ESCUELA POLITECNICA NACIONAL

FACULTAD DE INGENIERIA ELECTRICA

"CODIFICADORES Y DECODIFICADORES PARA CODIGOS DE BLOQUE LINEALES. TECNICAS DE DISEÑO.-

TESIS PREVIA A LA OBTENCION DEL TITULO DE INGENIERO EN LA ESPECIALIZACION DE ELECTRONICA Y TELECOMUNICACIONES.-

CARLOS ROBERTO EGAS ACOSTA

- Director de Tesis:

Sr. Ing. Ricardo Dávila.

Quito, Agosto de 1987

Certifico que esta Tesis ha sido elaborada en su totalidad por el Señor Carlos R. Egas Acosta.

Ing. Ricardo Dávila F.
DIRECTOR DE TESIS.-

D E D I C A T O R I A:

A mis Padres:

Por el apoyo y respaldo total que siempre me brindaron, hoy, con mi esfuerzo les retribuyo, en mínima parte y me esforzaré por ser útil a la sociedad y a la Patria.

Carlos Roberto.

AGRADECIMIENTO:

Dejo constancia de mi profundo reconocimiento para la prestigiosa Escuela Politécnica Nacional y sus dignos profesores; y, en particular, al Sr. Ing. RICARDO DAVILA, por sus sabias enseñanzas y acertada dirección en la elaboración de esta Tesis de Grado y, a todas las personas que me prestaron su valiosa y sincera colaboración.

Carlos R. Egas Acosta.-

I N D I C E

- Introducción.

CAPITULO I

Página

1.1. Tipo de Códigos 2
1.2. Decodificación de máxima probabilidad 4
1.3. Tipos de errores 7
1.4. Estrategias para el control de errores 8

CAPITULO II

2.1. Grupos 11
2.2. Campos 13
2.2.1. Campos finitos 14
2.2.2. Campos de Galois 15
2.3. Espacios vectoriales 21
2.4. Matrices 23

CAPITULO III

3.1. Código de Bloque 27
3.1.1. Síndrome 33
3.1.2. Capacidad para detectar y corregir un error . . . 38

Página

3.1.3. Arreglo Standard 44

3.1.4. Probabilidad de no detectar un error en un canal binario simétrico 48

3.2. Códigos Cíclicos 51

3.2.1. Matriz Generadora y verificadora de paridad . . . 57

3.2.2. Codificación de códigos cíclicos 60

3.2.3. Decodificación de códigos cíclicos 62

3.2.4. Códigos Hamming 68

3.2.5. Códigos Hamming cíclicos 70

3.2.6. Acortamiento de códigos cíclicos 73

3.3. Códigos BCH 80

3.3.1. Decodificación de códigos BCH 86

- Referencias 96

CAPITULO IV

4.1. Consideraciones sobre el hardware 98

4.2. Requerimientos del control de errores 100

4.2.1. Programas que comandan el control de errores . . 103

4.3. Implementación del sistema 106

4.3.1. Microprocesador 107

4.3.2. Decodificador de direcciones 107

4.3.3. Generador de reloj y reset 107

4.3.4. Interfaz programable 109

4.3.5. Display y leds 111

		Página
4.3.6.	Teclado	112
4.3.7.	PROM	113
4.3.8.	RAM	114
4.3.9.	Tarjetas codificadoras y decodificadoras. . .	115
4.3.10.	Fuente de alimentación	115
4.4.	Código de Bloque	116
4.4.1.	Codificador	117
4.4.2.	Decodificador	118
4.4.3.	Algoritmos	120
4.5.	Códigos BCH	124
4.5.1.	Codificador	126
4.5.2.	Decodificación y algoritmo decodificador . . .	127
4.6.	Códigos cíclicos	130
4.6.1.	Codificador	132
4.6.2.	Decodificador	133
4.7.	Diagramas de flujo	140
4.7.1.	Diagramas del código de bloque	151
4.7.2.	Diagramas del código BCH	153
4.7.3.	Diagramas del código cíclico	157
4.8.	Operación del aparato	161
4.8.1.	Configuración externa	164

CAPITULO V.-

5.1.	Resultados experimentales	166

INTRODUCCION

En los últimos años la demanda de sistemas confiables de transmisión y almacenamiento digital ha aumentado considerablemente por la gran aplicación de que son objeto.

Esta confiabilidad puede ser incrementada si se utiliza adecuadamente códigos para la corrección y detección de errores que permiten disminuir considerablemente la probabilidad de una comunicación errónea.

En esta tesis se va a estudiar los conceptos teóricos que se requieren para poder crear sistemas, que utilizando códigos de bloque, permitan detectar o corregir los efectos que producen la introducción de errores en la transmisión digital.

Se estudiarán una clase de códigos detectores y correctores de errores denominados como códigos de bloque. De manera especial se pondrá atención en dos tipos de códigos ampliamente usados en la práctica como son los códigos cíclicos y los códigos BCH.

Se indicará la manera de crear los circuitos codificadores y decodificadores para los tipos de códigos tratados y la manera de optimizarlos de tal manera que su circuitería sea lo más sencilla posible.

Para los códigos BCH se presentará un algoritmo utilizado para su decodificación en la corrección de errores múltiples.

Como aplicación práctica de lo anteriormente visto, se construirá un aparato didáctico con el cual se podrá apreciar detenidamente la manera como estos códigos realizan el control de errores.

Parte importante de este aparato serán los codificadores y decodificadores de dos códigos con parámetros definidos. Estos códigos se denominarán como códigos de bloque y códigos BCH.

Para los códigos cíclicos se brinda la oportunidad de implementar sus circuitos codificadores y decodificadores de una manera independiente al aparato, con parámetros que posean diferentes valores.

CAPITULO I

La probabilidad de error para un sistema particular de comunicaciones función de la relación señal ruido a la entrada del receptor, así como de la velocidad de transmisión. En sistemas prácticos la potencia máxima de la señal y el ancho de banda del canal, son restringidos a valores específicos proporcionados por regulaciones estatales para canales públicos o regulaciones impuestas por compañías privadas, si el canal es arrendado.

Junto con otros parámetros, tales como, la densidad espectral de ruido del canal, número y tipo de señales transmitidas,etc., proporcionan una probabilidad total de error para una aplicación dada. Dados estos parámetros, la única alternativa práctica para reducir la probabilidad de error es el uso de la codificación para el control de errores que equivale a su corrección y detección.

El mayor problema de la ingeniería en la actualidad, es el de implementar canales cofificadores y decodificadores que permitan transmitir y recibir la información en sistemas ruidosos de tal manera de evitar cualquier cambio en la información que ha de ser recibida.

> "El propósito de un canal codificador y decodificador es permitir que una secuencia de datos binarios sea exactamente reproducida a la salida del canal decodificador, y uno de los principales problemas a resolverse es cómo lograrlo." (1)

El control de errores se hace necesario en canales de información y

en sistemas de almacenamiento. Típicos canales de información son por ejemplo las líneas telefónicas, enlaces telemétricos, enlaces de radio en altas frecuencias, enlaces de microondas, enlaces satelitales, entre otros.

Como sistemas de almacenamiento se puede mencionar a memorias de semiconductor o magnéticas, unidades de memoria ópticas, entre otros.

1.1. Tipos de Códigos.-

Actualmente se utilizan con frecuencia dos tipos de códigos. Ellos son los códigos bloque y los códigos convolucionales.

El código bloque al realizar la codificación, divide una secuencia de información en varios bloques cada uno de los cuales posee k - dígitos de información, para luego procesarlos independientemente y - formar lo que se denomina como palabra código, la cual consta de n dígitos. Para esto es necesario que el codificador agregue n-k dígitos, llamados dígitos de control.

El codificador puede formar 2^k palabras código diferentes de longitud n, las cuales conforman el llamado código bloque (n,k).

> "El codificador toma un bloque de k dígitos binarios - sucesivos y los convierte en un bloque equivalente de n - k dígitos binarios. Estos a su vez se suministran a un modulador que genera la forma de onda para la - transmisión". (2).

Los n - k dígitos introducidos en cada bloque de información para formar la palabra código, son los que proporcionan al código la capacidad de combatir al ruido que aparece en la transmisión.

La razón R = k/n es llamada eficiencia de tranamisión y puede ser interpretada como el número de bits de información que entran al codificador por símbolo transmitido.

Los códigos largos con un número relativamente grande de bits de control, reducirían entonces la probabilidad total de error, tales códigos sin embargo son más costosos y difíciles de construir. Cuando los bits de control y la longitud de la palabra código aumenta, el ancho de banda requerido para la transmisión, también crece y la eficiencia de - transmisión disminuye, por lo que el objetivo de diseño es, el de elegir el número de bits de control necesarios para corregir el máximo de errores, pero, manteniendo la eficiencia lo más alta posible.

El codificador de un código convolucional también acepta bloques de k bits de información para producir una secuencia de información llamada palabra código conformada por n símbolos.

En este código cada bloque codificado depende no sólo del correspondiente bloque de mensaje de k bits, que al mismo tiempo esta entrando al codificador, sino también de los m bloques de mensaje anteriores. Es por esto que se dice que el codificador tiene una memoria de orden m.

El conjunto de secuencias codificadas, producidas por k dígitos - de información, de longitud n con memoria de orden m, determinan el código convolucional (n, k, m).

> "El sistema de transmisión por sí mismo puede jugar un papel importante en la determinación del tipo de codificación utilizado" (3).

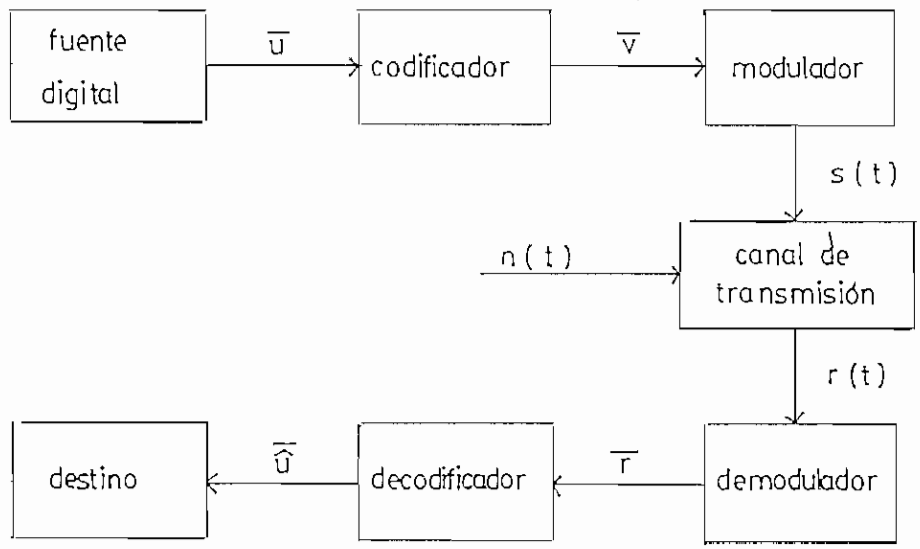
1.2. Decodificación de Máxima Probabilidad.-

La forma más común de disturbio ruidoso presente en un sistema de comunicación es el ruido blanco aditivo gausiano. Si la señal transmitida es s (t), la señal recibida es entonces:

$$r(t) = s(t) + n(t)$$

donde n (t) es el proceso aleatorio gausiano con una densidad de potencia espectral No.

El diagrama de bloques de un sistema que posee un canal de transmisión con ruido blanco gausiano es el siguiente:



La salida $\overline{u}$ de la fuente digital representa el mensaje de k bits a ser codificado. La salida $\overline{v}$ del decodificador representa la palabra código de n símbolos. La salida $\overline{r}$ del demodulador representa la correspondiente secuencia recibida de n dígitos.

El decodificador produce una secuencia de información $\overline{\hat{u}}$, la cual

es estimada en base a la secuencia recibida $\bar{r}$. Existe una correspondencia de uno a uno entre secuencia de información $\bar{u}$ y la palabra código $\bar{v}$, el decodificador por lo tanto estima una palabra código $\bar{v}$, claramente - $\bar{\hat{u}} = \bar{u}$ si y sólo si $\bar{r} = \bar{v}$.

> "El fundamental problema de la comunicación es que la reproducción en un punto sea exactamente o aproximadamente el mensaje seleccionado en otro punto" (4).

Si la palabra código $\bar{v}$ es transmitida, una decodificación errónea ocurre si y sólo si $\bar{\hat{v}} \neq \bar{v}$. Puesto que $\bar{r}$ es recibida, la probabilidad-condicional de error del decodificador esta definida como:

$$P\left(\frac{E}{\bar{r}}\right) \triangleq P\left(\frac{\bar{\hat{v}} \neq \bar{v}}{\bar{r}}\right) \qquad 1.1.$$

y por lo tanto la probabilidad de error del decodificador esta dada por:

$$P(E) = \sum_{\bar{r}} P(E/\bar{r})\, P(\bar{r}) \qquad 1.2$$

Si $P_{\cdot}(\bar{r})$ es independiente del tipo de decodificación usado, una óptima-decodificación se da cuando $P(E/\bar{r}) = P(\bar{\hat{v}} \neq \bar{v}/\bar{r})$ toma el valor mínimo para - todo $\bar{r}$.

Minimizar $P(\bar{\hat{v}} \neq \bar{v}/\bar{r})$ es equivalente a maximizar $P(\bar{\hat{v}} = \bar{v}/\bar{r})$

con $$P(\bar{v}/\bar{r}) = \frac{P(\bar{r}/\bar{v})\, P(\bar{v})}{P(\bar{r})} \qquad 1.3.$$

De esta forma, $\bar{v}$ es la palabra código más probable que se escoge, en base del vector recibido $\bar{r}$.

Si toda la secuencia de información y todas las palabras código son igualmente probables, es decir, P ($\bar{v}$) tiene el mismo valor para todo $\bar{v}$, maximizar la expresión 1.3 es equivalente a maximizar P ($\bar{r}/\bar{v}$).

Para un canal discreto, se tiene que

$$P(\bar{r} / \bar{v}) = \prod_i P(r_i / v_i) \qquad 1.4$$

ya que, en un canal sin memoria cada símbolo recibido depende sólo del correspondiente símbolo transmitido. Un decodificador que escoja una secuencia estimativa al maximinizar la expresión 1.3 es llamado decodificador de máxima probabilidad. DMP.

> "En muchos sistemas, las probabilidades de la palabra código no son conocidas exactamente en el receptor, haciendo imposible una decodificación óptima, y a un decodificador de máxima probabilidad le conviene una mejor y más factible regla de decodificación" (5).

En un canal binario simétrico la decodificación de máxima probabilidad escoge a $\bar{v}$ como la palabra código $\bar{v}$, minimizando la distancia d ($\bar{r},\bar{v}$) entre $\bar{r}$ y $\bar{v}$. En otras palabras el decodificador de máxima probabilidad escoge la palabra código que difiere de la secuencia recibida en el menor número de posiciones y es por esta razón que también se conoce al decodificador de máxima probabilidad para un canal binario simétrico como decodificador de distancia mínima o de mínimo error.

> "La regla de decodificación por probabilidad de mínimo error es una regla que minimiza la probabilidad de decodificación errónea dado un mensaje recuperado, un conjunto de palabras código y un canal" (6).

La capacidad de un canal ruidoso para transmitir información confiable fue determinado por Shanon y cuyo resultado fue llamado teorema de codificación de un canal ruidoso, el cual establece que cada canal de información posee una capacidad de canal C y que para cualquier eficiencia de transmisión R tal que $R<C$ existen códigos que al utilizar la decodificación de máxima probabilidad poseen una pequeña probabilidad de decodificación errónea P (E).

1.3. Tipos de Errores.-

La transmisión errónea en un sistema de comunicación digital es causada por ruido introducido en el canal de comunicación, generalmente dos tipos de ruido pueden ser identificados como los causantes de los errores en la transmisión.

El primer tipo de ruido llamado ruido gausiano, el cual tiene relación con los errores aleatorios y el segundo tipo de ruido llamado ruido impulsivo, generalmente causado por trascientes, es el causante de los errores tipo ráfaga.

En canales de transmisión en los cuales la señal recibida depende sólo de la señal transmitida, el ruido afecta a cada símbolo transmitido independientemente de los demás y la transmisión de errores ocurre aleatoriamente en la secuencia recibida.

Estos canales se llaman canales de error aleatorio y ejemplos típicos de estos canales son el espacio libre y muchos canales satelitales. Los códigos diseñados para corregir errores aleatorios son llamados

códigos correctores de errores aleatorios.

En los canales con memoria, el ruido no es independiente de una - transmisión a otra, ya que la decodificación depende de los estados anteriores. A estos canales se los llama canales de error tipo ráfaga y - como ejemplos se tiene a los canales de radio, transmisión por cable - que puede ser afectado por ruido de switcheo, y grabación magnética, la cual esta sujeta a los defectos de la cinta y partículas indeseables.

> "Dos tipos de errores condicionan la transmisión digital de datos. El primer tipo de errores aleatorios implican una no correlación entre los dígitos en el- error. El segundo tipo implica errores tipo ráfaga,- en el cual un número consecutivo de dígitos, más que individuales, son afectados" (7).

Los canales que contienen errores aleatorios como tipo ráfaga se denominan canales compuestos que utilizan códigos correctores de errores aleatorios y tipo ráfaga.

1.4. Estrategias para el Control de Errores.-

La manera en que se decida realizar el control de errores dependerá mucho del sistema de comunicación y de la aplicación muy particular- de ese sistema.

Cuando la transmisión de información es estrictamente en una dirección actuando un punto solo como transmisor y otro sólo como receptor,- el control de errores para estos sistemas de una sola via puede ser realizado, utilizando la corrección directa de errores, empleando códigos-

en los cuales automáticamente el decodificador corrige los errores detectados en el receptor.

Tal es el caso de sistemas de almacenamiento en cinta magnética - en el cual, la información almacenada puede ser recuperada mucho tiempo después sin errores gracias a la codificación de la información antes - de su almacenamiento. Los errores son corregidos en el momento de recuperar la información.

Cuando los sistemas de información son de dos vías, es decir la - información se envía en ambos sentidos, actuando el transmisor como receptor y viceversa (transreceptor), el control de errores puede requerir tan solo de la detección de errores y el consiguiente pedido de retransmisión de la información denominado pedido automático de repetición ARQ.

Como ejemplo de estos sistemas se puede mencionar, los canales telefónicos, algunos sistemas de comunicación satelitales, entre otros.

Los sistemas de codificación y decodificación de códigos correctores de errores son mucho más complicados que los sistemas utilizados solo para la detección. La utilización en forma particular o en forma conjunta de estas dos maneras para el control de errores, como se ve, mucho dependerá del campo de aplicación.

> "El canal codificador y decodificador son bloques funcionales en el sistema que, por labor conjunta, reducen la probabilidad de error" (8).

REFERENCIAS :

1. Gallager R. INFORMATION THEORY AND REALIABLE COMUNICATION, pág. 23.

2. Misha Schwartz. TRANSMISION DE INFORMACION MODULACION Y RUIDO, pág. 553.

3. R. F. Coates. MODERN COMUNICATION, pág. 219.

4. Mc Eliece R. THE THEORY OF INFORMATION AND CODING, pág. 1

5. Shu Lin, Costelo. ERROR CONTROL CODING. FUNDAMENTALS AND APLICA - TIONS, pág. 10.-

6. Gallager R. INFORMATION THEORY AND REALIABLE COMUNICATION, pág. 120.-

7. R. F. Coates. MODERN COMUNICATION, Pág. 219.

8. K. Sam Shanmugan. DIGITAL AND ANALOG COMUNICATION SYSTEMS, pág. 219.

*** & ***

CAPITULO II

Para una adecuada comprensión de los procesos de codificación y de codificación de los códigos de bloque lineales, se necesita entender - ciertos conceptos matemáticos en los cuales se basa la teoría del control de errores para códigos no convolucionales.

Estos conceptos matemáticos también nos indicarán la manera óptima de realizar la implementación de dichos códigos.

Se enunciaran los teoremas que se creen necesarios y cuya aplica - ción práctica se lo hará en capítulos posteriores. Muchas demostraciones se omitirán, ya que se las puede hallar en cualquier libro de matemáticas básicas y sólo se mencionarán demostraciones que sean fundamentales para el desarrollo de la teoría de codificación y decodificación de este trabajo.

2.1. Grupos.-

Si G es un conjunto de elementos, en el cual la operación binaria "*" es una regla que asigna a cada par de elementos a y b un tercer elemento c tal que $c = a*b$ y $c \in G$. Entonces G es cerrado bajo la operación "*".

Un grupo es un conjunto G en el cual la operación binaria "*", esta

definida y en el cual se cumplen las siguientes propiedades:

Asociativa: $a * (b*c) = (a*b)*c$

Modulativa: $a = a*e = e*a$ e elemento identidad

Invertiva: $a*a' = a'*a = e$ $a, a' \in G$

Un grupo G es conmutativo si la operación binaria "*", satisface la siguiente condición.

$$a*b = a*b$$

- Teorema No. 1 El elemento identidad en un grupo G es único.

- Teorema No. 2 El inverso de cualquier elemento del grupo es único y todo elemento del grupo tiene un inverso.

El número de elementos de un grupo es el orden del grupo y un grupo con un número finito de elementos es un grupo finito.

Para cualquier entero positivo m, es posible construir un grupo de orden m bajo una operación binaria muy similar a la multiplicación de reales, o bajo una operación binaria muy similar a la adición de reales.

Dado un subconjunto H de G, se dice que H es un subgrupo de G si H cerrado bajo la operación binaria de G y si además satisface todas las condiciones de un grupo.

Dado a un elemento de un grupo finito G y la secuencia de elementos a, a^2 , a^3 , donde $a^2 = a.a$, entonces si i y j son exponentes, tales que; $a^i = a^j = a^i a^j$ con $j > i$

tenemos que estos elementos forman un subgrupo cíclico.

Suponiendo que los elementos de un grupo G son g, g2, g3 .... y los elementos de un subgrupo H son h1 , h2, h3 ... se puede considerar el siguiente arreglo:

$$\begin{array}{rcl} h_1 & = & 1 \ , \ h_2 \ , \ h_3 \ , \ h_4 \ , \ \ldots \ h_n \\ g_1 h_1 & = & g_1 \ , \ g_1 h_2 , \ g_1 h_3 , \ g_1 h_4 \ , \ \ldots \ g_1 h_n \\ g_2 h_1 & = & g_2 \ , \ g_2 h_2 , \ g_2 h_3 , \ g_2 h_4 \ , \ \ldots \ g_2 h_n \\ \vdots & & \vdots \\ g_m h_1 & = & g_m , \ g_m h_2 , \ g_m h_3 , \ g_m h_4 \ , \ \ldots \ g_m h_n \end{array} \qquad 2.1$$

Los elementos de una fila de este arreglo forman un coset, y el elemento que aparece en la primera columna se lo llama líder del coset.

2.2. Campos.-

Una considerable parte de la teoría algébrica de codificación se mueve alrededor del concepto de campo finito.

Si F es un conjunto en el cual estan definidas dos operaciones binarias llamadas adición "+" y multiplicación " . " , F se llama campo si junto con las dos operaciones, se cumplen las siguientes condiciones:

- F es un grupo conmutativo cerrado bajo la adición. El elemento identidad con respecto a la adición es llamado elemento cero ó la identidad aditiva de F y se lo representa como 0.

- El conjunto de elementos diferentes de cero que pertenecen a F, es un grupo conmutativo bajo la multiplicación, y su correspondiente elemento identidad se llama elemento unitario y se representa con 1.

- La multiplicación es distritutiva sobre la adición.

2.2.1. Campos Finitos.-

El número de elementos del campo determina el orden del campo y un campo con un número finito de elementos es un campo finito.

Las propiedades principales de los campos son las siguientes:

- Propiedad 1.- Para cualquier elemento de un campo se tiene que:

$$a \cdot 0 = 0 \cdot a = 0$$

- Propiedad 2.- Si dos elementos de un campo son diferentes de cero - se tiene que:

$$a \cdot b \neq 0$$

- Propiedad 3.- Si $a.b = 0$ y $a \neq 0$ implica que $b = 0$.

- Propiedad 4.- Para dos elementos cualesquiera de un campo:

$$-(a.b) = (-a).b = a.(-b)$$

- Propiedad 5.- Si $a \neq 0$ y $a.b = a.c$ entonces $b = c$.

Para el conjunto (0,1) podemos definir la adición módulo 2 y la multiplicación módulo 2 de la siguiente manera:

Adición

+	0	1
0	0	1
1	1	0

Multiplicación

•	0	1
0	0	0
1	0	1

El conjunto y las operaciones definidas anteriormente forman un - campo binario, representado por GF (2). Este campo es la base matemática para el estudio de los sistemas de transmisión y almacenamiento digital.

Para cualquier número primo p, existe un campo finito de p elementos representado como GF (p), además para cualquier entero positivo m es posible extender el número de elementos del campo primo GF (p) de- p elementos a un campo de p^m elementos representado por GF (p^m).

2.2.2. Campos de Galois.-

Los campos finitos son llamados campos de Galois en honor a su - descubridor y su importancia en la teoría de control de errores radica- en que la construcción de códigos y los procesos de codificación y decodificación giran alrededor de estos campos finitos. Concretamente, en- nuestro caso, serán muy útiles para el estudio de códigos BCH.

Para un campo finito GF (q) existen dos enteros positivos m y n tal que:

$$\sum_{i=1}^{n-m} 1 = 0 \qquad m < n$$

por lo cual existe un entero positivo de menor valor representado por- λ tal que se cumple la siguiente condición:

$$\sum_{i=1}^{\lambda} 1 = 0$$

Este entero λ es llamado "característica del campo GF (q)". Así tenemos que la característica del campo binario GF (2) tiene el valor de 2 ya que 1+ 1 = 0.

- Teorema 3.- La característica λ de un campo finito es un número primo,

- Teorema 4.- Si a es un elemento diferente de cero perteneciente a un campo finito GF (q) y n es el orden de a tal que $a^n = 1$, entonces se tiene que n divide a q - 1.

- Teorema 5.- Si a es un elemento diferente de cero que pertenece a un campo finito GF (q) entonces se cumple la siguiente relación:

$$a^{q-1} = 1$$

En un campo finito GF (q) un elemento diferente de cero es primitivo si el orden de este elemento es q-1. Las potencias de un elemento primitivo generan todos los elementos diferentes de cero de GF (q) y cualquier campo finito tiene su elemento primitivo.

En la práctica se pueden construir códigos con símbolos que pertenezcan a un campo de Galois GF (q) donde q puede ser un número primo, representado por p, o una potencia de él.

Los códigos más ampliamente usado son aquellos cuyos símbolos pertenecen a campos binarios GF (2) o a sus extensiones GF (2^m) ya que la información que se transmite o se almacena en los sistemas de comunicación es universalmente codificada en formas binaria por muchas razones prácticas.

El análisis de control de errores requiere en muchas ocasiones, de diversas formas matemáticas para cumplir su objetivo. Es así como se utiliza polinomios, cuyos coeficientes pertenecen a GF (2), para representar una palabra codificada.

$$f(x) = f_0 + f_1 x^1 + f_2 x^2 + \cdots\cdots f_n x^n \qquad 2.2$$

donde $f_i = 0$ ó 1 para $0 \leq i \leq n$, el grado del polinomio es la potencia de más alto valor de X con coeficiente diferente de cero.

Los polinomios sobre GF (2) pueden ser sumados, restados, multiplicados y divididos entre sí, utilizando para ello la multiplicación y suma módulo 2. Se cumplen las siguientes propiedades:

- Conmutativa $a(x)+b(x) = b(x)+a(x)$

  $a(x) \cdot b(x) = b(x) \cdot a(x)$

- Asociativa $a(x)+[b(x)+c(x)] = [a(x)+b(x)]+c(x)$

  $a(x) \cdot [b(x) \cdot c(x)] = [a(x) \cdot b(x)] \cdot c(x)$

- Distritutiva

  $a(x) \cdot [b(x)+c(x)] = [a(x) \cdot b(x)]+[a(x) \cdot c(x)]$

La división de un polinomio f (x) para otro g (x) diferente de cero se la puede expresar de la siguiente manera:

$$f(x) = q(x) \cdot g(x) + r(x)$$

q (x) se llama cociente.

r (x) es el residuo con grado menor a g (x).

Un polinomio p (x) sobre GF (2) cuyo grado es m, es irreducible sobre GF (2) si p (x) no es divisible para cualquier polinomio sobre GF (2) cuyo grado sea menor o igual a m.

- Teorema 6.- Cualquier polinomio sobre GF (2) que sea irreducible y de-
  grado m divide a la expresión $X^{2^m-1} + 1$

Un polinomio irreducible p (x) de grado m es primitivo si el más pequeño entero positivo n para el cual p (x) divide a $x^n + 1$ es $n = 2^{m-1}$

No es fácil reconocer a polinomios primitivos, ya que para una m da da pueden existir más de un polinomio primitivo de grado m. Es por este motivo, que es necesario tener presente la siguiente lista de polinomios primitivos que tienen el más pequeño número de términos:

m = 3	$1 + X + X^3$	
4	$1 + X + X^4$	
5	$1 + X^2 + X^5$	
6	$1 + X + X^6$	
7	$1 + X^3 + X^7$	
8	$1 + X^2 + X^3 + X^4 + X^8$	
9	$1 + X^4 + X^9$	2.3
10	$1 + X^3 + X^{10}$	
11	$1 + X^2 + X^{11}$	
12	$1 + X + X^4 + X^6 + X^{12}$	
13	$1 + X + X^3 + X^4 + X^{13}$	
14	$1 + X + X^6 + X^{10} + X^{14}$	
15	$1 + X + X^{15}$	

A continuación se verán algunas características y propiedades de los campos binarios de Galois de 2^m elementos con $m>1$ que resultan de la extensión de GF (2) que tiene como sus elementos a 0 y 1.

Los elementos de GF (2^m) son representados en base a los elementos 0, 1 y un nuevo elemento representado por α .

Entre estos elementos existen las siguientes relaciones.

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

$$
\begin{aligned}
0 \cdot \alpha &= \alpha \cdot 0 = 0 \\
1 \cdot \alpha &= \alpha \cdot 1 = \alpha \\
\alpha^2 &= \alpha \cdot \alpha \\
\alpha^3 &= \alpha \cdot \alpha \cdot \alpha \\
&\vdots \\
\alpha^j &= \alpha \cdot \alpha \cdot \alpha \cdots \alpha \quad (j \text{ veces})
\end{aligned}
\qquad 2.4
$$

Esta es la manera de generar todos los elementos de GF (2^m), con los cuales se puede formar un conjunto F en el cual está definida la operación multiplicación " . "

$$F = (0, 1, \alpha, \alpha^2, \ldots \alpha^j)$$

Los elementos generados a partir de α llegan hasta un elemento dado por α^{2^m-2}, a partir del cual comienzan a repetirse. Estos elementos forman un conjunto finito F^* que forman un grupo de orden 2^m cerrado bajo la suma y multiplicación módulo dos

$$F^* = (0, 1, \alpha, \alpha^2, \ldots \alpha^{2^m-2})$$

Los 2^{m-1} elementos diferentes de cero representados por α^0, α^1 $\alpha^2, \ldots \alpha^{2^m-2}$ en F^*, también pueden ser representados en forma polinomial por 2^{m-1} polinomios diferentes de cero sobre GF (2) que tienen un grado menor o igual a m-1. El elemento cero será representado por el polinomio cero.

Una representación exponencial de los elementos de GF (2^m) es conveniente cuando se realiza la multiplicación y división, la representación polinomial, en cambio, es útil para la adición.

- Teorema 7.- Dado el polinomio f (x) cuyos coeficientes pertenecen a GF (2) y β un elemento perteneciente a una extensión del campo GF (2^m). Si β es una raíz de f (x), entonces para cualquier $z \geq 0$, β^{2z} es también una raíz de f (x).

- Teorema 8.- Los 2^{m-1} elementos diferentes de cero pertenecientes a GF (2^m), son todas las raíces de $X^{2^m-1} + 1$

De esta forma se puede deducir que los elementos de GF (2^m) son todas las raíces de $X^{2^m} + X$

Si cualquier elemento β en GF (2^m) es raíz del polinomio $X^{2^m} + X$ entonces β puede ser raíz de un polinomio sobre GF (2) con grado menor o igual a 2^m .. Si $\phi(x)$ es el polinomio de grado más pequeño sobre GF (2) tal que $\phi(\beta)=0$, entonces este polinomio $\phi(x)$ es el polinomio mínimo de β siendo además irreducible.

- Teorema 9.- Dado un polinomio f (x) sobre GF (2) y el polinomio mínimo $\phi(x)$ de un elemento de campo β , entonces f (x) es divisible por $\phi(x)$ si β es raíz de f (x).

- Teorema 10.- Si β es un elemento en GF (2^m) y e es el entero no negativo más pequeño tal que $\beta^{2^e} = \beta$ entonces

$$f(x) = \prod_{i=0}^{e-1} (x + \beta^{2^i}) \qquad 2.5$$

F (x) es un polinomio irreducible de GF (2)

- Teorema 11.- Dado $\phi(x)$ el polinomio mínimo de un elemento β de GF(2^m) y e el entero más pequeño tal que $\beta^{2^e} = \beta$, entonces

$$\phi(x) = \prod_{i=0}^{e-1} (x + \beta^{2^i})$$

Este teorema nos proporciona la manera para encontrar el polinomio mínimo de un elemento del campo.

- Teorema 12.- Si $\phi(x)$ es el polinomio mínimo de un elemento β que pertenece a GF (2^m) y e es el grado de $\phi(x)$, entonces e es el elemento -

más pequeño tal que $\beta^{2^e} = \beta$ con $e \leq m$.

- Teorema 13.- Si β es un elemento primitivo de GF (2^m), entonces, todos los conjugados $\beta^1, \beta^{2^1}, \ldots$ son también elementos primitivos de GF (2^m).

Todos estos teoremas y propiedades enunciados anteriormente, los cuales rigen el comportamiento de los campos GF (2^m), son importantes para el diseño e implementación de los codificadores y decodificadores de los códigos BCH, ya que la construcción de campos GF (2^m) y la utilización de estos conceptos simplifica mucho la elaboración de estos códigos.

2.3. Espacios Vectoriales.-

Dados; V un conjunto de elementos, en el cual, la operación binaria llamada adición + está definida, y un campo F. Si la operación multiplicación denotada por "." está definida entre los elementos de F y V, se dice entonces que V es un espacio vectorial sobre el campo F si se satisfacen las siguientes condiciones.

- V es un grupo conmutativo bajo la adición.
- Para cualquier elemento a en F y cualquier elemento $\overline{v}$ en V $a.\overline{v}$ es un elemento de V.
- Para cualquier elemento $\overline{u}$ y $\overline{v}$ en V y cualquier elemento a y b en F se cumple que

$$a.(\overline{u}+\overline{v}) = a.\overline{u} + b.\overline{v}$$

$$(a+b).\overline{v} = a.\overline{v} + b.\overline{v}$$

$$(a.b).\overline{v} = a.(b.\overline{v})$$

- Si 1 es el elemento unitario de F entonces para cualquier $\overline{v}$ en V, 1. $\overline{v}$ = $\overline{v}$.

Además si c es un escalar y $\overline{v}$ pertenece a V se tiene que

$$(-c).\overline{v} = c.(-v) = -(c.\overline{v})$$

$$0.\overline{v} = 0$$

- Teorema 14.- Si S es un subconjunto del espacio vector V sobre un campo F, entonces, S es un subespacio de V si se cumplen las siguientes condiciones.

  - Para dos elementos cualesquiera $\overline{u}$ y $\overline{v}$ en S, $\overline{u} + \overline{v}$ es también un vector en S.

  - Para cualquier elemento a en F y cualquier vector $\overline{u}$ en S, a.$\overline{u}$ pertenece a S.

- Teorema 15.- Dados $\overline{v}_1, \overline{v}_2, \ldots\ldots \overline{v}_k$, k vectores en un espacio vectorial V sobre un campo F, el conjunto de todas las combinaciones lineales de $\overline{v}_1$ $\overline{v}_2$, $\ldots\ldots\overline{v}_k$ forman un subespacio de V.

Como consecuencia un conjunto de vectores $\overline{v}_1$, $\overline{v}_2$, $\ldots\ldots$ $\overline{v}_k$ se dice que es linealmente independiente si para que $a_1\overline{v}_1 + a_2\overline{v}_2 + \ldots a_k\overline{v}_k = 0$ debe cumplirse que $a_1 = a_2 = a_3 = \ldots\ldots a_k = 0$

Un conjunto de vectores se llama generador de un espacio vectorial V, si cada vector en V es una combinación lineal de los vectores de dicho conjunto. A este conjunto también se lo llama base del espacio vectorial, al cual generan y el número de vectores que forman una base determina la dimensión del espacio vectorial.

Una propiedad importante es la ortogonalidad de dos vectores. Dados dos vectores $\bar{u}$ y $\bar{v}$ que pertenecen a V, se dice que $\bar{u}$ y $\bar{v}$ son ortogonales si cumplen las siguientes condiciones

$$\bar{u} \cdot \bar{v} = \bar{v} \cdot \bar{u}$$

$$\bar{u} \cdot (\bar{v} + \bar{w}) = \bar{u} \cdot \bar{v} + \bar{u} \cdot \bar{w}$$

$$(a \cdot \bar{u}) \cdot \bar{v} = a(\bar{u} \cdot \bar{v})$$

2.4. Matrices.-

Una matriz $k \times n$ sobre GF (2), es decir cuyos elementos pertenezcan al campo GF (2), es un arreglo de k filas y n columnas tal como se indica a continuación

$$G = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-1} \\ \vdots & \vdots & & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix} = \begin{bmatrix} \bar{g}_0 \\ \bar{g}_1 \\ \vdots \\ \bar{g}_{k-1} \end{bmatrix} \qquad 2.6$$

donde cada elemento $g_{i,j}$ con $0 \leq i \leq k$ y $0 \leq j < n$, es un elemento binario.

Si las k filas de la matriz son linealmente independientes, entonces las 2^k combinaciones lineales de estas filas forman un subespacio k dimensional del espacio vector V.

Se pueden intercambiar cualesquiera de las filas una con otra para obtener otra matriz G' la cual genera el mismo subespacio obtenido a partir de G.

Para cualquier matriz $k \times n$ G sobre GF (2) con k filas linealmente

independientes, existe una matriz $|H$ $(n-k) \times n$ sobre GF (2) con n-k filas linealmente independientes tal que para cualquier $\bar{g}_i$ en $(G$ y cualquier $\bar{h}_j$ en $|H$ se tiene que $\bar{g}_i \cdot \bar{h}_j = 0$

Dos matrices pueden ser sumadas si tienen el mismo número de filas y de columnas. Sumar dos matrices representados como $/A = [a_{ij}]$ y $|B = [b_{ij}]$ es simplemente sumar sus elementos correspondientes

$$[a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}] \qquad 2.7$$

Multiplicar una matriz $k \times n$ $/A = [a_{ij}]$ por una matriz $n \times 1$ $|B = [b_{ij}]$ es obtener otra matriz $(C$ dada por

$$(C = /A \times |B = [c_{ij}]$$

tal que $[c_{ij}]$ viene dada por

$$\bar{c}_{ij} = \bar{a}_i \cdot \bar{b}_j = \sum_{t=0}^{n-1} a_{it} \cdot b_{tj} \qquad 2.8$$

CAPITULO III

3.1. Código de Bloque.-

Una vez que se han introducido los elementos básicos matemáticos para la mejor comprensión de las reglas que rigen a los códigos de bloque lineales, tanto en su parte teórica como en su implementación, vamos a empezar su estudio cualitativa y cuantitativamente.

> "La razón porque restringimos nuestra atención a códigos bloque no es porque ellos sean los mejores en cualquier sentido con respecto a otros tipos de códigos, simplemente es porque ellos son fácilmente tratables". (1)

En la comunicación digital de datos, la información es procesada como una secuencia de dígitos binarios que toman el valor cero a uno lógico. Cuando nosotros queremos generar un código bloque es necesario segmentar toda la secuencia de información a transmitirse en bloques, cada uno de los cuales será codificado independientemente.

Cada bloque tendrá información binaria y será representado por el vector $\bar{u}$, el cual contiene k dígitos de información, con lo que se tendrá la posibilidad de tener 2^k bloques distintos de información (mensajes diferentes) que podemos transmitir.

El codificador, por cada mensaje de k dígitos, generará un nuevo bloque de información $\bar{v}$ de n elementos que cumplirán con la condición de

n>K. De esta manera se ha incrementado en n-k dígitos el bloque inicial, denominándose a estos dígitos verificadores de paridad o de control.

"Un gran porcentaje de dígitos verificadores incrementa la habilidad correctora del código" (2)

Al nuevo bloque $\bar{v}$ se lo llamará en adelante, palabra de código o vector de código del mensaje $\bar{u}$, correspondiendo a los 2^k mensajes, 2^k vedores de código. A este conjunto de vectores se le llama CODIGO BLOQUE.

Este código bloque puede ser fácilmente codificado o decodificado si su estructura posee linealidad, de lo contrario será muy complejo hacerlo.

Un código bloque de longitud n y de 2^k palabras código se dice que es un código lineal (n,k) si y solo si estas 2^k palabras de código forman un subespacio k dimensional, del espacio formado por todas las n-tuplas que pertenecen al campo GF (2).

Un código bloque binario es lineal si y solo si la suma módulo dos de dos palabras de código es también una palabra de código.

Si C es un subespacio de dimensión k del espacio Vn, es posible encontrar k palabras linealmente independientes $\bar{g}_0, \bar{g}_1, \ldots \bar{g}_{k-1}$ en C tal que cualquier palabra de código en C resulta de una combinación lineal de estas k palabras de código.

Una ventaja inmediata de los códigos lineales sobre los no lineales,

es que los primeros pueden ser fácilmente especificados y descritos por cualquier conjunto de k palabras código linealmente independientes.

Se puede construir una matriz k x n, cuyas filas son las k palabras de código linealmente independientes.

> "Cualquier conjunto de vectores base de un código lineal C puede ser considerado como las filas de una matriz (G , llamada matriz generadora de C". (3).

Esta matriz tiene la siguiente forma;

$$(G = \begin{bmatrix} \overline{g}_0 \\ \overline{g}_1 \\ \overline{g}_2 \\ \vdots \\ \overline{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & g_{0,2} & \cdots\cdots & g_{0,n-1} \\ g_{1,0} & g_{1,1} & g_{1,2} & \cdots\cdots & g_{1,n-1} \\ g_{2,0} & g_{2,1} & g_{2,2} & \cdots\cdots & g_{2,n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots\cdots & g_{k-1,n-1} \end{bmatrix} \quad 3.1$$

donde $\overline{g}_i = (g_{i,0} , g_{i,1} , g_{i,2} \ldots . g_{i,n-1}) \qquad 0 \leq i \leq k$

Si la información a codificar es $\overline{u} = (u_0 , u_1 , u_2 \ldots\ldots u_{k-1})$ la correspondiente palabra codificada $\overline{v}$ se la obtiene realizando la siguiente operación:

$$\overline{v} = \overline{u} . (G \qquad 3.2$$

$$\overline{v} = (u_0 , u_1 , u_2 \ldots\ldots u_{k-1}) \cdot \begin{bmatrix} \overline{g}_0 \\ \overline{g}_1 \\ \overline{g}_2 \\ \vdots \\ \overline{g}_{k-1} \end{bmatrix} \qquad 3.3.$$

$$\overline{v} = u_0 \overline{g}_0 + u_1 \overline{g}_1 + u_2 \overline{g}_2 + \ldots\ldots u_{k-1} \overline{g}_{k-1} \qquad 3.4$$

Como se puede observar la matriz G es suficiente para generar un código lineal (n,k) , por esta razón la matriz G es llamada matriz generadora de C.

> "Un código lineal (n,k) está completamente especificado por las k filas de la matriz generadora G . Por lo tanto el codificador sólo tiene que almacenar las k filas de G". (4).

Bajo cierta estructura de la matriz G , es posible obtener la palabra codificada con la siguiente propiedad:

bits introducidos en la codificación	bits de la información que se codifico

Cuando la palabra de código posee esta estructura, se dice que el código bloque lineal tiene una estructura sistemática y se lo denomina - código lineal sistemático.

> "Todos los métodos obtenidos para protección de errores requieren la adición extra de dígitos a la palabra código binaria básica. Estos dígitos son usados para verificar la validez de los dígitos de información". (5)

Un código lineal sistemático (n,k) esta completamente definido por una matriz G $k \times n$ que posea la siguiente forma:

$$G = \left[\begin{array}{cccc:cccc} P_{0,0} & P_{0,1} & P_{0,2} & \cdots P_{0,n-k-1} & 1 & 0 & 0 \cdots & 0 \\ P_{1,0} & P_{1,1} & P_{1,2} & \cdots P_{1,n-k-1} & 0 & 1 & 0 \cdots & 0 \\ P_{2,0} & P_{2,1} & P_{2,2} & \cdots P_{2,n-k-1} & 0 & 0 & 1 \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{k-1,0} & P_{k-1,1} & P_{k-1,2} & \cdots P_{k-1,n-k-1} & 0 & 0 & 0 \cdots & 1 \end{array}\right] \qquad 3.5$$

$\leftarrow P \rightarrow \quad \leftarrow I \rightarrow$

donde p_{ij} = 0 ó 1

$|I$ es la matriz identidad $k \times k$

$|P$ es una matriz $k \times (n-k)$

> "Un paso importante en el diseño de un código bloque (n,k) es la selección de una matriz $|P$ tal que el código generado por (G tenga ciertas propiedades tales como; fácil implementación, habilidad para corregir errores aleatorios o tipo-ráfaga". (6).

Desarrollando la expresión 3.2. se tiene que:

$$\overline{v} = (v_0, v_1, \ldots, v_{n-1}) = (u_0, u_1, \ldots, u_{k-1}) \cdot (G \qquad 3.6$$

donde $v_{n-k+i} = u_i \qquad 0 \leq i < k$ 3.7

y $v_j = u_0 p_{0j} + u_1 p_{1j} + u_2 p_{2j} \ldots\ldots + u_{k-1} p_{k-1,j}$ para $0 \leq j < n-k$ 3.8

con lo cual $\overline{v} = (v_0, v_1, v_2, \ldots, v_{n-k-1} u_0, u_1, \ldots, u_{k-1})$ 3.9

En la palabra código quedan definidos claramente la parte del mensaje y la parte de los bits introducidos en la codificación.

> "La selección de la proporción de redundancia dentro de un código bloque depende mucho de la naturaleza de aplicación del código". (7)

Como se puede notar estos bits introducidos son el resultado de la suma lineal de los dígitos de información. Las (n-k) ecuaciones dadas por 3.8 son llamadas ecuaciones verificadoras de paridad del código.

Si para cualquier (G $k \times n$ con k filas linealmente independientes - existe una matriz $|H$ $(n-k) \times n$ con n-k filas linealmente independiente tal

que cualquier vector fila en $\mathbb{G}$ es ortogonal a las filas de $\mathbb{H}$, nosotros podemos decir que una n-tupla $\bar{v}$ es una palabra código generado por $\mathbb{G}$ si y solo si

$$\bar{v} \cdot \mathbb{H}^T = 0 \qquad 3.10$$

Si v_j y h_{ij} son elementos de $\bar{v}$ y $\mathbb{H}$ respectivamente, la expresión anterior puede ser representada como:

$$\sum_j v_j \; h_{ij} = 0 \qquad 3.11$$

esto significa que los componentes de $\bar{v}$ satisfacen un conjunto de n-k - ecuaciones linealmente independientes, las cuales se las llama ecuaciones verificadoras de paridad.

La matriz $\mathbb{H}$ es llamada matriz verificadora de paridad del código. Las 2^{n-k} combinaciones lineales de las filas de la matriz $\mathbb{H}$ forman un - código lineal (n,n-k) representado por C_d cuyos elementos son todos diferentes a los que pertenecen al código C, es decir que para todo $\bar{v}$ que pertenece a C y para todo $\bar{w}$ que pertenece a C_d se tiene que:

$$\bar{V} \cdot \bar{W} = 0 \qquad 3.12$$

a C_d se lo conoce con el nombre de código dual de C.

La matriz verificadora de paridad viene dada en base a los mismos- elementos que conforman la matriz generadora.

$$\mathbb{H} = \left[\mathbb{I}_{n-k} \; \mathbb{P}^T \right] \qquad 3.13$$

$$|H = \left[\begin{array}{cccc|cccc} 1 & 0 & \cdots & 0 & P_{0,0} & P_{1,0} & \cdots & P_{k-1,0} \\ 0 & 1 & \cdots & 0 & P_{0,1} & P_{1,1} & \cdots & P_{k-1,1} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & P_{0,n-k-1} & P_{1,n-k-1} & \cdots & P_{k-1,n-k-1} \end{array}\right] = \left[\begin{array}{c} \bar{h}_0 \\ \bar{h}_1 \\ \vdots \\ \bar{h}_{n-k} \end{array}\right] \qquad 3.14$$

$|P^T$ es la matriz transpuesta de $|P$ además entre las filas de $|H$ y $(G$ se tiene que:

$$\bar{g}_i \cdot \bar{h}_j = P_{ij} + P_{ij} = 0 \qquad 3.15$$

para $0 \leqslant i < k$ y $0 \leqslant j < n-k$

o lo que es lo mismo

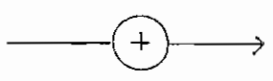
$$(G \cdot |H^T = (0 \qquad 3.16$$

> "La ventaja de la descripción matricial es que,.. es mucho más compacta que una lista de códigos vectores". (8)

Las ecuaciones dadas por 3.8 también pueden ser obtenidas a partir de la matriz $|H$ utilizando la relación $\bar{v} \cdot |H^T = (0$

Con estas ecuaciones se puede implementar el circuito codificador para un código lineal sistemático, el cual se lo representa en la figura 3.1. Donde:

Registro de Desplazamiento



Sumador módulo 2 (or exclusiva)

$P_{ij} = 1$ conexión

$P_{ij} = 0$ no conexión.

El mensaje $\bar{u} = (u_0, u_1, u_2 \ldots, u_{k-1})$ a la vez que entra al canal de información, simultáneamente es desplazado en los registros del mensaje. Inmediatamente después de que el último bit del mensaje es almacenado - los registros de paridad ya contienen almacenados los n-k, bits de con - trol que deben ser aumentados al mensaje para formar la palabra código.- El contenido de los registros de paridad es desplazado hacia el canal de información en forma serial.

> "La operación de codificación de bloque lineal consiste en dos pasos básicos. 1. La secuencia de información es colocada en el bloque del mensaje. 2. El codificador transforma cada bloque de mensaje en un bloque más largo de n bits". (9).

La complejidad del circuito codificador es proporcional a la longitud del código bloque.

3.1.1. Síndrome.-

En el receptor, el decodificador primero calcula una n-k tupla a partir de la matriz verificadora de paridad y el vector recibido que son elementos conocidos. El resultado del cálculo se lo llama síndrome del- vector recibido:

$$\bar{S} = \bar{r} \cdot IH^T \qquad 3.17$$

$$\bar{s} = (s_0, s_1, s_2 \ldots, s_{n-k}) \qquad 3.18$$

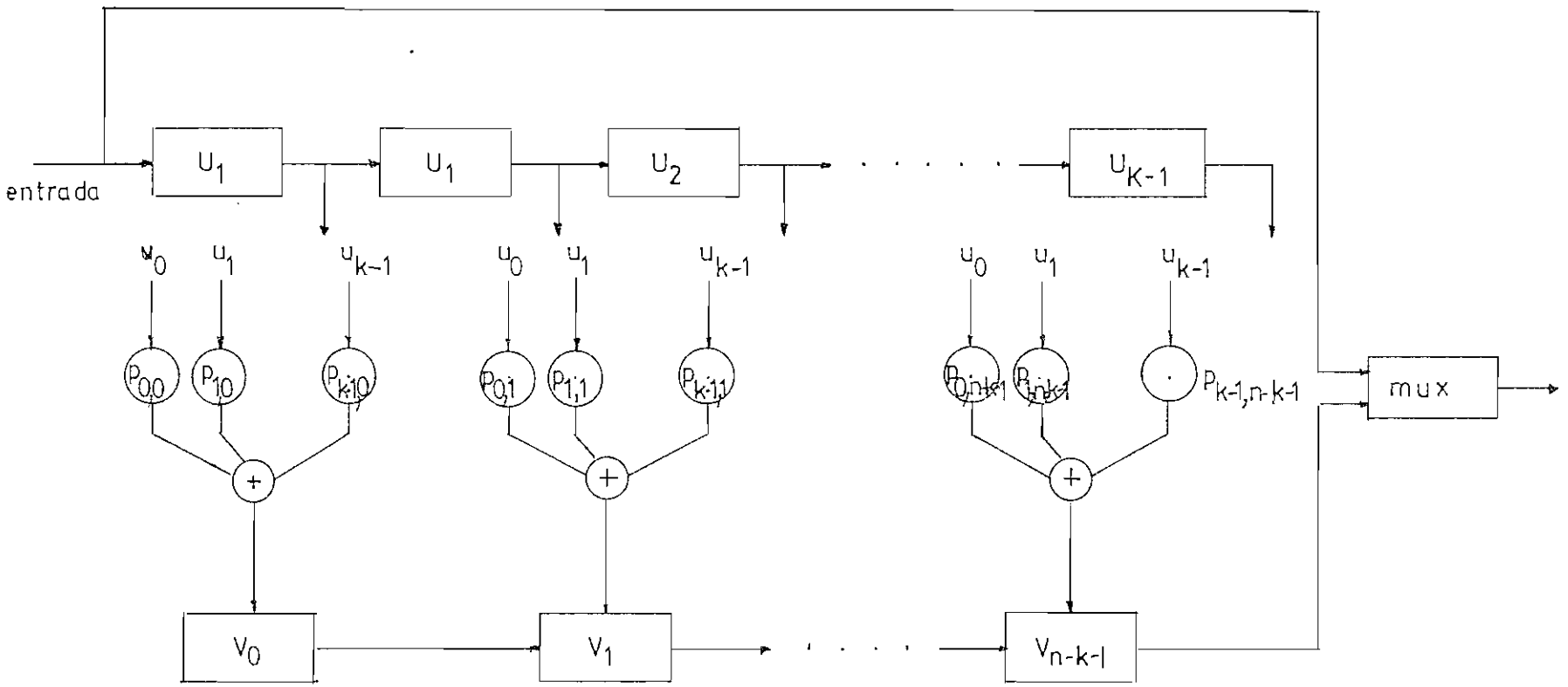


FIGURA 3.1 Circuito codificador para un código lineal sistemático (n,k)

$\bar{s} = 0$ si y solo si $\bar{r}$ es una palabra de código y $\bar{s} \neq 0$ si y solo si $\bar{r}$ no es una palabra de código, por lo que existen errores en $\bar{r}$.

En general si cada fila de H corresponde a una ecuación verificadora de paridad, que todo vector de código debe satisfacer, entonces los componentes de $\bar{s}$ son ceros cuando las ecuaciones han sido satisfechas, es decir que al evaluarlas el resultado ha sido cero.

Es posible que se produzcan errores que no puedan ser detectados - ya que puede darse el caso de que el síndrome sea igual a cero, inclusive con un vector recibido erróneo. Esto se debe a que la introducción de errores en una palabra de código la puede transformar en otra palabra - código. A este tipo de error se lo llama indetectable, y si sabemos que existen 2^{k-1} palabras código diferentes de cero entonces existirán 2^{k-1} errores indetectables. La aparición de este tipo de error dará lugar a- una decodificación errónea.

> "Puesto que no todos los patrones de error son detectables para cualquier código, la condi - ción $\bar{s} = 0$ no es ninguna garantía de una trans misión libre de error". (10).

Desarrollando la expresión 3.17 se obtiene las siguientes ecuaciones:

$$
\begin{aligned}
s_0 &= r_0 + r_{n-k} p_{0,0} + r_{n-k+1} p_{1,0} + \cdots + r_{n-1} p_{k-1,0} \\
s_1 &= r_1 + r_{n-k} p_{0,1} + r_{n-k+1} p_{1,1} + \cdots + r_{n-1} p_{k-1,1} \\
&\vdots \qquad\qquad \vdots \\
s_{n-k-1} &= r_{n-k-1} + r_{n-k} p_{0,n-k-1} + r_{n-k+1} p_{1,n-k-1} + \cdots + r_{n-1} p_{k-1,n-k-1}
\end{aligned}
\qquad 3.19
$$

Estas ecuaciones utilizadas para calcular los elementos del síndrome son similares a las ecuaciones para calcular los dígitos de control, por lo que la implementación del circuito que calculará el síndrome es similar al circuito codificador y por lo tanto su funcionamiento. Este circuito viene dado en la figura 3.2

Si $\bar{e}$ es la representación del tipo de error introducido durante la transmisión, se tienen las siguientes relaciones:

$$\bar{r} = \bar{e} + \bar{v}$$

$$\bar{e} = \bar{r} + \bar{v} \qquad \bar{e} - \bar{v} = \bar{r} \quad \rightarrow \quad \bar{e} = \bar{r} - \bar{v} \qquad 3.20$$

con $\bar{e} = (e_0, e_1, e_2 \cdots, e_{n-1})$

si $\bar{s} = \bar{r} \cdot |H^T$ y $\bar{r} = \bar{e} + \bar{v}$

entonces $\bar{s} = \bar{v} \cdot |H^T + \bar{e} \cdot |H^T$

pero como $\bar{v} \cdot |H^T = 0 \qquad \bar{s} = 0 + \bar{e} \cdot |H^T$

con lo cual $\bar{s} = \bar{e} \cdot |H^T \qquad 3.21$

> "El hecho vital acerca del síndrome es que depende sólo del tipo de error $\bar{e}$ y no de la palabra código transmitida". (11)

Utilizando la matriz $|H$ expresada como en 3.14 y multiplicando por el tipo de error e se tiene las siguientes ecuaciones:

$$\begin{aligned}
S_0 &= e_0 + e_{n-k}\, p_{0,0} + \cdots e_{n-1}\, p_{k-1,0} \\
S_1 &= e_1 + e_{n-k}\, p_{0,1} + \cdots e_{n-1}\, p_{k-1,1} \\
&\vdots \qquad\qquad\qquad\qquad \vdots \qquad\qquad 3.22 \\
S_{n-k-1} &= e_{n-k-1} + e_{n-k}\, p_{0,n-k-1} + \cdots e_{n-1}\, p_{k-1,n-k-1}
\end{aligned}$$

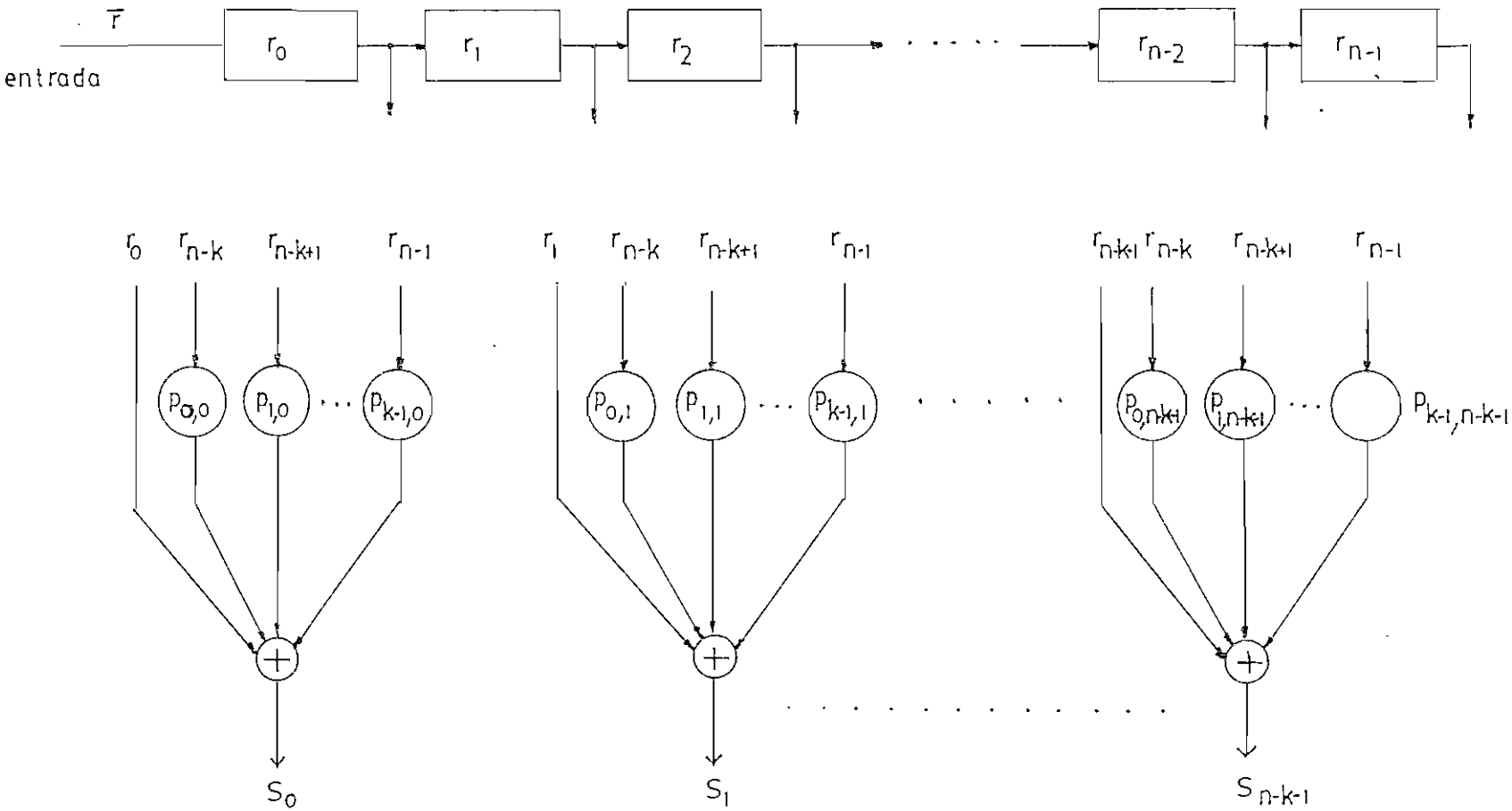


FIGURA 3.2 Circuito que calcula el síndrome

Cualquier método utilizado para resolver este sistema de ecuaciones en las cuales e_i son variables, es un método de decodificación para corregir un error. Sin embargo determinar el tipo de error $\bar{e}$ no es tan fácil, ya que éstas ecuaciones tienen 2^k soluciones.

Estas soluciones representan a todos los tipos de errores que tienen el mismo síndrome y sólo uno de esos errores es el verdadero, por lo cual el decodificador tiene que elegirlo de entre todos ellos.

Al momento de elegirlo existirá la probabilidad de no elegirlo correctamente. Para minimizar esta probabilidad que produce una decodificación errónea, primero se debe analizar el tipo de error más probable que satisfacen las ecuaciones dadas por 3.22.

Se demostrará que el tipo de error más probable es aquel que tiene el más pequeño número de dígitos diferente de cero en el vector error $\bar{e}$.

3.1.2. Capacidad para detectar y corregir un error.-

En este punto se introduce parte de una terminología básica que puede ser usada en la definición de la capacidad total de controlar errores por parte de un código bloque lineal. Esto es; el ancho de Hamming de una palabra código de C y la distancia de Hamming entre dos palabras código de C.

