



This electronic version (PDF) was scanned by the International Telecommunication Union (ITU) Library & Archives Service from an original paper document in the ITU Library & Archives collections.

La présente version électronique (PDF) a été numérisée par le Service de la bibliothèque et des archives de l'Union internationale des télécommunications (UIT) à partir d'un document papier original des collections de ce service.

Esta versión electrónica (PDF) ha sido escaneada por el Servicio de Biblioteca y Archivos de la Unión Internacional de Telecomunicaciones (UIT) a partir de un documento impreso original de las colecciones del Servicio de Biblioteca y Archivos de la UIT.

(ITU) للاتصالات الدولي الاتحاد في والمحفوظات المكتبة قسم أجراه الضوئي بالمسح تصوير نتاج (PDF) الإلكترونية النسخة هذه والمحفوظات المكتبة قسم في المتوفرة الوثائق ضمن أصلية ورقية وثيقة من نقلًا.

此电子版（PDF版本）由国际电信联盟（ITU）图书馆和档案室利用存于该处的纸质文件扫描提供。

Настоящий электронный вариант (PDF) был подготовлен в библиотечно-архивной службе Международного союза электросвязи путем сканирования исходного документа в бумажной форме из библиотечно-архивной службы МСЭ.



INTERNATIONAL TELECOMMUNICATION UNION

# CCITT

THE INTERNATIONAL  
TELEGRAPH AND TELEPHONE  
CONSULTATIVE COMMITTEE

**YELLOW BOOK**

---

**VOLUME VI - FASCICLE VI.7**

## **FUNCTIONAL SPECIFICATION AND DESCRIPTION LANGUAGE (SDL)**

**RECOMMENDATIONS Z.101-Z.104**

## **MAN-MACHINE LANGUAGE (MML)**

**RECOMMENDATIONS Z.311-Z.341**

---



**VII<sup>TH</sup> PLENARY ASSEMBLY**  
GENEVA, 10-21 NOVEMBER 1980

Geneva 1981



## A NOTE FROM ITU LIBRARY & ARCHIVES

---

Due to technical restrictions, the template of SDL symbols has not been included in the scanned version of this document.

\*\*\*\*\*

En raison de contraintes techniques, le gabarit de symboles du LDS n'a pas été inclus dans la version scannée de ce document.

\*\*\*\*\*

Debido a restricciones de técnicas, la plantilla de símbolos del LED no se ha incluido en la versión escaneada de este documento.



INTERNATIONAL TELECOMMUNICATION UNION

# CCITT

THE INTERNATIONAL  
TELEGRAPH AND TELEPHONE  
CONSULTATIVE COMMITTEE



**YELLOW BOOK**

---

**VOLUME VI - FASCICLE VI.7**

## **FUNCTIONAL SPECIFICATION AND DESCRIPTION LANGUAGE (SDL)**

**RECOMMENDATIONS Z.101-Z.104**

## **MAN-MACHINE LANGUAGE (MML)**

**RECOMMENDATIONS Z.311-Z.341**

---



**VII<sup>TH</sup> PLENARY ASSEMBLY**

GENEVA, 10-21 NOVEMBER 1980

Geneva 1981

ISBN 92-61-01111-X



**CONTENTS OF THE CCITT BOOK  
APPLICABLE AFTER THE SEVENTH PLENARY ASSEMBLY (1980)**

**YELLOW BOOK**

- Volume I**
- Minutes and reports of the Plenary Assembly.  
Opinions and Resolutions.  
Recommendations on:
    - the organization and working procedures of the CCITT (Series A);
    - means of expression (Series B);
    - general telecommunication statistics (Series C).List of Study Groups and Questions under study.
- Volume II**
- FASCICLE II.1 - General tariff principles - Charging and accounting in international telecommunications services. Serie D Recommendations (Study Group III).
- FASCICLE II.2 - International telephone service - Operation. Recommendation E.100 - E.323 (Study Group II).
- FASCICLE II.3 - International telephone service - Network management - Traffic engineering. Recommendations E.401 - E.543 (Study Group II).
- FASCICLE II.4 - Telegraph and "telematic services"<sup>1)</sup> operations and tariffs. Series F Recommendations (Study Group I).
- Volume III**
- FASCICLE III.1 - General characteristics of international telephone connections and circuits. Recommendations G.101 - G.171 (Study Group XV, XVI, CMBD).
- FASCICLE III.2 - International analogue carrier systems. Transmission media - characteristics. Recommendations G.211 - G.651 (Study Group XV, CMBD).
- FASCICLE III.3 - Digital networks - transmission systems and multiplexing equipments. Recommendations G.701 - G.941 (Study Group XVIII).
- FASCICLE III.4 - Line transmission of non telephone signals. Transmission of sound programme and television signals. Series H, J Recommendations (Study Group XV).
- Volume IV**
- FASCICLE IV.1 - Maintenance; general principles, international carrier systems, international telephone circuits. Recommendations M.10 - M.761 (Study Group IV).
- FASCICLE IV.2 - Maintenance; international voice frequency telegraphy and facsimile, international leased circuits. Recommendations M.800 - M.1235 (Study Group IV).
- FASCICLE IV.3 - Maintenance; international sound programme and television transmission circuits. Series N Recommendations (Study Group IV).
- FASCICLE IV.4 - Specifications of measuring equipment. Series O Recommendations (Study Group IV).

---

<sup>1)</sup> "Telematic services" is used provisionally.

**Volume V** – Telephone transmission quality. Series P Recommendations (Study Group XII).

**Volume VI**

- FASCICLE VI.1 – General Recommendations on telephone switching and signalling. Interface with the maritime service. Recommendations Q.1 - Q.118 *bis* (Study Group XI).
- FASCICLE VI.2 – Specifications of signalling systems Nos. 4 and 5. Recommendations Q.120 - Q.180 (Study Group XI).
- FASCICLE VI.3 – Specifications of signalling system No. 6. Recommendations Q.251 - Q.300 (Study Group XI).
- FASCICLE VI.4 – Specifications of signalling systems R1 and R2. Recommendations Q.310 - Q.480 (Study Group XI).
- FASCICLE VI.5 – Digital transit exchanges for national and international applications. Interworking of signalling systems. Recommendations Q.501 - Q.685 (Study Group XI).
- FASCICLE VI.6 – Specifications of signalling system No. 7. Recommendations Q.701 - Q.741 (Study Group XI).
- FASCICLE VI.7 – Functional Specification and Description Language (SDL). Man-machine language (MML). Recommendations Z.101 - Z.104 and Z.311 - Z.341 (Study Group XI).
- FASCICLE VI.8 – CCITT high level language (CHILL). Recommendation Z.200 (Study Group XI).

**Volume VII**

- FASCICLE VII.1 – Telegraph transmission and switching. Series R, U Recommendations (Study Group IX).
- FASCICLE VII.2 – Telegraph and “telematic services”<sup>1)</sup> terminal equipment. Series S, T Recommendations (Study Group VIII).

**Volume VIII**

- FASCICLE VIII.1 – Data communication over the telephone network. Series V Recommendations (Study Group XVII).
- FASCICLE VIII.2 – Data communication networks; services and facilities, terminal equipment and interfaces. Recommendations X.1 - X.29 (Study Group VII).
- FASCICLE VIII.3 – Data communication networks; transmission, signalling and switching, network aspects, maintenance, administrative arrangements. Recommendations X.40 - X.180 (Study Group VII).

**Volume IX** – Protection against interference. Series K Recommendations (Study Group V). Protection of cable sheaths and poles. Series L Recommendations (Study Group VI).

**Volume X**

- FASCICLE X.1 – Terms and definitions.
- FASCICLE X.2 – Index of the Yellow Book.

---

<sup>1)</sup> “Telematic services” is used provisionally.

## CONTENTS OF FASCICLE VI.7 OF THE YELLOW BOOK

### Part I – Recommendations Z.101 to Z.104

#### Functional specification and description language (SDL)

Rec. No.		Page
Z.101	1. General explanation of the specification and description language (SDL) . . . . .	3
Z.102	2. Symbols and rules . . . . .	6
Z.103	3. Use of pictorial elements within state symbols . . . . .	8
Z.104	4. Semantics . . . . .	17

### Part II – Recommendations Z.311 to Z.341

#### Man-machine language (MML)

##### SECTION 1 – *General principles*

Z.311	1. Introduction . . . . .	75
Z.312	2. Basic form layout . . . . .	77
Z.313	3. The meta-language for describing the syntax and procedures . . . . .	78
Z.314	4. The character set and basic elements . . . . .	79
Z.315	5. Input (command) language syntax specification . . . . .	86
Z.316	6. Output language syntax specification . . . . .	92
Z.317	7. Man-machine dialogue . . . . .	100
Z.318	8. List of functions . . . . .	114

SECTION 2 – <i>Specification of functions</i> . . . . .	119
---	-----

SECTION 3 – <i>Users manual</i> . . . . .	119
---	-----

##### SECTION 4 – *Glossary of terms*

Z.341	Glossary of terms . . . . .	121
-------	-----------------------------	-----

SECTION 5 – <i>Implementors guide</i> . . . . .	135
---	-----



REMARK

The Questions entrusted to each Study Group for the Study Period 1981-1984 can be found in Contribution No. 1 to that Study Group.

---

CCITT NOTE

In this Fascicle, the expression "Administration" is used for shortness to indicate both a telecommunication Administration and a recognized private operating agency.

**PART I**

**Recommendations Z.101 to Z.104**

**FUNCTIONAL SPECIFICATION AND  
DESCRIPTION LANGUAGE (SDL)**



## Recommendation Z.101

### 1. GENERAL EXPLANATION OF THE SPECIFICATION AND DESCRIPTION LANGUAGE (SDL)

This Recommendation deals with the presentation of the functional specification and of the description of the internal logic processes in stored-programme control (SPC) switching systems.

#### 1.1 *Introduction*

##### 1.1.1 *Methods of presentation*

The methods of presentation of functional specifications and of descriptions of internal logic processes in SPC switching systems can be subdivided into the following categories:

- narrative methods (natural language and numerical information supported by drawings and lists, etc.);
- formalized presentation methods.

The narrative methods, which can be used to a large extent for both specifications and descriptions of SPC switching systems, need no standardization by the CCITT.

Considering the formalized methods of presentation, the subject of this Recommendation is a graphical method SDL/GR, based on state transition diagrams, using the symbols and rules of the Functional Specification and Description Language (SDL) described below. (It may be noted that some processes of an SPC switching system may require specifications and/or descriptions by methods other than in this Recommendation.) There is also a programme-like form of the SDL (SDL/PR) under study.

##### 1.1.2 *General objective*

The objective of the SDL is to provide a standardized method of presentation:

- that is easy to learn, to use and to interpret in relation to the needs of operation organizations;
- that provides unambiguous specifications and/or descriptions for tendering and ordering;
- that provides the capability of meaningful comparisons between competitive types of SPC switching systems;
- that is open-ended to be extended to cover new developments.

##### 1.1.3 *Area of application*

The main area of application covers all types of SPC switching systems. Within these systems the following functions are included amongst others:

- call processing (e.g. call handling, routing, signalling, metering, etc);
- maintenance and fault treatment (e.g. alarms, automatic fault clearing, configuration control, routine tests, etc.);
- system control (e.g. overload control, modification and extension procedures, etc.).

SDL is also applicable to a wider range of systems.

#### 1.2 *Framework*

##### 1.2.1 *Basic definitions*

- a) To clarify the meaning of terms used in the SDL, a number of definitions are given below.
- b) Some of the terms defined below have been in use in other fields and will have connotations related to those fields. Care should therefore be exercised by those concerned with the SDL that their use and understanding of such terms is in accord with the definitions contained in this section.

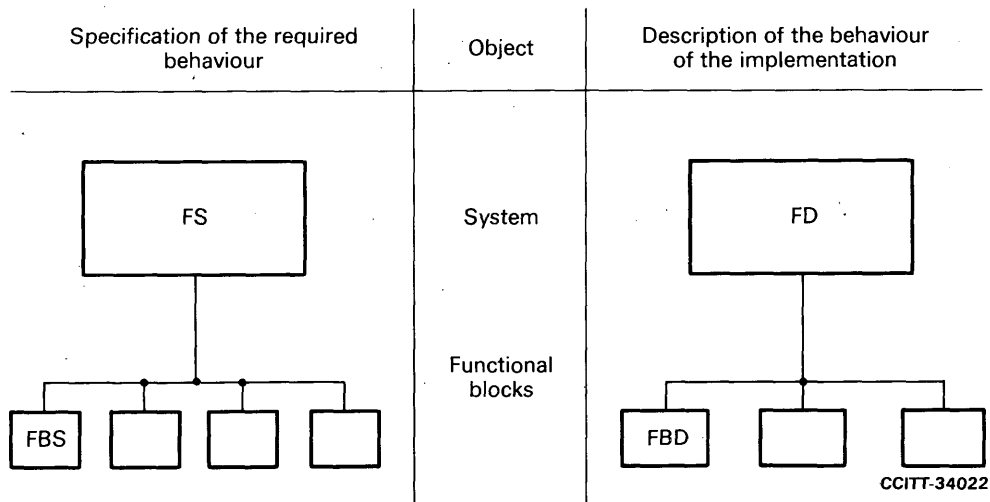
1.2.2 *Specifications and descriptions*

- a) The requirements of a system are defined in a *specification* of that system and the implementation of those requirements is described in a *description* of the system.
- b) Both specifications and descriptions consist of two parts: *Specifications* consist of general parameters required of the system and the functional specifications (FS) of its required behaviour. *Descriptions* consist of general parameters of the system as implemented and the functional description (FD) of its actual behaviour.
- c) The *general parameters* in both cases relate to such matters as temperature limits, transmission limits, construction, exchange capacity, grade of service, etc.

1.2.3 *Functional specifications and functional descriptions* (See Figure 1/Z.101)

The *functional specification* (FS) of a system is a specification of the total functional requirements of that system from all significant points of view.

The *functional description* (FD) describes the actual behaviour of the implementation of those functional requirements in terms of the internal structure and logic processes within the system.



FS = functional specification  
 FD = functional description  
 FBS = functional block specification  
 FBD = functional block description

*Note* – The partitioning of an FS into FBSs for a particular system does not necessarily correspond to the partitioning of the FD into FBDs for this same system.

FIGURE 1/Z.101  
**Partitioning**

1.2.4 *Functional block specification and functional block description* (See Figure 1/Z.101)

A *functional block* is an object of manageable size and relevant internal relationship, containing one or more processes.

A *functional block specification* specifies the required behaviour of the one or more processes within a functional block.

A *functional block description* describes the means by which the required behaviour of processes within a functional block is achieved.

### 1.2.5 *Process specification and process description*

A *process* performs a logic function that requires a series of information items to proceed, where these items become available at different points in time.

The behaviour of a process is specified in a *process specification*, and is described in a *process description*, in terms of inputs, saves, states, transitions, decisions, tasks and outputs.

### 1.3 *Definition of basic concepts for the SDL process*

The SDL is based on the following definitions:

#### 1.3.1 *Signal*

- a) A *signal* is a flow of data conveying information to a process.
- b) A *signal* may be either in hardware or in software form.
- c) If the information flow is from a process in one functional block to a process in another functional block it is an external signal. If the flow is between processes within the same functional block it is an internal signal.

#### 1.3.2 *Input*

An *input* is an incoming *signal* which is recognized by a process. (It is not to be confused with input as applied to normal data processing.)

In accordance with the definition of *signals*, an input can be internal or external.

#### 1.3.3 *State*

A *state* is a condition in which the action of a process is suspended awaiting an *input*.

#### 1.3.4 *Save*

A *save* is the postponement of recognition of a signal when a process is in a state in which recognition of that signal does not occur.

#### 1.3.5 *Transition*

A *transition* is a sequence of actions which occurs when a process changes from one *state* to another in response to an *input*.

A process can be either in one of its states or in a transition at any one instant.

#### 1.3.6 *Output*

An *output* is an action within a *transition* which generates a *signal* which in turn acts as an *input* elsewhere. (It is not to be confused with output as applied to normal data processing.)

In accordance with the definition of signals an output can be either internal or external.

#### 1.3.7 *Decision*

A *decision* is an action within a *transition* which asks a question to which the answer can be obtained at that instant and chooses one of several paths to continue the *transition*.

#### 1.3.8 *Task*

A *task* is any action within a *transition* which is neither a *decision* nor an *output*.

#### 1.3.9 *Process*

In the context of SDL, a *process* is an object that either is in a *state* awaiting an *input*, or is in a *transition*.

2. SYMBOLS AND RULES

2.1 General

Each process represented consists of states and the transitions between them. An input causes the process to leave a state and travel along a transition, executing tasks, generating output signals and branching on decisions until another state is reached. The representations may be linear, with multiple appearances of a single state if convenient, or may be of mesh form or any combination of the two.

The concepts of state, input, task, output, decision and save are represented by their respective symbols. The appropriate interconnection of such symbols by flow lines represents the logical flow of a process.

2.2 Symbols

The recommended symbols appear in Figure 2/Z.102. Drawing conventions are explained in § 2.8.

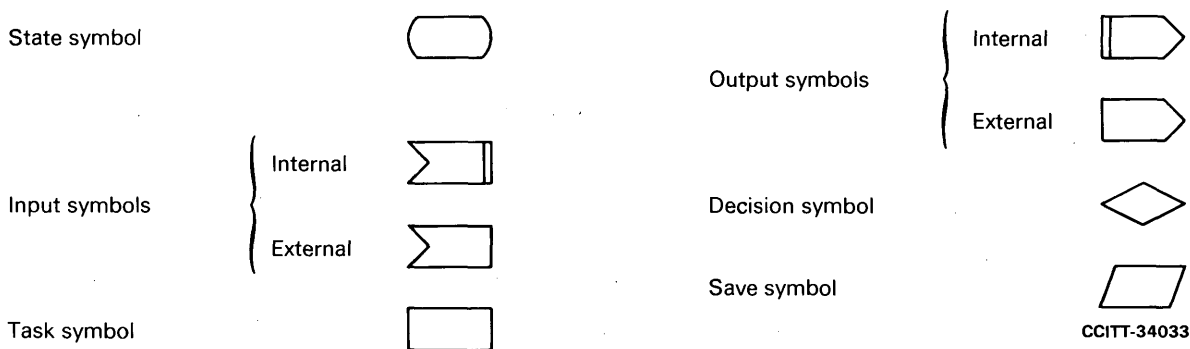


FIGURE 2/Z.102  
Recommended symbols

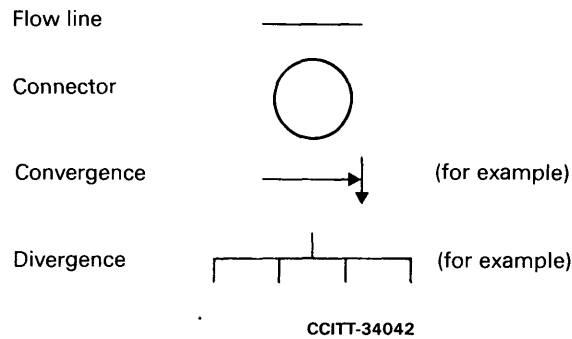
2.3 Sequence rules

Certain rules for the use of the symbols and their interconnections are required to ensure a valid representation of a process. For the purpose of these rules, *follow* means *follow immediately*.

- 2.3.1 A state symbol may only be followed either by input symbols or by input and save symbols.
- 2.3.2 Each input symbol and each save symbol follow one and only one symbol which must be a state symbol.
- 2.3.3 Each input symbol is followed by one and only one symbol, which may be any symbol except a save symbol or another input symbol.
- 2.3.4 Each task or output symbol is followed by one and only one symbol, which may be any symbol except an input or save symbol.
- 2.3.5 A decision symbol must be followed by two or more symbols, which may not be input or save symbols.

2.3.6 A save symbol must not be followed by any symbol.

## 2.4 *Flow lines and connectors*



## 2.5 *Flow line rules*

2.5.1 Every symbol is connected to the symbol it follows by a flow line.

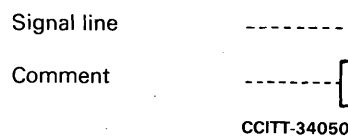
2.5.2 A solid flow line may be broken by a pair of associated connectors, with the flow assumed to be from the out-connector to its associated in-connector.

2.5.3 Where two or more symbols are followed by a single symbol the flow lines leading to that symbol converge. This convergence may appear as one flow line flowing into another or as more than one out-connector associated with a single in-connector, or as separate flow lines entering the same symbol.

2.5.4 Where a symbol is followed by two or more other symbols a flow line leading from that symbol may diverge into two or more flow lines.

2.5.5 Arrow heads are required whenever two flow lines converge and whenever a flow line enters an out-connector or a state symbol. Arrow heads are prohibited on flow lines entering input symbols.

## 2.6 *Annotations*



## 2.7 *Annotation rules*

2.7.1 Where an output symbol and an associated input symbol represent a signal from one process to another a dashed line from one symbol to the other may be included to indicate the association. Signal lines may diverge or converge.

2.7.2 Comments may be attached to a single square bracket connected by a dashed line to any symbol or flow line.



## 2.8 *Drawing conventions*

2.8.1 All symbols boxes of the same type shall preferably be of the same size within any one diagram.

2.8.2 The preferred orientation of symbols is horizontal as shown in Figure 2/Z.102, and the preferred aspect ratio of symbols is 2:1.

2.8.3 Mirror images of input and output symbols are allowed.

2.8.4 Flow lines are horizontal or vertical and have sharp corners.

2.8.5 Flow lines that cross have no logical relationship.

2.8.6 The text associated with a symbol should be placed within that symbol where practicable.

## 2.9 *SDL template*

A template suitable for hand drawing the basic set of SDL symbols is enclosed in the inside back cover of this fascicle.

## Recommendation Z.103

### 3. USE OF PICTORIAL ELEMENTS WITHIN STATE SYMBOLS

#### 3.1 *General*

3.1.1 The use of pictorial elements (PEs) within a state symbol forms an optional part of the SDL.

Such pictorial elements can provide advantages when applied to certain functional specifications and functional descriptions, resulting in a more compact and less verbal diagram.

3.1.2 With pictorial elements each state is represented by a state symbol containing a state picture and a state name (normally consisting of a state number and a state title) with the format shown in Figure 3/Z.103.

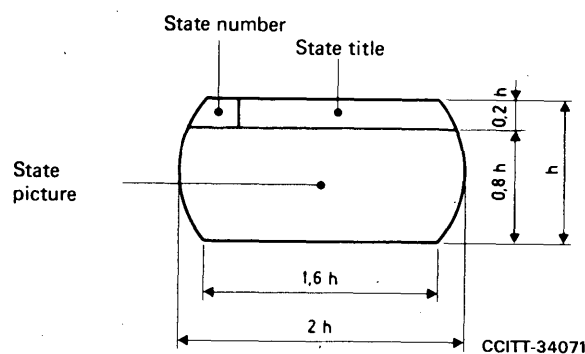







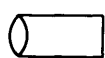











FIGURE 3/Z.103

Recommended format for a state symbol with pictorial elements

3.1.3 A basic set of PEs has been standardized for use in the SDL with application to the specification and description of telephone call-handling processes, including signal recognition and signalling interworking processes. Many of these PEs are capable of being applied in applications of the SDL beyond telephone call-handling processes, and their application to other processes in telecommunications systems, where appropriate, is encouraged.

### 3.2 Recommended symbols for pictorial elements

3.2.1 The recommended symbols for the basic set of PE concepts are as follows:

1) Functional block boundary				4) Signalling receiver	
2) Terminal equipment	(a) telephone on-hook			5) Signalling sender	
	telephone off-hook			6) Combined signalling sender and receiver	
	(b) trunk			7) Timer supervising of a process	
	(c) subscriber line			8) Charging in progress	
	(d) switchboard			9) Subscriber of terminal category	
	(e) other			10) Uncertainty symbol	*
3) Switching path	(a) connected			11) Switching module	
	(b) reserved			12) Control element	

CCITT-34100

Note – Examples of the use of these PEs can be found in Annex A.

3.2.2 The choice of symbols for PEs has been based upon the considerations and general selection criteria presented in Annex B, which should be consulted before developing additional PE symbols for wider applications of the SDL.

3.2.3 The recommended proportions of the PE symbols are shown in Annex C.

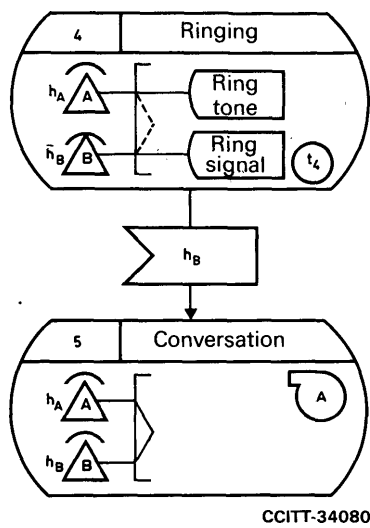
3.2.4 A template suitable for hand drawing the basic set of SDL symbols is enclosed in the inside back cover of this fascicle.

### 3.3 Range of applications

3.3.1 The total processing involved when going from one state to the following state is that required to effect the changes in the state pictures, together with the processing indicated in any decisions, outputs or tasks appearing in the transition between the states. For example, the total processing specified in the transition shown in Figure 4/Z.103, using state pictures, and implicit processing actions, is equivalent to the total processing specified in the transition shown in Figure 5/Z.103, where all processing actions are explicitly shown in the transition. A combination of state pictures and explicit transition elements is also valid.

3.3.2 When used in functional specification, the PEs appearing in state pictures as well as the text appearing in transition elements (tasks, decisions and outputs) should be chosen carefully so as to avoid prejudicing the system realization. In particular the PEs should represent functions rather than system-dependent components.

One of the general objectives of the SDL is that it should be open-ended, to be extended to cover new developments (Recommendation Z.101, § 1.1.2). The set of PEs that can be used effectively in the SDL is expected to grow, as the SDL is applied to a wider range of processes in telecommunications systems.



Note –  $h_B$  is equivalent to B off-hook in Figure 5/Z.103.

FIGURE 4/Z.103

Example of a transition between two states, where all processing actions are implied by the differences in the state pictures

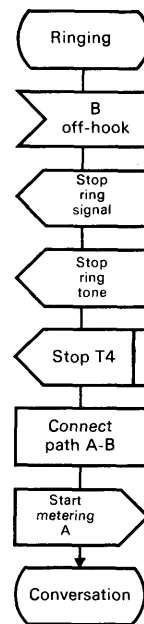
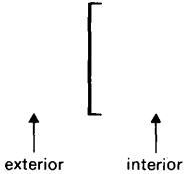
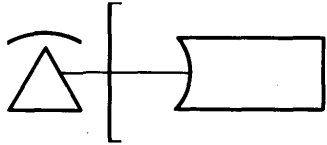
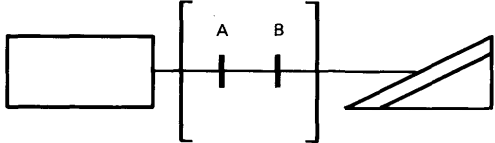









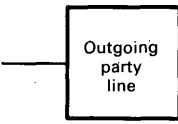

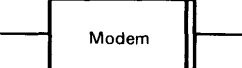




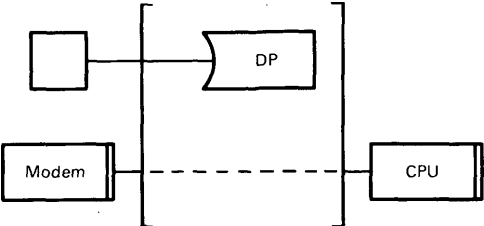
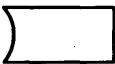



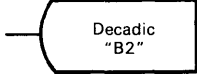





FIGURE 5/Z.103




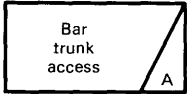
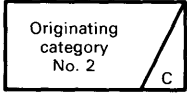



Example of a transition between two states, where all processing actions are explicitly shown in the transition

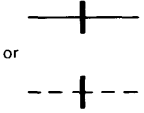
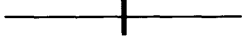
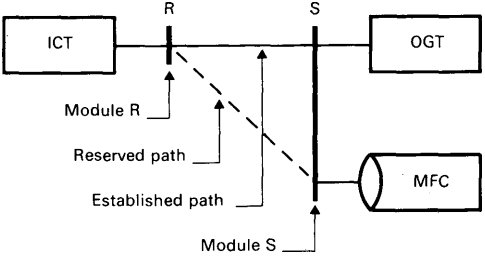

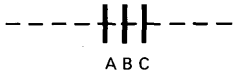
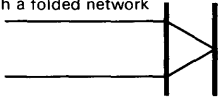

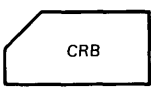
ANNEX A  
(to Recommendation Z.103)

**Examples of the use of the basic set of pictorial elements**

No.	Pictorial element	Comment	Examples
1.	<p><i>Functional block (FB) boundary</i></p> 	<p>To distinguish elements inside and outside the FB boundary. Only the states of elements within the boundary can be changed directly by this process.</p>	<p>1.1 A handset outside the FB boundary connected to a digit receiver inside the FB boundary</p>  <p>1.2 A trunk outside the FB boundary connected via a two-stage switching unit to a switchboard outside the FB boundary</p> 
2.	<p><i>Terminal equipment</i></p> <p>a) Telephone set</p> <p>on-hook </p> <p>off-hook </p> <p>b) Trunk </p> <p>c) Subscriber line [except a)] </p> <p>d) Switchboard </p> <p>e) Other </p>	<p>It can be useful to show terminal equipment (e.g. telephone set and switchboard equipment) outside the FB boundary, to improve understanding of the processing work.</p>	<p>2.1 A on-hook <math>\bar{h}</math> </p> <p>2.2 B off-hook <math>h_B</math> </p> <p>2.3 Incoming decadic trunk junctor (from a space division switching exchange) </p> <p>2.4 Outgoing subscriber line to a party line </p> <p>2.5 PBX switchboard </p> <p>2.6 Modem </p>

No.	Pictorial element	Comment	Examples
3.	<p><i>Switching path</i></p> <p>a) connected </p> <p>b) reserved </p>	<p>To show connectivity between terminal equipment and/or signalling devices involved in the process.</p>	<p>3.1 Subscriber line connected to a dial-pulse digit receiver and a modem with a reserved path to a central processing unit (CPU)</p> 
4.	<p><i>Signalling receiver</i></p> 	<p>To specify a signal reception process, and to indicate the nature of the signals received, especially those crossing the functional block boundary.</p>	<p>4.1 Multi-frequency code signalling receiver</p>  <p>4.2 MFC/decadic signalling receiver</p> 
5.	<p><i>Signalling sender</i></p> 	<p>To specify a signal sending process, and to indicate the nature of the signals sent, especially those required to cross the functional block boundary.</p>	<p>5.1 Decadic signalling sender with a backward signal "B2" being sent</p> 
6.	<p><i>Combined signalling sender and receiver</i></p> 	<p>This conveniently combines the functions of a signalling sender and signalling receiver.</p>	<p>6.1 MFC sender-receiver</p> 
7.	<p><i>Timer supervising a process</i></p> 	<p>Timers affect the subsequent behaviour of the process.</p> <p><i>Note</i> The related input symbol indicating timeout expiry, may be shown as <math>\bar{t}_i</math>.</p>	<p>7.1 Timer <math>t_3</math> is running</p>  <p>7.2 Generic timer <math>t_s</math> is running</p>  <p>where <math>s = 1, 2, \dots, n</math> define different service tones.</p>

No.	Pictorial element	Comment	Examples
8.	<p><i>Charging in progress (and which customer is being charged)</i></p> 	<p>The charging policy is significant to the Administration, the manufacturer and the customer.</p>	<p>8.1 Subscriber A is currently being charged</p> 
9.	<p><i>Subscriber or terminal category (and identity information)</i></p> 	<p>Changes in the subscriber or terminal category, for each party in a multi-party call, can affect the behaviour of the process.</p>	<p>9.1 The A party has trunk access barred</p>  <p>9.2 The C party has originating category No. 2</p> 
10.	<p><i>Uncertainty symbol</i></p> <p>*</p>	<p>This substitutes for deliberately undefined information that is shown unambiguously in other state pictures. In certain cases, two or more states may be safely merged into one, with a net gain in the intelligibility of the diagram, by using the uncertainty symbol.</p>	<p>10.1 Handset either on-hook or off-hook</p>  <p>10.2 Subscriber category either "bar trunk access" or not, in this state of the process</p>  <p>10.3 An undefined MFC signal is being sent in this state</p> 

No.	Pictorial element	Comment	Examples
11.	<p data-bbox="231 280 375 302"><i>Switching module</i></p> 	<p data-bbox="534 280 869 324">To show what switching modules are involved in the process.</p> <p data-bbox="534 336 869 481"><i>Note</i> – The horizontal line is the pictorial element for a switching path, which may be connected or reserved. The vertical line can be used to represent either a complete switching module (when the internal structure of the module is not required) or else one of the switching stages within a switching module.</p>	<p data-bbox="901 280 1340 302">11.1 A path connected through one switching module</p> <p data-bbox="949 313 1141 336">LLN = Line link network</p>  <p data-bbox="901 526 1412 571">11.2 Paths connected and reserved through two switching modules</p>  <p data-bbox="949 840 1181 907">ICT – Incoming trunk OGT – Outgoing trunk MFC – Multi-frequency code</p> <p data-bbox="901 918 1412 963"><i>Note</i> – In this example, ICT is connected to OGT, but ICT is not connected to the MFC sender/receiver.</p> <p data-bbox="901 1086 1412 1131">11.3 A path connected through a three-stage switching module RSN</p>  <p data-bbox="901 1310 1412 1355">11.4 A path reserved through a three-stage switching module ABC</p>  <p data-bbox="901 1512 1412 1545">11.5 A path connected through a folded network</p> 
12.	<p data-bbox="231 1780 406 1825"><i>Control element (assigned to a process)</i></p> 	<p data-bbox="534 1780 869 1892">To show what control equipment is involved in the process (especially modules that must be dimensioned). This symbol can be used to indicate that particular software elements have been assigned to the process.</p>	<p data-bbox="901 1780 1101 1803">12.1 Call register buffer</p> 

## ANNEX B

(to Recommendation Z.103)

### Selection criteria for pictorial elements

#### B.1 *General*

The choice of symbols for PEs has been based upon the following considerations and general selection criteria, which should be consulted before developing additional PE symbols for wider applications of the SDL.

#### B.2 *Typical readers*

It is expected that the SDL diagrams using PEs will be read by both technical and non-technical people in the following contexts: marketing new facilities; specifying new facilities; developing hardware and software from a specification; project management; operation and maintenance of an exchange; traffic engineering; education and training courses in telephony. It is expected that SDL diagrams will serve as common documentation:

- a) between Administrations and manufacturers,
- b) between different departments within these organizations,
- c) as telephone exchange documentation, and
- d) in training manuals and textbooks.

It is not expected that SDL diagrams (as diagrams) will be directly read by machines. Instead it is expected that the SDL/PR form of the SDL (including PE information) will be read by machines which will draw diagrams (see b) and c) of § B.3).

#### B.3 *Typical drawing methods*

It is expected that SDL diagrams using PEs will usually be drawn by technical people, including draftsmen, either:

- a) by hand, using a template as a drawing aid, and/or
- b) displaying the diagram electronically on a graphics visual display unit, and/or
- c) by using an electronically controlled plotter.

#### B.4 *Methods of reproduction*

The typical methods of reproduction are expected to be:

- a) the dye-line or blue-print methods, as in conventional drafting;
- b) photocopying by office machines, including photo-reduction;
- c) photo-printing in general.

#### B.5 *Ease of reproduction*

In order to permit convenient reproduction of SDL diagrams using the dye-line or blue-print methods of reproduction as well as photocopying and photo-printing, PE symbols should consist of clear lines without shading.

#### B.6 *Ease of drawing*

The following criteria reflect the assumption that the primary drawing technique will be to draw by hand using a template, and the secondary techniques will be displaying a diagram on an electronically controlled screen, and drawing the diagram with an electronically controlled pen:

- a) each PE symbol should be easy to draw with pen or pencil, either free-hand or using a stencil;
- b) all PE symbols should be drawn using the same thickness of lines;
- c) PE symbols should be created by synthesis of very simple geometric lines and curves in order to permit easy electronic generation of PE symbols.



B.7 Ease of comprehension

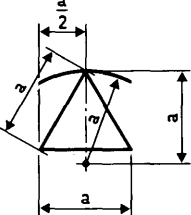
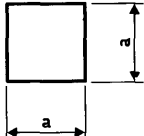
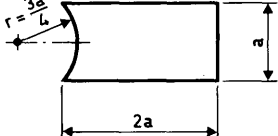
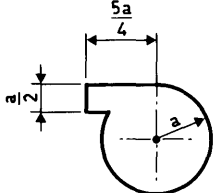
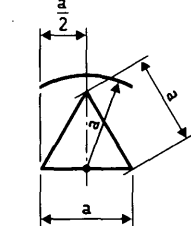
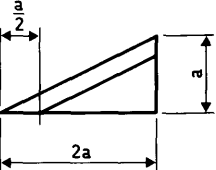
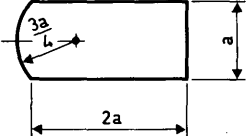
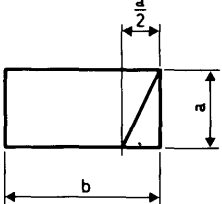
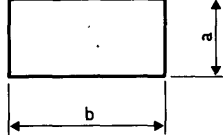
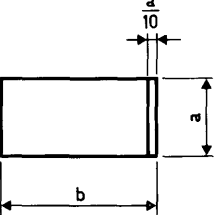
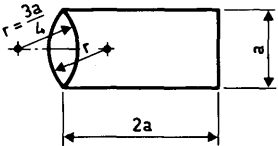
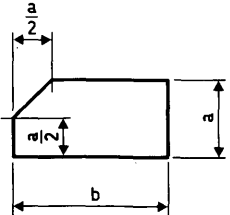
This is the most important consideration of all, since it is characteristic of SDL documentation that the readers are much greater in number than the drawers (or authors). This requirement is expressed by the following criteria concerning PE symbols:

- a) *Appropriateness* – The shape of each symbol should be appropriate to the concept that the symbol represents.
- b) *Distinctiveness* – When choosing a basic set of symbols, care should be taken to permit each symbol to be readily distinguishable from others in the set.
- c) *Affinity* – The shapes of PEs representing different but related functions, e.g. receivers and senders, should be related in some obvious way.
- d) *Association of abbreviated text with symbols* – In some cases it is expected that abbreviated text will be associated with a PE in order to indicate the class of PE; e.g. the letters MFC associated with a receiver symbol to indicate that multi-frequency coded signals are to be received. In these cases, the PEs should incorporate enclosed space to permit the use of a very small number of alphanumerical characters.
- e) *Limited set* – The total number of symbols should be kept to a minimum in order to permit easy learning of the pictorial method.

ANNEX C

(to Recommendation Z.103)

Recommended proportions for the basic sets of pictorial elements

<p>Terminal equipment</p> <p>a) Telephone set</p>  <p>– On-hook</p> <p>c) Subscriber line</p> 	<p>Signalling receiver</p> 	<p>Charging</p> 
<p>– Off-hook</p>  <p>d) Switchboard</p> 	<p>Signalling sender</p> 	<p>Subscriber or terminal category</p>  <p><math>b/a</math> any value <math>&gt; 1</math></p>
<p>b) Trunk</p>  <p><math>b/a</math> any value <math>&gt; 1</math></p> <p>e) Other</p>  <p><math>b/a</math> any value <math>&gt; 1</math></p>	<p>Combined signalling sender-receiver</p> 	<p>Control element</p>  <p><math>b/a</math> any value <math>&gt; 1</math></p>

CCITT-34110

## 4. SEMANTICS

### 4.1 *General*

This Recommendation describes a series of semantic rules to which an SDL representation (either an SDL diagram in SDL/GR or its program-like version) must conform, in order that the use of the SDL is uniform and clear.

### 4.2 *Basic structure of an SDL representation of a process*

4.2.1 Every SDL process can be represented by a directed closed graph, in the sense that every state can be reached from any other state by a suitable series of transitions. However, SDL diagrams may allow multiple appearance of the same state for ease of drawing or as an aid to better understanding. These diagrams are considered to be completely equivalent to the diagram resulting from merging all such identical states. States in SDL representations are considered to be unique.

4.2.2 State identification is determined by its name.

4.2.3 Some signals carry with them additional information. Signal names in input or save symbols attached to a given state must be unique. If some of the additional information is required to make the signal unique for recognition, that information must be made part of the name.

4.2.4 When a multiplicity of inputs leads to the same transition, one input symbol may be used containing all of these inputs, but from a semantic point of view this will be considered to be fully equivalent to a multiplicity of input symbols each containing only one signal name.

### 4.3 *Inter-process communication in SDL*

4.3.1 Communication between processes can only take place via signal sending and reception.

4.3.2 A signal is always sent to one particular process. The receiving process must be uniquely determinable from the SDL representation. This means that either the signal identification must occur only in the inputs of a unique process, or the output statements must be qualified by the intended process and/or functional block identifier (by means of signal lines or comments in SDL/GR, or the equivalent in SDL/PR). Process and/or functional block identifiers within a functional block must be unique.

### 4.4 *Semantics for signal reception*

The following describes the dynamic behaviour of signal reception.

4.4.1 When a signal arrives at a process, it is considered to be retained for that process (it is retained outside the process; hence it is not yet consumed by it).

4.4.2 When a state is entered, only *one* signal is selected out of the set of retained signals and is made available for consumption. A signal will only be made available if it is mentioned in an attached input symbol (possibly implied by Rule 4.4.5); and will not be made available if it appears in an attached "save symbol". If, at a given state, no retained signal can be made available, the process remains waiting at that state until a new signal arrives.

4.4.3 The selection method for becoming available is based upon the order of arrival (first in, first out).

4.4.4 A signal is said to be recognized as an input and consumed by the process if it leads to a transition.

4.4.5 At any given state, for all signal names not mentioned in explicit input or “save” symbols, there is an *implied* input symbol leading to an *implied* transition containing no actions and returning to the same state (thereby if the signal is made available it is consumed and the state is re-entered).

*Remark* – The semantics of the “save” symbol implies that, if a given signal arrives, the actual consumption is postponed until the process reaches a subsequent state (subsequent in time) where the signal is consumed.

