UNIVERSIDAD POLITECNICA DE LAS PALMAS
INGENIERIA TECNICA DE TELECOMUNICACION
PROYECTO FIN DE CARRERA

ORIGINAL

TITULO: - LENGUAJE DE PROGRAMACION PL/M-8Ø.
- SISTEMA DE SEGURIDAD CONTROLADO POR MICROPROCESADOR.

AUTOR: D. José Manuel Aldana García.

TUTOR: D. Sebastián Suárez Gil.

JUNIO-86.

INDICE.

PARTE III.

SISTEMA DE SEGURIDAD.

APENDICE.

APENDICE A. Esquema Eléctrico.

APENDICE B. Listado del Programa.

INTRODUCCION.

El objetivo del presente proyecto es, por un lado, el diseño de un sistema de seguridad para una vivienda cuya unidad de control está compuesta por un sistema microprocesador, y por otro, el estudio del lenguaje de programación de alto nivel para microprocesadores, PL/M-8Ø; lenguaje en el cual se basa el software empleado por el sistema de seguridad.

Para ello se ha dividido el proyecto en tres partes. En la primera de ellas se realiza el estudio del lenguaje de programación PL/M-8Ø. Se ha estructurado éste de un forma clara y concisa para conseguir un estudio del lenguaje lo más didáctico posible.

En la segunda parte se desarrollan una serie de conocimientos básicos sobre la estructura de los sistemas de seguridad y los dispositivos que lo forman. Estos conocimientos se hacen necesarios para llegar a un mejor entendimiento de los sistemas de seguridad y poder hacer valoraciones acerca de los mismos, máxime en la actualidad donde estos sistemas han alcanzado un auge sin precedentes, debido sobre todo, al constante aumento de los índices de criminalidad. Esto ha dado lugar, también, a un mayor perfeccionamien-

to de los sistemas de seguridad, incorporando sofisticados dispositivos de detección así como la utilización del microprocesador en las unidades de control, dando a estos sistemas prestaciones imposibles de alcanzar por sistemas que utilizan unidades de control convencionales.

En la tercera parte se realiza el diseño del sistema, particularizándolo para unas condiciones concretas de detección y respuestas del mismo ante determinados sucesos.

PARTE - I

LENGUAJE DE PROGRAMACION PL/M-80

I-1. INTRODUCCION.

El PL/M-8Ø es un lenguaje de alto nivel diseñado para su utilización en sistemas microprocesador en el que intervengan los microprocesadores de Intel 8Ø8Ø ó 8Ø85.

Este lenguaje facilita el desarrollo y puesta en marcha del programa con rapidez por lo que es de inestimable ayuda para la creación del software del microprocesador.

El PL/M-8Ø es un lenguaje algorítmico, lo que permite al programador ocuparse casi por entero en el desarrollo de su propio sistema sin necesidad de preocuparse por los detalles del lenguaje Assembly.

Un resumen de las prestaciones del PL/M-8Ø es el siguiente:

- Reduce el desarrollo del software.
- Acelera la realización del sistema proyectado, dándole además una mayor fiabilidad.
- Produce un código objeto reubicable y enlazable.
- Optimiza el código, reduciendo así la cantidad de memoria necesaria.

I-2. ELEMENTOS BASICOS DE PL/M-8Ø.

Al ser el PL/M un lenguaje algorítmico, los programas pueden ser escritos libremente, expresando con naturalidad el algoritmo a programar. Las lineas de entrada son columnas independientes y los espacios pueden ser insertados libremente entre los elementos del programa.

I-2-1. CARACTERES UTILIZADOS EN PL/M.

Los caracteres válidos utilizados en PL/M pueden ser:

- Alfanuméricos:

A B C D E F G H I J K L M N O P

Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p

q r s t u v w x y z

Ø 1 2 3 4 5 6 7 8 9

- Caracteres especiales:

= . / ( ) + - ' * , < > : ;

- Caracteres de espacio:

espacio tabulador retorno del carro (CR)

En el caso de que un programa PL/M contenga algún caracter diferente de los enunciados, el compilador dará un mensaje de error.

En los programas PL/M se pueden utilizar tanto letras mayúsculas como minúsculas. Por ejemplo, abc y ABc son equivalentes.

I-2-2. IDENTIFICADORES.

Los identificadores se utilizan para definir variables, procedimientos (procedure), macros y etiquetas. Un identificador puede tener un máximo de 31 caracteres alfanuméricos. El primer caracter del identificador debe de ser siempre alfabético, y los siguientes pueden ser alfabéticos, numéricos o una combinación de ambos.

El símbolo $ es totalmente ignorado por el compilador y puede ser utilizado como separador. Un identificador que contenga el símbolo $ es equivalente al mismo identificador que no lo contenga.

Identificadores válidos podrían ser los siguientes:

X

SELET

PROCESODEMASCARA

LECTURA1

LECTURA$1

Los dos últimos identificadores son exactamente iguales.

I-2-3. COMENTARIOS.

En PL/M existe la posibilidad de incluir comentarios. Estos comentarios pueden ser de gran ayuda, tanto para personas ajenas al trabajo de programación como al propio programador, ya que en dichos comentarios se pueden hacer aclaraciones o valoraciones sobre una sentencia o grupo de sentencias determinada.

Para incluir comentarios solo es necesario poner un indicativo al principio y al final de la linea de comentario. El indicativo de comienzo es /* y el de final */

Un ejemplo de comentario sería el siguiente:

```
/* Esta sentencia nos compara dos números */
```

En el momento de la compilación del programa el compilador ignorará lo que se encuentre en el interior de los indicativos de comienzo y final de comentario.

Los comentarios pueden ocupar tanto una única linea de programa como compartir dicha linea con una sentencia.

Una regla general que hay que tener en cuenta es que todo comentario debe de estar abierto y cerrado en la misma linea.

I-3. VARIABLES Y CONSTANTES. DECLARACIONES.

Los datos con los que trabaja el PL/M pueden ser variables ó constantes. Las variables son datos cuyo valor puede variar durante la ejecución del programa. Cada variable es conocida en el programa por medio de un identificador. Las constantes son valores fijos que no pueden cambiar a lo largo del programa; a éstas se hace referencia directamente.

La expresión:

```
INDEX = (INDICE$PROCE - 1) / 4
```

está formada por las variables INDEX e INDICE$PROCE y las constantes 1 y 4.

I-3-1. CONSTANTES NUMERICAS.

Una constante es un valor conocido en tiempo de compilación, el cual no cambia durante la ejecución del programa. Las constantes numéricas pueden ser expresadas en binario, decimal, octal y hexadecimal.

En general, la base en la que se encuentra un número es representada por una de las siguientes letras:

B Q O D H

Estas deben de ir colocadas justo detrás del número que se quiera expresar. La letra B indica una constante binaria. Las letras O y Q indican

una constante de base octal. La letra H indicará que se trata de una constante hexadecimal. La letra D indica que se trata de una constante en base decimal. En ésta base, decimal, la letra D es opcional.

Ejemplos de representación de constantes numéricas en PL/M serían:

2 44Q 1Ø1B 22D ØBFH 22

El símbolo $ puede ser libremente insertado entre los caracteres de una constante, obteniendose así una mejor lectura de la misma.

Las dos constantes binarias siguientes son exactamentes iguales:

11Ø1ØØ11Ø111B

11Ø1$ØØ11$Ø111B

Los límites de representación en cada base de las constantes numéricas son:

Hexadecimal, de ØH a ØFFFFH

Decimal, de ØD a 65535D

Octal, de ØQ a 177777Q

Binaria, de ØB a 1111$1111$1111$1111B

I-3-2. DECLARACIONES.

La sentencia DECLARE se utiliza en PL/M para declarar las variables que se van a utilizar a lo largo del programa. Esta es la forma que tiene

el compilador de conocer la existencia de cada variable y su emplazamiento dentro del programa.

I-3-2-1. TIPOS DE DECLARACIONES.

Existen en PL/M dos tipos de declaraciones, BYTE y ADDRESS. En el caso de que la variable a declarar tenga una longitud de un byte se utiliza el tipo BYTE. Si la variable tiene una longitud de dos bytes (16 bits), se utiliza el tipo ADDRESS.

El tipo de cada variable debe de ser formalmente declarado junto con la sentencia DECLARE.

La sentencia DECLARE para una variable o grupo de variables debe de empezar con la palabra DECLARE. A continuación debe de colocarse el identificador de la variable. En el caso de que se quiera declarar más de una variable en una misma linea, la lista de identificadores debe de ir entre paréntesis y separados por comas. Seguidamente irá el tipo de declaración, BYTE ó ADDRESS.

Analicemos las siguientes declaraciones:

```
DECLARE LECTURA1 BYTE;
DECLARE PUNTERO ADDRESS;
DECLARE (ENTRADA, SALIDA) BYTE;
```

La primera de éstas sentencias DECLARE introduce el identificador LECTURA1 al que se le

asigna el tipo BYTE. Cuando el compilador procese ésta sentencia de declaración asignará un espacio de un byte (8 bits) para almacenar dicha variable.

La segunda de éstas sentencias de declaración introduce el identificador PUNTERO al que se le asigna el tipo ADDRESS. Cuando el compilador procese ésta sentencia de declaración asignará un espacio de dos bytes (16 bits) para almacenar la variable identificada como PUNTERO.

La sentencia,

DECLARE (ENTRADA, SALIDA) BYTE;

es equivalente a las sentencias siguientes:

DECLARE ENTRADA BYTE;
DECLARE SALIDA BYTE;

Una norma muy importante a tener en cuenta en el lenguaje PL/M-8Ø es que al acabar cada linea de programa debe de cerrarse ésta mediante un punto y coma. En caso contrario, cuando se lleve a cabo la compilación, se producirá un mensaje de error.

I-3-3. ARRAYS.

Frecuentemente es deseable utilizar un único identificador para hacer referencia a un grupo de variables, y distinguir unas de otras mediante

subíndices. A éste grupo de variables se le denomina un "array".

I-3-3-1. DECLARACIONES DE ARRAYS.

Un array es declarado mediante la utilización de un "especificador de dimensión". El especificador de dimensión es un número constante encerrado entre paréntesis. El valor de la constante especifica el número de elementos del array.

Por ejemplo:

```
DECLARE TABLA (1ØØ) BYTE;
```

TABLA especifica a un array de 1ØØ elementos, cada uno del tipo BYTE. A cada elemento del array se le asignará un espacio de un byte.

La declaración:

```
DECLARE (INDEX, DATO$BIN, DIRE) (1ØØ) BYTE;
```

es equivalente a las siguientes sentencias:

```
DECLARE INDEX (1ØØ) BYTE;
DECLARE DATO$BIN (1ØØ) BYTE;
DECLARE DIRE (1ØØ) BYTE;
```

INDEX, DATO$BIN, DIRE especificará cada uno de ellos a un array de 1ØØ elementos del tipo BYTE.

I-3-3-2. VARIABLES SUBINDICADAS.

Para hacer referencia a un único elemen-

to de un array (previamente declarado), se utiliza el identificador del array seguido por el subíndice entre paréntesis, la cual es llamada una "variable subíndicada".

Por ejemplo:

DECLARE ITEM (1ØØ) ADDRESS;

Esta declaración hace referencia al array ITEM de 1ØØ elementos del tipo ADDRESS. Para hacer referencia a cada uno de los elementos de éste array utilizamos los subíndices: ITEM (Ø), ITEM (1), ITEM (2), ...,ITEM (99). Es de resaltar que el primer elemento de un array tiene el subíndice Ø.

Si quisieramos, por ejemplo, sumar el tercer elemento con el cuarto elemento de un array y almacenar el resultado en la quinta posición de ese mismo array, realizariamos la siguiente sentencia PL/M:

ITEM (4) = ITEM (2) + ITEM (3);

Analicemos seguidamente la expresión:

TEMP1 (MASK (4) + 7)

Esta hace referencia a un elemento del array TEMP1 el cual depende de la expresión MASK (4) + 7. Suponiendo que MASK (4) contenga el valor 3, entonces MASK (4) + 7 es igual a 1Ø.

I-3-4. ESTRUCTURAS. (STRUCTURE).

Al igual que un array dispone de un identificador para hacer referencia a un grupo de elementos del mismo tipo, una estructura dispone de un identificador para hacer referencia a un grupo de estructuras membreteadas las cuales pueden tener diferentes tipos.

Veamos el siguiente ejemplo de una declaración de estructura:

```
DECLARE CLIMA STRUCTURE (TEMP BYTE,
ALTITUD ADDRESS);
```

Esta declaración contiene dos variables. Una variable del tipo BYTE y otra del tipo ADDRESS, ambas asociadas con el identificador CLIMA. Una vez hecha ésta declaración, se puede hacer referencia al primer escalar de la forma CLIMA. TEMP y al segundo mediante la forma CLIMA. ALTITUD, siendo éstos los dos miembros de la estructura.

I-3-4-1. ARRAYS DE ESTRUCTURAS.

Una vez vista la forma de declaración de una estructura, pasemos a analizar la siguiente sentencia:

```
DECLARE CLIMA (2Ø) STRUCTURE (TEMP BYTE, ALT
ADDRESS);
```

Esta declara 2Ø estructuras asociadas con el identificador de array CLIMA, distinguidas mediante los subíndices Ø al 19. Cada una de éstas estructuras contienen dos miembros, uno del tipo BYTE, TEMP, y otro del tipo ADDRESS, ALT. Esto hará que se asignen 2Ø posiciones para variables tipo BYTE y otras 2Ø para las variables del tipo ADDRESS.

Para hacer referencia, por ejemplo, al tercer elemento de CLIMA.TEMP, escribiremos:

```
CLIMA(2).TEMP
```

La forma de almacenamiento en memoria es la siguiente:

```
CLIMA (Ø).TEMP, CLIMA (Ø).ALT,.....
.....,CLIMA (19).TEMP,CLIMA (19).ALT
```

I-3-4-2. ESTRUCTURAS QUE CONTIENEN ARRAYS.

Un array puede ser utilizado como miembro de una estructura. Veamos el siguiente ejemplo

```
DECLARE DATO$BIN STRUCTURE (PRIMERO (1Ø) BYTE,
SEGUNDO (1Ø) BYTE, INDICE ADDRESS);
```

Esta estructura está compuesta por los siguientes elementos: dos arrays del tipo BYTE de 1Ø elementos cada uno, y la variable INDICE del tipo ADDRESS.

Para hacer referencia al quinto elemento

del array DATO$BIN.SEGUNDO, escribiremos:

DATO$BIN.SEGUNDO (4)

I-3-4-3. ARRAYS DE ESTRUCTURAS CON ARRAY DENTRO DE LA ESTRUCTURA.

Hasta ahora hemos visto como un array puede estar formado por estructuras y también como una estructura puede tener arrays como miembros.

Para poder hacer una combinación de éstas dos construcciones debemos de escribir una sentencia DECLARE como la siguiente:

```
DECLARE X (1Ø) STRUCTURE (Y (1Ø) BYTE);
```

Conseguimos de ésta forma una estructura matricial de 1Ø filas por 1Ø columnas. Para hacer referencia a un elemento determinado, escribiríamos:

X (4).Y (6)

La forma de almacenamiento en memoria es la siguiente:

X (Ø).Y (Ø), X (Ø).Y (1), X (Ø).Y (2), ...
..... , X (9).Y (Ø), ... , X (9).Y(9)

Podemos declarar mediante una sola estructura varias matrices, con la condición de que todas tendrán el mismo número de filas dadas por la dimensión de la estructura. Por ejemplo:

DECLARE A(1Ø) STRUCTURE (B(5) BYTE, C(6) BYTE);

El orden de posicionamiento en memoria será el siguiente:

A(Ø).B(Ø), A(Ø).B(1),.....
..... , A(Ø).B(4), A(Ø).C(Ø), ...
..... , A(Ø).C(5), A(1).B(Ø), ...
..... , A(9).C(9)

I-3-5. VARIABLES CON REFERENCIA A BASE.

La referencia a base nos permite situar a una variable en una dirección determinada. Esta dirección está contenida a su vez en otra variable.

Veamos el siguiente ejemplo:

```
DECLARE INDICE ADDRESS;
DECLARE DATO BASED INDICE BYTE;
```

El identificador de la base será una variable, tipo ADDRESS, que contendrá la dirección donde se encuentra localizada la variable.

En el ejemplo, la variable INDICE contendrá la dirección de la variable DATO.

Para la utilización de una referencia a base deben de tenerse en cuenta las siguientes normas:

- Debe de declararse primero la variable que contendrá a la base (INDICE).
- El identificador de la base (INDICE) no puede ser subindicado.

- La variable baseada (DATO) no puede tener ni atributos ni inicializaciones.

I-3-6. FORMA DE ALMACENAMIENTO.

El PL/M asegura que las variables serán almacenadas contiguamente en las siguientes situaciones:

- Los elementos de un array son almacenados contiguamente con el elemento Ø en la posición de memoria más baja y el último elemento en la posición de memoria más alta.
- Los miembros de una estructura son almacenados contiguamente, en el orden en el que están especificados.
- Las variables no baseadas que han sido declaradas dentro de un paréntesis. Se almacenarán contiguamente siguiendo el orden en que fueron especificadas.

I-4. EXPRESIONES Y ASIGNACIONES.

Una expresión en PL/M consiste en la combinación de operandos con operadores aritméticos, lógicos o de relación. Veamos seguidamente algunos ejemplos:

A + B

A + B - C

A * B + C / D

A * (B + C) - (D - E) / F

donde +, -, *, / son los operadores aritméticos de adición, sustracción, multiplicación y división respectivamente. A, B, C, D, E, F representan los operandos. Los paréntesis se utilizan para agrupar operaciones y operandos.

I-4-1. OPERANDOS.

Los operandos constituyen la parte más importante de las expresiones. Un operando debe de ser un ente que especifique un determinado valor cuando el programa se ejecute.

Las constantes numéricas y determinadas variables pueden aparecer como operandos en las expresiones.

I-4-1-1. CONSTANTES.

Cualquier constante numérica puede ser utilizada como un operando en una expresión. Una

constante numérica es implicitamente del tipo BYTE si ésta no es mayor a 255. En el caso de que fuera mayor a ésta cantidad, implicitamente sería del tipo ADDRESS.

Una constante de 'string' que no contenga más de dos caracteres puede ser utilizada como un operando. Si ésta tiene unicamente un caracter, se la considerará como una constante numérica del tipo BYTE al corresponderle ocho bits en el código ASCII. En el caso de que tenga dos caracteres, se la considerará como una constante numérica del tipo ADDRESS al corresponderle dieciseis bits en el código ASCII. Recordemos que el código ASCII asigna ocho bits a cada caracter alfanumérico.

I-4-1-2. REFERENCIA A UNA DIRECCION.

Es una forma muy simple mediante la cual sin necesidad de conocer o de calcular analíticamente la posición de memoria de una variable, saber cual es su dirección en memoria. Esto se consigue añadiendo un punto a la izquierda del identificador de la variable cuya dirección queremos conocer. Por ejemplo:

.LECTURA3

I-4-2. OPERADORES ARITMETICOS.

Existen el PL/M cinco operadores aritméticos

+ - * / MOD

Estos operadores realizarán operaciones con números binarios enteros y positivos.

I-4-2-1. OPERADORES ARITMETICOS "+" Y "-".

Los operadores + y - realizan las operaciones de adición y sustracción respectivamente. Si ambos operandos son del mismo tipo, el resultado será también del mismo tipo. Si por ejemplo uno de los operandos es del tipo ADDRESS y el otro es del tipo BYTE, el operando tipo BYTE será incrementado añadiéndosele ocho bits cero a la izquierda del bit más significativo. La operación se realizará entonces con dieciseis bits y el resultado será del tipo ADDRESS.

I-4-2-2. OPERADORES ARITMETICOS "*" Y "/".

Los operadores aritméticos * y / realizan las operaciones binarias sin signo de multiplicación y división respectivamente con operandos que sean tanto del tipo BYTE como del tipo ADDRESS. El resultado de dichas operaciones será siempre del tipo ADDRESS. Si durante la ejecución se produce un 'overflow', el resultado será indefinido.

I-4-2-3. OPERADOR "MOD".

Ejecuta la división entre dos operandos pero en lugar de dar el resultado de ésta, proporciona el resto de la división.

I-4-3. OPERADORES LOGICOS.

Existen en PL/M cuatro operadores lógicos:

NOT AND OR XOR

Estos operadores lógicos realizan las operaciones en paralelo con ocho ó dieciseis bits.

El operador NOT es unitario, es decir, solo se puede utilizar con un operando. El resultado de una operación lógica NOT es la de complementar cada bit del operando.

El resto de las operaciones requiere la utilización de dos operandos. Las operaciones lógicas que realizan son: "and", "or" y "or-exclusiva".

Si ambos operadores son del tipo BYTE, el resultado será también del tipo BYTE. En caso contrario el resultado será del tipo ADDRESS.

I-4-4. ORDEN DE EVALUACION DE LOS OPERADORES.

En todos los lenguajes de alto nivel existe una preferencia de unos operadores sobre otros, de forma que en una expresión que contenga varios operadores distintos será analizada en un determinado orden por el compilador.

El orden de preferencia es el siguiente:

1. signo '-' (complemento a dos)
2. * / MOD
3. + -
4. NOT
5. AND
6. OR XOR

Existe la posibilidad de alterar el orden de evaluación utilizando los paréntesis, con los cuales daremos preferencia a una determinada operación.

I-4-5. ORERADORES DE RELACION.

Su utilidad estriba en la de poder comparar variables, constantes y expresiones entre sí. En PL/M-8Ø se pueden utilizar los siguientes:

"Menor que"
"Mayor que"
"Menor ó igual que"
"Mayor ó igual que"
"Distinto que"
"Igual que"

Los operandos pueden ser de tipo BYTE ó ADDRESS. El resultado de éste tipo de operación, siempre será del tipo BYTE, de forma que los ocho bits estarán a 1 cuando la relación es "verdadera"

ó a Ø cuando la relación es "falsa".

I-4-6. SENTENCIAS DE ASIGNACION.

Se utilizan para transferir el resultado de una operación ó el valor de una variable a otra variable que aparecerá a la izquierda del símbolo de asignación "=". La sintaxis general es la siguiente:

variable = expresión ó variable;

Por ejemplo:

MASK = ENTRADA3 AND 4FH;

Se pueden efectuar también asignaciones factorizadas, es decir, asignar el valor de una expresión ó variable a varias variable al mismo tiempo.

Veamos el siguiente ejemplo:

DATO, LECTURA$1, MASK = ENTRADA3 AND Ø6H;

Además de éste tipo de asignación, existe otra llamada "asignación incluida", cuya propiedad consiste en poder asignar resultados parciales de una expresión a una variable. Esto supone que podemos ahorrar tiempo a la hora de realizar operaciones que pueden repetirse con frecuencia a lo largo del programa.

Por ejemplo, tenemos la siguiente expresión:

PUNTUACION = (PRIMERO * A) / MULT + DATO;

y más adelante encontramos una expresión como:

PUNTUACION$2 = PRIMERO * A + B;

Para realizar una "asignación incluida" variamos la primera expresión de la siguiente manera:

PUNTUACION = (PARCIAL := PRIMERO * A) / MULT + DATO;

en la segunda expresión pondremos:

PUNTUACION$2 = PARCIAL + B;

En las sentencias de asignación hay que tener en cuenta que el tipo de las variables utilizadas sean iguales. En el caso de que el valor de la expresión no sea el mismo tipo que la variable, el resultado de la expresión es tratado de forma que tenga el mismo tipo de la variable.

Si la variable es del tipo ADDRESS y el resultado es del tipo BYTE, se añadirán ocho ceros a la izquierda del bit más significativo de la variable.

Si por el contrario el resultado es del tipo ADDRESS y la variable es del tipo BYTE, se perderán los ocho bits más significativos.

I-5. SENTENCIAS DE CONTROL DE EJECUCION.

Estas sentencias PL/M afectan a la secuencia de ejecucion del programa.

I-5-1. SENTENCIAS DO Y END.

En el lenguaje PL/M-80 existen cuatro clases diferentes de sentencias DO, las cuales serán objeto de un estudio más exaustivo a lo largo de éste capítulo. Estas sentencias DO son:

- Sentencias DO simple.
- Sentencias DO WHILE.
- Sentencias DO iterativas.
- Sentencias DO CASE.

En cuanto a la sentencia END, tendrá la forma:

```
END ( etiqueta );
```

donde la etiqueta es opcional, y en el caso de usarse, debe de ser la misma que se ha utilizado en el comienzo de la sentencia DO.

Por ejemplo, la sentencia

```
END COMPARACION;
```

debe de ser utilizada para cerrar un bloque que ha sido empezado con un DO que lleva la etiqueta COMPARACION.

La etiqueta en una sentencia END no tiene

ningún efecto sobre el programa y se utiliza para una mejor comprensión de los bloques que componen el programa y como ayuda en la depuración del mismo. El compilador detectará una etiqueta incorrecta y alertará al programador de la existencia de un error en la estructura del programa.

I-5-1-1. DO SIMPLE.

Un DO simple comienza con la sentencia DO y tiene la forma siguiente:

```
DO;
   sentencia-1;
   sentencia-2;
   ......
   sentencia-n;
END;
```

Veamos el siguiente ejemplo:

```
DO;
   MASK$5 = TEMP1 AND ØFH;
   INDICE = INDICE + 1;
END;
```

Dentro de un bloque DO puede haber cualquier tipo de sentencia PL/M, incluyendo sentencias ejecutables y declaraciones, pero éstas últimas con la restricción de que las variables que sean declaradas solo podrán utilizarse dentro del

bloque DO.

Las secuencias ejecutables que pudieran haber dentro de un bloque DO serán ejecutadas secuencialmente como si éstas no estubieran dentro de un bloque DO - END.

Los bloques DO pueden estar anidados unos con otros tal y como sigue:

```
DO;
   sentencia-1;
   sentencia-2;
   DO;
      sentencia-a;
      sentencia-b;
      sentencia-c;
   END;
   sentencia-3;
   sentencia-4;
END;
```

La primera sentencia DO y la segunda sentencia END encierran un bloque DO. La segunda sentencia DO y la primera sentencia END encierran otro bloque DO dentro del primero.

I-5-1-2. BLOQUE DO WHILE.

Se utiliza éste bloque para realizar iterativamente una serie de sentencias mientras no

se cumpla una determinada condición. La forma general es:

```
DO WHILE expresión;
   sentencia-1;
   sentencia-2;
   .....
   sentencia-n;
END;
```

El resultado de éstas sentencias es el siguiente:

Primero se evalúa la expresión que se encuentra a continuación del WHILE. Si ésta resulta ser 'verdadera' se ejecutan las sentencias contenidas en el bloque hasta que se llegue al END. Si la expresión resulta ser 'falsa' se ejecutarían las sentencias siguientes al END del bloque, continuando la ejecución normal del programa.

Consideremos el siguiente ejemplo:

```
INDICE = Ø;
DO WHILE INDICE <5;
   INDICE = INDICE + 1;
END;
```

La sentencia INDICE = INDICE + 1 será ejecutada siete veces. Cuando la secuencia del programa salga fuera del bloque el valor de INDI-

CE será de seis.

Por último decir que dentro de un bloque DO WHILE se pueden anidar otros bloques DO WHILE.

I-5-1-3. BLOQUES DO ITERATIVOS.

Un bloque DO iterativo comienza con una sentencia DO y ejecuta las sentencias existentes dentro del bloque repetidamente, un determinado número de veces.

La forma más simple de DO iterativo es la siguiente:

```
DO indice = expresión inicial TO expresión límite;
   sentencia-1;
   sentencia-2;
   .....
   sentencia-n;
END;
```

donde "indice" hace referencia a una variable que no puede ser subindicada.

El bloque es ejecutado de la siguiente forma:

1. Se evalúa la expresión inicial y se asigna su valor al índice.
2. Se evalúa la expresión límite y se compara su valor con el índice. Si el índice fuera mayor, el control pasaría a la sentencia que hubiera des-

pués del END, que indica el final del bloque iterativo.

3. Si el índice fuera menor se ejecutarán todas las sentencias incluidas en el bloque, incrementándose seguidamente el valor del índice en una unidad y volviendo al paso 2.

En la utilización de los bloques DO iterativos debemos de tener en cuenta:

Los valores de la expresión inicial y de la expresión límite serán convertidos al mismo tipo que la variable índice.

La expresión inicial es evaluada una sola vez. La expresión límite es evaluada cada vez que el bloque se repita.

Si el valor del índice es mayor a 255 (para un índice tipo BYTE) ó a 66635 (para un índice tipo ADDRESS) la estructura repetitiva no llegará a pararse, con lo que no se podrá salir nunca del bloque DO iterativo.

Hay que tener cuidado de no colocar dentro de un DO iterativo un sentencia de control de ejecución ya que ésta puede hacer que la secuencia del programa se salga del bloque prematuramente sin tener en cuenta los valores que han alcanzado el índice ó la expresión límite.

Una sentencia de éste tipo puede ser GOTO, la cual estudiaremos más adelante.

Dentro de éstos bloque se pueden anidar otros bloques DO iterativos.

Veamos a continuación un ejemplo de utilización de DO iterativo:

```
INDICE = Ø;
DO I = 1 TO 1Ø;
   INDICE = INDICE + I;
END;
```

Tanto INDICE como I son variables que han debido de ser previamente declaradas. La sentencia de asignación INDICE = INDICE + I será ejecutada un total de 1Ø veces, y cada una con un nuevo valor de I. El resultado será la suma desde el 1 hasta el 1Ø inclusive. Este resultado (que es 55) será el valor de índice.

En los bloques DO iterativos el valor del indice puede ser incrementado por el número que se desee. La forma general es:

```
DO indice = exp. inicial TO exp. límite BY exp.
pasos (step);
      sentencia-1;
      sentencia-2;
      .....
      sentencia-n;
END;
```

En éste caso, la variable índice será incrementada por el valor de la expresión pasos (step), en lugar de por 1, cada vez que se llegue al END.

Un ejemplo de ésto sería:

```
SUM = Ø;
DO I = 2 TO (2 * N) BY 2;
   SUM = SUM + I;
END;
```

Una vez ejecutadas las sentencias anteriores, el resultado será la suma de los N primeros números pares.

Cuando se utilicen los bloques DO iterativos en su forma de pasos definidos (step) hay que tener presente:

Como el PL/M utiliza solo números binarios enteros sin signo, no será posible realizar un paso (step) negativo.

En PL/M no es posible realizar pasos nacia abajo (cuenta atrás) ya que la expresión límite es siempre mayor que la expresión inicial.

El valor de la expresión pasos (step) es convertida al mismo tipo que el indice.

I-5-1-4. BLOQUES DO CASE.

Mediante éste tipo de bloques se puede ejecutar selectivamente cualquier sentencia en concreto, ó un bloque anidado dentro de éste, dependiendo del valor de la variable ó el resultado de una expresión.

La sintáxis general es como sigue:

```
DO CASE expresión;
        DO; sentencia-Ø.1;
            .....
            sentencia-Ø.n;
        END;
        DO;
            sentencia-1.1;
            .....
            sentencia-1.n;
        END;
        .........
        DO;
            sentencia-n.1;
            .....
            sentencia-n.n;
        END;
END;
```

En primer lugar se evalúa la expresión y dependiendo del valor que tome ésta se ejecuta un bloque u otro. Si por ejemplo, el valor de la expresión fuese 1, se ejecutaría el bloque cuya primera linea contiene la sentencia-1.1. Hay que tener en cuenta que la expresión deberá tomar valores entre Ø y n-1, y ninguno otro fuera de éste intervalo.

Una vez seleccionado y ejecutado el bloque ó sentencia, el control pasará a la sentencia siguiente al END, del bloque DO CASE.

Un ejemplo de bloque DO CASE podría ser el siguiente:

```
DO CASE PUNTERO;
        MASK = LECTURA$1 OR 48H;
        TEMP3 = ACTUAL AND ENTRADA1;
        INDICE = INDICE + 1;
END;
```

Cuando la ejecución del bloque DO CASE comienza, el valor de la variable PUNTERO debe de estar comprendido entre Ø y 2. En caso contrario, el resultado sería totalmente indefinido. Si por ejemplo, el valor de PUNTERO fuera 1, se ejecutaría la sentencia TEMP3 = ACTUAL AND ENTRADA1. Una vez ejecutada ésta sentencia, se saldrá del bloque DO CASE.

Hay que puntualizar que dentro de un bloque DO CASE se pueden anidar bloques DO iterativos.

Analicemos el ejemplo siguiente:

```
DO CASE PUNTERO;
        MASK = LECTURA1 AND ØEH;
        ;
        ;
        DO;
           X = X + 1;
           Y = Y + 5;
        END;
        ;
        INDICE = INDICE + 1;
END;
```

Obsérvese que dentro del boque DO CASE se encuentran lineas que están en blanco y que corresponden a los valores 1, 2 y 4. Si la variable PUNTERO tomara alguno de éstos valores, se saldría del bloque DO CASE sin haber realizado ninguna sentencia.

I-5-2. LA SENTENCIA IF.

Esta sentencia nos proporciona la posibilidad de elegir condicionalmente la ejecución de una sentencia.

La forma general es:

```
IF condición THEN sentencia-1;
(ELSE sentencia-2);
```

En primer lugar se evalúa la condición y si ésta es 'verdadera' entonces se ejecuta la sentencia-1. En el caso de que sea 'falsa' se ejecuta la sentencia-2. En la forma general hemos puesto a ELSE entre paréntesis denotando así la opcionalidad de su inclusión en el programa. Teniendo en cuenta ésto podemos poner:

```
IF condición THEN sentenci-1;
```

Consideremos a continuación el siguiente fragmento de programa:

```
IF TEMP2 = Ø THEN DATO = 8D;
ELSE DATO = 16D;
```

Vemos que a DATO se le asigna ó bien el valor 8D ó bien el valor 16D dependiendo del valor que tome TEMP2. Si TEMP2 = Ø a DATO se le asignará el valor 8D. Por el contrario si no se cumple la condición TEMP2 = Ø a DATO se le asignará el valor 16D.

Una característica importante del PL/M que aumenta la potencia de ésta sentencia, es la posibilidad de poder emplear bloques DO anidados dentro de la estructura IF.