Si $\bar{v} = (v_0, v_1, \ldots, v_{n-1})$ es una n-tupla binaria, entonces se define el ancho de Hamming de $\bar{v}$ representado como $\bar{w}(v)$, como el número de elementos diferentes de cero que tiene $\bar{v}$.

La distancia de Hamming entre $\overline{v}$ y $\overline{w}$ representado como d $(\overline{v},\overline{w})$ se - define como el número de lugares en los cuales ambas palabras código difieren.

Si $\overline{v}$, $\overline{w}$, y $\overline{x}$ son tres n-tuplas código vector se cumple que:

$$d\ (\overline{v},\overline{w}) + d(\overline{w},\overline{x}) \geq d(\overline{v},\overline{x}) \qquad 3.23$$

La distancia de Hamming entre dos n-tuplas $\overline{v}$ y $\overline{w}$ es igual al ancho de Hamming del vector resultante de sumar $\overline{x}$ y $\overline{w}$,

$$d\ (\overline{v}.\overline{w}) = w(\overline{v}+\overline{w}) \qquad 3.24$$

Definimos la distancia mínima de un código bloque lineal C, como- la distancia mínima que se obtiene al calcular la distancia de Hamming - entre todos sus elementos.

> "Este parámetro determina la capacidad, de un - código, de detección de errores aleatorios y - corrección de errores aleatorios". (12).

Como la suma de dos palabras de código da como resultado otra palabra de código utilizando la expresión 3.24 podemos decir que la distancia de Hamming de dos vectores en C es igual al ancho de Hamming de un - tercer vector en C. De esta forma, para encontrar la distancia mínima del código, basta encontrar un vector con ancho mínimo entre todos los vectores del código diferentes de cero.

$$d_{min} = W_{min} \qquad 3.25$$

> "Tanto la detección de error como la corrección de error están basadas en las propiedades de - distancia del código". (13).

Si C es un código lineal (n,k) con una matriz verificadora de paridad H, para cada vector de código de ancho de Hamming p, existen p columnas de H tal que el vector suma de estas columnas es igual al vector cero. Consecuentemente si existen p columnas de H cuyo vector suma es el vector cero, entonces existe un código vector de ancho de Hamming p.

Esto nos indica que la distancia mínima de un código C es igual al número más pequeño de columnas de H tal que éstas sean linealmente independientes.

Si sobre un vector de código transmitido $\bar{v}$, ocurren p errores, el vector recibido $\bar{r}$ será diferente al vector transmitido $\bar{v}$ en p posiciones. Si dos palabras código cualesquiera, pueden diferir como mínimo, en d_{min} posiciones, estaremos seguros que $\bar{r}$ no es una palabra código, si $p < d_{min}$ Es por esto que un código bloque con distancia mínima igual a d_{min}, es capaz de detectar todos los tipos de errores con $(d_{min}-1)$ errores como máximo.

El código no puede detectar todos los tipos de errores con dmin, errores que ocurren simultáneamente, ya que puede existir al menos un par de palabras código que difieren en dmin lugares y un tipo de error con dmin errores puede cambiar una palabra de código en otra palabra de código. El mismo argumento indicado anteriormente se puede aplicar cuando ocurren más de dmin errores simultáneamente.

Si n errores ocurren simultáneamente, entonces existen 2^{n-1} -1 tipos de errores de los cuales 2^k - 1 son palabras código, quedando por lo tan-

--to 2^{n-k} errores que pueden ser detectad os y $2^k - 1$ errores indetectables, ya que una palabra código con la introducción de n errores puede transformarse en otra palabra código.

Una vez determinado el número errores que puede detectar un código, en base de dmin, se puede establecer el número de errores que dicho código puede corregir para lo cual definimos un número positivo entero t tal que:

$$2t+1 \leqslant d_{min} \leqslant 2t+2 \qquad 3.26$$

Si $\bar{v}$ y $\bar{r}$ son los vectores transmitidos y recibidos simultáneamente, junto con otra palabra de código $\bar{w}$ en C se tiene la siguiente relación:

$$d(\bar{v},\bar{r}) + d(\bar{w},\bar{r}) \geqslant d(\bar{v},\bar{w}) \qquad 3.27$$

Si t´errores ocurren durante la transmisión de $\bar{v}$, entonces el vector recibido $\bar{r}$ difiere del vector transmitido en t´ lugares o lo que es lo mismo $d(\bar{v},\bar{r}) = t'$. Como $\bar{v}$ y $\bar{w}$ se sabe con seguridad que son palabras código en C se tiene que:

$$d(\bar{v},\bar{w}) \geqslant d_{min} \geqslant 2t+1 \qquad 3.28$$

utilizando 3.24 y 3.28 obtenemos que:

$$d(\bar{v},\bar{r}) + d(\bar{w},\bar{r}) \geqslant 2t-1 \qquad 3.29$$

como $\quad d(\bar{v}\,\bar{r}) = t' \qquad d(\bar{w},\bar{r}) \geqslant 2t+1-t' \qquad 3.30$

Si $t' \leqslant t$ la expresión anterior se puede escribir como:

$$d(\bar{w},\bar{r}) > t$$

La desigualdad anterior indica que si ocurre un tipo de error con

t' o menos errores, el vector recibido $\bar{r}$ está más cerca al código vector transmitido $\bar{v}$ que cualquier otro código vector $\bar{w}$ en C.

Para un canal binario simétrico esto significa que P $(\bar{r}/\bar{v})$, probabilidad de recibir $\bar{r}$ dado que se transmitió $\bar{v}$, es más grande que $P(\bar{r}/\bar{w})$ para $\bar{w} \neq \bar{v}$. Basados en esto, es decir, basados en la mayor probabilidad que tiene $\bar{r}$ de decodificarse como $\bar{v}$, se tiene por lo tanto, un resultado correcto producto de la decodificación.

> "La decodificación de máxima probabilidad puede ser realizada calculando $\bar{s} = \bar{y} \cdot H^T$, y encontrando el mínimo ancho de una secuencia z que satisfaga $\bar{s} = \bar{z} \cdot H^T$ para decodificar la palabra código como $\bar{x} = \bar{z} \oplus \bar{y}$ ". (14).

Dados $\bar{v}$ y $\bar{w}$ con d $(\bar{v},\bar{w})$ = dmin y dos tipos de errores $\bar{e}_1$ y $\bar{e}_2$ los cuales satisfacen las siguientes condiciones

$$\bar{e}_1 + \bar{e}_2 = \bar{v} + \bar{w}$$

$\bar{e}_1$ y $\bar{e}_2$ No tienen componentes diferentes de cero en lugares comunes.

entonces se cumple que:

$$W(\bar{e}_1) + W(\bar{e}_2) = W(\bar{v}+\bar{w}) \qquad 3.31$$

suponiendo que $\bar{r} = \bar{v} + \bar{e}_1$, entonces $\bar{r} + \bar{v} = \bar{e}_1$,

$$W(\bar{e}_1) + W(\bar{e}_2) = d(\bar{v},\bar{w}) = d_{min} \qquad 3.32$$

$$W(\bar{e}_1) + W(\bar{e}_2) = d(\bar{v},\bar{r}) + d(\bar{w},\bar{r}) \qquad 3.33$$

si $\bar{e}_1$, contiene más de t errores se cumple que w $(e_1) > t$ donde:

$2t+1 \leqslant dmin \leqslant 2t+2$

de 3.31 nos queda la siguiente expresión

$$W(\bar{e}_1) \leqslant t + 1$$

combinando 3.32 y 3.33, partiendo de que $w(\bar{e}_1) > t$ y $w(\bar{e}_2) \leqslant t+1$ se obtiene lo siguiente:

$$\begin{aligned} d(\bar{v};\bar{r}) &= W(\bar{e}_1) > t \\ d(\bar{w},\bar{r}) &= W(\bar{e}_2) \leqslant t+1 \\ d(\bar{v},\bar{r}) &\geqslant d(\bar{w},\bar{r}) \end{aligned} \qquad 3.34$$

Esta última expresión nos indica que existe un tipo de error con ℓ errores ($\ell > t$) con el cual resulta un vector recibido que está más cercano a una palabra código que no es la transmitida, dando como resultado una decodificación incorrecta.

Un código bloque que es capaz de corregir t errores aleatorios, usualmente es capaz de corregir tipos de errores con t + 1 errores. En general un código lineal (n,k) corrector de t errores es capaz de corregir un total de 2^{n-k} tipos de errores diferentes, en los cuales se incluyen aquellos tipos de errores con t o menos errores.

> "Un código bloque con distancia mínima dmin garantiza corregir todos los tipos de errores con t = (dmin - 1) / 2) o menos errores" (15).

El parámetro t es llamado también capacidad de corrección de errores aleatorios y representa el más grande entero no mayor que (dmin-1) /2.

Si este código es usado para la corrección de errores en un canal-

binario simétrico con probabilidad de transición p, la probabilidad de que el decodificador realice una decodificación errores viene dado por la siguiente expresión:

$$P(E) \leqslant \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i} \qquad 3.35$$

"La capacidad corregir errores aleatorios y detectar errores aleatorios de un código - bloque está determinada por la distancia - mínima del bloque". (16).

3.1.3. Arreglo Standard.-

Un esquema de decodificación el cual nos permite obtener el código vector correcto, se basa en la construcción de un arreglo de filas y columnas, cuya primera fila contiene todas las 2^k palabras código posibles y cuya primera columna contiene todos los tipos de errores que puedan ser corregidos.

El primer elemento común para la primera fila y columna es el código vector cero.

$$\begin{array}{cccccc}
\bar{v}_1 = 0 & \bar{v}_2 & \bar{v}_3 & \cdots & \bar{v}_i & \cdots \bar{v}_{2^k} \\
\bar{e}_2 & \bar{e}_2+\bar{v}_2 & \bar{e}_2+\bar{v}_3 & \cdots & \bar{e}_2+\bar{v}_i & \cdots \bar{e}_2+\bar{v}_{2^k} \\
\bar{e}_3 & \bar{e}_3+\bar{v}_2 & \bar{e}_3+\bar{v}_3 & \cdots & \bar{e}_3+\bar{v}_i & \cdots \bar{e}_3+\bar{v}_{2^k} \\
\vdots & \vdots & \vdots & & \vdots & \vdots \\
\bar{e}_\ell & \bar{e}_\ell+\bar{v}_2 & \bar{e}_\ell+\bar{v}_3 & \cdots & \bar{e}_\ell+\bar{v}_i & \cdots \bar{e}_\ell+\bar{v}_{2^k} \\
\vdots & \vdots & \vdots & & \vdots & \vdots \\
\bar{e}_{2^{n-k}} & \bar{e}_{2^{n-k}}+\bar{v}_2 & \bar{e}_{2^{n-k}}+\bar{v}_3 & \cdots & \bar{e}_{2^{n-k}}+\bar{v}_i & \cdots \bar{e}_{2^{n-k}}+\bar{v}_{2^k}
\end{array} \qquad 3.36$$

El arreglo se termina de formar calculando los demás elementos - por medio de la expresión $\bar{e}_\ell + \bar{v}_i$ con $1 \leqslant i \leqslant 2^k$ y cuyo resultado se coloca en la posición dada por la ℓ fila y la i columna. Cuando se ha

completado todas las filas y columnas se obtiene lo que se llama el arreglo standar del código lineal C. Cada columna se lo representa como Dj tal que $\overline{v}_j$ este contenido en esta columna.

El arreglo standar se caracteriza porque ninguno de los dos elementos, cualesquiera que pertenecen a una misma fila son iguales y cualesquiera de los n elementos del arreglo aparece en sólo una fila y en una sola columna.

Los 2^{n-k} filas que contiene el arreglo standar se llaman cosets del código C, y al primer elemento de cada fila se lo llama líder del coset.

> "El conjunto de las secuencias de errores que aparecen en la tabla decodificadora, es precisamente el conjunto de errores que pueden ser corregidos". (17).

El proceso de decodificación puede ser simplificado si se usa la tabla de decodificación formada. Cuando una palabra código $\overline{v}_j$ es transmitida en un canal ruidoso, en el receptor se tendrá un vector $\overline{r}$ que al ubicarse en la columna Dj nos indicará que el tipo de error que se ha producido es igual al líder del coset.

> "El procedimiento más simple, consiste conceptualmente en comparar la palabra codificada recibida con una tabla almacenada de palabras codificadas, seleccionando la que más probablemente haya sido transmitida". (18).

Una decodificación correcta se da si y solo si el tipo de error in

-- troducido por el canal ruidoso es igual a algún líder del coset, por - esta razón los 2^{n-k} líderes de los coset son llamados tipos de error co- rregibles. Así cualquier código bloque lineal (n,k) es capáz de corregir 2^{n-k} tipos de errores diferentes.

> "Si esta técnica es posible, es decir, si es- posible precalcular y almacenar los coset - cargadores correspondientes a cada uno de los ... síndromes, el resultante algoritmo codi- ficador es rápidamente realizado". (19).

Para un canal binario simétrico el tipo de error más probable es - aquel que tiene menor ancho de Hamming, y como se puede apreciar en el - arreglo standar, los coset cargadores son generalmente todos los tipos - de errores con ancho mínimo.

> "Al calcular la probabilidad de error para un código bloque, nosotros notamos que una trans misión correcta ocurre si y solo si el tipo- de error es un coset". (20).

Ya estamos en capacidad de poder deducir que los parámetros importantes para la construcción de códigos bloque óptimos son la probabilidad de decodificación errónea, la longitud del código bloque y la eficiencia de transmisión.

Como resultado de la expresión 3.21 podemos decir que el síndrome- de cualquier elemento del coset es igual al síndrome del líder del coset o del tipo de error producido.

> "Todas las 2^k n-tuplas de un coset tienen el mismo síndrome, los síndromes para diferentes coset son diferentes". (21).

Con este antecedente es claro que se puede formar una tabla de decodificación en la cual a un síndrome le corresponda uno y sólo un tipo de error.. Con esta conclusión el proceso de decodificación del vector recibido $\bar{r}$ se lo puede resumir en tres pasos:

1.- Calcular el síndrome de $\bar{r}$ con la siguiente expresión

$$S = \bar{r} \cdot H^T$$

2.- Localizar el coset cargador tipo de error $\bar{e}$ cuyo síndrome es igual al calculado anteriormente.

3.- Decodificar al vector recibido $\bar{r}$ como el vector de código resultante de la siguiente suma:

$$\bar{v} = \bar{r} + \bar{e}$$

El proceso de decodificación descrito anteriormente, es llamado decodificación por síndrome, el cual puede ser aplicado a cualquier código lineal (n,k). Los resultados de este tipo de decodificación se traducen a una mínima probabilidad de error y un retrazo mínimo de tiempo en la decodificación. Sin embargo para códigos con grandes valores de n y k, la implementación del codificador ya no resulta práctica porque se aumenta la capacidad de memoria requerida y la complejidad de los circuitos lógicos.

Para obtener una tabla de decodificación que nos permita relacionar el síndrome del vector recibido con el tipo de error, se sugiere la siguiente tabla de verdad, la cual contiene n diferentes funciones.

$$\begin{aligned} e_0 &= f_0 (s_0, s_1 \cdots, s_{n-k-1}) \\ e_1 &= f_1 (s_0, s_1 \cdots, s_{n-k-1}) \\ &\vdots \\ e_{n-1} &= f_{n-1} (s_0, s_1 \cdots, s_{n-k-1}) \end{aligned} \qquad 3.37$$

donde s_0 , $s_1 \ldots s_{n-k-1}$ son los elementos del síndrome y $e_1, e_0 \ldots e_{n-1}$ son los elementos (dígitos) estimados del tipo de error. Su implementación resulta en un circuito combinacional cuyas entradas son los elementos del síndrome y como salida tiene a los elementos del tipo de error asociado al síndrome.

De esta forma el diagrama de bloques de un sistema decodificador para códigos bloques lineales se presenta en la fig. 3.3.

3.1.4. Probabilidad de no detectar un error en un canal binario simétrico.-

Si C es un código lineal (n,k) y si además definimos a Ai como el número de palabras código con ancho i en C, a los números Ao, A1 , A2 ,.. An se los llama distribución de ancho del código C.

La probabilidad de no detectar un error en un canal binario simétrico viene dada por

$$P_u(E) = \sum_{i=1}^{n} A_i \, p^i (1-p)^{n-i} \qquad 3.38$$

Pu (E) = probabilidad de no detectar un error.

p = probabilidad de transición en un canal binario simétrico.

Se supone que cualquier dígito se convierte en el otro durante la transmisión con la misma probabilidad $p \leq \frac{1}{2}$ y como el canal se a supuesto ausente de memoria, cada dígito de una secuencia tiene la misma probabilidad de ser recibido en forma errónea.

Si $(A_0, A_1, \ldots A_n)$ y $(B_0, B_1, \ldots \ldots B_n)$ son las distribuciones

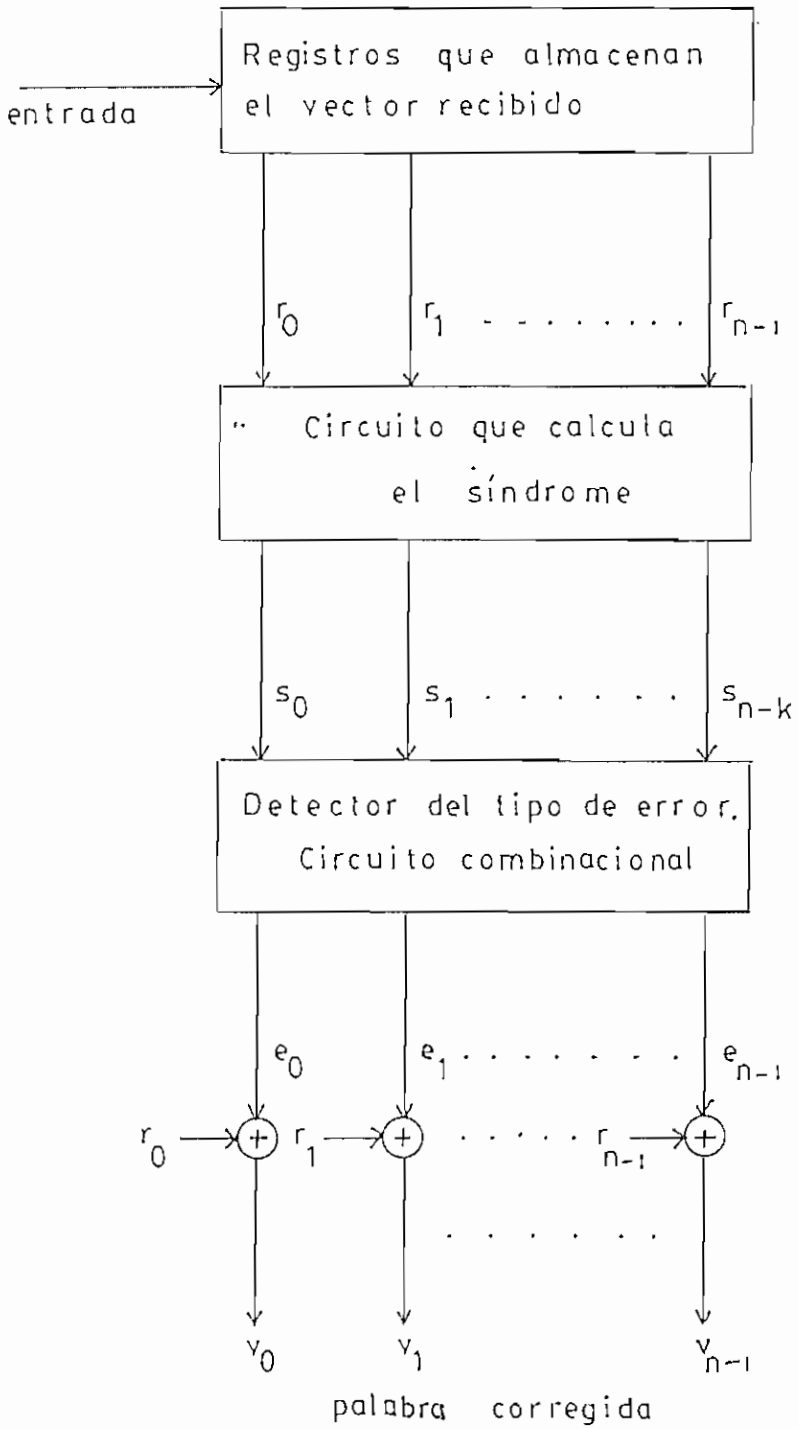


FIGURA 3.3 Decodificador para un código bloque lineal (n,k)

de ancho del código C y su dual respectivamente, una expresión para más fácil de manejar es la siguiente.

$$P_u(E) = (1-p)^n \left[A\left(\frac{p}{1-p}\right) - 1 \right] \qquad 3.39$$

$$P_u(E) = 2^{-(n-k)} B(1-2p)\ (1-p)^n \qquad 3.40$$

con $A(z) = A_0 + A_1 z \cdots + A_n z^n$ y $B(z) = B_0 + B_1 z \cdots + B_n z^n$

El cálculo de la probabilidad de no detectar un error en función del ancho de distribución del código C o de su dual Cd, dependerá mucho de los valores que tengan n y k . Por ejemplo si n-k es más pequeño que k es más fácil usar la expresión. 3.4 . siempre y cuando k no sea demasiado grande.

> "Excepto para algunos códigos lineales cortos y una pequeña clase de códigos lineales, la distribución de ancho para muchos códigos son desconocidas. Consecuentemente, es muy dificultoso pero no imposible calcular su probabilidad de no detectar error". (22).

Aunque es difícil calcular la probabilidad de no detectar un error para un código lineal con altos valores de n y k, es completamente fácil deducir un límite mayor para la probabilidad de no detectar un error.

Existen $2^{k(n-k)}$ tipos diferentes de matrices G que tienen la forma sistemática dada por 3.5. En base al conjunto de los códigos generados por estas $2^{k(n-k)}$ matrices se puede demostrar que el límite máximo de la probabilidad promedio de no detectar un error es:

$$\overline{P}_u(E) = 2^{-(n-k)} \sum_{i=1}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

$$\overline{P}_u(E) = 2^{-(n-k)} \left[1 - (1-p)^n \right] \qquad 3.41$$

Si $(1 - (1-p)^n) \leq 1$ se tiene que

$$IP(\bar{E}) \leq 2^{-(n-k)} \qquad 3.42$$

lo cual nos indica que existen códigos lineales con probabilidad de no detectar un error, representada por $IP(\bar{E})$, menor que $2^{-(n-k)}$
En otras palabras $IP_u(\bar{E})$ decrece exponencialmente con el número de dígitos verificadores de paridad que son (n-k).

3.2. Códigos Cíclicos.-

La distancia de Hamming ya analizada es un ejemplo de una técnica - utilizada en la determinación del tamaño del código que se ha de utilizar para el control de errores. Además se puntualizó que cualquier código (n, k) que satisfaga la desigualdad de Hamming, automáticamente proporciona- al código la capacidad de corregir errores simples.

Para avanzar en las capacidades de corrección de errores de orden- superior en forma sistemática, es necesario introducir una estructura adicional en la formulación del código. Aunque esta necesidad limita la elección de los códigos posibles, por lo menos de una manera de seleccionar los códigos. Por lo tanto, a continuación se tratará sobre una clase de códigos denominados cíclicos en los cuales los vectores que lo forman se obtienen por desplazamientos.

Las ventajas de estos códigos es que el proceso de codificación puede ser fácilmente implementado y su estructura matemática ha sido extensamente estudiada.

En el tratamiento de los códigos cíclicos se hace necesaria la definición de lo que es ideal.

Ideal se denomina a un subconjunto de elementos de un anillo con las siguientes propiedades.

- I es un subgrupo del grupo aditivo R.
- Para cualquier elemento a de I y cualquier r elemento de R a. r o r. a están en I.

Si los elementos del vector de código $\overline{v}=(v_0, v_1, \ldots v_{n-1})$ son cíclicamente desplazados un lugar hacia la derecha se tiene el siguiente vector $\overline{V}^{(1)}$ de n elementos.

$$\overline{V}^{(1)}=(v_{n-1}, v_0, v_1 \ldots\ldots, v_{n-2}) \qquad 3.43$$

si dicho desplazamiento cíclico se lo hace i-veces, el vector así formado sería

$$\overline{V}^{(i)}=(v_{n-i}, v_{n-i-1} \ldots, v_{n-1}, v_0, v_1 \ldots\ldots, v_{n-i-1})$$

El desplazar los elementos i-veces hacia la derecha produce el mismo efecto que desplazarlos n-i veces hacia la izquierda.

Si para cada desplazamiento cíclico de un vector de código, en un código C (n,k) se obtiene un vector resultante, el cual también pertenece a C, entonces dicho código es llamado cíclico.

> "Un código cíclico sobre GF(q) es un código-lineal con una propiedad especial por la cual cualquier desplazamiento de una palabra código es otra palabra código". (23).

Con los elementos del código vector $\bar{v}$ se puede formar un polinomio, de tal manera que, a cada vector de código le corresponde un polinomio de grado menor o igual a n-1. Al polinomio v (x), así formado se lo llama - polinomio de código de $\bar{v}$, y la correspondencia entre $\bar{v}$ y v(x) es de uno-a uno.

Una propiedad importante del desplazamiento cíclico de estos códigos es que $v^{(i)}(x)$ es igual al residuo que se obtiene de dividir para $x^i v(x)$ para x^n+1.

> "Los códigos cíclicos pueden describirse en forma polinomial, propiedad que es extremadamente útil para su análisis e implementación". (24).

Si suponemos la existencia de dos polinomios de grado mínimo que son:

$$g(x) = g_0 + g_1 x \cdots g_{r-1} x^{r-1} + x^r$$
$$g'(x) = g'_0 + g'_1 x \quad g'_{r-1} x^{r-1} + x^r$$

Como pertenecen a un código lineal, su suma debe dar otro código polinomial dado por

$$g(x) + g'(x) = (g_0 + g'_0) + (g_1 + g'_1) x + \cdots (g_{r-1} + g'_{r-1}) x^{r-1} + 0 x^r$$

el cual tiene un grado menor a r, por lo que el código polinomial de un código cíclico de grado mínimo y diferente de cero, es un polinomio único.

Además, se puede probar que el coeficiente g_0 debe ser igual a uno. Si esto no se cumpliría se tuviera como resultado del desplazamiento cíclico de este polinomio, otro polinomio de grado menor a r, lo cual contradice lo demostrado, quedando de esta manera el polinomio mínimo g(x) con la siguiente forma.

$$g(x) = 1 + g_1 x + \cdots g_{r-1} x^{r-1} + x^r \qquad 3.44$$

Si desplazamos cíclicamente n-r-1 veces el polinomio g(x) se tiene como resultado los siguientes polinomios:

$$x\,g(x) = g^{(1)}(x) \quad , \quad x^2 g(x) = g^{(2)}(x) \quad , \cdots \quad x^{n-r-1} g(x) = g^{(n-r-1)}(x)$$

los cuales son también códigos polinomiales en C y cualquier elemento en C puede ser obtenido por medio de una combinación lineal de estos polinomios.

$$v(x) = u_0 g(x) + u_1 x\,g(x) + \cdots u_{n-r-1} x^{n-r-1} g(x)$$

$$v(x) = (u_0 + u_1 x + \cdots u_{n-r-1} x^{n-r-1})\; g(x) \qquad 3.45$$

$u_i = 0$ ó 1 $\qquad\qquad$ $v(x) = u(x)\,g(x) \qquad 3.46$

Así tenemos que, si el polinomio de grado mínimo g(x), pertenèce a un código cíclico (n,k), cualquier polinomio binario u(x) de grado menor o igual a n-1 pertenece al código si y sólo si es múltiplo de g(x).

Si dividimos v (x) para g (x) se tiene que el residuo b (x) es igual a:

$$b(x) = a(x)g(x) + v(x) \qquad 3.47$$

en donde a(x) b(x) y v(x) son polinomios de código, por lo que su suma también es un polinomio de código, de esta forma si b (x) fuera diferente de cero entonces por ser residuo tendría un grado menor a g (x), lo cual no puede ser. Por lo que b(x) = 0.

El número de polinomios binarios de grado menor o igual a n-1 que son múltiplos de g(x) es igual a 2^{n-r} y son llamados polinomios de código del código cíclico C (n,k).

Si sabemos que existen 2^k palabras código en un código lineal (n,k) entonces debe cumplirse que

$$\text{si} \quad 2^k = 2^{n-r} \quad \text{ó} \quad k = n-r \quad \text{entonces} \quad r = n-k \qquad 3.48$$

con lo que resulta que el grado de g(x) es n-k. De esta manera el polinomio de grado mínimo con las características ya analizadas tiene la siguiente forma:

$$g(x) = 1 + g_1 x + g_2 x^2 + \cdots g_{n-k-1} x^{n-k-1} + x^{n-k} \qquad 3.49$$

Retomando la expresión 3.46 se tiene que

$$v(x) = (u_0 + u_1 x + \cdots u_{k-1} x^{k-1}) \cdot g(x) \qquad 3.50$$

en la cual, los coeficientes de $u(x) = (u_0, u_1, \ldots, u_{k-1})$ son los k dígitos de información a ser codificados.

La codificación puede ser realizada multiplicando el mensaje por g(x). Con lo cual el código cíclico (n,k) está completamente definido por el polinomio de grado mínimo g(x).

> "Cualquier código cíclico sobre GF (q) con L dígitos de información y longitud de bloque N es generado por un polinomio de grado N-L sobre GF (q)" (25).

Este polinomio g(x) es llamado polinomio generador del código cíclico (n,k), y determina las características del código.

Si realizamos la división $g(x)\, x^k$ para $x^n + 1$

$$g(x)x^k = (x^n+1) + g^{(k)}(x) = (x^n+1) + a(x) \cdot g(x)$$

$$(x^n+1) = g(x)(x^k+a(x)) \qquad 3.51$$

De esta última expresión nos podemos dar cuenta que el polinomio generador g(x) de un código cíclico (n,k) es factor de $x^n + 1$.

> "Las palabras codificadas no son, sin embargo, sistemáticas en el sentido de que los primeros k bits no representan a los bits de información y los restantes r=n-k a los bits de paridad". (26).

Cualquier factor de $x^n + 1$ con grado n-k genera un código cíclico (n,k). Para grandes valores de n, $x^n + 1$ puede tener muchos factores de grado n-k, algunos de estos factores generan códigos buenos; saber cuáles son, puede ser muy complicado.

La forma de generar códigos cíclicos sistemáticos se basa en lo siguiente.

Sea u(x) el mensaje a ser codificado.

$$x^{n-k}\,u(x) = u_0 x^{n-k} + u_1 x^{n-k-1} + \cdots u_{k-1} x^{n-1}$$

dividiendo para g(x)

$$x^{n-k}u(x) = a(x)\,g(x) + b(x)$$

$$b(x) + x^{n-k}u(x) = a(x)g(x) \qquad 3.53$$

como b (x) es el residuo de la división, tiene la siguiente forma:

$$b(x) = b_0 + b_1 x \cdots\cdots + b_{n-k-1}x^{n-k-1}$$

$b(x)+x^{n-k}u(x)$ es una palabra de código ya que es múltiplo de g(x) y es igual a: $b_0 + b_1x + \cdots b_{n-k-1}x^{n-k-1} + u_0x^{n-k} + u_1x^{n-k+1} + \cdots\cdots u_{k-1}x^{n-1}$

este polinomio equivale a un vector de código cuyos coeficientes son:

$$(b_0, b_1 \cdots\cdots, b_{n-k-1}, u_0, u_1 \cdots, u_{k-1}) \qquad 3.54$$

De esta forma, un vector de código consiste de k dígitos de información y n-k dígitos de control, los cuales son los coeficientes del residuo que resulta de dividir $x^{n-k}\, u(x)$ para $g(x)$.

"Cada código lineal es equivalente a un código sistemático". (27).

Ya nos podemos dar cuenta como poder codificar una información a transmitir. El proceso de codificación para códigos cíclicos tiene los siguientes pasos:

- Multiplicar el mensaje $u(x)$ por x^{n-k}
- Obtener el residuo que resulta de dividir $u(x)x^{n-k}$ para el polinomio generador $g(x)$.
- Combinar el residuo $b(x)$ y $x^{n-k}u(x)$ para obtener la palabra codificada dada por $b(x)+x^{n-k}u(x)$

3.2.1. Matriz Generadora y Verificadora de Paridad.-

A partir de los polinomios $g(x), x\,g(x), \ldots, x^{k-1}g(x)$ nosotros podemos formar una matriz $k \times n$ cuyos elementos son los coeficientes de estos polinomios. A esta matriz se la conoce como matriz generadora.

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & & 0 \\ 0 & 0 & g_0 & \cdots & \cdots & g_{n-k} & \cdots & & 0 \\ \vdots & \vdots & \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \cdots & g_0 & g_1 & \cdots & & g_{n-k} \end{bmatrix} \qquad 3.55$$

Si g(x) es factor de $x^n + 1$ se puede escribir que

$$x^n + 1 = g(x)\, h(x) \qquad 3.56$$

donde h(x) tiene la siguiente forma:

$$h(x) = h_0 + h_1 x + h_2 x^2 + \ldots . h_k x^k$$

con $h_0 = h_k = 1$. Con este polinomio h(x) nosotros podremos obtener - la correspondiente matriz verificadora de paridad de código.

$$v(x) = a(x)g(x)$$

$$v(x)h(x) = a(x)g(x)h(x) = a(x)(x^n+1)$$

$$v(x)h(x) = a(x)x^n + a(x)$$

Se puede ver que el grado de a(x) es menor o igual a k-1 y que las potencias $x^k, x^{k+1}, \ldots x^{n-1}$ no aparecen en $a(x) + x^n a(x)$. Si desplazamos el - producto v(x) h (x) al lado izquierdo, los coeficientes de $x^k, x^{k+1}, \ldots, x^{n-1}$ son iguales a cero, obteniéndose de esta manera n-k ecuaciones o igualdades dadas por la siguiente expresión:

$$\sum_{i=0}^{k} h_i\, v_{n-i-j} = 0 \qquad i \leq j \leq n-k \qquad 3.57$$

utilizando el recíproco de h(x) se obtiene que

$$x^k h(x^{-1}) = h_k + h_{k-1} x + \ldots . h_0 x^k$$

esta expresión también es factor de $x^n + 1$, y el polinomio $x^k h(x^{-1})$ genera un código cíclico (n,n-k) con una matriz generadora $(n-k) \times n$ la cual se representa por $\|H\|$.

$$\|H\| = \begin{bmatrix} h_k & h_{k-1} & h_{k-2} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & \cdots & & 0 \\ 0 & 0 & h_k & \cdots & \cdots & h_0 & \cdots & & 0 \\ \vdots & \vdots & \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \cdots & h_k & h_{k-1} & h_{k-2} & \cdots & h_0 \end{bmatrix} \qquad 3.58$$

Las n-k igualdades de 3.57 de cualquier vector de código $\bar{v}$ en C - son ortogonales a cada fila de $\mathbb{H}$, con lo que $\mathbb{H}$ es la matriz verificadora de paridad de código cíclico y por lo tanto la matriz $\mathbb{H}$ genera el código dual C_d de C. A h(x) se lo llama polinomio de paridad de C.

"Un código cíclico está también, únicamente especificado por el polinomio de paridad". (28).

Dado un código cíclico C (n,k) generado por g(x), el código dual C_d también es un código cíclico generado por el polinomio $x^k h(x^{-1})$ donde $h(x) = (x^n + 1) / g(x)$

Si dividimos x^{n-k+i} para g(x) con i = 0,1,2, ... k-1 se obtiene que;

$$x^{n-k+i} = a_i(x)\, g(x) + b_i(x)$$

con $$b_i(x) = b_{i,0} + b_{i,1} x \cdots\cdots + b_{i,n-k-1} x^{n-k-1}$$

Si $b_i(x) + x^{n-k+i}$ para i = 0,1,2 .... k-1 son múltiplos de g(x), entonces son polinomios de código y el arreglo de estos k polinomios son las filas de una matriz $k \times n$ que tiene la siguiente forma:

$$\mathbb{G} = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} \cdots\cdots b_{0,n-k-1} & 1 & 0 & 0 \cdots 0 \\ b_{1,0} & b_{1,1} & b_{1,2} \cdots\cdots b_{1,n-k-1} & 0 & 1 & 0 \cdots 0 \\ b_{2,0} & b_{2,1} & b_{2,2} \cdots\cdots b_{2,n-k-1} & 0 & 0 & 1 \cdots 0 \\ \vdots & & \vdots & & & \vdots \\ b_{k-1,0} & b_{k-1,1} & b_{k-1,2} \cdots b_{k-1,n-k-1} & 0 & 0 & 0 \cdots 1 \end{bmatrix} \qquad 3.59$$

Esta es la matriz generadora de C de forma sistemática y la correspondiente matriz verificadora de paridad de este código es la siguiente:

$$\mathbb{H} = \begin{bmatrix} 1 & 0 & 0 & b_{0,0} & b_{1,0} \cdots\cdots b_{k-1,0} \\ 0 & 1 & 0 & b_{0,1} & b_{1,1} \cdots\cdots b_{k-1,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & b_{0,n-k-1} & b_{1,n-k-1} \cdots\cdots b_{k-1,n-k-1} \end{bmatrix} \qquad 3.60$$

"La matriz generadora de un código cíclico puede ser representada en términos del grado N-L del polinomio generador g(d), y la matriz verificadora puede ser representada en términos del grado L del polinomio h(d) conocido como polinomio verificador". (29).

3.2.2. Codificación de Códigos Cíclicos.-

Ya se pudo ver anteriormente que los códigos cíclicos se prestan fácilmente para ser generados directamente a partir del polinomio generador g(x).;

Los tres pasos matemáticos necesarios para realizar la codificación pueden ser realizados por un circuito, el cual tiene n-k registros de desplazamiento con conexiones de realimentación basados en el polinomio generador g(x). Tal circuito tiene la siguiente forma.

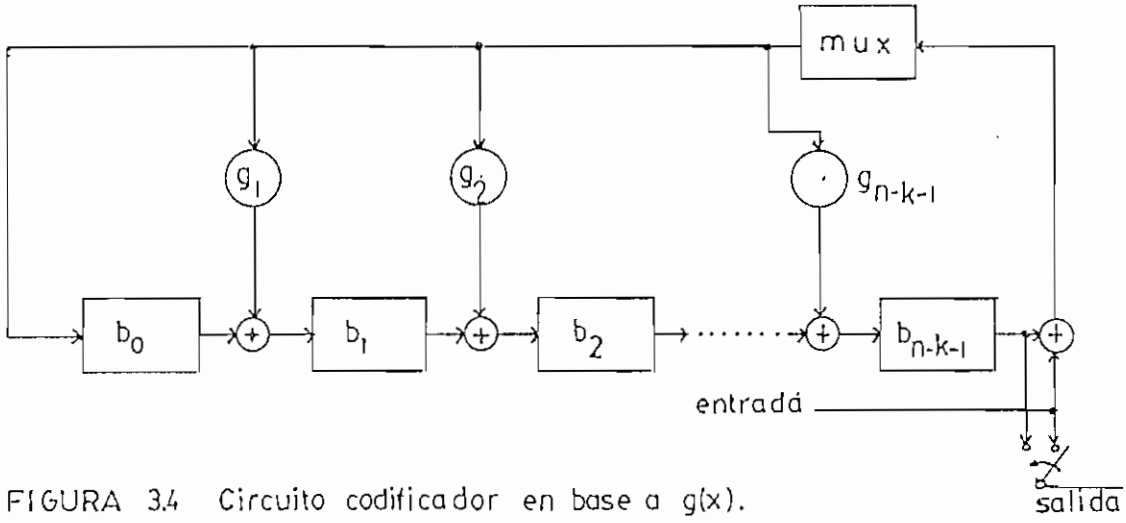


FIGURA 3.4 Circuito codificador en base a g(x).

"Para códigos binarios cíclicos el circuito es particularmente simple, involucra exactamente registros de desplazamiento binario y sumadores módulo dos". (30).

Al mismo tiempo que la palabra a codificar entra al circuito, también va directamente al canal de transmisión. El desplazamiento de u(x) hacia el circuito es equivalente a pre-multiplicar u(x) por x^{n-k} , de esta manera, cuando la totalidad del mensaje ha entrado al circuito, los n-k dígitos de control se hallan almacenados en los registros.

A continuación se desconecta la realimentación para desplazar serialmente estos dígitos de control, al canal de transmisión, para de esta forma crear el código vector del código cíclico.

La codificación también puede ser realizado, creando un circuito codificador en base al polinomio de paridad $h(x) = h_0 + h_1 x + \cdots h_k x^k$

Como h(x) es un polinomio generador, entonces h(x) y por lo tanto las desigualdades dadas por 3.57, pueden ser expresadas de la siguiente forma

$$v_{n-k-j} = \sum_{i=0}^{k-1} h_i \; v_{n-i-j} \qquad 3.61$$

esta expresión nos da una regla para determinar los n-k dígitos de control $v_0, v_1, \ldots, v_{n-k-1}$ que pertenecen al código vector.

El circuito codificador, que se basa en la expresión 3.61 tiene el siguiente diagrama de bloques.

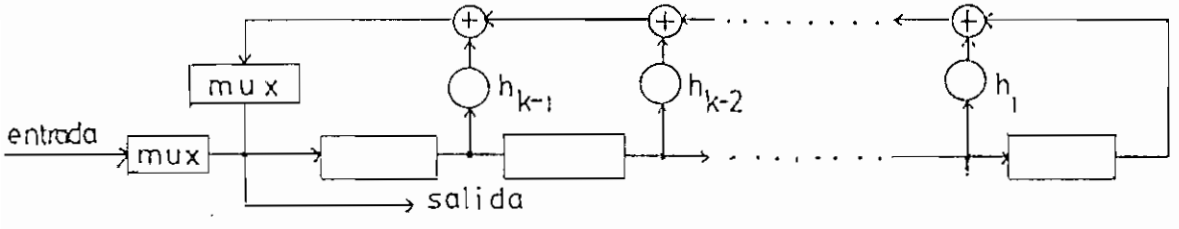


FIGURA 3.5 Circuito codificador en base a h(x)

La diferencia en las dos formas de implementar los circuitos codificadores, es que el primero utiliza (n-k) registros y este último emplea k registros de desplazamiento. De esta forma el más conveniente será aquel que menos registros utilice y eso dependerá de los valores de n y k que tenga el código.

> "Un problema, el cual es evidente cuando muchos códigos correctores o detectores de errores son diseñados, es la dificultad para implementarlos" (31)

3.2.3. Decodificación de Códigos Cíclicos.-

Como sabemos, el vector recibido r en un canal ruidoso no es necesariamente igual al vector transmitido. Así el primer paso en la codificación es el cálculo del síndrome utilizando la siguiente expresión

$$\bar{s} = \bar{r} \cdot H^T \qquad 3.62$$

Para un código cíclico el síndrome puede ser obtenido de la siguiente manera. Si r(x) es el polinomio del vector recibido con grado menor o igual a n-1, se tiene que

$$r(x) = a(x)\ g(x) + s(x)$$

Esta expresión nos indica que el proceso de decodificación en el receptor se lleva a cabo dividiendo el polinomio recibido r(x) para g(x), obteniendo de esta manera s(x) que es un polinomio de grado menor o igual a n-k-1 y sus n-k coeficientes conforman el síndrome que deseábamos calcular.

"El síndrome es igual al residuo que resulta de dividir el tipo de error para el polinomio generador, por lo que el síndrome contiene información sobre el tipo de error". (32).

Un circuito que realiza la división de r(k) para g(x) viene dado en la siguiente figura:

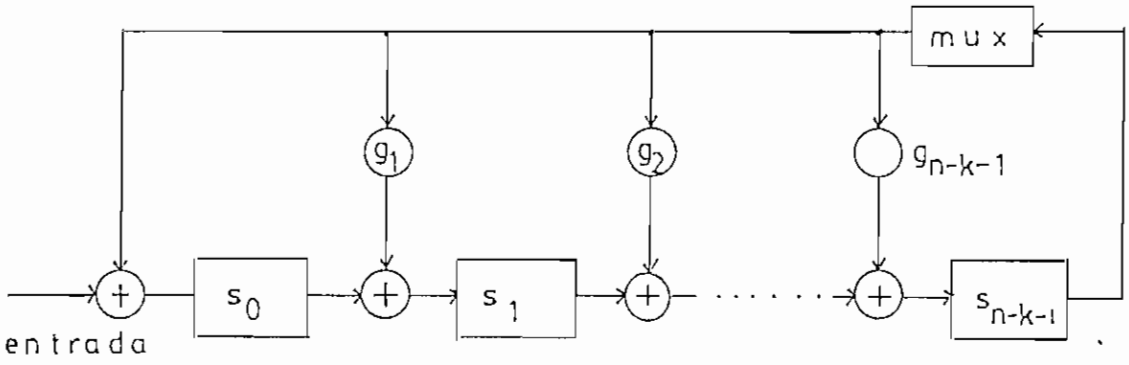


FIGURA 3.6 Circuito que calcula el síndrome

Si s (x) es el síndrome de un vector recibido r (x), el residuo $s^{(1)}(x)$ que resulta de dividir xs (x) para el polinomio generador g(x), es el síndrome de $r^{(1)}$ (x) el cual es el resultado de desplazar cíclicamente una vez el vector r (x).

$$x\,r(x) = r\,(x^n+1) + r^{(1)}(x)$$

$$r^{(1)}(x) = r_{n-1}(x^n+1) + x\,r(x)$$

$$\frac{r^{(1)}(x)}{g(x)} = \frac{r_{n-1}(x^n+1) + x\,r(x)}{g(x)}$$

$$x^n+1 = g(x)\,h(x) \qquad 3.63$$

$$c(x)\,g(x) + p(x) = r_{n-1}\,h(x)g(x) + x[a(x)g(x)+s(x)]$$

p(x) síndrome de $r^{(1)}(x)$

$$x\,s(x) = [\,c(x) + r_{n-1}\,h(x) + x\,a(x)\,]\,g(x) + p(x) \qquad 3.64$$

luego p(x) es el residuo de dividir xs(x) para g(x) con lo que p(x) = $s^{(i)}(x)$.

De forma general el residuo $s^{(i)}(x)$, que resulta de dividir $x^i s(x)$ para el polinomio g(x), es el síndrome de $r^i(x)$ que es el vector r(x) desplazado i-veces.

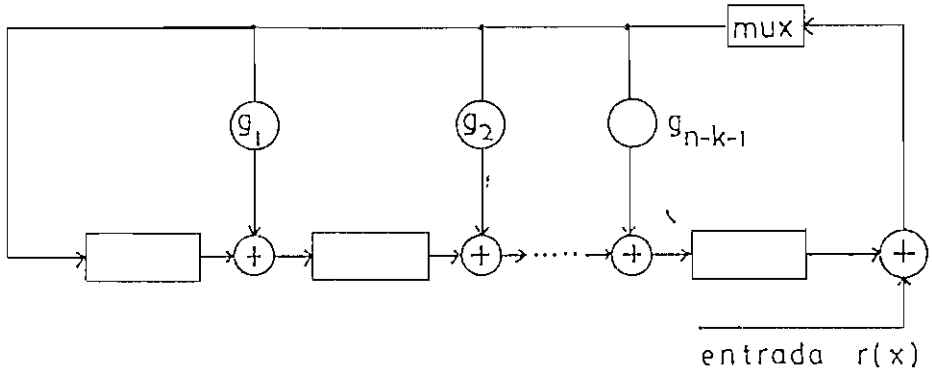


FIGURA 3.7 Circuito para calcular el síndrome.

Este circuito también nos permite calcular el síndrome, aquí, a pesar de que r(x) ha sido desplazado en su totalidad dentro de los registros, el contenido de estos forman el síndrome $s^{(n-k)}(x)$ de $r^{(n-k)}(x)$, por lo que para obtener el síndrome s(x) hay que desplazar (n-k) veces este resultado.

El síndrome debe ser calculado en base al vector recibido a pesar de desconocer el tipo de error con lo que el decodificador ha de estimar el tipo de error en base al síndrome.

> "Los códigos cíclicos son muy efectivos para detectar errores aleatorios o tipo ráfaga. El circuito detector de error es simplemente un circuito de síndrome con una compuerta OR con los dígitos del síndrome como entradas". (33).

La complejidad de un circuito que realiza la decodificación es directamente proporcional a la longitud del mismo, pero esta complejidad puede ser reducida si se usan adecuadamente las propiedades geométricas y algébricas que rigen a los códigos cíclicos.

La estructura cíclica de estos códigos permite decodificar al vector recibido en forma serial. Un circuito que realiza la decodificación es presentado en la figura 3.8 a.

En este circuito los dígitos recibidos son decodificados uno por uno. El registro síndrome calcula el síndrome de la palabra almacenada en el buffer, el cual es comparado con un tipo de error utilizando el circuito detector del error. Este es un circuito lógico combinacional diseñado de tal manera que su salida es 1 si y sólo si el valor del síndrome corresponde a un tipo de error igual a $e(x) = x^{n-1}$.

La corrección se realiza en el dígito que está en la posición n-1, es decir el dígito que está a la salida del buffer. La información almacenada en el buffer y en el registro síndrome son cíclicamente desplazados una vez para de esta manera decodificar al siguiente dígito.

> "La decodificación de los códigos cíclicos, consiste en los mismos tres pasos para la decodificación de códigos lineales: cálculo del síndrome, asociación del síndrome con el tipo de error y corrección del error". (34).

Si el contenido del registro, síndrome es diferente de cero después de la corrección, entonces al final del proceso de decodificación se puede decir que un tipo de error incorregible ha sido detectado.

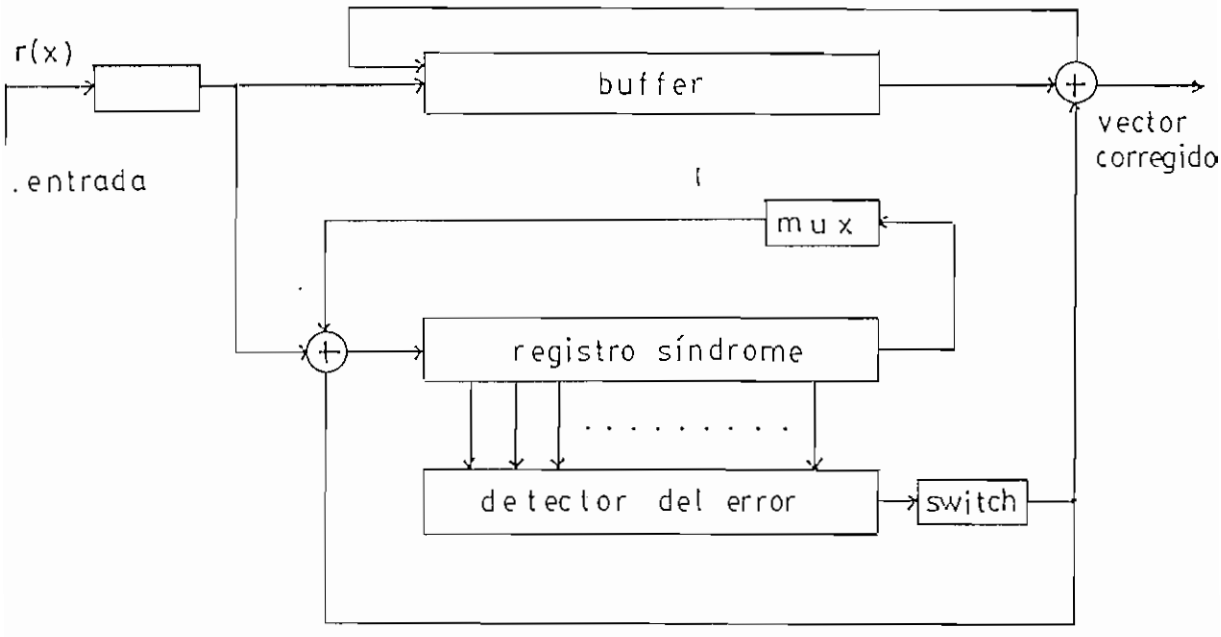


FIGURA 3.8.a Circuito decodificador bit por bit.

"El decodificador presentado puede ser usado para decodificar códigos cíclicos, sin embargo el que el decodificador sea o no sea práctico depende de la complejidad de los circuitos lógicos combinacionales del detector de error". (35).

Este decodificador se lo llama decodificador Meggit, el cual se usa como base para la decodificación de cualquier código cíclico.

En la implementación del decodificador, se puede usar también otro circuito que nos permite calcular el síndrome, en el cual el vector recibido se lo desplaza de derecha a izquierda, de tal forma que cuando todo el vector recibido $r(x)$ ha entrado totalmente en el buffer, el registro síndrome contiene $s^{(n-k)}(x)$, el cual es el síndrome de $r^{(n-k)}(x)$. Si $s^{(n-k)}(x)$ corresponde a un tipo de error $e(x) = x^{n-1}$ el dígito de más alto orden de $r(x)$, es decir r_{n-1}, es un dígito que debe ser corregido. En el polinomio $r^{(n-k)}(x)$ el dígito r_{n-1} se localiza en el coeficiente de x^{n-k-1}.

Cuando r_{n-1} es corregido, el síndrome $s^{(n-k)}(x)$ se modifica de tal manera que el nuevo síndrome así formado $s^{(n-k)}(x)$ resulta de la suma de $s^{(n-k)}(x)$ y el residuo $p(x)$ fue resuelta de la división de x^{n-k-1} para - el polinomio generador $g(x)$. Si $p(x) = x^{n-k-1}$ entonces

$$s_1^{(n-k)}(x) = s^{(n-k)}(x) + x^{n-k-1} \qquad 3.65$$

De esta forma el efecto que causa el error en el síndrome puede ser modificado si realimentamos el dígito erróneo en el registro síndrome de derecha a izquierda por medio de una compuerta OR exclusiva como se indica a continuación en la figura 3.8.b.

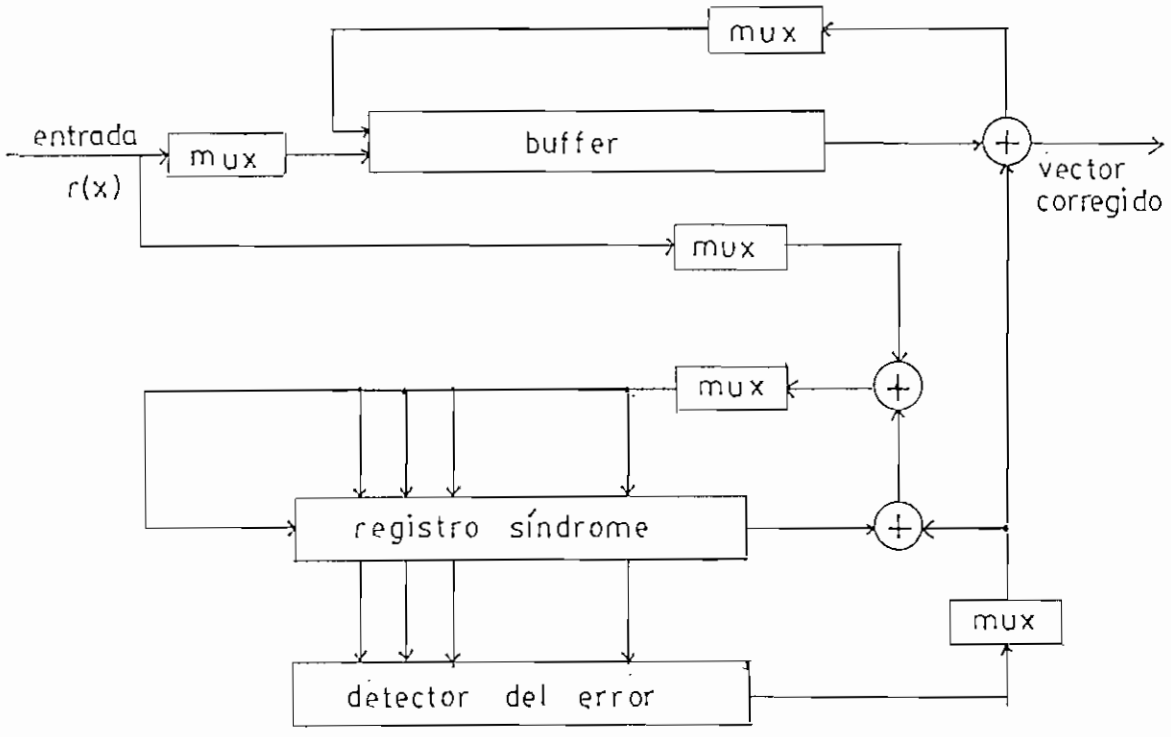


FIGURA 3.8.b Decodificador

3.2.4. Códigos Hamming.-

Se ha visto que el síndrome correspondiente a una secuencia de error recibida está dado por $\bar{s} = \bar{r}.H^T$. Si en $\bar{r}$ existe un sólo error ubicado en la i-ésima posición, entonces $r.H^T$ puede tener el mismo valor que la i-ésima fila de la matriz H. De esta forma si todas las filas de H son distintas de cero y diferentes una de otra, entonces, la secuencia de cero errores y todas las secuencias que posean un único error, poseen síndromes diferentes y así, el código es capaz de corregir todos los tipos de errores, con un error único, el código que presenta esta característica es llamado código Hamming.

Los códigos Hamming son la primera clase de códigos lineales que se estudian para la corrección de un error, junto con sus variaciones, han sido ampliamente usados para el control de errores en comunicación digital y en sistemas de almacenamiento de datos.

Para cualquier entero positivo $m \geq 3$ existe un código Hamming con los siguientes parámetros.

longitud del código	$n = 2^m - 1$
# de dígitos de información	$k = 2^m - m - 1$
# de dígitos de control	$m = n - k$
capacidad de corrección.	$t = 1$

La matriz verificadora de paridad H se forma con todas las m-tuplas diferentes de cero ubicadas como sus columnas. En forma sistemática las columnas de H están distribuídas de la siguiente forma:

$$\mathbb{H} = [\ \mathbb{I}_m \mathbb{Q}\ ]$$

donde $\mathbb{I}_m$ es la matriz identidad y la matriz $\mathbb{Q}$ consiste de 2^m -m-1 columnas cuyas m-tuplas tienen un ancho mayor a dos.

> "Los códigos Hamming son códigos en los cuales las filas de $\mathbb{H}$ son todas distintas e incluyen todas las secuencias diferentes de cero de longitud n-1" (36).

$$\mathbb{G} = [\ \mathbb{Q}^T \mathbb{I}_{2^m-m-1}\ ]$$

Todas las columnas de $\mathbb{H}$ son distintas y diferentes de cero, el resultado de la suma de dos columnas cualesquiera es diferente de cero, lo cual es debido a que la distancia mínima de un código Hamming es 3. De aquí el código es capaz de corregir tipos de error con un único error y detectar tipos de error con dos o menos errores.

> "Todos los códigos Hamming corrigen uno y solo un error". (37).

Los códigos Hamming presentan una interesante estructura que tiene que ver con los códigos perfectos.

Un código corrector de t errores es llamado código perfecto si su correspondiente arreglo standar contiene a todos los tipos de errores con t o menos errores como coset cargadores y no a otros tipos de errores.

Para la codificación y decodificación se pueden utilizar los es-

-- quemas indicados anteriormente, según la estructura que presenten.

Se puede disminuir en un número cualquiera ℓ a las columnas de la matriz H del código Hamming, resultando así un código Hamming acortado con los siguientes parámetros.

longitud del código	$n = 2^m - \ell - 1$
# de dígitos de información	$k = 2^m - m - \ell - 1$
# de dígitos de control	$m = n - k$
distancia mínima.	$d_{min} \geq 3$

3.2.5.- Códigos Hamming Cíclicos.-

Una parte muy importante de los códigos cíclicos es aquella que trata sobre los códigos Hamming cíclicos con una longitud del código igual a $2^m - 1$ con $m \geq 3$, los cuales son generados por un polinomio primitivo $p(x)$ de grado m.

Dividiendo x^{m+i} para el polinomio generador $p(x)$ se tiene que:

$$x^{m+i} = a_i(x)\, p(x) + b_i(x)$$

Como X no es un factor del polinomio primitivo $p(x)$, x^{m+i} y $p(x)$ son primos entonces se puede decir que $b_i(x)$ es diferente de cero y se conforma sólo de los dos últimos términos. Si $b_i(x)$ tuviese un sólo término entonces:

$$x^{m+i} = a(x)\, p(x) + x^j \qquad 0 \leq j \leq n \qquad 3.66$$

$$x^j (x^{m-i-j} + 1) = a_i(x)\, p(x) \qquad 3.67$$

como x^j y $p(x)$ son primos, $p(x)$ no puede dividir a $x^{m+i-j}+1$ ya que $m+i-j < 2^m - 1$ y $p(x)$ es un polinomio primitivo de grado m.- Luego, para $0 \leq i < 2^m - m - 1$, $b_i(x)$ contiene los dos últimos términos.

Dado $H = [\, I_m \; Q \,]$ la matriz verificadora de paridad del código-cíclico generado por $p(x)$, donde I_m es la matriz identidad mxn y Q es una matriz $m \times (2^m - m - 1)$ y dado $b_i = (b_{i,0}\; b_{i,1} \ldots b_{i,m-1})$ la m-tupla correspondiente a $b_i(x)$, entonces a partir de 3.60 la matriz Q tiene $(2^m - m-1)$ polinomios $b_i(x)$ con $0 \leq i < 2^m - m - 1$ formando parte de-sus columnas, las cuales son todas diferentes y cada columna tiene por lo menos dos elementos unos.

La decodificación de códigos Hamming cíclicos puede ser realizada fácilmente, ya que si un sólo error ocurre en la más alta posición del-vector $r(x)$, el cálculo del síndrome dará como resultado un valor $s(x) = x^{m-1}$, es decir que $s = (0,0,0,\ldots..0,1)$. Es por esto que una compuer-ta AND es suficiente para actuar como el circuito detector de error,-simplificando la complejidad del circuito decodificador, ya que el cir-cuito combinacional se reduce a una compuerta AND ,como en la siguien-te figura.

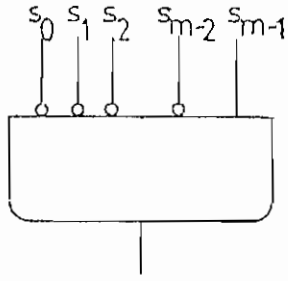


FIGURA 3.9 Circuito detector del error de código cíclico.

Mientras el síndrome no sea (0,0,0,.... 0,1) no se hace la corrección del dígito erróneo y la salida de la AND es cero. Cuando se cumple la condición anterior para el síndrome, la salida de la compuerta es uno y el dígito es corregido a la salida del buffer.

Ahora vamos a ver como un código Hamming puede detectar dos errores utilizando el circuito anterior.

Si r(x) es el polinomio recibido entonces se puede realizar las siguientes divisiones

$$x^m r(x) = a_i(x)\, p(x) + S_p(x)$$

$$r(x) = a(x)\,(x+1) + \sigma$$

con $S_p(x)$ de grado menor o igual a m-1 y σ es cero o uno.

Si $S_p(x)=0$ y $\sigma=0$ entonces r(x) es divisible por $(x+1)p(x)$ y por lo tanto es un código polinomial, de otra manera r(x) no es un código polinomial y por lo tanto contiene error.