## ANNEX A

(to Recommendations Z.101-Z.104)

### Examples of the use of SDL

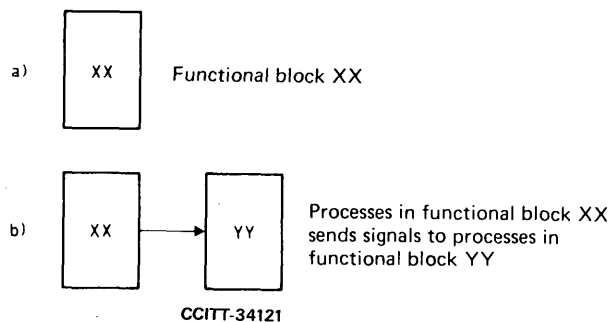
#### A General introduction

##### A.1 General

The examples in this Annex are intended to illustrate the versatility of the SDL in application to the specification (before design) and the description (after design) of several processes typical of SPC switching systems.

##### A.2 Functional block interactions

To provide a framework in which each example can be understood, the concepts of Recommendation Z.101 have been applied in the form of a simple diagram of the interactions between functional blocks for each example. The interpretation of the diagrams is as follows:

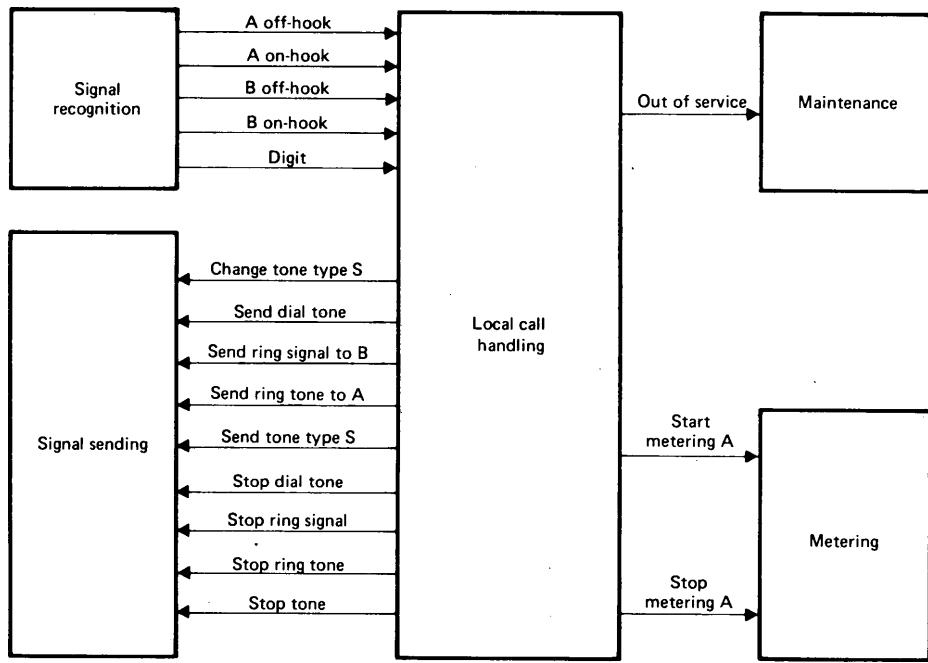


A.3 Examples

These examples have been designed to show the use of the SDL, and are not international specifications.

A.3.1 Local call handling process

A.3.1.1 Functional block interaction diagram



CCITT-34130

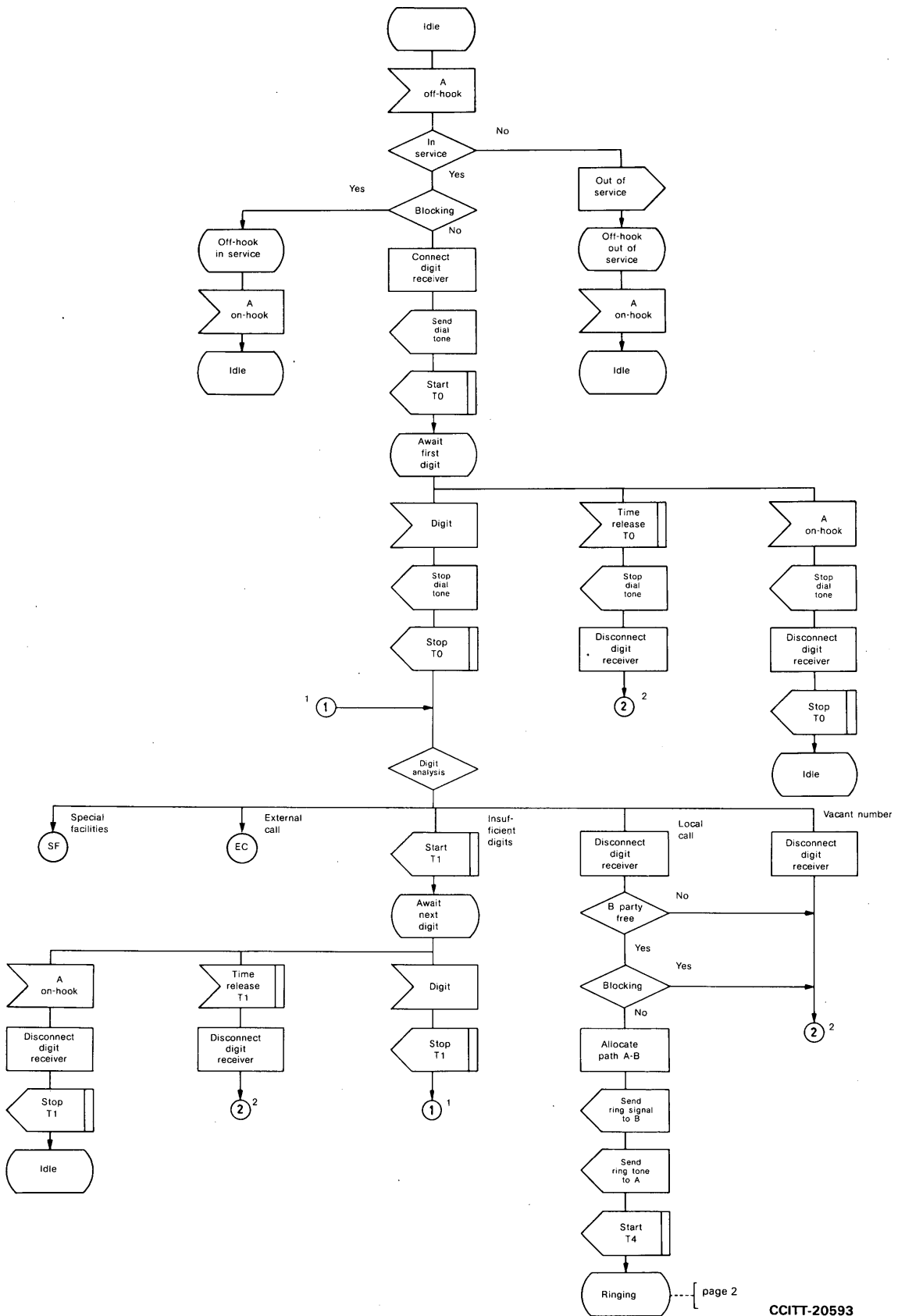
FIGURE A-1  
Functional block interaction diagram for local call handling

*Note* – In the subsequent SDL diagrams, the external input and output symbols have been drawn in such a way that their orientation indicates the direction of signals between the appropriate functional blocks.

A.3.1.2 Graphical representation without state pictures

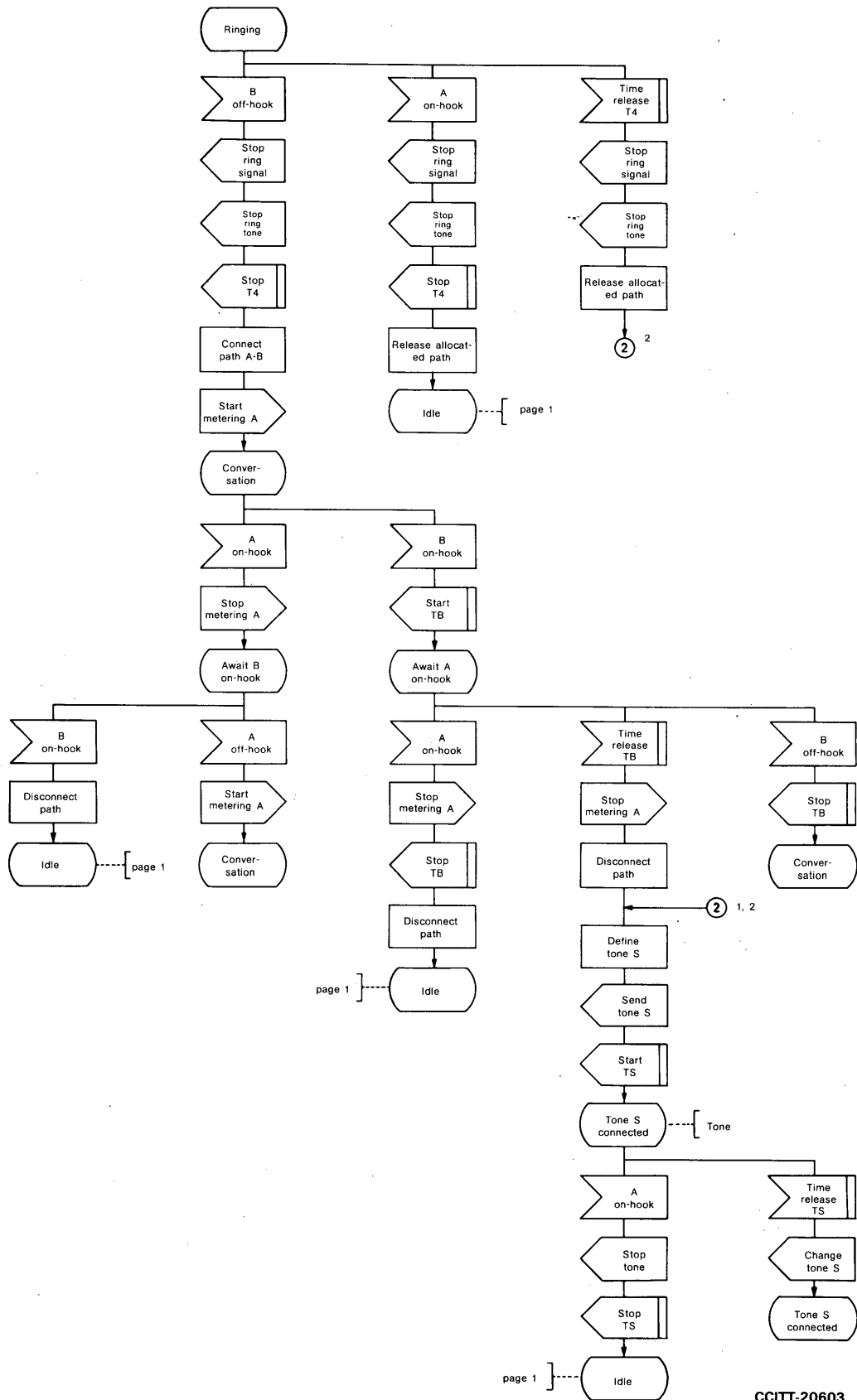
*Note 1* – The use of the blocking decisions in the SDL diagram accounts for the possibility of there being no free device or free path before device connection or path connection.

*Note 2* – The service tones are considered to be of various types  $S = 1, 2, 3, \dots$  that are not identified separately in Figure A-2.



Note – The numbers beside the connectors refer to the pages where the corresponding connectors are to be found.

FIGURE A-2 (Page 1 of 2)  
Local call handling process without state pictures



CCITT-20603

FIGURE A-2 (Page 2 of 2)  
Local handling process without state pictures

A.3.1.3 Graphical representation with state pictures

Note 1 – The use of the blocking decisions in the SDL diagram accounts for the possibility of there being no free device or free path before device connection or path connection.

Note 2 – The service tones are considered to be of various types  $S = 1, 2, 3, \dots$  that are not identified separately in Figure A-2.

Note 3 – Each state is shown only once in full, i.e. with its state number, state title and state picture. All other appearances of the state are shown by a diminished size state symbol containing the state number alone as sufficient identification.

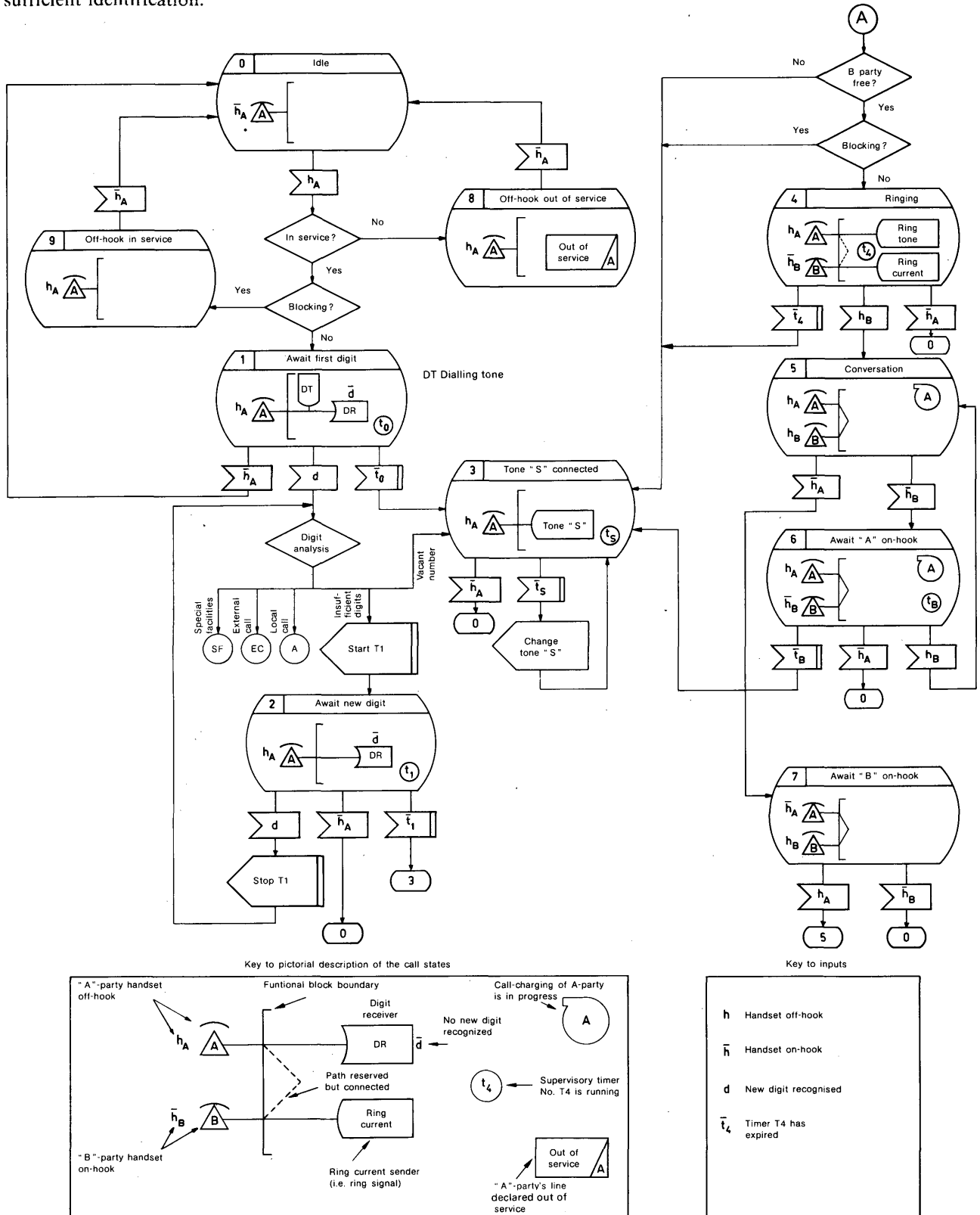


FIGURE A-3

Local call handling process, using state pictures

CCITT-20614

A.3.2 An R2 outgoing line signalling process

A.3.2.1 Functional block interaction diagram

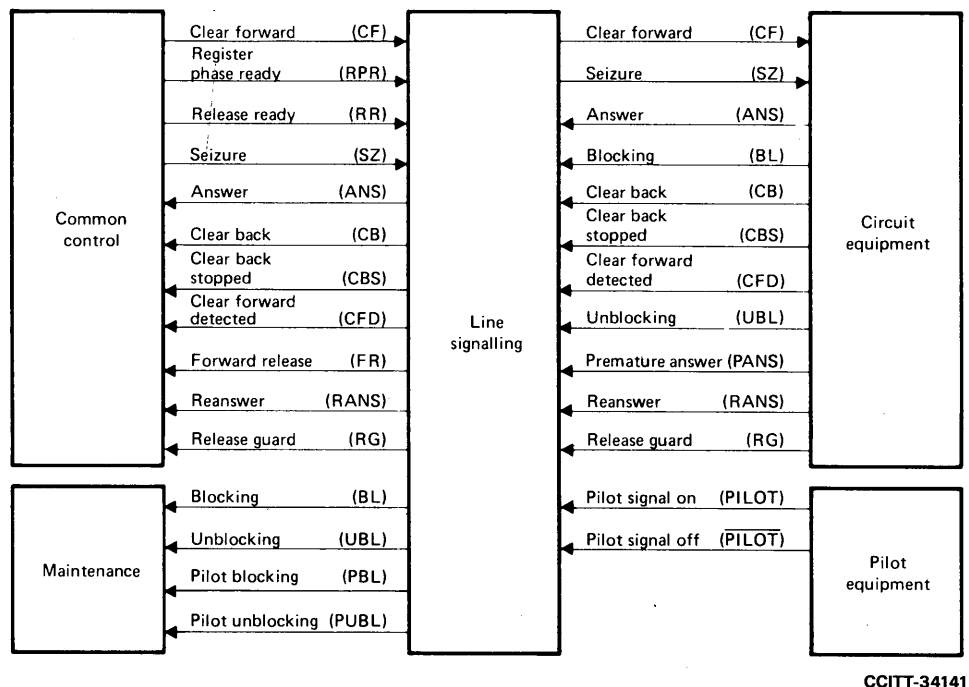


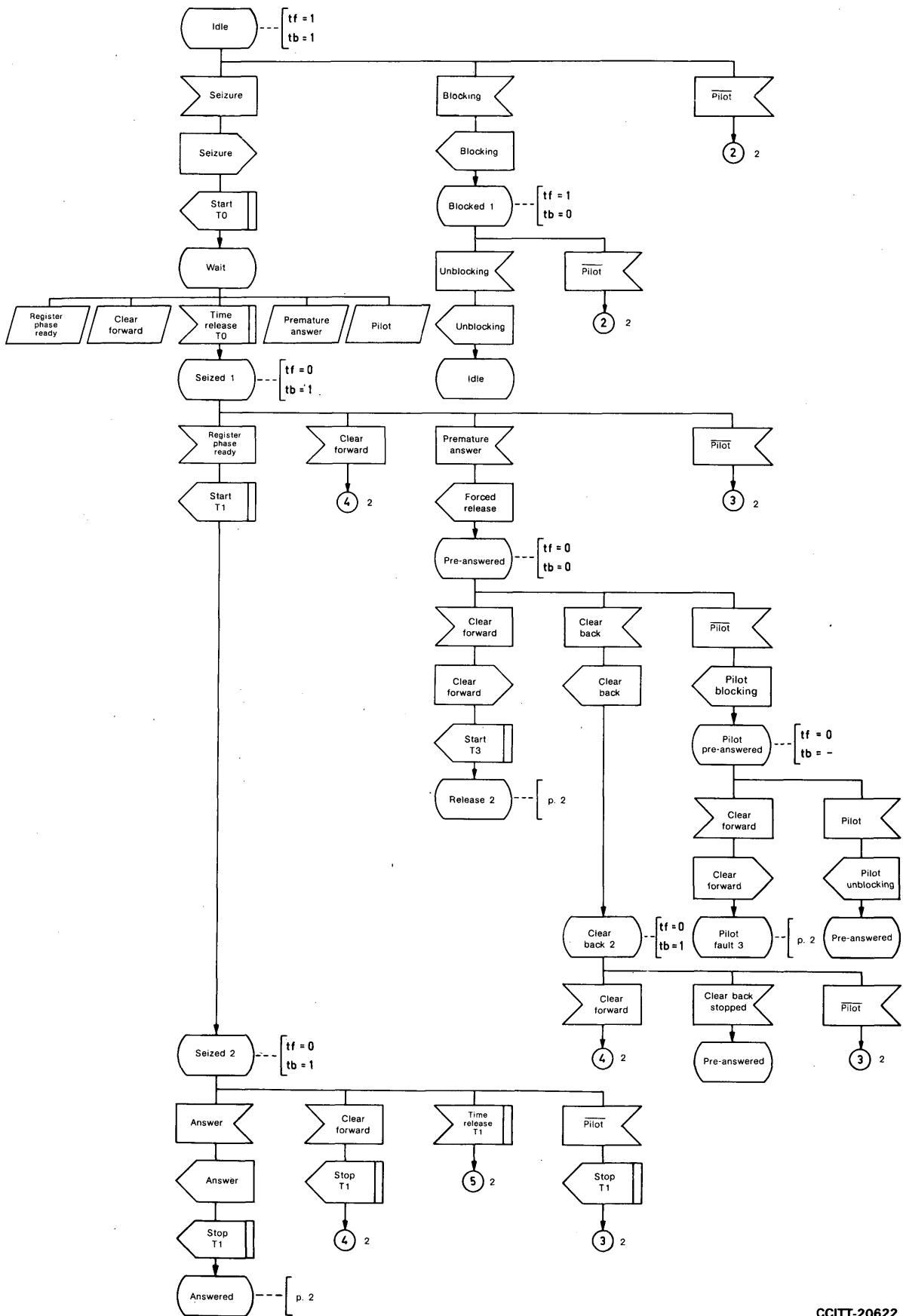
FIGURE A-4  
Functional block interaction diagram for R2 outgoing line signalling

Note – The external input and output symbols have been drawn in such a way that their orientation indicates the direction of the signals between the appropriate FBs.

A.3.2.2 Graphical representation without state pictures

Note – The abbreviation  $tf = 1$  ( $tf = 0$ ) indicates that the forward tone is on (off);  $tb$  refers to the backward tone.

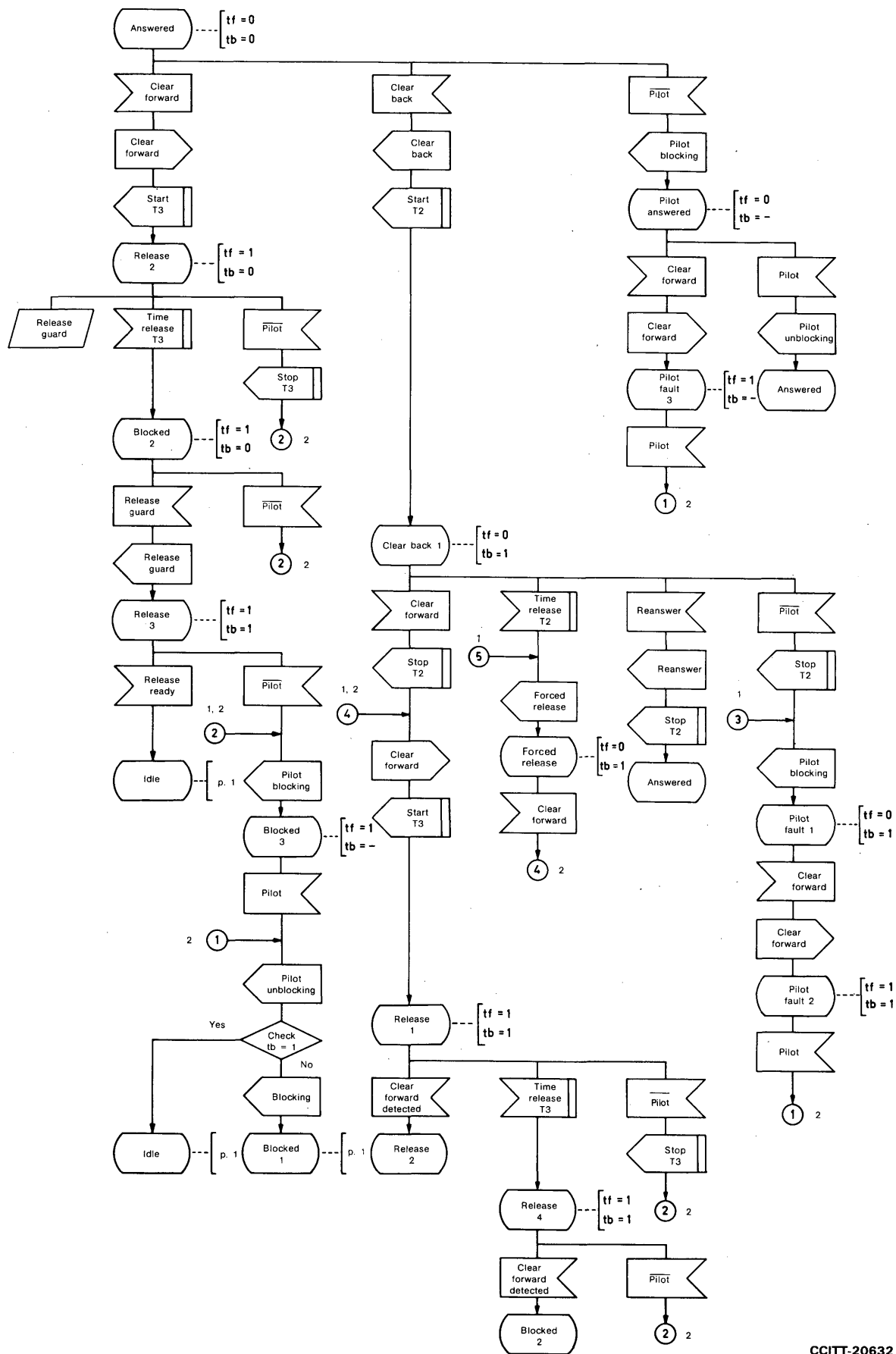




CCITT-20622

Note – The numbers beside the connectors refer to the pages where the corresponding connectors are to be found.

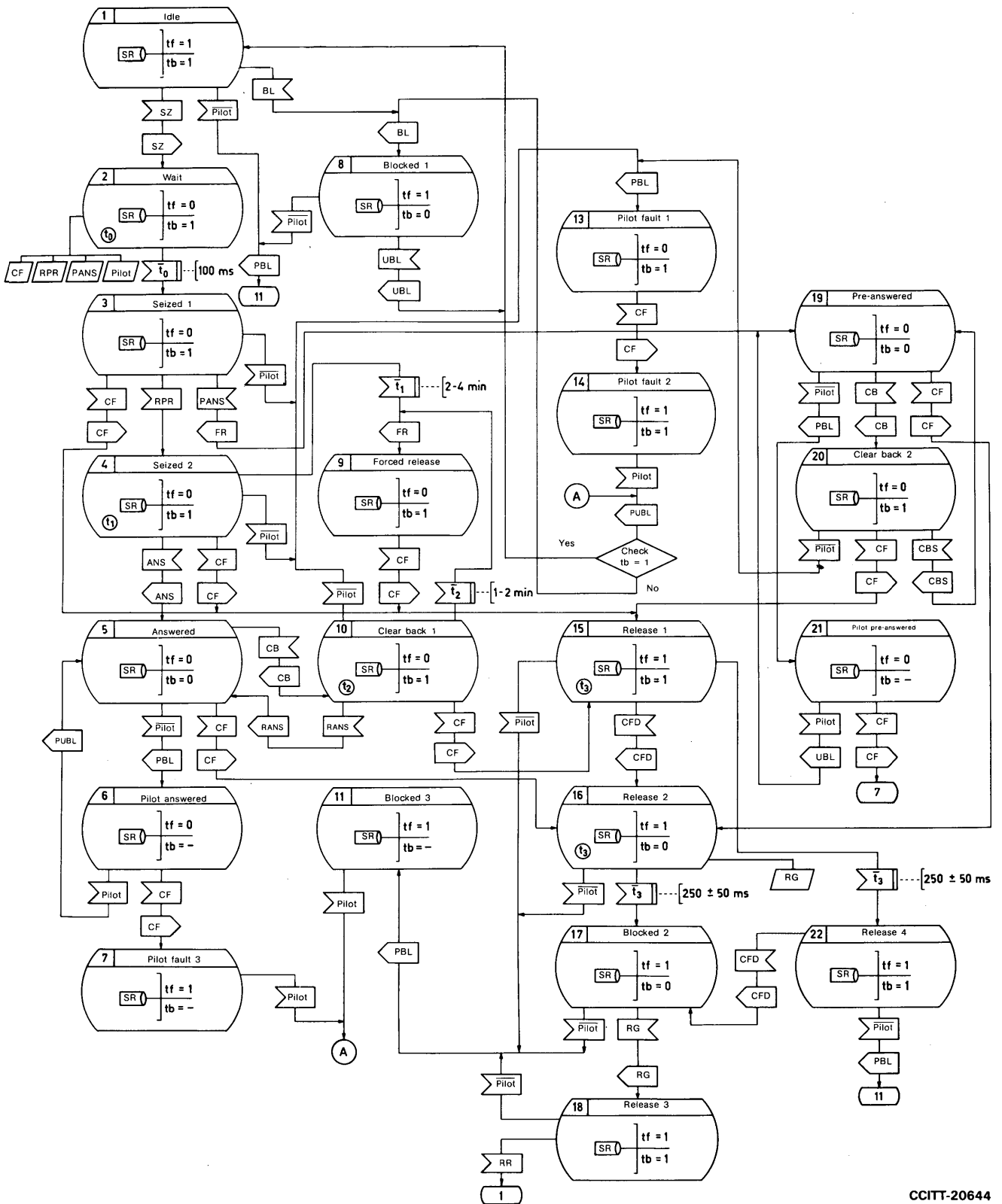
FIGURE A-5 (Page 1 of 2)  
An R2 outgoing line-signalling process without state pictures



CCITT-20632

FIGURE A-5 (Page 2 of 2)  
An R2 outgoing line-signalling process without state pictures

A.3.2.3 Graphical representation with state pictures



CCITT-20644

FIGURE A-6

An R2 outgoing line-signalling process using state pictures

A.3.3 System reconfiguration process

A.3.3.1 Functional block interaction diagram

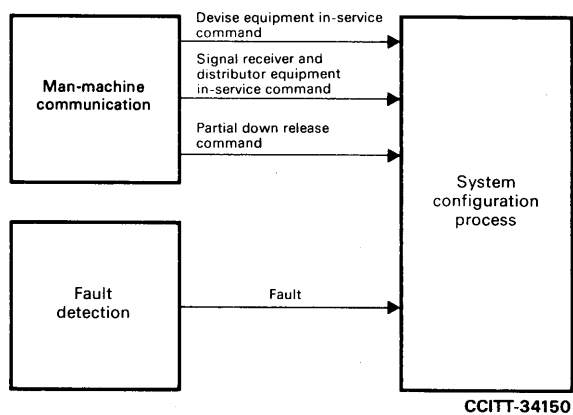
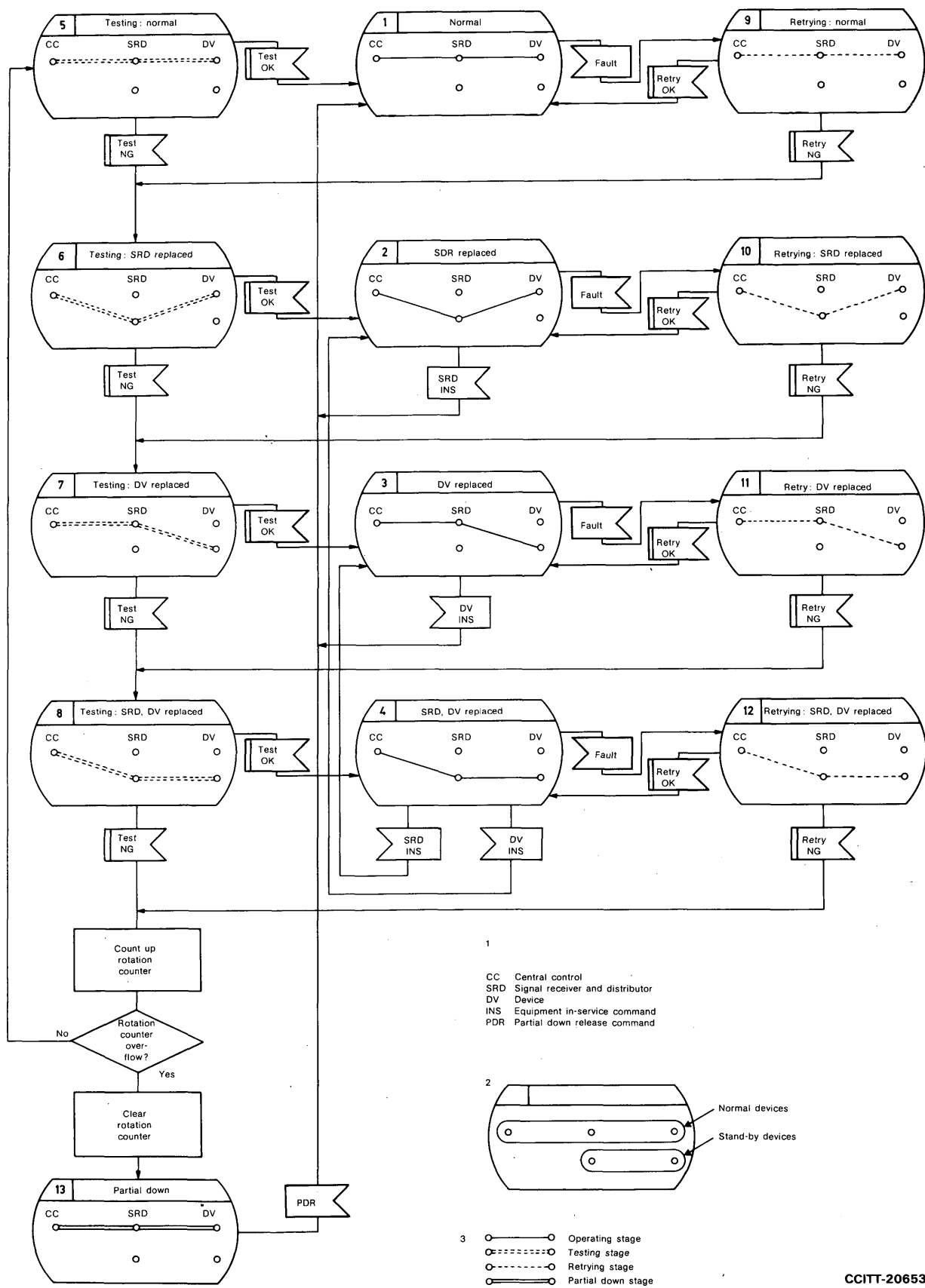


FIGURE A-7  
Functional block interaction diagram for  
system reconfiguration

A.3.3.2 Graphical representation



CCITT-20653

Note – The pictorial elements used in this example are beyond the basic set which have been standardized in Recommendation Z.103.

FIGURE A-8  
 A system reconfiguration process during fault conditions using state pictures

**Glossary of terms for the SDL**

The underlined terms are defined elsewhere in this glossary. References to the Z Recommendations and their Annexes are shown in brackets after each definition.

**B.1 action**

*F: action*

*S: acción*

An action is either a *decision*, a *task* or an *output*. A *transition* is composed of a sequence of actions. (Recommendation Z.101, § 1.3.5)

**B.2 annotation**

*F: annotation*

*S: anotación*

An annotation is either a *signal line* or a *comment*. (Recommendation Z.102, §§ 2.6, 2.7)

**B.3 associated connectors**

*F: connecteurs associés*


*S: conectores asociados*

A *flow line* may be broken by a pair of associated connectors, with the flow assumed to be from the *out-connector* to its associated *in-connector*. (Recommendation Z.102, §§ 2.4, 2.5.2)

**B.4 charging in progress PE**

*F: élément graphique de taxation en cours*


*S: elemento pictográfico de tasación en curso*

A *pictorial element* () indicating that charging is currently taking place. (Recommendation Z.103, § 3.2.1.8; Annex A to Recommendation Z.103, No. 8)

**B.5 combined signalling sender and receiver PE**

*F: élément graphique de combinaison d'émetteur-récepteur de signalisation*

*S: elemento pictográfico de emisor y receptor de señalización combinados*

A *pictorial element* () corresponding to a combined signalling sender and signalling receiver. [Recommendation Z.103 §§ 3.2.1, 6]; Annex A to Recommendation Z.103, No. 6]

**B.6 comment**

*F: commentaire*

*S: comentario*

Information which is in addition to or clarifies an SDL diagram. Comments may be attached by a single square bracket connected by a dashed line to a *symbol* or *flow line*. (Recommendation Z.102, §§ 2.6, 2.7.2)

**B.7 connected switching path PE**

*F: élément graphique de trajet de commutation établi*

*S: elemento pictográfico de trayecto de conmutación conectado*

A *pictorial element* (—) indicating connectivity between terminal equipment and/or signalling devices. [Recommendation Z.103, §§ 3.2.1, 3]; Annex A to Recommendation Z.103, No. 3]

**B.8 connector**

*F: connecteur*

*S: conector*

A connector (○) is either an *in-connector* or an *out-connector*. A *flow line* may be broken by a pair of *associated connectors*, with the flow assumed to be from the *out-connector* to its associated *in-connector*. (Recommendation Z.102, §§ 2.4, 2.5.2)

**B.9 consumption of a signal**

*F: absorption d'un signal*

*S: consumo de una señal*

A *signal* is said to be *recognized* or consumed if it leads to a *transition*. (Recommendation Z.104, § 4.4.2)

**B.10 control element PE**

*F: élément graphique d'élément de commande*

*S: elemento pictográfico de elemento de control*

A *pictorial element* (□) corresponding to some control equipment involved in a process. [Recommendation Z.103, §§ 3.2.1, 12]; Annex A to Recommendation Z.103, No. 12]

**B.11 convergence**

*F: convergence*

*S: convergencia*

Where two or more *symbols* are followed by a single *symbol* the *flow lines* leading to that *symbol* converge. This convergence may appear as one *flow line* flowing into another (→) or as more than one *out-connector* associated with a single *in-connector* or as separate *flow lines* entering the same *symbol*. (Recommendation Z.102, § 2.5.3)

**B.12 decision**

*F: décision*

*S: decisión*

A decision is an *action* within a *transition* which asks a question to which the answer can be obtained at that instant and chooses one of several paths to continue the *transition*. (Recommendation Z.101, § 1.3.7)

**B.13 decision symbol**

*F: symbole de décision*

*S: símbolo de decisión*

A *symbol* (◇) representing the SDL concept of a *decision*. (Recommendation Z.102, § 2.2)

**B.14 description**

*F: description*

*S: descripción*

The implementation of the requirements of a system is described in a description of the system. Descriptions consist of *general parameters* of the system as implemented and the *functional description (FD)* of its actual behaviour. [Recommendation Z.101, §§ 1.2.2 a), 1.2.2 b)]

**B.15 divergence**

*F: divergence*

*S: divergencia*

Where a *symbol* is followed by two or more other *symbols*, a *flow line* leading from that *symbol* may diverge into two or more *flow lines* (⊥). (Recommendation Z.102, § 2.5.4)

**B.16 external signal**

*F: signal externe*

*S: señal externa*

An external signal is a *signal* from a *process* in one *functional block* to a *process* in another *functional block*. [Recommendation Z.101, § 1.3.1 c)]

**B.17 flow line**

*F: ligne de liaison*

*S: línea de flujo*

A flow line (— or →) connects every *symbol* to the *symbol(s)* it follows. (Recommendation Z.102, § 2.5.1)

**B.18 functional block**

*F: bloc fonctionnel*

*S: bloque funcional*

A functional block is an object of manageable size and relevant internal relationship, containing one or more *processes*. (Recommendation Z.101, § 1.2.4)

**B.19 functional block boundary PE**

*F: élément graphique de limite de bloc fonctionnel*

*S: elemento pictográfico de frontera de bloque funcional*

A *pictorial element* (□) used to distinguish between elements inside and outside the *functional block*. [Recommendation Z.103, § 3.2.1, 1); Annex A to Recommendation Z.103, No. 1]

**B.20 functional block description**

*F: description de bloc fonctionnel*

*S: descripción de bloque funcional*

A functional block description describes the means by which the required behaviour of *processes* within a *functional block* is achieved. (Recommendation Z.101, § 1.2.4)

**B.21 functional block specification**

*F: spécification de bloc fonctionnel*

*S: especificación de bloque funcional*

A functional block specification specifies the required behaviour of the one or more *processes* within a *functional block*. (Recommendation Z.101, § 1.2.4)

**B.22 functional description (FD)**

*F: description fonctionnelle (DF)*

*F: descripción funcional (DF)*

The functional description (FD) of a system describes the actual behaviour of the implementation of the functional requirements of the system in terms of the internal structure and logic processes within the system. (Recommendation Z.101, § 1.2.3)

**B.23 functional specification (FS)**

*F: spécification fonctionnelle (SF)*

*F: especificación funcional (EF)*

The functional specification (FS) of a system is a specification of the total functional requirements of that system from all significant points of view. (Recommendation Z.101, § 1.2.3)



#### B.24 general parameters

*F: caractéristiques générales*

*F: parámetros generales*

The general parameters in both a *specification* and a *description* of a system relate to such matters as temperature limits, transmission limits, construction, exchange capacity, grade of service, etc. [Recommendation Z.101, § 1.2.2, c)]

#### B.25 in-connector

*F: connecteur d'entrée*

*S: conector de entrada*

A *flow line* may be broken by a pair of *associated connectors*, with the flow assumed to be from the *out-connector* to its associated in-connector. (Recommendation Z.102, § 2.5.2)

#### B.26 input

*F: entrée*



*S: entrada*

An input is an incoming *signal* which is *recognized* by a *process*. (Recommendation Z.101, § 1.3.2)

#### B.27 input symbol

*F: symbole d'entrée*

*S: símbolo de entrada*

Either of two *symbols* (  or  ) representing the SDL concept of an *input*. (Recommendation Z.102, § 2.2)

#### B.28 internal signal

*F: signal interne*

*S: señal interna*

An internal signal is a *signal* from a *process* in a *functional block* to a *process* in the same *functional block*. [Recommendation Z.101, § 1.3.1 c)]

#### B.29 out-connector

*F: connecteur de sortie*

*S: conector de salida*

A *flow line* may be broken by a pair of *associated connectors*, with the flow assumed to be from the *out-connector* to its associated *in-connector*. (Recommendation Z.102, § 2.5.2)

#### B.30 output

*F: sortie*



*S: salida*

An output is an *action* within a *transition* which generates a *signal* which in turn acts as an *input* elsewhere. (Recommendation Z.101, § 1.3.6)

#### B.31 output symbol

*F: symbole de sortie*

*S: símbolo de salida*

Either of two *symbols* (  or  ) representing the SDL concept of an *output*. (Recommendation Z.102, § 2.2)

**B.32 pictorial element (PE)**

*F: élément graphique (EG)*

*S: elemento pictográfico (EP)*

One of a number of standardized graphical entities used within *state pictures* to represent switching system concepts. (Recommendation Z.103, Annexes to Recommendation Z.103)

**B.33 process**

*F: processus*

*S: proceso*

A process performs a logic function that requires a series of information items to proceed, where these items become available at different points in time. In the context of SDL, a process is an object that either is in a *state* awaiting an *input* or in a *transition*. (Recommendation Z.101, §§ 1.2.5, 1.3.9)

**B.34 process description**

*F: description de processus*

*S: descripción de proceso*

The behaviour of a *process* is described in a process description, in terms of *inputs*, *saves*, *states*, *transitions*, *decisions*, *tasks* and *outputs*. (Recommendation Z.101, § 1.2.5)

**B.35 process specification**

*F: spécification de processus*

*S: especificación de proceso*

The behaviour of a *process* is specified in a process specification in terms of *inputs*, *saves*, *states*, *transitions*, *decisions*, *tasks* and *outputs*. (Recommendation Z.101, § 1.2.5)

**B.36 recognition of a signal**

*F: reconnaissance d'un signal*

*S: reconocimiento de una señal*

A *signal* is said to be recognized and *consumed* if it leads to a *transition* of the *process* from a *state*. (Recommendation Z.101, § 1.3.2; Recommendation Z.104, § 4.4.2)

**B.37 reserved switching path PE**

*F: élément graphique de trajet de commutation réservé*

*S: elemento pictográfico de trayecto de conmutación reservado*

A *pictorial element* (— — — —) representing a reserved connection between terminal equipment and/or signalling devices. [Recommendation Z.103, §§ 3.2.1, 3); Annex A to Recommendation Z.103, No. 3]

**B.38 retained signal**

*F: signal retenu*

*S: señal retenida*

When a *signal* arrives at a *process*, it is considered to be retained for that *process* (it is outside the *process*; hence it is not yet *consumed* by it). (Recommendation Z.104, § 4.4.1)

**B.39 save**

*F: mise en réserve*


*S: conservación*

A *save* is the postponement of *recognition of a signal* when a *process* is in a *state* in which *recognition of that signal* does not occur. (Recommendation Z.101, § 1.3.4)

**B.40 save symbol**

*F: symbole de mise en réserve*

*S: símbolo de conservación*

A symbol (  ) representing the SDL concept of a *save*. (Recommendation Z.102, § 2.2)

**B.41 SDL/GR**

*F: LDS/GR*

*S: LED/GR*

The graphical form of the SDL. (Recommendation Z.101, § 1.1.1)

**B.42 SDL/PR**

*F: LDS/PR*

*S: LED/PR*

The programme-like form of the SDL which is under study. (Recommendation Z.101, § 1.1.1)

**B.43 signal**

*F: signal*

*S: señal*

A signal is a flow of data conveying information to a *process*. (Recommendation Z.101, § 1.3.1)

**B.44 signal line**

*F: ligne de signal*


*S: línea de señal*

Where an *output symbol* and an associated *input symbol* represent a *signal* from one *process* to another, a dashed line from one *symbol* to the other may be included to indicate the association. (Recommendation Z.102, § 2.7.1)

**B.45 signalling receiver PE**

*F: élément graphique de récepteur de signalisation*


*S: elemento pictográfico de receptor de señalización*

A *pictorial element* (  ) representing a signalling receiver. [Recommendation Z.103, §§ 3.2.1, 4); Annex A to Recommendation Z.103, No. 4]

**B.46 signalling sender PE**

*F: élément graphique d'émetteur de signalisation*

*S: elemento pictográfico de emisor de señalización*

A *pictorial element* (  ) representing a signalling sender. [Recommendation Z.103, §§ 3.2.1, 5); Annex A to Z.103, No. 5]

**B.47 specification**

*F: spécification*

*S: especificación*

The requirements of a system are defined in a specification of that system. A specification consists of *general parameters* required of the system and the *functional specification (FS)* of its required behaviour. [Recommendation Z.101, §§ 1.2.2 a), 1.2.2 b)]

**B.48 specification and description language (SDL)**

*F: langage de spécification et de description (LDS)*

*S: lenguaje de especificación y descripción (LED)*

The CCITT language used in the presentation of the *functional specification* and *functional description* of the internal logic processes in stored programmed control (SPC) switching systems.

