UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

CCITT

COMITÉ CONSULTIVO
INTERNACIONAL
TELEGRÁFICO Y TELEFÓNICO

LIBRO ROJO

TOMO VI – FASCÍCULO VI.10

LENGUAJE DE ESPECIFICACIÓN Y DESCRIPCIÓN FUNCIONALES (LED)

RECOMENDACIONES Z.100 A Z.104

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

Ginebra 1985

UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

CCITT

COMITÉ CONSULTIVO
INTERNACIONAL
TELEGRÁFICO Y TELEFÓNICO

LIBRO ROJO

TOMO VI – FASCÍCULO VI.10

LENGUAJE DE ESPECIFICACIÓN Y DESCRIPCIÓN FUNCIONALES (LED)

RECOMENDACIONES Z.100 A Z.104

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

Ginebra 1985

ISBN 92-61-02233-2

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.10 DEL LIBRO ROJO

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.

FASCÍCULO VI.10

Recomendaciones Z.100 a Z.104

LENGUAJE DE ESPECIFICACIÓN Y DESCRIPCIÓN FUNCIONALES (LED)

PAGE INTENTIONALLY LEFT BLANK

PAGE LAISSEE EN BLANC INTENTIONNELLEMENT

INTRODUCCIÓN AL LED

1 Introducción

Esta Recomendación contiene una explicación general y una introducción del Lenguaje de Especificación y Descripción (LED) del CCITT. El lenguaje se define con todo detalle en las Recomendaciones Z.101 a Z.104.

1.1 *Consideraciones generales*

El objeto de formular una Recomendación sobre el LED es proporcionar un lenguaje que permita la *especificación* y *descripción* inequívocas del comportamiento de los *sistemas* de telecomunicaciones. Las *especificaciones y descripciones* que utilicen el LED pretenden ser formales en cuanto que es posible analizarlas e interpretarlas inequívocamente.

La *especificación de un sistema* consiste en la *especificación del comportamiento funcional* y de una serie de *parámetros generales* del *sistema*. El LED pretende solamente describir los aspectos de comportamiento del *sistema*; los *parámetros generales* describen propiedades como la capacidad y el peso que es preciso describir utilizando técnicas diferentes.

Los términos *especificación* y *descripción* se utilizan con el significado siguiente:

- la *especificación* de un *sistema* es la descripción de su *comportamiento* requerido, y
- la *descripción* de un *sistema* es la descripción de su *comportamiento* real.

El LED describe el *comportamiento* en una forma estímulo/respuesta, suponiendo que tanto los estímulos como la respuesta son entidades discretas. Este enfoque se basa en los conceptos de *máquinas ampliadas de estado finito*.

El *comportamiento* del *sistema* descrito en el LED es la secuencia de respuestas a cualquier secuencia dada de estímulos vista desde fuera del *sistema*. Todo par de *sistemas* descrito en LED que tenga el mismo *comportamiento* a este respecto se considera *funcionalmente equivalente*. El concepto de *equivalencia funcional* se utiliza para comparar *sistemas* entre sí, por ejemplo, la *especificación* de un *sistema* requerido con la *descripción* de un *sistema* ofrecido. Las directrices para el usuario del LED, que figuran como apéndice a las Recomendaciones, contienen un análisis de los criterios y métodos que permiten esta comparación.

El LED proporciona también conceptos de estructuración que permiten la partición de un *sistema* a fin de poderlo definir, desarrollar y comprender parte por parte.

Inicialmente, estos conceptos resultan útiles tanto para especificar un *sistema*, cuando es posible tratar independientemente aspectos diferentes, como más adelante para describir un *sistema*, cuando las estructuras de descripción deben adaptarse a la estructura del *sistema*.

1.2 *Objetivos*

Al definir el LED se ha perseguido el objetivo general de proporcionar un lenguaje que:

- sea fácil de aprender, utilizar e interpretar en relación con las necesidades de los organismos de explotación;
- permita especificaciones y descripciones inequívocas para la presentación de ofertas y la adquisición de equipos;
- pueda aplicarse a nuevos desarrollos;
- pueda soportar varias metodologías de especificaciones y diseño de sistemas, sin asumir ninguna de ellas.

1.3 *Campo de aplicación*

El campo principal de aplicación del LED es la descripción del comportamiento de aspectos de los *sistemas* de telecomunicación. Entre las aplicaciones cabe incluir:

- el proceso de las llamadas (por ejemplo tratamiento de las llamadas, telefonía, señalización, cómputo con fines de tasación) en los sistemas de conmutación SPC;
- mantenimiento y tratamiento de las averías (por ejemplo, alarmas, reparación automática de averías, pruebas periódicas) en los sistemas generales de telecomunicación;
- control del sistema (por ejemplo, protección contra las sobrecargas, procedimientos de modificación y ampliación);
- protocolos de comunicación de datos.

El LED puede también, por supuesto, utilizarse para la descripción de cualquier comportamiento que pueda ser descrito utilizando un modelo discreto, es decir, comunicando con su entorno por medio de mensajes discretos.

2 Estudio del lenguaje

El estudio siguiente del LED constituye una introducción a las Recomendaciones Z.101 a Z.104. Las explicaciones ofrecidas no son definiciones formales, excepto las del § 2.1, sino solamente explicaciones aclaratorias.

2.1 *Algunas definiciones básicas*

En las Recomendaciones Z.100 a Z.104 se utilizan algunos conceptos generales y se aceptan ciertas convenciones. Las definiciones son las siguientes:

Tipo, definición e instancia

En las Recomendaciones, una entidad está estrictamente separada de su *definición.* Se utilizan el esquema y terminología definidos a continuación e ilustrados en la figura 1/Z.100.

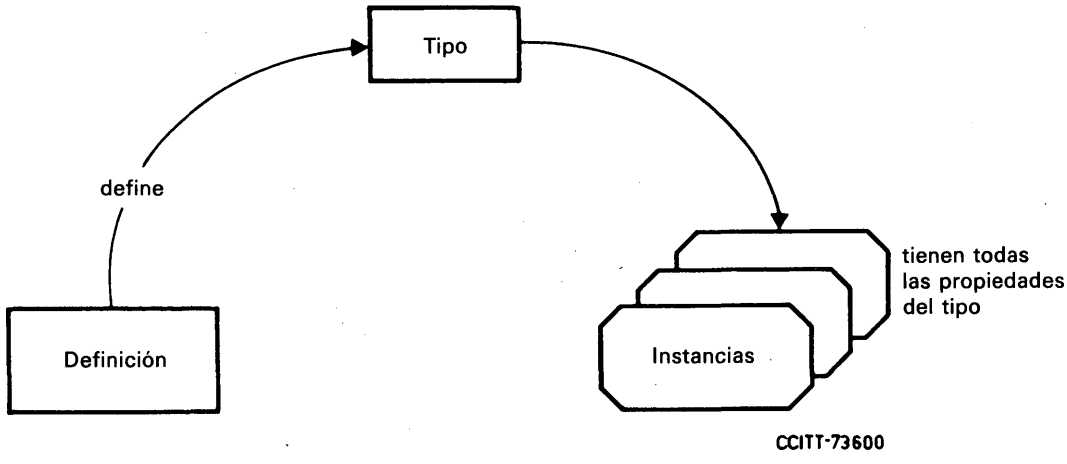


FIGURA 1/Z.100

El concepto de tipo

Un *tipo* se define por su *definición,* que define todas las propiedades asociadas a dicho *tipo.* Un *tipo* puede ser *instanciado* en cualquier número de *instancias.* Cualquier *instancia* de un *tipo* particular tiene todas las propiedades definidas por el *tipo.*

El esquema se aplica a varios conceptos del LED, es decir, tenemos *definiciones de sistemas* e *instancias de sistema, definiciones de proceso* e *instancias de proceso.*

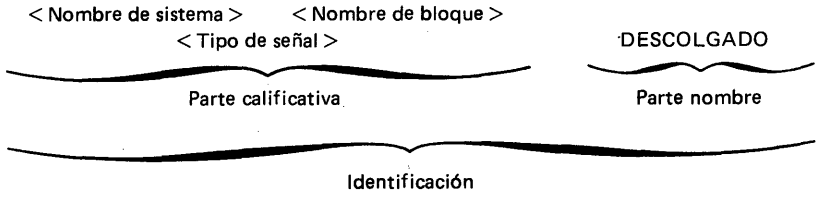
Las *instancias* de un *tipo* pueden ser *tipos* en sí mismas. Las propiedades asociadas a los *tipos* son heredadas de acuerdo con la aplicación jerárquica, es decir, las propiedades del *sistema* de tipo genérico se heredan por todas las *instancias* de *definiciones de sistemas* específicos.

Para evitar textos engorrosos, se ha convenido en que siempre que sea evidente del contexto al que se refiera la *definición* o la *instancia* de un *tipo,* se suprime el atributo.

Nombres

Para identificar *definiciones* e *instancias* se han adoptado los siguientes nombres y terminología.

Todos los *identificadores* utilizados en una representación son únicos. La identificación de una entidad consta de dos partes: una parte *nombre* y una *parte calificativa:*



La *parte calificativa* deriva del contexto jerárquico completo en que está definida la entidad y del *tipo* de la entidad. La *parte nombre* debe ser un *nombre* significativo en relación con el objeto o efecto de la entidad nombrada.

Cuando la identificación aparece en la sintaxis concreta, los *calificativos* evidentes pueden ser solamente implícitos. Por ejemplo, cuando no es equívoca sólo se debe utilizar la *parte nombre.*

Reglas de visibilidad

Cuando se introduce un *identificador* en una *especificación* o *descripción,* dicho *identificador* es *visible* en el punto de introducción y en niveles directamente por debajo del punto de introducción en la jerarquía de *especificación* o *descripción.* El nivel más elevado en que es visible el *identificador* se indica por medio del nivel jerárquico más bajo contenido en la *parte calificativa* del *identificador.*

Error de especificación

En aquellos casos en que las propiedades de una *especificación* LED son incoherentes o ambiguas, dicha *especificación* es *inválida.* Cuando la interpretación de una *especificación* LED viola propiedades de una *especificación* que es válida, dicha interpretación de *sistema* se halla en condición de *error.* Una interpretación de un *sistema* que conduzca a un error significa que el futuro comportamiento del *sistema* no puede predecirse tomando como base la especificación.

2.2 *El LED básico*

El *comportamiento* dinámico de un *sistema* LED viene generado por *instancias de proceso,* que actúan concurrentemente. Un *proceso* se modela como una *máquina ampliada de estado finito.* Sólo actuará en respuesta a estímulos exteriores discretos y puede entonces enviar respuestas discretas a su *entorno.* Otros *procesos* pueden aceptar las respuestas generadas como estímulo.

En LED, un *proceso* esperará en un estado hasta que reciba una *señal* válida de su *entorno.* A continuación hará una *transición* a otro *estado.* Durante la *transición* puede realizar *acciones* que pueden consistir en la manipulación de información local al *proceso* o en el *envío de señales* a otros *procesos* o al *entorno* externo del sistema.

Todos los *procesos* en un *sistema* o en su *entorno* tienen acceso al «tiempo absoluto», y pueden realizar mediciones de tiempo y temporización.

Las *instancias de proceso* se pueden *crear* y *terminar* dinámicamente. Una *instancia* se puede crear por otra *instancia* o puede existir en el momento de inicialización del *sistema.* Una *instancia de proceso* sólo se puede *terminar* por una *acción* explícita *de parada* realizada por sí misma.

Puede haber varias *instancias de señal* en espera de ser aceptadas por un *proceso.* Para manipular esta cola de *señales,* se asocia un *puerto de entrada* a cada *instancia de proceso.* La disciplina de la cola es básicamente del orden «primero en entrar, primero en salir», pero el *proceso* puede en sí manipular esta disciplina por un *conjunto de señales de excepción* asociado a un *estado.*

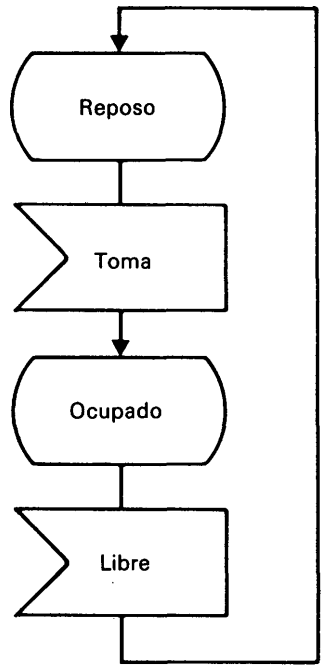


FIGURA 2/Z.100

Definición de un proceso

Las *instancias de señal* pueden transportar *valores de datos*, que estarán disponibles para la *instancia* de recepción *de proceso* cuando se reciba la *señal*. Los *valores de datos* pueden ser almacenados en *variables* locales del *proceso* y recuperados de los mismos.

La *estructura* estática de un *sistema* LED se describe en términos de *sistema*, *bloques* y *canales*, como se muestra en la figura 3/Z.100.

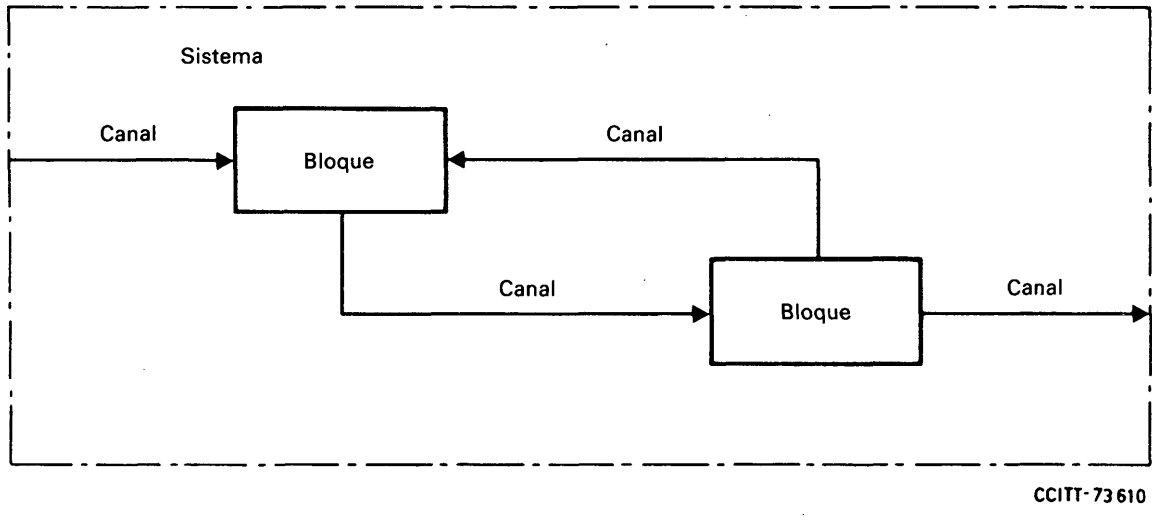


FIGURA 3/Z.100

Estructura estática de un sistema LED

El *sistema* es una composición de *bloques* conectados entre sí y a la *frontera del sistema* por *canales*. La *frontera del sistema* separa al *sistema* de su *entorno*. Se supone que el *entorno* se comporta «a la manera del LED». Es decir, envía *señales* al *sistema* y acepta respuestas del *sistema* en forma de *señales*.

Los *canales* actúan como medio de transporte de las *señales* entre *bloques* y entre *éstos* y la *frontera del sistema*. Los *canales* son unidireccionales. Los bloques son contenedores de los *procesos* y sirven para estructurar el *sistema* en bloques de construcción.

2.3 *Los datos en el LED*

Los procesos LED pueden retener y manipular *valores de datos*. Los *valores de datos* están *ligados* a *variables*, que son locales a un *proceso*. Sin embargo, se pueden enviar *valores de datos* entre *procesos*, y desde y hacia el *entorno*, por medio de *señales*. Las *variables* locales para un *proceso* se definen en la *definición del proceso*.

Los datos tratados se *mecanografían*. En el LED se define un conjunto de *tipos de datos predefinidos*. Además, el usuario puede definir nuevos *tipos de datos*. Los *tipos de datos predefinidos* son:

- *Booleanos* — Tienen los *valores* VERDADERO y FALSO y las operaciones lógicas normales en estos *valores*.
- *Enteros* — Tienen su significado matemático normal.
- *Naturales* — Tienen el significado matemático normal de los números naturales.
- *Reales* — Tienen el significado matemático normal de los números reales.
- *Carácter* — Tienen (como en los lenguajes de programación) el significado normal y los *valores* y representación del alfabeto N.º 5 del CCITT.
- *Matriz* — Colección indexada de elementos, todos ellos para el mismo *tipo de datos*. Se trata de una ampliación del significado normal de matriz, utilizado en lenguajes de programación.
- *Estructura* — Composición de una serie de *tipos de datos* que pueden diferir entre sí.
- *Cadena* — Lista de elementos de cualquier *tipo de datos* y de longitud arbitraria.
- *Cadena de caracteres* — Lista de caracteres de longitud arbitraria. Se observará que una *cadena de caracteres* es una *cadena* formada a base de la *cadena de tipo de datos* con el *carácter de tipo de datos*.
- *Conjuntista* — Tiene el significado matemático normal correspondiente a un *valor conjuntista* en un conjunto ordenado.
- *PId* — Se utiliza para identificar *instancias de proceso*.
- *Tiempo* — Tiempo absoluto.
- *Duración* — Intervalo entre dos instantes en tiempo absoluto.
- *Temporizador* — *Tipo de datos* utilizado para temporizadores y que define los operadores INICIALIZAR y REINICIALIZAR para variables de TEMPORIZADOR.

El concepto STRUCT permite construir otros tipos de datos con valores complejos compuestos por una serie de tipos de datos (posiblemente diferentes).

Los *tipos de datos* definidos por el usuario pueden estar definidos según los *tipos de datos predefinidos* junto con restricciones tales como el *alcance* de un *entero*, o pueden ser *tipos de datos abstractos* adicionales. Los *tipos de datos abstractos* se definen utilizando un método «axiomático».

Los *tipos de datos* pueden ser genéricos para un *sistema* o estar contenidos en una *definición de bloque*, una *definición de canal* o una *definición de proceso*. La *definición* es *visible*, y por consiguiente utilizable, dentro del campo de aplicación del objeto en que están contenidos. Por ejemplo, si una *definición de un tipo* está contenida en un *bloque*, los *elementos de datos* de ese *tipo* pueden ser utilizados en todos los *procesos* contenidos en el *bloque* y en los *sub-bloques* del *bloque*.

2.4 *Conceptos estructurales en el LED*

En el LED se proporcionan conceptos estructurales para facilitar las descripciones de *sistemas* complejos y/o grandes. Los conceptos se definen de forma tal que pueden soportar:

- La *partición* de grandes descripciones en módulos que permitan abordar y comprender independientemente las partes.
- La descripción de un *sistema* o partes de un *sistema* en varios *niveles de abstracción*.
- La descripción de la estructura real de un *sistema*.

Cuando se utilizan estos conceptos estructurales conviene aclarar si representan la *estructura* real o la necesaria, o si se utilizan únicamente para facilitar la representación.

En el LED básico, un sistema está compuesto de *bloques, canales* y *procesos.* Estos componentes pueden estar adicionalmente estructurados, es decir, los *bloques* en *sub-bloques* y *sub-canales,* los *canales* en *bloques* y *canales* y los *procesos* en *sub-procesos.* Una *definición de proceso* puede estar también más detallada en varios números de etapas utilizando *procedimientos* y *macros.*

Al *participar bloques* en *sub-bloques* y *sub-canales* se obtiene una descripción jerárquica del *sistema* en varios niveles. La información contenida en los *niveles* superiores debe estar contenida en los interfaces de los *niveles* inferiores. La estructura obtenida puede representarse por un *diagrama en árbol de bloques* como se muestra en la figura 4/Z.100.

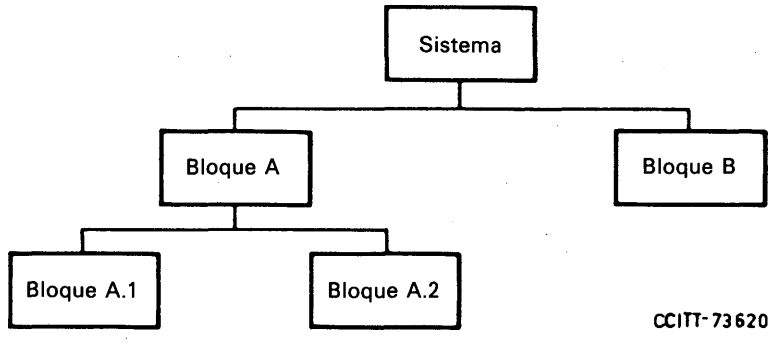


FIGURA 4/Z.100

Diagrama en árbol de bloques

Puede también describirse con mayor detalle por medio de un *diagrama de interacción de bloques,* como se muestra en la figura 5/Z.100:

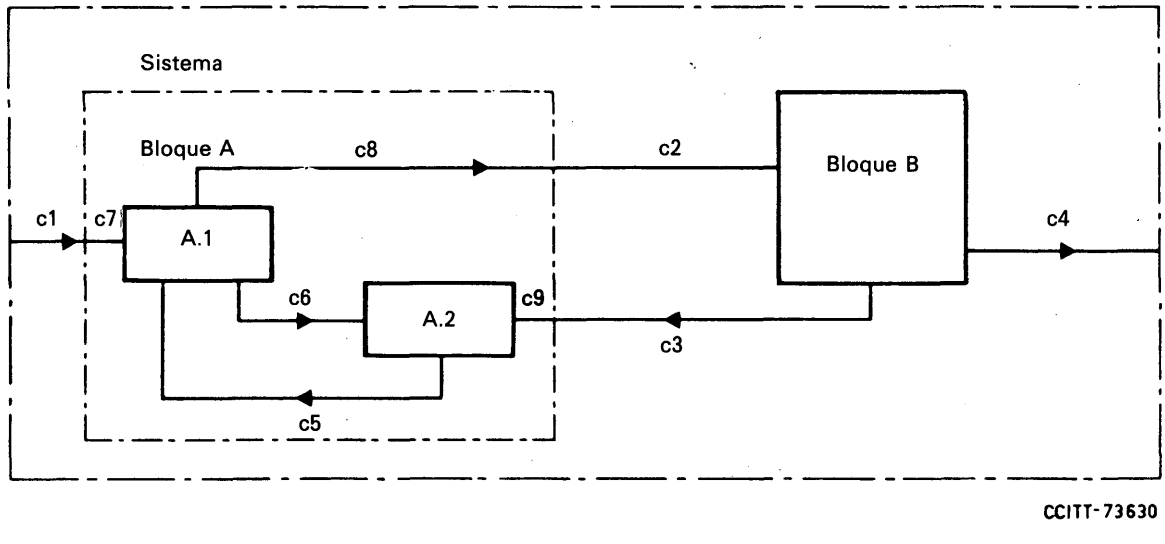


FIGURA 5/Z.100

Diagrama de interacción de bloques

Este último diagrama muestra también los *canales* y *sub-canales* que conectan la estructura de *bloques* y *sub-bloques*. Utilizando esta *partición de bloques*, lo único necesario para que la descripción sea significativa es que los *sub-bloques* más detallados (los «bloques hoja» del *diagrama en árbol de bloques*) contengan *procesos*.

La partición de *canales* en *canales* y *bloques* dará también como resultado una estructura jerárquica. La figura 6/Z.100 contiene un sencillo ejemplo:

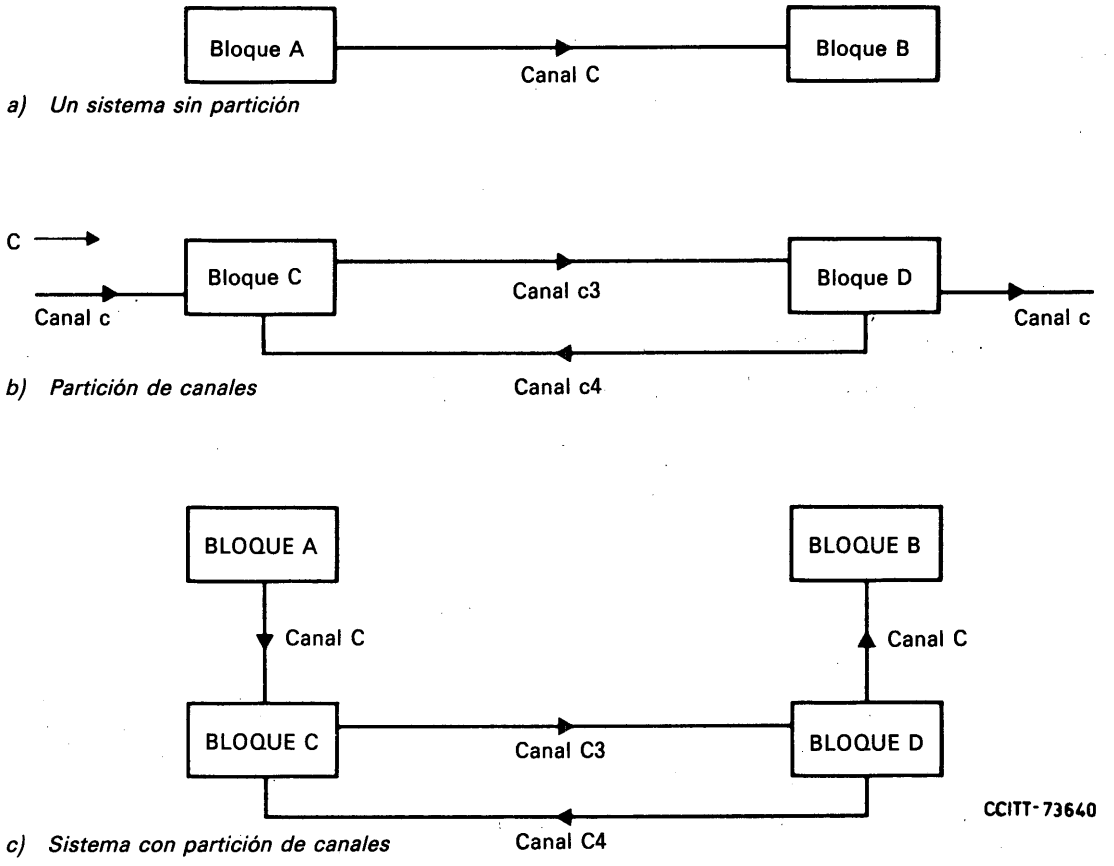


FIGURA 6/Z.100

Partición de canales

La *partición* de un proceso en una serie de *sub-procesos* guarda relación con la *partición de bloques*, pues los *sub-procesos* de un *proceso* deben estar siempre situados en los *sub-bloques* del *bloque* que contiene el *proceso*. Esto se muestra en la figura 7/Z.100.

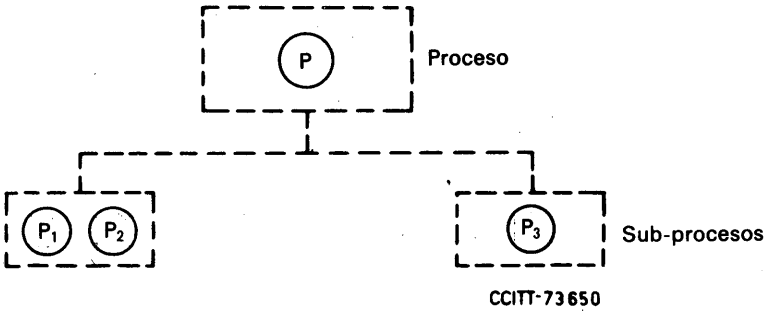


FIGURA 7/Z.100

Partición de procesos

Los *sub-procesos* de un *proceso* representan juntos una descripción alternativa más detallada del comportamiento descrito por el *proceso.* Así pues, cuando se tiene una descripción como la de la figura 7/Z.100, es preciso elegir si se debe interpretar el *comportamiento* detallado, representado por los *sub-procesos,* o el menos detallado, representado por el *proceso.* En el LED las descripciones alternativas soportan diferentes *niveles de abstracción.*

La *partición* de un *proceso* se representa por un *diagrama en árbol de procesos,* como se muestra en la figura 8/Z.100.

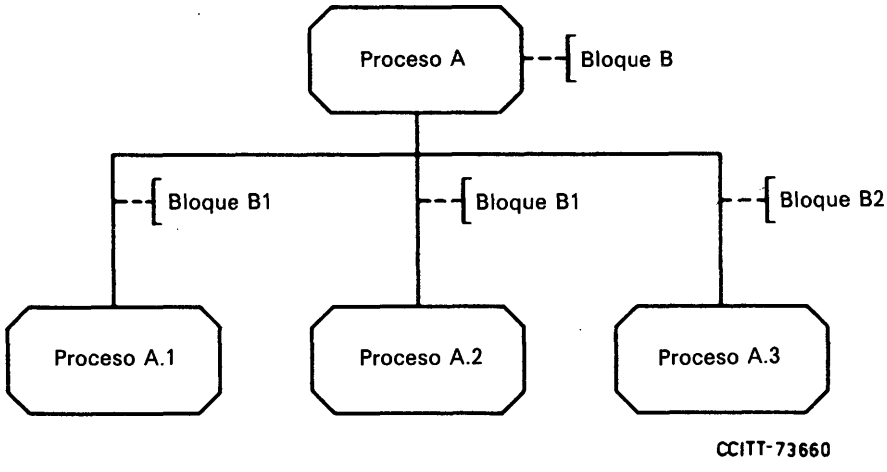


FIGURA 8/Z.100

Diagrama en árbol de procesos

Un *proceso* puede también estructurarse y detallarse con la utilización de *procedimientos.* Un *procedimiento* en LED es similar a los procedimientos de los lenguajes de programación. Representa un *comportamiento* parametrizado y predefinido que puede ser invocado por cualquier *proceso* que tenga acceso a su definición. Los *procedimientos* pueden ser predefinidos en las bibliotecas o definidos por el usuario.

Un *procedimiento* se define como una serie de *acciones* y puede incluir *estados,* de manera similar a un *proceso.* La figura 9/Z.100 da un ejemplo de la utilización de procedimientos.

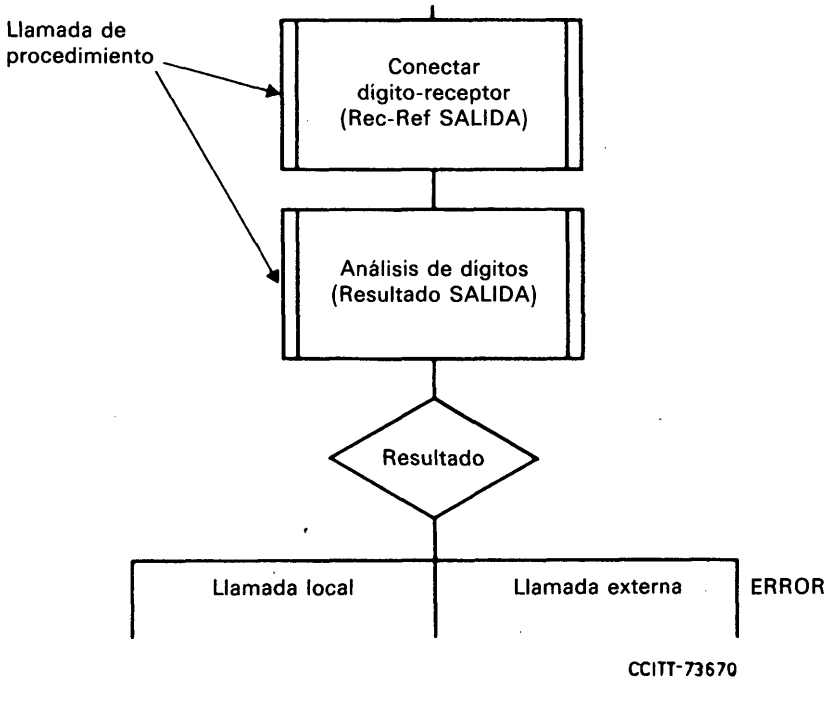
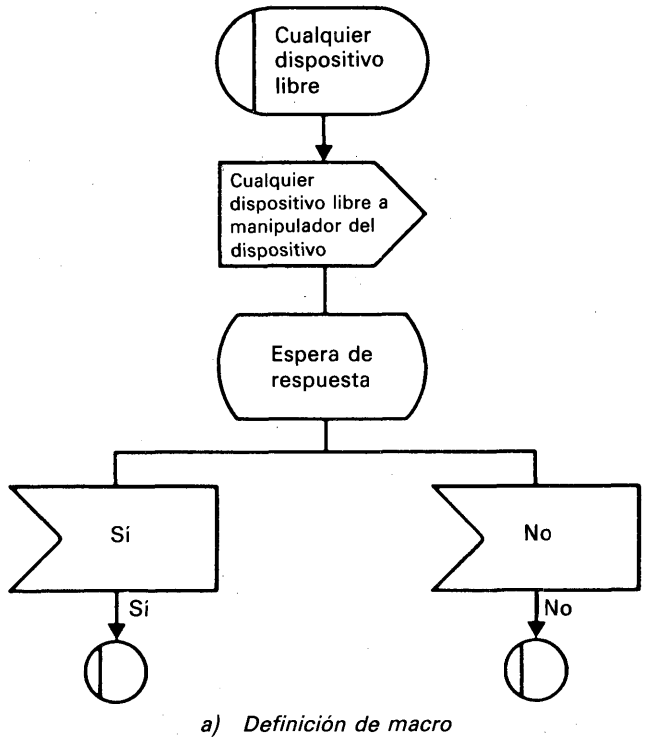


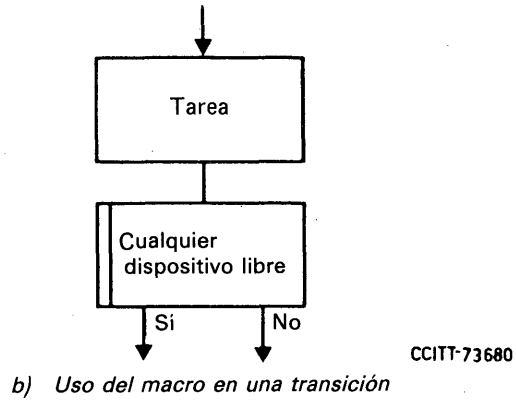
FIGURA 9/Z.100

Ejemplo de utilización de procedimientos en una transición

Las representaciones LED pueden también estructurarse por la utilización de *macros*. Un *macro* es un medio sintáctico de facilitar la escritura o el dibujo de representaciones LED y su comprensión. Un *macro* es una colección designada de elementos sintácticos, definida por el usuario. Siempre que en una descripción aparezca una referencia a un *macro*, puede entenderse como si sustituyera a los elementos en que está definida, es decir, no tiene semántica propia. Los *macros* se pueden utilizar en cualquier representación LED, por ejemplo, para representaciones de *proceso*, para diagramas estructurales, etc. La figura 10/Z.100 da un ejemplo de la utilización de un *macro*.



a) Definición de macro



b) Uso del macro en una transición

FIGURA 10/Z.100

Ejemplo de uso de un macro

Todos los conceptos estructurales del LED pueden por supuesto utilizarse en combinación.

2.5 *Operaciones compuestas*

Las *operaciones compuestas* en el LED son notaciones abreviadas típicas proporcionadas en el lenguaje para simplificar el diseño de *procesos* LED. Se definen en términos de otras operaciones LED y se deben interpretar como si fuesen sustituidos por esas operaciones.

Normalmente las *operaciones compuestas* implican *estados* ocultos e intercambio de *señales* con otros procesos, por lo que en ocasiones hay que tener cuidado con los efectos secundarios.

Las *operaciones compuestas* proporcionadas son:

Importación/exportación de *valores datos*

Notación abreviada para acceder a *valores* de *elementos de datos* locales a los otros procesos a través de un intercambio implícito de *señales.*

Condición de posibilitación

Notación abreviada para la capacidad de aceptar una *señal* como *entrada* o de *conservar* la *señal* dependiendo de una *condición* que puede contener *valores importados.* Se modela como si tuviera varios *estados* en los que la *señal* se acepta como *entrada* y otros en los que la *señal* se conserva. El *estado* implícito se escoge después de la evaluación de la condición. La operación puede implicar también un intercambio *de señales.*

Señales continuas

Notación abreviada para abandonar un *estado* y entrar en una *transición* cuando se hace cierta una condición que puede utilizar *valores importados.* Una *señal continua* tiene prioridad inferior a las *señales* «normales» y se puede utilizar para estimular una transición cuando cambia un *valor de datos* externo. La operación implica un cierto número de *estados* e interfuncionamiento de la *señal.*

Utilizando *operaciones compuestas,* se puede reducir el número de *estados* y *transiciones* en una definición de proceso.

2.6 *El concepto de opción en el LED*

Cuando se *especifican* o *describen* varias aplicaciones similares utilizando el LED es frecuente que la misma *definición de proceso* sea aplicable a varias aplicaciones sólo con leves modificaciones. El concepto de *OPCIÓN* da la posibilidad de disponer de partes opcionales en la *definición de proceso.*

Asimismo en una *especificación* permite la representación de *comportamientos* igualmente aceptables desde el punto de vista del especificador. El *sistema* real aplicará una de estas alternativas.

3 Prolegómenos de la especificación del lenguaje

La especificación del lenguaje LED está contenida en las Recomendaciones Z.101, Z.102, Z.103 y Z.104. En los prolegómenos siguientes se explican los métodos y estrategias utilizadas en la definición del lenguaje y se dan directrices sobre la forma de leer las Recomendaciones.

3.1 *Estrategia utilizada en la especificación del lenguaje*

El LED ofrece una elección entre dos formas sintácticas para representar descripciones LED: una representación gráfica (LED/GR) y una representación de frase textual (LED/PR). Como ambas son representaciones concretas de la misma semántica LED, son equivalentes desde el punto de vista semántico. Para asegurar que son equivalentes entre sí y por consiguiente transformables entre sí, la definición de la semántica del LED está estrictamente separada de las definiciones de las diferentes sintaxis concretas. Las relaciones entre la definición semántica del LED y las formas sintácticas concretas pueden representarse como se muestra en la figura 11/Z.100.

La semántica del LED se define utilizando una *sintaxis abstracta* sin representación concreta, asociada con reglas sobre interpretación y formación correcta. Esta definición se denomina modelo del lenguaje común. Cada una de las formas sintácticas concretas tiene, pues, su propia definición y la de su relación con la *sintaxis abstracta* (es decir, la forma de transformarla en la forma abstracta). Siguiendo este enfoque, sólo hay una definición de la semántica del LED; cada una de las formas de representación concreta heredará la semántica a través de sus relaciones con la *sintaxis abstracta.* Este enfoque asegura también que las formas sintácticas concretas son equivalentes. Como las transformaciones se producen en ambos sentidos, una representación se puede transformar en la otra forma a través de la *sintaxis abstracta.*

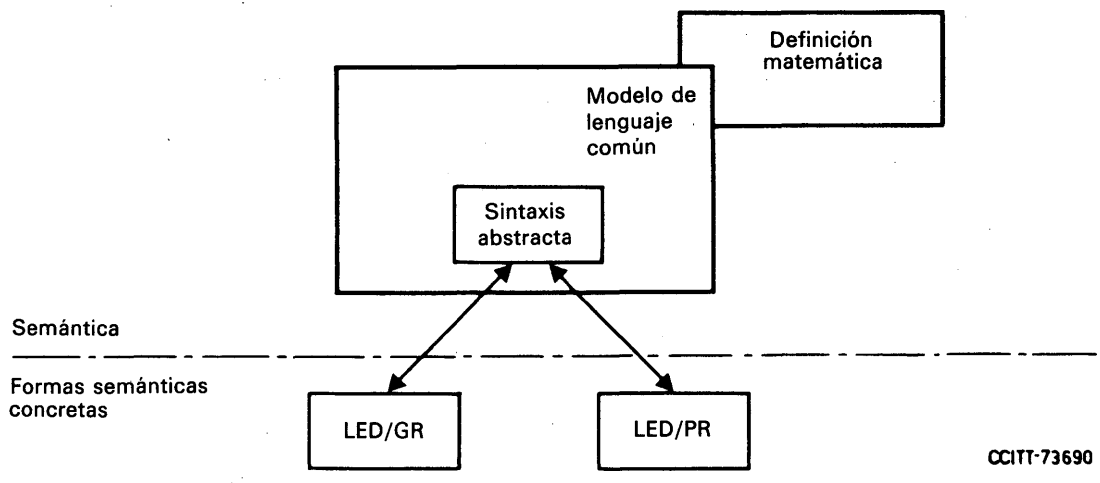


FIGURA 11/Z.100

Estructura de las especificaciones del lenguaje

Las reglas de interpretación del modelo de lenguaje común se definen de manera operacional, es decir, la definición describe cómo las instancias de los conceptos LED interpretan su definición como una «máquina abstracta LED». Se proporciona también una definición matemática del LED en semántica denotacional (pero no forma parte de las Recomendaciones). La definición matemática guarda estrecha relación con la estructura y *sintaxis abstracta* del modelo de lenguaje común.

Siguiendo esta estrategia en las Recomendaciones, para cada concepto LED se da primero una definición de la sintaxis abstracta y sus reglas, seguida de reglas sobre cómo interpretar el concepto. Por último se ofrecerán las formas sintácticas concretas de representar el concepto.

3.2 *Terminología*

Para cada concepto LED se utilizará siempre en las Recomendaciones el mismo término. Para distinguir entre los casos en que un término se refiere a un concepto LED y aquellos en que el sentido es más general, todos los términos de conceptos LED se indicarán en *negritas.*

En apéndice a las Recomendaciones figura como anexo A un glosario de términos LED. En él se da una breve explicación de cada término y una referencia al lugar donde se define.

3.3 *Definición del LED/GR*

El LED/GR se define según el esquema siguiente:

- Primero se definen la forma y contenido de los símbolos utilizados.
- A continuación y en su caso se dan reglas de conectividad, es decir, se indican las composiciones de símbolos permitidas.
- Por último se dan las relaciones con la *sintaxis abstracta* del modelo de lenguaje común.

3.4 *Definición del LED/PR*

El *LED/PR* se define utilizando *diagramas sintácticos* y reglas adicionales en lenguaje natural. Las definiciones siguen el esquema siguiente:

- Primero se define la sintaxis por diagramas y textos.
- A continuación se dan las relaciones del modelo de lenguaje común con la *sintaxis abstracta.*

3.4.1 *Diagramas sintácticos*

Un *diagrama sintáctico* consta de símbolos terminales y no terminales conectados por líneas de flujo.

Un símbolo terminal contiene un carácter o una secuencia de caracteres, y las reglas de generación determinan que cuando esos caracteres han pasado en un trayecto deben aparecer en el texto *LED/PR.*

Un símbolo no terminal es una referencia a otro *diagrama sintáctico* cuyo nombre aparece en el símbolo. Según las reglas de generación, cuando un símbolo no terminal aparece en un diagrama, el trayecto conduce al diagrama referenciado y el trayecto no abandonará el símbolo no-terminal hasta abandonar el diagrama referenciado.

Todos los símbolos terminales, símbolos no terminales y *diagramas sintácticos* tienen exactamente una línea de flujo que conduce a ellos y exactamente una línea de flujo que parte de ellos.

Los símbolos gráficos usados son los de la figura 12/Z.100.

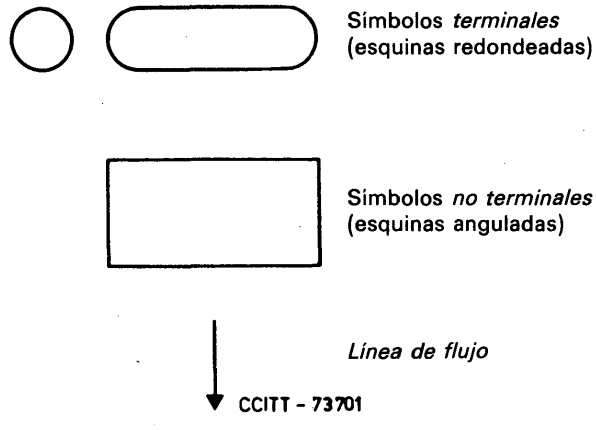


FIGURA 12/Z.100

Símbolos de los diagramas sintácticos

La definición de sintaxis LED/PR consiste, en su más alto nivel, en un diagrama de sintaxis, el sistema. Este diagrama remite, a través de no terminales, a una serie de otros diagramas. Todo trayecto que se inicie con la línea de flujo que conduce a este diagrama y que sale del diagrama generará en su trayecto un texto *LED/PR*. El texto será sintácticamente correcto si se han seguido las reglas de la sintaxis del LED.

3.5 *Elementos sintácticos comunes al LED/GR y al LED/PR*

Para determinados elementos sintácticos, se utiliza la misma sintaxis concreta en el LED/GR y en el LED/PR.

3.6 *Estructura de las Recomendaciones sobre el LED*

Esta Recomendación va seguida de otras cuatro Recomendaciones (Z.101 a Z.104). Se pretende guiar al usuario a seleccionar un subconjunto de LED apropiado para sus aplicaciones y metodología.

El LED se puede aplicar de muchas formas, para diferentes propósitos y métodos. El subconjunto mínimo LED que se puede escoger se describe en la Recomendación Z.101.

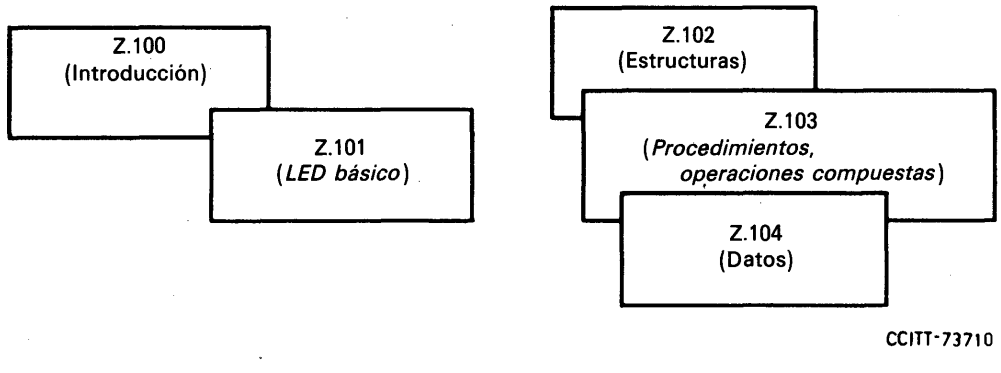


FIGURA 13/Z.100

Estructura de las recomendaciones sobre el LED

Las Recomendaciones Z.102 a Z.104 contienen extensiones del subconjunto mínimo LED y se deben aplicar cuando proceda. Las extensiones se pueden aplicar en cualquier combinación.

En los anexos se dan ejemplos de la utilización de conceptos definidos en las Recomendaciones. En las directrices para el usuario del LED se dan ejemplos más generales de la utilización del LED.

A continuación se ofrece un breve resumen del contenido de las Recomendaciones:

Z.101 Define los conceptos básicos del LED. La Recomendación forma el subconjunto mínimo de LED que se va a aplicar. Este subconjunto de LED es suficiente para describir el *comportamiento* de los *sistemas.*

Z.102 Define conceptos estructurales adicionales utilizados para describir *sistemas* amplios y/o complejos. Se pueden utilizar tanto para describir la *estructura* real del *sistema* como para describir el *sistema* en varios *niveles de abstracción.*

Z.103 Define el concepto de *procedimiento*, las *operaciones compuestas*, el concepto de *macro*, el concepto de *opción* y las extensiones pictográficas orientadas al estado (*LED/PE*). Estos conceptos se definen independientemente y pueden aplicarse en cualquier combinación.

Z.104 En la Recomendación Z.104 se definen ciertos conceptos de datos que se consideran predefinidos en las Recomendaciones Z.100 a Z.103. Conviene advertir que los *datos* se pueden utilizar de manera completamente informal en el LED.

La Recomendación Z.104 contiene también la definición del *concepto abstracto de tipos de datos* en el LED.

Además de estas Recomendaciones se dispone de una serie de documentos auxiliares que no tienen carácter de Recomendación y que describen y explican el lenguaje. Algunos de dichos documentos figuran anexos a las Recomendaciones. Los documentos auxiliares son:

Definición formal del LED

Este documento contiene la definición matemática del LED. La definición se expresa en semántica denotacional (VDM, META, IV). Existirá, en breve, como documento separado [1].

Glosario LED

Este documento contiene todos los términos relativos al LED. Cada término es objeto de una breve explicación y de una referencia al lugar en que está definido en las Recomendaciones. El documento constituye el anexo A a las Recomendaciones y aparece en el Fascículo VI.11 del Libro Rojo.

Resumen de la sintaxis abstracta del LED

Este documento contiene un resumen de la *sintaxis abstracta* completa del lenguaje. La *sintaxis abstracta* se describe en forma BNF abreviada. El documento constituye el anexo B a las Recomendaciones, y aparece en el Fascículo VI.11 del Libro Rojo.

Resumen de la sintaxis concreta del LED

Este documento contiene un resumen de todas las sintaxis concretas del LED, es decir, la representación gráfica (*LED/GR*), la extensión pictográfica orientada al estado de la forma gráfica (*LED/PE*) y la representación de frase textual (*LED/PR*). El documento constituye el anexo C a las Recomendaciones, y aparece en el Fascículo VI.11 del Libro Rojo.

Directrices para el usuario del LED

En este documento se dan ejemplos y explicaciones del uso del LED (sin definir el lenguaje). Contiene varios ejemplos y análisis de los diferentes usos del LED. Algunos de los conceptos definidos en las Recomendaciones se deben utilizar con especial cuidado, lo cual se explica también en las directrices para el usuario. El documento constituye el anexo D a las Recomendaciones, y aparece en el Fascículo VI.11 del Libro Rojo.

Curso sobre el LED

Este documento tiene por objeto enseñar la utilización del LED. Existe como documento separado [2].

Finalmente, en la parte interior de la contraportada de este fascículo se incluyen dos plantillas que deberán utilizarse al dibujar las formas gráficas del LED. Contienen todos los símbolos gráficos recomendados en su formato recomendado.

---

[1] Se podrá solicitar la Definición formal del LED, al Servicio de ventas de la Secretaría General de la Unión Internacional de Telecomunicaciones, Place des Nations, CH-1211 Ginebra 20 (Suiza).

[2] El Curso sobre el LED se podrá solicitar al Secretario General de la UIT – Sistema Internacional de Intercambio de la UIT sobre formación profesional – Departamento de Cooperación Técnica, División de formación profesional – Place des Nations, CH-1211 Ginebra 20, (Suiza).

LED BÁSICO

1 Introducción

En esta Recomendación se define la base del lenguaje de especificación y descripción (LED) del CCITT. La base del LED es el concepto de máquinas de comunicación de estado finito denominadas *procesos.* Un *sistema* LED es un conjunto de *bloques.* Los *bloques* están conectados entre sí y a su *entorno* por *canales.* Dentro de cada *bloque* hay uno o varios *procesos.* Estos *procesos* comunican entre sí por señales y se supone que se realizan al mismo tiempo.

Para definir el LED se ha considerado útil comenzar por definir un modelo común LED. Este modelo común forma una base abstracta para las *sintaxis concretas* y las enlaza entre sí. Las *sintaxis concretas* no son sino medios alternativos de representar los conceptos del LED. Actualmente hay dos *sintaxis concretas*: LED/GR y LED/PR. Como estas dos sintaxis representan los mismos conceptos LED, es posible referir un *sistema* definido utilizando una forma de *sintaxis concreta* LED a la otra *sintaxis concreta.*

En esta Recomendación se define el lenguaje definiendo en primer lugar el modelo de lenguaje común y a continuación las *sintaxis concretas* LED/GR y LED/PR.

2 Modelo de lenguaje común

2.1 *Introducción al modelo común*

En LED un *sistema* es un conjunto de *bloques* conectados entre sí y al *entorno* del *sistema* por medio de *canales* (véase la figura 1/Z.101). Los *canales* son unidireccionales.

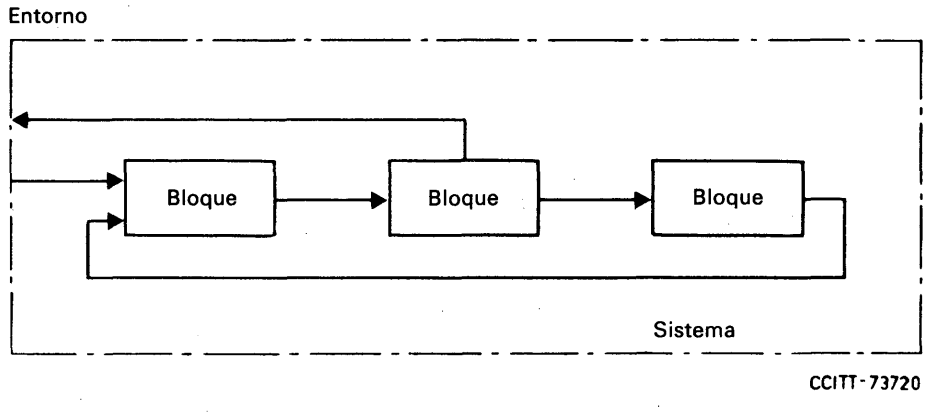


FIGURA 1/Z.101

Modelo LED

El comportamiento de cada *bloque* se modela por uno o varios *procesos.* Un *proceso* es definido por una *definición de proceso.*

Los *procesos* interactúan con otros procesos o con el *entorno* por medio de *señales.* Una *señal* es un flujo de *datos que transporta* información entre *procesos.* Cuando un *proceso* produce una *señal,* la *señal* será transportada al *proceso* al que está dirigida. El mecanismo de transporte de señales *transportadas* entre *procesos* en el mismo *bloque* es el mismo que el de *señales transportadas* entre *procesos* en diferentes *bloques.* Los *canales* representan la ruta de transporte de las *señales* intercambiadas entre *bloques.*

Cuando una *señal* llega al *proceso* al que está dirigida, será *retenida* fuera del *proceso* hasta que el *proceso* esté dispuesto a recibir la *señal de entrada.* Una *señal* será consumida cuando el *proceso* de destino la recibe.

Los modelos LED son *sistemas* abiertos, lo que significa que el *sistema* puede interactuar con su *entorno*. Esta interacción se realiza únicamente por medio del *transporte de señales* por *canales* que van o vienen del *entorno*. Se supone que el *entorno* actúa a la manera del LED, es decir, se puede considerar que el *entorno* contiene un *proceso* que produce *señales* a los *canales* que conducen al *sistema* y recibe *señales* de los *canales* que proceden del *sistema*.

Durante su vida útil el *proceso* se encuentra en un *estado* (en espera de recibir una o varias *señales*) o en una *transición* (ejecutando una serie de acciones). Cuando se encuentra en un *estado*, el *proceso* sólo puede recibir un conjunto especificado de *señales*. Si una de esas *señales de entrada* es *retenida* fuera del *proceso*, es recibida por el *proceso*. La recepción de la *entrada* hace que los *datos* transportados por la *señal* sean accesibles al *proceso* e *inicia* una *transición*. Durante la *transición*, se pueden manipular los *datos* del *proceso* y producir *señales*. La *transición* termina cuando el *proceso* entra en un nuevo *estado* o con una *parada*.

Una *parada* hace que el *proceso* deje de existir.

Común al *sistema* y a su *entorno* es el concepto de tiempo absoluto; este concepto es idéntico en todo el *sistema* y en el *entorno*.

2.2 *Sintaxis abstracta*

Definición de sistema

Una *definición de sistema* contiene *un nombre de sistema*, una o varias *definiciones de bloque*, un conjunto de *definiciones de canal* y un conjunto de *definiciones de señal*.

Una *definición de sistema* contiene la *definición de señal* de cada *nombre de señal* contenida en la *lista de señales* asociada a cada *definición de canal*.

Definición de bloque

Una *definición de bloque* contiene un *nombre de bloque*, una o varias *definiciones de proceso* y puede contener *definiciones de señal*.

Cada *definición de bloque* contiene la *definición de señal* de cada tipo de *señal* intercambiada entre *procesos* dentro del *bloque*.

Definicion de canal

Una *definición de canal* contiene un *nombre de canal*, un *identificador de definición de bloque* de origen, un *identificador de definición de bloque* de destino y una *lista de señales*. La *lista de señales* contiene el *identificador* de cada *tipo* de *señal* que se puede *transportar* a través del *canal*.

El *identificador de definición de bloque* de origen y el *identificador de definición de bloque* de destino asociados a una *definición de canal* deben ser diferentes y cada uno debe ser el *identificador* de una *definición de bloque* en la *definición del sistema* o debe ser el *entorno*.

La *lista de señales* asociada a la *definición de canal* contiene una *identificación de señal* como mínimo.

Definición de señal

Una *definición de señal* contiene un *nombre* de señal y puede contener una *lista de nombres de tipo de datos*.

Definición de proceso

La *definición de proceso* contiene un *nombre de proceso*, un par de *enteros* y un *gráfico de proceso* y puede contener una *lista de parámetros formales*, *definiciones de variable* y *definiciones de visión*.

Gráfico de proceso

Un *gráfico de proceso* es un gráfico cuyos *nodos* están conectados por *arcos* dirigidos. El *arco* que entra en un *nodo* es un *arco* de entrada y el *arco* que sale de un *nodo* es un *arco* de salida. El *nodo* que tenga un *arco* como *arco* de entrada sigue al *nodo* que tiene el mismo *arco* como *arco* de salida.

Existen las siguientes categorías de *nodos*:

Nodo de estado
Nodo de entrada
Nodo de tarea
Nodo de salida
Nodo de decisión
Nodo de arranque
Nodo de parada
Nodo de petición de crear.

La conectividad de un *gráfico de proceso* se rige por las reglas siguientes:

1) Cada *gráfico de proceso* contiene un *nodo de arranque* y solamente uno. El *nodo de arranque* viene seguido por una *cadena de transición.* El *nodo de arranque* no sigue a ningún otro *nodo.*

2) Una *cadena de transición* puede ser cualquiera de los siguientes puntos:

   a) nulo seguido de un *nodo de estado* o de un *nodo de parada,*
   b) una cadena de *acción* seguida de una *cadena de transición,*
   c) un *nodo de decisión.*

3) Una cadena de *acción* puede ser cualquiera de los siguientes puntos:

   a) un *nodo de tarea,*
   b) un *nodo de salida,*
   c) un *nodo de petición de crear.*

4) Un *nodo de decisión* va seguido de dos o más *arcos de decisión.*

5) Un *arco de decisión* es un *arco* nombrado seguido de una *cadena de transición.*

6) Un *nodo de estado* va seguido de uno o varios *nodos de entrada.*

7) Un *nodo de entrada* sigue a un *nodo de estado* y solamente uno.

8) Un *nodo de entrada* va seguido de una *cadena de transición.*

9) El *nodo de parada* no va seguido de ningún *nodo.*

10) Cada *gráfico de proceso* tiene como máximo un *nodo de parada.*

11) Cada *nodo* se puede alcanzar desde el *nodo de arranque.*

Nodo de estado

Un *nodo de estado* contiene un *nombre de estado* y puede contener un *conjunto de conservación de señales.* Los *nodos de estado* contenidos dentro de un *gráfico de proceso* tienen diferentes nombres.

