



**DISEÑO DE UN SISTEMA DE SEGURIDAD ACTIVADO
POR TARJETA PERFORADA CONTROLADO POR
COMPUTADORA**



**TRABAJO DE GRADUACION
PREPARADO PARA LA FACULTAD DE
ESTUDIOS TECNOLOGICOS**

**PARA OPTAR AL GRADO DE:
TECNICO EN INGENIERIA ELECTRONICA**

**ELABORADO POR:
GOMEZ LOZADA , CARLOS EDUARDO.
ALVAREZ MONTANO , JUAN CARLOS.
RODRIGUEZ GARAY , HUMBERTO ALONSO.**

MARZO DE 1999

SOYAPANGO

EL SALVADOR

CENTRO AMERICA

UNIVERSIDAD DON BOSCO.

RECTOR

ING. FEDERICO MIGUEL HUGUET RIVERA.

SECRETARIO GENERAL

PBRO. PEDRO JOSE GARCIA CASTRO, S.D.B.

DECANO DE LA FACULTAD DE ESTUDIOS TECNOLOGICOS

ING. OSCAR REYNALDO VILLALTA LARA

ASESOR DE TRABAJO DE GRADUACION

ING. OSCAR REYNALDO VILLALTA LARA

JURADO EXAMINADOR

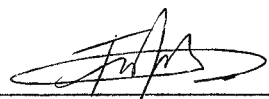
ING. FEDERICO LAINEZ

ING. WENCESLAO RIVAS

UNIVERSIDAD DON BOSCO.

FACULTAD DE ESTUDIOS TECNOLOGICOS

**JURADO EVALUADOR DEL TRABAJO DE GRADUACION:
DISEÑO DE UN SISTEMA DE SEGURIDAD ACTIVADO POR TARJETA
PERFORADA CONTROLADO POR COMPUTADORA**



ING. FEDERICO LAINEZ



ING. WENCESLAO RIVAS



ING. OSCAR R. VILLALTA LARA

AGRADECIMIENTO.

A DIOS TODOPODEROSO POR DARNOS LA SABIDURIA
PARA DESARROLLAR ESTE PROYECTO Y POR ESTAR SIEMPRE
EN NUESTRA AYUDA.

A NUESTROS PADRES Y FAMILIARES POR SU AYUDA EN
LA ELABORACION DE ESTE TRABAJO, PERO SOBRE TODO POR
SU AMOR, COMPRESION Y APOYO.

AL ING. OSCAR REYNALDO VILLALTA LARA POR SUS
ENSEÑANZAS Y A LA ASESORIA BRINDADA EN LA
ELABORACION DE ESTE TRABAJO.

A TODOS USTEDES QUE NOS BRINDARON SU AYUDA EN
EL MOMENTO OPORTUNO.

INDICE

INTRODUCCION	1
OBJETIVOS	2
<i>Objetivos Generales y Objetivos Específicos</i>	
ANTECEDENTES	3 – 4
DESCRIPCION DEL PROYECTO	5 – 6
<i>Explicación del diagrama de bloques</i>	
<i>Funcionamiento por bloques</i>	
<i>Descripción de cada bloque</i>	
<i>El programa principal</i>	
EXPLICACION DEL CIRCUITO	9 – 21
<i>Etapas de Optocuplas</i>	
<i>Interface Periférica Programable #1</i>	
<i>Interface Periférica Programable #2</i>	
<i>Etapas decodificadora</i>	
<i>Etapas de potencia</i>	
<i>Etapas de sensores</i>	
<i>Etapas detectora de cierre de puerta.</i>	
DIAGRAMA DEL CIRCUITO	22
FLUJOGRAMA DEL PROGRAMA	23 – 29
<i>Etapas detectora de tarjetas</i>	
<i>Etapas de entrada</i>	
<i>Subrutina COMPARE</i>	
<i>Subrutina ACTIVAR</i>	
<i>Etapas de salida</i>	
<i>Etapas del subsistema de seguridad</i>	
JUSTIFICACION	30 – 31
MARCO TEORICO	32 – 46

PLANTEAMIENTO DEL PROYECTO	47
METODOLOGIA DE LA INVESTIGACION	49 – 50
<i>Situación actual</i>	
<i>Solución Propuesta</i>	
LIMITANTES Y ALCANCES DEL PROYECTO	51 – 52
RECOMENDACIONES Y CONCLUSIONES	53 – 54
GLOSARIO TECNICO	55 – 56
BIBLIOGRAFIA	57
ANEXOS	58 - 150

INTRODUCCION

La seguridad es un aspecto de gran importancia para nuestra sociedad, mantener nuestros bienes protegidos no es un lujo sino una necesidad.

Gracias a los avances tecnológicos el hombre ha creado sistemas que protegen sus bienes contra intrusos, brindándole así mayor confianza. A estos dispositivos les llamaremos sistemas de seguridad.

Estos sistemas han ido evolucionando al mismo tiempo que avanza la tecnología, desarrollando así, sistemas de seguridad cada vez más confiables.

Los sistemas activados por tarjetas perforadas son un buen ejemplo de estos avances, el usuario puede tener acceso a sus bienes no con una llave de cerrojo convencional, sino que con una tarjeta especial la cual abrirá cualquier puerta o sistema electrónicamente.

El sistema planteado en este documento además de activarse por tarjeta perforada, también es programado por computadora, esto da mayores ventajas contra sistemas que poseen memoria y microprocesador propio.

El actual anteproyecto presenta una descripción detallada del funcionamiento general del sistema, así como también, enumera las ventajas que se tienen al utilizar una computadora como medio de programación para el sistema. También se incluyen algunas recomendaciones y puntos débiles de nuestro sistema.

OBJETIVOS

OBJETIVOS GENERALES

- Brindar a nuestra sociedad una opción más de seguridad para sus bienes mediante la implementación de un sistema de seguridad confiable y fácil de usar.
- Crear un sistema de seguridad fácil de programar por cualquier persona.

OBJETIVOS ESPECIFICOS

- Utilizar una interface de programación sencilla con el fin de facilitar al usuario la introducción de datos al sistema.
- Que el usuario que ya hizo una inversión en la compra de una computadora, no gaste su dinero en sistemas de seguridad mucho más caros que tengan su propio microprocesador y memoria.

ANTECEDENTES

El primer prototipo fue un sistema sin microprocesador, el cual fue diseñado con la teoría recibida en cursos de electrónica digital integrada y electrónica I. Los temas estudiados fueron: Teoría de Fototransistores y Dispositivos de memoria y comparación digital.

En nuestro medio no es muy común ver sistemas de seguridad activados por tarjetas perforadas, los existentes son carísimos (entre 10,000 y 15,000 Colones entre los más comerciales) y están en lugares muy exclusivos.

Algunos de estos sistemas tienen la desventaja de ser difíciles de programar, ya que esto implica desarmar el dispositivo y mover una serie de interruptores o conectar algún tipo de interface con el fin de introducir datos a la memoria del sistema. Por lo que su programación debe ser hecha por una persona capacitada que tenga un amplio conocimiento del sistema.

La mayoría de los sistemas de seguridad activados por tarjeta perforada que se fabrican en el país quedan a nivel de exhibiciones técnicas, las cuales muestran las desventajas antes mencionadas, debido a que no se tiene un modelo estándar, fácil de programar por cualquier persona que no esté familiarizado por el equipo.

Los modelos más versátiles existentes de estos sistemas son controlados por una computadora; su fácil programación vía teclado hace que cualquier persona que no esté familiarizada con el funcionamiento electrónico del equipo pueda programar libremente el sistema.

En el presente proyecto se trata de dar una solución o una visión lo más general posible, de lo que un sistema de seguridad puede llegar a alcanzar, teniendo en cuenta aspectos tales como la efectividad del sistema, complejidad, el tamaño, el costo y la compatibilidad con los diversos sistemas de seguridad existentes en el mercado precisamente en el ámbito nacional.

Actualmente en El Salvador existen pocas compañías dedicadas a la venta de sistemas de seguridad ya sea para casas, bancos y comerciales teniendo en cuenta que todo el equipo que se vende en dichas compañías proviene del extranjero.

Investigando más a fondo no existe compañía alguna dedicada al diseño de sistemas de seguridad activados por tarjetas perforadas y controlada por software, y las compañías dedicadas a la venta de este tipo de equipo solo instalan y reparan en algunos casos. Generalmente el costo de estos equipos es demasiado elevado; por lo tanto, solo pueden ser adquiridos por instituciones o empresas de altos recursos económicos.

La idea del tema surge a raíz de una revisión de los aspectos mencionados anteriormente.

Por mencionar algunos sistemas de seguridad existentes en el país, en el caso de algunos bancos se utilizan sistemas de seguridad controlados por software en determinadas puertas, nada más que comparando el sistema de seguridad a diseñar, con un sistema de seguridad utilizado en los bancos se tiene la ventaja de que el sistema a diseñar por nosotros se comunica automáticamente con un servicio de seguridad en el caso de que alguien quiera simular una tarjeta perforada, mientras que en los bancos únicamente suena una alarma (en algunos casos) y comúnmente en pocos bancos se tiene que presionar un botón para comunicar al banco con una estación de seguridad para mandar una señal de ayuda.

Una de las finalidades que se tiene es hacer una buena conceptualización para cualquier persona interesada en el sistema dando criterios o ideas que muchas veces no se toman en cuenta o si se conocen son desvirtuadas por el diseñador, para el buen funcionamiento y operación del sistema.

Basados en lo anterior se trata de despertar actitudes y expectativas que pueden plantearse para intentar solucionar problemas reales que pueden darse en cualquier sistema de seguridad existentes en el país.

DESCRIPCION DEL PROYECTO

En general el sistema de seguridad abre o cierra una chapa eléctrica dependiendo si el código generado por una tarjeta perforada es correcto o no.

Todo el sistema es controlado por una computadora (PC) mediante un programa, éste se encarga de leer la tarjeta perforada, compara el código generado por dicha tarjeta con los códigos ya almacenados en la memoria de la computadora y manda señales de control de abrir o cerrar la chapa eléctrica.

El sistema tiene la capacidad de almacenar un número especificado de códigos correctos, los cuales son comparados todos al mismo tiempo con el código generado por la tarjeta perforada que se ha introducido en el sistema, es decir, se tiene la capacidad de 300 usuarios, todos con diferente código correcto, los cuales podrán abrir la chapa eléctrica. Cualquier tarjeta que genere un código distinto al almacenado en la memoria de la computadora no abrirá la chapa eléctrica.

El usuario tiene dos tipos de acceso:

Acceso por clave: Se introduce la tarjeta perforada y después una clave vía teclado; obviamente tiene que ser válida para que el sistema pregunte por una clave de acceso, si lo es, la chapa eléctrica se abrirá.

Acceso Directo: Se introduce la tarjeta perforada en el lector óptico sin necesidad de digitar una clave. La chapa eléctrica se abrirá si la tarjeta es válida.

Además se tienen 3 niveles de seguridad: .

Nivel Día/Noche: este proporcionara un acceso con horas preestablecidas dentro del programa principal, se tendrá un rango de horas de acceso fijas para un usuario nocturno y uno diurno.

Nivel por horario: Este nivel posee horas de acceso programables, aquí se especifica una hora de entrada y otra de salida en cualquier rango de tiempo del día o la noche.

Cualquier usuario que tenga una tarjeta valida pero trata de accesar al sistema fuera de las horas especificadas dentro de cualquier nivel de seguridad no podrá abrir la puerta.

Nivel de supervisor: Este tendrá una tarjeta maestra capaz de tener acceso al sistema a cualquier hora.

Cuando el sistema lee una tarjeta perforada valida, el programa registrara el nombre del usuario, la fecha y hora en que se inserto dicha tarjeta haciendo la función de reloj marcador, el sistema esperara una segunda inserción de la tarjeta, lo cual representara la hora de salida del usuario. Si se inserta la tarjeta una tercera vez, se le negará el acceso a la puerta aunque su tarjeta sea valida o esté dentro de su rango de horas de acceso (esto es en las mismas 24 horas).

Finalmente, se ha protegido al lector de tarjetas con un sistema de seguridad secundario. Este detecta si se está tratando de engañar al sistema mediante la introducción de objetos en la ranura de inserción con el fin de simular una tarjeta perforada.

Cualquier objeto o superficie que se introduzca en la ranura de inserción que no sea una tarjeta perforada activara un tipo de alarma, la cual consiste en mandar un mensaje vía puerto serie de la computadora hacia otra computadora el cual hará constar del problema ocurrido y además proporcionara la ubicación del establecimiento.

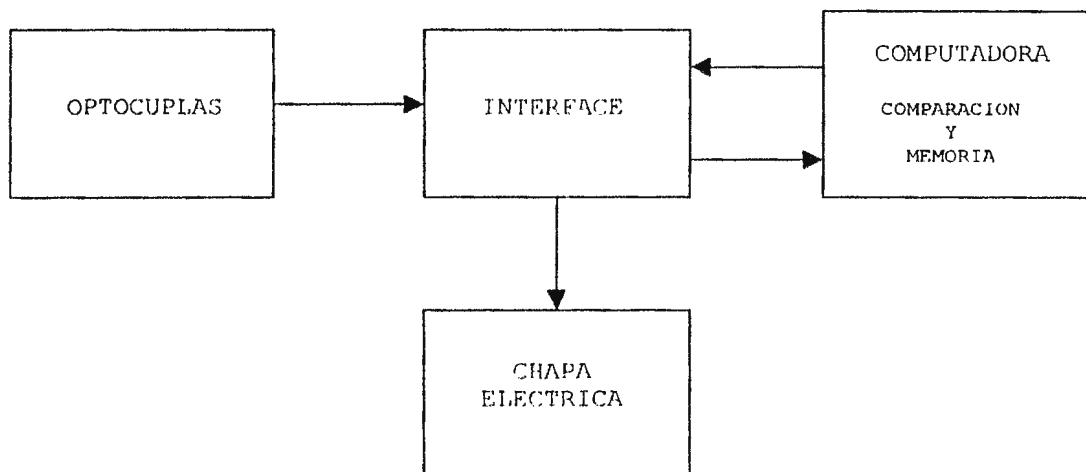


Figura 1. Diagrama de bloques

EXPLICACION DEL DIAGRAMA DE BLOQUES

Funcionamiento por bloques.

Cuando la tarjeta perforada se introduce en el bloque de optocuplas (Lector de tarjetas) éste genera un código binario, el cual viaja hasta la computadora gracias a la interface que se conecta entre las optocuplas y la computadora, una vez que el código este adentro, la computadora se encargará de comparar si el código que acaba de recibir es idéntico a uno de los códigos que están almacenados en su memoria, si el código recibido es igual a cualquiera de ellos, el programa mandará una señal a la interface que le indicará activar(Abrir) la chapa eléctrica; de lo contrario, la interface recibirá una señal de mantener cerrada la chapa eléctrica.

La etapa de optocopladores posee un subsistema de seguridad el cual protege de intrusos que intenten introducir objetos a la ranura de introducción de tarjetas perforadas.

El programa principal

El programa principal consta de 6 etapas:

Etapas de Almacenamiento de Códigos

Esta etapa se encarga de almacenar todos los códigos que abrirán la chapa eléctrica. Primero se especifica el numero total de usuarios, después el nombre de cada usuario con sus respectivos códigos de acceso, (Si el usuario utilizara un acceso por clave tendrá que almacenar su código de tarjeta perforada y su clave digital) y finalmente se especificara el nivel de seguridad para cada usuario.

Etapas de detección de tarjeta perforada

Esta rutina hará un chequeo de la información proveniente de la etapa de optocopladores. Cualquier objeto que se inserte en la ranura de inserción que no sea una tarjeta perforada, será

detectado por esta etapa activando un sistema secundario de seguridad.

Etapas de Lectura de Tarjeta Perforada

Aquí se lee la tarjeta insertada y se compara su código con los códigos almacenados dentro de la memoria del sistema. Si la tarjeta contiene un código válido, se pasa a la etapa de detección de acceso.

Etapas de Detección de Acceso.

Esta etapa detecta el tipo de acceso del usuario, si se detecta un acceso directo, el programa saltará a la etapa de registro y control de chapa eléctrica. Si se detecta un acceso por clave el programa irá hacia la etapa de introducción y comparación de claves.

Etapas de Introducción y Comparación de Claves.

Aquí el usuario introducirá su clave de acceso vía teclado, la cual será comparada con las demás claves almacenadas en la memoria del sistema. Si la clave es válida, el programa saltará a la etapa de registro y control de chapa eléctrica.

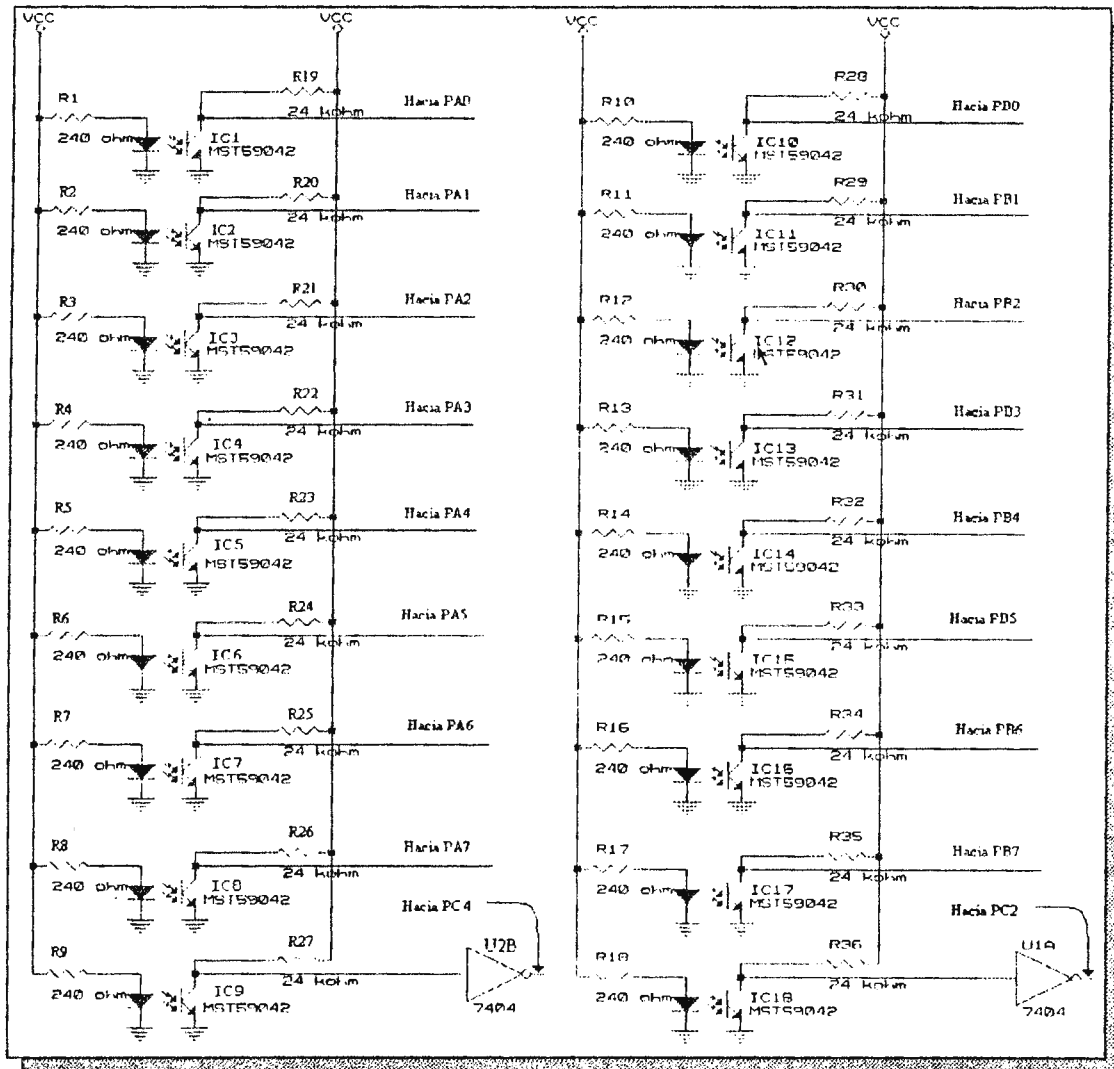
Etapas de Registro y Control de Chapa Eléctrica.

