Kommunikation zwischen dem Vermittlungspersonal und dem Steuerrechner abzuwickeln. Für die Kommunikation Mensch-Maschine wurden eigene Sprachen entwickelt /26/ und zur Zeit wird auch im CCITT an einer Empfehlung für die "Struktur einer Mensch - Maschine - Sprache für SPC - Fernsprechvermittlungssysteme" gearbeitet.

Für das Organisationsprogramm ORG4 wurde eine sehr einfache Form der Kommunikation gewählt. Der Aufruf einer bestimmten Funktion erfolgt durch Eingabe einer Nachricht nach Abschnitt 4.3 über den Bedienungsblattschreiber. Die Form der Nachricht wurde jedoch dahingehend geändert, dass das erste Codewort der Nachricht direkt die gewünschte Funktion angibt. Für die Codeworte wurden mnemotechnische Abkürzungen gewählt. So wird zum Beispiel eine globale Sprechberechtigung für einen Teilnehmer durch folgende Eingabe bewirkt:

```
BEGB  =  Kt, Ans;
          |    └──── Anschlussnummer
          └─────────── Konzentratornummer
```

Nach Durchführung der Aufgabe wird ein Text ausgedruckt, der angibt, dass die Aufgabe ausgeführt wurde. Die Funktionen, die durch die Dienstprogramme ausgeführt werden und die für die Bedienung notwendigen Codeworte sind in /27/ behandelt.

4.5.3 Organisationsprogramm ORG6

Das Organisationsprogramm ORG6 wird durch den Überwacher alle T_o ms aktiviert. Es hat die Aufgabe Arbeitsprogramme zur Ausführung zu bringen, die zyklisch in gewissen Zeitabständen aktiviert werden müssen. Die Zeitabstände sind das Vielfache der Taktperiode T_o. Für die Steueraufgaben besitzt das ORG6 Aktivierungslisten (Bild 4.9).

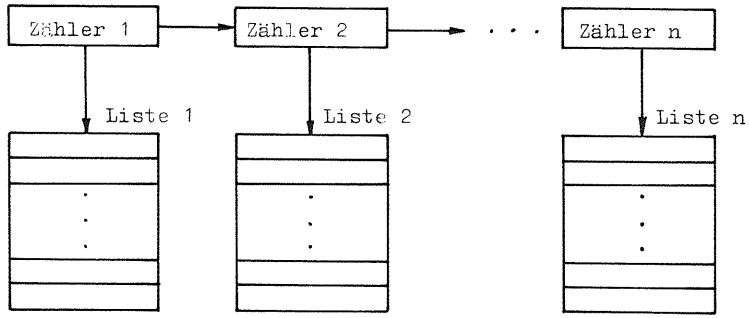


Bild 4.9: Aktivierungslisten für Zeitprogramme

Jede Liste enthält die Nummern der Arbeitsprogramme, die mit einer bestimmten Aufruffrequenz aktiviert werden müssen. Für die Bestimmung der zeitlichen Abstände zwischen den Aktivierungsphasen sind Zähler Z1 bis Zn vorhanden. Die Zähler haben bestimmte Anfangswerte Z1M bis ZnM. Der Zähler Z1 wird alle T_o ms um 1 erniedrigt; wenn er den Wert 0 erreicht, wird der nächste Zähler Z2 um 1 erniedrigt und Z1 wird wieder auf Z1M gesetzt. Jeder weitere Zähler Zi wird um 1 erniedrigt, wenn der Zähler Zi-1 den Wert 0 erreicht.

Durch die Zählerkette wird eine Taktuntersetzung des T_o-Taktes vorgenommen. Die Aufruffrequenz für die Arbeitsprogramme einer Liste i ist somit:

$$f = \frac{1}{T_o \cdot \prod_{k=1}^{i} ZkM}$$

4.5.4 Organisationsprogramm ORG7

Das Organisationsprogramm ORG7 steuert den Ablauf der Vermittlungsprogramme, der Dienst- und Routineprüfprogramme. Da ein Teil der Dienstprogramme und die Routineprüfprogramme nicht arbeitsspeicherresident sind, muss das ORG7 einen Programmteil enthalten, der das Laden dieser Programme veranlasst und die Ausführung überwacht. Die dazu notwendigen Abläufe sind im Bild 4.10 dargestellt.

Das Bild zeigt den Steuerfluss im Organisationsprogramm ORG7. Der linke Teil entspricht genau dem allgemeinen Ablaufteil in einem Organisationsprogramm nach Abschnitt 4.4. Als Arbeitsprogramme kommen hier die Vermittlungsprogramme und die Dienstprogramme zur Ausführung. Der rechte Teil des Bildes stellt die Ausführung von Arbeitsprogrammen dar, die nicht arbeitsspeicherresident sind. Dieser Teil wird im folgenden kurz erläutert.

Es erfolgt zunächst die Abfrage ob das Programm schon geladen ist. Wenn dies nicht der Fall ist, wird geprüft, ob der Überwacheraufruf dafür schon abgesetzt wurde. Wenn der Aufruf noch nicht erfolgt ist, werden die notwendigen Parameter für den Aufruf (Name der Datei in der das Programm abgelegt ist und Tasknummer) zusammengestellt und der Aufruf durchlaufen. Es wird nicht auf das Ende des Ladens mit dem Überwacheraufruf WARTEN (Aufruf) gewartet, da in der Wartezeit (einige 100 ms) keine neu eintreffenden Meldungen von der Vermittlungsperipherie bearbeitet werden können, sondern es wird geprüft, ob Meldungen eingetroffen sind und diese dann gegebenenfalls bearbeitet. Wenn keine Meldungen eingetroffen sind, wird im Programm die Schleife so lange durchlaufen, bis das Programm geladen ist. Wenn das Programm geladen ist, wird es als Task Nr. 7 gestartet. Damit nun diese Task aktiv werden kann, wird der Überwacheraufruf PRIORITÄTSVERZICHT durchlaufen. (Das ORG7 läuft als Task Nr. 6 und hat damit höhere Priorität als das Arbeitsprogramm Task Nr 7 .)

Wenn das Arbeitsprogramm z.B. bei einer Ein-Ausgabeoperation auf das Ende der Operation wartet, kann das Organisationsprogramme weiterlaufen (gestrichelter Pfeil in Bild 4.10). Es prüft, ob Meldungen von der Peripherie eingetroffen sind und arbeitet diese gegebenenfalls ab. Wenn keine Meldungen eingetroffen sind, wird eine weitere Programmschleife durchlaufen, in der auf das Ende der Task 7 (Arbeitsprogramm ist fertig) gewartet wird.

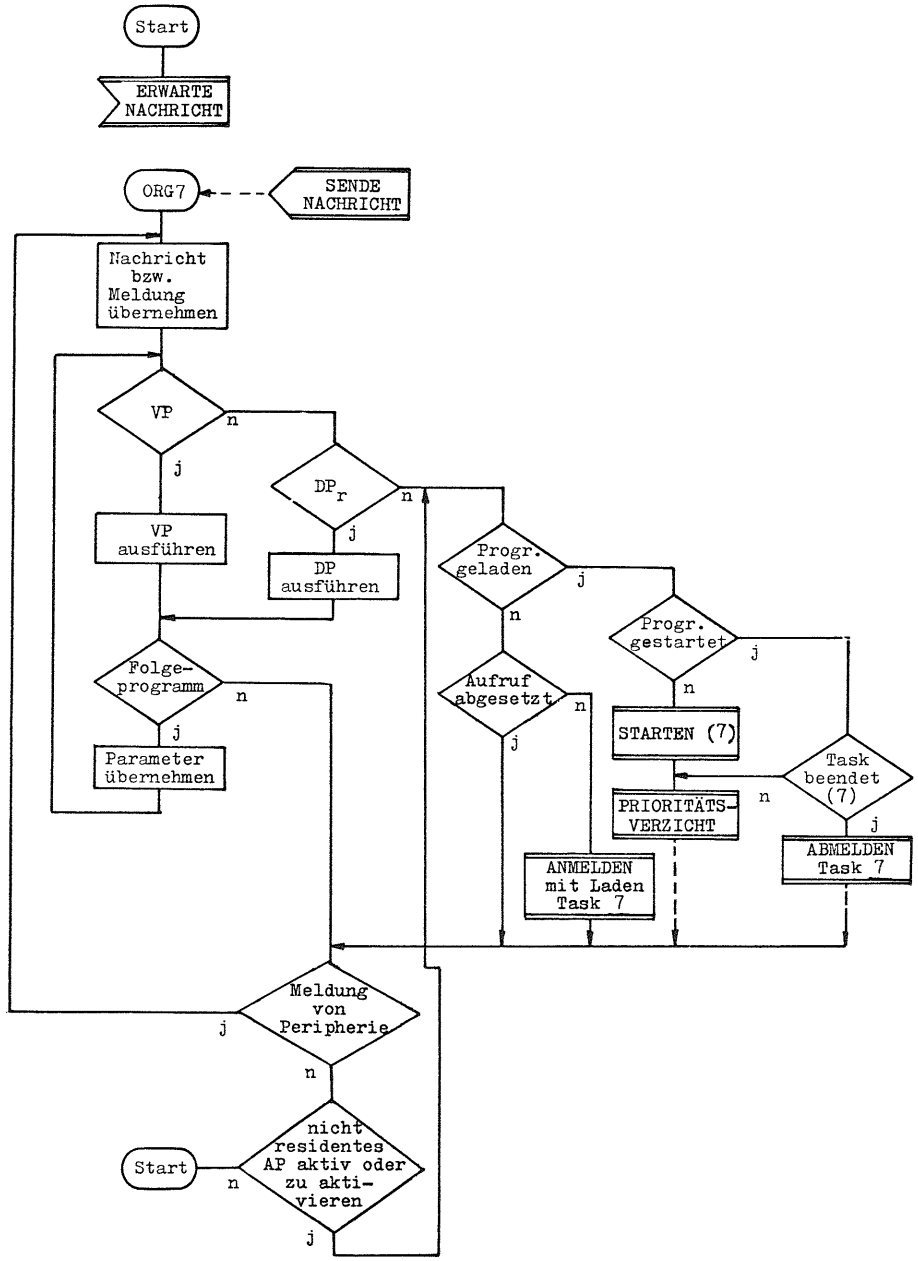


Bild 4.10: Steuerfluß im Organisationsprogramm ORG7

Wenn die Task 7 beendet ist, wird sie mit dem Überwacheraufruf ABMELDEN aus der Buchführung des Überwachers gelöscht und der benötigte Speicherplatz freigegeben. Danach wird wieder, wenn inzwischen keine Meldungen von der Peripherie eingetroffen sind, auf das Eintrffen einer Nachricht gewartet.

In der Zeit, in der ein nicht arbeitsspeicherresidentes Arbeitsprogramm geladen und ausgeführt wird, können keine Nachrichten von anderen Organisationsprogrammen entgegengenommen werden. Dies hat jedoch keine Auswirkungen, da die Nachricht ohnehin nicht bearbeitet werden kann.

Für Testzwecke wurde in das ORG7, die Möglichkeit einprogrammiert, den Inhalt der Pufferbereiche PAREIN und PARAUS (vgl. Abschnitt 4.4 - Bild 4.8) über den Schnelldrucker oder ein Datensichtgerät auszugeben. Es ist dadurch möglich, die Ablaufreihenfolge von Vermittlungsprogrammen zu verfolgen und die übergebenen Parameter zu prüfen.

5. Speicherorganisation

5.1 Allgemeines

Der Arbeitsspeicher eines Vermittlungsrechners enthält alle Programme, die für die Ausführung zeitkritischer Aufgaben notwendig sind. Programme, für deren Ausführung keine Zeitbedingungen gelten, werden auf einem Hintergrundspeicher bereitgehalten und können bei Bedarf in den Arbeitsspeicher geladen werden. Es ist auch möglich, wenn der Vermittlungsrechner keinen eigenen Hintergrundspeicher besitzt, diese Programme von einem zentralen Bedienrechner anzufordern /8/.

Der Speicherbedarf für die Programmbefehle liegt bei heute üblichen Vermittlungsrechnern in der Größenordnung von 100K - 500K byte (1K = 1024) und der Datenumfang beträgt je nach Ausbau der Vermittlungsstelle 10 - 500K byte. Bei den Daten werden feste und variable Daten unterschieden. Feste Daten sind z.B. Konstanten, Bitmuster und Zuordnungstabellen für die Zuordnung einer Rufnummer zu einem Anschluß. Variable Daten sind z.B. die Daten, die für den Auf- und Abbau einer Verbindung benötigt werden und sich je nach dem Stand der Verbindung ändern.

Derart umfangreiche Programmsysteme sind wegen ihrer Komplexität nie ganz fehlerfrei. Um nun die Befehle und festen Daten gegen Überschreiben durch falschlaufende Programme zu schützen, wird eine Trennung von Befehlen, festen Daten und variablen Daten vorgenommen (Bild 5.1).

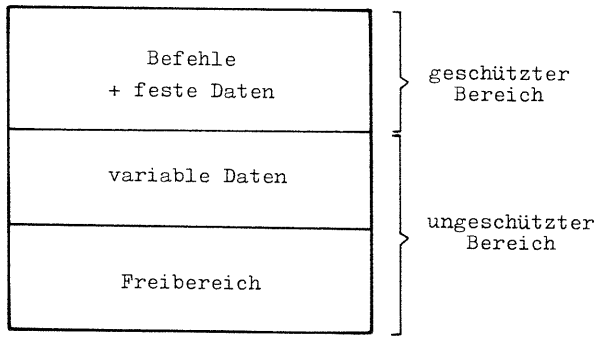


Bild 5.1: Prinzipielle Speicherorganisation eines Vermittlungsrechners

Die Befehle und festen Daten werden in einem geschützten Arbeitsspeicherbereich abgespeichert, aus dem nur gelesen werden darf,und die variablen Daten kommen in den ungeschützten Bereich des Arbeitsspeichers. Bei dem System No. 1ESS von Bell /28/ wurden aus wirtschaftlichen Gründen sogar unterschiedliche Speichermedien für die beiden Speicherbereiche verwendet. (Twistor-Speicher als "Read Only Memory" für Befehle und feste Daten und ein Ferritplättchen-Speicher für die variablen Daten.) Der Freibereich des Speichers dient für die nicht arbeitsspeicherresidenten Programme, die bei Bedarf von einem Hintergrundspeicher eingelesen werden.

Neben der Aufteilung des Speichers in einen geschützten und ungeschützten Bereich werden die Befehle und Daten noch durch zusätzliche Kontrollbits gesichert. Meist wird auch eine Duplizierung der Speicher vorgenommen.

Für die Systemsoftware des NVDV-Laborprojektes wird nun in den folgenden Abschnitten eine Speicherorganisation entworfen, die einen gewissen Schutz der Befehle und festen Daten gegen Überschreiben durch fehlerhafte Programme bietet. Zunächst werden jedoch die Randbedingungen beschrieben, die durch den Rechner Siemens 306 vorgegeben sind.

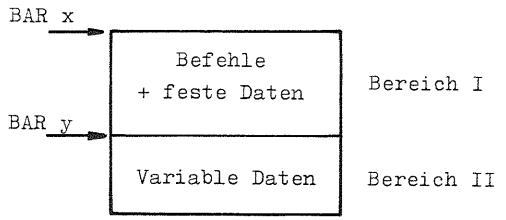
5.2 Randbedingungen durch den Rechner Siemens 306

Der Rechner Siemens 306 (S306) bietet die Möglichkeit, die ersten 10K Worte des Arbeitsspeichers durch einen Hardware-Schreibschutz zu schützen (vorderer Arbeitsspeicherbereich). Die Konzeption der Zentraleinheit ist jedoch darauf ausgelegt, dass in diesem Arbeitsspeicherbereich der Überwacher abgespeichert wird. (Es sind in diesem Bereich Arbeitsspeicherzellen, die für die Anzeigen von Ein-Ausgabegeräten und die Abspeicherung von Unterbrechungsursachen benötigt werden. Ferner kann im Alarmzustand, in dem die Taktbearbeitungsroutine zur Ausführung kommt, nur absolut adressiert werden, was bei einem Adressierungsbereich von 16K Worten zur Folge hat, daß diese Bearbeitungsroutine im vorderen Speicherbereich zu stehen hat.)

Der Überwacher, wie er in Kapitel 3 beschrieben wurde, benötigt etwa 8K Worte Speicherplatz. Da er aus oben angeführten Gründen im vorderen Bereich des Arbeitsspeichers (Adresse 0...8000) stehen muß, können außer dem Überwacher noch 2K Worte des Arbeitsspeichers durch den Hardware-Schreibschutz geschützt werden. Dies reicht jedoch nicht aus, da die Befehle der Organisations- und Arbeitsprogramme mehr Speicherplatz beanspruchen.

Für alle Programme, die im Normalzustand der Zentraleinheit zur Ausführung kommen, gilt die relative Adressierung, d.h., es wird grundsätzlich vor jedem Speicherzugriff zu einer Operandenadresse der Inhalt eines von zwei Basisadressenregistern addiert. Dasselbe gilt auch für die Adressierung von Befehlen über den Befehlszähler.

Diese Eigenschaft des Rechners kann dazu benutzt werden, einen zusätzlichen Speicherschutz zu realisieren. Man trennt dazu die Befehlsworte und festen Datenworte von den variablen Datenworten eines Programmes (Bild 5.2) und legt sie in getrennten Speicherbereichen ab. Für die Adressierung jedes Bereiches wird ein eigenes Basisadressenregister (BAR) verwendet.



BAR Basisadressenregister

Bild 5.2: Adressierungsmöglichkeit für Befehle und Daten

Für die Zuordnung der Basisadressenregister gibt es zwei Möglichkeiten:

I	Bereich I	BAR 9
	Bereich II	BAR 15
II	Bereich I	BAR 15
	Bereich II	BAR 9

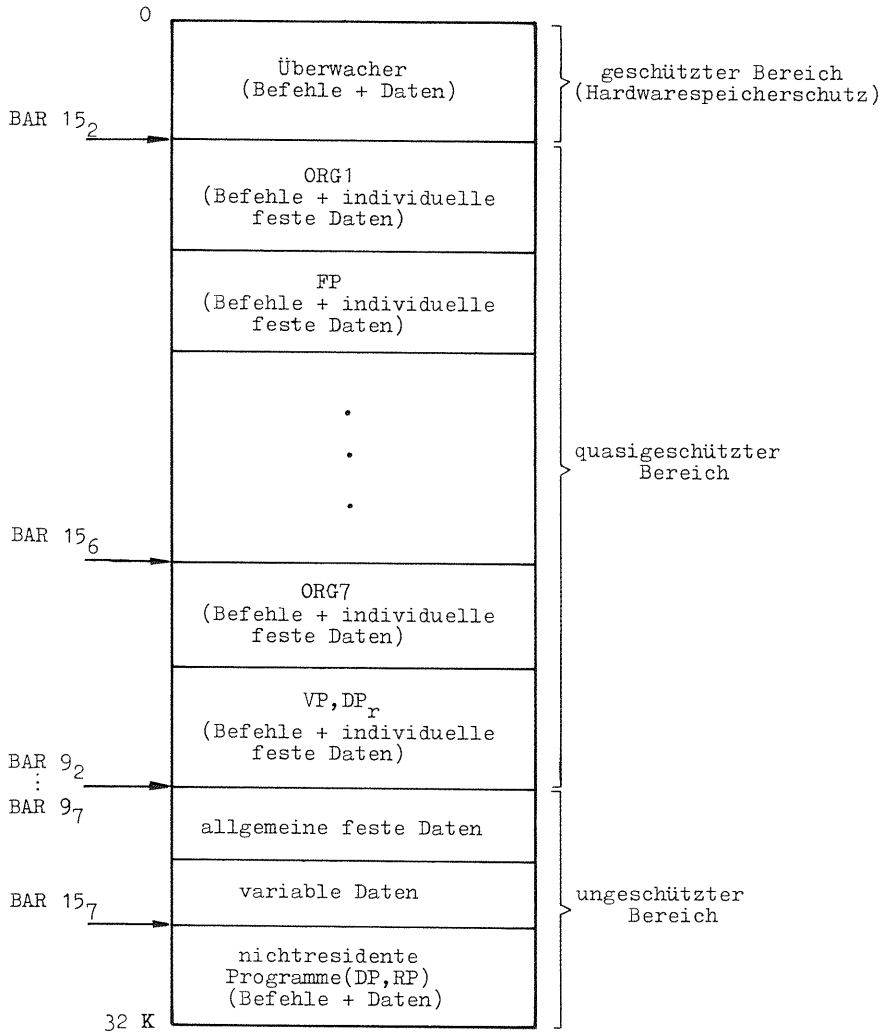
Zur Auswahl des Basisadressenregisters bei der Adressenumrechnung wird das Bit 16 des Befehlwortes herangezogen. (Bit 16 = 0 Adressierung mit Hilfe des BAR 9, Bit 16 = 1 Adressierung mit Hilfe des BAR 15.) Bei adressarithmetischen Befehlen muss der Befehl, wenn das Bit 16 gesetzt ist, durch den Überwacher simuliert werden, wozu natürlich Rechenzeit benötigt wird (Overhead, vgl. Abschnitt 3.6.3.1). Um den Overhead möglichst gering zu halten, sollte das Bit 16 bei adressarithmetischen Befehlen nicht gesetzt sein. Adressarithmetische Befehle werden in der Hauptsache ausgeführt,um zu Daten zuzugreifen. Aus diesem Grund sollte die Möglichkeit II realisiert werden, da hier die Daten über das BAR 9 (Bit 16=0) adressiert werden.

Durch diese Art der Aufteilung der Befehle und Daten ist ein gewisser Speicherschutz gewährleistet (quasigeschützter Bereich). Da es keine negativen Adressen gibt,ist ein Zugriff über das BAR 9 (=BAR y) zum Bereich I nicht möglich. Der Speicherschutz ist natürlich eingeschränkt, denn, wenn ein fehlerhaftes Programm Befehle durch Setzen von Bit 16 im Befehlswort verändert, ist auch ein Zugriff zu dem anderen Speicherbereich möglich. Dadurch könnten Befehle und feste Daten durch Überschreiben zerstört werden.

Bei Zugriffen zu festen Daten im Bereich I entsteht ein Overhead durch die Befehlssimulation im Überwacher sobald hierfür adressarithmetische Operationen erforderlich sind. Dieser Overhead kann vermieden werden, wenn die festen Daten auch in den Bereich II gelegt werden. Dadurch geht aber der Speicherschutz verloren.

5.3 Realisierte Speicherorganisation

Die Organisationsprogramme und die ihnen zugeordneten Arbeitsprogramme werden entsprechend ihrer Dringlichkeit unter individuellen Tasknummern vom Überwacher verwaltet und zur Ausführung gebracht. (vgl. Abschnitt 4.2, z.B. kommt das Organisationsprogramm ORG1 mit seinem Fehlerbehandlungsprogramm unter der Tasknummer 2 zur Ausführung.) Jeder Task i sind vom Überwacher individuelle Basisadressenregister BAR 9_i und BAR 15_i zugeordnet, die bei der Aktivierung der Task (Übergang der Task vom Zustand bereit in den Zustand aktiv) in die beiden Basisadressenregister der Zentraleinheit geladen werden. Damit ist es möglich für jedes Organisationsprogramm mit den Arbeitsprogrammen eine Aufteilung in die Bereiche I und II nach Bild 5.2 vorzunehmen. Dadurch ergibt sich die in Bild 5.3 angegebene Speicherorganisation.



BAR x_i	Basisadressenregister x der Index i gibt die Tasknummer an	VP	Vermittlungsprogramme
ORG	Organisationsprogramme	DP r	Dienstprogramme (resident im Arbeitsspeicher)
FP	Fehlerbehandlungsprogramme	RP	Routineprüfprogramme

Bild 5.3: Speicherorganisation für den Rechner Siemens 306

Im vorderen Bereich des Arbeitsspeichers wird der Überwacher abgelegt. Er kann durch den Hardwarespeicherschutz der Zentraleinheit geschützt werden. Zusätzlich ist er durch den Adressierungsmodus der Zentraleinheit im Normalzustand (relative Adressierung, vgl. Anhang A3) gegenüber allen Organisations- und Arbeitsprogrammen geschützt, die in diesem Zustand zur Ausführung kommen.

Hinter dem Überwacher stehen von den Organisations- und Arbeitsprogrammen die Befehle und die individuellen festen Daten. Die festen Daten, die für alle Programme gemeinsam sind, und die variablen Daten aller Programme wurden in dem Bereich dahinter zusammengefasst. Die Zusammenfassung der festen Daten erfolgte aus Gründen der Speicherplatzersparnis, da dadurch die Daten nur einmal im Arbeitsspeicher abgelegt werden müssen und nicht individuell pro Organisationsprogramm. Der freibleibende Rest des Arbeitsspeichers ist für die nicht arbeitsspeicherresidenten Dienstprogramme und Routineprüfprogramme des ORG7 vorgesehen. In diesem Bereich werden die Programme bei Bedarf vom ORG7 geladen.

Die Basisadressenregister BAR 15_i (i = 2...7) werden auf die jeweiligen Arbeitsspeicherbereiche beim Laden durch den Überwacher eingestellt. Die Basisadressenregister BAR 9_i (i = 2...6) müssen beim ersten Aktivieren des Systems durch die Organisationsprogramme auf den Anfang des allgemeinen festen Datenbereiches eingestellt werden. Bei den nicht arbeitsspeicherresidenten Arbeitsprogrammen, welche über das BAR 9_7 zu allgemeinen festen Daten zugreifen, muß dies nach dem Laden und Aktivieren des Programmes durch das Programm selbst erfolgen.

Bei der Programmierung wurde der Bereich der allgemeinen festen Daten und der variablen dem ORG7 zugeordnet. Die Einstellung des BAR 9_6 erfolgt durch einen Überwacheraufruf für das Verschieben von Basisadressenregistern. Die Einstellung der BAR 9_i ($i \neq 6$) erfolgt durch einen Überwacheraufruf zum Koppeln an das BAR 9_6, wobei das BAR 9_i den Wert von BAR 9_6 erhält.

Die Speicherorganisation nach Bild 5.3 ermöglicht einen quasigeschützten Arbeitsspeicherbereich für die Befehle und individuellen festen Daten der Organisations- und Arbeitsprogramme. Die allgemeinen festen und variablen Daten aller Programme sowie die nicht arbeitsspeicherresidenten Programme liegen im ungeschützten Bereich des Arbeitsspeichers.