Nodo de entrada

Un *nodo de entrada* contiene un *identificador de señal* y puede contener un conjunto ordenado de *identificadores de variable.*

Conjunto de conservación de señales

Un *conjunto de conservación de señales* contiene *identificadores de señal.*

Nodo de tarea

Un *nodo de tarea* contiene una secuencia de *sentencias* o *texto informal.*

Sentencia

Se entiende por *sentencia* una *sentencia de inicialización*, una *sentencia de reinicialización* o una *sentencia de asignación.*

Sentencia de inicialización

Una *sentencia de inicialización* contiene una *expresión de tiempo* y un *identificador de temporizador.*

Sentencia de reinicialización

Una *sentencia de reinicialización* contiene un *identificador de temporizador.*

Sentencia de asignación

Una *sentencia de asignación* contiene un *identificador de variable, un operador de asignación* y una *expresión.*

Nodo de salida

Un *nodo de salida* contiene un *identificador de señal,* una *expresión* de destino y puede contener una *lista de parámetros reales.*

Nodo de decisión

Un *nodo de decisión* contiene una *pregunta* y tiene por lo menos dos *arcos* de salida. Cada *arco* de salida tiene asociado un conjunto de una o varias *respuestas* a la *pregunta.* Cada posible *respuesta* a la *pregunta* debe estar asociada con un *arco* y solamente uno. Una *pregunta* es una *expresión* o *texto informal.* Una *respuesta* es un *identificador de variable* o *texto informal.*

Nodo de petición de crear

Un *nodo de petición de crear* contiene un *identificador de definición de proceso* y puede contener una *lista de parámetros reales.*

Tipo de datos

Existen *tipos de datos predefinidos* para los *tipos* siguientes: *natural, entero, real, carácter, cadena de caracteres, booleano, tiempo, duración, temporizador,* e *identificador de instancia de proceso.* Su definición figura en la Recomendación Z.104.

Definición de variable

Una *definición de variable* contiene un *nombre de variable* y un *identificador de tipo de datos* y puede contener un *atributo de revelación.*

Lista de parámetros formales

Una *lista de parámetros formales* es un conjunto ordenado de *parámetros formales.*

Parámetro formal

Un *parámetro formal* contiene un *nombre de parámetro formal* y un *identificador* de tipo.

Lista de parámetros reales

Una *lista de parámetros reales* es un conjunto ordenado de *parámetros reales.*

Parámetro real

Un *parámetro real* es una *expresión.*

Definición de visión

Una *definición de visión* contiene un *identificador de variable,* un *identificador de tipo de datos* y un *identificador de definición de proceso.*

La *variable* debe tener un *atributo de revelación* en la *definición de proceso* a que se refiere el *identificador de definición de proceso.* La *definición de proceso* de revelación a que se hace referencia debe pertenecer al mismo *bloque* que la *definición de proceso* de visión.

Expresión

Una *expresión* es un *identificador de valor,* o un *identificador de variable* o una *operación.*

Operación

Una *operación* contiene una *expresión de visión* o un *operador* y una lista de una o varias *expresiones.*

Expresión de visión

Una *expresión de visión* se compone de un *operador de visión* más un *nombre de variable* y un *identificador de instancia de proceso.*

2.3 *Reglas de interpretación*

2.3.1 *Sistema*

Un *sistema* es una entidad concreta, como una central telefónica y es una *instanciación* de un *tipo de sistema* definido por una *definición de sistema.* Un *sistema* está separado de su *entorno* por una *frontera de sistema* y contiene un conjunto de *bloques.* La comunicación entre el *sistema* y su *entorno* o entre los *bloques* del *sistema* sólo se puede efectuar utilizando *señales.* Dentro de un *sistema* estas *señales* se transportan por *canales.* Los *canales* conectan *bloques* entre sí o con las *fronteras del sistema.*

El *sistema* posee una función sin parámetro de *tipo tiempo* denominada NOW (AHORA), que indica el *tiempo* real. El tiempo *real* está inmediatamente disponible en todo el *sistema* y en su *entorno.* NOW (AHORA) se puede utilizar en *expresiones* de cualquier *proceso* del *sistema.*

2.3.2 *Canal*

Dentro del *sistema* hay un *canal* para cada *definición de canal* en la *definición de sistema.* Un *canal* es una ruta de transporte de *señales.* La ruta es unidireccional. Los puntos extremos del *canal* son un *bloque* o la *frontera del sistema.* Por lo menos uno de los puntos extremos del *canal* debe estar en un *bloque.* Si los dos puntos extremos están en *bloques,* los *bloques* deben ser diferentes. La *definición de canal* contiene la lista de todas las *señales* que se puedan transportar por el *canal.*

Cuando se da una *señal* al *canal*, la *señal* es transportada al *bloque* de destino. El orden en que las *señales* se dan al *canal* y el orden de las *señales* recibidas por el *canal* es el mismo. Si dos o más *señales* llegan simultáneamente a un canal, se les atribuye un orden arbitrario.

2.3.3 *Bloque*

Dentro del *sistema* hay un *bloque* para cada *definición de bloque* en la *definición del sistema.* Un *bloque* es un objeto de tamaño manejable en el que se pueden interpretar uno o varios *procesos.* Hay dos mecanismos de comunicación entre *procesos* situados dentro del mismo *bloque*: las *señales* y los *valores compartidos.*

Cuando una *señal* llega desde un *canal* al *bloque*, el *bloque* entrega la *señal* al *puerto de entrada* del *proceso* previsto por el *identificador de proceso* de la *señal.*

Cuando el *proceso* produce una *señal* dentro del *bloque*, se entrega al *proceso* previsto por el *identificador de proceso* de la *señal.* Si el proceso *previsto* está dentro del mismo *bloque*, el *bloque* entrega la *señal* a su *puerto de entrada.* Si el *proceso* previsto está en otro *bloque*, el *bloque* entrega la *señal* al *canal* capaz de transportar esa *señal.*

Los *valores compartidos* permiten a un *proceso* ver una *variable* revelada en otro *proceso.* Sólo el *proceso* de revelación puede cambiar el *valor* de la *variable.* El *proceso* de visión recibe el *valor* corriente de la *variable* revelada utilizando un *operador de visión.*

2.3.4 *Señal*

Una *señal* es un flujo de *datos* que transporta información entre *procesos* y es una instanciación de un *tipo de señal* definido por una *definición de señal.* Una *señal* puede ser enviada por el *entorno del sistema* o por un *proceso* y siempre está dirigida hacia un *proceso* o hacia el *entorno.*

Cada señal contiene el *identificador de señal* en la definición de *señal*, un *identificador de instancia de proceso de origen* y un *identificador de instancia de proceso de destino.* Además, se pueden transportar en una *señal* otros *valores*, mediante *variables.* En una *señal*, hay una *variable* para cada *nombre* en la lista de *tipos de datos* de la *definición de señal.*

2.3.5 *Proceso*

Un *proceso* es una máquina de comunicación de estado finito y es una instanciación de un *tipo de proceso* definido por una *definición de proceso* dentro de un *bloque.* Puede haber uno o varios *procesos* para cada *definición de proceso.* Los *procesos* pueden existir desde el momento de creación del *sistema* o pueden ser creados por *acciones de petición de crear* y cesar de existir por la ejecución de *acciones de parada.* Un *proceso* puede actuar independientemente de los demás *procesos* del *sistema* y concurrentemente con ellos.

Todos los *procesos* del *sistema* poseen cuatro *variables* predefinidas de *tipo de identificador de proceso* llamados: SELF (MISMO), PARENT (ASCENDIENTE), OFFSPRING (DESCENDIENTE) y SENDER (EMISOR). Estas *variables* son *identificadores de instancia de proceso* para:

- el *proceso* (SELF);
- el *proceso* creador (PARENT);
- el *proceso* más recientemente creado (OFFSPRING);
- el *proceso* del que se ha recibido la última *señal de entrada* (SENDER).

Estas *variables* se pueden utilizar en *expresiones* pero no se les puede asignar explícitamente un *valor.* Para todos los *procesos* presentes en la *inicialización* del *sistema*, PARENT tiene un mismo *valor* específico, diferente del *valor* de SELF en cualquier *proceso.*

Las *señales* enviadas al *proceso* son *señales de entrada* y las *señales* procedentes del *proceso* son *señales de salida.* Una *señal de entrada* es una entidad destinada a invocar el *proceso* y comunicarle información. Una *señal de salida* está destinada a invocar otro *proceso* y comunicarle información.

El conjunto de *identificadores de señal* que va asociado a *nodos de entrada* del *gráfico de proceso* denota el conjunto de *identificadores de señales de entrada válidas* para esta *definición de proceso.* Para cada *estado*, todos los *identificadores de señal de entrada* aparecen en un *conjunto de conservación de señales* o en un *nodo* de entrada.

El par de *enteros* contenido en la *definición de proceso* define el número de *instancias* del *proceso* que se crean al crear el *sistema* y el número máximo de *instancias* simultáneas del mismo *tipo de proceso.* Cuando se crea un *sistema*, los *procesos* iniciales se crean en orden aleatorio y no se envía ningún *parámetro real* al *proceso.*

Cuando se crea un *proceso*, se le da un *puerto de entrada* vacío y se crean *variables* con *valores no definidos*. A continuación *arranca el proceso* interpretando el *nodo de arranque* en el *gráfico de proceso*.

Cuando una *señal de entrada válida* llega al *proceso*, pasa al *puerto de entrada* del *proceso*. Cada *proceso* contiene un solo *puerto de entrada*. El *puerto de entrada* puede retener cualquier número de *señales de entrada*, con lo que puede haber varias *señales de entrada* haciendo cola para el *proceso*. El conjunto de *señales retenidas* se ordena en la cola de acuerdo con su hora de llegada. Si dos o más *señales* llegan al mismo tiempo, se ordenan arbitrariamente. Adjunto al *puerto de entrada* figura un eventual conjunto vacío de *temporizadores*. Cada temporizador contiene un *valor* del *tipo de tiempo*.

El *proceso* está en su estado, en espera, o activo, efectuando una *transición*. Para cada *estado* hay un *conjunto de conservación de señales*. Cuando está en espera en un *estado*, la primera *señal de entrada* cuyo *identificador* no se encuentra en el *conjunto de conservación de señales* se toma de la cola y es recibida por el *proceso*. El *puerto de entrada* compara continuamente NOW con el temporizador, si existe, que tenga el *valor* más bajo mayor que cero. Cuando el *valor* de NOW es superior o igual al *valor* de este temporizador, se coloca en la cola de espera una *señal* con el mismo *nombre* de este temporizador y se da el valor cero al temporizador.

Cuando un *proceso* está en espera, se encuentra siempre en un *estado* único denotado por el *nodo de estado* correspondiente del *gráfico de proceso*. Cuando se recibe una *señal de entrada*, el *proceso* se activa, interpreta el *nodo de entrada* que tiene el mismo *nombre* que la señal de *entrada* y efectúa una *transición* que conduce a un nuevo *estado*.

2.3.6 *Gráfico de proceso*

Un *nodo de arranque* se interpreta como una *acción de arranque*. La *acción de arranque* hace que se interprete el *nodo* que sigue al *nodo de arranque*.

Un *nodo de entrada* se interpreta como una *acción de entrada* que recibe y consume la *señal* dada y a continuación se interpreta el *nodo* siguiente al *nodo de entrada*. El consumo de la *señal* pone a disposición del *proceso* la información transportada por la *señal*. Se asignan a las *variables* del *nodo de entrada* los *valores* de las correspondientes *variables* de la *señal*. Si no hay *variable* en el *nodo de entrada* para una *variable* de la *señal*, se descarta el *valor* de la *variable*. SENDER en el *proceso* de recepción recibe el *valor* del *identificador de instancia de proceso* de origen transportado por la *señal*.

Un *nodo de tarea* se interpreta como una *acción de tarea*. La interpretación de la *acción de tarea* consiste en una secuencia de *sentencias* o *texto informal*. Cuando la *acción* está completa, se interpreta el *nodo* que sigue al *nodo de tarea*.

Una *sentencia de inicialización* causa una *sentencia de reinicialización* en el *temporizador* indicado en la *sentencia* y asigna al *temporizador* el *valor de tiempo* especificado.

Una *sentencia de reinicialización* fija el *valor* de tiempo del temporizador a cero. Toda señal con el mismo *nombre* del temporizador que esté presente en la cola de espera de entrada, se suprimirá de la cola de espera de entrada y se descartará. Todas las *señales* en cola de espera que tengan un *identificador* igual al *identificador de la señal* de temporización se retiran de la cola de espera y se descartan.

Una *sentencia de asignación* se interpreta de modo informal como el *valor* que la *variable* en la *sentencia de asignación* toma con respecto al *valor* de la *expresión* en la *sentencia de asignación*.

Un *nodo de salida* se interpreta como una *acción de salida* que crea una *señal* y la entrega al *bloque*. A continuación se interpreta el *nodo* que sigue al *nodo de salida*. La *señal de salida* es una *instalación* de un *tipo de señal* definido por la *definición de señal* indicada por el *identificador de señal* del *nodo de salida*. Se asignan a las *variables* de la *señal* los valores de los *parámetros reales* del *nodo de salida*. Si no hay *parámetro real* alguno en el *nodo de salida* para una *variable* de la *señal*, la *variable* tiene un *valor no definido*. Se asigna al *identificador de instancia* transmitido por la *señal* el *valor* de la *variable* SELF. Se asigna al *identificador de instancia de proceso* de destino de la *señal* el *valor* de la *expresión* contenida en el *nodo de salida*.

Un *nodo de decisión* se interpreta como una *acción de decisión* que responde a una *pregunta*. Se elige el *arco* que une la *respuesta* a la *pregunta* y se interpreta el *nodo* que sigue a este *arco*.

Un *nodo de estado* se interpreta como la terminación de una *transición* dando al proceso un nuevo *estado* denotado por el *nombre* contenido en el *nodo*. Se informa al *puerto de entrada* de que el *proceso* está en espera, en un *estado* y se le presenta también el *conjunto de señales de conservación* agregado al *nodo de estado* interpretado. Después de esto el *proceso* espera hasta recibir una nueva *señal de entrada*.

Un *nodo de parada* se interpreta como la terminación inmediata del *proceso*. Ello significa que las *señales de entrada* retenidas en el *puerto de entrada* se descartan y que las *variables* creadas para el *proceso*, el *puerto de entrada* y el *proceso* cesarán de existir.

Un *nodo de petición de crear* se interpreta como una *acción de petición de crear*. La *acción de petición de crear* provoca la creación de un *proceso* en el mismo *bloque*. La *definición* del *proceso* se encuentra en el mismo *bloque* y se identifica por el *identificador de proceso* en el *nodo de petición de crear*. Como parte de la *acción de petición de crear*, se asigna a la *variable* PARENT el *valor* de la *variable* SELF del *proceso* de creación. A la *variable* SELF del *proceso* creado y a la *variable* OFFSPRING del *proceso* de creación se les asigna el mismo y único *valor* de *identificación de instancia de proceso*. A los *parámetros formales* del *proceso* de nueva creación se les asignan los *valores* de los parámetros reales contenidos en el *nodo de petición de crear*.

2.3.7 *Tipo de datos*

Los tipos predefinidos se utilizan en el sentido normal.

2.3.8 *Variable*

A una *variable* se le puede asignar un *valor*. El contenido del *valor* asignado a una *variable* puede ser recuperado de la *variable* más adelante. Una *variable* tiene también un *tipo de datos* que restringe la clase de *valores* que se pueden asignar a la *variable*.

Dentro de un *proceso* existe una *variable* para cada *definición de variable* en la *definición de proceso*. El *valor* de una *variable* sólo puede ser modificado por el *proceso* que la define. El *valor* de una *variable* es conocido solamente por el *proceso* que la define, a menos que la *variable* posea el *atributo de revelación*. El *atributo de revelación* permite a otros *procesos* del *bloque* ver la *variable*. Las *variables* poseen la misma duración de vida útil que el *proceso* que las define (es decir, son creadas cuando se crea el *proceso* que las define y dejan de existir cuando el *proceso* que las defina deja a su vez de existir).

Dentro de una señal hay una *variable* anónima para cada aparición de cada *nombre de tipo de datos* en la *definición de señal*. El *valor* de la *variable* solamente se puede asignar en el *nodo de salida* que haya creado la *señal* y sólo se puede conocer en el *nodo de entrada* que recibe y consume la *señal*. Estas *variables* tienen una duración de vida útil limitada por la duración de vida útil de la *señal*.

2.3.9 *Expresión*

La interpretación informal de una *expresión* produce un *valor*.

2.3.10 *Definición de visión*

Una *definición de visión* define un *nombre de variable* que sólo se puede utilizar en una *expresión de visión* para obtener el *valor* de una *variable* perteneciente a otro *proceso*.

2.3.11 *Expresión de visión*

El *valor* de una *expresión de visión* es el *valor* de la *variable* identificada por el *nombre de la variable* y el *identificador de instancia de proceso* en la *expresión de visión*.

3 LED/GR

Una *definición de sistema* está representada en la sintaxis LED/GR por:

- un *diagrama de interacción de bloques* que contiene el *nombre del sistema* y las *definiciones de canal* y que identifica las *definiciones de bloque*. El *diagrama de interacción de bloques* identifica también las *definiciones de proceso* que modelan el comportamiento de cada *bloque*. El *diagrama de interacción de bloques* puede denotar: 1) las listas de *señales* que pasan entre *procesos* en el mismo *bloque*, 2) las listas de *señales* transportadas por *canales* entre *bloques* y 3) la creación de nuevas *instancias de proceso* por otras *instancias de proceso*;
- *diagramas de proceso* que definen el comportamiento de cada *proceso* y son la representación gráfica de las *definiciones de proceso*. Los *diagramas de proceso* pueden contener *definiciones de variable*, *parámetros formales* y *definiciones de visión*;

– *listas de señales* que nombran las *señales* transportadas por un *canal* o desde un *proceso* a otro dentro del *bloque*. Pueden incorporarse el *diagrama de interacción de bloques* utilizando *símbolos de lista de señales* o presentarse como listas separadas en cualquier forma conveniente;

– *definiciones de señal* que dan los *tipos de datos* y orden de los *valores de datos* que se pueden contener en una *señal*, para cada *señal* nombrada en las *listas de señales*. Se especifican utilizando la sintaxis LED/PR.

En una *definición de sistema*, se pueden utilizar *nombres* e *identificadores*. Los *nombres* se especifican utilizando la sintaxis LED/PR. Los *identificadores* constan de un *nombre* junto con *calificativos*. Un *calificativo* es el *tipo* de entidad con el que está asociado el *nombre* y representa el *nivel* jerárquico de la entidad, que se está identificando. Dos o más entidades no pueden tener el mismo *identificador*. En el LED/GR, los calificativos de *nombres* pueden inferirse del contexto, pero siempre que se utilizan se especifican usando la sintaxis LED/PR.

3.1 *Diagramas de interacción de bloques*

El *diagrama de interacción de bloques* de un *sistema* contiene un *nombre de sistema*, un conjunto de *símbolos de bloque*, *símbolos de entorno*, un conjunto de *símbolos de canal* y *símbolos de la lista de señales*.

3.1.1 *Símbolos*

Los *símbolos* recomendados aparecen en la figura 2/Z.101.

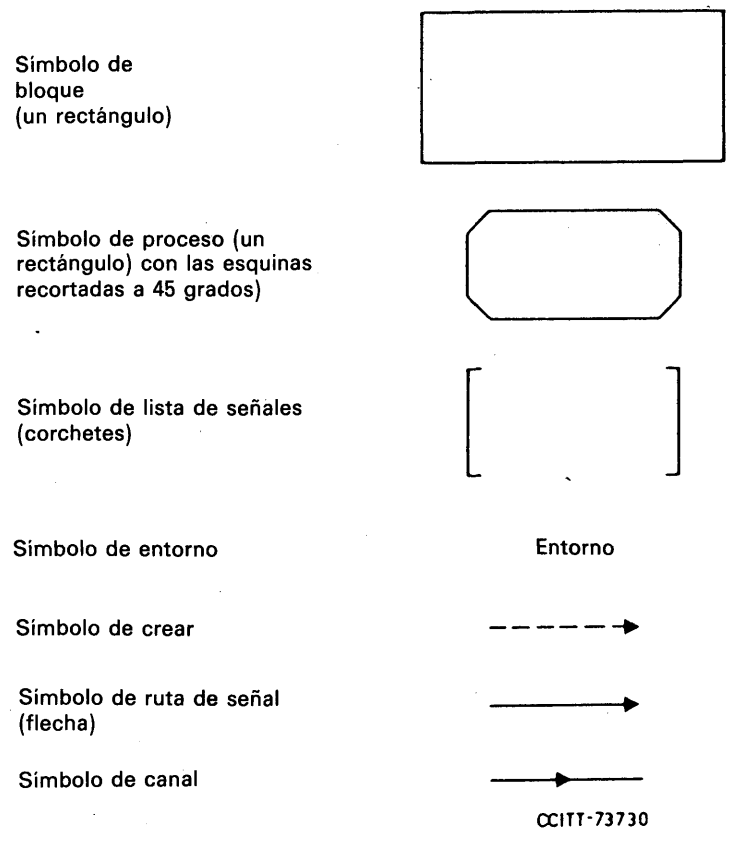


FIGURA 2/Z.101

Símbolos de diagrama de interacción de bloques

3.1.2 *Relación entre diagramas de interacción de bloques LED/GR y la sintaxis abstracta, y uso de los símbolos*

- El *símbolo de entorno* representa el *entorno del sistema* y puede aparecer varias veces.

- El símbolo de bloque que contiene un *nombre*, un conjunto no vacío de *símbolos de proceso*, *símbolos de ruta de señal* y puede contener *símbolos de crear* y *símbolos de lista de señales*.

- Un *símbolo de proceso* contiene un *nombre de proceso* y puede contener un *símbolo de lista de parámetros formales*. El *nombre del proceso* es el mismo que el *nombre* contenido en la *definición de proceso* que describe el comportamiento del *proceso*. El símbolo *de lista de parámetros formales* contiene una lista de los *nombres* de los *parámetros formales* que se inicializan cuando se crea el *sistema* o cuando se crea dinámicamente la *instancia de proceso*.

- Un par de *valores enteros* puede estar asociado a un *símbolo de proceso* ; el primer *valor* representa el número de instancias de *proceso* que existen cuando se crea el *sistema* y el segundo *valor* representa el número máximo de *instancias* simultáneas de este *tipo de proceso*. Los dos *valores* están situados en la esquina superior derecha del *símbolo de proceso*.

  El valor por defecto del primer *valor* es uno. El valor por defecto del segundo *valor* es infinito. Este par de *enteros* se especifica utilizando la sintaxis LED/PR. Cuando uno de los *enteros* no está especificado, se le asigna el *valor* por defecto.

- Un *símbolo de canal* tiene un *nombre de canal* atribuido al mismo. 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, la conexión de origen o la conexión de destino (pero no ambas) pueden estar conectadas a un *símbolo de entorno* en vez de a un *símbolo de bloque*. Al lado del *símbolo de canal* se puede poner un *símbolo de lista de señales* para identificar las señales transportadas por el *canal*.

- Un *símbolo de ruta de señal* en un bloque está asociado a 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 del *bloque*), o desde el extremo de destino de un *canal* (en la frontera del *bloque*) a un *proceso*.

- Una *lista de señales* en un *diagrama de interacción de bloques* es una lista de *nombres*, totalmente encerrada en el *símbolo de lista de señales* (es decir, entre corchetes). La *lista de señales* puede tener un *nombre*, que se escribe por encima del *símbolo*. Las inscripciones que figuran en la lista (que están separadas por comas y pueden aparecer en columnas o en filas) son los *nombres* de cada *definición de señal* y los *nombres* de otras *listas de señales*. Dentro de la lista, los *nombres* de listas de señales van entre corchetes para distinguirlos de los *nombres* de cada *definición de señal*. Si se decide incluir *listas de señales* en el *diagrama de interacción de bloques* utilizando *símbolos de listas de señales*, todas las *listas de señales* deben ser incluidas en el diagrama. La figura 3/Z.101 muestra ejemplos de *listas de señales*.

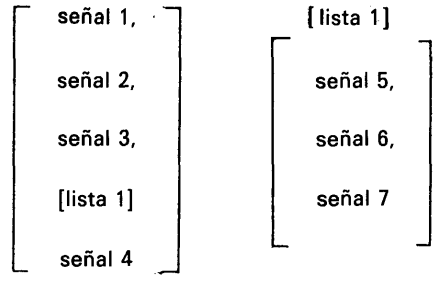


FIGURA 3/Z.101

Ejemplos de listas de señales en diagramas de interacción de bloques

– Los *símbolos de crear* conducen de un *símbolo de proceso* a otro *símbolo de proceso.* El primer *símbolo de proceso* representa el *proceso* de «creación». El último *símbolo de proceso* representa el *proceso* «creado».

3.1.3 *Reglas de representación gráfica*

– Los *símbolos de canal* están conectados a las fronteras de los *símbolos de bloque* a los que están asociados. Los *símbolos de canal* deben enlazar con las fronteras de los *símbolos de bloque* a 90°. De ser necesario, los *símbolos de canal* pueden contener curvas de 90°. Un *símbolo de canal* incluye una flecha que muestra la dirección de flujo de señales a lo largo del *canal.*

– Los *símbolos de ruta de señal* están conectados a las fronteras de los *símbolos de bloque* o *símbolos de proceso* a los que están asociados. De preferencia, los *símbolos de ruta de señal* se conectan con estas fronteras a ángulos de 90°. De ser necesario, los *símbolos de ruta de señal* pueden contener curvas de 90°. Varios *símbolos de ruta de señal* pueden confluir en el extremo del origen de un *canal* (en la frontera de *bloque*). Varios *símbolos de ruta de señal* pueden refluir del extremo de destino de un *canal* en la frontera del *bloque.* Un *símbolo de ruta de señal* incluye una flecha en un extremo para mostrar la dirección del flujo de *señales.*

– Los *símbolos de crear* están conectados a las fronteras de los *símbolos de proceso* por líneas de brazo interrumpido que enlazan con las fronteras de *símbolos de proceso* en ángulos de 90°. Se utilizan flechas en estas líneas de conexión de forma que el *símbolo de proceso* de «creación» apunte al *símbolo de* proceso «creado».

– La orientación preferida de los *símbolos* se representa en la figura 2/Z.101.

– La dimensión de los *símbolos* se deja al criterio de los usuarios.

– Las fronteras de los *símbolos* no se deben cruzar ni superponer. Una excepción a esta regla se aplica a los *símbolos de canal* y *símbolos de ruta de señal,* que se pueden cruzar. No existe relación lógica entre los *símbolos de canal* o *símbolos de ruta de señal* que se cruzan.

3.2 *Listas de señales*

La unión de todos los *nombres de señales* en las *listas de señales* asociadas a los *símbolos de ruta de señal* conectados a una determinada *definición de proceso* es igual al conjunto de *nombres de señales de entrada válidas* de ese *proceso.*

La unión de todos los *nombres de señales* en las *listas de señales* asociadas a los *símbolos de ruta de señal* conectados al extremo de origen de un *canal* es igual a la lista de *nombres* de la *lista de rutas de señales* asociadas a ese *canal* y es igual a la unión de todos los *nombres de señales* de la *lista de rutas de señales* asociadas a los *símbolos de señal* (en el *bloque* de destino del *canal*) conectados al extremo de destino de ese *canal.*

3.3 *Diagramas de proceso*

3.3.1 *Símbolos*

El comportamiento de un *proceso* se representa en forma gráfica por un *diagrama de proceso.* El *nombre* del *diagrama de proceso* es el mismo que el *nombre del proceso* que aparece en la *definición del proceso* que representa. La figura 4/Z.101 muestra los *símbolos* utilizados en los *diagramas de proceso* del formulario LED/GR.

3.3.2 *Relación entre los diagramas de proceso LED/GR y la sintaxis abstracta LED, y uso de los símbolos*

Cada *símbolo de diagrama de proceso* de la figura 4/Z.101 representa el *nodo de nombre* equivalente del *gráfico de proceso* en la *sintaxis abstracta.* Las líneas de flujo, que conectan *símbolos,* representan los *arcos* dirigidos que conectan *nodos.* En la figura 5/Z.101 se representan las conexiones admisibles de *símbolos* por medio de líneas de flujo en un *diagrama de proceso* LED/GR.

Los *parámetros formales,* el conjunto de *señales de entrada válidas,* las *definiciones de variable,* las *definiciones de visión,* las *expresiones* y las *expresiones de visión* se especifican utilizando la sintaxis LED/PR.

Un *símbolo de arranque* representa un *nodo de arranque* (véase también el § 2.2.3.3). El *símbolo de arranque* contiene el *nombre* del *proceso* que describe.

Un *símbolo de parada* representa el *nodo de parada.*

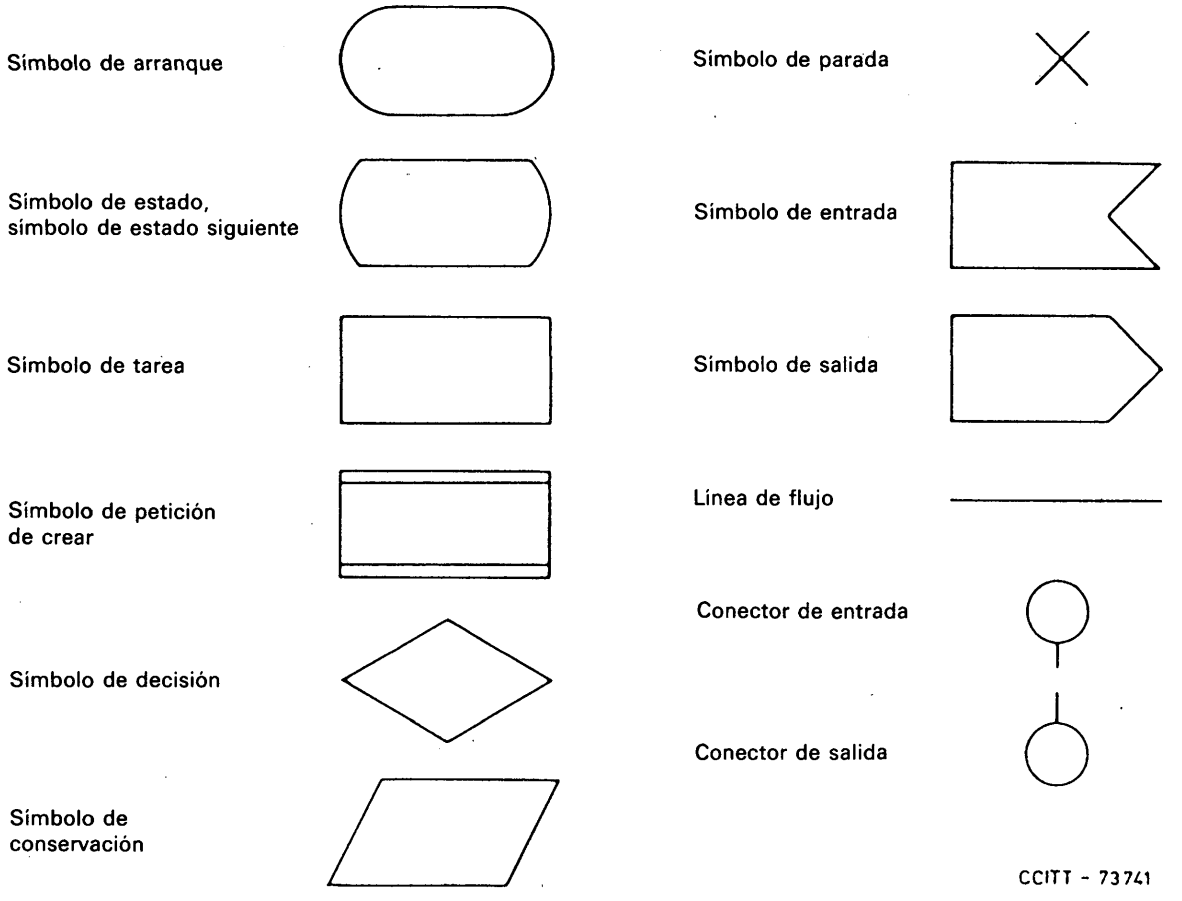


FIGURA 4/Z.101

Símbolos de diagrama de proceso LED/GR

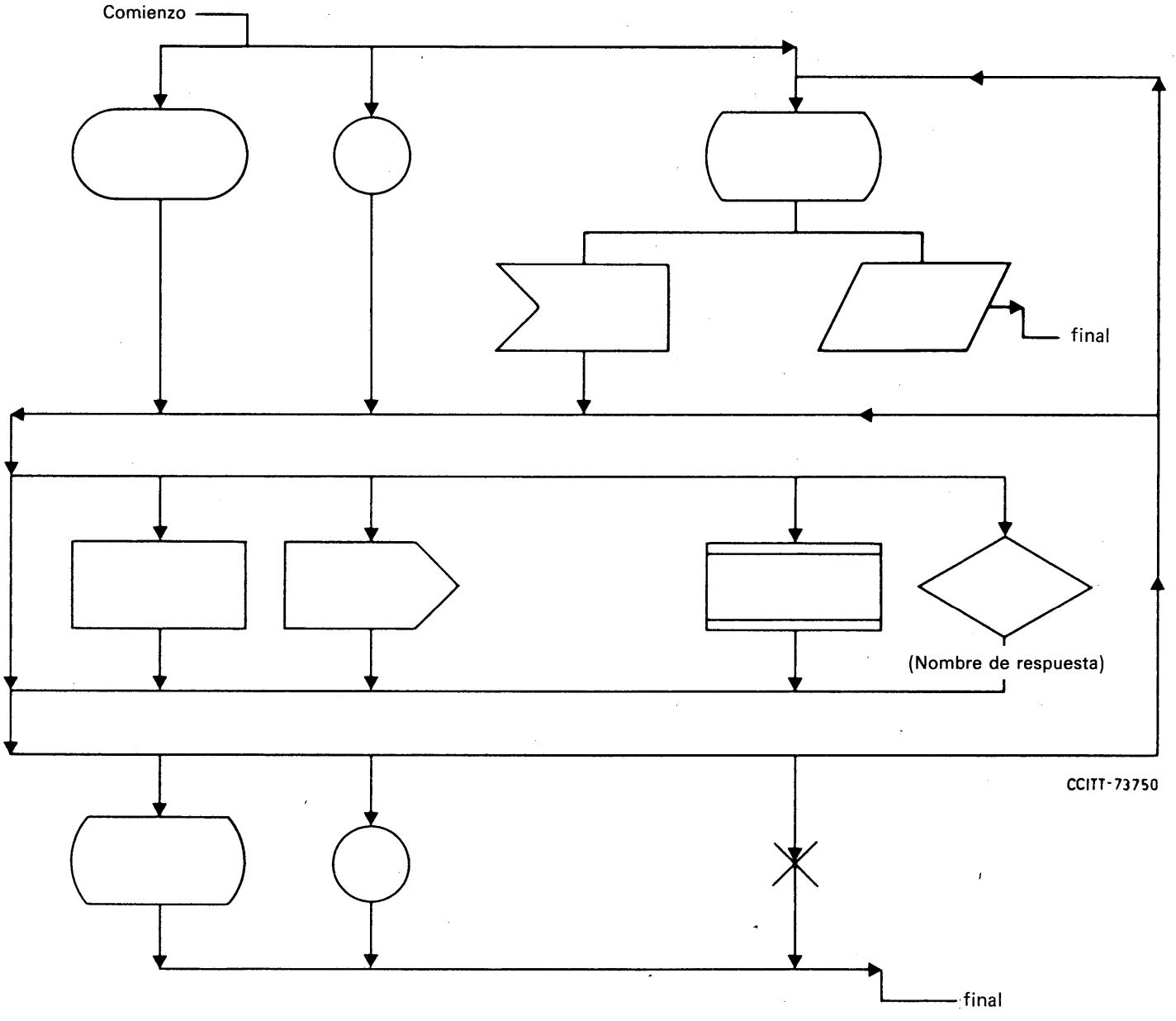


FIGURA 5/Z.101

Conexiones admisibles de símbolos por medio de líneas de flujo en un diagrama de proceso LED/GR

Un *símbolo de estado* representa uno o varios *nodos de estado* y contiene uno o varios *nombres de estado* separados por comas, o por un asterisco, o por un asterisco seguido de una lista de *nombres de estado* entre corchetes.

Un *símbolo de conservación* representa el conjunto de *señales conservadas* agregado a un *nodo de estado*. Contiene uno o varios *nombres de señal*, separados por comas o por un asterisco.

Un *símbolo de entrada* representa uno o varios *nodos de entrada*. Los *nombres de señal* contenidos en *símbolos de entrada* están separados por comas. Cada uno de estos *nombres de señal* da el *nombre* de uno de los *nodos de entrada* que representa este *símbolo de entrada*.

Un *símbolo de tarea* representa un *nodo de tarea*. El *símbolo de tarea* contiene un *nombre de tarea* y puede contener una secuencia de *sentencias* o texto informal.

Un *símbolo de petición de crear* representa un *nodo de petición de crear*. Contiene una *acción de petición de crear* especificada en la sintaxis LED/PR.

Un *símbolo de decisión* representa un *nodo de decisión*. Contiene una *pregunta* y puede contener un *nombre de decisión*. Dos o más *líneas de flujo* conducen del *símbolo de decisión* a otro *símbolo*; cada *línea de flujo* tiene agregado (es decir, escrito a lo largo de ella o insertado en una interrupción de la *línea de flujo*) su propio *nombre* de *respuesta*. La *respuesta* ELSE (OTRO) representa una *respuesta* que no sea una de las referidas por cualquier otro *nombre* de *respuesta*.

Un *símbolo de salida* representa uno o varios *nodos de salida*. Los *nombres de señal* contenidos en un *símbolo de salida* están separados por comas. Cada uno de estos *nombres de señal* da el *nombre* de un *nodo de salida* que representa este *símbolo de salida*. El *identificador de instancia de proceso* de destino puede ser dado facultativamente en el *símbolo de salida* utilizando la sintaxis LED/PR (es decir, una palabra clave TO (A) seguida de una *expresión* de *tipo identificador de instancia de proceso*. La palabra clave TO (A) sigue a la lista de *nombres de señal*). Cuando el *proceso* de destino de una *señal* no puede ser determinado de manera única porque ni el *nombre de señal* ni el contexto lo permiten suficientemente, se requiere el *identificador de instancia de proceso*.

Un *símbolo de estado siguiente* representa un arco que *conecta* el último *nodo* de una *cadena de transición* con el *nodo de estado* siguiente.

Una *línea de flujo* que conecta dos otros *símbolos* (que representan *nodos*) representa el *arco* que conecta los *nodos* correspondientes.

Se utiliza un *símbolo de comentario* para agregar *texto informal* a cualquier otro *símbolo*.

Un *conector de entrada* contiene una *etiqueta* y representa la continuación de una *línea de flujo* desde un *conector de salida* correspondiente que contiene la misma *etiqueta*.

3.3.3 *Convenciones gráficas*

3.3.3.1 *Transiciones implícitas*

La *sintaxis abstracta* del LED requiere la especificación de una *conservación* o de una *entrada* que conduce a una *transición* para cada *señal* de un conjunto de *señales de entrada válidas* de un *proceso* para cada *estado* del *proceso*. El LED/GR prevé *transiciones* «nulas» implícitas para cualquier *señal* para la que no se den explícitamente *transiciones* ni *conservaciones*. Una *transición* nula equivale a una conexión desde un *símbolo de estado* a un *símbolo de entrada* conectado a su vez al mismo *símbolo de estado*. Así pues, para un *estado* dado, LED/GR descarta cualquier *señal* que no esté explícitamente mencionada en relación con ese *estado*. Asimismo se descartan los *datos* contenidos por tales señales.

Si en el *diagrama de proceso* LED/GR no aparece ningún *símbolo de arranque*, se supone un *símbolo de arranque* implícito conectado directamente al *símbolo de estado* de «arrancar». El *estado* de arrancar puede identificarse implícitamente a partir de su *nombre* o por un *comentario*.

3.3.3.2 *Líneas de flujo y conectores*

Cuando dos o más *símbolos* van seguidos de un solo *símbolo*, las *líneas de flujo* que conducen a ese *símbolo* son convergentes. Esta *convergencia* puede aparecer como una *línea de flujo* que conduce a la otra o como varios *conectores de salida* asociados a un solo *conector de entrada*, o como *líneas de flujo* separadas que entran en el mismo *símbolo*.

Cuando un *símbolo* va seguido de dos o más *símbolos* distintos, una *línea de flujo* que sale de este *símbolo* puede separarse en dos o más *líneas de flujo*.

Se requieren flechas siempre que dos *líneas de flujo* confluyen y siempre que una *línea de flujo* entra en un *conector de salida* o en un *símbolo de estado*. Las flechas están prohibidas en las *líneas de flujo* que entran en *símbolos de entrada*. En todas las demás circunstancias, las flechas son facultativas.

3.3.3.3 *Aparición múltiple*

Siempre que aparezcan *símbolos de estado* o *símbolos de conector* que tengan el mismo *nombre*, se considera que representa la misma entidad semántica. Se considera que la entidad resultante tiene la unión de todas las *líneas de flujo* de entrada y salida procedentes de sus múltiples representaciones. Siempre que en el mismo *diagrama de proceso* aparezcan uno o varios *símbolos de parada*, representa el *nodo de parada*.

3.3.3.4 *Notación abreviada*

Una notación abreviada permite hacer referencia a todas, o a todas las demás *señales*, o *estados*, en el *símbolo de entrada*, *conservación* o *estado*.

Un *símbolo* de entrada agregado a un *estado* y que contiene un «*» (asterisco) indica que la *transición* siguiente se aplica a todas las *señales* de entrada que no aparecen por otra parte en *símbolos de entrada* o *símbolos de conservación* agregados a cualquier aparición de este *símbolo de estado*. En cualquier *estado* sólo se permite un *símbolo de entrada* o un *símbolo de conservación* que contenga un «*».

Un *símbolo de conservación* agregado a un *estado* y que contenga un «*» indica que todas las *señales* que no aparecen en *símbolos de entrada* agregados a cualquier aparición de ese *símbolo de estado* deben ser *conservadas*.

Un «*» en un *símbolo de estado* denota todos los *estados* en dicho *proceso* e indica que las *transiciones* o *conservaciones* siguientes se deben interpretar en cada *estado*. Un «*» seguido de una lista de *nombres de estado* entre corchetes indica que las *conservaciones* o *transiciones* siguientes se deben interpretar en cada *estado* excepto los indicados. Esos *símbolos de estado* no deben tener *líneas de flujo* de entrada.

Un «–» en un *símbolo de estado siguiente* significa que el *estado* siguiente es el mismo *estado* desde el que se inició la *transición*. El «–» no se permite en un *símbolo de estado siguiente* que siga al *símbolo de arranque*.

Un *símbolo de estado siguiente* y un *símbolo de estado* sólo se pueden combinar si representan el mismo *símbolo de estado*.

3.3.3.5 *Varios*

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 anchura/altura preferida de los *símbolos* es 2/1.

Se permiten representaciones invertidas de *símbolos de entrada* y *salida*.

Las *líneas de flujo* son horizontales o verticales y cambian de dirección en ángulo recto.

El cruce de *líneas de flujo* no implica relación lógica.

En la medida de lo posible, los textos relativos a un *símbolo* deben presentarse dentro de ese *símbolo*.

3.3.3.6 *Normógrafo LED*

En las Directrices para el usuario del LED se incluye un normógrafo para el dibujo del juego básico de *símbolos* del LED.

3.4 *Símbolo de ampliación de texto*

Se puede agregar un *símbolo de ampliación de texto* a todos los *símbolos* del LED/GR. Se debe considerar que el texto que figura en este *símbolo* está contenido en el *símbolo* al que se haya agregado el *símbolo de ampliación de texto*. El *símbolo de ampliación de texto* se muestra en la figura 6/Z.101.

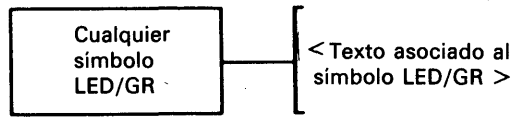


FIGURA 6/Z.101

Símbolo de ampliación de texto

3.5 *Comentarios en LED/GR*

En todos los diagramas LED/GR se pueden insertar *comentarios* siempre que el usuario lo considere adecuado. Los *comentarios* se pueden insertar utilizando el *símbolo de comentario* LED/GR (véase la figura 7/Z.101) o la sintaxis LED/PR para los *comentarios* (véase el § 4.3.2, regla de léxico 7).

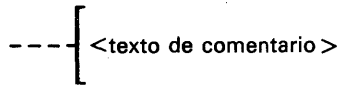


FIGURA 7/Z.101

Símbolo de comentario

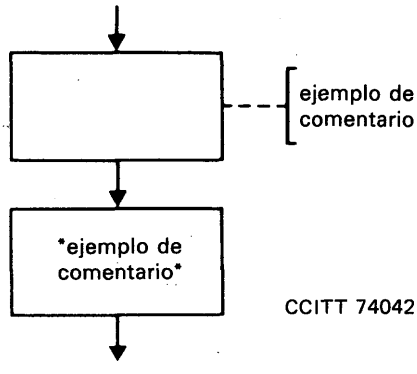


FIGURA 8/Z.101

Ejemplos de comentarios en LED/GR

4 Sintaxis lineal

4.1 *Consideraciones generales*

En este punto se define el LED/PR y su relación con el modelo de lenguaje común (véase el § 2).

Una *definición de sistema* en la sintaxis LED/PR se representa por una secuencia de sentencias limitada por las palabras SYSTEM y ENDSYSTEM.

Las reglas detalladas para la sintaxis LED/PR están contenidas en los *diagramas de sintaxis* (véase el § 4.3).

La referencia a una entidad nombrada por medios ajenos a su definición se efectúa mediante un identificador. El *identificador* comprende un *nombre* y una *parte cualificadora* facultativa. La *parte cualificadora* se tiene que utilizar cuando el *nombre* solo no puede determinar de forma única el elemento a que se hace referencia.

4.2 *Palabras clave*

El LED/PR utiliza una serie de palabras clave para expresar conceptos LED en la forma definida en la sintaxis abstracta. Algunas de estas palabras clave se utilizan por pares a fin de reflejar la estructura del LED dentro del LED/PR.

4.2.1 *Palabras clave de estructuración emparejadas que guardan relación con definiciones*

Se utiliza la palabra «engloba» con estos pares de palabras para indicar su función de delimitación.

SYSTEM (SISTEMA) ENDSYSTEM (FIN SISTEMA)	engloba el concepto de *definición de sistema* (la representación LED/PR de un sistema comienza con la palabra clave SYSTEM (SISTEMA) y finaliza con la palabra clave ENDSYSTEM (FIN SISTEMA).
BLOCK (BLOQUE) ENDBLOCK (FIN BLOQUE)	engloba el concepto de una *definición de bloque.*
PROCESS (PROCESO) ENDPROCESS (FIN PROCESO)	engloba el concepto de una *definición de proceso.*

4.2.2 *Palabras clave únicas que guardan relación con definiciones*

Las palabras clave de este punto se utilizan para indicar que sigue una definición.

DCL	introduce la representación de la *definición de variable* (la palabra clave REVEALED (REVELADO) se utiliza dentro de una sentencia DCL para identificar *variables* reveladas).
VIEWED (VISTO)	introduce la representación de la *definición de visión.*
SIGNAL (SEÑAL)	introduce la representación de la *definición de señal.*
CHANNEL (CANAL)	introduce la representación de la *definición de canal.* La palabra clave FROM (DE) se utiliza dentro de la definición CHANNEL (CANAL) para indicar el *bloque* de origen del *canal* y la palabra clave TO (A) se utiliza para indicar el *bloque* de destino. La palabra clave ENV (ENT) se utiliza para hacer referencia al *entorno.* La palabra clave WITH (CON) va seguida por la lista de las *señales* transportadas por los *canales.*
FPAR	introduce la representación de la *definición de parámetro formal.*
DURATION (DURACIÓN) TIME (TIEMPO) TIMER (TEMPORIZADOR) NATURAL (NATURAL) INTEGER (ENTERO) REAL (REAL) CHARSTRING (CADCARACTERES) CHARACTER (CARÁCTER) BOOLEAN (BOOLEANO) PID (PID)	representan los tipos de datos predefinidos
VIEW (VISIÓN)	introduce la representación de la *expresión de visión.* Se utiliza dentro de una *expresión* siempre que se utilice una *variable* declarada como VIEWED (VISTO).
SET (INICIALIZACIÓN)	introduce la representación de la *sentencia de inicialización.*
SET (REINICIALIZACIÓN)	introduce la representación de la *sentencia de reinicialización.*
SYSTEM (SISTEMA) BLOCK (BLOQUE) PROCESS (PROCESO)	introduce, la *parte calificadora* de un *identificador*

4.2.3 *Palabras clave asociadas con nodos en un gráfico de proceso*

El gráfico de proceso en la *sintaxis abstracta* está constituido por *nodos* conectados por *arcos* dirigidos.

Se escogen palabras clave que corresponden a *nodos* y los *arcos* que conectan *nodos* en la *sintaxis abstracta* se representan con arreglo al orden en que aparecen las palabras clave (véase el § 4.2.5).

START (ARRANQUE)	representa el *nodo de arranque.* Si esta palabra clave no está presente, la primera palabra clave STATE (ESTADO) que sigue a PROCESS (PROCESO), representa el *estado* de arranque.
STATE (ESTADO)	introduce la representación de uno o varios *nodos de estado.* El *conjunto de señales de conservación* agregado a un *nodo de estado* se representa mediante la palabra clave SAVE (CONSERVACIÓN) seguido de uno o varios *identificadores de señal.*

INPUT (ENTRADA)	introduce la representación de uno o varios *nodos de entrada.*
TASK (TAREA)	introduce la representación de un *nodo de tarea.*
OUTPUT (SALIDA)	introduce la representación de uno o varios *nodos de salida.* El *identificador de instancia de proceso* de destino puede ser dado facultativamente por la palabra clave TO (A) seguida por una *expresión,* que produce un *valor de identificador de instancia de proceso.* Cuando la *señal* de destino no se puede determinar de forma única, se requiere la palabra clave TO (A).
DECISION (DECISIÓN) ENDDECISION (FIN DECISIÓN)	engloba el concepto de un *nodo de decisión.* Se utiliza la palabra clave ELSE (OTRO) para representar la *respuesta* en todos los casos en que no hay designación explícita.
CREATE (CREAR)	introduce la representación de un *nodo de petición de crear.*
STOP (PARADA)	representa un *nodo de parada.*

4.2.4 *Palabras clave asociadas con arcos*

JOIN (UNIÓN)	representa un *arco* entre *nodos* que no son *nodos de estado.* El primer *nodo* se representa generalmente por la palabra clave colocada inmediatamente antes de la palabra clave JOIN (UNIÓN), el segundo *nodo* se identifica siempre dándole el mismo *identificador de etiqueta* que la palabra clave JOIN (UNIÓN). Hay algunas excepciones a esta explicación general en lo que hace referencia al primer *nodo* (véase el § 4.2.5). Si el segundo *nodo* es otra unión, el *arco* se conecta con el *nodo* a que se refiere esta unión. Si la palabra clave que tiene la *etiqueta* de unión es NEXTSTATE (ESTADO SIGUIENTE), el segundo *nodo* es el estado con el mismo *nombre* (las reglas aplicables a NEXTSTATE (ESTADO SIGUIENTE) son válidas.
	NEXTSTATE (ESTADO SIGUIENTE) representa un *arco.* El primer *nodo* del *arco* se representa por la palabra clave colocada inmediatamente antes de la palabra clave NEXTSTATE (ESTADO SIGUIENTE), el segundo *nodo* es el estado que lleva el mismo *nombre.*

4.2.5 *Representación de arcos en LED/PR*

La regla de representación de un *arco* en PR viene dada por el orden de las palabras clave.

Hay algunas excepciones a este significado general en el caso de las palabras clave JOIN (UNIÓN) y NEXTSTATE (ESTADO SIGUIENTE) como ya se indicaba en el punto precedente.

Además, cuando una palabra clave (asociada a un *nodo* o a un *arco*) sigue inmediatamente a una *respuesta,* el primer *nodo* del *arco* es la *decisión* de concordancia precedente.

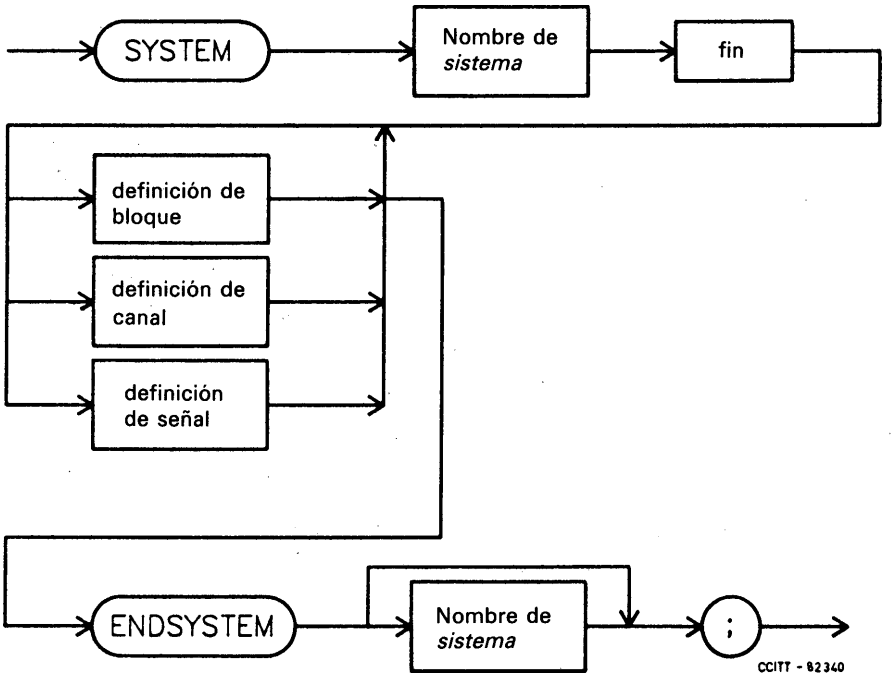
Si la palabra clave colocada inmediatamente antes de una palabra clave asociada a un nodo o a un arco es ENDDECISION (FIN DECISIÓN) los primeros *nodos* de los *arcos* se representan mediante las últimas palabras clave en todas las *cadenas de transición* de la *decisión* que no tengan sentencias de terminación.

La última palabra clave de una bifurcación de *decisión* representa un *nodo* no conectado con la palabra clave que sigue al *nombre* de resultado siguiente, pero que está conectado a la palabra clave que sigue a ENDDECISION (FIN DECISIÓN). Es evidente que esta disposición no es válida si la última palabra clave de una bifurcación de *decisión* es una sentencia de terminación.

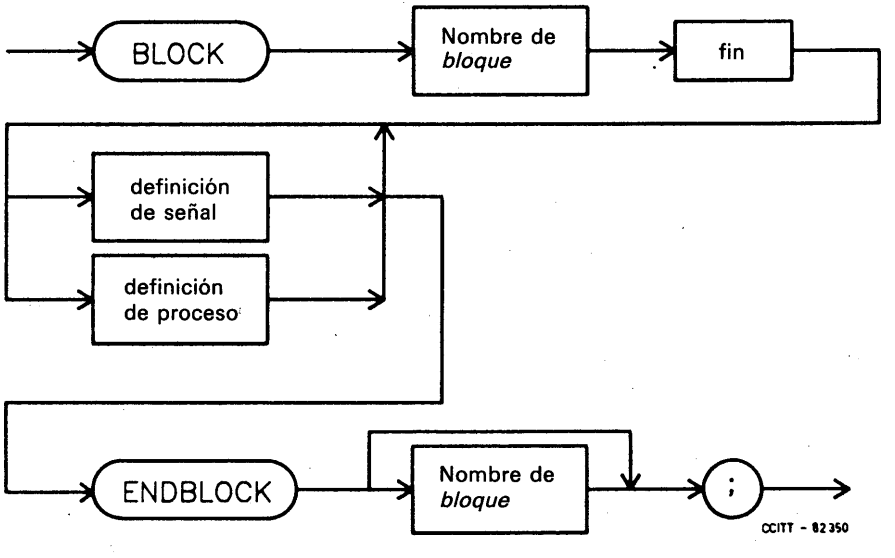
4.3 *Palabras reservadas en LED/PR*

Se han reservado algunas palabras en el LED/PR y en este caso no se pueden utilizar como *nombres.* La lista de palabras reservadas figura en el anexo C.2 a las Recomendaciones Z.100 a Z.104.