Esta etapa almacena el nombre del usuario y la fecha en que introdujo la tarjeta perforada en el lector óptico. Después el programa mandará una señal vía interface a la chapa eléctrica indicando que debe abrirse.

EXPLICACIÓN DEL CIRCUITO.

El circuito se divide en las siguientes partes las cuales son explicadas a continuación:

Etapas de optocouplas: .



Esta etapa esta compuesta por las optocouplas que van desde IC1 hasta IC18. Las optocouplas que están desde IC1 hasta IC9 forman la etapa de entrada y las optocouplas que están desde IC10 hasta IC18 forman lo que es la etapa de salida. A lo que se refiere cuando se habla de etapa de entrada y etapa de salida se

refiere a que estas optocuplas sensaran cuando una tarjeta haya sido introducida ya sea en la entrada o en la salida, estas optocuplas generaran un código binario el cual será leído por la PPI #1 en el puerto A y en el puerto B respectivamente, cuyo código pertenecerá al usuario correspondiente.

Para saber cuando una tarjeta ha sido introducida ya sea en la entrada o en la salida, las optocuplas IC9 e IC18 respectivamente se encargaran de generar un cambio de estado de 0 a 1, el cual es invertido por los inversores U2B e U1A para obtener un cambio de estado de 1 a 0, lo que se pretende con esto es generar un tipo de interrupción en el pin PC4 (STRA) y PC2 (STRB) de la PPI #1 para sensar cuando una tarjeta ha sido introducida , por lo tanto en el programa habrá una subrutina que este leyendo continuamente estos dos bits, para saber cuando se solicita servicio para abrir la puerta, si hay una interrupción de este tipo el programa saltara hacia otra subrutina que mas adelante se explica en programa del sistema.

INTERFACE PERIFERICA PROGAMABLE 1 (PPI #1).

Esta es una interface periférica programable, la cual se encarga de leer el dato correspondiente al código generado por las optocuplas ya sea las de entrada o las de salida, un punto a tomar en cuenta en este sistema es que la etapa de optocuplas que tendrá mayor prioridad será la etapa de salida por lo tanto si ambas tarjetas son insertadas al mismo tiempo la de mayor prioridad será la de salida.

En esta PPI todos los puertos trabajan en el modo de operación 1(entrada mediante habilitación).

Las palabras de comando utilizadas para su debida programación son las siguientes:

Byte A.

7	6	5	4	3	2	1	0	
1	0	1	1	1	1	1	1	#SBFh

- Bit 0 = 1.

Puerto C entrada (PC3 a PC0).

- Bit 1 = 1.

Puerto B entrada.

El puerto B es el encargado de leer el código generado por las optocuplas que van desde IC10 hasta IC17 las cuales forman la etapa de salida.

- Bit 2 = 1.

Selección del modo 1 para el grupo B.

- Bit 3 = 1.

Puerto C entrada(PC7 - PC4).

Los pines PC7 y PC6 son pines de entrada los cuales son los encargados de verificar cuando se ha activado tanto la etapa de sensores como la etapa de cierre de puerta.

- Bit 4 = 1.

Puerto A entrada.

El puerto A es el encargado de leer el código generado por las optocuplas que están desde IC1 hasta IC8 los cuales forman la etapa de entrada.

- Bit 5 = 1 y Bit 6 = 0.

Estos bits seleccionan el modo de operación de la PPI, para este caso se eligió el Modo 1 para el grupo A.

- Bit 7 = 1.

Selección del byte A

Byte B.

7	6	5	4	3	2	1	0	
0	0	0	0	0	1	0	1	#\$05H
0	0	0	0	1	0	0	1	#\$09H

- Bit 0 = 1.

Activación de bits.

Con este byte lo que se hace es activar o habilitar el bit especificado por los bits 1, 2, 3.

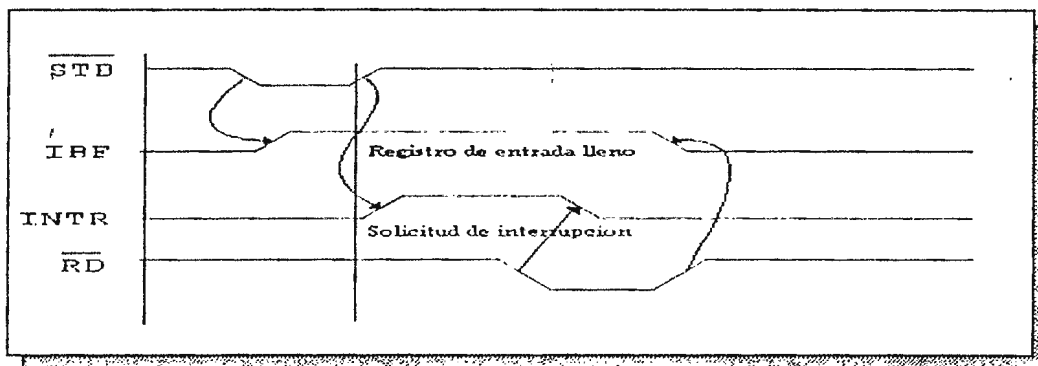
- Bit 1, 2, 3.

Selección del bit ha utilizar los cuales pueden ser 0,1,2,3,4,5,6,7.

3	2	1	Bits
0	1	0	Habilitación de STBB (PC2).
1	0	0	Habilitación de STBA (PC4).

La habilitación de STBA y STBB servirá para verificar cuando una tarjeta haya sido introducida. ¿Pero como se logra esto?

A continuación se muestra un diagrama de tiempo donde se puede observar los cambios de estado en STBB y STBA y los cambios que estas entradas producen en las demás señales.



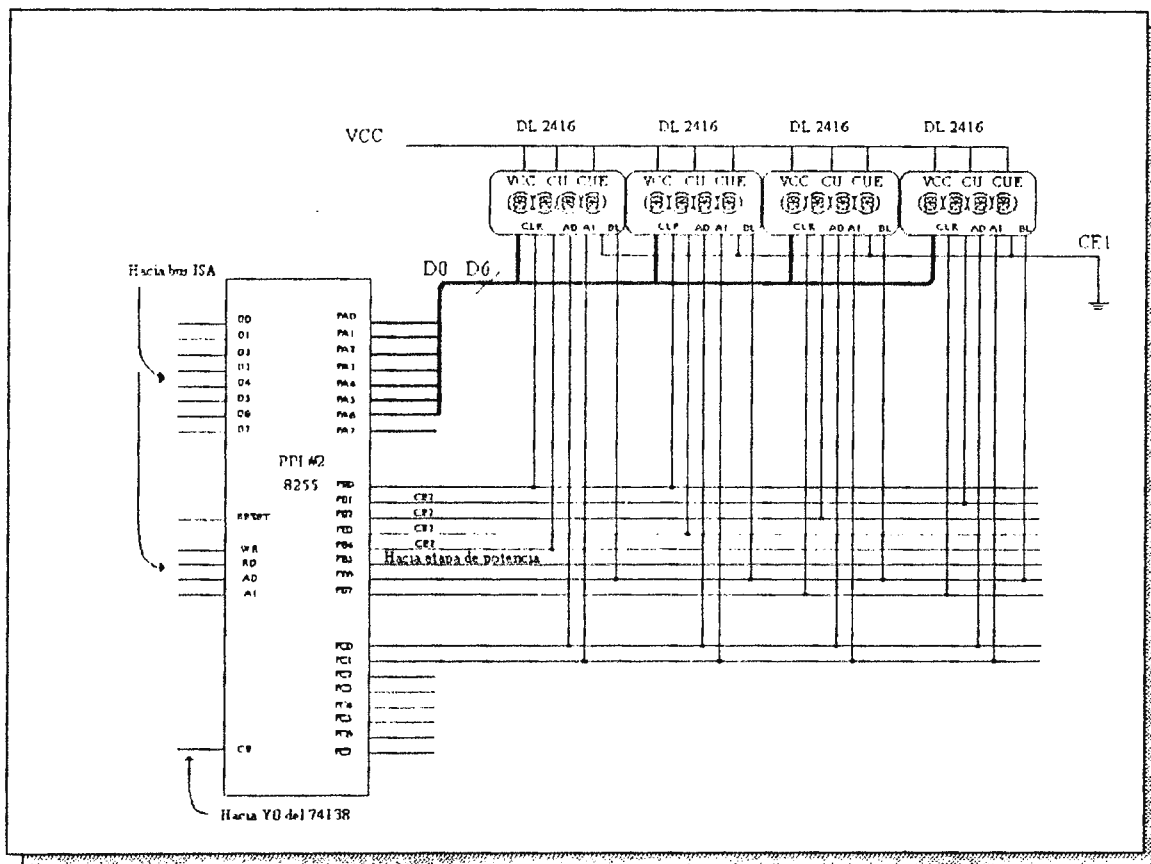
Si se observa en el diagrama de temporizacion un cambio de 1 a 0 en la línea STB provoca un cambio en el bit IBF(Input buffer Full) la cual es una salida que indica que el registro de entrada contiene información. Específicamente la información retenida se encontrara tanto en el puerto A como en el puerto B, ya que a estos dos puertos están conectados a las optocuplas las cuales además de generar la interrupción en STB, generan el dato binario que será leído en el puerto correspondiente.

El puerto A leerá las optocuplas que servirán para indicar cuando un usuario necesita entrar, y el puerto B leerá código correspondiente al usuario que solicita salir.

- Bit 4, 5, 6.

- Bit 7 = 0.

INTERFACE PERIFERICA PROGAMABLE 2 (PPI #2).



13

Con esta PPI lo que se logra es controlar lo cuatro visualizadores alfanuméricos, los cuales se encargan de proporcionar los mensajes necesarios para la factibilidad de uso del sistema por parte del usuario tanto en la etapa de entrada como en la etapa de salida.

Palabra de comando.

7	6	5	4	3	2	1	0	
1	0	0	0	1	0	0	0	#88H

- Bit 0 = 0. Selección del Puerto C como salida.
- Bit 1 = 0. Selección del Puerto B como salida.
- Bit 2 = 0. Selección de modo 0 para el grupo B.
- Bit 3 = 1. Selección de la parte alta del Puerto C (PC4 - PC7) como entrada.
- Bit 4 = 0. Selección del puerto A como entrada.
- Bit 5 = 0 y Bit 6 = 0. Selección del modo de operación de la PPI para este caso modo 0.
- Bit 7 = 1. Selección del Byte A.

El tipo de display ha utilizar son visualizadores inteligentes alfanuméricos que incorporan decodificadores, multiplexores, memorias y excitadores.

Tipo de visualizador: DL 2416. Cada visualizador posee 4 display alfanuméricos.

Lo que se hace con la PPI es generar las diferentes señales de control para operación de los display.

Las señales de control que se generan con la PPI son las siguientes:

- D0 - D6: Hilos de datos.

Estos siete hilos de entrada de datos reciben caracteres codificados en código ASCII generado por los pines PA0 hasta PA6 de la PPI 2.

- A0 - A1: Hilos de dirección.

La dirección determina la posición del visualizador en que se inscribe, esta dirección es generada por los pines PC0 y PC1 de la PPI 2.

- CE2: Chip enable 2. (Activo en bajo)

Estas es una de las 2 entradas de habilitación que posee cada visualizador para poder ser habilitado. Como son cuatro visualizadores estas señales son generadas por los pines PB1, PB2, PB3 y PB4 de la PPI 2.

PB4	PB3	PB3	PB1	DL2416 - 4	DL2416 - 3	DL2416 - 2	DL2416 1
1	1	1	0	Deshabilitado	Deshabilitado	Deshabilitado	Habilitado
1	1	0	1	Deshabilitado	Deshabilitado	Habilitado	Deshabilitado
1	0	1	1	Deshabilitado	Habilitado	Deshabilitado	Deshabilitado
0	1	1	1	Habilitado	Deshabilitado	Deshabilitado	Deshabilitado

- CE1: Chip enable 1. (Activo en bajo)

Entrada directamente conectada a tierra en todos los visualizadores.

- CLR: Borrar (Clear, nivel bajo activo)

Si durante 15 ms, en esta entrada hay un nivel bajo la memoria de datos se borra, esta señal se genera con el pin PB0 y PB7 de la PPI #2, tanto para los visualizadores de entrada como para los visualizadores de salida respectivamente.

- CUE : liberación del cursor (cursor enable).

Esta entrada esta directamente conectada a VCC para la activación del cursor.

- CU: Esta entrada debe estar en nivel alto para la carga de datos en la memoria de datos, y en nivel bajo para la carga de datos en la memoria del cursor. En este caso esta entrada esta directamente conectada a VCC.

- BL: Oscurecimiento (display blank, nivel bajo activo)

Si esta entrada se mantiene en nivel bajo, todo el visualizador se mantiene sin ningún símbolo. Tan pronto como el nivel de BL cambia a alto se vuelven a visualizar los caracteres memorizados. Esta señal se genera con el pin PB6 de la PPI 2.

La siguiente tabla muestra el juego de caracteres codificados en código ASCII para su debida interpretación.

			D0	L	H	L	H	L	H	L	H	L	H	L	H	L	H
			D1	L	L	H	H	L	L	H	H	L	L	H	H	L	L
			D2	L	L	L	L	H	H	H	H	L	L	L	L	H	H
			D3	L	L	L	L	L	L	L	L	H	H	H	H	H	H
D6	D5	D4		0	1	2	3	4	5	6	7	8	9	A	B	C	D
L	H	L	2		!	"		\$	%	&	'	<	>	*	+	,	-
L	H	H	3	0	1	2	3	4	5	6	7	8	9		:		=
H	L	L	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M
H	L	H	5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]

¿CÓMO SE CONTROLAN LOS VISUALIZADORES?

Mensajes que se obtendrán en los visualizadores:

- INSERTAR TARJETA
- MOMENTO
- CLA. O DIR.
- INGRESAR CLAVE
- **** (La longitud de asteriscos depende de la longitud de la clave)
- BUSY
- ACCESO DENEGADO
- RETIRAR TARJETA
- GRACIAS

Todos los mensajes presentados anteriormente podrán ser visualizados en los displays, algunos de los mensajes estarán rotando hacia la izquierda continuamente, otros permanecerán fijos todo dependiendo de la operación que se este realizando.

A continuación se muestra un ejemplo de cómo poder cargar un carácter en uno de los visualizadores: .

Nota: cada visualizador posee cuatro display.

Visualizador 1 (DL2416 - 1)



Suponiendo que la figura anterior representa un visualizador en el cual se encuentra grabada la letra "I". ¿Cómo se hizo o como se logro cargar esta letra?

La PPI encargada de manejar los visualizadores es la PPI #2 es decir que el puerto A es el encargado de generar el dato en código ASCII, el puerto B es el encargado de la habilitación del visualizador y el puerto C genera la dirección.

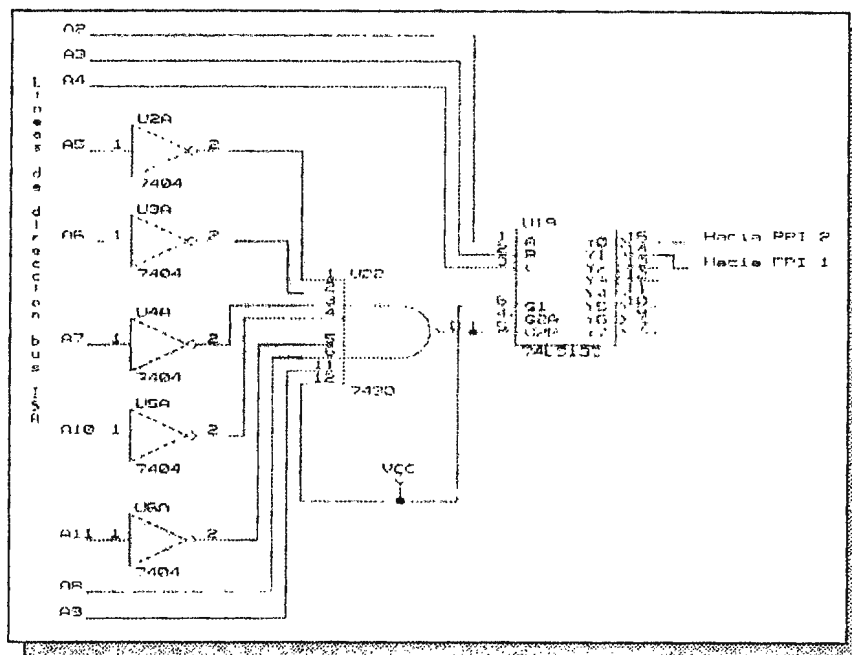
Por lo tanto si se quiere cargar la letra "I" en el display 1 lo que se hace es lo siguiente:

1. Habilitar visualizador 1. Dato a cargar en el puerto B = XXX1110X específicamente PB1 es el encargado de controlar el visualizador 1 por lo tanto este bit debe de estar en 0 para habilitar el visualizador o que los cuatro display del visualizador 1 permanezcan habilitados.
2. Posteriormente poner la dirección, como el puerto C es el que controla la dirección el dato a cargar en el puerto C será de XXXXXX00. Específicamente son los pines PC0 y PC1 los encargados de controlar la dirección.
3. Luego de los dos pasos anteriores se carga la letra en el puerto A para este caso el dato a cargar seria 49 que es el código ASCII de la letra "I".