La sintáxis general es:

```
IF condición THEN
   DO;
      sentencia-1;
      .....
      sentencia-n;
   END;
ELSE
    DO;
       sentencia-1;
       .....
       sentencia-n;
    END;
```

Veamos el siguiente ejemplo:

```
IF INDICE = 8D THEN
    DO;
       LECTURA1 = MASK AND ØBH;
       DATO = 9D;
    END;
ELSE
    DO;
       LECTURA1 = MASK OR 9EH;
```

```
        DATO = TEMP9 + 19H;
        SUM = SUM + 1;
    END;
```

En el lenguaje PL/M-8Ø se permiten también anidamientos de estructuras IF unas dentro de otras. Pero hay que tener en cuenta la siguiente norma, y es que si una sentencia IF es anidada dentro de la parte THEN de otra sentencia IF, entonces el IF exterior no puede tener la opción ELSE. Para clarificar conceptos, supongamos el siguiente fragmento de un programa:

```
IF condición-1 THEN
   IF condición-2 THEN sentencia-1;
ELSE sentencia-2;
```

Si no se tubiera en cuenta ésta norma, éste segmento de programa resultaría ambiguo, porque ¿a que sentencia IF pertenecerá la parte ELSE?. Por lo tanto se toma como norma que éste pertenece al IF anidado.

Aún así, existe la posibilidad de hacer que el ELSE corresponda al IF exterior. Veamos éstas sentencias:

```
IF condición-1 THEN
   DO;
      IF condición-2 THEN sentencia-1;
   END;
ELSE sentencia-2;
```

I-5-3. ETIQUETAS.

Una etiqueta es un identificador que está asociado con una sentencia ejecutable del programa y hace referencia a ella. Normalmente las etiquetas no hay que declararlas ya que éstas se declaran de forma implícita al aparecer a la izquierda de una sentencia ejecutable, que llamaremos sentencia etiquetada. Para poner una etiqueta basta colocar el identificador seguido de dos puntos y a continuación y a la derecha de éstos la sentencia etiquetada. Veamos por ejemplo:

```
DIGI1: CUENTA = MAYOR + INDICE;
```

donde DIGI1 es el identificador de la etiqueta. El identificador de la etiqueta puede tener hasta un máximo de 31 caracteres. Además de éste tipo de declaración, existe la posibilidad de que las etiquetas sean declaradas explícitamente, mediante la sentencia DECLARE, como si fuera una variable.

La sintáxis general es la siguiente:

```
DECLARE identificador LABEL;
```

El identificador tiene las mismas limitaciones que para las variables. LABEL es el indicativo de que éste tipo de declaración es para la etiquetas.

I-5-4. LA SENTENCIA GOTO.

Esta sentencia se utiliza para alterar el orden secuencial de ejecución de un programa, transfiriendo el control a una sentencia etiquetada. En PL/M-8Ø, ésta función se realiza mediante la sentencia GOTO.

La sintáxis general es:

```
GOTO etiqueta;
```

Hay que tener en cuenta que la aparición de una etiqueta en una sentencia GOTO no es una definición de la misma, sino una referencia a la propia etiqueta. Veamos el siguiente ejemplo:

```
GOTO SCAN;
```

Cuando en el programa se ejecute ésta sentencia se pasaría el control a la sentencia etiquetada SCAN.

Como norma general, en la utilización de sentencias GOTO, no se puede transferir el control desde el exterior al interior de un bloque DO iterativo ó a un bloque DO CASE.

I-6. SENTENCIAS DECLARE.

Como hemos visto anteriormente, una variable debe de ser declarada antes de poder hacer referencia a ella por su identificador. Esto se lleva a cabo mediante la sentencia DECLARE. En apartados anteriores se han realizado ejemplos de sentencias DECLARE, pero sin describir todos los tipos de informaciones que pueden ser incluidas dentro de una de éstas sentencias. En sucesivos apartados se dará una información más completa sobre la utilización de la sentencia DECLARE.

La etiquetas también pueden ser declaradas mediante la sentencia DECLARE, sin embargo, ésto no es normalmente necesario tal y como vimos en el capítulo anterior.

Los procedimientos (PROCEDURE) deben de ser también declarados, sin embargo, la declaración de los procedimientos será tratada en un capítulo aparte.

I-6-1. ATRIBUTOS.

A las variables en PL/M se le pueden asignar unas palabras claves que les confieren unas determinadas cualidades. Tres son los atributos que es posible asignar a una variable; PUBLIC, EXTERNAL y AT.

I-6-1-1. ATRIBUTOS PUBLIC Y EXTERNAL.

Estos dos atributos se emplean conjuntamente, si bien, no en la misma sentencia. Mediante éstos atributos se puede extender el alcance de un variable a otros módulos de programa. El atributo PUBLIC se utiliza para definir a una variable como común a todos los bloques ó subprogramas que se ejecutan junto con éste. El atributo EXTERNAL se emplea para definir en qué módulos ó bloques se puede utilizar la variable declarada como PUBLIC. Por ejemplo, en un módulo podríamos tener:

```
DECLARE DELTA ADDRESS PUBLIC;
```

y en el otro ó los otros donde queramos utilizar ésta variable pondremos:

```
DECLARE DELTA ADDRESS EXTERNAL;
```

de forma que tanto en un módulo como en el otro se trata de la misma variable, y en ambos podrá ser utilizada y alterada.

Para la utilización de los atributos PUBLIC Y EXTERNAL hay que tener en cuenta las siguientes normas:

1. Los atributos PUBLIC y EXTERNAL, solo podrán ser utilizados en sentencias DECLARE que estén en el nivel más alto

de un módulo ó bloque de programa.

2. No pueden usarse éstos atributos con variables con base. Sin embargo, la base si puede usarlos.
3. El atributo EXTERNAL no se puede utilizar si previamente no ha sido declarada como PUBLIC en otro módulo.
4. El atributo EXTERNAL no se puede usar con el atributo AT ó con una inicialización. Por el contrario en la declaración definitoria de la variable, es decir, la sentencia que lleva el atributo PUBLIC, si puede contener el atributo AT ó una inicialización.
5. Cuando una variable es declarada EXTERNAL, debe de ser del mismo tipo que donde se declara como PUBLIC.
6. Cuando un array es declarado EXTERNAL, éste debe de tener el mismo número de elementos y del mismo tipo que donde se ha declarado como PUBLIC.
7. Cuando una estructura,STRUCTURE, es declarada como EXTERNAL, ésta debe de tener el mismo número de miembros que la que está declarada PUBLIC.

Los atributos PUBLIC y EXTERNAL también pueden ser aplicados en la declaración de procedimientos (PROCEDURE), que serán objeto de estudio más adelante.

I-6-1-2. EL ATRIBUTO AT.

El atributo AT presenta la siguiente sintáxis:

AT (expresión restringida)

Una expresión restringida es una secuencia de uno ó más operandos separados por operadores, y que darán como resultado la dirección que va a tener esa variable. El término expresión restringida viene dado por el hecho de que aún pudiendo ser una expresión normal, tiene una serie de limitaciones y reglas a cumplir.

Las normas a que debe de atenerse el atributo AT son las siguientes:

1. El primer operando puede ser unicamente una constante numerica ó una referencia a una dirección. En el caso de que fuera una referencia a una dirección, debe de referirse a una variable que ya haya sido declarada. Si una referencia a una dirección contiene una expresión subindicada, ésta ex

presión no puede contener ningún operando excepto que sean constantes numéricas ó los operadores aritméticos '+' y '-'.

2. En el caso de que un operando se encuentre a continuación del primero, éste solo puede ser una constante numérica.
3. Solo se permiten los operadores aritméticos '+' y '-'.

Veamos a continuación algunos ejemplos:

```
AT (2ØØØH)
AT (.DIGI$4)
AT (.DIGI$1 + 4Ø)
AT (.DATO (INDICE + 1) + LECTURA - 5)
```

En el último ejemplo, INDICE y LECTURA representan constantes numéricas que han sido previamente declaradas con la declaración "LITERARY" (que estudiaremos más adelante). El compilador reemplaza éstos nombres por las constantes numéricas declaradas, de éste modo, se satisfacen las restricciones dadas anteriormente.

El resultado del atributo AT sobre una variable es localizarla en la dirección especificada por la expresión restringida. La variable direccionada es el primer escalar de la declaración.

Los otros escalares que puedan haber en la declaración serán direccionados a continuación del primero.

Consideremos el siguiente ejemplo:

```
DECLARE (DATO$BIN, DATO$ASC, SUMA) BYTE AT (.LIM);
```

La variable DATO$BIN es localizada en la dirección de memoria dada por LIM. Las variables DATO$ASC y SUMA son almacenadas en los dos bytes siguientes a DATO$BIN.

La declaración:

```
DECLARE SELET (1Ø) STRUCTURE (
        X (5) BYTE,
        Y (9) BYTE,
        Z (3) BYTE) AT (.ORIGEN);
```

direcciona al escalar SELET(Ø).X(Ø) en la posición de memoria previamente declarada como ORIGEN. El resto de los miembros de la estructura se situaran en las sucesivas direcciones de memoria.

I-6-2. INICIALIZACIONES.

Las inicializaciones se utilizan para proporcionar valores a las variables en tiempo de compilación. Existen dos tipos de inicializaciones, INITIAL Y DATA.

INITIAL lleva a cabo la inicialización durante la compilación, de tal manera que la variable así inicializada puede posteriormente cambiar de valor durante la ejecución del programa, como cualquier otra variable.

La inicialización DATA no sólo inicializa una variable, sino que también ésta es almacenada con el programa en código. Por lo tanto, la inicialización DATA tiene la misma función que INITIAL, pero a diferencia de ésta, el valor de las variables no podrán cambiarse durante la ejecución del programa.

Cuando apliquemos alguna de éstas inicializaciones deben de observarse las siguientes reglas:

1. INITIAL y DATA no pueden utilizarse juntas en la misma declaración.
2. INITIAL sólo puede utilizarse en la sentencia DECLARE más extrema, es decir, al principio. Sin embargo, DATA puede declararse en cualquier nivel de la zona dedicada a declaraciones.
3. No se pueden utilizar inicializaciones con variables baseadas ó que tengan el atributo EXTERNAL.
4. Las inicializaciones pueden usarse con el atributo AT. Sin embargo, si ocurriera que debido a éste atributo, la variable estu-

viera localizada fuera de la zona de memoria del bloque o módulo en que hacemos la declaración, entonces emplear la inicializaciones sería prohibitivo.

5. La inicialización DATA bajo ningún concepto podrá usarse conjuntamente con el atributo AT.

I-6-2-1. LA INICIALIZACION INITIAL.

La inicialización INITIAL tiene la siguiente forma:

INITIAL (lista de valores)

donde la lista de valores es una secuencia de uno ó más valores separados por comas. Cada valor puede ser ó una expresión restringida con un AT ó un string (cadena de caracteres).

La declaración:

DECLARE ESTADO$PREVIO BYTE INITIAL (5ØH);

realiza la declaración del escalar ESTADO$PREVIO y que es inicializado con el valor 5ØH.

La declaración:

DECLARE (INDICE, CONTADOR, P) ADDRESS INITIAL (2ØØØH, Ø, 8F4H);

realiza la declaración del tipo ADDRESS de los escalares INDICE, CONTADOR y P e inicializa INDICE con

el valor 2ØØØH, CONTADOR con el valor Ø y P con el valor 8F4H.

La declaración:

```
DECLARE ITEM (5) BYTE INITIAL (2,4,7,9,Ø);
```

realiza la declaración del tipo BYTE del array ITEM de cinco elementos y dichos elementos son inicializados respectivamente con los valores 2,4,7,9 y Ø.

La declaración:

```
DECLARE PUNTERO STRUCTURE (
DIREC ADDRESS,
VALOR (3) BYTE,
CONTE BYTE) INITIAL (4Ø, 5, 9, 12, Ø);
```

realiza la declaración de la estructura PUNTERO y que ha sidos inicializada de la siguiente forma:

```
PUNTERO.DIREC a 4Ø
PUNTERO.VALOR (Ø) a 5
PUNTERO.VALOR (1) a 9
PUNTERO.VALOR (2) a 12
PUNTERO.CONTE a Ø
```

Cuando se realiza una expresión restringida para inicializar un escalar del tipo BYTE, éste valor no debe de ser mayor a 255. En el caso de utilizarse para inicializar un escalar del tipo ADDRESS éste valor no debe de ser mayor a 65535.

La declaración:

DECLARE CONT (4) BYTE INITIAL ('AQUI');

se trata de la declaración del array CONT que tiene cuatro elementos del tipo BYTE y que han sido inicializados mediante un string (cadena de caracteres). Este string es: 'AQUI'.

Por lo tanto, CONT (Ø) será inicializado con el valor en código ASCII del caracter A, CONT(1) será inicializado con el valor en código ASCII del caracter Q, y así sucesivamente.

La declaración:

DECLARE RESP (5Ø) BYTE INITIAL (3, 1, Ø);

realiza la declaración del array RESP formado por cincuenta elementos del tipo BYTE. Los tres primeros elementos del array RESP serán inicializados con los valores 3, 1 y Ø respectivamente. El resto de los elementos del array no serán inicializados.

La lista de constantes de inicialización no puede ser mayor que el número de variables a inicializar. Por ello, cuando se inicializa un array es preciso que el número de elementos del mismo sea igual al número de constantes de la lista de inicialización. Pero si se desconoce la dimensión del array, éste puede ser dimensionado implícitamente mediante el símbolo * entre paréntesis, es decir:

(*)

No se puede dimensionar implicitamente en los siguientes casos:

1. Después de haber cerrado el paréntesis de una lista de identificadores en una declaración factorizada.
2. Para especificar un array el cual es miembro de una estructura.
3. Para especificar un array cuyos elementos son estructuras.

El dimensionamiento implícito (*) tiene la finalidad de que el número de elementos del array sea igual al número de elementos de la lista de constantes de inicialización.

I-6-2-2. LA INICIALIZACION DATA.

La inicialización DATA tiene la forma:

DATA (lista de caracteres)

La inicialización DATA es idéntica a la inicialización INITIAL, excepto por tres diferncias.

Estas son:

1. La inicialización DATA origina el almacenamiento junto al programa en código. Esto quiere decir que la variable inicializada mediante un DATA no podrá cambiar su valor a lo largo del programa.

2. Un identificador declarado con la inicialización DATA nunca podrá aparecer en el lado izquierdo de una sentencia de asignación.
3. Al contrario que en una inicialización INITIAL, la inicialización DATA puede ser declarada en cualquier parte del programa.

I-6-3. MACRO DECLARACION "LITERALLY".

Con la macro declaración LITERALLY se pueden asignar a un sólo identificador una cadena de caractéres (string), constantes e incluso expresiones.

La forma de declaración es la siguiente:

DECLARE identificador LITERALLY 'string';

donde el identificador es un identificador válido en PL/M y el string (cadena de caractéres) es una secuencia arbitraria de caracteres pertenecientes al juego de caracteres PL/M, que no excedan de una longitud de 255.

Veamos los siguientes ejemplos:

```
DECLARE TRUE LITERALLY 'ØFFH', FALSE 'Ø';
DECLARE DATO LITERALLY '2Ø';
```

I-6-4. COMBINACION DE SENTENCIAS DECLARE.

No es necesario realizar sentencias DECLARE separadas para cada variable que se desee declarar.

En lugar de escribir las dos sentencias:

```
DECLARE DATO BYTE INITIAL (15);
DECLARE PUNT ADDRESS;
```

podemos escribir ambas declaraciones en una única sentencia DECLARE como la siguiente:

```
DECLARE DATO BYTE INITIAL (15), PUNT ADDRESS;
```

Esta sentencia de declaración contiene dos elementos de declaración, separados por comas.

I-7. PROCEDIMIENTOS (PROCEDURE).

Un procedimiento es un subprograma ó subrutina el cual puede ser llamado desde otras partes del programa. La llamada al procedimiento (CALL) activa a éste, ejecutándose las sentencias que lo forman de una manera secuencial. Una vez se sale del procedimiento se devuelve el control a la sentencia siguiente al CALL de llamada al procedimiento.

La utilización de procedimientos confieren al lenguaje PL/M una gran potencia ya que permite una programación modular. La utilización de la sentencia PROCEDURE es la base de éste lenguaje para la creación de módulos enlazables con otros módulos.

I-7-1. DECLARACION DE PROCEDIMIENTOS.

Los procedimientos al igual que las variables deben de se declarados. La declaración consta de tres partes: primeramente la sentencia PROCEDURE, a continuación una sentencia ó una secuencia de sentencias que forman lo que se denomina el cuerpo del procedimiento, y por último la sentencia END.

Las tres partes anteriormente mencionadas tendrán la forma siguiente:

```
nombre: PROCEDURE (lista de parámetros) (tipo)
        (atributos);
        sentencia-1;
        sentencia-2;
        .....
        sentencia-n;
END (nombre);
```

Veamos a continuación un ejemplo sencillo de la estructura de un procedimiento:

```
ACTUAL: PROCEDURE;
        TEMP1 = MODULO AND 37H;
        CONT = CONT + 1;
END ACTUAL;
```

El nombre en una sentencia PROCEDURE tiene la misma apariencia que una definición de etiqueta, pero no se considera como tal, es decir, el nombre del procedimiento no es una etiqueta.

En cuanto a la sintáxis general tenemos:

1. El nombre del precedimiento es un identificador PL/M, el cual está asociado con el procedimiento.
2. El procedimiento puede ser llamado por el nombre usado en la declaración PROCEDURE, dentro del alcance de la misma, que viene gobernado por la situación de la declara-

ción. Esto será aclarado más adelante cuando hablemos de los atributos.

3. Una declaración de procedimiento es un bloque, al igual que lo es un DO simple.
4. Una declaración de procedimiento es un bloque y como tal puede contener, al igual que un DO simple, sentencias DECLARE en el cuerpo del procedimiento y éstas deben de preceder a la primera sentencia ejecutable dentro del cuerpo del procedimiento.
5. El cuerpo del procedimiento debe de contener al menos una sentencia ejecutable, a menos que el procedimiento tenga el atributo EXTERNAL.
6. Como en un bloque DO, el identificador en la sentencia END no tiene efecto sobre el programa, pero ayúda en la legibilidad y en el depurado del mismo. En el caso de que se use, éste debe de ser igual al nombre del procedimiento.

I-7-2. PARAMETROS.

Al igual que en otros lenguajes existen dos tipos de parámetros, en éste caso llamados parámetros formales y parámetros actuales.

Los parámetros formales son variables escala-

res sin base declaradas dentro de una declaración de procedimiento, cuyos identificadores aparecen en la lista de parámetros de la sentencia PROCEDURE. En la lista no se permiten variables subindicadas ó miembros identificadores de estructuras.

Cuando un procedimiento que tiene parámetros formales es llamado, la sentencia CALL ó referencia a función contiene una lista de parámetros actuales. Cada parámetro actual es una expresión cuyo valor es asignado al parámetro formal correspondiente en el procedimiento.

Analicemos seguidamente el siguiente ejemplo:

```
ACT: PROCEDURE (INDICE, L, MAS, MENOS);
     DECLARE INDICE ADDRESS;
     DECLARE (L, MAS, MENOS, ITEM) BYTE;
     IF ITEM = MAS THEN
         DO;
            INDICE = INDICE + 1;
            MENOS = Ø;
         END;
     ELSE
         DO;
            INDICE = INDICE + 9;
            L = MAS + 7;
         END;
END ACT;
```

El procedimiento ACT tiene cuatro parámetros INDICE, L, MAS y MENOS.

Supongamos que se realiza la llamada a éste procedimiento mediante la sentencia:

CALL ACT (DIRE, CONT, 1Ø, PUNT);

cuando se ejecuta dicha sentencia se produce lo siguiente:

1. Las cuatro expresiones de la sentencia CALL ACT (DIRE, CONT, 1Ø, PUNT) son evaluadas. Dichos parámetros son los parámetros actuales.
2. Los cuatro valores son asignados a los parámetros formales INDICE, L, MAS y MENOS, los cuales son declarados en el procedimiento ACT y que son nombrados en la lista de parámetros.
3. Seguidamente se ejecutan las sentencias que contienen el procedimiento ACT.
4. Finalmente, el control es devuelto a la sentencia siguiente al CALL de llamada al procedimiento.

I-7-3. PROCEDIMIENTOS "TIPADOS" Y "NO TIPADOS".

Existen dos clases de procedimientos, los "tipados" y los "no tipados". Se dice que un procedimiento es "no tipado" cuando en la sentencia PROCEDURE no se da el tipo, y además puede retornar algún valor ó ninguno, empleandose para la llamada de éstos procedimientos la sentencia CALL. Esto es lo que se conoce normalmente por subrutina.

Un procedimiento "tipado" es aquel que tiene un tipo, es decir BYTE ó ADDRESS en la sentencia PROCEDURE, y retorma siempre un sólo valor de éste tipo.

Para hacer referencia a ésta clase de procedimientos (tipados) se utiliza su nombre en una expresión. Esto es lo que normalmente se conoce como funciones. La forma de actuar de éste tipo de procedimientos es como sigue:

Cuando la expresión donde está incluida la referencia al procedimiento se evalúa en tiempo de compilación, se produce la ejecución del procedimiento. Una vez que el procedimiento se ha ejecutado, el valor que retorna de éste se reemplaza por la referencia de la función. La expresión es entonces terminada de evaluar y se continua la ejecución normal del programa.

Esta clase de procedimientos deben de contener

dentro del cuerpo del procedimiento una sentencia RETURN con una expresión o variable de las cuales se obtenga un valor.

I-7-4. SALIDA DE UN PROCEDIMIENTO.

La ejecución de un procedimiento se termina si ocurre alguno de los siguientes casos:

1. Debido a la ejecución de una sentencia RETURN dentro del cuerpo del procedimiento, que hace que el control sea transferido a la sentencia siguiente al punto de llamada ó bien a la expresión de donde fué llamado. Es regla obligada que los procedimientos "tipados" contengan éstas sentencias con una expresión ó variable que retorme el valor calculado.
2. Al llegar a la sentencia END que termina la declaración del procedimiento.
3. Por medio de un salto incondicional, mediante una sentencia GOTO a una sentencia exterior al cuerpo del procedimiento.

La sintáxis de la sentencia RETURN puede ser una de las dos formas siguientes:

```
RETURN;
```

ó bien

```
RETURN expresión;
```

La primera de las dos formas se utiliza para procedimientos "no tipados", es decir, para las subrutinas. Mientras que la segunda forma se utiliza para procedimientos "tipados".

I-7-5. EL CUERPO DEL PROCEDIMIENTO.

Cualquier sentencia PL/M puede estar dentro del cuerpo del procedimiento.

Analicemos a continuación algunos ejemplos de procedimientos.

El siguiente ejemplo es un procedimiento "tipado". Supongamos que en un programa PL/M tenemos el siguiente procedimiento:

```
INT: PROCEDURE (A, B) ADDRESS;
     DECLARE (A, B) ADDRESS;
     RETURN (A+B)/2;
END INT;
```

y también tenemos las siguientes sentencias en otra parte del programa:

```
ITEM$5 = 3;
DATO = 9;
FUN = INT (ITEM$5 , DATO);
```

Como se puede ver, en la última de éstas sentencias existe una referencia al procedimiento INT. Cuando el control llega a ésta sentencia se llama al procedimiento INT, entonces el valor de ITEM$5 es

transferido a A y el valor de DATO se transfiere a B. Se realiza a continuación la operación (A+B)/2 que dará como resultado 6. Este valor es devuelto y se le asigna a la variable FUN.

El ejemplo siguiente es un procedimiento "no tipado".

```
REP: PROCEDURE (ANALOG);
     DECLARE ANALOG ADDRESS;
     IF ANALOG = Ø THEN A= A + 5;
     RETURN;
END REP;
```

Vemos que dentro del cuerpo de éste procedimiento existen dos variables, ANALOG y A. La variable ANALOG es del tipo ADDRESS y ha sido declarada dentro del cuerpo del procedimiento, mientras que la variable A ha sido declarada fuera del mismo.

I-7-6. ATRIBUTOS DE LOS PROCEDIMIENTOS.

I-7-6-1. ATRIBUTOS PUBLIC Y EXTERNAL.

Los atributos PUBLIC y EXTERNAL pueden ser incluidos en las sentencias PROCEDURE, consiguiéndose así una gran potencia y disponibilidad para emplear una estructura modular.

Como reglas generales para la utilización de los atributos tenemos las siguientes:

1. Dentro de cualquier programa, cada pro-

cedimiento con alcance extendido debe de se declarado una vez con el atributo PUBLIC.

2. El atributo PUBLIC sólo puede usarse en el nivel superior de un módulo.
3. El atributo EXTERNAL sólo se utilizará cuando el procedimiento ha sido declarado PUBLIC en otro módulo del programa.
4. El atributo EXTERNAL no se puede utilizar en declaraciones de procedimientos donde esté el atributo PUBLIC ó los que veremos más adelante como INTERRUPT y REENTRANT.
5. La declaración de un procedimiento EXTERNAL debe de constar del mismo número de paréntesis y parámetros que en su declaración como PUBLIC. Así mismo, los tipos de las variables y sus demensiones estarán en la misma secuencia en ambas declaraciones y como norma general el nombre de los parámetros será el mismo.
6. La sentencia END que utiliza la declaración no podrá estar etiquetada.

Analicemos el siguiente ejemplo:

```
INT: PROCEDURE (A, B) ADDRESS PUBLIC;
     DECLARE (A, B) ADDRESS;
     RETURN (A+B)/2;
END INT;
```

y en otro módulo del programa tenemos:

```
INT: PROCEDURE (A, B) ADDRESS EXTERNAL;
     DECLARE (A, B) ADDRESS;
END INT;
```

tal que en éste módulo se puede incluir la siguiente expresión:

```
INDEX = SUBSIT + INT ( DIGI, DATO);
```

El efecto de ésto es activar el procedimiento INT tal y como está declarado en el primer módulo.

I-7-6-2. ATRIBUTO INTERRUPT.

El atributo INTERRUPT permite la creación de subprogramas asociados a las interrupciones. Para poder utilizar éste atributo el procedimiento debe de ser "no tipado", y además no tener parámetros, así mismo, debe de ser declarado en el nivel superior de un módulo de programa. El procedimiento con éste atributo recibirá entonces el nombre de procedimiento de interrupción. La sitaxis general será:

```
nombre: PROCEDURE INTERRUPT n;
```

donde n es cualquier constante numérica entre Ø y 255 (ambas inclusive). Al declarar a un procedimiento con el atributo INTERRUPT, nos da la posibilidad de que el procedimiento pueda ser activado sin ser llamado, cuando la correspondiente interrupción del microprocesador 8Ø8Ø ó 8Ø85 correspondiente a n se produzca durante la ejecución del programa.

Es importante conocer si el sistema que estamos empleando dispone de Controlador Programable de Interrupciones (8259), puesto que si no lo tuviéramos el número de interrupciones posibles se limitaría, de forma que serían sólo del Ø al 7.

El mecanismo de interrupciones tiene dos estados, ENEBLE (habilitado) y DISABLE (deshabilitado). Por razones de seguridad la CPU empieza siempre en el estado DISABLE (deshabilitado), por lo cual, cuando queramos ejecutar un programa y que éste pueda ser interrumpido, necesitamos alguna forma de poder habilitar las interrupciones. Esto en PL/M-8Ø se consigue mediante la sentencia ENABLE, la cual hace que la CPU entre en el estado "habilitado". La forma de incluir ésta sentencia en un programa es la siguiente:

```
ENABLE;
```

También se debe de dispones de la posibilidad de deshabilitar las interrupciones. Por necesi-

dades propias del programa en un momento determinado de la ejecución pueden estarse realizando determinadas instrucciones que no pueden ser interrumpidas, por lo cual, durante éste periodo de tiempo se debe deshabilitar el sistema de interrupciones. La forma de incluir ésta sentencia en un programa es:

DISABLE;

Las interrupciones pueden producirse por dos causas distintas:

1. La linea de interrupción del microprocesador ha pasado a estado alto accionada por algún periférico.
2. Debido a que un periférico ha enviado por el bus una interrupción RST. Este tipo de instrucción lleva asociado un número, es decir, pueden aparecer RSTØ, RST1, ... etc, hasta RST7 inclusive.

En el caso de que las interrupciones estén deshabilitadas, la CPU ignorará éstas acciones. Si las interrupciones están habilitadas, el RST será ejecutado.

Si un procedimiento ha sido declarado con un atributo INTERRUPT con el mismo número que la instrucción RST, entonces se produce una invalidación de las interrupciones y se activa dicho procedimiento. Una vez terminada la ejecución del proce-

dimiento se validan las interrupciones y el control retorna al punto donde el programa fué interrumpido. Otra opción es que podríamos terminar el procedimiento con un salto incondicional mediante un GOTO hacia un punto fuera del precedimiento. Cuando esto sucede así, el control de ejecución nunca retornará al punto del programa donde fué interrumpido, y como consecuencia las interrupciones no serán validadas automáticamente.

Al utilizar el atibuto INTERRUPT deben de tenerse en cuenta los siguientes puntos:

1. El atributo INTERRUPT no puede combinarse con el atributo EXTERNAL.
2. El atributo INTERRUPT sólo puede usarse en una sentencia PROCEDURE que se encuentre en el nivel más alto del módulo del programa.
3. La constante numérica puede ser cualquier número del Ø al 7 (inclusives). Cada número sólo podrá utilizarse una sola vez dentro del programa.
4. El procedimiento debe de ser "no tipado" y no puede tener parámetros.

Un procedimiento que tenga el atributo INTERRUPT puede se activado también mediante una sen-

tencia CALL, como otro procedimiento "no tipado". Pero cuando se hace esto hay que tener en cuenta que las interrupciones no son invalidadas automáticamente y que al terminar ésta son validadas.

I-7-6-3. ATRIBUTO REENTRANT.

Cuando un procedimiento no consta de éste atributo el almacenamiento para sus valores es localizado estáticamente en memoria. Esto origina una limitación importante que comprenderemos mejor con la hipótesis siguiente:

Supongamos que tenemos el procedimiento RESP$5 tal que puede ser ejecutado mediante una llamada desde el programa principal y también puede ser llamado mediante una interrupción. El programa se ejecuta y el procedimiento RESP$5 es activado. Mientras el procedimiento RESP$5 está siendo ejecutado se produce una interrupción y la ejecución de RESP$5 es suspendida.

Los datos utilizados por el procedimiento RESP$5 se almacenarán en memoria hasta que termine la interrupción y puedan ser nuevamente utilizados por RESP$5.

La interrupción que se ha producido llamará al procedimiento RESP$5. Al ejecutarse el mismo procedimiento, y la localización en memoria es estática, se estarán empleando las mismas variables que

las utilizadas en el procedimiento RESP$5 que ha quedado interrumpido. Luego, cuando la interrupción termine y se vuelva al punto donde se había detenido el procesamiento de RESP$5, ya las variables en su mayoría ó todas habrán sido modificadas y por lo tanto no se podrá seguir ejecutando el programa con garantias de obtener un resultado sin errores.

Para subsanar esto se emplea el atributo REENTRANT. Los procedimientos que llevan éste atributo se denominan "procedimientos reentrantes".

Cuando un procedimiento es declarado con el atributo REENTRANT, el problema de multiples llamadas está resuelto, ya que ahor las variables del procedimiento en lugar de estar localizadas estáticamente, son almacenadas en el STACK. De esta forma cada vez que se active utilizará una zona de almacenamiento separada.

También se puede conseguir, mediante la utilización del atributo REENTRANT, que un procedimiento se llame a sí mismo.

En la utilización del atributo REENTRANT deben de tenerse en cuenta los siguientes puntos:

1. Cualquier procedimiento que pueda ser interrumpido y activado desde el interior de un procedimiento de interrupción debe de llevar el atributo REENTRANT.

Si éste procedimiento llama a otros procedimientos, estos también deberán llevar el atributo REENTRANT.

2. Cualquier procedimiento que se llame a sí mismo deberá llevar éste atributo.
3. No puede usarse éste atributo en una declaración que contenga el atributo EXTERNAL.
4. El atributo REENTRANT sólo puede utilizarse en una sentencia PROCEDURE en el nivel más alto de un módulo.
5. Los procedimientos que tengan éste atributo no pueden tener otra declaración de procedimiento anidada dentro de él.

I-7-7. LLAMADAS A PROCEDIMIENTOS.

Existen dos formas de llamadas a procedimientos según estos sean "tipados" ó "no tipados". Los procedimientos "no tipados" son llamados mediante la sentencia CALL, la cual tiene la forma siguiente:

CALL nombre (lista de parámetros);

Los procedimientos "tipados" son llamados por medio de una referencia a la función, de forma que ésta referencia la podemos encontrar como un operando dentro de una expresión, de la forma siguiente:

nombre (lista de parámetros);

También nos es posible una tercera forma de llamada, pero sólo para los procedimientos "no tipados", empleando para la llamada su dirección, es decir, la posición de memoria donde se encuentra la primera sentencia ejecutable del mismo. Esto es posible utilizando la sentencia CALL de la siguiente forma:

CALL identificador .miembro identificador (lista de parámetros);

El identificador debe de ser una variable tipo ADDRESS, cuyo valor es el indicativo de la dirección donde está localizado el punto de entrada del procedimiento. Este identificador no puede ser subindicado.

Cuando se utiliza la sentencia CALL para llamar a un procedimiento, el compilador supervisa que el número de parámetros proporcionados en la llamada sea correcto, y ejecuta automáticamente el trasvase de estos a los parámetros actuales. Sin embargo, cuando la llamada se realiza por medio de la dirección el compilador no chequea el múmero de parámetros ni ejecuta el trasvase, de forma que se el número de parámetros es erroneo, ó si uno de los parámetros no es el mismo tipo que el correspondiente parámetro formal, el resultado es imprevisible.

I-8. ENTRADAS Y SALIDAS (I/O).

El PL/M no proporciona funciones de entrada/salida en el sentido usual del término, cuando hablamos de lenguajes de alto nivel. Sin embargo, el PL/M-8Ø es un lenguaje creado para desarrollar programas para microprocesadores, cuya forma de comunicación con el exterior es por medio de los interfaces que contienen unos latchs llamados comunmente ports, que se utilizan como lineas de entrada y salida.

