UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

CCITT

COMITÉ CONSULTIVO
INTERNACIONAL
TELEGRÁFICO Y TELEFÓNICO

LIBRO ROJO

TOMO VI – FASCÍCULO VI.11

LENGUAJE DE ESPECIFICACIÓN Y DE DESCRIPCIÓN FUNCIONALES (LED)

ANEXOS A LAS RECOMENDACIONES Z.100 A Z.104

VIII ASAMBLEA PLENARIA
MÁLAGA-TORREMOLINOS, 8-19 DE OCTUBRE DE 1984

Ginebra 1985

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.

UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

CCITT

COMITÉ CONSULTIVO
INTERNACIONAL
TELEGRÁFICO Y TELEFÓNICO

LIBRO ROJO

TOMO VI – FASCÍCULO VI.11

LENGUAJE DE ESPECIFICACIÓN Y DE DESCRIPCIÓN FUNCIONALES (LED)

ANEXOS A LAS RECOMENDACIONES Z.100 A Z.104

VIII ASAMBLEA PLENARIA
MÁLAGA-TORREMOLINOS, 8-19 DE OCTUBRE DE 1984

Ginebra 1985

ISBN 92-61-02243-X

CONTENIDO DEL LIBRO DEL CCITT EN VIGOR DESPUÉS DE LA OCTAVA ASAMBLEA PLENARIA (1984)

LIBRO ROJO

Tomo I – Actas e Informes de la Asamblea Plenaria.

Resoluciones y Ruegos.

Recomendaciones sobre:

- la organización de los trabajos del CCITT (serie A);
- los medios de expresión (serie B);
- las estadísticas generales de las telecomunicaciones (serie C).

Lista de las Comisiones de Estudio y de las Cuestiones en estudio.

Tomo II – *(Cinco fascículos, vendidos por separado.)*

FASCÍCULO II.1 – Principios generales de tarificación – Tasación y contabilidad en los servicios internacionales de telecomunicaciones. Recomendaciones de la serie D (Comisión de Estudio III).

FASCÍCULO II.2 – Servicio telefónico internacional – Explotación. Recomendaciones E.100 a E.323 (Comisión de Estudio II).

FASCÍCULO II.3 – Servicio telefónico internacional – Gestión de la red, ingeniería de tráfico. Recomendaciones E.401 a E.600 (Comisión de Estudio II).

FASCÍCULO II.4 – Servicios de telegrafía – Explotación y calidad de servicio. Recomendaciones F.1 a F.150 (Comisión de Estudio I).

FASCÍCULO II.5 – Servicios de telemática – Explotación y calidad de servicio. Recomendaciones F.160 a F.350 (Comisión de Estudio I).

Tomo III – *(Cinco fascículos, vendidos por separado.)*

FASCÍCULO III.1 – Características generales de las conexiones y circuitos telefónicos internacionales. Recomendaciones G.101 a G.181 (Comisiones de Estudio XV, XVI y CMBD).

FASCÍCULO III.2 – Sistemas internacionales analógicos de portadoras. Características de los medios de transmisión. Recomendaciones G.211 a G.652 (Comisión de Estudio XV y CMBD).

FASCÍCULO III.3 – Redes digitales – Sistemas de transmisión y equipos de multiplexación. Recomendaciones G.700 a G.956 (Comisiones de Estudio XV y XVIII).

FASCÍCULO III.4 – Transmisión en línea de señales no telefónicas – Transmisión de señales radiofónicas y de televisión. Recomendaciones de las series H y J (Comisión de Estudio XV).

FASCÍCULO III.5 – Red digital de servicios integrados (RDSI). Recomendaciones de la serie I (Comisión de Estudio XVIII).

Tomo IV – *(Cuatro fascículos, vendidos por separado.)*

FASCÍCULO IV.1 – Mantenimiento: consideraciones generales, sistemas internacionales de transmisión, circuitos telefónicos internacionales. Recomendaciones M.10 a M.762 (Comisión de Estudio IV).

FASCÍCULO IV.2 – Mantenimiento de circuitos internacionales de telegrafía armónica y de facsímil y de circuitos internacionales arrendados. Recomendaciones M.800 a M.1375 (Comisión de Estudio IV).

FASCÍCULO IV.3 – Mantenimiento de circuitos internacionales para transmisiones radiofónicas y de televisión. Recomendaciones de la serie N (Comisión de Estudio IV).

FASCÍCULO IV.4 – Especificaciones de los aparatos de medida. Recomendaciones de la serie O (Comisión de Estudio IV).

Tomo V – Calidad de transmisión telefónica. Recomendaciones de la serie P (Comisión de Estudio XII).

Tomo VI – *(Trece fascículos, vendidos por separado.)*

FASCÍCULO VI.1 – Recomendaciones generales sobre la conmutación y la señalización telefónicas – Interfaz con el servicio móvil marítimo y el servicio móvil terrestre. Recomendaciones Q.1 a Q.118 *bis* (Comisión de Estudio XI).

FASCÍCULO VI.2 – Especificaciones de los sistemas de señalización N.^{os} 4 y 5. Recomendaciones Q.120 a Q.180 (Comisión de Estudio XI).

FASCÍCULO VI.3 – Especificaciones del sistema de señalización N.º 6. Recomendaciones Q.251 a Q.300 (Comisión de Estudio XI).

FASCÍCULO VI.4 – Especificaciones de los sistemas de señalización R1 y R2. Recomendaciones Q.310 a Q.490 (Comisión de Estudio XI).

FASCÍCULO VI.5 – Centrales digitales de tránsito en redes digitales integradas y en redes mixtas analógico-digitales. Centrales digitales locales y combinadas. Recomendaciones Q.501 a Q.517 (Comisión de Estudio XI).

FASCÍCULO VI.6 – Interfuncionamiento de los sistemas de señalización. Recomendaciones Q.601 a Q.685 (Comisión de Estudio XI).

FASCÍCULO VI.7 – Especificaciones del sistema de señalización N.º 7. Recomendaciones Q.701 a Q.714 (Comisión de Estudio XI).

FASCÍCULO VI.8 – Especificaciones del sistema de señalización N.º 7. Recomendaciones Q.721 a Q.795 (Comisión de Estudio XI).

FASCÍCULO VI.9 – Sistema de señalización de acceso digital. Recomendaciones Q.920 a Q.931 (Comisión de Estudio XI).

FASCÍCULO VI.10 – Lenguaje de especificación y descripción funcionales (LED). Recomendaciones Z.101 a Z.104 (Comisión de Estudio XI).

FASCÍCULO VI.11 – Lenguaje de especificación y descripción funcionales (LED). Anexos a las Recomendaciones Z.101 a Z.104 (Comisión de Estudio XI).

FASCÍCULO VI.12 – Lenguaje de alto nivel del CCITT (CHILL). Recomendación Z.200 (Comisión de Estudio XI).

FASCÍCULO VI.13 – Lenguaje hombre-máquina (LHM). Recomendaciones Z.301 a Z.341 (Comisión de Estudio XI).

Tomo VII – *(Tres fascículos, vendidos por separado.)*

FASCÍCULO VII.1 – Transmisión telegráfica. Recomendaciones de la serie R (Comisión de Estudio IX). Equipos terminales para los servicios de telegrafía. Recomendaciones de la serie S (Comisión de Estudio IX).

FASCÍCULO VII.2 – Conmutación telegráfica. Recomendaciones de la serie U (Comisión de Estudio IX).

FASCÍCULO VII.3 – Equipos terminales y protocolos para los servicios de telemática. Recomendaciones de la serie T (Comisión de Estudio VIII).

Tomo VIII – *(Siete fascículos, vendidos por separado.)*

FASCÍCULO VIII.1 – Comunicación de datos por la red telefónica. Recomendaciones de la serie V (Comisión de Estudio XVII).

FASCÍCULO VIII.2 – Redes de comunicación de datos: servicios y facilidades. Recomendaciones X.1 a X.15 (Comisión de Estudio VII).

FASCÍCULO VIII.3 – Redes de comunicación de datos: interfaces. Recomendaciones X.20 a X.32 (Comisión de Estudio VII).

FASCÍCULO VIII.4 – Redes de comunicación de datos: transmisión, señalización y conmutación, aspectos de redes, mantenimiento, disposiciones administrativas. Recomendaciones X.40 a X.181 (Comisión de Estudio VII).

FASCÍCULO VIII.5 – Redes de comunicación de datos: interconexión de sistemas abiertos (ISA), técnicas de descripción de sistemas. Recomendaciones X.200 a X.250 (Comisión de Estudio VII).

FASCÍCULO VIII.6 – Redes de comunicación de datos: interfuncionamiento entre redes, sistemas móviles de transmisión de datos. Recomendaciones X.300 a X.353 (Comisión de Estudio VII).

FASCÍCULO VIII.7 – Redes de comunicación de datos; sistemas de tratamiento de mensajes. Recomendaciones X.400 a X.430 (Comisión de Estudio VII).

Tomo IX – Protección contra las perturbaciones. Recomendaciones de la serie K (Comisión de Estudio V) – Construcción, instalación y protección de los cables y otros elementos de planta exterior. Recomendaciones de la serie L (Comisión de Estudio VI).

Tomo X – *(Dos fascículos, vendidos por separado.)*

FASCÍCULO X.1 – Términos y Definiciones.

FASCÍCULO X.2 – Índice del Libro Rojo.

PAGE INTENTIONALLY LEFT BLANK

PAGE LAISSEE EN BLANC INTENTIONNELLEMENT

ÍNDICE DEL FASCÍCULO VI.11 DEL LIBRO ROJO

Anexos a las Recomendaciones Z.100 a Z.104

Lenguaje de especificación y descripción funcionales (LED)

NOTAS PRELIMINARES

1 Las Cuestiones asignadas a cada Comisión de Estudio para el periodo de estudios 1985-1988 figuran en la contribución N.º 1 de dicha Comisión.

2 En este fascículo, la expresión «Administración» se utiliza para designar, en forma abreviada, tanto una Administración de telecomunicaciones como una empresa privada de explotación de telecomunicaciones reconocida.

Anexos a las Recomendaciones Z.100 a Z.104

LENGUAJE DE ESPECIFICACIÓN Y DESCRIPCIÓN FUNCIONALES (LED)

PAGE INTENTIONALLY LEFT BLANK

PAGE LAISSEE EN BLANC INTENTIONNELLEMENT

ANEXO A

(a las Recomendaciones Z.100 a Z.104)

Glosario LED

Todo término que figure en las Recomendaciones Z.100 a Z.104 relativas al LED en cursiva y que aparezca en este Glosario se utiliza estrictamente en el sentido definido en el mismo.

Si una frase en cursiva, por ejemplo *identificador de procedimiento*, no figura en el Glosario, puede figurar como concatenación de dos términos, en este caso, el término *identificador* seguido del término *de procedimiento*. Las palabras en cursiva que no figuren en el Glosario pueden ser palabras derivadas de un término del Glosario. Por ejemplo, *exportado* es el participio pasado de *exportar*.

Excepto cuando un término es sinónimo de otro, tras la definición del término hay una referencia principal al empleo del término en las Recomendaciones de la serie Z.100. Estas referencias se indican entre corchetes [ ] después de las definiciones. Por ejemplo, [Recomendación Z.100, § 3.2] indica que la referencia principal es el § 3.2 de la Recomendación Z.100. Dado que a veces los términos aparecen en contextos no definitorios, estas referencias son sólo una orientación.

El anexo C1 contiene la representación gráfica de los símbolos y diagramas, las palabras clave para la representación gráfica de los símbolos y diagramas, las palabras clave para la representación de expresiones textuales y el empleo de estas palabras clave, por lo que esta información no se repite en el Glosario.

tipo de datos abstractos

E: abstract data type

Un *tipo de datos abstractos* es un *tipo* definido por la relación algebraica entre valores del *tipo* y la aplicación de *operadores* en estos valores. [Recomendación Z.100, § 2.3; Z.104, § 1]

sintaxis abstracta

E: abstract syntax

La *sintaxis abstracta* del LED se da en las Recomendaciones Z.101, Z.102, Z.103 y Z.104, y se resume en el apéndice B para describir la estructura conceptual de una definición *LED* frente a las *sintaxis concretas* que existen para cada forma de *LED*, es decir, *LED/GR*, *LED/PR* y *LED/EP*. [Recomendación Z.100, § 3.1]

acceso

E: access

El *operador* predefinido implícito con todos los tipos de datos que se aplica siempre que una variable arroja el valor asociado con el mismo. [Recomendación Z.104, § 4.5]

operador activo

E: active operator

Un *operador* que requiere una o más *variables* como parámetros, dado que puede cambiar los *valores* asociados con estas *variables*. [Recomendación Z.104, § 4.5]

parámetro efectivo

E: actual parameter

Un *parámetro efectivo* es un *valor* entregado a un *proceso* o a un *procedimiento* para el *parámetro formal* correspondiente cuando el *proceso* (o *procedimiento*) es *creado* (o *llamado*, respectivamente). [Recomendación Z.101, § 2.3; Z.103, § 2.1]

lista de parámetros efectivos

E: actual parameter list

Una *lista de parámetros efectivos* es la lista de *valores* asociados con una *petición de crear* o una *llamada de procedimiento.* [Recomendación Z.101, § 2.3; Z.103, § 2.1]

conjunto de conservación adicional

E: additional - save - set

Un *conjunto de conservación adicional* es el conjunto de señales adicionales conservadas en un *procedimiento.* [Recomendación Z.103, § 2.1]

símbolo de atribución

E: allocation symbol

El *símbolo* en un *diagrama de árbol de procesos* del *LED/GR* asociado a un *símbolo de proceso.* El *proceso* o *subproceso* asociado con el *símbolo de proceso* está atribuido al *bloque* cuyo *nombre* está contenido en el *símbolo de atribución.* [Recomendación Z.102, § 2.3]

anotación

E: annotation

Una *anotación* en *LED/GR* es un *comentario.* Una *anotación* en *LED/PR* es un *comentario* o una *nota.* Las *anotaciones* no cambian el significado del LED. [Recomendación Z.101, § 3.3, 4.3]

arco

E: arc

Un *arco* es una conexión dirigida entre los *nodos* de un *gráfico de proceso.* [Recomendación Z.101, § 2.2]

Matriz

E: array

El *generador predefinido* utilizado para introducir el concepto de matrices. [Recomendación Z.104, § 5.3]

sentencia de asignación

E: assignment statement

Una *sentencia de asignación* es una *acción* que asocia un *valor* con una *variable.* [Recomendación Z.101, § 2.2; Recomendación Z.104, § 4.11]

axioma

E: axiom

Una *expresión booleana* que es cierta para todos los posibles valores de las variables utilizadas en el axioma. [Recomendación Z.104, § 4.6]

B'

La palabra clave del *LED/PR* que introduce una denotación binaria de un *valor* numérico. [Recomendación Z.104, § 4.8]

LED básico

E: basic SDL

El subconjunto mínimo del *LED* definido en la Recomendación Z.101.

comportamiento

E: behaviour

El *comportamiento* o *comportamiento funcional* de un sistema en *LED* se describe como respuestas discretas a estímulos discretos. [Recomendación Z.101, § 1]

bloque

E: block

Un *bloque* es sinónimo de *instancia de bloque.*

definición de bloque

E: block definition

Una *definición de bloque* define las propiedades estructurales y de conectividad de un tipo de *bloque* nombrado. [Recomendación Z.101, § 2.2]

diagrama de interacción de bloques

E: block interaction diagram

El *diagrama de interacción de bloques* del *LED/GR* representa la estructura de *bloques, canales, listas de señales, listas de datos, procesos* y *flujo de señales* entre *procesos.* Un *diagrama de interacción de bloques* puede también mostrar la partición de un sistema en *bloques, sub-bloques, canales* y *subcanales.* [Recomendación Z.101, § 3.1; Recomendación Z.102, § 2]

subestructura de bloque

E: block substructure

Una *subestructura de bloques* para un *bloque* es la *partición* del *bloque* en *sub-bloques* y nuevos *canales* en *niveles de abstracción* inferiores. [Recomendación Z.102, § 2.2]

definición de subestructura de bloque

E: block substructure definition

Una *definición de subestructura de bloque* forma parte facultativamente de una *definición de bloque* y define una *subestructura de bloque.* [Recomendación Z.102, § 2.2]

símbolo de bloque

E: block symbol

El *símbolo de bloque* representa el concepto de un *bloque funcional* en un *diagrama de interacción de bloques* del *LED/GR.* [Recomendación Z.101, § 3.1]

diagrama de árbol de bloques

E: block tree diagram

Un *diagrama de árbol de bloques* es la representación *LED/GR* de la partición de un *sistema* en *bloques* en *niveles de abstracción inferiores* por medio de un diagrama de árbol invertido. [Recomendación Z.102, § 3.1]

booleano

E: boolean

El *tipo de datos* lógico que tiene los valores VERDADERO y FALSO, y el conjunto normal de *operadores* lógicos. [Recomendación Z.104, § 5.4]

nodo de llamada

E: call node

El término *nodo de llamada* es sinónimo de *nodo de llamada de procedimiento.*

símbolo de llamada

E: call symbol

El término *símbolo de llamada* es sinónimo de *símbolo de llamada de procedimiento.*

canal

E: channel

Un *canal* es la clase de entidad que transporta *señales* de un *bloque* a otro *bloque.* Los *canales* transportan también *señales* hacia el *entorno* y a partir de éste. Los *canales* son unidireccionales, es decir, sólo transportan señales en una dirección. [Recomendación Z.101, § 2.1]

definición de canal

E: channel definition

Una *definición de canal* define las propiedades de un *canal* nombrado. Estas propiedades son el *bloque* de origen, el *bloque* de destino, el conjunto de *señales* que el *canal* puede transportar y, facultativamente, una *definición de subestructura de canal.* El *bloque* de destino puede ser el *entorno del sistema.* [Recomendación Z.101, § 2.2; Recomendación Z.102, § 2.2]

subestructura de canal

E: channel substructure

Una *subestructura de canal* es una *partición* de un *canal* en un conjunto de *canales* y de *bloques* en un *nivel de abstracción* inferior. [Recomendación Z.102, § 2.2]

definición de subestructura de canal

E: channel substructure definition

Una *definición de subestructura de canal* es una parte facultativa de una *definición de canal* que define la *subestructura de canal.* [Recomendación Z.102, § 2.2]

diagrama de subestructura de canal

E: channel substructure diagram

Un *diagrama de subestructura de canal* es la representación *LED/GR* de una *subestructura de canal.* [Recomendación Z.102, § 3.4]

símbolo de canal

E: channel symbol

El *símbolo* de un *diagrama de interacción de bloques* de *LED/GR* que representa un *canal* o *subcanal.* La flecha del símbolo apunta al *bloque* de destino y el otro extremo del símbolo identifica el *bloque* de origen. El conjunto de *señales* transportadas por el *canal* puede representarse mediante un *símbolo de lista de señales* asociado. [Recomendación Z.101, § 3.1]

carácter

E: character

El *tipo de carácter predefinido* para el que los *literales* son *literales de cadena de caracteres* de longitud 1, y los *operadores* son iguales, o desiguales, y la concatenación permite formar un *valor de cadena de caracteres.* [Recomendación Z.104, § 5.5]

elemento pictográfico de tasación en curso

E: charging in progress PE

Elemento pictográfico que indica que *tiene lugar* tasación en ese momento. [Recomendación Z.103, § 6.1]

cadena de caracteres

E: charstring

El *tipo de datos predefinido* para el que los *literales* son cadenas de caracteres del alfabeto N.º 5 del CCITT, y los *operadores* son los del *generador predefinido de cadena* previsto para la representación de caracteres. [Recomendación Z.104, § 5.13]

elemento pictográfico de emisor y receptor de señalización combinados

E: combined signalling sender and receiver PE

Elemento pictográfico correspondiente a un emisor de señalización y a un receptor de señalización combinados. [Recomendación Z.103, § 6.1].

comentario

E: comment

Información que completa o precisa el gráfico del *LED*. En *LED/GR* los *comentarios* pueden agregarse mediante un solo corchete conectado por una línea de trazo interrumpido a una *línea de flujo* de un *diagrama de proceso* o cualquier símbolo. En *LED/PR* los *comentarios* se introducen mediante la palabra clave *COMENTARIO*. [Recomendación Z.101, § 3.3]

operaciones compuestas

E: composite operation

Una notación abreviada que representa una combinación normalizada de conceptos *LED* primitivos. Una *operación compuesta* se puede mapear con conceptos de la *sintaxis abstracta* del LED de forma sistemática. [Recomendación Z.103, § 3]

forma sintáctica concreta

E: concrete syntactical form

El *LED/GR*, el *LED/PR* y el *LED/PE* son las *formas sintácticas concretas* de *LED* que se pueden mapear entre sí teniendo en cuenta los conceptos subyacentes en la representación matemática del gráfico del *LED*. [Recomendación Z.100, § 3.1]

sintaxis concreta

E: concrete syntax

La *sintaxis concreta* para las diversas representaciones del LED son los símbolos efectivos utilizados para representar el *LED* en las formas *LED/GR*, *LED/PR* o *LED/PE*. [Recomendación Z.100, § 3.1]

expresión condicional

E: conditional expression

Expresión de *forma booleana* que sigue a SI (IF), que controla si es interpretada la *expresión* de consecuencia que sigue a ENTONCES (THEN) o la *expresión* alternativa que sigue a SÍ NO. [Recomendación Z.104, § 4.10]

elemento pictográfico de trayecto de conmutación conectado

E: connected switching path PE

Elemento pictográfico que indica conectividad entre el equipo terminal y/o los dispositivos de señalización. [Recomendación Z.103, § 6.1]

conector

E: connector

Un *conector* es un *símbolo* utilizado en *LED/GR*. Un *conector* (un círculo) es o bien un *conector de entrada* o un *conector de salida*. Una *línea de flujo* puede ser interrumpida por un par de *conectores asociados*, cuyo flujo se supone va del *conector de salida* al *conector de entrada* asociado. [Recomendación Z.101, § 3.1]

valor constante

E: constant value

Se trata de un *literal* o de un *símbolo.* [Recomendación Z.104, § 2.2]

expresión constante

E: constant expression

Un *valor constante* o un *operador* con sólo *expresiones constantes* como parámetros. [Recomendación Z.104, § 2.2]

señal continua

E: continuous signal

Una *señal continua* es una *operación compuesta* que contiene una *condición.* Cuando la *condición* pasa a ser verdadera, se sale del *estado* que tiene asociada la *señal continua.* [Recomendación Z.103, § 3.3]

convergencia

E: convergence

En un *diagrama de proceso* de *LED/GR* cuando dos o más *símbolos* van seguidos de un solo *símbolo,* las *líneas de flujo* convergen en el símbolo. La convergencia puede manifestarse como una *línea de flujo* que confluye en otra, o como más de un *conector de salida* asociado con un solo *conector de entrada,* o como *líneas de flujo* separadas que entran en el mismo *símbolo.* [Recomendación Z.101, § 3.3]

acción de petición de crear

E: create request action

Una *acción* que causa la creación y arranque de una nueva *instancia de proceso* a partir de una *definición de proceso* especificada. [Recomendación Z.101, § 2.3]

nodo de petición de crear

E: create request node

Un *nodo* en una *transición* en un *gráfico de proceso* o un *gráfico de procedimiento* en que se produce una *acción de petición de crear.* [Recomendación Z.101, § 2.2]

símbolo de crear

E: create symbol

El *símbolo* es un *diagrama de interacción de bloques* del *LED/GR* que conecta el *símbolo de proceso* del *proceso creador* con el *símbolo de proceso* del *proceso creado.* La flecha identifica el *proceso descendiente* y el otro extremo del símbolo de petición de crear identifica el *proceso ascendiente.* [Recomendación Z.101, § 3.1]

símbolo de petición de crear

E: create request symbol

El *símbolo* en un *diagrama de proceso* del *LED/GR* que representa una *petición de crear.* [Recomendación Z.101, § 3.3]

D'

La palabra clave del *LED/PR* que introduce una denotación decimal de un *valor* numérico. Teniendo cuenta que se trata de una denotación supletoria, *D'* es facultativo. [Recomendación Z.104, § 4.8]

datos

E: data

El término *datos* es sinónimo de *ítem de datos.*

ítem (elemento) de datos

E: data item

Un *ítem de datos* es una *variable* o un *valor.*

tipo de datos

E: data type

Un *tipo de datos* de *ítems de datos* determina la *gama,* el significado de los *valores* de esa *gama* y el conjunto válido de *operadores* que pueden utilizarse con ítems de ese *tipo de datos.* (Véase asimismo *tipo de datos predefinido*). [Recomendación Z.104, § 1]

definición de tipo de datos

E: data type definition

Define el *nombre,* los *valores de datos* y los *operadores* para un *tipo de datos.* [Recomendación Z.104, § 2.2]

decisión

E: decision

Una *decisión* es una *acción* en un *nodo de decisión* dentro de una *transición,* en virtud de la cual se hace una pregunta, cuya respuesta puede obtenerse en ese instante y que determina la elección de uno o varios *arcos* de salida del *nodo* para continuar la *transición.* [Recomendación Z.101, § 2.3]

nodo de decisión

E: decision node

Un *nodo de decisión* es un *nodo* de una *transición* de un *gráfico de proceso* o *gráfico de procedimiento* en el que se tiene lugar una *decisión.* [Recomendación Z.101, § 2.2]

nombre de decisión

decision name

Un *nombre de decisión* es el *nombre* asociado con una *decisión.* El nombre de una *decisión* debe ser una pregunta que el interpretador comprende claramente. Los *nombres* de los *arcos de decisión* que salen de una *decisión* deben constituir todos los resultados posibles de la pregunta. [Recomendación Z.101, § 2.3]

símbolo de decisión

E: decision symbol

Un *símbolo* que representa el concepto LED de una decisión en un *diagrama de proceso* del *LED/GR.* [Recomendación Z.101, § 3.2]

declarar!

E: declare!

El *operador* predefinido implícito con todos los tipos de datos asociados con la instanciación de *instancias variables.* [Recomendación Z.104, § 4.5]

definición

E: definition

El término *definición* es sinónimo de *definición de tipo.*

descripción

E: description

La realización de los requisitos de un sistema se describe en una descripción del sistema. Las descripciones se componen de los *parámetros generales* del sistema tal como ha sido realizado y la *descripción funcional* (DF) de su comportamiento real. [Recomendación Z.100, § 1.1]

divergencia

E: divergence

En el *LED/GR* cuando siguen a un *símbolo* dos o más *símbolos*, una *línea de flujo* que conduce a dicho símbolo puede *divergir* y formar dos o más *líneas de flujo*. [Recomendación Z.101, § 3.3]

duración

E: duration

El *tipo de datos predefinido* que representa el intervalo entre dos instantes de tiempo. [Recomendación Z.104, § 5.11]

condición habilitante

E: enabling condition

Una *condición habilitante* es una *operación compuesta* para un método de aceptación condicional de *entrada* o, alternativamente, *conservación* de una *señal* dependiente de una *condición*. [Recomendación Z.103, § 3.2]

símbolo de condición habilitante

E: enabling condition symbol

El *símbolo* de un *diagrama de proceso* o *diagrama de procedimiento* del *LED/GR* que representa una *condición habilitante* (cuando sigue a un *símbolo de entrada*) o una *señal continua* (cuando sigue a un *símbolo de estado*). [Recomendación Z.103, § 3.2]

entorno

E: environment

El término *entorno* es sinónimo de *entorno de un sistema*.

símbolo de entorno

E: environment symbol

El *símbolo* de un *diagrama de proceso* del *LED/GR* que representa el *entorno de un sistema*. [Recomendación Z.101, § 3.1.1, 3.1.2]

entorno de un sistema

E: environment of a system

El *entorno de un sistema* es una parte del sistema cuyo comportamiento no se muestra en el LED, pero interactúa con el resto del sistema enviando *instancias de señal* al *sistema*. [Recomendación Z.101, § 2.1]

comportamiento equivalente

E: equivalent behaviour

El término *comportamiento equivalente* es sinónimo de *comportamiento funcional equivalente*.

comportamiento funcional equivalente

E: equivalent functional behaviour

Dos *sistemas* (o *bloques* o *procesos*) tienen *comportamiento funcional equivalente* si ofrecen la misma respuesta a una secuencia dada de *instancias de señal* vistas desde el exterior de los *sistemas* (o *bloques* o *procesos*, respectivamente). [Recomendación Z.100, § 1.1]

EXPORT

E: EXPORT

La construcción sintáctica EXPORT (nombre variable) se utiliza en *LED/GR* y en *LED/PR* para representar la *exportación* del *valor* de una *variable*. [Recomendación Z.103, § 3.1]

exportación

E: export

El término *exportación* es sinónimo de *operación de exportación.*

EXPORTADO

E: EXPORTED

La palabra clave del *LED/PR* que indica en una *definición de variable* que la *variable* se ha *exportado.* [Recomendación Z.103, § 3.1]

exportador

E: exporter

Un *exportador* de una *variable* es la *instancia de proceso* a la cual pertenece la *variable* cuyo *valor* se *exporta.* [Recomendación Z.103, § 3.1]

operación de exportación

E: export operation

Una *operación de exportación* es la *operación compuesta* que permite a un proceso *exportar* los *valores* de *ítems de datos* de forma que otro proceso pueda tener acceso a los *valores.* [Recomendación Z.103, § 3.1]

expresión

E: expression

Una *expresión* es un *nombre de valor,* un *nombre de sinónimo,* un *nombre de variable,* una *expresión condicional* o un *operador* aplicado a una o más *expresiones.* [Recomendación Z.104, § 4.7]

extracto

E: extract!

El *operador* que está implicado fuera de los *axiomas* cuando una *variable* va *inmediatamente seguida* por *expresiones* entre corchetes (excepto si va seguida por :=, en cuyo caso se *implica* insertar!). [Recomendación Z.104, § 4.5]

línea de flujo

E: flow line

Cada *símbolo* está conectado al *símbolo* (o *símbolos*) que le preceden en un *diagrama de proceso* del *LED/GR* por medio de una *línea de flujo.* [Recomendación Z.101, § 3.3]

parámetro formal

E: formal parameter

Un *parámetro formal* es un *nombre de variable* contenido en una *definición de proceso* o *definición de procedimiento* para el que se han asignado *parámetros efectivos* a las *variables* cuando se crea el *proceso* o se llama el *procedimiento,* respectivamente. [Recomendación Z.101, § 2.2; Recomendación Z.103, § 2.1]

lista de parámetros formales

E: formal parameter list

Una lista de *parámetros formales* en una *definición de proceso* o *definición de procedimiento.* Los *parámetros efectivos* corresponden a la posición que ocupan en sus respectivas listas. [Recomendación Z.101, § 2.2; Recomendación Z.103, § 2.1]

casilla (1)

E: frame (1)

Una *casilla* en un *diagrama de interacción de bloques* del *LED/GR* representa el *bloque* cuya *partición* se define por el *diagrama de interacción de bloques.* [Recomendación Z.102, § 3.2]

casilla (2)

E: frame (2)

Una *casilla* en un *diagrama de subestructura de canales* del *LED/GR* representa el *canal* cuya *partición* se define por el *diagrama de subestructura de canales.* [Recomendación Z.102, § 3.4]

comportamiento funcional

E: functional behaviour

(Véase *comportamiento.*)

bloque funcional

E: functional block

Un *bloque funcional* es un objeto de tamaño manejable y relación interna pertinente, que tiene un nombre, un conjunto de *canales* que lo conectan con otros *bloques* (o el *entorno*) y *procesos* internos o una *definición de bloque de parte interna.* [Recomendación Z.101, § 2.1; Recomendación Z.102, § 2.1]

parámetros generales

E: general parameters

Los *parámetros generales* en una *especificación* y en una *descripción* de un sistema se refieren a aspectos tales como límites de temperatura, construcción, capacidad de la central, grado de servicio, etc. [Recomendación Z.100, § 1.1]

generador

E: generator

El término *generador* es un sinónimo de *generador de tipo de datos.*

sintaxis gráfica

E: graphic syntax

(Véase *LED/GR.*)

H'

La palabra clave del *LED/PR* que introduce una denotación hexadecimal de un *valor* numérico. [Recomendación Z.104, § 4.8]

identificador

E: identifier

Un *identificador* es un *nombre* único para un *tipo* o una *instancia* de un *tipo,* y consiste en una *parte calificadora* y en una *parte nombre.* [Recomendación Z.100, § 2.1]

transición implícita

E: implicit transition

En un diagrama de procesos *LED/GR* si no hay *símbolos de entrada* explícitos ni *símbolos de conservación* explícitos asociados a un *símbolo de estado* para una de las *señales de entrada válidas,* se produce una *transición implícita* que consiste en un *nodo de entrada* conectado directamente hacia atrás con el mismo *estado.* Por consiguiente, estas *señales* se descartan. [Recomendación Z.101, § 3.3]

IMPORT

E: IMPORT

La construcción sintáctica IMPORT (nombre de variable, instancia de proceso) se utiliza tanto en *LED/GR* como en *LED/PR* para representar la *importación* del *valor* desde una *instancia de proceso.* [Recomendación Z.103, § 3.1]

importación

E: import

El término *importación* es sinónimo de *operación de importación.*

IMPORTADO

E: IMPORTED

La palabra clave del *LED/PR* que indica en una *definición de variable* que la *variable* se utiliza para contener *valores* que se han *importado.* [Recomendación Z.103, § 3.1]

importador

E: importer

Un *importador* de un *valor importado* es la *instancia de proceso* que *importa* el *valor.* [Recomendación Z.103, § 3.1]

operación de importación

E: import operation

Una *operación de importación* es la *operación compuesta* que permite a un *proceso importar* y tener acceso a los *valores* de las *variables* pertenecientes a otro *proceso.* [Recomendación Z.103, § 3.1]

valor importado

E: imported value

El *valor* visto por un *proceso* para un *ítem de datos* que se *importa.* [Recomendación Z.103, § 3.1]

DENTRO

E: IN

El atributo de *parámetro formal* que denota el caso en que un *valor* se pasa a un *procedimiento.* [Recomendación Z.103, § 2.3]

DENTRO/FUERA

E: IN/OUT

El atributo de *parámetro formal* que denota el caso en que se utiliza un *nombre de parámetro formal* como sinónimo para la *variable.* [Recomendación Z.103, § 2.3]

conector de entrada

E: in-connector

Una *línea de flujo* puede ser interrumpida por un par de *conectores asociados*; se supone que el flujo va del *conector de salida* al *conector de entrada* asociado. [Recomendación Z.101, § 3.3]

canal entrante

E: incoming channel

Un *canal entrante* es un nuevo *canal* formado cuando se *particiona* un *canal.* Un *canal entrante* transporta a los nuevos *bloques* formados por la *partición en canales* de todas las *señales* que el *canal particionado* transporta. [Recomendación Z.102, § 2.1]

operador de infijo

E: infix operator

Uno de los *operadores* diádicos predefinidos del *LED/PR* (=> O OEX Y DENTRO/= = > < <= >= + − // # / MOD REM) que se sitúan entre los dos parámetros y no antes de los parámetros entre corchetes, o uno de los *operadores* de prefijos monádicos (+ − NO). [Recomendación Z.104, § 4.5]

acceso de entrada

E: inlet

Un *acceso de entrada* de un *macro* es el punto por donde una línea entra en el *macro.* [Recomendación Z.103, § 4.1]

entrada

E: input

El término *entrada* es, por sí mismo, sinónimo de *acción de entrada.*

acción de entrada

E: input action

Una *acción de entrada* es una *acción* que recibe y consume una *señal de entrada* correspondiente a un *nombre de señal* y permite a la *instancia de proceso* que interprete la *acción de entrada* tener acceso a la información contenida en la *señal de entrada.* [Recomendación Z.101, § 2.3]

nodo de entrada

E: input node

Un *nodo de entrada* es un *nodo* de un *gráfico de proceso* o de un *gráfico de procedimiento* en el que tiene lugar una *acción de entrada* y tiene el mismo *nombre* que la *señal* que consume la *acción de entrada.* [Recomendación Z.101, § 2.2]

puerto de entrada

E: input port

El *puerto de entrada* de un *proceso* recibe y conserva las *señales* en el orden de llegada hasta que las *señales* son consumidas por una *acción de entrada.* [Recomendación Z.101, § 2.3]

señal de entrada

E: input signal

Una *señal de entrada* de un *proceso* es una señal del conjunto de *señales* nombradas que el *proceso* puede recibir en uno cualquiera de sus *nodos de estado.* El conjunto de *señales de entrada* válidas que un *proceso* puede recibir es el conjunto de todos los *nombres de señal* que aparecen en cualquier *nodo de entrada* del proceso. [Recomendación Z.101, § 2.3]

símbolo de entrada

E: input symbol

Símbolo de un *diagrama de proceso* del *LED/GR* que representa el concepto LED de una *entrada.* [Recomendación Z.101, § 3.3]

insertar!

E: insert!

El *operador* que se implica fuera de los *axiomas* cuando una *variable* va seguida inmediatamente por *expresiones* entre corchetes y luego : =. [Recomendación Z.104, § 4.5]

instancia

E: instance

Una *instancia* de un *tipo* es un valor que tiene todas las propiedades del *tipo* y puede distinguirse de otras *instancias* del mismo tipo por un *identificador.* [Recomendación Z.100, § 2.1]

instanciación

E: instantiation

Una *instanciación* es la creación de una *instancia* de una entidad a partir de un *tipo.* [Recomendación Z.100, § 2.1]

entero

E: integer

El *tipo entero* se define por el conjunto de valores −(infinito), ..., −2, −1, 0, +1, +2, ..., +(infinito) y sus operaciones matemáticas normales, es decir, suma, resta, multiplicación, división, etc. [Recomendación Z.104, § 3.5.1]

etiqueta

E: label

Una *etiqueta* es un *nombre* facultativo asociado a un *acceso de entrada* o *acceso de salida* a partir de un *macro.* [Recomendación Z.103, § 4.2]

nivel

E: level

El término *nivel* es sinónimo de *nivel de abstracción.*

nivel de abstracción

E: level of abstraction

Un *nivel de abstracción* es uno de los niveles de un *diagrama de árbol de bloques.* Una descripción de un *sistema* es un *bloque* en el *nivel de abstracción* más elevado y se muestra como un *bloque* único en la parte superior de un *diagrama de árbol de bloque.* [Recomendación Z.102, § 3.1]

literal

E: literal

Un *literal* denota una *identidad de valor.* Los *literales* 12, B'1100, 0'14 y H'C denotan todos la misma *identidad de valor de entero.* [Recomendación Z.104, § 4.4]

macro

E: macro

Un *macro* es una colección nombrada de elementos sintácticos definida por el usuario del LED, que sustituye la utilización del *nombre de macro* ANTES de considerar el significado de la representación LED. [Recomendación Z.103, § 4]

definición de macro

E: macro definition

Una *definición de macro* en *LED/GR* es una parte nombrada de cualquier diagrama *LED/GR* con *accesos de entrada* y *accesos de salida* representados por líneas que entran y salen (respectivamente) del diagrama de la parte. Estas líneas pueden estar *etiquetadas.* [Recomendación Z.103, § 4.2]

símbolo de macro

E: macro symbol

El *símbolo* utilizado en *LED/GR* para denotar una referencia a una *definición de macro* por nombre. [Recomendación Z.103, § 4.2]

nombre

E: name

El término *nombre* es sinónimo de la *parte nombre* de un *identificador.*

parte nombre

E: name part

La *parte nombre* de un *identificador* es una frase significativa en lenguaje natural que puede utilizarse en combinación con una *parte calificadora* del *identificador* para identificar un *tipo* o *instancia* de una entidad. [Recomendación Z.100, § 2.1]

natural

E: natural

El *tipo número natural* es un *tipo de gama* del *tipo entero* con los *valores* 0, 1, 2, ... hasta infinito. [Recomendación Z.104, § 5.6]

neotipo

E: newtype

Un *neotipo* introduce conjuntos de *literales* y *operadores* que son distintos de cualesquiera otros *literales* y *operadores* (aun cuando puedan tener los mismos nombres para que las diferentes identidades tengan que resolverse por cualificación). Las propiedades de un *neotipo* se definen por el uso de los *literales* y *operadores* en los *axiomas.* [Recomendación Z.104, § 4.3]

nodo

E: node

Un *nodo* es un lugar nombrado en un *gráfico de proceso* unido a otros *nodos* por *arcos.* Las clases de *nodos* en LED son *nodos de estado, nodos de entrada, nodos de tarea, nodos de salida, nodos de decisión, nodos de arranque, nodos de parada, nodos de arranque de procedimiento, nodos de llamada de procedimiento, nodos de retorno de procedimiento* y *nodos de petición de crear.* [Recomendación Z.101, § 2.2]

nota

E: note

Una *anotación* en *LED/PR* aplicable solamente a la representación *LED/PR.* Una *nota* es una cadena de texto encerrada entre /* y */. [Recomendación Z.101, § 4.3]

O'

La palabra clave del *LED/PR* que introduce una denotación octal de un *valor* numérico.

DESCENDIENTE (VÁSTAGO)

E: OFFSPRING

El *DESCENDIENTE* de un *proceso* de creación es un *ítem de datos* que tiene el mismo *valor* que el *ítem de datos MISMO* del *proceso* más recientemente creado por este *proceso* de creación. Si un *proceso* no ha creado ningún *proceso* su *ítem de datos DESCENDIENTE* es *indefinido.* [Recomendación Z.101, § 2.3]

operador

E: operator

Un *operador,* cuando se aplica a uno o más *valores* entrega un *valor* determinado por el uso del *operador* en los *axiomas.* Los símbolos + − * / son *operadores* aritméticos. [Recomendación Z.104, § 4.5]

operador tipificación

E: operator typing

Define los *tipos de datos* de los *ítems de datos* a los que el operador se aplica y el *tipo de datos* del *valor* resultante (si lo hubiere). [Recomendación Z.104, § 4.5]

opción

E: option

Una *opción* es una construcción de *sintaxis concreta* en una *definición de proceso* que permite escoger diferentes *comportamientos* ANTES de interpretar el *gráfico de proceso.* [Recomendación Z.103, § 5.1]

expresión de opción

E: option expression

Una *expresión* contenida en una *opción* que se evalúa para determinar qué *comportamiento* hay que escoger. [Recomendación Z.103, § 5.2]

símbolo de opción

E: option symbol

El *símbolo* en un *diagrama de proceso* o *diagrama de procedimiento* de *LED/GR* que representa una *opción.* [Recomendación Z.103, § 5.2]

conector de salida

E: out-connector

Una *línea de flujo* puede ser interrumpida por un par de *conectores asociados*; se supone que el flujo va del *conector de salida* al *conector de entrada* asociado. [Recomendación Z.101, § 3.3]

canal de salida

E: outgoing channel

Un *canal de salida* es un nuevo *canal* formado cuando un *canal* se *particiona.* Un *canal de salida* transporta desde los nuevos *bloques* formados por la *partición del canal,* todas las *señales* que transporta el *canal particionado.* [Recomendación Z.102, § 2.1]

acceso de salida

E: outlet

Un *acceso de salida* de un *macro* es el punto en que una línea abandona el *macro.* [Recomendación Z.103, § 4.2]

salida

E: outlet

El término *salida,* por sí mismo, es sinónimo de *acción de salida.*

acción de salida

E: output

Una *salida* es una *acción* dentro de una *transición,* que genera una *señal,* la cual, a su vez, actúa como *señal de entrada* en otro lugar. [Recomendación Z.101, § 2.3]

nodo de salida

E: output node

Un *nodo* en un *gráfico de proceso* en que se produce una *acción de salida* y tiene el mismo nombre que la *señal* que genera. [Recomendación Z.101, §2.2]

símbolo de salida

E: output symbol

El símbolo de un *diagrama de proceso* del *LED/GR* que representa el concepto LED de una *salida.* [Recomendación Z.101, § 3.3]

ASCENDIENTE

E: PARENT

El *ítem de datos ASCENDIENTE* de un *proceso* es el *ítem de datos MISMO* de su *proceso* ascendiente, es decir, el *proceso* que interpretó la *acción de petición de crear* que inició el *proceso.* [Recomendación Z.101, § 2.3]

partición

E: partitioning

Una *partición* es la elaboración del comportamiento de sistemas complejos y/o grandes en subsistemas a fin de proporcionar una partición lógica del comportamiento del sistema y puntos de vista abstractos diferentes del mismo sistema. [Recomendación Z.100, § 2.1]

operador pasivo

E: passive operator

Un *operador* que requiere sólo *valores* como parámetros, y produce como resultado un *valor*; un *operador pasivo* no puede cambiar los *valores* asociados con las *variables.* [Recomendación Z.104, § 4.5]

elemento pictográfico (EP)

E: pictoral elemento (PE)

Cada una de las entidades gráficas normalizadas utilizadas dentro de *pictogramas de estado* del *LED/PE* para representar conceptos relativos a los sistemas de conmutación. [Recomendación Z.103, § 6]

PId

E: PId

El *tipo de datos predefinido* utilizado para identificar *instancias de proceso.* [Recomendación Z.104, § 5.8]

tipo de datos predefinido

E: predefined data type

Por sencillez de descripción, el término *tipo de datos predefinido* se aplica a *nombres* predefinidos de *tipos de datos* y a *nombres* predefinidos de *generadores de tipos de datos. Booleano, Carácter, Cadena de caracteres, Duración, Entero, Natural, PId, Real, Tiempo* y *Temporizador* son *nombres de tipos de datos* predefinidos. *Matriz, Conjuntista* y *Cadena* son *nombres de generadores de tipos de datos* predefinidos. [Recomendación Z.104, § 5]

conjuntista

E: powerset

El *generador de tipos de datos predefinido* que genera *tipos de datos* con *valores* que son conjuntos ordenados matemáticos de *valores.* Cada *valor conjuntista* es un conjunto de *valores* de *tipo de datos* utilizado para parametrizar el *generador conjuntista.* [Recomendación Z.104, § 5.7]

procedimiento

E: procedure

Una sección de un *gráfico de proceso* que puede considerarse separadamente. Un *procedimiento* se define en un lugar, pero puede referirse a varios elementos, incluso en *procesos* diferentes. Las *señales* y *variables* afectadas por la interpretación de un *procedimiento* son controladas por el paso de un parámetro. [Recomendación Z.103, § 2]

llamada de procedimiento

E: procedure call

Una *llamada de procedimiento* es el medio para invocar un *procedimiento nombrado* para la interpretación y paso de parámetros al *procedimiento.* [Recomendación Z.103, § 2.1]

nodo de llamada de procedimiento

E: procedure call node

Un *nodo de llamada de procedimiento* es un *nodo* en un *gráfico de proceso* o *gráfico de procedimiento* en que se produce una *llamada de procedimiento.* [Recomendación Z.103, § 2.1]

símbolo de llamada de procedimiento

E: procedure call symbol

El *símbolo* del *LED/GR* que representa una *llamada de procedimiento.* [Recomendación Z.103, § 2.2]

definición de procedimiento

E: procedure definition

Una *definición de procedimiento* define una sección de un *gráfico de proceso.* La definición asocia el *gráfico de procedimiento* con un *nombre de procedimiento,* una *lista de parámetros formales,* un *conjunto de conservación adicional* y, facultativamente, otras *definiciones de procedimiento* y *definiciones de datos.* [Recomendación Z.103, § 2.1]

diagrama de procedimiento

E: procedure diagram

Un *diagrama de procedimiento* es la representación *LED/GR* de un *gráfico de procedimiento.* [Recomendación Z.103, § 2.2]

gráfico de procedimiento

E: procedure graph

Un *gráfico de procedimiento* es un gráfico conectado por *arcos* dirigidos para describir el *comportamiento* de un *procedimiento* y puede formar una sección de un *gráfico de proceso.* [Recomendación Z.103, § 2.1]

retorno de procedimiento

E: procedure return

(Véase *retorno.*)

nodo de arranque de procedimiento

E: procedure start node

El *nodo de arranque de procedimiento* es un *nodo* de un *gráfico de procedimiento* en que se inicia la interpretación del *procedimiento* mediante una llamada del *procedimiento.* [Recomendación Z.103, § 2.1]

símbolo de arranque de procedimiento

E: procedure start symbol

El símbolo en un *diagrama de procedimiento* del *LED/GR* que representa un *nodo de arranque de procedimiento.* [Recomendación Z.103, § 2.2]

proceso

E: process

Un *proceso* ejecuta una función para la cual se necesitan varios ítems de información para realizar subfunciones. Las subfunciones realizadas dependen del orden temporal en que la información se pone a disposición del *proceso.* En el contexto del LED, un proceso es una clase de entidad, cuyas instancias son un «estado» en espera de una *entrada,* o en una *transición.* El término *proceso,* por sí mismo, es sinónimo de *instancia de proceso.* [Recomendación Z.101, § 2.1]

definición de proceso

E: process definition

El comportamiento de un *tipo* de la clase *proceso* se describe en una *definición de proceso,* en términos de gráfico directo cerrado de *entradas, conservaciones, estados, transiciones, decisiones, tareas* y *salidas.* [Recomendación Z.101, § 2.1]

diagrama de proceso

E: process diagram

Un *diagrama de proceso* es la representación *LED/GR* de un *gráfico de proceso.* [Recomendación Z.101, § 2.2]

gráfico de proceso

E: process graph

Un *gráfico de proceso* es un gráfico conectado por *arcos* dirigidos para describir el comportamiento de un *proceso.* [Recomendación Z.101, § 2.2]

instancia de proceso

E: process instance

Una *instancia de proceso* es una *instancia* de un *proceso* creada dinámicamente. [Recomendación Z.101, § 2.3]

subestructura de proceso

E: process substructure

La *subestructura de proceso* de un *proceso* es la *partición* del *proceso* en *subprocesos* y la atribución de estos *procesos* a *sub-bloques.* [Recomendación Z.102, § 2.2]

definición de subestructura de proceso

E: process substructure definition

Una *definición de subestructura de proceso* es una parte facultativa de una *definición de proceso* que define la *subestructura de proceso* de un *proceso.* [Recomendación Z.102, § 2.2]

símbolo de proceso

E: process symbol

El *símbolo* en un *diagrama de interacción de bloques* o *diagrama de árbol de proceso* del *LED/GR* que representa cero o más *instancias de proceso.* El *símbolo de proceso* contiene el *nombre de proceso,* que identifica la *definición de proceso,* y una *lista de parámetros formales.* [Recomendación Z.101, § 3.1; Recomendación Z.102, § 3.3]

diagrama de árbol de proceso

E: process tree diagram

Un *diagrama de árbol de proceso* del *LED/GR* representa la *partición* de un *proceso* de un *bloque* en *subproceso.* La atribución de los *subprocesos* a *subbloques* se muestra mediante *símbolos de atribución* en el *diagrama de árbol de procesos.* El diagrama es en forma de árbol invertido. [Recomendación Z.102, § 3.3]

calificadores

E: qualifiers

El término *calificadores* es sinónimo de la *parte calificadora* de un *identificador.*

parte calificadora

E: qualifying part

La *parte calificadora* de un *identificador* es la información que ha de añadirse a la *parte nombre* del *identificador* para formar un *nombre* único. La *parte calificadora* de un *identificador* puede derivarse del contexto de utilización de la *parte nombre.* [Recomendación Z.100, § 2.1]

real

E: real

El *tipo real* se define por el conjunto de TODOS los valores entre −(infinito) y +(infinito) y las operaciones matemáticas normales, a saber, suma, resta, multiplicación, elevación a potencias, etc. [Recomendación Z.104, § 5.2]

elemento pictográfico (EP) de trayecto de conmutación reservado

E: reserved switching path PE

Un *elemento pictográfico* que representa una conexión reservada entre equipos terminales y/o dispositivos de señalización. [Recomendación Z.103, § 6.1]

reponer («Reset»)

E: RESET

El *operador* definido para el *tipo de datos de temporizador* que permite a los temporizadores liberarse. [Recomendación Z.104, § 5.12]

señal retenida

E: retained signal

Cuando una *señal* llega a un *proceso* se considera que queda *recibida* y *retenida* para ese *proceso* (la retención se opera fuera del *proceso*; por tanto, una *señal retenida* no ha sido aún *consumida* por el *proceso*). [Recomendación Z.101, § 2.1]

retorno

E: return

El *retorno* de un *procedimiento* es la destrucción de las *variables* y los *sinónimos* creados en el *arranque de procedimiento*, seguido por la realización de la interpretación del *arranque de procedimiento*. [Recomendación Z.103, § 2.1]

nodo de retorno

E: return node

Un *nodo de retorno* es el *nodo* en un *gráfico de procedimiento* en que se produce el *retorno* desde el *procedimiento*. [Recomendación Z.103, § 2.1]

símbolo de retorno

E: return symbol

El *símbolo* de un *diagrama de procedimiento* que representa el *retorno* desde el *procedimiento*. [Recomendación Z.103, § 2.2]

atributo revelar

E: reveal attribute

Una *variable* perteneciente a un *proceso* puede tener un *atributo revelar*, en cuyo caso otro *proceso* del mismo *bloque* puede *ver* el *valor asociado* con la *variable*. [Recomendación Z.101, § 2.3]

conservación

E: save

Una *conservación* es la posposición del *reconocimiento de una señal* cuando un *proceso* se encuentra en un *estado* particular en que no se permite la *entrada* de dicha *señal*. [Recomendación Z.100, § 2.2]

conjunto de señales de conservación

E: save-signal-set

El *conjunto de señales de conservación* de un *estado* de un *proceso* es el conjunto de *nombres de señal conservados* para dicho *estado*. [Recomendación Z.100, § 2.2]

símbolo de conservación

E: save symbol

Un *símbolo* de un *diagrama de proceso* del *LED/GR* que representa el concepto LED de una conservación. [Recomendación Z.101, § 3.3]

LED/GR

E: SDL/GR

La forma gráfica del LED. [Recomendación Z.100, § 3.1]

LED/PR

E: SDL/PR

La representación del LED en frases de texto. [Recomendación Z.100, § 3.1]

LED/EP

E: SDL/PE

La forma a modo de elementos pictográficos del LED, que constituye una ampliación del *LED/GR* basada en estados. [Recomendación Z.100, § 3.5]

MISMO («SELF»)

E: SELF

El *ítem de datos MISMO* de un *proceso* es el único valor de *instancia de proceso* que lo distingue de cualquier otra *instancia de proceso.* [Recomendación Z.101, § 2.3]

EMISOR

E: SENDER

El *ítem de datos EMISOR* de un *proceso* es igual al ítem de datos de proceso de origen de la *señal* más recientemente consumida. [Recomendación Z.101, § 2.3]

poner («Set»)

E: SET

El *operador* definido para el *tipo de datos de temporizador* que permite poner los temporizadores. [Recomendación Z.104, § 5.12]

valor compartido

E: shared value

Un *valor compartido* es el *valor* asociado con una *variable* que es *revelado* por un *proceso* y *visto* por otro. [Recomendación Z.101, § 2.3]

SEÑAL

E: SIGNAL

El atributo de *parámetro formal* para un parámetro *señal* de un *procedimiento.* [Recomendación Z.103, § 2.3]

señal

E: signal

El término *señal* es, por sí mismo, sinónimo de *instancia de señal.*

definición de señal

E: signal definition

Una *definición de señal* define un *nombre* en forma de *nombre de señal* y asocia una lista de cero o varios *tipos de datos* con el *nombre de señal.* [Recomendación Z.101, § 2.2]

instancia de señal

E: signal instance

Una *instancia de señal* es una *instancia* de una *señal* que comunica información a una *instancia de proceso* a partir de una *acción de salida* de otra *instancia de proceso* o del *entorno.* Alternativamente, una *instancia* de señal se utiliza para comunicar información desde la *acción de salida* de un *proceso* al *entorno.* [Recomendación Z.101, § 2.3]

elemento pictográfico de receptor de señalización

E: signalling receiver PE

Un *elemento pictográfico* que representa un receptor de señalización. [Recomendación Z.103, § 6.1]

elemento pictográfico de emisor de señalización

E: signalling sender PE

Un *elemento pictográfico* que representa un emisor de señalización. [Recomendación Z.103, § 6.1]

lista de señales

E: signal lista

La lista de *nombres* de todas las *señales* que pueden ser transportadas por un *canal* o, por medios internos, en un *bloque* de un *proceso* a otro. [Recomendación Z.101, § 2.2]

símbolo de lista de señales

E: signal list symbol

El *símbolo* de un *diagrama de interacción de bloques* del *LED/GR* que representa una *lista de señales* asociada con un *canal* o con un *símbolo de ruta de señal.* [Recomendación Z.101, § 3.1]

símbolo de ruta de señal

E: signal route symbol

El *símbolo* de un *diagrama de interacción* del *LED/GR* que indica el flujo de *señales* entre un *proceso* y otro *proceso* en el mismo *bloque* o los *canales* conectados al *bloque.* [Recomendación Z.101, § 3.1]

especificación

E: specification

Los requisitos de un sistema se definen en una *especificación* de ese sistema. Una *especificación* comprende *parámetros generales* requeridos del sistema y la *especificación funcional* (EF) del comportamiento deseado. [Recomendación Z.100, § 1.1]

lenguaje de especificación y descripción (LED)

E: specification and description language (SLD)

El lenguaje CCITT utilizado en la presentación de la *especificación funcional* y la *descripción funcional* de los procesos lógicos internos en sistemas de conmutación con control por programa almacenado (CPA). [Recomendación Z.100, § 1.1]

acción de arranque

E: start action

La *acción de arranque* de un *proceso* se interpreta antes de cualquier otra *acción.* La *acción de arranque* inicializa los *parámetros formales* del proceso. [Recomendación Z.101, § 2.3]

nodo de arranque

E: start node

El *nodo de arranque* en un *gráfico de proceso* es el único nodo que no sigue a otro nodo. Hay un solo *nodo de arranque* en un *gráfico de proceso* y la interpretación de un *proceso* se inicia en este nodo. El *nodo de arranque* es el lugar en que se produce una *acción de entrada.* [Recomendación Z.101, § 2.2]

símbolo de arranque

E: start symbol

El *símbolo* de un *diagrama de proceso* del *LED/GR* que representa un *nodo de arranque.* [Recomendación Z.101, § 3.3]

estado

E: state

Un *estado* es una condición en la cual la acción de un *proceso* está en *suspenso* en espera de una *señal de entrada.* [Recomendación Z.101, § 2.1]

nodo de estado

E: state node

Un *nodo* de un *gráfico de proceso* o *gráfico de procedimiento* en que el *proceso* pasa a un *estado.* [Recomendación Z.101, § 2.2]

pictograma de estado

E: state picture

Un *pictograma de estado* es un *símbolo de estado* que comprende *elementos pictográficos* utilizados para ampliar el *LED/GR* al *LED/EP.* [Recomendación Z.103, § 6]

símbolo de estado

E: state symbol

Un *símbolo* de un *diagrama de proceso* del *LED/GR* que representa el concepto LED de uno o varios estados. [Recomendación Z.101, § 3.3]

parada

E: stop

Una *acción* que termina una *instancia de proceso.* [Recomendación Z.101, § 2.3]

nodo de parada

E: stop node

Un *nodo* de un *gráfico de proceso* en el que se produce la *parada.* [Recomendación Z.101, § 2.2]

símbolo de parada

E: stop symbol

El *símbolo* de un *diagrama de proceso* del *LED/GR* que representa una *parada.* [Recomendación Z.101, § 3.1]

cadena

E: string

El *generador de tipos de datos predefinido* que genera *tipos de datos* con *valores* que son listas de ítems del *tipo datos* utilizado para parametrizar el *generador cadena.* [Recomendación Z.104, § 5.9]

struct

E: struct

Una *definición de tipo de datos* con *struct* introduce implícitamente *tipos de datos* para nombres de campos y *axiomas* implícitos que definen el uso de nombres de campos con *extraer!* e *insertar!* para valores parciales de estructuras. [Recomendación Z.104, § 4.3]

sub-bloque

E: sub-block

Un *sub-bloque* es un *bloque* contenido dentro de otro *bloque.* Se forman *sub-bloques* cuando se efectúa la *partición* de un *bloque.* [Recomendación Z.100, § 2.1]

definición de sub-bloque

E: sub-block definition

Una *definición de sub-bloque* define un *bloque* y forma parte de una *definición de subestructura de bloque.* [Recomendación Z.102, § 2.2]

símbolo de sub-bloque

E: sub-block symbol

El *símbolo* de un *diagrama de interacción de bloques* o *diagrama de árbol de bloques* del *LED/GR* que representa un *sub-bloque* y es idéntico a un *símbolo de bloque.* [Recomendación Z.102, § 3.2]

subcanal

E: sub-channel

Un *subcanal* es un *canal* formado cuando un *bloque* es objeto de *partición.* [Recomendación Z.102, § 2.1]

subproceso

E: sub-process

Un *subproceso* es un *proceso* formado cuando un *proceso* es objeto de *partición.* [Recomendación Z.102, § 2.3]

elemento pictográfico de línea de abonado

E: subscriber line PE

Elemento pictográfico que representa una línea de abonado. [Recomendación Z.103, § 6.1]

elemento pictográfico de cuadro de conmutación

E: switchboard PE

Un *elemento pictográfico* que representa un cuadro de conmutación de equipo terminal. [Recomendación Z.103, § 6.1]

elemento pictográfico de módulo de conmutación

E: switching module PE

Un *elemento pictográfico* que representa un módulo de conmutación asociado con un trayecto de conmutación conectado o reservado. [Recomendación Z.103, § 6.1]

definición de sinónimo

E: synonym definition

Una *definición* de un *nombre* para un *valor de datos.* [Recomendación Z.104, § 4.12]

diagrama de sintaxis

E: syntax diagram

Los *diagramas de sintaxis* son diagramas utilizados para definir la sintaxis concreta del *LED/PR.* [Recomendación Z.100, § 3.4]

sintipo

E: syntype

Un *sintipo* introduce un conjunto de *valores* que corresponde a un subconjunto de los *valores* del *neotipo* ascendiente. *Acceder!, declarar!* y *asignar!* son los únicos operadores para *sintipos,* dado que los *valores* anteriores a la asignación y posteriores a la extracción a partir de *variables sintipo* son siempre *valores* del *neotipo* ascendiente. [Recomendación Z.104, § 4.3]

sistema

E: system

Un *sistema* es un conjunto de *bloques* conectados entre sí y al *entorno* por *canales.* El término *sistema,* por sí mismo, es sinónimo de *instancia de sistema.* [Recomendación Z.101, § 2.1]

frontera de sistema

E: system boundary

La *frontera de sistema* es el límite entre los *bloques* definidos en términos de LED y el *entorno.* [Recomendación Z.101, § 2.3]

definición de sistema

E: system definition

Una *definición de sistema* define las propiedades de un *sistema.* Las propiedades de un *sistema* son los *bloques, canales, señales* asociadas con los *canales, tipos de datos,* y *sinónimos* del *sistema.* [Recomendación Z.101, § 2.2]

tarea

E: task

Una *tarea* es una acción dentro de una *transición* que contiene o bien una secuencia de *sentencias de asignación, sentencias de poner* o *sentencias de reponer,* o bien texto informal. La interpretación de una *tarea* depende de, y puede actuar, sobre la información mantenida por el sistema. [Recomendación Z.101, § 2.2]

nodo de tarea

E: task node

Un *nodo* de un *gráfico de proceso* o *gráfico de procedimiento* en el que se produce una *tarea.* [Recomendación Z.101, § 2.2]

símbolo de tarea

E: task symbol

Un *símbolo* de un *diagrama de proceso* del *LED/GR* que representa el concepto LED de una tarea. [Recomendación Z.101, § 3.3]

elemento pictográfico de equipo terminal

E: terminal equipment PE

Uno de los seis *elementos pictográficos* posibles que representan los siguientes tipos de equipo terminal: teléfono colgado, teléfono descolgado, línea de enlace, línea de abonado, cuadro conmutador u otro. [Recomendación Z.103, § 6.1]

símbolo de ampliación de texto

E: test extension symbol

El texto asociado con este *símbolo* se considera que pertenece al símbolo del *LED/GR* al que se adjunta el símbolo de ampliación de texto. [Recomendación Z.101, § 3]

tiempo

E: time

El *tipo de datos predefinido* que representa el tiempo absoluto. [Recomendación Z.104, § 5.10]

temporizador

E: timer

El *tipo de datos predefinido* utilizado para temporizadores y que define los *operadores PONER* y *REPONER* para los temporizadores. [Recomendación Z.104, § 5.12]

elemento pictográfico de supervisión de tiempo de un proceso

E: time supervision of a process PE

Un *elemento pictográfico* que representa la marcha de un temporizador de supervisión. [Recomendación Z.103, § 6.1]

transición

E: transition

Una *transición* es una secuencia de *acciones* que se producen cuando una *instancia de proceso* pasa de un *estado* a otro en respuesta a una *entrada.* [Recomendación Z.101, § 2.2]

cadena de transición

E: transition string

Una *cadena de transición* es una secuencia de cero o más *acciones* entre un *nodo de entrada* y el siguiente *nodo de estado, nodo de parada* o *nodo de retorno de procedimiento.* [Recomendación Z.101, § 2.2]

elemento pictográfico de línea troncal

E: trunk PE

Un *elemento pictográfico* que representa un interfaz de línea troncal. [Recomendación Z.103, § 6.1]

tipo

E: type

Un *tipo* es un conjunto de propiedades para entidades. Las clases de *tipos* en LED son *bloques, canales, ítems de datos, procedimientos, procesos, señales* y *sistemas.* Un *tipo* puede componerse a base de entidades de la misma clase, en cuyo caso estas entidades son subtipos (por ejemplo, un *bloque* compuesto de *sub-bloques.* [Recomendación Z.100, § 2.1]

definición de tipo

E: type definition

Una *definición de tipo* define las propiedades de un *tipo.* [Recomendación Z.100, § 2.1]

señal de entrada válida

E: valid input signal

En cada *estado* de un *proceso* hay un conjunto de *nombres de señal* para *señales* que pueden además ser *entradas.* Una *señal de entrada válida* es un *nombre de señal* que es miembro de uno cualquiera de estos conjuntos. [Recomendación Z.101, § 2.3]

valor

E: value

Un *valor* de datos de un *tipo de datos* es uno de los *valores* que se asocian con una *variable* de este *tipo de datos,* y que pueden utilizarse con un *operador* que requiere un *valor* de ese *tipo de datos.* [Recomendación Z.104, § 1]

variable

E: variable

Una *variable* es una entidad que pertenece a un *proceso* y a la que se le puede asignar un *valor.* Una *variable* produce el último valor que se le ha asignado al tener acceso a la misma. [Recomendación Z.101, § 2.2]

definición de variable

E: variable definition

Una *definición de variable* define una *instancia* de un *tipo de datos.* [Recomendación Z.101, § 2.2]

visión

E: view

Una *variable* puede ser *vista* si el *valor* asociado con la *variable* es *revelado* por el *proceso* a que pertenece la *variable* y otro *proceso* puede tener acceso al *valor.* [Recomendación Z.101, § 2.3]

definición de visión

E: view definition

Una *definición de visión* es una parte de una *definición de proceso* que define las *variables* que pertenecen a otro *proceso* y son sólo *vistas* por el proceso que contiene la *definición de visión.* [Recomendación Z.101, § 2.3]

expresión de visión

E: viewing expression

Una *expresión de visión* se utiliza dentro de una *expresión* para indicar que se obtiene el último *valor* de una *variable vista.* [Recomendación Z.101, § 2.3]

ANEXO B

(a las Recomendaciones Z.100 a Z.104)

Resumen de la sintaxis abstracta

Este resumen contiene toda la sintaxis abstracta y las reglas de formación correcta asociadas que se definen en las Recomendaciones Z.101, Z.102, Z.103 y Z.104. La sintaxis se explica utilizando una forma FBN [1] ampliada que se define en la Recomendación Z.200 y las reglas se dan en inglés. Como se trata de un resumen, para las definiciones exactas deben consultarse las Recomendaciones.

B.1 *Sistema*

(1) <Definición de sistema> ::=

(Z.101) Nombre de *sistema*

(Z.101) [<Definición de bloque>]+

(Z.101) [<Definición de canal>]*

(Z.101) [<Definición de señal>]*

(Z.104) [<Definición de datos>]*

(Z.103) [<Definición de procedimiento>]*

Formación correcta

La definición de estructura debe contener una definición de señal para cada nombre de las listas de señales en cada una de las definiciones de canales que también se contienen. [Recomendación Z.101]

---

[1] FBN = Forma Backus-Naur.

B.2 *Bloque*

(2) <Definición de bloque> ::=

(Z.101) Nombre de *bloque*

(Z.101) [<Definición de proceso>]*

(Z.101) [<Definición de señal>]*

(Z.104) [<Definición de datos>]*

(Z.103) [<Definición de procedimiento>]*

(Z.102) [<Definición de subestructura de bloque> !]

Formación correcta

Para cada uno de los nombres de señal asociados con las operaciones de entrada y salida de los procesos contenidos, el bloque contiene una definición de señal o es una definición de señal contenida en la estructura circundante.

Si una definición de bloque contiene una definición de subestructura de bloque, no es necesario que contenga definiciones de proceso. [Recomendación Z.102]

B.3 *Subestructura de bloque*

(3) <Definición de subestructura de bloque> ::=

(Z.102) [<Definición de sub-bloque>]+

(Z.102) [<Definición de subcanal>]*

(Z.102) [<Definición de canal>]*

(Z.102) [<Definición de subestructura de proceso>]*

(Z.102) [<Definición de señal>]*

(Z.104) [<Definición de datos>]*

(Z.103) [<Definición de procedimiento>]*

(3.1) <Definición de sub-bloque> ::=

(Z.102) <Definición de bloque>

(3.2) <Definición de subcanal> ::=

(Z.102) <Definición de canal>

Formación correcta

Para cada una de las definiciones de proceso contenidas en el bloque circundante, la definición de subestructura de bloque debe contener una definición de subestructura de proceso.

Para cada uno de los puntos extremos de terminación de los canales del bloque circundante debe haber como mínimo una definición de subcanal que tenga ese punto extremo como punto extremo de origen y la inversa sucede con todos los puntos extremos de canal de origen de bloque circundante. La unión de las listas de señal de las definiciones de subcanal que tienen el mismo punto extremo que un canal que conduce o procede del bloque circundante, debe ser idéntica a la lista de señal de ese bloque; además de ello, estas listas de señal de las definiciones de canal que se originan en un punto extremo de terminación deben ser discontinuas. [Recomendación Z.102].

Una definición de canal contenida en la subestructura debe conectar sub-bloques entre sí. Todos los subcanales deben conectar puntos extremos de canales del bloque circundante a los sub-bloques.

B.4 *Canal*

(4) <Definición de canal> ::=

(Z.101) Nombre de *canal*

(Z.101) [Identificador de *bloque de origen*! ENVIRONMENT]

(Z.101) [Identificador de *bloque de destino*! ENVIRONMENT]

(Z.101) <Lista de señales>

(Z.102) [<Definición de subestructura de canal> !]

(4.1) <Lista de señales> ::=

(Z.101) [Nombre de *señal*]+

Formación correcta

Sólo uno de los identificadores de bloque de origen o bloque de destino puede ser sustituido por una referencia al entorno. [Recomendación Z.101]

El identificador de bloque de origen y el identificador de bloque de destino asociados con un canal han de ser distintos y cada uno ha de ser el identificador de un bloque del sistema o de un sub-bloque del bloque, o ha de ser el ENTORNO. [Recomendación Z.101]

B.5 *Subestructura de canal*

(5) <Definición de subestructura de canal> ::=

(Z.102) <Definición de canal entrante>

(Z.102) <Definición de canal saliente>

(Z.102) [<Definición de canal>]*

(Z.102) [<Definición de bloque>]+

(Z.102) [<Definición de señal>]*

(Z.104) [<Definición de datos>]*

(Z.103) [<Definición de procedimiento>]*

(5.1) <Definición de canal entrante> ::=

(Z.102) <Definición de canal>

(5.2) <Definición de canal saliente> ::=

(Z.102) <Definición de canal>

Formación correcta

La definición de canal de entrada tiene el mismo punto extremo de origen que la definición de canal circundante. El canal de salida tiene el mismo punto extremo de terminación que la definición de canal circundante. Las definiciones de canal de entrada y de salida tienen una lista de señal idéntica a la de la definición de canal circundante. [Recomendación Z.102].

B.6 *Señal*

(6) <Definición de señal> ::=

(Z.101) Nombre de *señal*

(Z.101) [Identificador de *tipo de datos*]*

Formación correcta

Los identificadores de tipo de datos utilizados han de ser predefinidos o nombres de definiciones de tipo de datos de las entidades estructurales circundantes.

B.7 *Proceso*

(7) <Definición de proceso> ::=

(Z.101) Nombre de *proceso*

(Z.101) <Número de instancias>

(Z.101) [<Parámetro formal>]

(Z.104) [<Definición de datos>]*

(Z.101) [<Definición de visión>]*

(Z.103) [<Definición de procedimiento>]*

(Z.101) <Gráfico de proceso>

(7.1) <Número de instancias> ::=
<*entero* identificador de *valor*> <*entero* identificador de *valor*>

(7.2) <Definición de visión> ::=
(Z.101) <Identificador de *variable*>
<Identificador de *tipo*>
<Identificador de definición de *proceso*>

Formación correcta

Todos los nombres de tipo mencionados han de ser predefinidos, definidos en la definición de proceso o definidos en los conceptos estructurales circundantes.

B.8 *Definición de subestructura de proceso*

(8) <Definición de subestructura de proceso> ::=
(Z.102) Nombre de *proceso*
(Z.102) [Nombre de *subproceso* Nombre de *sub-bloque*]*

Formación correcta

El nombre de proceso debe ser el nombre de una definición de proceso contenida en el bloque circundante. Todos los nombres de proceso contenidos deben ser nombres de subprocesos contenidos en el sub-bloque que tenga el nombre de sub-bloque asociado.

Cada nombre de señal del conjunto de señales de entrada válidas del proceso debe aparecer exactamente en uno de los conjuntos de señales de entrada válidas del subproceso. Cada nombre de señal agregado al nodo de salida del proceso debe ser agregado como mínimo a un nodo de salida de uno de los subprocesos.

B.9 *Gráfico de proceso*

(9) <Gráfico de proceso> ::=
(Z.101) <Nodo de arranque de *proceso*> <Transición de proceso>

(9.1) <Transición de proceso> ::=
<Cadena de transición> <Nodo de estado>
!<Cadena de transición> <Nodo de parada>

B.10 *Cadena de transición*

(10) <Cadena de transición> ::=
(Z.101) <Nodo de tarea> <Cadena de transición>
(Z.101) !<Nodo de salida> <Cadena de transición>
(Z.101) !<Nodo de decisión>
(Z.101) !<Nodo de petición de crear> <Cadena de transición>
(Z.103) !<Nodo de llamada de procedimiento> <Cadena de transición>
(Z.101) !

B.11 *Nodo de arranque*

(11) <Nodo de arranque> ::=

B.12 *Nodo de estado*

(12) <Nodo de estado> ::=
(Z.101) Nombre de *estado*
(Z.101) <Conjunto de señales de conservación>
(Z.101) [<Nodo de entrada>]+

(12.1) <Conjunto de conservación> ::=

(Z.101) [Identificador de *señal*]*

B.13 *Nodo de parada*

(13) <Nodo de parada> ::=

B.14 *Nodo de tarea*

(14) <Nodo de tarea> ::=

(Z.101) [[<Sentencia>]+ ![prueba informal>]*]

B.14.1 *Sentencia*

(14.1) <Sentencia> ::=

(Z.101) <Sentencia de poner>

(Z.101) !<Sentencia de reponer>

(Z.101) !<Sentencia de asignación>

(14.1.1) <Sentencia de> ::=

(Z.101) <Expresión de *tiempo*> identificador de *temporizador*

(14.1.2) <Sentencia de reponer> ::=

(Z.101) *Identificador de temporizador*

(14.1.3) <Sentencia de asignación> ::=

(Z.101) <Operador de *asignación*>

(Z.101) <Identificador de *variable*

(Z.101) <Expresión>

B.15 *Nodo de salida*

(15) <Nodo de salida> ::=

(Z.101) <Identificador de *señal*>

(Z.101) [<Expresión>]*

(Z.101) <Destino>

(15.1) <Destino> ::=

(Z.101) <Expresión de *PId*>

B.16 *Nodo de entrada*

(16) <Nodo de entrada>

(Z.101) Identificador de *señal de entrada* <Transición de proceso>

(Z.103) Identificador de *señal de entrada* <Transición de procedimiento>

Formación correcta

Una transición de proceso sólo puede aparecer en una definición de proceso, y una transición de procedimiento sólo puede aparecer en una definición de procedimiento.

B.17 *Nodo de decisión*

(17) <Nodo de decisión> ::=

(Z.101) <Pregunta>

(Z.101) <respuesta> [<respuesta>]+

(17.1) <Pregunta> ::=

(Z.101) <Expresión>

!<texto informal>

(17.2) <Respuesta> ::=

(Z.101) [<Identificador de *valor*>]+

[<transición de procedimiento> ! <transición de proceso>]

Formación correcta

Un nodo de decisión debe ir seguido de dos o más respuestas. [Recomendación Z.101]

Las expresiones del nombre de decisión y las respuestas han de ser del mismo tipo.

B.18 *Nodo de llamada de procedimiento*

(18) <Nodo de llamada de procedimiento> ::=

(Z.103) Identificador de *procedimiento*

(Z.103) [<Expresión>]*

(Z.103) [identificador de *señal*]*

(Z.103) <conjunto de conservación adicional>

(18.1) <conjunto de conservación adicional> ::=

[<Nombre de señal>]*

Formación correcta

Cada expresión ha de ser del mismo tipo que el parámetro formal correspondiente en la definición de procedimiento que tenga el nombre de procedimiento.

Ha de haber una expresión para cada parámetro formal de señal no tipo en la definición de procedimiento y un identificador de señal para cada parámetro formal de señal tipo.

B.19 *Nodo de petición de crear*

(19) <Nodo de petición de crear> ::=

(Z.101) Identificador de *proceso*

(Z.101) [<Expresión>]*

Formación correcta

Ha de haber una expresión para cada parámetro formal en la definición de proceso referenciada por el identificador de proceso, y cada expresión ha de concordar en tipo con el parámetro formal asociado. [Recomendación Z.101]

B.20 *Procedimiento*

(20) <Definición de procedimiento> ::=

(Z.103) Nombre de *procedimiento*

(Z.103) [<Definición de datos>]*

(Z.104) [<Definición de variable>]*

(Z.103) [<Definición de procedimiento>]

(Z.103) [<Parámetro formal>]

(Z.103) <Gráfico de procedimiento>

Formación correcta

Cada identificador de señal que aparece en el gráfico de procedimiento debe aparecer también como parámetro formal de señal tipo.

B.21 *Gráfico de procedimiento*

(21) <Gráfico de procedimiento> ::=

(Z.103) <Nodo de arranque de procedimiento> <Transición de procedimiento>

(21.1) <Transición de procedimiento> ::=

(Z.103) <Cadena de transición> <Nodo de estado>

(Z.103) <Cadena de transición> <Nodo de retorno>

B.22 *Nodo de arranque de procedimiento*

(22) <Nodo de arranque de procedimiento> ::=

B.23 *Nodo de retorno*

(23) <Nodo de retorno> ::=

B.24 *Definición de datos*

(24) <Definición de datos> ::=

(Z.104) <Definición de tipo de datos>

(Z.104) <Definición de sinónimo>

B.25 *Definición de variable*

(25) <Definición de variable> ::=

(Z.101) Nombre de *variable*

(Z.101) Identificador de *tipo*

(Z.101) [<Atributo revelar> !

B.26 *Identificador*

(26) <Identificador> ::=

(Z.100) <Calificador> Name

B.27 *Calificador*

(27) <Calificador> ::=

(Z.100) <Nombre estructural> <Nombre de tipo de entidad>

B.28 *Nombre estructural*

(28) <Nombre estructural> ::=

(Z.100) Nombre del *sistema*

(Z.100) [Nombre de *bloque*]* [Nombre de *canal*]*

(Z.100) [Nombre de *señal*]* [Nombre de *proceso*]*

(Z.100) [Nombre de *procedimiento*]* [Nombre de *tipo*]*

B.29 *Nombre de tipo de identidad*

(29) <Nombre de tipo de entidad> ::=

(Z.100) Sistema !Bloque !Proceso

(Z.100) !Procedimiento !Señal !Canal

(Z.100) !Tarea !Decisión !Arranque

(Z.100) !Parada !Petición de crear

(Z.100) !Retorno !Arranque de procedimiento

(Z.100) !Llamada !Variable !Tipo de datos

(Z.100) !Operador !Valor

B.30 *Definición de tipo de datos*

(30) <Definición de tipo de datos> ::=

(Z.104) Nombre de *tipo de datos* <Descripción de tipo de datos>

(Z.104) !Nombre de *tipo de datos* <Descripción de tipo syn>

B.31 *Descripción de tipo de datos*

(31) <Descripción de tipo de datos>

(Z.104) [nombre de *valor*]**

(Z.104) [<Introducción de operador>]+

(Z.104) [Axioma de tipo]*

Nota 1 – La notación [ ]** significa lista de cero o más y, posiblemente, de un número infinito de elementos. Fuera de la propiedad de que la lista puede ser infinita, es igual que [ ]*.

Nota 2 – Axioma de tipo no se define en la sintaxis abstracta. Sin embargo, hay una sintaxis concreta para axiomas.

Nota 3 – Un signo de más (+) indica que el grupo ha de estar presente y puede repetirse indefinidamente. Si los elementos sintácticos se agrupan entre corchetes, entonces el grupo es facultativo.

B.32 *Introducción de operador*

(32) <Introducción de operador> ::=

(Z.104) <Operador universal>

(Z.104) !Nombre de operador <Operador de tipificación>

B.33 *Operador universal*

(33) <Operador universal> ::=

(Z.104) <Operador de variable>

(Z.104) <Comparador>

B.34 *Operador de variable*

(34) <Operador de variable> ::=

(Z.104) Asignar

(Z.104) !Acceder

(Z.104) !Declarar

B.35 *Comparador*

(35) <Comparador> ::=

(Z.104) Menor que

(Z.104) !Mayor que

(Z.104) !Igual

B.36 *Operador de tipificación*

(36) <Operador de tipificación> ::=

(Z.104) <Identificador de *tipo de resultado*>

B.37 *Lista de tipos de datos*

(37) <Lista de tipos de datos> ::=

(Z.104) [<Identificador de *tipo de datos*>]+

Formación correcta (Para 30-37)

Todos los *nombres de valor* deben ser mutuamente excluyentes en una *descripción de tipo.*

Todos los *nombres de operador* deben ser mutuamente excluyentes en una *descripción de tipo.*

Para cada operador de tipificación, uno de los *identificadores de tipo de datos* de la *lista de tipos de datos* debe ser el *identificador de tipo de datos* del tipo que se define.

B.38 *Sintaxis abstracta de tipo syn*

(38) <Descripción de tipo syn> ::=

(Z.104) <Identificador de *tipo ascendiente*>

[<Identificador de *valor*>]*

Formación correcta

Todos los *identificadores de valor* deben ser miembros del conjunto de *identificadores de valor* del *identificador de tipo de datos* ascendiente.

B.39 *Texto informal*

(39) <texto informal> ::=

(Z.104) Nombre *bien comprendido*

B.40 *Expresión*

(40) <Expresión> ::=

(Z.104) <Primario>

(Z.104) !<Operación>

!<Expresión condicional;>

B.40.1 *Operación*

(40.1) <Operación> ::=

<Operador> [<Expresión>]*

B.40.2 *Expresión condicional*

(Z.104) <Expresión condicional> ::=

(Z.104) <Expresión *booleana*>

(Z.104) <Expresión> <Expresión>

B.41 *Primario*

(41) <Primario> ::=

(Z.104) Identificador de sinónimo

(Z.104) !Identificador de *valor*

(Z.104) !Identificador de *variable*

B.42 *Operador*

(42) <Operador> ::=

(Z.104) Identificador de *operador*

(Z.104) !<Operador universal> Identificador de *tipo*

B.43 *Definición de sinónimo*

(43) <Definición de sinónimo> ::=

(Z.104) Nombre de *sinónimo* <Expresión de constante>

B.44 *Expresión de constante*

(44) Expresión de constante::=

(Z.104) <Valor de constante>

(Z.104) <Operador> [<Expresión de constante>]+

B.45 *Valor de constante*

(45) <Valor de constante> ::=

(Z.104) Identificador de *valor*

(Z.104) !Identificador de *sinónimo*

ANEXO C1

(a las Recomendaciones Z.100 a Z.104)

Resumen del LED/GR

En LED/GR, un sistema comprende lo siguiente:

- diagrama de interacción de bloques (DIB);
- diagrama de subestructura de canales;
- árbol de bloques;
- árbol de procesos;
- diagrama de procesos;
- diagrama de ilustración general de estados.

Algunos de estos documentos son esenciales para proporcionar una especificación/descripción del sistema, en tanto que otros son documentos auxiliares que pueden facilitar la comprensión de la especificación/descripción.

C1.1 *Diagrama de interacción de bloques (DIB)*

C1.1.1 *Símbolos*

El DIB contiene un nombre de sistema, un conjunto de símbolos de bloque, símbolos de entorno, un conjunto de símbolos de canal y puede contener símbolos de proceso.

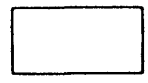
C1.1.1.1 *Línea de casilla*

Encierra el diagrama y representa el límite de los bloques particionados.

CCITT-74040

C1.1.1.2 *Símbolo de bloque*

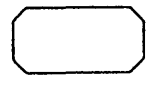
Contiene el nombre de bloque (véanse la Recomendación Z.101, § 3.1.1 y las Directrices para el usuario, § D.4.3.2).



CCITT-74040

C1.1.1.3 *Símbolo de proceso*

Contiene el nombre de proceso (véanse la Recomendación Z.101, § 3.1.1 y las Directrices para el usuario, § D.4.3.4) y puede contener un par de paréntisis que encierran la lista de los parámetros formales.



CCITT-74040

C1.1.1.4 *Símbolo de entorno*

Se representa solamente mediante una palabra clave (véanse la Recomendación Z.101, § 3.1.1).

ENTORNO

C1.1.1.5 *Símbolo de canal*

Contiene un nombre de canal. El símbolo de canal tiene un extremo de origen conectado a un símbolo de bloque y un extremo de destino conectado a otro símbolo de bloque. Alternativamente, los puntos extremos de origen o de destino (pero no ambos a la vez) se pueden conectar a un símbolo de entorno en lugar de un símbolo de bloque.

Un símbolo de canal incluye una flecha en medio de la línea, que muestra la dirección del flujo de señales.

CCITT-74040

C1.1.1.6 *Símbolo de ruta de señal*

Un símbolo de ruta de señal en un bloque puede estar asociado con un símbolo de lista de señales.

Un símbolo de ruta de señal conduce de un proceso a otro o de un proceso al extremo de origen de un canal (en la frontera de bloque), o de un extremo de destino a un canal (en la frontera de bloque) a un proceso.

Los símbolos de ruta de señal pueden converger en el extremo de origen de un canal (en la frontera de bloque) y divergir desde el extremo de destino de un canal en la frontera de bloque. Un símbolo de ruta de señal comprende una flecha dirigida a un extremo que muestra la dirección del flujo de señales.

CCITT-74040

C1.1.1.7 *Símbolo de lista de señales*

CCITT-74040

El símbolo de lista de señales contiene una lista de nombres. La lista de señales propiamente dicha puede tener un nombre, que se escribe debajo del símbolo. Las inscripciones en la lista, que están separadas por comas y ordenadas en columnas o renglones, son los nombres de señales o nombres individuales de otras listas de señales. Dentro de la lista, los nombres de listas de señales se distinguen de los nombres de señales individuales encerrando cada nombre de lista de señal en el interior de un nuevo par de corchetes rectangulares.

C1.1.1.8 *Símbolo de crear*

CCITT-74040

Se utiliza entre símbolos de proceso para indicar que una instancia de proceso en el extremo de la flecha es creada por una instancia de proceso en el extremo de origen de la línea de trazo interrumpido. El símbolo de crear puede conectar el mismo proceso para indicar la creación de una instancia de dicho proceso por otra instancia del mismo proceso.

C1.1.2 *Reglas*

Cada bloque tiene cierto número de líneas dirigidas para interconectarlo con otros bloques y/o con el entorno.

Si se representan procesos en el diagrama de interacción, las líneas de señal entre procesos tienen que existir solamente entre procesos del mismo bloque que comunican entre sí.

C1.1.3 *Convenios*

Los símbolos de canal y líneas de señal deben, respectivamente, unir fronteras de símbolo de bloque y fronteras de símbolo de proceso de preferencia en ángulos de 90°. De ser necesario, los símbolos de canal pueden contener curvas de 90°

El cruce de líneas no tiene significado.

C1.2 *Diagrama de subestructura de canales*

Teniendo en cuenta que los componentes son bloques y canales, el diagrama se parece a un diagrama de subestructura de bloques.

C1.2.1 *Símbolos*

Los símbolos utilizados en este diagrama son los mismos que se utilizan en el diagrama de subestructura de bloques.

C1.2.2 *Reglas*

Las reglas para la conexión del diagrama son las mismas que se utilizan para el diagrama de subestructura de bloques.

C1.2.3 *Convenios*

También los mismos convenios gráficos, descritos para el DIB, se aplican al diagrama de subestructura de canales.

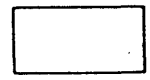
C1.3 *Árbol de bloques*

El diagrama de árbol de bloques contiene casillas y líneas de «partición en».

C1.3.1 *Símbolos*

C1.3.1.1 *Casilla*

Una casilla representa un sistema o un bloque. El nombre del objeto representado debe figurar dentro del símbolo.



CCITT-74040

C1.3.1.2 *Líneas de «partición en»*

Representan la relación entre el objeto situado en la parte superior que es objeto de partición en los sub-bloques que figuran en la parte inferior.

CCITT-74040

C1.3.2 *Reglas*

Los símbolos están conectados para que formen un árbol jerárquico. Para la conexión del diagrama se aplican las reglas siguientes:

- hay exactamente una sola casilla raíz (casilla superior);
- toda otra casilla tiene que seguir una bifurcación de una línea de «partición en»;
- cualquier casilla puede ir seguida de una línea de «partición en»;
- una línea de «partición en» debe ir seguida por casillas en todas sus bifurcaciones, y tiene que seguir a una casilla.

C1.3.3 *Convenios*

El árbol debe establecerse de tal forma que un nivel de partición aparezca como un nivel coherente en las representaciones, es decir, las líneas de «partición en» deben tener igual longitud en el resto del diagrama.

C1.4 *Árbol de procesos*

El árbol de procesos contiene símbolos de proceso y líneas de «partición en».

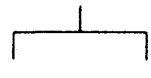
C1.4.1 *Símbolos*

C1.4.1.1 *Símbolo de proceso*

Como se define en la Recomendación Z.101 (§ 3.1.2), el nombre del proceso a que se hace referencia debe aparecer en el interior del símbolo. La sintaxis de comentario se utiliza para mostrar el bloque en que se ha atribuido el proceso.

C1.4.1.2 *Líneas de «partición en»*

Representan la relación del proceso superior cuya partición ha originado los sub-procesos situados en la parte inferior, y puede ser sustituido por éstos.



CCITT-74040

C1.4.2 *Reglas*

Los símbolos están conectados para que formen un árbol jerárquico. Para la conexión del diagrama se aplican las siguientes reglas:

- hay exactamente un solo símbolo de proceso raíz (proceso superior), que va seguido de una línea de «partición en»;
- la línea de «partición en» va seguida, en cada una de sus bifurcaciones, por un símbolo de proceso.

C1.4.3 *Convenios*

Si el diagrama de árbol de proceso es grande, puede ser conveniente dividirlo en varios diagramas. Esta división debe efectuarse de forma tal que el primer diagrama esté cortado de modo que un conjunto de procesos que sean objeto de nueva partición aparezca como no particionado. En los siguientes diagramas estos procesos aparecen como raíces.

C1.5 *Diagrama de procesos*

Un diagrama de procesos es un conjunto de símbolos conectado por líneas de flujo.

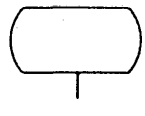
C1.5.1 *Símbolos*

C1.5.1.1 *Símbolo de arranque*

Contiene el nombre de proceso del proceso que describe (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4.3 y § D.6.3).

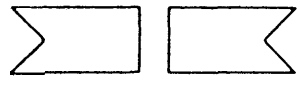
C1.5.1.2 *Símbolo de estado*

Contiene el nombre de estado o un asterisco, o un asterisco seguido por nombres de estado dentro de corchetes rectangulares (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4.3 y D.6.3).



C1.5.1.3 *Símbolo de entrada*

Contiene los nombres de señal separados por comas o un asterisco (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4.3 y § D.6.3).

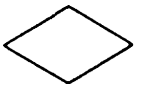


C1.5.1.4 *Símbolo de conservación*

Contiene los nombres de señal separados por comas o por una estrella (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4.3 y § D.6.3).

C1.5.1.5 *Símbolo de decisión*

Contiene el nombre de decisión (facultativo) y una frase de texto formal o informal (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4.3 y § D.6.3).



C1.5.1.6 *Símbolo de salida*

Contiene los nombres de señal separados por comas (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4.3 y § D.6.3).

C1.5.1.7 *Símbolo de tarea*

Contiene el nombre de tarea (facultativo) y una frase de texto formal o informal (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4.3 y § D.6.3).

C1.5.1.8 *Símbolo de llamada de procedimiento*

Contiene el nombre de procedimiento y los parámetros actuales dentro de corchetes curvos (véanse la Recomendación Z.103, § 2.2 y las Directrices para el usuario, § D.4.3 y § D.6.3).

C1.5.1.9 *Símbolo de arranque de procedimiento*

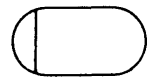
Contiene el nombre de procedimiento y los parámetros formales dentro de corchetes curvos (véanse la Recomendación Z.103, § 2.2 y las Directrices para el usuario, § D.4.3 y § D.6.3).

C1.5.1.10 *Símbolo de retorno*

Es un círculo con una cruz en el interior (véanse la Recomendación Z.103, § 2.2 y las Directrices para el usuario, § D.4.3 y § D.6.3).

C1.5.1.11 *Símbolo de acceso de entrada de macro*

Contiene el nombre del macro (véanse la Recomendación Z.103, § 4.2 y las Directrices para el usuario, § D.4.3 y § D.6.3).

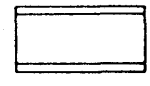


C1.5.1.12 *Símbolo de acceso de salida de macro*

Es un círculo con una raya en el interior (véanse la Recomendación Z.103, § 4.2 y las Directrices para el usuario, § D.4.3 y § D.6.3).

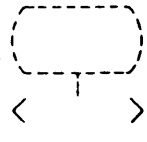
C1.5.1.13 *Símbolo de crear*

Contiene el nombre de proceso y los parámetros actuales entre corchetes (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4.3 y § D.6.3)



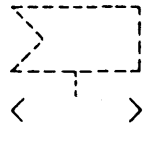
C1.5.1.14 *Símbolo de señal continua*

Contiene el texto de la condición y, seguidamente, la palabra clave PRIORIDAD seguida por el número de prioridad asociado (véanse la Recomendación Z.103, § 3.3.3 y las Directrices para el usuario, § D.4).



C1.5.1.15 *Símbolo de condición habilitante*

Contiene el texto de la condición (véanse las Directrices para el usuario, § D.4).



C1.5.1.16 *Símbolo de alternativa*

Contiene el texto alternativo (véanse la Recomendación Z.103, § 5.2 y las Directrices para el usuario, § D.4).

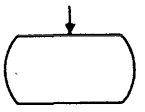
C1.5.1.17 *Símbolo de unión o conector*

Contiene el nombre de unión (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4).

CCITT-74040

C1.5.1.18 *Símbolo de estado siguiente*

Contiene el nombre de estado o de un sistema (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4).



CCITT-74040

C1.5.1.19 *Símbolo de parada*

Es una cruz (véanse la Recomendación Z.101, § 3.3.1 y las Directrices para el usuario, § D.4).

CCITT-74040

C1.5.2 *Reglas*

- Un estado se tiene que conectar con uno o varios símbolos de entrada o símbolos de conservación (cualquier conexión con otros símbolos es errónea).
- Un nodo de procedimiento de llamada puede seguir a cualquier nodo que no sea un nodo de estado, un nodo de parada o un nodo de retorno.
- Un nodo de llamada puede ir seguido por cualquier nodo, salvo un nodo de arranque, un nodo de arranque de procedimiento, un nodo de entrada, un nodo de conservación.
- Las condiciones habilitantes pueden estar asociadas a cualquier símbolo de entrada.
- Una señal continua va asociada a una línea de flujo que sale de un símbolo de estado, pero no tiene un símbolo de entrada.
- El símbolo de llamada de macro puede insertarse en cualquier diagrama (diagrama de procesos, DIB, diagrama de ilustración general de estados), y en cualquier lugar del diagrama. Puede tener uno o más accesos de entrada. En el caso en que hay varios accesos de entrada, cada uno de ellos tiene que tener asociada una etiqueta. Puede tener ninguno, uno o varios accesos de salida; en el último caso, cada acceso de salida tiene que tener asociada una etiqueta. Los accesos de entrada se representan mediante flechas que apuntan al símbolo de macro; los accesos de salida se representan mediante flechas que salen del símbolo de macro. Los accesos de entrada/accesos de salida de símbolo de macro están conectados a otros símbolos mediante líneas de flujo del tipo adecuado, según su significado.
- Un símbolo de parada o un símbolo de retorno de un procedimiento y de un macro no van seguidos de ningún símbolo.
- El símbolo de crear se puede colocar en todo lugar en que pueda existir una tarea.
- El símbolo de alternativa se puede insertar directamente en una transición sólo si los comportamientos alternativos no incluyen estados y si terminan en el mismo punto.
- Una línea de flujo de trazo ininterrumpido puede ser interrumpida por un par de conectores asociados; se supone que el flujo va del conector de salida a su conector de entrada asociado (símbolo de unión).
- Cuando a dos o más símbolos sigue un solo símbolo, las líneas de flujo que conducen a este último convergen. Esta convergencia puede indicarse como una línea de flujo que confluye con otra, como más de un conector de salida asociado a un solo conector de entrada, o como líneas de flujo separadas que entran al mismo símbolo.