**B.49 state**

*F: état*

*S: estado*


A state is a condition in which the action of a *process* is *suspended* awaiting an *input*. (Recommendation Z.101, § 1.3.3)

State name	Each <i>state</i> is represented by a <i>state symbol</i> containing a state name (normally consisting of a state number and a state title) and, where <i>pictorial elements</i> are used, also containing a state picture. (Recommendation Z.103, § 3.1.2, Figure 4/Z.103)
State number	
State picture	
State title	

**B.50 state symbol**

*F: symbole d'état*


*S: simbolo de estado*

A *symbol* (  ) representing the SDL concept of a *state*. (Recommendation Z.102, § 2.2)

**B.51 subscriber line PE**

*F: élément graphique de ligne d'abonné*


*S: elemento pictográfico de línea de abonado*

A *pictorial element* (  ) representing a subscriber line. [Recommendation Z.103, §§ 3.2.1, 2); Annex A to Recommendation Z.103, No. 2)

**B.52 subscriber or terminal category PE**

*F: élément graphique de catégorie d'abonné ou d'équipement terminal*

*S: elemento pictográfico de categoría de abonado o terminal*

A *pictorial element* (  ) representing the category information associated with a specified subscriber or piece of terminal equipment. [Recommendation Z.103, §§ 3.2.1, 9); Annex A to Recommendation Z.103, No. 9]

**B.53 suspended process**

*F: processus suspendu*


*S: proceso en suspenso*

A *process* is suspended when it is in a *state*, awaiting an *input*. (Recommendation Z.101, §§ 1.3.3, 1.3.9)

**B.54 switchboard PE**

*F: élément graphique de tableau commutateur*


*S: elemento pictográfico de cuadro de conmutación*

A *pictorial element* (  ) representing a switchboard piece of terminal equipment. (Recommendation Z.103, §§ 3.2.1, 2); Annex A to Recommendation Z.103, No. 2)

**B.55 switching module PE**

*F: élément graphique de module de commutation*



*S: elemento pictográfico de módulo de conmutación*

A *pictorial element* (  ) representing a switching module associated with a connected or reserved switching path. [Recommendation Z.103, §§ 3.2.1, 11]; Annex A to Recommendation Z.103, No. 11]

**B.56 switching path PE**

*F: élément graphique de trajet de commutation*

*S: elemento pictográfico de trayecto de conmutación*

Either one of two *pictorial elements* (  or  ) representing a connected switching path or a reserved switching path. [Recommendation Z.103, §§ 3.2.1, 3]; Annex A to Recommendation Z.103, No. 3]

**B.57 symbol**

*F: symbole*

*S: simbolo*

In the context of SDL, a symbol is a representation of the concept of either a *state*, *input*, *task*, *output*, *decision* or *save*. (Recommendation Z.102, § 2.2)

**B.58 task**

*F: tâche*


*S: tarea*

A task is any *action* within a *transition* which is neither a *decision* nor an *output*. (Recommendation Z.101, § 1.3.8)

**B.59 task symbol**

*F: symbole de tâche*

*S: símbolo de tarea*

A *symbol* (  ) representing the SDL concept of a *task*. (Recommendation Z.102, § 2.2)

**B.60 terminal equipment PE**

*F: élément graphique d'équipement terminal*


*S: elemento pictográfico de equipo terminal*

One out of a possible six *pictorial elements* representing the following types of terminal equipment: telephone on-hook, telephone off-hook, trunk, subscriber line, switchboard or other. (Recommendation Z.103, § 3.2.1; Annex A to Recommendation Z.103, No. 2)

**B.61 time supervision of a process PE**

*F: élément graphique de temporisateur de contrôle d'un processus*

*S: elemento pictográfico de supervisión de un proceso por temporizador*

A *pictorial element* (  ) representing the running of supervisory timer,  $t_i$ . [Recommendation Z.103, §§ 3.2.1, 7]; Annex A to Recommendation Z.103, No. 7]

**B.62 transition**

*F: transition*


*S: transición*

A transition is a sequence of *actions* which occurs when a *process* changes from one *state* to another in response to an *input*. (Recommendation Z.101, § 1.3.5)

## B.63 trunk PE

*F: élément graphique de circuit*

*S: elemento pictográfico de enlace*

A pictorial element (  ) representing a trunk line interface. [Recommendation Z.103, §§ 3.2.1, 2); Annex A to Recommendation Z.103, No. 2]

## B.64 uncertainty symbol PE

*F: élément graphique de symbole d'incertitude*

*S: elemento pictográfico de símbolo de incertidumbre*

A pictorial element ( \* ) substituting for deliberately undefined information that is shown unambiguously in other state pictures. [Recommendation Z.103, §§ 3.2.1, 10); Annex A to Recommendation Z.103, No. 10]

## ANNEX C

(to Recommendations Z.101 to Z.104)

### SDL user guidelines

#### C.1 Preface

The CCITT Specification and Description Language, known as SDL, was first defined by Recommendations Z.101 to Z.103 in 1976 (the Orange Book, Volume VI.4), and later extended in Recommendations Z.101 to Z.104 in 1980 (the Yellow Book, this fascicle).

By 1977 it was already evident to Study Group XI that user guidelines were needed to facilitate the use of the SDL in application to a wide range of processes in telecommunications switching systems. It was difficult to produce a comprehensive set of user guidelines during the Study Period 1977-1980 because the SDL itself kept evolving. Consequently the user guidelines in this fascicle, having been prepared by a team of experts between May 1979 and January 1980, will themselves require improvement and extension during the following Study Period 1981-1984, and contributions to the user guidelines based upon practical experience with the SDL will be welcomed by the CCITT.

It is evident now in 1980 that the SDL is being used more widely by the CCITT and its member organizations, and that the range of applications of the SDL will continue to increase. These user guidelines are intended to assist people considering or starting to use the SDL, by supplementing the SDL Recommendations Z.101 to Z.104 with useful advice and helpful examples. The user guidelines are subordinate to the Recommendations, which should be consulted for the specification of the language in any case of doubt. It is realized that there will be some redundancy between the user guidelines and the Recommendations; this is believed to be desirable, to make the user guidelines self sufficient and readable.

#### C.2 Introduction

##### C.2.1 General

SDL can be used both in specification (to specify the required behaviour of a system) and in description (to describe the actual behaviour of a system). SDL has been designed to suit the specification and description of logic processes in telecommunications switching systems, but may be capable of wider application.

These guidelines have been prepared to help users understand better Recommendations Z.101-Z.104. It must be understood that these guidelines provide explanations and additional examples of the use of the Recommendations to help users in the application of SDL. The intent of further SDL study is to provide a methodology of documentation to completely represent the functional behaviour of stored programme control (SPC) switching systems.

SDL may be used to represent, at various levels of detail, the behaviour of a system, or a function or facility, in either specifications or descriptions.

Specifications may be very broad and general when an Administration wishes to explore the possibilities of a system with new features, new services, new technology, etc., while allowing the supplier to offer a wide range of design solutions. This type of specification would probably have only a few levels of detail. The other extreme is a specification in which an Administration is requesting a replacement or addition to an existing exchange. This specification would probably have a greater level of detail, because of the very detailed interfaces necessary.

In general, descriptions are written by suppliers in response to a specification (although they can be written to describe systems that the supplier wishes to sell). A description will usually have more levels of detail than the specification because of the need to describe the detailed behaviour of the system.

### C.2.2 *Forms of SDL*

SDL as defined by Recommendations Z.101-Z.104 is a language with a set of standardized graphical symbols representing most of the SDL concepts. However, a “programme-like form” of the SDL is currently under study, to permit the entry and modification of SDL diagrams in automated documentation systems. For this reason, a distinction is sometimes made between the graphical form (SDL/GR) and the programme-like form (SDL/PR) of the SDL. This edition of the user guidelines is aimed at explaining the use of SDL/GR.

### C.3 *SDL scenario*

Figure C-1 shows a range of possible uses of SDL, in the context of the purchase and supply of telecommunications switching systems.

In this figure, the rectangles illustrate typical functional groups, whose precise names may vary from organization to organization, but whose activities would be typical of many Administrations and manufacturers. Each of the directed lines (flow lines) represents a set of documents passing from one functional group to another; SDL can be used as part of each of these sets of documents. The figure is intended to be merely illustrative and is neither definitive nor exhaustive.

### C.4 *General explanation and basic concepts of SDL*

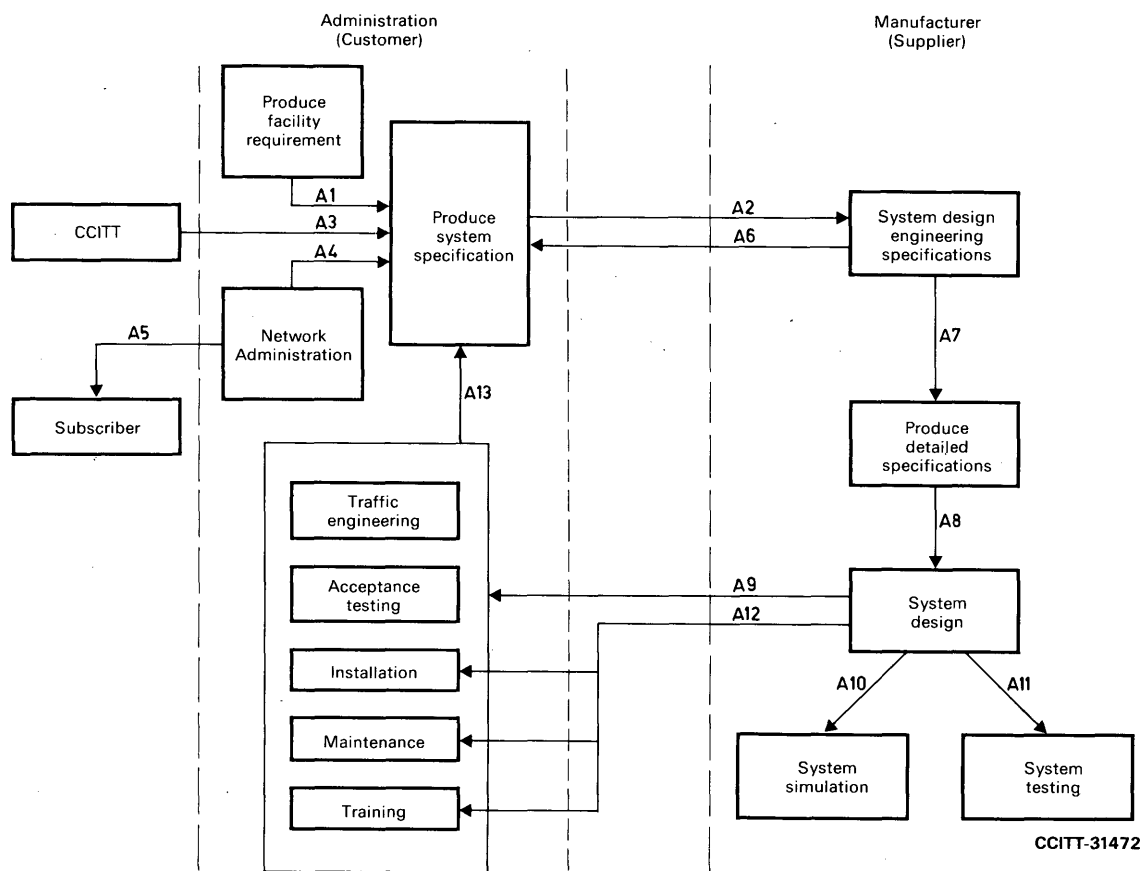
SDL is a means of representing the specification of the functional requirements and also the description of the logic processes necessary to implement the specification, in stored programme control (SPC) switching systems. The method of presentation is based on state transition diagrams, using the symbols and rules of SDL which are informally described below.

The main areas of application cover all types of SPC switching systems. Within these systems examples of processes which can be documented using SDL are: call processing (e.g. call handling, routing, signalling, metering, etc.), maintenance and fault treatment (e.g. alarms, automatic fault clearing, configuration control, routine tests, etc.) and system control (e.g. overload control).

The requirements of a system are defined in the specification of that system and the implementation of those requirements is defined in the description of that system.

A specification, or description, comprises two broad categories of information. The first category comprises the functional specification or functional description. These respectively detail the requirement, in terms of functions, and describe the actual behaviour of the system in terms of the logic processes. The second category comprises the more general parameters of the system and details such information as the environmental conditions – temperature limits, humidity, etc. – transmission limits, exchange capacity and grade of service, which are not represented in SDL. Functional specifications and functional descriptions are divided or partitioned into subsidiary functional specification blocks and functional description blocks respectively each containing one or more processes. Further details regarding partitioning are given in § C.5.

A *process* is described in terms of inputs, states, transitions, decisions, tasks, save and outputs. It is an object that either is suspended in a state awaiting an input or, on receipt of such an input, performs a sequence of actions made up of the execution of tasks, the making of decisions and the sending of outputs, moving along the transition leading from one state to the next. A process has a set of defined states connected by the defined transitions. Communication takes place between processes via signal sending and reception.



CCITT-31472

- A1 An implementation-independent and network-independent specification of a facility or feature
- A2 An implementation-independent and network-dependent system specification, including a description of the system environment
- A3 CCITT Recommendations and guidelines
- A4 Contributions to the system specification, showing the network administration and operational requirements
- A5 A description of facilities and features available to the subscriber
- A6 Description of an implementation proposal
- A7 A project specification
- A8 A detailed design specification
- A9 A complete system description
- A10 Appropriate system and environment description documentation for system simulation
- A11 Appropriate system and environment description documentation for system testing
- A12 Installation and operation manuals
- A13 Contributions to the system specification from specialized functional groups within the Administration

Note 1 – Iteration is possible at all levels.

Note 2 – In some circumstances, SDL documentation that is here shown as being internal to one organization, e.g. A1, A7, A8, could be supplied to another organization.

Note 3 – It is not expected that A12 would usually be implemented using SDL.

FIGURE C-1  
General scenario for the use of the SDL



A *signal* is a flow of data conveying information to a process. It may be provided by, and be in the form of, either hardware or software. It may be the presence, or absence, of a voltage or frequency. It may also be a digital message consisting of many bits of data. If it flows from a process belonging to a block to a process belonging to another block, it is an external signal. If flow is between processes described by the same block, it is an internal signal. A signal is always sent to one particular process. The receiving process must be uniquely determinable from the SDL representation. This means that either the signal identification must occur only in the inputs of a unique process, or the output statements must be qualified by the intended process and/or functional block identifier (by means of dashed lines or comments). Some examples of signals are: “off hook”, “timer T4 expires”, “answer”, “clear forward”, etc.

An *input* is an incoming signal which is recognized by a process. (It is not to be confused with input as applied to normal data processing.) In accordance with the definition of signals, an input may be internal or external.

A *state* is a condition in which the action of a process is suspended awaiting an input.

A *transition* is a sequence of actions which occurs when a process changes from one state to another in response to an input. A process can be either in one of its states or in a transition at any one instant.

An *output* is an action within a transition which generates a signal which in turn acts as an input elsewhere. (This is not to be confused with “output” as applied to normal data processing.) In accordance with the definition of signals, an output can be internal or external. Some examples of outputs are: “stop timer”, “send dial tone”, “stop ring tone”, etc.

A *decision* is an action within a transition which asks a question to which an answer can be obtained at that instant and according to that answer chooses one of several paths to continue the transition. Examples are: “in service?”, “blocking?”, “busy?”, “number of digit?”, “subscriber category?”, etc.

A *task* is any action within a transition which is neither a decision nor an output. Examples are: “connect digit receiver”, “connect path A-B”, “store digit”, etc.

The SDL signal reception procedure has a number of associated rules. Among these is the assumption that the simultaneous recognition of two or more signals is not possible. In any given system, it is, of course, possible for two or more signals for the same process to arrive in an interval of time such that their order of arrival is indeterminate. If this happens, the implied rule of SDL is that they are presented to the process in an arbitrary order.

An explicit rule of SDL associated with signal reception involves the save symbol. In some circumstances, a signal may arrive for a process and the process does not wish to consider it at its immediate state, but at a subsequent state. The inclusion of the save symbol at a state serves to indicate to the signal reception procedure that this situation exists and the save signal must be presented to the process at its next subsequent, in time, state.

## C.5 *SDL partitioning*

### C.5.1 *General*

SDL partitioning is the conceptual division of a system or subsystem into smaller entities, called functional blocks. It is a method of identifying particular features of the system to be specified or described. Each functional block contains a set of one or more processes which serve a common purpose. A block is separated from its environment by a boundary across which communication by means of signals may take place.

A block is a single object; however in general, a block may itself be partitioned into subsidiary blocks which are said to belong to a lower level. A block is then the equivalent of its subsidiary set of blocks. A functional specification, or a functional description, may itself be regarded as a block at the highest level of a hierarchy of system documentation.

### C.5.2 *Criteria for partitioning*

The particular criteria used for partitioning a system are at the user’s discretion. Any of the following criteria may be appropriate:

- a) to define blocks or processes of intellectually manageable size;
- b) to create a correspondence with actual software and/or hardware divisions;
- c) to follow natural functional subdivisions;
- d) to minimize interaction between blocks.

The actual criteria adopted may depend on whether a specification or a description is being considered and on the degree of detail required.

Each block can be further partitioned using the same or different criteria.

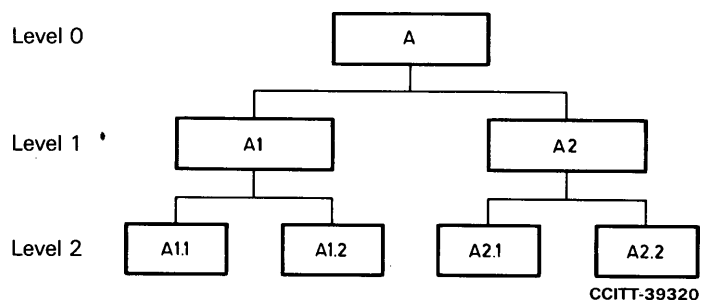
Since the relationship between levels will depend on the chosen partitioning criteria, it is important to state clearly which criteria have been chosen in order to allow easy comprehension of the documentation.

### C.5.3 SDL representation of one level

#### C.5.3.1 Partitioning diagram

The partitioning of a system or block can be represented by a partitioning diagram as shown in Figure C-2. This figure shows how Function A has been partitioned into blocks A1 and A2 which have each been further partitioned into two blocks.

Blocks, generated by the single act of partitioning a block at level  $i$ , are considered to be at level  $i + 1$ . In each block of the partitioning diagram, a unique name for that block should be written. The name should reflect the main function of the block.



*Note* – Each rectangle represents a functional block whose name is indicated inside the rectangle.

FIGURE C-2  
Example of a partitioning diagram

#### C.5.3.2 Functional block interaction diagrams

The behaviour of a functional block is represented by the means of one or more SDL diagrams each representing a single process.

Each of the SDL diagrams shows the handling of incoming signals and the transitions performed as a consequence.

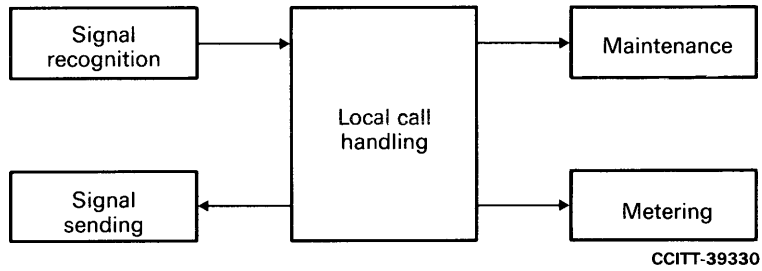
It is essential to indicate for all signals the sending functional block and the receiving functional block.

To achieve this a diagram of the type shown in Figure C-3 should be produced.

Figure C-3 provides an example of a functional block interaction diagram according to the example in Figure A-1 in Annex A to Recommendations Z.101 to Z.104.

Every functional block interaction diagram should be supplemented by a signal list. An example of a signal list structure is illustrated in the example shown in Table C-1.

*Note* – Where any signal carries associated data the nature of these associated data should be described in the signal list as shown in Table C-1.



*Note* – Each rectangle represents a functional block and each directed line represents the complete set of signals which pass between the functional blocks.

**FIGURE C-3**  
**Example of a functional block interaction diagram**

**TABLE C-1**  
**Example of a signal list extracted from Figure A-1 in Annex A**  
**to the Recommendations Z.101 to Z.104**

Signal		Functional block	
Name	Associated data (if any)	Sending	Receiving
A off-hook	Subscriber identity	Signal recognition	Local call handling
A on-hook	"	"	"
B off-hook	"	"	"
B on-hook	"	"	"
Digit	Value of digit	"	"
Change tone type S	"	Local call handling	Signal sending
Send dial tone	"	"	"
Send ring signal to B	"	"	"
Send ring tone to A	"	"	"
Send tone type S	"	"	"
Stop dial tone	"	"	"
Stop ring signal	"	"	"
Stop ring tone	"	"	"
Stop tone	"	"	"
Out of service	"	"	Maintenance
Start metering A	Subscriber identity	"	Metering
Stop metering A	"	"	Metering

#### C.5.4 *Relationship between functional blocks and processes*

A functional block can contain more than one process. Internal input and internal output symbols are used to represent communication between processes belonging to the same functional block, whereas external input and external output symbols are used to represent communication between processes belonging to different functional blocks. For this reason, the choice of partitioning may affect the symbols used for inputs and outputs in SDL diagrams.

The more general question of the relationship between functional blocks and processes is still under study.

#### C.6 *Guidelines for representing processes*

##### C.6.1 *General*

##### C.6.1.1 *Graphical and programme-like forms of SDL*

Depending upon which form of SDL is used, processes may be represented in two ways, one graphical (SDL/GR) and one programme-like (SDL/PR).

The SDL/PR will be developed in order to facilitate the automatic generation, modification and analysis of SDL diagrams. Guidelines on SDL/PR will be produced after SDL/PR has been developed.

§ C.6 provides general guidelines on process representation and special guidelines for the graphical form of SDL.

A template for the SDL graphical symbols has been provided in the inside back cover of this fascicle. A schematic design for this template is given in Figure C-4.

##### C.6.1.2 *Version of SDL/GR*

If in an SDL/GR diagram, transitions are described exclusively by explicit action symbols, this is called the transition-oriented approach.

If on the other hand the states are described using state pictures, and the difference in state pictures is used to imply transition actions, this is called the state-oriented approach.

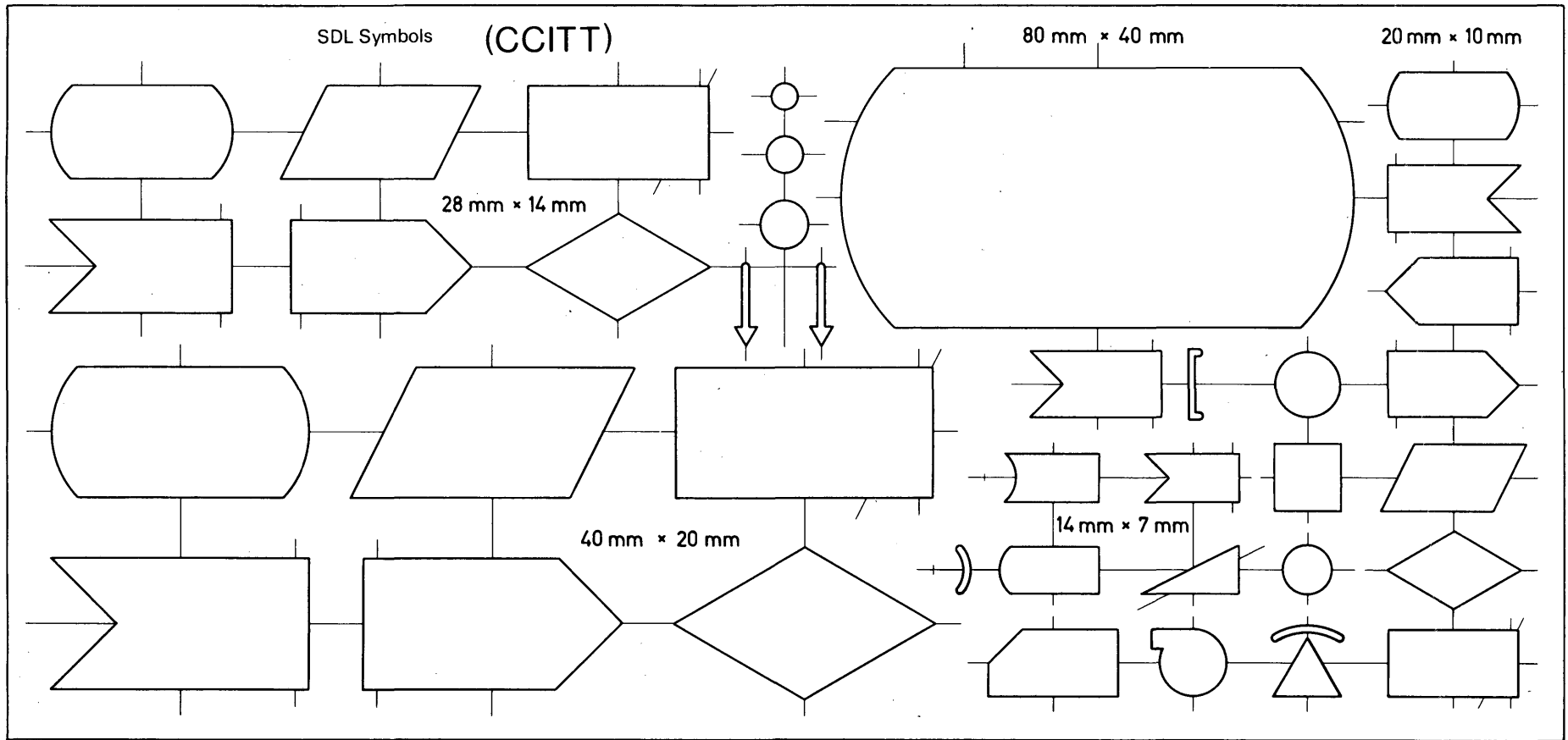
A combination of both approaches is called the combined approach.

Examples of these three approaches are given in Figures C-5 to C-7.

The transition-oriented approach is suitable when the sequence of actions is important and detailed state descriptions are less important.

The state-oriented approach is suitable when the sequence of actions within each transition is not important, when pictorial explanation is desirable, and when compact representation is desirable.

The combined approach is suitable when the redundancy implied is helpful.



CCITT-39341

Note –  $40 \text{ mm} / \sqrt{2} = 28 \text{ mm}$ ;  $28 \text{ mm} / \sqrt{2} = 20 \text{ mm}$ ; etc. This permits photoreduction from A3 to A4 paper with compatible symbols.

FIGURE C-4  
Schematic design of the SDL template

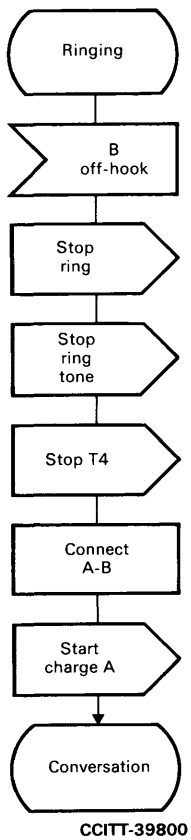


FIGURE C-5  
Transition-oriented form

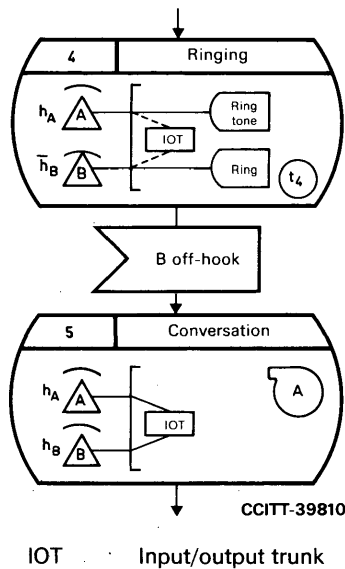


FIGURE C-6  
State-oriented form

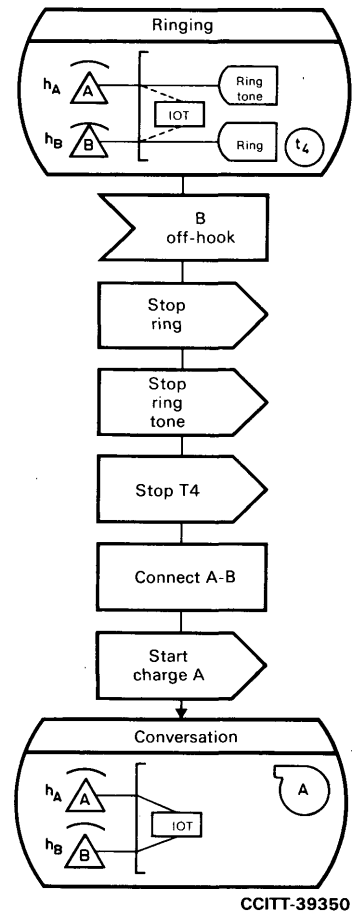


FIGURE C-7  
Combined form

## C.6.2 States

### C.6.2.1 General

In SDL, a process is represented by a network of states and transitions. A state is a point in the process where no actions are being executed by that process but monitoring for incoming signals is being carried out on behalf of the process. According to the signal names given in the input and save symbols associated with a particular state, the arrival of any one of a defined set of signals will trigger the process to leave the state and begin the execution of a transition. A signal which has arrived and has triggered the execution of a transition has been “consumed” and, by definition, ceases to exist. However any data associated with the signal is available during the transition.

A state is represented by a state symbol and has associated input and save symbols. Two examples of states are shown in Figures C-8 and C-9.

If two states, in the same SDL diagram, appear to be the same but differ from each other in any of the following ways:

- the number or names of inputs and saves,
- the sequence of actions to be executed following an input,
- the “next state” to be entered following the execution of a transition,

then the two states are not the same, and therefore they should be given different names.

Usually the author of an SDL diagram has some flexibility available in his approach to defining the states of a process. Good judgement (obtained through practice) is required in order to produce SDL diagrams which are neither unnecessarily complicated through the identification of too many distinct states or fail to exploit the inherent advantages of SDL through having an artificially reduced number of states.

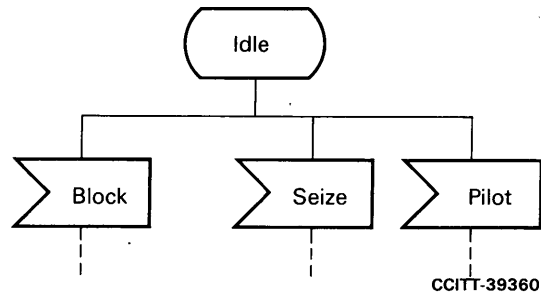


FIGURE C-8  
Example of a state

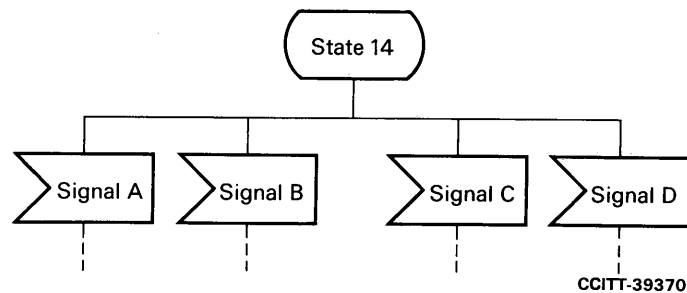
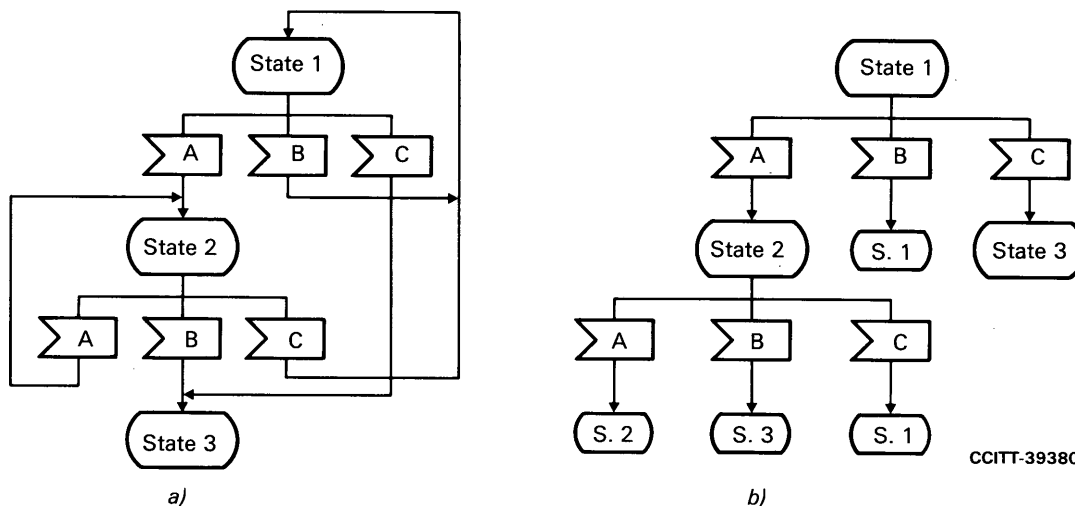


FIGURE C-9  
Example of another state

### C.6.2.2 Multiple appearances

For convenience, to simplify drawing or as an aid to better understanding, the same state may appear a number of times in an SDL diagram. Where this is done, the diagram is considered to be completely equivalent to the diagram which would result from merging all multiple appearances of the same state. Concerning multiple state appearances, two following approaches can be used.

C.6.2.2.1 The state has one main appearance where all of its related input and save symbols are shown. Where this state appears at the other points in the diagram it is shown (as the terminating point of a transition) without any input or save symbols and, in effect is a simple way of indicating the next state without the need for a connecting flowline or a pair of connectors. To avoid confusion it is useful to use a state symbol of reduced dimensions for these secondary appearances (see Figure C-10).



a) and b) are equivalent.  
 In b) any state symbol with reduced dimensions has been used as a connector to the state having the same name.

FIGURE C-10  
 Example of multiple appearances of a state

C.6.2.2.2 At each multiple appearance of the state, the state is shown together with a subset of the input or save symbols. This approach has been successfully used where states have a relatively large number of input or saves but does have the danger that readers may misinterpret the diagram because they are unaware that there are multiple appearances. Appropriate use of the comment concept can avoid this misunderstanding.

#### C.6.2.3 Elapsed time

On entering a state, a process may have to wait for some time for one of the defined set of inputs which can trigger the execution of a transition from that state. In certain circumstances (discussed in §§ C.6.3 and C.6.4) the execution of a transition can be triggered as a result of a signal which arrived before the entry into the state and in this case an immediate exit from the state could occur.

In passing from one state to another, time may or may not elapse, but SDL syntax/semantics do not directly cover this aspect. Readers of SDL diagrams (particularly those relating to high level specification) may assume that negligible time elapses during transitions – a concept which is common in finite state automata theory. Writers of SDL diagrams covering processes where transition times are significant should bear this in mind; it may be appropriate to include additional comments which indicate transition times.

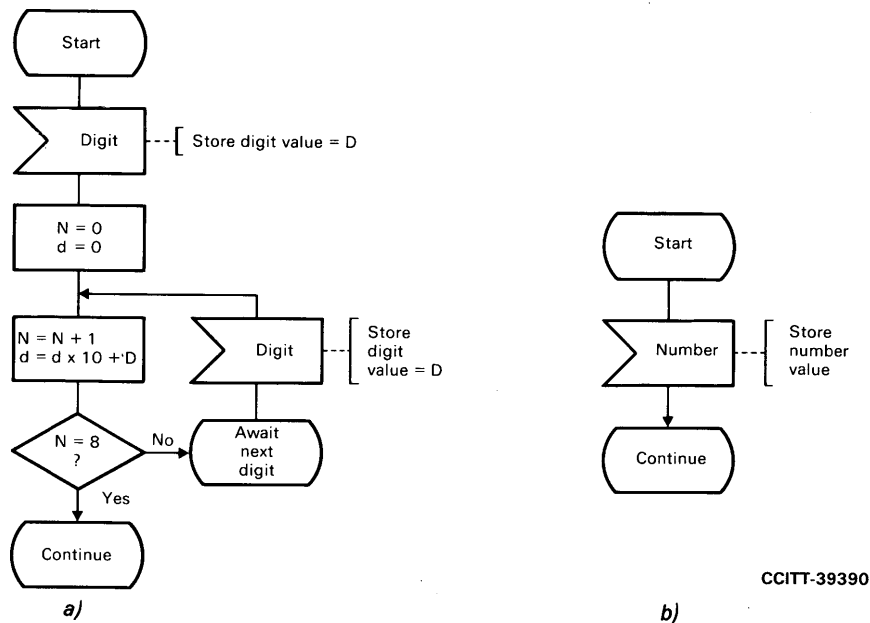
#### C.6.2.4 Determination of the required states

The author of an SDL diagram requires a strategy which enables him to identify the states of the process and this strategy can be informal or formal. Before starting to draw the diagram, necessary preliminaries (discussed in C.5) must have been completed, for example:

- the partitioning of the system into functional blocks
- the representation of a functional block by means of one or more SDL processes
- the choice of input and output signals.

All the above factors have a crucial effect in determining the states of each process. The effect of the “choice” of input signals upon the numbers of states in an SDL diagram is shown by the two examples in Figure C-11.





Examples a) and b) represent the same function with different level of details. As a consequence the number of states varies.

FIGURE C-11  
Reception of an 8-digit telephone number

#### C.6.2.5 Reduction in the number of states

Having used a strategy for identifying the states of a process, the author of an SDL diagram may feel that “too many states” have been used. The number of states is important because the size and complexity of an SDL diagram is often closely related to the number of states. There may be some means available for reducing the number of states, but the fact that an SDL diagram is complex is not, by itself, a reason for changing it as the complexity of the diagram may simply be a reflection of the inherent complexity of the process which it defines.