Definiendo al síndrome de r(x) como

$$s(x) = x\, S_p(x) + \sigma$$

se puede decir que un simple error ocurre si $S_p(x) \neq 0$ y $\sigma = 1$. Cuando un doble error ocurre entonces se tiene que $\sigma = 0$ con estos antecedentes se puede implementar el siguiente decodificador corrector de un simple error y detector de dos errores:

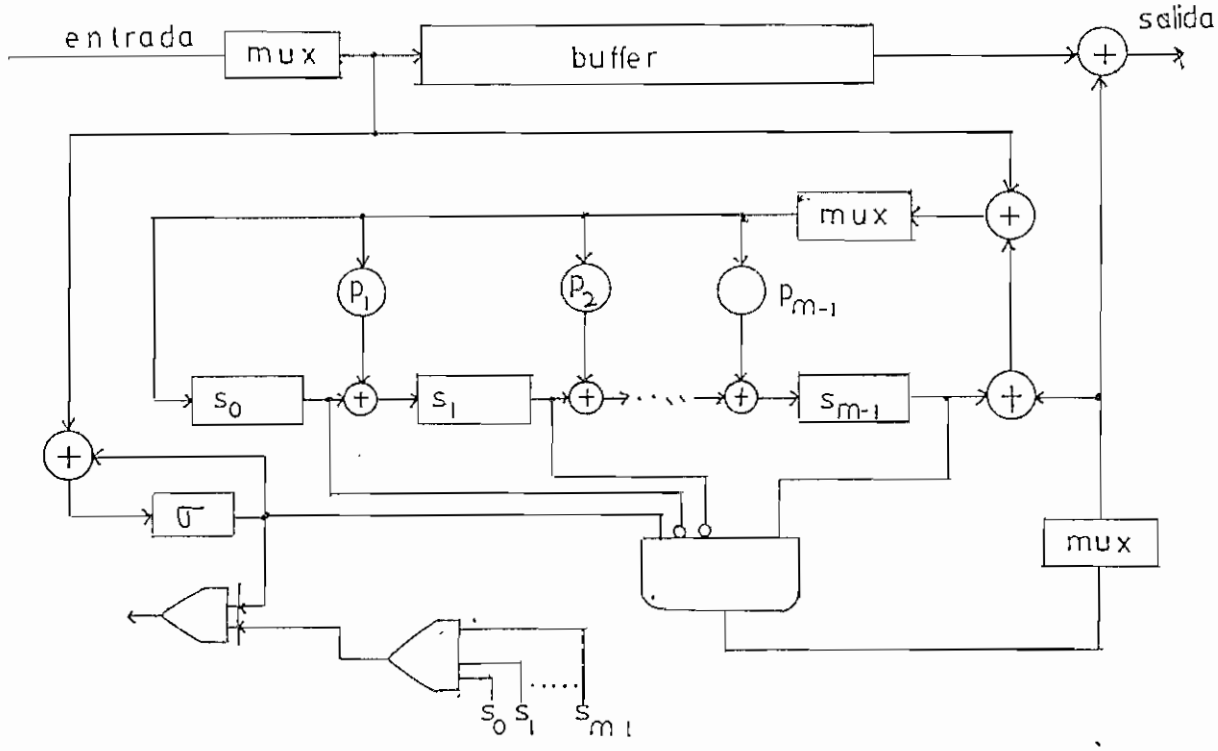


FIGURA 3.10 Circuito detector y corrector de error.

Así la decodificación se realiza de la siguiente forma:

- Con $\sigma = 0$ y $S_p(x) = 0$ no existe error alguno y por lo tanto no hay corrección.

- Con $\sigma = 1$ y $S_p(x) \neq 0$ el decodificador asume que ha ocurrido un simple error y procede a la corrección ya explicada.

- Con $\sigma = 0$ y $S_p(x) \neq 0$ el decodificador asume la existencia de un doble error e indica la detección de errores.

- Con $\sigma = 1$ y $S_p(x) = 0$ se detecta tipos de errores que tienen número impar de errores y que sean divisibles para $p(x)$.

3.2.6.- Acortamiento de Códigos Cíclicos.-

Muchas veces es prudente reducir la longitud de un código para tratar de encontrar otro más corto, así, de un código cíclico (n,K) en el cual los códigos vectores tienen los q bits de información más altos iguales a cero, se puede formar un subcódigo de C formado por los 2^{k-q} vectores de código a los cuales se les ha quitado los bits más altos que son iguales a cero.

De esta forma se obtiene un conjunto de 2^{k-q} vectores acortados de longitud $n-q$ que forman un código lineal $(n-q, k-q)$ Este código es llamado código cíclico acortado, conservando la misma capacidad de corregir errores pero no la propiedad cíclica del código del cual fue obtenido.

> "El código cíclico acortado es un subconjunto del código cíclico del cual se deriva, y por ende tiene la mínima distancia y la habilidad de corregir errores como la del código original". (38).

La codificación y decodificación de este código puede ser implementada con los mismos circuitos diseñados para el código cíclico original, ya que el quitar dígitos de información no afecta a la verificación de paridad ni al cálculo del síndrome.

Sin embargo, el utilizar estos circuitos para grandes valores de puede ser una pérdida de tiempo en la transmisión, ya que a pesar de existir q dígitos menos, estos circuitos deben completar los n desplazamientos generando por lo tanto un retardo en la codificación y en la decodificación.

El retardo en la decodificación se puede eliminar de la siguiente - manera.

Si $r(x) = r_0 + r_1 x + r_2 x^2 + \cdots\cdots r_{n-q-1} x^{n-q-1}$ es el polinomio correspondiente al vector recibido, el síndrome para decodificar el dígito-recibido r_{n-q-1}, al utilizar el decodificador original, es igual al residuo que resulta de la división de $x^{n-k-q} r(x)$ por el polinomio generador - $g(x)$. De esta forma

$$x^{n-k-q} r(x) = a_1(x)\, g(x) \quad s^{(n-k+q)}(x) \qquad 3.68$$

donde $s^{(n-k+q)}(x)$ es el residuo y por lo tanto es el síndrome del dígito recibido r_{n-q-1}

Representando a $p(x)$ como el residuo de la siguiente división

$$x^{n-k+q} = a_2(x)\, g(x) + p(x) \qquad 3.69$$

$$p(x) = x^{n-k+q} + a(x)\, g(x) \qquad 3.70$$

con

$$p(x) = p_0 + p_1 x + \cdots\cdots p_{n-k-1} x^{n-k-1}$$

$$r(x)\, p(x) = a_1(x)\, g(x) - s^{(n-k+q)}(x) - a_2(x)\, g(x)\, r(x)$$

$$r(x)\, p(x) = [a(x) + a(x)\, r(x)]\, g(x) + s^{(n-k+q)}(x)$$

Esta última expresión nos sugiere que el síndrome se puede obtener realizando la multiplicación $r(x)p(x)$ para luego este resultado dividirlo para $g(x)$, eliminando de esta forma los q desplazamientos indeseados,- al calcular el síndrome.

Estas operaciones de multiplicación y división de dos polinomios - binarios puede ser realizada por el siguiente circuito.

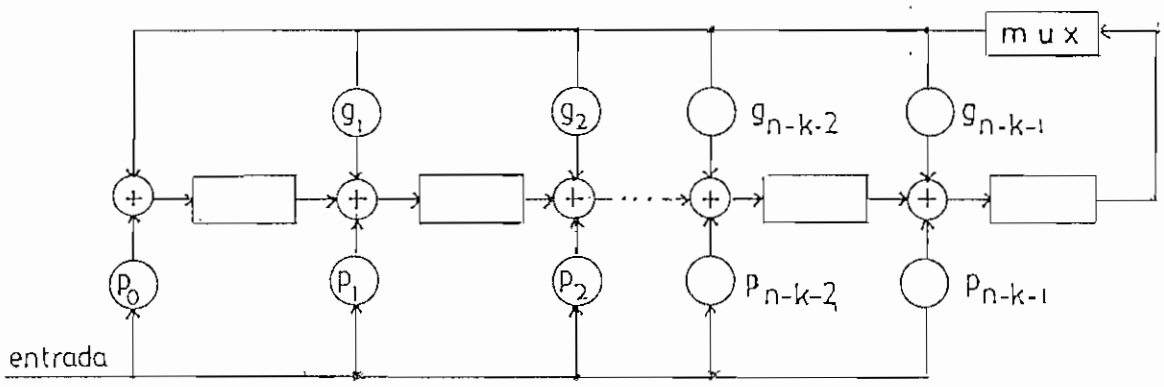


FIGURA 3.11 Multiplicador por g(x) y divisor por p(x)

Cuando r(x) ha entrado en su totalidad en el circuito, en los registros ya se dispone inmediatamente de los coeficientes del síndrome.

Para la realización de este circuito vemos que es necesario conocer el polinomio p(x), el cual a partir de la expresión 3.70 puede ser obtenido de la siguiente forma:

$$x^{n-k-q} = a_2(x)\, g(x) + p(x)$$

al realizar la división, encontramos a p(x) como el residuo.

En el circuito de la figura 3.9 se realizan los siguientes cambios.

En primer lugar el circuito para el cálculo del síndrome se lo cambia por el presentado anteriormente. En segundo lugar el circuito para la detección del error debe ser modificado para que detecte y corrija un error en la posición x^{n-q-1}.

Si el síndrome es calculado haciendo desplazar al vector recibido r(x) de derecha a izquierda el polinomio p(x) es el residuo que resulta de la división de $x^{n-q-1}\, x^{n-k} = x^{2n-k-q-1}$ para el polinomio generador g(x), de esta forma el efecto del dígito erróneo e_{n-q-1} en el-

síndrome es quitado sumando p(x) al síndrome en el registro síndrome.

3.3. Códigos BCH.-

Este tipo de códigos son una generalización de los códigos Hamming para la corrección múltiple de errores.

> "Fueron descubiertos por Hocquenghem (1959) e independientemente por Bose y Chaudhuri (1960), los cuales tienen propiedades poderosas para la corrección de errores y algoritmos de decodificación simples". (39).

La estructura cíclica de estos códigos fue propuesta por Peterson, el cual también propuso su primer algoritmo de decodificación que se ha generalizado y se ha optimizado.

En esta tesis se consideran los códigos binarios BCH, que son importantes tanto en la teoría como en la implementación.

Para cualquier entero positivo m tal que $m \geq 3$ y cualquier t con $t < 2^{m-1}$ existe un código binario BCH que cumple con los siguientes parámetros:

Longitud de bloque	$n = 2^m - 1$
# de dígitos de control	$n - k \leq m t$
distancia mínima	$d_{min} \geq 2t + 1$

Estos parámetros nos indican que este código es capaz de corregir

simultáneamente hasta t errores que puedan ocurrir en un bloque de $n = 2^m - 1$ dígitos.

> "Ellos son claramente poderosos, es decir, ellos tienen una capacidad simultánea para altas velocidades y habilidad para una buena corrección de errores". (40).

El polinomio generador de este código está definido en términos de las raíces que pertenecen al campo de Galois $GF(2^m)$. Si α es un elemento primitivo en $GF(2^m)$, se define el polinomio generador $g(x)$ de un código BCH corrector de t errores y longitud 2^m-1 como el polinomio de más alto grado sobre GF(2) que tiene a $\alpha, \alpha^2, \alpha^3, \ldots \alpha^{2t}$ como sus raíces, es decir se cumple que $g(\alpha^i) = 0$ para $1 \leqslant i \leqslant 2t$

> "Los códigos Bose-Chaudhuri, son códigos cíclicos que son bien definidos en términos de las raíces del polinomio generador". (41).

Ya se demostró que todas las raíces de $g(x)$ son $\alpha, \alpha^2, \ldots \alpha^{2t}$ conjuntamente con todas sus conjugadas.

Si $\phi_i(x)$ es el mínimo polinomio de α^i, entonces $g(x)$ es el mínimo común múltiplo de $\phi_1(x), \phi_2(x), \ldots \phi_{2t}(x)$

$$g(x) = MCM(\phi_1(x), \phi_2(x) \ldots \phi_{2t}(x)) \qquad 3.71$$

Cuando i es un entero impar, puede ser expresado de la siguiente manera; $i = i' 2^p$

donde i' es un número par y $p \geqslant 1$. De esta forma $\alpha^i = (\alpha^{i'})^{2^p}$ es el

conjugado de $\alpha^{i'}$ y por lo tanto α^{i} y $\alpha^{i'}$ tienen el mismo polinomio mínimo. Esto se traduce a la siguiente igualdad.

$$\phi_{i'}(x) \quad \phi_{i}(x)$$

De esta forma, cada potencia impar de α en la expresión 3.71 tiene el mismo polinomio mínimo que la precedente potencia par dada en dicha expresión. El polinomio generador g(x) del código BCH viene dado por la siguiente expresión.

$$g(x) = MCM\,(\phi_1(x), \phi_3(x), \ldots . \phi_{2t-1}(x)) \qquad 3.72$$

El grado de cada polinomio mínimo es menor o igual a m y el grado de g(x) es a lo mucho igual a mt, esto indica que el número de dígitos verificadores de paridad, n-k, del código es menor o igual a mt.- Estos códigos son usualmente llamados códigos BCH primitivos.

Un código BCH corrector de un solo error de longitud 2^{m-1} según la expresión 3.72, es generado por un polinomio g(x) dado por

$$g(x) = \phi_1(x)$$

donde α es un elemento primitivo de $GF(2^m)$ y $\phi_1(x)$ es un polinomio primitivo de grado m. Esta clase de códigos son los códigos Hamming vistos en la sección anterior.

Si $v(x) = v_0 + v_1 x + \cdots v_{n-1} x^{n-1}$ es un polinomio cuyos coeficientes pertenecen a GF(2) y si además v(x) tiene a $\alpha, \alpha^2, \ldots . \alpha^{2t}$ como sus raíces, entonces v(x) es divisible para los polinomios mínimos $\phi_1(x), \phi_2(x)$ $\ldots . \phi_{2t}(x)$ de $\alpha, \alpha^2, \ldots . \alpha^{2t}$. Obviamente v(x) también es divisible para el mínimo común múltiplo de estos polinomios que es el polinomio generador.

De esta forma v(x) es un código polinomial y consecuentemente se puede definir a un código BCH corrector de t errores y de longitud $n = 2^m - 1$ en términos de la palabra código del código BCH.

> "Una n-tupla binaria $\bar{v} = (v_0, v_1, \ldots . v_{n-1})$ es una-palabra código si y sólo si el polinomio $v(x) = v_0 + v_1 x + \ldots . v_{n-1} x^{n-1}$ tiene a $\alpha, \alpha^2, \ldots \alpha^{2t}$ como raíces". (42).

Si $v(x) = v_0 + v_1 x + \ldots . v_{n-1} x^{n-1}$ es un polinomio de código, de un código BCH corrector de t errores, entonces se cumple la siguiente igualdad

$$v(\alpha^i) = 0 \qquad 1 \leq i \leq 2t \qquad 3.73$$

Esta expresión puede ser escrita en forma matricial con el siguiente producto.

$$(v_0, v_1, \ldots . v_{n-1}) \cdot \begin{bmatrix} 1 \\ \alpha^i \\ \alpha^{2i} \\ \\ \alpha^{(n-1)i} \end{bmatrix} = 0 \qquad 3.74$$

Si creamos la siguiente matriz, representada por $|H|$.

$$|H| = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \ldots\ldots & \alpha^{n-1} \\ 1 & \alpha^2 & (\alpha^2)^2 & (\alpha^2)^3 & \ldots\ldots & (\alpha^2)^{n-1} \\ \vdots & & & & & \vdots \\ 1 & \alpha^i & (\alpha^i)^2 & (\alpha^i)^3 & \ldots\ldots & (\alpha^i)^{n-1} \\ \vdots & & & & & \vdots \\ 1 & \alpha^{2t} & (\alpha^{2t})^2 & (\alpha^{2t})^3 & \ldots\ldots & (\alpha^{2t})^{n-1} \end{bmatrix} \qquad 3.75$$

podemos realizar el producto $\bar{v}.IH^T$ con $\bar{v} = (v_0, v_1 \ldots\ldots v_{n-1})$.
El resultado del producto será igual a cero.

$$(v_0 , v_1 , \ldots\ldots v_{n-1}) \cdot \begin{bmatrix} 1 & 1 & 1 \cdots\cdots 1 \\ \alpha & \alpha^2 & \alpha^i \ldots\ldots \alpha^{2t} \\ \alpha^2 & (\alpha^2)^2 & (\alpha^i)^2 \ldots\ldots (\alpha^{2t})^2 \\ \vdots & & \vdots \\ \alpha^{n-1} & (\alpha^2)^{n-1} & (\alpha^i)^{n-1} \ldots\ldots (\alpha^{2t})^{n-1} \end{bmatrix} = 0 \qquad 3.76$$

Esta es una manera de comprobar la condición de que el polinomio dado por $\bar{v} = (v_0, v_1, \ldots\ldots v_{n-1})$ tiene como raíces a α^i para todo $1 \leq i \leq 2t$

Podemos establecer que si un vector cumple con la condición dada por 3.76, entonces esta es una palabra de código y por lo tanto IH es la matriz verificadora de paridad de este código.

Si para algún valor de i e j , α^j es la conjugada de α^i entonces $v(\alpha^j) = 0$ si y solo si $v(\alpha^i) = 0$, es por esto que la comprobación de que α^j es raíz de v(x) puede ser omitida y tan sólo se puede comprobar que α^i es raíz de v(x). De esta manera se puede quitar la j-esima columna de IH . Generalizando se puede obtener una matriz IH reducida, dada por:

$$|H| = \begin{bmatrix} 1 & \alpha & \alpha^2 \cdots\cdots \alpha^{n-1} \\ 1 & \alpha^3 & (\alpha^3) \cdots\cdots (\alpha^3)^{n-1} \\ 1 & \alpha^5 & (\alpha^5) \cdots\cdots (\alpha^5)^{n-1} \\ \vdots & & \vdots \\ 1 & \alpha^{2t-1} & (\alpha^{2t-1}) \cdots\cdots (\alpha^{2t-1})^{n-1} \end{bmatrix} \qquad 3.77$$

Los elementos de esta matriz H son elementos de $GF(2^m)$ y cada uno de estos elementos pueden ser representados por arreglos de m elementos que pertenecen a GF(2). Se debe recordar que los elementos de una extensión de campo $GF(q^m)$ de un campo GF(q), pueden ser representados como m vectores sobre GF(q).

Para probar que la distancia mínima del código BCH corrector de t errores es por lo menos 2t+1, es necesario primero probar que el número de columnas necesarias de H para que su suma sea cero, es menor o igual a 2t.

Suponiendo que existe un código vector $\overline{v} = (v_0, v_1, \ldots v_{n-1})$ con un ancho $d \leq 2t$ y dados $v_{j1}, v_{j2}, \ldots v_{j\delta}$ los cuales son los elementos diferentes de cero de $\overline{v}$, es decir que $v_{j1} = v_{j2} = \cdots v_{j\delta} = 1$ se tiene que:

$$\overline{v} \cdot H^T = 0$$

$$(0 = (v_{j1}, v_{j2}, \ldots v_{j\delta}) \cdot \begin{bmatrix} \alpha^{j1} & (\alpha^{2})^{j1} & \ldots & (\alpha^{2t})^{j1} \\ \alpha^{j2} & (\alpha^{2})^{j2} & \ldots & (\alpha^{2t})^{j2} \\ \vdots & \vdots & & \vdots \\ \alpha^{j\delta} & (\alpha^{2})^{j\delta} & \ldots & (\alpha^{2t})^{j\delta} \end{bmatrix}$$

$$(0 = (1, 1, \ldots 1) \cdot \begin{bmatrix} \alpha^{j1} & (\alpha^{j1})^2 & \ldots & (\alpha^{j1})^{2t} \\ \alpha^{j2} & (\alpha^{j2})^2 & \ldots & (\alpha^{j2})^{2t} \\ \vdots & \vdots & & \vdots \\ \alpha^{j\delta} & (\alpha^{j\delta})^2 & \ldots & (\alpha^{j\delta})^{2t} \end{bmatrix}$$

o lo que es lo mismo

$$0 = (1, 1, \ldots 1) \cdot \begin{vmatrix} \alpha^{j1} & (\alpha^{j1})^2 & \ldots & (\alpha^{j1})^{\delta} \\ \alpha^{j2} & (\alpha^{j2})^2 & \ldots & (\alpha^{j2})^{\delta} \\ \vdots & \vdots & & \vdots \\ \alpha^{j\delta} & (\alpha^{j\delta})^2 & \ldots & (\alpha^{j\delta})^{\delta} \end{vmatrix} .$$

donde la segunda matriz cuadrada es $\delta \times \delta$. Para satisfacer la igualdad de la última matriz, el determinante de la matriz $\delta \times \delta$ debe ser cero.

$$\begin{vmatrix} \alpha^{j1} & (\alpha^{j1})^2 & \ldots & (\alpha^{j1})^{\delta} \\ \alpha^{j2} & (\alpha^{j2})^2 & \ldots & (\alpha^{j2})^{\delta} \\ \vdots & \vdots & & \vdots \\ \alpha^{j\delta} & (\alpha^{j\delta})^2 & \ldots & (\alpha^{j\delta})^{\delta} \end{vmatrix} = 0$$

sacando factor común a cada fila del determinante se tiene que:

$$\alpha^{(j1+j2+\cdots j\delta)} \begin{vmatrix} 1 & \alpha^{j1} & \ldots & \alpha^{j1(\delta-1)} \\ 1 & \alpha^{j2} & \ldots & \alpha^{j2(\delta-1)} \\ \vdots & \vdots & & \vdots \\ 1 & \alpha^{j\delta} & \ldots & \alpha^{j\delta(\delta-1)} \end{vmatrix} .$$

El determinante que se ha creado es el determinante de Vandermonde, el cual, es diferente de cero. Con lo que el factor restante $\alpha^{(j1+j2\cdots j\delta)}$ tiene que ser diferente de cero para que se cumpla esa igualdad, lo cual es una contradicción y por lo tanto el código vector $\bar{v}$ diferente de cero que se asumió tenía un ancho $\delta \leq 2t$ es invalido.

Esto implica que el mínimo ancho de un código BCH corrector de t errores es 2t + 1, consecuentemente la mínima distancia del código es 2t + 1.

El parámetro 2t + 1 es usualmente llamada distancia de diseño del código.

La verdadera distancia mínima de un código BCH puede o no puede ser igual a la distancia de diseño, sin embargo hay muchos casos donde la verdadera distancia mínima es igual a la distancia de diseño y en otros donde la verdadera distancia mínima es más grande que la distancia de diseño.

Los códigos binarios BCH de longitud $n = 2^m - 1$ pueden ser formados de la misma manera con la que se forman aquellos códigos de longitud $n = 2^m - 1$. Dado β un elemento de orden n en el campo $GF(2^m)$ y n factor de $2^m - 1$ se define un polinomio binario $g(x)$ de grado mínimo que tiene como raíces a los siguientes elementos:

$$\beta, \beta^2, \beta^3, \ldots\ldots \beta^{2t}$$

Si $\Psi_1(x), \Psi_2(x), \ldots\ldots \Psi_{2t}(x)$ son los polinomios mínimos de $\beta_1, \beta_2, \ldots\ldots \beta_{2t}$ respectivamente, entonces se tiene que

$$g(x) = MCM(\Psi_1(x), \Psi_2(x), \ldots\ldots \Psi_{2t}(x)) \qquad 3.78$$

donde $\beta^n = 1, \beta, \beta^2, \ldots\ldots \beta^{2t}$ son raíces de $X^n + 1$.

Con $g(x)$ factor de $X^n + 1$, el código cíclico generado por $g(x)$ es un código BCH corrector de t errores de longitud n. De una manera similar, para el caso en que $n = 2^m - 1$ se puede aumentar un número de dígitos verificadores de paridad que sea menor o igual a mt siendo la distancia mínima de este código al menos $2t + 1$.

Si β no es un elemento primitivo de $GF(2^m)$ con $n \neq 2^m - 1$, el código así formado es llamado código BCH no primitivo.

Una nueva definición para códigos binarios BCH, con otros términos puede ser ya entendida, y se da a continuación.

Dado β un elemento de $GF(2^m)$ y cualquier entero no negativo ℓ_o. Entonces, un código BCH binario con una distancia de diseño do , es generado por un polinomio binario g(x) de grado mínimo en el cual todas sus raíces pueden ser obtenidas a partir de potencias consecutivas de β.

$$\beta^{\ell_o}, \beta^{\ell_o+1}, \ldots\ldots \beta^{\ell_o+do-2}$$

Para $0 \leq i < do - 1$ Dados $\Psi_i(x)$ el polinomio mínimo y n_i el orden de β^{ℓ_o+i} se tiene que.

$$g(x) = MCM(\Psi_0(x), \Psi_1(x), \ldots\ldots \Psi_{do-2}(x)) \quad 3.79$$

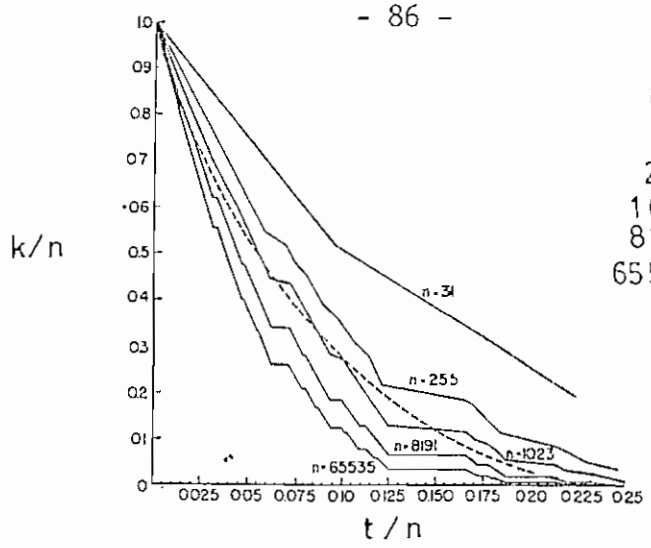
g(x) genera un código de longitud;

$$n = MCM\ (n_0, n_1, \ldots\ldots n_{do-2})$$

El código BCH definido anteriormente tiene una distancia mínima con un valor de al menos do y no tiene más que m(do - 1) dígitos verificadores de paridad, además es capaz de corregir hasta (do - 1/ 2)errores.

Si $\ell_o = 1$ $do = 2t+1$ y β es un elemento primitivo de $GF(2^m)$, el código se convierte en un código BCH primitivo corrector de t errores de longitud n y orden β .

En la definición dada para el código BCH con distancia de diseño do se requería que g(x) tenga como raíces las do - 1 potencias consecutivas de β . Esto garantiza que el código tenga al menos una distancia mínima igual a do .



n	Nºtotal de códigos
31	5
255	33
1023	105
8191	629
65535	4113

3.3.1. Decodificación de Códigos BCH.-

Como los códigos BCH son códigos cíclicos, el proceso de codificación puede ser realizado con las mismas normas tratadas para códigos cíclicos en temas anteriores. En cambio los procesos de decodificación requieren de algoritmos decodificadores para simplificar la decodificación, puesto que corrigen más de un error simultáneamente.

Si una palabra código $v(x) = v_0 + v_1 x + v_2 x + \ldots\ldots v_{n-1} x^{n-1}$ es transmitida, y como resultado de los errores introducidos en ella, se recibe un vector $r(x) = r_0 + r_1 x + r_2 x^2 + \ldots\ldots r_{n-1} x^{n-1}$. Se puede definir al tipo de error introducido como e(x) tal que

$$r(x) = v(x) + e(x)$$

Como ya es usual, lo primero que se realiza al decodificar el código es calcular el síndrome del vector recibido. La decodificación de un código BCH primitivo corrector de t errores, nos proporciona un síndrome dado por

$$\bar{S} = (s_1, s_2, \ldots\ldots s_{2t}) = r \cdot |H^T \qquad 3.80$$

el cual consta de 2t elementos, donde la matriz $|H$ viene dada por 3.75.

De las ecuaciones 3.80 y 3.75 se puede encontrar el i-ésimo componente del síndrome evaluando el polinomio r(x) con el valor α^i de tal forma que

$$s_i = r(\alpha^i) \qquad 1 < i \leq 2t$$

$$s_i = r_0 + r_1\alpha^i + r_2\alpha^{2i} + \ldots r_{n-1}\,\alpha^{(n-1)i}$$

Los componentes del síndrome resultan ser elementos que pertenecen al campo $GF(2^m)$.

Si dividimos r(x) para el polinomio primitivo $\phi_i(x)$ de α^i se obtiene la siguiente expresión

$$r(x) = a_1(x)\,\phi_i(x) - b_i(x) \qquad 3.81$$

en la cual $b_i(x)$ representa el residuo de esta división con un grado menor al que tiene $\phi_i(x)$.

como $\qquad \phi_i(\alpha^i) = 0$

$$r(\alpha^i) = a_1(\alpha^i).0 + b_i(\alpha^i) = b_i(\alpha^i)$$

Esta última igualdad nos da otra forma de obtener el síndrome, evaluando el residuo $b_i(x)$ con $x = \alpha^i$.

Sabiendo que $\alpha, \alpha^2, \ldots\ldots \alpha^{2t}$ son las raíces de cada polinomio de código, es decir que $v(\alpha^i) = 0$ para $1 \leq i \leq 2t$ se tiene que:

$$r(x) = e(x) + v(x)$$
$$r(\alpha^i) = e(\alpha^i) + \cancel{v(\alpha^i)}^{\,0}$$
$$r(\alpha^i) = e(\alpha^i)$$

por lo tanto $s_i = e(\alpha^i)$ 3.82

Claramente se ve que el síndrome s_i depende solamente del tipo de error $\bar{e}$ que se ha introducido en el mensaje. Si e(x) tiene z errores que estan ubicados en las posiciones $x^{j1}, x^{j2}, \cdots\cdots x^{jz}$, e(x) tiene la forma

$$e(x) = x^{j1} + x^{j2} + \cdots x^{jz} \qquad 0 \leqslant j1 < j2 \cdots jz < n$$

reemplazando esta expresión en 3.82 y desarrollando para todo i, los componentes del síndrome vienen dados por las siguientes expresiones:

$$\begin{aligned} s_1 &= \alpha^{j1} + \alpha^{j2} + \cdots\cdots \alpha^{jz} \\ s_2 &= (\alpha^{j1})^2 + (\alpha^{j2})^2 + \cdots\cdots (\alpha^{jz})^2 \\ &\vdots \\ s_{2t} &= (\alpha^{j1})^{2t} + (\alpha^{j2})^{2t} + \cdots\cdots (\alpha^{jz})^{2t} \end{aligned} \qquad 3.83$$

donde $\alpha^{j1}, \alpha^{j2}, \cdots\cdots \alpha^{jz}$ son elementos desconocidos.

Deteniéndonos un poco en estas ecuaciones, podemos darnos cuenta que cualquier método para resolverlas representa un algoritmo decodificador para códigos BCH, ya que una vez calculados $\alpha^{j1}, \alpha^{j2}, \ldots\ldots \alpha^{jz}$ las potencias $j1, j2, \ldots\ldots jz$ nos indican las posiciones erróneas en el mensaje.

El sistema de ecuaciones tiene muchas soluciones 2^k en total, y cada solución corresponde a un tipo de error diferente. Si el número de errores existentes en el tipo de error e(X) es menor o igual a t, es decir que $z \leqslant t$, la solución correcta es aquella en la cual el tipo de error tiene el más pequeño número de errores, es decir el polinomio e(x)

tiene el menor número de coeficientes iguales a 1. Esta solución corresponde al tipo de error e(X) más probable causado por el canal ruidoso.

> "Para grandes valores de t, resolver las ecuaciones directamente es dificultoso e inefectivo". (43).

A continuación se da un método para determinar $\alpha^{j\ell}$ para $\ell = 1,2,3,\ldots z$ a partir de los componentes del síndrome S_i .

Con el objeto de agilitar la escritura, se plantea la siguiente igualdad

$$\beta = \alpha^{j\ell} \qquad 1 \leq \ell \leq z$$

A estos elementos se los llamará números localizadores del error. Utilizando esta igualdad, el sistema de ecuaciones dado por 3.83 tendrá la siguiente forma:

$$\begin{aligned} S_1 &= \beta_1 + \beta_2 + \cdots\cdots \beta_z \\ S_2 &= \beta_1^2 + \beta_2^2 + \cdots\cdots \beta_z^2 \\ &\vdots \qquad\quad \vdots \qquad\quad \vdots \\ S_{2t} &= \beta_1^{2t} + \beta_2^{2t} + \cdots \beta_z^{2t} \end{aligned} \qquad 3.84$$

Estas 2t ecuaciones, son funciones simétricas en términos de $\beta_1, \beta_2, \ldots \beta_z$ y cada una de ellas son conocidas como funciones simétricas de suma de potencias.

> "En general, estas ecuaciones pueden tener muchas soluciones, cada una correspondiente a un tipo de error en el mismo coset de un grupo aditivo de palabras código". (44).

Definimos ahora un polinomio $\sigma(x)$ como:

$$\sigma(x) = (1+\beta_1 x)(1+\beta_2 x)\cdots\cdots(1+\beta_z x)$$

$$\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots\cdots \sigma_z x^z \qquad 3.85$$

tal que las raíces de este polinomio sean $\beta_1^{-1}, \beta_2^{-1}, \ldots\ldots \beta_z^{-1}$ que son los inversos de los números localizadores de error.

Es por esta razón que $\sigma(x)$ se lo llama polinomio localizador de errores. $\sigma(x)$ es un polinomio desconocido cuyos coeficientes deben ser encontrados en el proceso de decodificación.

Se puede relacionar a σ_i y β_j utilizando las siguientes expresiones:

$$\begin{aligned}
\sigma_0 &= 1 \\
\sigma_1 &= \beta_1 + \beta_2 + \cdots + \beta_z \\
\sigma_2 &= \beta_1\beta_2 + \beta_2\beta_3 + \cdots\cdots + \beta_{z-1}\beta_z \\
\sigma_z &= \beta_1\beta_2\beta_3\cdots\beta_z
\end{aligned} \qquad 3.86$$

los coeficientes σ_i son conocidos como funciones elementales simétricas de los números localizadores de errores β_ℓ .

Utilizando las expresiones 3.84 y 3.86 se puede relacionar σ_i con los componentes del síndrome s_j dando lugar a lo que se denomina como identidades de Newton.

$$\begin{aligned}
&S_1 - \sigma_1 = 0 \\
&S_2 - S_1\sigma_1 + 2\sigma_2 = 0 \\
&S_3 - S_2\sigma_1 + S_1\sigma_2 - 3\sigma_3 = 0 \\
&S_4 - S_3\sigma_1 + S_2\sigma_2 - S_1\sigma_3 + 4\sigma_4 = 0 \\
&\qquad\vdots
\end{aligned} \qquad 3.87$$

para el caso binario en el cual 1 + 1 = 2 = 0 se tiene que:

$$i\sigma_i \begin{cases} \sigma_i & i \text{ par} \\ 0 & i \text{ impar} \end{cases}$$

Es posible determinar las funciones simétricas elementales σ_1, $\sigma_2, \ldots . \sigma_z$ a partir de las ecuaciones dadas por 3.87 y consecuentemente determinar los números localizadores de error $\beta_1, \beta_2, \ldots, \beta_z$, calculando las raíces del polinomio localizador de error $\sigma(x)$.

Como ya se mencionó, estas ecuaciones pueden tener muchas soluciones, pero se excogerá la solución de mínimo grado que equivale a un tipo de error con mínimo número de errores.

A continuación se proporcionan los pasos que se siguen para la decodificación de un código BCH.

- Se calcula el síndrome $\bar{S} = (s_1, s_2, \ldots . s_{2t})$ del polinomio recibido $r(x)$.
- Se calcula el polinomio localizador de error $\sigma(x)$ a partir de los componentes del síndrome. $s_1, s_2, \ldots, s_{2t}$
- Se determina los números localizadores de error $\beta_1, \beta_2, \ldots, \beta_z$ calculando las raíces de $\sigma(x)$ para luego, una vez conocidas las posiciones erróneas, realizar la corrección.

El primero y el tercer paso se pueden realizar fácilmente, el segundo paso es el más complicado en todo el proceso de decodificación de los códigos BCH, el cual se lo puede realizar utilizando varios métodos de los cuales proponemos el más utilizado.

El diagrama de flujo de la figura 3.13 nos proporciona un método para encontrar $\sigma(x)$ basado en el algoritmo, presentado por Berle Kamp.- Para esto es necesario definir el siguiente polinomio $\sigma^{u}(x)$ formado en el u-ésimo paso de la iteracción.

$$\sigma^{u}(x) = 1 + \sigma_1^{u} x + \sigma_2^{u} x^2 + \cdots \sigma_{l_u}^{u} x^{l_u}$$

y la discrepancia u-ésima du

$$du = S_{u+1} + \sigma_1^{u} S_u + \sigma_2^{u} S_{u-1} + \cdots \sigma_{l_u}^{u} S_{u+1-l_u} \qquad 3.88$$

Si du = 0 entonces $\sigma^{u+1}(x) = \sigma^{u}(x)$

Si du $\neq$ 0 entonces $\sigma^{u+1}(x) \neq \sigma^{u}(x)$ y por lo tanto hay que encontrar $\sigma^{u+1}(x)$ con la siguiente expresión.

$$\sigma^{u+1}(x) = \sigma^{u}(x) + \frac{du}{dp} x^{(u-p)} \sigma^{p}(x) \qquad 3.89$$

donde $\sigma^{p}(x)$ es un polinomio de la p-ésima interacción anterior a la u-ésimo paso tal que $dp \neq 0$ y $p - l_p$ tenga el más alto valor.

El polinomio dado por 3.89 es de grado mínimo cuyos coeficientes satisfacen las primeras u + 1 identidades de Newton.

Si el grado del polinomio $\sigma(x)$ es más grande que t entonces existen más de t errores en el polinomio recibido r(x) y generalmente no es posible localizar sus posiciones en el polinomio r(x).

Si el número de errores en el polinomio recibido r(x) es menor que la capacidad diseñada, para corregir t errores en el código, no es necesario realizar los 2t pasos para encontrar el polinomio localizador $\sigma(x)$.

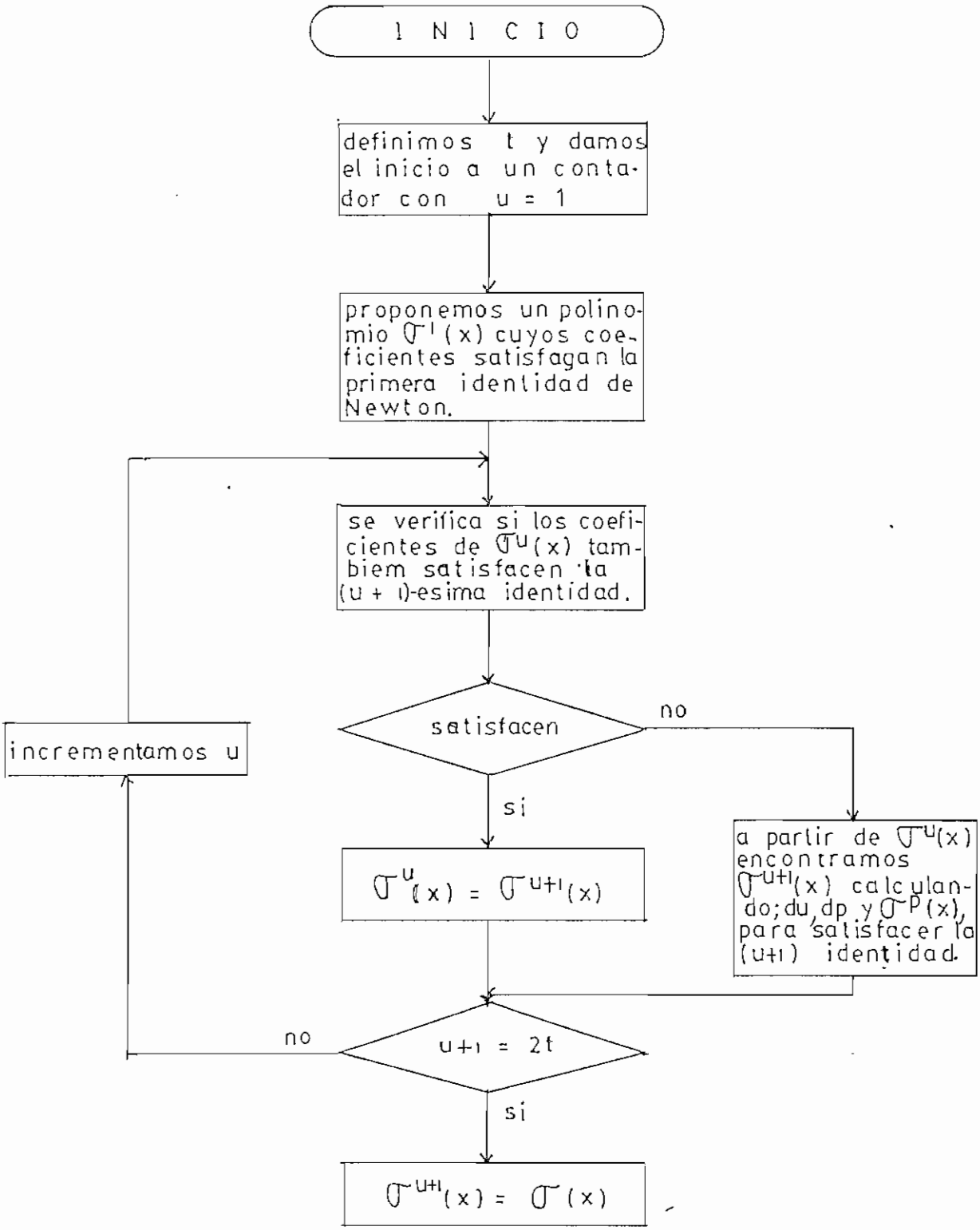

I N I C I O
definimos t y damos el inicio a un contador con u = 1
proponemos un polinomio $\sigma^1(x)$ cuyos coeficientes satisfagan la primera identidad de Newton.
se verifica si los coeficientes de $\sigma^u(x)$ tambiem satisfacen la (u + 1)-esima identidad.
satisfacen
no
si
incrementamos u
a partir de $\sigma^u(x)$ encontramos $\sigma^{u+1}(x)$ calculando; du, dp y $\sigma^p(x)$, para satisfacer la (u+1) identidad.
$\sigma^u(x) = \sigma^{u+1}(x)$
u+1 = 2t
no
si
$\sigma^{u+1}(x) = \sigma(x)$

FIGURA 3.13

Para códigos binarios BCH no es necesario realizar los 2t pasos de la iteracción, ya que solo t pasos son necesarios.

El proceso para encontrar $\sigma(x)$ puede ser realizado fácilmente, si utilizamos la siguiente tabla.

u	$\sigma(x)$	du	ℓ_u	$2u-\ell_u$
-1/2	1	1	0	-1
0	1	S_1	0	0
1	-	-	-	-
⋮				⋮
t	-	-	-	-

Si la tabla se tiene llena hasta la u-ésima fila para encontrar la (u+1)- ésima fila se sigue los siguientes pasos.

si $d_u = 0$ entonces $\sigma^{u+1}(x) = \sigma^{u}(x)$

si $d_u \neq 0$ se busca otra fila anterior a la u-ésima fila, definida como la p-ésima fila, tal que el número $2p-\ell_p$ en la última columna de esa fila, posea el más alto valor posible y que $d_p \neq 0$, entonces se tiene - que:

$$\sigma^{u+1}(x) = \sigma^{u}(x) + \frac{d_u}{d_p} x^{2(u-p)} \sigma^{p}(x) \qquad 3.91$$

en cualquier caso ℓ_{u+1} es el grado de $\sigma^{u+1}(x)$ y la discrepancia para el (u+1) paso se puede calcular utilizando la siguiente expresión.

$$d_{u+1} = S_{2u+3} + \sigma_1^{u+1} S_{2u+2} + \cdots\cdots\cdots\cdots \sigma_{\ell_{u+1}}^{u+1} S_{2u+3-\ell_{u-1}} \qquad 3.92$$

el polinomio $\sigma^{t}(x)$ correspondiente a la última fila es el polinomio buscado $\sigma(x)$ el cual, si tiene un grado mayor que t indica que existen más de t errores y no es posible corregirlos.

Nuevamente si el número de errores en el polinomio recibido r(x) -

es menor que t no es necesario realizar los t pasos de la iteracción para encontrar $\sigma(x)$. Si para d_u, u y la discrepancia de los siguientes $\frac{t-\ell_u-1}{2}$ pasos de la iteracción son ceros, $\sigma^u(x)$ es el polinomio localizador de error y si el número de errores en el polinomio recibido es ν con $\nu \leq t$, solo $\frac{t+\nu}{2}$ pasos son necesarios para encontrar $\sigma(x)$.

El último paso en la decodificación de códigos BCH es encontrar los números localizadores de error, los cuales son los recíprocos de las raíces de $\sigma(x)$. Las raíces de $\sigma(x)$ pueden ser calculadas fácilmente sustituyendo $1, \alpha, \alpha^2, \ldots . \alpha^{m-1}$ en $\sigma(x)$ con $\alpha^n = 1, \alpha^{-\ell} \alpha^{n-\ell}$ y $n = 2^m-1$

Si al evaluar $\sigma(x)$ resulta que $\alpha^{n-\ell}$ es su raíz, $\alpha^{n-\ell}$ es el número localizador del error y el dígito recibido $r_{n-\ell}$ es el dígito erróneo.

El concepto de la decodificación para códigos BCH, resulta complicado con relación a los códigos anteriores. Aparte del almacenamiento de la palabra recibida y de un circuito que trabaje con elementos en $GF(2^m)$, el hardware requerido es proporcional a m, n y t.

"Ha concluído este capítulo con el examen de [illegible] métodos de detección y corrección de [illegible] rios como una forma de mejorar notablemente el rendimiento de estos sistemas binarios". (45).

REFERENCIAS :

1. Gallager R. INFORMATION THEORY AND REALIABLE COMUNICATION, pág. 119.

2. Ash Robert INFORMATION THEORY, pág. 53.

3. Peterson Wesley ERROR CORREGTING CODES. Pág. 30.

4. Shu Lin,Costelo. ERROR CONTROL CODING FUNDAMENTALS AND APLICATION. Pág. 53.

5. R.F. Coates MODERN COMUNICATION SYSTEMS, pág. 219

6. K. Sam Shasmugan DIGITAL AND ANALOG COMUNICATION SYSTEMS.Pág.451.

7. R.F.Coates MODERN COMUNICATION SYSTEMS, Pág. 231.

8. Peterson Wesley ERROR CORREGTING CODES, Pág. 31.

9. K. Sam Shanmugam DIGITAL AND ANALOG COMUNICATION SYSTEMS.Pág.450.

10. Schwartz Misha. TRANSMISION DE INFORMACION MODULACION Y RUIDO, Pág. 561.

11. Mc Eliece R. THE THEORY OF INFORMATION AND CODING, Pág. 138.

12. Shu Lin,Costelo ERROR CONTROL CODING FUNDAMENTALS AND APLICATIONS, Pág. 63.

13. R. F. Coates MODERN COMUNICATION SYSTEMS, Pág. 232.

14. Gallager R. INFORMATION THEORY AND REALIABLE COMUNICATION, Pág. 201.

15. Shu Lin Costelo. ERROR CONTROL CODING FUNDAMENTALS AND APLICA - TIONS, Pág. 67.

16. Idem. Pág. 68.-

17. Gallager R. INFORMATION THEORY AND RFALIABLE COMUNICATION, Pág. 203.

18. Schwartz Nisha. TRANSMISION DE INFORMACION MODULACION Y RUIDO. Pág. 558.

19. Mc Eliece R. THE THEORY OF INFORMATION AND CODING. Pág.140.

20. Ash Robert INFORMATION THEORY. Pág. 104.

21. Shu Lin,Costelo. ERROR CONTROL CODING FUNDAMENTALS AND APLICA - TIONS, Pág. 72.

22. Idem. Pág. 78.

23. Gallager R. INFORMATION THEORY AND RFALIABLE COMUNICATION. Pág. 221.

24. Misha Schwartz. TRANSMISION DE INFORMACION MODULACION Y RUIDO. Pág. 568.

25. Gallager R. INFORMATION THEORY AND REALIABLE COMUNICATION. Pág. 223.

26. Misha Schwartz. TRANSMISION DE INFORMACION MODULACION Y RUIDO. Pág. 569.

27. Peterson Wesley. ERROR CORREGTING CODES. Pág. 39.

28. Shu Lin, Costelo. ERROR CONTROL CODING FUNDAMENTALS AND APLICA- TIONS. Pág. 93.

29. Gallager R. INFORMATION THEORY AND REALIABLE COMUNICATION. Pág, 233.

30. Idem. Pág. 224.

31. Coates R.F.W. MODERN COMUNICATION SYSTEMS. Pág. 226.

32. Sam Shanmugan. DIGITAL AND ANALOG COMUNICATION SYSTEMS .- Pág. 469.

33. Shu Lin, Costelo. ERROR CONIROL CODING FUNDAMENTALS AND APLICATION. Pág. 102.

34. Idem. Pág. 103.

35. Sam Shanmugan. DIGITAL AND ANALOG COMUNICATION SYSTEMS. Pág. 473.

36. Gallager R. INFORMATION THEORY AND REALIABLE COMUNICATION. Pág. 203.

37. Viterbi A. PRINCIPLES OF DIGITAL COMUNICATION AND CODING. Pág. 94.

38. San Shanmugan DIGITAL AND ANALOG COMUNICATIONS SYSTEMS..- Pág. 473.

39. Gallager R. INFORMATION THEORY AND REALIABLE COMUNICATION. Pág. 238.

40. Mc. Eliece R. THE THEORY OF INFORMATION AND CODING. Pág. 170.

41. Peterson Wesley ERROR CORRECTING CODES. Pág. 162.

42. Shu Lin Costelo ERROR CONIROL CODING FUNDAMENTALS AND APLICATIONS. Pág. 147.

43. Idem. Pág. 153.

44. Bec Kenlamp E.R. ALGEBRAIC CODING THEORY. Pág. 178.

45. Misha Schwartz. TRANSMISION DE INFORMACION MODULACION Y RUIDO. Pág. 584.

*** & ***

CAPITULO IV

Una vez terminado el estudio de la parte teórica del control de errores, en lo que tiene que ver a los códigos lineales, vamos a proceder a implementar tres clases de códigos de bloque, considerando, tanto el codificador como el decodificador. Estos tres códigos se denominarán: de Bloque, Cíclico y BCH, los cuales formarán parte de un aparato en el cual se podrá apreciar detenidamente todo el proceso del control de errores que va desde la codificación e introducción de error, hasta la decodificación y la consecuente corrección.

4.1. Consideraciones sobre el Hardware.-

Para la implementación de los códigos, se seleccionó los siguientes valores para n y k: código de bloque (12,4), código cíclico (12,8) y código BCH (15,5).

La decodificación del código BCH se basa en un algoritmo decodificador que necesita de un micropocesador para que su implementación sea sencilla en lo que tiene que ver al número de elementos requeridos.

A pesar de que la longitud escogida para el código de bloque es pequeña, se requiere un circuito combinacional que utiliza muchos elementos discretos para obtener el tipo de error introducido, justificando de esta forma la utilización del microprocesador en la implementación de este tipo de código.

El querer visualizar los resultados más importantes que se obtienen durante el proceso del control de errores para los tres tipos de códigos, involucra la creación de un algoritmo, en el cual esté definidotodo este proceso.

Una vez decidida la utilización del microprocesador debido a la complejidad de los circuitos, se decidió aprovechar al máximo su utilización creando un programa que nos permita realizar el control de errores con códigos cíclicos con diferentes valores de n y k, cuyos codificadores y decodificadores puedan ser implementados con circuitos que utilicen elementos discretos.

4.2. Requerimientos del Control de Errores.-

La visualización del proceso del control de errores abarca los siguientes aspectos:

- Ingreso de la información a codificar.

  El ingreso de datos se lo hará por medio de un teclado en el cual aceptará información decimal que posteriormente será transformada a forma digital. Además deberá tener teclas de control que permitirán comandar el proceso de control de errores.

- Presentación de la información a codificar.

  Para esto se utilizará dos display de ocho segmentos, en los cuales se presentará en forma hexadecimal la información a codificar.

- Presentación del vector de código.

Se proporcionará un máximo de 15 leds, en los cuales se podrá ver la palabra codificada en forma binaria.

- Introducción del error.

Por medio del teclado se tendrá la capacidad de modificar el código vector bit por bit, simulando de esta manera la introducción de errores en la información transmitida. El error se introducirá indicando la posición del bit a modificar.

- Visualización del error introducido.-

En los 15 leds se podrá apreciar el cambio que ocurre en los bits del vector de código, cuando en cualquiera de las posiciones se haya introducido un error, si el error se introduce en los dígitos pertenecientes a la información que se codificó, el resultado erróneo se indicará en dos display en los cuales se presentará el equivalente hexadecimal de la palabra que se codificó pero con el error.

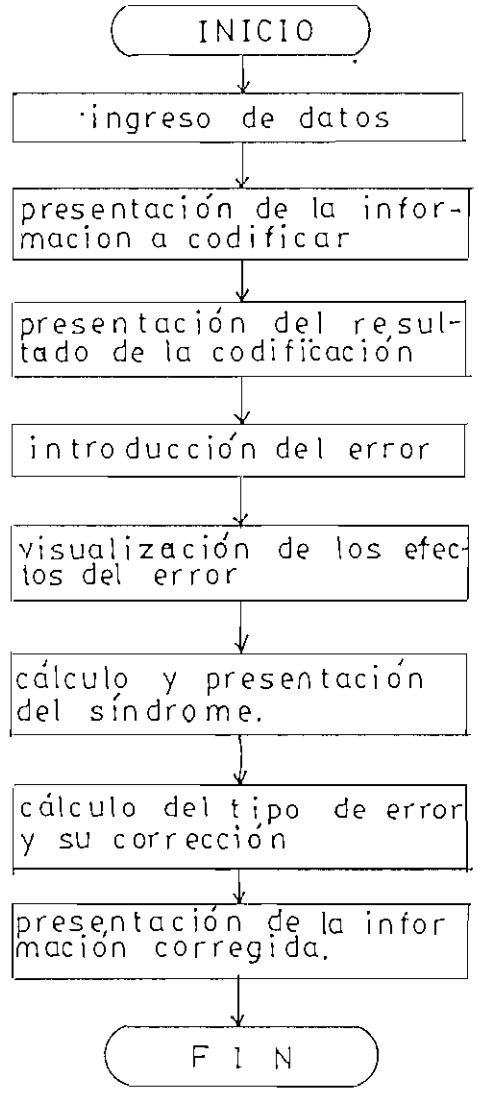
- Presentación del Síndrome.-

Adicionalmente a los 15 leds ya indicados, se incluye 8 leds más, en los cuales se apreciará el síndrome en forma binaria, calculado del vector que se halla representado en forma binaria en los 15 leds.

- Corrección del error.-

El contenido de los 15 leds y de los dos display en los cuales se halla el vector de código erróneo y la información errónea respectivamente, cambiará a sus valores correctos una vez que se ha ejecutado todo el proceso de control de errores.

Definidos estos aspectos se presenta un diagrama de flujo que será aplicado a cualquiera de los códigos cuyo proceso de control de errores se desea visualizar.



4.2.1. Programas que comandan el Control de Errores.-

El programa requerido para llevar a cabo todo el proceso de control de errores necesita de varias subrutinas que facilitan llevar a cabo todos los puntos definidos anteriormente para cada uno de los códigos. A continuación procedemos a explicar lo que realiza cada subrutina y presentaremos en base a esto, su diagrama del flujo.

Las primeras instrucciones al ejecutarse el programa es el borrado de todas las localidades de la RAM, que se utulizarán a lo largo de todo el proceso.

Es necesario especificar el tipo de código con el cual se va a trabajar, para lo cual es necesario la creación de una tecla de control representada por [S] que nos permitirá escoger el código. Con esta información la microprocesadora definirá los parámetros tales como, longitud del código, longitud de la información a codificar, posición máxima del error que puede introducir, valor máximo de la palabra a codificar, etc.

Si el código seleccionado es el cíclico, el usuario debe ingresar en forma decimal los valores de n y k del código.

El diagrama del flujo correspondiente a esta parte, está dado en forma conjunta con todos los diagramas de flujo correspondientes a las subrutinas siguientes:

- REALI: Una vez seleccionado el código con el cual se va a trabajar, se procede a llamar a esta subrutina cuya función permite visualizar todo el proceso de control de errores. Se hace necesario la creación de una tecla representada por [C] la cual permitirá seguir paso a paso todo el proceso.

Una tecla de control representada por [F] se hace necesaria en el momento que se debe aclarar a la microprocesadora que la información introducida pertenece a la posición en el vector de código en el cual se va a introducir un error.

Una tecla de control [I] nos permite ingresar un nuevo dato a codificar utilizando el código ya seleccionado. Se puede hacer uso de esta tecla en cualquier momento.

Esta subrutina será la que efectúe el control de errores para cualesquiera de los tres códigos efectuando la codificación, decodificación, introducción del mensaje y del error, presentación en los indicadores luminosos, etc. Para realizar todo esto se utilizan varias subrutinas que se detallan a continuación.

- SCAN : Esta subrutina al ser llamada tiene por objeto presentar en los leds y displays la información que está almacenada en ciertas localidades de la RAM, denominadas Buffer del Display, utilizando la técnica de barrido, la cual abarca también las teclas para determinar si se ingresa o no información al ser presionada alguna de ellas.

La información recogida, se procesa para darle un equivalente binario. Se está en esta subrutina a menos que se introduzca información de control, a codificar o de error.

- TREN : Nos permite expresar en forma binaria cualquier información introducida en forma decimal al presionar más de una tecla numérica. Se acepta las siguientes teclas de control: [C], [I], [F].

- BORA : Borra el contenido del buffer del display en el cual está la información que enciende los indicadores luminosos y la información almacenada durante el proceso de control de errores.

- GBLO : Permite almacenar un bit de cada byte en una sola localidad de memoria en forma ordenada (ascendente) empezando por el bit menos significativo, generalmente el byte es de la información a codificar.

- BUFL : A partir de los cuatro bits menos significativos de un byte, por medio de esta subrutina encontramos la información binaria que nos permitirá encender adecuadamente los correspondientes segmentos de los display para indicar en forma hexadecimal el contenido binario.

- BUFH : Esta subrutina es similar a la anterior, con la particularidad de que escoge los cuatro bits más significativos.

- CODI : Dependiendo de cual es el código escogido se llama a la correspondiente subrutina que realiza la codificación, el resul-

-- tado se lo guarda en un bloque de memoria que almacena la palabra codificada cualquiera sea el código utilizado.

- TRAN :- Traspasa la información codificada de un bloque de memoria - original, hacia otro bloque de memoria en el cual se introducirán los errores cambiando el contenido de alguna localidad de memoria en la cual esta almacenada un bit del código vector.

- MODI : Esta subrutina modifica el contenido de la palabra codificada cuando se ha introducido un error indicando la posición - que se quiere cambiar.

- SIND : Selecciona la subrutina adecuada que calculara el síndrome del vector con o sin error de acuerdo al código con el que se está trabajando.

- PRESS : Esta subrutina procesa la información recibida para su presentación en los indicadores luminosos, además presenta en los dos displays la parte de la palabra código que corresponde al mensaje con o sin error.

- ERR7 : Permite presentar en los display una señal que indica que - se ha cometido un error en la última operación efectuada, - almacena la información que estaba antes del error.

- ARRE : Permite colocar en un solo byte los bits que estan almacenados en un bloque de ocho localidades de memoria.

4.3. Implementación del Sistema.-

Un sistema capaz de realizar todos los requerimientos expuestos anteriormente esta descrito por el diagrama de bloques dado por la figura 4.1. Sus componentes que se describen a continuación se han escogido tomando como base su disponibilidad y precio.

4.3.1. Microprocesador.-

Se utiliza la Z-80A, ya que en el laboratorio existen aparatos que nos permiten trabajar adecuadamente con él.

4.3.2. Decodificador de Direcciones.-

Como sólo se utiliza un periférico programable (PPI), este se lo puede direccionar directamente utilizando el IORQ. Por lo que para direccionar la RAM y la PROM, se necesita de un decodificador tal como el 74LS138 (decoder 2 a 4), el cual tiene como entradas a A_{12}, $\overline{MREQ}$ y $\overline{M1}$ cuya información habilita a una memoria a la vez. Se direcciona desde la localidad 0000H hasta la 1400H.

4.3.3. Generador de Reloj y Reset.-

Se hará trabajar al microprocesador a una frecuencia de 2 MHz. para lo cual se utilizara un cristal de 4 MHz. La única restricción para la señal de reloj esta dada por el tipo de microprocesador utilizado.

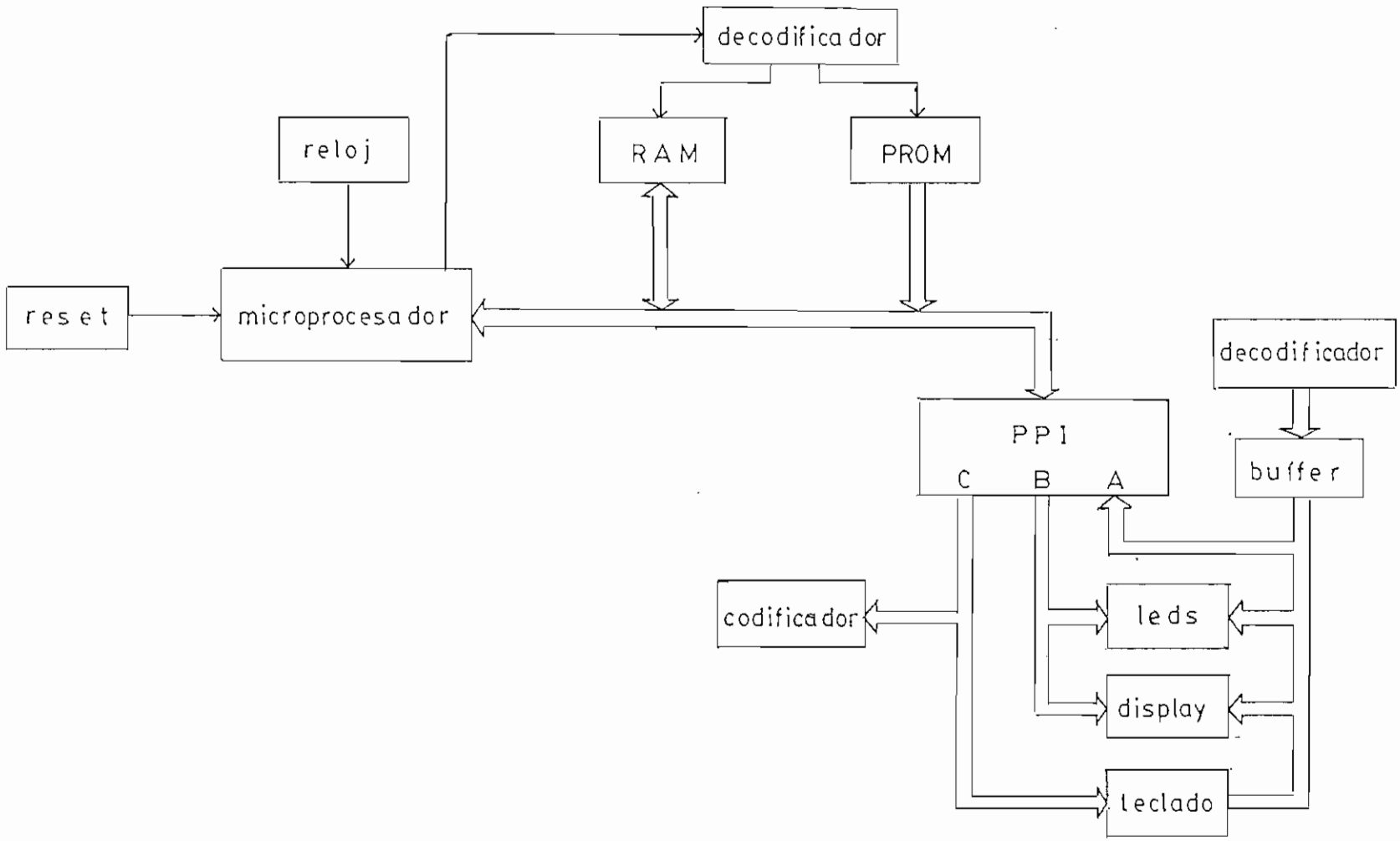
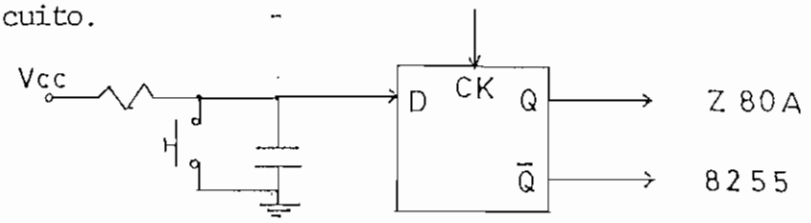


FIGURA 4.1 Diagrama de bloques del sistema.