6. Simulation der Vermittlungsperipherie

6.1 Allgemeines

Beim Entwurf von komplexen Hardwaresteuerungen wird heute oft das Hilfsmittel der Simulation auf einem Digitalrechner eingesetzt. Dabei wird die Steuerung auf ein Modell abgebildet, das in einem Programm realisiert wird und die determinierten Abläufe in der Steuerung wiedergibt. Am Beispiel einer dezentralen Steuereinheit (Konzentratoranschlußschaltung ,KAS, vgl. Abschnitt 1.2) werden in Abschnitt 6.2 die Möglichkeiten aufgezeigt, welche diese Art der Simulation bietet.

Die Systemsoftware der Vermittlungsstelle umfaßt sehr viele umfangreiche Programme. Die Entwicklung der Software läuft in der Regel parallel zur Hardwareentwicklung der Systemkomponenten. Es ist daher wünschenswert, eine Möglichkeit zu besitzen, die Software auszutesten, auch wenn die Hardware noch nicht oder nur teilweise vorhanden ist. Hierzu wird ebenfalls das Hilfsmittel der digitalen Simulation eingesetzt. Dabei wird die Umwelt der Software (Vermittlungsperipherie) in einem Modell nachgebildet (Umweltsimulation). Diese Art der Simulation wird in Abschnitt 6.3 behandelt.

6.2 Simulation einer Konzentratoranschlußschaltung

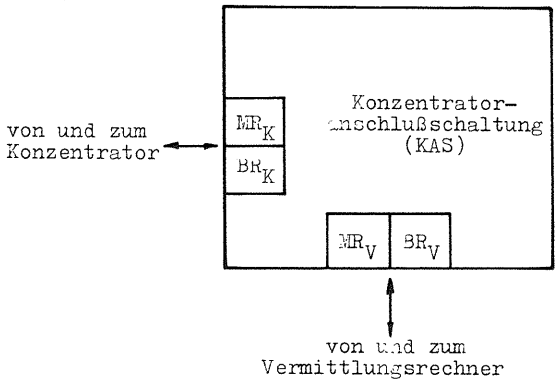
6.2.1 Möglichkeiten der Simulation

Bei dem PCM-Projekt des Instituts NVDV sind in der Vermittlungsstelle dezentrale Steuereinheiten vorhanden (vgl. Abschnitt 1.2). Jedem Konzentrator im Vorfeld ist eine Konzentratoranschlußschaltung (KAS) in der Vermittlungsstelle zugeordnet, welche die Durchschaltung der Teilnehmer über deren Konzentrator von und zum Zentral-Koppelfeld vornimmt. Die KAS führt also eine dezentrale Vorverarbeitung der Steuerinformationen durch und entlastet damit den Steuerrechner.

Als Steuerung für die KAS ist eine Mikroprogrammsteuerung mit mehreren Subsystemen für einzelne Aufgaben vorgesehen. Beim Entwurf und der Realisierung der KAS würden ohne Rechnerunterstützung folgende Probleme auftreten:

- die Überprüfung der logischen Funktion auf prinzipielle Fehler ist erst nach dem Aufbau einer KAS möglich
- die Auswirkung einer fehlerhaften Umgebung ist schlecht nachprüfbar
- die Fehlersuche ist durch mangelnden Bedienungskomfort erschwert
- der Vergleich mehrerer Entwurfsvarianten ist nur schwer möglich
- **eventuelle Änderungen oder Erweiterungen sind oft nur unter großem Zeit- und Kostenaufwand möglich**
- die Untersuchung des Zusammenwirkens der KAS mit den anderen Systemkomponenten Konzentrator und Vermittlungsrechner ist u. U. nicht möglich, da diese zu dem Zeitpunkt noch nicht oder nur teilweise zur Verfügung stehen

Ein Hilfsmittel zur Lösung dieses Problems ist die Simulation der KAS mit Hilfe eines Digitalrechners. Dabei wird die KAS auf ein Modell abgebildet. Das Modell muss bezüglich der Schnittstellen nach aussen dieselben Eigenschaften besitzen, wie die reale KAS.



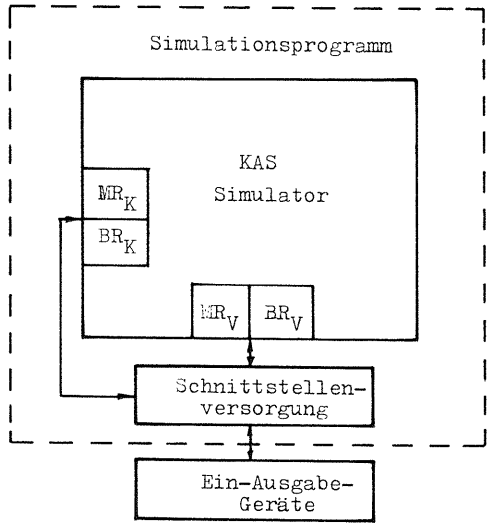
MR Melderegister — Index : K Konzentrator
BR Befehlsregister — V Vermittlungsrechner

Bild 6.1: Schnittstellen der Konzentratoranschlußschaltung

Die KAS besitzt nach aussen zwei Schnittstellen, eine zum Konzentrator (Kt) und eine zum Vermittlungsrechner (VR). Diese bestehen jeweils aus zwei Registern, über die der Meldungs- und Befehlsaustausch mit dem Kt bzw. dem VR durchgeführt wird.

Vom Kt werden Meldungen empfangen (z.B. Teilnehmerzustandsänderung) und Befehle an diesen abgegeben (z.B. Durchschalten im Konzentratorkoppelnetz). Dagegen werden vom VR Befehle empfangen (z.B. zum Anschalten eines Hörtones) und es werden Meldungen an diesen generiert (z.B. Teilnehmer x hat abgehoben und ist mit Kanal y verbunden). Auf eine Meldung vom Kt oder auf einen Befehl vom VR hin wird in der KAS ein Mikroprogramm aktiviert und die entsprechende Aufgabe ausgeführt. Dabei können Befehle an den Kt oder eine Meldung an den VR generiert werden. Neben den Mikroprogrammen sind auch Speicher in den einzelnen Subsystemen der KAS vorhanden, in denen der Belegungszustand von Kanälen und Wählzeichenempfängern vermerkt ist.

Das Modell der KAS muss nun alle Abläufe und die Register und Speicher in der KAS nachbilden. Die Schnittstellen nach aussen werden durch ein Ein-Ausgabegerät (z.B. Blattschreiber) ersetzt (Bild 6.2). Dadurch können Meldungen bzw. Befehle eingegeben werden und die Reaktionen der KAS verfolgt werden. (Das Simulationsprogramm wird im Abschnitt 6.2.2 beschrieben.)



MR Melderegister — Index : K Konzentrator
BR Befehlsregister — V Vermittlungsrechner

Bild 6.2: Logische Simulation der Konzentratoranschlußschaltung

Diese Art der Simulation wird als LOGISCHE SIMULATION bezeichnet, da sie die logischen Abläufe in der KAS wiedergibt. Es können bei der Simulation logische Fehler sofort erkannt werden, da die Reaktionen auf Befehle vom VR bzw. auf Meldungen vom Kt durch den Entwurf bekannt sind. Die Fehlersuche wird erleichtert, da die Speicher- und Registerinhalte ausgedruckt werden können.

Die Untersuchung der Auswirkung einer fehlerbehafteten Umgebung (der Kt bringt falsche Meldungen oder die Meldungen sind durch Übertragungsstörungen verfälscht) kann einfach durch die Eingabe von falschen Informationen durchgeführt werden.

Das Zusammenwirken der KAS mit dem Kt und dem VR kann dadurch untersucht werden, dass die Meldungen und Befehle in der Reihenfolge eingegeben werden, wie sie im realen Zusammenspiel der Systemkomponenten auftreten.

Weiterhin können mehrere Entwurfsvarianten für eine KAS durch entsprechende Programmvarianten einfach miteinander verglichen werden.

Die Protokolle von solchen Simulationsläufen eignen sich außerdem besonders gut zur Änderungskontrolle und zur Dokumentation.

Durch diese Art der Simulation ist eine Untersuchung der KAS auf der Schaltungsebene (Aufbau mit verschiedenen digitalen Bausteinen und deren Verschaltung) nicht möglich. Man könnte diese Untersuchung jedoch auch mittels Simulation durchführen. Hierzu sind in der Literatur einige Programmsysteme bekannt /29,30/.

6.2.2 Simulationsprogramm für eine KAS

Das Simulationsprogramm /31/ zur logischen Simulation der Konzentratoranschlußschaltung wird im folgenden nur im Prinzip an einem vereinfachten Flußdiagramm (Bild 6.3) beschrieben.

Das Programm gliedert sich in einen Teil für die Schnittstellenversorgung und in einen Teil, der den eigentlichen Simulator darstellt. Die Schnittstellenversorgung dient dazu, Meldungen und Befehle, die vom Konzentrator bzw. vom Vermittlungsrechner kommen, in die betreffenden Register einzutragen bzw. Meldungen und Befehle aus diesen Registern auszugeben. Die Meldungen stellen Bitmuster dar und enthalten in der Regel eine Kennung und verschiedene Parameter. (Die Formate für Rechnermeldungen und -befehle sind im Anhang A1.4 angegeben.)

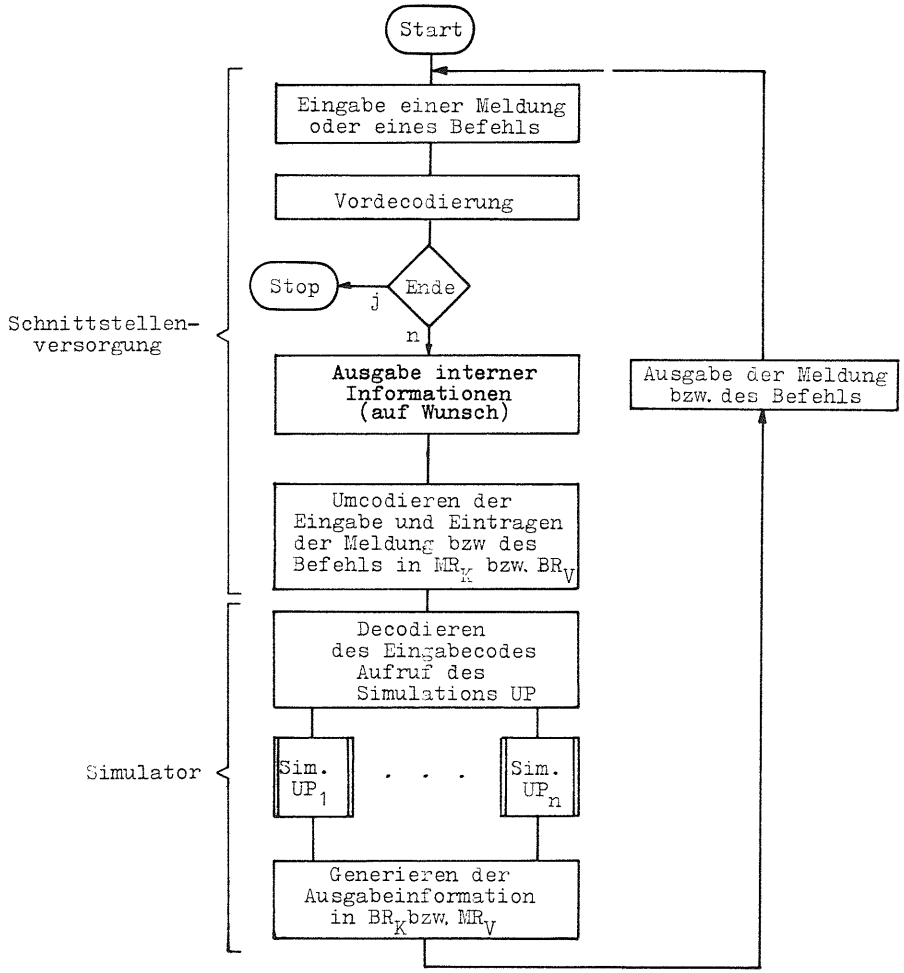


Bild 6.3: Flußdiagramm für die Simulation der Kontentrator-anschlußschaltung

Um die Benutzung des Programmes zu erleichtern, können die Meldungen und Befehle mit mnemotechnischen Abkürzungen und die verschiedenen Parameter, die neben der Kennung enthalten sind, als Zahlen ein- und ausgegeben werden. Dadurch erhält man eine leicht überschaubare Dokumentation. Weiter ermöglicht die Schnittstellenversorgung auf Wunsch eine Ausgabe der Inhalte der Datenspeicher in der simulierten KAS.

Nach dem Eintragen der eingegebenen Meldung bzw. der Befehle in das entsprechende Register wird der eigentliche Simulator angesteuert. Dieser ist als Makroprogrammsimulator realisiert /32/. Der Decodierungsteil entnimmt die in einem der Register stehende Meldung oder den Befehl und steuert ein dazu spezifisches Unterprogramm an. Dieses Unterprogramm führt dieselben Aufgaben wie das zu simulierende Mikroprogramm aus. Das Ergebnis des Unterprogrammlaufes ergibt dann wieder eine Meldung oder einen Befehl, welche oder welcher in das entsprechende Register eingetragen wird. Damit ist die Aufgabe des Simulators beendet und der Schnittstellenversorgungsteil gibt die im simulierten Register stehende Information (Meldung oder Befehl) aus.

6.2.3 Betriebsmäßiger Ersatz der KAS durch das Simualtionsprogramm

Das oben beschriebene Simulationsprogramm zur logischen Simulation der Konzentratoranschlußschaltung kann auch dazu benutzt werden, die KAS funktionsmäßig zu ersetzen. Dabei wird der Konzentrator direkt an den Vermittlungsrechner angeschlossen und die Funktionen der KAS werden im Vermittlungsrechner "simuliert" (Bild 6.4).

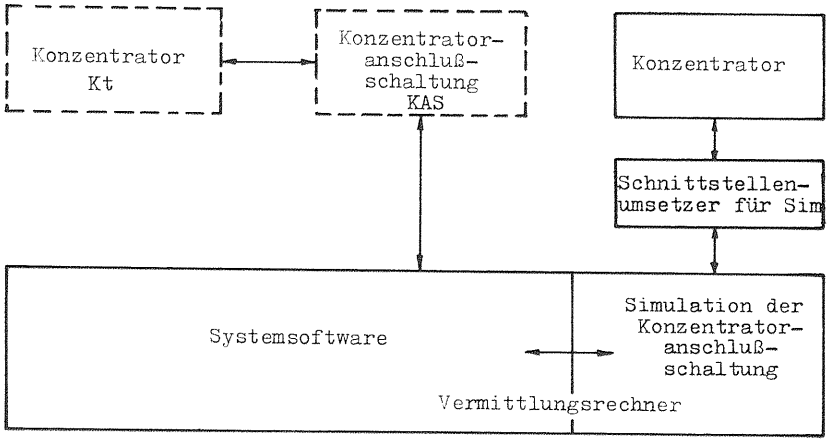


Bild 6.4: Funktionelle Simulation der Konzentratoranschlußschaltung

— — — realer Betrieb von Kt und KAS
——— Betrieb bei Simulation der KAS

In diesem Fall müssen bezüglich der Funktion der KAS natürlich geringe Einschränkungen in Kauf genommen werden (z.B. können keine Hörtöne mehr angeschaltet werden). Diese Art der Simulation erlaubt es jedoch im Laborprojekt das Zusammenwirken mehrerer Konzentratoren über die Konzentratoranschlußschaltungen mit dem Rechner auszutesten. Dabei ist allerdings zu beachten, dass das reale zeitliche Verhalten des Vermittlungsrechners durch das Simulationsprogramm verändert wird.

Die beschriebene Art der logischen und funktionellen Simulation der KAS lässt sich in derselben Weise auch auf die Vermittlungsstellenanschlußschaltung (VAS) anwenden. Die Simulation wurde jedoch nicht durchgeführt, da für das Laborprojekt keine VAS aufgebaut wird. (Die VAS ist notwendig, um Externverkehr zu anderen Vermittlungsstellen abzuwickeln. Da kein weiterer Rechner für eine zweite Vermittlungsstelle vorhanden ist, erübrigt es sich auch, die für Externverkehr notwendigen Steuereinheiten aufzubauen.)

6.3 Testmöglichkeiten für die Systemsoftware durch Simulation

6.3.1 Allgemeines

Die Erstellung der Systemsoftware läuft im allgemeinen parallel zur Hardwareentwicklung für die Vermittlungsperipherie. Es ist daher wünschenswert, ein Hilfsmittel zu besitzen, das es gestattet, die Programme auch schon zu testen, wenn diese Hardware noch nicht oder nur teilweise vorhanden ist. Weitere Gründe dafür sind:

- Die Peripherie arbeitet zum Zeitpunkt des Testens selbst noch nicht ganz fehlerfrei.
- Es ist meistens unmöglich den fehlerverursachenden Zustand der Vermittlungsperipherie wieder herzustellen.
- Bei echter Umgebung des Rechners können Zeitverhältnisse sehr schwer geändert werden, um dadurch Zustände herbeizuführen, die nur sehr selten auftreten (z.B. verzögerte Bearbeitung eines Befehls vom Vermittlungsrechner in der Konzentratoranschlußschaltung).
- Der Zeitaufwand um sehr viele"Ereignisse"(z.B. Abheben von Teilnehmern) zu erzeugen ist beachtlich.
- Es ist teilweise schwierig, Fehler in die Peripherie einzubauen, um auch falsche Meldungen an den Rechner zu generieren.

Als Hilfsmittel zum Austesten der Programme bietet sich die Simulation an. Es gibt dazu verschiedene Möglichkeiten (Bild 6.5).

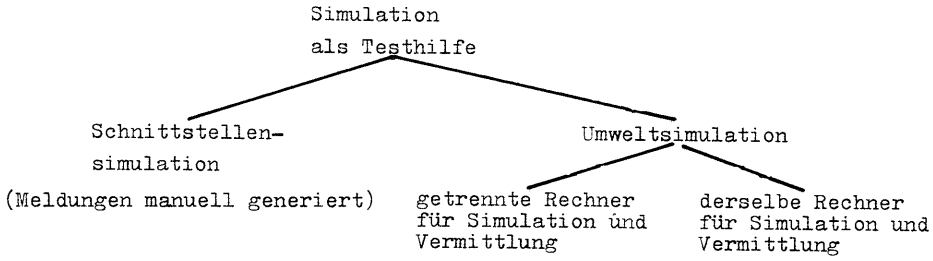


Bild 6.5: Simulation als Testhilfe für die Systemsoftware

Als einfachste Möglichkeit bietet sich die Schnittstellensimulation an /33,34/.

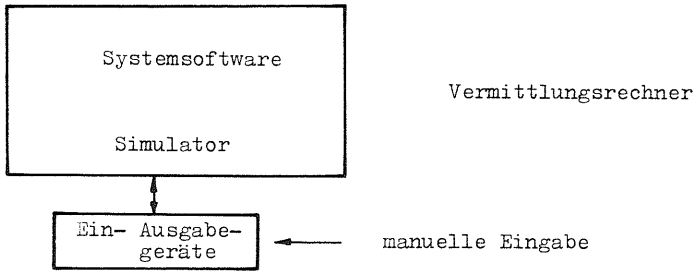


Bild 6.6:Schnittstellensimulation

Bei der Schnittstellensimulation wird die Ein-Ausgabeschnittstelle zur Vermittlungsperipherie durch einen Simulator ersetzt, der es gestattet, über ein Eingabegerät beliebige Meldungen in den Vermittlungsrechner (VR) einzugeben und Reaktionen des VR (Befehle) auf einem Ausgabegerät auszugeben. Darüber hinaus können beliebige Speicherinhalte des VR protokolliert werden.

Die Ein-Ausgabe der Meldungen und Befehle kann über einen Blattschreiber erfolgen oder es können auch ganze Ketten von Meldungen von Lochkarten oder einem Magnetbandgerät eingelesen werden. Es ist auch leicht möglich, falsche Meldungen oder Meldungen in falscher Reihenfolge einzugeben, um das Verhalten der Programme in einer

fehlerbehafteten Umgebung (Vermittlungsperipherie) auszutesten. Darüberhinaus lassen sich die Abläufe beliebig oft unter denselben Voraussetzungen wiederholen. Mit der Schnittstellensimulation läßt sich allerdings das zeitliche Verhalten des Systems nicht austesten.

Die Schnittstellensimulation hat den Nachteil, daß die Meldungen an den Rechner durch das Testpersonal erstellt werden müssen und ebenso auch die Reaktionen des Vermittlungsrechners überprüft werden müssen.

Bei der Umweltsimulation wird dieser Nachteil ausgeschaltet, da hierbei die gesamte Vermittlungsperipherie einschließlich des Teilnehmerverhaltens durch ein Programm (Simulator) nachgebildet wird. Man unterscheidet dabei zwei Verfahren (Bild 6.7).

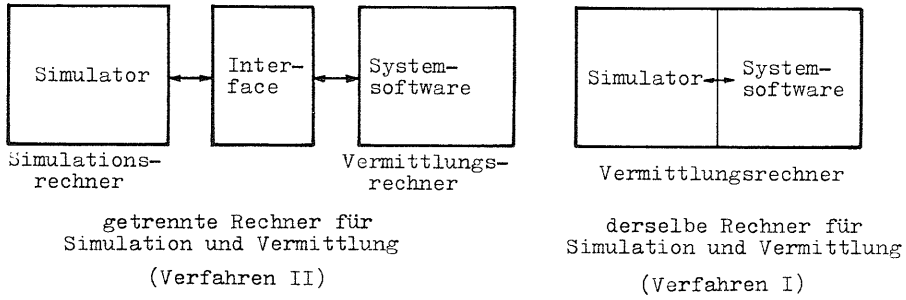


Bild 6.7: Umweltsimulation

Beim Verfahren I kommt der Simulator zusammen mit der Vermittlungssoftware auf demselben Steuerrechner zur Ausführung /35/. Bei dieser Art der Simulation muß der Umweltsimulator die Aufgaben der Schnittstellensimulation mit übernehmen. Die Abläufe können nicht in Echtzeit ausgeführt werden, da der Simulator auch Rechenzeit beansprucht.

Beim Verfahren II kommt der Simulator auf einem zweiten Rechner zur Ausführung /36-40/. Der zweite Rechner (Simulationsrechner) ist über die Schnittstelle, an die normalerweise die Vermittlungsperipherie angeschlossen ist, mit dem Vermittlungsrechner gekoppelt. Im VR brauchen daher keine Veränderungen durchgeführt zu werden.

Bei beiden Verfahren werden durch den Simulator Meldungen an die Vermittlungssoftware generiert und Befehle von dieser empfangen. Die Befehle werden in der Regel dabei auf Zulässigkeit geprüft (Erkennung von Softwarefehlern in der Vermittlungssoftware).

Das Verfahren II ermöglicht darüberhinaus auch noch die "echten" zeitlichen Abhängigkeiten zwischen aufeinanderfolgenden Meldungen zu berücksichtigen (Echtzeitsimulation). Wenn der Simulator auf dem Simulationsrechner zu langsam ist, um die Vorgänge in Echtzeit nachzubilden, kann der Taktgenerator (Uhr) des Vermittlungsrechners während der Zeit angehalten werden, in welcher der Simulator neue Meldungen generiert /37,38/.Dadurch laufen die Vorgänge aus der Sicht des VR in Echtzeit ab. Durch die Echtzeitsimulation läßt sich dann auch die Leistungsfähigkeit des VR bestimmen.

Falls der VR noch nicht zur Verfügung steht,aber Programme der Systemsoftware getestet werden sollen, kann dies auch auf einem anderen (größeren) Rechner geschehen. Für diesen Rechner muß dann ein spezielles Programm geschrieben werden, welches die Eigenschaften des VR nachbildet /41,42/.

Für das PCM-Projekt NVDV steht nur der Rechner Siemens 306 zur Verfügung. Es wurden deshalb die Verfahren der Schnittstellensimulation und der Umweltsimulation nach dem Verfahren I gewählt, welche in den Abschnitten 6.3.2 und 6.3.3 behandelt werden.

6.3.2 Schnittstellensimulation

Im PCM-Projekt NVDV besteht die Schnittstelle zwischen Systemsoftware des Vermittlungsrechners und der Vermittlungsperipherie in den Ein-Ausgabepuffern (EPU bzw. APU) und den dazugehörenden Warteschlangen im Arbeitsspeicher (vgl. Abschnitt 3.5). In diese Puffer werden in festen Taktintervallen Meldungen von der Peripherie über einen Multiplexer eingeschrieben und Befehle ausgelesen. Die Übernahme der Meldungen aus dem Eingabepuffer (EPU) in die Eingabewarteschlange (EWS) bzw. die Übergabe von Befehlen aus der Ausgabewarteschlange (AWS) in den Ausgabepuffer (APU) erfolgt **durch die Taktbearbeitungsroutine (TBR) ebenfalls in festen Taktintervallen (vgl. Abschnitt 3.5.2.1).**

Die Meldungen, die von der Peripherie eintreffen, sind entweder Fehlermeldungen, die vom Organisationsprogramm ORG1 entgegengenommen werden oder reguläre Meldungen, die für das Organisationsprogramm ORG7 bestimmt sind.

Im folgenden wird die Schnittstellensimulation nur für die Klasse der regulären Meldungen behandelt (ORG7), da sie sich für beide Klassen nur durch die Aktivierung des betreffenden Organisationsprogrammes unterscheidet. Das Prinzip der Schnittstellensimulation ist im Bild 6.8 angegeben.

