AEG TELEFUNKEN DATA PROCESSING

TR 440

Time-Sharing Computing System

Introduction

TR 440

Time-Sharing Computing System

INTRODUCTION
as of March 1971

The brief description is to familiarize the reader with the most important characteristics of the computing system and to give him a general survey of the programs connected with the Time-Sharing Computing System.

Application systems which also may be used in connection with the TR 440 Time-Sharing Computing System have not been taken into account. This includes the TELDOK Documentation System, the DBS 440 Data Bank System and the PSS Planning and Control System for Production.

Order-No N31.B0.04 E
Edition 0371 - V24
If reproduced, source must
be acknowledged
Printed in Western Germany

AEG-TELEFUNKEN
Information Systems Division
D-775 Konstanz, Bücklestr. 1-5

CONTENTS

1. SUMMARY

1.1. Concept

The development of Time-Sharing Computing Systems was triggered by the demand to decrease the turn-around time of users programs. The users desired the direct, ever ready access to the processing capabilities of the computer thus being able to feed in their program on-line and to test it.

The demand for 'ever ready access' was satiesfied by time-sharing of the limited resources (e.g. processor, storage etc.) by the programs of the individual users. Periodic assignment of the resources and the speed of the computer, gives the individual user the impression that the total capacity of the computer is at his sole disposal.

The demand for the 'direct access' led to the application of terminals (e.g. display units and teleprinters), installed directly at the users place of work thus permitting the direct communication with the computer.

The TR 440 Time-Sharing Computing System presented here has all the properties of a modern time-sharing computing system:

- easy conversational mode facilities
- large number of simultaneously serviced terminals
- efficient test aids

1.2. Configuration

Figure 1 presents a possible configuration of the TR 440 Time-Sharing Computing System.

The RD 186 Digital Computer - otherwise central unit of the TR 86 Computing System - controls the input and output intensive operations for the terminals and prepares the information for processing by the main frame RD 441 computer. Furthermore another RD 186 may be connected to this RD 186 Front End Computer as subordinate computer via coaxial cables or via remote data transfer lines.

The terminals are connected to the RD 186 and may be either display units or teleprinters. Background storage (drum storage and disc storage) and the conventional I/O-devices of the Time-Sharing Computing System are connected to the RD 441.

The following average configuration is recommended for the Time-Sharing Computing System.

- Central Unit
  1 RD 441 Computer inclusive core storage for 128 K whole words (minimum 64 K)
  6 Standard I/O-Channel Units (minimum 4)
  2 High-Speed I/O-Channel Units (for drum and disc storage, a minimum of 1 for one background storage)

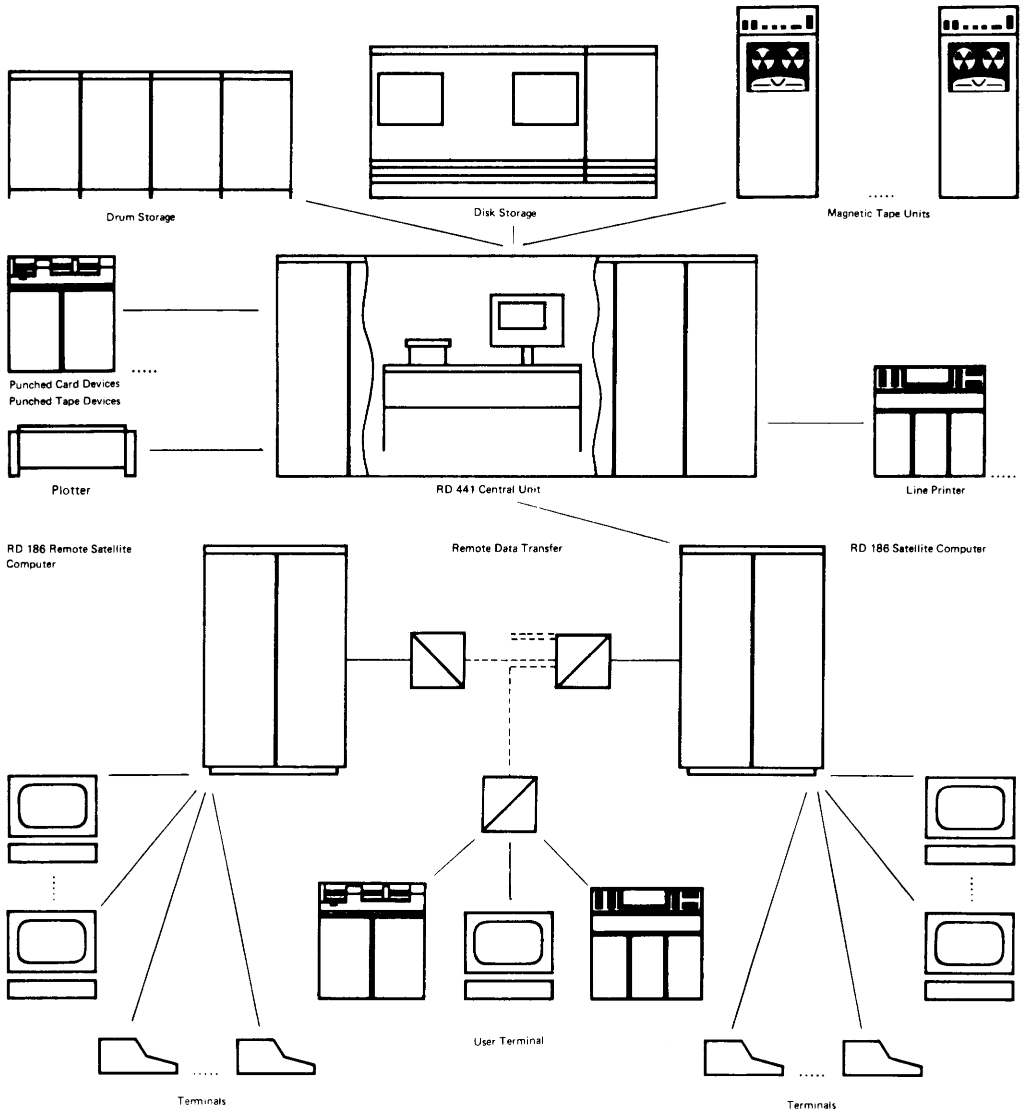

Fig. 1 Configuration

- Peripherial Units
  1 Drum Storage (2 modules)
  1 Disc Storage (3 sets of discs)
  4 Magnetic Tape Units (minimum of 2)
  2 High-Speed Printers
  1 Card Reader
  1 Tape Reader
- Satellite Units
  1 RD 186 Satellite Computer
  Display Units
  Teleprinters

The technical data of the system configuration is presented in chapters 2 and 3.

1.3. Facilities

Certain minimum characteristics are required of a Time-Sharing Computing System, they are presented below:

- Conventional batch processing must be possible parallel to the time-sharing operation mode.
- The user may only input data from the terminals - remote job entry - or conduct a dialog - conversational mode.
- The total computing capacity of the computing system must be accessible by using a terminal, especially to programs written in high level programming languages.
- Programs entered in the conversational mode are processed with equal priority - but ahead of the ones formulated in batch mode. -

The following characteristics are also of importance:

- All programs are processed in multiprogramming mode. In that way up to seven users programs are quasi-simultaneously processed.
- Spooling of I/O-information is carried out onto background storage. By this means it is possible to gather information for up to 72 users programs.
- The data management allows for supplementation and correction of source programs and data.
- In the data management the following file types are available for use:

SEQ	Sequential access according to the physical storage of the information.
RAN	Random access with record-numbers, the index is organised sequentially.
RAM	Random access with record-keys, the index is organised index-sequentially.

- Frequently required information may be held on disc storage under separate file administration (long term data retention).
- Time-Sharing mode becomes effective and significant by the means of the extensive test aids in the programming system.

1.4. Modes of Operation

Two kinds of jobs are available to the user of the Time-Sharing Computing System:

- A job is called a conversational job or a dialog if the user wishes to interrupt the processing, spontaneously, or as a reaction to interrogation by the system (correcting, input and output data, change variables).
- If the user is not interested in influencing the processing after the input of the job the job is entered and processed in the batch mode.

Conversational jobs may be entered via the terminal keyboard (also with the help of the attached tape reader at the teleprinter). Once started the job may request further input via the terminal or from the long term data retention storage.
Output tasks during conversational processing may be assigned to all output devices, whereby the information is intermediately stored in the disc storage (spooling).

Batch jobs may be entered via any input device and via the terminal keyboard Input to batch jobs being processed is only possible via magnetic tape units or long term data retention storage.
The output tasks are intermediately stored in the disc storage if they are not directed to a magnetic tape unit.