Si se quiere borrar esa letra lo único que se hace además de los primeros dos pasos mencionados anteriormente es cargar el código 20H en el puerto A con lo cual el display queda en blanco, si desea cargar la letra "I" en el display 2, lo que se tiene que

hacer es solo cambiar la dirección generada por el puerto C, para este caso seria XXXXXX01 y repetir los pasos 1 y 3.

ETAPA DECODIFICADORA.



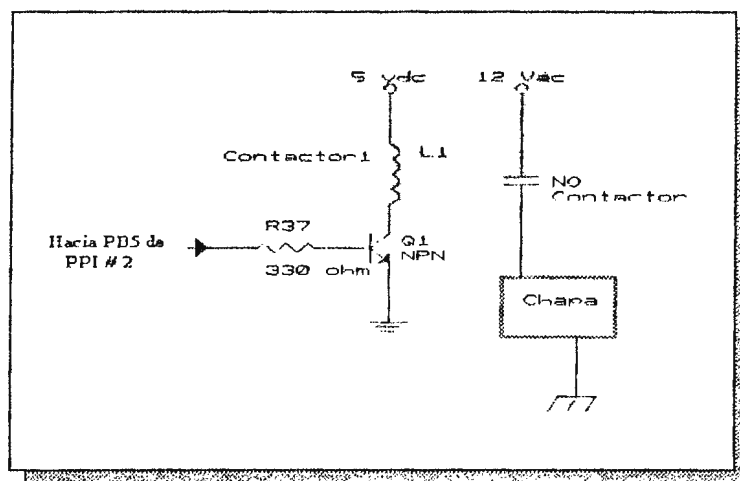
Nota: Las líneas de dirección A0 y A1 del bus ISA van hacia las entradas A0 y A1 de la PPI #1.

Mapa de memoria.

Direcciones	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	PPI #1	PPI #2
300h	0	0	1	1	0	0	0	0	0	0	0	0	DESHAB.	PORT A
301h	0	0	1	1	0	0	0	0	0	0	0	1	DESHAB.	PORT B
302h	0	0	1	1	0	0	0	0	0	0	1	0	DESHAB.	PORT C
303h	0	0	1	1	0	0	0	0	0	0	1	1	DESHAB.	COMMAND
304h	0	0	1	1	0	0	0	0	0	1	0	0	PORT A	DESHAB.
305h	0	0	1	1	0	0	0	0	0	1	0	1	PORT B	DESHAB.
306h	0	0	1	1	0	0	0	0	0	1	1	0	PORT C	DESHAB.
307h	0	0	1	1	0	0	0	0	0	1	1	1	COMMAND	DESHAB.

Como se puede observar en la figura del decodificador y en el mapa de memorias se tienen las direcciones correspondientes a las dos interfaces periféricas en las cuales se tiene en cuenta las direcciones de los puertos los cuales son: puerto A, puerto B y puerto C. Además se puede observar la dirección de la palabra de comando de ambas PPI's las cuales sirven para configurar o programar las interfaces periféricas.

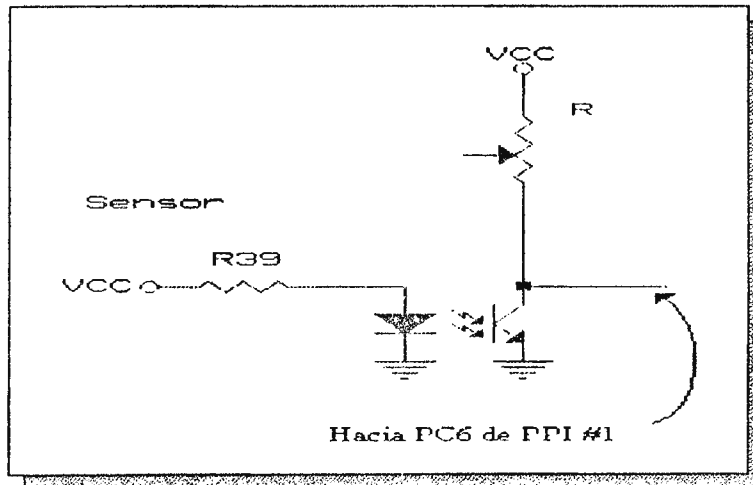
ETAPA DE POTENCIA.



El diagrama anterior muestra la etapa de potencia la cual es la encargada de controlar lo que es la Chapa Eléctrica. Como se puede observar en el diagrama una señal proveniente de PB5 de la PPI #2 proporciona un voltaje de control (+5Vdc), para saturar el transistor Q1, al saturarse este transistor fluye una corriente a través de la bobina del relay 1 lo cual hace que se cierre el contactor 1 dejando pasar corriente por la bobina de la chapa, activándose y abriéndose la puerta, cuando la puerta esta abierta no es necesario que la chapa continúe energizada pero si es necesario sensar cuando la puerta es abierta, esto es logrado por la etapa de cierre de puerta, por lo tanto el pulso de control (+5Vdc) solo dura unos cuantos segundos, luego pasa a cero voltios para desactivarla y poder cerrar la puerta.

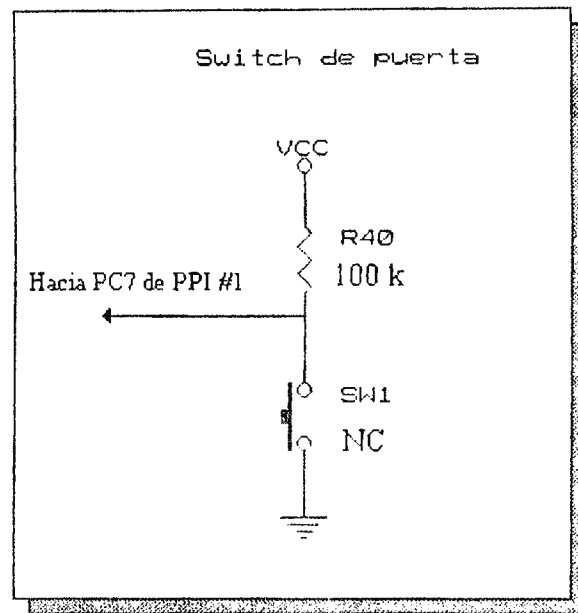
ETAPA DE SENSORES.

En esta etapa lo que se realiza es el control de las personas que entran y salen del cuarto de seguridad esto se hace mediante el circuito mostrado en la figura siguiente:

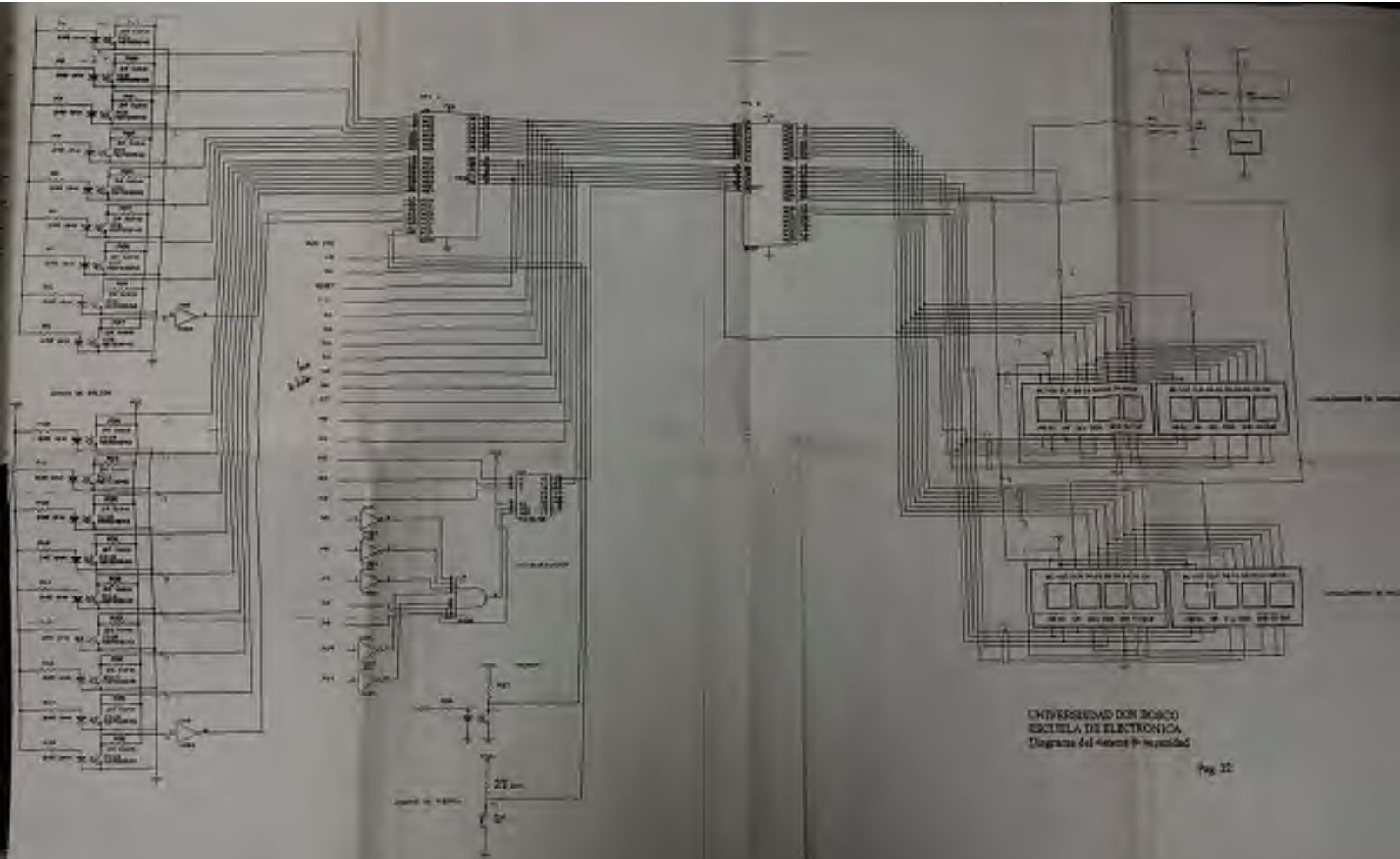


El funcionamiento del circuito es detectar cuando una persona ha pasado por la puerta. Un haz de luz mantiene en condiciones de saturación al fototransistor mostrado en la figura, media vez que el haz de luz es interrumpido ocasiona que el transistor entre en estado de corte poniendo 5 Vdc en la salida, estos 5 Vdc son tomados como un 1 lógico, el cual se dirige hacia PC6 de la PPI #1, este cambio de estado de 0 a 1 es leído por la PPI #1 lo cual ocasiona que se incremente o decremente un contador colocado en el programa, pero hay que tomar en cuenta que el incremento y decremento del contador va a depender de si este es una entrada o es una salida, si la persona va a entrar el contador se incrementa pero si la persona va a salir el contador se decrementa. Mas que todo este simple circuito sirve para contar el numero de personas que salen y entran en el cuarto de seguridad.

ETAPA DETECTORA DE CIERRE DE PUERTA.



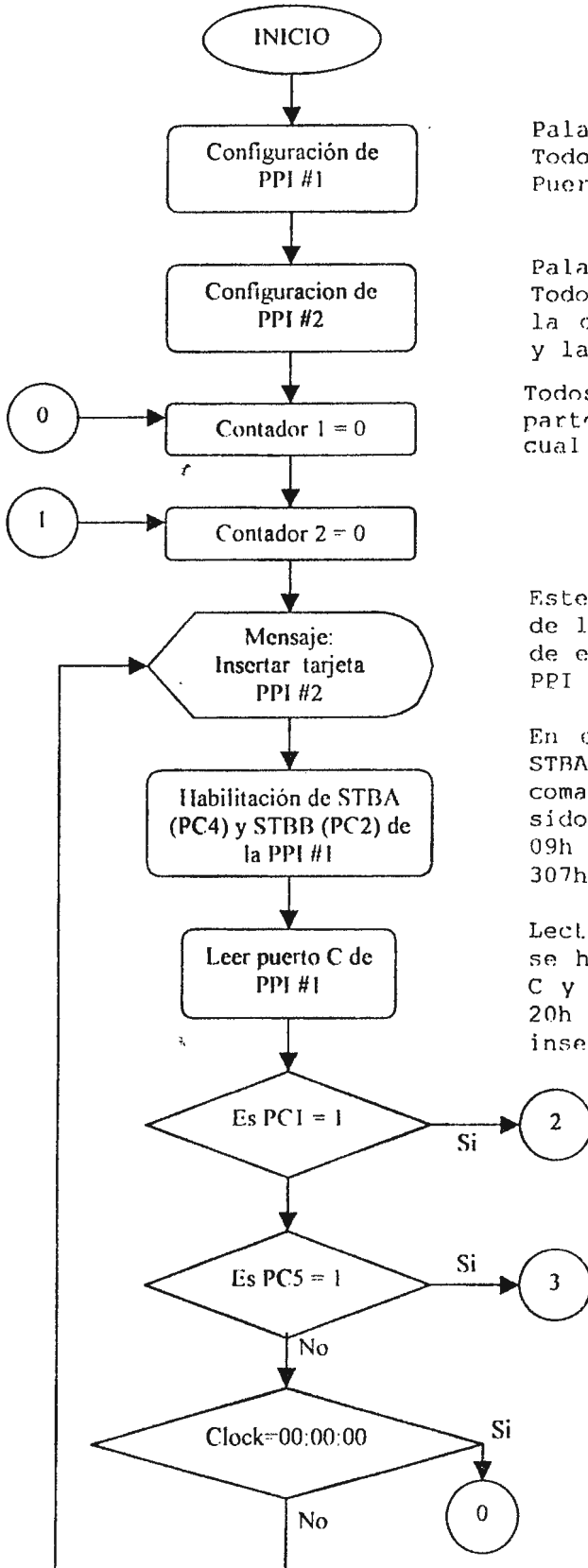
Con esta etapa se pretende detectar cuando la puerta ha sido abierta o cerrada, mediante el micro switch SW1 el cual estará escondido en la puerta para que no sea tan fácil de encontrar. Si la puerta esta abierta SW1 estará abierto colocando un 0 lógico en la entrada PC7 de la PPI #1, debido ha esto el sistema no podrá inicializarse, hasta que el pulso cambie de 0 lógico a 1 lógico, lo cual ocasionara que el sistema vuelva a inicializarse.



UNIVERSIDAD DON BOSCO
ESCUELA DE ELECTRONICA
Diagrama del 4-bit adder con carry-in y carry-out

FLUJOGRAMA DEL PROGRAMA.

Etapla detectora de tarjetas.



Palabra de comando: BFh
Todos los puertos en modo 1. Puerto A, Puerto B y Puerto C entradas.

Palabra de comando: 88h
Todos los puertos en modo 0. Esta PPI es la que controla todos los visualizadores y la etapa de potencia.

Todos los puertos son salidas excepto la parte alta del puerto C (PC4 - PC7) la cual es entrada.

Este mensaje es visualizado en el monitor de la computadora y en los visualizadores de entrada y de salida controlados por la PPI #2.

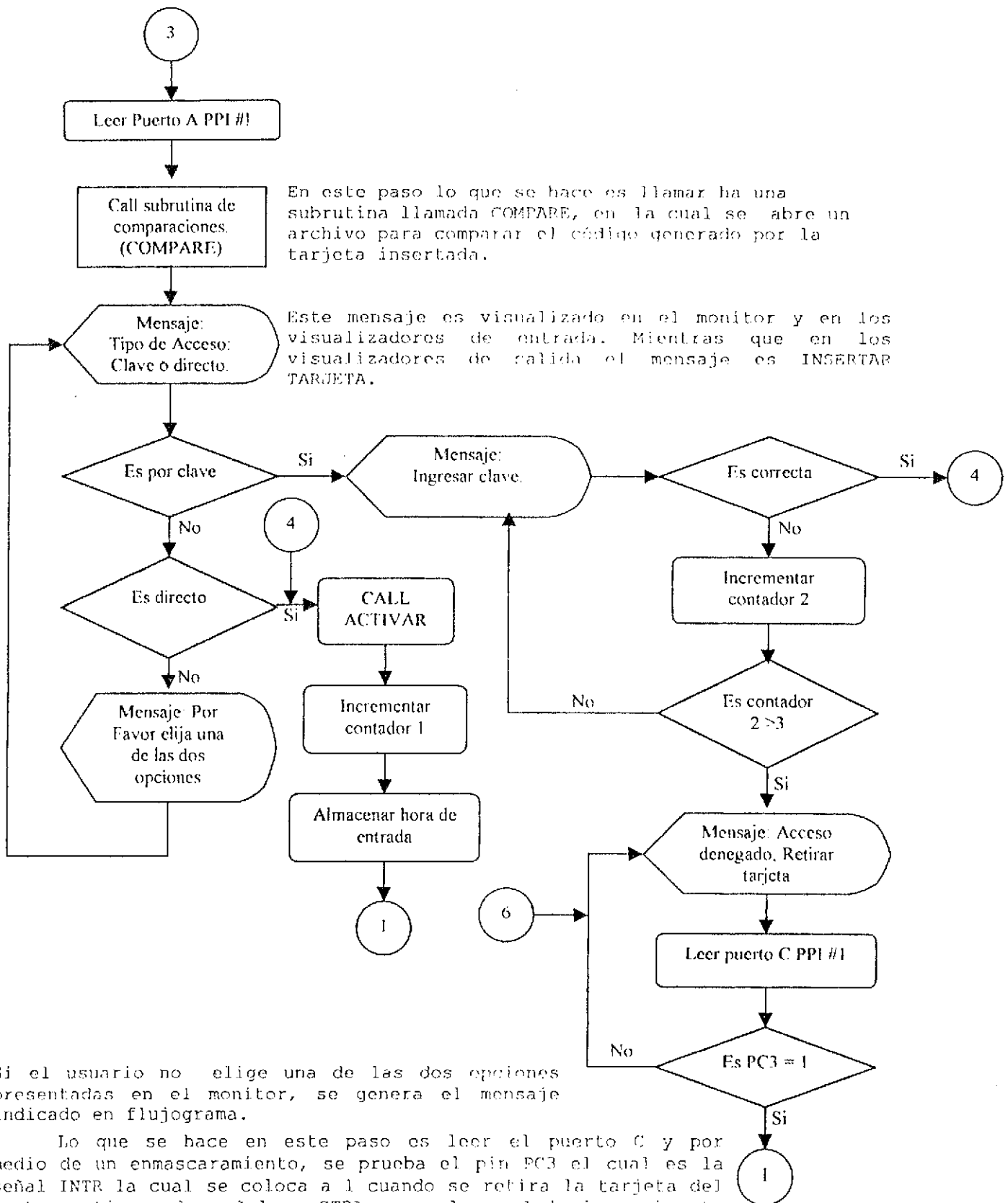
En este paso lo que se hace es activar STBA y STBB por medio del byte de comando para saber cuando una tarjeta ha sido introducida. Palabras de comando: 09h y 05h programadas en la dirección 307h (palabra de comando).