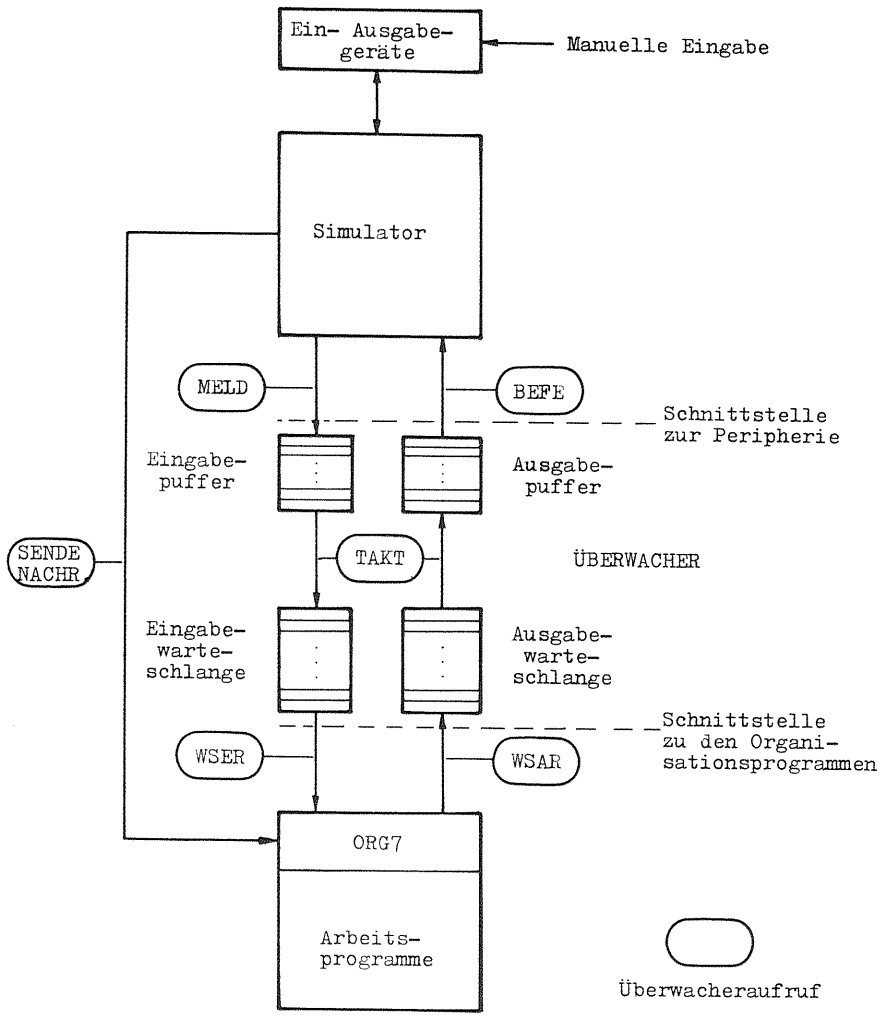


Bild 6.8: Prinzip der Schnittstellensimulation

Im regulären Betrieb wird das ORG7 durch die Taktbearbeitungsroutine (TBR) aktiviert, wenn eine reguläre Meldung von der Peripherie in der Eingabewarteschlange EWSR eingetroffen ist. Das ORG7 kann eine Meldung aus der Eingabewarteschlange mit dem Überwacheraufruf WSER übernehmen und die entsprechenden Aufgaben ausführen. Befehle, die ausgegeben werden müssen, können mit dem Überwacheraufruf WSAR in die Ausgabewarteschlange eingetragen werden.

Durch den Überwacher werden nun Funktionen bereitgestellt, um von einem Programm aus Meldungen in den Eingabepuffer einzuschreiben (Aufruf MELD) und Befehle aus dem Ausgabepuffer auszulesen (Aufruf BEFE). Die Aufgabe der TBR , die Meldungen vom EPU in die EWS zu übernehmen bzw. Befehle aus der AWS in den APU zu transferieren, kann durch den Überwacheraufruf TAKT durchgeführt werden. (Es wird dabei vorausgesetzt, dass der Taktgeber für Alarmsignale zur Aktivierung der TBR abgeschaltet ist.)

Durch die beschriebenen Aufrufe ist es möglich, von dem Simulator aus Meldungen an die Systemsoftware abzugeben bzw. Befehle entgegenzunehmen. Der Simulator muss, wenn er eine Meldung übergeben hat, das ORG mit dem Überwacheraufruf SENDE NACHRICHT aktivieren.

Die Abläufe im Programm /43/ zur Schnittstellensimulation sind im Bild 6.9 im Prinzip dargestellt. Um den Ablauf besser erklären zu können, wird zusätzlich der prinzipielle Ablauf im Organisationsprogramm ORG7 dargestellt.

Das ORG 7 wartet durch den Überwacheraufruf ERWARTE NACHRICHT auf das Eintreffen einer Nachricht. Die Nachricht sagt dem ORG7, dass eine oder mehrere Meldungen in der Eingabewarteschlange enthalten sind. Es wird eine Meldung übernommen und nach der Decodierung die entsprechenden Aufgaben durch ein Vermittlungsprogramm ausgeführt. Nach Durchführung der Aufgabe wird geprüft, ob noch weitere Meldungen vorhanden sind; ggf. werden sie ausgeführt und anschließend wird wieder auf eine Nachricht gewartet.

Der Simulator führt zuerst eine Eingabe durch. Dabei wird eine Meldung über Blattschreiber übernommen. Falls es sich bei der Eingabe um eine Endemeldung handelt, wird die Simulation beendet. Andernfalls wird die Meldung für das ORG7 generiert und mit dem Überwacheraufruf MELD in den Eingabepuffer eingetragen. Um die Bedienung des Programmes zu erleichtern, können die Kennungen der Meldungen mit mnemotechnischen Abkürzungen und die Parameter als Zahlen eingegeben werden.

Nach erfolgter Übergabe der Meldung wird die Meldung vom Eingabepuffer in die Eingabewarteschlange transferiert (Überwacheraufruf TAKT). Dieser Transfer muss durch den Simulator erfolgen, da der Taktgeber für die Alarmsignale abgeschaltet ist. Danach wird das ORG7 mit dem Überwacheraufruf SENDE NACHRICHT aktiviert. Der nächste Aufruf SENDE NACHRICHT ist für die Synchronisation vom Simulator und ORG7 notwendig. Bei diesem Aufruf wird auf die Annahme der Nachricht durch das ORG7 gewartet.

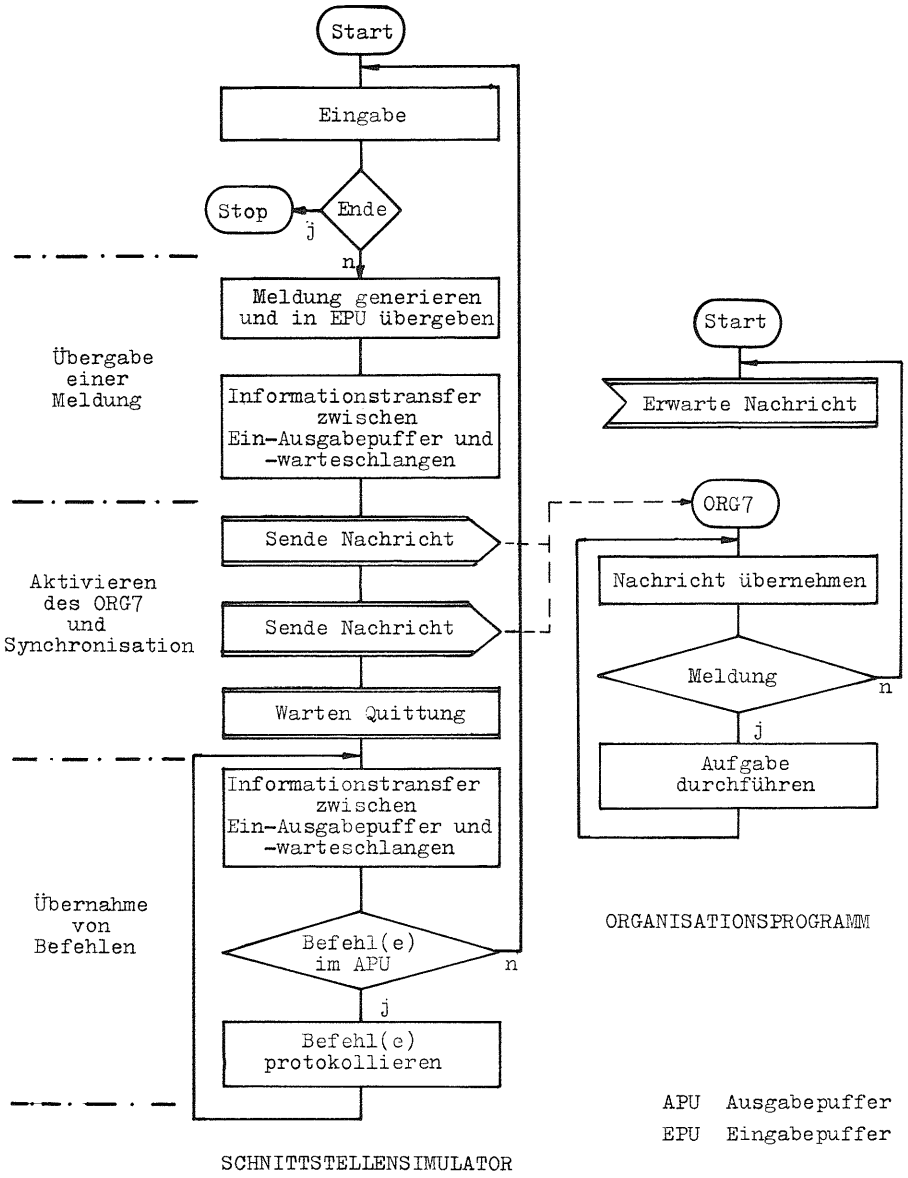


Bild 6.9: Flußdiagramm für den Schnittstellensimulator und das Organisationsprogramm ORG7

Diese Nachricht kann durch das ORG7 jedoch erst angenommen werden, wenn es die zuvor durch die Meldung erhaltenen Aufgaben erledigt hat. (Es wartet dann wieder auf eine Nachricht.) Bei Eintreffen der zweiten Nachricht ist keine Meldung in der Eingabewarteschlange und somit geht das ORG7 wieder in die Ausgangsstellung. (Es wartet wieder auf eine Nachricht.) Damit ist sichergestellt daß, wenn der Simulator weiterläuft, auch die Meldung im ORG7 bearbeitet ist.

Im Simulator werden nun eventuell vom ORG7 ausgegebene Befehle aus der Ausgabewarteschlange ausgelesen. Zu diesem Zweck wird der Transfer von Befehlen aus der Ausgabewarteschlange in den Ausgabepuffer veranlasst (Aufruf TAKT) und die Befehle aus dem Ausgabepuffer übernommen (Aufruf BEFA). Wenn kein Befehl vorhanden ist, wird wieder zur Eingabe verzweigt andernfalls werden die Befehle auf Blattschreiber protokolliert. (Bei einem Transfer durch den Aufruf TAKT kann an je eine dezentrale Steuereinheit (KAS, VAS) nur ein Befehl von der Ausgabewarteschlange in den Ausgabepuffer eingetragen werden. Es können natürlich für verschiedene Steuereinheiten zusammen mehrere Befehle transferiert werden.)Nach erfolgter Protokollierung wird geprüft, ob weitere Befehle in der Ausgabewarteschlange vorhanden sind. Wenn dies der Fall ist, werden sie vom Simulator übernommen und ebenfalls protokolliert. Dieser Vorgang wird solange wiederholt, bis die Ausgabewarteschlange leer ist.

Die Eingabe von Meldungen durch den Simulator kann natürlich auch über Lochkarteneingabe erfolgen. Ebenso kann der Schnelldrucker zur Protokollierung herangezogen werden.

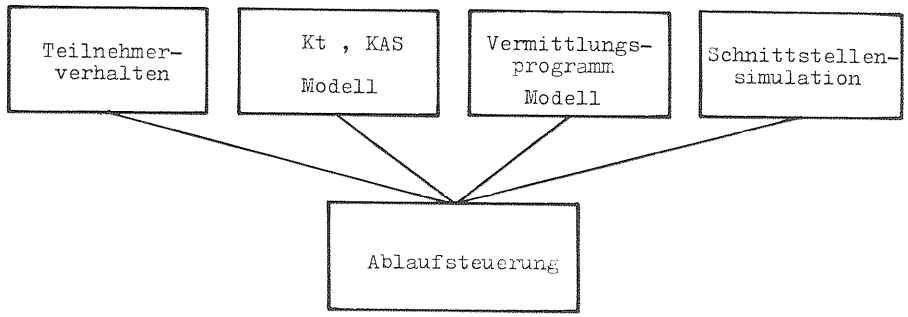
Die Synchronisation von ORG7 und Simulator, die verhindert, dass der Simulator weiterläuft, bevor das ORG7 die Meldung bearbeitet hat, funktioniert unabhängig davon, unter welcher Tasknummer der Simulator zur Ausführung kommt. Wenn der Simulator eine niederere Priorität als das ORG7 hat (größere Tasknummer), könnte sogar auf den zweiten Aufruf SENDE NACHRICHT verzichtet werden, da in diesem Fall der Simulator wegen der Priorität erst weiterläuft, wenn das ORG7 wieder auf eine Nachricht wartet. Wenn die Möglichkeit des ORG7 benutzt wird, die Parameter der aufgerufenen Arbeitsprogramme zu protokollieren (vgl. Abschnitt 4.5.4), ist allerdings der Aufruf wieder notwendig, da während der Ausgabeoperation der Simulator weiterlaufen könnte.

Zum Eintragen einer Meldung in die Eingabewarteschlange sind zwei Überwacheraufrufe notwendig (MELD und TAKT). Mit dem ersten Aufruf wird die Meldung in den Eingabepuffer eingetragen und mit dem zweiten Aufruf vom Eingabepuffer in die Eingabewarteschlange übernommen.

Es ist dadurch auch möglich, die Warteschlangenorganisation in der Taktbearbeitungsroutine (vgl. Abschnitt 3.5.3) mit auszutesten. Um mehrere Einträge in der Warteschlange warten zu lassen, wird man das ORG7 nicht nach jeder eingegebenen Meldung aktivieren sondern z.B. erst nach jeder dritten Meldung.

6.3.3 Umweltsimulation

Das Simulationsprogramm für die Umweltsimulation muß das gesamte Verhalten der Vermittlungsperipherie einschließlich des Teilnehmerverhaltens nachbilden. Zu diesem Zweck enthält es mehrere Bausteine (Bild 6.10).



Kt Konzentrator KAS Konzentratoranschlußschaltung

Bild 6.10: Bausteine des Umweltsimulators

Der Baustein TEILNEHMERVERHALTEN hat die Aufgabe, das Verhalten der Teilnehmer zu simulieren und die notwendigen Ereignisse (z.B. Teilnehmer hebt ab, legt auf, wählt eine Ziffer) zu generieren. Die Teilnehmer können unterschiedliche Verhaltensweisen zeigen, die durch den Zeitpunkt des Auflegens (nach dem Abheben) charakterisiert sind:

- Auflegen ohne Wahl
- Auflegen während der Wahl (Abbrechen der Wahl)
- Auflegen bei Gassen- oder Teilnehmerbesetzt
- Auflegen nach Falschwahl
- Auflegen während des Ruftones
- Auflegen nach Gesprächsende

Zusätzlich ist es möglich, die verschiedenen "Facilities" (z.B. Kurzwahl) aufzurufen.

Bei einer simulierten Wahl müssen die einzelnen Wählziffern in der Reihenfolge generiert werden, wie sie der Rufnummer eines gerufenen Teilnehmers entsprechen.

Damit die zeitliche Reihenfolge der einzelnen Ereignisse gewährleistet ist, muss die Simulation nach der zeittreuen (event by event) Simulationsmethode /44/ erfolgen. Dabei werden die einzelnen Ereignisse mit Hilfe von Zufallszahlen ermittelt und in einen Kalender (simulierte Zeit) eingetragen. Die Abstände zwischen den einzelnen Ereignissen (z.B. zwischen Abheben eines Teilnehmers und der Wahl der ersten Ziffer) sind durch Verteilungsfunktionen vorgegeben. Da gleichzeitig das Verhalten mehrerer Teilnehmer simuliert wird, sind die Ereignisse der Teilnehmer zeitlich ineinander verschachtelt.

Der Baustein Kt, KAS MODELL bildet zusammen die Funktionen des Konzentrators (Kt) und der Konzentratoranschlußschaltung (KAS) nach. Das Modell ist einfacher als das Modell zur logischen Simulation der KAS, da der Informationsaustausch zwischen Kt und KAS entfallen kann. Die Bearbeitungsdauer für die Funktionen in der KAS sind ebenfalls vorgebbar. Im Simulationsprogramm simuliert dieser Baustein zwei KAS und Kt.

Weiterhin wird mit diesem Baustein noch die Funktion einer Vermittlungsstellenanschlußschaltung (VAS) simuliert. Die VAS ist bei Externverkehr kommend und gehend für den Informationsaustausch mit dem Steuerrechner der anderen Vermittlungsstelle notwendig. Da keine zweite Vermittlungsstelle mit einem zweiten Rechner zur Verfügung steht, werden Befehle , die über eine VAS ausgegeben werden, rückgeschleift und als Meldungen von der Ziel-Vermittlungsstelle wieder eingegeben. Dazu wird lediglich im Simulationsprogramm die Nummer der VAS geändert. Diese Art der Simulation ist möglich, da im realen Betrieb ein ausgegebener Befehl für den anderen Steuerrechner auch einer dort eintreffenden Meldung entspricht.

Um die Reaktionen der Systemsoftware (ausgegebener Befehl) überprüfen zu können, ist der Baustein VERMITTLUNGSPROGRAMM MODELL vorhanden. Dieser Baustein hat die Aufgabe für eine Meldung, die an die Systemsoftware übergeben wird, alle möglichen Reaktionen (Befehle) zu ermitteln. Z.B. nach Eingabe einer Ziffer kann der Befehl kommen " Wählzeichenempfänger abschalten", da Wahlende vorliegt.

Zusätzlich kann der Befehl "Belegton anschalten " kommen, wenn es Falschwahl war oder wenn Gassenbesetzt vorliegt. Bei richtiger Wahl kommt der Befehl " Durchschalten zum gerufenen Teilnehmer". Wenn ein Befehl ausgegeben wird, der aufgrund der eingegebenen Meldung nicht ausgegeben werden dürfte, oder wenn ein logisch notwendiger Befehl ausbleibt, so wird eine Fehlermeldung ausgedruckt.

Der Baustein SCHNITTSTELLENSIMULATION hat die Aufgabe, die Meldungen an die Systemsoftware zu übergeben und Befehle von ihr entgegenzunehmen. Er entspricht im wesentlichen dem Schnittstellensimulator nach Abschnitt 6.3.2.

Der Baustein ABLAUFSTEUERUNG verbindet die einzelnen Bausteine untereinander und regelt deren Zusammenwirken. Er bildet das Rahmenprogramm für die anderen Bausteine.

Da die Bearbeitungsdauern für Befehle in der KAS und die zeitlichen Abstände zwischen den Teilnehmerereignissen durch Verteilungsfunktionen vorgebbar sind, können bei den Testläufen im zeitlichen Ereignis-Ablauf absichtlich Maßstabsverzerrungen vorgenommen werden. Dadurch ist es möglich, seltene Ereignisse, die im realen Betrieb erst nach Jahren eintreten und dann möglicherweise zu Fehlern führen würden, häufiger auftreten zu lassen.

Die Abläufe im Simulationsprogramm /45/ sind im Bild 6.11 dargestellt.

Nach einem Vorlauf, in dem die ersten Ereignisse (Abheben) aller simulierten Teilnehmer in den Kalender eingetragen werden, wird das erste Ereignis im Kalender bestimmt. Es wird zu diesem Ereignis das Folgeereignis ermittelt und in den Kalender eingetragen. Das Folgeereignis stellt die nächste Reaktion des Teilnehmers dar (z.B. Wählen einer Ziffer nach dem Ereignis Abheben).

Für das zuvor ermittelte Ereignis, das als Meldung an die Systemsoftware übergeben werden soll, werden die möglichen Reaktionen der Systemsoftware bestimmt und in eine Liste abgespeichert. Danach wird die Meldung generiert und in den Eingabepuffer eingetragen. Jetzt kann der Informationstransfer zwischen dem Eingabepuffer und der Eingabewarteschlange bzw. zwischen der Ausgabewarteschlange und dem Ausgabepuffer veranlaßt werden. (Dabei wird die Meldung in die Eingabewarteschlange übernommen,und es werden Befehle in den Ausgabepuffer eingetragen.)

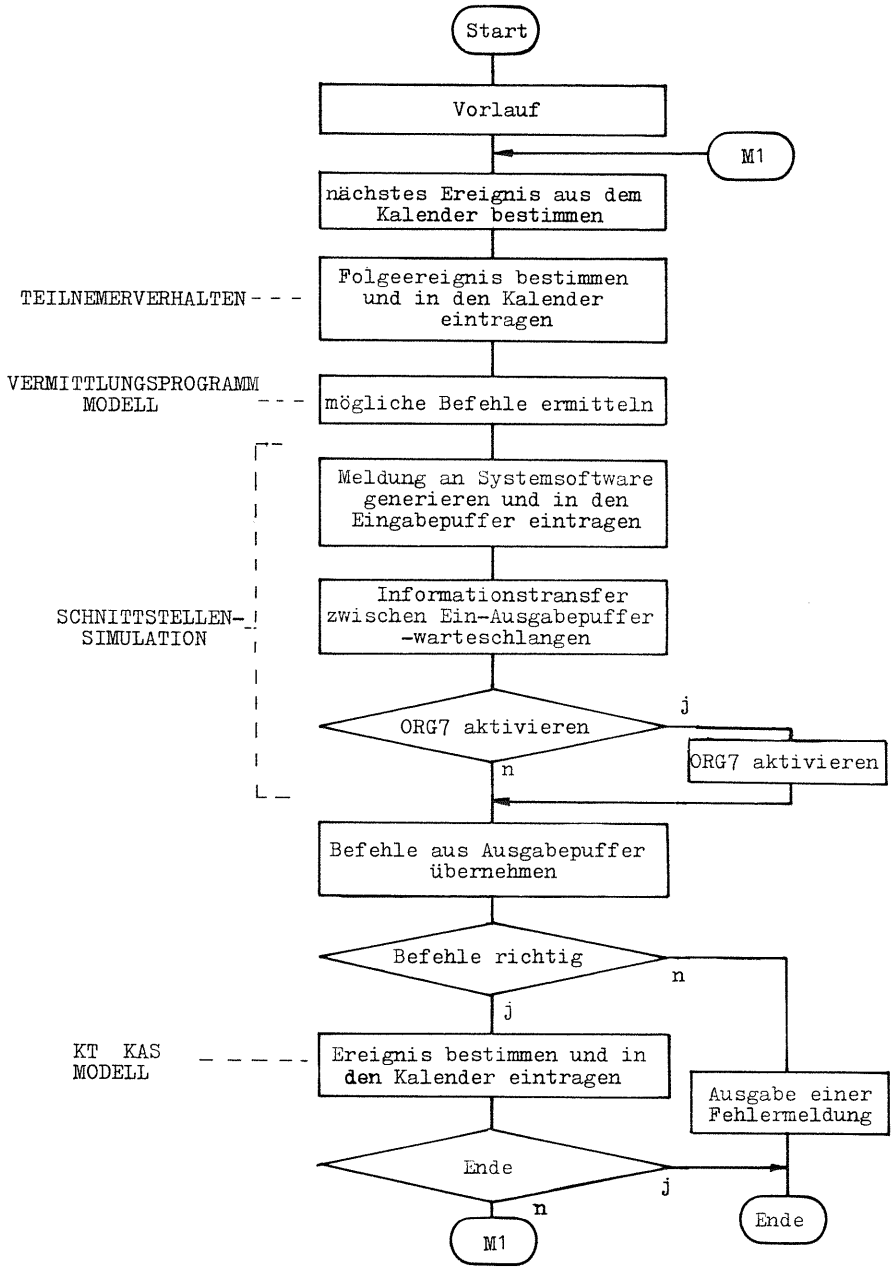


Bild 6.11: Flußdiagramm für den Umweltsimulator

Mit der Abfrage, ob das ORG7 aktiviert werden soll, wird die Möglichkeit geschaffen, erst einige Meldungen in die Eingabewarteschlange einzutragen und dann erst das ORG7 zu aktivieren, um damit das Abarbeiten mehrerer Meldungen hintereinander zu veranlassen. Das ORG7 wird analog zum vorherigen Abschnitt aktiviert.

Die Befehle, die beim Informationstransfer vor der Aktivierung des ORG7 in den Ausgabepuffer eingetragen wurden, stehen noch in dem Ausgabepuffer. Diese werden jetzt übernommen. Wenn sie nicht mit den erwarteten Befehlen, die in eine Liste eingetragen wurden, übereinstimmen, wird eine Fehlermeldung ausgegeben. Bei richtigen Befehlen wird das Ereignis bestimmt, das auf einen Befehl zu folgen hat und in den Kalender eingetragen (z. B. bei dem Befehl "zum gerufenen Teilnehmer durchschalten" das Ereignis "gerufener Teilnehmer hebt ab").Bei mehreren Befehlen aus dem Ausgabepuffer werden entsprechend mehrere Ereignisse eingetragen.

Nach einer Abfrage auf"Simulationsende"wird das nächste Ereignis aus dem Kalender bestimmt und der Ablauf wiederholt sich.

Bei der Übergabe einer Meldung an die Systemsoftware werden gleichzeitig die Befehle übernommen, die gerade in der Ausgabewarteschlange stehen. Nach der Aktivierung des ORG7 werden im Gegensatz zur Schnittstellensimulation in Abschnitt 6.3.2 keine Befehle mehr von der Ausgabewarteschlange in den Ausgabepuffer eingetragen. Dadurch trifft wie im realen Betrieb die Reaktion auf eine Meldung frühestens bei der Übergabe der nächsten Meldung im Simulator ein.

Der Umweltsimulator ermöglicht beliebige Ruffolgen zu erzeugen und beliebig oft unter denselben Voraussetzungen zu wiederholen. Es sind zusätzlich Möglichkeiten vorgesehen, alle Abläufe zu protokollieren, um dadurch Fehler leichter lokalisieren zu können.

6.4 Bestimmung der Bearbeitungszeit von Meldungen durch Vermittlungsprogramme

6.4.1 Allgemeines

Die Leistungsfähigkeit von rechnergesteuerten Vermittlungssystemen wird im wesentlichen durch die Struktur und die Dimensionierung der Koppelnetze und der Leistungsfähigkeit der zentralen und dezentralen Steuerungen charakterisiert. Ein wesentliches Kriterium für die Leistungsfähigkeit des Vermittlungsrechners ist die max. Zahl der Verbindungswünsche, die pro Zeiteinheit bearbeitet werden können. Für diesen charakteristischen Wert darf die mittlere Wartezeit für eine Meldung in der Eingabewarteschlange einen bestimmten Wert nicht überschreiten (Rufverzug) und es dürfen auch keine Meldungen, z.B. wegen Warteschlangenüberlauf, verlorengehen.

Die Abschätzung der Leistungsfähigkeit des Vermittlungsrechners kann durch Simulation oder durch Rechnung erfolgen. In beiden Fällen müssen jedoch die Bearbeitungszeiten für die einzelnen Meldungen (Programmlaufzeiten) bekannt sein. Zu diesem Zweck wird im folgenden eine Meßmethode zur Bestimmung dieser Bearbeitungszeiten beschrieben.

6.4.2 Meßmethode

Die Vermittlungsprogramme kommen unter dem Organisationsprogramm ORG7 zur Ausführung. Das ORG7 übernimmt eine Meldung aus der Eingabewarteschlange und bestimmt aus deren Code das betreffende Vermittlungsprogramm (vgl. Abschnitt 4.5.4). Ein Vermittlungsprogramm kann auch noch Folgeprogramme nach sich ziehen.