2.1. Block Diagram

In the basic configuration the RD 441 Central Computer of the Time-Sharing Computing System comprises of the Central Processor, the Priority Unit and the I/O-Processor.

The processor is composed of the control unit and the arithmetic unit. The central storage is equiped with a high-speed core storage with 64, 128 or 256 K words of 52 bits each.
The I/O-processor consists of the I/O-control unit, the peripheral interrupt unit and from 5 to 16 channel units for input and output devices.

The priority unit controls the access requests of the processor and those of the channel units to the central storage.

The central storage may be further expanded with a mass-core storage (cycle time 2,1 µsec, access time 1,1 µsec). Furthermore the incorporation of an additional processor is possible (Double Processor System).

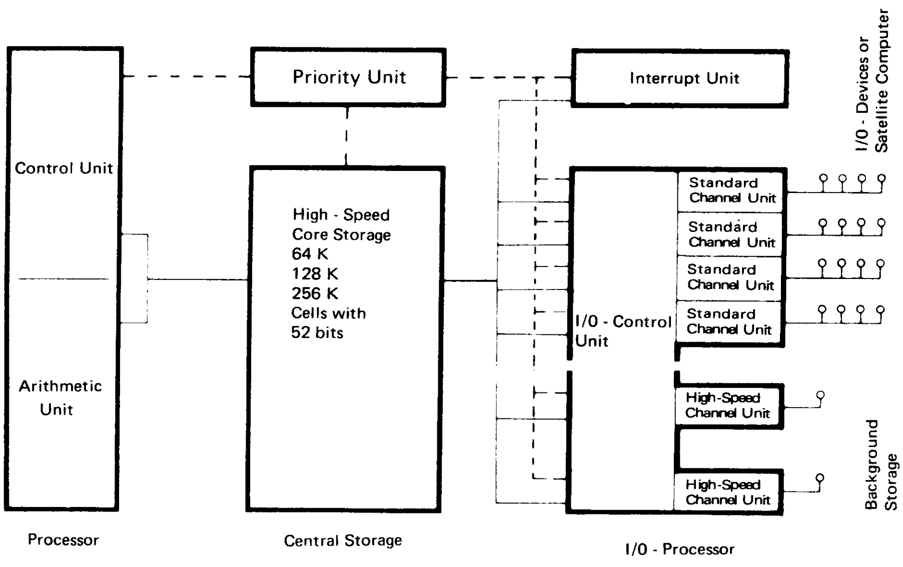

Fig. 2 Block Diagram

The computer is offered in three basic configurations varying in the size of the high-speed core storage and in the number of channel units. Every computer may be adapted, at the users office, to the growing needs of the user without exchanging the devices installed. The minimum configuration consists of 64 K words of high-speed core storage, one high-speed I/O-channel and 4 standard I/O-channels.

2.2. Data Structure

The registers of the central processor and the core storage locations each accomodate one word, thus the data format of the RD 441 is word-oriented.

A number of specific instructions are available for the selection and computation with word partitions of selective lenghts (especially bytes of 8 bits), special attention was paid to the effectivness of these instructions during development.

A whole word consists of 52 bits, of which the first two are used as mod 3 check bits and the next two as word type identifier bits.

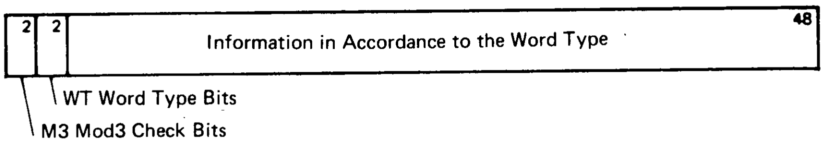

Fig. 3a Machine Word

In core storage, each numerical word (word types 0 or 1) contains a flag bit adjacent to the sign bit. When the word is loaded into an arithmetic register, the flag bit is stored separately and the bit position vaccated is set identical to the sign bit and used to catch overflows.

Sedecimal Floating Point Number

The floating point presentation of numbers is used in computations in which the order of the result can not be estimated or where the range of the results vary very much. The accuracy of the calculation is equal to 10 decimal digits.

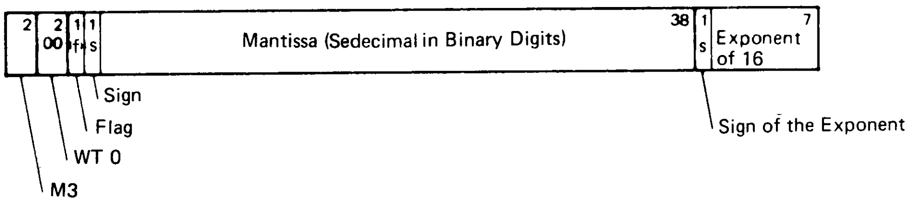

Fig. 3b Sedecimal Floating Point Number

Standardized Floating Point Numbers of normal accuracy are within the range of

$$7.4 \cdot 10^{-155} \leq |x| \leq 8.3 \cdot 10^{152}$$

If floating point numbers are represented by two successive whole words (double word) then a computation accuracy equal to 24 decimal digits may be attained.

Binary Fixed Point Number

The use of fixed point binary numbers is suitable for applications where the order of the input data, of the intermediate result and that of the final result can be surveyed. The simple computation accuracy is equal to 13 decimal digits.

On the other hand a fixed point number may also be stored in a half word which is equal to an accuracy of 6 decimal digits.

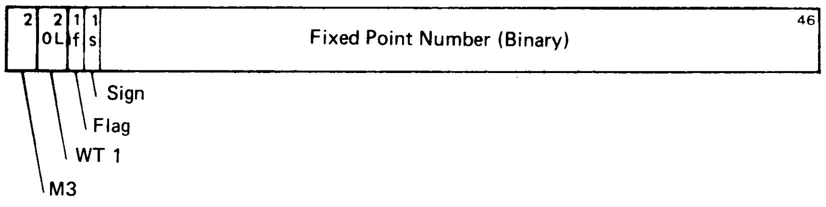

Fig. 3c Binary Fixed Point Number

Instruction Pair

A machine word accepts two instructions. The address field may be divided into a left and right address portions which then address index locations or represent specifications of the instruction.

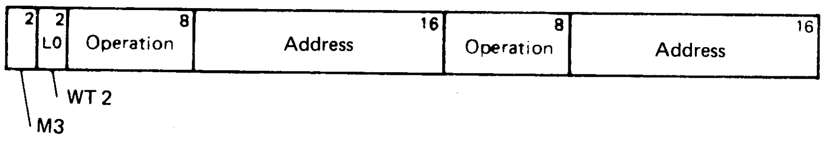

Fig. 3d Instruction Pair

Text Word

Standard RD 441 characters consist of 8 bits, called also bytes. Thus 6 bytes may be accomodated in one whole word.

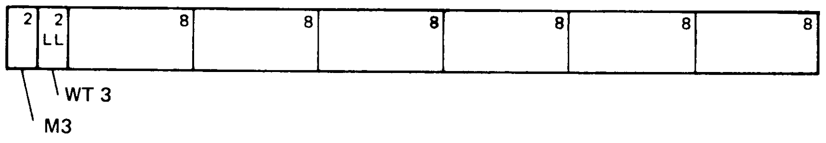

Fig. 3e Bytes as Example of Text Words

2.3.
Storage Unit

The Central Storage is composed of modules of 16 K whole words. Since every half word has an address assigned, the so-called address area of a module comprises 32 K. Every storage module has its own functional logic. If the module is busy because of an access to a word then no additional access to that module is possible within the cycle time of 900 nsec. However access to a word in another storage module can be made. The processor and the I/O-channel units are able to operate in parallel as long as they make access to different modules.

Address Interleaving

Since successive locations are frequently required to be addressed, addresses are interleaved, i.e. successive locations are found in different modules. Thus the effective cycle time depends on the expansion of the core storage and can only be given statistically. It may be as low as 125 nsec in the case of 16 storage modules. Figures 4a and 4b show how the addresses are distributed among the modules.

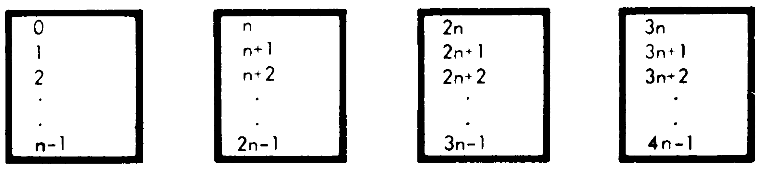

Fig. 4a Non-interleaved Addressing

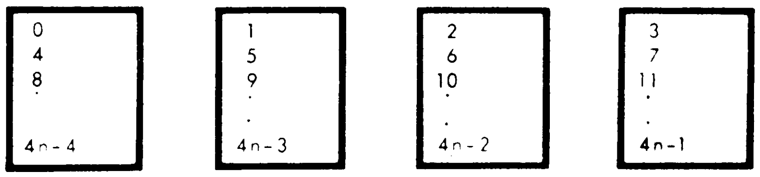

Fig. 4b Address Interleaving

1024 consecutively addressable storage locations form one physical page, and because of address interleaving, a page spreads across all modules.
A total access prohibit or a write prohibit may be set for each page, under program control.