La señal de reset será proporcionada por medio de una tecla que - cortocircuita un condensador, tal como se aprecia en el siguiente cir - cuito.



4.3.4. Interfaz Programable.-

El integrado utilizado será el 8255 que resulta muy apropiado - para manejar muchos periféricos como son teclado, indicadores luminosos, utilizando la técnica de barrido.

Así tenemos que el pórtico denominado A posee una alta capacidad de corriente en relación a los pórticos B y C. De esta forma se - convierte en el pórtico más adecuado para entregar la información digital que permita encender o apagar a los indicadores luminosos.

El pórtico B nos servirá para habilitar uno por uno, cada indicador luminoso, manejando cada uno de ellos un máximo de 8 bits de - información

PB0	DISPLAY	mensaje a codificar
PB1	DISPLAY	
PB2	DISPLAY	mensaje con o sin error
PB3	DISPLAY	
PB4	8 LEDS	vector de código

PB5	7 LEDS	vector de código
PB6	7 LEDS	síndrome
PB7	3 LEDS	tipo de código

El pórtico C nos servirá para habilitar el teclado y para proporcionar la información necesaria para que las tarjetas codificadoras y-decodificadores puedan funcionar

PC0	teclado, primera columna
PC1	segunda columna
PC2	habilitador del buffer
PC3	entrada del decodificador
PC4	entrada del codificador
PC5	señal de selección del multiplexer
PC6	señal de clear
PC5	señal de reloj

El 8255 será direccionado como una localidad de memoria cuya dirección estará dada por A_0 y A_1

A_1	A_0	
0	0	pórtico A
0	1	pórtico B
1	0	pórtico C
1	1	programación del 8255.

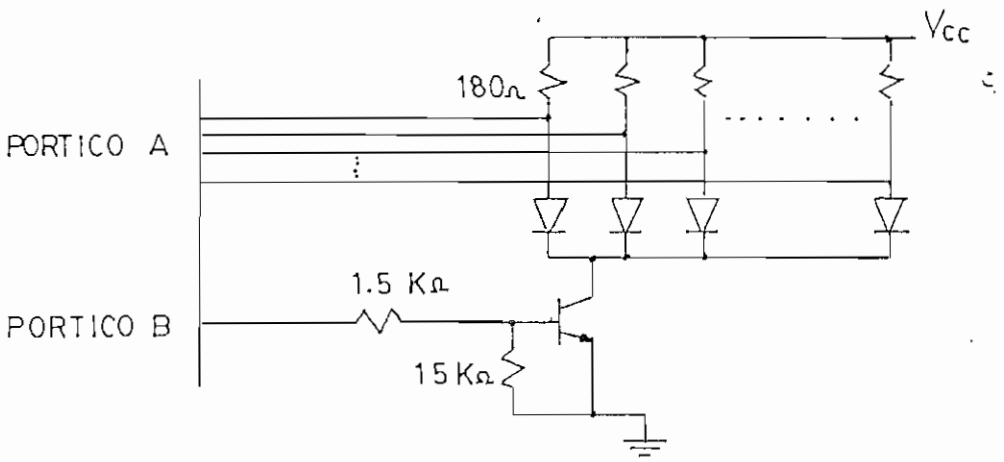
4.3.5. Display y Leds.-

Los display son de siete segmentos y catodo común, los leds estan ordenados en conjuntos de ocho elementos con los catodos conectados en forma común, de tal manera que la habilitación sea similar a la de los displays utilizando la técnica de barrido.

Los leds nos proporcionarán indicación digital de 1L y 0L para lo cual los leds se encenderán o apagarán respectivamente. Los display nos presentarán información hexadecimal.

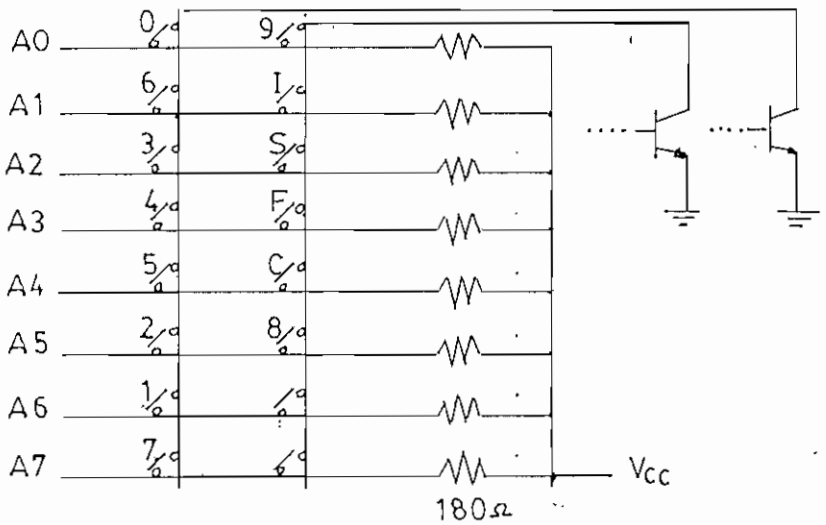
Información en el pórtico A	segmentos encendidos	información en en el pórtico A	segmentos encendidos
3F		7F	
06		6F	
5B		77	
4F		7C	
66		39	
6D		55	
7D		79	
07		71	

El circuito que manejará a los indicadores luminosos tiene la siguiente estructura:



4.3.6. Teclado.-

El teclado está ordenado en forma de una matriz 2 X 8. Cuando no se presiona una tecla, el contenido del pórtico A es FF, cuando está presionada y está habilitada la columna a la cual pertenece, la información en dicho pórtico cambia. El microprocesador lee esta información y con ayuda de la columna habilitada lo procesa.



La información que entra en el pórtico A cuando se presiona una tecla es la siguiente:

teclas	información
0,9	FE

6 , I	FD
3 , S	FB
4 , F	F7
5 , C	EF
2 , 8	DF
1	BF
7	7F

4.3.7. PROM.-

Se utilizará la 2764 la cual es una PROM 8K×8 bits. Esta capacidad excede los requerimientos de memoria, por lo que se utilizará la mitad de su capacidad total. Se direcciona desde la 0000H hasta la 0FFFH.

La distribución de sus localidades de memoria es la siguiente:

0000	programas
0BFF	
0E00	tabla de símbolos para simular el circuito combinacio-
0EFF	nal que asocia un tipo de error al síndrome.
0F00	tabla que contiene los elementos de $GF(2^4)$
0F0F	
0F10	tabla de patrones del display
0F1F	
0F20	tabla de patrones para identificar tecla presionada
0F2F	

0F30	otros patrones
0FsF	

4.3.8. RAM.-

La RAM utilizada será el integrado HM 6116 que tiene una capacidad de 2K X 8 bits y se la direcciona desde 1000H hasta 1400H.- Existen bloques de memoria que serán utilizados por los tres códigos para almacenar resultados comunes que se indican a continuación.

1000	palabras a codificar
1007	
1010	palabra codificada
101E	
1020	palabra codificada con error
102E	
1031	síndrome
1038	
1039	variables utilizadas por el código BCH
105F	
1060	buffer del display
1067	
1068	otras variables
10E0	

4.3.9. Tarjetas Codificadoras y Decodificadoras.-

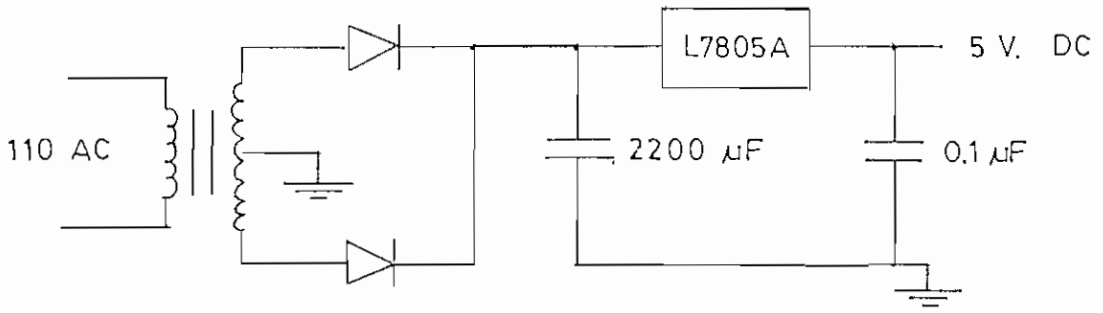
Las señales de control necesarias para el buen funcionamiento de estas tarjetas o circuitos, así como el conjunto de bits a codificar y decodificar serán proporcionados en forma directa por el pórtico C, según lo indicado anteriormente.

Los resultados que se quieran obtener de las tarjetas, serán introducidos al microprocesador a través del pórtico A del 8255. Como el pórtico A es bidireccional es necesario colocar entre las salidas de las tarjetas y el bus de datos de este pórtico, un buffer tres estados que será habilitado por una línea del pórtico C, sólo cuando se desee sacar información de las tarjetas. Los diagramas circuitales y características de las tarjetas se verán en la sección correspondiente de códigos cíclicos.

4.3.10. Fuente de Alimentación.-

Todos los elementos utilizados en la implementación del circuito dado por el diagrama de bloques de la figura 4.1 operan con 5 voltios. En las pruebas efectuadas, inclusive con las tarjetas exteriores al sistema, el consumo de corriente fue de 600 mA, por lo que un regulador semiconductor de 5 Voltios y 1 amperio resulta suficiente. El diagrama circuital de la fuente es el siguiente:

DU



4.4. Código de Bloque.-

El código seleccionado para su implementación proviene del acortamiento del código de bloque (15,7) cuya matriz generadora es la siguiente:

$$G = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

A partir de esta matriz nosotros podemos obtener ciertas características como son: el ancho mínimo es igual a cinco, sus filas son-lienalmente independientes, su estructura es sistemática, el número de columnas nos da la longitud del vector de código y el número de filas representa el número de dígitos de la palabra a codificar.

Como el ancho mínimo es igual a cinco, la distancia mínima también será igual a cinco, por lo que la capacidad de corrección viene dada por:

$$t = \frac{dmin - 1}{2} = \frac{5 - 1}{2} = 2$$

de aquí, el código puede corregir hasta dos errores que ocurran simultáneamente.

A partir de esta matriz se puede obtener la matriz generadora de un código bloque lineal (12.4). Esto se logra por medio del acortamiento de la matriz, eliminando filas y columnas sin que por ello se alteren las propiedades originales del código bloque inicial. De esta forma se obtiene la matriz que nos genera un código de bloque (12.4).

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

4.4.1. Codificador.-

Itilizando los elementos de esta matriz y con la ayuda del diagrama de la figura 3.1, presentamos un circuito codificador que se muestra en la figura 4.2.

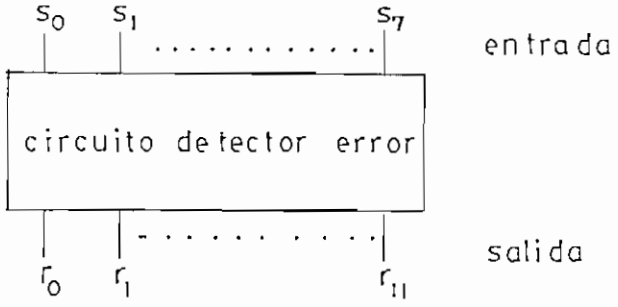
A partir de la matriz generadora G podemos obtener la matriz verificadora de paridad H para el código bloque (12,4).

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

4.4.2. Decodificador.-

El circuito que nos permita calcular el síndrome puede ser obtenido a partir de los elementos de la matriz $\mathbb{H}$ o $\mathbb{G}$ conjuntamente con el esquema dado por la figura 3.2. El circuito que calcula el síndrome-lo presentamos en la figura 4.3.

El síndrome calculado está univocamente relacionado con un sólo tipo de error, el cual debemos encontrar para realizar la corrección. Para relacionar un síndrome con un tipo de error definido es necesario la utilización de un circuito combinacional en el cual, como entradas se tiene a los elementos del síndrome y como salidas a los elementos del tipo de error:



La tabla de verdad de este circuito combinacional está dada en la tabla 4.1.

Por simple inspección podemos darnos cuenta que implementar este circuito combinacional con elementos discretos resulta impráctico debido a la gran cantidad de elementos que se necesita. En cambio con-la utilización del microprocesador, se simplifica grandemente.

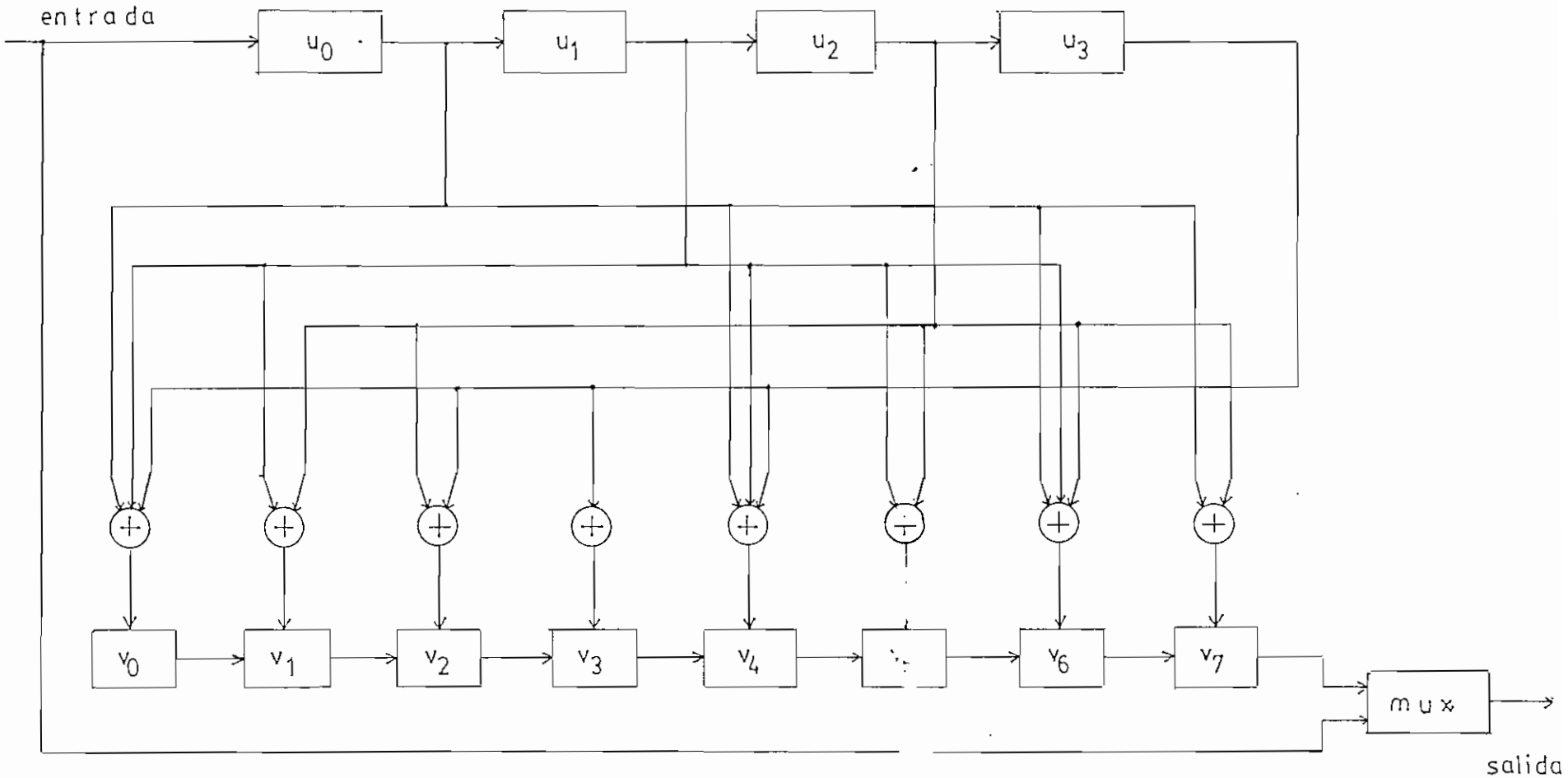


FIGURA 4.2 Codificador del codigo de bloque.

4.4.3. Algoritmos.-

Todo lo expuesto anteriormente nos servirá para crear los programas que nos permitan utilizar a la microprocesadora para implementar los codificadores y decodificadores. Desarrollando la expresión- $\bar{v} = \bar{u} \cdot G$ se obtienen las siguientes ecuaciones:

si $\bar{u} = (u_0, u_1, u_2, u_3)$

$v_0 = u_0 + u_1 + u_3$ $\quad$ $v_6 = u_0 + u_1 + u_2$

$v_1 = u_1 + u_2$ $\quad$ $v_7 = u_0 + u_2$

$v_2 = u_2 + u_3$ $\quad$ $v_8 = u_0$

$v_3 = u_3$ $\quad$ $v_9 = u_1$

$v_4 = u_0 + u_1 + u_3$ $\quad$ $v_{10} = u_2$

$v_5 = u_1 + u_2$ $\quad$ $v_{11} = u_3$

Como se puede apreciar, realizando la suma módulo 2 entre las palabras a codificar, se obtiene el vector de código. El programa - que realiza la codificación utilizando estas expresiones, simula el circuito dado por la figura 4.2. Este programa saca la información- de la palabra a codificar de localidades de memoria cada una de las cuales almacena un bit de información. En este programa llamado CODB se realiza todas las operaciones indicadas anteriormente y el resultado se lo almacena en un bloque de memoria asignado para la pala - bra codificada.

El cálculo del síndrome parte de un vector de código, el cual-

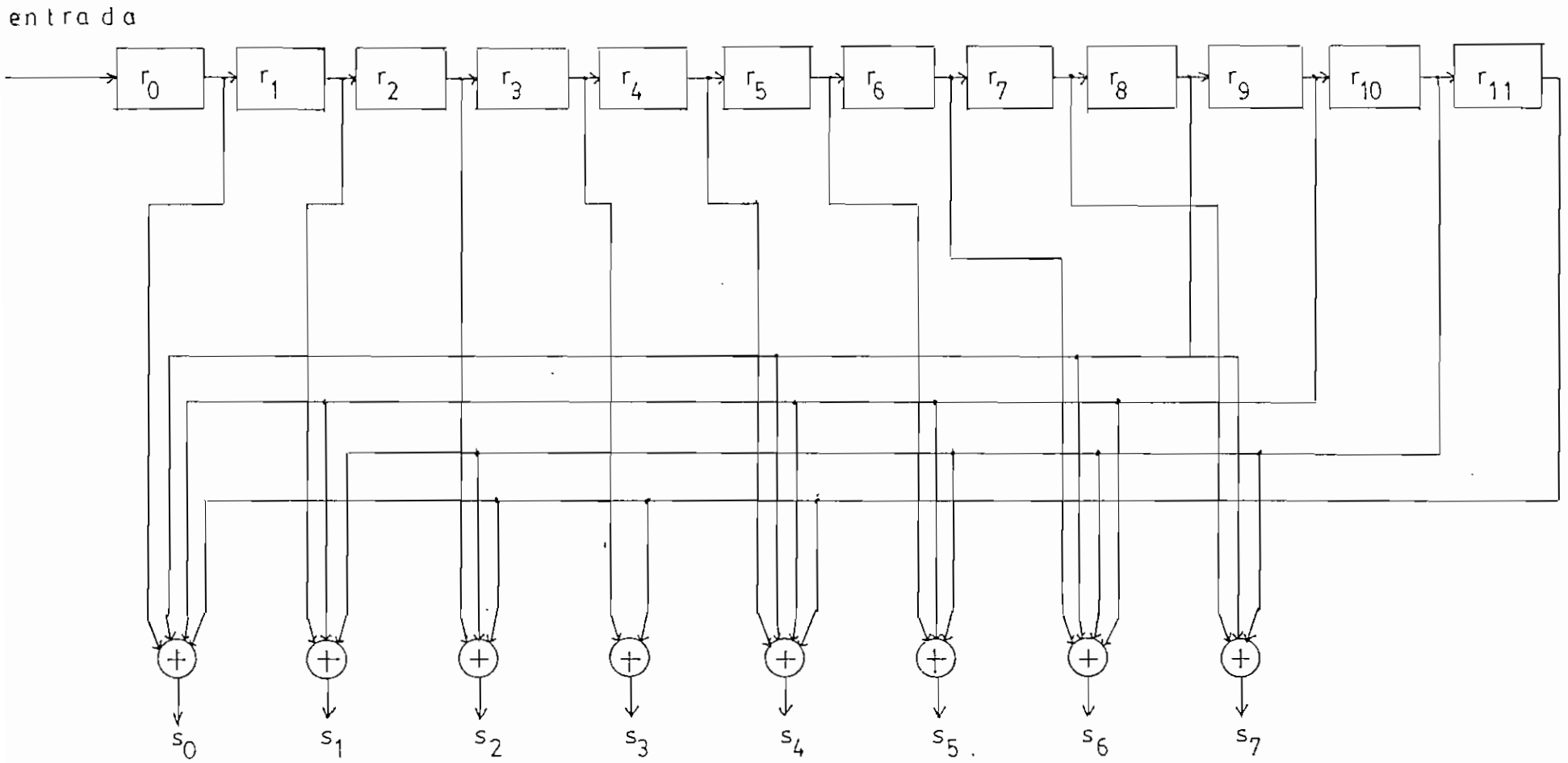


FIGURA 4.3 Circuito que calcula el síndrome

puede estar con o sin error, este vector de código junto con la matriz H nos proporciona expresiones que nos permitirán calcular cada elemento del síndrome.

Partiendo de que $\bar{s} = \bar{r}\ H^T$ con $r = (r_0, r_1, \ldots r_{11})$, se obtienen las siguientes expresiones.

$$s_0 = r_0 + r_8 + r_9 + r_{11} \qquad s_4 = r_4 + r_8 + r_9 + r_{11}$$

$$s_1 = r_1 + r_9 + r_{10} \qquad s_5 = r_5 + r_9 + r_{10}$$

$$s_2 = r_2 + r_{10} + r_{11} \qquad s_6 = r_6 + r_8 + r_9 + r_{10}$$

$$s_3 = r_3 + r_{11} \qquad s_7 = r_7 + r_8 + r_{10}$$

con $\bar{s} = (s_0, s_1 \ldots\ldots s_2)$

nuevamente estas ecuaciones nos ayudarán a simular el circuito dado por la figura 4.3, que calcula el síndrome.

El programa que realiza el cálculo de los elementos del síndrome se lo llama SIBL, el cual toma el vector recibido del bloque de memoria correspondiente colocando los resultados en otro bloque de memoria donde se almacena el síndrome.

La parte crítica en el circuito codificador, por su complejidad, es el circuito combinacional que asocia a cada síndrome un tipo de error específico. Para realizar esto se crea un programa que utiliza los ocho bits que conforman el síndrome para direccionar una localidad de memoria en la cual se almacena las posiciones del vector de código, en las cuales el tipo de error cambia su valor, o las posiciones en las cuales el tipo de error tiene sus componentes iguales a uno.

El contenido de estas localidades de memoria se basa en la tabla 4.1.

Una vez que se ha determinado las posiciones erróneas del vector de código recibido, se procede a corregirlas. El programa que determina las posiciones erróneas y la posterior corrección del vector de código se lo llama CRRB.

La complejidad de los codificadores y de los decodificadores para este tipo de códigos dependerá principalmente de la longitud del código y de su capacidad para corregir errores.

Cada uno de estos subprogramas serán utilizados por el programa-principal si el código seleccionado es el bloque. Como ya se mencionó los datos a procesar y los resultados estarán en bloques de memoria, de la RAM, comunes a estos tres códigos.

4.5. Códigos BCH.-

El código BCH a implementar utiliza los siguientes parámetros, n = 15 y k = 5. La matriz generadora de este código es la siguiente:

$$G = \begin{vmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

A partir de esta matriz podemos determinar que el ancho mínimo es igual a siete y por tanto la distancia mínima también es igual a siete. Su capacidad para corregir errores viene dada por:

$$t = \frac{dmin - 1}{2} = \frac{7 - 1}{2} = 3$$

Este código puede corregir hasta tres errores que ocurran simultáneamente en el vector de código.

Los procesos de codificación de este código se basan en las propiedades matemáticas de los elementos de GF (2^4), por lo que a continuación se da una lista de los elementos que conforman este campo y su representación:

representación exponencial	representación polinomial	representación binaria
1	1	1 0 0 0
α	α	0 1 0 0
α^2	α^2	0 0 1 0
α^3	α^3	0 0 0 1
α^4	$1 + \alpha$	1 1 0 0
α^5	$\alpha + \alpha^2$	0 1 1 0
α^6	$\alpha^2 + \alpha^3$	0 0 1 1
α^7	$1 + \alpha + \alpha^3$	1 1 0 1
α^8	$1 + \alpha^2$	1 0 1 0
α^9	$\alpha + \alpha^3$	0 1 0 1
α^{10}	$1 + \alpha + \alpha^2$	1 1 1 0
α^{11}	$\alpha + \alpha^2 + \alpha^3$	0 1 1 1
α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$	1 1 1 1
α^{13}	$1 + \alpha^2 + \alpha^3$	1 0 1 1
α^{14}	$1 + \alpha^3$	1 0 0 1

Cada elemento de GF(2^4) tiene su polinomio mínimo. Estos son:

elementos	polinomio mínimo
1	$x + 1$
$\alpha, \alpha^2, \alpha^4, \alpha^8$	$x^4 + x + 1$
$\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$	$x^4 + x^3 + x^2 + x + 1$
α^5, α^{10}	$x^2 + x + 1$

$\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$ $\qquad\qquad$ $x^4 + x^3 + 1$

Con estas definiciones ya podemos encontrar el polinomio generador g(x), el cual está definido en términos de los polinomios mínimos $\emptyset_i(x)$ de α^i . Así tenemos que:

$$g(x) = MCM(\emptyset_1(x), \emptyset_3(x), \ldots\ldots \emptyset_{2t}(x))$$

$$g(x) = MCM(\emptyset_1(x), \emptyset_3(x), \emptyset_5(x))$$

$$g(x) = (1 + x + x^4)(1 + x + x^2 + x^3 + x^4)(1 + x + x^2)$$

$$g(x) = 1 + x + x^2 + x^5 + x^8 + x^{10}$$

4.5.1. Codificador.-

Como los códigos BCH son códigos cíclicos, su codificación se la puede realizar fácilmente utilizando los polinomios g(x) o h(x) para crear el circuito codificador. Calculando el polinomio h(x):

$$h(x) = \frac{x^n + 1}{g(x)} = 1 + x + x^3 + x^5$$

El circuito codificador a partir de h(x), tiene menos elementos que el circuito proporcionado a por g(x). El circuito en base a h(x) - está dado en la figura 4.4.

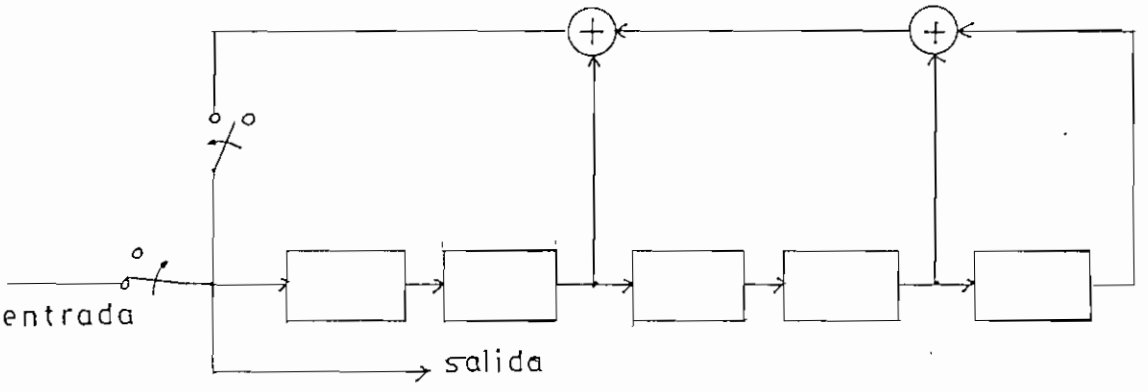


FIGURA 4.4 Codificador del código BCH (15,5)

Como estamos utilizando un microprocesador, vamos a simular el funcionamiento de este circuito con un programa llamado CORC, el cual requiere de los bits a codificar, que serán tomados del bloque de memoria correspondiente en donde se almacena el mensaje a codificar.

Es muy interesante en este momento proceder a construir la matriz H para este código .En base a los elementos de $GF(2^4)$ y tomando la estructura dada por 3.77 se obtiene la siguiente matriz.

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} & \alpha^{14} \\ 1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^0 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^0 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} \\ 1 & \alpha^5 & \alpha^{10} & \alpha^0 & \alpha^5 & \alpha^{10} & \alpha^0 & \alpha^5 & \alpha^{10} & \alpha^0 & \alpha^5 & \alpha^{10} & \alpha^0 & \alpha^5 & \alpha^{10} \end{bmatrix}$$

Con esta matriz, nosotros podremos saber si el vector recibido es un vector de código realizando el producto $\bar{v} \cdot H^T$.

4.5.2. Decodificación y Algoritmo Decodificador.-

La decodificación de estos códigos correctores de errores múltiples, consta de tres pasos; cálculo del síndrome, algoritmo para encontrar $\sigma(x)$ y por último la corrección de los errores.

Todos estos pasos se realizan utilizando operaciones de suma y división y multiplicación entre elementos de GF (2^4), para lo cual es necesario implementar algoritmos que nos permitan realizar estas operaciones.

La suma entre dos elementos de $GF(2^4)$ es simplemente realizar la operación OR exclusiva entre los bits que representan a los dos elementos. Para la multiplicación y división se parte de las siguientes expresiones para la realización de los programas.

multiplicar α^i por α^j $\qquad \alpha^i \alpha^j = \alpha^{i+j}$

dividir α^i para α^j $\qquad \frac{\alpha^i}{\alpha^j} = \alpha^i \alpha^{15-j}$

Los programas que realizan estas operaciones están dados por XPON, OPE, DIVI, y MUL, los cuales se dan conjuntamente con los demás.

El cálculo del síndrome se realiza evaluando el polinomio recibido r(x), con los elementos de $GF(2^4)$. Para el caso del código (15,5) el polinomio r(x) tiene la siguiente forma:

$$r(x) = r_0 + r_1 x + r_2 x^2 + \cdots r_{14} x^{14}$$

la evaluación de este polinomio se lo hace con los siguientes elementos: $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6$ de tal forma que si $\vec{s} = (s_1, s_2, s_3, s_4, s_5, s_6)$ entonces $\qquad s_i = r(\alpha^i) \qquad 1 \leq i \leq 6$

la expresión base para crear el programa que nos permita evaluar r(x) es la siguiente:

$$r(\alpha^i) = (\cdots\cdots(((r_{14})\alpha^i + r_{13})\alpha^i + r_{12})\alpha^i \cdots\cdots)\alpha^i + r_0$$

De esta forma se obtiene los elementos que pertenecen al síndrome. El programa que realiza este cálculo lo llama SIBC.

Una vez calculado el síndrome y todos sus elementos están ya almacenados en el bloque de memoria correspondiente, el siguiente paso es el cálculo de los coeficientes de $\sigma(x)$. Para realizar este cálculo

es necesario desarrollar el algoritmo dado en la figura 3.13 conjuntamente con las expresiones 3.91 y 3.92.

$$do = 1 \qquad \Gamma^{0}(x) = 1$$

$$\Gamma^{1}(x) = \Gamma^{0}(x) + \frac{do}{d_{-1/2}} x^{2(0-(-1/2))} \Gamma^{-1}(x)$$

$$\Gamma^{1}(x) = 1 + S_1 x \; 1 = 1 + S_1 x$$

$$d_1 = S_3 + \Gamma_1^{1} \cdot S_2 + \Gamma_2^{1} \cdot S_1$$

$$d_1 = S_3 + S_2 S_1$$

$$\Gamma^{2}(x) = \Gamma^{1}(x) + \frac{d_1}{do} \cdot x^{2(1-0)} \cdot \Gamma^{0}(x)$$

$$\Gamma^{2}(x) = 1 + S_1 x + \frac{d_1}{S_1} x^2$$

$$d2 = S_5 + \Gamma_1^{2} S_4 + \Gamma_2^{2} S_3 + \Gamma_3^{2} S_2$$

$$d2 = S_5 + S_1 S_4 + \frac{d}{S_1} S_3$$

$$\Gamma^{3}(x) = \Gamma^{2}(x) + \frac{d2}{d1} x^{2(2-1)} \Gamma^{1}(x)$$

$$\Gamma^{3}(x) = 1 + S_1 x + \frac{d1}{S_1} x^2 + \frac{d2}{d1} x^2 (1 + S_1 x)$$

$$\Gamma^{3}(x) = 1 + S_1 x + \left[\frac{d1}{S_1} + \frac{d2}{d1}\right] x^2 + \frac{d2}{d1} S_1 x^3$$

$$\Gamma_0 = 1$$

$$\Gamma_1 = S_1$$

$$\Gamma_2 = \frac{d1}{S_1} + \frac{d2}{d1}$$

$$\Gamma_3 = \frac{d2}{d1} S_1$$

Hay que tomar en cuenta que los elementos del síndrome que han sido calculados son elementos de $GF(2^4)$ que pueden ser representados por cuatro bits. Por esta razón en los indicadores (LEDS) proporcionados para enseñar el síndrome en forma binaria, aparecerá para este tipo de código un solo elemento del síndrome, el cual podrá ser escogido por el usuario con la ayuda de la tecla [S] .

Una vez calculados los coeficientes de $\sigma(x)$ se puede encontrar las raíces de este polinomio que son necesarias para determinar los dígitos erróneos en el vector recibido.

Las raíces se encuentran evaluando $\sigma(x)$ con todos los elementos de $GF(2^4)$, de tal forma que α^i es raíz si $\sigma(\alpha^i) = 0$.

Con $\sigma(x)$ de la siguiente forma;

$$\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \sigma_3 x^3$$

el programa que calcula la posición de los dígitos erróneos se basa en la siguiente expresión:

$$1 \leq i \leq 14 \qquad \sigma(\alpha^i) = (((\sigma_3)\alpha^i + \sigma_2)\alpha^i + \sigma_1)\alpha^i + \sigma_0$$

Una vez encontrado la raíz α^i la posición errónea del vector recibido es 15 - i . De esta forma ya podemos corregir el error.

El programa que en conjunto calcula los coeficientes de $\sigma(x)$, las raíces de $\sigma(x)$ y la corrección del error se lo llama CRBC.

4.6. Códigos Cíclicos.-

El programa principal implementado para poder seguir paso a paso

el proceso de control de errores, permite para esta clase de códigos, probar varios codificadores y decodificadores que se pueden implementar en base a diversos valores de n y k.

Debido a que el sistema nos permite visualizar un vector de código de hasta 15 dígitos, y los displays nos permiten enseñar información hexadecimal correspondiente a ocho dígitos, los parámetros límites de n y k son respectivamente quince y ocho. Esto realmente no es una limitación ya que cualquier código cíclico puede ser acortado para que sus parámetros cumplan o estén dentro de estos valores límites.

Considerando a un código cíclico (15,11) cuyo polinomio generador es $g(x) = 1 + X + X^4$, podemos presentar la siguiente matriz generadora en forma sistemática.

$$G = \begin{vmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

Observando la matriz podemos ver que la distancia mínima de este código es igual a tres, por lo que su capacidad de corregir errores viene dada por:

$$t = \frac{dmin - 1}{2} = \frac{3 - 1}{2} = 1$$

con lo que resulta que este código es capaz de corregir máximo un error. lo cual es una característica de este tipo de código.

Este código es susceptible de ser acortado en tres elementos de tal manera de obtener un código cíclico acortado (12,8). La matriz generadora de este código se obtiene fácilmente, quitando tres filas y tres columnas de la matriz presentada anteriormente, de esta forma se obtiene una matriz (12,8).

$$G = \begin{vmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

4.6.1. Codificador.-

Se puede observar que el valor de la distancia mínima no ha cambiado y que se conserva las características del código original. Además, el polinomio generador es el mismo con lo cual el circuito codificador para ambos códigos no cambia.

A partir del diagrama de bloques dado por 3.4 se puede obtener el siguiente codificador para este código acortado.

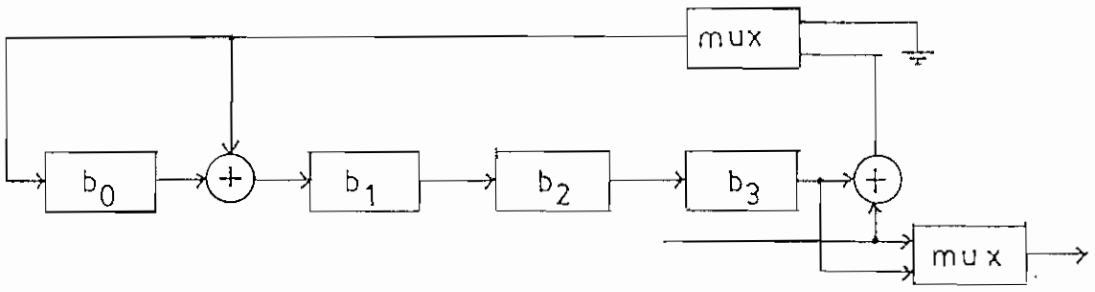


figura 4.5 Circuito codificador del código cíclico (12,8).

4.6.2. Decodificador.-

Acortar un código significa que los bits más altos de la palabra a codificar son siempre cero. A partir de esto podemos darnos cuenta - que no es necesario modificar el circuito codificador del código original, ya que cuando han entrado todos los bits correspondientes al mensaje, el mux 2 pone OL en su salida.

Para el circuito decodificador, los $\ell = 3$ desplazamientos ex - tras que realiza el circuito dado por la figura 3.6 pueden ser eliminados modificando el circuito que calcula el síndrome.

El nuevo circuito para obtener el síndrome se presenta en la figura 4.6 el cual, para su obtención necesita del cálculo del polinomio $p(x)$ utilizando la siguiente expresión ya presentada.

$$p(x) = x^{n-k-\ell} - a(x) \cdot g(x)$$

De esta forma dividiendo x^7 para $g(x) = 1 + X + X^4$ se obtiene el residuo $p(x) = 1 + x + x^3$ El circuito que calcula el síndrome eliminando los $\ell = 3$ desplazamientos extras esta dado en la figura 4.6.

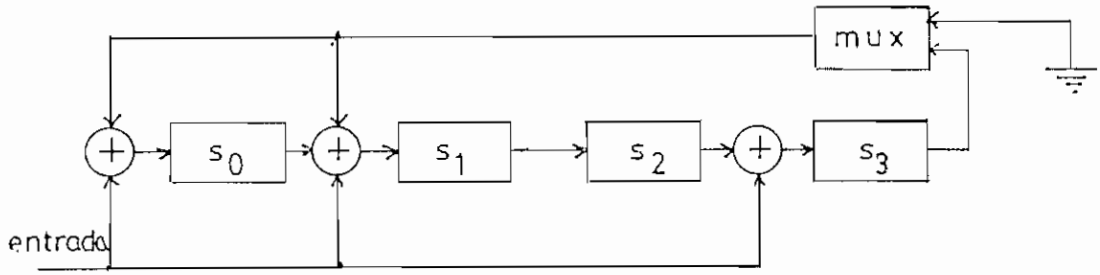


figura 4.6 circuito modificado para calcular el síndrome.

Con este circuito que calcula el síndrome, en la figura 4.7. presentamos el circuito decodificador general.

Tanto el circuito codificador como el decodificador nedesitan de ciertas señales de control para su correcto funcionamiento. Así tenemos la señal de reloj CK que comanda el desplazamiento serial de los dígitos a través de los registros y del canal de información.

Una señal de borrado CL que nos permite inicializar el contenido de los registros con un valor igual a cero. Una señal de selección SE que determina el funcionamiento de los multiplexers.

Además necesitan de los dígitos a codificar y decodificar para - luego de su procesamiento proporcionar a la microprocesadora los datos codificados, decodificados y el síndrome.

Estos circuitos deben trabajar a una frecuencia algo menor de la frecuencia de trabajo de la microprocesadora, es por esto que los integrados deben ser LS. Así para la implementación de estos circuitos se- utilizaron los siguientes integrados.

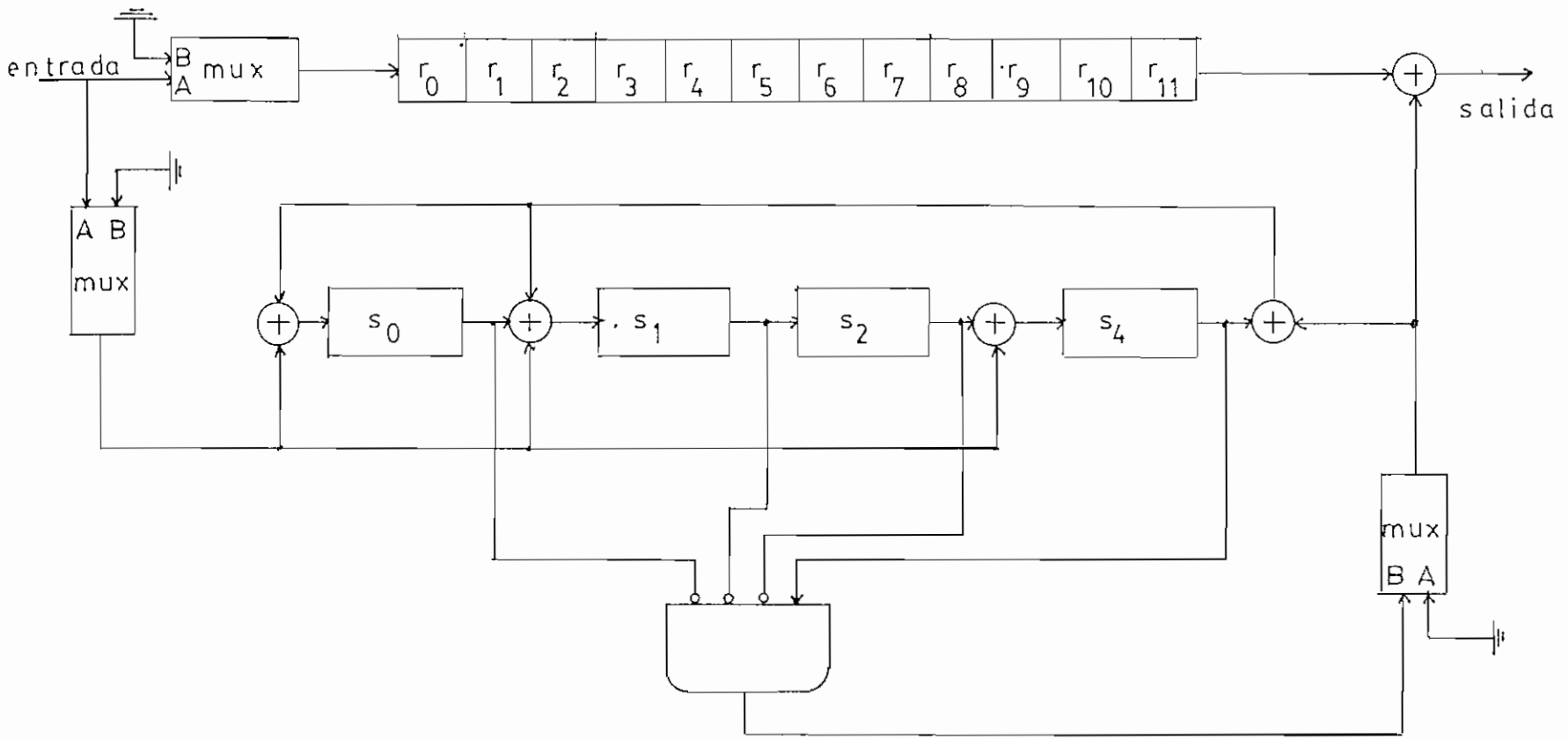


FIGURA 4.7 Decodificador del codigo cíclico (12,8)

74LS157	multiplexers 2 a 1
74LS86	OR exclusivo
74LS175	cuatro flip-flop tipo D
74LS174	seis flip-flop tipo D
74LS21	dos AND de cuatro entradas

Los diagramas circuitales del decodificador y decodificador están presentados en las figuras 4.8 y 4.10 respectivamente. Además, se incluye la ubicación de los integrados en las tarjetas y la ubicación de las señales en los conectores.

Los algoritmos de codificación y decodificación proporcionan el orden adecuado a la microprocesadora para que envíe las señales de control y de datos hacia las tarjetas que contiene los respectivos circuitos.

El programa que envía adecuadamente las señales para realizar la decodificación se llama COCI, el que las envía para la tarjeta codificadora se llama SICI. Los diagramas de flujo y los programas estan presentados en forma conjunta con los anteriores.

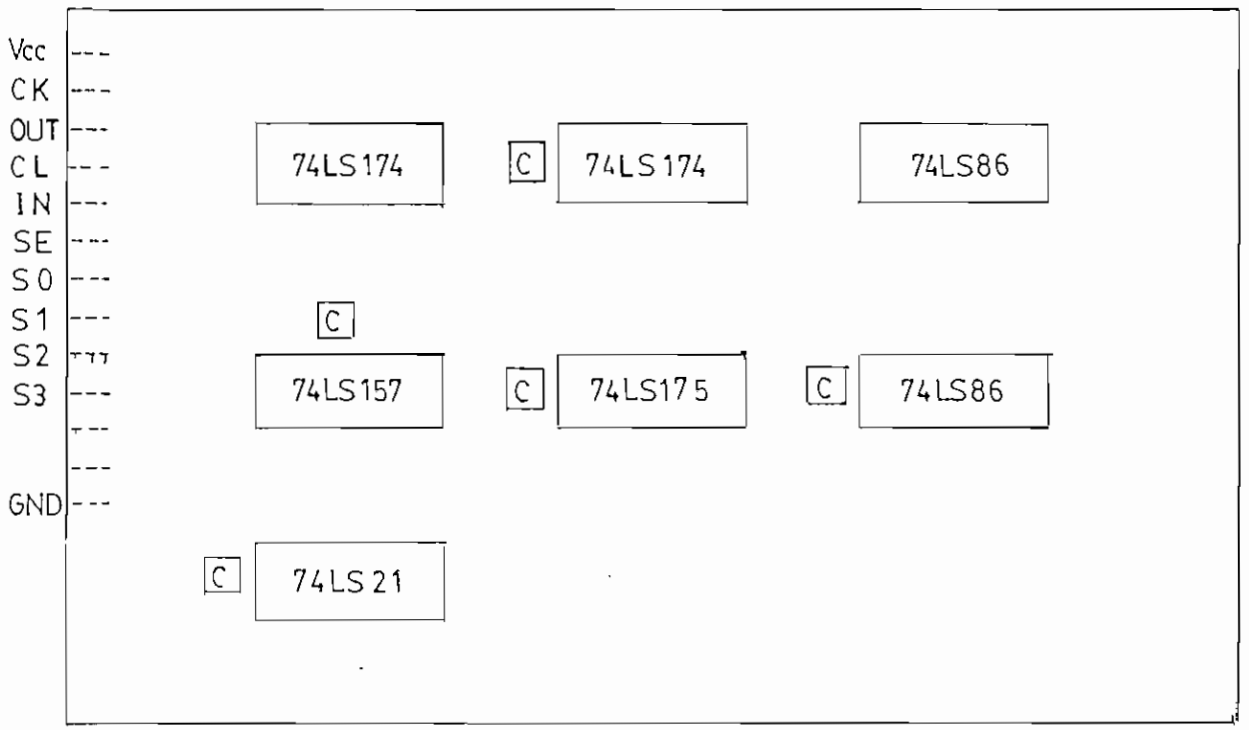


FIGURA 4.11 Ubicación de los elementos

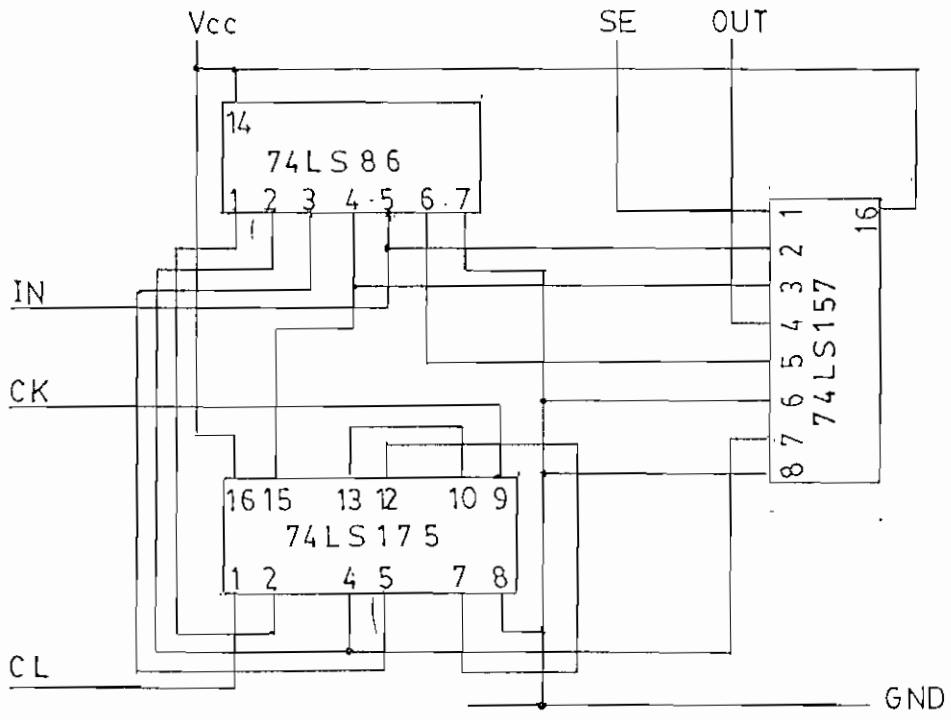


FIGURA 4.8 Diagrama circuital del codificador

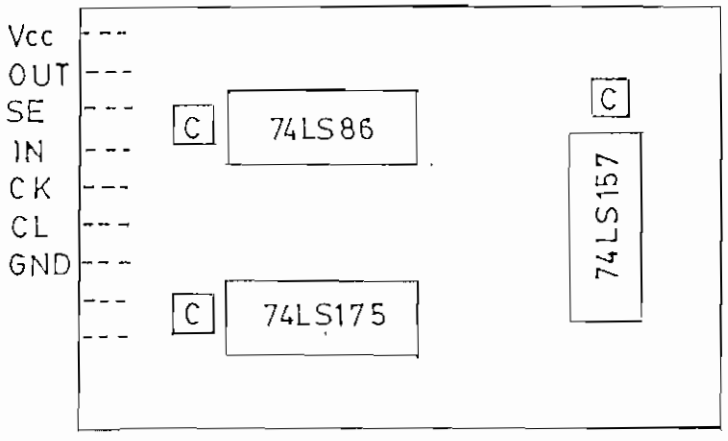


FIGURA 4.9 Ubicación de los elementos

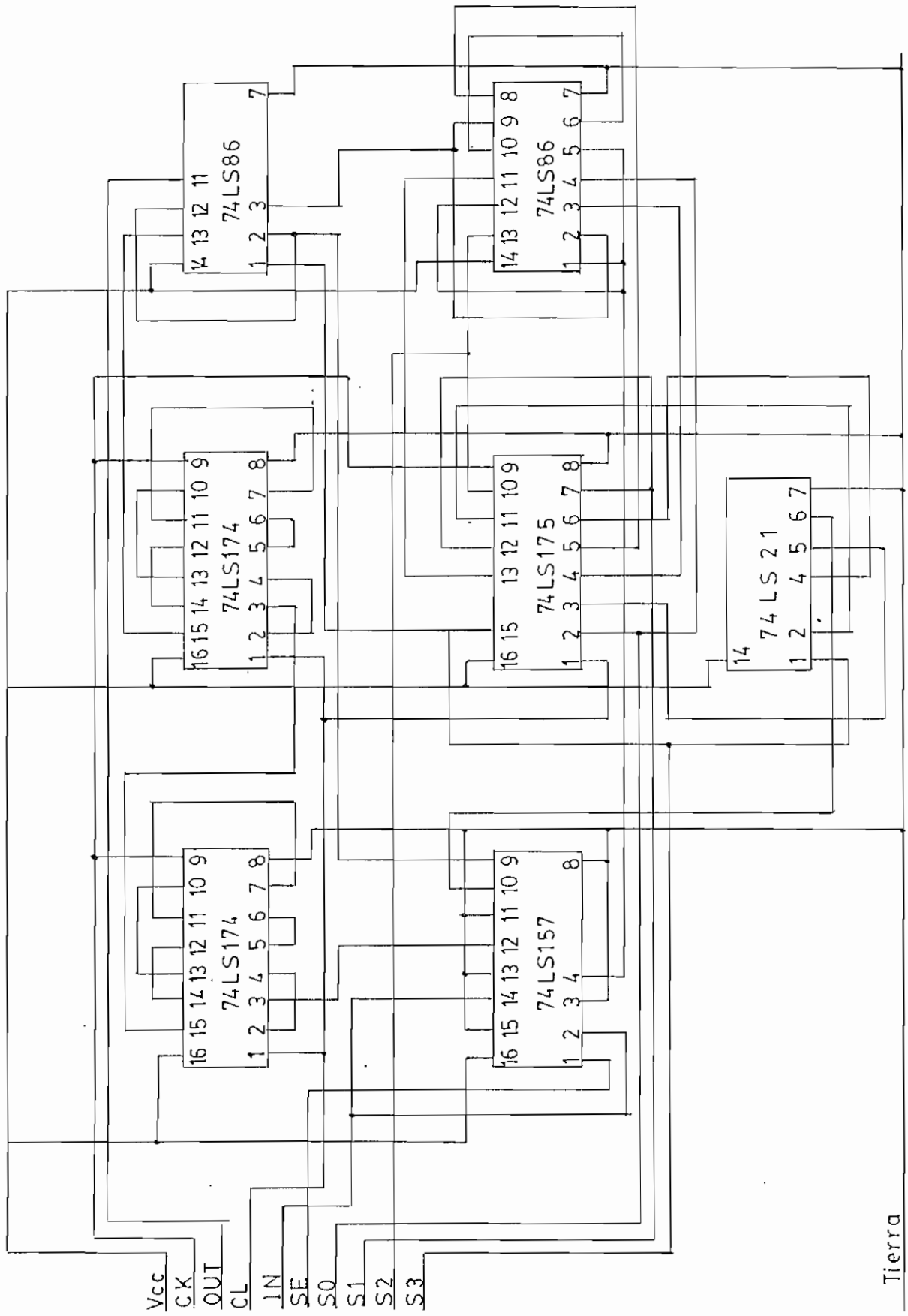
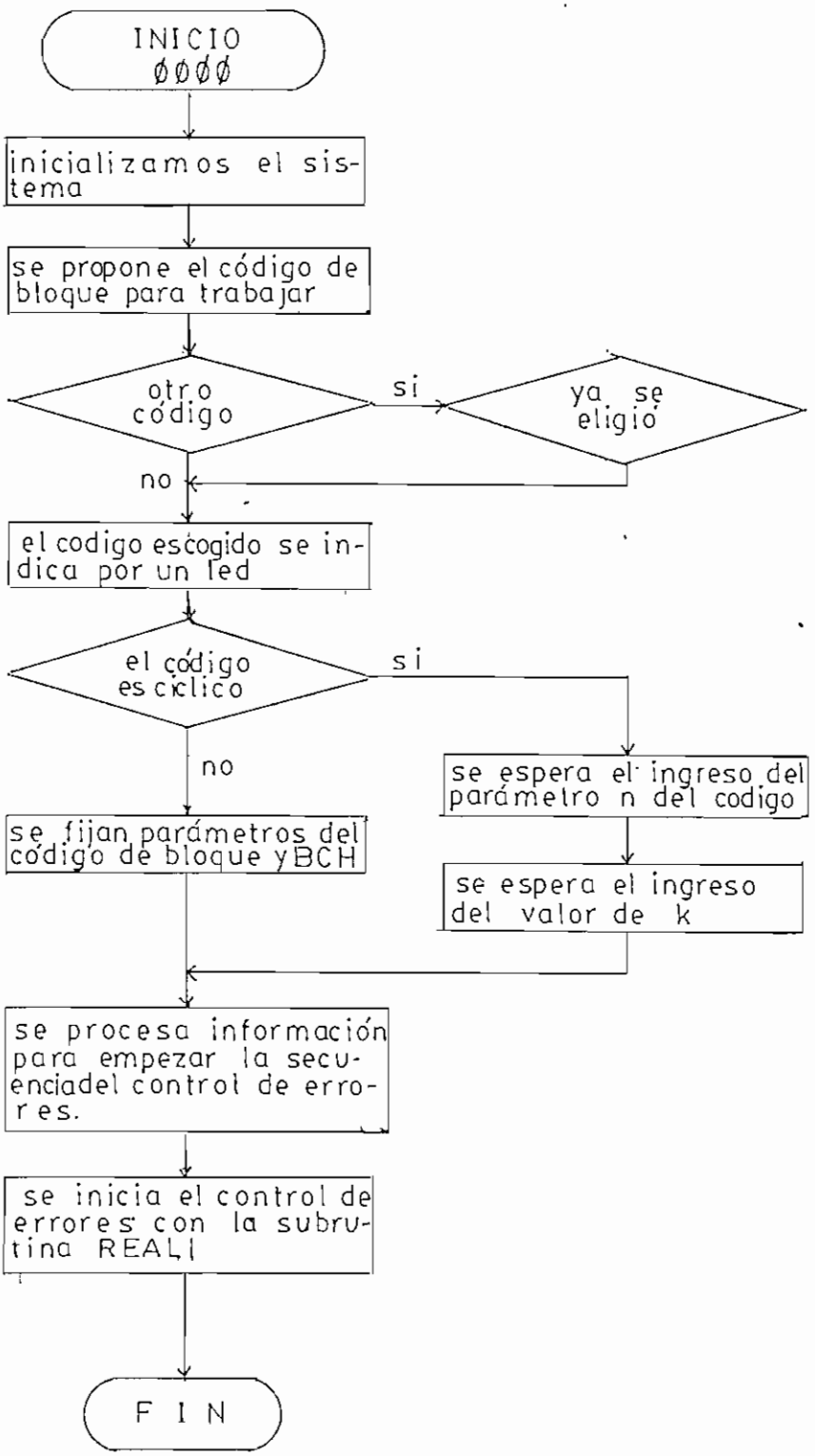
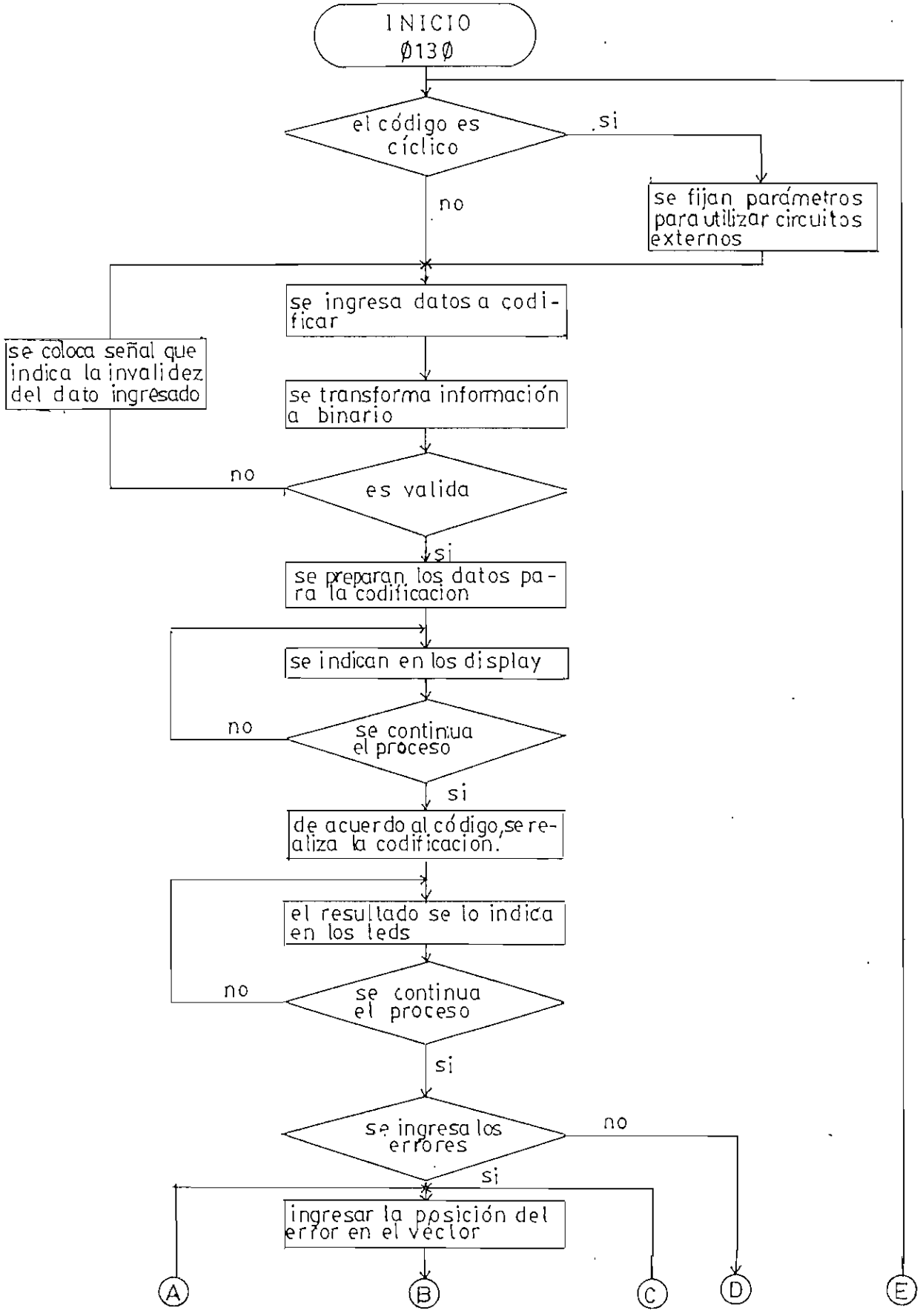


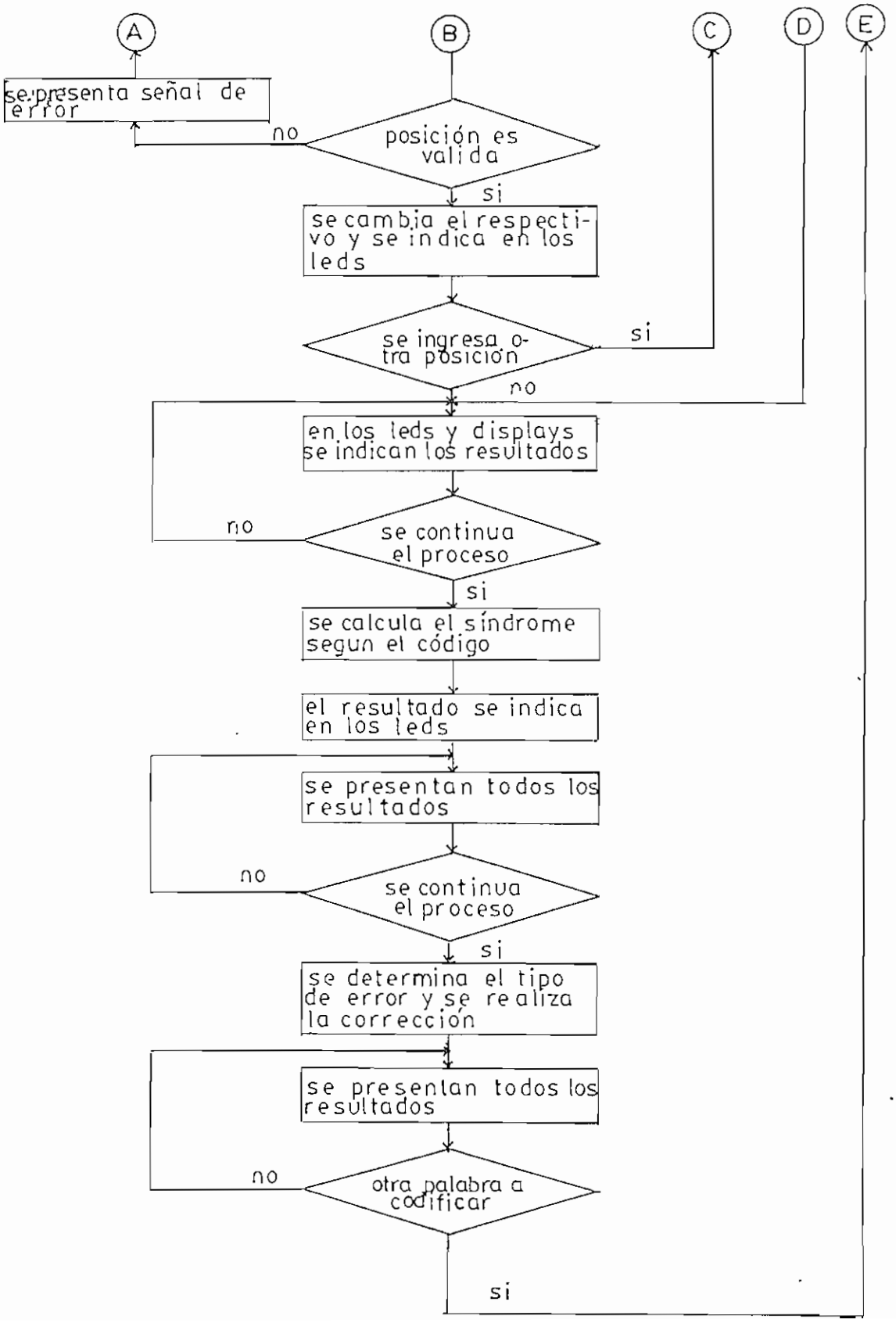
FIGURA 4.10 Diagrama circuital del decodificador.