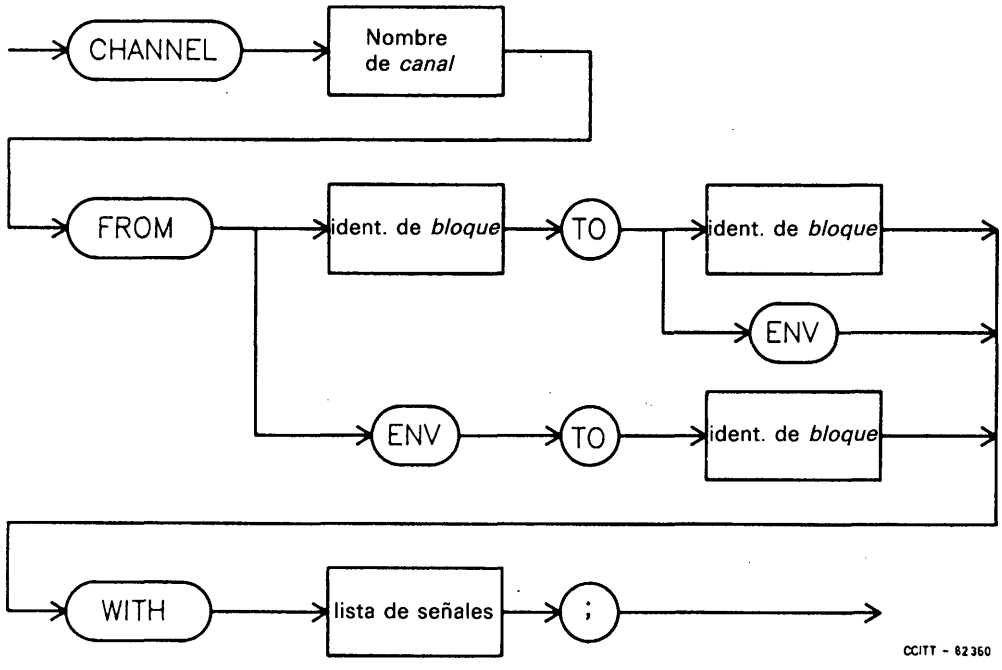
DEFINICIÓN DE SISTEMA



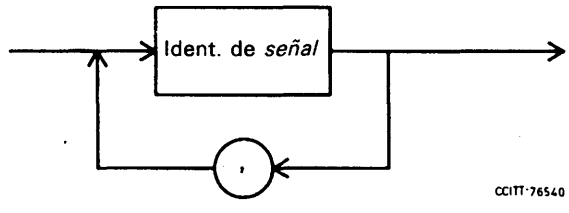
DEFINICIÓN DE BLOQUE



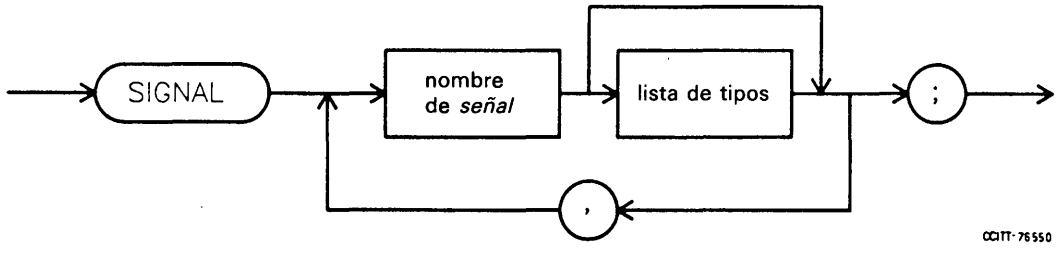
DEFINICIÓN DE CANAL



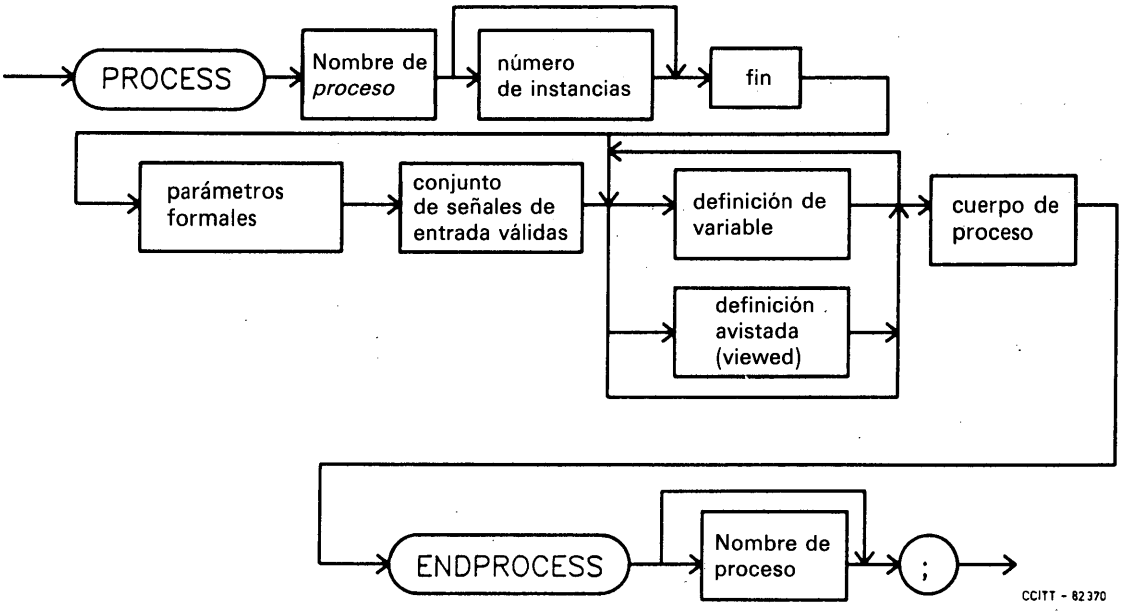
LISTA DE SEÑALES



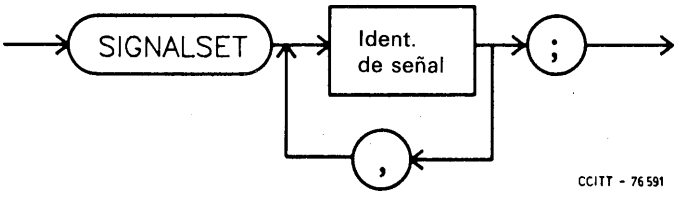
DEFINICIÓN DE SEÑAL



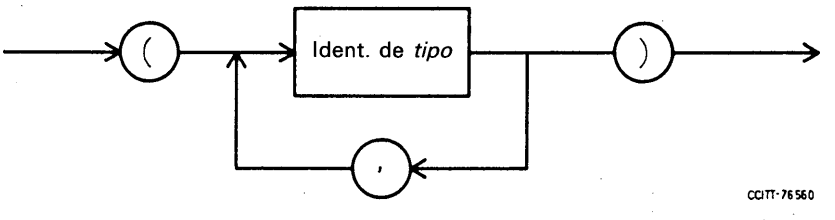
DEFINICIÓN DE PROCESO



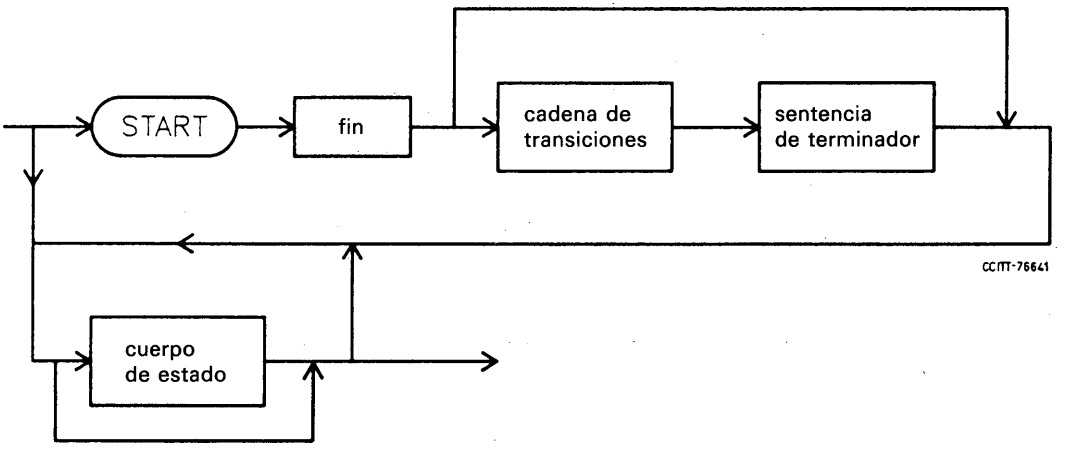
CONJUNTO DE SEÑALES DE ENTRADA VÁLIDAS



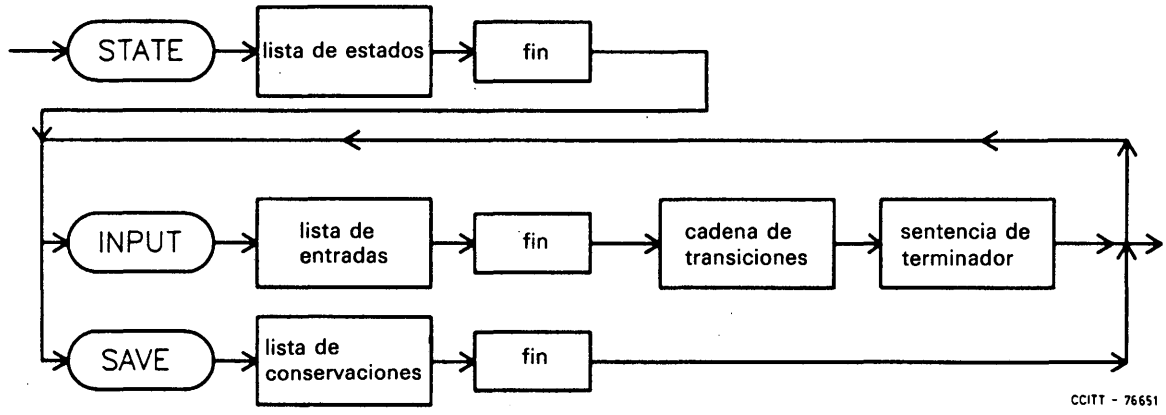
LISTA DE TIPOS



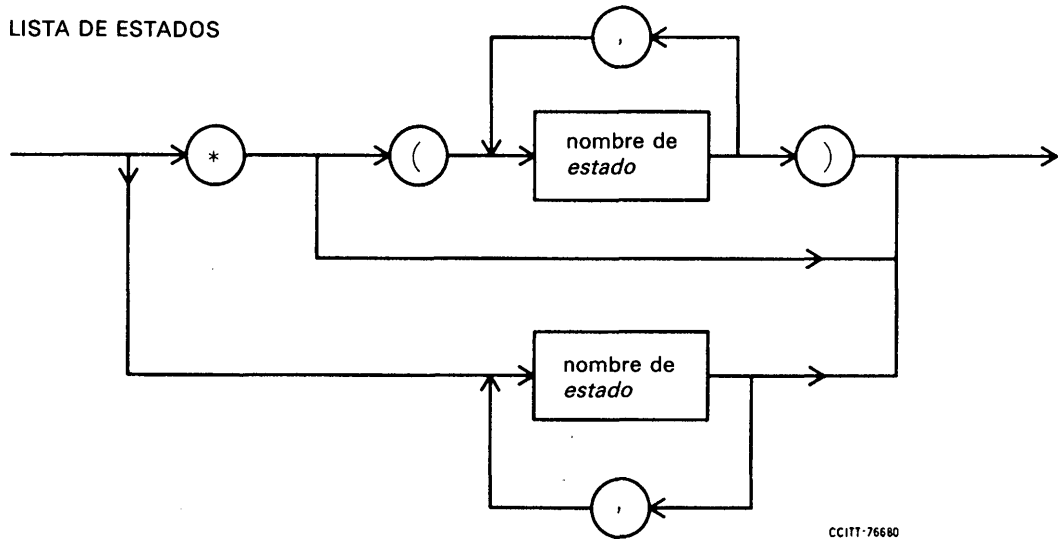
CUERPO DE PROCESO



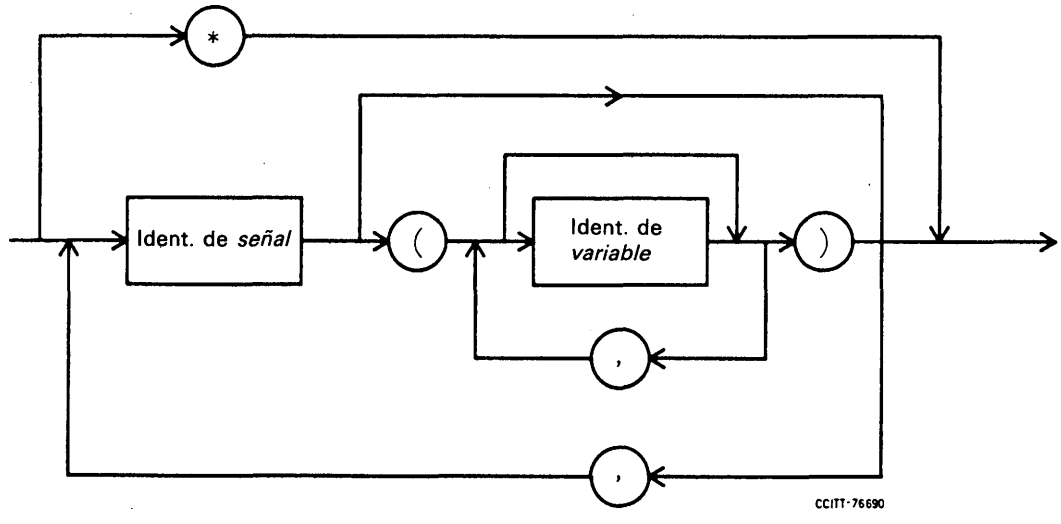
CUERPO DE ESTADO



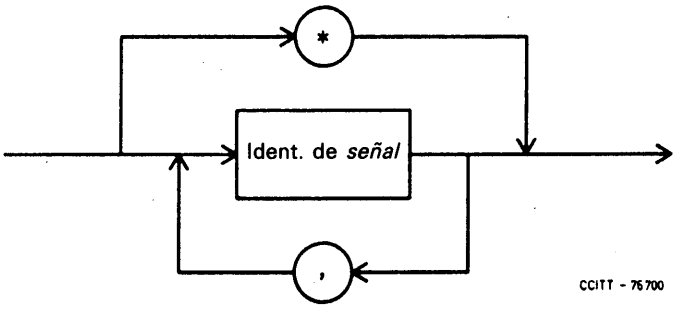
LISTA DE ESTADOS



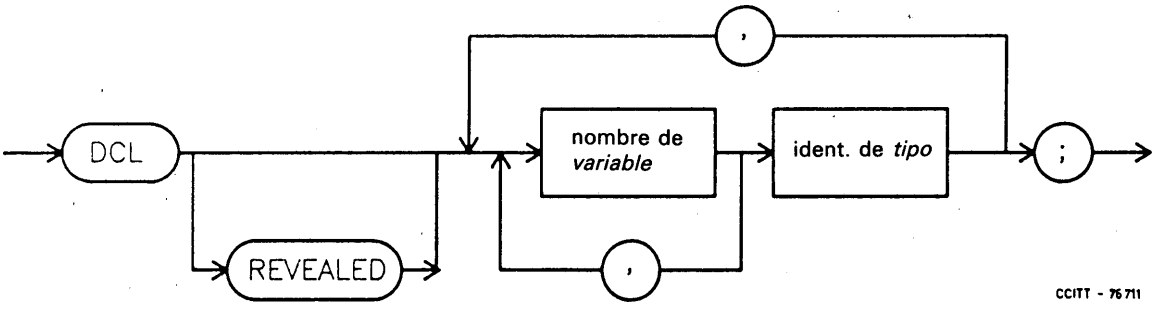
LISTA DE ENTRADAS



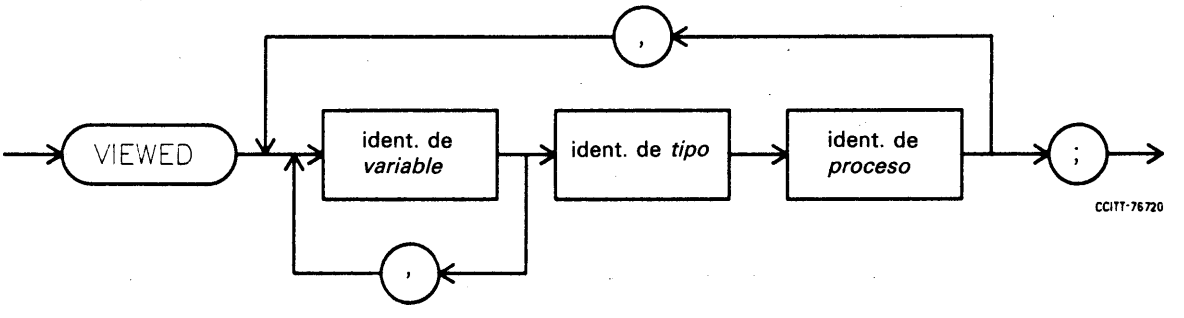
LISTA DE CONSERVACIONES



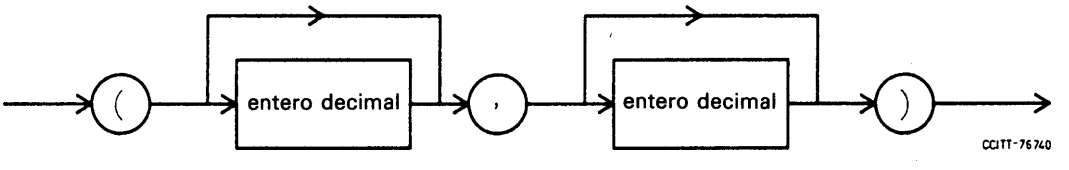
DEFINICIÓN DE VARIABLE



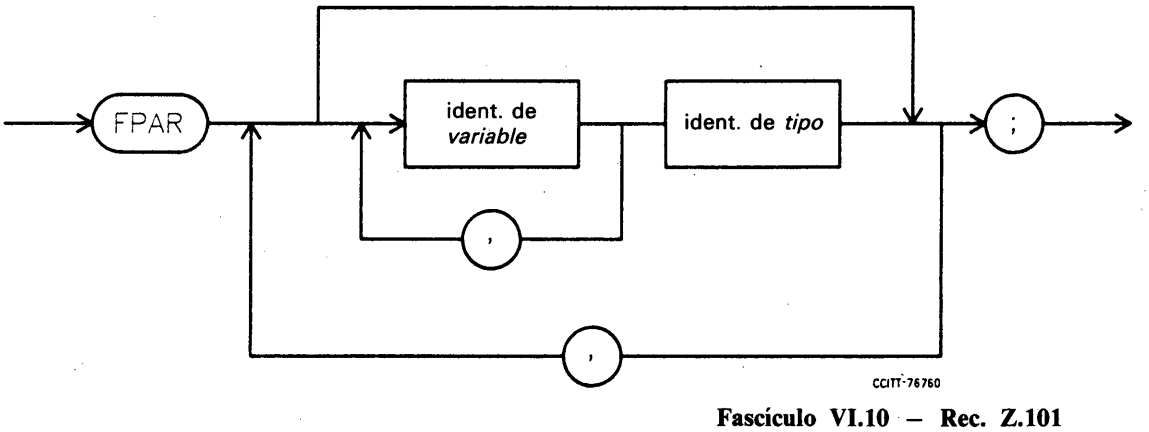
DEFINICIÓN DE AVISTAR



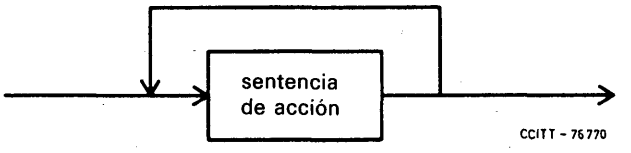
NÚMERO DE INSTANCIAS



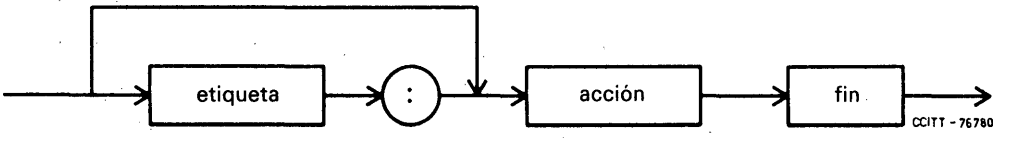
PARÁMETROS FORMALES



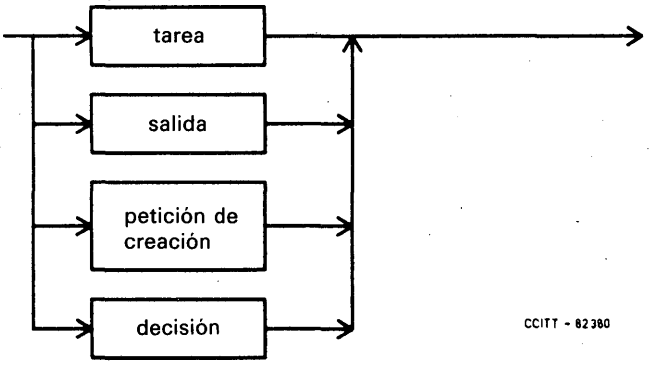
CADENA DE TRANSICIONES



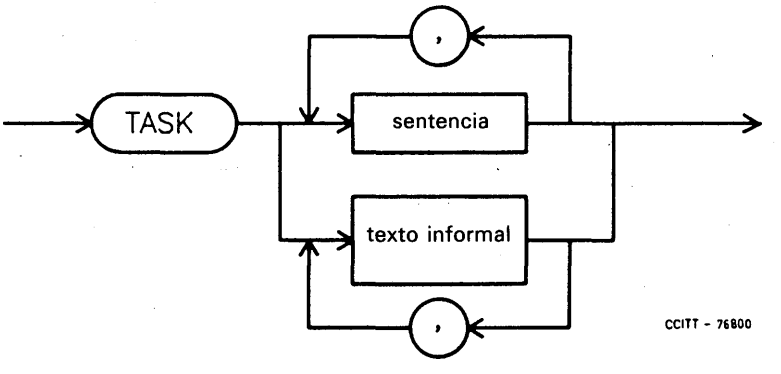
SENTENCIA DE ACCIÓN



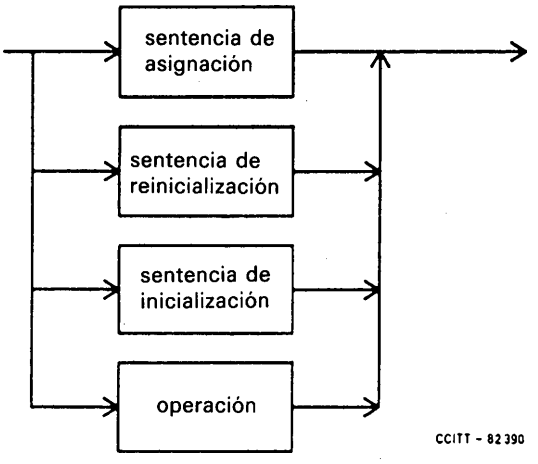
ACCIÓN



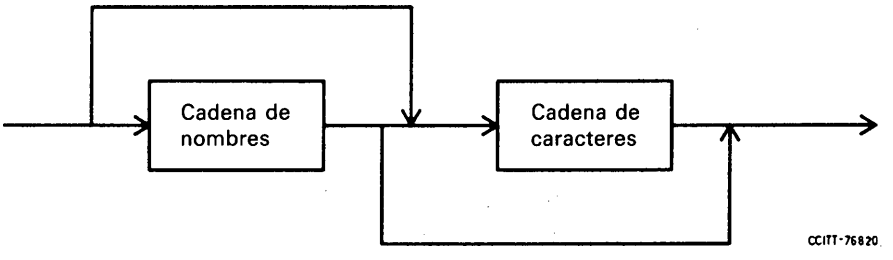
TAREA



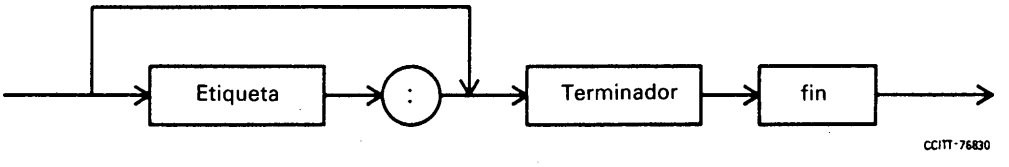
SENTENCIA



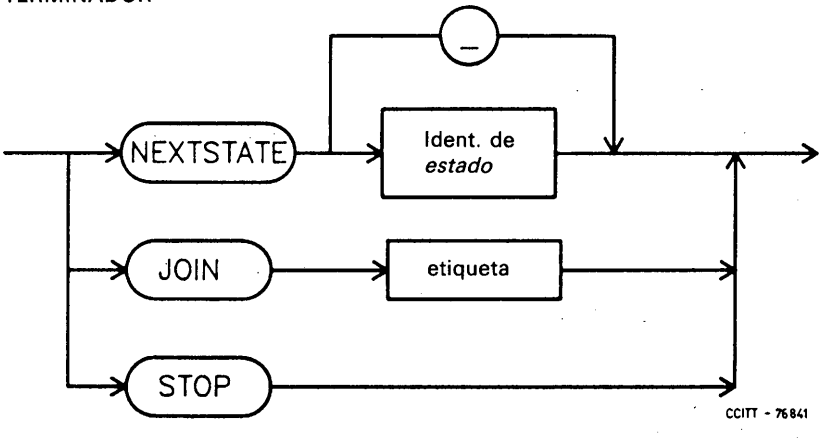
TEXTO INFORMAL



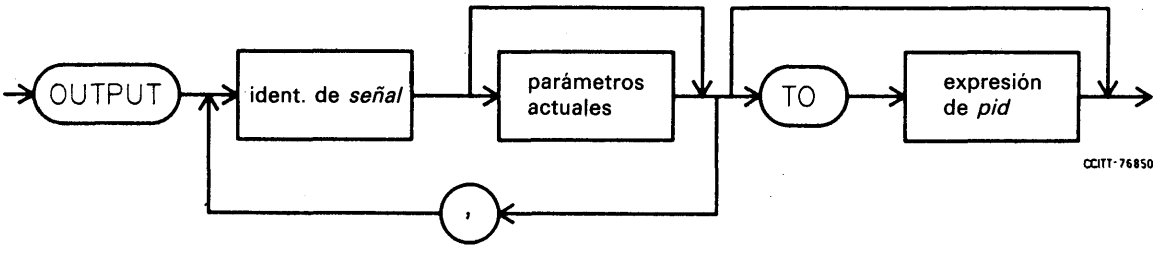
SENTENCIA DE TERMINADOR



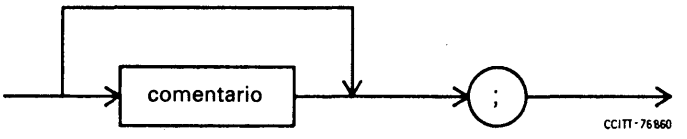
TERMINADOR



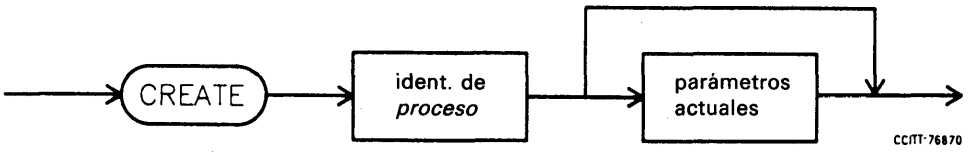
SALIDA



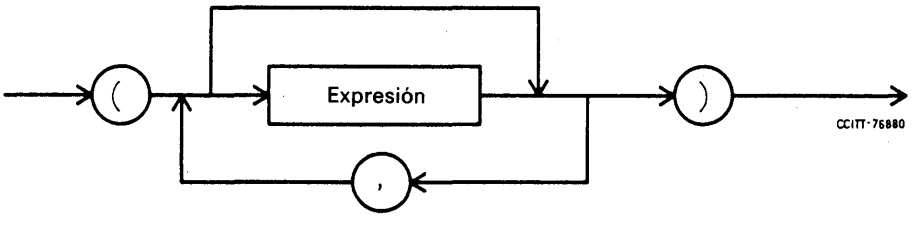
FIN



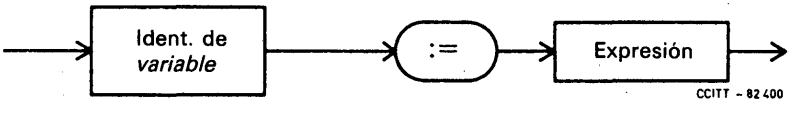
PETICIÓN DE CREACIÓN



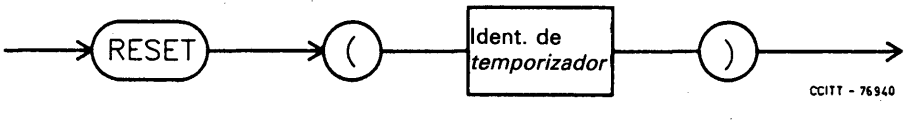
PARÁMETROS ACTUALES



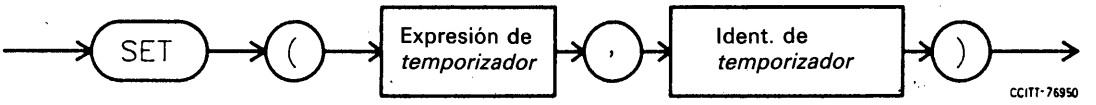
SENTENCIA DE ASIGNACIÓN



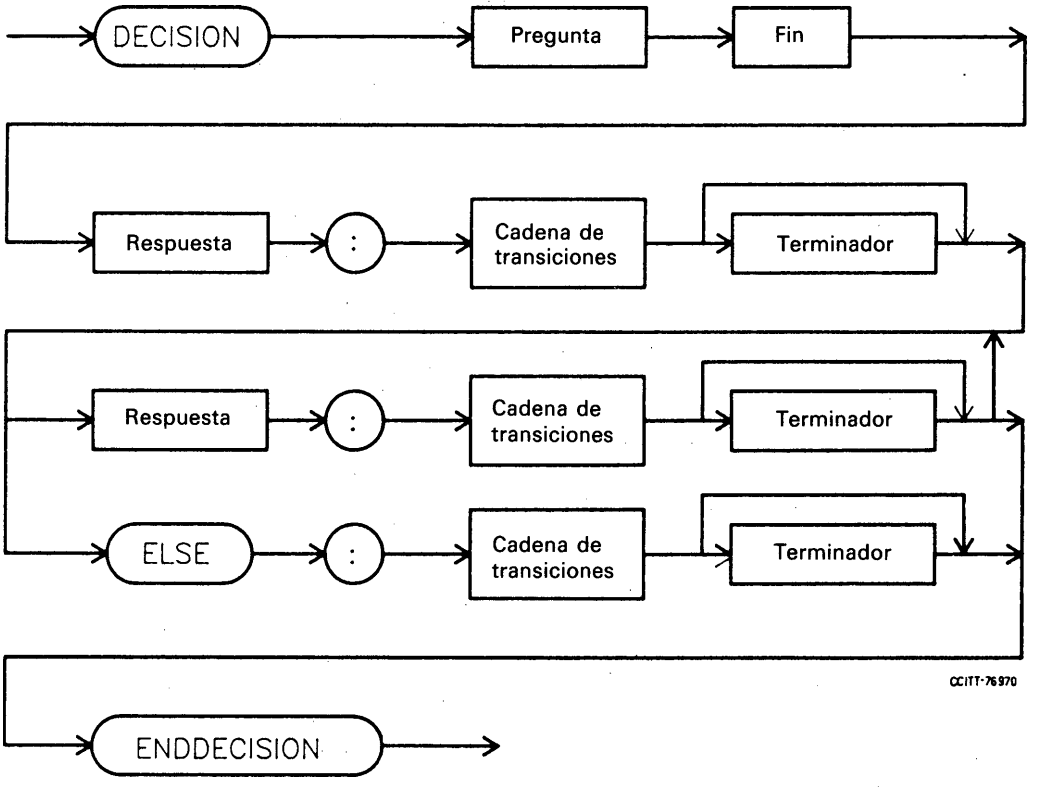
SENTENCIA DE REINICIALIZACIÓN



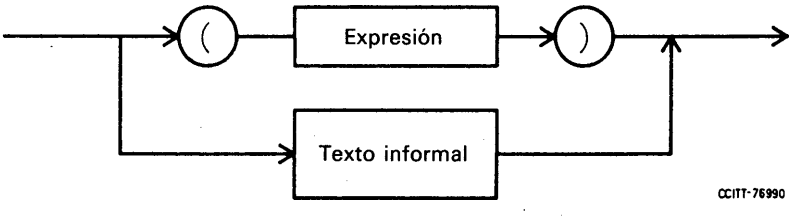
SENTENCIA DE INICIALIZACIÓN



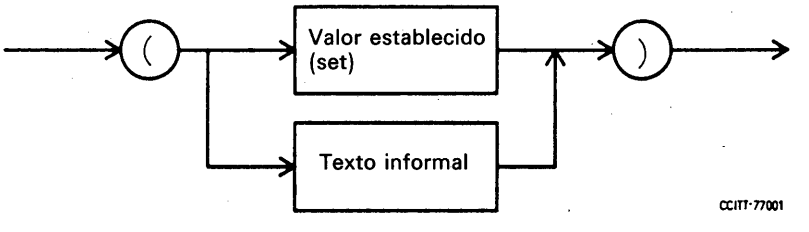
DECISIÓN



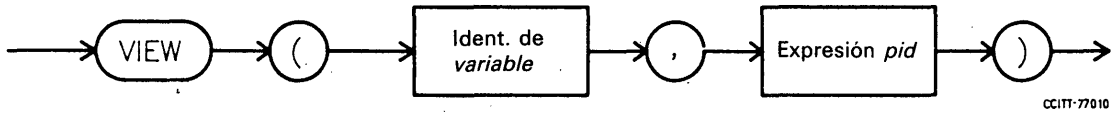
PREGUNTA



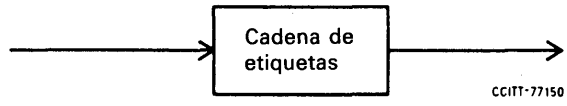
RESPUESTA



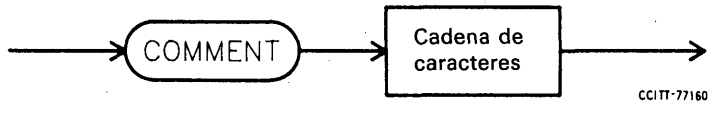
OPERADOR DE AVISTAR



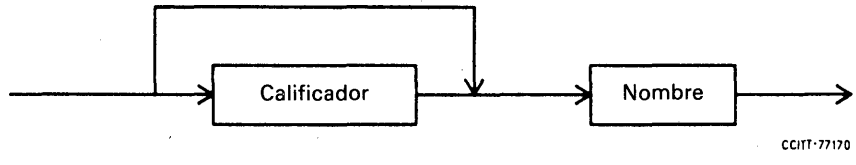
ETIQUETA



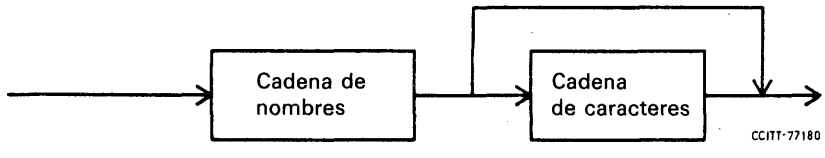
COMENTARIO



IDENT.



NOMBRE

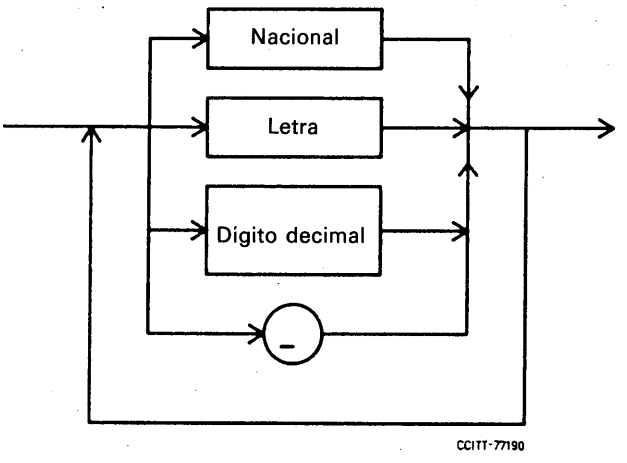


4.4.1 *Unidades lexicales*

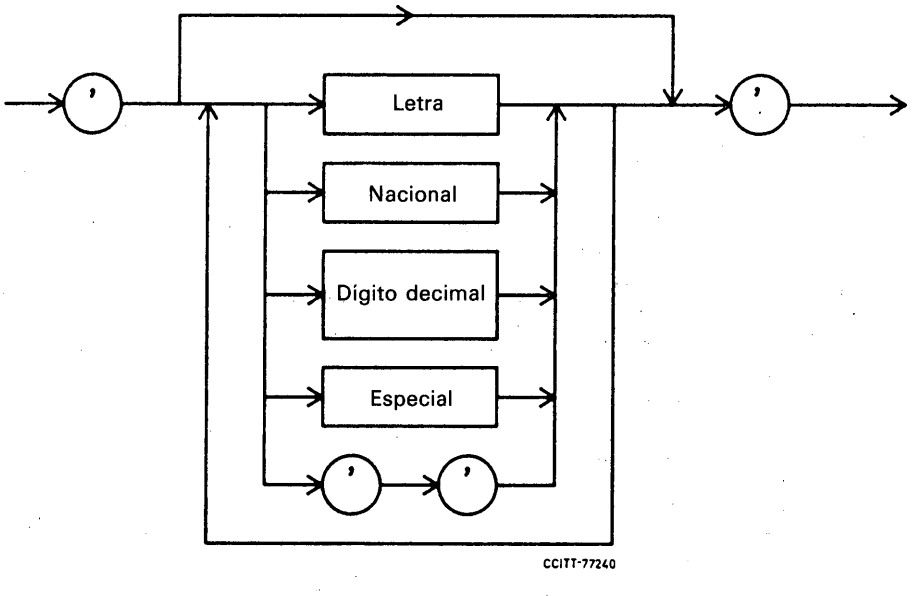
4.4.1.1 *Reglas lexicales*

- Todos los signos de puntuación [por ejemplo , . ; ' : ! = ( )] y símbolos de operación (por ejemplo +, −, *, <, >, ...) son unidades lexicales que pueden ocupar el lugar de espacios.
- Dos unidades lexicales deben ir separadas por uno o más espacios.
- Las palabras clave pertenecen a la misma categoría lexical que la cadena de nombres, y son palabras reservadas.
- Fuera de las unidades lexicales, varios espacios tienen el mismo «significado» que un espacio.
- Los caracteres de tabulación (VT, HT, CR, BS ...) pueden considerarse como espacios.
- Se considera que todas las letras y todos los nacionales pertenecen a la posición «mayúsculas», excepto dentro de una cadena de caracteres.
- Dondequiera que pueda haber espacios se pueden insertar comentarios delimitados por ' /*' y '*/'; los comentarios tienen el mismo significado que un espacio. El comentario no debe contener la secuencia especial '*/'.

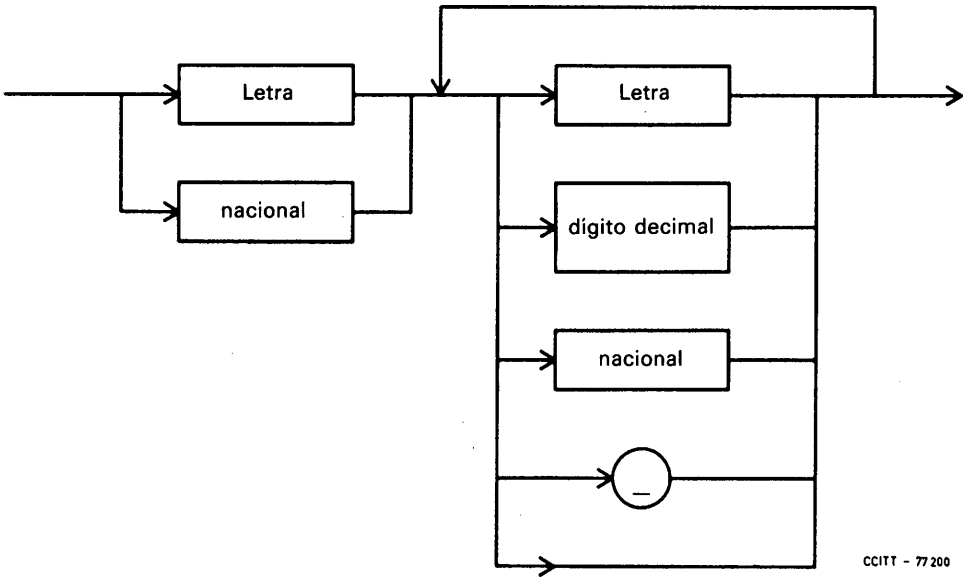
CADENA DE ETIQUETAS



CADENA DE CARACTERES

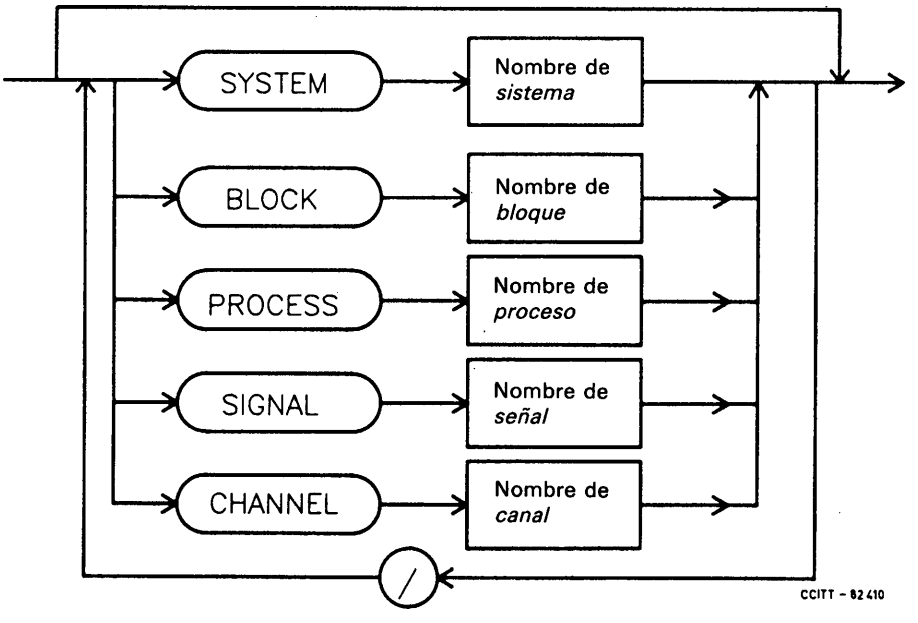


CADENA DE NOMBRES

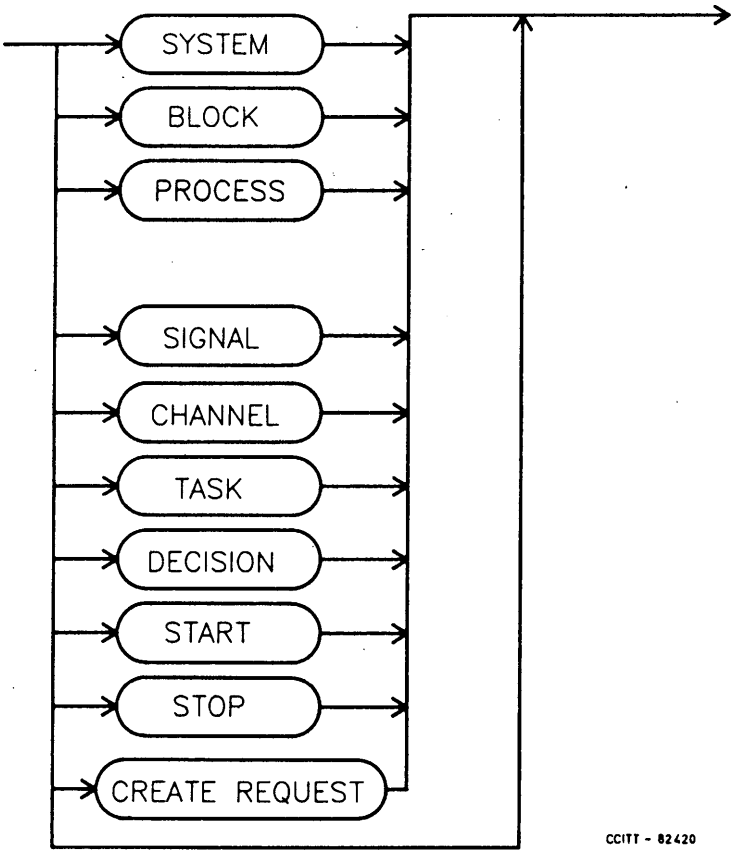


CALIFICADOR

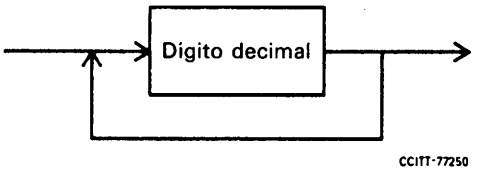
NOMBRE ESTRUCTURAL



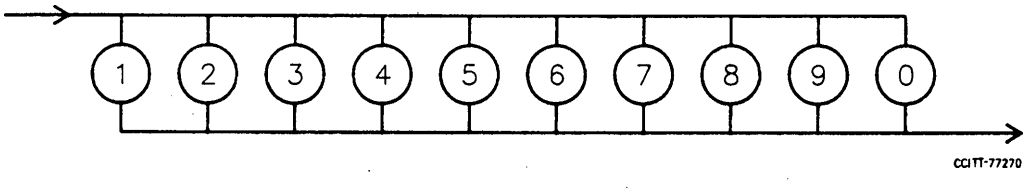
NOMBRE DE TIPO DE ENTIDAD



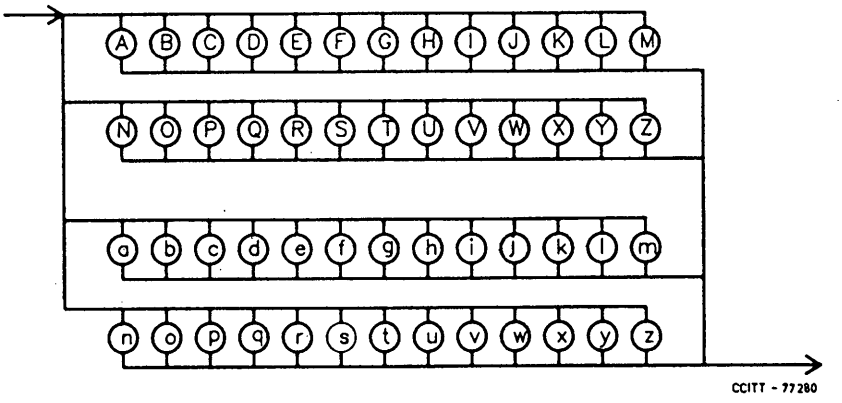
ENTERO DECIMAL



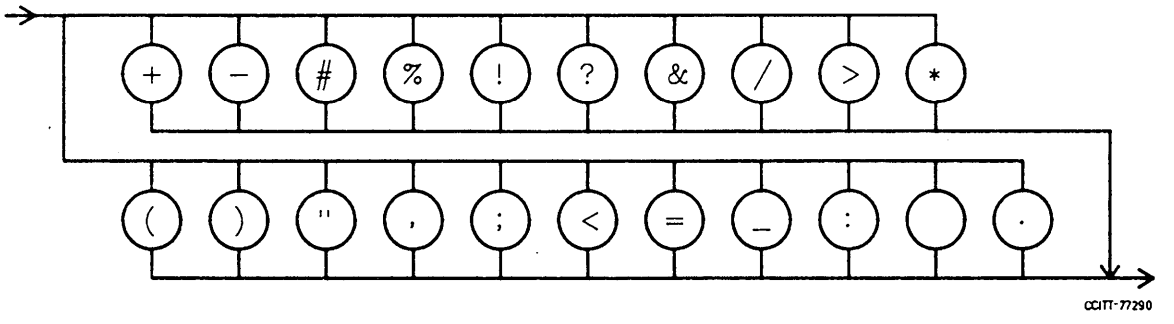
DÍGITO DECIMAL



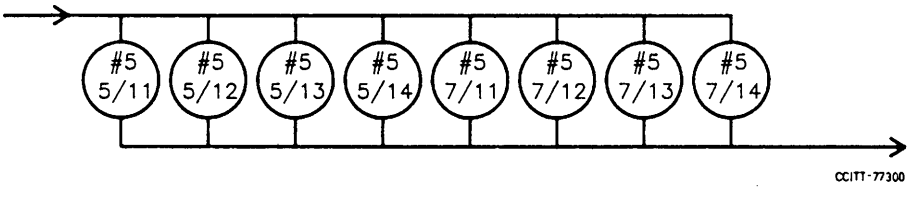
LETRA



ESPECIAL

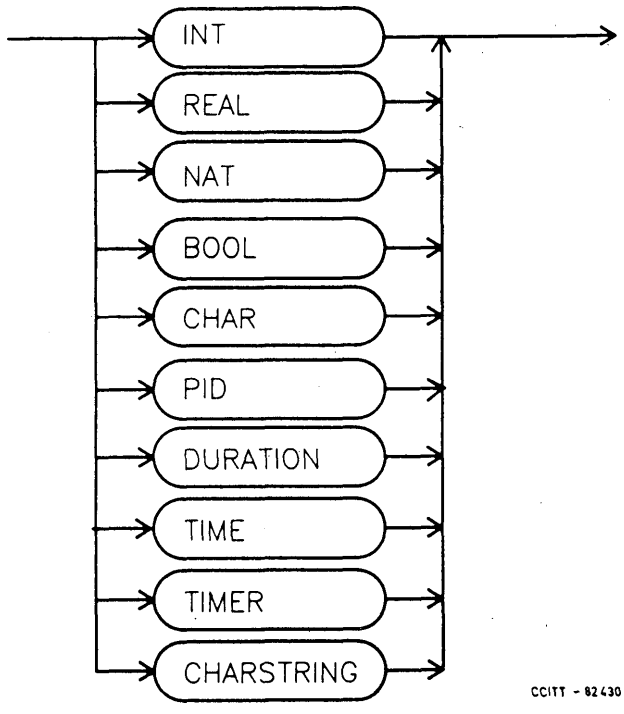


NACIONAL



Las posiciones indicadas se refieren a las posiciones del Alfabeto Internacional N.° 5 reservadas para uso nacional.

NOMBRES DE TIPOS DE DATOS PREDEFINIDOS



Recomendación Z.102

CONCEPTOS ESTRUCTURALES DEL LED

1 Introducción

En esta Recomendación se define cierto número de conceptos necesarios para el tratamiento de estructuras jerárquicas en el LED. La base de estos conceptos es el LED definido en la Recomendación Z.101 y los conceptos definidos no son sino adiciones a los definidos en la Recomendación Z.101. No hay conflicto entre las definiciones contenidas en esta Recomendación y las contenidas en las Recomendaciones Z.103 y Z.104.

Los conceptos introducidos en esta Recomendación tienen por objeto proporcionar al usuario del LED medios para describir sistemas amplios y/o complejos. El LED definido en la Recomendación Z.101, es idóneo para especificar o describir sistemas relativamente pequeños que pueden ser comprendidos y gobernados a un solo nivel de *bloques*. Cuando se trata de representar sistemas mayores o más complejos, es necesaria la partición de la *especificación* o *descripción* del sistema en unidades manejables, que puedan ser gobernadas y comprendidas independientemente. Suele ser conveniente efectuar la partición en varias etapas, lo que da como resultado una estructura jerárquica de unidades que representan el sistema.

Es también necesario utilizar conceptos estructurales a fin de especificar o describir la estructura real o requerida del sistema.

Esta Recomendación define conceptos referentes a la *partición* de:

bloques en *sub-bloques*, *sub-canales* y nuevos *canales*,

canales en *bloques* y *canales*,

procesos en *sub-procesos*.

Los conceptos son tales que la estructura jerárquica resultante, representativa del sistema, proporcionará al lector una serie de vistas generales que le permitirán formarse una apreciación general antes de descender a descripciones más detenidas. Ello significa también que los conceptos serán la base de tecnologías de diseño destinadas al «refinamiento por etapas» con la adición de información más detallada en cierto número de etapas.

2 Modelo de lenguaje común

2.1 *Consideraciones generales*

En la Recomendación Z.101 se describe un *sistema* compuesto de una serie de *bloques* conectados entre sí y a la *frontera del sistema* por medio de canales unidireccionales. En dicha Recomendación se introducen conceptos que describen la *partición* de *bloques*, *canales* y *procesos* en subcomponentes.

En un *sistema* cada *bloque* se puede dividir en uno o varios *sub-bloques*. En esta *partición* se introducen nuevos *canales* para conectar entre sí los *sub-bloques*; además los *canales* que se inician o terminan en el *bloque* dividido pueden dividirse a su vez en *sub-canales*.

El bloque A está dividido en:

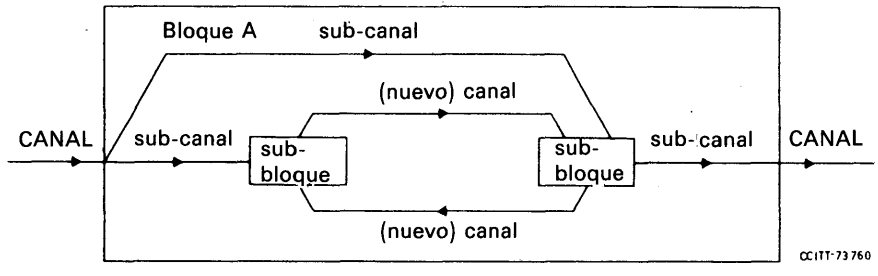


FIGURA 1/Z.102

Partición de un bloque

Un *sub-bloque* es un *bloque* y puede ser dividido. Esta *partición* puede repetirse cualquier número de veces, dando como resultado una estructura jerárquica de *bloques* y sus *sub-bloques*. Se dice que los *sub-bloques* de un *bloque* existen en el siguiente *nivel* inferior del *árbol de bloques*:

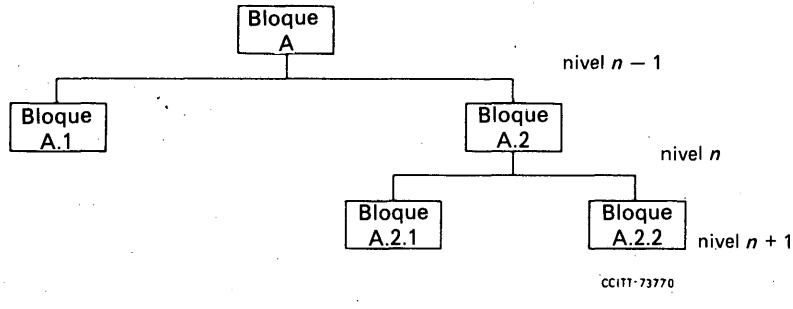


FIGURA 2/Z.102

Un árbol de bloques

Cuando un *bloque* se divide en *sub-bloques*, se suaviza laregla de la Recomendación Z.101 de que una *definición de bloque* debe contener una o varias *definiciones de proceso*, y sólo es válida para un *bloque* que no sea objeto de ulteriores particiones (es decir, los «*bloques* de hojas» del *árbol de bloques* que describe el *sistema* deben contener *procesos*). Sin embargo, si una *definición del bloque* contiene *definiciones de proceso*, las *definiciones* de los *sub-bloques* deben incluir como mínimo las *definiciones* de los *sub-procesos* definidos como resultado de la *partición* de los *procesos*.

Los *sub-canales* resultantes de la *partición de bloques* son *canales*.

Los *canales* pueden ser también objeto de partición, lo que da como resultado una serie de nuevos *bloques*, nuevos *canales*, un *canal de entrada* y un *canal de salida*:

$C\ [1_1]$

El CANAL C se particiona en:

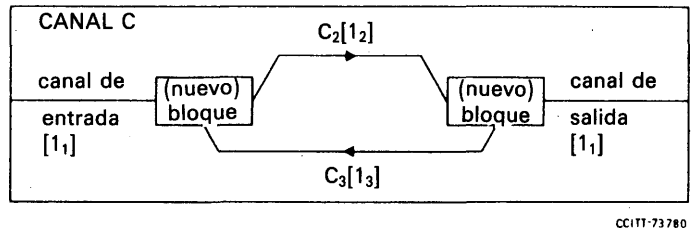


FIGURA 3/Z.102

Partición de un canal

Adviértase que la *lista de señal* asociada al *canal* original C está asociada a los canales de *entrada* y de *salida*.

Tanto la *partición* de *bloques* como la *partición* de *canales* deja inalterados los interfaces de los objetos divididos.

Una *definición de proceso* se puede dividir en una serie de *definiciones de sub-procesos*. Estas dos descripciones del comportamiento son alternativas en el sentido de que cuando se interpreta la *definición del sistema*, se interpreta bien la serie de *definiciones de sub-procesos* o la *definición de proceso*. La *definición de proceso* se debe considerar como descripción alternativa con relación a la serie de *definiciones de sub-procesos*.

Un *sub-proceso* es un *proceso*, y a su vez se puede dividir en *sub-procesos*. La jerarquía resultante de *procesos* se representa en un *árbol de procesos*.

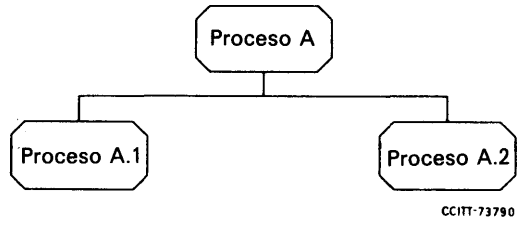


FIGURA 4/Z.102

Relación proceso-sub-proceso

Como los *sub-procesos* son *procesos*, el comportamiento descrito por el *proceso* dividido se divide en una serie de «sub-comportamientos» concurrentes. Se necesitan métodos para garantizar que esta *partición* es correcta, pero no forman parte del LED.

La *partición* de *bloques* en *sub-bloques* y de *procesos* en *sub-procesos* puede hacerse al mismo tiempo. De la misma manera que un *proceso* tiene que estar contenido en un *bloque*, los *sub-procesos* tienen que estar contenidos en *sub-bloques* del *bloque* que contiene el *proceso* dividido. Los *bloques* y *procesos* divididos pueden considerarse como estructuras separadas relacionadas entre sí:

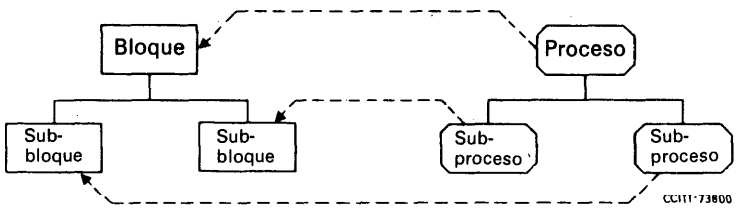


FIGURA 5/Z.102

Relación entre árboles de bloques y árboles de procesos

Si un *bloque* se divide, cualquier *proceso* que contenga puede también dividirse, en cuyo caso todos sus *sub-procesos* deben aparecer en los *sub-bloques* del *bloque*. Si el *proceso* no se divide, debe aparecer en uno de los *sub-bloques*. Para describir con más detalle el comportamiento de los *sub-bloques*, se pueden también incluir nuevos *procesos* y señales en los *sub-bloques*, independientemente de si eran *procesos* existentes en el *bloque*. Sin embargo, los *bloques*-«hoja» deben contener *procesos*.

Si en la *representación* total de la *partición* de un *sistema* aparecen *definiciones de proceso* en más de un *nivel*, pueden existir varios subconjuntos coherentes de la representación. Un subconjunto coherente es una selección de las *definiciones de bloque* y *definiciones de proceso* de la representación total tal que:

a) contiene el *nivel* más alto en el *árbol de bloques*;

b) si contiene un *bloque*, debe contener también su ascendiente;

c) si contiene un *sub-bloque* de un *bloque* debe contener también todos los demás *sub-bloques* de ese *bloque*;

d) todos los *bloques*-«hoja» de la estructura resultante contienen *procesos*.

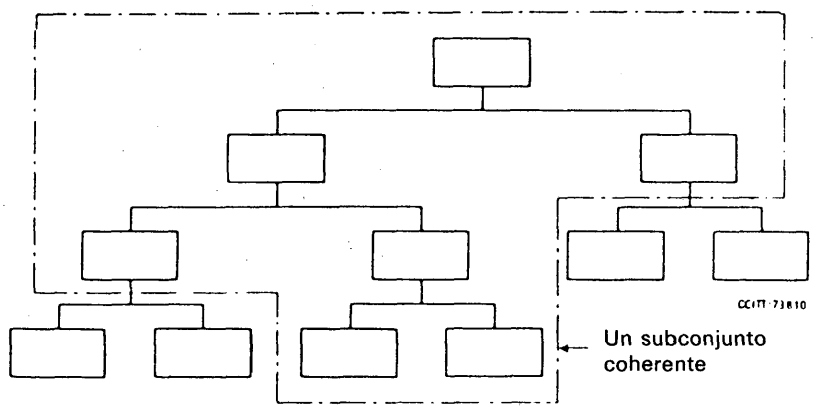


FIGURA 6/Z.102

Subconjunto coherente de una representación del sistema

Si en una *definición del sistema* aparecen *procesos* a más de un nivel, pueden hallarse entonces varios subconjuntos coherentes de las representaciones. Estos conjuntos representan descripciones alternativas del *sistema*, con varios grados de detalle. Se pueden utilizar para proporcionar al lector descripciones generales y descripciones detalladas. Pueden ser elegidos de manera que permitan la interpretación del sistema con varios niveles de abstracción. Si existen varias representaciones alternativas del comportamiento, todas, salvo la más detallada, se deberán considerar descripciones generales del comportamiento.

2.2 *Sintaxis abstracta*

Esta *sintaxis abstracta* se basa en la *sintaxis abstracta* dada en la Recomendación Z.101. Sólo se exponen aquí las adiciones a las definiciones que figuran en la Recomendación Z.101.