I-8-1. FUNCION INPUT.

Mediante ésta función podemos leer una cantidad de ocho bits que están latcheados en uno de los 256 ports de entrada de los que consta un sistema que contenga el microprocesador 8Ø8Ø ó 8Ø85.

La sintaxis para la utilización de ésta función es como sigue:

variable = INPUT (dirección del port);

donde la variable es un identificador del tipo BYTE, que es donde se almacena la lectura obtenida en el port. La dirección del port debe de ser una constante numérica que indicará el port donde se encuentra el dato.

I-8-2. FUNCION OUTPUT.

Se utiliza para dar salida a ocho bits por uno

de los 256 ports del microprocesador 8Ø8Ø ó 8Ø85.
La forma de utilización es como sigue:

OUTPUT (dirección del port) = variable;

donde la variable es un identificador del tipo BYTE.
La dirección del port debe de ser una constante numérica que indicará el port por el cual se va a sacar el dato al exterior del sistema.

I-9. PROCEDIMIENTOS Y MACROINSTRUCCIONES DE LA LIBRERIA DEL PL/M-8Ø.

Una macroinstrucción es un conjunto de instrucciones, agrupadas bajo un nombre (nombre de la macroinstrucción), las cuales se escriben una sóla vez en el programa fuente (macro definición), pudiendo ser utilizadas en cualquier parte del programa que se desee, sin más que escribir una sola vel en el programa el nombre de la macroinstrucción y su lista de parámetros si es que los tiene. En nuestro caso, la macro definición se encuentra dentro del compilador, y al pones nosotros el nombre de la macro, ésta es incluida inmediatamente en la operación de enlazado, puesto que el compilador lo que hace es poner una referencia, después hay que enlazarla con la librería del PL/M-8Ø, para poder obtener el código de ésta macroinstrucción.

I-9-1. PROCEDIMIENTOS LENGTH, LAST Y SIZE.

I-9-1-1. LENGTH.

Es un procedimiento tipado cuyo tipo, BYTE ó ADDRESS, dependerá del valor calculado. Si éste valor es menor ó igual a 255, entonces retorna como un valor del tipo BYTE; en caso contrario, retorna con un valor del tipo ADDRESS. La forma de llamada es la siguiente:

LENGTH (identificador)

donde el identificador debe de ser una referencia no subindicada a un vector ó matriz. El array puede ser miembro de una estructura.

El valor del BYTE ó ADDRESS retornado es el número de elementos en el array, es decir, igual a la dimensión especificada en la declaración del array.

I-9-1-2. LAST.

Se trata de un procedimiento tipado cuyo tipo, BYTE ó ADDRESS, dependerá del valor calculado. Si este valor es menor ó igual a 255, entonces retorna como un valor del tipo BYTE, en el caso contrario retornará como un valor del tipo ADDRESS. La forma de llamada es la siguiente:

LAST (identificador)

donde el identificador es una referencia a un vector ó matriz. Este identificador no puede llevar subíndice.

El valor del BYTE ó ADDRESS retornado es el subíndice del último elemento de un array. LAST será siempre una unidad menor que el valor de LENGTH.

I-9-1-3. SIZE.

A diferencia de los anteriores, SIZE es un procedimiento del tipo ADDRESS. Este procedimiento

calcula el número de bytes requeridos por el objeto referenciado en la llamada, que tiene la forma:

SIZE (referencia ó variable)

Es de reseñar que el compilador no pone en el programa ninguno de estos procedimientos, sino que los calcula en tiempo de compilación y pone unicamente el valor obtenido, de ésta forma se ahorra tiempo de ejecución y además el espacio de memoria necesario es menor que si lo hubiera incluido.

I-9-2. PROCEDIMIENTOS LOW, HIGH Y DOUBLE.

Los procedimientos LOW y HIGH son análogos en cuanto al tipo de resultado, pero diferentes en la función que realizan. Ambos covierten un valor del tipo ADDRESS en un valor del tipo BYTE. La llamada a estos procedimientos se realiza de la siguiente forma:

LOW (expresión ó variable);
HIGH (expresión ó variable);

donde la expresión ó variable tiene un valor del tipo ADDRESS. El procedimiento LOW retorna el byte menos significativo del valor de la expresión ó variable; mientras que el HIGH retorna el byte más significativo.

Si la expresión ó variable es del tipo BYTE, al utilizar LOW retornará el mismo valor sin ningún

cambio. Si por el contrario se utiliza HIGH, retornará un valor cero.

El procedimiento DOUBLE es del tipo ADDRESS y se emplea para convertir un valor BYTE en un valor ADDRESS. La llamada a éste procedimiento es de la forma siguiente:

DOUBLE (expresión)

Si la expresión es del tipo BYTE, el procedimiento añade ocho bits cero al bit más significativo, retornando así un valor tipo ADDRESS.

Si la expresión es del tipo ADDRESS el procedimiento retornará sin haber efectuado ningún cambio.

I-9-3. PROCEDIMIENTOS SHIFT Y ROTATION.

Las operaciones de desplazamiento (SHIFT) y rotación (ROTATION) consisten en manipular un valor de ocho bits (BYTE) ó de dieciseis bits (ADDRESS), de forma que el valor es trasladado bit por bit a la izquierda ó a la derecha un número determinado de veces que se indicará en la llamada.

Los desplazamientos se diferencian de las rotaciones debido a que los bits que salgan fuera del formato de ocho ó dieciseis bits se perderán, ocupando ceros su lugar. Sin embargo, en las rotaciones, los bits que salen fuera del formato vuelven a entrar por el extremo contrario, estableciéndose de ésta

forma una rotación en todo el sentido de la palabra. Dentro de las rotaciones existen dos tipos, las llamadas rotaciones sin carry y las rotaciones con carry.

Las rotaciones sin carry son ROL y ROR, que son de un único tipo BYTE. Respectivamente indican rotaciones hacia la izquierda y hacia la derecha. La forma de llamar a estos procedimientos es como sigue:

```
ROL (constante ó variable, número de rotacio-
     nes);
ROR (constante ó variable, número de rotacio-
     nes);
```

donde el primer parámetro puede ser una expresión, además de los indicados de constante ó variable, teniendo en cuenta que el tamaño debe de ser de un byte. El segundo parámetro indica el número de rotaciones que deben de realizarse.

Las rotaciones con carry son SCL y SCR. Tienen como diferencias con las rotaciones sin carry las siguientes; en estos procedimientos el tipo depende del valor de una expresión dada como un parámetro actual, y además incluyen el bit de carry. Este es una celdilla de un solo bit de capacidad que se encuentra asociada al acumulador de la CPU.

Su utilidad radica en que se pone a 1 cuando ha existido un desbordamiento en alguna operación aritmética hecha con el acumulador. Cuando se reali-

zan éste tipo de rotaciones se incluye dicho bit, dé forma que el bit que sale fuera del formato debido a la rotación es colocado en ésta celdilla destinada al carry. El valor del bit que se encontraba en el carry será introducido por el otro extremo de la variable. La forma de llamar a estos procedimientos es como sigue:

SCL (expresión ó variable, número de rotaciones);
SCR (expresión ó variable, número de rotaciones);

Los procedimientos de desplazamiento, conocidos también como desplazamientos lógicos, son procedimientos cuyos tipos dependen del tipo de valor de la expresión dada como un parámetro actual. Son llamados de la forma siguiente:

SHL (expresión ó variable, número de veces);
SHR (expresión ó variable, número de veces);

siendo el valor de la expresión ó variable del tipo BYTE ó ADDRESS, retornando siempre un valor del mismo tipo que la muestra.

I-9-4. PROCEDIMIENTO MOVE.

Se trata de un procedimiento "no tipado" que se utiliza para transportar números de bytes contiguos de una zona de memoria a otra. La llamada se realiza mediante la sentencia CALL seguida de una lista de parámetros.

La forma de llamada a éste procedimiento es la siguiente:

CALL MOVE (contador, fuente, destino);

El parámetro llamado contador indica el número de bytes que deben de ser transferidos. El parámetro fuente sirve para significar la localización del primer byte a transferir. Por último, el parámetro actual, indica la posición de memoria a partir de la cual serán almacenados los bytes que serán transferidos.

I-9-5. PROCEDIMIENTO TIME.

Proporciona un retardo cuya duración es proporcional al parámetro actual enviado en la llamada.

La forma de llamar a éste procedimiento es la siguiente:

CALL TIME (expresión, variable ó constante);

donde el parámetro debe de ser del tipo BYTE. El procedimiento tiene un retardo de 1ØØ microsegundos. El parámetro máximo es 255, por lo tanto el tiempo máximo de retardo que se puede conseguir es de 1ØØ por 255, es decir, 2'5 milisegundos.

Hay que tener en cuenta que éste procedimiénto está basado en el reloj del microprocesador 8Ø8Ø, y por lo tanto, asume que el sistema está trabajando a 2 MHz. sin interrupciones.

I-9-6. PROCEDIMIENTOS DEC, CARRY, SIGN, ZERO Y PARYTY.

El procedimiento DEC realiza un ajuste decimal sobre el valor del parámetro de la llamada. Tanto el parámetro enviado como el devuelto son del tipo BYTE.

La forma de llamada es la siguiente:

DEC (expresión ó variable);

El resto de los procedimientos son todos del tipo BYTE y no necesitan parámetros, siendo la llamada a los mismos de la siguiente forma:

CARRY

ZERO

SIGN

PARITY

Al incluir alguna de éstas llamadas en la expresión, se genera un test para ver la condición del flag, de tal forma que si éste fuera 1 devolverá el valor ØFFH y si fuera Ø, entonces el valor de retorno será Ø.

I-9-7. SUBPROGRAMAS Y FUNCIONES DE APOYO AL 8Ø85.

El microprocesador 8Ø85 proporciona un tratamiento más completo de las interrupciones que el microprocesador 8Ø8Ø. Para apoyar al microprocesador 8Ø85, la librería PL/M8Ø.LIB (proporcionada por el compilador) contiene dos procedimientos que corres-

ponden a las instrucciones RIM (lectura de la máscara de interrupciones) y SIM (escritura de la máscara de interrupciones) del microprocesador 8Ø85.

Estos procedimientos, a diferencia de los vistos hasta ahora, son extensos, es decir, hay que declararlos en la zona de programa dedicada a éste menester.

El procedimiento equivalente a la instrucción RIM es R$MASK y que debe de ser declarado como EXTERNAL de la forma siguiente:

```
R$MASK: PROCEDURE BYTE EXTERNAL;
END R$MASK;
```

Una referencia a éste procedimiento en el cuerpo del programa se haría de la siguiente forma:

```
MASK$INT = R$MASK;
```

siendo el valor retornado del tipo BYTE, teniendo cada uno de los bits un significado determinado.

El procedimiento equivalente a la instrucción SIM es S$MASK, el cual acepta un parámetro BYTE y utiliza estos ocho bits para programar el mecanismo de interrupciones del microprocesador 8Ø85 y para enviar datos a través de la salida serie del 8Ø85.

I-9-8. VARIABLE PREDECLARADA, STACKPTR.

Su función consiste en posicionar el registro STACK POINTER en una dirección dada. Por lo tanto, el

valor que se le asigna debe de ser del tipo ADDRESS. Al utilizar ésta variable predeclarada hay que tener cuidado en saber la situación donde empieza el stack, y estar seguros que no existen datos almacenados en dicha posición

I-9-9. VARIABLE PREDECLARADA, MEMORY.

Es un vector del tipo BYTE de longitud no especificada y que representa el espacio de memoria libre y localizable cuando se realiza la localización del programa. Su utilidad consiste en guardar una serie de espacios de memoria libres, para después ser utilizados por el programa. La referencia a MEMORY puede ser subindicada. El máximo subíndice permitido depende del sistema con el que estemos trabajando. Hay que tener en cuenta que una referencia a MEMORY, tendrá como resultado la obtención del principio del espacio libre de memoria.

I-9-1Ø. HALT.

Es una sentencia que se puede poner así directamente en una linea de ejecución del programa. La función de ésta sentencia es hacer que el microprocesador entre en el llamado estado HALT (procesador parado) y que al mismo tiempo se invaliden las interrupciones, de forma que la única manera de salir de éste estado es por medio de la activación de alguna inte-

rrupción. La forma de incluirla dentro del programa es poniendo sencillamente:

HALT;

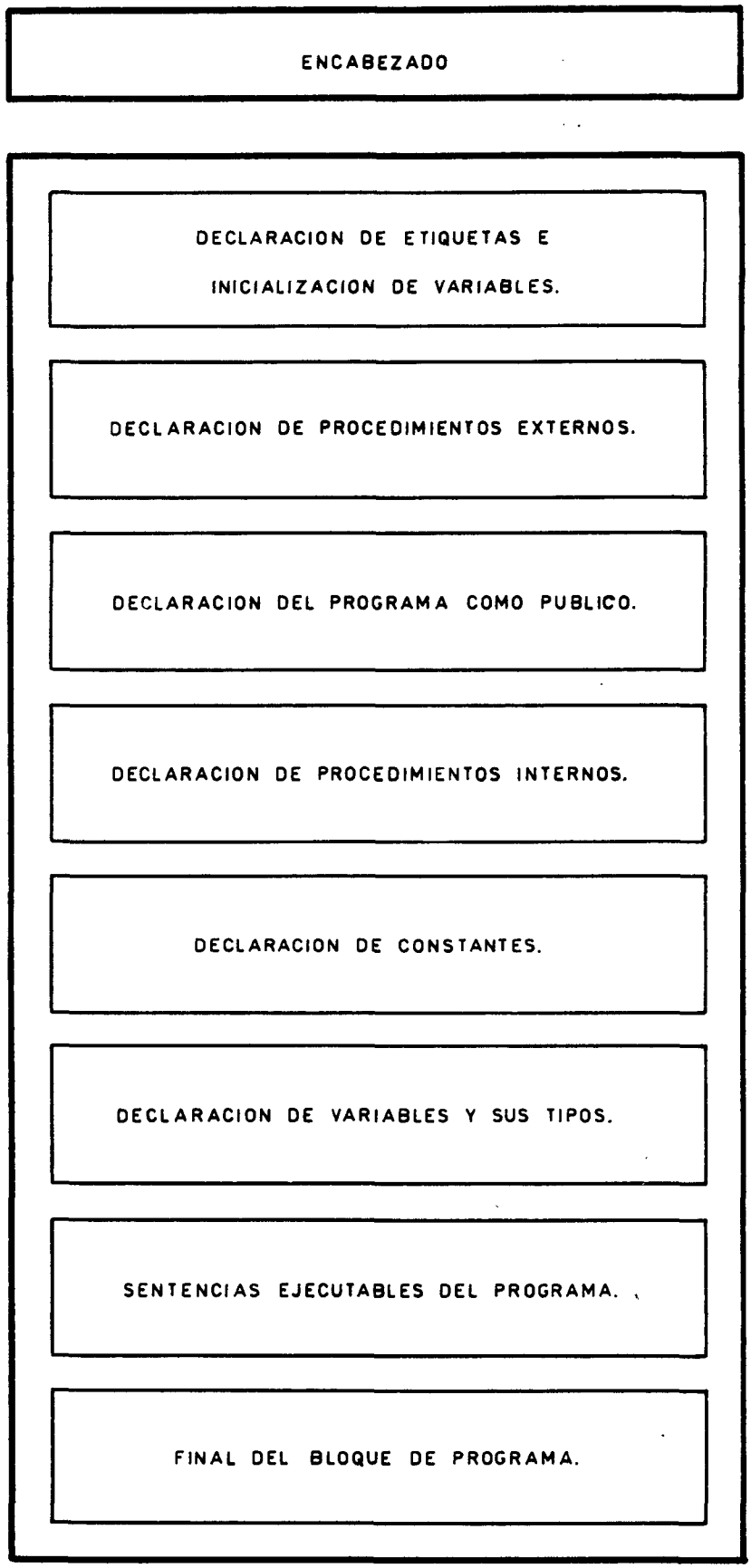
I-1Ø. PROGRAMACION MODULAR.

I-1Ø-1. ESTRUCTURA DEL PROGRAMA.

Los módulos ó bloques de programa en PL/M se dividen en dos partes, una llamada el encabezamiento y otra que es en sí el cuerpo del programa. En la primera parte se encluyen el nombre del programa y la inicialización del módulo mediante un DO simple. La segunda parte, llamada cuerpo del programa, se compone de siete apartados que en orden de preferencia son: declaración de etiquetas y inicialización de variables, definición de procedimientos externos, especificación de si el programa es PUBLIC (opcional), definición de procedimientos internos, declaración de constantes, declaración de variables y por último el apartado dedicado a las sentencias de ejecución.

En la figura se observa en primer lugar que todo módulo de programa empieza con la sentencia DO etiquetado con el nombre del programa. Esto indica el comienzo del bloque cuyo nombre le precede. Encontramos a continuación la declaración, mediante la sentencia DECLARE, de las etiquetas que se utilizan en el programa, seguidas por la parabra LABEL. En ésta misma sección se declaran también aquellas variables que vayan a tener un valor inicial con INITIAL. Seguidamente son declarados todos los procedimientos. En primer lugar se declaran los procedimien

Figura 1



tos externos. A continuación, y de forma opcional, se asignaría a nuestro módulo el atributo PUBLIC, de forma que éste módulo sería un procedimiento que se podría enlazar con otros módulos. Por último se declaran los procedimientos internos, que constituyen en sí pequeños programas que se van a emplear varias veces en la ejecución del programa.

Posteriormente a ésta zona se situarán aquellas variables que vallan a contener datos fijos, lo cual es especificado mediante DATA ó LITERALLY, dependiendo de la aplicación que se le vaya a dar a dicha variable. El siguiente apartado es dedicado a la declaración de variables y sus tipos. La sentencia DECLARE hace que a cada una de las variables se le asigne una posición de memoria.

A partir de ésta zona es donde irán las sentencias ejecutables del programa.

El programa concluirá con una sentencia END, y debido a las numerosas sentencias de éste tipo que pueden aparecer a lo largo del programa, es conveniente que ésta sentencia END de fin de programa vaya precedida con el nombre del mismo.

I-1Ø-2. VENTAJAS DE LA PROGRAMACION MODULAR.

En programación de alto nivel se tiende cada vez más a contar con varios módulos de programas, cada uno de los cuales, debidamente enlazados, ejecutarán una función determinada. Un lenguaje de programación puede tener una librería donde se guardan una serie de módulos, y a la hora de una aplicación determinada, se utilizan enlazados con el módulo principal, todos aquellos que sean necesarios y que existan dentro de la librería de módulos de programa.

En PL/M se entiende por módulo a un simple bloque DO etiquetado con el nombre del programa, el cual no está anidado a otro bloque.

En éste lenguaje existen dos tipos de bloques, los DO simples, las estructuras iterativas y de decisión y por otro lado las declaraciones de procedimientos. Llegamos de ésta manera al concepto de Estructura de Bloques Multinivel, es decir, cada parte de un programa en PL/M es parte integrante en algún bloque. Podemos decir por tanto que un módulo es un bloque que tiene anidadas en su interior otros bloques.

Tenemos que tener presente:

1. Los módulos deben de contener un módulo de programa principal, es decir, todos deben de ser enlazados dentro de algún módulo.

2. El módulo principal no necesita tener atributos que extiendan su alcance, pero todos los demás deberán tener el atributo PUBLIC, y en el módulo principal se hará referencia a ellos mediante el atributo EXTERNAL.

Hay que puntualizar que existen bloques que no se pueden considerar como módulos, sin ir más lejos, un bloque DO iterativo no puede ser un módulo de programa ya que entre otras cosas no pueden declararse variables en su interior, pudiendo ser sólo éste tipo de bloques partes constituyentes de un módulo de programa.

Analicemos seguidamente un concepto importante en PL/I como es el alcance de las declaraciones, es decir, hasta donde se pude hacer referencia a una variable, a donde puede llegar una etiqueta, etc.

- Alcance inclusivo. El alcance inclusivo de un bloque va desde la sentencia DO ó PROCEDURE que encabeza el bloque hasta la sentencia END que lo termina. De forma que tanto la sentencia DO ó PROCEDURE como la END forman parte del alcance inclusivo de un bloque. Sin embargo, la etiqueta que precede a la sentencia DO que empieza un módulo ó bloque no forma parte del alcance efecti

vo del bloque ó módulo, es decir, está fuera del bloque. Esto es válido también para la sentencia procedure, de tal forma que cuando se declare un módulo de programa como PUBLIC, la zona de declaración de procedimientos internos, etiquetas e inicializaciones está fuera del alcance inclusivo de la sentencia PROCEDURE, pero dentro del alcance inclusivo del bloque DO, que empieza el módulo del programa.

- Alcance exclusivo. Es el alcance inclusivo del bloque menos el alcance inclusivo de todos los bloques anidados en él.

Cada objeto que se declara en PL/M, ya sean variables, procedimientos, etiquetas, macros ,etc, tienen un alcance que se indica en los siguientes puntos:

1. El alcance de un objeto es la parte del programa en el cual el identificador del objeto es reconocido y manipulado de acuerdo a su declaración.
2. La declaración de un objeto debe hacerse en el alcance exclusivo de un bloque y su alcance está en el alcance inclusivo de cualquier bloque ó bloques anidados en los cuales se declara al mismo identificador.

3. El alcance de las variables, los procedimientos no reentrantes y macros está restringido y no se puede hacer referncia hasta después de su declaración. Esta restricción no se aplica a las etiquetas.

Las etiquetas están sujetas a todas las reglas vistas anteriormente exceptuando aquellas donde se diga lo contrario. Hay que añadir las siguientes:

1. No es posible declarar explicitamente la etiqueta de un módulo. Esto quiere decir que la etiqueta de un módulo no puede tener el PUELIC ó el EXTERNAL.
2. Cualquier etiqueta con el atributo PUBLIC debe de etiquetar a una sentencia ejecutable en el nivel más alto del módulo principal, es decir, debe de ser la primera sentencia ejecutable del módulo.

En cuanto a los saltos incondicionales, también tienen un alcance limitado, que viene regulado por las siguientes reglas:

1. No es posible para un GOTO transferir el control desde un bloque extremo a una sentencia etiquetada y situada en un bloque anidado.
2. No se permite transferir el control desde

un bloque a cualquier boque (en el mismo módulo) que no incluya el bloque contenedor del GOTO.

Las bifurcaciones siguientes son permitidas:

1. Desde un punto en el alcance exclusivo de un bloque a una sentencia en el alcance exclusivo del mismo bloque.
2. Desde cualquier módulo cuya etiqueta es declarada EXTERNAL, a una sentencia en el nivel más alto del módulo del programa principal, donde la etiqueta ha sido declarada con el atributo PUBLIC.
3. Desde un bloque superior a una sentencia de alcance exclusivo de un bloque anidado dentro del superior (no tiene porqué ser el más pequeño). Sin embargo, si el bloque incluido es un procedimiento, la transferencia puede hacerse solamente a una sentencia etiquetada en el nivel extremo del módulo de programa principal.

PARTE - II

ASPECTOS GENERALES DE UN SISTEMA DE SEGURIDAD

II-1. GENERALIDADES DE UN SISTEMA DE SEGURIDAD.

Los sistemas de seguridad tienen como misión fundamental la detección de cualquier tipo de anomalía (fuego, intrusos, etc) en aquellos lugares específicos donde se han colocado sensores adecuados, informando al usuario, mediante una señal de alarma acústica, luminosa ó de cualquier otro tipo.

II-1-1. FIABILIDAD.

La fiabilidad es uno de los factores más importantes. Si el circuito fallara, puede ser justamente cuando se le necesita. Un fallo en el momento inoportuno, anularía completamente el fin para el que fue montado.

La fiabilidad de una unidad de alarma no puede ser determinada facilmente de antemano. Solamente cuando esté instalada y funcionando, se pueden descubrir los fallos básicos de fiabilidad. Cualquier componente electrónico es suceptible de defectos. Estos pueden ser debidos a una debilidad inherente a un esfuerzo normal ó anormal durante la operación y al envejecimiento.

El que no se pueda garantizar el cien por cien de fiabilidad para cualquier sistema que emplee componente electrónicos, no es muy tranquilizador. Es obvio que cuanto menor sea el número de componentes, mayor será la fiabilidad total, debido a que

cada componente puede ser suceptible de fallo.

Otro factor de fiabilidad es la elección de los componentes: algunos tipos han resultado ser mucho más fiables que otros. Por ejemplo, algunos tipos de transistores tienen un excelente registro de confianza, mientras que otros tienen una tasa muy alta de fallos.

II-1-2. FALSAS ALARMAS.

Un sistema que es propenso a dar falsas alarmas no es seguro y tenderá a ser ignorado.

Las falsas alarmas se producen generalmente por defectos de instalación, descuidos en la seguridad de los protectores de puertas y ventanas, sensores supersensitivos, ó puesta en funcionamiento por una causa no humana (animales domesticos, corriente de aire, vibraciones por el tráfico, etc.). Las falsas alarmas no son producidas, casi nunca, por la unidad de control, aunque esto tampoco es imposible.

II-1-3. INMUNIDAD A LA DESACTIVACION.

Un sistema de seguridad no es útil si puede ser desactivado facilmente por un intruso. En cuanto sea posible, el sistema deberá, por lo tanto, estar contenido dentro del area protegida, de forma que no pueda ser alcanzado desde el exterior sin la activación de uno de los sensores de alarma. Debería, además, estar diseñado para que, incluso si la entrada

ya se ha producido, la alarma no pueda ser silenciada rápidamente por el intruso. Esto significa ocultar las partes vulnerables del sistema, como son, la unidad de control, la fuente de alimentación y la instalación de las alarmas sonoras.

II-2. COMPOSICION DE UN SISTEMA DE SEGURIDAD.

Basicamente los circuitos que componen un sistema de seguridad son:

- El sensor: es un elemento fundamental puesto que es el encargado de transmitir la información desde el lugar que se desea controlar hasta el circuito de control propiamente dicho.
- El circuito de control: es el encargado de recibir la información desde los diverso sensores, interpretarlos y enviar la señal de alarma correspondiente.
- El circuito de activación: se encarga de suministrar las señales necesarias para que se activen los elementos que avisarán al usuario de cuando se ha producido una anomalía.

II-2-1. SISTEMAS DE DETECCION.

II-2-1-1. SISTEMAS DE DETECCION PASIVOS.

Los dispositívos de detección pasivos funcionan como resultado de una señal generada por el intruso, ruido, vibración, presión, luz ó movimiento de una puerta ó de otro objeto.

II-2-1-1-1. MICRORRUPTORES.

La mayoría de los sensores constan de un interruptor; cada uno de los contactos está

normalmente abierto y se cierran cuando actuan, ó están normalmente cerrados y se abren cuando funcionan. Algunos sensores tienen dos pares de contactos, los cuales conmutan, cierran unos mientras abren los otros; algunos tienen tres contactos, uno que es común y que conmuta con los otros dos. Los contactos normalmente cerrados se utilizan en dispositivos de bucles contínuos, en donde una corriente circula cuando el sistema está alerta y el cese de la corriente dispara la alarma. Los contactos normalmente abiertos sueltan la alarma cuando se cierran, mientras que la conmutación del tipo de tres ó cuatro alambres ofrece mayor seguridad, capacitando a ambos tipos de circuito para ser utilizados simultáneamente.

El tipo más sencillo de sensor es el microrruptor. Como su nombre indica es un pequeño interruptor, que puede ser fijado fácilmente y escondido en los marcos de las puertas y ventanas.

A diferencia de los interruptores ordinarios que actúan con un movimiento de vaivén de la palanca, el microrruptor básico está impulsado por un contacto de presión que está distendido, de suerte que está oprimido cuando funciona y vuelve por sí mismo a la posición de reposo, cuando desaparece la presión.

Algunos fabricantes dan especificaciones completas desde el punto de vista mecánico, así como diversos parámetros del trabajo del sistema de contacto a presión lo cual es muy útil cuando se diseña un sistema o cuando se selecciona un interruptos para un propósito particular. A partir de la posición de partida, la cantidad de trabajo del contacto de presión, antes de que los contactos del interruptor actúen, se conoce como recorrido hasta la posición de trabajo. El movimiento desde éste punto avanzado es el denominado recorrido muerto y aunque se necesita un cierto incremento para asegurar que el contacto está de hecho actuando, no excederá el estipulado límite de trabajo.

El interruptor básico se complementa, a veces, con un accesorio externo para variar el modo de actuación. Existen tres tipos principales: accesorio de ballesta ó palanca, accesorio de varilla y accesorio de rodillo.

El accesorio de ballesta ó palanca está articulado en uno de sus extremos y pasa sobre el contacto de presión de forma que el movimiento del extremo libre actúa sobre él. El efecto es amplificar la magnitud del movimiento necesario para hacer funcionar el interruptor y la longitud de la palanca aumenta el movimiento reque

rido.

El segundo accesorio también es una palanca, sin embargo, en vez de una chapa plana, éste consiste en una varilla fuerte, con un doblez en ángulo recto y el extremo libre. La operación es la misma que para la palanca de chapa, sin embargo, la varilla sirve mejor como palpador. Se puede atar a la varilla una cuerda delgada y flexible y así poder ser utilizada en dondequiera que se necesite un contacto sensitivo.

Ninguno de estos dispositivos puede ser utilizado donde se requiera un contacto deslizante. Para esto es necesario el tipo de accesorio de rodillo. Los rodillos, están montados generalmente al final de las palancas, sin embargo cumplen la misma función si están fijados en el contacto de presión.

II-2-1-1-2. INTERRUPTORES MAGNETICOS.

Los microrruptores, no obstante ser absolutamente apropiados para ciertas aplicaciones, tienen sus limitaciones, especialmente como sensores en puertas y ventanas. Mientras que las unidades de buena calidad tienen una vida larga, si están sometidas a un uso normal son vulnerables de averiarse, tanto accidental como deliberadamente. La falta de cuidado en el dispositivo de arrastre, ó la entrada de objetos voluminosos a través

de la puerta, por ejemplo, pueden dar como resultado la rotura ó el descentrado del contactacto de presión ó de la palanca y la prsión excesiva, más allá del límite de trabajo, puede dañar el interruptor.

Un intruso que tenga acceso a un local durante el día, puede atascar un microrruptor con chicle. De esta forma el interruptor se quedaría detenido en la misma posición de cerrado. La puerta ó ventana no registrará una falta aún cuando el sistema del conectado fuera verificado previamente.

El interruptor está encapsulado en una caja de plástico y montado en el marco de la puerta ó ventana. Una caja semejante conteniendo una barra magnética se fija en la puerta ó en la ventana, de forma que cuando se cierren, el imán esté situado cerca del interruptor. De esta forma los contactos del interruptor están magnetizados y por tanto cerrados, pudiendo ser conectados, por esta razón, a un circuito de bucle cerrado. Cuando la puerta ó ventana se abren, el imán se desplaza y los contactos del interruptor se abren.

Este dispositivo ofrece numerosas ventajas sobre los microrruptores. Al no tener partes externas móviles, es mucho menos vulnerable a

ser deteriorado y tiene una mayor dificultad para su anulación. No cuenta con un contacto efectivo con la puerta y por esta razón no está afectado por vibraciones ó por irregularidades menores, como puedean ser el hinchamiento ó contracciones de la madera. No existen restricciones referentes al ángulo de actuación del imán, de debe de estar hacia adelante ó hacia atrás, respecto del interruptor. La única desventaja posible es que tengan que ser instaladas dos unidades en lugar de una. La posición y la distancia entre las unidades no es tan crítica como con el microrruptor y su superficie de actuación.

La distancia a la cual el imán influirá al interruptor, varía de acuerdo con el tipo de interruptor y la fuerza del imán. Los fabricantes han citado dos distancias, la distancia operante y la distancia de desexcetación. La primera es la distancia en la cual el acercamiento del imán cerrará los contactos y la segunda es la distancia a la que el alejamiento del imán permitirá abrir los contactos. La distancia de desexcitación es aproximadamente una vez y media la distancia operante.

Las distancias prescritas por los fabricantes están sujetas a una cierta tolerancia entre unidades individuales y las que están medi-

das con un imán nuevo. Como el magnetismo se pierde con el tiempo, las distancias se reducirán conforme pase éste. Para asegurar una actuación fiable en todo tiempo y con todas las unidades de un tipo particular, la distancia mínima deberá ser reducida en un veinticinco por ciento.

Los contactos de los interruptores son de metales preciosos y están siendo precintados para no verse afectados por la humedad ó la polución atmosférica. Por ésta razón tienen una vida extraordinariamente larga, por encima de los cien millones de operaciones, unas diez veces más que los microrruptores. Es probable que su vida no se acabe nunca cuando se utilizan como sensores de puertas, ello da una idea de la alta fiabilidad del interruptor magnético y es una razón más de por qué se utiliza ahora siempre para este fin, en sistemas de alarma.

Además de los interruptores sencillos de dos contactos, existen interruptores que conmutan tres contactos que facilitan el dispositivo para ser utilizados en el modo abierto normal en vez de hacerlo en un bucle cerrado, ó para mayor seguridad, en ambos al mismo tiempo.

Hay diferentes encapsulados para ajustarse a las diferente aplicaciones de montajes.

El tipo de empotramiento circular es-

tá inserto dentro de un agujero circular verificado en el marco, con un disco plano en la superficie. El imán está alojado en un agujero similar perforado en la puerta. Como las unidades son de alrededor de 35 milímetros de longitud, éstas deberán al menos estar introducidas en la madera y acomodadas allí. Por ello no es posible utilizar éste tipo en ventanas y puertas de cristales.

Otro factor es que la superficie de acoplamiento es pequeña, el interruptor y el imán deben de ser alineados muy cuidadosamente, el uno con el otro.

Otro tipo para la fijación por empotramiento es la variedad de forma rectangular estrecha. Como ésta tiene solo unos pocos milímetros de profundidad, puede instalarse en donde la madera no se muy gruesa, como en el marco de una ventana.