Lectura del puerto C. En este paso lo que se hace es capturar el dato en el puerto C y realizar una operación AND con 02h y 20h para saber si hay una tarjeta insertada. Proceso de enmascaramiento.

Verificar si IBFB=1. Si el dato de la operación de enmascaramiento es 02h salta hacia la etapa de salida.

Verificar si IBFA=1. Si el resultado de la operación AND es igual a 20h salta hacia la etapa de entrada de lo contrario sigue en el lazo indicado por el flujograma.

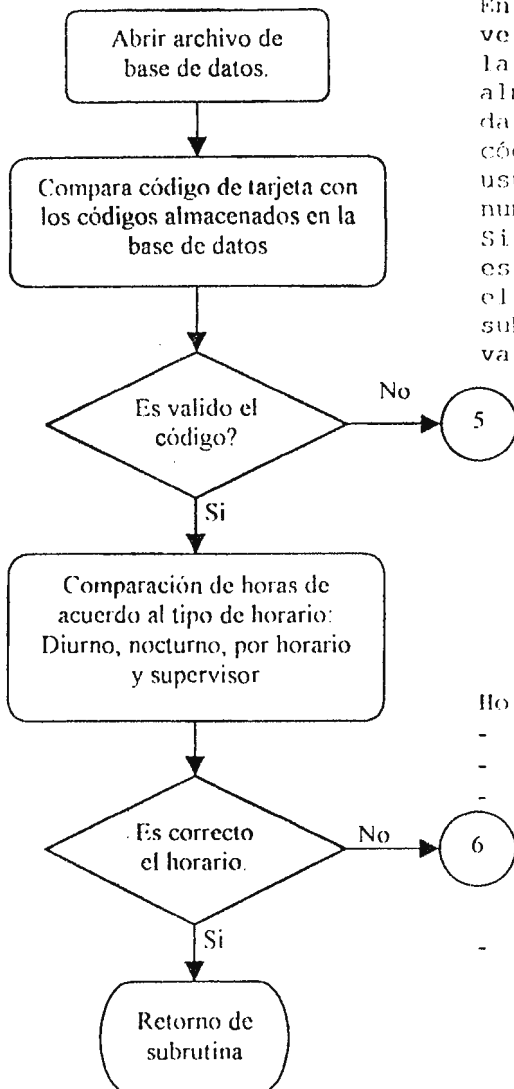
Etapa de entrada.



Si el usuario no elige una de las dos opciones presentadas en el monitor, se genera el mensaje indicado en flujograma.

Lo que se hace en este paso es leer el puerto C y por medio de un enmascaramiento, se prueba el pin PC3 el cual es la señal INTR la cual se coloca a 1 cuando se retira la tarjeta del lector optico o la señal en STRA se vuelve a 1 logico, si esta es 1 se vuelve ha reinicializar el sistema de lo contrario permanece en el lazo INTR se limpia media vez se lee el puerto.

Subrutina COMPARE.



En esta subrutina lo que se hace es verificar que el código correspondiente a la tarjeta introducida se encuentre almacenado en el archivo de la base de datos, en el cual se encuentran todos los códigos validos de las tarjetas, nombre de usuarios, horas de entrada y de salida y el numero de veces que ha ingresado.

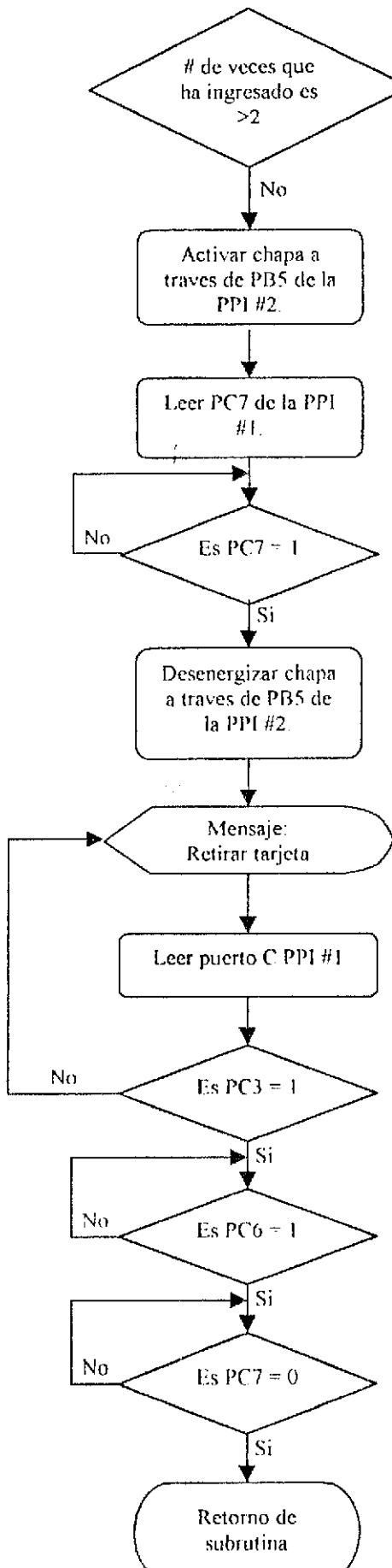
Si el código generado no se encuentra o no es valido salta la rutina especificada por el conector 5 la cual corresponde al subsistema de seguridad, si el código es valido hay un retorno de subrutina.

En esta misma subrutina se verifica a que tipo de usuario corresponde la tarjeta es decir: Nocturno, diurno, por horario y supervisor. Si el horario no es el correcto se envía por medio de los visualizadores y del monitor de la computadora un mensaje de "ACCESO DENEGADO". Ver hacia donde se dirige el conector 6 para una mejor comprensión.

Horarios:

- Diurno: 7:00 a.m. - 4:00 p.m.
- Nocturno: 8:00 a.m. - 4:00 a.m.
- Por horario: De acuerdo al horario escogido por el usuario.
- Supervisor: puede accesar a cualquier hora del día.

Subrutina ACTIVAR.



Lo primero que se hace en esta subrutina es verificar que el usuario no haya entrado mas de dos veces de lo contrario el acceso será denegado.

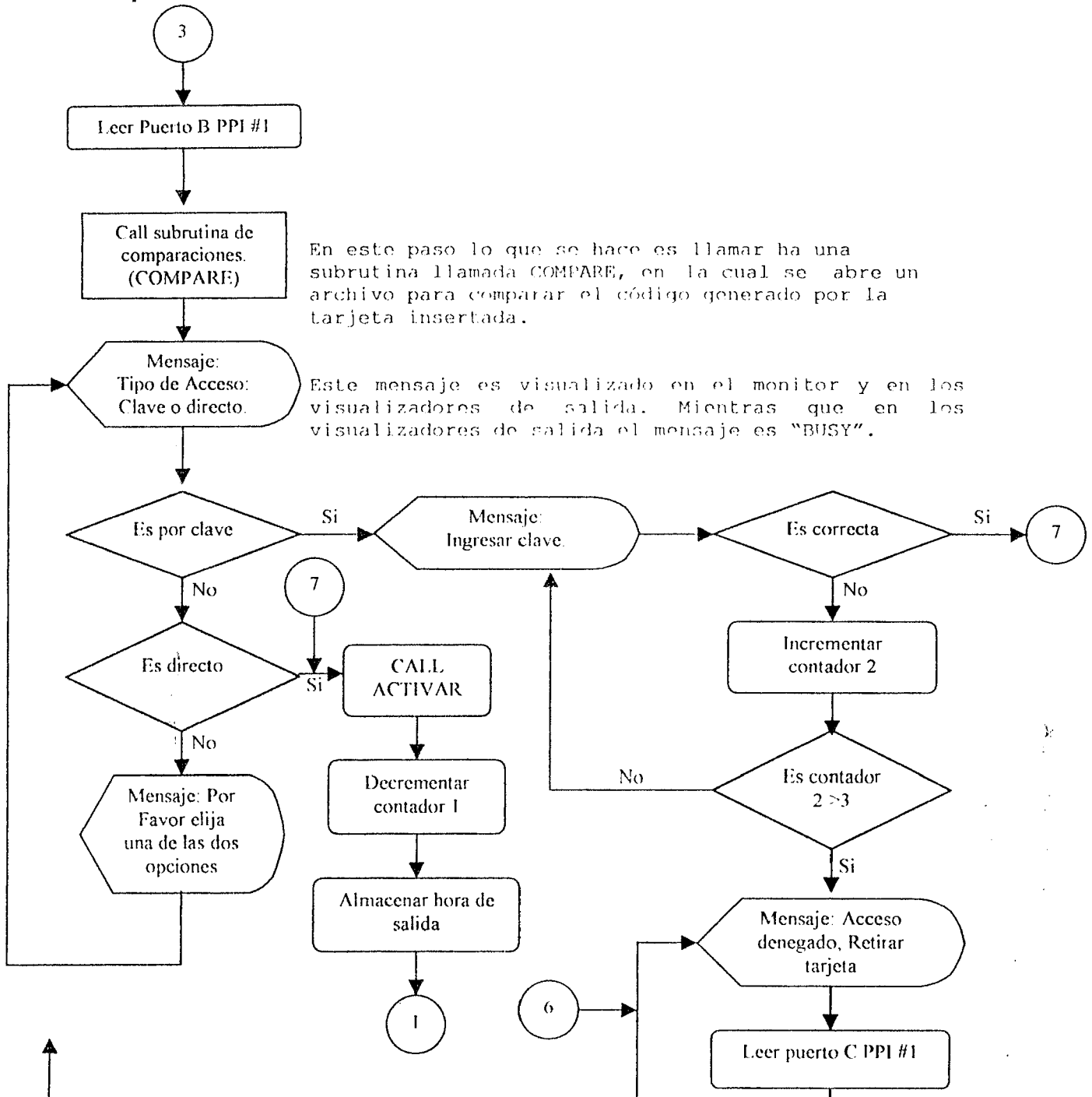
En esta subrutina lo que se hace es activar la chapa eléctrica a través del pin PB5 de la PPI #2, una vez que la chapa ha sido activada se debe esperar ha que la puerta se abra, esto se logra probando el pin PC7 de la PPI #1, cuando este pase de 0 a 1 esto indica que la puerta ha sido abierta por lo tanto no es necesario que la chapa siga energizada.

Lo que se hace en este paso es leer el puerto C y por medio de un enmascaramiento, se prueba el pin PC3 el cual es la señal INTR la cual se coloca a 1 cuando se retira la tarjeta del lector óptico o la señal en STBA se vuelve a 1 lógico.

Lo que se hace en esta etapa es probar el sensor que esta colocado a la entrada PC6 de la PPI #1 para saber cuando una persona ha entrado. Media vez la persona ha entrado se verifica que la puerta haya sido cerrada verificando el pin PC7 de la PPI #1, el cual esta conectado a la salida de la etapa de cierre de puerta. Cuando la señal en PC7 es un 0 lógico esto indica que la puerta ha sido cerrada por lo tanto se sale del lazo y hay un retorno de subrutina.

Despues de hacer el retorno de subrutina se almacena la hora en que se dio una entrada o una salida, además se incrementa o decrementa el contador 1, esto se realiza para llevar un control de las personas que entran o salen del cuarto.

Etapa de salida.



Si el usuario no elige una de las dos opciones presentadas en el monitor, se genera el mensaje indicado en flujograma.

Lo que se hace en este paso es leer el puerto C y por medio de un enmascaramiento, se prueba el pin PC0 el cual es la señal INTRB la cual se coloca a 1 cuando se retira la tarjeta del lector óptico o la señal en STBB se vuelve a 1 lógico, si esta es 1 se vuelve a reinicializar el sistema de lo contrario permanece en el lazo INTR hasta que se coloca en ese estado. Este bit se limpia media vez se lee el puerto.

Tanto en la etapa de entrada como en la etapa de salida puede observarse que existe un paso en el cual se almacena la hora de entrada y la hora de salida. Esto se hace para llevar un control de las personas que ingresan o salen del cuarto a determinada hora.

La hora de ingreso o de salida es almacenada en un archivo, el cual contiene solo el código del usuario y la hora y fecha en que este individuo ingreso o salió por ultima vez.

Etapas del subsistema de seguridad.

En esta etapa lo que se pretende es mandar un mensaje a un cuerpo de seguridad para informarles que hay una emergencia en cierta localidad por lo tanto se necesita de su ayuda, proporcionándoles la dirección del local.

En la defensa solo se simulara esa llamada.

Existen varios métodos de simular esa llamada uno de ellos es la simulación de la llamada o, el envío del mensaje utilizando el microcontrolador MC68HC11 para fines didácticos, lo que se hará en esta etapa es utilizar la SCI(Sistema de comunicación asincrono) del Microcontrolador para enviar un mensaje por el puerto serie de la computadora estableciendo una comunicación entre la tarjeta del microcontrolador y la PC.

Prácticamente serán dos computadoras utilizadas para la comunicación:

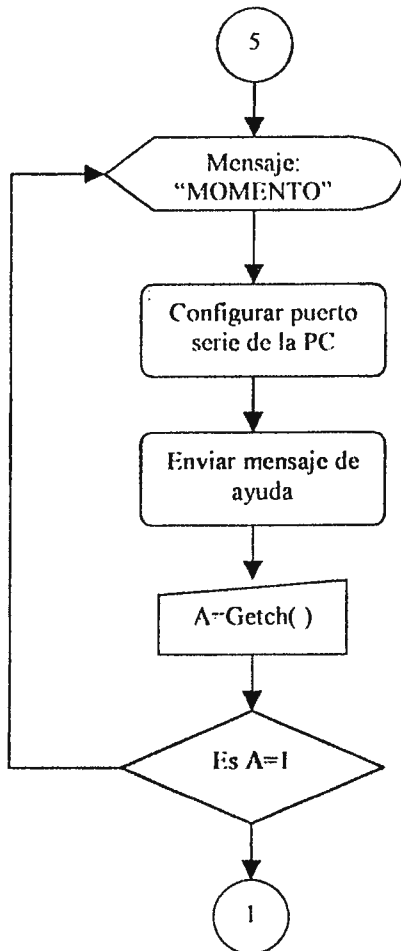
Una computadora donde se encuentre el programa principal (Sistema) y otra computadora en la cual se encuentre conectada la tarjeta EVB MC68HC11 en la cual estará residente un programa, el cual hace la función de leer datos en serie proporcionados por la computadora del programa principal.

Lo bueno es que en la computadora donde esta conectada la EVB MC68HC11 no será necesario que el programa este corriendo continuamente sino que el usuario puede estar trabajando en cualquier programa pero si hay una interrupción el programa residente en la EVB se ejecutara leyendo el mensaje proporcionado por el puerto serie de la otra computadora a través de la SCI (Sistema de comunicación asincrono) y.

almacenándolo en un bloque de memoria o mandándolo al monitor de la computadora 2.

Otro método es el de utilizar los modems de las computadoras.

En la pagina siguiente se muestra una parte del flujograma la cual hace énfasis al subsistema de seguridad.



Para la configuración del puerto serie se emplea la interrupción int 14h en la cual se inicializa el puerto especificándole la velocidad en baudios, la paridad, el bit de paro y la longitud de la palabra

La función getch lo que hace es capturar un dato del teclado y lo almacena en la variable A, si el dato capturado es indiferente de 1 sigue en el lazo, mandando el mensaje hacia la otra computadora de lo contrario se sale del lazo y se reinicializa el sistema. Esto se hace para entretener al intruso, a modo de que él al apretar una tecla este visualizando en los displays el mensaje de "MOMENTO" y crea que ha podido ingresar al sistema.

JUSTIFICACION

Las chapas eléctrica
s o electromecánicas en la mayoría de los casos son abiertas por switches ocultos los cuales únicamente el personal de seguridad o las personas encargadas de abrir la puerta tienen acceso a ellos. Este proceso es muy tedioso debido a que se tiene que estar oyendo un timbre o vigilando la puerta cada vez que alguien autorizado quiera tener acceso a ella. Las chapas eléctricas para estos casos pueden ser mejor aprovechadas si cada persona que quiera tener acceso a una puerta con dichas chapas tenga también una llave eléctrica.

Una chapa abierta por una tarjeta perforada es un buen ejemplo de ello, con esto el usuario tiene acceso las veces que él desee y a cualquier hora al sitio protegido por esta chapa tan sólo con insertar una tarjeta en un lector electrónico.

Este sistema es aun más seguro debido a que el intruso además de violar la parte mecánica de la chapa, también tiene que trabajar su control electrónico, con el cual él no está familiarizado.

Los sistemas de seguridad activados por tarjetas perforadas en nuestro país tienen la principal desventaja de ser difíciles de programar y solo el personal especializado o familiarizado con el sistema puede tener acceso a su programación.

En vista de lo anterior, se puntualizan las siguientes características que justifican el desarrollo del sistema de seguridad planteado en este documento

- **Fácil programación:** El sistema propuesto puede ser programado por cualquier persona, debido a que esto se realiza mediante una computadora, es decir, se introducen los datos fácilmente con un teclado, evitándonos desarmar el equipo e introducir datos con alguna interface especial o con el movimiento de minúsculos switches minidip como en la mayoría de los sistemas de seguridad existentes. También se tiene la ventaja de no trabajar a ciegas debido a que todos los datos introducidos son visualizados en la

pantalla de la computadora.

- **Bajo Costo:** Debido a que se quiere aprovechar la inversión de alguna empresa o algún usuario que haya comprado una computadora, ahorrándole así, comprar sistemas de seguridad que poseen fuente propia, microprocesador y memoria, los cuales les resultarían mas caros.
- **Mayor Velocidad y eficiencia:** También se quiere aprovechar la rapidez del microprocesador de la computadora y su capacidad de memoria, lo que hace al sistema mucho más confiable comparado con los sistemas existentes en el mercado.

MARCO TEORICO.

PUERTO SERIE.

Terminales de conectores serie.

Conector de p terminales del puerto serie (AT)			
Terminal	Descripción	Señal	Dirección
1	Detector de portadora	CD	Entrada
2	Recepción de datos	RD	Entrada
3	Transmisión de datos	TD	Salida
4	Lista la terminal de datos	DTR	Salida
5	Tierra de la señal	SG	-
6	Listo el conjunto de datos	DSR	Entrada
7	Petición para transmitir	RTS	Salida
8	Borrar para transmitir	CTS	Entrada
9	Indicador de timbre	RI	Entrada

Inicialización del puerto de comunicación.

AH = 00h

AL = parámetros de inicialización

DX = Puerto (0 = COM1, 1 = COM2, etc.)

D7	D6	D5	Velocidad (baudios)
0	0	0	110 bd
0	0	1	150 bd
0	1	0	300 bd
0	1	1	600 bd
1	0	0	1200 bd
1	0	1	2400 bd
1	1	0	4800 bd
1	1	1	9600 bd

D4	D3	Paridad
0	0	Ninguna
0	1	Impar
1	0	Par

D1	D0	Longitud de palabra
0	0	5 bit's
0	1	6 bit's
1	0	7 bit's
1	1	8 bit's

Transmisión de datos.

