This chapter contains papers which are produced for education purposes within the TSS-18 department.

On section level you will find the main areas to be taught.
Each section is divided into subsections.
Subsection 0 gives a contents of the section.
Subsection 1 contains the topics to be handled within this main area.
The subsequent subsections describe the topics mentioned in subsection 1 in full detail.

Subsection 0: CONTENTS OF SUBSECTION 2

Philips' Telecommunicatie Industrie B.V. Hilversum The Netherlands Nederland
3522 000 71631

Subsection 1: TOPICS TO BE HANDLED FOR INTRODUCTION IN TCP-18 APPLICATION.

1 Introduction.

1.1 Aim of this paper.

The aim of this paper is to give a list of subjects to be handled as an introduction to TCP-18 application programming. These subjects are structured in such a way that a selection can be made according to the knowledge level of the trainee. The handling of the subjects can be done by different people who are familiar with the details of that particular subject.

Desired result: Understanding of technical solutions in TCP-18 for call processing.

1.2 Abbreviations used in Section 2.

A-side : Side of the exchange with the calling subscriber or the incoming trunk.
ABLC : Abnormal blocking condition.
ADB : Associated data block.
AJ : A-subscriber Junction circuit. (Connected to an LB.)
AJI : Incoming A-subscriber Junction circuit. (Connected to an incoming line.)
AMA : Automatic message accounting.
BJ : B-subscriber Junction circuit.
B-side : Side of the exchange with the called subscriber or the outgoing trunk.
CCA : Call coordination A-side.
CCB : Call coordination B-side.
CCM : Call coordination message.
CIM : Call input message.
CLME : Close message.
CME : Central/Common memory.
CPH : Call phase.
CPM : Call processing machine
CR : Call register.
DC : Device controller.
DCIM : Direct call input message.
DM : Data message.
DSP : Digit send packet.
DUEM : Direct unit event message.
E/M : Receiving/Sending. (For 'Empfangen und melden'.)
EP : Event packet.
EXIT : Give up control to MCP.
FD : Fast driver.
FIFO : First in first out.
HW : Hardware.
I/O : Input / Output.
ID : Identification packet.
IOM : Input / Output machine. (Part of the "MCU".)
ITFR : Incoming trunk forced released.
JGR : Junction group.
LB : Line link block.
LC : Line circuit (For subscriber line).
ITFR : Incoming trunk forced released.
MCP : Master control program.
MCU : Main control unit.
MFC : Multi frequency code.
MR : Marker.
MS : Memory switch.
OTBB : Outgoing trunk backwards blocked.

PCM : Pulse code modulation.
PCS : Peripheral control subsystem.
PG : Project generator.
PME : Processor/private memory.
PPM : Primary PCM multiplexer.
RC : Response command.
RGAT : Macro 'Return to the gate'.
RM : Response message.
RSB : Receiver / Sender data block.
SD : Slow driver.
SH : Signal handler.
SH-B : Build-up signal handler.
SH-L : Line signal handler.
SNETS : System-network testsystem.
SCM : Start confirmation message.
SCP : System condition panel.
SCU : Secondary control unit.
STRT : Start message.
SW : Software.
TB : Trunk block.
TCP-18 : 18 bits PRX processor.
TOS : Telephony operating system ; part of the programming that supports the 'Application' programming.
UEM : Unit event message.
UC : Unit controller.

2 General lecture without implementation aspects.

Available.

2.1 Goal of the call processing.

-Look at an telephone exchange as a black box.
-Separation of a call in an A and B-side.
-Advantages of such a split (source-destination separation).

2.2 Relation between hardware and software.

-Events are the base for software triggering.
-Scan for changes in hardware from outside. I/O commands for changes in hardware from inside.
-Split between TOS and application (definition in Interface manuals).
-Event distribution by TOS.

2.3 Multicontrol aspects.

-Principle of multicontrol for call processing (hardware structure).
-Explanation multicontrol not necessary for call processing.
-Parallel processing means:
 --Split of work between different machines (up to 3).
 --I/O philosophy.
 --Time aspects (time critical work/ non-critical work).

3 Application (Call Processing) structure.

Available.

3.1 Introduction.

-Necessity of programming in modular,functional blocks.
-Structure with levels
-HW --> telephony functions.
-Function of UC/DC's, SH's and Call Coordinators.

3.2 Communication between different levels.

-Necessity of sequential processing.
-Quick translation of HW-events into SW-functions.
(Minimum holding time of event 16 msec in hardware to enable software to react and process it)
-Solution: Messages which are handled sequential with the help of a gate mechanism.

3.3 Message types with explanation of their meaning.

-UEM , CIM , CCM + DM + SCM , STRT , RM , RC.

3.4 Detailed description of CP structure.

-Further split of horizontal levels:
 --Event handling <--> response handling in UC/DC and SH
 --Line SH <--> build-up SH.
 --Link definitions between SH's.
 --Background of splits in the SH.
-Communication between both SH's.

3.5 Message flow in above mentioned structure.

-Base of message flow = ADB (packet address)
-Working of the gate.
-Handling of the gate outside the gate program itself (locking etc.)
-Interface with gate (GEME, CLME, RGAT).
-Direct and normal messages.

Philips' Telecommunicatie Industrie B.V. Hilversum The Netherlands Nederland
3522 000 71631

3.6 Character of application programs

-Re-entrancy (danger of program worklocations).
-Necessity of transaction data fields (ADB's, CR's, etc.) for data storage of dynamic call data.
-Danger of use of a program in more than one priority-level.
-Advantage of using 'scheduling in worklist' instead of 'scheduling in place to go tables'. (Effect of sheer mass of traffic on CP programs).

4 Call processing types and phases:

Available.

4.1 General definition.

-Transaction data fields seen as memory for the step by step situation in a call.
-Types for: What are we handling?
-States/phases for: Where are we?
-Types/states/phases mainly for distribution to modular program blocks. Indicators and flags for info used inside programs.

4.2 Types.

-Unit type
-Signalling type.
_Line type

4.3 Phases.

-Unit state.
-Signal handler phase.
-Call phase.