Definición de bloque

Una *definición de bloque* puede también contener una *definición de bloque de parte interna* y en tal caso no necesita contener *definiciones de proceso.*

Definiciones de bloque de parte interna

La *definición de bloque de parte interna* contiene una *definición de subestructura de bloques* y una *definición de subestructura de procesos* para cada una de las *definiciones de proceso* contenidas en la *definición de bloque.* Puede también contener *definiciones de señal* y *definiciones de tipo de datos.*

Definición de subestructura de bloques

Una *definición de subestructura de bloques* puede contener una o varias *definiciones de sub-bloque* y una o varias *definiciones de canal.*

Para cada uno de los puntos extremos de terminación de los *sub-canales* de *bloque* circundante debe haber como mínimo una *definición de sub-canal* que tenga ese punto extremo como punto extremo de origen y la inversa sucede con todos los puntos extremos de *sub-canal* de origen de *bloque* circundante. La unión de las *listas de señal* de las *definiciones de sub-canal* que tienen el mismo punto extremo que un *sub-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 sub-canal* que se originan en un punto extremo de terminación deben ser discontinuas.

Todas las *definiciones de canal* contenidas en la *definición de subestructura de bloques* deben contener *sub-bloques* a los puntos extremos del *canal* del *bloque* circundante a los *sub-bloques* entre sí.

Definición de sub-bloque

Una definición de *sub-bloque* es una definición de *bloque.*

Definición de canal

Una *definición de canal* puede también contener una *definición de subestructura de canal.*

Definición de sub-canal

Una definición de *sub-canal* es una definición de *canal.*

Definición de subestructura de canales

Una *definición de subestructura de canales* contiene dos o más *definiciones de canal,* una o varias *definiciones de bloque* y puede contener *definiciones de señal.*

Todas las *definiciones de canal* contenidas en la *definición de subestructura de bloques* deben conectar *sub-bloques* entre sí, y todas las *definiciones de sub-canal* deben conectar puntos extremos de *canal* del *bloque* circundante a *subcanales.*

Definición de subestructura de procesos

Una *definición de subestructura de procesos* está asociada con un *nombre de proceso* y contiene uno o varios *nombres de sub-proceso,* cada uno de los cuales está asociado a un *nombre de sub-bloque.*

El *nombre de sub-proceso* asociado debe ser el nombre de una *definición de proceso* contenida en la *definición de bloque* circundante. Los *nombres de proceso* contenidos deben ser *nombres* de *definiciones de sub-proceso* contenidos en la *definición de bloque* que tenga el *nombre de sub-bloque* asociado.

Cada *nombre de señal* del conjunto de *señales de entrada válidas* del *proceso* asociado debe aparecer exactamente en uno de los conjuntos de *señales de entrada válidas* de las *definiciones de sub-proceso* que tienen los *nombres de sub-proceso* contenidos. Cada *nombre de señal* agregado al *nodo de salida* del *proceso* asociado debe ser agregado como mínimo a un *nodo de salida* de las *definiciones de sub-proceso* que tengan los *nombres de sub-proceso* contenidos.

Definición de sub-proceso

Una definición de *sub-proceso* es una *definición de proceso.*

2.3 *Interpretación*

Las reglas de interpretación dadas a continuación se definen como adiciones al correspondiente conjunto de reglas de la Recomendación Z.101.

Canal

Si una *definición de canal* contiene una *definición de subestructura de canales* se puede interpretar el *canal* de acuerdo con lo definido en la Recomendación Z.101, o se puede interpretar la definición de subestructura de canales.

Si se interpreta la *definición de subestructura de canales*, cada *señal* suministrada al punto extremo de origen del *canal* se entrega a la *subestructura de canales*, y cada *señal* suministrada por la *subestructura de canales* se entrega al punto extremo de terminación de *canal*.

Subestructura de canales

Una *señal* suministrada a la *subestructura de canales* se entrega al *canal de entrada* y una *señal* suministrada por el *punto extremo de terminación* del *canal de salida* se suministra al *canal* circundante.

Bloque

Si una *definición de bloque* contiene una o varias *definiciones de proceso* y también una *definición de bloque de parte interna*, el *bloque* puede ser interpretado de acuerdo con lo definido en la Recomendación Z.101 o se puede interpretar *el bloque de parte interna*. Si el *bloque* no contiene ninguna *definición de proceso*, se debe interpretar el *bloque de parte interna*.

Si se interpreta el *bloque de parte interna*, todas las *señales* suministradas al *bloque* se darán al *bloque de parte interna* y todas las *señales* suministradas por el *bloque de parte interna* se suministrarán de nuevo a los *canales* que conducen desde el *bloque* de la misma manera que si fueran suministradas desde un *proceso* contenido en el *bloque*.

Bloque de parte interna

Cada *proceso* del *bloque* circundante es sustituido por una *subestructura de proceso*.

Si una *señal* dada al *bloque de parte interna* desde el *bloque* circundante se dirige a un *proceso*, la *señal* se da a la *subestructura de proceso* o en otro caso a la *subestructura de bloque*.

Una *señal* suministrada por la *subestructura de bloques* se dará al *bloque* circundante. Si esa *señal* fue enviada desde un *proceso* que aparece como un *sub-proceso* o una *subestructura de procesos* el atributo emisor de la *señal* se cambiará por el *identificador de instancia de proceso* del *proceso* sustituido.

Subestructura de bloques

Una *subestructura de bloques* contiene *bloques* y *canales*, según la *definición de subestructura de bloques*, que se interpretan según las reglas definidas para *bloques* y *canales*.

Subestructura de proceso

Una *instancia de subestructura de proceso* sustituye a una *instancia de proceso* de la *definición de proceso* referenciada. Denota también una serie de *instancias de sub-procesos*, una *instancia* para cada *nombre de sub-proceso* de la *definición*. Cada *sub-proceso* se atribuye al *bloque* que lleva el *nombre de bloque* asociado.

Cada *señal* dada a la *subestructura de proceso* se redireccionará al *sub-proceso* que tiene el *nombre de señal* del conjunto de *señales de entrada* válidas y se dará a la *subestructura de bloques* del *bloque* circundante.

3 Sintaxis gráfica

La sintaxis gráfica siguiente es una adición a la sintaxis definida en la Recomendación Z.101. La adición cubre la representación de la estructura y la *partición* de un *sistema*.

Una vista general de la estructura de un *sistema* viene dada por el *diagrama de árbol de bloques*. La *partición* de *bloques*, *procesos* y *canales* en subcomponentes se representa en el *diagrama interacción de bloques* y en el *diagrama de subestructura de canales*.

El conjunto de documentos y diagramas que describen el sistema puede ser grande. Es esencial que los documentos se relacionen entre sí por medio de referencias y títulos adecuados. Sin embargo, los medios sintácticos de proceder a esta relación no forman parte de la sintaxis gráfica del LED.

3.1 *Diagrama de árbol de bloques*

El *diagrama de árbol de bloques* tiene por objeto dar una visión general de la estructura de un *sistema*, es decir, la *partición* de *sistema* en una estructura jerárquica de *bloques*. En el *diagrama de interacción de bloques* se dan detalles sobre cómo se conectan los *bloques* por medio de *canales*.

Una parte del diagrama se puede utilizar también para dar una visión general de cómo un *bloque* está dividido en *sub-bloques*. En este caso, el *bloque* dividido aparece como la casilla raíz.

3.1.1 *Símbolos*

El símbolo utilizado en un *diagrama de árbol de bloques* es una casilla que representa un *sistema* o un *bloque*. El *nombre* del objeto representado debe aparecer dentro de la casilla.

Cada *bloque* (casilla) está conectado hacia abajo hacia sus *sub-bloques* (casillas) para formar un árbol jerárquico, como en el ejemplo dado a continuación en la figura 7/Z.102.

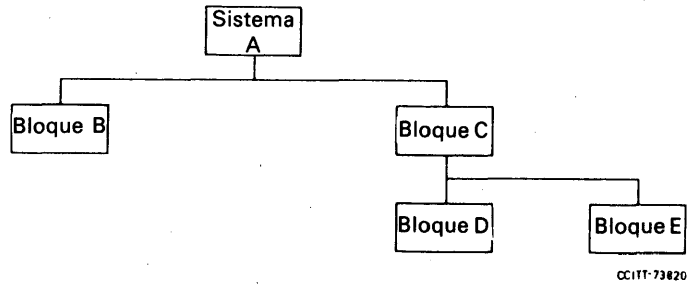


FIGURA 7/Z.102

Ejemplo de diagrama de árbol de bloques

3.1.2 *Relación con la sintaxis abstracta del LED*

La estructura mostrada tiene su equivalencia en la *definición de sistema* y en las *definiciones de subestructura de bloque* de los *bloques* del *sistema*.

3.1.3 *Convenciones gráficas*

El árbol se dibujará preferentemente de forma que los *bloques* del mismo *nivel* aparezcan contiguos en la representación.

Como un *diagrama de árbol de bloque* de un *sistema* grande será también grande, quizá sea conveniente en dividirlo en varios diagramas. Esta división será tal que el primer diagrama, que tenga el *sistema* como raíz, aparezca cortado con el fin de que la serie de los *bloques* divididos siguientes aparezca no dividida. En los diagramas siguientes estos *bloques* aparecen como raíces. Por ejemplo, en la figura 8/Z.102 el diagrama de la figura 7/Z.102 se ha dividido en dos diagramas.

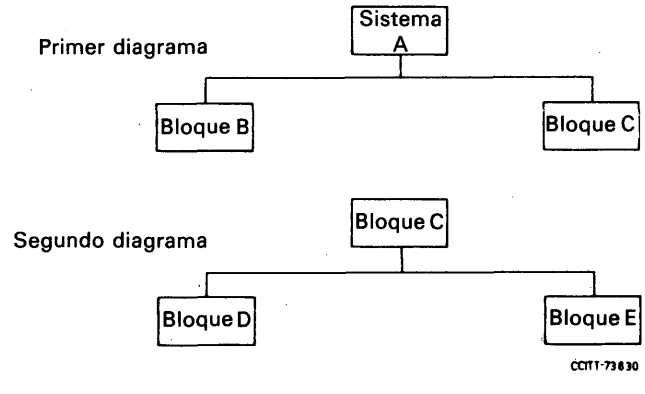


FIGURA 8/Z.102

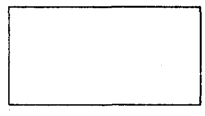
Ejemplo de división de un diagrama de árbol de bloques en varios diagramas

3.2 *Diagrama de interacción de bloques*

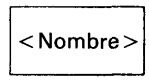
Este diagrama representa la *partición* de un *bloque* en *sub-bloques*, *sub-canales* y (nuevos) *canales*. El diagrama tiene básicamente la misma forma que el *diagrama de interacción de bloques* presentado en la Recomendación Z.101, que representa la partición de un *sistema* en *bloques* y *canales*.

3.2.1 *Símbolos*

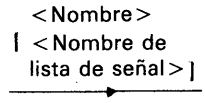
Los símbolos utilizados para representar un *diagrama de interacción de bloques* se muestra a continuación en la figura 9/Z.102.



Marco
Rodea el diagrama y representa la frontera del *bloque* o *sistema* dividido.



Símbolo de sub-bloque
Esta casilla representa un *sub-bloque*. El *nombre* del *sub-bloque* debe aparecer dentro del símbolo.



Símbolo de canal

Este símbolo representa un *canal* o un *sub-canal*. El *nombre* del *canal* debe aparecer al lado del símbolo. Puede ir asociada una *lista de señal* opcional (véase el § 2.2.1 de la Recomendación Z.101).

FIGURA 9/Z.102

Símbolos utilizados en un diagrama de interacción de bloques

Además de estos símbolos, pueden aparecer en el diagrama *definiciones de señal*, utilizando la sintaxis LED/PR.

Las reglas para conectar los símbolos son las mismas que para el *diagrama de interacción de bloques* (véase la Recomendación Z.101) con la única excepción de que en el título del diagrama debe quedar claro que el diagrama es un *diagrama de interacción de bloques* para un *bloque*. Un sencillo ejemplo de *diagrama de interacción de bloques* se da a continuación en la figura 10/Z.102.

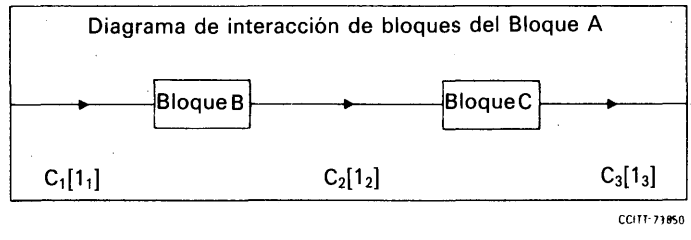


FIGURA 10/Z.102

Ejemplo de diagrama de interacción de bloques

3.2.2 *Relación con la sintaxis abstracta del LED*

Un *diagrama de interacción de bloques* representa una *definición de subestructura de bloques*. Las definiciones contenidas en la *definición de subestructura de bloques* están representadas por los símbolos de *bloque* y de *canal* junto con *definiciones de señal* dadas en *LED/PR*.

3.2.3 *Convenciones gráficas*

Las convenciones gráficas descritas para el *diagrama de interacción* definidas en la Recomendación Z.101 se aplican al *diagrama de interacción de bloques.*

Además, suele resultar útil describir varios *niveles* de *partición de bloques* en un diagrama. Para ello se sustituye un *bloque* o *símbolo* en un diagrama por el *diagrama de interacción de bloques* correspondiente a ese *bloque.* Un ejemplo de ello aparece a continuación en la figura 11/Z.102.

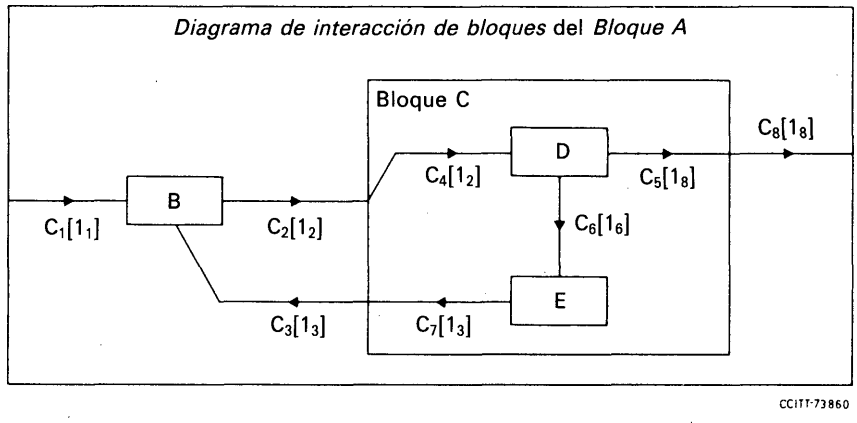


FIGURA 11/Z.102

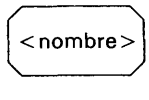
Ejemplo de un diagrama de interacción de bloques agrupado

3.3 *Diagrama de árbol de procesos*

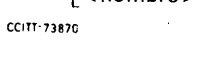
Un *diagrama de árbol de procesos* describe la *partición* de un *proceso* en *sub-procesos* y la atribución de los *sub-procesos.*

3.3.1 *Símbolos*

Los símbolos utilizados para componer un *diagrama de árbol de procesos* se representan en la figura 12/Z.102.



Símbolo de proceso
El símbolo representa un *proceso,* cuyo *nombre* debe aparecer dentro del símbolo



Símbolo de atribución
El símbolo de comentario se utiliza para indicar donde se ha atribuido el *proceso* o *sub-proceso.* El *nombre* del comentario debe ser el nombre del *bloque* al que se ha atribuido el *proceso.*

FIGURA 12/Z.102

Símbolos utilizados en el diagrama de árbol de procesos

Cada proceso se conecta hacia abajo con sus *sub-procesos* para formar un árbol jerárquico, como en la figura 13/Z.102.

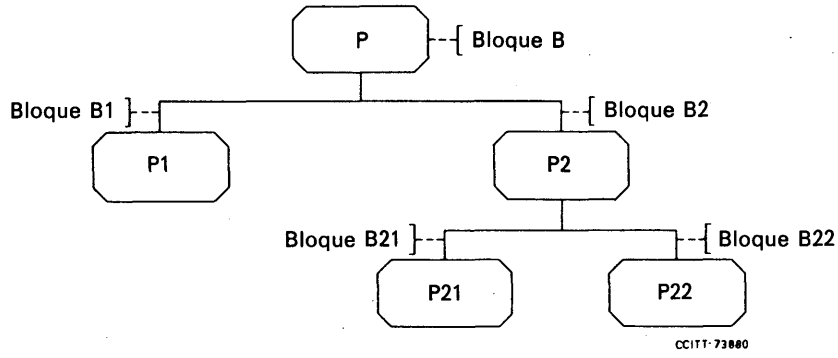


FIGURA 13/Z.102

Ejemplo de diagrama de árbol de procesos

3.3.2 *Relación con la sintaxis abstracta del LED*

El diagrama representa una *definición de subestructura de proceso.* El *nombre* que figura en el símbolo raíz es el *nombre de proceso* asociado, y los *nombres* que figuran en los símbolos hoja son los *nombres de sub-proceso* contenidos. Los *nombres de bloque* de los símbolos de atribución son los *nombres del sub-bloque* asociado.

3.3.3 *Convenciones gráficas*

El árbol se dibujará preferentemente de forma que los *procesos* situados en el mismo *nivel de partición* aparezcan contiguos en el diagrama.

Si el *diagrama de árbol de procesos* es grande, podría ser conveniente dividirlo en varios diagramas. Esta división será tal que el primer diagrama aparezca cortado con el fin de que la serie de los *procesos* divididos siguientes aparezca no dividida. En los diagramas siguientes estos *procesos* aparecen como raíces. Por ejemplo, en la figura 14/Z.102 el diagrama de la figura 12/Z.102 se ha dividido en dos diagramas.

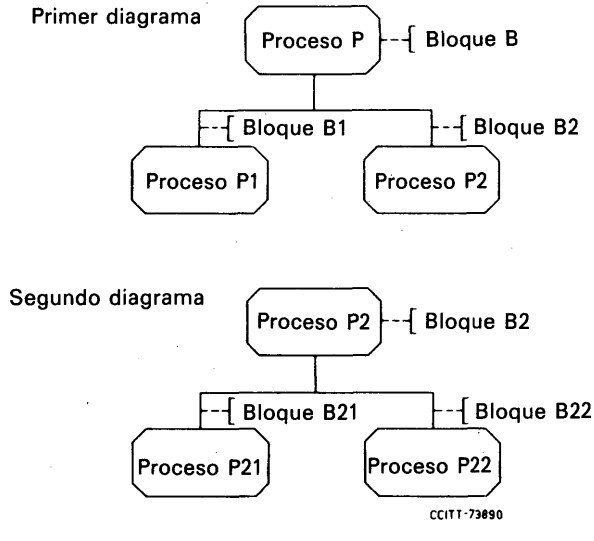


FIGURA 14/Z.102

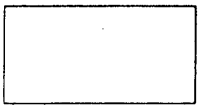
Ejemplo de división de un diagrama de árbol de procesos en varios diagramas

3.4 *Diagrama de subestructura de canales*

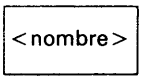
Este diagrama representa la *partición* de un canal en subcomponentes. Como los componentes son *bloques* y *canales*, el diagrama se asimila a un *diagrama de interacción de bloques.*

3.4.1 *Símbolos*

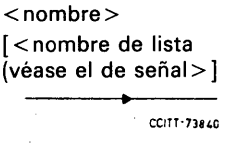
Los símbolos utilizados para representar un *diagrama de subestructura de canales* son los representados en la figura 15/Z.102.



Marco
Rodea el diagrama y representa el *canal* dividido.



Símbolo de bloque
Representa un *bloque*. El *nombre* del *bloque* debe aparecer dentro del símbolo.



Símbolo del canal
Este símbolo representa un *sub-canal* o un nuevo *canal*. El *nombre* del *canal* debe aparecer al lado del símbolo. Se puede asociar una *lista de señal* opcional (§ 2.2.1 de la Recomendación Z.101).

FIGURA 15/Z.102

Símbolos utilizados en el diagrama de subestructura de canales

Además de estos símbolos pueden aparecer en el diagrama *definiciones de señal*, utilizando la sintaxis LED/PR.

Las reglas de conexión de diagrama son las mismas que para el *diagrama de interacción* (véase la Recomendación Z.101), con la sola excepción de que en el título del diagrama debe quedar claro que se trata de un *diagrama de subestructura de canales.*

Un sencillo ejemplo de *diagrama de subestructura de canales* viene dado en la figura 16/Z.102.

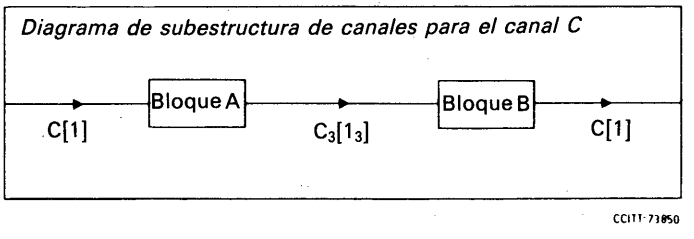


FIGURA 16/Z.102

Ejemplo de diagrama de subestructura de canales

3.4.2 *Relación con la sintaxis abstracta del LED*

Un *diagrama de subestructura de canales* representa una *definición de subestructura de canales.* Las definiciones contenidas en la *definición de subestructura de canales* están representadas por los símbolos de *bloque* y de *canal* junto con las definiciones de la sintaxis *LED/PR.*

El *canal* que conduce desde el marco hasta el diagrama representa el *canal de entrada* y el *canal* que termina en el marco representa el *canal de salida.*

3.4.3 *Convenciones gráficas*

Las convenciones gráficas descritas para el *diagrama de interacción* en la Recomendación Z.101 se aplican al *diagrama de subestructura de canales.*

4 LED/PR

La siguiente sintaxis LED/PR constituye una adición a la sintaxis definida en la Recomendación Z.101. Las adiciones cubren la representación de la estructura y *partición* de un *sistema.*

Adviértase que en los ejemplos, las palabras clave *LED/PR* aparecen en letras mayúsculas.

4.1 *Definición de bloque*

La *definición de bloque* del *LED/PR* se amplía para incluir opcionalmente la «*Definición de subestructura de bloque interno*» no terminal.

4.1.1 *Sintaxis*

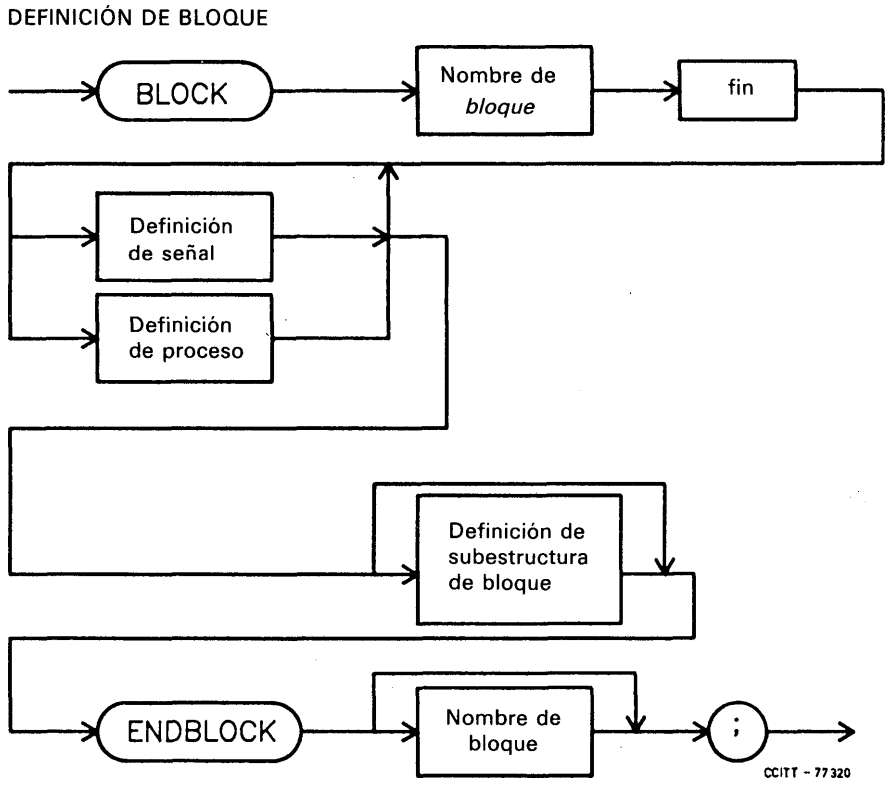


FIGURA 17/Z.102

Diagrama de sintaxis para un bloque

4.2 *Definición de subestructura de bloque*

La *definición de subestructura de bloque* representa la *partición* de un *bloque* en *sub-bloques, sub-canales* y nuevos *canales.*

4.2.1 *Sintaxis*

DEFINICIÓN DE SUBESTRUCTURA DE BLOQUE

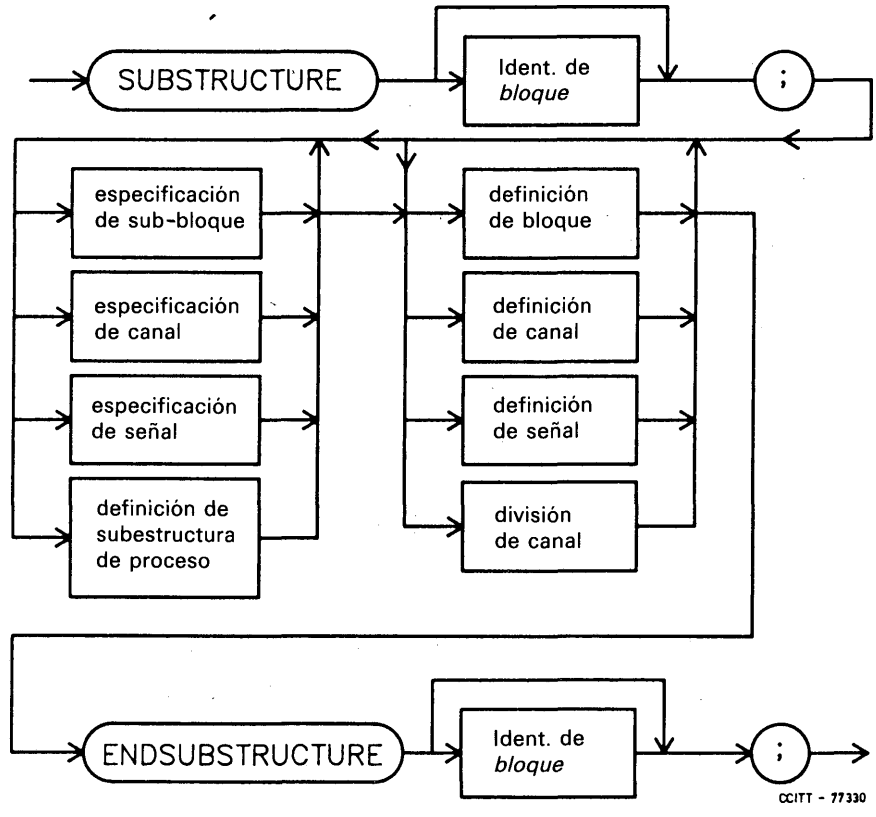
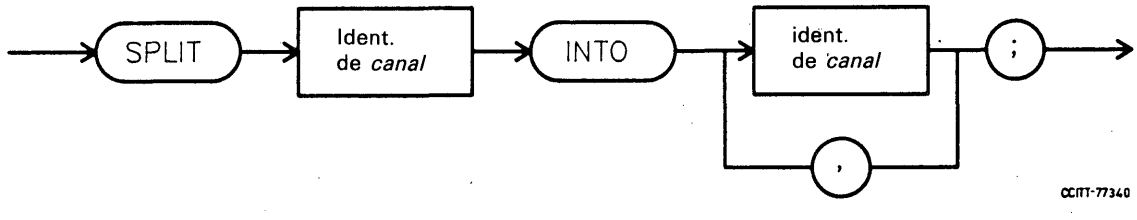


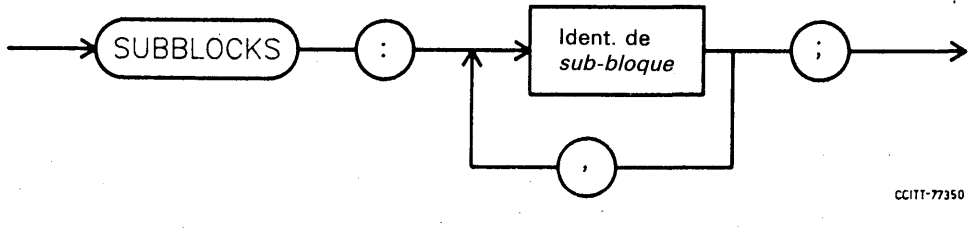
FIGURA 18/Z.102

Diagrama de sintaxis para una definición de subestructura de bloque

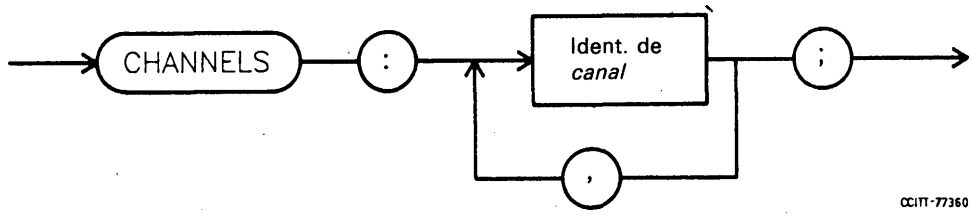
DIVISIÓN DE CANAL



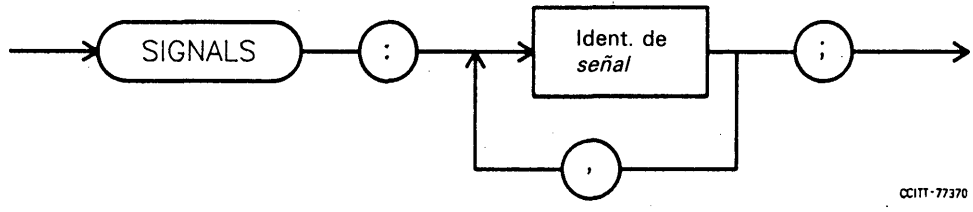
ESPECIFICACIÓN DE SUB-BLOQUE



ESPECIFICACIÓN DE CANALES



ESPECIFICACIÓN DE SEÑALES



Ejemplo:

```
BLOCK b:

    ...
    SUBSTRUCTURE b;
        SUBBLOCKS:  b1,b2,b3;
        CHANNELS:   c1,c2,d1,d2,e;
        SIGNALS:    s1,s2,s3;

    SPLIT c INTO c1,c2;
    SPLIT d INTO d1,d2;

    ...
        /* Definiciones de canales */
        /* Definiciones de bloques */
        /* Definiciones de señales */

    ENDSUBSTRUCTURE;

ENDBLOCK b;
```

4.2.2 *Relación con la sintaxis abstracta del LED*

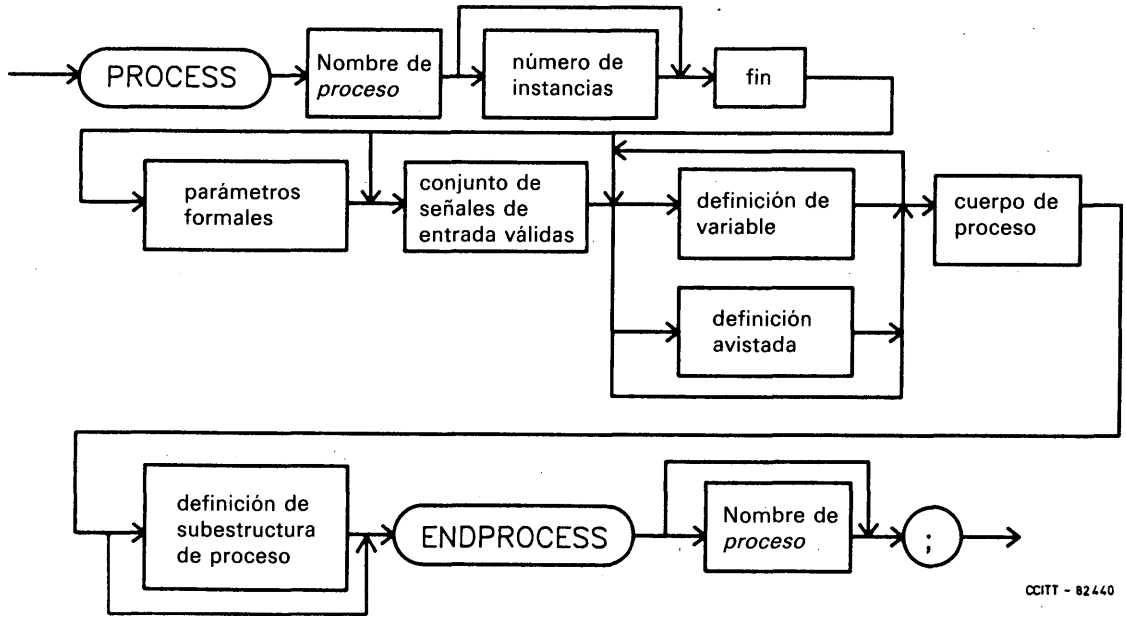
La sintaxis representa la *definición de subestructura de bloques* y la *definición de bloque de parte interna* en la *sintaxis abstracta.* Los *nombres* de *bloques, señales* y *canales* corresponden a las *definiciones de bloque,* las *definiciones de señal* y las *definiciones de canal* contenidas.

4.3 *Subestructura de proceso*

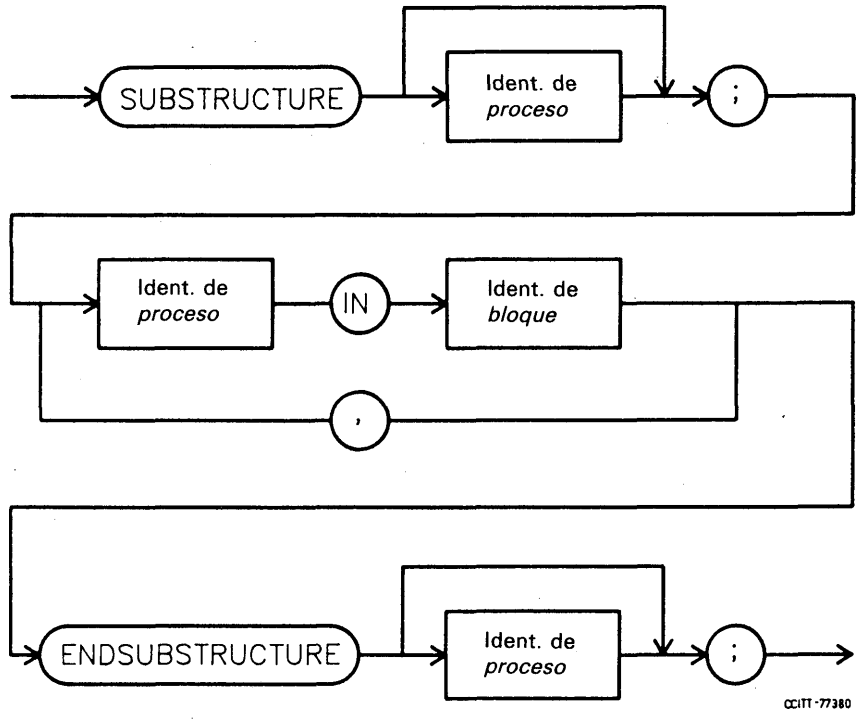
Esta sintaxis representa la *partición* de un proceso en *sub-procesos* y la atribución de éstos a *sub-bloques.* La *partición* se puede mostrar tanto en la *definición de proceso* como en la *definición de subestructura de bloque.*

4.3.1 *Sintaxis*

DEFINICIÓN DE PROCESO



DEFINICIÓN DE SUBESTRUCTURA DE PROCESO



El siguiente ejemplo muestra la *subestructura de proceso* como parte de una *definición de proceso:*

Ejemplo:

```
...
    SUBSTRUCTURE p1;
        p2 IN b2;
        p3 IN b1 ;
    ENDSUBSTRUCTURE;
...
ENDPROCESS p1;
```

El ejemplo que figura a continuación muestra la *subestructura de proceso* como parte de una *definición de subestructura de bloque:*

Ejemplo:

```
...
    SUBSTRUCTURE b;
        SUBSTRUCTURE P;
            P1 IN b2;
            P2 IN b1;
        ENDSUBSTRUCTURE;
    ...
    ENDSUBSTRUCTURE;
ENDBLOCK b;
```

4.3.2 *Relación con la sintaxis abstracta del LED*

La construcción sintáctica representa la *definición de subestructura de proceso* en la sintaxis abstracta. El *nombre de proceso* asociado es el *nombre* de la cláusula de iniciación y el conjunto contenido de *nombres de proceso*, asociado cada uno a un *nombre de bloque*, son los *nombres* separados por la palabra clave «IN».

4.4 *Subestructura de canales*

Esta sintaxis representa la *partición* de un *canal* en una serie de *canales* y *bloques.*

4.4.1 *Sintaxis*

La sintaxis de un *canal* se amplía para contener «*subestructuras de canales*» no terminales como parte opcional.

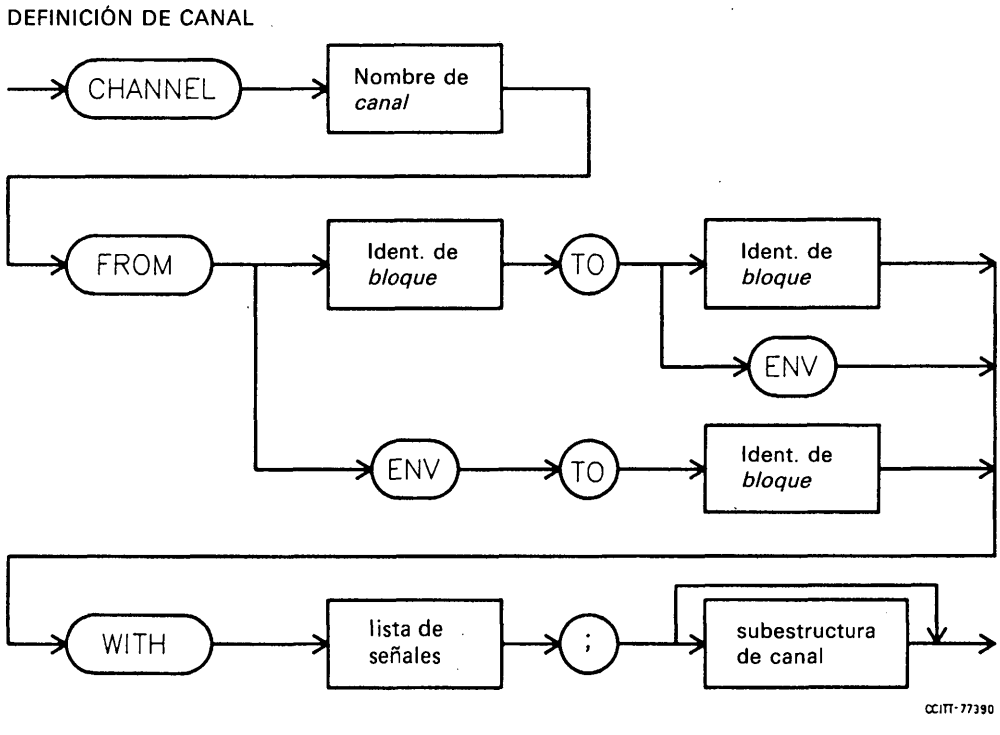


FIGURA 19/Z.102 (1 de 4)

Diagrama de sintaxis para una subestructura de canal

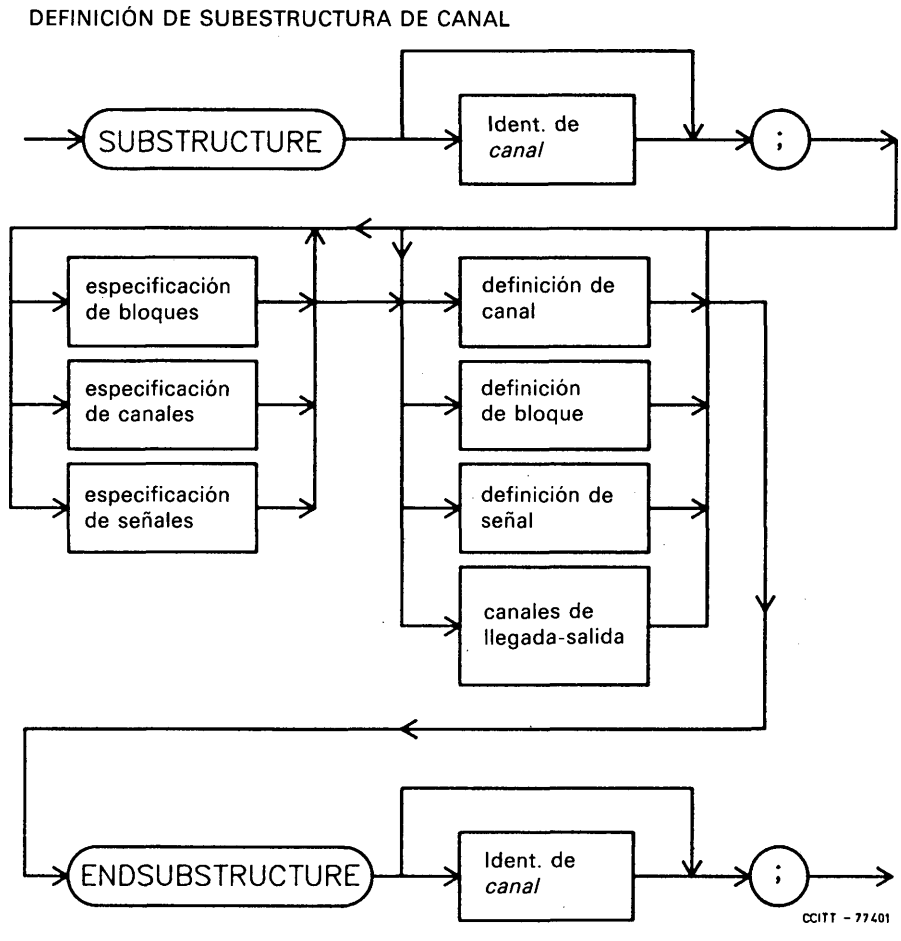


FIGURA 19/Z.102 (2 de 4)

Diagrama de sintaxis para una subestructura de canal

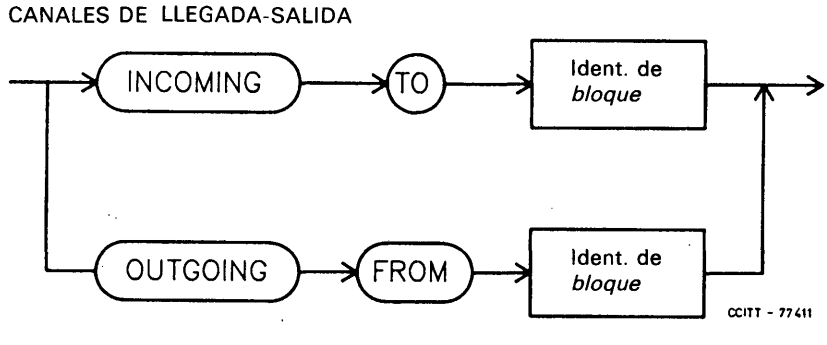


FIGURA 19/Z.102 (3 de 4)

Diagrama de sintaxis para una subestructura de canal

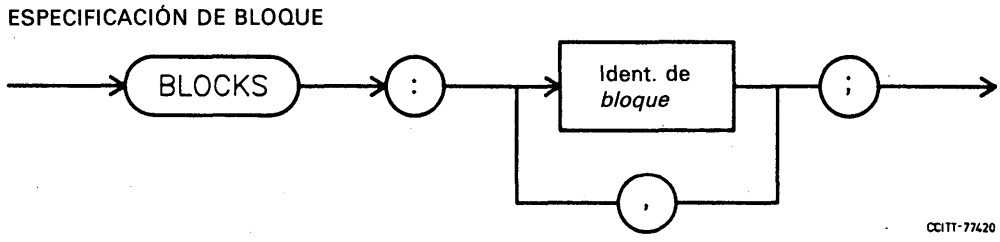


FIGURA 19/Z.102 (4 de 4)

Diagrama de sintaxis para una subestructura de canal

La *subestructura de canales* es un conjunto de referencias a los componentes y definiciones adicionales de la subestructura.

Ejemplo:

```
CHANNEL c FROM b TO d;
    SUBSTRUCTURE
        INCOMING TO e;
        OUTCOING FROM f;
        BLOCKS: e,f;
        CHANNEL c1 FROM e TO f WITH s1,s2,s3;
        BLOCK e;
                    ...
        ENDBLOCK e;
        BLOCK f;
                    ...
        ENDBLOCK f;
    ENDSUBSTRUCTURE;
```

4.4.2 *Relación con la sintaxis abstracta LED*

La sintaxis representa la *definición de subestructura de canales* en la *sintaxis abstracta.* Los *identificadores de bloques, canales, señales* y *datos* que aparecen en la sintaxis son referencias a las definiciones contenidas.

Recomendación Z.103

AMPLIACIONES FUNCIONALES DEL LED

1 Introducción

Esta Recomendación define un número de conceptos y notaciones abreviadas adicionales en el Lenguaje de Especificación y Descripción (LED). Para estas adiciones se ha tomado como base el LED fundamental, definido en la Recomendación Z.101. Estas adiciones tienen por objeto proporcionar a los usuarios del LED conceptos y notaciones abreviadas convenientes.

En esta Recomendación se definen los siguientes conceptos y notaciones abreviadas:

Procedimientos:

Proporcionan una manera de representar una porción de un *gráfico de proceso* por un elemento, al cual se puede hacer referencia varias veces. El *comportamiento* detallado del *procedimiento* se define en otro lugar, dentro o fuera del *gráfico de proceso.* Se pueden utilizar *procedimientos* para apoyar el uso de métodos de diseño estructurado con LED, al permitir la descomposición de un *gráfico de proceso* en unas secciones jerarquizadas. El concepto es similar al concepto de procedimiento empleado en lenguajes de programación.

Importación y exportación de valores:

Esta es una notación abreviada para el interfuncionamiento de *señales* entre *instancias de proceso* cuando éstas deban compartir un valor de una variable poseída por uno de los *procesos.*

Condición habilitante:

Esta es una notación abreviada que se emplea para evitar la «explosión de estado» cuando la recepción o conservación de un conjunto de *señales* es condicional.

Señal continua:

Esta es una notación abreviada que se emplea para representar el interfuncionamiento de *señales* cuando se observa una condición continua, externa al *proceso.*

Macro:

Es un método sintáctico que emplea el usuario para definir una notación abreviada. Un *macro* es la definición de un elemento de sintaxis compuesto en términos de otros elementos de sintaxis ya definidos. Un *macro* no tiene una semántica propia.

Opción:

Facilidad sintáctica para representar varios comportamientos alternativos en un diagrama, o texto lineal. Antes de interpretar el diagrama o texto lineal hay que decidir qué alternativa ofrecida por una *opción* debe elegirse.

Esta Recomendación define también una sintaxis gráfica ampliada del LED en la cual se utilizan elementos pictográficos en símbolos de *estado.*

2 Procedimientos

Procedimiento es un medio en virtud del cual se da un nombre a un conjunto de ítems, representándose este conjunto por una sola referencia. Las reglas relativas a los *procedimientos* imponen una disciplina en cuanto a la forma en que ha de elegirse el conjunto de ítems, y limita el alcance del nombre de *ítems de datos* y *señales.*

Los procedimientos tienen por objeto:

a) permitir la estructuración de un *gráfico de proceso* en varios niveles de detalle;

b) mantener la compacidad de las especificaciones al permitir que un conjunto completo de ítems pueda ser considerado aisladamente y representado por un solo ítem;

c) hacer posible que los conjunto de ítems que se utilizan corrientemente puedan ser predefinidos y utilizados repetidamente.

Los *procedimientos* se definen por medio de *definiciones de procedimiento*. Un *procedimiento* es invocado por medio de *una llamada de procedimiento* que hace referencia a la *definición de procedimiento.* Una llamada de *procedimiento* tiene asociado parámetros: éstos se utilizan para pasar *valores*, y también para controlar el alcance de *datos* y *señales* para la ejecución del *procedimiento.* El mecanismo de pase de parámetros controla qué *señales* e *ítems de datos* son afectados por la interpretación de un *procedimiento.*

2.1 *Modelo de lenguaje común*

Las siguientes definiciones deben considerarse estrictamente como adiciones a las definiciones contenidas en la Recomendación Z.101.

2.1.1 *Introducción al modelo común*

Un *procedimiento* es una sección de un *gráfico de proceso* que puede considerarse aisladamente. Tiene un solo punto de entrada y un solo punto de salida. Una *llamada de procedimiento* puede aparecer en todo punto en que pueda aparecer una *tarea* en un *gráfico de proceso* o un *gráfico de procedimiento.* Un *procedimiento* puede contener *estados.* Todos los *ítems de datos* utilizados en un *procedimiento* tienen que estar definidos dentro de la *definición de procedimiento.* Si un *ítem de datos* está definido como un *parámetro formal,* utilizará *valores* de *ítems de datos* en el entorno llamante y/o dará *valores* a *ítems de datos* en el entorno llamante; en todo otro caso, el ítem es local al *procedimiento* y tiene que estar definido en la *definición de procedimiento.*

Un *procedimiento* sólo es interpretado cuando es llamado por una *instancia de proceso*; la interpretación la hace la misma *instancia de proceso* que lo llamó. Cuando una *llamada de procedimiento* es interpretada, el *gráfico de procedimiento* tiene que ser interpretado antes de continuar la interpretación de la *transición* en la cual aparece la *llamada.* Esto significa que el *puerto de entrada de la instancia de proceso* llamante y el conjunto de *señales de entrada válidas* se utilizan mientras se está interpretando el *procedimiento.* Todas las *señales* referenciadas dentro de un *procedimiento* tienen que ser *nombradas* como *parámetros formales* del *procedimiento.* Todas las *señales* en el conjunto de nombres de *señales de entrada válidas* del entorno llamante que no están referenciadas dentro del *procedimiento* deben ser también *nombradas* como *parámetros* del *procedimiento,* es decir, el *conjunto de conservación adicional.* Esto permite al *procedimiento conservar automáticamente* cualesquiera otras

señales que pudieran llegar al *puerto de entrada.* Si no se hiciese esto, la introducción en una transición, de una *llamada de procedimiento* a un *procedimiento* que contiene un *estado* tendría efectos marginales ocultos de pérdida de *señales* (por «transiciones nulas» implícitas) que llegaron cuando una *instancia de proceso* estaba interpretando un *procedimiento.*

Una *definición de procedimiento* se da por un *gráfico de procedimiento*, y puede aparecer en cualquier punto en que pueda aparecer una *definición de tipo de datos.*

Un *gráfico de procedimiento* sigue las mismas reglas que un *gráfico de proceso,* con las siguientes excepciones:

- un *nodo de arranque* se sustituye por un *nodo de arranque de procedimiento*;
- un *nodo de parada* se sustituye por un *nodo de retorno*;
- sólo los *ítems de datos*, *parámetros formales* y *nombres de señales* que están declarados dentro del *procedimiento* son visibles por la *instancia de proceso* mientras se está interpretando el *procedimiento*; es decir, un *procedimiento* no puede hacer referencia a ningún *ítem de datos* o *señal* que esté fuera del *procedimiento* a menos que esté declarado como un *parámetro formal* (las *señales* tienen necesariamente que pertenecer al entorno llamante del *procedimiento,* y todas las *señales* que sean objeto de operaciones de *entrada, conservación* o *salida* en el *procedimiento* deberán estar *declaradas*);
- el mismo *parámetro actual de nombre de señal* puede no estar mapeado con respecto a más de un *parámetro formal de nombre de señal*;
- en la *sintaxis abstracta*, el conjunto de *señales* que ha de *conservarse* se pasa como *parámetros* de la *llamada de procedimiento.* Sin embargo, en la sintaxis concreta, todos los *nodos de estado* contenidos en un *procedimiento* tienen un *conjunto de señales de conservación* implícito que contiene todas las *señales de entrada* que no están declaradas como *parámetros formales en el procedimiento.*

2.1.2 *Sintaxis abstracta*

A la *sintaxis abstracta* definida en la Recomendación Z.101 deberán hacerse las siguientes adiciones:

Definición de sistema

Una *definición de sistema* puede también contener una o más *definiciones de procedimiento.*

Definición de bloque

Una *definición de bloque* puede también contener una o más *definiciones de procedimiento.*

Definición de proceso

Una *definición de proceso* puede también contener una o más *definiciones de procedimiento.*

Gráfico de proceso

Un *gráfico de proceso* puede también contener un *nodo de llamada de procedimiento.*

Una *cadena de transición* puede también contener un *nodo de llamada de procedimiento* seguido de una *cadena de transición.*

Un *nodo de llamada de procedimiento* contiene un *identificador de procedimiento* y una *lista de parámetros actuales.*

Definición de procedimiento

La *definición de procedimiento* está asociada con un *nombre de procedimiento* y contiene una *lista de parámetros formales,* un *conjunto de conservación adicional* y un *gráfico de procedimiento,* y puede contener *definiciones de procedimiento* y *definiciones de datos.*

La *lista de parámetros formales* puede estar vacía. A cada *parámetro formal* en la lista hay que atribuirle uno de los atributos *DENTRO (IN), DENTRO/FUERA (IN/OUT)* o *SEÑAL (SIGNAL).*

Los *nombres* de los *parámetros formales* atribuidos con *DENTRO* o *DENTRO/FUERA* pueden no utilizarse en *definiciones de variables* contenidas en la *definición de procedimiento*; y el conjunto de *nombres de parámetros formales* atribuidos con *SEÑAL* deberá contener todos los *nombres de señal* referenciados en el *gráfico de procedimiento.*

Gráfico de procedimiento

Un *gráfico de procedimiento* es un *gráfico* cuyos *nodos* están conectados por *arcos* dirigidos. Un *arco* que entra en un *nodo* se denomina *arco* entrante y un *arco* que sale del *nodo* se denomina *arco* saliente. Un *nodo* para el cual un determinado *arco* es un *arco* entrante sigue al *nodo* que tiene ese mismo *arco* como *arco* saliente.

En un *gráfico de procedimiento* están permitidos los siguientes tipos de *nodos*:

Nodo de estado
Nodo de entrada
Nodo de tarea
Nodo de salida

Nodo de decisión
Nodo de llamada de procedimiento
Nodo de arranque de procedimiento
Nodo de retorno
Nodo de petición de creación

Cada *nodo* del *gráfico* tiene un *nombre* y un *tipo.* Los *nodos de estado* dentro de un *gráfico de procedimiento* tienen *nombres* diferentes.

Las reglas siguientes definen la conectividad de un *gráfico de procedimiento*:

- Cada *gráfico de procedimiento* contiene un *nodo de arranque de procedimiento*, y sólo uno. El *nodo de arranque de procedimiento* va seguido de una *cadena de transición.* El *nodo de arranque* no sigue a ningún otro *nodo.*
- Una *cadena de transición* puede tener una de las composiciones siguientes:
  a) nula seguida de un *nodo de estado* o un *nodo de retorno*;
  b) una cadena de acción seguida de una *cadena de transición*;
  c) un *nodo de decisión.*
- Una cadena de acción puede tener una de las composiciones siguientes:
  a) un *nodo de tarea*;
  b) un *nodo de salida*;
  c) un *nodo de petición de crear.*
- Un *nodo de decisión* va seguido de dos o más *arcos de decisión.*
- Un *arco de decisión* es un *arco* nombrado seguido de una *cadena de transición.*
- Un *nodo de estado* va seguido de uno o más *nodos de entrada.*
- Un *nodo de entrada* va seguido de una *cadena de transición.*
- El *nodo de retorno* no va seguido de ningún *nodo.*
- Cada gráfico tiene por lo menos un *nodo de retorno.*
- Cada *nodo* se puede alcanzar desde el *nodo de arranque de procedimiento.*

Un *nodo de llamada* contiene un *identificador de procedimiento,* una *lista de conservación adicional* y una *lista de parámetros actuales.* En esta lista tiene que haber un *parámetro actual* para cada uno de los *parámetros formales* en la *definición de procedimiento* referenciada. Cada *parámetro actual* tiene que concordar con el *tipo* del correspondiente *parámetro formal.* Un *parámetro actual de nombre de señal* cualquiera puede no aparecer más de una vez en la *lista de parámetros actuales.*

Cada *parámetro actual* para el cual el correspondiente *parámetro formal* tiene el atributo *DENTRO/FUERA* tiene que ser un *nombre de variable.*

El *conjunto de conservación adicional* es un conjunto posiblemente vacío de *identificadores de señal.*

2.1.3 *Interpretación*

Las reglas de interpretación de la Recomendación Z.101 se completan con las adiciones siguientes:

Proceso

Un *nodo de llamada* se interpreta como la interpretación del *nodo de arranque de procedimiento* de la *definición de procedimiento* referenciada por el *nombre* del *nodo*; después se interpreta el *nodo* que sigue al *nodo de llamada.*

Se da al *conjunto de conservación adicional* el *valor* de todos los *identificadores de señal* contenidos en el *conjunto de señales de entrada válidas* del *proceso* llamante, que no aparezcan como *parámetros actuales* del *nodo.*

Procedimiento

Cuando un *proceso* interpreta un *nodo de llamada* que hace referencia a la *definición de procedimiento* que contiene el *gráfico de procedimiento,* se interpreta el *gráfico de procedimiento.* Los *nodos* del *gráfico de procedimiento* se interpretan de la misma manera que los *nodos* equivalentes del *gráfico de proceso* con las siguientes excepciones y adiciones:

- Cada *parámetro formal* atribuido con *DENTRO* denota una *variable* con tipo. Esta *variable* es local al *procedimiento* y se crea cuando el *nodo de arranque de procedimiento* es interpretado, y deja de existir cuando se interpreta el *nodo de retorno.*
- Cada *parámetro formal* atribuido con *DENTRO/FUERA* denota un *nombre sinónimo* para la *variable* que se da como *parámetro actual.* Este *nombre sinónimo* se utiliza en toda la interpretación del *gráfico de procedimiento* cuando se hace referencia al *valor* de la *variable* o cuando se asigna un nuevo *valor* a la *variable.*

Nodo de arranque de procedimiento

Para la interpretación de un *nodo de arranque de procedimiento*:

- Se crea una *variable* local para cada *parámetro formal* atribuido con *DENTRO*, la cual tiene el *nombre* y *tipo de datos* del *parámetro formal*. Se asigna a la *variable* el *valor* del *parámetro actual*, que puede estar indefinido.
- Cada *nombre de variable* dado en los *parámetros formales* atribuidos con *DENTRO/FUERA* se utiliza como un *nombre sinónimo* para la *variable* dada como *parámetro actual*. La *variable* representada por el *nombre sinónimo* se evalúa una vez solamente en el *nodo de arranque de procedimiento*, y no cada vez que se utilice el *parámetro formal* en el *procedimiento*.
- Cada *nombre de señal* dado en los *parámetros formales* atribuidos con *SEÑAL* se utiliza como un *nombre sinónimo* para el *identificador de señal* dado como *parámetro actual*.
- Se interpreta el *nodo* que sigue al *nodo de arranque de procedimiento*.

Nodo de estado

El *nodo de estado* se interpreta de la misma forma que un *gráfico de proceso*, con la excepción de que el conjunto de *señales de conservación* presentado al *puerto de entrada* es la unión del *conjunto de conservación adicional* y el *conjunto de señales de conservación* del *proceso* o *procedimiento* que llamó al *procedimiento* en curso.

2.2 *LED/GR*

La adición del concepto de *procedimiento* en el LED provoca la adición de un símbolo nuevo en el *diagrama de proceso*: el símbolo de *llamada*.