Ejemplo: convergencia

CCITT-74040

- Un conector de salida no va seguido de ningún otro símbolo, y de él no sale ninguna línea de flujo.
- Cuando a un símbolo siguen dos o más símbolos, una línea de flujo procedente del primer símbolo puede divergir y formar dos o más líneas de flujo.

Ejemplo: divergencia

CCITT-74040

- Deberá indicarse mediante flechas que dos líneas de flujo convergen, o que una línea de flujo entra en un conector de salida o en un símbolo de estado. No podrán utilizarse flechas para indicar líneas de flujo que entran a símbolos de entrada.
- El símbolo de conector se conecta a un símbolo o a una línea de flujo de trazo ininterrumpido.

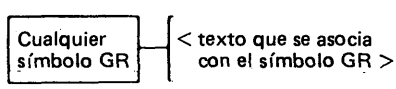
C1.5.3 *Convenios*

- Es preferible que, para un diagrama dado, todos los símbolos del mismo tipo sean del mismo tamaño.
- La orientación preferida de los símbolos es la horizontal, y la relación de aspecto (anchura/altura) preferida de los símbolos es 2:1.

C1.5.4 *Símbolos generales*

C1.5.4.1 *Símbolo de ampliación de texto*

Puede afectarse a todos los símbolos LED/GR. El texto contenido en este símbolo debe considerarse contenido en el símbolo al que se afecta el símbolo de ampliación de texto.



CCITT-74040

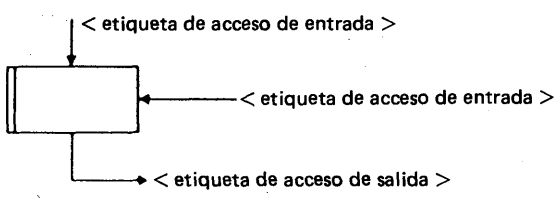
C1.5.4.2 *Símbolo de comentario*

El símbolo de comentario contiene el texto del comentario.

CCITT-74040

C1.5.4.3 *Símbolo de macro, acceso de entrada y acceso de salida*

El símbolo de macro es la referencia a la definición de macro. Los accesos de entrada de macro y los accesos de salida de macro se representan por líneas de flujo que conducen al símbolo y salen del mismo. Pueden afectarse facultativamente etiquetas a las líneas de flujo. El símbolo de macro contiene el nombre de la definición de macro.



CCITT-74040

C1.6 *Diagrama de ilustración general de estados*

Este diagrama representa la secuencia de estados en el proceso. Es posible ver los estados a los que se puede tener acceso a partir de un estado determinado (véanse las Directrices para el usuario, § D.4.2 y § D.4.3).

C1.6.1 *Símbolos*

C1.6.1.1 *Símbolo de estado*

En este diagrama el símbolo de estado es un círculo que contiene el nombre de estado (véanse las Directrices para el usuario, § D.4.2 y § D.4.3).

C1.6.2 *Reglas*

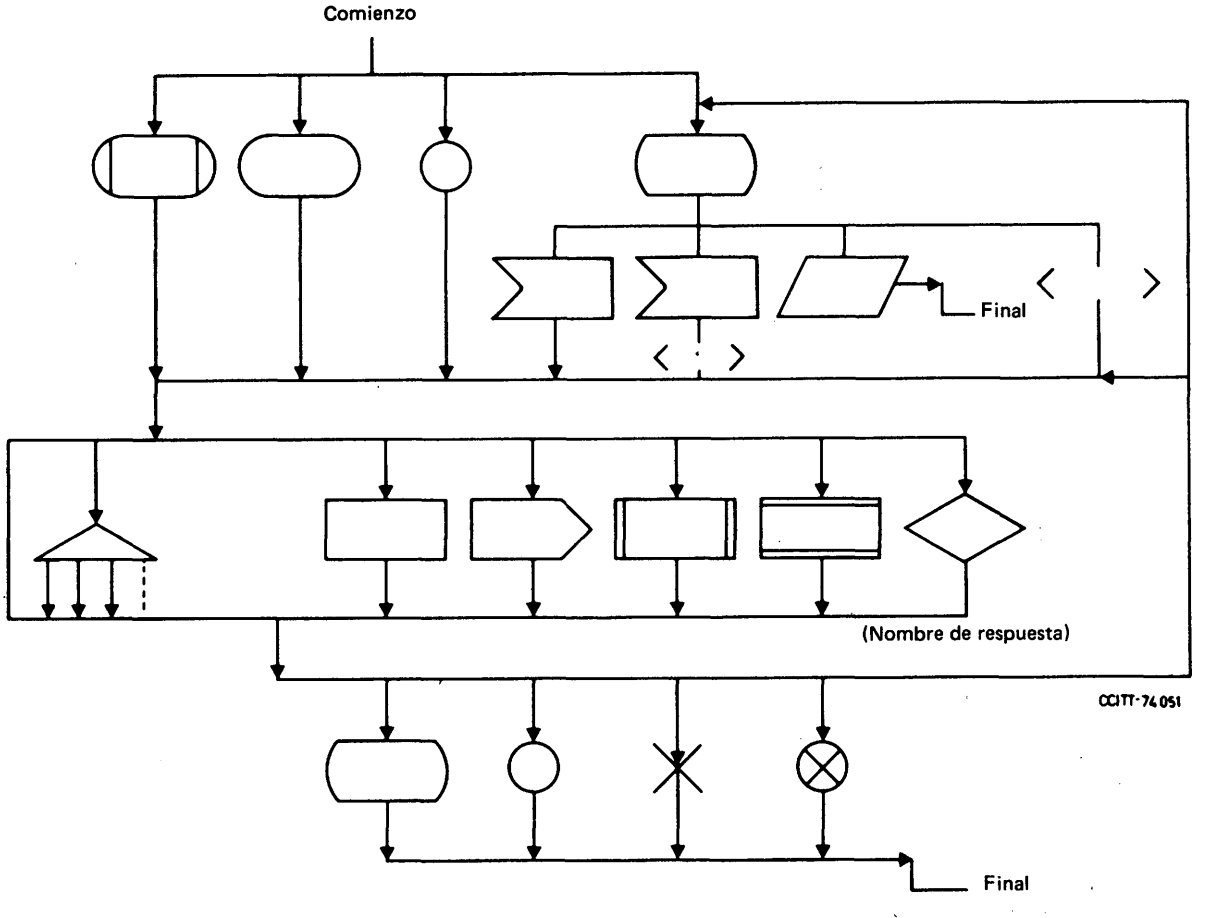
- Los estados están conectados por arcos. Los nombres de señal de entrada se pueden, a título facultativo, escribir en los arcos.

C1.7 *Convenios generales*

En la sintaxis gráfica hay algunos convenios de dibujo que son válidos para todos los tipos de diagramas:

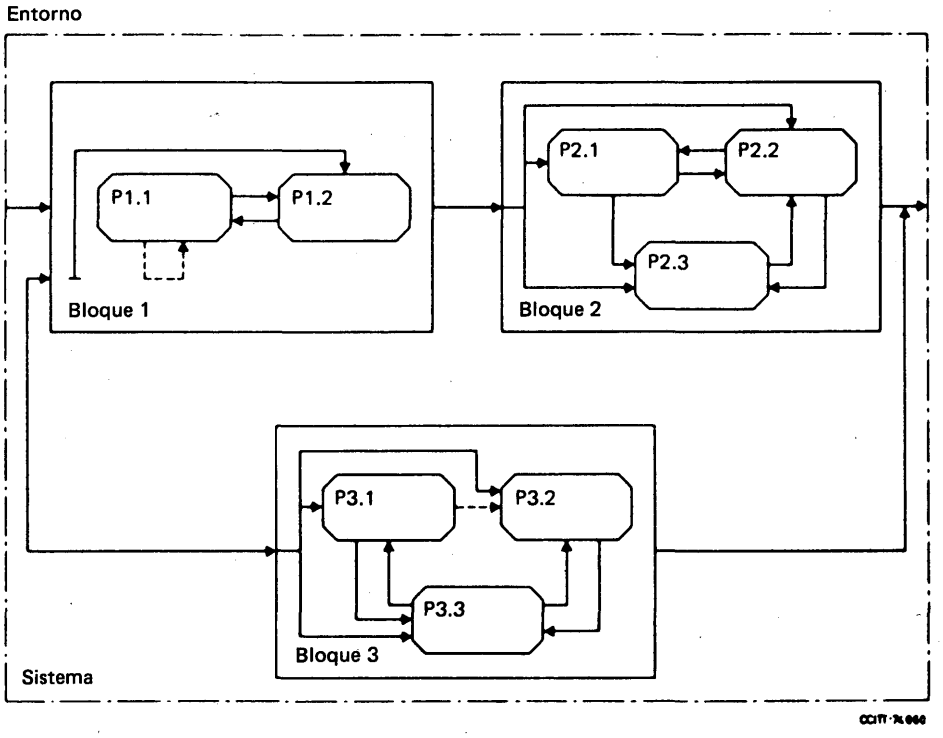
- la relación de aspecto (anchura/altura) y tamaño de los símbolos es variable y se deja a criterio de los usuarios;
- los límites de símbolo no deben superponerse o cruzarse. El caso de los símbolos de canal y de flujo de señales constituye una excepción a esta regla, ya que estos símbolos pueden cruzarse entre sí. No hay relación lógica entre símbolos de canal o símbolos de flujo de señal que se cruzan.

C1.8 *Conexiones admisibles de símbolos por medio de líneas de flujo en un diagrama de proceso LED/GR*

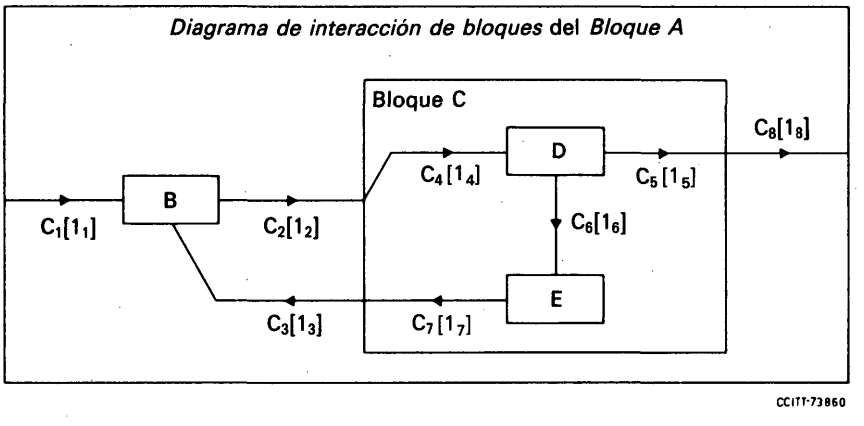


C1.9 *Ejemplos*

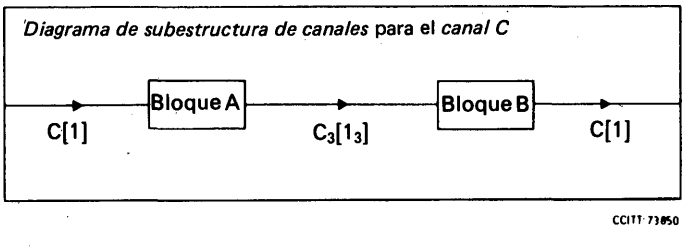
C1.9.1 *Diagrama de interacción de bloques*



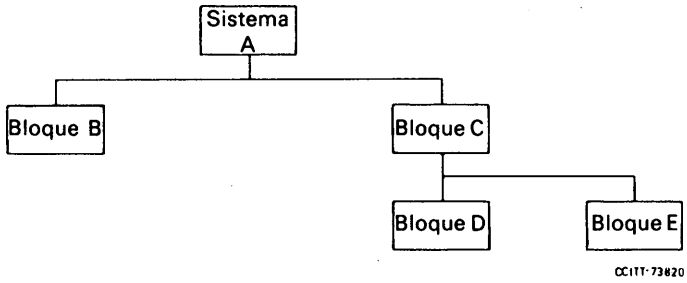
C1.9.2 *Diagrama de interacción de bloques*



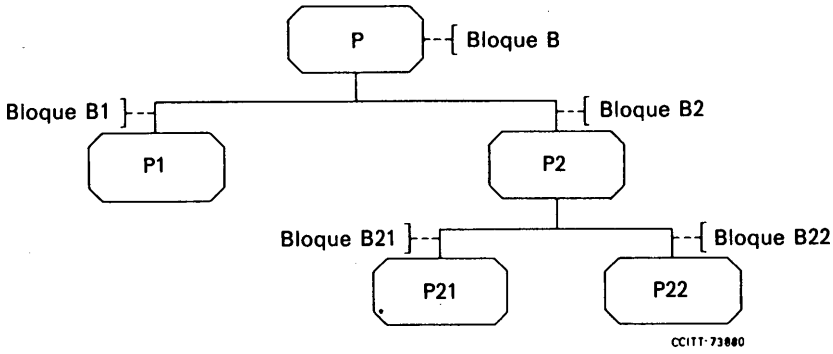
C1.9.3 *Diagrama de subestructura de canales*



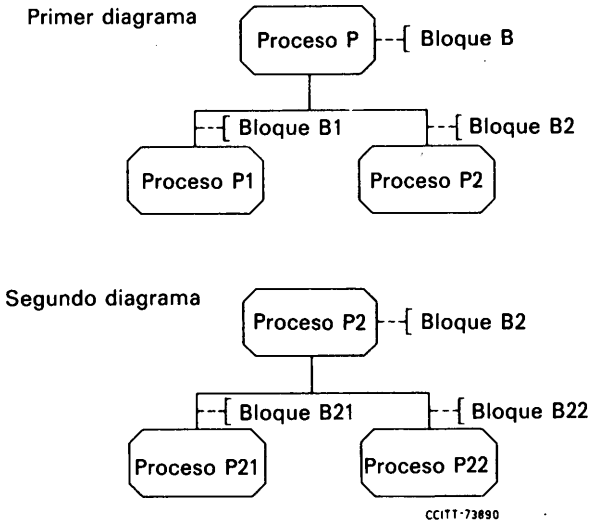
C1.9.4 *Árbol de bloques*



C1.9.5 *Árbol de procesos*



Otra representación del mismo árbol de procesos (útil cuando el árbol de procesos es de gran tamaño).



ANEXO C2

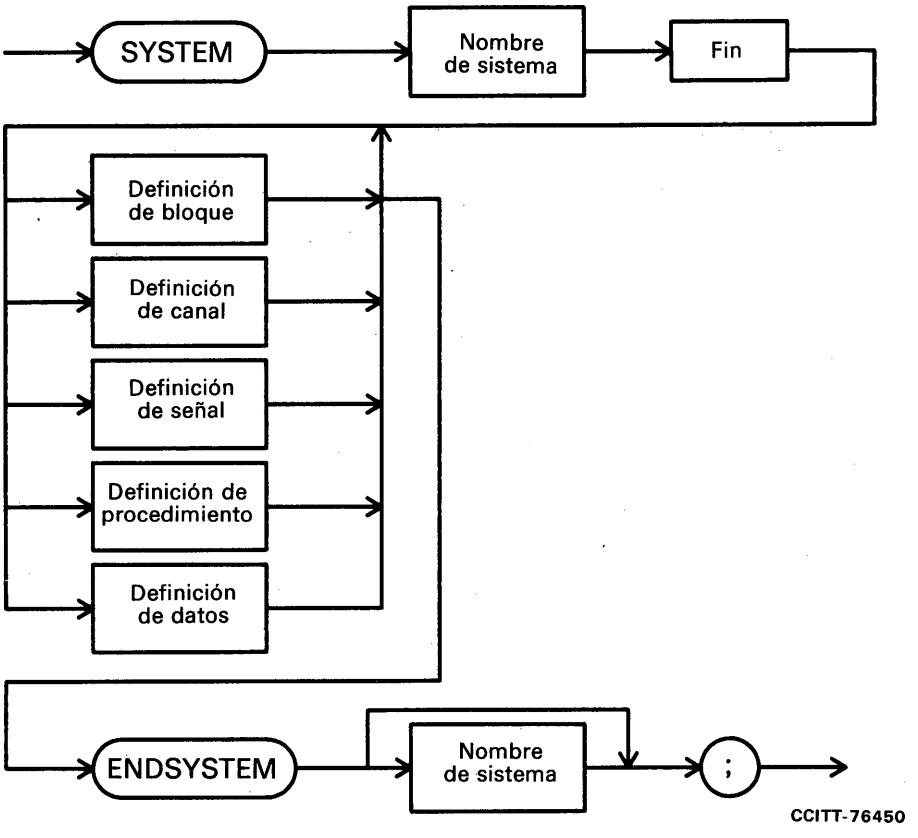
(a las Recomendaciones Z.100 a Z.104)

Resumen del LED/PR

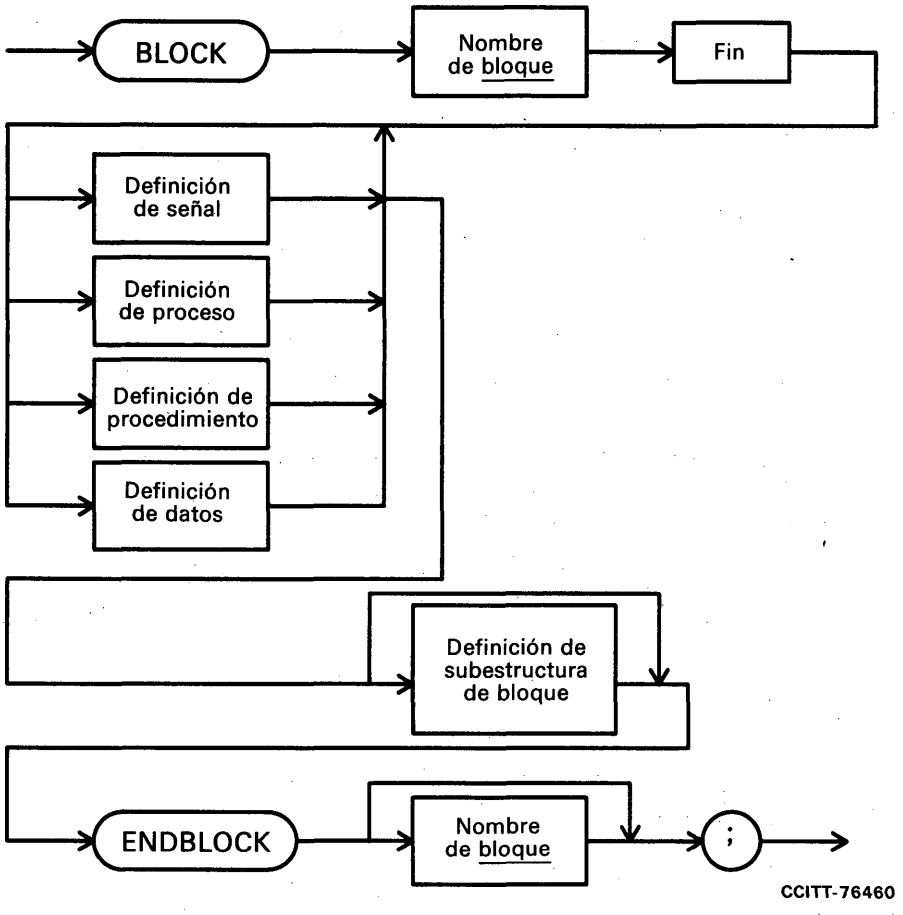
C2.1 *Sintaxis*

Diagramas de sintaxis

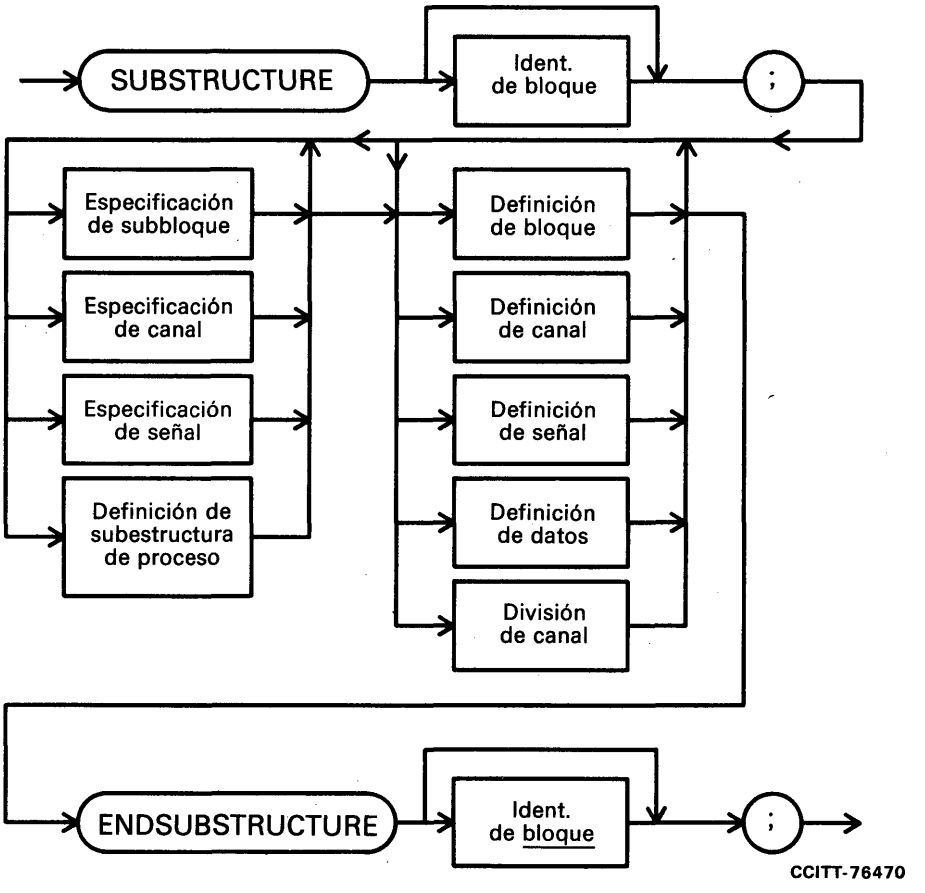
DEFINICIÓN DE SISTEMA



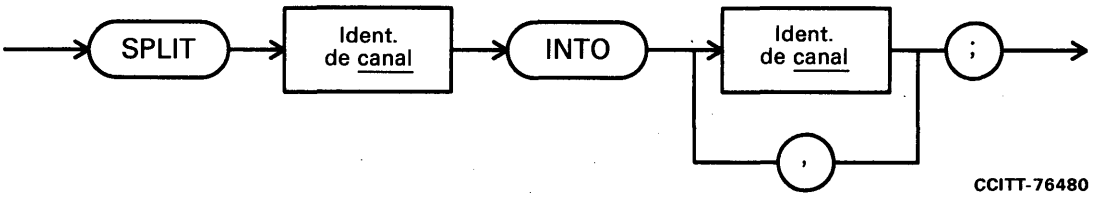
DEFINICIÓN DE BLOQUE



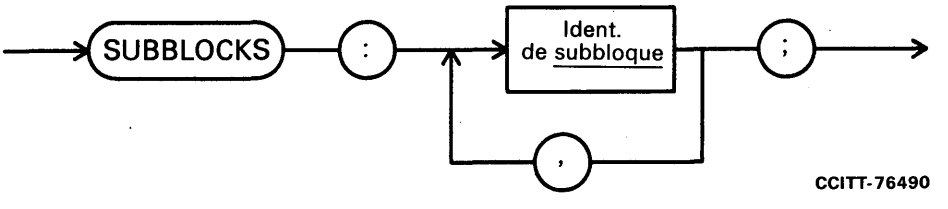
DEFINICIÓN DE SUBESTRUCTURA DE BLOQUE



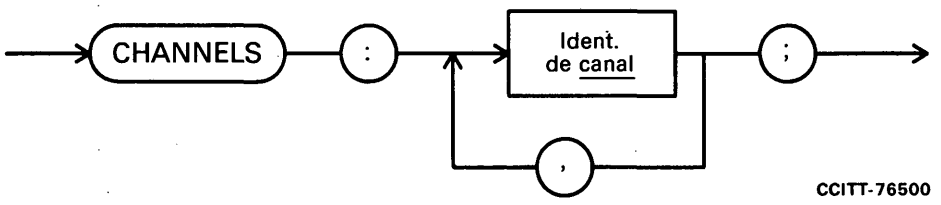
DIVISIÓN DE CANAL



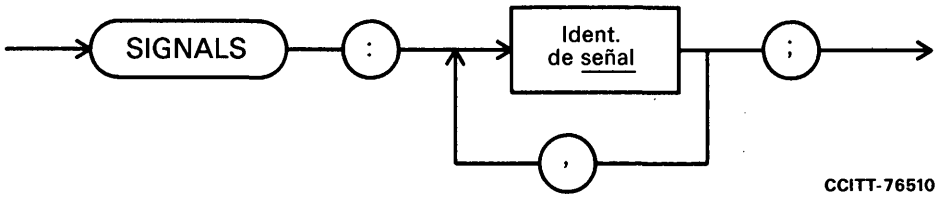
ESPECIFICACIÓN DE SUBBLOQUE



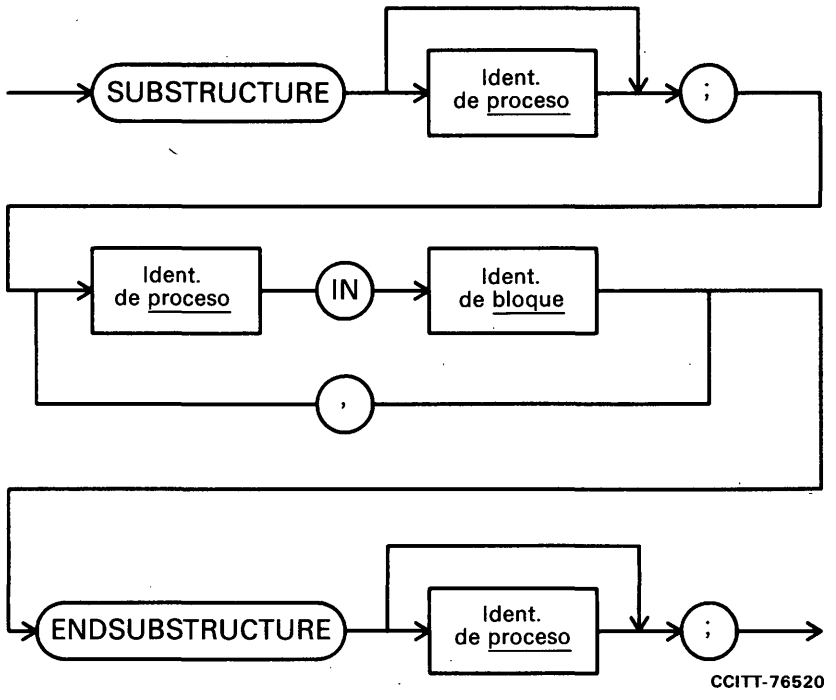
ESPECIFICACIÓN DE CANAL



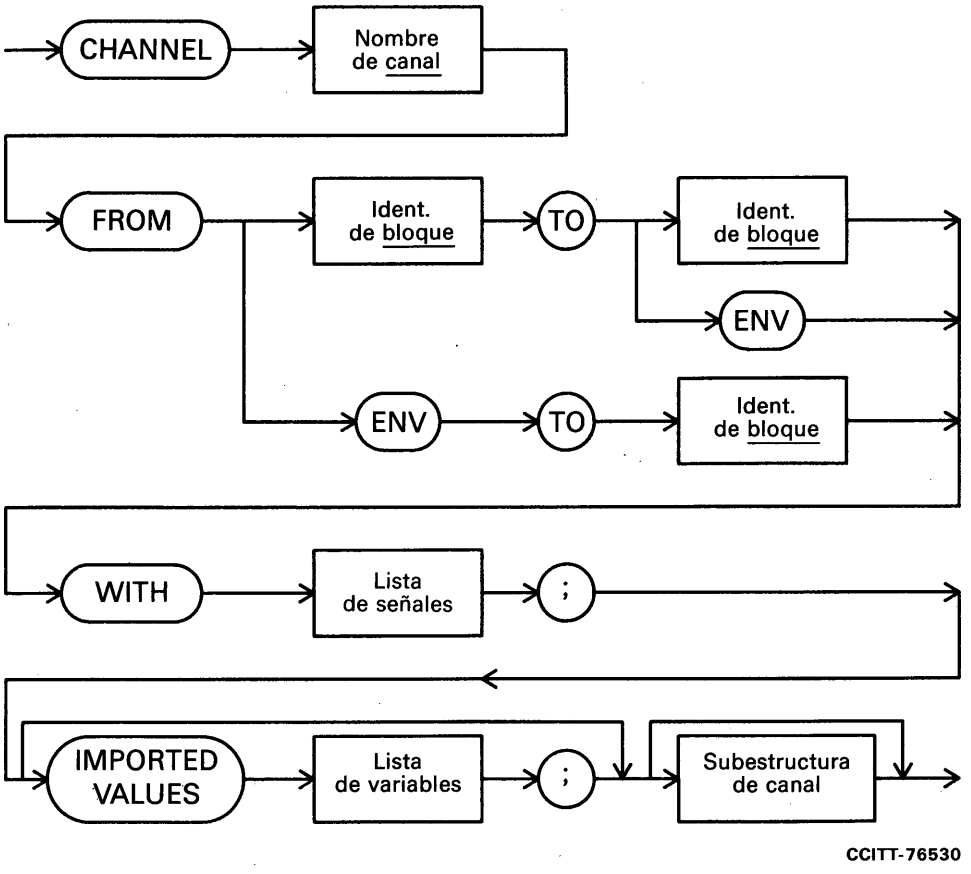
ESPECIFICACIÓN DE SEÑAL



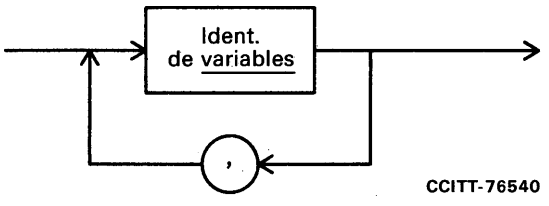
DEFINICIÓN DE SUBESTRUCTURA DE PROCESO



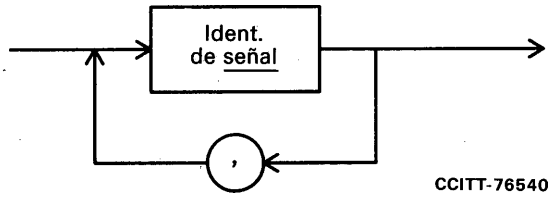
DEFINICIÓN DE CANAL



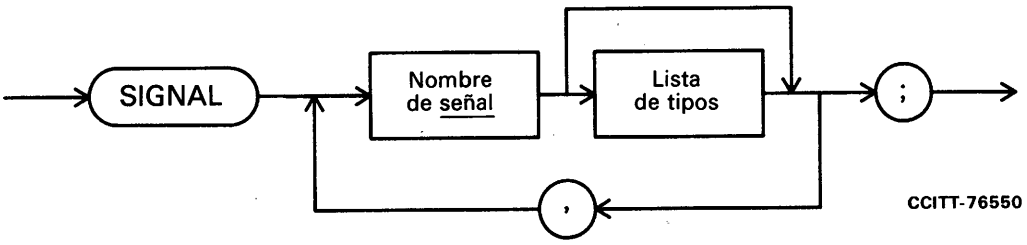
LISTA DE VARIABLES



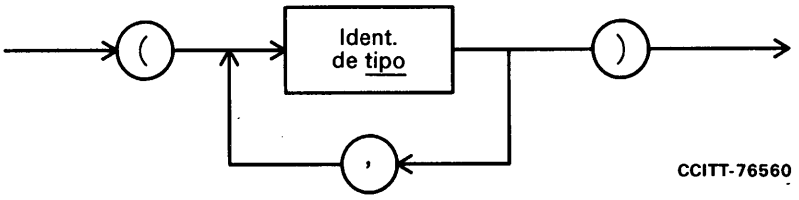
LISTA DE SEÑALES



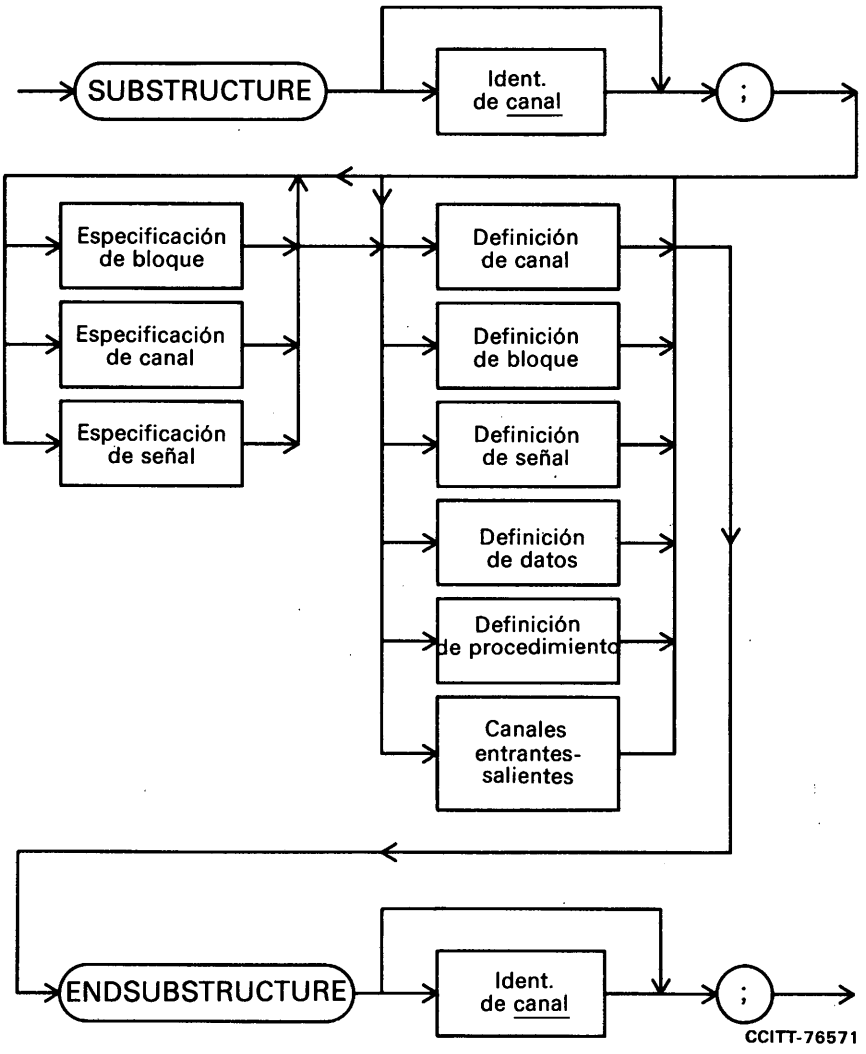
DEFINICIÓN DE SEÑAL



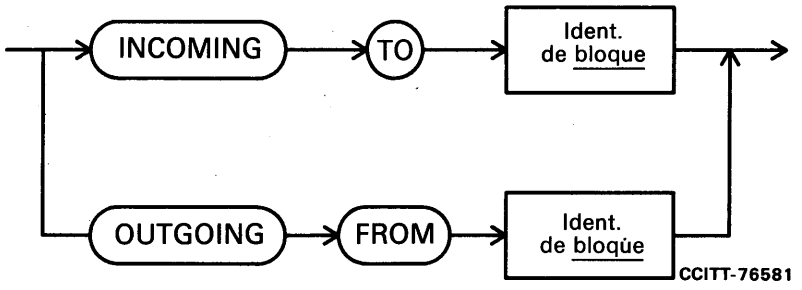
LISTA DE TIPOS



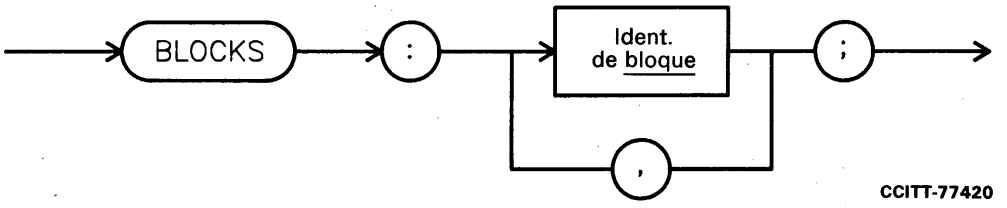
DEFINICIÓN DE SUBESTRUCTURA DE CANAL



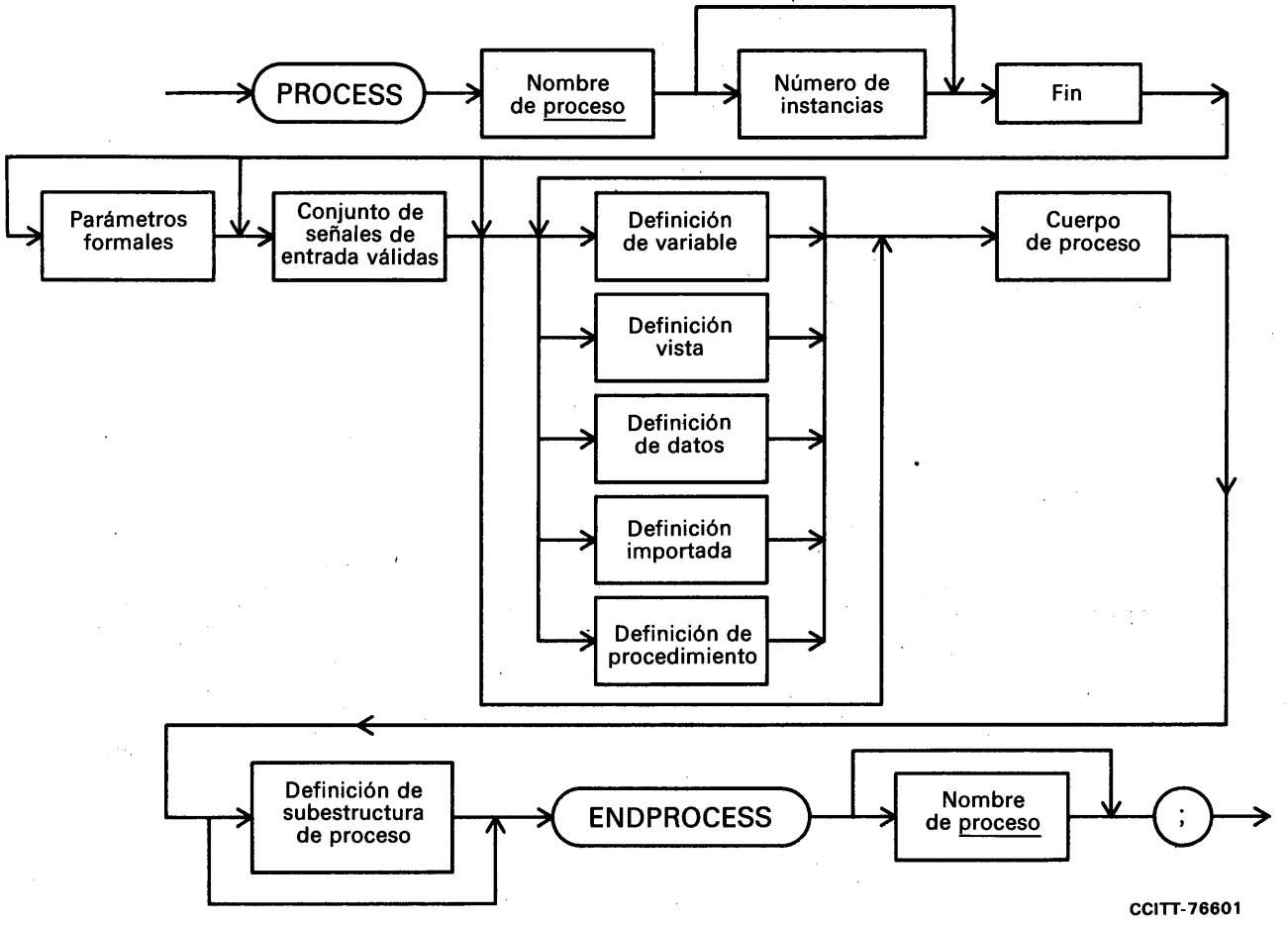
CANALES ENTRANTES-SALIENTES



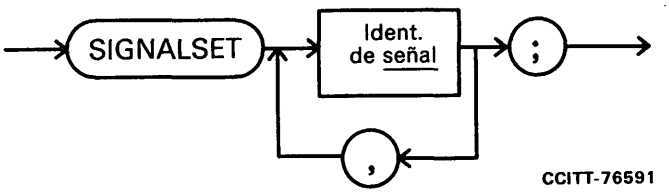
ESPECIFICACIÓN DE BLOQUE



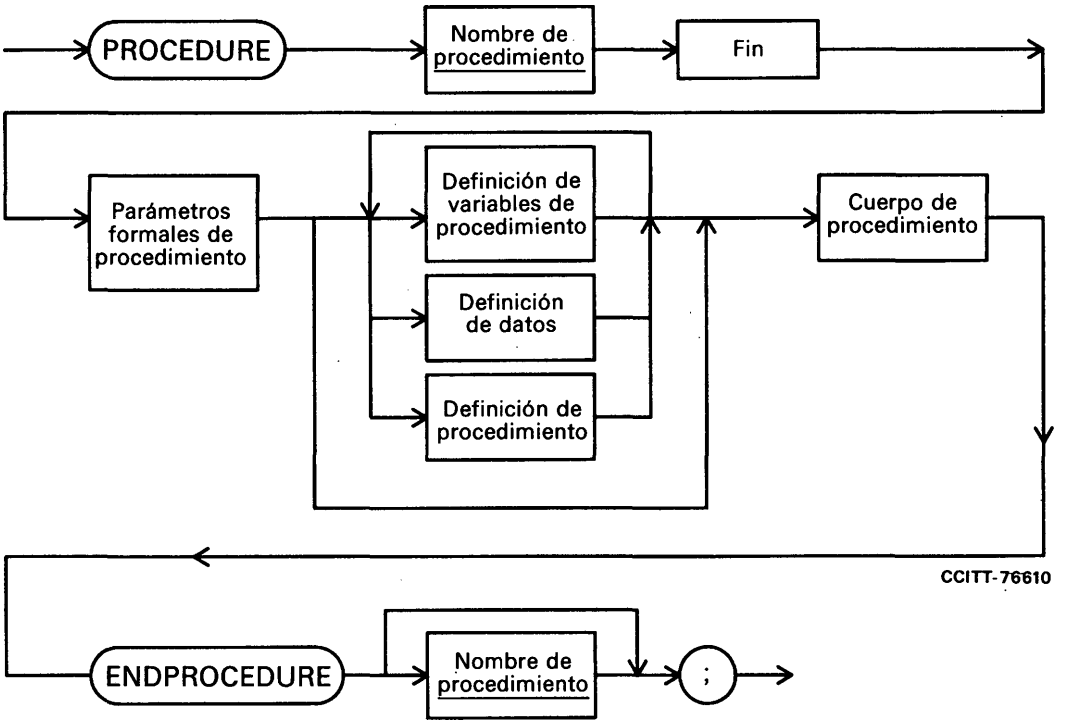
DEFINICIÓN DE PROCESO



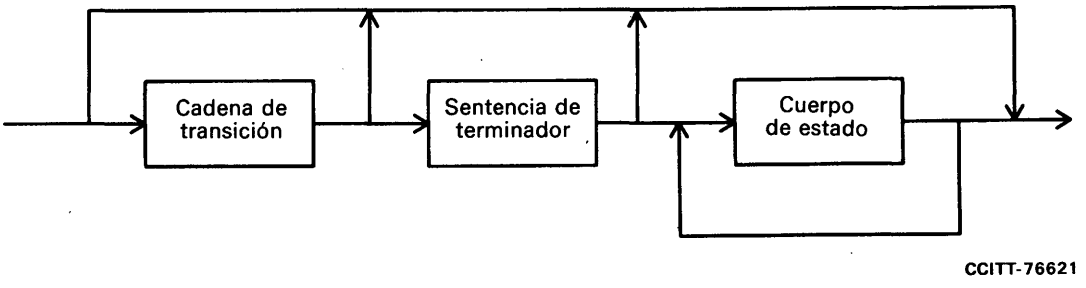
CONJUNTO DE SEÑALES DE ENTRADA VÁLIDAS



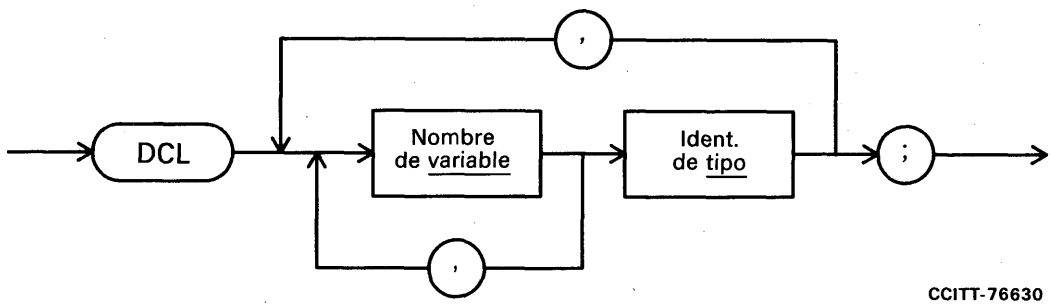
DEFINICIÓN DEL PROCEDIMIENTO



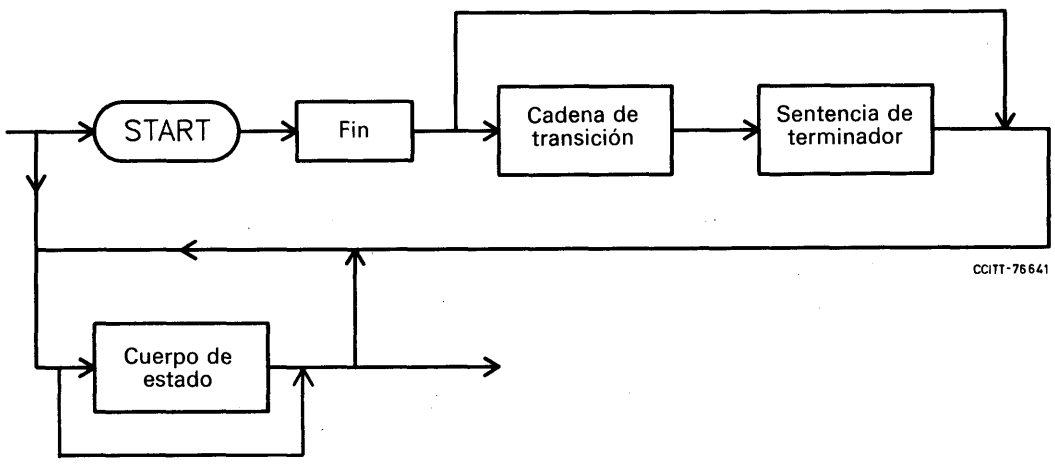
CUERPO DE PROCEDIMIENTO



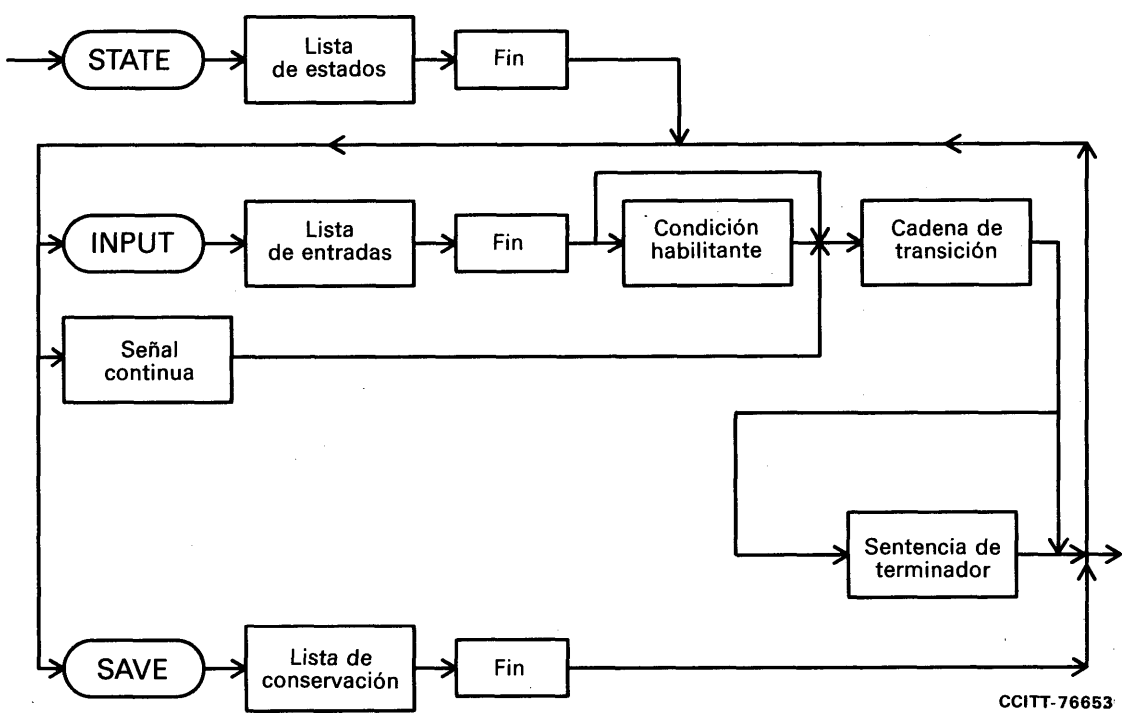
DEFINICIÓN DE VARIABLES DE PROCEDIMIENTO