The number of states can be reduced by separation of common functions (see § C.6.2.6) or by merging states (see § C.6.2.7).

#### C.6.2.6 Separation of common functions

It may become clear when planning an SDL diagram, that to define a particular and repetitive aspect of a process would require the representation of repetitive states. In Figure C-12 part of a line-signalling process SDL diagram is shown which illustrates the requirement that a line signalling tone must be present for a particular length of time before the line signal is considered to have been detected. To specify this an intermediate state is required between the *No line signal detected* and *Conversation* states. Let us assume that in a complete diagram, such a common function would have to be repeated at every point where the signal is detected. An alternative way is to define a separate process which is responsible for monitoring the line signalling tone and detecting signals on the basis of the specified recognition time. The existence of this new process would enable the diagram shown in Figure C-12 to be shown in Figure C-13. (In a given context, the figures can be made equivalent, at the expense of introducing a new signal, *Valid line signal*.)

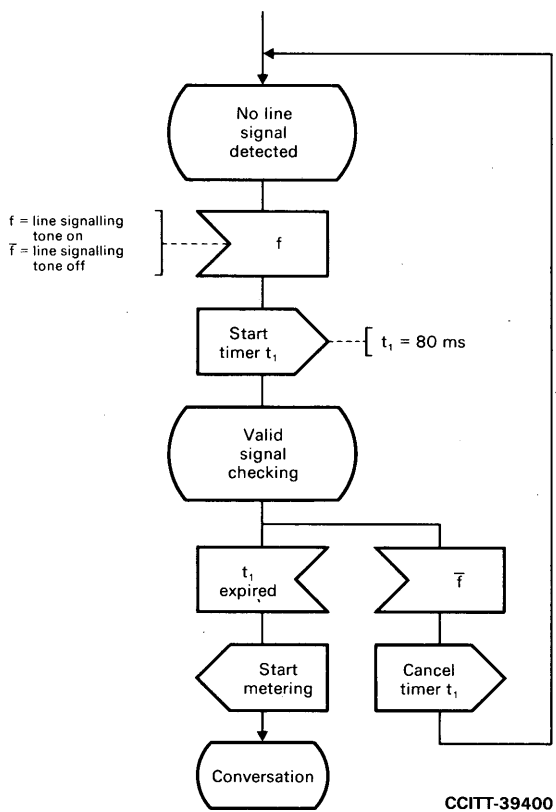


FIGURE C-12

Example of an SDL diagram for a composite call-handling and line signal detection

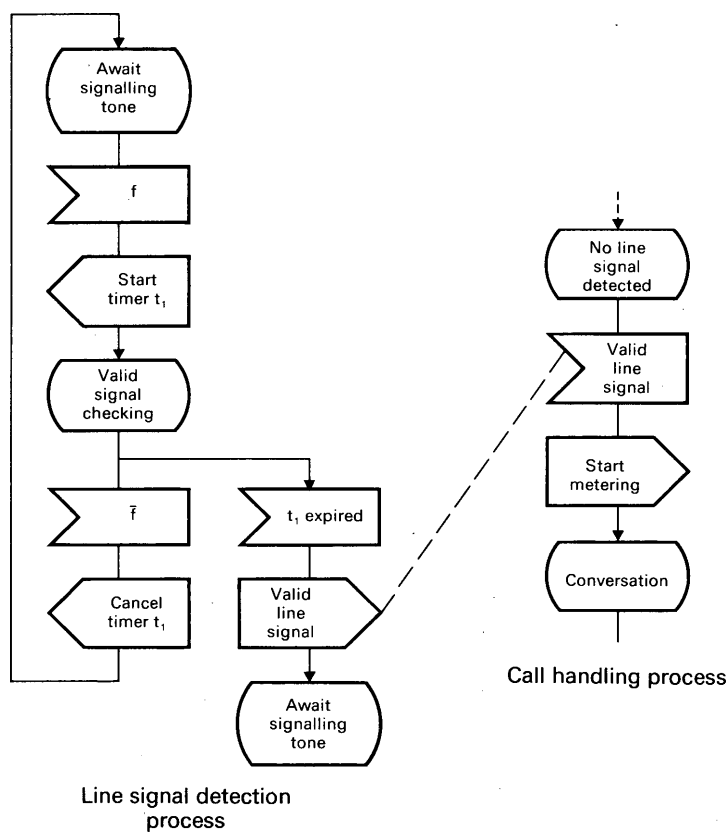


FIGURE C-13

Example of the separation of a common function (line signal detection) to avoid repetitive states such as valid signal checking (compare with Figure C-12)

### C.6.2.7 Merging of states

If in an SDL diagram, the future of two states are the same, then, irrespective of their history they can be merged into one without affecting the logic of the diagram.

Figure C-14 shows part of an SDL diagram with two states which differ in their history but their futures are "identical". In Figure C-15, the two states have been combined to form one state. This is a fairly trivial example where the reduction in complexity is small but the technique can be used to obtain significant simplification. The SDL semantics do not allow the possibility of a decision following a state to determine the history of the process before the state (i.e. for this example, was A6 or B4 sent?) unless this information had been explicitly stored prior to entry into the state.

A state should represent a logical situation of the process, therefore it is not advisable to merge different logical situations in one state.

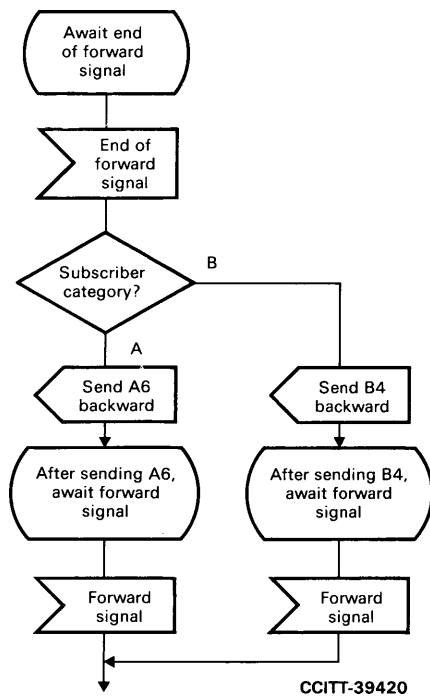


FIGURE C-14  
Example of part of an  
SDL diagram before merging  
states

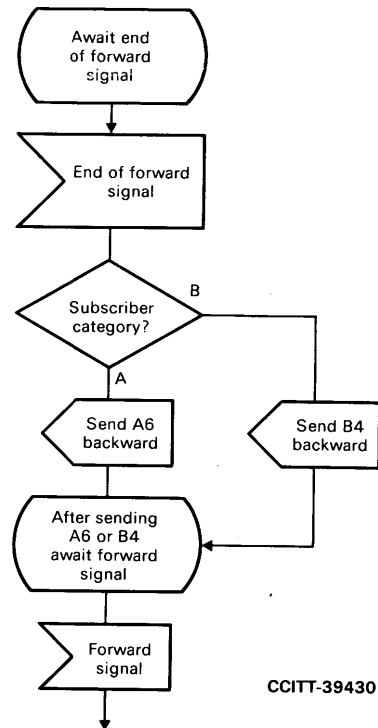


FIGURE C-15  
Example of a part of an  
SDL diagram with merged  
states

### C.6.3 Inputs

§ C.6.3 has been written to explain the input concept and the use of inputs in SDL diagrams without the save concept. The save concept and the use of inputs and saves together in an SDL diagram is covered in § C.6.4.

#### C.6.3.1 General

An input symbol attached to a state means that if the signal named within the input symbol arrives while the process is in this state the transition, which follows the input symbol, should be executed. When a signal has triggered the execution of a transition, the signal no longer exists and is said to have been consumed. However any associated data carried by the signal is available to the process.

In SDL it is unnecessary to draw input symbols to represent signals whose arrival would necessitate a null transition (i.e. a transition containing no actions and which leads back to the same state). The convention is that for any signal not shown in an explicit input symbol at a particular state, there exists at that state an implicit input symbol and null transition. By this convention, the two diagrams shown in Figures C-16 are logically equivalent, and either can be used.

Where a number of inputs lead to the same transition, all the relevant signal names may be placed within one input symbol. Figure C-17 illustrates this point and the two diagrams are logically equivalent. If this convention is used, care must be taken to distinguish the individual signal names by appropriate formatting.

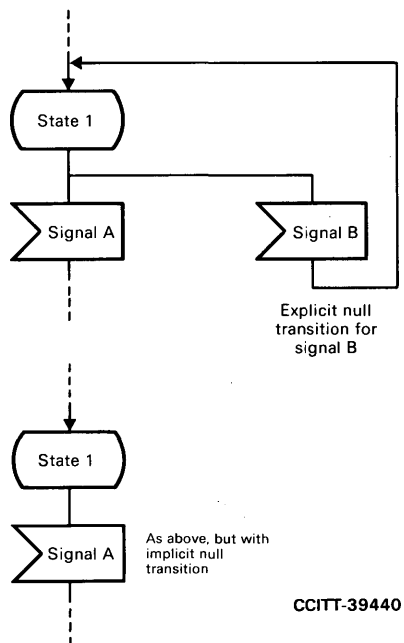


FIGURE C-16  
Explicit and implicit representation  
of a "null" transition

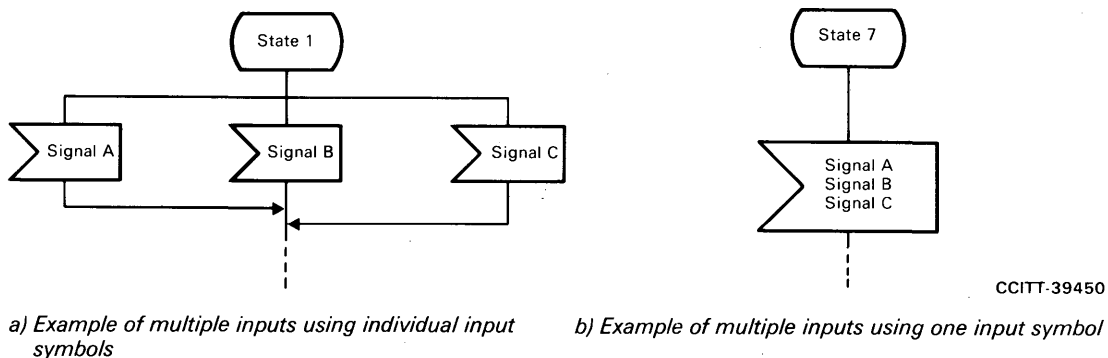


FIGURE C-17  
Alternative representations of multiple inputs

### C.6.3.2 Implicit queuing mechanism

The SDL Recommendations allow for the possibility of either zero or nonzero transition time. If the transition times are nonzero, then one or more signals may be waiting for consumption when a process reaches a new state and this means that signals must be queued in some way, or they will be lost. The SDL semantics define a first-in-first out conceptual queuing mechanism where signals are considered for consumption by a process in the order of their arrival at that process.

Figure C-18 uses the conceptual queue to explain the operation of the depicted SDL process where transition times are nonzero. It should be noted:

- The signals are consumed in the order in which they arrive.
- The sequence of signal arrival is important. Had 'C' arrived before 'B' in the transition between State 1 and State 2, the state sequence would have been 1, 2, 3 instead of 1, 2, 4.
- Because the queue is not empty when the process arrives at both State 2 and State 4, the process does not wait at either of the states.

If transition times are zero, then every signal will be consumed at the time it arrives at a process, except if the save operation is used (see § C.6.4).

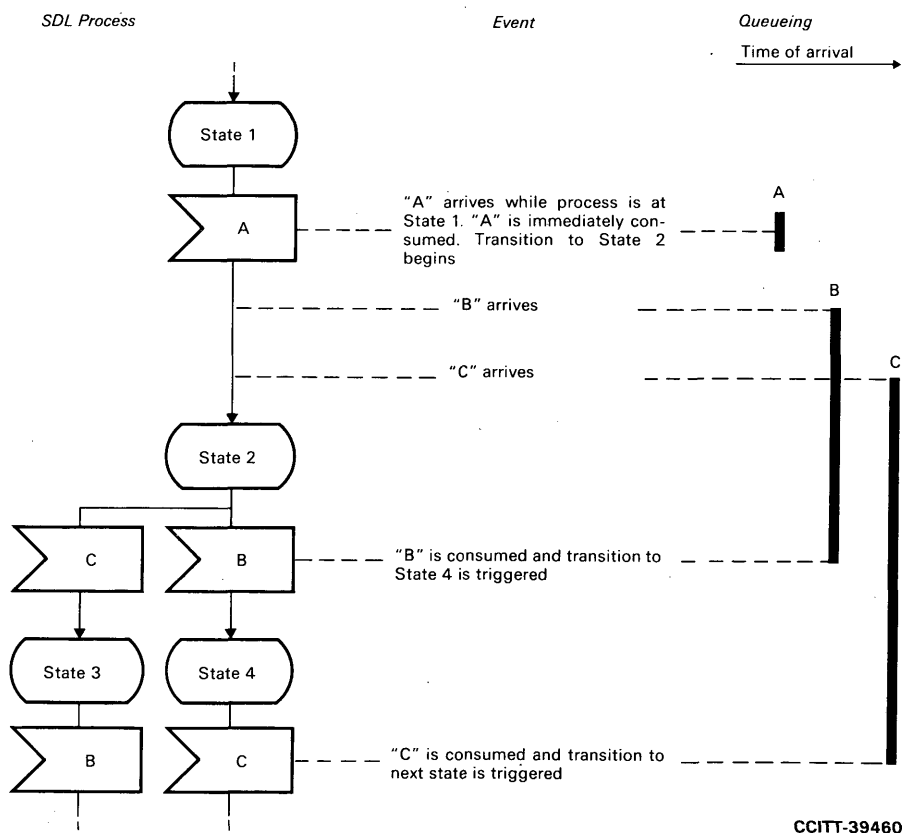


FIGURE C-18

Example of the operation of the implicit queuing mechanism assuming nonzero transition time

### C.6.3.3 Simultaneous arrival of signals

SDL semantics do not explicitly cover the possibility that signals can arrive simultaneously at one process; however, some SDL users may believe it necessary to take account of this possibility. It can be assumed that signals which arrive simultaneously are ordered before entering the queue and that this order remains the same while they are in the queue. The ordering strategy is not defined and therefore it should be ensured that the nature of the ordering strategy has no importance in relation to the "correct" operation of the SDL process.

If several signals are available when the process enters a state, only one signal is presented to the process and thus recognized as an input. The SDL semantics implies that the other signals are in fact retained.

### C.6.3.4 Signals with associated data

Signals may also carry associated data. For example, a signal named “digit” serves not only to trigger the execution of a transition by the receiving process, but also to carry with it the value of the digit (0-9) – information which can be used by the receiving process. The precise moment at which the information carried by a signal is made available for use by the process is not directly defined by SDL semantics, but it is implied that the information is made available at the moment of consumption of the signal.

### C.6.3.5 External vs. internal inputs

An external signal is a signal between processes of different functional blocks; an internal signal is a signal between processes in the same functional block. Accordingly, the recognition of an external signal is defined to be an external input and recognition of an internal signal is defined to be an internal input which has a slightly different symbol. Users who wish to avoid making this distinction should establish a one to one correspondence between processes and functional blocks.

### C.6.3.6 Signal lines

It may be useful to draw diagrams of two or more intercommunicating processes on the same sheet of paper. The “signalling” notation is available to show the signal flow between an output symbol in one diagram and a corresponding input in another diagram. An example of the use of signal lines is shown in Figure C-19. The SDL comment symbol can be used to serve the same purpose.

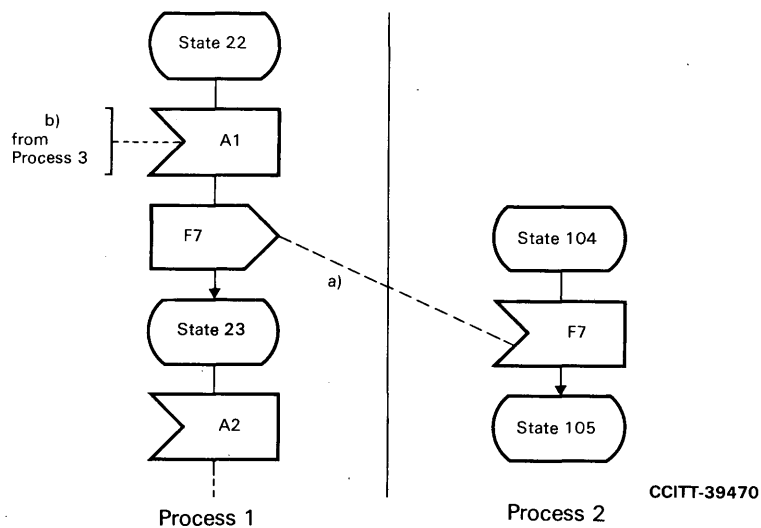


FIGURE C-19  
Example of the use of a signal line *a)*, and a comment *b)*,  
to define interprocess communication

### C.6.4 Saves

The save concept allows the consumption of a signal to be delayed until one or more other signals, which arrive subsequently, have been consumed. As discussed in § C.6.3, without the use of the save concept signals are consumed in the order in which they arrive.

At every state, every retained signal is treated in one of the following ways:

- it is shown as an input symbol or
- it is shown as a save symbol or
- (by convention) it is covered by an implicit input leading to an implicit null transition.

The operation of the implicit queueing mechanism, introduced in § C.6.3, can be expanded to cover the save concept. On arrival, signals enter the queue and when the process reaches a state, the signals in the queue are reviewed one at a time and in the order in which they arrived. A signal covered by an explicit, or implicit, input symbol is consumed and the related transition executed. A signal shown in a save symbol is not consumed and remains in the queue in the same sequential position and the next signal in the queue is considered.

Figure C-20 shows an example of an SDL process which incorporates a save symbol. It should be noted that signals S and R are consumed in the order R, S – the reverse order to that in which they were received. A single save symbol can save a signal only while the process is at the state where the symbol appears, and saves it for the duration of the transition to the next state. At the next state, the signal will be consumed via an explicit or implicit input (as shown in Figure C-20) unless either the save symbol with the signal name is repeated, or there happens to be another save signal available for consumption ahead of it in the implicit queue (as shown in Figure C-21).

A saved signal becomes available to a process only through a corresponding input symbol (explicit or implicit); in particular no questions about a saved signal may be asked in a decision prior to its recognition as an input.

At a state where more than one signal is to be saved, either a save symbol may be given for each signal, or they may all be shown within the one save symbol.

If several signals are to be saved, the semantics of the save symbol implies that the order of their arrival is preserved.

For saving signals, no distinction is made between “external” and “internal” signals, i.e. signals of both types can be saved using the same symbol type.

A third example of the use of the save is given in Figure C-22 with a description of the operation of the conceptual queueing mechanism in Figure C-23.

The use of save can simplify diagrams and in the specification can free the designer from the task of specifying details which would prejudice a particular realization.

Although the save can be used at every level of description, at lower levels it may be necessary to describe the actual mechanism which performs the save concepts.

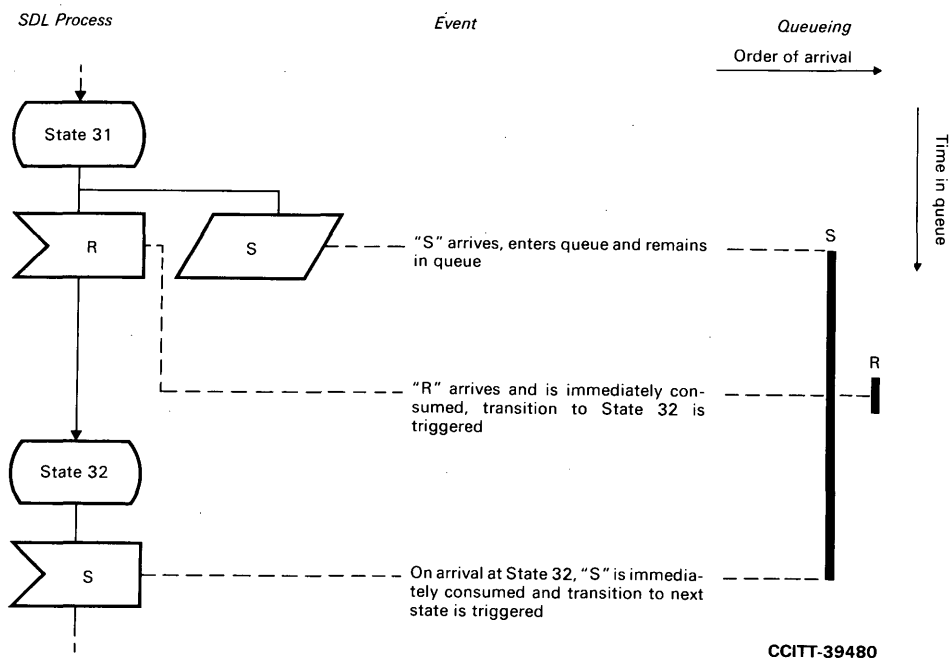
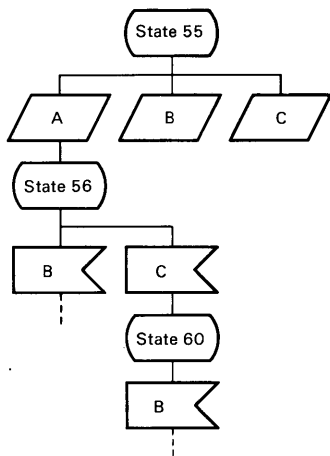
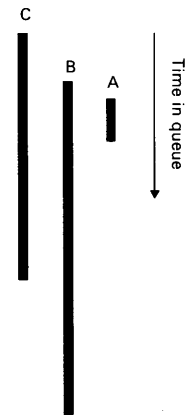


FIGURE C-20

Example of an SDL diagram with save symbol showing operation of implicit queueing mechanism

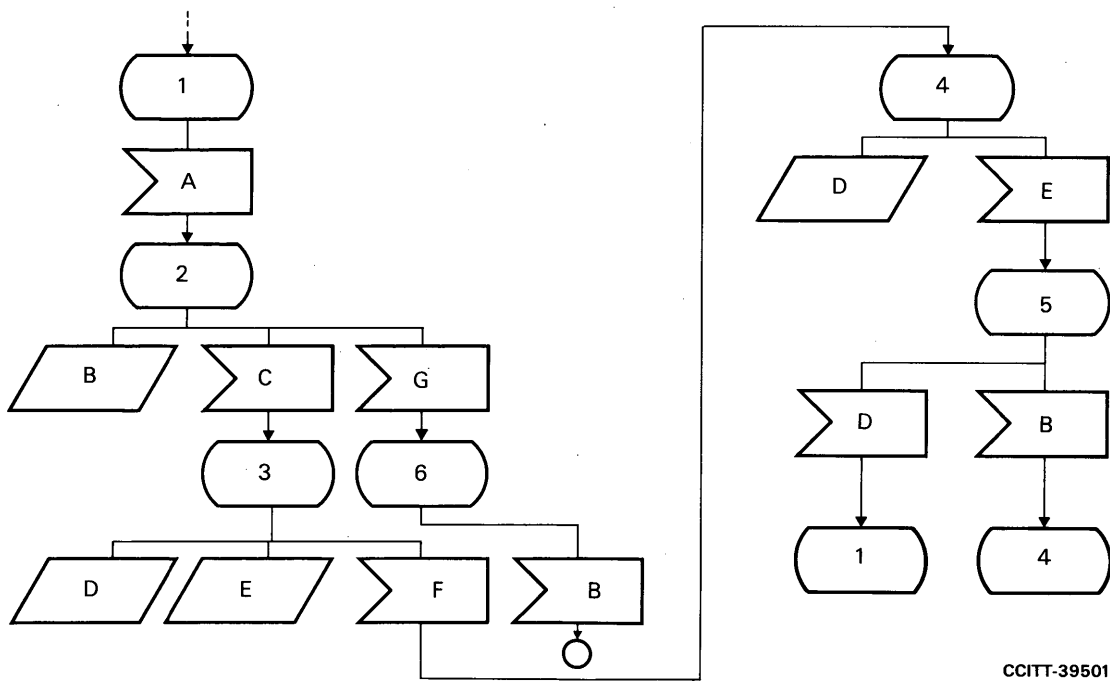


"C" arrives, enters queue and remains in queue  
 "B" arrives, enters queue and remains in queue  
 "A" arrives and is immediately consumed, transition to State 56 is triggered  
 On arrival at State 56, "C" is immediately consumed and transition to State 60 is triggered. "B" remains in the queue  
 On arrival at State 60, "B" is immediately consumed and the following transition is triggered



CCITT-39490

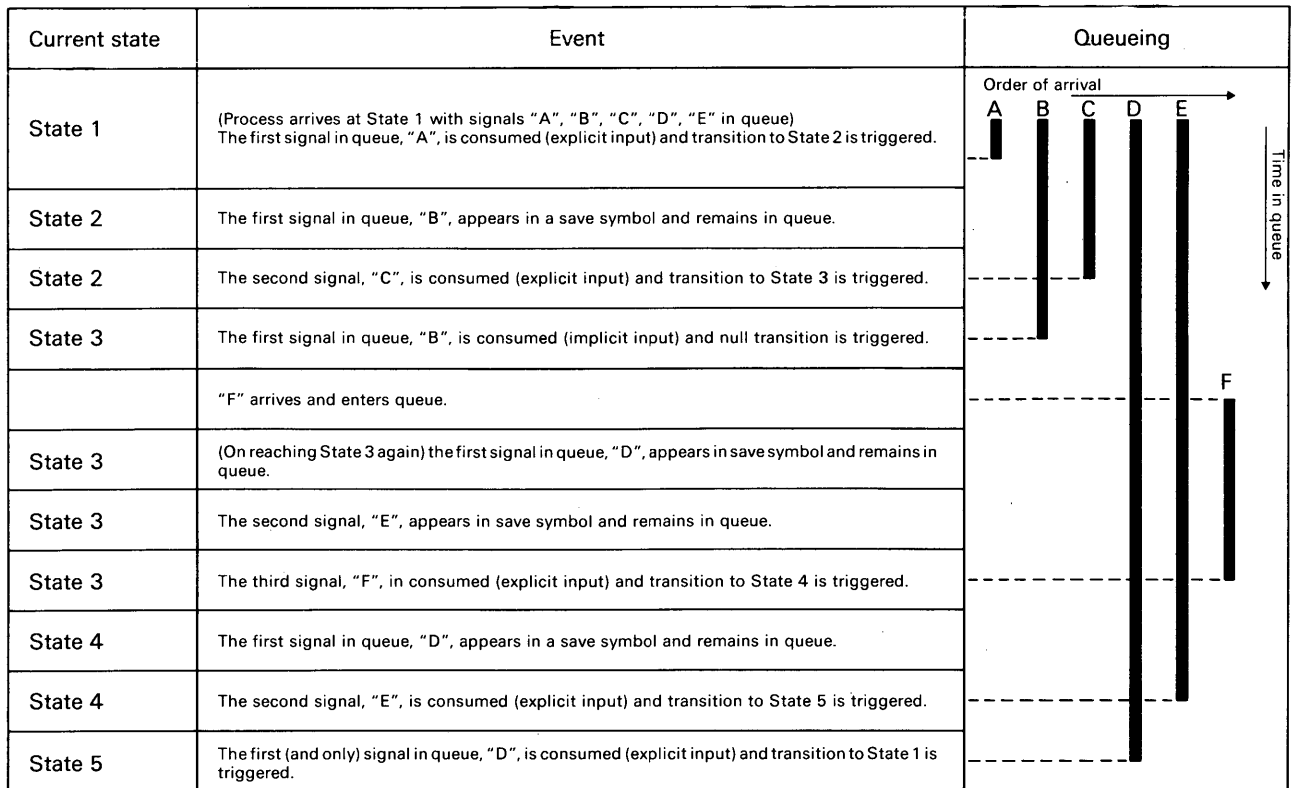
FIGURE C-21  
 Second example of the use of the save symbol



CCITT-39501

FIGURE C-22  
 Example of a more complex SDL diagram with numerous inputs and saves





CCITT-39510

FIGURE C-23  
Operation of conceptual queueing mechanism

### C.6.5 Outputs

An output symbol represents the sending of a signal from one process to another. Because control over the reception and consumption of the signal is associated with the receiving process (see § C.6.3) the semantics directly relating to the output symbol are relatively simple. From the point of view of the sending process an output can often be regarded as an instantaneous action which, once completed, has no further direct effect on the sending process which will not be directly aware of the fate of the signal.

In the early stages of producing an SDL diagram, the author may have some difficulty in deciding whether an action should be represented as an output or a task; this is discussed in § C.6.6.

An output symbol may also represent the sending of a signal which carries with it associated data (see § C.6.3.4).

An output can be external or internal (see § C.6.3.5).

### C.6.6 Tasks

A task symbol represents any action which a process can perform which is not an output or a decision. The nature of the tasks appearing in any diagram will be determined by the nature of the process being described and the level of detail required. Examples of actions which may be covered by tasks are:

- a) hardware actions such as connecting switch paths;
- b) the manipulation of data.

SDL users may occasionally have difficulty in deciding whether some aspect of the system being defined should be represented by a task or an output. Consider the example process shown in Figure C-24; should “connect switch path” be represented as a task or an output? If a separate switch path control process has not been identified then the task symbol would be appropriate. If a separate switch path control process is identified, then the use of the output symbol must be used.

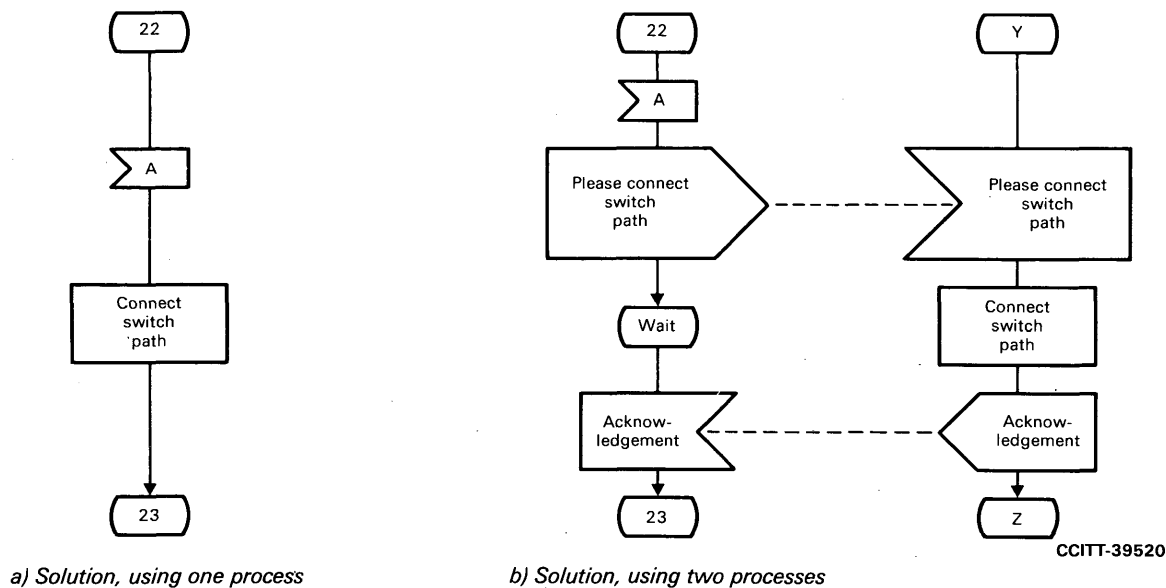


FIGURE C-24  
Two possible solutions for “connect switch path”

### C.6.7 Decisions

#### C.6.7.1 General

A decision is an action within a transition which asks a question to which the answer is available at the instant of executing the decision; the process follows one of the two or more paths following the decision according to the answer. SDL diagram authors should ensure that processes are defined so that they cannot attempt to execute decisions for which the answer (or data) is not available; such decisions would make the diagram invalid and could cause considerable confusion.

Data on which decisions are based have been explicitly stored using the task concept or may arrive at a process by signals which carry associated data (see § C.6.3.4).

#### C.6.7.2 Use of the decision symbol

The text of a question related to a certain decision is placed in the decision symbol. The symbol must have two or more branches. The answer to the question is placed directly beside the corresponding branch. Some possible formats are shown in Figure C-25.

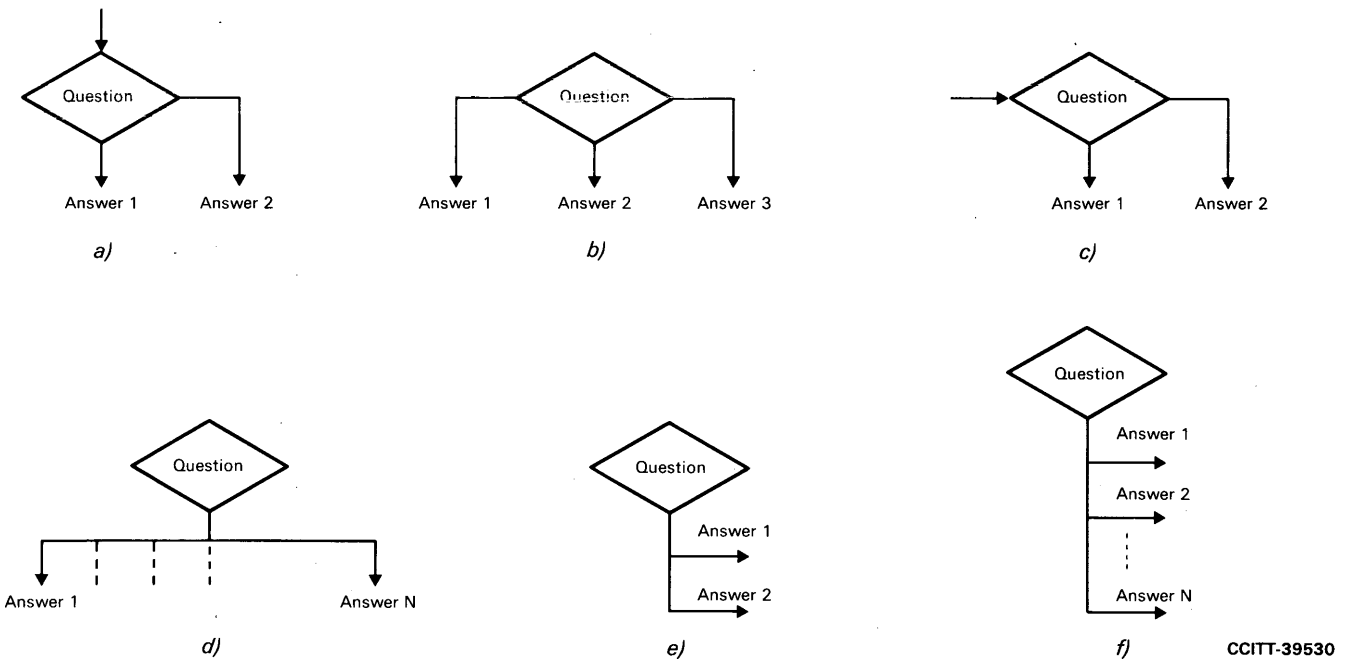


FIGURE C-25  
Possible formats for the decision symbol

CCITT-39530

Many questions are related to one specific condition in which the only possible answers are “Yes” or “No”.

For example:

B subscriber busy?  
 telephone set “on-hook”?  
 digit 2 received?  
 $Z = 1$ ?

In those cases the letters Y and N may be used to denote the answers YES or NO.

An example is given in Figure C-26.

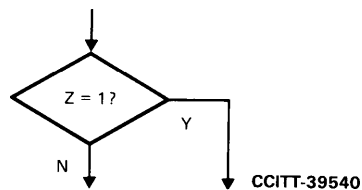


FIGURE C-26  
Formatting of a simple  
binary decision

In the case of more than two answers, the formats in Figure C-27 are possible, assuming  $Z$  to be a positive integer.

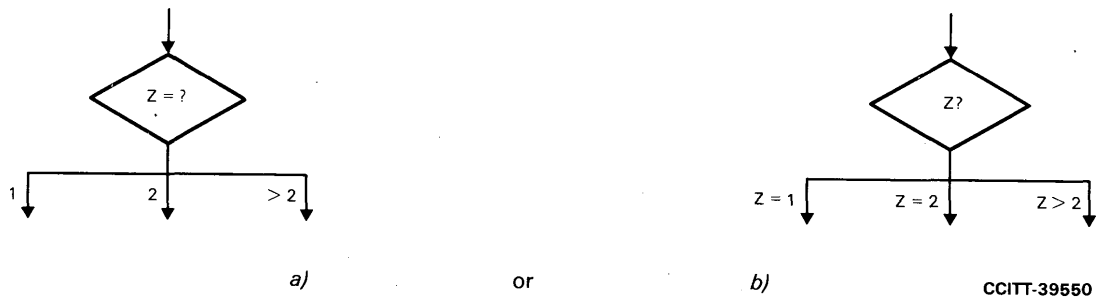


FIGURE C-27  
Formatting of questions and answers

Of course the decision in Figure C-27 can also be expressed by the diagram in Figure C-28, using 2 decisions.

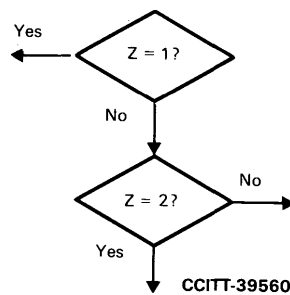


FIGURE C-28  
Alternative formatting  
of Figure C-27

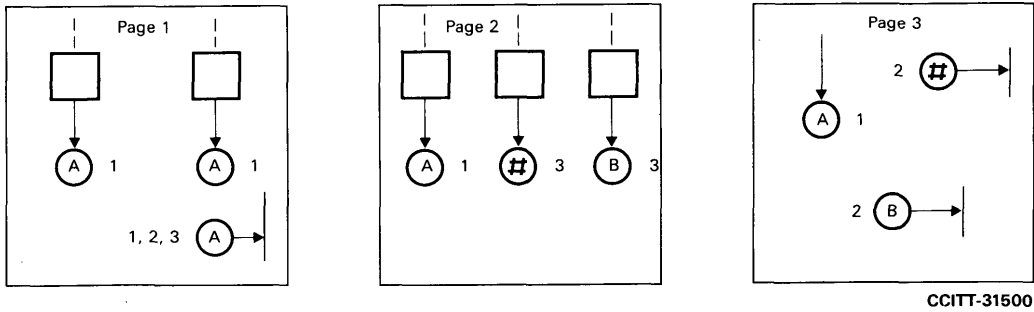
#### C.6.8 Use of connectors to show implied flow lines

Any flow line may be broken by a pair of associated connectors, with the flow assumed to be from the out-connector to the in-connector.

It is desirable that the page reference for the appropriate in-connector should be given next to each out-connector, and that the page reference(s) for the appropriate out-connector(s) should be given next to each in-connector.

Each in-connector label (within the symbol) should be unique within an SDL document for a given process.

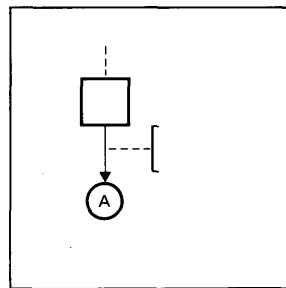
A connector symbol must not be used to imply the conveying of information between different processes. Output and input symbols should be used for that purpose.



CCITT-31500

*Note* – An alternative and equally acceptable, method of drawing page references is to use comments, as shown in Figure C-30.

FIGURE C-29  
Method of drawing page references



CCITT-39580

FIGURE C-30  
Alternative method of  
drawing page references

## C.6.9 Use of divergence and convergence

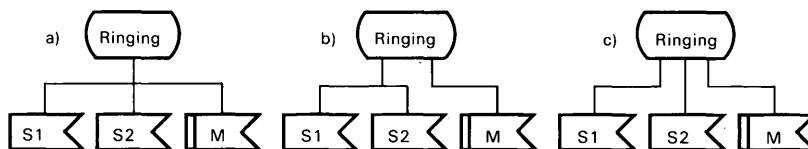
### C.6.9.1 Divergence

Divergence within a transition of an SDL diagram may only occur:

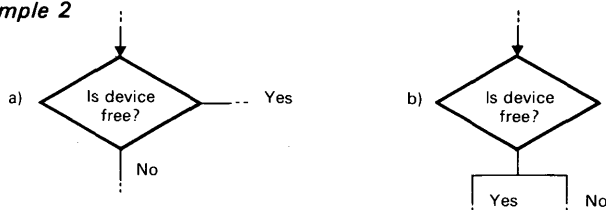
- a) between a state symbol and its associated input and save symbols, or
- b) immediately after a decision symbol.

Each of the examples in Figure C-31 contain logically equivalent SDL representations.

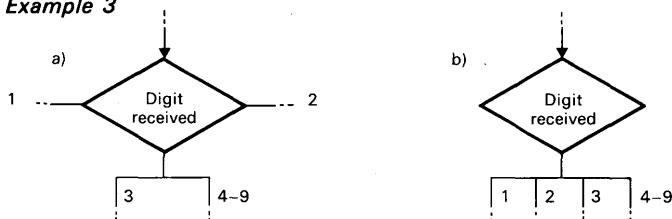
**Example 1**



**Example 2**



**Example 3**



CCITT-39590

**FIGURE C-31**  
**Examples of divergence**

**C.6.9.2 Convergence**

The point of convergence cannot occur between a state symbol and an input or save symbol, but may occur at any other point in an SDL diagram.

Convergence may appear in any of the four ways shown in Figure C-32.

The use of convergence can reduce the number of symbols in an SDL diagram where a sequence of symbols and any associated text is repeated within that same SDL diagram.

**C.6.10 Comments**

Comments may be added to an SDL diagram either to clarify some part of the diagram or to provide information which cannot be provided using SDL symbols.

A comment is attached to a flow line or SDL symbol by a single square bracket connected by a dashed line to the flow line or symbol. See Figure C-34.