5 Implementation of the call processing structure.

Available.

5.1 Data structure.

-Msg/event distribution tables.
-Packets:
 -- ADB: Internal packets,
 -- CR: Use bits, internal packets,
 -- RSB, EP, ID, DSP
 -- Methods of definitions for symbols of data items.
-Data access/modification (IOM-CPM, symbolic).
 --Dedicated items, semaphore protection, Phase dependent use to be avoided.
-Soft data (only which data, not how defined by transaction)

5.2 Function of program components.

-Unit/Device controller:
 --Call allocation/de-allocation.
-Line SH and Build-up SH.
-Call coordination A and B (give a complete picture).
-Start handler.

-Relation with signalling specs.

Above functions must be explained with mentioning simple to understand messages (use flowcharts).

5.3 Recovery.

-Recovery levels
-Software initiation
 --Core clear
 --Pools chaining
 --Stable call saving (data collection).
-Hardware initiation.
 --Subsystems, units, network.
 --Stable call saving (not available hardware not to be released).
-Start scan / polling.

Philips' Telecommunicatie Industrie B.V. Hilversum The Netherlands Nederland

3522 000 71631

6 List of items with special implementation aspects.

Available: 6.1 till 6.8.

6.1 Definition of table outlets.

-How and where (program selectability influence).

6.2 I/O philosophy.

-I/O commands to different subsystems (SD, FD, PCS, MR).
-Restrictions to switching sequence and time delays.
-Hardware errors (returns).

6.3 Timing.

-Normal timeguards
-Quick timeguards.

6.4 Clearance of unit event data buffer (Flushing mechanism).

6.5 Unit handling.

-Unit conditions (serviceability, accessibility, service condition).
-Interface with configuration monitors (old, PPM)

6.6 Error detection with the aid of unit testpoints.

-Signalling conditions (Backward blocking, Forced release, Abnormal blocking).
-Transmission conditions (Transmission blocked).
-Description of unit testpoint error detection.
 --(Lamps ABLC, ITFR, OTBB)
 --ABLC message.
 -- Execution of the error detection.
-Re-allocation of call in error from CPM to IOM.

6.7 Interface with maintenance configuration monitors.

-Switch test interface.

6.8 LB/LC handling.

-LC event distribution.
-Function of Originating call initialiser and AJ/AJI initialiser.

6.9 Digit analysis.

-Initialisation in Signal handler.
-Source and traffic classes.
-Conclusions and relation with analysis building blocks.
-Junction group restriction (analysis conclusion not only decided by wanted destination, but also by the 'source' of the call).

6.10 Pad switching.

-Sequence

6.11 Scan delay/signalling interruption.

7 Practical solution in existing programs.

-SH E/M, pulse (not splitted signalling).
-Build-up SH, MFC.

Note: Make use of message tracer output.

8 SNETS.

-Structure and implementation.
-Coordinator, Test conf. handlers, Test signal handlers.

9 Method for program development.

-System description language (SDL) flow -> flowchart -> coding.
-Standards in volume 1050.

10 Other features (e.g. Surinam project group).

-Subscriber facilities
-AMA
-Traffic observation and call reporting.

11 Control tape aspects.

-Table definitions for application.
-Introduction of new facilities.

12 Management aspect of application testing and verification.

-Supply the 'Application manual'
(Contact similar application project groups)
-Project group support.
 --Administration.
 --Loadtape production.
 --Complaints.
-Supply testfloor / Trolley box.
 --Logbook.
 --Manuals.
 --Tapes / Cartridges.
-Real branch-test. (Use flowcharts!) and check special for the interfaces [ADB/CR].) This module test can be done in simulated environment.
-Integration test in real environment.

Literature.

DBA-7400e-2 System manual
1066/6/6
1066/10/40
Memo P. Kindseth about "Flushing mechanism" (800926)
Repair descr. compl 00-4842 (3522-272-32701)
Tos nucleus interface manual
Signalling specifications PTT administrations
Improved LC handling (1056)

Subsection 2: GENERAL LECTURE WITHOUT IMPLEMENTATION ASPECTS.

1 Goal of the call processing.

What do we want to achieve with our activities?

- We want to make a telephone call. The actual translation is: communicating with another person over a long distance. (For 'person' we can read nowadays also 'system').

- We could lay a wire-pair to that other person/system but when it is not used frequently it is better to share it with other people.

  In addition, we don't want to speak to the same person everytime.

- The solution is to make all the lines switchable. This is done by a so called telephone exchange.

  So: a telephone exchange connects one party to another. The party that takes the initiative to reach another party is called the calling party. The wanted party is the called party. To simplify the situation we call the first one A(-side) and the latter B(-side).

```
 !___!        _________         !___!
  _!_        !         !         _!_
 !   !_____!Telephone!_____!   !
 ! A !      !exchange !       ! B !
 !___!      !_________!       !___!
```

Diagram 2.1.

From the above picture we can derive that it is useful to make a functional split between A and B-side. The advantage of such a view is that any source can be connected to any destination.

By introducing the terms 'source' and 'destination' we have extended the situation.

By connecting two parties we don't want to be limited to one switching stage. Therefore one telephone exchange can be connected to other exchanges as well. To generalise the situation we just talk about source and destination, because for the structure of the system it is not important whether the parties belong to our exchange or not.

```
!___!    _________   .              .   _________    !___!
 _!_    !         !  .              .  !         !    _!_
!   !__!Telephone!__.              .__!Telephone!__!   !
! A !   !exchange !  \   _________  /  !exchange !  ! B !
!___!   !_________!  .\_!         !_/. !_________!  !___!
                     .   !Telephone!   .
          !___!      . _!exchange !_  .      !___!
           _!_       ./ !_________! \.        _!_
          !   !_____/                \____  !   !
          ! A !      .                 .    ! B !
          !___!      .                 .    !___!
                     .                 .
                     .                 .
          Sources                       Destinations
```

Diagram 2.2.

