UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE INGENIERIA

PROCESAMIENTO EN PARALELO EN LA DECODIFICACION DE CODIGOS DE REED - SOLOMON

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N :

OSCAR GARCIA CARDENAS
SAMUEL GUTIERREZ CARDENAS

Director de Tesis:
Dr. Francisco J. Garcia Ugalde

MEXICO, D. F. 1989

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

Lista de Figuras

LISTA DE TABLAS

LISTA DE GRAFICAS

Página

LISTA DE DIAGRAMAS

LISTA DE CIRCUITOS

INTRODUCCION

En la transmisión de datos digitales por un canal con ruido, hay una probabilidad de que los datos recibidos contengan errores. El usuario generalmente establece una tasa de errores arriba de la cual esos datos no se pueden utilizar. Si los datos recibidos no cumplen con la tasa de errores requerida, frecuentemente se puede utilizar una codificación para la corrección de errores, con el propósito de reducir los errores a un nivel tolerable. En los últimos años el uso de la codificación para la corrección de errores, en sistemas de comunicaciones digitales, se ha utilizado ampliamente para resolver este tipo de problema.

La utilidad de la corrección fue demostrada por el trabajo de Shannon [1]. En 1948, estableció que si la tasa de transmisión no sobrepasa una cantidad llamada la capacidad del canal, es posible obtener una comunicación a través de un canal con ruido con una probabilidad de error tan pequeña como se desée, utilizando adecuadamente un esquema de codificación y de decodificación. Escencialmente el trabajo de Shannon establece que la potencia de la señal, el ruido del canal y el ancho de banda disponible establecen un límite solamente sobre la tasa de comunicación y no sobre la precisión.

La codificación para la corrección de errores es escencialmente una técnica de procesamiento de señales que se utiliza para mejorar el desempeño de la comunicación en canales digitales. Aunque los esquemas individuales de codificación cuentan con muchas formas diferentes todos ellos tienen dos ingredientes comunes: la redundancia y el ruido promediado. Los mensajes digitales codificados siempre contienen símbolos redundantes. Esos símbolos

se utilizan para acentuar el carácter único de cada mensaje, siempre se escogen de tal manera que sea muy improbable que la perturbación del canal modifique los símbolos de un mensaje en grado tal que, como resultado se destruya su carácter único. El efecto de tener un ruido promediado se obtiene haciendo que los símbolos redundantes dependan de una secuencia de varios símbolos de información.

Actualmente existen dos tipos de códigos de uso común: los códigos de bloque y los códigos convolucionales. El codificador de un código de bloque binario divide la secuencia de información en bloques del mensaje de k bits de información cada uno. Un bloque del mensaje está representado por la secuencia binaria de k símbolos de información $i = (i_1, i_2, \ldots, i_k)$, llamada k-tuple. Existe un total de 2^k mensajes posibles diferentes. El codificador transforma cada mensaje independiente i, en una n-tuple $c = (c_1, c_2, \ldots, c_n)$ de símbolos discretos, llamada una palabra del código, donde $k<n$. Por lo tanto, a cada uno de los 2^k diferentes mensajes posibles, les corresponde una de las 2^k diferentes palabras del código a la salida del codificador. A este conjunto de 2^k palabras del código de longitud n se le llama un *código de bloque* (n,k). A la proporción $R = k/n$ se le llama tasa del código y se le puede interpretar como el número de bits de información que entran al codificador por cada símbolo transmitido. Dado que la palabra del código de n símbolos depende solamente del correspondiente mensaje de entrada de k bits, al codificador se le llama *sin memoria*. Considerando que $k<n$, se pueden agregar al mensaje n-k bits de redundancia para formar una palabra del código. Estos bits de redundancia proporcionan al código la capacidad de combatir el ruido del canal. La manera de escoger los bits de redundancia para alcanzar una transmisión confiable sobre un canal con ruido, es el principal problema en el diseño del codificador.

El codificador de un código convolucional también acepta bloques de k bits de la secuencia de información *I* y produce una secuencia codificada *C* de bloques de n símbolos cada uno, donde k<n. Sin embargo, cada bloque codificado depende no solamente del correspondiente bloque de k bits del mensaje en la misma unidad de tiempo, sino también de los *m* bloques previos del mensaje. Es decir que el codificador tiene una memoria de orden *m*. El conjunto de secuencias codificadas producido por un codificador de k bits de entrada, n símbolos de salida y memoria de orden *m*, es llamado *código convolucional* (n,k,*m*). A la proporción $R = k/n$, se le llama tasa del código. Dado que el codificador contiene memoria, se debe realizar con un circuito lógico secuencial. Típicamente en estos códigos, k y n son enteros pequeños y se puede mejorar el desempeño del código con respecto al ruido del canal aumentando el orden *m* de la memoria del código, mientras se mantienen fijos los valores de k y n. El principal problema en el diseño del codificador convolucional consiste en saber cómo utilizar la memoria para alcanzar una transmisión confiable sobre un canal con ruido.

Dentro de los códigos de bloque existen varias familias como son: los códigos de Hamming, los códigos BCH, los códigos de Reed-Solomon, etc. En particular los códigos de Reed-Solomon cuentan con una gran habilidad para corregir errores, debido a esta característica fueron seleccionados para este trabajo.

Uno de los objetivos principales del trabajo que se presenta está orientado a la simulación de tres algoritmos de decodificación de códigos de Reed-Solomon y su evaluación, con el fin de encontrar el algoritmo más rápido de los tres.

Los tres algoritmos a estudiar son: el Algoritmo Clásico, una modificación de este algoritmo, llamado Algoritmo Modificado y el tercero que es un

algoritmo que opera en el dominio de la frecuencia.

La simulación mencionada de estos algoritmos se llevó a cabo mediante programas realizados en lenguaje FORTRAN. Tal simulación permitió hacer la evaluación de los tres algoritmos en cuanto a su velocidad de decodificación para así poder seleccionar el más rápido de ellos y posteriormente realizar su diseño en hardware con elementos discretos y una arquitectura en paralelo.

A continuación se hace un pequeño resumen del contenido de cada capítulo de esta tesis:

En el primer capítulo se describen de manera muy breve varios tipos de códigos correctores de errores con sus principales características. En el capítulo dos se muestra en detalle cada uno de los tres algoritmos de decodificación a estudiar, se describen paso a paso las etapas de que consta cada uno de ellos y se incluye además un diagrama de flujo de cada algoritmo y un diagrama de flujo de su simulación en el cual se pueden observar cada una de sus etapas con sus respectivas variables de entrada y salida. En el capítulo tres se presentan tablas y gráficas correspondientes a los tres algoritmos las cuales permitieron hacer una comparación entre ellos para así seleccionar el más rápido. Y en el cuarto y último capítulo se encuentra el diseño del decodificador correspondiente al algoritmo seleccionado, con arquitectura en paralelo con el propósito de disminuir su tiempo de ejecución.

Esto último puede ser considerado como la aportación más importante de este trabajo ya que el ahorro de tiempo en la decodificación de información es de gran importancia en sistemas digitales.

Existen trabajos ya realizados en cuanto al diseño de codificadores y decodificadores Reed-Solomon con tecnología VLSI y arquitectura pipeline [6,7,10,11]. Las ventajas de este tipo de tecnología consisten en la

reducción del tamaño, el peso y el consumo de potencia a la vez que proveen una alta velocidad de operación por encima de codificadores y decodificadores Reed-Solomon implementados con circuitos lógicos discretos.

CAPITULO I. REVISION TEORICA DE LAS TECNICAS DE DETECCION Y CORRECCION DE ERRORES.

En este capitulo se describen brevemente las principales técnicas de detección y corrección de errores así como sus principales caracteristicas.

I.1 CODIGOS POLINOMIALES.

En principio, una cadena de n bits se puede representar como un polinomio de grado n-1. Mediante el uso de esta notación, un código (n,k) puede ser definido como el conjunto de todos los polinomios de grado n-1 o menor, el cual contiene a g(x), que es un polinomio de grado n-k, como un factor del código y es llamado polinomio generador del código.

Para ciertos valores de n, los códigos polinomiales son ciclicos ya que si rotamos ciclicamente una palabra del código obtenemos otra palabra del código [3,4]. Esto se muestra en la siguiente figura:

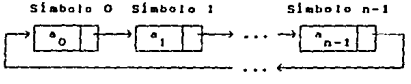


fig I.1 Codigos Ciclicos

Para este tipo de códigos se puede definir un polinomio de chequeo de paridad h(x), que sirve para la detección y corrección de errores en algunos algoritmos de decodificación [4].

I.2 CODIGOS DE BLOQUE.

El principio fundamental de los códigos de bloque es el agrupamiento de k bits de información dentro de bloques codificados de largo n. En el caso de los sistemas binarios, se añaden ℓ bits de chequeo de paridad al bloque y se tienen $n=\ell+k$ bits del código hacia el sistema digital [4].

I.3 CODIGOS DE BLOQUE LINEALES.

Un código de bloque de longitud n y 2^k palabras del código es llamado código de bloque lineal (n,k) si y sólo si sus 2^k palabras del código forman un subespacio k-dimensional del espacio vectorial de todas las n-tuples sobre el campo CG(2) [4].

Un código de bloque binario es lineal si y sólo si la suma módulo-2 de dos palabras del código es también una palabra del código [4].

Para estos códigos cada palabra del código C es una combinación lineal de k vectores independientes, donde k es el número de bits de información. El código se denota por (n,k,t) donde n es el largo del bloque codificado y n-k es el número de bits de redundancia que se utilizan para la detección y corrección de errores. El parámetro t es el número de bits erróneos que se pueden corregir dentro del bloque de largo n y es frecuentemente omitido cuando su valor es 1 [4].

La distancia mínima de un código de bloque lineal es igual al peso mínimo de sus palabras diferentes de cero. El peso mínimo de un código C es el número más pequeño de componentes diferentes de cero de cualquier palabra

diferente de cero del código [3,4].

Dado un código de bloque lineal de distancia mínima d, es posible corregir t errores [3,4] donde:

$$2t+1 <= d$$

I.4 CODIGOS DE HAMMING.

Una clase importante de códigos de bloque lineales son los llamados códigos de Hamming, fueron unos de los primeros grupos de códigos descritos sistematicamente. El código de Hamming puede corregir sólo un error y su distancia mínima es d=3 [4].

A partir de un código de Hamming (n,k) se puede obtener un código (n-i,k-i), donde i es un entero, tal que 0<=i<k, llamado código de Hamming acortado [4].

Ahora bien, para poder corregir los errores simples y al mismo tiempo detectar los errores dobles, se ideó un código llamado código de Hamming extendido cuya distancia mínima es d=4 [3].

I.5 CODIGOS B.C.H.

Los códigos B.C.H. (llamados así por Bose, Chaudhuri y Hocquenghem) son códigos correctores de errores múltiples y de distancia mínima variable. Debe cumplirse la siguiente desigualdad para su distancia mínima:

$$d >= 2t + 1$$

donde t es el número de errores que se pueden corregir en un bloque de largo

n. A medida que t aumenta, la redundancia aumenta.

Estos códigos se definen sobre un campo finito de Galois $CG(2^m)$. Son códigos cíclicos, tienen un polinomio generador g(x) de grado n-k, donde n es el largo del bloque codificado y k es el largo del bloque de información. La diferencia n-k es la redundancia [3,4].

I.6 CODIGOS DE REED - SOLOMON (R.S.).

Los códigos de Reed-Solomon fueron definidos por I.S. Reed y G. Solomon en 1960. Estos son una subclase especial de los códigos B.C.H. y tienen especial habilidad para la corrección de borrados y errores múltiples en una palabra del código.

Para un valor específico de t y m podemos construir un código de Reed-Solomon $(2^m-1, 2^m-1-2t)$, el cual es capaz de corregir t errores. Sus símbolos son elementos del campo $CG(2^m)$ y su polinomio generador g(x) está dado por:

$$g(x) = (x+\alpha)(x+\alpha^2)\ldots(x+\alpha^{2t})$$

donde α es un elemento primitivo en $CG(2^m)$ [2,4,16].

Todos los múltiplos de g(x) de grado (2^m-2) o menor son palabras del código. Cada palabra del código tiene $n=2^m-1$ símbolos, de los cuales k=n-2t son símbolos de información y 2t es el número de símbolos de redundancia (figura I.2). Un código Reed-Solomon (n,k) tiene una distancia mínima d=2t+1 [3,4].

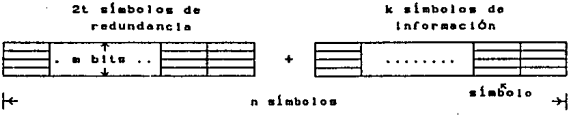


fig I.2 Palabra del Código

Un código Reed-Solomon es capaz de corregir tanto errores como borrados siempre que se cumpla la siguiente desigualdad d>=2e+f+1, siendo e el número de simbolos con error y f el número de simbolos borrados en un bloque de largo n. Para el decodificador un simbolo borrado es un error cuya posición se conoce antes de empezar la decodificación y su magnitud puede ser o no ser cero. La diferencia entre un error y un borrado es que, un error es un simbolo que sufrió una alteración en el canal durante la transmisión y se detecta hasta que se inicia la decodificación, mientras que un borrado es un simbolo marcado por el demodulador como desconfiable por no tener un nivel de voltaje bien definido.

I.7 CODIGOS CONVOLUCIONALES.

El segundo grupo más grande de códigos es el de los códigos llamados convolucionales o de árbol.

Estos esquemas son muy efectivos cuando el requerimiento mayor de un sistema está en el aprovechamiento óptimo de la energia (tal como en el caso de la potencia limitada en los canales de comunicaciones de un satélite).

La entrada y la salida de un codificador convolucional (cuando k=1) es una secuencia semi-infinita de simbolos [3,4].

Para este trabajo se seleccionó un código de bloque de tipo Reed-Solomon debido en parte a su habilidad para corregir errores y borrados y a que actualmente existen muchas aplicaciones [17-22] en donde la información viene ya en forma de bloques lo cual facilita su manipulación , aparte de que el ancho de banda que requieren estos códigos para transmitir la redundancia, es menor al que requiere un código convolucional.

Capitulo II. Metodos Alternativos de Decodificacion de Codigos R.S.

En este capitulo se estudiarán tres diferentes algoritmos de decodificación para códigos de Reed-Solomon (R.S.). Dos de ellos son algoritmos que trabajan en el dominio del "tiempo", mientras que el tercero es un algoritmo que trabaja en el dominio de la "frecuencia" y emplea un método basado en una transformada rápida.

Antes de entrar a los algoritmos de decodificación se describirán en forma breve las partes de un sistema digital típico de transmisión de datos y también el proceso de codificación para un código de Reed-Solomon.

En años recientes, se ha incrementado la demanda por sistemas digitales eficientes y confiables de transmisión de datos. Esta demanda se ha acelerado debido a la necesidad de redes de datos a gran escala y de alta velocidad para el intercambio, procesamiento y almacenamiento de información digital. Recientes desarrollos han contribuido para lograr la confiabilidad que los sistemas digitales de alta velocidad requieren hoy en dia y el uso de la codificación para el control de errores se ha convertido en parte integral en el diseño de sistemas modernos de comunicaciones y de computadoras digitales.

La transmisión y almacenamiento de información digital tienen mucho en común. Ambas transfieren datos de una fuente de información a un destino (o usuario). Un sistema típico de transmisión (o de almacenamiento) puede representarse como se muestra en la figura II.0. La *fuente de información* puede ser una persona o una máquina (computadora digital). La salida de la fuente, puede ser una onda continua o una secuencia de símbolos discretos. El *codificador de la fuente* transforma la salida de la fuente en una secuencia de dígitos binarios (bits) llamada secuencia de información i. En el caso de

una fuente continua, este involucra una conversión analógica a digital (A/D).

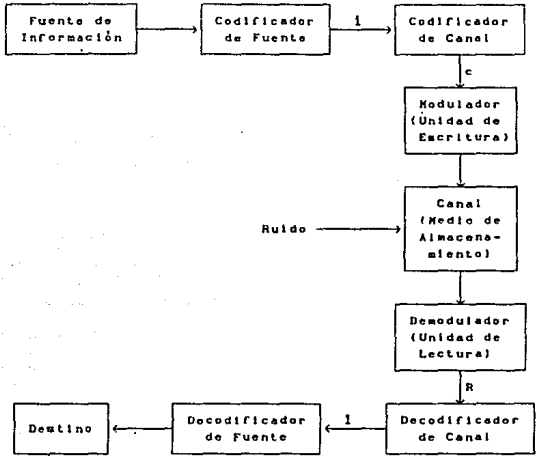


fig. II.0 Diagrama de bloques de un sistema digital típica de transmisión de datos.

El *codificador de canal* transforma la secuencia de información i en una *secuencia codificada* discreta c llamada *palabra del código*. Los símbolos discretos no son los apropiados para la transmisión a través de un canal físico o grabación en un medio de almacenamiento digital, en consecuencia, el *modulador* (o *unidad de escritura*) transforma cada símbolo de salida del codificador de canal en una forma de onda de T segundos de duración, la cual si es adecuada para la transmisión (o grabación). Esta forma de onda entra al *canal* (o *medio de almacenamiento*) y frecuentemente es contaminada por el

ruido. El *demodulador* (o *unidad de lectura*) procesa cada forma de onda de duración T recibida y produce una salida que puede ser discreta (cuantizada) o continua (no cuantizada). La secuencia de salida del demodulador correspondiente a la secuencia codificada c es llamada *secuencia recibida* R.

El *decodificador de canal* transforma la secuencia recibida R en una secuencia binaria î llamada *secuencia estimada*. El *decodificador de la fuente* transforma la *secuencia estimada* î en una estimación de la salida de la fuente y la envía al *destino*. Cuando la fuente es continua, este involucra una conversión digital a analógica (D/A). En un sistema bien diseñado, la estimación será una réplica de la salida de la fuente excepto cuando el canal (o medio de almacenamiento) sea muy ruidoso.

Ahora bien, pasando a la descripción del proceso de codificación, en un codificador R.S., la información se agrupa en bloques de k símbolos. A cada uno de estos bloques se le agregan n-k símbolos de redundancia, obteniéndose una palabra del código de n símbolos donde n>k. Los n-k símbolos de redundancia se utilizan para la corrección de errores y se obtienen de los coeficientes del residuo de:

$$x^{n-k}\ i(x)/g(x)$$

donde i(x) es el polinomio de información , cuyos coeficientes son los k símbolos de información, esto es:

$$i(x) = \sum_{l=0}^{k-1} i_l x^l = i_0 + i_1 x + i_2 x^2 + \ldots + i_{k-1} x^{k-1}$$

donde $i_l \in GG(2^m)$. g(x) es el polinomio generador del código definido por:

$$g(x) = \prod_{l=j}^{j+2t-1} (x-\alpha^l) = \sum_{l=0}^{2t} g_l x^l$$

donde j es un entero no negativo, frecuentemente su valor es 1; α es un

elemento primitivo en el $CG(2^m)$ y los coeficientes g_i's $\in CG(2^m)$ donde $g_{2t} = 1$. El polinomio generador g(x) definido no tiene coeficientes simétricos, es decir:

$$g_i \neq g_{2t-i} \quad \text{para } 0 \leq i \leq 2t$$

excepto cuando $j = 2^{m-1} - t$, en este caso:

$$g_i = g_{2t-i} \quad \text{para } 0 \leq i \leq 2t$$

y

$$g_0 = g_{2t} = 1$$

Nótese que en este caso, sólo t multiplicadores son necesarios en un codificador. Usando este polinomio generador se reducirá el número de multiplicadores requeridos para implementar el codificador R.S.

La palabra del código c(x) obtenida de la codificación, está formada por una primera parte que es igual a los símbolos de información:

$$c_i = i_{i-n+k} \quad \text{para } n-1 \geq i \geq n-k$$

y una segunda parte que es igual a los símbolos de redundancia:

$$c_i = r_i \quad \text{para } n-k-1 \geq i \geq 0$$

donde los coeficientes c_i's de c(x) son los símbolos de la palabra del código y los r_i's son los coeficientes del polinomio de redundancia r(x). Entonces la palabra del código a transmitir está dada por:

$$c(x) = x^{n-k} i(x) + r(x) = \sum_{i=0}^{n-1} c_i x^i = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1}$$

El siguiente paso, después de la transmisión de la palabra codificada c(x) por un canal con ruido, es su decodificación, lo cual se puede realizar mediante alguno de los métodos mencionados y que se describen a continuación.

II.1 ALGORITMO CLASICO DE DECODIFICACION PARA CODIGOS R.S.

Después de la transmisión sobre un canal con ruido, la palabra recibida es:

$$R(x)=\sum_{i=0}^{n-1} R_i x^i = R_0 + R_1 x + \ldots + R_{n-1} x^{n-1} \text{ ; donde } R_i \in CG(2^m) \quad \ldots(1)$$

El patrón de error sumado por el canal es:

$$E(x) = R(x) - c(x) = \sum_{i=0}^{n-1} E_i x^i = E_0 + E_1 x + \ldots + E_{n-1} x^{n-1} \quad \ldots(2)$$

donde

$$E_i = R_i - c_i \; ; \; E_i \in CG(2^m) \quad 0 \leq i \leq n-1 \quad \ldots(3)$$

y c(x) es la palabra del código transmitida.

La función del decodificador es encontrar la palabra del código que tenga la distancia más pequeña con respecto a la palabra recibida. Si el patrón de error es mayor a d, la decodificación falla y se pueden presentar cualquiera de las dos condiciones siguientes: se detecta una decodificación imposible, o bien el decodificador decodificará incorrectamente y resultará un error de pos-decodificación no detectable. Este último caso se presenta cuando el patrón de error está localizado dentro de una esfera de radio t (en un espacio de n dimensiones) que rodea a una palabra del código diferente de la palabra del código transmitida, el patrón de error se decodifica como la palabra del código del centro de la esfera. La figura II.1 ilustra estos casos.

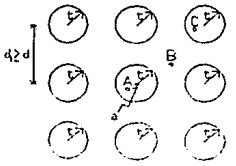


fig II.1 Representación esquemática de la decodificación.

a. Palabra de código transmitida.

A. Palabra recibida que producirá una decodificación correcta.

B. Palabra recibida que ocasionará la detección de una decodificación imposible.

C. Palabra recibida que ocasionará una decodificación incorrecta, no detectable.

Para el decodificador, un borrado es un símbolo en una palabra recibida que ha sido etiquetado como "desconfiable" por el demodulador, es un error cuya ubicación se conoce antes que la decodificación comience y su magnitud puede ser o no ser cero. El algoritmo que se presenta a continuación permite corregir con éxito patrones de e errores y f borrados en una palabra de longitud n, tal que 2e+f<d. De aquí en adelante, la relación entre la palabra del código transmitida c(x) y la palabra recibida R(x) estará dada por:

$$R(x) = c(x) + E(x) + F(x) \quad \ldots(4)$$

donde

$$E(x)=\sum_{i=0}^{n-1} E_i x^i \quad y \quad F(x)=\sum_{i=0}^{n-1} F_i x^i \quad \ldots(5)$$

E(x) es el polinomio de errores y F(x) es el polinomio de borrados, donde E_i

y $F_i \in CG(2^m)$.

El primer paso del procedimiento de decodificación es almacenar la palabra recibida R(x) y luego calcular los componentes S_i del síndrome, usando la ecuación:

$$S_i = R(\alpha^i) = \sum_{j=0}^{n-1} R_j \alpha^{ij} \quad ; \quad 1 \leq i \leq 2t \quad \ldots (6)$$

Mediante la ecuación (4), S_i se puede escribir como:

$$S_i = c(\alpha^i) + E(\alpha^i) + F(\alpha^i) \quad \ldots (7)$$

Dado que c(x) es un múltiplo de g(x), cuyas raíces son α^i, para $1 \leq i \leq 2t$, la ecuación (7) se puede expresar como:

$$S_i = E(\alpha^i) + F(\alpha^i) = \sum_{j=0}^{n-1} E_j \alpha^{ij} + \sum_{k=0}^{n-1} F_k \alpha^{ik} \quad ; \quad 1 \leq i \leq 2t \quad \ldots (8)$$

dado que $c(\alpha^i)=0$ para $1 \leq i \leq 2t$.

Sea, Y_j la magnitud del j-ésimo error y $X_j = \alpha^j$ su posición. Y sea, Z_k la magnitud del k-ésimo borrado y $W_k = \alpha^k$ su posición, donde $X_j \neq W_k$. Entonces:

$$E_i = \sum_{j=0}^{n-1} E_j \alpha^{ij} = \sum_{j=1}^{e} Y_j X_j^i \quad ; \quad 1 \leq i \leq 2t \quad \ldots (9)$$

y

$$F_i = \sum_{k=0}^{n-1} F_k \alpha^{ik} = \sum_{k=1}^{f} Z_k W_k^i \quad ; \quad 1 \leq i \leq 2t \quad \ldots (10)$$

De aquí, a partir de (8), (9) y (10), S_i se puede escribir como:

$$S_i = E_i + F_i = \sum_{j=1}^{e} Y_j X_j^i + \sum_{k=1}^{f} Z_k W_k^i \quad ; \quad 1 \leq i \leq 2t \quad \ldots (11)$$

El polinomio localizador de borrados $\tau(x)$, está definido como:

$$\tau(x) = \prod_{k=1}^{f} (1 - W_k x) = \sum_{i=0}^{f} \tau_i x^i = 1 + \sum_{i=1}^{f} \tau_i x^i \quad \ldots (12)$$

Los coeficientes τ_l's son funciones simétricas elementales de W_k. Dado que,

$$\tau[W_k^{-1}]= 0 = 1+ \sum_{l=1}^{f} \tau_l W_k^{-l} \quad ; \quad 1 \leq k \leq f \quad \ldots(13)$$

las raices de $\tau(x)$ son el inverso de las posiciones de los borrados. Una de las caracteristicas principales de este algoritmo es el cálculo de los sindromes de Forney. El polinomio de Forney, T(x), se calcula a partir del polinomio $\tau(x)$ y del polinomio de sindromes

$$S(x)= \sum_{l=1}^{2t} S_l x^l \quad \ldots(14)$$

de la siguiente manera:

$$T(x)= (1+S(x))\tau(x)-1 \text{ módulo } x^{2t+1}= \sum_{l=1}^{2t} T_l x^l \quad \ldots(15)$$

Los coeficientes T_l's son llamados sindromes de Forney. En consecuencia, el segundo y tercer paso del procedimiento de decodificación son respectivamente, el cálculo de τ_l para $1 \leq l \leq f$ y los sindromes de Forney.

El polinomio localizador de errores $\sigma(x)$, está definido como:

$$\sigma(x)= \prod_{j=1}^{e} (1-X_j x)= \sum_{l=0}^{e} \sigma_l x^l= 1+ \sum_{l=1}^{e} \sigma_l x^l \quad \ldots(16)$$

Similarmente, los σ_l's son funciones simétricas elementales de X_j. Dado que,

$$\sigma(X_j^{-1})= 0 =1+ \sum_{l=1}^{e} \sigma_l X_j^{-l} \quad ; \quad 1 \leq j \leq e \quad \ldots(17)$$

las raices de $\sigma(x)$, son el inverso de las posiciones de los errores. Mediante (11), (13), (15) y (17), se puede demostrar que los coeficientes σ_l's satisfacen:

$$T_{1+f+e} + \sum_{l=1}^{e} T_{1+f+e-l}\ \sigma_l = 0 \quad ; \quad 1 \leq l \leq 2t-f-e \quad \ldots(18)$$

De aqui, el cuarto paso del procedimiento de decodificación es calcular σ_l

para $1 \leq l \leq e$, a partir de los síndromes de Forney, ec. (18). Esto se puede realizar mediante el algoritmo iterativo de Berlekamp [15], o el algoritmo de síntesis de registro de corrimiento con realimentación lineal (LFSR) de Massey [3].

Además de obtener los σ_j's, el quinto paso del procedimiento de decodificación consiste en calcular las raíces del polinomio localizador de errores mediante la búsqueda de Chien. Esta consiste simplemente en sustituir en el polinomio $\sigma(x)$, las n posibles posiciones de errores diferentes de cero. Es decir, para $x = X_j^{-1} = \alpha^{-j}$ para $0 \leq j \leq n-1$. Si $\sigma(\alpha^{-j}) = 0$, entonces j representa la posición del error.

Conociendo las posiciones de los errores, el sexto paso es calcular las magnitudes de errores y de borrados, lo cual se hace calculando primero el polinomio evaluador de borrados y de errores, $\Omega(x)$.

$$\Omega(x) = (1+T(x))\sigma(x) \text{ módulo } x^{2t+1} \quad \ldots(19)$$

Enseguida, las magnitudes de errores y de borrados pueden ser calculadas a partir de la fórmula:

$$B_j = \frac{A_j^{(e+f-1)}\Omega(A_j^{-1})}{\prod_{j \neq i} (A_j - A_i)} \quad ; \quad 1 \leq j \leq e+f \quad \ldots(20)$$

donde las A_j's son las posiciones de borrados y de errores. Si A_j es igual a una posición de un error, $A_j = X_j$, entonces B_j es la correspondiente magnitud del error, $B_j = Y_j$. Por otro lado, si A_j es igual a una posición de un borrado, $A_j = W_k$, entonces B_j es la correspondiente magnitud del borrado, $B_j = Z_k$.

La palabra del código corregida se obtiene sustrayendo el polinomio de errores E(x) y el polinomio de borrados F(x) de la palabra recibida R(x).

Se hace notar que si f=0, de (12), $\tau(x)=1$. Entonces a partir de (15), $T_l = S_l$. Y los σ_l's pueden ser calculados por el algoritmo de Berlekamp o el de

Massey a partir de los componentes S_i's del síndrome, en lugar de los síndromes de Forney T_i's. En este caso los coeficientes σ_i's satisfacen el sistema de ecuaciones siguiente:

$$S_{i+e} + \sum_{l=1}^{e} S_{i+e-l}\ \sigma_l = 0 \quad ;\ 1 \leq i \leq 2t-e \quad \ldots(21)$$

Por otro lado si e=0, de (16), $\sigma(x)=1$. Entonces, la ecuación (19) se reduce a:

$$\Omega(x) = (1+T(x)) \text{ módulo } x^{2t}+1 \quad \ldots(22)$$

En las figuras II.2 y II.3 se presentan respectivamente, el diagrama de flujo del algoritmo clásico de la decodificación de códigos R.S. que se acaba de describir y el diagrama de bloques de la simulación correspondiente.

FIG. II.2 ALGORITMO CLASICO DE DECODIFICACION DE CODIGOS R.S.

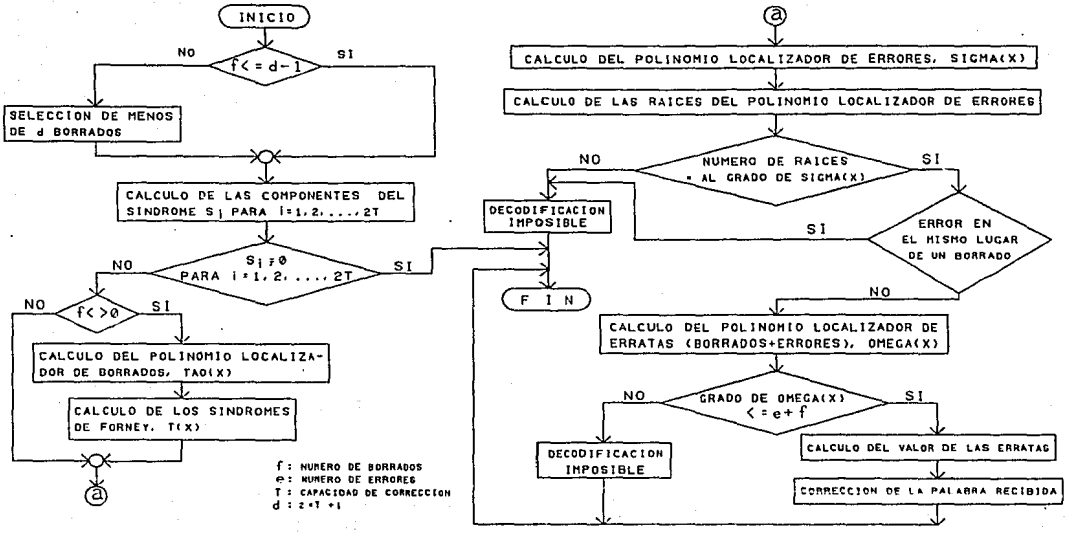
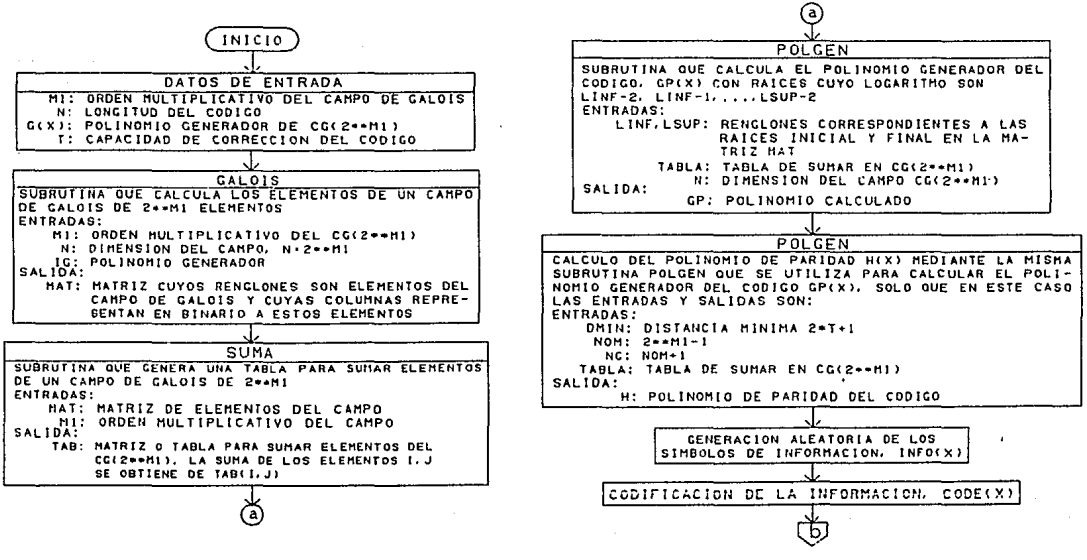
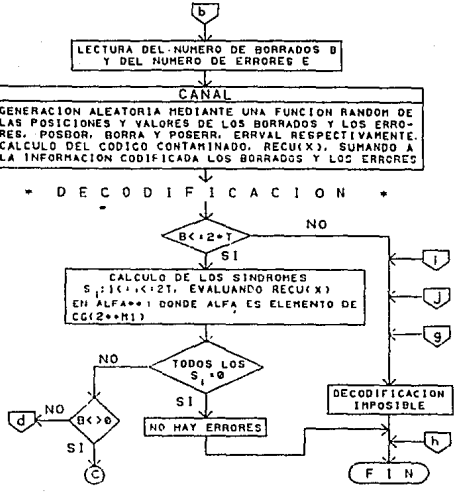
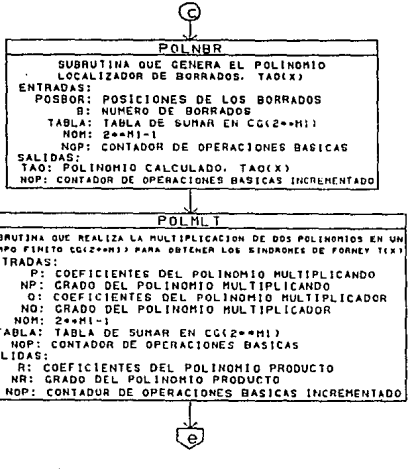
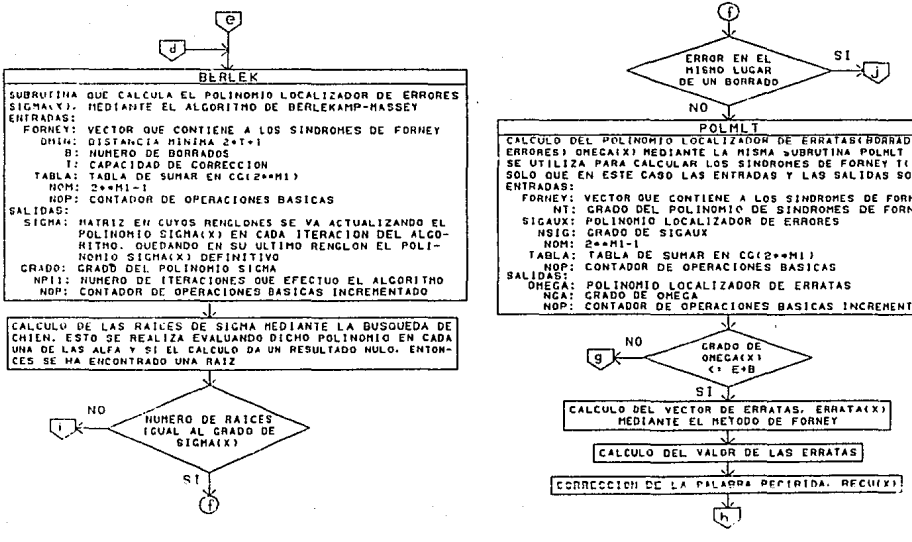


FIG. II.3 DIAGRAMA A BLOQUES DE LA SIMULACION DEL ALGORITMO CLASICO DE DECODIFICACION DE CODIGOS R.S.




b
LECTURA DEL NUMERO DE BORRADOS B
Y DEL NUMERO DE ERRORES E
CANAL
GENERACION ALEATORIA MEDIANTE UNA FUNCION RANDOM DE LAS POSICIONES Y VALORES DE LOS BORRADOS Y LOS ERRORES. POSBOR, BORRA Y POSERR, ERRVAL RESPECTIVAMENTE. CALCULO DEL CODIGO CONTAMINADO, RECU(X), SUMANDO A LA INFORMACION CODIFICADA LOS BORRADOS Y LOS ERRORES
* D E C O D I F I C A C I O N *
B<=2*T
NO
SI
CALCULO DE LOS SINDROMES
S_i ; 1<=i<=2T, EVALUANDO RECU(X)
EN ALFA**i DONDE ALFA ES ELEMENTO DE
GG(2**M1)
TODOS LOS
S_i =0
NO
SI
B<>0
NO
SI
NO HAY ERRORES
DECODIFICACION
IMPOSIBLE
i
J
g
d
h
c
F I N


c
POLNBR
SUBRUTINA QUE GENERA EL POLINOMIO
LOCALIZADOR DE BORRADOS. TAO(X)
ENTRADAS:
POSBOR: POSICIONES DE LOS BORRADOS
B: NUMERO DE BORRADOS
TABLA: TABLA DE SUMAR EN GG(2**M1)
NOM: 2**M1-1
NOP: CONTADOR DE OPERACIONES BASICAS
SALIDAS:
TAO: POLINOMIO CALCULADO, TAO(X)
NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO
POLMLT
SUBRUTINA QUE REALIZA LA MULTIPLICACION DE DOS POLINOMIOS EN UN CAMPO FINITO GG(2**M1) PARA OBTENER LOS SINDROMES DE FORNEY T(X)
ENTRADAS:
P: COEFICIENTES DEL POLINOMIO MULTIPLICANDO
NP: GRADO DEL POLINOMIO MULTIPLICANDO
Q: COEFICIENTES DEL POLINOMIO MULTIPLICADOR
NQ: GRADO DEL POLINOMIO MULTIPLICADOR
NOM: 2**M1-1
TABLA: TABLA DE SUMAR EN GG(2**M1)
NOP: CONTADOR DE OPERACIONES BASICAS
SALIDAS:
R: COEFICIENTES DEL POLINOMIO PRODUCTO
NR: GRADO DEL POLINOMIO PRODUCTO
NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO
e


e
d
BERLEK
SUBRUTINA QUE CALCULA EL POLINOMIO LOCALIZADOR DE ERRORES SIGMA(X), MEDIANTE EL ALGORITMO DE BERLEKAMP-MASSEY
ENTRADAS:
FORNEY: VECTOR QUE CONTIENE A LOS SINDROMES DE FORNEY
DMIN: DISTANCIA MINIMA 2*T+1
B: NUMERO DE BORRADOS
T: CAPACIDAD DE CORRECCION
TABLA: TABLA DE SUMAR EN CG(2**M1)
NOM: 2**M1-1
NOP: CONTADOR DE OPERACIONES BASICAS
SALIDAS:
SIGMA: MATRIZ EN CUYOS RENGLONES SE VA ACTUALIZANDO EL POLINOMIO SIGMA(X) EN CADA ITERACION DEL ALGORITMO. QUEDANDO EN SU ULTIMO RENGLON EL POLINOMIO SIGMA(X) DEFINITIVO
GRADO: GRADO DEL POLINOMIO SIGMA
NPI1: NUMERO DE ITERACIONES QUE EFECTUO EL ALGORITMO
NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO
CALCULO DE LAS RAICES DE SIGMA MEDIANTE LA BUSQUEDA DE CHIEN. ESTO SE REALIZA EVALUANDO DICHO POLINOMIO EN CADA UNA DE LAS ALFA Y SI EL CALCULO DA UN RESULTADO NULO, ENTONCES SE HA ENCONTRADO UNA RAIZ
NO
i
NUMERO DE RAICES IGUAL AL GRADO DE SIGMA(X)
SI
f
f
ERROR EN EL MISMO LUGAR DE UN BORRADO
SI
j
NO
POLMLT
CALCULO DEL POLINOMIO LOCALIZADOR DE ERRATAS(BORRADOS+ERRORES) OMEGA(X) MEDIANTE LA MISMA SUBRUTINA POLMLT QUE SE UTILIZA PARA CALCULAR LOS SINDROMES DE FORNEY T(X), SOLO QUE EN ESTE CASO LAS ENTRADAS Y LAS SALIDAS SON:
ENTRADAS:
FORNEY: VECTOR QUE CONTIENE A LOS SINDROMES DE FORNEY
NT: GRADO DEL POLINOMIO DE SINDROMES DE FORNEY
SIGAUX: POLINOMIO LOCALIZADOR DE ERRORES
NSIG: GRADO DE SIGAUX
NOM: 2**M1-1
TABLA: TABLA DE SUMAR EN CG(2**M1)
NOP: CONTADOR DE OPERACIONES BASICAS
SALIDAS:
OMEGA: POLINOMIO LOCALIZADOR DE ERRATAS
NGA: GRADO DE OMEGA
NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO
NO
g
GRADO DE OMEGA(X) <= E*B
SI
CALCULO DEL VECTOR DE ERRATAS, ERRATA(X) MEDIANTE EL METODO DE FORNEY
CALCULO DEL VALOR DE LAS ERRATAS
CORRECCION DE LA PALABRA RECIBIDA, RECU(X)
h

FUNCIONES UTILIZADAS EN LA SIMULACION

MULT

```
FUNCION ENTERA QUE REALIZA LA MULTIPLICACION
  ENTRE DOS ELEMENTOS DE UN CAMPO DE GALOIS
ENTRADAS:
   I, J: ELEMENTOS A MULTIPLICAR
     N1: 2**M1-1
SALIDA:
    MULT: RESULTADO DE LA MULTIPLICACION
```

IEXP

```
FUNCION ENTERA QUE EFECTUA LA OPERACION DE ELEVAR
A UNA POTENCIA ENTERA J, UN ELEMENTO DEL CAMPO DE
GALOIS CUYO LOGARITMO ES I-2
ENTRADAS:
    I: ELEMENTO A EXPONENCIAR
    J: POTENCIA DESEADA
   N1: 2**M1-1
SALIDA:
   IEXP: ELEMENTO EXPONENCIADO
```

II.2 ALGORITMO MODIFICADO DE DECODIFICACION PARA CODIGOS R.S.

La diferencia más importante de este algoritmo con respecto al anterior, consiste en la alimentación directa del polinomio localizador de borrados $\tau(x)$ en el algoritmo de Berlekamp-Massey [3]. Esto nos permite obtener el polinomio combinado localizador de borrados y de errores $P(x)$, por medio del cual se evita el cálculo de los sindromes de Forney, necesarios en el caso anterior para poder obtener el polinomio localizador de errores $\sigma(x)$.