Paging

At program load time, assignment of logical pages to physical pages is carried out by the operating system, and recorded in the page assignment table associated with the loaded program. Since this assignment may be completely random, access to data or instructions is carried out via the page assignment table. However, to save the time overhead of permanent table access (thereby double core access) to obtain data or instructions, the four last used logical/physical page relationships are stored in four page registers.

Storage Protection

Storage protection implies the protection of stored programs from mutual interference and the protection of certain parts of the program from changes through the program itself. For this purpose the page assignment table includes indicators for the storage protection.
Thereby it is possible to differentiate between write access (write and read), read access (read only) and access prohibit.

The mutual storage protection of the program results from the fact that every program interpretes only its assignment table but is not able to change it.

Index Storage

A consecutive area of 256 half words of the core storage is designated as index storage. The primary address of this area is stored in an index base register. The index address (8 bits) is added to the primary address for the purpose of addressing an index cell.

Four index registers contain the addresses and contents of the four last used index cells, thus avoiding the necessity of continuous core memory access to obtain index values.

2.4. Processor

In a wider sense the processor should be considered as a unit for computations. The Processor of a RD 441 comprises the Control Unit and the Arithmetic Unit inclusive Mikroprogramm Unit. A second processor can be installed thus permitting a doubling of the computing capacity.

Control Unit

The Control Unit handles the recalling of instructions from the core storage in the sequence fixed by the program. It prepares the individual instruction for execution and sends the respective enabling signals to the microprogram control unit.

If only the arithmetic unit is used by the microprogram for the execution of an instruction (e.g. during arithmetic operations) then additional instructions may be processed simultaneously if the arithmetic unit is not necessary for their execution.

The Control Unit of the RD 441 consists of 30 registers (5 of which are adressable by the programmer).

Arithmetic Unit

The Arithmetic Unit executes the arithmetic operations initiated by instructions. The execution takes place in parallel to the I/O-processor. The connection of the Control Unit and Arithmetic Unit to central storage is given by a so-called Collection Register.

The Arithmetic Unit of the RD 441 consists of 16 registers (6 of which are adressable by the programmer) and includes the Microprogram Unit.

Instruction Repertoire

The RD 441 has an extremely comprehensive instruction repertoire. 240 codes are differentiated in the 8 bits of the operation field. Moreover in some instructions a second operation field spezification or two addresses are placed in the address field (the 16 operation codes not occupied may be designated to macros).

The instruction repertoire may be divided into the following classes:

- floating point arithmetic of normal and double accuracy (21 instructions permanently wired)
- fixed point arithmetic of normal and half accuracy (27 instructions)

- non numerical operations for the processing of data with fixed or variable lengths (e.g. strings) or logical variables. These include
  branching operations (35 instructions)
  conversion and preparatory instructions
  setting and clearing instructions
  modifying and replacing instructions
  character transfer instructions for characters with 4, 6, 8 or 12 bits (byte instructions)

- System instructions for the transition between different levels of the programming. i.e. Switch between the mode of execution (see below).

- Included in the non-numerical operations are the processor instructions for input and output, which however are executed in the I/O-processor. The control of the I/O-processor is soley that of the operating system.

Execution Modi

There are five modi in the RD 441 for the execution of instructions. The reason for these modi is to enable varying address interpreations during the execution of instructions and to fix the use of certain types of instruction at different levels of the programming (storage protection, easy programming for the user and others).

Normal Mode is used for the user programs. The instruction unit considers all addresses transferred to it as virtual (relative to imagened primary address 0) addresses and assignes to them real (actually present in the core storage) addresses via the page assignment table.
The Process Monitor Mode varies from the normal mode only as far as an extended address area is admissable.

The System Mode is used for the next higher program hierarchy, especially the parts closely related to the hardware of the operating system. In this mode the control unit regards all addresses transferred to it as absolute addresses.

The Special Mode is in between the normal and the system mode and is used for the process of the subroutines which are actually part of the operating system but which may be called-up by an user program and then are to operate on users data.

The Maintenance Mode permits the process of checking and maintenance programs (addressing of additional registers for example).

2.5. I/O-Processor

The I/O-Processor comprises the I/O-Control Unit, the Standard Channel and the High-Speed Channel Units and the Interrupt Unit. The I/O-Control Unit is able to simultaneously execute up to 5 IOC-instructions, one each for the maximum of four High-Speed Channel Units and one for the maximum of 12 Standard Channel Units operating in time multiplex.

The central processor is occupied by the I/O-sequence only for the duration of one start instruction with which it triggers a channel unit. The I/O-Control Unit recalls the IOC-instructions from the storage for the started channel unit and executes them completely in parallel to the operation of the (central) processor.

Channel Units

The minimum configuration comprises 1 High-Speed and 4 Standard Channel Units and the maximum configuration is 4 High-Speed and 12 Standard Channel Units.

A Standard Channel Unit has interfaces (sub-channels) for four peripheral units. The transmission rate is up to 700,000 bytes of 8 bits each per second, whereby the characters are transferred in series via a coaxial cable.

A High-Speed Channel Unit has only one interface, by which a background storage device may be connected via an interface unit. However the rate of transmission is up to 3 million bytes per second, transferred in parallel via several coaxial cables.

Interrupt Unit

I/O-processes are conducted asynchronous from the central processor, and information regarding the condition of any I/O-process, device or channel is supplied to the operating system by means of peripheral interrupts. The interrupt unit generates an interrupt request, and when granted, places the 'Interrupt Word' in the appropriate memory location.

Types of Interrupt Requests

Call Interrupt Requests, which - externally triggered - may happen at any time are confronted by Block-, Error and Stop Interrupt Requests which - programmed or because of the state - may only happen during an I/O-process conducted at that time. Peripheral units or (in case of computer coupling) another computer report their readiness to transfer or accept information by means of a Call Interrupt Request.

Interrupt requests can be ignored or blocked by programming or during the execution of a microprogram.

Alarms

Alarms are interrupt requests to a processor triggered by an occurence in the computer (e.g. power failure). There are two categories of alarms:

- alarms of category 1 whose handling is a logical prerequesite for subsequent processing steps (arithmetical alarms, storage protection alarm and others)

- alarms of category 2 whose handling may at least be delayed for a short time (counter alarm)

There are, independent of the type of alarm, two alarm blocks. Alarm block 1 is set when an alarm occurs. An additional alarm of category 1 generates alarm block 2. If there is an alarm of category 1 during alarm block 2 then this leads to a computer stop. If an alarm of category 2 occurs then it is delayed by means of the set alarm blocks.

3. PERIPHERAL UNITS

This chapter gives a brief survey of the peripheral units which may be connected to the RD 441.

3.1. Background Storage

The background storage serves as residence medium for the software, as buffer for program input and output or for the swapping as well as storage for users data which are often recalled or which are very changing intensive.
This include:

TSP 500 Drum Storage

A disc storage with drum characteristics because of the fixed read and write heads and composed of 1 up to 5 modules.
Capacity: 1 module - 7.8 million bytes
5 modules - 39.2 million bytes
Average access time: 20 msec
Transfer rate: 979,000 bytes/sec

PSP 600 Disc Storage

A disc storage consisting of 8, 12, 16, 20 or 24 discs with moveable read and write heads which are combined on a flying head assembly at each disc face.
Capacity: 8 discs - 125.8 million bytes
24 discs - 377.4 million bytes
Average access time: 186 msec
Transfer rate: up to 794,000 bytes/sec

WSP 414 Disc Storage

A disc storage for exchangeable disc packs of 11 discs. Up to 8 drives may be connected to one standard channel. Two drives may be simultaneously operated via two interface units.
Capacity: 1 disc pack - 24.5 million bytes
8 disc packs - 196.6 million bytes
Average access time: 45 msec (per drive)
Average latency time: 12.5 msec
Transfer rate: 233,000 bytes/sec

3.2. I/O-Units

The users non-conversational jobs are introduced into the computing system via the input/output units the same as the results of the computations are read out via these units.

MDS 252 Magnetic Tape Unit

For 1/2'' magnetic tapes for ISO-9-track-operation, adaptable to 7-track operation.
Storage capacity: about 15 million bytes (at 800 b.p.i. and 800 yd of tape)
Transfer rate: max. 68,600 bytes/sec (at 800 b.p.i.)
Character density: 200, 550 and 800 b.p.i.