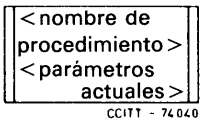
La *definición de procedimiento* se representa en la sintaxis gráfica mediante un *diagrama de procedimiento*. Este diagrama es análogo al *diagrama de proceso*, salvo en lo que respecta al *símbolo de arranque de procedimiento* y al *símbolo de retorno*.

Seguidamente sólo se indican las adiciones que exige la introducción del *procedimiento* en la sintaxis gráfica definida en la Recomendación Z.101.

2.2.1 *Diagrama de proceso*

2.2.1.1 *Símbolos*

Para representar la *llamada de procedimiento* se utiliza el siguiente símbolo:



El símbolo puede aparecer en un *diagrama de proceso* siempre que se admita un símbolo de *tarea*.

El *nombre de procedimiento* y los *parámetros actuales* se indican utilizando la sintaxis LED/PR.

FIGURA 1/Z.103

Símbolo de llamada de procedimiento

2.2.1.2 *Relación con la sintaxis abstracta LED*

El *símbolo de llamada de procedimiento* representa el *nodo de llamada de procedimiento* en un *gráfico de proceso*.

Los *parámetros actuales* que se dan con los símbolos representan los *parámetros actuales* asociados con el *nodo de llamada de procedimiento*. Toda *señal de entrada válida* que no se dé como *parámetro actual* forma parte del *conjunto de conservación adicional* correspondiente al *nodo de llamada de procedimiento* e indica que esa *señal* se ha de *conservar* mediante la ejecución del *procedimiento*.

Nodo de retorno

Para la interpretación del *nodo de retorno*:

- todas las *variables* creadas por la interpretación del *nodo de arranque de procedimiento* dejarán de existir;
- todos los *nombres sinónimos* establecidos por la interpretación del *nodo de arranque de procedimiento* dejarán de existir;
- la interpretación del *nodo de retorno* completa la interpretación del *nodo de arranque de procedimiento.*

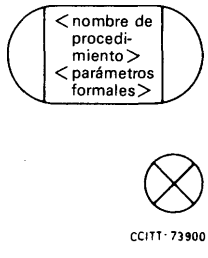
2.2.2 *Diagrama de procedimiento*

Un *diagrama de procedimiento* es similar a un *diagrama de proceso*, del cual sólo se diferencia en que el *símbolo arranque* se sustituye por el *símbolo arranque de procedimiento*, y el *símbolo parada* se sustituye por el *símbolo retorno.*

A continuación se define la sintaxis adicional solamente.

2.2.2.1 *Símbolos*

En la figura 2/Z.103 siguiente se definen los dos símbolos adicionales:



Símbolo de arranque de procedimiento
El *nombre de procedimiento* y los *parámetros formales* se dan con la sintaxis *LED/PR*

Símbolo de retorno

FIGURA 2/Z.103

Símbolos adicionales para diagrama de procedimiento

2.2.2.2 *Relación con la sintaxis abstracta LED*

El *diagrama de procedimiento* representa la *definición de procedimiento* en la *sintaxis abstracta* LED.

Los símbolos comunes con el *diagrama de procedimiento* guardan la misma relación con la *sintaxis abstracta* cuando aparecen en un *diagrama de proceso.* Los símbolos adicionales de *arranque de procedimiento* y de *retorno* representan respectivamente el *nodo de arranque de procedimiento* y el *nodo de retorno.*

Los *parámetros* asociados con el *símbolo arranque de procedimiento* representan los *parámetros formales* asociados con el *nodo de arranque de procedimiento.*

2.2.2.3 *Convenios sobre la presentación gráfica*

Generalmente, se siguen los mismos convenios para la presentación gráfica aplicados al *diagrama de proceso.* Los convenios para el *símbolo de arranque de procedimiento* y para el *símbolo de retorno* son los mismos que para el *símbolo de arranque de proceso* y el *símbolo de parada* respectivamente.

Un *símbolo de retorno*, en cualquier punto en que aparezca, representa el *nodo de retorno* (aparición múltiple de un símbolo de *retorno*).

2.3 *LED/PR*

A continuación se da la sintaxis *LED/PR* para una *definición de procedimiento.* La *definición de procedimiento* puede aparecer en todo punto en que pueda aparecer una *definición de tipo de datos* en la sintaxis. La sintaxis es similar a la sintaxis de la *definición de proceso.*

A continuación se dan, solamente, las adiciones a la sintaxis ya definida en la Recomendación Z.101.

2.3.1 *Sistema*

2.3.1.1 *Sintaxis*

La sintaxis de *definiciones de procedimiento* se añade a la sintaxis de *sistema.*

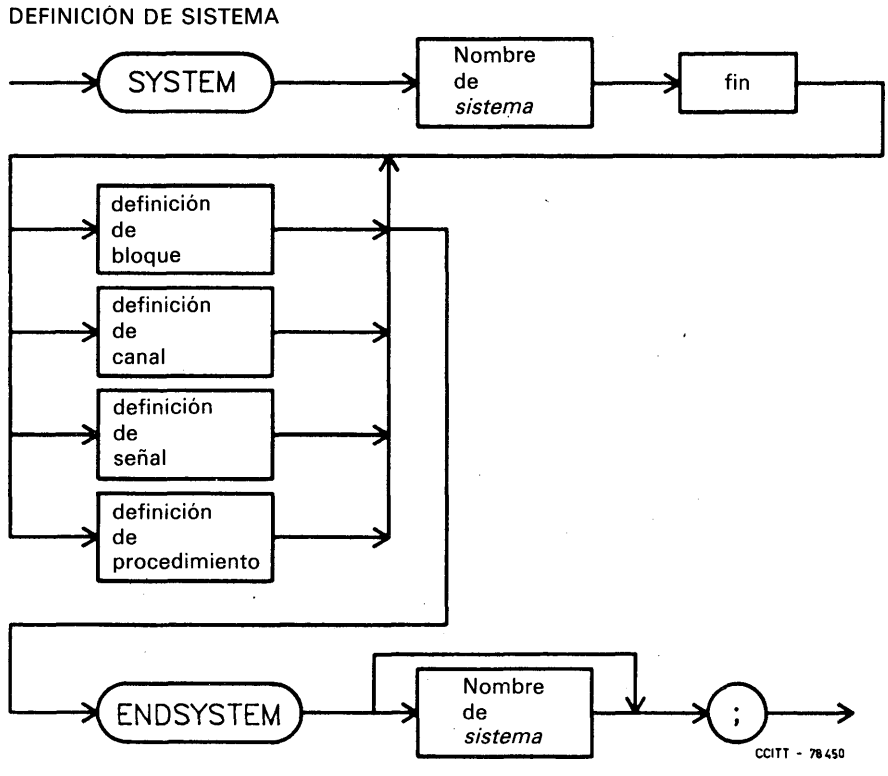


FIGURA 3/Z.103

Diagrama de sintaxis para un sistema

2.3.2 *Bloque*

2.3.2.1 *Sintaxis*

La sintaxis de *definiciones de procedimiento* se añade a la sintaxis de *bloque*:

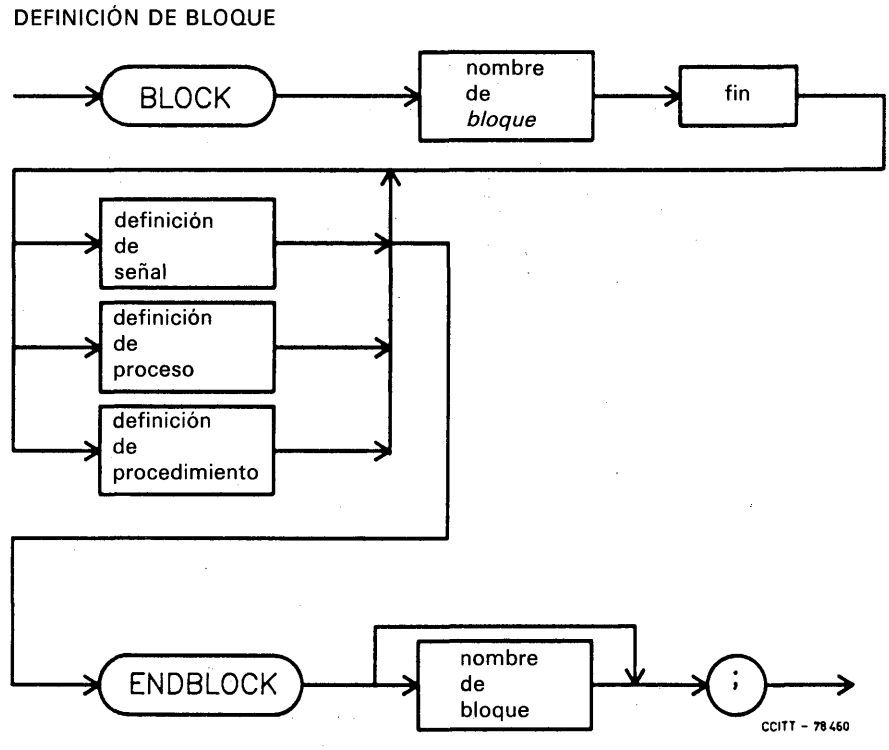


FIGURA 4/Z.103

Diagrama de sintaxis para un bloque

2.3.3 *Proceso*

2.3.3.1 *Sintaxis*

La sintaxis de *definición de procedimiento* se añade a la sintaxis de *proceso*, y la sintaxis de *llamada de procedimiento* se añade a la sintaxis de *cadena de transición*:

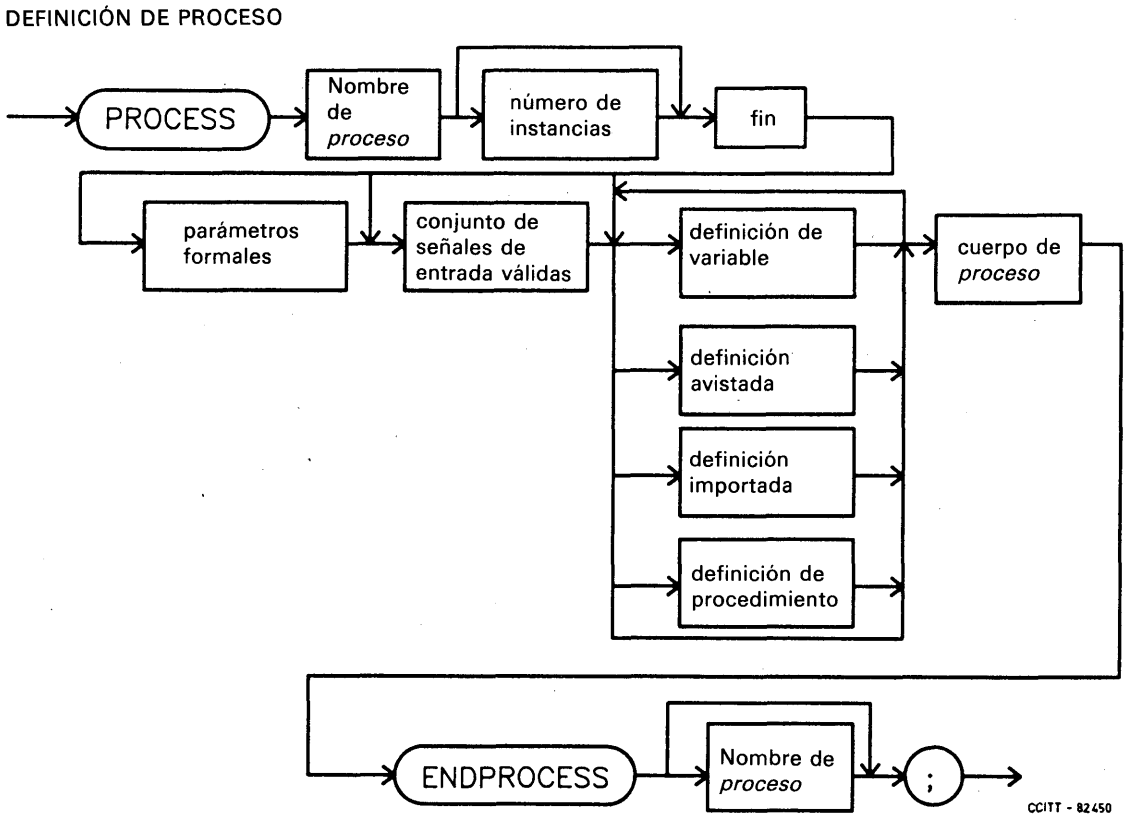


FIGURA 5/Z.103

Diagrama de sintaxis para un proceso

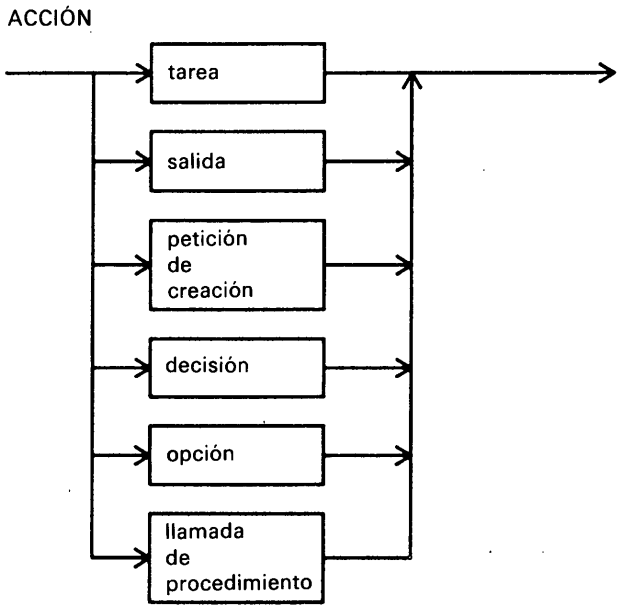


FIGURA 6/Z.103 (1 de 2)

Diagrama de sintaxis para una acción

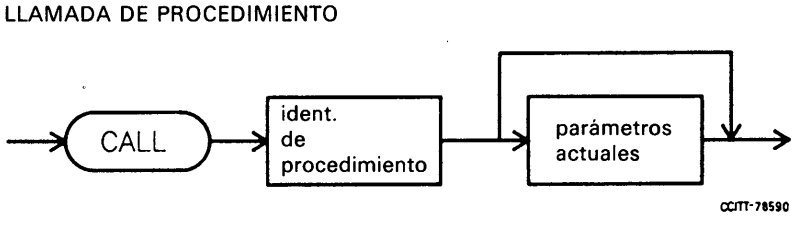


FIGURA 6/Z.103 (2 de 2)

Diagrama de sintaxis para una llamada de procedimiento

2.3.3.2 *Relación con la sintaxis abstracta LED*

La *llamada de procedimiento* en una *transición* representa el *nodo de llamada* en la *sintaxis abstracta*, y los *parámetros actuales* representan los *parámetros actuales* asociados con el *nodo de llamada.*

Cada *parámetro actual* dado tiene que ajustarse al *tipo de datos* del *parámetro formal* correspondiente en la *definición de procedimiento* referenciada.

2.3.4 *Procedimiento*

2.3.4.1 *Sintaxis*

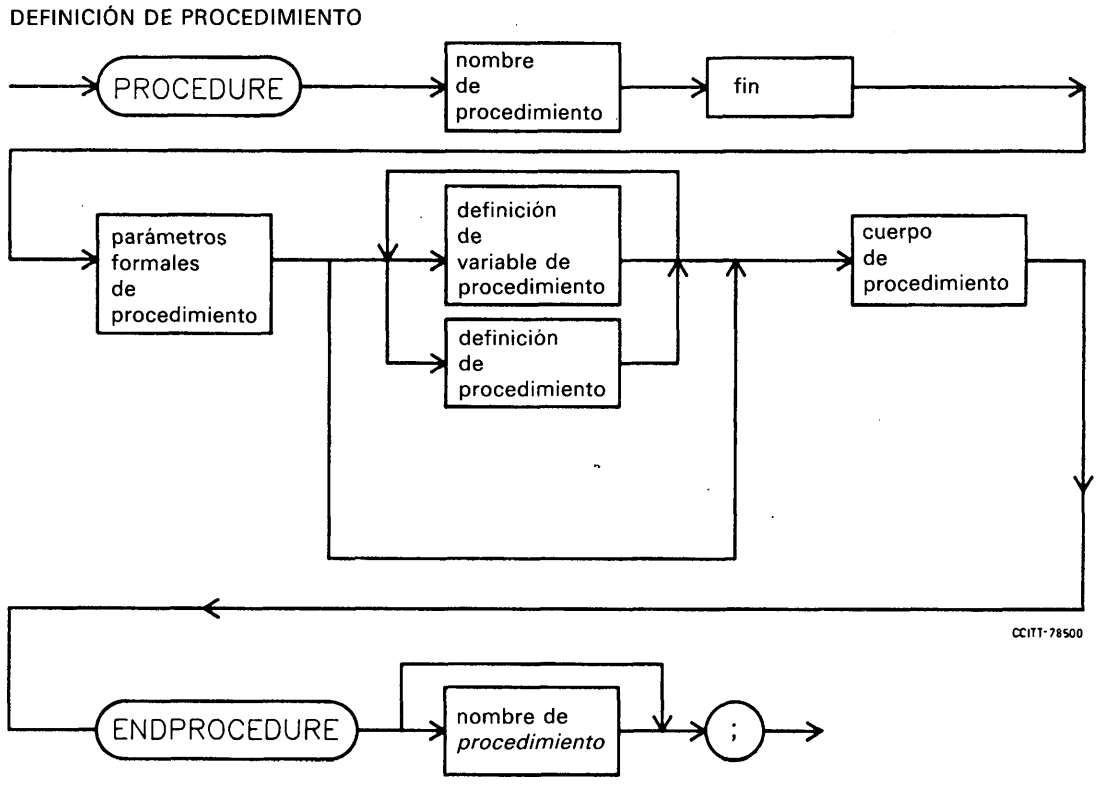


FIGURA 7/Z.103 (1 de 5)

Diagramas de sintaxis para un procedimiento

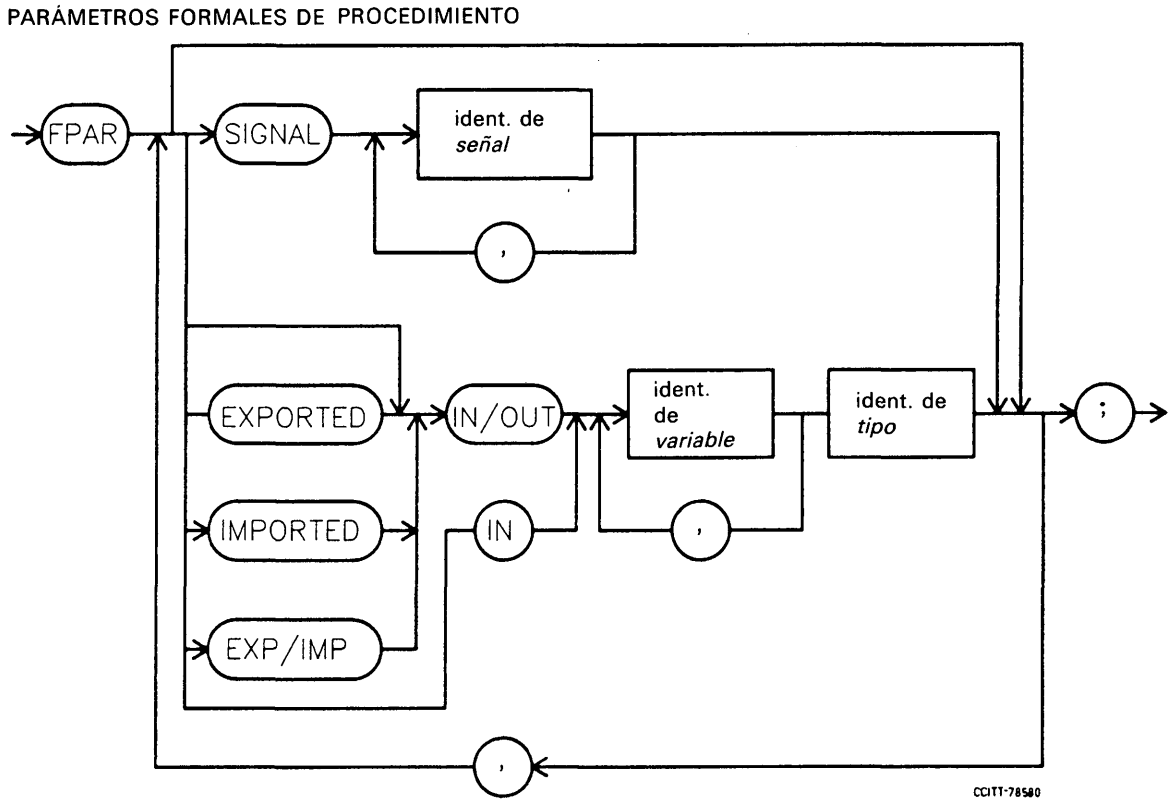


FIGURA 7/Z.103 (2 de 5)

Diagramas de sintaxis para un procedimiento

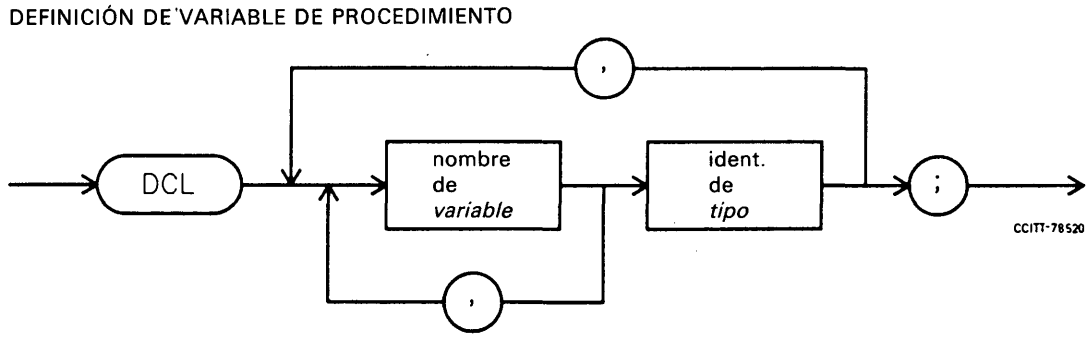


FIGURA 7/Z.103 (3 de 5)

Diagramas de sintaxis para un procedimiento

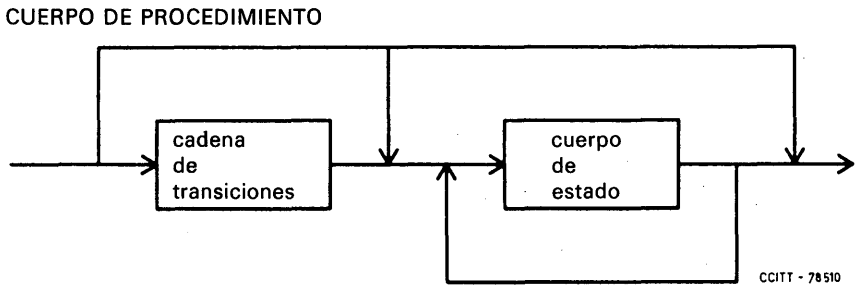


FIGURA 7/Z.103 (4 de 5)

Diagramas de sintaxis para un procedimiento

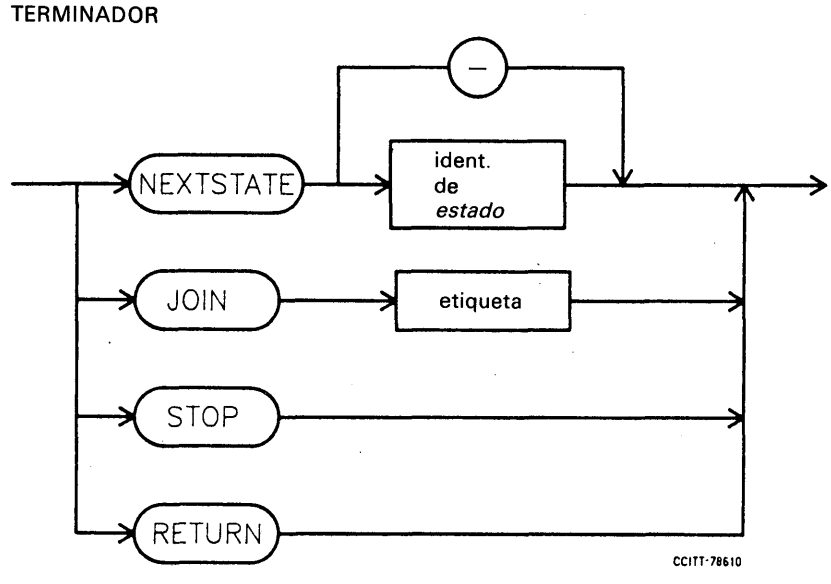


FIGURA 7/Z.103 (5 de 5)

Diagramas de sintaxis para un procedimiento

2.3.4.2 *Relación con la sintaxis abstracta LED*

El diagrama de la sintaxis de *procedimiento* representa la *definición de procedimiento* de la sintaxis abstracta. Los *parámetros formales* del diagrama de sintaxis representan los *parámetros formales* asociados con el *nodo de arranque de procedimiento.*

Las construcciones sintácticas permitidas en un *procedimiento* y que son comunes a las permitidas en un *proceso* guardan la misma relación con la sintaxis abstracta que cuando aparecen en un *proceso.* El *nodo de arranque de procedimiento* es implicado y va seguido por la primera construcción sintáctica del *procedimiento,* y las sentencias de *retorno* representan el *nodo de retorno.*

3 Operaciones compuestas para comunicación entre procesos LED

Una *operación compuesta* es una notación abreviada normalizada para reunión de conceptos LED. Las *operaciones compuestas* se definen en términos de conceptos LED ya presentes y no se adicionan a la semántica del lenguaje. Tampoco cambian la interpretación de la *sintaxis abstracta.* Se introducen para la conveniencia de los usuarios del LED.

Las *operaciones compuestas* definidas aquí proporcionan algunos métodos alternativos de comunicación entre *instancias de proceso* LED. Si bien parecen al usuario esencialmente diferentes del mecanismo normal de intercambio de *señales* del LED, estas operaciones se basan en realidad en la semántica LED normal.

Las *operaciones compuestas* proporcionan:

a) la compartición de *valores* de *ítems de datos* entre *procesos*, incluso si están atribuidos en *bloques* diferentes;

b) la capacidad para habilitar o inhabilitar temporalmente la recepción de determinadas *señales* sin necesidad de que esto se muestre exactamente con *estados* separados;

c) *señales continuas*, para las cuales se causan *transiciones* por un cambio de un *valor* de la *señal.*

Cada una de las *operaciones compuestas* se ve atribuida una sintaxis propia y se define en términos de conceptos LED «normales». Así, las operaciones compuestas incluyen, para el usuario, *estados*, *señales* y *procedimientos* implicados u ocultos. Las *operaciones compuestas* se definen en § 3.1 a 3.3 utilizando nombres ilustrativos para los símbolos LED normales. Cuando se utiliza una *operación compuesta* cualquiera en una representación LED, los símbolos LED implicados tienen nombres implicados únicos. Los nombres implicados se eligen de tal manera que no se produzcan conflictos entre las diferentes apariciones de la *operación compuesta*, o en otros nombres en la representación LED. El usuario está así en libertad de emplear estos nombres para otras finalidades, si lo desea.

Dado que las *operaciones compuestas* están definidas en otros términos LED, puede, en algunas situaciones, producir efectos marginales, que se examinan en las Directrices para el usuario del LED.

3.1 *Valores importados y exportados*

En el LED una *variable* es siempre poseída por, y es local a, una *instancia de proceso* . Normalmente, la variable sólo es visible para la *instancia de proceso* a que pertenece, aunque puede ser *declarada* como un *valor compartido* (véase la Recomendación Z.101), lo que permite a otras *instancias de proceso* en el mismo *bloque ver* el *valor* de la *variable.* Si una *instancia de proceso* en otro bloque necesita *ver* una *variable*, es necesario un intercambio de *señales* con la *instancia de proceso* propietaria de la *variable.*

Los párrafos que siguen describen un método normalizado y una notación abreviada para aplicarlo. Esta técnica se denominará *valor(es) importado(s)* ya que la comunicación es mediante copias del *valor* de la *variable*, siendo solamente la *instancia de proceso* propietaria la que tiene acceso a la *variable* en cuestión. Esta técnica puede utilizarse también para exportar *valores* a otras *instancias de proceso* dentro del mismo *bloque*, en cuyo caso proporciona una alternativa al uso de *valores compartidos.* Esta alternativa es necesaria o bien cuando sea probable que el *bloque* se descompondrá de tal manera que las *instancias de proceso* concernidas caigan en *sub-bloques* diferentes, o cuando los *valores* se utilicen en *condiciones habilitantes* (véase el § 3.3).

La *instancia de proceso* que posee una *variable* cuyos *valores* se ponen a disposición de otras *instancias de proceso* se denomina el *exportador* de la *variable.* Otras *instancias de proceso* que utilizan la *variable* se denominan *importadores* de la *variable.*

El acceso al *valor* de la *variable* se obtiene mediante un intercambio de *señales.* El *importador* envía una *señal* al *exportador*, y espera la respuesta. En respuesta a esta *señal*, el *exportador* devuelve al *importador* una *señal* con el *valor* que tenía la *variable* cuando se efectuó la última *operación de exportación.*

Las *variables* cuyos *valores* son *importados* y *exportados* están definidas como tales en sus *definiciones de variable.* Estas *variables* se identifican también mediante *definiciones* en los *canales* que vehiculan el intercambio de *señales* implicado.

La definición de una *variable* como *exportada* produce una definición implícita de una copa de la misma que se tiene que utilizar en las operaciones de *importación* y *exportación.*

La *instancia de proceso* revela el *valor* de una *variable* definida como *exportada* por medio de la sentencia:

EXPORT(x) donde x es el *nombre* de la *variable.*

EXPORT(x) provoca el almacenamiento del *valor* vigente de x en la copia implícita y el envío de *señales* a *instancias de proceso* que estén a la espera de *condiciones habilitantes* (§ 3.2) o *señales continuas* (§ 3.3).

La operación EXPORT se puede efectuar junto con la manipulación de la *variable* o con independencia de la misma, por ejemplo:

EXPORT (x):= <expresión>;

o

EXPORT (x).

Esta construcción sólo puede aparecer en una *tarea.*

El acceso desde otra *instancia* sólo puede hacerse por medio de la construcción sintáctica:

IMPORT(x, pid) donde x es el *nombre* de la *variable,* y Pid es una referencia a la *instancia de proceso* propietaria.

Esta construcción puede aparecer en una *tarea* y en una *decisión.*

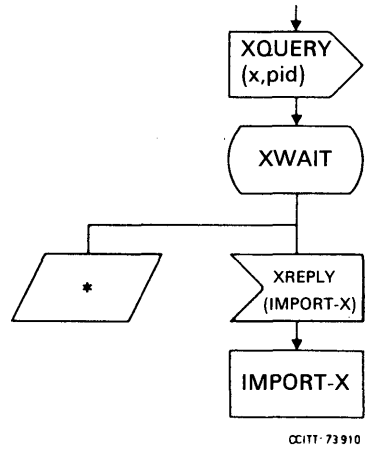
3.1.1 *Definición*

3.1.1.1 *Operación de importación por el importador*

La *operación de importación* se modela en el *gráfico de proceso del importador* por la siguiente secuencia de *nodos*:

- un *nodo de salida,* con el *nombre* XQUERY, que lleva una referencia al *nombre* de la *variable* a cuyo *valor* debe accederse, y direccionada a la *instancia de proceso* propietaria de la *variable* (el *exportador*);
- un *nodo de estado* con un conjunto de *señales de conservación* que incluye todas las *señales,* excepto XREPLY;
- un *nodo de entrada* para la *señal* XREPLY que retorna el *valor* de la *variable* solicitada;
- se asigna a una *variable* implícita del mismo *tipo* que la *variable exportada* el *valor* asociado a la *señal* XREPLY. Este *nombre de variable* implícita sustituye a la operación de *importación.* Cada vez que aparece una operación de importación, se utiliza una nueva *variable* implícita.

La secuencia de operaciones que definen la operación de *importación* se explica a continuación utilizando la sintaxis gráfica del LED. La sentencia *IMPORT*(x, pid) corresponderá a la siguiente *cadena de transición*:



Nota — Este diagrama sólo tiene carácter explicativo. Un usuario de la *operación* de importación escribiría solamente: IMPORT (x,pid).

FIGURA 8/Z.103

Operación de importación explicada en LED/GR

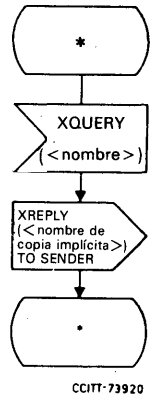
3.1.1.2 *Operación de importación por el exportador*

Como la operación de *importación* implica acciones realizadas por el *exportador,* la siguiente *transición* implicada se añade a cada *estado* del *exportador* (incluidos todos los estados implícitos):

- un *nodo de entrada* para la *señal* XQUERY;
- un *nodo de salida* con el *nombre* XREPLY, que retorna el *valor* de la *variable* solicitada a la *instancia de proceso* solicitante;
- un retorno al *estado* a partir del cual se originó la *transición.*

(Debe observarse que el *exportador* se modifica también con otras *transiciones* si los *valores exportados* se utilizan en *condiciones habilitantes* y/o *señales continuas*; véanse los § 3.2 y 3.3.)

A continuación se muestra la *transición* implicada utilizando la sintaxis gráfica del LED.



Nota – Este diagrama sólo tiene carácter explicativo. Como la *transición* está implicada, el usuario no escribe nada.

FIGURA 9/Z.103

Operación de importación por el exportador explicada en LED/GR

3.1.1.3 *Operación de exportación*

La operación de *exportación* es el medio que tiene el propietario de una *variable exportada* para revelar a los importadores el *valor* vigente de la *variable*. Como resultado de la operación de *exportación*, se da a la copia implícita de la *variable* el *valor* vigente de la *variable*. Cuando no hay *condiciones habilitantes* ni *señales continuas*, la operación propiamente dicha se modela como un *nodo de tarea*, asignando el *valor* a la *variable*. Sin embargo, la operación de *exportador* interactúa con los modelos cuando hay *condición habilitante* y cuando hay *señal continua*; véanse los § 3.2 y 3.3.

3.1.2 *Reglas para la utilización de valores importados*

- Todos los *valores importados* tienen que estar definidos como *IMPORTADOS* en el *importador* y como *EXPORTADOS* en el *exportador*. El atributo exportador en una *definición de variable* hace que el *identificador de variable* sea *visible* en el conjunto del sistema.
- La fuente (u origen) y el destino de la *operación de importación* se muestra incluyendo el *nombre* del *valor importado* en una *lista de señales* asociada a un *canal* o a un *trayecto de señal* dentro de un bloque. La fuente y el destino pueden estar en *bloques* diferentes o en el mismo *bloque*.
- Una *instancia de proceso* que *importa valores* puede también *exportar valores*, pero no podrá *importar valores* de sus propias *variables*.
- Un valor importado <nombre> es *importado* utilizando la sentencia:

  IMPORT(<nombre>, <pid>)

  en una *tarea*, *decisión* o *condición habilitante*. El <nombre> es el *nombre* del ítem importado y el <pid> es el *identificador de instancia de proceso* de la *instancia de proceso* propietaria. Todas las otras referencias en el *proceso* a <nombre> serán interpretadas como referencias a la copia local del *valor* de <nombre>.
- La revelación del *valor* de una *variable* exportada se efectúa por medio de la sentencia:

  EXPORT(<nombre>):= <expresión>

  o

  EXPORT(<nombres>)

  contenida en una *tarea*.

3.1.3 *Sintaxis concreta*

Los *valores importados* son referenciados en tres puntos en una *definición de proceso.* La sintaxis para éstos es la misma en *LED/PR* y *LED/GR.*

3.1.3.1 En *definiciones de variable* y *definiciones de importación*, las palabras clave *IMPORTADO* y *EXPORTADO* identifican el uso que ha de hacerse de la *variable* declarada.

DEFINICIÓN DE VARIABLE

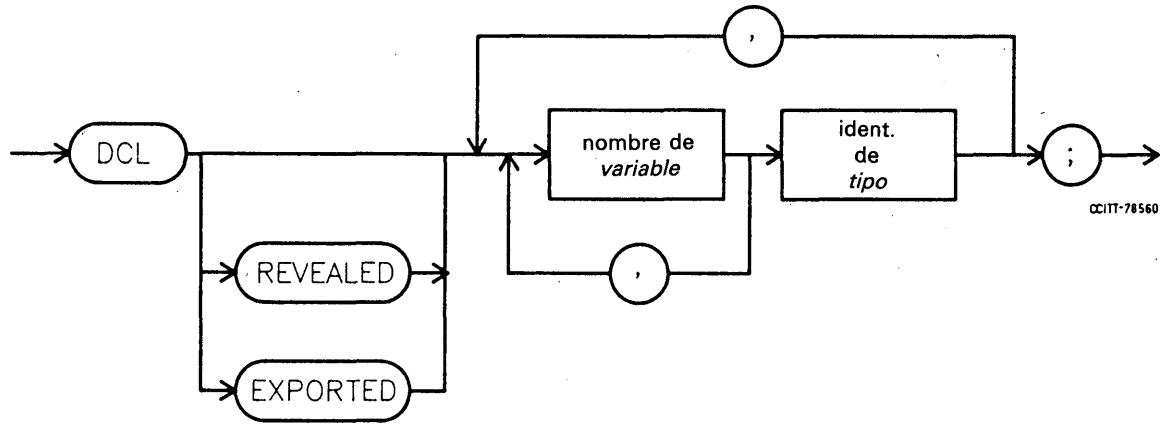


FIGURA 10/Z.103 (1 de 2)

Diagramas de sintaxis para una definición de variable y una definición de importación

DEFINICIÓN DE IMPORTACIÓN

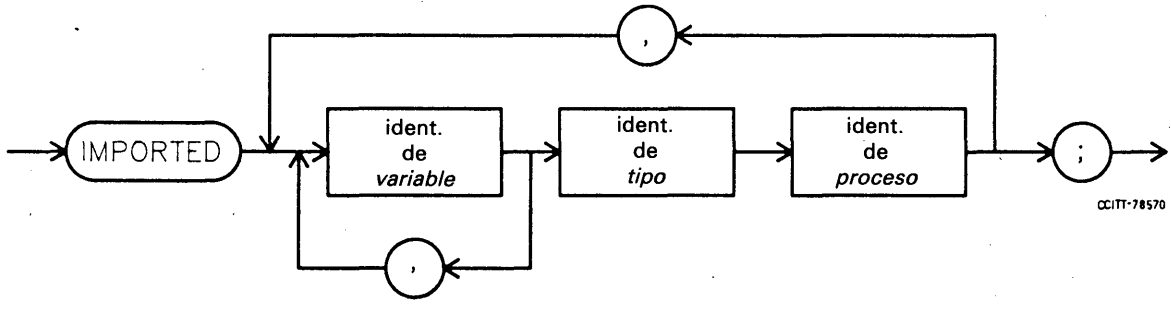


FIGURA 10/Z.103 (2 de 2)

Diagramas de sintaxis para una definición de variable y una definición de importación

3.1.3.2 En *listas de señales* en LED/GR un *valor importado* se distingue de una *señal* encerrando el *nombre* del *valor importado* entre corchetes curvos.

```
⎡ Nombre-señal-1
⎢ (Nombre de variable)
⎣ Nombre-señal-2 ⎤
```

La sintaxis en LED/GR es:

DEFINICIÓN DE CANAL

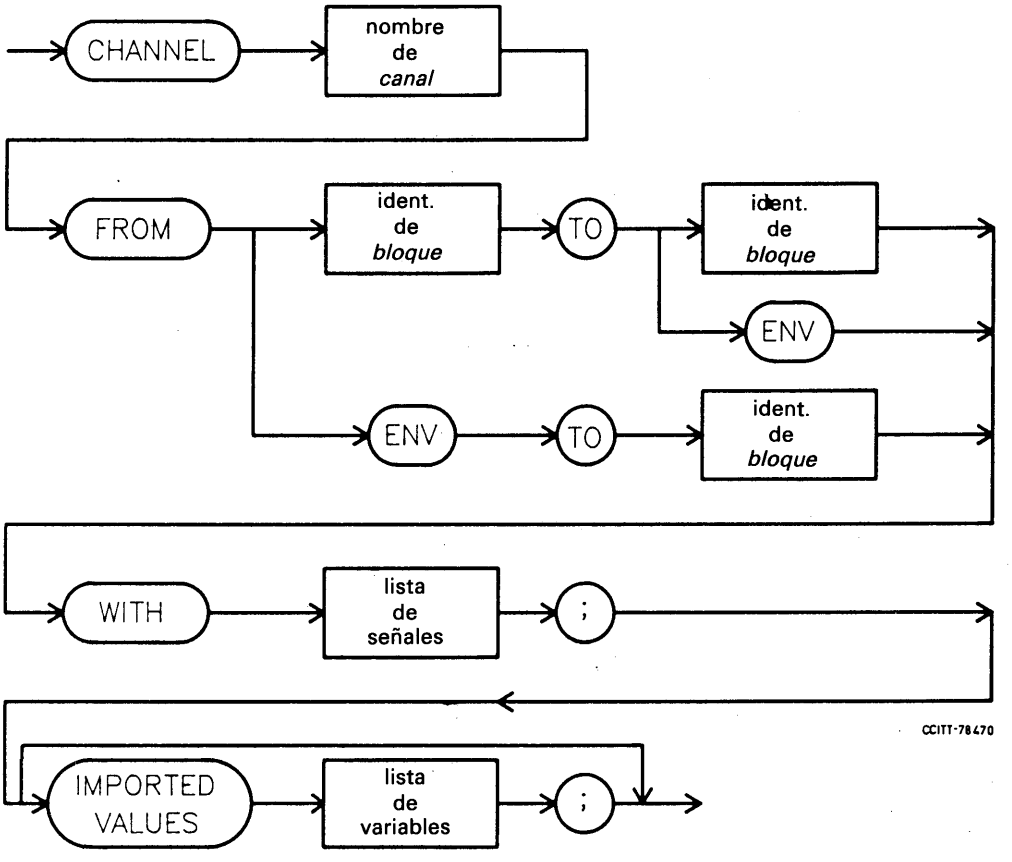


FIGURA 11/Z.103 (1 de 2)

Diagramas de sintaxis para una definición de canal y una lista de variables

LISTA DE VARIABLES

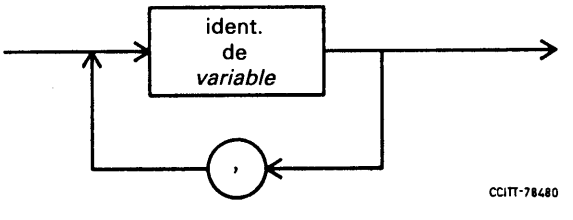


FIGURA 11/Z.103 (2 de 2)

Diagramas de sintaxis para una definición de canal y una lista de variables

3.1.3.3 La sentencia *EXPORT* (<nombre>) puede utilizarse en un *nombre de tarea*; de manera similar, la sentencia *IMPORT*(<nombre>, <pid>) puede utilizarse en *nombres* de *tarea, decisión* y *condición habilitante.*

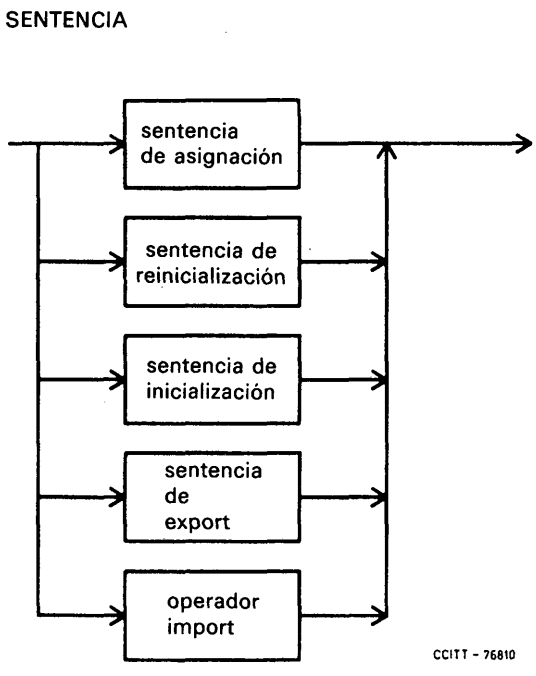


FIGURA 12/Z.103 (1 de 3)

Diagramas de sintaxis para una sentencia de exportación

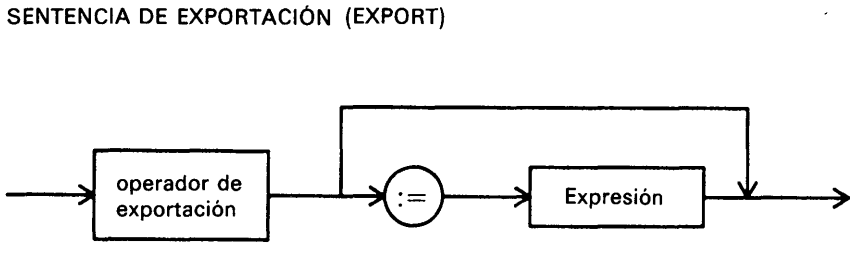


FIGURA 12/Z.103 (2 de 3)

Diagramas de sintaxis para una sentencia de exportación

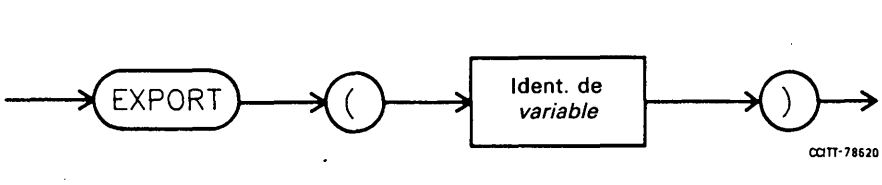


FIGURA 12/Z.103 (3 de 3)

Diagramas de sintaxis para una sentencia de exportación

OPERADOR DE IMPORTACIÓN

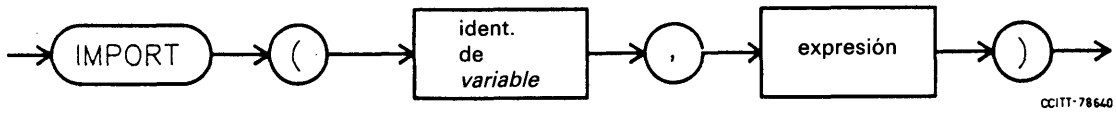


FIGURA 13/Z.103

Diagrama de sintaxis para una sentencia de importación

3.2 *Condición habilitante*

Se puede a menudo conseguir una reducción en el número de *estados* en un *gráfico de proceso* procurando los medios de asociar condiciones al comienzo de *transiciones*. Las condiciones son enunciados en que interviene una variable y se conocen por *condiciones habilitantes*. Si la condición es verdadera (dícese también cierta) y la *señal de entrada* asociada está retenida, la transición se produce normalmente. Si la condición habilitante no es verdadera, la transición queda inhabilitada (dícese también inhibida) y en lugar de la transición se produce una *conservación* de la *señal de entrada*.

En el LED normal esta ejecución condicional de *transiciones* habría que modelarla con *estados* separados para cada valor de la *condición habilitante*. La notación concisa proporcionada por el empleo de *condiciones habilitantes* es útil para la simplificación del *gráfico de proceso*.

Las *condiciones habilitantes* pueden expresarse utilizando valores locales y/o *valores importados*. No podrán utilizarse *valores compartidos*.

La *condición habilitante* se describe en términos de conceptos del LED básico (véase la Recomendación Z.101). Una *condición habilitante* se utiliza en un *gráfico de proceso* y está asociada a un símbolo de *entrada* o a una sentencia de *entrada*. Es un enunciado en que intervienen *valores de datos* que son locales o *valores importados*. La condición tiene que ser una expresión booleana, y por tanto retorna los valores VERDADERO (o CIERTO) (TRUE) o FALSO (FALSE). Cada *estado* que va seguido de *transiciones* controladas por *condiciones habilitantes* de hecho representa un conjunto de *estados*, uno para cada combinación posible de las *condiciones habilitantes*. Las condiciones se evalúan antes de que se seleccione, como el *estado* siguiente, un miembro apropiado del conjunto de *estados*.

La interpretación de una *condición habilitante* depende de los *valores de datos* utilizados en la condición. Si una *condición habilitante* sólo contiene *valores* que son locales al *proceso*, sólo hay que evaluarla una vez. Si contiene *valores importados* (véase el § 3.1), la condición debe evaluarse cada vez que cambian los *valores importados*: la evaluación se invoca por un intercambio de *señales* implicado entre el *importador* y el *exportador* o *exportadores* de los *valores*.

Las *condiciones habilitantes* que utilizan *valores importados* emplean, además de XQUERY definida en el § 3.1, otras dos *señales*, ocultas, que son XATTACH y XDETACH. Estas señales son peticiones al *exportador* de que añada o suprima el *importador* en una lista de las *instancias de proceso* a las cuales es necesario informar los cambios del *valor exportado*. El *exportador* mantiene tal lista implícita, XLIST, para cada *valor* que *exporta*. Existe otra *señal*, XWAKE, que la envía el *exportador* a cada miembro de la lista cada vez que efectúa una operación de *exportación* en el correspondiente *ítem de datos*.

3.2.1 *Definición*

3.2.1.1 *Condición habilitante única basada en valores locales*

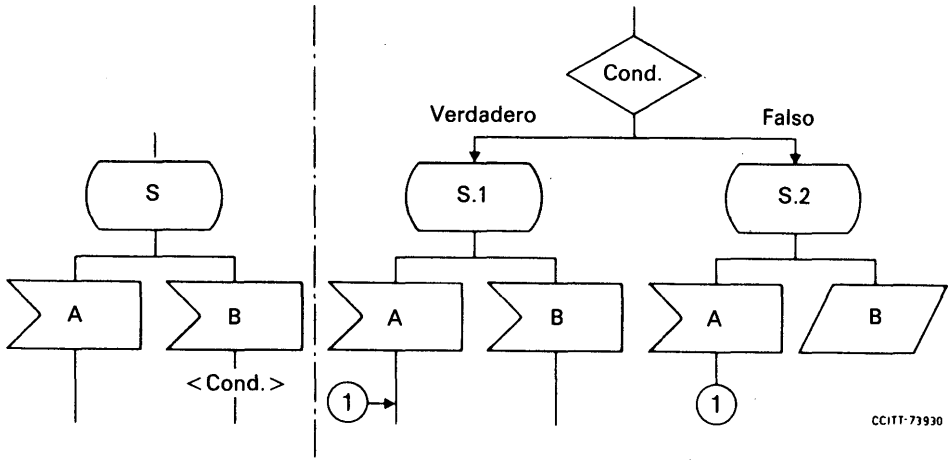
Cuando una sola de las *entradas* que siguen a un *estado* tiene asociada una *condición habilitante*, y cuando esa condición contiene sólo *valores* locales, la interpretación es la siguiente:

El *estado*, las *entradas* y *conservaciones* siguientes son reemplazadas por:

- un *nodo de decisión* que evalúa la *condición habilitante*;
- la rama «VERDADERO» de la *decisión* va seguida por un *nodo de estado* con el mismo *conjunto de señales de conservación* que el *estado* original y seguido por el mismo conjunto de *entradas* que el *estado* original (la *entrada* a la cual estaba asociada una *condición habilitante* aparece como un *nodo de entrada*) ;
- la rama «FALSO» de la *decisión* va seguida por un *nodo de estado* que tiene el *conjunto de señales de conservación* del *estado* original ampliado con el *nombre de señal* de la *entrada* que tiene asociada la *condición habilitante*;
- cada uno de los *nodos de entrada* va seguido de las mismas *transiciones* que aparecen en el gráfico original.

En la siguiente figura se muestra la definición de la *condición habilitante* única basada en *valores* locales solamente, utilizándose para ello la sintaxis gráfica de LED.

Observación – El ususario escribe esto:



Observación — Este diagrama sólo tiene carácter explicativo.

FIGURA 14/Z.103

Condición habilitante única basada en valores locales solamente, explicada en LED/GR

3.2.1.2 *Condición habilitante única que contiene valores importados*

Cuando una sola de las *entradas* que siguen a un *estado* tiene asociada una *condición habilitante* que contiene por lo menos un *valor importado*, la interpretación conducirá a un intercambio de *señales* implicado con los propietarios de los *valores importados.*

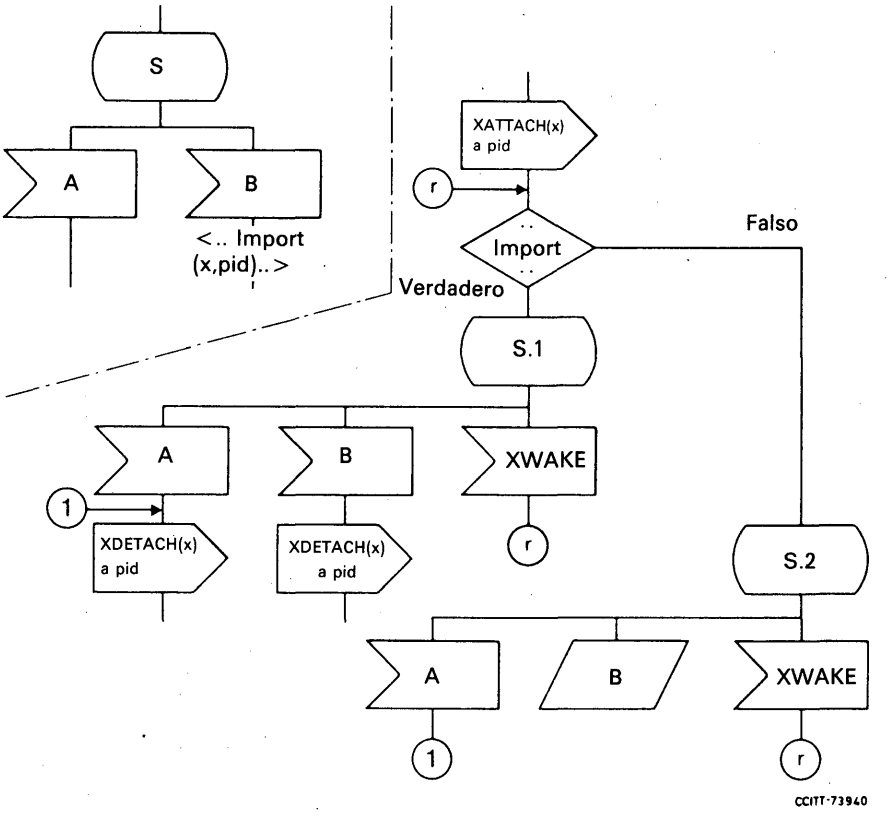
Como las *variables* que retienen los *valores importados* son manipuladas por otras *instancias de proceso*, que trabajan concurrentemente, el *valor* de la *condición habilitante* puede cambiar mientras la *instancia de proceso* está en espera de un *estado.* Todo cambio de ese género debe conducir a una nueva evaluación de la condición. Cuando se introduce una construcción de *condición habilitante*, los propietarios de los *valores importados* son informados de que se debe notificar a esta *instancia* los eventuales cambios de *valor* que se produzcan. Cuando se cambia un *valor importado*, se envía también una *señal* a todos los *procesos* que deben ser informados, a fin de que éstos procedan a una nueva evaluación de las condiciones.

La definición es como sigue:

El conjunto de *señales de entrada válidas* del *proceso* que contiene la *condición habilitante* es ampliado por la *señal* XWAKE implícita. El *estado* y el conjunto de *entrada* y *conservaciones* que le sigue son reemplazados por:

- una secuencia de *nodos de salida* para la señal XATTACH (una para cada uno de los *valores importados* utilizados en la *condición habilitante*);
- la secuencia de la XATTACH va seguida por una *decisión* y dos *nodos de estado*, como en el § 3.2.1.1;
- la *decisión*, por el hecho de contener *valores importados*, se amplía como se indica en el § 3.1.1.2, de manera que vaya precedida por *operaciones de importación* para cada *valor importado* que utiliza;
- además de la ampliación definida en el § 3.2.1.1, los dos *nodos de estado* son seguidos también por un *nodo de entrada* para la *señal* XWAKE, y la transición siguiente hace retroceder a la *decisión* que evalúa la condición;
- además de la ampliación definida en el § 3.2.1.1 cada *nodo de entrada*, con excepción de XWAKE, va seguido de una secuencia de *nodos de salida* que conducen a la *señal* XDETACH (una para cada uno de los *valores importados* utilizados en la *condición habilitante*), después de lo cual la secuencia va seguida de la misma *transición* que aparece en la construcción original.

Observación — El usuario dibuja esto:



Observación — Este diagrama sólo tiene carácter explicativo.

FIGURA 15/Z.103

Condición habilitante única basada en valores importados, explicada en LED/GR

3.2.1.3 *Acciones que debe realizar el exportador cuando se utilizan valores importados en condiciones habilitantes*

Cuando un *valor exportado* es utilizado en una *condición habilitante* el *exportador* tiene las siguientes propiedades implícitas además de las definidas en los § 3.1.1.2 y 3.1.1.3.

Para invocar la reevaluación de *condiciones habilitantes* que utilizan *valores importados*, el *exportador* mantiene explícitamente una lista, XLIST, de *instancias de proceso* que deben ser informadas cuando los *valores* cambian. Las *instancias de proceso* se incorporarán a esa lista enviando la *señal* XATTACH, y se retirarán de la lista enviando la *señal* XDETACH.

Cada vez que se *exporta* un nuevo *valor* por medio de la operación de *exportación*, todas las *instancias de proceso* en la XLIST serán informadas de este nuevo *valor* por medio de la *señal* XWAKE.

Las definiciones de la adición son:

Cada *proceso* que *exporta valores* es ampliado:

a) añadiendo las *señales* implícitas XATTACH y XDETACH al conjunto de *señales de entrada válidas*;

b) definiendo una lista XLIST.<nombre> para cada valor que es *exportado.* La lista retendrá *identificadores de instancia de proceso.*

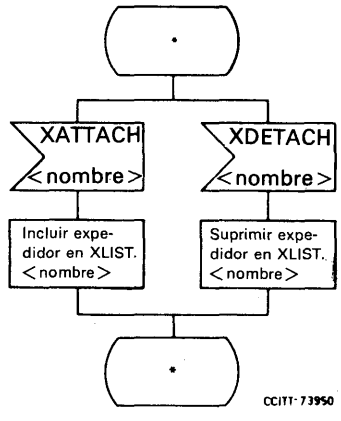
A cada *estado* del *proceso*, incluidos los *estados* implicados, se añaden dos *transiciones* implicadas, además de la *transición* implicada definida en el § 3.1.1.2. La primera de estas *transiciones* consiste en:

- un *nodo de entrada* para la *señal* XATTACH;
- una *tarea* que añade el *identificador instancia de proceso* del expedidor de la *señal* XATTACH a la XLIST correspondiente a la *variable* nominada en la *señal* cuyo *valor* se *exporta*;
- un retorno al *estado* en que se originó la *transición.*

La segunda *transición* implicada consiste en:

- un *nodo de entrada* para la *señal* XDETACH;
- una *tarea* que suprime el *identificador de instancia de proceso* del expedidor de la *señal* XDETACH en la lista nominada en la *señal*;
- un retorno al *estado* en que se originó la *transición.*

Las *transiciones* implicadas adicionales de cada *estado* del *exportador* se muestran a continuación mediante la sintaxis gráfica del LED:



Observación — Este diagrama sólo tiene carácter informativo.
Todas las transiciones están implicadas.