Este algoritmo empieza una vez que se tiene almacenada la palabra recibida $R(x)$, enseguida se procede a calcular los coeficientes S_i del polinomio de sindromes $S(x)$ y también a calcular el polinomio localizador de borrados $\tau(x)$, por medio de las ecuaciones (6) y (12) (punto II.1) respectivamente. Luego se procede a calcular el polinomio localizador de borrados y de errores $P(x)$, el cual es inicializado con el polinomio localizador de borrados $\tau(x)$, esto es $P(x)=\tau(x)$. Posteriormente $P(x)$ se obtiene por medio del algoritmo de Berlekamp-Massey [3], con las ecuaciones recursivas:

$$D_r = \sum_{j=0}^{n-1} P_j^{(r-1)} S_{r-j}$$

$$\tau^{(r)}(x) = (1-\delta_r)x\tau^{(r-1)}(x) + \delta_r D_r^{-1} P^{(r-1)}(x) \quad \ldots(23)$$

$$P^{(r)}(x) = P^{(r-1)}(x) - D_r x\tau^{(r-1)}(x)$$

D_r se conoce como la r-ésima discrepancia, P_j es el j-ésimo coeficiente de $P(x)$ y S_{r-j} representa los coeficientes de $S(x)$. El valor de δ_r, definido como $\delta_r = 2L-r+f-1$, depende del número de borrados f, del grado del polinomio $P(x)$ y de la r-ésima iteración del algoritmo. El valor de L se incrementa en uno por cada borrado y posteriormente se incrementa de acuerdo al procedimiento del algoritmo. El indice r es un contador, que cuando excede a

2t, termina el algoritmo y en este punto se obtiene el polinomio localizador de borrados y de errores P(x).

A continuación, se procede a calcular las raices de P(x) por medio de la búsqueda de Chien, descrita en el punto II.1. Conocidas las posiciones de los borrados y de los errores, se calculan las magnitudes de los mismos. Para esto se obtiene primero el polinomio evaluador de borrados y de errores Ω(x).

$$\Omega(x) = (1+S(x))P(x) \text{ módulo } x^{d} \quad \ldots(24)$$

Enseguida, las magnitudes de los borrados y de los errores pueden ser calculadas con la fórmula:

$$B_j = \frac{A_j^{(e+f-1)}\Omega(A_j^{-1})}{P'(A_j)} \quad ; \quad 1 \leq j \leq e+f \quad \ldots(25)$$

Donde como en el caso anterior, las A_j's son las posiciones de los borrados y de los errores. Si A_j corresponde a la posición de un error, B_j es la correspondiente magnitud. De la misma manera, si A_j corresponde a la posición de un borrado, B_j es su magnitud.

Otra diferencia entre este algoritmo "modificado" y el algoritmo "clásico", está en el cálculo de las magnitudes de los borrados y de los errores. A partir de las ecuaciones (20) y (25), se puede observar que el divisor de la ecuación (20) se sustituye por la derivada del polinomio P(x) evaluada en A_j, para formar la ecuación (25). La derivada del polinomio P(x) se calcula de la siguiente manera:

Haciendo

$$P(x) = \sum_{j=0}^{e+f} P_j x^j \quad y \quad \bar{P}(x) = \sum_{j=0}^{e+f} P_{e+f-j} x^j$$

donde P(x) y $\bar{P}(x)$ son idénticos, excepto que este último tiene sus coeficientes con orden invertido con respecto a P(x).

La derivada de P(x) se obtiene por:

$$P'(x) = \frac{d\,\bar{P}(x)}{dx}$$

Sin embargo, los divisores de las ecuaciones (20) y (25) resultan ser equivalentes, esto se demuestra a continuación:

Para llegar a la igualdad de estos divisores es necesario redefinir el polinomio localizador de borrados $\tau(x)$ y el polinomio localizador de errores $\sigma(x)$ como:

$$\tau(x)=\prod_{i=1}^{f}(X_i+x) \quad y \quad \sigma(x)=\prod_{i=1}^{e}(X_i+x)$$

así el polinomio localizador de borrados y de errores P(x) quedará definido como:

$$P(x)=\tau(x)\sigma(x)=\prod_{i=1}^{e+f}(X_i+x)$$

ahora, si se desarrolla P(x) y se deriva, se observa que la derivada de P(x) se puede expresar como:

$$P'(x)=\sum_{j=1}^{e+f}\prod_{i\neq j}(X_i+x)$$

evaluando P'(x) en un punto X_k, tenemos:

$$P'(X_k)=\sum_{j=1}^{e+f}\prod_{i\neq j}(X_i+X_k) \quad \text{para } 1\leq k\leq e+f$$

desarrollando y considerando las operaciones módulo dos, quedará:

$$P'(X_k)=\prod_{i\neq k}(X_i+X_k) \quad \text{para } 1\leq k\leq e+f$$

que es equivalente al denominador de (20).

Por último, la palabra del código corregida se obtiene restando el polinomio de errores E(x) y el polinomio de borrados F(x) (ver punto II.1) de la palabra recibida R(x).

En las figuras II.4, II.5 y II.6 siguientes se presentan respectivamente, el diagrama de flujo del algoritmo de Berlekamp-Massey, el diagrama de flujo

del algoritmo modificado de decodificación de códigos R.S. y el diagrama de bloques de la simulación correspondiente.

FIG. II.4 DIAGRAMA DE FLUJO DEL ALGORITMO DE BERLEKAMP-MASSEY

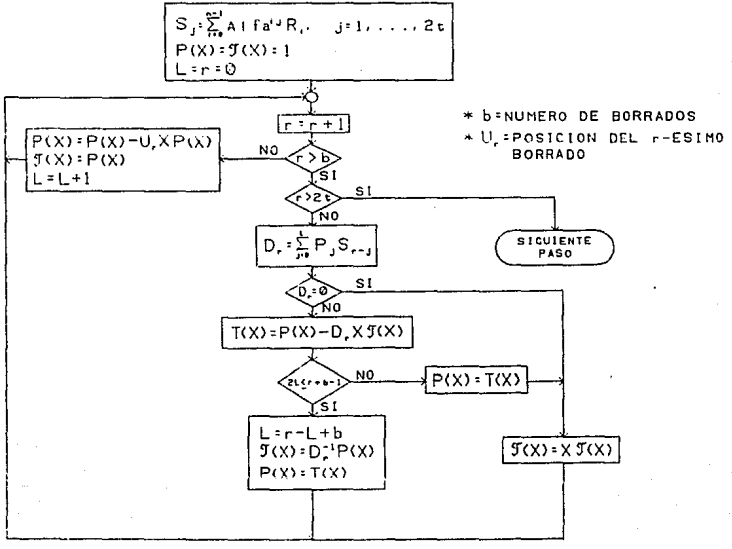


FIG. II.5 DIAGRAMA DE FLUJO DEL ALGORITMO MODIFICADO DE DECODIFICACION PARA CODIGOS R.S.

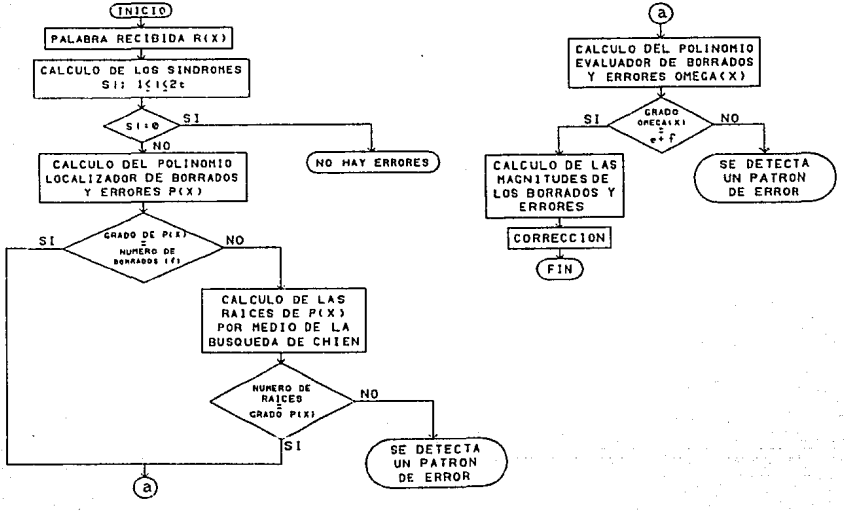
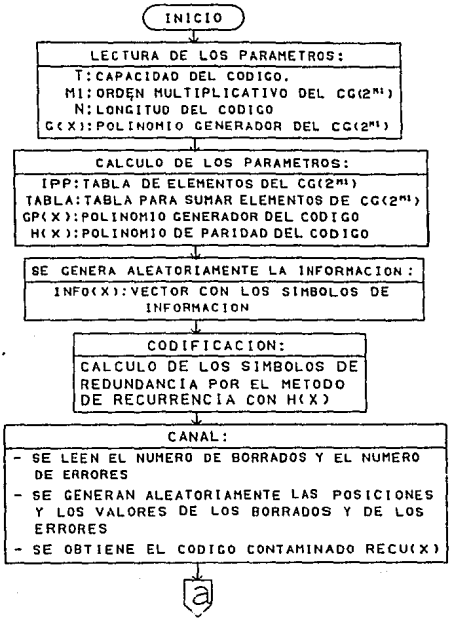
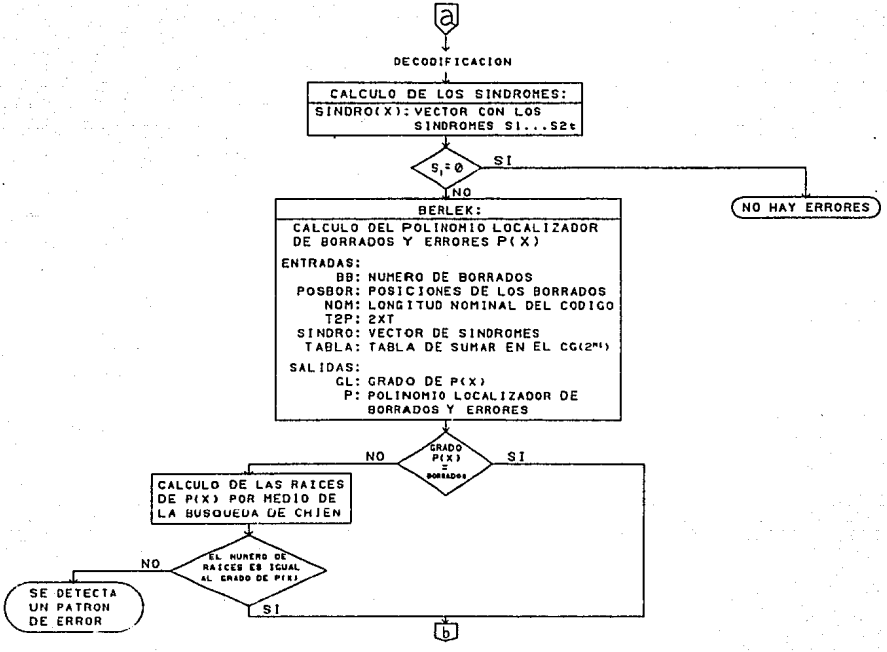
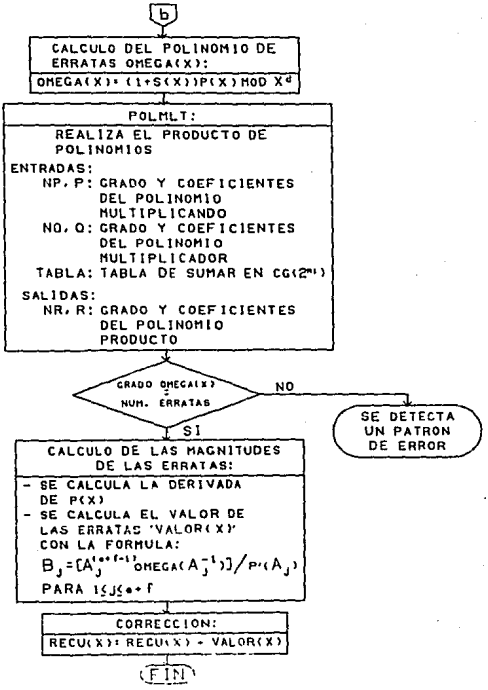


FIG. II.6 SIMULACION DEL ALGORITMO MODIFICADO DE DECODIFICACION PARA CODIGOS R.S.




a
DECODIFICACION
CALCULO DE LOS SINDROMES:
SINDRO(X): VECTOR CON LOS SINDROMES S1...S2t
Si = 0
SI
NO
NO HAY ERRORES
BERLEK:
CALCULO DEL POLINOMIO LOCALIZADOR DE BORRADOS Y ERRORES P(X)
ENTRADAS:
BB: NUMERO DE BORRADOS
POSBOR: POSICIONES DE LOS BORRADOS
NOM: LONGITUD NOMINAL DEL CODIGO
T2P: 2XT
SINDRO: VECTOR DE SINDROMES
TABLA: TABLA DE SUMAR EN EL CG(2^M)
SALIDAS:
GL: GRADO DE P(X)
P: POLINOMIO LOCALIZADOR DE BORRADOS Y ERRORES
GRADO P(X) = BORRADOS
NO
SI
CALCULO DE LAS RAICES DE P(X) POR MEDIO DE LA BUSQUEDA DE CHIEN
EL NUMERO DE RAICES ES IGUAL AL GRADO DE P(X)
NO
SI
SE DETECTA UN PATRON DE ERROR
b


b
CALCULO DEL POLINOMIO DE ERRATAS OMEGA(X):
OMEGA(X)= (1+S(X))P(X) MOD X^d
POLMLT:
REALIZA EL PRODUCTO DE POLINOMIOS
ENTRADAS:
NP, P: GRADO Y COEFICIENTES DEL POLINOMIO MULTIPLICANDO
NQ, Q: GRADO Y COEFICIENTES DEL POLINOMIO MULTIPLICADOR
TABLA: TABLA DE SUMAR EN CG(2^M)
SALIDAS:
NR, R: GRADO Y COEFICIENTES DEL POLINOMIO PRODUCTO
GRADO OMEGA(X) = NUM. ERRATAS
NO
SE DETECTA UN PATRON DE ERROR
SI
CALCULO DE LAS MAGNITUDES DE LAS ERRATAS:
- SE CALCULA LA DERIVADA DE P(X)
- SE CALCULA EL VALOR DE LAS ERRATAS 'VALOR(X)' CON LA FORMULA:
B_J=[A_J^(1-f) OMEGA(A_J^-1)]/P'(A_J)
PARA 1≤J≤n-f
CORRECCION:
RECU(X)= RECU(X) - VALOR(X)
FIN

II.3 ALGORITMO DE DECODIFICACION PARA CODIGOS R.S. MEDIANTE EL USO DE UNA TRANSFORMADA RAPIDA.

Se ha desarrollado un algoritmo de decodificación simplificado, para corregir borrados y errores con códigos R.S. sobre el campo finito $CG(p^m)$, donde p es un número primo y m es un número entero.

Hay que señalar que este algoritmo se usa aquí para corregir patrones de e errores y f borrados de palabras del código RS(255,223), donde $2e+f<33$ y donde los símbolos pertenecen al campo finito $CG(2^8)$. Se utiliza este código en particular ya que es un estándar recomendado por el CCSDS (Consultative Committee for Space Data Systems) para la codificación en canales de telemetría [23]. Para su decodificación, se ha desarrollado una transformada rápida que nos permite calcular los vectores de errores y de borrados de las palabras del código transmitidas.

Se definen los siguientes cinco vectores:

$$(c_0, c_1, \ldots, c_{254}) = c, \text{ vector del código.}$$

$$(r_0, r_1, \ldots, r_{254}) = r, \text{ vector recibido.}$$

$$(\mu_0, \mu_1, \ldots, \mu_{254}) = \mu, \text{ vector de borrados.}$$

$$(\hat{e}_0, \hat{e}_1, \ldots, \hat{e}_{254}) = \hat{e}, \text{ vector de errores.}$$

$$(\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_{254}) = \hat{u}, \text{ vector de erratas.}$$

Estos vectores se relacionan como sigue: $r = c+\mu+\hat{e}$ y $\hat{u} = \hat{e}+\mu$.

Supóngase que en el vector recibido r de 255 símbolos, hay e errores y f borrados y asúmase que $2e+f<33$. Entonces, el procedimiento de decodificación consiste en los siguientes cinco pasos:

(1) Cálculo de los síndromes S_k $(1 \leq k \leq 32)$ del vector recibido $(r_0, r_1, \ldots, r_{254})$.

$$S_k = \sum_{i=0}^{254} r_i \alpha^{ik} \quad \text{para } k = 1,2,\ldots,32 \quad \ldots(26)$$

donde α es un elemento del campo finito $CG(2^8)$. Si $S_k = 0$ para $1 \leq k \leq 32$, entonces r es una palabra del código y se puede terminar la decodificación. Si esto no se cumple, entonces:

(ii) Cálculo de los coeficientes del polinomio localizador de borrados, τ_j para $j=0,1,2,\ldots,f$, a partir de:

$$\tau(x) = \prod_{j=1}^{f} (x - Z_j) = \sum_{j=0}^{f} (-1)^j \tau_j x^{f-j} \quad \ldots(27)$$

donde f es el número de borrados en el vector recibido y Z_j $(1 \leq j \leq f)$ son las posiciones conocidas de los borrados.

(iii) En seguida, se calcula el polinomio combinado localizador de borrados y de errores.

$$P(x) = \sigma(x)\tau(x) = \sum_{k=0}^{f+e} (-1)^k P_k x^{f+e-k} \quad \ldots(28)$$

donde $\sigma(x)$ es el polinomio localizador de errores. El polinomio P(x) se obtiene con el algoritmo de Berlekamp-Massey [3], descrito en el punto II.2 de este capítulo.

(iv) Después de esto, se calcula el resto de los síndromes con la ecuación:

$$S_i = -\sum_{k=1}^{f+e} (-1)^k P_k S_{i-k} \quad \text{para } i > 32 \quad \ldots(29)$$

se hace notar que $S_{255} = S_0$.

(v) Cálculo de la transformada inversa del vector de síndromes $(S_0,\ldots,S_{254})$, para obtener el vector de errores y de borrados Q:

$$Q_i = \hat{e}_i + \mu_i = \sum_{k=0}^{254} S_k \alpha^{-ik} \quad \text{para } i = 0,1,2,\ldots,254 \quad \ldots(30)$$

Finalmente, se sustrae el vector de errores y de borrados $\hat{u}$ del vector recibido r para corregirlo.

En este algoritmo la búsqueda de Chien se elimina. Esta búsqueda es reemplazada por el cálculo de una transformada de 255 puntos usando una técnica similar a la de la transformada rápida de Fourier (FFT). El resultado es un decodificador más simple y más rápido.

II.3.1 TECNICA RAPIDA PARA EL CALCULO DE LAS MAGNITUDES DE ERRORES Y DE BORRADOS.

La idea es usar una técnica con transformada rápida sobre el campo finito $CG(2^8)$. Esos conceptos son usados para calcular eficientemente la expresión:

$$A_j = \sum_{i=0}^{254} a_i \alpha^{ij} \quad \text{para } 0 \leq j \leq 254 \quad \ldots(31)$$

donde α es un elemento del campo $CG(2^8)$. Para comenzar, si $n = n_1 n_2 \ldots n_r$ donde n_k para $(1 \leq k \leq r)$ es un número primo, entonces un entero J puede representarse por una r-tuple $(J_1, J_2, \ldots, J_r)$, donde $J_k = J$ módulo n_k con $1 \leq k \leq r$. Ahora, para nuestro caso $n = 255 = 3 \times 5 \times 17$, por lo que $n_1 = 3$, $n_2 = 5$ y $n_3 = 17$. Entonces los términos a_i de la ecuación (31) pueden representarse por la matriz $a(i_1, i_2, i_3)$, donde $i_1 = i$ módulo 3, $i_2 = i$ módulo 5, $i_3 = i$ módulo 17.

La ecuación (31) puede ser descompuesta en las siguientes tres etapas:

Etapa 1.

$$A^1_{(i1,i2,j3)} = \sum_{i3=0}^{3-1} a_{(i1,i2,i3)} \alpha^{i3j3} \quad \text{para} \quad \begin{matrix} 0 \leq i1 \leq 16 \\ 0 \leq i2 \leq 4 \\ 0 \leq j3 \leq 2 \end{matrix}$$

Etapa 2.

$$A^{2}_{(i1,j2,j3)} = \sum_{i2=0}^{5-1} A^{1}_{(i1,i2,j3)} \alpha_{2}^{i2j2} \quad \text{para} \quad \begin{matrix} 0 \le i1 \le 16 \\ 0 \le j2 \le 4 \\ 0 \le j3 \le 2 \end{matrix} \quad \ldots(32)$$

Etapa 3.

$$S_{(j1,j2,j3)} = \sum_{i1=0}^{17-1} A^{2}_{(i1,j2,j3)} \alpha_{1}^{i1j1} \quad \text{para} \quad \begin{matrix} 0 \le j1 \le 16 \\ 0 \le j2 \le 4 \\ 0 \le j3 \le 2 \end{matrix} .$$

donde $\alpha_3 = \alpha^{85}$, $\alpha_2 = \alpha^{51}$ y $\alpha_1 = \alpha^{120}$. En las ecuaciones (32), la etapa 1, la etapa 2 y la etapa 3 son transformadas de 3, 5 y 17 puntos respectivamente. Los algoritmos detallados para el cálculo de las transformadas de 3, 5 y 17 puntos sobre el campo finito $CG(2^8)$ se encuentran en el apéndice. De esos algoritmos se tiene que el número de multiplicaciones necesarias para calcular una transformada de 3, 5 y 17 puntos es 1, 5 y 53 respectivamente. En forma similar, el número de sumas que se necesitan para calcular una transformada de 3, 5 y 17 puntos es 5, 17 y 176 respectivamente. Entonces, el número total de multiplicaciones y sumas que se necesitan para calcular A_j para $0 \le j \le 254$ es 17x5x1 + 17x5x3 + 53x5x3= 1135 y 17x5x5 + 17x3x17 + 176x5x3= 3932, respectivamente.

En las figuras II.7 y II.8 se presentan, el algoritmo de decodificación para códigos R.S. en el dominio de la frecuencia y el diagrama de bloques de la simulación del algoritmo de decodificación anterior.

FIG. II.7 ALGORITMO DE DECODIFICACION PARA CODIGOS R.S. EN EL DOMINIO DE LA FRECUENCIA

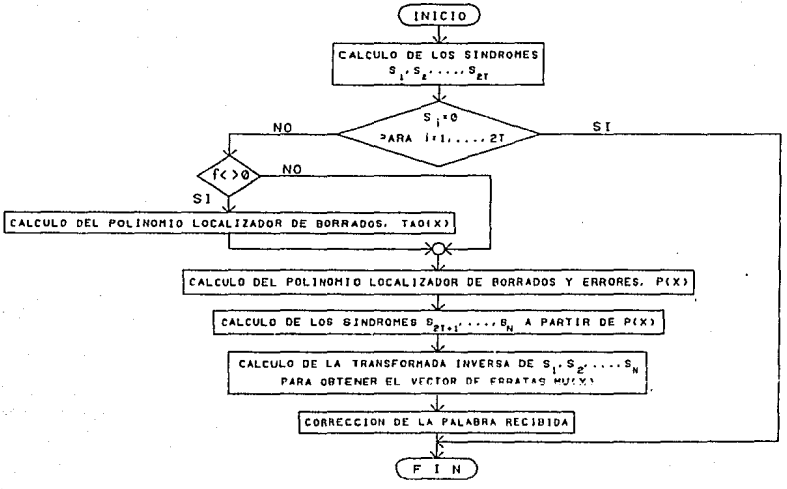
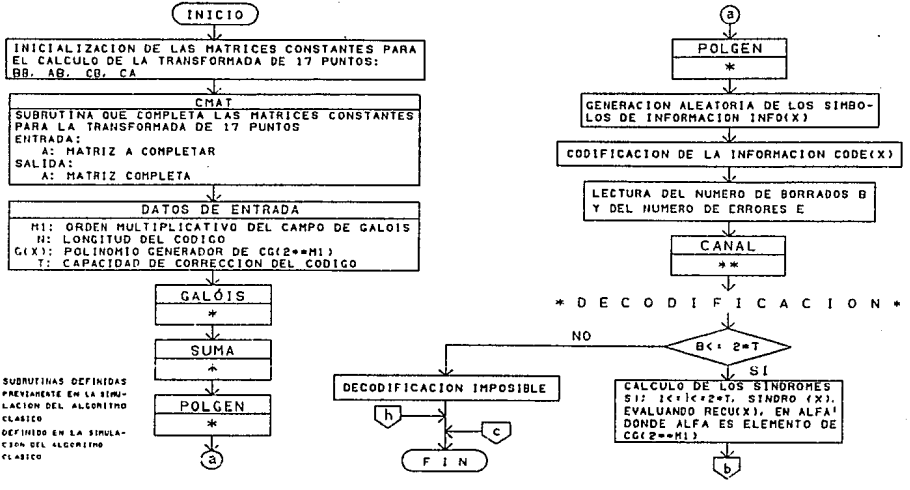
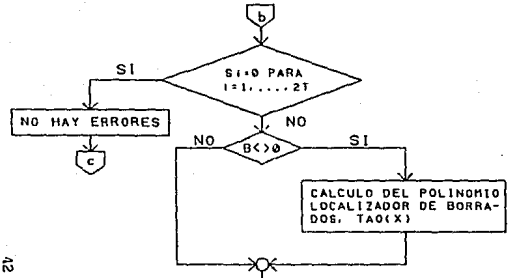


FIG. II.8 DIAGRAMA A BLOQUES DE LA SIMULACION DEL ALGORITMO DE DECODIFICACION EN EL DOMINIO DE LA FRECUENCIA DE CODIGOS R.S.





BERLEK

SUBRUTINA QUE CALCULA EL POLINOMIO LOCALIZADOR DE BORRADOS Y ERRORES, P(X), SIN CALCULAR LOS SINDROMES DE FORNEY

ENTRADAS:

- BB: NUMERO DE BORRADOS
- POSBOR: POSICION DE LOS BORRADOS
- NOM: LONGITUD NOMINAL DEL CODIGO 2**M1-1
- T2P: 2*T
- SINDRO: VECTOR DE SINDROMES
- TABLA: TABLA DE SUMAR EN CG(2**M1)
- NOP: CONTADOR DE OPERACIONES BASICAS

SALIDAS:

- GL: GRADO DE P(X)
- P: POLINOMIO LOCALIZADOR DE BORRADOS Y ERRORES
- NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO

d

d

SINDCOMP

SUBRUTINA QUE CALCULA LOS SINDROMES S(2*T+1)...S(N)

ENTRADAS:

- T2P: 2*T
- N: LONGITUD DEL CODIGO
- GL: GRADO DE P(X)
- P: POLINOMIO LOCALIZADOR DE BORRADOS Y ERRORES
- TAB: TABLA DE SUMAR EN CG(2**M1)
- NOM: LONGITUD NOMINAL DEL CODIGO 2**M1-1
- SIND: VECTOR QUE CONTIENE LOS 2*T PRIMEROS SINDROMES
- NOP: CONTADOR DE OPERACIONES BASICAS

SALIDAS:

- SIND: VECTOR RESULTANTE CON LOS N SINDROMES
- NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO

TRANSF

SUBRUTINA QUE CALCULA EL VECTOR DE ERRATAS(BORRADOS+ERRORES) MU(X) POR EL METODO DE LA TRANSFORMADA RAPIDA

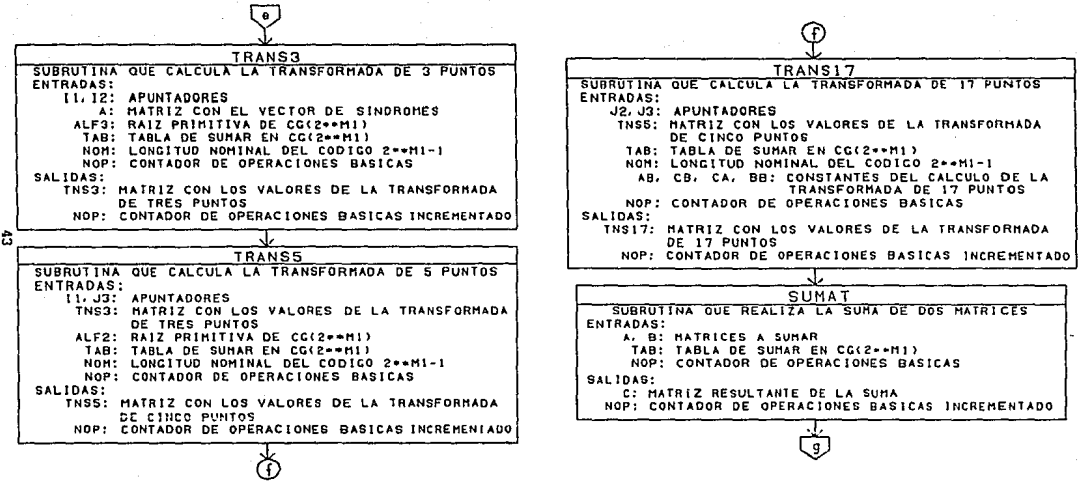
ENTRADAS:

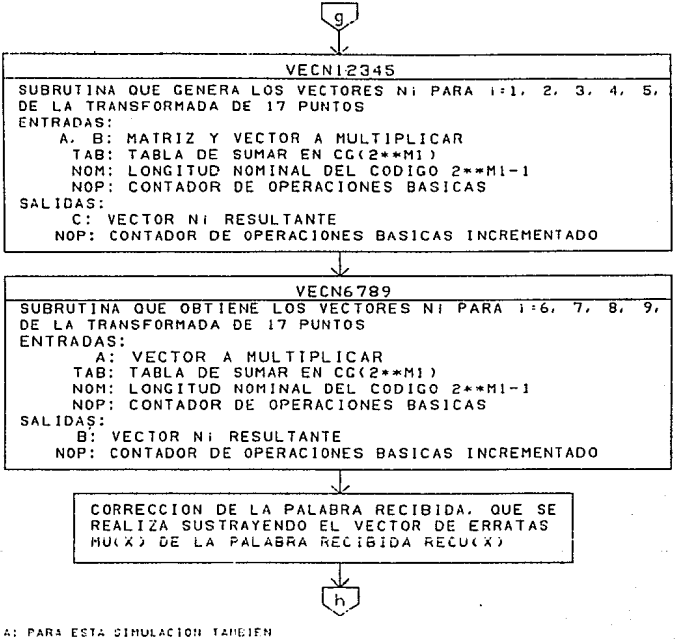
- VECMAT: VECTOR DE SINDROMES
- AB, CB, CA, BB: CONSTANTES PARA EL CALCULO DE LA TRANSFORMADA DE 17 PUNTOS
- L: CONSTANTES PARA CALCULAR TRANSFORMADAS O ANTITRANSFORMADAS SEGUN SEA SU VALOR
- TAB: TABLA DE SUMAR EN CG(2**M1)
- NOM: LONGITUD NOMINAL DEL CODIGO 2**M1-1
- NOP: CONTADOR DE OPERACIONES BASICAS

SALIDAS:

- MATVEC: VECTOR DE ERRATAS, MU(X)
- NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO

e


e
TRANS3
SUBRUTINA QUE CALCULA LA TRANSFORMADA DE 3 PUNTOS
ENTRADAS:
I1, I2: APUNTADORES
A: MATRIZ CON EL VECTOR DE SINDROMES
ALF3: RAIZ PRIMITIVA DE CG(2**M1)
TAB: TABLA DE SUMAR EN CG(2**M1)
NOM: LONGITUD NOMINAL DEL CODIGO 2**M1-1
NOP: CONTADOR DE OPERACIONES BASICAS
SALIDAS:
TNS3: MATRIZ CON LOS VALORES DE LA TRANSFORMADA DE TRES PUNTOS
NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO
TRANS5
SUBRUTINA QUE CALCULA LA TRANSFORMADA DE 5 PUNTOS
ENTRADAS:
I1, J3: APUNTADORES
TNS3: MATRIZ CON LOS VALORES DE LA TRANSFORMADA DE TRES PUNTOS
ALF2: RAIZ PRIMITIVA DE CG(2**M1)
TAB: TABLA DE SUMAR EN CG(2**M1)
NOM: LONGITUD NOMINAL DEL CODIGO 2**M1-1
NOP: CONTADOR DE OPERACIONES BASICAS
SALIDAS:
TNS5: MATRIZ CON LOS VALORES DE LA TRANSFORMADA DE CINCO PUNTOS
NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO
f
f
TRANS17
SUBRUTINA QUE CALCULA LA TRANSFORMADA DE 17 PUNTOS
ENTRADAS:
J2, J3: APUNTADORES
TNS5: MATRIZ CON LOS VALORES DE LA TRANSFORMADA DE CINCO PUNTOS
TAB: TABLA DE SUMAR EN CG(2**M1)
NOM: LONGITUD NOMINAL DEL CODIGO 2**M1-1
AB, CB, CA, BB: CONSTANTES DEL CALCULO DE LA TRANSFORMADA DE 17 PUNTOS
NOP: CONTADOR DE OPERACIONES BASICAS
SALIDAS:
TNS17: MATRIZ CON LOS VALORES DE LA TRANSFORMADA DE 17 PUNTOS
NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO
SUMAT
SUBRUTINA QUE REALIZA LA SUMA DE DOS MATRICES
ENTRADAS:
A, B: MATRICES A SUMAR
TAB: TABLA DE SUMAR EN CG(2**M1)
NOP: CONTADOR DE OPERACIONES BASICAS
SALIDAS:
C: MATRIZ RESULTANTE DE LA SUMA
NOP: CONTADOR DE OPERACIONES BASICAS INCREMENTADO
g



* NOTA: PARA ESTA SIMULACION TAMBIEN SE UTILIZARON LAS FUNCIONES MULTIPLICACION DEFINIDAS EN EL ALGORITMO [illegible]

Capitulo III. Resultados de la Simulacion de los Algoritmos de Decodificacion

En el presente capitulo hablaremos acerca de la simulación realizada para los tres algoritmos de decodificación y el objetivo de ésta.

Para los tres algoritmos descritos en el capítulo anterior, se hizo el conteo de los ciclos de reloj (operaciones) que lleva ejecutar cada una de las instrucciones de las etapas de decodificación en un microprocesador 68000 de Motorola.

El conteo de operaciones mencionado se hizo con el fin de comparar la rapidez de decodificación entre los tres algoritmos lo cual permitirá seleccionar el método de decodificación óptimo para su posterior microprogramación y diseño con una arquitectura en base a su paralelización. La decodificación óptima implica que el algoritmo más rápido es el mmejor.

A continuación se mencionan las etapas de decodificación que se contabilizaron para cada uno de los tres algoritmos.

En el algoritmo "Clásico" se contabilizaron las siguientes cuatro etapas:

a) Cálculo del Polinomio Localizador de Borrados, Síndromes de Forney y Polinomio Localizador de Errores.

b) Cálculo de las raíces del Polinomio Localizador de Errores mediante la búsqueda de Chien.

c) Cálculo del valor de las Erratas.

d) Decodificación Total.

Para el algoritmo "Modificado" se contabilizaron las cuatro etapas siguientes:

a) Cálculo del Polinomio Localizador de Borrados y Errores.

b) Cálculo de las raíces del Polinomio Localizador de Borrados y Errores

mediante la búsqueda de Chien.

c) Cálculo del valor de las Erratas.

d) Decodificación Total.

La comparación entre estos dos algoritmos se hizo etapa por etapa.

Habiendo observado que el cálculo de las raices es la etapa más lenta, se recurrió a un algoritmo que trabaja en el dominio de la frecuencia el cual no requiere de esta etapa y se aprovecharon las partes más rápidas de los otros dos algoritmos para tener un algoritmo de decodificación más rápido.

Para este algoritmo se contabilizó sólo el número de operaciones en la decodificación total y se comparó con la contabilización respectiva hecha para los otros dos algoritmos.

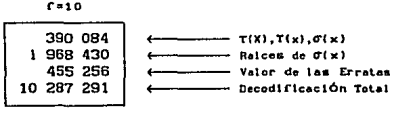
III.1 TABLAS DE LOS ALGORITMOS DE DECODIFICACION.

Con la simulación de los tres algoritmos se generaron tres tablas (una por cada uno), las cuales contienen el número de operaciones para cada una de las etapas contabilizadas.

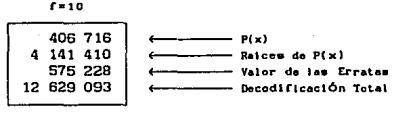
Una vez que se comprobó su rapidez, el algoritmo que trabaja en el dominio de la frecuencia se tradujo a lenguaje ensamblador de la VAX 11-730 debido a que la memoria de la computadora VME/10 de Motorola (que cuenta con un microprocesador 68000) no era suficiente. Luego, se obtuvo el tiempo en segundos que tarda cada subrutina en ejecutarse, con el fin de tener una idea de la rapidez de cada etapa y poder compararlas entre sí. Con estos tiempos se elaboró una cuarta tabla.

El significado de los valores contenidos en cada celda de cada una de las tablas es el siguiente:

Para el algoritmo clásico consideremos como ejemplo la lectura de la celda de la tabla correspondiente al error 8 y borrado 10.

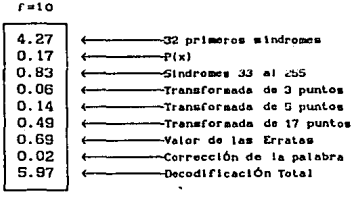


De igual manera, para el algoritmo modificado considerando el mismo ejemplo, se tiene:



Para el algoritmo en el dominio de la frecuencia el único valor contenido en cada celda de la tabla corresponde a la decodificación total.

Y finalmente, para la tabla de tiempos en segundos de este último algoritmo, considerando el mismo ejemplo de error 8 y borrado 10, tenemos:



A continuación se muestran las tablas de los tres algoritmos mencionados y la correspondiente al algoritmo en el dominio de la frecuencia pero en segundos.

TABLA DEL ALGORITMO CLASICO DE DECODIFICACION PARA CODIGOS R.S.

i\ ii je\	0	1	2	3	4	5	6	7	8
	0	65 412	81 046	100 420	118 810	139 716	159 742	183 180	205 342
	0	230 030	230 030	230 030	230 030	230 030	230 030	230 030	230 030
0	0	2 002	6 744	14 226	24 448	37 410	53 112	71 554	92 736
	7 159 164	7 488 976	7 539 852	7 556 312	7 614 924	7 657 072	7 691 492	7 745 696	7 763 188
	47 716	82 872	97 974	117 248	134 506	155 912	175 306	198 744	220 274
	447 330	447 330	447 330	447 330	447 330	447 330	447 330	447 330	447 330
1	2 002	6 744	14 226	24 448	37 410	53 112	71 554	92 736	116 658
	7 716 356	7 752 432	7 773 639	7 838 308	7 867 251	7 880 342	7 935 991	7 987 524	8 072 595
	65 638	109 794	115 164	134 538	151 164	172 570	191 332	214 770	235 668
	664 630	664 630	664 630	664 630	664 630	664 630	664 630	664 630	664 630
2	6 744	14 226	24 448	37 410	53 112	71 554	92 736	116 658	143 320
	7 991 332	8 041 811	8 065 155	8 114 419	8 127 323	8 206 819	8 222 755	8 291 627	8 360 659
	84 380	119 536	133 274	152 648	168 642	190 048	208 178	231 616	251 882
	881 930	881 930	881 930	881 930	881 930	881 930	881 930	881 930	881 930
3	14 226	24 448	37 410	53 112	71 554	92 736	116 658	143 320	172 722
	8 236 089	8 298 396	8 323 849	8 394 668	8 409 131	8 441 412	8 504 967	8 549 304	8 647 739
	103 942	139 538	152 204	171 978	186 940	208 706	255 844	249 282	268 916
	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230
4	24 448	37 410	53 112	71 554	92 736	116 658	143 320	172 722	204 864
	8 525 244	8 547 260	8 596 103	8 650 156	8 737 099	8 752 631	8 795 183	8 868 964	8 929 291
	124 324	159 880	171 954	191 688	206 058	227 784	244 350	268 048	286 770
	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530
5	37 410	53 112	71 554	92 736	116 658	143 320	172 722	204 864	239 746
	8 804 356	8 863 544	8 888 120	8 955 321	8 973 807	9 057,920	9 107 255	9 170 925	9 257 122
	145 526	180 682	192 524	211 698	225 998	247 402	264 196	287 074	305 884
	1 533 830	1 533 830	1 533 830	1 533 830	1 533 830	1 533 850	1 533 830	1 533 830	1 533 830
6	53 112	71 554	92 736	116 658	143 320	172 722	204 864	239 746	277 368
	9 068 082	9 121 325	9 180 483	9 256 283	9 280 459	9 286 284	9 360 225	9 466 387	9 516 556
	168 148	202 704	214 434	233 288	247 194	268 160	284 122	307 200	325 218
	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130
7	71 554	92 736	116 658	143 320	172 722	204 864	239 746	277 368	317 730
	9 360 228	9 433 044	9 476 567	9 511 907	9 522 419	9 623 441	9 682 417	9 761 024	9 831 229
	189 792	225 546	236 724	255 498	268 772	289 738	305 028	328 146	345 452
	1 968 430	1 968 430	1 968 430	1 968 430	1 968 430	1 968 430	1 968 430	1 968 430	1 968 430
8	92 736	116 658	143 320	172 722	204 864	239 746	277 368	317 730	360 832
	9 602 075	9 720 965	9 748 687	9 814 031	9 871 891	9 913 079	9 957 723	10 008 799	10 127 531

n\	9	10	11	12	13	14	15	16	17
	230 812	255 110	282 612	309 046	338 580	367 150	398 716	429 422	463 020
	230 030	230 030	230 030	230 030	230 030	230 030	230 030	230 030	230 030
0	116 658	145 320	172 722	204 864	239 746	277 368	317 730	360 832	406 674
	7 840 360	7 903 644	7 972 872	8 025 596	8 097 792	8 176 308	8 269 648	8 323 972	8 433 912
	245 744	269 610	296 912	322 714	352 248	380 186	411 752	441 826	475 424
	447 330	447 330	447 330	447 330	447 330	447 330	447 330	447 330	447 330
1	145 320	172 722	204 864	239 746	277 368	317 730	360 832	406 674	455 256
	8 118 904	8 184 327	8 242 692	8 315 731	8 399 784	8 484 983	8 560 644	8 642 099	8 745 720
	261 138	284 172	311 674	336 844	366 378	393 684	425 250	454 692	488 290
	664 630	664 630	664 630	664 630	664 630	664 630	664 630	664 630	664 630
2	172 722	204 864	239 746	277 368	317 730	360 832	406 674	455 256	506 578
	8 423 371	8 486 387	8 538 435	8 627 243	8 664 083	8 742 331	8 852 429	8 963 427	9 024 438
	277 952	299 754	327 776	351 794	381 768	408 002	439 928	468 378	502 256
	881 930	881 930	881 930	881 930	881 930	881 930	881 930	881 930	881 930
3	204 864	239 746	277 368	317 730	360 832	406 674	455 256	506 578	560 640
	8 771 492	8 782 919	8 856 422	8 919 595	8 986 908	9 071 399	9 168 864	9 265 595	9 370 492
	294 028	316 156	343 658	367 564	397 458	423 140	454 986	482 884	516 682
	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230
4	239 746	277 368	317 730	360 832	406 674	455 256	506 578	560 640	617 442
	9 006 571	9 055 687	9 167 863	9 206 299	9 345 570	9 390 395	9 463 287	9 562 175	9 669 823
	312 240	333 378	360 880	384 154	413 688	439 298	470 664	498 330	531 808
	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530
5	277 368	317 730	360 832	406 674	455 256	506 578	560 640	617 442	676 984
	9 307 434	9 309 587	9 442 476	9 524 067	9 610 112	9 738 031	9 783 788	9 880 731	9 949 693
	330 198	351 024	378 922	401 764	431 098	455 996	487 442	514 396	562 590
	1 533 830	1 533 830	1 533 830	1 533 830	1 533 830	1 533 830	1 533 830	1 533 830	1 533 830
6	317 730	360 832	406 674	455 256	506 578	560 640	617 442	676 984	739 266
	9 590 379	9 666 811	9 747 915	9 827 179	9 936 117	10 002 835	10 099 043	10 193 499	10 240 219
	350 408	370 282	397 784	419 794	449 328	473 474	505 040	531 322	564 204
	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130	1 751 130
7	360 832	406 674	455 256	506 578	560 640	617 442	676 984	739 266	806 288
	9 903 676	9 926 345	10 014 527	10 133 691	10 224 447	10 309 671	10 426 944	10 528 043	10 616 257
	370 722	390 084	417 486	433 190	485 022	491 892	523 458	549 108	
	1 968 430	1 968 430	1 968 430	1 968 430	1 968 430	1 968 430	1 968 430	1 968 430	
8	406 674	455 256	506 578	560 640	617 442	676 984	739 266	806 288	
	10 166 079	10 287 291	10 317 398	10 429 301	10 606 359	10 635 819	10 775 627	10 829 596	

i\ fj / le\	18	19	20	21	22	23	24	25	26
	495 862	531 492	566 470	604 132	641 246	690 940	720 190	761 916	803 302
	230 020	230 030	230 030	230 030	230 030	230 030	230 030	230 030	230 030
0	455 256	506 578	560 640	617 442	676 984	739 266	804 288	872 050	942 552
	8 523 116	8 622 292	8 723 756	8 785 104	8 934 980	9 022 016	9 134 068	9 255 880	9 380 092
	507 634	543 264	577 610	615 272	651 754	691 448	730 066	771 792	812 546
	447 330	447 330	447 330	447 330	447 330	447 330	447 330	447 330	447 330
1	506 578	560 640	617 442	676 984	739 266	804 288	872 050	942 552	1 015 794
	8 899 799	8 916 388	9 028 979	9 115 816	9 245 111	9 334 916	9 453 651	9 578 232	9 686 407
	519 868	555 498	577 770	626 874	662 724	702 418	740 404	782 130	822 252
	664 630	664 630	664 630	664 630	664 630	664 630	664 630	664 630	664 630
2	560 640	617 442	676 984	739 266	804 288	872 050	942 552	1 015 794	1 091 776
	9 113 315	9 235 185	9 285 268	9 436 099	9 540 267	9 660 107	9 790 067	9 854 085	10 014 099
	532 922	568 752	601 631	639 416	674 514	714 248	751 562	793 288	832 778
	881 930	881 930	881 930	881 930	881 930	881 930	881 930	881 930	881 930
3	617 442	676 984	739 266	804 288	872 050	942 552	1 015 794	1 091 776	1 170 498
	9 456 683	9 489 423	9 635 755	9 752 108	9 879 228	9 984 352	10 120 955	10 228 356	10 363 527
	546 796	582 546	614 876	652 578	687 124	726 818	763 540		
	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230	1 099 230		
4	676 984	739 266	804 288	872 050	942 552	1 015 794	1 091 776		
	9 479 203	9 868 767	9 960 387	10 082 839	10 200 331	10 298 447	10 382 243		
	581 490	597 120	628 938	666 600	700 554				
	1 316 530	1 316 530	1 316 530	1 316 530	1 316 530				
5	739 266	804 288	872 050	942 552	1 015 794				
	10 072 487	10 184 702	10 293 123	10 368 452	10 519 799				
	577 004	612 634	643 820						
	1 533 830	1 533 830	1 533 830						
6	664 288	872 050	942 552						
	10 399 331	10 510 683	10 615 787						
	589 792								
	1 751 120								
7	872 050								
	10 720 055								

i\ fi						
n\	27	28	29	30	31	32
	847 060	890 582	936 372	982 030	1 029 852	1 078 618
	230 030	230 030	230 030	230 030	230 030	230 030
0	1 015 794	1 091 776	1 170 498	1 251 960	1 336 162	1 423 104
	9 486 696	9 641 244	9 759 904	9 899 348	10 025 520	10 178 127
	856 304	899 196	944 984	990 010		
	447 330	447 330	447 330	447 330		
1	1 091 776	1 170 498	1 251 960	1 336 162		
	9 823 044	9 952 467	10 105 704	10 224 567		
	866 010	908 268				
	664 630	664 630				
2	1 170 498	1 251 960				
	10 171 683	10 293 584				

i\ fi										
i\	0	1	2	3	4	5	6	7	8	
	214 652	249 208	259 594	278 968	291 050	312 456	326 674	349 754	366 506	
	2 185 730	2 185 730	2 185 730	2 185 730	2 185 730	2 185 730	2 185 730	2 185 730	2 185 730	
9	116 658	143 320	172 722	204 864	239 746	277 368	317 730	360 832	406 674	
	9 934 284	10 011 989	10 041 593	10 110 956	10 160 635	10 219 297	10 259 207	10 319 172	10 435 891	
	237 818	274 130	283 004	302 020	313 590	335 354	349 060	372 498	388 340	
	2 403 030	2 403 030	2 403 030	2 403 030	2 403 030	2 403 030	2 403 030	2 403 030	2 403 030	
10	143 220	172 722	264 864	239 746	277 368	317 730	360 832	406 674	455 256	
	10 177 185	10 293 757	10 352 651	10 414 927	10 444 105	10 486 884	10 597 907	10 632 851	10 691 360	
	263 836	298 634	304 128	327 248	337 986	358 796	372 466	395 944	411 114	
	2 620 330	2 620 330	2 620 330	2 620 330	2 620 330	2 620 330	2 620 330	2 620 330	2 620 330	
11	172 722	264 864	239 746	277 368	317 730	360 832	406 674	455 256	506 578	
	10 472 659	10 606 417	10 657 559	10 718 554	10 764 252	10 840 943	10 856 787	10 987 839	11 058 549	
	289 958	325 314	333 164	352 658	362 844	383 574	396 692	420 130	434 708	
	2 837 630	2 837 630	2 837 630	2 837 630	2 837 630	2 837 630	2 837 630	2 837 630	2 837 630	
12	204 864	239 746	277 368	317 730	360 832	406 674	455 256	506 578	560 640	
	10 829 309	10 890 639	10 957 113	11 021 216	11 044 858	11 191 765	11 230 672	11 292 159	11 354 357	
	316 184	352 056	354 138	378 132	388 522	409 570	421 738			
	3 054 930	3 054 930	3 054 930	3 054 930	3 054 930	3 054 930	3 054 930			
13	239 746	277 368	317 730	360 832	406 674	455 256	506 578			
	11 135 246	11 202 940	11 274 157	11 340 791	11 377 452	11 455 668	11 530 805			
	343 946	379 858	386 604	405 978	413 946					
	3 272 230	3 272 230	3 272 230	3 272 230	3 272 230					
14	277 368	317 730	360 832	406 674	455 256					
	11 440 093	11 510 040	11 572 573	11 641 564	11 698 241					
	373 244	408 042	413 838							
	3 489 530	3 489 530	3 489 530							
15	317 730	360 832	406 674							
	11 735 488	11 818 632	11 877 552							
	401 930									
	3 706 830									
16	360 832									
	12 048 991									

i\ fi in\ .i	9	10	11	12	13	14
	391 976	410 506	438 008	457 998	487 890	511 113
	2 185 730	2 185 730	2 185 730	2 185 730	2 185 730	2 185 730
9	455 256	506 578	560 640	617 442	676 984	739 266
	10 513 568	10 586 807	10 637 375	10 757 316	10 859 045	10 975 159
	413 850	431 072	459 290	479 404		
	2 403 030	2 403 030	2 403 030	2 403 030		
10	506 578	560 640	617 442	676 984		
	10 823 715	10 917 307	10 973 339	11 048 097		
	436 584	453 930				
	2 620 330	2 620 330				
11	560 640	617 442				
	11 151 092	11 179 820				