Hay además un tipo de fijación de superficie rectangular; éste está montado en la superficie del marco de la puerta, en el lado de la apertura y la unidad magnética en la superficie de la puerta. Siendo éste visible y accesible, es mucho menos seguro que los que están empotrados. Los empotrados son recomendables en todos los casos en los que sea posible; en donde haya

puertas o ventanas metálicas, puede ser fijada únicamente la superficie de montaje de las unidades.

Aunque difícil, no es imposible anular el interruptor utilizando un imán externo. Si el intruso conoce la posición del interruptor y la distancia de desexcitación, puede abrir la puerta levemente sin quitar la alarma, lo suficiente para introducir una pequeña barra imanada sobre el interruptor y de ésta forma retenerle en la posición "conectado". Entonces puede abrir la puerta del todo y retener el imán con cinta adhesiva.

Es evidente que un interruptor con fijación empotrada, si los contornos de madera trabajada y pintada han sido realizados convenientemente, puede ayudar a la seguridad, pues en este caso es muy difícil encontrar la posición exacta, especialmente con la puerta abierta solamente unos milímitros.

Para reducir aún más la posibilidad de anular un interruptor de lengüetas con imán exterior, existe un tipo de alta seguridad que contiene dos contactos de lengüeta, los cuales necesitan ser activados por los dos polos de un imán. Una barra imantada ó una pieza de acero imanado no activarán el interruptor y de ésta forma no puede ser neutralizado.

II-2-1-1-3. ESTERAS DE PRESION.

Las esteras de presión tienen un gran número de contactos, extendidos por toda la superficie de la misma, de forma que la prsión en cualquier parte de ella, cerrará los contactos: cuando la presión desaparece, los contactos se abren. Una envoltura de PVC (cloruro de polivinilo) cierra completamente la unidad, para que esté protegida de la suciedad y de la humedad.

En la mayoría de los casos, las esteras de presión se utilizan como una segunda línea de defensa. Si por alguna razón un intruso gana la entrada sin desmontar los sensores en las puertas y ventanas exteriores, es casi seguro que pisará una estera de presión colocada extrategicamente. La colocación correcta es muy importante, por lo que éstas deberán ser colocadas en donde un intruso tenga que pisar forzosamente.

Un inconveniente en las esteras de presión es que la mayoría tienen normalmente los dispositivos abiertos y el funcionamiento cierra los contactos. Esto significa que estos no pueden tomar parte en un bucle cerrado y por esto no pueden ser comprobados juntamente con los sensores de las puertas cada vez que se coloque la alarma.

II-2-1-1-4. CINTAS CONDUCTORAS.

Para proteger superficies de cristal, aparte de las ventanas corrientes en las que podrían fijarse otros sensores, pueden utilizarse cintas de hojas de metal. Generalmente son de aluminio, aunque se usa también el plomo en espacios en donde los productos de la polución atmosférica pueden corroer el aluminio. Las cintas se suministran en versiones tanto autoadhesvas como no adhesivas, aunque la versión autoadhesiva es la más conveniente para las aplicaciones normales.

Las cintas de hojas se ponen a lo largo de la parte interior del cristal y se conectan al circuito de bucle cerrado. Si se rompe el cristal, la hoja se rompe y el bucle queda en circuito abierto. Una simple cinta a través del cristal, es suficiente para pequeñas superficies, aunque serán necesas dos ó más para superficies más amplias.

Una característica importante de las hojas metálicas, es que son visibles desde el exterior. Por ésta razón, un futuro intruso puede ver que el cristal está protegido por una alarma y sabe que aún rompiéndolo, no conseguirá entrar.

Aún cuando en la mayoría de los casos las hojas metálicas ofrecen buena protección, no son recomendables. Un intruso determinado puede

utilizar un cortacristales y hacer un agujero en el cristal, eludiendo las cintas metálicas. Una alternativa puede ser la utilización de cristal alambrado. Este es un cristal especial que consiste en dos láminas selladas juntas, con una serie de alambres de plata entre ellos, completando casi la anchura del cristal. El alambre es muy fino y no molesta, estando espaciado aproximadamente unos 50 milímetros. Da una protección máxima. Lo mismo que la hoja metálica, el alambre está conectado en el bucle cerrado. Este sistema es, obviamente más caro que el anterior, pero ofrece una mayor seguridad.

II-2-1-1-5. CONTACTOS POR VIBRACION.

El contacto por vibración es un péndulo con contactos montados muy cercanos y cerrados dentro de una caja rectangular de plástico. Generalmente están fijados en grandes superficies de cristal.

Si el cristal se rompe, la vibración mueve la unidad y por consiguiente a los contactos. El péndulo permanece estacionario y de esta manera el movimiento relativo entre los contactos y el péndulo, activan el interruptor. Esta provisto de un medio para ajstar el espacio entre el péndulo y los contactos, para que el dispositivo pueda ser

colocado de forma que responda tanto a pequeñas como a grandes vibraciones. Los contactos por vibraciones pueden ser utilizados para otros propósitos que no sean la protección de cristales. Cualquier estructura, no protegida convenientemente por sensores convencionales, aunque ésta pudiera ser somentida a forzamientos o golpes para efectuar la entrada, podría estar equipada con contactores por vibraciones. En un ambiente normalmente tranquilo ó libre de vibraciones, la sensibilidad del dispositivo puede ponerse afinadamente para que dé la alarma a los primeros instantes de ingerencia. En cualquier otra aplicación, el éxito depende fundamentalmente del ajuste correcto.

II-2-1-1-6. DETECTORES ACUSTICOS.

El término abarca un margen de sensores que funcionan por el sonido generado por una entrada violenta. Comúnmente, éste sonido es el de un cristal roto y mientras que otros sensores pueden ser utilizados para detectar éste, los detectores acústicos tienen ciertas ventajas.

La principal ventaja es que la unidad está pensada para responder solamente a aquellas frecuencias de sonido que resultan de la rotura de un cristal; estas frecuencias van desde 6 a 8 KHz. La restricción de la respuesta de éste mar

gen limitado, significa que el dispositivo no actuará para otros sonidos y por esto no es susceptible de falsas alarmas, como el contactor de vibración.

El detector consiste en un micrófono, un amplificador montado con transistores y circuitos de filtrado, que dejan pasar solamente el margen deseado de frecuencias.

II-2-1-1-7. DETECTORES POR INERCIA.

Los detectores por inercia captan movimientos o vibraciones y por eso no son diferentes de los contactos de vibraciones, aunque es mucho mejor que un detector de movimientos de baja frecuencia. Estos pueden ser utilizados para proteger cristales y paredes y para detectar los movimientos de puertas, entradas, cercas, etc.

Los sensores se construyen utilizando una esfera de plata, asentada en un par de contactos, formando de éste modo, un interruptor normalmente cerrado; cualquier movimiento rompe el contacto. Las versiones de alta seguridad de este sensor, tienen además un aro alrededor de la esfera que sirve como un contacto normalmente abierto; una conmoción produce un contacto entre la esfera y el aro. De esta forma el sensor tiene ambos con

tactos, el normalmente cerrado y el normalmente abierto, y puede formar parte así de un sistema de tres ó cuantro conductores. Donde las vibraciones del ruido de fondo puedan ser altas, se utilizarán los sensores que están magnéticamente amortiguados, para evitar falsas alarmas.

II-2-1-1-8. DETECTORES FOTOELECTRICOS.

Son sensores muy especializados para utilizarlos en zonas normalmente oscuras, como pueden ser una bodeba o una cámara blindada. Además pueden utilizarse en cualquier lugar que no tenga ventanas y necesite protección. En tal situación, un intruso utilizará obviamente una linterna y la luz reflejada activará el sensor.

La sensibilidad puede ajustarse hasta menos de 1 lux. La luz procedente de una cerilla ó de un soplete es suficiente para activarlo. El sensor tiene un conmutador con un contacto de un solo polo, lo que significa que puede ser utilizado tanto como dispositivo normalmente abierto, como normalmente cerrado, o de ambas formas.

II-2-1-2. SISTEMAS DE DETECCION ACTIVOS.

Los dispositivos de detección activos generan su propia señal, la cual es radiada dentro del espacio protegido y recibida posteriormente, respondiendo la alarma a cualquier variación producida por la presencia de un intruso.

II-2-1-2-1. DETECTORES ULTRASONICOS.

El término "ultrasónico" quiere decir frecuencias de sonido que están por encima del margen del oido humano, cuyo límite superior es aproximadamente 16 kHz. Las frecuencias utilizadas en estos detectores varían en margenes comprendidos entre 23 y 40 kHz. El principio de funcionamiento es que un oscilador electrónico genera una frecuencia ultrasónica que alimenta a uno ó más traductores de ultrasonido. Los transductores, para producir frecuencias muy altas, necesitan ser pequeños, ya que es necesario que las partes móviles se muevan muy rápidamente y por eso sus masas tendrán que ser muy pequeñas. Esta es una ventaja para sistemas de alarma, porque así pueden tener una apariencia discreta.

Los sonidos de alta frecuencia se producen en el espacio protegido y se reciben después por un receptor de ultrasonidos que se encuentra alojado generalmente en la misma unidad.

Si hay cualquier movimiento dentro del campo de la unidad, el sonido reflejado procedente del objeto en movimiento experimentará un cambio de frecuencia debido al efecto Doppler. Por ello, el receptor ultrasónico captará las frecuencias diferentes, la directa procedente del emisor, la reflejad procedente de los objetos estacionarios y la frecuencia reflejada y cambiada, procedente del objeto en movimiento. Si se mezclan dos frecuencias cercanas, se producirá una tercera frecuencia que es la diferencia entre las anterriores esto se conoce con el nombre de nota de batido. En este caso, si la frecuencia directa es de 23 kHz y la reflejada Doppler de 22,7 kHz, la frecuencia de batido que aparecerá es de 0,3 kHz. También se genera una cuarta frecuencia, que es la suma de las dos frecuencias, pero ésta es demasiado alta para tener un valor práctico para ésta aplicación.

La frecuencia de la nota de batido depende de la velocidad del movimiento relativo de la superficie reflectora respecto de la unidad detectora, aunque en todos los casos ésta será mucho más baja que las frecuencias ultrasónicas que se producen.

El principio de funcionamiento es como sigue. La salida del receptor pasa primeramen-

te a través de un filtro paso alto, el cual elimina cualquier sonido ordinario procedente del exterior y que puede ser captado por el receptor de ultrasonidos, que solamente deja pasar los ultrasonidos. Otro circuito analiza ahora la señal de cualquier contenido de baja frecuencia; si está presente puede ser debido a una nota de batido, procedente de un filtrado Doppler. Esta es separada por un filtro paso bajo, la salida de éste es amplificada y finalmente hace funcionar un relé. Los contactos del relé pueden estar conectados a cualquier sistema de alarma. Generalmente el relé está activado continuamente y la señal de alarma desenergizada. De este modo, si la fuente de alimentación de la unidad falla, el relé se desenergiza y la alarma suena.

La principal diferencia entre las aplicaciones de un detector ultrasónico y los sensores descritos anteriormente, es que el tipo ultrasónico protege espacio, mientras que los sensores ordinarios protegen puntos de entrada. Con la detección ultrasónica cualquier cosa que se mueva en el espacio protegido genera una alarma. Por ésta razón puede ser utilizado también separadamente como un sistema de protección, conectado al sistema principal, ó bien para proteger un espacio que pudiera ser difícil ó impracticable el protegerlo por de-

tectores de entrada. Es virtualmente imposible destrozar el sistema si está bien situado enfrente de unos puntos posibles de entrada, ya que el movimiento es esencial para acercarse a la unidad. El inutilizar la unidad con anterioridad es casi el único medio para neutralizarla, aunque la mayoría de los modelos incluyen unos microrruptores antimanipulación para detectar enterferencias durante el día. Además, las fijaciones están diseñadas para observar discretamente, estando disfrazadas para parecer que son otra cosa.

El margen de detección está nominalmente entre los 4,5 y los 6 metros aunque está afectado por la naturaleza de las superficies en el entorno. Si ésta es dura, como por ejemplo madera, ladrillo, yeso ó piedra, el margen podrá extenderse, pero si es blanda y absorbente, como unos tapices gruesos, cortinas y acolchados, el margen es reducido.

La desventaja de los detectores de ultrasonidos es que tienen problemas de falsas alarmas. Una frecuencia Doppler filtrad puede ser producida por el aire, a través del cual viaja el sonido, riviéndose de la misma forma que lo hace la superficie reflectora. Debido a esto, los detectores de ultrasonido no pueden utilizarse en exteriores, porque la más ligera brisa podría dar una

alarma. Incluso en los interiores, existen muchas fuentes de aire en movimiento, que podrían dar la alarma. De entre éstas, están las corrientes que vienen de las puertas y ventanas, el movimiento descendente del aire frio debido a las ventanas, las corrientes de convección producidas por los radiadores y las turbulencias causadas por los condicionadores de aire y ventiladores. El movimiento de cortinas debe ser evitado también y los animales excluidos del recinto.

Otra causa posible de falsas alarmas, es que los sonidos de tono muy alto originan otros cercanos. Estos pueden ser armónicos que se extienden dentro del margen de los ultrasonidos y debido a esto podrían producir notas de batido en el detector. Otras posibles fuentes son los frenazos de los vehículos en calles adyacentes, el silbido de la base de tiempo de líneas procedentes de un receptor de televisión cercano, fugas en las canalizaciones de aire comprimido, de agua y de gas. Cuando se imponen condiciones y el lugar propuesto, se deberá tener muchísimo cuidado en verificar si la detección ultrasónica es adecuada y no es probable que esté plagada de falsas alarmas.

II-2-1-2-2. DETECTORES DE MICROONDAS.

La mayor parte de los detectores de microondas trabajan de una forma similar a los sistemas de ultrasonidos, excepto que se utilizan ondas de radio en lugar de ondas sonoras. El cerebro del generador es un cristal de arseniuro de galio, conocido como un diodo Gunn; se llama efecto Gunn a las oscilaciones de corriente a ritmo de RF que tienen lugar cuando se aplica un campo eléctrico del orden de 3.000 a 13.000 V/cm a un trozo corto (alrededor de 0,1 mm) de un semiconductor, por ejemplo arseniuro de galio, del tipo N. El mencionado diodo Gunn oscila en un margen elevadísimo, siendo en la mayoría de los modelos de aproximadamente 10,7 GHz.

Estas oscilaciones están proyectadas en forma de ondas de radio desde un radiador, como un haz. La potencia radiada varía, aunque generalmente es de unos 10 mW. El haz se dirige dentro del area protegida y las señales reflejadas son detectadas por un receptor y mezcladas con una muestra de la señal del transmisor. Cualquier diferencia enla freduencia es debida al efecto Doppler desde un objeto móvil y el batido resultante es amplificado y utilizado para disparar la alarma.

Lo más notable de las microondas es que pueden atravesar la madera, el cristal, el yeso e incluso, con una extensión limitada, los ladrillos. Esto puede tener tantas ventajas como desventajas. En un espacio en donde hay grandes objetos, tales como cajas para embalaje en un almacén, en donde podría cobijarse un intruso, las microondas pasarían a través de estos y de éste modo revelarían cualquier objeto ó persona en movimiento que pueda estar oculto mediante un sistema de ultrasonido ó de vigilancia visual. Esto es útil en una situación en la que los objetos sean trasladados de lugar continuamente y en donde no sea posible disponer siempre de una visión clara desde un punto de verificación, para la seguridad del personal. Hay que hacer notar de que a pesar de que los objetos de metal reflejarán las microondas, la maquinaria se comportará como un escudo.

La desventaja consiste en que el rayo atravesará los suelos, las paredes y particularmente el techo y por esta razón podría disparar una falsa alarma, debido a los objetos que se mueven fuera del espacio protegido. Para superarlo, los transmisores pueden ser fijados en placas deflectoras que dan una conformación al rayo, tanto en el eje horizontal, como en el vertical. El eje

vertical es generalmete plano, parecido al rayo del faro de un coche y no penetrará enel techo. La distribución horizontal puede ser ancha, estrecha e incluso con una cobertura devidida si hiciera falta.

El margen de un sistema de microondas es mucho más grande que el de un detector de ultrasonidos, yendo normalmente desde 4,5 a 45,5 metros. Para evitar falsas alarmas, el rayo no deberá ser mayor al recinto que tiene que ser protegido. La mayoría de los sistemas de detección por microondas ofrecen la posibilidad de variar la potencia y,por consiguiente, el margen de radiación. Cuando se pone en las longitudes más cortas, la anchura y la altura se reducen también en proporción.

Las microondas pueden pasar a través de materiales plásticos; debido a esto, las unidades transmisoras y receptoras pueden estar contenidas en cajas de plástico ó en una metálica con el frente de plástico. Un modelo para el techo está pensado para montarlo a haces, de modo que todo él aparece desde abajo como un panel rectangular. Algunas unidades están fijadas a un soporte por medio de una articulación de rótula esférica y de ésta forma puede situarse formando un ángulo en cualquier dirección.

Aparte de la superposición de gamas, más alla de la extensión requerida, hay pocas posibilidades de falsas alarmas. Las turbulencias del aire no tienen efecto, como ocurre con los dispositivos de ultrasonidos. Algunas lámparas de atmósfera gaseosa, como las lámparas fluorescentes y de neón, pueden generar radiofrecuencias que pueden ser detectadas por el receptor, aunque éstas pueden ser filtradas fuera electrónicamente. La mayoría de los modelos tienen estos filtros incorporados.

Otros tipos de sistemas de microondas no utilizan el efecto Doppler, sino que transmiten un rayo estrecho al receptor situado en un punto remoto, que puede ser mayor a 150 metros. La alarma se produce si cualquier objeto interrumpe el rayo. La dispersión del rayo viajando desde el transmisor, tiene una anchura comprendida entre 1 y 6 metros en su margen mayor. Tiene un factor incorporado de "auto-protección"; si el transmisor dejara de funcionar por cualquier razón, el receptor disparará la alarma.

Este sistema está proyectado para su uso en el exterior, en donde las unidades con efecto Doppler son inadecuadas porque la parte delantera, que consiste en una protección de madera ó rejilla de alambre, es transparente a las microóndas.

También cuando el lindero es una valla de ladrillos, podría producirse un escape considerable de radiaciones de microondas en el remate. Puede haber fuertes posibilidades de falsas alarmas debido a los objetos que se muevan fuera de ella, a menos que el margen sea restringido para caer dentro del perímetro, en cuyo caso la protección se quita justamente cuando se necesita en las cercanías de los puntos de entrada.

El sistema de rayo interrumpido puede ser instalado de forma que transmita justamente dentro y en en paralelo a la protección de la parte delantera. Para un recinto rectangular típico, se necesitarían cuatro unidades transmisoras y receptoras, una para cada lado. Estas podrían guardar más de 61 Km de linderos vallados, para los cuales los detectores de inercia necesitarian unos doscientos sensores. La elección entre estos dos sistemas se rige, obviamente, en una gran parte por el tamaño de la extensión que se vaya a proteger y por su forma. En un perímetro con una configuración irregular, no se puede realizar fácilmente una protección con microondas de rayo interceptado.

Las microondas están atenuadas por la lluvia y la niebla por lo que las unidades que se

instalan en exteriores van equipadas con controles automáticos de ganancia para compensarlo. Otro riesgo en las instalaciones exteriores son pequeños objetos, pájaros y papeles arrastrados por el viento que pueden interceptar el rayo y originar una falsa alarma. La solución adoptada por algunos fabricantes es establecer un tamaño mínimo de detección. De este modo, cualquier objeto con una superficie menor que la mínima considerada, no dispararía la alarma.

Lo mismo que con los detectores ultrasónicos es virtualmente imposible anular un sistema de microondas que se encuentre en funcionamiento, ya que será detectada cualquier aproximación a las unidades. En todo caso, un sistema Doppler podría ser anulado durante el periodo en que éste se encuentre desconectado. Si la unidad se encuentra montada en un soporte ajustable, podría ser girada en redondo, para poner su cara en otra dirección, ó quizás se podría fijar una placa metálica sobre la cara transmisora. Cuando se conecta el receptor interno recibirá, sin embargo, las reflexiones desde la placa ó pared, pero nunca desde la extensión protegida; de ésta forma no se detectará ningún movimiento.

La posición cambiada ó la placa deflectora pueden pasar inadvertidas en un establecimien

to ocupado, pero donde su manipulación sea posible, se puede instalar un monitor de microondas en un punto remoto, desde la unidad. Siendo un receptor sensible, éste detectaría microondas y generaría señal de alarma, si no se recibe señal.

II-2-1-2-3. SENSORES DE INFRARROJOS.

La radiación infrarroja es una emisión electromagnética que se extiende justamente por debajo de la parte de luz visible en el espectro, pero mucho más alta que las radiofrecuencias normales. El margen actual de la frecuencia es mucho más ancho que el de la luz visible, siendo aproximadamente de 1.000 a 100.000 GHz. Afortunadamente, la radiación en ésta región en generada facilmente, bien por una lámpara de infrarrojos que es una lámpara con un filamento ennegrecido trabajando a baja temperatura ó, de forma más eficiente, por un cristal de arseniuro de galio. En el segundo caso, la radiación se denomina a veces como "rayos de arseniuro de galio".

El sistema utilizado es del tipo de rayo interceptado. Un proyector transmite un rayo estrecho a través de la entrada que se ha de proteger, con un receptor en el otro lado. Si el rayo es interceptado, el receptor dispara la alarma, de forma similar al sistema de microondas para ex

teriores.

La radiación infrarroja está generada por la mayoría de las fuentes que producen calor y las dos están estrechamente relacionadas en el espectro. Así, las bombillas y otros calentadores eléctricos e incluso una linterna de bolsillo, generan rayos infrarrojos de mayor ó menor extensión. Cualquiera de estos podría interferirse con el sistema y un intruso podría anularlo encendiendo una fuente de infrarrojos portátil, en la dirección del receptor, mientras que él se mueve atravesando y rompiendo el rayo original. Este tipo de interferencia está previsto por la modulación de la radiación procedente del proyector. Este necesita que el rayo no sea uniforme, sino que esté generado por una serie de impulsos rápidos con un régimen predeterminado. Este régimen varía de acuerdo con el tipo de equipo, pero generalmente es de aproximadamente 200 veces por segundo. El receptor dispone de unos filtros electrónicos que aceptan solamente la frecuencia modulada y el relé que dispara está alejado de la presencia de las señales entrantes de ésta frecuencia. Si las señales cesan, se desengancha el relé y se da la alarma, incluso si en éste lugar se recibe una radiación infrarroja constante. Las unidades del transmisor y el receptor están alojadas en cajas ideñ-

cas, por lo tanto, como el rayo es invisible, no es posible identificar las unidades. Esto ayuda a complicar cualquier intento de inutilizar el sistema.

La única trayectoria del rayo puede eludirse si puede ser evitada físicamente, por ejemplo, arrastrándose por debajo de él. Para hacer esto, el intruso debe de conocer la presencia del sensor de infrarrojos y de su trayectoria. La evasión se hace imposible si el rayo está "enlazado" a través de puntos de la entrada, haciendo zigzag. Esto puede realizarse por medio de espejos en cada lado; siendo diferente del rayo de microondas, que se difunde demasiado, el rayo de infrarrojos puede ser ser reflejado en varios puntos por pequeños reflectores, justamente como un rayo de luz; de ésta forma, un solo proyector consu receptor puede proteger una trayectoria de contorno irregular, o ser enlazados a través de puertas exteriores ó espacios entre edificaciones.

El margen de los sistemas de infrarrojos varía desde un máximo de 12 metros para unidades pequeñas, hasta los 300 metros para las más grandes. Cuando se refleja en un espejo, el rayo es atenuado y por eso cada reflexión acorta el margen.

Los rayos también podrán atravesar cristales transparentes y por ello la protección puede ser extendida a través de tabiques de cristal y ventanas. Hay de nuevo ahora una reducción en el magen por cada superficie de cristal, aunque la atenuación es menor que en un espejo reflector.

Este método de protección es, por esta razón, versátil. Puede ser utilizado alrededor de perímetros, enlaces a través de entradas, enlazando escaleras en el piso inferior y a través de habitaciones y oficinas. Pueden utilizarse también para transmitir señales de alarma desde un espacio auxiliar protegido hasta el espacio principal.

Cuando se utilizan en exteriores, existe la posibilidad de que el rayo sea interrumpido por pájaros u objetos que lleve el viento. Para solucionar éste problema, se utilizan dos proyectores y dos receptores para producir rayos paralelos espaciados unos de otros de 450 a 600 milímetros. Los contactos de la alarma del receptor están conectados en paralelo y de ésta forma, si cualquiera de los rayos fuera interceptado, el bucle se mantendría a través del otro contacto. Así, los objetos pequeños no producirían una falsa alarma. Solamente cuando ambos rayos están interceptados,

es cuando los contactos se abren y la alarma se dispara.

II-2-1-3. SISTEMAS DE DETECCION DE INCENDIOS.

Es de vital importancia la detección de un incendio desde los primeros momentos en que éste se produce. Esta rápida detección redundará no solamente en una más rápida y eficaz extinción del incendio, sino que también permitirá una más rápida y segura evacuación de las personas que en esos momentos se encuentran en el lugar.

Con ésta finalidad se utilizan sofisticados sistemas de detección de incendios, como los que seguidamente se mencionan.

II-2-1-3-1. DETECTOR DE HUMOS POR IONIZACION.

Este detector está basado en la fuente radiactiva del Americio 241, con doble cámara de ionización. La fuente radiactiva está compuesta por dos cámaras perfectamente aisladas entre sí estando una de ellas cerrada al paso del aire y la otra abierta en contacto directo y continuo con el mismo.

Cuando la cámara en contacto con el ambiente detecta concentraciones de partículas iónicas (a veces invisibles, como son las producidas por un cigarro encendido caído sobre una moqueta o el tostado del aislante de los conductores eléctricos cuando sufren un cortocircuito), experimen

ta una variación de tensión rompiendo el equilibrio eléctrico que mantenía con la cerrada, dando origen al proceso de alarma.

Esta variación es enviada a un circuito, el cual después de analizarla, la compara con la sensibilidad que tiene ajustada y determina su entrada en alarma.

El ajuste de sensibilidad del detector se realiza por medio de un potenciómetro, con lo cual la regulación de la misma queda simplificada.

II-2-1-3-2. DETECTOR TERMOVELOCIMETRICO.

Es un detector electrónico con sensor de temperatura NTC. Ofrece la doble protección termovelocimétrica y térmica.

Reacciona ante un incremento brusco de la temperatura, su instalación es aconsejable en lugares donde el detector iónico no puede aplicarse, por existir influencias perturbadoras, como puedan ser humos o gases que no procedan de incendios.

Está diseñado para aquellos riesgos donde se prevee un incremento rápido de la temperatura en caso de incendio.

II-2-1-3-3. DETECTORES TERMICOS.

Su activación se consigue cuando la temperatura del lugar donde se encuentra situado

el sensor alcanza un tope fijo, sin tener en cuenta el que su evolución se haya producido lenta o bruscamente. Por lo general, estos sensores disponen de un piloto indicador de alarma y se recupera automáticamente cuando la temperatura desciende por debajo de los 48 °C.

Este tipo de sensores son adecuados para sustituir a los iónicos y termovelocimétricos en aquellos recintos donde habitualmente existen humos o fluctuaciones térmicas tales como secadores, cocinas, etc.

II-2-2. CIRCUITOS DE ACTIVACION.

II-2-2-1. INDICADORES.

Son señales luminosas que se utilizan para informar al usuario de la localización exacta del sensor que ha sido activado. Una vez que la señal luminosa ha sido activada debe de permanecer en dicha posición hasta que sea "reseteada" por el usuario.

Los indicadores se emplean también para informar que el sistema se encuentra en funcionamiento ó que no hay ningún fallo de alimentación en el mismo.

Generalmente, se utilizan diodos emisores de luz (LEDs) como indicadores. Estos están fabricados de material semiconductor que eminten luz cuando son atravesados por una corriente eléctrica. Los LEDs tienen una mayor vida de funcionamiento y un consumo mucho menor que una lámpara de filamento.

II-2-2-2. ALARMA SONORA.

II-2-2-2-1. NIVELES DE SONIDO.

El nivel de sonido en decibelios (dB) generado por un dispositivo de alarma, necesita ser de un orden elevado.

El oido humano no responde de una forma lineal al incremento de la presión sonora, si-

no que sigue una ley logarítmica. Esta característica permite al oido humano distinguir entre un amplio margen de niveles sonoros, desde un estampido cercano hasta el murmullo de las hojas. Por ésta razón, la relación de dB es también logarítmica. Algunos valores comunes son:

6 dB = x 2

10 dB = x 3

12 dB = x 4

18 dB = x 8

20 dB = x 10

Cuando se aplica un nivel de presion acústica, se hace referencia al umbral del oido humano, el cual es considerado como 0 dB.

Para dar algún significado práctico a las cifras citadas, nos servirán de ayuda algunos ejemplos. En el interior de una habitación, en una vecindad tranquila, en donde parece haber una aparente silencio, habrá un nivel de presión acústica de alrededor de 20 dB, el tictac de un reloj es aproximadamente de 30 dB. Un cuchicheo a una distancia de 1 metro, está cercano a los 45 dB, mientras que una voz hablando da un promedio de 70 dB en la misma distancia. Cualquier ruido por encima de los 80 dB se considera, generalmente alto, los 100 dB son de igual modo desagradables. Una tala-

dradora neumática a 1 metro, genera de 105 a 110 dB. Este nivel de ruido, si se mantiene durante un largo período, puede producir daños en el oido. El umbral del dolor se alcanza en los 130 dB.

Una cuestión a tener en cuenta, es que la presión sonora decrece con la distancia a la fuente que lo produce. Esto se produce porque es logarítmico, pero si se utiliza la notación en decibelios, podemos calcularlo de acuerdo con los valores mencionados previamente. De este modo, al doble de la distancia el nivel es 6 dB menor, a tres veces la distancia 10 dB menor y así sucesivamente. La mayoría de los dispositivos de alarma se evalúan a una distancia de 3 metros.

II-2-2-2-2. TIMBRES.

El dispositivo más común en alarmas es el timbre. El principio de funcionamiento es sencillo; la corriente es aplicada a una bobina con un núcleo y una varilla de inmersión que es atraida magnéticamente en el centro, golpea el borde de una campanilla. A causa de esto, se separan un par de contactos y se interrumpe la corriente a través de la bobina, la varilla de la bobina es llevada a su posición de reposo por medio de un muelle, los contactos se cierran y la corriente

vuelve a aplicarse. La corriente, de este modo, fluye en impulsos que suponen que el valor medio es menor que el valor instantáneo del impulso.

La distancia del recorrido de la varilla de inmersión regula la calidad del sonido. Si está demasiado cerca de la campanilla, una llamada es amortiguada por el golpe que retorna desde el inmediato y el resultado es un castañeteo más que un sonido resonante. Si la varilla de inmersión está demasiado separada, no golpeará a la campanilla con toda la fuerza y los golpes, aislados, están demasiado alejados esta vez; por consiguiente el sonido es débil. Hay entonces una distancia óptima, la cual produce el sonido adecuado. Esta distancia puede establecerse con un tornillo de ajuste de los contactos, ó girando ligeramente la campanilla excéntrica.

El nivel de presión acústica óptimo para un timbre de alarma está dado entre 81 y 84 dB.

Es conveniente la práctica de encerrar el timbre en una caja metálica ó de fibra de vidrio. Esto tiene dos objetivos, proteger el timbre de la interperie y de las manipulaciones. Para aumentar la seguridad, en muchos casos están colocados con microrruptores anti-manipulación,

dichos circuitos están conectados las 24 horas en las unidades de control y, por esta razón, siempre están activados. Cuando el timbre está montado en una posición que permite un fácil acceso, puede ser útil encerrarlo. Aún así, puede ser silenciado utilizando espuma plástica. Un timbre descubierto puede experimentar más dificultades. Inevitablemente, encerrado el timbre se reduce el sonido radiado. La mejor protección es instalar el timbre en una posición en la que la interferencia tuviera mucha dificultad.

Un argumento en contra de los timbres descubiertos, es que pueden ser silenciados con un sencillo movimiento de la campanilla, la cual está fijada por un solo tornillo, en su centro. No obstante algunos timbres utilizan un tipo especial de tornillos, los cuales necesitan una herramienta especial para desmontanrlos.

El tamaño normal de la campanilla de un timbre de alarma es de 150 milímetros, pero en trabajos interiores se utilizan los más pequeños de 100 milímetros de diámetro. Como regla general, las campanillas grandes generan más volumen de sonido que las más pequeñas, con el mismo mecanismo.

La parte más vulnerable de un sistema de alarma, es el alambrado del timbre. En la

mayoría de los casos, si el alambrado del sensor es manipulado con éste, resultará que la alarma sonará, o inmediatamente (si el sistema está conectado ó el alambrado está 24 horas en circuito de alerta), o la interferencia está puesta de manifiesto cuando la alarma se conecte por la noche.

Al contrario de la instalación del sen sor, que debe de hacerse en un lugar que sea fácil su mantenimiento, la instalación del timbre tiene la ventaja de poderse realizar en posiciones inaccesibles, por regla general, añadiendo que el cableado puede ser protegido por conductores metálicos, otros enterrados en el yeso ó debajo del piso. Así, aunque vulnerable, éste puede ser protegido realmente y se le da una adecuada protección, de forma que el riesgo en la seguridad es pequeño.

Aún cuando el cableado puede representar el eslabón débil de la cadena, aunque solamente un centímetro ó dos de cable estén descubiertos, sería suficiente para que alguien que haya hecho de esto su oficio y esté familiarizado con el sistema, ponerlo fuera de actuación. Para superficies en donde es esencial un alto grado de seguridad, incluso éste riesgo es inaceptable, por lo que se emplea el timbre activado. Este está contenido en una caja de acero, juntamente con

una batería de alimentación y con su propio circuito de enganche. De éste modo es independiente de cualquier fuente de alimentación externa y tampoco está afectado por un cambio de corriente procedente de la unidad de control.

II-2-2-2-3. SIRENAS.