4.7 Diagramas de flujo

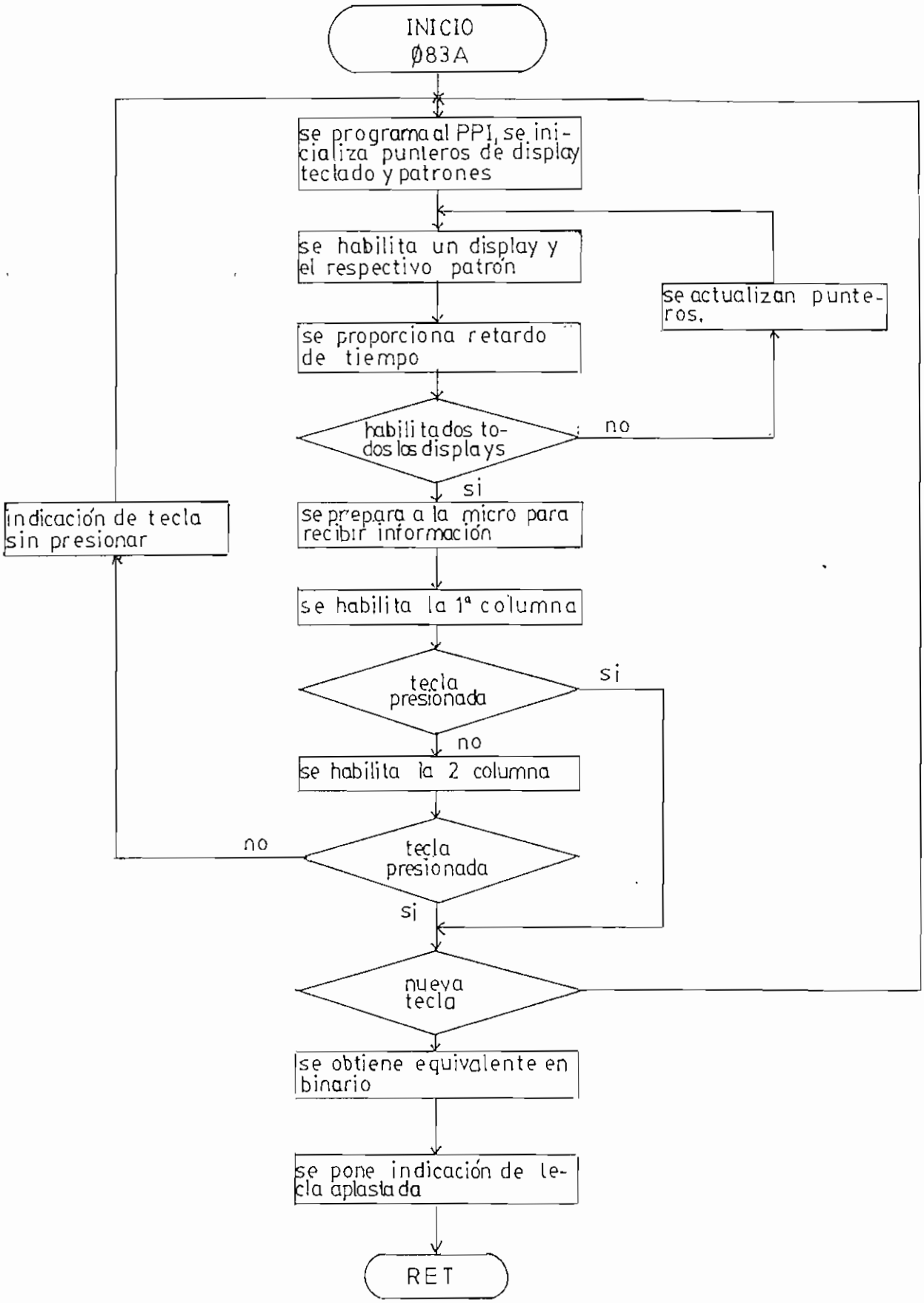


REAL I

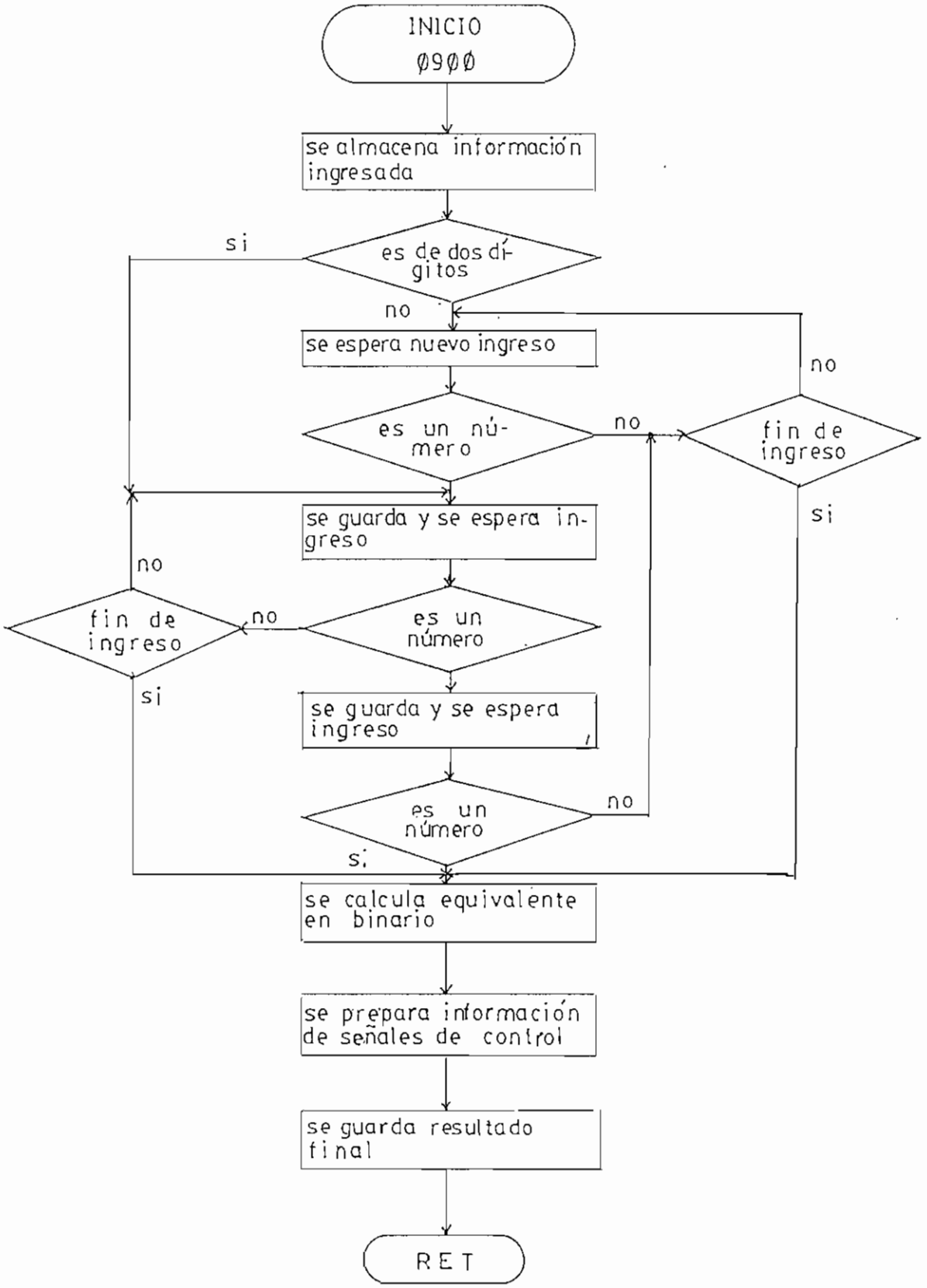

INICIO
Ø13Ø
el código es cíclico
si
no
se fijan parámetros para utilizar circuitos externos
se ingresa datos a codificar
se coloca señal que indica la invalidez del dato ingresado
se transforma información a binario
no
es valida
si
se preparan los datos para la codificacion
se indican en los display
no
se continua el proceso
si
de acuerdo al código, se realiza la codificacion.
el resultado se lo indica en los leds
no
se continua el proceso
si
se ingresa los errores
no
si
ingresar la posición del error en el vector
A
B
C
D
E


A
B
C
D
E
se presenta señal de error
no
posición es valida
si
se cambia el respectivo y se indica en los leds
se ingresa otra posición
si
no
en los leds y displays se indican los resultados
no
se continua el proceso
si
se calcula el síndrome segun el código
el resultado se indica en los leds
se presentan todos los resultados
no
se continua el proceso
si
se determina el tipo de error y se realiza la corrección
se presentan todos los resultados
no
otra palabra a codificar
si

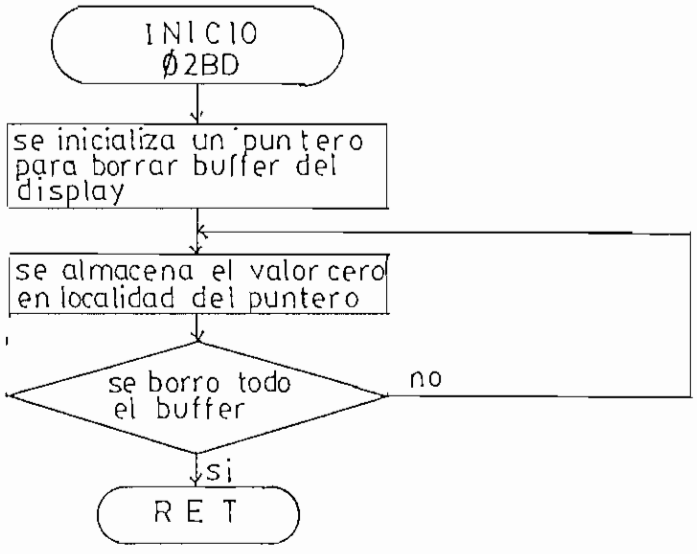
SCAN


INICIO
Ø83A
se programa al PPI, se inicializa punteros de display teclado y patrones
se habilita un display y el respectivo patrón
se actualizan punteros.
se proporciona retardo de tiempo
habilitados todos los displays
no
si
indicación de tecla sin presionar
se prepara a la micro para recibir información
se habilita la 1ª columna
tecla presionada
si
no
se habilita la 2 columna
no
tecla presionada
si
nueva tecla
se obtiene equivalente en binario
se pone indicación de tecla aplastada
RET

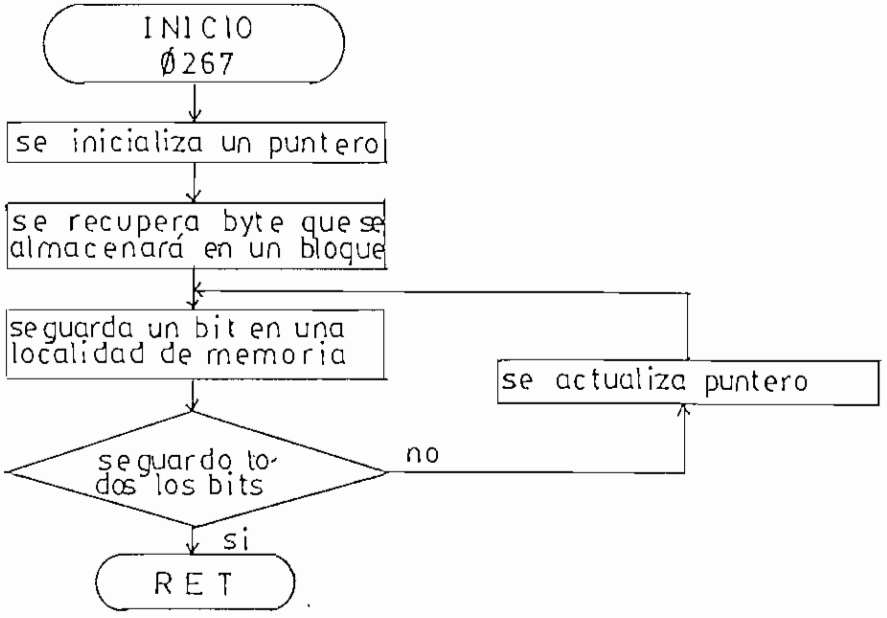
TREN


INICIO
Ø9ØØ
se almacena información ingresada
es de dos dígitos
si
no
se espera nuevo ingreso
es un número
no
fin de ingreso
no
si
se guarda y se espera ingreso
fin de ingreso
no
es un número
no
si
se guarda y se espera ingreso
es un número
no
si
se calcula equivalente en binario
se prepara información de señales de control
se guarda resultado final
RET

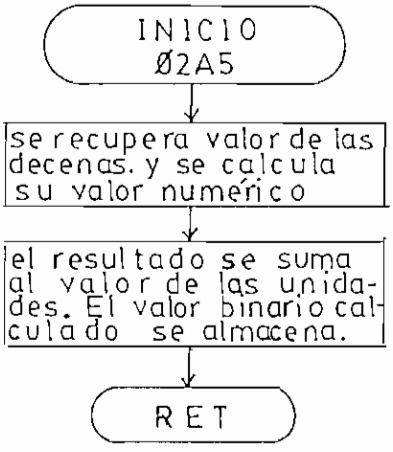
BORA



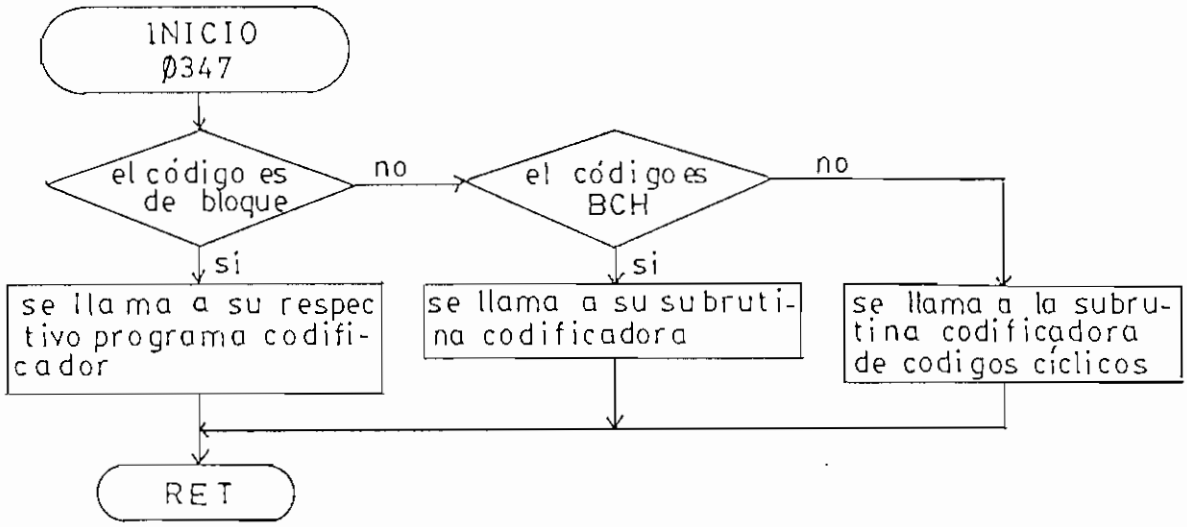
GBLO



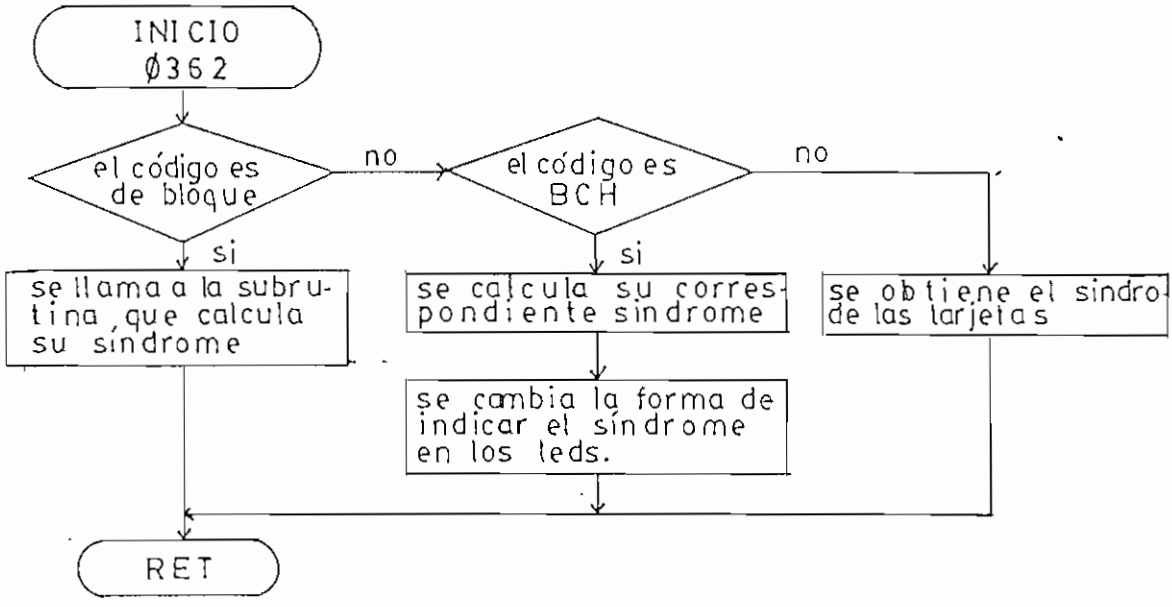
BIDI



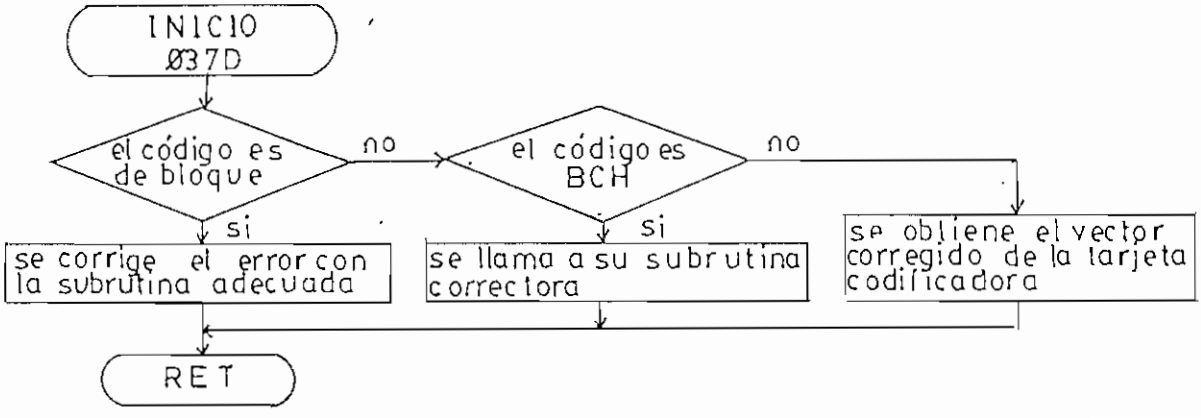
CODI



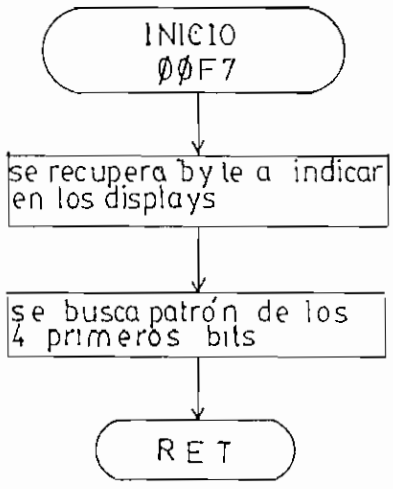
SINDRO



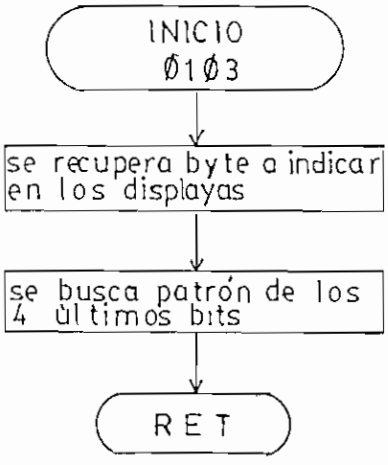
CORR



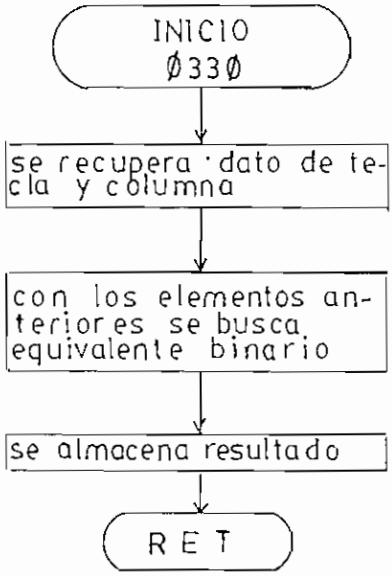
BUFL



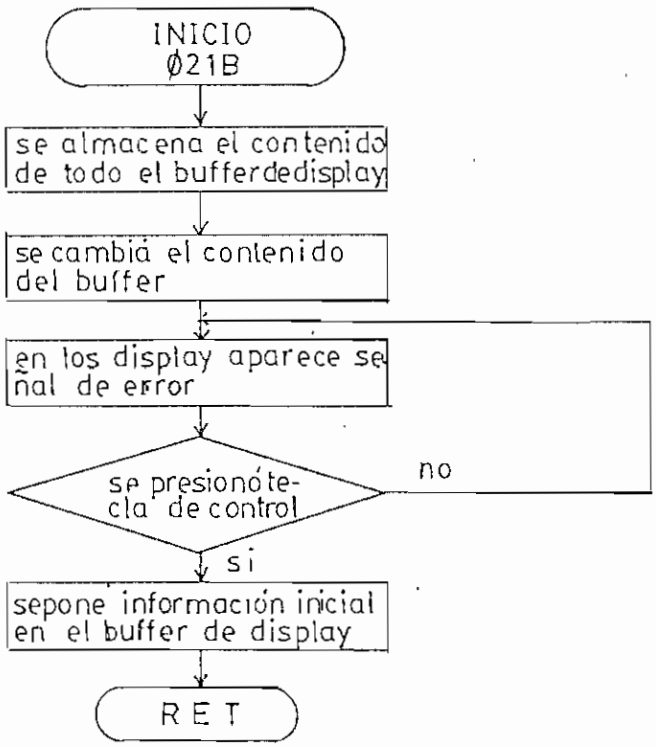
BUFH



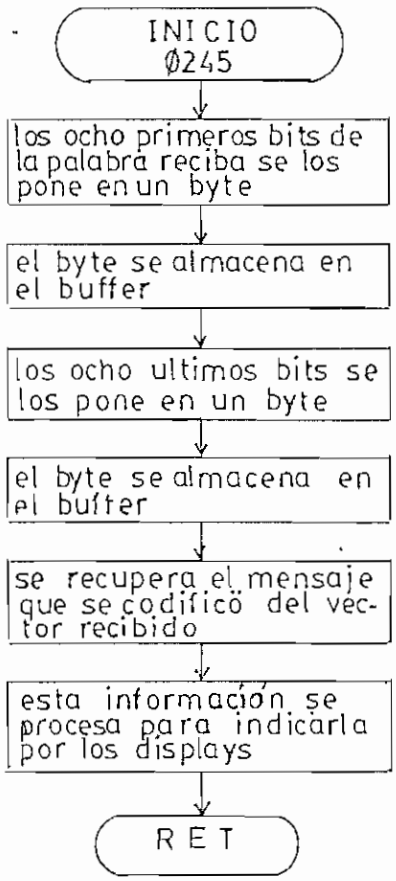
PROCE



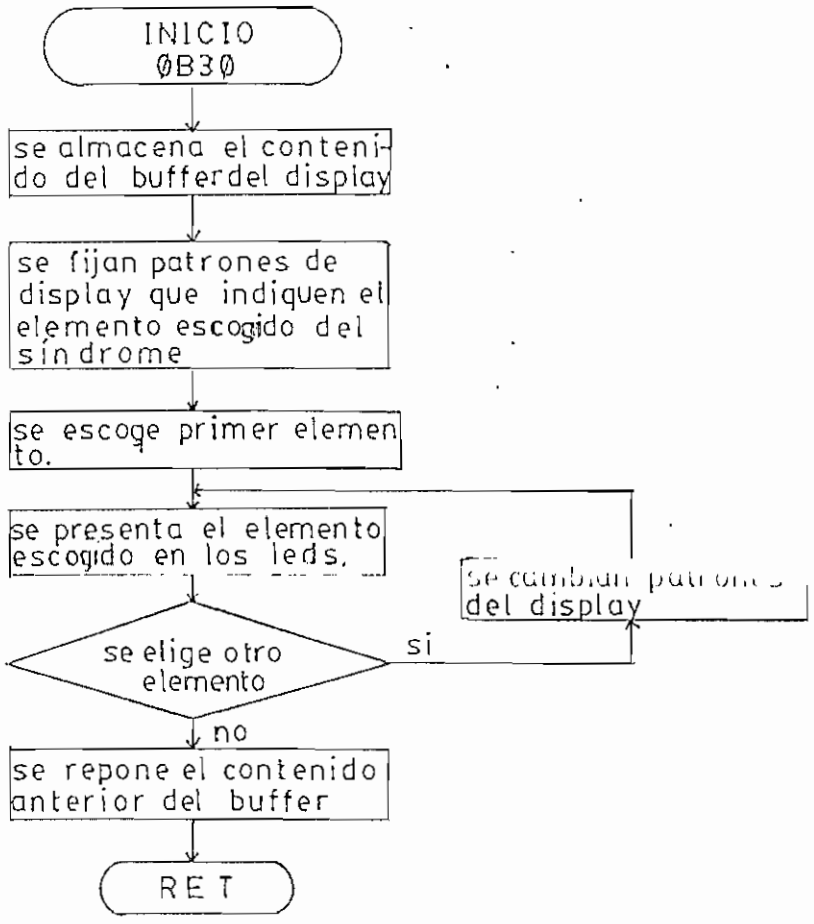
ERR7


INICIO
Ø21B
se almacena el contenido de todo el bufferdedisplay
se cambia el contenido del buffer
en los display aparece señal de error
se presionóte-cla de control
no
si
sepone información inicial en el buffer de display
RET

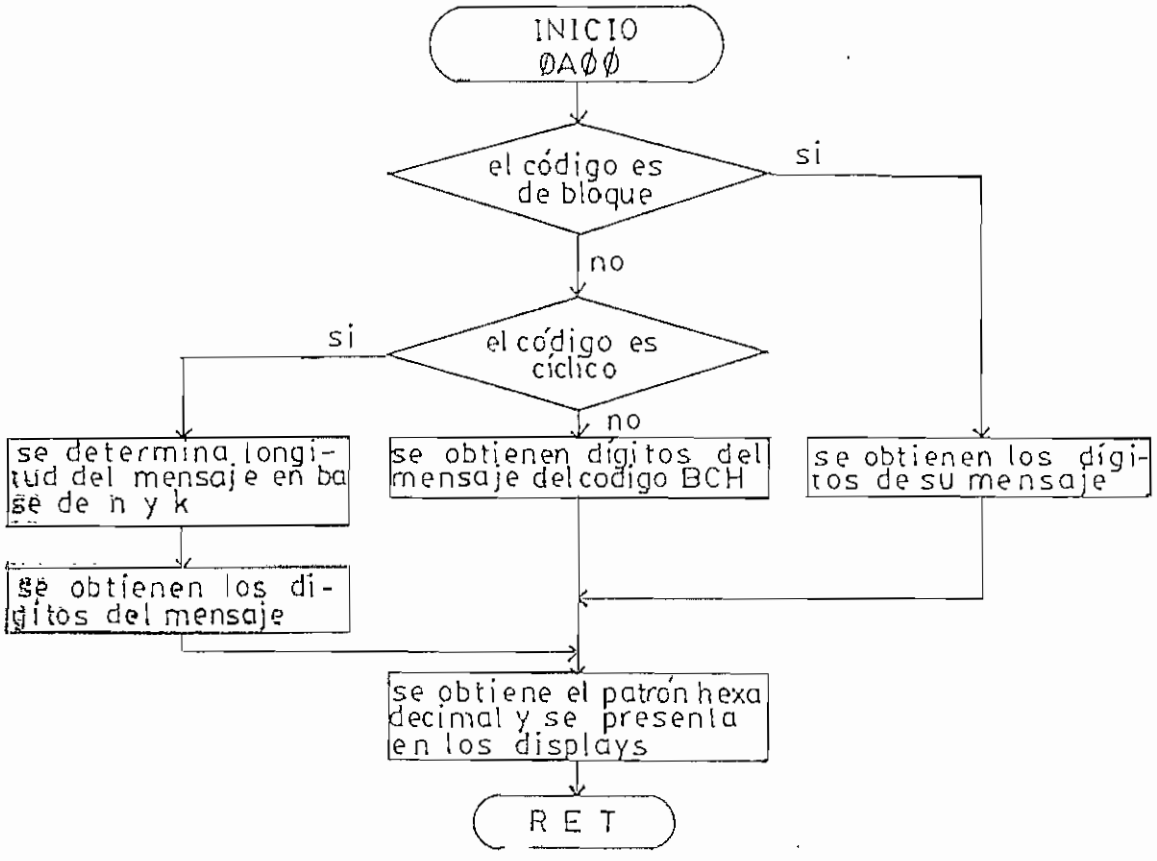
PRESS


INICIO
Ø245
los ocho primeros bits de la palabra reciba se los pone en un byte
el byte se almacena en el buffer
los ocho ultimos bits se los pone en un byte
el byte se almacena en el buffer
se recupera el mensaje que se codificö del vector recibido
esta información se procesa para indicarla por los displays
RET

MUES

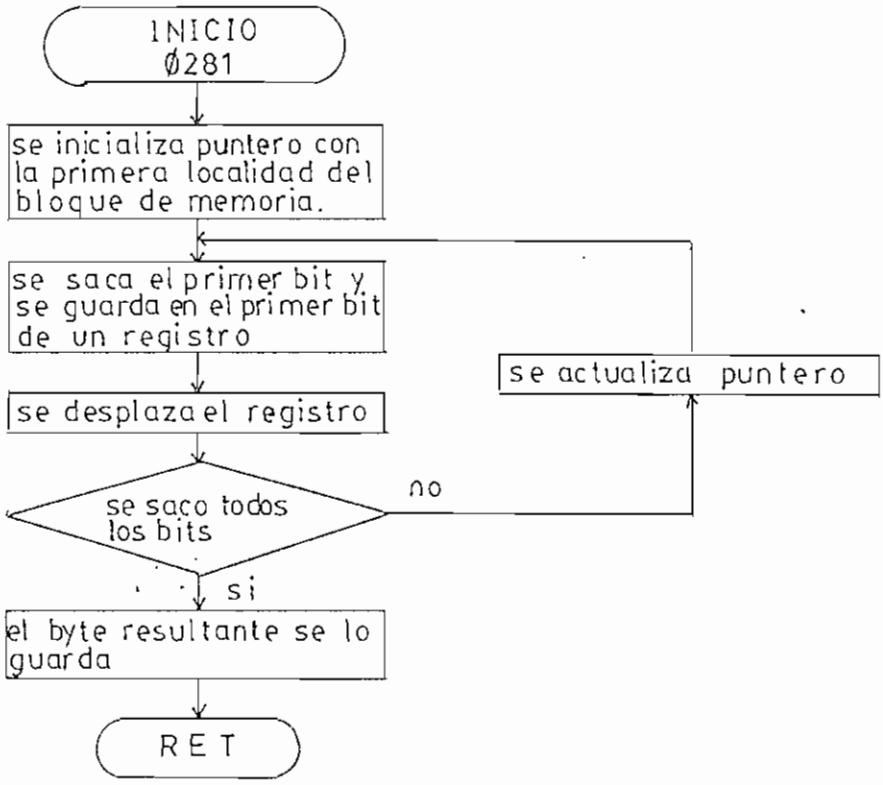

INICIO
ØB3Ø
se almacena el contenido del bufferdel display
se fijan patrones de display que indiquen el elemento escogido del sindrome
se escoge primer elemento.
se presenta el elemento escogido en los leds.
se elige otro elemento
si
no
se repone el contenido anterior del buffer
RET

VAR I

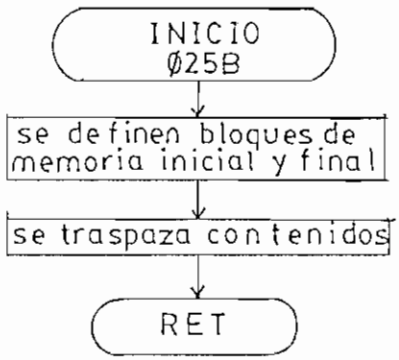

INICIO
ØAØØ
el código es de bloque
si
no
el código es ciclico
si
no
se determina longitud del mensaje en base de n y k
se obtienen los dígitos del mensaje
se obtienen dígitos del mensaje delcodigo BCH
se obtienen los dígitos de su mensaje
se obtiene el patrón hexadecimal y se presenta en los displays
RET

MODI

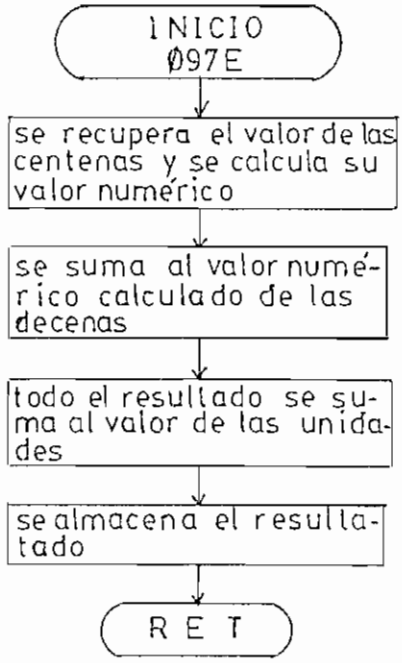
ARRE



TRAN

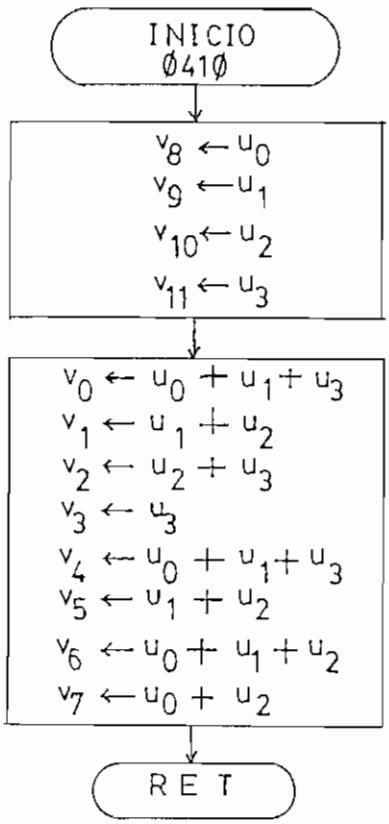


BIDE

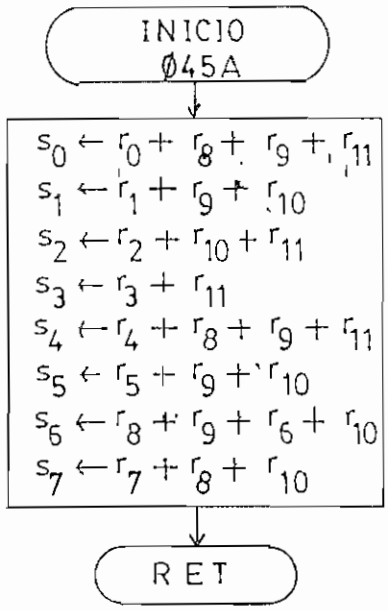


4.7.1 Diagramas del código de bloque

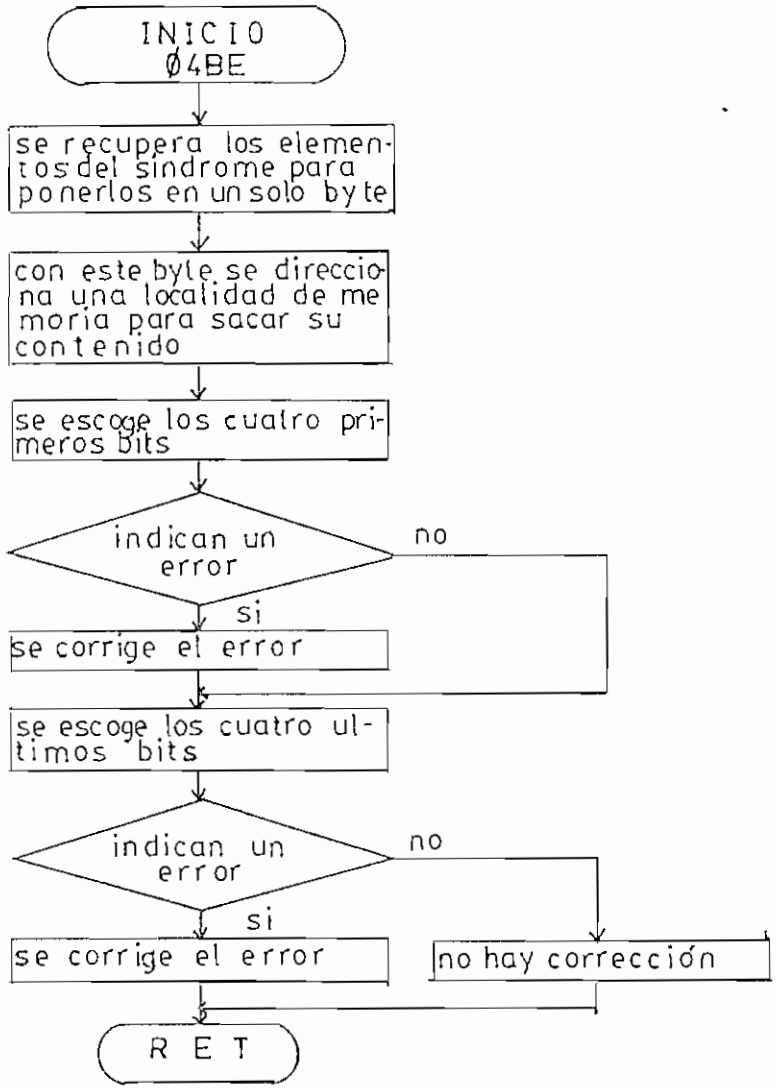
CODB



SIBL

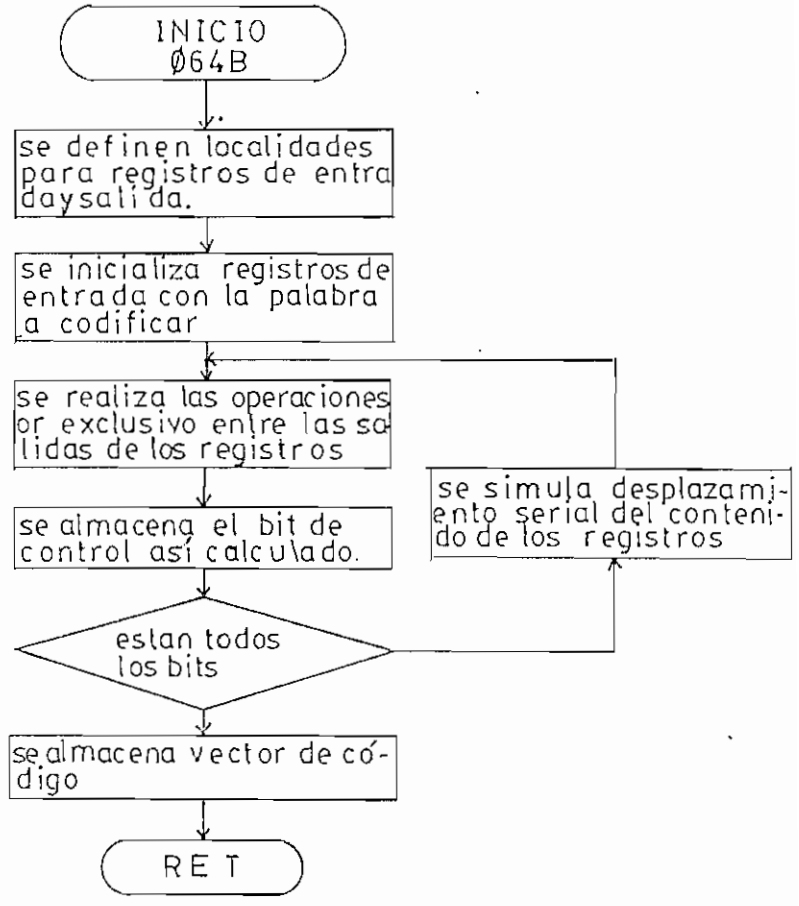

INICIO
Ø45A
$s_0 \leftarrow r_0 + r_8 + r_9 + r_{11}$
$s_1 \leftarrow r_1 + r_9 + r_{10}$
$s_2 \leftarrow r_2 + r_{10} + r_{11}$
$s_3 \leftarrow r_3 + r_{11}$
$s_4 \leftarrow r_4 + r_8 + r_9 + r_{11}$
$s_5 \leftarrow r_5 + r_9 + r_{10}$
$s_6 \leftarrow r_8 + r_9 + r_6 + r_{10}$
$s_7 \leftarrow r_7 + r_8 + r_{10}$
RET

CRRB

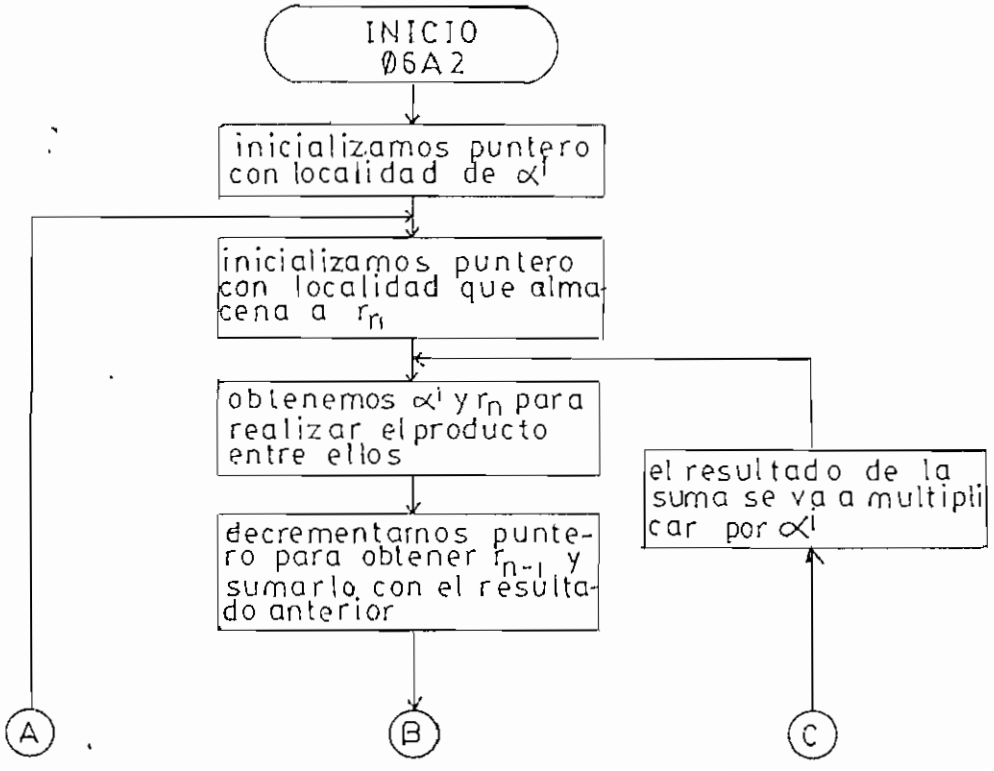

INICIO
Ø4BE
se recupera los elementos del sindrome para ponerlos en un solo byte
con este byte se direcciona una localidad de memoria para sacar su contenido
se escoge los cuatro primeros bits
indican un error
no
si
se corrige el error
se escoge los cuatro ultimos bits
indican un error
no
si
se corrige el error
no hay corrección
RET

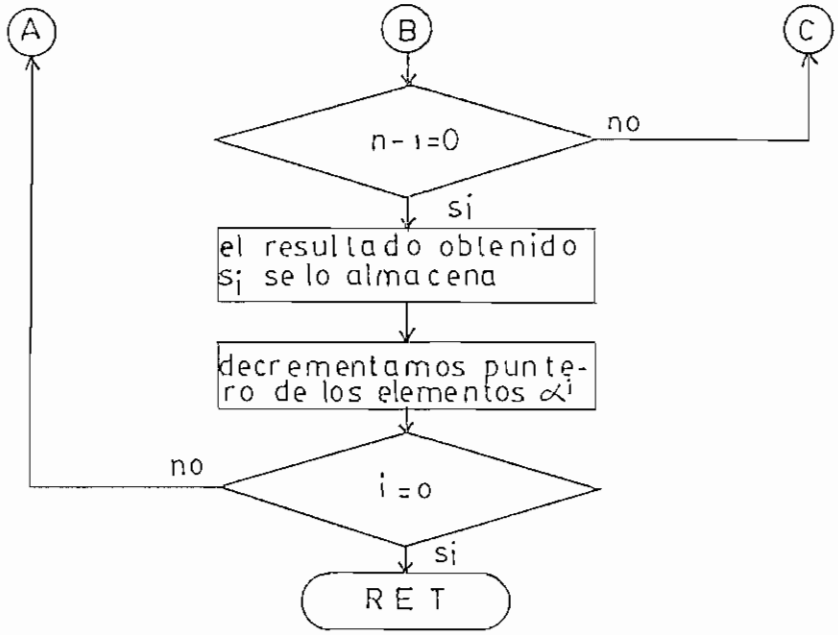
4.7.2 Diagramas del código BCH

COBC

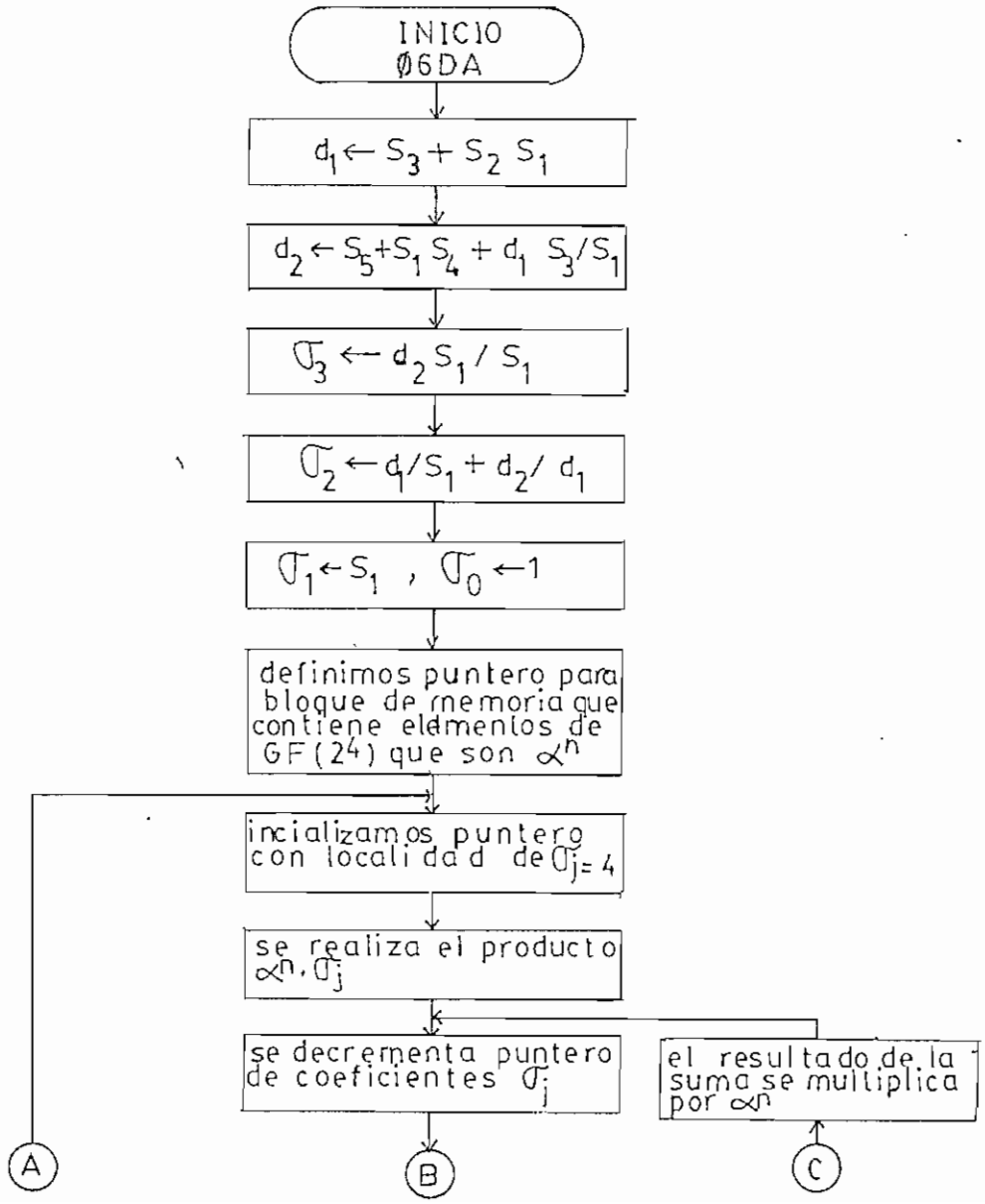


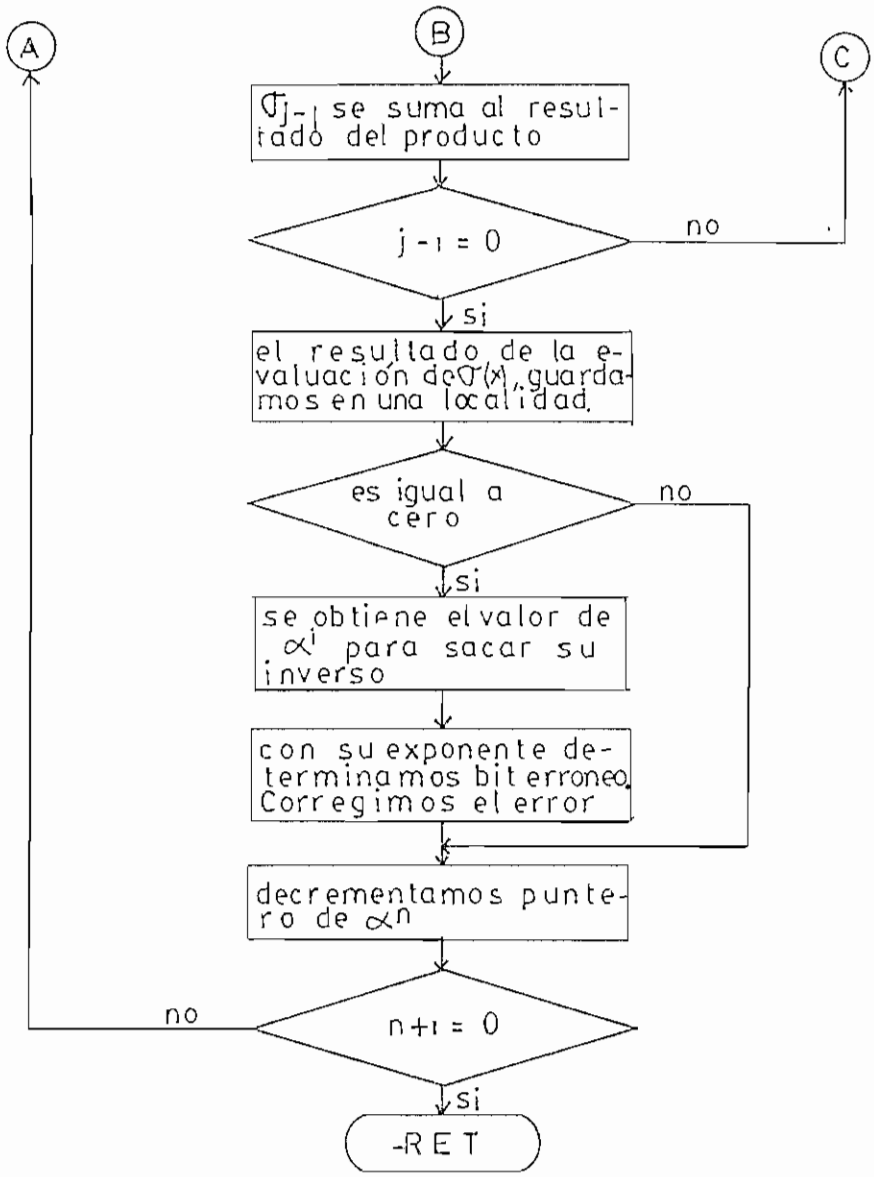
SIBC



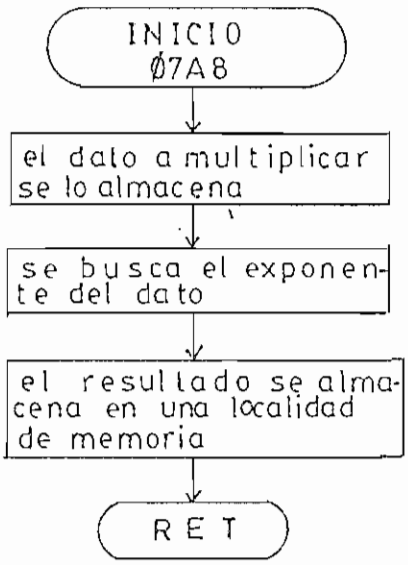

A
B
C
n-1=0
no
si
el resultado obtenido s_i se lo almacena
decrementamos puntero de los elementos α^i
i=0
no
si
RET

CRBC

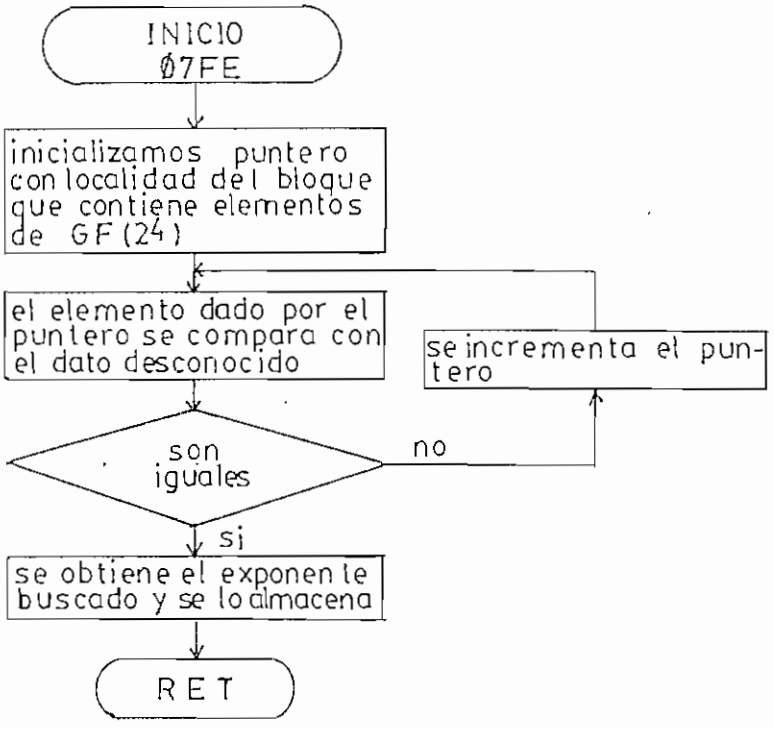

INICIO
Ø6DA
$d_1 \leftarrow S_3 + S_2 S_1$
$d_2 \leftarrow S_5 + S_1 S_4 + d_1 S_3/S_1$
$\sigma_3 \leftarrow d_2 S_1 / S_1$
$\sigma_2 \leftarrow d_1/S_1 + d_2/d_1$
$\sigma_1 \leftarrow S_1$, $\sigma_0 \leftarrow 1$
definimos puntero para bloque de memoria que contiene elementos de $GF(2^4)$ que son α^n
incializamos puntero con localidad de $\sigma_{j=4}$
se realiza el producto $\alpha^n . \sigma_j$
se decrementa puntero de coeficientes σ_j
el resultado de la suma se multiplica por α^n
A
B
C


A
B
C
σ_{j-1} se suma al resultado del producto
j-1 = 0
no
si
el resultado de la evaluación de $\sigma(x)$, guardamos en una localidad.
es igual a cero
no
si
se obtiene el valor de α^i para sacar su inverso
con su exponente determinamos bit erroneo. Corregimos el error
decrementamos puntero de α^n
n+1 = 0
no
si
-RET

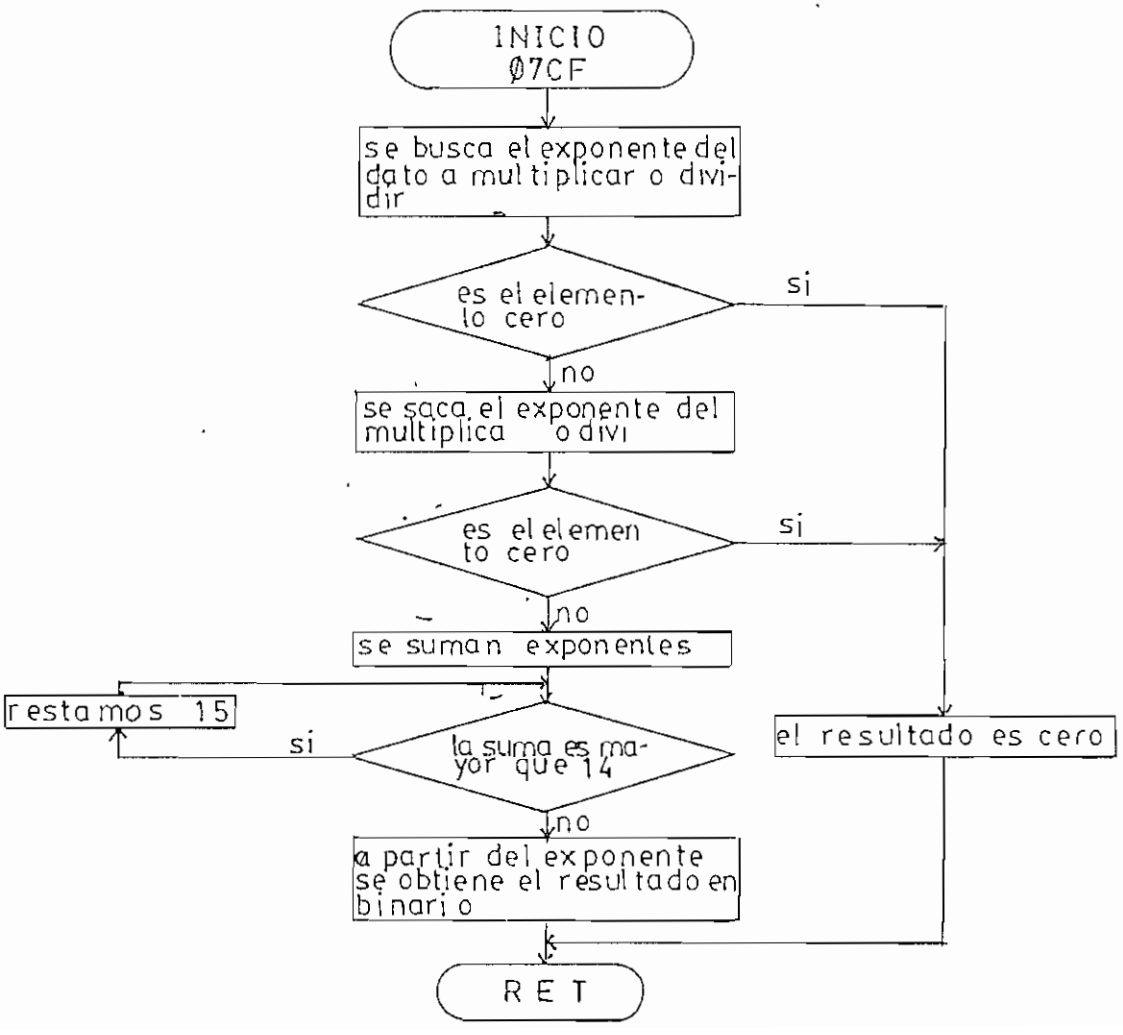
MUL


INICIO
Ø7A8
el dato a multiplicar se lo almacena
se busca el exponente del dato
el resultado se almacena en una localidad de memoria
RET