TABLA DEL ALGORITMO MODIFICADO DE DECODIFICACION PARA CODIGOS R.S.

i \ fi	0	1	2	3	4	5	6	7	8
	0	46 270	62 068	77 866	93 724	109 582	125 460	141 358	157 276
	0	0	0	0	0	0	0	0	0
0	0	3 112	9 404	19 312	32 420	49 144	69 068	92 608	119 348
	7 154 620	7 252 474	7 298 488	7 348 138	7 401 008	7 457 515	7 517 240	7 580 603	7 647 184
	47 826	63 526	80 250	96 994	113 758	130 542	147 346	164 170	181 014
	447 310	644 610	891 910	1 099 210	1 316 510	1 533 810	1 751 110	1 968 410	2 185 710
1	3 112	9 404	19 312	32 420	49 144	69 068	92 608	119 348	149 704
	7 701 371	7 964 587	8 232 443	8 503 519	8 778 231	9 056 163	9 337 731	9 622 519	9 910 943
	64 614	81 240	98 890	116 560	134 250	151 960	169 690	187 440	205 210
	664 610	881 910	1 099 210	1 316 510	1 533 810	1 751 110	1 968 410	2 185 710	2 403 010
2	9 404	19 312	32 420	49 144	69 068	92 608	119 348	149 704	183 260
	7 965 674	8 233 432	8 505 415	8 781 032	9 059 871	9 342 344	9 628 039	9 917 368	10 208 499
	81 860	99 412	117 988	136 584	155 200	173 836	192 492	211 168	229 864
	881 910	1 099 210	1 316 510	1 533 810	1 751 110	1 968 410	2 185 710	2 403 010	2 620 310
3	19 312	32 420	49 144	69 068	92 608	119 348	149 704	183 260	220 432
	8 234 052	8 505 936	8 782 460	9 062 264	9 345 584	9 817 513	9 922 420	10 423 567	10 512 968
	99 564	115 202	137 544	157 066	176 608	196 170	215 752	235 354	254 976
	1 099 210	1 316 510	1 533 810	1 751 110	1 968 410	2 185 710	2 403 010	2 620 310	2 837 610
4	32 420	49 144	69 068	92 608	119 348	149 704	183 260	220 432	260 804
	8 505 703	8 782 041	9 222 403	9 347 451	9 634 956	9 926 099	10 219 040	10 518 459	10 819 676
	117 726	137 130	157 558	178 006	198 474	218 962	239 470	259 998	280 546
	1 316 510	1 533 810	1 751 110	1 968 410	2 185 710	2 403 010	2 620 310	2 837 610	3 054 910
5	49 144	69 068	92 608	119 348	149 704	183 260	220 432	260 804	304 792
	8 736 181	9 016 045	9 300 480	9 588 264	9 879 495	10 172 655	10 712 247	10 772 569	11 078 662
	136 346	156 676	178 030	199 404	220 798	242 212	263 646	285 100	306 574
	1 533 810	1 751 110	1 968 410	2 185 710	2 403 010	2 620 310	2 837 610	3 054 910	3 272 210
6	69 068	92 608	119 348	149 704	183 260	220 432	260 804	304 792	351 980
	9 015 339	9 299 675	9 588 206	9 880 503	10 174 569	10 475 042	10 984 739	11 085 293	11 392 492
	155 424	176 680	198 960	221 260	243 580	265 920	288 280	310 660	469 710
	1 751 110	1 968 410	2 185 710	2 403 010	2 620 310	2 837 610	3 054 910	3 272 210	3 489 510
7	92 608	119 348	149 704	183 260	220 432	260 804	304 792	351 980	402 784
	9 298 501	9 587 033	9 880 137	10 175 100	10 476 488	10 779 736	11 086 551	11 396 656	11 783 029
	174 960	197 142	220 348	243 574	266 820	290 086	313 372	336 678	360 004
	1 968 410	2 185 710	2 403 010	2 620 310	2 837 610	3 054 910	3 272 210	3 489 510	3 706 810
8	119 348	149 704	183 260	220 432	260 804	304 792	351 980	402 784	456 788
	9 584 742	9 878 397	10 174 275	10 476 561	10 888 152	11 089 436	11 398 796	11 714 023	12 031 889

i\ ti te\	9	10	11	12	13	14	15	16	17
	173 214	189 172	205 150	221 148	237 166	253 204	269 262	285 340	301 438
	0	0	0	0	0	0	0	0	0
0	149 704	163 260	220 432	260 804	304 792	351 980	402 784	456 788	514 408
	7 717 403	7 789 420	7 867 915	7 948 208	8 032 139	8 119 288	8 210 075	8 304 080	8 401 723
	197 878	214 762	231 646	248 590	265 534	282 498	299 482	316 486	333 510
	2 403 010	2 620 310	2 837 610	3 054 910	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410
1	182 250	220 432	260 804	304 792	351 980	402 784	456 788	514 408	575 228
	10 201 167	10 497 867	10 796 367	11 098 503	11 403 659	11 712 851	12 025 063	12 340 911	12 659 979
	223 000	240 810	258 640	276 496	294 360	312 250	330 160	348 090	366 040
	2 620 310	2 837 610	3 054 910	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410	4 358 710
2	220 432	260 804	304 792	351 980	402 784	456 788	514 408	575 228	639 664
	10 500 104	10 805 511	11 108 552	11 414 815	11 724 712	12 037 831	12 354 584	12 674 559	12 998 168
	248 580	267 316	286 072	304 84	323 644	342 460	361 296	380 152	399 028
	2 837 610	3 054 910	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410	4 358 710	4 576 010
3	260 804	304 792	351 980	402 784	456 788	514 408	575 228	639 664	707 300
	10 815 280	11 117 228	11 424 396	11 735 200	12 049 224	12 366 884	12 687 784	13 011 251	13 340 016
	274 618	294 280	313 962	333 664	353 386	373 128	392 890	412 672	432 474
	3 054 910	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410	4 358 710	4 576 010	4 793 310
4	304 792	351 980	402 784	456 788	514 408	575 228	639 664	707 300	778 552
	11 124 531	11 432 604	11 744 315	12 054 244	12 377 811	12 699 596	13 025 019	13 353 660	13 685 939
	301 114	321 702	342 310	362 938	383 586	404 254	424 942	445 650	466 378
	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410	4 358 710	4 576 010	4 793 310	5 010 610
5	351 980	402 784	456 788	514 408	575 228	639 664	707 300	778 552	853 004
	11 386 955	11 698 813	12 013 961	12 331 323	12 654 681	12 980 252	13 309 112	13 641 539	13 975 287
	328 068	349 582	371 116	392 670	414 244	435 838	457 452	479 086	500 740
	3 489 510	3 706 810	3 924 110	4 141 410	4 358 710	4 576 010	4 793 310	5 010 610	5 227 910
6	402 784	456 788	514 408	575 228	639 664	707 300	778 552	853 004	931 072
	11 705 257	12 080 412	12 340 932	12 711 355	12 990 220	13 316 381	13 651 891	13 990 041	14 330 231
	355 480	377 920	400 360	422 860	445 380	467 880	490 420	512 980	535 560
	3 706 810	3 924 110	4 141 410	4 358 710	4 576 010	4 793 310	5 010 610	5 227 910	5 445 210
7	456 788	514 408	575 228	639 664	707 300	778 552	853 004	931 072	1 012 340
	12 027 267	12 347 815	12 671 631	12 999 013	13 249 460	13 665 925	14 001 423	14 342 549	14 686 933
	383 350	406 716	430 102	453 508	476 934	500 380	523 845	547 332	
	3 924 110	4 141 410	4 358 710	4 576 010	4 793 310	5 010 610	5 227 910	5 445 210	
8	514 408	575 228	639 664	707 300	778 552	853 004	931 072	1 012 340	
	12 357 245	12 629 093	13 006 233	13 337 912	13 534 863	14 011 491	14 351 611	14 425 313	

i\ t	18	19	20	21	22	23	24	25	26
	317 556	333 694	349 852	366 030	382 228	398 446	414 684	430 942	447 220
	0	0	0	0	0	0	0	0	0
0	575 220	639 664	737 300	778 552	853 004	931 072	1 012 340	1 097 224	1 185 308
	8 502 584	8 607 083	8 714 800	8 826 155	8 940 728	9 058 939	9 180 368	9 305 435	9 433 720
	350 556	367 618	384 702	401 806	418 930	436 074	453 238	470 422	487 626
	4 358 710	4 576 010	4 793 310	5 010 610	5 227 910	5 445 210	5 662 510	5 879 810	6 097 110
1	639 664	707 300	778 552	853 004	931 072	1 012 340	1 097 224	1 185 308	1 277 008
	12 982 683	13 308 607	13 638 167	13 970 947	14 307 363	14 646 999	14 990 271	15 336 763	15 682 347
	384 010	402 009	420 010	438 040	456 090	474 160	492 250	510 360	528 490
	4 576 010	4 793 310	5 010 610	5 227 910	5 445 210	5 662 510	5 879 810	6 097 110	6 314 410
2	707 300	778 552	853 004	931 072	1 012 340	1 097 224	1 185 308	1 277 008	1 371 908
	13 324 999	13 655 464	13 989 151	14 326 472	14 667 015	15 011 192	15 358 591	15 705 080	16 059 335
	417 924	436 840	455 776	474 732	493 708	512 704	531 720	550 756	569 812
	4 793 310	5 010 610	5 227 910	5 445 210	5 662 510	5 879 810	6 097 110	6 314 410	6 531 710
3	778 552	853 004	931 072	1 012 340	1 097 224	1 185 308	1 277 008	1 371 908	1 470 424
	13 671 389	14 005 981	14 344 208	14 685 656	15 030 741	15 153 292	15 726 440	16 081 600	16 440 397
	452 296	472 138	492 000	511 882	531 784	551 706	571 648		
	5 010 610	5 227 910	5 445 210	5 662 510	5 879 810	6 097 110	6 314 410		
4	853 004	931 072	1 012 340	1 097 224	1 185 308	1 277 008	1 371 908		
	14 021 437	14 360 571	14 702 925	14 841 617	15 398 125	15 746 427	16 102 493		
	487 126	507 894	528 682	549 490	570 318				
	5 227 910	5 445 210	5 662 510	5 879 810	6 097 110				
5	931 072	1 012 340	1 097 224	1 185 308	1 277 008				
	14 316 539	14 657 141	15 005 249	15 354 679	15 703 129				
	522 414	544 108	565 822						
	5 445 210	5 662 510	5 879 810						
6	1 012 340	1 097 224	1 185 308						
	14 673 708	15 020 755	15 371 088						
	558 180								
	5 662 510								
7	1 097 224								
	15 034 884								

i\ fi n\ i	27	28	29	30	31	32
	463 518	479 836	496 174	512 532	528 910	545 308
	0	0	0	0	0	0
0	1 277 008	1 371 908	1 470 424	1 572 140	1 677 472	1 786 004
	9 561 099	9 696 240	9 835 019	9 977 016	10 122 651	10 271 535
	504 850	522 094	539 358	556 642		
	6 314 410	6 531 710	6 749 010	6 966 310		
1	1 371 908	1 470 424	1 572 140	1 677 472		
	16 035 695	16 392 679	16 752 883	17 116 722		
	546 640	564 810				
	6 531 710	6 749 010				
2	1 470 424	1 572 140				
	16 417 224	16 778 334				

i \ f	0	1	2	3	4	5	6	7	8
	194 954	216 806	242 194	266 346	290 518	314 710	338 922	363 154	387 406
	2 185 710	2 403 010	2 620 310	2 837 610	3 054 910	3 272 210	3 489 510	3 706 810	3 924 110
9	149 704	183 260	220 452	260 804	304 792	351 980	402 784	456 788	514 408
	10 010 650	10 122 575	10 425 218	10 729 591	11 037 461	11 348 689	11 663 416	11 981 500	12 303 083
	215 406	239 440	264 498	289 576	314 674	339 792	364 930	390 088	415 266
	2 403 010	2 620 310	2 837 610	3 054 910	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410
10	183 260	220 432	260 804	304 792	351 980	402 784	456 788	514 408	575 228
	10 120 284	10 422 619	10 727 899	11 035 849	11 348 809	11 693 236	11 983 432	12 305 920	12 631 767
	236 316	261 276	287 260	313 264	339 288	365 332	382 748	417 480	438 104
	2 620 310	2 837 610	3 054 910	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410	4 358 710
11	220 432	260 804	304 792	351 980	402 784	456 788	514 409	575 228	639 664
	10 419 178	10 724 623	11 033 337	11 347 555	11 664 093	11 983 076	12 305 029	12 634 136	12 962 120
	257 684	283 570	310 480	331 402	364 360	391 330	418 320	445 330	472 360
	2 837 610	3 054 910	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410	4 358 710	4 576 010
12	260 804	304 792	351 980	402 784	456 788	514 408	575 228	639 664	707 300
	10 721 396	11 030 981	11 347 683	11 661 371	11 983 173	12 307 475	12 635 152	12 872 949	13 300 903
	279 510	306 322	341 580	362 014	389 890	417 786	445 702		
	3 054 910	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410	4 358 710		
13	304 792	351 980	402 784	456 788	514 408	575 228	639 664		
	11 027 077	11 340 925	11 659 276	11 979 983	12 306 191	12 634 755	12 966 817		
	301 794	329 532	358 294	387 076	415 878				
	3 272 210	3 489 510	3 706 810	3 924 110	4 141 410				
14	351 980	402 784	456 788	514 408	575 228				
	11 336 344	11 654 805	11 977 420	12 185 521	12 633 003				
	324 536	358 060	382 808						
	3 489 510	3 706 810	3 924 110						
15	402 784	456 788	514 408						
	11 649 756	11 865 385	12 299 500						
	347 736								
	3 706 810								
16	456 788								
	11 913 629								

f \ f e\	9	10	11	12	13	14
	411 678	435 970	460 282	484 614	508 966	533 338
	4 141 410	4 358 710	4 576 010	4 793 310	5 010 610	5 227 910
9	575 228	639 664	707 300	778 552	853 004	931-072
	12 577 819	12 956 461	13 288 257	13 623 552	13 962 204	14 111 065
	440 464	465 682	490 920	516 178		
	4 358 710	4 576 010	4 793 310	5 010 610		
10	639 664	707 300	778 552	853 004		
	12 961 111	13 293 813	13 630 013	13 969 572		
	469 708	495 852				
	4 576 010	4 793 310				
11	707 300	778 552				
	13 135 700	13 635 101				

TABLA DEL ALGORITMO DE DECODIFICACION QUE OPERA EN EL DOMINIO DE LA FRECUENCIA PARA CODIGOS R.S.

# \	0	1	2	3	4	5	6	7
0	7 159 164	8 348 626	8 473 159	8 597 892	8 727 099	8 851 780	8 976 463	9 101 204
1	8 350 184	8 474 708	8 600 250	8 730 368	8 855 956	8 981 564	9 107 192	9 232 840
2	8 475 796	8 601 247	8 732 264	8 858 759	8 985 272	9 111 807	9 238 360	9 364 935
3	8 601 867	8 732 787	8 860 187	8 987 607	9 115 047	9 242 507	9 369 987	9 506 574
4	8 732 939	8 860 240	8 988 005	9 116 912	9 244 629	9 372 927	9 502 071	9 630 496
5	8 859 924	8 988 152	9 117 404	9 246 676	9 375 495	9 505 280	9 633 875	9 763 964
6	8 987 368	9 116 523	9 256 700	9 376 699	9 507 116	9 634 615	9 767 612	9 897 891
7	9 115 271	9 245 351	9 376 455	9 507 579	9 637 309	9 769 325	9 910 158	10 031 185
8	9 243 245	9 374 636	9 506 667	9 638 716	9 770 787	9 902 876	10 034 987	10 167 116
9	9 372 448	9 504 380	9 637 336	9 770 312	9 903 308	10 036 324	10 169 360	10 302 416
10	9 501 427	9 634 583	9 768 464	9 902 367	10 036 288	10 170 231	10 304 192	10 438 175
11	9 631 859	9 765 243	9 899 401	10 034 879	10 190 928	10 303 329	10 439 465	10 574 351
12	9 761 651	9 900 663	10 032 095	10 236 400	10 303 623	10 439 419	10 575 231	10 711 064
13	9 892 300	10 027 936	10 200 340	10 301 276	10 437 976	10 565 187	10 711 436	
14	10 040 395	10 182 650	10 309 110	10 435 163	10 572 788			
15	10 154 975	10 291 637	10 430 975					
16	10 286 999							

f \	8	9	10	11	12	13	14	15
0	9 225 947	9 330 708	9 475 491	9 600 292	9 725 115	9 849 956	9 974 819	10 099 700
1	9 356 508	9 484 196	9 609 904	9 735 632	9 861 380	9 987 148	10 112 936	10 238 744
2	9 491 528	9 618 143	9 744 776	9 870 349	9 999 104	10 124 799	10 251 512	10 378 247
3	9 625 607	9 752 547	9 880 107	10 007 687	10 135 287	10 262 907	10 390 547	10 518 207
4	9 758 943	9 887 408	10 015 895	10 144 400	10 272 927	10 401 472	10 548 126	10 692 125
5	9 893 336	10 022 728	10 152 140	10 281 572	10 411 024	10 537 349	10 669 988	10 799 500
6	10 026 923	10 158 507	10 287 491	10 419 203	10 635 527	10 679 979	10 846 748	10 940 835
7	10 163 499	10 331 094	10 426 007	10 557 291	10 688 595	10 818 477	10 951 263	11 082 627
8	10 299 267	10 431 436	10 563 627	10 695 836	10 828 067	10 960 316	11 092 587	11 224 876
9	10 435 492	10 568 588	10 700 615	10 833 311	10 964 849	11 101 172	11 234 348	
10	10 572 176	10 706 199	10 840 240	10 974 303	11 105 680			
11	10 708 141	10 844 267	10 979 235					
12	10 845 125							

f \	16	17	18	19	20	21	22	23
0	10 224 603	10 349 524	10 474 467	10 599 428	10 724 411	10 849 412	10 974 435	11 099 476
1	10 364 572	10 490 420	10 616 288	10 740 383	10 868 084	10 994 012	11 119 960	11 245 928
2	10 505 000	10 631 775	10 758 568	10 885 383	11 012 216	11 139 071	11 265 944	11 392 839
3	10 645 887	10 773 587	10 901 307	11 029 047	11 156 807	11 284 587	11 412 387	11 540 207
4	10 787 251	10 916 869	11 046 503	11 173 168	11 320 401	11 430 560	11 559 287	11 688 032
5	10 927 151	11 058 584	11 188 156	11 317 748	11 447 360	11 576 992	11 706 644	
6	11 071 292	11 201 771	11 332 268	11 462 787	11 591 091			
7	11 229 764	11 345 415	11 476 839					
8	11 357 187							

n \ f	24	25	26	27	28	29	30	31
0	11 224 539	11 349 620	11 474 725	11 599 844	11 724 987	11 850 148	11 975 331	12 100 536
1	11 371 916	11 497 924	11 623 952	11 750 000	11 876 068	12 002 156	12 128 268	
2	11 519 752	11 646 687	11 773 640	11 900 615	12 027 608			
3	11 668 047	11 795 907	11 923 787					
4	11 816 799							

n \ f	32
0	12 230 328

TABLA DEL ALGORITMO DE DECODIFICACION QUE OPERA EN EL DOMINIO DE LA FRECUENCIA PARA CODIGOS R.S. (Tiempo en segundos).

i\ fi e\	0	1	2	3	4	5	6	7	8	9	10	11
	4.28	4.25	4.25	4.25	4.25	4.24	4.25	4.25	4.24	4.24	4.25	4.24
	0.00	0.03	0.04	0.04	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	0.00	0.05	0.10	0.14	0.19	0.23	0.28	0.32	0.36	0.42	0.46	0.49
	0.00	0.06	0.07	0.05	0.04	0.07	0.07	0.05	0.05	0.04	0.04	0.01
0	0.00	0.07	0.18	0.18	0.17	0.16	0.16	0.17	0.16	0.14	0.17	0.14
	0.00	0.43	0.44	0.49	0.46	0.45	0.45	0.47	0.45	0.46	0.47	0.46
	0.00	0.66	0.69	0.72	0.67	0.68	0.65	0.72	0.65	0.64	0.69	0.61
	0.00	0.01	0.01	0.01	0.02	0.02	0.02	0.01	0.02	0.02	0.01	0.02
	4.28	5.03	5.08	5.15	5.17	5.21	5.27	5.27	5.24	5.35	5.50	5.45
	4.28	4.28	4.25	4.25	4.25	4.26	4.26	4.26	4.28	4.28	4.29	4.30
	0.05	0.02	0.04	0.04	0.04	0.06	0.07	0.04	0.04	0.04	0.10	0.11
	0.05	0.10	0.14	0.19	0.23	0.28	0.31	0.37	0.42	0.46	0.51	0.56
	0.04	0.05	0.04	0.05	0.05	0.08	0.04	0.04	0.04	0.04	0.06	0.04
1	0.17	0.16	0.17	0.17	0.15	0.16	0.17	0.16	0.14	0.14	0.14	0.15
	0.49	0.46	0.46	0.46	0.46	0.47	0.46	0.49	0.45	0.48	0.48	0.48
	0.70	0.67	0.70	0.68	0.65	0.71	0.67	0.68	0.64	0.70	0.65	0.66
	0.01	0.02	0.02	0.01	0.01	0.02	0.01	0.02	0.02	0.01	0.02	0.01
	5.07	5.09	5.15	5.17	5.21	5.33	5.31	5.41	5.44	5.54	5.59	5.64
	4.28	4.27	4.28	4.26	4.25	4.26	4.25	4.25	4.25	4.25	4.26	4.25
	0.04	0.04	0.05	0.05	0.06	0.07	0.02	0.04	0.10	0.10	0.12	0.11
	0.11	0.14	0.19	0.24	0.27	0.32	0.37	0.42	0.46	0.50	0.62	0.59
	0.05	0.06	0.06	0.04	0.02	0.06	0.05	0.06	0.04	0.03	0.06	0.07
2	0.15	0.16	0.15	0.17	0.17	0.16	0.15	0.15	0.15	0.14	0.15	0.15
	0.44	0.47	0.47	0.45	0.45	0.47	0.47	0.49	0.47	0.48	0.49	0.47
	0.64	0.69	0.68	0.65	0.69	0.69	0.67	0.70	0.66	0.65	0.70	0.69
	0.01	0.02	0.02	0.01	0.01	0.01	0.02	0.02	0.02	0.01	0.02	0.02
	5.07	5.16	5.21	5.22	5.29	5.35	5.36	5.49	5.49	5.51	5.70	5.65
	4.25	4.25	4.27	4.25	4.29	4.30	4.30	4.35	4.34	4.34	4.36	4.37
	0.04	0.05	0.06	0.06	0.07	0.08	0.09	0.10	0.10	0.11	0.12	0.13
	0.14	0.19	0.25	0.27	0.36	0.37	0.43	0.48	0.50	0.56	0.60	0.66
	0.07	0.14	0.06	0.02	0.09	0.04	0.07	0.05	0.05	0.06	0.06	0.06
3	0.16	0.17	0.16	0.17	0.17	0.11	0.09	0.09	0.10	0.09	0.10	0.09
	0.47	0.46	0.47	0.46	0.48	0.50	0.46	0.47	0.45	0.45	0.48	0.50
	0.70	0.67	0.68	0.65	0.69	0.65	0.62	0.61	0.63	0.61	0.67	0.65
	0.02	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
	5.14	5.17	5.22	5.25	5.43	5.41	5.45	5.55	5.50	5.64	5.73	5.82
	4.29	4.30	4.29	4.29	4.29	4.29	4.29	4.30	4.35	4.31	4.30	4.30
	0.05	0.06	0.07	0.07	0.08	0.09	0.10	0.12	0.11	0.12	0.14	0.15
	0.19	0.25	0.29	0.32	0.37	0.42	0.44	0.50	0.56	0.60	0.65	0.70
	0.04	0.18	0.16	0.05	0.05	0.05	0.05	0.05	0.06	0.05	0.05	0.07
4	0.17	0.14	0.15	0.13	0.14	0.12	0.13	0.12	0.11	0.12	0.12	0.12
	0.47	0.47	0.46	0.46	0.46	0.46	0.47	0.47	0.45	0.47	0.50	0.47
	0.67	0.65	0.67	0.64	0.65	0.62	0.65	0.66	0.62	0.66	0.68	0.65
	0.01	0.01	0.02	0.02	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02
	5.22	5.27	5.32	5.25	5.40	5.48	5.52	5.60	5.58	5.71	5.79	5.82

		12	13	14	15	16	17	18	19	20	21	22	23
		4.24	4.24	4.25	4.24	4.25	4.25	4.25	4.24	4.24	4.24	4.25	4.25
		0.11	0.11	0.11	0.12	0.12	0.13	0.14	0.14	0.14	0.15	0.16	0.16
		0.55	0.59	0.65	0.69	0.73	0.78	0.84	0.86	1.02	0.96	1.00	1.17
		0.05	0.06	0.07	0.03	0.05	0.04	0.08	0.04	0.09	0.07	0.05	0.07
0		0.15	0.15	0.16	0.15	0.15	0.15	0.15	0.15	0.15	0.14	0.15	0.14
		0.46	0.49	0.46	0.48	0.53	0.48	0.48	0.47	0.48	0.50	0.51	0.47
		0.66	0.70	0.69	0.66	0.73	0.67	0.70	0.66	0.71	0.72	0.70	0.67
		0.02	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.01	0.02	0.01
		5.58	5.65	5.71	5.73	5.85	5.84	5.94	5.92	6.13	6.08	6.13	6.27
		4.30	4.30	4.30	4.30	4.30	4.30	4.30	4.27	4.27	4.27	4.27	4.29
		0.10	0.12	0.12	0.13	0.14	0.14	0.17	0.16	0.18	0.17	0.17	0.18
		0.60	0.65	0.70	0.74	0.79	0.85	0.97	0.92	0.97	1.02	1.18	1.10
		0.05	0.05	0.05	0.05	0.04	0.06	0.04	0.05	0.04	0.04	0.07	0.06
1		0.14	0.15	0.14	0.15	0.14	0.14	0.15	0.14	0.15	0.14	0.14	0.15
		0.46	0.49	0.49	0.51	0.47	0.49	0.48	0.56	0.51	0.51	0.47	0.49
		0.65	0.69	0.67	0.70	0.65	0.68	0.66	0.75	0.70	0.69	0.68	0.70
		0.01	0.02	0.01	0.02	0.02	0.02	0.02	0.01	0.02	0.02	0.02	0.02
		5.66	5.77	5.81	5.89	5.90	5.99	6.01	6.11	6.13	6.16	6.32	6.27
		4.25	4.25	4.25	4.25	4.26	4.25	4.25	4.25	4.25	4.25	4.25	4.25
		0.12	0.13	0.14	0.14	0.15	0.15	0.17	0.17	0.18	0.18	0.19	0.20
		0.64	0.69	0.73	0.78	0.82	0.87	0.92	0.96	1.13	1.03	1.23	1.13
		0.04	0.07	0.06	0.05	0.05	0.05	0.06	0.02	0.03	0.06	0.04	0.05
2		0.15	0.16	0.17	0.16	0.15	0.15	0.15	0.14	0.14	0.14	0.15	0.14
		0.47	0.48	0.47	0.46	0.48	0.49	0.51	0.50	0.49	0.46	0.49	0.48
		0.65	0.71	0.70	0.66	0.68	0.69	0.72	0.65	0.66	0.66	0.68	0.68
		0.01	0.02	0.02	0.01	0.01	0.02	0.02	0.02	0.01	0.02	0.02	0.02
		5.68	5.80	5.83	5.93	5.92	5.98	6.07	6.05	6.24	6.15	6.36	6.28
		4.37	4.31	4.30	4.30	4.30	4.30	4.30	4.31	4.31	4.31	4.31	4.31
		0.14	0.14	0.15	0.16	0.17	0.17	0.18	0.19	0.19	0.20	0.21	0.22
		0.71	0.74	0.79	0.84	0.88	0.92	0.98	1.02	1.20	1.25	1.16	1.20
		0.07	0.06	0.04	0.05	0.06	0.05	0.05	0.05	0.06	0.06	0.06	0.04
3		0.08	0.12	0.13	0.14	0.13	0.13	0.12	0.12	0.14	0.12	0.12	0.12
		0.48	0.48	0.52	0.52	0.50	0.48	0.49	0.53	0.50	0.48	0.48	1.43
		0.63	0.65	0.69	0.71	0.68	0.66	0.65	0.70	0.70	0.66	0.65	1.59
		0.02	0.02	0.02	0.01	0.02	0.02	0.01	0.02	0.02	0.02	0.02	0.02
		5.86	5.85	5.95	6.02	6.15	6.07	6.12	6.24	6.42	6.44	6.34	7.33
		4.31	4.29	4.27	4.28	4.28	4.27	4.28	4.28	4.28	4.28	4.28	4.27
		0.19	0.16	0.18	0.18	0.18	0.20	0.21	0.23	0.22	0.21	0.22	0.26
		0.74	0.78	1.02	0.93	0.92	1.01	1.01	1.05	1.13	1.14	1.19	1.23
		0.06	0.05	0.05	0.04	0.06	0.04	0.03	0.05	0.06	0.05	0.05	0.04
4		0.12	0.13	0.13	0.12	0.13	0.14	0.14	0.14	0.14	0.13	0.14	0.14
		0.47	0.48	0.49	0.47	0.49	0.51	0.49	0.48	0.53	0.49	0.50	0.49
		0.65	0.66	0.68	0.65	0.67	0.69	0.66	0.67	0.73	0.67	0.69	0.67
		0.02	0.01	0.01	0.01	0.02	0.01	0.02	0.02	0.01	0.02	0.02	0.02
		5.90	5.89	6.17	6.05	6.06	6.17	6.18	6.24	6.37	6.32	6.40	6.45

		24	25	26	27	28	29	30	31	32
		4.25	4.25	4.25	4.27	4.27	4.27	4.27	4.27	4.28
		0.17	0.18	0.18	0.19	0.20	0.20	0.21	0.21	0.22
		1.10	1.14	1.18	1.39	1.28	1.50	1.37	1.61	1.48
		0.04	0.05	0.10	0.06	0.05	0.05	0.05	0.05	0.05
0		0.15	0.14	0.15	0.14	0.13	0.13	0.14	0.13	0.13
		0.46	0.49	0.49	0.51	0.50	0.48	0.50	0.49	0.50
		0.65	0.67	0.73	0.71	0.68	0.66	0.69	0.66	0.68
		0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
		6.18	6.25	6.36	6.57	6.45	6.64	6.55	6.77	6.67

		4.27	4.26	4.25	4.27	4.28	4.27	4.28
		0.19	0.20	0.20	0.21	0.21	0.22	0.23
		1.28	1.18	1.22	1.29	1.34	1.38	1.44
		0.04	0.02	0.07	0.06	0.05	0.05	0.05
1		0.15	0.15	0.14	0.14	0.15	0.13	0.13
		0.47	0.50	0.48	0.53	0.47	0.51	0.47
		0.66	0.68	0.69	0.72	0.67	0.69	0.65
		0.01	0.02	0.02	0.02	0.01	0.01	0.02
		6.40	6.33	6.37	6.51	6.31	6.57	6.62

		4.25	4.25	4.25	4.24	4.25
		0.22	0.21	0.22	0.22	0.24
		1.18	1.23	1.43	1.33	1.37
		0.05	0.06	0.06	0.05	0.05
2		0.15	0.14	0.13	0.15	0.16
		0.50	0.47	0.50	0.50	0.50
		0.69	0.66	0.69	0.70	0.72
		0.02	0.02	0.01	0.02	0.01
		6.37	6.37	6.61	6.50	6.58

		4.31	4.31	4.31
		0.22	0.26	0.26
		1.24	1.30	1.35
		0.06	0.05	0.07
3		0.11	0.11	0.12
		0.48	0.53	0.51
		0.65	0.71	0.70
		0.02	0.02	0.03
		6.44	6.59	6.84

		4.26
		0.24
		1.28
		0.05
4		0.15
		0.49
		0.67
		0.02
		6.47

		0	1	2	3	4	5	6	7	8	9	10	11
		4.31	4.30	4.30	4.30	4.31	4.34	4.31	4.31	4.31	4.30	4.31	4.31
		0.05	0.07	0.08	0.08	0.09	0.10	0.11	0.12	0.13	0.13	0.14	0.15
		0.26	0.28	0.33	0.37	0.48	0.48	0.51	0.56	0.60	0.74	0.79	0.74
		0.05	0.06	0.06	0.05	0.05	0.05	0.05	0.04	0.05	0.06	0.05	0.06
5		0.14	0.14	0.13	0.11	0.12	0.10	0.14	0.12	0.12	0.13	0.12	0.12
		0.47	0.46	0.46	0.48	0.47	0.47	0.46	0.47	0.55	0.50	0.49	0.54
		0.66	0.66	0.65	0.64	0.64	0.62	0.65	0.63	0.73	0.68	0.66	0.72
		0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
		5.30	5.32	5.37	5.42	5.53	5.55	5.59	5.63	5.78	5.86	5.90	5.93
		4.28	4.29	4.28	4.28	4.28	4.28	4.28	4.27	4.28	4.28	4.28	4.29
		0.06	0.08	0.07	0.09	0.10	0.11	0.12	0.13	0.15	0.14	0.15	0.16
		0.28	0.36	0.41	0.42	0.47	0.50	0.56	0.59	0.65	0.69	0.74	0.79
		0.07	0.08	0.07	0.07	0.05	0.05	0.05	0.05	0.06	0.05	0.07	0.05
6		0.15	0.14	0.15	0.14	0.14	0.13	0.14	0.12	0.14	0.14	0.14	0.12
		0.48	0.53	0.46	0.45	0.50	0.48	0.47	0.46	0.49	0.47	0.50	0.48
		0.69	0.74	0.68	0.66	0.68	0.65	0.65	0.63	0.69	0.65	0.70	0.66
		0.02	0.02	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.02
		5.33	5.49	5.45	5.47	5.54	5.56	5.62	5.64	5.78	5.78	5.88	5.91
		4.27	4.27	4.28	4.28	4.28	4.28	4.28	4.28	4.29	4.27	4.27	4.27
		0.07	0.08	0.09	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0.19	0.19
		0.32	0.37	0.42	0.47	0.50	0.56	0.59	0.65	0.69	0.74	0.78	0.93
		0.06	0.04	0.05	0.05	0.06	0.07	0.05	0.05	0.05	0.07	0.06	0.07
7		0.14	0.12	0.13	0.14	0.13	0.14	0.14	0.13	0.12	0.13	0.15	0.12
		0.47	0.48	0.49	0.45	0.48	0.46	0.47	0.48	0.47	0.50	0.47	0.47
		0.67	0.65	0.66	0.64	0.66	0.66	0.66	0.66	0.65	0.69	0.68	0.66
		0.02	0.01	0.02	0.01	0.02	0.02	0.02	0.01	0.02	0.02	0.02	0.02
		5.36	5.37	5.46	5.50	5.57	5.63	5.67	5.72	5.79	5.88	5.94	6.08
		4.25	4.25	4.25	4.25	4.25	4.25	4.25	4.25	4.25	4.26	4.27	4.25
		0.08	0.09	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0.17	0.17	0.19
		0.41	0.42	0.47	0.50	0.61	0.59	0.64	0.69	0.73	0.79	0.83	0.87
		0.05	0.04	0.06	0.04	0.07	0.06	0.05	0.04	0.04	0.08	0.06	0.06
8		0.15	0.16	0.15	0.15	0.16	0.15	0.15	0.15	0.15	0.15	0.14	0.15
		0.49	0.48	0.45	0.54	0.46	0.47	0.49	0.49	0.51	0.48	0.49	0.47
		0.69	0.67	0.66	0.73	0.68	0.68	0.69	0.68	0.69	0.71	0.69	0.68
		0.01	0.02	0.02	0.01	0.02	0.02	0.01	0.01	0.02	0.01	0.02	0.02
		5.43	5.44	5.49	5.60	5.69	5.66	5.73	5.78	5.85	5.93	5.97	6.00
		4.25	4.26	4.25	4.26	4.26	4.27	4.28	4.28	4.27	4.27	4.27	4.27
		0.10	0.11	0.10	0.12	0.13	0.14	0.15	0.16	0.17	0.18	0.19	0.20
		0.42	0.46	0.50	0.55	0.59	0.65	0.69	0.74	0.79	0.84	0.87	0.92
		0.08	0.07	0.04	0.03	0.06	0.03	0.05	0.04	0.03	0.02	0.06	0.05
9		0.16	0.15	0.16	0.15	0.16	0.15	0.15	0.14	0.15	0.14	0.15	0.15
		0.47	0.47	0.46	0.47	0.46	0.47	0.46	0.48	0.50	0.49	0.48	0.50
		0.70	0.68	0.66	0.65	0.67	0.65	0.66	0.66	0.67	0.65	0.69	0.70
		0.02	0.02	0.01	0.02	0.02	0.02	0.02	0.01	0.02	0.02	0.01	0.02
		5.46	5.53	5.53	5.56	5.66	5.73	5.80	5.85	5.91	5.95	6.03	6.10

		12	13	14	15	16	17	18	19	20	21	22
		4.30	4.30	4.31	4.31	4.30	4.30	4.31	4.31	4.31	4.31	4.31
		0.18	0.17	0.18	0.19	0.19	0.22	0.21	0.21	0.22	0.23	0.23
		0.79	0.84	0.88	0.93	0.97	1.02	1.06	1.11	1.15	1.20	1.24
		0.05	0.05	0.05	0.06	0.05	0.05	0.05	0.05	0.05	0.05	0.06
5		0.12	0.11	0.11	0.10	0.11	0.13	0.12	0.13	0.14	0.13	0.13
		0.48	0.49	0.48	0.47	0.51	0.57	0.49	0.49	0.48	0.49	0.50
		0.65	0.65	0.64	0.63	0.67	0.75	0.66	0.67	0.67	0.67	0.69
		0.01	0.02	0.03	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
		5.94	5.97	6.02	6.06	6.15	6.21	6.26	6.32	6.37	6.42	6.52