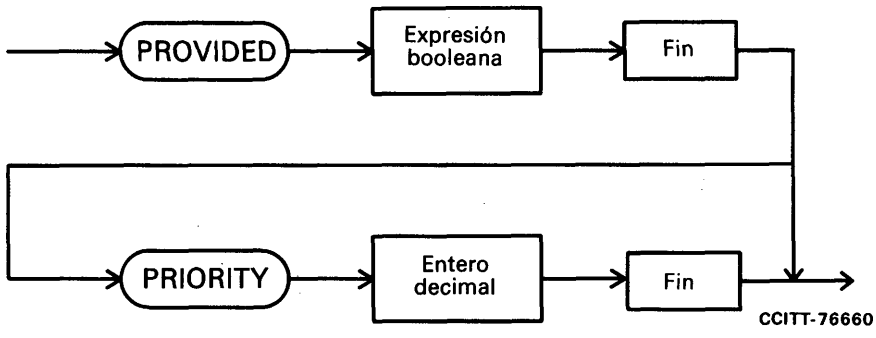
CUERPO DE PROCESO



CUERPO DE ESTADO

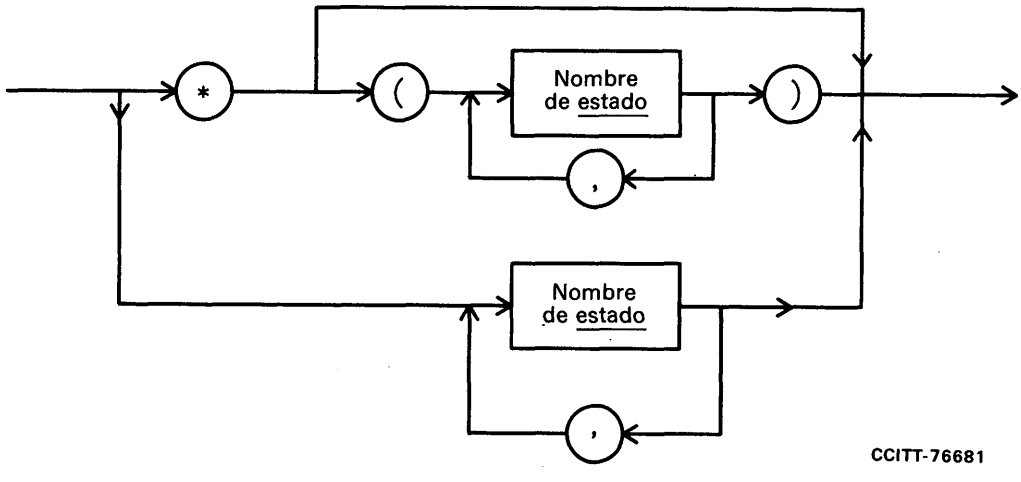


SEÑAL CONTINUA

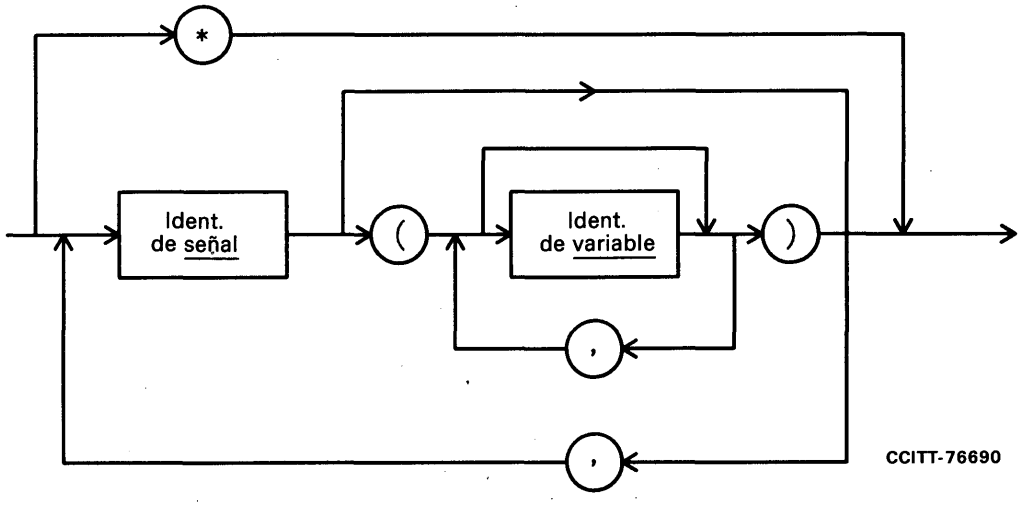


CONDICIÓN HABILITANTE

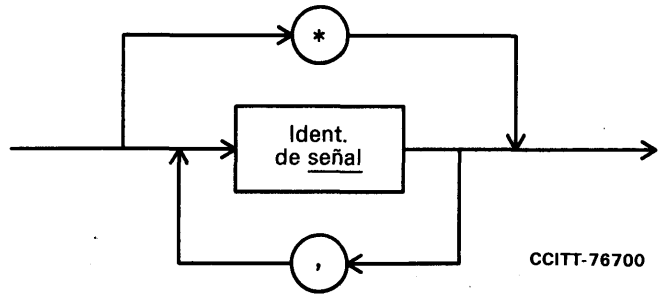
LISTA DE ESTADOS



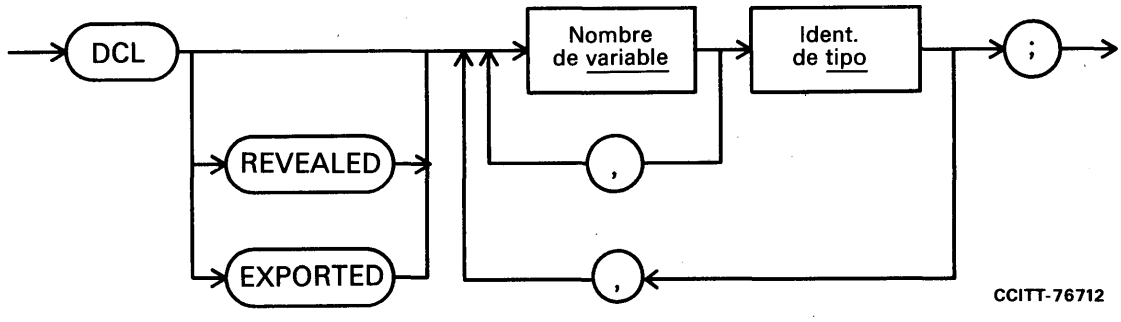
LISTA DE ENTRADAS



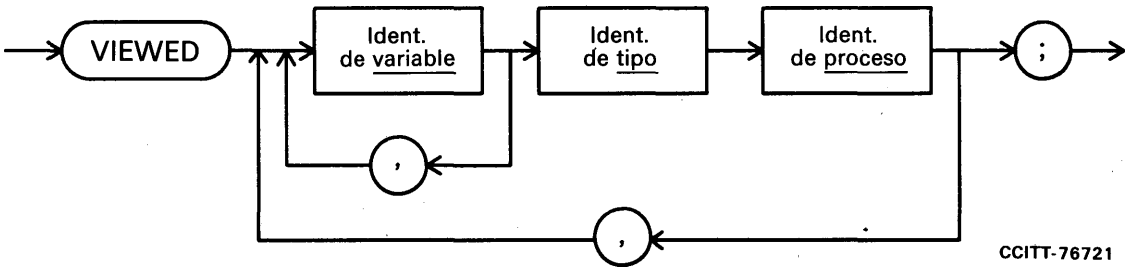
LISTA DE CONSERVACIÓN



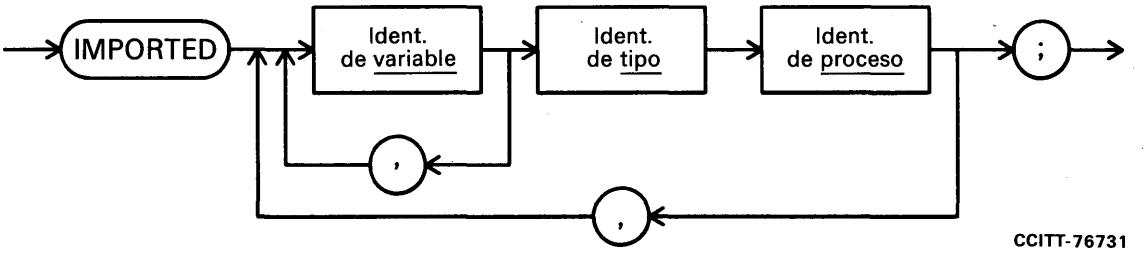
DEFINICIÓN DE VARIABLE



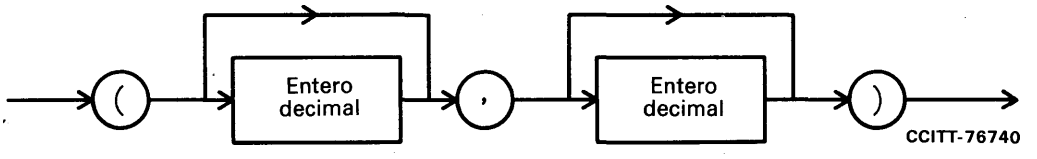
DEFINICIÓN DE VISIÓN



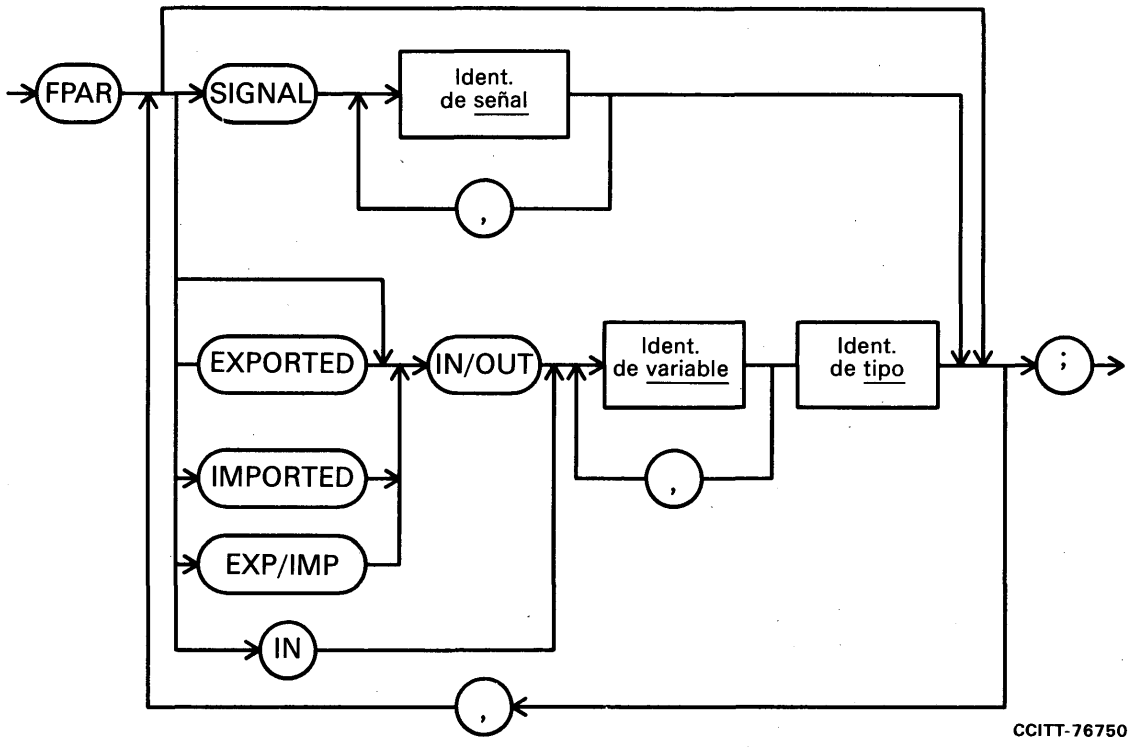
DEFINICIÓN DE IMPORTACIÓN



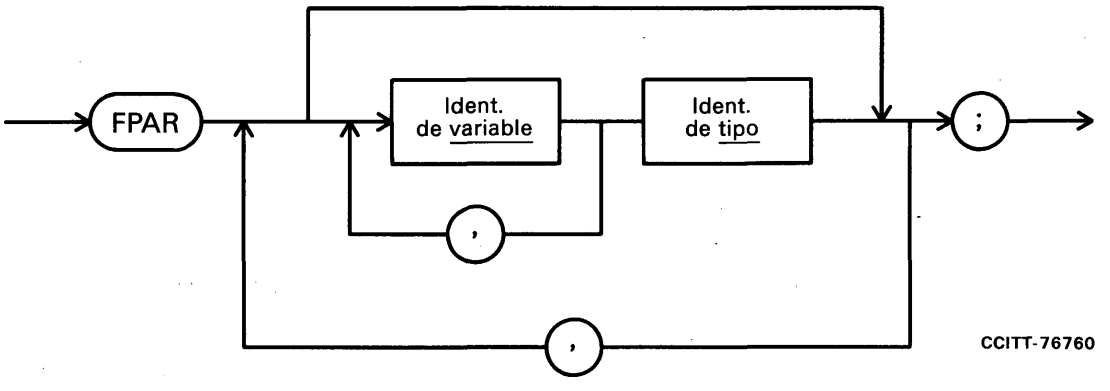
NÚMERO DE INSTANCIAS



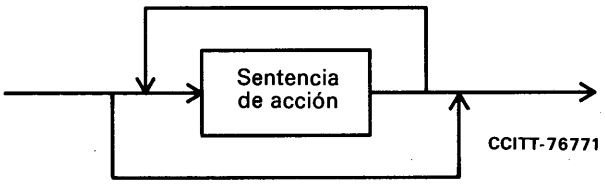
PARÁMETROS FORMALES DE PROCEDIMIENTO



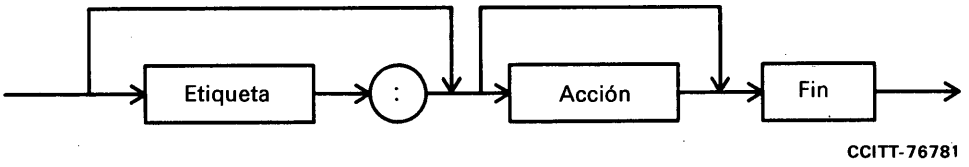
PARÁMETROS FORMALES



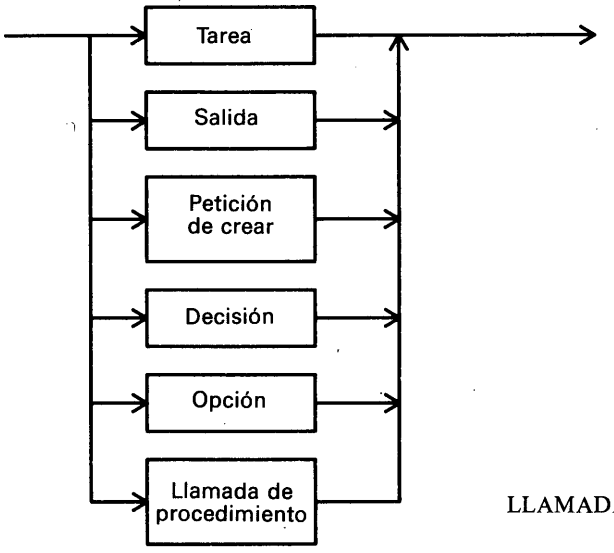
CADENA DE TRANSICIÓN



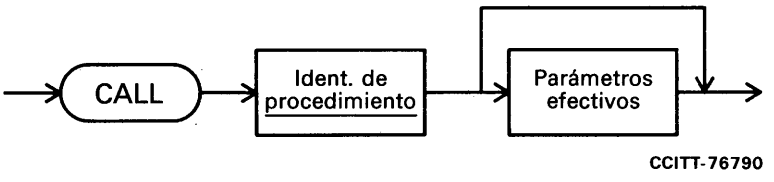
SENTENCIA DE ACCIÓN



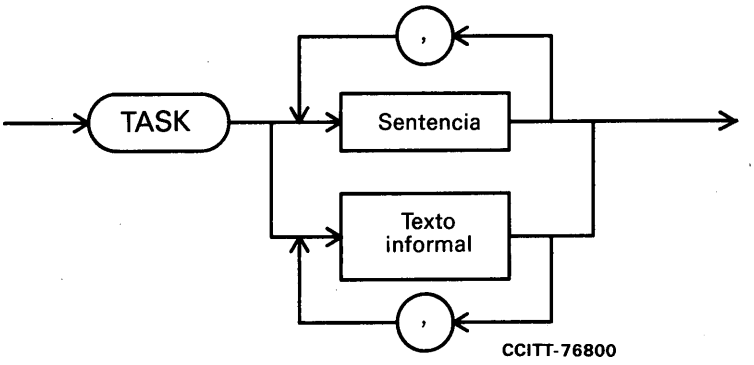
ACCIÓN



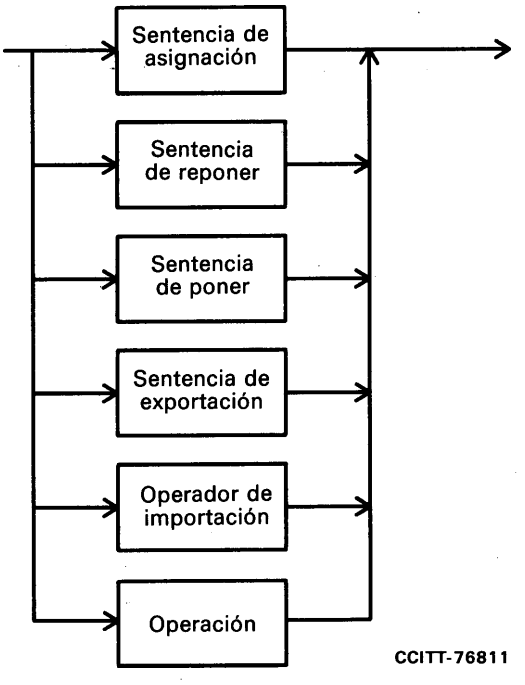
LLAMADA DE PROCEDIMIENTO



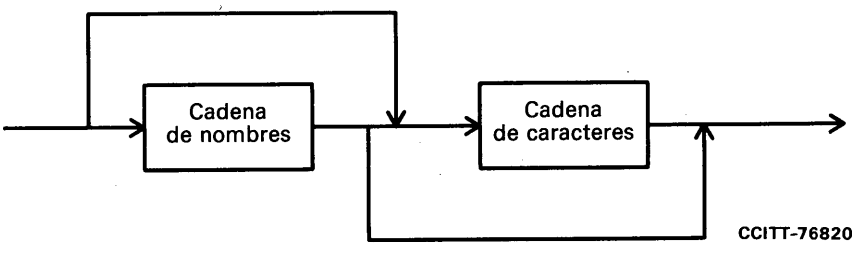
TAREA



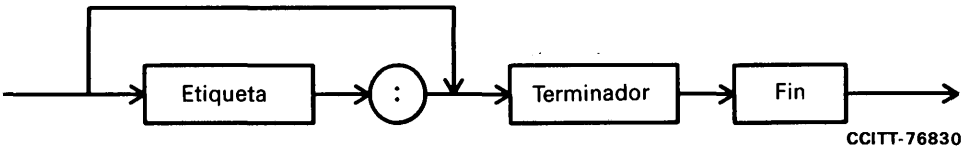
SENTENCIA



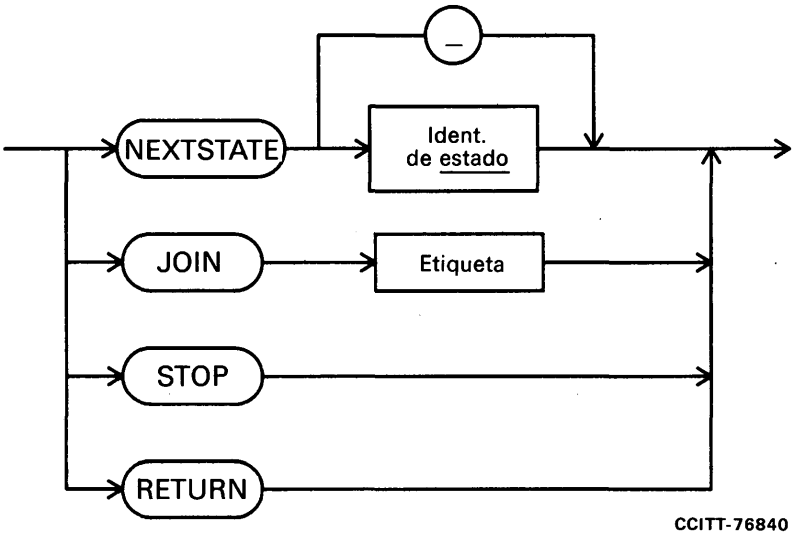
TEXTO INFORMAL



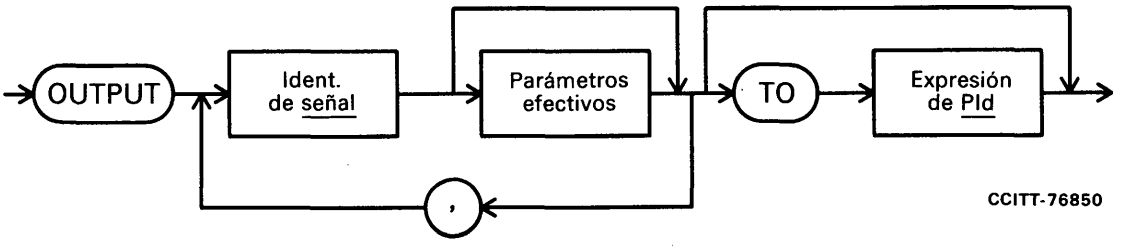
SENTENCIA DE TERMINADOR



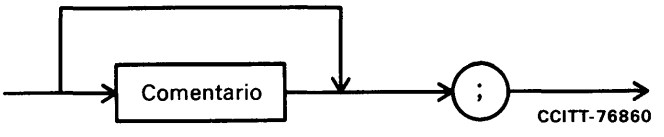
TERMINADOR



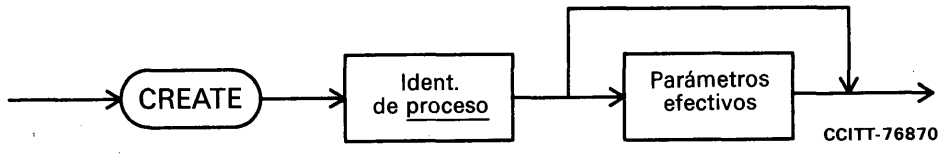
SALIDA



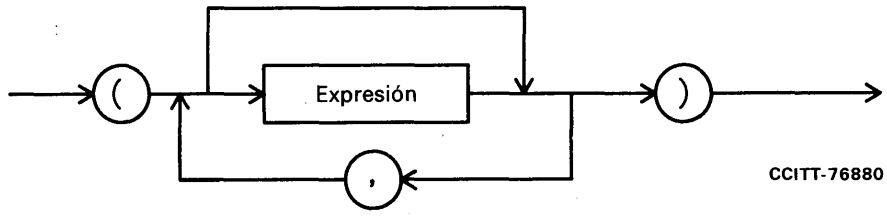
FIN



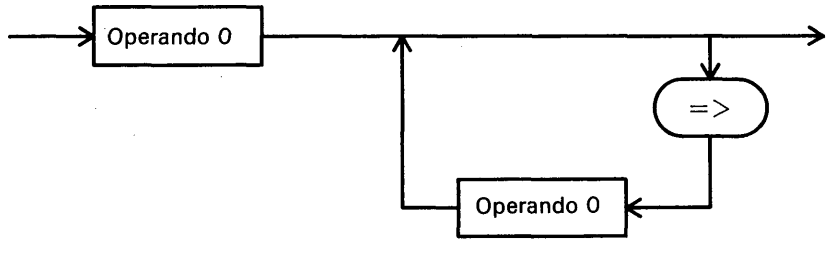
PETICIÓN DE CREAR



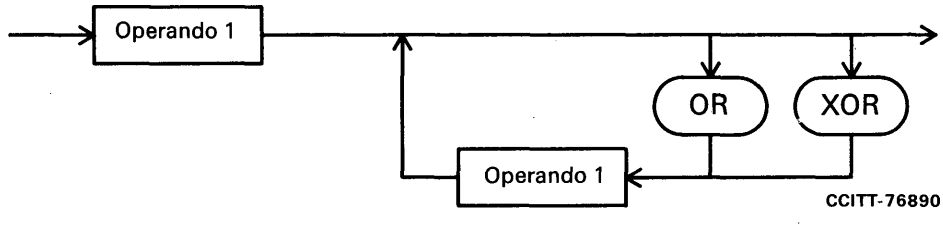
PARÁMETROS EFECTIVOS



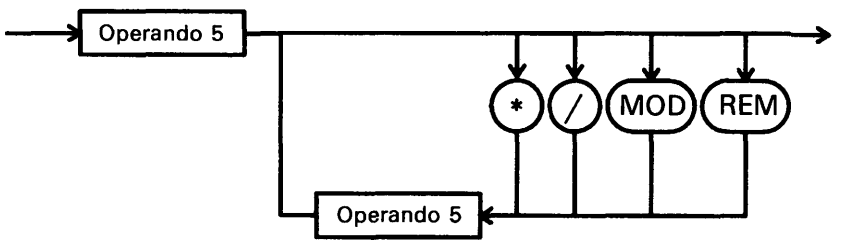
EXPRESIÓN



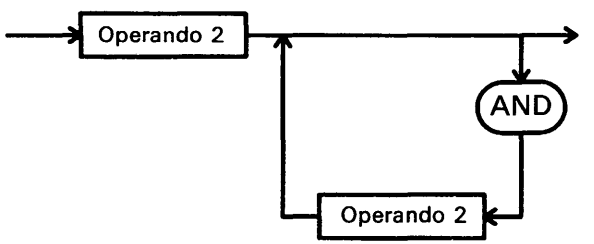
OPERANDO 0



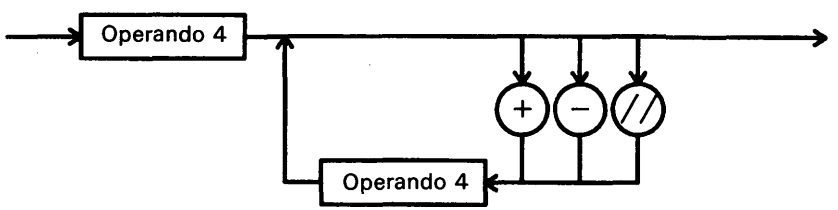
OPERANDO 4



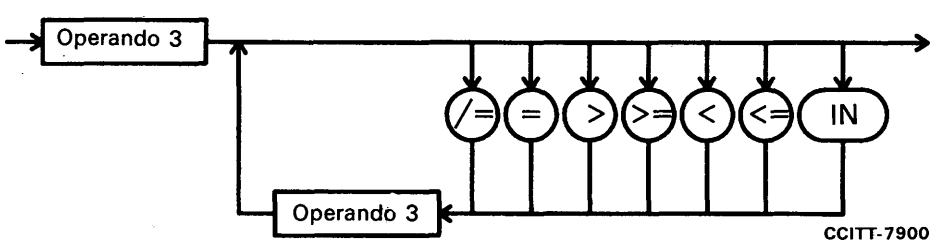
OPERANDO 1



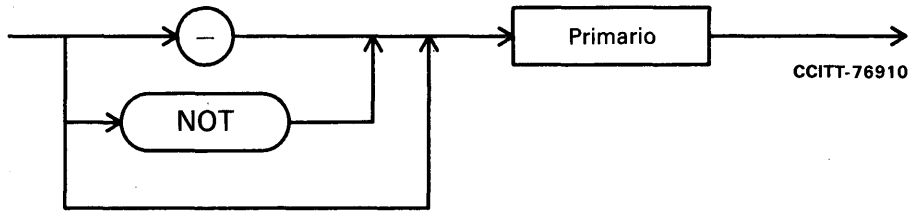
OPERANDO 3



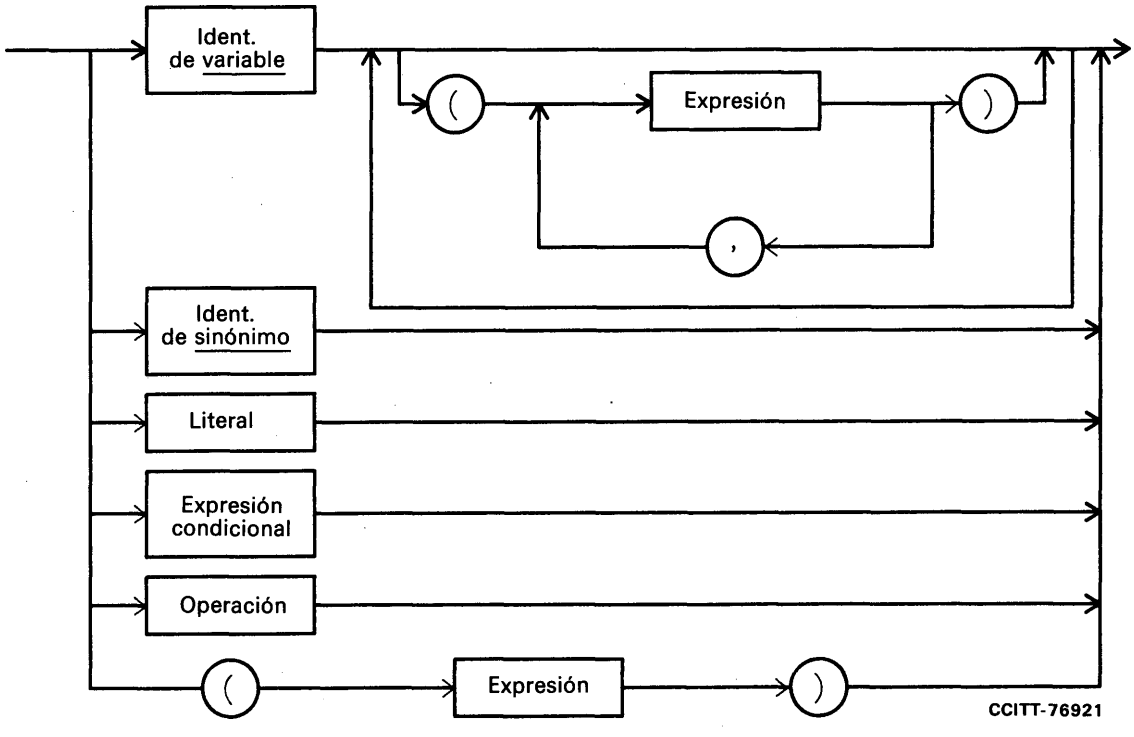
OPERANDO 2



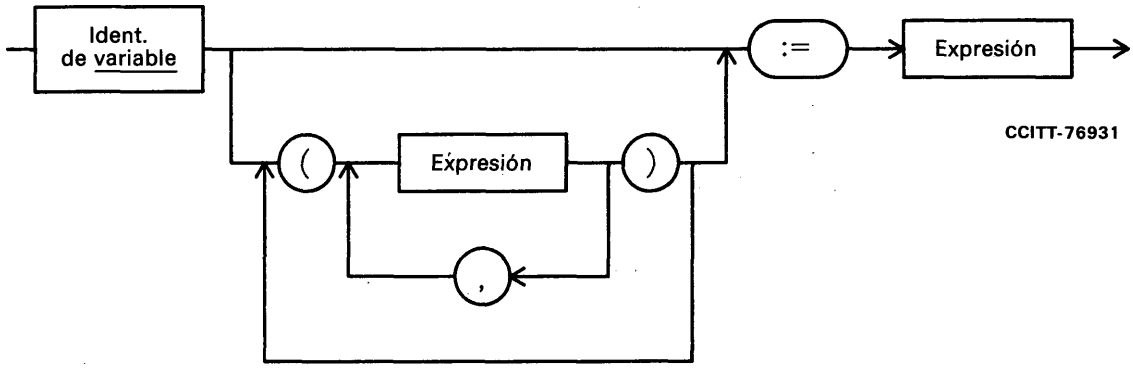
OPERANDO 5



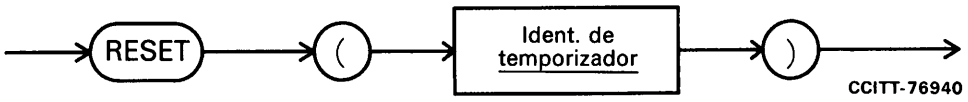
PRIMARIO



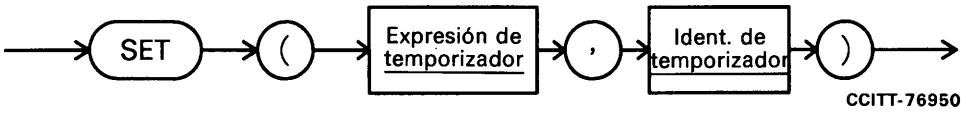
SENTENCIA DE ASIGNACIÓN



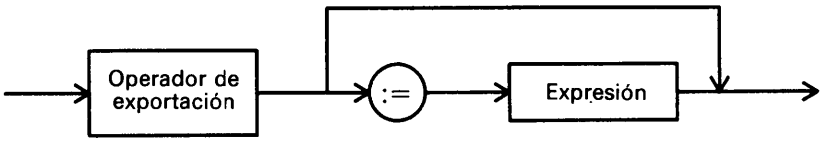
SENTENCIA DE REPONER



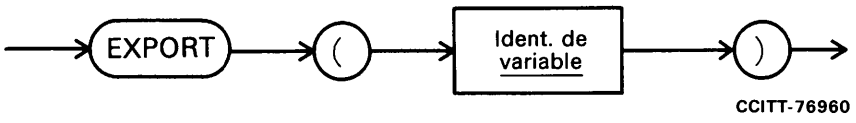
SENTENCIA DE PONER



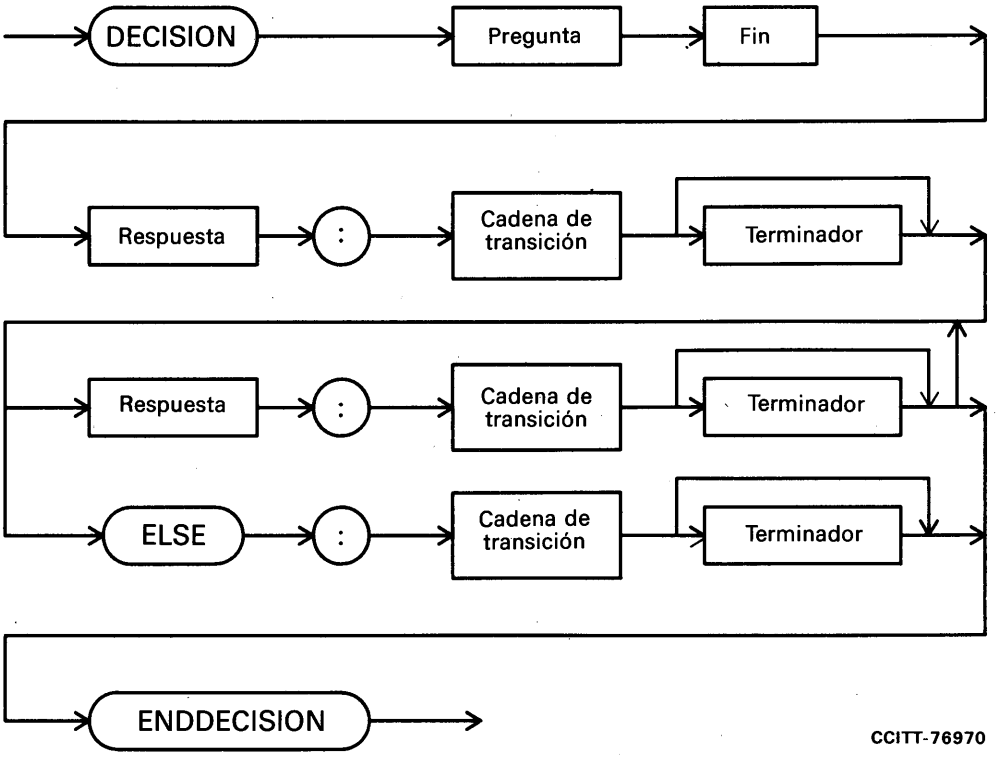
SENTENCIA DE EXPORTACIÓN



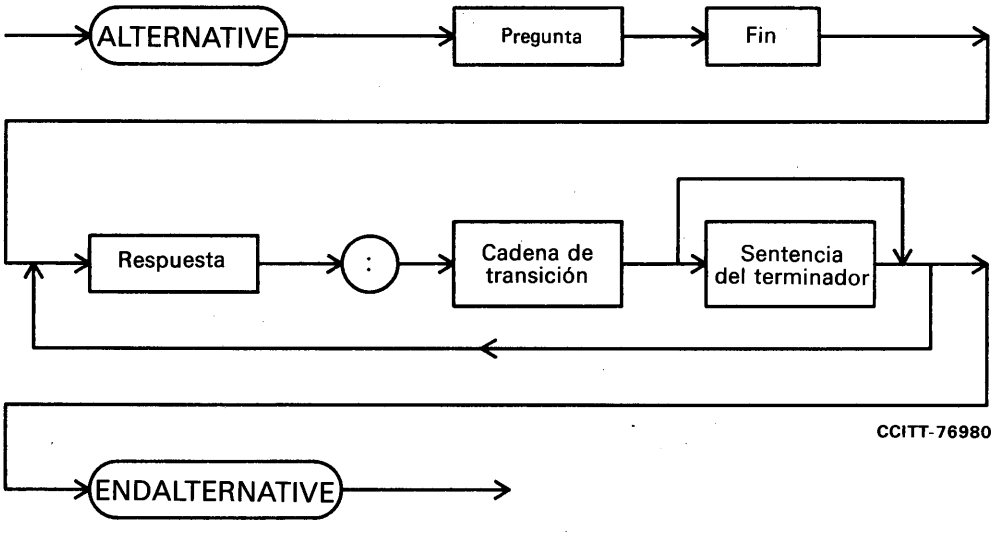
OPERADOR DE EXPORTACIÓN



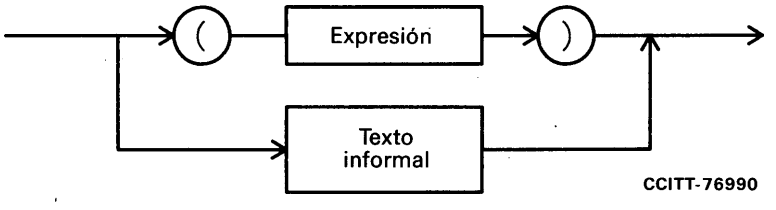
DECISIÓN



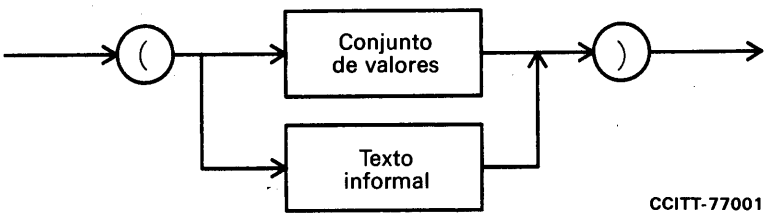
OPCIÓN



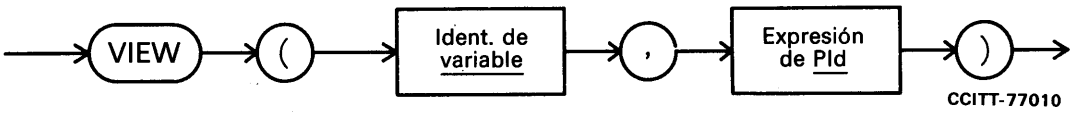
PREGUNTA



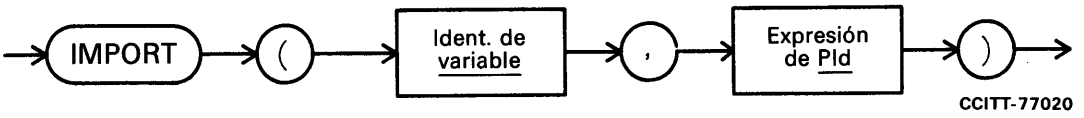
RESPUESTA



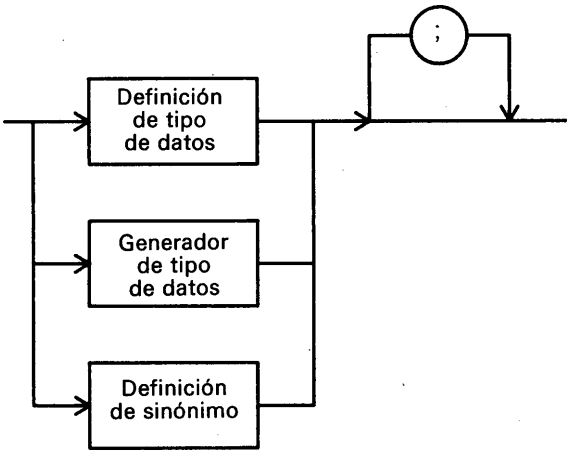
OPERADOR DE VISIÓN



OPERADOR DE IMPORTACIÓN

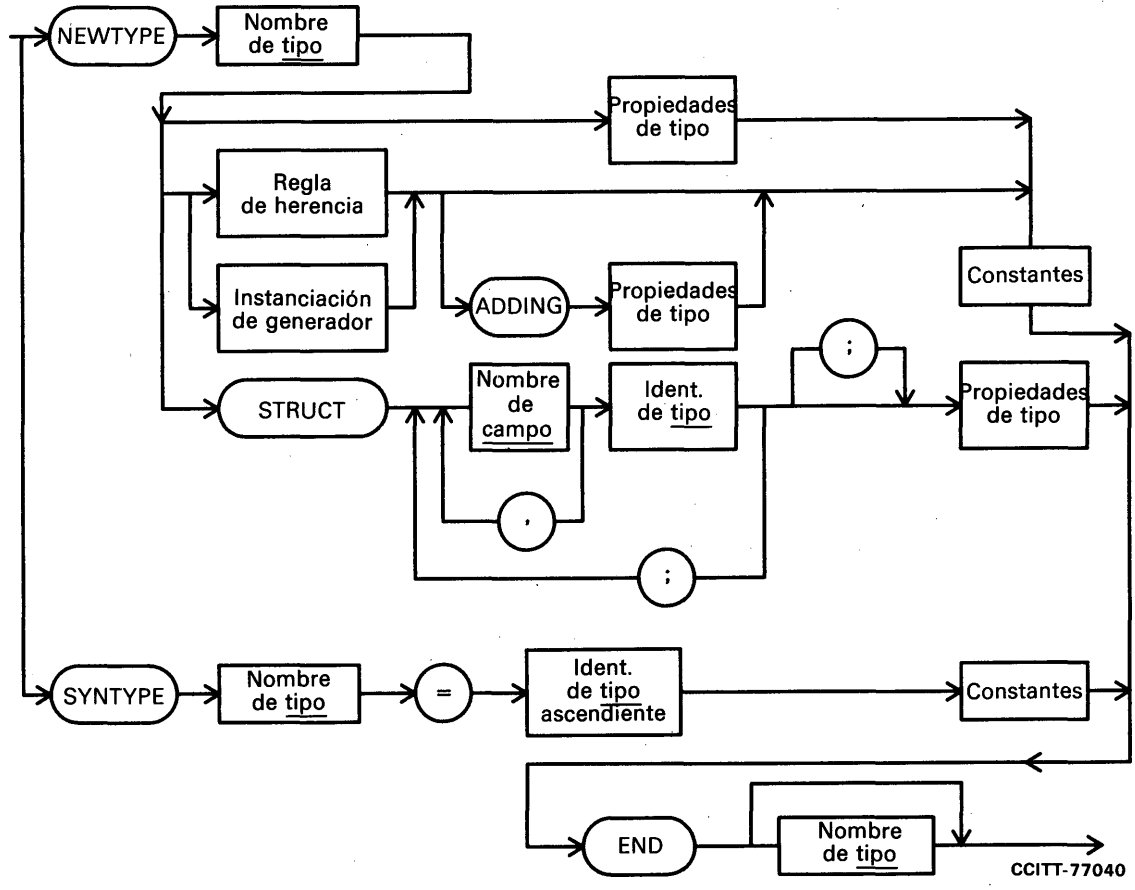


DEFINICIÓN DE DATOS

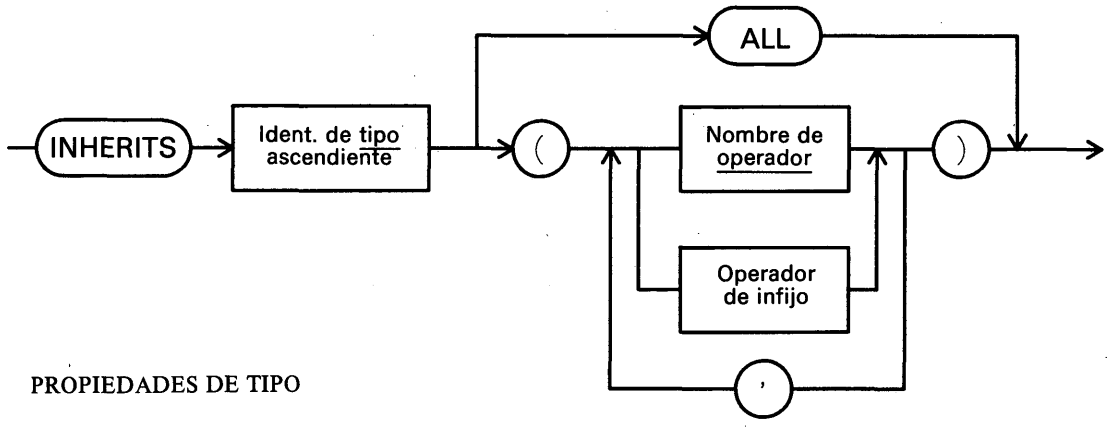


DEFINICIÓN DE SINÓNIMO

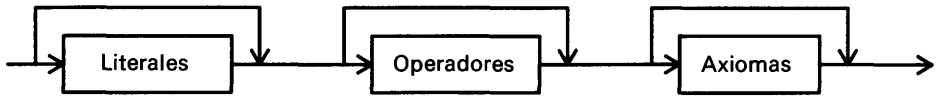
DEFINICIÓN DE TIPO DE DATOS



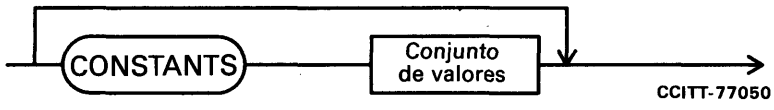
REGLAS DE HERENCIA



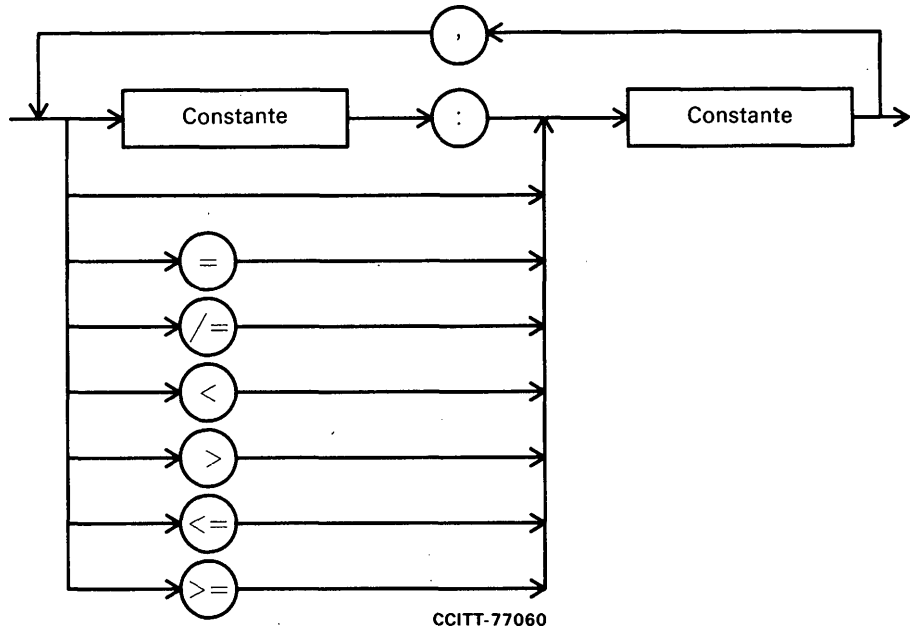
PROPIEDADES DE TIPO



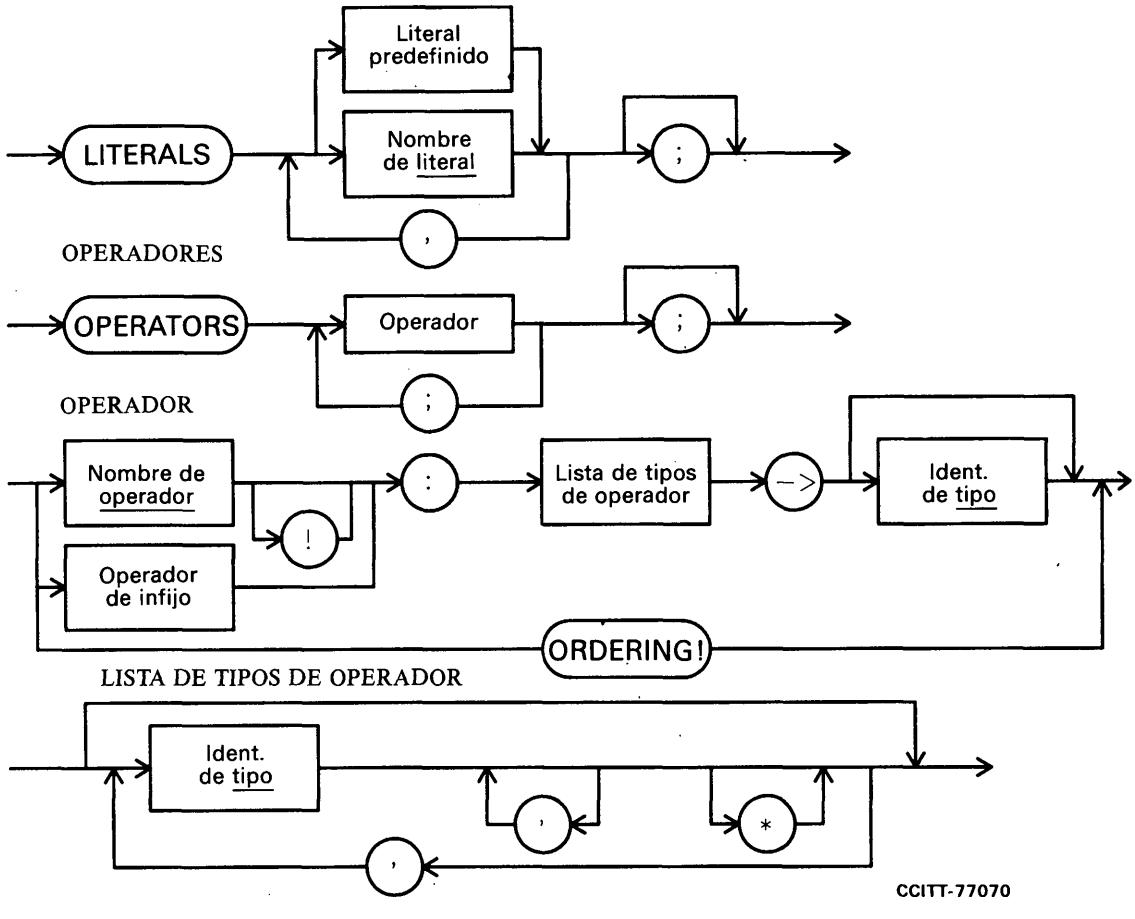
CONSTANTES



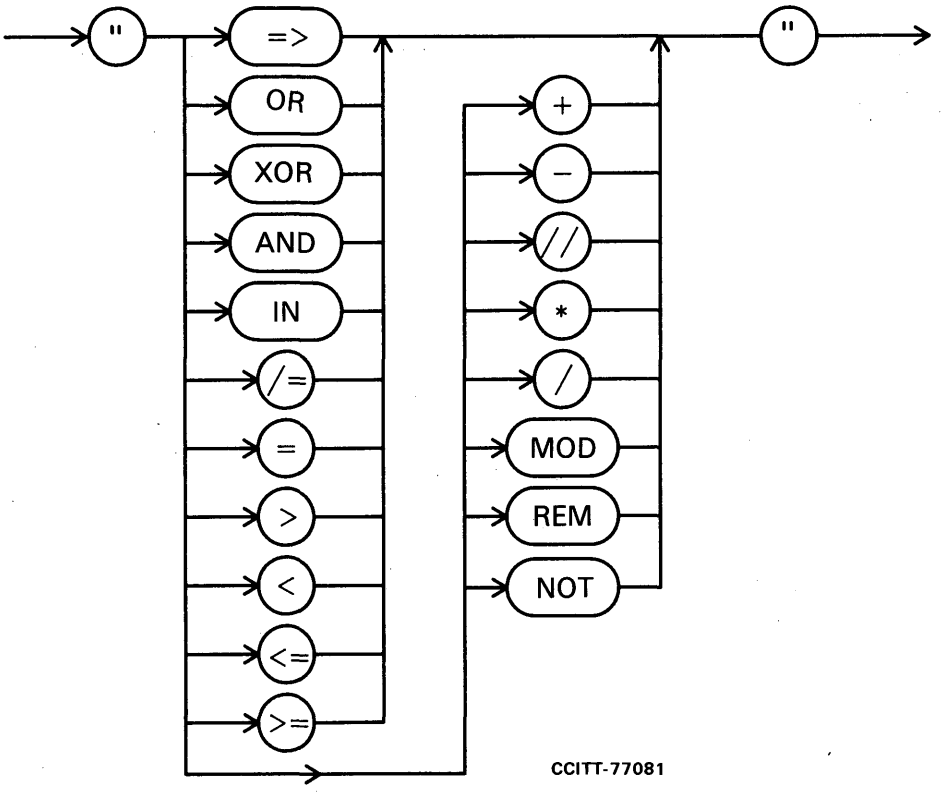
CONJUNTO DE VALORES



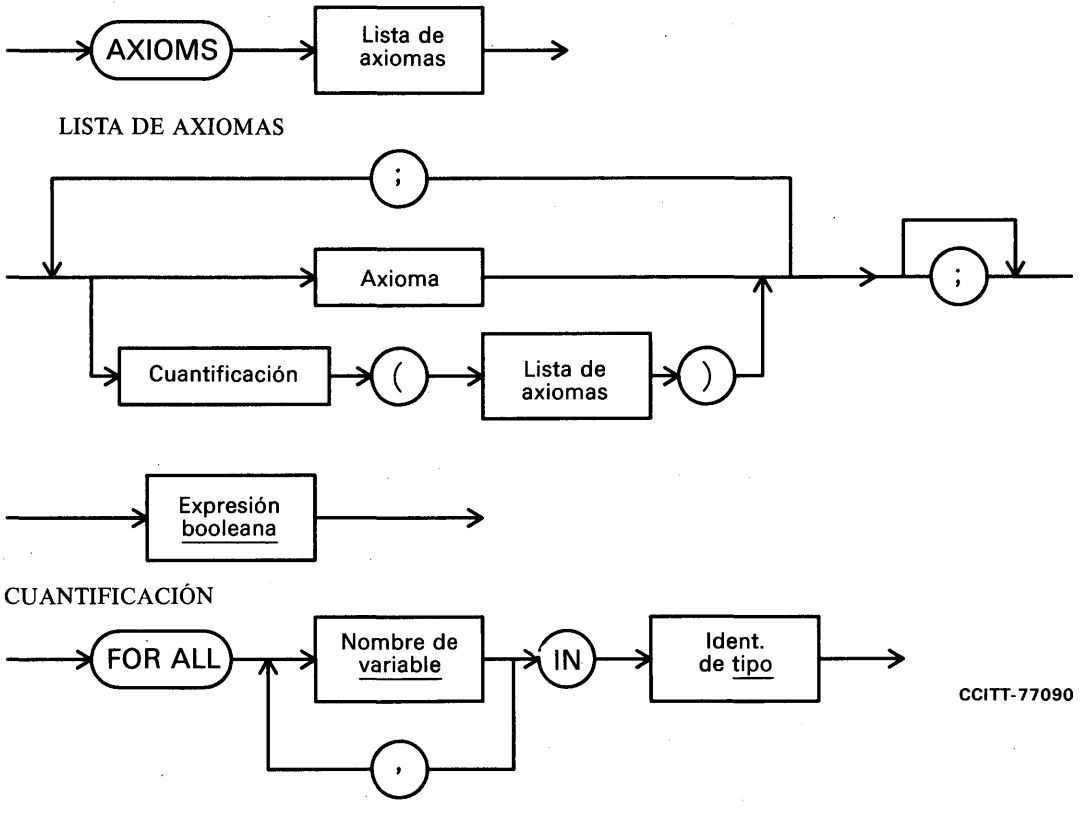
LITERALES



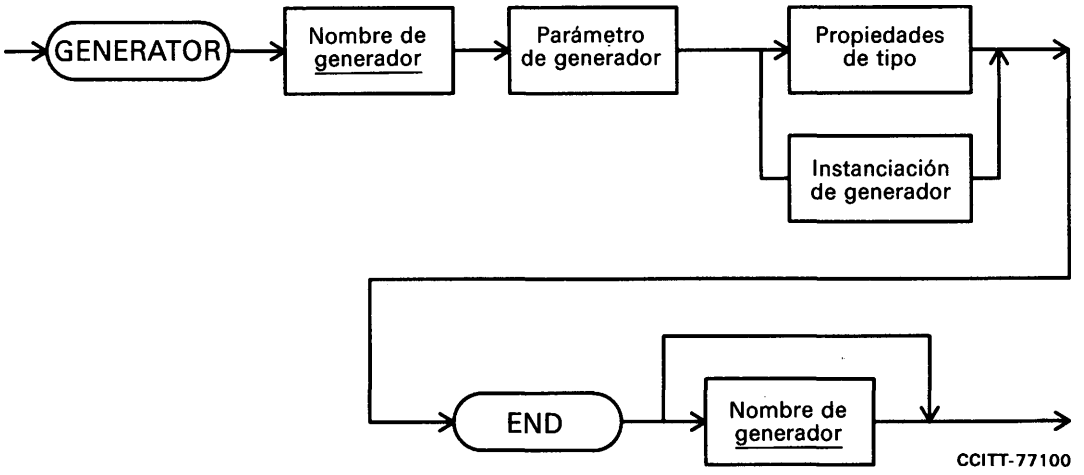
OPERADOR DE INFIJO



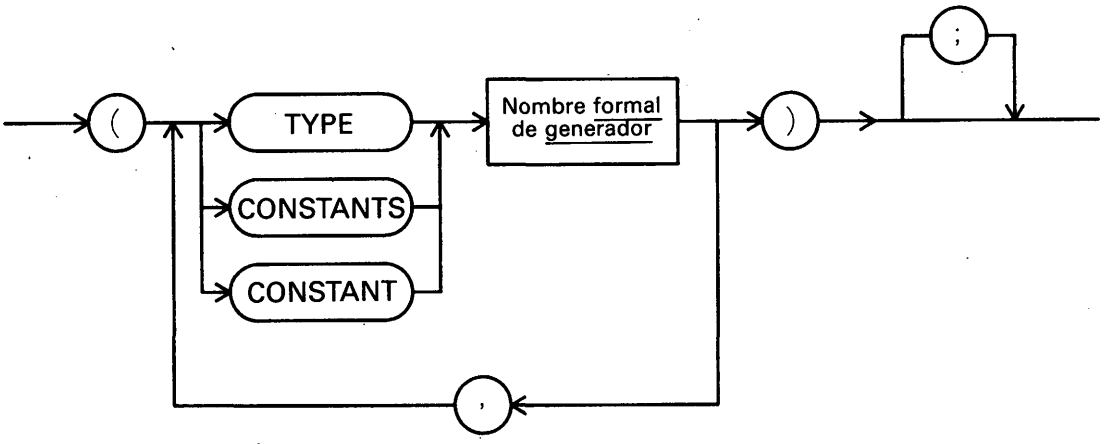
AXIOMAS



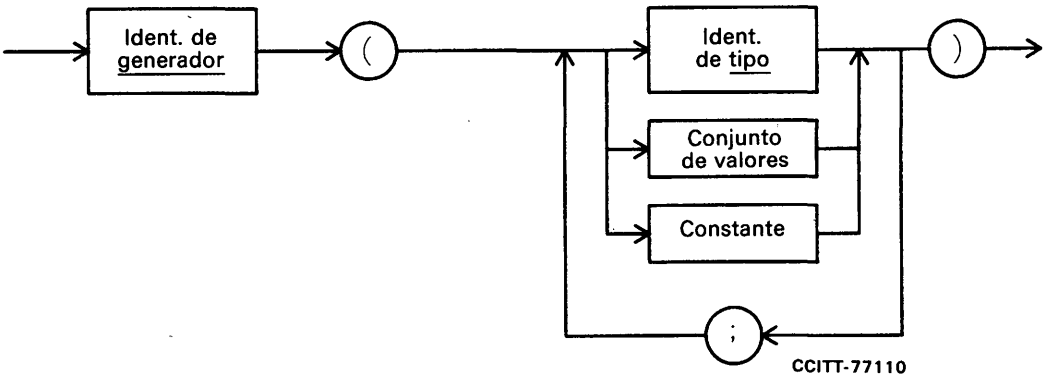
GENERADOR DE TIPOS DE DATOS



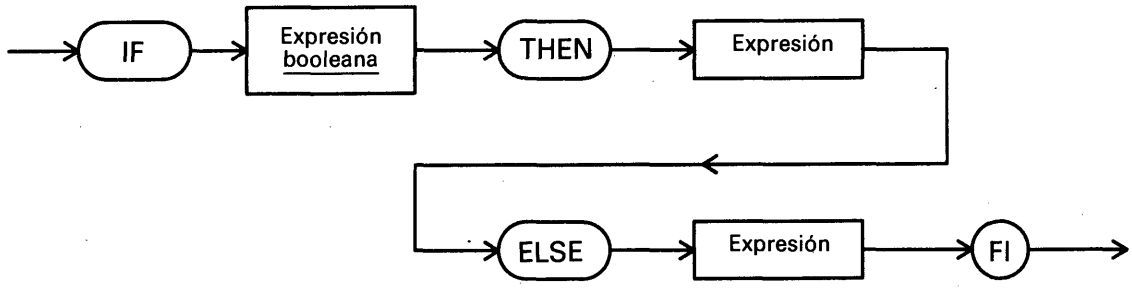
PARÁMETROS DE GENERADOR



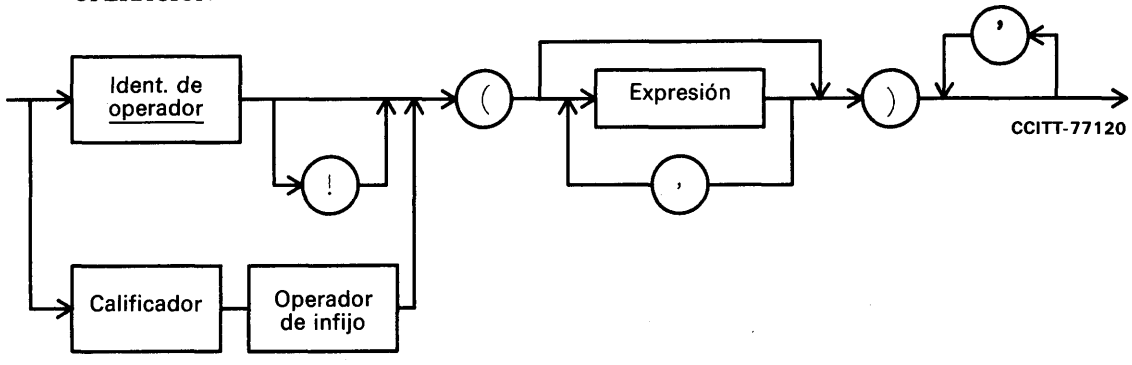
INSTANCIACIÓN DEL GENERADOR



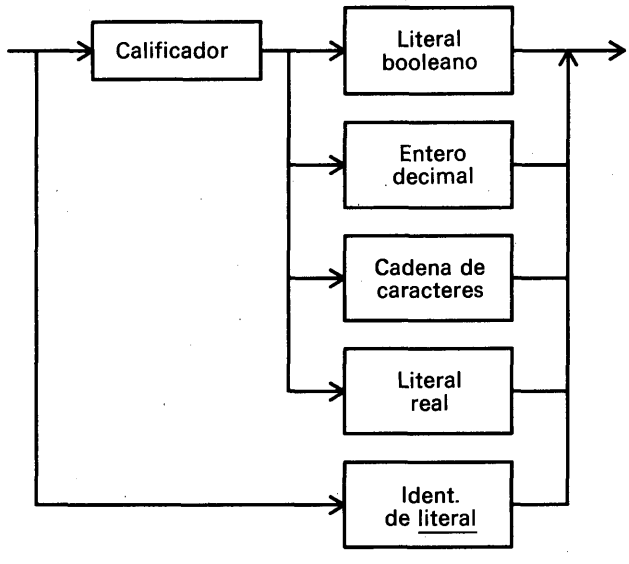
EXPRESIÓN CONDICIONAL



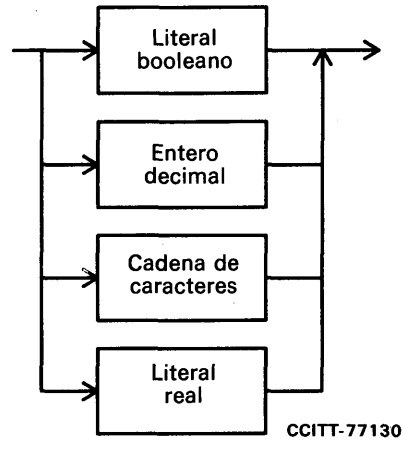
OPERACIÓN



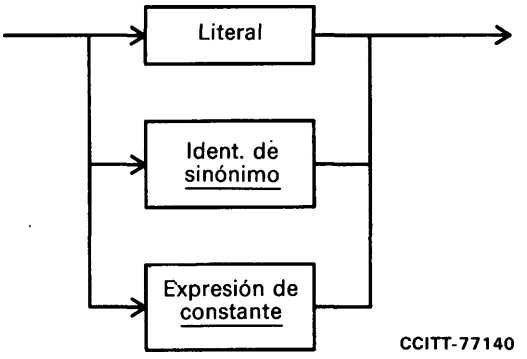
LITERAL



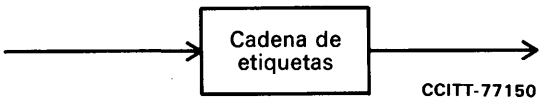
LITERAL PREDEFINIDO



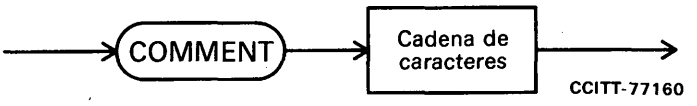
CONSTANTE



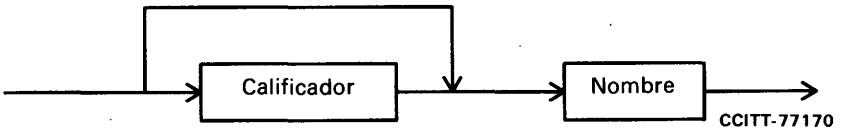
ETIQUETA



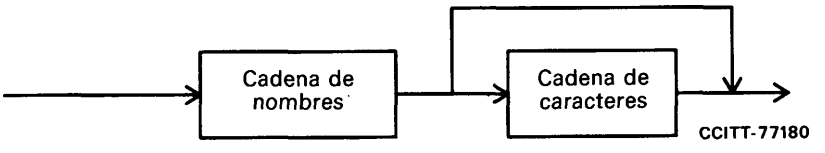
COMENTARIO



IDENT



NOMBRE



C2.2 *Reglas léxicas*

1) Todos los signos de puntuación (ejemplo, . ; ' : ! = ( )) y símbolos de operación (ejemplo +, −, *, ‹, › . . .) son unidades léxicas que pueden ocupar espacios.

2) Dos unidades léxicas deben ir separadas por uno o más espacios.

3) Las palabras clave pertenecen a la misma categoría léxica que la cadena de nombres, y se reservan.

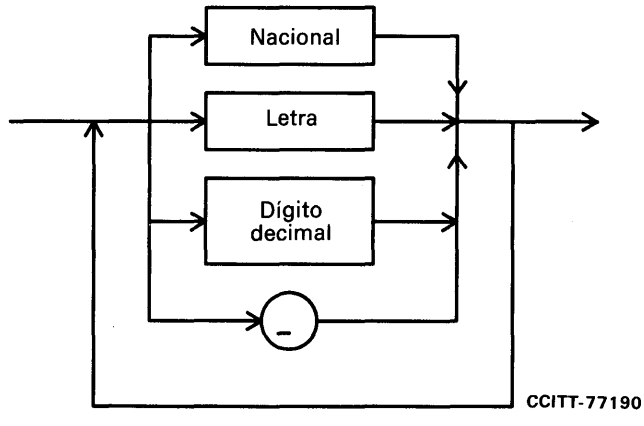
4) Fuera de las unidades léxicas varios espacios tienen el mismo «significado» que un espacio.

5) Los caracteres de tabulación (VT, HT, CR, BS . . .) pueden considerarse como espacios.

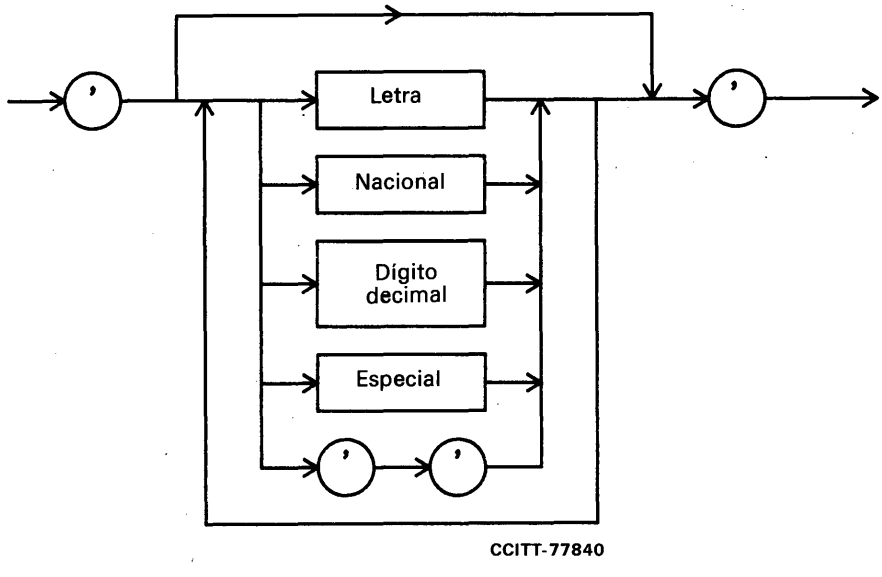
6) Todas las letras y nacionales se interpretan siempre como si fuesen mayúsculas, excepto dentro de una cadena de caracteres.

7) Siempre que puedan producirse espacios, pueden insertarse comentarios delimitados por ' /*' y '*/'; estos comentarios tienen el mismo significado que un espacio. El comentario no debe contener la secuencia especial: ' */'.

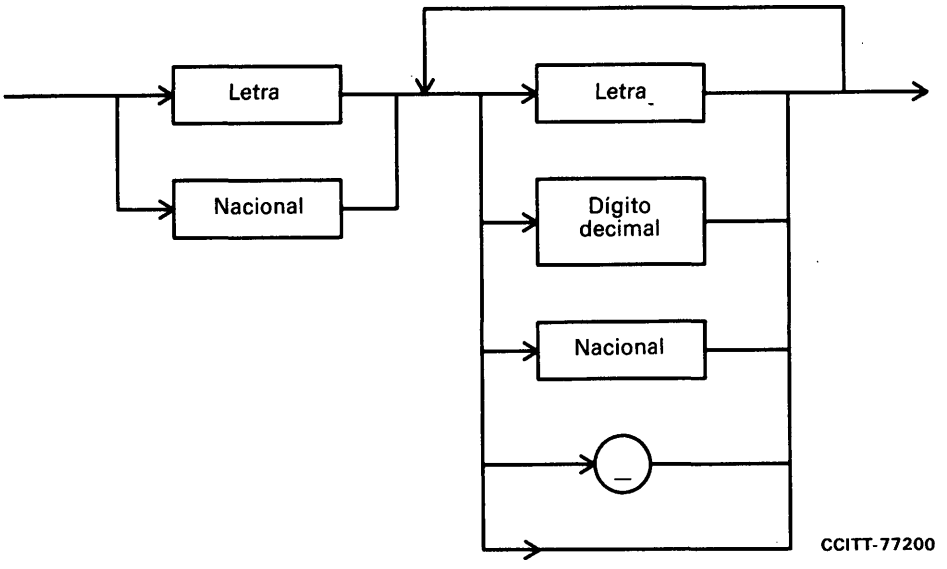
CADENA DE ETIQUETAS



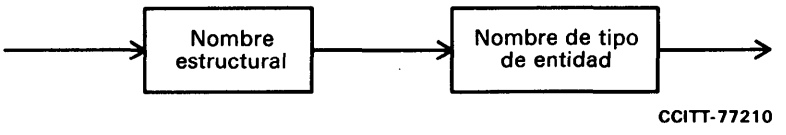
CADENA DE CARACTERES



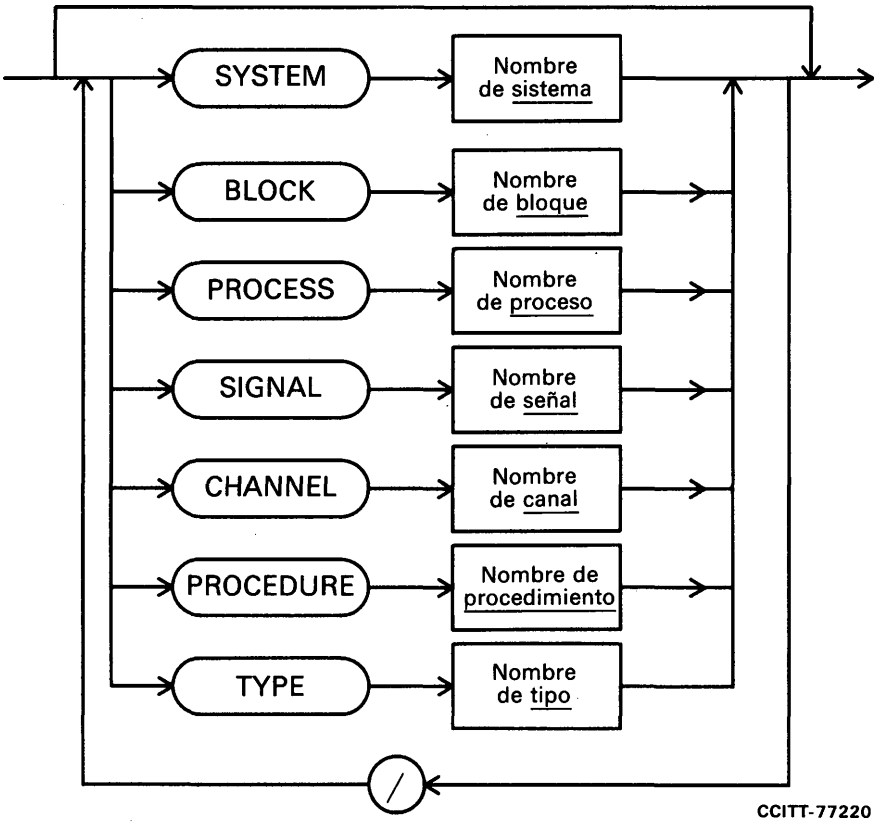
CADENA DE NOMBRES



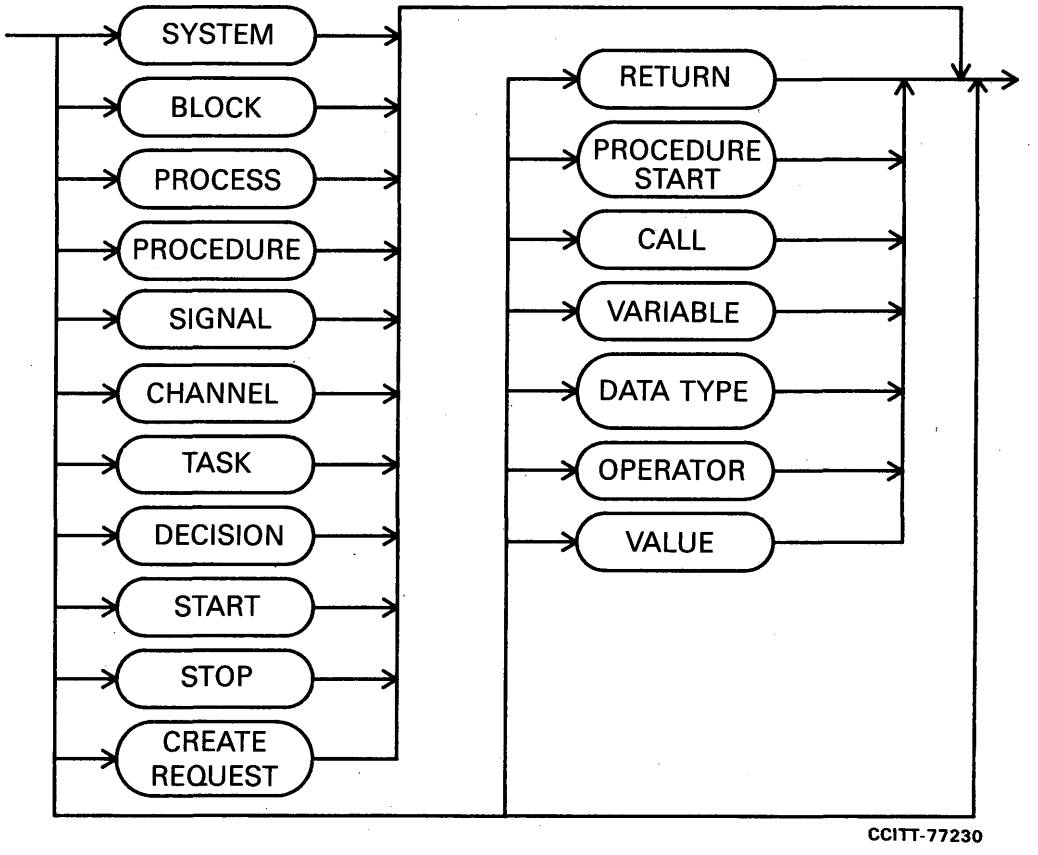
CALIFICADOR



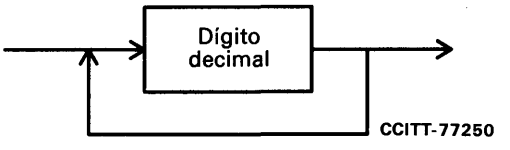
NOMBRE ESTRUCTURAL



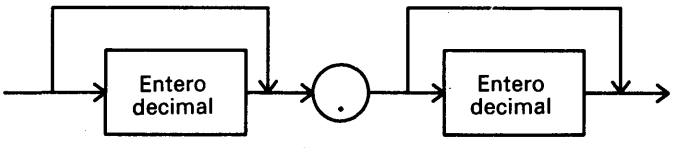
NOMBRE DE TIPO DE ENTIDAD



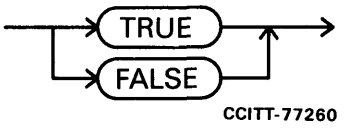
ENTERO DECIMAL



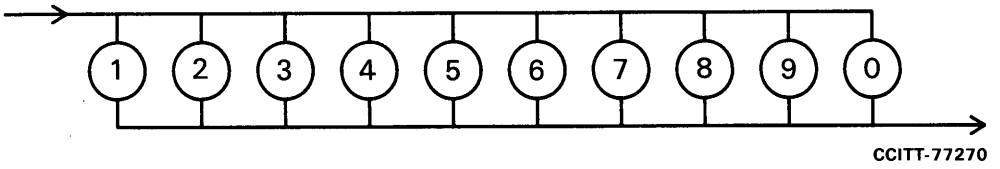
LITERAL REAL



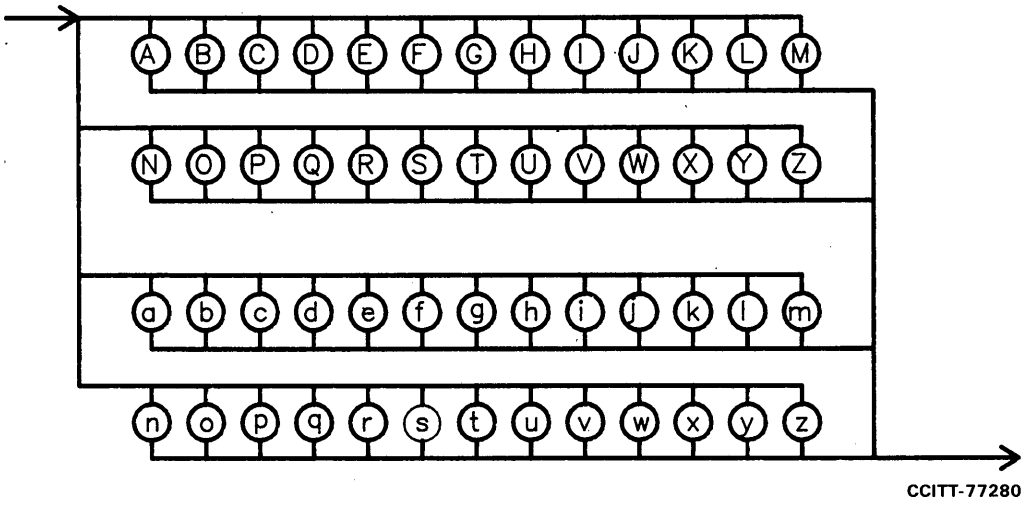
LITERAL BOOLEANO



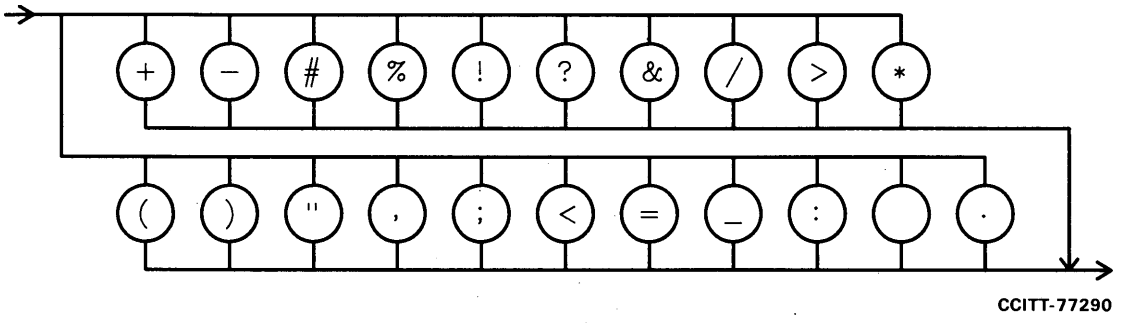
DÍGITO DECIMAL



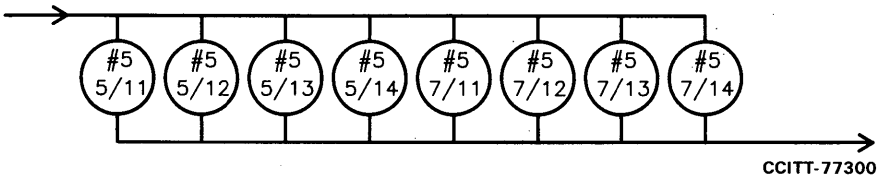
LETRA



ESPECIAL

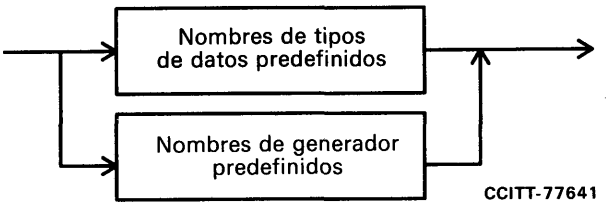


NACIONAL

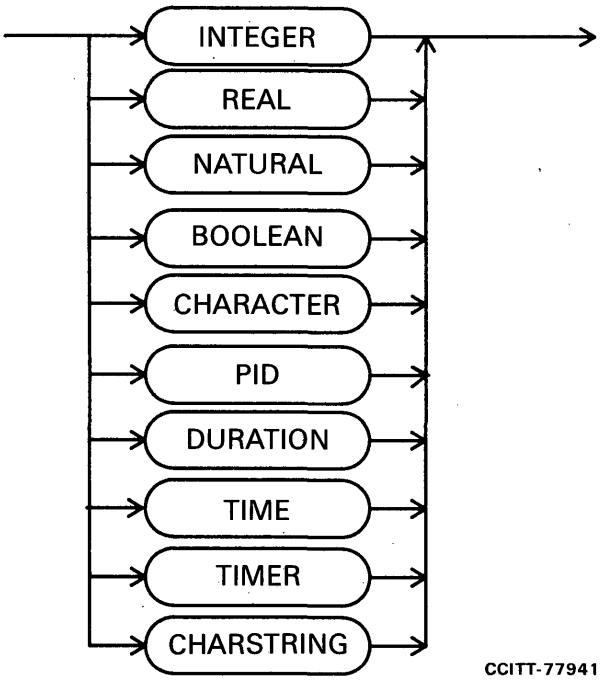


Nota – Las posiciones referenciadas corresponden a las posiciones del Alfabeto Internacional N.º 5 del CCITT reservadas para uso nacional.

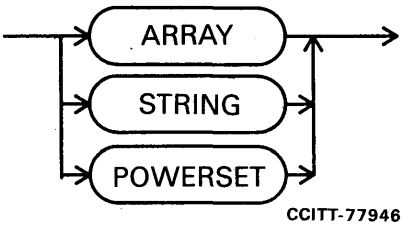
NOMBRE DE TIPO PREDEFINIDO



NOMBRES DE TIPOS DE DATOS PREDEFINIDOS



NOMBRES DE GENERADOR PREDEFINIDOS



C2.3 *Palabras reservadas utilizadas en el PR*

SYSTEM
ENDSYSTEM
BLOCK
ENDBLOCK
SUBSTRUCTURE
ENDSUBSTRUCTURE
SPLIT
INTO
SUBBLOCKS
CHANNELS
SIGNALS
IN
CHANNEL
FROM
ENV
TO
WITH
IMPORTEDVALUES
SIGNAL
INCOMING
OUTGOING
BLOCKS
DECISION
ELSE
ENDDECISION
ALTERNATIVE
ENDALTERNATIVE
VIEW
EXPORT
IMPORT
CREATEREQUEST
CALL
DATATYPE
VALUE
PROCESS
ENDPROCESS
PROCEDURE
ENDPROCEDURE
DCL
START
STATE
INPUT
SAVE
PROVIDED
PRIORITY
REVEALED
EXPORTED
VIEWED
IMPORTED
FPAR
EXPORTED/IMPORTED
IN/OUT
IN
TASK
NEXTSTATE
JOIN
STOP
RETURN
OUTPUT
CREATE
COMMENT
RESET
SET
EXPORT
PROCEDURESTART
VARIABLE
OPERATOR

Nota 1 – La llamada macro puede ponerse en cualquier diagrama utilizando la sintaxis:

Nombre de macro MACRO;

La ampliación de macro es un trozo de un programa LED/PR que comienza por:

Nombre de macro MACRO EXPANSIÓN;

y termina por:

Nombre de macro ENDMACRO;

en la última sentencia el nombre de macro no es obligatorio.

Nota 2– En el diagrama de definición de bloque es válida la regla siguiente:

- Si no existe definición de subestructura de bloque debe haber al menos una definición de proceso (que pueda definirse explícitamente en otro módulo).
- Si existe la definición de subestructura de bloque, existe una definición de subestructura de proceso para cada una de las definiciones de proceso contenidas en la definición de bloque.

ANEXO D

(a las Recomendaciones Z.100 a Z.104)

Directrices para el usuario del LED

Las presentes directrices para el usuario pueden dividirse en tres partes.

La primera parte comprende el índice (D.0), el prefacio (D.1), la introducción (D.2) y los campos de aplicación del LED (D.3) que dan información sobre el LED, el contenido y los sectores de aplicación.

La segunda parte abarca la explicación general, los conceptos básicos del LED (D.4) y la estructuración del LED (D.5), facilitando información sobre el modo de modelar los sistemas en el LED y sobre los conceptos para tal fin. Se trata de una parte general que no da detalles sobre formas concretas del LED.

La última parte consiste en directrices para la representación de sistemas en el LED/GR (D.6), directrices para la representación de sistemas en el LED/PR (D.7), mapeados entre el LED/GR, el LED/PR y el CHILL (D.8), y ejemplos de aplicación del LED (D.9) con directrices sobre el empleo de las dos formas existentes del LED, es decir, el LED/GR y el LED/PR. Por último, figura una sección sobre los instrumentos del LED (D.10).

ÍNDICE

D.1 *Prefacio*

El lenguaje de especificación y de descripción del CCITT, conocido por la sigla LED, fue definido por primera vez en las Recomendaciones Z.101 a Z.103 en 1976 (Libro Naranja, Tomo VI.4), ampliado más tarde en las Recomendaciones Z.101 a Z.104 en 1980 (Libro Amarillo) y ampliado de nuevo y reestructurado en las Recomendaciones Z.100-Z.104 en 1984 (Libro Rojo, presente fascículo).

Para la Comisión de Estudio XI resulta evidente que se necesitan Directrices para el usuario a fin de facilitar la aplicación del LED a una amplia gama de sistemas de conmutación para telecomunicaciones. La finalidad de las Directrices para el usuario consiste en ayudar a los usuarios a comprender las Recomendaciones relativas al LED y su aplicación en distintos sectores. Es difícil preparar un conjunto completo de Directrices para el usuario porque el sector de aplicación del propio LED está aún en evolución. Por consiguiente, las Directrices para el usuario contenidas en este fascículo requerirán perfeccionamiento y ampliación en el próximo Periodo de Estudios de 1984-1988, y las Contribuciones a las Directrices para el usuario basadas en la experiencia práctica obtenida con el LED serán recibidas con agrado por el CCITT.

Resulta evidente ahora, en 1984, que el LED es utilizado cada vez más ampliamente por el CCITT y sus organizaciones miembros, y que la gama de aplicaciones del LED seguirá en aumento. Estas Directrices para el usuario se hallan destinadas a ayudar a las personas que se plantean el empleo del LED o comienzan a usarlo, complementando las Recomendaciones Z.100 a Z.104 sobre el LED con útiles consejos y valiosos ejemplos. Debe tenerse en cuenta que habrá cierta redundancia entre las Directrices para el usuario y las Recomendaciones; se cree que ello es conveniente para que las Directrices para el usuario sean autónomas y de fácil lectura. No obstante, las Recomendaciones son el documento decisivo.

D.2 *Introducción*

D.2.1 *Consideraciones generales*

El LED puede utilizarse para la especificación (especificar el comportamiento deseado de un sistema) y la descripción (describir el comportamiento real de un sistema). El LED se ha diseñado para realizar concretamente la especificación y descripción del comportamiento de los sistemas de conmutación para telecomunicaciones, pero también puede utilizarse en otras aplicaciones. En realidad, el LED se presta para todos los sistemas cuyo comportamiento pueda modelarse efectivamente por medio de Máquinas de Estado Finito Ampliadas y cuando el acento deba recaer en particular en los aspectos interactivos.

El LED puede ser también la base de una metodología de documentación que represente completamente la especificación o descripción de un sistema. En este contexto, el significado de especificación y descripción guarda relación con su uso en el ciclo de vida de un sistema. Ambas definen las propiedades funcionales de un sistema de una manera abstracta. La descripción comprenderá habitualmente algunos aspectos dependientes del diseño (por ejemplo, el tratamiento de errores) y será más completa en materia de detalles funcionales. Ambas deben estar relacionadas con el diseño del sistema concreto de una manera uniforme. En consecuencia, ambas sirven de especificaciones antes de la realización práctica del sistema, y de documentación (descripciones) después de ella.

El LED puede utilizarse para representar, con distintos niveles de detalle, las propiedades funcionales de un sistema, una función o una facilidad en forma de especificación o descripción. Las propiedades funcionales consisten en algunas propiedades estructurales (diagramas de interacción de bloque) así como de comportamiento. Por «comportamiento» se entiende el modo de reaccionar frente a las señales recibidas (entradas), esto es, mediante la ejecución de acciones, por ejemplo, envío de señales (salidas), respuesta a preguntas (decisiones) y realización de tareas.

Cabe que las especificaciones sean muy amplias y generales cuando una administración desea explorar las posibilidades de actualizar un sistema con nuevas características, nuevos servicios, nueva tecnología, etc., permitiendo al proveedor que ofrezca una amplia gama de soluciones en los diseños. Este tipo de especificación a menudo no es muy detallado. El otro extremo es una especificación en la que una administración solicita la sustitución de una central existente o una adición a la misma. Esa especificación tendrá probablemente un elevado nivel de detalle, debido a la necesidad de una especificación muy pormenorizada de los interfaces.

Una especificación y una descripción pueden ser idénticas. En todo caso, en un nuevo desarrollo es preferible que el diseño se derive de la especificación a fin de garantizar su cumplimiento.

Por lo general, los proveedores escriben las descripciones en respuesta a una especificación (aunque pueden escribirse para describir sistemas que el proveedor desea vender). Una descripción tendrá habitualmente un nivel de detalle superior al de una especificación, debido a la necesidad de describir el comportamiento pormenorizado del sistema.

Cabe observar también que el LED ofrece medios para describir un sistema con diversos grados de formalidad.

Primeramente, es posible describir un sistema utilizando las construcciones LED con un lenguaje natural asociado. La descripción resultante es significativa únicamente para un lector que conozca el contexto, pero no para una máquina. Son limitadísimas las verificaciones que pueden efectuarse en forma automática.

En segundo lugar, es posible asociar a las construcciones LED declaraciones formales consistentes en elementos de tipos definidos y operadores relativos a esos elementos. Las propiedades de estos elementos no quedan especificadas: un ejemplo es «Conexión A-B», en donde A y B son el tipo de abonado, y conexión es una operación permitida para este tipo. La descripción resultante es significativa para los lectores que conocen el significado de los operadores utilizados. Una máquina puede comprender la descripción hasta cierto nivel y puede efectuar verificaciones en base a ella, pero no puede hacer verificaciones completas ni «implementar» el sistema porque las propiedades de los operadores son desconocidas.

En tercer lugar, es posible indicar también todas las propiedades de todos los operadores. En este último caso la descripción es completamente formal y una máquina puede hacer todas las verificaciones e implementar conceptualmente el sistema descrito.

Según cual sea la meta perseguida, la descripción puede adaptarse a las necesidades del usuario utilizando estos diferentes niveles de formalismo. Como es natural, cuanto más formal es la descripción, tanto más engorrosa es su lectura por el ser humano.

D.2.2 *Formas sintácticas del LED*

El LED, tal como está definido en las Recomendaciones Z.100-Z.104, es un lenguaje único que tiene dos sintaxis distintas, basadas ambas en el mismo modelo semántico. Una, llamada LED/GR (SDL *G*raphical *R*epresentation – Representación Gráfica en el LED), está basada en un conjunto de símbolos y reglas gráficos normalizados. La otra, llamada LED/PR (SDL textual *P*hrase *R*epresentation – Representación de frases textuales del LED), está fundada en declaraciones del tipo de programa. Ambas representan los mismos conceptos del LED.

D.2.3 *El LED basado en un modelo de Máquina de Estado Finito Ampliada*

En la aplicación del LED, el sistema por especificar se representa por un número de máquinas abstractas interconectadas. Una especificación completa requiere:

1) la definición de la estructura del sistema en términos de las máquinas y su interconexión;
2) el comportamiento dinámico de cada máquina en términos de sus interacciones con las otras máquinas;
3) el entorno y las operaciones respecto a los datos asociados a las interacciones.

El comportamiento dinámico se describe con ayuda de modelos que definen los mecanismos de funcionamiento de estas máquinas abstractas y la comunicación entre máquinas. La máquina abstracta utilizada en el LED es una extensión de la Máquina de Estado Finito (MEF) determinista. La MEF tiene una memoria de estado interno finita y opera con un conjunto discreto y finito de entradas y salidas. Para cada combinación de entrada y estado, la memoria define una salida y el estado siguiente. Habitualmente se considera que las transiciones de un estado a otro toman un tiempo cero.

Una limitación de la MEF radica en el hecho de que toda la información que hace falta almacenar debe estar representada en forma de estados explícitos. Aunque es posible representar de esta manera la mayoría de los sistemas, no siempre resulta práctico. Puede que haya muchos valores por recordar que son significativos para la secuencia futura pero que no contribuyen mucho a la comprensión global del sistema. Esta información no debe formar parte del espacio del estado explícito, pues abarrotaría la presentación. Para aplicaciones así se puede ampliar la MEF con un almacenamiento auxiliar y operaciones auxiliares respecto a ese almacenamiento. La información de dirección y los números secuenciales son ejemplos de la información que se presta para almacenamiento en memoria auxiliar.

Las Recomendaciones relativas al LED definen dos operaciones auxiliares que pueden incluirse en las transiciones de la Máquina de Estado Finito Ampliada (MEFA) esto es, decisiones y tareas. Las «decisiones» inspeccionan los parámetros asociados a entradas e información en memoria auxiliar cuando tal información es importante para la secuencia de la máquina principal. Las «tareas» realizan funciones tales como el cómputo, la operación respecto a la memoria auxiliar y la manipulación de parámetros de entrada y salida.

En el LED, las interacciones de las máquinas están representadas por señales, o sea que la MEFA recibe señales como entrada y genera señales como salida. Las señales se componen de un identificador de señal único y, facultativamente, un conjunto de parámetros. El LED prevé la posibilidad de un tiempo de transición distinto de cero, y define un mecanismo de fila de espera conceptual basada en el método de «el primero que entra es el primero que sale» para las señales que llegan a una máquina mientras ésta ejecuta una transición. Las señales se consideran una por vez, por orden de llegada.

D.3 *Aplicabilidad del LED*

La figura D-1 muestra una gama de posibles utilizaciones del LED en el campo de la compra y el suministro de sistemas de conmutación para telecomunicaciones.

En esa figura, los rectángulos representan grupos funcionales típicos cuyos nombres precisos pudieran variar de una organización a otra, pero cuyas actividades serían típicas de muchas administraciones y fabricantes. Cada flecha (línea de flujo) representa un conjunto de documentos que pasan de un grupo funcional a otro; el LED puede utilizarse como parte de cada uno de esos conjuntos de documentos. Esta figura tiene carácter puramente ilustrativo y no se pretende que sea definitiva ni exhaustiva.

Los sectores de aplicación son aquéllos modelados efectivamente por máquinas de estado finito ampliadas en comunicación, por ejemplo, conmutación telefónica, télex y de datos, sistemas de señalización (por ejemplo, sistema de señalización N.º 7), interfuncionamiento entre sistemas de señalización, protocolos de datos e interfaces de usuario (LHM).

Al considerar en particular los sistemas de conmutación SPC, figuran los siguientes que pueden ser documentados utilizando el LED: el tratamiento de las comunicaciones (por ejemplo, tratamiento de llamadas, encaminamiento, señalización, tasación, etc.), mantenimiento y reparación de averías (por ejemplo, alarma, liberación automática en caso de fallo, configuración del sistema, pruebas periódicas, etc.), control del sistema (por ejemplo, control de sobrecarga) e interfaces hombre-máquina. En el § D.9 pueden verse ejemplos de aplicación del LED.

La especificación de protocolos utilizando el LED se considera en las Recomendaciones de la Serie X del CCITT, y las directrices auxiliares para el usuario sobre la aplicación del LED en este sector figuran en anexo a dichas Recomendaciones.

D.4 *Explicación general y conceptos del LED*

Este punto expone los conceptos del LED sin dar detalles sobre las formas concretas, esto es, LED/GR y LED/PR, detalles que aparecen en los § D.6 y D.7 respectivamente.

D.4.1 *Descripción general del LED*

El LED es un medio para representar las propiedades funcionales, conforme se especifican y realizan en un sistema de telecomunicación.

La especificación o descripción de un sistema abarca dos grandes categorías de información. La primera categoría comprende la especificación funcional o la descripción funcional. La segunda categoría comprende los parámetros más generales del sistema y detalles tales como información relativa a las condiciones ambientales (límites de temperatura, humedad, etc.), límites de transmisión y grado de servicio de la central, que no están representados en el LED.

Uno de los principales objetivos que perseguía al desarrollar el LED fue conseguir que los usuarios pudiesen tomar sistemas grandes y descomponerlos en partes lo suficientemente pequeñas para poder ser comprendidas fácilmente por los usuarios del LED (tanto el autor como el lector). Por ello, en el LED cada sistema se compone de bloques. Los bloques están conectados entre sí por canales que muestran el trayecto de comunicación entre los bloques. Estos bloques y canales pueden subdividirse a su vez en canales y bloques.

En el nivel más bajo (y a menudo en niveles superiores), los bloques contienen uno o más procesos. Los procesos son las Máquinas de Estado Finito Ampliadas en comunicación que se emplean para modelar el comportamiento dinámico del sistema. También los procesos pueden ser grandes y complejos. Los procedimientos permiten reducir más un proceso en partes manejables más pequeñas.

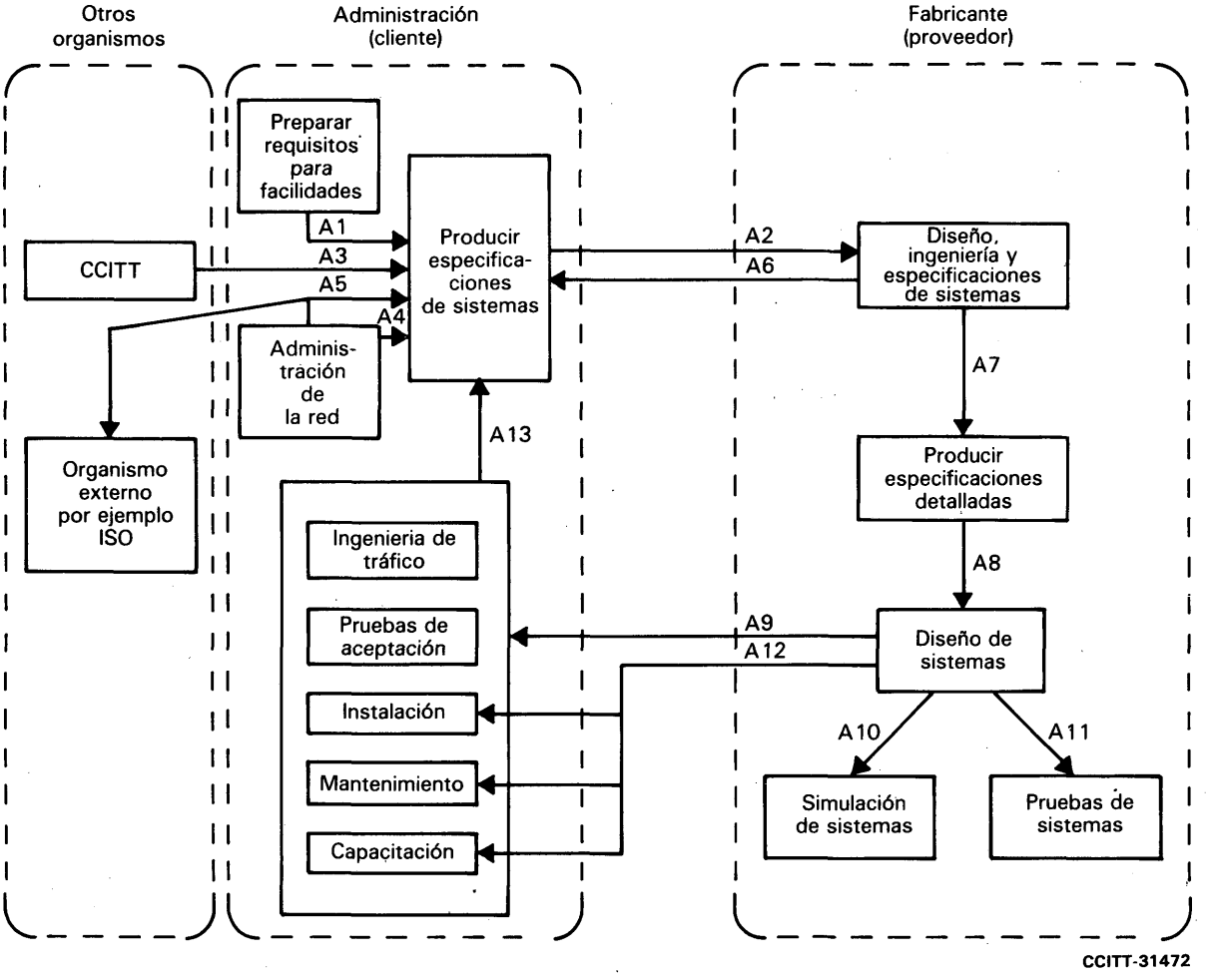
D.4.2 *Estructuración de sistemas en el LED*

D.4.2.1 *Generalidades*

En este capítulo se abordarán algunas construcciones LED que permiten establecer diferentes estructuras de un sistema, de manera que, comenzando por una descripción general de la estructura del sistema se puedan presentar cada vez más detalles de la estructura, que muestren la estructura interna de cada parte. La estructura mínima de un sistema en el LED es lo que se describe en la Recomendación Z.101; quiere decir que un sistema consiste en un conjunto de bloques conectados por canales, y los bloques contienen procesos (véase la figura D-2).

Para los sistemas que no necesitan una subdivisión ulterior, no hacen falta los conceptos de la Recomendación Z.102, que trata de los conceptos relativos a la partición en varios niveles de detalle.

La observación global de la estructura interna de todas las partes de un sistema nos muestra diferentes estructuras, más o menos detalladas, de dicho sistema. Decimos que la estructura del sistema está representada con diferentes niveles de detalles.



A1 Especificación de una facilidad o característica independiente de la realización y de la red
A2 Especificación de un sistema independiente de la realización y de la red; incluye una descripción del entorno del sistema
A3 Recomendaciones y directrices del CCITT
A4 Contribuciones a la especificación del sistema que indican los requisitos de administración de la red y operacionales
A5 Otras Recomendaciones pertinentes
A6 Descripción de una propuesta de realización
A7 Especificación de proyecto
A8 Especificación detallada de un diseño
A9 Especificación completa de un sistema
A10 Documentación adecuada para la descripción de un sistema y de su entorno, para la simulación del sistema
A11 Documentación adecuada para la descripción de un sistema y de su entorno, para la prueba del sistema
A12 Manuales de instalación y explotación
A13 Contribuciones sobre la especificación del sistema preparadas por grupos funcionales especializados dentro de la Administración

Observación 1 — Es posible la iteración en todos los niveles.

Observación 2 — En ciertas circunstancias, la documentación LED que en este diagrama se indica como interna de una organización, por ejemplo A1, A7, A8, podría suministrarse a otra organización.

FIGURA D-1

Ámbito general del uso del LED

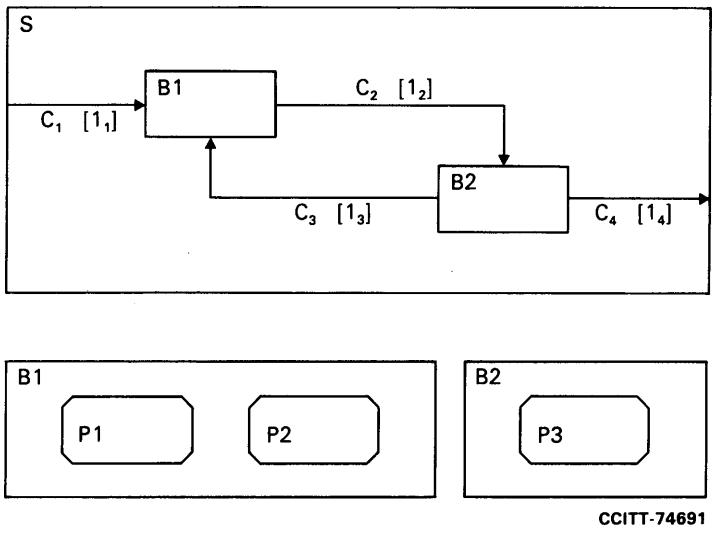


FIGURA D-2

Ejemplo de estructura mínima de un sistema

Obsérvese que la estructura interna de una parte del sistema da más detalles de la estructura, pero no necesariamente más detalles del comportamiento del sistema. Conceptualmente, es posible distinguir entre el aspecto de una representación más detallada del comportamiento (por ejemplo, el tratamiento de una nueva señal), y el aspecto de una estructura más detallada (por ejemplo, que comprende tanto estructura de partes como del comportamiento del sistema), pero en la práctica estos dos aspectos suelen combinarse, de manera que al introducir nuevos detalles de la estructura del sistema facilitamos también nuevos detalles sobre el comportamiento de éste. Por razones de claridad, en este capítulo se trata el aspecto de la estructuración únicamente.

D.4.2.2 *Criterios de partición*

La técnica que consiste en comenzar con una visión de alto nivel de la representación de un sistema y descomponerla en partes manejables se llama partición. Este proceso de partición añade estructura a un sistema.

Los criterios conducentes a la partición de la representación del sistema son diversos e incluyen los siguientes:

a) definir bloques o procesos de tamaño manejable desde el punto de vista intelectual;
b) crear una correspondencia con las divisiones reales del equipo lógico (software) y/o del equipo físico (hardware);
c) seguir las subdivisiones funcionales naturales;
d) minimizar la interacción entre los bloques;
e) reutilizar las representaciones ya existentes (por ejemplo, un sistema de señalización);
f) hacer corresponder una descripción a una cierta especificación.

Los criterios que se adopten de hecho dependerán tal vez de que se trate de una especificación o de una descripción, y del grado de detalle requerido.

Cada bloque puede subdividirse a su vez aplicando los mismos criterios u otros diferentes.

Como la relación entre los niveles dependerá de los criterios de partición adoptados, es importante indicar éstos claramente para facilitar la comprensión de la representación.

Los criterios de partición dependen del usuario pero existen algunas limitaciones a fin de garantizar una representación correcta en el LED. Las mismas se examinan en los siguientes puntos.

D.4.2.3 *Partición de un bloque*

Un bloque puede ser subdividido en un conjunto de bloques y canales, casi de la misma manera en que se subdivide un sistema en bloques y canales (véase la figura D-3).

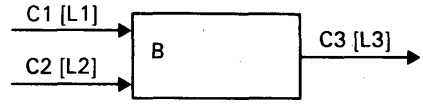
En el proceso de partición hay varias reglas importantes que recordar:

1) Los subcanales (por ejemplo, C1.1 y C1.2 en la figura D-3) conectados a un canal entrante (por ejemplo, C1) no deben contener ninguna señal nueva en sus listas de señales, y sus listas de señales deben contener todas las señales del canal original. Así, para el ejemplo de la figura D-3, L1.1 y L1.2 contienen todas las señales de L1. Además, en L1.2 no puede aparecer ninguna señal contenida en L1.1.

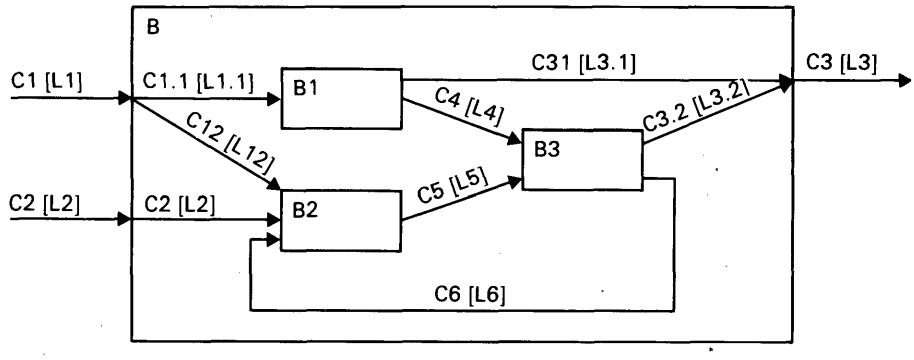
2) Los subcanales (por ejemplo, C3.1 y C3.2) conectados a un canal saliente (por ejemplo, C3) no deben contener el nombre de ninguna señal nueva en sus listas de señales, y sus listas de señales deben contener todos los nombres de señal del canal original. Así, L3.1 y L3.2 contienen todos los nombres de señal de L3. A diferencia de los subcanales conectados a un canal entrante, L3.1 y L3.2 pueden contener los mismos nombres de señal.

3) Si los nuevos canales (por ejemplo, C4, C5 y C6) tienen en sus listas de señales unos nombres de señal que no eran conocidos para el bloque original, estas nuevas señales tienen que definirse en la definición de la parte interna del bloque original (B).

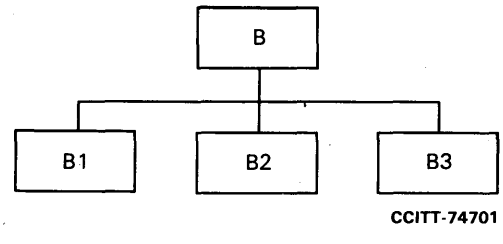
4) Si el bloque original contiene procesos, se dispone de tres opciones. Primero, cada proceso puede asignarse directamente a uno de los nuevos sub-bloques. Segundo, los procesos originales pueden descartarse y reemplazarse por nuevos procesos en los sub-bloques. Tercero, los mismos pueden subdividirse aplicando las reglas de partición para la partición de procesos (véase el § D.4.2.4).



a) Diagrama de interacción del bloque original



b) Diagrama de interacción del bloque subdividido



c) Diagrama en árbol del bloque

FIGURA D-3

Partición de un bloque

5) En el bloque de origen hay definiciones de datos que están disponibles para sus sub-bloques, de manera que cada uno de ellos puede utilizar un tipo de datos definido en el bloque de origen sin tener que redefinirlo.

6) Si un tipo definido en el bloque de origen está redefinido en un su-bloque, la nueva definición se aplica al sub-bloque de definición mientras que la anterior es válida para los demás sub-bloques. Se desaconseja hacer una redefinición con el único fin de caracterizar un sub-bloque, pues el lector podría pasarla por alto y suponer que es válida la definición anterior. En algunos casos conviene hacer una redefinición, sobre todo cuando se trata de una mejora del comportamiento. Debe hacerse resaltar esta redefinición por medio de anotaciones apropiadas.

D.4.2.4 *Partición de un proceso*

Los procesos son el medio de representar (una parte) del comportamiento funcional de un bloque.

Al observar este comportamiento puede considerarse necesario subdividirlo entre subcomportamientos. Esto equivale a decir que un proceso se subdivide en subprocesos.

Los subprocesos resultantes deben representar (todos juntos) el mismo comportamiento que el proceso de origen. Cada señal recibida por el proceso de origen debe ser recibida por un solo subproceso, y cada señal generada por el proceso de origen debe ser generada al menos por uno de los subprocesos.

La partición de un proceso puede ser consecuencia de la partición del bloque que contiene el proceso de origen, o puede ser independiente de la partición de cualquier otra construcción, o sea que pueden hacerse por sí misma.

D.4.2.4.1 La partición de un proceso puede obedecer a la partición de un bloque. Primeramente, debe quedar bien en claro que la partición de un bloque no implica por fuerza la partición del proceso o procesos que representan su comportamiento. Puede suceder que un proceso represente el comportamiento (o parte del comportamiento) de uno de los sub-bloques. En este caso no hace falta subdividir el proceso (véase la figura D-4).

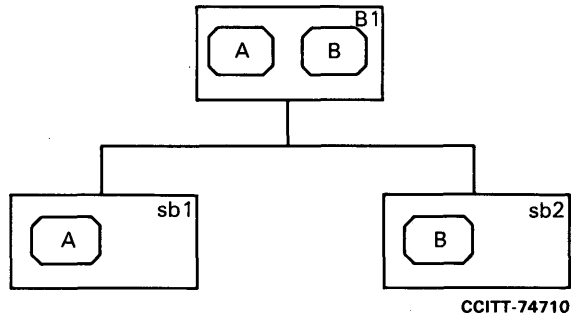


FIGURA D-4

Participación de un bloque que no entraña la partición de los procesos del bloque

En los casos en que el comportamiento representado por el proceso corresponde al comportamiento de dos o más sub-bloques, el proceso debe subdividirse en tantos procesos como sea necesario, respetando la regla de que cada proceso puede representar el comportamiento de (una parte de) un solo bloque únicamente.

Si el bloque ha sido subdividido en «n» sub-bloques, cada uno de los procesos asociados a ese bloque puede subdividirse en «m» subprocesos, en donde m puede ser cualquier número mayor que 0 independientemente de «n». Además, conviene advertir que cada uno de los procesos asociados al bloque se subdivide independientemente de los demás, de modo que la partición de uno puede entrañar dos subprocesos, otro puede entrañar cuatro subprocesos y un tercero un solo subproceso.

Si un proceso se subdivide en un solo subproceso desde el punto de vista de la estructura, ese subproceso es igual al proceso de origen.

D.4.2.4.2 Un proceso puede subdividirse independientemente de los bloques a los que está asociado. En este caso, los subprocesos resultantes de la partición del proceso de origen reemplazan a éste en el mismo bloque al que estaba asociado.

D.4.2.4.3 El motivo de la partición del proceso en paralelo con la partición del bloque puede ser representar el comportamiento de los diversos sub-bloques. El motivo de subdividir un proceso independientemente del bloque puede ser simplificar la representación aislando ciertos aspectos del comportamiento.

Adviértase que, si estos aspectos se consideran como subcomportamientos (como por ejemplo validación de frecuencia o reconocimiento de cifra), sería mucho mejor representarlos por medio de procedimientos y macros.

Por el contrario, si los comportamientos aislados son comportamientos «principales» que tienen todos el mismo rango (importancia), es mejor representarlos por subprocesos. En el primer caso, los subcomportamientos pueden tratar las mismas señales de entrada que el proceso principal, mientras que en el segundo caso cada subproceso tiene un conjunto de señales de entrada que está disociado de los conjuntos de señales de entrada de los otros subprocesos.

Un ejemplo de este último caso puede ser la partición del «proceso de tratamiento de la llamada» en el «subproceso de tratamiento del abonado», el «subproceso de encaminamiento» y el «subproceso de tratamiento del enlace» (véase la figura D-5).

Al subdividir un proceso hay algunas reglas importantes que recordar:

1) La partición de un proceso entraña una partición del conjunto de señales entrantes, de modo que un determinado subconjunto de señales será recibido por un cierto subproceso, otro subconjunto por otro subproceso y así sucesivamente. Los subconjuntos están disociados y contienen globalmente el conjunto de señales recibido por el proceso de origen. También pueden contener otras señales producidas por la partición; estas señales se utilizan para transferir información entre subprocesos.

2) Las definiciones de datos en el proceso de origen son utilizables por los subprocesos sin necesidad de redefinición. Si un tipo de datos definido en el proceso de origen está redefinido en un subproceso, la nueva definición se aplica al subproceso de definición únicamente, y la antigua es válida para los demás subprocesos.

3) Todas las variables tienen que ser redefinidas en los subprocesos. Los procedimientos definidos en el proceso de origen o en el bloque asociado al proceso de origen son visibles para los subprocesos.

4) Para cada definición de proceso es posible tener un número variable de instancias de proceso. Cuando un proceso es subdividido en subprocesos, una petición de creación para el proceso causa la creación de una instancia de cada uno de los subprocesos enumerados en la definición de la subestructura del proceso. Cada uno de estos subprocesos es un proceso y posee todas las características de cualquier proceso LED.

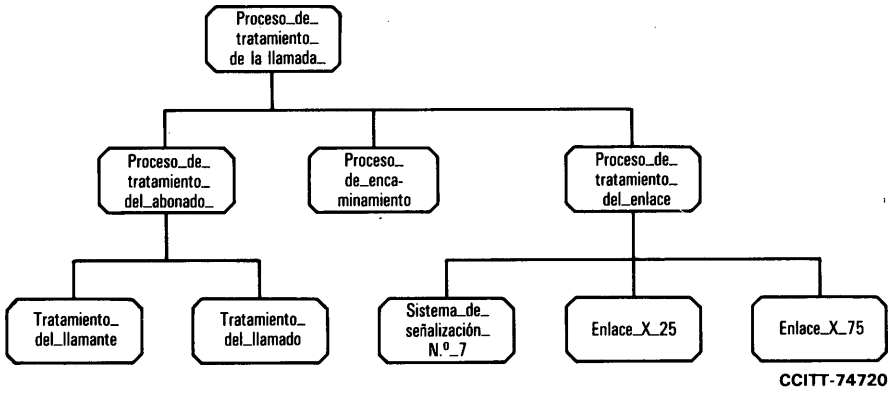


FIGURA D-5

Ejemplo de partición de un proceso

D.4.2.5 *Partición de un canal*

Un canal puede subdividirse independientemente de los bloques que conecta. Esto permite representar el comportamiento del canal cuando transmite señales. Para obtener una representación exacta de la manera en que se transmite una señal, a veces puede ser necesario representar el comportamiento del canal. Esto se hace considerando el canal como un elemento en sí mismo, cuyo entorno son los dos bloques que conecta, conforme ilustra la figura D-6.

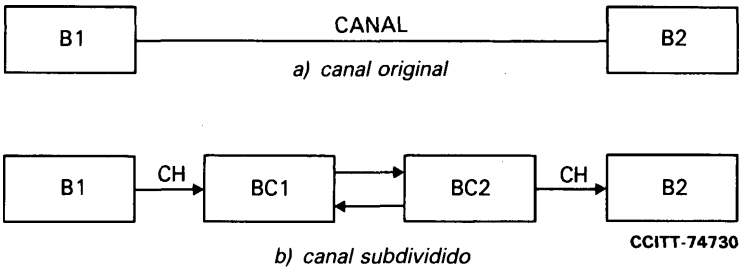


FIGURA D-6

Ejemplo de la partición de un canal

Presentando un canal de esta manera se puede mostrar su estructura (los bloques en que está estructurado) y, para cada bloque, su interconexión con los demás bloques y los procesos que representan el comportamiento de los bloques.

Al subdividir un canal, los canales entrantes y salientes no se dividen en múltiples subcanales pero deben estar conectados a un solo bloque. Con excepción de esta restricción, el canal subdividido tiene los mismos atributos que un bloque subdividido. Los bloques y canales contenidos en el diagrama de interacción del canal pueden también subdividirse aún más.

D.4.2.6 *Influencia mútua de las particiones de bloques, procesos y canales*

Hasta ahora hemos considerado la partición de bloques, procesos y canales independientemente entre sí. En realidad, estas actividades guardan estrechas relaciones. Como se ha visto, la partición de un bloque entraña habitualmente una partición de los canales de interconexión y de los procesos asociados.

Recíprocamente, la partición de un proceso singulariza determinados comportamientos que suelen estar a cargo de partes específicas de un sistema (transformable cada una en el concepto de bloque).

La partición de la representación de un sistema puede obedecer a distintos motivos, conforme se explica en el § D.4.2.2, pero el resultado es la partición de todas las entidades en que está representado un sistema.

Esto se examina con más detalle en los puntos siguientes. La partición de bloques, procesos y canales es evidente por sí misma cuando se consideran aspectos tales como la complejidad, la gestión o el reflejo de la estructura real del sistema.

Las Recomendaciones relativas al LED ofrecen construcciones, reglas y notaciones para la partición de estas entidades, pero los usuarios tendrán que asumir el hecho de que intervienen otras entidades LED en la partición, y la consiguiente necesidad de relacionar las diversas entidades entre sí.

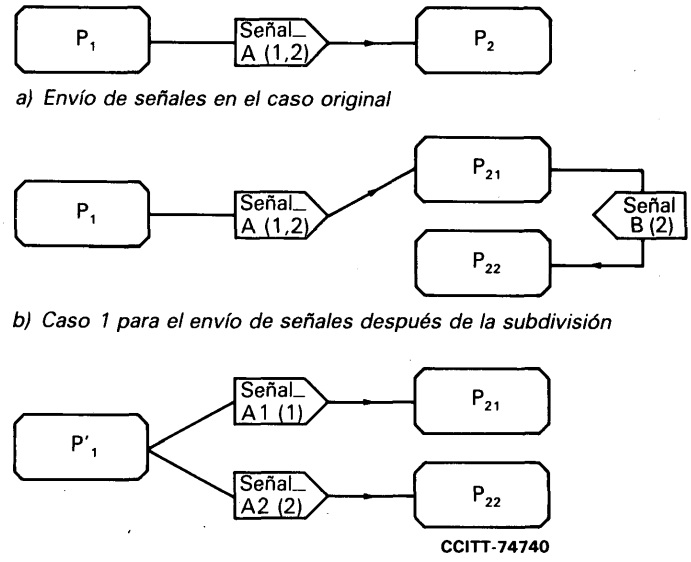
D.4.2.6.1 *Señales*

Una señal transporta información desde un proceso a otro (el comportamiento del entorno puede modelarse por uno o más procesos LED). Esta información puede ser muy compleja y se transmite en un determinado instante de la vida del sistema, es decir, cuando el proceso emisor interpreta una cierta transición y, después, cuando el proceso receptor llega a cierto estado.

La partición de un proceso en subprocesos puede entrañar la necesidad de distribuir entre varios subprocesos la información «envasada» en una señal.

Esto puede resolverse de dos maneras. Según la primera manera, la señal no resulta afectada, será recibida por un subproceso y el subproceso generará las señales necesarias para transmitir la parte de la información que necesitan los otros subprocesos. La segunda manera afecta a la señal. En este caso se la subdivide en varias señales, una para cada uno de los subprocesos que necesitan la información.

Estos dos mecanismos se ilustran en la figura D-7.



a) Envío de señales en el caso original

b) Caso 1 para el envío de señales después de la subdivisión

c) Caso 2 para el envío de señales después de la subdivisión

FIGURA D-7

Mecanismo de envío de señales en el caso de partición de un proceso

Los dos mecanismos expuestos parecen equivalentes y, en apariencia, el segundo podría mapearse siempre al primero, pero la diferencia radica en el hecho de que el primero efectúa una transferencia secuencial de la información mientras que el segundo la efectúa en forma potencialmente paralela. Asimismo, en el segundo caso el proceso de envío no es idéntico al del caso original pues los dos procesos envían señales diferentes.

Aparte de la precedente consideración existe un caso en el cual el segundo mecanismo es el único que puede utilizarse. Supóngase que en un alto nivel de descripción se utilice una señal potente que transporta varios tipos de información.

Al subdividir la representación del sistema (dando así más detalles sobre la estructura), puede que las informaciones contenidas en la señal no puedan ya envasarse por el hecho de que no estén disponibles en el mismo lugar y/o al mismo tiempo. Esto puede deberse al hecho de que la partición sitúa la información en bloques diferentes (origen diferente de la señal) o la pone a disposición en instantes diferentes. Un ejemplo de este último caso puede ser la señal de «selección del abonado llamado» en el nivel alto, que en el nivel bajo se convierte en una secuencia de señales de «cifra». Adviértase que se podría concebir un procedimiento de envase que acumule todas las «cifras» y envíe entonces la «selección del abonado llamado», pero esto puede ser una representación distorsionada o, peor aún, una representación imposible si se desea mostrar que puede tener lugar una liberación hacia atrás desde el enlace saliente después de cada cifra.

Un ejemplo de la situación anterior puede ser la señal de «establecimiento de línea completado», que en un nivel inferior se subdivide en un conjunto de señales originadas por diferentes sub-bloques, cada uno de los cuales realiza una parte del establecimiento total.

Como muestran estos ejemplos, hay varios casos en que una señal debe subdividirse como resultado de la partición del sistema.

Las Recomendaciones relativas al LED no preconizan ninguna notación particular, aun cuando se hayan propuesto y examinado algunas en el grupo encargado de la definición del LED.

En las representaciones LED con diversos niveles es aconsejable anotar las señales que resultan de la partición de una señal en un nivel superior. Esto mejora muchísimo la legibilidad de la representación al relacionar entre sí los diversos niveles.

La notación debiera indicar la señal de origen para una cierta señal, cuando esa señal está definida, así como una referencia hacia adelante que indique, para cada señal, las señales resultantes de su partición.

Adviértase que la partición de una señal afecta tanto al emisor como al receptor, así como al canal, que contiene ahora nuevos nombres de señal. Cuando se subdividen varias señales se pierden los beneficios resultantes de mantener el antiguo canal, con la adición de subcanales en sus extremos, y de cierta manera esta representación resulta inconveniente pues hace necesario redefinir parte del antiguo canal. El alcance de esta redefinición depende del número de señales subdivididas.

El segundo método indicado en la figura D-7, consistente en sustituir el antiguo canal por otros nuevos, resulta más interesante en este caso por su simplicidad, teniendo en cuenta que de todas maneras el emisor y el receptor resultan afectados.

D.4.2.6.2 *Estados*

La partición de un proceso en subprocesos puede también considerarse respecto a sus componentes, esto es, los estados, decisiones, tareas y datos (las entradas y salidas son afectadas por la partición de la señal).

Un estado debiera ser la representación de una situación lógica de un proceso, por ejemplo, la fase de conversación o la fase de prueba. La partición del proceso en subprocesos entraña la partición de los estados lógicos en un conjunto de estados que abarca varios subprocesos (pero no necesariamente todos ellos).

Eso influye tanto en la interpretación humana como en la interpretación automática de la correlación entre nivels.

Al aumentar el número de detalles, el ser humano tiene tendencia a perder de vista la situación general y ello puede tener efectos indeseados (piénsese cuán a menudo la actualización de un programa que se suponía perfecto genera efectos indeseados en el sistema global).

La dispersión de detalles, lógicamente agrupados, tiene un efecto más perjudicial aún sobre la capacidad de comprensión de una situación global. La organización del LED de una representación en varios niveles procura evitar este problema, pero es esencial correlacionar los diversos niveles entre sí.

Puede obtenerse esta correlación mediante notaciones apropiadas que remitan a la representación más abstracta que ofrece una imagen general (el nivel superior) y referenciando las otras partes que deban considerarse juntas.

La cuestión de la interpretación automática es totalmente diferente de la interpretación humana. La máquina no tiene ninguna imagen general, o sea que tiene en cuenta todos los elementos, considerando cada uno con la misma atención que los demás, y los tiene en cuenta uno por vez, una relación por vez. Por ello la dispersión de informaciones relacionadas entre sí no afecta a la interpretación por una máquina (dejando de lado el tiempo que toma la interpretación). Con una máquina, el problema radica en la formalización de la partición del proceso, a fin de que la máquina comprenda que, lo que antes se llamaba «reposo» es ahora una serie de estados de «reposo» distribuidos en varios subprocesos, y así sucesivamente.

La única posibilidad de conseguir que una máquina sea consciente de esta correspondencia es declarándola explícitamente (lo que resulta sumamente ineficaz y raramente está al resguardo de la posibilidad de olvidar algunas declaraciones) o formalizando las reglas de partición de un proceso en subprocesos.

Este asunto sólo se aborda (y resuelve) por medio de la metodología y, deliberadamente, no se trata en las Recomendaciones relativas al LED.

D.4.2.6.3 *Decisiones*

Las decisiones pueden resultar afectadas por la partición de un proceso si se subdividen los datos en que se basan. Si un dato deja de estar disponible para un subproceso que tiene que tomar una decisión en base a él, hay que enviar una señal para pedir el valor del dato, y el subproceso debe esperar la respuesta. Se puede ocultar el mecanismo de envío (petición) y de recepción (respuesta) de una señal declarando que los datos son importados y aplicando la decisión a los datos importados.

Adviértase que, como los subprocesos están asignados usualmente a bloques diferentes, el mecanismo de visión de datos compartidos no es utilizable.

D.4.2.6.4 *Tareas*

Una tarea es la representación de un conjunto de acciones que no tiene ningún efecto directo fuera del proceso. Cuando se subdivide el proceso, se puede subdividir también el conjunto de acciones ejecutadas por una tarea, de manera tal que algunas partes se ejecuten en un subproceso, otras en subprocesos diferentes, etc.

Adviértase que la subdivisión tiene lugar únicamente en los subprocesos de ese proceso. Si en un determinado proceso se ejecutaba un cierto conjunto de acciones, el mismo conjunto es ejecutado por la totalidad de los subprocesos de ese proceso. En este caso hay una pequeña diferencia con el estado, cuya subdivisión puede depender de la partición de otros procesos, porque un estado de un proceso es parte de una situación lógica que puede incluir más procesos.

Volviendo a la tarea, la partición de las acciones ejecutadas en varios subprocesos exigirá habitualmente la introducción de nuevas señales para activar la ejecución de un subconjunto de acciones en un subproceso al terminar las acciones ejecutadas por otro subproceso.

La partición de una tarea es por lo común un efecto derivado de la partición de los datos asociados a un proceso. También en este caso una metodología de diseño/representación puede ofrecer notaciones formales para seguir el itinerario de la partición de un conjunto de acciones en varios subprocesos.

En relación con la partición de una tarea vienen al caso los mismos consejos relativos a la anotación de las señales para ayudar al lector.

D.4.2.6.5 *Datos*

Los datos están asociados a un proceso. Cuando se subdivide un proceso se subdividen también sus datos. En el LED no hay una manera formal de indicar esta partición. Cada uno de los subprocesos tiene que definir sus variables y los nuevos sinónimos y tipos, llegado el caso.

Obsérvese que la partición de datos afecta no solamente a las acciones del proceso (una cierta salida que transporta una información no puede ser enviada si la información no está ya disponible para ese subproceso, una tarea basada en esa información no puede ejecutarse, etc.) sino que puede afectar también a otros procesos si intervienen datos exportados o revelados.

Los procesos importadores deben ser notificados de la partición del exportador, a fin de que puedan dirigirse al nuevo subproceso para obtener la información. Unos datos que eran visibles puede que ya no sean visibles si el subproceso que los posee está asignado a un sub-bloque diferente del observador y por ende su valor debe ser importado/exportado o debe ejecutarse un intercambio explícito de señales.

D.4.2.7 *Representación del sistema en caso de partición*

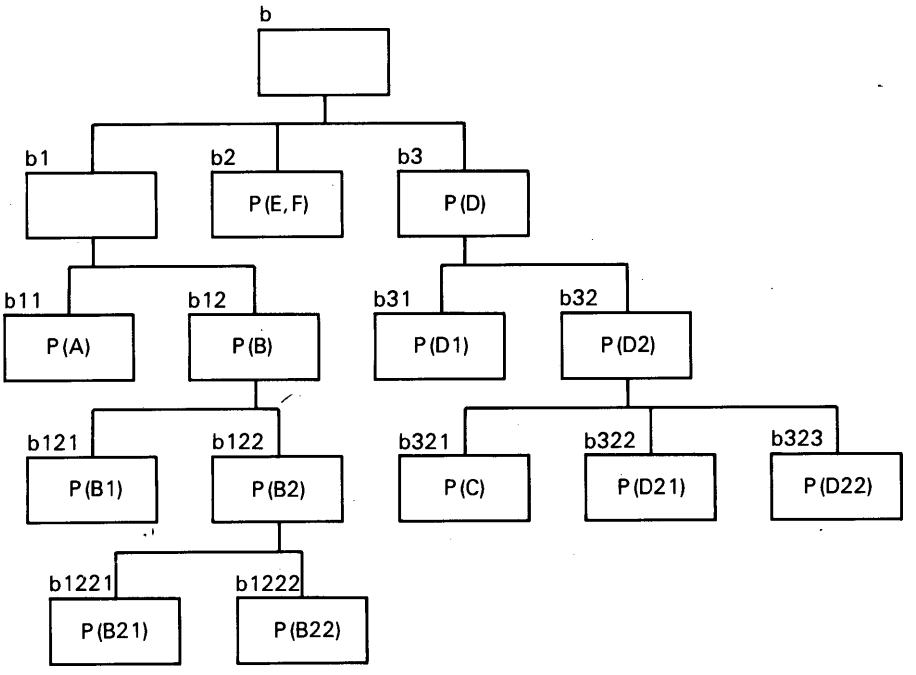
En los casos en que un sistema está representado por un conjunto de bloques interconectados por canales y en que el comportamiento de cada bloque está expresado por uno o varios procesos, tenemos una representación mononivel. Esto significa que podemos ver todos los elementos de la representación en el mismo nivel. Cuando introducimos la partición, insertamos una relación jerárquica entre los diversos documentos. Tendremos un documento con la representación de la estructura del sistema, estando compuesto éste por n bloques.

Un documento diferente puede presentar el sistema como compuesto por un conjunto diferente de bloques, algunos de los cuales se derivan de los bloques del documento anterior (algunos bloques del documento anterior han sido sustituidos por los sub-bloques obtenidos por la partición de los bloques). Este último documento debe estar relacionado con el anterior.

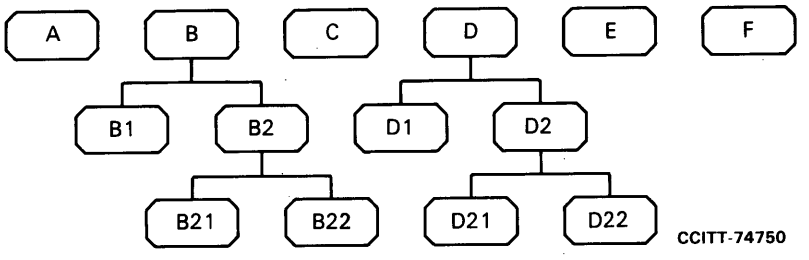
Para obtener una representación completa del sistema no basta simplemente con relacionar los documentos entre sí, sino que los mismos deben estar organizados de manera tal que sea posible tener acceso a la representación del sistema por niveles, comenzando con una imagen general y pasando a representaciones cada vez más detalladas. Esto hace necesario agrupar los diversos documentos a fin de que formen diversos niveles de representación del sistema.

No todos los niveles deben contener los mismos elementos. En un primer nivel, la representación del sistema puede consistir en representaciones de bloques y canales sin incluir los procesos que describen el comportamiento de cada bloque. En un nivel más bajo podría incluirse la representación del comportamiento de algunos bloques pero no el de otros, o incluso representar parte del comportamiemto de un bloque pero no su comportamiento completo, que aparecerá solamente en un nivel inferior. El nivel de representación más bajo de todos (el más detallado) debe comprender la representación completa de los comportamientos de todos los bloques, o sea el conjunto completo de procesos que expresan este comportamiento.

El primer efecto derivado de este tipo de representación por niveles consiste en que hay que indicar la asociación de proceso a bloque. La partición del sistema (considerado como un solo bloque) genera una estructura en árbol; la partición de los «n» procesos que representan el comportamiento del sistema genera «n» árboles y cada uno de los procesos de estos árboles tiene que asociarse a un bloque indicado en el árbol de bloques. Además, puede que se introduzcan nuevos procesos en un cierto nivel, quizás subdivididos aún más a partir de ese nivel. Cada uno de ellos, si es subdividido, generará una estructura en árbol y también en ese caso cada proceso en la estructura debe asociarse a un bloque en el árbol de bloques, conforme ilustra la figura D-8.



a) Árbol de bloques



b) Árboles de procesos

Observación — Los bloques b y b1 no tienen procesos asociados.
El bloque b2 contiene dos procesos (E, F).
El bloque b321 contiene el proceso C que aparece en ese nivel solamente.

FIGURA D-8

Asignación de procesos a los bloques

Observando el árbol de bloques podemos formular varias consideraciones.

Primeramente, el árbol tiene siempre una sola y única raíz, a saber, el bloque del sistema. Tal ocurre incluso en los casos en que un sistema está representado desde el comienzo como compuesto de varios bloques. En el árbol de bloques estos bloques estarán representados, por ejemplo, en el nivel 1. El bloque raíz puede consistir en el nombre del sistema únicamente. Las definiciones de los canales pueden indicarse para los bloques en el nivel 1, aunque esto no es obligatorio a menos que los bloques tengan procesos asociados.

Surge un segundo punto observando las hojas del árbol: no están todas en el mismo nivel. Esto es el resultado de un número diferente de particiones de los bloques del árbol. El número de particiones depende de varios aspectos, la mayoría de los cuales dependen de la apreciación subjetiva del especificador/proyectista.

El LED sólo plantea la condición de que los bloques hoja pueden quedar sin más partición solamente si su comportamiento está completamente especificado (es decir, si las definiciones asociadas de los procesos son suficientes para representar su comportamiento). En consecuencia, un bloque hoja debe contener por lo menos un proceso asociado.

Cuando la representación de un sistema se expresa en varios niveles de abstracción, podemos seleccionar cualquiera de esos niveles para representar el sistema. La elección de un cierto nivel implica la consideración de los bloques de ese nivel, de sus procesos asociados y de todos los bloques que son bloques hoja en niveles más altos junto con sus procesos asociados (figura D-9).

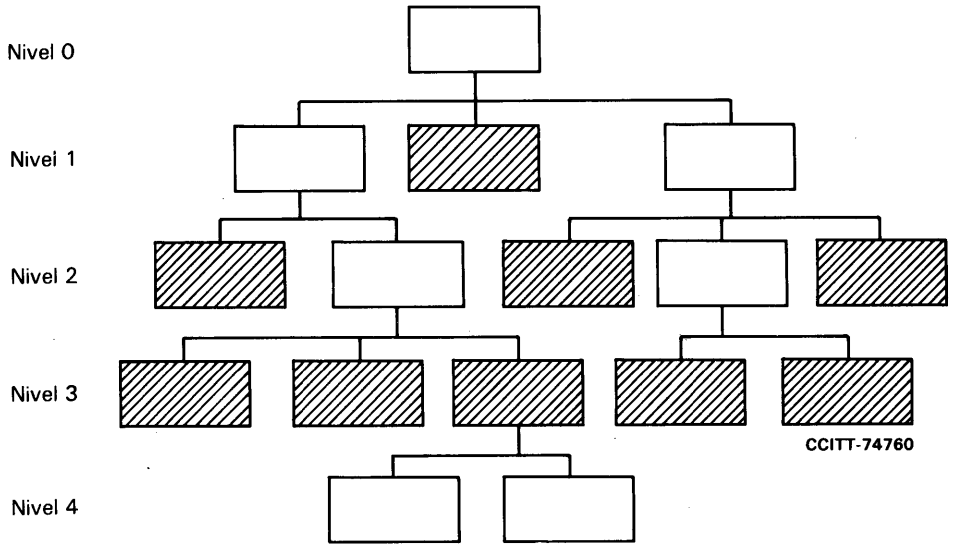


FIGURA D-9

Nivel 3 de representación

La representación de un sistema en un cierto nivel puede ser incompleta en el sentido de que alguno de los bloques de ese nivel no tiene procesos asociados o que los asociados no representan completamente el comportamiento de aquellos.

Los motivos para que se hable de niveles de representación y de la selección de un cierto nivel de representación son los siguientes:

- puede que lleguemos en nuetro diseño a un cierto «nivel» de detalle que esté representado por ese nivel de representación (ien este caso los bloques de ese nivel son bloques hoja y no están completos pues hace falta seguir trabajando!);
- deseamos observar la representación del sistema con un cierto nivel de detalle; en consecuencia, elegimos el nivel de representación que mejor corresponda a la abstracción que buscamos. Adviértase que en algunos casos un cierto nivel de representación puede consistir en documentos con diferentes niveles de abstracción. Una parte del sistema puede representarse en forma muy dellada en el nivel 2, mientras que otra parte puede ser aún abstracta en el nivel 4. Quiere decir que cuando elegimos una representación en el nivel 3 podemos tener partes muy detalladas junto con partes que sólo pueden considerarse como una visión general;
- la metodología de representación/diseño puede ser tal que cada nivel tenga un significado preciso. Por ejemplo, el nivel 1 puede corresponder a la especificación, el nivel 2 presentar la estructura global del sistema, el nivel 3 la estructura modular (bastidores, funciones del equipo físico), el nivel 4 la estructura detallada (tarjetas de circuito impreso, procedimientos, módulos de equipo lógico). En este caso la elección de un cierto nivel responde a las necesidades de un determinado lector, y cabe señalar que la metodología evitará una situación de discrepancias en el nivel de detalles de las partes que componen un determinado nivel de representación.