Entrada:

AH = 01h

AL = Carácter a transmitir

Salida:

AH: bit 7 = 0 Transmisor vacío

Bit 7 = 1 Transmisor lleno.

OPTOCUPLAS.

El bloque de optocuplas es un dispositivo que genera un código binario dependiendo de las perforaciones hechas en la tarjeta, los dispositivos electrónicos que utilizaremos para la lectura de la tarjeta son los fototransistores, los cuales leerán las perforaciones hechas en las tarjetas.

EL FOTOTRANSISTOR

Son transistores con bases sensibles a la luz, es decir, existe una gran corriente de colector a emisor si se incide una luz en la base y cae a un valor muy bajo si se retira la luz. Este efecto se aprovecha al tomar como salida el colector, cuando no hay luz en la base, el voltaje de colector es máximo (Voltaje de la fuente) y cuando hay luz el voltaje en el colector es cero. Esto significa un 1 y 0 para dispositivos TTL..

Cada fototransistor generara un cero o un uno lógico dependiendo si la tarjeta posee una perforación o no. El tamaño del código binario generado será igual al numero de fototransistores que tengamos en el bloque de optocuplas; es decir, si tenemos 8 fototransistores registrando las perforaciones de la tarjeta, se generara un código binario de 8 bits.

INTERFACE PERIFERICA PROGRAMABLE (PPI) 8255.

La interfase es el dispositivo que conecta a la computadora con el bloque de optocopladores y la chapa eléctrica.

Se utilizara una PPI 8255 como interface, se escogió este chip debido a su compatibilidad con las líneas de control y datos de la computadora. Este dispositivo posee tres puertos bidireccionales, es decir que podemos leer y escribir datos en cada uno de ellos. La tabla 1 muestra un resumen de la operación de lectura y escritura de la PPI en los diferentes puertos.

Pines

RD	WR	A0	A1	Operación
0	1	0	0	Lectura en puerto A
1	0	0	0	Escritura en puerto A
0	1	0	1	Lectura en puerto B
1	0	0	1	Escritura en puerto B
0	1	1	0	Lectura en puerto C
1	0	1	0	Escritura en puerto C

Tabla 1.

Esta operación es hecha por el programa principal, mediante instrucciones de lectura y escritura a puertos.

El puerto A esta conectado directamente con la salida de los optocopladores, la computadora podrá leer estos datos gracias a una instrucción de lectura de puertos del programa principal.

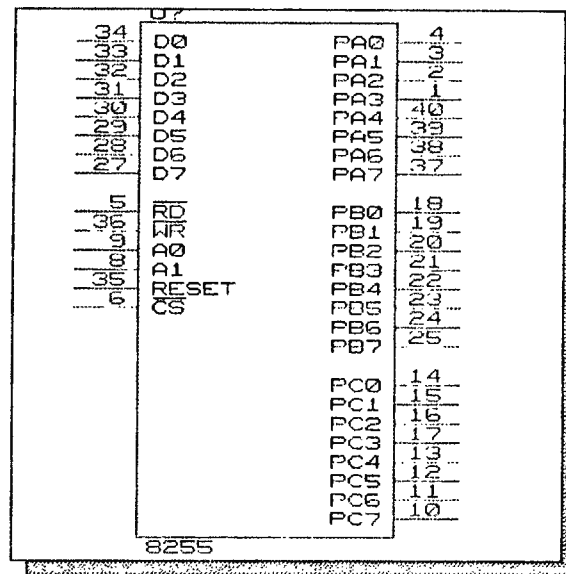
La señal de control de la chapa eléctrica y del sistema secundario de seguridad se generará en el puerto B, esta señal es generada por el programa principal gracias a una instrucción de escritura de puertos.

La interface periférica programable (PPI) 8255 es muy popular componente de bajo costo para interfaces, que se encuentra en muchas aplicaciones. La PPI tiene 24 terminales para E/S, programables por grupos de 12 terminales, que se utilizan en tres modos diferentes de funcionamiento.

La PPI 8255 puede tener interface con cualquier dispositivo de E/S compatible con TTL para el microprocesador. El 8255 (versión CMOS) requiere introducir estados de espera si trabaja con un microprocesador que tenga reloj mas de 8 MHz. Debido a que los dispositivos E/S son lentos por naturaleza, los estados de espera utilizados durante las transferencias de E/S no producen un efecto importante en la velocidad del sistema. El 8255 todavía tiene aplicación (compatible para la programación aunque no aparezca en un sistema como un 8255 discreto), incluso en los sistemas de computadora mas recientes basados en 80486. El 8255 se emplea para interface con el teclado y con el puerto paralelo de la impresora en estas computadoras personales.

DESCRIPCION BASICA DE LA 8255.

En la siguiente figura se ilustra el diagrama de base del 8255.



Sus tres puertos de E/S (Marcados A,B y C) se programan en grupos de 12 terminales. Las conexiones del grupo A constan del puerto A (PA7-PA0) y de la mitad superior del puerto C (PC7-PC4); el grupo B consiste en el puerto B (PB7-PB0) y la mitad inferior del puerto C (PC3-PC0). El 8255 se selecciona con su terminal CS para programarlo o para leer o escribir en un puerto. La selección de sus registros se logra por medio de las terminales A1 y A0, que seleccionan un registro interno para programación u operación. En la siguiente tabla se muestran las asignaciones de puertos de E/S usadas para programación y acceso a esos puertos. En la computadora personal, un 8255 o su equivalente se decodifican en

los puertos E/S 60h-63h.

A1	A0	FUNCION
0	0	PUERTO A
0	1	PUERTO B
1	0	PUERTO C
1	1	REGISTRO DE COM.

El 8255 es bastante sencillo de conectar (hacer interface) con el microprocesador y el programa. Para que se pueda leer o escribir en la 8255, la entrada CS debe ser un 0 lógico y la dirección correcta de E/S se debe aplicar en las terminales A1 y A0. Las terminales restantes de dirección de puerto son no importa y se decodifican en el exterior para seleccionar 8255.

PROGRAMACION DEL 8255.

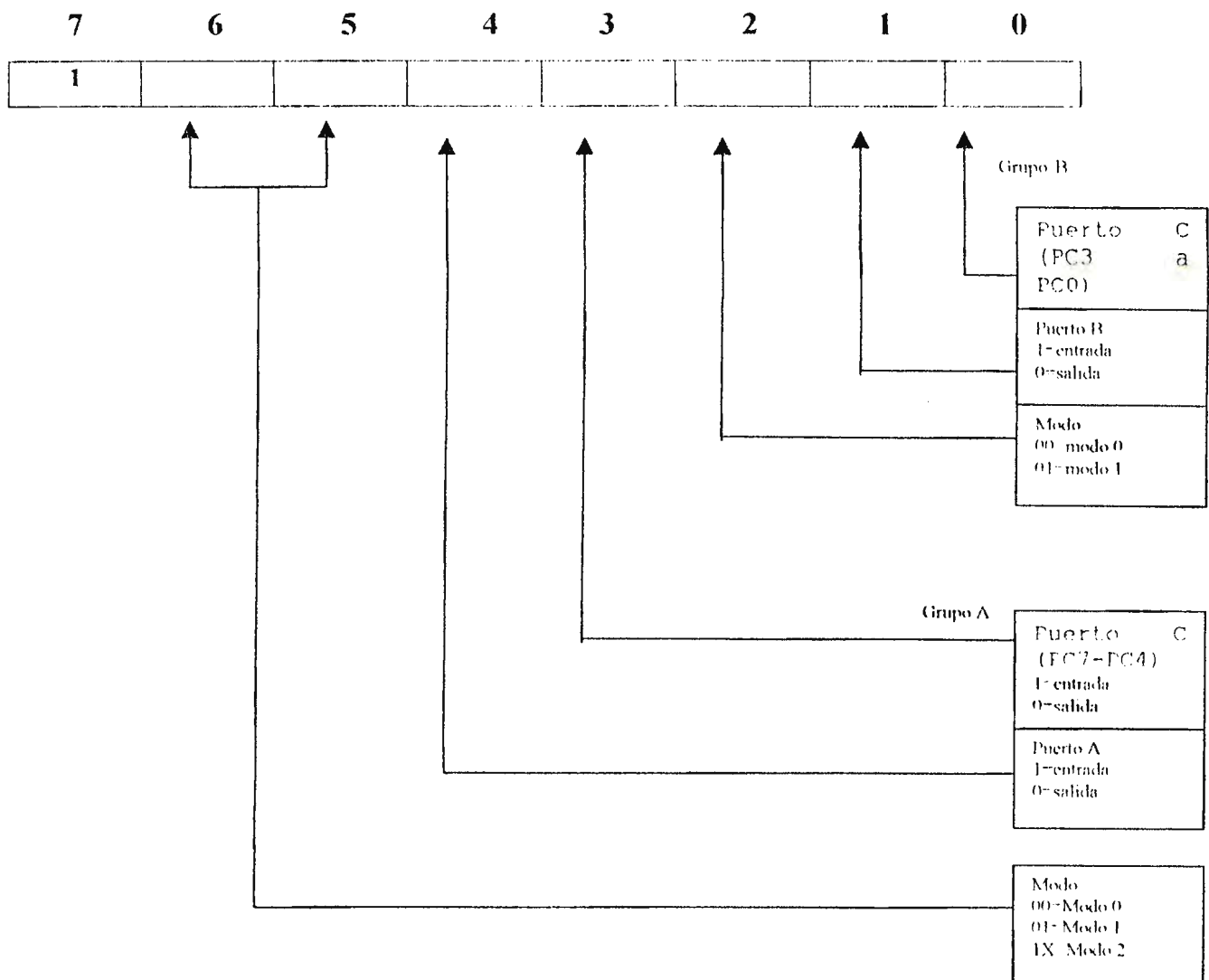
Es fácil programar el 8255 porque solo contiene dos posibles comandos básicos, como se ilustrará en la siguiente figura. Se verá que el bit de la posición 7 selecciona comandos A o al B. El comando A programa la función del grupo A y B, mientras que el comando B activa (1) bit o desactiva (0) bits del puerto C, solo si el 8255 se programa en el modo 1 o 2.

Las terminales del grupo B (puerto B y parte inferior del puerto C) se programan como terminales de entrada o de salida. El grupo B puede funcionar en el modo 0 o en el modo 1. El modo 0 es el modo básico de entrada y salida (E/S) que permite programar a las terminales del grupo B como conexiones simples de entrada o de salida con retención. El modo 1 es el funcionamiento con señales de habilitación estroboscópica en algunos bits del grupo B cuando se transfieren datos por el puerto B y C suministra señales de reconocimiento (handshake).

Las terminales del grupo A (puerto A y parte superior del puerto C) también se programan como terminales de entrada o de salida. La diferencia es que el grupo A puede funcionar en los modos 0,1 y 2. El funcionamiento en el modo 2 es un funcionamiento bidireccional

para el puerto A.

Si se pone un 0 en la posición 7 del byte de comando, se selecciona el comando B. Este comando permite que cualquier bit del puerto C se active (1) o se desactive (0) si se hace funcionar al 8255 en el modo 1 o el modo 2. De lo contrario, este byte de comando no se utiliza. A menudo se utiliza la función de activar bits en sistemas de control, para establecer o borrar un bit de control en el puerto C.



FUNCIONAMIENTO EN MODO 0.

El funcionamiento en modo 0 permite que la 8255 actúe como registro de entrada o como dispositivo de salida con registro transparente. Son lo mismo que los circuitos básicos de entrada y salida.

Modo 1: Entrada mediante habilitación.

El funcionamiento en modo 1 hace que el puerto A o el B funcionen

como registros de entrada. Esto permite que los datos externos se almacenen en el puerto hasta que el microprocesador esta listo para leerlos. El puerto C se utiliza también en el funcionamiento en Modo 1, no para datos, sino para señales de control de reconocimiento que hacen funcionar al puerto A o al puerto B como puertos de entrada mediante una señal de habilitación estroboscópica. El puerto de entrada mediante la señal de habilitación captura estroboscópica los datos de las terminales del puerto cuando se aplica la señal STB. Se debe tener en cuenta que esa señal captura los datos del puerto en la transición de 0 a 1. La señal STB hace que se capturen los datos en el puerto y también activa las señales IBF (entrada llena) e INTR (solicitar interrupción). Una vez que el microprocesador, por medio de un programa (IBF) o una interrupción (INTR) ha recibido aviso de que hay datos de entrada en el puerto, ejecuta una instrucción IN para leer el puerto (RD). La acción de lectura del puerto, lleva a IBF y a INTR a su estado inactivo hasta que hay un nuevo dato en el puerto.

Definición de señales para entrada en modo 1.

1. STB: Habilitación estroboscópica. Entrada utilizada para cargar datos en el registro del puerto, que retiene la información y se le da entrada al microprocesador por medio de la instrucción IN.
2. IBF: Registro de entrada, lleno. Una salida que indica que el registro de entrada contiene información.
3. INTR: Solicitud de interrupción. Es una salida que solicita una interrupción. La terminal INTR se activa con un 1 lógico cuando la entrada STB vuelve al 1 lógico y se desactiva cuando el microprocesador da entrada a los datos del puerto.
4. INTE: Habilitación interrupción ni entrada ni salida, sino un bit interno programado por medio de la posición de bit PC4 (puerto A) o PC2 (puerto B).
5. PC7,PC6: Terminales 7 y 6 de puerto. Son terminales de E/S de uso general que están disponibles para lo que se desee.

Modo 1: Salida mediante habilitación.

El funcionamiento con salida por habilitación estroboscópica es similar a la salida en Modo 0, excepto que se incluyen las señales de control para que haya un protocolo de reconocimiento.

Siempre que se escriben datos en un puerto programado como salida por habilitación estroboscópica, la señal OBF (registro de salida, lleno) se activa en 0 lógico para indicar que los datos están en el registro del puerto. Esta señal indica que los datos están disponibles para un dispositivo de E/S externo que, al tomar los datos activa la entrada ACK (reconocimiento) de habilitación al puerto. La señal ACK desactiva la señal OBF otra vez a un 1 lógico para indicar que el registro de salida está vacío.

Definiciones de señales de salida en modo 1.

1. OBF: Registro de salida, lleno. Una salida que va abajo siempre que hay salida a datos (OUT) por el puerto A o por el B. A esta señal se desactiva a 1 lógico siempre que el dispositivo externo proporciona el pulso ACK.
2. ACK: Reconocimiento. Una señal que hace que la terminal OBF se desactive con un 1 lógico. La señal ACK es la respuesta de un dispositivo externo con la que indica que recibió los datos desde el puerto 8255.
3. INTR: Solicitud de interrupción. Señal que interrumpe al microprocesador cuando el dispositivo externo recibe los datos por medio de la señal ACK. A esta terminal la califica el bit interno INTE (habilitación de interrupción).
4. INTE: Habilitación de interrupción. No es entrada ni salida, sino un bit interno programado para habilitar o deshabilitar a la terminal INTR. El bit INTEA se programa en PC6 y el INTEB en PC2.
5. PC5, PC4: Bits 5 y 4 del puerto C son terminales de E/S para uso general. Las instrucciones para activar y desactivar el bit se puede utilizar para estas dos terminales.

Modo 2 : Funcionamiento bidireccional.

En el modo 2, que solo se permite para el grupo A, el puerto A se vuelve bidireccional y permite transmitir y recibir datos en las mismas ocho terminales. Un canal de datos bidireccional es útil cuando se conectan (interface) dos computadoras. También se utilizan para la interface paralela estándar IEEE-488, de alta velocidad (canal de instrumentación de uso general GPIB).

Definiciones de señales para el modo 2.

1. INTR: Solicitud de interrupción. Salida utilizada para interrumpir al microprocesador para condiciones de entrada y de salida.
2. ORF: Registro de salida, lleno. Una salida que indica que el registro de salida contiene datos para el canal bidireccional.
3. ACK: Reconocimiento. Entrada que habilita los registros de tres estados de modo que los datos puedan aparecer en el puerto A. Si ACK es un 1 lógico, los registros de salida del puerto A están en su estado de alta impedancia.
4. STB: Habilitación estroboscópica. Entrada utilizada para cargar el registro de entrada del puerto A con datos externos que vienen del canal bidireccional del puerto A.
5. IBF: Registro de entrada, lleno. Salida utilizada para señalar que el registro de entrada contiene datos para el canal bidireccional externo.
6. INTE : Habilitación de interrupción. Bits internos (INTE1 e INTE2) que habilitan a la terminal INTR. El estado de la terminal INTR se controla con los bits del puerto C, PC6 (INTE1) y PC4 (INTE2).
7. PC2, PC1 y PC0 : Terminales de E/S de uso general en el modo 2 , controladas mediante las instrucciones y activación y desactivación de bit.

El canal bidireccional.

El canal bidireccional se utiliza con referencia al puerto A con

las instrucciones IN y OUT. Para transmitir datos por el canal bidireccional, el programa, prueba primero la señal OBF para determinar si el registro de salida esta lleno. Si lo esta se envían los datos al registro de salida con la instrucción OUT. Los circuitos externos también monitorean la señal OBF para decidir si el microprocesador ha enviado datos al canal. Tan pronto como los circuitos de salida ven un 0 en OBF envían la señal ACK para tomar los datos del registro de salida. La señal ACK desactiva al bit OBF y habilita a los registros de tres estados de la salida, a fin de poder leer los datos.

La terminal INTR (solicitud de interrupción) se puede activar desde ambos sentidos del flujo de datos por el canal. Si se habilita INTR con ambos bits INTE, entonces los registros de salida y de entrada producen solicitudes de interrupción. Esto ocurre cuando se hacen entrar los datos mediante una señal estroboscópica a los registros con el empleo de STB o cuando se escriben los datos con OUT.

ESTRUCTURA INTERNA DEL BUS ISA 8/16 bits

El bus clásico de un PC (ISA BUS) se compone de dos partes:

- La clásica de 8 bits, pertenece a los PC, XT y AT
- La extensión de 16 bits de los AT

Entre ambos forman el bus ISA que todos los ordenadores PC actuales poseen (no confundir con VESA o PCI, siendo el primero una tercera ampliación del bus ISA de 8 bits y el PCI un bus totalmente diferente).