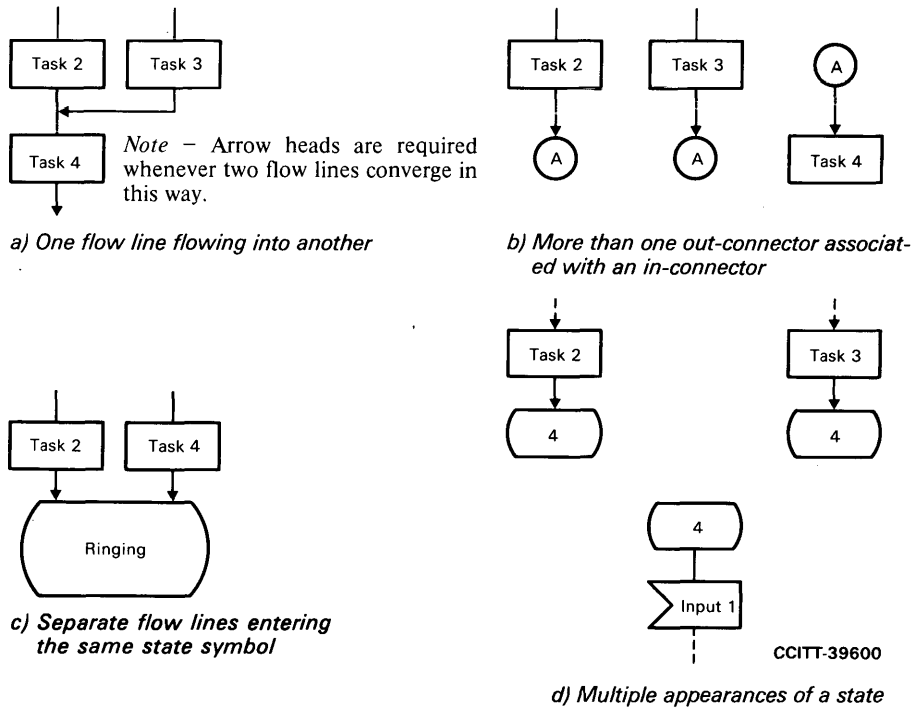


FIGURE C-32  
Examples of convergence

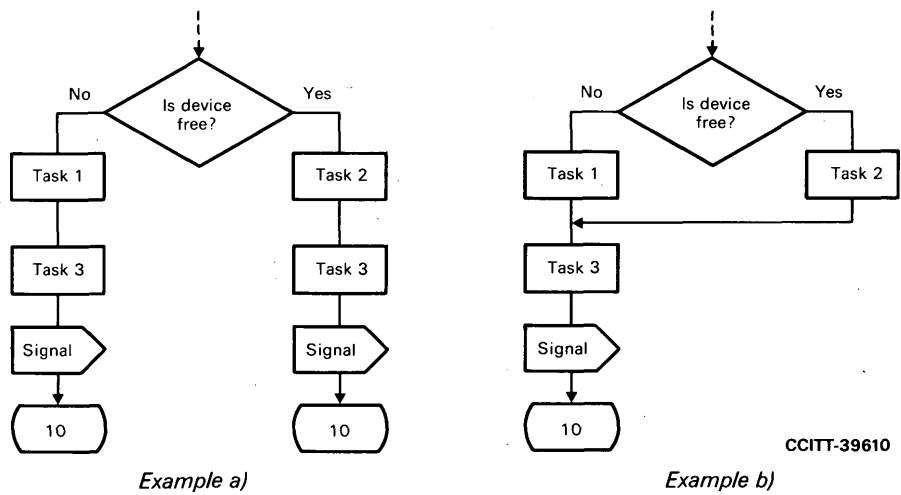


FIGURE C-33  
Use of convergence  
Example b) is equivalent to Example a)

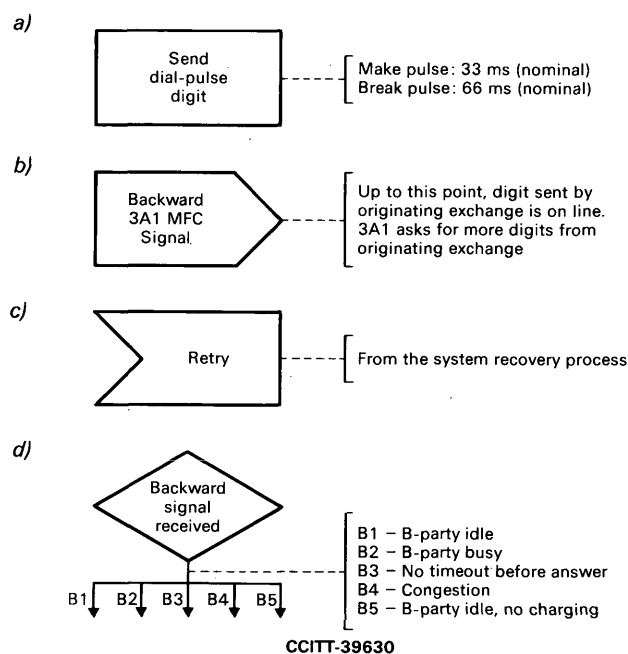


FIGURE C-34  
Examples of use of comments

### C.6.11 Drawing a complete SDL diagram

#### C.6.11.1 Starting and ending of a diagram

SDL diagrams must start and end (if applicable) with state symbols.

It is normally preferable to draw an SDL diagram with the flow from the top to the bottom of the page.

#### C.6.11.2 Alternative arrangements of a diagram

The suspension of a process is represented by the state symbols. Whether these suspensions should be emphasized (by disconnecting the diagrams starting at each state) or not, is left to the SDL-user. According to the SDL rules there are three possible methods:

- a) *Method A* – The diagram is interrupted at every state symbol and will continue again with that state symbol

Using Method A, the complete SDL diagram consists of a set of  $n$  subdiagrams, where  $n$  is the number of states of the process. Each subdiagram begins with a different initial state, shows all transitions from that state and ends with the final states of those transitions. The same final state can appear in several subdiagrams.

An example is shown in Figure C-35.

Note that the transitions, together with the inputs, are not explicitly shown.



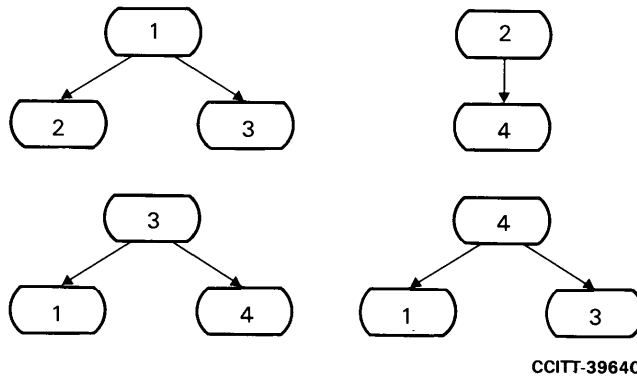


FIGURE C-35  
Method A

b) *Method B – The diagram is nowhere interrupted*

In Method B the SDL diagram is drawn as a fully connected graph, where each state appears only once. The example of Figure C-35 is redrawn in Figure C-36 as a fully connected graph.

A variation of this method, where one state only (for instance the idle state) appears several times in a diagram, is in common use.

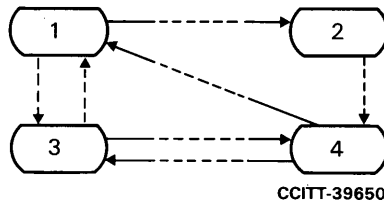


FIGURE C-36  
Method B

c) *Method C – The diagram is interrupted at some state symbols*

If the diagram is interrupted with some state symbols only, there are several equivalent ways of redrawing the Figure C-35. One of these, consisting of two subdiagrams, is shown in Figure C-37.

In Method C some states will appear in several subdiagrams. Using this method it is of course advisable to draw those parts of the diagram connected which belong logically together. For instance, it is very common that a relatively small part of the process represents the behaviour of most usual situations. This part could be suitably drawn in a separate subdiagram.

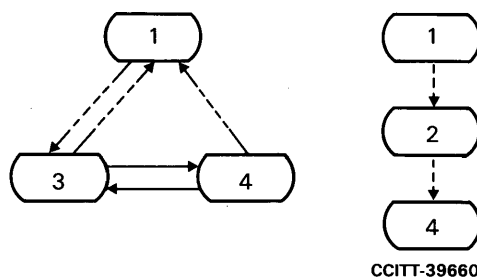


FIGURE C-37  
Method C

### C.6.11.2.1 Comparison of the three methods

The advantage of the Method A lies in the minimal time spent deliberating how to draw a diagram and in the ease of modification and the ease of information retrieval.

The advantage of the Method B lies in the total process overview, whereas the Method C leads to the structuring of the process according to predetermined criteria, as to which parts of the process are of greatest interest.

### C.6.11.2.2 Page references for multiple appearing states

If a state appears more than once in a diagram, page references annotated to these state symbols are necessary.

There are two types of multiple appearances of states:

- a) the connector-like type. (Final states of transitions can appear several times, but initial states appear only once.);
- b) the structuring type. (Initial states also appear several times, each time displaying only a subset of the possible inputs and saves at that particular state.)

It might be desirable to distinguish between page references for the above two types of multiple appearances. This could be done by putting page references for the multiple appearing initial states in parentheses. Examples are shown in Figure C-38.

Alternatively, page references could be included in auxiliary documentation. This alternative might be especially suitable in the case of large complex diagrams.

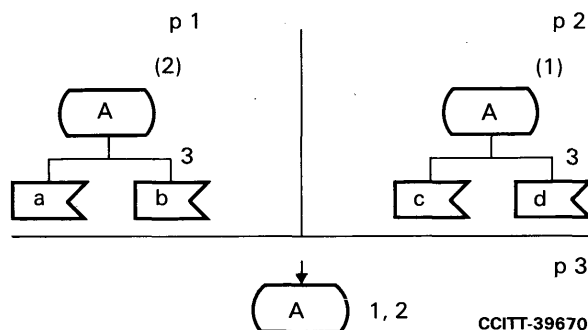


FIGURE C-38  
Example showing page references to other appearances of the same state

### C.6.11.3 Use of connectors

In drawing a large SDL diagram, it also may be necessary to interrupt the diagram due to the lack of space. Connectors are used for this purpose. (See also § C.6.8.)

Connectors should be used also to avoid the crossing of flow lines which would make diagrams somewhat unclear. It is normally preferable to draw an SDL diagram with the flow from the top to the bottom of the page.

If "major" flow paths exist, attention can be concentrated on them by not breaking them up with connectors except at the top and bottom of pages. A "minor" flow path within an SDL diagram can instead be broken with an out-connector, the associated in-connector being placed on some other suitable page, such as a page following the end of the major flow paths.

C.6.11.4 Clarity of the diagrams

Since SDL diagrams are a medium of communication between the authors and readers, the clarity of an SDL diagram is highly desirable. The clarity of the diagram depends not only on the method of arranging the diagram, described in § C.6.11.2, but also on the complexity of the process, (e.g. the number of states, inputs and decisions) which in turn is a function of partitioning.

With respect to the aim that diagrams should be easy to read and comprehend, some care should be taken to achieve a suitable layout for the diagrams.

In searching for the clearest method of representation, the author should be aware of different methods of using SDL. The examples shown in a), b) and c) of Figure C-39 can be considered to be alternative ways of representing the same part of similar processes in a given context.

Similarly, the user's attention is drawn to Figure C-40 in which a) and b) show alternative ways of separating input signal information.

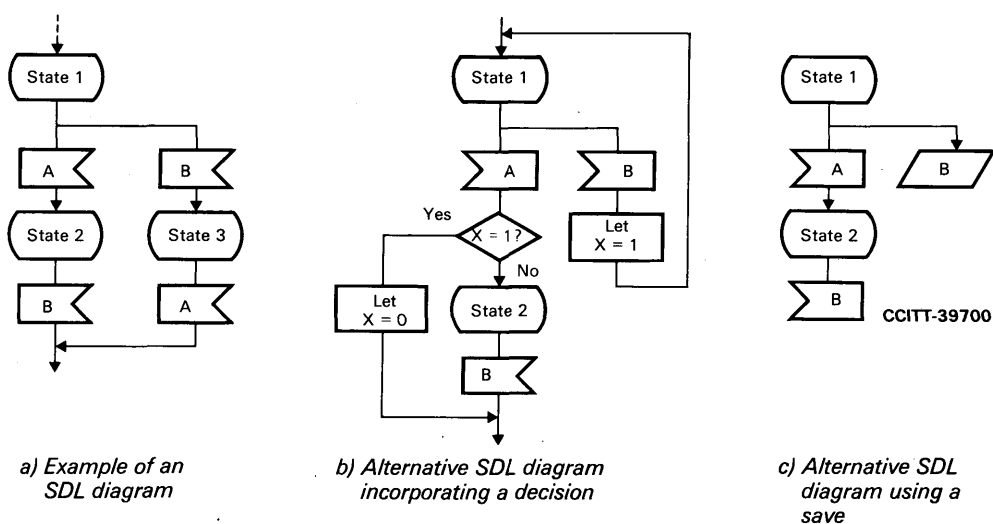


FIGURE C-39

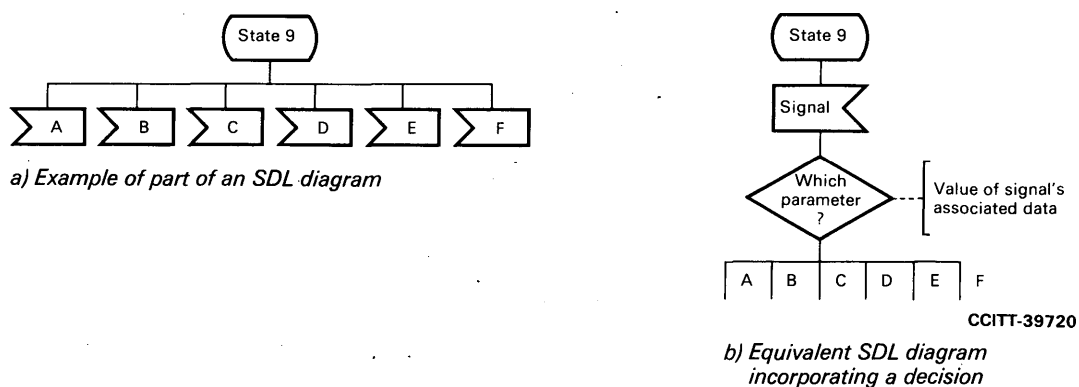


FIGURE C-40

C.6.12 Treatment of data in SDL

C.6.12.1 General

In SDL, communication between processes takes place with the use of signals. The signal is initiated by a process with the use of an output. This signal is a flow of data conveying information to another particular process. If the receiving process recognizes this signal, by means of an input, the data of this signal becomes available to this process.

From the point of view of a particular process, and at a particular time, there are two major and complementary categories of data:

- 1) data available to this process.
- 2) data not available to this process. (This second category includes data retained but not yet available to this process in the form of an arriving signal.)

Only available data can be used by a process in performing any of its actions: decisions, tasks or outputs.

When used in a decision or output the data itself, at this level of abstraction, is not modified. A special task, however, can create, store, modify or destroy data.

If it is intended to preserve data arriving by signals for future use in other transitions, these data should be stored explicitly using a task.

C.6.12.2 Examples of treatment of data

- a) For a decision, see Figure C-41.
- b) For an output, see Figure C-42.
- c) For a task, see Figure C-43.

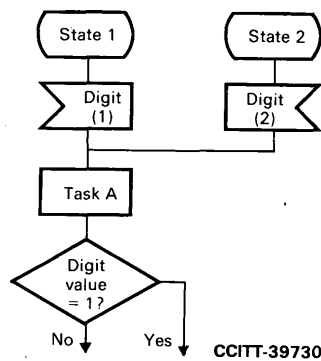
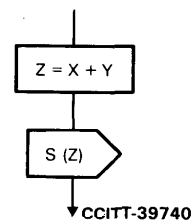
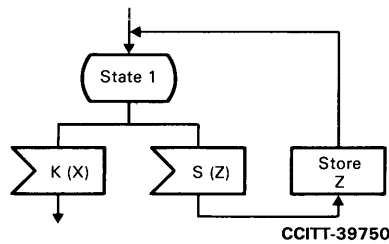


FIGURE C-41  
Questioning of data associated with inputs  
by means of a decision



Note - S is the signal name. Z is the associated data.

FIGURE C-42  
Sending data by means of an output




Note – S and K are signal names. Z and X are the associated data.

FIGURE C-43  
Storing incoming data for future use by means of a task


### C.6.13 Construction of state pictures

#### C.6.13.1 State picture option

If the state picture option is chosen, it is useful to consider each state picture as consisting of the following three types of elements:

- a) pictorial elements, e.g.  (meaning a signalling receiver);
- b) input variables, e.g.  $\bar{f}$  (meaning that a forward signal has yet to be recognized); and
- c) qualifying text, e.g. MFC (meaning multi-frequency code).

These three types of elements are often combined to form a composite pictorial element that can be understood well in isolation from the rest of a state picture.

e.g.   $\bar{f}$  (meaning a multi-frequency code receiver that is awaiting a forward signal)

Note that the recommended pictorial element symbol for time supervising of a process incorporates the relevant input variable  $t_i$ .

#### C.6.13.2 Input variables

Input variables are a valuable way of representing those conditions associated with the process which, if changed, will result in a transition by the process. Input variables should be shown using lower-case letters, in order to distinguish them readily from qualifying text, which should be shown using upper-case letters. (Changes in input variables correspond to input signals, which cause the process to cease its current state; changes in qualifying text do not correspond to input signals.)

#### C.6.13.3 Qualifying text

Qualifying text should be heavily abbreviated, and where possible should be placed inside appropriate pictorial elements, so that it is quite obvious which pictorial elements are being qualified.

#### C.6.13.4 Completeness of state pictures

Sufficient pictorial elements should be assigned to each state picture in order to show:

- what resources are being considered by this process during the current state. Examples of resources are: terminal equipment, switching paths, signalling receivers, signalling senders, switching modules and control elements;
- whether one or more timers are currently supervising this process or not;
- in the case that this process concerns call-handling, whether call charging is currently in progress or not, and which subscribers are being charged during this state of the call;
- the current categories (or status) of equipment allocated to this process, where this information affects the behaviour of this process;
- the status of those input variables which are monitored by the process during the current state.

#### C.6.13.5 Example

An example of the application of the principles expressed above is illustrated by the state picture in Figure C-44 which has been extracted from the larger example Figure A-3 in Annex A to Recommendations Z.101-104. It will be seen that during this state:

- the resources considered by the process consist of a subscriber's handset, a digit receiver, a dial tone sender and switching paths connecting these items;
- that a timer  $T_0$  (whose current condition is  $t_0$ ) is currently supervising this process;
- that no charging of the call is in progress;
- that the subscriber has been identified as an A-party, but no other category information is taken into account;
- that the following input variable conditions apply:  $h_A$  (i.e. the handset is off-hook,  $\bar{d}$  (i.e. the digit receiver is awaiting a digit) and  $t_0$  (i.e. the supervising timer  $T_0$  is running).

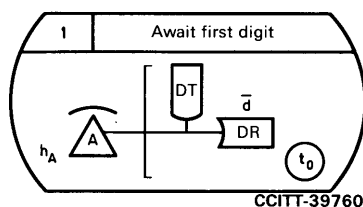


FIGURE C-44  
Example of a state in a  
call-handling process

#### C.6.13.6 Consistency checking feature of SDL diagrams with pictorial elements

If the principle expressed in § C.6.13.4 e) is followed, it is always possible to deduce the set of inputs which will cause the process to leave each of its states, simply by looking at the input variable shown in the state picture. For example, from looking at the state picture in Figure C-44, one can deduce that three different inputs will cause the process to leave this state: the handset condition changing to on-hook (input  $\bar{h}$ ); the arrival of a digit (input  $d$ ); or the expiry of timer  $T_0$  (input  $\bar{t}_0$ ). In this way, some "surprise transitions" can be avoided when implementing systems from very complex specifications, using the SDL.

### C.6.13.7 Use of the functional block boundary symbol

Pictorial elements shown outside the functional block boundary symbol imply elements that are not directly controlled by the given process, and pictorial elements shown within the functional block boundary symbol imply elements that are controlled directly by the given process. For example, the call process that is partially specified in Figure C-45 can connect or disconnect ring current and start or stop timer  $T_4$ , but it cannot change either subscriber's handset condition.

In designing logic from an SDL specification with pictorial elements, only those pictorial elements shown inside the functional block boundary will affect the processing actions performed during transition sequences. The composite pictorial elements shown outside the functional block boundary are normally included in a state picture:

- a) because they indicate input variables which must be monitored by the process during the given state; and/or
- b) to improve the intelligibility of the diagram.

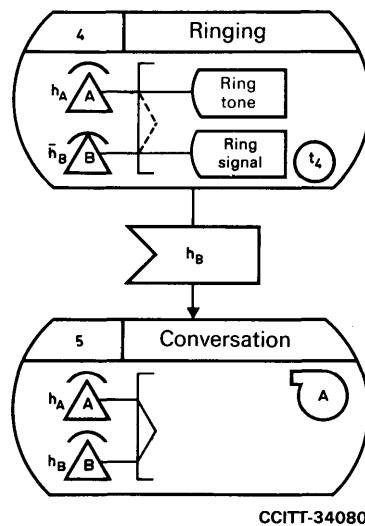


FIGURE C-45

Example of a transition between two states, where all processing actions are implied by the differences in the state pictures

### C.6.13.8 Task or output

The interpretation of a processing action as either a task or an output sometimes appears to be arbitrary. In fact the decision to interpret the appearance or disappearance of a pictorial element within the functional block boundary as either a task or an output can only be resolved by consulting the process signal list.

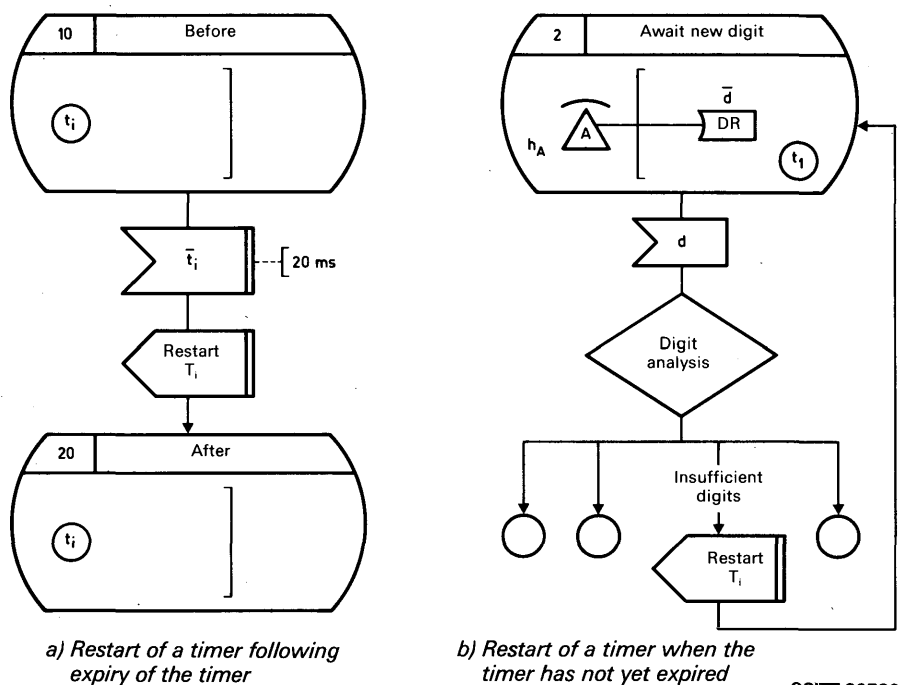
### C.6.13.9 Use of the timer symbol

A timing process is considered to be essentially carried out concurrently with the process that it is supervising, and for that reason the SDL when used without pictorial elements shows the starting or stopping of timing processes ("timers") by means of outputs. Whether or not pictorial elements are used, the interruption of the supervised process upon expiry of the timer is always represented by an input.

The presence of a timer symbol in a state picture implies that a timer is running during that state. Following the general principle stated in § 3.1.3 of Recommendation Z.103, the starting, stopping, re-starting and expiry of timers is shown using pictorial elements in the following manner:

- To show that a timer is started during a given transition, the timer symbol should appear in the state picture corresponding to the end of that transition but not in the state picture corresponding to the start of that transition.
- To show that a timer has been stopped during a transition, the timer symbol should appear in the state picture corresponding to the start of that transition but not in the state picture corresponding to the end of that transition.
- To show that a timer has been restarted during a transition, an explicit output symbol should be shown in that transition. (Two examples are shown in Figure C-46).
- The expiry of a particular timer is shown by an input symbol associated with a state in which the state picture includes the corresponding timer symbol. Of course more than one timer may concurrently supervise the same process, if required (see Figure C-47).

In the examples shown in Figures C-46 and C-47, it has been assumed that the timing processes belong to the same functional block as the supervised process, and hence internal input and internal output symbols have been used. If the timing processes are assumed to belong to different functional blocks from that of the supervised process, then external input and external output symbols should be used.

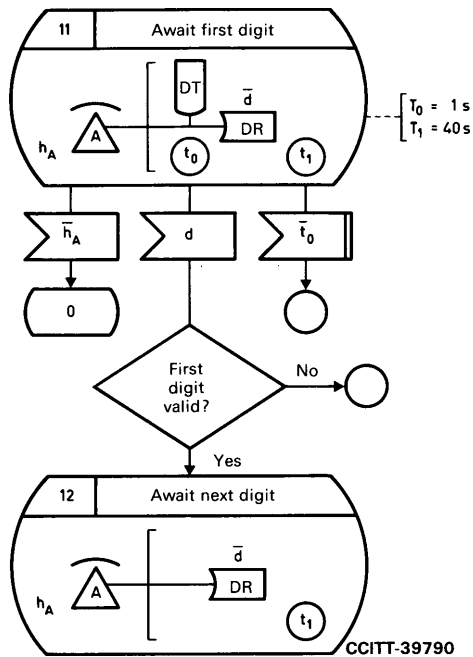


CCITT-39780

Note - Each timer  $T_i$  has two mutually exclusive conditions  $t_i$  and  $\bar{t}_i$ .

FIGURE C-46  
Examples showing the restart of a timer





Timer  $T_0$  supervises the arrival of the first digit, whereupon dial tone is removed from the call and timer  $T_0$  is stopped. Timer  $T_1$  continues supervising the arrival of sufficient digits to permit routing of the call.

FIGURE C-47

Example of the use of two supervising timers  
in the same state

**PART II**

**Recommendations Z.311 to Z.341**

**MAN-MACHINE LANGUAGE (MML)**

**PAGE INTENTIONALLY LEFT BLANK**

**PAGE LAISSEE EN BLANC INTENTIONNELLEMENT**

## SECTION 1

### GENERAL PRINCIPLES

#### Recommendation Z.311

#### 1. INTRODUCTION

##### 1.1 *Field of application*

The man-machine language (CCITT MML) can be used to facilitate operation and maintenance functions of SPC switching systems of different types. According to different national requirements CCITT MML can also be used to facilitate installation and testing of such systems. The contribution of the man-machine language toward testing will also include diagnosis of hardware faults and hardware/software design deficiencies.

A list of such functions is given in Recommendation Z.318.

In many cases, SPC systems will be supported by auxiliary processors (e.g., in operation and maintenance centres and/or centres for other purposes, such as sales, subscribers' complaints, etc.) to carry out functions in cooperation with the SPC system. Different types of communication may be required for this cooperation. To clarify where the CCITT MML is intended to be used, a network configuration is shown in Figure 1/Z.311 which shows the case of three separate processors. Local and remote man-machine terminals may be used (remote including the international case). The configuration of processors in a system may vary but this has no effect on the principles governing the field of application of the MML.

The CCITT MML is intended to handle the functions required at the interface marked 1 while other methods may be required for the interface marked 2. Interface 2 is not considered.

##### 1.2 *Basis of MML*

The MML contains inputs (commands), outputs, control actions and procedures sufficient to ensure that all relevant functions for the operation, maintenance, installation and testing of SPC systems can be performed.

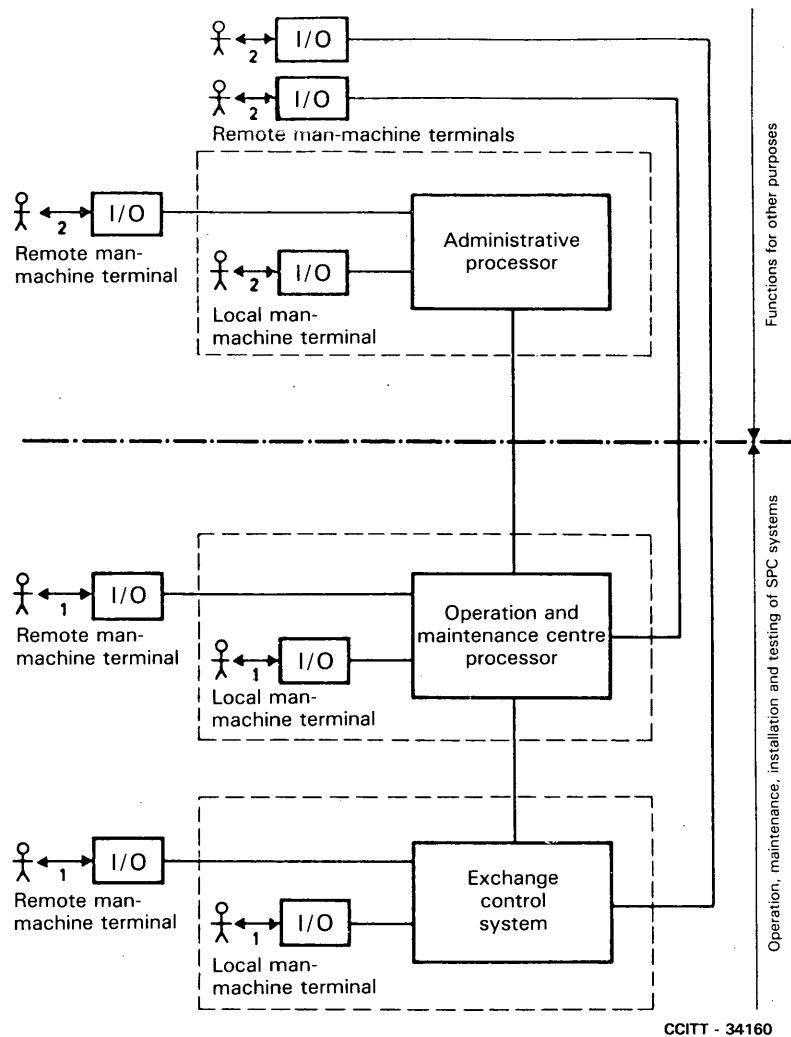
It is understood that some of these matters will arise by being initiated by man and some by being initiated by machine.

The basic attributes of the language are summarized in the following:

- a) The MML is easy to learn and to use. It is easy to input commands and to interpret outputs.
- b) Both the interactive and the menu modes of operation are available as well as direct and format parameter input.
- c) The MML is adaptable to different kinds of personnel and to different national language and organizational requirements.

According to different Administration requirements, the use of certain commands or procedures may be restricted by the implementation to certain staff, terminals, etc.

The MML should be implemented in such a way that errors in commands or control actions must not cause the system to stop, unduly alter the system configuration or take up undue resources.



1 = man-machine terminals where CCITT MML is recommended to be used  
 2 = man-machine terminals where the use of CCITT MML is not considered  
 I/O = input/output

FIGURE 1/Z.311

### 1.3 Input/output

Input can be from any device that produces the code of the characters of the CCITT International Alphabet No. 5.

A keyboard input device will normally be used; however, recorded input may be used (e.g., paper tape, magnetic tape, cassette, etc.).

Output can be to any device that accepts the code of the characters of the CCITT No. 5 alphabet (paper tape punchers, teletypewriters, line printers, visual display units, etc.).

Two basic formats F1 and F2 for printing are recommended (see Recommendation Z.312).

### 1.4 Extensibility and sub-setting

The MML has an open ended structure in such a way that any new function or requirement added will have no influence on the existing ones.

The language structure is such that sub-sets can be created. Sub-setting may be for various purposes, e.g., staff sub-sets, in which the selection is done for needs of certain sections of staff; application sub-sets, in which selection is made for convenience of application; etc.

## 1.5 *Organization of the Recommendations*

The Recommendations Z.311 to Z.317 define the full syntax and operational aspects of the MML and together form a complete Recommendation.

Recommendation Z.312 discusses formats that are defined for use.

Recommendation Z.313 describes the meta-language defining the syntax diagrams used in Recommendations Z.314, Z.315, Z.316 and Z.317.

Recommendation Z.314 describes the CCITT MML character set and identifies the use of some of the special characters. It also defines the basic elements used in the syntax.

The three Recommendations Z.315, Z.316 and Z.317 together define the syntax and dialogue procedures. In each case both text and syntax diagrams are used. For any element the text and the related syntax diagram together constitute the complete definition of the element.

Recommendation Z.315 describes all the basic syntactic elements used for input and also shows the combinational aspects of input, i.e. the grouping of syntactic elements required to initiate functions via the MML. In this and the following Recommendations a variety of syntax options is shown. It is to be noted that options must be chosen in any one system in a consistent manner.

Recommendation Z.316 describes output outside dialogue in terms of the specific elements and the allowable combinations. In the syntax diagrams where an element is used which is the same as one defined in Recommendation Z.315, reference is made to that Recommendation rather than duplicating the definition.

Recommendation Z.317 describes a variety of dialogue procedures in which special operating sequences are defined in terms of both input and output elements as described in Recommendations Z.315 and Z.316, and also special elements to control the dialogue procedures.

## **Recommendation Z.312**

## **2. BASIC FORM LAYOUT**

### 2.1 *General*

To facilitate filing and retrieval of recorded information in MML, it is recommended that this information should be recorded on forms or pages with an identification header at the top of each page. The top and bottom line of the page should not be used but should be left empty.

It is further recommended that the layout for printing information in MML should be based on a maximum of 72 characters per line, and 66 lines per form, as this format can be accommodated on both the A4 and the 11-inch standard size paper and can be printed by standard teletypewriters.

Where a number of characters/line in excess of 72 is required, a second format is recommended. This accommodates 120 characters/line and would be used, for example, on typewriters and line printers.

In order to save paper or where paging to facilitate filing of output is not required, paging may be suppressed by suppressing the generation of all superfluous line feeds.

To distinguish between the formats recommended, they are further indicated as format F1 for the paper sizes A4 and A5L and format F2 for the paper size A4L. In the recommended formats specified below, International Standard ISO/2784 [1] has been taken into account.

### 2.2 *Recommended formats for presenting information in MML*

#### 2.2.1 *Format F1*

According to this format, which is based on the A4 and the 11-inch standard size paper, the maximum number of characters per line is 72. The number of lines per form may be 66, using the full 11-inch and A4 paper sizes or 33, using half the paper size (5.5 inch or A5L).

Information presented in this format can be displayed on most of the visual alpha-numerical displays available on the market. However, the number of lines which can be displayed at the same time on these devices is, in general, not more than 20 to 25 lines.

### 2.2.2 *Format F2*

This format allows a maximum of 120 characters printed on a line and has 66 lines per form. It can be accommodated on paper having a width equal to the A4L standard.

#### **Reference**

- [1] *Continuous forms used for information processing. Sizes and Sprocket feed holes*, ISO 2784-1974.

### **Recommendation Z.313**

## **3. THE META-LANGUAGE FOR DESCRIBING THE SYNTAX AND PROCEDURES**

### 3.1 *Introduction*

Syntax diagrams are a method of defining language syntax.<sup>1)</sup> A syntax diagram consists of terminal and non-terminal symbol boxes connected by flowlines. An annotation symbol is used to insert comments. The syntax of a language can be defined by a series of syntax diagrams. Each diagram defines a particular non-terminal symbol. In the MML Recommendations syntax diagrams are used to assist in specifying the syntax of the MML input, MML output and the man-machine dialogue procedures. A path through a syntax diagram defines an MML input, an MML output or a man-machine dialogue structure.

The following describes the use of syntax diagrams and states a set of rules for their use.

### 3.2 *Terminology*

3.2.1 Terminal symbols are those characters or strings of characters which actually appear in the input or output. The character set to be used in CCITT MML is described in Recommendation Z.314.

3.2.2 A non-terminal symbol does not immediately appear in MML input or MML output; it represents, within a syntax diagram, another syntax diagram by name. Hence it is an abbreviated symbol for a more complex construct (consisting of a set of terminal and/or non-terminal symbols) used in several places.

3.2.3 Annotation symbols (see § 3.3.7 below) are used to insert comments. For example, they may be used to indicate mutually exclusive paths through a diagram.

### 3.3 *Rules*

3.3.1 Every symbol box (terminal or non-terminal) and consequently each diagram must have one, and one only, entry and one, and one only, exit flowline.

3.3.2 Each diagram must fit on a single page. Thus there is no off-page connector symbol.

3.3.3 Flowlines are always unidirectional. The preferred direction for flowlines which select alternatives is down. The preferred direction for flowlines which connect symbols is left-to-right. The preferred direction for flowlines which indicate repetitions (loops) is counterclockwise.

3.3.4 An arrowhead is required wherever any two flowlines come together, and wherever a flowline enters a symbol box. Additional arrowheads may be inserted wherever it is felt that this will improve the clarity of the diagram.

---

<sup>1)</sup> The syntax diagrams used in MML are based on those used to describe the programming language PASCAL [1].

3.3.5 Terminal symbols are surrounded by boxes with rounded corners. The width of the box is proportional to the number of characters contained in the box. For short terminal symbols, the box may become a circle. Symbols representing system input are surrounded by a single solid line and those representing system output by a double solid line. For terminal input symbols see Figure 1a)/Z.313 and Figure 1b)/Z.313. For terminal output symbols see Figure 1c)/Z.313 and Figure 1d)/Z.313.

3.3.6 Non-terminal symbols are surrounded by rectangular boxes. The name of the non-terminal symbol must be written in lower case characters. Every non-terminal symbol must have an associated syntax diagram except where the symbol is annotated "Not further expanded in diagram form". The non-terminal symbol used to name a particular syntax diagram must appear underlined at the upper left corner of the diagram. Symbols representing system input are surrounded by a single solid line, those representing system output by a double solid line and symbols representing a combination of input and output by an outer solid and an inner dashed line:

- a) non-terminal input symbol see Figure 1e)/Z.313,
- b) non-terminal output symbol see Figure 1f)/Z.313,
- c) non-terminal input/output symbol used in dialogue procedures see Figure 1g)/Z.313.

3.3.7 An annotation is denoted by the following symbol:

- - - [n

where n is a number referring to a comment. The text of the comment must be located at the foot of the diagram (this splitting of reference and text should help to reduce translation problems).

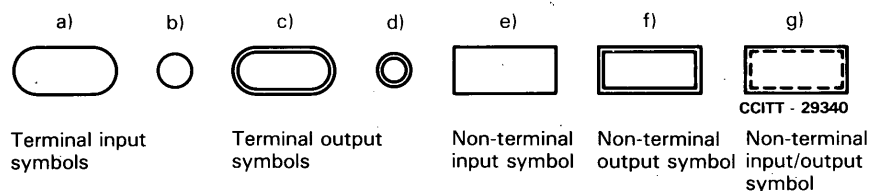


FIGURE 1/Z.313  
Terminal and non-terminal symbols to be used for  
the CCITT man-machine language

#### Reference

[1] JENSEN (K.), WIRTH (N.): PASCAL, User Manual and Report, *Springer Verlag*, New York, 1975.

#### Recommendation Z.314

### 4. THE CHARACTER SET AND BASIC ELEMENTS

#### 4.1 General

The character set and the basic elements used in the syntax are essential components of MML inputs, MML outputs and the man-machine dialogue procedures.

#### 4.2 The character set

The character set to be used for the CCITT MML is a sub-set of the CCITT International Alphabet No. 5 which has been established jointly by the CCITT and the International Organization for Standardization.

To allow for possible implementation of the CCITT MML using national languages, the sub-set is taken from the basic code table given in Recommendation V.3. [1] The code positions reserved in this table for national use are not contained in the basic character set of the CCITT MML, but may be used in these national implementations.



According to Recommendation V.3 [1] transmission control characters and information separators are intended to control or to facilitate transmission of information over telecommunication networks. Hence these control characters are not used in the MML. This will avoid interference with data transmission procedures when information in the MML is transmitted via a data transmission network.

It is furthermore recommended when information is printed or displayed that devices are used which print or display different graphic symbols for the digit zero and the capital letter O.

The characters selected for use in the CCITT MML are given in Table 1/Z.314.

TABLE 1/Z.314  
Character set to be used for the CCITT man-machine language

				b <sub>7</sub>	0	0	0	0	1	1	1	1
				b <sub>6</sub>	0	0	1	1	0	0	1	1
				b <sub>5</sub>	0	1	0	1	0	1	0	1
				Pos.	0	1	2	3	4	5	6	7
b <sub>4</sub>	b <sub>3</sub>	b <sub>2</sub>	b <sub>1</sub>									
0	0	0	0	0	NUL		SP	0	ⓐ	P	ⓐ	p
0	0	0	1	1		DC <sub>1</sub>	!	1	A	Q	a	q
0	0	1	0	2		DC <sub>2</sub>	"	2	B	R	b	r
0	0	1	1	3		DC <sub>3</sub>	#	3	C	S	c	s
0	1	0	0	4		DC <sub>4</sub>	\$	4	D	T	d	t
0	1	0	1	5			%	5	E	U	e	u
0	1	1	0	6			&	6	F	V	f	v
0	1	1	1	7	BEL		'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT (FE1)	EM	)	9	I	Y	i	y
1	0	1	0	10	LF (FE2)	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT (FE3)	ESC	+	;	K	ⓐ	k	ⓐ
1	1	0	0	12	FF (FE4)		,	<	L	ⓐ	l	ⓐ
1	1	0	1	13	CR (FE5)		-	=	M	ⓐ	m	ⓐ
1	1	1	0	14	SO		.	>	N	ⓐ	n	ⓐ
1	1	1	1	15	SI		/	?	O	_	o	DEL

ⓐ These positions are reserved for national use. CCITT - 26622

*General remarks* – The characters on the open positions are considered as outside the MML. They are implementation dependent and, together with the characters named in the table but not included in the MML, may be used in accordance with the rules given in Recommendation V.3 [1]. The position of a character in the table can be indicated by its column and row number, e.g. Pos. 3/1 gives the position of the digit 1 in the table. The table gives also the binary codes allocated to the table positions according to Recommendation V.3 [1]. The bits are identified by b<sub>7</sub>, b<sub>6</sub>, ... b<sub>1</sub>, where b<sub>7</sub> is the highest order, or most significant bit, and b<sub>1</sub> is the lowest order, or least significant bit.