Además de la representación del sistema total y de la expresada por niveles, el LED posee el concepto de «representación homogénea del sistema» (figura D-10).

Definimos así cualquier representación del sistema considerada como una representación mononivel en la cual todos los bloques pueden tomarse de cualquier nivel de la estructura del sistema, a condición de que:

- se pueda elegir que un bloque forme parte de la representación homogénea del sistema si el mismo puede considerarse como un bloque hoja (o bien es un bloque hoja, o bien tiene asociados todos los procesos necesarios para representar su comportamiento);
- si se elige un bloque, todos los bloques obtenidos con la partición de su bloque de origen deben incluirse ya sea directamente o por inclusión de sus vástagos;
- se faciliten todos los documentos que definen las señales que circulan por el canal que conecta un bloque de la representación. Esto puede entrañar según la estrategia de partición adoptada, que una vez que se ha tomado un bloque, su origen deba tomarse también, al menos para las partes que definen los datos y las señales.

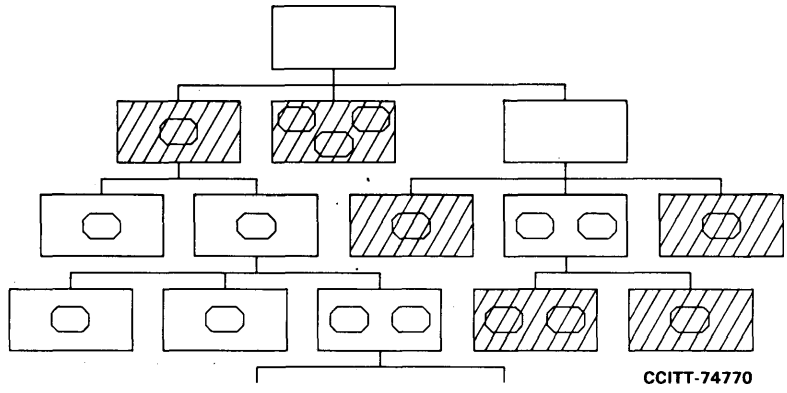


FIGURA D-10

Ejemplo de representación homogénea de un sistema

En los casos en que algunos canales se subdividen considerándolos como sistemas, debe facilitarse la representación de cada uno de estos «sistemas».

La representación tiene los mismos tipos de documentos que la representación usual de un sistema. Debe agregarse una notación para relacionar estos sistemas con el sistema principal. En cierto modo, podemos considerar estos sistemas como sistemas internos con respecto al «sistema principal». Cada uno de estos sistemas puede tener varios niveles de representación y tener también sistemas internos si alguno de los canales contenidos en ellos se trata aún como un sistema en sí (figura D-11).

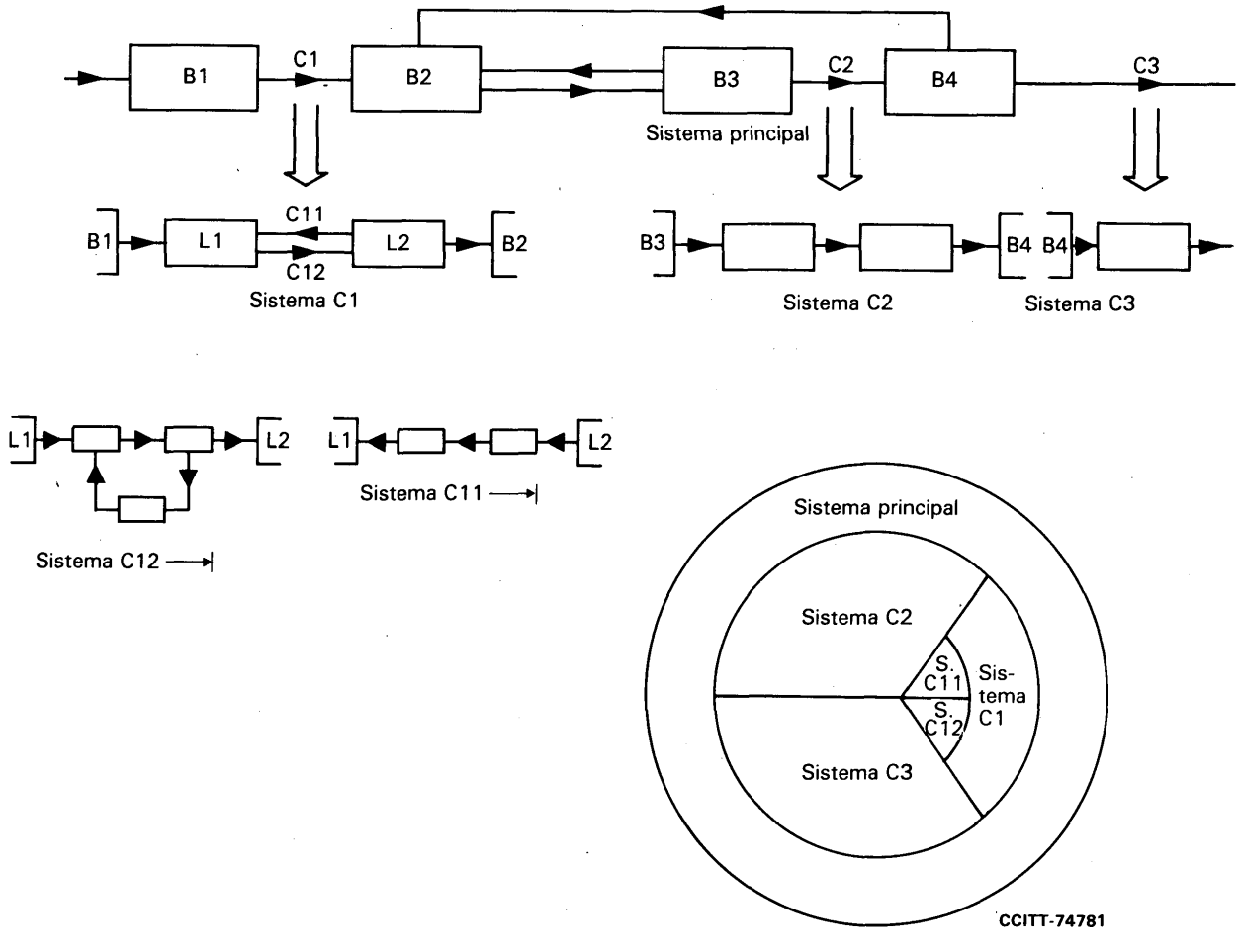


FIGURA D-11

Partición de un canal a modo de representación de un sistema virtual

D.4.3 *Conceptos del LED*

En esta sección se presenta cada uno de los conceptos del LED y se dan algunas directrices generales sobre la manera de utilizarlos. Para los ejemplos se utilizan casi siempre el LED/GR. Las mismas directrices son aplicables también al caso del LED/PR.

D.4.3.1 *Sistema*

Como ya se ha dicho, el LED modela sistemas. Un sistema es, pues, lo que la especificación o descripción LED define. Como tal, un sistema LED puede modelar una parte de un sistema (o centro) telefónico o una red completa de sistemas telefónicos o partes de una multiciplidad de centrales telefónicas (por ejemplo, los controladores interurbanos de ambos extremos de un enlace interurbano). Lo esencial es que, desde el punto de vista del LED, el sistema LED contiene todo lo que la especificación o descripción trata de definir. El entorno queda fuera de la especificación y no está definido en LED.

El sistema se relaciona con el entorno a través de canales. En teoría hace falta un solo canal entrante y un solo canal saliente para el interfaz con el entorno. En la práctica suelen definirse canales para cada interfaz lógico con el entorno.

Cada sistema consta de cierto número de bloques conectados entre sí por canales. Cada bloque es independiente de cualquier otro bloque. El único medio de comunicación entre procesos situados en dos bloques diferentes es el envío de señales por los canales.

D.4.3.2 *Bloque*

Conforme se ha mostrado en el punto relativo a la estructuración en LED, un bloque puede contener una estructura de uno o más procesos. La definición completa de un bloque requiere que los bloques en la parte inferior del diagrama del árbol de bloques contengan una o más definiciones de proceso.

Dentro de un bloque, los procesos pueden comunicar entre sí mediante señales o mediante valores compartidos. El bloque constituye así, no sólo un mecanismo conveniente para agrupar procesos, sino también una frontera para la visibilidad de los datos. Por ello al definir bloques hay que cerciorarse de que la agrupación de procesos dentro de un bloque es una agrupación funcional razonable. Las más de las veces resulta útil desglosar primero el sistema (o bloque) en unidades funcionales y definir luego los procesos que van en el bloque.

D.4.3.3 *Canal*

Los canales son el medio de comunicación entre bloques. Normalmente, un canal es una entidad funcional que puede usarse para denotar un trayecto de información específico. De hecho, mediante la partición de canales es posible especificar formalmente el comportamiento de cada canal.

Las definiciones de canal contienen una lista de señales que enumera todas las señales que pueden transportarse por ese canal. Esta lista de señales tiene por objeto garantizar que cada señal enviada por un proceso en un extremo del canal pueda ser recibida por el proceso del bloque situado en el otro extremo del canal. La definición del canal forma parte así de la definición del interfaz de cada bloque. En el caso de los proyectos grandes en que intervienen varias personas, el hecho de ponerse de acuerdo desde un principio sobre las señales que transmitirá un canal y sobre la definición de esas señales reduce la probabilidad de que dos procesos no puedan comunicar entre sí.

D.4.3.4 *Proceso*

Un proceso es una máquina de estado finito ampliada y define el comportamiento dinámico de un sistema. En esencia, los procesos se encuentran en un estado en espera de señales. Al recibir una señal, el proceso responde efectuando las acciones específicas que están especificadas para cada tipo de señal que el proceso puede recibir. Los proceos contienen muchos estados diferentes, para que puedan efectuar diferentes acciones cuando reciben una señal. Esos estados constituyen la memoria de las acciones que ha tenido lugar previamente. Una vez que han tenido lugar todas las acciones asociadas a la recepción de una determinada señal, se pasa a un nuevo estado y el proceso espera otra señal.

Los procesos pueden ya sea existir en el momento en que se crea el sistema o pueden crearse de resultas de una petición de creación proveniente de otro proceso. Además, los procesos pueden existir siempre o pueden cesar mediante la ejecución de una acción de parada interna. Algunos atributos importantes de la creación de procesos son los siguientes:

1) una vez creado el sistema, los procesos sólo pueden ser creados por otro proceso del mismo bloque. Una manera de permitir la creación de procesos en otros bloques consiste en prever procesos especiales en cada bloque que causen la creación de un proceso cuando reciban una señal de un proceso de otro bloque. En muchos casos este proceso especial es una especie de proceso de «sistema operativo»;

2) una vez creados, los procesos tienen un periodo de vida propio. Los procesos mueren solamente cuando ejecutan una acción de parada durante una transición. Una manera de modelar sistemas que permite operaciones externas que matan procesos consiste en prever una señal eliminadora especial. Cuando se recibe la señal eliminadora, el proceso ejecuta una acción de parada;

3) al crearse un proceso, éste conoce su propia identidad y la identidad de su origen. Si no se le da ninguna otra información, no puede comunicar con ningún otro proceso. Hay dos maneras de resolver este problema. Primero, uno o más de los parámetros del proceso contienen la identidad de los procesos con los cuales comunicará este proceso. Segundo, este puede recibir una señal que contiene un identificador de instancia de proceso que representa uno de sus parámetros actuales.

D.4.3.4.1 *Estados*

Un estado es un punto del proceso en el que no se ejecutan acciones sino que la fila de espera de entrada espera la llegada de señales entrantes. Según el identificador de señal que aparece en la señal de entrada, la llegada de la señal hará que el proceso deje el estado y ejecute una determinada secuencia de acciones. Una señal que ha llegado y causado una transición ha sido «consumida» y deja de existir. Durante una transición, un proceso no sabe explícitamente qué señal de entrada ha causado la transición. Esto sólo puede inferirse del contexto (es decir, esa transición solo podría ocurrir si se recibiese una determinada señal). En la figura D-12, la tarea T1 se efectúa únicamente si se recibe I1. Pero la tarea T2 será ejecutada si se recibe I2 ó I3. Si es importante que T2 sepa qué entrada se ha recibido, será mejor diseñar el proceso como se muestra en la figura D-13.

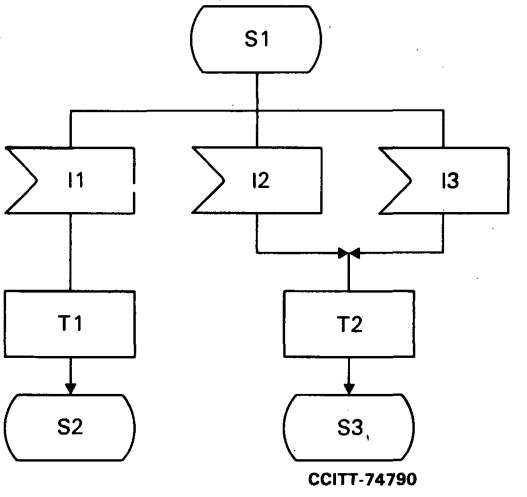


FIGURA D-12

Ejecución de una tarea en función de dos de tres señales recibidas pero independientemente de cual de ellas

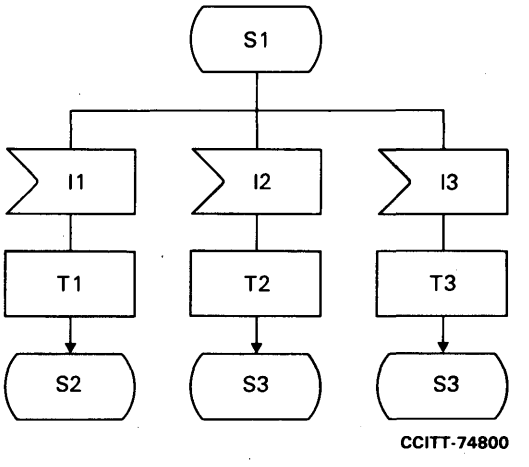


FIGURA D-13

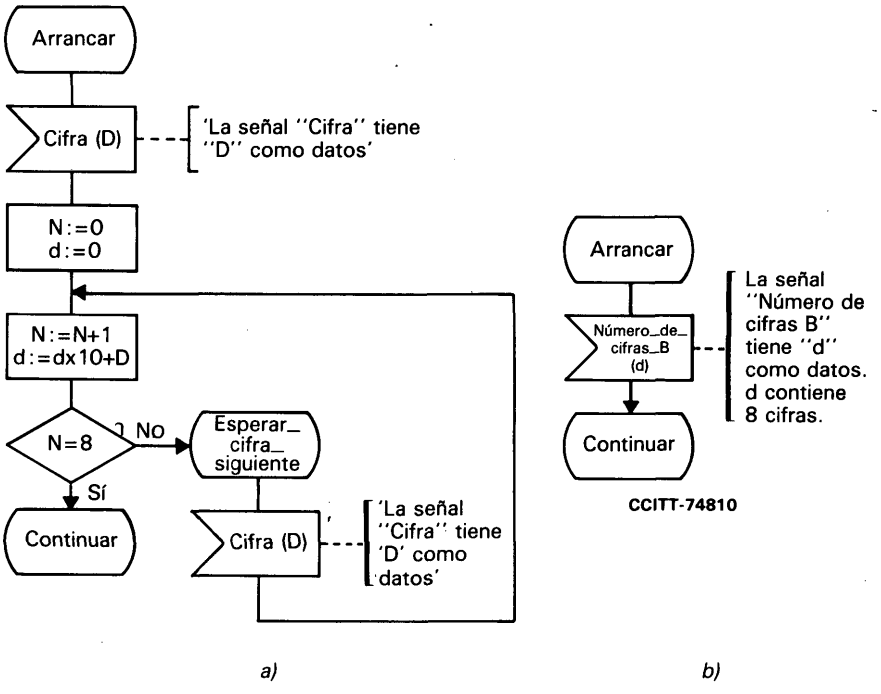
Ejecución de una tarea en función de la señal recibida

D.4.3.4.1.1 *Determinación de los estados requeridos*

Habitualmente, el autor de un diagrama LED dispone de cierta flexibilidad en su método de definir los estados de un proceso. Puede necesitar una estrategia que le permita identificar los estados de proceso; esta estrategia puede ser informal o formal. Se requiere un juicio (obtenido mediante la práctica) para producir diagramas LED que no sean innecesariamente complejos, debido a la identificación de un número excesivo de estados distintos, o que dejen de aprovechar las ventajas propias del LED reduciendo artificialmente el número de estados. Antes de que el autor comience la construcción del diagrama deben haber quedado resueltas las cuestiones preliminares (examinadas en el § D.4.2), por ejemplo:

- la estructuración del sistema en bloques funcionales;
- la representación de los bloques funcionales por medio de uno o más procesos LED por bloque;
- la elección de señales de entrada y de salida;
- la utilización de datos en el proceso.

Todos esos factores son de capital importancia para la determinación de los estados de cada proceso. La influencia de la «elección» de las señales de entrada en el número de estados de un diagrama LED se ilustra en los dos ejemplos de la figura D-14.



Observación — Los ejemplos *a)* y *b)* representan la misma función con diferentes niveles de detalle. En consecuencia, el número de estados varía.

FIGURA D-14

Recepción de un número telefónico de 8 cifras

D.4.3.4.1.2 *Reducción del número de estados*

Después de haber seguido determinados principios para la identificación de los estados de un proceso el autor de un diagrama LED puede considerar que ha utilizado «demasiados estados». El número de estados es importante porque el tamaño y la complejidad de un diagrama LED suelen estar estrechamente relacionados con el número de estados. Hay ciertos medios para reducir el número de estados; no obstante, el hecho de que un diagrama LED sea complejo no es motivo, por sí solo para modificarlo, pues la complejidad del diagrama puede no ser más que la consecuencia natural de la complejidad del proceso definido. En general, debe elegirse un conjunto de estados tal que la interacción secuencial entre el proceso y su entorno tenga una claridad máxima. Por lo general no puede obtenerse esta claridad reduciendo al mínimo el número de estados. El número de secuencias independientes tratadas por un proceso tiene un efecto multiplicativo sobre el número de estados. Por ello conviene tratar las secuencias independientes en procesos separados, pues esto reduce el número de estados y aumenta la claridad.

El número de estados puede reducirse separando funciones comunes, fusionando estados o utilizando el concepto de procedimiento. Ciertas estructuras de datos pueden conducir también a un número reducido de estados. Otra representación puede beneficiarse del empleo de macros.

D.4.3.4.1.3 *Separación de funciones comunes*

Cuando se proyecta un diagrama LED puede resultar evidente que, para definir un aspecto particular y repetitivo de un proceso se requiere la representación estados repetitivos. En la figura D-15 se ilustra una parte de un diagrama LED, de un proceso de señalización, que ilustra la siguiente condición: para considerar que se ha detectado una señal de línea deberá identificarse la presencia de un tono de señalización de línea durante un determinado periodo de tiempo.

Para especificar esto se necesita un estado intermedio entre los estados de no'detecta_señal_de_línea y conversación. Supongamos que, en un diagrama completo, esta función tuviera que repetirse en cada punto en que se detecta la señal. Podría procederse de otro modo: definir un proceso separado que se encargara de supervisar el tono de señalización de línea y detectar señales sobre la base del tiempo de identificación especificado. Al existir este nuevo procedimiento, el diagrama de la figura D-15 se podría dibujar como muestra la figura D-16. (En un contexto dado, las figuras podrían hacerse equivalentes, si bien para ello se necesitaría introducir una nueva señal: señal_de_línea_válida.)

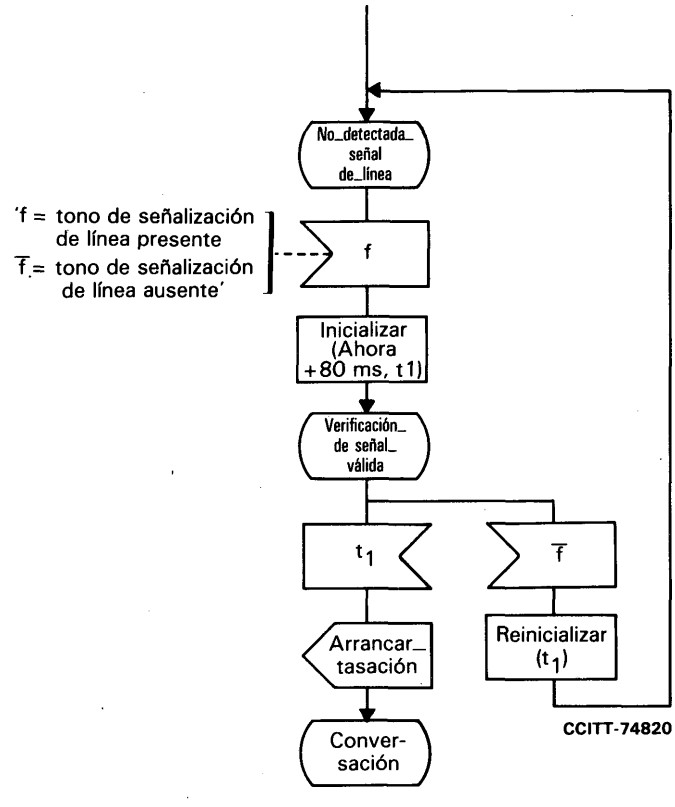
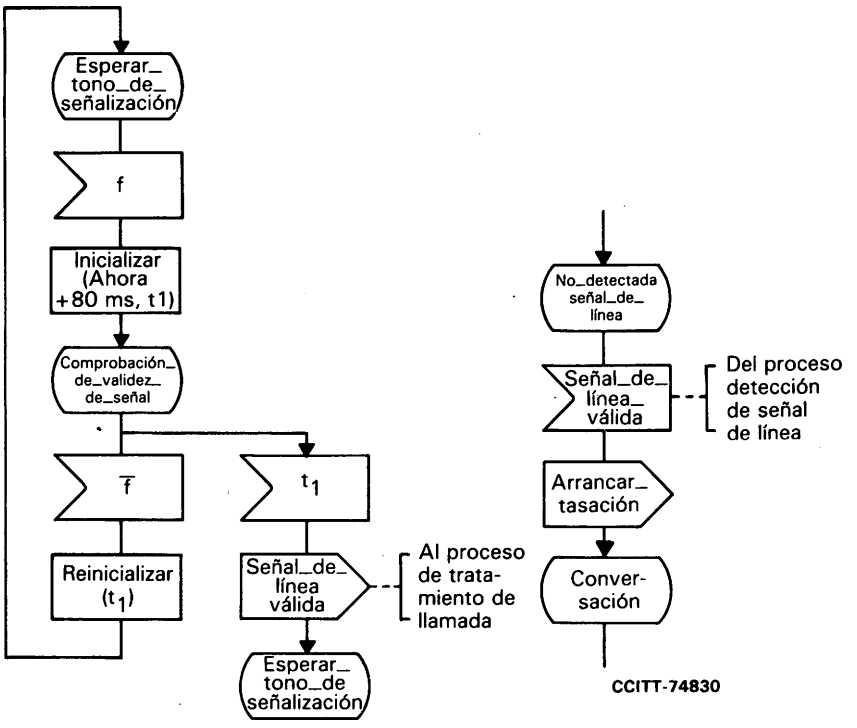


FIGURA D-15

Ejemplo de un diagrama LED para un proceso compuesto de tratamiento de llamada y detección de señal de línea

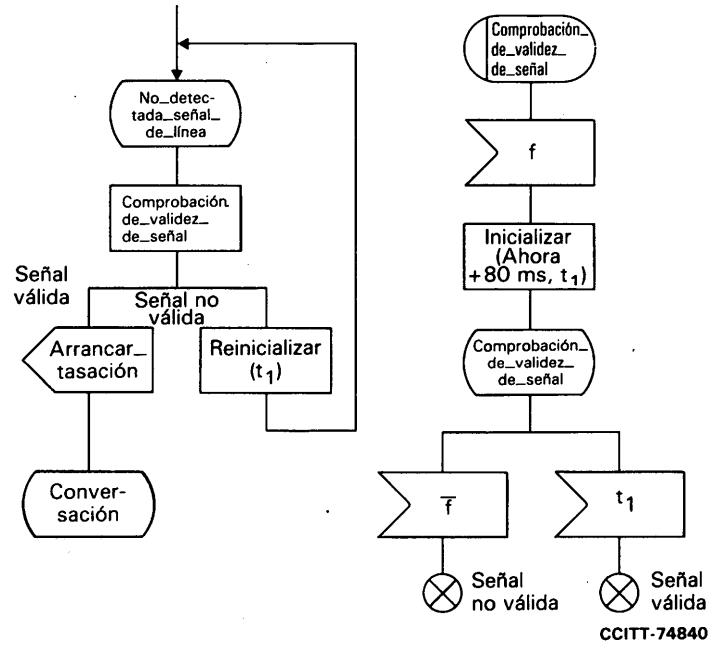


a) Proceso de detección de señal de línea

b) Proceso de tratamiento de llamada

FIGURA D-16

Ejemplo de segregación de una función común (detección de señal de línea) para evitar estados repetitivos como el de comprobación de validez de señal (compárese con la figura D-15)



a) *b)*

FIGURA D-17

El ejemplo de la figura D-16, con utilización de macros

Conviene notar la ligera diferencia existente entre el proceso mostrado en la figura D-15 y los dos mostrados en la figura D-16.

El proceso de la figura D-15 comienza a medir inmediatamente después de la recepción de t_1, mientras que el proceso de tratamiento de llamada (proceso b) en la figura D-16) comienza a medir al recibirse la señal_de línea_válida, que es generada y enviada a la recepción de t_1 por el proceso a) de la figura D-16. Ello significa que en este segundo caso, la medición comienza después de t_1 + generación de la señal, envío y tiempo de recepción. Además, puede haber otras señales que se han acumulado en espera al momento que se necesita para generar y enviar la señal_ línea_válida.

Un segundo aspecto que conviene notar es que esta separación de una subfunción no sería posible si la señal que ha de recibir la subfunción ha de recibirla también la función principal, porque una señal es dirigida siempre a un solo proceso.

Si en el ejemplo, la misma señal f ha de recibirse en otro estado, donde no es preciso validarla durante el tiempo t_1, no sería posible separar la subfunción de validación en otro proceso.

En general, las soluciones con procesos separados resultan útiles cuando las señales deben procesarse de una manera que es independiente de los estados del proceso principal. En este caso, unos procesos previos y posteriores pueden tratar las secuencias detalladas y liberar al proceso principal de los detalles. A menudo esto resultará en una útil modularidad también, ya que las particularidades de, por ejemplo, los sistemas de señalización, pueden aislarse del proceso principal, que está más orientado al servicio.

Una solución distinta de este problema consistiría en utilizar la anotación MACRO, como se indica en la figura D-17. En este caso obtenemos un diagrama compacto como el deseado sin alterar en lo más mínimo la semántica del diagrama original. Además, el MARCRO puede ser llamado desde varios estados si así lo exige la lógica del proceso.

D.4.3.4.1.4 *Fusión de estados*

Si en un diagrama LED dos estados tienen el mismo futuro, estos estados cualesquiera que hayan sido sus antecedentes (historia), podrán fusionarse en uno solo sin que esto afecte a la lógica del diagrama.

La figura D-18 muestra una parte de un diagrama LED con dos estados que difieren en su historia pero cuyos futuros son «idénticos». En la figura D-19 los dos estados han sido combinados para formar uno solo. Éste es un ejemplo bastante trivial en que sólo se obtiene una pequeña reducción de la complejidad, pero esta técnica puede utilizarse para obtener una simplificación apreciable. La semántica del LED no permite que un estado vaya seguido de una decisión en virtud de la cual se determine la historia del proceso antes del estado (esto es, a los efectos de este ejemplo «¿se envió A6 o B4?» a menos que esta información haya sido almacenada explícitamente antes de que el proceso haya entrado a este estado. Obsérvese que el nombre de los datos en una entrada hace que el valor sea almacenado.

Un estado debe tener una relación clara con las posibles situaciones lógicas del proceso, y por lo tanto no es aconsejable fusionar situaciones lógicas diferentes en un solo estado.

Ha de prestarse atención cuando un diagrama fusionado se modifica más adelante. El usuario debe investigar si el cambio previsto influye del mismo modo o no en las dos (o más) ramas originales de la línea de flujo.

D.4.3.4.2 *Entradas*

El presente § D.4.3.4.2 tiene por objeto explicar el concepto de entrada y el uso de entradas en diagramas LED sin el concepto de conservación. El concepto de conservación y el uso conjunto de entradas y conservaciones en un diagrama LED se explica en el § D.4.3.4.3.

D.4.3.4.2.1 *Consideraciones generales*

Un símbolo de entrada asociado a un estado significa que si la señal cuya denominación está inscrita en el símbolo de entrada llega cuando el proceso se encuentra en ese estado habrá que ejecutar la transición que sigue al símbolo de entrada. Cuando una señal provoca la interpretación de una transición, deja de existir y se dice que la señal ha sido consumida.

Una señal puede tener datos asociados. Por ejemplo, una señal llamada «cifra» sirve no sólo para provocar la ejecución de una transición por el proceso receptor sino también para transportar el valor de la cifra (0-9), dato que puede utilizar el proceso receptor.

Para que el proceso disponga de esos datos, deben estar nombrados en los símbolos de entrada, entre paréntesis. Procediendo de esta manera, los datos están disponibles durante la transición en curso. Si los datos no se almacenan explícitamente, no serán accesibles en otra transición más tarde.

Un elemento de datos está separado de otros por comas. En las figuras D-20, D-21 y D-22 aparecen ejemplos de la recepción de datos procedentes de entradas.

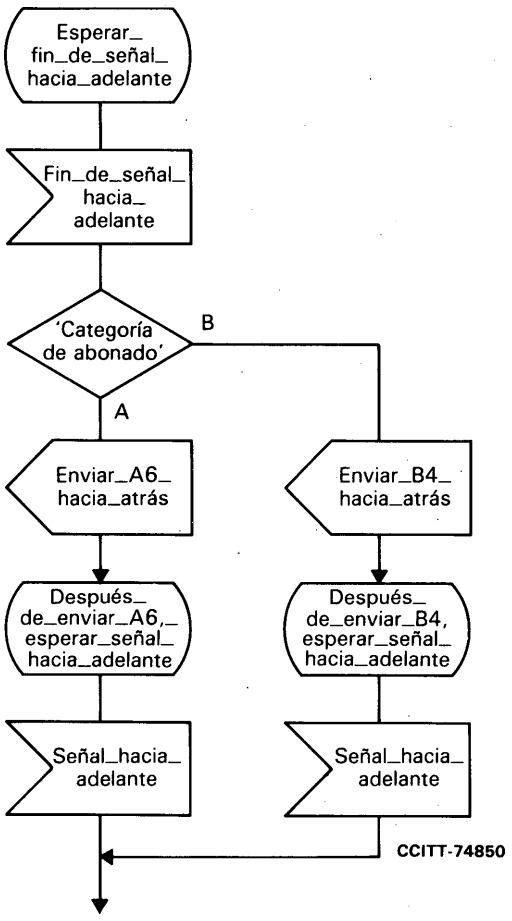


FIGURA D-18

Ejemplo de una parte de un diagrama LED antes de la fusión de estados

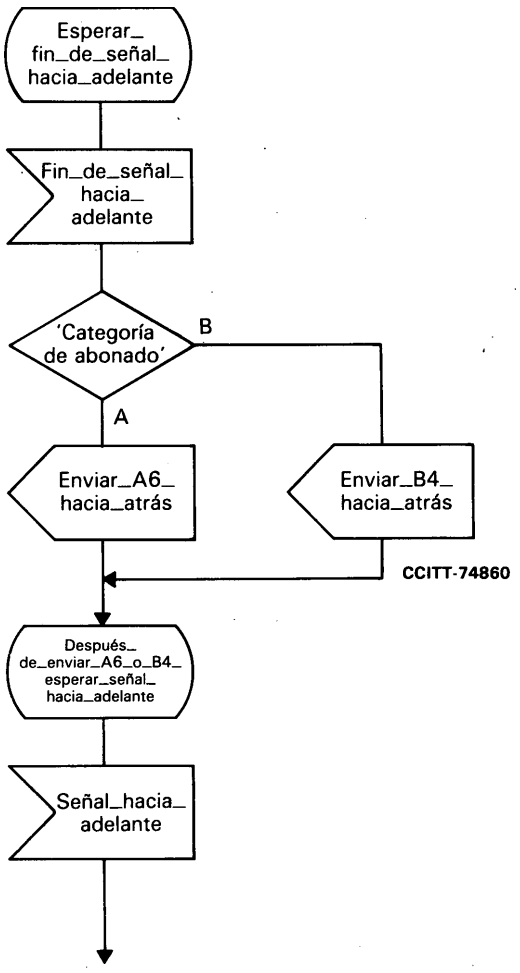


FIGURA D-19

Ejemplo de una parte de un diagrama LED con estados fusionados

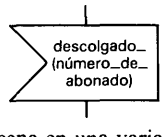
Los datos nombrados quedan a disposición del proceso receptor cuando se interpreta la entrada.

La figura D-20 muestra la recepción de la señal *descolgado*. La señal *descolgado* tiene datos asociados (número de abonado) con el valor 9269. Esta señal puede recibirse de tres maneras, como se indica en las partes a) a c) de la figura.

La figura D-22 muestra cómo enviar y recibir varios elementos de datos en una señal. Cada elemento debe estar separado del siguiente por una coma. En la figura D-22 c), mostramos cómo ignorar los elementos de datos no deseados dejando un hueco en la lista de elementos.

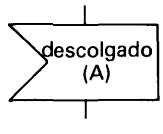
Obsérvese que en la señal de salida, podemos escribir expresiones para E1, E2 o E3, pero en la señal de entrada hemos de utilizar variables para recibir los valores enviados.

En el LED no es necesario incluir símbolos de entrada para representar señales cuya llegada necesitaría una transición nula (vacía), esto es, una transición que no contuviera acciones y que condujera al mismo estado de que se partió. Por convenio, para una señal cualquiera no indicada en un símbolo de entrada explícito en un estado particular, existe, en ese estado, un símbolo de entrada implícito y una transición nula. En virtud de este convenio, los dos diagramas indicados en la figura D-23 son lógicamente equivalentes y tanto da utilizar el uno como el otro.



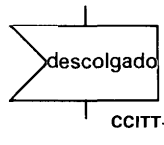
Observación — El valor 9269 se almacena en una variable denominada número de abonado.

a)



Observación — El valor 9269 se almacena en una variable denominada A.

b)



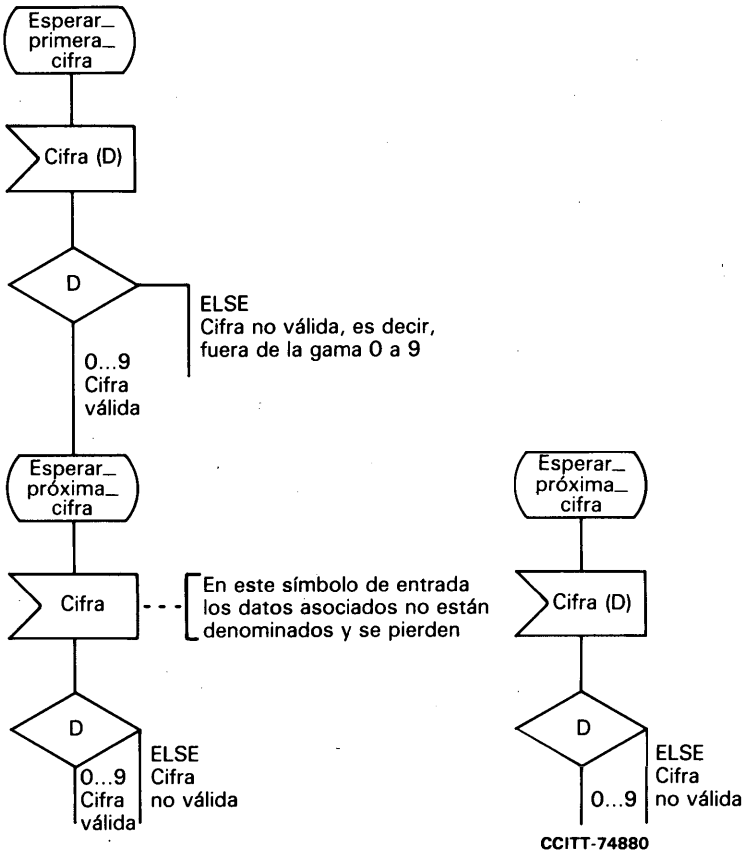
Observación 1 — No hay nombre de datos y el valor 9269 se pierde y no está disponible para el proceso receptor.

Observación 2 — El nombre de la señal (descolgado) debe corresponder al nombre de la señal de salida pero los nombres de datos pueden elegirse de modo que correspondan o no.

c)

FIGURA D-20

Ejemplos de recepción de datos en un proceso

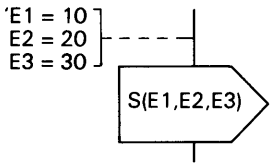


a) Muestra una versión incorrecta *b) Muestra una versión correcta*

Observación 1 — En a), la segunda decisión utilizará el valor de D obtenido de la primera cifra.
Observación 2 — En b), el valor de D será el de la cifra actual. Los valores de las cifras anteriores serán sobre escritos.
Observación 3 — Como D está declarado como un parámetro normal de cifra, el valor de D se almacena automáticamente.

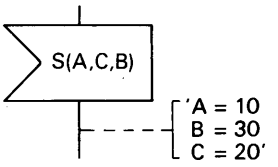
FIGURA D-21

Parte del proceso de recepción de una cifra



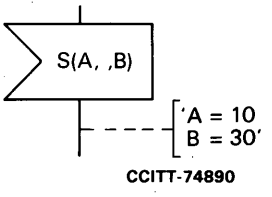
Observación — La señal de salida S tiene tres variables denominadas E1, E2 y E3. Estos elementos se relacionan con tres valores, por ejemplo, actualmente, 10, 20 y 30.

a)



Observación — La señal de entrada correspondiente S en el proceso receptor denomina estos elementos A, C y B respectivamente.

b)

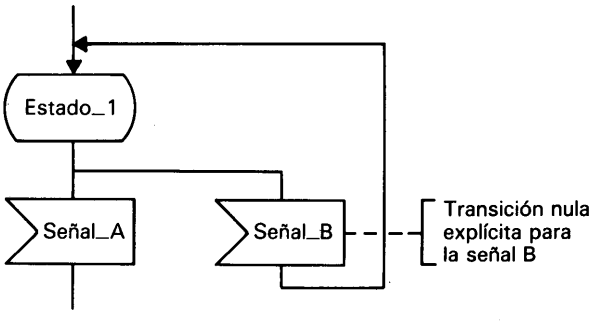


Observación — Esta señal de entrada sólo denomina dos variables. El valor de datos intermedio se pierde.

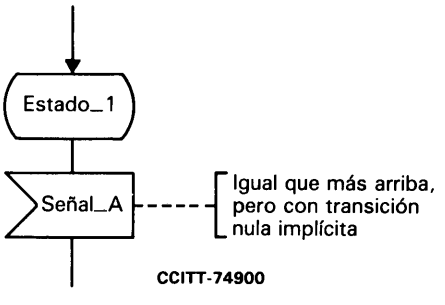
c)

FIGURA D-22

Señal con varios elementos de datos asociados



a) Transición nula explícita



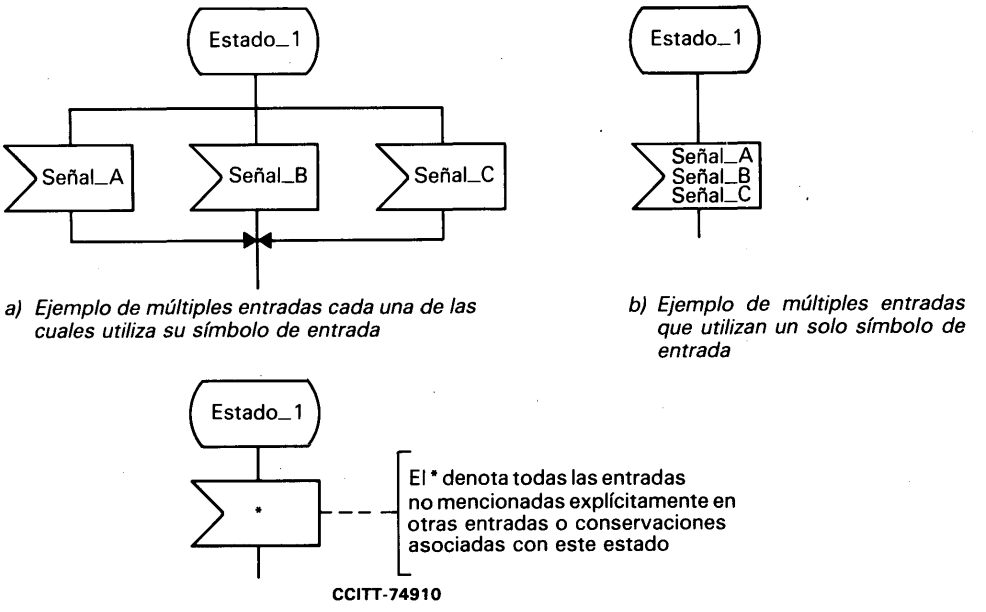
b) Transición nula implícita

Observación — Si hay datos asociados a la señal B, se pierden en ambos casos. Sin embargo, si se muestra un nombre de datos (por ejemplo, señal_B(x)) en el caso *a)*, se mantiene el valor del dato. En esta circunstancia los dos casos ya no son idénticos.

FIGURA D-23

Representación explícita e implícita de una transición «nula»

Cuando varias entradas conducen a la misma transición, todas las denominaciones de señales importantes para ese proceso pueden hacerse figurar dentro de un solo símbolo de entrada. La figura D-43 ilustra este punto; los dos diagramas de esta figura son lógicamente equivalentes. Si se adopta este convenio, las denominaciones de las señales deberán estar separadas por comas.



a) Ejemplo de múltiples entradas cada una de las cuales utiliza su símbolo de entrada

b) Ejemplo de múltiples entradas que utilizan un solo símbolo de entrada

c) Ejemplo de múltiples entradas que utilizan la notación asterisco

FIGURA D-24

Dos posibles representaciones de múltiples entradas

D.4.3.4.2.2 *Mecanismo implícito de cola de espera*

Puede haber una o más señales en espera de ser consumidas cuando un proceso llega a un nuevo estado. Esto entraña que, o bien las señales tienen que formar, de alguna manera, una cola de espera, o se pierden. Cuando una señal llega al bloque de destino, se la entrega a la cola de espera de entrada del proceso receptor. La semántica del LED define para cada proceso un mecanismo conceptual de cola de espera según el principio de «primero en entrar primero en salir», esto es, un proceso que considera las señales, con vista a su consumo, en el orden de llegada de éstas al proceso. Cuando un proceso llega a un estado, se le entrega una y sólo una señal de la cola de espera. Quiere decir que si la cola no está vacía, el proceso consume la primera señal de cola. Si está vacía, el proceso espera en el estado hasta que llegue una señal a la cola de espera, a continuación de lo cual la misma es consumida por el proceso.

La figura D-25 utiliza el mecanismo conceptual de cola de espera para explicar la operación del proceso LED representado cuando los tiempos de transición no son nulos. Debe señalarse lo siguiente:

- no se utiliza el concepto de conservación, y por consiguiente las señales son consumidas en el orden de llegada;
- el orden de llegada de las señales es importante. Si C hubiera llegado antes que B en la transición entre el estado 1 y el estado 2, la secuencia de los estados hubiera sido 1, 2, 3 en lugar de 1, 2, 4;
- como la cola no está vacía cuando el proceso llega al estado 4, el proceso no espera en ninguno de estos estados;
- no es posible asignar ninguna prioridad a una señal.

Si los tiempos de transición son nulos, toda señal será consumida en el momento en que llega a un proceso, a menos que se utilice la operación de conservación (véase el § D.4.3.4.3).

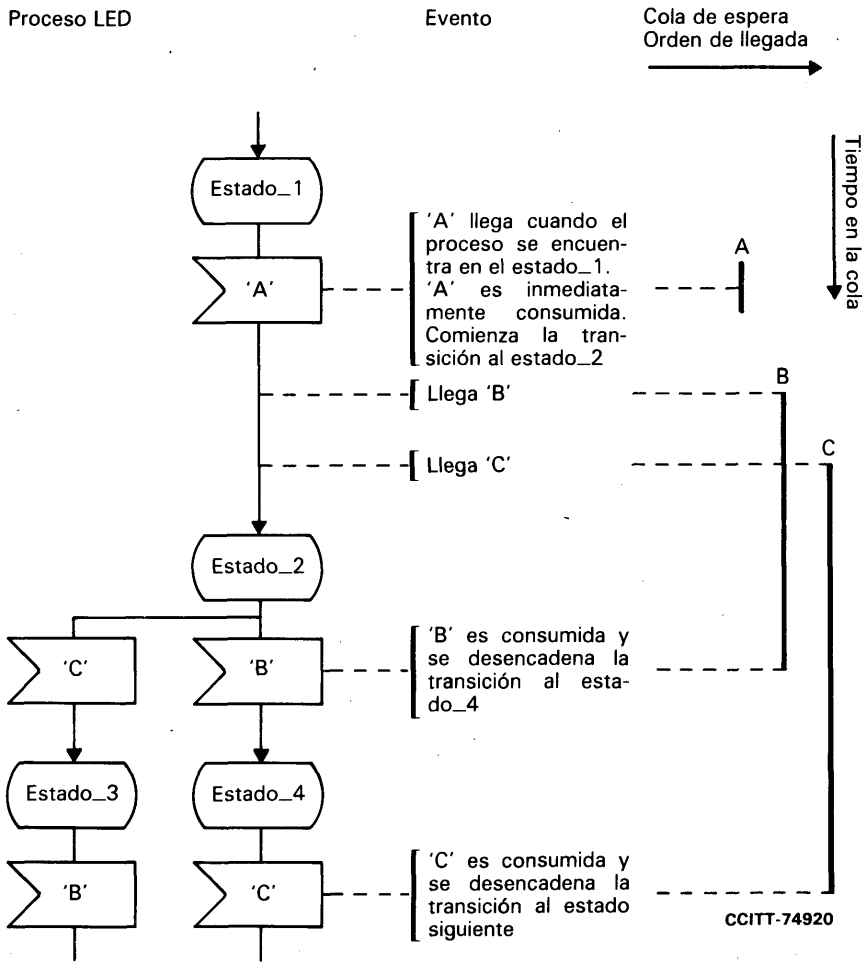


FIGURA D-25

Ejemplo de la operación del mecanismo implícito de cola de espera

D.4.3.4.2.3 *Recepción de señales que no aparecen normalmente*

En cada estado deben presentarse todas las señales posibles en forma implícita o explícita. Pueden producirse excepciones (señales inesperadas pero teóricamente posibles, señales no definidas o señales lógicamente erróneas en ese lugar, etc.) en casi todos los estados. Normalmente, el autor no presenta tales posibilidades, y ello conduce a la eliminación de la señal cuando llega. Sin embargo, si el autor desea incluir excepciones en su diagrama, *todos* los estados deberán ser ampliados con una entrada adicional.

Otra posibilidad consiste en utilizar los múltiples aspectos de un estado (véase el § D.6.3.6.5.1) junto con el símbolo «todo» (*) (véase el § D.6.3.6.21). Por ejemplo, si puede recibirse la señal A_COLGADO en todos los estados y si las acciones subsiguientes son idénticas, podrá utilizarse el método indicado en la figura D-26.

D.4.3.4.2.4 *Llegada simultánea de señales*

La Recomendación Z.101 prevé la posibilidad de que lleguen simultáneamente varias señales a un proceso e indica que se ordenarán arbitrariamente.

Si un usuario diseña un proceso que puede recibir señales simultáneas, debe cuidar de que el orden de llegada no pueda alterar el funcionamiento deseado del proceso.

El LED no prevé una prioridad de las señales; quiere decir que al llegar señales simultáneamente se elige una de ellas en forma indeterminista.

Si, cuando un proceso pasa a un estado, hay varias señales disponibles, sólo una señal es presentada al proceso e identificada como entrada. La semántica del LED implica que las otras señales quedan, de hecho, retenidas.

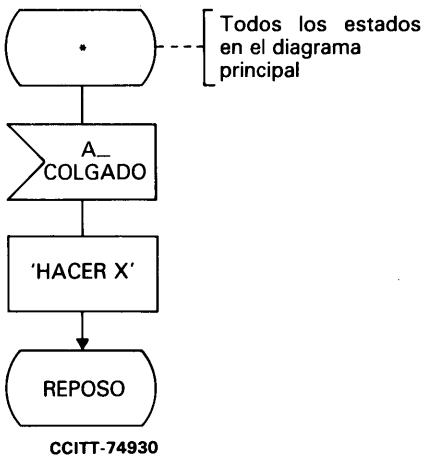


FIGURA D-26

Ejemplo de tratamiento de señales para que aparezcan en varios estados

D.4.3.4.2.5 *Entrada externa o interna*

Una señal externa es una señal entre procesos pertenecientes a bloques funcionales diferentes; una señal interna es una señal entre procesos pertenecientes al mismo bloque funcional. Desde el punto de vista semántico no hay diferencias entre las señales externas e internas, pero, en las Recomendaciones precedentes soble el LED se hacía una distinción sintáctica. Esas diferencias de la sintaxis se han eliminado y en la actualidad las señales internas y externas tienen sintaxis idénticas. La anterior sintaxis GR para una señal interna (dos líneas verticales en el símbolo) ya no se utiliza. Se permiten los antiguos diagramas que emplean señales internas, pero éstas son interpretadas del mismo modo que las señales externas.

D.4.3.4.2.6 *Identificación del proceso emisor*

Cada señal lleva consigo el identificador de instancia de proceso del proceso emisor. Cuando se recibe una señal, una variable del proceso llamada EMISOR toma el valor identificador de instancia de proceso del proceso emisor que está contenido en la señal. La figura D-27 muestra un ejemplo de la utilización de esta variable.

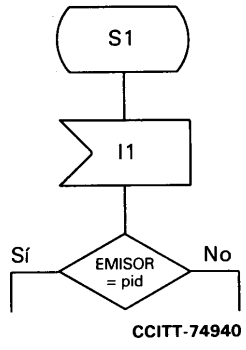


FIGURA D-27

Uso de la variable EMISOR

D.4.3.4.3 *Conservaciones*

El concepto de conservación permite aplazar el consumo de una señal hasta que hayan sido consumidas una o más señales que llegaron después de aquélla. Como se indicó en el § D.4.3.4.2, de no utilizarse el concepto de conservación, las señales son consumidas en el mismo orden en que llegan.

El concepto de conservación puede utilizarse para simplificar procesos cuando el orden relativo de llegada de algunas señales no es importante y el orden de llegada real es indeterminista.

En cada estado, toda señal se trata de una de las tres formas siguientes:

- se indica como un símbolo de entrada, o
- se indica como un símbolo de conservación, o
- se trata mediante una entrada implícita que conduce a una transición nula implícita.

La operación del mecanismo implícito de cola de espera presentado en el § D.4.3.4.2 abarca también el concepto de conservación. Las señales, al llegar, entran en una cola de espera y, cuando el proceso llega a determinado estado, las señales que están en la cola son examinadas una a una en el mismo orden en que llegaron. Una señal indicada en un símbolo de entrada, sea éste explícito o implícito, es consumida y se ejecuta la correspondiente transición. Una señal indicada en un símbolo de conservación no es consumida y permanece en la cola en la misma posición secuencial que tenía, y se pasa a considerar la señal siguiente en la cola.

La figura D-28 ilustra un ejemplo de un proceso LED que comprende un símbolo de conservación. Debe observarse que las señales S y R son consumidas en el orden R, S, es decir, en el orden inverso a aquél en que se recibieron. Un símbolo único de conservación sólo puede conservar una señal mientras el proceso se encuentra en el estado en que aparece el símbolo, y la conservación durará el tiempo que toma la transición al siguiente estado. En el estado siguiente, la señal será consumida mediante una entrada explícita o implícita a menos que se repita el símbolo de conservación con la denominación de la señal (como se indica en la figura D-28) o que haya en la cola implícita otra señal conservada, disponible para ser consumida antes que ésta (como se ilustra en la figura D-29).

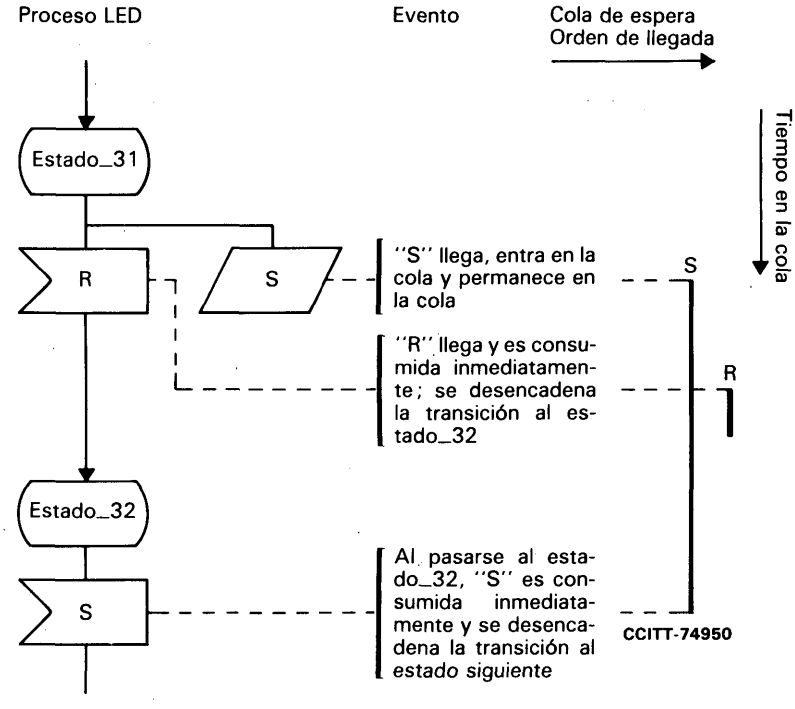


FIGURA D-28

Ejemplo de un diagrama LED con símbolo de conservación que ilustra la operación del mecanismo de cola de espera implícita

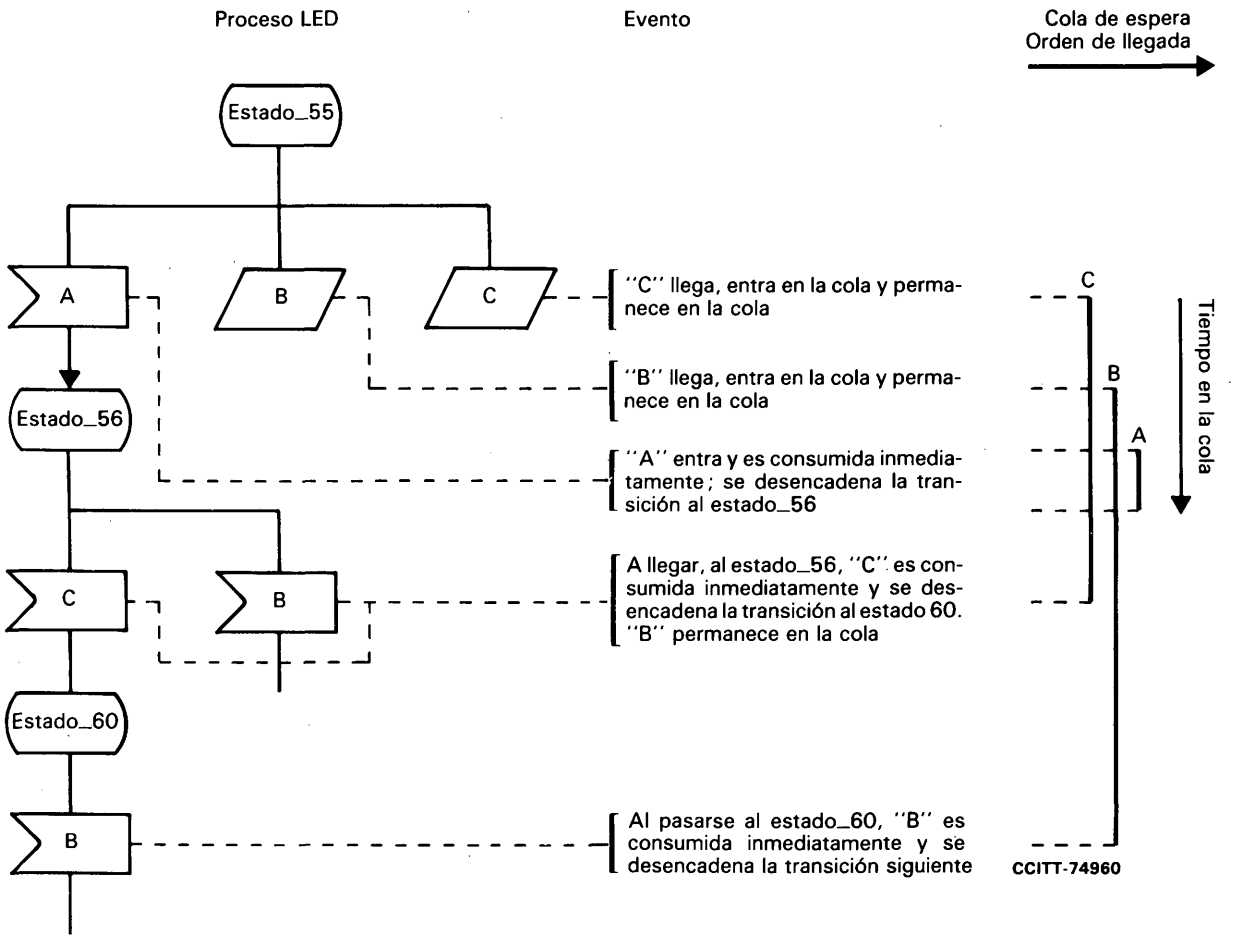


FIGURA D-29

Segundo ejemplo de utilización del símbolo de conservación

Una señal conservada se pone a disposición de un porceso solamente a través de un símbolo de entrada (explícita o implícita) correspondientes. En particular, una señal conservada no puede ser objeto de preguntas mediante una casilla de decisión antes de su identificación como entrada, ni tampoco están disponibles los datos asociados.

En un estado en que deba conservarse más de una señal, se puede, o bien emplear un símbolo de conservación para cada señal o inscribir todas las señales en un solo símbolo de conservación.

Si deben conservarse varias señales, la semántica del símbolo de conservación implica que se mantiene el orden de llegada.

Un tercer ejemplo del uso de la conservación se ilustra en la figura D-30; la operación del mecanismo conceptual de cola de espera se describe en la figura D-31.

El símbolo de conservación permite simplificar los diagramas. Por ejemplo, conservando una señal es posible evitar su recepción y tener que almacenar sus datos asociados hasta el próximo estado.

Aunque la conservación puede utilizarse en todos los niveles de descripción, en los niveles inferiores pudiera ser necesario describir el mecanismo real que ejecuta los conceptos de conservación.

Si no se pone cuidado en el uso de conservación, la fila de señales conservadas puede volverse demasiado grande o recordar señales demasiado largas, y podría consumirse entonces una señal antigua cuando lo que se necesitaba era una nueva.

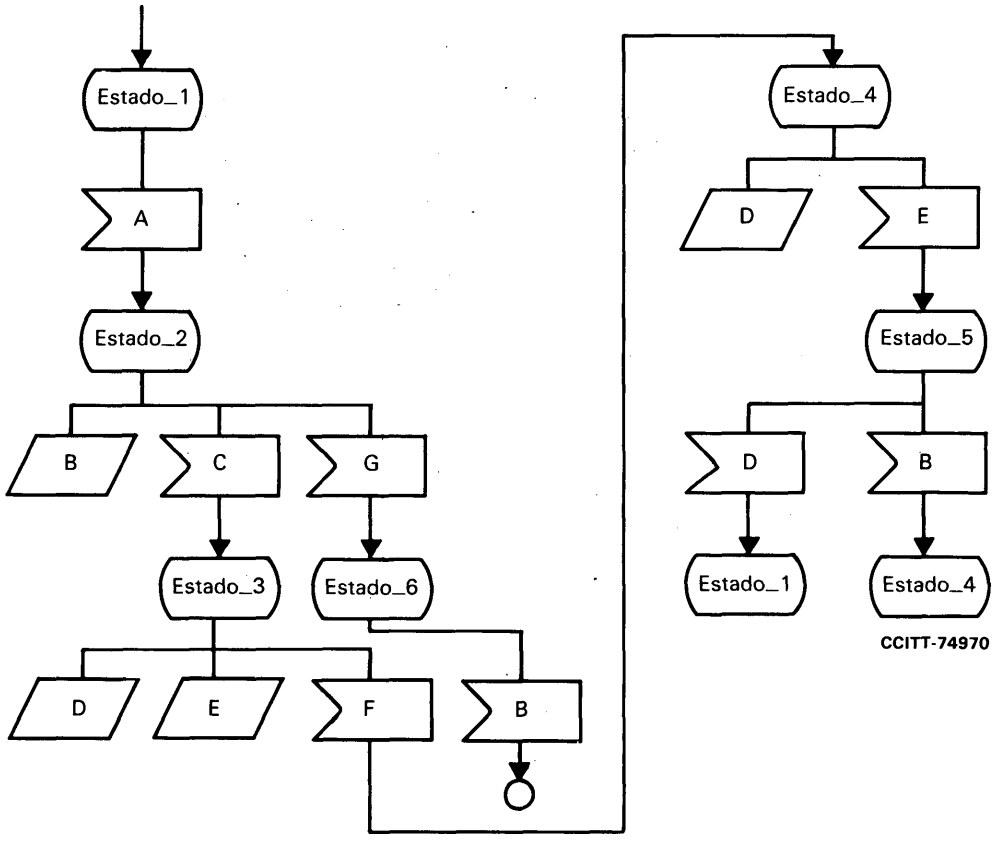


FIGURA D-30

Ejemplo de un diagrama LED más complejo con numerosas entradas y conservaciones

Estado vigente	Evento
Estado 1	(El proceso pasa al estado 1 estando en la cola las señales «A», «B», «C», «D», «E») La primera señal en la cola, «A», es consumida (entrada explícita) y se dispara la transición al estado 2
Estado 2	La primera señal en la cola, «B», aparece en un símbolo de conservación y permanece en la cola
Estado 2	La segunda señal, «C», es consumida (entrada explícita) y se dispara la transición al estado 3
Estado 3	La primera señal en la cola, «B», es consumida (entrada implícita) y se dispara una transición nula
	«F» llega y entra en la cola
Estado 3	(Al pasarse al estado 3 de nuevo) la primera señal en la cola, «D», aparece en un símbolo de conservación y permanece en la cola
Estado 3	La segunda señal, «E», aparece en un símbolo de conservación y permanece en la cola
Estado 3	La tercera señal, «F», es consumida (entrada explícita) y se dispara la transición al estado 4
Estado 4	La primera señal en la cola, «D», aparece en un símbolo de conservación y permanece en la cola
Estado 4	La segunda señal, «E», es consumida (entrada explícita) y se dispara la transición al estado 5
Estado 5	La primera (y única) señal en la cola, «D», es consumida (entrada explícita) y se dispara la transición al estado 1

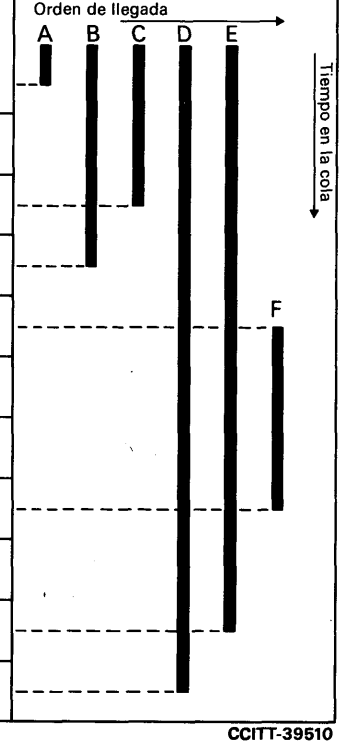


FIGURA D-31

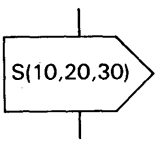
Operación del mecanismo conceptual de cola de espera

D.4.3.4.4 *Salidas*

Un símbolo de salida representa el envío de una señal de un proceso a otro. Puesto que el control de la recepción y del consumo de la señal está asociado al proceso receptor (véase el § D.4.3.4.2), la semántica relacionada directamente con el símbolo de salida es relativamente sencilla. Desde el punto de vista del proceso emisor, una salida puede considerarse a menudo como una acción instantánea que, una vez completada, no ejercerá influencia ulterior alguna en el proceso emisor, el cual no tendrá por qué conocer directamente el destino de la señal.

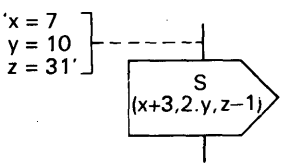
Si el autor tiene dificultades para decidir si una acción ha de representarse como una salida o una tarea, consultará el § D.4.3.4.6. Una acción de salida representa el envío de una señal y de los valores de datos asociadas, si existen. Los valores de datos pueden asociarse con una señal de salida colocando los valores entre paréntesis o colocando las expresiones que tienen valores entre paréntesis. (Véanse las figuras D-32 y D-22.)

Cada salida debe estar dirigida a una instancia de proceso específica. Como suele ser imposible conocer el identificador de instancia de proceso de ningún proceso en el momento de establecer una especificación o descripción, el método normal de direccionamiento de señales consiste en utilizar una variable o expresión en el símbolo TO o contraseña. Las figuras D-33, D-34 y D-35, muestran algunos ejemplos. En la figura D-33, el parámetro de proceso salida a recibe el valor de un identificador de instancia de proceso en el momento en que se crea el proceso. Se utiliza entonces salida a dentro del proceso como enlace entre este proceso y su proceso conectado. Al diseñar el sistema debe ponerse cuidado en garantizar que el tipo de proceso indicado por salida a sea capaz de recibir las señales enviadas. En la figura D-34 la variable del proceso incorporada se usa para retornar una señal al proceso que envió la señal recién recibida. En la figura D-35 la señal se dirige al proceso vástago más recientemente creado del proceso.



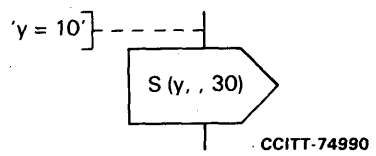
Observación — La señal S tiene tres valores, 10, 20 y 30 asociados a ella.

a)



Observación — Al interpretar S; x, y y z tienen (en este ejemplo) los valores 7, 10 y 31 respectivamente. S envía los valores 10, 20 y 30.

b)



Observación — Al interpretar S, y tiene (en este ejemplo) el valor 10. S envía el valor 10, un valor indefinido y 30, respectivamente.

c)

FIGURA D-32

Salida con datos asociados

```
PROCESO X; (salida_a Pld)
.
.
.
salida_SEÑAL DE SALIDA HACIA;
.
.
.
```

FIGURA D-33

Direccionamiento de señales por medio de sus parámetros formales

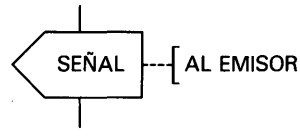


FIGURA D-34

Direccionamiento de señales de retorno al EMISOR

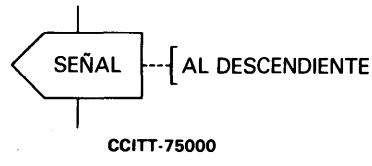


FIGURA D-35

Direccionamiento de una señal a un proceso descendiente

D.4.3.4.5 *Condiciones habilitadoras y señales continuas*

Las condiciones habilitadoras permiten la recepción condicional de señales, en base a la condición habilitadora especificada. Si la condición es verdadera, la señal es recibida y la transición es interpretada. Si la condición es falsa, la señal es conservada y el proceso permanece en el estado hasta que llega otra señal o hasta que la condición cambia de falsa a verdadera. Esto puede ilustrarse mediante el ejemplo de la figura D-36. El proceso comienza en el estado S1. En el proceso P2, el valor de X es 9. Cuando llega la señal B, el valor de X se compara con 10. Obsérvese que, dado que éste es un valor de otro proceso, hace falta una operación de importación para determinar su valor actual. Como X no es 10, el proceso permanece en S1 hasta que se recibe otra entrada o hasta que P2 envía un nuevo valor de X. En este ejemplo, llega A y causa una transición a S2. Durante la transición, X cambió a 11, de manera que ahora la condición asociada a la señal B en el estado S2 es verdadera. Como B es la primera señal de la cola de espera, se efectúa la transición que le sigue y el proceso termina en el estado S3.

Algunos atributos importantes de las condiciones habilitadoras son los siguientes:

1) La condición habilitadora es probada solamente cuando el proceso llega a un estado. Por ende, si el valor de X hubiese pasado de 9 a 11 y luego a 12 durante la ejecución de la transición que sigue a la recepción de A, el proceso habría permanecido en S2.

2) Las condiciones habilitadoras sólo pueden basarse en expresiones que contienen variables IMPORTADAS o locales y no variables VISTAS. Esto se debe al mecanismo subyacente utilizado para llevar a efecto la condición habilitadora. La condición habilitadora funciona utilizando señales para determinar cuándo ha cambiado un valor en la condición habilitadora. Para las variables locales, la condición es verificada antes de entrar en el estado. Como las variables son locales para el proceso, no pueden cambiar mientras el proceso esté en el estado. En consecuencia, la condición no necesita ser supervisada constantemente. Para las variables IMPORTADAS, el LED utiliza señales para cambios de las variables IMPORTADAS como elemento desencadenador para saber cuándo hay que verificar la condición habilitadora. Las variables VISTAS acceden al valor de la variable directamente. Como no hay envío de señales, una vez que el proceso haya entrado en un estado, el proceso no tendría ninguna manera de verificar el valor de la variable.

3) Aunque es posible utilizar más de una condición habilitadora por estado, no se permite utilizar más de una condición habilitadora para una misma señal. Así pues, la condición mostrada en la figura D-37 no está permitida. Si hacen falta múltiples condiciones para una determinada señal, las mismas pueden combinarse en una expresión booleana, como se ve en la figura D-38. Esto elimina la ambigüedad que se produce cuando dos o más condiciones son verdaderas al mismo tiempo. En la figura D-38, el usuario ha decidido que la condición de X tiene más prioridad que la condición de Y. Por ende, si ambas fuesen verdaderas, se habría seguido el trayecto C1. También en este caso ello se debe a que el sistema LED subyacente no asocia prioridades a las señales.

Las señales continuas tienen las mismas propiedades básicas que las condiciones habilitadoras, con la excepción de que no tienen ninguna señal asociada. Así, al entrarse al estado, ninguna señal en la cola de espera puede causar una transición, se verifican las señales continuas y, si una es verdadera, se ejecuta la transición que le sigue. Esto puede ilustrarse con el ejemplo de la figura D-39. Inicialmente el proceso está en el estado S1 y el valor de X es 9. Cuando llega la señal A, causa la transición a S2. Durante la transición, X cambió a 11. Como no había ninguna otra señal en la cola de espera, se espera la transición a S3.

Algunos atributos importantes de las señles continuas son los siguientes:

1) Al igual que para las condiciones habilitadoras, el valor de la condición se verifica solamente al llegarse a un estado.

2) También en el caso de las señales continuas, sólo pueden utilizarse variables IMPORTADAS y locales.

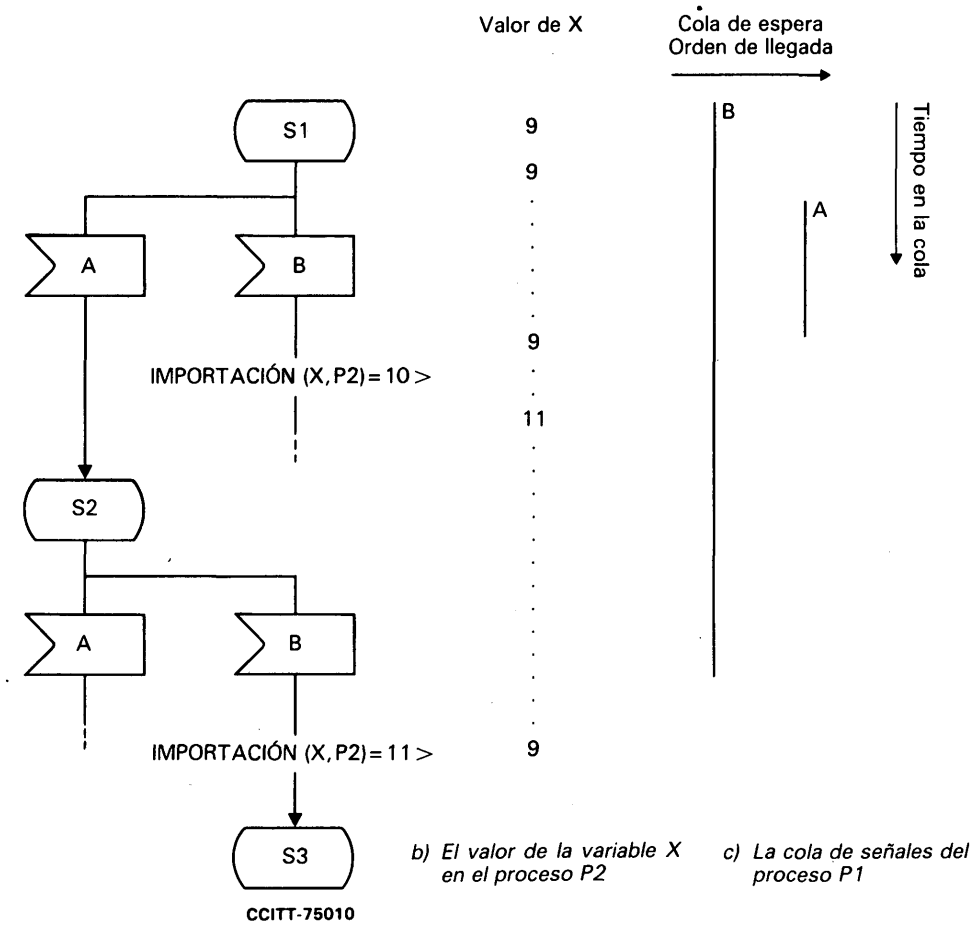
3) Se permiten señales continuas múltiples para cada estado. Cuando hay más de una señal continua asociada a un estado, se probará primero la señal continua que tenga el número de prioridad más bajo. Dos señales continuas, cualesquiera que sean, no pueden tener el mismo número de prioridad. En todos los casos, la prioridad de la señal continua es más baja que la de cualquier otras señal. Las señales continuas constituyen el único elemento en que se utilizan prioridades en el LED. Esto también se debe al sistema subyacente del LED; no obstante, la manera en que se modelan las señales continuas en el LED, mediante el empleo de las señales enviadas al importar variables, permite utilizar prioridades para las señales continuas y, de hecho, lo hace necesario a fin de impedir ambigüedades cuando hay más de una señal continua presente. Esto se ilustra en la figura D-40. Inicialmente el proceso está en el estado S1 y sus variables locales tienen los valores 10 para X y 11 para Y. Como ambas señales continuas son verdaderas, se elige la que tiene la prioridad más alta (número de prioridad más bajo) y se ejecuta la transición a S2. En S2, la condición de Y ya no es verdadera y por ello, si se quiere la prioridad de la señal continua para X es menor que la correspondiente a Y, se efectúa la transición que le sigue y el proceso llega al estado S3.

D.4.3.4.6 *Tarea*

Se utiliza una tarea para representar operaciones respecto a datos efectuadas en una transición, con la excepción de la generación de señales de salida y de decisiones. Los datos sólo pueden ser modificados por el proceso que los posee. La naturaleza de las tareas que aparecen en cualquier diagrama estará determinada por la naturaleza del proceso descrito y el grado de detalle necesario. Los siguientes son ejemplos de acciones que pueden tratarse por tareas:

a) acciones del equipo físico, como por ejemplo el envío del tono de ocupado;

b) la manipulación de datos.



a) Parte del proceso P1

b) El valor de la variable X en el proceso P2

c) La cola de señales del proceso P1

FIGURA D-36

Condición habilitadora

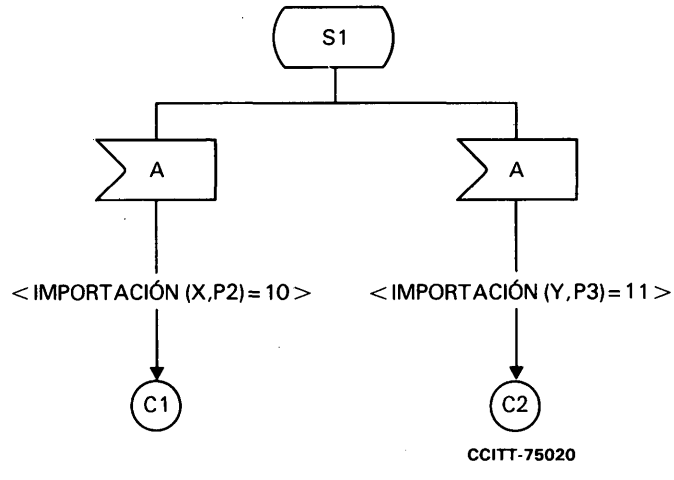


FIGURA D-37

Condición habilitadora ilegal

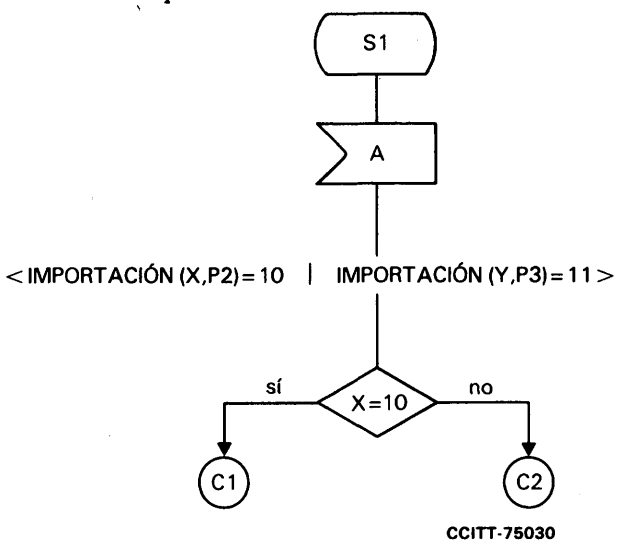
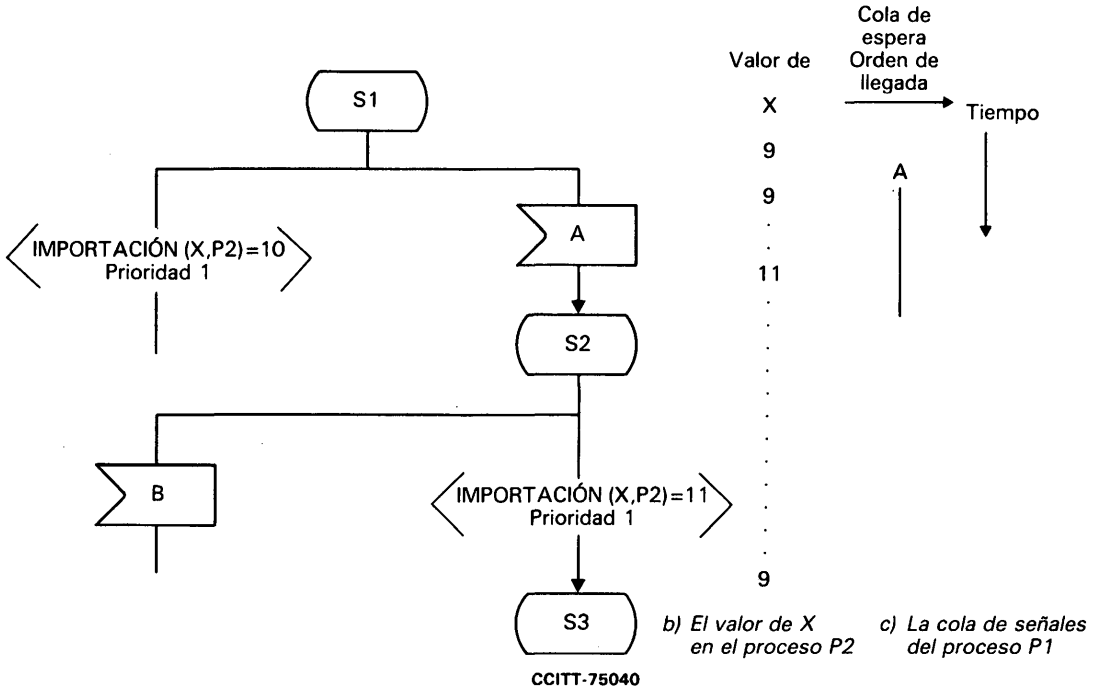


FIGURA D-38

Solución correcta para la figura D-37



a) Parte del proceso P1

FIGURA D-39

Señales continuas

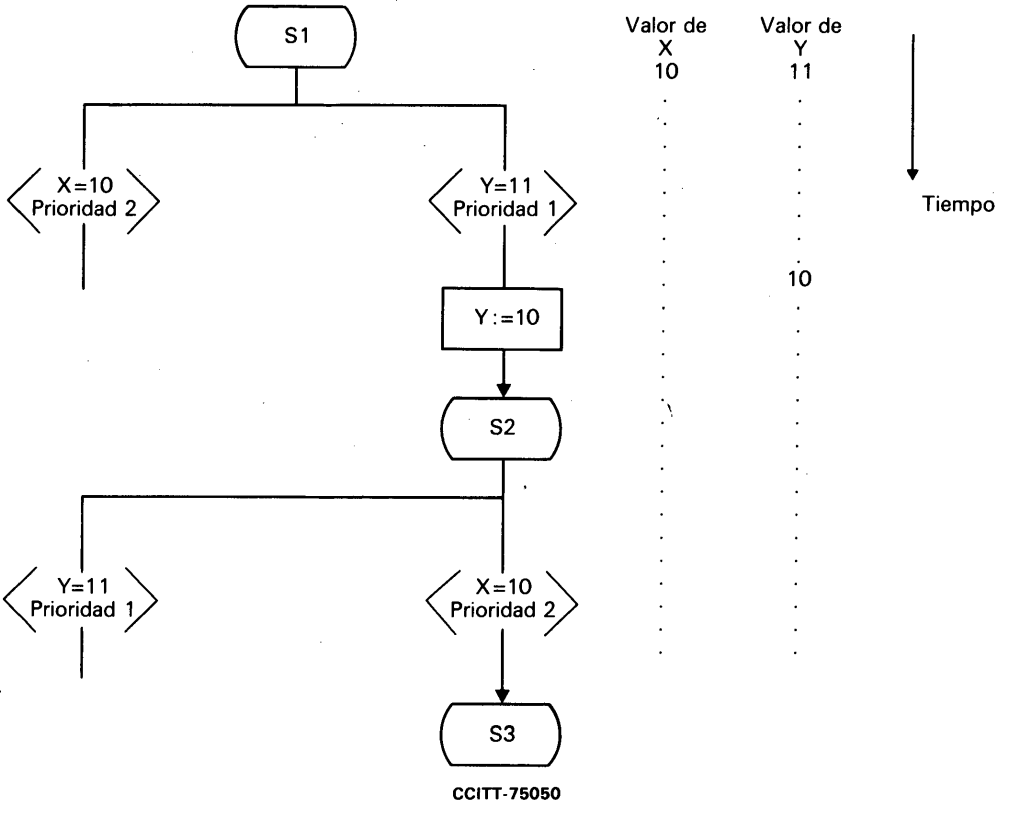


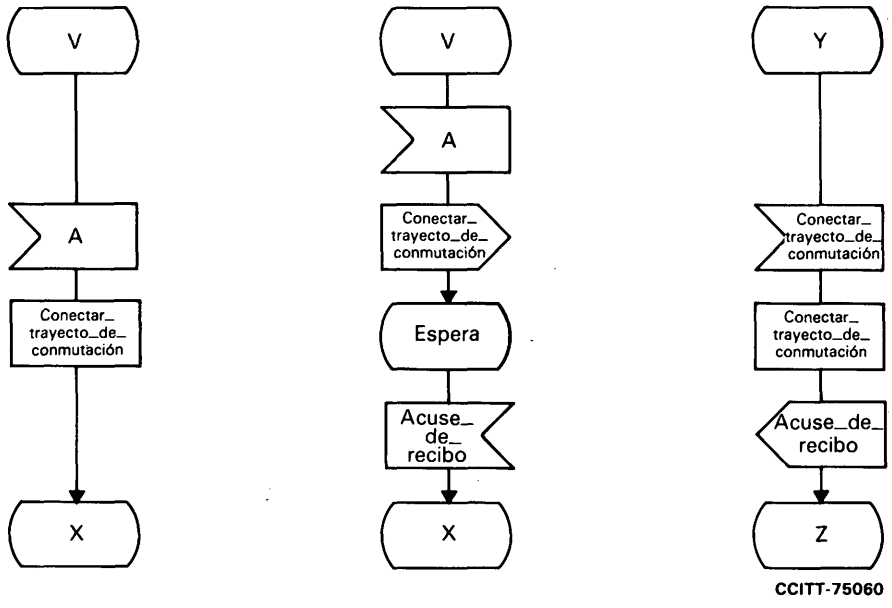
FIGURA D-40

Señales continuas con prioridad

Puede que los usuarios del LED experimentes dificultades a la hora de decidir si algunos aspectos del sistema que se define deben representarse por una tarea o por un proceso separado. Considérese el proceso mostrado en la figura D-41 ¿la acción «conectar trayecto de conmutación» debe representarse como una tarea o como un proceso separado? Si no se ha identificado un proceso separado de control del trayecto de conmutación, el símbolo de tarea sería apropiado (figura D-41 a). Si se ha ientificado tal proceso separado, hay que utilizar señales que comuniquen con el proceso de control (figura D-41 b).

D.4.3.4.7 *Decisiones*

Una decisión es una acción, dentro de una transición, en virtud de la cual se hace una pregunta respecto al valor de elementos de datos disponibles para el proceso en el instante de ejecutar la acción. El proceso continúa por uno de los dos o más trayectos que siguen a la decisión, de acuerdo con la respuesta. Los autores de diagramas LED deben cerciorarse de que los procesos queden definidos de tal modo que no puedan tratar de ejecutar decisiones para las cuales no se dispone de respuesta (o datos); tales decisiones invalidarían el diagrama y podrían causar una confusión considerable.



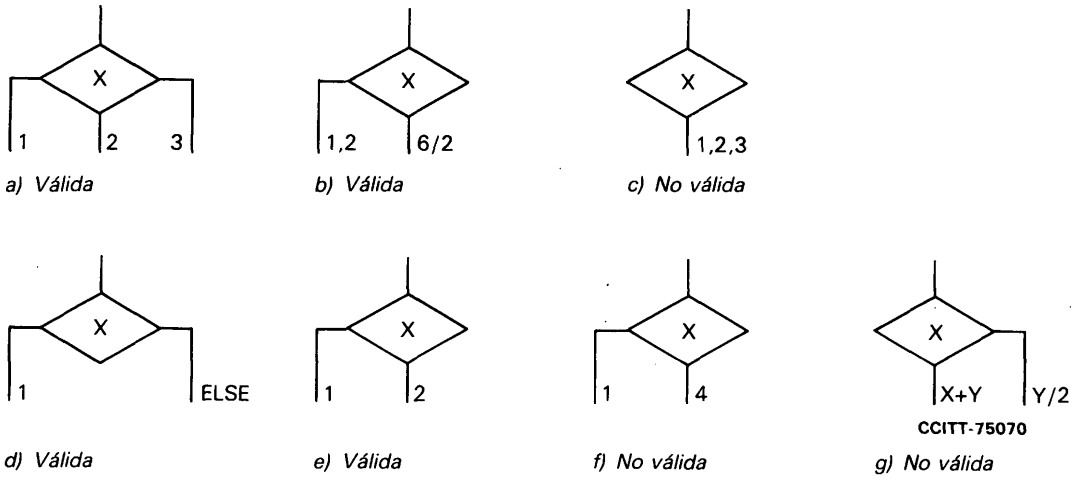
a) Solución con un solo proceso

b) Solución con dos procesos

FIGURA D-41

Dos posibles soluciones para «conectar trayecto de conmutación»

En la figura D-42 se muestran algunos ejemplos del uso de decisiones. La variable X en estos ejemplos puede tener los valores 1, 2 ó 3.



a) Válida

b) Válida

c) No válida

d) Válida

e) Válida

f) No válida

g) No válida

FIGURA D-42

Ejemplos del uso de decisiones

La posibildad c) no es válida pues la decisión no va seguida de por lo menos dos trayectos. La posibilidad e) es válida ya que en el LED existe la convención de que, si es imposible decidir en una disyuntiva, la misma se interpretará como una alternativa implicita que conduce directamente a un símbolo de parada. La posibilidad f) no es válida porque una de las alternativas está fuera de la gama. La posibilidad g) no es válida pues las condiciones de la alternativa contienen variables y por consiguiente no pueden evaluarse en forma estática.

Si una respuesta conduce a una decisión en la misma transición, deben realizarse algunas acciones que influyen en la pregunta de las decisiones. Sin embargo, incluso con esa norma pueden crearse puntos muertos, como se indica en la figura D-43. Por consiguiente ha de prestarse siempre atención cuando se tengan respuestas que conduzcan a una decisión de la misma transición.

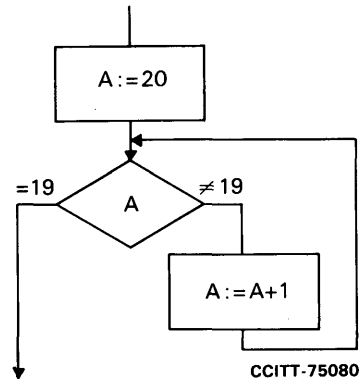


FIGURA D-43

Ejemplo del uso legal de una decisión que crea un punto muerto

Pueden adoptarse decisiones utilizando cualquier valor disponible actualmente en el proceso. Ello incluye:

- datos almacenados por una tarea;
- datos recibidos por una entrada;
- datos establecidos cuando se creó el proceso (datos transmitidos como un parámetro real);
- datos compartidos.

En una decisión, la pregunta puede pedir el valor de una expresión, y la expresión puede comprender constantes y cualquiera de los tipos de valores de datos antes indicados.

D.4.3.5 *Procedimientos*

Los procedimientos son una técnica para añadir estructura a procesos. En numerosos aspectos, los procedimientos del LED son similares a los de los lenguajes de programación, pero se advierte al lector que hay algunas diferencias importantes y que hay que proceder con precaución. Dicho esto, podemos comenzar la descripción de los procedimientos.

Desde el punto de vista sintáctico, un procedimiento es muy similar a un proceso. El gráfico de un procedimiento tiene un nodo de comienzo que difiere muy poco del nodo de comienzo de un proceso, y un nodo de retorno en lugar de un nodo de parada. Desde el punto de vista semántico, es bastante diferente, aunque los nodos idénticos de los procesos y procedimientos pueden tener el mismo significado.

Pueden aparecer llamadas de procedimiento en cualquier punto donde se permite un nodo de tarea en el gráfico de un proceso o de un procedimiento. En cierto modo, un procedimiento puede ser interpretado como una tarea, con las siguientes excepciones:

1) Un procedimiento puede contener estados y, por ende, recibirá señales. Así, una parte de la definición del procedimiento prevé un conjunto de conservaciones adicionales que enumera las señales que han de conservarse cuando se llama al procedimiento. Como pueden definirse procedimientos a nivel del sistema, es importante enumerar todas las señales que deben conservarse en cualquiera de los procesos que llaman el procedimiento.
2) Un procedimiento puede enviar señales de salida; el identificador de instancia del proceso originador es el identificador de instancia de proceso del proceso que ha llamado el procedimiento.
3) Un procedimiento tiene sus propias variables locales y no tiene acceso a ninguna de las variables del proceso, a menos que las variables se hayan transmitido al procedimiento como parámetros de ENTRADA/SALIDA.
4) A fin de que el procedimiento pueda ser llamado en varios lugares, se permiten parámetros de SEÑAL para crear sinónimos de los nombres reales de señal para el proceso llamante y para los nombres de señal usados en el procedimiento.

Cuando se llama un procedimiento, se crea el entorno del procedimiento y el procedimiento comienza a ser interpretado. La interpretación del procedimiento continúa hasta que se llega al nodo de retorno. MIentras se interpreta el procedimiento, todas las señales dirigidas al proceso son ya sea conservadas o recibidas por el procedimiento. El procedimiento no tiene su propia cola de espera de entrada, sino que utiliza la cola de espera de entrada del proceso que lo ha llamado. De ahí la importancia de disponer de un conjunto de conservaciones adicionales apropiado para un procedimiento. El conjunto de conservaciones adicionales se construye dinámicamente cuando se llama el procesamiento, y consiste en todas las señales de entrada válidas del procesamiento llamante que no se indican como parámetros para la llamada. Si una señal no está en ese conjunto ni en el conjunto normal de señales conservadas correspondiente al nodo de estado del gráfico del procedimiento, será descartada.

En el § D.9 puede encontrarse un ejemplo del uso de procedimientos. El mismo ilustra su empleo para la especificación de protocolos.

D.4.3.6 *Expresión del tiempo en el LED*

En varios lugares del LED puede hacer falta representar el tiempo:

a) activación de un proceso en un determinado momento (tiempo absoluto o tiempo relativo);
b) tiempo consumido en la ejecución de acciones de transición;
c) tiempo necesario para transferir una señal de un proceso a otro, quizás por conducto de un canal.

D.4.3.6.1 *Temporizadores y periodos de temporización*

Para medir el tiempo y pedir periodos de temporización en un sistema se recurre a temporizadores y a un conjunto de operaciones respecto a ellos.

El «temporizador» es de un tipo previamente definido, y debe declararse toda utilización de temporizadores en un proceso. Pueden efectuarse operaciones de «INICIALIZACIÓN» y «REINICIALIZACIÓN» respecto a los temporizadores. La operación de INICIALIZACIÓN pide que tenga lugar un periodo de temporización en un estante especificado, y la operación de REINICIALIZACIÓN anula cualquier periodo de temporización pedido. (Adviértase que una operación de INICIALIZACIÓN incluye implícitamente una operación de REINICIALIZACIÓN de cualquier periodo de temporización que no ha expirado en el temporizador.)

```
DCL Temporizador T1;
INICIALIZAR (AHORA + 20 segundos, T1);
REINICIALIZAR (T1).
```

FIGURA D-44

Ejemplo de declaración y operación de un temporizador

En la operación de INICIALIZACIÓN debe especificarse un tiempo absoluto. El tiempo relativo se transforma en un valor de tiempo absoluto añadiendo la operación «AHORA», que arroja al tiempo actual. Por ejemplo: 20 segundos desde ahora se expresa así: AHORA + 20 s.

Los temporizadores son supervisados únicamente cuando el proceso se halla en un estado. Se coloca una señal en la cola de espera de entrada del proceso cuando se verifica un temporizador y el mismo es igual o inferior al tiempo actual. El temporizador es entonces reinicializado implícitamente.

La señal de periodo de temporización tiene el mismo nombre que el temporizador y no transporta ningún valor de datos.

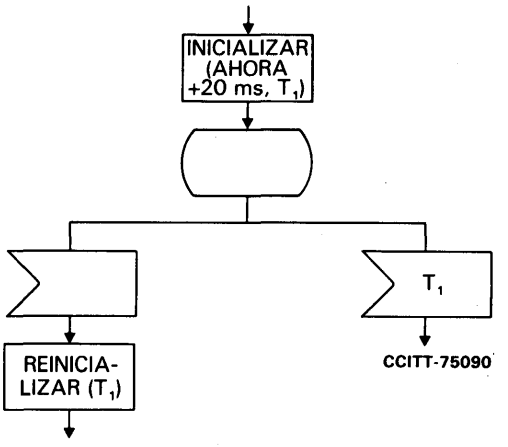


FIGURA D-45

Ejemplo del uso de temporizadores

D.4.3.6.2 *Especificación del tiempo consumido por acciones*

Cuando se utiliza el LED para simular el funcionamiento de un sistema, es preciso especificar el tiempo consumido en las transiciones. Aunque el LED actual no prevé esto, pueden formularse las siguientes directrices:

En tales casos se asocia al nombre de la acción, como parámetro, un valor de tiempo (el periodo de tiempo consumido en la ejecución de una determinada acción). Este parámetro facultativo puede identificarse por una palabra clave (por ejemplo: TIEMPO CONSUMIDO = valor de tiempo).

a) Una acción que dura 32 ms se representa por un símbolo de tarea que incluye el consumo de tiempo

b) La misma acción representada en LED/PR

FIGURA D-46

Indicación de tiempo de transición

La indicación del tiempo en forma de una parámetro asociado al nombre, en lugar de un comentario, se debe al hecho de que el comentario debe proporcionar una explicación a seres humanos y no necesita ser comprendido por una máquina.

D.4.3.6.3 *Tiempo de transferencia de una señal*

La expresión del tiempo que transcurre entre la generación (salida) y el consumo (entrada) de una señal es muy compleja pues depende de varios factores.

Puede ser un tiempo absoluto consumido por la acción de transporte de la señal de un bloque a otro y esta parte puede expresarse (si el canal está representado en forma explícita) mediante la suma de los tiempos consumidos por las acciones efectuadas por el canal.