Source:
Other exchange: incoming trunk (IT)
Own exchange: originating subscriber line

Destination:
Other exchange: outgoing trunk (OT)
Own exchange: terminating subscriber line

2 Relation between hardware and software.

Our telephone exchange is an SPC-(stored program controlled-) exchange. The system is built up in two parts: Hardware and Software.

```
 ----------
!          !
! Hardware !
!----------!     Diagram 2.3.
! Software !
!__________!
```

To integrate both parts an interface between them must be defined. In general our philosophy about the system is that the software reflects the situation in hardware. Therefore changes in HW must be made known to the SW. Such a change we call an event and it is those events that trigger the software (N.B. An event is not an interrupt, supposed to be known).

```
       ----------
      !          !
      ! Hardware !
      !__________!
           !
   Events  !           Diagram 2.4.
          \!/
       ----!-----
      !          !
      ! Software !
      !__________!
```

We also want to cause changes in the HW. This is done via I/O commands for which I/O instructions have been defined.

```
           ----------
          !          !
          ! Hardware !
          !__________!
            !      !
   Events   !     /!\
           \!/     !  I/O Commands
           _!______!_
          !          !
          ! Software !
          !__________!     Diagram 2.5.
```

How do we detect a change in HW?

The software is waiting for interrupts from the hardware. Every 12,5 msec there is a Real Time Clock interrupt. At certain intervals (each interrupt) it will cause the software to issue a start of scan command to be issued. This means that a hardware device is triggered to look for changes in the hardware. When it has finished it will cause another interrupt: end-of-scan. This triggers the software to see what the changes were (stored in memory by the hardware device).

Now we turn to the SW. We see then that in practice a lot of the work to be performed does not depend on the wishes of a particular customer but is only dependent on our system philosophy. Therefore both parts have been split. The software which deals with the implementation of the system philosophy is called the Telephony Operating system (TOS). 'Telephony' because that is why we have developed the system concept. TOS makes is possible to operate and use the HW.

All the services TOS provides are described in so called TOS interface manuals. These enable different users to implement the wishes and specifications of customers (application) via a standard interface on top of TOS.

The hardware has been split into system hardware and application hardware. To operate the application hardware a hardware/software interface has been defined. This interface description can be found in the Appliction Hardware Manual or for new equipment in a volume document.
This hardware/software interface is reflected in the TOS interface manuals. Diagram 2.6 gives a picture from the different layers in hard- and software.

```
+------------------------------------------------------------------+
|  Peripheral hardware:       Telephony units (+ network)          |
|                                                                  |
|  ----------------HW/SW interface--------------------             |
|                                                                  |
|  Control hardware    :      Subsystems                           |
|                                                                  |
|==================================================================|
|                                                                  |
|  TOS software        :      Configuration management             |
|                             I/O handling                         |
|                             Services                             |
|                                                                  |
|  ---------------TOS/Appl. interface------------------            |
|                                                                  |
|  Application software:      Call processing                      |
|                                                                  |
+------------------------------------------------------------------+
```

Diagram 2.6.

The interface manuals give details of what the user has to provide in case he wants to make use of a certain TOS service.

TOS acts as a layer between the hardware and application.

Event distribution by TOS

It has been explained before that the autonomous scan registers changes in HW in the memory of the processor. TOS contains a program which is able to supply these events with identification parameters (unit nr, testpoint identity). TOS then passes the event on to application

3 Multicontrol aspects

The environment in which call processing must work is recorded in the figure below.

```
                 ----------------------
                 !                    !
                 ! Telephony network  !
                 !____________________!
                       !      !
                       !     /!\ I/O Commands
           Events     \!/     !
                     __!______!__
 -----------        !            !       -----------
 !         !        !   MCU =    !       !         !
 !  SCU1=  !        !            !       !  SCU2=  !
 !  CPM1   !        !IOM + CPM0  !       !  CPM2   !
 !_________!        !____________!       !_________!
 !         !        !            !       !         !
 !   PME   !        !    PME     !       !   PME   !
 !_________!        !____________!       !_________!
    !   !              !      !             !    !
    !  /!\             !     /!\            !   /!\
   \!/  !             \!/     !            \!/   !
 ___!___!______________!______!_____________!____!___
 !                                                  !
 !                       MS                         !
 !__________________________________________________!     Diagram 2.7.
                 !                    !
                 !                   /!\
                \!/                   !
 ________________!____________________!______________
 !          !   !            !   !           !
 !   PME    !   !    PME     !   !    PME    !
 !__________!   !____________!   !___________!
 !                                           !
 !                  CME                      !
 !___________________________________________!
```

A multi control system consists of a main control unit (MCU) and up to 2 secondary control units (SCU1, SCU2). All are connected with each other via the central memory. The multi control concept was necessary because of the processing capacity needed to handle all the work caused by hardware events. Only the MCU is connected to the telephony switching network and therefore called the IOM.

The principle is that an event is accepted in the I/O machine and the most time critical work is also done in this machine. The work that is not (or less) time critical can be given to another processor so that the IOM is available to handle new events. It is then also available to handle I/O requests.

We call the processor that does the not time critical work the call processing machine (CPM) because here actually telephony functions are performed.

It must be stressed that the CPM is not necessarily a different processor than the IOM. The call processing structure is such that each processor can act as a CPM

MCU = IOM + CPM0
SCU1 = CPM1
SCU2 = CPM2

It must be clear now that work for a call can be split between up to 3 processors and performed in parallel. This can cause problems if no precautions are taken. The practical solutions for call processing will be discussed later on.

Because there is only one processor connected to the telephony switching network, all commands to the hardware must be issued in the IOM. To avoid knowledge of which call processing machine is used, TOS provides I/O services that can be requested to issue a particular command. TOS will take care of transfer of the request, if necessary, to the IOM.

What normally is a "fast" I/O command will still be "fast" when it was requested in the MCU. It will be slow, however, when the request has issued in a SCU.
Whether a "slow" command (SD request) is requested from a SCU or the MCU makes no difference.

Subsection 3: APPLICATION (CALL PROCESSING) STRUCTURE.

1 Introduction

We have seen a functional split between the A- and B-side of a call. In addition each source and destination can have a different communication procedure with the exchange. Therefore it is necessary that CP programs are split into modular functional blocks. To obtain modular blocks we have introduced different levels in call processing. The aim is to translate events into telephony functions which are independent of the signalling procedures of the A- and B-side with the outside world.