Die Laufzeit des Vermittlungsprogrammes, das die Meldung verarbeitet, einschließlich der Folgeprogramme, kann durch Zeitdifferenzmessung bestimmt werden. Zu diesem Zweck bestimmt man mit einem Überwacheraufruf die Uhrzeit, wann die Meldung aus der Eingabewarteschlange übernommen und umcodiert ist und dann, wann das Vermittlungsprogramm und die Folgeprogramme beendet sind (bevor die nächste Meldung aus der Eingabewarteschlange übernommen wird). Der prinzipielle Ablauf ist in Bild 6.12 dargestellt.

Die Bearbeitungszeiten für die Meldungen sollen im Simulator ausgewertet werden. Zu diesem Zweck müssen die Meßwerte dem Simulator übergeben werden. Sie werden dazu als Befehl an eine fiktive KAS (nicht simulierte KAS-Nummer) codiert und werden automatisch wie Befehle an die Vermittlungsperipherie ausgegeben.

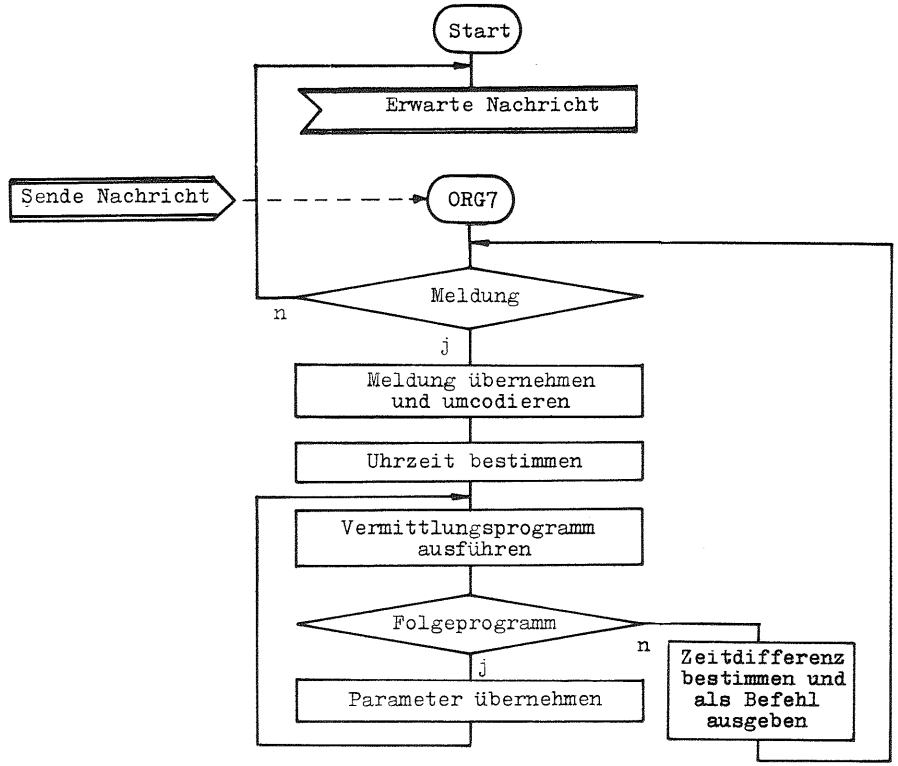


Bild 6.12: Bestimmung der Bearbeitungszeit einer Meldung

Allerdings werden nur Meßzeiten ausgegeben und keine Zuordnung zu welcher Meldung diese Meßzeit gehört. Wenn ein Befehl an die fiktive KAS kommt, erkennt der Simulator daß es sich um eine Meßzeit handelt.

Die Zuordnung zwischen bearbeiteter Meldung und Meßzeit nimmt der Simulator vor. Er trägt die Nummern der Meldungen (mit einer zusätzlichen Kennung, zu welcher Verbindung die Meldungen gehören) in der Reihenfolge, wie sie in die Eingabewarteschlange eingetragen werden, in eine Zuordnungsliste ein.Die Meßzeiten werden vom ORG7 in der Reihenfolge ausgegeben ,wie sie bearbeitet wurden. Diese Reihenfolge entspricht genau der Eingabereihenfolge der Meldungen. Jedesmal, wenn eine Meßzeit im Simulator eintrifft, entspricht genau der nächste Eintrag der Zuordnungsliste der Meldung, die im ORG7 bearbeitet wurde.

Die Zuordnungsliste muss geführt werden, da u. U. nicht jedesmal, wenn eine Meldung vom Simulator eingegeben wird, das ORG7 aktiviert werden muss. D.h., es können mehrere Meldungen in der Eingabewarteschlange stehen, bis die Aktivierung des ORG7 erfolgt.

Für die Auswertung der Meßzeiten wurde in den Simulator ein Auswerteprogramm /46/ eingebaut. Dieses ermittelt für alle Meßgrössen (Bearbeitungsdauern) die Mittelwerte und die Varianz. Die Bearbeitungsdauer für eine Meldung ist nicht konstant, da die Abläufe in den Vermittlungsprogrammen vom Zustand des Vermittlungssystems abhängen; (z.B. müssen im Wegesuchprogramm für das Zentral-Koppelfeld weniger Befehle ausgeführt werden, wenn keine Belegungen im Koppelfeld sind).

Es werden folgende Größen gemessen:

a) Bearbeitungszeit für eine Meldung (individuell für jeden Meldungstyp)

b) Gesamtbearbeitungszeit für alle Meldungen, die zu einer Verbindung gehören. Es sind folgende unterschiedliche Verbindungstypen möglich:

   - Auflegen ohne Wahl
   - Auflegen während der Wahl
   - Auflegen nach vollendeter Wahl
   - Auflegen nach Gesprächsende
     (es wird unterschieden, ob der rufende oder der gerufene Teilnehmer zuerst auflegt)
   - Auflegen, wenn B-Teilnehmer besetzt oder Gassenbesetzt

c) Bearbeitungszeitverteilung für Meldungen

Die Meßergebnisse sind in folgendem Abschnitt dargestellt.

6.4.3 Meßergebnisse

In der Tabelle, Bild 6.13, sind, getrennt für Intern- und Externverkehr, die gesamten Bearbeitungszeiten für alle Meldungen, die zu einer Verbindung gehören, dargestellt.

In der Tabelle, Bild 6.14, sind die Bearbeitungszeiten individuell pro Meldung angegeben und zwar getrennt für Intern- und Externverkehr. Zusätzlich ist bei Externverkehr angegeben, in welcher Vermittlungsstelle (VSt) die Meldung bearbeitet wird. (A-VSt bedeutet VSt des rufenden Teilnehmers, A-Tln, und B-VSt die VSt des gerufenen Teilnehmers, B-Tln.)

Die Varianzen in den beiden Tabellen spiegeln die Tatsache wieder, daß die Bearbeitungszeiten nicht konstant sind. Dies ist darauf zurückzuführen, daß in den Programmen je nach dem Zustand des gesamten Vermittlungssystems unterschiedliche Zweige durchlaufen werden.

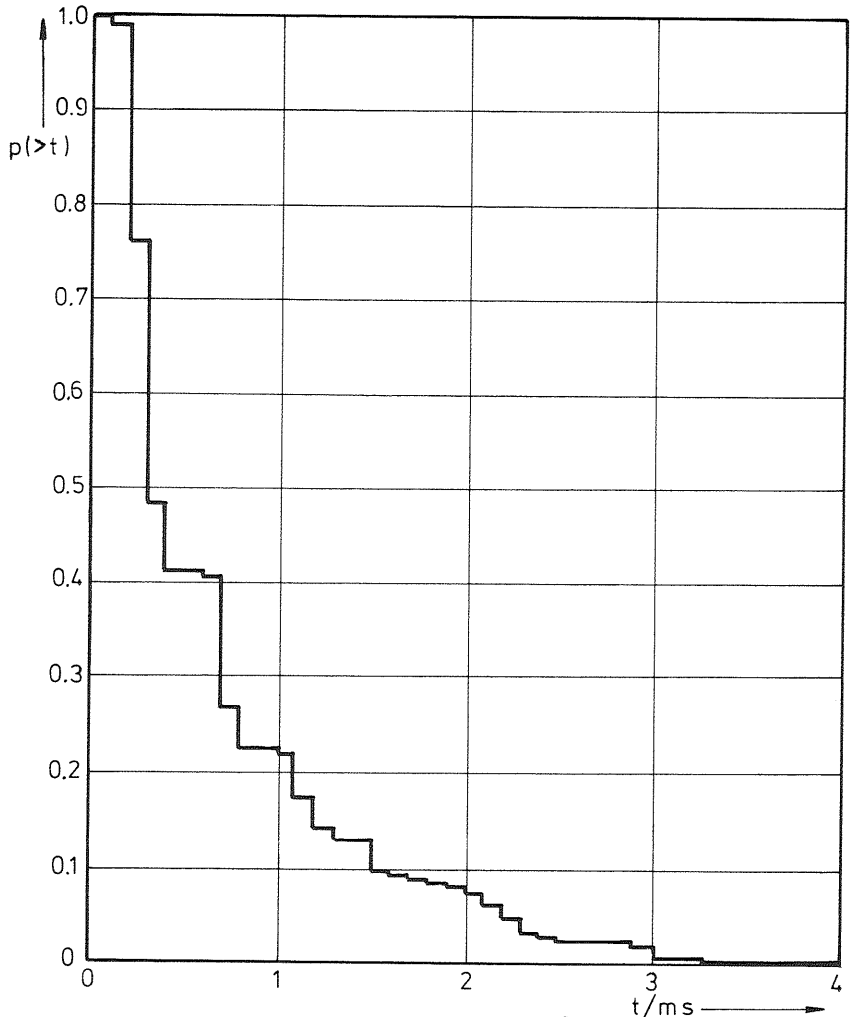
In den Bildern 6.15 bis 6.17 sind die Bearbeitungszeitverteilungen für Meldungen angegeben. Diese Verteilungsfunktionen können für die näherungsweise Berechnung der Leistungsfähigkeit des Vermittlungsrechners herangezogen werden (vgl. Kapitel 7).
Aus Bild 6.16 ist ersichtlich, daß die mittlere Bearbeitungszeit für die Meldungen bei Externverkehr etwas größer ist als bei Internverkehr. Dies rührt daher, daß bei Externverkehr ein Informationsaustausch mit dem Steuerrechner der Ziel-Vermittlungsstelle stattfindet und dadurch mehr Befehle ausgegeben werden.

Verbindung Typ	Internverkehr Mittelwert (ms)	Internverkehr Varianz (ms^2)	Externverkehr Mittelwert (ms)	Externverkehr Varianz (ms^2)
Auflegen ohne Wahl	1.00	0.005	0.99	0.004
Auflegen während der Wahl	1.55	0.075	7.97	4.321
Auflegen nach vollendeter Wahl	3.48	0.494	13.53	2.174
Auflegen nach dem Gespräch				
A-Teilnehmer legt zuerst auf	8.00	0.473	17.14	0.468
B-Teilnehmer legt zuerst auf	7.67	0.360	16.75	0.581
Auflegen wenn B-Teilnehmer besetzt oder Gassenbesetzt	2.99	0.582	11.60	0.975

Bild 6.13: Bearbeitungszeiten für alle Meldungen einer Verbindung

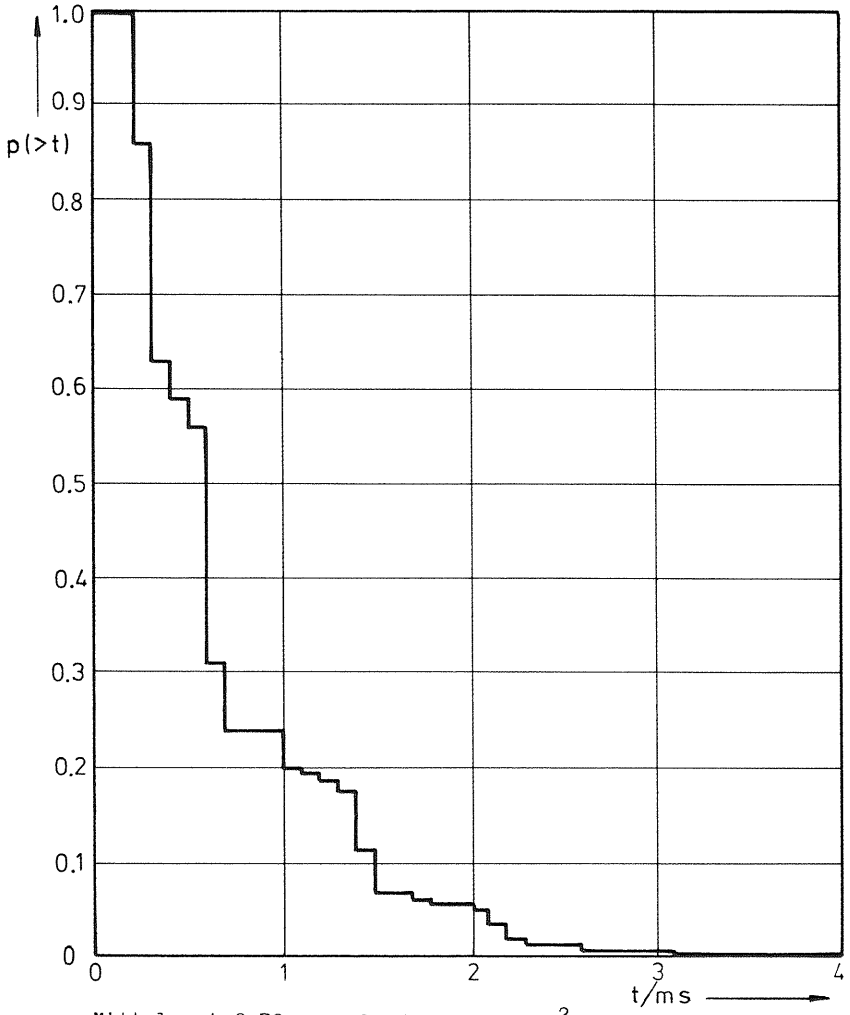
Meldung Typ	Internverkehr		Externverkehr		
	Mittelwert (ms)	Varianz (ms^2)	Mittelwert (ms)	Varianz (ms^2)	VSt
Abheben	0.33	0.002	0.33	0.002	A
Ziffer (1. bis vorletzte)	0.27	0.031	0.87	0.173	A
Ziffer (letzte)	2.00	0.477	0.62	0.001	A
Auflegen (während der Wahl)	0.38	0.035	0.483	0.201	A
Auflegen (nach dem Gespräch) A-Teilnehmer legt zuerst auf	1.79	0.322	1.31	0.002	A
B-Teilnehmer legt zuerst auf	0.66	0.431	0.58	0.190	A
B-Teilnehmer besetzt	0.91	0.014	0.91	0.009	B
Gesprächsbeginn (B-Tln hebt ab)	0.36	0.004	0.55	0.002	B
Kanal nicht verbunden	1.84	0.006	1.14	0.020	B
Ziffer			0.56	0.553	B
Aufbau nicht möglich			1.39	0.388	A
Kanal und Wählzeichenempfänger belegt			0.57	0.002	A
Wahlende			0.64	0.002	A
Gesprächsbeginn			0.36	0.009	A
A-Teilnehmer hat aufgelegt			0.74	0.207	B
Verbindung zum A-Tln getrennt			1.61	0.010	B
Verbindungsabbau			0.95	0.083	A

Bild 6.14: Bearbeitungszeiten für Meldungen



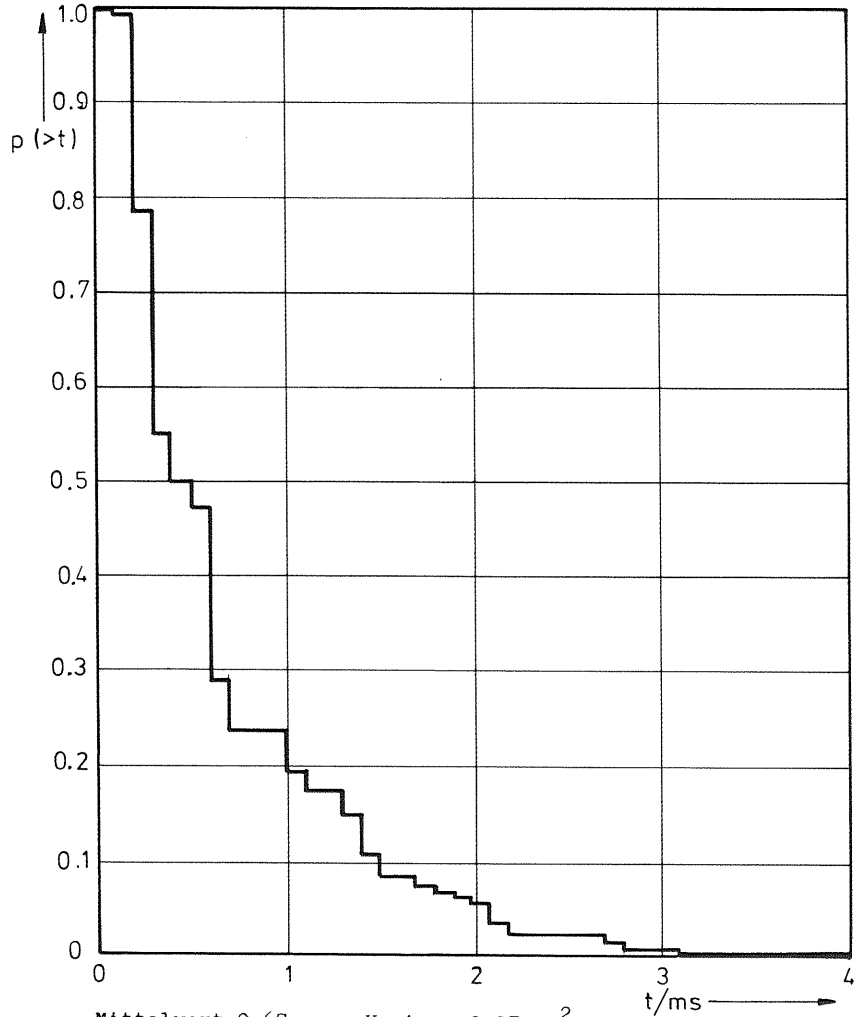
Mittelwert 0.63 ms Varianz 0.37 ms^2

Bild 6.15: Bearbeitungszeitverteilung für Meldungen, nur Internverkehr innerhalb der betrachteten Vermittlungsstelle



Mittelwert 0.70 ms Varianz 0.37 ms^2

Bild 6.16: Bearbeitungszeitverteilung für Meldungen.
nur Externverkehr zu anderen Vermittlungsstellen



Mittelwert 0.67 ms Varianz 0.37 ms^2

Bild 6.17: Bearbeitungszeitverteilung für Meldungen
50 % Intern- und 50 % Externverkehr

7. VERKEHRSTHEORETISCHE UNTERSUCHUNG DES VERMITTLUNGSRECHNERS

7.1 Allgemeines

Der Vermittlungsrechner (VR) stellt die zentrale Steuereinheit eines rechnergesteuerten Vermittlungssystems dar. Die Leistungsfähigkeit des Vermittlungssystems ist bezüglich der Steuerung im wesentlichen durch die Leistungsfähigkeit dieses VR bestimmt. Zur Bestimmung der Leistungsfähigkeit des VR kann die Umweltsimulation (vgl. Abschnitt 6.3.1) eingesetzt werden. Dieses Verfahren ist jedoch sehr aufwendig und kann erst durchgeführt werden, wenn das System bis ins Detail geplant ist. Voruntersuchungen werden deshalb sehr oft an mehr oder weniger vereinfachten Modellen durchgeführt. Diese Modelle bilden die Abläufe im VR und in der Vermittlungsperipherie nach. Je nach Komplexität des Modelles /47/ wird die Analyse durch Simulation oder durch Rechnung vorgenommen.

In den folgenden Abschnitten wird ein einfaches Modell für die Abläufe im VR dargestellt und analytisch behandelt.

7.2 Modell des Vermittlungsrechners

Die Hauptaufgabe des VR besteht darin, die Meldungen, welche von der Vermittlungsperipherie kommen, zu verarbeiten, und die entsprechenden Befehle an die Vermittlungsperipherie zu generieren. Die Eingabe von Meldungen in den VR und die Ausgabe von Befehlen aus dem VR erfolgt zu festen Taktzeitpunkten (vgl. Abschnitt 3.5). Das nachstehende Modell zur Beschreibung der Abläufe im VR (Bild 7.1) umfaßt nur die Eingabe der Meldungen in den VR und deren Bearbeitung. Die Ausgabe von Befehlen aus dem VR erfolgt über einen Augabepuffer, wie auch die Eingabe von Meldungen in den Pufferspeicher, im Zeitintervall t_v (s. Bild 7.2) und beeinflußt den Eingabe- und Verarbeitungsprozeß nicht.

Das Modell (Bild 7.1) umfaßt einen Pufferspeicher mit s Speicherplätzen, welcher der Eingabewarteschlange für Meldungen entspricht (vgl. Abschnitt 3.5.1), und eine Bedienungseinheit, welche die Zentraleinheit des VR repräsentiert. Der Pufferspeicher wird im Modell auch als "Wartespeicher" bezeichnet.

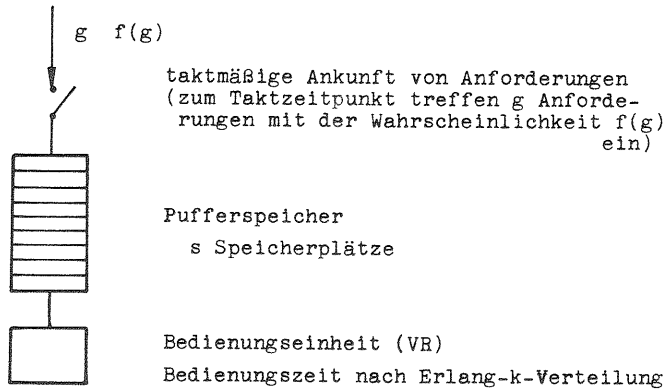


Bild 7.1: Modell des Vermittlungsrechners

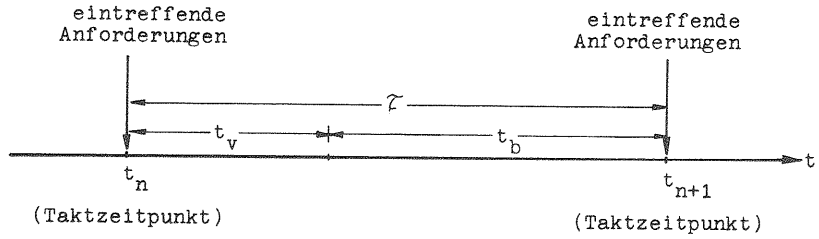


Bild 7.2: Zeitliche Abläufe im Vermittlungsrechner

Zu den äquidistanten Taktzeitpunkten t_n, t_{n+1} ... (Bild 7.2) trifft jeweils eine Gruppe mit einer zufälligen Anzahl von $g \geq 0$ Meldungen, die im folgenden als "Anforderungen" bezeichnet werden, mit der Wahrscheinlichkeit f(g) zur Bearbeitung ein.

Falls zum Taktzeitpunkt die Bedienungseinheit gerade durch eine Anforderung belegt ist, so wird die Bearbeitung dieser Anforderung unterbrochen. Sie wird nicht in den Pufferspeicher zurückgeschrieben, sondern in einem speziellen Register zwischengespeichert.

Danach erfolgt die Übernahme der neuen Anforderungen in den Pufferspeicher. Wenn die Bedienungseinheit zum Taktzeitpunkt gerade frei ist, kann die Übernahme dieser Anforderungen in den Pufferspeicher sofort erfolgen.

Zur Übernahme der neuen Anforderungen in den Pufferspeicher ist eine feste Verwaltungszeit t_v notwendig (Bild 7.2), die im realen System der Ausführungszeit für die Taktbearbeitungsroutine (vgl. Abschnitt 3.5.2.1) entspricht. Die Zeit für die Unterbrechung einer gerade in Bearbeitung befindlichen Anforderung ist vernachlässigbar klein.

Wenn zum Taktzeitpunkt eine Gruppe mit mehr Anforderungen eintrifft als noch Plätze im Pufferspeicher frei sind, so werden nur so viele Anforderungen angenommen als noch Plätze verfügbar sind. Die restlichen Anforderungen werden abgewiesen und nicht bearbeitet. Man bezeichnet die abgewiesenen Anforderungen als "Verlustanforderungen".

Nach der Verwaltungszeit t_v wird die unterbrochene Anforderung weiter bearbeitet bzw. eine der neu eingetroffenen Anforderungen zur Bedienung aus dem Pufferspeicher in die Bedienungseinheit übernommen. Die Anforderungen aus dem Pufferspeicher werden in der Reihenfolge ihres Eintreffens (first in, first out) in die Bedienungseinheit übernommen (FIFO-Abfertigungsdisziplin). Zur Bearbeitung dieser Anforderungen steht die Zeit $t_b = \tau - t_v$ zwischen zwei Taktzeitpunkten zur Verfügung.

Die Gruppenankunft von Anforderungen ergibt sich dadurch, daß zum Taktzeitpunkt von jeder dezentralen Steuereinheit des Vermittlungssystems (KAS,VAS)nur eine Meldung in den VR übernommen werden kann. Die maximale Gruppengröße g_{max} entspricht daher der Gesamtzahl dieser dezentralen Steuereinheiten. Die Gruppengrößen g zu aufeinanderfolgenden Taktzeitpunkten können als voneinander unabhängig betrachtet werden.

Die mittlere Zahl von eintreffenden Anforderungen pro Taktzeitpunkt ist:

$$E\left[g\right] = g_m = \sum_{i=1}^{g_{max}} i \cdot f(i) \qquad (7\text{-}1a)$$

Das Produkt aus der mittleren Anzahl von eintreffenden Anforderungen g_m je Taktzeit τ und der mittleren Bedienungszeit h für eine Anforderung wird als Angebot A bezeichnet.

$$A = \frac{g_m}{\tau} \cdot h = \lambda \cdot h \qquad (7\text{-}1b)$$

Die zufällige Bedienungszeit T_h einer Anforderung durch die Bedienungseinheit entspricht im realen System der Programmlaufzeit des Arbeitsprogrammes (Vermittlungsprogramm), welches die Meldung (Anforderung) bearbeitet.