Existe un consumo de tiempo relativo dependiente de la carga del sistema y de la estrategia de manipulación de la señal. El tiempo gastado en la activación del proceso de recepción depende también de las características de funcionamiento del sistema y de la prioridad del proceso, la carga del sistema, los acontecimientos simultáneos, etc. Este tiempo sólo puede suponerse y tiene importancia únicamente en los estudios relativos a la capacidad del sistema.

Otro tiempo que no puede determinarse estáticamente es el gastado en colas de espera hasta el consumo de las señales precedentes.

Normalmente, este tiempo no es fácil de representar en forma estática y sólo tiene interés para los estudios de simulación y capacidad.

Sin embargo, si todas las acciones en el nivel más detallado de descripción del sistema tienen una indicación del consumo de tiempo y si la descripción contiene una representación de los canales (sistema de funcionamiento), es posible simular el sistema y evaluarlo desde el punto de vista de la capacidad por distintas técnicas.

D.4.4 *Texto asociado a construcciones LED*

Las convenciones que se exponen seguidamente en lo tocante al texto asociado a construcciones LED se aplican a ambas versiones GR y PR.

Un texto en LED puede ser:

- un texto formal, o bien
- un texto informal, o bien
- comentarios.

Un texto formal es un texto que puede ser interpretado formalmente. Un texto informal sólo puede ser interpretado informalmente. Los comentarios son solamente anotaciones y no pueden ser interpretados. El significado de un diagrama debe ser el mismo, con o sin comentarios.

D.4.4.1 *Texto formal*

Para representar un texto formal se utiliza la sintaxis LED/TR. En el LED/PR, las combinaciones de texto permitidas están definidas en la sintaxis. En el LED/GR, el texto formal asociado a una construcción gráfica debe estar contenido ya sea en ese símbolo o en un símbolo conexo de extensión del texto (véase el §D.6.3.6.19).

En un texto formal se pueden mezclar letras mayúsculas y minúsculas libremente. Da igual que un nombre esté representado en mayúsculas o en minúsculas, pero se recomienda utilizar las letras mayúsculas y minúsculas de una manera consecuente.

D.4.4.1.1 *Nombre*

Un nombre representa la identificación de la entidad a la que está asociado dentro del contexto. Se requiere un nombre para:

BLOQUE, PROCESO, SEÑAL, CANAL, ESTADO, CONSERVACIÓN, ENTRADA, SALIDA, MACRO, CREACIÓN, COMIENZO, PROCEDIMIENTO y CONECTOR.

Un nombre debe comenzar con una letra y puede contener letras, cifras, caracteres nacionales y subrayados. No se permite la existencia de espacios.

Ejemplos de nombres legales:

PROCESADOR_DE_LLAMADA
TONO_DE_LLAMADA

Ejemplo de nombres ilegales:

COMENZAR TASACIÓN (contiene un espacio)
1982 (comienza con una cifra)

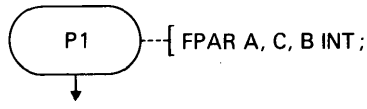
El nombre ENTRADA identifica la SEÑAL recibida y, viceversa, el nombre SALIDA identifica la SEÑAL enviada, de modo que SALIDA – SEÑAL – ENTRADA conforman el mismo nombre. Los conectores (JOIN en PR) tienen un nombre asociado. Este nombre puede comenzar con una cifra. Los conectores son también peculiares en el sentido de que no tienen una cadena de texto asociada.

Todas las palabras clave en LED/PR son palabras reservadas y no pueden usarse como nombres. Todas las palabras clave reservadas se indican en el resumen del LED/PR.

D.4.4.1.2 *Parámetros formales*

Se utilizan parámetros formales en las sentencias o en los símbolos de comienzo de proceso y de procedimiento.

Los parámetros formales se indican por la palabra clave «FPAR».



CCITT-75110

FIGURA D-47

Ejemplo de parámetros formales

Véase el § D.4.3.5 sobre el uso de parámetros formales.

D.4.4.1.3 *Parámetros reales*

Se utilizan parámetros reales en construcciones tales como la petición de creación, llamada de procedimiento, salida, etc. Los parámetros reales están delimitados por un par de paréntesis.

SALIDA cifras (a, b, c, d)

FIGURA D-48

Ejemplo de parámetros reales en una salida

D.4.4.1.4 *Sentencias y expresiones*

El texto en las tareas y decisiones se ajusta a la sintaxis LED/PR para las sentencias y expresiones.

D.4.4.1.5 *Definiciones y declaraciones*

Las definiciones de señal, las definiciones de tipo de datos y las declaraciones de variable se representan siempre por la sintaxis LED/PR para dichas construcciones.

D.4.4.2 *Texto informal*

El texto informal asociado a símbolo GR o sentencias PR sólo puede ser interpretado de una manera informal. Para el texto informal se utiliza la sintaxis LED/PR para cadenas de texto, o sea que el texto va encerrado entre un par de apóstrofes.

Ejemplos de cadenas de texto:

'Está libre el abonado'

'$a^2 + 2ab + b^2$'

Obsérvese que en la cadena de texto se permite cualquier carácter, con excepticón (apóstrofe). Si se requiere un ' puede representarse por un par de ".

Ejemplo:

John's house.– 'John"s house'

Si se utiliza únicamente texto informal en toda la representación LED, pueden omitirse los apóstrofes. En tal caso, debe indicarse esta circunstancia al comienzo de la representación LED.

El texto informal puede tener cualquier forma adecuada para presentar la información. En este caso el autor da por sentada la existencia de alguna base común de comprensión entre él mismo y el lector. Hay que tener en cuenta que el lector puede ser una máquina. En este sentido, el texto informal debe ser, desde el punto de vista del lector, lo suficientemente formal para que pueda comprenderse en forma inequívoca (figura D-49).

Ejemplos de textos informales:

- TAREA 'hacer mientras x < 5 ... '
- ENTRADA x 'contiene la identidad del abonado'
- DECISIÓN 'abonado ocupado?'
- (ocupado):

FIGURA D-49

Ejemplos de texto informal

El texto CHILL asociado a sentencias LED es un ejemplo de texto «informal» desde el punto de vista del LED, pese a que constituye una manera de representar acciones que es muy formal.

Existe una distinción clarísima entre texto informal y comentarios.

Los comentarios se introducen con el objeto de ayudar a los lectores a comprender la representación. Su ausencia o presencia no altera la representación.

El texto informal forma parte de la representación propiamente dicha, por lo que no podremos comprender ésta sin considerar el texto asociado (tanto formal como informal).

D.4.4.3 *Comentarios*

Los comentarios en LED no tienen ningún significado formal o informal. Se introducen con el único objeto de facilitar a los lectores la comprensión del diagrama GR o del texto PR.

En LED/GR se pueden insertar comentarios en cualquier parte de un diagrama. Se reconocen los comentarios por estar asociados a un símbolo de comentario o por estar delimitados por '/*' y '*/' (figura D-50).

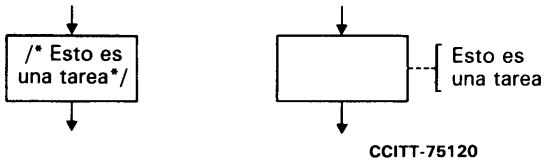


FIGURA D-50

Ejemplos de comentarios en GR

En PR, los comentarios están ya sea identificados por la palabra clave 'COMENTARIO' o delimitados por '/*' y '*/'.

```
TAREA A:=2 COMENTARIO esto es una tarea;
ó
TAREA A:=2 /* esto es una tarea */;
```

FIGURA D-51

Ejemplos de comentarios en PR

D.4.5 *Situaciones indefinidas*

D.4.5.1 *Generalidades*

En la vida de los sistemas, desde el comienzo hasta la retirada del servicio, se producen normalmente fallos. Algunos de ellos (y en el caso de los sistemas de equipo lógico la mayor parte de ellos) se deben a situaciones indefinidas en la especificación. Por consiguiente, importa verificar una especificación LED con objeto de descubrir el mayor número posible de ellas antes de la realización. Más tarde, cuando se ha llevado a la práctica una especificación y se ha establecido una descripción, debe verificarse si la descripción corresponde al sistema en un 100% – ¡ni más ni menos!

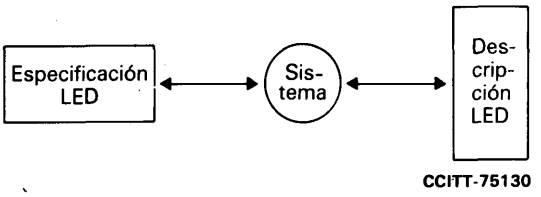


FIGURA D-52

Correlación uno a uno entre especificación, sistema y descripción

D.4.5.2 *Ejemplos de situaciones indefinidas en el LED*

D.4.5.2.1 Un objeto de datos que trata de asignar un valor fuera de su gama o un valor que no pertenece a su tipo; por ejemplo, la variable NÚMERO_DE_ABONADO tiene el tipo GAMA_DE_ENTEROS (100 000 : 999 999) y trata de asignar el valor FALSO, 1 000 000, 87 ó BUSY.

D.4.5.2.2 Se envía una señal a un proceso inexistente.

D.4.5.2.3 Una señal enviada a una instancia de proceso (recién) muerta. Sin embargo, puede que no sea un error del sistema, pero en la especificación/descripción LED es una situación indefinida que debe tratarse por el sistema subyacente/programa de simulación.

D.4.5.2.4 Una señal direccionada a un canal inexistente.

D.4.5.2.5 En la descripción LED también habrá un error en el comportamiento del sistema físico si se ha obtenido una correlación uno a uno.

D.4.5.2.6 Direccionamiento de señales a un canal o a un proceso sin haber declarado convenientemente las señales, el canal o el proceso.

D.4.5.2.7 Una decisión para la que la respuesta o los datos no están disponibles cuando el proceso trata de ejecutar la decisión.

D.4.5.2.8 Una prueba para ejemplificar concretamente un proceso para el cual se ha alcanzado ya el número máximo de instancias.

D.4.6 *Directrices relativas a los datos primitivos en LED*

La noción de datos avanzados se trata en el § D.4.7.

D.4.6.1 *Consideraciones generales*

En el LED, la comunicación entre procesos tiene lugar mediante señales, valores compartidos y valores exportados/importados.

Un proceso inicia una señal por medio de una salida. Esta señal constituye un flujo de datos que aporta información a otro proceso particular. Si el proceso receptor reconoce la señal, por medio de una entrada, los datos contenidos en esta señal quedan a la disposición de este proceso.

Un proceso puede leer un valor de datos de otro proceso si pertenece al mismo bloque y si esos datos se declaran como valor compartido.

Además, un proceso puede revelar un elemento de datos declarándolo exportable. Todos los demás procesos que tienen una declaración de importación correspondiente reciben, a petición, copias del valor de datos.

Desde el punto de vista de un determinado proceso, y en un determinado instante, existen dos categorías principales y complementarias de datos:

1) datos disponibles para este proceso;
2) datos no puestos a la disposición de este proceso. (Esta segunda categoría incluye los datos retenidos pero no puestos aún a disposición del proceso en forma de señal entrante).

Un proceso sólo puede utilizar datos disponibles para ejecutar una de sus acciones: decisiones, tareas o salidas.

Los datos propiamente dichos, cuando se utilizan en una decisión o en una salida, en este nivel de abstracción, no se modifican. Una tarea especial puede, sin embargo, crear, almacenar, modificar o destruir datos.

La figura D-53 (para una decisión), la figura D-54 (para una salida) y la figura D-55 (para una tarea) presentan ejemplos de tratamiento de datos.

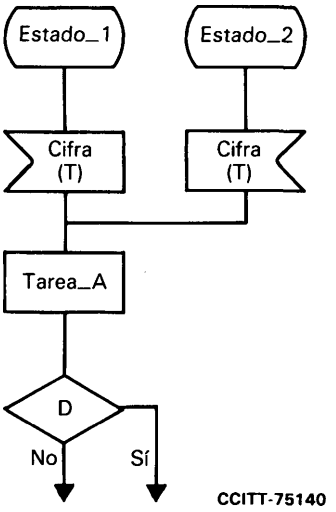
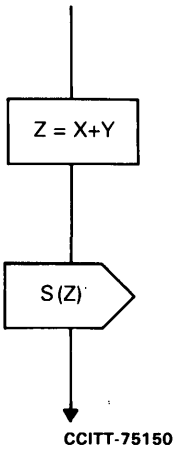


FIGURA D-53

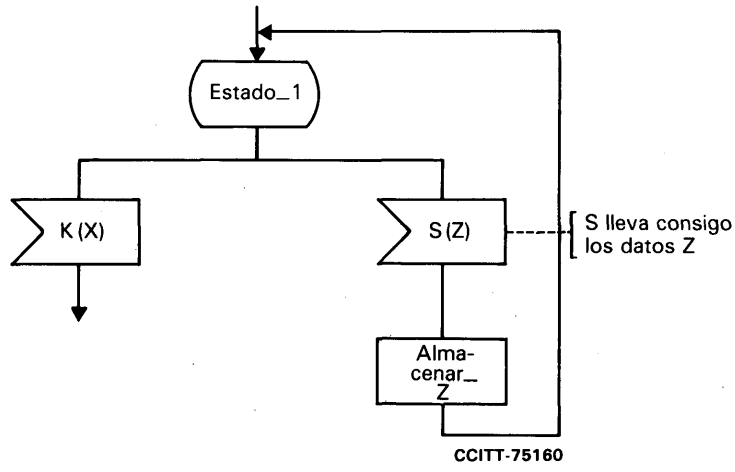
Utilización de una decisión para interrogar sobre datos asociados con entradas



Observación — S es el nombre de la señal. Z es el dato asociado.

FIGURA D-54

Envío de datos mediante una salida



Observación — S y K son nombres de señales. Z y X son los datos asociados.

FIGURA D-55

Almacenamiento, mediante una tarea, de datos entrantes para futura utilización

Existen los conceptos de valores compartidos y valores importados/exportados, que pueden utilizarse en lugar de señales para el intercambio de información.

D.4.6.2 *Tratamiento de datos dentro de un proceso*

Existen tres medios posibles de comunicación entre procesos, esto es, vía SEÑALES, VALORES COMPARTIDOS o VALORES IMPORTADOS/EXPORTADOS. Estas tres técnicas presentan notables diferencias. Por ejemplo, puede suceder que un IMPORTADOR de ciertos elementos de datos no pueda obtener la revelación de los mismos elementos de datos como VALORES COMPARTIDOS.

Los datos son locales respecto a una instancia de proceso, lo que significa que todos los datos tienen una — y sólo una — instancia de proceso propietario.

La instancia del proceso propietario puede modificar el valor real de los datos.

Ejemplo: Si los elementos de datos a y d son números enteros y locales respecto al proceso P, los ejemplos en la figura D-56 de la secuencia de acciones son legales dentro de P:

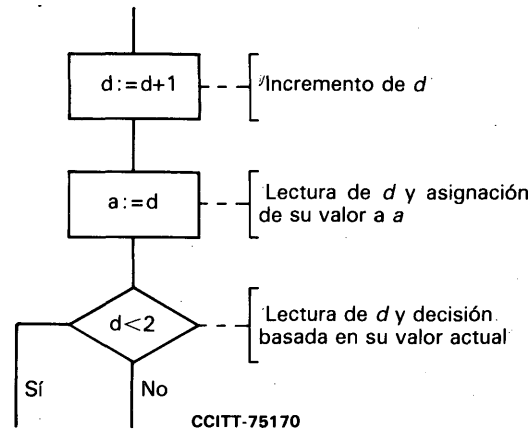


FIGURA D-56

Ejemplo de acciones respecto a datos locales en un proceso

D.4.6.3 *Tratamiento de datos entre procesos*

Dos procesos pueden intercambiar datos de otro modo que por medio de señales. Un proceso puede leer un valor de datos de otro proceso si pertenece al mismo bloque y si estos datos se declaran como «valor compartido» (véase el § D.4.6.3.1). Además, un proceso puede leer un valor de datos de otro proceso si esos datos se declaran como «valor exportable» (véase el § D.4.6.3.2). El proceso que no es propietario de los datos no puede modificar su valor, pero puede hacer una copia local susceptible de manipulación al igual que todos los demás elementos de datos locales.

Un elemento de datos no puede ser a la vez importado y compartido.

D.4.6.3.1 *Lectura de valores compartidos*

Un proceso puede revelar una parte o la totalidad de sus valores de datos declarándolos "valores compartidos", lo que tiene por efecto que todos los demás procesos (que pertenecen al mismo bloque que el proceso revelador) tendrán la posibilidad de leer el valor compartido, igual que si este valor compartido fuera un valor local. Ello significa que el valor que está viendo un proceso de visión, es siempre el mismo valor que el que ve el proceso revelador.

Existen algunas limitaciones respecto a lo que puede revelarse.

D.4.6.3.2 *Lectura de valores exportables*

Un proceso puede declarar una parte o la totalidad de sus datos como "datos exportables", lo que tiene por efecto que todos los demás procesos (con una "definición de datos importables" correspondiente) quedan facultados para obtener una copia del valor de datos después del tiempo de petición. Ello significa que la copia del valor de datos, entregada a un proceso importador a petición, permanece constante incluso si el valor de datos es modificado en un momento ulterior por el proceso propietario exportador.

D.4.6.3.3 *Ejemplo de diferencias entre el uso de valores compartidos y exportables*

En el ejemplo de la figura D-57 existe una sola instancia del proceso 1. Si puede producirse más de una instancia debe preverse algún medio de identificación exclusiva del elemento de datos *d* pertinente.

D.4.6.3.4 *Valores compartidos*

D.4.6.3.4.1 *Consideraciones generales*

El usuario del LED puede observar que la definición de datos compartidos permite especificar con facilidad la comunicación entre dos procesos. Sin embargo, al realizar sistemas así *especificados* se plantean algunos problemas, y la presente sección está destinada a guiar a los usuarios para que eviten y resuelvan tales problemas. Es menos probable que resulte difícil la *descripción* LED de sistemas que aplican datos compartidos, porque los problemas se habrán superado en la realización y será posible traducir la solución elegida al LED.

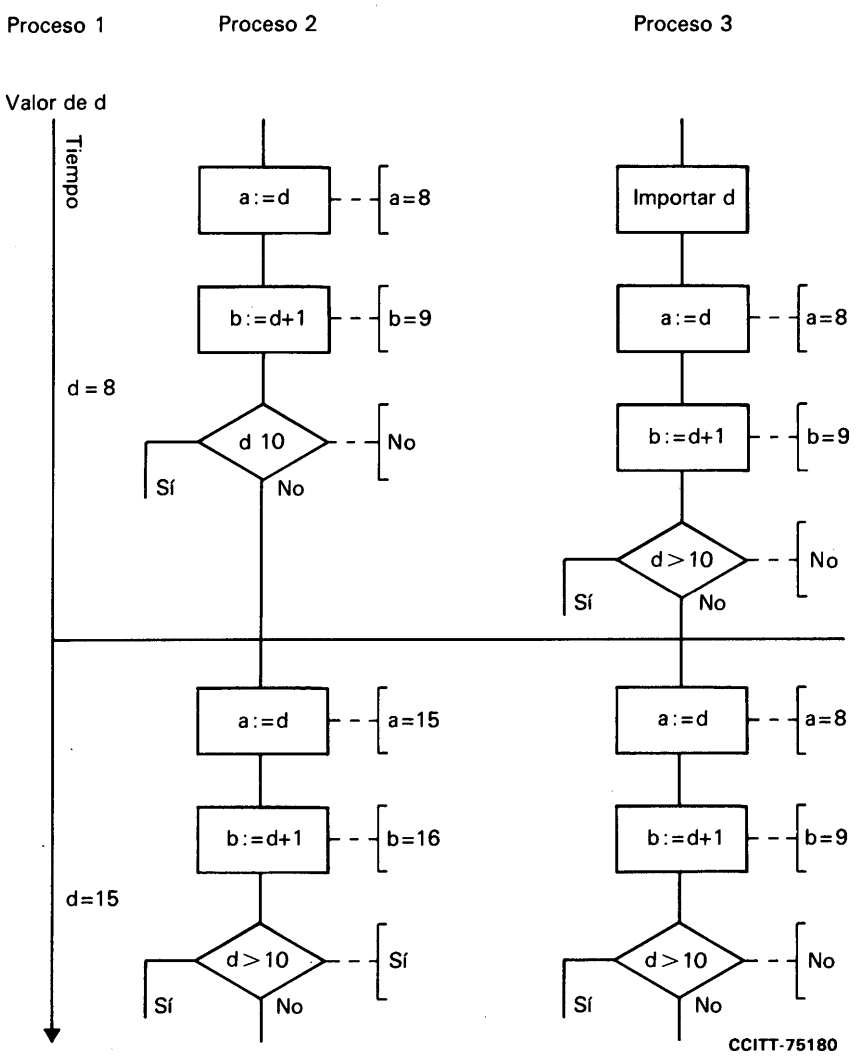
En el resto de la presente sección se supone que un proceso A posee datos que revela a un proceso B que los ve.

D.4.6.3.4.2 *Problemas de creación*

El resultado de la tentativa de ver datos en un proceso antes de crear el proceso y los datos se traduce en un error en el LED. El usuario puede evitar ese problema de dos modos:

- cuidando de que la distancia del proceso revelador A sea creada y haya inicializado los dtos pertinentes antes que la instancia del proceso de visión B; o bien
- cuidando de que B no efectúe transiciones que utilicen datos compartidos hasta que A haya sido creado y haya inicializado los datos pertinentes.

Un modo sencillo de realizar el primer caso consiste en hacer que A sea el padre (o un ascendiente) de B, o en dejar que A sea creado al mismo tiempo que el sistema (creación implícita). En este último caso puede disponerse que la transición pertinente de B sólo puede ser desencadenada por una señal procedente de A.



Observación — El proceso 1 es el proceso propietario de d. d es compartido con el proceso 2 y se hace exportable. El proceso 3 tiene d como un valor importable.

FIGURA D-57

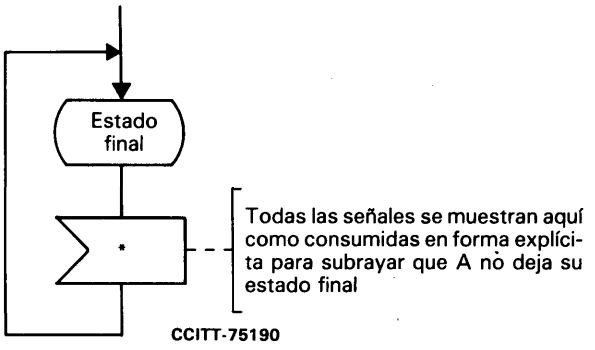
Ejemplo de diferencias en el uso de valores compartidos y exportables

D.4.6.3.4.3 *Problemas relacionados con la parada del proceso*

El proceso A ya no podrá revelar datos después de alcanzar un símbolo de parada. Toda tentativa de ver datos sería entonces un error en el LED. El usuario puede evitar este problema de dos modos:

- ya sea no utilizando en absoluto un símbolo de parada en A. En la figura D-58 aparece el proceso A en estado latente con datos compartidos visibles pero que no pueden modificarse más;
- o bien cuidando de que B esté al tanto de que A está a punto de parar y no efectúa ninguna otra tentativa de ver los datos pertinentes. En la figura D-59 se presenta un ejemplo, con utilización de señales.

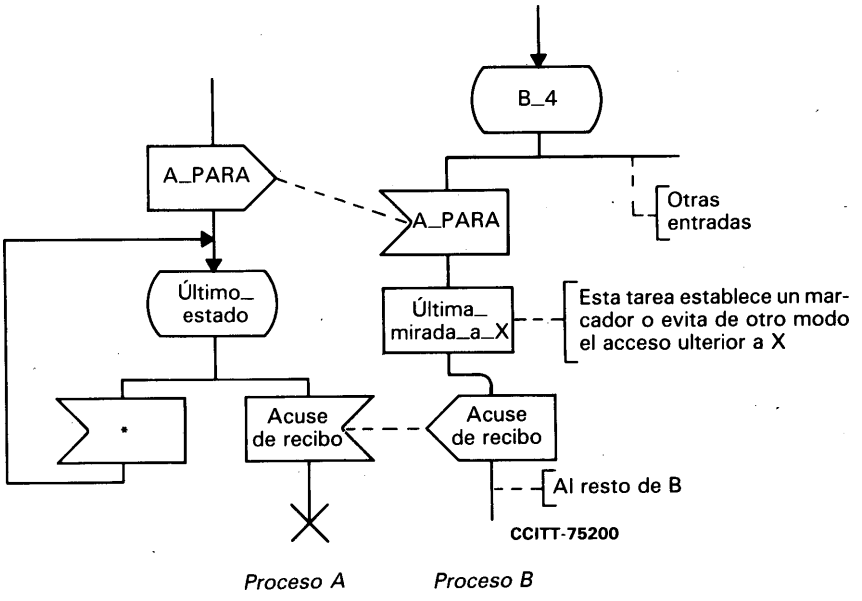
Para el realizador, la primera solución tiene el inconveniente de que A no ha liberado la memoria de datos que estaba usando.



Observación — El proceso A tiene datos X que se revelan.

FIGURA D-58

Un proceso que posee un estado latente es todavía capaz de revelar datos, pero no puede cambiarlos



Observación — El proceso A tiene datos X que revela al proceso B.

FIGURA D-59

El proceso A notifica al proceso B y obtiene el acuerdo antes de parar

D.4.6.3.4.4 *Problemas debidos a instancias múltiples de B*

Si el proceso B tiene simultáneamente varias instancias vivas, por ejemplo, B1, B2, B3 ... todas las cuales ven los datos de A, las soluciones dadas a los problemas anteriores en el § D.4.6.3.4.2 necesitan una ligera modificación.

El modo más fácil de evitar que cualquier caso de B trate de ver a A antes de que A exista y haya inicializado los datos pertinentes, será que A sea el padre de todas las B.

Otra posibilidad, si A tiene que enviar señales a cada B, es que sólo pueda hacerlo después de haber sido notificado por el padre (B) de la existencia y del identificador de proceso de cada B.

Existe también una ligera modificación de la solución del paro del proceso del § D.4.6.3.4.3. Es evidente que la solución del estado latente de la figura D-58 sigue siendo correcta.

La solución de la figura D-59 ya no es satisfactoria. Puede construirse una solución del siguiente tenor:

a) A existe antes del padre de B;

b) A es informado por el padre de B enviando una señal cada vez que se crea una nueva B;

c) A es informado por cada B enviando una señal cuando va a parar – o en todo caso cuando no verá de nuevo los datos de A;

d) A lleva la cuenta del número de B que están vivas y que ven o tienen la posibilidad de ver;

e) A envía una señal al padre de B para detener la creación de más B y obtiene un acuse de recibo de que no se crearán más;

f) cuando la cuenta de las B que pueden ver llega a cero, A puede parar.

Esto se muestra en la figura D-60 en relación con el proceso A.

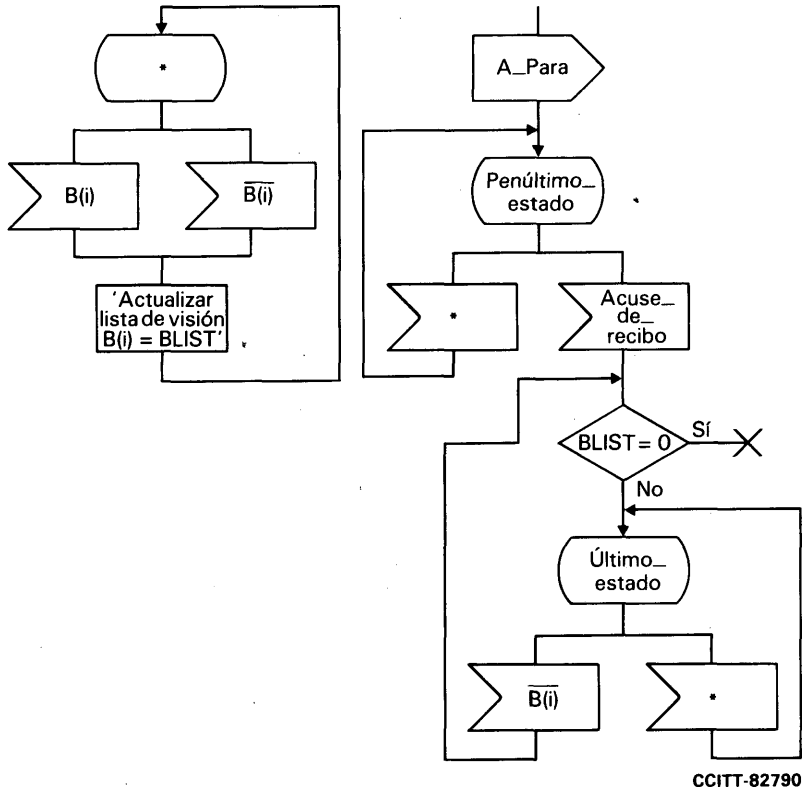


FIGURA D-60

Dos diagramas del proceso A

D.4.6.3.4.5 *Problemas debidos a instancias múltiples de A*

Si el proceso A tiene simultáneamente varias instancias vivas, esto es, A1, A2 ... y B desea ver datos, B tiene que especificar cuál A posee los datos pertinentes. Esto significa que B debe conocer la identidad de proceso de la instancia A pertinente y debe tener la seguridad de que existe la instancia A cuyos datos han de verse.

Puede establecerse una solución del problema en los siguientes términos:

a) el padre de A envía una señal a B que contiene el identificador de proceso de uno de sus descendientes (por ejemplo, A6) que posee los datos que B desea ver;

b) B ve los datos.

c) A6 envía una señal a B para informar a B de que A6 va a parar y que los datos no se verán más;

d) B acusa recibo contestando a A6 con una señal;

e) A6 para.

D.4.6.3.4.6 *Problemas debidos a instancias múltiples de A y B*

El caso práctico más probable de instancias múltiples de los procesos A y B se produce cuando la relación de compartición de datos tiene lugar entre pares de As y Bs. Los problemas señalados en los § D.4.6.3.4.2 y D.4.6.3.4.3 pueden evitarse aplicando lo siguiente:

a) Tan pronto como se crea una nueva instancia de A, como es Ai, la misma crea una nueva instancia de B, como es Bj. Ai conocerá entonces la identidad de proceso de Bj como descendiente y Bj conocerá a Ai como ascendiente;

b) Bj ve los datos;

c) cuando Bj ha terminado de ver los datos por última vez, envía una señal a su padre Ai y para (o continúa con otras transiciones);

d) Ai para.

D.4.6.3.4.7 *Compartición de varios elementos de datos*

Considérese el caso en el que A posee varios elementos de datos, como X1, X2 y X3, todos los cuales han de ser vistos por B. En aras de la sencillez supongamos que hay una sola instancia del proceso A y una sola del proceso B.

Si A actualiza X1, X2 y X3 mientras B está tratando de verlos, puede suceder que B obtenga una mezcla de valores nuevos y antiguos, esto es, un conjunto incoherente.

En algunos sistemas, la coherencia es más importante que el hecho de poseer valores actualizados, y en tales casos deben aplicarse medios para lograr esa coherencia. Se consideran aquí dos posibilidades:

a) En lugar de compartir X1, X2 y X3, definimos una estructura compuesta X, que tiene X1, X2 y X3 como campos y/o subestructuras.

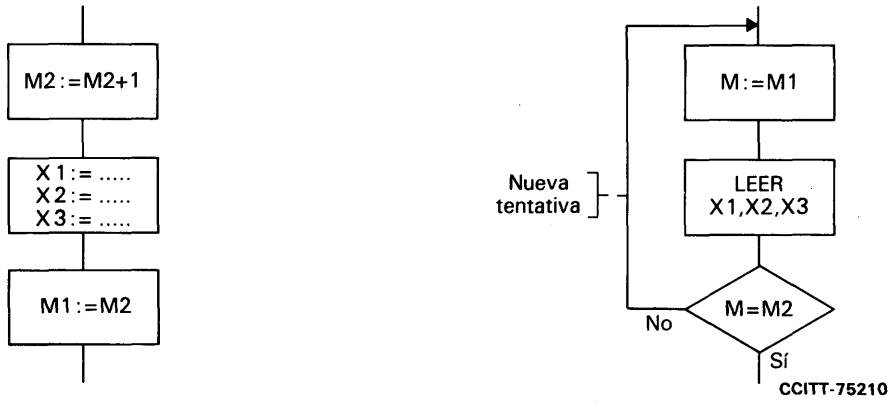
X es revelado por A y visto por B.

A obtiene ahora valores actualizados de X1, X2 y X3 y, cuando están todos actualizados, asigna todos los subcampos nuevos a X en una sola asignación (suponiendo que el LED tome la asignación como atómica).

De ese modo, nunca puede verse un conjunto incoherente.

Sin embargo, el problema no ha quedado realmente resuelto, sino que se ha transferido simplemente al realizador, porque en la mayoría de las realizaciones las asignaciones son sólo atómicas si los campos están en una palabra de máquina.

b) Para resolver el problema planteado, necesitamos especificar alguna forma de cierre en el LED. La figura D-61 presenta un ejemplo sencillo, exento de punto muerto, que exige la introducción de dos marcadores M1 y M2.



a) Proceso A

Actualización de X1, X2, X3. La importancia de la secuencia, primero M2, después las X y después M1, queda subrayada utilizando tareas separadas

b) Proceso B

Lectura de X1, X2, X3. Nueva tentativa si halla que M1 difiere de M2

Observación — El proceso A revela M1, X1, X2, X3, M2 al proceso B. M1 y M2 son del tipo ENTERO y se fijan inicialmente en cero.

FIGURA D-61

Cierre para lograr valores coherentes

Los marcadores deben realizase de modo que puedan modificarse en forma atómica (esto es, en una palabra).

El realizador debe examinar la frecuencia y el tiempo necesarios para la actualización, investigar la probabilidad de tentaivas repetidas y cuidar de que obtiene el comportamiento buscado.

D.4.6.3.4.8 *Conclusiones*

Los párrafos anteriores ponen de manifiesto algunos problemas en la especificación de sistemas que utilizan datos compartidos y facilitan ciertos ejemplos sobre el modo de evitarlos. Ni los problemas ni las soluciones expuestos son exhaustivos y los usuarios pueden encontrar otros medios de evitar esos problemas, así como otros problemas que evitar. En cualquier caso, es conveniente elegir soluciones que estén en armonía con otros aspectos del sistema en curso de diseño a fin de reducir la complejidad. Se aconseja a los usuarios que analicen con cuidado el comportamiento de los sistemas que emplean datos compartidos.

Por último, tal vez conviene señalar que la mayoría de esos problemas se evitan mediante el uso de señales para enviar datos.

En el LED, los valores de datos se consideran "indivisibles", esto es, un objeto de datos tiene un valor u otro dentro de su gama. Cuando se realizan objetos de datos, una realización puede modificar solamente una parte de un objeto complejo por vez. Cuando esos objetos son compartidos, el realizador debe prever mecanismos para garantizar la coherencia (por ejemplo, cierres, etc.).

Observación – Este problema es evitado normalmente por medio de datos exportables, porque el proceso propietario no será capaz de responder a una demanda en el momento de cambiar los datos (cuando los datos pueden ser incoherentes). Un valor de datos se modifica totalmente en una transición y, como el proceso sólo puede interrumpirse en un estado, los datos serán coherentes. Queda garantizada así la «exclusión mutua».

D.4.7 *Directrices relativas a los datos avanzados en LED*

D.4.7.1 *Definición de tipos de datos*

D.4.7.1.1 *Generalidades*

El LED ofrece cierto número de tipos de datos predefinidos. Pero los usuarios pueden definir nuevos tipos de datos. Una vez definidos, estos tipos pueden usarse al declarar variables, señales, etc. de la misma manera que los tipos predefinidos.

Al igual que la mayoría de los conceptos LED, el mecanismo de definición de tipo puede utilizarse en varios niveles de abstracción, desde una definición informal de la sintaxis básica hasta una definición totalmente formal de la sintaxis y semántica del tipo.

Los tipos pueden ser "parametrizados", obteniéndose así "bloques de construcción" mediante los cuales es posible crear una clase de tipos aún más amplia. Estos se conocen por el nombre de generadores de tipo y, a semajanza de los procesos LED, deben ser instanciados para crear uno o más tipos similares con propiedades ligeramente diferentes. El tipo incorporado Matriz (built-in type Array) es un ejemplo de ello pues su «contenido» se suministra únicamente cuando se utiliza.

La facilidad consistente en definir nuevos tipos de datos debe utilizarse cuando resulta esencial una especificación sumamente rigurosa y exenta de ambigüedades. A menudo, el mero hecho de indicar el nombre de un tipo no permite a todos los lectores de una especificación determinar las propiedades exactas de ese tipo, siquiera tratándose de tipos muy familiares. En estos casos deben utilizarse los conceptos relativos a datos avanzados del LED.

D.4.7.1.2 *Introducción a los tipos abstractos*

Todos los tipos de datos en el LED están definidos como tipos abstractos. Un tipo define una serie de valores y una serie de operaciones que pueden efectuarse respecto a esos valores. Por ejemplo, el conjunto de valores para el tipo Booleano son Verdadero y Falso, y las operciones son las clásicas operaciones booleanas, a saber, No, Y, O, etc. Aunque a primera vista parezca restrictivo, el concepto de tipo abstracto goza de gran aceptación en la teoría moderna del lenguaje y permite al usuario definir *cualquier* tipo de datos necesario. En los lenguajes de programación tradicionales, las estructuras de datos son meramente un subconjunto de tipos abstractos. El comportamiento de las operaciones puede definirse de modo informal, por un texto de «comentario», o por un conjunto formal de axiomas.

Este principio permite definir los tipos con total independencia de su realización práctica, y así el empleo de tipos de datos en el LED no incluye en la elección final de las técnicas de realización.

D.4.7.1.3 *Definición de datos*

Las definiciones de datos se dividen en tres categorías distintas: definiciones de tipo de datos, generadores de tipo de datos y definiciones de sinónimo.

D.4.7.1.3.1 *Definición de sinónimo*

Las definiciones de sinónimo permiten asemejar un nombre al valor arrojado por una expresión (que depende del contexto). Una «determinación potente de tipos» exige que el tipo de la expresión sea determinable sin ambigüedades; si no, el tipo debe indicarse explícitamente (por ejemplo, la expresión '4' es ambigua pues puede pertenecer al tipo Real o al tipo Número Entero (Integer)).

Las definiciones de sinónimos se utilizarán normalmente como notación «estenográfica» cuando una expresión se emplee en varios lugares o para establecer un punto centralizado que facilite la modificación de una constante a la que se accede con frecuencia.

Ejemplo:

```
SYN Single Integer = 1
SYN BlockSize = 512 * BitsPerByte
SYN Legs = (4 * Dogs) + (2 * Birds).
```

D.4.7.1.3.2 *Definición de tipo de datos*

El mecanismo de definición de tipo de datos constituye la técnica principal de definición formal de nuevos tipos en el LED. Se prevén dos posibilidades, *newtype* y *syntype.*

Un tipo *syntype* define un tipo que representa un subconjunto de los valores de algún tipo ya existente (conocido por el nombre de tipo de origen), utilizado típicamente como la «restricción de alcance» de un lenguaje de programación para permitir un control más estrecho de los valores que puede tomar una variable.

Esto permite la «equivalencia estructural» entre tipos, o sea que los tipos son compatibles si tienen la misma estructura subyacente (tipo de origen) aun cuando tengan nombres diferentes.

Ejemplo:

```
SYNTYPE Digit = Integer
    CONSTANTS 0:9
END Digit

SYNTYPE WarmColours = Colours
    CONSTANTS Yellow, Orange, Red
END WarmColours
```

Un tipo *newtype* introduce un tipo de datos completamente nuevo que puede basarse o no en un tipo existente. Los nuevos caracteres alfabéticos (literals) y nombres de operador pueden introducirse solamente vía definiciones de *newtype.*

Las definiciones *newtype* hacen posible la equivalencia de nombre entre tipos, o sea que sólo se considera que unas variables tienen el mismo tipo si sus tipos declarados tienen textualmente el mismo nombre.

Un *newtype* puede ser definido con independencia de todos los demás tipos, como extensión de un tipo existente o como instanciación de un generador de tipo o el generador de tipo incorporado (built-in type) previsto para STRUCT. En los dos primeros casos, un rasgo importante es la posibilidad de definir propiedades del tipo como axiomas formales. Estos axiomas optativos definen la semántica de las operaciones definidas para el tipo mediante la indicación del efecto de combinar operaciones. Así, todas las operaciones están definidas en término de las demás. El tipo Booleano se usa como base para la definición de todos los tipos abstractos, constituyendo el núcleo del conjunto total de axiomas para todos los tipos (aunque el Booleano no necesita aparecer en los axiomas para un tipo particular, un examen recursivo de los tipos demostrará en último término que todos los tipos están definidos en función del Booleano).

Al construir estos axiomas el usuario del LED debe tener cuidado en evitar dos escollos importantes: insuficiencia e incoherencia. Ambos se traducen en una especificación LED potencialmente ambigua. En el primer caso puede que el conjunto de axiomas sea insuficiente para caracterizar por completo el tipo que se trata de definir. En el segundo caso, los axiomas discrepan entre sí, lo que evidentemente invalida la especificación. Desafortunadamente, la detección de estos problemas queda en manos del usuario del LED: el LED no ofrece ninguna técnica automática para detectar insuficiencias o incoherencias en las definiciones algebraicas de tipos; el usuario debe basarse en una evaluación intuitiva de un determinado conjunto de axiomas.

D.4.7.1.3.3 *Generador de tipo de datos*

Desde el punto de vista informal, un generador de tipo de datos es una "plantilla" a partir de la cual puede crearse cualquier cantidad de nuevos tipos. Los parámetros del generador son reemplazados textualmente en la totalidad de una copia textual del generador original y este nuevo generador puede interpretarse entonces como parte de la definición de tipo en que se encuentra la instanciación del generador.

Es evidente que esto evita una duplicación redundante de definiciones de tipo en los casos en que hacen falta varios tipos que son idénticos con excepción de una parte del tipo que no afecta a los axiomas básicos.

D.4.7.2 *Ejemplos de definiciones de datos*

Se presentan a continuación algunos ejemplos de definiciones de datos. Los mismos no son exhaustivos y sólo tienen por objeto dar una idea general de la capacidad del mecanismo de definición de datos adoptado en el LED.

Además de las definiciones de tipo, se presentan ejemplos de declaraciones de objetos de datos de esos tipos y de las operaciones efectuadas respecto a ellos.

Según la versión del LED que se utilice, las declaraciones y operaciones estarán contenidas (versión GR) o no (versión PR) en símbolos gráficos, pero éstas tendrán siempre la misma sintaxis.

D.4.7.2.1 *Ejemplo 1: Definición de un medidor de abonado*

Se presenta aquí una definición posible del medidor de abonado. El medidor de abonado se define basándose en las propiedades generales del tipo natural y limitando el conjunto de operaciones admisibles a la adición únicamente.

DEFINICIÓN

```
NEWTYPE SUB_METER
INHERITS NATURAL ("+")
END SUB_METER
```

EJEMPLO DE UTILIZACIÓN

```
DOC A_METER SUB_METER;
.
.
.
A_METER : = A_METER + 10
```

D.4.7.2.2 *Ejemplo 2: Definición de un tipo genérico «matriz bidimensional» («bidimensional array»)*

Se define aquí una matriz bidimensional. La definición es general y no impone ninguna restricción en cuanto a las gamas de los dos índices ni en cuanto a los componentes de la matriz. Esto permite instanciar el tipo utilizando cualquier gama y cualquier tipo de componente.

Se permiten tres operaciones distintas (aparte de la implícita, que es la operación de asignación): declaración, inserción y extracción.

Es posible así declarar nuevos objetos de ese tipo, insertar un nuevo valor en una posición seleccionada de la matriz y extraer un valor de una posición seleccionada de la matriz, respectivamente.

De acuerdo con la sintaxis de la definición de datos, para cada operador permitido los dominios y codominios del operador se indican en la parte *OPERATORS* de la definición[1]. La parte *axioms* de la semántica de los operadores.

Cada nombre de operador que va seguido de un signo de exclamación (en este caso todos ellos) indica que este nombre se usa en la definición y que se usa un nombre diferente en la sintaxis concreta cuando se emplean objetos de este tipo.

---

[1] Cuando no se indica ningún tipo de resultado (después de la flecha), el resultado es un reemplazo en los parámetros que se escogen (primed).

DEFINICIÓN

```
GENERADOR BIDI_ARRAY (TYPE index1, TYPE index2, TYPE A_TYPE);
    OPERATORS
        INSERT! : BIDI_ARRAY', index1, index2, A_type → ;
        EXTRACT! : BIDI_ARRAY, index1, index2 → A_type;
    AXIOMS
        EXTRACT!(DECLARE!(v), I1,I2) = Error!;
        EXTRACT!(INSERT!(A,Ip1, Ip2, E1),Ipr,Ipc)=
                    if Ipr = Ip1 and Ipc = Ip2 Then E1 else
                    Extract!(A,Ipr,Ipc)fi;
END BIDI_ARRAY
```

EJEMPLO DE UTILIZACIÓN

```
SYNTYPE INDEX = INTEGER
    CONSTANTS 1;20
END INDEX
NEWTYPE
    ARRAY2 BIDI_ARRAY (INDEX; INDEX; SUB_METER);
END ARRAY2
.
.
DCL DOUB_COL ARRAY2;
DCL A SUB_METER;
.
.
.
.
A :=DOUB_COL(5,16)
.
.
.
DOUB_COL(5,16):=A
.
.
```

D.4.7.2.3 *Ejemplo 3: Definición de un tipo bit*

El newtype bit se define como un tipo que tiene solamente los valores 0 y 1. Se introduce el operador flip para inicializar/reinicializar el valor de un bit (el codominio está omitido) y todas las demás operaciones (Not, And, Or, « + ») devuelven un valor.

DEFINICIÓN

```
NEWTYPE bit
    Literals (0,1);
    OPERATORS
        NOT : bit → bit;
        AND : bit,bit → bit;
        OR : bit,bit → bit;
        "+": bit,bit → bit;
        FLIP : bit' → ;
AXIOMS
    Flip (0)'=1;
    Flip (1)'=0;
    Not (1)=0;
    Not (0)=1;
    And (A,B)=if A=0 then 0 else B fi;
    Or (A,B)=if A=1 then 1 else B fi;
    "+" (A,B)=if A=0 then B else if B=1 then 0 else 1 fi
END bit
```

EJEMPLO DE UTILIZACIÓN

```
DCL CNTRL_BIT BIT;
DCL TMP_BIT, DUM_BIT BIT
.
.
.
.
TMP_BIT :=NOT(CNTRL_BIT)
DUM_BIT :=OR(TMP_BIT, CNTRL_BIT)
.
.
.
.
FLIP(CNTRL_BIT)
.
.
.
```

D.4.7.2.4 *Ejemplo 4: Definición de un tipo byte*

El tipo byte ha sido definido como una matriz de bits (newtype bitarray); como matriz, hereda todas las propiedades de las matrices. Otras operaciones permitidas son: Shr, Shl, And-Mask que permiten un desplazamiento hacia la izquierda de n posiciones ($0 \leqslant n \leqslant 7$), un desplazamiento hacia la derecha de n posiciones ($0 \leqslant n \leqslant 7$) y la operación AND con otro byte, respectivamente.

DEFINICIÓN

```
NEWTYPE bitarray (integer, bit) END bitarray;

NEWTYPE byte INHERITS bitarray ALL
    ADDING OPERATORS
                    shr : byte', integer → ;
                    shl : byte', integer → ;
                    And-Mask : byte', byte → ;
    AXIOMS
                    extract!(shr(B,1)',J)=  if J<0 or J>7 then Error!
                                            else if J=7 then 0 else extract!
                                            (B,J+1)' fi
                                            fi;
                    extract!(Shl(B,1)',J)=  if J<0 or J>7 then Error!
                                            else if J=0 then 0
                                            else extract (B,J-1).
                                            fi
                                            fi;
                    shr(B,n)'=  if n<0 or n>7 then Error!
                                else if n=0 then B
                                else shr (shr(B,1)',n-1)'
                                fi
                                fi;
                    shl(B,n)'=  if n<0 or n>7 then Error!
                                else if n=0 then B
                                else shl (shl(B,1)',n-1)'
                                fi
                                fi;

FOR ALL i IN INTEGER (extract!(And-Mask(B,MB)', i))
                            = IF i<0 OR i>7 THEN error!
                              ELSE
                              and (extract!(B,i),extract!(MB,i)) FI;
                              /* When two bytes are added together, each bit in the resulting byte is equal
                              to adding together the corresponding bits in the original bytes */

end byte
```

EJEMPLO DE UTILIZACIÓN

```
DCL F_BYTE, S_BYTE, T_BYTE BYTE;
DCL TMP_BIT BIT;
.
.
.
shr (F_BYTE, Z)
.
.
shl(S_BYTE, 3)
.
.
.
And–Mask (F_BYTE, S_BYTE)
.
.
.
TMP_ := F_BYTE (7)
.
.
```

D.4.7.2.5 *Ejemplo 5: Definición simplificada de un tipo byte*

Las definiciones de datos LED permiten a los usuarios definir tipos de datos «informalmente». Esta facilidad puede resultar muy útil, sobre todo cuando hace falta un método de refinamiento por pasos para definir el sistema. Este ejemplo muestra una manera de definir «informalmente» el mismo tipo byte definido en el ejemplo 4. (*Nota* – Se supone la definición de tipo bitarray.)

DEFINICIÓN

```
NEWTYPE byte INHERITS bitarray ALL
        ADDING OPERATORS
          shr : byte', integer → ;
          shl : byte', integer → ;
          And–Mask : byte', byte → ;

    /* Shr efectúa un desplazamiento de n posiciones, con 0≤n≥7, hacia la derecha */

    /* Shl efectúa un desplazamiento de n posiciones, con 0≤n≥7, hacia la izquierda */

    /* And–Mask efectúa la operación and en los bits correspondientes de dos bytes diferentes; el resultado se
       almacena en el primer operando */

end byte
```

D.4.7.2.6 *Ejemplo 6: Definición de un abonado*

Se presenta seguidamente un ejemplo muy simple de la definición de un abonado. Hasta aquí, un abonado es una estructura (un registro en muchos lenguajes de programación o, más formalmente, una colección de varios campos, cada uno de los cuales puede tener un tipo diferente y puede seleccionarse cuando hace falta) que contiene un campo de número (el número telefónico del abonado) y un campo de estado (el estado del abonado). Se definen previamente estos dos campos.

Las dos únicas operaciones permitidas respecto a un objeto abonado son las de conexión y desconexión; su efecto consiste en que ponen el estado del abonado en ocupado o libre, respectivamente.

DEFINICIÓN

```
NEWTYPE digitstring string(digit) END digitstring;
NEWTYPE status_id LITERALS(free, busy) END status_id;

NEWTYPE subscriber
    STRUCT   number digitstring;
             status status_id;

    OPERATORS
             connect : subscriber', subscriber" → ;
             disconnect : subscriber', subscriber" → ;
```

```
AXIOMS
        S=connect(S1,S2)' → S1(status)=busy;
        S=connect(S1,S2)" → S2(status)=busy;
        S=disconnect(S1,S2)' → S1(status)=free;
        S=disconnect(S1,S2)" → S2(status)=free;

end subscriber
```

EJEMPLO DE UTILIZACIÓN

```
DCL SUB_EX, CALD_SUB SUBSCRIBER
DCL NUM DIGITSTRING
.
.
.
SUB_EX (STATUS) := FREE
.
.
.
NUM := SUB_EX (NUMBER)
.
.
.
CONNECT (SUB_EX, CALD_SUB)
.
.
.
```

D.4.7.2.7-8 *Ejemplos 7 y 8: Definición más detallada de un abonado*

Se presentan otros dos ejemplos de tipos de datos informalmente definidos que guardan relación con la posibilidad de estructuración. Cada nuevo tipo se define como un tipo que contiene el antiguo como campo. Esta propiedad permite al usuario concentrarse en aspectos particulares del tipo de datos, difiriendo a un nivel más detallado los otros aspectos.

DEFINICIÓN

```
NEWTYPE subscriber_rev1
    STRUCT   first_app subscriber;
             counter sub_meter;

END subscriber_rev1
```

Esta definición añade la facilidad de medición a un abonado, llamada aquí suscriber_REV1.

```
NEWTYPE subscriber_REV2
    STRUCT   old_sub subscriber_rev1;
             enabling Three_cond;

END subscriber_rev2
```

Esto añade condiciones de habilitación (enabling) a un abonado (field enabling). El tipo Three_cond puede definirse así:

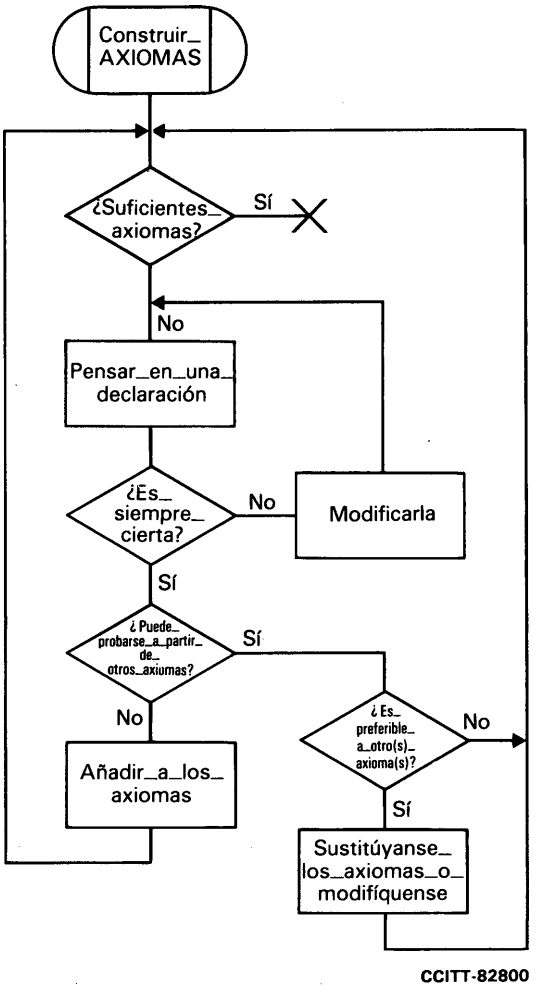
```
NEWTYPE Three_cond LITERALS (LOCAL_AB, LONG_AB, LONG_AB, INT_AB) END Three_cond;
```

D.4.7.2.9 *Ejemplo 9: Procedimiento para construir axiomas para un NEWTYPE*

Cuando se introducen nuevos tipos de datos es esencial establecer suficientes axiomas, lo que significa que:

- cada operador aparece como mínimo una vez en el conjunto de axiomas;
- son previsibles las declaraciones ciertas (distintas a los axiomas);
- no pueden detectarse incoherencias,

a fin de cumplir los requisitos del nuevo tipo de datos.



D.5 *Documentación*

D.5.1 *¿Qué es un documento?*

La ISO define un documento como un «volumen limitado y coherente de información almacenado en un soporte de una forma recuperable». Por tanto, ha de considerarse como una unidad lógica que está estrictamente delimitada. Se utilizan documentos para facilitar toda la información relativa a un sistema que se especifica o describe por medio del LED.

Cuando se utiliza papel como soporte físico para el almacenamiento de un documento, el término documento se aplica a menudo incorrectamente a las hojas de papel más bien que a su contenido lógico. Con la utilización creciente de medios de almacenamiento magnéticos, el término va volviendo a su significado original.

D.5.2 *Introducción*

El presente § 5 trata de los aspectos de la documentación. Se refiere a la organización lógica de los documentos más bien que a su organización física, la cual queda a criterio de los usuarios. La semejanza que presentan los requisitos de las organizaciones lógica y física de los documentos permite ofrecer al lector, en lo que sigue, algunas indicaciones útiles para facilitarle el establecimiento de una organización física de los documentos.

D.5.3 *¿Por qué hacen falta documentos?*

Subdividiendo la información relativa a un sistema en varios documentos, el sistema puede resultar más legible y manejable. Como el LED no posee un concepto formal de documento, el usuario está obligado a emplear un mecanismo informal o a tomar uno prestado de otro lenguaje (de preferencia el CHILL). Únicamente cuando se especifica o describe un sistema completo en forma de una sola cadena de texto utilizando la versión PR no es necesario subdividir el sistema en documentos separados.

D.5.4 *Estructura de documento*

El conjunto de documentos que abarca la definición completa del sistema puede considerarse como un conjunto de estructuras.

Se asocia a cada bloque de la estructura del sistema, incluido el nivel de sistema, una estructura de documento en la cual los documentos se refieren a subdocumentos. Véase la siguiente figura D-62.

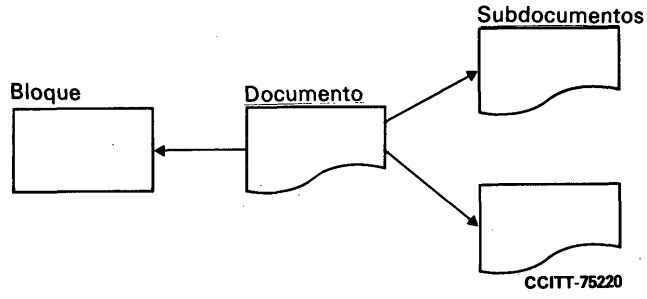


FIGURA D-62

Una estructura de documento

D.5.5 *Mecanismo de referenciación*

Todos los documentos deben estar relacionados exactamente entre sí, sobre todo porque las actualizaciones introducidas en uno de ellos pueden tener repercusiones en los demás.

Cada documento debe tener una identificación única y su frontera debe estar claramente especificada. La frontera puede indicarse especificando los números de página de las hojas pertenecientes a un mismo documento. En el caso de un documento de una sola página, como es un diagrama de interacción de bloque, el símbolo de cuadro podría indicar la frontera.

Un mismo documento puede aparecer como un subdocumento para más de un bloque de un mismo sistema.

En el § D.9, «Ejemplos», se indican dos métodos de referenciación. Uno se basa en las sintaxis PR y el otro utiliza el mecanismo de comentario de los diagramas GR. Para más detalles, véase el § D.9.

D.5.6 *Tipos de documentos*

Los diferentes tipos de diagramas que se presentan en las sintaxis GR son en general documentos, según la definición establecida. También deben incluir una indicación de su frontera. Los documentos, que juntos abarcan todas las definiciones presentadas en una definición de bloque, deben, como ya se ha dicho, formar una estructura de documento. En esta estructura al menos las definiciones de proceso deben colocarse en documentos diferentes. Si se emplea un diagrama de interacción de bloque, las definiciones de canal figuran también en el mismo documento. Es preferible que las definiciones de señal y las definiciones de datos aparezcan juntas, en documentos separados.

Análogamente, las definiciones de procedimiento y las definiciones de macro deben formar subdocumentos del documento del proceso.

Las definiciones de las señales que forman una lista de señales pueden colocarse en documentos separados denominados lista de señales X, Y, Z, etc. La figura D-63 siguiente muestra una posible estructura de documentos para un bloque:

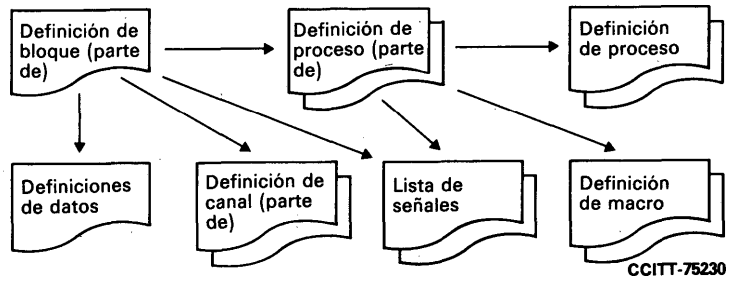


FIGURA D-63

Posible estructura de documentos para un bloque

D.5.7 *Combinación de GR y PR*

La definición del proceso puede establecerse en parte utilizando la versión GR. El gráfico del proceso queda cubierto ya sea por el diagrama del proceso (en GR) o por el cuerpo del proceso (en PR). Para poder mezclar la definición del gráfico del proceso establecida en GR con el resto establecido en PR, la definición del gráfico del proceso debe constituir un documento en sí misma.

También la definición de un sistema o bloque puede definirse parcialmente utilizando la versión GR. En tal caso el diagrama de interacción debe formar un documento. Como quiera que el diagrama de interacción del bloque abarca solamente ciertas partes de la definición de un sistema o bloque, las partes restantes deben aparecer en documentos separados.

Al mezclar las versiones GR y PR, la representación en PR tiene que dividirse en varios documentos. Desafortunadamente, la sintaxis PR no prevé esta posibilidad, razón por la cual la subdivisión tiene que hacerse de una manera informal.

En el § D.7 se examinan los métodos para formar partes sintácticamente conectadas de una representación PR.

D.6 *Directrices para la representación de sistemas por medio del LED/GR*

En el presente § 6 de las Directrices para el usuario del LED se explica el empleo de la representación de sintaxis gráfica del LED, denominada LED/GR.

D.6.1 *¿Por qué una sintaxis gráfica?*

Un lenguaje gráfico tiene la ventaja de que muestra claramente la estructura de un sistema y permite visualizar fácilmente el flujo del control. Al ser un instrumento para el diseño, el LED debe tener una forma que permita al usuario expresar sus ideas en forma clara y concisa. La sintaxis gráfica permite hacerlo y se amolda más a la representación tradicional de las máquinas de estado finito ampliadas, mientras que la representación de frases textuales se presta mejor para la utilización por máquinas.

La sintaxis de representación gráfica del LED (LED/GR) es la forma original del LED. Fue elaborada durante el Periodo de Estudios 1973-1976 y se publicó por primera vez en la versión de 1976 de las Recomendaciones de la Serie Z.100.

La sintaxis de representación gráfica se basa en los lenguajes gráficos desarrollados por diferentes organizaciones para su propio uso.

D.6.2 *Diagramas LED/GR*

En esta sección se explican los diagramas siguientes:

Diagrama de interacción de bloque que describe la estructura de un sistema o bloque.

Diagrama en árbol de un bloque que describe la partición de un sistema en bloques y sub-bloques.

Diagrama en árbol de un proceso que define la partición de un proceso en subprocesos.

Diagrama de subestructura de canal que define la estructura interna de un canal.

Diagrama de definición de macro que define un conjunto de símbolos. Cada aparición de una referencia a la definición del macro debe ser reemplazada por la definición antes de ser interpretada.

Diagrama de procedimiento que define el comportamiento en la ejecución de un procedimiento.

Diagrama de proceso que define el comportamiento de todas las *instancias de proceso* en la *definición de ese proceso.*

Las definiciones de señal y las definiciones de datos se efectúan por medio de la sintaxis LED/PR (véase el § D.7.3.2) y están referenciadas a partir de los diagramas en que se utiliza la información.

D.6.2.1 *Directrices generales*

Las directrices generales para la elaboración de diagramas gráficos son las siguientes:

- Los diagramas deben establecerse de manera que la secuencia normal de lectura sea de arriba hacia abajo y de izquierda a derecha.
- No deben contener demasiados detalles; a menudo conviene subdividir los diagramas grandes en subdiagramas que traten partes o aspectos diferentes.
- De preferencia, un segmento conectado de un diagrama debiera caber en una página.
- Pueden insertarse comentarios en cualquier parte de un diagrama, ya sea utilizando el símbolo de comentario GR o la sintaxis de comentario PR.
- El texto debe aparecer de preferencia en el símbolo al cual está asociado, pero si ello no es posible puede colocarse en un símbolo de extensión de texto conectado al símbolo.

D.6.2.2 *Referenciación*

Las Recomendaciones actuales relativas al LED no prevén ningún método para la inserción de referencias entre diagramas GR, entre definiciones en LED/PR o entre diagramas GR y definiciones PR. Tanto para conseguir una representación LED completa de un sistema como para estructurar su documentación, hace falta un mecanismo de referenciación.

El mecanismo de referenciación puede articularse de diversas maneras. Ello depende de la organización preferida de la documentación y de las metodologías. Como las Recomendaciones no prevén ningún mecanismo, los usuarios pueden elegir el que consideren más conveniente.

Un ejemplo del empleo de referencias en la sintaxis PR es la utilización de notas o de la sintaxis de comentario como se muestra en la figura D-64.

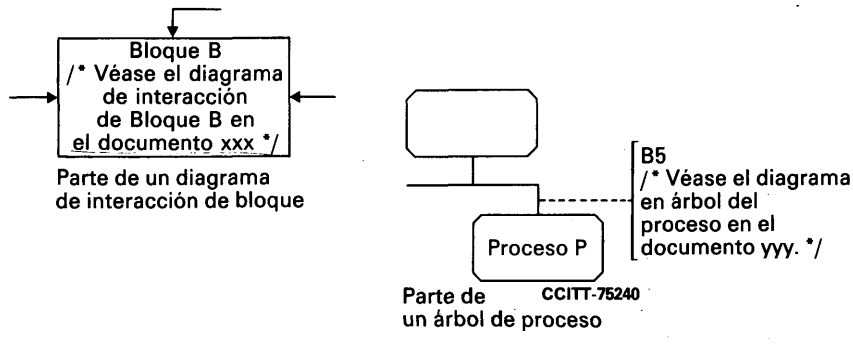


FIGURA D-64

Ejemplo de la utilización de notas o comentarios como referencias

Hay varios otros métodos posibles. En el § D.9 aparecen otros ejemplos.

D.6.2.3 *Normógrafo*

En la parte interior de la tapa posterior de este fascículo hay un normógrafo de la representación gráfica LED. El mismo está representado esquemáticamente en la figura D-65.

En la figura, los símbolos de Entrada, Salida, Decisión, Alternativa, Proceso, Comienzo, Tarea, Estado, Conservación, Conector y Parada se muestran directamente en 3 tamaños, esto es, 20×40 mm, $20/\sqrt{2} \times 40/\sqrt{2}$ mm y 10×20 mm. También se indican las relaciones internas para el tamaño más grande.

Los símbolos de Llamada de procedimiento, Macro y Creación pueden construirse a base del símbolo de Tarea trazando las rayas horizontales o verticales adicionales indicadas.

Los símbolos de Definición de procedimiento e Ingreso de macro pueden construirse a base del símbolo de Comienzo trazando las rayas verticales adicionales indicadas.

El símbolo de Egreso de macro puede construirse a partir del símbolo de Conector trazando la raya vertical adicional indicada.

El símbolo de Retorno es una combinación de los símbolos de Conector y Fin de proceso.

El símbolo de Comentario se traza mediante uno u otro extremo del símbolo de Tarea.

El símbolo de Condición habilitadora puede trazarse utilizando el símbolo de Parada.

El símbolo de Todo (*) no figura en la plantilla pues debe imprimirse con el texto asociado a los símbolos.

El símbolo de Canal y la raya al símbolo de Expansión de texto son una línea de trazo lleno.

La raya de señal y la raya a un símbolo de Comentario son una línea de trazos y blancos que guardan la relación 1 : 1.

En las figuras D-209 y D-210 pueden verse todos los símbolos recomendados.

D.6.3 *Utilización del LED/GR*

La descripción de todos los símbolos utilizados en la versión LED/GR aparece en el resumen de la sintaxis gráfica (anexo C a las Recomendaciones Z.100 a Z.104).

D.6.3.1 *Expresión de la estructura en el LED/GR*

En este punto se explica la partición de un sistema en bloques y canales y la partición adicional facultativa de los bloques y canales.

Cabe señalar que, cuando un sistema se subdivide en varios niveles con un grado creciente de detalle, puede contener diagramas alternativos (véase la Recomendación Z.102). Cuando existen dos descripciones alternativas, la menos detallada ha de considerarse como una imagen general.

D.6.3.1.1 *Diagrama de interacción de bloque*

Un *diagrama de interacción de bloque* muestra la estructura interna de un *sistema* o un *bloque* en términos de *canales* y *bloques.* Puede representarse el conjunto de *procesos* contenido en los bloques.

También pueden representarse los *procesos* que pueden crear otros *procesos.*

El diagrama consta de:

- el símbolo de cuadro (obligatorio para los bloques pero facultativo si se utiliza alrededor de un sistema);
- el símbolo de bloque;
- el símbolo de canal;
- listas de señales;

y facultativamente:

- el símbolo de proceso;
- el símbolo de ruta de señal;
- el símbolo de petición;
- el símbolo de entorno.

Aparece seguidamente un ejemplo sencillo de un diagrama de interacción de bloque (véase la figura D-66) que representa un pequeño sistema (S). Consta de dos bloques (B1 y B2). Obsérvese que en el caso de un sistema, el símbolo de cuadro enmarcador es facultativo.

De acuerdo con la convención relativa a la lectura de arriba hacia abajo y de izquierda a derecha, es preferible que los nombres aparezcan en la esquina superior izquierda del aspecto nombrado o al lado de él. Pueden colocarse definiciones referenciadas fuera del diagrama, como ocurre con la lista de señales L_1 en este ejemplo.

Pueden colocarse en el diagrama definiciones de señal y definiciones de datos, expresadas en la sintaxis PR, de preferencia en la esquina superior izquierda o fuera del diagrama. Si se colocan definiciones fuera del diagrama y no es evidente dónde pueden consultarse, el diagrama debe contener una referencia que indique dónde pueden encontrarse.

En el símbolo de bloque pueden mostrarse los procesos contenidos, indicados en la figura D-67.

Para construir diagramas más legibles, éstos pueden subdividirse a fin de mostrar separadamente el contenido de los bloques. Por ejemplo, el segmento mostrado en la figura D-67 puede formar un subdiagrama, representado separadamente, del diagrama de interacción del bloque.

Con frecuencia es útil mostrar varios niveles de la estructura en un mismo diagrama de interacción de bloque. Para ello se reemplaza un símbolo de bloque en un diagrama por el diagrama de interacción de bloque de ese bloque. Veamos un ejemplo en la figura D-68.

Esto puede repetirse cualquier número de veces.

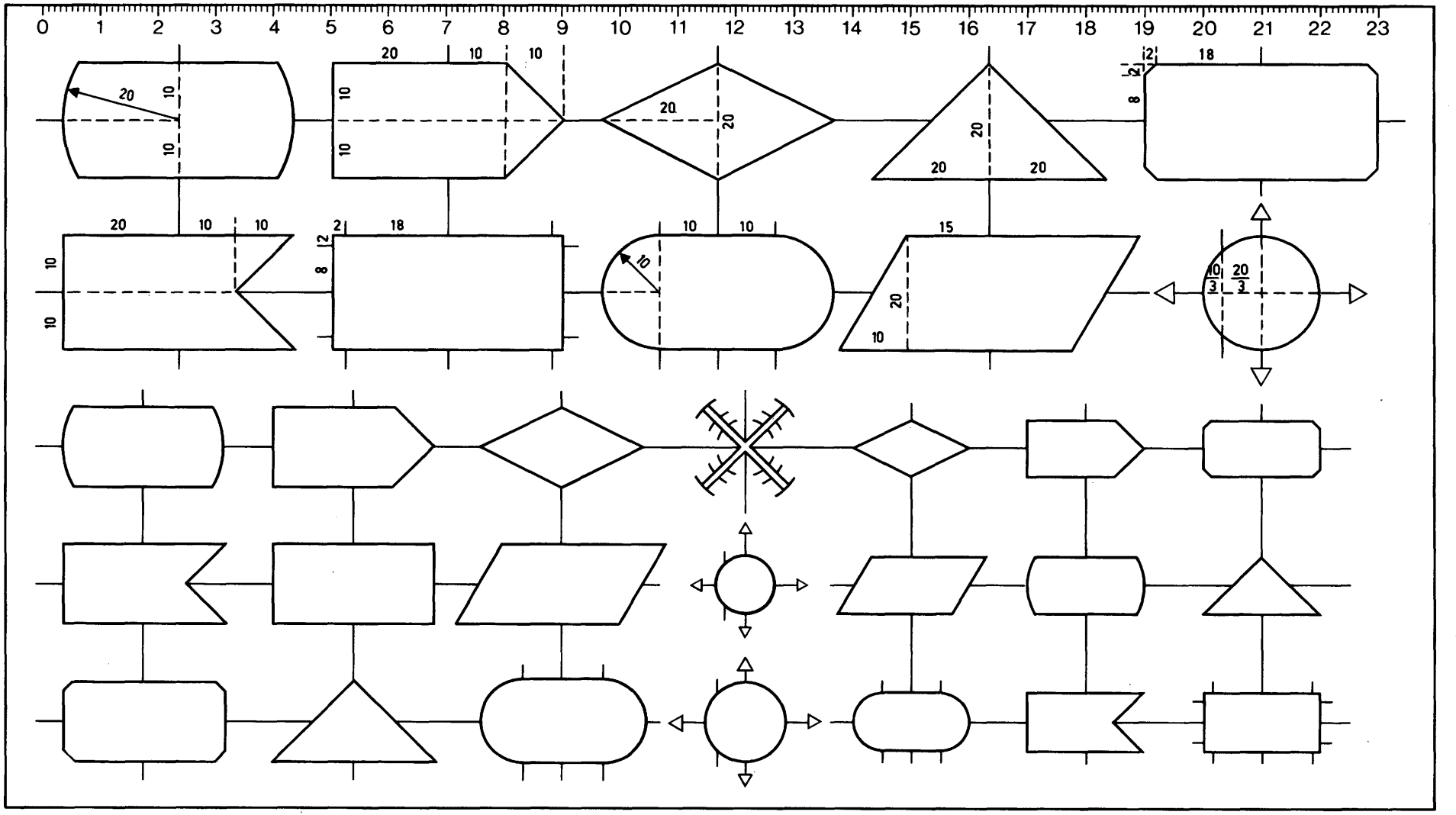
Para conseguir un diagrama legible y bien estructurado, debe tenerse en cuenta lo siguiente:

- Los diagramas no deben ser tan complejos que no quepan en una página.
- La posibilidad de subdividir un diagrama en subdiagramas reduce la cantidad de detalles y hace más legible el diagrama.
- El macro puede ser utilizado como medio de reducir la complejidad de los diagramas grandes.

D.6.3.1.2 *Diagrama en árbol de bloque*

Un diagrama en árbol de bloque representa la estructura de un sistema en términos de bloques y sub-bloques. La finalidad del diagrama es ofrecer al lector una imagen general de la estructura total del sistema.

El diagrama es un árbol jerárquico de símbolos de bloque con líneas de «partición».



Nota 1 – Relación altura : longitud = 1 ; 2

Nota 2 – Se indican 3 tamaños : longitud de 40 mm, 28 mm y 20 mm. 40 mm/$\sqrt{2}$ = 28 mm/$\sqrt{2}$ = 20 mm, etc.
Esto permite la fotorreducción del formato A3 al formato A4 con tamaños compatibles de los símbolos.

FIGURA D-65

Diseño esquemático del normógrafo del LED

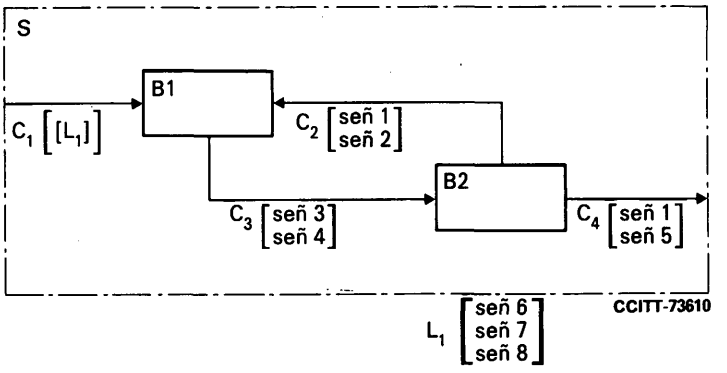


FIGURA D-66

Ejemplo de un diagrama de interacción de bloque que representa un sistema

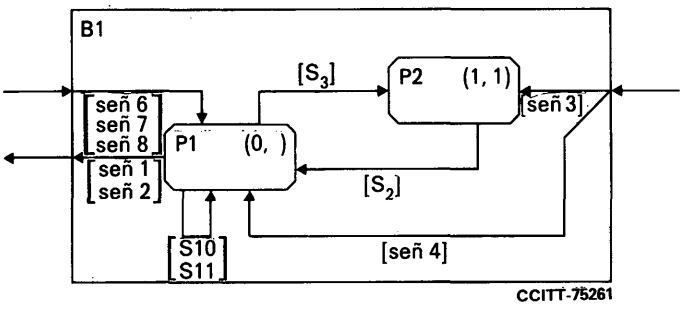


FIGURA D-67

Segmento de un diagrama de interacción de bloque

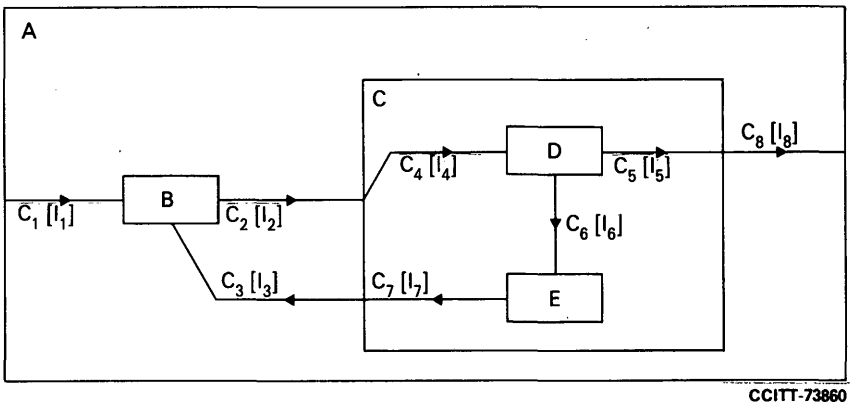


FIGURA D-68

Ejemplo de la representación en varios niveles

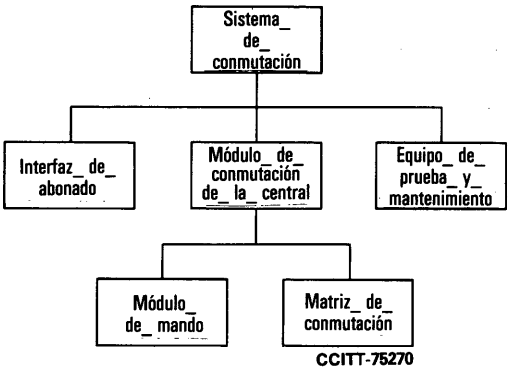


FIGURA D-69

Ejemplo de un diagrama en árbol de bloque

El diagrama debe trazarse utilizando de preferencia símbolos de bloque del mismo tamaño. De esta manera, los bloques situados en un mismo nivel de partición aparecen como un nivel uniforme. Si el diagrama es tan grande que hace falta más de una página, debe subdividirse en diagramas en árbol de bloques «parciales» como se indica en la figura D-70.

A menudo resulta conveniente subdividir un diagrama en árbol de bloque en diagramas parciales.

Para ello, el primer diagrama, que tiene el sistema como raíz, se corta de manera que aparezca un conjunto de bloques separado como si no estuviese subdividido. Los bloques, en el lugar donde se cortó el diagrama original, aparecen como raíces en los diagramas que muestran la subdivisión ulterior.

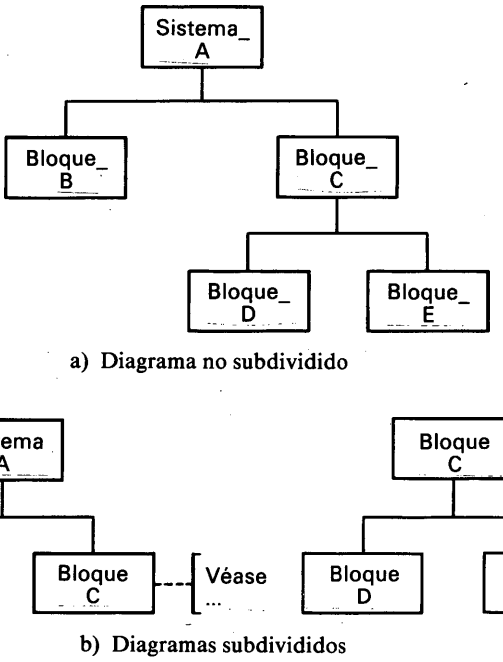


FIGURA D-70

Ejemplo de diagramas en árbol parciales de bloque

Si se utilizan diagramas parciales y no está claro que un bloque está subdividido aún más y/o dónde aparecen los diagramas de continuación, deben insertarse referencias por medio del símbolo de comentario.

D.6.3.1.3 *Diagrama en árbol de proceso*

Un diagrama en árbol de un proceso describe la partición de un proceso en subprocesos y muestra dónde van los procesos. El diagrama es un árbol jerárquico de símbolos de proceso con líneas de «partición». La asignación de los procesos se describe mediante el símbolo de comentario.

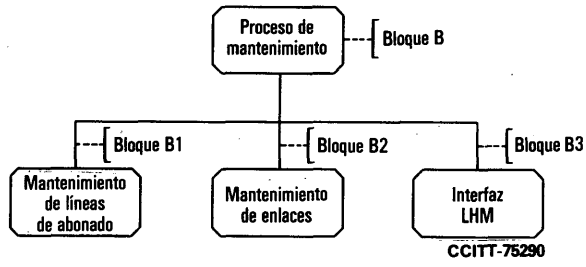
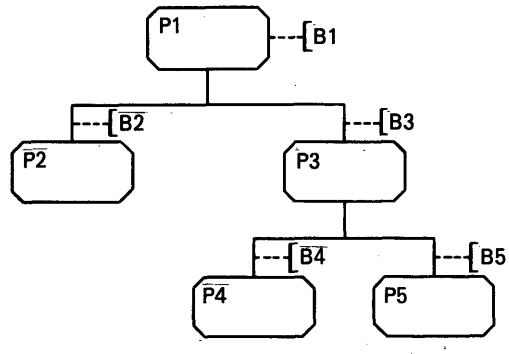


FIGURA D-71

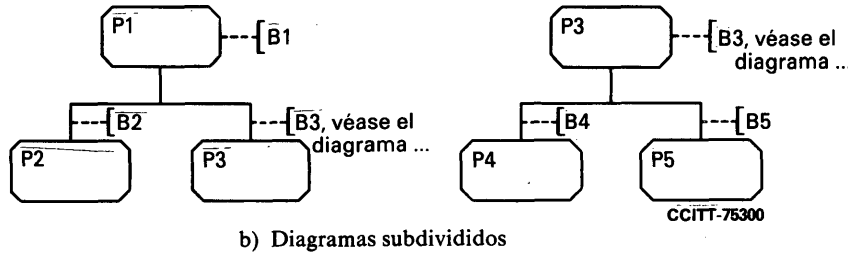
Ejemplo de un diagrama en árbol de proceso

Para que esté completo, la raíz debe ser un proceso que no es un subproceso de ningún otro proceso. Asimismo, ninguno de los procesos hoja debe estar subdividido aún más. De preferencia deben utilizarse símbolos de proceso del mismo tamaño a fin de que todos los subprocesos de un mismo nivel de partición aparezcan como un nivel coherente en el diagrama.

Si el diagrama se vuelve tan grande que no cabe en una página, o por razones de legibilidad, puede dividirse en diagramas parciales. Se obtiene un conjunto de diagramas parciales haciendo que un conjunto de procesos subdivididos aparezcan como no subdivididos en el diagrama original. Estos procesos aparecen como raíces en otros diagramas que describen la partición ulterior.



a) Diagrama no subdividido



b) Diagramas subdivididos

FIGURA D-72

Ejemplo de diagramas en árbol parciales del proceso

Si se utilizan diagramas parciales y no está claro que un proceso está subdividido aún más o dónde aparece el diagrama de continuación, deben insertarse referencias por medio de símbolos de comentario.

D.6.3.1.4 *Diagrama de subestructura de canal*

Un diagrama de subestructura de canal describe cómo se subdivide un canal en subcomponentes. El diagrama es parecido al de interacción de bloque. Todas las directrices expuestas en el § D.6.3.1.1 son aplicables también al diagrama de subestructura de canal.

En el diagrama de interacción de bloque en el que aparece el canal subdividido debe haber una referencia al diagrama de subestructura de canal que describe la partición. Véanse las figuras D-73 y D-74.

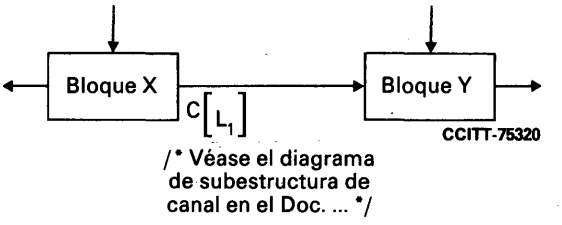


FIGURA D-73

Segmento de un diagrama de interacción de bloque que contiene un canal subdividido

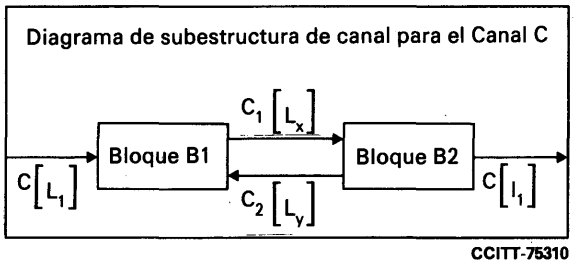


FIGURA D-74

Ejemplo de un diagrama de subestructura de canal

Si es evidente que el canal está subdividido y también está claro dónde aparece el diagrama de subestructura, puede omitirse la referencia.

D.6.3.2 *Definiciones de señal*

No existe una sintaxis gráfica especial para las definiciones de señal. En los diagramas LED/GR que emplean definiciones de señales se utiliza la sintaxis LED/PR (véase el § D.7.3.2). El texto PR puede insertarse en el diagrama, pero de preferencia sólo debiera referenciarse en los diagramas.

Los diagramas en los que pueden referenciarse definiciones de señal son:

- los diagramas de interacción de bloque;
- los diagramas de subestructura de canal.

Cuando las definiciones aparecen en el diagrama, debieran figurar de preferencia en la esquina superior izquierda del diagrama como se muestra en la figura D-75.

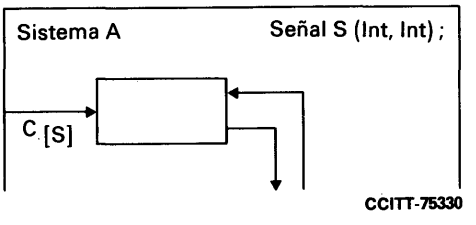


FIGURA D-75

Segmento de un diagrama de interacción de bloque que contiene una definición de señal

La presencia de largos textos PR en un diagrama disminuye la legibilidad de éste.

Es preferible reemplazar el texto PR por una referencia a su ubicación; por ejemplo;

/* Para el tipo de señal y las definiciones de procedimiento véase el documento ... */

D.6.3.3 *Definiciones de datos*

No hay ninguna sintaxis gráfica especial para la definición de datos en el LED/GR. Cuando hacen falta definiciones de datos en GR, se utiliza la sintaxis PR (véase el § D.7.3.4).

Hay dos tipos de definiciones de datos:

- definiciones de tipo de datos;
- definiciones de variable.

En el caso de las definiciones de señal, es preferible referenciar el texto PR a partir de los diagramas GR.

D.6.3.3.1 *Definiciones de tipo de datos*

Las definiciones de tipo de datos pueden ser referenciadas en los siguientes diagramas LED/GR:

- diagramas de interacción de bloque;
- diagramas de subestructura de canal;
- diagramas de definición de macro;
- diagramas de proceso;
- diagramas de procedimiento.

Respecto a las definiciones de tipo de datos se aplican las mismas directrices que para la inserción de definiciones de señal.

D.6.3.3.2 *Definiciones de variable*

Pueden aparecer definiciones de variable en los siguientes diagramas LED/GR:

- diagramas de proceso;
- diagramas de procedimiento.

Respecto a las definiciones de variable se aplican las mismas directrices que para la inserción de definiciones de señal.

D.6.3.4 *Diagramas de definición de macro*

Un diagrama de definición de macro define el conjunto de símbolos por el cual debe reemplazarse cada referencia a ese macro antes de su interpretación.

El diagrama puede tener al menos uno o más símbolos de ingreso, seguido por una línea de flujo que entre en el diagrama, y uno o más símbolos de egreso que siga a una línea de flujo que sale del diagrama. Aparte de ello, un diagrama de definición de macro puede contener cualquier conjunto de símbolos gráficos y texto en cualquier combinación. La exactitud y el significado del macro sólo pueden determinarse después de efectuado el reemplazo.

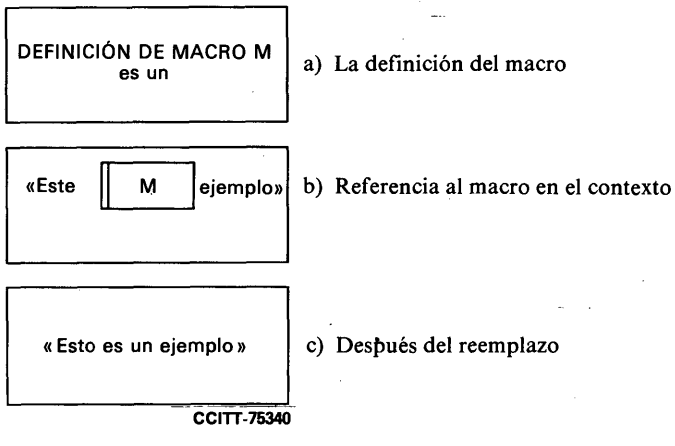


FIGURA D-76

Ejemplo del uso del macro

Las directrices generales para obtener diagramas bien estructurados se aplican también al diagrama de definición de macro.

D.6.3.5 *Diagramas de procedimiento*

El diagrama de procedimiento define el comportamiento que debe tener lugar para cada llamada de ese procedimiento. Los procedimientos en el LED son similares a los procedimientos en el CHILL y otros lenguajes de programación. Los procedimientos están destinados a:

a) permitir la estructuración de un *gráfico de proceso* en varios niveles de detalle;

b) conseguir una especificación compacta, al permitir la representación por medio de un solo elemento de un conjunto complejo de elementos que puede considerarse aisladamente;

c) permitir la definición previa y el empleo repetido de los conjuntos de elementos de uso común.

Como puede haber definiciones de procedimiento contenidas en definiciones de sistema, definiciones de bloque, definiciones de proceso y definiciones de procedimiento, un mismo procedimiento puede ser llamado desde varios procesos. Se permite el uso de bibliotecas de procedimientos comunes.

El diagrama que define el procedimiento es similar a un diagrama de proceso. Las diferencias estriban en que el símbolo de estado del diagrama de proceso se reemplaza por el símbolo de estado del procedimiento y en que el símbolo de parada se reemplaza por el símbolo de retorno.

Las directrices relativas al diagrama de proceso se aplican también al diagrama de procedimiento; véase el § D.6.3.6. En la figura D-77 se muestra un ejemplo de un diagrama de procedimiento.

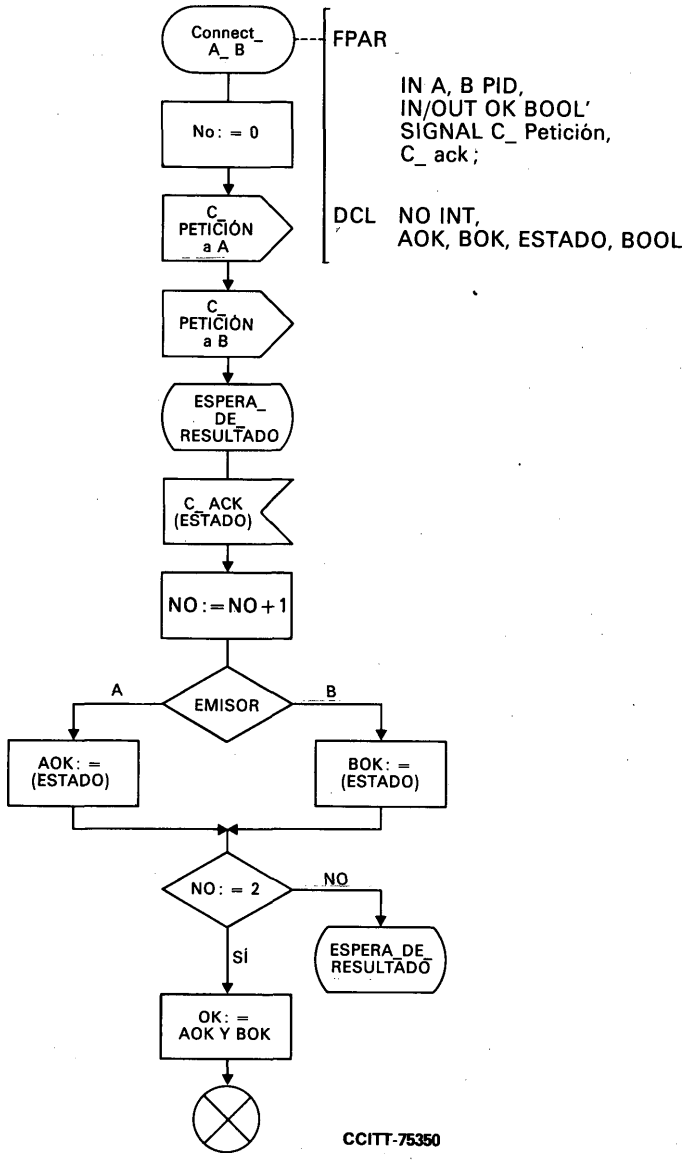


FIGURA D-77

Ejemplo de diagrama de procedimiento

D.6.3.6 *Diagrama de proceso*

Un diagrama de proceso define el comportamiento de un proceso por medio del LED/GR. Cuando los procesos descritos son largos o complejos, a menudo resulta útil disponer de documentos auxiliares generales de carácter preliminar. Al final de esta sección se presentan algunos ejemplos de documentos de esa naturaleza.

D.6.3.6.1 *Versiones de los diagramas de proceso LED/GR*

Si, en un diagrama LED/GR, las transiciones se describen exclusivamente por símbolos explícitos de acciones, se dice que se ha seguido la versión basada en transiciones del LED/GR.

Si, por el contrario, los estados se describen utilizando pictogramas de estado y las acciones efectuadas durante las transiciones se deducen de las diferencias entre estos pictogramas, se dice que se sigue la versión basada en estados del LED/GR.

Si se combinan ambas versiones se dice que se sigue la versión combinada.

En las figuras D-78 a D-80 se ilustran ejemplos de estas tres versiones.

La versión basada en las transacciones es adecuada cuando la secuencia de acciones es importante y las descripciones detalladas de los estados son menos importantes.

La versión basada en los estados es adecuada cuando la secuencia de acciones dentro de cada transición no es importante, cuando conviene la explicación pictográfica y cuando conviene la representación compacta. Puede utilizarse en las aplicaciones para las cuales se han definido elementos pictográficos idóneos (véase el § D.6.3.6.22).

La versión combinada es adecuada cuando sea ventajosa la redundancia que lleva consigo.

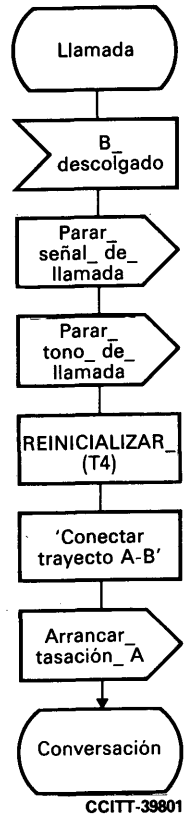
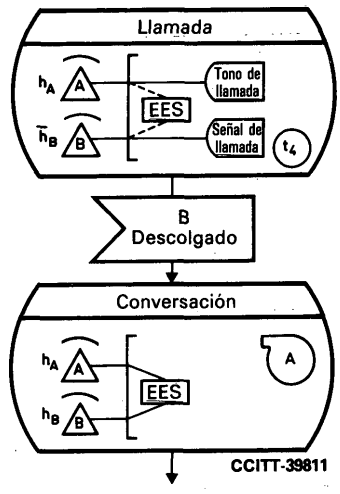


FIGURA D-78

Versión basada en las transiciones



EES : Enlace entrante/saliente

FIGURA D-79

Versión basada en los estados

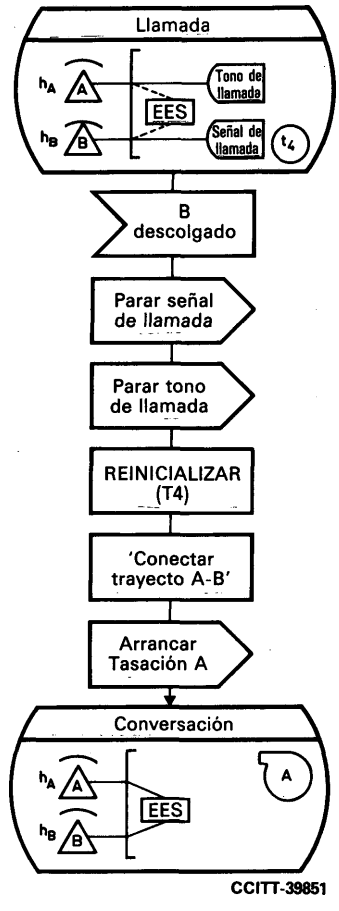


FIGURA D-80

Versión combinada

D.6.3.6.2 *Símbolos y reglas de conexión*

En el anexo C a las Recomendaciones Z.100 a Z.104 se indican los símbolos que pueden utilizarse en diagramas de proceso y la manera de conectarlos.