The principle levels which we distinguish in application software are given in diagram 3.1. This structure is valid for A-side as well as for the B-side.

```
 --------------------------------------------------------------------
|                                                                    |
|     TOS                                                            |
|     ===========================================================    |
|     Application                                                    |
|                                                                    |
|             IOM      .------------.                                |
|                      |   Unit     |   Unit control                 |
|                      | controller |   (translation into            |
|                      `------------'    signalling events)          |
|                                                                    |
|             -------------------------------------------------      |
|                                                                    |
|             CPM      .------------.                                |
|                      |   Signal   |   Signal handling              |
|                      |   handler  |   (translation into            |
|                      |            |    telephony functions;        |
|                      `------------'    call handling events)       |
|                                                                    |
|                      .------------.                                |
|                      |    Call    |   Call handling                |
|                      |coordination|   (Telephony functions)        |
|                      `------------'                                |
|                                                                    |
|____________________________________________________________________|
```

Diagram 3.1.

In diagram 3.2 you will find a complete picture of the application structure.

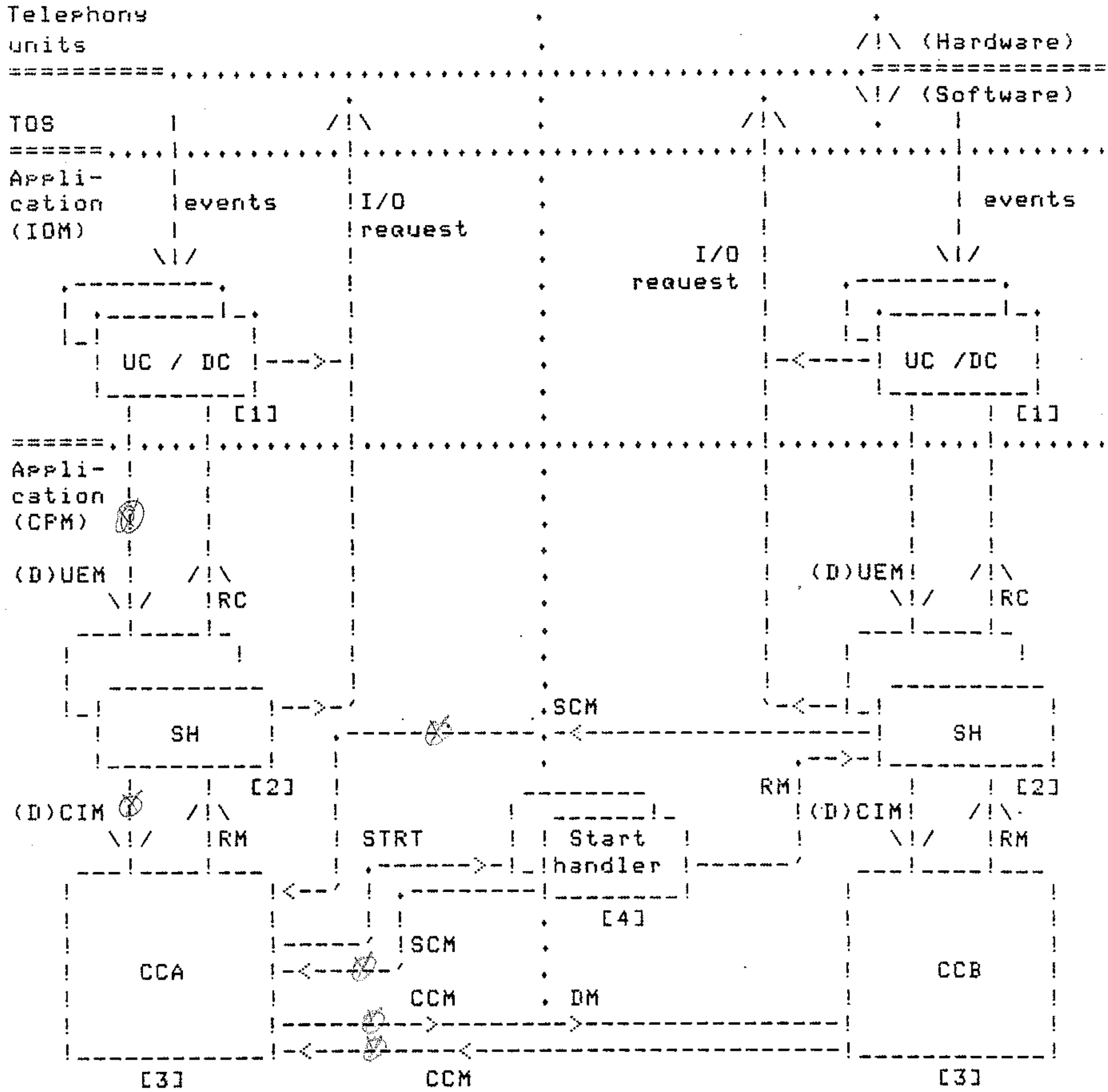

Diagram 3.2.

Remarks:

[1] : Time critical work. Translation of events into meaningful messages. Also for 'Initiation' and 'Blocking'.

[2] : Interpretation of signalling messages. Translation into general telephony functions.

[3] : Coordination task between A- and B-side.

[4] : The 'Starthandlers' are designed for initiating the link between A- and B-side.

⊗ signal via the GATE

Difference between UC and DC:

A device controller is used for equipment behind the new PCS subsystem, while a unit controller is related to a physical telephony unit in the 'old' PRX hardware. For each type of unit a unit/device controller exists and for each signalling procedure a SH is built.