Die Bedienungszeit T_h stellt eine Zufallsvariable dar, deren realisierte Werte durch eine Erlank-k-Verteilung gut beschrieben werden können (vgl. nachfolgende Abschnitte). Für die Mindestwertverteilung, d.h. die Wahrscheinlichkeit $P\{T_h > t\}$, daß eine Bedienungszeit T_h größer als die Zeit t ist, gilt bei der Erlang-k-Verteilung:

$$P\{T_h > t\} = e^{-\frac{t}{h} \cdot k} \sum_{r=0}^{k-1} \frac{\left(\frac{t}{h} k\right)^r}{r!} \qquad (7\text{-}2)$$

$$\text{Varianz } \sigma^2 = \frac{h^2}{k}$$

h mittlere Bedienungszeit
k Parameter der Erlang-k-Verteilung

Aufeinanderfolgende Realisierungen von Bedienungszeiten sollen voneinander unabhängig sein. Diese Annahme ist gerechtfertigt, da die Anforderungen zu einem Taktzeitpunkt aus verschiedenen dezentralen Steuereinheiten eintreffen und dadurch praktisch voneinander unabhängig sind.

Die Erlank-k-Verteilung erlaubt es, Bedienungszeiten zu beschreiben, deren Verteilungsfunktionen zwischen negativ exponentieller Verteilung (k=1) und konstanter Verteilung ($k \to \infty$) liegen. Dieser Sachverhalt ist im Bild 7.3 dargestellt.

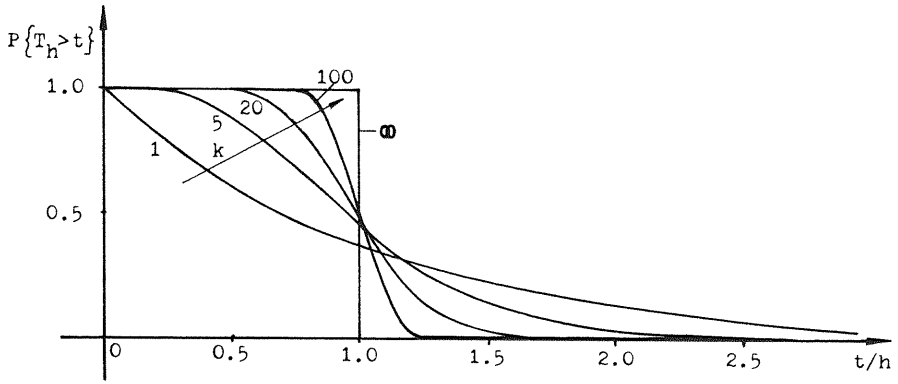


Bild 7.3: Verteilungsfunktion $P\{T_h>t\}$ der Erlang-k-Verteilung

Die Meßkurven für die Bearbeitungszeiten von Meldungen im Abschnitt 6.4 (Bilder 6.15 bis 6.17) stellen diskrete Verteilungsfunktionen (VF) dar. Diese diskreten VF sind i.a. für eine mathematische Untersuchung des Modelles nicht gut geeignet. Man versucht daher, die diskreten VF durch kontinuierliche VF anzunähern. Die kontinuierliche VF sollte, um das Modell einem Berechnungsverfahren zugänglich zu machen, die Markoffeigenschaft besitzen. Die Erlang-k-Verteilung besitzt, wie im folgenden Abschnitt 7.3 gezeigt wird, diese Eigenschaft.

Das Bild 7.4 zeigt Näherungskurven nach der Erlang-k-Verteilung für eine gemessene Verteilung. Die Näherungskurven besitzen denselben Mittelwert wie die gemessenen Kurven. Die Varianz der Näherungskurven kann wegen des ganzzahligen Parameters k der Erlang-k-Verteilung nicht exakt wiedergegeben werden. Deshalb wurde eine Kurve mit größerer Varianz (k=1) und eine Kurve mit kleinerer Varianz (k=2) eingezeichnet. Für die gemessene Bearbeitungszeitverteilung kann die Kurve mit k=2 in guter Näherung für die Untersuchungen verwendet werden.

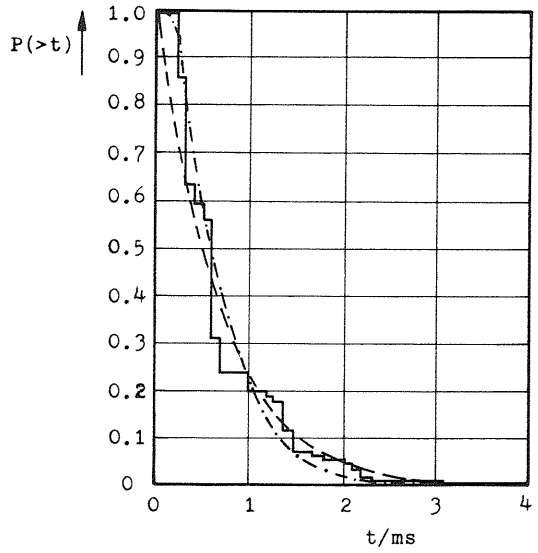


Bild 7.4: Näherung für eine gemessene Bearbeitungszeitverteilungsfunktion

Das Modell nach Bild 7.1 und 7.2 erfordert je nach Größe des Pufferspeichers, Bedienungsdauerverteilung und je nach Vernachlässigung bzw. Berücksichtigung der Verwaltungszeit t_V unterschiedliche analytische Lösungswege. Die nachstehende Übersicht (Bild 7.5) zeigt die verschiedenen Modelleigenschaften und diesbezügliche Veröffentlichungen. Der schraffierte Bereich wird nachstehend analytisch behandelt,und er entspricht den Eigenschaften des hier untersuchten Vermittlungsrechners.

	Bedienungsdauerverteilung					
	konstant		negativ exponentiell		Erlang-k	
	mit t_V	ohne t_V	mit t_V	ohne t_V	mit t_V	ohne t_V
Pufferspeicher unbegrenzt	/48/	/48,50/	/49/	/49,51/	/52/	/52/
Pufferspeicher begrenzt						

▨ in dieser Arbeit behandelt

Bild 7.5: Modelleigenschaften und diesbezügliche Veröffentlichungen

7.3 Herleitung des Gleichungssystems der Zustandswahrscheinlichkeiten

7.3.1 Allgemeines

Das Verkehrsgeschehen im System kann beschrieben werden durch die Anzahl von Anforderungen, die sich im System befinden. Dabei wird der Zustand des Systems durch die Zufallsvariable X(t) gekennzeichnet, die Werte von 0 bis s+1 annehmen kann. Wenn die Zufallsvariable X(t) den Wert 0 hat, ist die Bedienungseinheit frei und der Pufferspeicher leer. (Die Anzahl der Anforderungen im System ist Null.) Der Zustand X(t)=x bedeutet " x Anforderungen sind im System ", wobei eine Anforderung gerade die Bedienungseinheit belegt und x-1 Anforderungen im Pufferspeicher warten. Die Wahrscheinlichkeit für das Auftreten des Zustandes X(t)=x wird mit $P\{X(t)=x\}$ bezeichnet.

Der Zustand $X(t)=x$ des Systems besteht so lange, bis er durch eine Gruppe von g neu eintreffenden Anforderungen in den Zustand $x+g$ oder durch eine fertigwerdende Anforderung in den benachbarten Zustand $x-1$ übergeht. Die zeitliche Aufeinanderfolge der (diskreten) Zufallsvariablen $X(t)$ bildet den Zufallsprozeß der Systemzustände (Zustandsprozeß) $\{X(t),\ t \geq 0\}$.

Betrachtet man den Zufallsprozeß zu einem beliebigen Zeitpunkt t_n und ist der weitere Verlauf des Prozesses nur vom Zustand $X(t_n)$ zur Zeit t_n und nicht von der Vorgeschichte (Zeitpunkte t_i, $i<n$) abhängig, so bezeichnet man den Prozeß als Markoffprozeß /54/.

Die Unabhängigkeit des Prozesses von der Vorgeschichte wird durch folgende Gleichung ausgedrückt:

$$P\left\{X(t_{n+1})=x_{n+1} \,\middle|\, X(t_n)=x_n,\ X(t_{n-1})=x_{n-1},\ \dots,\ X(t_0)=x_0\right\} = P\left\{X(t_{n+1})=x_{n+1} \,\middle|\, X(t_n)=x_n\right\} \qquad (7\text{-}3)$$

$$t_0 < t_1 < \dots < t_n < t_{n+1} \qquad n = 0,1,2\dots$$

Weiterhin gilt für einen Markoffprozeß:

$$P\left\{X(t_{n+1})=x_{n+1}\right\} = \sum_{X_n} P\left\{X(t_n)=x_n\right\} \cdot P\left\{X(t_{n+1})=x_{n+1} \,\middle|\, X(t_n)=x_n\right\} \qquad (7\text{-}4)$$

Wobei $P\left\{X(t_{n+1})=x_{n+1} \,\middle|\, X(t_n)=x_n\right\}$ als bedingte Übergangswahrscheinlichkeit bezeichnet wird. Sie gibt die Wahrscheinlichkeit an, mit der das System vom Zustand $X(t_n)=x_n$ zur Zeit t_n in den Zustand $X(t_{n+1})=x_{n+1}$ zur Zeit t_{n+1} übergeht.

Der zeitliche Verlauf des Zustandsprozesses $\{X(t),\ t \geq 0\}$, d.h. die Anzahl von Anforderungen im System, ist im folgenden Bild dargestellt.

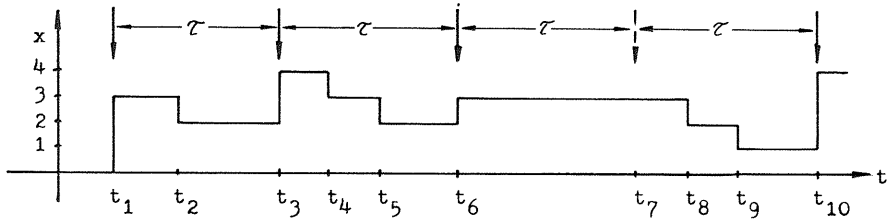


Bild 7.6: Zustandsprozeß $\{X(t_i)=x_i,\ i=1,2,\dots\}$

Die Zeitpunkte t_1, t_3, t_6, t_7 und t_{10} entsprechen den äquidistanten Taktzeitpunkten, zu denen Anforderungen im System eintreffen können. Der Wert der Zufallsvariablen $X(t) = x$ erhöht sich jeweils um die zufällige Anzahl g eintreffender Anforderungen. Zu den Zeitpunkten t_2, t_4, t_5, t_8, t_9 ist jeweils eine Anforderung fertig bearbeitet, d.h. der Wert der Zufallsvariablen $X(t) = x$ wird um 1 kleiner. (Die Verwaltungszeit t_v nach Bild 7.1 werde zunächst nicht betrachtet.)

Betrachtet man nun den Zustandsprozess zu einem Zeitpunkt, an dem Anforderungen im System eintreffen, z.B. zum Zeitpunkt t_3, so hängt der Zustand $X(t_3)$ nicht nur vom Zustand $X(t_2)$, sondern auch vom Zustand $X(t_1)$ ab, denn neue Anforderungen können nur in bestimmten Abständen τ eintreffen. Der Zustandsprozess $X(t_i)$, $i = 1\dots$ stellt also keinen Markoffprozess dar, da die Gleichung (7-3) nicht erfüllt ist.

Betrachtet man nun den Prozeß nur zu den Zeitpunkten $t_1^* = t_1-0$, $t_3^* = t_3-0$ usw., d.h unmittelbar bevor neue Anforderungen im System eintreffen, so zeigt sich, daß der dadurch entstehende Zustandsprozeß $X^*(t_i^*)$, $i=1,3,6,7\dots$ die Markoffeigenschaft nach Gleichung (7-3) besitzt. Z.B. hängt der "neue" Zustand $X^*(t_6^*)$ nur ab vom vorhergehenden Zustand $X^*(t_3^*)$ und von der anzahl neu eingetroffener Anforderungen zur Zeit t_3, sowie von den bis zum Zeitpunkt t_6^* abgefertigten Anforderungen. Diese Anforderungs- und Endeereignisse werden durch eine Übergangswahrscheinlichkeit charakterisiert (s. Abschnitt 7.3.2).

Dieser Zustandsprozeß X^* beschreibt also das Systemverhalten nur zu bestimmten Zeitpunkten. Man bezeichnet diesen Zustandsprozeß auch als eingebettete Markoffkette /54/. Die Zeitpunkte t_i^* werden als Regenerationspunkte bezeichnet.

Um die Zustandswahrscheinlichkeiten für das System nach Gleichung (7-4) berechnen zu können, werden im folgenden Abschnitt zunächst die Übergangswahrscheinlichkeiten bestimmt.

7.3.2 Übergangswahrscheinlichkeiten der Markoffkette

Die Übergangswahrscheinlichkeit $P\{X^*(t^*_{n+1})=x^*_{n+1} \mid X^*(t^*_n)=x^*_n\}$ analog zu Gleichung (7-4) gibt die Wahrscheinlichkeit an, mit des das System vom Zustand $X^*(t^*_n)=x^*_n$ zur Zeit t^*_n in den Zustand $X^*(t^*_{n+1})=x^*_{n+1}$ zur Zeit t^*_{n+1} übergeht. Die Differenz zwischen x^*_{n+1} und x^*_n gibt die Anzahl der neu eingetroffenen Anforderungen minus der im Zeitintervall (t^*_{n+1}, t^*_n) abgefertigten Anforderungen an.

Die Anzahl der Anforderungen, die pro Taktzeitpunkt eintreffen, wird mit g bezeichnet und die Wahrscheinlichkeit für das Auftreten dieser Gruppengröße ist durch die Wahrscheinlichkeit f(g) gegeben.

Die Bedienungszeiten in der Bedienungseinheit (VR) zur Bearbeitung einer Anforderung sollen nach einer Erlang-k-Verteilung verteilt sein (vgl. Abschnitt 7.2). Die Bedienungszeit, die sich aus der Erlang-k-Verteilung ergibt, kann auch interpretiert werden als Summe von k aufeinanderfolgenden "fiktiven" negativ exponentiell verteilten Bedienungszeiten mit dem einheitlichen Mittelwert $h' = \frac{h}{k}$ /55,56/ (s. Bild 7.7).

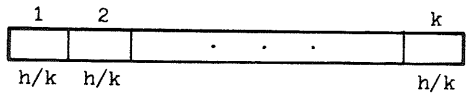


Bild 7.7: Erlang-k-verteilte Bedienungszeit als k negativ exponentiell verteilte Bedienungszeiten

Eine Anforderung verläßt die betrachtete Bedienungseinheit erst dann wenn sie alle k negativ exponentiell verteilten "Teilbedienungszeiten" durchlaufen hat. Aus der Sicht der Bedienungseinheit kann das Durchlaufen von k Teilbedienungszeiten aber auch auch als die serielle Abfertigung von k "Teilanforderungen" mit negativ exponentiell verteilten Bedienungszeiten (Mittelwert h') interpretiert werden /55,56/.

Das Modell nach Bild 7.1 und Bild 7.2 läßt sich deshalb in der Weise analytisch behandeln, daß man negativ exponentiell verteilte Bedienungszeiten für jede Teilanforderung annimmt und zum Taktzeitpunkt nun Gruppen der Größe g·k von Teilanforderungen ankommen.

Zur Beschreibung des Verkehrsgeschehens wird die neue Zufallsvariable $X'(t^*_i)$ eingeführt, welche die Anzahl von Teilanforderungen in System unmittelbar vor den Taktzeitpunkten angibt. Sie kann die Werte von 0 bis (s+1)·k annehmen, da anstelle von einer Anforderung nun k Teilanforderungen im System sind. Der Zusammenhang zwischen den Zufallsvariablen $X^*(t^*_i)$ und $X'(t^*_i)$ ist in Bild 7.8 dargestellt.

Ist z.B. k=5 und befindet sich das System-Modell im Zustand $X'(t^*_i)=7$, so bedeutet dies, daß im realen System zwei Anforderungen enthalten sind. Eine davon hat bereits 3 negativ exponentiell verteilte Teilbedienungszeiten durchlaufen und befindet sich zur Zeit t^*_i in der vorletzten Bedienungsphase. Die zweite Anforderung wartet noch im Pufferspeicher.

Die neue Zufallsvariable $X'(t^*_i)$ beschreibt also ein Wartesystem mit einer Bedienungseinheit, deren Bedienungszeiten negativ exponentiell verteilt sind, und mit einer Warteschlange, die (s+1)·k-1 Speicherplätze für Teilanforderungen besitzt.

Zufallsvariable $X^*(t)$ $X^*(t)=x^*$ Zahl der realen Anforderungen im System	Zufallsvariable $X'(t)$ $X'(t)=x'$ Zahl der Teilanforderungen im System
0	0
1	1 . . . k
2	k+1 . . . 2k
⋮	⋮
s	(s-1)k+1 . . . sk

Bild 7.8: Zusammenhang zwischen den Zufallsvariablen $X^*(t)$ und $X'(t)$

Bei negativ exponentiell verteilten Bedienungszeiten ist zu jedem beliebigen Zeitpunkt t_j die Wahrscheinlichkeit $P(T_R > t)$, daß die Restbedienungszeit T_R länger als die Zeit t dauert unabhängig davon, wie lange die Bedienungszeit vorher schon gedauert hat /57/.

Sind im System zur Zeit $t=t_j$ mehr als z Teilanforderungen enthalten, so ist die Wahrscheinlichkeit r(z) , daß zu einem beliebigen Zeitpunkt t_j+t eine Anzahl von z Teilanforderungen mit negativ exponentiell verteilten Teilbedienungszeiten (Mittelwert h') abgefertigt wurden, durch die Poissonverteilung gegeben:

$$r(z) = e^{-\frac{t}{h'}} \frac{\left(\frac{t}{h'}\right)^z}{z!} \qquad (7\text{-}5)$$

Diese Poissonverteilung gilt dann, wenn während der Zeit t die Bedienungseinheit lückenlos belegt bleibt.

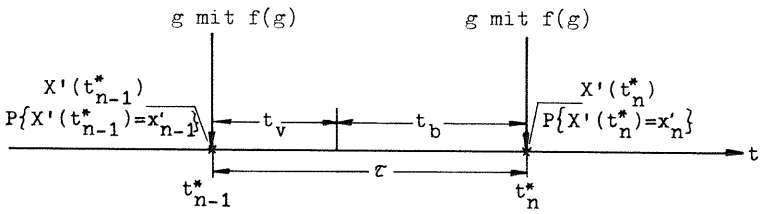
Sind dagegen zur Zeit $t=t_j$ nur genau z Teilanforderungen im System und werden diese innerhalb der Zeit t alle abgearbeitet, so ist die Wahrscheinlichkeit für dieses Ereignis gegeben durch:

$$r_{min}(z) = \sum_{i=z}^{\infty} r(z) = 1 - e^{-\frac{t}{h'}} \sum_{i=0}^{z-1} \frac{\left(\frac{t}{h'}\right)^i}{i!} \qquad (7\text{-}6)$$

$r_{min}(z)$ bedeutet also die Wahrscheinlichkeit, daß "mindestens" diese vorhandenen z Teilanforderungen abgearbeitet werden.

Die Wahrscheinlichkeiten nach den Gleichungen (7-5) und (7-6) sind unabhängig vom betrachteten Taktintervall (t^*_{n-1}, t^*_n).

Das System nach Bild 7.1 und Bild 7.2 werde jetzt nur zu den Zeitpunkten t^*_i unmittelbar vor den Taktzeitpunkten betrachtet, d.h. unmittelbar vor dem möglichen Eintreffen von Anforderungen.



t_v Verwaltungszeit des VR (Overhead)
t_b verfügbare Bearbeitungszeit im Taktintervall

Bild 7.9: Zeitliches Systemverhalten

Zum Taktzeitpunkt trifft jeweils eine Gruppe mit einer zufälligen Anzahl von g Anforderungen ein, die g·k Teilanforderungen entsprechen. Zum Zeitpunkt t^*_{n-1} seien im System x'_{n-1} Teilanforderungen und es mögen weitere g Anforderungen eintreffen für die noch Platz im Pufferspeicher ist. Zum nächsten Taktzeitpunkt t^*_n seien noch genau x'_n Teilanforderungen im System. Damit dieser Fall eintrifft, müssen in der verfügbaren Bearbeitungszeit (s. Bild 7.9) genau $z=x'_n - x'_{n-1} + g\cdot k$ Teilanforderungen bedient werden. Dieser Sachverhalt wird in der Tabelle (Bild 7.10) veranschaulicht.

Zustand $X'(t^*_{n-1})=x'_{n-1}$	Anzahl eintreffender Anforderungen	Anzahl z der bedienten Teilanforderungen	Zustand $X'(t^*_n)=x'_n$
x'_{n-1}	0	$x'_{n-1} - x'_n$	x'_n
	1	$x'_{n-1} - x'_n + k$	
	2	$x'_{n-1} - x'_n + 2k$	
	⋮	⋮	
	g	$x'_{n-1} - x'_n + gk$	
	⋮	⋮	
	g_{max}	$x'_{n-1} - x'_n + g_{max}k$	

Bild 7.10: Zusammenhang zwischen den Zuständen $X'(t^*_{n-1})$ und $X'(t^*_n)$

Wenn die Anzahl der zum Taktzeitpunkt t^*_{n-1} eintreffenden Teilanforderungen g·k größer ist als die Anzahl der noch zur Verfügung stehenden Plätze $N-x'_{n-1}$, so werden nur so viele Teilanforderungen angenommen, wie im realen System vollständige Anforderungen g' Platz hätten. (Die Zufallsvariable $X'(t^*_i)$ kann sich deshalb zu den Taktzeitpunkten nur um ganzzahlige Vielfache von k erhöhen.)

$$g' = \mathrm{Min}\left\{\left[\frac{N-x'_n}{k}\right] , g\right\} \qquad (7\text{-}7)$$

wobei $\mathrm{Min}\left\{\left[\frac{N-x'_n}{k}\right], g\right\} = \begin{cases} \left[\frac{N-x'_n}{k}\right] & \text{für } \left[\frac{N-x'_n}{k}\right] < g \\ g & \text{für } \left[\frac{N-x'_n}{k}\right] \geq g \end{cases}$

$\left[\frac{N-x'_n}{k}\right]$ bedeutet das Ganze von der Division

$N=(s+1)\cdot k$ Maximalwert der Zufallsvariablen $X'(t_i^*)$

In der Tabelle (Bild 7.10) ist die Anzahl der zu bedienenden Teilanforderungen z angegeben, wenn das System vom Zustand $X'(t^*_{n-1})=x'_{n-1}$ im Zeitintervall $(t^*_{n-1}\ ,\ t^*_n)$ in den Zustand $X'(t^*_n)=x'_n$ übergehen soll unter der Voraussetzung, daß alle Anforderungen einer eintreffenden Gruppe vom System angenommen werden können.

Wenn aber neu eintreffende Anforderungen abgewiesen werden müssen weil der im Pufferspeicher noch freie Platz dafür nicht ausreicht, ergibt sich der Wert z (Bild 7.10) allgemein zu:

$$z = x'_{n-1} - x'_n + g'\cdot k \qquad (7\text{-}8)$$

wobei g' nach Gleichung (7-7) gegeben ist.

Da z nur Werte größer oder gleich Null annehmen kann, gilt:

$$x'_n \leq x'_{n-1} + g'\cdot k \qquad (7\text{-}9)$$

Der Zustand des Sytems $X'(t^*_n)=x'_n$ zu einem Zeitpunkt t^*_n kann nur aus einem Zustand $X'(t^*_{n-1})=x'_{n-1}$ entstehen, für welchen die Bedingung (7-9) erfüllt ist.

Wenn nun zum Taktzeitpunkt t^*_{n-1} eine Gruppe von g Anforderungen mit der Wahrscheinlichkeit f(g) eintrifft,und das System vom Zustand $X'(t^*_{n-1})=x'_{n-1}$ in den Zustand $X'(t^*_n)=x'_n$ (mit $x'_n > 0$) übergehen soll, so müssen genau z Teilanforderungen nach Gleichung (7-8) abgefertigt werden. Die Wahrscheinlichkeit r(z) dafür ist nach Gleichung (7-5) gegeben.

Wenn das System während des Taktintervalles leer wird, d.h. in den Zustand $X'(t^*_n)=0$ übergeht, so müssen mindestens z Teilanforderungen abgefertigt werden. Die Wahrscheinlichkeit $r_{min}(z)$ dafür kann nach Gleichung (7-6) angegeben werden.

Die Wahrscheinlichkeit f(g) für das Eintreffen einer Gruppe aus g Anforderungen ist unabhängig von den Wahrscheinlichkeiten r(z) bzw. $r_{min}(z)$. Daher kann die Wahrscheinlichkeit, daß g Anforderungen eintreffen und z Teilanforderungen bedient werden, als Produkt aus den Wahrscheinlichkeiten f(g) und r(z) bzw. $r_{min}(z)$ angegeben werden.

Für die Übergangswahrscheinlichkeiten gilt damit:

$$P\left\{X'(t^*_n)=x'_n \,\middle|\, X'(t^*_{n-1})=x'_{n-1}\right\}\begin{cases} = \sum_{g=0}^{g_{max}} f(g)\cdot r(x'_{n-1}+g'k-x'_n) \\ \qquad \text{für } 0 < x'_n \leq x'_{n-1}+g'k \\ = 0 \quad \text{für } x'_n > x'_{n-1}+g'k \end{cases} \qquad (7\text{-}10a)$$

$$P\left\{X'(t^*_n)=0 \,\middle|\, X'(t^*_{n-1})=x'_{n-1}\right\} = \sum_{g=0}^{g_{max}} f(g)\cdot r_{min}(x'_{n-1}+g'k)$$

wobei: g' nach Gleichung (7-7)
r(z) nach Gleichung (7-5)
$r_{min}(z)$ nach Gleichung (7-6)
} mit $t=t_b$

Da die Wahrscheinlichkeiten f(g) und r(z) bzw. $r_{min}(z)$ für alle t^*_{n-1} gelten sind auch die Übergangswahrscheinlichkeiten zwischen den Taktzeitpunkten t^*_{n-1}, t^*_n zeitinvariant. Es wird deshalb folgende vereinfachte Schreibweise eingeführt:

$$P\left\{X'(t^*_n)=x'_n \,\middle|\, X'(t^*_{n-1})=x'_{n-1}\right\} = \ddot{u}(x'_{n-1}\ ,\ x'_n) \qquad (7\text{-}10b)$$

7.3.3 Zustandswahrscheinlichkeiten

7.3.3.1 Beobachtungszeitpunkt kurz vor den Taktzeitpunkten

Nach Gleichung (7-4) gilt für die Zustandswahrscheinlichkeiten des Systems kurz vor den Taktzeitpunkten:

$$P\left\{X'(t^*_n)=x'_n\right\} = \sum_{x'_{n-1}=0}^{N} P\left\{X'(t^*_{n-1})=x'_{n-1}\right\}\cdot\ddot{u}(x'_{n-1}\ ,\ x'_n) \qquad (7\text{-}11)$$

Für die folgenden Betrachtungen soll das System im stationären Zustand betrachtet werden, d.h. die Zustandswahrscheinlichkeit $P\left\{X'(t^*_n)=x'_n\right\}$ soll unabhängig von der Zeit und vom Anfangszustand zur Zeit t_0 sein. Es soll sein:

$$\lim_{n\to\infty} P\left\{X'(t^*_n)=x'_n\right\} = p'(x') \qquad (7\text{-}12)$$

Wird Gleichung (7-12) auf die Gleichung (7-11) angewendet, so ergibt sich:

$$p'(x') = \sum_{i=0}^{N} p'(i)\cdot\ddot{u}(i,x') \qquad (7\text{-}13a)$$

mit $\ddot{u}(i,x')$ nach Gleichung (7-10)

Die Gleichung (7-13a) stellt ein lineares, homogenes Gleichungssystem für die Zustandswahrscheinlichkeiten $p'(x')$ unseres Systems (vgl. Bild 7.1) dar. Zur vollständigen Bestimmung der Zustandswahrscheinlichkeiten muß noch folgende Normierungsbedingung herangezogen werden.

$$\sum_{i=0}^{N} p'(i) = 1 \qquad (7\text{-}13b)$$

Dieses Gleichungssystem hat den Rang N+1. Für eine rechnergesteuerte Vermittlungsstelle, welche z.B. 50000 Teilnehrer bedient, und welche einen Vermittlungsrechner besitzt, dessen Geschwindigkeit etwa dem Rechner Siemens 306 entspricht, würde $N+1 \approx 500$, wenn eine Erlang-2-Verteilung für die Bearbeitungszeiten angenommen wird.