#### 4.3 Summary of use of characters

The use of characters in the character set, except for letters, digits, and characters used solely as graphic characters and format effectors, is described in Table 2/Z.314. CCITT International Alphabet No. 5 code is indicated by position number (see Table 1/Z.314).

#### 4.3.1 *Letter*

A letter is one of the characters listed in Table 1/Z.314, columns 4, 5, 6 and 7. However positions 5/15 and 7/15 are excluded. The characters reserved for national use may be used as letters or as graphic characters.

#### 4.3.2 *Digit*

A digit is one of the characters listed in Table 1/Z.314, column 3, positions 0 to 9.

#### 4.3.3 *Graphic characters*

Graphic characters are a collection of characters one or more of which may be used to improve readability. Graphic characters which have other syntactic uses are listed in Table 2/Z.314. The \$ (position 2/4 in Table 1/Z.314) is the only character used solely as a graphic character.

#### 4.3.4 *Format effector*

The format effectors used in MML are the characters FE1 to FE5 and space as defined in Table 1/Z.314. The character BACK SPACE (FE0 in Recommendation V.3 [1]) is not regarded as a format effector in the MML.

#### 4.4 *Basic elements used in the syntax*

Syntax diagrams of the basic elements used in the syntax are given in § 4.5 in paragraphs with numbers corresponding to those in § 4.4.

##### 4.4.1 *Identifier*

An identifier is a string of one or more characters which begins with a letter and, if applicable, subsequently contains only digits and/or letters e.g. U, UPDATE, UPD8.

##### 4.4.2 *Symbolic name*

A symbolic name is a string of one or more characters containing at least one letter and/or at least one of the graphic characters + (plus sign), # (number sign), % (per cent sign) plus any number of digits including none. The characters may appear in any order, e.g., 24H, #6, +4687191818, X%.

##### 4.4.3 *Decimal numeral*

A decimal numeral is a character combination, consisting of a digit or digits and an optional . (full stop), preceded by the special character combination D' (D apostrophe). If the numeric default base for an information unit (see Recommendation Z.315) is decimal, then the D' is optional.

##### 4.4.4 *Nondecimal numerals*

A nondecimal numeral is a character combination preceded by a special character combination indicating the type of numeral.

4.4.4.1 H' (H apostrophe) is used to indicate a hexadecimal numeral, the following characters thus being any of the digits 0 to 9 or letters A, B, C, D, E, F.

4.4.4.2 O' (letter O apostrophe) is used to indicate an octal numeral, the following characters thus being any of: digits 0, 1, 2, 3, 4, 5, 6, 7.

4.4.4.3 B' (B apostrophe) is used to indicate a binary numeral, the following characters thus being digit(s) 0 and/or digit(s) 1.

4.4.4.4 K' (K apostrophe) is used to indicate a keyed numeral, the following characters thus being any of: digits, \* (asterisk) # (number sign), or letters A, B, C, D.

4.4.4.5 When the default base for an information unit (see Recommendation Z.315) is one of the nondecimal numerals e.g. hexadecimal, the corresponding character combination, i.e. H' in this example, is optional.

TABLE 2/Z.314  
Summary of use of characters

CCITT International Alphabet No. 5 (Recommendation V.3) [1]			Man-machine language use
Character or character string	Position number	Name	
CAN	1/8	cancel	Used as a deletion character.
!	2/1	exclamation mark	An indicator used in dialogue procedures (continuation character in input language).
"	2/2	quotation mark	A text string delimiter and a graphic character.
#	2/3	number sign	A character which may be used in symbolic names and keyed numerals and as a graphic character.
%	2/5	per cent sign	A character which may be used in symbolic names and as a graphic character.
&	2/6	ampersand	A separator for information grouping and a graphic character.
'	2/7	apostrophe	A separator used when indication of type of numeral is required. The character is placed between a letter indicating the type of numeral and the numeral itself. Also used as a graphic character.
(	2/8	left parenthesis	Used for delimiting arithmetical expressions and as a graphic character.
)	2/9	right parenthesis	Used for delimiting arithmetical expressions and as a graphic character.
*	2/10	asterisk	Used for keyed numerals, in arithmetical expressions and as a graphic character.
+	2/11	plus sign	A character which may be used in symbolic names, in arithmetical expressions and as a graphic character.
,	2/12	comma	A separator used to separate parameters (if more than one) within a block of parameters.
-	2/13	hyphen	A separator used to separate information units. Also used in arithmetical expressions and as a graphic character.
.	2/14	full stop	A separator used for subdividing a number into an integer part and a fraction part and as a graphic character.
/	2/15	solidus	Used in arithmetical expressions and as a graphic character.
:	3/10	colon	A separator used to separate blocks of parameters from each other and from the command code, an indicator used in the parameter block request indication and a separator used in output.
;	3/11	semicolon	An indicator used to terminate a command (execution character).
<	3/12	less than sign	An indicator used as a ready indicator for the system to output that it is ready to receive information.
=	3/13	equal sign	A separator used to separate the parameter name and the parameter value of a parameter.
>	3/14	greater than sign	A separator to terminate the destination identifier. Also a graphic character.
?	3/15	question mark	A indicator used for prompting.
&&	2/6, 2/6	ampersand ampersand	Separator used for information grouping.
&-	2/6, 2/13	ampersand hyphen	Separator used for information grouping.
&&-	2/6, 2/6, 2/13	ampersand ampersand hyphen	Separator used for information grouping.
/*	2/15, 2/10	solidus asterisk	Used to open a comment.
*/	2/10, 2/15	asterisk solidus	Used to close a comment.

#### 4.4.5 *Text string*

A text string is a string of zero or more characters enclosed by a " (quotation mark) at beginning and end. The string may contain any of the characters belonging to the character set defined in § 4.2 except correction characters (see Recommendation Z.315). If " (quotation mark) is required within a string, it is represented by "" (double quotation marks).

#### 4.4.6 *Arithmetical expression*

An arithmetical expression is a combination of certain basic elements and arithmetic operators enclosed within parentheses.

#### 4.4.7 *Ancillary facilities*

Additional facilities have been provided when using MML commands as follows.

##### 4.4.7.1 *Comment facility*

A comment is defined as a character string enclosed between the separators /\* (solidus asterisk) and \*/ (asterisk solidus), where the character string may contain any characters except the sequence \*/ (asterisk solidus) and correction characters (see Recommendation Z.315). The character string, including the delimiters, has neither MML syntactical nor semantical significance. However, if it occurs in a text string, it is regarded as being part of the text string. A comment may be inserted only before and/or after a separator, indicator, arithmetical delimiter [+ (plus sign), - (hyphen), ( (left parenthesis), ) (right parenthesis), / (solidus), \* (asterisk)], identifier or information unit [excluding the ' (apostrophe) between the type of numeral and the numeral itself and the . (full stop) between the integer and fractional part of a number].

##### 4.4.7.2 *Escape syntax*

In some systems it is not possible to use characters with syntactical meaning [e.g. ; (semi colon), - (hyphen)] or correction characters as data. In such systems an escape indication may be used in order to introduce the following character as data.

A specific escape indication is not proposed due to the diverse nature of terminals.

No syntax diagram is given.

##### 4.4.7.3 *Format effector*

A format effector (see § 4.3.4) is used to format input and output in a suitable manner. Format effectors have no significance in a command and may appear anywhere in input.

No syntax diagram is given.

#### 4.4.8 *Separator*

A separator is a character or a string of characters used to separate items of information in the input or output and it may, in addition, have structural, semantic or other significance.

No syntax diagram is given.

#### 4.4.9 *Indicator*

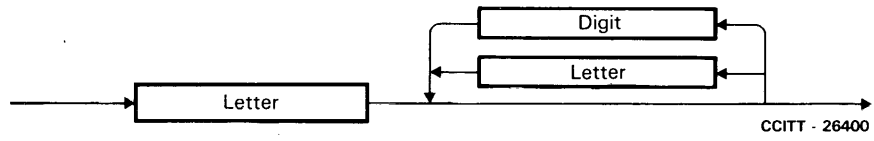
An indicator is a character used to indicate a state or make a request.

No syntax diagram is given.

#### 4.5 *Definition of the basic elements used in the syntax in diagrams*

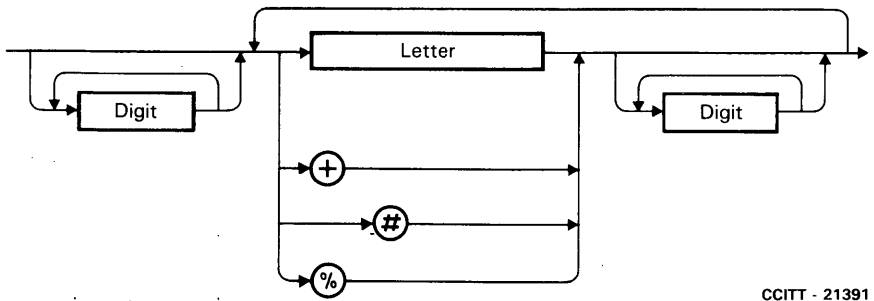
All these elements may be used in both input and output but for simplicity only the input elements are shown in the diagrams. The output elements are identical to the input elements.

4.5.1 Identifier



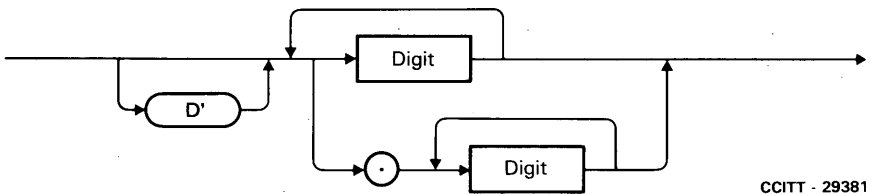
CCITT - 26400

4.5.2 Symbolic name



CCITT - 21391

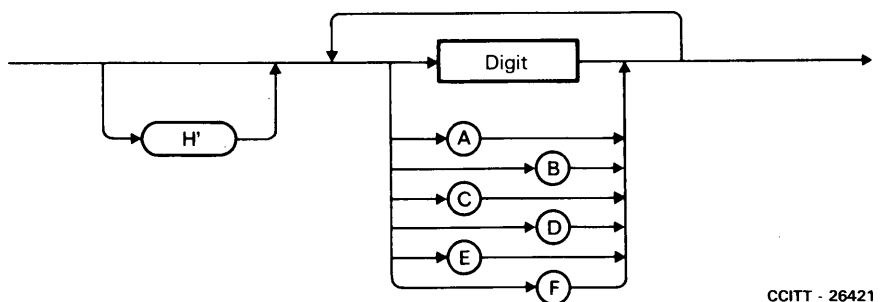
4.5.3 Decimal numeral



CCITT - 29381

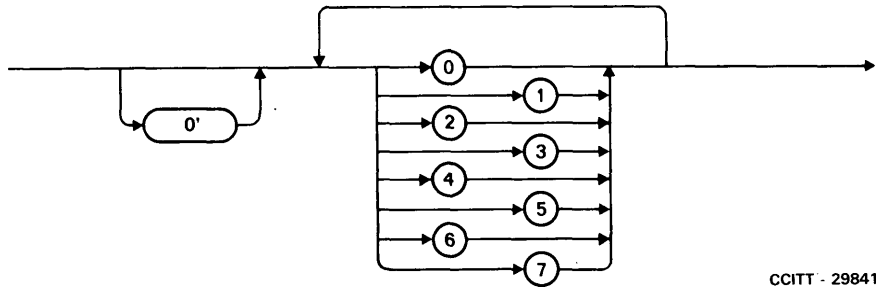
4.5.4 Nondecimal numerals

4.5.4.1 Hexadecimal numeral



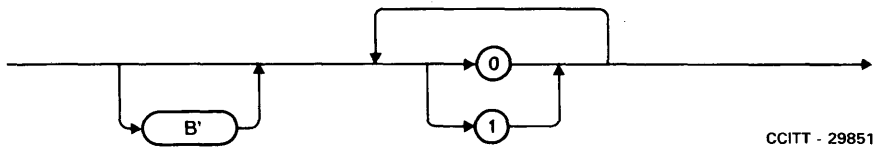
CCITT - 26421

4.5.4.2 Octal numeral



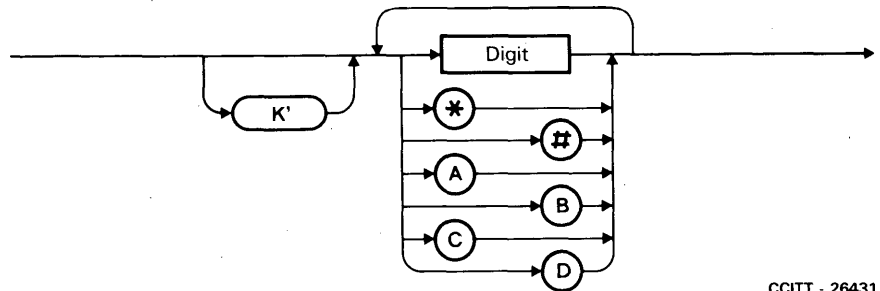
CCITT - 29841

4.5.4.3 Binary numeral



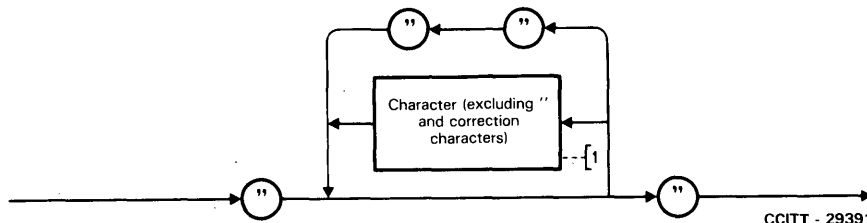
CCITT - 29851

4.5.4.4 Keyed numeral



CCITT - 26431

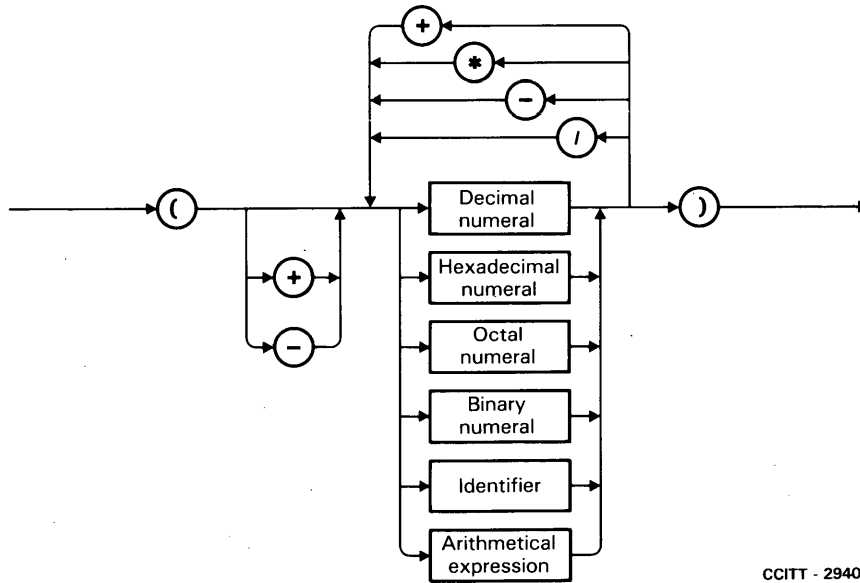
4.5.5 Text string



CCITT - 29391

1) Not further expanded in diagram form.

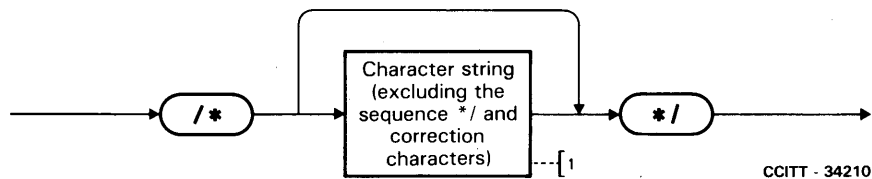
4.5.6 *Arithmetical expression*



CCITT - 29401

Note – The deepest level of the arithmetical expression has to satisfy the diagram in a form where the box “arithmetical expression” is omitted.

4.5.7 *Comment*



CCITT - 34210

1) Not further expanded in diagram form.

**References**

- [1] CCITT Recommendation *International Alphabet No. 5*, Vol. VIII, Fascicle VIII.1, Rec. V.3.

**Recommendation Z.315**

**5. INPUT (COMMAND) LANGUAGE SYNTAX SPECIFICATION**

5.1 *General*

The following text describes the elements of the input language. Syntax diagrams of the input language are given in § 5.4 in paragraphs with numbers corresponding to those in § 5.2. Where input elements are used in output, reference to these elements is made in the output language description Recommendation Z.316. Procedural aspects are taken into account in Recommendation Z.317. It should be noted that certain areas of the syntax allow options to be taken which could result in a syntax clash. The taking of such options must be chosen to suit the particular system involved.

## 5.2 *Command structure*

### 5.2.1 *Command*

A command begins with the command code, which defines the function to be performed by the system. If further information is required a command code can be followed by a parameter part from which it is separated by a : (colon). The parameter part consists of one or more blocks of parameters (see §§ 5.2.3 and 5.2.9.1). A command is always completed by an execution character (see Recommendation Z.317).

### 5.2.2 *Command code*

The command code is composed of up to three identifiers separated by a - (hyphen) (e.g. functional area - object type - action). Where command codes are in the form of single mnemonic abbreviations, it is recommended that they consist of the same number of characters.

### 5.2.3 *Block of parameters*

A block of parameters contains information necessary to execute the function specified in the command code. The information in a block of parameters is expressed in the form of a number of parameters specific to the command. If more than one parameter is included, they shall be separated by a , (comma). All parameters in any one block shall be of the same kind i.e. either position defined parameters or parameter name defined parameters.

### 5.2.4 *Parameters*

A parameter identifies and contains a piece of information and may be either position defined or parameter name defined. Non-relevant parameters may be omitted in accordance with §§ 5.2.4.1 and 5.2.4.2 below.

#### 5.2.4.1 *Position defined parameter*

A position defined parameter consists of a parameter value which may be preceded by a parameter name from which it is separated by an = (equal sign). Parameters must be given in a predetermined order within the parameter block. Where a parameter value is not to be given, the parameter is omitted leaving the appropriate separator or the appropriate indicator used to terminate a command. This indicates the parameter's position in the block of parameters. Parameter omission can imply that the default value is meant. The default value can also be indicated by giving a parameter value assigned for this purpose.

#### 5.2.4.2 *Parameter name defined parameter*

A parameter name defined parameter consists of a parameter name followed by a parameter value from which it is separated by an = (equal sign). These parameters may be given in an arbitrary order within the parameter block. Where a parameter value is not to be given, the parameter name and separator = (equal sign) and the separator , (comma) following the parameter are also omitted. This omission can imply that the default value is meant. The default value can also be indicated by giving a parameter value assigned for this purpose. Where a parameter value implies the parameter name the latter and the separator = (equal sign) can be omitted.

### 5.2.5 *Parameter name*

A parameter name unambiguously indicates the kind and structure of the subsequent parameter value and thereby defines the parameter value and how it shall be interpreted. It is an identifier.

### 5.2.6 *Parameter value*

A parameter value contains the information required to specify the appropriate object(s) or value(s) and consists of one or more information units. In the case where no information grouping (see § 5.2.9) is applied a parameter value reduces to a parameter argument.

### 5.2.7 *Parameter argument*

A parameter argument contains the information required to specify the appropriate object or value. It is the form of a parameter value when no information grouping is applied (see § 5.2.9). A parameter argument consists of a simple or a compound parameter argument.



### 5.2.7.1 *Simple parameter argument*

A simple parameter argument consists of one information unit.

### 5.2.7.2 *Compound parameter argument*

A compound parameter argument consists of two or more information units separated by a - (hyphen).

### 5.2.8 *Information unit*

An information unit constitutes the smallest unit of information in the language from a syntactical point of view. An information unit can be a numeral, an identifier, a symbolic name, a text string or an arithmetical expression. A numeral always has a default base (e.g. hexadecimal) which can be overwritten, if required, by introducing the desired base as specified in Recommendation Z.314. However the default base for a keyed numeral cannot be overwritten by another base.

### 5.2.9 *Information grouping*

Information grouping is used to improve the speed and ease of input activities. It is performed by grouping sets of information of the same type within the same command.

#### 5.2.9.1 *Grouping of blocks of parameters*

If several blocks of parameters are to be included in one command they shall be separated by a : (colon).

#### 5.2.9.2 *Grouping of simple parameter arguments*

If several simple parameter arguments are to be given within one parameter of one command it is possible to indicate the relevant simple parameter arguments within the same parameter value separated by an & (ampersand). Example: 5&9 means the simple parameter arguments 5 and 9.

In the case of a sequence of consecutive (increment by 1) simple parameter arguments, it is possible to indicate the arguments by writing the lower and upper simple parameter arguments separated by an && (ampersand ampersand)<sup>1)</sup>. Example: 5&&9 means the simple parameter arguments 5, 6, 7, 8 and 9.

A combination of the above possibilities may also be applied when required. Example: 5&&7&9 means the simple parameter arguments 5, 6, 7 and 9.

#### 5.2.9.3 *Grouping of compound parameter arguments*

If several compound parameter arguments are to be given within one parameter of one command it is possible to indicate the relevant compound parameter arguments within the same parameter value separated by an & (ampersand). Example: 5-1&6-3 means the two compound parameter arguments 5-1 and 6-3.

If a group of compound parameter arguments differ only in the last information unit the first compound parameter argument is completely specified whereas all subsequent compound parameter arguments are represented only by their last information units, separated by an &- (ampersand hyphen). Example: 7-1&-3 means the two compound parameter arguments 7-1 and 7-3.

If a group of compound parameter arguments differ only in the last information unit and constitutes a consecutive sequence (increment by 1) it is possible to indicate the arguments by writing the lower and upper information units separated by an &&- (ampersand ampersand hyphen)<sup>1)</sup>. Example 1: 7-1&&-3 means the three compound parameter arguments 7-1, 7-2 and 7-3. Example 2: 7-1&-3&&-5 means the four compound parameter arguments 7-1, 7-3, 7-4 and 7-5.

Any combination of the above possibilities may also be applied when required. Example: 5-1&&-3&8-2&-5&-6 means the six compound parameter arguments 5-1, 5-2, 5-3, 8-2, 8-5 and 8-6.

---

<sup>1)</sup> The interpretation of the separators && (ampersand ampersand) and &&- (ampersand ampersand hyphen) is under study. Other interpretations exist. One alternative would imply that no specific increment is inherent in the syntax. That is, the relationship of the values between the upper and lower values in the sequence is a semantic relationship dependent upon the function for which the sequence is being specified.

### 5.3 Corrections and delete command

Corrections can be made by the deletion and resubmission of input.

Specific characters are not proposed because of the diverse nature of Input/Output terminal devices available.

#### 5.3.1 Delete last character

The facility may be used to delete successive input characters back to the last system output (see § 5.3.2).

#### 5.3.2 Delete to last system output

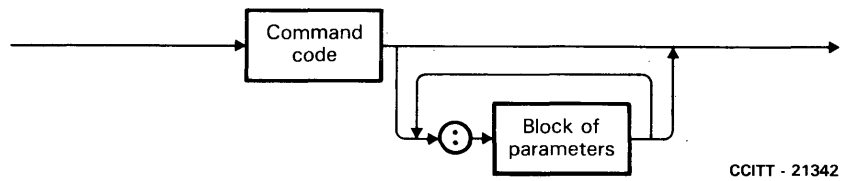
This facility deletes all input characters after the last system output, being either the ready indication or prompting output (see Recommendation Z.317).

#### 5.3.3 Delete command

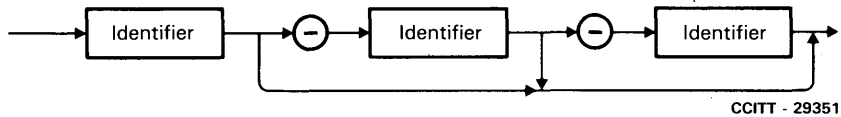
The delete command request is conveyed by the CAN character (cancel). The use of this character causes the system to respond with an acknowledgement that present input after the last command executed is cancelled. The system should respond with a new ready indication to indicate that it is waiting for a new command code (see Recommendation Z.317).

### 5.4 Definition of the input (command) language structure in syntax diagrams

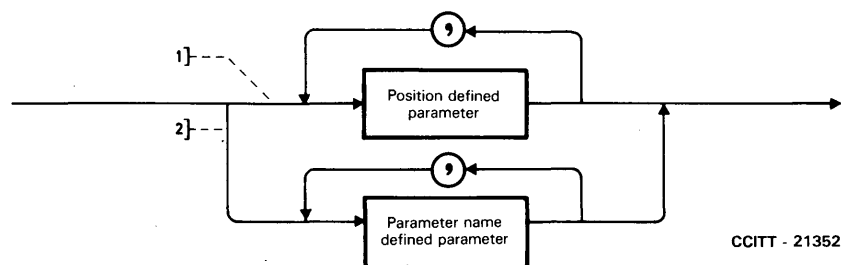
#### 5.4.1 Command



#### 5.4.2 Command code



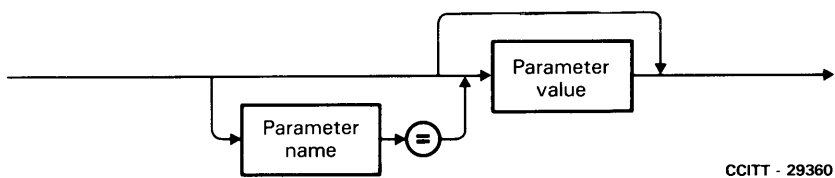
#### 5.4.3 Block of parameters



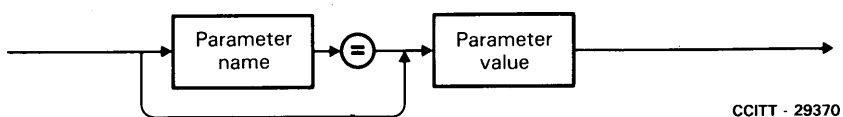
- 1) Upper main branch valid only for block of position defined parameters.
- 2) Lower main branch valid only for block of parameter name defined parameters.

5.4.4 Parameters

5.4.4.1 Position defined parameter



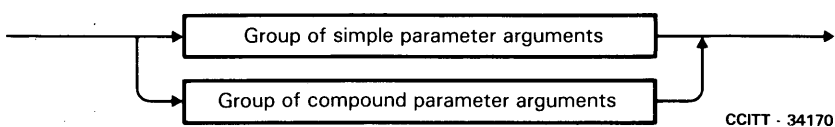
5.4.4.2 Parameter name defined parameter



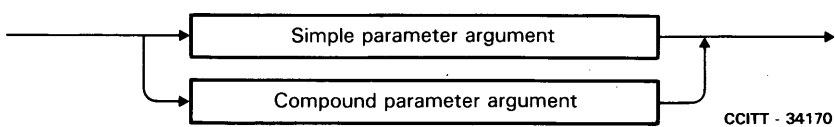
5.4.5 Parameter name



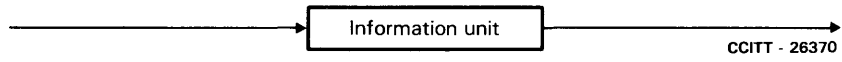
5.4.6 Parameter value



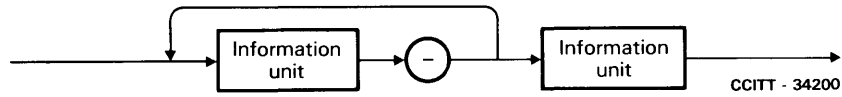
5.4.7 Parameter argument



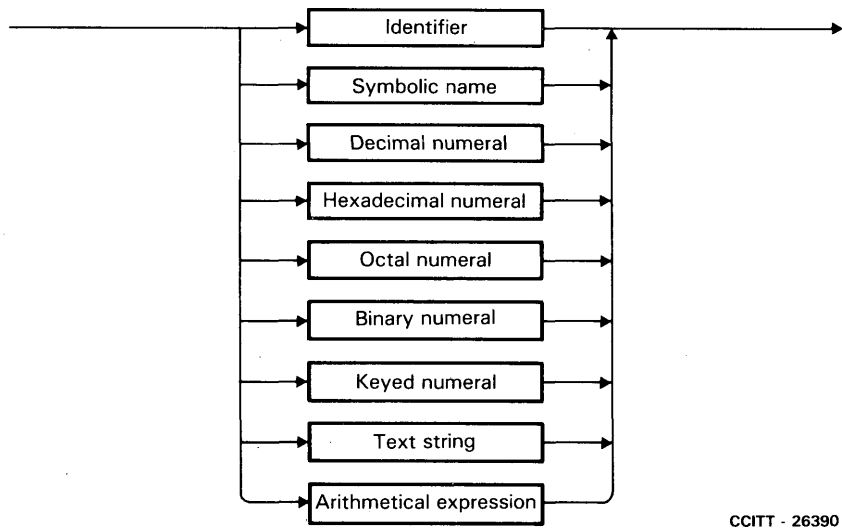
5.4.7.1 *Simple parameter argument*



5.4.7.2 *Compound parameter argument*



5.4.8 *Information unit*

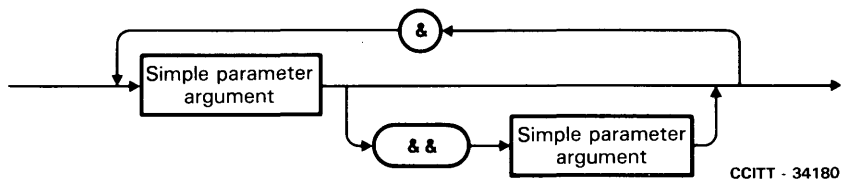


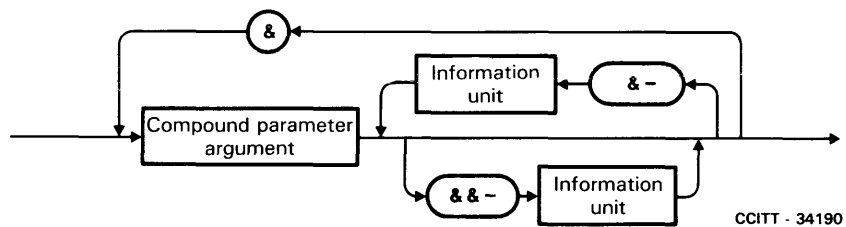
5.4.9 *Information grouping*

5.4.9.1 *Group of blocks of parameters*

See syntax diagram § 5.4.1.

5.4.9.2 *Group of simple parameter arguments*





**Recommendation Z.316**

**6. OUTPUT LANGUAGE SYNTAX SPECIFICATION**

6.1 *General*

Syntax diagrams of the output language are given in § 6.3 in paragraphs having numbers corresponding to those in § 6.2. Where input elements are used in output, a reference is made to the input language description Recommendation Z.315. Procedural aspects utilizing output other than output outside dialogue are taken into account in Recommendation Z.317.

6.2 *Output structure*

6.2.1 *Output outside dialogue*

The output described is output outside dialogue. This output is either a spontaneous output indicating a certain event, e.g. an alarm situation, or it is a delayed response to an interactive mode operating sequence (see Recommendation Z.317). An example of such a delayed response is a traffic measurement result.

6.2.2 *Header*

The header is given in output outside dialogue. It is also used in the dialogue procedure (see Recommendation Z.317). The main purpose of the header is to mark the output or the dialogue for identification and information. The header can also be used for special purposes for an operation and maintenance centre. Recommended contents are information related to source identification, date and time. More information not related to the input or output function can be added to the header as additional header information.

The header is introduced by format effectors and/or graphic characters selected from a layout option.

6.2.2.1 *Layout option*

A layout option is a combination of format effectors and graphic characters used to bound elements of the output in a clear and readable form.

6.2.2.1.1 *Graphic characters*

Graphic characters are used to improve readability of output.

6.2.2.1.2 *Format effector*

A format effector is used to format output in a suitable manner. Certain format effectors are specifically incorporated in the output definition but where the format effector element is shown any of the format effectors specified for MML can be used. No syntax diagram is shown.

6.2.2.2 *Source identifier*

A source identifier indicates the physical area in which an output was generated.

### 6.2.2.3 Calendar date

The output of the date in the header is based on the International Standard (ISO 2014) [1] for the writing of calendar dates in all-numeric form. The calendar date shall be written in the following order: year, month, day. The calendar date shall consist of a two decimal digit or four decimal digit year, a two decimal digit month, and a two decimal digit day of the month. The allowable characters between year and month and between month and day are hyphen or space.

Examples:

The 4th October 1979 shall be written in one of the following ways:

- a) 19791004
- b) 1979-10-04
- c) 1979 10 04
- d) 791004
- e) 79-10-04
- f) 79 10 04

The calendar date in input should preferably have a layout similar to that in output.

### 6.2.2.4 Time of day

The output of the time in the header is based on the International Standard (ISO 3307) [2]. However, in MML the output of a decimal fraction of hours, minutes, or seconds is not utilized in the header.

Time representations are based upon the 24-hour timekeeping system. The sequencing of time elements shall be from high order to low order (left to right): hours, minutes, seconds. The hour shall be represented by a two-digit decimal number ranging from 00 up to and including 23. The minute shall be represented by a two-digit decimal number ranging from 00 up to and including 59. The second shall be represented by a two digit decimal number ranging from 00 up to and including 59.

Examples:

Hours, minutes	1225	or	12:25
Hours, minutes, seconds	122501	or	12:25:01

### 6.2.2.5 Additional header information

Additional header information is general information which has no relation to the function of the output, e.g.:

- sequence number,
- processor number,
- output device,
- day of the week.

### 6.2.3 Alarm statement

The alarm statement may give information of a general class such as the degree of alarm or the source of alarm.

#### 6.2.3.1 Variable text

Variable text is a set of information units which contains information unique to the event which caused the output.

#### 6.2.4 Additional information

Additional information is general information related to the output, e.g.:

- type of output e.g. maintenance, statistics. This is not the same as identification of output, see § 6.2.6,
- output recipient identification.

#### 6.2.5 Command reference

A command reference supplies a command sequence number when needed in output outside dialogue as a reference to a previous input. In addition to the command sequence number it may also include clarifying text. It also may appear in dialogue procedures (see Recommendation Z.317).

### 6.2.5.1 Clarifying text

Clarifying text is a set of information units used to make the purpose and contents of the output more clear to the reader. Several clarifying texts could appear in an output.

### 6.2.6 Identification of output

Identification of output provides a unique identity for an output in a system's repertoire of outputs. Therefore it could be used as a reference to the explanation of the output in a manual.

### 6.2.7 Text block

A text block is any combination of clarifying texts, variable texts, parameter name defined parameters and/or tables which gives information wherever it is needed or requested.

### 6.2.8 Table

A table is an ordered presentation of interrelated information.

Clarifying text within a table can be used as labels to each column contained within the table. Where a table name or additional information associated with the table is required the clarifying text appearing at the beginning of the table in the syntax diagram of § 6.3.8 could be used.

When parameter name defined parameters are used to label columns each parameter should be complete, i.e. contain a parameter value (see Recommendation Z.315).

#### 6.2.8.1 New line

New line is a character combination necessary to reset an output device to the beginning of a new line. It is recognized that the character combination is device dependent but can contain the characters CR (carriage return) and LF (line feed). No syntax diagram is shown.

### 6.2.9 End of output

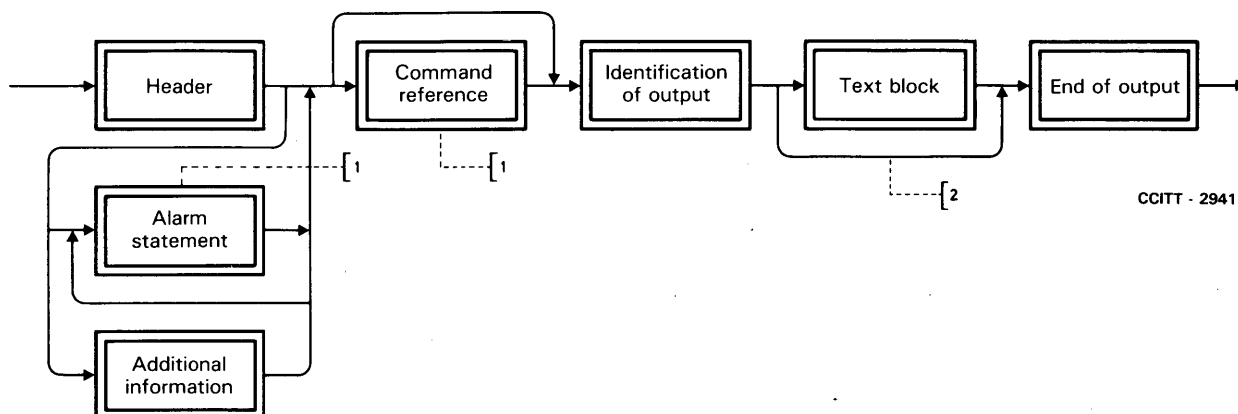
An end of output is an indication that an output is finished.

### 6.2.10 Comments in output

The purpose of a comment in output is as for clarifying text (see § 6.2.5.1 above) with the exception that the syntax is as for comment in input so that it may be discarded during a subsequent re-input. No syntax diagram is shown.

## 6.3 Definition of the output language syntax in diagrams

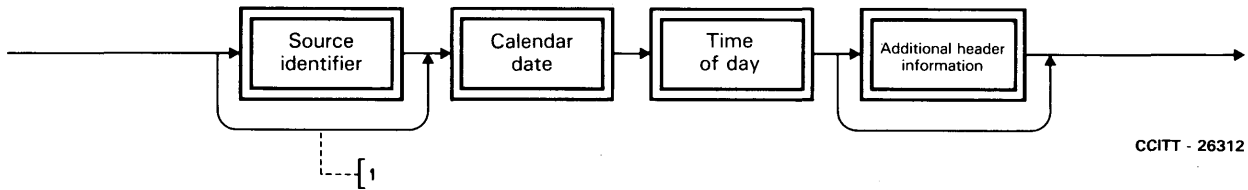
### 6.3.1 Output outside dialogue



CCITT - 29411

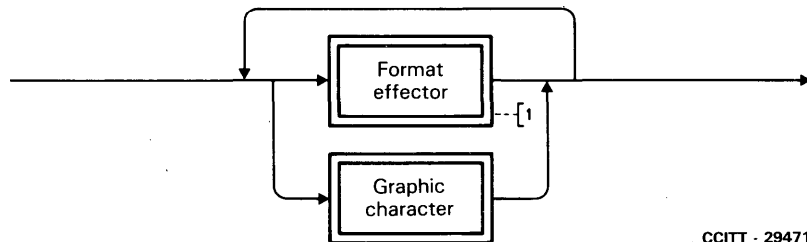
- 1) Command reference and alarm statement could appear in the same output, e.g. if a control system unit is taken out of service by means of a command.
- 2) This by-pass can be taken only when the identification of output contains sufficient information.

6.3.2 Header



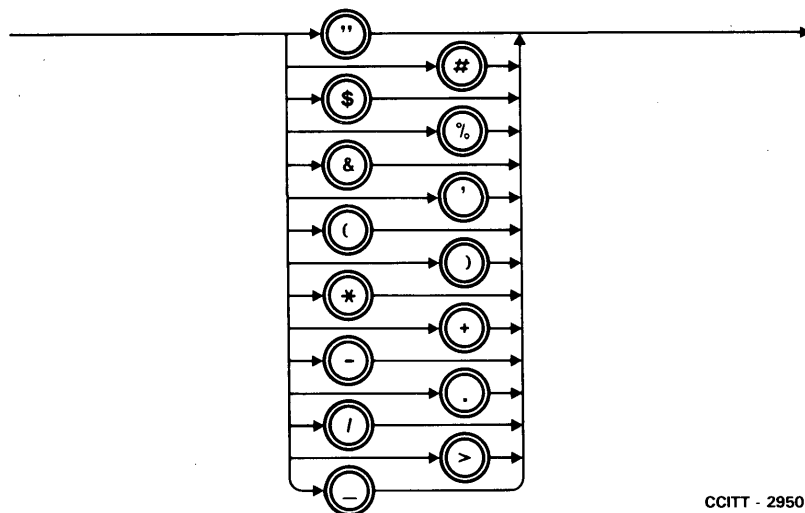
1) Source identifier may be omitted where there is only one source producing outputs.

6.3.2.1 Layout option



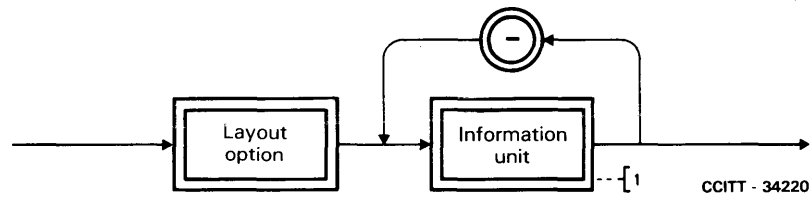
1) Not further expanded in diagram form.

6.3.2.1.1 Graphic character



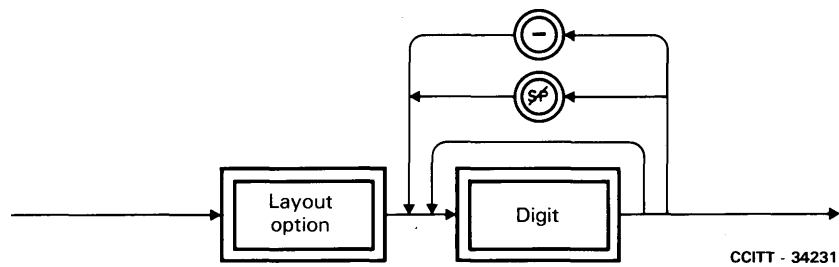


### 6.3.2.2 Source identifier

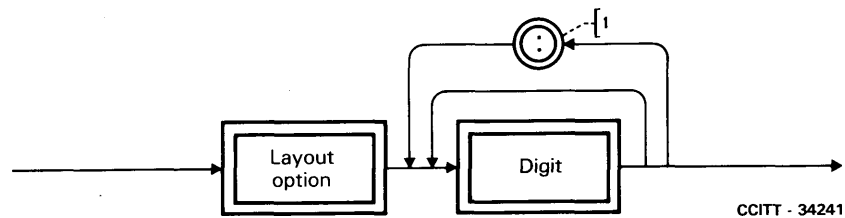


1) See Recommendation Z.315.