D.6.3.6.3 *Diagramas bien estructurados*

Consta seguidamente una serie de directrices generales para la elaboración de diagramas bien estructurados.

D.6.3.6.3.1 *Comienzo y fin de un diagrama*

Las instancias de proceso pueden crearse en forma explícita, y en tal caso debe dibujarse el diagrama del proceso de modo que empiece con un símbolo de comienzo. En el caso de procesos que carecen de una creación explícita de instancias, existe un estado de comienzo identificado por su posición en la parte superior del diagrama del proceso.

Normalmente, es preferible construir el diagrama LED de manera que el flujo tenga sentido descendente.

Un diagrama puede descomponerse en varias páginas. El flujo de una página a otra puede hacerse sólo por medio de conectores (véase el § D.6.3.6.16) o por apariciones múltiples de un mismo estado (véase el § D.6.3.6.5).

D.6.3.6.3.2 *Posibles disposiciones de un diagrama*

La suspensión de un proceso se representa por el símbolo de estado. Queda a discreción del usuario del LED el destacar o no estas suspensiones (desconectando los diagramas en cada estado). De acuerdo con las reglas del LED, pueden emplearse tres métodos:

Método A

El diagrama se interrumpe en cada símbolo de estado y continúa después a partir de ese mismo símbolo de estado.

Cuando se emplea el método A, el diagrama LED completo consiste en un conjunto de n diagramas, siendo n el número de estados del proceso. Cada diagrama comienza por un estado diferente, muestra todas las transiciones a partir de ese estado y termina por los estados de esas transiciones. El mismo estado puede aparecer en varios diagramas.

En la figura D-81 se presentan ejemplos.

Obsérvese que las transiciones, junto con las entradas, se indican solamente por medio de líneas dirigidas.

Si el proceso tiene un comienzo, el símbolo de comienzo y la transición de comienzo aparecerán también como diagrama separado (véase la figura D-81 b)). Asimismo, si el proceso tiene una parada, se mostrarán por separado los estados y las transiciones que conduzcan a símbolos de parada (véase la figura D-81 c)).

Método B

El método no se interrumpe en ninguna parte.

Según el método B, el diagrama LED completo se construye como un gráfico totalmente interconectado. El ejemplo de la figura D-81 se ha reconstruido en la figura D-82 en forma de un gráfico totalmente interconectado.

Método C

El diagrama se interrumpe en algunos símbolos de estado.

Cuando el diagrama LED completo se interrumpe en algunos símbolos de estado solamente, existen varias formas equivalentes de reconstruir el diagrama de la figura D-81, una de las cuales se representa en la figura D-83 que consta de dos subdiagramas.

En el método C, algunos estados aparecen en varios subdiagramas. Cuando se emplea este método conviene dibujar conexiones entre las partes del diagrama que tienen correspondencia lógica. Por ejemplo, es muy corriente que una parte relativamente pequeña de los procesos represente el comportamiento en las situaciones más usuales. Convendría representar esta parte en un subdiagrama separado.

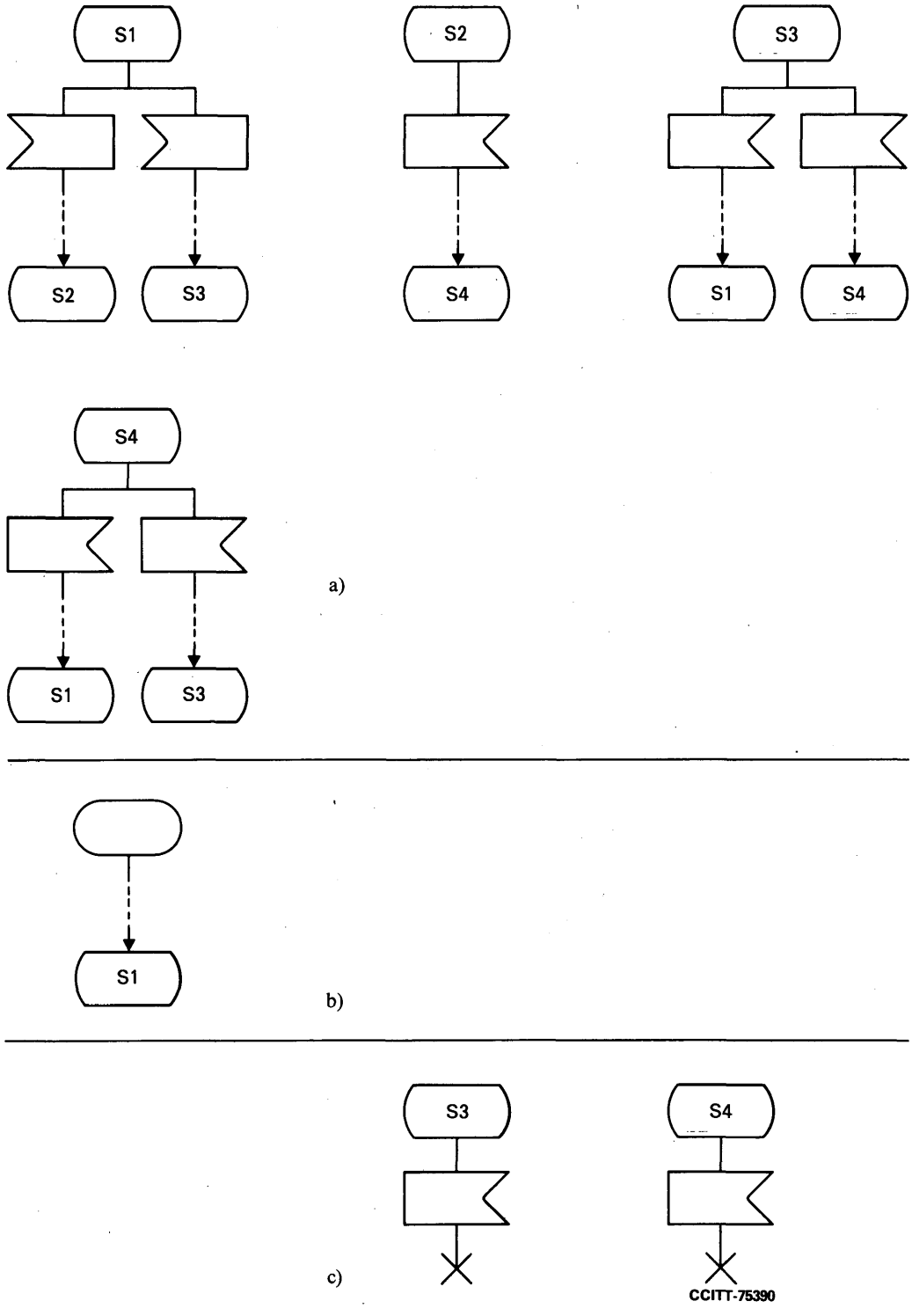


FIGURA D-81

Gráficos completamente separados

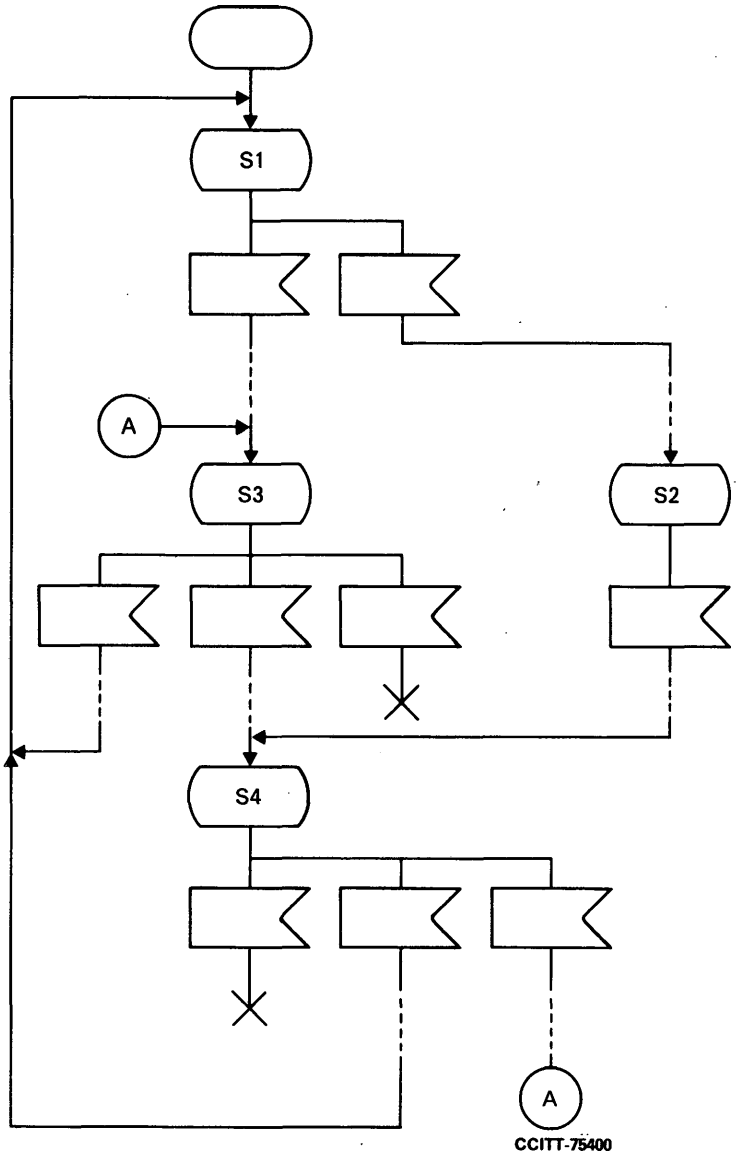


FIGURA D-82

Gráfico totalmente interconectado

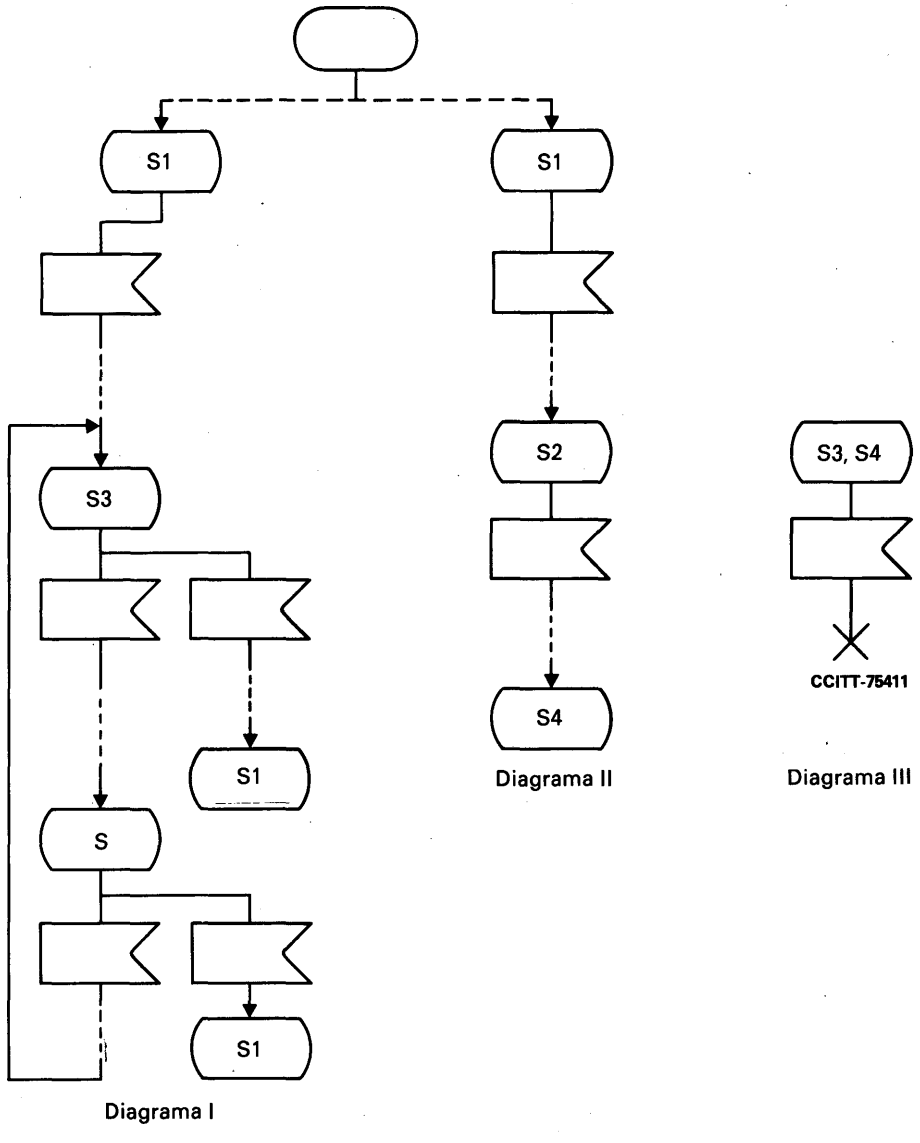


FIGURA D-83

Gráficos parcialmente conectados

Comparación de los tres métodos

El método A tiene las ventajas de reducir al mínimo el tiempo necesario para adoptar una forma determinada de dibujar el diagrama, su modificación es muy fácil, así como también lo es la obtención de información a partir del diagrama.

El método B ofrece la ventaja de dar una visión de conjunto del proceso, mientras que el método C, por su parte, ofrece la ventaja de permitir la estructuración del proceso de acuerdo con los criterios predeterminados en cuanto a las partes del proceso que ofrecen mayor interés.

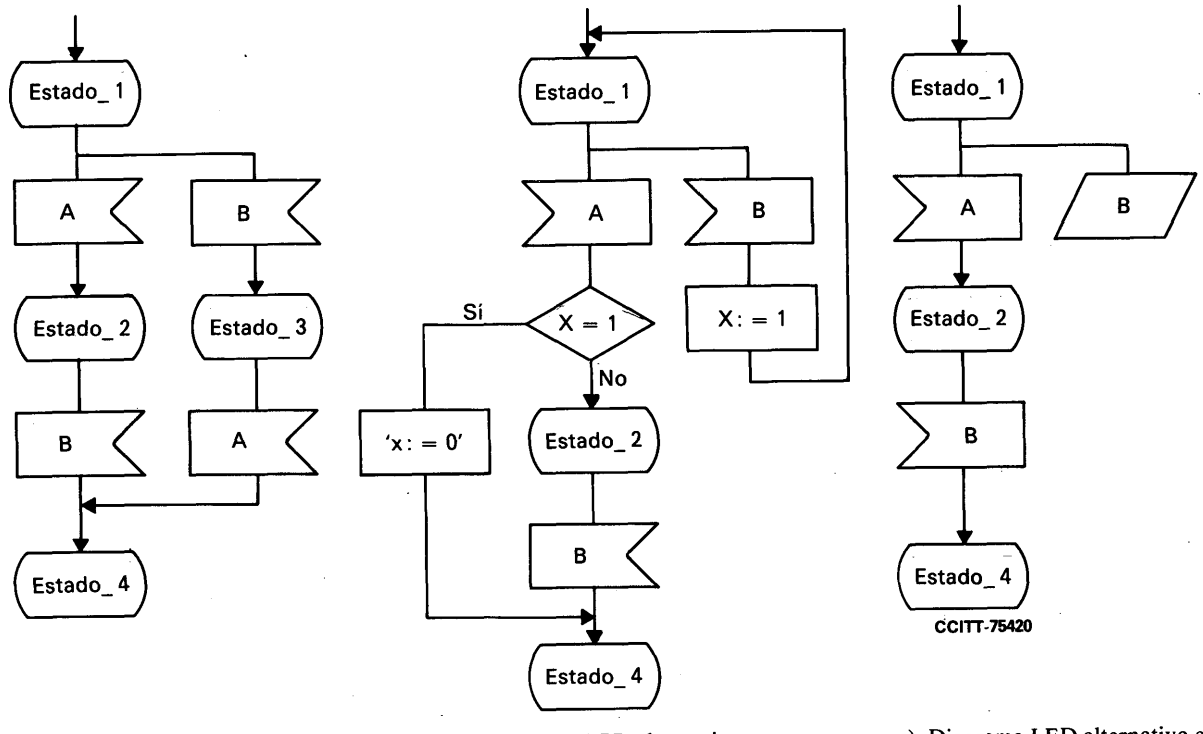
Claridad de los diagramas

Como los diagramas LED constituyen un medio de comunicación entre los autores y los lectores, su claridad es una propiedad muy apreciada. La claridad del diagrama puede, no sólo del método empleado para estructurar el diagrama, sino también de la complejidad del proceso (por ejemplo, el número de estados, entradas y decisiones), lo que depende a su vez del modo de estructuración.

Como los diagramas deben ser fáciles de leer y de comprender, debe procurarse que tengan una estructuración adecuada.

A la hora de elegir el método más claro de representación, el autor debe conocer los diferentes procedimientos para la aplicación del LED. Los ejemplos indicados en las partes a), b) y c) de la figura D-84 pueden considerarse formas alternativas de representar una parte del proceso. Como no se realizan tareas ni salidas (salvo la tarea artificial de inicializar y reinicializar la variable X), los tres ejemplos son lógicamente idénticos.

Se considera que la posibilidad c) constituye la representación más clara.



a) Ejemplo de un diagrama LED

b) Diagrama LED alternativo en que se emplea una decisión

c) Diagrama LED alternativo en que se emplea una conservación

FIGURA D-84

Los tres diagramas muestran el proceso que pasa del estado 1 al estado 4 al recibir las entradas A y B en un orden aleatorio

D.6.3.6.3.3 *LED/PR en diagramas de proceso*

Cuando se describe la definición de un proceso en LED/GR, las definiciones de tipo de datos y las declaraciones de variable se describen en la sintaxis LED/PR. Ese texto puede incluirse directamente en el diagrama o bien figurar en un documento separado. Cuando el texto no se incluye directamente en el diagrama, tiene que haber una referencia al documento en el que aparece. Véanse las figuras D-85 y D-86.

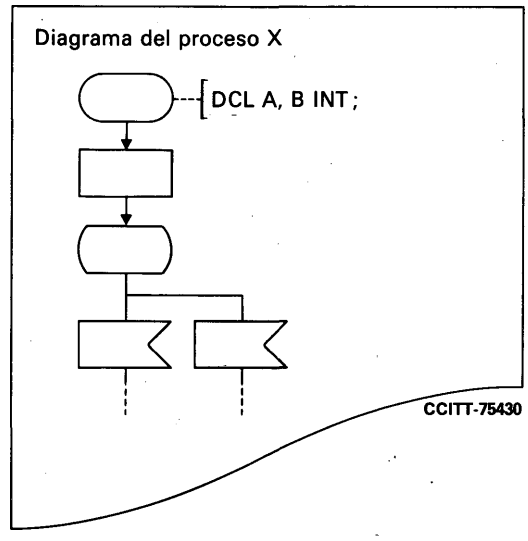


FIGURA D-85

Ejemplo de texto LED/PR incluido en el diagrama de un proceso

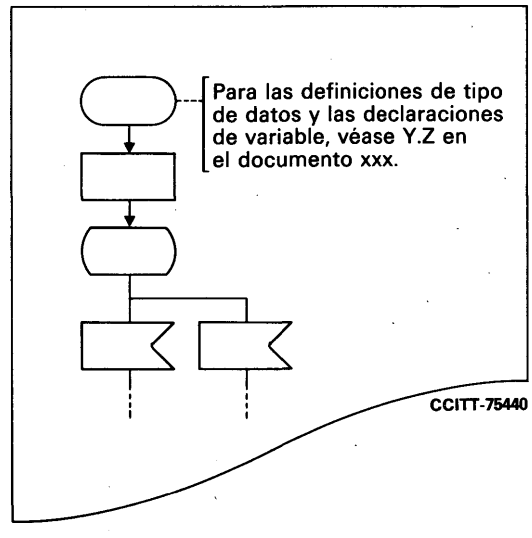


FIGURA D-86

Ejemplos de referencias a definiciones que aparecen en otra parte

El método que ha de aplicarse depende de la cantidad de texto que haya que incluir o referenciar. La inclusión de grandes cantidades de texto PR puede destruir la claridad de un diagrama, pero si el volumen del texto no es pequeño puede resultar más legible si se incluye el texto en el diagrama.

D.6.3.6.3.4 *Definiciones de proceso*

Cuando se definen procedimientos que son locales respecto a procesos, los diagramas de procedimiento pueden incluirse en los diagramas de proceso o bien referenciarse.

Cuando se incluyen diagramas de procedimiento, es importante que el gráfico que define el procedimiento esté separado del gráfico que define el proceso. Una manera de hacerlo consiste en formar capítulos o secciones dentro del diagrama de proceso, como puede verse en la figura D-87.

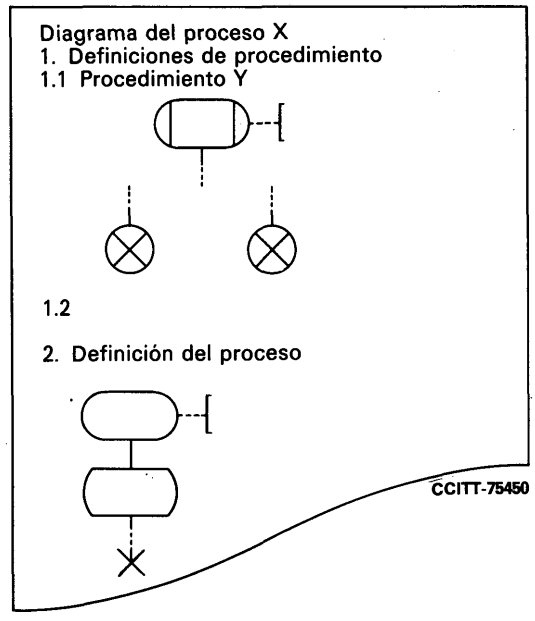


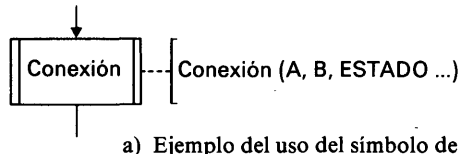
FIGURA D-87

Ejemplo de la inclusión de definiciones de procedimiento en el diagrama de un proceso

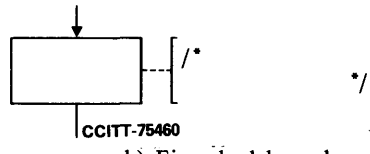
D.6.3.6.3.5 *Texto de símbolos*

Normalmente, el texto asociado a un símbolo debe colocarse dentro de este símbolo. Pero ello no es siempre práctico debido a la longitud del texto y/o al tamaño de los símbolos.

De ser necesario, el texto puede colocarse fuera del símbolo. Si se opta por esta posibilidad, debe quedar claro que el texto colocado fuera del símbolo está asociado a él y que no es un comentario. Para ello se utiliza el símbolo de extensión de texto mostrado en la figura D-88.



a) Ejemplo del uso del símbolo de extensión de texto para texto formal diferido



b) Ejemplo del uso de un mecanismo de « nota » para asociar texto formal

FIGURA D-88

D.6.3.6.4 *Creación de procesos*

D.6.3.6.4.1 *Petición de creación*

La creación de una nueva instancia de proceso se describe por el símbolo de petición de creación. Cabe señalar que sólo puede crearse instancias de proceso dentro del mismo bloque de la instancia creadora.

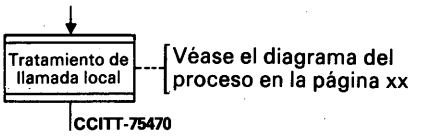


FIGURA D-89

Ejemplo del uso del símbolo de petición de creación

Para hacer más claro el diagrama debe insertarse, a modo de comentario, una referencia al diagrama del proceso creado. Cuando hacen falta parámetros reales, deben facilitarse utilizando la sintaxis LED/PR.

D.6.3.6.4.2 *Comienzo*

El símbolo de comienzo de un diagrama de proceso describe el punto de partida del comportamiento de una instancia de proceso cuando ésta es creada. Los parámetros formales del proceso deben estar asociados al símbolo.

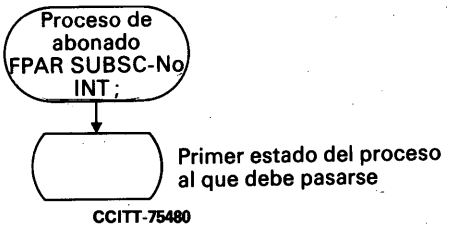
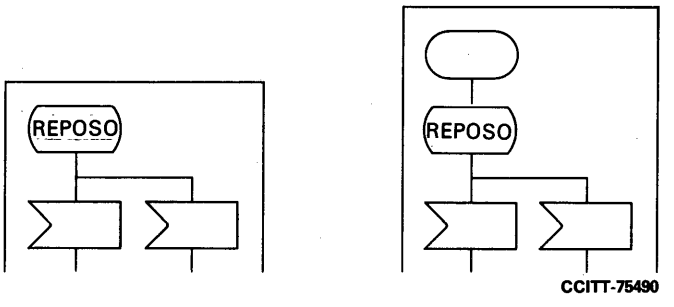


FIGURA D-90

Ejemplo del uso del símbolo de comienzo

A fin de no invalidar los diagramas LED antiguos, el símbolo de proceso requerido puede estar implícito; se supone en tal caso que aparece antes del «primer» símbolo de estado en el diagrama del proceso, entendiéndose por «primero» el primer símbolo de estado que aparece en la parte superior de la primera página del diagrama.



a) El diagrama, tal como aparece b) Interpretación del diagrama

FIGURA D-91

Símbolo de comienzo implícito

El símbolo de comienzo debe aparecer como primer símbolo de un diagrama; si queda oculto dentro del diagrama, el lector puede sufrir confusiones.

D.6.3.6.5 *Estados*

Un estado se representa por un símbolo de estado, tiene símbolo de entrada asociados y puede tener también símbolos de conservación. La figura D-92 muestra un ejemplo de un estado.

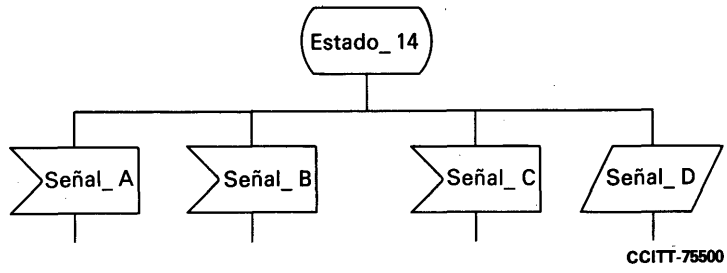


FIGURA D-92

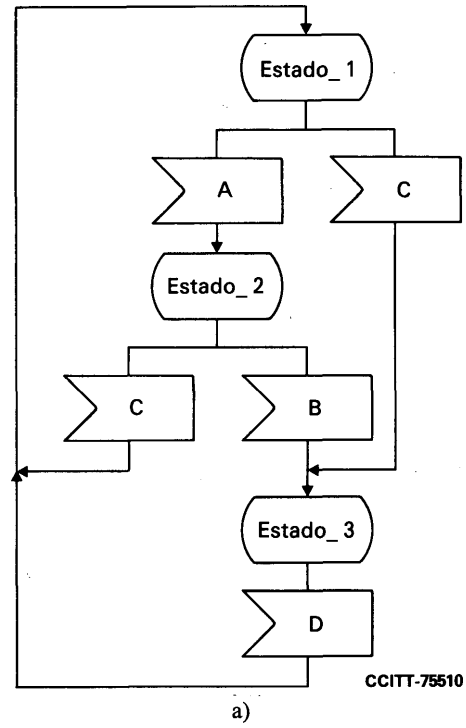
Ejemplo del uso de un estado

D.6.3.6.5.1 *Apariciones múltiples y símbolos de estado siguiente*

Por razones de conveniencia, para simplificar el dibujo o mejorar la comprensión, un mismo estado puede aparecer varias veces en un diagrama LED. Cuando tal ocurre, el diagrama se considera completamente equivalente al diagrama que se obtendría fusionando todas las apariciones múltiples de un mismo estado. Las figuras D-93 y D-94 muestran ejemplos. En la figura D-93 b) se emplea un símbolo de estado como conector con el estado principal que tiene el mismo nombre, cuando está referenciado en un símbolo de estado siguiente. En la figura D-94 se muestra un estado representado por múltiples símbolos, cada uno de los cuales tiene solamente un subconjunto de sus entradas (o conservaciones).

En la figura D-93, los diagramas a) y b) son lógicamente equivalentes. El diagrama a) contiene una sola aparición de cada estado, mientras que el diagrama b) utiliza apariciones múltiples. En el diagrama b) el estado tiene una aparición principal en la que aparecen los símbolos de todas sus entradas (y conservaciones) conexas. Cuando se puede llegar a este estado desde otros puntos del diagrama (como punto terminal de una transición), el mismo aparece como un estado sin ninguna entrada o conservación asociada. Mediante un comentario que referencie el símbolo de estado desde el símbolo de estado siguiente se puede mejorar la claridad, sobre todo cuando las apariciones están en páginas diferentes.

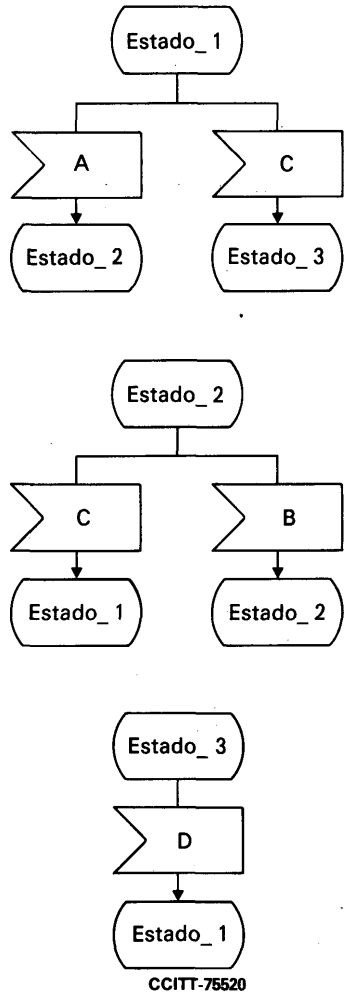
En la figura D-94 se usan apariciones múltiples de un estado para construir el conjunto total de entradas (y conservaciones). Cada aparición del estado se muestra con un subconjunto de ellas únicamente.



a)

FIGURA D-93

Un diagrama completo



b)

FIGURA D-93b

Diagrama a) con los estados principales y estado subsiguiente utilizados como conectores a los estados

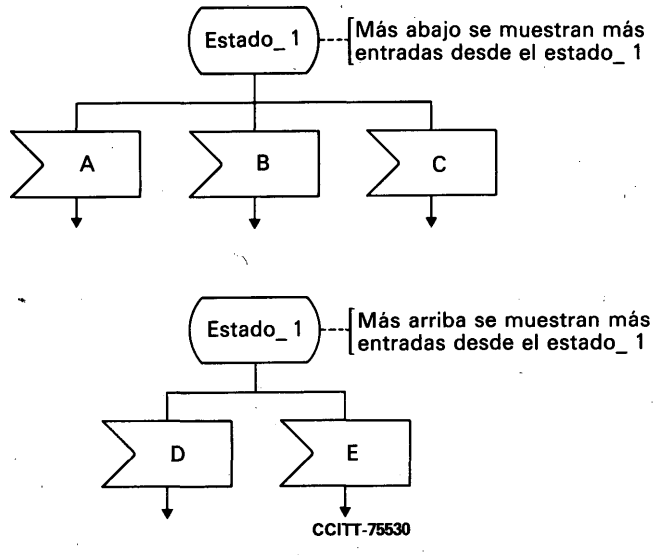


FIGURA D-94

Apariciones múltiples de un estado, cuando no pueden representarse claramente todas las entradas desde un mismo símbolo

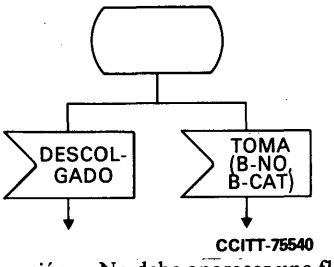
Este método resulta satisfactorio cuando hay estados que tienen una cantidad relativamente grande de entradas o conservaciones, pero puede entrañar el riesgo de que los lectores interpreten incorrectamente el diagrama si no saben que hay múltiples apariciones. Para evitarlo, los estados que muestran solamente un subconjunto de las entradas/conservaciones deben anotarse con un comentario que haga referencia a los otros estados con sus entradas y conservaciones asociadas, como se ve en la figura D-94.

Pueden utilizarse apariciones múltiples para centrar la atención del lector en ciertos aspectos (por ejemplo, la secuencia normal de señales procesadas) difiriendo otros aspectos a otras páginas (por ejemplo, el tratamiento de una situación de alarma).

D.6.3.6.6 *Entradas*

El símbolo de entrada, conectado a un símbolo de estado, representa el concepto de entrada.

Cuando la señal que ha de recibir la entrada transporta datos, la lista de variables locales para almacenar los valores se indica en el símbolo por medio de la sintaxis LED/GR. Si no se transporta ningún dato, la lista puede omitirse. Si se transportan datos y se omite la lista, debe interpretarse que los valores serán descartados al recibirse. Véase el ejemplo de la figura D-95.



Observación – No debe aparecer una flecha en la línea que lleva al símbolo de entrada

FIGURA D-95

Ejemplo del uso del símbolo de entrada

Si varias señales pueden iniciar la misma transición después de un estado, todas las entradas pueden representarse por el mismo símbolo de entrada con una lista de señales en su interior. En la lista de señales, todos los nombres de las señales (y sus listas de variables) aparecen separados por comas.

En las versiones anteriores del LED se distinguía entre entradas externas e internas. La diferencia radicaba en el hecho de que las entradas externas recibían símbolo enviados desde el exterior del bloque en que estaba contenido el proceso. Como no hay ninguna diferencia semántica entre los símbolos cuyos orígenes están dentro y fuera del bloque, se han suprimido esos conceptos. Si se encuentra en un diagrama el símbolo especial que se utilizaba anteriormente para representar una entrada interna, el mismo debe interpretarse como un símbolo de entrada. (Véase la figura D-97.)

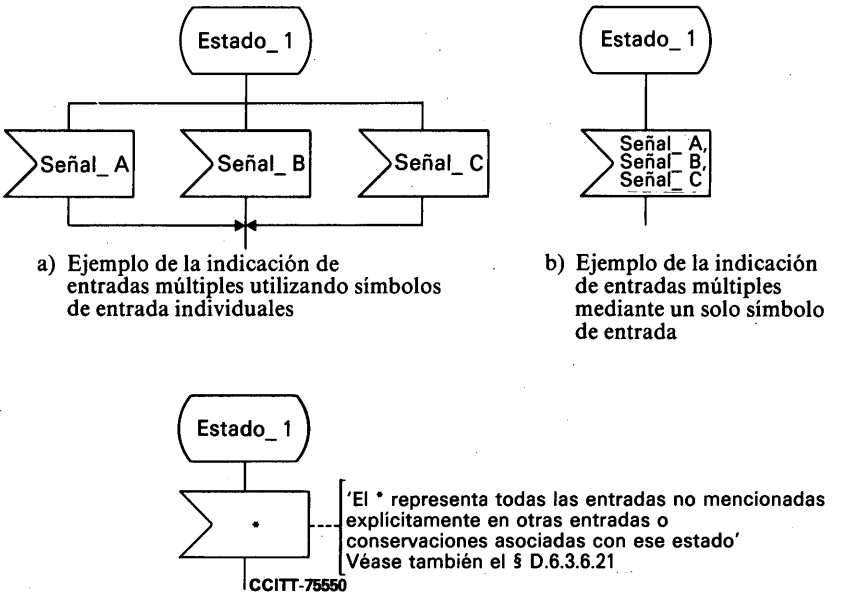
D.6.3.6.7 *Conservaciones*

El símbolo de conservación representa la lista de señales conservadas en un estado. Las señales que deben conservarse deben mencionarse por su nombre dentro del símbolo. Si se emplean varios símbolos conectados a un estado, dichos símbolos deben considerarse como un solo símbolo de conservación que contiene la lista de todos los nombres indicados en todos esos símbolos.

En la lista de nombres, estos nombres deben estar separados por comas. Existe también una representación estenográfica por medio de la «notación asterisco», conforme se explica en el § D.6.3.6.21.

D.6.3.6.8 *Salidas*

Una salida que envía una señal y le proporciona los valores que debe transmitir se representa por el símbolo de salida. La lista de valores que deben transmitirse se especifica mediante el LED/PR, y de preferencia debiera aparecer dentro del símbolo. El símbolo debe contener también la dirección de la instancia de proceso a la que se envía la señal. Véanse los ejemplos de la figura D-99.



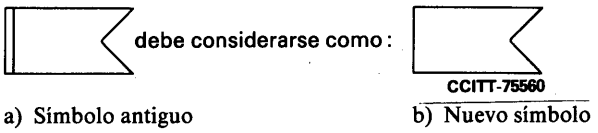
a) Ejemplo de la indicación de entradas múltiples utilizando símbolos de entrada individuales

b) Ejemplo de la indicación de entradas múltiples mediante un solo símbolo de entrada

c) Ejemplo de la indicación de entradas múltiples mediante la notación asterisco

FIGURA D-96

Diversas representaciones posibles de entradas múltiples

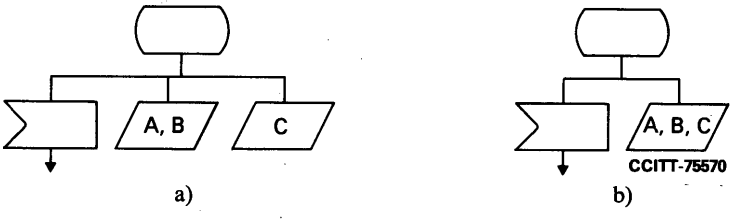


a) Símbolo antiguo

b) Nuevo símbolo

FIGURA D-97

Equivalencia de entrada externa e interna

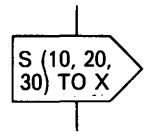


a)

b)

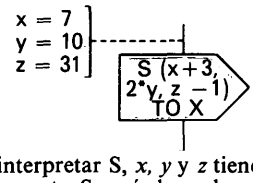
FIGURA D-98

Diferentes representaciones de una misma lista de conversación



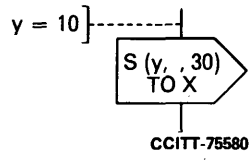
Observación – La señal S tiene asociados tres valores, 10, 20 y 30.

a)



Observación – Al interpretar S, *x*, *y* y *z* tienen (en este ejemplo) los valores 7, 10 y 31 respectivamente. S envía los valores 10, 20 y 30.

b)



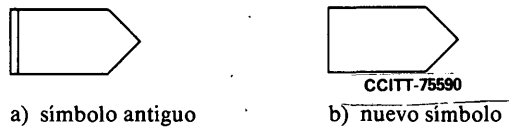
Observación – Al interpretar S, *y* tiene (en este ejemplo) el valor 10. S envía los valores 10, un valor indefinido y 30.

c)

FIGURA D-99

Salida con datos asociados

En versiones anteriores del LED se distinguía entre salidas externas e internas, de la misma manera que para las entradas. Cualquier símbolo de salida interna antiguo que se encuentre en un diagrama debe ser interpretado como un símbolo de salida. Véase la figura D-100.



a) símbolo antiguo b) nuevo símbolo

FIGURA D-100

Equivalencia de salida externa e interna

D.6.3.6.9 *Condiciones habilitadoras*

La condición habilitadora se representa por el símbolo de condición habilitadora, que es una combinación del símbolo de entrada y una sintaxis gráfica que representa la condición booleana:

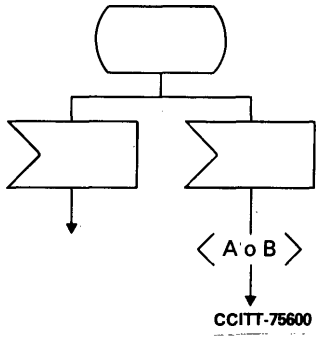


FIGURA D-101

Esta combinación debe considerarse como un solo símbolo, razón por la cual no debe haber una punta de flecha en la línea que conecta sus dos partes. Véase la figura D-101.

D.6.3.6.10 *Señales continuas*

La señal continua se representa por el símbolo de la señal continua.

La expresión contenida en él debe ser booleana.

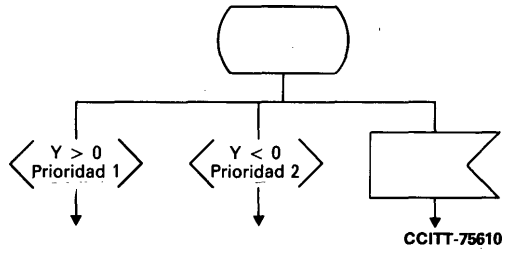


FIGURA D-102

Ejemplo de condiciones habilitadoras

La cláusula de prioridad es obligatoria si hay más de una condición habilitadora asociada a un estado.

D.6.3.6.11 *Tareas*

Una tarea se representa por el símbolo de tarea. La descripción de la tarea, ya sea en texto LED/PR o en texto informal, debiera aparecer de preferencia dentro del símbolo de tarea.

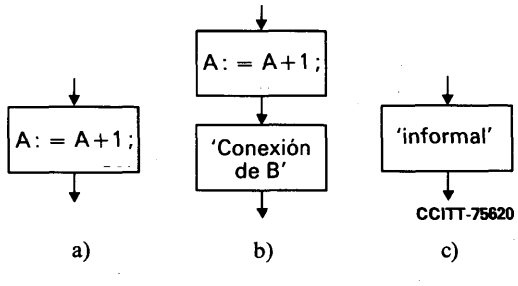


FIGURA D-103

Ejemplos de símbolos de tarea

En los diagramas que comprenden tanto sentencias formales como texto informal, el texto informal debe estar encerrado entre apóstrofos.

D.6.3.6.12 *Decisiones*

Una decisión se representa por un símbolo de decisión.

El símbolo de decisión encierra una pregunta relativa a una determinada decisión. Las preguntas no necesitan tener signos de interrogación sino sólo la expresión que se ha de evaluar, por ejemplo, número, A + B, Z. El símbolo tendrá dos o más ramas. La respuesta a la pregunta aparecerá junto a la rama correspondiente, a su derecha o encima de ella. Las ramas contendrán juntas todas las respuestas posibles. Cada respuesta puede mostrarse con el valor de la respuesta (por ejemplo: 17, 12) o con un operador como prefijo (por ejemplo: > 18, ≠ 34, = 25. En la figura D-104 se muestran algunos posibles formatos.

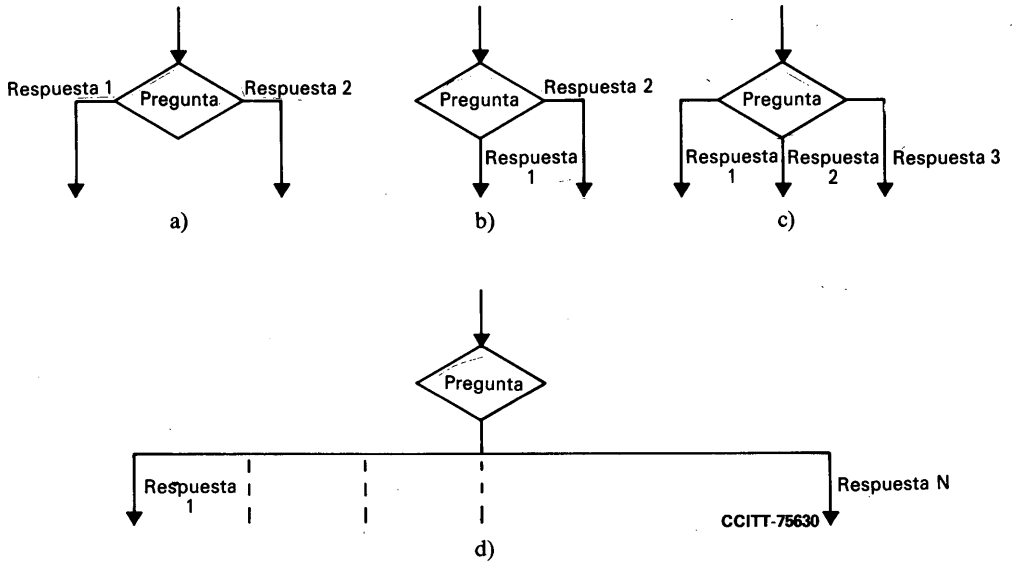


FIGURA D-104

Algunos posibles formatos del símbolo de decisión

Muchas preguntas se refieren a una condición específica para la que sólo hay dos respuestas posibles: «Sí» o «No», o sea CIERTO o FALSO.

Por ejemplo:

Abonado B ocupado

Aparato telefónico «colgado»

Cifra 2 recibida

Z = 1

Se da un ejemplo en la figura D-105.

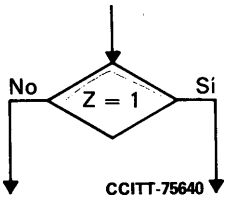


FIGURA D-105

Formato de una decisión binaria simple

En el caso de dos o más respuestas pueden emplearse los formatos de la figura D-106, suponiendo que Z es un número entero positivo. La respuesta ELSE significa cualquier respuesta no cubierta por las otras respuestas, esto es, los valores distintos de los del conjunto (1, 2, 3, 9, 10, 11, 12, 13, 14, 15).

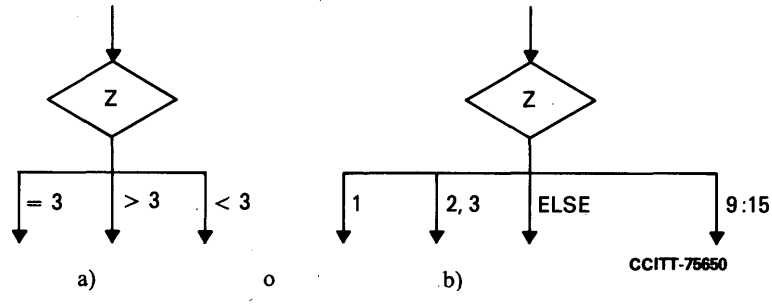


FIGURA D-106

Formatos de preguntas y respuestas

D.6.3.6.13 *Macro*

Pueden colocarse símbolos de macro en cualquier parte de un diagrama. La exactitud del diagrama sólo puede determinarse una vez que se reemplaza el símbolo de macro por el contenido de su definición. Pero se prefiere utilizar macros en los diagramas de proceso en que se puede aparecer un símbolo de tarea.

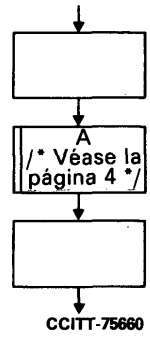


FIGURA D-107

Uso del símbolo de macro

Cuando se emplean macros, resulta útil insertar en el símbolo de macro una referencia que indique dónde se encuentra la definición (en la figura D-107 se muestra un ejemplo).

Cuando utiliza macros, el usuario debe ser consciente de que los macros pueden ocultar efectos secundarios. Como un macro aparece de ordinario en lugar de un símbolo de tarea, el lector puede suponer la semántica de una tarea.

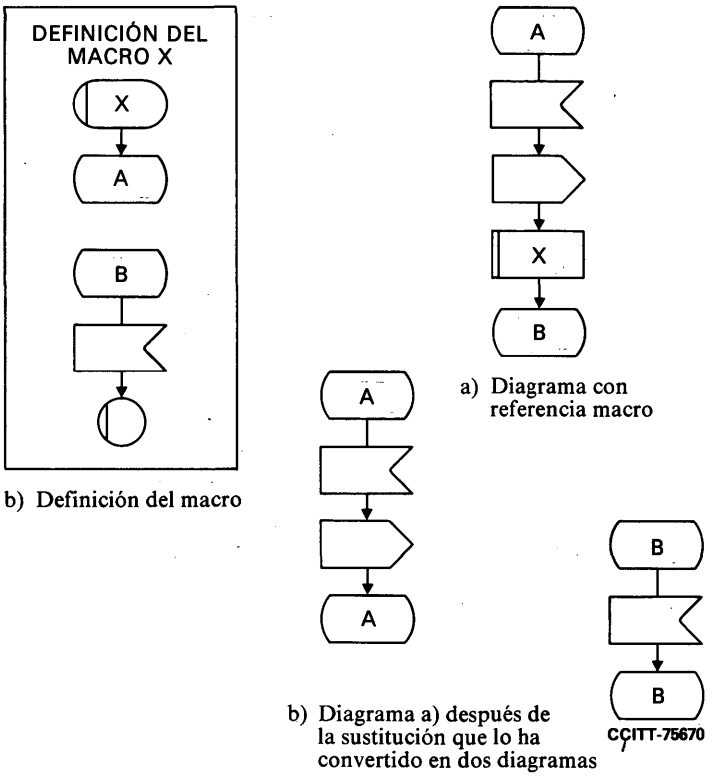


FIGURA D-108

Ejemplo de la manera no recomendada de utilizar macros

En la figura D-108 anterior se muestra cómo el reemplazo de una llamada de macro puede alterar la secuencia de un diagrama de procedimiento. Cuando los macros contienen estados, debe tenerse en cuenta que la transición, que contiene la referencia, puede subdividirse en varias transiciones.

Cuando los macros contienen varios ingresos y/o egresos, éstos deben identificarse claramente mediante etiquetas, colocadas de preferencia a la derecha de las líneas de flujo entrantes o salientes.

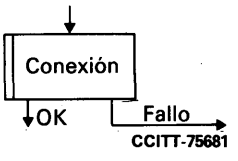


FIGURA D-109

Macro con varios egresos

D.6.3.6.14 *Procedimientos*

El símbolo de llamada de procedimiento representa la llamada de un procedimiento. Los parámetros facilitados con la llamada debieran colocarse de preferencia dentro del símbolo, pero si ello no es posible o práctico pueden colocarse en un símbolo de extensión de texto a un lado.

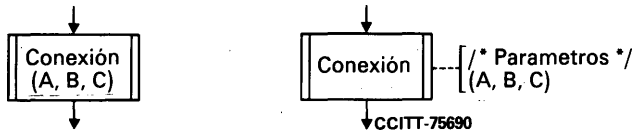


FIGURA D-110

Ejemplo del uso del símbolo de llamada de procedimiento

Este símbolo puede aparecer dondequiera pueda colocarse un símbolo de tarea.

El procedimiento constituye un instrumento que permite reducir la complejidad y utilizar soluciones normalizadas.

D.6.3.6.15 *Opción*

El símbolo de opción se usa para describir varios comportamientos alternativos posibles de un proceso mediante un solo diagrama. Esto resulta útil si hay varios comportamientos de proceso que sólo difieren en pequeñas partes. El símbolo contiene una expresión de opción. Esta expresión debe ser de naturaleza tal que pueda evaluarse antes de que se interprete el proceso, es decir que no debe ser una condición dinámica. La expresión debe ser también de naturaleza tal que su evaluación conduzca a un conjunto discreto de alternativas con las cuales pueden etiquetarse las líneas de flujo salientes.

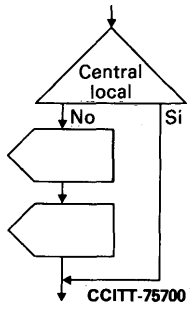


FIGURA D-111

Ejemplo del uso de opciones

Es preferible colocar las alternativas de la opción a la derecha de las líneas de flujo salientes.

El diagrama no puede interpretarse hasta que se evalúen todas las expresiones de la opción, y el diagrama debe considerarse entonces como si se hubiesen suprimido todas las ramas inalcanzables que siguen a la opción.

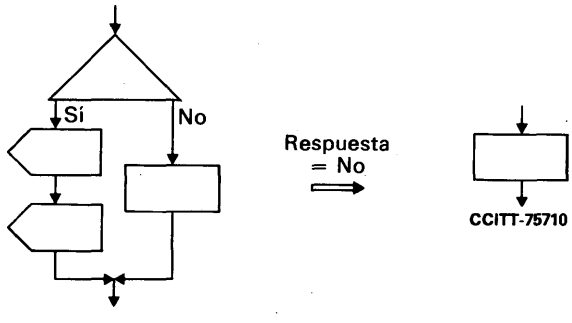


FIGURA D-112

Interpretación de opciones

D.6.3.6.16 *Conectores*

Consideraciones generales

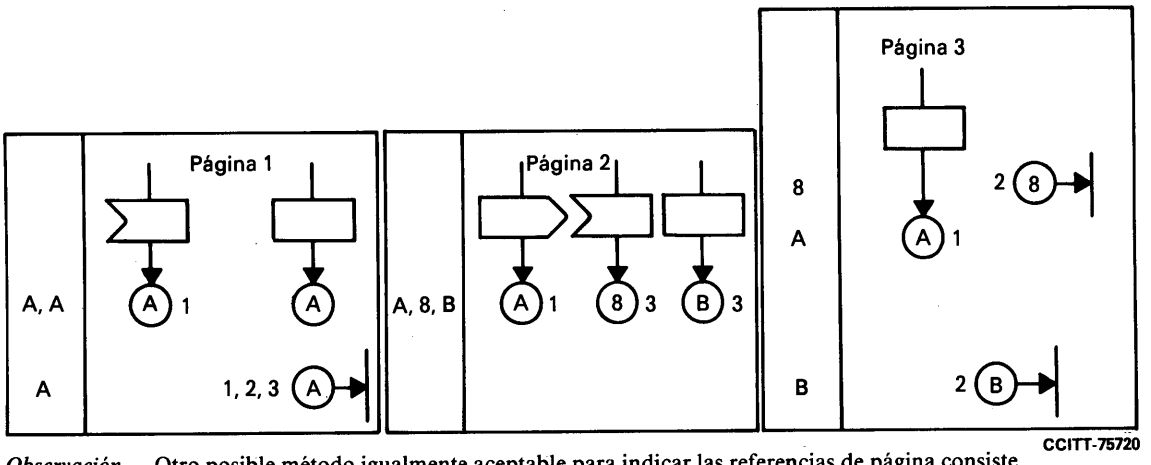
Al dibujar un diagrama LED grande, también puede ser necesario dividirlo debido a la falta de espacio. Para ello se utilizarán conectores.

Se utilizan también conectores para evitar cruces de líneas de flujo que harían que los diagramas queden algo confusos. Normalmente es preferible dibujar un diagrama LED de manera que el flujo vaya de la parte alta a la parte baja de la página.

Utilización de conectores para indicar líneas de flujo implícitas

Toda línea de flujo puede ser interrumpida por un par de conectores asociados; se supone que el flujo va del conector de salida al conector de entrada. Cada conector tiene un nombre. Los conectores asociados tienen el mismo nombre. Para cada nombre existe un solo conector de entrada, pero puede haber uno o más conectores de salida.

Es conveniente que la referencia de página para el conector de entrada apropiado aparezca junto a cada conector de salida y que la referencia o referencias de página para el conector o conectores de salida correspondientes aparezcan junto a cada conector de entrada (véase la figura D-113). Es también conveniente establecer una columna de referencia de conectores en el margen de la página, que indique el lugar horizontal y el orden de los conectores.



Observación – Otro posible método igualmente aceptable para indicar las referencias de página consiste en utilizar comentarios, como ilustra la figura D-114.

FIGURA D-113

Método para indicar referencias de página y referencias de conector

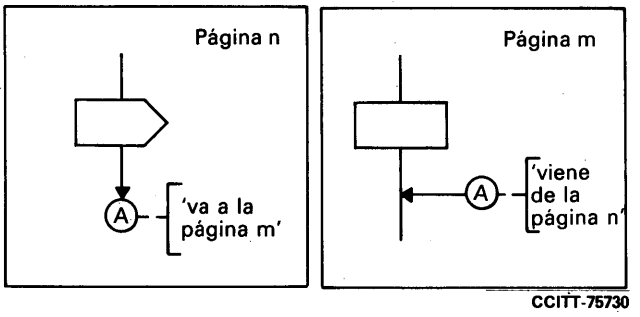


FIGURA D-114

Otro posible método para indicar referencias de página

Sólo pueden utilizarse conectores para conexiones dentro de un mismo proceso.

Cuando existen trayectos de flujo importantes, puede centrarse la atención en ellos no interrumpiéndolos con conectores excepto en las partes superior e inferior de las páginas. En cambio, un trayecto de flujo menor, dentro de un diagrama LED, puede interrumpirse con un conector de salida, colocando el conector de entrada asociado en alguna otra página adecuada, como la página que sigue al final de los trayectos de flujo principales.

D.6.3.6.17 *Divergencia y convergencia*

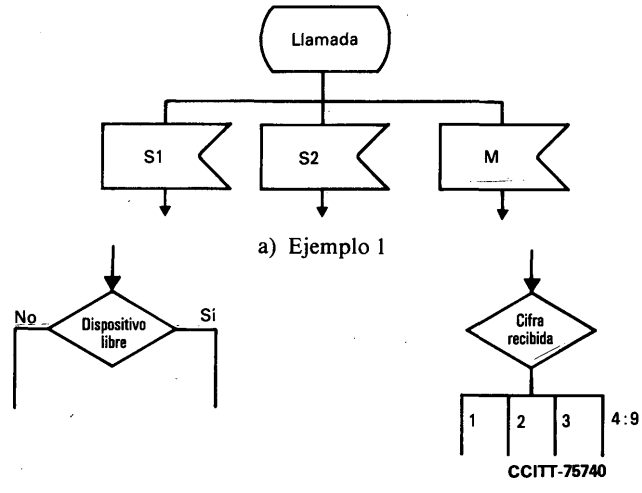
Divergencia

En una transición de un diagrama LED sólo puede producirse divergencia:

a) entre un símbolo de estado y sus símbolos asociados de entrada y de conservación, o

b) inmediatamente después de un símbolo de decisión.

La figura D-115 presenta algunos ejemplos de divergencias.



a) Ejemplo 1

b) Ejemplo 2

c) Ejemplo 3

FIGURA D-115

Ejemplos de divergencia

Convergencia

No puede producirse convergencia entre un símbolo de estado y uno de entrada o de conservación, pero puede producirse en cualquier otro punto de un diagrama LED.

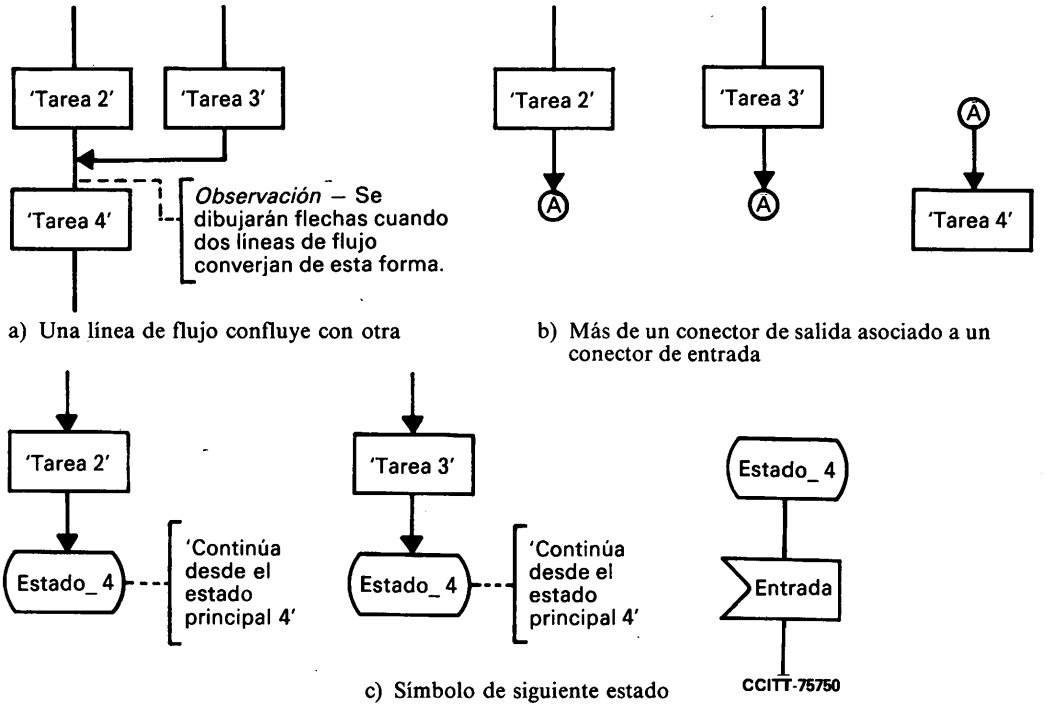
La convergencia puede aparecer de cualquiera de las cuatro formas mostradas en la figura D-116.

Utilizando la convergencia se puede reducir el número de símbolos en aquellos diagramas en los que una secuencia de símbolos y eventuales textos asociados se repite varias veces, como se muestra en la figura D-117.

D.6.3.6.18 *Comentarios*

Se pueden insertar comentarios en un diagrama LED para aclarar alguna parte del mismo. Los comentarios tienen por objeto ayudar al lector y carecen de efecto sobre la interpretación del diagrama LED. Por consiguiente, los comentarios no deben contradecir la semántica del diagrama ni añadir nada a la misma.

Un comentario va ligado a una línea de flujo o a un símbolo LED mediante un solo corchete conectado a la línea de flujo o al símbolo por medio de una línea de trazo discontinuo. Véase la figura D-118.



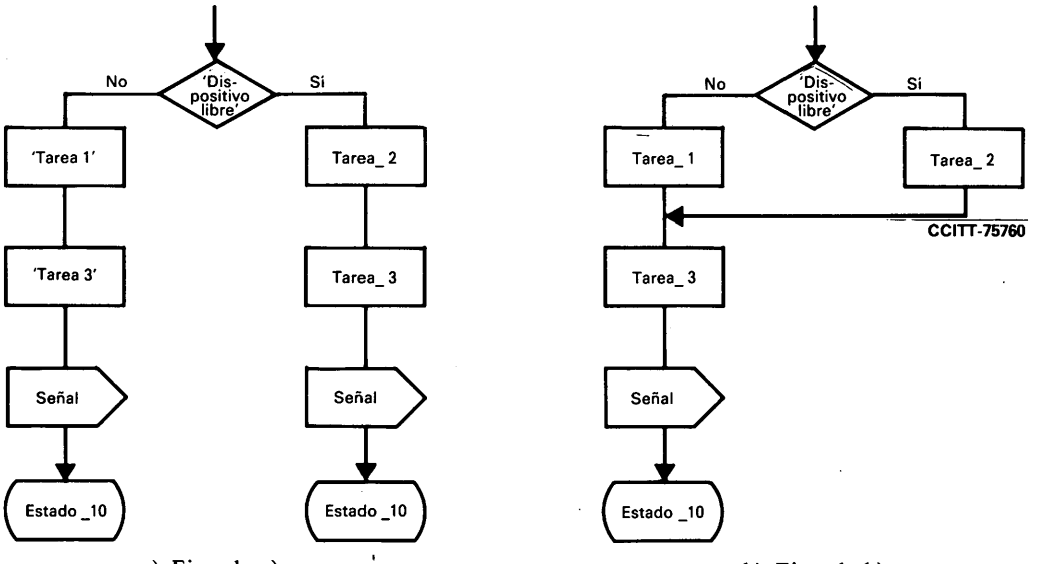
a) Una línea de flujo confluye con otra

b) Más de un conector de salida asociado a un conector de entrada

c) Símbolo de siguiente estado

FIGURA D-116

Ejemplos de convergencia



a) Ejemplo a)

b) Ejemplo b)

Observación – El ejemplo b) es equivalente al ejemplo a)

FIGURA D-117

Utilización de la convergencia

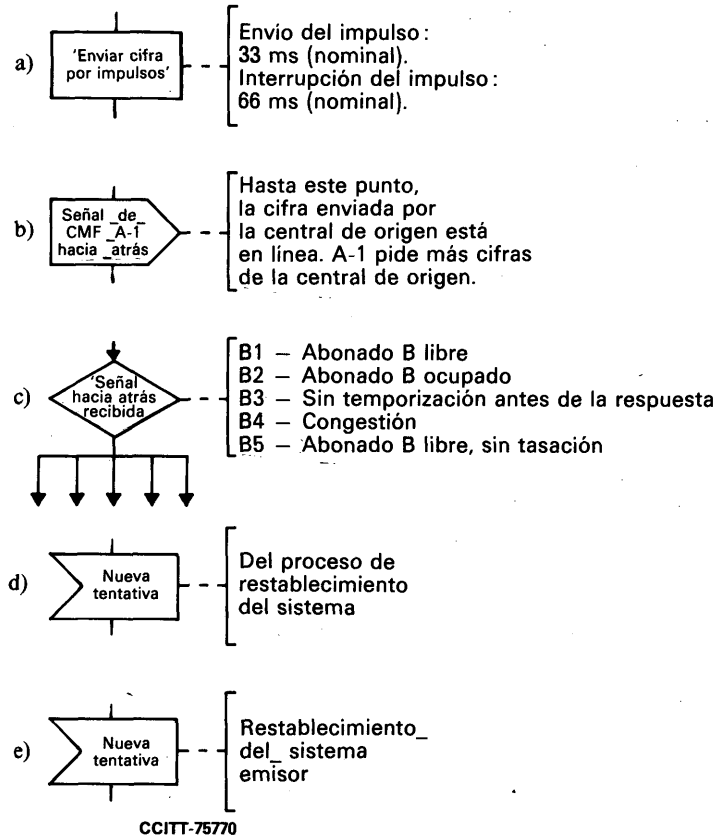


FIGURA D-118

Ejemplos de la utilización de comentarios

También pueden isertarse comentarios utilizando la sintaxis LED/PR (/*...*/) como puede verse en la figura D-119.

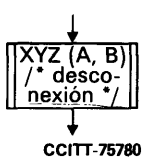


FIGURA D-119

Ejemplo del uso de la sintaxis de comentario LED/PR en el LED/GR

D.6.3.6.19 *Extensión de texto*

Normalmente, el texto asociado a un símbolo gráfico debe colocarse dentro de ese símbolo. Pero a menudo ello no resulta práctico o posible y una solución consiste entonces en colocar el texto en un símbolo de extensión de texto conectado al símbolo en cuestión. Véase la figura D-120.

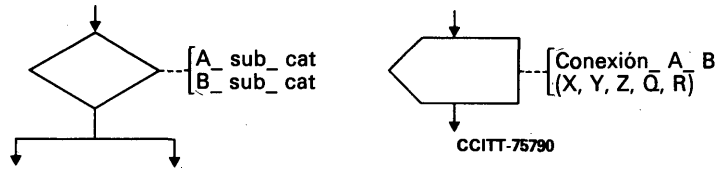


FIGURA D-120

Ejemplo del uso del símbolo de extensión de texto

Como quiera que la sintaxis es similar a la sintaxis de comentario, es importante que la línea de unión sea una línea de trazo lleno, a diferencia de la línea de trazos discontinuos del símbolo de comentario.

Si se utiliza texto informal en un símbolo de extensión de texto, el mismo debe estar encerrado entre apóstrofos.

D.6.3.6.20 *Datos*

Generalmente se utiliza la sintaxis LED/PR para las construcciones de datos en el LED/GR. Cuando se utilizan datos en tareas, salidas, decisiones o entradas, el texto LED/PR apropiado va asociado al símbolo.

Las definiciones de tipo de datos y las declaraciones de variable, o bien constan en documentos separados y están referenciadas, o bien aparecen en el diagrama del proceso. Estas definiciones deben estar asociadas al símbolo de comienzo del diagrama.

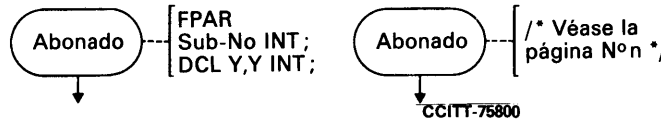


FIGURA D-121

Ejemplos de definiciones de datos en un diagrama de proceso LED/GR

La asociación con el símbolo se efectúa por medio del símbolo de extensión de texto, como puede verse en la figura D-121.

D.6.3.6.21 *Notaciones estenográficas*

Para hacer más legibles los diagramas y evitar largas listas de nombres, se introducen notaciones estenográficas. Los símbolos utilizados para ello son el asterisco (*) y el guión (–). Por lo general, el «*» tiene el significado de «todo» o «todos excepto», y el «–» el significado de «el mismo».

Estas notaciones estenográficas pueden emplearse en los símbolos de ESTADO, ENTRADA, CONSERVACIÓN y SIGUIENTE ESTADO.

En el símbolo de estado se puede usar un asterisco, ya sea solo o en combinación con una lista de nombres de estados encerrada entre paréntesis:

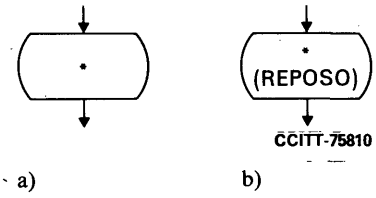


FIGURA D-122

Ejemplo de la notación «*» en estados

El asterisco que aparece solo en la figura D-122 debe interpretarse como «todos los estados» para (a) y la combinación (b) como «todos los estados excepto REPOSO». La notación utiliza la facilidad de apariciones múltiples de símbolos de estado (véase el § D.6.3.6.5).

En el caso de las entradas, únicamente puede utilizarse un asterisco en una de las entradas conectadas y un estado, a condición de que no aparezca en un símbolo de conservación conectado al mismo estado.

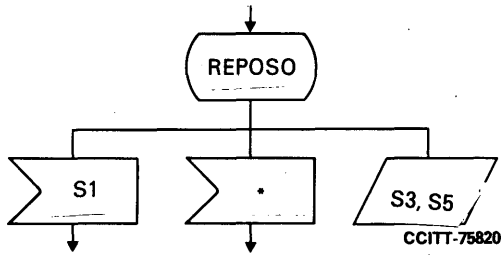


FIGURA D-123

Uso de un asterisco en un símbolo de entrada

La interpretación del asterisco en la figura D-123 es «todas las señales exceptuadas las señales mencionadas en otras entradas o conservaciones, es decir, S1, S3 y S5».

El uso de un asterisco en un símbolo de conservación es similar. Puede aparecer también en una de las conservaciones conectadas a un estado, a condición de que no aparezca en ninguna entrada conectada al mismo estado.

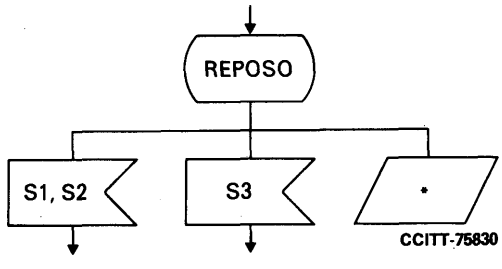


FIGURA D-124

Uso de un asterisco en un símbolo de conservación

En la figura D-124 la interpretación del asterisco es «todas las señales exceptuadas S1, S2 y S3».

En un símbolo de estado siguiente puede utilizarse un guión («–») para representar «el mismo estado que aquél en que se originó la transición».

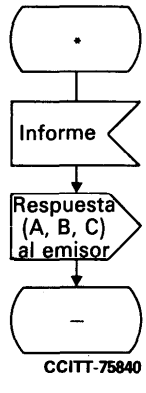


FIGURA D-125

Uso de un guión en un símbolo de estado siguiente

La interpretación de la figura D-125 anterior es que en cualquier estado del proceso puede recibirse la señal «INFORME». Su recepción causará el envío de la señal «RESPUESTA», y la transición terminará en el estado en que se originó.

En combinación con la facilidad de apariciones múltiples, estas notaciones estenográficas poseen un gran poder expresivo. Obsérvese que unas adiciones simples a un diagrama, utilizando «*» y «–», pueden alterar su significado de una manera inesperada.

A modo de ejemplo, la adición al diagrama mostrado en la figura D-126 siguiente entraña que en todos los estados no habrá ninguna transición implícita.

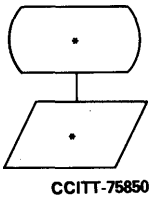


FIGURA D-126

Un efecto del diagrama es que no se permite « * » en ningún símbolo de entrada, estado o conservación en ninguna parte del diagrama

D.6.3.6.22 *Pictogramas de estado*

D.6.3.6.22.1 *Opción de pictograma de estado*

Si se elige la opción de pictograma de estado, resulta útil considerar que cada pictograma de estado conste de los tres tipos de elementos que se ilustran en la figura D-127 a).

Estos tres tipos se combinan a menudo para formar un elemento pictográfico compuesto que pueda comprenderse aisladamente del resto del pictograma de un estado. Por ejemplo, la figura D-127 b) significa un receptor de código multifrecuencia que está esperando una señal hacia adelante.

Obsérvese que el símbolo del elemento pictográfico recomendado para la supervisión del tiempo en un proceso comprende la variable de entrada t_i correspondiente.

1) Elementos pictográficos, por ejemplo ⟩ (significa un receptor de señalización);
2) Variables de entrada, por ejemplo, f (significa que todavía ha de reconocerse una señal hacia adelante); y
3) Texto calificador, por ejemplo, CMF (significa «código multifrecuencia»).

a) Contenido de un pictograma de estado

b) Elemento pictográfico compuesto

FIGURA D-127

Construcción de elementos pictográficos

D.6.3.6.22.2 *Variables de entrada*

Conviene representar como variables de entrada las condiciones asociadas con el proceso que, al ser cambiadas, provocan una transición del proceso. Las variables de entrada deben indicarse con letras minúsculas, para distinguirlas fácilmente del texto calificador (que debe escribirse con mayúsculas). (Los cambios en las variables de entrada corresponden a señales de entrada, que hacen que el proceso abandone su estado vigente, en tanto que los cambios en el texto calificador no corresponden a señales de entrada.)

D.6.3.6.22.3 *Texto calificador*

El texto calificador debe ser muy abreviado y, de ser posible, colocarse dentro de los elementos pictográficos correspondientes. De esta manera se distinguen claramente los elementos pictográficos que se califican.

D.6.3.6.22.4 *Pictogramas de estado completos*

A cada pictograma de estado debe asignarse un número suficiente de elementos pictográficos que permitan indicar:

a) los recursos que está considerando el proceso en el estado vigente. Ejemplos de recursos son: trayectos de conmutación, receptores de señalización, emisores de señalización y módulos de conmutación;
b) si el proceso está supervisado en esos momentos por uno o más temporizadores;
c) en caso de que el proceso en cuestión concierna al tratamiento de la llamada, si hay o no tasación en curso y los abonados afectados por la tasación en este estado de la llamada;
d) las categorías (o estados) vigentes de equipo asignados a este proceso, cuando esta información afecte al comportamiento del proceso;
e) el estado de las variables de entrada que son supervisadas por el proceso durante el estado vigente.

D.6.3.6.22.5 *Ejemplo*

Un ejemplo de la aplicación de los principios mencionados se muestra en el pictograma de estado de la figura D-128. Puede observarse que durante este estado:

a) los recursos considerados por el proceso consisten en un aparato telefónico de abonado, un receptor de cifras, un emisor de tono de invitación a marcar y trayectos de conmutación que conectan estos elementos;
b) que el proceso está supervisado en ese momento por un temporizador T_0 (cuya condición vigente es t_0);
c) que no hay en curso tasación de la comunicación;
d) que el abonado ha sido identificado como abonado A, pero no se tiene en cuenta otra información acerca de su categoría;
e) que son aplicables las siguientes condiciones de las variables de entrada: h_A (esto es, el teléfono está descolgado), $\bar{d}$ (esto es, el receptor de cifras está esperando una cifra) y t_0 (esto es, el temporizador de supervisión T_0 está funcionando).

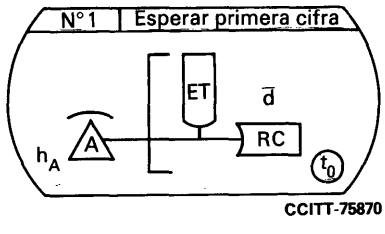


FIGURA D-128

Ejemplo de un estado del proceso de tratamiento de una llamada

D.6.3.6.22.6 *Verificación de la coherencia de los diagramas LED con elementos pictográficos*

Si se sigue el principio expresado en el apartado e) del § D.6.3.6.22.4 será siempre posible deducir observando simplemente la variable de entrada indicada en el pictograma de estado, el conjunto de entradas como consecuencia de las cuales el proceso abandonará cada uno de sus estados. Por ejemplo, si observamos el pictograma de estado de la figura D-128, podemos deducir que tres entradas diferentes harán que el proceso abandone este estado: la condición de teléfono cambia a colgado (entrada $\bar{h}_A$); llegada de una cifra (entrada d); o expiración del temporizador T_0 (entrada $\bar{t}_0$). De esta manera A puede evitarse algunas «transiciones sorpresa» cuando se procede a la realización de sistemas a partir de especificaciones LED muy complejas.

No cabe duda de que un pictograma es más conciso ni de que, en cierto modo, pone más información ante los ojos del lector, pero al mismo tiempo uno tiene que estudiarlo detenidamente para deducir el conjunto exacto de operaciones que tienen lugar en la transición.

Además, no es posible decir, mirando solamente el diagrama pictográfico, si una acción tiene lugar en virtud de una salida o de una tarea (véase el § D.6.3.6.22.8).

D.6.3.6.22.7 *Utilización de símbolos de frontera de bloque*

Los elementos pictográficos indicados fuera del símbolo de frontera de bloque son elementos que no están directamente controlados por el proceso dado, y los elementos pictograficos indicados dentro del símbolo de frontera de bloque son elementos que están controlados directamente por el proceso dado. Por ejemplo, el proceso de llamada parcialmente especificado en la figura D-129 puede conectar o desconectar la corriente de llamada y arrancar o parar el temporizador T_4, pero no puede cambiar ninguna de las condiciones del teléfono del abonado.

En el diseño de la lógica a partir de una especificación LED con elementos pictográficos, sólo los elementos pictográficos indicados dentro de la frontera de bloque influirán en las secuencias de acciones de proceso que tienen lugar durante las transiciones. Normalmente, los elementos pictográficos compuestos que aparecen fuera de la frontera de bloque se incluyen en un pictograma de estado por las razones siguientes:

a) porque indican variables de entrada que deben ser supervisadas por el proceso durante el estado en cuestión, y/o

b) para mejorar la inteligibilidad del diagrama.

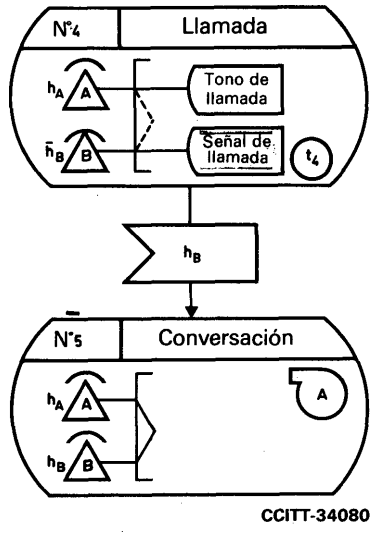


FIGURA D-129

Ejemplo de transición entre dos estados en que todas las acciones de proceso se deducen de las diferencias entre los pictogramas de estado

D.6.3.6.22.8 *Tarea o salida*

La interpretación de una acción de proceso como una tarea o como una salida a veces parece arbitraria. En realidad, la decisión de interpretar la aparición o desaparición de un elemento pictográfico dentro de la frontera de bloque como una tarea o como una salida sólo puede resolverse consultando la lista de señales del proceso.

D.6.3.6.22.9 *Utilización del símbolo de temporizador*

Empléense o no elementos pictográficos, la interrupción de los procesos supervisados al expirar un periodo del temporizador se representa siempre por una entrada.

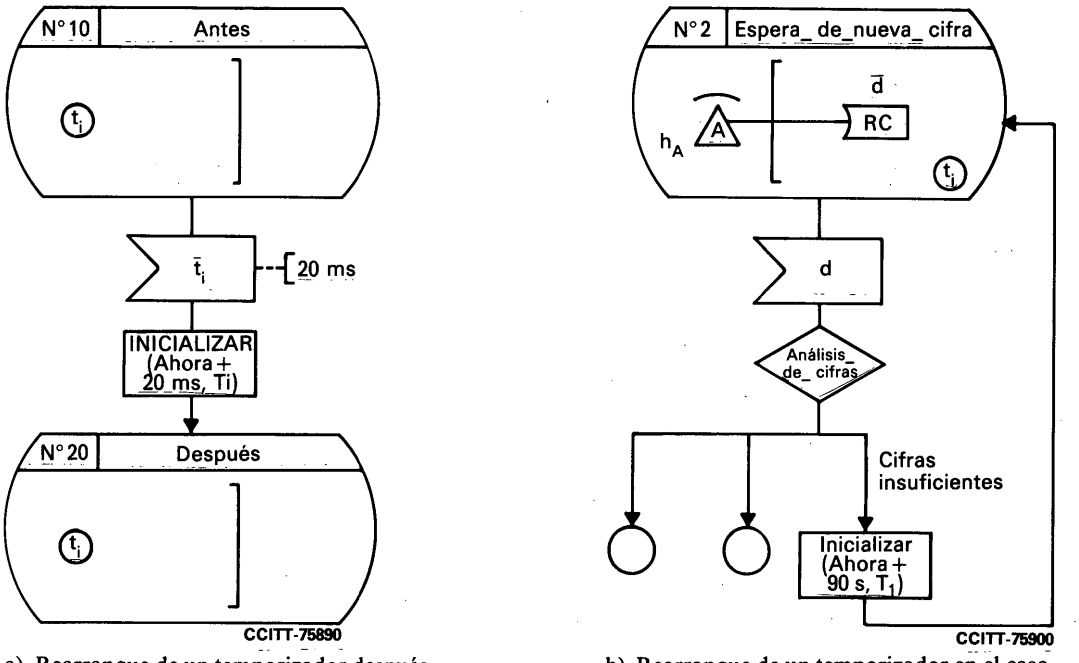
La presencia de un símbolo de temporizador en un pictograma de estado implica que un temporizador está funcionando durante ese estado. De acuerdo con el principio general enunciado en las Recomendaciones, el arranque, parada, rearranque y expiración de temporizadores se indican utilizando elementos pictográficos de la manera siguiente:

a) Para indicar que un temporizador se arranca (se pone en marcha) durante una transición dada, el símbolo del temporizador debe aparecer en el pictograma de estado que corresponde al fin de la transición, pero no en el pictograma de estado que corresponde a la transición.

b) Para indicar que un temporizador ha sido parado (detenido) en el curso de una transición, el símbolo del temporizador debe aparecer en el pictograma de estado que corresponde al comienzo de la transición, pero no en el pictograma de estado que corresponde al fin de la transición.

c) Para indicar que un temporizador ha sido rearrancado durante una transición debe indicarse en esta transición un símbolo explícito de tarea (en la figura D-130 se presentan dos ejemplos).

d) La expiración del periodo de un temporizador determinado se indica mediante un símbolo de entrada asociado con un estado en que el pictograma de estado incluye el símbolo de temporizador correspondiente. Desde luego, si es preciso, más de un temporizador pueden estar supervisando el mismo proceso en un momento dado (véase la figura D-131).



a) Rearranque de un temporizador después de expirar el periodo del temporizador

b) Rearranque de un temporizador en el caso de que el periodo del temporizador no haya expirado todavía

Observación – Cada temporizador T_i tiene dos condiciones t_i y $\bar{t}_i$, que se excluyen mutuamente.

FIGURA D-130

Ejemplos que muestran el rearranque de un temporizador

D.6.3.6.23 *Documentos auxiliares*

Para facilitar la lectura y comprensión de los diagramas de proceso grandes resulta útil utilizar documentos auxiliares que presenten imágenes generales. Estos documentos son informales y sirven únicamente de imagen general, a modo de «entrada» a los diagramas LED. Los usuarios son libres de determinar por sí mismos una sintaxis adecuada.

En esta sección se ofrecen ejemplos de dos tipos de documentos auxiliares: el diagrama general de estados y la matriz de estados/señales.

D.6.3.6.23.1 *Diagrama general de estados*

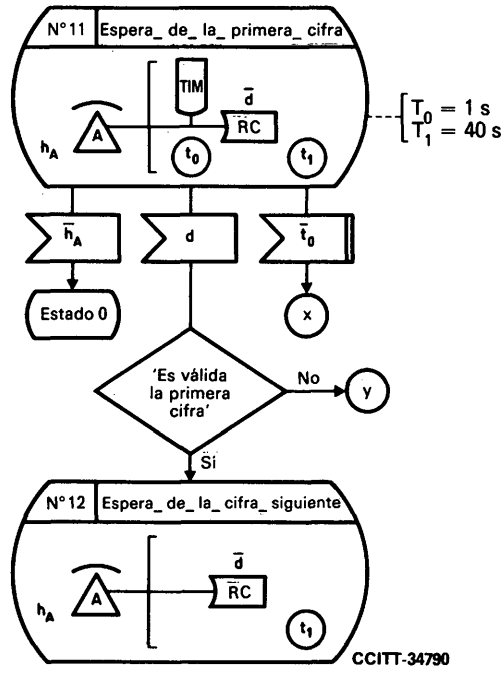
La finalidad del diagrama general de estados es dar una idea general de los estados de un proceso y de las transiciones que son posibles entre los mismos. Como su objeto es presentar una imagen general, pueden omitirse los estados o transiciones que no son importantes.

Los diagramas se componen de símbolos de estado, arcos dirigidos que representan las transiciones y, facultativamente, símbolos de comienzo y parada.

El símbolo de estado debe contener el nombre del estado referenciado. Puede contener varios nombres de estado, y puede utilizarse un asterisco (*) como notación de todos los estados.

A cada uno de los arcos dirigidos puede darse el nombre de la señal o conjunto de señales que causan la transición.

El empleo del símbolo de comienzo y de parada es facultativo.



El temporizador T_0 supervisa la llegada de la primera cifra, en cuyo momento se suprime el tono de invitación a marcar (TIM), y se para el temporizador T_0. El temporizador T_1 sigue supervisando la llegada de un número suficiente de cifras que permita el encaminamiento de la llamada.

FIGURA D-131

Ejemplo de utilización de dos temporizadores de supervisión en un mismo estado

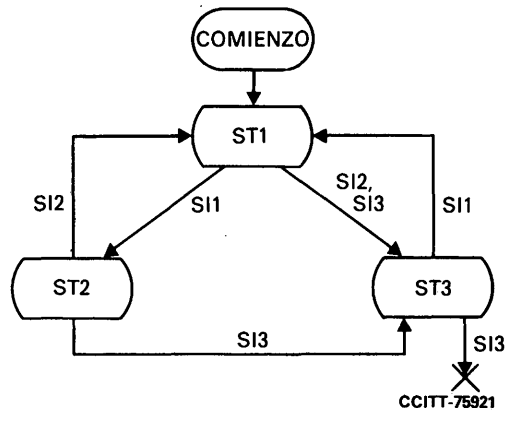


FIGURA D-132

Ejemplo de un diagrama general de estados

El diagrama general de estados de un proceso puede dividirse en varios diagramas relativos a diferentes aspectos, como por ejemplo «caso normal», tratamiento de fallo, etc.

FIGURA D-133

Ejemplo de un diagrama general de estados separados

Para que queden bien estructurados y sean fáciles de leer, los diagramas generales de estados debieran, de preferencia:

- amoldarse a las direcciones normales de lectura, de arriba hacia abajo y de izquierda a derecha;
- ocupar una sola página.

D.6.3.6.23.2 *Matriz de estados/señales*

El objeto de la matriz de estados/señales es constituir un documento de «entrada» a un gran diagrama de proceso. Ofrece referencias a los lugares del diagrama en que aparece una combinación de un estado y una señal determinados.

Consiste en una matriz bidimensional, en uno de cuyos ejes están indexados todos los estados de un proceso, y en el otro todas las señales de entrada válidas para un proceso. En cada elemento de la matriz se indica dónde se encuentra la combinación determinada por los índices, en caso de que exista.

ESTADOS / SEÑALES	REPOSO	OCUPADO	BLOQUEADO	------
DESCOLGADO	P5	–	–	
COLGADO	–	P6	–	
TOMA	P7	P8	–	
BLOQUEO	P6	P6	–	

CCITT-75940

FIGURA D-134

Ejemplo de una matriz de estados/señales

La matriz puede dividirse en subpartes que aparezcan en páginas diferentes. Las referencias son las normales que utiliza el usuario en la documentación.

Es preferible agrupar las señales y estados de manera tal que cada parte de la matriz cubra un aspecto del comportamiento del proceso; por ejemplo, el «caso normal», el «tratamiento de excepción» y la «parte de mantenimiento» debieran aparecer en subgrupos diferentes.

D.6.3.6.23.3 *Esquemas secuenciales*

Como complemento del diagrama de interacciones puede elaborarse un esquema secuencial para mostrar las secuencias permitidas de las señales intercambiadas entre uno o varios procesos y sus entornos.

Este esquema tiene por objeto dar una idea general del intercambio de señales entre las partes del sistema. Puede representar el intercambio completo de señales o sólo partes de él, según los aspectos que se desee resaltar.

Las columnas del esquema indican el entorno (colocado por lo general en los bordes del diagrama) y los diversos procesos o bloques.

Sus interacciones se representan mediante un conjunto de líneas diagonales dirigidas, cada una de las cuales representa una señal o un conjunto de señales. Cuando un conjunto de señales representado por una línea fluye en ambos sentidos, se sugiere que dicha línea sea horizontal (un ejemplo puede ser el «establecimiento de la conexión»), con flechas en ambos extremos.

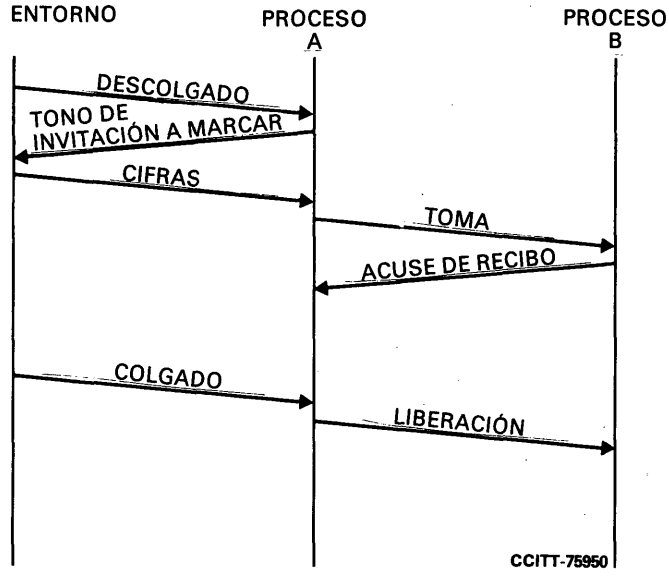


FIGURA D-135

Ejemplo de un esquema secuencial

Se puede anotar cada secuencia para que quede claro que el conjunto de informaciones se intercambia. Cada línea se anota con la información pertinente (nombres de señales o procedimientos).

En el punto de llegada de una línea se puede colocar un símbolo de decisión para indicar que la secuencia siguiente es válida si la condición indicada es cierta. En este caso el símbolo de decisión suele aparecer varias veces, indicando las diferentes secuencias que resultan de los diferentes valores de la condición.

Este diagrama puede mostrar completamente todas las secuencias de señales intercambiadas o solamente un subconjunto significativo de ellas.

Una aplicación útil de este diagrama es la representación de la interacción mutua de subproceso de resultas de la partición de un proceso. En este caso, el entorno de los subprocesos es el entorno del proceso subdividido.

D.7 *Directrices para representar sistemas por medio del LED/PR*

El § 7 de las Directrices para el usuario explica la representación de frases textuales del LED, denominada PR, que tiene un gran parecido con un lenguaje de programación.

La primera parte (§ D.7.1) contiene observaciones generales sobre el LED/PR, su conveniencia para utilizarlo como entrada en una máquina, su analogía con un programa y sus diferencias.

En la segunda parte (§ D.7.2) se explica cómo se define el LED/PR en las Recomendaciones, en donde se utilizan los «diagramas sintácticos».

La tercera parte (§ D.7.3) es la más importante desde el punto de vista del usuario. Explica la pragmática del uso del LED, y en particular las peculiaridades y la capacidad expresiva de la representación de frases textuales, citando aspectos tales como el orden de estados, la ordenación de las transiciones dentro de un estado, la multiplicidad de estados, los procedimientos, los macros, las etiquetas y las notaciones abreviadas.

D.7.1 *¿Por qué el PR?*

La representación de frases textuales del LED, el LED/PR, se elaboró en el Periodo de Estudios 1977-1980; en 1980 se decidió incorporar su definición como un anexo a las Recomendaciones porque se estimó que se necesitaban ciertos perfeccionamientos antes de recomendarla. Esos perfeccionamientos se efectuaron en el siguiente Periodo de Estudios y ahora el LED/PR es una de las sintaxis concretas recomendadas del LED.

Al principio, el LED/PR estaba destinado a ser un modo fácil de introducir documentos LED en un aparato, poruqe la forma GR es de entrada más difícil (se necesitan dispositivos periféricos gráficos para tratarla). Por ese motivo se hizo hincapié en el mapeado uno a uno entre el PR y el GR. La evolución de los terminales gráficos (aumento de las capacidades y disminución del costo) ha hecho que la forma GR sea ahora apropiada para la entrada en un aparato. Ello no disminuye la importancia y el empleo de la forma PR, que agrada más a ciertos usuarios, en particular a los que trabajan con lenguajes de programación.

D.7.1.1 *¿Cuán cerca de un programa está el LED/PR?*

Esta evolución condujo a una correlación más estrecha entre el GR y el PR, de modo que todavía podamos mapear (con facilidad) uno en otro, pero cada lenguaje tiene sus propias características. A primera vista, la forma PR tiene un gran parecido con un lenguaje de programación, véase la figura D-136.

```
–
–
–
–
STATE aw_off_hook;
INPUT off_hook;
TASK 'activate charging';
TASK 'connect';
OUTPUT 'reset timer';
NEXTSTATE conversation.
–
–
–
–
```

FIGURA D-136

Ejemplo del LED/PR

En la práctica todo depende de lo que caracteriza a un texto como lenguaje de programación.

Si suponemos que un programa es por definición un conjunto de información interpretable por un aparato, entonces no sólo el PR sino también el GR son «programas». Si se limita la definición a un conjunto de información y directrices interpretables y realizables por un aparato, hallamos la primera diferencia: no es indispensable que una representación PR sea realizable por el aparato (aunque no está prohibido), pero lo esencial es su capacidad de transmitir información precisa de una persona a otra.

Lo que podemos estimar como un «PR erróneo», si lo consideramos como un programa (debido a su forma incompleta o a su texto informal), es perfectamentne válido si se considera al lenguaje PR como una representación de los requisitos funcionales de un sistema.

Otra diferencia reside en el «estilo» de una representación PR en comparación con una representación usual de programa.

Dado que el PR está destinado a facilitar la comunicación entre personas, se tienen que pemitir distintas presentaciones de modo que esta presentación PR pueda utilizarse para guiar al lector a fin de que se centre en determinados aspectos que se consideran más importantes. Es evidente que ello carece de importancia en un programa que se supone ha de ser interpretado por un aparato. El aparato no se centra en un aspecto determinado sino que considera por igual el conjunto; tampoco trata de «comprender» el programa.

Por su analogía con un programa (que es más una similitud psicológica porque la forma GR es tan parecida «semánticamente» a un programa como lo es la forma PR), el PR es preferido por algunos programadores que probablemente utilizarán el lenguaje CHILL para realizar los requisitos expresados en PR. Por consiguiente, existe la fuerte tentación de hallar un mapeado de uno a uno del PR al CHILL, de modo que los requisitos expresados en el PR puedan transformarse automáticamente en el código CHILL. Lo contrario es también interesante porque permitiría la derivación de una descripción PR a partir de un programa CHILL.

En el § D.8 se ilustran los posibles modos de mapeado entre el LED y el CHILL.

D.7.2 *Metalenguaje para describir la sintaxis LED/PR: diagramas sintácticos*

Un diagrama sintáctico consiste en símbolos terminales y no terminales interconectados por líneas de flujo.

Un símbolo no terminal representa otro diagrama sintáctico con el mismo nombre. Es un símbolo abreviado para estructuras más complejas utilizadas en distintos lugares (estructuras que consisten también en símbolos terminales y no terminales).

Existe una notación especial para insertar comentarios.

Cada símbolo, esto es, cada diagrama, debe tener un solo acceso de entrada y un solo acceso de salida.

Las líneas de flujo son unidireccionales y una flecha indica el sentido del flujo.

Los símbolos terminales están indicados por rectángulos, cuyos lados verticales se han sustituido por semicírculos. Un símbolo no terminal está señalado por un rectángulo.

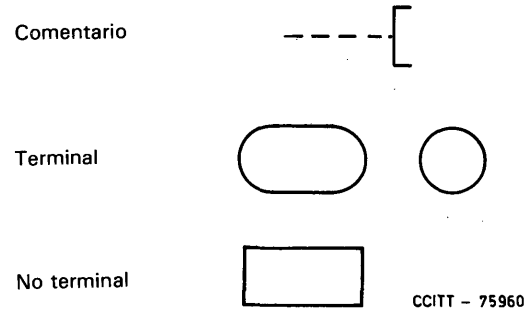


FIGURA D-137
Símbolos gráficos en diagramas sintácticos

D.7.3 *Utilización del PR*

En el LED/PR, la descripción completa del sistema comprende numerosos documentos, lo que facilita el tratamiento de la información. El diagrama de interacción de bloques representa la estructura del sistema y la comunicación entre bloques y entre procesos vía canales y señales, en tanto que los diagramas de procesos se representan en otros documentos.