Cuando se requieran alarmas extraordinariamente fuertes o que deban ser oídas a mayor distancia de lo normal, se utilizará la sirena. Esta consiste en un motor electrico que dirigiendo e impulsando aire con fuerza a través de una abertura en la cubierta, produce un tono fuerte. La unidad completa está cerrada totalmente, en la mayor parte está diseñada para aplicaciones en el exterior, aunque las unidades más pequeñas que no son a prueba de interperie, pueden utilizarse en interiores.

El margen de trabajo en decibelios va desde 90 a 110, a 3 metros, siendo el valor más alto extremadamente ruidoso. Los fabricantes, generalmente dan el margen sobre el cual la sirena es audible. Como el nivel de presión sonora en decibelios es inversamente proporcional a la distancia, el margen deberá ser determinado con el nivel de presión acústica producido a 3 metros. Así, un nivel de presión acústica de 110 dB a 3 metros,

descenderá hasta 70 dB a 300 metros y 60 dB a 1800 metros.

El margen de frecuencia normal está entre 600 y 1500 Hz. El oido no es sensible de forma uniforme a todas las frecuencias, siendo la región más sensible entre 2 y 4 KHz. Sin embargo, como las frecuencias más altas son absorbidas más pronto sufren,a consecuencia de esto, otras pérdidas, la parte más baja de la región sensitiva es la mejor. Por consiguiente, los tonos desde 1,5 a 2 KHz dan una audibilidad mayor y más efectiva para sirenas de alarma.

II-2-1-2-4. SIRENAS ELECTRONICAS.

Como una alternativa de las sirenas movidas con motores convencionales, existen dispositivos electrónicos que consisten en un oscilador, un amplificador de potencia y un altavoz.

La generación del tono electrónico tiene la ventaja de ser versátil. La forma de onda puede ser conformada para dar un sonido penetrante, algo parecido a un grito, en definitiva, algo que puede ser dificilmente ignorado.

II-2-2-3. MARCADOR-TRANSMISOR DE ALARMAS.

II-2-2-3-1. DESCRIPCION GENERAL.

El marcador-transmisor de alarmas MTA /2 tiene como misión la transmisión de información de alarmas a través de la red telefónica conmutada. Para ello se conecta a dicha red por medio de una ó dos líneas convencionales, que comparte con otros tantos aparatos de abonado, con cuyo funcionamiento no interfiere en ausencias de alarma.

Por otra parte, se conecta a la instalación privada de seguridad por medio de cinco líneas de alarma. Además de las condiciones de alarma detectadas en éstas líneas puede transmitir tres alarmas técnicas generadas localmente (fallos de lineas y batería baja), otra de prueba local o remota y una alarma de intrusión producida cuando se intenta abrir el compartimento no accesible al usuario. Después del envío de información de alarmas puede transmitir la información analógica en banda de audio presente en las entradas de que dispone a tal efecto.

Para realizar las funciones citadas el MTA /2 dispone de un protocolo mediante el cual comunica con un centro receptor, al que transmite además un código de identificación de abonado.

El equipo permite programar dos números telefónicos de hasta nueve cifras, correspondientes a dos centros receptores, principal y reserva. La marcación se realiza mediante impulsos de apertura de bucle, con detección de tono de invitación a marcar.

El MTA /2 es capaz de vigilar en todo momento el estado de las líneas mediante sensores de tensión, corriente de línea y corriente de llamada asociadas a las mismas. Por ello es capaz de hacer la elección más acertada entre ambas, al iniciar cada ciclo de transmisión.

El equipo proporciona también las siguientes alarmas locales por medio de contactos libres de potencial normalmente abiertos o cerrados:

- Alarma de intrusión: se produce al intentar abrir el compartimento que contiene los circuitos. Simultáneamente se transmite esta alarma por el canal 3 de servicio.
- Alarma técnica: se produce por fallo de alguna de las líneas telefónicas.

Junto con la alarma técnica local, el MTA /2 posee otros medios para comprobar su correcto funcionamiento:

- Indicador luminoso de que se está realizando

la alimentación a partir de la red de distribución eléctrica.

- Indicadores luminosos de nivel de batería y estado de ambas líneas telefónicas.
- Toma para auricular monitor de línea.
- Pulsador de alarma de prueba.
- Posibilidad de realizar una prueba remota de su funcionamiento por medio de una secuencia de llamada programable.

Todas las opciones disponibles en el MTA /2 son facilmente programadas mediante inserción de tornillos en una matriz o por puentes enchufables.

El MTA /2 se alimenta a partir de la red de distribución eléctrica e incluye una batería que asegura el funcionamiento ininterrumpido en reposo de 72 horas en ausencia de red.

El MTA /2 dispone de un microprocesador de bajo consumo para lograr un alto grado de integración y flexibilidad en la circuitería lógica, junto con la posibilidad del funcionamiento con baterias por períodos prolongados.

El equipo va alojado en un armario metálico cuya puerta frontal lleva cerradura de seguridad. Interiormente se divide en dos compartimentos, uno inferior, accesible al usuario median

El MTA /2 está equipado para detectar tanto el tono de invitación a marcar enviado por la central telefónica como los tonos de señalización del receptor de alarmas.

El tono de señalización enviado por el receptor, de una frecuencia nominal de 2100 Hz, se utiliza para tres propósitos diferentes a lo largo de la secuencia operativa del aparato: identificación, confirmación y desconexión. Tanto estos tonos como el de invitación a marcar necesitan, para ser dados por correctos, tener una duración mínima, y además, los de confirmación y desconexión no deben superar una duración máxima, con objeto de dar mayor seguridad a la detección y proteger contra llamadas maliciosas.

Puesto que la detección del tono de desconexión se realiza durante la emisión de información de audio por la vía de escucha, existe la posibilidad de detectar falsos tonos procedentes de dicha vía. Para evitarlo, durante la detección de dicho tono, y una vez transcurridos 240 milisegundos de su detección, se corta la vía de audio hasta que se detecte como correcto el tono, 300 milisegundos después, o hasta que se detecte ausencia de tono antes de dicho tiempo. En éste último caso el tono es ignorado.

te llave de seguridad, y otro superior, no accesible.

El compartimento inferior contiene las regletas de conexión a la red eléctrica, a las líneas telefónicas y a la instalación privada de seguridad, así como elementos de mantenimiento, como fusibles e indicadores. El compartimento superior contiene los circuitos electrónicos, y está alojado en una parte móvil o compartimento extraible.

II-2-2-3-2. DESCRIPCION FUNCIONAL.

La matriz de programación contiene los dos elementos que componen los dos números telefónicos de los receptores., el código de identificación del transmisor, los números máximo y mínimo de golpes de llamada y retardo máximo entre series de golpes para reconocimiento de prueba remota, criterio adoptado para cada canal de alarma de servicio, información sobre el tiempo de espera de los tonos de invitación a marcar e identificación y sobre la existencia de la segunda línea telefónica.

Estas informaciones son leídas una sola vez despueés de accionar el interruptor de encendido, por lo que cualquier cambio en la matriz de programación durante el funcionamiento del aparato será ignorado por éste.

Los tipos de alarma que maneja el MTA /2 se sistematizan en la siguiente tabla:

FUENTE DE LA ALARMA	TIPO DE ALARMA	CODIGO
CANAL 1	DE SERVICIO	1/11
CANAL 2	DE SERVICIO	2/22
CANAL 3	DE SERVICIO	3/33
CANAL 4	DE SERVICIO	4/44
CANAL 5	DE SERVICIO	5/55
FALLO LINEA 1	TECNICA	6
FALLO LINEA 2	TECNICA	7
BATERIA BAJA	TECNICA	8
PRUEBA LOCAL	DE PRUEBA	9
PRUEBA REMOTA	DE PRUEBA	9
FALLO LINEA 1 ó 2	TECNICA LOCAL	LOCAL
INTRUSION	LOCAL	LOCAL
INTRUSION	DE SERVICIO	3

Las alarmas de servicio, técnicas y de prueba son transmitidas a la línea mediante los códigos que figuran en la tabla.

Las alarmas de servicio (canales 1 a 5) se ajustan a las denominadas "caracteristicas técnicas". La condición de reposo nominal es una resistencia de 15 kohmios. Se distinguen dos condiciones de alarma diferentes: La correspondiente

a resistencia alta (mayor de 45 kohm) y la correspondiente a resistencia baja (menor de 5 kohm). Según el criterio que se haya programado en la matriz para cada canal, en uno de los dos casos se transmite el código correspondiente dos veces seguidas. En tal caso se interpretará la información como de avería.

La alarma de batería baja se produce en cuanto se detecta una tensión de batería por debajo de lo especificado.

La alarma de prueba local se produce cuando se pulsa el botón correspondiente, situado en el compartimento de usuario del equipo.

La alarma de prueba remota se produce tras la detección de una secuencia de dos series de golpes de corriente de llamada, que cumpla las siguientes condiciones: que el número de golpes de cada serie esté comprendido entre un máximo y un mínimo programados en la matriz, y que el intervalo de tiempo entre ambas series no supere un máximo, también programado en la matriz.

La información de alarmas se almacena, cuando alguna se activa, en una cola. Además de los códigos de los códigos de las alarmas, la cola contiene información que indica si cada alarma permanece activa y si se ha transmitido. La alarma

de prueba no se almacena en la cola, sino en una posición específica, dado su peculiar funcionamiento.

Cuando se produce cualquier alarma que no sea de prueba, la alarma de prueba, sea local o remota, es desactivada incondicionalmente de modo que si no se ha transmitido no llegará a realizar el envío.

En el momento de la secuencia operativa del transmisor en que ha de verificarse la transmisión de alarmas, ésta se hace en el mismo orden en que se han almacenado, transmitiéndose primero los códigos de las alarmas aún no transmitidas, a continuación el de prueba, si estuviera activa, y por último los correspondientes a las alarmas ya transmitidas que permanezcan activas.

El número máximo de códigos (cifras) que pueden transmitirse en cada ciclo del transmisor es de 9. La información de alarmas aún no transmitidas que no pueda por este motivo enviarse en un ciclo se enviará en otro ciclo generado inmediatamente a continuación.

Tras la recepción del tono de confirmación se borra de la cola la información concerniente a las alarmas ya transmitidas e inactivas.

El MTA /2 es capaz de vigilar en todo momento el estado de las líneas mediante sensores de tensión, corriente de línea y corriente de llamada asociada a las mismas. Por ello es capaz de hacer la elección más adecuada entre cada una de las dos líneas que lo equipan, al iniciar cada ciclo de transmisión.

Hay que distinguir a tal efecto dos casos: el ciclo primario, que es aquél directamente originado por la presencia de alarmas, y el ciclo secundario, que se origina al interrumpirse un ciclo primario por algún fallo en la secuencia prevista.

En el caso del ciclo primario, la elección depende de la condición de libre u ocupada de la línea previamente a la exclusión de los teléfonos asociados, y del posible estado de fallo, de acuerdo con la siguiente tabla, y teniendo en cuenta que la condición de ocupada se activa también al detectarse corriente de llamada en la línea.

ESTADO LINEA 1	ESTADO LINEA 2	LINEA ELEGIDA
LIBRE	CUALQUIERA	1
OCUPADA	LIBRE	2
OCUPADA	OCUPADA	1
OCUPADA	EN FALLO	1
EN FALLO	LIBRE	2
EN FALLO	OCUPADA	2
EN FALLO	EN FALLO	1

En el ciclo primario se marca siempre el primer número telefónico (el programado en la segunda zona de la matriz).

En el caso de ciclo secundario, se va cambiando alternativamente de línea y de número a marcar, con las siguientes peculiaridades:

a) Si el fallo en la secuencia de ciclo primario se produce por no detección del tono de identificación tras la marcación, en el siguiente ciclo (secundario) se cambia de número.

b) Si el fallo en la secuencia del ciclo primario se produce por no detección del tono de invitación a marcar, en el siguiente ciclo se cambia de línea.

c) Si en una de las líneas se detecta fallo y en la otra no, se intentará por ésta última

incluso en el caso descrito en b).

En el caso de que el MTA /2 vaya conectado a una sola línea (la línea 1) y que tal circunstancia esté programada en la matriz, se tomará siempre dicha línea, pero en los cíclos secundarios esta toma se hará tras un cuelgue de 120 segundos para evitar que una llamada entrante pueda bloquear la línea.

II-2-2-3-3. SECUENCIA OPERATIVA.

A continuación se detalla la secuencia operativa del Marcador, distinguiendo entre los Ciclos Primario y Secundario.

Al accionarse el interruptor de encendido se efectúa una espera de 350 milisegundos para dar tiempo a que los circuitos del transmisor alcancen su régimen permanente, y se lee la matriz de programación.

A continuación se pasa a un estado de espera de condición a transmitir. De este estado se sale cuando se detecta una o varias alarmas de acuerdo con lo especificado en Descripción Funcional, lo que provoca la iniciación de un ciclo primario.

A continuación se efectua la elección de número y línea de acuerdo con los criterios expuestos en el apartado anterior, para el caso de

ciclo primario. Una vez hecha la elección, se actúa el relé de exclusión, lo que produce la desconexión de los teléfonos asociados y se hace una espera de 3 segundos.

Finalizado este período de cuelgue preventivo, se realiza el descuelgue sobre la línea seleccionada, se da alimentación a los circuitos de audio del aparato, y se pasa al estado de espera de tono de invitación a marcar. De este estado se sale, bien por cumplirse el período de espera (12 ó 24 segundos según se haya programado en la matriz) sin detectar el tono, bien por detección válida del mismo, de acuerdo con los criterios fijados en el apartado anterior. En el primer caso, si sólo una línea está equipada, se pasa a fase de marcación. Si se dispone de las dos líneas se simula un cuelgue y se inicia un ciclo secundario. En el caso de detección válida se pasa la fase de marcación.

En esta fase se generan sobre la línea los impulsos correspondientes al primer número telefónico programado (segunda zona de la matriz).

Terminada la marcación se pasa al estado de espera del tono de identificación del receptor de alarmas. En el caso de que transcurra el tiempo programado (25 ó 50 segundos) sin reci-

birse tono válido, se simula un cuelgue sobre la línea y se inicia un ciclo secundario.

Caso de recibirse tono válido durante el período de espera, de acuerdo con las condiciones expuestas en el apartado anterior, se pasa a la fase de envío de información. La información enviada en primer lugar es la correspondiente al código de usuario, incluyendo cuatro dígitos y otro de protección (suma módulo 10). A continuación se transmiten los códigos de alarma, y por último se envía un cero (diez impulsos) como indicador de fin de transmisión.

Terminada la transmisión se pasa al estado de espera del tono de confirmación del receptor de alarmas. En el caso de que transcurran 45 segundos sin recibirse tono válido, o de que dentro de dicho tiempo se reciba un tono de frecuencia y nivel correspondiente pero de duración superior a 1,28 segundos, se simula un cuelgue temporizado sobre la línea y se inicia un ciclo secundario.

Caso de recibirse tono válido durante el período de tiempo de espera, se pasa a eliminar de la cola de alarmas todas las transmitidas e inactivas, se vuelven a conectar los teléfonos asociados, y se activa la vía escucha, pasándose a

la fase de transmisión de información en banda vocal.

De esta fase se sale, bien por vencimiento de la temporización establecida (13 minutos) sin haberse recibido tono válido del receptor, bien por recepción de dicho tono antes del citado plazo con las condiciones expuestas en el apartado anterior. Si se recibe durante esta fase un tono de duración mayor que la máxima prevista de 1,28 segundos, el aparato permanece en fase de escucha hasta que cumpla la temporización o se reciba tono válido.

Una vez finalizado el período de escucha, se abre el bucle durante 3,4 segundos y a continuación se corta la alimentación de los circuitos de audio y se prolongan las líneas sobre los teléfono asociados. Este modo de finalizar la llamada impide la retención de la línea y el consiguiente bloqueo temporal del receptor en el caso de que el teléfono asociado a la misma se encuentre descolgado.

El ciclo secundario sigue las mismas fases que el primario, comenzando con el cuelgue temporizado de línea, excepto en lo siguiente:

a) Transcurrida la espera del tono de invitación a marcar sin haber recibido tono válido, el transmisor pasa a la fase de marcación.

b) La elección de línea y número telefónico se hace de acuerdo con los criterios expuestos en el apartado anterior.

c) Si sólo se dispone de una línea telefónica se hace de acuerdo a un cuelgue temporizado de 120 segundos para evitar el bloqueo de la línea por llamadas entrantes.

Cuando la receptora recibe un código no válido, no acepta la llamada, lo que obliga al Marcador-Transmisor a repetirla, iniciándose así un ciclo secundario. Si continúa la recepción de un código incorrecto el Marcador sigue intentando la comunicación, pudiendo dejar bloqueada la Receptora de Alarmas. Es preciso por tanto poner límite al número de intentos y al espaciamiento entre ellos.

Las probables causas del problema pueden ser, la avería del Marcador, defectos en la línea, errores repetitivos en la transmisión, etc.

Para solucionar este problema se ha dotado al Marcador-Transmisor de un contador, el cual, transcurrido un número de intentos de comunicación, activa una alarma local y pasa a ejecutar una espaciación de tiempo mayor en los intentos sucesivos. Este espaciamiento se fija en 15 minutos, lo cual es suficiente para no bloquear

la Central de Alarmas y es además un tiempo prudencial de inactividad del Marcador.

En el caso de fallo de línea, la comunicación no llegará a establecerse, pero los intentos se producirán de la misma forma activándose la extensión de Alarma Local "Fallo de línea"; dado que esta alarma activa el mismo relé que la que se produce cuando se pasa de 32 intentos, para discriminar entre ellos habrá que hacer uso del pulsador manual de prueba de línea.

II-2-3. UNIDAD DE CONTROL.

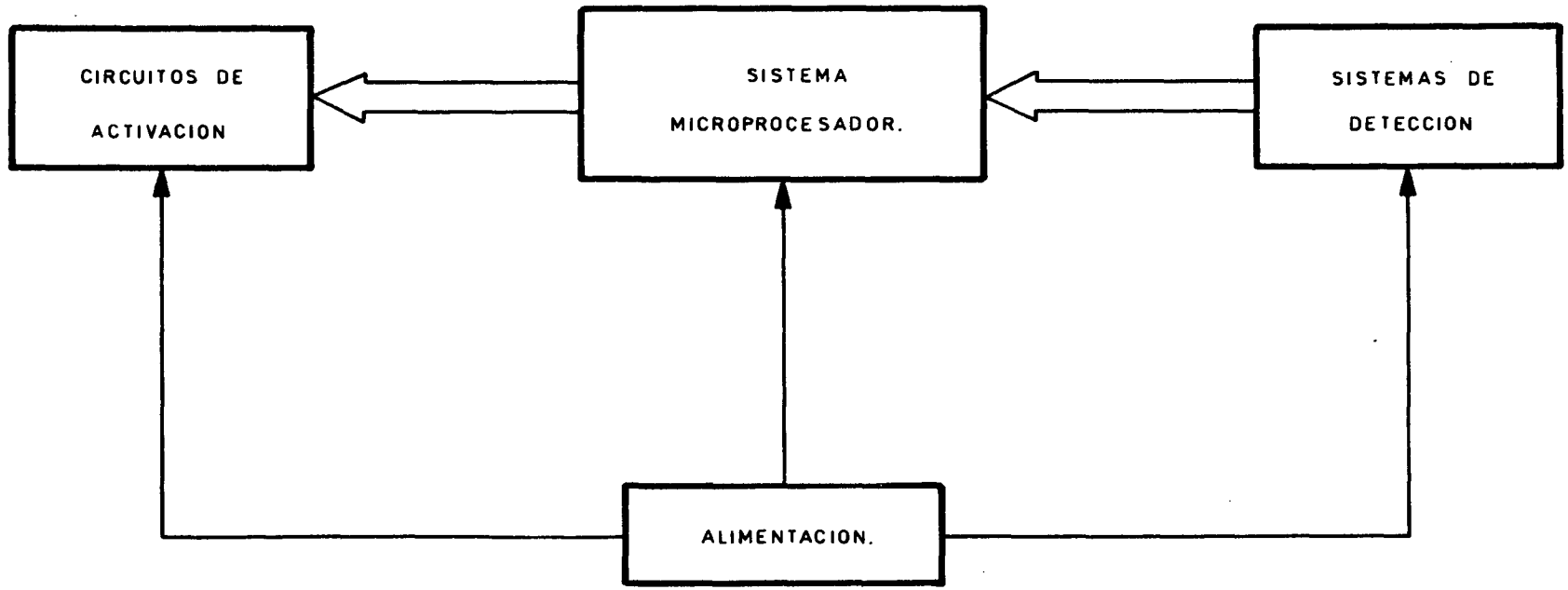
La unidad de control es el corazón de la totalidad del sistema de seguridad. Otras partes del sistema como pueden ser los dispositivos de detección o los circuitos de activación, varían poco y se encuentran en su mayoría estandarizados. Por ello, cuando se diseña un sistema de seguridad, la mayor parte del mismo va encaminado, generalmente, al estudio y puesta en funcionamiento de la unidad de control, ya que ésta determina la flexibilidad, la eficacia e incluso la fiabilidad del sistema.

La unidad de control tiene como misión recibir la información procedente de los diversos dispositivos de detección, interpretar dicha información y enviar las señales de alarma correspondientes.

En un sistema de seguridad, la unidad de control utilizada puede ser de dos tipos, o bién del tipo tradicional o bien microprocesada. Las ventajas que supone el empleo de un microprocesador como elemento de control frente a un sistema tradicional son las siguientes:

- Mayor sencillez en cuanto al diseño del circuito hardware.
- Mayor flexibilidad en posteriores aplicaciones.
- Un cambio en la concepción del sistema solo implica un cambio de programa.

Figura 2



- UNIDAD DE CONTROL MICROPROCESADA -

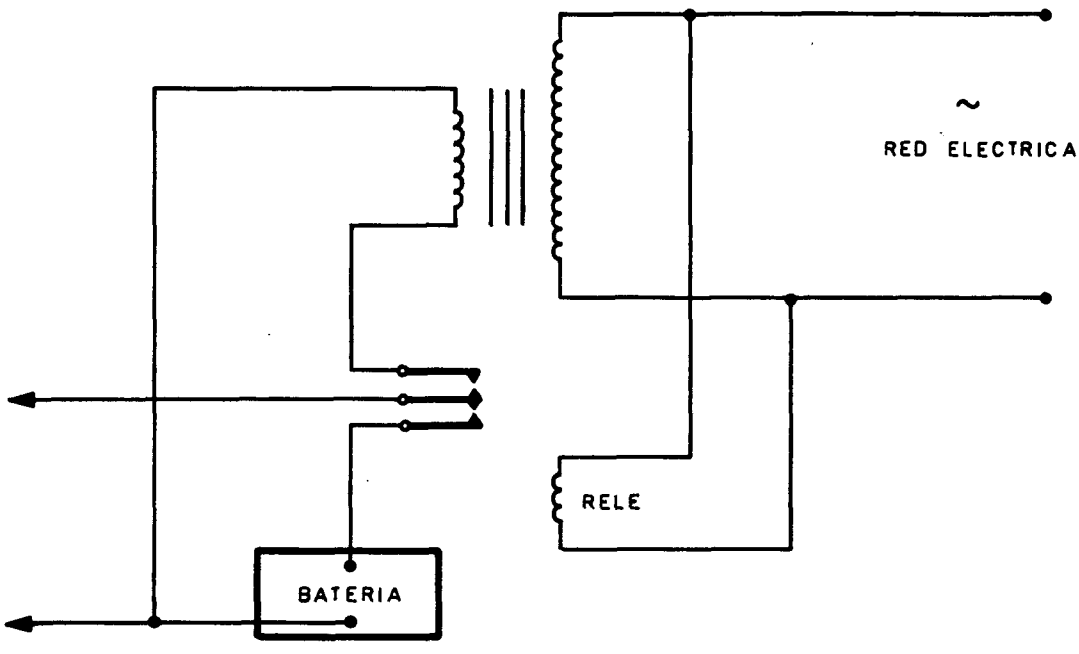
II-2-4. FUENTE DE ALIMENTACION.

Un sistema de seguridad debe de contar con una fuente de alimentación constante y fiable, de forma que pueda mantener el sistema activo en momentos en que no haya corriente eléctrica en la red. Se necisita, por tanto, una fuente de alimentación que no se interrumpa. Esto es proporcionado por baterias.

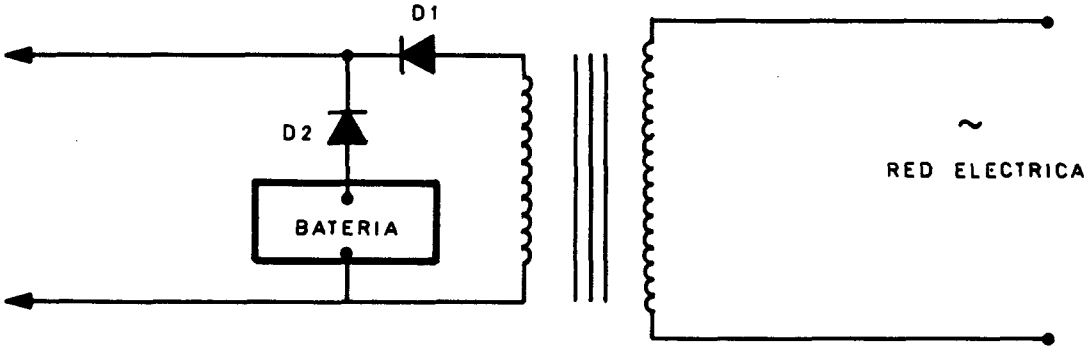
Si la corriente eléctrica de la re falla, el sistema deberá conectarse automáticamente a la batería de reserva. Esto se efectúa, generalmente, por medio de un relé. La bobina activa del mismo se conecta directamente a través de la entrada de la red, lo que significa que cuando el voltaje está presente, el relé se activará. En esta posición, los contactos del relé conectan el sistema de seguridad al secundario del transformador de red de donde se alimenta. Si desaparece el voltaje de la red, el relé no estará activado más tiempo y se conmutará a los contactos de reposo. Este conecta el circuito a la batería que, de éste modo, se encarga de alimentar al sistema. Cuando se restablece la alimentación de la red, el relé se activa de nuevo y el sistema se conecta automáticamente en las operaciones de alimentación.

Existe otro método de conmutación que elimi

Figura 3



CIRCUITO CONMUTADOR POR RELE



CIRCUITO CONMUTADOR POR DIODO

na completamente la utilización de un relé. Consiste en utilizar un par de diodos como interruptores. En éste caso, el voltaje procedente del transformador debe de ser ligeramente más alto.

Cuando se trabaja con la alimentación de la red, la corriente fluye del transformador a través de D1 (ver figura) al circuito de alarma. La corriente no puede pasar a la batería, debido al diodo D2. Como el voltaje procedente del transformador es más alto que el voltaje de la batería, D2 está polarizado inversamente, lo que significa que no pasa corriente en sentido directo. De ésta forma, la batería está aislada y la corriente no pasará desde ésta al sistema de seguridad.

Si la alimentación de la red desapareciera, la polarización inversa de D2 disminuiría y la corriente pasaría por el circuito externo. Esta no pasará a través del secundario del transformador, porque está impedido por D1.

PARTE - III

SISTEMA DE SEGURIDAD

III-1. SITUACION DEL SISTEMA DE SEGURIDAD.

El sistema de seguridad está concebido para una vivienda unifamiliar de aproximadamente 350 metros cuadrados y situada en una zona residencial a las afueras de la ciudad. Esta ubicación de la vivienda la hace especialmente atractiva para posibles ladrones, debido principalmente a dos motivos.

En primer lugar, debido a que por regla general, en una zona residencial viven personas adineradas que poseen objetos de valor en el interior de sus viviendas. En segundo lugar, porque éstas zonas son poco transitadas y la entrada de un intruso en una de las viviendas puede pasar completamente desapercibida.

Se hace por tanto imprescindible dotar a estas viviendas de fiables sistemas de seguridad que impidan la entrada en ellas de personas no autorizadas.

Por otro lado, el sistema de seguridad cuenta también con un sistema de detección automática de incendios, que asegura la rápida localización de un incendio en el interior de la vivienda desde el momento en que éste se origina, asegurando por tanto, una rápida intervención en la extinción del mismo.

III-2. PRESTACIONES DEL SISTEMA.

La característica principal que posee éste sistema es su flexibilidad. Esta viene dada por la utilización de un sistema microprocesador como unidad de control. Esto permite un sistema con funciones muy variadas, tales como llamadas autoráticas al Servicio de Extinción de Incendios ó la activación de un indicador luminoso, correspondiente a un determinado sensor.

Una vez activado el sistema, éste comprobará secuencialmente el estado en el que se encuentran todos y cada uno de los sensores. Una vez hecho esto, comparará los estados recién adquiridos (estado actual) con aquellos que fueron tomados en la anterior lectura (estado previo), procesando seguidamente cualquier diferencia existente entre ambos estados.

El sistema procesará cualquier cambio de estado que se produzca, tanto si se pasa del estado RESET a SET como si lo hace de SET a RESET. Esta característica hace posible la instalación de diferentes tipos de sensores. Tanto si estos pasan al estado SET al ser activados como si lo hacen al estado RESET.

Pero quizás, la característica más destacada del sistema sea que los sensores son trata

dos con total independencia unos de otros. Es decir, cada sensor es procesado individualmente, teniendo cada uno de ellos unas respuestas asociadas concretas. Por ejemplo, supongamos que se activan dos sensores anti-intrusos. El primero de ellos puede dar, por ejemplo, las siguientes respuestas: encendido de un indicador luminoso, dando la posición exacta del sensor activado; la activación de una sirena de alarma y la llamada automática a la comisaría de policía más cercana. Por el contrario, la activación del segundo de los sensores puede dar respuestas completamente diferentes a las dadas por la activación del primero, como por ejemplo, la puesta en funcionamiento de una cámara de circuito cerrado de televisión.

Esta característica confiere al sistema de seguridad una gran versatilidad, ya que las respuestas dadas por el sistema serán independientes para cada sensor, consiguiendose así, una combinación de respuestas que aumentarán la eficacia del sistema de seguridad.

III-2-1. GESTION DE ALARMAS.

El sistema considera tres tipos de alarmas:

- NIVEL 1.

Activación. Las alarmas del nivel 1 están activadas siempre que el sistema esté operativo. Esta activación se lleva a cabo mediante cerradura de seguridad en la unidad de control. Las alarmas de éste nivel son proporcionadas por detectores de humo y calor.

Respuestas. Se produce una alarma sonora en el interior de la vivienda. Se inician llamadas automáticas al Servicio de Extinción de Incendios, conectándose un indicador luminoso que corroborará que la llamada se ha producido. Se activa un indicador luminoso parpadeante que proporciona la localización exacta del sensor que ha sido activado. Los indicadores luminosos permanecerán activados hasta que el sistema sea reinicializado.

- NIVEL 2.

Activación. Las alarmas de éste nivel se activan mediante cerradura de seguridad en el exterior de la vivienda y son utilizadas para la detección de intrusos. Esta activación en el exterior

de la vivienda asegura una protección máxima ya que no se precisarán tiempos de retardo de activación de alarma, que podrían dar tiempo a los intruso a llegar a la unidad de control, que en esos momentos se encuentra desprotegida.

Esta inexistencia de temporizaciones asegura así mismo, una puesta en funcionamiento inmediata de la alarma en caso de entrada en la vivienda de personas no autorizadas.

Respuestas. Se produce una alarma sonora en el exterior de la casa por espacio de 4 minutos (el máximo tiempo permitido por la legislación española es de 5 minutos). El sistema llama automáticamente a la comisaría de policía, conectándose un indicador luminoso que corroborará que la llamada se ha producido. Se activa el indicador luminoso correspondiente al sensor activado. Los indicadores luminosos serán siempre parpadeantes.

- NIVEL 3.

Activación. Este nivel permanece siempre activado y tiene como misión detec

tar cualquier manipulación que se haga a cualquier componente del sistema de seguridad.

Respuestas. Las respuestas dadas por el sistema ante la activación de una alarma del nivel 3 son las mismas que las dadas para el nivel 2.

Los indicadores luminosos de los tres niveles de alarma permanecerán activados hasta que no se reinicialice el sistema. Esta reinicialización se efectua mediante llave de seguridad en la unidad de control del sistema.

III-2-2. LOCALIZACION DE LOS SENSORES.

Para facilitar una rápida localización del lugar donde se ha producido la alarma, se divide la vivienda en zonas, correspondiendo cada una de ellas a las distintas dependencias de la casa.

Todos los sensores que correspondan a un mismo nivel que se encuentren situados en una misma zona tendrán una entrada común en la unidad de control.

Las zonas estarán divididas de la siguiente forma:

- NIVEL 1.

1. Cocina.
2. Comedor.
3. Salón.
4. Habitación 3.
5. Garaje.
6. Habitación 1.
7. Habitación 2.

- NIVEL 2.

1. Comedor.
2. Cocina.
3. Lavadero.
4. Puerta Principal.
5. Salón.
6. Habitación 3.
7. Habitación 2.
8. Habitación 1.
9. Garaje.

- NIVEL 3.