Estructura de BUS de 8 bits PC, XT y AT:

Tierra	B1	A1	-I/O CH CHK (NMI)
+Reset DRV	B2	A2	+D7
+5V	B3	A3	+D6
+IRQ2	B4	A4	+D5
-5V	B5	A5	+D4
+DRQ2	B6	A6	+D3
-12V	B7	A7	+D2
-CARD SLCTD	B8	A8	+D1
+12V	B9	A9	+D0
Tierra	B10	A10	+I/O CH RDY
-MEMW	B11	A11	+AEN
-MEMR	B12	A12	+A19
-IOW	B13	A13	+A18
-IOR	B14	A14	+A17
-DACK3	B15	A15	+A16
+DRQ3	B16	A16	+A15
DACK1	B17	A17	+A14
+DRQ1	B18	A18	+A13
-DACK0 (MREF)	B19	A19	+A12
CLK	B20	A20	+A11
+IRQ7	B21	A21	+A10
+IRQ6	B22	A22	+A9
+IRQ5	B23	A23	+A8
+IRQ4	B24	A24	+A7
+IRQ3	B25	A25	+A6
-DACK2	B26	A26	+A5
+TC	B27	A27	+A4
+ALE	B28	A28	+A3
+5V	B29	A29	+A2
+OSC	B30	A30	+A1
Tierra	B31	A31	+A0

Extensión AT de 16 Bit:

-MEM CS16	D1	C1	SRHE
-I/O CS16	D2	C2	A23
IRQ10	D3	C3	A22
IRQ11	D4	C4	A21
IRQ12	D5	C5	A20
IRQ15	D6	C6	A19
IRQ14	D7	C7	A18
-DACK0	D8	C8	A17
DRQ0	D9	C9	-MEMR
-DACK5	D10	C10	-MEMW
DRQ5	D11	C11	D8
DACK6	D12	C12	D9
DRQ6	D13	C13	D10
-DACK7	D14	C14	D11
DRQ7	D15	C15	D12
+5V	D16	C16	D13
-Master	D17	C17	D14
Tierra	D18	C18	D15

La numeración empieza desde la parte posterior de la maquina.

SEÑAL	DESCRIPCION
A0 - A19	Bits de dirección "0-19," permiten direccionar 1Mb de memoria de e/s
A17 - A23	Bits de dirección 17-23, permiten direccionar desde 256Kb de memoria a 16Mb
AEN	Address Enable; cuando está activa el controlador DMA posee el control de las líneas de dirección y del BUS de datos, conforme se indique en MEMR/MEMW. Cuando está activa la CPU tiene el control de estas líneas.
ALE	Address Latch Enable (válida); se emplea para que la CPU esté aislada de líneas de dirección (triestado). Es forzado activado durante los ciclos DMA.
CARD SLCTD	Card Selected; indica que una tarjeta ha sido activada en el slot XT de 8.
CLK	Señal de reloj del sistema (actual velocidad del BUS)
D0 - D7	Bits de datos 0-7 para e/s a memoria o puertos de e/s.
DACK0-DACK3	Reconocimiento DMA para los canales 0 al 3; empleada por el controlador para reconocer una petición DMA (validación de acceso DMA). DACK0 es empleada para el refresco de memoria (MREF)
DRQ0-DRQ3	Petición DMA 0-3; empleada por periféricos que desean los servicios del controlador DMA; se mantiene activa hasta que la correspondiente señal DACKx se hace activa.
I/O CH CHK	I/O Channel Check; genera una interrupción no enmascarable.
I/O CH RDY	I/O Channel Ready; es puesta inactiva por memoria o dispositivos de e/s para retardar el acceso a memoria o los ciclos de e/s. Normalmente es empleada por dispositivos lentos para añadir estados de espera.
I/O CS16	I/O Chip select 16 Bit; indica ciclo de e/s de 16 bits.

IOR	I/O Chip Read; indica a un dispositivo de e/s que coloque su dato en el bus del sistema.
IOW	I/O Write; indica a un dispositivo de e/s a leer un dato del BUS del sistema.
IRQ2-IRQ7	Petición de interrupción 2-7; indica a la CPU que un dispositivo de e/s necesita servicio.
MASTER	Empleado por DRO para generar el control del sistema.
MEM CS16	Memory Chip Select 16 Bit; indica ciclo de memoria de 16 bits.
MEMR	Memory Read; esta señal es producida por la CPU o el controlador DMA e indica a la memoria que debe de introducir el dato direccionado en el BUS del sistema. Presente tanto en el BUS PC como en la extensión AT.
MEMW	Memory Write; esta señal es producida por la CPU o el controlador DMA e indica a la memoria que debe de leer y almacenar el dato presente en el BUS. Presente tanto en el BUS PC como en la extensión AT.
OSC	Oscilador, señal de reloj de 14.31818 MHZ (periodo de 70nS); 50% del ciclo de servicio.
RESET DRV	Reset Drive; empleada para resetear la lógica del sistema.
SBHE	System BUS High Enable; activa los bits de datos 8-15 de la extensión AT del BUS.
TC	Terminal Count; produce un impulso cuando la cuenta final de un canal DMA es alcanzado.

Todas las señales del BUS ISA emplean niveles TTL estándar.
La entrada y salida es con respecto a la CPU.
E/S significa entrada/salida.

PLANTEAMIENTO DEL PROYECTO

El proyecto es un sistema de seguridad activado por tarjeta perforada controlado por computadora. Este sistema abre o deja cerrada una chapa eléctrica cuando introducimos una llave eléctrica dentro de un dispositivo lector.

Para nosotros el dispositivo de seguridad es la chapa eléctrica, la cual será bloqueada o abierta, si esta tiene un código valido, la chapa eléctrica se abrirá, de lo contrario permanecerá cerrada.

Tanto la chapa como la lectura de la tarjeta es controlada por una computadora, ella controla todo el funcionamiento del sistema y hace que la tarjeta se lea y decida si ésta es correcta o no y así el sistema tomara la decisión de abrir o cerrar la chapa eléctrica.

La computadora también nos dejará especificar el número de usuarios que podrán abrir la chapa, ella preguntará el nombre y código a cada usuario que va a tener tarjetas válidas.

PRESENTACION

Las partes visibles de nuestro proyecto son 4: El bloque de optocuplas, la chapa eléctrica, los visualizadores y la computadora.

La chapa eléctrica y los visualizadores estarán ubicados en una puerta, la cual podrá abrirse cuando la chapa eléctrica contraiga su pasador. A un lado de ella estará el bloque de optocuplas, esta será una pequeña caja con una ranura del tamaño de una tarjeta de identificación.

La interface será una tarjeta que estará conectada en un Slot ISA dentro de la computadora, por lo que será una parte interna de ella, todos los cables de datos o control que van hacia el bloque de optocuplas y a la chapa eléctrica saldrán por la parte trasera de la computadora.

METODOLOGIA DE LA INVESTIGACION

Se desea diseñar un sistema de seguridad activado por tarjeta perforada confiable y fácil de programar, las razones por las que se selecciono este tipo de sistema fueron explicadas en la justificación.

El primer diseño contaba con un comparador de magnitud (7485) y una memoria RAM 16x4 (7489). La etapa de optocopladores tenía 4 sensores, debido a que en la entrada del comparador de magnitud sólo se necesitaban 4 bits. En la memoria RAM se almacenaban los códigos válidos y ésta se programaba gracias a interruptores minidip. La prueba resultó satisfactoria pero se observaron muchas desventajas: el tamaño del código de la tarjeta perforada era muy pequeño, la programación era muy difícil y los datos se borraban de la memoria cada vez que se apagaba el sistema.

La solución fue eliminar el comparador de magnitud y la memoria volátil y sustituir su operación con un programa, el cual comparará datos y los almacenara en una memoria permanente. Se penso utilizar un microprocesador y una memoria EPROM pero esto resultaba caro y la programación aun era complicada de realizar. Así que se opto por utilizar un medio que ya tenia microprocesador y memoria incorporado: la computadora, la cual solamente necesita una interface para ser conectada con los demás dispositivos del sistema.

Se recopiló información acerca de teoría de interfaces para computadora y programación de microprocesadores, en donde se llego a la conclusión que se utilizaría una PPI 8255 como interface debido a su compatibilidad con microprocesadores Intel.

Como el sistema podía ser violado al simular tarjetas perforadas en la ranura de inserción, se diseñara un subsistema de seguridad el cual activará una alarma la cual consiste en enviar un mensaje via puerto serie hacia otra computadora, dando los datos del hecho a personal de seguridad.

SITUACION ACTUAL

Actualmente se ha diseñado y armado la etapa de optoacopladores, la interface y el flujograma del programa principal. Además se tiene una parte del programa principal.

Se ha recopilado información acerca de los amplificadores que llevan la señal de los optocopladores a niveles TTL y la etapa de potencia que controlara la chapa eléctrica.

SOLUCION PROPUESTA

Se tiene un sistema de seguridad activado por tarjeta perforada fácil de programar por cualquier persona, con un novedosos subsistema que enviara un mensaje vía puerto serie hacia otra computadora indicando que existe la presencia de un intruso sin que este se dé cuenta.

El sistema tendrá la función de reloj marcador con la capacidad de registrar a 300 usuarios.

Se constara con 3 tipos de niveles de seguridad: Nivel Día/Noche: el cual tendrá horas restringidas prefijadas dentro del programa principal; Nivel de seguridad por horario: este será programable, se podrá especificar las horas de acceso por el día y por la noche; Acceso de supervisor: el cual tendrá una tarjeta maestra que le proporcionara un acceso al sistema sin restricciones.

LIMITANTES Y ALCANCES DEL PROYECTO

ALCANCES DEL PROYECTO

- El sistema llevara un registro de todas las personas que abran la chapa eléctrica.
- Se tendrá un control de la hora, fecha y las veces en que cualquier usuario active la chapa eléctrica. Esta información podrá ser accesada periódicamente o cuando se desee.
- Se tendrán dos tipos de acceso:
 - Acceso por Clave:** Además de introducir la tarjeta perforada, el usuario tendrá que digitar una clave de acceso para poder abrir la chapa eléctrica.
 - Acceso Directo:** No se necesitará una clave, la chapa eléctrica se abrirá con la introducción de la tarjeta perforada en el lector óptico.
- Se contara con un sistema de seguridad que protegerá la ranura de inserción de tarjetas perforadas, si algún intruso introduce objetos extraños en la ranura de inserción, se activara una alarma la cual consiste en enviar un mensaje vía puerto serie hacia otra computadora proporcionando información acerca de lo ocurrido y la ubicación exacta del establecimiento.
- La programación del sistema será fácil de realizar. Se tendrá una visualización de los datos a almacenar gracias al monitor de la computadora y una introducción de datos fácil y rápida mediante un teclado de la computadora. Se eliminarán los raros códigos o símbolos que se utilizan en sistemas que usan displays para su programación y en cambio podremos visualizar preguntas enteras y en nuestro idioma.
- El sistema tendrá la función de reloj marcador con la capacidad de registrar 300 usuarios.

- 3 tipos de niveles de seguridad:

Nivel Día/Noche: El cual tendrá horas restringidas de acceso prefijadas dentro del programa principal para usuarios diurnos y nocturnos.

Nivel por horario: El cual tendrá horas de acceso programables a cualquier hora del día.

Nivel Supervisor: Para personal de seguridad que desee acceder al sistema a cualquier hora.

LIMITACIONES DEL PROYECTO

- El máximo número de bit generado por la tarjeta perforada es de 8, debido a que el puerto de la interface conectado al bloque de optocuplas es de 8 bits; si queremos agregar un bit mas tenemos que utilizar la entrada de otro puerto o hacer un arreglo de compuertas lógicas.
- Gracias a que la ranura de inserción estará protegida por un subsistema de seguridad, el usuario deberá tener cuidado de no introducir cualquier objeto dentro de ella, ya que activará accidentalmente este sistema.
- Como la computadora no será de uso exclusivo para el sistema de seguridad, ya que estará en un disco duro donde se encuentran otro tipo de programas, el programa principal puede ser dañado o destruido a causa de un virus, causando así el mal funcionamiento del sistema.

RECOMENDACIONES Y CONCLUSIONES

RECOMENDACIONES

Este subsistema debe hacerse mediante la creación de un detector de tarjetas. El bloque de optocouplas solamente debe generar un código binario si hay una tarjeta dentro de la ranura del lector óptico. Un bit oculto en la tarjeta y una pequeña rutina de detección de ese bit en el programa principal puede solucionar el problema.

El material de la tarjeta perforada debe ser lo más oscuro posible a fin de que la luz de los emisores luminosos no traspase la tarjeta, debido a que si la luz traspasa un lugar de la tarjeta en donde se supone no tiene que haber una perforación, el receptor óptico captará esta luz como una perforación, creando así una lectura errónea y la generación de un mal código.

Debe protegerse al programa principal con una palabra clave antes de entrar a él, es decir, digitar un código de seguridad antes de entrar al programa con el fin de protegerlo de personas no autorizadas que quieran tener acceso a los códigos de los usuarios. Además de la palabra clave, todos los códigos de acceso deberán ser protegidos.

CONCLUSIONES

El aprovechamiento del microprocesador y la memoria de una computadora da al sistema a implementar mayor fiabilidad, precisión y flexibilidad debido a la exactitud y rapidez de las computadoras para procesar y almacenar información.

La experiencia recopilada en este trabajo, permite proponer soluciones viables para controles de sistemas eléctricos y mecánicos.

Con este trabajo se presenta documentación básica del proceso de diseño y se da apertura a la discusión sobre los beneficios de este proyecto desde el punto de vista prácticos e instructivos.

Resulta un proyecto interesante para reforzar el aprendizaje en asignaturas como Electrónica Digital, Interfaces y Periféricos, Electrónica Lineal Integrada y Computación.

GLOSARIO TECNICO.

Chapa eléctrica: dispositivo electromecánico el cual contrae o libera un pasador gracias a señales eléctricas, son parecidas a las chapas convencionales con la diferencia en su control eléctrico.

Código binario: son los conjuntos de bits base para nuestro sistema: Son generados por la etapa de optocuplas y también están dentro de la memoria de la computadora. Representan información de comparación.

Código Valido: serán los códigos binarios que abran o activen la chapa eléctrica.

Comparación de códigos: representa la comparación del código generado por la etapa optocuplas con los almacenados en la memoria de la computadora. Si los códigos son iguales el sistema procederá a activar(abrir) la chapa eléctrica, de lo contrario permanecerá cerrada. Esta función la realiza el programa principal.

Interface: es el dispositivo que conecta a la computadora con la chapa eléctrica y la etapa de optocuplas, gracias a ella el programa principal puede obtener información del bloque de optocuplas y controlar la chapa eléctrica.

Optocuplas: Representan nuestra etapa lectora de tarjetas perforadas, cada optocupla posee un emisor de luz y un receptor; dependiendo si se interrumpe la luz que proporciona el emisor al receptor, este generara un cero o un uno lógico.

Programación: Para nuestro sistema será la introducción de códigos binarios validos a la memoria de la computadora. Además, se preguntara el respectivo nombre del usuario para cada uno de ellos.

Tarjeta perforada: es una superficie plana de materia opaco y resistente, la cual será mas o menos del tamaño de una tarjeta de identificación. En ella habrán un numero de perforaciones u hoyos que pretenderán simular códigos binarios.

NO: Normalmente abierto.

NC: Normalmente cerrado.

BIBLIOGRAFIA.

- **LOS MICROPROCESADORES. 8086/8088/80186/80286/80386 Y 80486
INTEL. TERCERA EDICION. BARRY B. BREY. PRENTICE HALL.**
- **CURSO DE PROGRAMACION CON C. MICROSOFT C.
MACROBIT. FRANCISCO JAVIER CEBALLOS.**
- **ELECTRONICA INDUSTRIAL. MALONEY.**

ANEXOS

PROGRAMA DEL SISTEMA.

A continuación se muestra el programa que controla todo el sistema el cual esta en LENGUAJE DE PROGRAMACION C.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<time.h>
#include<dos.h>
#include<string.h>
int h=0,q,f=0,v,a;
struct tm *fh;
time_t segundos;
Retardo()
{
    delay(25);
}
borron()
{
    outp(0x302,0x3);
    outp(0x300,0x20);
    outp(0x302,0x2);
    outp(0x300,0x20);
    outp(0x302,0x1);
    outp(0x300,0x20);
    outp(0x302,0x0);
    outp(0x300,0x20);
}
BUSY()
{
    outp(0x302,0x3);
    outp(0x300,0x42);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x55);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x42);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x53);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x59);
    Retardo();
}
BUSY_()
{
    outp(0x302,0x3);
    outp(0x300,0x55);
    Retardo();
    outp(0x302,0x2);
```

```

    outp(0x300,0x53);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x55);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x59);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x20);
    Retardo();
}
SY__()
{
    outp(0x302,0x3);
    outp(0x300,0x53);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x59);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x53);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x20);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x20);
    Retardo();
}
BUSY1()
{
    int c,d,e;
    clrscr();
    outp(0x303,0x80);/*Palabra de comando*/

    outp(0x301,0x56);/*CLEAR*/
    Retardo();
    outp(0x301,q);
    Retardo();
    outp(0x302,0x00);
    outp(0x300,0x42);/* B */
    delay(300);

    outp(0x302,0x0);
    outp(0x300,0x20);

    outp(0x302,0x01);
    outp(0x300,0x42);/*B*/
    delay(300);

    outp(0x302,0x0);
    outp(0x300,0x55);/*U*/
    Retardo();

    outp(0x302,0x1);
    outp(0x300,0x20);

```

```

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x302,0x2);
outp(0x300,0x42);/*M*/

Retardo();
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x55);/*O*/

Retardo();
outp(0x302,0x0);
outp(0x300,0x53);/*M*/
Retardo();

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

BUSY();

outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x53);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
outp(0x302,0x0);
outp(0x300,0x42);
Retardo();

outp(0x301,q);
borron(); /*primer borron*/
USY_();

outp(0x301,v);/*segundo display*/
outp(0x302,0x0);
outp(0x300,0x20);

```

```

    outp(0x301,q);/*primero*/
    outp(0x302,0x3);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x59);

    outp(0x302,0x2);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x20);

    outp(0x302,0x0);
    outp(0x300,0x20);

    outp(0x301,v);
    outp(0x302,0x1);
    outp(0x300,0x42);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x55);
    Retardo();

    outp(0x301,q);
    horron();
    SY__();
}
ACCE()
{
    outp(0x302,0x3);
    outp(0x300,0x41);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x43);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x41);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x43);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x45);
    Retardo();
}

CCES()
{
    outp(0x302,0x3);
    outp(0x300,0x43);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x43);
    Retardo();
}

```

```

outp(0x302,0x3);
outp(0x300,0x43);
Retardo();
outp(0x302,0x1);
outp(0x300,0x45);
Retardo();
outp(0x302,0x0);
outp(0x300,0x53);
Retardo();
}

```

CESO()

```

{
outp(0x302,0x3);
outp(0x300,0x43);
Retardo();
outp(0x302,0x2);
outp(0x300,0x45);
Retardo();
outp(0x302,0x3);
outp(0x300,0x43);
Retardo();
outp(0x302,0x1);
outp(0x300,0x53);
Retardo();
outp(0x302,0x0);
outp(0x300,0x4F);
Retardo();
}

```

ESO_()