En PR, la estructura del sistema y su comunicación se pueden representar dividiendo el program PR completo del sistema en diversos módulos, cada uno de los cuales describe la estructura de uno o más procesos. De esta forma cada módulo se puede tratar con más facilidad.

Entre estos módulos puede existir un mecanismo de referencia, como se explica en el § D.5.

La versión PR consiste en una serie de sentencias terminales cada una por «;» (punto y coma). Comienza con la sentencia

SISTEMA nombre;

y termina con la sentencia

FINSISTEMA nombre;

En esta sentencia de cierre, el «nombre» es facultativo.

La representación PR no tiene que consistir sólo en un único conjunto de sentencias incluidas entre las sentencias SISTEMA-FINSISTEMA. Podemos tener varios conjuntos que comience cada uno con una sentencia SISTEMA y termine con una declaración FINSISTEMA. Pertenecen a la misma representación de SISTEMA siempre y cuando tengan el mismo nombre (véase la figura D-138). Cada módulo de proceso debe incluirse en un solo conjunto y no se puede dividir en dos conjuntos.

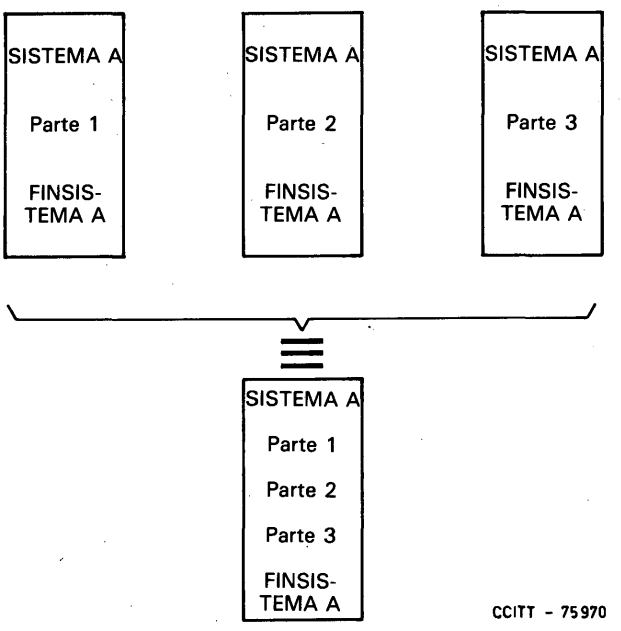


FIGURA D-138

Separación de una representación de sistema en tres

Se pueden agrupar las sentencias contenidas entre SISTEMA y FINSISTEMA en módulos de bloques y en definiciones de señal, canal, tipo de datos, macro y procedimientos, como puede observarse en la figura D-139.

SISTEMA .....

Definiciones de señal

Definiciones de canal

Definiciones de tipos de datos

Definiciones de macro

Definiciones de procedimiento

[BLOQUE... / FINBLOQUE] } Módulo de bloque

[BLOQUE... / FINBLOQUE] } Módulo de bloque

FINSISTEMA ....

FIGURA D-139

Agrupación de las sentencias contenidas en un módulo de SISTEMA

Cada módulo de bloque está incluido entre las sentencias BLOQUE y FINBLOQUE. Como una representación de sistema, una representación de bloque se puede dividir en dos partes. Se puede considerar a su vez cada módulo de BLOQUE como una serie de módulos PROCESO asociados con definiciones de señales, de datos, de procedimientos y de macro. Cada módulo de proceso está incluido entre las sentencias PROCESO y FINPROCESO. El módulo PROCESO no puede subdividirse de nuevo. Esto se ilustra en los ejemplos de la figura D-140, en la que un sistema está dividido en dos partes, a saber, una que contiene el bloque b1 con el proceso p2, el bloque b2 con el proceso p4 y el bloque b3 con el proceso p5 (figura D-140a)) y una parte que contiene el bloque b1 con los procesos p1 y p3 y el bloque b3 con el proceso p6 (figura D-140b)). La figura D-140c) contiene la representación global de esas dos partes.

```
SYSTEM a;
BLOCK b1;
    PROCESS p2;
    -
    -
    -
    ENDPROCESS p2;
ENDBLOCK b1;
BLOCK b2;
    PROCESS p4;
    -
    -
    -
    ENDPROCESS p4;
ENDBLOCK b2;
BLOCK b3;
    PROCESS p5;
    -
    -
    -
    ENDPROCESS p5;
  ENDBLOCK b3;
ENDSYSTEM a;
```

a) Parte 1

```
SYSTEM a;
  BLOCK b1;a
    PROCESS p1;
    -
    -
    -
    ENDPROCESS p1;
    PROCESS p3;
    -
    -
    -
    ENDPROCESS p3;
  ENDBLOCK b1;
  BLOCK b3;
    PROCESS p6;
    -
    -
    -
    ENDPROCESS p6;
  ENDBLOCK b3;
ENDSYSTEM a;
```

b) Parte 2

```
SYSTEM a;
  BLOCK b1;
    PROCESS p1;
    -
    -
    -
    ENDPROCESS p1;
    PROCESS p2;
    -
    -
    -
    ENDPROCESS p2;
    PROCESS p3;
    -
    -
    -
    ENDPROCESS p3;
  ENDBLOCK b1;
  BLOCK b2;
    PROCESS p4;
  -
    -
    -
    ENDPROCESS p4;
  ENDBLOCK b2;
  BLOCK b3;
    PROCESS p5;
    -
    -
    ENDPROCESS p5;
    PROCESS p6;
    -
    -
    ENDPROCESS p6;
  ENDBLOCK b3;
ENDSYSTEM a;
```

c) Global

FIGURA D-140

Representación de bloques

D.7.3.1 *Expresión de la estructura en PR*

Los documentos que son útiles en GR son el árbol de bloques, el árbol de procesos y el diagrama de subestructura de canales. Se pueden obtener representaciones equivalentes en PR utilizando subestructuras de bloques y canales en la forma explicada en la Recomendación Z.102.

En PR, el árbol de bloques, el árbol de procesos y el diagrama de interacción de bloques se emplean combinados. La estructura obtenida contiene asimismo las sentencias de proceso, es decir, representa el sistema completo.

La correspondencia directa con estos documentos se indica mediante la lista de sub-bloques, canales y señales que se puede incluir en la parte superestructura de bloques, en tanto que la estructura de sistemas se da por las definiciones de jerarquización de bloques.

Las figuras D-141 y D-142 contienen el árbol de procesos y el árbol de bloques GR a que corresponden los ejemplos de la figura D-143 relativos al PR.

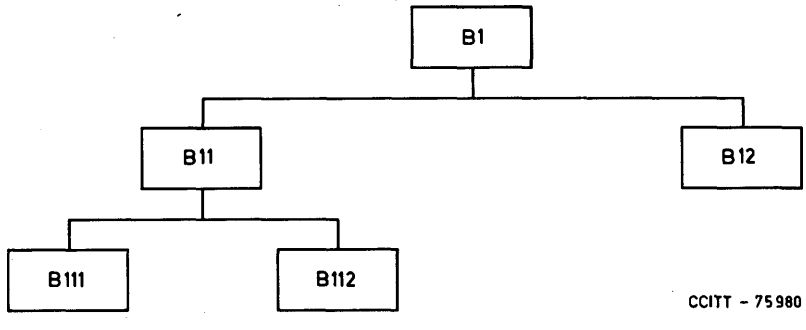


FIGURA D-141
Árbol de bloques

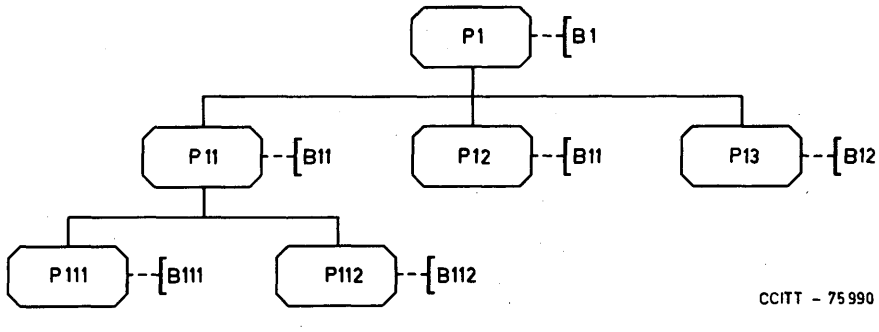


FIGURA D-142
Árbol de procesos

```
SYSTEM J;
- - - -
BLOCK B1;
- - - -
SUBSTRUCTURE B1;
SUBBLOCKS: B11, B12;
- - - -
BLOCK B11;
- - - -
SUBSTRUCTURE B11;
SUBBLOCKS: B111, B112;
- - - -
BLOCK B111;
- - - -
ENDBLOCK B111;
BLOCK B112;
- - - -
ENDBLOCK B112;
ENDSUBSTRUCTURE B11;
ENDBLOCK B11;
BLOCK B12;
- - - -
ENDBLOCK B12;
ENDSUBSTRUCTURE B1;
ENDBLOCK B1;
ENDSYSTEM J;
```

FIGURA D-143

Ejemplo de definiciones de jerarquización del bloque

En PR, los procesos y sus estructuras se representan en el interior de sus bloques. Hay dos formas posibles para mostrar la subestructura de procesos, a saber, en la subestructura de bloques de nivel superior o en el interior de cada proceso de nivel superior. Las figuras D-144 y D-145 ilustran estas dos posibilidades.

```
SYSTEM s;
- - - -
BLOCK b1;
- - - -
PROCESS p1;
- - - -
ENDPROCESS p1;
SUBSTRUCTURE b1;
SUBBLOCKS : b11, b12;
- - - -
SUBSTRUCTURE p1;
  p11 in b11,
  p12 in b12,
ENDSUBSTRUCTURE p1;
BLOCK b11;
- - - -
PROCESS p11;
- - - -
ENDPROCESS p11;
SUBSTRUCTURE b11;
SUBBLOCKS : b111, b112;
- - - -
SUBSTRUCTURE p11;
  p111 in b111,
  p112 in b112,
ENDSUBSTRUCTURE p11;
BLOCK b111;
- - - -
PROCESS p111;
- - - -
ENDPROCESS p111;
ENDBLOCK b111;
BLOCK b112;
- - - -
PROCESS p112;
- - - -
ENDPROCESS p112;
ENDBLOCK b112;
ENDSUBSTRUCTURE b11;
ENDBLOCK b11;
BLOCK b12;
- - - -
PROCESS p12;
- - - -
ENDPROCESS p12;
ENDBLOCK b12;
ENDSUBSTRUCTURE b1;
ENDBLOCK b1;
ENDSYSTEM s;
```

FIGURA D-144

Representación de un sistema con subestructura de procesos en el nivel de subestructura de bloques

```
SYSTEM s;
- - - -
BLOCK b1;
- - - -
PROCESS p1;
- - - -
SUBSTRUCTURE p1;
  p11 in b11,
  p12 in b12,
ENDSUBSTRUCTURE p1;
ENDPROCESS p1;
SUBSTRUCTURE b1;
SUBBLOCKS : b11, b12;
- - - -
BLOCK b11;
- - - -
PROCESS p11;
- - - -
SUBSTRUCTURE p11;
  p111 in b111;
  p112 in b112;
ENDSUBSTRUCTURE p11;
ENDPROCESS p11;
SUBSTRUCTURE b11;
SUBBLOCKS : b111, b112;
- - - -
BLOCK b111;
- - - -
PROCESS p111;
- - - -
ENDPROCESS p111;
ENDBLOCK b111;
BLOCK b112;
- - - -
PROCESS p112;
- - - -
ENDPROCESS p112;
ENDBLOCK b112;
ENDSUBSTRUCTURE b11;
ENDBLOK b11;
BLOCK b12;
- - - -
PROCESS p12;
- - - -
ENDPROCESS p12;
ENDBLOCK b12;
ENDSUBSTRUCTURE b1;
ENDBLOCK b1;
ENDSYSTEM s;
```

FIGURA D-145

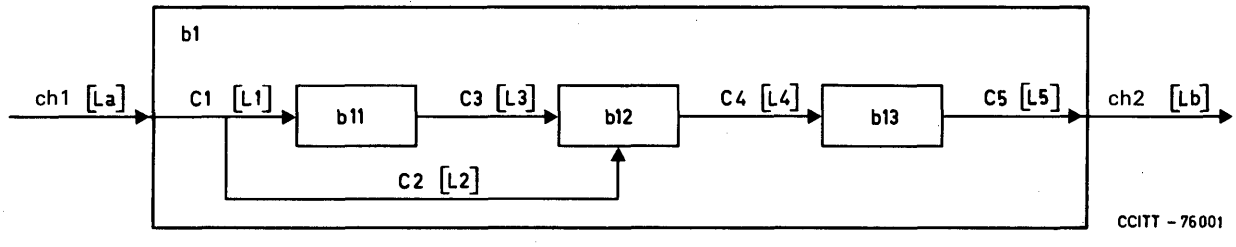
Representación de un sistema con subestructura de procesos dentro de la definición de proceso

Los canales que interconectan los sub-bloques obtenidos mediante la partición están definidos en la definición de la subestructura de bloques.

Se considera conveniente que la definición de canal figure antes que la definición de los sub-bloques, pues éstos son muy numerosos (véase la figura D-146).

Las señales correspondientes a estos canales están definidas en la parte de definición de la señal.

La figura D-147 se refiere a la representación GR de la figura D-146.



$La = \begin{bmatrix} s1, \\ s2 \end{bmatrix}$ $Lb = \begin{bmatrix} s3, \\ s4 \end{bmatrix}$

$L1 = [s1]$ $L2 = [s2]$

$L3 = \begin{bmatrix} s5, \\ s6 \end{bmatrix}$ $L4 = \begin{bmatrix} s7, \\ s8 \end{bmatrix}$

$L5 = \begin{bmatrix} s3, \\ s4 \end{bmatrix}$

FIGURA D-146

Representación de un sistema en dos niveles utilizando el diagrama de interacción de bloques

```
SYSTEM s;
CHANNEL ch1
  FROM ENV TO b1
  WITH s1, s2;
CHANNEL ch2
  FROM b1 TO ENV
  WITH s3, s4:
BLOCK b1;
- - - -
SUBSTRUCTURE b1;
SUBBLOCKS : b11, b12, b13;
CHANNELS : C1, C2, C3, C4, C5
  SPLIT ch1 INTO C1, C2;
  SPLIT ch2 INTO C5
CHANNEL C1
  FROM ENV TO b11
  WITH s1;
CHANNEL C2
  FROM ENV TO b12
  WITH s2;
CHANNEL C3
  FROM b11 TO b12
  WITH s5, s6;
CHANNEL C4
  FROM b12 TO b13
  WITH s7, s8;
CHANNEL C5
  FROM b13 TO ENV
  WITH s3, s4;
    SIGNAL - - - -;
BLOCK b11;
- - - -
ENDBLOCK b11;
BLOCK b12;
- - - -
ENDBLOCK b12;
BLOCK b13;
- - - -
ENDBLOCK b13;
ENDSUBSTRUCTURE b1;
ENDSYSTEM s;
```

FIGURA D-147

Representaciones de un sistema en dos niveles mostrando el empleo de canales

Las definiciones de canal aparecen o bien a nivel de sistema (para la definición de canales entre los bloques que constituyen el sistema) o en la definición de la subestructura de bloque (para la definición de los canales entre sub-bloques). En GR, el diagrama que describe la división de un canal es el diagrama de subestructura de canales, lo que corresponde en PR a una descripción dentro de la definición de canal y asimismo a la definición de los bloques y canales que lo definen en el nivel inferior.

La figura D-148 da un ejemplo de un canal entre dos bloques visto a dos niveles diferentes (el segundo es inferior y más detallado).

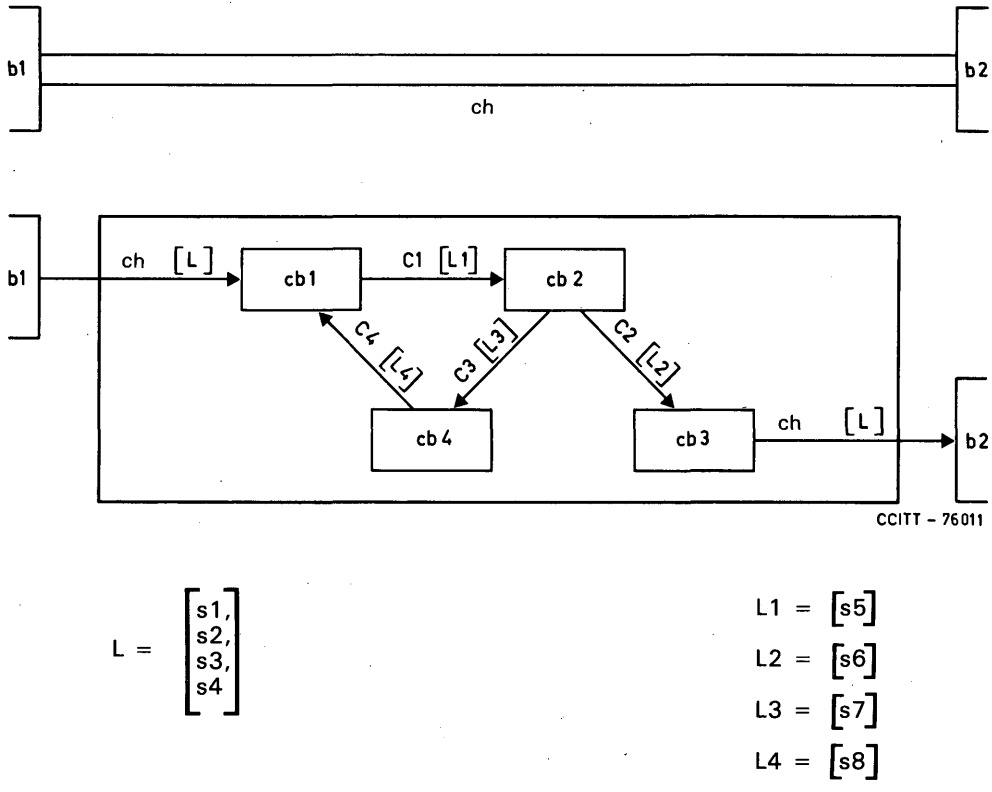


FIGURA D-148

Representación de un canal con bloques y canales

En la figura inferior de este ejemplo los bloques b1 y b2 no están a su vez divididos, pero el canal ch visto como un sistema en el nivel superior, muestra su estructura interna en la representación a nivel inferior. Es más adecuado mostrar la división del canal en el momento en que los bloques del sistema no se puedan dividir más, ya que de otro modo se tienen también que mostrar la interacción entre los bloques de los canales y, respectivamente, de los sub-bloques de los bloques. Esto aumenta la dificultad de representación de la estructura.

La figura D-149 muestra la representación PR de la división de canal de la figura D-148.

```
SYSTEM s;
- - - -
CHANNEL ch
  FROM b1 TO b2
  WITH s1, s2, s3, s4;
SUBSTRUCTURE ch;
BLOCKS : cb1, cb2, cb3, cb4;
CHANNELS : C1, C2, C3, C4;
- - - -
INCOMING ch to cb1;
OUTGOING ch FROM cb3;
CHANNEL C1
  FROM cb1 TO cb2
  WITH s5;
CHANNEL C2
  FROM cb2 TO cb3
  WITH s6;
CHANNEL C3
  FROM cb3 to cb4
  WITH s7;
CHANNEL C4
  FROM cb4 TO cb1
  WITH s8;
SIGNAL
- - - -
BLOCK cb1;
- - - -
ENDBLOCK cb1;
BLOCK cb2;
- - - -
ENDBLOCK cb2;
BLOCK cb3;
- - - -
ENDBLOCK cb3;
BLOCK cb4;
- - - -
ENDBLOCK cb4;
ENDSUBSTRUCTURE ch;
BLOCK b1;
- - - -
ENDBLOCK b1;
BLOCK b2;
- - - -
ENDBLOCK b2;
ENDSYSTEM s;
```

FIGURA D-149

Ejemplo de una partición de canal

En todos los ejemplos presentados se puede utilizar la notación macro para describir los bloques, los procesos y los canales de otra forma, si son demasiado largos o complejos. En el lugar en que se supone que hay que incluir la descripción, se requiere solamente una llamada de macro. La definición de macro contiene entonces esta información.

D.7.3.2 *Definición de señal*

Las señales se pueden definir a nivel de sistema, nivel de bloque, o en la parte interna de una definición de proceso. Las señales definidas a nivel de sistema representan señales intercambiadas con el entorno y entre bloques del sistema. Las señales definidas a nivel de bloque representan señales intercambiadas entre procesos del mismo bloque.

En la figura D-150, las señales SIG1 y SIG2 figuran enumeradas en un canal que conecta los bloques b1 y B2 o en un canal que conecta uno de estos bloques con el entorno. Las señales s1, s2 y s3 declaradas en el bloque 1 son utilizadas por los procesos pertenecientes a este bloque. Las señales s1, s2, s4 y s5 declaradas en el bloque B2 son diferentes de las declaradas en b1 y pueden ser utilizadas por los procesos del bloque B2.

En un sistema estructurado puede haber señales definidas a nivel de bloque que representen señal intercambiada entre bloques del nivel inferior (§ D.7.3.1). En la definición de canal puede haber la definición de las señales de los canales de nivel inferior que se hayan generado por partición del canal de que se trate (§ D.7.3.1). Los conjuntos de señales de los sub-canales se tienen que desglosar.

```
SYSTEM a;
    .
  SIGNAL SIG1(ty1, ty2), SIG2(ty2, ty3);
    .
BLOCK b1;
    .
    .
    .
   SIGNAL s1(t1,t2,t3),s2,s3(t4,t2);
    .
  ENDBLOCK b1;
  BLOCK B2;
    .
    .
    .
   SIGNAL s1(t1,t2,t3),s2,s4,s5(t4,t2);
    .
    .
  ENDBLOCK B2;
ENDSYSTEM a;
```

FIGURA D-150

Ejemplo de definiciones de señal a diversos niveles

D.7.3.3 *Definición de canal*

Los canales se pueden definir a nivel de sistema o, si el sistema está estructurado, también a nivel de bloque. Por último, el canal se puede definir dentro de una definición de canal. Los canales definidos a nivel de sistema representan los canales entre bloques de sistema y entre estos bloques y el entorno.

En la figura D-151 el bloque b1 utiliza los canales c1 y c2 para su comunicación con el entorno.

Si el sistema está estructurado, cada bloque puede contener la definición de los canales que permiten la comunicación entre los bloques generados por la partición del bloque de que se trate y entre estos bloques y el entorno (§ D.7.3.1).

En la definición de canal, puede haber definiciones de canal de los canales de nivel inferior obtenidos por división del canal de nivel inferior (§ D.7.3.1). La definición de canal contiene la lista de las señales de dicho canal que se tiene que definir en la sección de definición de señal.

```
SYSTEM a;
.
.
CHANNEL    c1 FROM b1 TO ENV
           WITH s1,s2,s3
           REFINEMENT CHANNELS: c1.1,c1.2;
CHANNEL    c2 FROM ENV TO b1
           WITH s4,s5
           REFINEMENT CHANNELS: c2.1,c2.2;
.
.
.
ENDSYSTEM a;
```

FIGURA D-151

Ejemplo de definiciones de canal

D.7.3.4 *Definición de datos*

De conformidad con la Recomendación Z.101, el LED tiene tipos de datos predefinidos que se ponen a la disposición de cada sistema. Estos tipos son: entero, real, carácter, cadena, booleano, identificador de instancia de proceso, tiempo, duración. No requieren una declaración y se pueden utilizar en una definición de variable con sus nombres predefinidos, a saber, ENTERO, REAL, CARÁCTER, CADENA, BOOLEANO, PID, TIEMPO, DURACIÓN.

La definición de variables se sitúa en una definición de proceso o de procedimiento. Toda variable que deba EXPLOTARSE, IMPORTARSE, REVELARSE o VERSE se tiene que declarar con estas características.

Cada dato es propiedad de una instancia de proceso. Los datos se definen en la definición de proceso (es decir, todas las instancias de proceso de un proceso tienen copia de la definición de datos dada a nivel de proceso).

En caso de revelación en el punto relativo a la declaración (DCL), sólo es necesario agregar la palabra clave REVELADO antes del nombre de tipo de variale.

La declaración VISTO se sitúa fuera del punto relativo a la DCL y se forma sintácticamente mediante la palabra clave VISTO seguida del nombre de variable, el tipo de dicha variable y, por último, el identificador del proceso que la revela.

En la figura D-152 se da un ejemplo de una declaración de «dígito» de variable VISTO y REVELADO en los procesos p1 y p2, respectivamente. Estos procesos pertenecen al bloque b1. En ese ejemplo, el proceso p1 es el propietario de la variable «dígito», que puede ser vista por el proceso p2.

```
SYSTEM a;
    .
    .
BLOCK b1;
    .
    .
  PROCESS p1;
    .
    .
  DCL
    REVEALED digit INT,
    counter, alarm number INT,
    a,b,c, BOOL;
      .
      .
      .
  ENDPROCESS p1;
  PROCESS p2;
  (  )
  DCL d,e,f, BOOL;
    VIEWED digit p1,
      .
      .
      .
    ENDPROCESS p2;
  ENDBLOCK b1;
ENDSYSTEM a;
```

FIGURA D-152

Ejemplo de visibilidad de datos entre procesos del mismo

En la figura D-153, todas las instancias del proceso b1 pueden ver la variable «*dígito*» en la forma en que se ha REVELADO y VISTO.

```
SYSTEM a;
      .
      .
      .
BLOCK b1;
      .
      .
PROCESS p1;
      .
      .
      .
DCL
      REVEALED digit INT;
      .
      .
VIEWED digit INT p1;
      .
      .
      .
END PROCESS p1;
      .
      .
      .
END BLOCK b1;
      .
      .
END SYSTEM a;
```

FIGURA D-153

Ejemplo de visibilidad entre instancias de proceso de la misma definición de proceso

Cuando hay que indicar que se requiere la operación de visión se utiliza la palabra clave VISIÓN seguida por el nombre de la variable que hay que ver y la identidad de la instancia de proceso a que pertenece la variable. El nombre de variable va separado de la identidad de instancia de proceso por una coma y los dos están encerrados dentro de paréntesis curvos. Si se omite la palabra clave VISIÓN la instancia de proceso busca una variable que tenga dicho nombre en sus datos locales (figura D-154).

```
SYSTEM s;
.......
  BLOCK b1;
    .......
    PROCESS p1;
    ......
    DCL
      REVEAL digit INT,
    .......
    ENDPROCESS p1;
    PROCESS p2;
    .......
    DCL
    t INT,
    VIEWED digit INT p1,
    .......
    TASK t:=5*VIEW(digit,p1);
      .......
    ENDPROCESS p2;
  ENDBLOCK b1;
ENDSYSTEM s;
```

FIGURA D-154

Ejemplo de utilización de una variable de visión

Las variables que se pueden exportar deben tener el atributo EXPORTADO en sus definiciones durante el proceso de exportación.

El proceso de importación declara las variables que tiene la intención de importar en un elemento importado utilizando la palabra clave IMPORTADO seguida por los nombres de las variables que hay que importar, sus tipos y el identificador del proceso de exportación.

Una variable se puede declarar como IMPORTADA y EXPORTADA al mismo tiempo. Esto permite a una instancia de proceso exportar una variable a las demás instancias del mismo proceso e importar dicha variable de las demás instancias. En este caso se tomarán precauciones para evitar colisiones de nombres: la asignación.

v:=IMPORT(v,Pid)

dará como resultado la sustitución de una v por la «v» de la instancia Pid.

Como puede verse en el ejemplo de la figura D-155, cada operación de importación requiere la palabra clave IMPORTACIÓN antes del nombre de la variable y el identificador de instancia de proceso, este último separado del anterior mediante una coma y ambos entre paréntesis curvos.

Si una variable se exporta y revela se tiene que agregar el atributo REVELADO al atributo EXPORTADO en la declaración (véase la figura D-155). Recuérdese que la revelación sólo es válida para las instancias de proceso del mismo bloque.

En el ejemplo de la figura D-155, la variable «contador» perteneciente al proceso p2 en el bloque b2 se exporta y revela. Se exporta para el proceso p1 y se revela para el proceso p3 en el mismo bloque b2.

```
SYSTEM a;
        .
        .
  BLOCK b1;
        .
        .
    PROCESS p1;
      DCL
    EXPORTED digit INT,
    IMPORTED counter INT BLOCK b2,p1;
        .
        .
    ENDPROCESS p1;
  ENDBLOCK b1;
  BLOCK b2;
        .
        .
    PROCESS p2;
      DCL
        EXPORTED,REVEALED counter INT,
        .
        .
        .
    ENDPROCESS p2;
    PROCESS p3;
        .
        .
      DCL
        VIEWED counter PROCESS p2,
        .
        .
        .
    ENDPROCESS p3;
  ENDBLOCK b2;
ENDSYSTEM a;
```

FIGURA D-155

Ejemplo de visibilidad de datos entre procesos pertenecientes a bloques diferentes

D.7.3.5 *Definición de macro*

La construcción macro es un medio para tratar las repeticiones. En programas PR se puede utilizar un macro para interrumpir el flujo de sentencias, en cuyo caso, se sustituyen éstas por una llamada de macro (§ D.7.3.7.8 y figura D-156 b)).

La definición de macro debe darse al comienzo de la definición de sistema, bloque, proceso o procedimiento (figura D-156 a)), según que pueda ser llamada desde todos los procesos, de los pertenecientes a un bloque, de un solo proceso o de un solo procedimiento. Hay que tener presente que en PR, el macro siempre tiene un acceso de entrada y un acceso de salida, de forma que es necesario utilizar etiquetas y uniones para representar el PR un diagrama GR con más de un acceso de entrada o acceso de salida, respectivamente.

El ejemplo de la figura D-156 representa un macro (figura D-156 a)) con dos accesos de salida a y b. Esto significa que en el programa principal (figura D-156 b)) hay dos etiquetas correspondientes a y b.

```
MACRO EXPANSION a;
  INPUT register_alarm;
        .
  JOIN a;
  INPUT tone;
        .
        .
  JOIN b;
ENDMACRO a;
```

a) Definición de macro

```
.....
.....
MACRO a;
a: .....
   .....
b: .....
   .....
```

b) LLamada de macro

FIGURA D-156

Ejemplo de utilización de un macro

D.7.3.6 *Definición de procedimiento*

Los procedimientos se pueden definir en varios niveles de la jerarquía de las construcciones LED, es decir, a nivel de sistema, bloque, proceso o procedimiento.

Según donde se defina el procedimiento, éste será visible a todos los procesos y procedimientos del sistema (definición a nivel de sistema), a todos los procesos y procedimientos de un bloque (definición a nivel de bloque), o sólo al proceso (procedimiento) dentro del cual se define.

La definición de parámetro formal se inicia con la palabra clave FPAR. Las variables enumeradas como parámetros formales pueden tener los atributos DENTRO, DENTRO/FUERA, o SEÑAL si el parámetro actual correspondiente es una señal.

Un parámetro con la palabra clave DENTRO se pasa por valor. Con la palabra clave DENTRO/FUERA, un parámetro se pasa por referencia (véase asimismo el § D.7.3.7.9). Un procedimiento tiene un solo acceso de salida, pero puede tener una o más sentencias RETORNO. Después de la interpretación de la palabra clave RETORNO la sentencia siguiente será la que sigue a la llamada de procedimiento en el programa principal (figura D-157).

El esquema de visibilidad es muy semejante al esquema de definición de tipo de datos. La sola excepción es que no hay procedimientos predefinidos de forma que todo procedimiento se tiene que definir de modo que pueda ser visto por el solicitante. La figura D-157 da un ejemplo de una definición de procedimiento. En la primera parte del ejemplo los parámetros formales son correctos y en la segunda sentencia de definición de procedimiento hay un error, puesto que el tipo de parámetros formales no figura en el orden correcto por comparación con los parámetros actuales de la llamada de procedimiento.

```
....
....
SIGNAL sig1(int, bool,int),
....
DCL number INTEGER,connected BOOLEAN;
....
PROCEDURE proc1
FPAR
      SIGNAL sig,
      in num INTEGER,
      IN/OUT conn BOOLEAN;
....
....
....
ENDPROCEDURE;
```

Esta sentencia de definición de procedimiento es correcta conforme a la llamada de procedimiento, pero

```
PROCEDURE proc1
FPAR
    IN num INTEGER,
    SIGNAL sig,
    IN/OUT conn BOOLEAN;
```

es falso!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

```
....
....
CALL proc1(sig1,number,conn);
....
....
```

FIGURA D-157

Ejemplos de utilización de procedimientos

D.7.3.7 *Definición de proceso*

La definición de proceso está incluida, como se ha visto antes, entre las sentencias PROCESO Y FINPROCESO. Un nombre tiene que asociarse a la palabra clave PROCESO. Este nombre (que tiene la calificación implícita de proceso) forma, junto con un nombre del bloque envolvente y el nombre del sistema, el identificador del proceso.

Si hay algún parámetro formal, éste tiene que declararse después de la sentencia de proceso. La palabra clave FPAR, como se ve en la figura D-158, precede los parámetros.

Los parámetros actuales se encuentran en la sentencia CREAR (véase el § D.7.3.7.1).

```
PROCESS nombre proceso;
FPAR
tipo de nombre de variable;
```

FIGURA D-158

Definición de parámetros formales

Los números de instancia, que deben incluirse entre paréntesis, son facultativos.

El número de instancia es un par de números enteros separados por una coma, el primero de los cuales indica el número de instancias de proceso que están presentes en la inicialización del sistema y el segundo señala el número máximo de instancias de proceso que pueden existir durante el ciclo de vida del sistema. Si se omite el primer número, hay una instancia en la inicialización del sistema. Si se omiten los números de instancia, hay una instancia de proceso en todo el ciclo de vida del sistema (figura D-159).

a) PROCESS p1(5,18);	hay cinco instancias en la inicialización y un máximo de 18 en el ciclo de vida del sistema
b) PROCESS p2(,5);	equivale a PROCESS p2(1,5);
c) PROCESS p3;	equivale a PROCESS p3(1,1);
d) error de proceso (5,3);FALSO!!!	

FIGURA D-159

Ejemplos de procesos

Es evidente que el número máximo de instancias (segundo número) tiene que ser igual o mayor que el número de instancias existentes en la inicialización del sistema (figura D-159 d)).

Tras la sentencia PROCESO, se tiene que definir los datos poseídos por el proceso y después los procedimientos y macros locales de ese proceso (véanse los § D.7.3.5 y D.7.3.6).

La representación del comportamiento comienza después de estas definiciones. Esto se indica por una sentencia ARRANQUE seguida por una cadena de transición o por una sentencia ESTADO.

Si no hay sentencia ARRANQUE, cada instancia de PROCESO comienza a existir en la primera sentencia ESTADO (figura D-160 b)).

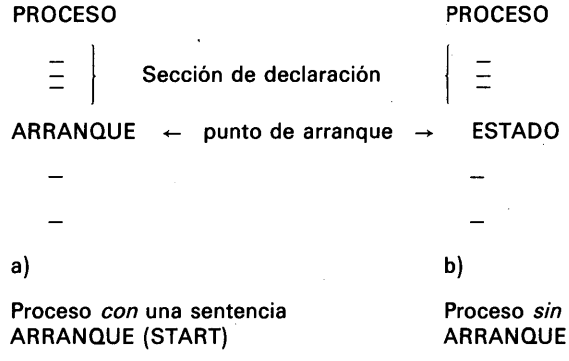


FIGURA D-160

Arranque de un proceso

D.7.3.7.1 *Creación de procesos*

Una instancia de proceso puede crear otras instancias del mismo proceso o de proceso diferentes del mismo bloque, emitiendo una acción de creación.

La sentencia CREAR puede contener la lista de los parámetros actuales dentro de paréntesis curvos, como se muestra en la figura D-161.

```
SYSTEM s;
  BLOCK b;
    PROCESS p;
    .......
    .......
    DCL    a,c   INTEGER,
           b     BOOLEAN,
    .......
    .......
      CREATE p1(a,b,c);
    .......
    ENDPROCESS p;
    .......
    PROCESS p1;
    FPAR
    .......
      digit INTEGER, connected BOOLEAN, number INTEGER;
    START
    .......
    ENDPROCESS p1;
  ENDBLOCK b;
ENDSYSTEM s;
```

FIGURA D-161

Ejemplo de creación de una instancia de proceso

D.7.3.7.2 *Estados y apariciones múltiples*

En PR, un estado se representa por la palabra clave ESTADO seguida del nombre de estado.

Un estado termina en la sentencia de estado siguiente (sentencia de ESTADO siguiente) o al final del proceso (sentencia FINPROCESO o PARADA).

Además, en PR hay siempre sentencias distintas para indicar que se entra en un estado (SIGUIENTE ESTADO ⬭) o que se sale de un estado (ESTADO ⬭)

No existe una sentencia común que indique ambos casos como en el GR.

Las múltiples apariciones de estados tienen la misma explicación y relación con el modelo semántico indicadas en el capítulo D6 (GR).

Figuran a continuación ejemplos de la versión PR de los ejemplos correspondientes al GR que figuran en el § D.6.3.6.5.

```
b:      NEXTSTATE state_1;
      STATE State_1;
        INPUT A;
        NEXTSTATE state_2;
        INPUT C;
        JOIN a;
      STATE State_2;
        INPUT C;
        JOIN b;
      INPUT B;
a:    NEXTSTATE State_3;

      STATE State_3;
        INPUT D;
        JOIN b;
```

a) Diagrama completo

```
      -
      -
      STATE State_1;
      INPUT A;
      NEXTSTATE State_2;
      INPUT C;
      NEXTSTATE State_3;
    STATE State_2;
      INPUT B;
      NEXTSTATE State_2;
      INPUT C;
      NEXTSTATE State_1;
    STATE State_3;
      INPUT D;
      NEXTSTATE State_1;
```

b) Diagrama a) con los estados principales y con los siguientes estados utilizados como conectores con el estado principal

FIGURA D-162

Ejemplo de apariciones múltiples de un estado (equivalente a la figura D-93)

D.7.3.7.3 *Sentencia PR y utilización de datos*

D.7.3.7.3.1 *Sentencia de entrada*

La sentencia de ENTRADA contiene una lista de señales. Los items de datos contenidos en las señales son nombrados utilizando identificadores de variable. Los identificadores de variable deben ser del tipo indicado en la definición de señal, de modo que su posición es muy importante. Estos identificadores de variable están encerrados dentro de paréntesis curvos y separados por comas. Si se rechazan uno o más items de datos, se omiten las variables correspondientes y esto se representa mediante dos comas consecutivas (figura D-163).

```
INPUT a (var1,var2,,var4);
```

Observación – En esta sentencia, el tercer elemento de datos de la señal a se descarta.

FIGURA D-163

Señal a como entrada con sólo tres de sus cuatro elementos de datos definidos

```
SIGNAL sig1 (INTEGER,BOOLEAN,INTEGER);

    DCL a INTEGER,b BOOLEAN,c INTEGER,
```

a) declaraciones

```
    INPUT sig1(a,b,c);
```

b) una entrada correcta

```
    INPUT sig1(a,c,b);
```

c) una entrada incorrecta

FIGURA D-164

Sentencias de ENTRADA

D.7.3.7.3.2 *Sentencia de conservación*

La sentencia de CONSERVACIÓN puede tener la lista de nombres de señal o un asterisco, si todas las señales entrantes no nombradas en sentencias de ENTRADA se conservan. La figura D-165 da un sencillo ejemplo de la utilización de CONSERVACIÓN.

```
–
–
–
STATE State_31;
  SAVE S;                          /*S' llega, entra en la cola y permanece en la cola, R llega
                                   y es consumida inmediatamente. Se produce la
                                   transición al estado_32.

  INPUT R;

  NEXTSTATE State_32;
STATE State_32;                    Al llegar al estado_32, 'S' es consumida inmediatamente
                                   y se produce la transición al estado siguiente. /*
  INPUT S;
–
–
–
```

FIGURA D-165

Ejemplo de la utilización de una CONSERVACIÓN

D.7.3.7.3.3 *Sentencia de salida*

La sentencia de SALIDA contiene la lista de señales enviadas a otros procesos. Para cada item de datos contenido en la señal, puede haber una expresión o valor actual correspondiente. Si un valor de uno o más items de datos es indefinido, se omite, y la posición vacía se representa mediante dos comas consecutivas (figura D-166).

```
OUTPUT sig1(a,,c);
```

FIGURA D-166

Señal sig1 como salida con solamente dos de sus tres elementos de datos definidos

La lista de los valores actuales se pone entre paréntesis curvos. El valor de cada item de datos considerado en la señal tiene que ser del tipo definido. Con referencia a las declaraciones de la figura D-164 a) para la sentencia de entrada, se muestran en la figura D-167 una salida correcta y una salida incorrecta.

```
    OUTPUT sig1(2,true,10)
a)  salida correcta
    OUTPUT sig1(false,true,10)
b)  salida incorrecta
```

FIGURA D-167

Sentencias de salida

D.7.3.7.3.4 *Sentencia de tarea*

La sentencia de tarea puede contener una o varias sentencias de asignación, conjunto de sentencias de texto y/o texto informal. Un texto informal consiste en un nombre y/o una frase delimitada por apóstrofos. Las sentencias o texto van separadas por comas (figura D-168).

```
TASK a:=b;
TASK 'connect the subscriber';
TASK RESET(TIME);
TASK main_assignement, c:=d+e
TASK var1:=var2*var3,
     var4:=var5 MOD var6;
```

FIGURA D-168

Sentencias de tarea

D.7.3.7.3.5 *Sentencia de DECISIÓN*

La decisión en el LED/PR se representa por la palabra clave DECISIÓN seguida por el nombre de la decisión y/o una expresión o cadena de texto. Esta última es un texto (formal o informal) contenido entre ' ' (apóstrofos).

El conjunto de trayectos de salida está delimitado por la sentencia DECISIÓN al comienzo y la sentencia FINDECISIÓN al final. (Véase la figura D-169.)

```
DECISION ....;
(result): ....;
(result): ....;
(result): ,,,,;
  ENDDECISION;
```

FIGURA D-169

Delimitación de una decisión

Los resultados de una decisión se indican dentro de paréntesis curvos, y se representan por una o más cadenas de texto delimitadas por apóstrofos, o por valores posibles obtenidos por la evaluación de la expresión contenida en la sentencia de decisión. Los diferentes resultados dentro de los paréntesis van separados por comas. Los valores se representan por expresiones constantes o por gamas cuyos límites superior e inferior son expresiones constantes. Los valores resultantes tienen que ser de tipo de la expresión contenida en la sentencia de decisión, como se ilustra en los ejemplos de las figuras D-170, D-171 y D-172.

Es posible indicar específicamente algunos resultados y agrupar todos los demás resultados posibles utilizando la palabra clave ELSE, como ilustra, por ejemplo, la figura D-170.

```
DECISION 'x';
(2):          .  | trayecto 1
              .  |
(6):          .  | trayecto 2
              .  |
(9,10):       .  | trayecto 3
              .  |
ELSE:         .  | trayecto 4
              .  |
ENDDECISION;
```

Observación – x es un número entero comprendido entre 1 y 10

El trayecto 1 se selecciona con x=2

El trayecto 2 se selecciona con x=6

El trayecto 3 se selecciona con x=9 ó x=10

El trayecto 4 se selecciona con x=1, =3, =4, =5, =7 ó =8.

FIGURA D-170

Decisión con ELSE

```
                    DECISIÓN 'Categoría de abonado'
('internacional', 'nacional'):  :  | trayecto 1
                                :  |
('local')                   :   :  | trayecto 2
                                :  |
```

FIGURA D-171

Ejemplo de DECISION

Ejemplo

```
DCL x INT,
  DECISION x;
(2+5)          :.....;
(6+8, 10+12)   :.....;
ELSE           :.....;
  ENDDECISION;
  DECISION a;
(true) :.....;
(false):.....;
  ENDDECISION;
  DECISION x;
    (10) :.....;
ELSE   :.....;
  ENDDECISION;
```

FIGURA D-172

Ejemplos de los resultados de decisiones

La sentencia FINDECISIÓN ofrece capacidades de estructuración idénticas a la programación estructurada, como se ilustra en la figura D-173.

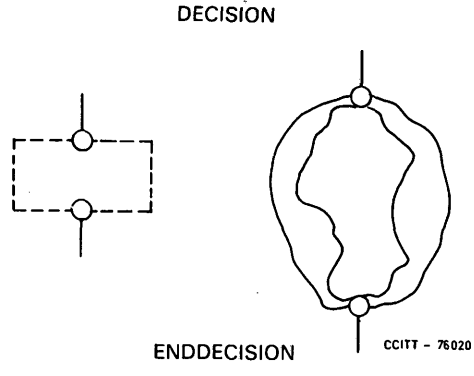


FIGURA D-173
Capacidades de estructuración en la DECISIÓN

Todos los trayectos terminan en la sentencia FINDECISIÓN. Los trayectos que no se han cerrado previamente continúan en la sentencia que sigue a FINDECISIÓN, como se ilustra en las figuras equivalentes D-174 y D-175.

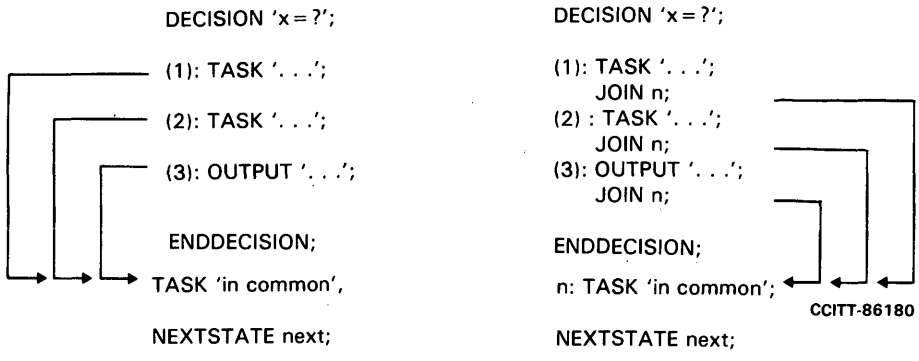


FIGURA D-174

Bifurcación de una DECISIÓN

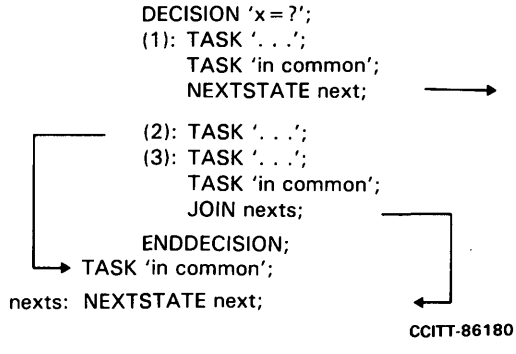


FIGURA D-175

Bifurcación equivalente del ejemplo ilustrado en el figura D-174

La sentencia de decisión se puede utilizar para modelar la estructura SI ENTONCES, la estructura HACER MIENTRAS y la estructura BUCLE HASTA.

La estructura SI ENTONCES se muestra en la figura D-176, la estructura SI ENTONCES OTRO en la figura D-177, la estructura HACER MIENTRAS en la figura D-178, y la estructura BUCLE HASTA en la figura D-179.

```
BLOCK a;
  PROCESS b;
    STATE s1;
      INPUT i1;
      DECISION 'i1 = ?'
(0): TASK t1;
ELSE:;
      ENDDECISION;
  ENDPROCESS b;
ENDBLOCK a;
```

FIGURA D-176

Estructura IF THEN

```
BLOCK a;
  PROCESS b;
    STATE s1;
      INPUT i1;
      DECISION 'i1=?'
(0): TASK t1;
ELSE: TASK t2;
      ENDDECISION;
  ENDPROCESS b;
ENDBLOCK a;
```

FIGURA D-177

Estructura IF THEN ELSE

```
BLOCK b1;
  PROCESS p1;
    STATE s1;
      INPUT i1;
1:    DECISION 'b=?';
      (false): JOIN 11;
      ELSE : ;
      ENDDECISION;
      TASK t1;
      JOIN 1;
11:    -
       -
       -
  ENDPROCESS p1;
ENDBLOCK b1;
```

FIGURA D-178

Estructura DO WHILE

```
BLOCK b1;
  PROCESS p1;
    STATE s1;
      INPUT i1;
1   DECISION 'b=?';
      (false): JOIN 1;
      ELSE : ;
      TASK t2;
      ENDDECISION;
  ENDPROCESS p1;
ENDBLOCK b1;
```

FIGURA D-179

Estructura LOOP...UNTIL...

D.7.3.7.3.6 *Sentencia de alternativa*

Cuando se especifica un sistema, pueden darse situaciones en las que cualquiera de varios comportamientos satisface nuestras necesidades. En este caso se tienen que indicar algunos comportamientos admisibles, dejando la elección de uno de ellos al realizador.

Las distintas posibilidades se indican por la palabra clave ALTERNATIVA con una expresión o texto informal asociado.

El conjunto de posibles comportamientos se cierra con la sentencia FINALTERNATIVA; cada alternativa se identifica por un nombre.

La estructura global es igual a la de DECISIÓN; la diferencia reside en la semántica. Mientras que en el caso de la DECISIÓN se elige una de las bifurcaciones conforme al valor de un item de datos que cambia en la duración de vida útil del proceso, en la alternativa sólo existe una bifurcación en el proceso de realización, de modo que se elige antes de la realización. La figura D-180 da un ejemplo.

```
ALTERNATIVE 'Alarm response';
  ('first choice') : OUTPUT ring_bell;
  ('second choice'): OUTPUT light_alarm;
ENDALTERNATIVE;
```

En la realización se podría tener:

ya sea

```
OUTPUT ring_bell;
```

o bien

```
OUTPUT light_alarm;
```

FIGURA D-180

Ejemplo de sentencia de ALTERNATIVA

Los resultados de una sentencia de ALTERNATIVA tienen la misma sintaxis que la decisión. La figura D-181 da un ejemplo de lo expuesto.

```
DCL a INTEGER 1:10;
ALTERNATIVE a;
(1,4):.....;
(5,10):.....;
ENDALTERNATIVE;
```

FIGURA D-181

Ejemplo de la sentencia ALTERNATIVA

D.7.3.7.4 *Pictogramas de estado*

No tienen correspondencia en PR.

D.7.3.7.5 *Tiempo*

El concepto de tiempo se utiliza en las sentencias ESTABLECER y REINICIALIZAR (SET y RESET). Pueden ser el texto formal en una sentencia de TAREA PR (véase el § D.7.3.7.3.4).

D.7.3.7.6 *Condición habilitadora*

En PR, la condición habilitadora se representa por la palabra clave SUMINISTRADO seguida por la condición que hay que evaluar. Esta palabra clave va asociada a la sentencia ENTRADA. La condición asociada a la ENTRADA es una expresión booleana. Si la condición es verdadera se habilita la transición. Si es falsa, la señal se conserva. Teniendo en cuenta que una entrada sólo puede aparecer una vez por estado, no se permite la posibilidad de que una entrada aparezca dos veces a partir del mismo estado, cada una de ellas con una condición habilitadora diferente.

Las variables utilizadas en la expresión asociada a la condición habilitadora pueden ser locales o importadas. Tampoco pueden ser vistas.

La figura D-182 da un ejemplo de la utilización de condiciones habilitadoras.

```
SYSTEM s;
  BLOCK b1;
  ........
    PROCESS p;
    ........
    DCL
      connected BOOLEAN,
    IMPORTED alarm BLOCK b2,p2;
    ........
    STATE s1;
      INPUT i1 PROVIDED connected;
      ........
      INPUT i2, PROVIDED IMPORT alarm,p2;
      ........
    ENDPROCESS p;
  ENDBLOCK b1;
  BLOCK b2;
  ........;
    PROCESS p2;
    ........
    DCL
      EXPORTED alarm BOOLEAN;
      .......
  ENDPROCESS p2;
  ........
  ENDBLOCK b2;
ENDSYSTEM s;
```

FIGURA D-182

Ejemplos de condiciones habilitadoras

D.7.3.7.7 *Señales continuas*

En PR, las señales continuas se representan por la palabra clave SUMINISTRADO seguida por la condición que hay que evaluar. Las señales continuas tienen una sintaxis diferente de las condiciones habilitantes, en el sentido de que no llevan asociada una sentencia de entrada. La condición es una expresión booleana en datos visibles a dicho proceso. Si, en el momento de la evaluación, la condición es verdadera se activa la transición. Si hay varias señales continuas, se tiene que indicar el orden de evaluación. Esto se efectúa asociando una prioridad a cada condición utilizando para ello la palabra clave: PRIORIDAD, seguida por un valor entero. Se da a las señales continuas una prioridad inferior a la de las señales de entrada usuales. La condición que hay que evaluar puede contener solamente variables locales o importadas. No puede contener ninguna variable «vista» (figuras D-184 y D-183).

```
SYSTEM s;
  ........
  BLOCK b1;
    ........
    PROCESS p1;
      ........
      DCL
        x  INT;
      ........
      STATE s1;
      PROVIDED x=2 PRIORITY 0
      ........
      PROVIDED x=5 PRIORITY 1
      ........
      IMPUT i1;
      ........
    ENDPROCESS p1;
  ENDBLOCK b1;
ENDSYSTEM s;
```

FIGURA D-183

Ejemplo de señales continuas para variables locales

```
SYSTEM s;
  ........
  BLOCK b1;
    ........
    PROCESS p1;
    ........
    DCL
      connected BOOL,
    IMPORTED alarm BLOCK b2,p2;
    ........
    STATE s1;
    PROVIDED connected PRIORITY 0;
    ........
    INPUT i1;
    ........
    PROVIDED IMPORT (alarm,p2) PRIORITY 1;
    ........
    ENDPROCESS p1;
  ENDBLOCK b1;
  BLOCK b2;
    ........
    PROCESS p2;
    ........
    DCL
      EXPORTED alarm BOOL;
    ........
    ENDPROCESS p2;
    ........
    ENDBLOCK b2;
    ........
ENDSYSTEM s;
```

FIGURA D-184

Ejemplo de señales continuas para variables importantes

D.7.3.7.8 *Llamada de MACRO*

La sentencia MACRO es: MACRO nombre.

El nombre asociado a la palabra clave MACRO indica una definición de MACRO con ese nombre (véase el § D.7.3.5). La sentencia MACRO se puede insertar en cualquier parte de una representación PR, pero *después* de la correspondiente definición de MACRO.

Para interpretar la representación, la definición de MACRO debe sustituir a cualquier aparición de la sentencia de llamada de MACRO, como se muestra en el ejemplo de la figura D-185.

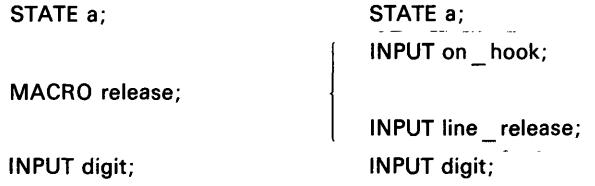


FIGURA D-185

Interpretación de una sentencia de llamada de MACRO

D.7.3.7.9 *Llamada de PROCEDIMIENTO*

La llamada de procedimiento contiene una lista de parámetros actuales para el procedimiento. Consiste en señales de entrada y de salida visibles desde el procedimiento. *Se observará que todas las señales visibles para el solicitante y no declaradas como parámetros actuales en la llamada de procedimiento, se conservan automáticamente durante toda la vida útil del procedimiento.* Además, los parámetros actuales contienen los datos visibles al solicitante que también pueden ser vistos por el procedimiento.

Se observará asimismo que la declaración DENTRO, o DENTRO/FUERA, se efectúa en la definición de procedimiento, de modo que la sentencia de llamada no tiene que repetirla. Se tendrá también en cuenta que los parámetros con el atributo DENTRO/FUERA en la definición de procedimiento corresponden a variables que pertenecen directa o indirectamente (cuando el solicitante es un procedimiento) al solicitante. Los datos mencionados en los parámetros actuales se asocian, en su orden, con los parámetros formales. Si un parámetro no figura, este particular se tiene que indicar mediante dos comas consecutivas. En este caso, el parámetro formal correspondiente tiene el valor «indefinido». La figura D-186 da un ejemplo.

```
CALL proc1(sig1,,conn);
```

(véase un ejemplo en el § D.7.3.6, figura D-157).

FIGURA D-186

Interpretación de una llamada a un procedimiento

D.7.3.7.10 *Etiquetas (conectores)*

Las etiquetas se utilizan como puntos de entrada asociados a sentencias. Esto permite transferir el control por medio de una sentencia UNIÓN (figura D-187).

FIGURA D-187

Etiqueta

No es posible transferir el control (y, por consiguiente, asociar etiquetas) a los tipos de sentencias mostrados en la figura D-188.

SYSTEM,	ENDSYSTEM,
BLOCK,	ENDBLOCK,
PROCESS,	ENDPROCESS,
STATE,	ENDDECISION,
INPUT,	SAVE,
ENDMACRO,	ENDPROCEDURE,
ENDALTERNATIVE	

FIGURA D-188

Puntos de etiqueta no admisibles

Sólo puede asociarse una etiqueta a una sentencia. Las etiquetas son siempre *locales* con respecto a un proceso. No es posible transferir el control de un proceso a otro por medio de una etiqueta.

D.7.3.7.11 *Asequibilidad de la sentencia*

A no ser que se haya indicado una transferencia de control, las sentencias se interpretan una por una.

En las figuras D-189 a D-193 se muestran sentencias que efectúan la transferencia de control.

STATE: se transfiere el control a la llegada de una señal a la sentencia INPUT o SAVE que contiene dicho nombre de señal.

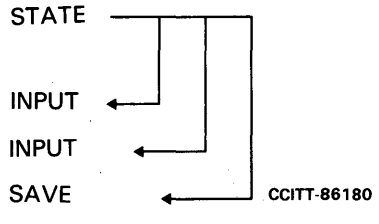


FIGURA D-189
Transferencia de control en STATE

JOIN: Se transfiere el control a la sentencia que tiene la etiqueta nombrada en la JOIN; tiene que haber una, y solo una, sentencia que tenga esta etiqueta.

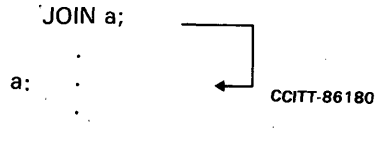


FIGURA D-190
Transferencia de control en JOIN

NEXTSTATE: Se transfiere el control al estado que tiene el nombre indicado en la sentencia SIGUIENTEESTADO.

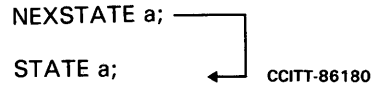


FIGURA D-191
Transferencia de control en NEXTSTATE

STOP: La interpretación termina; no hay transferencia de control.

```
STOP;
```

FIGURA D-192
Sin transferencia de control

Transferencia implícita: se efectúa de forma igual a la previamente explicada para las bifurcaciones de decisión.

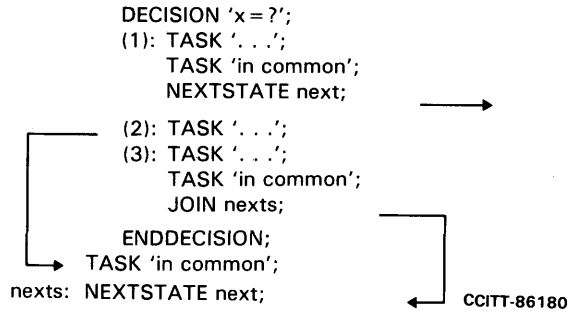


FIGURA D-193
Ejemplo de transferencia implícita de control

D.7.3.7.12 *Uso de divergencia y convergencia*

En PR, este punto va incluido en el texto correspondiente a las etiquetas, cuya definición figura en el § D.7.3.7.10.

D.7.3.7.13 *Comentarios*

Los comentarios se insertan por medio de la palabra clave: COMMENT.

La palabra clave puede insertarse, como una sentencia, siempre que se pueda introducir una sentencia TASK. Además, la palabra clave se puede asociar con cualquier otra sentencia al final de la sentencia:

```
STATE a COMMENT '. . .';
```

Con la excepción de las sentencias END, a las que no se les puede asociar el comentario.

Se puede agregar un comentario CHILL (/*. . . .*/) siempre que existe un espacio (intervalo). Las figuras D-194 y D-195 dan ejemplos de la utilización de este tipo de comentarios.

```
DCL
a INTEGER, /* explicación del uso de la variable a */
b BOOLEAN; /* explicación del uso de la variable b */
```

FIGURA D-194

Uso de comentarios CHILL en sentencias de variable

```
PROCEDURE ABC (IN a INTEGER, /* significado de este parámetro formal */,
IN/OUT b BOOLEAN; /* significado de este parámetro formal */;
```

FIGURA D-195

Utilización de comentarios CHILL en la definición de procedimiento

D.7.3.7.14 *Notaciones estenográficas*

Puede insertarse un * (asterisco) en una sentencia ESTADO/ENTRADA/CONSERVACIÓN para indicar que todos los nombres aplicables a dicha declaración son adecuados.

Ejemplo:

a) STATE *; b) STATE A,B,C,D,G;

Observación – a) es equivalente a b), si A,B,C,D,G es el conjunto de estados definidos en dicho proceso.

FIGURA D-196

Notación ALL-states

a) INPUT *; b) INPUT I1, I3, I6;

Observación – a) es equivalente a b), en donde I1, I3 e I6 son señales que entran en dicho proceso y *no declaradas como ENTRADA o CONSERVACIÓN en dicho estado.* Sólo puede asociarse a un estado determinado una sola ENTRADA o CONSERVACIÓN con un *.

FIGURA D-197

Notación ALL-incoming-signals

a) SAVE *; b) SAVE I1, I3, I6;

Observación – a) es equivalente a b), en donde I1, I3 e I6 son señales de entrada en dicho proceso y *no declaradas como entrada en ese estado*. No puede asociarse otra CONSERVACIÓN al mismo estado.

FIGURA D-198

Notación ALL-saves

A un ESTADO *; se puede asociar una lista de excepciones para excluir algunos estados de la lista. Los nombres de los estados que han de excluirse aparecen entre [ ] (corchetes).

a) STATE * [A,C,D]; b) STATE B, E, F, G;

Observación – a) y b) son equivalentes, si A,B,C,D,E,F,G son estados del proceso

FIGURA D-199

Notación ALL-but-some state

Puede insertarse un guión (–) en una sentencia NEXTSTATE para indicar que el estado del cual procede la transición es el mismo en el que termina la transición (figura D-200).

a)
```
STATE a;
INPUT i;

NEXTSTATE -;
```

b)
```
STATE a;
INPUT i;

NEXTSTATE a;
```

Observación – a) y b) son equivalentes

FIGURA D-200

Estados de origen y de terminación idénticos

Esta notación es particularmente útil en los casos en que se desea realizar un conjunto común de acciones en varios estados que quedan después en el mismo estado, como se muestra en la figura D-201.

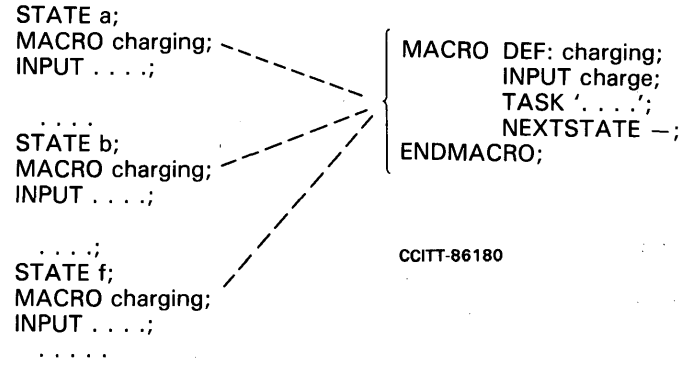


FIGURA D-201
Definición de MACRO

D.8 *Mapeados*

En este punto se describen algunos aspectos de la correspondencia entre LED y CHILL (§ D.8.1), y la correspondencia entre el LED/GR y LED/PR (§ D.8.2).

D.8.1 *Mapeado entre LED y CHILL*

En este punto se ilustran algunas formas posibles de mapeado entre LED y CHILL. Presenta este texto a título de ejemplo y no se pretende que sea exhaustivo ni tampoco se sugiere que sea alguna de estas formas la que tenga que utilizarse en la práctica.

De hecho, el mapeado debiera no solamente considerar el compilador CHILL disponible y el dispositivo objetivo, ya que en general el mapeado es una actividad intelectual muy compleja y sólo a través de la experiencia adquirida los diseñadores/programadores pueden decidir sobre una estructura de programación CHILL particular que hay que utilizar para realizar una determinada representación LED. Lo anterior se aplica asimismo a la representación en LED de las funciones realizadas por un programa CHILL. Un mapeado de uno a uno (de ser realizable) no es necesariamente el mejor método de utilización del LED para representar las funciones realizadas en CHILL.

En este procedimiento, la estructura global de un mapeado entre un diagrama LED completo y un programa CHILL (incompleto) se muestra en la figura D-202.

Las figuras D-203 a D-206 dan algunos ejemplos de mapeado entre construcciones de los dos lenguajes. Se refieren a las siguientes construcciones LED:

* estado y recepción/conservación de señales; selección de un siguienteestado;
* salida;
* unión;
* decisión.

El módulo declarante contiene la definición y la declaración de todas las señales utilizadas en el diagrama LED transformado y todas las variables asociadas con estas señales. Todas estas variables son adjudicadas al módulo que representa el bloque funcional del diagrama LED.

```
Declaring: MODULE
        /*  módulo CHILL que contiene las señales y variables asociadas, utilizando el diagrama LED */
            GRANT
        /*  adjudicación de señales y variables                                                      */
            SIGNALS
        /*  definición de señales                                                                    */
            SYNMODE (OR NEWMODE)
        /*  definición de tipo                                                                       */
END Declaring;

Bloque funcional: MODULE
        /*  módulo que contiene la parte procedimiento del diagrama LED                              */
            SEIZE
        */  toma de todas las señales y variables que pueden recibirse y transmitirse desde (hacia) este
            bloque funcional                                                                         */
        */  definición y declaración de datos. Estos datos son globales a todos los procesos pertene-
            cientes a este módulo                                                                    */
Process name: PROCESS ( );
        */  local data definition and declaration                                                    */
            nextstate:=....;
            join:=none;
            PARA SIEMPRE;
                state_loop:CASE nextstate OF
        */  bucle en la variable nextstate indicando el estado SDL                                  */
                                    (state_labell):RECEIVE CASE
                                    (signal namel):
                                            .
                                            .
                                            .
                                    (signal namen):
            ESAC state_ loop;
            DO WHILLE join/=none;
                    CASE join OF
                            (join_labl): join: = none;
                                            .
                                        .   .
                                            .
                            (join_labm): join: = none;
                                            .
                                            .
                        ESAC;
                    OD;
                OD
            END process_name;
END Functional_Block;
```

FIGURA D-202

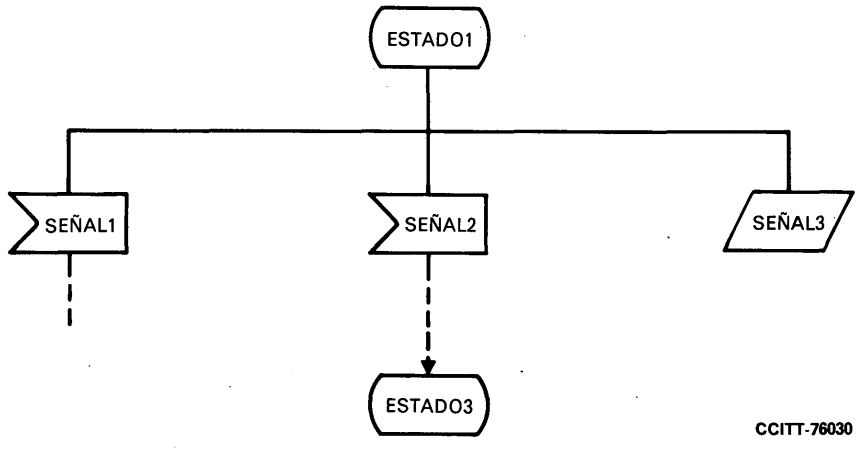
Ejemplo de mapeado uno a uno del LED al CHILL

El módulo de bloque funcional representa el comportamiento (parte procedimiento) de los procesos LED.

En este esquema de traducción, cada proceso LED se representa como un bucle infinito, una variable nombrada «siguienteestado» indica el estado que hay que examinar y una variable nombrada «unión» indica posibles puntos de unión que determinan conjuntos comunes de sentencias.

Se efectúa una selección del valor del siguiente estado, por medio de la construcción «caso» de CHILL; cada anotación del caso identifica un estado LED. En cada anotación se efectúa una selección entre las posibles señales de entrada. Cada señal de entrada determina el conjunto de acciones que hay que realizar (el «trayecto de transición»).

Cada trayecto de transición termina en una asignación relativa a la variable «siguienteestado», que determina directamente el siguiente estado que hay que examinar, o en la variable «unión». Un bucle de selección subsiguiente en el valor corriente de la variable «unión» permite terminar cada transición, en un sentido LED y finalmente asignar un valor a la variable «siguienteestado».



a) LED

```
STATE1:
   RECEIVE CASE
      (SIGNAL1): .................
                 .................
      (SIGNAL2): .................
                       NEXTSTATE: = STATE3;
      ELSE GETOUT (LIST 1)
ESAC STATE 1;
```

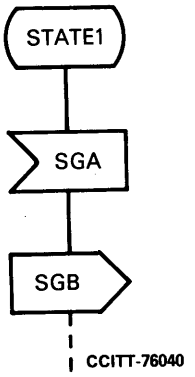
b) CHILL

FIGURA D-203

Ejemplos de mapeado STATE/INPUT/SAVE/NEXTSTATE

Uno de los principales problemas para relacionar el LED con el CHILL es la diferente semántica en la recepción de señales; de hecho, en tanto que el CHILL no consume (no consume y, por consiguiente, no destruye) ninguna señal a no ser que se hayan recibido (señales persistentes) el proceso LED consume (y, por consiguiente, destruye) todas las señales recibidas hasta llegar a una concordancia con una de las entradas enumeradas para dicho estado. La discrepancia semántica se ha resuelto introduciendo la rutina incorporada GETOUT, como alternativa (trayecto ELSE) en la construcción CHILL RECEIVE CASE como se muestra en la figura D-203. La rutina GETOUT, incorporada en el CHILL que conoce (por parámetro) la lista de señales de entrada y de conservación, destruye las demás señales puestas a disposición del proceso cuando es llamada.

Después de ejecutar la rutina GETOUT el selector de estado se inicializa a fin de repetir el bucle correspondiente a dicho estado hasta que se selecciona una señal de entrada válida (o llega si no está ya presente).



a) LED

```
STATE1:
   RECEIVE CASE
      (SGA): pi:=get_instance_value():
             send SGB to pi;
             nextstate:=.....;
      ELSE   nextstate:=state1;
ESAC STATE1;
```

b) CHILL

FIGURA D-204

Ejemplo de mapeado OUTPUT

Por ejemplo, una vez que se ha reconocido la señal de entrada SGA en la figura D-204, se selecciona la instancia de proceso de destino adecuada para la señal SGB y se transmite la señal SGB.

Antes de transmitir la señal SGB puede ser necesario rellenar algunos campos de información que la señal debe transportar. Esto puede efectuarse inmediatamente antes del envío de la señal o por anticipado.

Cuando se encuentra un punto de unión en el diagrama (véase la figura D-205), se asigna el valor adecuado a la variable «UNIÓN» (JOIN). Como se explica en la figura D-202, se realiza un bucle en el valor de la variable «unión» para determinar el estado siguiente que hay que examinar. Un punto de unión puede ser visto, desde el punto de vista del lenguaje de programación como una construcción «goto»; reuniendo todos los puntos de unión de forma que puedan examinarse, lo que permite escribir la estructura completa del programa sin utilizar ningún «goto», lo que facilita la lectura.

Una decisión LED tiene traducción directa a la construcción de caso del CHILL, como se muestra en la figura D-206.

D.8.2 *Mapeado entre GR y PR*

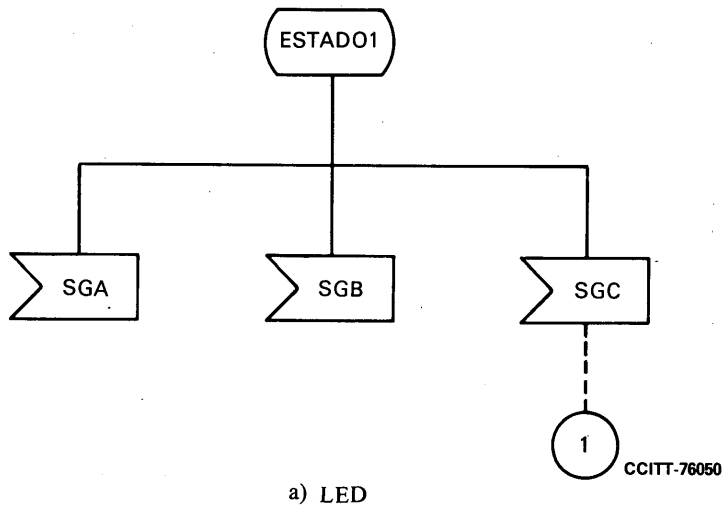
Algunas partes de un sistema sólo se pueden describir utilizando PR; por ejemplo, la definición de datos y la definición de señal. Por consiguiente, puede ser necesario completar los sistemas representados utilizando GR con algunas sintaxis PR. Esto significa que los sistemas representados utilizando GR pueden siempre mapearse con el PR, pero que una representación PR puede no poderse representar completamente en GR.

La figura D-207 muestra la parte interna de un bloque B dado por un diagrama de interacción de bloques y sus partes correspondientes en PR.

En la figura D-208 se muestra un ejemplo de una definición de proceso simple para el proceso PR 1. Se muestra tanto en forma GR como en forma PR. Sólo se muestran las partes que tienen una representación gráfica.

Para el diagrama de interacción de bloques, las construcciones GR tienen la correlación con las construcciones PR que se muestra en la figura D-209.

La figura D-210 muestra la correlación correspondiente para el diagrama de procesos.



a) LED

```
STATE1:
   RECEIVE CASE
      (S1 in m): case m. id of
                         (SGA):.....;
                         (SGB):.....;
                         (SGC):.....; JOIN:=1;
                         ELSE nextstate:=state1
                 esac;
ESAC STATE1;
```

b) CHILL

FIGURA D-205

Ejemplo de mapeado JOIN

D.9 *Ejemplos de utilización del LED*

D.9.1 *Introducción*

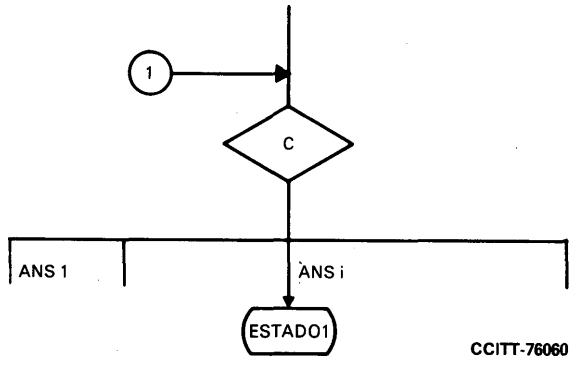
El § D.9 contiene tres ejemplos de utilización del LED. Los ejemplos se han tomado en el sector de aplicación de las telecomunicaciones y en ellos se han empleado diferentes subconjuntos del LED. Se ha procurado que los ejemplos sean lo más reales posible e incluyan el mayor número posible de conceptos del LED.

En cada ejemplo se presenta un sistema en un nivel elevado de abstracción y una sola parte del sistema en forma detallada, declarándose las demás partes como «indefinidas».

Una condición esencial para una definición de sistema es que todas sus distintas partes estén enlazadas de forma no ambigua en un conjunto completo (véase el § D.5). Teniendo en cuenta que las Recomendaciones no proporcionan construcciones de lenguaje que cumplan este requisito, se han tenido que preparar tales construcciones en el marco de los ejemplos. Estas construcciones de referencia están basadas en las reglas sintácticas del LED.

Como algunos de los usuarios del LED preferirán probablemente otros tipos de construcciones de referencia, se han utilizado dos procedimientos principales en los ejemplos. Un procedimiento está basado en la sintaxis LED/PR y el segundo utiliza comentarios en los diagramas gráficos.

Los ejemplos tienen por objeto mostrar la utilización del LED; no son especificaciones internacionales.



a) LED

```
1: CASE C OF
     (ANS1): ..... ;
     (ANS2): ..... ;
       .
       .
       .
     (ANSi): ..... ; nextstate := state1 ;
   ESAC 1 ;
```

b) CHILL

FIGURA D-206

Ejemplo de mapeado de DECISIÓN

D.9.2 *Sistema de conmutación telefónica*

En este ejemplo sólo se utiliza el LED básico (Recomendación Z.101).

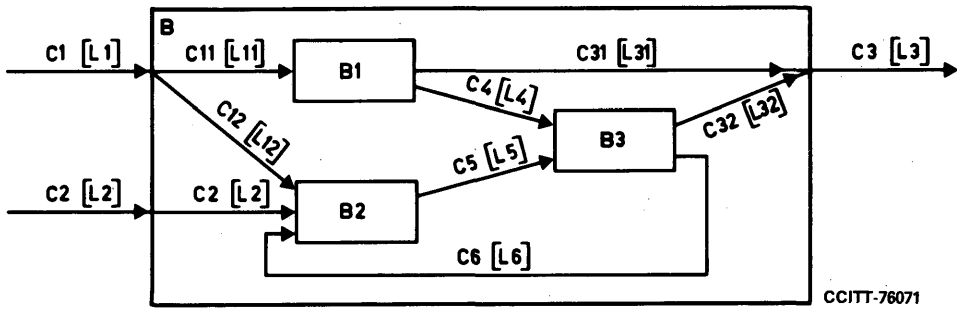
La definición del sistema se presenta en todas las formas sintácticas concretas del LED, a fin de facilitar su comparación.

Este ejemplo se refiere a un proceso de tratamiento de llamadas en un sistema de conmutación telefónica, y supone un conocimiento básico de telefonía. En la primera fase (establecimiento de la llamada) se establece una conexión entre el abonado que llama (A) y el abonado llamado (B), ambos pertenecientes a la misma central. En la segunda fase (conservación), el proceso está a la espera de señales de terminación de la llamada. En la última fase (terminación de la llamada), se desconectan los abonados y el proceso vuelve al estado de reposo.

D.9.2.1 *Forma sintáctica del LED/GR en las dos versiones*

La forma sintáctica gráfica normalmente utilizada (versión orientada a trasiciones) no emplea pictogramas de estado, en tanto que la otra (versión orientada a estados) los utiliza. El diagrama de interacción de bloques es común a ambas formas.

En la versión basada en estados del ejemplo no se han incluido definiciones variables y la iniciación y reinicialización de los temporizadores se muestra de forma implícita. Además, hay que tener en cuenta que los estados siguientes se muestran por un símbolo de estado de tamaño más pequeño que contiene, como identificación suficiente, el sólo número de estado.



a) Sintaxis GR

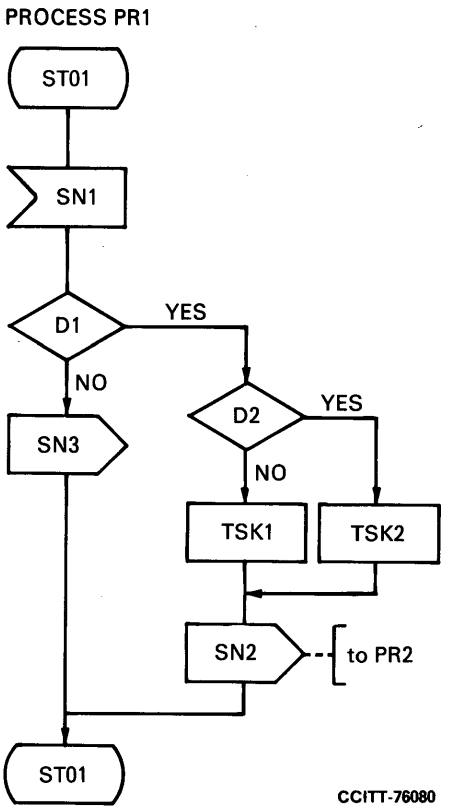
```
BLOCK B;
   SUBSTRUCTURE;
   SUBBLOCKS: B1, B2, B3;
   SPLIT C1 INTO C11, C12;
   SPLIT C2 INTO C2;
   SPLIT C3 INTO C31, C32;
   CHANNEL C11 FROM ENV TO B1 WITH /* L11 */;
   CHANNEL C12 FROM ENV TO B2 WITH /* L12 */;
   CHANNEL C2  FROM ENV TO B2 WITH /* L2 */;
   CHANNEL C31 FROM B1 TO ENV WITH /* L31 */;
   CHANNEL C32 FROM B3 TO ENV WITH /* L32 */;
   /* Los nuevos canales */
   CHANNEL C4 FROM B1 TO B3 WITH /* L4 */;
   CHANNEL C5 FROM B2 TO B3 WITH /* L5 */;
   CHANNEL C6 FROM B3 TO B2 WITH /* L6 */;
   ENDSUBSTRUCTURE;
ENDBLOCK B;
```

b) Sintaxis PR

Nota – El programa PR no está completo de acuerdo con las reglas de sintaxis.

FIGURA D-207

Correlación entre GR y PR para un diagrama de interacción de bloque




```
PROCESS PR1 ;
  STATE ST01 ;
    INPUT SN1 ;
     DECISION D1 ;
     (NO): OUTPUT SN3 ;
     (YES): DECISION D2 ;
            (NO): TASK TSK1 ;
            (YES): TASK TSK2 ;
            ENDDECISION ;
  OUTPUT SN2 TO PR2 ;
 ENDDECISION ;
NEXTSTATE ST01 ;
ENDPROCESS PR1 ;
```

Observación – D1 y D2 son datos definidos formalmente. PR2 son datos de tipo PID en el ejemplo PR, pero puede ser el nombre de proceso en el ejemplo GR.

FIGURA D-208

Correlación entre GR y PR para un diagrama de procesos

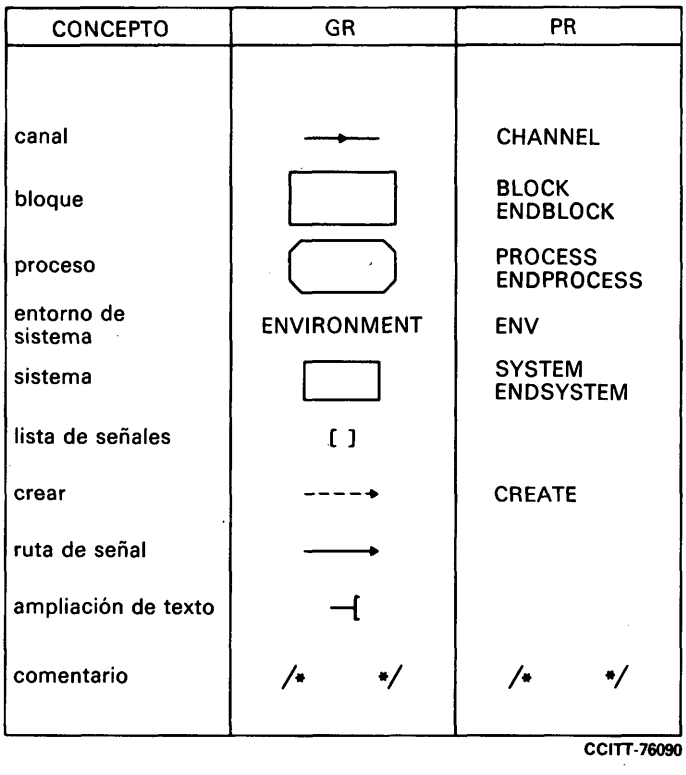


FIGURA D-209

Correlación entre las construcciones GR y PR para los símbolos utilizados en diagramas de interacción de bloques

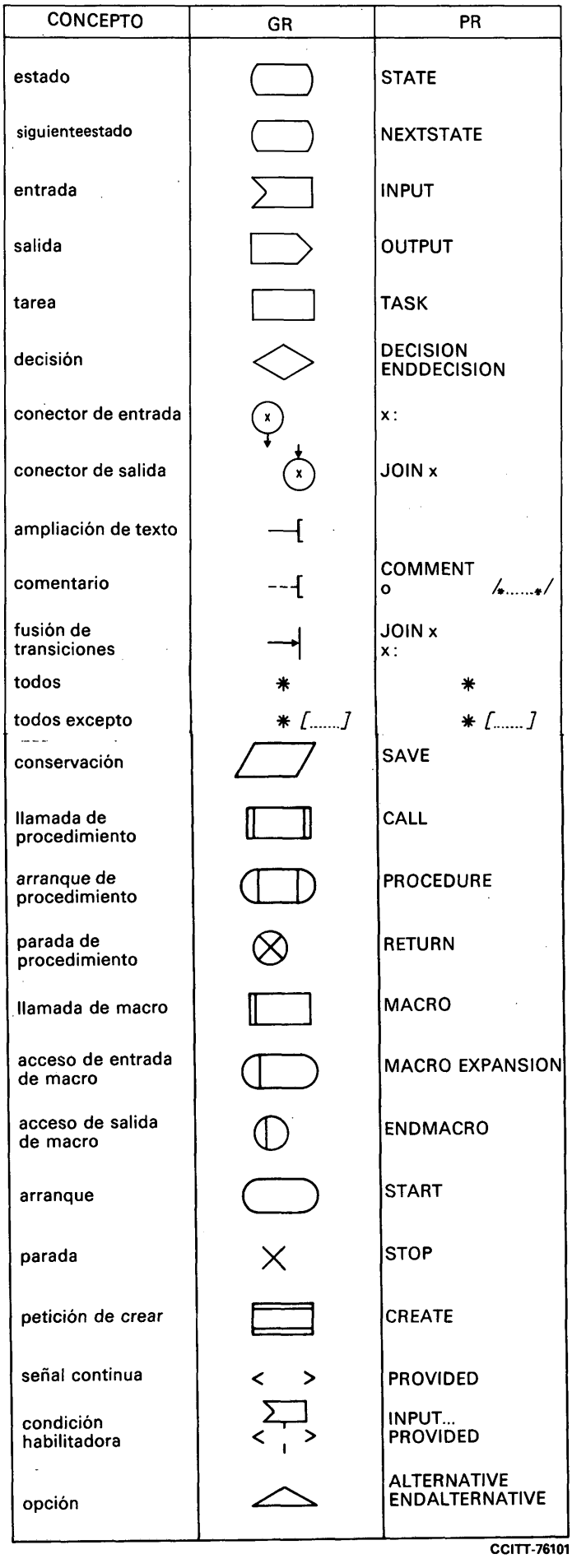


FIGURA D-210

Correlación entre las construcciones GR y PR dentro de un diagrama de proceso

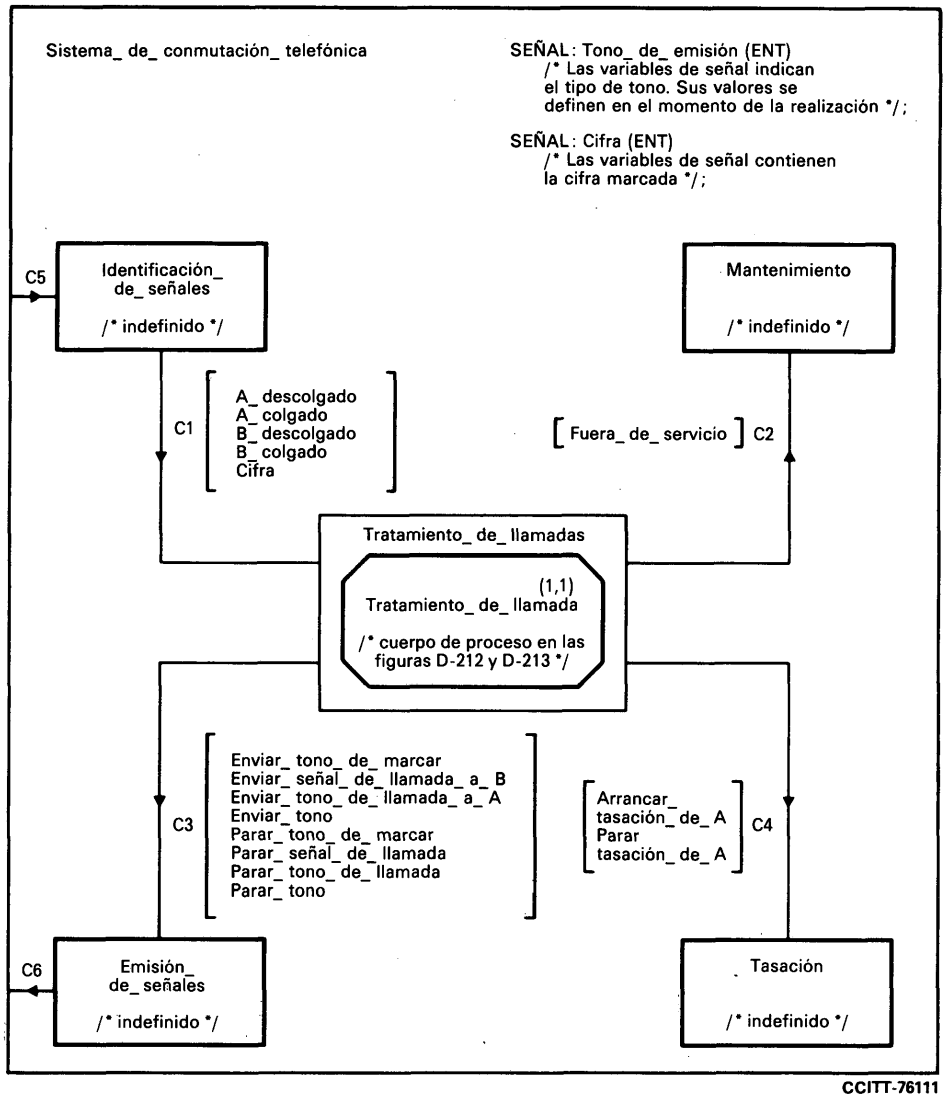


FIGURA D-211

Diagrama de interacción de bloques del sistema Sistema de conmutación telefónica

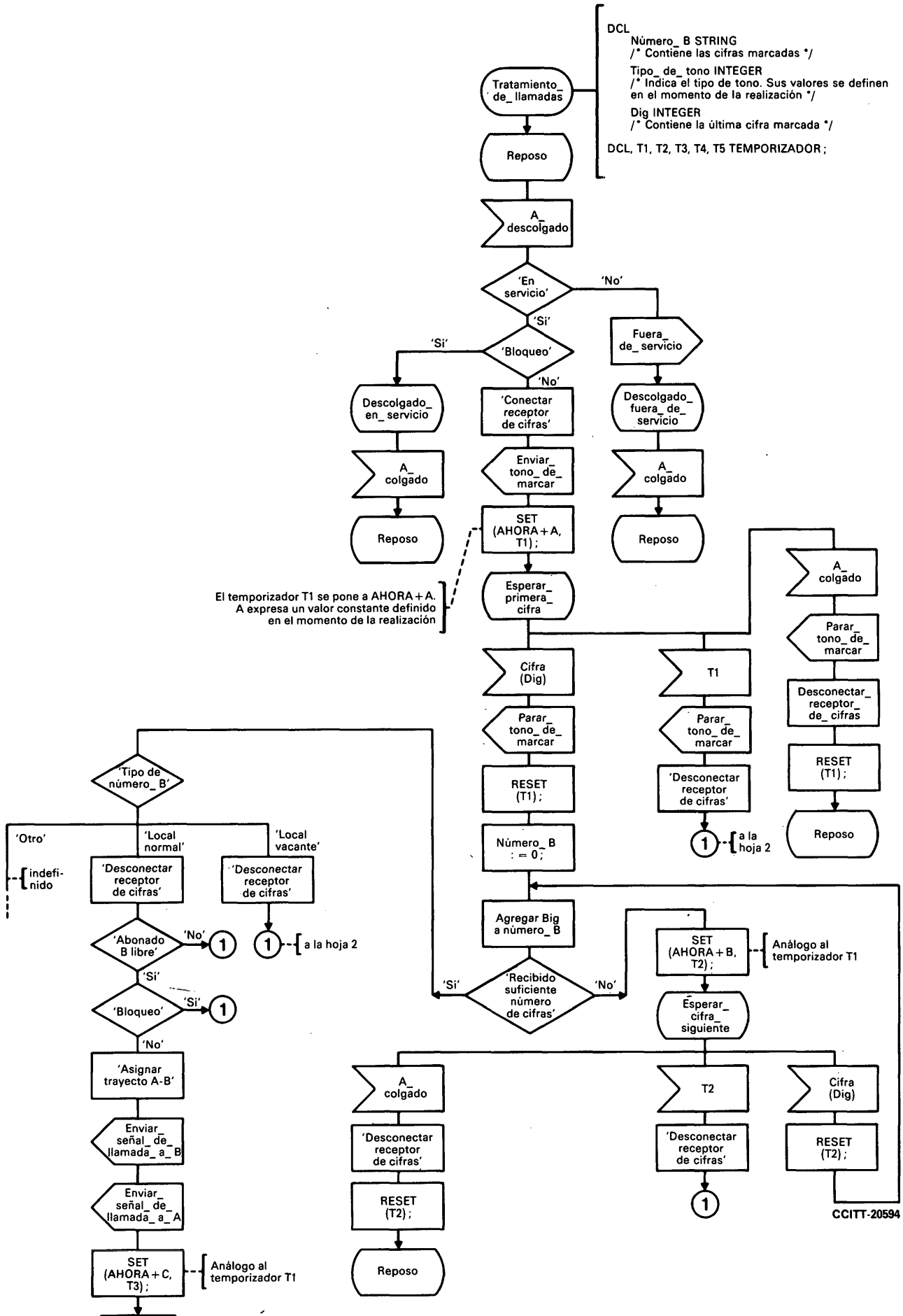


FIGURA D-212 (hoja 1 de 2)

Diagrama para el proceso de tratamiento_de_llamadas, versión basada en transiciones

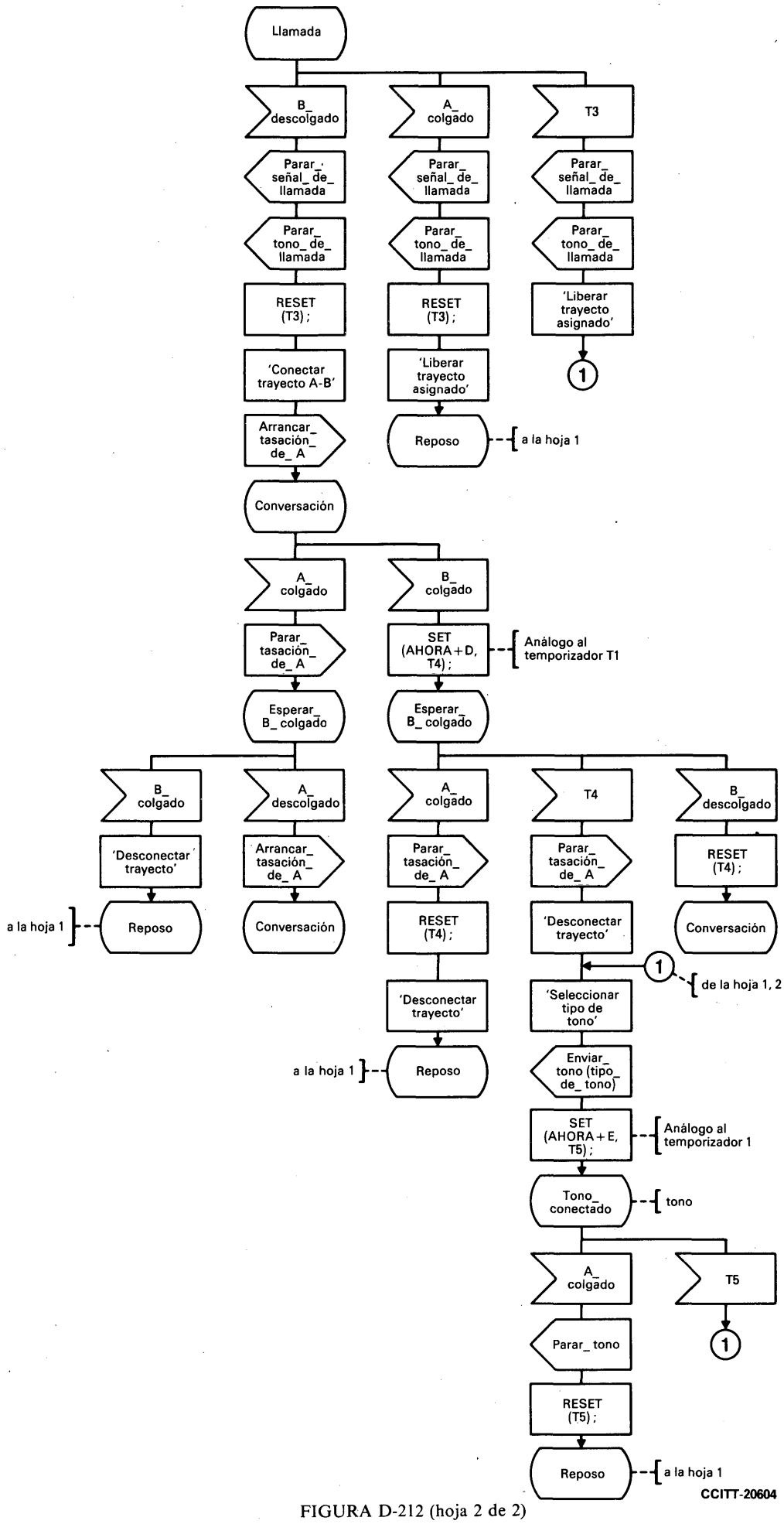


FIGURA D-212 (hoja 2 de 2)