2 Communication between different levels

A telephony unit has a limited number of testpoints. Thes testpoints reflect the state of the unit. The transitions int different states may result in testpoint values which occurre before. This means that we must process the events which belong t the testpoint changes, sequentially to interpret them in the correc way. At the same time it is necessary to process an event to do th time critical work. This work is done in the unit controller. B translating the event into a message, we can suspend the not tim critical work.
In fact a message is a code with additional information and i contained in one 16-bit word.
The message must be meaningful for the SH which has to process it This work can be done in the CPM. In the outlined configuratio (IOM + CPM), events will automatically be offered sequentially t application, because the IOM is part to the one and only MCU.
Delays in event handling must be avoided by translating th remaining work for an event as soon as possible into a 'message' s that another event can be handled. However, the suspended wor (suspended by producing a message) still must be handle sequenctially. To achieve this, the message is offered to a progra that can queue the message per call. This program is called th "gate". The exact working of the "gate" mechanism will be handle later on.

3 Message types with explanation of their meaning.

In the same way as a message is used to communicate from a UC/DC to a SH, the communication between other levels in also done with messages. These messages are offered to the gate as well. Thus the work for a message in execution can not be interrupted until this message has finished. The message types that have been defined to communicate between the levels are :

Unit event message (UEM): This message is generated in the UC to shift work to a SH as soon as time critical actions have been handled in the UC. An UEM is just an intermediate state between an event and a CIM.

Call input message (CIM): This message is generated in a SH to inform the call coordination to perform general telephony functions. At this level no signalling dependent info is sent.

Call coord. msg (CCM): This message is used to communicate between the A-side and B-side coordinator.

Start message: This message is used to initiate from call coordinator the other side of the call when its destination is known.

Some special CCM's have been defined:

Data message (DM): This msg has an additional function to a normal CCM: It can transfer data from one side to the other.

Start confirm. msg (SCM): This message is used to inform the call coordinator which has issued a start msg about the result of the start.

Remark: After issueing a start msg the coordinator will always wait for a SCM before allowing the next message to be handled.

The following two 'messages' take care of the communication in the opposite direction of the hierarchy.

Response msg (RM): This message is issued from a coordinator to a SH when some signalling dependent work for a telephony function must be done. This message is only offered for distribution to the appropriate SH and therefore will be handled immediately. An RM is handled as reaction on a message that has "locked" the gate. They are therefore only distributed by the gate.

Response command (RC): A response command is used for the communication from the SH to a UC. In this way a UC can be requested to perform a function that is the priveleage of the IOM. Both RC and RM must return to the originator to continue the message they were handling. So RC and RM act like a subroutine (but can give up control).

When we look at our picture, we see that certain messages cross the border between IOM and CPM. These messages are UEM and RC. When IOM and CPM are not in the same processor, these message must be transferred from one processor to the other. Therefore these msg are offered to a TOS service that will take care of this transfer.

4 Detailed description of CP structure

From the discussion about the messages you can see that there is a two-way communication on the different levels. Therefore internally UC and SH have been split into an event handling part and a response handling part (diagram 3.3).

```
                  !                 .
                  !                /!\
           Events!                 ! I/O requests
               \!/       .---------->!
             __!______!__________!__
            !              .          !
            ! Event        .  Response !     UC / DC
            ! handling     .  handling !     =======
            !_________________________!                    .
                  !                                        /!\
                  !                 .                       !  IOM
  ..........!............... /!\ ...................=====
         UEM !                 ! RC                         !  CPM
               \!/       .---------->!                     \!/
             __!______!__________!__                        .
            !              .          !
            ! Event        .  Response !     SH
            ! handling     .  handling !     ==
            !_________________________!
                  !            !
                  !<----------'      /!\
            CIM !                 .  ! RM
               \!/                   !
             __!____________________!__
            !                          !
            !            CC            !
            !                          !
            !__________________________!
```

Diagram 3.3.

Due to a response command the unit controller can send the signal handler some information by generating a UEM. The same is valid for response messages. These can generate new call input messages to arrange work to be done, which does not belong to the task of a SH. (Responsibility of the call coordinators).

Line SH and Build-up SH.

To set up a connection we have to inform the exchange of our wishes. The used communication procedure is called a signalling procedure. A signalling procedure can be split into 2 parts:

1. Signals that are sent to establish and break-down a communication path with our own exchange of with another exchange. These are the so called line signals.

2. Signals that are sent to give information about the destination we want to reach (routing info). They form the so called build-up signals.

It is possible to use line signals to send build-up information. In that case we make use of the length of the signals. Because you need a lot of changes to transfer your information it is a slow system. To speed up the build-up process new units have been made which are used only for (special) build-up signals. Another advantage of such a special unit is, that it is only needed for a small portion of the total call time. After finishing the build-up procedure, the unit can be used for another call set-up. If in the signalling procedure as well as in the hardware there is a functional split between both parts, why shouldn't we do that in software?

Therefore in such a case the signal handler is split into a part that handles the line and a part that handles the build_up signals. Because the function has been split in hardware we already need two unit controllers (for each different type of unit a unit controller is necessary). The special build up unit is called a sender or receiver.

Some examples of line signals:
-Seizure
-Seizure acknowledgement
-Answer
-Forward/Backward release
-Blocking

Some examples of build-up signals:
-Digits for routing or identification
-Category

The following picture arises (diagram 3.4.):

Philips' Telecommunicatie Industrie B.V. Hilversum The Netherlands Nederland
3522 000 71631

```
                 ----------               ----------
      ------! Line unit !------------! Build-up  !
            !           !            !   unit    !
            !___________!            !___________!
                 !                         !                 HW
 ................!.........................!............======
                 !    !                    !     !           SW
                 !   /!\                   !    /!\
          Events!    ! I/O           Events!    ! I/O
             \!/     ! requests         \!/     ! requests
            __!_____!__               __!_____!__
            !   UC      !             !   UC      !
            !  line     !             ! build-up  !
            !  unit     !             !   unit    !
            !___________!             !___________!
                 !    !                    !     !
            UEM  !   /!\              UEM  !    /!\
             \!/     !RC                \!/     !RC
            __!_____!__               __!_____!__
            !           !             !           !
            !   Line    !------------>! Build-up  !
            !    SH     !             !    SH     !
            !___________!<------------!___________!
                 !    .                    !     .
                 !   /!\                   !    /!\
            CIM  !    ! RM            CIM  !     ! RM
             \!/      !                 \!/      !
            __!_____!_________________!_____!__
            !                                    !
            !                CC                  !
            !                                    !
            !____________________________________!
```

Diagram 3.4.