```

{
outp(0x302,0x3);
outp(0x300,0x45);
Retardo();
outp(0x302,0x2);
outp(0x300,0x53);
Retardo();
outp(0x302,0x3);
outp(0x300,0x45);
Retardo();
outp(0x302,0x1);
outp(0x300,0x4F);
Retardo();
outp(0x302,0x0);
outp(0x300,0x20);
Retardo();
}

```

SO_D()

```

{
outp(0x302,0x3);
outp(0x300,0x53);
Retardo();
outp(0x302,0x2);
outp(0x300,0x4F);
Retardo();
}

```

```

    outp(0x302, 0x3);
    outp(0x300, 0x53);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x44);
    Retardo();
}

O_DE()
{
    outp(0x302, 0x3);
    outp(0x300, 0x4F);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x4F);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x44);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x45);
    Retardo();
}

DEN()
{
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x44);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x4F);
    Retardo();
}

DENE()
{
    outp(0x302, 0x3);
    outp(0x300, 0x44);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x44);
}

```

```

Retardo();
outp(0x302, 0x1);
outp(0x300, 0x4E);
Retardo();
outp(0x302, 0x0);
outp(0x300, 0x45);
Retardo();
}
ENEG()
{
outp(0x302, 0x3);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x2);
outp(0x300, 0x4E);
Retardo();
outp(0x302, 0x3);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x1);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x0);
outp(0x300, 0x47); /* G+ */
Retardo();
}
NEGA()
{
outp(0x302, 0x3);
outp(0x300, 0x4E);
Retardo();
outp(0x302, 0x2);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x3);
outp(0x300, 0x4E);
Retardo();
outp(0x302, 0x1);
outp(0x300, 0x47);
Retardo();
outp(0x302, 0x0);
outp(0x300, 0x41);
Retardo();
}
EGAD()
{
outp(0x302, 0x3);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x2);
outp(0x300, 0x47);
Retardo();
outp(0x302, 0x3);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x1);
outp(0x300, 0x41);
}

```

```

Retardo();
outp(0x302,0x0);
outp(0x300,0x44);
Retardo();
}

```

```

GADO()
{
outp(0x302,0x3);
outp(0x300,0x47);
Retardo();
outp(0x302,0x2);
outp(0x300,0x41);
Retardo();
outp(0x302,0x3);
outp(0x300,0x47);
Retardo();
outp(0x302,0x1);
outp(0x300,0x44);
Retardo();
outp(0x302,0x0);
outp(0x300,0x4F);
Retardo();
}

```

```

ADO_()
{
outp(0x302,0x3);
outp(0x300,0x41);
Retardo();
outp(0x302,0x2);
outp(0x300,0x44);
Retardo();
outp(0x302,0x3);
outp(0x300,0x41);
Retardo();
outp(0x302,0x1);
outp(0x300,0x4F);
Retardo();
outp(0x302,0x0);
outp(0x300,0x20);
Retardo();
}

```

```

DO_()
{
outp(0x302,0x3);
outp(0x300,0x44);
Retardo();
outp(0x302,0x2);
outp(0x300,0x4F);
Retardo();
outp(0x302,0x3);
outp(0x300,0x44);
Retardo();
outp(0x302,0x1);
outp(0x300,0x20);
Retardo();
outp(0x302,0x0);
outp(0x300,0x20);
}

```



```

Retardo();
}
O2____()
{
    outp(0x302,0x3);
    outp(0x300,0x4F);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x20);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x4F);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x20);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x20);
    Retardo();
}
DENGE() {
    int cla,b,c,d,e,f,g;
    int L;
    clrscr();

    outp(0x303,0x80);/*Palabra de comando PPI#2 controlador de los displays
modo 0*/

    outp(0x301,0x40);/*CLEAR PB0*/
    Retardo();
    outp(0x301,q);/*habilitacion y clear*/
    Retardo();
    outp(0x302,0x00);/*El puerto C controla A0 y A1*/
    outp(0x300,0x41);
    delay(300);

    outp(0x302,0x0);/*direcciones A0,A1 del display 1*/
    outp(0x300,0x20);

    outp(0x302,0x01);
    outp(0x300,0x41);
    delay(300);

    outp(0x302,0x0);
    outp(0x300,0x43);
    Retardo();

    outp(0x302,0x1);
    outp(0x300,0x20);

    outp(0x302,0x0);
    outp(0x300,0x20);

    outp(0x302,0x2);
    outp(0x300,0x41);/*I*/

    Retardo();

```

```

outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x43);/*N*/

Retardo();
outp(0x302,0x0);
outp(0x300,0x43);/*S*/
Retardo();

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

ACCE();

outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x43);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
outp(0x302,0x0);

outp(0x300,0x41);
Retardo();

outp(0x301,q);
borron(); /*primer borron*/

CCES();
outp(0x301,v);/*segundo display*/
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x301,q);/*primero*/
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x45);

```

```

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
outp(0x302,0x1);
outp(0x300,0x41);
Retardo();
outp(0x302,0x0);
outp(0x300,0x43);
Retardo();

outp(0x301,q);
horron();

CESO();
outp(0x301,v);
outp(0x302,0x01);
outp(0x300,0x20);
outp(0x302,0x00);
outp(0x300,0x20);

outp(0x301,q);

outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x53);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);

outp(0x302,0x2);
outp(0x300,0x41);/*R*/

Retardo();
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x43);/*N*/

```

```

Retardo();
outp(0x302,0x0);
outp(0x300,0x43);/*5*/
Retardo();

outp(0x301,q);
barron();
FSO_();

outp(0x301,v);
outp(0x302,0x02);
outp(0x300,0x20);
outp(0x302,0x01);
outp(0x300,0x20);
outp(0x302,0x00);
outp(0x300,0x20);

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x4F);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x41);

Retardo();
outp(0x302,0x2);
outp(0x300,0x43);

Retardo();
outp(0x302,0x3);
outp(0x300,0x41);

Retardo();
outp(0x302,0x1);
outp(0x300,0x53);

Retardo();
outp(0x302,0x0);
outp(0x300,0x45);
Retardo();

outp(0x301,q);
barron();

SO_D();

outp(0x301,v);/* BL, habilitar dispaly2*/

```

```
outp(0x302, 0x3);
outp(0x300, 0x20);
```

```
outp(0x302, 0x1);
outp(0x300, 0x53);
```

```
outp(0x302, 0x2);
outp(0x300, 0x20);
```

```
outp(0x302, 0x1);
outp(0x300, 0x20);
```

```
outp(0x302, 0x0);
outp(0x300, 0x20);
```

```
outp(0x301, q);
outp(0x302, 0x3);
outp(0x300, 0x20);
```

```
outp(0x302, 0x1);
outp(0x300, 0x20);
```

```
outp(0x302, 0x2);
outp(0x300, 0x20);
```

```
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);
outp(0x301, v);
```

```
EEEG();
```

```
outp(0x301, q);
barron();
```

```
OLE();
outp(0x301, v);
outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x45);
outp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);
```

```
outp(0x301, q);
outp(0x302, 0x0);
outp(0x300, 0x20);
outp(0x302, 0x3);
outp(0x300, 0x44);
outp(0x302, 0x1);
outp(0x300, 0x20);
```

```

outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
CESO();

```

```

outp(0x301,q);
borron();
__DEN();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x53);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
FCO_();

```

```

outp(0x301,q);
borron();
BENE();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x4F);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x4E);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
SO_D();
outp(0x301,q);
horron();
FNEG();

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);/*aquilh*/

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x43);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
Q_DE();

outp(0x301,q);
horron();
NEGA();

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x44);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);

```

```

outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x47);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
_DEN();

```

```

outp(0x301,q);
borron();
EGAD();

```

```

outp(0x301,v);

```

```

outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x45);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
IDENE();

```

```

outp(0x301,q);
borron();
GADO();

```



```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x4E);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x44);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
ENEG();

```

```

outp(0x301,q);
borron();
ADD_();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x45);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x4F);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);

```

```
outp(0x302,0x0);
outp(0x300,0x20);
```

```
outp(0x301,v);
NEGA();
```

```
outp(0x301,q);
borron();
DO___();
```

```
outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x47);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x302,0x3);
```

```
outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
```

```
outp(0x301,v);
EGAD();
outp(0x301,q);
borron();
O2___();
```

```
outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
```

```
outp(0x301,q);
```

```

outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);

```

```

outp(0x301, v);
GADO();

```

```

outp(0x301, q);
horren();
_____();

```

```

outp(0x301, v);
outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x44);
outp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);

```

```

outp(0x301, q);
outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);

```

```

outp(0x301, v);
ABO_();

```

```

outp(0x301, q);
horren();
_____();

```

```

outp(0x301, v);
outp(0x302, 0x3);

```

```

outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x4F);
outp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);

```

```

outp(0x301, q);
outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);

```

```

outp(0x301, v);
D0__();

```

```

outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);

```

```

Q2____();

```

```

outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);
__ __();

```

```

outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x2);
outp(0x300, 0x20);

```

```

    outp(0x302, 0x1);
    outp(0x300, 0x20);
    outp(0x302, 0x0);
    outp(0x300, 0x20);
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    outp(0x302, 0x2);
    outp(0x300, 0x20);
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    outp(0x302, 0x0);
    outp(0x300, 0x20);

```

```

    return 0;

```

```

}

```

```

RETI()
{
    outp(0x302, 0x3);
    outp(0x300, 0x52); /*I*/
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x45); /*N*/
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x52); /*I*/
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x54); /*S*/
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x40); /*E*/
    Retardo();
}

```

```

ETIR()
{
    outp(0x302, 0x3);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x40);
    Retardo();
}

```

```

    outp(0x302,0x0);
    outp(0x300,0x52);
    Retardo();
}

TIRA()
{
    outp(0x302,0x3);
    outp(0x300,0x54);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x49);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x54);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x52);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x41);
    Retardo();
}

IRAR()
{
    outp(0x302,0x3);
    outp(0x300,0x49);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x52);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x49);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x41);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x52);
    Retardo();
}

IRAR_()
{
    outp(0x302,0x3);
    outp(0x300,0x52);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x41);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x52);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x52);
    Retardo();
}

```

```

    outp(0x302, 0x0);
    outp(0x300, 0x20);
    Retardo();
}

AP_ ( )
{
    outp(0x302, 0x3);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x52);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x41); /*Blanquillo*/
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x20);
    Retardo();
}

R_ T ( )
{
    outp(0x302, 0x3);
    outp(0x300, 0x52);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x52);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x54);
    Retardo();
}

TA ( )
{
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x41);
}

```

```

Retardo();
}

PETIPAR()
{
    int c1,a,b,c,d,e,f,g;
    int L;

    outp(0x303,0x80);/*Palabra de comando PPI#2 controlador de los displays
modo 0*/

    L=0;
    outp(0x301,0x40);/*CLEAR PB0*/
    Retardo();
    outp(0x301,q);/*habilitacion y clear*/
    Retardo();
    outp(0x302,0x00);/*El puerto C controla A0 y A1*/
    outp(0x300,0x52);/* I */
    delay(300);

    outp(0x302,0x0);/*direcciones A0,A1);
    outp(0x300,0x20);

    outp(0x302,0x01);
    outp(0x300,0x52);/*R*/
    delay(300);

    outp(0x302,0x0);
    outp(0x300,0x45);/*H*/
    Retardo();

    outp(0x302,0x1);
    outp(0x300,0x20);

    outp(0x302,0x0);
    outp(0x300,0x20);

    outp(0x302,0x2);
    outp(0x300,0x52);/*I*/

    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x45);/*N*/

    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x54);/*S*/
    Retardo();

    outp(0x302,0x2);
    outp(0x300,0x20);

```



```

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

PETI();

outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x54);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);
;
outp(0x301,v);
outp(0x302,0x0);

outp(0x300,0x52);
Retardo();
;
outp(0x301,q);
borron();          /*primer borron*/

ETIR();

outp(0x301,v);/*segundo display*/
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,q);/*primero*/
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x49);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);

```

```

outp(0x302,0x1);
outp(0x300,0x52);
Retardo();
outp(0x302,0x0);
outp(0x300,0x45);
Retardo();

outp(0x301,q);
horron();

TIRA();

outp(0x301,v);
outp(0x302,0x01);
outp(0x300,0x20);
outp(0x302,0x00);
outp(0x300,0x20);

outp(0x301,q);

outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x52);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);

outp(0x302,0x2);
outp(0x300,0x52);

Retardo();
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x45);

Retardo();
outp(0x302,0x0);
outp(0x300,0x54);
Retardo();

outp(0x301,q);
horron();
IRAR();

```

```

outp(0x301,v);
outp(0x302,0x02);
outp(0x300,0x20);
outp(0x302,0x01);
outp(0x300,0x20);
outp(0x302,0x00);
outp(0x300,0x20);

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x52); /*R*/

Retardo();
outp(0x302,0x2);
outp(0x300,0x45); /*E*/

Retardo();
outp(0x302,0x3);
outp(0x300,0x52);

Retardo();
outp(0x302,0x1);
outp(0x300,0x54);

Retardo();
outp(0x302,0x0);
outp(0x300,0x49); /*I*/
Retardo();

outp(0x301,q);
borren();

RSP();

outp(0x301,v); /* BL, habilitar display2*/
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x54);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x302,0x1);

```

```

outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x52);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);/*aquilh*/

:
outp(0x301,v);
ETIR();

outp(0x301,q);
borron();

AR__();

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x49);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,q);
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);

```

outp(0x300,0x20);

outp(0x301,v);

TIRA();

outp(0x301,q);

barren();

R__T();

outp(0x301,v);

outp(0x302,0x3);

outp(0x300,0x20);

outp(0x302,0x1);

outp(0x300,0x52);

outp(0x302,0x2);

outp(0x300,0x20);

outp(0x302,0x1);

outp(0x300,0x20);

outp(0x302,0x0);

outp(0x300,0x20);

outp(0x301,q);

outp(0x302,0x3);

outp(0x300,0x20);

outp(0x302,0x1);

outp(0x300,0x20);

outp(0x302,0x2);

outp(0x300,0x20);

outp(0x302,0x1);

outp(0x300,0x20);

outp(0x302,0x0);

outp(0x300,0x20);

outp(0x301,v);

IFAR();

outp(0x301,q);

barren();

TA();

outp(0x301,v);

outp(0x302,0x3);

outp(0x300,0x20);

outp(0x302,0x1);

outp(0x300,0x41);

outp(0x302,0x2);

outp(0x300,0x20);

outp(0x302,0x1);

outp(0x300,0x20);

outp(0x302,0x0);

outp(0x300,0x20);

outp(0x301,q);

outp(0x302,0x3);

outp(0x300,0x20);

outp(0x302,0x1);

outp(0x300,0x54);

```

outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
RAR_();

outp(0x301,q);
borron();
TAR();

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x52);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);/*aquilh*/

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
RAR_();

outp(0x301,q);
borron();
TAR();

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x52);
outp(0x302,0x2);
outp(0x300,0x70);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
R__T();

```

```

outp(0x301,q);
horron();
ARJE();

```

```

outp(0x301,v);

```

```

outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x70);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x70);
outp(0x302,0x1);
outp(0x300,0x4a);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
TA();
outp(0x301,q);
horron();
FJET();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x54);

```

```

outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x45);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
TAR();
outp(0x301,q);
horron();
JETA();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x14);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
API();
outp(0x301,q);
horron();
ETA();

```



```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x52);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x302,0x3);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
APTE();
outp(0x301,q);
berron();
"AA_{}";

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x53);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
RJFT();
outp(0x301,q);
herror();
Lambda__();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x45);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
RJFTA();
outp(0x301,q);
herror();
Lambda__();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x54);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);

```

```

outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
ETA_();
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

TA_();

```

```

outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
A_();

```

```

outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
_();

```

```

outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

/*goto comienzo;*/
return 0;
}

```

MYME()

```

{
    outp(0x302,0x3);
    outp(0x300,0x4d);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x4f);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x4d);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x4d);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x45);
    Retardo();
}

```

CMEM()

```

{
    outp(0x302,0x3);
    outp(0x300,0x4f);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x4d);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x4f);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x45);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x4d);
    Retardo();
}

```

MMMT()

```

{
    outp(0x302,0x3);
    outp(0x300,0x4d);
    Retardo();
    outp(0x302,0x2);
    outp(0x300,0x45);
    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x4d);
    Retardo();
    outp(0x302,0x1);
    outp(0x300,0x4d);
    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x54);
}

```

```

    Retardo();
}
EMTO()
{
    outp(0x302, 0x3);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x4d);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x4f);
    Retardo();
}

```

```

MTQ_()
{
    outp(0x302, 0x3);
    outp(0x300, 0x4d);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x4d);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x4f);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x2A);
    Retardo();
}

```

```

MTQ_2()
{
    outp(0x302, 0x3);
    outp(0x300, 0x4d);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x4d);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x4f);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x20);
    Retardo();
}

```

```

TO__()
{
    outp(0x302, 0x3);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x4f);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x2A);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x2A);
    Retardo();
}

O__()
{
    outp(0x302, 0x3);
    outp(0x300, 0x4f);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x2A);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x4f);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x2A);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x2A);
    Retardo();
}

PLANK()
{
    outp(0x302, 0x3);
    outp(0x300, 0x2A);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x2A);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x2A);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x2A);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x2A);
    Retardo();
}

```

```

M_MENTO2 () {

    outp(0x303,0x80); /*Palabra de comando*/
    /*comienzo:*/
    outp(0x301,0x40); /*CLEAR*/
    Retardo();
    outp(0x301,q);
    Retardo();
    outp(0x302,0x00);
    outp(0x300,0x4d); /* M */
    delay(300);

    outp(0x302,0x0);
    outp(0x300,0x20);

    outp(0x302,0x01);
    outp(0x300,0x4d); /*M*/
    delay(300);

    outp(0x302,0x0);
    outp(0x300,0x4f); /*O*/
    Retardo();

    outp(0x302,0x1);
    outp(0x300,0x20);

    outp(0x302,0x0);
    outp(0x300,0x20);

    outp(0x302,0x2);
    outp(0x300,0x4d); /*M*/

    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x4f); /*O*/

    Retardo();
    outp(0x302,0x0);
    outp(0x300,0x4d); /*M*/
    Retardo();

    outp(0x302,0x2);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x20);

    outp(0x302,0x0);
    outp(0x300,0x20);

    MOME();

    outp(0x302,0x3);

```

```

outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x4d);

outp(0x302, 0x2);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x20);

outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x301, v);
outp(0x302, 0x0);
outp(0x300, 0x41);
Retardo();

outp(0x301, q);
borron(); /*primer borron*/

MEM();

outp(0x301, v); /*segundo display*/
outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x301, q); /*primero*/
outp(0x302, 0x3);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x41);

outp(0x302, 0x2);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x20);

outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x301, v);
outp(0x302, 0x1);
outp(0x300, 0x41);
Retardo();
outp(0x302, 0x0);
outp(0x300, 0x41);
Retardo();

outp(0x301, q);
borron();