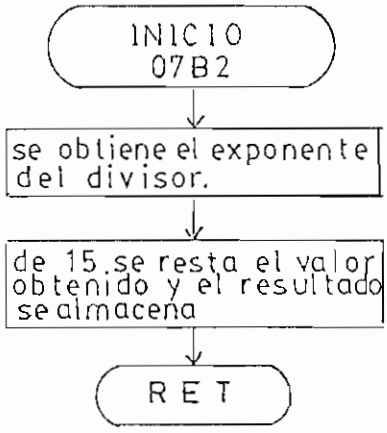
XPON


INICIO
Ø7FE
inicializamos puntero con localidad del bloque que contiene elementos de GF(2⁴)
el elemento dado por el puntero se compara con el dato desconocido
se incrementa el puntero
son iguales
no
si
se obtiene el exponente buscado y se lo almacena
RET

OPE

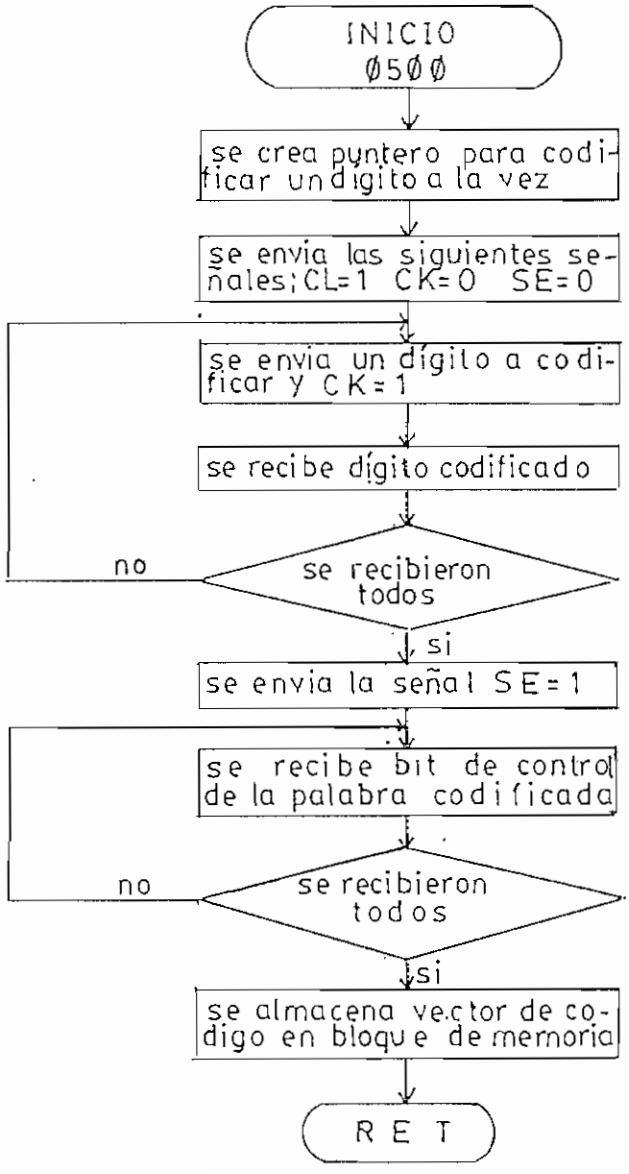

INICIO
Ø7CF
se busca el exponente del dato a multiplicar o dividir
es el elemento cero
si
no
se saca el exponente del multiplicando o divi
es el elemento cero
si
no
se suman exponentes
restamos 15
si
la suma es mayor que 14
no
el resultado es cero
a partir del exponente se obtiene el resultado en binario
RET

DIVI

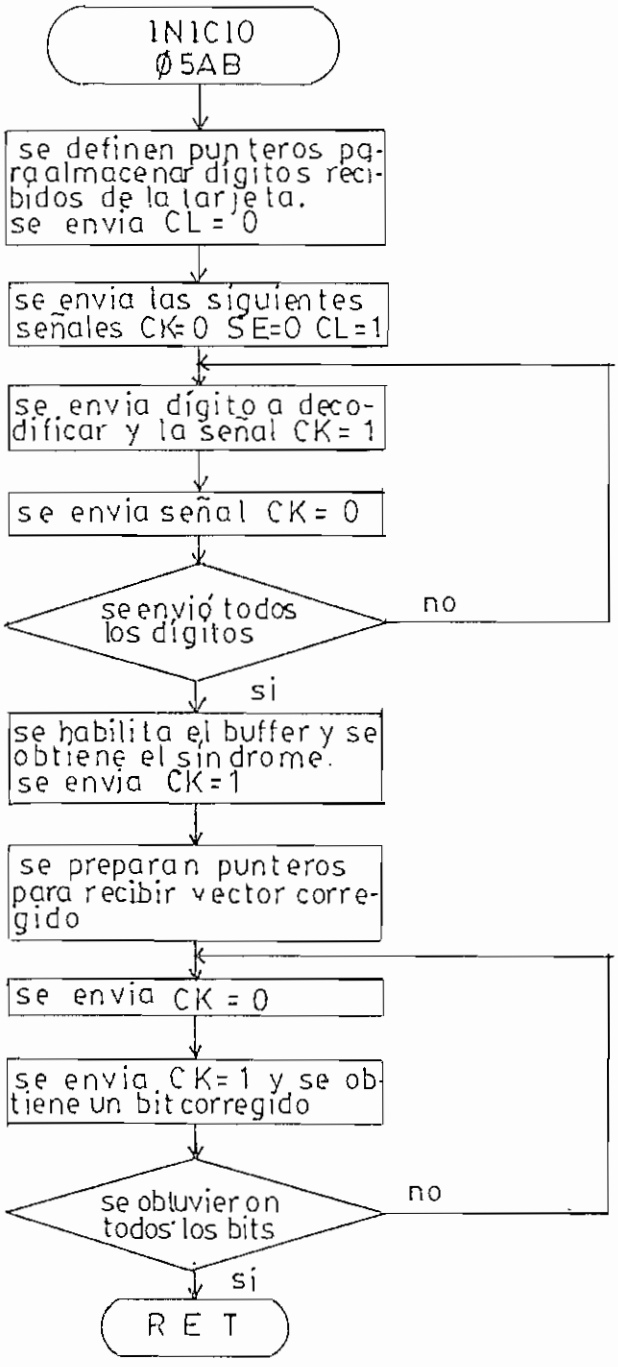


4.7.3 Diagramas del código cíclico

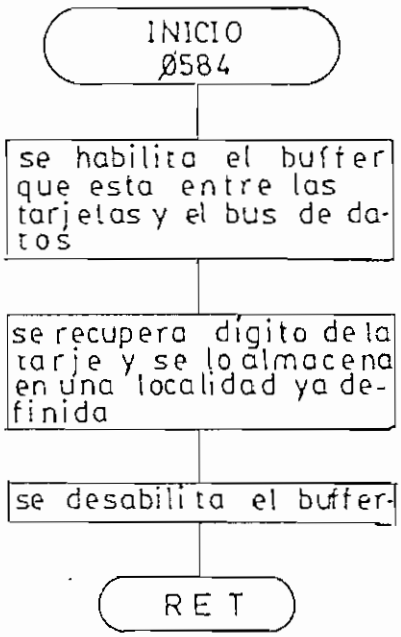
COCI



SIC1


INICIO
Ø5AB
se definen punteros para almacenar dígitos recibidos de la tarjeta.
se envia CL = 0
se envia las siguientes señales CK=0 SE=0 CL=1
se envia dígito a decodificar y la señal CK = 1
se envia señal CK = 0
se envió todos los dígitos
no
si
se habilita el buffer y se obtiene el síndrome.
se envia CK=1
se preparan punteros para recibir vector corregido
se envia CK = 0
se envia CK=1 y se obtiene un bit corregido
se obtuvieron todos los bits
no
si
R E T

SACA


INICIO
Ø584
se habilita el buffer que esta entre las tarjetas y el bus de datos
se recupera dígito de la tarje y se lo almacena en una localidad ya definida
se desabilita el buffer
RET

4.8. Operación del Aparato.-

Por medio del teclado podemos introducir la información necesaria para probar tanto los codificadores como los decodificadores. Las teclas utilizadas para introducir información a codificar y la posición donde queremos introducir un error, son las teclas numéricas.

La información decimal a codificar, está limitada por la longitud del mensaje dentro del vector de código. Para el código de bloque podemos introducir del 0 al 15, para el código BCH hasta el 32, y para el - cíclico dependerá del código que se implemente, en nuestro caso el código implementado acepta información hasta el número 255.

Existen cinco teclas de control cuya finalidad describimos a continuación.

Tecla C - Permite continuar con el proceso de control de errores.
- Cuando en los display aparece la señal de Error, este tecla nos permite regresar al estado anterior.

Tecla S - Selecciona el tipo de código con el cuál queremos trabajar.
- Seleccionamos un componente del síndrome cuyos elementos - queremos ver en los leds. Esto es sólo para los códigos BCH.

Tecla I - Inicializa el proceso de control de errores, es decir, podemos ingresar un nuevo dato a codificar con el código ya seleccionado.

- Una vez que se ha seleccionado el código con el cual se va a trabajar, después de presionar esta tecla, podemos ingresar el dato a codificar.

Tecla F - Al presionar este tecla estamos en capacidad de modificar nuevamente el vector de código, indicando la posición en la cual introduciremos un error.

Tecla R - Al presionar esta tecla estamos en capacidad de seleccionar otro código.

Por medio de los conectores se manda información a las tarjetas y se recibe información de ellas, por lo que, para su correcto funcionamiento es necesario que se realice adecuadamente la conexión de las tarjetas y los conectores.

Las correspondientes señales de los pines del conector son:

CONECTOR CODIFICADOR		CONECTOR DECODIFICADOR	
1	Vcc	1	Vcc
2	Salida	2	CK
3	SE	3	Salida
4	Entrada	4	CL
5	CK	5	Entrada
6	CL	6	SE
7	Tierra	7	SO

8	NC	8	S1
9	NC	9	S2
10	NC	10	S3
		11	S4
		12	S5
		13	Tierra
		14	NC

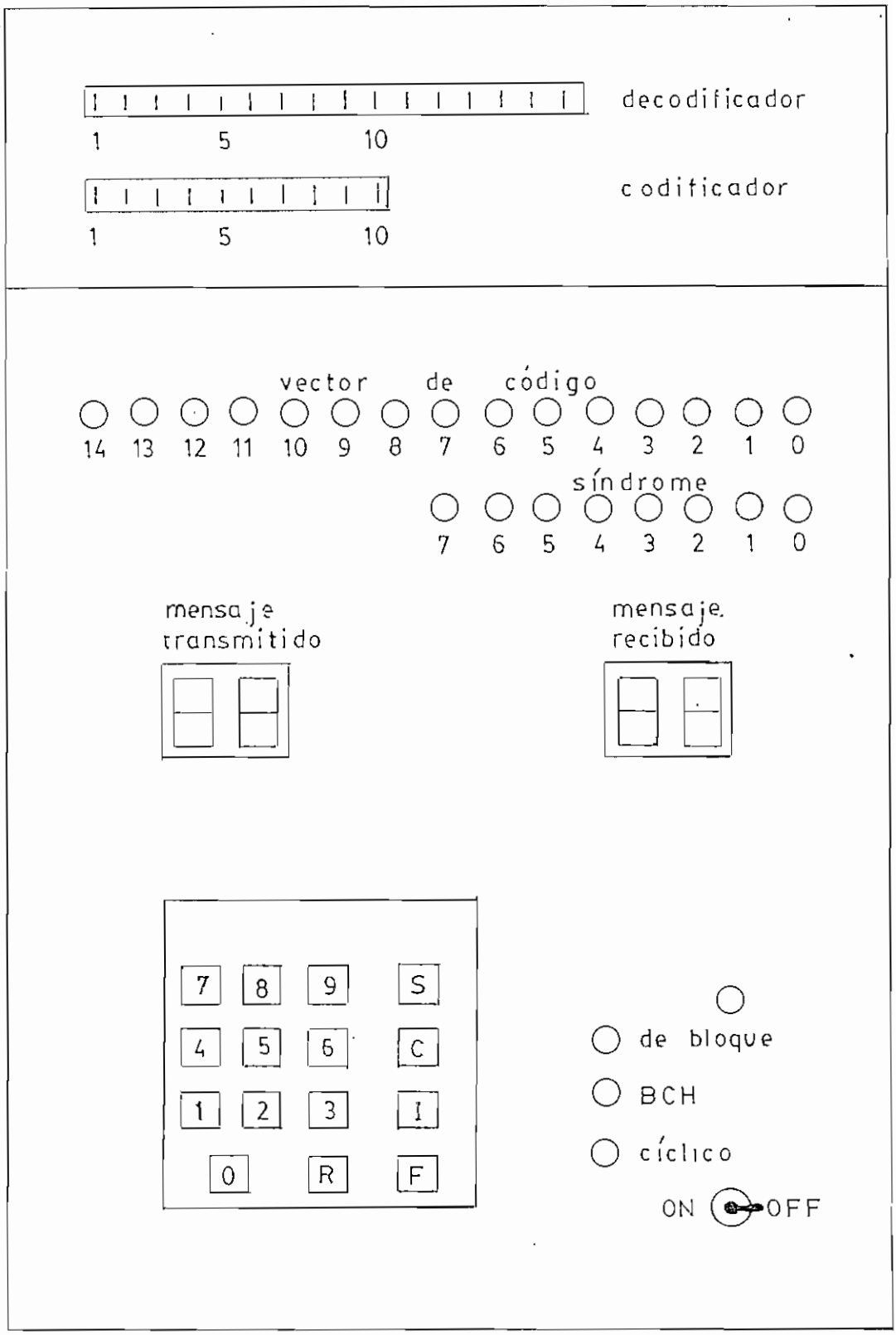
Cuando el código con el cual se va a trabajar es el código cíclico, es necesario primero introducir primero la longitud del código vector (n) y luego la longitud del mensaje a codificar (l) estos parámetros darán la información necesaria para que el microprocesador envíe las señales adecuadas a los circuitos codificadores y decodificadores exteriores al sistema.

Se debe recordar que los integrados que se utilicen en la implementación de los codificadores y decodificadores de los códigos cíclicos deben ser LS.

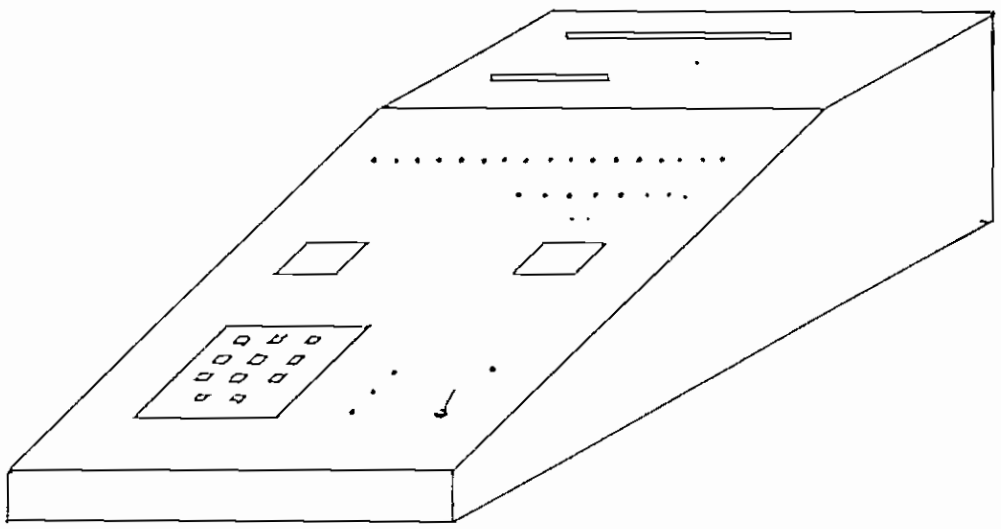
Cuando se quiera introducir un error en una posición que sea más grande que la longitud del código, el aparato proporcionará por medio de los displays una señal de error.

Lo mismo ocurrirá cuando se quiera codificar un dato cuyo número de dígitos para ser representado en forma digital, sea mayor que el parámetro k del código.

4.8.1 Configuración externa



Distribución física de los display, leds, teclado y conectores.



Vista exterior

CAPITULO V

5.1. Resultados Experimentales.-

La implementación de los codificadores y decodificadores permitió comprobar todas las propiedades teóricas que se plantearon respecto a los códigos de bloque.

Los resultados obtenidos corresponden a los siguientes parámetros.

(a) Información decimal a codificar introducida.

(b) Información hexadecimal a codificar.

(c) Vector de código. $\bar{v} = (v_0 , v_1 , v_2 \ldots\ldots v_n)$

(d) Posiciones erróneas introducidas.

(e) Vector de código erróneo. $\bar{r} = (r_0 , r_1 , r_2 , \ldots\ldots r_n)$

(f) Información que se halla en la parte del mensaje, del vector, en forma hexadecimal con o sin error.

(g) Síndrome. $\bar{s} = (s_1 , s_2 , s_3 , \ldots\ldots s_{n-k})$

A continuación se presentan los resultados que se obtienen como resultado del control de errores tal como nos presenta el aparato construído.

Para el código de bloque se tiene lo siguiente:

CONCLUSIONES

El objetivo del diseño en el control de errores, es el de elegir una mínima cantidad de bits de control para corregir y detectar el máximo de errores, de tal manera de mantener la eficiencia de transmisión - lo más alta posible.

La elección del tipo de código y de sus parámetros n y k, dependerá del sistema de comunicación y de las aplicaciones que se le den a ese sistema específico.

Las características de un buen código vienen dadas por las propiedades que posea su matriz generadora, tales como linealidad y estructura sistemática. Además, los parámetros importantes en la construcción - de códigos óptimos son la probabilidad de decodificación errónea, la- longitud del código y la eficiencia de transmisión.

Estas propiedades se traducen en una fácil implementación y habilidad para corregir errores aleatorios y tipo ráfaga.

En el proceso de decodificación, el punto crítico, es el momento en que se asocia al síndrome un tipo de error específico, ya que en primer lugar la complejidad de los circuitos decodificadores dependerán - en su totalidad del tipo de código y técnica que se utilice para lograr este propósito. En segundo lugar el momento de asociar el síndrome a un

tipo de error existe una probabilidad de que se dé una decodificación - errónea.

La base de la decodificación es que a un tipo de error le corresponda un síndrome específico.

Determinar la probabilidad de una decodificación errónea resulta en muchos casos imposible de saber, debido a los altos valores n y k que se utilizan en la práctica. Sin embargo, si se puede determinar de una- manera rápida su límite máximo que depende de los parámetros del código.

Los códigos cíclicos se caracterizan porque tanto el codificador como el decodificador, con respecto a otros códigos, resultan fácilmente implementados debido a la propiedad cíclica que poseen.

El uso adecuado de las propiedades geométricas y algábricas que - rigen a los códigos cíclicos, reducen la complejidad del circuito que realiza la decodificación.

Todo tipo de código es susceptible de ser acoratado cuando el número de bits que van a ser codificados es menor que el número de bits que puede aceptar el código para realizar la codificación. Este número de bits está determinado por el parámetro k.

El acortamiento no cambia la capacidad que tiene el código para- la detección y corrección de errores, ya que esta capacidad esta determinada por la distancia mínima que no varía cuando se acorta el código.

Los códigos BCH son códigos cíclicos que se analizan con ayuda de otros conceptos matemáticos como son los campos finitos. Este análisis se lo hace con el objeto de que la complejidad de su decodificación disminuya.

La estructura cíclica de los códigos de bloque permite disminuir considerablemente la complejidad de los circuitos codificadores y decodificadores, la cual sólo dependerá de la longitud de los códigos a implementar y de la capacidad de corrección y detección de éstos.

La complejidad de los circuitos cuando la capacidad de corrección de errores de los códigos aumenta se hace más notoria en la decodificación.

Matemáticamente, un código (n,k) es un espacio vectorial de dimensión k, en donde las palabras de código son los elementos que conforman este espacio vectorial. Las palabras de código que conforman la matriz generadora son vectores linealmente independientes que generan dicho - espacio vectorial. El código cumple con todas las propiedades de un espacio vectorial.

La utilización de programas en lenguaje de máquina para realizar la codificación y decodificación de estos códigos se traduce en un retrazo de tiempo con relación a los circuitos que se pueden implementar con elementos discretos.

Por último, utilización de los códigos correctores y detectores

de errores es una opción más que se puede utilizar para mejorar la confiabilidad de un sistema de transmisión y alnacenamiento digital, sacrificando la eficiencia de transmisión.

*** & ***

PROGRAMAS EN LENGUAJE DE MAQUINA DE LA Z-80

```
0000    3E    LD   A,80
0002    03    OUT(08),A
0004    3E    LD A,00
0006    D3    OUT (01),A
000A    03    OUT (02),A
000D    31    LD (1074),A
000D    31    LD SP,12FF
0010    CD    CALL 028D          se borran localidades de me-
0013    3A    LD A,(0F33)        moria.
0016    32    LD (1067),A
0019    CD    CALL 083A          se espera elección del código
001C    78    LD A,B
001D    FE    CP 0D
001F    28    JR Z,F039
0021    FE    CP 0A
0023    20    JR NZ,FD19
0025    3A    LD A,(1067)
0028    FE    CP 01
002A    20    JR NZ,F048
002C    3E    LD A,10
002E    32    LD (107C),A
0031    3E    LD A,0C            se fijan parámetros para el
0033    32    LD (107D),A        código de bloque
0036    C3    JP 00ED
0039    3A    LD A,(1067)
003C    CB    SLA A
003E    32    LD (1067),A        se almacena información que
0041    CB    BIT 3,A            encendera el respectivo led
0043    28    JR Z,FD19
0045    C3    JP 0013
0048    FE    CP 02
004A    20    JR NZ,F059
004C    3E    LD A,20
004E    32    LD (107C),A        se fijan parámetros para el
0051    3E    LD A.10            código de BCH
0053    32    LD (107D),A
0056    C3    JP 00ED
0059    CD    CALL 0AE0
005C    CD    CALL 083A          se espera valor de n;primer
005F    78    LD A,B             dígito.
0060    FE    CP 0A
0062    30    JR NC,F05C
0064    32    LD (1077),A
0067    32    LD (1070),A
006A    CD    CALL 00F7
006D    32    LD (1063),A
0070    CD    CALL 083A          se espera segundo dígito.
```

```
0073        78   LD A,B
0074        FE   CP 0A
0076        D2   JP NC,011D
0079        32   LD (1070),A
007C        32   LD (1078),A
007F        CD   CALL 00F7
0082        47   LD B,A
0083        3A   LD A,(1063)
0086        32   LD (107A),A
0089        78   LD A,B
008A        32   LD (1063),A
008D        3A   LD A,(107A)
0090        32   LD (1062),A
0093        CD   CALL 02A5          se calcula valor binario.
0096        32   LD (1072),A
0099        CD   CALL 083A          se espera tecla de control.
009C        78   LD A,B
009D        FE   CP 0B
009F        20   JR NZ,F099
00A1        CD   CALL 0AD0
00A4        CD   CALL 083A          se espera valor de k ; pri-
00A7        78   LD A,B             mer dígito.
00A8        FE   CP 0A
00AA        30   JR NC,F0A4
00AF        32   LD (1077),A
00B2        32   LD (1070),A
00B5        CD   CALL 00F7
00B8        32   LD (1063),A
00BB        CD   CALL 083A          se espera segundo dígito.
00BB        78   LD A,B
00BC        FE   CP 0A
00BE        30   JR NC,F113
00C0        32   LD (1078),A
00C3        32   LD (1070),A
00C6        CD   CALL 00F7
00C9        47   LD B,A
00CA        3A   LD A,(1063)
00CD        32   LD (107A),A
00D0        78   LD A,B
00D1        32   LD (1063),A
00D4        3A   LD A,(107A)
00D7        32   LD (1062),A
00DA        CD   CALL 02A5          se calcula su valor binario.
00DD        32   LD (1073),A
00E0        CD   CALL 083A          se espera tecla de control
00E3        78   LD A,B
00E4        FE   CP 0A
00E6        CA   JP Z,005C
00E9        FE   CP 0B
00EB        20   JR NZ,F0E0
00ED        CD   CALL 0130          se empieza el control de
00F0        C3   JP 0000            errores
```

SUBRUTINA BUFL

```
00F7        3A   LD A,(1070)        se recupera valor binario
00FA        E6   AND 0F
00FC        C6   ADD A,10
00FE        CF   LD L,A
00FF        26   LD H,0F
0101        7E   LD A,(HL)          se obtiene patron del display
0102        C9   RET
```

SUBRUTINA BUFH

```
0103        3A   LD A,(1070)        se recupera valor binario
0106        06   LD B,04
0108        CB   SRL A
010A        10   DJNZ F108
010C        C6   ADD A,10
010E        6F   LD L,A
010F        26   LD H,0F
0111        7E   LD A,(HL)          se obtiene patron del display
0112        C9   RET
0113        FE   CP 0A
0115        20   JR NZ,F0B8
0117        3A   LD A,(1077)
011A        C3   JP 00DD
011D        FE   CP 0A
011F        C2   JP NZ,0070
0122        3A   LD A,(1077)
0125        C3   JP 0096
0128        77   LD (HL),A
0129        10   DJNZ F0EE
012B        96   SUB (HL)
```

SUBRUTINA REAL

```
0130        00   NOP
0131        CD   CALL 028D          se pone ceros en localidades
0134        3A   LD A,(1067)
0137        FE   CP 04
0139        CA   JP Z,01F4
013C        06   LD B,00
013E        CD   CALL 0900          se recibe mensaje a codificar
0141        FE   CP 01
0143        28   JR Z,F131
0145        21   LD HL,107C
0148        3A   LD A,(1070)
014B        BE   CP (HL)            se determina si es dato vá-
014C        D2   JP NC,020E         lido.
014F        CD   CALL 0267          se pone en bloque de memória.
0152        CD   CALL 00F7          se busca patrón del display.
0155        32   LD (1061),A
0158        CD   CALL 0103          se busca patron del display.
015B        32   LD (1060),A
015E        CD   CALL 083A          se espera tecla de control.
```

```
0161    78    LD A,B
0162    FE    CP 0A
0164    28    JR Z,F131
0166    FE    CP 0B
0168    20    JR NZ,F15E
016A    CD    CALL 0347          se codifica el mensaje
016D    CD    CALL 025B
0170    21    LD HL,1027
0173    CD    CALL 0281          se almacena información que
0176    32    LD (1064),A        enciende los leds
0179    21    LD HL,102F
017C    CD    CALL 0281
017F    32    LD (1065),A
0182    CD    CALL 083A          se espera tecla de control
0185    78    LD A,B
0186    FE    CP 0B
0188    28    JR Z,F1C9
018A    FE    CP 0A
018C    28    JR Z,F131
018E    30    JR NC,F182
0190    32    LD (1076),A
0193    06    LD B,01
0195    CD    CALL 0900          se reciben posiciones erro-
0198    FE    CP 01              neas.
019A    28    JR Z,F131
019C    32    LD (107F),A
019F    21    LD HL,107D
01A2    3A    LD A,(1070)
01A5    BE    CP (HL)
01A6    30    JR NC,F280
01A8    32    LD (107A),A        dato binario de posición.
01AB    CD    CALL 0298          se modifica el dígito.
01AE    CD    CALL 0245
01B1    3A    LD A,(107F)
01B4    FE    CP 02
01B6    28    JR Z,F182
01B8    CD    CALL 083A          se espera tecla de control
01BB    78    LD A,B
01BC    FE    CP 0A
01BE    CA    JP Z,0131
01C1    FE    CP 0C
01C3    28    JR Z,F182
01C5    FE    CP 0B
01C7    20    JR NZ,F1B8
01C9    CD    CALL 0362          se calcula el síndrome
01CC    21    LD HL,1038
01CF    CD    CALL 0281
01D2    32    LD (1066),A
01D5    CD    CALL 083A          se espera tecla de control
01D8    78    LD A,B
01D9    FE    CP 0A
01DB    CA    JP Z,0131
01DE    FE    CP 0B
01E0    20    JR NZ,F1D5
```

```
01E2        CD   CALL 037D          se realiza la correción
01E5        CD   CALL 0245
01E8        CD   CALL 083A          se espera tecla de control
01EB        78   LD A,B
01EC        FE   CP 0A
01EE        CA   JP Z,0131
01F1        C3   JP 01E8
01F4        06   LD B,00
01F6        CD   CALL 0900
01F9        FE   CP 01
01FB        CA   JP Z,0131
01FE        3A   LD A,(1080)
0201        FE   CP 01
0203        28   JR Z,F214
0205        C3   JP 014F
0208        CD   CALL 021B          se presenta señal de error
020B        C3   JP 0182
020E        CD   CALL 021B
0211        C3   JP 013C
0214        CD   CALL 021B
0217        C3   JP 01F4
021A        C9   RET

SUBRUTINA ERR7

021B        21   LD HL,1060         se guardan datos del buffer
021E        11   LD DE,10A0         del display
0221        01   LD BC,0004
0224        ED   LDIR
0226        21   LD HL,0F34         en el buffer del display se
0229        11   LD DE,1060         almacena señal de error
022C        01   LD BC,0004
022F        ED   LDIR
0231        CD   CALL 083A          se espera tecla de control
0234        78   LD A,B
0235        FE   CP 0B
0237        20   JR NZ,F231
0239        21   LD HL,10A0         se repone datos en el buffer
023C        11   LD DE,1060
023F        01   LD BC,0004
0242        ED   LDIR
0244        C9   RET

SUBRUTINA PRES

0245        21   LD HL,1027
0248        CD   CALL 0281          se determinan ocho pri-
024B        32   LD (1064),A        meros digitos a mostrar
024E        21   LD HL,102F
0251        CD   CALL 0281          se determian ocho ulti-
0254        32   LD (1065),A        mos digitos a mostrar
0257        CD   CALL 0A00
025A        C9   RET
```

SUBRUTINA TRAN

```
025B    11   LD DE,1020        se traspasa un bloque de
025E    21   LD HL,1010        memoria.
0261    01   LD BC,0010
0264    ED   LDIR
0266    C9   RET
```

SUBRUTINA GBLO

```
0267    06   LD B,08
0269    21   LD HL,1000        se define bloque de memoria
026C    3A   LD A,(1070)
026F    4F   LD C,A
0270    CB   BIT 0,C
0272    28   JR Z,F279
0274    36   LD (HL),01        se almacena un dígito en una
0276    C3   JP 027B           localidad de memória
0279    79   LD (HL),00
027B    CB   SRL C
027D    23   INC HL            se define otra localidad
027E    10   DJNZ F270
0280    C9   RET
```

SUBRUTINA ARRE

```
0281    06   LD B,07
0283    7E   LD A,(HL)         se recupera un digito
0284    4F   LD C,A
0285    CB   SLA C             se lo pone en un byte
0287    2B   DEC HL
0288    7E   LD A,(HL)
0289    A9   XOR C
028A    10   DJNZ F284
028C    C9   RET
```

SUBRUTINA BORA

```
028D    21   LD HL,1000        se define bloque a borrar
0290    06   LD B,67
0292    36   LD (HL),0000      se almacena con ceros
0294    23   INC HL
0295    10   DJNZ F292
0297    C9   RET
```

SUBRUTINA MODI

```
029B    3A   LD A,(107A)       se recupera posición
029B    C6   ADD A,20
029D    6F   LD L,A
029E    26   LD H,10
02A0    7E   LD A,(HL)
02A1    EE   XOR 01            se modifica el bit.
02A3    77   LD (HL),A
```

```
02A4        C9    RET
02A5        3E    LD A,00
02A7        32    LD (1070)
02AA        3A    LD A,(1077)
02AD        CD    CALL 099C
02B0        3A    LD A,(1078)
02B3        CD    CALL 09AC
02B6        C9    RET

SUBRUTINA PROCE

0330        06    LD B,20
0332        3A    LD A,(106C)       información de columna
0335        80    ADD A,B           habilitada.
0336        6F    LD L,A
0337        26    LD H,0F
0339        3A    LD A,(106D)
033C        01    LD BC,0008
033F        ED    CPIR              se busca valor binario de
0341        2D    DEC L             tecla presionada.
0342        7D    LD A,L
0343        D6    SUB 20
0345        47    LD B,A            valor en binario de infor
0346        C9    RET               mación ingresada.

SUBRUTINA CODI

0347        3A    LD A,(1067)
034A        FE    CP 01
034C        20    NZ,F354
034E        CD    CALL 0410         se codifica código de bloque
0351        C3    JP 0631
0354        FE    CP 02
0356        20    JR NZ,F35E
0358        CD    CALL 064B         se codifica código BCH.
035B        C3    P 0361
035E        CD    CALL 0500         se codifica código cíclico
0361        C9    RET

SUBRUTINA SIND

0362        3A    LD A,(1067)
0365        FE    CP 01
0367        20    JR NZ,F36F
0369        CD    CALL 045A         síndrome de código de bloque.
036C        C3    JP 037C
036F        FE    CP 02
0371        20    JR NZ,F379
0373        CD    CALL 0B20         síndrome de código BCH.
0376        C3    JP 037C
0379        CD    CALL 05AB         síndrome de código cíclico.
037C        C9    RET
```

SUBRUTINA CORR

```
037D        3A    LD A,(1067)
0380        FE    CP 01
0382        20    JR NZ,F38A
0384        CD    CALL 04BE          se corrige a código de bloque
0387        C3    JP 039A
038A        FE    CP 02
038C        20    JR NZ,F397
038E        CD    CALL 06DA          se córrige a código BCH
0391        CD    CALL 075D
0394        C3    JP 039A
0397        CD    CALL 0A80          se corrige a código cíclico.
039A        C9    RET
```

SUBRUTINA CODB

```
0410        3A    LD A,(1000)
0413        32    LD (1018),A        V8
0416        4F    LD C,A
0417        3A    LD A,(1001)
041A        32    LD (1019),A        V19
041D        A9    XOR C
041E        47    LD B,A
041F        3A    LD A,(1002)
0422        32    LD (101A),A        V10
0425        A8    XOR B
0426        32    LD (1016),A        V6
0429        3A    LD A,(1003)
042C        32    LD (101B),A        V11
042F        A8    XOR B
0430        32    LD (1014),A        V4
0433        32    LD (1010),A        V0
0436        3A    LD A,(1010)
0439        A9    LD B,A
043A        32    LD A,(1002)
043D        A9    XOR B
043E        32    LD (1011)          V1
0441        32    LD (1015),A        V5
0444        3A    LD A,(1002)
0447        A9    XOR C
0448        32    LD (1017),A        V7
044B        3A    LD A,(1003)
044E        4F    LD C,A
044F        32    LD (1013),A        V3
0452        3A    LD A,(1002)
0455        A9    XOR C
0456        32    LD (1012),A        V2
0459        C9    RET
```

SUBRUTINA SIBL

```
045A        3A    LD A,(1028)
045D        4F    LD C,A
```

```
045E        3A    LD A,(1029)
0461        A9    XOR C
0462        4F    LD C,A
0463        3A    LD A,(102A)
0466        A9    XOR C
0467        47    LD B,A
0468        3A    LD A,(1026)
046B        A9    XOR B
046C        32    LD (1037),A          S6
046F        3A    LD A,(102B)
0472        A9    XOR C
0473        47    LD B,A
0474        3A    LD A,(1024)
0477        A8    XOR B
0478        32    LD (1035),A          S4
047B        3A    LD A,(1020)
047E        A8    XOR B
047F        32    LD (1031),A          S0
0482        3A    LD A,(1029)
0485        4F    LD C,A
0486        3A    LD A,(102A)
0489        A9    XOR C
048A        47    LD B,A
048B        3A    LD A,(1021)
048E        A8    XOR B
048F        32    LD (1032),A          S1
0492        3A    LD A,(1025)
0495        A8    XOR B
0496        32    LD (1036),A          S5
0499        3A    LD A,(102A)
049C        4F    LD C,A
049D        3A    LD A,(102B)
04A0        47    LD B,A
04A1        A9    XOR C
04A2        57    LD D,A
04A3        3A    LD A,(1022)
04A6        AA    XOR D
04A7        32    LD (1033),A          S2
04AA        3A    LD A,(1023)
04AD        A8    XOR B
04AE        32    LD (1034),A          S3
04B1        3A    LD A,(1028)
04B4        A9    XOR C
04B5        47    LD B,A
04B6        3A    LD A,(1027)
04B9        A8    XOR B
04BA        32    LD (1038),A          S7
04BD        C9    RET

SUBRUTINA CRRB

04BE        0E    LD C,00
04C0        06    LD B,08
04C2        21    LD HL,1038
```

```
04C5        CB   SLA C              se pone el síndrome en un
04C7        CB   BIT 0,(HL)         byte.
04C9        28   JR Z,F4D0
04CB        CB   SET 0,C
04CD        C3   JP 04D2
04D0        CB   RES 0,C
04D2        2B   DEC HL
04D3        10   DJNZ F4C5
04D5        06   LD B,0E           se direcciona localidad con
04D7        0A   LD A,(BC)         ayuda del síndrome.
04D8        57   LD D,A
04D9        E6   AND 0F
04DB        CD   CALL 04EB         se determina si hay error
04DE        7A   LD A,D
04DF        E6   AND F0
04E1        06   LD B,04
04E3        CB   SRL A
04E5        10   DJNZ F4E3
04E7        CD   CALL 04EB         se determina si hay error
04EA        C9   RET
```

SUBRUTINA RECO

```
04EB        FE   CP 0F
04ED        28   JR Z,04FF
04EF        C6   ADD A,20          se localiza dígito erroneo.
04F1        6F   LD L,A
04F2        26   LD H,10
04F4        CB   BIT 0,(HL)        se realiza la corrección
04F6        28   JR Z,F4FD
04F8        CB   RES 0,(HL)
04FA        C3   JP 04FF
04FD        CB   SET 0,(HL)
04FF        C9   RET
```

SUBRUTINA CODC

```
0500        21   LD HL,1000        se alista a los dígitos
0503        CD   CALL 080B         del mensaje a codificar
0506        21   LD HL,1000
0509        CD   CALL 0597
050C        11   LD   DE,1010
050F        21   LD   HL,1000
0512        3E   LD A,90           se programa al periférico.
0514        D3   OUT (03),A        para enviar datos.
0516        3E   LD A,00
0518        D3   OUT (01),A
051A        3E   LD A,00
051C        4F   LD C,A
051D        D3   OUT  (02),A
051F        3A   LD A,(1073)       se determina valor de k
0522        47   LD B,A
0523        CB   SET 6,C
```

```
0526        D3    OUT  (02),A        se envia señal de clear.
0528        7E    LD A,(HL)
0529        CB    BIT 0,A
052B        28    JR Z,DD32
052D        CB    SET 4,C
052F        C3    JP 0534
0532        CB    RES  4,C
0534        79    LD A,C
0535        D3    OUT (02),A         se envia digito a codificar
0537        CB    SET 7,C
0539        79    LD A,C
053A        D3    OUT (02),A         se envia señal de reloj
053C        CD    CALL 0584
053F        23    INC HL             se escoje otro dígito.
0540        10    DJNZ DD28
0542        3A    LD A,(1073)
0545        47    LD B,A
0546        3A    LD A,(1072)
0549        90    SUB B
054A        00    NOP
054B        47    LD B,A
054C        CB    SET 5,C
054E        79    LD A,C
054F        D3    OUT (02),A         se envia señal de select
0551        CD    CALL 0584
0554        CB    SET  7,C
0556        10    DJNZ DD51
0558        79    LD A,C
0559        CD    CALL 0AF0
055C        21    LD HL,1010
055F        CD    CALL 080B          se obtienen ocho primeros
0562        5F    LD E,A             bits del código vector
0563        21    LD HL,1018
0566        CD    CALL 080B          se obtienen ocho ultimos
0569        57    LD D,A             bits del código vector.
056A        21    LD HL,1010
056D        7A    LD A,D
056E        CD    CALL 0597
0571        7B    LD A,E
0572        21    LD HL,1018
0575        CD    CALL 0597
0578        C9    RET
0579        06    LD B,07
057B        7E    LD A,(HL)
057C        4F    LD C,A
057D        CB    SLA C
057F        7E    LD A,(HL)
0580        A9    XOR C
0581        10    DJNZ DD7C
0583        C9    RET

SUBRUTINA SACA

0584        CB    SET  2.C
```

```
0586        79    LD A,C
0587        D3    OUT (02),A
0589        DB    IN A,(00)          se recupera dato de la tar-
058B        E6    AND 01             jeta.
058D        12    LD (DE),A          se almacena el dato.
058E        13    INC DE
058F        CB    RES 7,C
0591        CB    RES 2,C
0593        79    LD A,C
0594        D3    OUT (02)           se envía señal de reloj
0596        C9    RET

SUBRUTINA GBL1

0597        4F    LD C,A             se recupera dato en binario
0598        06    LD B,08
059A        CB    BIT 0,C
059C        28    JR Z,DDA3
059E        36    LD (HL),01         se almacena un dígito en
05A0        C3    JP 05A5            una localidad de memoria.
05A3        36    LD (HL),00
05A5        CB    SRL C
05A7        23    INC HL
05A8        10    DJNZ DD9A
05AA        C9    RET

SUBRUTINA SICI

05AB        3A    LD A,(1072)
05AE        47    LD B,A
05AF        3E    LD A,90
05B1        D3    OUT (03),A         se programa al periferico
05B3        3E    LD A,00
05B5        4F    LD C,A
05B6        D3    OUT (01),A
05B8        D3    OUT (02),A
05BA        CB    SET 6,C
05BC        79    LD A,C
05BD        D3    OUT (02),A         se envia señal de CL
05BF        3A    LD A,(1072)
05C2        3D    DEC A
05C3        C6    ADD A,20
05C5        6F    LD L,A
05C6        26    LD H,10
05C8        7E    LD A,(HL)          se recupera dígito a decodi-
05C9        CB    BIT 0,A            ficar.
05CB        28    JR Z,D2
05CD        CB    SET 3,C
05CF        C3    JP   05D4
05D2        CB    RES  3,C
05D4        79    LD A,C
05D5        D3    OUT (02),A         se envia dato a decodificar
05D7        CB    SET 7,C
05D9        79    LD A,C
```

```
05DA    D3   OUT (02),A        se envia señal de reloj.
05DC    CB   RES 7,C
05DE    79   LD A,C
05DF    D3   OUT (02),A        se envia señal de reloj
05E1    2B   DEC  HL
05E2    10   DJNZ DDC8
05E4    CB   SET 2,C
05E6    79   LD A,C
05E7    D3   OUT (02),A        se habilita buffer.
05E9    DB   IN A,(00)         se recupera síndrome.
05EB    57   LD D,A
05EC    3E   LD A,08
05EE    47   LD B,A
05EF    21   LD HL,1031
05F2    7A   LD A,D
05F3    CB   BIT 2,A           se almacena un dígito del
05F5    28   JR Z,DDFC         síndrome.
05F7    36   LD (HL),01
05F9    C3   JP 05FE
05FC    36   LD (HL),00
05FE    CB   SRL A
0600    23   INC HL
0601    10   DJNZ DDF3
0603    CB   RES 2,C
0605    79   LD A,C
0606    D3   OUT (02),A        se desabilita buffer.
0608    3A   LD A,(1072)
060B    47   LD B,A
060C    CB   SET 5,C
060E    79   LD A,C
060F    D3   OUT (02),A        se envia señal de selección
0611    3A   LD A,(1072)
0614    C6   ADD A,BF
0616    6F   LD L,A
0617    26   LD H,10
0619    CB   SET 2,C
061B    79   LD A,C
061C    D3   OUT (02),A        se habilita buffer.
061E    DB   IN   A,(00)       se recupera dígito codi-
0620    CB   BIT 1,A           ficado.
0622    28   JR Z,DE29
0624    36   LD (HL),00
0626    C3   JP 062B
0629    36   LD (HL),00
062B    CB   RES 2,C
062D    79   LD A,C
0630    D3   OUT (02),A        se desabilita buffer
0630    2B   DEC HL
0631    CB   SET 7,C
0633    79   LD A,C
0634    D3   OUT (02),A        se envia señal de reloj
0636    10   DJNZ DE3B
0638    C3   JP 0643
063B    CB   RES 7,C
```

```
063D        79    LD A,C
063E        D3    OUT (02),A          se envia señal de reloj.
0640        C3    JP 0619
0643        C9    RET

SUBRUTINA COBC

064B        3E    LD A,19
064D        21    LD (104B),A
0650        21    LD HL,1000
0653        11    LD DE,101A
0656        01    LD BC,0005
0659        ED    LDIR
065B        21    LD HL,1000          se definen dígitos a codi-
065E        11    LD DE,1046          ficar.
0661        01    LD BC,0005
0664        ED    LDIR
0666        3A    LD A,(1049)         se realizan operaciones
0669        4F    LD C,A              con las salidas de los
066A        3A    LD A,(104A)         registros.
066D        A9    XOR C
066E        47    LD B,A
066F        3A    LD A,(1047)
0672        A8    XOR B
0673        4F    LD C,A
0674        3A    LD A,(104B)
0677        26    LD H,10
0679        6F    LD L,A
067A        71    LD (HL),C           se almacena dígito codi-
067B        2B    DEC HL              ficado
067C        7D    LD A,L
067D        FE    CP 0F
067F        28    JR Z,DEA1
0681        32    LD (104B),A
0684        79    LD A,C
0685        32    LD (1045),A
0688        21    LD HL,1045          se desplaza los dígitos a
068B        11    LD DE,1040          codificar
068E        01    LD BC,0005
0691        ED    LDIR
0693        C3    JP 0666
0696        C9    RET

SUBRUTINA SIBC

06A2        3E    LD A,06             se define exponencialmente
06A4        32    LD (1053),A         elemento a evaluar.
06A7        11    LD DE,102E
06AA        3A    LD A,(1053)
06AD        32    LD (1051),A
06B0        1A    LD A,(DE)
06B1        CD    CALL 07CF           se empieza a evaluar el po-
06B4        4F    LD C,A              linomio.
06B5        1B    DEC DE
```

```
06B6    1A    LD A,(DE)
06B7    A9    XOR C
06B8    4F    LD C,A
06B9    7B    LD A,E
06BA    FE    CP 20
06BC    20    JR NZ,DED5
06BE    3A    LD A,(1053)
06C1    C6    ADD A,30
06C3    6F    LD L,A
06C4    26    LD H,10
06C6    71    LD (HL),C          se almacena un elemento del
06C7    3A    LD A,(1053)        síndrome.
06CA    3D    DEC A
06CB    32    LD (1053),A        se escoje otro elemento pa-
06CE    FE    CP 00              ra la evaluación.
06D0    20    JR NZ,DEA7
06D2    C3    JP 06D9
06D5    79    LD A,C
06D6    C3    JP 06B1
06D9    C9    RET
```

SUBRUTINA CRBC

```
06DA    3A    LD A,(1031)
06DD    CD    CALL 07A8
06E0    3A    LD A,(1032)
06E3    CD    CALL 07CF
06E6    4F    LD C,A
06E7    3A    LD A,(1033)
06EA    A9    XOR C
06EB    32    LD (104D),A        d1
06EE    CD    CALL 07A8
06F1    3A    LD A,(1033)
06F4    CD    CALL 07FC
06F7    32    LD (1039),A
06FA    3A    LD A,(1031)
06FD    CD    CALL 07B2
0700    3A    LD A,(1039)
0703    CD    CALL 07CF
0706    32    LD (1039),A
0709    3A    LD A,(1031)
070C    CD    CALL 07A8
070F    3A    LD A,(1034)
0712    CD    CALL 07CF
0715    47    LD B,A
0716    3A    LD A,(1039)
0719    A8    XOR B
071A    47    LD B,A
071B    3A    LD A,(1035)
071E    A8    XOR B
071F    32    LD (104E),A        d2
0722    3A    LD A,(104D)
0725    CD    CALL 07B2
0728    3A    LD A,(104E)
```

```
072B    CD   CALL 07CF
072E    32   LD (104F),A          d2/d1
0731    CD   CALL 07A8
0734    3A   LD A,(1031)
0737    CD   CALL 07CF
073A    32   LD (103E),A          sigma 3
073D    3A   LD A,(1031)
0740    CD   CALL 07B2
0743    3A   LD A,(104D)
0746    CD   CALL 07CF
0749    47   LD B,A
074A    3A   LD A,(104F)
074D    A8   XOR B
074E    32   LD (103D),A          sigma 2
0751    3A   LD A,(1031)
0754    32   LD (103C),A          sigma 1
0757    3E   LD A,01
0759    32   LD (103B),A          sigma 0
075C    00   NOP
075D    3E   LD A,0F              se define exponencialmente
075F    32   LD (104C),A          elemento a evaluar
0762    11   LD DE.103E
0765    3A   LD A,(104C)          se empieza evaluación.
0768    D6   SUB 01
076A    32   LLD (1051),A
076D    1A   LD A,(DE)
076E    CD   CALL 07CF
0771    4F   LD C,A
0772    1B   DEC DE
0773    1A   LD A,(DE)
0774    A9   XOR C
0775    4F   LD C,A
0776    7B   LD A,E
0777    FE   CP 3B
0779    20   JR NZ,DFA3
077B    79   LD A,C
077C    FE   CP 00                se determina si es raiz.
077E    20   JR NZ,DE95
0780    3A   LD A,(104C)
0783    D6   SUB 01
0785    6F   LD L.A
0786    26   LD H,0F
0788    7E   LD A,(HL)
0789    CD   CALL 07B2            se saca el inverso del ex-
078C    C6   ADD A,20             ponente de la raiz.
078E    6F   LD L,A
078F    26   LD H,10
0791    7E   LD A,(HL)            se recupera dígito a corre-
0792    EE   XOR 01               gir.
0794    77   LD (HL),
0795    3A   LD A,(104C)
0798    3D   DEC A
0799    32   LD (104C),A
079C    FE   CP 00                se termina evaluación.
```

```
079E        20    JR NZ,DF62
07A0        C3    JP 07A7
07A3        79    LD A,C
07A4        C3    JP 076E
07A7        C9    RET

SUBRUTINA MUL

07A8        32    LD (1050),A
07AB        CD    CALL 07FE
07AE        32    LD (1051),A          se almacena el exponente
07B1        C9    RET

SUBRUTINA DIVI

07B2        32    LD (1050),A
07B5        FE    CP 00
07B7        20    JR NZ,DFBE
07B9        3E    LD A,0F
07BB        C3    JP 07CB
07BE        CD    CALL 07FE            se almacena el exponente
07C1        4F    LD C,A
07C2        3E    LD A,0F
07C4        91    SUB C                se saca el inverso
07C5        FE    CP 0F
07C7        20    JR NZ.DFCB
07C9        3E    LD A,00
07CB        32    LD (1051),A          se almacena el nuevo expo-
07CE        C9    RET                  nente

SUBRUTINA OPE

07CF        32    LD (1050),A
07D2        CD    CALL 07FE
07D5        32    LD (1052),A
07F8        FE    CP 0F
07DA        28    JR Z,DFF3
07DC        3A    LD A,(1051)          exponente del multiplicador
07DF        FE    CP 0F                o del divisor
07E1        28    JR Z,DFF3
07E3        4F    LD C,A
07E4        3A    LD A,(1052)
07E7        81    ADD A,C              se suman exponentes
07E8        FE    CP 0F
07EA        30    JR NC,DFF8
07EC        26    LD H,0F
07EE        6F    LD L,A
07EF        7E    LD A,(HL)            se obtiene resultado en bi-
07F0        C3    JP 07FD              nario
07F3        3E    LD A,00
07F5        C3    JP 07FD
07F8        D6    SUB 0F
07FA        C3    JP 07E8
07FD        C9    RET
```

SUBRUTINA XPON

```
07FE        21   LD HL,0F00
0801        3A   LD A,(1050)        se recupera dato en binario
0804        ED   CPI
0806        20   JR NZ,E001
0808        2D   DEC L
0809        7D   LD A,L             se obtiene el exponente
080A        C9   RET
```

SUBRUTINA ARE1

```
080B        06   LD B,07
080D        7E   LD A,(HL)
080E        4F   LD C,A
080F        CB   SLA C
0811        23   INC HL
0812        7E   LD A,(HL)
0813        A9   XOR C              se pone en un byte ocho
0814        A0   DJNZ E00E          digitos.
0816        C9   RET
```

SUBRUTINA SCAN

```
083A        00   NOP
083B        3E   LD A,80            se programa al periférico
083D        D3   OUT (03)           para enviar información
083F        11   LD DE,1060
0842        06   LD B,08
0844        0E   LD C,01
0846        3E   LD A,00
0848        D3   OUT (00),A
084A        79   LD A,C
084B        D3   OUT (01).A
084D        1A   LD A,(DE)
084E        D3   OUT (00),A         se saca un patron que en-
0850        3E   LD A,80            cendera los displays.
0852        32   (1069),A
0855        3A   LD A,(1069)
0858        3D   DEC A
0859        32   LD (1069),A
085C        20   JR NZ,E055
085E        10   DJNZ E063
0860        C3   JP 086D
0863        3E   LD A,00
0865        D3   OUT (00),A
0867        CB   SLA C
0869        13   INC DE             se elige otro patron
086A        C3   JP 084A
086D        3E   LD A,00
086F        D3   OUT (01),A
0871        3E   LD A,90            se programa al periferico
0873        D3   OUT (03),A         para recibir información.
```

```
0875        3A   LD A,(1074)
0878        CB   SET 0,A
087A        D3   OUT  (02),A
087C        3E   LD A,00
087E        32   LD (106C),A
0881        DB   IN A,(00)          se muestrea primera fila.
0883        FE   CP FF
0885        20   JR NZ,E0A6
0887        3E   LD A,08
0889        32   LD (106C),A
088C        3A   LD A,(1074)8
088F        CB   SET 1,A
0891        D3   OUT (02),A
0893        DB   IN A,(00)          se muestrea segunda fila
0895        FE   CP FF
0897        20   JR NZ,E0A6
0899        3E   LD A,00            no hay tecla presionada.
089B        32   LD (106B),A
089E        3A   LD A,(1074)
08A1        D3   OUT (02),A
08A3        C3   JP 083A
08A6        32   LD (106D),A
08A9        3A   LD A,(1074)
08AC        D3   OUT (02),A
08AE        3A   LD A,(106B)
08B1        FE   CP 01
08B3        28   JR Z,E03B
08B5        CD   CALL 0330          se obtiene equivalente bi-
08B8        3E   LD A,01            nario de tecla presionada.
08BA        32   LD (106B),A
08BD        C9   RET
```

SUBRUTINA TREN

```
0900        3E   LD A,00
0902        32   LD (107B),A
0905        32   LD (1077),A
0908        32   LD (1078),A
090B        78   LD A,B
090C        FE   CP   00
0903        20   JR NZ,E158
0910        CD   CALL 083A          se espera ingreso de
0913        78   LD A,B             un dato numérico.
0914        FE   CP 0A
0916        30   JR NC,E110
0918        32   LD (1078)
091B        CD   CALL 083A          se espera ingreso de
091E        78   LD A,B             un dato numérico.
091F        FE   CP 0A
0921        30   JR NC,E15E
0923        3A   LD A,(1078)
0926        32   LD (1077),A
0929        78   LD A,B
092A        32   LD (1078),A
```

```
092D    CD   CALL 083A        se espera el ingreso de
0930    78   LD A,B           un dato numérico.
0931    FE   CP 0A
0933    30   JR NC,E16F
0935    3A   LD A,(1077)
0938    32   LD (107B),A
03B9    3A   LD A,(1078)
093E    32   LD (1077),A
0941    78   LD A,B
0942    32   LD (1078),A
0945    CD   CALL 09B8
0948    CD   CALL 097E
094B    3E   LD A,00
094D    C3   JP   097D
0950    CD   CALL 097E        se obtiene equivalente bi-
0953    3E   LD A,02          nario.
0955    C3   JP 097D
0958    3A   LD A,(1076)
095B    C3   JP 0918
095E    FE   CP 0B            se determina la tecla de
0960    28   JR Z,E145        control presionada.
0962    FE   CP 0C
0964    28   JR Z,E150
0966    FE   CP 0A
0968    20   JR NZ,E11B
096A    3E   LD A,01
096C    C3   JP 097D
097F    FE   CP 0B            se determina la tecla de
0971    28   JR Z,E145        control presionada.
0973    FE   CP 0C
0975    28   JR Z,E150
0977    FE   CP 0A
0979    20   JR NZ,E12D
097B    3E   LD A,01
097D    C9   RET
```

SUBRUTINA BIDE

```
097E    3A   LD A,(107B)
0981    CD   CALL 0991        se calculan las centenas
0984    3A   LD A,(1077)
0987    CD   CALL 099C        se calculan las decenas
098A    3A   LD A,(1078)
098D    CD   CALL 09AC        se obtiene el valor ingresa-
0990    C9   RET              do en binario.
0991    4F   LD C,A
0992    06   LD B,63
0994    79   LD A,C
0995    81   ADD A,C          se suman cien veces el
0996    10   DJNZ E195        mismo número binario.
0998    32   LD (1070),A
099B    C9   RET
099C    4F   LD C,A
099D    06   LD B,09
```

```
099F        79    LD A,C
09A0        81    ADD A,C              se suman diez veces el
09A1        10    DJNZ E1A0            mismo número binario.
09A3        47    LD B,A
09A4        3A    LD (1070),A
09A7        80    ADD A,B
09A8        32    LD (1070),A
09AB        C9    RET
09AC        47    LD B,A
09AD        3A    LD A,(1070)
09B0        80    ADD A,B              se obtiene el resultado
09B1        32    LD (1070),A          total en binario.
09B4        32    LD (1070),A
09B7        C9    RET
09B8        3A    LD A,(107B)          se determina valides de
09BB        FE    CP 03                las centenas ingresadas.
09BD        30    JR NC,F9D5
09BF        FE    CP 02
09C1        38    JR C,F9DD
09C3        3A    LD A,(1077)          se determina valides de
09C6        FE    CP 06                las decenas ingresadas.
09C8        30    JR NC,F9D5
09CA        FE    CP 05
09CC        38    JR C,F9DD
09CE        3A    LD A,(1078)
09D1        FE    CP 06
09D3        38    JR C,F9DD
09D5        3E    LD A,01
09D7        32    LD (1080),A
09DA        C3    JP 09E2
09DD        3E    LD A,00
09DF        32    LD (1080),A
09E2        C9    RET

SUBRUTINA VARI

0A00        3A    LD A,(1067)
0A03        FE    CP 01
0A05        20    JR NZ,FA1E
0A07        3A    LD A,(1065)          bits del mensaje del
0A0A        E6    AND 0F               código de bloque.
0A0C        32    LD (1070),A
0A0F        CD    CALL 00F7
0A12        32    LD (1063),A          se obtiene su patron
0A15        CD    CALL 0103
0A18        32    LD (1062),A
0A1B        C3    JP 0A55
0A1E        FE    CP 02
0A20        20    JR NZ,FA30
0A22        3A    LD A,(1065)
0A25        06    LD B,02
0A27        CB    SRL A
0A29        10    DJNZ FA27
0A2B        E6    AND 1F               bits del mensaje del có.
```

```
0A2D    C3    JP 0A0C              digo BCH.
0A30    3A    LD A,(1073)
0A33    47    LD B,A
0A34    3A    LD A,(1072)
0A37    90    SUB B
0A38    47    LD B,A
0A39    3A    LD A,(1064)
0A3C    CB    SRL A
0A3E    10    DJNZ FA3C
0A40    32    LD (1070),A          primeros cuatro bits
0A43    CD    CALL 00F7            del código cíclico.
0046    32    LD (1063),A
0A49    3A    LD A,(1065)
0A4C    32    LD (1070),A          cuatro últimos bits del
0A4F    CD    CALL 00F7            mensaje del código.
0A52    32    LD (1062),A
0A55    C9    RET
0A60    E6    AND 1F
0A62    C3    JP 0A0C

0A80    21    LD HL,10C0
0A83    11    LD DE,1020
0A86    01    LD BC,000F
0A89    ED    LDIR
0A8B    C9    RET

0AD0    3E    LD A,38
0AD2    32    LD (1060),A          se almacena patron de L
0AD5    3E    LD A,00
0AD7    32    LD (1063),A
0ADA    32    LD (1062),A
0ADD    C9    RET

0AE0    3E    LD A,54
0AE2    32    LD (1060),A          se almacena patron N
0AE5    3E    LD A,48
0AE7    32    LD (1061),A          se almacena patron =
0AEA    C9    RET

0AF0    06    LD B,0F
0AF2    21    LD HL,10E0
0AF5    3E    LD A,00
0AF7    77    LD (HL),A
0AF8    23    INC HL
0AF9    10    DJNZ FAF5
0AFB    3A    LD A,(1073)
0AFE    47    LD B,A
0AFF    3A    LD A,(1072)
0B02    90    SUB B                número de dígitos del men-
0B03    C6    ADD A,E0             saje en el código cíclico.
0B05    5F    LD E,A
0B06    16    LD D,10
0B08    21    LD HL,1010
0B0B    01    LD BC.0010
```

```
0B0E        ED    LDIR
0B10        11    LD DE,1010
0B13        21    LD HL,10E0
0B16        01    LD BC,00010
0B19        ED    LDIR
0B1B        C9    RET

0B20        CD    CALL 06A2
0B23        CD    CALL 0B30
0B26        C9    RET

SUBRUTINA MUES

0B30        11    LD DE,10A0          se almacenan resultados que
0B33        21    LD HL,1060          estan en el buffer.
0B36        01    LD BC,0004
0B39        ED    LDIR
0B3B        3E    LD A,6D
0B3D        32    LD (1060),A         se almacena patrón de S
0B40        21    LD HL,1031
0B43        7E    LD A,(HL)           se recupera elemento  del
0B44        32    LD (1066),A         sindrome
0B47        7D    LD A,L
0B48        32    LD (1082),A
0B4B        D6    SUB 30
0B4D        32    LD (1070),A
0B50        CD    CALL 00F7
0B53        32    LD (1061),A
0B56        3E    LD A,48
0B58        32    LD (1062),A         se almacena patrón de = .
0B5B        3E    LD A,00
0B5D        32    LD (1063),A
0B60        CD    CALL 083A           se espera selección del ele-
0B63        78    LD A,B              mento del sindrome
0B64        FE    CP 0D
0B66        28    JR Z,E36F
0B68        FE    CP 0B
0B6A        28    JR Z,E37E
0B6C        C3    JP 0B60
0B6F        3A    LD A,(1082)
0B72        6F    LD L,A
0B73        26    LD H,10
0B75        23    INC HL              se elige otro elemento.
0B76        7D    LD A,L
0B77        FE    CP 37
0B79        28    JR Z,E340
0B7B        C3    JP 0B43
0B7E        11    LD DE,1060          se reponen resultados en
0B81        21    LD HL,10A0          el buffer.
0B84        01    LD BC,0004
0B87        ED    LDIR
0B89        C9    RET
```

CIRCUITO MULTIPLICADOR DE POLINOMIOS

Multiplicar dos polinomos a(x) y h(x) representados por;

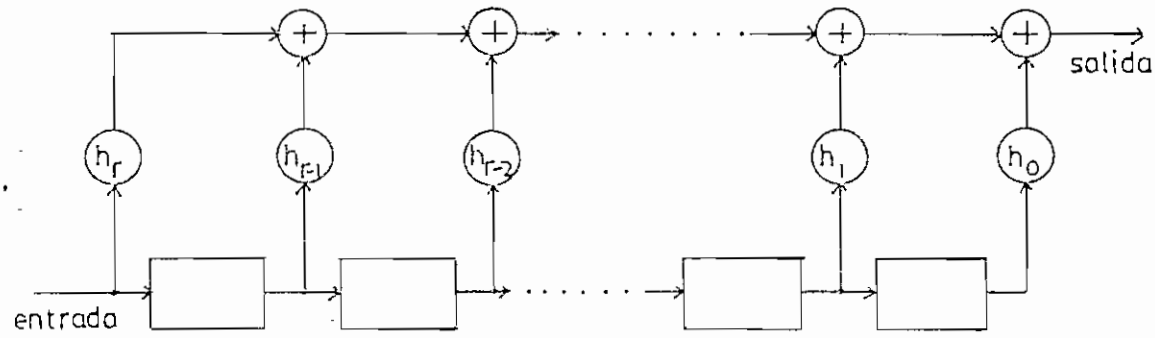
$a(x) = a_0 + a_1X + a_2X^2 + \ldots\ldots\ldots\ldots a_kX^k$

$h(x) = h_0 + h_1X + h_2X^2 + \ldots\ldots\ldots\ldots h_rX^r$

equivale a obtener la siguiente expresión

$$a_0h_0 + (a_0h_1 + h_0a_1)X + \ldots\ldots\ldots\ldots (a_{k-2}h_r + a_{k-1}h_{r-1} + a_kh_{r-2})X^{k+r-2} + (a_{k-1}h_r + a_kh_{r-1})X^{k+r-1} + (a_kh_r)X^{k+r}$$

El siguiente circuito definido en base de los coeficientes de h(x), multiplica cualquier polinomio por h(x).