### 6.3.2.3 Calendar date

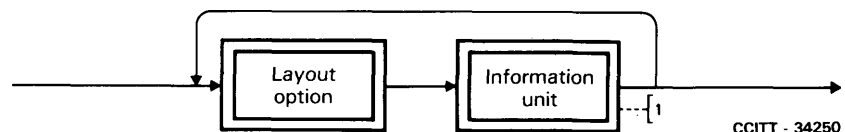


### 6.3.2.4 Time of day



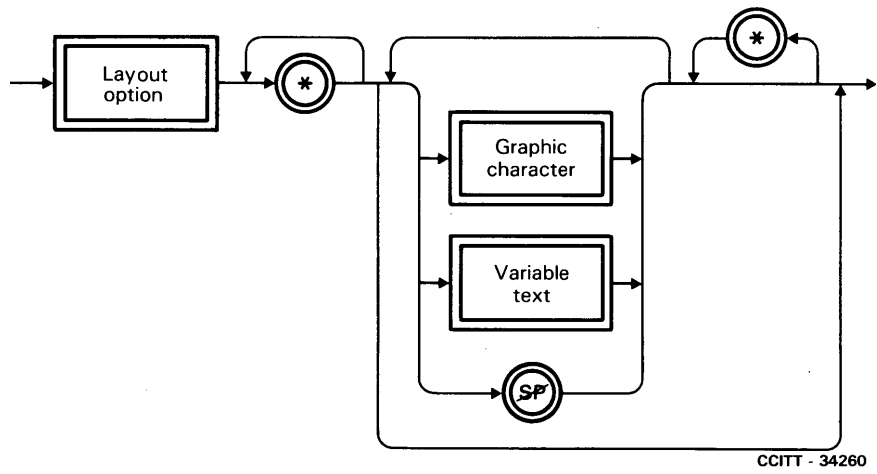
- 1) a) If required to facilitate visual human understanding of output, a : (colon) may be used to separate hours, minutes and seconds (refer to [2]).  
 b) This use of the : (colon) is not allowed in input since the character is used as a separator between blocks of parameters.

### 6.3.2.5 Additional header information

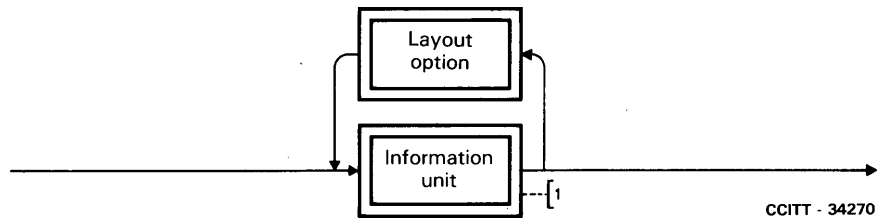


1) See Recommendation Z.315.

6.3.3 Alarm statement

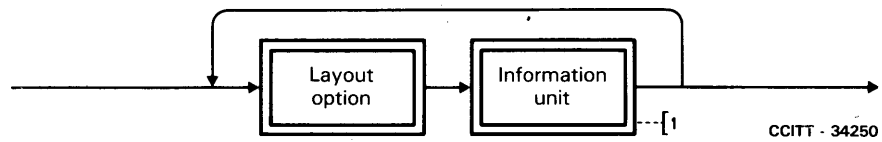


6.3.3.1 Variable text



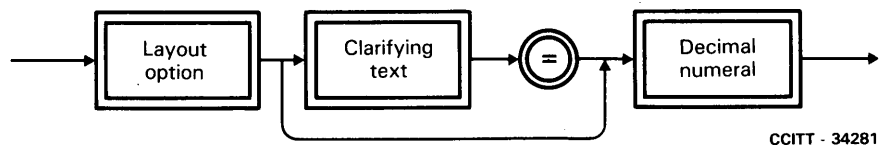
1) See Recommendation Z.315.

6.3.4 Additional information

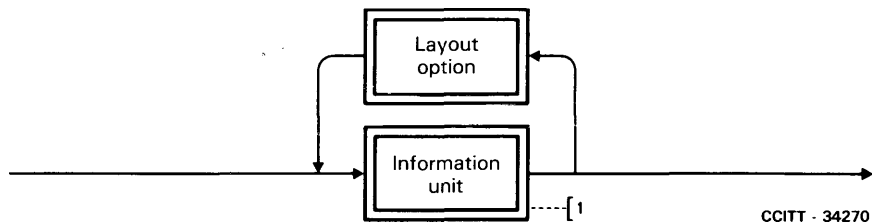


1) See Recommendation Z.315.

6.3.5 Command reference

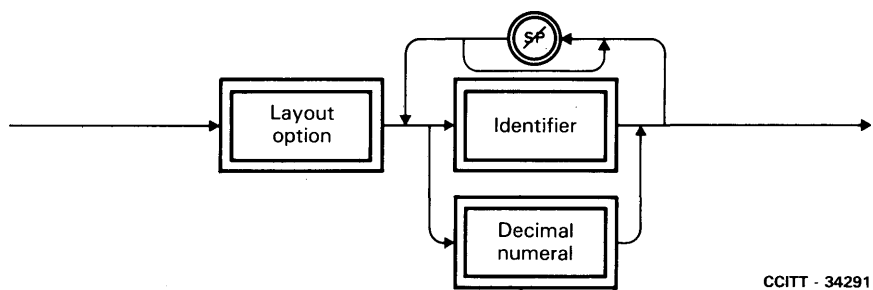


6.3.5.1 Clarifying text

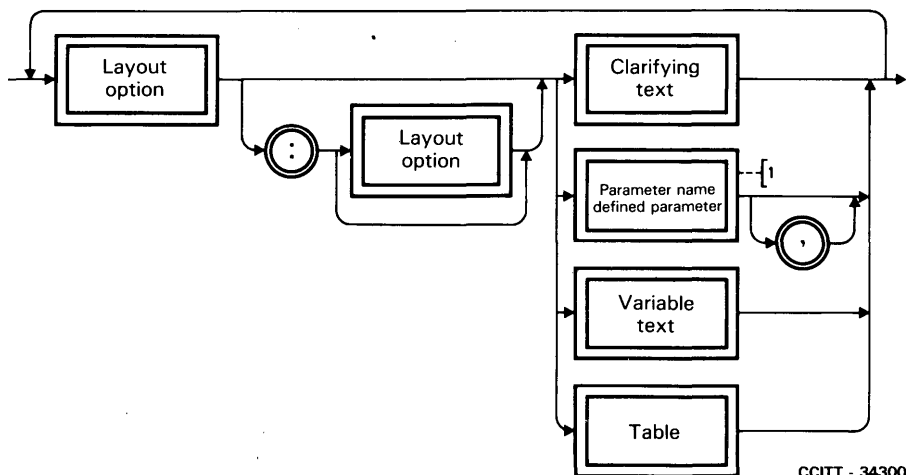


1) See Recommendation Z.315.

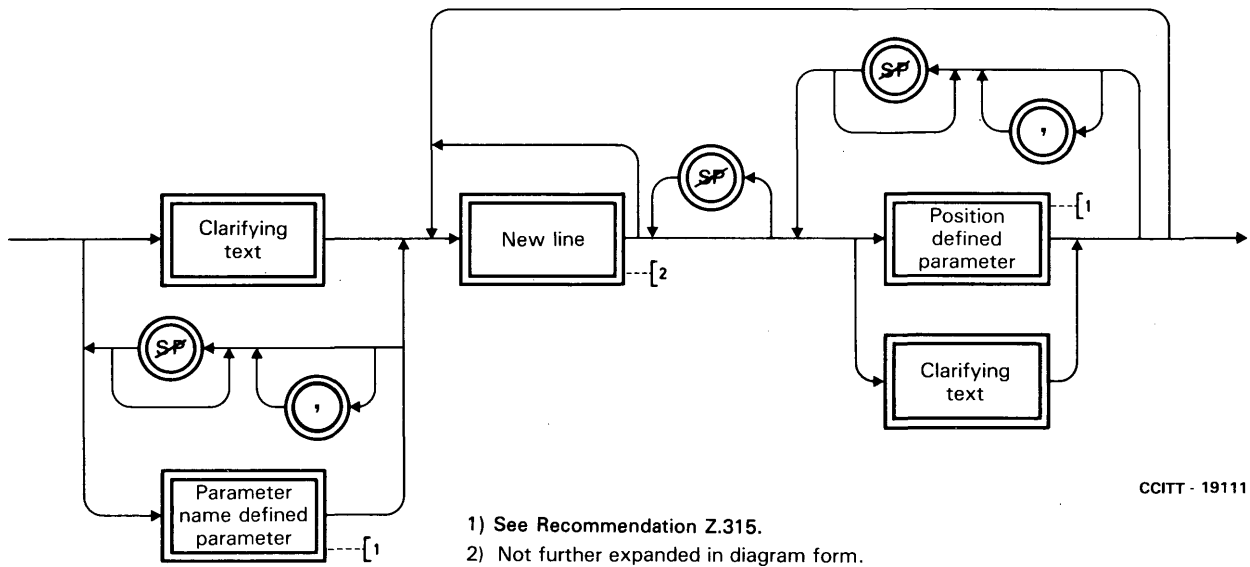
6.3.6 Identification of output



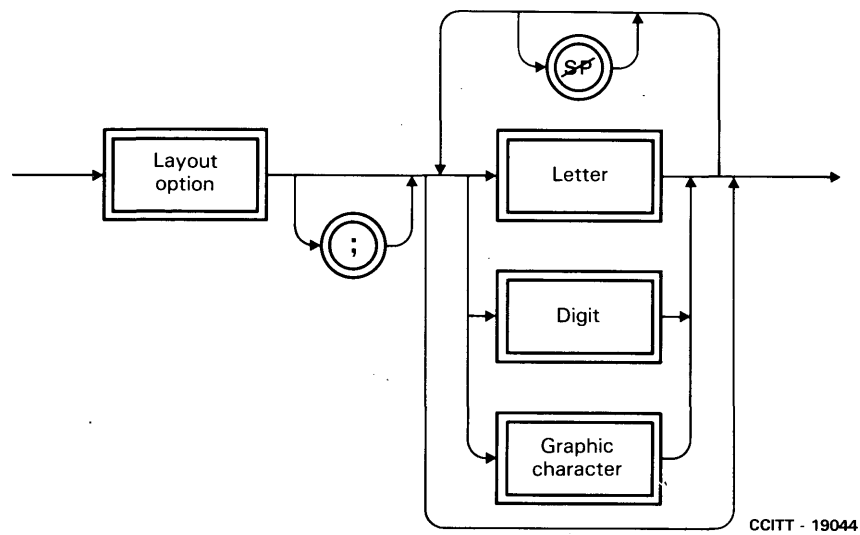
6.3.7 Text block



1) See Recommendation Z.315.



6.3.9 End of output



References

- [1] *Writing of Calendar Dates in All-Numeric Form*, ISO Standard 2014-1976.
- [2] *Information Interchange – Representation of Time of the Day*, ISO Standard 3307-1975.

## 7. MAN-MACHINE DIALOGUE

### 7.1 *General*

In the man-machine information interchange, one can define two types of sentences, namely dialogue procedure and output outside dialogue. Output outside dialogue is fully defined in Recommendation Z.316. Its position in the interchange in relation to dialogue procedure is given in the sentence procedure syntax diagram § 7.3. (See also § 7.5 for details.)

The text in § 7.2 describes the dialogue procedure. Syntax diagrams of the dialogue procedure are given in § 7.4 in paragraphs having numbers corresponding to those in § 7.2

A systematic analysis of possible errors made by initiators is not considered. Diagrams mainly refer to correctly given commands and only obvious error situations are considered. It is recognized that the proposed diagrams are not exhaustive and some of them might be modified when error recovery procedures have been completely considered.

### 7.2 *Definition of dialogue procedure*

#### 7.2.1 *Overview of the dialogue procedure*

The general forms of the dialogue procedure are the interactive mode operating sequence and the menu mode operating sequence. Other possible mode operating sequences form subsets of the interactive mode operating sequence, e.g. continuation mode operating sequence. The menu mode operating sequence is also capable of extension beyond the application to command code selection, e.g. selection of a parameter value out of a predefined set, selection of possible correction facilities and selection of possible execution characters.

A dialogue is opened by a procedure prologue. The procedure prologue contains the various preparations which must be performed before commands can be initiated. It may include a header from the system. Following the procedure prologue a destination prologue can precede one or more operating sequences. The dialogue can be terminated by a procedure epilogue.

#### 7.2.2 *Procedure prologue*

The procedure prologue may consist of three parts given in the following order:

- the request, which is an action to activate the man-machine terminal and the system;
- the identification of the initiator of the dialogue. The identification of the initiator is optional;
- a header, which is given from the system and contains the exchange identification, information relating to date and time, etc. The header can constitute the top line on a display or a paper form. Headers can be optional for a system or within a system for certain terminals.

The procedure prologue is intended to be executed only once at the beginning of a dialogue. The procedure prologue is followed by a ready indication inviting a destination prologue or an operating sequence.

The request, the identification of the initiator of the dialogue and the header are defined in the following paragraphs.

##### 7.2.2.1 *Request*

The request is a manual action to activate the man-machine terminal and the system or to cause an interrupt. The composition of the request is highly dependent on the type of man-machine terminal and implementation.

The request can consist of keying the break key or actuating a control switch, power on, etc and/or keying a sequence of characters on the keyboard.

##### 7.2.2.2 *Identification procedure*

The identification procedure is used for identification and authorization of an initiator of MML input. The identification can provide access to groups of commands which can have different security or functional classifications (e.g. traffic measurement function). The identification invitation may request the initiator to identify himself by means of a password or an identity card. The password must be input following a ready indication.

#### 7.2.2.2.1 *Ready indication*

The ready indication indicates that the direction of the dialogue has changed and that the system is waiting for information to be given at the man-machine terminal. The ready indication is defined as the character < (less than sign) optionally preceded by the appropriate format effectors.

#### 7.2.2.3 *Header*

The header (see Recommendation Z.316) is output by the system at the end of the procedure prologue.

#### 7.2.3 *Destination prologue*

The destination prologue consists of a destination identifier terminated by the separator > (greater than sign) so as to distinguish it from a command.

The destination identifier indicates the physical area where the command is to be mainly processed, e.g. exchange identification, processor number. It consists of one or more information units separated by - (hyphen). The destination could also be defined by a parameter in the command.

The destination identifier may be followed by a header to indicate that a selected destination is allowed, available and ready or alternatively by a rejection output to indicate the converse.

#### 7.2.4 *Procedure epilogue*

The procedure epilogue is used to terminate the dialogue procedure. The composition of the procedure epilogue is highly dependent on the type of man-machine terminal and implementation. The procedure epilogue can consist of actuating a control switch, power off, etc. and/or keying a sequence of characters on the keyboard and/or the output of end of dialogue from the system.

#### 7.2.5 *Operating sequences*

##### 7.2.5.1 *Interactive mode operating sequence*

The interactive mode operating sequence may consist of a single continuation mode operating sequence terminated by an optional end statement or of a series of continuation mode operating sequences or special actions. The latter occurs when, as a result of partial execution of a function, the system requests the supply of further information in the form of specific actions or further commands for which human judgement and/or decision is required.

The system generates an interaction request output in order to obtain these further actions.

Special actions can include manual responses, such as the actuation of keys on terminals or switchframes and the replacement of hardware packages, if recognized by the system.

##### 7.2.5.1.1 *Continuation mode operating sequence*

A continuation mode operating sequence contains a single command code, together with an alternating sequence of one or more parameter blocks and an appropriate number of executions.

Parameter input can be performed in two different ways: direct parameter input or format parameter input.

##### 7.2.5.2 *Menu mode operating sequence*

Menu mode operating sequence is similar to interactive mode operating sequence but provides a menu selection procedure both initially to arrive at the first command code and, if necessary, to provide a destination prologue, and also subsequently whenever a further command code is required.

The menu mode operating sequence may be entered following the procedure prologue in one of two ways. In one instance, menu mode will be entered automatically due to the fact that the menu mode has been requested on a general basis through a separate command, or because menu mode is the normal way of working in the system. The other mechanism is for a ? (question mark) to be given following the ready indication.

In all cases, having indicated the command code required through the menu selection procedure, parameters may be input using either direct parameter input or format parameter input. The menu mode operating sequence may optionally be terminated by an end statement.

#### 7.2.5.2.1 *Menu selection procedure*

Menu selection procedure can be used to arrive at a command code or to provide a destination prologue. The menu selection procedure also provides a means for assembling and executing a command with all its relevant parameters. This is done by selecting the appropriate item from subsequent menu outputs.

#### 7.2.5.2.2 *Command code output*

The command code output is the command code positioned as necessary in the output.

#### 7.2.5.2.3 *Menu output*

The menu output comprises a number of items one of which can be selected. The menu output may include appropriate instructional text.

#### 7.2.5.2.4 *Selection identity*

Each item of the menu output is preceded by a selection identity in order to allow a choice to be made.

#### 7.2.5.2.5 *Selection input*

The selection input is the character(s) of the selection identity of the chosen item from the menu output.

*Note* – Other means for indicating the selection available with advanced terminals are under study.

### 7.2.6 *Methods of inputting parameters*

There are two methods of inputting parameters: direct parameter input and format parameter input. Both methods can be used in the interactive mode operating sequence and in the menu mode operating sequence.

#### 7.2.6.1 *Direct parameter input*

The direct parameter input consists of an optional parameter block introduction sequence preceded by the separator : (colon). If only one sequence of parameter blocks is to be introduced the execution character ; (semi-colon) is given to initiate the required functions which will result in response output.

By giving the continuation character ! (exclamation mark) instead of the execution character the interpretation and execution of input information is done. The system is then required to return a parameter block request indication after the response output associated with the present block or sequence of blocks of parameters. The parameter block request indication, in turn, functions as an indication to proceed with the input of the next block or blocks of parameters. The parameter block request indication consists of a : (colon) optionally preceded by the appropriate format effectors. Following the parameter block request indication, the continuation mode operating sequence can be abandoned by invoking the delete command function.

The parameters are given in accordance with the parameter block introduction sequence. An optional end statement concludes a direct parameter input.

##### 7.2.6.1.1 *Parameter block introduction sequence*

The parameter block introduction sequence is used to input a block of parameters. All parameters are introduced according to the input syntax. The introduction of the parameters may be done directly without help from the system as described in Recommendation Z.315, or assistance from the system may be requested by calling the prompting facility. Prompting helps in providing a correct input by the system giving guidance on the next input requirement. Each information element down to parameter value (each part of the command bounded by separators) can be replaced by a "help" by means of a ? (question mark) in which case the system will prompt enough information to enable input to continue. The information is given in the form of a guidance output.

The contents of the guidance output depends on the place where the request for help was initiated and may cover information about:

- a) complete command (block of parameters);
- b) a part of the block of parameters, i.e. either the parameter name of the next parameter to be entered or information about a parameter value of the present parameter;

c) an indication that the information supplied is sufficient and that an execution order may be given.

The default value of a prompted parameter is given as a parameter value assigned for this purpose.

#### 7.2.6.2 *Format parameter input*

The format parameter input may be invoked in one of two ways. Either it is invoked directly or by giving the ? (question mark) following the command code. The system responds with an output listing the parameters for the command, i.e. the format output.

The format output is presented so that when it is filled in, and is input to the system, it completes the command. This process is the format parameter entry sequence. The format output, when subjected to the format parameter entry sequence, produces an input which conforms to Recommendation Z.315. This implies that where optional fields exist the system should output a default indication in the appropriate position – possibly including the default value. The precise construction of the format output and the method of preparing and offering the input, i.e. the format parameter entry sequence, depend on the system and terminal design and are not defined in this Recommendation. This Recommendation has assumed that all information required for a block of parameters can be provided in one format output. Where this is not the case the Recommendation has not yet been specified. The end of a format parameter input can be indicated to the system by giving one of the execution characters ; (semi-colon) or ! (exclamation mark) as appropriate.

#### 7.2.7 *Response output*

Response output covers all types of output conveying information about the state of an input.

Types of response output are acceptance output, rejection output and request output.

##### 7.2.7.1 *Acceptance output*

Acceptance output is an indication that an input to the system is syntactically correct and complete and that the appropriate system actions will be initiated, or have already been carried out.

In the latter case this indication may take the form of the result of the actual action.

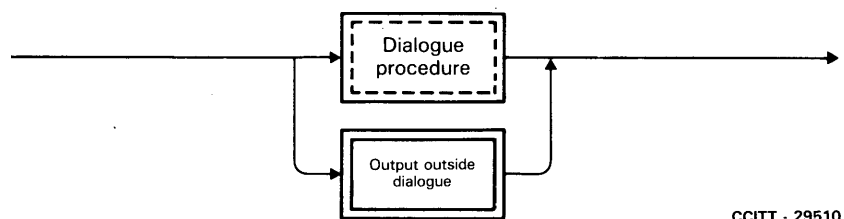
##### 7.2.7.2 *Rejection output*

Rejection output is an indication by the system that the input received is not valid and will not be acted upon, nor can correction be applied, e.g. when the system determines that the initiator of the input is not authorized to request the action required by the command.

##### 7.2.7.3 *Request output*

Request output is an output message which requests further input action, e.g. to correct an erroneous parameter.

#### 7.3 *Sentence procedure syntax diagram*



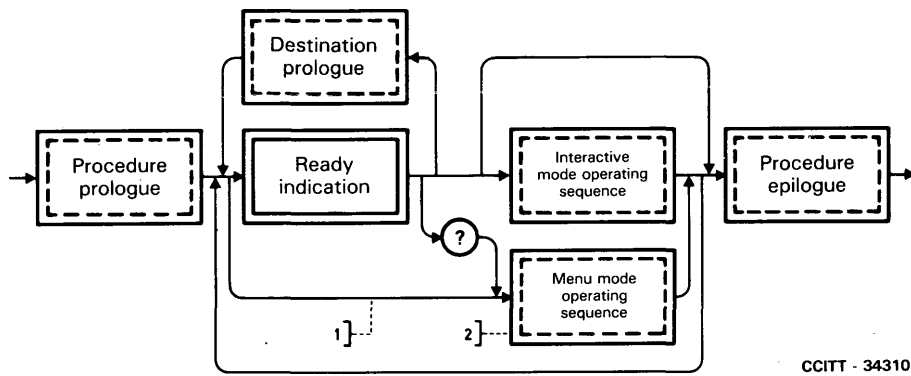
CCITT - 29510

#### 7.4 *Definition of the dialogue procedure syntax in diagrams*

Recommendations Z.315 and Z.316 describe the input and output syntactic elements used, but not defined, in this Recommendation.



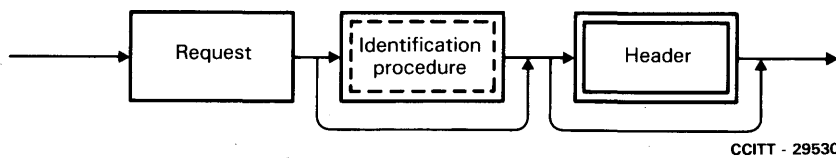
7.4.1 Dialogue procedure



CCITT - 34310

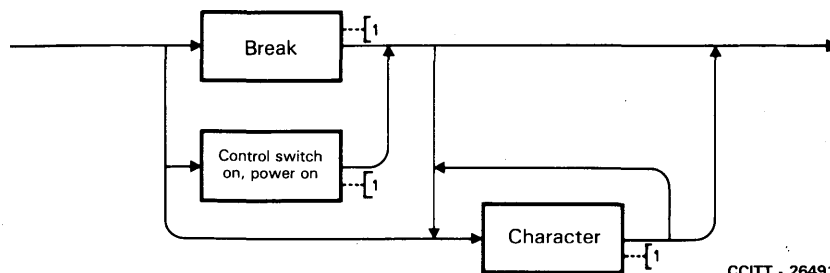
- 1) Menu mode may be the default operating system for a particular terminal, or a separate command may direct the system to operate in such a way. Therefore it is necessary to provide a direct path from procedure prologue to menu mode operating sequence. If neither of these methods is the case then the ready indication is given in the usual way, and if menu mode is required, a ? (question mark) must be input.
- 2) This operating sequence offers also the destination prologue facility in menu mode.

7.4.2 Procedure prologue



CCITT - 29530

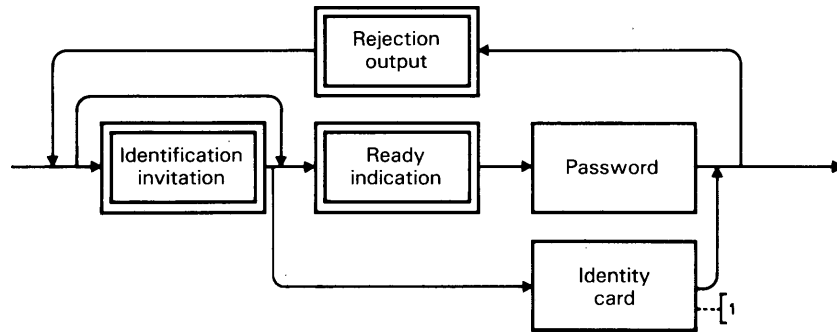
7.4.2.1 Request



CCITT - 26491

- 1) Not further expanded in diagram form.

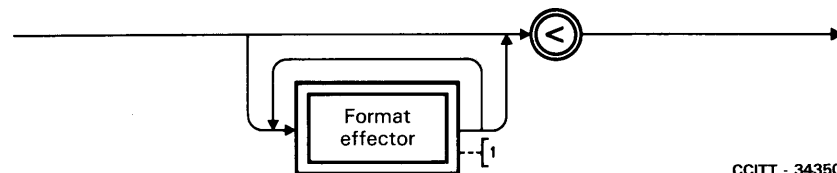
7.4.2.2 Identification procedure



1) Not further expanded in diagram form.

CCITT - 34321

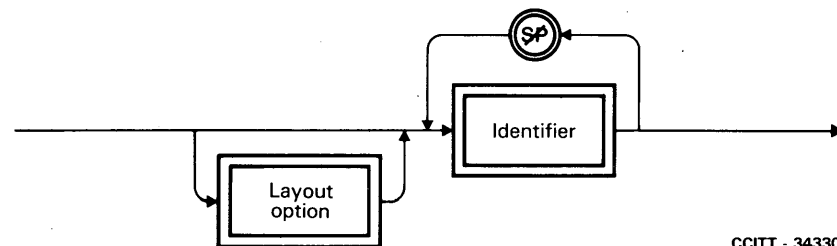
7.4.2.2.1 Ready indication



1) Not further expanded in diagram form.

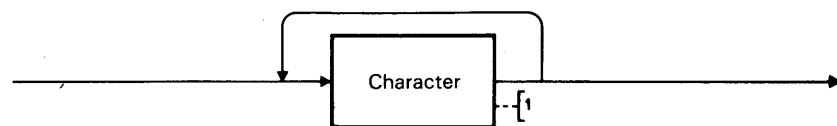
CCITT - 34350

7.4.2.2.2 Identification invitation



CCITT - 34330

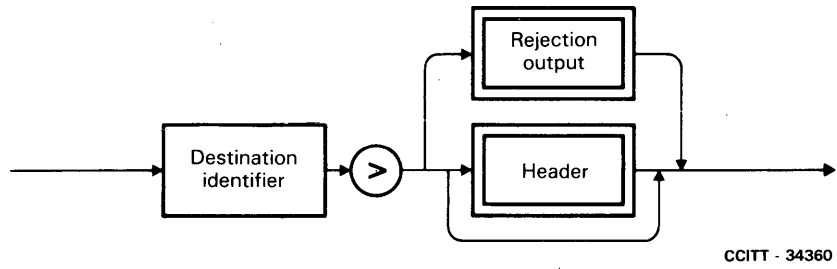
7.4.2.2.3 Password



CCITT - 34340

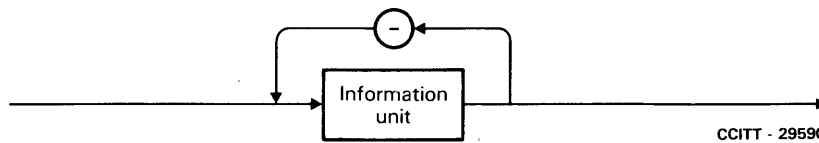
1) Not further expanded in diagram form.

7.4.3 *Destination prologue*



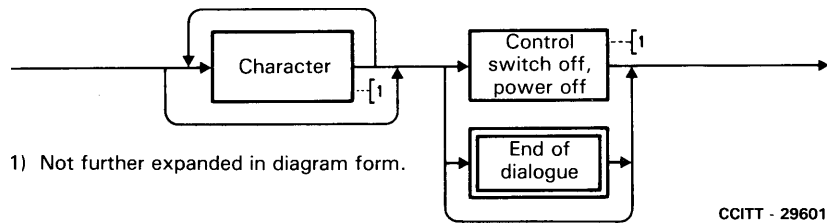
CCITT - 34360

7.4.3.1 *Destination identifier*



CCITT - 29590

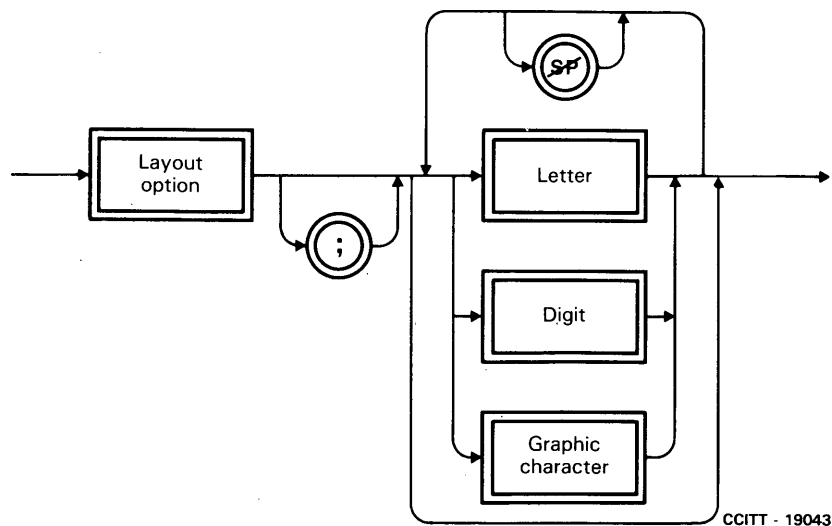
7.4.4 *Procedure epilogue*



1) Not further expanded in diagram form.

CCITT - 29601

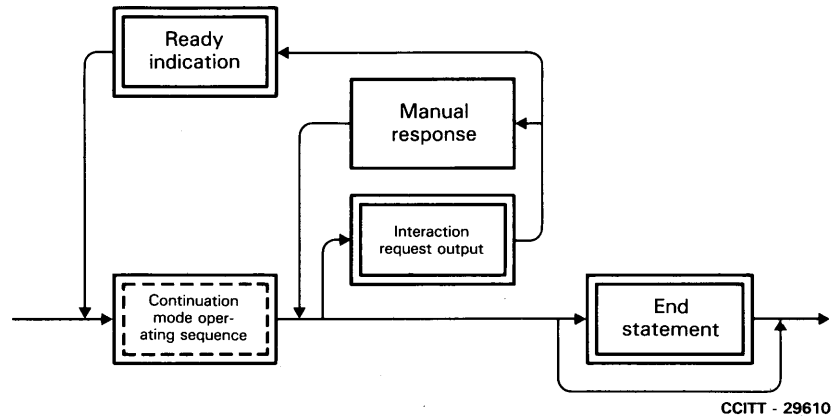
7.4.4.1 *End of dialogue*



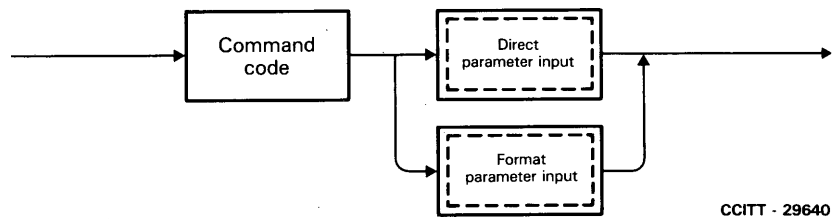
CCITT - 19043

7.4.5 Operating sequences

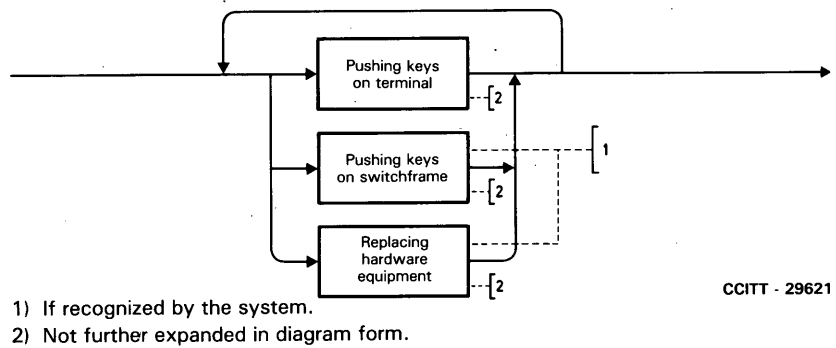
7.4.5.1 Interactive mode operating sequence



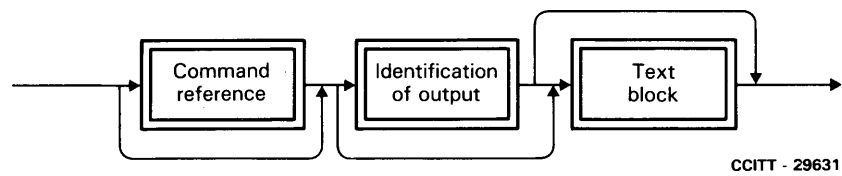
7.4.5.1.1 Continuation mode operating sequence



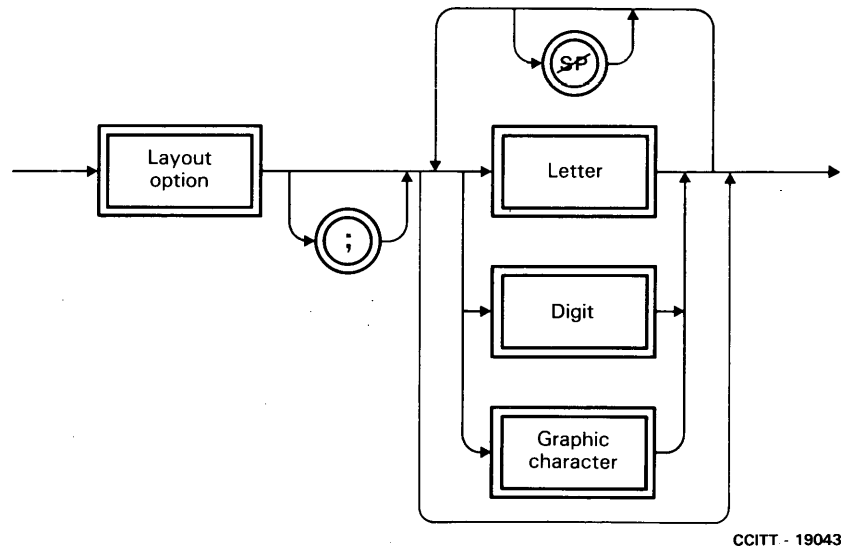
7.4.5.1.2 Manual response



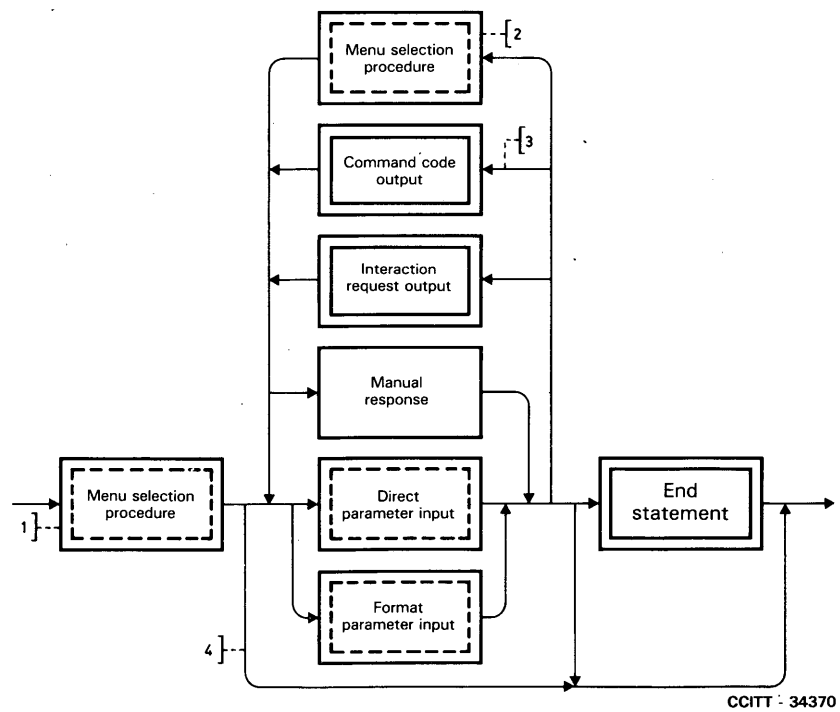
7.4.5.1.3 Interaction request output



7.4.5.1.4 End statement

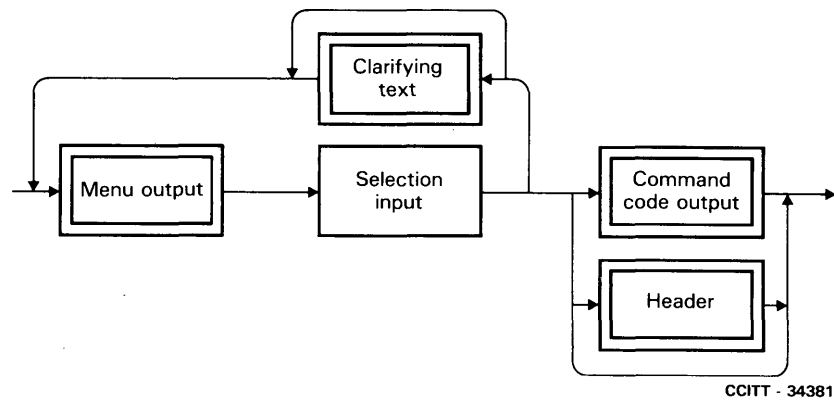


7.4.5.2 Menu-mode operating sequence

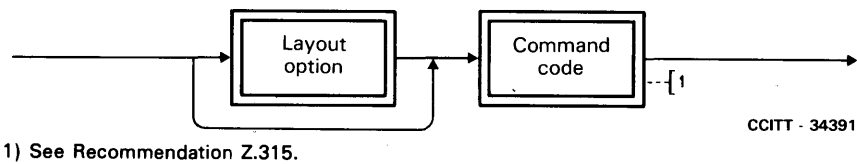


- 1) Function/command and area/destination selection.
- 2) No destination selection in this branch.
- 3) String of related commands.
- 4) Destination selection only.

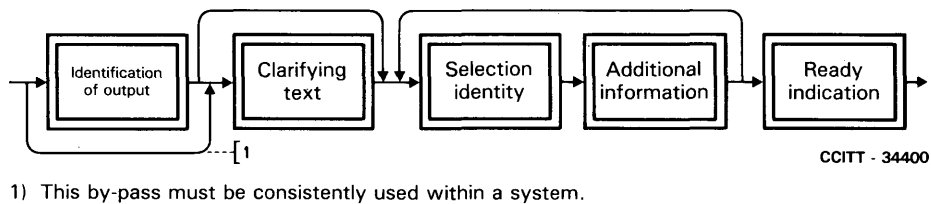
7.4.5.2.1 *Menu selection procedure*



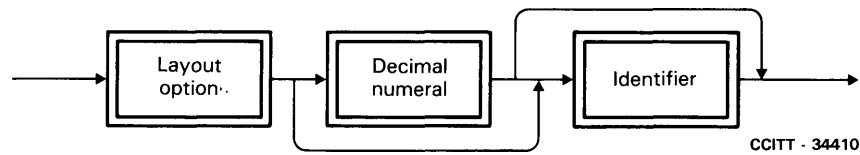
7.4.5.2.2 *Command code output*



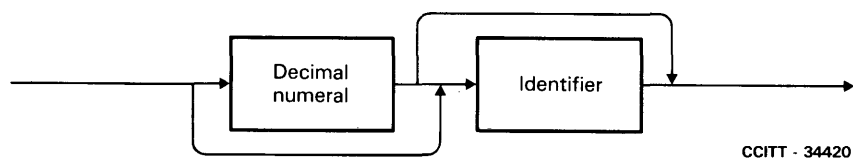
7.4.5.2.3 *Menu output*



7.4.5.2.4 *Selection identity*

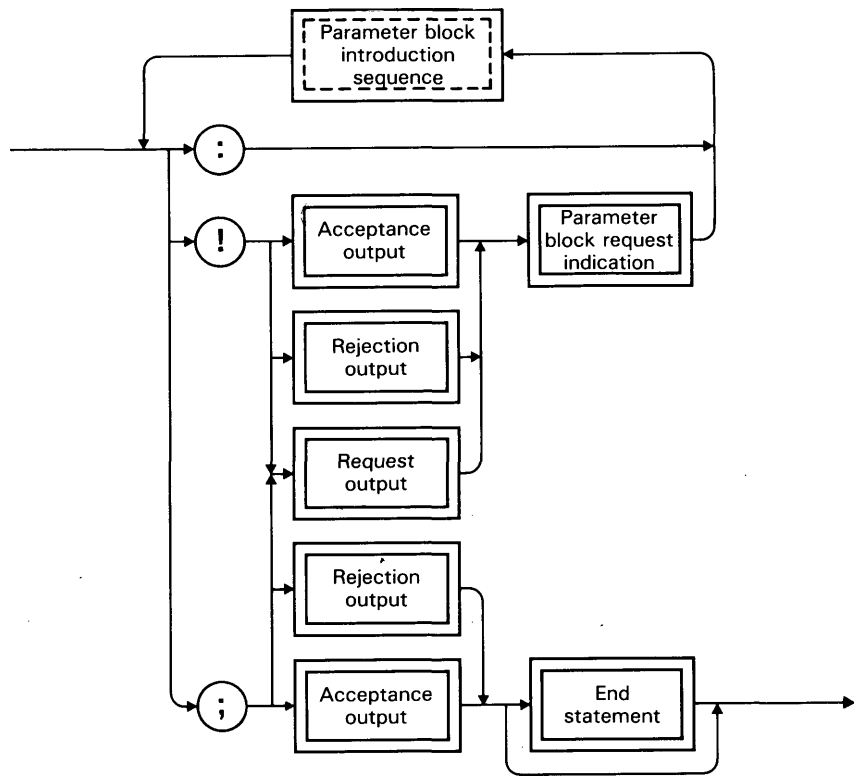


7.4.5.2.5 *Selection input*



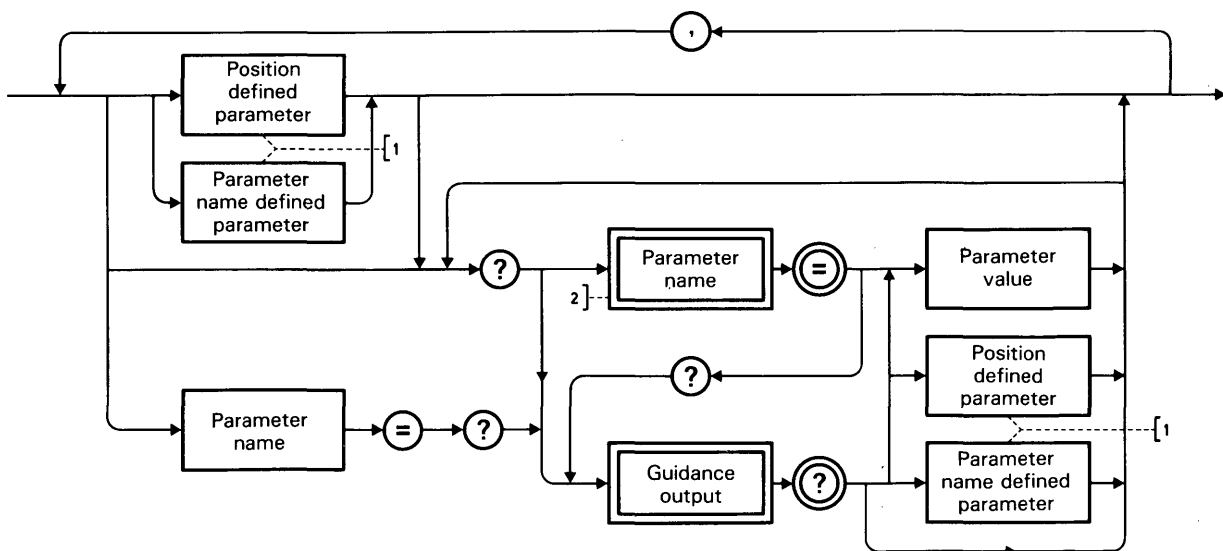
7.4.6 Methods of inputting parameters

7.4.6.1 Direct parameter input



CCITT - 34430

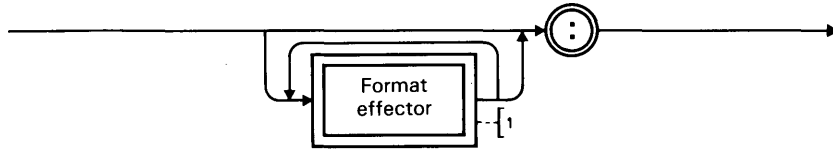
7.4.6.1.1 Parameter block introduction sequence



CCITT - 29671

- 1) It is not allowed to mix parameters of different types within a block of parameters.
- 2) See Recommendation Z.315.