Because the line and build-up signal handler form an integral part of the signalling procedure, some communication is necessary between both. This is done via so called function codes. The implementation will be discussed later. The configuration which is outlined above, gives now the possibility to connect any build-up to any line SH.

Example:
MFC build-up which is used in combination with different line signalling procedures. To connect a line and a build-up signal handler we have to declare certain data definitions. These definitions are called the link definitions. Different ways of implementation cause that these definitions cannot always be found in a single component.

5 Message flow

It may not have been very clear how the generated messages reach their destination and how they are connected to one call. At this stage of the course we want to go into more detail about this phenomenon.

5.1 How are messages bound to a particular call?

Each telephony unit is assigned a packet. This packet is called for normal trunk units Associated Data Block (ADB) and for the special build-up units Receiver/Sender Data Block (RSB).

The ADB is used as the base for a call. In this packet data belonging to the call is stored.

A call uses an A- and B-side unit ; so there are two ADB's in a complete call.

When a message is generated and offered to the gate program, the ADB is always supplied. This procedure ensures that different calls can be distinguished as concurrent processes.

For the B-side unit an ADB is fetched from a pool at the moment that the B-side is started.

For A-side units and build-up units a transaction data block is always present.

TOS provides a table which contains the references to the transaction data blocks. This table PEFXDT88 is indexed with the unit number. TOS stores the references for A-side units and build-up units, while application has to do this for the B-side units.

5.2 Working of the gate.

The gate is a TOS program that runs in the CPM.
The main functions of the gate are (per process):

- queuing of messages
- ensuring sequential processing of messages (FIFO)
- distribution of messages

When a message is offered to the gate it will check whether a message is already handled. The ADB contains an indicator (the lock bit - LCK) which in that case is set.

If the gate is locked the message will be queued by storing it at a certain place in the ADB and the gate will either return to the user of return to the MCP (depending on the message type and whether a return link is specified or not). When more than one message has to be queued a packet pool is available from which packets (EP's) can be obtained to store the message. These packets are chained to the ADB (see diagram 3.5.).

```
.-----------.
|    ADB    |
|           |
|           |
|-----------|
|  message  |
|-----------|               .----------.              .----------.
|  pointer  |------->|     EP     |------->|     EP     |
|-----------|               !__________!              !__________!
|           |
|           |
|           |                          Diagram 3.5.
!___________!
```

When a message has been executed, it must signal this back to the gate. The gate will look whether another message is available in the queue and schedule this message to be distributed to the appropriate program. In this case the gate stays locked. When the last available message has finished and signals this back to the gate the gate will unlock. Any other message offered will lock the gate again and be distributed immediately.

Normal UEM, CCM and CIM's are handled in this way. A start message is not handled by the gate. It is a direct jump to a start handler. However, a start message must be issued when the gate is locked because no interference is allowed until the other side has been initiated.

A start message is the result of handling a CIM.

The B-side must signal back that it has finished with the initiation. It will also indicate by the message name whether it was successful or not. The signalling back is done with a start confirmation message. This message is offered to the gate at a special entry where it will immediately be distributed as a call coordination msg.

The gate is still locked by the CIM that caused the start message. When the SCM has been processed message handling is allowed again at that side of the call.
Response messages are only distributed by the gate. Data messages are handled in the same way as a CCM, except that in this case a transmission packet (just to contain the data to be transferred to the other side) is involved.

A special message that exists is CIM 'abort'. The gate generates this message if there are no event packets available anymore (loss of info).

An abort message gets a special treatment. When it is offered to the gate and the gate is locked, all queued messages are removed and this message is the next one to be handled. The current message will be allowed to finish. Messages offered to the gate for queuing after an abort message has been received, will be ignored.

The gate offers some possibilities to initiate and terminate message handling.

- unlock the gate and remove all messages.
- lock the gate and remove all messages.
  (here the removal is optional)

Use of these features is made when a call is started and when it is ending. This makes it possible to avoid messages which are not wanted anymore by application.

When a message is offered to the gate it may be possible that the user wants back control after the message has been handled. For this purpose the gate needs some extra storage space which is found in a Call Register (CR). This is an extended data packet for the call needed during transitional periods.
When the gate cannot obtain a CR it will place the ADB in a 'wait for CR' queue. The gate will continue only for this call when a CR has become available.

The specified return is stored in the CR. When the call is locked the gate will store the message and use the return link to get back to the user. However, when the call is not locked a schedule to another entry in the gate has to be done to distribute the message and at the immediate return point to return to the user.

In the CR a 'CR in use' field has been defined. If this field is empty, the gate will return the CR to its pool.

It must be kept in mind that the basic function of the 'Gate' is: Regulating the actions for a call in such manner, that use of

"Work-fields" in ADB, RSB, or CR is limited to one action.

If for instance a certain field would be in use for a slow driver command and the gate would accept and distribute a message, that would "unknowingly" use the same field for a time delay action, the control chains of the slow driver service and the time service would be snarled up in a nasty way.

In general : All events must lock the gate when they are handled. RM's however are the result of another message, that locked the gate already, and must not "double lock" the gate! RC's are in the same situation , or are the result of RM's.

5.3 Direct and normal messages.

The difference between direct and normal messages is that direct messages are not offered to the gate and normal messages are. The direct messages are only distributed. Two types of direct messages exist: DUEM's and DCIM's. They are used when no interference can take place between a current message or action and the handling of the direct message. DUEM's are used in two ways:

- Connection of UC and SH as 'one' program. In this case it is possible to shift work to the CPM. This work is done without gate protection.

- Information transfer from the UC to the SH while the gate is locked. There are two reasons for this:
  A. Important information must be passed on to the SH.
  B. Communication between SH and UC where the signal handler phase is used to guide the "answer". The return of a response command is a DUEM in this case (see 'flushing mechanism').

DCIM's can be used when you want to be sure that this message is handled or when you are sure that it will not disturb other messages. This is signalling type dependent. The distribution of DCIM's happens in the same way as for normal CIM's.