Con las salidas de los registros iguales a cero, cuando ingresamos el primer dígito que corresponde al coeficiente de X^k del polinomio a(x), el correspondiente dígito a la salida del circuito será igual a a_kh_r.

Al primer desplazamiento a la salida se tiene un dígito igual a $a_kh_{r-1} + a_{k-1}h_r$

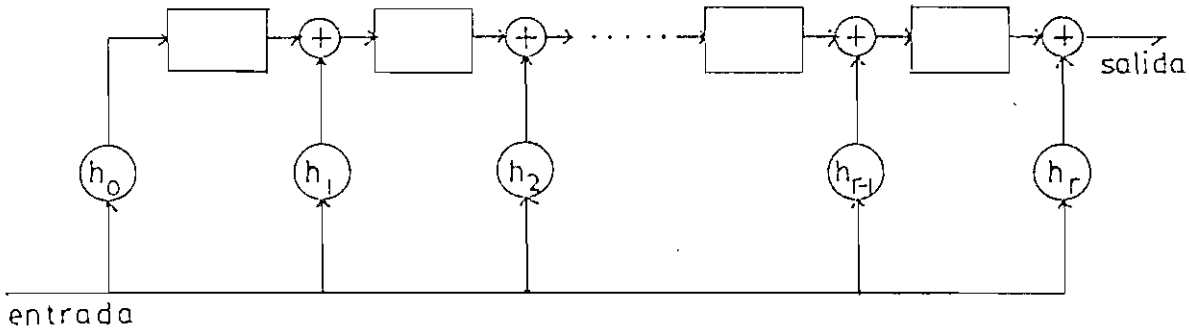
Al segundo desplazamiento se tiene que el valor en la salida del circuito es $a_{k-2}h_r + a_{k-1}h_{r-1} + a_kh_{r-2}$

Cuando el coeficiente de X se halla a la salida del último registro de desplazamiento, en la salida del circuito se

tiene un valor dado por $a_1 h_0$.

Al realizar el último desplazamiento, de tal manera que el último dígito en entrar se halla en la salida del último registro de desplazamiento, en la salida se tiene que el valor viene dado por $a_0 h_0$.

Otro circuito que realiza la multiplicación de h(x) por cualquier otro polinomio es el siguiente:



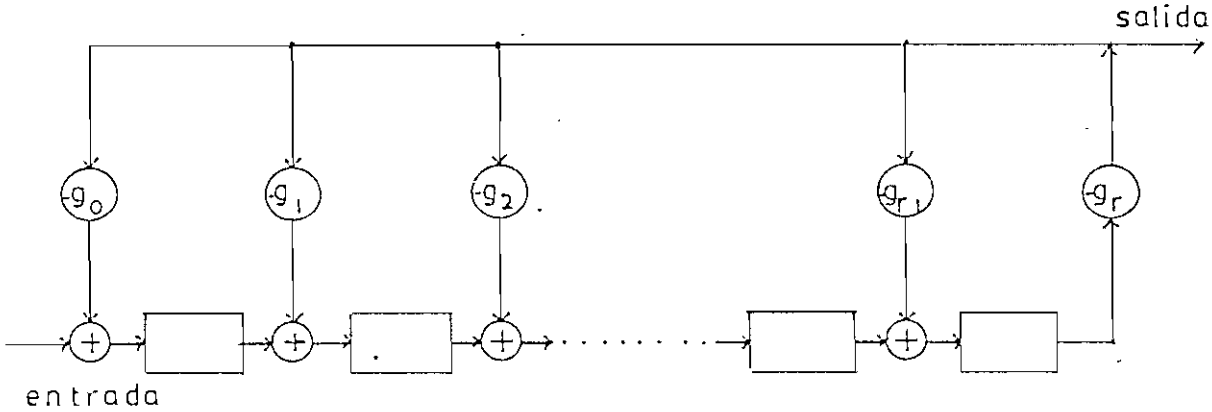
CIRCUITO QUE DIVIDE POLINOMIOS

Dividir un polinomio d(x) para otro polinomio g(x) con;

$d(x) = d_0 + d_1 X + d_2 X^2 + \ldots\ldots\ldots\ldots d_n X^n$

$g(x) = g_0 + g_1 X + g_2 X^2 + \ldots\ldots\ldots\ldots g_r X^r$

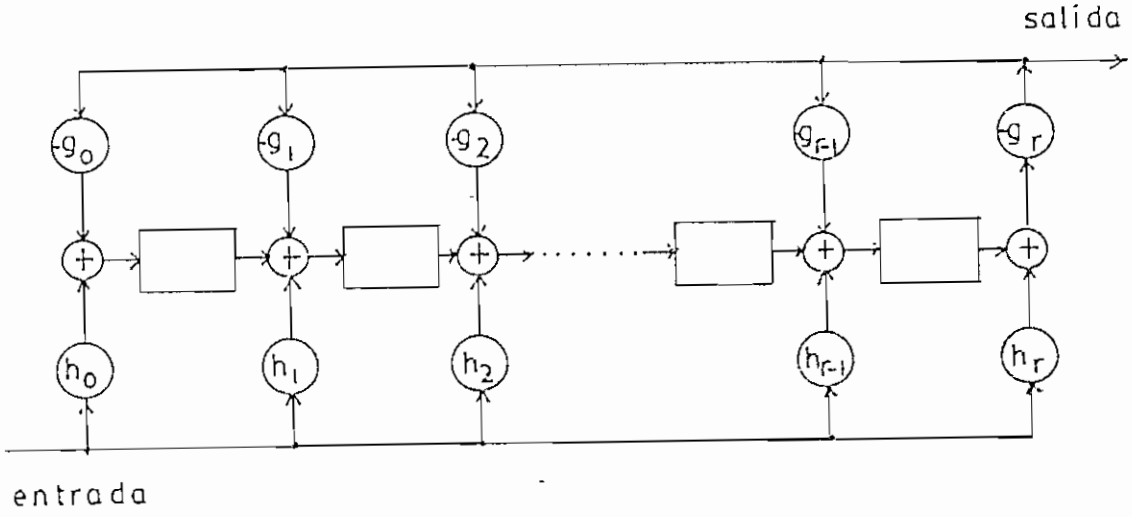
puede ser realizado con el siguiente circuito.



Este circuito divide cualquier polinomio por g(x) con $n \geq r$.

CIRCUITO MULTIPLICADOR Y DIVISOR

Multiplicar cualquier polinomio por h(x) y dividirlo para g(x), puede ser realizado por el siguiente circuito en forma conjunta.



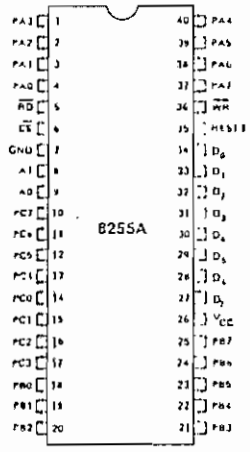
Este circuito resulta de una combinación de los circuitos anteriormente vistos.

8255A/8255A-5
PROGRAMMABLE PERIPHERAL INTERFACE

- MCS-85™ Compatible 8255A-5
- 24 Programmable I/O Pins
- Completely TTL Compatible
- Fully Compatible with Intel® Microprocessor Families
- Improved Timing Characteristics
- Direct Bit Set/Reset Capability Easing Control Application Interface
- 40-Pin Dual In-Line Package
- Reduces System Package Count
- Improved DC Driving Capability

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

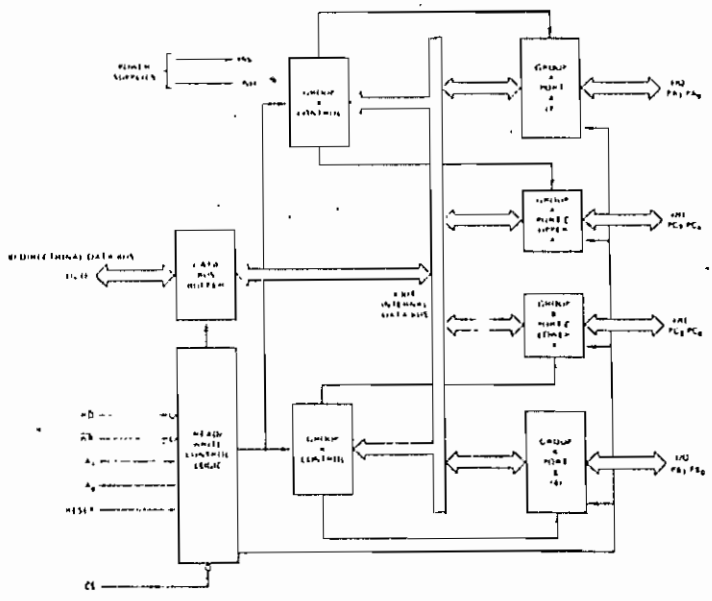
PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255A BLOCK DIAGRAM



8255A FUNCTIONAL DESCRIPTION

General

The 8255A is a programmable peripheral interface (PPI) device designed for use in Intel® microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 8255A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 8255A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

($\overline{CS}$)

Chip Select. A "low" on this input pin enables the communiction between the 8255A and the CPU.

($\overline{RD}$)

Read. A "low" on this input pin enables the 8255A to send the data or status information to the CPU on the data bus. In essence, it allows the CPU to "read from" the 8255A.

($\overline{WR}$)

Write. A "low" on this input pin enables the CPU to write data or control words into the 8255A.

(A_0 and A_1)

Port Select 0 and Port Select 1. These input signals, in conjunction with the RD and WR inputs, control the selection of one of the three ports or the control word registers. They are normally connected to the least significant bits of the address bus (A_0 and A_1).

8255A BASIC OPERATION

A_1	A_0	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0	0	0	1	0	PORT A → DATA BUS
0	1	0	1	0	PORT B → DATA BUS
1	0	0	1	0	PORT C → DATA BUS
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	DATA BUS → PORT A
0	1	1	0	0	DATA BUS → PORT B
1	0	1	0	0	DATA BUS → PORT C
1	1	1	0	0	DATA BUS → CONTROL
					DISABLE FUNCTION
X	X	X	X	1	DATA BUS → 3-STATE
1	1	0	1	0	ILLEGAL CONDITION
X	X	1	1	0	DATA BUS → 3-STATE

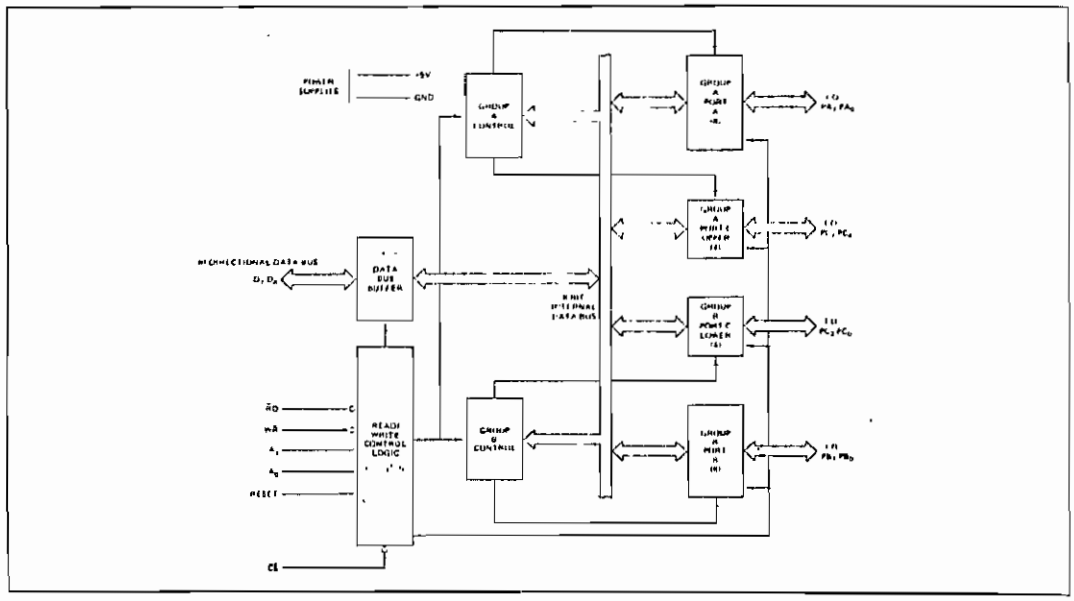


Figure 3. 8255A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions

(RESET)

Reset. A "high on this input clears the control register and all ports (A, C, C) are set to the input mode.

Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 8255A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 8255A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A – Port A and Port C upper (C7-C4)
Control Group B – Port B and Port C lower (C3-C0)

The Control Word Register can Only be written into. No Read operation of the Control Word Register is allowed.

Ports A, B, and C

The 8255A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 8255A.

Port A. One 8-bit data output latch/buffer and one 8-bit data input latch.

Port B. One 8-bit data input/output latch/buffer and one 8-bit data input buffer.

Port C. One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B.

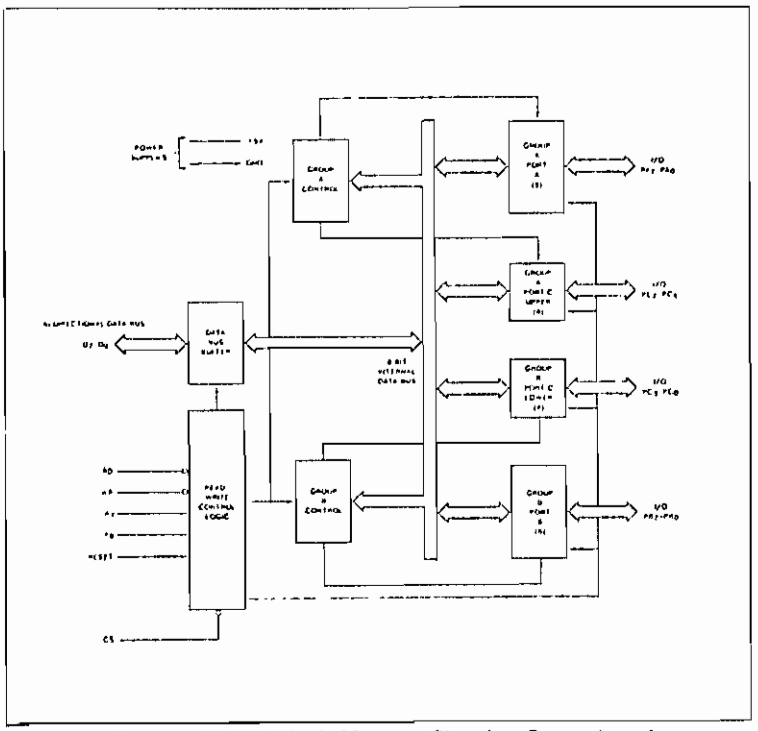
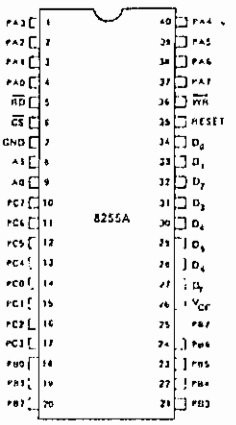


Figure 4. 8225A Block Diagram Showing Group A and Group B Control Functions

PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	0 VOLTS

8255A OPERATIONAL DESCRIPTION

Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 -- Basic Input/Output
Mode 1 -- Strobed Input/Output
Mode 2 -- Bi-Directional Bus

When the reset input goes "high" all ports will be set to the input mode (i.e., all 24 lines will be in the high impedance state). After the reset is removed the 8255A can remain in the input mode with no additional initialization required. During the execution of the system program any of the other modes may be selected using a single output instruction. This allows a single 8255A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

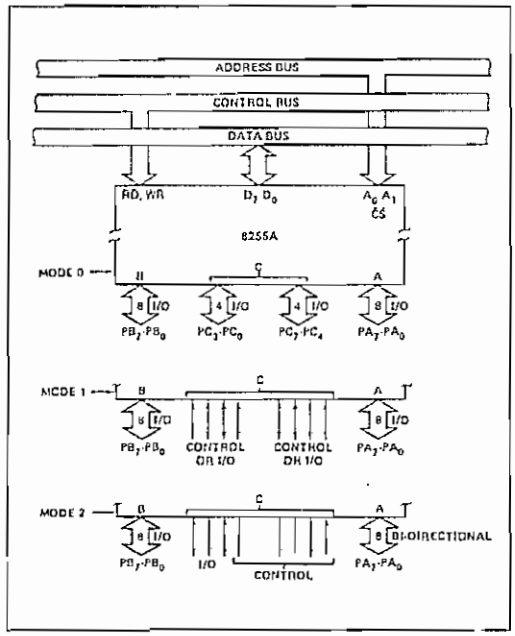


Figure 5. Basic Mode Definitions and Bus Interface

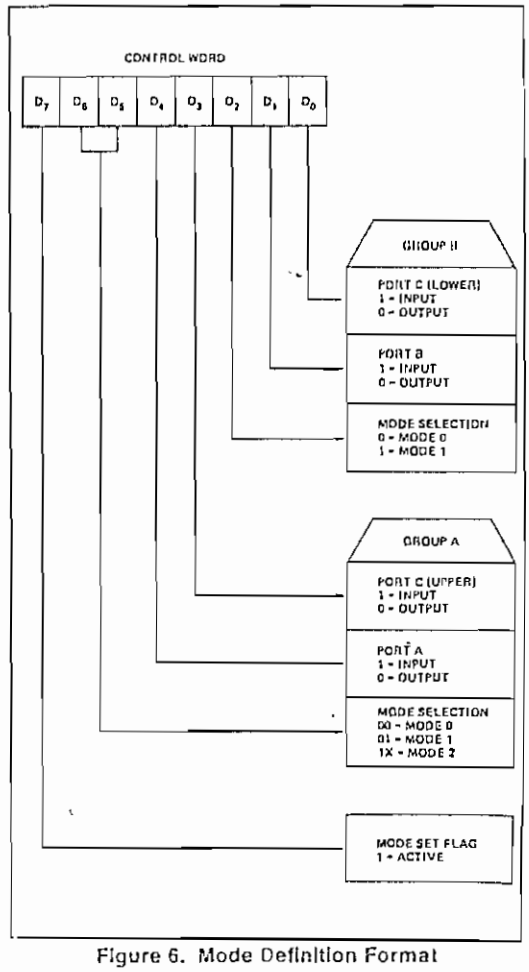


Figure 6. Mode Definition Format

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 8255A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

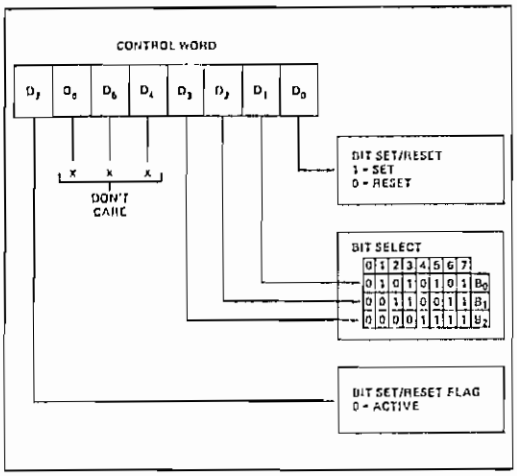


Figure 7. Bit Set/Reset Format

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

Interrupt Control Functions

When the 8255A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET) — INTE is SET — Interrupt enable
(BIT-RESET) — INTE is RESET — Interrupt disable

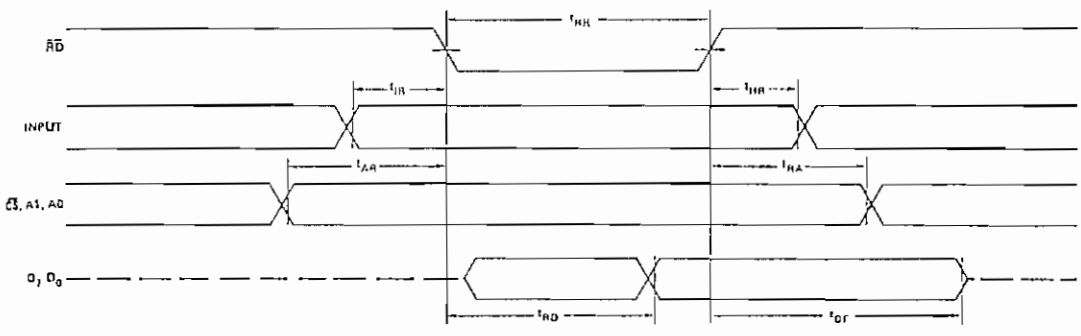
Note: All Mask flip-flops are automatically reset during mode selection and device Reset.

Operating Modes

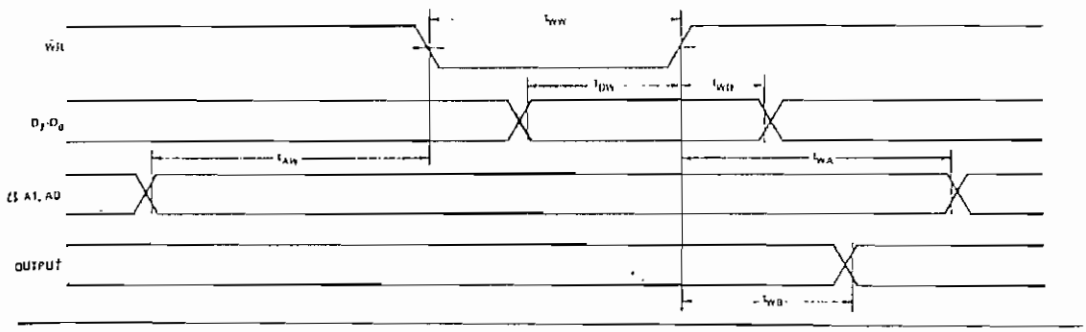
MODE 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.



MODE 0 (Basic Input)



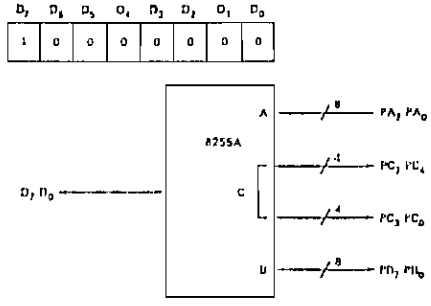
MODE 0 (Basic Output)

MODE 0 Port Definition

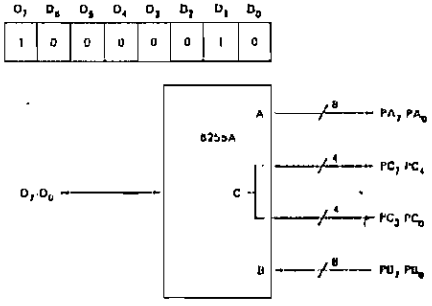
A		B		GROUP A			GROUP B	
D_4	D_3	D_1	D_0	PORT A	PORT C (UPPER)	#	PORT B	PORT C (LOWER)
0	0	0	0	OUTPUT	OUTPUT	0	OUTPUT	OUTPUT
0	0	0	1	OUTPUT	OUTPUT	1	OUTPUT	INPUT
0	0	1	0	OUTPUT	OUTPUT	2	INPUT	OUTPUT
0	0	1	1	OUTPUT	OUTPUT	3	INPUT	INPUT
0	1	0	0	OUTPUT	INPUT	4	OUTPUT	OUTPUT
0	1	0	1	OUTPUT	INPUT	5	OUTPUT	INPUT
0	1	1	0	OUTPUT	INPUT	6	INPUT	OUTPUT
0	1	1	1	OUTPUT	INPUT	7	INPUT	INPUT
1	0	0	0	INPUT	OUTPUT	8	OUTPUT	OUTPUT
1	0	0	1	INPUT	OUTPUT	9	OUTPUT	INPUT
1	0	1	0	INPUT	OUTPUT	10	INPUT	OUTPUT
1	0	1	1	INPUT	OUTPUT	11	INPUT	INPUT
1	1	0	0	INPUT	INPUT	12	OUTPUT	OUTPUT
1	1	0	1	INPUT	INPUT	13	OUTPUT	INPUT
1	1	1	0	INPUT	INPUT	14	INPUT	OUTPUT
1	1	1	1	INPUT	INPUT	15	INPUT	INPUT

MODE 0 Configurations

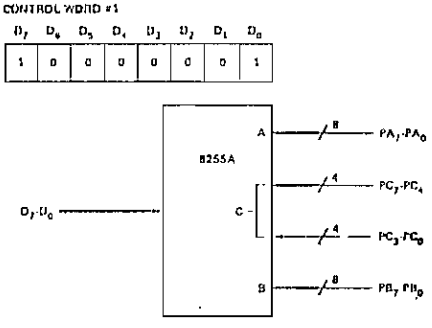
CONTROL WORD #0



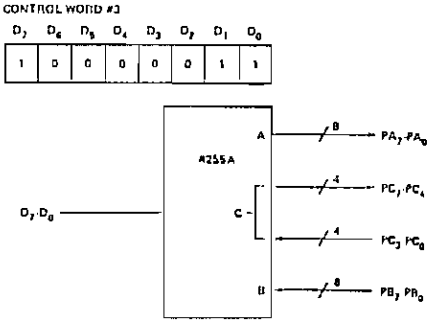
CONTROL WORD #2



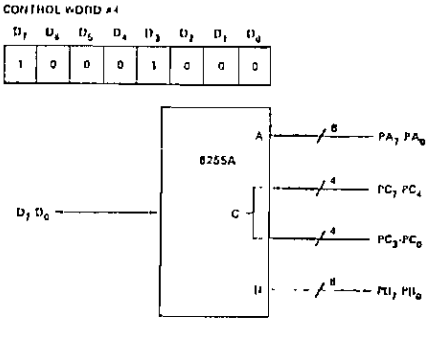
CONTROL WORD #1



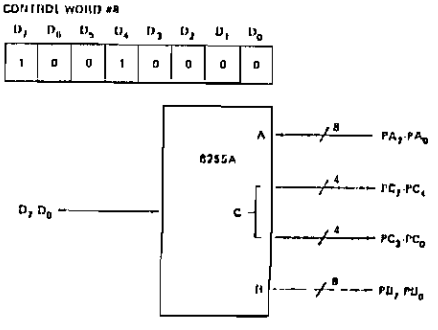
CONTROL WORD #3



CONTROL WORD #4

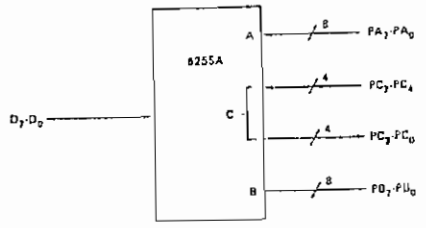


CONTROL WORD #8



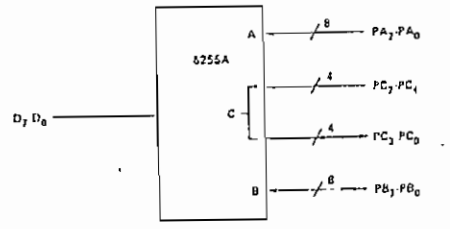
CONTROL WORD #12

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	0	0



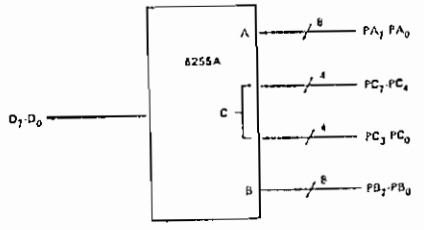
CONTROL WORD #14

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	1	0



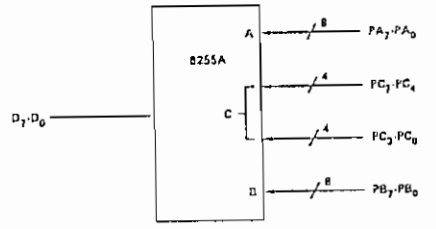
CONTROL WORD #13

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	0	1



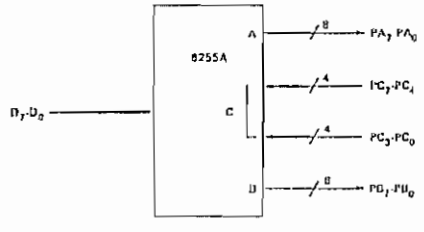
CONTROL WORD #15

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	1	0	1	1



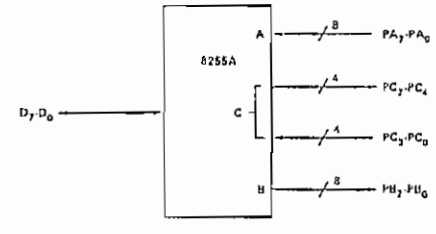
CONTROL WORD #5

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



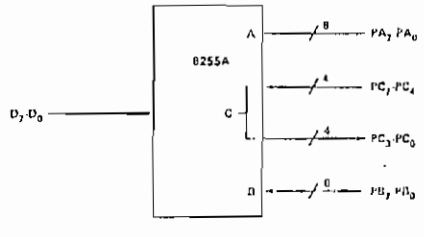
CONTROL WORD #9

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	0	1



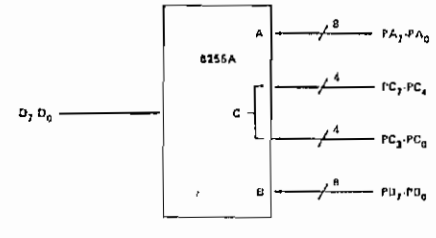
CONTROL WORD #6

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	0



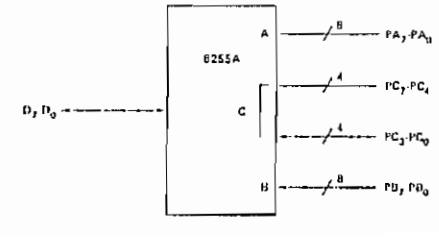
CONTROL WORD #10

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	0



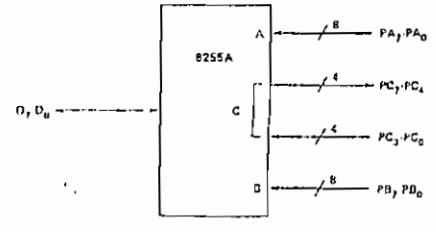
CONTROL WORD #7

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	0	1	0	1	1



CONTROL WORD #11

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	0	0	1	0	0	1	1



Input Control Signal Definition

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by STB input being low and is reset by the rising edge of the RD input.

INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

INTE A

Controlled by bit set/reset of PC_4.

INTE B

Controlled by bit set/reset of PC_2.

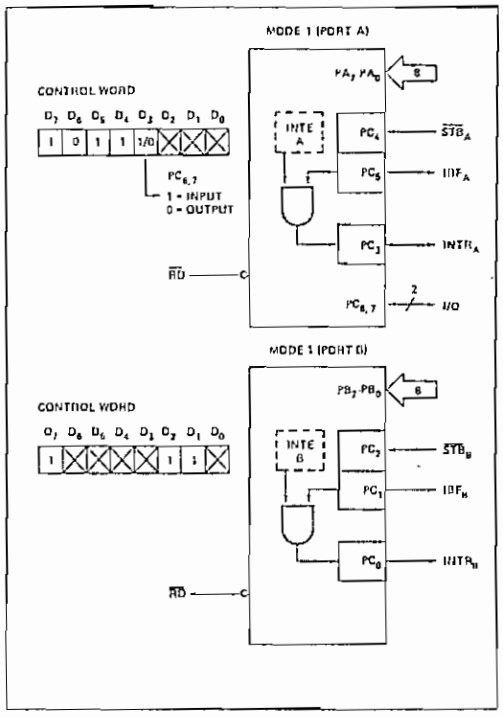


Figure 8. MODE 1 Input

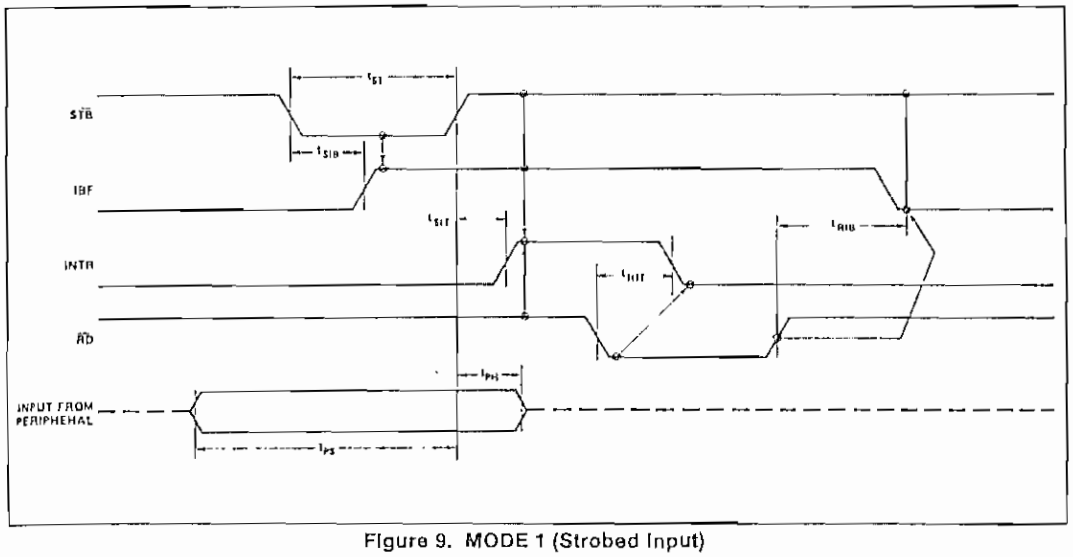


Figure 9. MODE 1 (Strobed Input)

Operating Modes

MODE 1 (Strobed Input/Output). This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, port A and Port B use the lines on port C to generate or accept these "handshaking" signals.

Mode 1 Basic Functional Definitions:

- Two Groups (Group A and Group B)
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output. Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

Output Control Signal Definition

$\overline{OBF}$ (Output Buffer Full F/F). The OBF output will go "low" to indicate that the CPU has written data out to the specified port. The OBF F/F will be set by the rising edge of the WR input and reset by ACK input being low.

$\overline{ACK}$ (Acknowledge Input). A "low" on this input informs the 8255A that the data from port A or port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

INTR (Interrupt Request). A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when ACK is a "one", OBF is a "one" and INTE is a "one". It is reset by the falling edge of WR.

INTE A

Controlled by bit set/reset of PC_6.

INTE B

Controlled by bit set/reset of PC_2.

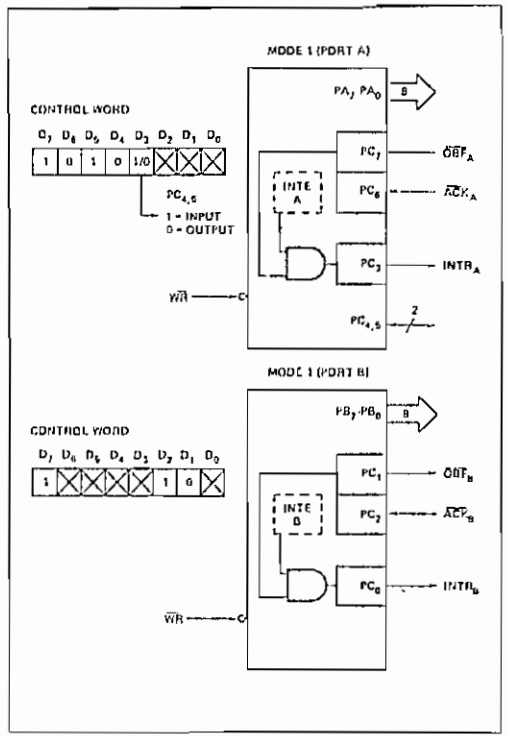


Figure 10. MODE 1 Output

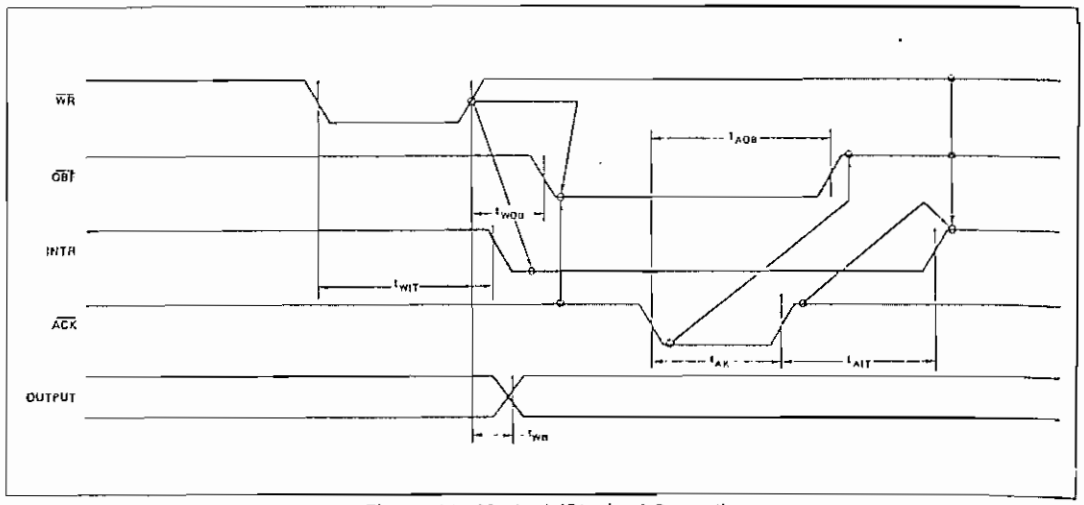


Figure 11. Mode 1 (Strobed Output)

Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

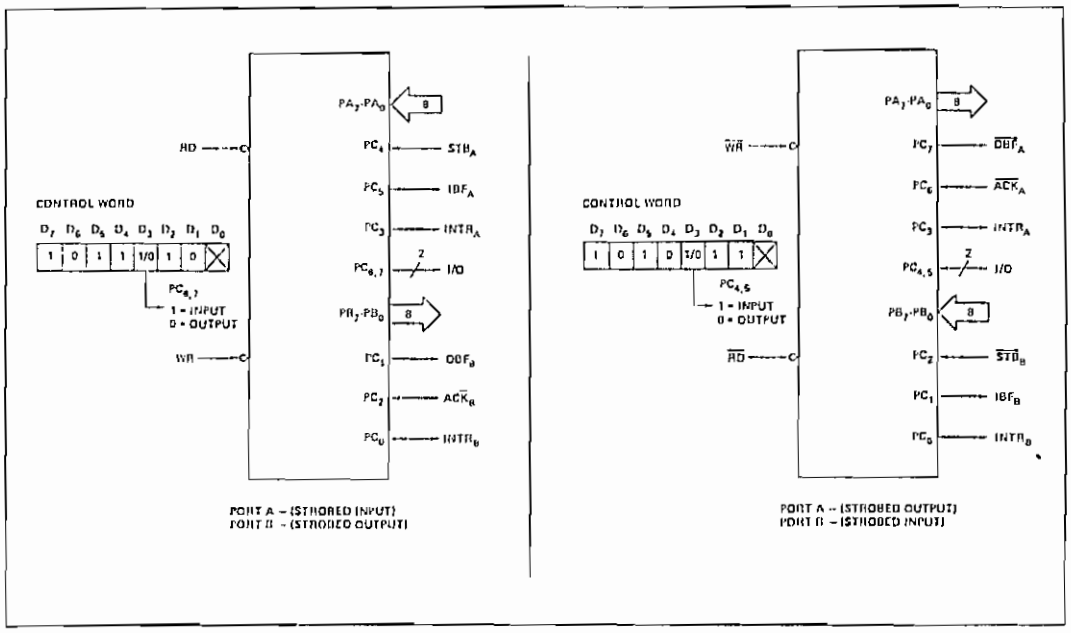


Figure 12. Combinations of MODE 1

Operating Modes

MODE 2 (Strobed Bidirectional Bus I/O). This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A only.
- One 8-bit, bi-directional bus Port (Port A) and a 5-bit control Port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

Bidirectional Bus I/O Control Signal Definition

INTR (Interrupt Request). A high on this output can be used to interrupt the CPU for both input or output operations.

Output Operations

$\overline{OBF}$ (Output Buffer Full). The OBF output will go "low" to indicate that the CPU has written data out to port A.

$\overline{ACK}$ (Acknowledge). A "low" on this input enables the tri-state output buffer of port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

INTE 1 (The INTE Flip-Flop Associated with OBF). Controlled by bit set/reset of PC_6.

Input Operations

$\overline{STB}$ (Strobe Input)

STB (Strobe Input). A "low" on this input loads data into the input latch.

IBF (Input Buffer Full F/F). A "high" on this output indicates that data has been loaded into the input latch.

INTE 2 (The INTE Flip-Flop Associated with IBF). Controlled by bit set/reset of PC_4.

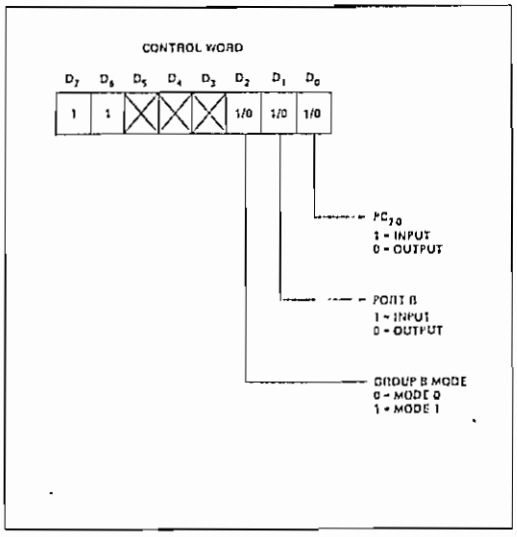


Figure 13. MODE Control Word

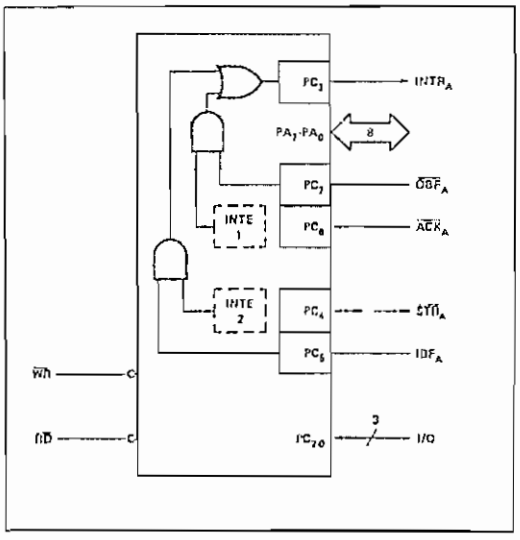


Figure 14. MODE 2

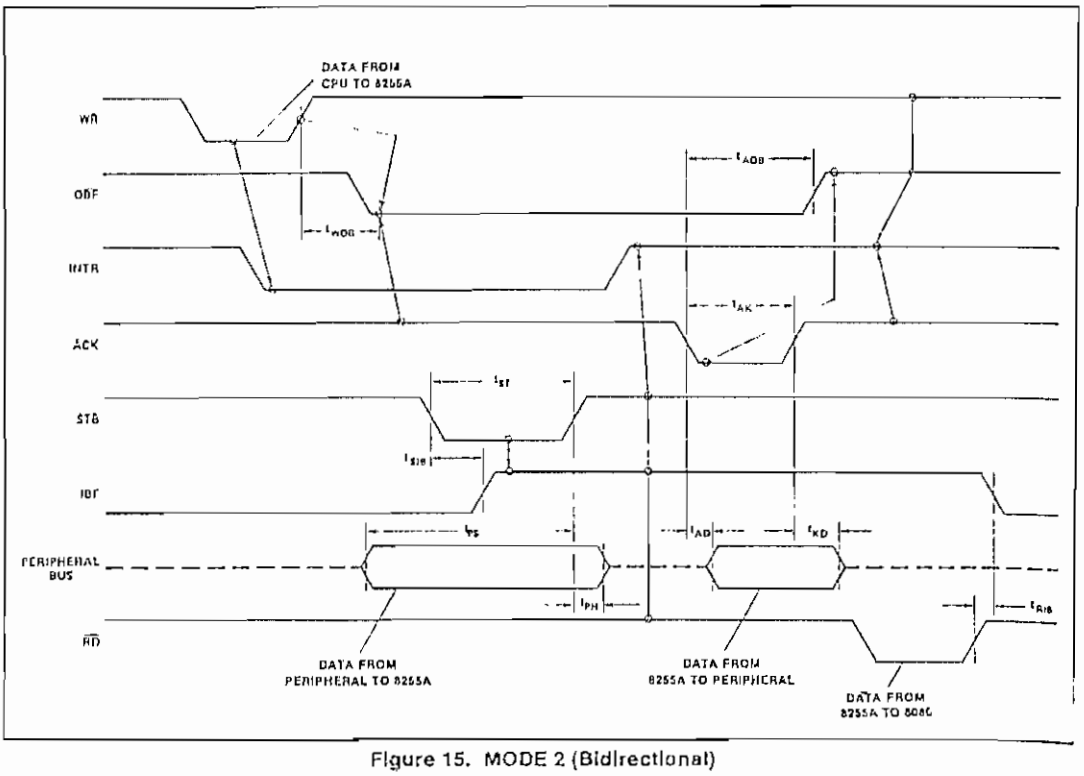


Figure 15. MODE 2 (Bidirectional)

NOTE: Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ and STB occurs before $\overline{RD}$ is permissible.
(INTR = IBF · $\overline{MASK}$ · $\overline{STB}$ · $\overline{RD}$ + $\overline{OBF}$ · $\overline{MASK}$ · $\overline{ACK}$ · $\overline{WR}$)

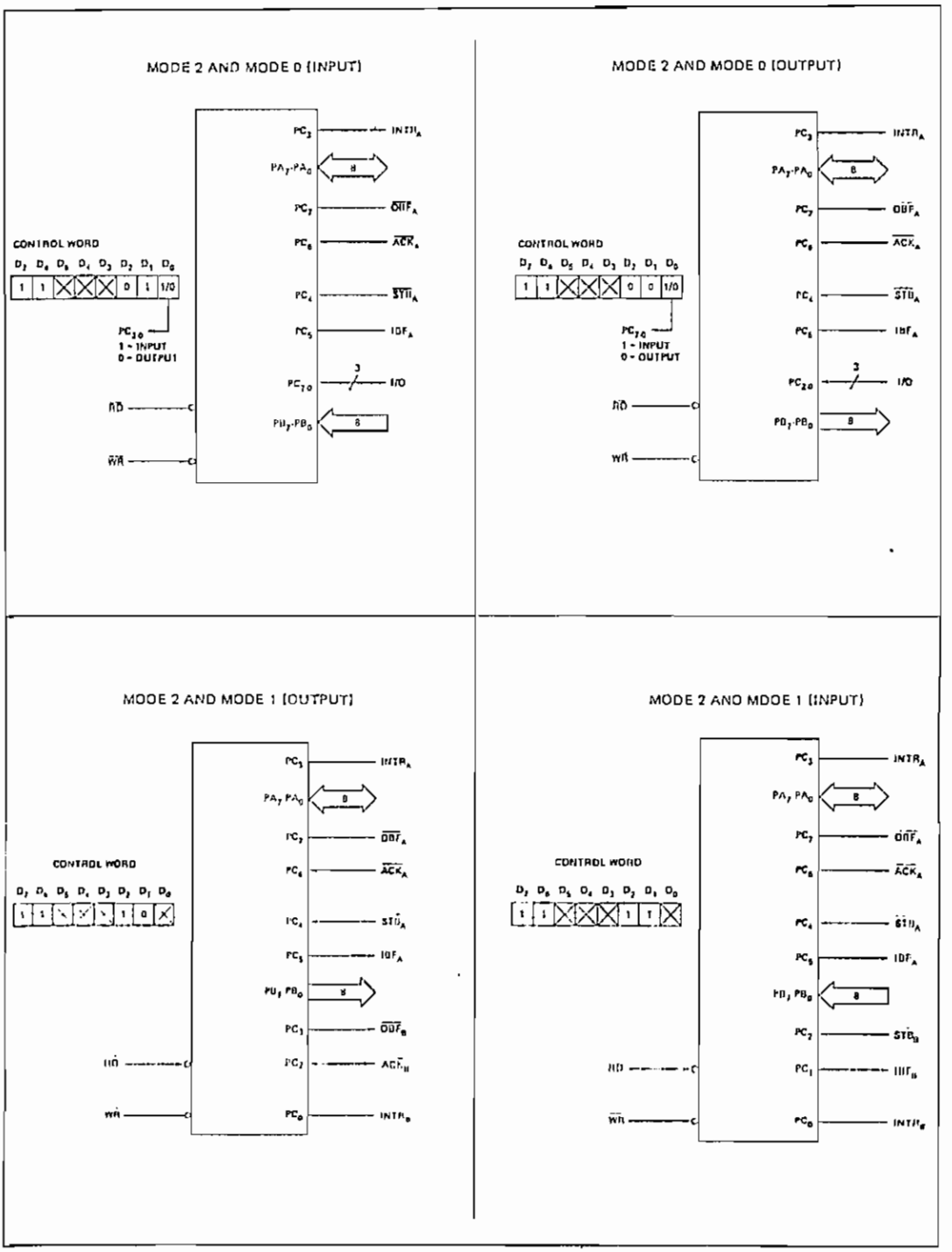


Figure 16. MODE ¼ Combinations

Mode Definition Summary

	MODE 0		MODE 1		MODE 2	
	IN	OUT	IN	OUT	GROUP A ONLY	
PA_0	IN	OUT	IN	OUT	◄—►	
PA_1	IN	OUT	IN	OUT	◄—►	
PA_2	IN	OUT	IN	OUT	◄—►	
PA_3	IN	OUT	IN	OUT	◄—►	
PA_4	IN	OUT	IN	OUT	◄—►	
PA_5	IN	OUT	IN	OUT	◄—►	
PA_6	IN	OUT	IN	OUT	◄—►	
PA_7	IN	OUT	IN	OUT	◄—►	
PB_0	IN	OUT	IN	OUT	—	MODE 0 OR MODE 1 ONLY
PB_1	IN	OUT	IN	OUT	—	
PB_2	IN	OUT	IN	OUT	—	
PB_3	IN	OUT	IN	OUT	—	
PB_4	IN	OUT	IN	OUT	—	
PB_5	IN	OUT	IN	OUT	—	
PB_6	IN	OUT	IN	OUT	—	
PB_7	IN	OUT	IN	OUT	—	
PC_0	IN	OUT	$INTR_B$	$INTR_B$	I/O	
PC_1	IN	OUT	IBF_B	$\overline{OBF}_B$	I/O	
PC_2	IN	OUT	$\overline{STB}_B$	$\overline{ACK}_B$	I/O	
PC_3	IN	OUT	$INTR_A$	$INTR_A$	$INTR_A$	
PC_4	IN	OUT	$\overline{STB}_A$	I/O	$\overline{STB}_A$	
PC_5	IN	OUT	IBF_A	I/O	IBF_A	
PC_6	IN	OUT	I/O	$\overline{ACK}_A$	$\overline{ACK}_A$	
PC_7	IN	OUT	I/O	$\overline{OBF}_A$	$\overline{OBF}_A$	

Special Mode Combination Considerations

There are several combinations of modes when not all of the bits in Port C are used for control or status. The remaining bits can be used as follows:

If Programmed as Inputs –
All input lines can be accessed during a normal Port C read.

If Programmed as Outputs –
Bits in C upper (PC_7-PC_4) must be individually accessed using the bit set/reset function.

Bits in C lower (PC_3-PC_0) can be accessed using the bit set/reset function or accessed as a threesome by writing into Port C.

Source Current Capability on Port B and Port C

Any set of eight output buffers, selected randomly from Ports B and C can source 1mA at 1.5 volts. This feature allows the 8255 to directly drive Darlington type drivers and high-voltage displays that require such source current.

Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 8255 is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

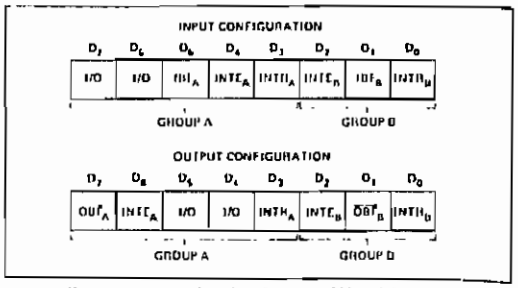


Figure 17. MODE 1 Status Word Format

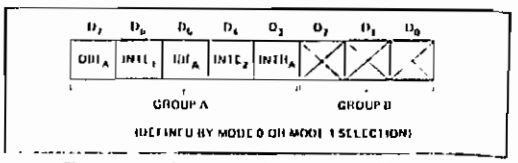
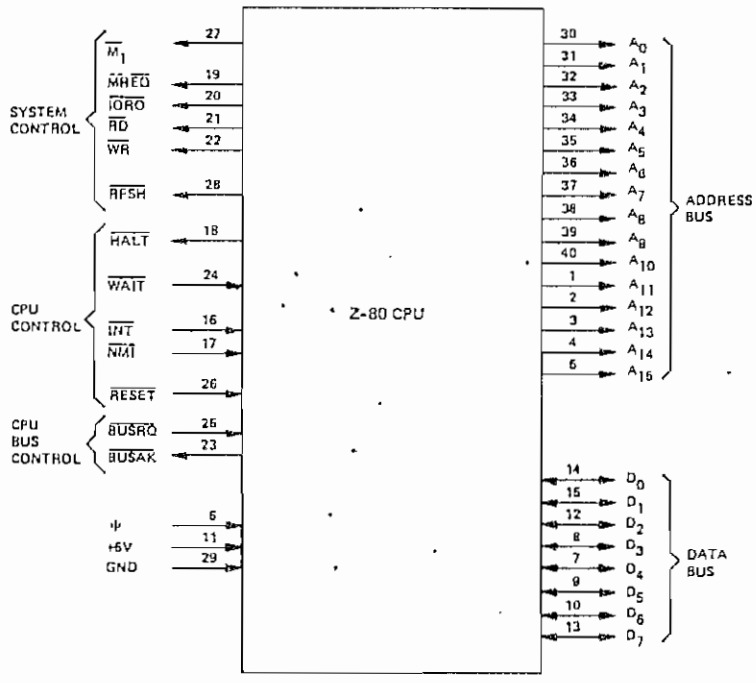


Figure 18. MODE 2 Status Word Format

AFN-00744B

3.0 Z-80 CPU PIN DESCRIPTION

The Z-80 CPU is packaged in an industry standard 40 pin Dual In-Line Package. The I/O pins are shown in figure 3.0-1 and the function of each is described below.



Z-80 PIN CONFIGURATION
FIGURE 3.0-1

A_0-A_{15}
(Address Bus)

Tri-state output, active high. A_0-A_{15} constitute a 16-bit address bus. The address bus provides the address for memory (up to 64K bytes) data exchanges and for I/O device data exchanges. I/O addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.

D_0-D_7
(Data Bus)

Tri-state input/output, active high. D_0-D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$
(Machine Cycle one)

Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution. Note that during execution of 2-byte op-codes, $\overline{M1}$ is generated as each op code byte is fetched. These two byte op-codes always begin with CBH, DDH, EDH or FDH. $\overline{M1}$ also occurs with $\overline{IORQ}$ to indicate an interrupt acknowledge cycle.

$\overline{MREQ}$
(Memory Request)

Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{IORQ}$
(Input/Output Request)

Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{IORQ}$ signal is also generated with an $\overline{M1}$ signal when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while I/O operations never occur during M_1 time.

$\overline{RD}$
(Memory Read)

Tri-state output, active low. $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{WR}$
(Memory Write)

Tri-state output, active low. $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

$\overline{RFSH}$
(Refresh)

Output, active low. $\overline{RFSH}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories.

$\overline{\text{HALT}}$ (Halt state)	Output, active low. $\overline{\text{HALT}}$ indicates that the CPU has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOP's to maintain memory refresh activity.
$\overline{\text{WAIT}}$ (Wait)	Input, active low. $\overline{\text{WAIT}}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active. This signal allows memory or I/O devices of any speed to be synchronized to the CPU.
$\overline{\text{INT}}$ (Interrupt Request)	Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled and if the $\overline{\text{BUSRQ}}$ signal is not active. When the CPU accepts the interrupt, an acknowledge signal ($\overline{\text{IORQ}}$ during M_1 time) is sent out at the beginning of the next instruction cycle. The CPU can respond to an interrupt in three different modes that are described in detail in section 5.4 (CPU Control Instructions).
$\overline{\text{NMI}}$ (Non Maskable Interrupt)	Input, negative edge triggered. The non maskable interrupt request line has a higher priority than $\overline{\text{INT}}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{\text{NMI}}$ automatically forces the Z-80 CPU to restart to location 0066_H. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted. Note that continuous WAIT cycles can prevent the current instruction from ending, and that a $\overline{\text{BUSRQ}}$ will override a $\overline{\text{NMI}}$.
$\overline{\text{RESET}}$	Input, active low. $\overline{\text{RESET}}$ forces the program counter to zero and initializes the CPU. The CPU initialization includes: 1) Disable the interrupt enable flip-flop 2) Set Register I = 00_H 3) Set Register R = 00_H 4) Set Interrupt Mode 0 During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state.
$\overline{\text{BUSRQ}}$ (Bus Request)	Input, active low. The bus request signal is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{\text{BUSRQ}}$ is activated, the CPU will set these buses to a high impedance state as soon as the current CPU machine cycle is terminated.
$\overline{\text{BUSAK}}$ (Bus Acknowledge)	Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.
Φ	Single phase TTL level clock which requires only a 330 ohm pull-up resistor to +5 volts to meet all clock requirements.

4.0 CPU TIMING

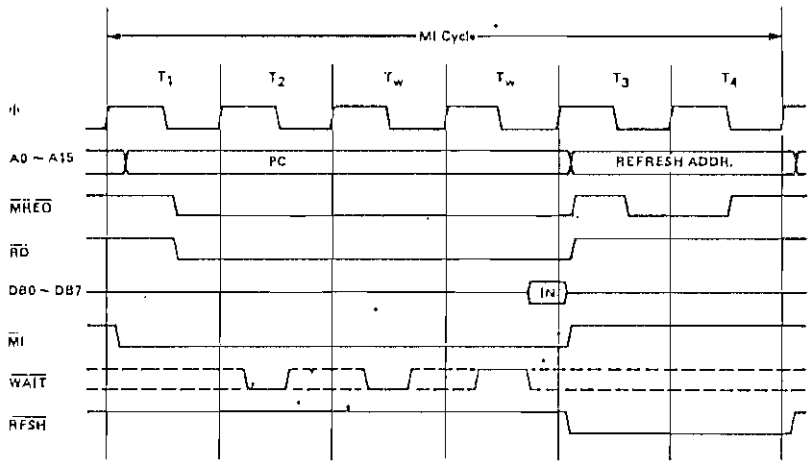
The Z-80 CPU executes instructions by stepping through a very precise set of a few basic operations. These include:

Memory read or write
I/O device read or write
Interrupt acknowledge

All instructions are merely a series of these basic operations. Each of these basic operations can take from three to six clock periods to complete or they can be lengthened to synchronize the CPU to the speed of external devices. The basic clock periods are referred to as T cycles and the basic operations are referred to as M (for machine) cycles. Figure 4.0-0 illustrates how a typical instruction will be merely a series of specific M and T cycles. Notice that this instruction consists of three machine cycles (M1, M2 and M3). The first machine cycle of any instruction is a fetch cycle which is four, five or six T cycles long (unless lengthened by the wait signal which will be fully described in the next section). The fetch cycle (M1) is used to fetch the OP code of the next instruction to be executed. Subsequent machine cycles move data between the CPU and memory or I/O devices and they may have anywhere from three to five T cycles (again they may be lengthened by wait states to synchronize the external devices to the CPU). The following paragraphs describe the timing which occurs within any of the basic machine cycles. In section 7, the exact timing for each instruction is specified.

All CPU timing can be broken down into a few very simple timing diagrams as shown in figure 4.0-1 through 4.0-7. These diagrams show the following basic operations with and without wait states (wait states are added to synchronize the CPU to slow memory or I/O devices).