LSL 195 Paper Tape Reader	Photoelectrically reading, adaptable to 5 and up to 8-track punched tape, with built-in buffer storage for 256 bytes. Speed: up to 2,000 frames/sec
LKL 720 Card Reader	Photoelectrically reading, column by column, with pneumatic advance, reading comparison via second reading station by means of the light-dark test. Speed: 1200 cards/min
LSS 150 Paper Tape Punch	With 8 punches for 5 and up to 8-track punched tape Speed: up to 150 frames/sec
LKS 145 Card Punch	Punching row by row, reading comparison via check reading station, with offset stacking of reject cards in a stacker. Speed: 250 cards/min
SDR 176 High-Speed Line Printer	Line Printer with rotating printing cylinder with constant rpms, at whose circumference the characters are arranged per print section, with a buffer storage for one printing line. Character set: 29 capital letters, 10 digits, 24 special characters in case of the SDR 176-1 additionally 30 small letters plus another 32 special characters in case of the SDR 176-2 Printing speed: 1,000 .....1250 lines/min (500 - 625 for the SDR 176-2) 1250 lines/min in case of numerical output
ZCH 231 Plotter (as example)	Incremental plotter with vacuum buffered paper transport and 8 step vectors, paper width 13 in. Drawing speed: 1,000 steps/sec Pin movement per second: 25
3.3. Satellite Units	Users conversational or remote batch jobs are fed into the computing system via satellite units. The RD 441 Computer thereby is used as concentrator of the information.
RD 186 Computer	Medium size computer, operating wordwise in parallel with 30 single address instructions, wired binary fixed point arithmetic (2 µsec per addition) core storage for 8 and up to 64 K words with 24 bits each (half word at the RD 441) Interrupt unit with 8 or 24 priority levels. Various channel units.
SIG 100 Display Unit	For the display of characters of different print type and of vectors of optional length and position on a 12 in x 12 in screen with 512 x 512 raster points. Capacity: max. 64 lines, max. 86 characters/line Video repetition: 33 1/3 cycles constant, the central storage of a RD 186 serves as the video repetition storage for several SIG 100.
FSR 105 Teleprinter	Full duplex teleprinter with modified ALGOL-keyboard; storage device; additional devices: attached paper tape reader and punch. 69 character/line; operating speed 75 baud.

4. BS 3 TIME - SHARING OPERATING SYSTEM

4.1. System Structure

The BS 3 Operating System is typified by its modular construction, and its resident segments are located in the core storage of the RD 441 and in the core storage of the RD 186 which is used as a front end computer.
The non-resident segments (as the whole Programming System) are located in the drum storage.

The RD 441 Supervisor consists of a System Nucleus and a fixed number of independent program units, called modules. The task of the modules may be classified as follows:

- tasks with absolute priority in time:
  Handling of events by System Modules
- tasks which are highly I/O-intensive; they cause long waiting periods - in comparison to the processor - due to the slow peripheral units:
  Handling of the tasks by I/O-Monitors
- tasks, which may be computing intensive and whose handling is subject to time conditions (response time at the terminals):
  Handling of the tasks by (Conversational) Job Monitors.
- tasks, which may be computing intensive but whose handling is not subject to time conditions:
  Handling of the tasks by (Non-Conversational) Job Monitors.

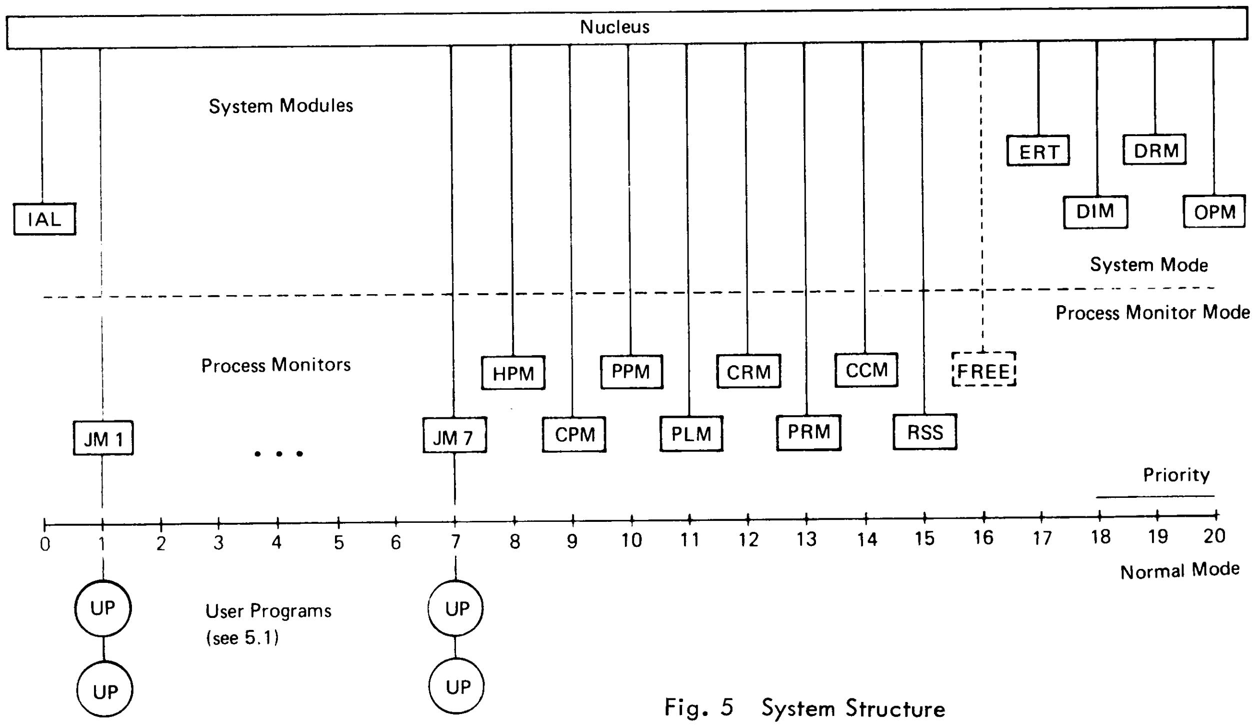

Fig. 5 System Structure

A fixed allocation has been made between the modules and priority because of the task classifications indicated, see fig. 5 (in a later version the priority will be allocated dynamically).

The indicated sequence of the task classification is necessary as to meet the requirements for acceptable response time and good through-put i.e. optimal use of the system.

The Modules are sub-divided into System Modules and Process Monitors which differ in the varying task assignment and in their mode of execution (see 2.4.). Figure 5 arranged according to priority and mode of execution. The explanations for the abbreviations used are presented in the text below.

4.2. Nucleus

In the BS 3 the Nucleus administers the processing resources: processor, core storage (in the RD 441), I/O-units and channels. Administration here signifies a 'physical' administration (assignment and release) of the resources; the 'logical' administration (scheduling and re-scheduling) is carried out by the resource scheduler (see 4.5.).

CPU-Assignment

The System Nucleus allocates the use of the central processor to the system modules (under demand or time slice multiprogramming operation), according to their relative priorities, and their ability to make use of processing time or not.

Storage Allocation

The storage media (core storage and background storage) are administered as pages of 1 K (1024) whole words, the core storage by the Nucleus, the background storage by the associated system modules (see 4.3.).
A process requests the required storage facilities from the System Nucleus and releases them upon completion of the task.

Coordination of the Peripheral Units

The Nucleus accepts input and output tasks from the modules, distributes them to the queues in front of the I/O-channels and processes them via the channel units. Following this the interrupt request of the I/O-channels are transferred to the corresponding modules.

In case of input-output transfer the I/O-monitors report the peripheral unit concerned to the Nucleus and release them after completion of the transfer.

The TR 86 S'peripheral unit' (RD 186 front end computer and its peripheral units, teleprinters and display units) does not have a special status in it's handling by the Nucleus. Output to the terminals is transferred to the channel unit to which the RD 186 is connected to.

4.3. System Modules

The System Modules fulfill the following activities:

Inactive Loop

The Inactive Loop is always in a nonstalled condition, i.e. can always make use of central processor time. When all other modules are idle or stalled, the IAL gets possession of the central processor.

Emergency Routine

The Emergency Routine ERT is activated in case of system failures. It is normally in an idle state and is overlooked during the allocation of the central processor.

Disc Storage Module

The Disc Storage Module DIM administers the disc storage and organizes the information transfer to and from the storage medium.

Drum Storage Module

The Drum Storage Module DRM controls and administers the drum in a similar way.