FIGURA 16/Z.103

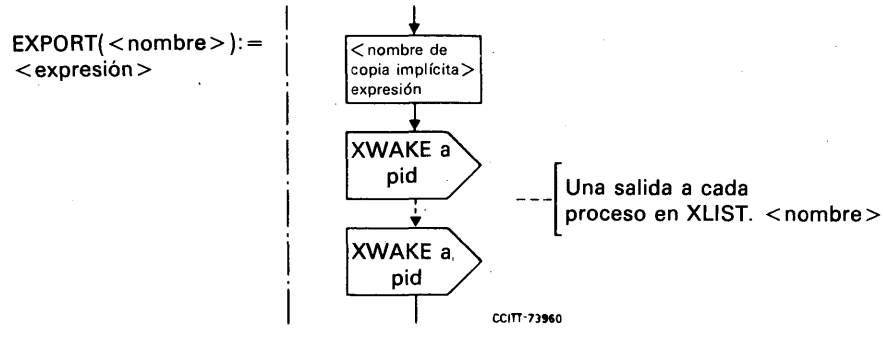
Transiciones implicadas del proceso exportador, explicadas en LED/GR

La siguiente *transición* implicada se agrega a la definición de operación de *exportación*, § 3.1.1.3:

- a continuación de la *tarea* que asigna el nuevo *valor* a la *variable* cuyo *valor* se *exporta* va una secuencia de *nodos de salida* para la *señal* XWAKE que transporta el nuevo *valor* asignado al ítem. Se enviará una *señal* XWAKE a cada *instancia de proceso* contemporáneamente incluida en la XLIST para la correspondiente *variable* a la que se asignó el *valor.*

La adición a la *operación de exportación* se muestra a continuación utilizando la sintaxis gráfica del LED:

El usuario escribe:



Observación — Este diagrama sólo tiene carácter explicativo.

FIGURA 17/Z.103

Acciones implicadas por la operación de exportación explicadas en LED/GR

3.2.1.4 *Condiciones habilitantes múltiples*

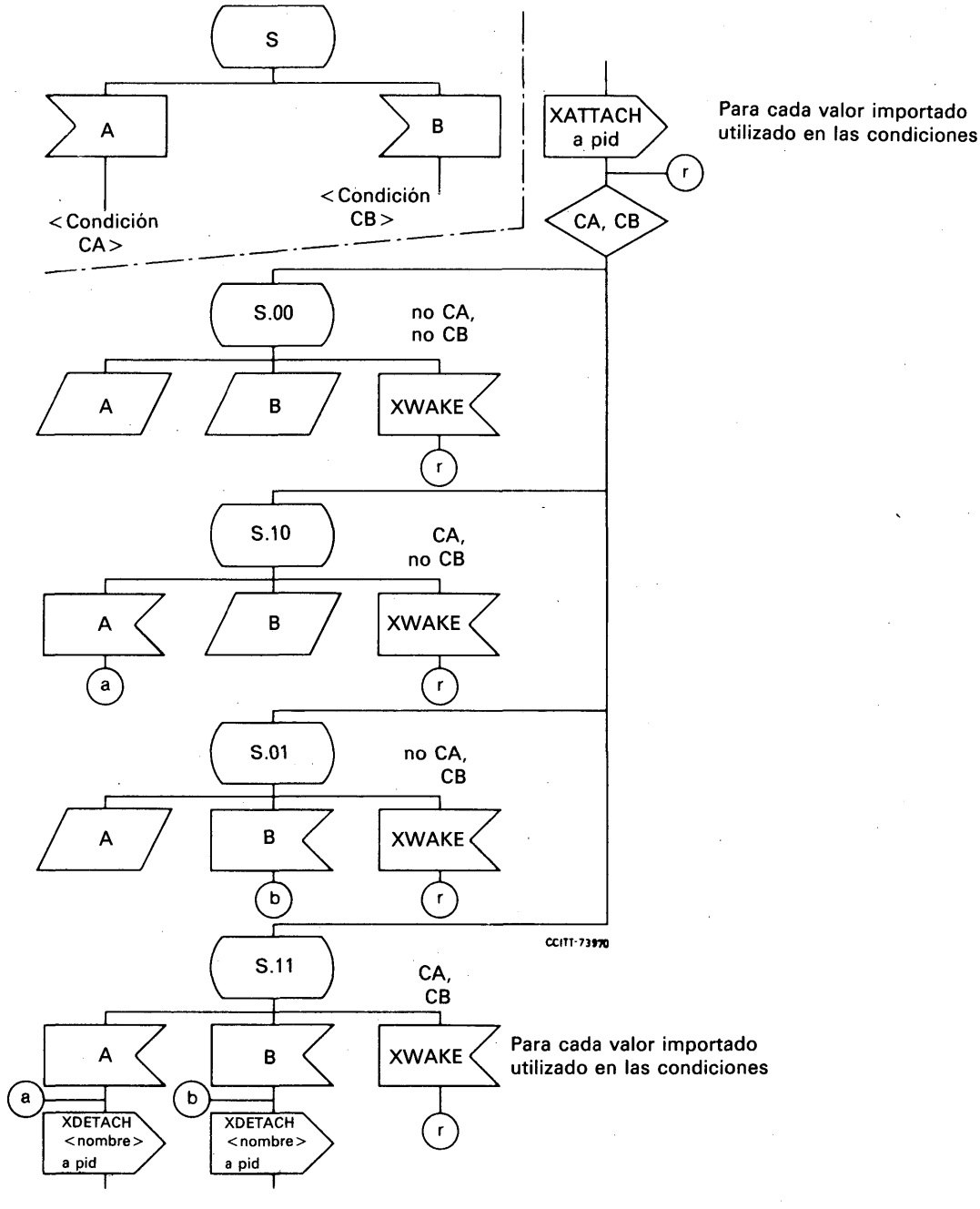
La existencia de varias *condiciones habilitantes* que siguen a un *estado* se conoce por *condiciones habilitantes* múltiples.

Si n de los *nodos de entrada* que siguen a un *estado* tienen asociadas *condiciones habilitantes*, las definiciones de *condiciones habilitantes* simples se extenderán como sigue:

a) El *estado* y las *entradas* y *conservaciones* siguientes se representan por un *nodo de decisión* para una condición compleja cuyo *valor* es una n-tupla, con elementos, el *valor* de las *condiciones habilitantes elementales*. Esto va seguido de 2**n *nodos de estado*, cada uno de los cuales va seguido por el procesamiento de *transición* que es apropiado para los *valores* particulares de las *condiciones habilitantes* que aparecen nombradas en el arco que conduce a ese *nodo de estado*.

b) Si una cualquiera de las *condiciones habilitantes* está referida a *valores importados*, las definiciones del § 3.2.1.2 son también aplicables.

El usuario escribe esto:



Observación — Este diagrama sólo tiene carácter explicativo.

FIGURA 18/Z.103

Condiciones habilitantes múltiples explicadas utilizando el LED/GR

3.2.2 *Reglas para la utilización de las condiciones habilitantes*

1) Se puede asociar *condiciones habilitantes* a cualquier *señal de entrada.*

2) Un *estado* que contiene una determinada *señal de entrada* sólo puede ir seguido de *una entrada,* independientemente de que la *entrada* tenga o no asociada una *condición habilitante.*

3.2.3 *Sintaxis concreta*

3.2.3.1 *LED/GR*

Las *condiciones habilitantes* se muestran en el *LED/GR* por un *símbolo de condición habilitante* que sigue a un *símbolo de entrada.* El elemento es un par de corchetes angulares que encierran una condición *booleana.* La sintaxis gráfica se muestra en la figura 19/Z.103.

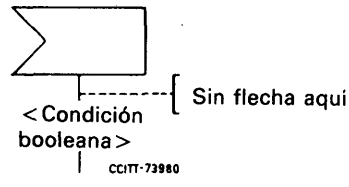


FIGURA 19/Z.103

Sintaxis del LED/GR para la condición habilitante

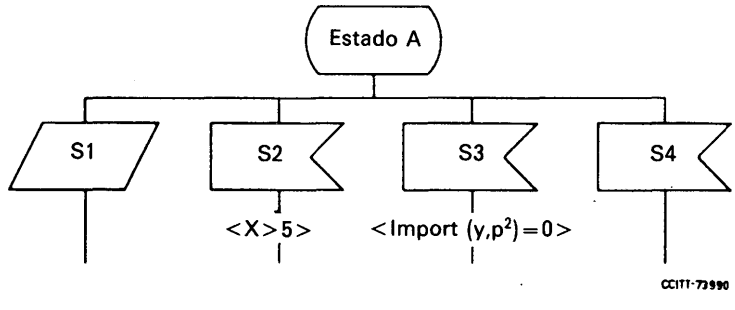


FIGURA 20/Z.103

Ejemplo de utilización de la sintaxis del LED/GR para la condición habilitante

3.2.3.2 *LED/PR*

En el *LED/PR* la sintaxis de la sentencia de *entrada* se modifica añadiendo la frase *SUMINISTRADO (PROVIDED).* La frase *SUMINISTRADO* contiene una expresión *booleana.* El diagrama de sintaxis para la sentencia de *entrada* modificada se da en la figura 21/Z.103

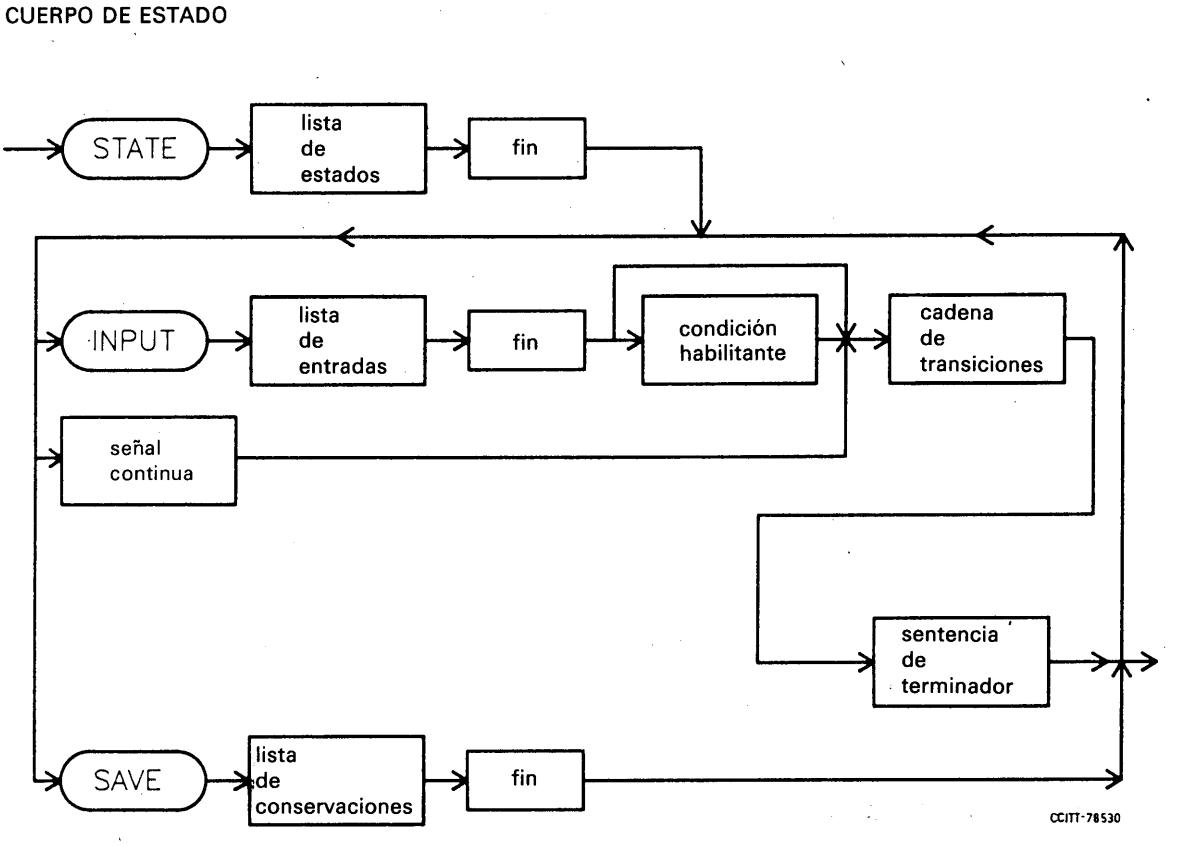


FIGURA 21/Z.103 (1 de 2)

Diagramas de sintaxis para la sentencia de entrada, incluida la frase PROVIDED

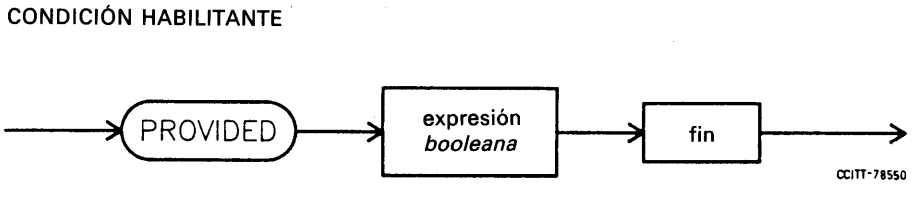


FIGURA 21/Z.103 (2 de 2)

Diagramas de sintaxis para la sentencia de entrada, incluida la frase PROVIDED

3.3 *Señal continua*

Al describir *sistemas* con el LED se presenta a menudo la situación en la cual un usuario quisiera mostrar que una *transición* es causada por un cambio en el *valor* de una *variable* externa al *proceso.* El *valor* pudiera ser, por ejemplo, una tensión alta o baja en una línea o en un número en un registro de status. La manera normal de conseguir esto en LED sería disponer que se genere una *señal* cuando se produce el cambio en el *valor* y basar la *transición* en la recepción de esa *señal.* La necesidad de *definir,* generar y recibir explícitamente esas señales puede complicar los *gráficos de proceso.* La *operación compuesta* denominada *señal continua* permite que un cambio en el *valor* de una condición inicie directamente una *transición.*

Una *condición habilitante* representa una *decisión* antes de pasar a un *estado.* Cuando se utilizan *valores importados,* las *condiciones habilitantes* proporcionan salidas de un *estado* por medio de la *señal* XWAKE implícita. Una *condición habilitante* puede utilizarse también sin una *señal de entrada* asociada cuando se convierte en una *señal continua.* En este caso, la *condición habilitante* no representa un *estado* implicado, suplementario, sino que define una circunstancia en la cual hay un escape desde el *estado* a partir del cual sigue la *condición habilitante.* Esta circunstancia tiene una prioridad inferior a la de las *señales retenidas.*

De un mismo *estado* pueden partir varias *señales continuas* y puede suceder que más de una de ellas sean verdaderas (TRUE) al mismo tiempo. Cada *señal continua* tiene atribuida una prioridad que determina el orden relativo en el que se prueban las *señales continuas.*

Una *señal continua* que sólo utiliza *valores* locales proporciona un medio para salir condicionalmente de un *estado* si en el instante de entrada no hay *señales* en espera en el *puerto de entrada.* Una *señal continua* que utiliza *valores importados* proporciona además la capacidad de reevaluar la condición cuando se produce un cambio en uno de los *valores importados* utilizados en la condición.

3.3.1 *Definición*

La siguiente definición se basa en las definiciones de *importación* y *exportación* de *valores,* véase el § 3.1, y *condiciones habilitantes,* véase el § 3.2.

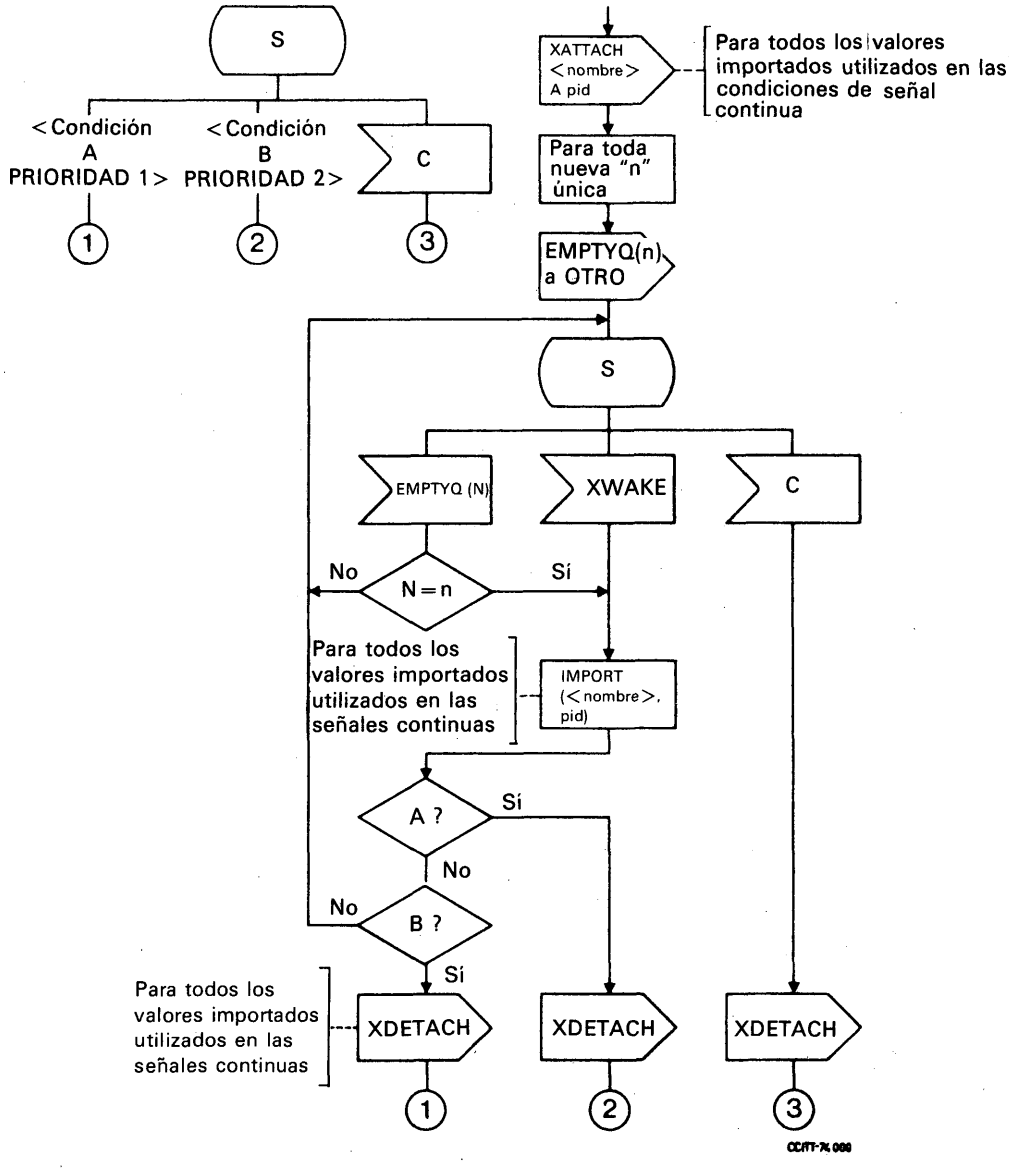
La interpretación de un *estado* que va seguido de *señales continuas* se presenta como un modelo general en el cual hay varias *señales continuas* que colectivamente hacen referencia a varios *valores importados.* Si no hay *valores importados* utilizados, el modelo se simplifica eliminando las *salidas* XATTACH y XDETACH y la *entrada* XWAKE.

El *estado* y el conjunto de *entradas* y *conservaciones* que le siguen, junto con las *señales continuas* se reemplazan por:

1) una secuencia de *nodos de salida* para la señal XATTACH, una para cada uno de los *valores importados* utilizados en la condición de *señal continua* ;
2) una *tarea* que crea un *valor* único que ha de utilizarse en la *señal* EMPTYQ, utilizada en 3);
3) un *nodo de salida* para una *señal* EMPTYQ que se envía a la *identidad de instancia de proceso* del expedidor, es decir, a su propio *puerto de entrada*;
4) un *nodo de estado* como en el *gráfico de proceso* o *gráfico de procedimiento* original, seguido por un conjunto de *nodos de entrada* que incluye el conjunto original y otros dos *nodos de entrada*;
5) cada *nodo de entrada* va seguido de una secuencia de *nodos de salida* para la *señal* XDETACH, uno para cada uno de los *valores importados* utilizados en las condiciones de *señal continua.* Esta secuencia va seguida entonces por el procesamiento de *transición* que seguía originalmente al *nodo de entrada* ;
6) el *nodo de estado* en 4) va también seguido por un *nodo de entrada* para la *señal* XWAKE; ésta inicia la transición en 7);
7) una secuencia de *nodos de tarea* para *importar* cada uno de los *valores importados* utilizados en las condiciones de *señal continua* ;
8) una secuencia de *nodos de decisión* para cada una de las condiciones de *señal continua,* siendo la primera *decisión* evaluada la que corresponde a la *señal continua* de prioridad más elevada (el número más bajo en la sintaxis concreta);
9) la rama FALSE (NO) de cada *decisión* conduce a un *nodo de decisión* para la condición de *señal continua* que sigue en orden de prioridad. La rama FALSE (NO) de la *decisión de señal continua* de prioridad más baja hace retornar al *estado* en 4);
10) la rama TRUE (SÍ) de cada *decisión* conduce a una secuencia de *nodos de salida* para la *señal* XDETACH, una para cada uno de los *valores importados* utilizados en las condiciones de *señal continua,* seguida por el procesamiento de *transición* que corresponde a la condición de *señal continua* probada en la decisión;
11) el *nodo de estado* en 4) va también seguido por un *nodo de entrada* para la *señal* EMPTYQ, el que a su vez va seguido de un *nodo de decisión* que verifica que la *señal* lleva el valor que se le ha dado en 2), es decir, que es la misma *señal* enviada en 3), y no una señal anterior, una *señal* EMPTYQ anterior no procesada. La rama TRUE (SÍ) de esta *decisión* conduce al procesamiento de *señal continua* en 7), y la rama FALSE (NO) hace retornar al nodo de *estado* de 4).

El conjunto de *señales de entrada válidas* del *proceso* que contiene *señales continuas* que utilizan *valores importados* se amplía por la *señal* XWAKE implícita. Los conjuntos de *señales de entrada válidas* de cualquier *proceso* que contienen *señales continuas* son ampliados por la *señal* EMPTYQ implícita.

El usuario escribe esto:



Observación – Este diagrama sólo tiene carácter explicativo.

FIGURA 22/Z.103

Señal continua explicada en LED/GR

3.3.2 *Reglas para la utilización de señales continuas*

1) Una *señal continua* puede seguir a cualquier *estado.*

2) La *condición de señal continua* puede basarse en *valores* locales y/o *valores importados.*

3) Dos *señales continuas* que siguen al mismo *estado* no pueden tener el mismo número de prioridad.

3.3.3 *Sintaxis concreta*

3.3.3.1 *LED/GR*

En el *LED/GR*, una *señal continua* se indica por un *símbolo de condición habilitante* que sigue directamente a un *símbolo de estado*, es decir, la *transición* no va encabezada por un *símbolo de entrada*. El símbolo contiene, además de la condición de *señal continua*, la palabra clave PRIORIDAD (PRIORITY) seguida de un número de prioridad (número natural). Cuanto menor es el número, más elevada es la prioridad de la *señal continua*.

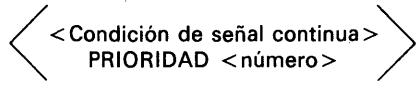


FIGURA 23/Z.103

Símbolo LED/GR para la señal continua

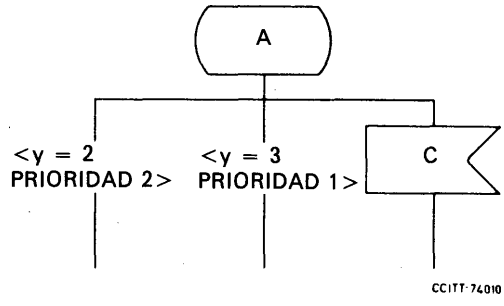


FIGURA 24/Z.103

Ejemplo de la utilización de señal continua LED/GR

Si una sola *señal continua* sigue a un *estado*, se puede omitir la cláusula de PRIORIDAD. Si se omite esta cláusula, el número de prioridad «1» va implicado.

3.3.3.2 *LED/PR*

En *LED/PR*, una sentencia SUMINISTRADO, seguida de una cadena de transición, representa la *señal continua*. La sentencia contiene una cláusula PRIORIDAD. En esta cláusula, cuanto menor es el número, más elevada es la *prioridad* de la *señal continua*.

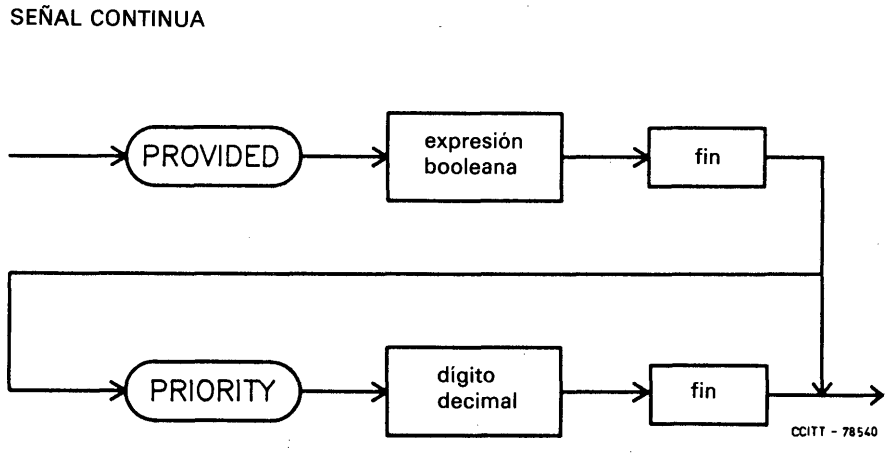


FIGURA 25/Z.103

Diagrama de sintaxis para la señal continua

4 Macros

Un *macro* es una notación abreviada, definida por el usuario, que puede incluirse en uno o más lugares en la representación LED concreta de un *sistema.* Representa una referencia a una definición en un documento que se encuentra en otro lugar. Un *macro* no es más que una parte de la sintaxis concreta y, para interpretar la representación LED en que aparece, es necesario sustituirlo por el cuerpo de su definición.

4.1 *Definición*

Un **macro** puede representar una colección de ítems sintácticos, aunque puede no ser recursivo por razones obvias (expansión infinita i).

El *macro* puede tener cero o más *accesos de entrada* y cero o más *accesos de salida.* Cuando hay más de un *acceso de entrada* debe haber una *etiqueta* asociada a cada *acceso de entrada* que corresponde a la *etiqueta de acceso de entrada* en la definición de *macro*; cuando sólo existe un *acceso de entrada* se puede prescindir de la *etiqueta.* Lo mismo es aplicable a los *accesos de salida.*

El concepto de *macro*, por su propia naturaleza, no lleva asociado un *alcance* o *visibilidad.* La interpretación de una referencia *macro* sólo puede obtenerse cuando el *macro* es sustituido por su definición.

4.2 *Sintaxis concreta*

4.2.1 *LED/GR*

4.2.1.1 *Sintaxis*

La referencia a una *definición de macro* se muestra por el *símbolo de macro* en el *LED/GR.*

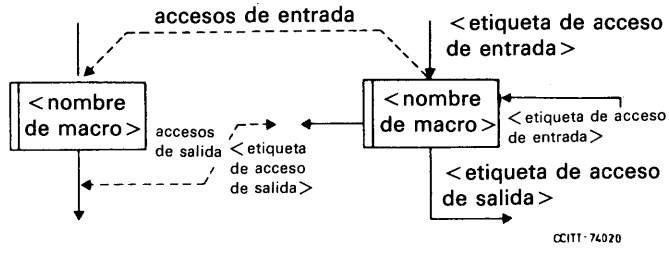


FIGURA 26/Z.103

Símbolo de macro en el LED/GR

Los *accesos de entrada* al *macro* y los *accesos de salida* del *macro* se representan por *líneas de flujo* que conducen al símbolo o salen del símbolo. Facultativamente pueden asociarse *etiquetas* a las líneas de flujo.

El símbolo de *macro* contiene el *nombre* de la definición de *macro* a la que hace referencia; el símbolo puede también tener asociado un *comentario.*

La *definición de macro* lleva el siguiente título:

<nombre> DEFINICIÓN MACRO

donde el <nombre> es el *nombre* del *macro*, utilizado en un *símbolo de macro.*

La *definición de macro* contiene la representación gráfica que reemplaza al *símbolo de macro* antes de que tenga lugar la interpretación. Los *accesos de entrada* a la *definición de macro* y los *accesos de salida* de esta definición se representan por líneas de flujo que respectivamente conducen a, o parten de, los símbolos en la definición. Tanto los *accesos de entrada* como los *accesos de salida* pueden tener asociada *etiquetas.*

4.2.1.2 *Símbolos*

En la figura 27/Z.103 se definen dos símbolos adicionales:

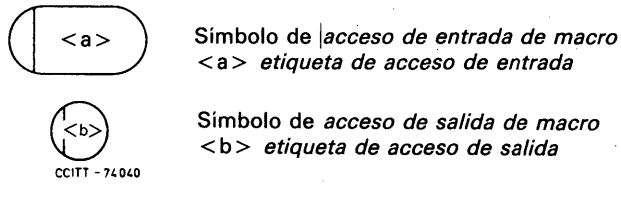


FIGURA 27/Z.103

Símbolos adicionales para macros

4.2.1.3 *Reglas para la utilización de macros en el LED/GR*

El *símbolo de macro* puede insertarse en cualquier lugar en un diagrama, y puede representar una colección cualquiera de símbolos del *LED/GR.*

El número de *accesos de entrada* deberá ser el mismo en un *símbolo de macro* y en la *definición de macro* a que se hace referencia. El conjunto de *etiquetas* en los *accesos de entrada* en el *símbolo de macro* debe ser el mismo que el conjunto de *etiquetas* en los *accesos de entrada* en la *definición de macro.* La misma regla se aplica a los *accesos de salida* y las *etiquetas de acceso de salida.*

Los *accesos de entrada/accesos de salida* pueden terminar/originarse en cualquier lado del símbolo.

Dado que la semántica de un *macro* se obtiene sustituyendo la referencia por la colección de símbolos en la definición, todos los convenios relativos a la representación gráfica son aplicables al diagrama en el cual el *macro* ha sido reemplazado en todos los lugares en que aparecía. Esto puede dar lugar a consecuencias imprevistas al aplicarse ciertas reglas, por ejemplo la múltiple aparición de *estados.* Esta situación se examina con mayor profundidad en las Directrices para el usuario del LED.

4.2.2 *LED/PR*

4.2.2.1 *Sintaxis*

La llamada macro se puede insertar en cualquier diagrama utilizando la sintaxis:

MACRO nombre de macro;

La ampliación macro es un elemento de un programa LED/PR que comienza con:

MACRO EXPANSION nombre de macro;

y termina con:

ENDMACRO nombre de macro;

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

Esta definición debe colocarse inmediatamente después de la construcción SISTEMA, BLOQUE o PROCESO respectivamente, lo que dependerá del lugar en que el *macro* es referenciado. La *definición de macro* puede contener cualquier carácter y sólo se podrá determinar si es o no correcta después que haya reemplazado a la sentencia *macro.*

4.2.2.2 *Reglas para la utilización de macros en el LED/PR*

En el *LED/PR* se aplicarán las mismas reglas y se tendrán presentes las mismas preocupaciones que cuando se utilizan *macros* en el *LED/GR.*

5 Opciones

Cuando varias aplicaciones similares se especifican o describen utilizando el mismo LED, en muchas ocasiones la misma *definición de proceso* puede utilizarse en los diferentes *sistemas* si se le introduce una pequeña modificación. La facilidad *OPCIÓN* en el LED proporciona un medio de definir *procesos* genéricos para varias aplicaciones, introduciendo partes opcionales alternativas de las descripciones.

5.1 *Definición*

Una *opción* es la selección de partes alternativas de una *definición de proceso,* de acuerdo con la evaluación de una *expresión de opción.* La selección se hace antes de que se interprete la *definición de proceso.*

La facilidad *opción* no es más que una parte de la sintaxis concreta, y se debe considerar como una notación abreviada que da una descripción genérica para varias aplicaciones, más bien que una descripción específica para cada aplicación.

5.2 *Sintaxis concreta*

5.2.1 *LED/GR*

5.2.1.1 *Símbolos*

El siguiente símbolo se utiliza en el *LED/GR* para representar *opción*:

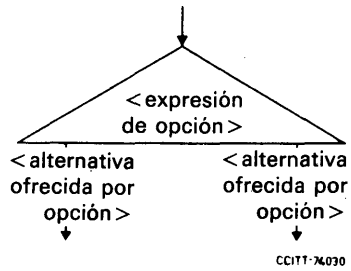


FIGURA 28/Z.103

Símbolo de opción en el LED/GR

5.2.1.2 *Reglas para la utilización de opciones en el LED/GR*

El símbolo de *opción* puede seguir a un símbolo de *tarea*, un acceso de salida de *símbolo de decisión*, un *símbolo de salida* o un *símbolo de procedimiento* en un *diagrama de proceso*. El *símbolo de opción* puede ir seguido de un *símbolo de estado*, un *símbolo de tarea*, un *símbolo de decisión*, un *símbolo de salida* o un *símbolo de procedimiento*.

La *<expresión de opción>* contenida en el símbolo es una expresión que tiene estas dos particularidades: por un lado, después de evaluada, entre las *<alternativa ofrecida por opción>* que siguen al símbolo, se elige una, y sólo una, y por otro lado puede ser evaluada antes de interpretar el proceso. Cada *alternativa ofrecida por opción* debe ser un *valor* del mismo *tipo* que el de la *expresión de opción*. Cuando se interpreta el *proceso* resultante, las partes inalcanzables de la *definición de proceso* deben considerarse suprimidas.

5.2.2 *LED/PR*

5.2.2.1 *Sintaxis*

La *opción* se representa en el *LED/PR* por la siguiente sintaxis:

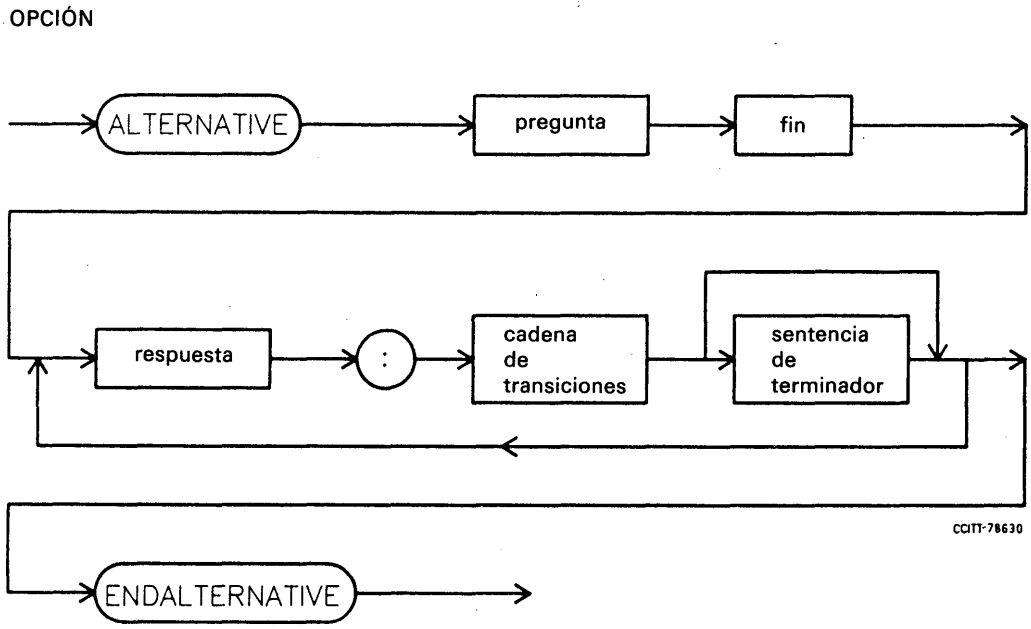


FIGURA 29/Z.103

Diagrama de sintaxis para una opción

5.2.2.2 *Reglas para la utilización de opciones en el LED/PR*

La < *expresión de opción* > contenida en la sentencia es una expresión que tiene esas dos particularidades: por un lado, después de evaluada, entre las < *alternativa ofrecida por opción*> que siguen al símbolo se elige una y sólo una y por otro lado, se evalúa antes de interpretar el *proceso.* Cada < *alternativa ofrecida por opción*> debe ser un *valor* del mismo *tipo* que la *expresión de opción.* Cuando se interpreta el *proceso* resultante, las partes inalcanzables de la *definición de proceso* deben considerarse suprimidas.

6 Elementos pictográficos en el LED/PR

Cuando se utiliza la sintaxis gráfica del LED para representar *definiciones de proceso,* la utilización de *elementos pictográficos* para construir un *pictograma de estado* dentro de un *símbolo de estado* es una parte opcional del *LED/GR.*

Tales *pictogramas de estado* pueden ofrecer ventajas cuando se aplican a *ciertas definiciones de sistema,* pues permiten obtener diagramas de proceso más compactos y con menos texto. El *pictograma de estado* describe en términos de *elementos pictográficos* y texto calificador, el status real del *proceso* cuando se está en ese *estado.* En el *pictograma de estado* se puede describir también el status supuesto del entorno del *proceso.* Cuando se utilizan *pictogramas de estado,* las acciones que han de ejecutarse en *transiciones* entre los *estados* están implicadas y se deducen analizando las diferencias de los status descritos.

Cuando se utilizan *elementos pictográficos,* son aplicables la sintaxis y la semántica definidas en la Recomendación Z.101. Sin embargo, estas semántica y sintaxis están ampliadas como se define en los párrafos siguientes.

6.1 *Semántica de los pictogramas de estado*

Cuando se utilizan *elementos pictográficos*, un *nodo de estado* se representa por un *símbolo de estado.* El *símbolo de estado* se identifica por su *nombre,* y contiene un *pictograma de estado* que consiste en *elementos pictográficos, valores de variables, variables de entrada* y texto calificador.

Un *pictograma de estado* puede representar:

- utilizando *elementos pictográficos* y texto calificador, los *valores* en ese *estado* en un subconjunto seleccionado del conjunto total de *variables* asociadas con ese proceso. El subconjunto seleccionado puede incluir *variables* que sirven, simplemente, de «procuradores» («proxy») de *variables* asociadas con otros *procesos.* Estas *variables* «procuradoras» llevan el *valor* de *variables* asociadas con otros *procesos.* Estos *valores* se obtienen por *visión* o *importación* del *valor* ;
- utilizando los *valores* de *variables de entrada*, las *acciones de entrada* con respecto a las *señales de entrada válidas* para ese *estado.*

El repertorio de *elementos pictográficos* es en principio ilimitado, pues pueden crearse nuevos *elementos pictográficos* para responder a toda nueva aplicación del LED. No obstante, se ha considerado que, para aplicaciones de conmutación y señalización en telecomunicaciones, el siguiente repertorio de *elementos pictográficos* es satisfactorio, pues su versatilidad es considerable:

- frontera de proceso (izquierda o derecha),
- equipo terminal (diverso),
- receptor de señalización,
- emisor de señalización,
- emisor y receptor de señalización combinados,
- supervisión de un proceso por temporizador,
- trayecto de conmutación (conectado, reservado),
- módulo de conmutación,
- tasación en curso,
- elemento de control,
- símbolo de incertidumbre.

Los símbolos normalizados para estos *elementos pictográficos* se especifican en el § 6.3.

6.2 *Reglas de interpretación*

1) Las *variables de entrada* son *variables booleanas* y cada *variable de entrada* responde a una y solamente una *señal* del conjunto de *señales de entrada válidas* para el *proceso.* Un cambio en el *valor* de una *variable de entrada* entre *pictogramas de estado* representa siempre consunción de una *señal* por una *acción de entrada* de un *proceso.* En consecuencia, se pueden utilizar *variables de entrada* para representar las condiciones de un *proceso* que, si cambian, hacen que el proceso efectúe una *transición.* Los *valores* de una *variable de entrada* pueden asociarse con un *elemento pictográfico.*

2) La presencia de un *elemento pictográfico* en un *pictograma de estado* indica *valores* específicos para un subconjunto de *variables* mientras ese *proceso* está en ese *estado*. Los *valores* de *variables* adicionales, particularmente los asociados con este subconjunto inicial, pueden ser indicados mediante la calificación del *elemento pictográfico* por un texto calificador. El texto calificador no es una *variable de entrada*; los cambios en el texto calificador NO representan la consunción de una *señal de entrada* por una *acción de entrada* del *proceso*.

3) Posición:

   a) La posición de un *elemento pictográfico* cualquiera (que no sea una *frontera de proceso*) con relación a una *frontera de proceso* determina si el *elemento pictográfico* es «interno» o «externo» al *proceso*. Un *elemento pictográfico* interno representa *variables* que pertenecen al proceso. Un *elemento pictográfico* externo representa *variables* que pertenecen a otro proceso, de forma que para tener acceso a estas *variables* se tiene que utilizar el mecanismo de *visión* e *importación*.

   b) La regla a) es también aplicable a la distinción entre texto calificador interno y externo, sustituyendo «*elemento pictográfico*» por el término «texto calificador» en esta regla.

4) Regla cardinal:

   El procesamiento total cuando se pasa de un *estado* al *estado* siguiente es el requerido para efectuar los cambios en los *pictogramas de estado*, así como el procedimiento indicado en las *decisiones*, *salidas* o *tareas* que aparecen en la *transición* entre los *estados*. Así:

   a) El cambio de la aparición de un *elemento pictográfico* interno en un *estado* a la ausencia de ese *elemento pictográfico* en el *estado* siguiente, o viceversa, corresponde a un cambio en los *valores* de algunas *variables* que puede ser representado equivalentemente mediante una *tarea* en la *transición* entre los *estados*.

   b) El cambio de la aparición de un *elemento pictográfico* externo en un *estado* a la ausencia de ese *elemento pictográfico* en el *estado* siguiente, o viceversa, corresponde a un cambio en *variables* pertenecientes a otro *proceso*. Este cambio puede representarse equivalentemente o bien por una *señal de salida* hacia ese otro *proceso* o sencillamente por la *señal de entrada* desde ese *proceso*.

   c) Las reglas a) y b) son también aplicables a la aparición o desaparición, en el *pictograma de estado*, de un texto calificador, sustituyendo *elemento pictográfico* por el término «texto calificador» en esas reglas.

5) Para un *diagrama de proceso* dado, un determinado *elemento pictográfico* (o una determinada combinación de *elemento pictográfico* y texto calificador) tiene una posición única dentro del *pictograma de estado*, de modo que la presencia o ausencia de este *elemento pictográfico* (o de esta combinación) en un símbolo de *estado* puede determinarse rápidamente comparando el *pictograma de estado* con otros pictogramas de estado del *diagrama de proceso*.

6) Cuando un *emisor de señalización* aparece en un *pictograma de estado*, su texto calificador identifica una *señal* que ha sido objeto de una *salida* antes de este *estado*, o (en el caso de una *señal continua* controlada por el *proceso*), antes y durante este *estado*.

6.3 *Símbolos recomendados para los elementos pictográficos*

Cuando se utilizan *elementos pictográficos*, cada *estado* está representado por un símbolo de *estado* que contiene un *pictograma de estado* cuyo formato se muestra en la figura 30/Z.103:

Se ha normalizado un conjunto básico de *elementos pictográficos* para uso en el *LED/GR* con aplicación a la *descripción del sistema* de procesos de tratamiento de llamadas de telecomunicaciones, incluidos protocolos de señalización, servicios de red y procesos de interfuncionamiento de señalización. Muchos de estos *elementos pictográficos* pueden ser utilizados en aplicaciones del *LED/GR* que van más allá de los procesos de tratamiento de llamadas, y se estimula su utilización en otros procesos de telecomunicaciones, cuando proceda.

Los símbolos recomendados para el conjunto básico de *elementos pictográficos* se muestran en la figura 31/Z.103 siguiente:

La elección de dibujos para los *elementos pictográficos* se ha basado en las consideraciones y criterios de selección generales presentados en el anexo A a esta Recomendación, que debe consultarse antes de crear símbolos de *elementos pictográficos* adicionales para aplicaciones más vastas del *LED/GR*.

Las proporciones recomendadas para los símbolos de *elementos pictográficos* se indican en el anexo B a esta Recomendación.

La plantilla insertada en la cubierta posterior de este fascículo facilita el dibujo manual del conjunto básico de los símbolos del *LED/GR*, e incluye el conjunto básico de símbolos de *elementos pictográficos* indicados en la figura 31/Z.103.

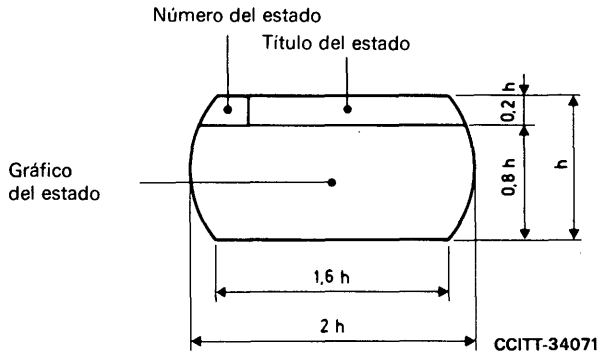


FIGURA 30/Z.103

Formato recomendado de un símbolo de estado con pictograma de estado

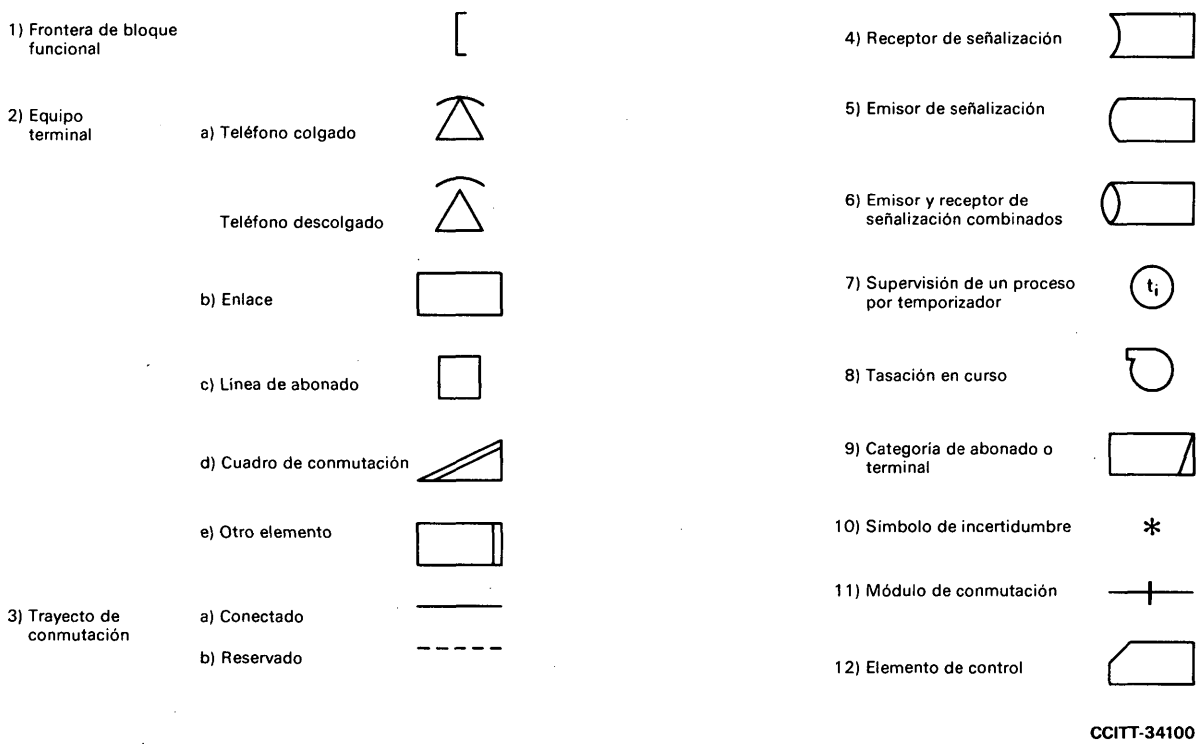


FIGURA 31/Z.103

Símbolos recomendados para el conjunto básico de conceptos de elemento pictográfico

6.4 *Convenios e interpretaciones especiales aplicables a la ampliación del LED/GR basada en estados*

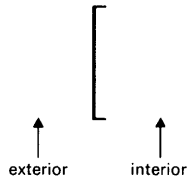
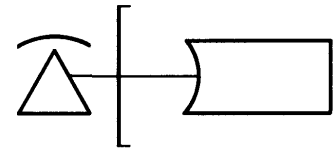
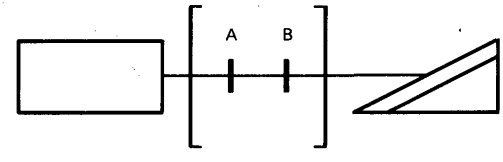
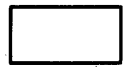
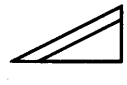
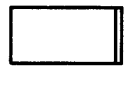
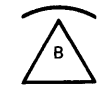
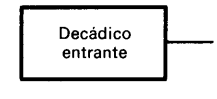
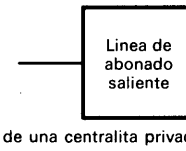
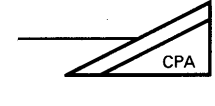
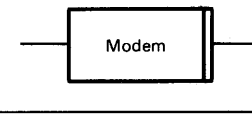
En esta sección se han definido ciertos convenios e interpretaciones con respecto a las ampliaciones del *LED/GR* basadas en estados, a saber:

- Interpretación especial que debe darse a los *diagramas de proceso* conformes a la denominada *REGLA CARDINAL* (véase § 6.2, regla 4).
- El posicionamiento único de los *EP*s (o *EP*s y texto calificador) dentro de un *pictograma de estado*, lo cual es necesario cuando se utilizan *EP*s (véase § 6.2, regla 5).
- La interpretación especial requerida para las *variables* representadas por *EP externos* y *texto calificador* externo como *variables* procuradoras de otras *variables* asociadas con otros *procesos*.

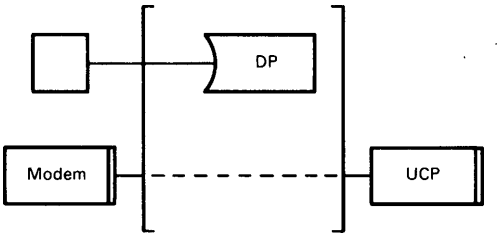
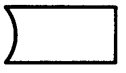
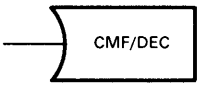
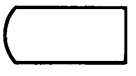
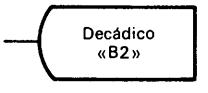
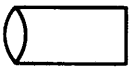
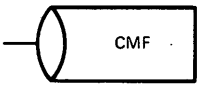
ANEXO A

(a la Recomendación Z.103)

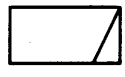
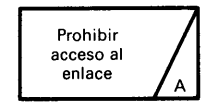
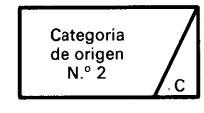
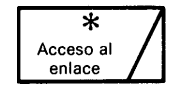
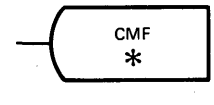
Ejemplos de utilización del conjunto básico de elementos pictográficos

N.°	Elemento pictográfico	Comentarios	Ejemplos
1.	*Frontera de bloque funcional* (BF)  	Para distinguir elementos dentro y fuera de la frontera del BF. Sólo los estados de elementos que están dentro de la frontera pueden ser cambiados directamente por este proceso.	1.1 Un microteléfono fuera de la frontera del BF conectado con un receptor de cifras dentro de la frontera del BF.  1.2 Un enlace fuera de la frontera del BF conectado a través de una unidad de conmutación de dos etapas, con un cuadro de conmutación (manual) fuera de la frontera del BF. 
2.	*Equipo terminal* a) Aparato telefónico colgado  descolgado  b) Enlace  c) Linea de abonado [excepto a)]  d) Cuadro de conmutación (manual)  e) Otro elemento 	Puede ser útil mostrar el equipo terminal (por ejemplo, aparato telefónico y cuadros de conmutación) fuera de la frontera BF, a fin de facilitar la comprensión del tratamiento.	2.1 A colgado $\bar{h}$  2.2 B descolgado h_B  2.3 Circuito de enlace decádico entrante (desde una central con conmutación espacial)  2.4 Linea de abonado saliente hacia un usuario  2.5 Cuadro de conmutación de una centralita privada de abonado  2.6 Modem 

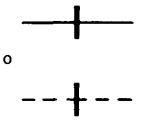
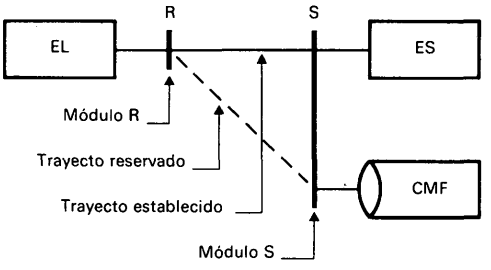
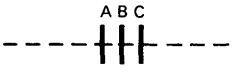
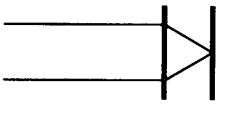
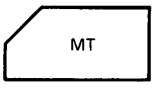
CCITT-20880

N.º	Elemento pictográfico	Comentarios	Ejemplos
3.	*Trayecto de conmutación* a) conectado ——— b) reservado - - - - - -	Para mostrar la conectividad entre equipos terminales y/o dispositivos de señalización que intervienen en el proceso.	3.1 Línea de abonado conectada a un receptor de impulsos de cifras de marcación (DP) y a un modem con un trayecto reservado hacia la unidad central de proceso (UCP) 
4.	*Receptor de señalización* 	Para especificar un proceso de recepción de señales e indicar la naturaleza de las señales recibidas, especialmente las que atraviesan la frontera de bloque funcional.	4.1 Receptor de señalización de código multifrecuencia (CMF)  4.2 Receptor de señalización CMF/decádica 
5.	*Emisor de señalización* 	Para especificar un proceso de emisión de señales e indicar la naturaleza de las señales emitidas, especialmente las que deban atravesar la frontera de bloque funcional.	5.1 Emisor de señalización decádica que está enviando una señal hacia atrás «B2» 
6.	*Emisor y receptor de señalización combinados* 	Para combinar adecuadamente las funciones de los emisores y receptores de señalización.	6.1 Emisor-receptor de CMF 
7.	*Temporizador para la supervisión de un proceso* 	Los temporizadores influyen en los estados subsiguientes del proceso. *Observación* – El simbolo de entrada conexo que indica que ha transcurrido el periodo de temporización puede representarse por $\bar{t}_i$.	7.1 El temporizador t_3 está funcionando  7.2 El temporizador genérico t_s está funcionando  donde s = 1, 2, ... n definen diferentes tonos de servicio

CCITT-20890

N.°	Elemento pictográfico	Comentarios	Ejemplos
8.	*Tasación en curso (y abonado al que corresponde la tasación)* 	La política de tasación es importante para la Administración, el fabricante y el abonado.	8.1 Tasación en curso del abonado A 
9.	*Categoría del abonado o terminal (e información sobre su identidad)* 	Los cambios en la categoría del abonado o del terminal, para cada abonado en una llamada pluripartita pueden influir en el proceso.	9.1 El abonado A tiene prohibido el acceso al enlace  9.2 El usuario C tiene la categoría de origen N.° 2. 
10.	*Símbolo de incertidumbre* *	Sustituye a una información que deliberadamente no se define y que se indica unívocamente en otros pictogramas de estado. En ciertos casos, utilizando el símbolo de incertidumbre se pueden fundir dos o más estados en uno solo, sin pérdida de información, mejorándose así la inteligibilidad del diagrama.	10.1 Teléfono, o bien colgado, o descolgado  10.2 La categoría de abonado es, o no, «prohibir acceso al enlace» o no, en este estado del proceso  10.3 Se está enviando una señal CMF en este estado 

CCITT-20900

N.°	Elemento pictográfico	Comentarios	Ejemplos
11.	*Módulo de conmutación*  o 	Para indicar qué modulos de conmutación intervienen en el proceso. *Observación* – La línea horizontal es el elemento pictográfico para un trayecto de conmutación, que puede estar establecido o reservado. La línea vertical puede utilizarse para representar o bien un módulo de conmutación completo (cuando no es preciso representar la estructura interna) o, también, una de las etapas de conmutación dentro de un módulo de conmutación.	11.1 Un trayecto conectado a través de un módulo de conmutación RCL = Red de conexión de líneas  11.2 Trayectos establecidos y reservados a través de dos módulos de conmutación  EL – Enlace de llegada ES – Enlace de salida CMF – Código multifrecuencia *Observación* – En este ejemplo, el EL está conectado al ES, pero no lo está el emisor/receptor CMF. 11.3 Un trayecto establecido a través de un módulo de conmutación de tres etapas RSN  11.4 Un trayecto reservado a través de un módulo de conmutación de tres etapas ABC  11.5 Un trayecto establecido a través de una red de concentración 
12.	*Elemento de control (asignado a un proceso)* 	Para indicar qué equipo de control interviene en el proceso (sobre todo, módulos que deben dimensionarse). Este símbolo puede utilizarse para indicar que han sido asignados al proceso determinados elementos de soporte lógico.	12.1 Memoria tampón (MT) de registrador 

CCITT-20910

ANEXO B

(a la Recomendación Z.103)

Criterios para la selección de elementos pictográficos

B.1 *Consideraciones generales*

La elección de símbolos para los EP se ha basado en las siguientes consideraciones y criterios generales de selección, que convendría consultar antes de crear nuevos símbolos EP para aplicaciones más extensas del LED.

B.2 *Lectores típicos*

Cabe esperar que las personas que lean los diagramas LED construidos con elementos pictográficos (EP) serán o no técnicas en los siguientes contextos: comercialización de nuevas facilidades, especificación de nuevas facilidades, desarrollo del soporte físico y del soporte lógico según una especificación, gestión de proyectos, explotación y mantenimiento de centrales, ingeniería de tráfico, cursos de capacitación y adiestramiento en telefonía, etc. Asimismo se espera que los diagramas LED servirán de documentación común:

a) entre Administraciones y fabricantes,

b) entre diferentes departamentos dentro de estas organizaciones,

c) sobre centrales telefónicas, y

d) en manuales de capacitación y libros de texto.

No se espera que los diagramas LED (como tales) sean leídos directamente por máquinas. En cambio, se prevé que la forma LED/PR del LED (incluida información en EP) será leída por máquinas que dibujarán diagramas [véanse los apartados b) y c) del § B.3].

B.3 *Métodos de dibujo típicos*

Se espera que los diagramas LED con elementos pictográficos serán hechos generalmente por personal técnico, incluidos delineantes, ya sea:

a) a mano, con la ayuda de una plantilla, y/o

b) presentando el diagrama electrónicamente en una unidad de presentación visual de gráficos, y/o

c) utilizando un trazador controlado electrónicamente.

B.4 *Métodos de reproducción*

Se prevé que los métodos típicos de reproducción sean:

a) los métodos de línea coloreada o copia al ferroprusiato, como en el dibujo convencional,

b) fotocopia por máquinas fotocopiadoras de oficina, incluida la fotorreducción,

c) fotoimpresión en general.