Remark: The transfer of a UEM via the UEM buffer takes time. The buffer works on a FIFO basis, so that there is no difference between direct and normal messages as far as the transport time to the CPM is concerned.

5.4 Interface with the gate (provided by TOS)

How and where are messages generated?

DUEM: These messages can be generated in two places:

1. Unit controller
2. Signal handler

ad 1)

When a UEM is generated in the unit controller it must be done by a macro call UEME. This is a call to the UEM reception which will store it in the unit event msg buffer to be transferred to the CPM. The UEM buffer is only used when the CPM is another processor than the IOM. If the CPM is the same processor as the IOM the UEM is offered to the gate immediately. When no return is wanted control to MCP will be given. When a return is specified the user must give up control to MCP a.s.a.p. to enable the event distributor to continue with the next event.
DUEM's can only be generated in the IOM with the macro DUEM. It is not allowed to specify a return. The distribution is done in the sameway as normal UEM's.

ad 2)

When a UEM is generated in a SH this is normally done to come under the gate but is also allowed if additional work has to be done within the signal handler. This message must be generated by a macro call GEME. A return is allowed.

Diagrams 3.6a. and 3.6b. give a survey of the possibilities for UEM generation.

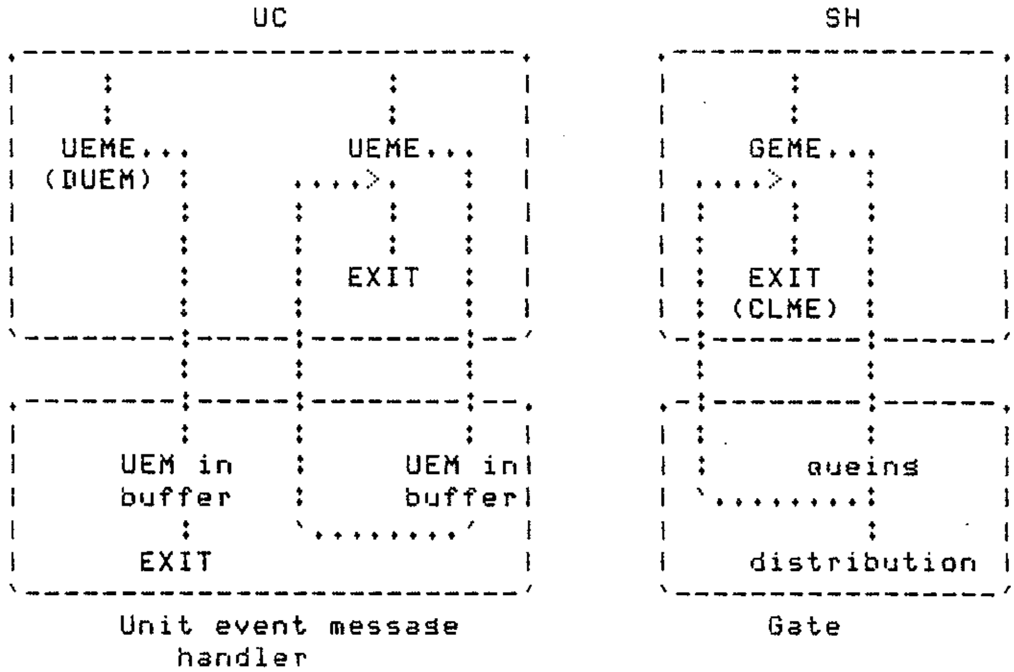
UC
SH
UEME...
(DUEM)
UEME...
EXIT
GEME...
EXIT
(CLME)
UEM in buffer
EXIT
UEM in buffer
queing
distribution
Unit event message handler
Gate

Diagram 3.6a.

```
                          UC
 .-------------------------------------------------------.
 |    :            :                     :                |
 |    :            :                     :                |
 | UEME...      DUEM...               UEME...             |
 |         :            :        ....>.       :           |
 |         :            :        :     :      :           |
 |         :            :        :     :      :           |
 |         :            :        :   EXIT     :           |
 |         :            :        :            :           |
 `---------:------------:----:--------------:--'
           :            :        :            :
 .---------:------------:----:--------------:--.
 |         :            :        :            :           |
 |     queuing      distri-     :        queuing|
 |     and/or       bution      :             :           |
 |    scheduling                 `.........:              |
 |  for distrib.                              :           |
 |         |                                distr.|
 |       EXIT                                             |
 `-------------------------------------------------------'
                        Gate
```

Diagram 3.6b.

Resp. command A response command is not a real message. It does not make use of the gate at all. It is generated by the macro call GRCM. It is offered to the response command handler, which will transfer it to the IOM. To return back to the signal handler normally you will call the macro RRCM. RC's are given while the gate is locked.

Start msg A start message is generated by the macro call STRT, which results in a direct jump to the required start handler. It must be done while the gate is locked.

Other messages CIM, DM, CCM and SCM's are generated with the macro call GEME. They are handled by the gate in the previously described way. A SCM must have a return specified to de-couple A and B-side of the call. The other messages need a return when they are generated while the gate is locked to unlock it. DCIM's may only be generated with macro GEME as a direct result of UEM handling in the SH. The gate must still be locked by this UEM. A return is not allowed.

Philips' Telecommunicatie Industrie B.V. Hilversum The Netherlands Nederland

3522 000 71631

Response msg A response msg must be generated by a macro call GEME and when it arrives in the gate, it will be distributed to the appropriate SH immediately.

Return to the gate:

After execution of a message by the program to which it was distributed it is necessary to inform the gate. Because there are 2 types of messages (and which make use of the gate):

A. Message that after generation allows the generating program to get back control immediately.

B. Message that after generation allows the generating program to get back control after the message has been executed,

there are also 2 ways of return.

ad A. Use the macro RGAT (return to gate). This is valid for UEM, DM CCM, CIM, SCM's.

ad B. Use the macro CLME (close message). This is valid for RM's

Remarks:

1. This difference in return may cause problems when it is not very clear which type of msg initially started the process.