Operator Module

The Operator Module OPM has the highest priority, and gives the operator the oppurtunity to intervene in the processing sequence with the help of operator commands (e.g. interrogation for the state of the system).
Besides that, the Operator Module administers the console and the control panel and enables the input and output of information between the modules and the operator.

4.4. Process Monitors

Processes are understood to be programs which give clearly defined services to the user of the computing system. Whereas the System Modules receive their tasks from the Process Monitors and fulfill system-related tasks, that is they are not directly accessible to the user, the Process Monitors are representatives of the Nucleus to the user.

Process Monitors are sub-classified as those which operate I/O-units - I/O-Monitors - and those which administer computing tasks for the user - Job Monitors -.

I/O-Monitors

The I/O-Monitors HPM, CPM, PPM, PLM, CRM and PRM (compare fig. 5) organize the information transfer between the background storage and the peripheral units high-speed printer, card punch, paper tape punch, plotter, card reader and paper tape reader.

Magnetic tape units are operated by an I/O-Monitor which is part of the Job Monitors.

Satellite System

The Communication Monitor CCM is a mutual communications partner to all program sequences related to the terminals. It accepts buffered outputs in the core storage or in background storage and after request transmits them in partitions to the RD 186.

The Communication Monitor accepts input information from the terminals by way of the RD 186. It buffers incomplete information in the background storage, complete information is transferred to the respective processes (see fig. 6).

Because of the nature of it's task, the software in the RD 186 is called Terminal Module (TLM). It organizes the terminal I/O and the transfer of information and from the Communication Monitor via the computer coupling. Thereby it takes care of the registration of various terminal states, it buffers small partitions of I/O-information and looks after the regulated flow of information. The Communication Monitor CCM (in the RD 441) and the Terminal Module TLM (in the RD 186) together form the Satellite System SAS.

Job Management

A user job (conversational or non-conversational job) results in the operation of an user program sequence (command language interpreter, compiler etc.). The program entered by the user becomes a job step - the objekt sequence -.

The Job Monitor has the task to administrate the user program sequence and to offer system services to the individual user program.

The Job Steps are part of the Programming System (see chapter 5).

4.5. Resource Scheduler

Normally the processing of any job starts with the acceptance of the source program and ends with the output of the results at whatever output media was requested. In between is the processing of the job for which 'resources' are needed.

Resources

The main task of the Resource Scheduler RSS is to decide over the assignment of the resources.
In this connection the resources are:

- the processor
- the storage media (core and background storage)
- the Job Monitors
- the input and output devices (via I/O-Monitors)
- the job elements (the core storage areas for the internal description of a job).

The following requirements are taken into consideration by the Resource Scheduler:

1. The resources, especially the processor and the core storage are to be put to maximum use.
2. Processing takes place according to a defined importance (weight, sequence of input etc.)
3. Jobs are to be processed as soon as possible. The response time especially in case of dialog should be within such limits that the user has the impression that he is adequately served according to his own working speed and the problem posed by him.

In some points these demands are contradictory, thus parameter evaluation is conducted by the Resource Scheduler and a decision is made according to the following rules:

- a good processor loading is to be reached by parallel processing of up to 7 jobs (multiprogramming). Latency time of a job (e.g. because of I/O-transfer) is to be utilized by other jobs.

- dialogs are more important then non-conversational jobs (response time).

- batch jobs are 'weighted' dependant upon installation specific and job specific parameters and are completed according to their relative priorities. It is possible that the arrival of an important job will cause a temporary withdrawal of certain resources from less important jobs.

  The total system time (not only processor time) is to be distributed to the jobs in such a way that rule 3 is fulfilled (time-sharing). In that way an appropriate reaction time for all dialogs is attained by cyclic assignment of the required core storage and processor (time slicing). The 'next' dialog is always the most important one in this cycle.

To be able to conduct the resource assignment the Resource Scheduler is set 'ready' at fixed time intervals or because of a message from the Process Monitors.

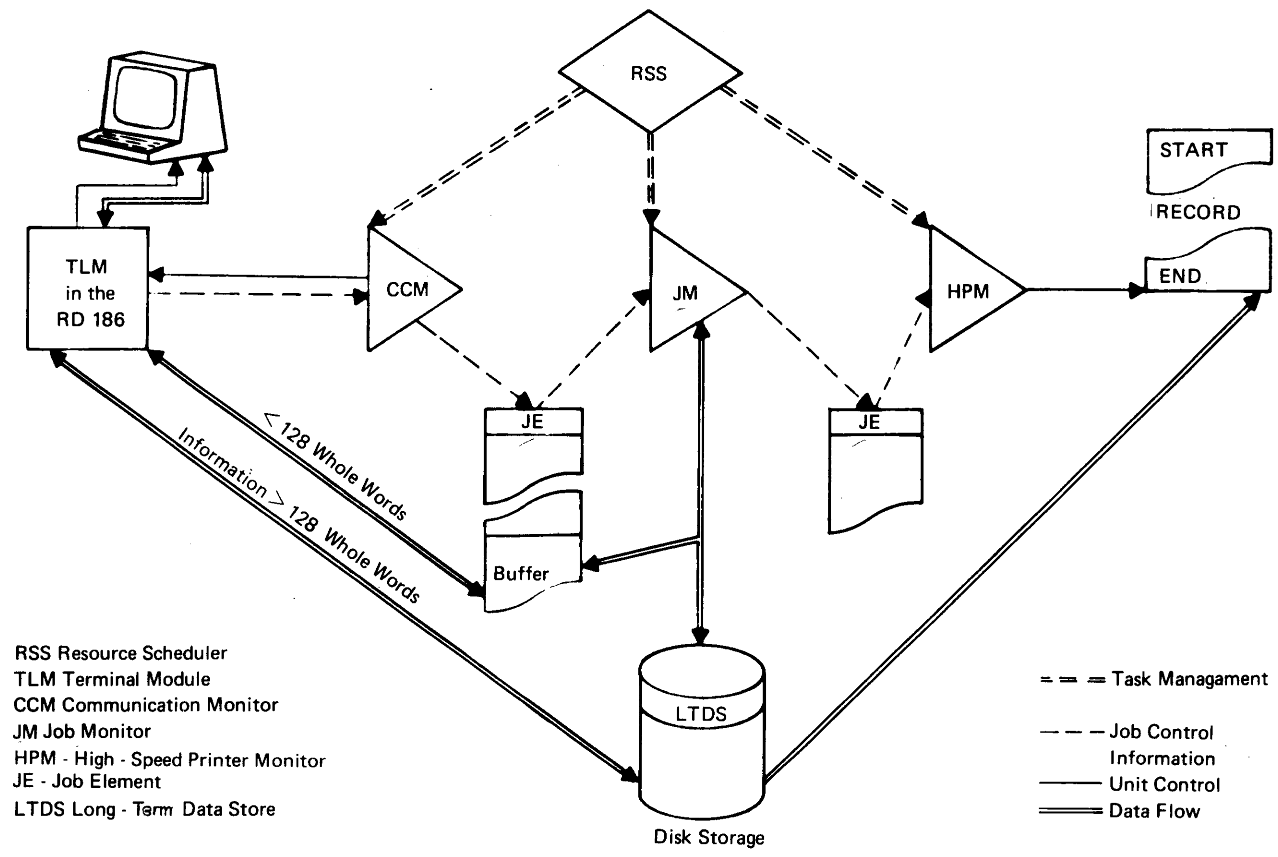

Fig. 6 Dialog Processing

4.6. Job Processing

A job is started with the XBA-(start batch job) or XBG (start dialog)-command, it contains a sequence of command language statements (chapter 5) and is terminated by an XEN-(terminate job)-command. If the job is entered in the conversational mode then the user is able to influence the processing. Otherwise the CL statements contain the sequence of the requests which are entered with the user programs.

In Figure 6, representative of other modes, the dialog sequence presented via a display unit which only create a print task as output is shown.

The Communication Monitor accepts the XBG-command and informs the Resource Scheduler that a Job Monitor is to take over the job. This results in a conversation between the user and the user program, administered by the Job Monitor. Thereby the information is exchanged either in the core storage or via the disc storage depending on the amount. Additionally the user is able to use the data base of the Long-Term Data Store (compare 4.7.). At the end of the conversation the High-Speed Printer Monitor is directed to output the print information accumulated on the disc storage.

Swapping

The basic idea of a time-sharing computing system is the time-sharing of the limited resources available to the programs of the user. Since the human being has a relatively long reaction time in comparison to the machine he will in most cases not notice the time slicing. He will have the impression that the service of the computing system is at his sole disposal.