B.5 *Facilidad de reproducción*

A fin de que los diagramas LED puedan reproducirse convenientemente empleando los métodos de línea coloreada o copia al ferroprusiato, así como mediante fotocopia y fotoimpresión, los símbolos EP deben estar formados por líneas de trazo definido, sin sombreado.

B.6 *Facilidad de dibujo*

Los siguientes criterios reflejan la hipótesis de que la técnica primaria de dibujo será hacer éste a mano utilizando una plantilla, y que las técnicas secundarias serán la presentación visual de un diagrama en una pantalla controlada electrónicamente y el dibujo del diagrama con un estilete electrónico:

a) cada símbolo EP debe ser fácil de dibujar con pluma o con lápiz, ya sea a mano alzada o empleando una plantilla,

b) todos los símbolos EP deben dibujarse con el mismo grosor de línea,

c) los símbolos EP deben crearse mediante la síntesis de líneas y curvas geométricas muy sencillas, a fin de facilitar su generación electrónica.

B.7 *Facilidad de comprensión*

Esta es la consideración más importante, puesto que es característico de la documentación LED que el número de los lectores sea muy superior al de los dibujantes (o autores). Esta exigencia se expresa mediante los siguientes criterios relativos a los símbolos EP:

a) *Idoneidad* – La forma de cada símbolo debe ser idónea para el concepto que el símbolo representa.

b) *Distintividad* – Al elegir un conjunto básico de símbolos debe tenerse cuidado de que cada símbolo pueda distinguirse fácilmente de los demás símbolos del conjunto.

c) *Afinidad* – Las formas de los EP que representan funciones diferentes, pero relacionadas por ejemplo, receptores y emisores, deben estar también relacionados de una manera que salte a la vista.

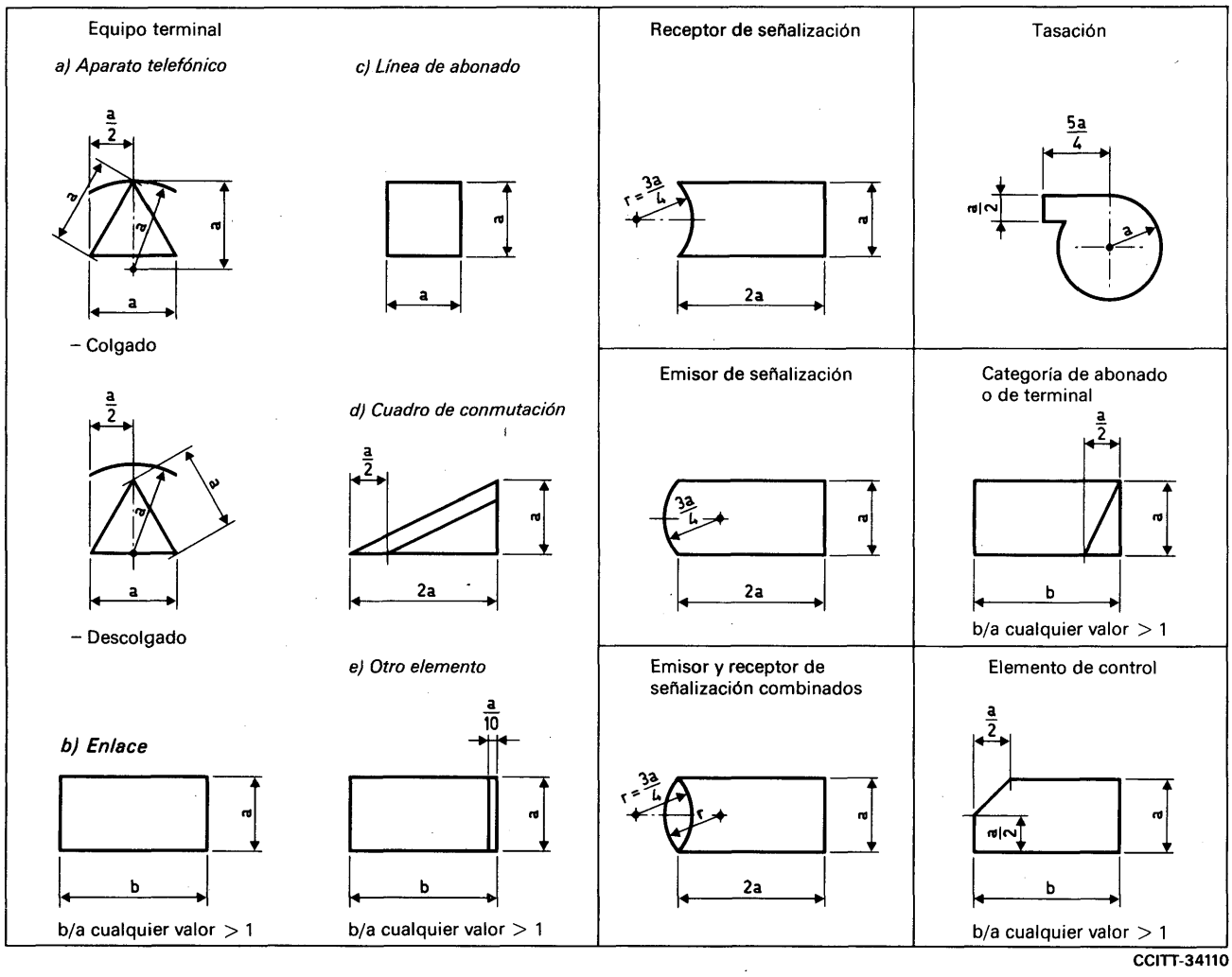
d) *Asociación de texto abreviado con símbolos* – En algunos casos cabe esperar que se asociará un texto abreviado con un EP para indicar la clase de EP; por ejemplo, las letras «CMF» asociadas con un símbolo de receptor indican que deben recibirse señales con código multifrecuencia. En estos casos, los EP deben tener un espacio cerrado para permitir el uso de un número muy pequeño de caracteres alfanuméricos.

e) *Conjunto limitado* – El número total de símbolos debe ser mínimo, para facilitar el aprendizaje del método pictográfico.

ANEXO C

(a la Recomendación Z.103)

Dimensiones relativas recomendadas para el conjunto básico de elementos pictográficos



CCITT-34110

DATOS EN EL LED

1 Introducción

En esta Recomendación se definen el concepto de *datos* en el LED, la terminología de los *datos* del LED, la utilización de *tipos de datos predefinidos* y la facilidad para definir *nuevos tipos de datos.*

En el LED, los *datos* aparecen principalmente en *definiciones de tipos de datos, expresiones,* y en la aplicación de *operadores, variables, valores, constantes* y *literales.*

Definiciones de tipos de datos

Los *datos* en el LED se refieren principalmente a *tipos de datos.* Un *tipo de datos* define un conjunto de *valores,* un conjunto de *operadores* que se pueden aplicar a estos *valores,* y un conjunto de *axiomas* que definen el comportamiento cuando estos *operadores* se aplican a los *valores.* Los *valores, operadores* y *axiomas* definen colectivamente las *propiedades* de los *tipos de datos.* Estas *propiedades* se definen por medio de *definiciones de tipos de datos.*

El LED permite la definición de cualquier *tipo de datos* necesario, incluidos los mecanismos de *estructuración* (tipos compuestos), con la única condición de que se especifique formalmente esa definición. A diferencia de los lenguajes de programación, consideraciones relativas a la aplicación requieren que el conjunto de *tipos de datos* disponibles y, en particular, los mecanismos de estructuración previstos (*matriz, estructura,* etc.) sean limitados.

Expresiones y operadores

Las *expresiones* permiten la manipulación de *valores* (aplicando *operadores* adecuados) para obtener nuevos *valores.*

Variables

Las *variables* son objetos que pueden asociarse a un *valor* por *asignación explícita* o *implícita.* Cuando se *accede* a la *variable,* se devuelve el *valor.*

Valores, constantes y literales

Todos los *tipos de datos* tienen por lo menos un *valor.* Para la mayoría de *tipos de datos* hay formas *literales* (sintácticas) que denotan los *valores constantes* del *tipo de datos* (por ejemplo, los *enteros*). De los *tipos de datos* para los cuales hay *literales* que denotan *valores* se dice que tienen *constantes denotables.* Puede haber más de un *literal* para denotar el mismo *valor constante* (por ejemplo, 12 y H'C denotan el mismo *entero* constante) y se puede utilizar la misma denotación literal para más de un tipo de datos. Algunos tipos no tienen constantes denotables. Por ejemplo, los valores de tipo pilas sólo pueden ser generados por *operadores* que dan *valores* de tipo pilas.

En un lenguaje de especificación, es esencial que puedan describirse formalmente *tipos de datos* en términos de su comportamiento, más bien que componiéndolos a partir de primitivos proporcionados, como en el lenguaje de programación. Este último método entraña invariablemente una aplicación particular del *tipo de datos* y, por tanto, restringe la libertad del aplicador para elegir representaciones apropiadas del *tipo de datos.* El método LED permite cualquier realización, siempre que sea viable y correcta con respecto al LED.

En el LED, todos los *tipos de datos* son *tipos de datos abstractos.* En el § 5 se dan algunos ejemplos de *tipos,* en los que se definen los *tipos de datos predefinidos* del lenguaje.

Aunque todos los *tipos de datos* son abstractos y los *tipos de datos predefinidos* pueden ser redefinidos por el usuario, se ha procurado proporcionar en el LED *tipos de datos predefinidos* que sean familiares en su comportamiento y en su sintaxis. Son los siguientes:

Matriz (Array); Booleano (Boolean); Carácter (Character); Cadena de caracteres (Charstring); Duración (Duration); Entero (Integer); Natural (Natural); Conjuntista (Powerset); PId (PId); Real (Real); Cadena (String); Tiempo (Time); y Temporizador (Timer).

Se puede formar tipos compuestos utilizando *tipos de datos estructurados* (Struct).

1.1 *Formalismo de datos en el LED*

En el LED, los *datos* se modelan mediante un *álgebra de tipos*. Un *álgebra de tipos* es un conjunto de *dominios*, un *dominio designado* y un conjunto de funciones que mapean entre los *dominios*. Cada *dominio* es la colección de todos los *valores* posibles para un *tipo de datos*. El *dominio designado* es el *tipo de datos* que se está describiendo. Las funciones representan las *operaciones* del *tipo de datos*.

El *dominio* y las *operaciones*, junto con el comportamiento (especificado por los *axiomas*) del *tipo de datos*, forman las *propiedades* del *tipo de datos*.

La introducción de un *sintipo* (Syntype) crea un subconjunto de los *valores* de un *tipo* ya definido. La introducción de un *neotipo* crea un *nuevo tipo de datos* distinto, con *propiedades heredadas* del *ascendiente*, pero con *identificadores* diferentes para esas *propiedades*. En la sintaxis concreta, no es necesario que estos *nombres* sean distintos, y esta ambigüedad tiene que resolverse por el contexto.

Un *tipo de generador* es una *descripción de tipo* incompleta; antes de que asuma el estado de un *tipo de datos*, tiene que ser *instanciado* proporcionando esta información que falta.

Las funciones que mapean entre los dominios de un tipo de datos se dividen nominalmente en dos clases. Las funciones de generador, que mapean en el *dominio designado* y producen *valores* (posiblemente nuevos) del *tipo de datos*. Todas las demás funciones son funciones semánticas y atribuyen significado al *tipo* mapeado en otros *tipos* definidos. Las funciones semánticas incluyen los predicados que mapean en el *dominio booleano*.

2 Modelo de lenguaje común

2.1 *Consideraciones generales*

En la Recomendación Z.104, se utilizan los conceptos de *variables*, *tipos de datos predefinidos*, *valores*, *expresiones* y *expresiones* informales. Esta Recomendación define rigurosamente *variables*, *tipos de datos predefinidos*, *valores* y *expresiones*, así como ampliaciones de las Recomendaciones Z.101, Z.102 y Z.103 que permiten la introducción de nuevos *tipos de datos*.

2.2 *Sintaxis abstracta*

Esta sintaxis abstracta amplía la definida por las Recomendaciones Z.101, Z.102 y Z.103.

Definición de datos

Una *definición de datos* es una *definición de tipo de datos* o una *definición de sinónimo*.

Definición de sistema

Una *definición de sistema* puede contener *definiciones de datos*.

Definición de bloque

Una *definición de bloque* puede contener *definiciones de datos*.

Definición de bloque de parte interna

Una *definición de bloque de parte interna* puede contener *definiciones de tipos de datos*.

Definición de subestructura de canal

Una *definición de subestructura de canal* puede contener *definiciones de datos*.

Definición de proceso

Una *definición de proceso* puede contener *definiciones de datos* y *definiciones de variables*.

Definición de procedimiento

Una *definición de procedimiento* puede contener *definiciones de datos* y *definiciones de variables*.

Nodo de entrada

Una *variable* mencionada en un *nodo de entrada* tiene que ser definida en una *definición de variable* y ha de tener el mismo *tipo de datos* que el *tipo de datos* correspondiente en la *definición de señal*.

Definición de tipo de datos

Una *definición de tipo de datos* contiene un *nombre de tipo* y una descripción de *neotipo* o una descripción de *sintipo.*

Descripción de neotipo

Una descripción de *neotipo* contiene un conjunto posiblemente vacío de *nombres de valor,* un conjunto de una o más introducciones de *operador* y un conjunto posiblemente vacío de *axiomas de tipo de datos.*

Todos los *nombres de valor* de la *descripción de tipo de datos* tienen que ser únicos dentro del *tipo de datos.*

Todos los *nombres de operador* de la *descripción de tipo de datos* tienen que ser mutuamente exclusivos.

Todos los *nombres de tipo de datos* en el mismo contexto tienen que ser únicos.

Introducción de operador

Una introducción de *operador* o bien introduce uno de los *operadores universales* que es permisible introducir con cualquier *tipo de datos,* o introduce un *nombre de operador* junto con la *tipificación de operador.*

Operador

Un *operador* es o bien un *operador universal* con un *identificador de tipo de datos,* o un *identificador de operador* definido por el usuario con un *identificador de tipo de datos.* En cualquiera de estos casos, el *identificador de tipo de datos* permite la *definición de tipo de datos* que define el *operador* que ha de establecerse.

Tipificación de operador

Una *tipificación de operador* contiene la lista de *identificadores de tipo de datos* de los *parámetros* al *operador* y el *identificador de tipo de datos* del *resultado* de aplicar el *operador.*

Al menos una de las *identidades de tipo de datos* de la lista, o el resultado, tiene que ser la del *tipo de datos* que se define.

El *tipo* de resultado puede no ser un *sintipo.*

Axiomas

Los *axiomas* son sentencias de verdad que se mantienen en todas las condiciones para el *tipo* que se define y, de esta forma, especifican el comportamiento de los tipos.

Sentencia de asignación

Una *sentencia de asignación* contiene un *operador de asignación,* una *identidad de variable* y una *expresión.* El *operador de asignación* es o bien el *operador universal* para *asignar cualificado* con la *identidad de tipo de datos* del *tipo de datos de la variable,* o el *operador insertado definido por el usuario.*

Descripción de sintipo

Una descripción de *sintipo* contiene un *nombre de sintipo,* la *identidad de tipo de datos del ascendiente* y el conjunto de *identidades de valor* del *tipo de datos del ascendiente* que son válidas para el *tipo de datos sin.*

Expresión

Una *expresión* es o bien un *primario,* o una *operación.*

Operación

Una *operación* contiene un *operador* y una lista de una o más *expresiones.* Hay tantas *expresiones* contenidas en la *operación* como *tipos de datos* definidos en los *parámetros* del *operador.*

Primario

Un primario es uno de los siguientes:

- una *identidad de sinónimo,*
- una *identidad de valor,*
- una *identidad de variable,* o
- una *expresión condicional.*

Expresión condicional

Una *expresión condicional* es una *expresión booleana* y una lista de dos *expresiones* del mismo *tipo de datos.*

Definición de variable

Una *definición de variable* consta de una lista de *nombres de variable* y un *identificador de tipo de datos.*

Operación universal

Un *operador universal* es o bien un *operador de variable* o un *comparador.*

Un *operador de variable* es *declaración, asignación,* o *acceso. Declaración* se utiliza para *declarar variables*; *asignación* se utiliza para *asignar* a *variables* y *acceso* se utiliza siempre que una *identidad* de *variable* se *interpreta* como un *valor.*

Un *comparador* es o bien uno de los *operadores de orden* o un *operador de igualdad.* Un *operador de orden* es o bien menor que o mayor que.

Todos los *operadores universales* incluyen el *tipo de datos* al que corresponden como un *calificador,* de modo que distintos *tipos de datos* diferentes *operadores.* Por ejemplo, los dos *tipos de datos* cuadrado y cubo introducen dos *identidades de operador* para *asignar,* a saber, asignar! cuadrado, y asignar! cubo.

Para un *tipo de datos* D, la *tipificación de operador* para comparadores es:

D, D– > Booleano

Para un *tipo de datos* D, el *operador de asignación* requiere una *identidad de variable* de *tipo de datos* D y un *valor* de *tipo de datos* D.

Para un *tipo de datos* D, el *operador de acceso* requiere un *identificador de variable* de *tipo de datos* D y entrega un *valor* de *tipo de datos* D.

Para un *tipo de datos* D, el *operador de declaración* requiere un *identificador de variable.*

Definición de sinónimo

Una definición de *sinónimo* contiene un *nombre de sinónimo* y una *expresión constante.*

Expresión constante

Una *expresión constante* es o bien un *valor constante,* o una *operación* cuyos parámetros son todos *expresiones constantes.*

Valor constante

Valor constante es o bien un *identificador de valor* o un *identificador de sinónimo.* Un *sinónimo* no puede ser definido reiterativamente.

2.3 *Reglas de interpretación*

2.3.1 *Proceso*

La *instanciación* de un *proceso* se produce antes de que se *interprete* el *nodo de arranque* y hace que una *operación de declaración* se aplique a cada *nombre de variable* que aparezca en una *definición de variable* en la *definición de proceso.* Como resultado del *operador de declaración,* las *variables* declaradas se convierten en asociaciones de las *identidades de variable* a un *valor inicial* (que será el *valor indefinido* si no se indica otra cosa en los *axiomas* para los *tipos de datos* de las *variables*). Las *identidades de variable* declaradas al instanciar un *proceso* contienen el *nombre de variable,* la *identidad de instancia de proceso* y la *identidad de tipo de datos* de la *definición de variable.*

2.3.2 *Nodo de arranque de procedimiento*

Llamar un *procedimiento* causa *variables* definidas dentro del *procedimiento* que ha de *crearse* para un *procedimiento* en modo semejante a *instanciar* un *proceso.*

2.3.3 *Gráfico de proceso*

La *interpretación* de un *nodo de salida* hace que cada una de las *expresiones* del *nodo de salida* se *interpreten* en la secuencia especificada y los valores resultantes se *asignen* a las *variables implícitas anónimas* que están asociadas con la *señal.* Estas *variables* se considera que están *declaradas* cuando se *interpreta* el *nodo de salida.* Cada una de estas *variables* tiene el *tipo de datos* o el *tipo de datos sin* asociado con la posición correspondiente en las *definiciones de señal.* Los *valores* que se *asignan* a estas *variables* son los *valores* transmitidos por la *señal.*

La *interpretación* de un *nodo de decisión* hace que se *interprete* la *expresión* contenida en el *nodo de decisión* seguida de la elección del *arco* que está asociado con el *valor* entregado por la *expresión.*

Petición de crear

La *interpretación* de un *nodo de petición de crear* hace que cada una de las *expresiones* en el *nodo de petición de crear* se *interprete* en la secuencia especificada.

La *instanciación* del *proceso* que se *crea* se produce después junto con la *declaración* del *parámetro formal* del *proceso* y la *asignación* del *valor* resultante correspondiente de cada *expresión* en el *nodo de petición de crear* a cada *parámetro formal.* El *proceso creado* se ejecuta después por separado, pero al mismo tiempo que otros *procesos.*

Nodo de llamada

Un *nodo de llamada* hace que las *expresiones* utilizadas como *parámetros formales* atribuidos con *DENTRO* se *interpreten* antes que la *interpretación* del *nodo de arranque* del *procedimiento.* Cada una de las *expresiones asigna* su valor al *parámetro actual* correspondiente.

2.3.4 *Procedimiento*

Un *parámetro formal* atribuido con *DENTRO/FUERA* se *interpreta* como el *identificador de variable* del *parámetro actual* correspondiente en el contexto de la *llamada de procedimiento.* Un *parámetro formal* atribuido con *SEÑAL* se *interpreta* como el *identificador de señal* del *parámetro actual* correspondiente en el contexto de la *llamada de procedimiento.*

2.3.5 *Sentencia de asignación*

La *sentencia de asignación* se *interpreta* como combinación del *valor* antiguo de la *variable* con el *valor* de una *expresión* y como vinculación de la *identidad de variable* con este nuevo *valor.*

El *operador de asignación* determina las reglas para combinar el antiguo *valor* de la *variable* con el *valor* de la *expresión.* Estas reglas se determinan mediante la utilización del *operador de asignación* en *axiomas de tipo de datos* del *tipo de datos.* Si el *operador de asignación* es el *operador universal* para *asignación*, la *variable* está *vinculada* al *valor* de la *expresión.*

El *valor* de la *expresión* tiene que ser uno de los *valores* del *tipo de datos* de la *variable* del *operador de asignación.* En el *operador universal* para *asignación*, el *tipo de datos* del *parámetro* es el de la *variable.*

2.3.6 *Expresión y primario*

Una *expresión* se *interpreta* como el *primario* que forma la *expresión.* El *primario* es una *operación*, un *sinónimo*, un *identificador de valor*, o un *identificador de variable*, y así se *interpreta.*

2.3.6.1 *Operación*

Una *operación* se *interpreta* como una aplicación del *operador* a la lista de *valores* obtenidos *interpretando* la lista de *expresiones.* La *interpretación* de la *operación* se determina mediante la utilización del *operador* en los *axiomas de tipo de datos* del *tipo de datos.*

2.3.6.2 *Identificador de sinónimo*

Un *identificador de sinónimo* se *interpreta* como la *expresión constante* definida en la *definición de sinónimo.* La *expresión constante* se *interpreta* del mismo modo que una *expresión.*

2.3.6.3 *Identificador de valor*

Un *identificador de valor* se *interpreta* como el *valor* que denota.

La semántica del *valor* que denota un *identificador de valor* se determina mediante la utilización del *identificador de valor* en los *axiomas de tipo de datos* del *tipo de datos.*

2.3.6.4 *Identificador de variable*

Un *identificador de variable* se *interpreta* de uno de dos modos, según el contexto. Dentro de una *expresión*, un *identificador de variable* se *interpreta* como un *acceso.* En el contexto de una *sentencia de asignación*, un *identificador de variable* se *interpreta* como una *variable*, que es el *vínculo* del *identificador de variable* con el *valor.* El *acceso* a una *variable* se *interpreta* como el *valor* al que la *variable* está *vinculada*, salvo que el *acceso* al *valor indefinido* se *interpreta* como un *error.*

2.3.6.5 *Expresión condicional*

Una *expresión condicional* se *interpreta* como la primera o segunda *expresión* de la lista, según que la interpretación de la *expresión booleana* produzca *verdadero* o *falso.*

2.3.7 *Definiciones de tipo de datos*

Estas definiciones no se *interpretan.*

3 LED/GR

La especificación normal del LED del comportamiento de *tareas, decisiones,* etc. (es decir, la estructura interna de esos *nodos*) es la forma LED/PR. Así pues, no hay *sintaxis gráfica* específica para datos.

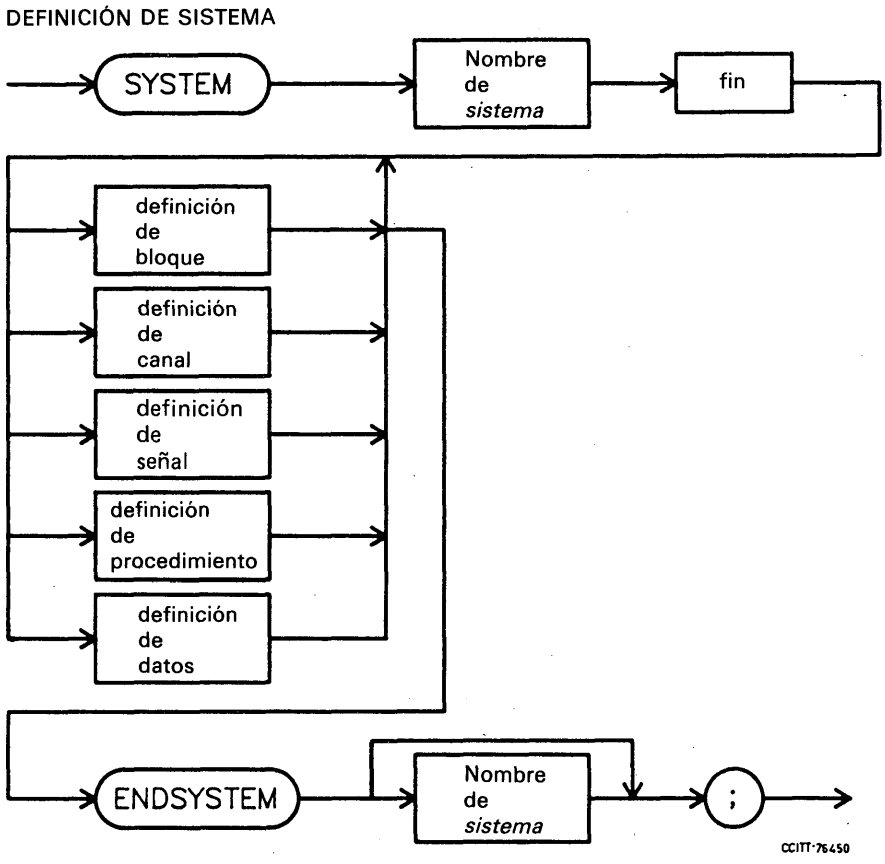
Cuando se incluyen las *definiciones de datos* (para *tipos de datos, variables* o *sinónimos*), deben definirse o referenciarse en la parte superior del diagrama en que están incluidas.

4 LED/PR

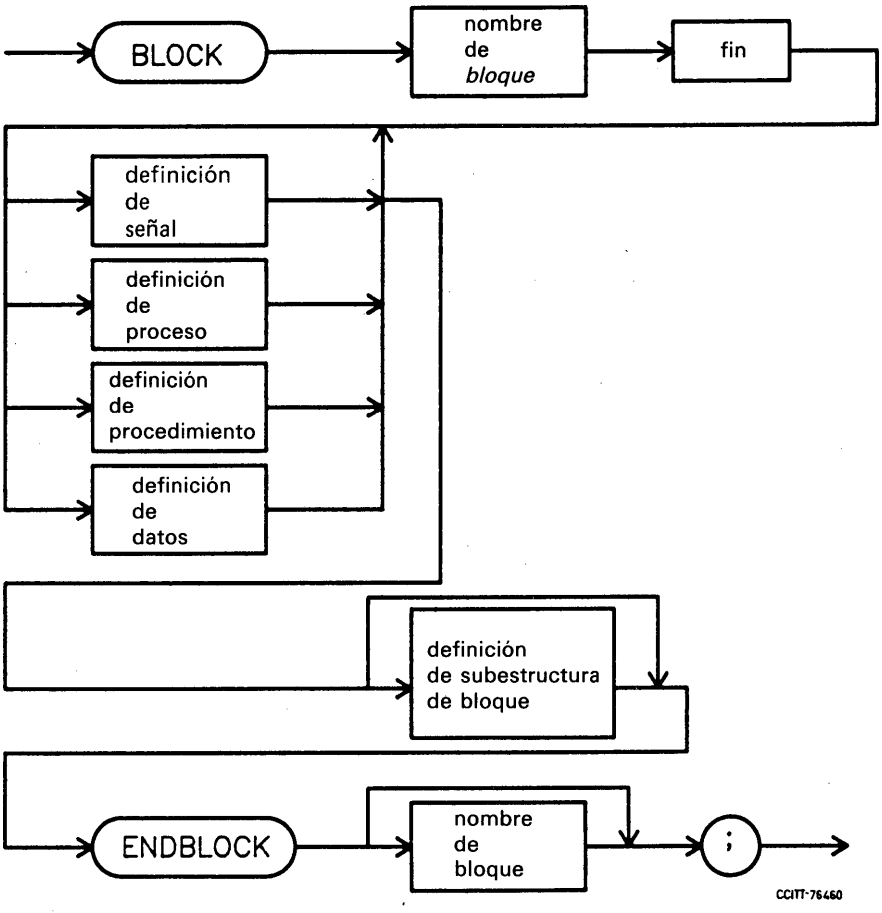
4.1 *Adición a la sintaxis*

Las definiciones de datos se añaden a la sintanxis como se define en las Recomendaciones Z.101, Z.102 y Z.103.

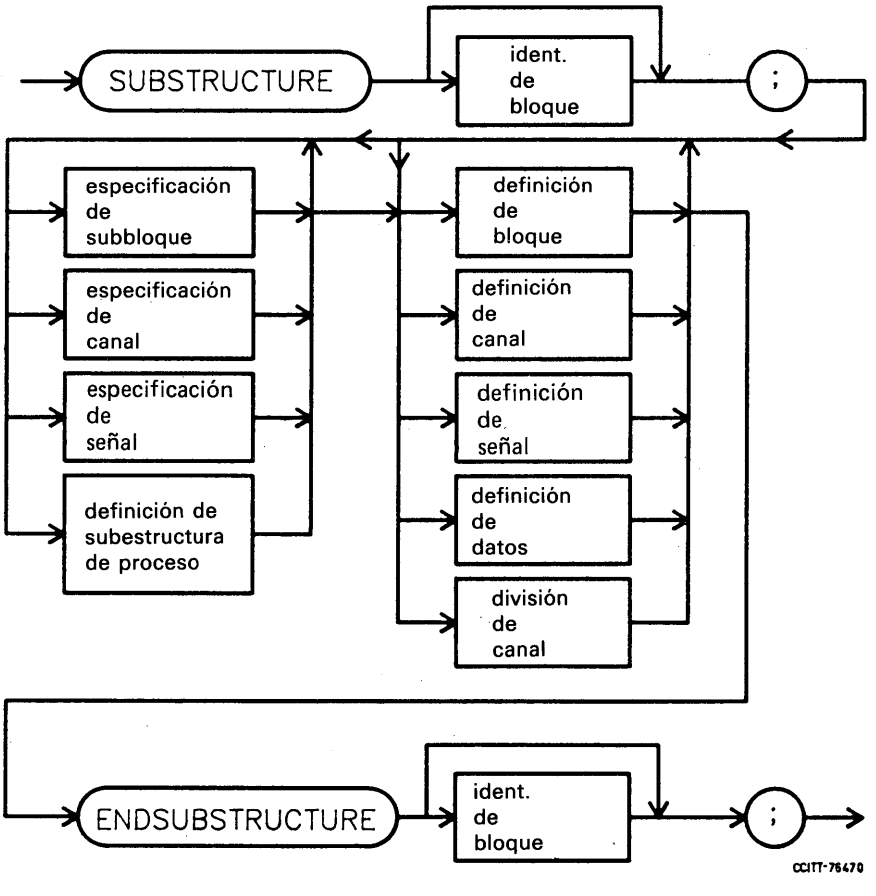
Las *definiciones de datos* pueden aparecer cuando una *definición de señal* aparece en una *definición de sistema,* una *definición de bloque,* una *definición de bloque de parte interna* o una *definición de subestructura de canal.* Una *definición de datos* puede aparecer también cuando una *definición de variable* puede aparecer en una *definición de proceso,* o cuando una *definición de variable de procedimiento* puede aparecer en una *definición de procedimiento.*



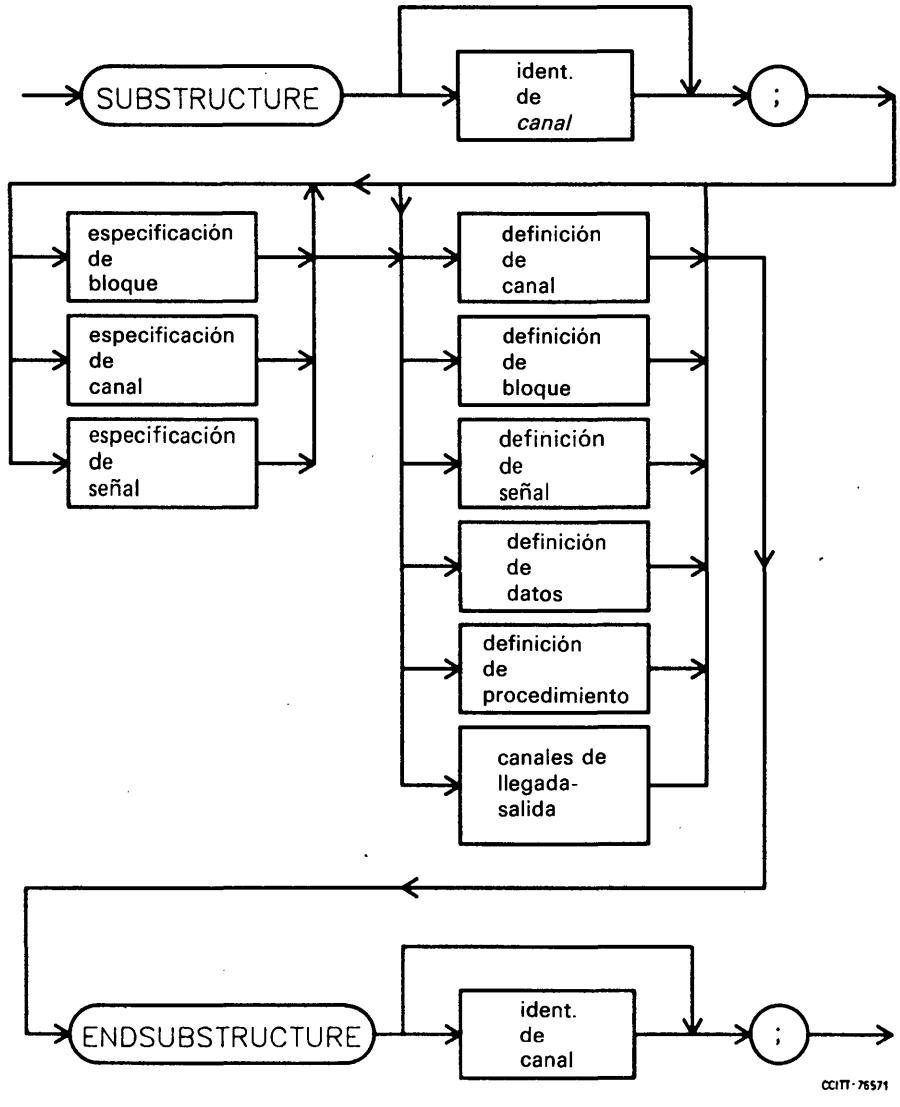
DEFINICIÓN DE BLOQUE

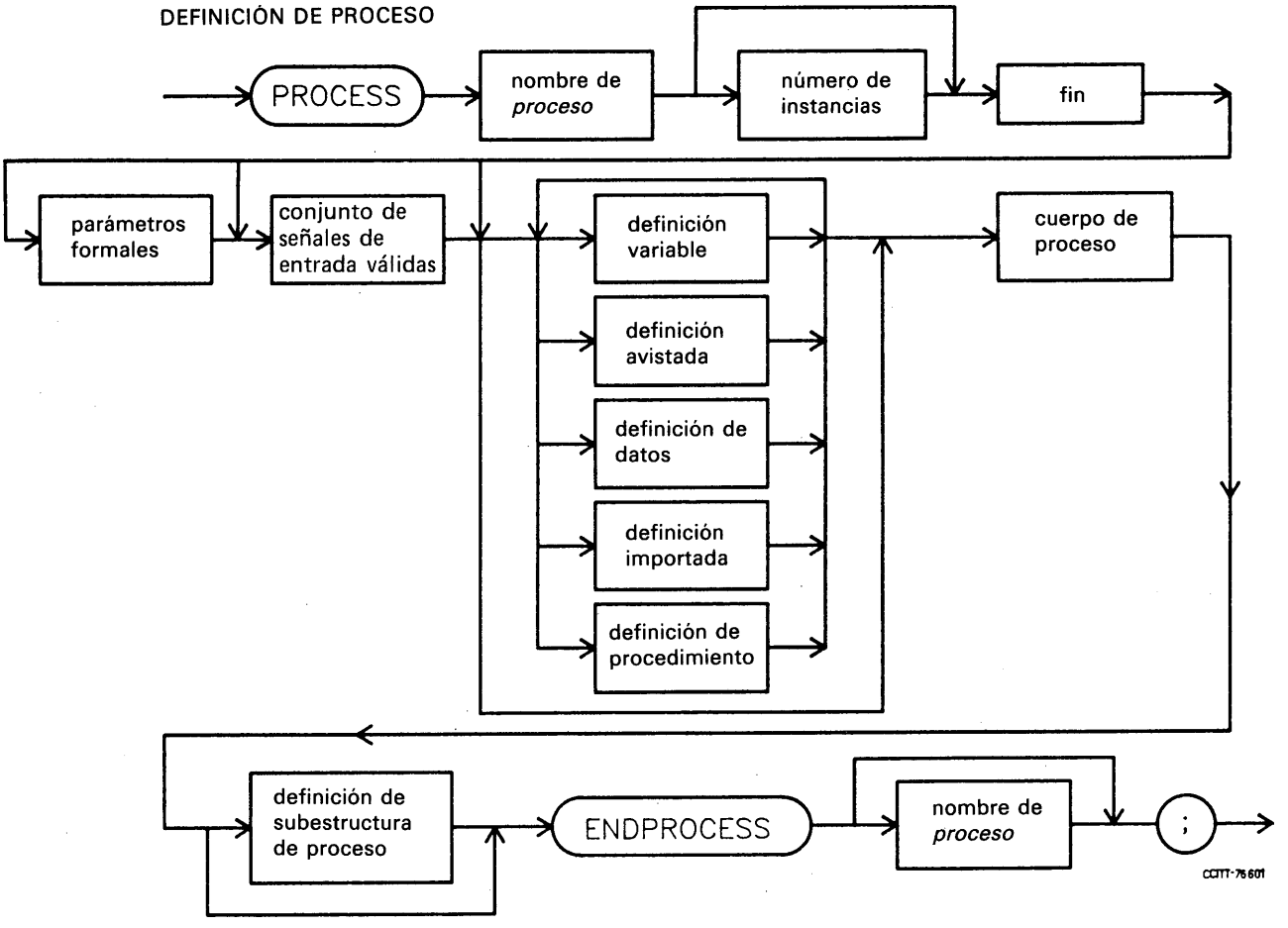


DEFINICIÓN DE SUBESTRUCTURA DE BLOQUE



DEFINICIÓN DE SUBESTRUCTURA DE CANAL



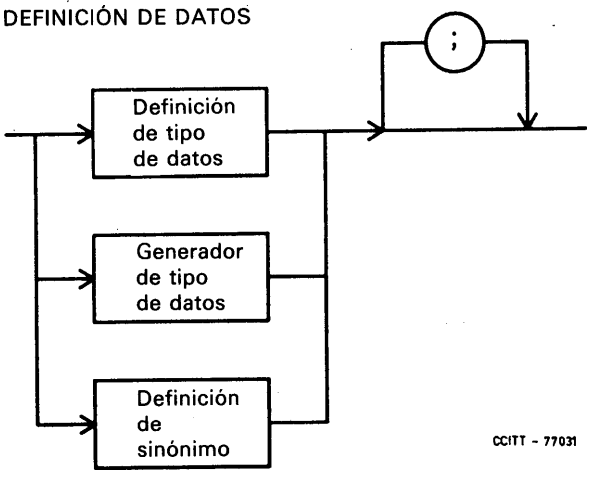


La sintaxis de *tipo de datos* se amplía para incluir *identificadores de tipo de datos* definidos por el usuario.

La sintaxis para *sentencias de asignación* y *expresiones* viene dada. (Obsérvese que las expresiones y sentencias de asignación se mencionan en la Recomendación Z.101, pero no se definen).

4.2 *Definición de datos*

4.2.1 *Sintaxis*



4.2.2 *Semántica*

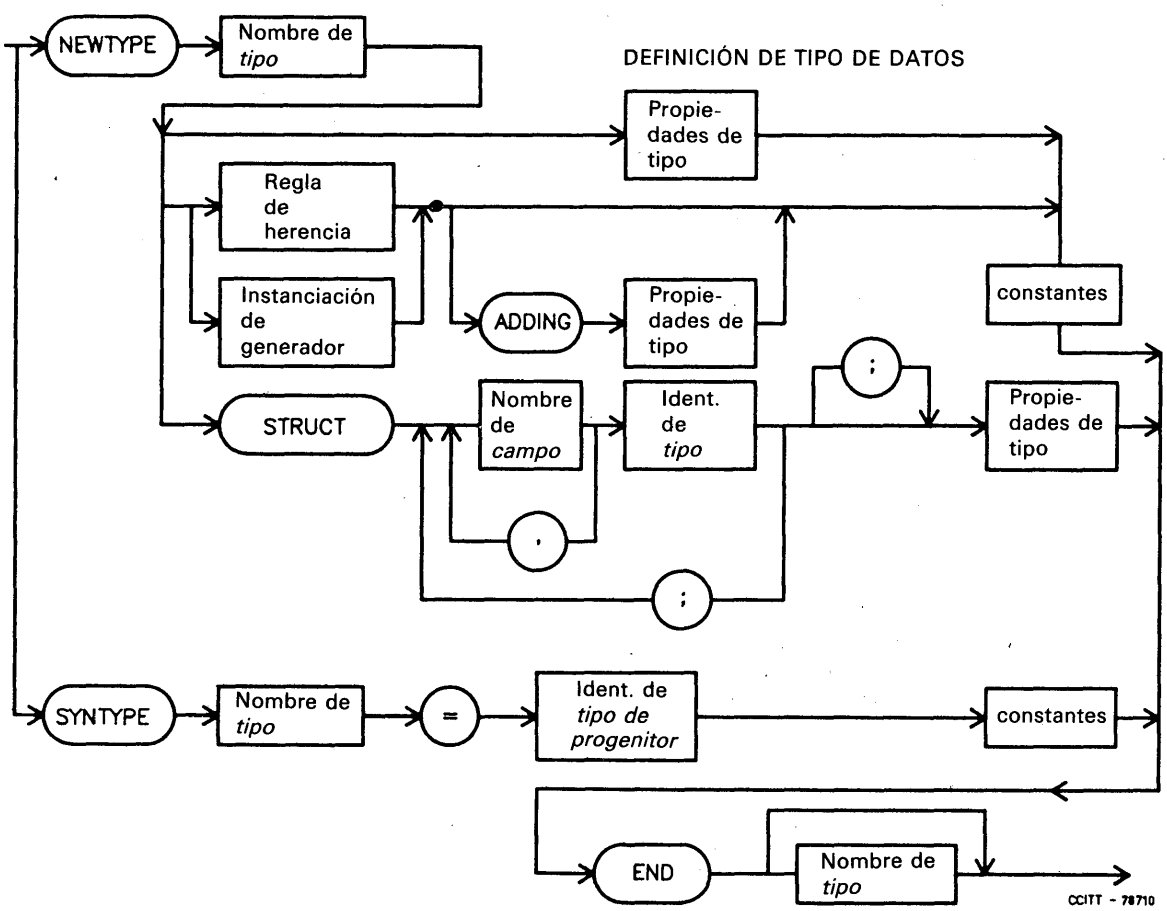
Una *definición de datos* se utiliza para introducir los *nombres* y *propiedades* de *tipos de datos, generadores de tipo de datos* o *sinónimos.*

4.2.3 *Relación con la sintaxis abstracta del LED*

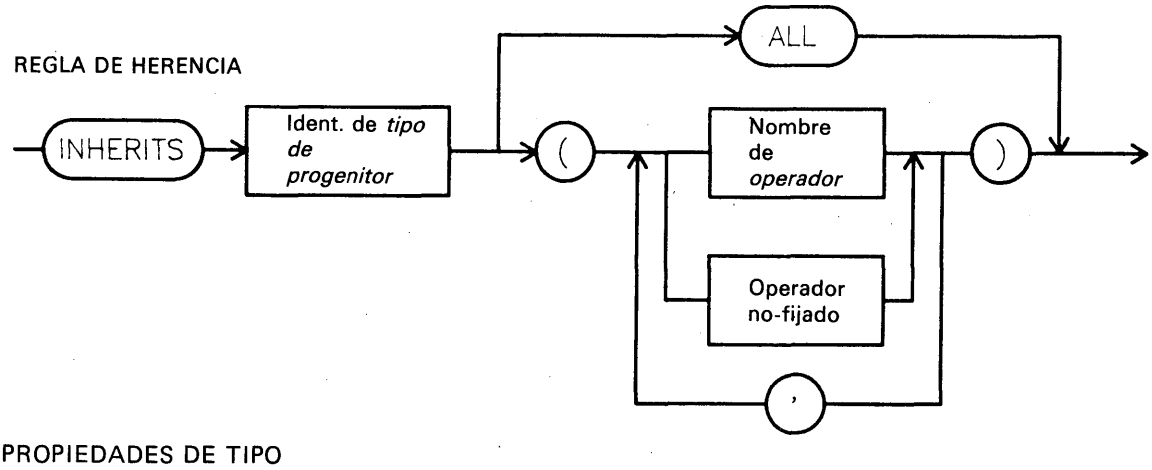
Una *definición de datos* en el LED/PR representa una *definición de datos* en la *sintaxis abstracta.* Si la *definición de datos* es un *generador de tipo de datos,* no hay correspondencia directa con la *sintaxis abstracta,* ya que el *generador de tipo de datos* sirve sólo para definir texto que se considera ampliado textualmente en la *instanciación de generador.*

4.3 *Definición de tipo de datos*

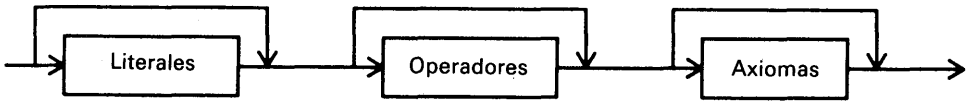
4.3.1 *Sintaxis*



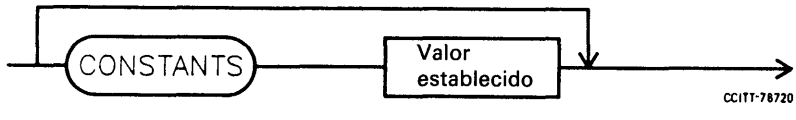
REGLA DE HERENCIA



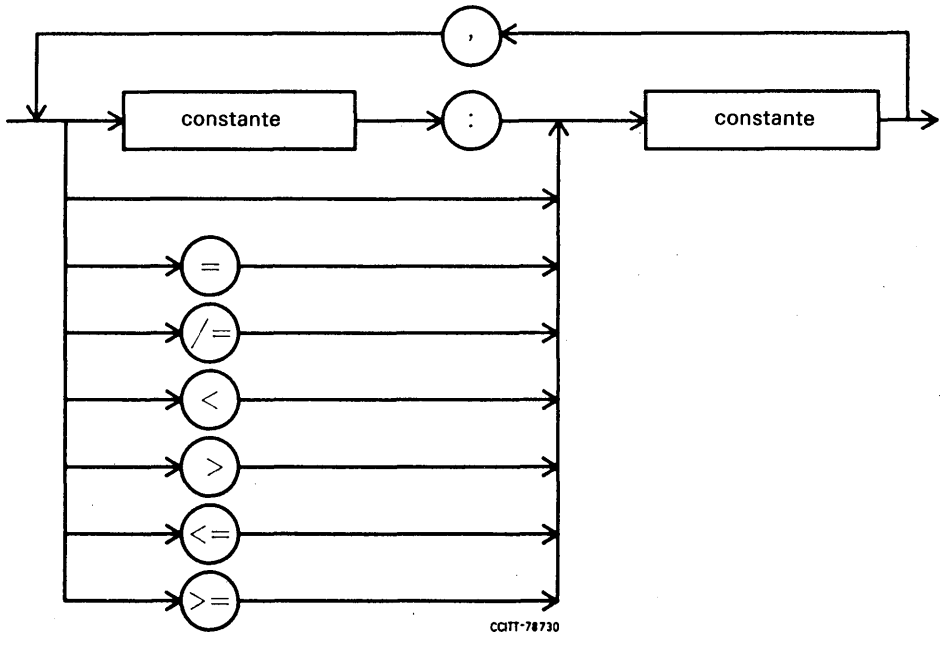
PROPIEDADES DE TIPO

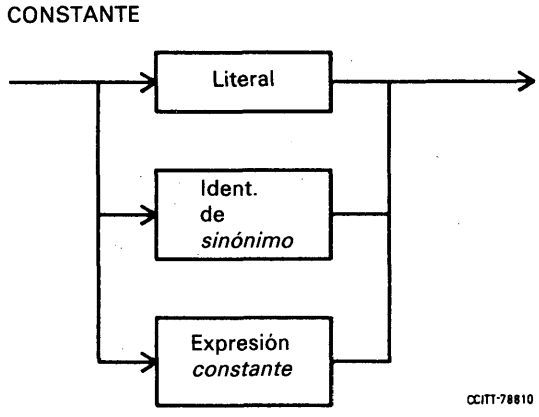


CONSTANTES



VALOR ESTABLECIDO





4.3.2 *Semántica*

El nombre dado en una *definición de tipo de datos* es un *nombre de tipo de datos.*

4.3.2.1 *NEOTIPO*

Una *definición de tipo de datos NEOTIPO* introduce un *tipo de datos nuevo.*

Si no se especifica ninguna *regla de herencia,* el *tipo de datos nuevo* no se basa en ningún otro *tipo.* Las *propiedades del tipo* se utilizan para introducir cualquier *literal* para ese *tipo,* los *operadores* aplicables al *tipo* y (facultativamente) las *propiedades* del *tipo* estableciendo *axiomas* que son verdaderos para ese *tipo.*

Un *nuevo tipo* puede basarse en otro *tipo* utilizando *NEOTIPO* en combinación con *reglas de herencia.* En este caso, el *conjunto de valores* del *nuevo tipo de datos* se separa del *conjunto de valores* del *tipo ascendiente.* Aunque los *valores* y *operadores* del *nuevo tipo de datos* son distintos de los del tipo de datos ascendiente, los *literales* y *nombres* de los *operadores* para el *nuevo tipo* estarán *sobrecargados,* es decir, serán los mismos *literales* y *nombres* que los del *tipo ascendiente* y habrá que decidir si un *literal* o *nombre* es apropiado para el *nuevo tipo* o *tipo de datos ascendiente,* ya sea por *cualificación* o por el *contexto.* Si la *vinculación* de un *literal* o *nombre de operador* a un *tipo* no puede determinarse, las especificaciones del LED son *ambiguas* y, por tanto, inválidas. Cuando *TODO* (ALL) viene dado para una *regla de herencia,* todos los *nombres de operador* son *sobrecargados* para el *nuevo tipo de datos.* En caso contrario, los *nombres de operador* especificados en la *regla de herencia* deben ser *nombres de operador* del *tipo ascendiente* y estos *nombres* se definen para el *nuevo tipo.* El *conjunto de axiomas* y el *conjunto de literales* del *tipo de datos ascendiente* se heredan.

Así como los *literales, nombres de operador* y *axiomas heredados,* un *nuevo tipo de datos* pueden tener *literales, operadores* y *axiomas* adicionales especificados como *propiedades de tipo* después de la palabra clave *AÑADIR* (ADDING). Estos *literales, nombres de operador* y *axiomas* no deben estar en contradicción con los *heredados.*

Una *definición de datos* de la forma:

```
NEOTIPO X/* detalles /*
    CONSTANTES/* lista de constantes */
    FIN X;
```

es equivalente a

```
NEOTIPO Anon/* detalles /*
    FIN Anon;
```

seguido de

```
SINTIPO X = Anon
    CONSTANTES /* lista de constantes */
    FIN X;
```

La utilización de una *restricción* de *constantes* en un *NEOTIPO implícitamente declara* un *NEOTIPO anónimo* (Anon anterior) sin esa restricción, que se utiliza después como el *ascendiente* de un *SINTIPO* con la *restricción* de *constantes.* Para obligar al *anonimato,* el *nombre ascendiente* se declara como distinto de todos los demás *nombres* denotados en la especificación LED fuera de la *declaración implícita* particular.

4.3.2.2 *SINTIPO*

Un *tipo de datos* puede también definirse de modo que tenga un subconjunto de los *valores* del *tipo de datos ascendiente* utilizando *SINTIPO*. En este caso el *conjunto de valores* se especifica después de la *palabra clave CONSTANTES*, o todos los *valores* del *tipo de datos ascendiente* tienen *valores* correspondientes en el *SINTIPO*. Las *variables declaradas* con un *SINTIPO* sólo pueden *asignarse* a los *valores* especificados.

El *acceso* a una *variable* con un *SINTIPO* produce un *valor* del *tipo de datos ascendiente*. Estas *operaciones* de *declaración*, *asignación* y *acceso* son las únicas *operaciones* permitidas para *SINTIPO*.

Todos los *valores* especificados para un *tipo de datos SINTIPO* tienen que ser *valores* del *tipo de datos ascendiente*. El *ascendiente* de un *SINTIPO* es el segundo *tipo* nombrado en la definición del *SINTIPO*, siempre que este *tipo* sea un *NEOTIPO*. En caso contrario, el *ascendiente* es el *ascendiente* del *tipo* nombrado.

4.3.2.3 *Instanciación de generador*

Una *instanciación de generador* es equivalente al texto del *generador* con los *parámetros formales* textualmente sustituidos por los *parámetros actuales*. Cada vez que el *nombre de generador* se utiliza en el texto del *generador*, se sustituye por el *nombre* del *tipo de datos* o *generador* que llama a la *instanciación de generador*. El texto equivalente tiene que completar una *definición* válida del *tipo de datos NEOTIPO*. Esta *definición de tipo de datos* formada por la ampliación textual define las *propiedades* del *nombre de tipo de datos*.

4.3.2.4 *ESTRUCT*

La *definición de tipo de datos* que incluye un *ESTRUCT* implica *tipos de datos* para cada *nombre de campo*. Para un *tipo de estructura* determinado S, para cada *nombre de campo* FL y el *identificador de tipo* correspondiente TL, se introducen implícitamente los *axiomas* siguientes (sujetos a *reforzamiento*; véase más adelante):

```
el axioma único        :  extraer!(insertar!(S, Fi, I), Fi) = I
el conjunto de axiomas :  extraer!(insertar!(S, Fi, I), Fj)
                       =  Extraer(S, Fj);
                          /* para todos los Fi, Fj distintos */
```

Cuando hay N *nombres de campo* en una *estructura*, habrá N * N *axiomas introducidos implícitamente* de esta forma. Para garantizar la resolución de la *ambigüedad*, el LED require que los *nombres de campo* dentro de una *estructura* determinada sean únicos.

Con la *estructura* S está asociado un conjunto de *tipos*, uno para cada *campo*, con el *nombre de tipo* S!Fi, el *literal* único Fi, y ninguna otra *propiedad*. Este *tipo* se convierte en el *portador* del *nombre de campo* utilizado en las operaciones *Insertar!* y *Extraer!*.

El efecto de una definición de *Estruct* es crear una estructura o registro (como el lenguaje de programación), aunque la definición puede incluir *axiomas* adicionales para *reforzar* el *comportamiento* del tipo. Cuando los *axiomas* adicionales explícitamente introducidos por el usuario entran en conflicto con el *conjunto* de *axiomas* de subsidiarios implícitos para esa *Estruct*, la incoherencia se resuelve descartando los *axiomas* implícitos. La introducción de *axiomas* explícitos en una *Estruct* require mucho cuidado.

4.3.3 *Relación con la sintaxis abstracta*

Una *definición de tipo de datos* representa una *definición de tipo de datos* en la *sintaxis abstracta*. El *nombre de tipo* representa el *nombre de tipo* en la *sintaxis abstracta*.

4.3.3.1 *NEOTIPO*

Las *palabras clave NEOTIPO* y *FIN* abarcan el concepto de *sintaxis abstracta* de una *descripción de tipo de datos*. El *conjunto de nombres de valor*, *introducciones de operador* y *conjunto de axiomas de tipo* en la *sintaxis abstracta* se representan del modo siguiente:

a) *Conjunto de nombres de valor*

El conjunto de *literales* dado por los *literales* en las *propiedades de tipo* combinado con el conjunto de *literales* para el *tipo de datos ascendiente* si se especifican *HEREDA*.

b) *Conjunto de introducción de operador*

El conjunto de *operadores* dados por los *operadores* en las *propiedades de tipo* combinado con el *conjunto* de *operadores* del *tipo de datos ascendiente* si se especifica *HEREDA*.

c) *Conjunto de axiomas de tipo*

El conjunto de *axiomas* (si los hay) dados por los *axiomas* en las *propiedades de tipo de datos* combinado con el conjunto de *axiomas* del *tipo de datos ascendiente* si se especifica *HEREDA*. Si el *conjunto de axiomas* se omite, o es incompleto, al menos algunas operaciones sólo pueden *interpretarse informalmente*.

4.3.3.2 *SINTIPO*

Las *palabras clave SINTIPO* y *FIN* abarcan el concepto de *sintaxis abstracta* de una *descripción* de *sintipo.* El *identificador de tipo ascendiente* representa el *identificador de tipo de datos ascendiente* en la *sintaxis abstracta.* El *conjunto de valores* representa el *conjunto de valores* en la *sintaxis abstracta.*

4.3.3.3 *Instanciación de generador*

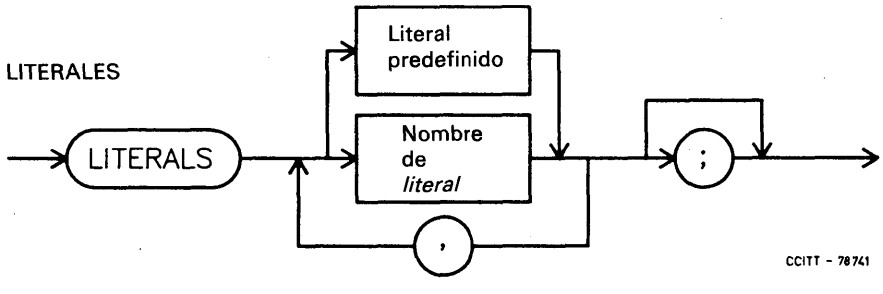
Una *instanciación de generador* denota el texto que se obtendrá ampliando textualmente el *generador* como en el § 4.3.2.3, de modo que tenga la misma relación con la *sintaxis abstracta* que el texto equivalente.