Diagrama para el proceso de tratamiento_de_llamadas, versión basada en transiciones

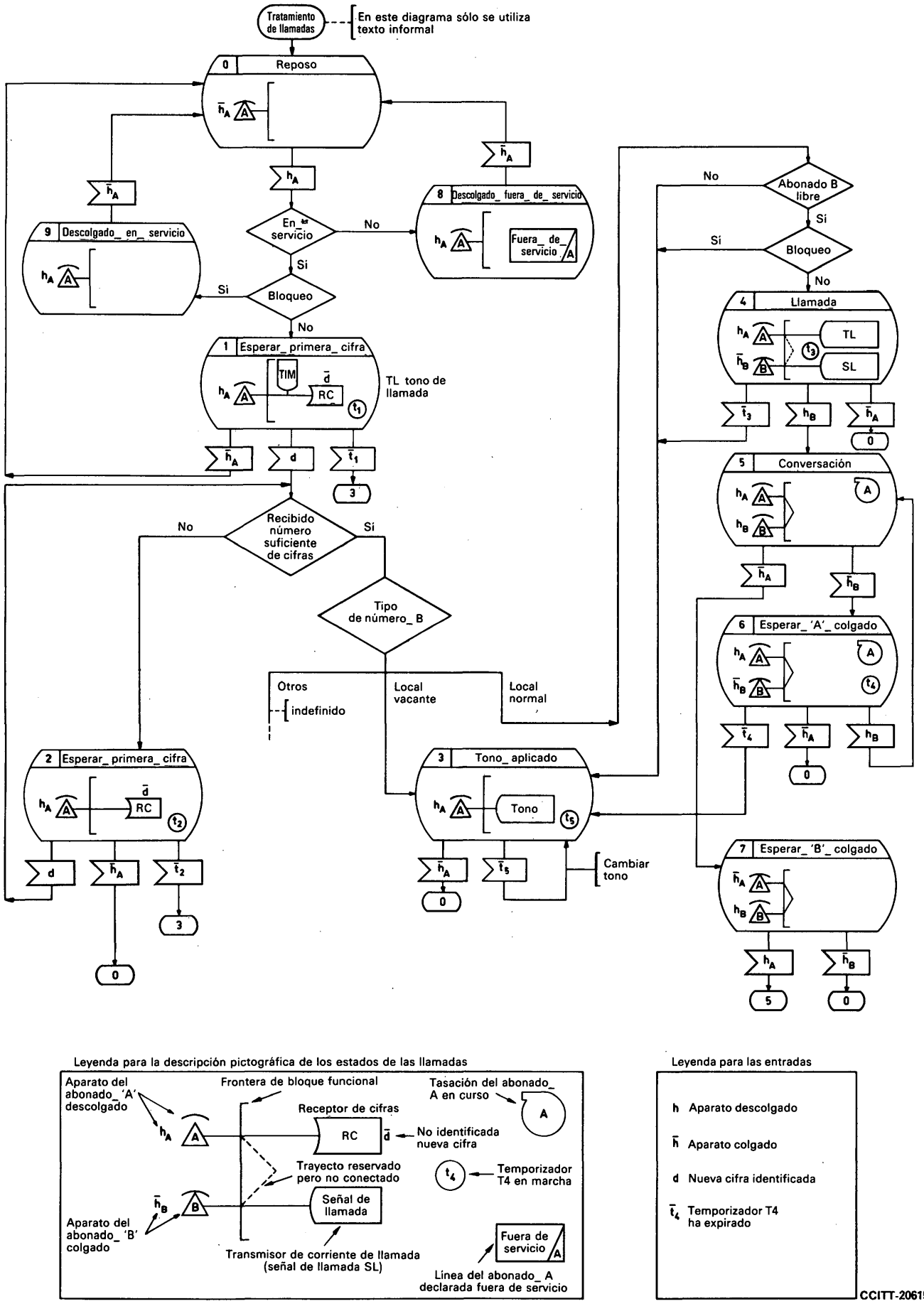


FIGURA D-213

Diagrama para el proceso de tratamiento_ de_ llamadas, versión basada en estados

D.9.2.2 *Forma sintáctica del LED/PR*

```
SYSTEM Telephone_switching_system;
        CHANNEL C1 FROM Signal_recognition TO Call_handling
                WITH    A_off_hook,
                        A_on_hook,
                        B_off_hook,
                        B_on_hook,
                        Digit;
        CHANNEL C2 FROM Call_handling TO Maintenance
                WITH    Out_of_service;
        CHANNEL C3 FROM Call_handling TO Signal_sending
                WITH    Send_dial_tone,
                        Send_ring_signal_to_B,
                        Send_ring_tone_to_A,
                        Send_tone,
                        Stop_dial_tone,
                        Stop_ring_signal,
                        Stop_ring_tone,
                        Stop_tone;
        CHANNEL C4 FROM Call_handling TO Metering
                WITH    Start_metering_A,
                        Stop_metering_A;
        CHANNEL C5 FROM ENV TO Signal_recognition
                WITH    /* undefined */;
        CHANNEL C6 FROM Signal_sending to ENV
                WITH    /* undefined;
        SIGNAL Send_tone (INT)
                /* The signal variable indicates the type of the tone.
                Its values are implementation defined */;
        SIGNAL Digit (INT)
                /* The signal variable contains the dialled digit */;


BLOCK Signal_recognition /* undefined */;
ENDBLOCK;
BLOCK Signal_sending /* undefined */;
ENDBLOCK;
BLOCK Maintenance /* undefined */;
ENDBLOCK;
BLOCK Metering /* undefined */;
ENDBLOCK;
```

FIGURA D-214 (hoja 1 de 4)

Definición del sistema mediante la sintaxis LED/PR

```
BLOCK Call_handling
    PROCESS Call_handling (1,1);
        DCL B_number STRING
            /* contains the dialled digit */;
        DCL Tone_type INTEGER
            /* Indicates the type of the tone. Its values are implementation defined */;
        DCL Dig INTEGER
            /* contains the last dialled digit */;
        START; NEXSTATE Idle;
        STATE Idle;
            INPUT A_off_hook;
                DECISION 'In service';
                    ('No'):     OUTPUT Out_of_service;
                                NEXTSTATE Off_hook_out_of_service;
                    ('Yes'):    DECISION 'Blocking';
                                    ('Yes'):    NEXTSTATE Off_hook_in_service;
                                    ('No'):     TASK 'Connect digit receivers';
                                                OUTPUT Send_dial_tone;
                                                TASK SET (NOW +A, T1);
                                                /* Timer T1 is set to NOW +A. A denotes a constant value that is
                                                implementation defined */
                                                NEXTSTATE Await_first_digit;
                                ENDDECISION;
                ENDDECISION;
        STATE Off_hook_in_service;
            INPUT A_on_hook;
                NEXTSTATE Idle;
        STATE Off_hook_out_of_service;
            INPUT A_on_hook;
                NEXTSTATE Idle;
        STATE Await_first_digit;
            INPUT Digit (Dig);
                OUTPUT Stop_dial_tone;
                TASK RESET (T1),
                    B number:=0;
            10: TASK 'Append Dig to B_number';
                DECISION 'Sufficient number of digits received';
                    ('No'):     TASK SET (NOW +B,T2);
                                NEXTSTATE Await_next_digit;
                    ('Yes'):    DECISION 'Type of B_number';
                                    ('Local normal'): TASK 'Disconnect digit receiver';
                                        DECISION 'B party free';
                                            ('No'):     JOIN 1;
                                            ('Yes'):    DECISION 'Blocking';
                                                            ('Yes'):    JOIN 1;
                                                            ('No'):     TASK 'Allocate path A-B';
                                                                        OUTPUT Send_ring_signal_to_B;
                                                                        OUTPUT Send_ring_tone_to_A;
                                                                        TASK SET (NOW +C, T3);
                                                                        NEXTSTATE Ringing;
                                                        ENDDECISION;
                                        ENDDECISION:
                                    ('Local vacant'):       TASK 'Disconnect digit receiver';
                                                            JOIN 1;
                                    ('Other'):              /* undefined */;
                                ENDECISION;
                ENDDECISION;
```

FIGURA D-214 (hoja 2 de 4)

Definición del sistema mediante la sintaxis LED/PR

```
        INPUT T1;
            OUTPUT Stop_dial_tone;
            TASK 'Disconnect digit receiver';
            JOIN 1;
        INPUT A_on_hook;
            OUTPUT Stop_dial_tone;
            TASK 'Disconnect digit receiver';
            TASK RESET (T1);
            NEXTSTATE Idle;

    STATE Await_next_digit;
        INPUT Digit (Dig);
            TASK RESET (T2);
            JOIN 10;
        INPUT T2;
            TASK 'Disconnect digit receiver';
            JOIN 1;
        INPUT A_on_hook;
            TASK 'Disconnect digit receiver';
            TASK RESET (T2);
            NEXTSTATE Idle;

    STATE Ringing;
        INPUT B_off_hook;
            OUTPUT   Stop_ring_signal,
                     Stop_ring_tone;
            TASK RESET (T3);
            TASK 'Connect path A-B';
            OUTPUT Start_metering_A;
            NEXTSTATE Conversation;
        INPUT A_on_hook;
            OUTPUT   Stop_ring_signal,
                     Stop_ring_tone;
            TASK RESET (T3);
            TASK 'Release allocated path';
            NEXTSTATE Idle;
        INPUT T3;
            OUTPUT   Stop_ring_signal,
                     Stop_ring_tone;
            TASK 'Release allocated path';
            JOIN 1;

    STATE Conversation;
        INPUT A_on_hook;
            OUTPUT Stop_metering_A
            NEXTSTATE Await_B_on_hook;
        INPUT B_on_hook;
            TASK SET (NOW +D, T4);
            NEXTSTATE Await_A_on_hook;
```

FIGURA D-214 (hoja 3 de 4)

Definición del sistema mediante la sintaxis LED/PR

```
        STATE Await_B_on_hook;
            INPUT A_off_hook;
                OUTPUT Start_metering_A;
                NEXTSTATE Conversation;
            INPUT B_on_hook;
                TASK 'Disconnect path';
                NEXTSTATE Idle;
        STATE Await_A_on_hook;
            INPUT A_on_hook
                OUTPUT Stop_metering_A;
                TASK RESET (T4);
                TASK 'Disconnect path';
                NEXTSTATE Idle;
            INPUT B_off_hook;
                TASK RESET (T4);
                NEXTSTATE Conversation;
            INPUT T4;
                OUTPUT Stop_metering_A;
                TASK 'Disconnect path';
            1:  TASK 'Select tone type';
                OUTPUT Send_tone (Tone_type);
                TASK SET (NOW +E, T5);
                NEXTSTATE Tone_connected;


        STATE Tone_connected;
            INPUT A_on_hook;
                OUTPUT Stop_tone;
                TASK RESET (T5);
                NEXTSTATE Idle;
            INPUT T5;
                JOIN 1;
ENDPROCESS Call_handling;
ENDBLOCK Call_handling;
ENDSYSTEM;
```

FIGURA D-214 (hoja 4 de 4)
Definición del sistema mediante la sintaxis LED/PR

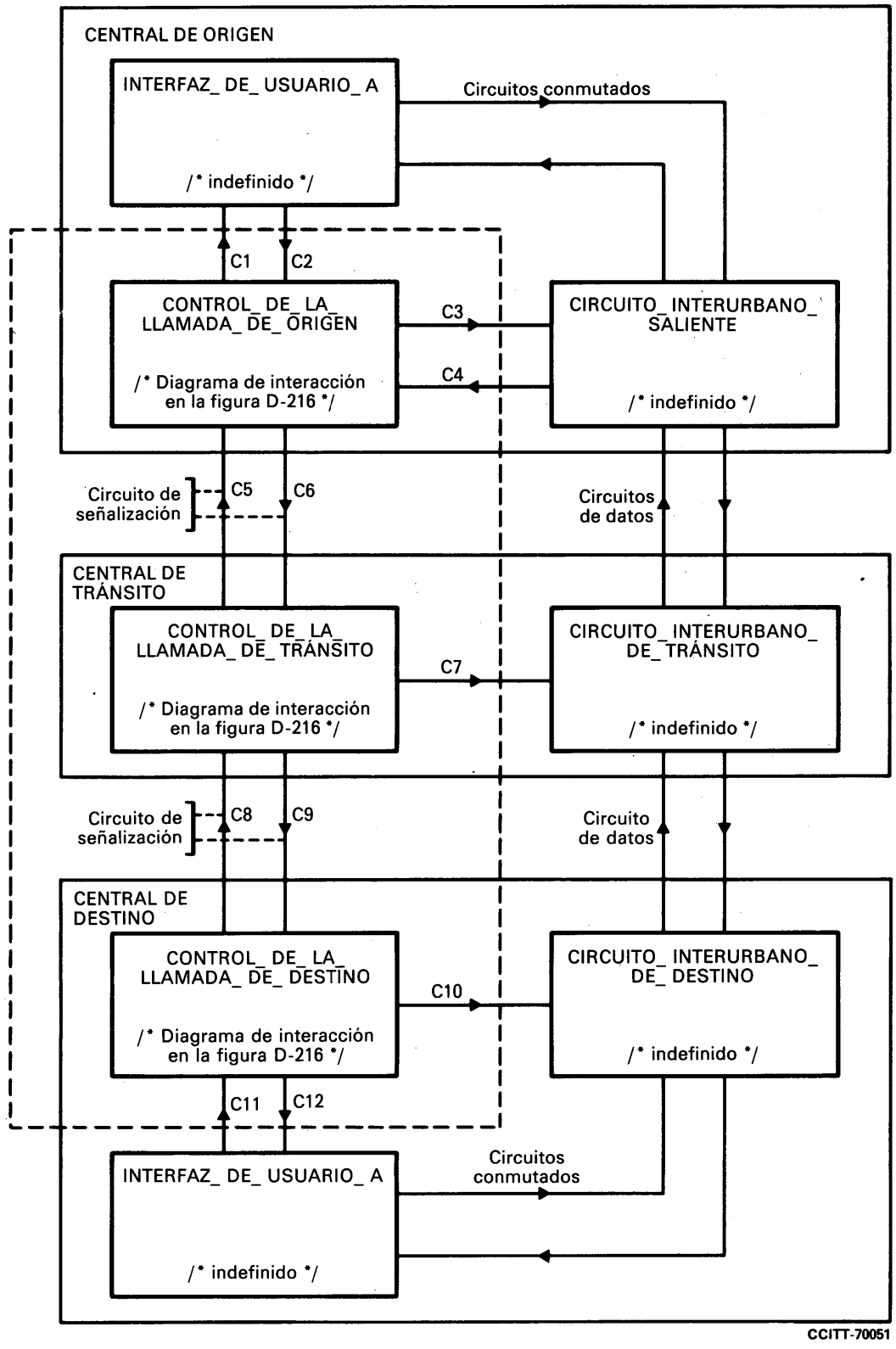
D.9.3 *Parte usuario de datos del sistema de señalización por canal común*

El ejemplo siguiente está basado en la Recomendación X.61. No equivale necesariamente a la Recomendación X.61 y no se ha previsto que constituya una norma internacional. El ejemplo ilustra el empleo del procedimiento y de los conceptos facultativos definidos en la Recomendación Z.103.

En una red con conmutación de circuitos que emplea señalización por canal común, los mensajes de señalización para un grupo de circuitos interurbanos entre centrales se transmiten por una red de señalización centralizada. La figura D-215 da una visión global de la red de datos con conmutación de circuitos que comprende tres centrales. La parte usuario de datos del sistema de señalización por canal común se representa por los bloques funcionales de control de la llamada en las centrales de origen, tránsito y destino. El diagrama global no es parte formal de la especificación, pero se incluye para explicar la relación entre el sistema de señalización y los demás componentes de la red de datos. La figura D-216 contiene el diagrama de interacción de bloques del sistema de señalización.

Los procesos de origen, tránsito y destino (figuras D-217, D-218 y D-219) representan las funciones necesarias para establecer, mantener y terminar una conexión de datos entre dos interfaces de usuario. Durante la fase de establecimiento de la llamada, la central de destino puede optar por una de dos respuestas a la llamada entrante. Estas alternativas se definen por la utilización del símbolo de opción. La aceptación de la llamada por el usuario llamado da como resultado una señal preparado para datos que se envía al proceso de origen cuando se ha efectuado la conexión de la central de destino.

Los procesos de origen, tránsito y destino realizan funciones similares para liberar los circuitos interurbanos cuando la llamada se ha completado. Estas funciones se definen mediante los procedimientos globales representados en las figuras D-220 y D-221.



Observación – La línea de trazo interrumpido muestra los límites de la parte de usuario de datos del sistema de señalización por canal común.

FIGURA D-215

Visión global de una red de datos con conmutación de circuitos que comprende tres centrales

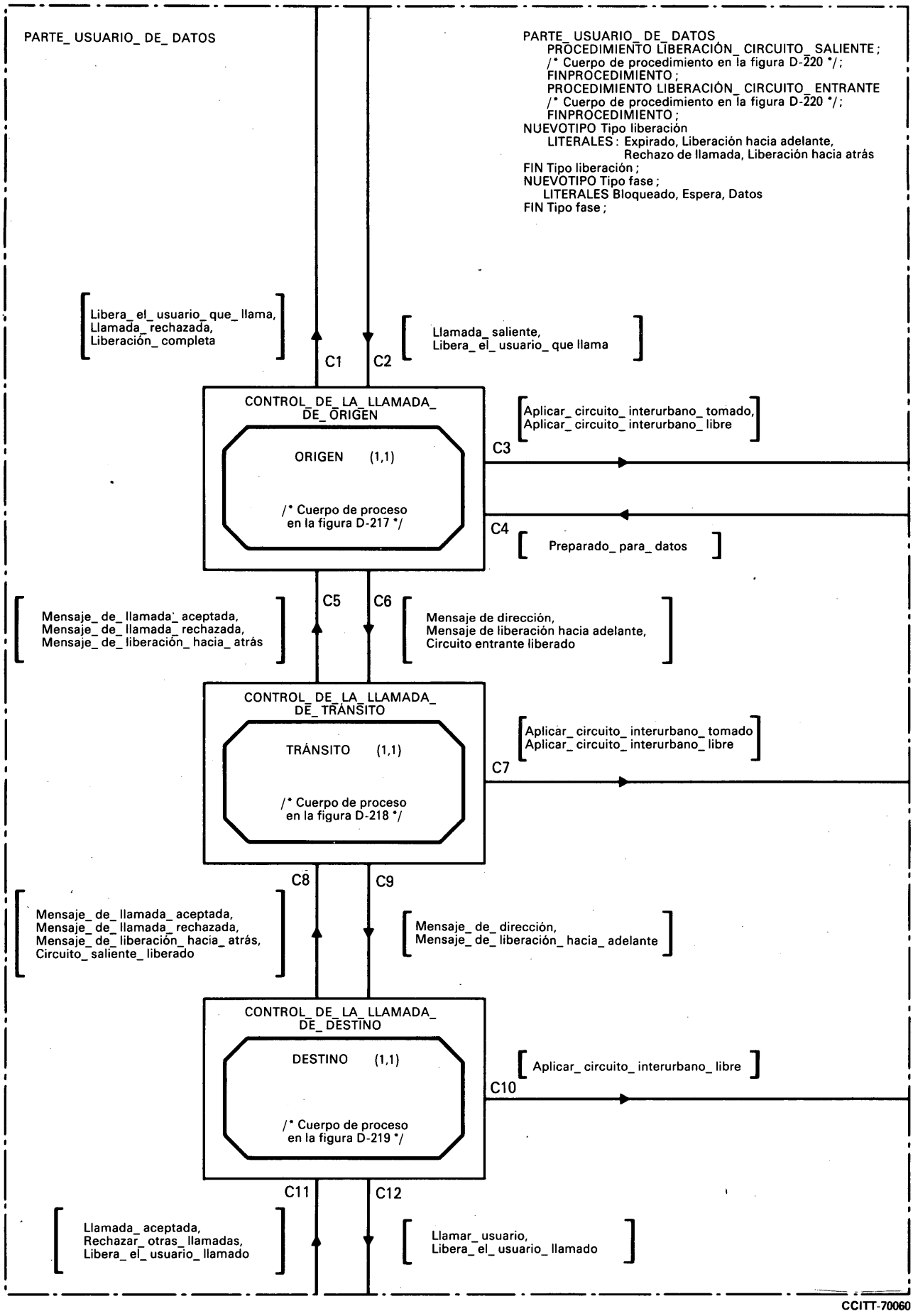


FIGURA D-216

Diagrama de interacción de bloques para el sistema PARTE_ USUARIO_ DE_ DATOS

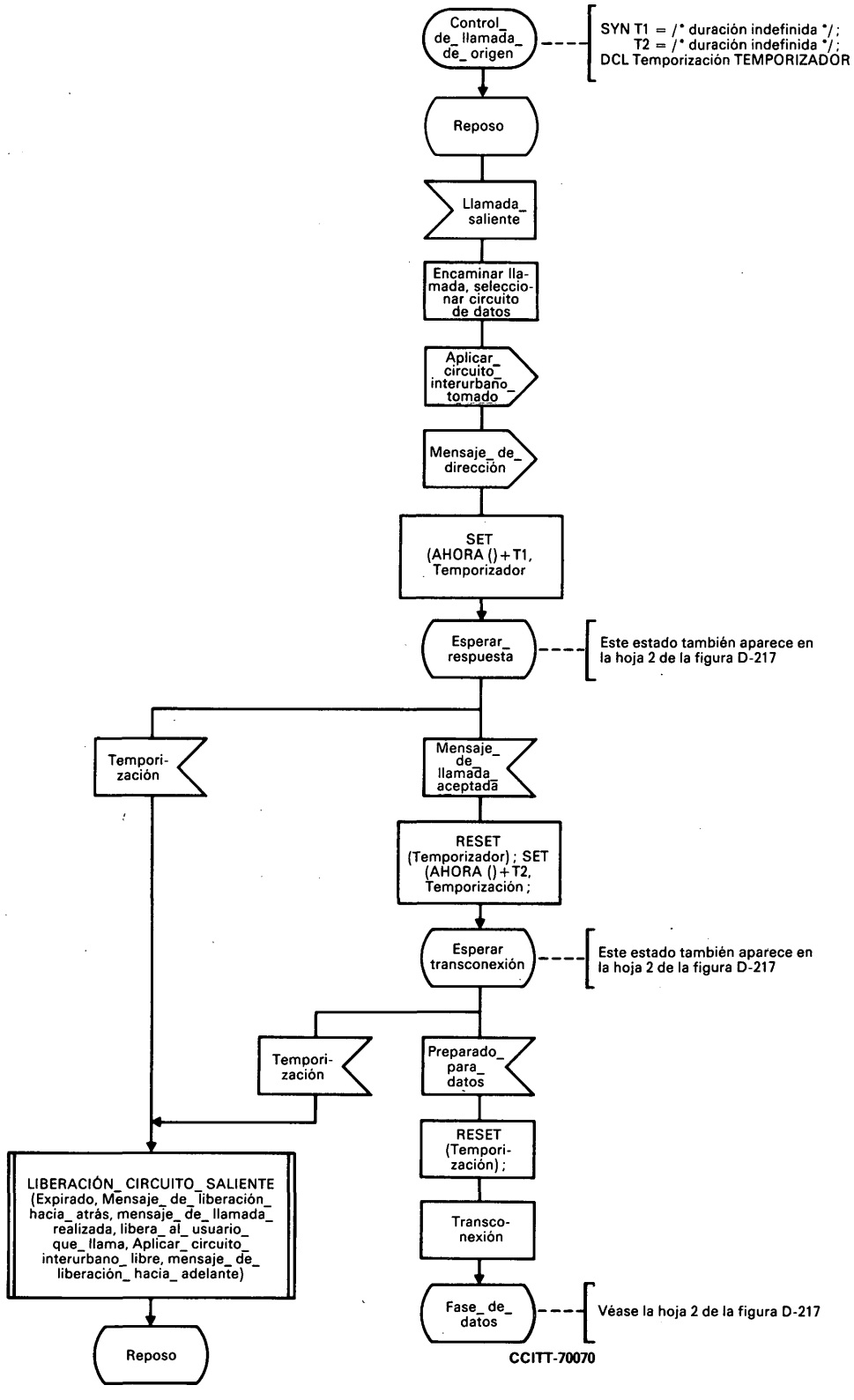


FIGURA D-217 (hoja 1 de 2)

Diagrama de procesos para el control_ de llamadas_ de_ origen

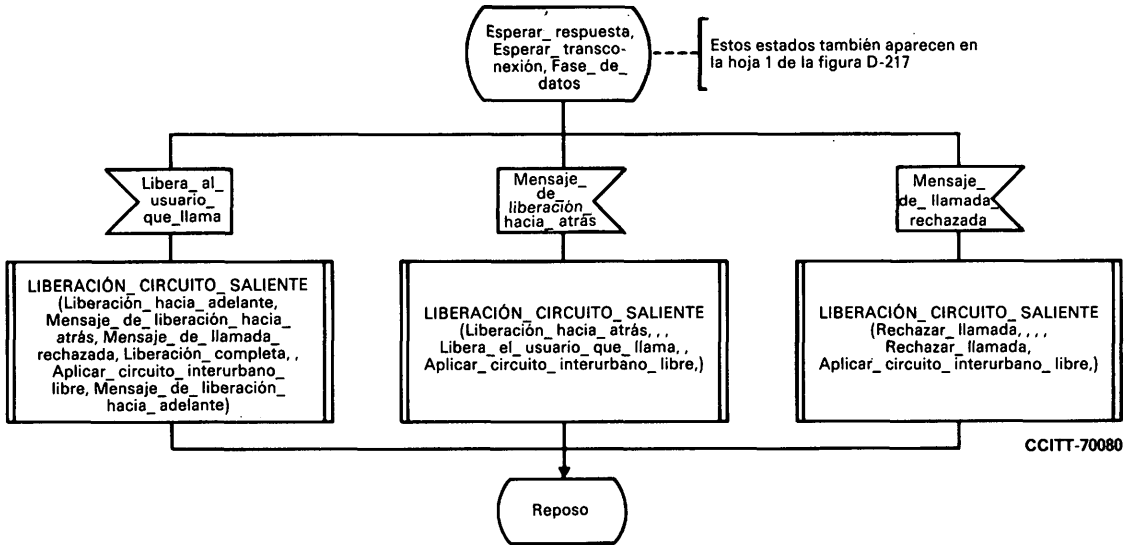


FIGURA D-217 (hoja 2 de 2)

Diagrama de procesos para el control_ de_ llamadas_ de_ origen

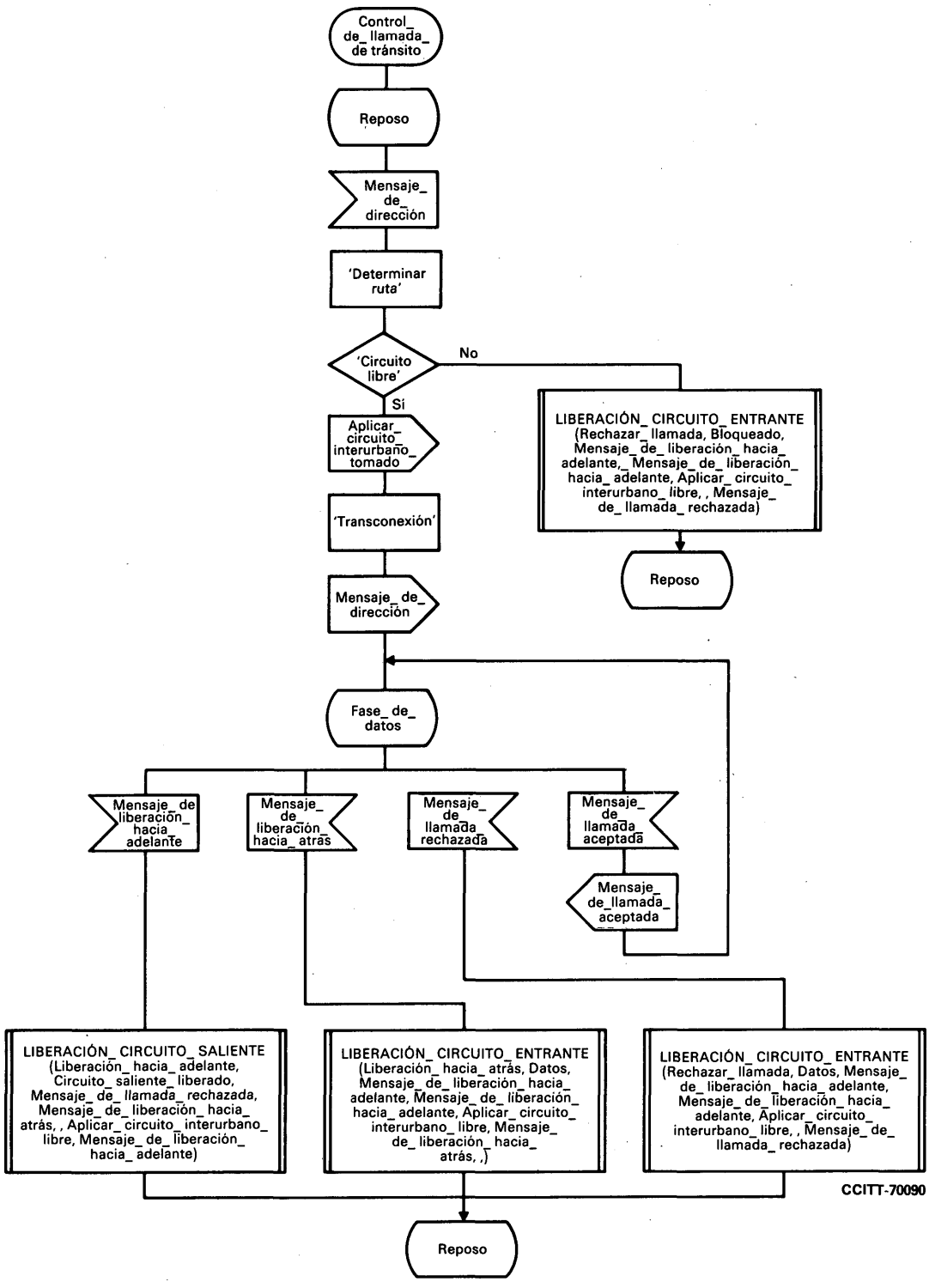


FIGURA D-218

Diagrama de procesos para el control_ de_ llamada_de_ tránsito

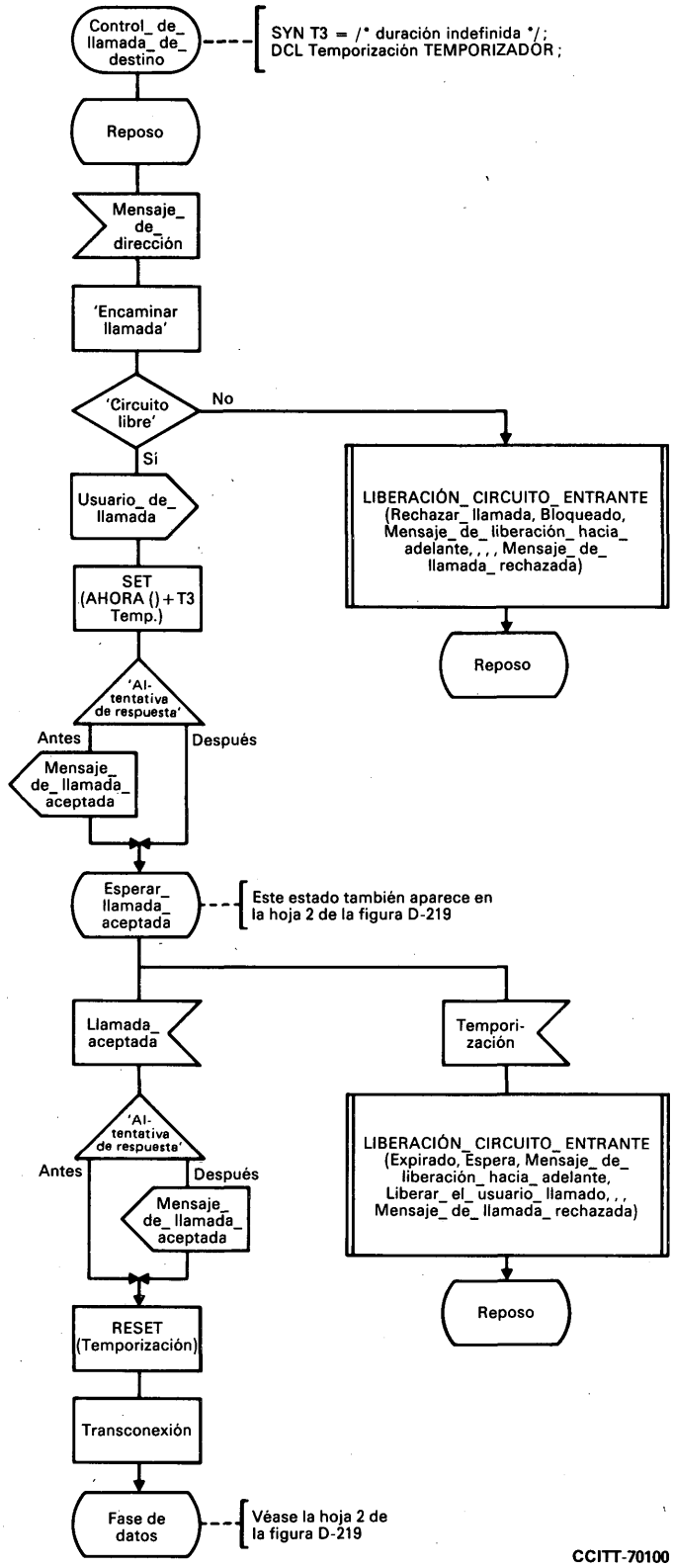


FIGURA D-219 (hoja 1 de 2)

Diagrama de procesos para el control_ de_ llamada_ de_ destino

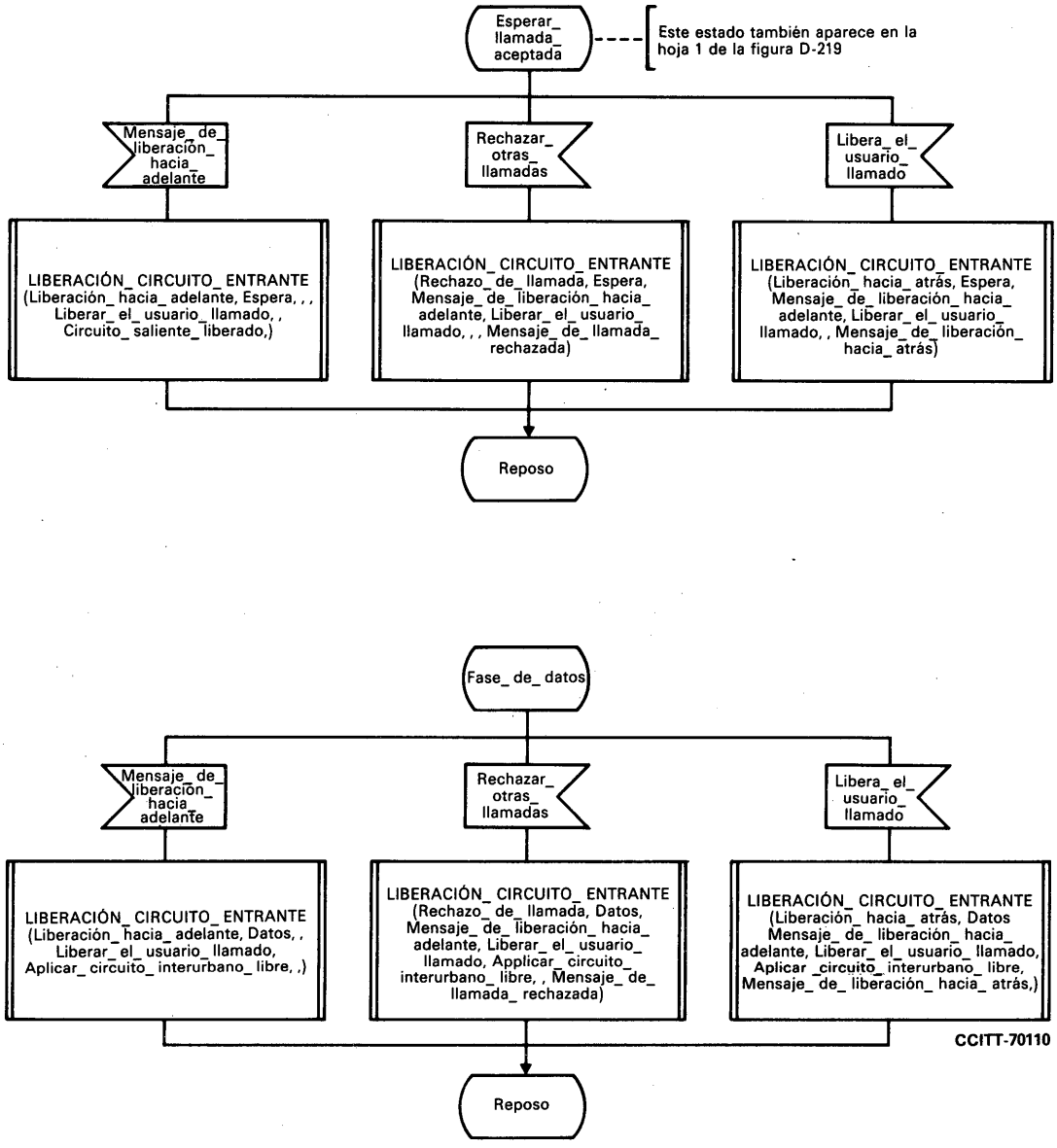


FIGURA D-219 (hoja 2 de 2)

Diagrama de procesos para el control_ de_ llamada_ de_ destino

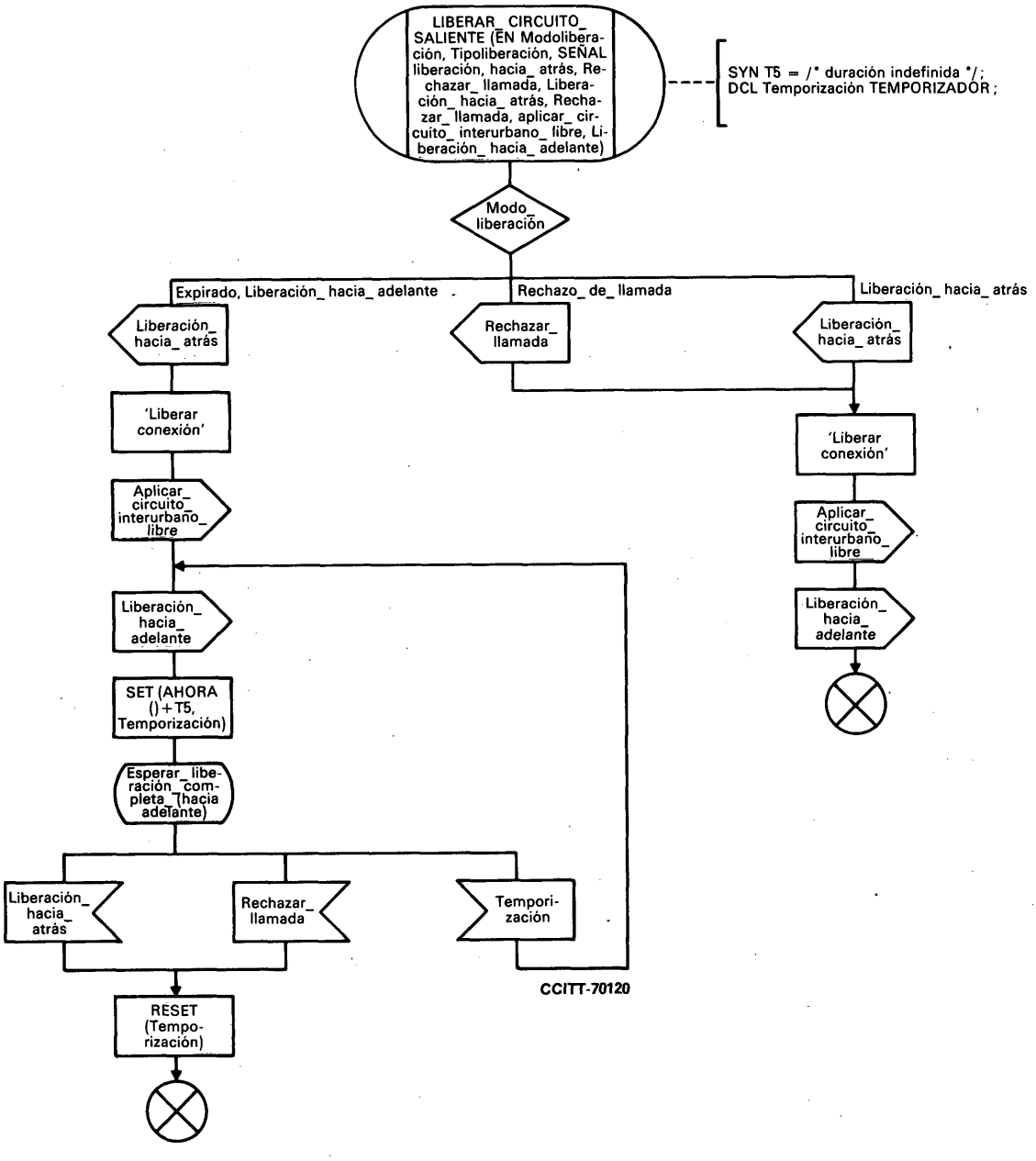


FIGURA D-220

Diagrama de procedimiento para el procedimiento de liberación_del_ circuito_ saliente

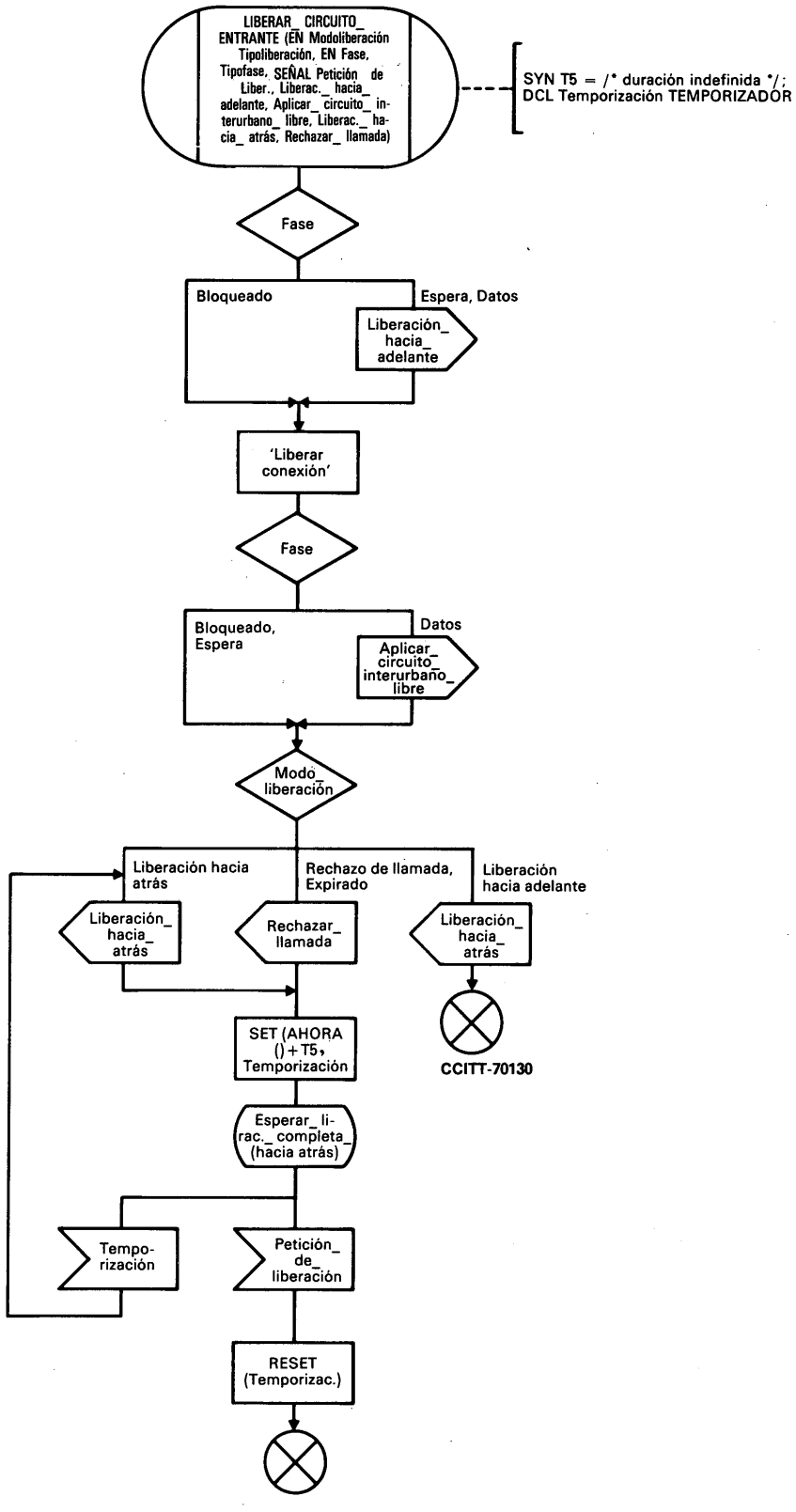


FIGURA D-221

Diagrama de procedimiento para el procedimiento de liberación_ del_ circuito_ entrante

D.9.4 *Protocolo de transporte de clase 0*

El ejemplo siguiente está basado en la Recomendación sobre Protocolo de Transporte para la Interconexión de Sistemas Abiertos. El ejemplo es un subconjunto del Protocolo de Trasporte, que no equivale necesariamente a la Recomendación y no se ha previsto que constituya una norma internacional.

Se ha utilizado el LED, de conformidad con las Recomendaciones Z.101 y Z.103. Se han utilizado datos de manera algo informal, sin aplicar la Recomendación Z.104. Los tipos de enumeración de la Recomendación Z.104 se han simulado con variables ENT, y se han dado nombres a constantes indefinidas. Las operaciones con datos no se han definido formalmente.

D.9.4.1 *Características generales*

La capa de transporte del modelo de referencia de la ISA facilita servicio a sus usuarios proporcionándoles canales de comunicación (denominados conexiones de transporte) entre pares de usuarios. En cada punto de conexión de usuario, la capa de transporte está representada por una entidad de transporte. Las entidades de transporte comunican entre sí utilizando un protocolo de transporte y los servicios de la capa de red. La figura D-223 contiene un diagrama de conjunto. El ejemplo describe una entidad de transporte que utiliza solamente el protocolo de transporte de clase 0.

Se establece una conexión de transporte entre dos entidades de transporte. El modelo en que se ha basado la especificación describe el comportamiento de una entidad independientemente y puede aplicarse a ambos extremos de la conexión. El diagrama de interacción, figura D-224 muestra un bloque simple que representa la entidad de transporte, enlazado al entorno por canales denominados punto de acceso al servicio de transporte (TSAP) y punto de acceso al servicio de red (NSAP). Se considera que el usuario de servicio de transporte y la capa de red constituyen el entorno.

La entidad de transporte contiene dos procesos. El primero, el DIRECTOR, tiene una sola instancia que siempre está presente (indicada por los números en el ángulo superior derecho del símbolo de proceso). El segundo proceso, el PUNTOEXTREMO, sólo tiene instancias creadas dinámicamente, una de las cuales se establece para la duración útil de cada tentativa de establecimiento de una conexión de transporte. Si el DIRECTOR recibe una señal «petición de CONEXIÓN_T» o «indicación de CONEXIÓN_N», este último crea inmediatamente una nueva instancia de PUNTOEXTREMO para tramitar todas las señales subsiguientes asociadas con una tentativa de conexión. Cada instancia de PUNTOEXTREMO tiene una propiedad «tipo», con uno de los valores INICIADOR o RESPONDEDOR, que determina el comportamiento inicial de la instancia; la instancia termina por sí misma cuando uno de los usuarios del servicio de transporte pone fin a la conexión, o después de un error.

El comportamiento de DIRECTOR se indica mediante el diagrama de procesos de la figura D-225. El del PUNTOEXTREMO se indica en las figuras D-226 a D-230. La especificación de PUNTOEXTREMO utiliza el procedimiento LED para las fases de establecimiento de la conexión, transferencia de datos y terminación.

D.9.4.2 *Especificación LED del sistema*

El sistema se especifica mediante el siguiente texto LED-PR. El texto difiere de las Recomendaciones, ya que algunas partes se dan en forma LED-GR con referencias a las partes dadas en el LED-PR en forma de comentarios.

```
SYSTEM TRANSPORT_ENTITY;
(ENTIDAD DE SISTEMA DE TRANSPORTE)
        /*      la función de una ENTIDAD DE TRANSPORTE en una conexión de transporte entre dos usuarios
                se muestra en el diagrama global de la figura D-223 */
        /*      los canales y señales asociadas se muestran en el diagrama de interacción, figura D-224 */
    SIGNAL /* input-list */
        T_CONN_REQ      (PID,STRING)    ,
        T_CONN_RSP      (PID,STRING)    ,
        T_CONN_RPLY                     ,
        T_DATA_REQ      (STRING)        ,
        T_DISC_REQ                      ,
        N_CONN_IND      (PID,STRING)    ,
        N_CONN_CFM      (PID,STRING)    ,
        N_CONN_RPLY                     ,
        N_DATA_IND      (STRING)        ,
        N_DISC_IND                      ,
        N_RESET_IND                     ;
            MACRO EXPANSION signals;
                T_CONN_REQ, T_CONN_RSP, T_CONN_RPLY, T_DATA_REQ, T_DISC_REQ,
        N_CONN_IND, N_CONN_CFM, N_CONN_RPLY, N_DATA_IND, N_DISC_IND,
                N_RESET_IND,
            ENDMACRO signals;
    SIGNAL /* output-list */
        T_CONN_IND      (PID,STRING)    ,
        T_CONN_CFM      (PID,STRING)    ,
        T_DATA_IND      (STRING)        ,
        T_DISC_IND                      ,
        N_CONN_REQ      (PID,STRING)    ,
        N_CONN_RSP      (PID,STRING)    ,
        N_DATA_REQ      (STRING)        ,
        N_DISC_REQ                      ;
```

FIGURA D-222 (hoja 1 de 3)

Especificación del sistema mediante el LED/PR

```
BLOCK TRANSPORT_ENTITY;
    PROCESS DIRECTOR (1,1);
        /* the single instance of this process controls the assignment of ENDPOINTs */
        DCL  conn_params STRING,
             conn_id PID ;
        /* process diagram, Figure D-225, gives process body */
    ENDPROCESS DIRECTOR;
        PROCESS ENDPOINT (0, );
            /* an instance of this process is created by DIRECTOR for each connection requested */
            FPAR
                    kind            INT /* values INITIATOR, RESPONDER */ ,
                    conn_params     STRING ,
                    conn_id         PID;
            DCL
                    ref             STRING ,
                    tc_id           PID ,
                    nc_id           PID ,
                    tpdu_size       INT  /* values                                    */,
                    cstat           INT  /* values DATA_TRANSFER, DISCONNECTED   */;
                        IMPORTED /* from tc_id via channel TSAP_in */
                    t_ind_flo       BOOL ,
                        IMPORTED /* from nc_id via channel NSAP_in */
                    n_req_flo       BOOL ,
                        EXPORTED /* to tc_id via channel TSAP_out */
                    t_req_flo       BOOL ,
                        EXPORTED /* to nc_id via channel NSAP_out */
                    n_ind_flo       BOOL ;

    PROCEDURE INITIATE_CONNECT;
        SIGNAL      MACRO signals;
        IN          conn_params     STRING ,
        IN          tc_id           PID ,
        IN/OUT      nc_id           PID ,
        IN/OUT      tpdu_size       INT ,
        IN/OUT      cstat           INT /* values DATA_TRANSFER, DISCONNECTED */
      DCL
        n_conn_params STRING ,
        prop_tpdu_size INT ,
        tpdu_flag       INT /* values CC, DR, ER, CR, DT, INV */ ,
        tpdu_data       STRING ,
        nbin            STRING ,
        nbout           STRING ,
        tbout           STRING ,
        taskr           INT /* values CLOSE, NC, TC, NC&TC, NONE */;
      /* procedure diagram, Figure D-227, gives procedure body */
    ENDPROCEDURE INITIATE_CONNECT;
    PROCEDURE RESPOND_CONNECT
        SIGNAL      MACRO signals;
        IN          conn_params     STRING ,
        IN/OUT      tc_id           PID ,
        IN          nc_id           PID ,
        IN/OUT      tpdu_size       INT ,
        IN/OUT      cstat INT /* values DATA_TRANSFER, DISCONNECTED */ ;
      DCL
        n_conn_params STRING ,
        prop_tpdu_size INT ,
        tpdu_flag       INT /* values CC, DR, ER, CR, DT, INV */ ,
        tpdu_data       STRING ,
        nbin            STRING ,
        nbout           STRING ,
        tbin            STRING ,
        tbout           STRING ,
        taskr           INT /* values CLOSE, NC, TC, NC&TC, NONE */
      /* procedure diagram, Figure D-228, gives procedure body */
    ENDPROCEDURE RESPOND_CONNECT;
```

FIGURA D-222 (hoja 2 de 3)

Especificación del sistema mediante el LED/PR

```
    PROCEDURE DATA_PHASE;
        SIGNAL      MACRO signals;
        IN          ref             STRING ,
        IN          tc_id           PID ,
        IN          nc_id           PID ,
        IN          tpdu_size       INT ,
        IN/OUT      cstat           INT /* values DATA_TRANSFER, DISCONNECTED */,
        IN/OUT IMPORTED /* from tc_id via channel TSAP_IN */
        t_ind_flo                   BOOL ,
        IN/OUT IMPORTED /* from nc_id via channel NSPA_in */
        n_req_flo                   BOOL ,
        IN/OUT EXPORTED /* to tc_id via channel TSAP_out */
        t_req_flo                   BOOL ,
        IN/OUT EXPORTED /* to nc_id via channel NSAP_out */
        n_ind_flo                   BOOL ;
      DCL
        input_present               BOOL ,
        output_present              BOOL ,
        tbin                        STRING ,
        nbin                        STRING ,
        tbout                       STRING ,
        nbout                       STRING ,
        output_segment              STRING ,
        input_tsdu                  STRING ,
        tpdu_flag                   INT /* values CC, DR, ER, CR, DT, INV */ ,
        taskr                       INT /* values CLOSE, NC, TC, NC&TC, NONE */;
      /* procedure diagram, Figure D-229, gives procedure body */
    ENDPROCEDURE DATA_PHASE;
    PROCEDURE RELEASE_CONNECT
        SIGNAL      MACRO signals;

        IN          taskr           INT /* values CLOSE, NC, TC, NC&TC, NONE */,
        IN          tc_id           PID ,

        IN          nc_id           PID ,
        IN/OUT      cstat           INT /* values DATA_TRANSFER, DISCONNECTED */,
        IN/OUT      tpdu_flag       INT /* values CC, DR, ER, DT, INV */ ,
        IN/OUT      last_tpdu       STRING;
      DCL
        nbin                        STRING ,
        nbout                       STRING ,
        tbout                       STRING ,
    /* procedure diagram, Figure D-230, gives procedure body */
    ENDPROCEDURE RELEASE_CONNECT;
  /* process diagram, Figure D-226, gives process body */
ENDPROCESS ENDPOINT;
ENDBLOCK TRANSPORT_ENTITY;
ENDSYSTEM TRANSPORT_ENTITY;
```

FIGURA D-222 (hoja 3 de 3)

Especificación del sistema mediante el LED/PR

D.9.4.3 *Operaciones con relación a los elementos de datos*

Las operaciones relativas a los elementos de datos se representan de modo informal. Son las siguientes:

alloc_ref: Da una referencia de conexión de protocolo de transporte única.

form_t_conn_ind:
form_t_conn_cfm:
form_t_data_ind:
form_t_disc_ind:
form_n_conn_req:
form_n_data_req:
form_n_conn_rsp:
form_n_disc_req:

Cada una de estas operaciones constituye el conjunto de campos de datos para el primitivo de servicio correspondiente de los parámetros suministrados, junto con antecedentes sobre la entidad de transporte (por ejemplo, mapeado entre direcciones de transporte y de red en el caso de n_conn_req).

conn_feasible: El acuse de recibo de la entidad de transporte en el TSAP indica que la conexión se puede realizar (recursos locales disponibles, TSAP llamado que se puede alcanzar, calidad de servicio que se puede lograr) – booleano

conn_acceptable: La entidad de transporte en el NSAP llamado tiene recursos para aceptar la conexión de red – booleano

conn_providable: El acuse de recibo de la entidad de transporte en el extremo llamado indica que la conexión de transporte se puede realizar (TSAP llamado disponible, calidad de servicio alcanzada) – booleano

tpdu-type: Determina el contenido de n_data_ ind (valores: CR, CC, DT, DR, ER, o INV – esta última indica un TPDU que por cualquier motivo no es válido)

prop_tpdu_size: Proposición del iniciador relativa al valor de tpdu_size

agree_tpdu_size: Valor aceptado por el respondedor para tpdu_size

extract_tpdu_size: Determina el valor de parámetro contenido en CC TPDU

take_segment: Toma el siguiente DT TPDU desde TSDU

append_segment: Agrega el DT TPDU vigente al TSDU parcialmente completado

last_ tpdu: El DT TPDU vigente lleva la indicación Fin de TSDU

Los valores para las entradas a operaciones se indican entre paréntesis después del nombre de la operación.

FIGURA D-223

Diagrama global que ilustra la función de una ENTIDAD_DE_TRANSPORTE en una conexión de transporte

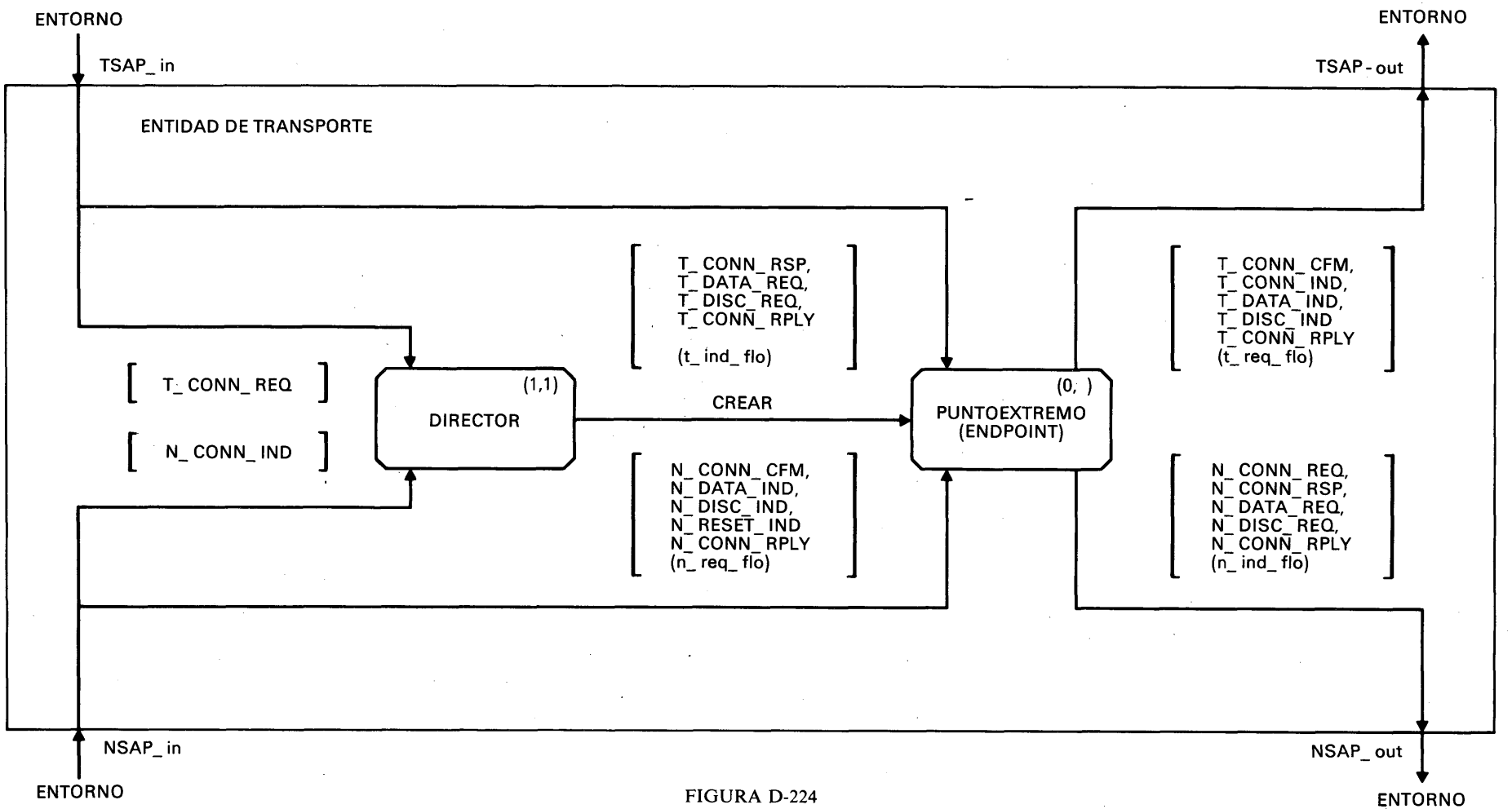


FIGURA D-224

Diagrama de interacción para el bloque ENTIDAD_ DE_ TRANSPORTE

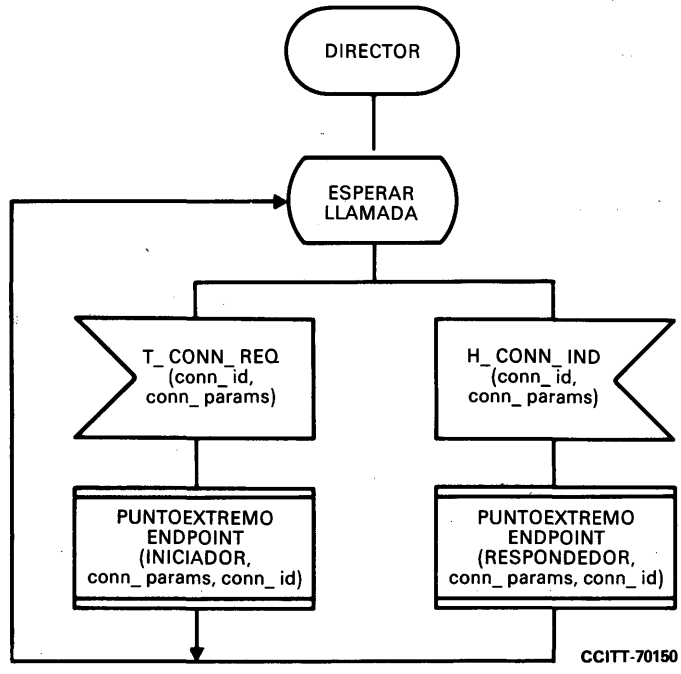


FIGURA D-225

Diagrama de procesos para DIRECTOR

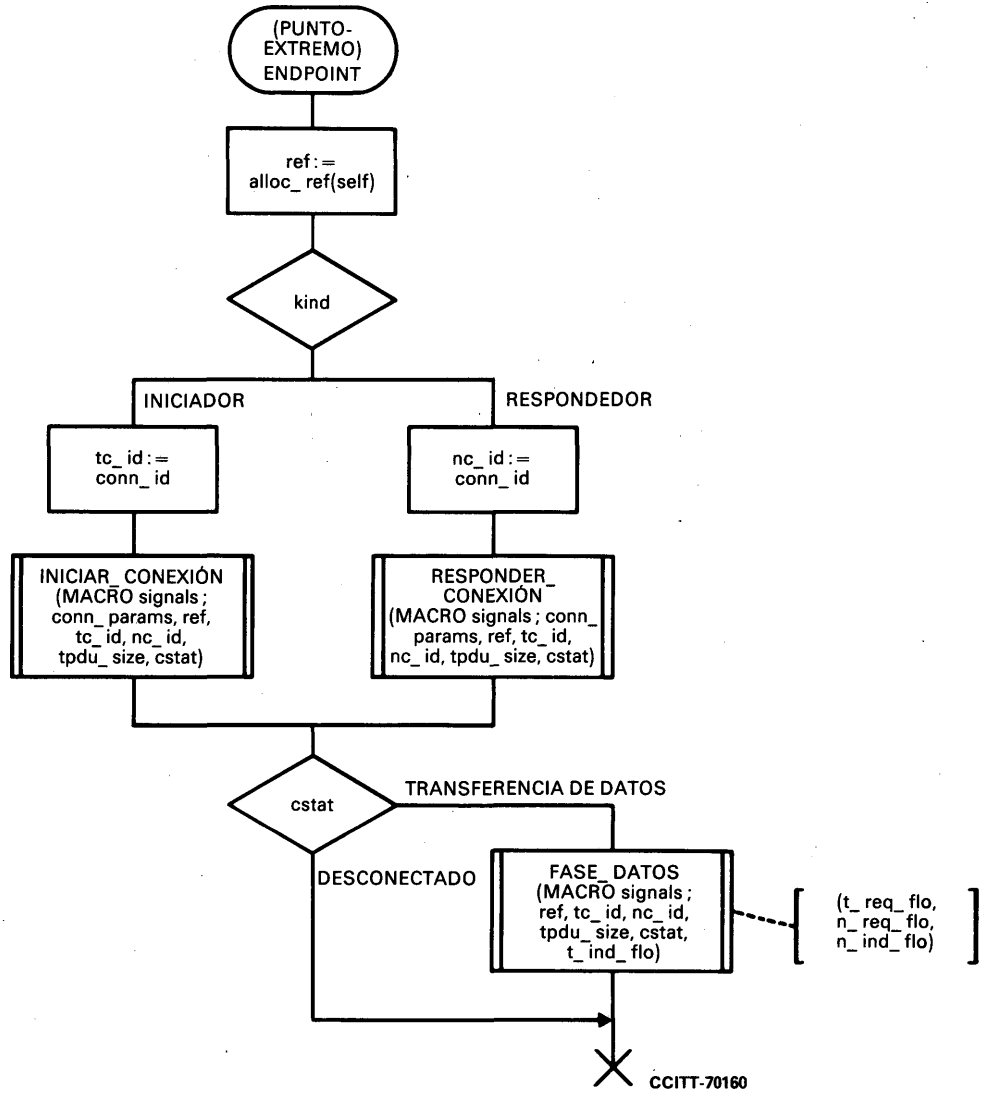


FIGURA D-226

Diagrama del procesos para ENDPOINT

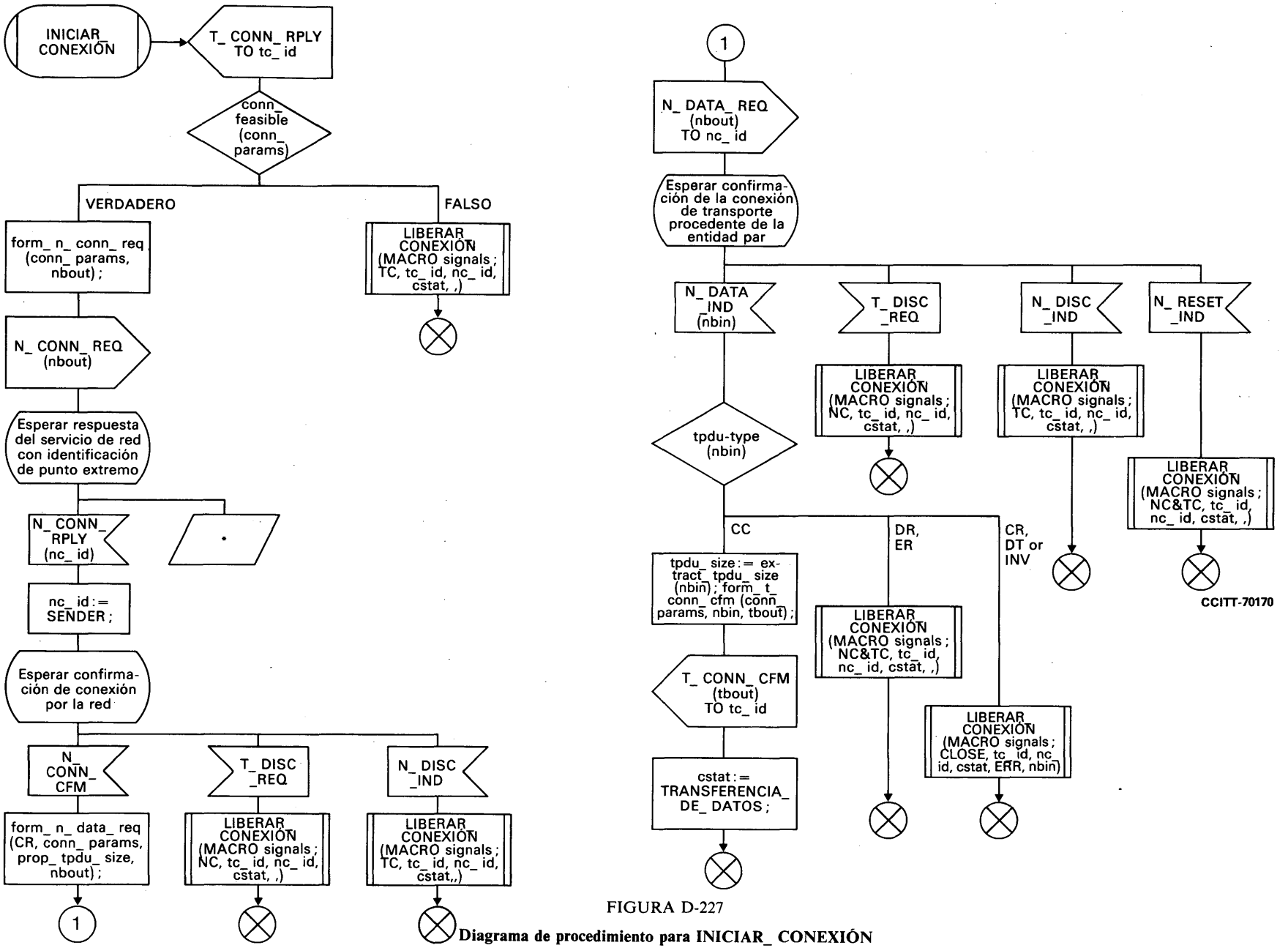


FIGURA D-227

Diagrama de procedimiento para INICIAR_ CONEXIÓN

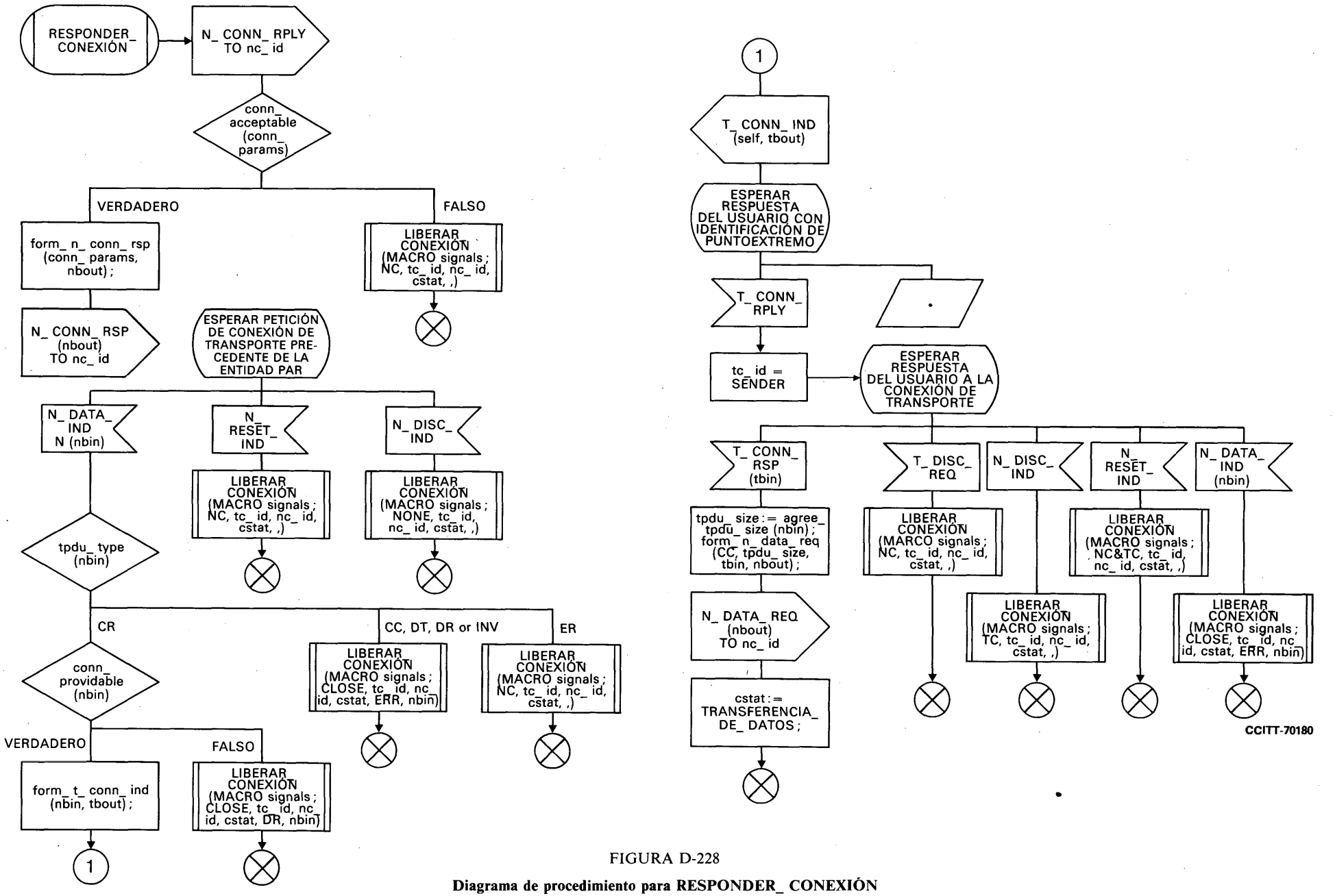


FIGURA D-228

Diagrama de procedimiento para RESPONDER_ CONEXIÓN

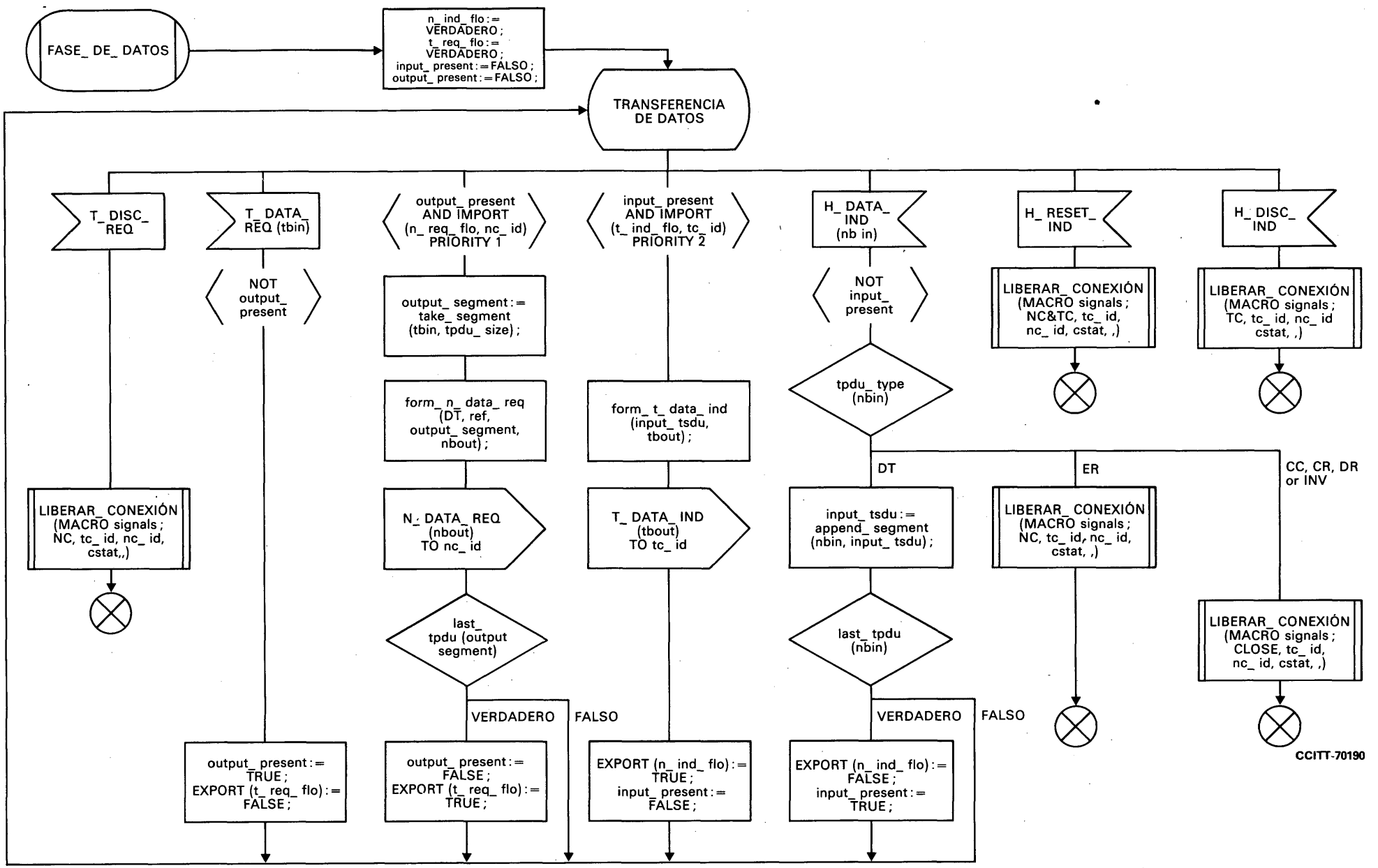


FIGURA D-229

Diagrama de procedimiento para la FASE_ DE_ DATOS

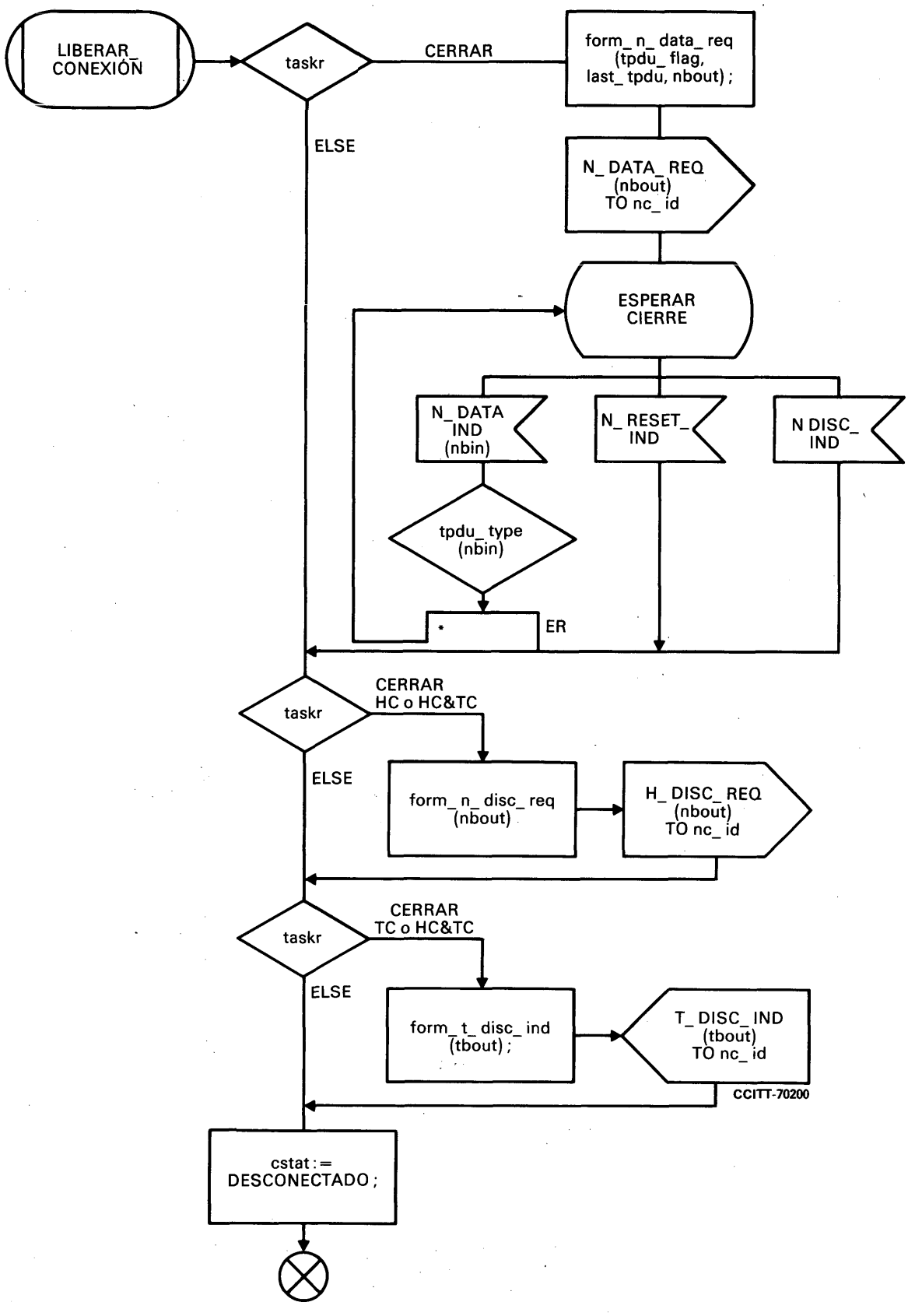


FIGURA D-230

Diagrama de procedimiento para LIBERAR_ CONEXIÓN

D.10 *Instrumentos para el LED*

D.10.1 *Introducción*

En este punto se describe un conjunto de instrumentos que pueden utilizarse para el LED. Estos instrumentos se pueden aplicar a la generación de documentos, diagramas LED (forma GR) o listas (forma PR), y/o la validación de representaciones LED.

Estas directrices no contienen la enumeración completa de los posibles instrumentos. Los instrumentos requeridos son función de la metodología escogida por el usuario.

En principio, el LED se puede utilizar sin ningún tipo de instrumento. Sin embargo, la complejidad propia de los sistemas modernos hace que las representaciones LED sean a menudo complicadas. Por consiguiente, se requieren instrumentos automáticos que faciliten la gestión de la especificación, diseño y documentación de muchos sistemas. Por ejemplo, la complejidad y el costo del trazado manual y la eventual actualización de la documentación relativa a una central de conmutación se podrían reducir considerablemente utilizando ayudas adecuadas.

Teniendo en cuenta estas consideraciones, se ha concebido el LED de forma que se puedan utilizar eficazmente instrumentos que faciliten su empleo.

D.10.2 *Clases de instrumentos*

Los instrumentos del LED se pueden clasificar según las actividades efectuadas durante la producción de documentación LED, por ejemplo:

* Instrumentos para entradas

Según las formas sintácticas, existen ayudas a las entradas para las formas gráficas de frases textuales y pictográficas del LED.

* Instrumentos para verificación sintáctica

Incluyen analizadores de sintaxis para cada una de las tres sintaxis.

* Instrumentos para la generación de documentos

Una vez que los documentos LED están almacenados en una máquina, los instrumentos pueden tener acceso a la misma y reproducirlos, utilizando posiblemente diversos periféricos. Éstos pueden emplear una forma sintáctica diferente de la utilizada para la entrada del documento. Además, los instrumentos pueden estar en condiciones de generar nuevos tipos de documentos derivados de los que se entraron originalmente.

* Instrumentos para el modelado y análisis de sistemas

Los documentos LED que representan un sistema se pueden utilizar para producir un modelo abstracto del sistema. Se pueden efectuar pruebas con dicho modelo, para la búsqueda de conflictos, para la comparación entre diversos modelos de un mismo sistema (por ejemplo, entre una especificación y una descripción), para la ejecución de una simulación del comportamiento del sistema, etc.

* Instrumentos que pueden utilizarse para la generación de códigos

Se pueden utilizar representaciones LED muy detalladas para facilitar la preparación del soporte lógico. Los instrumentos se pueden preparar de tal forma que pueda realizarse una secuencia orientada semiautomática en código.

La clase indicada a continuación constituye un tipo de instrumento específico pero útil:

* Instrumentos para la capacitación en LED

Estos instrumentos se pueden presentar por separado o integrados con otros. La integración permite facilitar ayuda, cuando es necesario, a los utilizadores de otras funciones.

Teniendo en cuenta que el LED se utiliza en muchas de las diversas fases del ciclo de vida útil de los sistemas, puede concebirse fácilmente una utilización para todos los tipos de instrumentos en el entorno de un proyecto integrado.

D.10.3 *Entrada de documentos*

Para la entrada de la forma de frases textuales del LED no se requieren requisitos especiales, ya que la sintaxis PR es equivalente, desde el punto de vista de la entrada, a cualquier entrada de cadenas de caracteres. Por consiguiente, se pueden utilizar los mismos instrumentos (editores de cadenas de caracteres). Sin embargo, las otras dos sintaxis suponen una capacidad de tratamiento gráfico.

Es evidente que en tanto que el apoyo para la entrada PR puede ser ventajoso, el apoyo para las entradas GR o PE es esencial si se tiene la intención de utilizar estas sintaxis como medio de entrada.

La entrada de documentos GR/PE se puede examinar conjuntamente, puesto que ambas sintaxis utilizan gráficos. Se requiere un editor gráfico y un dispositivo que produzca diagramas en forma gráfica. No se requiere un dispositivo de entrada gráfico ya que es posible proceder a la entrada de un diagrama utilizando una retícula predefinida. Cada símbolo se puede asociar a una cadena determinada. Por ejemplo, es posible entrar un estado indicando su posición en una retícula (par de coordenadas, número de casilla, etc.) y el tipo (estado) como una cadena de caracteres.

Un dispositivo de entrada gráfico puede facilitar información inmediata al usuario. No obstante, el «método de retícula» se puede aplicar más rápidamente y con mayor facilidad.

Se requiere en todos los casos un editor gráfico para funciones como la conexión de dos símbolos, el desplazamiento de un conjunto de símbolos a otra parte de la hoja o a otras hojas, y la supresión concatenada (la supresión de un símbolo implica la supresión de la conexión a dicho símbolo). Como para los instrumentos PR los instrumentos de entrada GR/PE se tienen que modelar utilizando la semántica/sintaxis LED. Por consiguiente, debería impedir las conexiones no válidas e incitar al usuario a rellenar todas las partes no completadas, etc.

Los instrumentos se tienen que enfrentar con diversos problemas que derivan de las limitaciones físicas de los dispositivos gráficos, tales como la «resolución». Es casi imposible disponer de un número suficiente de caracteres, que sean legibles y además permitan la presentación de un número razonable de símbolos en la pantalla.

Si bien deben tenerse en cuenta las soluciones del tipo «ventana con zoom» (zooming window) o de «desfile» (scrolling), las mismas no son completamente satisfactorias. Una gran resolución no se tiene que considerar obligatoria si los diagramas son copiados por el usuario, pero es muy conveniente si los diagramas son producidos directamente por el usuario. Por el mismo motivo (requisito de una visión global y de cierta cantidad de detalles) una alta resolución es conveniente en la presentación de diagramas.

Los instrumentos para facilitar la entrada PR pueden ser útiles: pueden incitar al usuario a utilizar palabras claves PR esperadas, indicando que una determinada palabra clave de cierre falta todavía (por ejemplo, fin de decisión, fin de estado, etc.).

Pueden formar inmediatamente el PR según las palabras clave recibidas, insertar automáticamente delimitadores y presentar al usuario claves de función orientadas al PR, etc.

La aplicación de esos instrumentos se puede basar en editores de cadenas de caracteres existentes, que pueden ampliarse insertando las características arriba mencionadas.

D.10.4 *Verificación de documentos*

Una vez que los documentos están almacenados en una máquina, la siguiente etapa es su verificación. En primer lugar se tienen que verificar individualmente y seguidamente combinar diagramas conexos y verificarlos hasta que se haya logrado la comprobación del sistema completo.

Si la entrada se ha efectuado mediante un instrumento basado en el LED, se puede ya haber efectuado una gran parte de la verificación de cada documento individual.

Todos los errores derivados de operaciones «no posibles» (por ejemplo, entradas o conservaciones que sigan a cualquier elemento que no sea un estado) deben detectarse y corregirse durante la fase de entrada. No obstante, la detección de ciertos errores sólo es posible una vez que se ha completado la fase de entrada, tanto en un solo documento como, desde luego, en el caso de discrepancias entre documentos.

Hay varias reglas LED que se pueden verificar automáticamente. Por instancia, el requisito de que todas las salidas tengan su correspondiente entrada.

En el caso de una representación de múltiples niveles se puede verificar en cierta medida la coherencia entre niveles.

El modelo LED formal se puede utilizar para derivar un conjunto de procedimientos de verificación.

D.10.5 *Reproducción de documentos*

Los documentos LED almacenados en una máquina se tienen que poder recuperar, presentar visualmente y reproducir. Se requieren instrumentos para todas estas actividades. Puede ser conveniente recuperar sólo una parte, o subconjuntos, de documentos. La recuperación puede estar basada en el LED por ejemplo, «buscar todos los procesos que emiten» una señal determinada o bien «en qué estados» se efectúa una determinada acción, etc. Los instrumentos para la presentación de información son particularmente interesantes cuando la información se tiene que presentar utilizando la sintaxis gráfica. Son pertinentes las mismas observaciones formuladas para la entrada de documentos en la sintaxis GR/PE. La reproducción de documentos es función del tipo de documento que hay que reproducir, de la forma en que el documento se ha almacenado y de las características del periférico de salida. Puede depender asimismo de la forma en que se efectuó la entrada. Los usuarios pueden desear la salida en una sintaxis diferente de la utilizada para entrar el documento.

Las limitaciones de salida de los periféricos tienen repercusiones en la reproducción de documentos. Por ejemplo, un diagrama demasiado grande que no pueda reproducirse en una determinada superficie de papel y, por consiguiente, tenga que distribuirse en varias hojas. Se tienen que agregar conectores y referencias. Puede ser conveniente establecer una diferencia entre una «adición» efectuada por el instrumento y las características de entrada originales. Otras limitaciones físicas pueden dificultar la salida de toda la información disponible, por ejemplo, el tamaño de un determinado símbolo es demasiado pequeño para contener todo el texto asociado. Pueden aplicarse en este caso diversos métodos, quizás a juicio del usuario. Figuran entre ellos el aumento de tamaño del símbolo, el recorte del texto, el recorte del texto añadiendo el texto completo como nota al pie de la página, la inserción del texto fuera del símbolo... Es conveniente disponer de instrumentos con cierta flexibilidad en cuanto al formato de salida: estas características incluyen tamaños de símbolos diferentes, formatos de salida diferentes, representación vertical u horizontal, etc.

Tiene que ser posible en todos los casos reproducir exactamente un documento en la misma forma en que se efectuó su entrada.

D.10.6 *Generación de documentos*

Partiendo de los documentos LED entrados por los usuarios y almacenados en la máquina, se pueden generar automáticamente otros varios documentos, entre los que figuran:

- Listas de señales, organizadas por proceso, por bloque o por sistema.
- Diagramas de interacción de procesos, mostrando las interacciones y las siguientes secuencias de acciones en procesos en comunicación.
- Diagramas globales de estado, mostrando el gráfico de proceso como un conjunto de estados conectados por arcos que representan las transiciones.
- Cuadro de referencias, organizado por proceso, por bloque o por sistema.
- Diagrama de partición, mostrando la estructura de los bloques y niveles.
- Comportamiento del sistema, como respuesta a secuencias de acciones del entorno.
- Índices: los documentos generados se tendrán que reproducir, y por ello las mismas consideraciones antes expuestas son válidas.

Los documentos LED entrados en forma GR se pueden traducir automáticamente a la forma PR equivalente y viceversa.

A fin de producir una forma PR correcta, todos los diagramas GR que representan procesos de un bloque se tienen que considerar conjuntamente con el diagrama de interacción de bloques GR asociado.

Se tienen que tener en cuenta las siguientes consideraciones:

- La forma GR contiene información visual que no se puede traducir a la forma PR (no existe en PR). Por ejemplo, las coordenadas del símbolo no tienen ningún significado en la forma PR.
- Se pueden eliminar los conectores que enlazan líneas de flujo de hojas diferentes.

No obstante, la traducción inversa, de PR a GR, es mucho más compleja y es probable que no satisfaga completamente a todos los eventuales lectores. Fuera de la indentación no hay criterios subjetivos con respecto a la presentación de la forma PR, en tanto que para la forma GR existe una amplia variedad de criterios subjetivos.

Debido a la representación de la forma GR en dos dimensiones, se pueden suprimir algunas etiquetas insertadas a causa de la estructura secuencial del PR, ya que es suficiente una línea de conexión. De una forma PR se pueden derivar dos representaciones GR diferentes, a saber, el diagrama de interacción de bloques funcionales y el diagrama de procesos (desde luego, si el PR representa un solo proceso, sólo se puede derivar el diagrama de proceso).

Por regla general, la traducción genera un modelo de los diagramas GR. Este modelo contiene toda la información necesaria para que un instrumento pueda formar y reproducir el diagrama en un dispositivo gráfico.

Hay que tener en cuenta que dos instrumentos diferentes que efectúen la traducción de PR en GR pueden arrojar dos representaciones GR de estructura diferente. Las representaciones GR obtenidas de esta forma son en ambos casos correctas, a condición de que mantengan la semántica expresada en la representación original.

D.10.7 *Modelado y análisis de sistemas*

Los documentos LED, tanto si especifican como si describen un sistema, son esencialmente un modelo de dicho sistema.

Este modelo, cuya función principal es transferir información de una persona a otra, puede también ser interpretado por instrumentos, verificar su coherencia, si está completo (este aspecto puede no satisfacerse en casos de especificación que tengan por objeto especificar sólo algunas partes del sistema) así como la corrección y respeto de las reglas LED (en la forma descrita en el punto relativo a la verificación de documentos).

Además, se pueden desarrollar instrumentos que permitan utilizar el modelo para simular el comportamiento funcional de los sistemas. El simulador puede interactuar con el entorno y formular conclusiones sobre la respuesta del modelo a las previsiones de los usuarios.

Si se agrega información adicional para indicar el tiempo consumido para ejecutar cada acción y para dimensionar los recursos disponibles (colas de espera, instancias, etc.), la simulación permite también estudiar la capacidad del sistema.

Se pueden preparar instrumentos para crear un modelo del entorno, partiendo del modelo del sistema, para crear secuencias significativas que permitan verificar el sistema real. Los análisis de proyectos pueden detectar las incoherencias del modelo.

El modelo de sistemas se puede también utilizar como documentación en línea. Si existen enlaces apropiados entre el sistema real y la documentación almacenada, se puede desarrollar un instrumento que permita trazar eventos del sistema en tiempo real en el modelo.

Para conseguirlo, debe existir una correlación entre los eventos físicos, vistos por el sistema, y los eventos lógicos tratados mediante la documentación LED. Si la documentación se ha organizado en varios niveles de abstracción, el usuario puede escoger el nivel que hay que trazar. Esto puede ser muy útil ya que permite a usuarios con conocimientos y capacitación diferentes investigar las actividades del sistema.

Los instrumentos que interpretan el modelo LED se pueden asimismo utilizar para destacar diferencias de comportamiento de diferentes modelos del mismo sistema. Pueden además utilizarse para comparar descripciones de sistemas diferentes (sistemas producidos por empresas diferentes) o para comparar la especificación del sistema con la descripción de sistema se ajusta a la especificación original.

D.10.8 *Generación de códigos*

Las provisiones de una sintaxis definida formalmente y de una definición matemática formal del LED, permiten realizar instrumentos que puedan mapear la semántica de representaciones LED con la semántica de lenguajes de programación. Es posible que tales instrumentos no estén en condiciones de facilitar programas completos que puedan utilizarse, pero pueden ser muy útiles para proporcionar como mínimo el marco para un programa real.

El § D.8.1 de las presentes Directrices para el Usuario contiene un ejemplo de cómo puede realizarse el mapeado entre las construcciones LED y CHILL.

D.10.9 *Capacitación*

Se ha elaborado un curso de capacitación completo sobre el LED, que comprende unas 200 páginas de texto y una colección de diapositivas (unas 200). El curso abarca todos los aspectos del lenguaje y proporciona también ejemplos, así como algunas sugerencias sobre el uso del LED.

El curso sobre el LED puede solicitarse a la Sección de Ventas de la Secretaría General de la Unión Internacional de Telecomunicaciones – Place des Nations, CH-1211 Genève 20 (Suiza).

Impreso en Suiza — ISBN 92-61-02243-X