1. Linea anti-sabotaje.

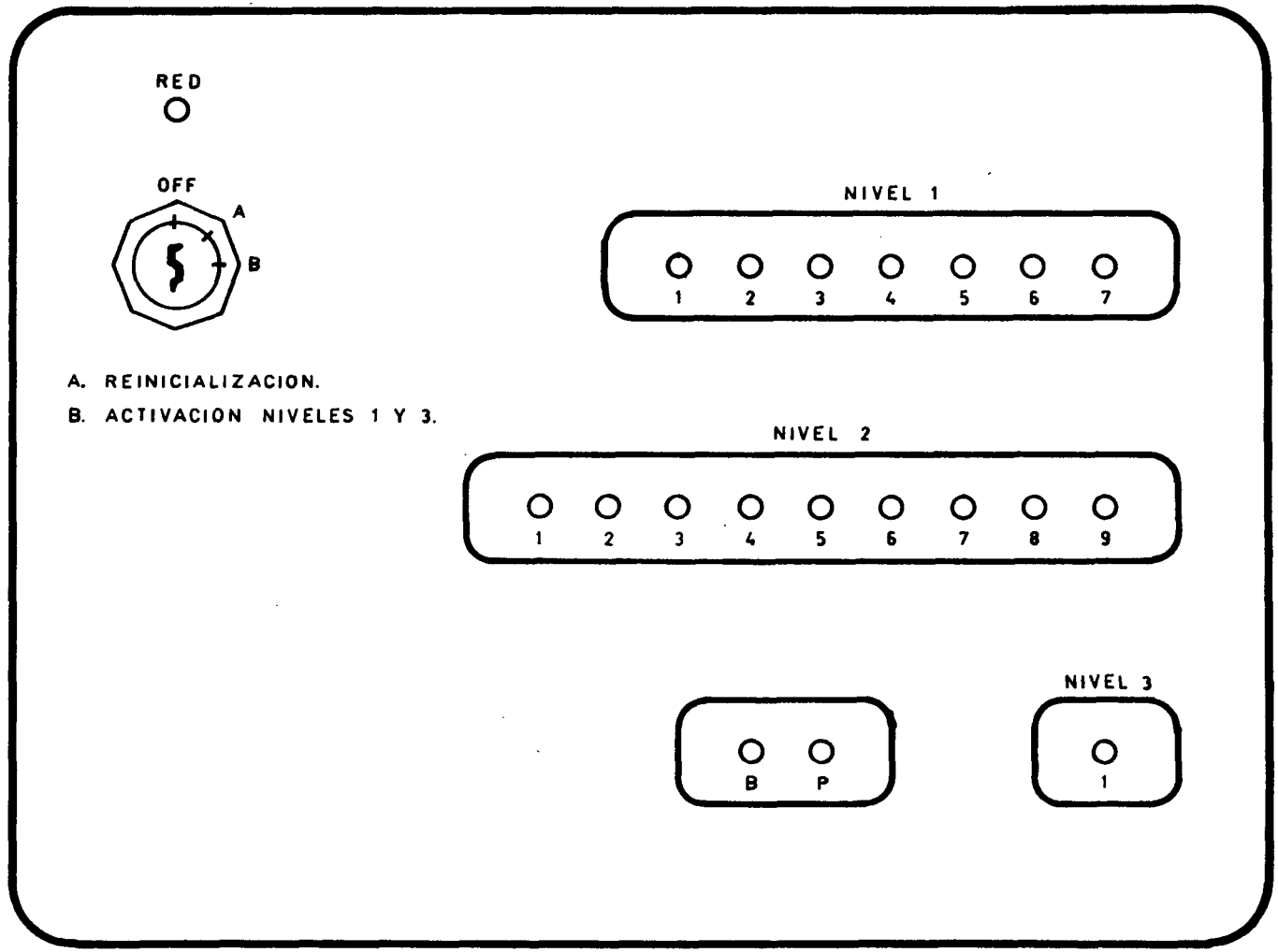
Los sensores empleados por el nivel 1 serán termovelocimétricos, detectores de humo por ionización y detectores térmicos. Emplean-

dose estos en los lugares adecuados siguiéndo se como criterio para ello lo expuesto en el a partado II-2-1-3.

Los sensores utilizados para detección de anomalías del nivel 2 serán detectores de infrarrojos e interruptores magnéticos. Los interruptores magnéticos irán instalados en to das las puertas y ventanas que comuniquen con el exterior. Los detectores de infrarrojos i- rán instalados en aquellos lugares de paso obli gado para cualquier intruso, consiguiendose así una protección adicional en el caso, bastante improbable, de que algún interruptor magnético sea inutilizado. Estos lugares de paso obligado son: el salón de la vivienda, la habitación principal (señalada aquí como habitación 1) y los pasillos de la vivienda.

Los dispositivos de detección empliados por el nivel 3 serán principalmente microrruptores. Estos estarán instalados en todos los dispositivos exteriores, como en el cajetín de la sirena de alarma ó en la cerradura de seguridad que se encuentra en el exterior de la vivienda. Desde el momento en que se manipulen los tornillos de fijación de cualquiera de éstas unidades, el microrruptor se disparará pro

Figura 4



vocando la alarma. De igual forma, todos los cables de conexión del sistema de seguridad llevarán aparejado un cable anti-sabotaje, de forma que si se lleva a cabo un corte en el cable de conexión, la alarma se dispara.

En la figura 4 se puede observar la composición del panel de control de la central de alarma. En él se pueden distinguir los señalizadores luminosos correspondientes a cada una de las zonas de alarma, divididas en sus niveles correspondientes. Dichos señalizadores estarán formados por LEDs de diferente colores según al nivel de alarma al que pertenezcan. Los LEDs pertenecientes al nivel 1, alarmas por incendio, serán de color verde. Los pertenecientes al nivel 2, alarmas por intrusos, serán de color rojo; y por último, el señalizador de alarma de nivel 3 será de color amarillo. Cuando cualquiera de estos señalizadores se active, emitirá luz intermitentemente. Esta intermitencia a parpadeo de los señalizadores luminosos redundará en una mejor percepción del estado de alarma por parte del usuario.

En el panel de control también irán situados dos señalizadores de iluminación fija. Uno será de color verde y señalado con la le-

tra B. y el otro será de color rojo y estará señalado con la letra P. Se tratan de los testigos de llamada telefónica al Servicio de Extinción de Incendios y a la Comisaría de Policía respectivamente. Hay también un señalizador de iluminación fija y de color rojo que se encuentra señalado con la palabra RED. Este está destinado a indicar al usuario que el sistema está siendo alimentado a través de la red eléctrica.

En la figura 4 puede observarse también una cerradura de seguridad de tres posiciones, marcadas éstas por OFF, A y B. Cuando se encuentra en la posición OFF, el sistema estará inoperativo. Al girar la llave hacia la derecha, posición A, se lleva a cabo la reinicialización del sistema, reseteando todos los dispositivos que lo componen. Por último, cuando se llega a la posición B, se activan los niveles de alarma 1 y 3. En cuanto al nivel de alarma 2, éste se activa mediante cerradura de seguridad, alojada en un cajetín en el exterior de la vivienda.

III-3. DESCRIPCION DE LOS CIRCUITOS.

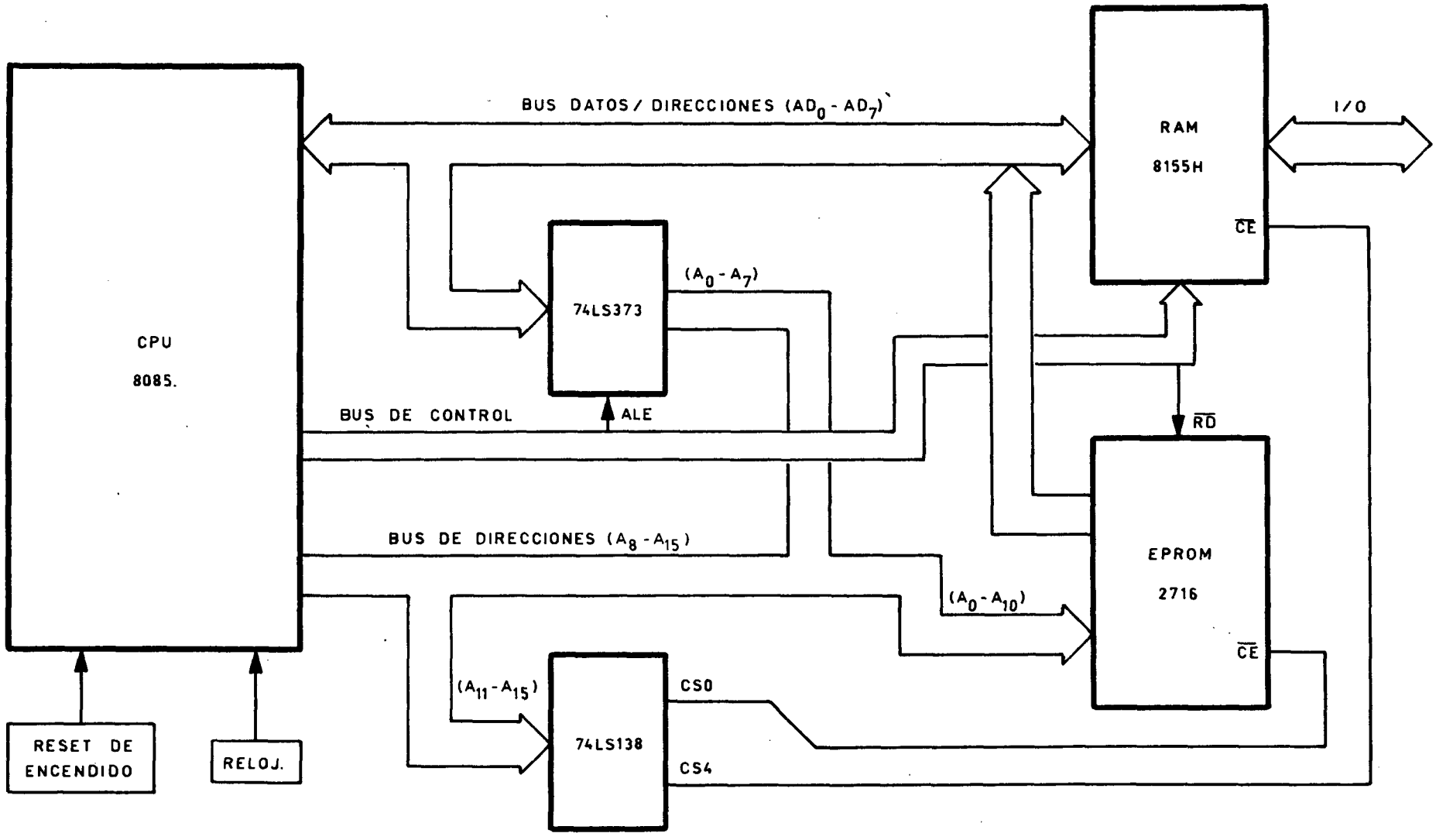
III-3-1. UNIDAD DE CONTROL.

La unidad de control del sistema de seguridad está constituida por un sistema mínimo microprocesador y por circuitos multiplexores y decodificadores asociados a éste. Los circuitos multiplexores tienen como misión recibir los estados en los que se encuentran los diversos dispositivos de detección, multiplexarlos y seguidamente transferirlos al sistema microprocesador para su posterior procesamiento. Una vez realizado éste, las respuestas que deben de producirse en caso de estado de alarma en alguno de los sensores, son decodificadas y enviadas a los diversos dispositivos de respuesta.

III-3-1-1. SISTEMA MINIMO MICROPROCESADOR.

Se encarga éste de procesar toda la información recibida a través de los multiplexores, para dar posteriormente las respuestas adecuadas. El diagrama de bloques de éste sistema se encuentra en la figura 5. En ésta se pueden observar los diferentes dispositivos que lo forman, así como los buses de conexionado con los mismos.

- Figura 5 -



III-3-1-1-1. UNIDAD CENTRAL DE PROCESOS.

La unidad central de procesos (CPU) está constituida por el microprocesador 8Ø85A de Intel. Se trata éste de un microprocesador de 8 bits de propósito general y con una capacidad de direccionamiento de 64 Kbytes. El 8Ø85A suministra las funciones de generación de señales de reloj, control de buses y selección de prioridad de interrupciones.

La transferencia de datos hacia ó desde el exterior del microprocesador se realiza a través de un bus bidireccional de 8 bits multiplexado, por el que también se transmiten los 8 bits de menor peso de la dirección. Un bus adicional de direcciones de 8 bits aumenta la capacidad de direccionamiento a 16 Kbytes de memoria. Los buses de datos y direcciones presentan configuración triestado, es decir, alta impedancia, Ø y 1. La CPU 8Ø85A genera además señales de control que permiten seleccionar dispositivos externos y funciones para ejecutar operaciones de lectura y escrutura y seleccionar también memorias ó puertos de entrada / salida. Se pueden direccionar hasta 256 posiciones de E/S.

El microprocesador 8Ø85A contiene registros de 8 y de 16 bits. Hay 8 registros direccionables de 8 bits, 6 pueden utilizarse como registros de 8 bits ó como 3 registros de 16 bits.

Considerados individualmente, son referidos en el programa por los números:

Nº	Nombre del registro
Ø	B
1	C
2	D
3	E
4	H
5	L

También pueden ser referidos por el propio nombre del registro.

Considerados como registros dobles los números o nombres de registros asignados son:

Nº	Nombre de registro par	Registros
Ø	B	B,C
1	D	D,E
2	H	H,L

Estos registros son de uso general, se puede hacer referencia a ellos en las instrucciones del programa.

El contador de programas (PC) es un registro de 16 bits que contiene la dirección de memoria donde se encuentra la siguiente instrucción a ejecutar. Después de cada ciclo de búsqueda se incrementa automáticamente.

Una instrucción del 8Ø85 puede ocupar 1,2 ó 3 octetos. Las instrucciones de más de un octeto se almacenan en posiciones de memoria sucesivas, siendo la posición del primer byte el que marque la dirección de la instrucción. Se necesitan por tanto varios ciclos de búsqueda en memoria conel puntero PC para obtener la dirección completa.

El microprocesador 8Ø85 dispone también de otro registro de 16 bits denominado Puntero de Pila y que contiene la última dirección de la última posición ocupada en la pila.

La pila es una zona de memoria organizada como una cola LIFO (último en entran, primero en salir).

El puntero de pila puede ser inicializado con un valor para utilizar cualquier posición de la memoria como una pila. Su contenido es decrementado cada vez que un dato

se almacena en la pila y se incrementa cuando un dato se extrae de la pila. Los incrementos y decrementos son siempre de dos bytes debido a que todas las operaciones de pila se hacen sobre registros pares.

Los registros temporales son de de ocho bits y son destinados al uso interno de la máquina e inaccesibles al programador. Una de sus funciones por ejemplo, consiste en almacenar temporalmente la dirección de un operando cuando es preciso para su obtención efectuar dos cíclos de búsqueda en memoria.

El acumulador es un registro de ocho bits. Para hacer referencia a él en un programa como registro simple, se le atribuye el número 7 ó la letra A.

La unidad aritmética lógica es un subsistema que integra un conjunto de puertas y es capaz de realizar una serie de operaciones aritméticas, lógicas y de desplazamiento. Recibe los datos a través de dos buses de entrada y dispone de un bus de salida para los resultados. Estos resultados pueden ser depositados en el acumulador o ser transferidos al bus de datos internos.

El registro de indicadores es un registro que contiene 5 indicadores de un

bit que registran información a cerca del resultado de una operación. Los cinco indicadores son:

- El indicador de acarreo, CY: que se actualiza en las operaciones aritméticas. Es puesto a 1 si existe acarreo en el bit octabo del acumulador; si no lo hubiera toma el valor Ø. Sin embargo, si la operación es una recta, el comportamiento es el inverso: se pone a uno si no existe acarreo y a cero si lo hay. En definitiva indica si ha habido desbordamiento.
- El indicador de acarreo auxiliar, AC: se pone a 1 si se introduce acarreo en el bit cuarto; en caso contrario toma el valor Ø.
- Indicador de signo, F: Si el bit de signo del acumulador tiene valor 1 (dato negativo) el indicador se pone a 1, en caso contrario se pone a Ø.
- Indicador de cero, Z: si el resultado de una operación es cero, toma el valor 1. Si el resultado es distinto de cero se pone a Ø.
- Indicador de paridad, P: es puesto a 1 si la paridad del acumulador es par. Si es impar se pone a Ø.

La Unidad de Control dispone de:

- El registro de instrucción, es de ocho bits y recibe a través del bus interno el primer octeto de la instrucción que contiene siempre el código de operación.
- Decodificador de instrucciónes y generador de ciclos de máquina. Trabaja con el contenido del registro de instrucciones determinando las operaciones a realizar. La salida del decodificador controla los registros, la ALU y los buffers de datos y direcciones.
- Secuenciador, genera las micro-ordenes necesarias hacia dentro y hacia fuera del microprocesador en función de la información necesaria suministrada por el decodificador.
- Generador de señales internas del reloj. La CPU 8Ø85 lleva incorporado un generador de señales de reloj que requiere unicamente la adición de un cristal de cuarzo que establezca el secuenciamiento para su operación.

El microprocesador 8Ø85 dispone de cinco interrupciones hardware que pueden ser de tres tipos:

- Interrupción INTR: Provoca la búsqueda por parte de la CPU de una instrucción de reinicialización RST, externamente presentada a través del bus de datos por el periférico que realiza la interrupción. Es decir, cuando un periférico provoca la interrupción del microprocesador por medio de INTR, el periférico interruptor activa la señal INTR, y al mismo tiempo, coloca el código en el bus de datos. Este código indica al microprocesador interrumpido que instrucción RST tiene que ejecutar. La instrucción RST provoca un salto incondicional a una de ocho posiciones fijas de memoria. Para discernir a cual de esas ocho posiciones se debe de saltar, la instrucción RST va seguida de un número del Ø al 7. Cuando la instrucción RST es provocada externamente , mediante una interrupción, salta a una de las ocho posiciones de memorias mencionadas dependiendo del código que esté presente en el bus de datos. Cuando se recibe la instrucción RST, se produce un salto a la dirección de memoria donde se encuentra almacenado el programa de tratamiento de la situación

que provocó la interrupción, guardándose en la pila el contenido del contador de programa. En la dirección a la que salta se encuentra almacenada la dirección de salto al programa de tratamiento. La última instrucción del programa de tratamiento de la interrupción deberá ser una instrucción de retorno, de forma que continúe la ejecución del programa anteriormente interrumpido. La interrupción INTR es enmascarable, es decir, puede ser permitida o prohibida a través de las instrucciones software EI (interrupción permitida) y DI (interrupción prohibida). El microprocesador activa la señal INTA para decirle al periférico que ha reconocido la interrupción.

- Interrupciones RST 5.5, RST 6.5 y RST 7.5. Estas interrupciones hardware se caracterizan por el hecho de que cuando se producen, el procesador bifurca a una dirección fija de memoria donde se resolverá la situación que provocó la interrupción. Antes de que se produzca la bifurcación el contenido del contador de programa se almacena en la pila.

- Interrupción TRAP. No es enmascarable, se responde a la interrupción independientemente del estado de máscara o indicadores. Cuando se atiende el microprocesador bifurca a la posición fija de memoria 24H, guardando en la pila el contenido del contador de programa. La señal de entrada correspondiente a TRAP debe mantenerse en un nivel alto de tensión hasta que sea internamente reconocida.

El microprocesador 8085 dispone de entradas y salidas en serie. El sistema de E/S en serie por las lineas SID (dato entrada serie) y SOD (dato salida serie) es controlado por las instrucciones RIM y SIM. Cuando una instrucción RIM se ejecuta, el dato en la línea SID se carga en el bit 7 del acumulador. Por el contrario, si se ejecuta SIM, con el bit 6 del acumulador previamente puesto a 1, el contenido del bit 7 del acumulador se pondrá en la línea de salida SOD a través de un biestable interno.

III-3-1-1-2. DECODIFICADOR DE DIRECCIONES.

El decodificador de direcciones se encuentra conectado al bus de direcciones

(bits más significativos), cuya función es determinar a que subsistema se está dirigiendo la CPU, decodificando adecuadamente la información de entrada y generando en función de ésta una señal de validación de subsistema, de tal forma que, en el momento que la CPU coloque una dirección en el bus de direcciones, una y solo una de las memorias ó dispositivos de entrada/salida esté listo para recibir (o enviar) los datos del (o hacia) el bus.

Es decir, este subsistema establece el mapa de memoria, indicándonos que parte del espacio direccionable (64K) está ocupado por RAM (y cual es la dirección de la primera posición de memoria de éste integrado) que parte está ocupada por ROM (o EPROM) y cuales son las direcciones de los puertos de entrada/salida.

Por tanto, la dirección de una determinada posición de memoria se establece en el momento del cableado.

El decodificador de direcciones (74LS138) tiene ocho salidas que validan otras tantas pastillas, dichas pastillas cubren las siguientes posiciones de memoria al ser validadas:

SALIDA	RANGO DE DIRECCIONES
CSØ	ØØØØ - Ø7FF
CS1	Ø8ØØ - ØFFF
CS2	1ØØØ - 17FF
CS3	18ØØ - 1FFF
CS4	2ØØØ - 27FF
CS5	28ØØ - 2FFF
CS6	3ØØØ - 37FF
CS7	38ØØ - 3FFF

III-3-1-1-3. LATCH DE DIRECCIONES.

El microprocesador 8Ø85 utiliza un bus multiplexado de datos/direcciones que contienen los ocho bits menos significativos de la informacion de la dirección durante la primera parte del ciclo de máquina. El mismo bus contiene algún tiempo después los datos. El 8Ø85 genera una señal de habilitación de enclavamiento de direcciones (ALE) para ser usada por el integrado de latch de direcciones (74LS373) para enclavar la dirección de modo que esté disponible durante el ciclo de máquina completo.

III-3-1-1-4. MEMORIAS.

El programa en código utilizado

por el sistema de seguridad irá alojado en una memoria EPROM (memoria de solo lectura, programable electricamente y borrable mediante exposición a la luz ultravioleta). La memoria EPROM utilizada es la 2716 de Intel. La memoria 2716 funciona con una fuente de alimentación sencilla de 5 voltios y está diseñada para ser utilizada con los microprocesadores de Intel 8Ø85 y 8Ø86.

Por conveniencia, el programa en código se encuentra localizado a partir de la dirección ØØØØ del mapa de memoria, por lo que utilizaremos la salida CSØ del decodificador de direcciones para activar dicho dispositivo de memoria. La activación de la memoria 2716 se realiza a través de la patilla $\overline{CE}$ (chip enable) activa a nivel bajo. La habilitación de salida $\overline{OE}$ (output enable) es el control de salida y debe de usarse para leer los datos en las patillas de salida, independientemente de la selección del chip. La habilitación de salida ($\overline{OE}$) irá conexionada con la patilla de control $\overline{RD}$ del microprocesador 8Ø85.

La memoria 2716 dispone de 11 entradas de direcciones (AØ - A1Ø) mediante

las que se pueden direccionar 2Kbytes de memoria. Las direcciones AØ - A7 provienen del latch de direcciones (74LS373), mientras que las direcciones A8 - A1Ø provienen directamente del bus de direcciones más significativas del microprocesador. Las salidas de datos de la memoria (0Ø - 07) se conectarán directamente al bus multiplexado de datos/direcciones del microprocesador.

Los datos generados por la ejecución del programa del sistema se almacenarán en una memoria RAM (memoria de lectura y escritura). La memoria utilizada es la 8155H de Intel. Esta memoria dispone también de tres puertos de entrada y salida. Esta memoria es compatible con el bus multiplexado del microprocesador 8Ø85, por lo que no necesita tomar las entradas de direcciones del lach de direcciones. Por lo tanto esta memoria irá conectada al microprocesador mediante el bus de datos/direcciones (ADØ - AD7) y por el bus de control.

La habilitación del dispositivo se lleva a cabo mediante la patilla $\overline{CE}$ que se conectará a la salida CS4 del decodificador de direcciones. La salida CS4 corresponden-

de a la dirección 2ØØØ del mapa de memoria.

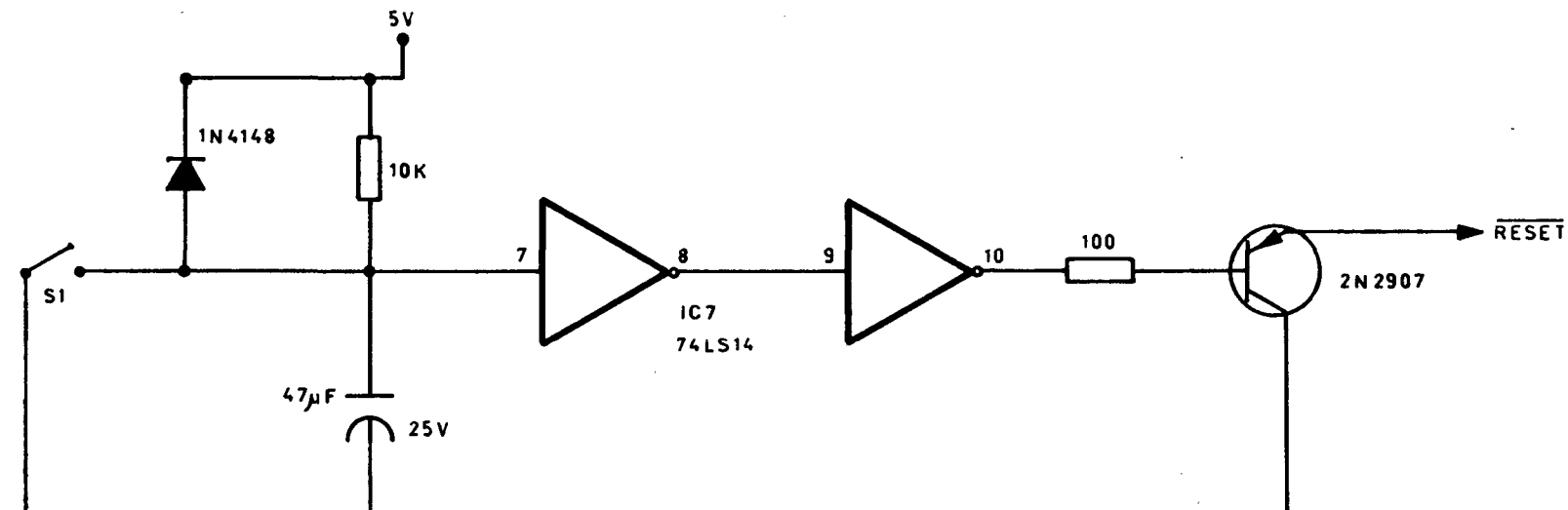
A través de los puertos de entrada/salida de los que dispone este dispositivo se llevarán a cabo tanto las lecturas del estado de los sensores como se realizarán las salidas que correspondan hacia los circuitos de activación.

III-3-1-1-3. RESET DE ENCENDIDO.

El RESET de encendido tiene como misión reinicializar el sistema colocando en el contador de programas la dirección ØØH. El programa en código del sistema de seguridad se encuentra alojado en memoria a partir de la dirección ØØH, por lo tanto, cuando se produce el RESET de encendido el programa comenzará a ejecutarse automaticamente.

En la figura 6 se muestra el circuito de un RESET manual y automático. Este circuito permite al sistema comenzar la ejecución del programa inmediatamente después del encendido. El programa puede ser detenido y reinicializado pulsando el botón de RESET, que en nuestro caso se trata de una llave de seguridad. Este reset de encendido no solamente actuará sobre el microprocesador, sino que

- Figura 6 -



- RESET DE ENCENDIDO -

también lo hará sobre el resto del sistema, evitando de ésta forma que se disparen las alarmas al ser conectadas.

El funcionamiento del circuito de RESET es elsiguiente. Cuando se conecta la alimentación, el condensador se encontrará descargado y hasta que éste alcance el régimen permanente habrá a la entrada del inversor (patilla 7) un nivel lógico Ø, esto provocará que la salida del segundo inversor (patilla 1Ø) se encuentre también a un nivel lógico Ø que propiciará el $\overline{\text{RESET}}$ de todo el sistema. Los inversores con Schmitt (74LS14) aumentan la fiabilidad del diseño. Cuando se desconecta la alimentación, el empleo de un diodo para descargar rápidamente el condensador, asegura que se genere un pulso si se vuelve a aplicar repentinamente la alimentación. El transistor 2N29Ø7 funciona como un 'switch' y asegura una buena provisión de corriente a todos los dispositivos a resetear.

III-3-1-2. DISPOSITIVOS DE ENTRADA/SALIDA.

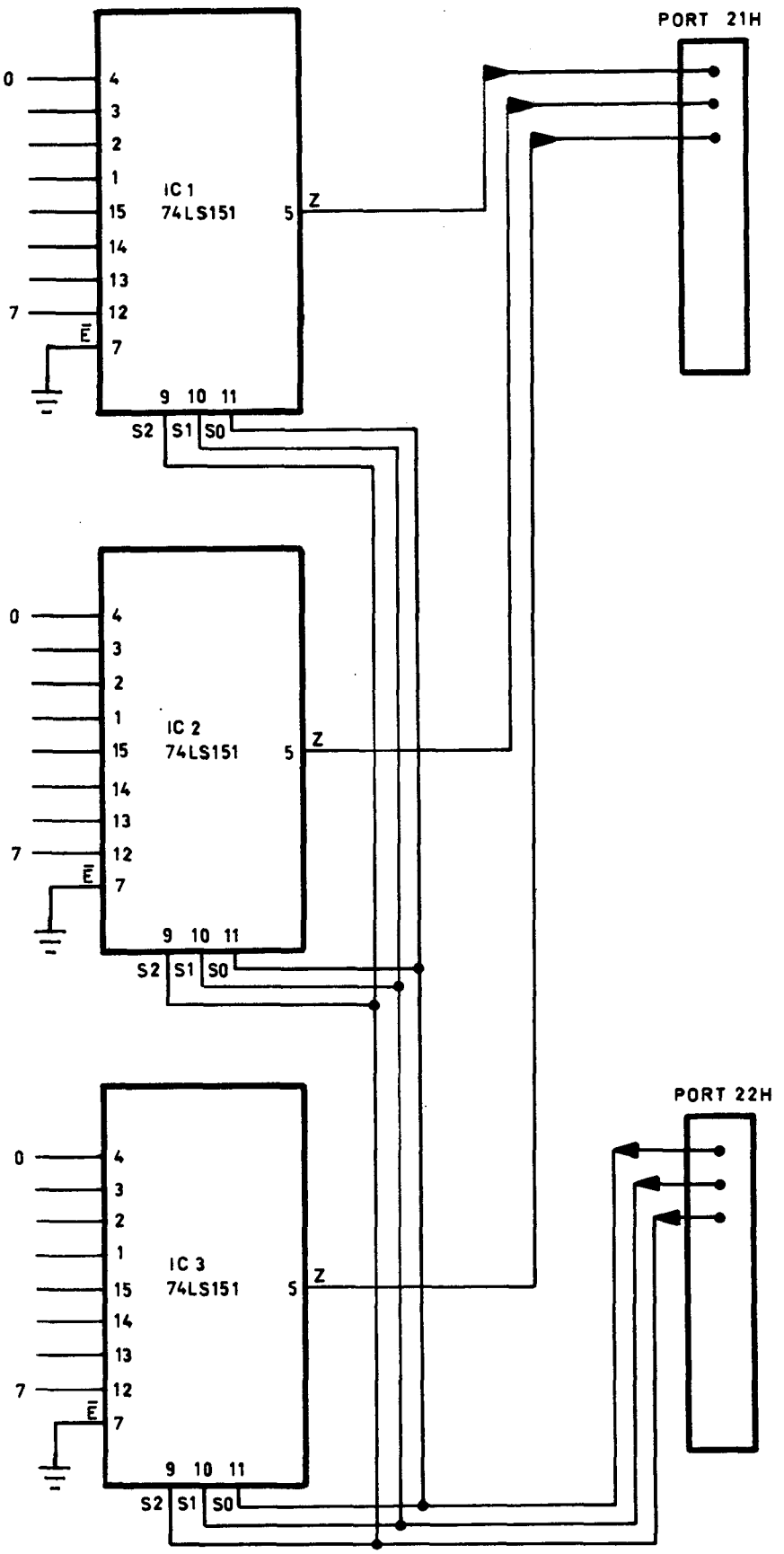
El sistema dispone de tres puertos de entrada/salida para su conexión con el exterior. Estos puertos se encuentran integrados en la memoria RAM 8155H, siendo dos de ellos de 8 bits y el tercero de 6 bits.

Es evidente que éste número tan limitado de entradas/salidas es insuficiente para un sistema de seguridad que debe de evaluar un número importante de dispositivos de detección y al mismo tiempo realizar la activación de señales de alarma en función de los sensores que hayan sido activados. Por ello, tanto las entradas como las salidas deben de ser multiplexadas para obtener así un mayor número de conexiones con el exterior.

Para llevar a cabo el multiplexado de entrada se utilizan tres circuitos integrados multiplexores 74LS151. Cada uno de estos dispositivos dispone de ocho entradas digitales cuyo estado lógico será transferido a una única salida en función de la combinación lógica en la que se encuentren sus entradas de selección.