	4.29	4.31	4.25	4.28	4.25	4.28	4.28	4.28	4.28	4.28	
	0.19	0.20	0.22	0.20	0.21	0.22	0.23	0.23	0.24	0.24	
	0.98	0.97	0.91	0.97	1.01	1.06	1.10	1.15	1.19	1.19	
	0.04	0.05	0.05	0.04	0.05	0.05	0.06	0.07	0.04	0.04	
6	0.13	0.15	0.14	0.15	0.14	0.14	0.14	0.14	0.14	0.15	
	0.47	0.51	0.48	0.47	0.55	0.49	0.56	0.49	0.50	0.50	
	0.64	0.70	0.67	0.66	0.74	0.65	0.76	0.71	0.70	0.70	
	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	
	6.00	6.08	6.15	6.12	6.26	6.22	6.38	6.27	6.41		

	4.27	4.27	4.27	4.28	4.27	4.25	4.25	
	0.18	0.19	0.22	0.22	0.22	0.23	0.24	
	0.87	0.93	0.98	1.01	1.06	1.10	1.14	
	0.07	0.06	0.05	0.05	0.08	0.04	0.05	
7	0.13	0.13	0.12	0.14	0.13	0.16	0.15	
	0.56	0.49	0.49	0.49	0.48	0.47	0.55	
	0.76	0.67	0.65	0.67	0.69	0.67	0.75	
	0.02	0.02	0.02	0.01	0.01	0.02	0.01	
	6.10	6.08	6.11	6.19	6.24	6.29	6.39	

	4.25	4.25	4.25	4.24	4.25
	0.20	0.20	0.22	0.22	0.24
	0.91	0.96	1.00	1.04	1.10
	0.04	0.05	0.07	0.05	0.04
8	0.15	0.14	0.15	0.15	0.14
	0.49	0.47	0.49	0.52	0.50
	0.69	0.66	0.72	0.72	0.67
	0.02	0.02	0.01	0.02	0.02
	6.06	6.09	6.20	6.23	6.27

	4.27	4.27	4.28
	0.20	0.22	0.23
	0.96	1.01	1.05
	0.07	0.04	0.04
9	0.15	0.13	0.15
	0.48	0.51	0.47
	0.71	0.65	0.66
	0.01	0.01	0.02
	6.16	6.20	6.23

		0	1	2	3	4	5	6	7	8	9	10	11
		4.28	4.27	4.27	4.27	4.28	4.27	4.28	4.27	4.27	4.28	4.28	4.28
		0.09	0.12	0.12	0.13	0.14	0.15	0.15	0.17	0.18	0.19	0.20	0.22
		0.47	0.50	0.61	0.59	0.65	0.69	0.74	0.87	0.84	0.87	0.93	0.97
		0.04	0.02	0.06	0.05	0.05	0.06	0.05	0.01	0.03	0.06	0.06	0.06
10		0.15	0.14	0.15	0.15	0.15	0.16	0.15	0.15	0.14	0.14	0.13	0.13
		0.48	0.48	0.47	0.48	0.52	0.49	0.49	0.45	0.49	0.46	0.49	0.50
		0.67	0.65	0.69	0.69	0.72	0.71	0.69	0.65	0.66	0.66	0.68	0.69
		0.02	0.02	0.02	0.02	0.02	0.01	0.02	0.01	0.02	0.02	0.02	0.01
		5.92	5.95	5.72	5.86	5.81	5.83	5.85	5.94	5.97	6.01	6.10	6.16

		4.27	4.27	4.27	4.27	4.27	4.27	4.27	4.26	4.26	4.27	4.25	
		0.11	0.13	0.15	0.14	0.15	0.16	0.17	0.18	0.19	0.20	0.21	
		0.59	0.63	0.65	0.65	0.69	0.74	0.78	0.82	0.87	0.92	0.97	
		0.07	0.05	0.06	0.06	0.05	0.04	0.04	0.05	0.03	0.06	0.05	
11		0.14	0.15	0.15	0.15	0.16	0.14	0.15	0.15	0.15	0.13	0.14	
		0.47	0.49	0.46	0.47	0.51	0.49	0.49	0.48	0.48	0.47	0.50	
		0.69	0.65	0.66	0.64	0.72	0.67	0.66	0.66	0.66	0.76	0.69	
		0.02	0.01	0.01	0.02	0.02	0.01	0.01	0.02	0.01	0.02	0.02	
		5.90	5.91	5.95	5.71	5.95	5.84	5.90	5.97	6.02	6.17	6.15	

		4.26	4.25	4.25	4.25	4.24	4.25	4.25	4.25		
		0.11	0.13	0.16	0.16	0.16	0.17	0.20	0.20		
		0.55	0.63	0.64	0.73	0.74	0.78	0.83	0.86		
		0.06	0.04	0.05	0.05	0.05	0.05	0.04	0.04		
12		0.15	0.15	0.14	0.15	0.15	0.16	0.15	0.15		
		0.46	0.49	0.52	0.51	0.46	0.47	0.50	0.49		
		0.67	0.68	0.71	0.71	0.66	0.68	0.74	0.69		
		0.02	0.01	0.01	0.01	0.02	0.01	0.02	0.01		
		5.80	5.71	5.78	5.81	5.82	5.87	6.00	5.99		

		4.25	4.25	4.25	4.25	4.25	4.25	4.26	
		0.12	0.13	0.15	0.15	0.17	0.18	0.19	
		0.66	0.65	0.70	0.74	0.79	0.84	0.87	
		0.05	0.01	0.06	0.04	0.06	0.02	0.05	
13		0.14	0.15	0.15	0.15	0.15	0.15	0.14	
		0.49	0.50	0.48	0.47	0.47	0.46	0.49	
		0.69	0.68	0.69	0.67	0.69	0.64	0.69	
		0.01	0.02	0.02	0.01	0.02	0.02	0.01	
		5.73	5.72	5.81	5.85	5.89	5.83	6.02	

		4.25	4.25	4.24	4.25	4.25
		0.13	0.14	0.16	0.17	0.18
		0.66	0.69	0.72	0.76	0.83
		0.04	0.07	0.07	0.05	0.05
14		0.16	0.15	0.14	0.15	0.15
		0.47	0.46	0.47	0.45	0.49
		0.67	0.69	0.67	0.72	0.70
		0.01	0.02	0.02	0.02	0.01
		5.72	5.78	5.84	5.97	5.97

\	|	
le\		12
		4.22
		0.22
		1.21
		1.34
	10	0.15
		0.46
		0.67
		0.01
		6.19

\	|			
le\		0	1	2	
		4.25	4.23	4.25	
		0.14	0.15	0.17	
		0.72	0.74	0.87	
		0.44	0.65	0.26	
	15	0.15	0.15	0.14	
		0.47	0.51	0.47	
		0.67	0.71	0.67	
		0.02	0.01	0.02	
		5.84	5.85	5.99	

		4.25
		0.15
		0.72
		0.06
	16	0.14
		0.47
		0.67
		0.01
		5.31

III.2 GRAFICAS DE LOS ALGORITMOS DE DECODIFICACION.

Para facilitar la comparación de las tablas anteriores, estas se vaciaron en gráficas obteniéndose así las curvas del comportamiento de los tres algoritmos y para diferenciar las que corresponden a un algoritmo u otro, enseguida del titulo de la gráfica se encuentra entre paréntesis la inicial del nombre del algoritmo al cual pertenece dicha gráfica, es decir:

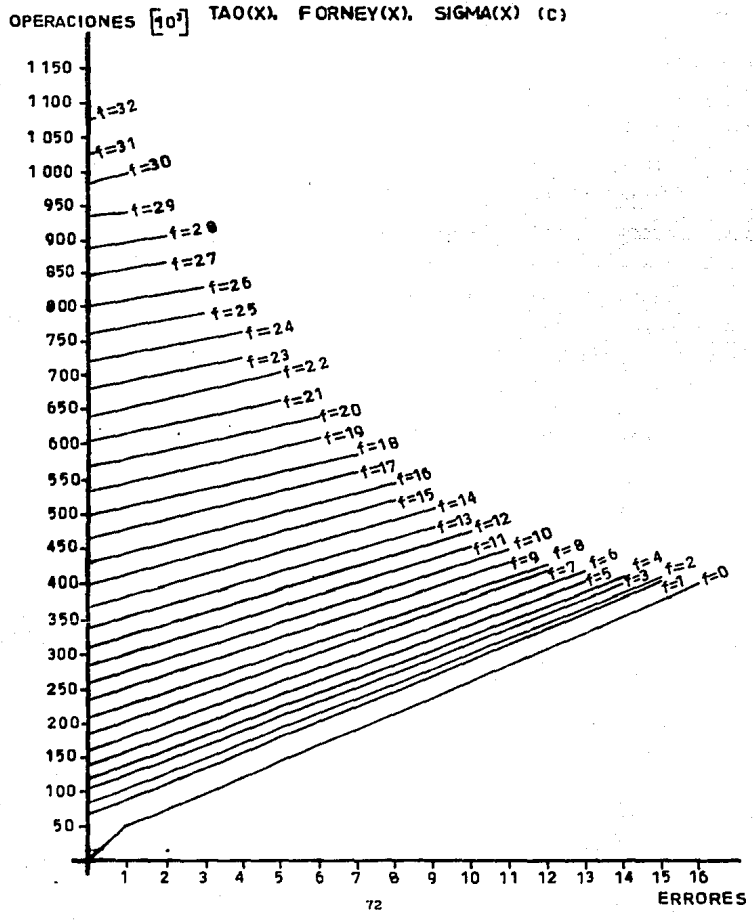
(C) Algoritmo Clásico.

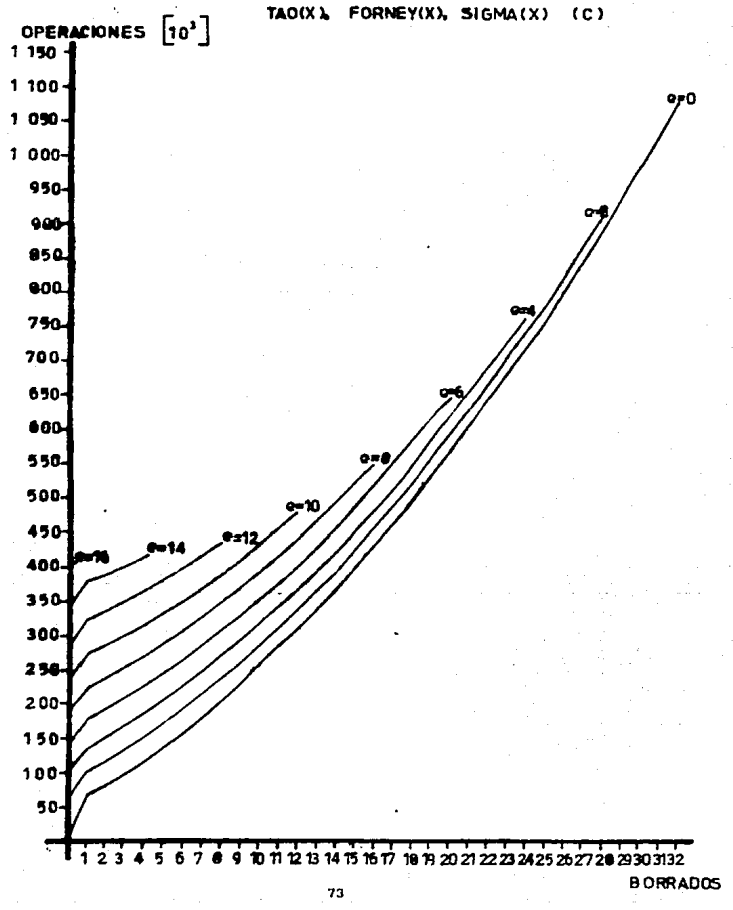
(M) Algoritmo Modificado.

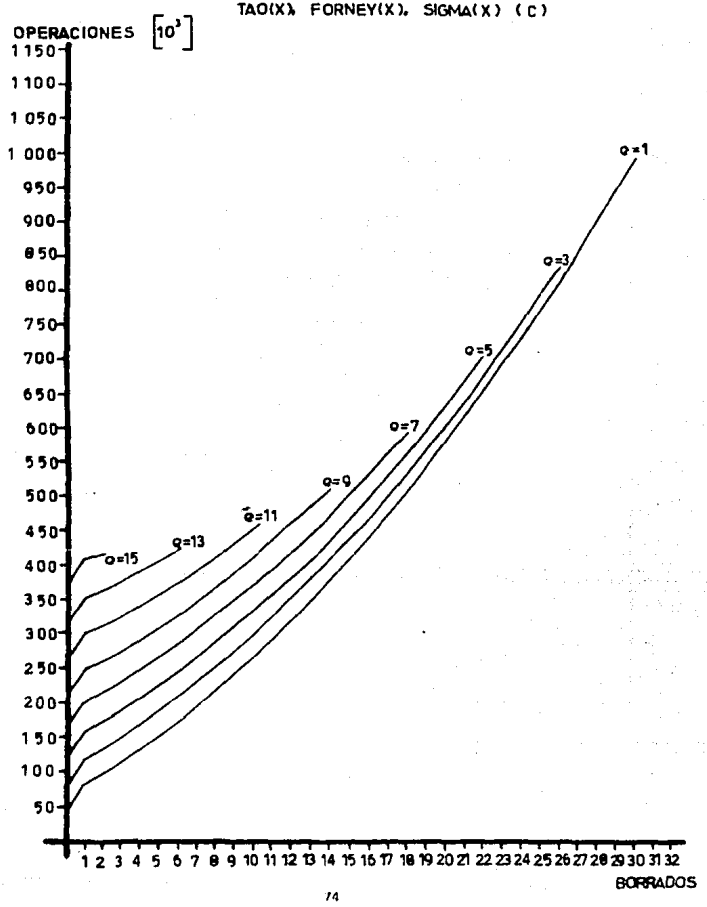
(F) Algoritmo en el dominio de la Frecuencia.

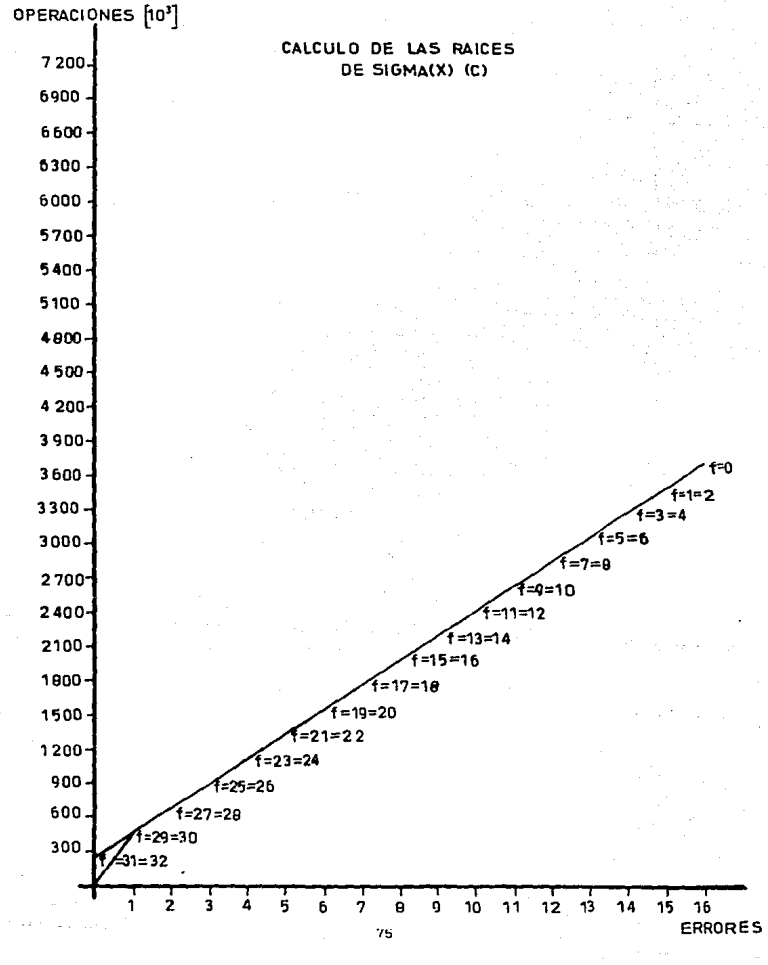
Cabe señalar que algunos puntos de las gráficas no coinciden con los valores que se encuentran en las tablas, esto se debe a que las curvas fueron suavizadas para evitar los quiebres que se presentaron. Tales quiebres se deben a que en la simulación se utilizaron funciones random, lo que provocó en algunos casos una variación en el número de operaciones, haciendo que ciertos puntos salieran del comportamiento observado en la curva.

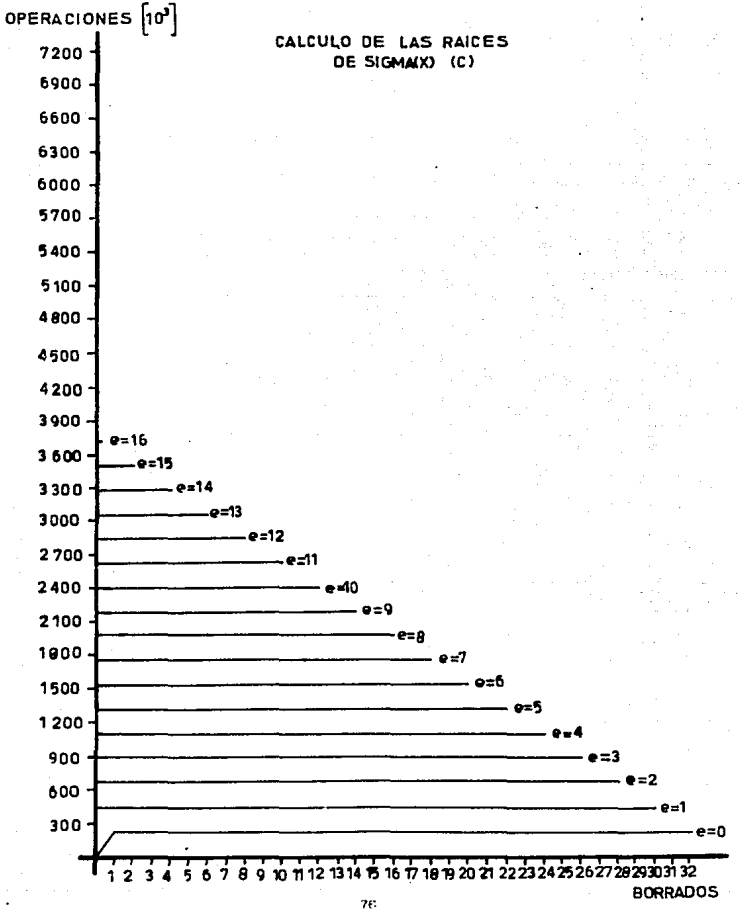
Enseguida se presentan las gráficas de los tres algoritmos y también las correspondientes al algoritmo en el dominio de la frecuencia en segundos.

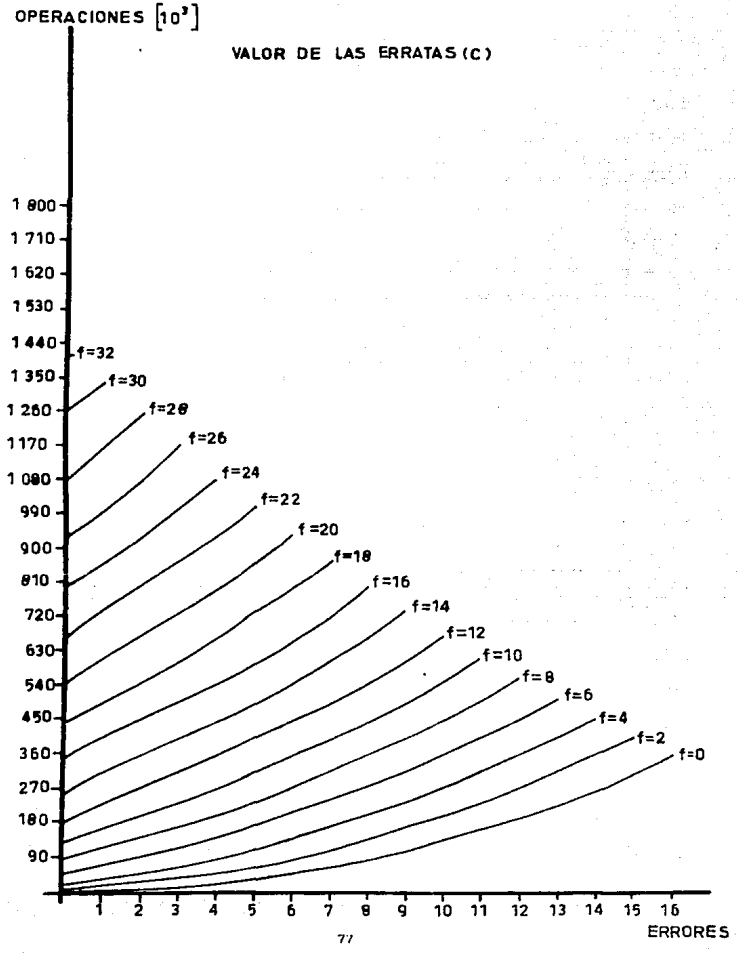

OPERACIONES [10³] TAO(X). FORNEY(X). SIGMA(X) (C)
1 150
1 100
1 050
1 000
950
900
850
800
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
f=32
f=31
f=30
f=29
f=28
f=27
f=26
f=25
f=24
f=23
f=22
f=21
f=20
f=19
f=18
f=17
f=16
f=15
f=14
f=13
f=12
f=11
f=10
f=9
f=8
f=7
f=6
f=5
f=4
f=3
f=2
f=1
f=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ERRORES

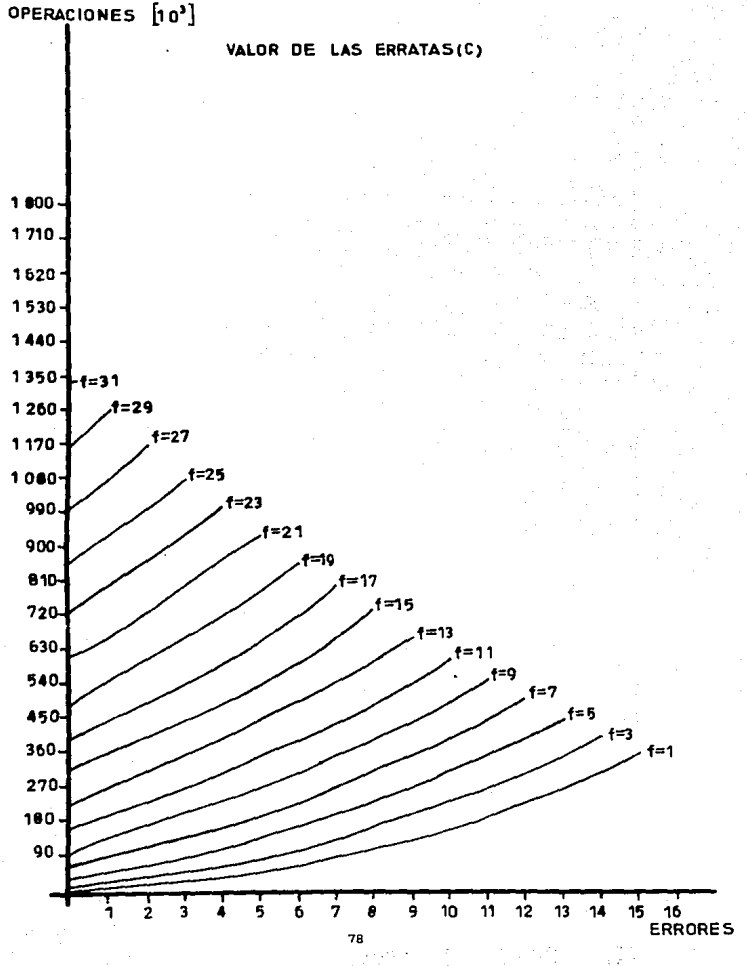

TAO(X), FORNEY(X), SIGMA(X) (C)
OPERACIONES [10³]
1 150
1 100
1 050
1 000
950
900
850
800
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
c=0
c=2
c=4
c=6
c=8
c=10
c=12
c=14
c=16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS

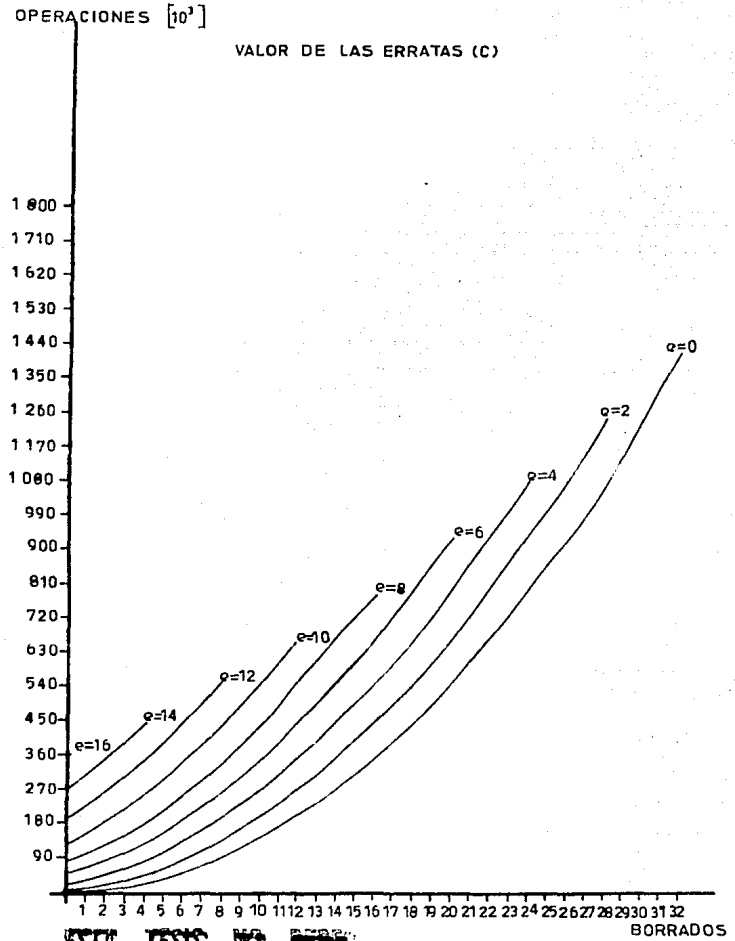

TAO(X), FORNEY(X), SIGMA(X) (C)
OPERACIONES [10³]
1 150
1 100
1 050
1 000
950
900
850
800
750
700
650
600
550
500
450
400
350
300
250
200
150
100
50
q=1
q=3
q=5
q=7
q=9
q=11
q=13
q=15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS

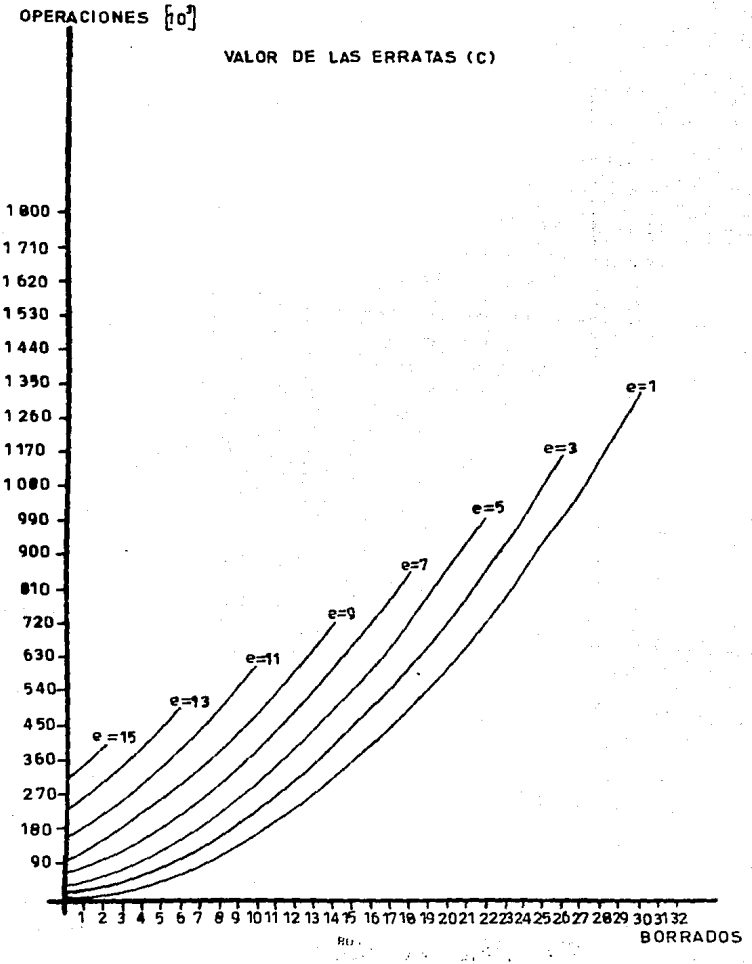

OPERACIONES [10³]
CALCULO DE LAS RAICES
DE SIGMA(X) (C)
7 200
6900
6600
6300
6000
5700
5400
5100
4800
4 500
4 200
3900
3600
3300
3000
2700
2400
2100
1800
1500
1200
900
600
300
f=0
f=1=2
f=3=4
f=5=6
f=7=8
f=9=10
f=11=12
f=13=14
f=15=16
f=17=18
f=19=20
f=21=22
f=23=24
f=25=26
f=27=28
f=29=30
f=31=32
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ERRORES

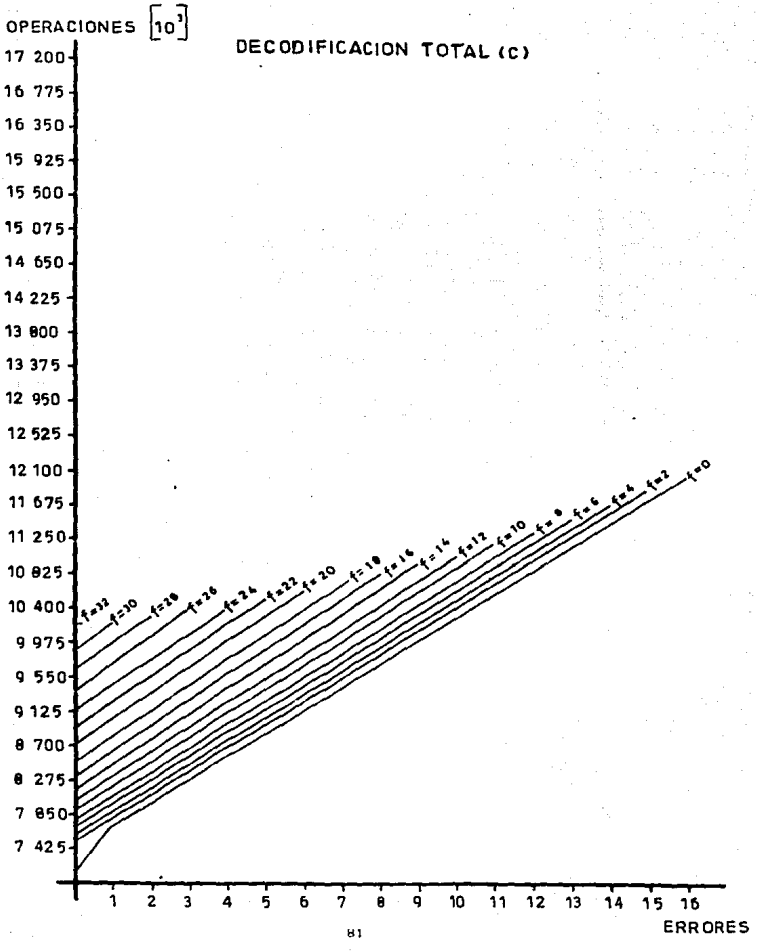

OPERACIONES [10³]
CALCULO DE LAS RAICES
DE SIGMA(X) (C)
7200
6900
6600
6300
6000
5700
5400
5100
4800
4500
4200
3900
3600
3300
3000
2700
2400
2100
1800
1500
1200
900
600
300
e=16
e=15
e=14
e=13
e=12
e=11
e=10
e=9
e=8
e=7
e=6
e=5
e=4
e=3
e=2
e=1
e=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS

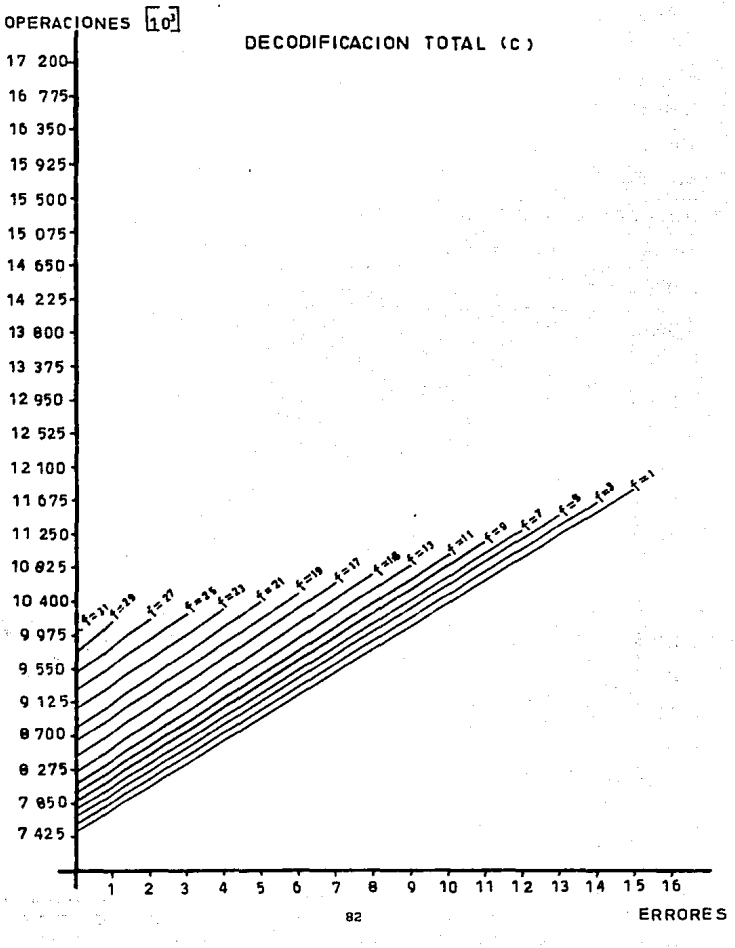

OPERACIONES [10³]
VALOR DE LAS ERRATAS (C)
1 800
1 710
1 620
1 530
1 440
1 350
1 260
1 170
1 080
990
900
810
720
630
540
450
360
270
180
90
f=32
f=30
f=28
f=26
f=24
f=22
f=20
f=18
f=16
f=14
f=12
f=10
f=8
f=6
f=4
f=2
f=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ERRORES

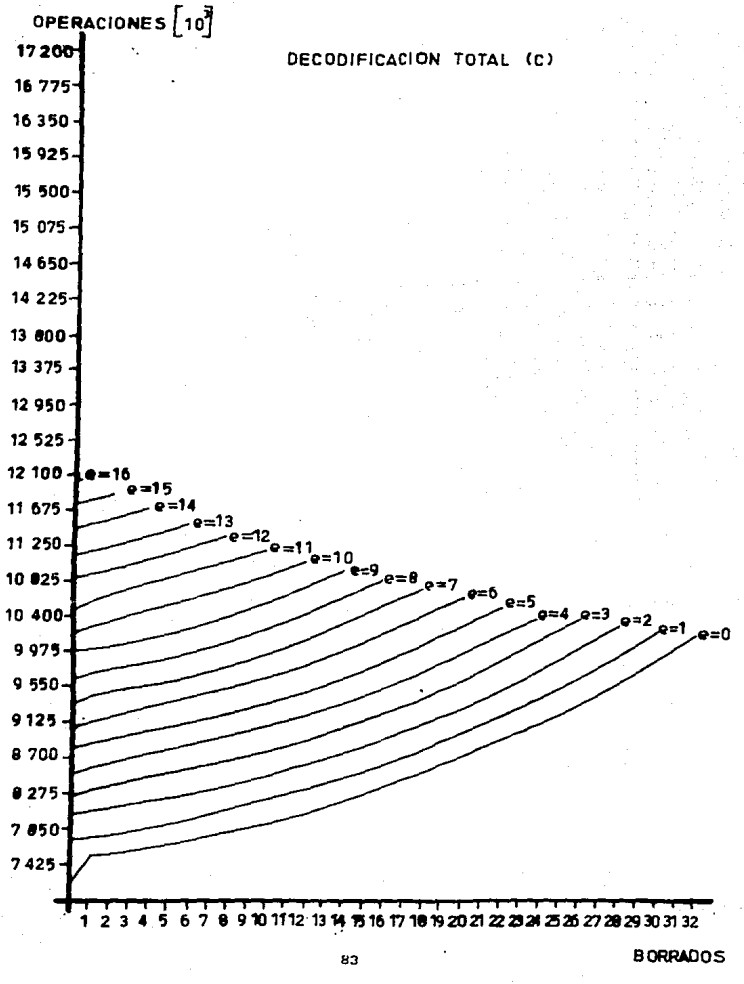

OPERACIONES [10³]
VALOR DE LAS ERRATAS(C)
1 800
1 710
1 620
1 530
1 440
1 350
1 260
1 170
1 080
990
900
810
720
630
540
450
360
270
180
90
f=31
f=29
f=27
f=25
f=23
f=21
f=19
f=17
f=15
f=13
f=11
f=9
f=7
f=5
f=3
f=1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ERRORES



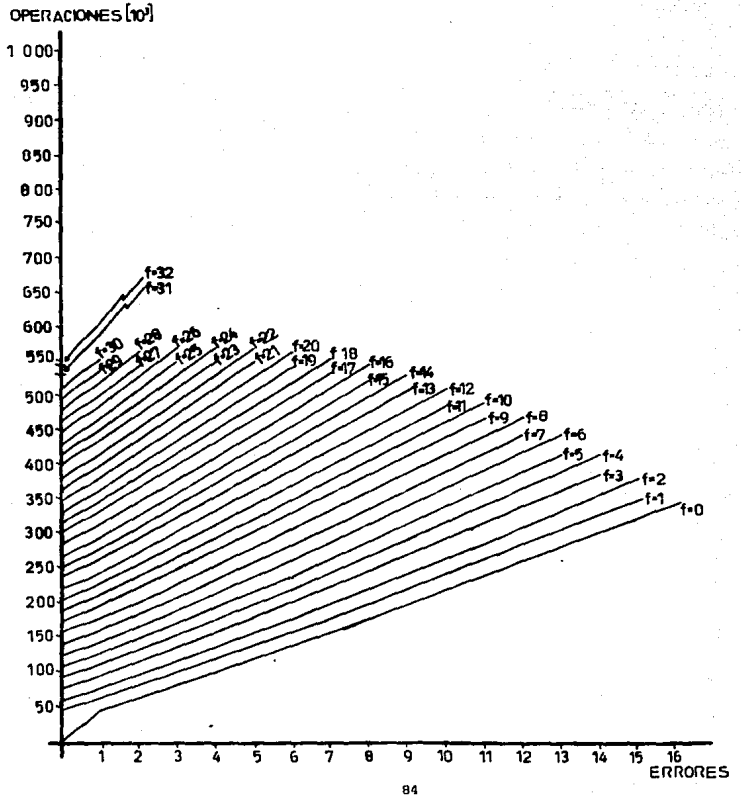

OPERACIONES [10³]
VALOR DE LAS ERRATAS (C)
1 800
1 710
1 620
1 530
1 440
1 350
1 260
1 170
1 080
990
900
810
720
630
540
450
360
270
180
90
e=1
e=3
e=5
e=7
e=9
e=11
e=13
e=15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS


OPERACIONES [10³]
DECODIFICACION TOTAL (C)
17 200
16 775
16 350
15 925
15 500
15 075
14 650
14 225
13 800
13 375
12 950
12 525
12 100
11 675
11 250
10 825
10 400
9 975
9 550
9 125
8 700
8 275
7 850
7 425
f=32
f=30
f=28
f=26
f=24
f=22
f=20
f=18
f=16
f=14
f=12
f=10
f=8
f=6
f=4
f=2
f=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ERRORES

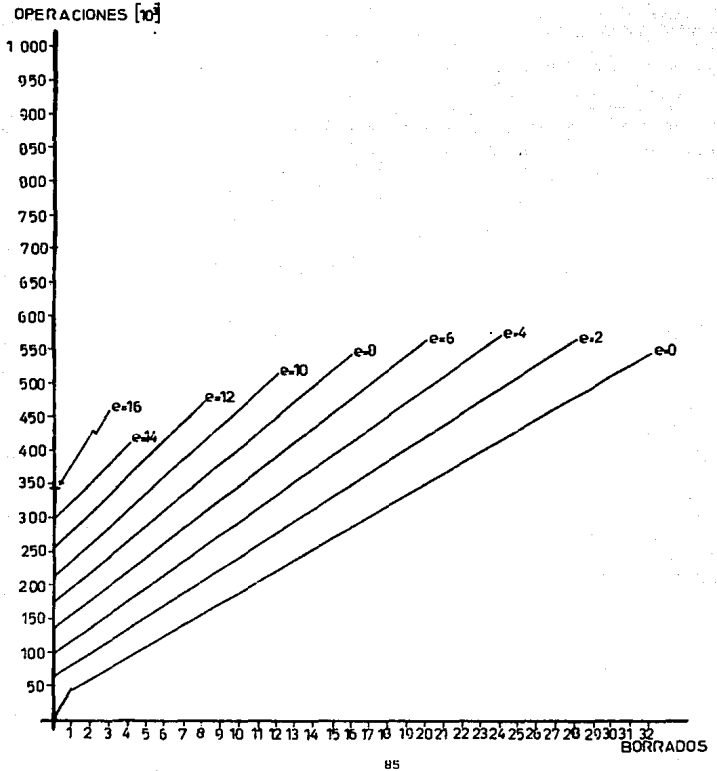

OPERACIONES [10³]
DECODIFICACION TOTAL (C)
17 200
16 775
16 350
15 925
15 500
15 075
14 650
14 225
13 800
13 375
12 950
12 525
12 100
11 675
11 250
10 825
10 400
9 975
9 550
9 125
8 700
8 275
7 850
7 425
f=31
f=29
f=27
f=25
f=23
f=21
f=19
f=17
f=15
f=13
f=11
f=9
f=7
f=5
f=3
f=1
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ERRORES


OPERACIONES [10³]
DECODIFICACION TOTAL (C)
17 200
16 775
16 350
15 925
15 500
15 075
14 650
14 225
13 800
13 375
12 950
12 525
12 100
11 675
11 250
10 825
10 400
9 975
9 550
9 125
8 700
8 275
7 850
7 425
e=16
e=15
e=14
e=13
e=12
e=11
e=10
e=9
e=8
e=7
e=6
e=5
e=4
e=3
e=2
e=1
e=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS

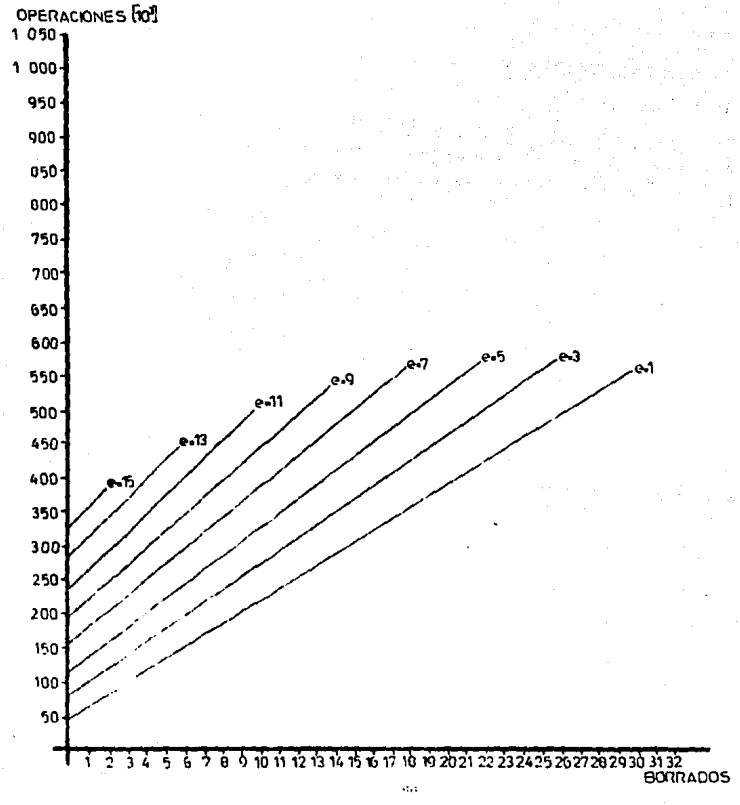
CALCULO DE P(X) (M)