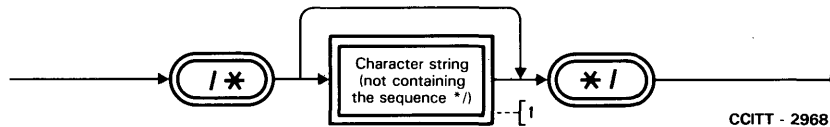
7.4.6.1.2 *Parameter block request indication*



CCITT - 34440

1) Not further expanded in diagram form.

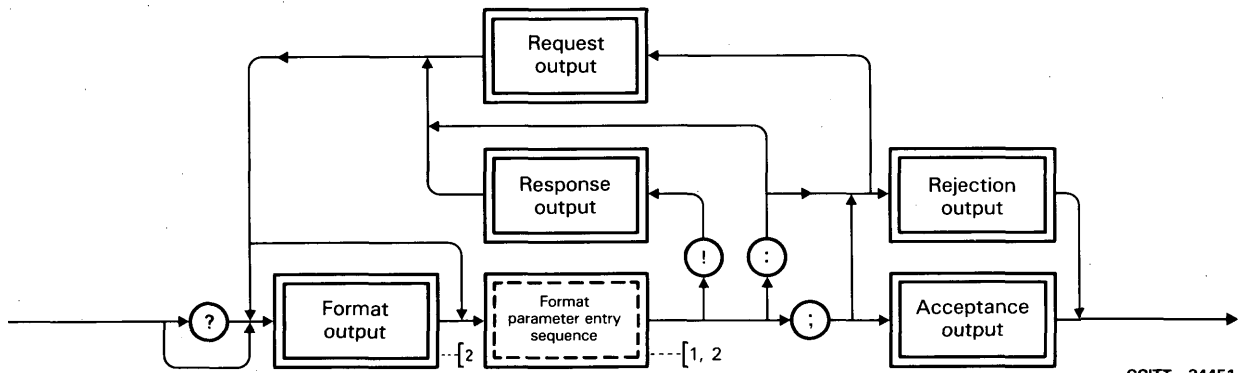
7.4.6.1.3 *Guidance output*



CCITT - 29681

1) Not further expanded in diagram form.

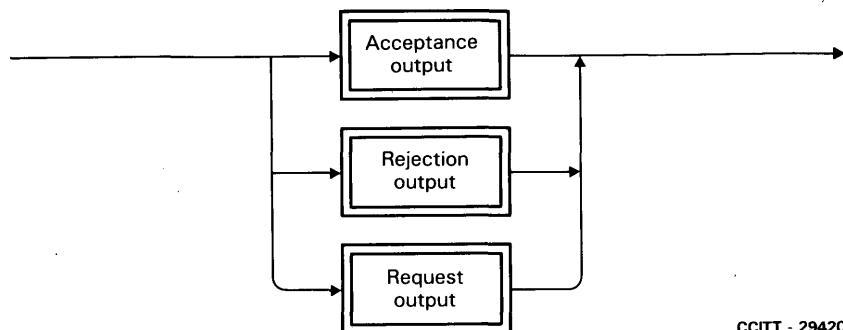
7.4.6.2 *Format parameter input*



CCITT - 34451

- 1) Format parameter entry sequence can include system prompted error recovery sequences.
- 2) Not at present further expanded in diagram form in a Recommendation.

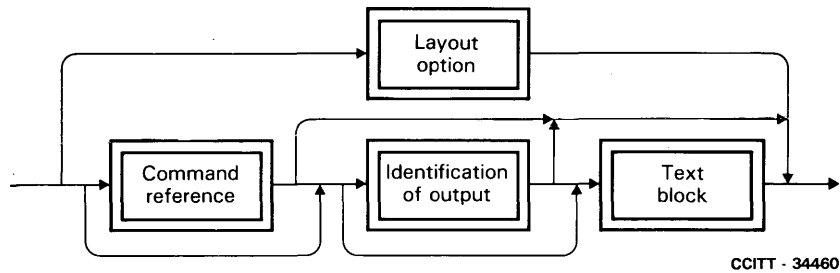
7.4.7 *Response output*



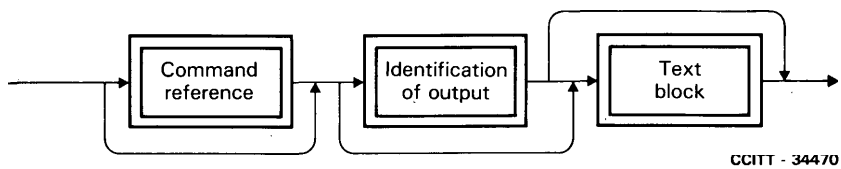
CCITT - 29420



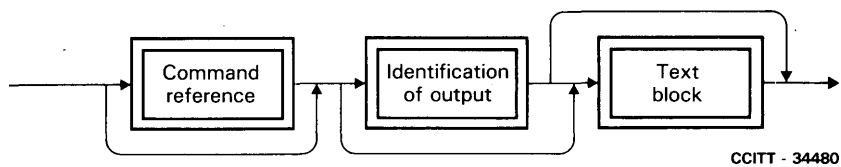
### 7.4.7.1 Acceptance output



### 7.4.7.2 Rejection output



### 7.4.7.3 Request output



## 7.5 Input output management

### 7.5.1 General

The question of input output management is highly hardware and system dependent. Input output management strategies should be provided to:

- solve any conflict of output directed to an input/output (I/O) device involved in a dialogue procedure;
- solve any conflict of more than one output competing for the same I/O device;
- permit the initiator to perform a dialogue at any time.

### 7.5.2 Priorities of output

The priority of an output will determine the behaviour of the output in relation to a dialogue procedure and in relation to other outputs. System crash messages and those outputs that occur after a dangerous situation, implying an immediate recovery procedure such as system reload, are not governed by the following input/output management procedures but may be output at any time.

The priority of an output is the property of the output and dictates the sequence of the output. When several outputs are competing for the use of the same I/O device, the output with the highest priority is output first. Outputs of the same priority are output on a first come first served basis. From an input/output management point of view there shall be two classes of priority for output: high, low.

Lengthy outputs shall be divided into convenient units. Interruptions of output shall only occur at the end of an output unit. A suitable dimension for a unit of output shall be sufficient to allow the output of a meaningful message.

### 7.5.3 Output to a device not in a dialogue procedure

An output directed to an I/O device not involved in a dialogue procedure is always output, unless another output is in progress on that I/O device, in which case the current output must be completed first. These outputs may be interrupted by input (see § 7.5.5).

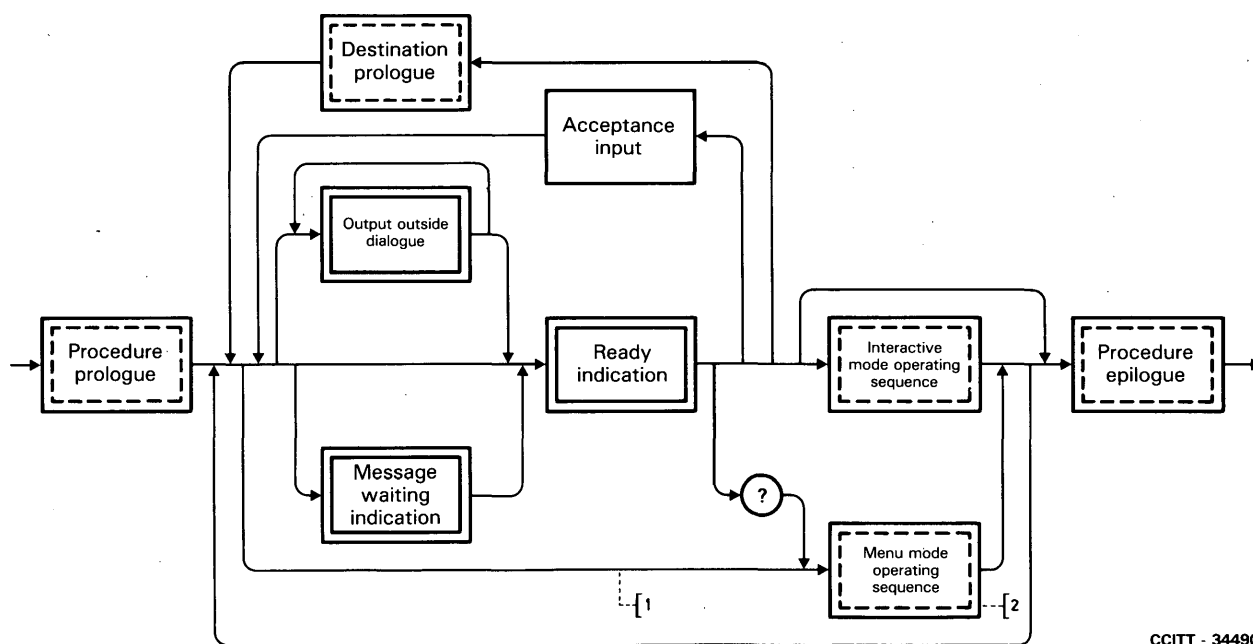
Optionally a system may choose to output the current output only up to the end of the current unit of output before outputting a waiting high priority output.

### 7.5.4 Output to a device in a dialogue procedure

High priority outputs, which are outputs outside dialogue, are allowed either to be announced or to interrupt the dialogue between interactive mode operating sequences<sup>1)</sup>. When a high priority output is announced, by means of a message waiting indication, an acceptance input can be given which will cause the waiting output to take place (see § 7.5.4.1 for an extended syntax diagram for output interrupting input).

Low priority outputs, which are outputs outside dialogue, are not allowed to be announced or to interrupt the dialogue and should be delayed until the end of the dialogue.

#### 7.5.4.1 Interruption in dialogue due to input/output management



- 1) Menu mode may be the default operating system for a particular terminal or a separate command may direct the system to operate in such a way. Therefore it is necessary to provide a direct path from procedure prologue to menu mode operating sequence. If neither of these methods is the case, then the ready indication is given in the usual way, and if the menu mode is required, a ? (question mark) must be input.
- 2) This operating sequence offers also the destination prologue facility in menu mode.

<sup>1)</sup> Interruption in other places requires further study.

#### 7.5.4.2 *Message waiting indication*

Recommendation still under study.

#### 7.5.4.3 *Acceptance input*

Recommendation still under study.

#### 7.5.5 *Input interrupting output*

A facility is provided to allow the interruption of an output occurring at an I/O device. However, a request, rejection or acceptance output (where it is not used as the result of the actual action) cannot be interrupted. The output may be interrupted by means of a request as defined in § 7.2.2.1. When the above request has been made the dialogue with the machine can be started/continued.

The interrupted output may be managed by giving an instruction to resume, cancel or restart it. Alternatively, the interrupted output may be managed according to the property of the message itself, assigned at the time of message design.

When the interrupt request is given, the interrupt shall be carried out after the current unit of output.

#### 7.6 *Time-out control inside dialogue*

Two particular time-outs are identified within a dialogue. The time-outs are provided to prevent lockout of outputs and/or to prove the presence of the initiator. The latter is used when the system has functions for procedure prologue and epilogue. In this case two time-outs may be provided where the first one is used within any input. The second time-out is set after completion of the procedure prologue, the destination prologue, continuation and menu mode operating sequence. Both time-outs are cancelled by the receipt of any input.

When the first time-out elapses, it is suggested that cancellation of the actual input should occur. When the second time-out elapses, it is suggested that the epilogue procedure should take place. Any output can take place when the first time-out has elapsed.

### **Recommendation Z.318**

## **8. LIST OF FUNCTIONS**

### 8.1 *Introduction*

A preliminary list of functions which are expected to be controlled by means of the MML is given in this Recommendation <sup>1)</sup>.

The functions are listed under four main function areas:

operation, maintenance, installation and testing.

Because of potentially different national needs some functions appear under more than one heading.

### 8.2 *Functions*

#### 8.2.1 *General functions*

In addition to those functions indicated below which imply changing or setting data there are also the corresponding interrogation functions.

---

<sup>1)</sup> This list was agreed jointly in studies of Study Groups XI and XIII during the Study Period 1973-1976. The list is not to be regarded as complete. The purpose of preparing a list of functions was to ensure that the MML recommended by the CCITT would be adequate to cover all the necessary functions.

## 8.2.2 *Operational functions*

### 8.2.2.1 *Subscriber administration*

- taking subscribers' lines into/out of service <sup>2)</sup>;
- allocate change and remove classes of subscriber service;
- changing of a subscriber's number;
- blocking and unblocking a subscriber's line;
- interrogation of a subscriber's classes of service;
- interrogation of blocked subscriber lines;
- reading of subscriber's charging information;
- retrieval of charging information;
- malicious call tracing;
- putting a subscriber on to *subscriber charging observation*, etc.

### 8.2.2.2 *Routing administration*

#### 8.2.2.2.1 *Changing data relating to a group of circuits*

- changing of signalling system;
- inserting a new group of circuits;
- changing search order in a bidirectional group of circuits;
- adding a new circuit;
- changing the group affiliation of a circuit;
- changing the position of a circuit from one inlet or outlet to another inlet or outlet in the switching matrix.

#### 8.2.2.2.2 *Changing routing and analysis data*

- changing the alternative routing table;
- changing of traffic routing for network management purposes;
- changing of analysis data (start of selection, number of digits to be sent, etc.).

### 8.2.2.3 *Traffic administration*

- traffic recording according to CCITT method No. 1 (Recommendation E.261);
- traffic recording according to CCITT method No. 2 (Recommendation E.261);
- changing of traffic load control criteria;
- analysis of destinations of incoming traffic;
- measurements of subscribers' behaviour;
- means for interrogating the measurement parameters of different measurement groups;
- means for inserting a given route, circuit, etc., into, or removing it from, a series of measurements;
- outputting of data relating to grade of service.

See network management actions in § 5.2 of Recommendation E.410 (former Recommendation Q.55).

See measuring and recording of traffic, Recommendations E.500 and E.501.

See Recommendations E.420, E.421 and E.422 to E.425 of the *Orange Book* (formerly Recommendations Q.60, Q.60 bis, and Q.61 to Q.64 of the *Green Book*).

---

<sup>2)</sup> Subscribers' lines includes lines on PABX.

#### 8.2.2.4 *Tariff and charging administration*

- changing tariff for a certain traffic destination;
- changing parameters for a charging rate;
- changing time for switching between day and night rate;
- reading of accounting statistics (accounting between operating companies);
- changing the parameters involved in the accounting methods for traffic between different operating companies.

#### 8.2.2.5 *System control operation*

- setting and reading of calendar;
- loading of overlay programmes;
- changing authority for an input device;
- changing output device for a certain output;
- changing the code to be transmitted by a tape reader unit or to a tape punch as, for example, in ATME;
- defining a file being set up on a magnetic tape equipment;
- changing working mode for a certain programme;
- changing of system configuration;
- starting tape reader or magnetic tape equipment;
- loading a new programme package.

#### 8.2.3 *Maintenance functions*

##### 8.2.3.1 *Maintenance of subscribers' lines*

- test of one subscriber's line and associated equipment;
- test of a group of subscribers' lines and associated equipment;
- measurement of one subscriber's line and associated equipment;
- measurement of a group of subscribers' lines and associated equipment;
- blocking or unblocking a subscriber's line for maintenance purposes;
- observation or supervision of subscribers' lines and equipment.

##### 8.2.3.2 *Maintenance of lines between exchanges*

- test of one line and associated equipment;
- test of a group of lines and associated equipment;
- measurement of one line and associated equipment;
- measurement of a group of lines and associated equipment;
- blocking or unblocking a line for maintenance purposes;
- observation or supervision of lines and equipment.

##### 8.2.3.3 *Switching network maintenance*

- making test calls;
- call trace;
- holding faulty connections;
- testing and measuring peripheral equipment (relay sets, signalling receivers and senders, etc.);
- testing and measuring switch units;
- reducing service for low priority subscribers;
- setting up a connection via a specific path through the network;

- supervision and measurement of quality of service of switching network;
- fault localization in the speech path network;
- access for traffic observation for maintenance purposes;
- alarm reporting;
- switch unit status recording.

#### 8.2.3.4 *Control system maintenance*

##### Hardware:

- system status reporting;
- alarm reporting and fault localization;
- functional testing after repair;
- initiating periodic testing operations;
- changing system configuration for maintenance purposes.

##### Software:

- checking consistency of data (e.g. checksumming);
- initiating restart;
- setting traps for programme fault tracing;
- changing memory contents;
- memory dumping for maintenance purposes.

##### Load control:

- omitting non-essential functions in overload situations;
- changing the criteria for the recognition of degradation of service;
- reducing service for low-priority subscribers.

##### Network management:

- changing routing information;
- changing criteria for initiating network management actions.

#### 8.2.4 *Plant installation functions*

The configuration of an exchange is described by several parameters of limits such as:

- maximum number of subscribers;
- maximum number of trunks;
- wiring of equipments;
- list of existing services or facilities;
- list of existing signalling systems;
- maximum size of data tables, etc.

Any change in software or hardware under these limits is considered as an operational function, e.g. adding equipments or features.

To initiate or change some of these limits (first installation or extension) specific functions have to be executed (the same kind of functions in both cases are called plant installation functions).

##### 8.2.4.1 *Switching system installation*

Testing and data generation functions for:

- new network-blocks installation;
- new trunks installation;
- new signalling-equipment installation;
- new test-equipment installation;
- new blocks of subscriber-circuits installation;
- new interface-equipment installation.

#### 8.2.4.2 *System control installation*

Installation of new software:

- new operational packages;
- new test programmes;
- new statistics programmes;
- new overlay programmes;
- new signalling systems;
- new services, facilities and tariff.

Extension of system control:

- control unit;
- memory;
- input/output devices.

Table generation:

- subscribers;
- routing and tariffs.

#### 8.2.5 *Testing functions*

The memory dump expressing:

- the begin and end of the dump in absolute addressing;
- the begin and end of the dump in symbolic addressing;
- the wish to have the dump in a binary (hexadecimal/decimal) form;
- changing programmes and data.

Adding patches:

- patches in a case of programme changing;
- reloading patches in a reload phase;
- starting and stopping a programme trace;
- starting and stopping a data trace.

Under simulated conditions:

- test programmes;
- test hardware equipment;
- start test calls;
- stop at a particular address;
- loading new programme packages;
- restarting the machine (from certain conditions).

**SECTION 2**

**SPECIFICATION OF FUNCTIONS**

(not yet defined; it will contain Recommendations Z.321 to Z.329)

**SECTION 3**

**USERS MANUAL**

(not yet defined; it will contain Recommendations Z.331 to Z.339)



**PAGE INTENTIONALLY LEFT BLANK**

**PAGE LAISSEE EN BLANC INTENTIONNELLEMENT**

## SECTION 4

### GLOSSARY OF TERMS

#### Recommendation Z.341

### GLOSSARY OF TERMS

#### 1 General

The aim of the glossary for the man-machine language is to include terms used in describing the man-machine language. It comprises, in alphabetical order, all terms well established in current usage which are not made up of compound words in their ordinary sense (i.e. unambiguous and self-explanatory words) and hence require definition.

The terms in italics in the text of the definitions are defined elsewhere in this glossary.

#### 2 List of terms

##### acceptance input

*F: entrée d'acceptation*

*S: entrada de aceptación*

An *input* used to allow the *system* to *output* an announced high priority *output*, shown by a *message waiting indication*.

##### acceptance output

*F: sortie d'acceptation*

*S: salida de aceptación*

An *output* message indicating that an *input* to the system is syntactically correct and complete and that the appropriate system actions will be initiated, or have already been carried out. In the latter case, this indication may take the form of the actual result.

##### additional information

*F: information supplémentaire*

*S: información adicional*

Provides information supplementary to the actual *output*, such as type of *output*, e.g. maintenance, traffic data or *output* recipient identification.

##### additional header information

*F: information d'en-tête supplémentaire*

*S: información adicional de encabezamiento*

Provides information supplementary to the actual *output header*, such as sequence number, processor number, *output* device, or day of the week.

**administrative processor**

*F: processeur de gestion*

*S: unidad de proceso para la gestión*

A centralized processor for administrative purposes, e.g. billing, which serves several switching centres.

**alarm statement**

*F: instruction d'alarme*

*S: sentencia de alarma*

A statement providing information concerning an alarm condition, such as the degree (level) of alarm or the source of the alarm.

**annotation symbol**

*F: symbole d'annotation*

*S: símbolo de anotación*

A symbol (— — — [n where n is a number referencing a note) used in the *meta-language* to add descriptive comments or explanatory notes as clarification.

**arithmetic delimiters**

*F: délimiteurs arithmétiques*

*S: delimitadores aritméticos*

A symbol used to denote the arithmetic operation(s) to be performed in an *arithmetic expression*. Allowed delimiters are: + (plus sign), - (hyphen), / (solidus), \* (asterisk), ( ) (left and right parentheses).

**arithmetic expression**

*F: expression arithmétique*

*S: expresión aritmética*

A combination of *arithmetic delimiters*, *numerals (decimal, hexadecimal, octal, or binary)* and *identifiers* enclosed by parentheses.

**binary numeral**

*F: nombre binaire*

*S: numeral binario*

A *numeral* in the binary (base 2) *numbering system*, represented by the characters 0 (zero), 1 (one) and optionally preceded by B' (B apostrophe).

**block of parameters**

*F: bloc de paramètres*

*S: bloque de parámetros*

A set of interrelated *parameters* containing information necessary for the *system* to perform the action or function specified in the *command*.

**CCITT MML**

*F: LHM du CCITT*

*S: LHM del CCITT*

The *man-machine language* (MML) for stored program controlled switching systems developed by the International Telegraph and Telephone Consultative Committee (CCITT).

**character**

*F: caractère*

*S: carácter*

A member of the *character set* which is used for the organization, control, or representation of data.

**character set**

*F: ensemble de caractères*

*S: conjunto de caracteres : (o juego de caracteres)*

The finite set of different *characters* used in *CCITT MML*.

**clarifying text**

*F: texte explicatif*

*S: texto aclaratorio*

A set of *information units* used to make the purpose and content of the output clearer.

**command**

*F: commande*

*S: instrucción*

A specification of an expected action or function by the *system*.

**command code**

*F: code de commande*

*S: código de instrucción*

A set of up to 3 *identifiers*, separated by – (hyphen), used to specify the expected action or function the *system* is to perform.

**command reference**

*F: référence de commande*

*S: referencia de instrucción*

A reference to a previous *input*, appearing in *output outside dialogue* and *dialogue procedures*, in the form of a *command sequence number* and, possibly, *clarifying text*.

**command sequence number**

*F: numéro de séquence de commande*

*S: número secuencial de instrucción*

A reference number uniquely identifying a previous *input*.

**comment**

*F: commentaire*

*S: comentario*

A *character* string enclosed between the separators /\* (solidus asterisk) and \*/ (asterisk solidus). It has no *MML* syntactical or semantical meaning.

**compound parameter argument**

*F: argument de paramètre composé*

*S: argumento de parámetro compuesto*

A *parameter argument* made up of more than one *information unit*. It is used to specify a multidimensional object or value. E.g. a date can be expressed as 1979-12-31.

**continuation character**

*F: caractère de répétition*

*S: carácter de continuación*

A *character* that allows for continuation of the man-machine dialogue in the *continuation mode*.

**continuation mode**

*F: mode répétitif*

*S: modo continuación*

A mode of operation in which more than one *command* with the same *command code* can be given in a sequence without repeating the *command code*.

**continuation mode operating sequence**

*F: séquence de fonctionnement dans le mode répétitif*

*S: secuencia de funcionamiento en el modo continuación*

The sequence of operations required to implement *continuation mode*.

**control character**

*F: caractère de commande*

*S: carácter de control*

A *character* whose occurrence in a particular context initiates, modifies, or stops an action that affects the recording, processing or interpretation of data.

**correction character**

*F: caractère de correction*

*S: carácter de corrección*

A *character* used to invoke correction facilities prior to analysis of *input*.

**decimal numeral**

*F: nombre décimal*

*S: decimal numeral*

A *numeral* in the decimal (base 10) *numbering system*, represented by the *characters*, 0 (zero), 1, 2, 3, 4, 5, 6, 7, 8, 9 optionally preceded by D' (D apostrophe).

**default value**

*F: valeur de défaut*

*S: valor por defecto*

The value given to any *parameter* in the absence of a specific value in the *input*.

**deletion character**

*F: caractère d'effacement*

*S: carácter de borrado*

A *character* that causes the *system* to respond with an acknowledgement that present *input* after the last *command* executed is cancelled.

**delimiter**

*F: délimiteur*

*S: delimitador*

A *character* that organizes and separates items of data.

**destination identifier**

*F: identificateur de destination*

*S: identificador de destino*

Indicates the physical area where the *command* is processed.

**digit**

*F: chiffre*

*S: dígito (cifra)*

A *character* of the *character set* representing an integer, listed in Table 1/Z.314, column 3, positions 0 (zero) to 9.

**end of dialogue**

*F: fin de dialogue*

*S: fin de diálogo*

The indication that the dialogue is finished.

**end of output**

*F: fin de sortie*

*S: fin de salida*

The indication that the *output outside dialogue* is finished.

**end statement**

*F: instruction de fin*

*S: sentencia de fin*

Terminates *output* information from the system in an operating sequence where termination is not obvious.

**escape indication**

*F: indication d'échappement*

*S: indicación de escape*

A mechanism to indicate that following *character(s)* are not to be interpreted according to the normal *syntax* rules.

**exchange control system**

*F: système de commande du central*

*S: sistema de control de central*

The central control *system* of a stored program controlled switching *system*. It may consist of one or more *processors*.

**execution character**

*F: caractère d'exécution*

*S: carácter de ejecución*

A *character* which requests that the *command* be executed.

**flowline**

*F: ligne de liaison*

*S: línea de flujo*

A line representing a connection path between *symbols* in a *syntax diagram*.

**format**

*F: format*

*S: formato*

The arrangement or layout of data on a data medium.

**format effector**

*F: caractère de mise en page*

*S: determinante de formato*

Any *character(s)* used to control the positioning of printed, displayed or recorded data.

**format parameter entry sequence**

*F: séquence d'entrée de paramètres avec format*

*S: secuencia de introducción formatizada de parámetro*

The sequence performed when entering *parameter values* in a *format output*.

**format parameter input**

*F: entrée de paramètre avec format*

*S: introducción formatizada de parámetro*

A *format output*, and a *format parameter entry sequence* followed by an execution.

**format output**

*F: sortie de format*

*S: salida de formato (formatizada)*

An *output* of the *parameters* belonging to a *command* leaving empty positions for insertion of *parameter values*.

**function**

*F: fonction*

*S: función*

An action which various groups of staff wish to carry out, e.g. add subscriber's line, initiate a testing routine, read a subscriber's class of service. To carry out one function, one or more *commands* may be necessary. The function is characterized by the *command code(s)*.

**graphic characters**

*F: caractères graphiques*

*S: caracteres gráficos*

A collection of *characters* within the *character set* used to improve readability of *output*.

**guidance output**

*F: sortie d'instruction*

*S: salida de orientación*

*Output* which appears as a *comment* in *output* providing direction to the *initiator of MML input*.

**header**

*F: en-tête*

*S: encabezamiento*

General information which could comprise identification information, date and time, etc.

**hexadecimal numeral**

*F: nombre hexadécimal*

*S: numeral hexadecimal*

A *numeral* in the hexadecimal (base 16) *numbering system*, represented by the *characters* 0 (zero), 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, optionally preceded by H' (H apostrophe).

**identifier**

*F: identificateur*

*S: identificador*

A representation of an entity, typically consisting of one or more *characters*. It is used to identify or name a unique item of data. In the *man-machine language*, the first character is a letter.

**indicator**

*F: indicateur*

*S: indicador*

A *character* given by an initiator or machine (*system*) used to indicate a state or to *request* initiator or machine action.

**information unit**

*F: unité d'information*

*S: unidad de información*

The smallest part of data in the *input* or *output*.

**initiator of MML input**

*F: opérateur d'une entrée LHM*

*S: iniciador de entrada LHM*

The "man" in the term "Man/Machine Language". Any person able to input information via a keyboard or similar device and to observe *output* on any visible medium.

**input**

*F: entrée*

*S: entrada*

The process that constitutes the introduction of data into a data processing *system* or any part of it.

**interactive mode operating sequence**

*F: séquence de fonctionnement en mode interactif*

*S: secuencia de funcionamiento en el modo interactivo*

A sequence which may consist of a single *continuation mode operating sequence* terminated by an optional *end statement* or of a series of *continuation mode operating sequences* or *special actions*. The latter occurs when, as a result of partial execution of a function, the machine requests the *initiator* to supply it with further information in the form of specific actions or further *commands* for which *initiator* judgment and/or decision is required.

**I/O devices**

*F: dispositif d'entrée-sortie*

*S: dispositivos de entrada/salida*

Memory and keyboard devices for entering or receiving data to or from the *system*. Can be controlled manually for entering or receiving data.

**keyed numeral**

*F: nombre composé au clavier*

*S: numeral de teclado*

A *numeral* in a *numbering system* based on keypad *input*, represented by the *characters* 0 (zero), 1, 2, 3, 4, 5, 6, 7, 8, 9, \*, #, A, B, C, D, optionally preceded by K' (K apostrophe).

**layout option**

*F: option de présentation*

*S: opción de estructuración*

A combination of *format effectors* and/or *graphic characters* used to bound elements of the *output* in a clear and readable form.



**letter**

*F: lettre*

*S: letra*

A *character* of the *character set* representing the alphabet, listed in Table 1/Z.314, columns 4, 5, 6 and 7 excluding table positions 5/15 and 7/15.

**man-machine language (MML)**

*F: langage homme-machine (LHM)*

*S: lenguaje hombre-máquina (LHM)*

The communication medium used between the *initiator* of MML *input* and the *system*.

**man-machine terminal**

*F: terminal homme-machine*

*S: terminal hombre-máquina*

An input/output device, consisting of a keyboard and a display unit, used to allow the *initiator* of MML *input* and the *system* to communicate with each other.

**menu mode**

*F: mode menu*

*S: modo menú*

A mode of operation in which certain actions can be performed by choosing the appropriate item presented in the menu *outputs*.

**menu mode operating sequence**

*F: séquence de fonctionnement en mode menu*

*S: secuencia de funcionamiento en modo menú*

Provides a *menu selection procedure* both initially to arrive at the first *command code* and if necessary, to provide a destination prologue, and also subsequently whenever a further *command code* is required.

**menu output**

*F: sortie en mode menu*

*S: salida en modo menú*

A list of items, together with optional instructional text. One of these items may be chosen.

**menu selection procedure**

*F: procédure de sélection en mode menu*

*S: procedimiento de selección en modo menú*

A selection procedure used to obtain a *command code*, a destination prologue, or execute a complete *command*.

**message waiting indication**

*F: indication d'attente de message*

*S: indicación de mensaje en espera*

A means of announcing, within a dialogue procedure, the presence of a high priority *output* for this *I/O device*.

**meta-language**

*F: métalangage*

*S: metalenguaje*

A symbolic method for defining MML *input* and *output syntax*.

**mnemonic abbreviation**

*F: abréviation mnémonique*

*S: abreviatura nemotécnica*

A representation of an entity typically consisting of one or more *characters* chosen to assist the human memory.

**non-decimal numeral**

*F: nombre non décimal*

*S: numeral no decimal*

A numeral in a numbering system other than decimal.

**non-terminal symbol**

*F: symbole non terminal*

*S: símbolo no terminal*

Representation, within a *syntax diagram*, of another *syntax diagram* by name. It is an abbreviated symbol for a more complex construct.

**numeral**

*F: désignation numérique*

*S: numeral*

A discrete representation of a number within a numbering system.

**numbering system**

*F: système de numération*

*S: sistema de numeración*

Any notation for the representation of numbers.

**octal numeral**

*F: nombre octal*

*S: numeral octal*

A numeral in the octal (base 8) numbering system, represented by the characters 0, 1, 2, 3, 4, 5, 6, 7, optionally preceded by O' (letter O apostrophe).

**operation and maintenance centre processor**

*F: processeur de centre d'exploitation et de maintenance*

*S: unidad de proceso (procesador) para centro de explotación y mantenimiento*

A centralized processor for operation and maintenance purposes which serves one or more switching centres.

**output**

*F: sortie*

*S: salida*

The process that consists of the delivery of data from a data processing system or from any part of it.

**output outside dialogue**

*F: sortie hors dialogue*

*S: salida fuera de diálogo*

A spontaneous output indicating a certain event, e.g. an alarm situation, or an output in response to a command entered in an interactive mode operating sequence, e.g. a traffic measurement result.

**parameter**

*F: paramètre*

*S: parámetro*

Data which identifies and contains pieces of necessary information to execute a *command*.

**parameter argument**

*F: argument de paramètre*

*S: argumento de parámetro*

A part of a *parameter value* which specifies an appropriate object or value. It consists of one or more *information units* separated by - (hyphen).

**parameter block**

*F: bloc de paramètres spécifiques*

*S: bloque de parámetros específicos*

A number of *parameters* specific to a *command*.

**parameter block introduction sequence**

*F: séquence d'introduction de bloc de paramètres*

*S: secuencia de introducción de bloque de parámetros*

A procedure used to *input a block of parameters*.

**parameter block request indication**

*F: indication de demande de bloc de paramètres*

*S: indicación de petición de bloque de parámetros*

An indication from the *system* to the *initiator* to proceed with input of *parameters*.

**parameter name**

*F: nom de paramètre*

*S: nombre de parámetro*

An *identifier* which indicates unambiguously the meaning and structure of the subsequent *parameter value*.

**parameter name defined parameter**

*F: paramètre défini par le nom de paramètre*

*S: parámetro definido por el nombre de parámetro*

A *parameter* which is identified by its *parameter name*.

**parameter value**

*F: valeur de paramètre*

*S: valor de parámetro*

The part of a *parameter* that contains the information required to specify any appropriate object(s) or value(s). It consists of one or more *information units*.

**password**

*F: mot de passe*

*S: contraseña*

A *character string* used for identification and authorization of an *initiator*.

**position defined parameter**

*F: paramètre défini par la position*

*S: parámetro definido por la posición*

A *parameter* whose nature is identified by its position in the *parameter block* of a *command*.

**procedure epilogue**

*F: épilogue de procédure*

*S: epílogo de procedimiento*

The procedure used to terminate the *dialogue procedure* and may consist of a manual action by the *initiator* to the machine to deactivate the *man-machine terminal* and/or an *output* from the system to indicate the end of the dialogue.

**procedure prologue**

*F: prologue de procédure*

*S: prólogo de procedimiento*

A set of actions to activate the *man-machine terminal*, to call the *system* and to identify the *initiator*.

**processor**

*F: processeur*

*S: unidad de proceso (procesador)*

A device capable of performing systematic execution of operations upon data.

**prompting**

*F: intervention*

*S: sugerencia*

A method used by the *system* to request *input* from the *initiator* in a *dialogue procedure*.

**prompting output**

*F: sortie d'intervention*

*S: salida de sugerencia*

An *output* from the system consisting of the *parameter name* and an = (equal sign) as requested by the *initiator*.

**ready indication**

*F: indication "prêt"*

*S: indicación de preparado*

An *output* element used in a *dialogue procedure* to indicate that the direction of the dialogue has changed and that the *system* is ready to receive a *command* or a *destination identifier*. It is also used in the *identification invitation* to prompt the *initiator* to input his *password*.

**ready indicator**

*F: indicateur "prêt"*

*S: indicador de preparado*

An *indicator* used in the *ready indication* to indicate that the *system* is ready to receive information.

**rejection output**

*F: sortie de rejet*

*S: salida de rechazo*

An *output* message indicating that an *input* to the *system* is invalid and will not be acted upon, and corrections by the *input initiator* cannot be applied.

**request**

*F: demande*

*S: petición*

A manual action used to activate a *man-machine terminal* and the *system*.

**request output**

*F: sortie de demande*

*S: salida de petición*

An *output* message in the *dialogue procedure* which can appear as response to an *input* by the *initiator*. It requests further *input* action from the *initiator*, e.g. correction of an erroneous *parameter*, or supplying further information.

**response output**

*F: sortie de réponse*

*S: salida de respuesta*

An *output* message in the *dialogue procedure* which gives information about the state of an *input*. The *output* can be of any of the following types: *acceptance output*, *rejection output* and *request output*.

**selection acknowledgement output**

*F: sortie d'accusé de réception de sélection*

*S: salida de confirmación (acuse de recibo) de selección*

An *output* message in the *menu selection procedure* which is used to acknowledge a selection between menu items made by the *initiator* in the *menu mode operating sequence*.

**selection identity**

*F: identité de sélection*

*S: identidad de selección*

A label preceding each item of the *menu output* in order to allow a choice to be made.

**selection input**

*F: entrée de sélection*

*S: entrada de selección*

An *input* in the *menu selection procedure* indicating the menu item selected by the *initiator*.

**semantics**

*F: sémantique*

*S: semántica*

The rules and conventions governing the interpretation and assignment of meaning to constructions in a language.

**separator**

*F: séparateur*

*S: separador*

A *character* used to delimit *syntax* elements.

**simple parameter argument**

*F: argument de paramètre simple*

*S: argumento de parámetro simple*

A *parameter argument* made up of only one *information unit*.

**source identifier**

*F: identificateur d'origine*

*S: identificador de fuente*

One or more *information units* indicating the physical area where an *output* was generated.

**special character combination**

*F: combinaison spéciale de caractères*

*S: combinación especial de caracteres*

A pre-defined string of *characters* used to specify the base of a *numeral*.

**spontaneous output**

*F: sortie spontanée*

*S: salida espontánea*

An *output* generated by internal events of the *system*, e.g. an alarm.

**symbol**

*F: symbole*

*S: símbolo*

A conventional representation of a concept or a representation of a concept upon which agreement has been reached.

**symbolic name**

*F: nom symbolique*

*S: nombre simbólico*

A *character* string used for the representation of an entity.

**syntax**

*F: syntaxe*

*S: sintaxis*

The rules for the formation of permissible constructions (e.g. *character* strings) in a language, without regard to meaning.

**syntax diagram**

*F: diagramme syntaxique*

*S: diagrama sintáctico*

A method of defining the *syntax* of the *input* and *output* language by pictorial representation.

**system**

*F: système*

*S: sistema*

Refers to a stored program control switching *system* and also to its man-machine communication facility.

**table**

*F: tableau*

*S: tabla (cuadro)*

An ordered presentation of interrelated information.

**terminal**

*F: terminal*

*S: terminal*

Abbreviation for *man-machine terminal*.

**terminal symbol**

*F: symbole terminal*

*S: símbolo terminal*

A terminal symbol is a *character* or string of *characters* which actually appear in the *input* or *output*.

**text block**

*F: bloc de texte*

*S: bloque de texto*

Any combination of *clarifying texts*, *variable texts*, *parameter name defined parameters* and/or *tables* which gives *output* information wherever it is needed or requested.

**text string**

*F: chaîne de texte*

*S: cadena de texto*

A *character* string [excluding “ (quotation mark) and *correction characters*] not interpreted within the *man-machine language* but stored in the machine. Its *output format* is identical to its *input format*.

**variable text**

*F: texte variable*

*S: texto variable*

A string of *information units* which contains information unique to the event which caused the *output*.

**SECTION 5**

**IMPLEMENTORS GUIDE**

(not yet defined; it will contain Recommendations Z.351 to Z.359)