Every dialog may use the processor for a certain time interval-service interval-after the last assignment of a Conversational Job Monitor. If the conversational job does not give the cause for swapping within a service interval - terminal output with subsequent input -, then the program set of the conversational job is forcibly displaced to the drum storage. The thus unoccupied resources are made available to other dialog.

4.7. Data Management

The Data Organisation as part of the Job Monitor represents a complete program complex. The user of the TR 440 Time-Sharing Computing System has the storage media drum, disc and magnetic tape at his disposal for the storage of data bases. Services of the Data Organisation are offered to him for the creation, administration and processing of the data bases.

Files

The user (and partly the operating system) keeps his information in the form of files. The term file signifies that the information is structurized and that a location reference is kept about the file. The files are composed of records-the smallest addressable unit of information. The types of files are differentiated according to the manner of access to their records.

- SEQ Sequential access to the records
  The physical sequence of the records on the storage medium determines the logical sequence.

- RAN Random access with record-numbers
  The information is tightly packed in the sequence of their input. The logical sequence is represented by indices which are filed whithout gap in ascending order according to their binary interpretation.

- RAM Random access with record-keys
  The difference to the RAN is that the variation possibility of the indices with equal storage need is much higher since no indices are filed for undefined records.

Up to 255 files can be simultaneously administered within one task. The user has the possibility to logically combine the files under the term of a data base. The file name has to be clearly defined only within the data base. The user may handle the information within a file at will, however, a file may be secured against illegal access.

Libraries

The number of data bases administrable by a Job Monitor is limited to 8. Two special data bases are created and processed by the operating system.

- A Utility Data Base (System Library)
  The Utility Data Base comprises the program sets of the Programming System. It is available to all users for reading only. The Utility Data Base is created anew for every system generation, it's lifetime is therefore the same as that of the system.

- A Standard Data Base (Job Library)
  A Standard Data Base is available to every user's job, containing job specific data (e.g. link modules, retrace lists, description of the user programs).
  The Standard Data Base is created by an user program - the CL Interpreter (also see 5.). The user implicitly puts his files into the Standard Data Base if he does not define a private data base.
  The Standard Data Base is cleared at the end of each job.

Long-Term Data Store
LTDS

The Long-Term Data Store offers the possibility to permanently store files. Thereby the users' files are combined under a common users' identifier which takes the place of a data base name (Users Library). The decision about the contents and the lifetime of the file in the LTDS is that of the user.

The file in the LTDS - all of the above mentioned types are allowed - differ in their access characteristics.

- C-files (common files) appointed for the access of several users and for which a coordination of simultaneous processing has to take place.

- P-file (private file) may only be changed and cleared by the user who has set-up the file.

Manipulation of the Long-Term Data Store files is possible by means of Source Handling commands (see 5.7.) and through higher programming languages.

Segment Concept

The necessity of transferability of larger information units among various storage media - e.g. swapping of a program set - leads to the indirect data addressing.
Coherent information units of one or more pages (see 2.3.) are designated as segments.
Access to these information sets organized in these segments by the user may either be implicitly (via the Data Organisation) or explicitly (service performances of the Job Monitor).

5. PS PROGRAMMING SYSTEM

The total of all programs available to the user of the TR 440 Time-Sharing Computing System - system programs - and all routines held as link modules, are designated as Programming System.

5.1. User programs

A program in the sense of the software-organisation of the TR 440 is every program which fulfills the following conditions:

- all instructions are conducted in normal mode (compare 2.4.)
- all contact to the operating system is made via the Job Monitor, which processes the corresponding users program.
- a description of this program exists in the Job Monitor (start address, size etc.).

Because of the last mentioned description the Job Monitor is able to start an user program.

Linkage Editor

Naturally it is not the task of the user to generate the necessary description. This is done by means of a special program - the Linkage Editor -.
It compiles a runable program out of the individual link modules and it deposits the description in the Standard Data Base of the user (compare 4.7.).

On request of the user this description, inclusive reference lists, allows the many programming aids to read out information with the description from the source language in case of an error in the program sequence (source related dump) and to locate the error position.

C.L. Interpreter

A job comprises the actual information to be processed in the computing system (source, data) and instructions as to the form in which it is to be conducted. The control information in form of commands (see below) is decoded by a special user program, the (Command Language) Interpreter, and the requested programs are started.

5.2. Command Language

The TR 440 Command Language was created to give the user a standardized and easy means of control for the processing sequence. It's validity is not limited to the Programming System but it allows the complete external control of the computing system.
The language elements are commands which are preceeded by a special character, the so-called escape symbol (from here on represented by ¤), so as to separate them from the rest of the input data.

Activity Commands

The basic element of the command language is the Activity Command by which a certain performance is requested of the Programming System. It starts with the activity identifier followed by the specifications e.g.:

```
¤UEBERSETZE,SPRACHE=FTN,QUELLE=TEXT,...
  (COMPILE,LANGUAGE=FTN,SOURCE=TEXT,...)
```

Many specifications are defined for every activity (in the example SPRACHE,QUELLE etc.). The specification identifier may be omitted if a preset sequence is followed. Also activity and specification identifier may be abbreviated as long as the meaning is unique, e.g.:

```
¤UEB.,,FTN,QU.=TEXT,...
```

Presetting

There is a universal presetting for the individual specifications, which are divided into the obligatory and the optional ones.
This presetting becomes valid if no statement is made to a specification.
The basic C.L. repertoire can be sub-divided in:

- Standard Commands
  Compile, Assemble, Link Edit, Start, Compute
- General File Handling
  Create, Declare, Secure, Block
- Source Language Handling
  Enter, Correct, Clear, Copy, Merge
- Data Handling
  Sort, Collate, Copy, Compress (sources)
- Data Transfer Handling
  Input and Output of object, Output of data
- Assembler Macro Handling
  Enter, Clear and Inform
- Long-Term Data Store
  e.g. Create, Release of long-time data

Extension of the Repertoire

The existing C.L. repertoire may be extended via the DEFINIERE-command or via a procedure definition without mistakingly altering existing commands.

A number of existing commands are combined to form a new command by means of a procedure definition. Thereby formal parameters may be introducted which then appear as specifications of the new command. Example:

Procedure definition:

```
¤*RECHNE(TEXT,SPRACHE,PROG) (COMPUTE(TEXT,LANGUAGE,PROGRAM))
¤UEBERS.,QUELLE=*TEXT,SPR.=*SPRACHE (COMPILE,...)
¤MONTIERE,PROGRAMM=*~~SPRACHE~~ PROG (LINK EDIT,...)
¤STARTE,PROGR.=*PROG,DUMP=F-NEST'A-NEST (START,...)
```

Procedure call-up:

```
¤RECHNE,TEXT=QUELLE,SPR.=FTN,PROG=TEST
(COMPUTE,TEXT=SOURCE,LANGUAGE=FTN,PROGRAM=TEST)
```

Command Sequence

If the job is processed in the batch mode then the compilation of the source may be erroneous. A subsequent link editing and start atempt would be senseless. For this reason there are two special commands which interrupt the job after the occurence of an error, (FEHLERHALT, 'ERROR-HALT') or (SPRINGE,BRANCH) which jumps out of the preset command sequence depending on the setting of specific variables.

5.3. Source Language Translation

One of the most important services of the Programming System is the translation of source language into runable object coding. This procedure is always completed in two step~~/~~s (see Fig. 7).

The source language, located either in a background storage LTDS file or a job work file, is first processed by the appropriate language translator and an intermediary relocatable coding is generated. The intermediary coding is then processed by the link editor and a runable object programme is produced out of one or more linked modules (compare 5.1.).
A typical example is shown in figure 7.