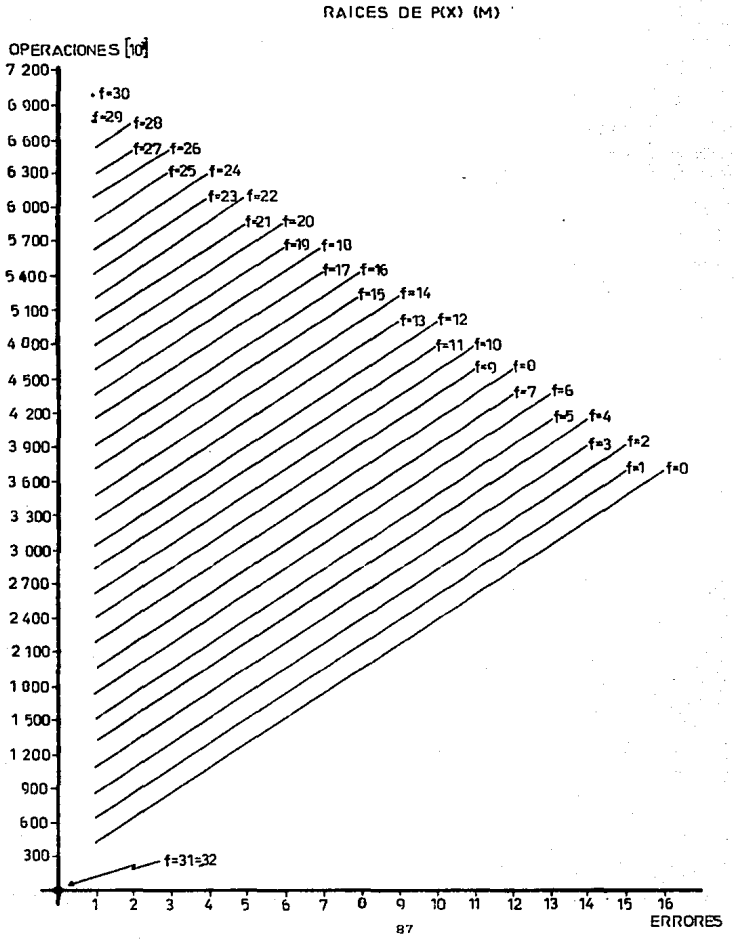


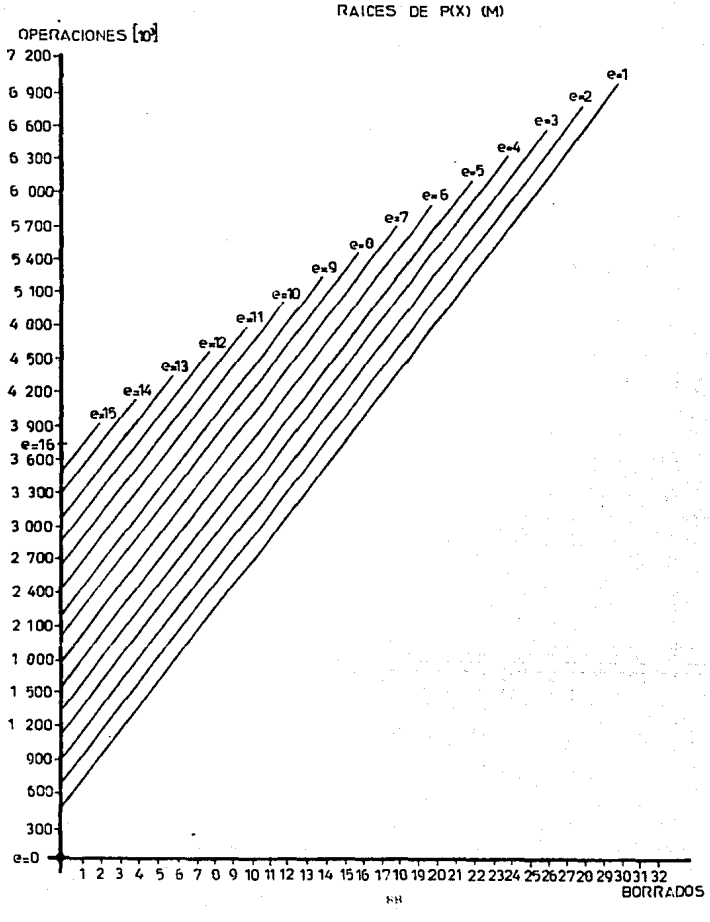
CALCULO DE P(X) (M)



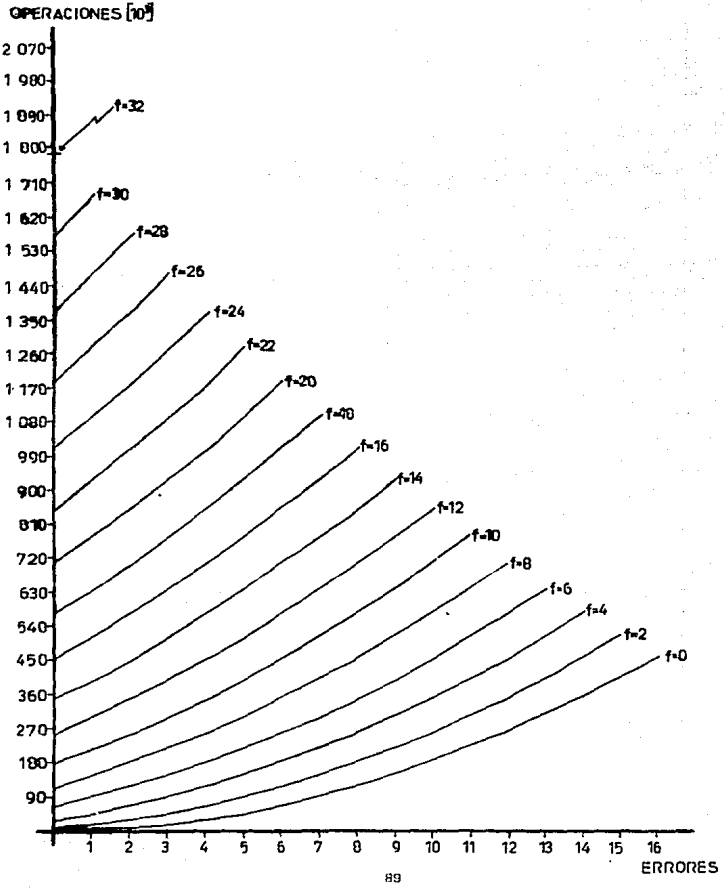
CALCULO DE P(X) (M)



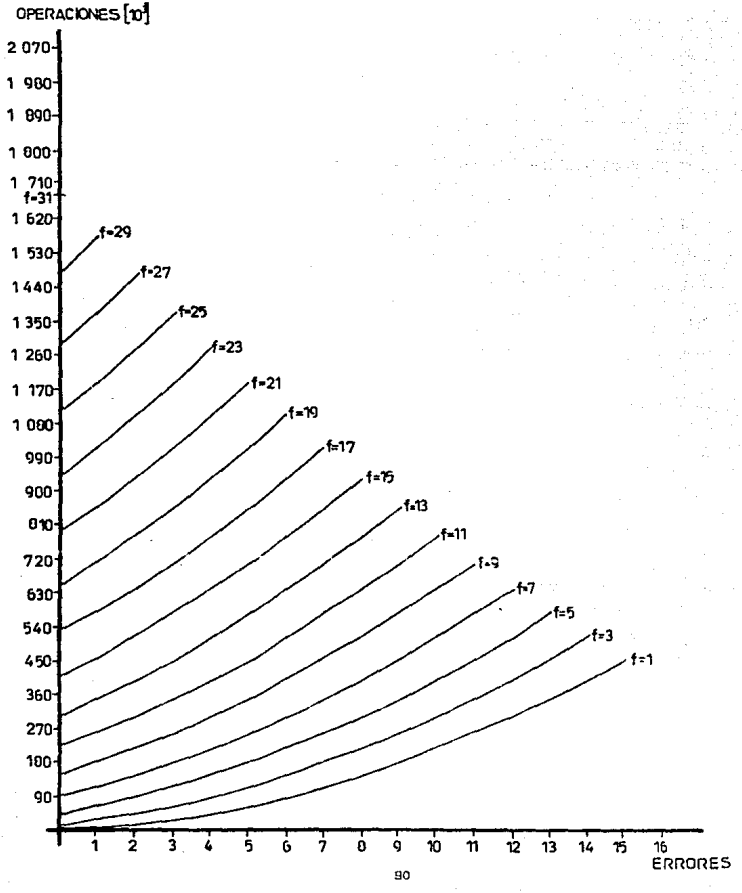

RAICES DE P(X) (M)
OPERACIONES [10³]
7 200
6 900
6 600
6 300
6 000
5 700
5 400
5 100
4 800
4 500
4 200
3 900
3 600
3 300
3 000
2 700
2 400
2 100
1 800
1 500
1 200
900
600
300
f=30
f=29
f=28
f=27
f=26
f=25
f=24
f=23
f=22
f=21
f=20
f=19
f=18
f=17
f=16
f=15
f=14
f=13
f=12
f=11
f=10
f=9
f=8
f=7
f=6
f=5
f=4
f=3
f=2
f=1
f=0
f=31=32
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ERRORES


RAICES DE P(X) (M)
OPERACIONES [10³]
7 200
6 900
6 600
6 300
6 000
5 700
5 400
5 100
4 000
4 500
4 200
3 900
3 600
3 300
3 000
2 700
2 400
2 100
1 000
1 500
1 200
900
600
300
e=0
e=1
e=2
e=3
e=4
e=5
e=6
e=7
e=8
e=9
e=10
e=11
e=12
e=13
e=14
e=15
e=16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 10 19 20 21 22 23 24 25 26 27 20 29 30 31 32
BORRADOS

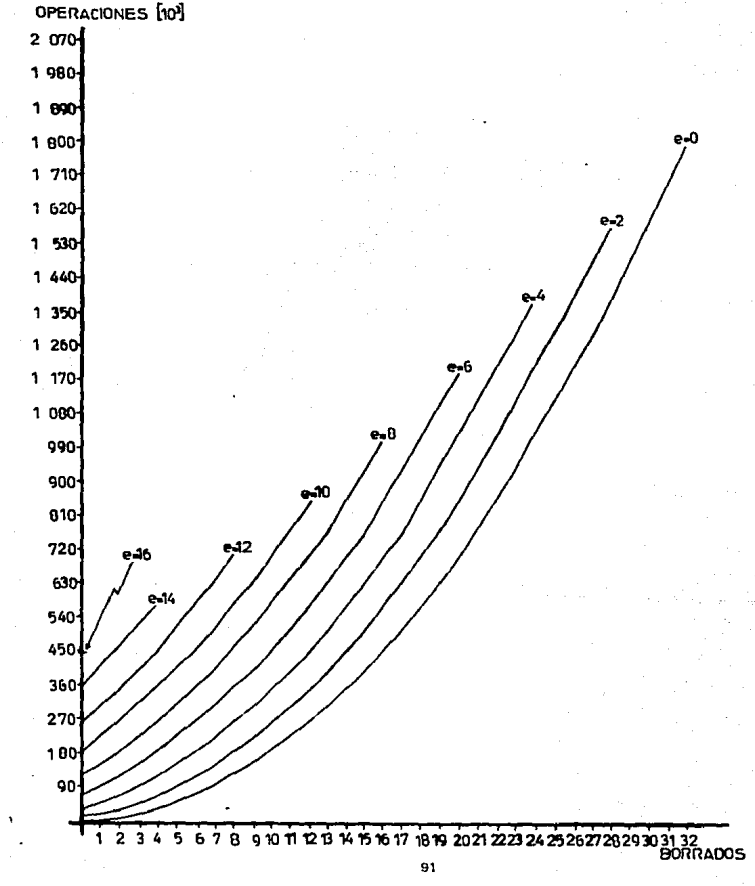
VALOR DE LAS ERRATAS (M)



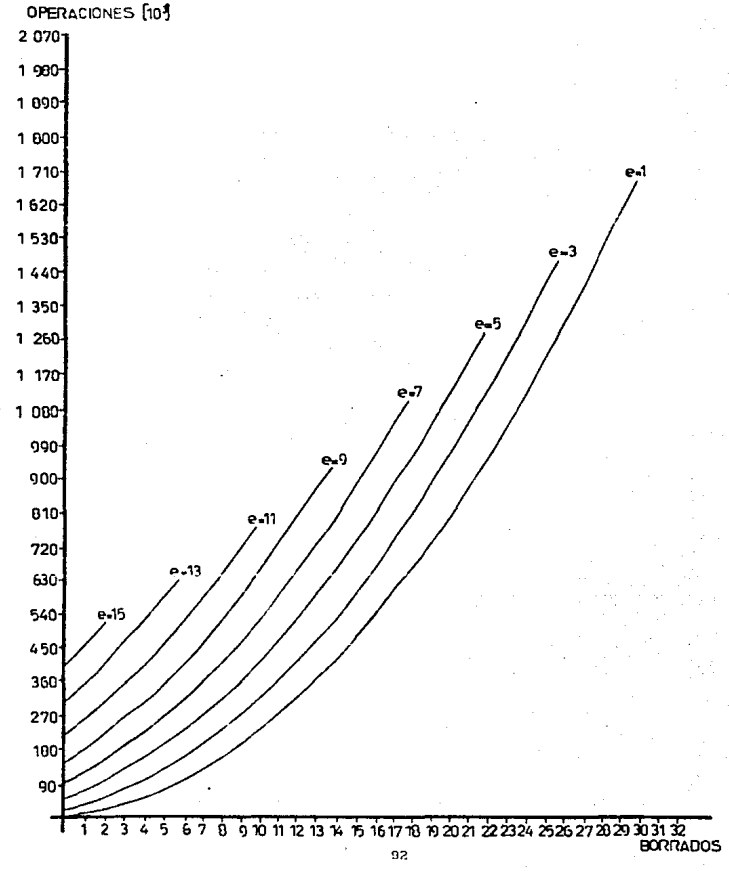
VALOR DE LAS ERRATAS (M)



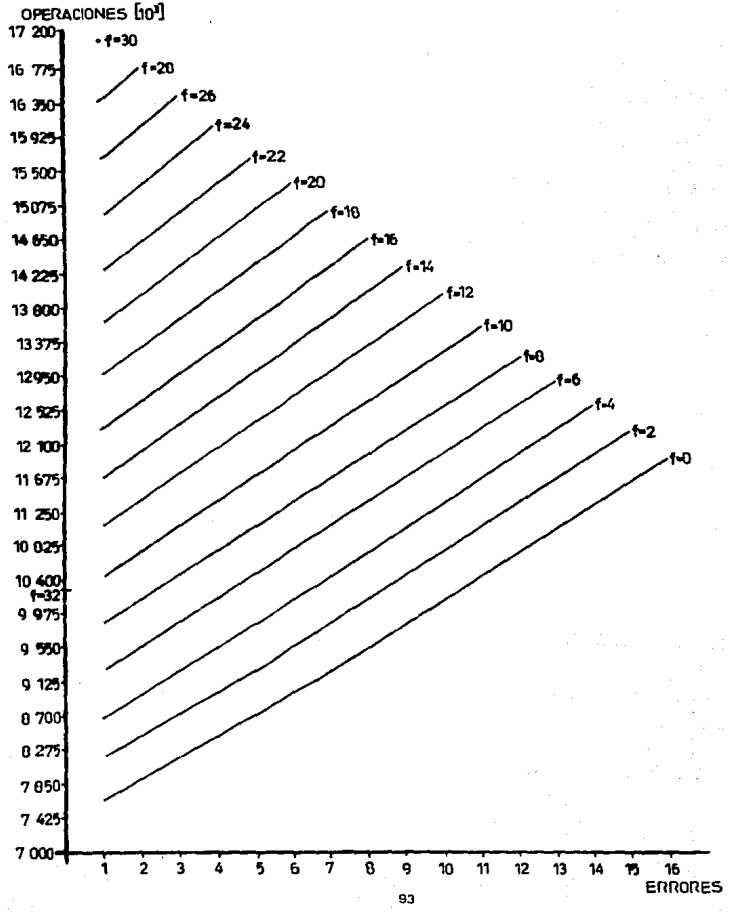
VALOR DE LAS ERRATAS (M)

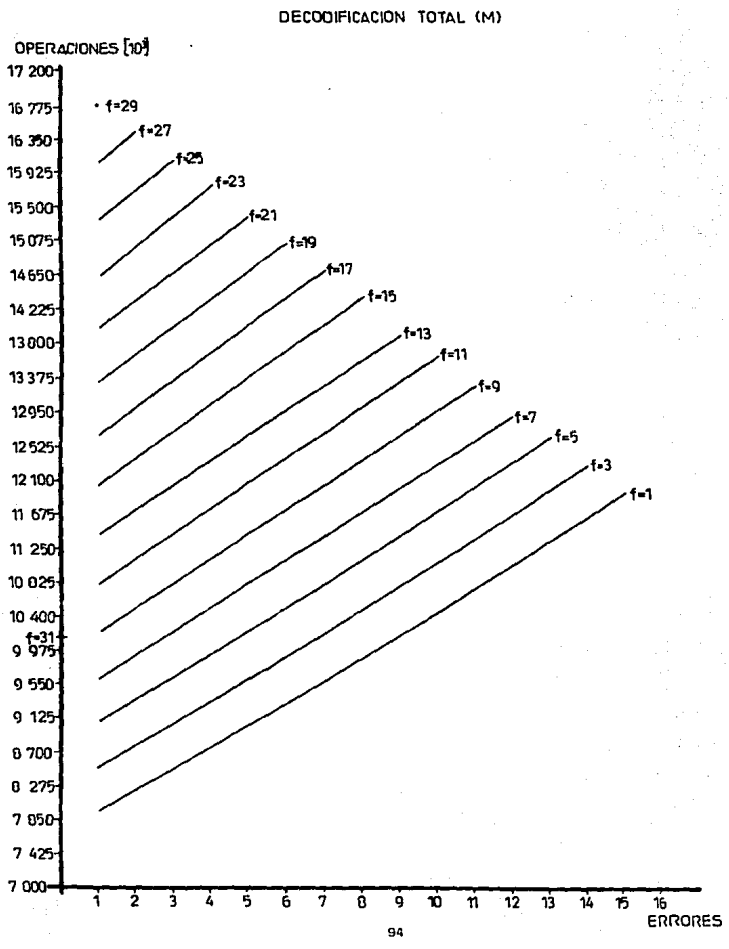


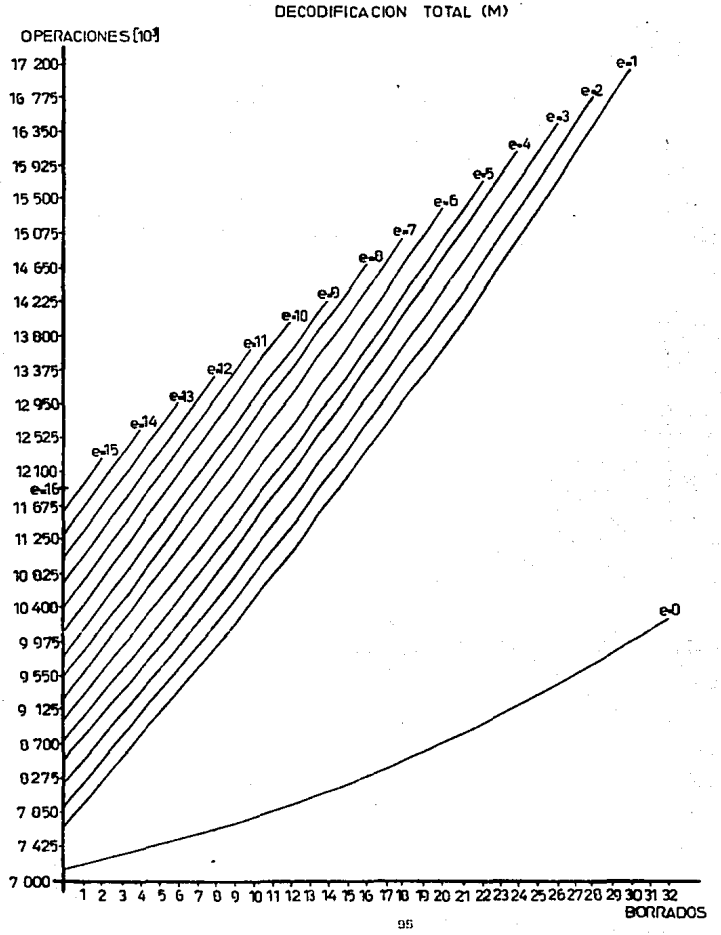
VALOR DE LAS ERRATAS (M)

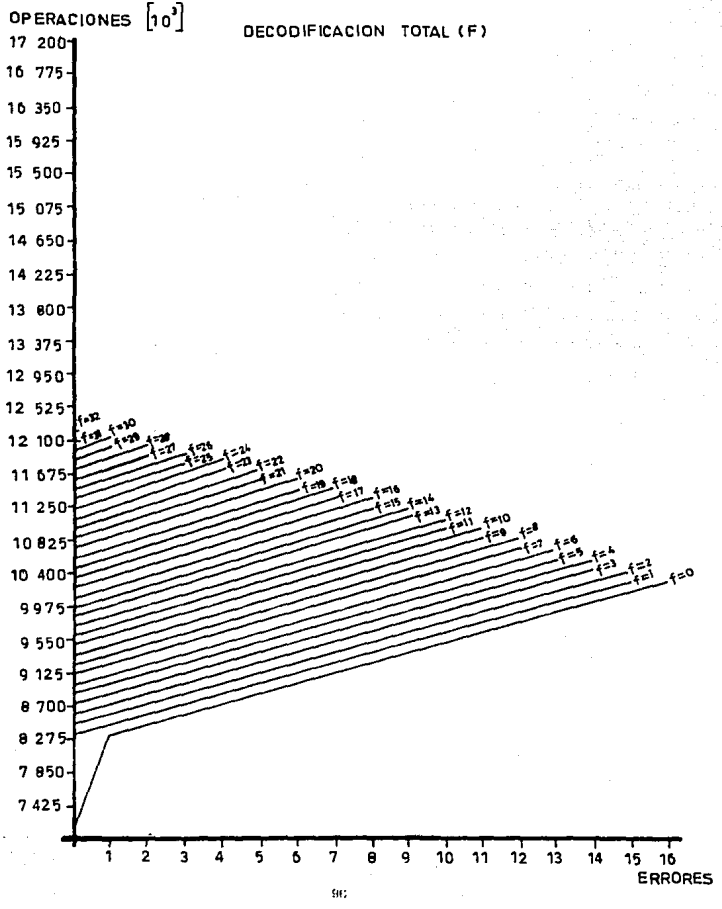


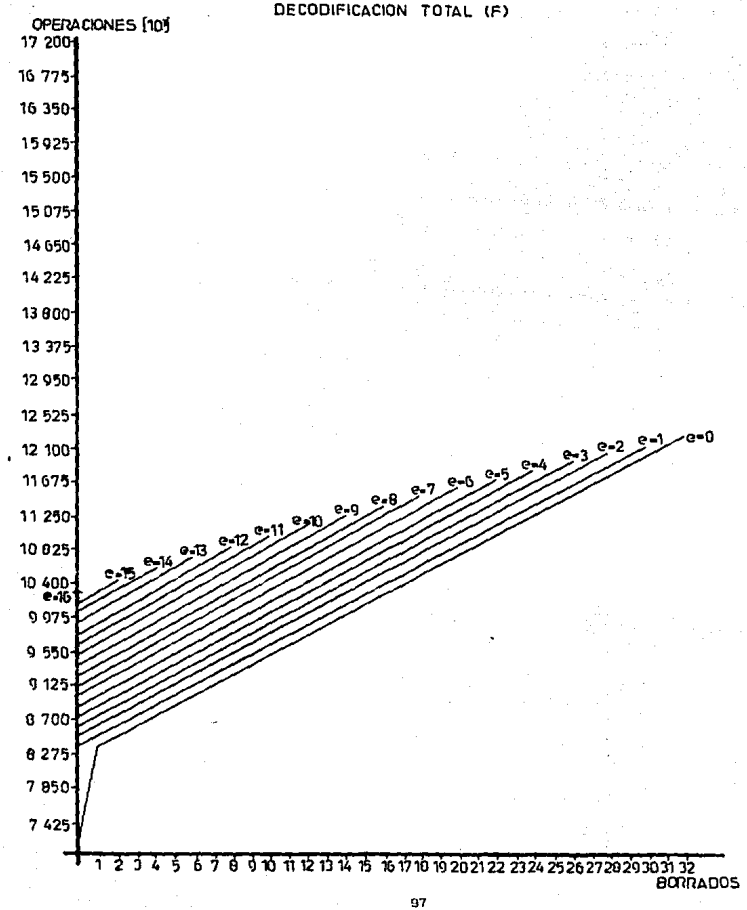
DECODIFICACION TOTAL (M)

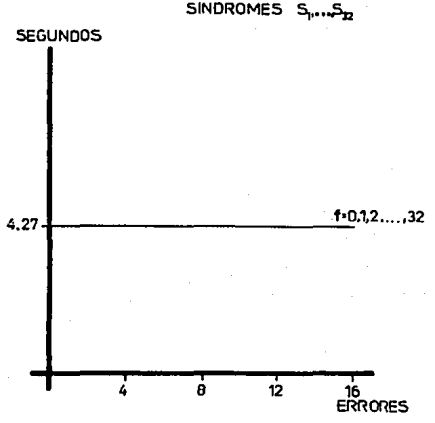


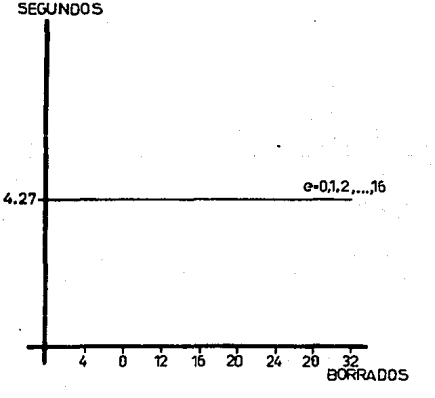

DECODIFICACION TOTAL (M)
OPERACIONES [10³]
17 200
16 775
16 350
15 925
15 500
15 075
14 650
14 225
13 800
13 375
12 950
12 525
12 100
11 675
11 250
10 825
10 400
9 975
9 550
9 125
8 700
8 275
7 850
7 425
7 000
f=29
f=27
f=25
f=23
f=21
f=19
f=17
f=15
f=13
f=11
f=9
f=7
f=5
f=3
f=1
f=31
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ERRORES


DECODIFICACION TOTAL (M)
OPERACIONES [10³]
17 200
16 775
16 350
15 925
15 500
15 075
14 650
14 225
13 800
13 375
12 950
12 525
12 100
11 675
11 250
10 825
10 400
9 975
9 550
9 125
8 700
8 275
7 850
7 425
7 000
e=1
e=2
e=3
e=4
e=5
e=6
e=7
e=8
e=9
e=10
e=11
e=12
e=13
e=14
e=15
e=16
e=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS

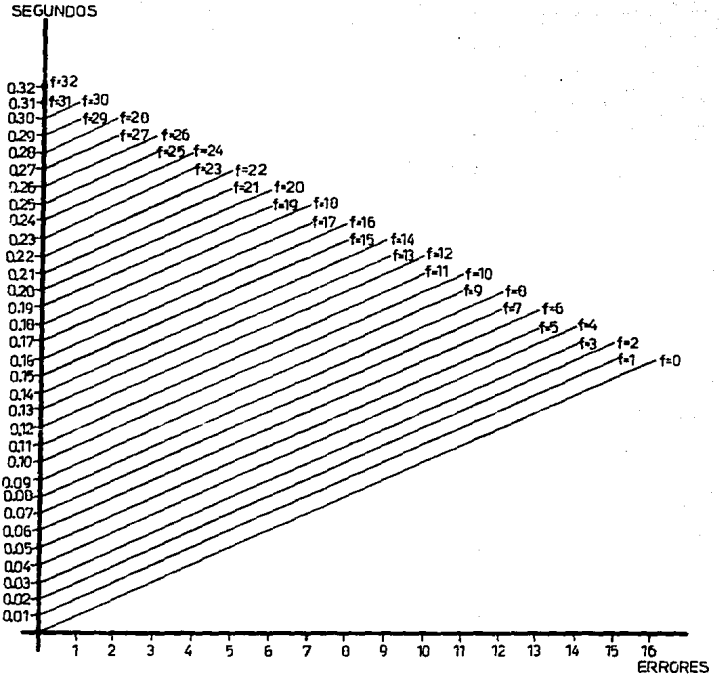

OPERACIONES [10³]
DECODIFICACION TOTAL (F)
17 200
16 775
16 350
16 925
15 500
15 075
14 650
14 225
13 800
13 375
12 950
12 525
12 100
11 675
11 250
10 825
10 400
9 975
9 550
9 125
8 700
8 275
7 850
7 425
f=32
f=31
f=30
f=29
f=28
f=27
f=26
f=25
f=24
f=23
f=22
f=21
f=20
f=19
f=18
f=17
f=16
f=15
f=14
f=13
f=12
f=11
f=10
f=9
f=8
f=7
f=6
f=5
f=4
f=3
f=2
f=1
f=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
ERRORES

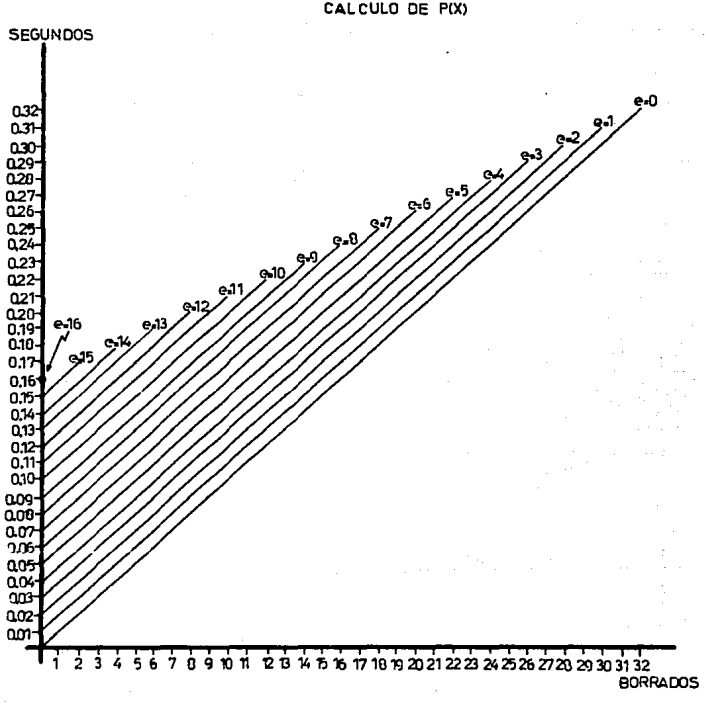

DECODIFICACION TOTAL (F)
OPERACIONES [10³]
17 200
16 775
16 350
15 925
15 500
15 075
14 650
14 225
13 800
13 375
12 950
12 525
12 100
11 675
11 250
10 825
10 400
9 975
9 550
9 125
8 700
8 275
7 850
7 425
e=16
e=15
e=14
e=13
e=12
e=11
e=10
e=9
e=8
e=7
e=6
e=5
e=4
e=3
e=2
e=1
e=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS


SINDROMES $S_1,\ldots,S_{12}$
SEGUNDOS
4.27
f=0,1,2....,32
4
8
12
16
ERRORES

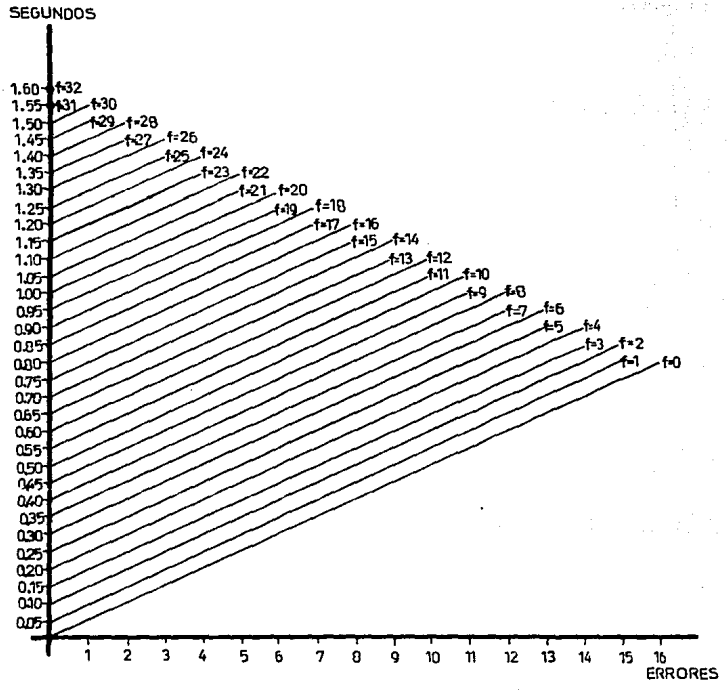

SEGUNDOS
4.27
e=0,1,2,...,16
4
8
12
16
20
24
28
32
BORRADOS

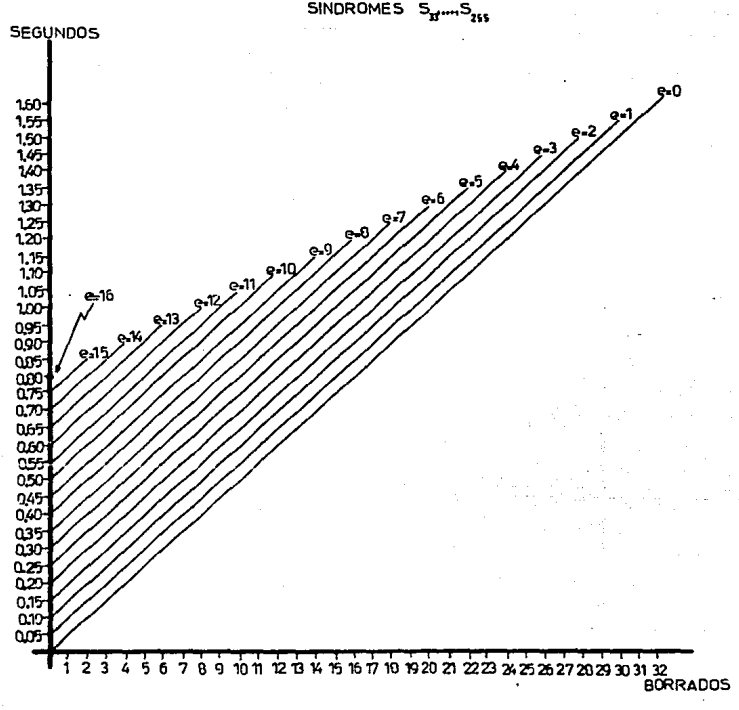
CALCULO DE P(X)



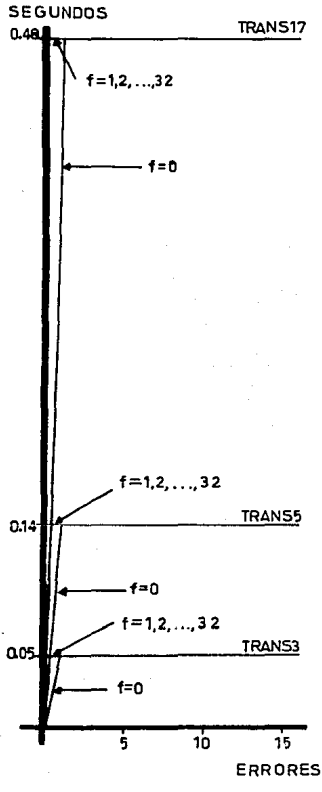

CALCULO DE P(X)
SEGUNDOS
0.32
0.31
0.30
0.29
0.28
0.27
0.26
0.25
0.24
0.23
0.22
0.21
0.20
0.19
0.10
0.17
0.16
0.15
0.14
0.13
0.12
0.11
0.10
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
e=16
e=15
e=14
e=13
e=12
e=11
e=10
e=9
e=8
e=7
e=6
e=5
e=4
e=3
e=2
e=1
e=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS

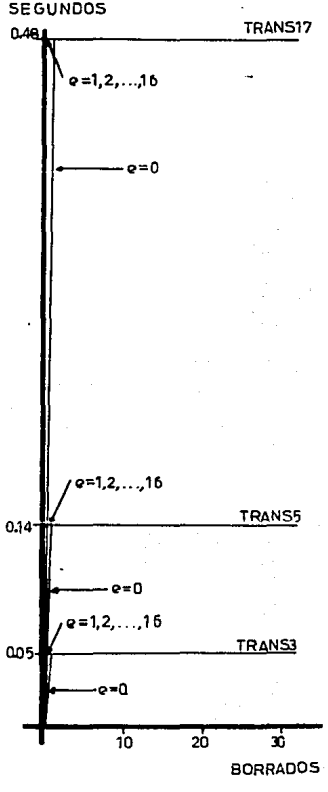
SINDROMES S_{33}.....S_{255}



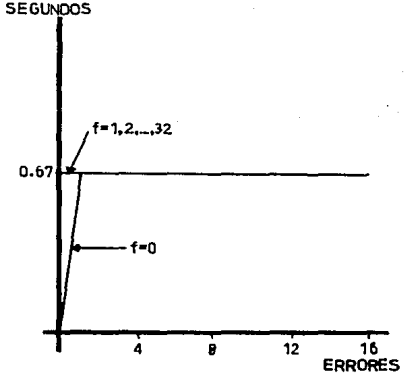

SINDROMES S_{17}....S_{255}
SEGUNDOS
1,60
1,55
1,50
1,45
1,40
1,35
1,30
1,25
1,20
1,15
1,10
1,05
1,00
0,95
0,90
0,85
0,80
0,75
0,70
0,65
0,60
0,55
0,50
0,45
0,40
0,35
0,30
0,25
0,20
0,15
0,10
0,05
e=16
e=15
e=14
e=13
e=12
e=11
e=10
e=9
e=8
e=7
e=6
e=5
e=4
e=3
e=2
e=1
e=0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 10 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS

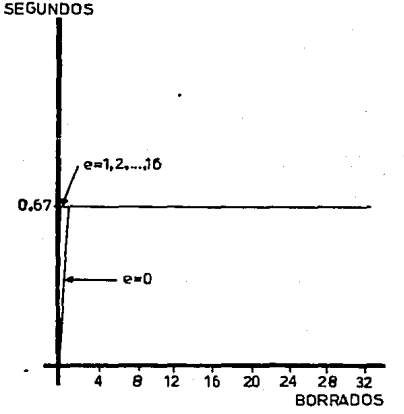
TRANSFORMADA DE 3,5 Y 17 PUNTOS



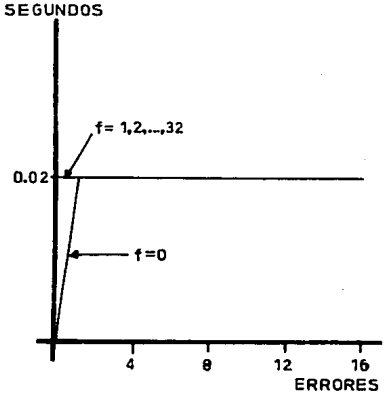


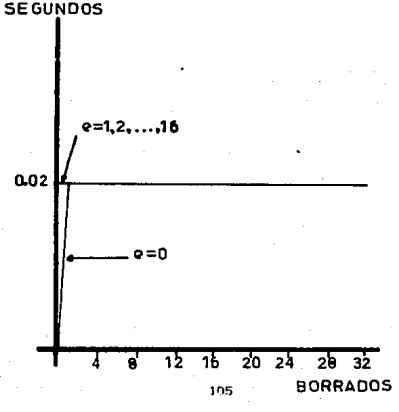
CALCULO DEL VALOR DE LAS ERRATAS

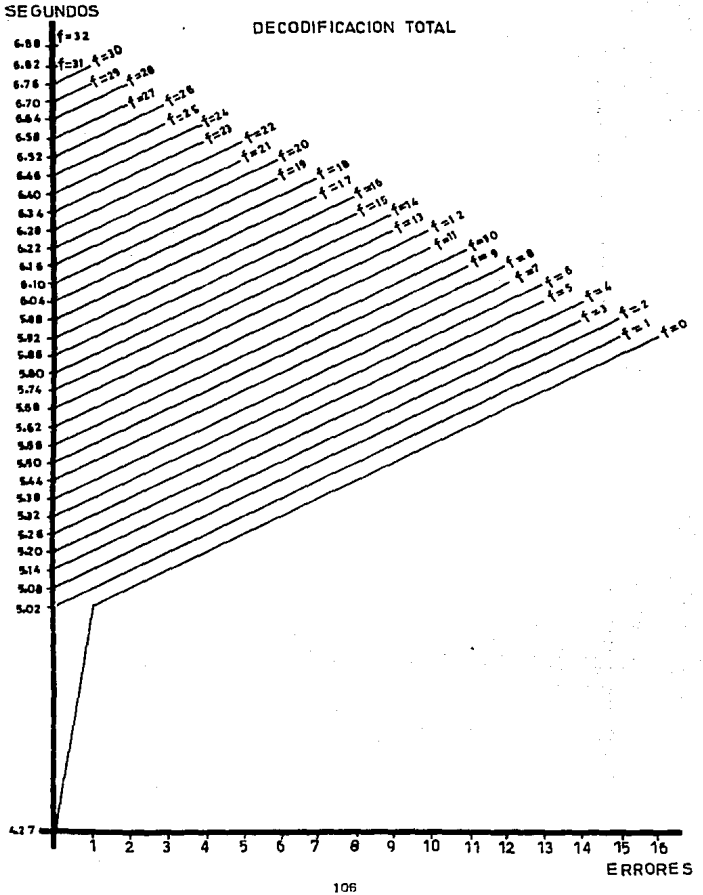


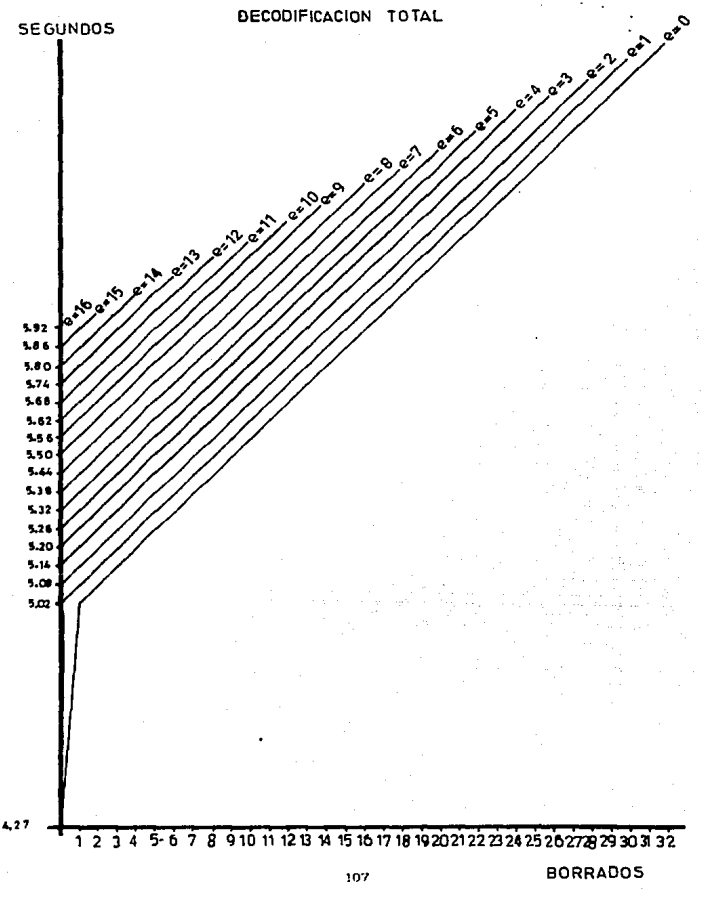


CORRECCION DE LA PALABRA RECIBIDA






SEGUNDOS
DECODIFICACION TOTAL
6.88
6.82
6.76
6.70
6.64
6.58
6.52
6.46
6.40
6.34
6.28
6.22
6.16
6.10
6.04
5.98
5.92
5.86
5.80
5.74
5.68
5.62
5.56
5.50
5.44
5.38
5.32
5.26
5.20
5.14
5.08
5.02
4.27
f=32
f=31
f=30
f=29
f=28
f=27
f=26
f=25
f=24
f=23
f=22
f=21
f=20
f=19
f=18
f=17
f=16
f=15
f=14
f=13
f=12
f=11
f=10
f=9
f=8
f=7
f=6
f=5
f=4
f=3
f=2
f=1
f=0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
ERRORES


DECODIFICACION TOTAL
SEGUNDOS
e=16
e=15
e=14
e=13
e=12
e=11
e=10
e=9
e=8
e=7
e=6
e=5
e=4
e=3
e=2
e=1
e=0
5.92
5.86
5.80
5.74
5.68
5.62
5.56
5.50
5.44
5.38
5.32
5.26
5.20
5.14
5.08
5.02
4.27
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
BORRADOS

III.3 CONCLUSIONES DE LA SIMULACION.