Die Auflösung des Gleichungssytems wurde nach dem Relaxationsverfahren /58/ durchgeführt.

7.3.3.2 Beobachtungszeitpunkt zu einer beliebigen Zeit t_d nach dem Taktzeitpunkt

Das Systemverhalten soll nun zu einem beliebigen Zeitpunkt t_d nach dem Zeitpunkt t^*_{n-1} betrachtet werden, wobei für t_d die Einschränkung $0 \leq t_d \leq \tau$ gilt (Bild 7.11).

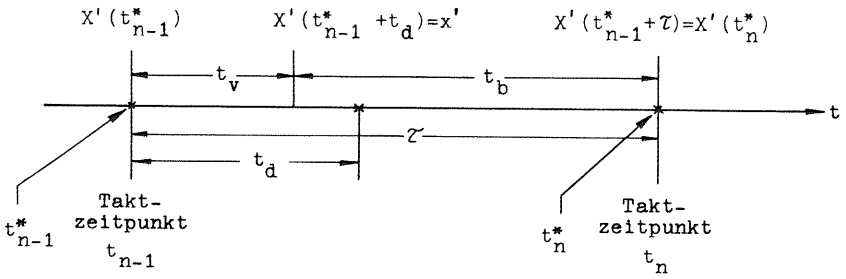


Bild 7.11: Zeitliches Systemverhalten

Der Belegungszustand des Systems sei durch die Zufallsvariable $X'(t^*_{n-1} + t_d)$ beschrieben. Der Zustand $X'(t^*_{n-1} + t_d)$ gibt die Zahl der Teilanforderungen im System zu dem Zeitpunkt t_d nach dem Zeitpunkt t^*_{n-1} (vgl. Bild 7.11) an.

Der Zustand $X'(t^*_{n-1} + t_d)$ für $t_d=0$ entspricht genau dem Zustand kurz vor den Taktzeitpunkten, der im vorigen Abschnitt behandelt wurde. Der Zustand $X'(t^*_{n-1} + \tau)$ ist identisch mit dem Zustand $X'(t^*_n)$.

Im stationären Fall gilt analog zu Gleichung (7-12):

$$\lim_{n\to\infty} P\left\{X'(t^*_n + t_d)\right\} = p'(x', t_d) \qquad (7\text{-}14)$$

Ferner gelten wegen der Stationarität folgende Zusammenhänge:

$$p'(x',0) = p'(x',\tau) = p'(x') \qquad (7\text{-}15)$$

mit $p'(x')$ nach Gleichung (7-13)

Für den Zusammenhang zwischen den stationären Zustandswahrscheinlichkeiten für allgemeines $t_d > 0$ gilt:

$$p'(x', t_d) = \sum_{i=0}^{N} p'(i)\cdot \ddot{u}_{td}(i,x') \qquad (7\text{-}16)$$

Die Übergangswahrscheinlichkeit $\ddot{u}_{td}(i,x')$ stellt jene Wahrscheinlichkeit dar, mit der das System in der Zeit $t_d > 0$ von einem bestimmten Zustand i in den Zustand x' übergeht. Die Wahrscheinlichkeiten $p'(i)$ sind nach Gleichung (7-13) gegeben.

Zur Bestimmung der Übergangswahrscheinlichkeiten $\ddot{u}_{td}(i,x')$ müssen zwei Fälle unterschieden werden:

Fall I $\quad 0 < t_d \leq t_v$

Fall II $\quad t_v < t_d \leq \tau$

Im Fall I ändert sich der Systemzustand nur einmal, nämlich durch die zum Taktzeitpunkt eintreffenden Anforderungen, da während der Verwaltungszeit t_v keine Teilanforderungen bedient werden (vgl. Abschnitt 7.2). Die Übergangswahrscheinlichkeit kann deshalb durch die Wahrscheinlichkeit $f(g)$ für das Auftreten der Gruppengröße g angegeben werden. Dabei ist zu beachten, dass nur "vollständige" Anforderungen vom realen System angenommen werden, d.h. die Größen i und x' (Zahl der Teilanforderungen im System) unterscheiden sich durch die $g\cdot k$ eingetroffenen Teilanforderungen.

Es gilt für die Übergangswahrscheinlichkeit:

$$\ddot{u}_{td}(i,x') = \begin{cases} 0 & \text{für } x' \neq i + g\,k \\ f(g) & \text{für } x' \leq N - k \\ \sum_{g=g_{grenz}}^{g_{max}} f(g) & \text{für } N - k < x' \leq N \end{cases} \qquad g = 0,1,\ldots,g_{max} \qquad (7\text{-}17)$$

$$\text{mit } g_{grenz} = \left[\frac{N - i}{k}\right]$$

Im Fall II können die Übergangswahrscheinlichkeiten aus der Gleichung (7-10a,b) bestimmt werden, wenn $x'_{n-1}=i$ und $x'_n=x'$ gesetzt wird und wenn außerdem in den Ausdrücken für $r(z)$ und $r_{min}(z)$ nach den Gleichungen (7-5) und (7-6) für die Zeit $t=t_d-t_v$ gesetzt wird.

7.4 Charakteristische Verkehrsgrößen

7.4.1 Allgemeines

Die Zustandswahrscheinlichkeiten sind in den meisten Fällen für sich allein zur einfachen Beurteilung des Systemverhaltens nicht ausreichend. Man leitet daher sog. charakteristische Verkehrsgrößen aus den Zustandswahrscheinlichkeiten ab /57/.

Im folgenden werden einige Definitionen angegeben:

Die Verlustwahrscheinlichkeit B ist die Wahrscheinlichkeit, mit der eine eintreffende Anforderung abgewiesen wird, da kein Platz mehr im Wartespeicher frei ist.

Die Verlustrate c_v ist die Zahl von Anforderungen , die im Mittel pro Taktintervall abgewiesen wird.

Die Belastung Y ist definiert als die mittlere Belegungsdauer der Bedienungseinheit pro Taktintervall. Sie wird auch als Verkehrswert bezeichnet.

Die Wartewahrscheinlichkeit W ist die Wahrscheinlichkeit, mit der eine eintreffende Anforderung warten muss, d.h. weder sofort bedient wird noch abgewiesen wird.

Die Wartebelastung Ω(auch als Warteverkehrswert bezeichnet) ist die mittlere Warteschlangenlänge.

Die mittlere Wartezeit w ist die Zeit, die eine beliebige eintreffende Anforderung im Mittel bis zu ihrer Bedienung warten muss.

Die verschiedenen charakteristischen Verkehrsgrößen werden in den folgenden Abschnitten berechnet.

7.4.2 Verlustwahrscheinlichkeit und mittlere Zahl von Verlustanforderungen (Verlustrate c_v).

Wegen des begrenzten Wartespeichers mit s Speicherplätzen können eintreffende Anforderungen abgewiesen werden, wenn alle Speicherplätze belegt sind. Es ist dabei möglich, dass von einer eintreffenden Gruppe mit g Anforderungen ein Teil in den Wartespeicher übernommen wird und der Rest abgewiesen wird.

Betrachtet man den Zustand X' = x' des Systems kurz vor dem Taktzeitpunkt, so können eintreffende Anforderungen nur verlorengehen, wenn die Gruppengröße $g>\left[\frac{N-x'}{k}\right]$ ist, wobei die eckige Klammer das Ganze aus der Division darstellt.

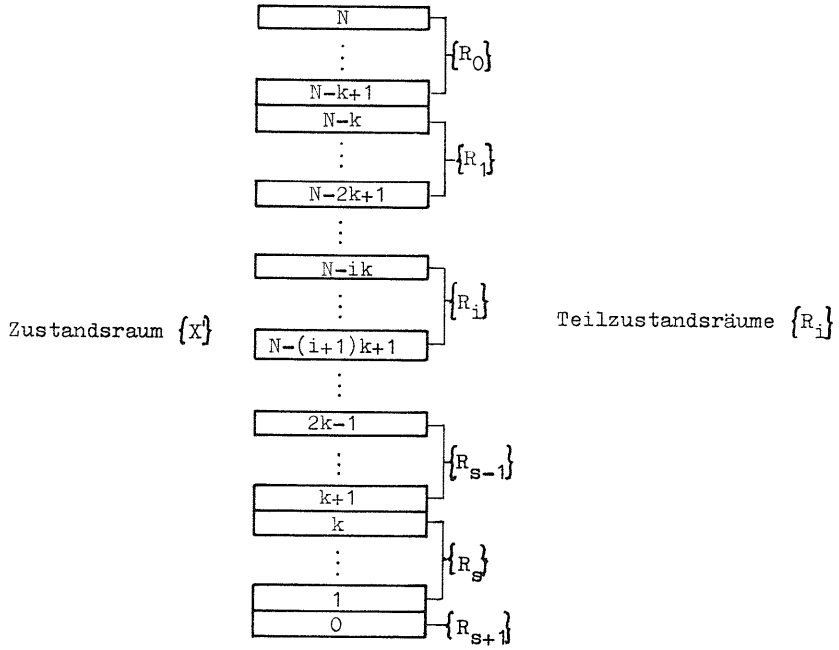


Bild 7.12: Zustandsraum $\{X'\}$ mit den Teilzustandsräumen $\{R_i\}$

Der Zustandsraum $\{X'\}$ soll nun in Teilzustandsräume $\{R_i\}$ unterteilt werden, wobei $\{R_i\} = \{N-((i+1)\cdot k+1),\ldots,N-ik\}$ mit $N = (s+1)\cdot k$ (Bild 7.12). Befindet sich das System in einem der Zustände aus $\{R_0\}$, so werden alle Anforderungen abgewiesen, da eine eintreffende Anforderung den Wert der Zufallsvariablen $X' = x'$ um k erhöhen würde, und dies aber nicht möglich ist. Bei eintreffenden Anforderungen in einem Zustand aus $\{R_1\}$ kann eine Anforderung übernommen werden und der Rest wird abgewiesen. (Der Index i von R_i gibt an, wieviele der eintreffenden Anforderungen vom System angenommen, d.h. nicht abgewiesen werden.)

Besteht unmittelbar vor dem Taktzeitpunkt der Zustandsraum $\{R_i\}$, so ist die mittlere Anzahl, der zum Taktzeitpunkt abgewiesenen Anforderungen v_i:

$$
\begin{aligned}
v_0 &= \sum_{x'\in R_0} p'(x') \sum_{g=1}^{g_{max}} g\cdot f(g) \\
&\vdots \\
v_i &= \sum_{x'\in R_i} p'(x') \sum_{g=i+1}^{g_{max}} (g-i)\cdot f(g) \\
&\vdots \\
v_s &= \sum_{x'\in R_s} p'(x') \sum_{g=s+1}^{g_{max}} (g-s)\cdot f(g) \\
v_{s+1} &= p'(0) \sum_{g=s+2}^{g_{max}} (g-s-1)\cdot f(g)
\end{aligned}
\qquad (7\text{-}18)
$$

Dabei ist natürlich zu beachten, dass nur in jenen Teilzustandsräumen $\{R_i\}$ Verluste auftreten können, für die $i+1 \leq g_{max}$ ist. (Die Anzahl der restlichen freien Speicherplätze ist kleiner als die maximale Gruppengröße.)

Die mittlere Anzahl v von Verlustanforderungen pro Taktzeitpunkt bzw. die Verlustrate c_v ergibt sich aus (7-18) zu:

$$
v = \sum_{i=0}^{s+1} v_i = \sum_{i=0}^{s+1} \sum_{x'=N-(i+1)k+1}^{N-ik} p'(x') \sum_{g=i+1}^{g_{max}} (g-i)\cdot f(g) \qquad (7\text{-}19)
$$

$$
c_v = \frac{v}{\tau} \qquad \text{mit } p'(x') = 0 \quad \text{für } x'<0
$$

Die Verlustwahrscheinlichkeit ist das Verhältnis der mittleren Anzahl von Verlustanforderungen zur Gesamtzahl von eintreffenden Anforderungen.

$$
B = \frac{v}{g_m} \qquad (7\text{-}20)
$$

g_m ist nach Gleichung (7-1a) gegeben und v nach (7-19).

7.4.3 Belastung Y

Die Belastung Y (mittlere Belegung) der Bedienungseinheit (Vermittlungsrechner) setzt sich aus zwei Komponenten Y_v und Y_b zusammen. Die erste Komponente Y_v ist gegeben durch die Verwaltungszeiten t_v, die jeweils zu den Taktzeitpunkten auftreten, und die zweite Komponente Y_b ist gegeben durch den Zeitbedarf für die Abfertigung der Anforderungen (Teilanforderungen). Da alle Anforderungen, die in den Wartespeicher übernommen werden, bedient werden, kann die Belastung durch eine einfache Mittelwertbetrachtung angegeben werden:

$$
\begin{aligned}
Y &= Y_v + Y_b \\
Y_v &= \frac{1}{\tau}\cdot t_v \\
Y_b &= (g_m - v)\cdot\frac{1}{\tau}\cdot h = A - c_v\cdot h = A(1 - B)
\end{aligned}
\qquad (7\text{-}21)
$$

mit g_m nach Gleichung (7-1a) und
v, c_v nach Gleichung (7-19)

Die Größe h stellt die mittlere Bearbeitungszeit für eine Anforderung dar und (g_m - v) ist die mittlere Zahl von Anforderungen pro Taktzeitpunkt, die nicht abgewiesen werden.

7.4.4 Wartewahrscheinlichkeit

Die Wartewahrscheinlichkeit W gibt die Wahrscheinlichkeit an, mit der eine eintreffende Anforderung warten muss, weil die Bedienungseinheit entweder durch eine Anforderung oder durch die Verwaltungszeit t_v belegt ist. Wenn $t_v>0$ ist, ist W = 1.

Die Wartewahrscheinlichkeit W' gibt die Wahrscheinlichkeit an, mit der eine eintreffende Anforderung nach Ablauf der Verwaltungszeit t_v warten muss, weil die Bedienungseinheit noch durch eine andere Anforderung belegt ist. Wenn $t_v = 0$ ist, ist W = W'.

Zur Berechnung der Wartewahrscheinlichkeit wird zuerst die Wahrscheinlichkeit 1-W' berechnet, die angibt, mit welcher Wahrscheinlichkeit eine eintreffende Anforderung nach Ablauf der Verwaltungszeit t_v sofort bearbeitet wird.

Die Möglichkeit, für eine eintreffende Anforderung nach Ablauf der Verwaltungszeit t_v nicht warten zu müssen, ist nur für die erste Anforderung einer Gruppe von Anforderungen gegeben, wenn diese ein leeres System vorfindet.

Die Wahrscheinlichkeit, dass zum Taktzeitpunkt überhaupt eine oder mehrere Anforderungen eintreffen, ist:

$$1 - f(0)$$

Die Wahrscheinlichkeit, dass mit dem Takt eine Gruppe von $g > 0$ Anforderungen eintrifft und ein leeres System vorfindet, ist wegen der Unabhängigkeit der Gruppengröße g vom Systemzustand

$$p'(0) \cdot (1-f(0))$$

Die Wahrscheinlichkeit $p'(0)$ ist nach Gleichung (7-13) gegeben.

Da von einer eintreffenden Gruppe von Anforderungen bei leerem System nur die erste Anforderung der Gruppe nicht warten muß, werden also im Mittel $p'(0)\cdot(1-f(0))\cdot 1$ Anforderungen pro Taktzeitpunkt sofort bedient. Die Wahrscheinlichkeit, daß eine Anforderung nach der Verwaltungszeit t_v sofort bedient wird, ist nun der Quotient aus mittlerer Zahl von sofort bearbeiteten Anforderungen zur Gesamtzahl von eintreffenden Anforderungen.

Damit ist $1 - W' = \frac{p'(0)\cdot(1-f(0)}{g_m}$

Die Wahrscheinlichkeit W' ist nun dadurch gegeben, dass eine eintreffende Anforderung weder sofort bedient wird, noch abgewiesen wird. Es ergibt sich also:

$$W' = 1-(1-W') - B$$

$$W' = 1- \frac{p'(0)\cdot(1-f(0))}{g_m} - B \qquad (7\text{-}22)$$

Die Verlustwahrscheinlichkeit B ist nach Gleichung (7-20) gegeben.

7.4.5 Wartebelastung

Die Wartebelastung Ω stellt die mittlere Zahl belegter Speicherplätze im Wartespeicher dar. Das für die Berechnung zugrunde gelegte Modell besitzt $(s+1)\cdot k-1$ Speicherplätze, während das reale System nur s Speicherplätze besitzt. Die Wartebelastung soll für das reale System mit den s Speicherplätzen bestimmt werden. Der Zusammenhang zwischen den Zuständen $X'(t)$, die das Rechenmodell beschreiben, und den Zuständen $X(t)$, die das reale System beschreiben, ist bereits im Abschnitt 7.3.2 (Bild 7.8) angegeben. Befindet sich das dem Rechenmodell zugrunde gelegte System in einem der Zustände $i\cdot k+1 \leq x' \leq (i+1)\cdot k$, so sind im realen System i Speicherplätze belegt.

Die Zustandswahrscheinlichkeiten für die Zustände x' sind abhängig vom Zeitpunkt, zu dem das System betrachtet wird (vgl. Abschnitt 7.3.2).

Es ist daher auch die Wartebelastung abhängig vom betrachteten Zeitpunkt. Es werden deshalb für die Berechnung der Wartebelastung die Zustandswahrscheinlichkeiten $p'(x', t_d)$ nach Gleichung (7-16) zugrunde gelegt, die den Zustand des Systems zu einem beliebigen Beobachtungszeitpunkt beschreiben.

Es gilt:

$$\Omega(t_d) = \sum_{i=1}^{N} i \cdot \sum_{x'=ik+1}^{(i+1)k} p'(x', t_d) \qquad (7\text{-}23)$$

Für den Fall $t_d = 0$ sind die Wahrscheinlichkeiten $p'(x', 0) = p'(x')$ nach Gleichung (7-13) gegeben.

Wenn die Wartebelastung $\Omega(0)$ kurz vor den Taktzeitpunkten bekannt ist, kann $\Omega(t_d)$ für $0 < t_d \leq t_v$ nach folgender Mittelwertbetrachtung angegeben werden.

Da während der Verwaltungszeit t_v keine Teilanforderungen abgearbeitet werden, muss die mittlere Warteschlangenlänge nach dem Taktzeitpunkt und während der Zeit t_v um die Anzahl von Anforderungen, die zum Taktzeitpunkt eintreffen und nicht abgewiesen werden, größer sein, als die mittlere Warteschlangenlänge vor dem Taktzeitpunkt.

$$\Omega(t_d) = \Omega(0) + g_m - v \qquad \text{für } 0 < t_d \leq t_v \qquad (7\text{-}24)$$

Die mittlere Anzahl pro Taktzeitpunkt eintreffender Anforderungen g_m ist nach Gleichung (7-1a) gegeben und die mittlere Zahl abgewiesener Anforderungen v nach Gleichung (7-19).

7.4.6 Mittlere Wartezeit

Die mittlere Wartezeit w ist die Zeit, die eine eintreffende Anforderung warten muss, bis sie bedient wird. Diese Wartezeit w setzt sich aus zwei Anteilen w_1 und w_2 zusammen. Der erste Anteil w_1 ergibt sich durch die Verwaltungszeiten t_v und der zweite Anteil w_2 durch die zur Bedienung der bereits wartenden Teilanforderungen notwendigen Bedienungszeiten.

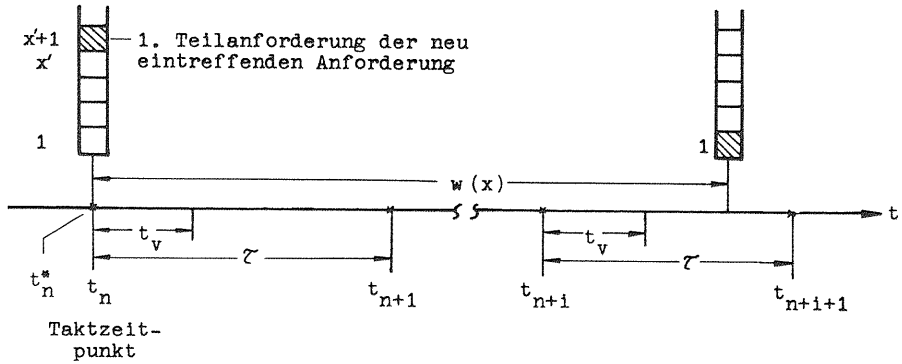


Bild 7.13: Wartezeit w (x)

Trifft eine Anforderung das System im Zustand X'=x' an, so bedeutet dies, dass x' Teilanforderungen im System sind (Bild 7.13). Die k Teilanforderungen, die zu einer neu eintreffenden Anforderung gehören, kommen auf die Warteplätze x'+1 bis x'+k.

Von den x' Teilanforderungen, die im System sind, sind x'-1 im Wartespeicher und eine belegt gerade die Bedienungseinheit. Die Bedienung dieser Teilanforderung durch die Bedienungseinheit wird zum Taktzeitpunkt für die Dauer der Verwaltungszeit t_v unterbrochen. Die Restbedienungszeit dieser Teilanforderung, d.h. die Zeit, die noch notwendig ist, um die unterbrochene Teilanforderung fertig zu bedienen, ist nach /57/ wegen der negativ exponentiell verteilten Bedienungsdauern gleich der mittleren Bedienungszeit h' einer Teilanforderung. Damit ergibt sich die mittlere Wartezeit $w_2(x)$ einer Anforderung, die x' Teilanforderungen im System antrifft, zu:

$$w_2(x') = x' \cdot h' \qquad (7\text{-}25)$$

Die Wahrscheinlichkeit, dass die x' negativ exponentiell verteilten Teilbedienungszeiten mit Mittelwert h', die von den zu bearbeitenden x' Teilanforderungen zusammen benötigt werden (Gesamtzeit T), kürzer als die Zeit t dauern ist nach /55/:

$$P\left\{ T \leq t \right\} = 1 - e^{-\frac{t}{h'}} \cdot \sum_{\gamma=0}^{x'-1} \frac{\left(\frac{t}{h'}\right)^{\gamma}}{\gamma!} \qquad (7\text{-}26)$$

Dabei stellt t die reine Bedienungszeit dar, die in den Bearbeitungsphasen $\tau - t_v = t_b$ der Bedienungseinheit, ausgeführt wird.

Wenn die Gesamtbedienungszeit T im Intervall($i \cdot t_b$, $(i+1) \cdot t_b$] liegt, bedeutet dies, dass im Verlauf von T insgesamt i+1 Verwaltungszeiten t_v anfallen, bis alle x' Teilanforderungen fertig bedient sind.

Die Wahrscheinlichkeit q_i, dass die Gesamtbedienungszeit T in das Intervall ($i \cdot t_b$, $(i+1) \cdot t_b$] fällt, ist:

$$q_i = P\left\{ it_b < T \leq (i+1)t_b \right\} = P\left\{ T \leq (i+1)t_b \right\} - P\left\{ T \leq it_b \right\} \qquad (7\text{-}27a)$$

mit Gleichung (7-26) ergibt sich:

$$q_i = e^{-\frac{it_b}{h'}} \sum_{\gamma=0}^{x'-1} \frac{\left(\frac{it_b}{h'}\right)^{\gamma}}{\gamma!} - e^{-\frac{(i+1)t_b}{h'}} \sum_{\gamma=0}^{x'-1} \frac{\left(\frac{(i+1)t_b}{h'}\right)^{\gamma}}{\gamma!} \qquad (7\text{-}27b)$$

Diese mittlere Wartezeit $w_1(x')$ der betrachteten Anforderung, die aus den Verwaltungszeiten resultiert, ergibt sich zu:

$$w_1(x') = \sum_{i=0}^{\infty} (i+1)t_v \cdot q_i \qquad (7\text{-}27c)$$

Wird Gleichung (7-27b) in Gleichung (7-27c) eingesetzt, so ergibt sich nach einigen Umformungen:

$$w_1(x') = t_v \sum_{i=0}^{\infty} e^{-\frac{it_b}{h'}} \sum_{\gamma=0}^{x'-1} \frac{\left(\frac{it_b}{h'}\right)^{\gamma}}{\gamma!} \qquad (7\text{-}27d)$$

Die mittlere Wartezeit $w(x')$ einer Anforderung, die bei ihrer Ankunft x' Teilanforderungen im System antrifft, ist gegeben durch:

$$w(x') = w_1(x') + w_2(x') \qquad (7\text{-}28)$$

wobei $w_1(x')$ nach Gleichung (7-27d) und $w_2(x')$ nach Gleichung (7-25) einzusetzen sind.

Die Gleichung (7-28) gilt für die erste Anforderung einer eintreffenden Gruppe von Anforderungen. Trifft eine Gruppe von g Anforderungen ein, so hat die erste Anforderung der Gruppe die Wartezeit $w(x')$, die zweite Anforderung hat die Wartezeit $w(x'+k)$, da durch die erste Anforderung k Teilanforderungen ins System kommen, die vor der zweiten Anforderung bedient werden müssen, usw.