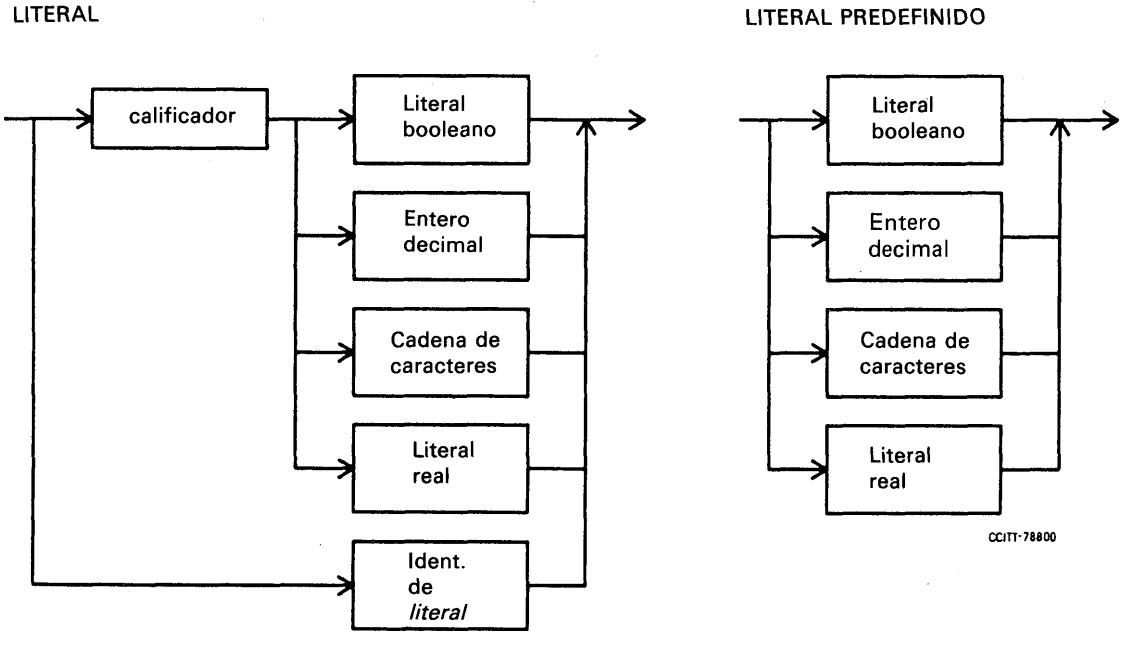
4.3.3.4 *ESTRUCT*

Un *ESTRUCT* denota el texto obtenido nombrando explícitamente todas las *propiedades* pertinentes y, por tanto, (como para un generador) tiene la misma relación con la *sintaxis abstracta* que el texto así denotado.

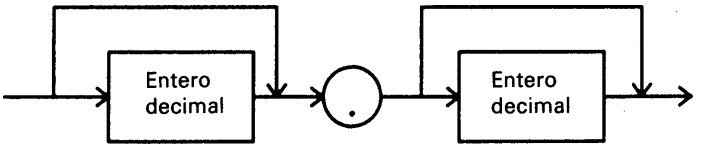
4.4 *LITERALES*

4.4.1 *Sintaxis*

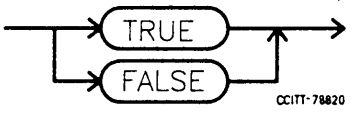




LITERAL REAL



LITERAL BOOLEANO



4.4.2 *Semántica*

Los *literales* que se utilizan para denotar los *valores* de un *tipo de datos* o bien son *predefinidos* (para *tipos de datos predefinidos* o *tipos de datos* basados en *tipos de datos predefinidos*) o se introducen mediante la lista de denotaciones para los *literales* de un *tipo de datos* después de la *palabra clave LITERALES*. Cuando un *tipo* comprende las operaciones *Ordenar*!, los *literales* deben nombrarse convencionalmente en orden ascendente.

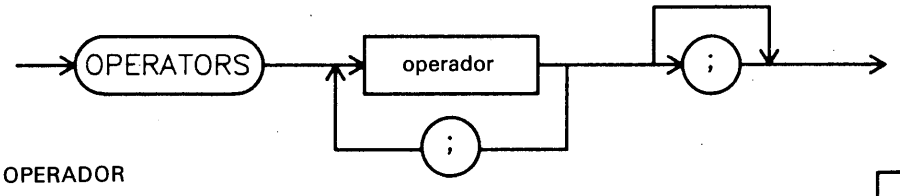
4.4.3 *Relación con la sintaxis abstracta*

Los *nombres de literal* introducidos por la parte *literales* de las *propiedades de tipo* de datos representan los *nombres de valores* de una *descripción* de *tipo de datos* en la *sintaxis abstracta*.

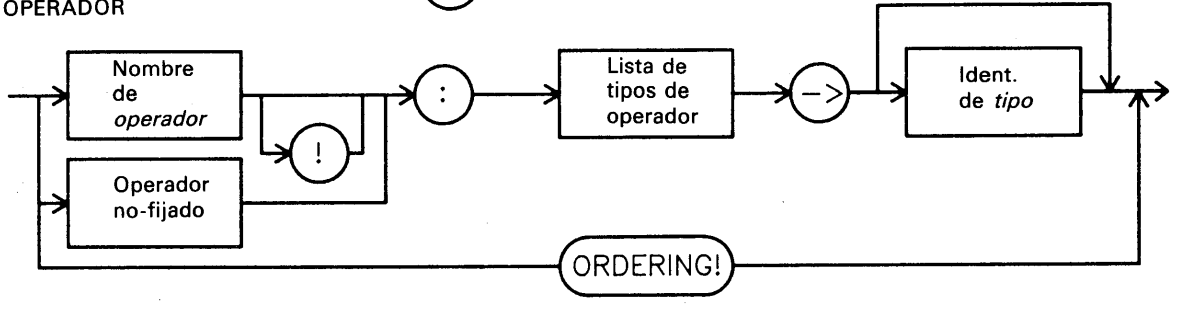
4.5 *Operadores*

4.5.1 *Sintaxis*

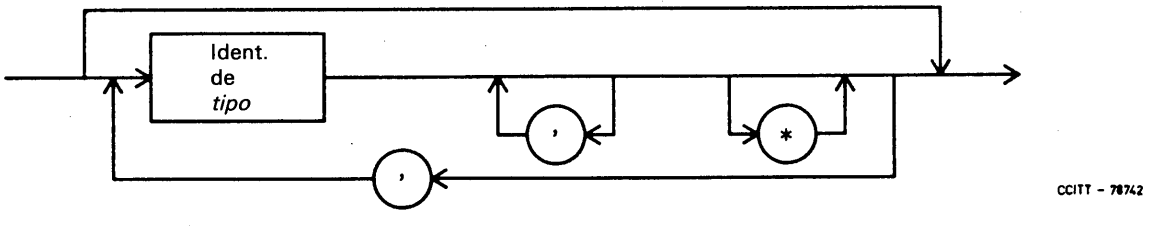
OPERADORES



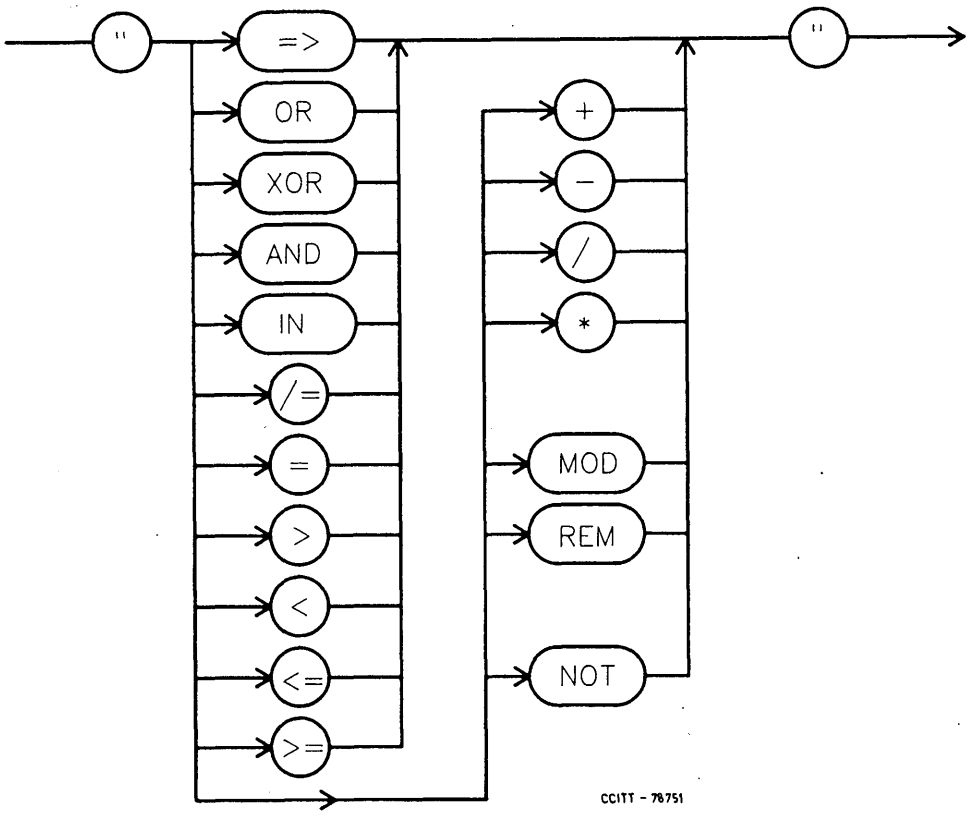
OPERADOR



LISTA DE TIPOS DE OPERADOR



OPERADOR NO-FIJADO



4.5.2 *Semántica*

Los *operadores* de *propiedades de tipo de datos* introducen los *nombres* para *operadores* y la *parametrización* de esos *operadores.* La *parametrización* determina el número de *parámetros* necesarios y el *tipo de datos* de cada *parámetro* así como el *tipo de datos* de cualquier *valor* obtenido.

4.5.2.1 *Nombres de operador*

Los *operadores de ordenación* se especifican incluyendo *Ordenar!* en los *operadores.* Esta es una notación abreviada para introducir los *operadores* siguientes para un tipo de datos D.

«<» : D, D –> Booleano
«>» : D, D –> Booleano
«<=» : D, D –> Booleano
«>=» : D, D –> Booleano

Los nombres de *operadores* infijos, por ejemplo +, Y, O, se encierran entre comillas en los *operadores.* Pueden utilizarse como *operadores de prefijos* utilizando esta forma entrecomillada, a saber:

«+» (a,2)

es equivalente a: a + 2.

Un *nombre de operadores* puede ir seguido facultativamente por un signo de exclamación, que denota que la *identidad de operador* sólo puede mencionarse directamente en *definiciones de tipo de datos.* El signo de exclamación forma parte del *nombre* del *operador* y por tanto tiene que darse siempre cuando se utiliza el *operador.*

Los *nombres asignar!, declarar!, acceder!*, «=» y «/=» se definen implícitamente como *operadores* para todos los *tipos de datos* con la *tipificación* implicada para un *tipo de datos* D.

```
asignar!  : D'*, D- >
declarar! : D'* - >
acceder!  : D' - >
«=»       : D, D- > Booleano
«/=»      : D', D- > Booleano
```

El *operador de asignación* es el operador infijo, «:=». No hay ningún operador de *declaración*, ya que su aplicación está implicada por las *declaraciones*. El operador de *acceso* está implicado siempre que se menciona una *variable* en un contexto que requiere un *valor*. Los *operadores* iguales y no iguales son los *operadores* infijos «=» y «/=», respectivamente.

4.5.2.2 *Tipificación de operador*

La lista de *identificadores de tipos de datos* después de dos puntos (:) y antes del símbolo (- >) se denomina una *lista de tipos de operador*. Esta *lista de tipos de operador* especifica los *tipos de datos* de *valores* que requiere el *operador*. Si uno o más *identificadores de tipos de datos* tiene un *atributo prima*, el *operador* es un *operador activo*, y en caso contrario es un *operador pasivo*. Un *operador activo* puede cambiar los *valores* asociados con *variables*, mientras que un *operador pasivo* es puramente funcional y, por tanto, no puede cambiar los *valores* asociados con *variables*.

Para un *operador pasivo*, todas las *identidades de tipos* en la *lista de tipos* especifica que *parámetros actuales* del *operador* tienen que interpretarse como *expresiones*. Cada una de estas *expresiones* tiene que producir un *valor*, que es un miembro del *conjunto de valores* del *tipo de datos* de la posición correspondiente en la *lista de tipos de datos*.

Para un *operador pasivo*, tiene que haber un *identificador de tipo de datos* después del símbolo (- >). Este *identificador de tipo de datos* especifica que el *operador*, si se aplica, producirá un *valor* de este *tipo de datos*. Para un *operador activo*, algunas de las *identidades de tipo de datos* de la *lista de tipos de operador* van seguidas de signos prima (').

Un signo *prima* especifica que el *operador* requiere una *variable* del *tipo de datos* dado, como *parámetro*, y el *valor* asociado con la *variable* puede cambiarse. Dos *identificadores de tipo de datos* no pueden ir seguidos por el mismo número de signos *prima* en la *lista de tipos de datos* de entrada.

El número de signos *prima* distingue un *parámetro primado* de otro cuando se utiliza una *operación activa* con primación para denotar el *valor* asociado con una *variable* dada como un *parámetro* (esto sólo es permisible en *axiomas*). Por ejemplo:

```
OPERADORES SwapAndAdd: int', int'' - > int
        /* fragmento de definición de tipo de datos */
AXIOMAS SwapAndAdd (a,b)' = b;
        /* axioma que indica el primer parámetro recibe el valor del segundo parámetro */
    SwapAndAdd (a,b)'' = a;
        /* axioma para segundo parámetro */
    SwapAndAdd (a,b) = a+b
        /* axioma para el resultado */
```

Si un *parámetro primado* va seguido de un asterisco, no se accede al *valor inicial* de la *variable* con la *operación* que se utiliza. (Obsérvese que los *axiomas* tienen que ser coherentes con esto, pues de otro modo el LED es *inválido*).

Si el *identificador de tipo de datos* después del símbolo (- >) se omite para una *operador activo*, el *operador* no puede utilizarse dentro de una *expresión*. En una *lista de tipos de datos* de entrada puede haber *parámetros primados* y no *primados*.

Hay una ambigüedad sintáctica entre un *nombre de tipo de datos* seguido por una cadena de caracteres entrecomillada en una cadena de nombres, y un *identificador de tipo de datos* con prima en una *lista de tipos de datos*. Estas ambigüedades surgen cuando una prima después de un *nombre de tipo* en una *lista de tipos* va seguida de una *prima*, una coma o un signo menos (parte de - >). En todos los casos, la *prima* se toma como relativa al *identificador de tipo de datos* y no como comienzo de una cadena de caracteres entre comillas.

4.5.2.3 *Operadores Insertar! y Extraer*

Para permitir la *definición axiomática* de *matrices* y *estructuras*, hay dos *nombres de operador predefinidos*, que tienen una denotación especial fuera de las *definiciones de tipos de datos*. Estos *operadores* son *Insertar!* y *Extraer!*. *Insertar!* es un *operador activo* y debe definirse con *identidades de tipo de datos* tales que el primer *tipo de datos* lleve *prima* y los demás *tipos* no.

Para aplicar *Insertar!* fuera de *definiciones de tipo* de datos se escribe el primer *parámetro* (que debe ser una *variable*) seguido de un paréntesis izquierdo, todos los restantes *parámetros* excepto el último, después un paréntesis derecho y «:=», y finalmente el último parámetro.

Así pues, siendo A una *variable* cuyo *tipo de datos* tiene *Insertar!* definido, y siendo i1, i2 y e *expresiones* que corresponden a *Insertar!* para este *tipo de datos.*

Insertar! (A, il, i2, e)

se escribe

A(il, i2) := e

Teniendo en cuenta que *Insertar!* se puede definir para más de un *tipo de datos,* la aplicación de *Insertar!* se determina a partir del *tipo* de la *variable. Insertar!* tiene que definirse con al menos dos *parámetros* y tiene que dar el mismo *tipo de datos* que el primer *parámetro.*

Extraer! es un *operador pasivo* que require una *variable* como primer *parámetro* por razones semánticas. Para aplicar *Extraer!,* se escribe el primer *parámetro* seguido de todos los demás *parámetros* entre paréntesis.

Así pues,

Extraer! (A, il, i2)

se escribe

A(il, i2)

La aplicación de *Extraer!* se determina por el *tipo* de la *variable.*

4.5.3 *Relación con la sintaxis abstracta*

Para un *operador pasivo,* un *nombre de operador* u *operador infijo* representa un *nombre de operador* en la *sintaxis abstracta.* Para un *operador pasivo,* la *lista de tipos* de entrada representa la lista de *identidades de tipo* de *parámetros* en la *sintaxis abstracta.* El *identificador de tipo de datos* después del símbolo (– >) representa la *identidad de tipo de datos* del resultado de aplicar el operador.

Un *operador activo* se relaciona con la *sintaxis abstracta* convirtiéndolo en *operadores pasivos.* Esto también define el comportamiento de la *operación* en cuanto a la ordenación: para cada *parámetro con prima,* hay un *operador pasivo* implícito que da el *valor* que requiere el *conjunto de axiomas.* Para cada *parámetro con prima* sin un asterisco, hay una *variable implícita* en cada *instancia de proceso* que utiliza el *operador* con el mismo *tipo de datos* que el *parámetro* que recibe el *valor inicial* del *parámetro.*

La lista de *identidades de tipo de datos* de *parámetros* en la *sintaxis abstracta* para cada *operador pasivo implícito* se representa por la *lista de tipos de datos* de entrada prescindiendo de cualquier parámetro con asterisco. Hay tantos *operadores pasivos implícitos* como *parámetros con prima* en la *lista de tipos de datos* de entrada y cada *tipo de datos de parámetro con prima* se utiliza como la *identidad de tipo de datos* del resultado de aplicar uno de esos *operadores.* Por ejemplo:

```
OPERADORES
    complejo: Entero', Entero'', Entero, Entero''' * – > Bool
        /* cambia los dos primeros parámetros, pone la suma de los tres primeros parámetros en el
        cuarto parámetro y da verdadero si el segundo y tercer parámetros son iguales */
Axiomas
    complejo (a,b,c,d)' = b;
    complejo (a,b,c,d)'' = a;
    complejo (a,b,c,d)''' = a+b+c;
    complejo (a,b,c,d) = (b=c);
        /* Véase el § 4.9.2 en cuanto al empleo de primas en axiomas */.
```

Los operadores implícitos son:

```
implicado 1! : Entero, Entero, Entero – > Entero
implicado 2! : Entero, Entero, Entero – > Entero
implicado 3! : Entero, Entero, Entero – > Entero
implicado 4! : Entero, Entero, Entero – > Booleano
```

Si las *variables implicadas* son V1, y V2, una aplicación de complejo en una sentencia es equivalente a:

V1 : = a ; V2 : = b;

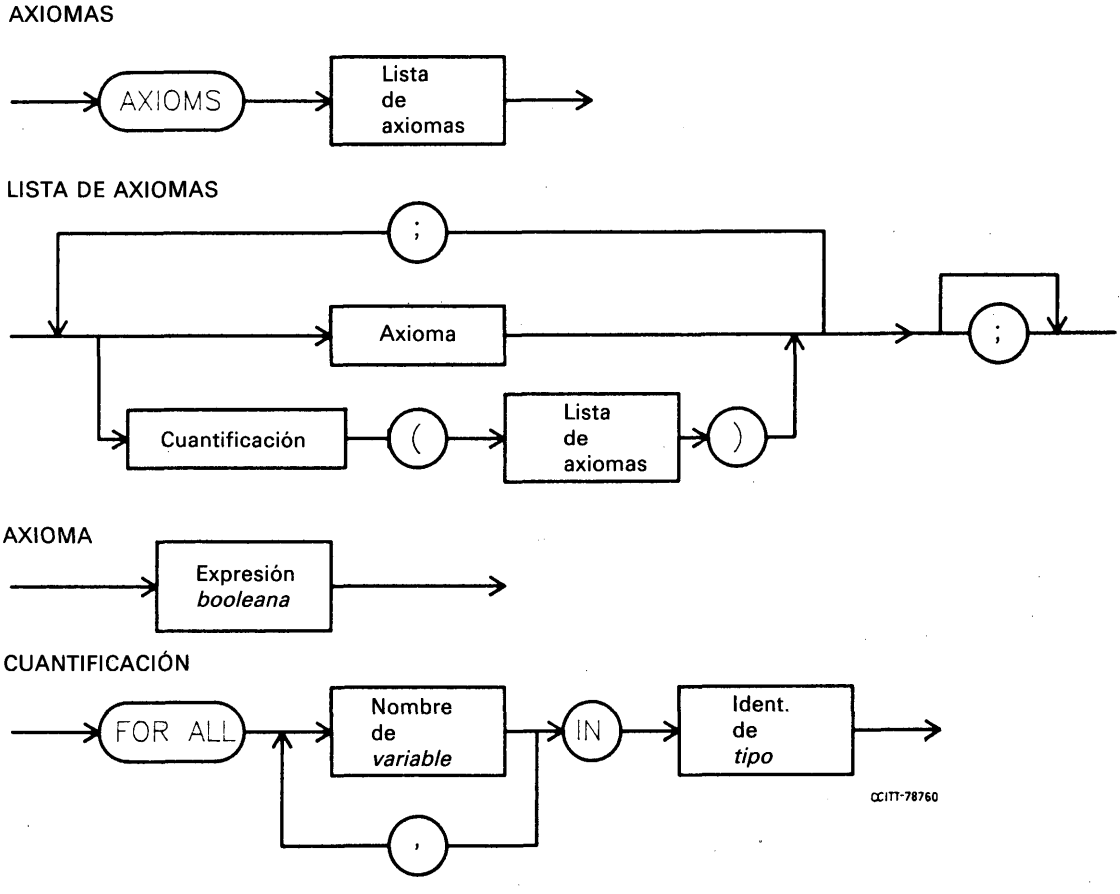
Si a se sustituye por V1, b se sustituye por V2 y complejo se sustituye por implicado 4!.

a: implicado 1! (V1,V2,c);
b: implicado 2! (V1,V2,c);
d: implicado 3! (V1,V2,c);

Cuando en una sentencia hay más de un *operador activo*, se sustituyen estos en el orden en que han de *interpretarse.*

4.6 *Axiomas*

4.6.1 *Sintaxis*



4.6.2 *Semántica*

Los *axiomas* son un conjunto de *expresiones Booleanas* que son verdaderas para todos los *valores* de las *variables* en los *axiomas.*

Dentro de un *axioma*, una *«variable»* nunca es una *variable de proceso* o de *procedimiento*, que tiene un *valor* asociado con él, sino que se utiliza para denotar que todos los *valores* de un *tipo* específico pueden sustituir a la *variable* y el *axioma* sigue siendo verdadero. Al considerar esta sustitución, un *nombre de variable* dado siempre representa el mismo *valor* en un *axioma.* Por ejemplo, en:

```
OPERADORES par: Entero – > Booleano
AXIOMAS par (0)  = Verdadero;        /* axioma 1 */
        par (i)  = NO par (i+1);     /* axioma 2 */
```

El *axioma* 2 tiene que ser válido para 1 = 2, 1 = 3, 1 = 4, etc., es decir:

par (2) = NO par (2+1)
par (3) = NO par (3+1)
par (4) = NO par (4+1)

De ordinario, el *tipo de datos* de una *variable* en un *axioma* puede determinarse por el *contexto*; por ejemplo, en el ejemplo anterior se requiere que la *sintaxis de operador* sea del *tipo de datos* int.

A veces, debido a la *sobrecarga* de *nombres* y símbolos (tales como «+») en el LED, no es posible determinar el *tipo de datos* de una *variable* de *axioma* por el *contexto*, por lo que se necesita la *cuantificación* facultativa. La *cuantificación* obliga a que una *variable* sea de *tipo* particular. Si la utilización de una *variable* dentro de un *axioma* es *ambigua* o *incoherente*, el LED/PR es *inválido*.

La *cuantificación* permite también que un *nombre de variable* represente la misma sustitución en más de un *axioma*.

Los *nombres de variable* elegidos para *variables* en *axiomas* tienen que ser distintos de los *literales* apropiados al *contexto* en que se utiliza la *variable*.

Como los operadores «=» y «/=» están *implicados* para todos los *tipos de datos*, siempre están *implicados* los *axiomas* siguientes:

«/=» (a,b) = NO («=» (a,b))
«=» (a,a);
«=» (a,b) Y «=» (b,c) => «=» (a,c);
«=» (a,b) => «=» (b,a);

Siempre que se especifica *Ordenar!*, están *implicados* los *axiomas* siguientes:

« < » (a,b) => NO « > » (a,b);
« > » (a,b) => NO « < » (a,b);
« < » (a,b) Y « < » (b,c) => « < » (a,c);
NO « < » (a,a);
« < = » (a,b) => « < » (a,b) 0 « = » (a,b)
« > = » (a,b) => « > » (a,b) 0 « = » (a,b)

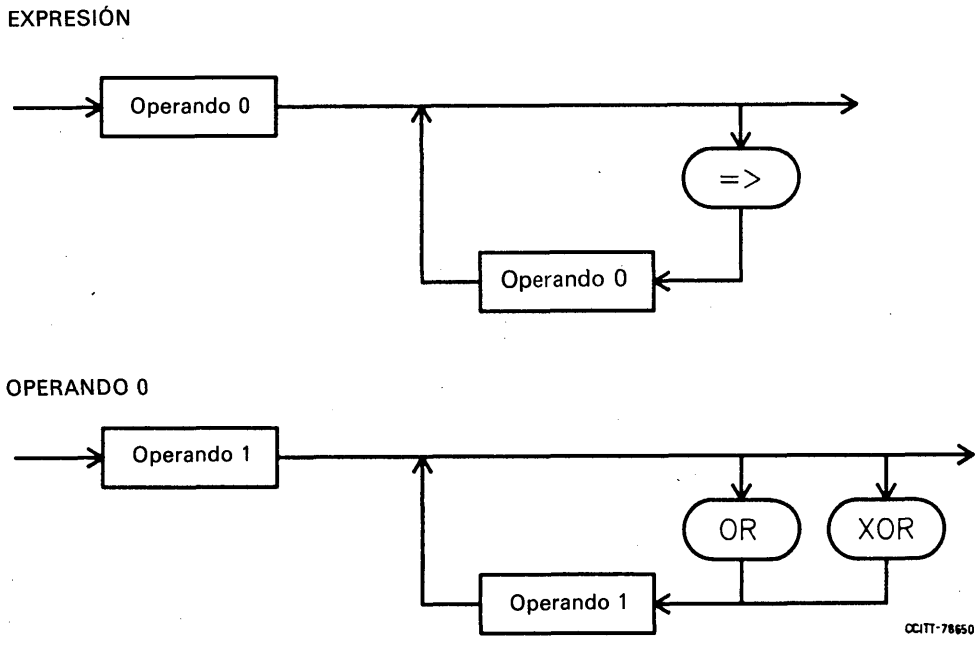
4.6.3 *Relación con la sintaxis abstracta*

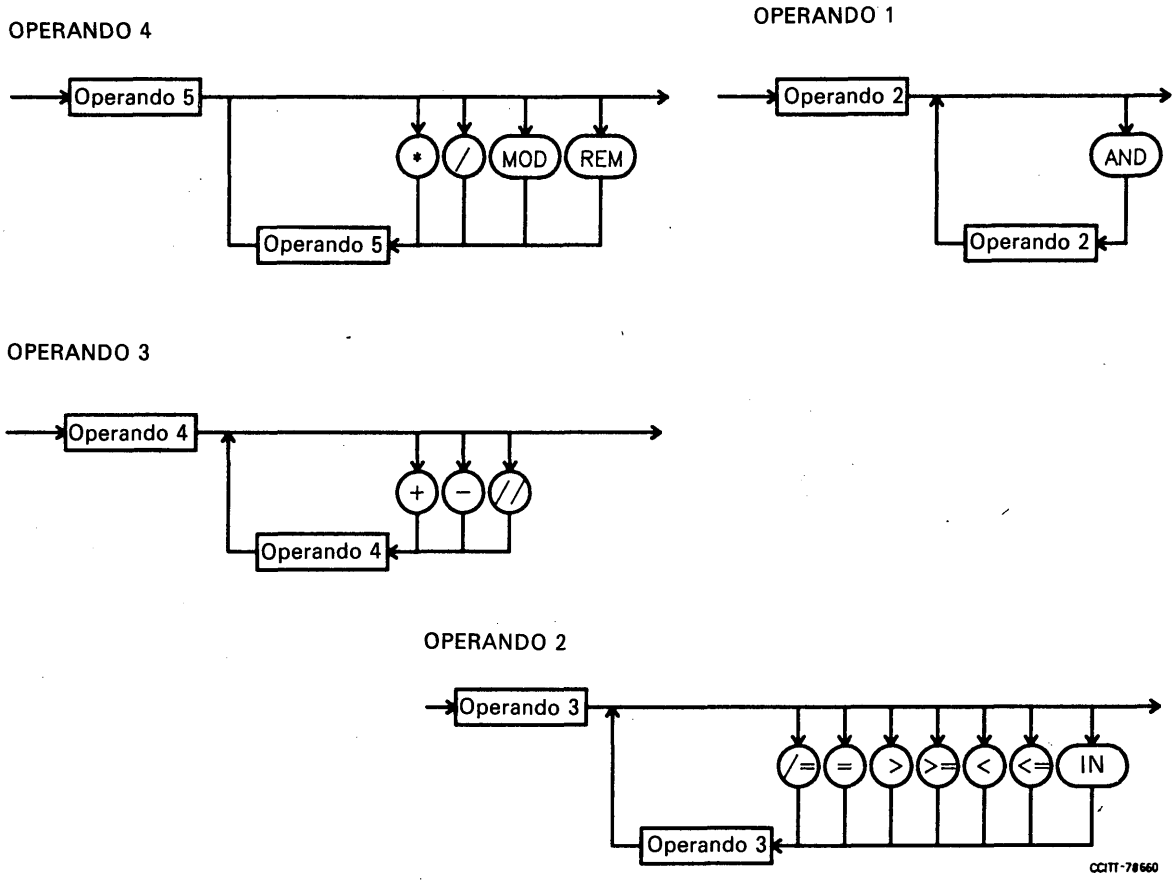
Los *axiomas* representan los *axiomas de tipos de datos* en la *sintaxis abstracta*. Cada *axioma* representa un *axioma* en la *sintaxis abstracta*.

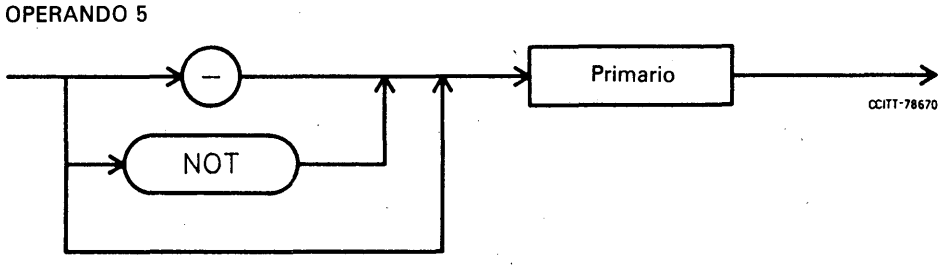
La *cuantificación* representa *cuantificación* en la *sintaxis abstracta*, salvo que hay *cuantificación implicada* en la *sintaxis concreta* para todas las *variables de axioma* cuyo *tipo de datos* se determina por el *contexto*.

4.7 *Expresiones*

4.7.1 *Sintaxis*







4.7.2 *Semántica*

Una *expresión* es o bien un *primario* o la aplicación de un número de operadores «*infijos*».

El orden de la aplicación de los *operadores* viene determinado por su aparición en la *sintaxis*, de modo semejante a los lenguajes de programación tales como CHILL (véase la Recomendación Z.200). Sin embargo, el LED permite también el *operador de implicación Booleana* (= >), que tiene menor *precedencia* que cualquier otro *operador*. Este tiene el valor *FALSO* para *operandos Booleanos* si el *operando* de la izquierda es *VERDADERO* y el *operando* de la derecha es *FALSO*. En caso contrario, para *operandos Booleanos* la *operación implicada* tiene el valor *VERDADERO*.

De ordinario, todos los *operadores* tendrán las mismas *propiedades* y la misma validez, como se define en los lenguajes de programación, pero hay que observar que en el LED el usuario puede definir nuevos significados para esos *operadores* incluyéndolos en *definiciones de tipo de datos*. Sin embargo, es posible que no cambie la *precedencia* de los *operadores* «*infijos*».

El *valor* producido por una *operación* válida se determina por los *axiomas* en las *definiciones de tipo de datos*.

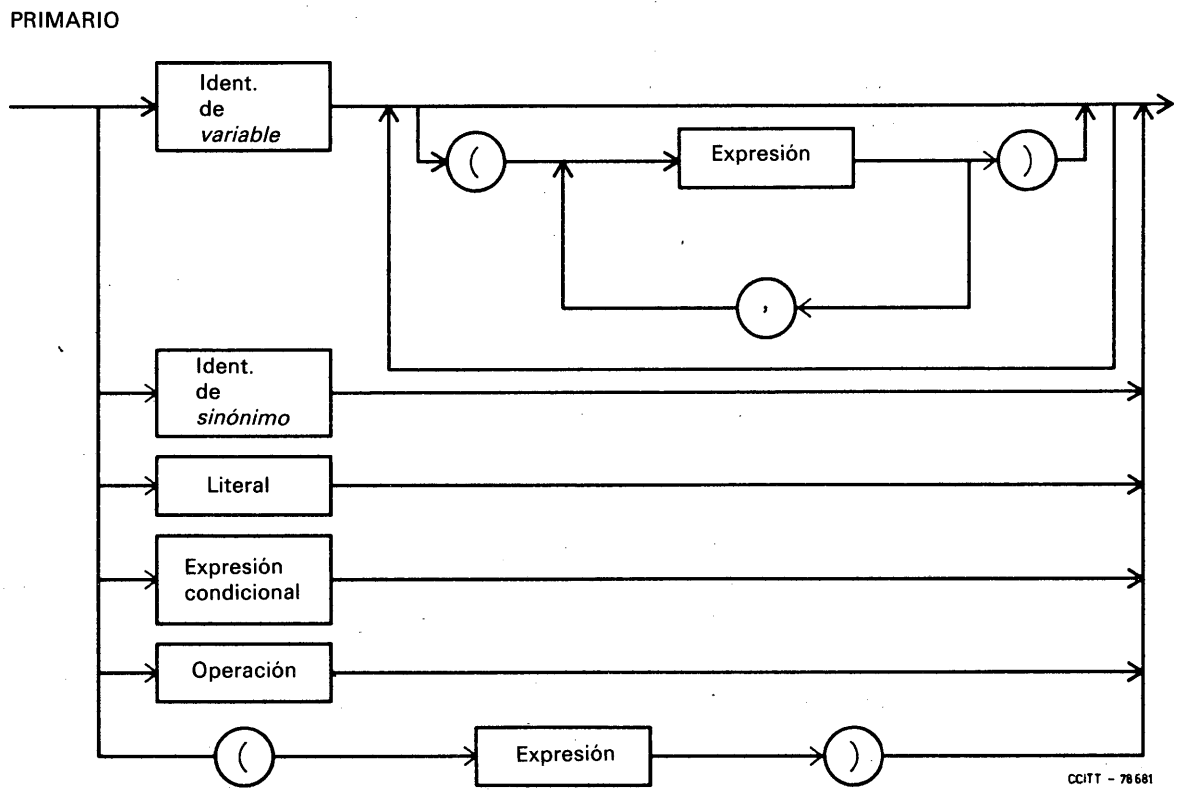
4.7.3 *Relación con la sintaxis abstracta*

Una *expresión* representa una *expresión* en la *sintaxis abstracta.*

Un *operador* «*infijo*» puede estar *sobrecargado* y puede representar uno cualquiera de varios *operadores* en la *sintaxis abstracta.* La *sobrecarga* se resuelve de dos modos: o bien por el número y *tipo* de *parámetros* o, en el caso en que los propios *parámetros* están *sobrecargados,* por el *tipo de datos* requerido en el *contexto* en que se utiliza la *operación.*

4.8 *Primario*

4.8.1 *Sintaxis*



4.8.2 *Semántica*

Un *primario* es una *identidad de variable,* o una *identidad de literal* o de *sinónimo,* o una *expresión condicional,* o una *operación,* o una *expresión entre paréntesis.*

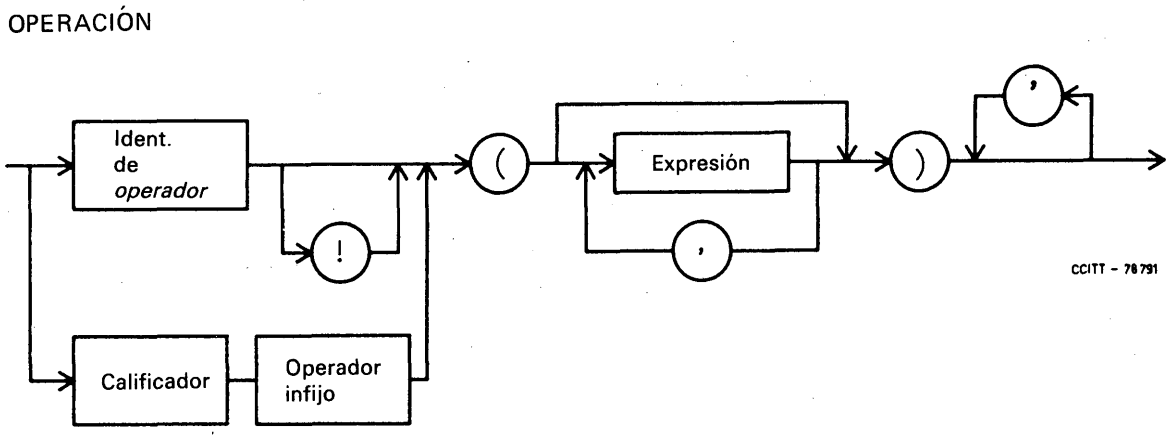
4.8.3 *Relación con la sintaxis abstracta*

Una *identidad de variable,* o una *identidad de literal* o de *sinónimo,* o una *expresión condicional* o *expresión entre paréntesis* representa un *acceso de identidad de variable, identidad de valor, identidad de sinónimo, expresión condicional* o *expresión,* respectivamente, en la *sintaxis abstracta.*

Cuando una *identidad de variable* se menciona en un *axioma,* representa una *variable axiomática* más bien que una referencia a un *acceso* a una *variable* declarada para un *proceso* o *procedimiento.* El *tipo de datos* de esa *variable axiomática* se determina por el *contexto.*

4.9 *Operación*

4.9.1 *Sintaxis*



4.9.2 *Semántica*

Una *operación* es la aplicación de un *operador* definido en una *definición de tipo.* El número y *tipo* de las *expresiones* utilizadas como *parámetros actuales* tiene que ser coherente con la definición de la *identidad de operador.* Estos *parámetros* pueden utilizarse para determinar el *operador* que se aplica, si el *nombre de operador* está *sobrecargado.*

Si el *operador* se aplica en un *axiona* y el *nombre* se define con un signo de exclamación, este signo de exclamación tiene que repetirse en los *axiomas.*

Un *operador* definido con un signo de exclamación no puede utilizarse fuera de una *definición de tipo.*

Las *primas* después del paréntesis derecho del *operador* sólo pueden utilizarse en un *axioma* y denotan que la *operación* tiene el *valor* resultante del *parámetro* definido con ese número de *primas.* Por ejemplo:

```
OPERADORES
    ejemplo: tf', t2'' - >

AXIOMAS
    ejemplo (v1,v2)'' = v1
    /* valor de v1 puesto en v2 */
```

Cuando la *tipificación* de un *parámetro* de un *operador* se especifica con *primas*, el parámetro actual tiene que ser una *variable*, salvo en el *contexto* de *un axioma.*

4.9.3 *Relación con la sintaxis abstracta*

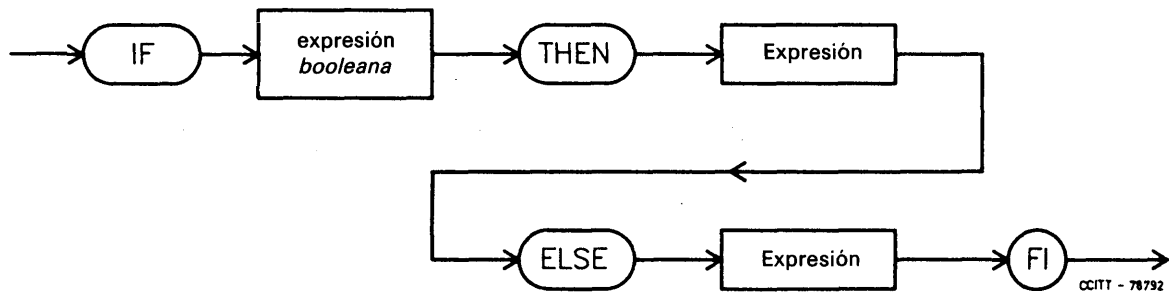
Un *identificador de operador pasivo* representa un *operador* en la *sintaxis abstracta.* Para una *operación pasiva*, la lista de *expresiones* representa la lista de *expresiones* en la *sintaxis abstracta* para esa *operación.*

Una *operación activa* representa la *asignación* a *variables implícitas* que se utilizan después como *argumentos* para *operaciones pasivas implícitas*, cuyos *valores* se *reasignan* a *variables* dadas como *parámetros actuales* (véase el § 4.5.3). Dentro de un *axioma*, una *operación activa* representa una aplicación de la *operación pasiva implícita* apropiada determinada por el número de *primas* añadidas a la *operación.*

4.10 *Expresión condicional*

4.10.1 *Sintaxis*

EXPRESIÓN CONDICIONAL



4.10.2 *Semántica*

La *expresión condicional* se *interpreta interpretando* la *expresión* después de ENTONCES (*THEN*) si la *expresión Booleana es verdadera*, y en caso contrario produce el *valor* de la *expresión* después de *SI NO* (*ELSE*).

Dentro de un *axioma*, cada rama de la *expresión condicional* sólo tiene que ser válida para las condiciones en las que se elige. Por ejemplo, como log(x) no se define para números negativos, en

SI r > 0 ENTONCES log (r) SI NO 0.0

no importa que para r < = 0, log(r) no esté definido

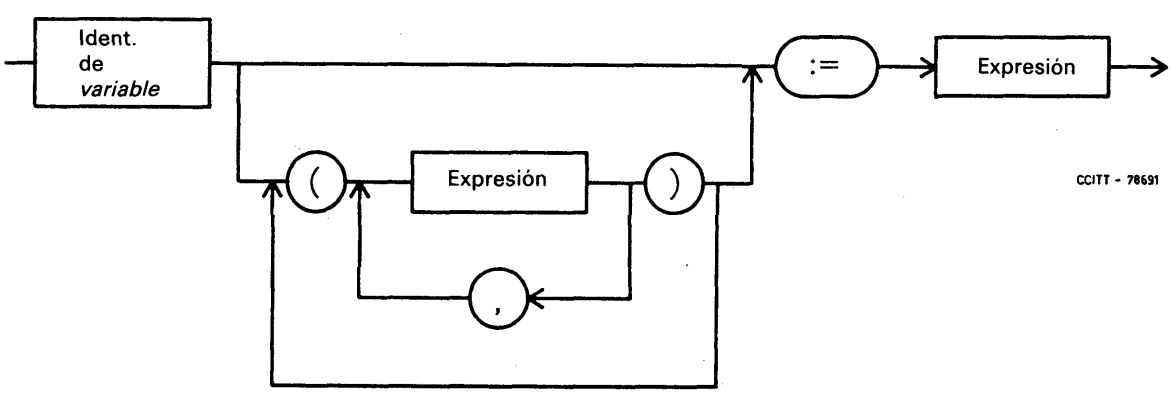
4.10.3 *Relación con la sintaxis abstracta*

Una *expresión condicional* representa una *expresión condicional* en la *sintaxis abstracta.*

4.11 *Sentencia de asignación*

4.11.1 *Sintaxis*

SENTENCIA DE ASIGNACIÓN



4.11.2 *Semántica*

Una *sentencia de asignación* permite asociar un *valor* con una *variable.*

El *valor* producido por la *expresión* del lado derecho se *asigna* a la *variable* o a un elemento de la *variable* del lado izquierdo.

4.11.3 *Relación con la sintaxis abstracta*

Una *sentencia de asignación* representa la aplicación o bien de un operador de *asignar!* o de un operador de *insertar!*

Hay un mapeado entre la *sintaxis* de una *asignación* y la aplicación del *operador* apropiado.

Cuando no se utilizan paréntesis en el lado izquierdo de una *asignación*, la *asignación* representa la utilización de *Asignar!*, de modo que

V := e representa Asignar! (V,e)

Cuando se utiliza un solo par de paréntesis, la asignación representa *Insertar!*, de modo que

a(i) := e representa Insertar! (a,i,e)

y

a(i,j) := e representa Insertar! (a,i,j,e)

Cuando se utilizan múltiples paréntesis, una *asignación* se representa por sustitución repetida, de modo que

a(i)(j) := e

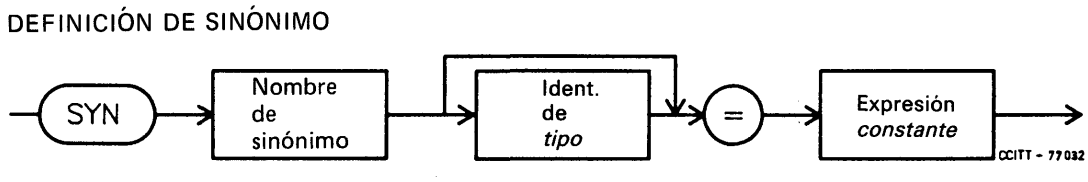
representa

```
vi := a(i);
Insertar! (Vi,j,e);
a(i) := vi;
```

donde vi es una *variable implícita* cuyo *tipo* es el mismo que a(i). Las operaciones *Insertar!* implicadas son *operaciones activas*, que se representan en la *sintaxis abstracta* del modo normal para *operadores activos* (véase el § 4.9.3).

4.12 *Definición de sinónimo*

4.12.1 *Sintaxis*



4.12.2 *Semántica*

El *sinónimo* es equivalente a la *expresión constante*. Si el *tipo* de la *expresión* no puede determinarse por la *constante* o el *contexto*, hay que especificar un *tipo*. El *valor* y *tipo* de la *expresión constante* (y, por tanto, el valor del *sinónimo*) se determinan por el *contexto* en que aparece la *definición de sinónimo*.

4.12.3 *Relación con la sintaxis abstracta*

Un *sinónimo* representa una *definición de sinónimo* en la *sintaxis abstracta*. Si se omite el *tipo de datos*, éste está implicado por el *contexto*.

4.13 *Generador de tipo de datos*

4.13.1 *Sintaxis*

4.13.2 *Semántica*

El *nombre* dado en un *generador de tipo de datos* es un *nombre de generador de tipo de datos*. Las *propiedades* suministrdas en un *generador* forman una especificación parcial de un *tipo de datos*. Cuando se utiliza un *generador* en una *definición de tipo* o en otro *generador de tipo de datos* los *parámetros* para el *generador* se sustituyen textualmente en la *definición de generador* y (junto con cualquier propiedad añadida en una *definición*

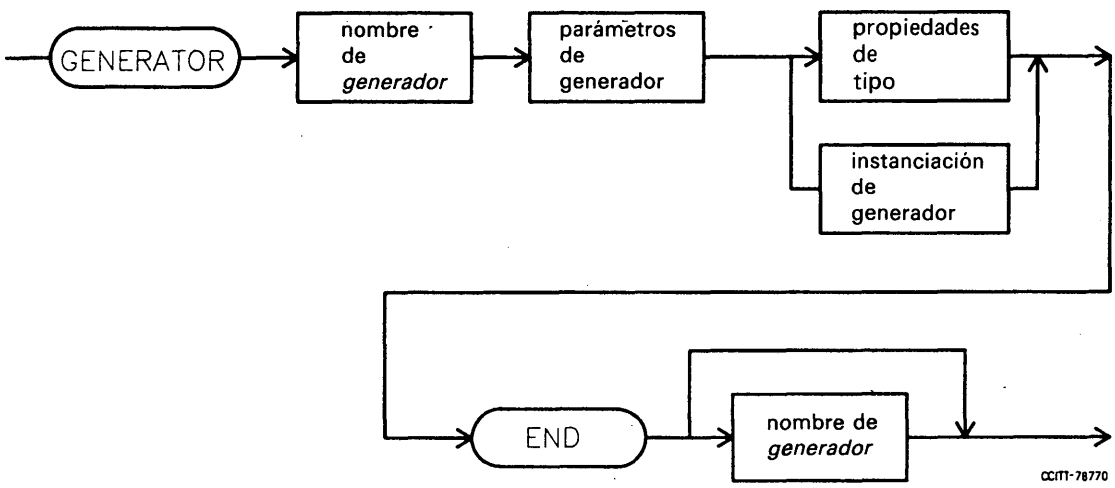
de tipo) entonces tienen que formar una *definición de tipo* completa u otro *generador de tipo de datos.* Cuando un *generador de tipo de datos* se define en términos de una *instanciación de generador,* los *parámetros de generador* pueden tener el mismo *nombre* que los suministrados a la *instanciación.* Ese *generador* es una *instanciación parcial.* Por ejemplo:

```
Generador pila (Componente de tipo, Tamaño máximo constante)
    /* detalles */
    Fin pila;
Generador Ipila (Max constante) Pila (Entero, Max)
    Fin Ipila
    /* Una pila de enteros, máximo tamaño no especificado */
```

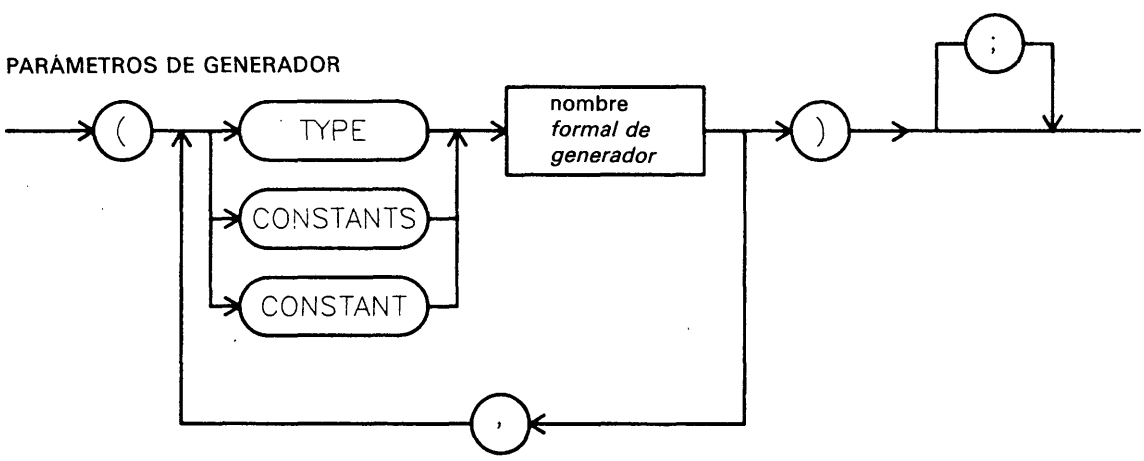
4.13.3 *Relación con la sintaxis abstracta*

El generador de tipo de datos no tiene contraparte en la sintaxis abstracta. La utilización de una instanciación de generador en una definición de tipo de datos denotará el texto formado por sustitución paramétrica.

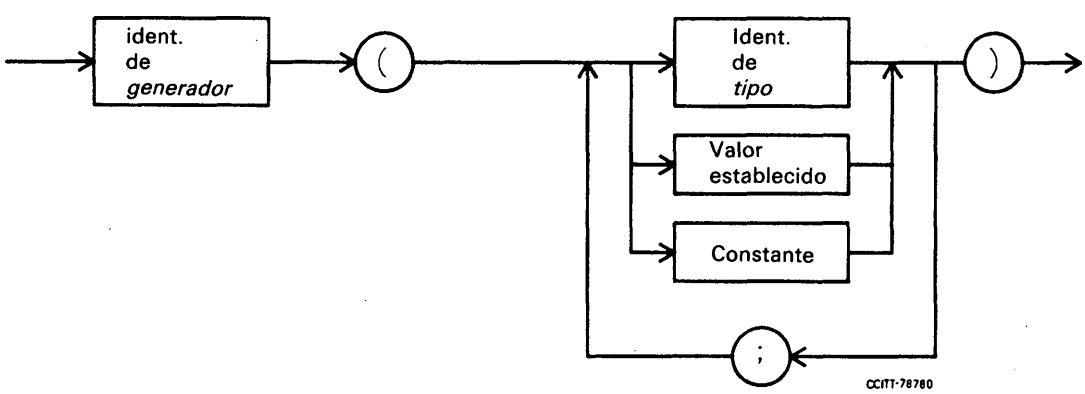
GENERADOR DE TIPOS DE DATOS



PARÁMETROS DE GENERADOR



INSTANCIACIÓN DE GENERADOR



5 Tipos de datos predefinidos

5.1 *Entero*

Neotipo entero
/* Literales con arreglo a sintaxis de literal entero */

Operadores
«+»: Entero, Entero – > Entero;
«–»: Entero, Entero – > Entero;
«*»: Entero, Entero – > Entero;
«/»: Entero, Entero – > Entero;
Flotante: Entero – > Real;
Fijo: Real – > Entero;

AXIOMAS
/*heredado de enteros matemáticos, añadiendo: */
Fijo (Flotante (r)) = r;
r – 1 < Flotante(Fijo(r)) <= r;
i/j = Fijo (Flotante(i)/Flotante(j))

Fin entero;

5.2 *Real*

Neotipo real
/* Literales de la forma especificada en «literal real» */

Operadores
«–»: Real, Real – > Real;
«+»: Real, Real – > Real;
«*»: Real, Real – > Real;
«/»: Real, Real – > Real;
«–»: Real – > Real;

/* Los axiomas se heredan de los reales matemáticos; este trabajo necesita ulterior estudio para especificarlo aquí */

FIN Real;

5.3 *Matriz (Array)*

Matriz de generador (Índice de tipo, Item de tipo);

Operadores
Insertar!: Matriz', Índice, Item – >;
Extraer!: Matriz, Índice – > Item;

Axiomas
Extraer! (Declarar!(v)',]) = Error!;
Extraer! (Insertar!(A, IPos, It), EPos) =
Si Epos = Ipos Entonces It Si no Extraer (A, EPos);

FIN Matriz;

5.4 *Booleano*

Neotipo Booleano

Literales Verdadero, Falso;

Operadores
«No»: Booleano' – > Booleano;
Y: Booleano, Booleano – > Booleano
O: Booleano, Booleano – > Booleano;
«– >»: Booleano, Booleano – > Booleano;

Axiomas
No (Verdadero) = Falso;
No (Falso) = Verdadero;
Y (A, B) = Si A = Falso Entonces Falso Si no B;
O (A, B) = Si A = Verdadero Entonces Verdadero Si no B;
«= >» (A, B) = Si A = Verdadero y B = Falso Entonces Falso Si no Verdadero;

FIN Booleano;

5.5 *Carácter*

Neotipo carácter

Literales /* el alfabeto del CCITT */;

Operadores
Ordenar!;

FIN carácter;

5.6 *Natural*

Sin tipo natural = Constantes Entero = O Fin natural;

5.7 *Conjuntista (Powerset)*

GENERADOR Conjuntista (Item de tipo);

Literales Vacío;

Operadores
«Dentro»: Item, Conjuntista – > Booleano;
Incl: Item, Conjuntista' – >;
Del: Item, Conjuntista ' – >;
Ordenar!;

Axiomas;
I DENTRO Declarar!' (v) = Falso;
I DENTRO Incl (I2, S) = SI I = I2 ENTONCES Verdadero ELSE I IN I2;
Del (I, Incl (I2, S)) = SI I = I2 ENTONCES Del (I, S) SI NO Incl (I2, Del (I, S));
Del (I, Incl (I2, 5) = SI I = I2 ENTONCES Del (I, S) SI NO Incl (I2 Del (I, S)) FI;
Para todos los S1, S2 en el conjuntista (para todos los I en el Item (S1 < S2 = > (I EN S1 = > I EN S2)));
NO (I EN S) = > Del (I, S) = S

FIN Conjuntista;

5.8 *PId (Identificador de proceso)*

Neotipo PId

Literales CERO;

Operadores
Crear! – PId;

Axiomas;
Declarar! (v)' = Nulo;
Crear! /= Crear!;
/* Un modo débil de declarar que todos los Crear producen valores únicos */
Crear! /= Nulo

/* Los valores PId vienen dados por la interpretación de un nodo de petición de crear (se denotan como generados por la función crear! anterior). Cada imterpretación de petición de crear genera un valor PId único que no es cero. Cada instancia de proceso declara implícitamente tres variables PId, denominadas «Ascendiente», «Mismo» y «Descendiente».

La interpretación de un nodo de petición de crear genera un valor PId nuevo y único, y lo asigna al Descendiente» para la tarea creadora. Al identificador (Mismo) de la tarea creada se asigna ese mismo valor, mientras que a Ascendiente de la tarea creada se asigna el valor de Mismo del creador. La longitud de Descendiente en la tarea creada se fija en cero.

*/ FIN PId;

5.9 *Cadena*

```
Generador cadena (Tipo item);
        /* Literales especificados por literales de Cadena de tarea, */
        /* Sólo si el generador se instancia con Carácter */

    Operadores
        Declarar!  – > cadena;
        «//»:      Cadena, cadena – > cadena;
        Longitud:  Cadena – > Natural;
        Primero:   Cadena – > Item;
        Último:    Cadena – > Item;
        Extraer:   Cadena, Natural – > Item;
        Insertar:  Cadena', Natural, Cadena – > Cadena;
        Insertar:  Cadena', Natural, Item – >;

    Axiomas;
        Longitud (Declarar!) = 0;
        Longitud (S1 // S2) = Longitud (S1) + Longitud (S2);
        Primero (S1) = Extraer (S1, 1);
        Último (S1) – Extraer (S1, Longitud (S1));
        Extraer (Insertar (S1, I, S2), j) =
          SI j < 1 ENTONCES Error i SI NO
            SI j < = I ENTONCES Extraer i (Si, j) SI NO
              SI j < = Longitud (S2) + I ENTONCES Extraeri (S2, J-I)) SI NO
                SI j < = Longitud (S1) + Longitud (S2) ENTONCES
                          Extraeri (S1,J-Longitud(S2)) SI NO Errori
                FI
              FI
            FI
          FI;
        S1 // S2 = Insertar (S1, Longitud(S1),(S2);
        Extraeri (Insertari(S1, I, It)', J) =
          Si I > J Entonces It Si no Extraeri(S1, J) FI;

    FIN Cadena;
```

5.10 *Tiempo*

```
NEOTIPO Tiempo Hereda Real

AÑADIENDO
    Operadores
        Ahora: – > Tiempo /* el tiempo «real» */;

Fin Tiempo;
```

5.11 *Duración*

```
Neotipo Duración Hereda Real («+», «–», «*», «/»)

Añadiendo
    Operadores
        «+»: Tiempo, Duración – > Tiempo;
        «+»: Duración, Tiempo – > Tiempo;
        «–»: Tiempo, Duración – > Tiempo;

Constantes > = 0.0

Fin Duración;
```

5.12 *Temporizador*

```
Neotipo Temporizador

    Operadores
        Inicialización: Tiempo, Temporizador'* – >;
        Reinicialización: Temporizador' – >;
        Activo: Temporizador – > Booleano;

    Axiomas
        Activo (Inicialización (Tm, Tmr)' ) = Tm > Ahora ();
        Activo ( (Reinicialización (Tmr) ) = Falso;
        /* Obsérvese que un temporizador activo envía una señal al proceso cuando pasa a ser inactivo.
        Esta señal se denomina con el mismo nombre que la variable de temporizador en la Iniciali-
        zación de llamada */

    Fin Temporizador;
```

5.13 *Cadena de caracteres*

```
NEOTIPO Cadena de caracteres Cadena (Carácter)
    Añadiendo LITERALES /* carácter cadena literales*/
    FIN Cadena de caracteres;
```

Impreso en Suiza — ISBN 92-61-02233-2