2. A message that is allowed to be generated without a return will be handled in the proper way by the gate.

Diagrams 3.7 and 3.8 give an overview of the possible message flows.

```
Handling of messages in cooperation with the gate.
--------------------------------------------------

Messages distributed to SH:
---------------------------

UEM      UEM     UEM     DUEM     DUEM     DUEM     DUEM
 |        |       |       |        |        |        |
 |        |       |       |        |        |        |
RGAT     DCIM    CIM     EXIT     CIM      CIM      UEM
                  |                         |
                 RGAT                      RGAT

Messages distributed to CC/SH:
------------------------------

(D)CIM/CCM    (D)CIM/CCM       (D)CIM/CCM        (D)CIM/CCM
    |             |                |                 |
    |             |                |                 |
  CIM/CCM       CIM/CCM          CIM/CCM             |
    |             |                |                 |
    |             |                |                 |
  RGAT          RM.........RM    RM..........RM     RM.......RM
                            |                 |               |
                            |                 |               |
                            |                 |               |
       (return)........CLME                 CLME            CLME
              |                           (=RGAT)         (=RGAT)
              |
            RGAT
```

Diagram 3.7.

```
 .-----------------------------------------------------------------------.
 |           CC-A              START HANDLER           SH B-side         |
 |                             A-side | B-side                           |
 |                                    |                                  |
 |  CIM                               |                                  |
 |   :                                |                                  |
 |   :                                |                                  |
 |  STRT...................STRT       |                                  |
 |                           :        |                                  |
 |                           :*       |                                  |
 |                           :        |                                  |
 |                         CLCK       |                                  |
 |                       (B-side)     |                                  |
 |                           :...........(schedule)                      |
 |                                    |       :                          |
 |                                    |       :**                        |
 |                                    |       :                          |
 |                                    |     RM (B-side)......RM 'initiate'|
 |                                    |                       :          |
 |                                    |                       :          |
 |          SCM...................................................SCM   |
 |           :                        |                       :          |
 |           :                        |                       :          |
 |          RGAT                      |                   CLME(B-side)   |
 |                                                                       |
 |*  Reserve B-ADB and CR, copy info from A-ADB and CR                   |
 |** Search available B-unit and reserve speech paths                    |
 `-----------------------------------------------------------------------'
```

Diagram 3.8.

NB The gate has a special entry to receive a SCM in case no B-side could be found (no B-side ADB).

6 Character of application programs.

From the previous explanation it must be clear, that calls offered to the system form concurrent processes. Therefore program worklocations can be used by different processes. This can happen when the program is temporarily interrupted to perform another task. At this moment another process is allowed to use the program and can overwrite the worklocations. After return to the program these location can only be used when restored with own data. Because there is a lot of data to be kept with a process we use transaction data control fields (data area for one process only).

These fields are already mentioned before. They are ADB, CR, RSB. Other ones with more dedicated tasks will be explained later on.

Another characteristic, which needs attention, is the different priority levels application makes use of. When a schedule in batch level is performed, the program can be re-entered on real time level. In that case worklocations can have been destroyed before a schedule back to real time level is done.

When schedules have to be done it is advised to make use of worklists instead of place to go tables. In the first case you offer your own worklocations and in the latter case you make use of declared worklocations and therefore can overflow when a lot of calls are offered to the system. This also means that you can loose information and the system will crash.

Subsection 4: CALL PROCESSING TYPES AND PHASES.

1 General definition.

The existence of transaction data fields has been explained before. They are keeping data that belongs to the process (a call). One kind of data that is stored in an ADB or RSB relates to the situation in which the call is. We have noted that each telephony unit has an ADB or RSB connected to it. Because not all telephony units are of the same type and serve the same signalling procedure we have to record this difference in the ADB or RSB. So the types in the ADB tell us what we are handling. Also we have seen before that events along the signalling procedure may appear to look the same; But it is also true that messages at a certain point in a call have a different impact or are not wanted. Therefore you will find states and phases in the ADB so that we know where we are. Therefore types/states and phases have a practical impact as well. They are used to distribute messages and functions to the appropriate programs.

Another way of knowing where you are is by using indicators or flags. This is impractical when distribution must take place by general programs, but is useful when applied within a program or program area.

2 Types

The types that we are using in call processing are:

2.1 Unit types

Unit types exist in hardware and software. A hardware unit type is associated with a physical unit. Software unit types make it possible to use units which are identical in hardware in a different way (e.g. for a different signalling procedure).
Each unit type has a set of characteristics and when we assign a type to a unit we determine the possibilities of that unit.
To each unit type a unit controller belongs, which can handle events and I/O commands on units of this type. The unit event and response command distributor will use the unit to distribute an event.

2.2 Signalling type

Each signalling procedure has assigned a signalling type. In case the signalling procedure is splitted into a line and build-up part the signalling type will be the same for both. An extra indicator has been introduced to indicated that the build-up signalling is active. To each signalling type belongs a signal handler or signal handler pair (SH-L + SH-B). Distributors that make use of the signalling type are:
Response message distributor
Unit event message distributor.

2.3 Line type

A line type is assigned to each subscriber line. Such a line needs a separate type because bothway traffic can be possible on such lines. To obtain the correct signalling type this type must be translated dependent on the way the line is used (originating or terminating). The line type will not be used for distribution, but the signalling type belonging to it.

3 Phases

To know where we are in the signalling procedure and hardware we have introduced states and phases:

3.1 Unit state

A unit state is used in a unit controller to keep in phase with the state of the unit. It is a subdivision of the unit type and will be used to distribute a unit event (the same event can have a different meaning in a different state).

Note: RC's are not distributed on unit state.

3.2 Signal handler phase

Within a signal handler it is useful to interpret UEM's according to the phase of the signalling procedure. Not relevant messages can be ignored easily. It is a subdivision of the signalling type and will be used by the UEM distributor.
Note: RM's are not distributed on SH-phase.

3.3 Call phase

At coordination level we are only interested in telephony functions. Because these functions have a particular meaning at certain points in the process we have divided the call in certain phases. Logically we use the name call phase. CIM's, CCM, DM, SCM's distributors use this phase to distribute these messages to the call coordination.