Die mittlere Wartezeit $w^*(g)$ für eine Gruppe von Anforderungen der Größe g ist:

$$w^*(g) = \underbrace{\sum_{x'=0}^{N-k} w(x')\cdot p'(x')}_{\text{1. Anforderung}} + \underbrace{\sum_{x'=0}^{N-2k} w(x'+k)\cdot p'(x')}_{\text{2. Anforderung}} + \ldots + \underbrace{\sum_{x'=0}^{N-gk} w(x'+(g-1)k)\cdot p'(x')}_{g.\ \text{Anforderung}} \qquad (7\text{-}29)$$

$$w^*(g) = \sum_{\gamma=1}^{g} \sum_{x'=0}^{N-k} w(x'+(\gamma-1)k)\cdot p'(x')$$

Die oberen Summengrenzen sind so festgelegt, dass die betrachtete Anforderung der Gruppe gerade noch Platz im Wartespeicher findet. Die Größe $w^*(g)$ gibt also die mittlere Wartezeit aller Anforderungen einer Gruppe an, die nicht abgewiesen werden. Die Wahrscheinlichkeit für das Eintreffen von einer Gruppe mit g Anforderungen ist nach Voraussetzung $f(g)$.

Die mittlere Wartezeit <u>aller</u> dem System angebotenen Anforderungen (einschließlich der Verlustanforderungen) ergibt sich zu:

$$w = \frac{1}{g_m} \sum_{g=1}^{g_{max}} w^*(g)\cdot f(g) \qquad (7\text{-}30)$$

g_m ist nach Gleichung (7-1) und $w(g)$ nach Gleichung (7-29) einzusetzen.

7.5 Numerische Ergebnisse für die charakteristischen Verkehrsgrößen

Im folgenden werden einige Ergebnisse für charakteristische Verkehrsgrößen gezeigt. Diese Verkehrsgrößen werden durch folgende Parameter beeinflußt:

- Ankunftsprozess, charakterisiert durch die maximale Gruppengröße g_m und die Wahrscheinlichkeiten für das Auftreten der einzelnen Gruppengrößen $f(g)$.
- Abtastprozess, charakterisiert durch die Taktperiode τ des Abtasttaktes und durch die Verwaltungszeit t_v.
- Speicherplatzkapazität des Pufferspeichers s .
- Bedienungsprozess, charakterisiert durch die mittlere Bedienungszeit h und den Parameter k der Erlang-k-Verteilung.

In den folgenden Beispielen wurde der Einfachheit halber für die Wahrscheinlichkeiten der einzelnen Gruppengrößen eine Poissonverteilung mit Mittelwert g_m zugrundegelegt.

Diese hat die Form:

$$f(g) = e^{-g_m} \frac{(g_m)^g}{g!} \qquad (7\text{-}31)$$

weiterhin gilt: $\frac{Var(g)}{g_m} = 1$

Da die Poissonverteilung Werte für Gruppengrößen von Null bis unendlich liefert, wurde die maximale Gruppengröße g_{max} so festgelegt, dass die Wahrscheinlichkeiten $f(g)$ für $g > g_{max}$ im Rahmen der Rechengenauigkeit vernachlässigt werden können. (Die Poissonverteilung beschreibt den Ankunftsprozess richtig, wenn die einzelnen Anforderungen, die jeweils zu den Taktzeitpunkten übernommen werden, negativ exponentiell verteilte Ankunftsabstände haben.)

Im Bild 7.14 wird der Einfluß des Parameters k der Erlank-k-Verteilung auf die normierte mittlere Wartezeit $\frac{w}{h}$ aufgezeigt. Die mittlere Bedienungszeit für dieses spezielle Beispiel ist $h = \tau$.

Im dargestellten Bereich der Kurven (Bild 7.14) ist wegen s = 100 der Verlust B vernachlässigbar klein ($< 10^{-8}$). Deshalb stellt die Abszisse - hier $A = g_m \cdot \frac{h}{\tau} = g_m$ - wegen $B \to 0$ gleichzeitig auch die Belastung $Y = A(1-B) \approx A$ der Bedienungseinheit dar.

Für ein konstantes Angebot A ist deutlich erkennbar, daß mit wachsendem Parameter k der Erlang-k-Verteilung, d.h. mit abnehmender Varianz der Bedienungsdauerverteilung, die normierte Wartezeit $\frac{w}{h}$ stark abnimmt. Die Grenzkurve für $k \to \infty$ (konstante Bedienungszeit) ist nach /50/ durch folgende Gleichung gegeben:

$$\frac{w}{h} = \frac{1}{2}\left(\frac{Var(g)}{g_m} \cdot \frac{1}{1 - g_m} - 1\right) = \frac{1}{2} \cdot \frac{g_m}{1 - g_m} \qquad (7\text{-}32)$$

(da für die Poissonverteilung $\frac{Var(g)}{g_m} = 1$ ist)

Bild 7.15 zeigt den Einfluß der mittleren Gruppengröße g_m auf die normierte mittlere Wartezeit.

Das auf der Abszisse aufgetragene Angebot $A = \frac{g_m}{\tau} \cdot h$ wächst ausschließlich durch eine Vergrößerung der mittleren Bedienungsdauer h. Dies entspricht der Verwendung langsamerer Zentraleinheiten. (Wegen s=100 stellt die Abszisse wie im Bild 7.14 gleichzeitig auch hier die Rechnerbelastung Y dar.)

Für die Verteilungsfunktion der Bedienungsdauern wurde eine Erlang-2-Verteilung gewählt; die Verwaltungszeit t_v wurde für dieses Beispiel vernachlässigt.

Für einen konstanten Wert $A \approx Y$, d.h. für eine bestimmte mittlere Belegungsdauer steigt die mittlere Wartezeit $\frac{w}{h}$ etwa proportional mit der Gruppengröße g_m an. Die Ursache liegt darin, dass - auch bei minimaler Rechnerbelastung - höchstens die erste Anforderung einer Gruppe wartezeitlos bedient werden kann. Alle anderen Anforderungen einer Gruppe müssen warten, bis die "Vordermänner" ihrer Gruppe bedient sind. Je größer also der Wert g_m desto größer wird auch die mittlere Wartezeit für die Anforderungen einer Gruppe.

Wenn die mittlere Bedienungszeit h (und damit auch das Angebot A) gegen Null strebt, so strebt in diesem Fall die normierte mittlere Wartezeit $\frac{w}{h}$ nach / 53/ gegen den Grenzwert:

$$\lim_{h \to 0} \frac{w}{h} = \frac{1}{2}\left(\frac{Var(g)}{g_m} + g_m - 1\right) = \frac{g_m}{2} \qquad (7\text{-}33)$$

(da für die Poissonverteilung $\frac{Var(g)}{g_m} = 1$ ist)

Wegen der Beziehung $A = \frac{g_m}{\tau} \cdot h$ stellt für ein konstantes Angebot A (d.h. auch für eine konstante mittlere Bedienungsdauer h) eine Vergrößerung der Gruppengröße g_m auch eine Vergrößerung der Taktintervalldauer τ dar. Die Kurven stellen daher auch den Einfluß der Taktintervalldauer auf die normierte mittlere Wartezeit dar.

Die untere Grenzkurve gilt für eine "unendlich schnelle" Taktfrequenz, d.h. für eine kontinuierliche Übernahme von Anforderungen aus der Peripherie, deren Ankunftsabstände negativ exponentiell verteilt sind. (Dieses Wartesystem wird auch als Wartesystem $M|E_2|1$ bezeichnet.)

Bei der kontinuierlichen Übernahme der Anforderungen entstehen offenbar die kleinsten Wartezeiten. Eine vorteilhafte Auswirkung einer getakteten Übernahme von Anforderungen ergibt sich jedoch, wenn man den Einfluß der Verwaltungszeit auf die mittlere Wartezeit berücksichtigt.

Das Bild 7.16 zeigt die Auswirkungen der Verwaltungszeit t_v auf die normierte mittlere Wartezeit $\frac{w}{h}$. Die effektive Rechnerbelastung Y_b (Belastung durch Anforderungen) ist konstant zu 0,5 Erl angenommen. Für die Verteilungsfunktion der Bedienungsdauern wurde wie im Bild 7.15 eine Erlang-2-Verteilung gewählt. Die Abszisse gibt die auf die mittlere Bedienungsdauer normierte Taktintervalldauer $\frac{\tau}{h}$ an. Parameter der Kurven ist die auf die mittlere Bedienungsdauer normierte Verwaltungszeit $\frac{t_v}{h}$.

Durch die Verwaltungszeit t_v wird die zusätzliche Rechnerbelastung $Y_v = \frac{t_v}{\tau}$ verursacht. Mit abnehmender Taktintervalldauer $\frac{\tau}{h}$, d.h. mit zunehmender Abtastfrequenz, steigt die Rechnerbelastung Y_v. Daher steigt auch die mittlere Wartezeit. Die normierte mittlere Wartezeit strebt gegen eine Asymptote. Diese Asymptote liegt bei der Taktintervalldauer τ_a, bei der die Gesamtbelastung $Y = Y_b + Y_v$ den Wert 1 erreicht. Dies ist die minimale zulässige Taktintervalldauer. Sie ist gegeben durch:

$$Y = Y_b + Y_v = 1$$

mit $Y_b = 0{,}5$ und $Y_v = \frac{t_v}{\tau}$ ergibt sich:

$$\tau_a = 2t_v \quad \text{oder} \quad \frac{\tau_a}{h} = 2\,\frac{t_v}{h} \qquad (7\text{-}34)$$

Mit zunehmender Taktintervalldauer wächst die mittlere Gruppengröße g_m. Für das betrachtete Beispiel gilt:

$$Y_b = 0{,}5 = g_m \frac{h}{\tau}$$
$$g_m = 0{,}5 \frac{\tau}{h} \qquad (7\text{-}35)$$

Die mittlere Wartezeit enthält einen Anteil w_g, der durch das Abfertigen der "Vordermänner" in der eigenen Gruppe gegeben ist. Dieser Anteil ist nach /53/, wenn die Gruppengrößen nach einer Poissonverteilung verteilt sind, $w_g = 0{,}5 \cdot g_m \cdot h$. Wegen der Beziehung (7-35) gilt:

$$\frac{w_g}{h} = 0{,}5 \cdot g_m = 0{,}25 \frac{\tau}{h} \qquad (7\text{-}36)$$

Für sehr große Taktintervalldauern überwiegt dieser Anteil an der Wartezeit und die Kurven (Bild 7.16) gehen in Geraden mit der Steigung 0,25 über. Die Gerade $\frac{w_g}{h}$ gibt die Grenzkurve für diese Verwaltungszeit $\frac{t_v}{h} = 0$ an. Wenn die Verwaltungszeit größer 0 ist, ist die entstehende Grenzkurve um $\frac{t_v}{h}$ zur eingezeichneten Kurve parallel verschoben.

Die Kurven (Bild 7.16) zeigen, dass bei einer bestimmten Verwaltungszeit, eine optimale Taktintervalldauer existiert, bei der die mittlere Wartezeit ein Minimum wird.

Bild 7.17 zeigt die Auswirkungen der Verwaltungszeit t_v auf die mittlere Wartezeit einer Anforderung. Für dieses Beispiel sind die Bedienungszeiten als negativ exponentiell verteilt angenommen und der Mittelwert h ist gleich der Taktintervalldauer ($h = \tau$). Auf der Abszisse ist die effektive Rechnerbelastung Y_b aufgetragen. Parameter der Kurven ist die Belastung Y_v durch die Verwaltungszeit.

Die Wartezeitkurven besitzen Asymtoten bei der effektiven Belastung Y_{b_a}, für welche gilt:

$$Y_v + Y_{b_a} = 1$$
$$Y_{b_a} = 1 - Y_v \qquad (7\text{-}37)$$

Wenn die effektive Belastung Y_b gegen Null strebt, so ist die normierte mittlere Wartezeit $\frac{w}{h} = Y_v$.

Für die Wartewahrscheinlichkeit W und die Verlustwahrscheinlichkeit B ist in Bild 7.18 ein Beispiel gegeben. Damit Verluste auftreten, muss die Zahl der Warteplätze klein gewählt werden (s = 10 Warteplätze). Die Bedienungszeiten sind nach einer Erlang-2-Verteilung angenommen. Die mittlere Gruppengröße g_m wurde konstant zu $g_m = 0{,}5$ gewählt. Der Grenzwert für die Wartewahrscheinlichkeit W, wenn das Angebot gegen Null strebt, ist:

$$\lim_{A \to 0} W = \frac{1}{g_m} \sum_{g=2}^{g_{max}} (g - 1) \cdot f(g) \qquad (7\text{-}38)$$

Der Grenzwert ergibt sich dadurch, dass Anforderungen nur wegen ihrer Vordermänner in derselben Gruppe warten müssen.

Die beiden Kurven in Bild 7.18 zeigen denselben prinzipiellen Verlauf, wie er auch bei Warte-Verlustsystemen ohne taktmäßige Übernahme von Anforderungen auftritt.

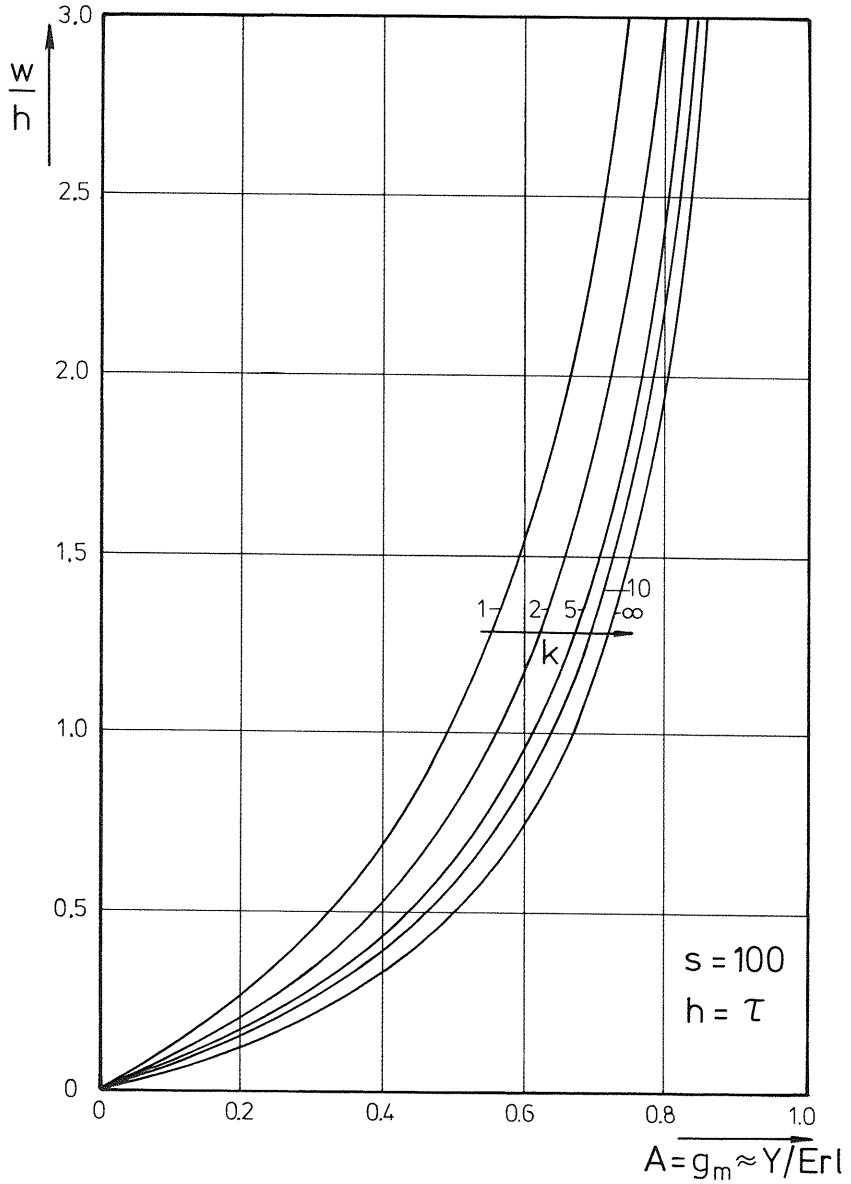


Bild 7.14: Einfluß des Parameters k der Erlang-k-Verteilung auf die normierte mittlere Wartezeit $\frac{w}{h}$

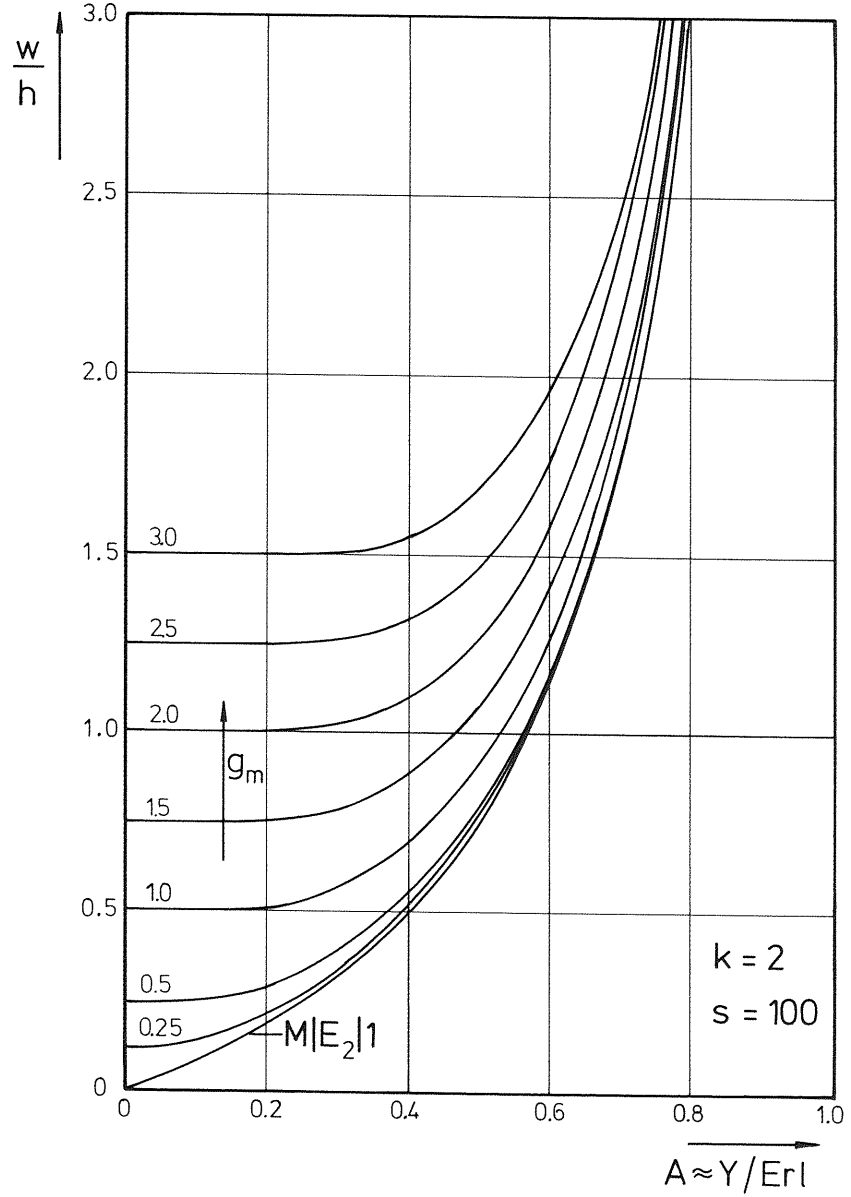


Bild 7.15: Einfluß der mittleren Gruppengröße g_m auf die normierte mittlere Wartezeit $\frac{w}{h}$

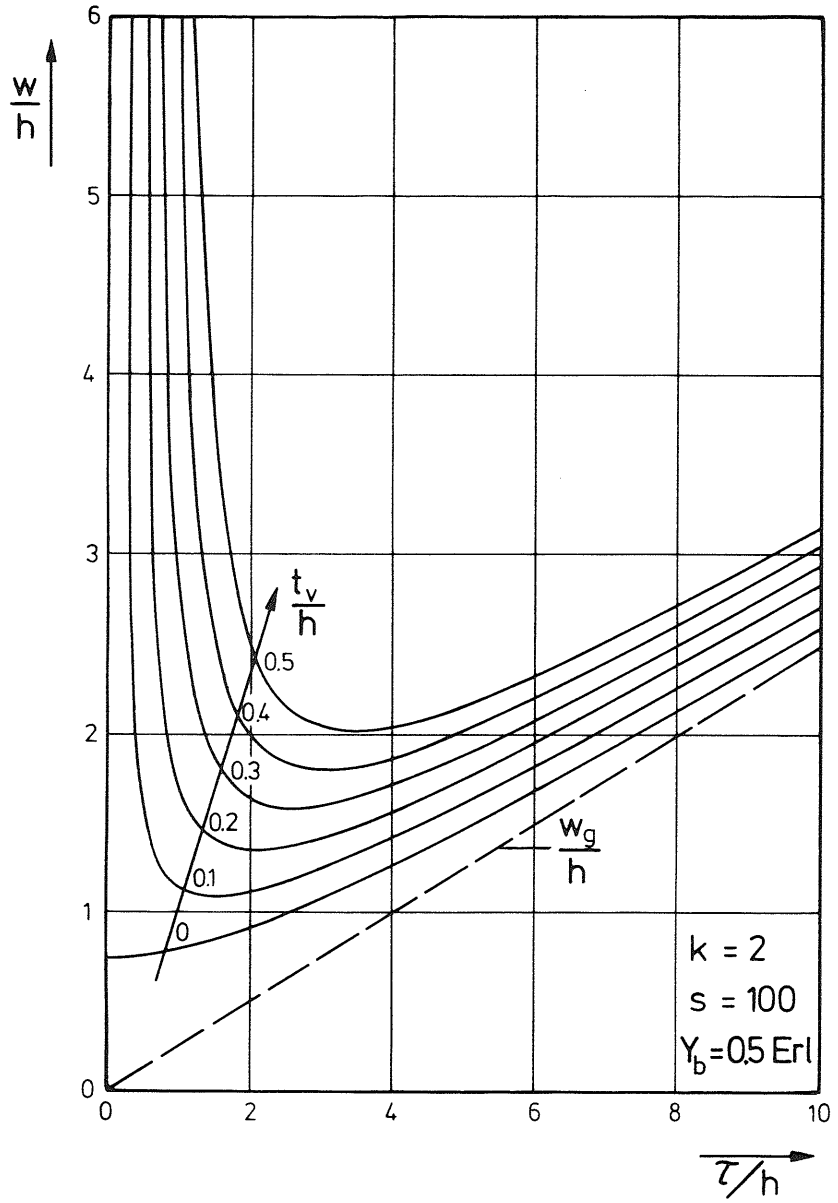


Bild 7.16: Einfluß der Verwaltungszeit t_v auf die normierte mittlere Wartezeit $\frac{w}{h}$

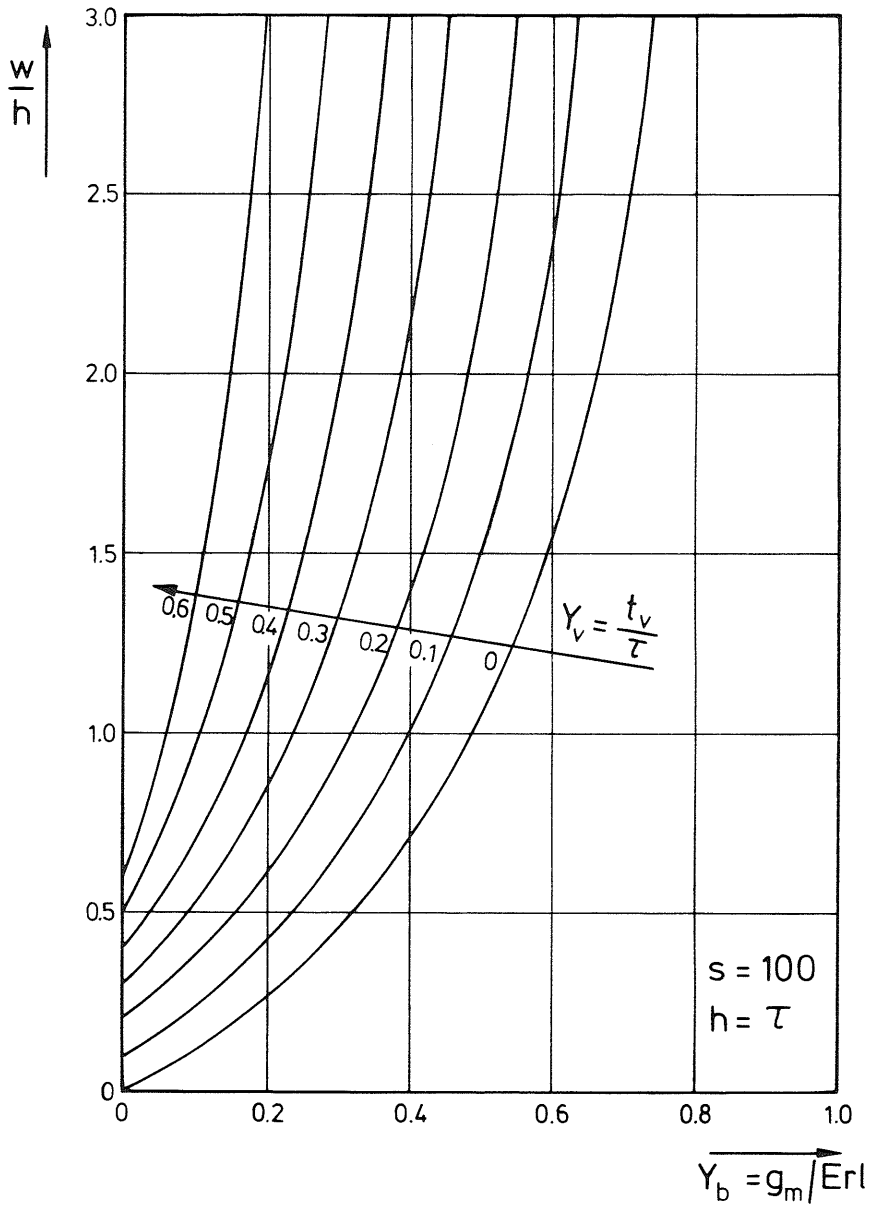


Bild 7.17 Einfluß der Verwaltungszeit t_v auf die normierte mittlere Wartezeit $\frac{w}{h}$

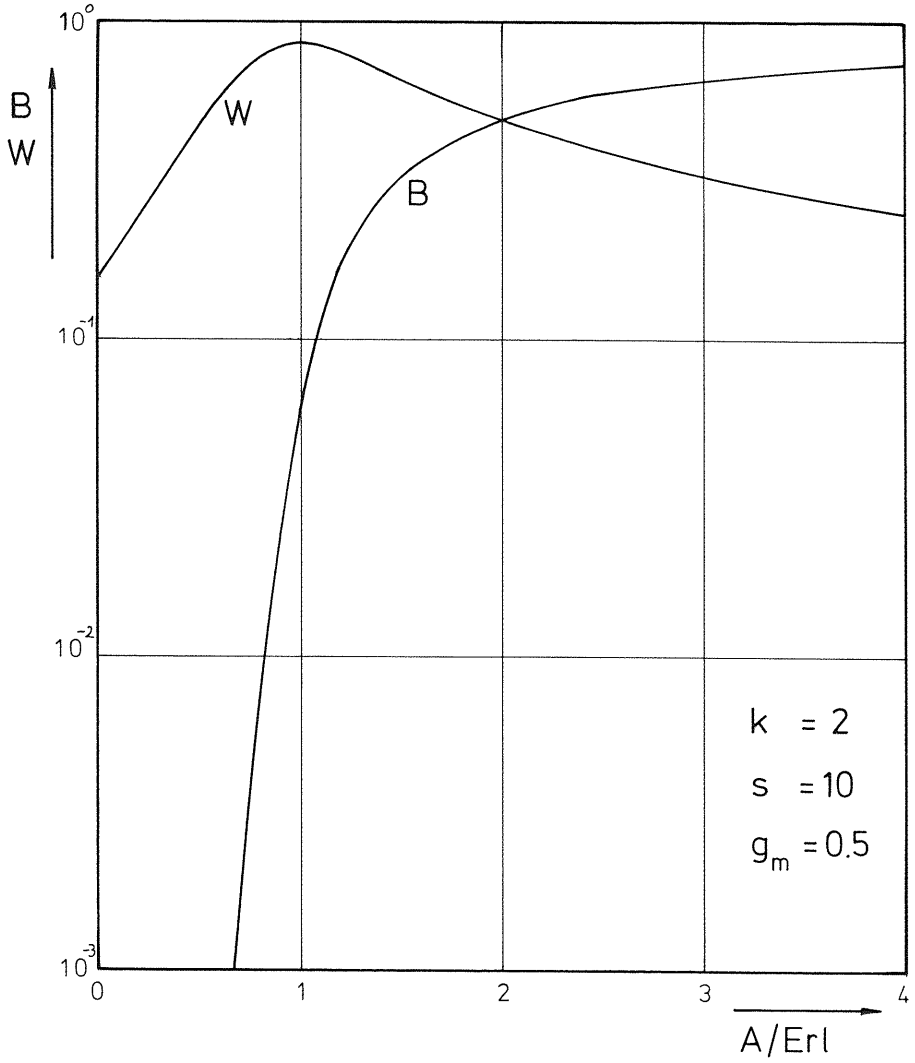


Bild 7.18 Wartewahrscheinlichkeit W und Verlustwahrscheinlichkeit B in Abhängigkeit vom Angebot

8. Zusammenfassung

In der vorliegenden Arbeit wurde die Steuersoftware für den Steuerrechner eines Labormodelles einer rechnergesteuerten Vermittlungsstelle entwickelt. Als Steuerrechner kommt ein handelsüblicher Prozessrechner (Siemens 306) zum Einsatz.