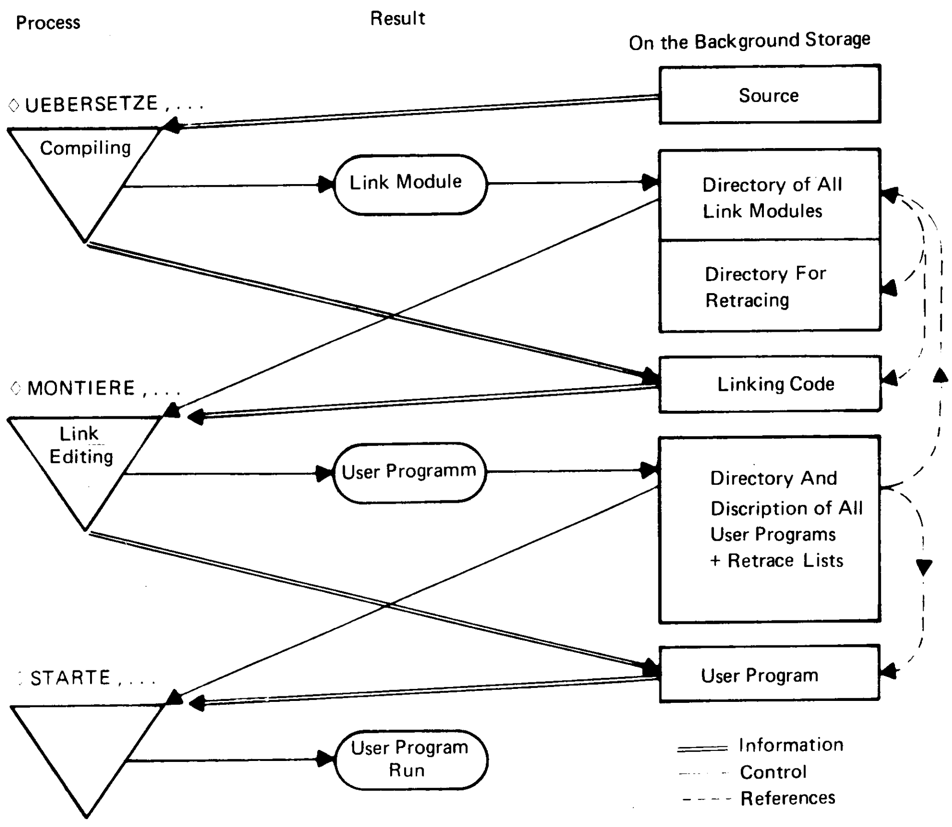

Fig. 7 Source Language Translation

5.4. Combination of Languages

The two steps of the translation are used among other things to permit the use of the different languages. Thereby the mutual call-up facilities in the languages ALGOL 60, FORTRAN and TAS (Telefunken-Assembler) has been realized. COBOL-routines may be called-up by TAS, and the rest of the languages indicated called up by COBOL-routines.

The combination is made during the link editing. The translation between two languages is not possible during one pass (e.g. insertion of assembler instructions into an ALGOL-program). The linkage of assembler procedures to higher languages only has to observe the conventions according to which the compiler generates its link modules.

The combination of higher language procedures will be explained with an example of ALGOL and FORTRAN. Some adaption procedures are necessary because of the different program structures, they are conducted in an intermediate procedure invisible to the user, this procedure may also be called up recursively.

Example

It is required that FORTRAN be able to administer dynamic fields with the help of an ALGOL-procedure:

- The FORTRAN-program for matrix manipulation is organized as a subroutine:

```
SUBROUTINE MATRIX (N,A1,A2)
DIMENSION A1 (N,N), A2(N)
.
.
.
```

- The following ALGOL-program contains the declaration of the two fields a1, a2 and the call-up for the FORTRAN-procedure:

```
'begin' 'integer' n;
        'procedure' matrix (x,y,z); 'fortran';
    read (n);
    'begin' 'array' a1 [1:n,1:n] , a2 [1:n] ;
    matrix (n,a1,a2)
    'end'
'end'
```

The transition between the two languages permits the almost unlimited use of procedures from the program library without consideration as to the language in which the procedure was written.

5.5. Programming Languages

In the following a brief survey of the programming languages implemented in the TR 440 Time-Sharing Computing System is given.

TAS
Telefunken Assembler

TAS is a fromat-free language with symbolic addressing which not only includes the actual 240 instructions (machine instructions) but also so-called pseudo instructions (assembler directives). A macro-library (with macros for input and output) is available to the user and it may be expanded by him.

FORTRAN	The FORTRAN-compiler processes two versions: - FORTRAN according to ASA-standards - FORTRAN-440, which is fully compatible to FORTRAN IV H, with language expansion for terminal input and output. Besides the FORTRAN-standard procedures there are FORTRAN-routines which permit string manipulation.
ALGOL	All but the own-variable of the complete language range of the ALGOL 60 has been implemented. The fortran-like COMMON concept has been realized as expansion. The input and output routines available comprise the ALCOR and ISO-procedures, all Knuth-Proposals-procedures and some special procedures for file handling.
COBOL	The realized language range corresponds to the ANSI-standard-COBOL 68 without the reportwriter but with the COBOL-library, the language specific segmention, the processing of three-dimensional fields and the sort facilities. For COBOL-objects there are not only dynamic controls and the procedure tracing (compare 5.6.) but also control of variables. There is a dump which upon request prints out in readable form variables according to the description in the data hierarchy.
GPSS General Purpose Simulation System	GPSS is a language for discrete system simulation. It is used for the formulation and treatment of problems in the fields of operations research and organisation analysis as well as in technical-scientific fields. The GPSS as an interpretative system is implemented with a pre-translator and a simulation processor, whereby an extensive compatibility with the GPSS/360 has to be taken into consideration.
RPG Report Program Generator	RPG is a problem oriented, fixed format language for commercial and organisation related programming. Essential language extensions are related to the configuration of tables. The access was extended to direct access (indexed) and sequential access.
BASIC	BASIC is a fortran-like language, especially conceived for time-sharing systems and easy to learn. The interpreter is fully conversational and permits almost unlimited manipulation of the sources. The realized language repertoire comprises that implemented by SIEMENS and GE.

5.6. Test Aids

This chapter is used to describe the aids which are made available by the Programming System for program checkout.

Source-related Dumps

In case of irregular program operation, the user may request that the contents of all or specific variables together with their source names or labels be dumped, thus avoiding the use of reference lists.

Retracer

A special system program - the Retracer - will additionally locate an error position in reference to the original source language. Furthermore the actual procedure hierarchy is given out. The dumps, and the form in which they are to be executed in case of an alarm are specified in the STARTE-command.

Tracing

A further test aid is the possibility of having a controlled program execution. The printing of the register contents or storage area can be made dependent on the addressing of a certain storage area or from the appearance of a certain instruction (e.g. recording of all appearing branch instructions).

This facility is also possible for programs generated from higher program languages. These routines are requested by the UEBERSETZE-command and are activated by the STARTE-command.

Dynamic controls

Upon request of the user dynamic controls may be compiled during source translations with reference to:

- observation of index limits
- compatability of actual and formal parameters
- permissability of loop parameters

Thus statically and syntactically unrecognisable errors may be located.

5.7. Conversational mode

During the dialog from a terminal all services of the Programming System with the exception of a few unimportant restrictions caused by the dialog are available.

After the start of the dialog it is possible to insert a single command or a sequence of commands.
In the conversational mode therefore it is possible to insert commands into an already present command sequence, which then are handled with priority. For example the error in an erroneous translation may be corrected with a command and the translation will then be repeated. Only the explicit request of the user will cause the continuation at the point of interrupt after these priority commands have been executed.

Interventions

In the conversational mode not only commands but also so-called interventions may be given to the Job Monitor, which processes the job, or to the C.L. Interpreter and the objekt run of the user. Interventions are only processed individually and after request. For example the Interpreter unterstands interventions to the continuation at the point of interrupt and for the clearing of all inputs and for the not yet executed commands.

If the user conducts his dialog with extensive source language and a large amount of data then it would be impractical to put them in via a terminal. In this case the use of the Long-Term Data Store is to be recommended (comp. 4.7.).

Source Language Handling

The Source Handling acts independently of or in connection with the LTDS-files. With their help the sources may be entered in the files, single lines may be corrected, cleared or exchanged. Even single characters of a line of text may be inserted, exchanged or cleared. Furthermore the user is able to combine several sources to a new one and to merge or correct partitions of any sources.

Extended Test Aids

Especially in dialog , the testing of a program is facilitated and accelerated by means of the possibility to react to results right away. An user program linked to be able to run in conversational mode may be stopped at any time during it's run. So-called Check Points are defined in the UEBERSETZE-command by assigning names to the numbers of the source lines.

If such a check point is active (indication in the STARTE-command) then the user program stops at the corresponding point and it reports the name of the checkpoint to the terminal. This may be followed by several reactions:

- continuation or completion of the user program run
- activation or release of defined check points
- bringing or storing of individual variables contents which may be chosen freely out of the program.
- dumping of all variables at the terminals or at the high-speed printer (buffered) according to the specifications.
- insertion of different commands
- stopping at the end of the user program run, i.e. before the execution of the following command.