Como se puede observar en las tablas y gráficas anteriores, el algoritmo que trabaja en el dominio de la frecuencia tiende a ser el más rápido de los tres conforme aumenta el número de errores y para una distancia d tal que:

$$d_{min} >= d >= 2e + f + 1$$

donde $d_{min} = 33$ y para el número de errores y borrados siguientes:

e	f
6	0,1,2
7	0,1,2,3
8	0,1,2,3,4,5
9	0,1,2,3,4,5,6,7
10	0,1,2,3,4,5,6,7,8,9,10,11,12
11	0,1,2,3,4,5,6,7,8,9,10
12	0,1,2,3,4,5,6,7,8
13	0,1,2,3,4,5,6
14	0,1,2,3,4
15	0,1,2
16	0

Para un número de errores menor a 6 los algoritmos clásico y modificado son un poco más rápidos que el algoritmo que trabaja en el dominio de la frecuencia pero la diferencia no es muy significativa. Debido a esto y a que en la realidad se presentan más errores que borrados se seleccionará como el mejor algoritmo de decodificación el que trabaja en el dominio de la frecuencia.

De aquí en adelante nos basaremos en este algoritmo.

CAPITULO IV. DISEÑO DE UN DECODIFICADOR PARA CODIGOS REED-SOLOMON CON ARQUITECTURA EN PARALELO.

En este capitulo se presenta el diseño de un decodificador de códigos R.S. con arquitectura en paralelo, para el algoritmo de decodificación que trabaja en el dominio de la frecuencia, ya que resultó ser el más rápido de los tres algoritmos estudiados. Este diseño comprende todas las etapas de la decodificación, no así el diseño del controlador de este proceso, pues no era uno de los objetivos del trabajo.

La paralelización del algoritmo se realizó en base al análisis hecho primero a nivel de grandes bloques, o etapas del algoritmo de decodificación y luego a nivel de subtareas dentro de cada una de esas etapas, obteniéndose así la paralelización completa del algoritmo a nivel de subtareas y a nivel de grandes bloques. Esta paralelización se hizo buscando que el decodificador fuera lo más rápido posible.

IV.1 DIAGRAMA DE BLOQUES DEL PROCESO DE DECODIFICACION.

En primer lugar se muestra un diagrama de bloques en el que se ve cómo se va ejecutando el algoritmo de decodificación por etapas y las señales de control de cada una de ellas.

En este diagrama los bloques que se encuentran uno sobre otro indican que estas etapas inician su ejecución paralelamente. Los bloques que se encuentran uno sobre otro, pero con un defasamiento entre sí, indican que cierto tiempo después de iniciar la ejecución de la etapa del primer bloque

comienza la ejecución de la etapa del siguiente bloque, o sea que, algunas de las etapas del algoritmo pueden comenzar con resultados parciales de una etapa anterior. Los bloques que se encuentran uno después de otro indican que estas etapas se ejecutan en ese orden secuencialmente. El diagrama es el siguiente.

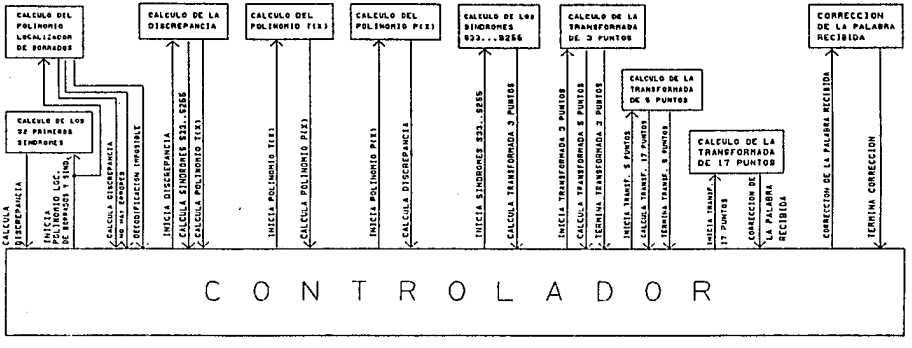


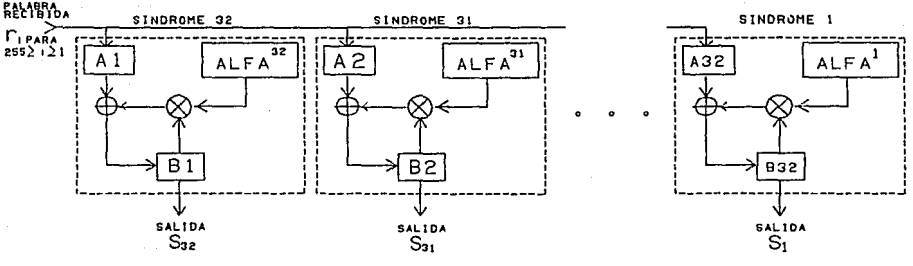
DIAGRAMA DE BLOQUES DEL PROCESO DE DECODIFICACION

IV.2 DIAGRAMAS DE FLUJO DE LOS CIRCUITOS DEL DECODIFICADOR DISEÑADO.

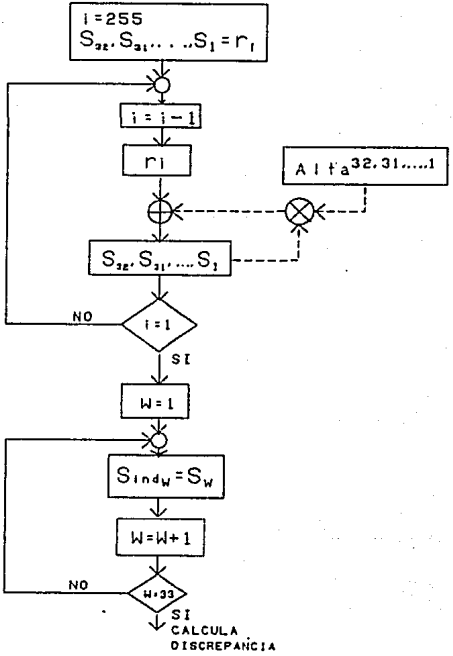
En segundo lugar, como material de refuerzo en la comprensión del funcionamiento de cada circuito, se realizó un diagrama de flujo para cada uno de ellos en donde se muestra qué cálculos realizan y también se puede observar qué se hace en paralelo y qué secuencialmente.

Además, los nombres de las variables usadas en estos diagramas corresponden a los nombres de las variables utilizadas en la simulación de los algoritmos. Los diagramas son los siguientes.

CALCULO DE LOS
32 PRIMEROS
SINDROMES

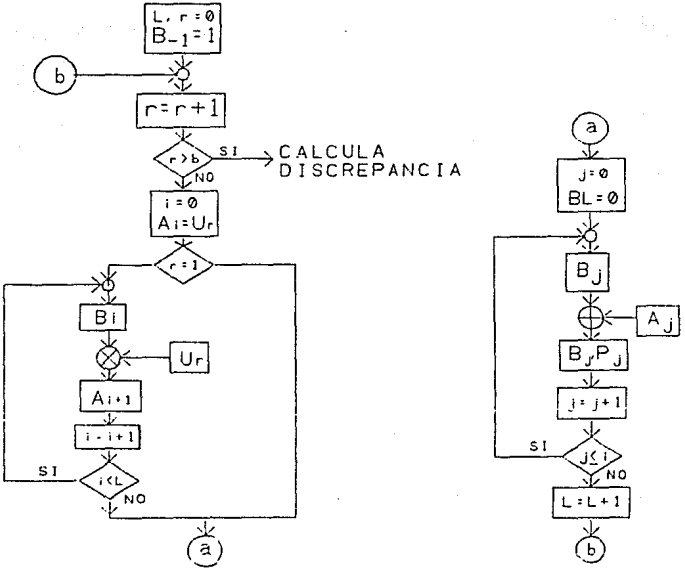


CALCULO DE LOS 32 PRIMEROS SINDROMES Y SU ALMACENAMIENTO EN EL VECTOR Sind

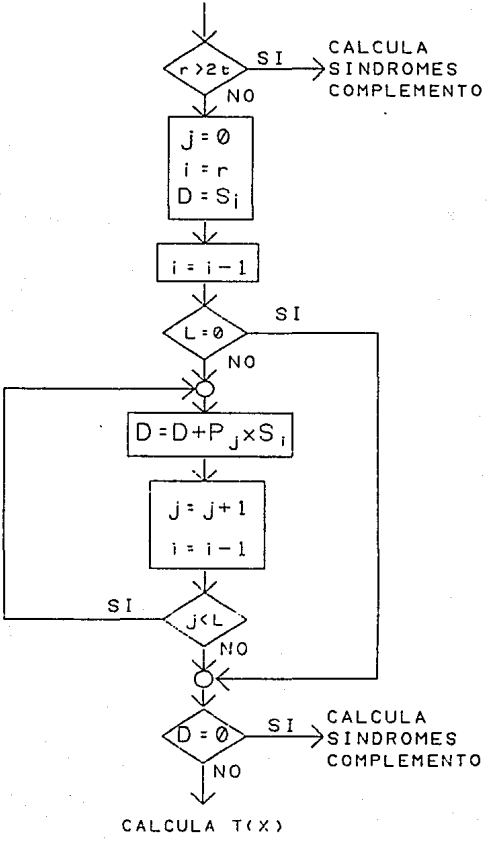


CALCULO DEL POLINOMIO

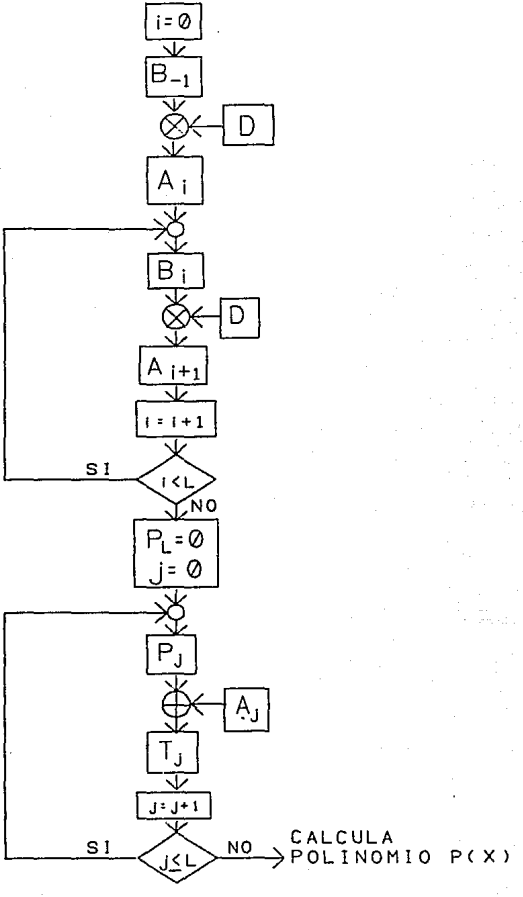
LOCALIZADOR DE BORRADOS



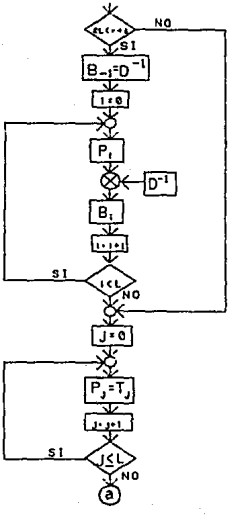
CALCULO DE LA DISCREPANCIA

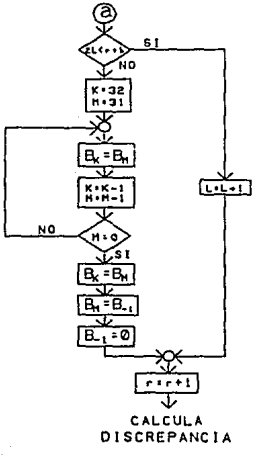


CALCULO DEL POLINOMIO T(X)



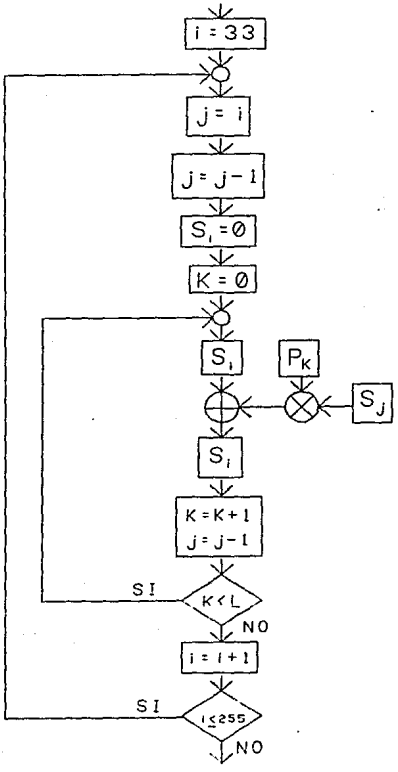
CALCULO DEL POLINOMIO P(X)





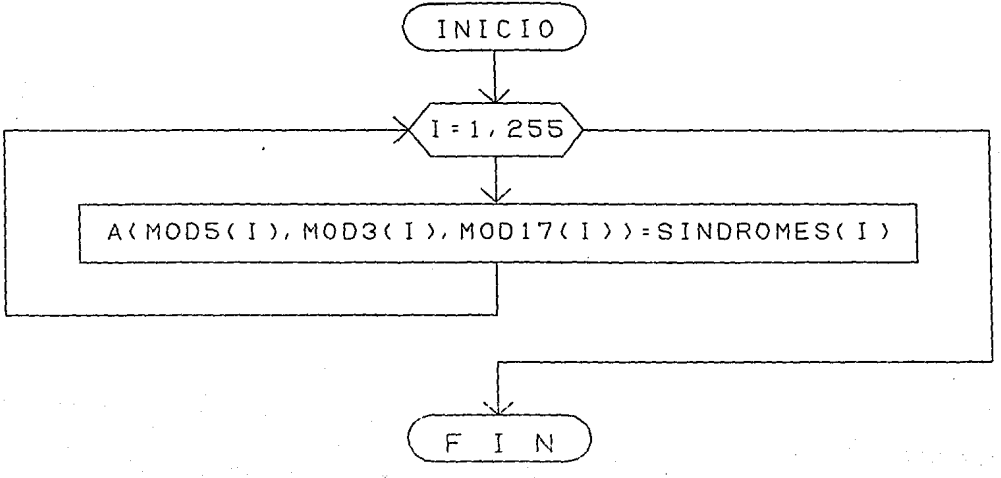
CALCULO DE LOS SINDROMES

$S_{33} \ldots S_{255}$

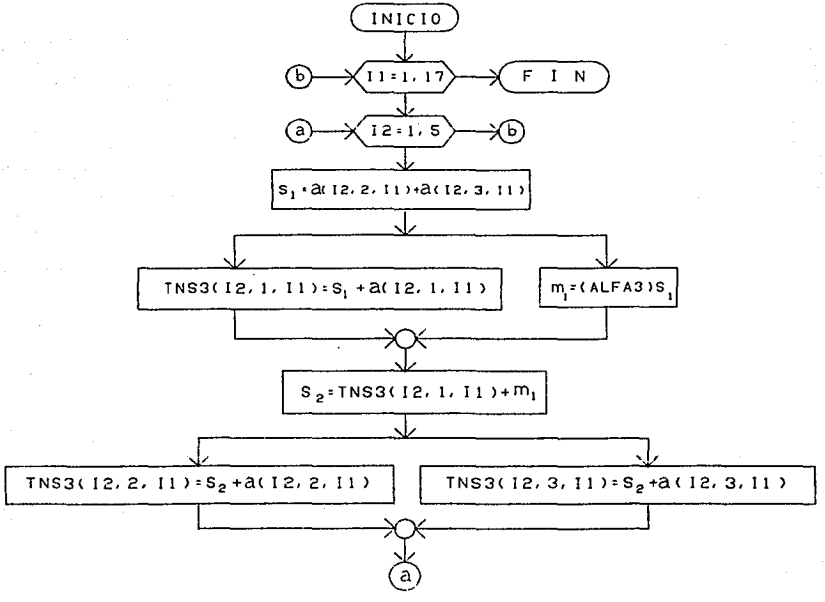


CALCULO DE LA TRANSFORMADA

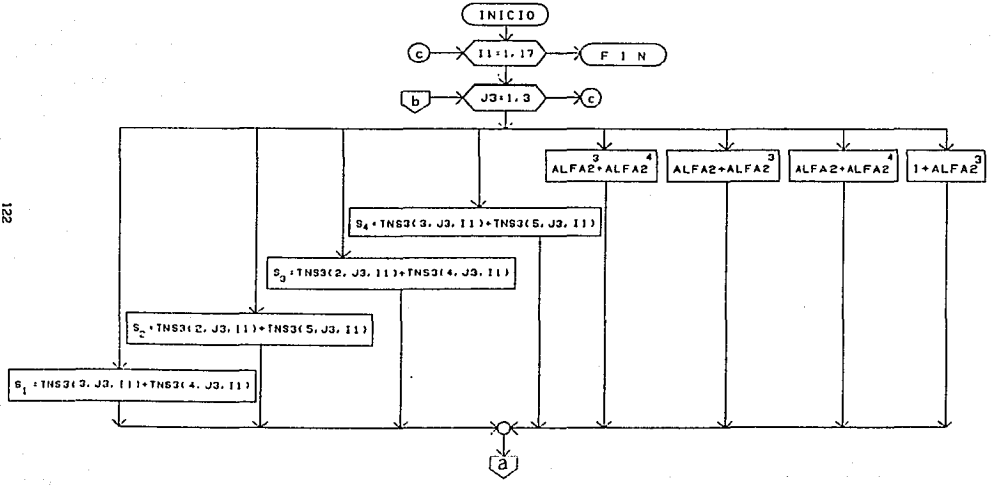
PASO DEL VECTOR DE SINDROMES A MATRIZ DE TRES DIMENSIONES

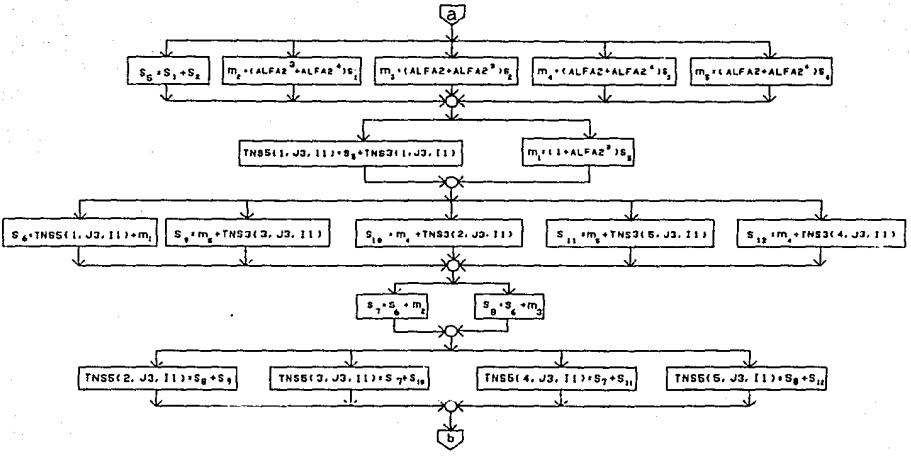


CALCULO DE LA TRANSFORMADA DE 3 PUNTOS

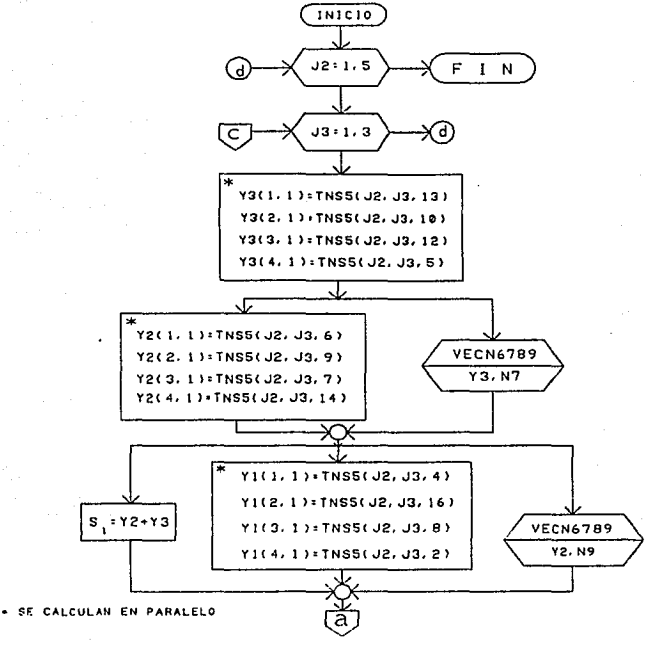


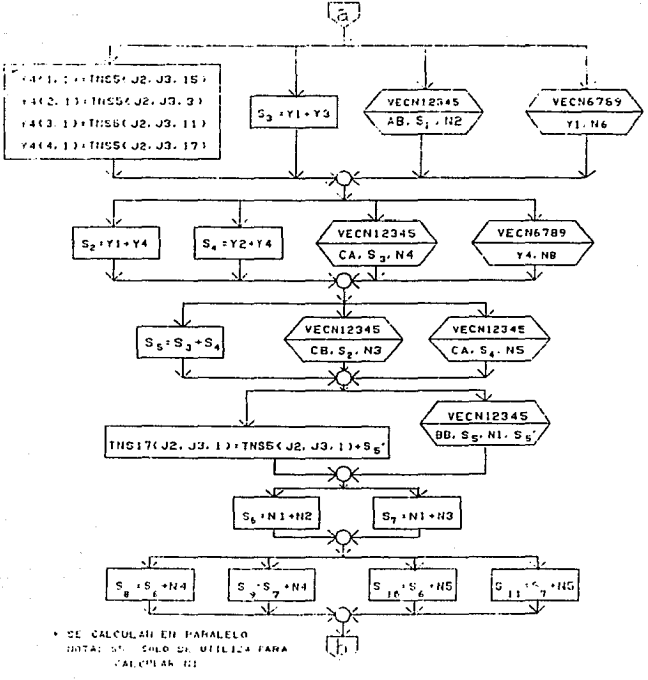
CALCULO DE LA TRANSFORMADA DE 5 PUNTOS



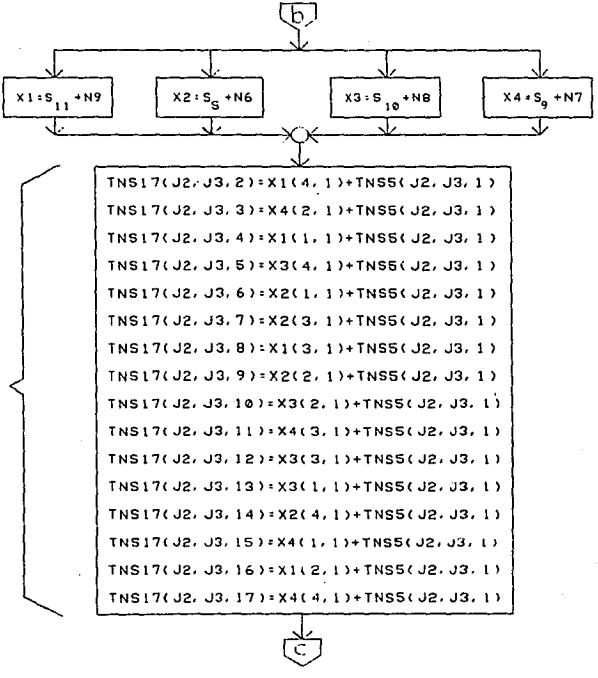

a
S6 + S1 + S4
m2 + (ALFA2^3 + ALFA2^4)S1
m3 + (ALFA2 + ALFA2^3)S2
m4 + (ALFA2 + ALFA2^4)S3
m5 + (ALFA2 + ALFA2^4)S4
TNS5(1, J3, I1) + S8 + TNS3(1, J3, I1)
m1 + (1 + ALFA2^3)S5
S6 + TNS5(1, J3, I1) + m1
S9 + m6 + TNS3(3, J3, I1)
S10 + m4 + TNS3(2, J3, I1)
S11 + m5 + TNS3(5, J3, I1)
S12 + m4 + TNS3(4, J3, I1)
S7 + S6 + m2
S8 + S4 + m3
TNS5(2, J3, I1) + S8 + S9
TNS5(3, J3, I1) + S7 + S10
TNS5(4, J3, I1) + S7 + S11
TNS5(5, J3, I1) + S8 + S12
b

CALCULO DE LA TRANSFORMADA DE 17 PUNTOS

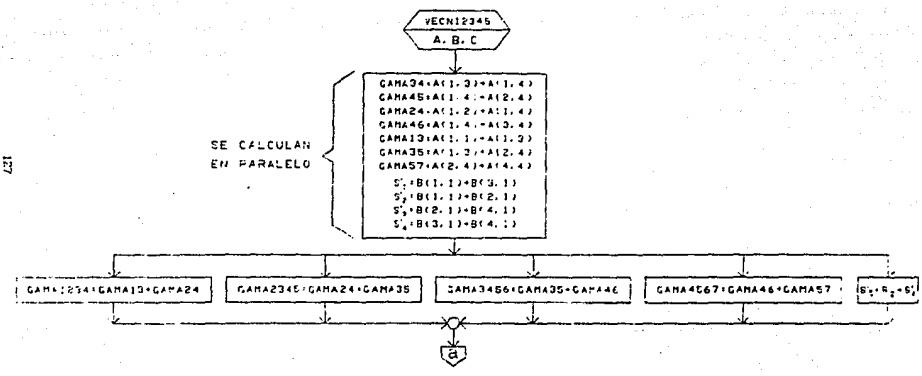


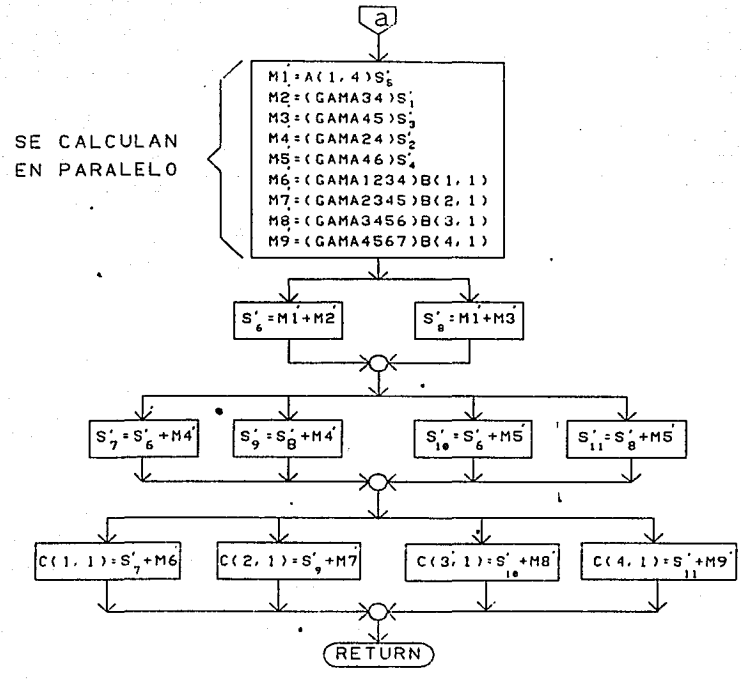

a
Y4(1,1)=TNS5(J2,J3,15)
Y4(2,1)=TNS5(J2,J3,3)
Y4(3,1)=TNS6(J2,J3,11)
Y4(4,1)=TNS5(J2,J3,17)
S3=Y1+Y3
VECN12345
AB, S1, N2
VECN6789
Y1, N6
S2=Y1+Y4
S4=Y2+Y4
VECN12345
CA, S3, N4
VECN6789
Y4, N8
S5=S3+S4
VECN12345
CB, S2, N3
VECN12345
CA, S4, N5
VECN12345
BB, S5, N1, S5'
TNS17(J2,J3,1)+TNS5(J2,J3,1)+S5'
S6=N1+N2
S7=N1+N3
S8=S6+N4
S9=S7+N4
S10=S6+N5
S11=S7+N5
J1
* SE CALCULAN EN PARALELO
DATA SE SOLO SE UTILIZA PARA
CALCULAR J1

SE CALCULAN
EN PARALELO

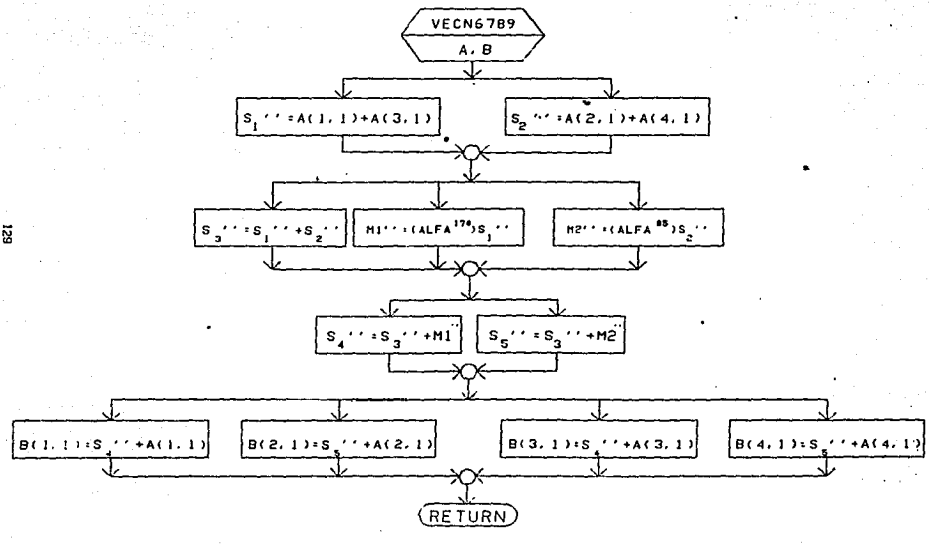


SUBRUTINA VECN12345

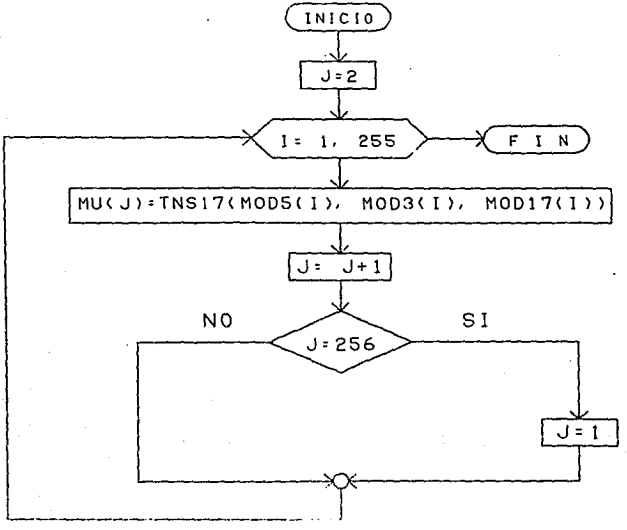



a
SE CALCULAN
EN PARALELO
M1'=A(1,4)S'6
M2'=(GAMA34)S'1
M3'=(GAMA45)S'3
M4'=(GAMA24)S'2
M5'=(GAMA46)S'4
M6'=(GAMA1234)B(1,1)
M7'=(GAMA2345)B(2,1)
M8'=(GAMA3456)B(3,1)
M9'=(GAMA4567)B(4,1)
S'6=M1'+M2'
S'8=M1'+M3'
S'7=S'6+M4'
S'9=S'8+M4'
S'10=S'6+M5'
S'11=S'8+M5'
C(1,1)=S'7+M6'
C(2,1)=S'9+M7'
C(3,1)=S'10+M8'
C(4,1)=S'11+M9'
RETURN

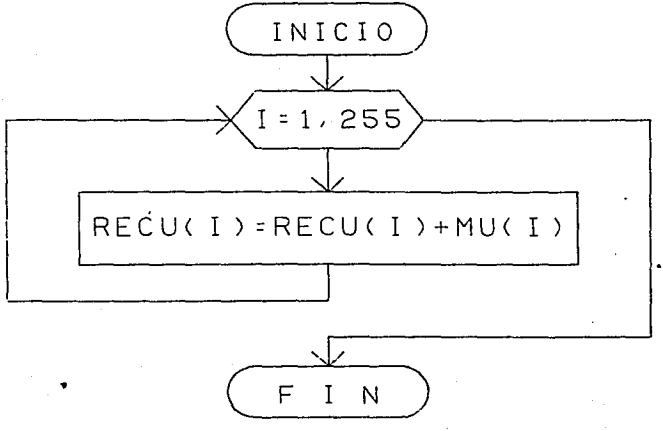
SUBRUTINA VECN6789



PASO DE LA MATRIZ DE TRES DIMENSIONES TNS17 AL VECTOR DE ERRATAS MU



CORRECCION DE LA PALABRA RECIBIDA



IV.3 CIRCUITOS DEL DECODIFICADOR DISEÑADO.

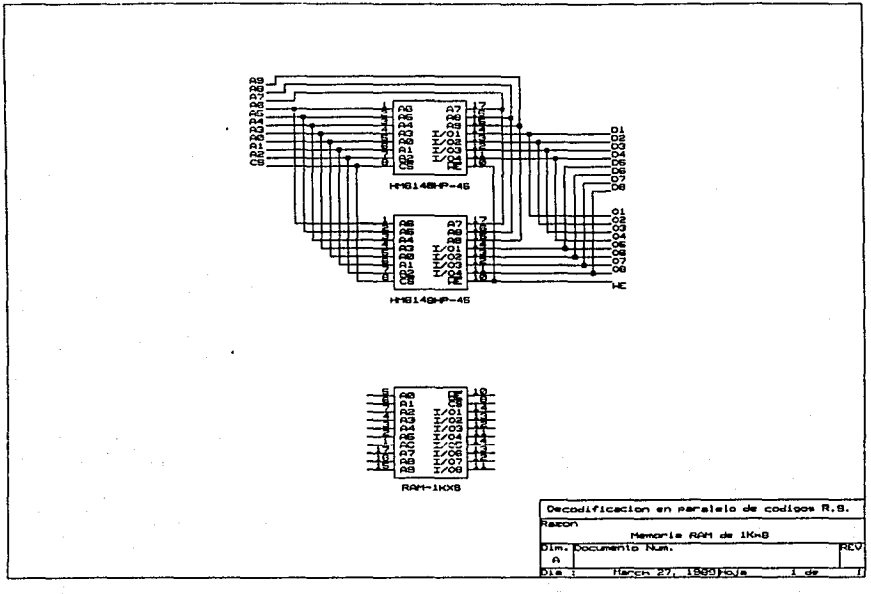
En tercer lugar se muestra el diseño del decodificador en base a elementos discretos. Primero se presenta el alambrado de los elementos que fueron representados como un sólo bloque o "chip" por facilidad en el diseño de las etapas del decodificador y después se muestra el circuito diseñado para cada una de estas etapas.

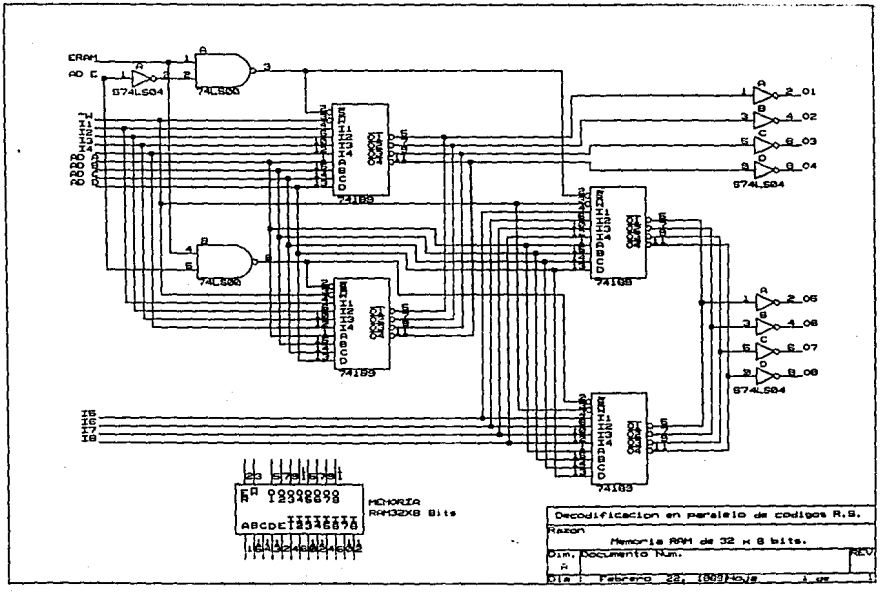
Nótese que algunos elementos están marcados con un *, lo que significa que se usaron en alguna etapa anterior y que es el mismo elemento.

Se considera que las memorias PROM utilizadas en el diseño se encuentran ya grabadas con su respectiva información cada una.

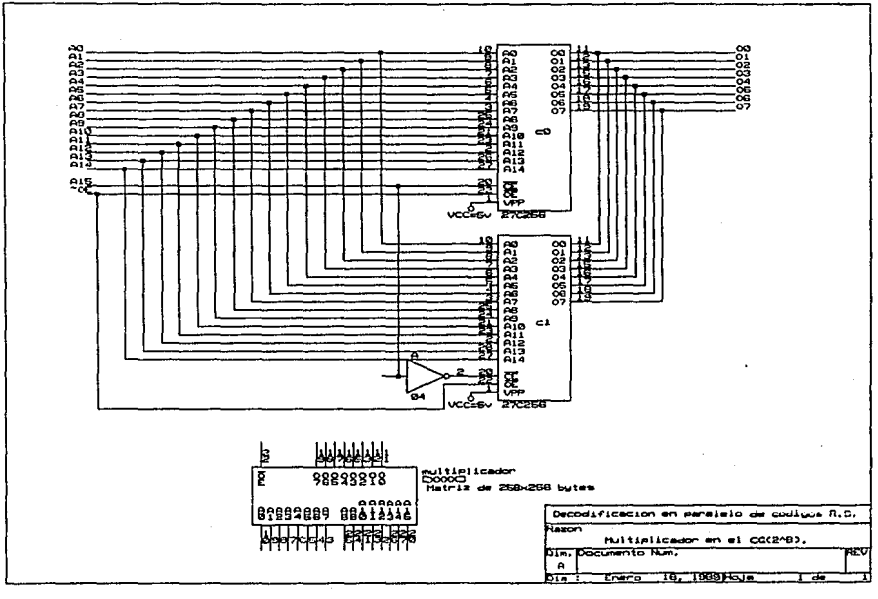
En cada uno de los circuitos aparecen ciertos nombres que corresponden a las señales de control del circuito y que posteriormente aparecen en los diagramas de control, además en algunos elementos se encuentra otro nombre que indica lo que ese elemento calcula, o en el caso de las memorias indica lo que contienen.

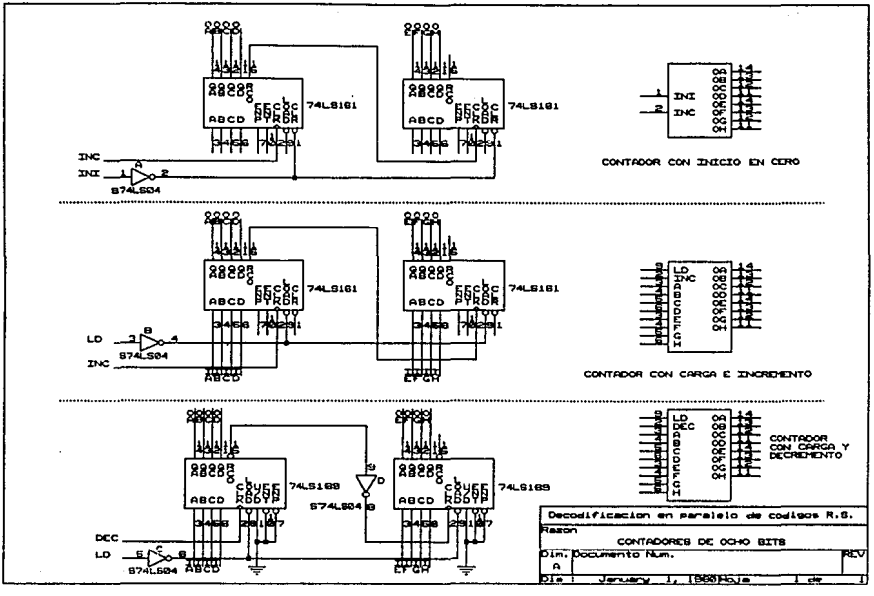
A continuación se muestran los circuitos diseñados para el decodificador.