Die allgemeine Struktur der Software des Steuerrechners, die noch unabhängig vom verwendeten Rechner war, wurde den speziellen Eigenschaften des verwendeten Rechners angepaßt. Dazu war es notwendig, zwischen der eigentlichen Vermittlungssoftware und der Hardware eine Softwareebene einzufügen, die als Überwacher bezeichnet wird.

Nach einer ausführlichen Beschreibung dieses Überwachers wurden die Steuerprogramme (Organisationsprogramme) beschrieben, die den Ablauf von sogenannten Arbeitsprogrammen steuern und überwachen. (Die Arbeitsprogramme führen die eigentlichen vermittlungstechnischen Aufgaben durch.) Hierbei wurde insbesondere das Zusammenspiel dieser Organisationsprogramme mit den Arbeitsprogrammen und der verschiedenen Organisationsprogramme untereinander, die unterschiedliche Prioritäten haben, beschrieben.

Das Problem des Speicherschutzes, der die Programme im Arbeitsspeicher vor Überschreiben durch fehlerhafte Programme schützen soll, wurde in einem eigenen Kapitel behandelt und eine mögliche Realisierung angegeben.

Für Testzwecke wurden Schnittstellen in die Steuersoftware eingebaut, die es gestatten, mit Hilfe der digitalen Simulation der Vermittlungsperipherie die gesamte Vermittlungssoftware auszutesten, bevor die vermittlungstechnische Hardware zur Verfügung steht. Weiterhin wurden mit Hilfe eines Umweltsimulators die Programmlaufzeiten der einzelnen Vermittlungsprogramme, die vom Systemzustand abhängen, gemessen und die Ergebnisse dargestellt.

Im abschließenden Kapitel wurde ein vereinfachtes mathematisches Modell für die Steuerungsabläufe im Steuerrechner entwickelt und analytisch untersucht.

ANHANG

A1 Allgemeine Begriffe

A1.1 Task

Ein Programm stellt nach DIN 44300 eine Folge von Anweisungen und Datenerklärungen dar, die zur Lösung einer Aufgabe erforderlich sind. Die Ausführung dieser Anweisungen erfolgt unter der Regie des Betriebssystems als Task /59,60,61/.

Eine Task stellt aus der Sicht des Betriebssystems (Überwacher) eine Verwaltungseinheit für die Prozessorzuteilung dar. Anstelle des Begriffes Task wird auch der Begriff Rechenprozeß /14/ oder auch nur Prozeß verwendet /62,63/.

Der Zusammenhang zwischen Task und Programm wird anhand des Bildes A1.1 kurz erläutert.

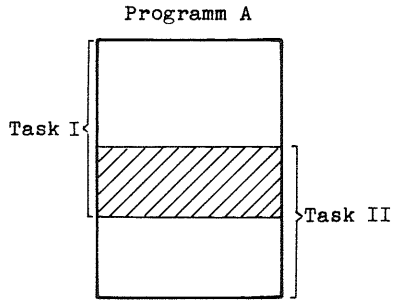


Bild A1.1: Zusammenhang zwischen Task und Programm

Das Programm A ist in den Arbeitsspeicher des Rechners eingelesen worden und als Task I registriert. Die Task I benutzt die durch die geschweifte Klammer angegebenen Befehle und Daten. Die Task I generiert während ihres Ablaufes durch eine Funktion des Betriebssystems die Task II. Beide Tasks können nun simultan zueinander ablaufen. Die von der Task II benutzten Befehle und Daten sind auf der rechten Seite des Bildes durch die geschweifte Klammer angegeben. Der straffierte Bereich des Programmes A wird von beiden Tasks gemeinsam benutzt. Es kann sich dabei sowohl um Befehle als auch um Daten handeln.

Ein Anwendungsfall dafür wäre z. B. die Verarbeitung von Daten, die auf Lochkarten gespeichert sind,und das Ausdrucken der Ergebnisse auf Schnelldrucker. Die Task I nimmt dabei das Einlesen und Verarbeiten der Daten vor,und die Task II druckt die Ergebnisse aus. Während die Task I Daten einliest und verarbeitet, kann die Task II Ergebnisse ausdrucken.

Die Einführung des Begriffes Task ist auch erforderlich, da bei der Unterbrechung einer Task die für die Fortsetzung der Task notwendigen Registerinhalte (z. B. der Stand des Befehlszählers) zwischengespeichert werden müssen. Diese Registerinhalte können in obigem Beispiel nicht dem Programm A sondern nur einer von beiden Tasks zugeordnet werden. Zur Speicherung dieser Registerinhalte wird jeder Task ein Taskkontrollblock zugeordnet.

A1.2 Listen

Wie bereits in Abschnitt A1.1 beschrieben wurde, können, durch Unterbrechungen angestoßen, mehrere Tasks parallel ablaufen. Zur Steuerung der Parallelarbeit ist es erforderlich, daß im Betriebssystem (Überwacher) "Buch geführt" wird über die verschiedenen Vorgänge. Für diese Buchführung sind Listen angelegt. Man unterscheidet drei Arten von Listen:

- Bitweise organisierte Liste
- Wortweise organisierte Liste
- Blockweise organisierte Liste

A1.2.1 Bitweise organisierte Liste

Bei einer bitweise organisierten Liste besteht ein Listenelement aus einem Bit. Die Liste wird i.a. aus einem Arbeitsspeichertwort aufgebaut, wobei die Lage des Bits im Wort die Adresse des Listeneintrages darstellt. Die Liste kann sich auch über mehrere Arbeitsspeicherworte erstrecken.

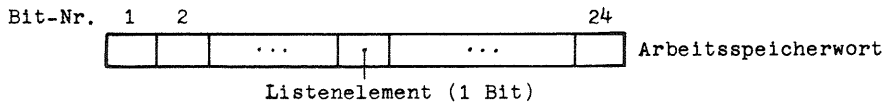


Bild A1.2: Bitweise organisierte Liste

A1.2.2 Wortweise organisierte Liste

Bei einer wortweise organisierten Liste besteht ein Listenelement aus einem Arbeitsspeicherwort. Mehrere zusammenhängende Arbeitsspeicherworte bilden die Liste, wobei die Nummer des Arbeitsspeicherwortes relativ zum Listenanfang die Adresse des Listeneintrages darstellt. Die Liste hat eine feste Länge.

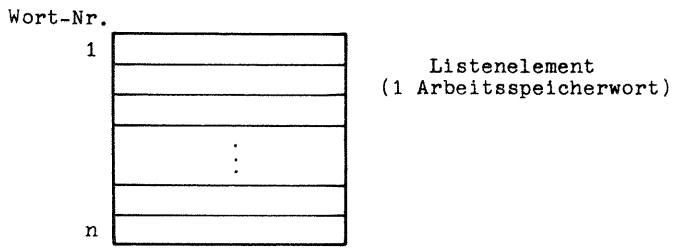


Bild A1.3: Wortweise organisierte Liste

A.1.2.3 Blockweise organisierte Liste

Bei einer blockweise organisierten Liste besteht ein Listenelement aus mehreren Arbeitsspeicherworten (Block von Arbeitsspeicherworten). Man unterscheidet dabei zwischen zusammenhängenden Listen und geketteten Listen. Bei zusammenhängenden Listen stehen zwei verschiedene Listenelemente in aufeinanderfolgenden Arbeitsspeicherzellen und die Liste hat eine feste Länge.

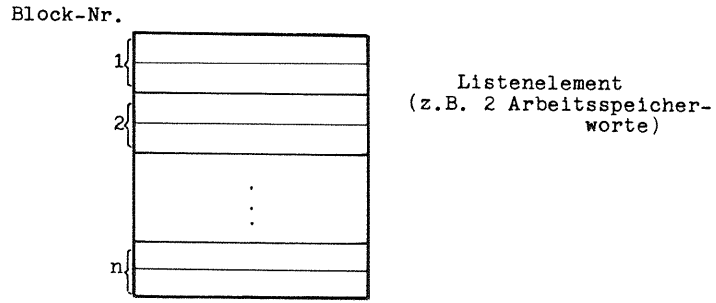


Bild A1.4: Blockweise organisierte (zusammenhängende) Liste

Bei einer geketteten Liste stehen die verschiedenen Listeneinträge nicht in aufeinanderfolgenden Arbeitsspeicherzellen. Die einzelnen Listeneinträge sind über Adressen miteinander verkettet.

In der vorliegenden Arbeit wurden einfach gekettete Listen verwendet (vgl. Bild A1.5). Dazu sind noch zwei Hilfszellen (ANFANG,ENDE) notwendig, die den ersten und letzten Listeneintrag angeben.

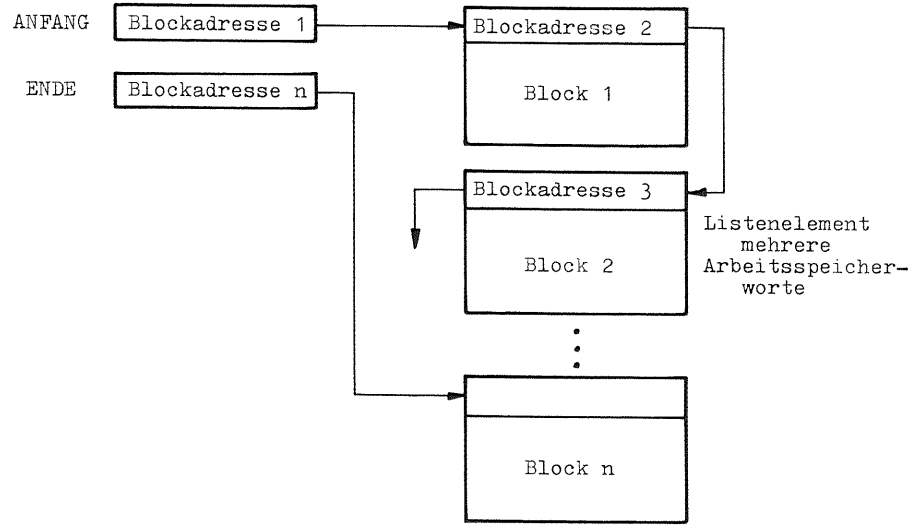


Bild A1.5: Blockweise organisierte (gekettete) Liste

In der Hilfszelle ANFANG steht die Adresse des ersten Blockes der Liste (im Bild die Adresse (Blockadresse 1) des 1. Blockes). Ist diese Zelle leer, so enthält die Liste keine Eintragungen. Pro Block ist eine Zelle notwendig, die den auf diesen Block folgenden Block der Liste angibt (im Bild bei Block 1 die Adresse (Blockadresse 2) des 2. Blockes). Im letzten Block der Liste ist diese Zelle ohne Bedeutung.

Die Adresse des letzten Blockes der Liste (im Bild Blockadresse n) ist in der Zelle ENDE eingegeben. Damit ist es möglich, ohne die Liste von vorn nach dem letzten Block durchzusuchen, einen neuen Block anzuketten, indem einfach die neue Blockadresse in der Zelle ENDE und in den bisher letzten Block eingetragen wird.

Die gekettete Liste braucht keine feste Länge zu haben, da beliebig viele Listenelemente angekettet werden können.

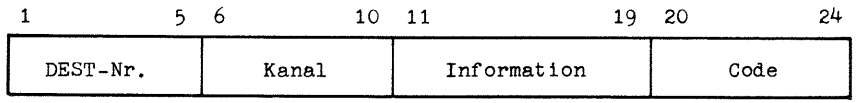
Die Organisation der Liste erfolgt nach dem FIFO-Prinzip (first in, first out), d.h. die Listeneinträge werden in der zeitlichen Reihenfolge des Eintrages wieder ausgetragen (abgearbeitet). Der Block 1 wird zuerst und der Block n zuletzt ausgetragen.

A1.3 Warteschlangen

Anforderungen unterschiedlichster Art an das Betriebssystem (Überwacher) werden in Listen zwischengespeichert. Z. B. können mehrere Tasks auf die Prozessorzuteilung warten. Die einzelnen Tasks sind dazu in einer bitweise organisierten Liste vermerkt. Man bezeichnet diese Liste als Warteschlange. Es finden alle in A1.2 beschriebenen Listen als Warteschlangen Verwendung.

A1.4 Meldungen und Befehle für den Informationsaustausch zwischen dem Vermittlungsrechner und der vermittlungstechnischen Peripherie

Der Informationsaustausch zwischen dem Vermittlungsrechner und der vermittlungstechnischen Peripherie erfolgt über sog. Meldungen und Befehle. Die Eingabeinformation wird als Meldung bezeichnet und die Ausgabeinformation als Befehl. Meldungen und Befehle haben ein einheitliches Format von 24 bit, das der Wortlänge des Rechners Siemens 306 entspricht.



DEST dezentrale Steuereinheit

Bild A1.6: Format einer Meldung bzw. eines Befehls

Die ersten 5 Bit geben die Herkunftsadresse bei Meldungen bzw. die Zieladresse bei Befehlen an (Nummer der dezentralen Steuereinheit KAS, VAS, ZKF). Bit 6 bis Bit 10 enthalten, wenn die Information auf einen Sprachkanal bezogen ist, die Nummer dieses Kanales. Bit 20 bis Bit 24 kennzeichnen die Meldung bzw. den Befehl. Die restlichen Bits enthalten Zusatzinformation, wie z.B. eine Teilnehmernummer oder die Nummer eines Hörtongenerators.

A2 Zusammenstellung aller realisierten Überwacheraufrufe

In diesem Kapitel sind alle realisierten Überwacheraufrufe zusammengestellt. Die verschiedenen Funktionen wurden im Rahmen von Studienarbeiten realisiert /64,65,66,67/. Es ist daher bei jeder Funktion die Studienarbeit angegeben, in der sie realisiert wurde. Zusätzlich wird, wenn in der vorliegenden Arbeit die Funktion behandelt wird, die Kapitelnummer des entsprechenden Kapitels angegeben.

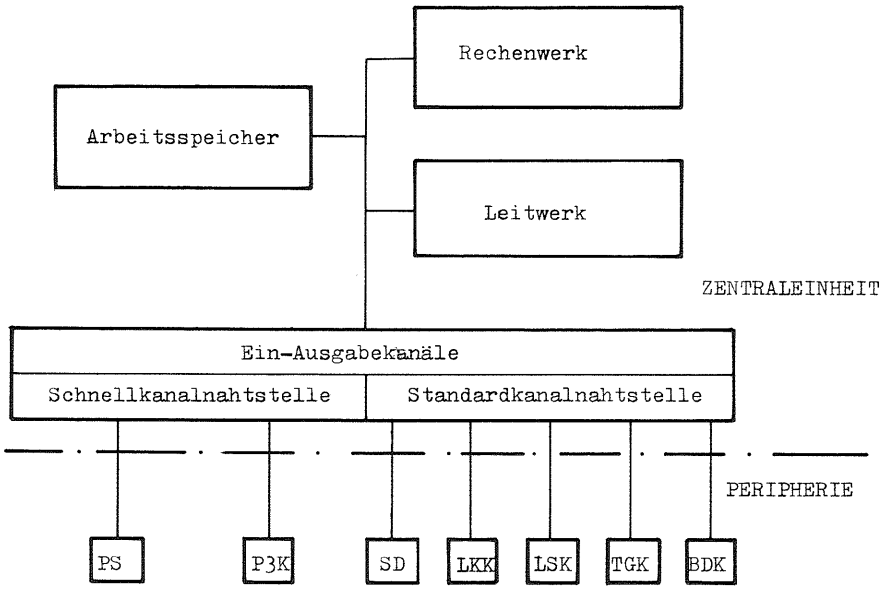
Die Überwacheraufrufe sind im allgemeinen kompatibel zum ORG306. Falls Abweichungen vorliegen, oder wenn der Überwacheraufruf neu eingeführt wurde, so ist dies in der Spalte Bemerkung durch ein A (bei Abweichungen) bzw. durch ein N (bei neuen Aufrufen) gekennzeichnet. Die Namen der Überwacheraufrufe entsprechen den Namen, unter denen die Überwacheraufrufe in der Makrobibliothek für das ORG306 enthalten sind. Die Überwacheraufrufe für das ORG306 sind in /15,68/ beschrieben.

Name	Funktion		Beschreibung	Bem.
BRLK	Anmelden mit Laden	Ablauf-organisation	3.3.3.2 /67/	A
BRST	Anmelden mit Laden			A
PNZU	Anmelden ohne Laden			
PLOE	Abmelden			
STRT	Starten			
ENDE	Ende			
ENDW	Unterbrechen			A
REST	Fortsetzen			N
EXWA	Warten			
EXWE	Warten			
WESL	Prioritätsverzicht			A
OBKO				
UBKO				N
OBRV	Basisadressenregister -			
OBVV	verwaltung			
UBRV				
UBVV				
BELG	Sperren	Synchroni-sation	3.3.4.2 /67/	A
FREI	Freigeben			A
BEWA	Erwarte Nachricht	Kommuni-kation	3.3.5.2 /67/	
BEUB	Sende Nachricht			A
QUIR	Sende Quittung			
QUIS	Sende Quittung			
NEXTZEI	Nachrichtenzeichen			
BSEI	Blattschreibereingabe		/67/	
BSAU	Blattschreiberausgabe			
LKEI	Lochkarteneingabe			
LKAU	Lochkartenausgabe			
LSEA		Lochstreifeneingabe	/66/	
LSEB				
LSEB1				
LSEB2				
LSEB3				
LSAA		Lochstreifenausgabe		
LSAB				
LSAB1				
LSAB2				
LSAB3				

Name	Funktion		Beschreibung	Bem.
LSAM		Lochstreifenausgabe	/66/	
LSAM1				
LSAM2				
LSAM3				
SDAT		Schnelldruckerausgabe		
SDAV				
DAER		Dateiverwaltung und Ein-Ausgabe für Magnetplatte	/64/	
DAERA				
DASL				
STEI				
ESVI				
STEI				
STAU				
WSEF		Ein-Ausgabe für die Vermittlungs-peripherie	3.5.2.3 /65/	N
WSER				N
WSAF				N
WSAR				N
MELD		Simulationsfunk-tionen für die Ver-mittlungsperipherie	3.5.4.2 /65/	N
BEFA				N
TAKT				N
BEDBA	gerätebezogene Dienstfunk-tionen	Funktionen für das Bedienungsfeld des Rechners	/67/	
BEDBB				
BEDKA				
BEDKE				
BEDSU				
DATU		Abfrage der Uhr im Taktgeber		A
ZEIT				A
MSEC				N
LIES	taskbezogene Dienstfunk-tionen			
SCHREIB				
BFUE				
BFZU				
NUMR				
STUE				
TEST				
UAAU				
UAEI				

A3 Siemens Rechner 306

Als Vermittlungsrechner für das Laborprojekt NVDV wird ein handelsüblicher Prozeßrechner, der Rechner Siemens 306 (S306), eingesetzt. Im folgenden werden kurz die Anlagenkonfiguration und einige charakteristische Eigenschaften des Rechners beschrieben.

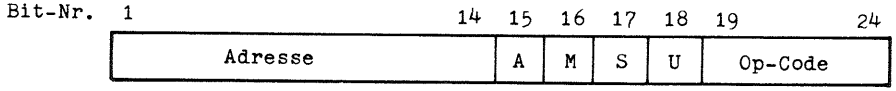


BDK Bedienungsblattschreiber
LSK Lochstreifenleser und -stanzer
LKK Lochkartenleser und -stanzer
TGK Taktgeber
SD Schnelldrucker
PS Plattenspeicher
P3K Prozeßelement

Bild A3.1: Anlagenkonfiguration des Rechners Siemens 306

Die Arbeitsspeicherkapazität des S306 beträgt 32K Worte zu je 24 bit, wobei ein zusätzliches Paritybit zur Sicherung der Information im Speicher dient. Die Zykluszeit des Arbeitsspeichers beträgt 0,6 µs für ein Wort, das die kleinste adressierbare Informationseinheit im Arbeitsspeicher darstellt.

Ein Arbeitsspeicherwort kann interpretiert werden als Festkommazahl, als Teil einer Gleitkommazahl (Gleitkommazahlen sind in zwei Worten dargestellt), als Binärmuster, als vier alphanumerische Zeichen, oder als Maschinenbefehl. Ein Befehlswort hat folgende Struktur:



A Akkumulatorkennzeichnung
M Modifikationsbit zur Auswahl des Basisadressenregisters
S Substitutionsbit
U Unterbrechbarkeitsbit (bzw. Modifikationsbit)

Bild A3.2: Befehlswortstruktur

Bit 1 - 14 geben eine Adresse an, die je nach Zustand der Zentraleinheit unterschiedlich interpretiert wird. Für die Interpretation der Adresse sind die Bit 16 und 18 maßgebend (Bild A3.3). Das Bit 15 dient zur Auswahl eines von zwei Akkumulatoren des Rechenwerkes und das Bit 17 kennzeichnet eine Adressensubstitution. Bit 18 dient zusätzlich auch zur Kennzeichnung von Unterbrechbarkeitsstellen. Bit 19 - 24 stellen den Operationscode dar, mit Hilfe dessen 55 Einadressbefehle unterschieden werden, darunter Gleitkommabefehle für die vier Grundrechenarten.

Die Arbeitsweise des Leitwerks und damit der Zentraleinheit ist durch drei Funktionszustände gekennzeichnet:

- Normalzustand (NZ)
- Privilegierter Zustand (PZ)
- Alarmzustand (AZ)

Je nach Funktionszustand werden die Operanden bzw. die Befehlsadressen unterschiedlich interpretiert (vgl. Bild A3.3).

Im Zustand NZ kann nur relativ über eines der Basisadressenregister (BAR) BAR 9 oder BAR 15 adressiert werden. Die Auswahl des betreffenden BAR erfolgt durch das Bit 16 des Befehlswortes. Im Zustand PZ bestimmen Bit 16 und 18 die Art der Adressierung und im Zustand AZ kann nur absolut adressiert werden.

Bit 16	Normal-zustand	Privil. Zustand	Alarm-zustand	Bit 18
0	relativ BAR 9	absolut	absolut	0
0	relativ BAR 9	relativ BAR 9	absolut	1
1	relativ BAR 15	relativ BAR 15	absolut	1
1	relativ BAR 15	absolut	absolut	0

Bild A3.3: Adressierungsmodi des Rechners Siemens 306

Im Zustand NZ bzw. AZ dient das Bit 18 des Befehlswortes nur zur Kennzeichnung, dass eine Unterbrechung durch eine Kanalanforderung bzw. durch ein Alarmsignal zugelassen ist, und nicht zur Adressenmodifikation.

Der Normalzustand dient zur Bearbeitung von Anwenderprogrammen, während der Privilegierte Zustand für den Ablauf des Betriebssystems (Überwacher) vorgesehen ist. Ein Teil der Befehle (z. B. Ein-Ausgabebefehle) ist nur im Zustand PZ erlaubt. Der Alarmzustand ist für Programmteile vorgesehen, die eine schnelle Reaktion auf ein Unterbrechungssignal (Alarmsignal) hin erfordern.

Von der Peripherie des Rechners sind besonders der Taktgeber TGK und das Prozeßelement P3K zu erwähnen. Das Prozeßelement stellt die Schnittstelle des Rechners zum Multiplexer für die Ein-Ausgabe zu den dezentralen Steuereinheiten der Vermittlungsperipherie dar (s. auch Abschnitt 3.5.1). Der Taktgeber liefert alle 10 ms ein Alarmsignal an den Rechner zur Aktivierung der Taktbearbeitungsroutine für die Ein-Ausgabe zur Vermittlungsperipherie. Er enthält zusätzlich einen Uhrenbaustein, über den per Programm die Uhrzeit und das Datum abgefragt werden können.