Tal y como se puede ver en la figura 7, se utilizan los puertos 21H y 22H como

Vcc = Pin 16
GND = Pin 8

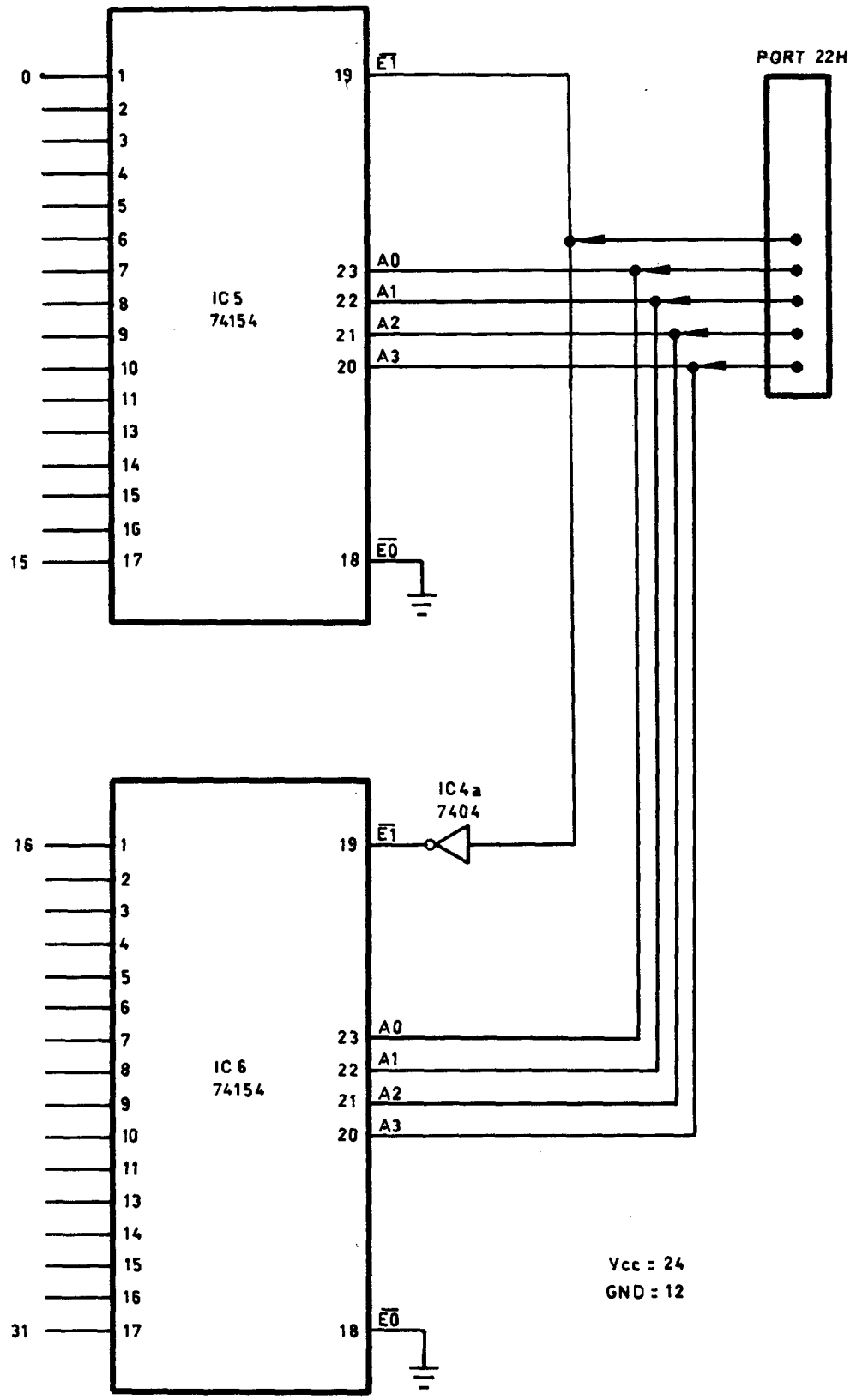


- Figura 7 -

entrada y salida respectivamente. A través del port 22H se envian las codificaciones de selección a los multiplexores y las salidas de estos se conexionan con el puerto de entrada 21H. Este puerto, por tanto, obtendrá las lecturas del estado en que se encuentran los diversos dispositivos de seguridad que se encuentran instalados en la vivienda. Los códigos de selección se presentan conjuntamente en los tres multiplexores, lo que redunda en una mayor rapidez de lectura. Estos códigos de selección son generados por una subrutina del programa del sistema. Esta subrutina realiza la codificación de Ø a 7, lo que asegura que los niveles lógicos de todas las entradas de los multiplexores sean transferidas a la salida del mismo.

Para efectuar el multiplexado de salida se utilizan dos circuitos integrados decodificadores 74154 tal y como se muestra en la figura 8. Cada uno de estos decodificadores dispone de 16 salidas que permanecerán a nivel lógico alto hasta que una de ellas sea seleccionada mediante cuatro entradas de selección. Cuando esto ocurre, la salida seleccionada pasa a nivel bajo.

- Figura 8 -



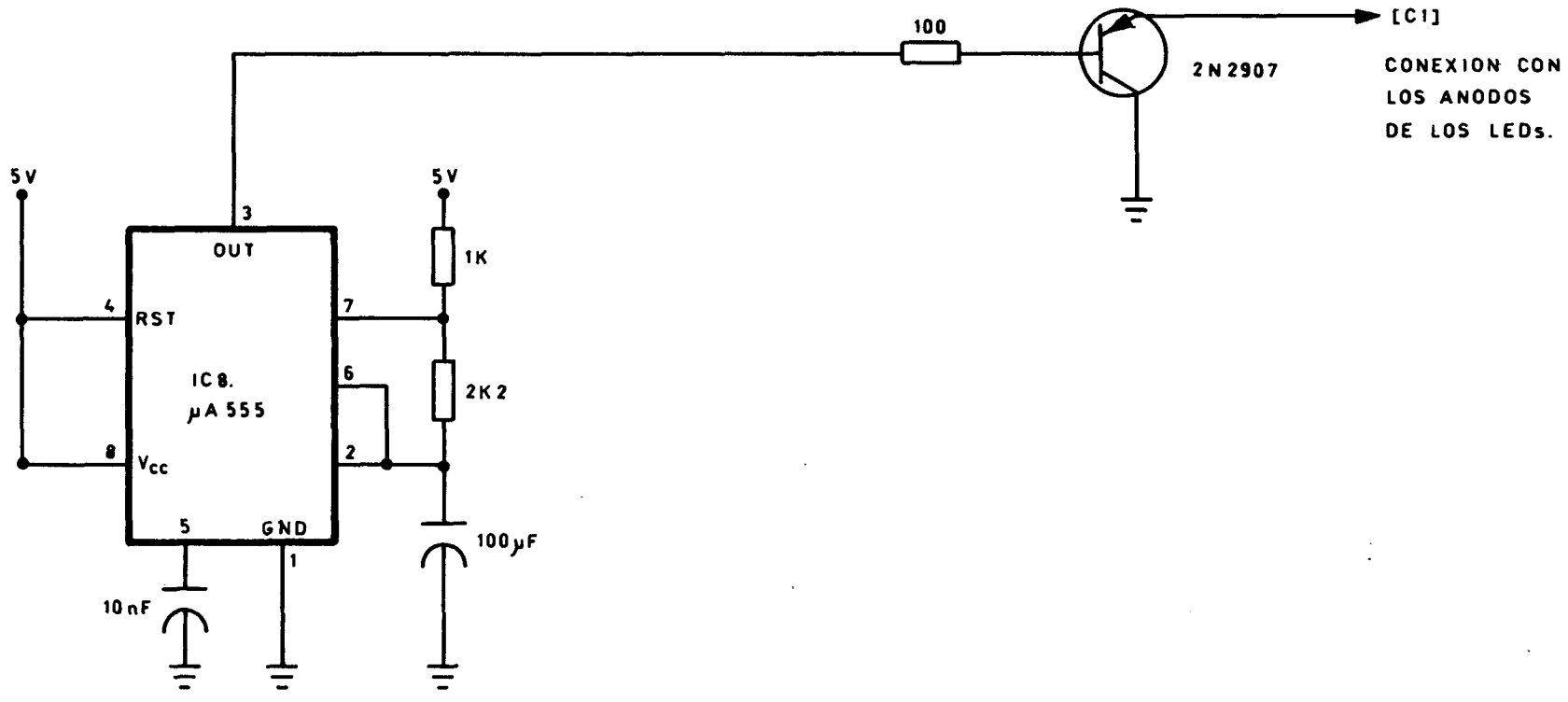
El puerto de salida 22H es utilizado para transferir la codificación de selección a los decodificadores. Ambos decodificadores recibirán el código de selección al mismo tiempo, pero mediante una puerta inversora se habilitará a uno de ambos para llevar a cabo la puesta a nivel lógico Ø de la salida seleccionada.

Las salidas de estos decodificadores activarán los diferentes dispositivos de respuesta con los que cuenta en sistema de seguridad.

III-3-2. CIRCUITOS DE ACTIVACION.

Son los encargados de informar al usuario de que se ha producido una anomalía en el sistema. El presente sistema de seguridad utiliza tres tipos de circuitos de activación: indicadores luminosos, activación de alarmas sonoras y activación de señales TTL que pondrán en funcionamiento el marcador-transmisor de alarmas (MTA/2), que tiene como misión la llamada telefónica automática al Servicio de Extinción de Incendios ó a la Comisaría de Policía. Los indicadores luminosos de zona de alarma estarán formados por diodos emisores de luz (LED)

- Figura 9 -

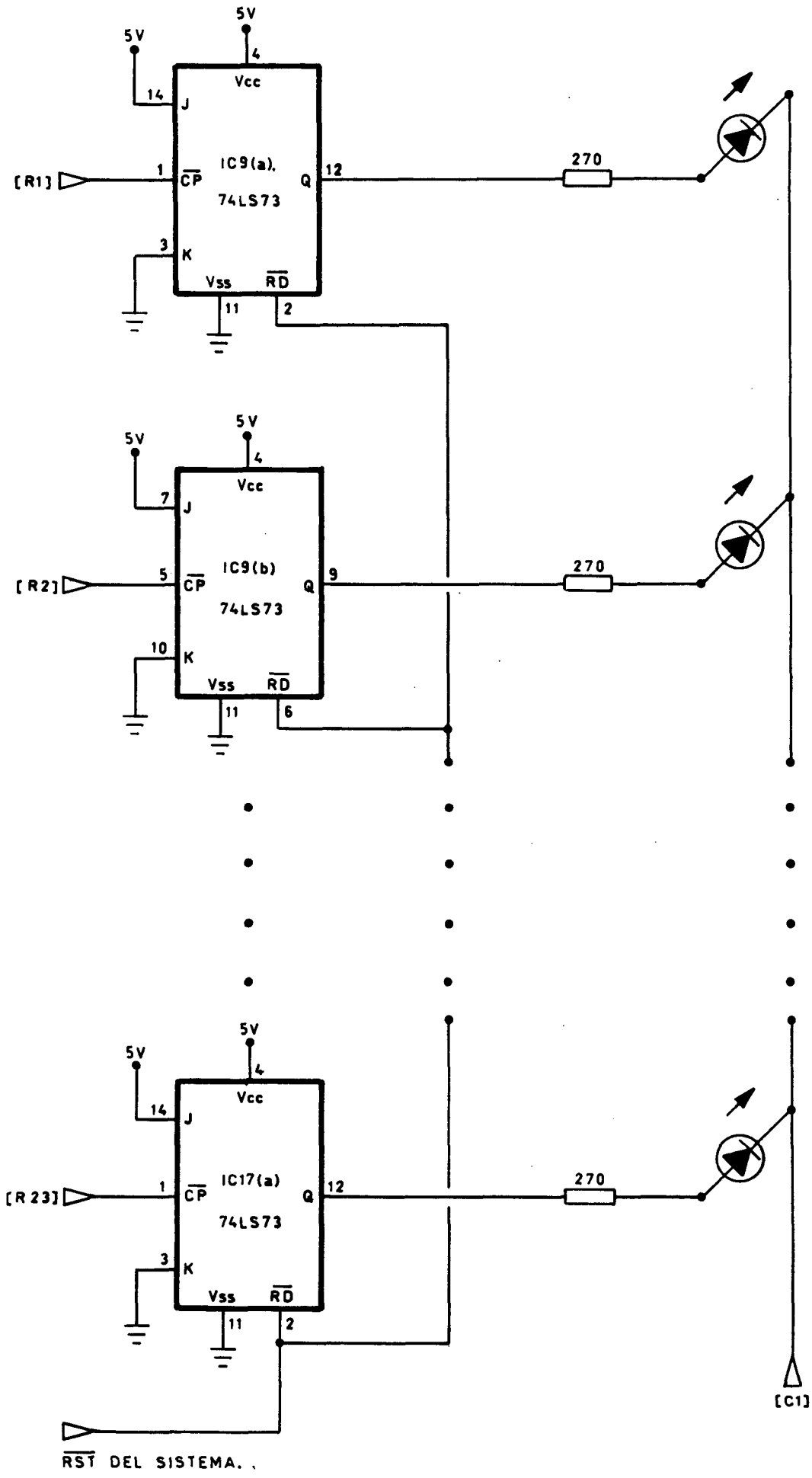


que deberán de producir un parpadeo al entrar en funcionamiento. También deberán de permanecer encendidos hasta que sea reinicializado el sistema.

Para conseguir el parpadeo de los LEDs se utiliza un temporizador 555 en una configuración como la que muestra la figura 9. El temporizador utiliza una red externa de resistencias y condensadores para ajustar la frecuencia de oscilación deseada, que en nuestro caso es aproximadamente de un ciclo por segundo. La salida del oscilador ataca a la base de un transistor 2N2907 que se utiliza como un 'switch' y que asegura un buen funcionamiento en el suministro de corriente al conjunto de los indicadores.

El enclavamiento de los LEDs se consigue mediante la utilización de flip-flops JK. Estos flip-flops vienen integrados en el circuito 74LS73. Los flips-flops Jk se disparan al recibir el pulso por flanco de bajada que proviene de uno de los dos decodificadores de salida con que cuenta el sistema. Una vez que el biestable ha sido disparado, su estado no podrá ser cambiado hasta que el sistema sea reinicializado. En la figura 10 se muestra el

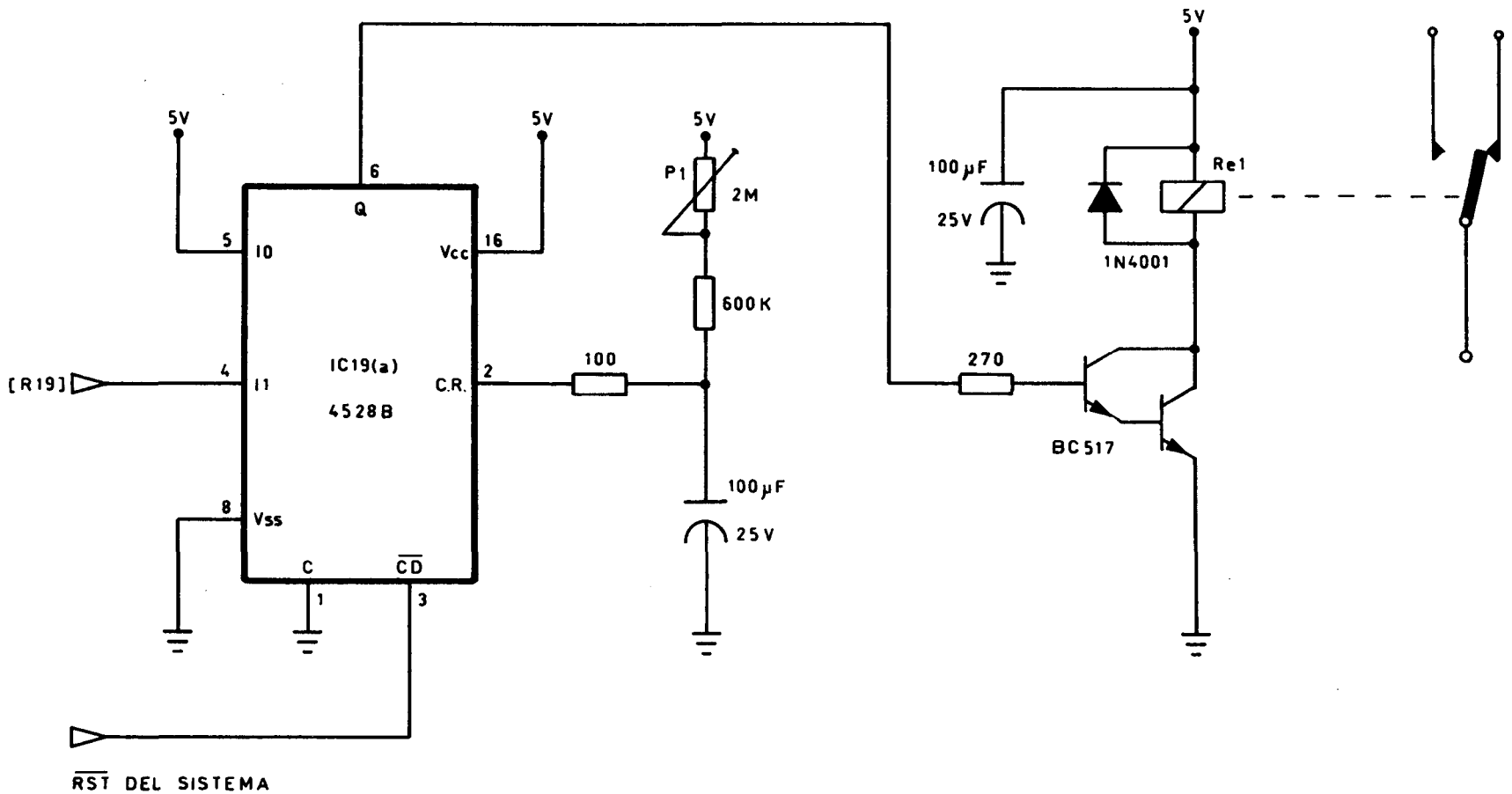
- Figura 10 -



conjunto de biestables JK con sus correspondientes LEDs. Las salidas de los flip-flops se conectan directamente a la parte positiva (cátodo) de los diodos emisores de luz, mientras que el ánodo de los mismos se conexiona con el emisor del transistor utilizado por el oscilador. De ésta forma, cuando se produce la activación del biestable JK, el LED comienza su parpadeo.

Las dos alarmas sonoras, una interior y otra exterior, de las que dispone el sistema de seguridad son activadas por la acción de un relé. En nuestro país, la legislación indica que la sirena de alarma no debe de permanecer más de cinco minutos conectada. Por ello se hace necesario la utilización de un temporizador que desconecte la sirena de alarma una vez ha transcurrido éste tiempo. El circuito de temporización y de activación del relé se encuentra en la figura 11. En la temporización se ha utilizado el circuito integrado C-MOS 4528B. Cuando el circuito temporizador recibe un pulso proviniente de los decodificadores de salida se produce su activación, dando lugar a un nivel lógico alto en su salida. Esto produce la activación del relé mediante la acción del transistor Darlington. La magnitud del

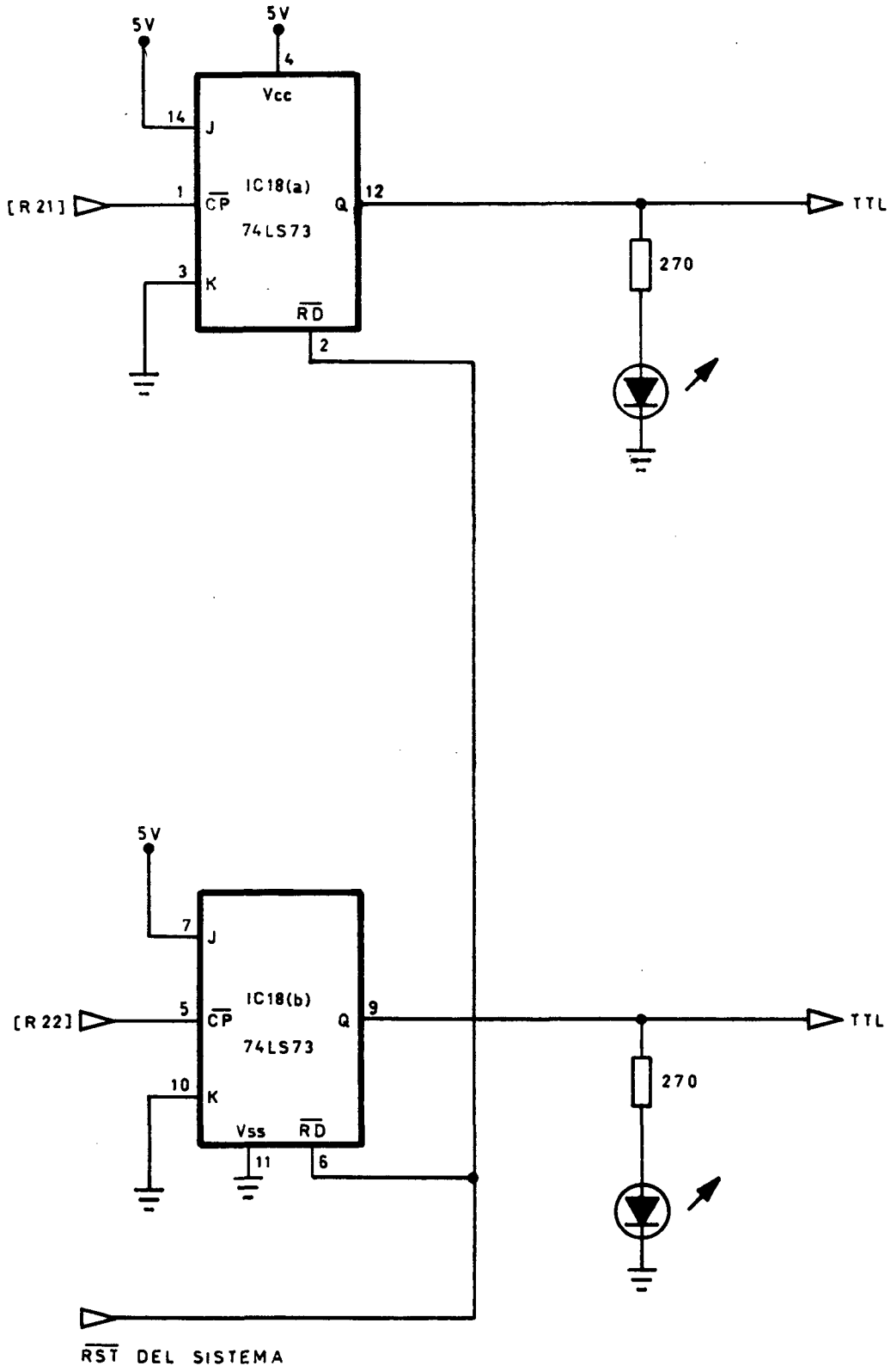
- Figura 11 -



tiempo en que el relé se mantiene activado puede preestablecerse entre 10 segundos y 5 minutos mediante la acción del potenciometro que se observa en la figura 11.

La activación de una señal TTL de puesta en funcionamiento del marcador-transmisor de alarma se lleva a cabo por medio de dos flip-flops JK. Estos activarán al mismo tiempo dos diodos emisores de luz que servirán de testigos de llamada telefónica automática. En la figura 12 se muestra el circuito electrico.

- Figura 12 -

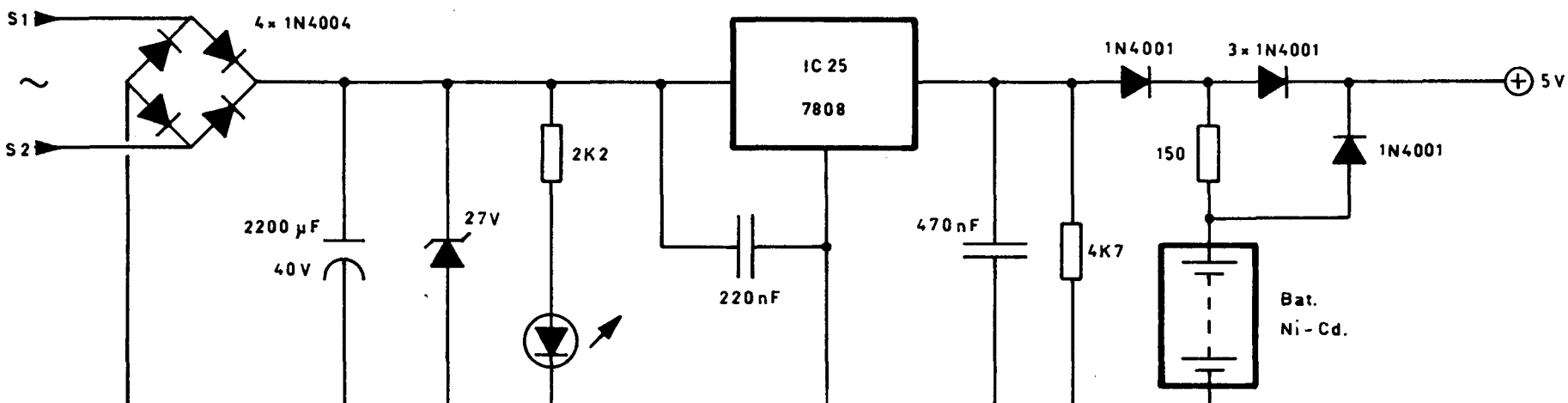


III-3-3. FUENTE DE ALIMENTACION.

La fuente de alimentación del sistema debe de asegurar el flujo de corriente hacia el mismo, incluso en el caso de corte en la tensión de la red. Esto se consigue mediante la utilización de baterias de Niquel-Cadmio. Estas se mantienen cargadas mediante un cargador automático conectado a la red, que recarga las baterias cuando su voltaje cae por debajo de un cierto límite.

El primario del transformador estará conectado a la red electrica de la que tomará una onda sinusoidal de 125 ó 220 voltios. La salida del secundario del transformador será una onda sinusoidal similar pero reducida a una tensión de unos 12 voltios. A continuación se aplica a un puente de onda completa cuya forma de onda no tendrá valores negativos. En el puente de onda completa se utilizan diodos de silicio. Estos presentan una caída de tensión en la unión de aproximadamente 1 voltio. La caída de tensión de dos voltios en el puente es una consideración importante a tener en cuenta.

El regulador de tensión requiere un cierto nivel de corriente continua mínimo para mantener una tensión de salida constante. Si la



- Figura 13 -

tensión aplicada cayera por debajo de éste punto, la estabilidad de salida quedará gravemente deteriorada. Por consiguiente, es necesario emplear un condensador de filtrado para suavizar las protuberancias en la onda sinusoidal rectificada. Cuando los diodos están conduciendo, el condensador almacena suficiente carga para mantener la tensión mínima requerida hasta el siguiente ciclo de carga.

El condensador de filtrado empleado es de un valor de 2200 uF. El diodo zener del circuito (ver figura 13) proporciona una protección adicional frente a transitorios que pudieran presentarse a la entrada de la red. A través de la resistencia de 2K2 se alimenta un diodo emisor de luz , que será utilizado como testigo de red.

El regulador de tensión IC25 (7808) reduce la tensión obtenida en los terminales del condensador de filtrado a una tensión estabilizada de unos 8 voltios, que se utilizan para la carga de las baterias y la alimentación del sistema.

La batería de seis elementos de Níquel Cadmio se recarga con la corriente suministrada por el regulador de tensión IC25, a través del

diodo 1N4ØØ1 en serie con una resistencia de 15Ø Ohm. El diodo evita el retorno de corriente en caso de fallo de la red y la resistencia limita el paso de corriente hacia las baterías.

La tensión de alimentación del sistema se obtiene combinando la tensión de la batería con la de la salida del regulador. Los tres diodos 1N4ØØ1 conectados en serie se utilizan para reducir la tensión del regulador a 5 voltios.

Las baterías de Níquel-Cadmio utilizadas han de ser del tipo C, que poseen una tensión nominal de 1,25 voltios y una capacidad de 2 amperios hora.

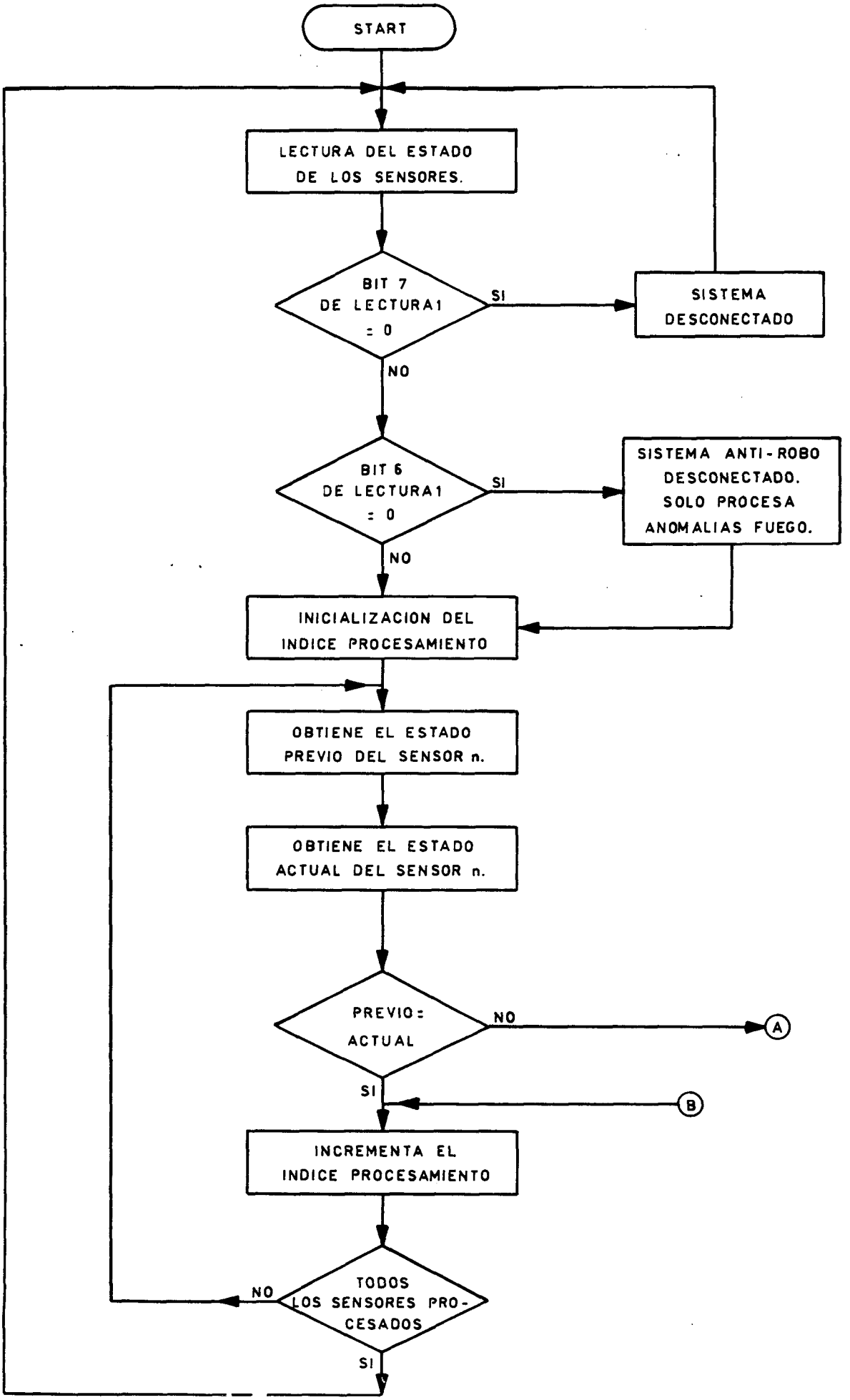
III-4. DESCRIPCION DEL PROGRAMA.

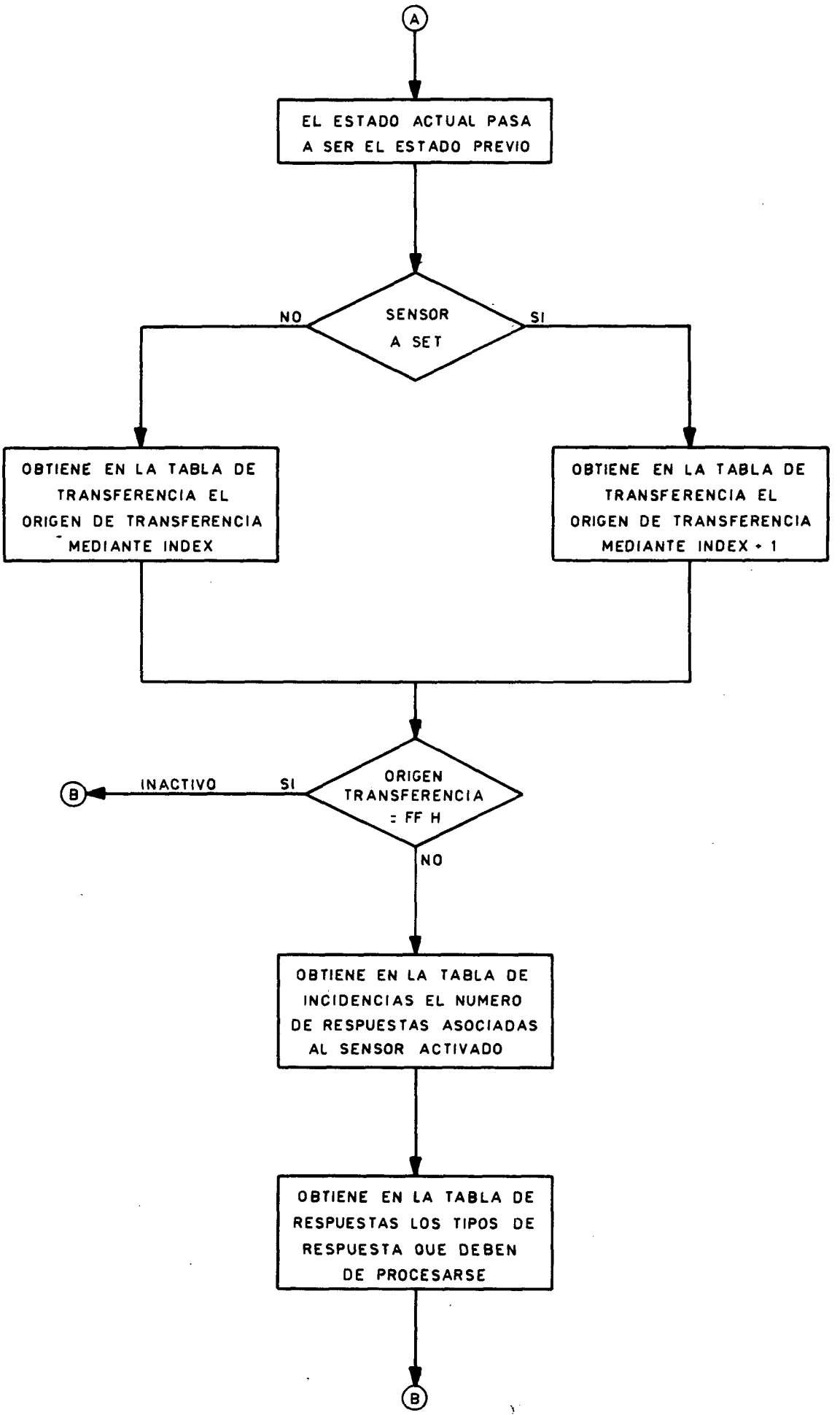
El lenguaje de programación utilizado en el desarrollo del sistema de seguridad es el PL/M-8Ø. Tal y como se ha descrito en la PARTE I del presente proyecto, se trata de un lenguaje de alto nivel destinado a su utilización con los microprocesadores 8Ø8Ø y 8Ø85.

Las buenas cualidades de éste lenguaje para la estructuración modular del programa así como la posibilidad de utilizar subrutinas y matrices han contribuido poderosamente en la optimización del sistema, así como a una mayor facilidad en la depuración del mismo.