Example for a Dialog

Below a dialog is shown, into which explanatory text has been introduced. The user terminates his inputs by ¤. (escape symbol, point) and he may input again after the output of ¤: .

```
¤¤KOS*KONSOLE FREI
¤XBG,BEN=000003 BENUTZERNAME¤.
0059*

START  BENUTZERVERWALTUNG  MV01

      09.02.71    10:29

ENDE  BENUTZERVERWALTUNG
GIB KOMMANDOS¤:¤TDEKLARIERE,ANTON,U200
¤TEINTRAGE,ANTON,PROT.=KO,INF=/
         DIMENSION A(10),B(10)
1        SCHREIBE(9,2)
2        FORMAT(1H0,10X'BERECHNUNG VON 10 QUADRATZAHLEN',/,
      1  10X,'ERWARTE ANFANGSWERT ')
         READ(8,3) C
3        FORMAT(F10.0)
         IF(C.EQ.0) GOTO 20
         A(1)=C
         B(1)=C*C
         DO 10 I=1,9
         A(I+1)=A(I)+1
10       B(I+1)=(A(I)+1)**2
         WRITE(9,4)
4        FORMAT(1H0,10X,'QUADRATZAHLEN ')
         WRITE(9,5) (A(I),B(I),I=1,10)
5        FORMAT(5X,F10.0,5X,F10.0)
         GOTO 1
20       STOP
         END¤.
```

The job number is given out after signing on. The user's index at the computing center checks the justification of the user to conduct a dialog. Then the C.L. Interpreter waits for commands. A file, ANTON, will be declared and filled with a fortran source program without a program listing.

```
GIB KOMMANDOS¤:¤TKORRIGIERE,ANTON,20,INF.=/
1       WRITE(9,2)¤.
```

Subsequently, additional commands are expected: A compilation could be requested which in this example would end with an error. See line 20 SCHREIBE (9,2). This error must first be corrected.

```
GIB KOMMANDOS¤:¤UEBERSETZE,SPR.=FTN,VAR.=KV'D,TRACE=-STD-,
KE=20-FTN1'90-FTN2'120-FTN3,QUELLE=ANTON¤.

START PS&FTNCOMP  MV63C
ANFANG PROTOKOLL
000010              DIMENSION A(10),B(10)
000020   1          WRITE(9,2)
000030   2          FORMAT(1H0,10X'BERECHNUNG VON 10 QUADRATZAHLEN',/,
000040         1    10X,'ERWARTE ANFANGSWERT ')
000050              READ(8,3) C
000060   3          FORMAT(F10.0)
000070              IF(C.EQ.0) GOTO 20
000080              A(1)=C
000090              B(1)=C*C
000100              DO 10 I=1,9
000110              A(I+1)=A(I)+1
000120   10         B(I+1)=(A(I)+1)**2
000130              WRITE(9,4)
000140   4          FORMAT(1H0,10X,'QUADRATZAHLEN ')
000150              WRITE(9,5) (A(I),B(I),I=1,10)
000160   5          FORMAT(5X,F10.0,5X,F10.0)
000170              GOTO 1
000180   20         STOP
ENDE PROTOKOLL

ERZEUGTES MO : STDHP
UEBERSETZUNG FEHLERFREI

ENDE PS&FTNCOMP .RECHENZEIT:0,39 SEK.
GIB KOMMANDOS¤:¤MONTIERE
¤STARTE,AKTIV=KEINE(FTN1),DUMP=F-KONSOL(A,B)¤.
```

Now the UEBERSETZE-command is given. Thereby the specification VARIANTE (VAR.) indicates if the object program is able to run in conversational mode. By means of TRACE it is possible to engage a program execution control later on. Finally the reaching of source lines 20, 90 and 120 (out of the recording of TEINTRAGE, see also listing of the compiler) is defined as check point (KE) with the names FTN1 through FTN3.

This translation was error-free and it generates the link module STDHP (Standard Program). Subsequently the program is to be linked and started (the listing of the link editor is omitted here). The check point FTN1 is activated in the STARTE-command. The specification DUMP indicates that fields A and B will be output at the terminal in case of an error.

```
START STDHP
STDHP *KE= FTN1¤:KEAKTIV(FTN2,FTN3-5)¤.
STDHP *KE= FTN1¤:KTRACEEIN(ASSIGN)¤.
STDHP *KE= FTN1¤:¤.
```

The program is stopped at check point FTN1 (line 20). Then the check points FTN2 and FTN3 are activated by the user by means of the intervention KEAKTIV. FTN3 is to report only after it has been passed through five times. In the following the intervention effects the recording of all assignments (ASSIGN). The program is continued with an 'empty' intervention.

```
          BERECHNUNG VON 10 QUADRATZAHLEN
         ERWARTE ANFANGSWERT ¤:         5¤.
 ** IN ZEILE 80 ZUW.:  A(1) =   0.500000000000E+001
STDHP *KE= FTN2¤:KEPASSIV(FTN1,FTN2)¤.
STDHP *KE= FTN2¤:¤.
 ** IN ZEILE 90 ZUW.:  B(1) =   0.250000000000E+002
 ** IN ZEILE 100 ZUW.:  I =   1
 ** IN ZEILE 110 ZUW.:  A(2) =   0.600000000000E+001
 ** IN ZEILE 120 ZUW.:  B(2) =   0.360000000000E+002
 ** IN ZEILE 120 ZUW.:  I =   2
 ** IN ZEILE 110 ZUW.:  A(3) =   0.700000000000E+001
 ** IN ZEILE 120 ZUW.:  B(3) =   0.490000000000E+002
 ** IN ZEILE 120 ZUW.:  I =   3
 ** IN ZEILE 110 ZUW.:  A(4) =   0.800000000000E+001
 ** IN ZEILE 120 ZUW.:  B(4) =   0.640000000000E+002
 ** IN ZEILE 120 ZUW.:  I =   4
 ** IN ZEILE 110 ZUW.:  A(5) =   0.900000000000E+001
 ** IN ZEILE 120 ZUW.:  B(5) =   0.810000000000E+002
 ** IN ZEILE 120 ZUW.:  I =   5
 ** IN ZEILE 110 ZUW.:  A(6) =   0.100000000000E+002
STDHP *KE= FTN3¤:KTRACEAUS¤.
STDHP *KE= FTN3¤:KDUMPE¤.
```

The program requests a starting value for it's computation (line 50) and assigns this value to the variable A(1). The check point FTN2 is reached in line 90. FTN1 and FTN2 are released by the intervention KEPASSIV. After FTN3 (line 120) has been passed four times this check point reports too (see above). Then first of all the Tracing is released by the intervention and an explicit dump request (not because of an error case). Only field A will be shown as to give an idea of a source-related dump.

```
FELD A
A (1)
 0.50000000000 E 001     0.60000000000 E 001     0.70000000000 E 001

 0.80000000000 E 001     0.90000000000 E 001     0.10000000000 E 002

4 *                      !!!!!!!!!!!!!!!!!!!!
```

```
STDHP •KE= FTN3¤:KEPASSIV(FTN3)¤.
STDHP •KE= FTN3¤:¤.

            QUADRATZAHLEN
              5.             25.
              6.             36.
              7.             49.
              8.             64.
              9.             81.
             10.            100.
             11.            121.
             12.            144.
             13.            169.
             14.            196.

            BERECHNUNG VON 10 QUADRATZAHLEN
           ERWARTE ANFANGSWERT ¤:=       20¤.
READ, DATEI 8, SATZ 2, ELEMENT 1, REAL•4, SATZ KUERZER ALS VOM FORMAT
 ODER VON E/A-LISTE GEFORDERT.''=       20''.

START PS&RUECKVERF   MV20

     1. FORTRAN-HP   STDHP
     ZEILE     50,   ADRESSZONE    1,  ADRESSE      7

ENDE PS&RUECKVERF .RECHENZEIT:0.11 SEK.
```

After the dump FTN3 again reports for new intervention. Thereby FTN3 also is released. The new starting value had been given in a false format so as to demonstrate the effectiveness of the Retracer. The error, as expected, appeared in line 50.

Subsequently the dump requested in the STARTE-command would follow, it has the same configuration as in the intervention KDUMPE. Thereby the program is stopped. Finally it is started again by the user but this time without test aids.

```
ENDE STDHP .RECHENZEIT:1.24 SEK.
+++++ZULETZT BEARBEITETES KOMMANDO:
¤STARTE,AKTIV=KEINE(FTN1),DUMP=F-KONSOL(A,B)

FEHLER: OPERATORLAUF MIT FEHLER BEENDET
GIB KOMMANDOS¤:¤STARTE¤.
START STDHP

            BERECHNUNG VON 10 QUADRATZAHLEN
           ERWARTE ANFANGSWERT ¤:          20¤.

            QUADRATZAHLEN
             20.            400.
             21.            441.
             22.            484.
             23.            529.
             24.            576.
             25.            625.
             26.            676.
             27.            729.
             28.            784.
             29.            841.

            BERECHNUNG VON 10 QUADRATZAHLEN
           ERWARTE ANFANGSWERT ¤:           0¤.

ENDE STDHP .RECHENZEIT:0.42 SEK.
GIB KOMMANDOS¤:¤XEN¤.
¤¤KOS•KONSOLE FREI
```