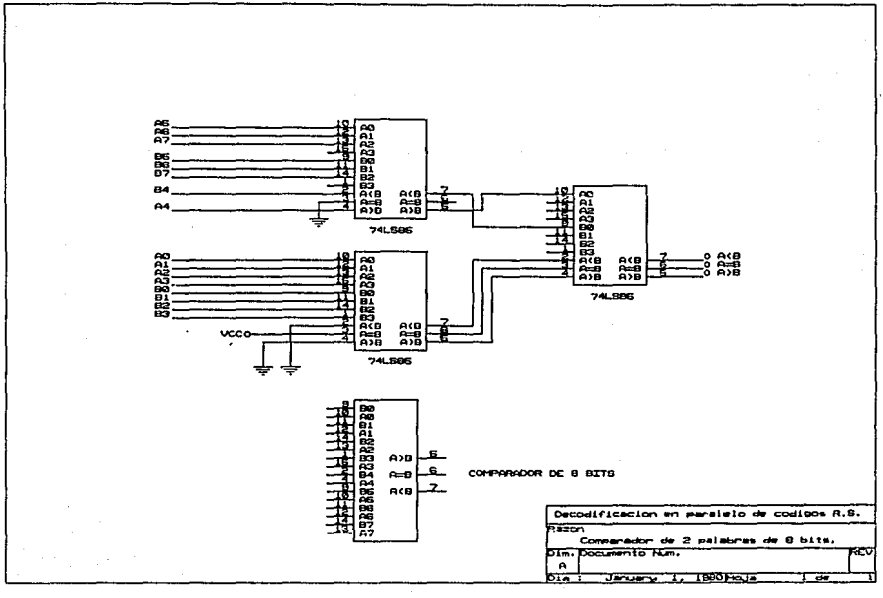

HM6148HP-45
HM6148HP-45
RAM-1KX8
Decodificacion en paralelo de codigos R.S.
Razon
Memoria RAM de 1Kx8
Dim.
A
Documento Num.
REV
Dia :
March 27, 1990
Hoja
1 de

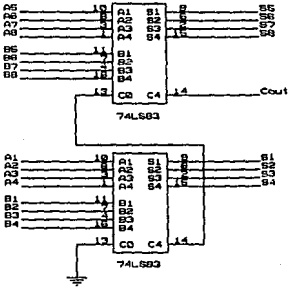

CRAM
AD C
74LS04
74LS00
74189
74LS04
Decodificacion en paralelo de codigos R.S.
Memoria RAM de 32 x 8 bits.
Documento Num.
Hoja
Dia
Febrero 22, 1989

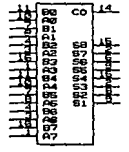
MCM93422
MCM93422
RAM DE 256 PALABRAS DE 8 BITS
Decodificacion en paralelo de codigos R.S.
RAM DE 256 PALABRAS DE 8 BITS.
Dim. Documento Num.
REV
A
Dia : January 1, 1900 Hoja 1 de 1


Multiplicador
Matriz de 256x256 bytes
Decodificacion en paralelo de codigos R.D.
Razon
Multiplicador en el CC(2^8).
Dim. Documento Num.
REV
A
Dia : Enero 16, 1988 Hoja 1 de 1


74LS161
74LS04
74LS169
INC
INI
LD
DEC
CONTADOR CON INICIO EN CERO
CONTADOR CON CARGA E INCREMENTO
CONTADOR CON CARGA Y DECREMENTO
Decodificacion en paralelo de codigos R.S.
CONTADORES DE OCHO BITS

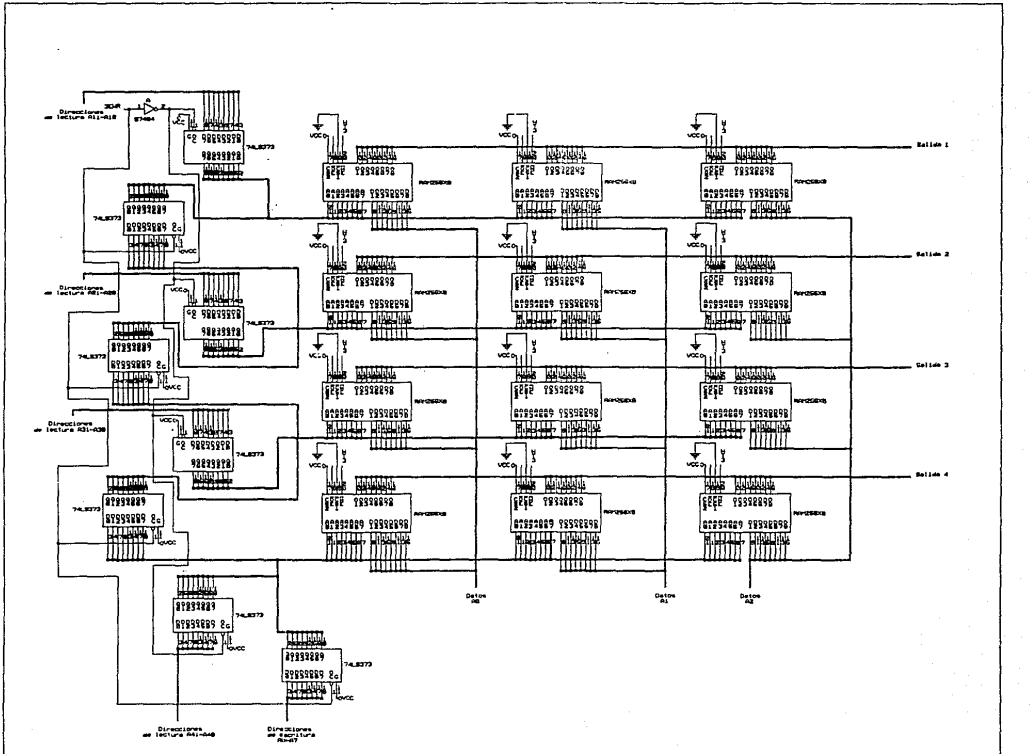

74LS86
74LS86
74LS86
VCCO
COMPARADOR DE 8 BITS
Decodificacion en paralelo de codigos R.S.
Comparador de 2 palabras de 8 bits.
Documento Num:
Hoja 1 de 1

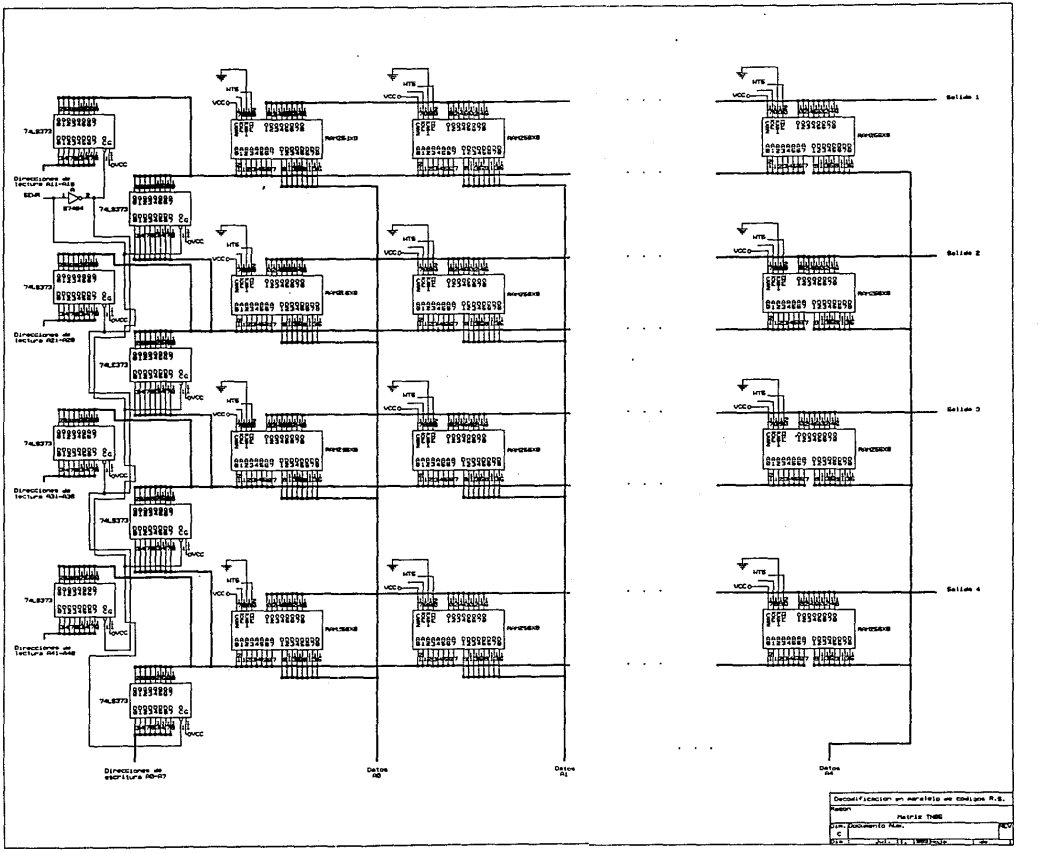




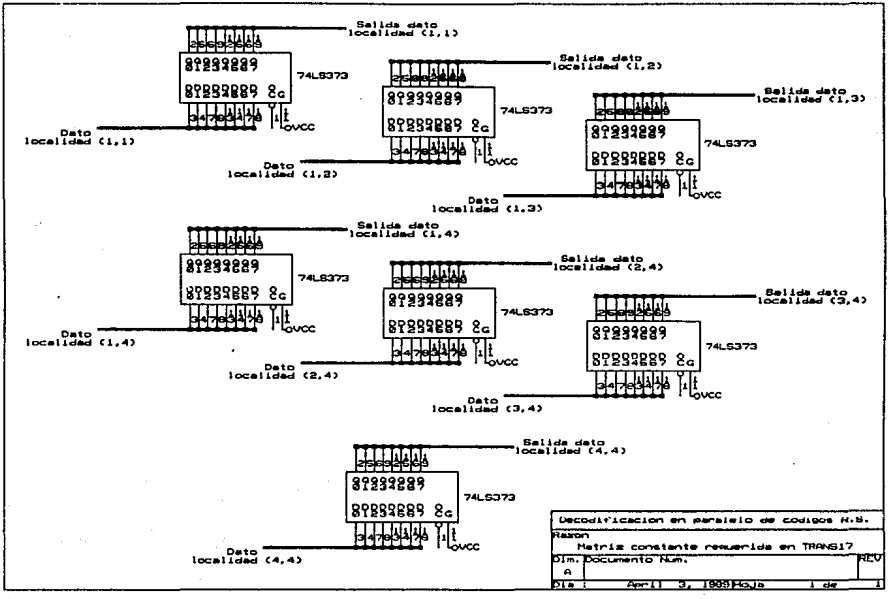
SUMADOR COMPLETO DE OCHO BITS

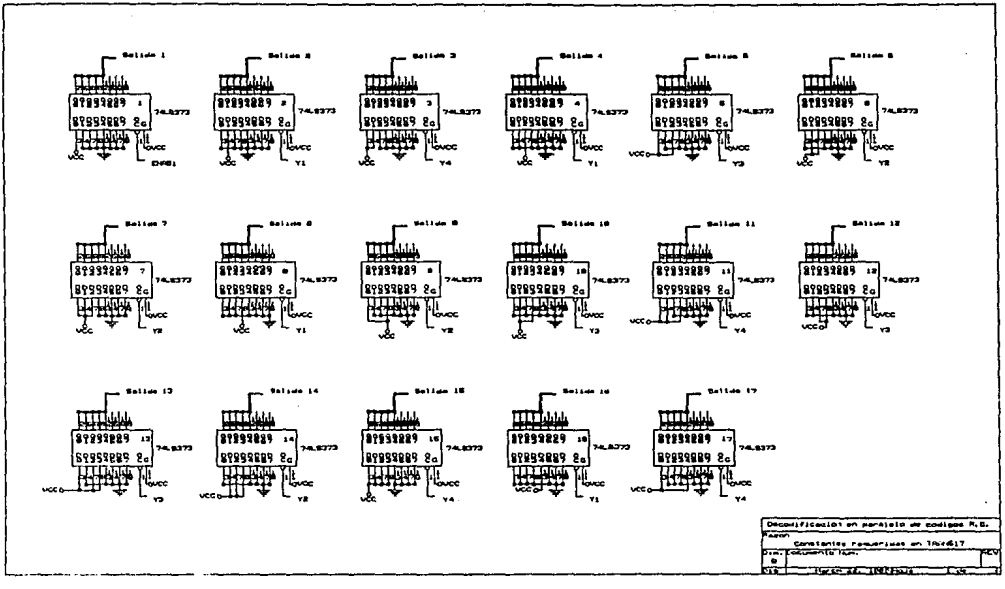
Decodificacion en paralelo de codigos R.S.			
Razon: SUMADOR DE DOS PALABRAS DE OCHO BITS			
Dim. A	Documento Num.		REV
Dia : January 1, 1900	Hoja	de	

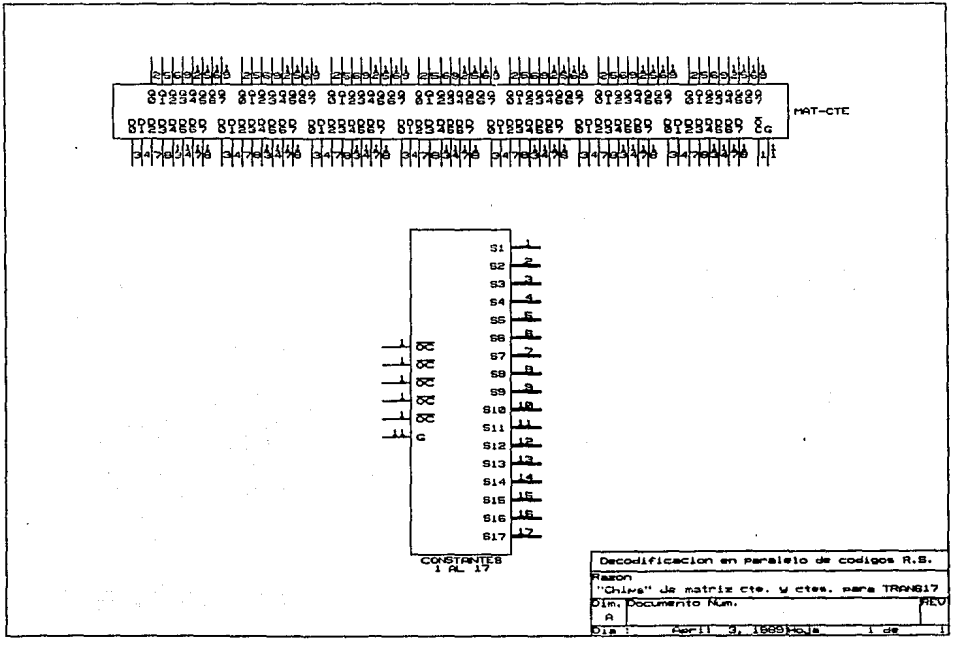

Salida 1
Salida 2
Salida 3
Salida 4
Datos A0
Datos A1
Datos A2
Direcciones de lectura A11-A19
Direcciones de lectura A21-A29
Direcciones de lectura A31-A39
Direcciones de lectura A41-A49
Direcciones de escritura A5-A9
74LS373
74LS138
74LS04
VCC
Decodificacion en paralelo de codigos R.S.
Matrix THB3


Salida 1
Salida 2
Salida 3
Salida 4
Datos A0
Datos A1
Datos A4
Direcciones de escritura A0-A7
Decodificacion en paralelo de Codigos R.S.
Master THESE

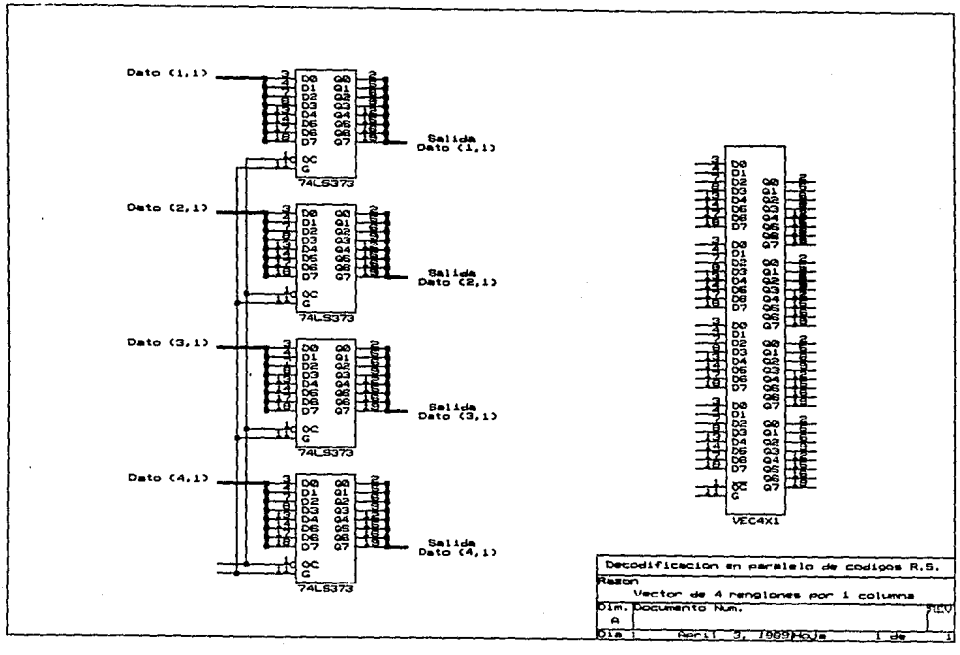
MAT-TNS5
MAT-TNS3
Decodificacion en paralelo de codigos R.S.
Razon
"Chips" de las matrices TNS3 y TNS5
Dim.
Documento Num.
REV
A
Dia
Hoja
1 de

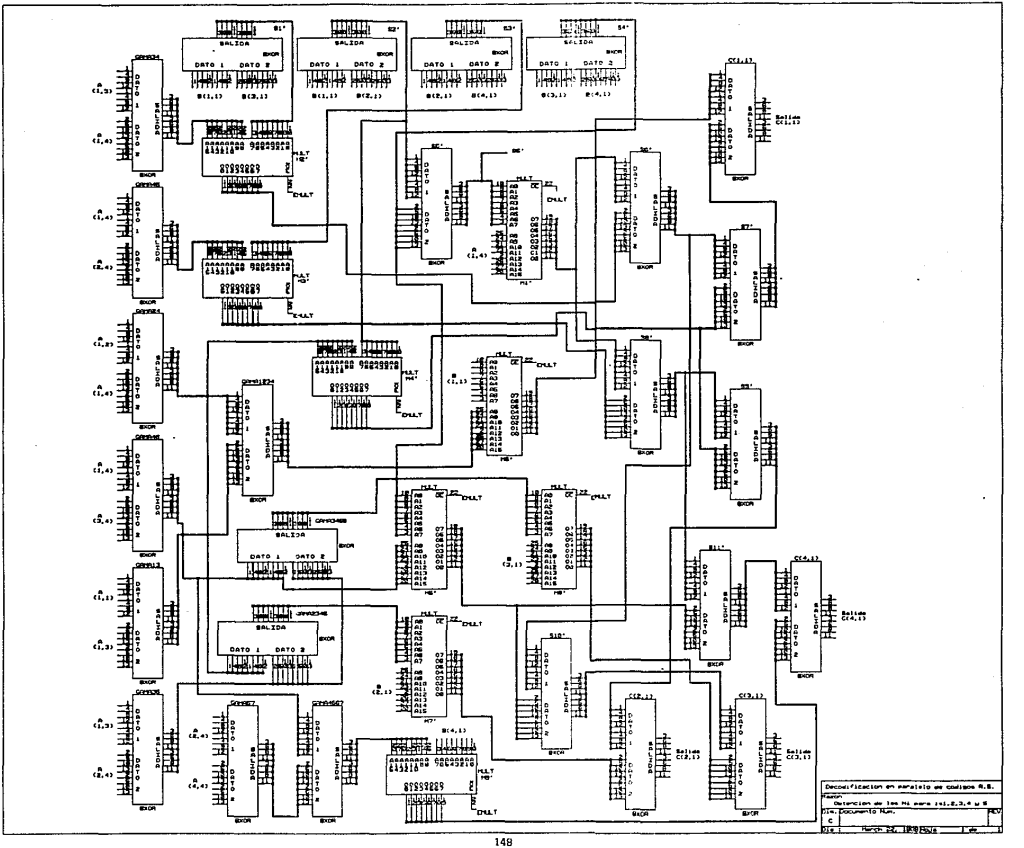

Salida dato localidad (1,1)
Salida dato localidad (1,2)
Salida dato localidad (1,3)
Salida dato localidad (1,4)
Salida dato localidad (2,4)
Salida dato localidad (3,4)
Salida dato localidad (4,4)
74LS373
Dato localidad (1,1)
Dato localidad (1,2)
Dato localidad (1,3)
Dato localidad (1,4)
Dato localidad (2,4)
Dato localidad (3,4)
Dato localidad (4,4)
+5VCC
Decodificacion en paralelo de codigos R.S.
Matriz constante resuelta en TRANS17
Documento Num.
REV
Dia : April 3, 1989 Hoja 1 de 1

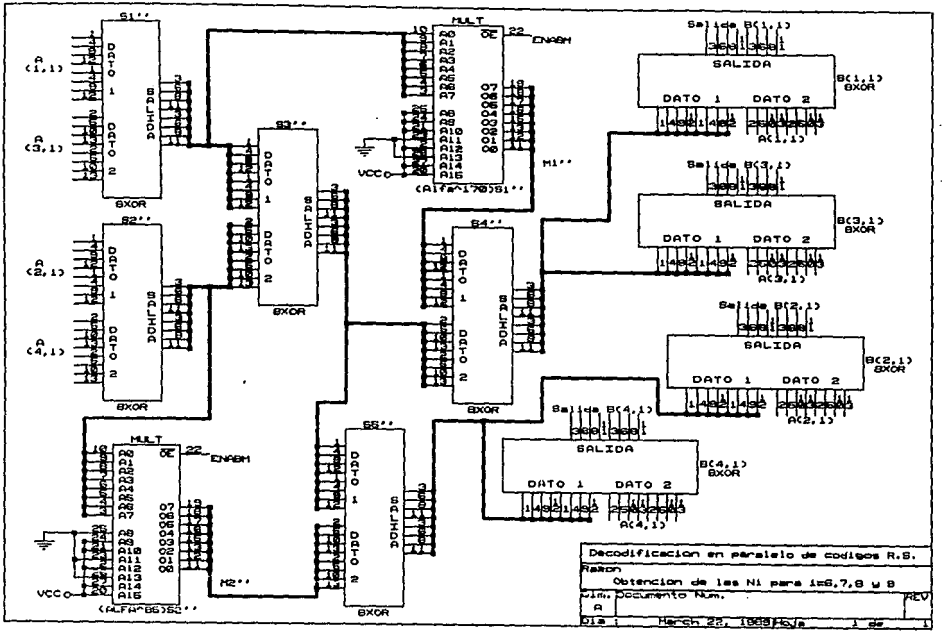

Salida 1
Salida 2
Salida 3
Salida 4
Salida 5
Salida 6
Salida 7
Salida 8
Salida 9
Salida 10
Salida 11
Salida 12
Salida 13
Salida 14
Salida 15
Salida 16
Salida 17
74LS373
VCC
GND
Y1
Y2
Y3
Y4
Decodificador en paralelo de codigos H.E.
Fecha:
Contenido:
Dim:
Rev:
Hoja de: 1 Hojas

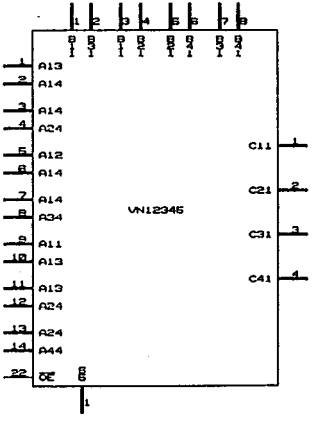

MAT-CTE
CONSTANTES
1 AL 17
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
G
Decodificacion en paralelo de codigos R.S.
Razon
"Chivo" de matriz cte. y ctes. para TRANS17
Dim. Documento Num.
REV
A
Dia : Abril 3, 1989 Hoja 1 de 1

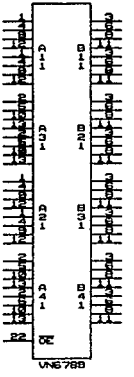
A
1
2
3
7486
B
4
5
6
7486
C
8
10
9
7486
D
12
13
11
7486
A
1
2
3
7486
B
4
5
6
7486
C
8
10
9
7486
D
12
13
11
7486
8XOR
Decodificacion en paralelo de codigos R.S.
Razon
Paquete de ocho compuertas XOR
Dim.
A
Documento Num.
REV
Dia : February 18, 1989 Hoja 1 de 1


Dato (1,1)
Salida Dato (1,1)
74LS373
Dato (2,1)
Salida Dato (2,1)
74LS373
Dato (3,1)
Salida Dato (3,1)
74LS373
Dato (4,1)
Salida Dato (4,1)
74LS373
VEC4X1
Decodificacion en paralelo de codigos R.S.
Razon
Vector de 4 renglones por 1 columna
Dim. Documento Num.
REV
A
Dia
Hoja
de

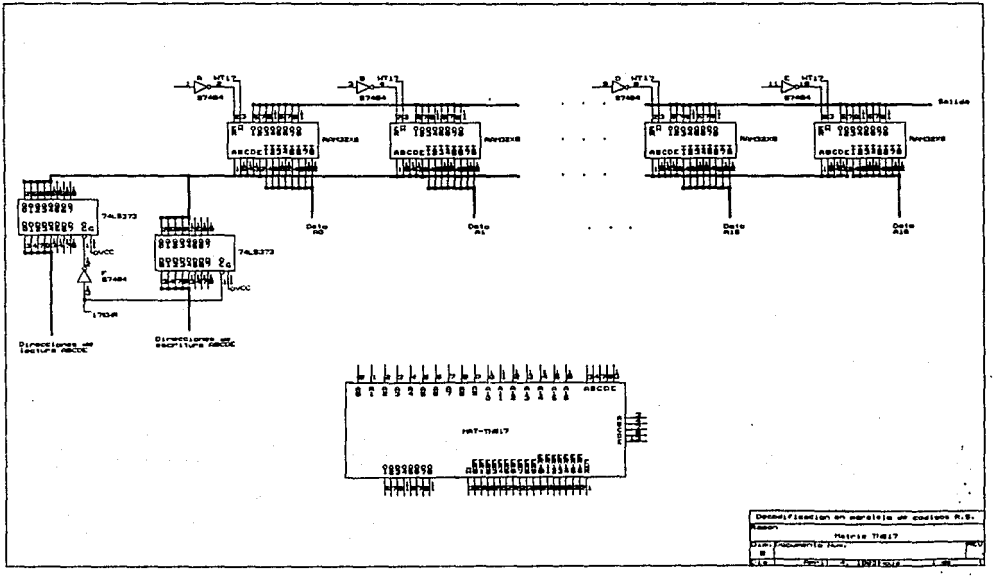

Decodificacion en paralelo de codigos R.S.
Obtencion de los Ki para i=1,2,3,4 y 5
Dib. Documento Num.
REV
Fecha: Marzo 84
Hoja

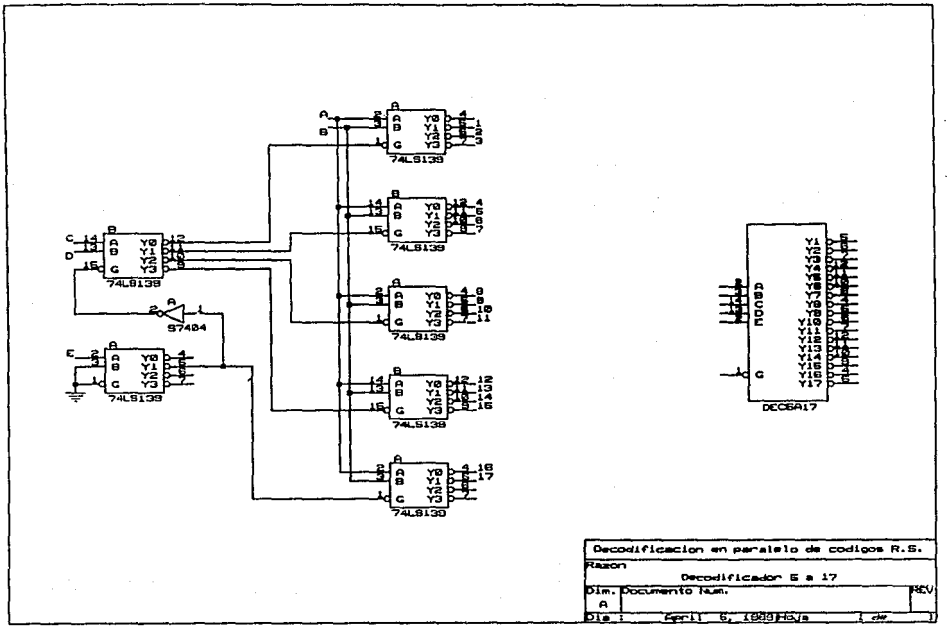

S1''
S2''
S3''
S4''
S5''
MULT
ENABM
M1''
M2''
BXOR
VCC
SALIDA
DATO 1
DATO 2
Salida B(1,1)
Salida B(3,1)
Salida B(2,1)
Salida B(4,1)
B(1,1) BXOR
B(3,1) BXOR
B(2,1) BXOR
B(4,1) BXOR
A(1,1)
A(3,1)
A(2,1)
A(4,1)
Decodificacion en paralelo de codigos R.S.
Obtencion de las Ni para i=6,7,8 y 9
Documento Num.
REV
Hoja 1 de 1

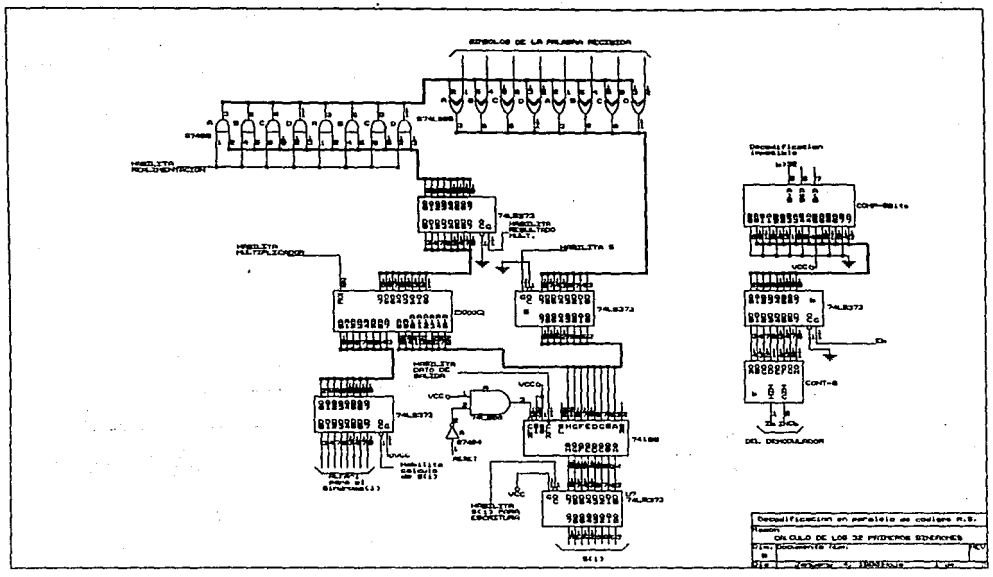


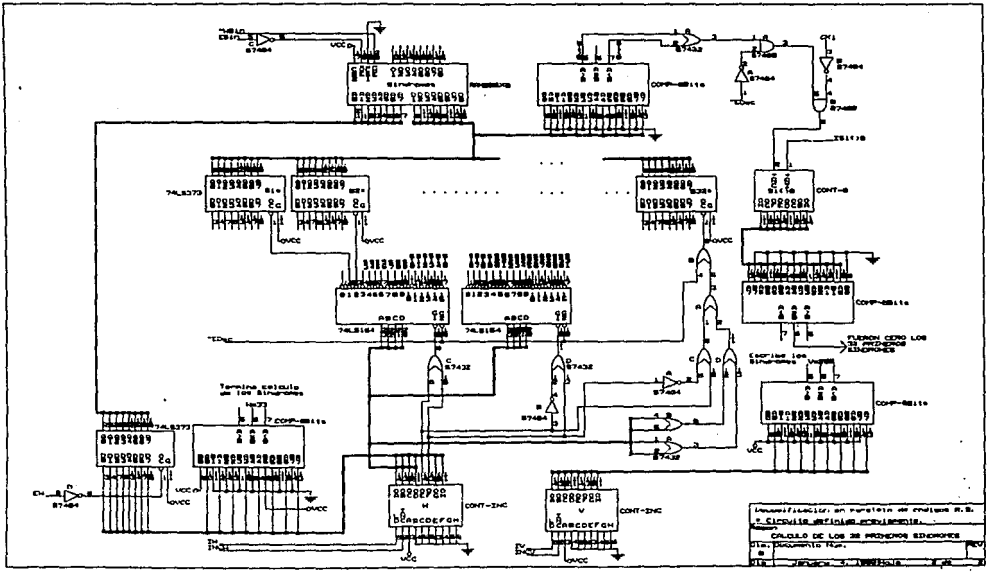


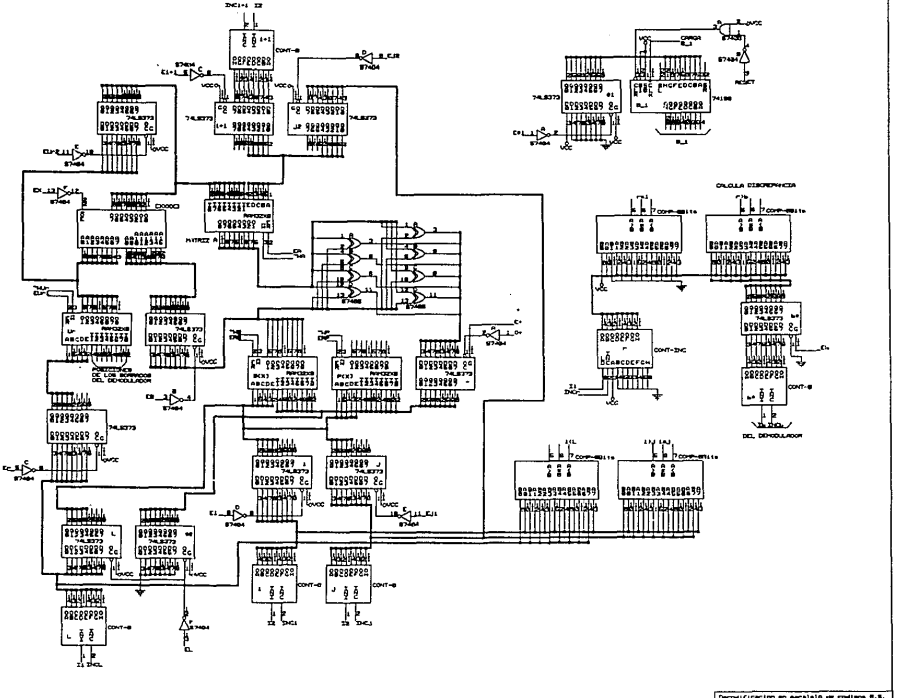
Decodificacion en paralelo de codigos R.B.	
Razon: "Chips" para calcular los vectores N	
Dim. A	Documento Num. REV
Dia: Abril 3, 1985 Hoja 1 de 1	

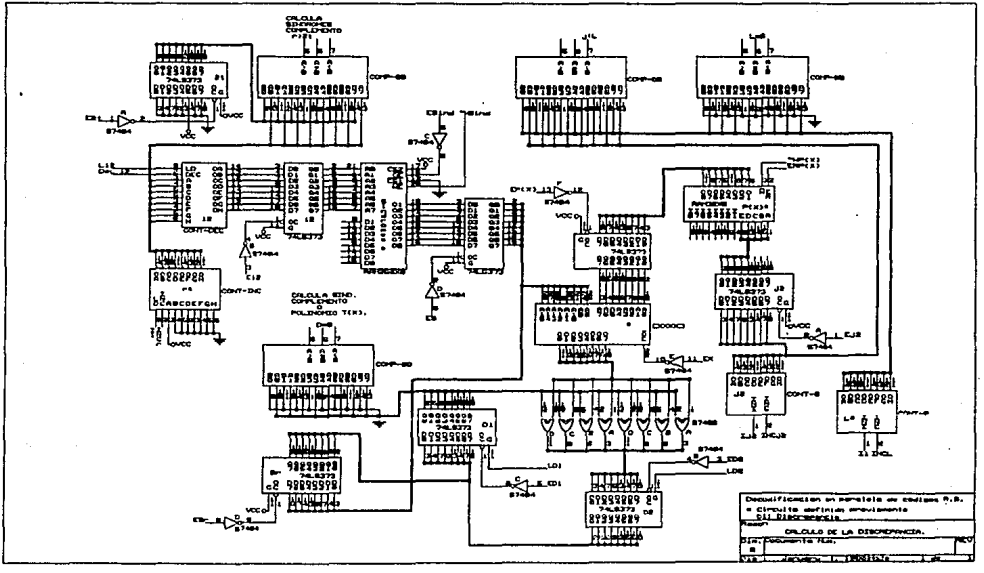

Salida
Direcciones de lectura ABCDE
Direcciones de escritura ABCDE
MAT-TM17
ABCDE
Decodificacion en paralelo de codigos R.S.
Matriz TM17


74LS139
S7404
DECOBA17
Decodificacion en paralelo de codigos R.S.
Razon
Decodificador 5 a 17
Dim.
A
Documento Num.
REV
Dia : April 6, 1988 Hoja

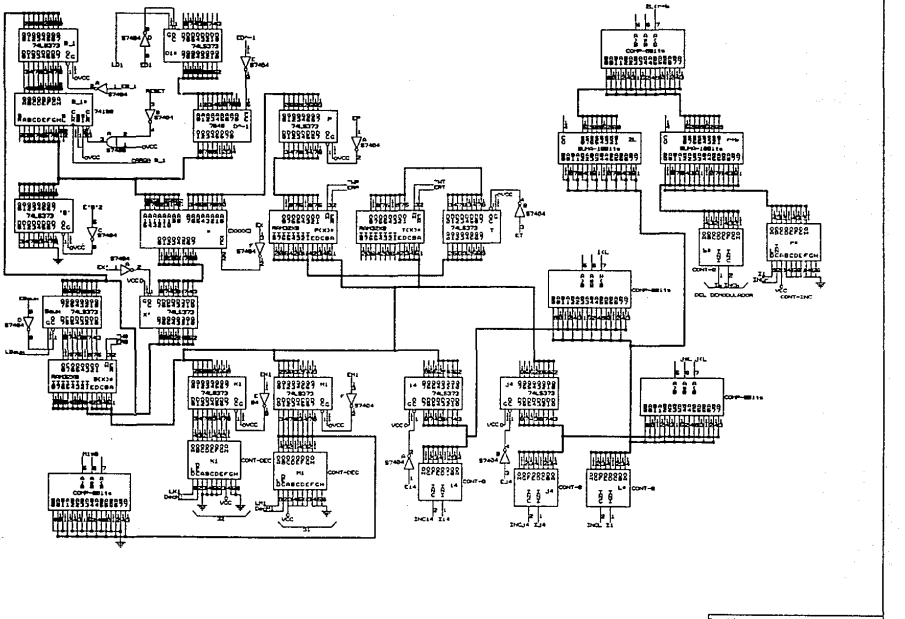

SIMBOLOS DE LA PALABRA RECIBIDA
74LS85
HABILITA RECALENTAMIENTO
74LS373
HABILITA REGISTRO FALET
HABILITA S
HABILITA MULTIPLEXOR
74LS372
DATO DE SALIDA
74LS86
74LS08
ALTAS para el simbolo(1)
Habilita salidas de 8bits
HABILITA 8bit PARA ESCRITURA
8bit
Decodificacion inmediato
COMP-8bits
CONT-8
DEL DECODIFICADOR
Decodificacion en paralelo de codigos R.S.
CALCULO DE LOS 32 PRIMEROS SINDROMES
Documento fuente:
Fecha: Febrero 4, 1988

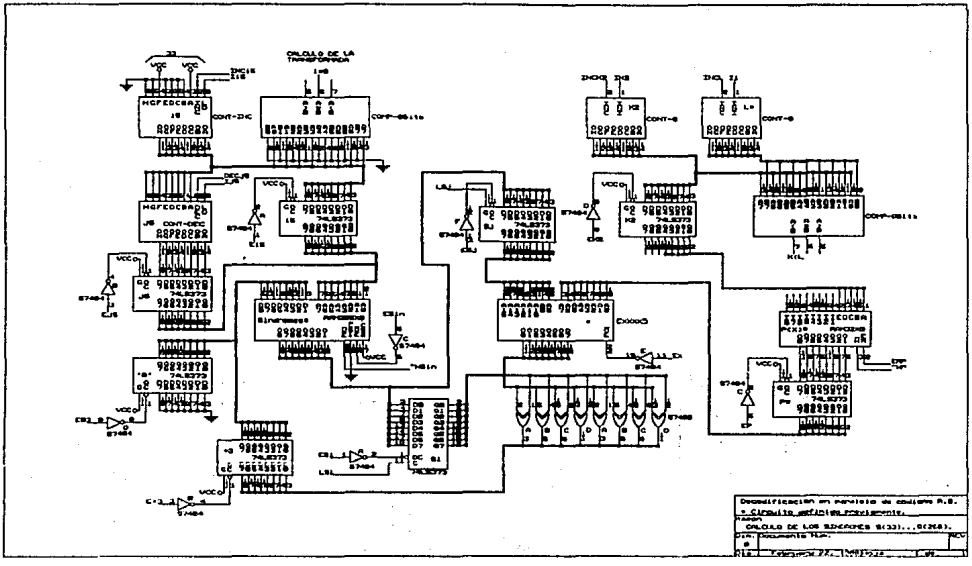

74LS373
74LS164
CONT-8Bits
CONT-4Bits
CONT-2Bits
CONT-INC
Termina calculo en los Sinumeros
Cuenta los Sinumeros
FUERON CERO LOS 30 PRIMEROS SINUMEROS
Identificación: en ruteiro de cruces N.S. y Circuito anticuas encruzando.
CALCULO DE LOS 30 PRIMEROS SINUMEROS

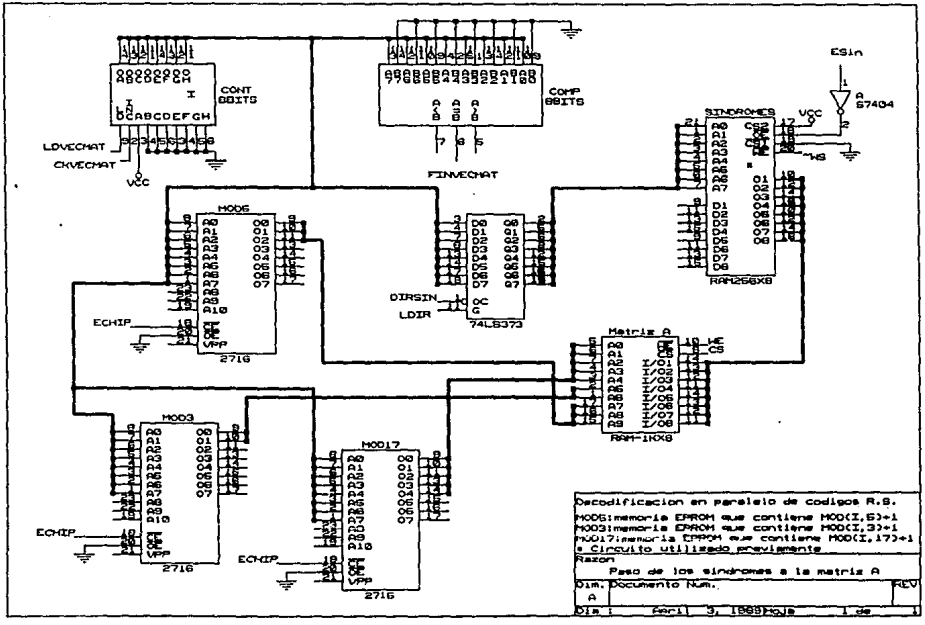

CALCULA DISCREPANCIA
DEL DEMODULADOR
Decodificacion en paralelo de codigos R.S.
Circuito definido previamente.
Bloque: Calculo localizadores de posiciones.
CALCULO DEL POLINOMIO S(X)

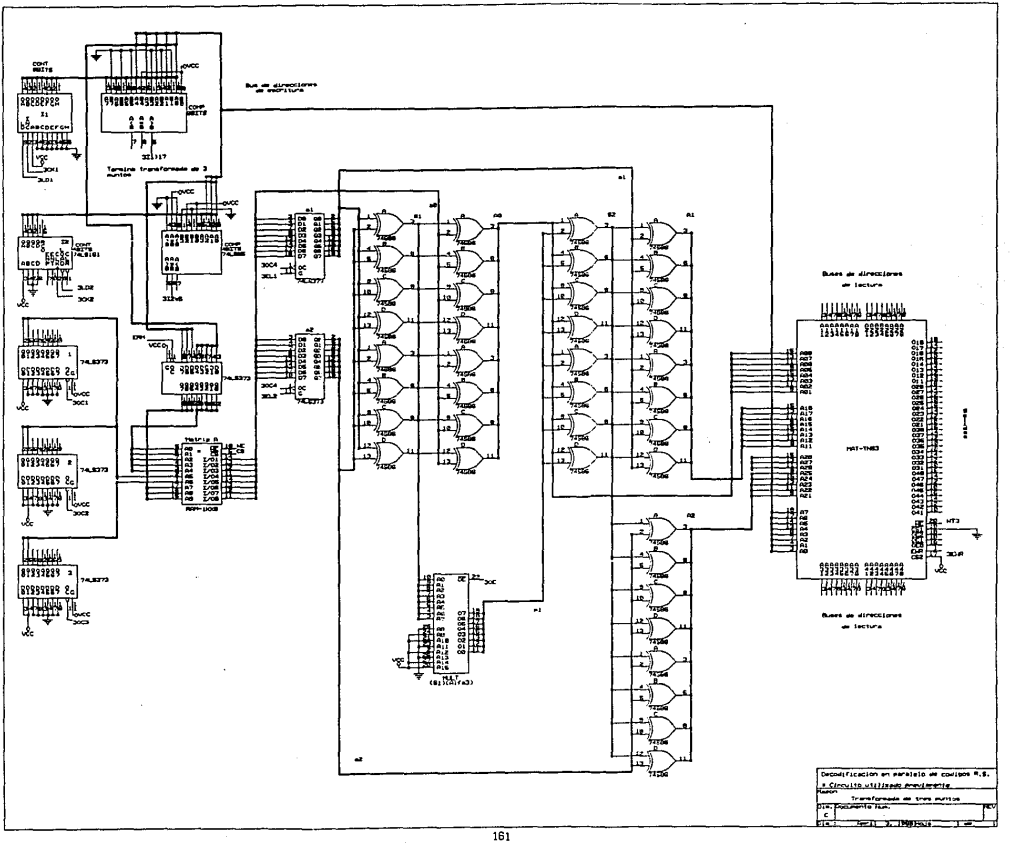

CALCULA SINDROME COMPLEMENTO POLINOMIO T(X).
CONT-DEC
CONT-INC
CONT-0
Decodificacion en paralelo de codigos R.S.
+ circuito optimo encontrado
sin DISCORDANCIA.
CALCULO DE LA DISCORDANCIA.