En la figura 14 se puede observar el diagrama de flujo del software del sistema de seguridad. En cuanto el sistema es reinicializado procede a realizar la lectura de los estados en que se encuentran los diferentes dispositivos de detección de que consta el sistema de seguridad. Para realizar la lectura el programa utiliza el denominado "MODULO DE PUSQUEDA". Este genera un código digital que va de Ø a 7 de forma secuencial y que mediante el puerto de salida 22H es aplicado a las entradas de selección de los multiplexores de entrada. Cada vez que es extraido un código se realiza una lectura a

- Figura 14 -





través del puerto 21H del nivel digital que presentan las salidas de los multiplexores, de forma que cuando se ha completado la secuencia Ø - 7 se ha realizado la lectura de todas las entradas correspondientes a los tres multiplexores, es decir, tres bytes. Cada uno de estos bytes se conocerán en el programa mediante los identificadores LECTURA1, LECTURA2 y LECTURA3. El "MODULO DE BUSQUEDA" tiene también encomendada la misión de poner a Ø el índice de procesamiento. Se trata éste de un índice que se encarga de asignar valores consecutivos a cada uno de los bits que componen cada byte leído, pudiéndose determinar con facilidad el bit, que corresponde al estado de un sensor, que se está procesando. Una vez ejecutado el "MODULO DE BUSQUEDA" el programa pasa a realizar el siguient módulo, en el cual se realiza un análisis de los bits 6 y 7 del byte LECTURA1, para comprobar que partes del sistema están activadas y cuales no lo están. Si el bit 7 de LECTURA1 es Ø, el sistema estará desconectado; en caso contrario, examina el estatus del bit 6. Si éste bit es Ø, el sistema de seguridad anti robo estará desconectado y solo procesara las anomalías que se pueden producir por incendio

o por emanación de humos. Si el bit 6 es 1, se procesarán todas las anomalías que se produzcan en los dispositivos de detección. Llegado a éste punto, el programa realiza una comparación entre el "estado actual" y el "estado previo".

El "estado actual" corresponde al estado en que se encuentran los sensores en la última lectura. El "estado proevio" corresponde al estado en que se encontraban los sensores en la lectura anterior a la que se ha llevado a cabo en último lugar. Inicialmente, el estado previo contendrá la configuración en que deben de encontrarse los sensores en ausencia de alarma.

La comparación entre el estado actual y el estado previo se realiza en blo ues de 8 bits, es decir, se comparan dos bytes; uno, el byte de estado previo, y el otro, el byte de estado actual. Si una vez realizada la comparación se comprueba que ambos bytes son iguales, no ha habido cambio en el estado de los sensores, se incrementa el índice de procesamiento en 8 unidades, comprobándose seguidamente si todos los bytes han sido procesados. De haberse procesado, es decir, el índice de procesamiento ha llegado al valor 24, el programa vuelve al módulo

de búsqueda para realizar nuevas lecturas. En el caso de que no hayan sido procesados todos los bytes, se realizará la comparación de los estados previo y actual correspondientes al siguiente byte.

Si una vez realizada la comparación entre los estados actual y previo se comprueba que son diferentes, es decir, que se ha producido un cambio en el estado de algún sensor, se pasa a comparar bit a bit cual ha sido el que ha cambiado, hecho lo cual, el estado actual pasa a ser el estado previo. Una vez realizado ésto se comprueba como ha sido el cambio experimentado por el sensor, es decir, si éste ha pasado de SET a RESET (Ø a 1) ó por el contrario lo ha hecho de RESET a SET (1 a Ø). Determinar este cambio de estado es importante, ya que de esta forma pueden coexistir en el sistema de seguridad sensores que se activen al pasar de RESET a SET con otros sensores cuya activación se produce al pasar de SET a RESET.

Hay que recordar que mediante el índice de procesamiento el programa reconoce el sensor que está siendo procesado en cada momento. Una vez determinado el tipo de cambio que se ha producido en el sensor (Ø a 1 ó 1 a Ø) se pasa a

- Figura 15 -

TABLA DE TRANSFERENCIAS

ORIGEN DE TRANSFERENCIA

INDEX	HEX.	DEC.
0	FF H	
1	0	0D
2	FF	
3	04	4
4	FF	
5	08	8
6	FF	
7	0C	12
8	FF	
9	10	16
10	FF	
11	14	20
12	FF	
13	FF	
14	FF	
15	FF	
16	FF	
17	18	24
18	FF	
19	1C	28
20	FF	
21	20	32
22	FF	
23	24	36
24	FF H	
25	28	40 D
26	FF	
27	2C	44
28	FF	
29	30	48
30	FF	
31	34	52
32	FF	
33	38	56
34	FF	
35	3C	60
36	FF	
37	40	64
38	FF	
39	44	68
40	FF	
41	48	72
42	FF	
43	4C	76
44	FF	
45	50	80
46	FF	
47	54	84

comprobar si el sensor al realizar dicho cambio de estado se ha acticado o por el contrario ha quedado desactivado despues de producir una señal de alarma. Para realizar este cometido se utiliza la TABLA DE TRANSFERENCIAS (Figura 15). La TABLA DE TRANSFERENCIAS está formada por una secuencia de valores almacenados consecutivamente en memoria. Cada sensor tiene asignado 2 bytes consecutivos en esta TABLA DE TRANSFERENCIA. En el caso de que el sensor haya pasado de SET a RESET, el programa tomará el primer byte de la TABLA DE TRANSFERENCIAS que corresponde al sensor que está procesando. En el caso de que el sensor pase de RESET a SET, se tomará el valor correspondiente al segundo byte. Si por ejemplo un sensor pasa al estado de alarma cuando se produce el cambio RESET a SET, debe de colocarse en el primer byte que tiene asignado dicho sensor en la TABLA DE TRANSFERENCIA el valor hexadecimal FFH, para indicar que al pasar de SET a RESET no debe de producirse ninguna alarma, ya que se encuentra inactivo para éste tipo de cambio de estado (recuerdese que el primer byte de la TABLA DE TRANSFERENCIA corresponde al cambio SET a RESET). En el segundo byte que le corresponde al sensor en la

- Figura 16 -

TABLA DE INCIDENCIAS:

ORIGEN DE TRANSFERENCIA.

ORIGEN DE INCIDENCIAS.

0	RESP:3
	01
	02
	03
04	RESP:3
	04
	05
	06
08	RESP:3
	07
	08
	09
12	RESP:3
	0A
	0B
	0C
16	RESP:3
	0D
	0E
	0F
20	RESP:3
	10
	11
	12
24	RESP:3
	13
	14
	15
28	RESP:3
	16
	17
	18
32	RESP:3
	19
	1A
	1B
36	RESP:3
	1C
	1D
	1E
40	RESP:3
	1F
	20
	21

44	RESP:3
	22
	23
	24
48	RESP:3
	25
	26
	27
52	RESP:3
	28
	29
	2A
56	RESP:3
	2B
	2C
	2D
60	RESP:3
	2E
	2F
	30
64	RESP:3
	31
	32
	33
68	RESP:3
	34
	35
	36
72	RESP:3
	37
	38
	39
76	RESP:3
	3A
	3B
	3C
80	RESP:3
	3D
	3E
	3F
84	RESP:3
	40
	41
	42

tabla debe de colocarse un valor que corresponde a un puntero de direccionamiento, mediante el que se dirigirá a un lugar concreto de otra tabla denominada TABLA DE INCIDENCIAS y que estudiaremos más adelante.

Por lo tanto, cuando se produce un cambio de estado en alguno de los sensores, el programa comprueba se ésto cambio ha sido de RESET a SET, ó por el contrario de SET a RESET; realizado lo cual se traslada a la TABLA DE TRANSFERENCIA para comprobar mediante los valores que tiene asignado el sensor en dicha tabla, si el sensor ha pasado a un estado activo ó por el contrario loha hecho a un estado inactivo. En el caso de que haya pasado a un estado inactivo (INH), se realiza el incremento del índice de procesamiento para seguir posteriormente testeando otros sensores. En el caso de que el sensor haya pasado a un estado de activadas, tomará el valor que se ha asignado en la TABLA DE TRANSFERENCIA y mediante el mismo accederá a una posición concreta de la TABLA DE INCIDENCIAS (Figura 16). Esta tabla estará estructurada de la siguiente forma; en el primer byte se encuentra el número de respuestas asociadas al sensor que ha sido activado. Por ejemplo, si

TABLA DE RESPUESTAS:

PUNTERO	TIPO DE RESPUESTA.
0	TIPO 1
1	TIPO 2
2	TIPO 3
3	TIPO 1
4	TIPO 2
5	TIPO 3
6	TIPO 1
7	TIPO 2
8	TIPO 3
9	TIPO 1
10	TIPO 2
11	TIPO 3
12	TIPO 1
13	TIPO 2
14	TIPO 3
15	TIPO 1
16	TIPO 2
17	TIPO 3
18	TIPO 1
19	TIPO 2
20	TIPO 3
21	TIPO 1
22	TIPO 2
23	TIPO 3
24	TIPO 1
25	TIPO 2
26	TIPO 3
27	TIPO 1
28	TIPO 2
29	TIPO 3
30	TIPO 0
31	TIPO 2
32	TIPO 4
33	TIPO 0
34	TIPO 2
35	TIPO 4
36	TIPO 0
37	TIPO 2
38	TIPO 4
39	TIPO 0
40	TIPO 2
41	TIPO 4
42	TIPO 0
43	TIPO 2
44	TIPO 4
45	TIPO 0
46	TIPO 2
47	TIPO 4
48	TIPO 0
49	TIPO 2
50	TIPO 4

TIPO 0 : SIRENA POR INCENDIO.
TIPO 1 : SIRENA POR INTRUSOS.
TIPO 2 : INDICADOR EN PANEL.
TIPO 3 : LLAMADA POLICIA.
TIPO 4 : LLAMADA BOMBEROS.

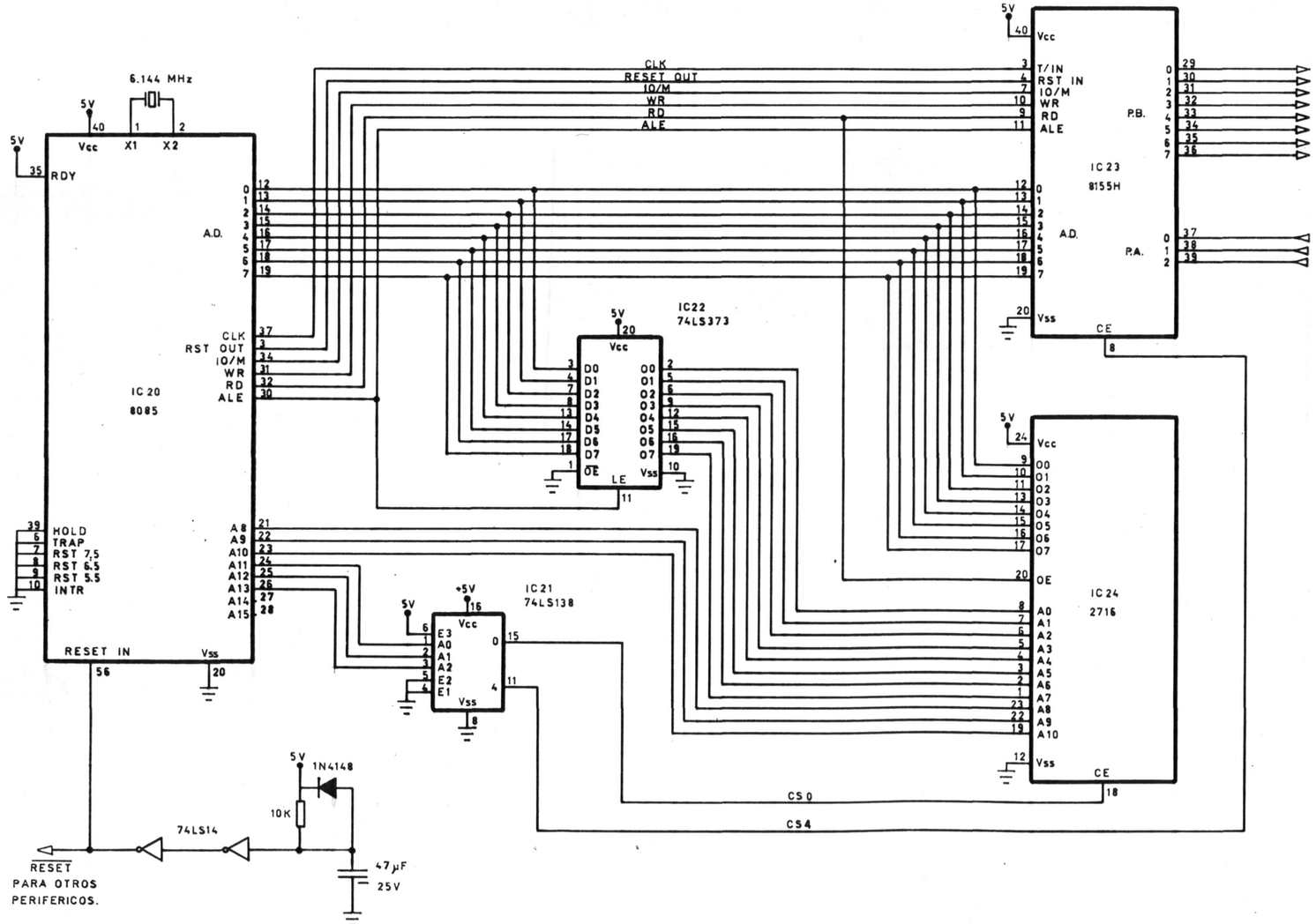
uno de los sensores del sistema debe de producir tres respuestas al ser activado, en el primer byte que se le ha asignado a dicho sensor en la TABLA DE INCIDENCIAS se pondrá el valor Ø3. A continuación de éste valor deben de ir otros tres valores consecutivos que corresponden a un puntero de direccionamiento en una tercera tabla denominada TABLA DE RESPUESTAS (Figura 17). En ésta tabla se encontrarán los tipos de respuestas asignadas a cada sensor. En el presente sistema de seguridad se han definido cinco tipos diferentes de respuestas, correspondiendole a cada uno de ellos un valor numérico de Ø a 4, mediante el cual el programa reconoce el que debe de llevarse a cabo. Estos valores numéricos son los que integran la denominada TABLA DE RESPUESTAS.

Una vez que se han llevado a cabo todas las respuestas asociadas al sensor activado, el programa incrementa el índice de procesamiento para seguidamente continuar testeando los demás estados obtenidos en la lectura.

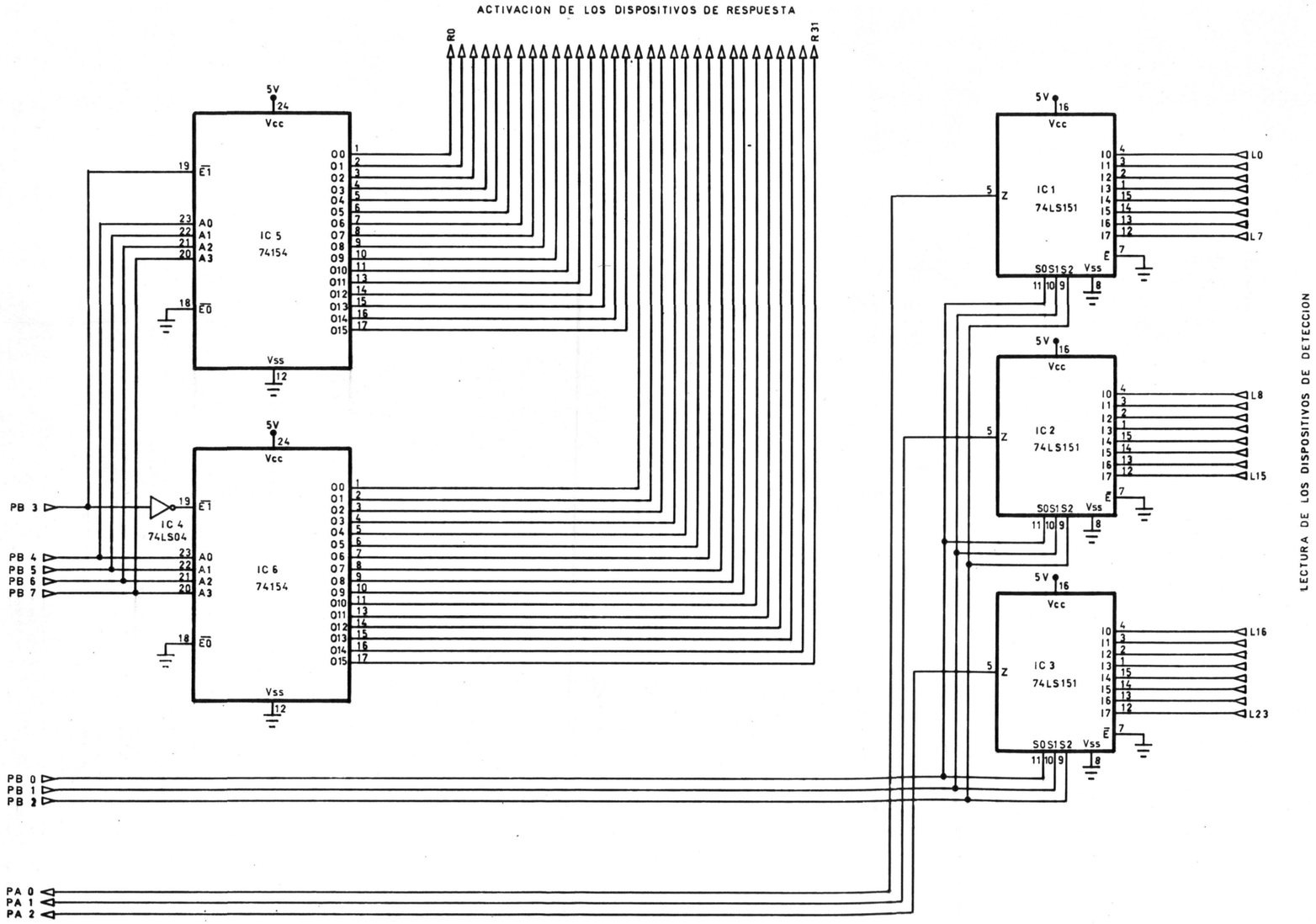
APENDICE.

APENDICE A.

ESQUEMA ELECTRICO.



UNIVERSIDAD POLITECNICA DE LAS PALMAS INGENIERIA TECNICA DE TELECOMUNICACION	
APENDICE A HOJA 1	TITULO DEL PROYECTO: SISTEMA DE SEGURIDAD CONTROLADO POR MICROPROCESADOR
FECHA: JUNIO - 86	PLANO: UNIDAD DE CONTROL
S/E	



UNIVERSIDAD POLITECNICA DE LAS PALMAS INGENIERIA TECNICA DE TELECOMUNICACION	
APENDICE A HOJA 2	TITULO DEL PROYECTO: SISTEMA DE SEGURIDAD CONTROLADO POR MICROPROCESADOR
FECHA: JUNIO - 86	PLANO: UNIDAD DE CONTROL
S/E	

APENDICE B.

LISTADO DEL PROGRAMA.

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE SAFETY
OBJECT MODULE PLACED IN SAFETY.OBJ
COMPILER INVOKED BY:  PLM80 SAFETY.V2 WORKFILES (:F0:,:F0:) PAGELENGTH (39) PAGEWIDTH (100)
```

```
   1            SAFETY: DO;
                /*##########################################################################*/
                /*##                                                                      ##*/
                /*##              SISTEMA DE SEGURIDAD PARA UNA VIVIENDA                  ##*/
                /*##                  CONTROLADO POR MICROPROCESADOR.                     ##*/
                /*##                   DESARROLLADO EN ENERO DE 1986.                     ##*/
                /*##                   (C) JOSE MANUEL ALDANA GARCIA.                     ##*/
                /*##                                                                      ##*/
                /*##########################################################################*/
                /*##########################################################################*/
                /*DECLARACION DE ETIQUETAS*/
   2    1       DECLARE SCAN LABEL;
   3    1       DECLARE DIGI1 LABEL;
   4    1       DECLARE DIG2 LABEL;
                /*##########################################################################*/
                /*INICIALIZACION DE LAS VARIABLES*/
   5    1       DECLARE (LECTURA1,LECTURA2,LECTURA3) BYTE INITIAL (0,0,0);
   6    1       DECLARE (LECTURA$PREVIA1,LECTURA$PREVIA2,LECTURA$PREVIA3) BYTE INITIAL
                     (0,0,0);
   7    1       DECLARE (ESTADO$ACTUAL,ESTADO$PREVIO,NUEVO$PREVIO) BYTE INITIAL
                       (0,0,0);
   8    1       DECLARE (TEMP1,TEMP2,TEMP3,TEMP4) BYTE INITIAL (0,0,0,0);
   9    1       DECLARE (MASCARA$SISTEMA,MASCARA$INTRUSO,ANOMALIAS$FUEGO) BYTE
                     INITIAL (0,0,0);
  10    1       DECLARE (INDICE$PROCE,COMP$MASCARA) BYTE INITIAL (0,0);
  11    1       DECLARE (PUNTERO,MULT,MASK5,SELET) BYTE;
  12    1       DECLARE (ENTRADA1,ENTRADA2,ENTRADA3) BYTE;
  13    1       DECLARE (ORIGEN$RESPUESTA,ORIGEN$INCIDENCIAS,ORIGEN$TRANSFERENCIA) BYTE;
                /*##########################################################################*/
                /*DECLARACION DE PROCEDIMIENTOS*/
```

```
                /*=============================================================================*/
14    1         PROCESO$DE$MASCARA: PROCEDURE;
15    2              IF TEMP2=0 THEN TEMP2= 0000$0001B;
                     ELSE
17    2                   DO;
18    3                      TEMP2= ROL(TEMP2,1);
19    3                   END;
20    2              TEMP3= ESTADO$ACTUAL AND TEMP2;
21    2              TEMP4= ESTADO$PREVIO AND TEMP2;
22    2         END PROCESO$DE$MASCARA;
                /*=============================================================================*/
23    1         ACTUALIZACION: PROCEDURE;
24    2              IF INDICE$PROCE< 8D THEN
25    2                   DO;
26    3                      LECTURA$PREVIA1= NUEVO$PREVIO;
27    3                      INDICE$PROCE= INDICE$PROCE + 1;
28    3                   END;
                     ELSE
29    2                   DO;
30    3                      IF INDICE$PROCE< 16D THEN
31    3                      DO;
32    4                         LECTURA$PREVIA2= NUEVO$PREVIO;
33    4                         INDICE$PROCE= INDICE$PROCE + 1;
34    4                      END;
                              ELSE
35    3                          DO;
36    4                         LECTURA$PREVIA3= NUEVO$PREVIO;
37    4                         INDICE$PROCE= INDICE$PROCE + 1;
38    4                      END;
39    3                   END;
40    2         END ACTUALIZACION;
                /*=============================================================================*/
41    1         COMPARACION: PROCEDURE;
42    2              IF ESTADO$ACTUAL=ESTADO$PREVIO THEN
43    2                   DO;
44    3                      IF INDICE$PROCE < 8D THEN INDICE$PROCE = 8D;
```

```
                        ELSE
46    3                      DO;
47    4                     IF INDICE$PROCE < 16D THEN INDICE$PROCE = 16D;
                            ELSE
49    4                     DO;
50    5                        INDICE$PROCE = 24D;
51    5                     END;
52    4                      END;
53    3                  TEMP2 = 0;
54    3                  TEMP1 = 0;
55    3              END;
                   ELSE
56    2                  DO;
57    3                     CALL PROCESO$DE$MASCARA;
58    3                     DO WHILE TEMP3= TEMP4;
59    4                       INDICE$PROCE= INDICE$PROCE + 1;
60    4                       CALL PROCESO$DE$MASCARA;
61    4                     END;
62    3                     IF TEMP3= 0 THEN
63    3                     DO;
64    4                        TEMP1= 2;
65    4                        COMP$MASCARA= NOT (TEMP2);
66    4                        NUEVO$PREVIO= ESTADO$PREVIO AND COMP$MASCARA;
67    4                        CALL ACTUALIZACION;
68    4                     END;
                            ELSE
69    3                     DO;
70    4                        TEMP1= 1;
71    4                        NUEVO$PREVIO=ESTADO$PREVIO OR TEMP2;
72    4                        CALL ACTUALIZACION;
73    4                     END;
74    3                      END;
75    2          END COMPARACION;
                 /*=============================================================================*/
76    1          CONTINUACION: PROCEDURE;
77    2               IF INDICE$PROCE= 24D THEN GOTO SCAN;
```

```
 79   2            ELSE GOTO DIGI1;
 80   2        END CONTINUACION;
              /*=============================================================================*/
 81   1        SIRENA$FUEGO: PROCEDURE;
 82   2            OUTPUT (22H) = SALIDA$FUEGO;
 83   2        END SIRENA$FUEGO;
              /*=============================================================================*/
 84   1        SIRENA$INTRUSOS: PROCEDURE;
 85   2            OUTPUT (22H) = SALIDA$INTRUSO;
 86   2        END SIRENA$INTRUSOS;
              /*=============================================================================*/
 87   1        BOMBEROS: PROCEDURE;
 88   2            OUTPUT (22H) = SALIDA$BOMBEROS;
 89   2        END BOMBEROS;
              /*=============================================================================*/
 90   1        POLICIA: PROCEDURE;
 91   2            OUTPUT (22H) = SALIDA$POLICIA;
 92   2        END POLICIA;
              /*=============================================================================*/
 93   1        PANEL: PROCEDURE;
 94   2            DECLARE CURSOR BYTE;
 95   2            DECLARE DATO$PANEL BYTE;
 96   2            CURSOR = (INDICE$PROCE - 1);
 97   2            DATO$PANEL = TABLA$PANEL (CURSOR);
 98   2            OUTPUT (22H) = DATO$PANEL;
 99   2        END PANEL;
              /*=============================================================================*/
100   1        MODULO$RESPUESTA: PROCEDURE;
101   2            DECLARE TIPO$RESPUESTA BYTE;
102   2            PUNTERO = (PUNTERO - 1);
103   2            TIPO$RESPUESTA = TABLA$RESPUESTA (PUNTERO);
104   2                DO CASE TIPO$RESPUESTA;
105   3                CALL SIRENA$FUEGO;
106   3                CALL SIRENA$INTRUSOS;
107   3                CALL PANEL;
108   3                CALL POLICIA;
```

```
109   3                 CALL BOMBEROS;
110   3             END;
111   2         END MODULO$RESPUESTA;
                /*=============================================================================*/
112   1         PROCESADOR$INCIDENCIAS: PROCEDURE;
113   2             DECLARE INC BYTE;
114   2             DECLARE I BYTE;
115   2             INC = 1;
116   2             DO I = 1 TO ORIGEN$INCIDENCIAS;
117   3                 PUNTERO = TABLA$INCIDENCIAS (ORIGEN$TRANSFERENCIA +
                              INC);
118   3                 CALL MODULO$RESPUESTA;
119   3                 INC = INC + 1;
120   3             END;
121   2         END PROCESADOR$INCIDENCIAS;
                /*=============================================================================*/
122   1         PROCESAMIENTO: PROCEDURE;
123   2             DECLARE INDEX BYTE;
124   2             INDEX= (INDICE$PROCE - 1) * 2;
125   2             IF TEMP1 = 1 THEN ORIGEN$TRANSFERENCIA = TABLA$TRANSFERENCIA
                        (INDEX + 1);
127   2             ELSE ORIGEN$TRANSFERENCIA = TABLA$TRANSFERENCIA (INDEX);
128   2             IF ORIGEN$TRANSFERENCIA = 0FFH THEN CALL CONTINUACION;
                    ELSE
130   2                 DO;
131   3                     ORIGEN$INCIDENCIAS = TABLA$INCIDENCIAS (ORIGEN$TRANSFERENCIA);
132   3                     CALL PROCESADOR$INCIDENCIAS;
133   3                 END;
134   2             CALL CONTINUACION;
135   2         END PROCESAMIENTO;
                /*#############################################################################*/
                /*DECLARACION DE CONSTANTES*/
136   1         DECLARE MASK1 BYTE DATA (80H);
137   1         DECLARE MASK2 BYTE DATA (40H);
138   1         DECLARE MASK3 BYTE DATA (0FBH);
139   1         DECLARE SALIDA$FUEGO BYTE DATA (0011$1000B);
```

```
140   1       DECLARE SALIDA$INTRUSO BYTE DATA (0100$1000B);
141   1       DECLARE SALIDA$BOMBEROS BYTE DATA (0101$1000B);
142   1       DECLARE SALIDA$POLICIA BYTE DATA (0110$1000B);
143   1       DECLARE TABLA$TRANSFERENCIA (*) BYTE DATA (0FFH,0,0FFH,04H,0FFH,08H,
                  0FFH,0CH,0FFH,10H,0FFH,14H,0FFH,0FFH,0FFH,0FFH,0FFH,18H,0FFH,
                  1CH,0FFH,20H,0FFH,24H,0FFH,28H,0FFH,2CH,0FFH,30H,0FFH,34H,
                  0FFH,38H,0FFH,3CH,0FFH,40H,0FFH,44H,0FFH,48H,0FFH,4CH,0FFH,
                  50H,0FFH,54H);
144   1       DECLARE TABLA$INCIDENCIAS (*) BYTE DATA (03H,01H,02H,03H,03H,04H,05H,
                  06H,03H,07H,08H,09H,03H,0AH,0BH,0CH,03H,0DH,0EH,0FH,03H,10H,
                  11H,12H,03H,13H,14H,15H,03H,16H,17H,18H,03H,19H,1AH,1BH,03H,
                  1CH,1DH,1EH,03H,1FH,20H,21H,03H,22H,23H,24H,03H,25H,26H,27H,
                  03H,28H,29H,2AH,03H,2BH,2CH,2DH,03H,2EH,2FH,30H,03H,31H,32H,
                  33H,03H,34H,35H,36H,03H,37H,38H,39H,03H,3AH,3BH,3CH,03H,3DH,
                  3EH,3FH,03H,40H,41H,42H);
145   1       DECLARE TABLA$RESPUESTA (*) BYTE DATA (01H,02H,03H,01H,02H,03H,01H,
                  02H,03H,01H,02H,03H,01H,02H,03H,01H,02H,03H,01H,02H,03H,01H,
                  02H,03H,01H,02H,03H,01H,02H,03H,0,02H,04H,0,02H,04H,0,02H,
                  04H,0,02H,04H,0,02H,04H,0,02H,04H,0,02H,04H);
146   1       DECLARE TABLA$PANEL (*) BYTE DATA (10H,20H,30H,40H,50H,60H,70H,80H,
                  90H,0A0H,0B0H,0C0H,0D0H,0E0H,0F0H,08H,18H,28H,78H);
                /*##############################################################################*/
                /*SECUENCIAS EJECUTABLES DEL PROGRAMA*/
                /*########## MODULO DE BUSQUEDA ##########*/
147   1       SCAN:   LECTURA1 = 0;
148   1           LECTURA2 = 0;
149   1           LECTURA3 = 0;
150   1           OUTPUT (20H) = 02H;
151   1               MASK5 = 0000$0001B;
152   1           INDICE$PROCE = 0;
153   1           ENTRADA1 = 0;
154   1           ENTRADA2 = 0;
155   1           ENTRADA3 = 0;
156   1           DO SELET = 0 TO 7;
157   2               OUTPUT (22H) = SELET;
158   2               MULT = INPUT (21H);
```

```
159   2             ENTRADA1 = MULT AND 0000$0001B;
160   2             ENTRADA2 = MULT AND 0000$0010B;
161   2             ENTRADA3 = MULT AND 0000$0100B;
162   2            IF ENTRADA1 <> 0 THEN LECTURA1 = LECTURA1 OR MASK5;
164   2            IF ENTRADA2 <> 0 THEN LECTURA2 = LECTURA2 OR MASK5;
166   2            IF ENTRADA3 <> 0 THEN LECTURA3 = LECTURA3 OR MASK5;
168   2             MASK5 = ROL(MASK5,1);
169   2         END;
170   1     LECTURA3 = LECTURA3 AND 0000$0111B;
          /*### ACTIVACION Y DESACTIVACION DEL SISTEMA ###*/
171   1         MASCARA$SISTEMA= LECTURA1 AND MASK1;
172   1         IF MASCARA$SISTEMA= 0 THEN GOTO SCAN;
          ELSE
174   1             DO;
175   2                 MASCARA$INTRUSO= LECTURA1 AND MASK2;
176   2                 IF MASCARA$INTRUSO= 40H THEN GOTO DIGI1;
                      ELSE
178   2                 DO;
179   3                     INDICE$PROCE= 8D;
180   3                     ANOMALIAS$FUEGO= LECTURA2 AND MASK3;
181   3                     LECTURA2= ANOMALIAS$FUEGO;
182   3                 END;
183   2             END;
          /*### CUANTIFICACION DEL INDICE DE PROCESAMIEMTO ###*/
184   1     DIGI1:  IF INDICE$PROCE< 8D THEN
185   1             DO;
186   2                 ESTADO$ACTUAL= LECTURA1;
187   2                 ESTADO$PREVIO= LECTURA$PREVIA1;
188   2                 GOTO DIG2;
189   2             END;
          ELSE
190   1             DO;
191   2                 IF INDICE$PROCE< 16D THEN
192   2                 DO;
193   3                     ESTADO$ACTUAL= LECTURA2;
194   3                     ESTADO$PREVIO= LECTURA$PREVIA2;
```

```
195   3                 GOTO DIG2;
196   3              END;
                     ELSE
197   2              DO;
198   3                 ESTADO$ACTUAL= LECTURA3;
199   3                 ESTADO$PREVIO= LECTURA_PRL(A3;
200   3                 GOTO DIG2;
201   3              END;
202   2           END;
203   1   DIG2:   CALL COMPARACION;
204   1        IF TEMP1= 0 THEN CALL CONTINUACION;
206   1        ELSE CALL PROCESAMIENTO;
207   1   END SAFETY;
```

MODULE INFORMATION:

```
    CODE AREA SIZE      = 03E8H    1000D
    VARIABLE AREA SIZE  = 0022H      34D
    MAXIMUM STACK SIZE  = 0008H       8D
    259 LINES READ
    0 PROGRAM ERROR(S)
```

END OF PL/M-80 COMPILATION