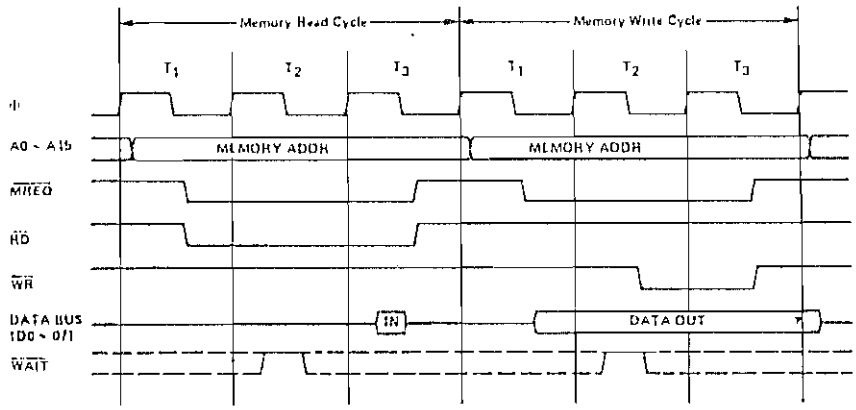
4.0-1. Instruction OP code fetch (M1 cycle)
4.0-2. Memory data read or write cycles
4.0-3. I/O read or write cycles
4.0-4. Bus Request/Acknowledge Cycle
4.0-5. Interrupt Request/Acknowledge Cycle
4.0-6. Non maskable Interrupt Request/Acknowledge Cycle
4.0-7. Exit from a HALT instruction



INSTRUCTION OP CODE FETCH WITH WAIT STATES
FIGURE 4.0-1A

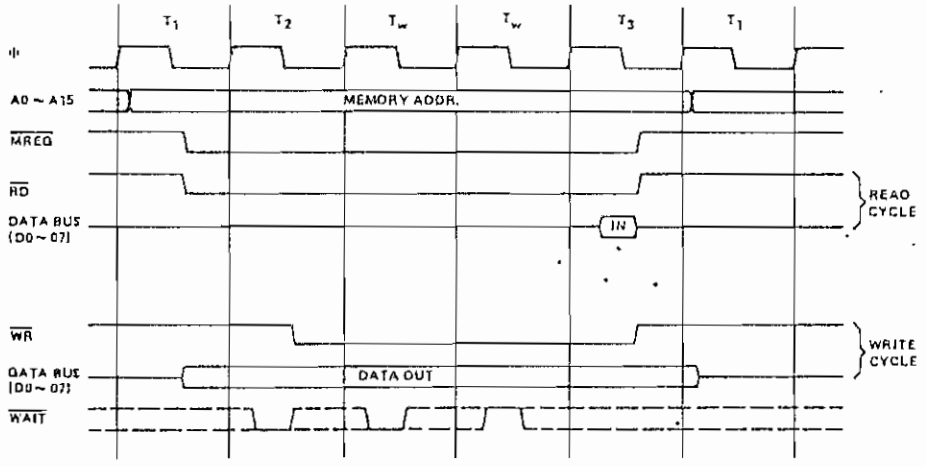
MEMORY READ OR WRITE

Figure 4.0-2 illustrates the timing of memory read or write cycles other than an OP code fetch (M1 cycle). These cycles are generally three clock periods long unless wait states are requested by the memory via the WAIT signal. The MREQ signal and the RD signal are used the same as in the fetch cycle. In the case of a memory write cycle, the MREQ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The WR line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory. Furthermore the WR signal goes inactive one half T state before the address and data bus contents are changed so that the overlap requirements for virtually any type of semiconductor memory type will be met.



MEMORY READ OR WRITE CYCLES
FIGURE 4.0-2

Figure 4.0-2A illustrates how a $\overline{WAIT}$ request signal will lengthen any memory read or write operation. This operation is identical to that previously described for a fetch cycle. Notice in this figure that a separate read and a separate write cycle are shown in the same figure although read and write cycles can never occur simultaneously.



MEMORY READ OR WRITE CYCLES WITH WAIT STATES
FIGURE 4.0-2A

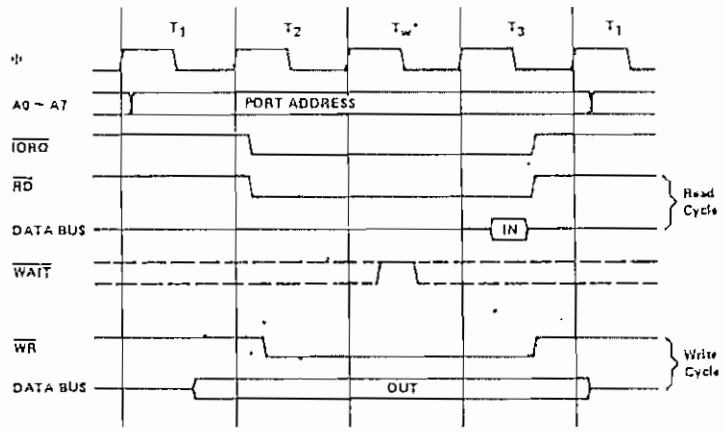
INPUT OR OUTPUT CYCLES

Figure 4.0-3 illustrates an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted. The reason for this is that during I/O operations, the time from when the $\overline{IORQ}$ signal goes active until the CPU must sample the $\overline{WAIT}$ line is very short and without this extra state sufficient time does not exist for an I/O port to decode its address and activate the $\overline{WAIT}$ line if a wait is required. Also, without this wait state it is difficult to design MOS I/O devices that can operate at full CPU speed. During this wait state time the $\overline{WAIT}$ request signal is sampled. During a read I/O operation, the $\overline{RD}$ line is used to enable the addressed port onto the data bus just as in the case of a memory read. For I/O write operations, the $\overline{WR}$ line is used as a clock to the I/O port, again with sufficient overlap timing automatically provided so that the rising edge may be used as a data clock.

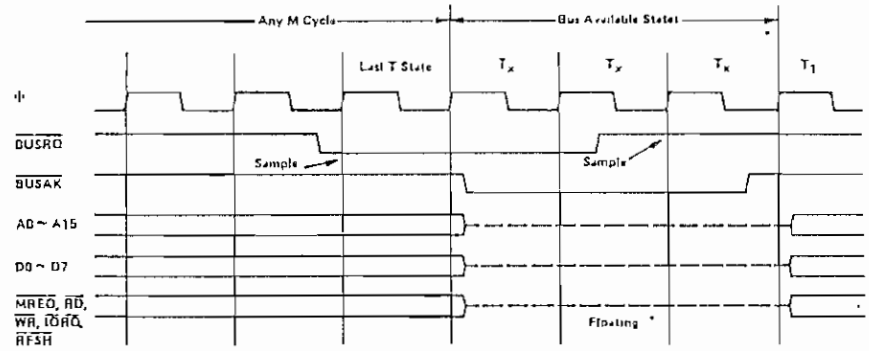
Figure 4.0-3A illustrates how additional wait states may be added with the $\overline{WAIT}$ line. The operation is identical to that previously described.

BUS REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-4 illustrates the timing for a Bus Request/Acknowledge cycle. The $\overline{BUSRQ}$ signal is sampled by the CPU with the rising edge of the last clock period of any machine cycle. If the $\overline{BUSRQ}$ signal is active, the CPU will set its address, data and tri-state control signals to the high impedance state with the rising edge of the next clock pulse. At that time any external device can control the buses to transfer data between memory and I/O devices. (This is generally known as Direct Memory Access [DMA] using cycle stealing). The maximum time for the CPU to respond to a bus request is the length of a machine cycle and the external controller can maintain control of the bus for as many clock cycles as is desired. Note, however, that if very long DMA cycles are used, and dynamic memories are being used, the external controller must also perform the refresh function. This situation only occurs if very large blocks of data are transferred under DMA control. Also note that during a bus request cycle, the CPU cannot be interrupted by either a $\overline{NMI}$ or an $\overline{INT}$ signal.



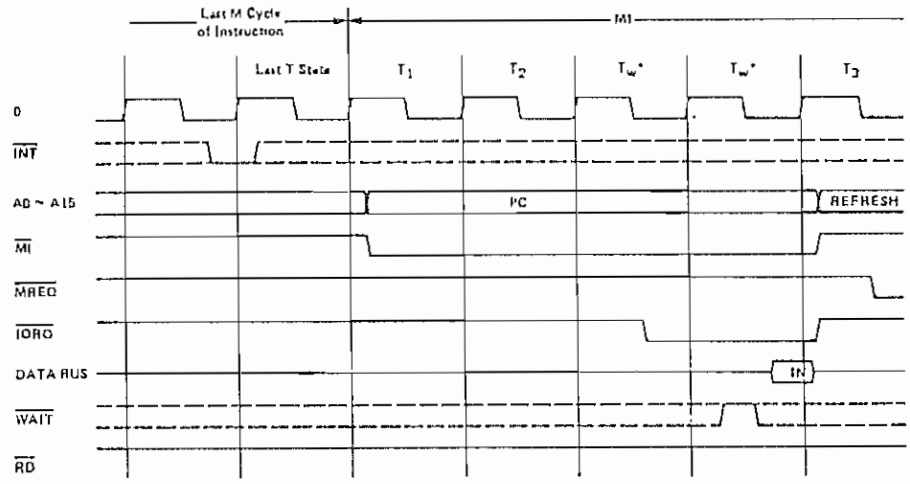
INPUT OR OUTPUT CYCLES
FIGURE 4.0-3



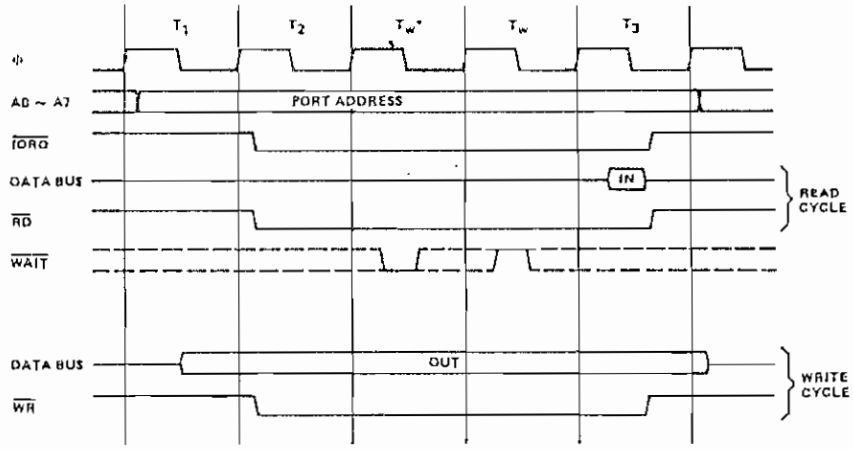
BUS REQUEST/ACKNOWLEDGE CYCLE
FIGURE 4.0-4

INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

Figure 4.0-5 illustrates the timing associated with an interrupt cycle. The interrupt signal ($\overline{INT}$) is sampled by the CPU with the rising edge of the last clock at the end of any instruction. The signal will not be accepted if the internal CPU software controlled interrupt enable flip-flop is not set or if the $\overline{BUSRQ}$ signal is active. When the signal is accepted a special M1 cycle is generated. During this special M1 cycle the $\overline{IORQ}$ signal becomes active (instead of the normal $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Notice that two wait states are automatically added to this cycle. These states are added so that a ripple priority interrupt scheme can be easily implemented. The two wait states allow sufficient time for the ripple signals to stabilize and identify which I/O device must insert the response vector. Refer to section 8.0 for details on how the interrupt response vector is utilized by the CPU.



INTERRUPT REQUEST/ACKNOWLEDGE CYCLE
FIGURE 4.0-5



INPUT OR OUTPUT CYCLES WITH WAIT STATES
FIGURE 4.0-3A

National Semiconductor

MM54HC74/MM74HC74
Dual D Flip-Flop with Preset and Clear

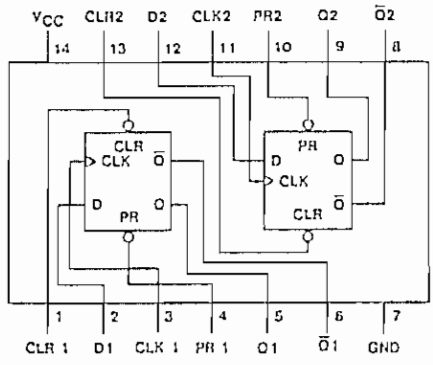
General Description

The MM54HC74/MM74HC74 utilize silicon gate CMOS technology to achieve operating speeds similar to the equivalent LS-TTL part. It possesses the high noise immunity and low power consumption of standard CMOS integrated circuits, along with the ability to drive 10 LS-TTL loads (8 LS-TTL loads for 54HC).

This flip flop has independent DATA, PRESET, CLEAR, and CLOCK inputs and Q and $\overline{Q}$ outputs. The logic level present at the data input is transferred to the output during the positive going transition of the clock pulse. Preset and clear are independent of the clock and accomplished by a low level at the appropriate input.

The 54HC/74HC logic family is functionally as well as pin out compatible with the standard 54LS/74LS logic family. All inputs are protected from damage due to static discharge by internal diode clamps to V_{CC} and Ground.

Connection Diagram



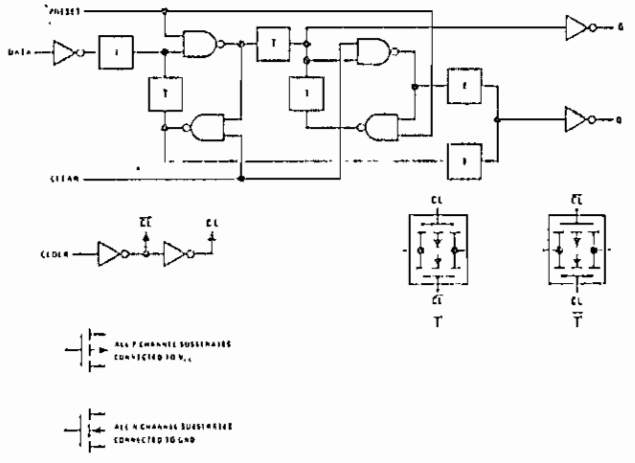
Truth Table

Inputs				Outputs	
PR	CLR	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q0	$\overline{Q}$0

Notes: Q0 = the level of Q before the indicated input conditions were established

*This configuration is nonstable, that is it will not persist when preset and clear inputs return to their inactive (high) level

Logic Diagram



National Semiconductor

MM54HC138/MM74HC138 3-To-8 Line Decoder
MM54HC139/MM74HC139 Dual 2-To-4 Line Decoder

General Description

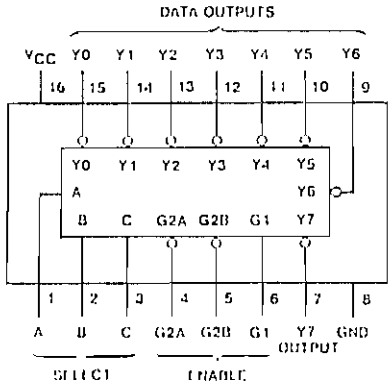
These devices are high speed silicon gate CMOS decoders, and are well suited to memory address decoding or data routing applications. Both circuits feature high noise immunity and low power consumption usually associated with CMOS circuitry, yet have speeds comparable to low power Schottky TTL logic.

The MM54HC138/MM74HC138 have 3 binary select inputs (A, B, and C). If the device is enabled these inputs determine which one of the eight normally high outputs will go low. Two active low and one active high enables (G1, G2A and G2B) are provided to ease the cascading of decoders.

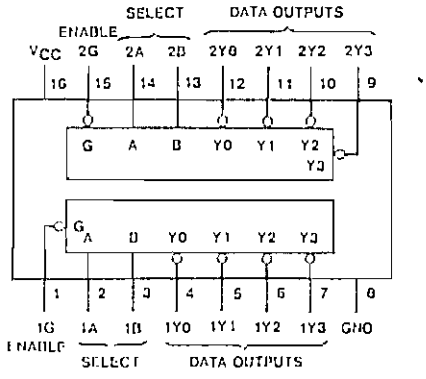
The MM54HC139/MM74HC139 contain two independent one-of-four decoders each with a single active low enable input (G1, or G2). Data on the select inputs (A1, and B1 or A2, and B2) cause one of the four normally high outputs to go low.

The decoder's outputs can drive 10 low power Schottky TTL equivalent loads (8 loads for 54HC), and are functionally as well as pin equivalent to the 54LS138/76LS138, and the 54LS139/74LS139 respectively. All inputs are protected from damage due to static discharge by diodes to V_{CC} and ground.

Connection Diagrams



MM54HC138/MM74HC138



MM54HC139/MM54HC139

Truth Table

'HC138

Inputs					Outputs							
Enable		Select										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

*G2 = G2A + G2B

H = High level, L = low level, X = don't care

'HC139

Inputs			Outputs			
Enable	Select					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = high level, L = low level, X = don't care

National Semiconductor

MM54HC245/MM74HC245 Octal TRI-STATE® Transceiver
MM54HC640/MM74HC640 Inverting Octal TRI-STATE® Transceiver
MM54HC643/MM74HC643 True-Inverting Octal TRI-STATE® Transceiver

General Description

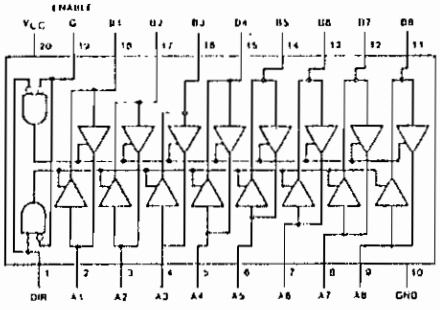
These silicon gate CMOS TRI-STATE bi-directional buffers are intended for two-way asynchronous communication between data buses. They have high drive current outputs which enable high speed operation even when driving large bus capacitances. These circuits possess the low power consumption and high noise immunity usually associated with CMOS circuitry, yet have speeds comparable to low power Schottky TTL circuits.

The MM54HC245/MM74HC245 has one active low enable input (G), and a direction control (DIR). When the DIR input is high, data flows from the A inputs to the B outputs. When DIR is low, date flows from B to A.

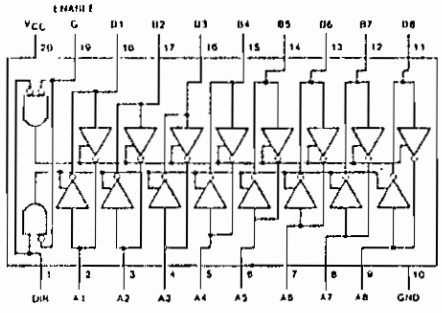
Both the MM54HC640/MM74HC640 and the MM54HC643/MM74HC643 have one active low enable input (G), and a direction control (DIR). When the DIR input is high, data flows from the A inputs to the B outputs. When DIR is low, data flows from B to A. The MM54HC640/MM74HC640 transfers inverted data from one bus to the other. The MM54HC643/MM74HC643 transfers inverted data from the A bus to the B bus, and non-inverted data from the B bus to the A bus.

All three devices can drive up to 15 LS-TTL Loads (12 loads for 54HC), and all inputs are protected from damage due to static discharge by diodes to V_{CC} and ground.

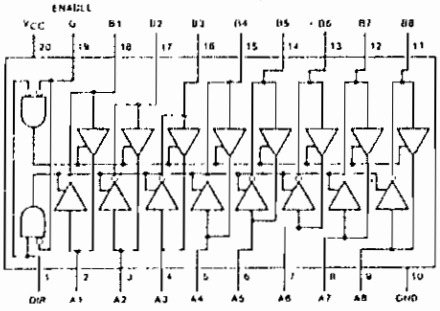
Connection Diagrams



MM54HC245/MM74HC245



MM54HC640/MM74HC640



MM54HC643/MM74HC643

Truth Table

Control Inputs		Operation		
G	DIR	245	640	643
L	L	B data to A bus	$\overline{B}$ data to A bus	B data to A bus
L	H	A data to B bus	$\overline{A}$ data to B bus	$\overline{A}$ data to B bus
H	X	Isolation	Isolation	Isolation

H = high level, L = low level, X = irrelevant

MOTOROLA

MCM65116
HM6116

Advance Information

HCMOS
(COMPLEMENTARY MOS)

2,048 × 8 BIT
STATIC RANDOM
ACCESS MEMORY

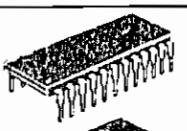
16K BIT STATIC RANOOM ACCESS MEMORY

The MCM65116 is a 16,384-bit Static Random Access Memory organized as 2048 words by 8-bits, fabricated using Motorola's High-performance silicon-gate CMOS (HCMOS) technology. It uses a design approach which provides the simple timing features associated with fully static memories and the reduced power associated with CMOS memories. This means low standby power without the need for clocks, nor reduced data rates due to cycle times that exceed access time.

Chip Enable ($\overline{E}$) controls the power-down feature. It is not a clock but rather a chip control that affects power consumption. In less than a cycle time after chip enable ($\overline{E}$) goes high, the part automatically reduces its power requirements and remains in this low-power standby as long as the chip enable ($\overline{E}$) remains high. The automatic power-down feature causes no performance degradation.

The MCM65116 is in a 24-pin dual-in-line package with the industry standard JEDEC approved pinout and is pinout compatible with the industry standard 16K EPROM/ROM

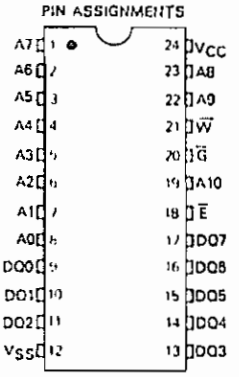
- Single +5 V Supply
- 2048 Words by 8-Bit Organization
- HCMOS Technology
- Fully Static: No Clock or Timing Strobe Required
- Maximum Access Time: MCM65116-12 — 120 ns
  MCM65116-15 — 150 ns
  MCM65116-20 — 200 ns
- Power Dissipation: 55 mA Maximum (Active)
  10 mA Maximum (Standby-TTL Levels)
  2 mA Maximum (Standby)
  100 μA Maximum (Standby-MCM65L116)
- Low Voltage Data Retention (MCM65L116 only) 100 μW Maximum



P SUFFIX
PLASTIC PACKAGE
CASE 709

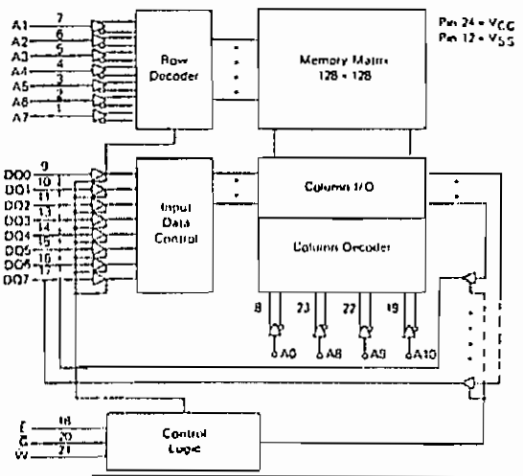
C SUFFIX
FRIT-SEAL CERAMIC PACKAGE
CASE 623

PIN ASSIGNMENTS



PIN NAMES	
A0-A10	Address Input
DO0-DO7	Data Input/Output
$\overline{W}$	Write Enable
$\overline{G}$	Output Enable
$\overline{E}$	Chip Enable
V_{CC}	Power (+5 V)
V_{SS}	Ground

BLOCK DIAGRAM



This document contains information on a new product. Specifications and information herein are subject to change without notice.

AOI-911/5-82

ABSOLUTE MAXIMUM RATINGS (See Note)

Rating	Value	Unit
Temperature Under Bias	−10 to +80	°C
Voltage on Any Pin With Respect to V_{SS}	−1.0 to +7.0	V
DC Output Current	20	mA
Power Dissipation	1.2	Watt
Operating Temperature Range	0 to +70	°C
Storage Temperature Range	−65 to +150	°C

NOTE: Permanent device damage may occur if ABSOLUTE MAXIMUM RATINGS are exceeded. Functional operation should be restricted to RECOMMENDED OPERATING CONDITIONS. Exposure to higher than recommended voltages for extended periods of time could affect device reliability.

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit.

DC OPERATING CONDITIONS AND CHARACTERISTICS

(Full operating voltage and temperature ranges unless otherwise noted.)

RECOMMENDED OPERATING CONDITIONS

Parameter	Symbol	Min	Typ	Max	Unit
Supply Voltage	V_{CC}	4.5	5.0	5.5	V
	V_{SS}	0	0	0	V
Input Voltage	V_{IH}	2.2	3.5	6.0	V
	V_{IL}	−1.0*	—	0.8	V

*The device will withstand undershoots to the −1.0 volt level with a maximum pulse width of 50 ns at the −0.3 volt level. This is periodically sampled rather than 100% tested.

RECOMMENDED OPERATING CHARACTERISTICS

Parameter	Symbol	MCM65116			MCM65L116			Unit
		Min	Typ*	Max	Min	Typ*	Max	
Input Leakage Current (V_{CC} = 5.5 V, V_{in} = GND to V_{CC})	$\|I_{LI}\|$	—	—	1	—	—	1	μA
Output Leakage Current ($\overline{E}$ = V_{IH} or $\overline{G}$ = V_{IH} $V_{I/O}$ = GND to V_{CC})	$\|I_{LO}\|$	—	—	1	—	—	1	μA
Operating Power Supply Current ($\overline{E}$ = V_{IL}, $I_{I/O}$ = 0 mA)	I_{CC}	—	35	55	—	35	55	mA
Average Operating Current Minimum cycle, duty = 100%	I_{CC2}	—	35	55	—	35	55	mA
Standby Power ($\overline{E}$ = V_{IH})	I_{SB}	—	5	10	—	5	10	mA
Supply Current ($\overline{E}$ ≥ V_{CC} − 0.2 V, V_{in} ≥ V_{CC} − 0.2 V or V_{in} ≤ 0.2 V)	I_{SB1}	—	20	2000	—	4	100	μA
Output Low Voltage (I_{OL} = 2.1 mA)	V_{OL}	—	—	0.4	—	—	0.4	V
Output High Voltage (I_{OH} = −1.0 mA)**	V_{OH}	2.4	—	—	2.4	—	—	V

*V_{CC} = 5 V, T_A = 25°C

**Also, output voltages are compatible with Motorola's new high-speed CMOS logic family if the same power supply voltage is used.

CAPACITANCE (f = 1.0 MHz, T_A = 25°C, periodically sampled rather than 100% tested.)

Characteristic	Symbol	Typ	Max	Unit
Input Capacitance except $\overline{E}$	C_{in}	3	5	pF
Input/Output Capacitance and $\overline{E}$ Input Capacitance	$C_{I/O}$	5	7	pF

MODE SELECTION

Mode	$\overline{E}$	$\overline{G}$	$\overline{W}$	V_{CC} Current	DQ
Standby	H	X	X	I_{SB}, I_{SB1}	High Z
Read	L	L	H	I_{CC}	Q
Write Cycle (1)	L	H	L	I_{CC}	D
Write Cycle (2)	L	L	L	I_{CC}	D

AC OPERATING CONDITIONS AND CHARACTERISTICS

(Full operating voltage and temperature unless otherwise noted.)

Input Pulse Levels . . . . . 0.8 Volt to 2.4 Volts
Input Rise and Fall Times . . . . . 10 ns
Input and Output Timing Levels . . . . . 1.5 Volts
Output Load . . . . . 1 TTL Gate and C_L = 100 pF

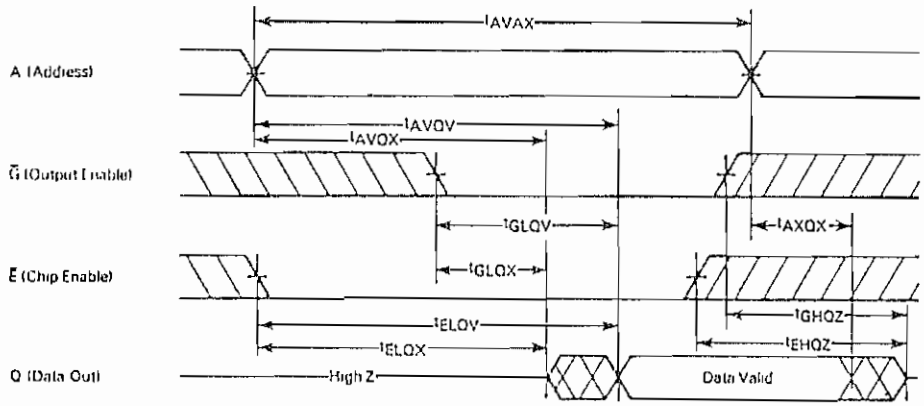
READ CYCLE

Parameter	Symbol	MCM65116-12 MCM65L116-12		MCM65116-15 MCM65L116-15		MCM65116-20 MCM65L116-20		Unit
		Min	Max	Min	Max	Min	Max	
Address Valid to Address Don't Care (Cycle Time when Chip Enable is Held Active)	t_{AVAX}	120	—	150	—	200	—	ns
Chip Enable Low to Chip Enable High	t_{ELEH}	120	—	150	—	200	—	ns
Address Valid to Output Valid (Access)	t_{AVQV}	—	120	—	150	—	200	ns
Chip Enable Low to Output Valid (Access)	t_{ELQV}	—	120	—	150	—	200	ns
Address Valid to Output Invalid	t_{AVQX}	10	—	15	—	15	—	ns
Chip Enable Low to Output Invalid	t_{ELQX}	10	—	15	—	15	—	ns
Chip Enable High to Output High Z	t_{EHQZ}	0	40	0	50	0	60	ns
Output Enable to Output Valid	t_{GLQV}	—	80	—	100	—	120	ns
Output Enable to Output Invalid	t_{GLQX}	10	—	15	—	15	—	ns
Output Enable to Output High Z	t_{GLQZ}	0	40	0	50	0	60	ns
Address Invalid to Output Invalid	t_{AXQX}	10	—	15	—	15	—	ns
Address Valid to Chip Enable Low (Address Setup)	t_{AVEL}	0	—	0	—	0	—	ns
Chip Enable to Power Up Time	t_{PU}	0		0		0		ns
Chip Disable to Power Down Time	t_{PD}	—	30	—	30	—	30	ns

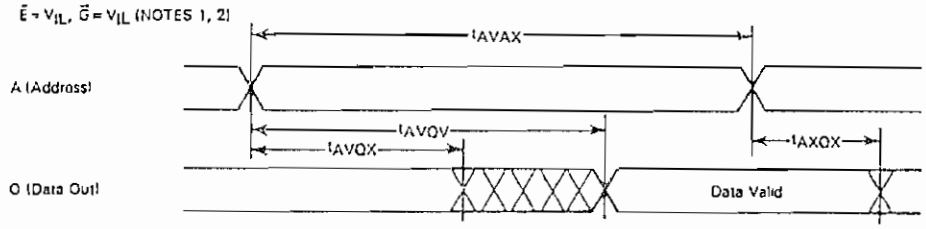
WRITE CYCLE

Parameter	Symbol	MCM65116-12 MCM65L116-12		MCM65116-15 MCM65L116-15		MCM65116-20 MCM65L116-20		Unit
		Min	Max	Min	Max	Min	Max	
Chip Enable Low to Write High	t_{ELWH}	70	–	90	–	120	–	ns
Address Valid to Write High	t_{AVWH}	105		120		140		ns
Address Valid to Write Low (Address Setup)	t_{AVWL}	20		20	–	20	–	ns
Write Low to Write High (Write Pulse Width)	t_{WLWH}	70	–	90	–	120	–	ns
Write High to Address Don't Care	t_{WHAX}	5	–	10	–	10	–	ns
Data Valid to Write High	t_{DVWH}	35	–	40	–	60	–	ns
Write High to Data Don't Care (Data Hold)	t_{WHDX}	5	–	10	–	10	–	ns
Write Low to Output High Z	t_{WLQZ}	0	50	0	60	0	60	ns
Write High to Output Valid	t_{WHQV}	5	–	10	–	10	–	ns
Output Disable to Output High Z	t_{GHQZ}	0	40	0	50	0	60	ns

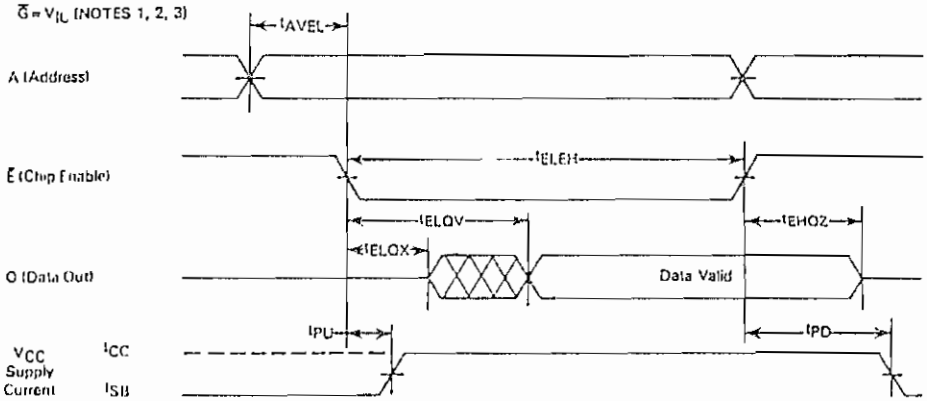
READ CYCLE TIMING 1 (NOTES 1 AND 2)



READ CYCLE TIMING 2
$\overline{E} = V_{IL}$, $\overline{G} = V_{IL}$ (NOTES 1, 2)

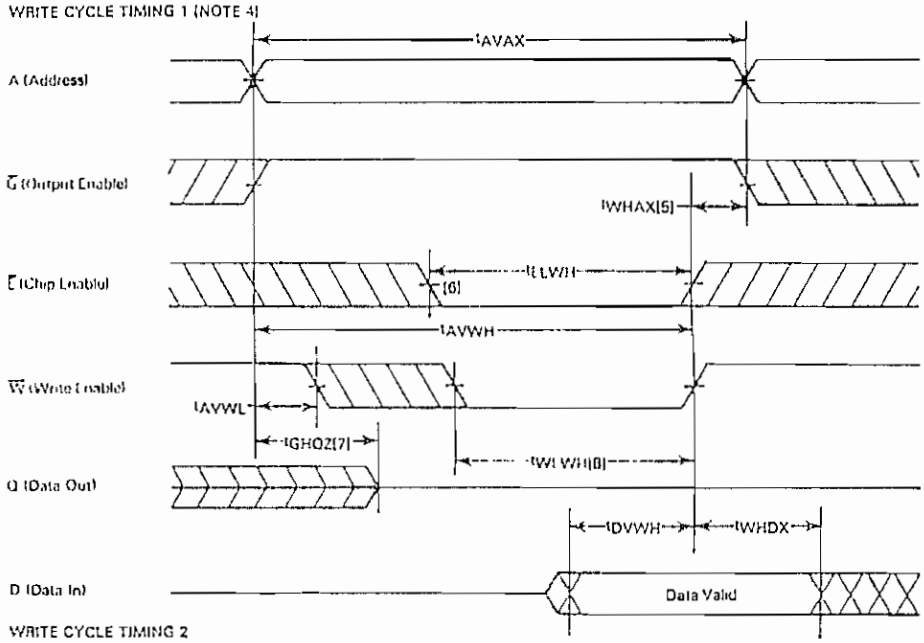


READ CYCLE TIMING 3
$\overline{G} = V_{IL}$ (NOTES 1, 2, 3)

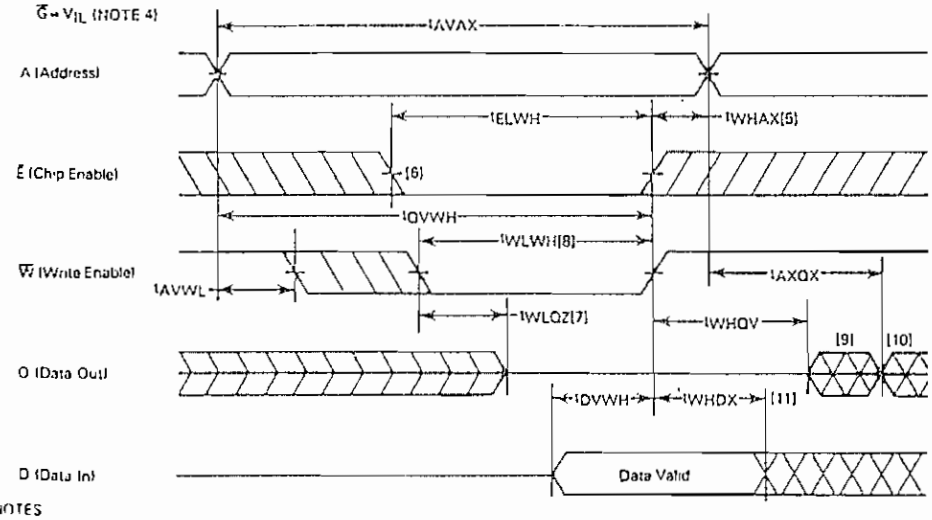


NOTES
1. Write Enable ($\overline{W}$) is High for Read Cycle
2. When Chip Enable ($\overline{E}$) is Low, the address input must not be in the high impedance state
3. Address Valid prior to or coincident with Chip Enable ($\overline{E}$) transition Low

WRITE CYCLE TIMING 1 (NOTE 4)



WRITE CYCLE TIMING 2
$\overline{G} = V_{IL}$ (NOTE 4)



NOTES

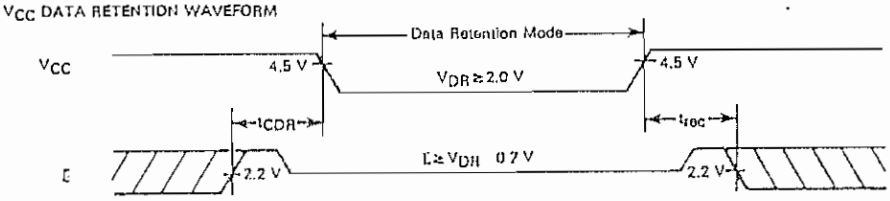
4. Write Enable ($\overline{W}$) must be high during all address transitions.
5. t_{WHAX} is measured from the earlier of Chip Enable ($\overline{E}$) or Write Enable ($\overline{W}$) going high to the end of write cycle.
6. If the Chip Enable ($\overline{E}$) low transition occurs simultaneously with the Write Enable ($\overline{W}$) low transitions or after the Write Enable ($\overline{W}$) transition, the output remains in a high impedance state.
7. During this period, DQ pins are in the output state so that the input signals of opposite phase to the outputs must not be applied.
8. A write occurs during the overlap of a low Chip Enable ($\overline{E}$) and a low Write Enable ($\overline{W}$).
9. Q (Data Out) is the same phase as write data of this write cycle.
10. Q (Data Out) is the read of the next address.
11. If Chip Enable ($\overline{E}$) is low during this period, DQ pins are in the output state. Then the data input signals of opposite phase to the outputs must not be applied to them.

LOW V_{CC} DATA RETENTION CHARACTERISTICS (T_A = 0 to + 70°C) (MCM65L116 Only)

Parameter	Conditions	Symbol	Min	Typ	Max	Unit
V_{CC} for Data Retention	$\overline{E} \geq V_{CC} - 0.2$ V $V_{in} \geq V_{CC} - 0.2$ V or $V_{in} \leq 0.2$ V	V_{DR}	2.0	–	–	V
Data Retention Current	V_{CC} = 3.0 V, $\overline{E} \geq 2.8$ V $V_{in} \geq 2.8$ V or $V_{in} \leq 0.2$ V	I_{CCDR}	–	–	50	μA
Chip Disable to Data Retention Time	See Retention Waveform	t_{CDR}	0	–	–	ns
Operation Recovery Time		t_{rec}	*t_{AVAX}	–	–	ns

*t_{AVAX} = Read Cycle Time.

LOW V_{CC} DATA RETENTION WAVEFORM



intel

PRELIMINARY

2764
(8K x 8) UV ERASABLE PROM

- 200 ns (2764-2) Maximum Access Time . . . HMOS*-E Technology
- Compatible to High Speed 8mHz 8086-2 MPU . . . Zero WAIT State
- Two Line Control
- Pin Compatible to 2732A EPROM
- Industry Standard Pinout . . . JEDEC Approved
- Low Active Current...100mA Max.

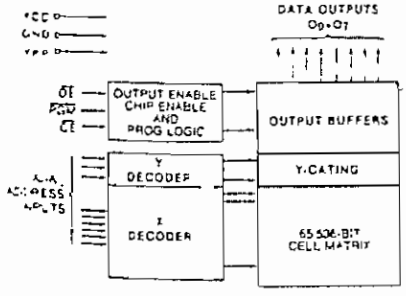
The Intel* 2764 is a 5V only, 65,536-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The standard 2764 access time is 250ns with speed selection available at 200ns. The access time is compatible to high performance microprocessors, such as Intel's 8mHz 8086-2. In these systems, the 2764 allows the microprocessor to operate without the addition of WAIT states.

An important 2764 feature is the separate output control, Output Enable ($\overline{OE}$) from the Chip Enable control ($\overline{CE}$). The $\overline{OE}$ control eliminates bus contention in multiple bus microprocessor systems. Intel's Application Note AP-72 describes the microprocessor system implementation of the $\overline{OE}$ and $\overline{CE}$ controls on Intel's EPROMs. AP-72 is available from Intel's Literature Department.

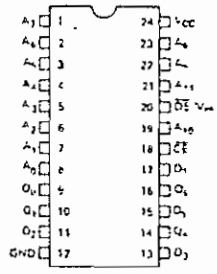
The 2764 has a standby mode which reduces the power dissipation without increasing access time. The active current is 150mA while the standby current is only 50mA. The standby mode is achieved by applying a TTL-high signal to the $\overline{CE}$ input.

The 2764 is fabricated with HMOS*-E technology, Intel's high-speed N-channel MOS Silicon Gate Technology.

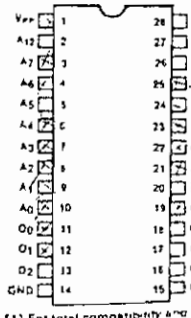
BLOCK DIAGRAM



2732A PIN CONFIGURATION



2764 PIN CONFIGURATION



(1) For total compatibility and upgradability from the 2732A and ROMs provide a trace to pin 26

MODE SELECTION

MODE \ PINS	CE (20)	OE (22)	PGM (27)	V_{PP} (1)	V_{CC} (28)	Outputs (11-13, 15-19)
Read	V_{IL}	V_{IL}	V_{IH}	V_{CC}	V_{CC}	D_{OUT}
Standby	V_{IH}	X	X	V_{CC}	V_{CC}	High Z
Program	V_{IL}	X	V_{IL}	V_{PP}	V_{CC}	D_{IN}
Program Verify	V_{IH}	X	V_{IH}	V_{PP}	V_{CC}	D_{OUT}
Program Inhibit	V_{IH}	X	X	V_{PP}	V_{CC}	High Z

X can be either V_{IL} or V_{IH}.

PIN NAMES

A0-A12	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
O0-O7	OUTPUTS
PGM	PROGRAM
N.C.	NO CONNECT

PROGRAMMING

The programming specifications are described in the Data Catalog PROM/ROM Programming Instructions Section.

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .....−10°C to +80°C
Storage Temperature .....−65°C to +125°C
All Input or Output Voltages with Respect to Ground .....+6V to −0.6V
V_{PP} Supply Voltage with Respect to Ground During Programming .....+22V to −0.6V

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability

READ OPERATION

D.C. AND OPERATING CHARACTERISTICS

Symbol	Parameter	Limits			Unit	Conditions
		Min	Typ[3]	Max		
I_{LI}	Input Load Current			10	μA	V_{IN} = 5.25V
I_{LO}	Output Leakage Current			10	μA	V_{OUT} = 5.25V
I_{PP1}[2]	V_{PP} Current Read			15	mA	V_{PP} = 5.25V
I_{CC1}[2]	V_{CC} Current Standby			50	mA	$\overline{CE}$ = V_{IH}
I_{CC2}[2]	V_{CC} Current Active		70	150	mA	$\overline{CE}$ = $\overline{OE}$ = V_{IL}
V_{IL}	Input Low Voltage	−.1		+.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}+1	V	
V_{OL}	Output Low Voltage			.45	V	I_{OL} = 2.1 mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −400 μA

NOTES: 1 V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2 V_{PP} may be connected directly to V_{CC} except during programming. The supply current would then be the sum of I_{CC} and I_{PP}.
3 Typical values are for T_A = 25°C and nominal supply voltages.

A.C. CHARACTERISTICS

Symbol	Parameter	2764-2 Limits		2764 Limits		2764-3 Limits		2764-4 Limits		Unit	Test Conditions
		Min	Max	Min	Max	Min	Max	Min	Max		
t_{ACC}	Address to Output Delay		200		250		300		450	ns	$\overline{CE}$=$\overline{OE}$=V_{IL}
t_{CE}	$\overline{CE}$ to Output Delay		200		250		300		450	ns	$\overline{OE}$=V_{IL}
t_{OE}	Output Enable to Output Delay	10	70	10	100	10	150	10	150	ns	$\overline{CE}$=V_{IL}
t_{OF}	Output Enable High to Output Float	0	60	0	90	0	130	0	130	ns	$\overline{CE}$=V_{IL}
t_{OH}	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$ Whichever Occurred First	0		0		0		0		ns	$\overline{CE}$=$\overline{OE}$=V_{IL}

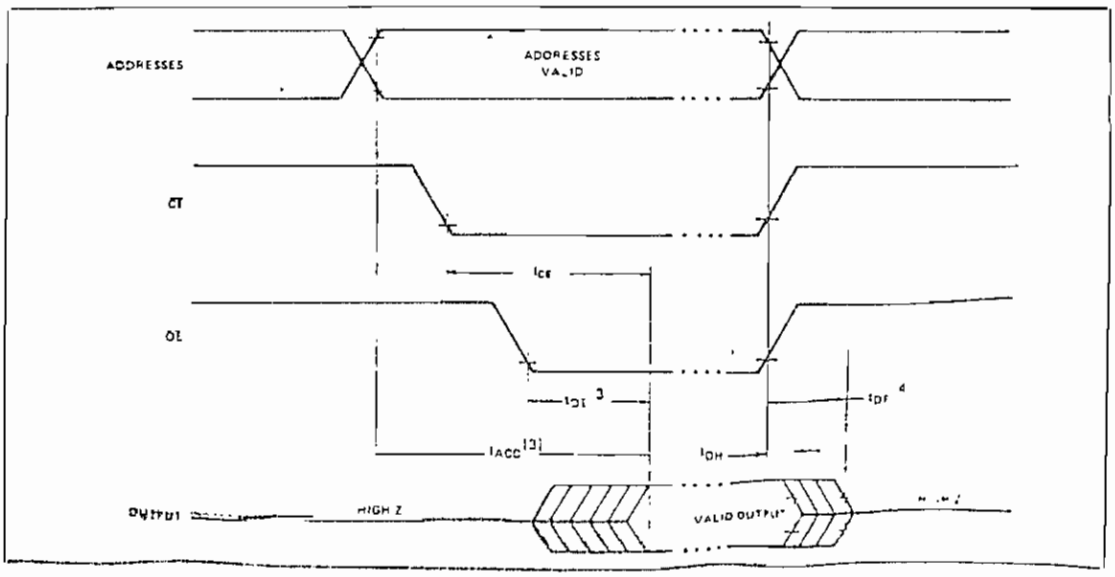
CAPACITANCE [1] T_A = 25°C, f = 1MHz

Symbol	Parameter	Typ.	Max.	Unit	Conditions
C_{IN}	Input Capacitance	4	6	pF	V_{IN} = 0V
C_{OUT}	Output Capacitance	8	12	pF	V_{OUT} = 0V

A.C. TEST CONDITIONS

Output Load: 1 TTL gate and C_L = 100pF
Input Rise and Fall Times: ≤ 20ns
Input Pulse Levels: 0.8V to 2.2V
Timing Measurement Reference Level:
Inputs 1V and 2V
Outputs 0.8V and 2V

A.C. WAVEFORMS



ERASURE CHARACTERISTICS

The erasure characteristics of the 2764 are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000 Å range. Data show that constant exposure to room level fluorescent lighting could erase the typical 2764 in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 2764 is to be exposed to these types of lighting conditions for extended periods of time, opaque labels are available from Intel which should be placed over the 2764 window to prevent unintentional erasure.

The recommended erasure procedure for the 2764 is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity X exposure time) for erasure should be a minimum of 15 W-sec/cm². The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with 12000 μW/cm² power rating. The 2764 should be placed within 1 inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure.

DEVICE OPERATION

The five modes of operation of the 2764 are listed in Table 1. A single 5V power supply is required in the read mode. All inputs are TTL levels except for V_{PP}.

TABLE 1. MODE SELECTION

MODE \ PINS	$\overline{CE}$ (20)	$\overline{OE}$ (22)	$\overline{PGM}$ (27)	V_{PP} (1)	V_{CC} (28)	Outputs (11-13, 15-19)
Read	V_{IL}	V_{IL}	V_{IH}	V_{CC}	V_{CC}	D_{OUT}
Standby	V_{IH}	x	x	V_{CC}	V_{CC}	High Z
Program	V_{IL}	x	V_{IL}	V_{PP}	V_{CC}	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IH}	V_{PP}	V_{CC}	D_{OUT}
Program Inhibit	V_{IH}	x	x	V_{PP}	V_{CC}	High Z

x can be either V_{IL} or V_{IH}.

READ MODE

The 2764 has two control functions, both of which must be logically satisfied in order to obtain data at the outputs. Chip Enable ($\overline{CE}$) is the power control and should be used for device selection. Output Enable ($\overline{OE}$) is the output control and should be used to gate data to the output pins, independent of device selection. Assuming that addresses are stable, address access time (t_{ACC}) is equal to the delay from $\overline{CE}$ to output (t_{CE}). Data is available at the outputs after the falling edge of $\overline{CE}$, assuming that $\overline{CE}$ has been low and addresses have been stable for at least $t_{ACC} - t_{CE}$.

Standby Mode

The 2764 has a standby mode which reduces the active power current from 150mA to 50mA. The 2764 is placed in the standby mode by applying a TTL high signal to the $\overline{CE}$ input. When in standby mode, the outputs are in a high impedance state, independent of the $\overline{OE}$ input.

Output OR-Tieing

Because EPROMs are usually used in larger memory arrays, Intel has provided a 2 line control function that accommodates this use of multiple memory connection. The two line control function allows for:

a) the lowest possible memory power dissipation, and
b) complete assurance that output bus contention will not occur.

To most efficiently use these two control lines, it is recommended that $\overline{CE}$ (pin 20) be decoded and used as the primary device selecting function, while $\overline{OE}$ (pin 22) be made a common connection to all devices in the array and connected to the READ line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are only active when data is desired from a particular memory device.

PROGRAMMING (See Programming Instruction Section for Waveforms.)
Programming is the same as Intel's 2732A except that $\overline{OE}/V_{PP}$ is not multiplexed. They have separate pins. Like the 2732A, ***exceeding 21.5V will damage the 2764.***

Initially, and after each erasure, all bits of the 2764 are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The 2764 is in the programming mode when V_{PP} input is at 21V and $\overline{CE}$ and $\overline{PGM}$ are both at TTL low. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

For programming, $\overline{CE}$ should be kept TTL low at all times while V_{PP} is kept at 21V. When the address and data are stable, a 50 msec, active low, TTL program pulse is applied to $\overline{PGM}$ input. A program pulse must be applied at each address location to be programmed. You can program any location at any time—either individually, sequentially, or at random. The program pulse has a maximum width of 55 msec.

Programming of multiple 2764s in parallel with the same data can be easily accomplished due to the simplicity of the programming requirements. Like inputs of the paralleled 2764s may be connected together when they are programmed with the same data. A low level TTL pulse applied to the $\overline{PGM}$ input programs the paralleled 2764s.

Program Inhibit

Programming of multiple 2764s in parallel with different data is also easily accomplished. A high level $\overline{CE}$ or $\overline{PGM}$ input inhibits the other 2764s from being programmed. Except for $\overline{CE}$ (or $\overline{PGM}$), all like inputs (including $\overline{OE}$) of the parallel 2764s may be common. A TTL low level pulse applied to a 2764 $\overline{CE}$ and $\overline{PGM}$ input with V_{PP} at 21V will program that 2764.

Program Verify

A verify should be performed on the programmed bits to determine that they were correctly programmed. The verify is accomplished with $\overline{CE}$ and $\overline{OE}$ at V_{IL}. However, $\overline{PGM}$ is at V_{IH}.

National Semiconductor

LM78XX Series Voltage Regulators

General Description

The LM78XX series of three terminal regulators is available with several fixed output voltages making them useful in a wide range of applications. One of these is local on card regulation, eliminating the distribution problems associated with single point regulation. The voltages available allow these regulators to be used in logic systems, instrumentation, HiFi, and other solid state electronic equipment. Although designed primarily as fixed voltage regulators these devices can be used with external components to obtain adjustable voltages and currents.

The LM78XX series is available in an aluminum TO-3 package which will allow over 1.0A load current if adequate heat sinking is provided. Current limiting is included to limit the peak output current to a safe value. Safe area protection for the output transistor is provided to limit internal power dissipation. If internal power dissipation becomes too high for the heat sinking provided, the thermal shutdown circuit takes over preventing the IC from overheating.

Considerable effort was expended to make the LM78XX series of regulators easy to use and minimize the number of external components. It is not necessary to bypass the output, although this does improve transient response. Input bypassing is needed only if the regulator is located far from the filter capacitor of the power supply.

For output voltage other than 5V, 12V and 15V the LM117 series provides an output voltage range from 1.2V to 57V.

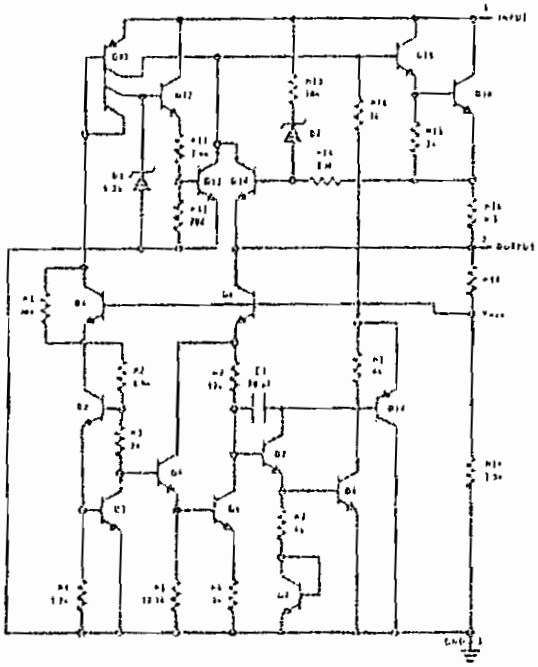
Features

- Output current in excess of 1A
- Internal thermal overload protection
- No external components required
- Output transistor safe area protection
- Internal short circuit current limit
- Available in the aluminum TO-3 package

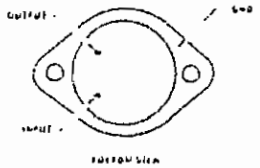
Voltage Range

LM7805C	5V
LM7812C	12V
LM7815C	15V

Schematic and Connection Diagrams



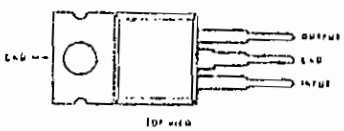
Metal Can Package
TO-3 (K)
Aluminum



bottom view

Order Numbers:
LM7805CK
LM7812CK
LM7815CK
See Package K02A

Plastic Package
TO-220 (T)



top view

Order Numbers:
LM7805CT
LM7812CT
LM7815CT
See Package T03B

bsolute Maximum Ratings

put Voltage (V_O = 5V, 12V and 15V)	35V
ternal Power Dissipation (Note 1)	Internally Limited
perating Temperature Range (T_A)	0°C to +70°C
aximum Junction Temperature	
(K Package)	150°C
(T Package)	125°C
torage Temperature Range	−65°C to +150°C
ead Temperature (Soldering, 10 seconds)	
TO-3 Package K	300°C
TO-220 Package T	230°C

lectrical Characteristics LM78XXC (Note 2) 0°C ≤ TJ ≤ 125°C unless otherwise noted.

OUTPUT VOLTAGE			5V			12V			15V			UNITS
INPUT VOLTAGE (unless otherwise noted)			10V			19V			23V			
PARAMETER		CONDITIONS	MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
		TJ = 25°C, 5 mA ≤ I_O ≤ 1A	4.8	5	5.2	11.5	12	12.5	14.4	15	15.6	V
V_O	Output Voltage	P_D ≤ 15W, 5 mA ≤ I_O ≤ 1A V_{MIN} ≤ V_{IN} ≤ V_{MAX}	4.75 (7 ≤ V_{IN} ≤ 20)		5.25	11.4 (14.5 ≤ V_{IN} ≤ 27)		12.6	14.25 (17.5 ≤ V_{IN} ≤ 30)		15.75	V V
ΔV_O	Line Regulation	I_O = 500 mA, TJ = 25°C		3	50		4	120		4	150	mV
		I_O = 500 mA, ΔV_{IN}		(7 ≤ V_{IN} ≤ 25)			(14.5 ≤ V_{IN} ≤ 30)			(17.5 ≤ V_{IN} ≤ 30)		V
		I_O = 500 mA, 0°C ≤ TJ ≤ +125°C, ΔV_{IN}			50			120			150	mV
		I_O = 500 mA, ΔV_{IN}		(8 ≤ V_{IN} ≤ 20)			(15 ≤ V_{IN} ≤ 27)			(18.5 ≤ V_{IN} ≤ 30)		V
		I_O ≤ 1A, TJ = 25°C, ΔV_{IN}			50			120			150	mV
		I_O ≤ 1A, ΔV_{IN}		(7.3 ≤ V_{IN} ≤ 20)			(14.8 ≤ V_{IN} ≤ 27)			(17.7 ≤ V_{IN} ≤ 30)		V
		I_O ≤ 1A, 0°C ≤ TJ ≤ +125°C, ΔV_{IN}			25			60			75	mV
		I_O ≤ 1A, ΔV_{IN}		(6 ≤ V_{IN} ≤ 12)			(10 ≤ V_{IN} ≤ 22)			(20 ≤ V_{IN} ≤ 26)		V
ΔV_O	Load Regulation	TJ = 25°C, 5 mA ≤ I_O ≤ 1.5A		10	50		12	120		12	150	mV
		TJ = 25°C, 250 mA ≤ I_O ≤ 750 mA			25			60			75	mV
		5 mA ≤ I_O ≤ 1A, 0°C ≤ TJ ≤ +125°C			50			120			150	mV
I_Q	Quiescent Current	I_O ≤ 1A, TJ = 25°C			8			8			8	mA
		I_O ≤ 1A, 0°C ≤ TJ ≤ +125°C			8.5			8.5			8.5	mA
ΔI_Q	Quiescent Current Change	5 mA ≤ I_O ≤ 1A			0.5			0.5			0.5	mA
		TJ = 25°C, I_O ≤ 1A, V_{MIN} ≤ V_{IN} ≤ V_{MAX}			1.0			1.0			1.0	mA
		I_O ≤ 500 mA, 0°C ≤ TJ ≤ +125°C, V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(7 ≤ V_{IN} ≤ 25)	1.0		(14.5 ≤ V_{IN} ≤ 30)	1.0		(17.5 ≤ V_{IN} ≤ 30)	1.0	V mA V
V_N	Output Noise Voltage	TA = 25°C, 10 Hz ≤ f ≤ 100 kHz		40			75			90		μV
$\frac{\Delta V_{IN}}{\Delta V_{OUT}}$	Ripple Rejection	f = 120 Hz, I_O ≤ 1A, TJ = 25°C or I_O ≤ 500 mA, 0°C ≤ TJ ≤ +125°C	62 62	80		55 55	72		54 54	70		dB dB
		V_{MIN} ≤ V_{IN} ≤ V_{MAX}		(8 ≤ V_{IN} ≤ 18)			(15 ≤ V_{IN} ≤ 25)			(18.5 ≤ V_{IN} ≤ 28.5)		V
	Dropout Voltage	TJ = 25°C, I_{OUT} = 1A		2.0			2.0			2.0		V
	Output Resistance	f = 1 kHz		8			18			19		mΩ
I_O	Short Circuit Current	TJ = 25°C		2.1			1.5			1.2		A
	Peak Output Current	TJ = 25°C		2.4			2.4			2.4		A
	Average TC of V_{OUT}	0°C ≤ TJ ≤ +125°C, I_O = 5 mA		0.6			1.5			1.8		mV/°C
V_{IN}	Input Voltage Required to Maintain Line Regulation	TJ = 25°C, I_O ≤ 1A	7.3			14.6			17.7			V

NOTE 1: Thermal resistance of the TO-3 package (K, KC) is typically 1°C/W junction to case and 35°C/W case to ambient. Thermal resistance of the TO-220 package (T) is typically 4°C/W junction to case and 50°C/W case to ambient.

NOTE 2: All characteristics are measured with capacitor across the input of 0.22 μF, and a capacitor across the output of 0.1 μF. All characteristics except noise voltage and ripple rejection ratio are measured using pulse techniques (t_W ≤ 10 ms, duty cycle ≤ 5%). Output voltage changes due to

B I B L I O G R A F I A

- Misha Schwartz — TRANSMISION DE INFORMACION MODULACION Y RUIDO.- 3a. Edición, Mc Graw - Hill, 1983, México.

- Vitervi A, Omura Jim — PRINCIPLES OF DIGITAL COMUNICATION AND CODING.- Mc Graw - Hill, 1979.

- Gallager Robert. — INFORMATION THEORY AND REALIABLE COMUNICATION. John Wiley and Sons Inc, 1968, USA.

Mc. Eliece Robert — THE THEORY OF INFORMATION AND CODING.- 2a. Edición, vol III, Addison Wesley, 1979.

Shu Lin, Costelo — ERROR CONTROL CODING FUNDAMENTALS AND APLICATIONS. Prentice Hall, 1983.

Peterson Wesley — ERROR CORRECTING CODES, 2a. Edición, MIT Press, 1.962.

R. F. W. Coates — MODERN.COMUNICATION SYSTEMS, 5a. Edición. Macmillan Press Ltd, 1981.

- Berlekamp E. R. — ALGEBRAIC CODING THEORY, Mc Graw - Hill, 1968. New York.

- Ash Robert — INFORMATION THEORY, John Wileym 1976.

- Wozencraft and Jacobs — PRINCIPLES OF COMUNICATION ENGINEEERING. John Wileym 1976, New York.

- K. Sam Sammugan — DIGITAL AND ANALOG COMUNICATION SYSTEMS.=- John Wiley, 1979.

- Carlson Bruce — SISTEMAS DE COMUNICATION. Mc. Graw - Hill, 1980, México.

- Hamming Richard — CODING AND INFORMATION THEORY.- Prentice - Hall, 1980.

- Abramson Norman — INFORMATION THEORY AND CODING. Mc Graw - Hill, 1963, New York.

- Dixon R. Doll — DATA COMUNICATIONS. Wiley Interscience,1978.

- Rodnay Zaks — MICROPROCESADORES. Marcombo, 1979, Barcelona.

- IEEE Press — DATA COMUNICATIONS VIA FANDING CHANNELS.1975.

- Multitech — MICRO-PROFFESOR USERS MANUAL, 1983.

- Intel — COMPONENT DATA CATALOG, 1978.

- Motorola — MEMORY DATA MA=NUAL, 1980.

- Zilog, — ZILOG COMPONENTS DATA BOOK, 1983- 1984.

*** & ***