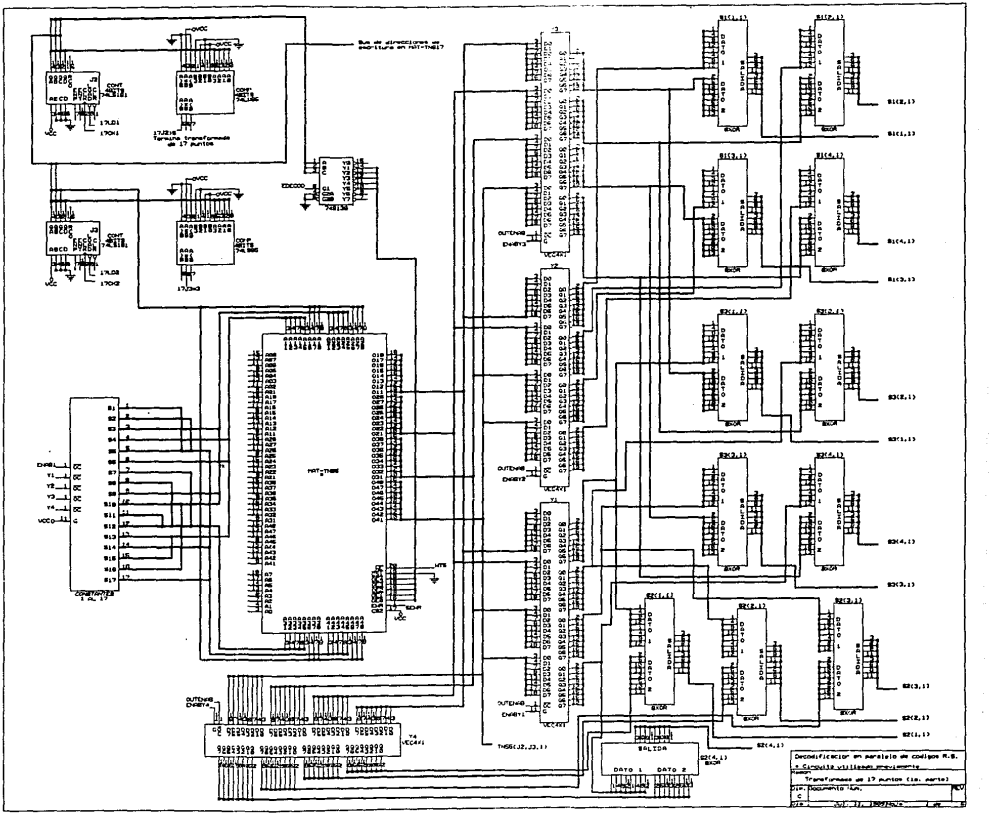
CALCULO DEL POLINOMIO T(x)

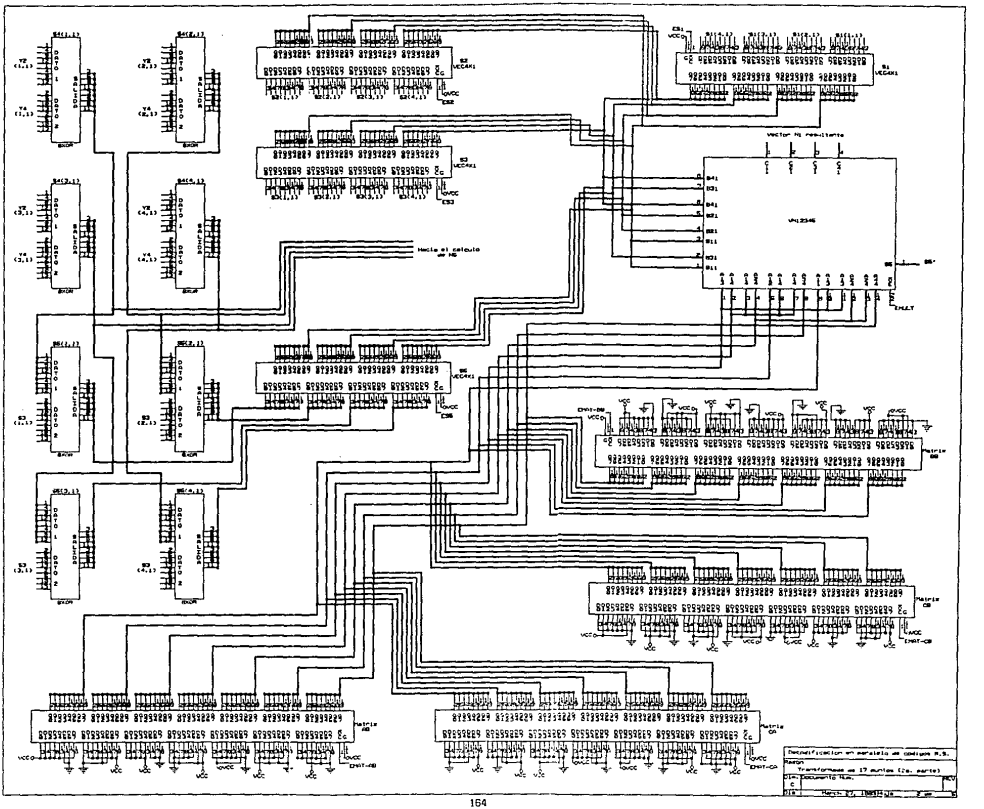

Descodificacion en ensamblador de codigos M.S.
Circuito definido anteriormente.
PCK31 Polinomio Incalculador de bornamos y errores.
CALCULO DEL POLINOMIO PCK3.
Fecha: Febrero 22, 1988 16:16

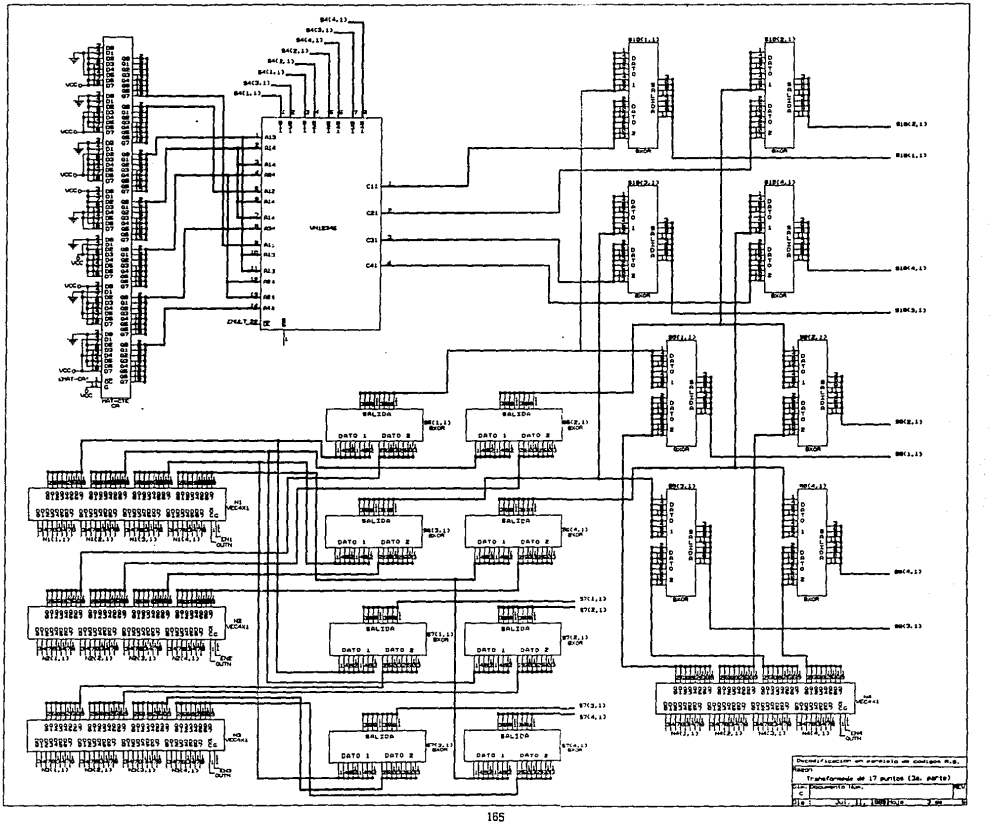

CALCULO DE LA TRANSFORMADA
CONT-I2C
CONT-DEC
CONT-6
CONT-9
Documentación en revisión de edición R.B.
CALCULO DE LOS SINCROS S(33)...S(36).

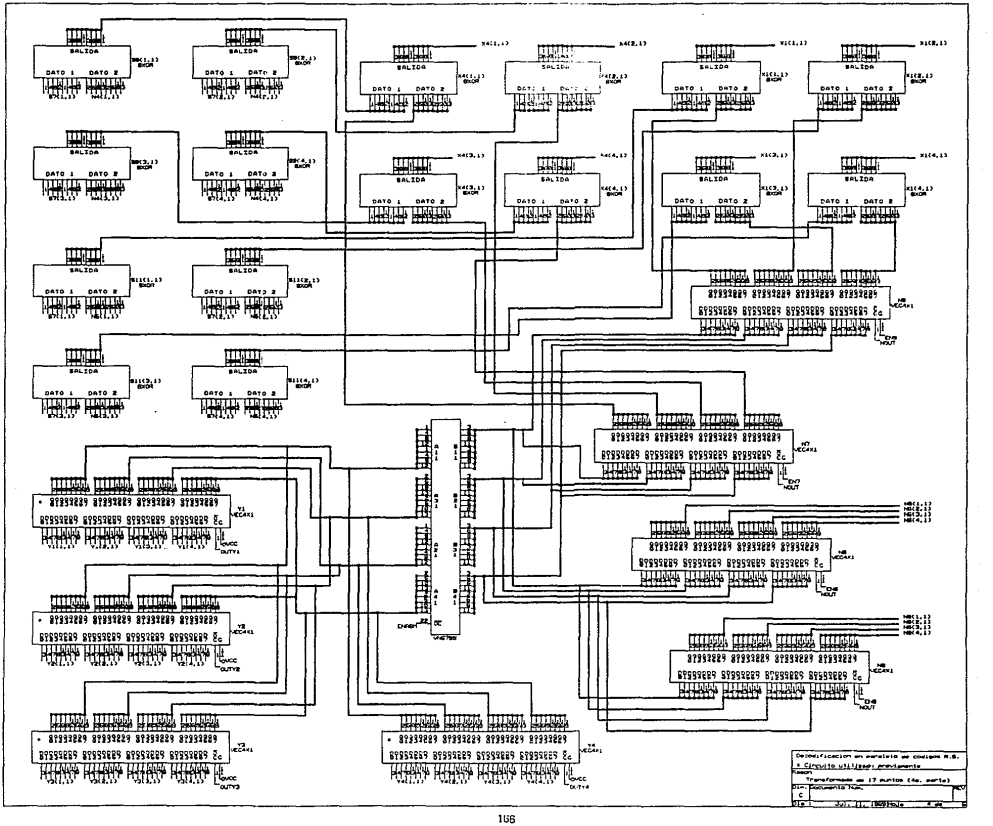

CONT 88ITS
COMP 88ITS
LDVECHAT
CKVECHAT
Vcc
FINVECHAT
ESin
SINDROMES
RAM256X8
MOD6
2716
DIRSIN
LDIR
74LS373
ECHIP
Matriz A
RAM-1MX8
MOD3
MOD17
Decodificacion en paralelo de codigos R.S.
MOD6:memoria EPROM que contiene MOD(I,6)+i
MOD3:memoria EPROM que contiene MOD(I,3)+i
MOD17:memoria EPROM que contiene MOD(I,17)+i
+ Circuito utilizado previamente
Razon:
Paso de los sindromes a la matriz A
Dim. Documento Num.
REV
A
Dia
Abril
9, 1989
Hoja
1 de
1

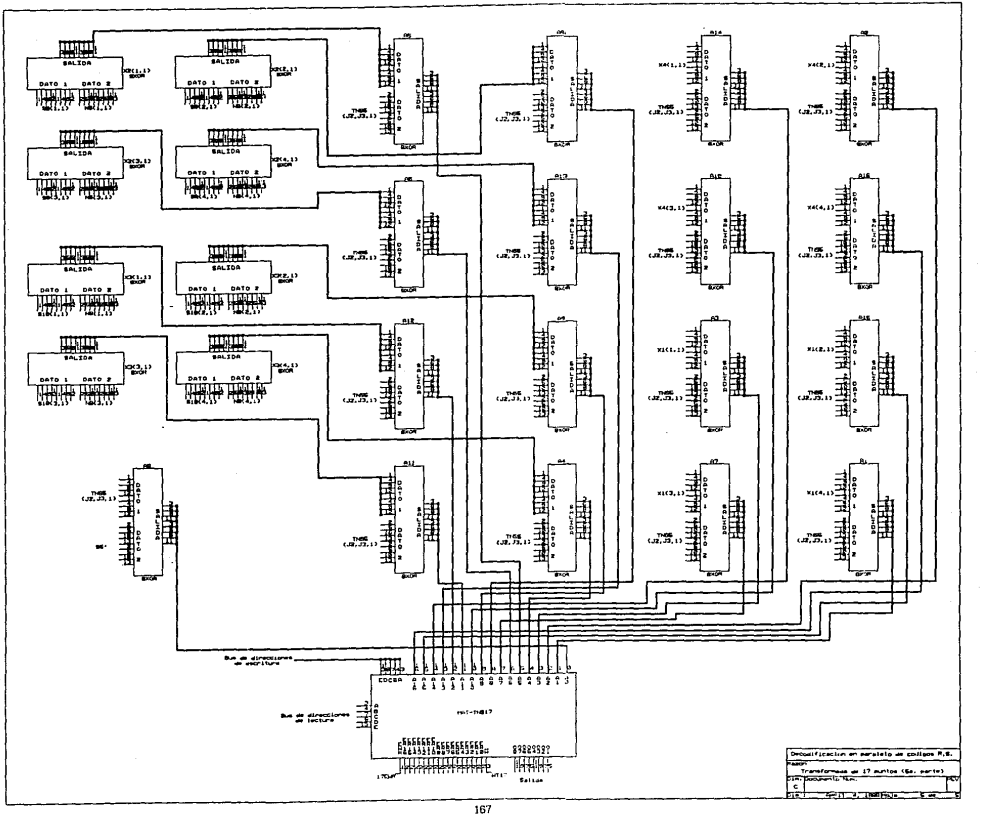

Buses de direcciones de lectura
Buses de direcciones de lectura
Buses de direcciones de lectura
Descodificacion en paralelo de conbus M.S.
Circuito utilizado anteriormente
Transformada de tres puntos

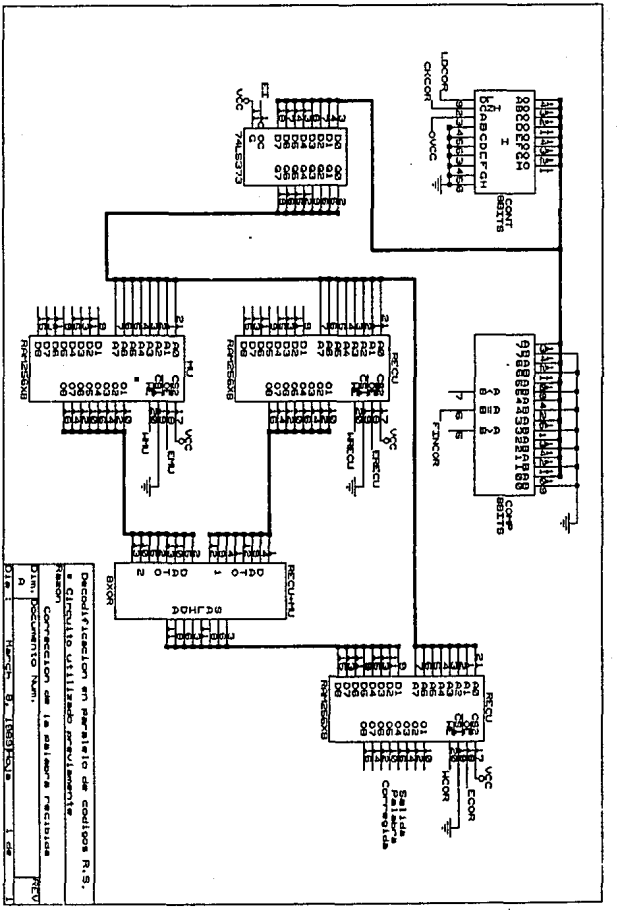

Decodificador en paralelo de codigo R.S.
Transformada de 17 puntos (1a. parte)


Vector de resultante
Matrice AB
Matrice CB
Matrice CA
Multiplexeur
Demultiplexeur
Transformee en 17 mutlex (2e. partie)


Decodificador en paralelo de codigos R.B.
Transformada de 17 puntos (3a. parte)


SALIDA
DATO 1
DATO 2
Desmultiplexación en paralelo de canales M.B.
Circuito utilizado previamente
Transformada de 17 puntos (4a. parte)
Documento Nº:
Rev:
Hoja:
Jun 11, 1989

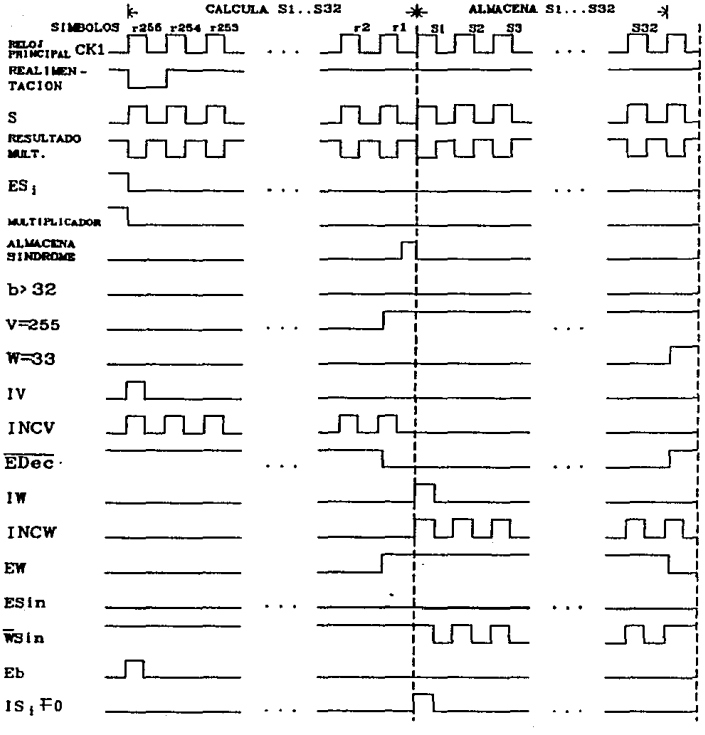

SALIDA
DATO 1
DATO 2
Bus de direcciones de escritura
Bus de direcciones de lectura
Salida
Decodificacion en paralelo de codigos R.S.
Transformacion de 17 puntos (6a. parte)


Decodificación en Paralelo de codigos R.S.
Circuito utilizado previamente.
Correccion de la palabra recibida
Dim. Documento Num.
A
Hoja: 8 de: 10
REV
CONT BITS
CORP BITS
74LS373
RAM256X8
RECU
HCOR
ECOR
FINCOR
Salida Corregida
BXOR

IV.4 DIAGRAMAS DE CONTROL DE LOS CIRCUITOS.

Por último se incluyen los diagramas de control de cada circuito a manera de diagramas de tiempo, donde aparecen cada una de las señales de control del circuito, con el fin de mostrar más a detalle cómo interactuan entre sí los elementos que forman cada etapa del proceso de decodificación.

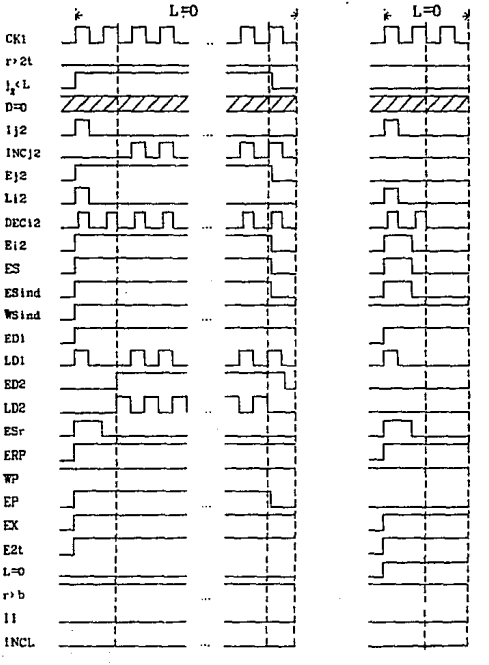
CALCULO DE LOS 32 PRIMEROS SINDROMES



CALCULO DE LA DISCREPANCIA

POLINOMIO LOCALIZADOR DE BORRADOS
CALCULO DE LA DISCREPANCIA
CK1
I1
r>b
r=1
I2
Ei
EL
Ej1
Ei+1
EU2
EU1
WUr
EX
EA
WA
ERB
WB
ERP
WP
INC i+1
INC i
Ej2
E+
O+
i<L
INCj
j≤1
INCL
INCr
Er
E≠1
Eb
ED
Eb

CALCULO DE LA DISCREPANCIA

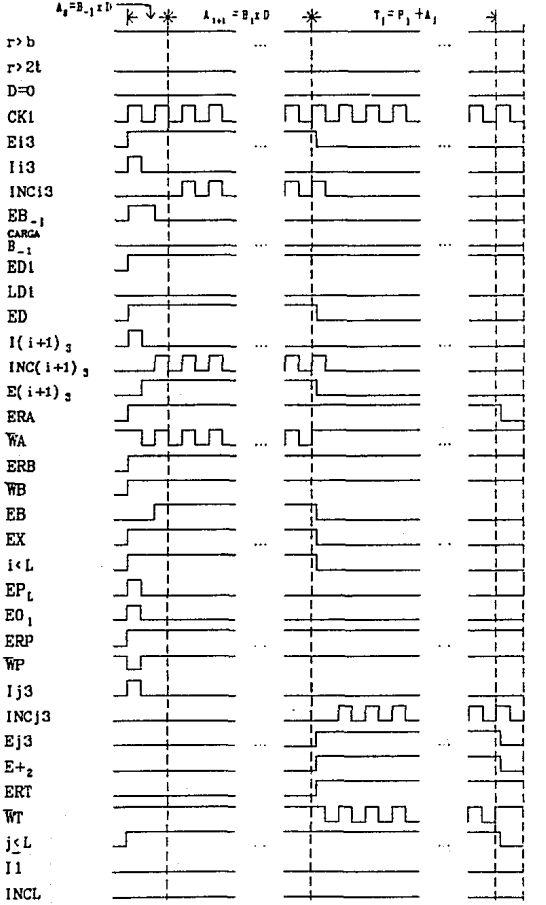


SI D=0 CALCULO DE LOS SINDROMES S33...S255

SI D=1 CALCULO DEL POLINOMIO T(X)

SI r>2t CALCULO DE LOS SINDROMES S33...S255

CALCULO DEL POLINOMIO T(X)



CALCULO DEL POLINOMIO P(X)

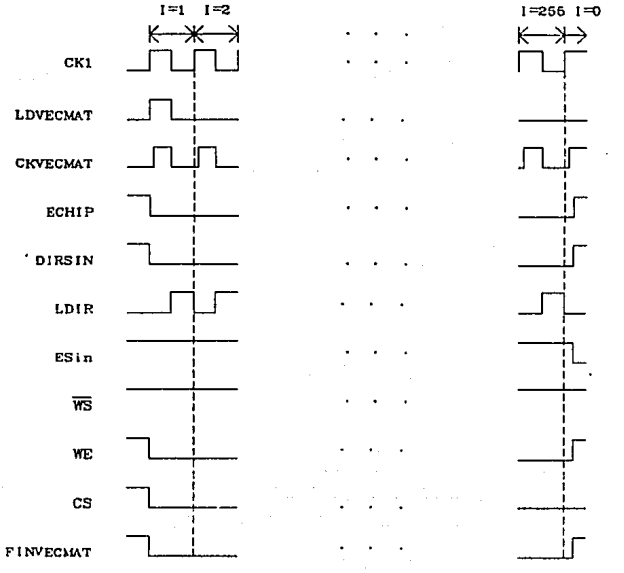
CALCULO DEL POLINOMIO P(X)

CALCULO DE LA DISCREPANCIA

CALCULO DE LOS SINDROMES S33...S255

CALCULO DE LA TRANSFORMADA

PASO DEL VECTOR DE SINDROMES A LA MATRIZ A

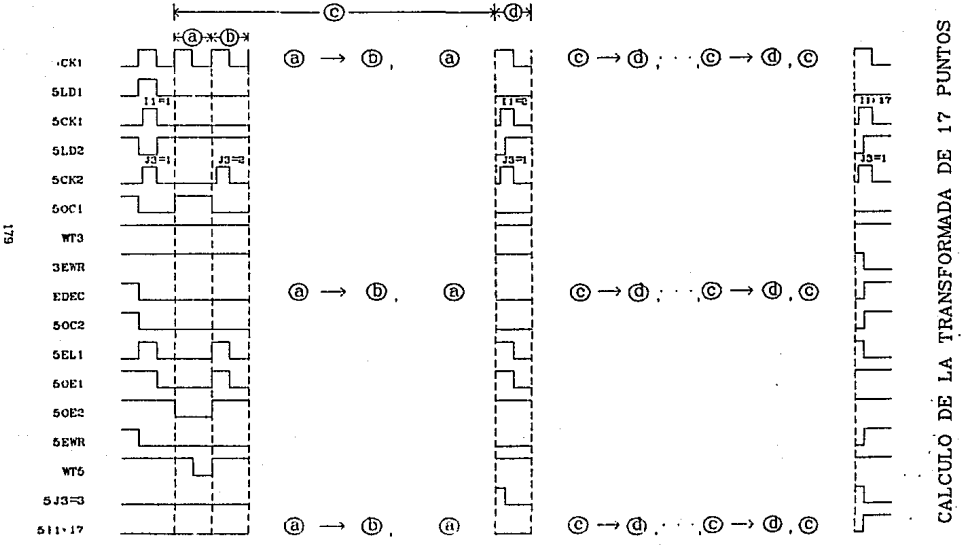


CALCULO DE LA TRANSFORMADA DE 3 PUNTOS

CALCULO DE LA TRANSFORMADA DE 3 PUNTOS

CALCULO DE LA TRANSFORMADA DE 5 PUNTOS

CALCULO DE LA TRANSFORMADA DE 5 PUNTOS

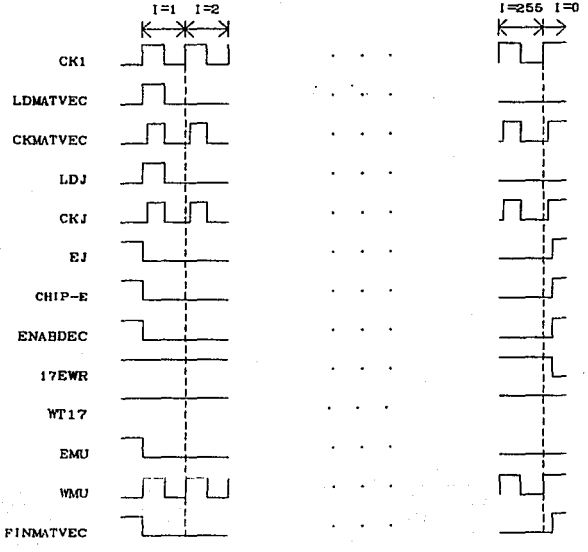


CALCULO DE LA TRANSFORMADA DE 17 PUNTOS

CALCULO DE LA TRANSFORMADA DE 17 PUNTOS

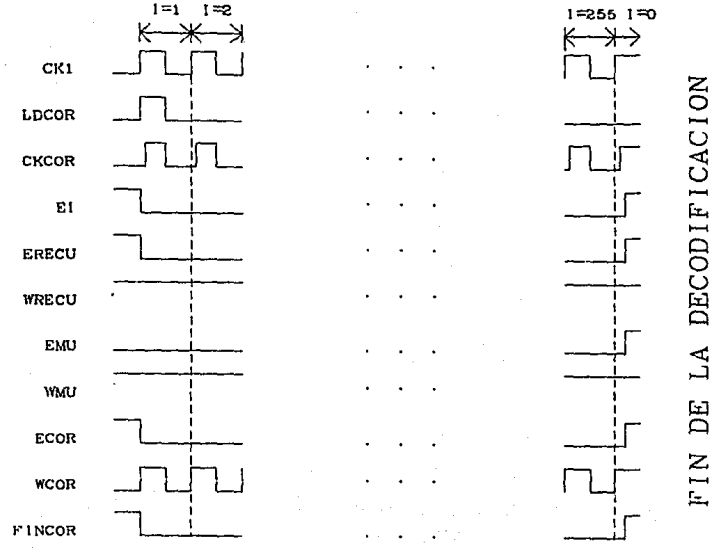
PASO DE LA MATRIZ TNS17 AL VECTOR MU

PASO DE LA MATRIZ TNS17 AL VECTOR MU



CORRECCION DE LA PALABRA RECIBIDA

CORRECCION DE LA PALABRA RECIBIDA



Conclusiones

Una de las contribuciones de este trabajo es el desarrollo de varios paquetes que nos permiten hacer la simulación de la codificación y decodificación de códigos de Reed-Solomon.

Mediante la simulación de tres algoritmos para códigos R.S. fue posible seleccionar el más rápido de ellos. Este algoritmo emplea una transformada rápida y además utiliza las partes más rápidas de los otros algoritmos.

Otra contribución, es el diseño realizado del decodificador con elementos discretos, basado en el algoritmo que resultó ser el más rápido. Debido a que este diseño cuenta con arquitectura en paralelo, tiene la gran ventaja de realizar procesos de decodificación en tiempos menores, sobre decodificadores que no cuentan con este tipo de arquitectura. El ahorro de tiempo es la mayor ventaja del diseño realizado, sin embargo se tiene la desventaja de que se requiere mayor circuiteria para su implementación lo cual implica un mayor costo. Por otro lado, este diseño puede servir como punto de partida para una futura implementación en tecnología VLSI, lo cual simplificaría considerablemente la misma.

La implementación del decodificador no se llevó a cabo debido a que no se contó con los recursos económicos necesarios para tal efecto.

Como trabajo futuro, para una posible implementación, queda hacer un análisis de velocidad contra costo del decodificador para poder evaluar cuánto conviene paralelizar en función del ahorro de tiempo y el costo que esto tendría.

También queda por hacer el diseño del controlador, que es el encargado de coordinar todo el proceso de decodificación.

BIBLIOGRAFIA

[1] C.E. Shannon.
A Mathematical Theory of Communication.
Bell Syst. Tech. J., vol 27.
pp. 379 - 423, Julio 1948. (Parte 1)
pp. 623 - 656, Octubre 1948. (Parte 2)

[2] Peterson and Weldon.
Error Correcting Codes.
Editorial MIT Press 1972.

[3] Richard E. Blahut.
Theory and Practice of Error Control Codes.
Editorial Addison-Wesley 1983.

[4] Shu Lin, Daniel J. Costello, Jr.
Error Control Coding: Fundamentals and Applications.
Editorial Prentice-Hall 1983.

[5] Francisco J. García Ugalde.
Coding and Decoding Algorithms of Reed-Solomon Codes executed on a M68000 Microprocessor.
Lecture Notes in Computer Science Proceedings, pp. 183-196,
Springer-Verlag, Abril 1988.

[6] Kuang Yung Liu.
Architecture for VLSI Design of Reed-Solomon Encoders.
IEEE Trans. Computers, vol. c-31, No. 2, pp. 170-175,
Febrero 1982.

[7] In-Shek Hsu, Irving S. Reed, T.K. Truong, Ke Wang, Chiunn-Shyong Yeh, Leslie J. Deutsch.
The VLSI Implementation of a Reed-Solomon Encoder using Berlekamp's Bit-Serial Multiplier Algorithm.
IEEE Trans. Computers, vol. c-33, No. 10, pp. 906-911,
Octubre 1984.

[8] Elwyn R. Berlekamp.
Bit-Serial Reed-Solomon Encoders.
IEEE Trans. Inform. Theory, vol. IT-28, No. 6, pp. 869-874,
Noviembre 1982.

[9] Hirokazu Okano and Hideki Imai.
A Construction Method of High-Speed Decoders using ROM's for Bose-Chaudhuri-Hocquenghem and Reed-Solomon Codes.
IEEE Trans. Computers, vol. c-36, No. 10, pp. 1165-1171, Octubre 1987.

[10] Howard M. Shao, T.K. Truong, Leslie J. Deutsch, Joseph H. Yuen and Irving S. Reed.
A VLSI Design of a Pipeline Reed-Solomon Decoder.
IEEE Trans. Computers, vol. c-34, No. 5, pp. 393-403, Mayo 1985.

[11] Kuang Yung Liu.
Architecture for VLSI Design of Reed-Solomon Decoders.
IEEE Trans. Computers, vol. c-33, No. 2, pp. 178-189, Febrero 1984.

[12] Irving S. Reed, T.K. Truong, R.L. Miller and B. Benjauthrit.
Further results on Fast Transforms for Decoding Reed-Solomon Codes over $GF(2^m)$. *For* $m = 4,5,6,8$.
Deep Space Network Progress Report 42-50, Jet Propulsion, Laboratory, Pasadena, California, pp. 132-154, Enero 1979.

[13] R.L. Miller, T.K. Truong, Irving S. Reed.
Efficient Program for Decoding the (255,223) Reed-Solomon Code over $GF(2^8)$ *with both errors and erasures, using Transform Decoding.*
IEEE PROC. vol.127 Pt. E, No. 4, Julio 1980.

[14] T.K. Truong, R.L. Miller, Irving S. Reed.
Fast Technique for Computing Syndromes of B.C.H. and Reed-Solomon Codes.
Electron. Lett., pp. 720-721, Septiembre 1979.

[15] Elwyn R. Berlekamp.
Algebraic Coding Theory.
New York: McGraw-Hill, 1968.

[16] K. Blair Benson.
Television Engineering Handbook.
McGraw-Hill, pp. 18.7-18.22, 1986.

[17] S. Harari.
Protection contre les erreurs en enregistrement magnétique.
Ann. Télécommunic., 34, No.7-8, pp. 389-399, Marzo 1979.

[18] T. Wolff.
The (31,21) BCH Code for Meteosat Data-Collection-Platform Address Transmission and Recognition.
ESA Journal, vol.3, pp. 317-324, 1979.

[19] Brian C. Mortimer, Michael J. Moore, and Mike Sablatash.
The Design of a High-Performance Error-Correcting Coding Scheme for the Canadian Broadcast Teildon System Based on Reed-Solomon Codes.
IEEE Trans. Communications, vol. com-35, No. 11, pp. 1113-1123, Noviembre 1987.

[20] Jean-Michel Bois, Max Ferreol.
A Reed-Solomon (255,223) Decoder Equipment for Space Telemetry Links With Multiprocessors Implementation.
Fifth International Conference on Applied Algebra, Algebraic Algorithms and Error-Correction-Codes, Menorca, España, Junio 15-19, 1987.

[21] Kwan Ying Muramoto.
Implementation of a Packet Recovery Code Using Reed-Solomon Codes.
Technical Report B84-2, Mayo 1984.

[22] P. Stammnitz.
Error Protection of 34 Mbit/s DPCM Encoded TV Signals with Multiple Error Correcting BCH Codes.
Heinrich-Hertz-Institut, Berlin, W. Germany.

[23] Consultative Committee for Space Data Systems (CCSDS).
Telemetry Channel Coding.
CCSDS 101.0-B-2, Blue Book, Enero 1987.

APENDICE

Este apéndice contiene el algoritmo para calcular las transformadas de 3, 5 y 17 puntos.

Sea α un elemento del campo finito $CG(2^8)$.

La transformada de 3 puntos está dada por:

$$A_k = \sum_{n=0}^{3-1} a_n \alpha_3^{nk} \quad \text{para } 0 \le k \le 2$$

donde $\alpha_3 = \alpha^{85}$ es una raiz cúbica primitiva.

Algoritmo para calcular la transformada de 3 puntos:

$$s_1 = a_1 + a_2, \quad A_0 = s_1 + a_0, \quad m_1 = \alpha_3 s_1,$$

$$s_2 = A_0 + m_1, \quad A_1 = s_2 + a_1, \quad A_2 = s_2 + a_2$$

la transformada de 3 puntos requiere sólo una multiplicación y cinco sumas.

La transformada de 5 puntos está dada por:

$$A_k = \sum_{n=0}^{5-1} a_n \alpha_2^{nk} \quad \text{para } 0 \le k \le 4$$

donde $\alpha_2 = \alpha^{51}$ es una raiz quinta primitiva.

Algoritmo para calcular la transformada de 5 puntos:

$$s_1 = a_2 + a_3, \quad s_2 = a_1 + a_4, \quad s_3 = a_1 + a_3,$$

$$s_4 = a_2 + a_4, \quad s_5 = s_1 + s_2, \quad A_0 = s_5 + a_0,$$

$$m_1 = (1+\alpha_2^3)s_5, \quad m_2 = (\alpha_2^3 + \alpha_2^4)s_1,$$

$$m_3 = (\alpha_2 + \alpha_2^3)s_2, \quad m_4 = (\alpha_2 + \alpha_2^4)s_3,$$

$$m_5 = (\alpha_2 + \alpha_2^4)s_4, \quad s_6 = A_0 + m_1, \quad s_7 = s_6 + m_2,$$

$$s_8 = s_8 + m_3, \quad s_9 = m_5 + a_2, \quad s_{10} = m_4 + a_1,$$

$$s_{11} = m_5 + a_4, \quad s_{12} = m_4 + a_3, \quad A_1 = s_8 + s_9,$$

$$A_2 = s_7+s_{10}, \quad A_3 = s_7+s_{11}, \quad A_4 = s_8+s_{12}$$

la transformada de 5 puntos requiere sólo cinco multiplicaciones y diecisiete sumas.

La transformada de 17 puntos está dada por:

$$A_k := \sum_{n=0}^{17-1} a_n \alpha_1^{nk} \quad \text{para } 0 \le k \le 16$$

donde $\alpha_1 = \alpha^{120}$ es una raiz decimoséptima primitiva.

Algoritmo para calcular la transformada de 17 puntos:

$$\begin{aligned}
&S_1 = Y_2+Y_3, \quad S_2 = Y_1+Y_4, \quad S_3 = Y_2+Y_3, \\
&S_4 = Y_2+Y_4, \quad S_5 = S_3+S_4, \quad N_1 = BS_5, \\
&N_2 = (A+B)S_1, \quad N_3 = (C+B)S_2, \quad N_4 = (C+A)S_3, \\
&N_5 = (C+A)S_4, \quad N_6 = EY_1, \quad N_7 = EY_3, \qquad \ldots(A.1) \\
&N_8 = EY_4, \quad N_9 = EY_2, \quad S_6 = N_1+N_2, \\
&S_7 = N_1+N_3, \quad S_8 = S_6+N_4, \quad S_9 = S_7+N_4, \\
&S_{10} = S_6+N_5, \quad S_{11} = S_7+N_5, \quad X_1 = S_{11}+N_6, \\
&X_2 = S_8+N_6, \quad X_3 = S_{10}+N_6, \quad X_4 = S_9+N_7
\end{aligned}$$

donde

$$A = \begin{bmatrix} \alpha_1^8 & \alpha_1^0 & \alpha_1^{13} & \alpha_1^{14} \\ \alpha_1^6 & \alpha_1^{13} & \alpha_1^{14} & \alpha_1^2 \\ \alpha_1^{13} & \alpha_1^{14} & \alpha_1^2 & \alpha_1^{10} \\ \alpha_1^{14} & \alpha_1^2 & \alpha_1^{10} & \alpha_1^{16} \end{bmatrix}, \quad B = \begin{bmatrix} \alpha_1^2 & \alpha_1^{10} & \alpha_1^{16} & \alpha_1^{12} \\ \alpha_1^{10} & \alpha_1^{16} & \alpha_1^{12} & \alpha_1^9 \\ \alpha_1^{16} & \alpha_1^{12} & \alpha_1^9 & \alpha_1^{11} \\ \alpha_1^{12} & \alpha_1^9 & \alpha_1^{11} & \alpha_1^4 \end{bmatrix},$$

$$C = \begin{bmatrix} \alpha_1^8 & \alpha_1^{11} & \alpha_1^4 & \alpha_1^3 \\ \alpha_1^{11} & \alpha_1^4 & \alpha_1^3 & \alpha_1^{15} \\ \alpha_1^4 & \alpha_1^3 & \alpha_1^{15} & \alpha_1^7 \\ \alpha_1^7 & \alpha_1^{15} & \alpha_1^7 & \alpha_1^1 \end{bmatrix}, \quad D = \begin{bmatrix} \alpha_1^{15} & \alpha_1^7 & \alpha_1^1 & \alpha_1^5 \\ \alpha_1^7 & \alpha_1^1 & \alpha_1^5 & \alpha_1^8 \\ \alpha_1^1 & \alpha_1^5 & \alpha_1^8 & \alpha_1^6 \\ \alpha_1^5 & \alpha_1^8 & \alpha_1^6 & \alpha_1^{13} \end{bmatrix},$$

$$X_1 = [A'_3, A'_{15}, A'_7, A'_1]^T,$$

$$X_2 = [A'_8, A'_9, A'_6, A'_{13}]^T,$$

$$X_3 = [A'_{12}, A'_9, A'_{11}, A'_4]^T,$$

$$X_4 = [A'_{14}, A'_2, A'_{10}, A'_{16}]^T.$$

Los términos A_k para $1 \leq k \leq 16$ se obtienen de la siguiente manera:

$$A_k = A'_k + a_0$$

además, $E = A+B+C+D$ y Y_1 a Y_4 se obtienen a partir de las expresiones para X_1 a X_4 reemplazando cada A'_i por a_i para $1 \leq i \leq 16$.

De las ecuaciones (A.1), N_i para $i = 1, 2, 3, 4$ y 5, pueden obtenerse con el siguiente algoritmo [12]:

Si representamos a una de las N_i para $1 \leq i \leq 5$ como:

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} \gamma^1 & \gamma^2 & \gamma^3 & \gamma^4 \\ \gamma^2 & \gamma^3 & \gamma^4 & \gamma^5 \\ \gamma^3 & \gamma^4 & \gamma^5 & \gamma^6 \\ \gamma^4 & \gamma^5 & \gamma^6 & \gamma^7 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

entonces

$$s_1 = c_1+c_3, \quad s_2 = c_1+c_2, \quad s_3 = c_2+c_4,$$

$$s_4 = c_3+c_4, \quad s_5 = s_2+s_4, \quad m_1 = \gamma^4 s_5,$$

$$m_2 = (\gamma^3+\gamma^4)s_1, \quad m_3 = (\gamma^4+\gamma^5)s_3, \quad m_4 = (\gamma^2+\gamma^4)s_2,$$

$$m_5 = (\gamma^4+\gamma^6)s_4, \quad m_6 = (\gamma^1+\gamma^3+\gamma^2+\gamma^4)c_1, \quad m_7 = (\gamma^2+\gamma^4+\gamma^3+\gamma^5)c_2,$$

$$m_8 = (\gamma^3+\gamma^5+\gamma^4+\gamma^6)c_3, \quad m_9 = (\gamma^4+\gamma^6+\gamma^5+\gamma^7)c_4,$$

$$s_6 = m_1, \quad s_7 = s_6+m_2, \quad s_8 = s_7+m_4,$$

$$s_9 = s_6+m_3, \quad s_{10} = s_8+m_4, \quad s_{11} = s_7+m_5,$$

$$s_{12} = s_8+m_5, \quad b_1 = s_8+m_8, \quad b_2 = s_{10}+m_7,$$

$$b_3 = s_{11}+m_6, \quad b_4 = s_{12}+m_9$$

de aquí que el número total de multiplicaciones y sumas en el campo de Galois que se necesitan para calcular una de las N_i $1 \leq i \leq 5$ son nueve y dieciseis, respectivamente. El término A_0 se obtiene cuando se calcula N_1 de la

siguiente manera:

$$A_0 = s_8 + a_0$$

Las N_i para $i = 6, 7, 8$ y 9, pueden obtenerse con un procedimiento similar al anterior [12]:

Si representamos a una de las N_i para $6 \leq i \leq 9$ como:

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} = \begin{bmatrix} \alpha^{170} & \alpha^{0} & \alpha^{85} & \alpha^{0} \\ \alpha^{0} & \alpha^{85} & \alpha^{0} & \alpha^{170} \\ \alpha^{85} & \alpha^{0} & \alpha^{170} & \alpha^{0} \\ \alpha^{0} & \alpha^{170} & \alpha^{0} & \alpha^{85} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

entonces

$$s_1 = c_1 + c_3, \quad s_2 = c_2 + c_4, \quad s_3 = s_1 + s_2,$$
$$m_1 = s_3, \quad m_2 = \alpha^{170} s_1, \quad m_3 = \alpha^{85} s_2,$$
$$s_4 = m_1 + m_2, \quad s_5 = m_1 + m_3, \quad b_1 = s_4 + c_1,$$
$$b_2 = s_5 + c_2, \quad b_3 = s_4 + c_3, \quad b_4 = s_5 + c_4$$

el número total de multiplicaciones y sumas que se necesitan para calcular alguna de las N_i para $i = 6, 7, 8$ y 9, es dos y nueve respectivamente. Combinando los resultados obtenidos, el número total de multiplicaciones y sumas que se necesitan para calcular la transformada de 17 puntos en el $CG(2^8)$ son 53 y 176, respectivamente.