MEM();

```



```

outp(0x301,v);
outp(0x302,0x01);
outp(0x300,0x20);
outp(0x302,0x00);
outp(0x300,0x20);

outp(0x301,q);

outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x4d);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);

outp(0x302,0x2);
outp(0x300,0x4d);/*I*/

Retardo();
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x1f);/*N*/

Retardo();
outp(0x302,0x0);
outp(0x300,0x4d);/*S*/
Retardo();

outp(0x301,q);
Retardo();
/*MT0*/

outp(0x301,v);
outp(0x302,0x02);
outp(0x300,0x20);
outp(0x302,0x01);
outp(0x300,0x20);
outp(0x302,0x00);
outp(0x300,0x20);

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);

```

```

outp(0x302,0x1);
outp(0x300,0x54);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x4d); /*I*/

```

```

Retardo();
outp(0x302,0x2);
outp(0x300,0x4f); /*N*/

```

```

Retardo();
outp(0x302,0x3);
outp(0x300,0x4d); /*I*/

```

```

Retardo();
outp(0x302,0x1);
outp(0x300,0x4d); /*S*/

```

```

Retardo();
outp(0x302,0x0);
outp(0x300,0x45); /*E*/
Retardo();

```

```

outp(0x301,q);
Retardo();

```

```

HFO2();
}

```

/* MENTO() {

```

outp(0x301,0x50); /*Palabra de comando*/
/*comienzo:*/
outp(0x301,0xc0); /*CLEAR*/
Retardo();
outp(0x301,q);
Retardo();
outp(0x302,0x0);
outp(0x300,0x41); /* M */
delay(300);

```

```

outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x302,0x0);
outp(0x300,0x41); /*M*/
delay(300);

```

```

outp(0x302,0x0);

```

```

outp(0x300, 0x4f); /*O*/
Retardo();

outp(0x302, 0x1);
outp(0x300, 0x20);

outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x302, 0x2);
outp(0x300, 0x4d); /*M*/

Retardo();
outp(0x302, 0x3);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x4f); /*O*/

Retardo();
outp(0x302, 0x0);
outp(0x300, 0x4d); /*M*/
Retardo();

outp(0x302, 0x2);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x20);

outp(0x302, 0x0);
outp(0x300, 0x20);

i

MOME();

outp(0x302, 0x3);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x4d);

outp(0x302, 0x2);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x20);

outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x301, v);
outp(0x302, 0x0);
outp(0x300, 0x4d);
Retardo();

outp(0x301, q);

```

```

borron();          /*primer borron*/

MEM();

outp(0x301,v); /*segundo display*/
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,q); /*primero*/
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x45);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
outp(0x302,0x1);
outp(0x300,0x4d);
Retardo();
outp(0x302,0x0);
outp(0x300,0x4f);
Retardo();

outp(0x301,q);
borron();

MEM();

outp(0x301,v);
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,q);

outp(0x301,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x4d);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);

```

```

outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);

outp(0x302,0x2);
outp(0x300,0x4d);/*I*/

Retardo();
outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x4f);/*N*/

Retardo();
outp(0x302,0x0);
outp(0x300,0x4d);/*S*/
Retardo();

outp(0x301,q);
borren();
EMTO();

outp(0x301,v);
outp(0x302,0x02);
outp(0x300,0x20);
outp(0x302,0x01);
outp(0x300,0x20);
outp(0x302,0x00);
outp(0x300,0x20);

outp(0x301,q);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x54);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x4f);/*I*/

Retardo();
outp(0x302,0x1);
outp(0x300,0x4f);/*N*/

Retardo();
outp(0x302,0x3);
outp(0x300,0x4d);/*I*/

```

```

Retardo();
outp(0x302, 0x1);
outp(0x300, 0x40); /*S*/

```

```

Retardo();
outp(0x302, 0x0);
outp(0x300, 0x45); /*E*/
Retardo();

```

```

outp(0x301, q);
borron();

```

```

MTO_();

```

```

outp(0x301, v); /*Aqui*/
outp(0x302, 0x3);
outp(0x300, 0x20);

```

```

outp(0x302, 0x1);
outp(0x300, 0x40);

```

```

outp(0x302, 0x2);
outp(0x300, 0x20);

```

```

outp(0x302, 0x1);
outp(0x300, 0x20);

```

```

outp(0x302, 0x0);
outp(0x300, 0x20);

```

```

outp(0x301, q);
outp(0x302, 0x3);
outp(0x300, 0x20);

```

```

outp(0x302, 0x1);
outp(0x300, 0x40);

```

```

outp(0x302, 0x1);
outp(0x300, 0x20);

```

```

outp(0x302, 0x1);
outp(0x300, 0x20);

```

```

outp(0x302, 0x0);
outp(0x300, 0x20); /*aquih*/

```

```

outp(0x301, v);
borron();

```

```

outp(0x301, q);
borron();

```

TO__();

```
outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x45);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
```

```
outp(0x301,q);
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x2A);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
```

```
outp(0x301,v);
MEMT();
```

```
outp(0x301,q);
barron();
```

```
TO__();
outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x45);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
```

```
outp(0x301,q);
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x2A);
outp(0x302,0x2);
outp(0x300,0x20);
```

```

outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x301, v);
EMTO();

outp(0x301, q);
borron();
BLANK();

    utp(0x301, v);
outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x54);
outp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x301, q);
outp(0x302, 0x3);
outp(0x300, 0x20);
    utp(0x301, 0x1);
outp(0x300, 0x20);
    utp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
outp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x301, v);
MTQ();

outp(0x301, q);
borron();
BLANK();

    utp(0x301, v);
outp(0x302, 0x3);
outp(0x300, 0x20);
outp(0x302, 0x1);
    utp(0x300, 0x4f);
    utp(0x302, 0x2);
outp(0x300, 0x20);
outp(0x302, 0x1);
    utp(0x300, 0x20);
outp(0x302, 0x0);
outp(0x300, 0x20); /*aquilh*/

```



```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x30);
outp(0x302,0x1);
outp(0x300,0x2A);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
TO__();

```

```

outp(0x301,q);
horron();
BLANK();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x30);
outp(0x302,0x1);
outp(0x300,0x2A);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x2A);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
TO__();

```

```

outp(0x301,q);
horron();
BLANK();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x2A);

```

```

    outp(0x302, 0x2);
    outp(0x300, 0x20);
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    outp(0x302, 0x0);
    outp(0x300, 0x20);

    outp(0x301, q);
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    outp(0x302, 0x0);
    outp(0x300, 0x20);
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    outp(0x302, 0x0);
    outp(0x300, 0x20);

    outp(0x301, v);
    PLANK();

    outp(0x301, q);
    barren();
    PLANK();
}

HULL()
{
    outp(0x302, 0x3);
    outp(0x300, 0x40); /*I*/
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x40); /*H*/
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x40); /*I*/
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x60); /*S*/
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x40); /*E*/
    Retardo();
}

Newp()
{
    outp(0x302, 0x3);
    outp(0x300, 0x40);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x53);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x40);
}

```

```

Retardo();
outp(0x302, 0x1);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x0);
outp(0x300, 0x52);
Retardo();
}

CERT()
{
outp(0x302, 0x3);
outp(0x300, 0x53);
Retardo();
outp(0x302, 0x2);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x3);
outp(0x300, 0x53);
Retardo();
outp(0x302, 0x1);
outp(0x300, 0x50);
Retardo();
outp(0x302, 0x0);
outp(0x300, 0x54);
Retardo();
}

PETA()
{
outp(0x302, 0x3);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x2);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x3);
outp(0x300, 0x45);
Retardo();
outp(0x302, 0x1);
outp(0x300, 0x44);
Retardo();
outp(0x302, 0x0);
outp(0x300, 0x44);
Retardo();
}

BTSP()
{
outp(0x302, 0x4);
outp(0x300, 0x50);
Retardo();
outp(0x302, 0x2);
outp(0x300, 0x50);
Retardo();
outp(0x302, 0x3);
outp(0x300, 0x50);
}

```

```

    Retardo();
    outp(0x302, 0x11);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x52);
    Retardo();
}

TAB_1()
{
    outp(0x302, 0x3);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x54); /*Blanquillo*/
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x52);
    Retardo();
}

AB_1T()
{
    outp(0x302, 0x3);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x52);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x41);
    Retardo();
}

TAB_2()
{
    outp(0x302, 0x0);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x51);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x0);

```

```

    outp(0x300, 0x41);
    Retardo();
}

TAR()
{
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x52); /* R*/
    Retardo();
}

TARJ()
{
    outp(0x302, 0x3);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x50);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x4a);
    Retardo();
}

/* F */
F()
{
    outp(0x302, 0x3);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x50);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x4a);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x45);
}

```

```

    Retardo();
}
PJET()
{
    outp(0x302, 0x70);
    outp(0x300, 0x52);
    Retardo();
    outp(0x302, 0x20);
    outp(0x300, 0x40);
    Retardo();
    outp(0x302, 0x30);
    outp(0x300, 0x52);
    Retardo();
    outp(0x302, 0x10);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x00);
    outp(0x300, 0x54);
    Retardo();
}
PTTA()
{
    outp(0x302, 0x30);
    outp(0x300, 0x40);
    Retardo();
    outp(0x302, 0x30);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x30);
    outp(0x300, 0x40);
    Retardo();
    outp(0x302, 0x10);
    outp(0x300, 0x40);
    Retardo();
    outp(0x302, 0x10);
    outp(0x300, 0x41);
    Retardo();
}
PTTA_()
{
    outp(0x302, 0x30);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x20);
    outp(0x300, 0x40);
    Retardo();
    outp(0x302, 0x30);
    outp(0x300, 0x45);
    Retardo();
    outp(0x302, 0x10);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x00);
    outp(0x300, 0x20);
    Retardo();
}

```

```

TA_11()
{
    outp(0x302, 0x3);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x54);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x20);
    Retardo();
}

```

```

A_11()
{
    outp(0x302, 0x3);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x41);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x20);
    Retardo();
}

```

```

_11()
{
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x2);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x3);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x1);
    outp(0x300, 0x20);
    Retardo();
    outp(0x302, 0x0);
    outp(0x300, 0x20);
    Retardo();
}

```

```

ACTIVAR() {
    int a1,h,g,c,d;
    time_t hora;
    FILE *cfPtr;
    time(&hora);
    a1=ctime(&hora);
    gotoxy(25,15);
    cprintf("Fecha y Hora de entrada: s",a1);

    if((cfPtr=fopen("entrada.txt","a"))==NULL)
        printf("No way, no firulais");
    else
        fprintf(cfPtr,"d = s\n",a,a1);

        fclose(cfPtr);

    outp(0x301,0x20);
    delay(700);
    outp(0x301,0x00);
    gotoxy(25,12);
    clrscr();
    cprintf("PETIPAR TARJETA");

    outp(0x301,0x00);
    delay(500);
    again:
        RETIPAR();
        cinp(0x300);

        d=c&0x0;
        g=c&0x1;
        h=c&0x2;
        if((g==0x01 || h==0x00))
            goto again;

        goto again;
}

ACTIVAR2() {
    int a1,h,g,c,d;
    time_t hora;
    FILE *cfPtr;
    time(&hora);
    a1=ctime(&hora);
    gotoxy(25,15);
    cprintf("s/n",a1);

    if((cfPtr=fopen("salida.txt","a"))==NULL)
        printf("No way, no firulais");
    else
        fprintf(cfPtr,"d = s\n",a,a1);

        fclose(cfPtr);

    outp(0x301,0x00);
    delay(1000);

```



```

outp(0x301, 0x00);
clrscr();
gotoxy(25, 13);
cprintf("RETIRAR TARJETA");

```

```

outp(0x301, 0x00);
delay(2000);
again:
    RETIRAR();
    c=inp(0x306);
    gotoxy(30, 10);
    /*printf("X", c);*/
    d=c&0x2;
    g=c&0x1;
    if((g==0x01) || (d==0x08))
        {return(0);}

    goto again;
}

```

```

DENEGAR() {
    int c,d,g;
    clrscr();
    /*FILE *fP1r;*/
    printf("Acceso denegado\n");
    printf("Fuera de Horario\n");
    delay(2000);
    outp(0x301, 0x00);
    printf("POR FAVOR RETIRAR TARJETA");
    delay(1000);
    clrscr();
    again:
        RETIRAR();
        c=inp(0x306);
        gotoxy(30, 10);
        d=c&0x2;
        g=c&0x1;
        if((g==0x01) || (d==0x08))
            {
                return(0);
            }
        goto again;
}

```

```

DENEGAR() {
    int c, d, g;
    clrscr();
    printf("ACCESO DENEGADO\n");

    delay(2000);
    outp(0x301, 0x00);
    printf("POR FAVOR RETIRAR TARJETA\n");
}

```

```

        again:RETIRAR();
            c=inp(0x306);
            d=c&0x8;
            g=c&0x1;
            if((g==0x1) || (d==0x8))
            {
                return(0);
            }
            goto again;
    }

    MENSAJE(){
        union REGS regs;
        int i=0;
        char dato;
        char *mensaje={"EMERGENCIA \"INTRUSO\" . CIUDADELA DON BOSCO "};
        regs.x.ax=0xf3;
        regs.x.dx=0x01;
        int86(0x14,&regs,&regs);

        do{
            dato=mensaje[i];

            regs.h.ah=0x1;
            regs.h.al=dato;
            regs.x.dx=0x1;
            regs.h.ah=0x1;
            int86(0x14,&regs,&regs);

            i++;
        }
        while(i!=strlen(mensaje));
    }

    ALERTA(){
        char d;
        clrscr();
        MOMENTO2();
        alarma:
            outp(0x301,0x45); /*PB4=1, PB5=1, PB6=1 (BL DISPLAY)*/
            delay(2000);
            MENSAJE();
            d=getch();
            gotoxy(30,10);
            cprintf("Hay un intruso en la puerta");
            if(d!='1')
            {
                outp(0x301,0x05);
                delay(2000);
                goto alarma;
            }
            else {clrscr();

```

```

printf("esperar");
delay(3000);
clrscr();
outp(0x301, 0x45);
outp(0x301, 0x10);
delay(2000);
return(0);
}

```

```

}

```

```

int menu(void);
void crear();
void visualizar();
void actualizar();
void txtFile();
void AgregarArchivo();
void BorrarCodigo();
void Entrar();
void Salir();
void Arrancar();
void clave();
void Vitruclo();
void Amecula();
void Percent();

struct clientData{
    int codigo;
    char lastName[15];
    char firstName[10];
    int DirCta;
    int contader;
    int nivel;
};

struct ClaveData{
    int clave;
};

main()
{
    int opcion, password;

    while((opcion=menu())!=13){
        switch(opcion){
            case 1:
                crear();
                break;
            case 2:
                visualizar();
                break;

```

```

case 3:
    AgregarArchivo();
    break;
case 4:
    actualizar();
    break;
case 5:
    TextFile();
    break;

case 6:
    Perrarcodigo();
    break;
case 7:
    Entradas();
    break;
case 8:
    Salidas();
    break;
case 9:
    Arrancar();
    break;
case 10:
    claves();
    break;
case 11:
    Agreecla();
    break;
case 12:
    Vignola();
    break;
    }

return 0;
}

void crear()
{
    int i;
    struct clientData blankClient = {0, "", "", 0, 0, 0};
    FILE *cfptr;
    printf("Espera un momento por favor\n");
    if((cfptr = fopen("LECTOR.txt", "w")) == NULL)
        printf("File could not be opened.\n");
    else
        for(i = 1; i < 1000; i++)
            fwrite(&blankClient, sizeof(struct clientData), 1, cfptr);

    fclose(cfptr);
}

return 0;
}

```

```

void Regcont()
{
    FILE *cfPtr;
    int account;
    struct clientData client;
    a=inp(9*304);
    if((cfPtr=fopen("LECTOR.txt", "r"))!=NULL)
        printf("I'm Sorry\n");
    else{
        fseek(cfPtr, (a-1)*sizeof(struct clientData), SEEK_SET);
        fread(&client, sizeof(struct clientData), 1, cfPtr);
        if(client.codigo==0)
            printf("No hay informacion");

        else{
            client.contador=client.contador+1;
            printf("-%d",client.contador);
            delay(5000);
            fseek(cfPtr, (client.contador-1)*sizeof(struct clientData),
SEEK_SET);
            fwrite(&client, sizeof(struct clientData), 1, cfPtr);
        }
    }
}

void claves()
{
    int i;
    struct ClavesData blankClave= {0};
    FILE *cfPtr;
    printf("¿Que nombre quieres ponerle al archivo?\n");
    if((cfPtr=fopen("CLAVES.txt", "w"))!=NULL)
        printf("File could not be opened.\n");
    else{
        for(i=1; i<=10000; i++)
            fwrite(&blankClave, sizeof(struct ClavesData), 1, cfPtr);

        fclose(cfPtr);
    }
    return 0;
}

```

```

void Amortiza()
{
    FILE *cfPtr;
    struct ClavesData client;

    if((cfPtr=fopen("CLAVES.txt", "r"))!=NULL)
        printf("I'm Sorry\n");
    else{
        printf("Pulse 0 para finalizar\n\n");
    }
}

```

```

        printf("Introduzca clave\n? ");
        scanf(" %d", &clont.clave);

        while(clont.clave!= 0)
        {

            fseek(cfPtr, (clont.clave-1)*sizeof(struct ClavesData),
SEEK_SET);
            fwrite(&clont, sizeof(struct ClavesData), 1, cfPtr);
            printf("Introduzca clave:\n? ");
            scanf("%d", &clont.clave);
        }

        fclose(cfPtr);
    }
    return 0;
}

```

```

void Visualiza()
{
    int y;
    FILE *cfPtr;
    struct ClavesData clont;

    if((cfPtr=fopen("ClAVES.txt", "r"))==NULL)
        printf("No way\n");
    else{

        printf("\n-0s\n", "Claves Validas\n");

        while(!feof(cfPtr)) {
            fread(&clont, sizeof(struct ClavesData), 1, cfPtr);

            if(clont.clave!=0)
                printf("%-9d\n", clont.clave);

        }

        fclose(cfPtr);
        getch();
        return 0;
    }
}

```

```

void AgregarArchivo()
{
    FILE *cfPtr;
    struct clientData client;

    if((cfPtr=fopen("LECTOR.txt", "r"))==NULL)
        printf("I'm Sorry\n");
    else{
        b=0;
        printf("Pulse 0 para finalizar\n\n");
        printf("Introduzca codigo\n? ");
        scanf("%d", &client.codigo);

        while(client.codigo != 0)
        {
            printf("Introduzca Apellido, Nombre, Acceso Directo o por  
clave y el Tipo de Nivel\n");
            scanf("%s%s%d%d", &client.lastName, &client.firstName,
            &client.Pirula, &client.nivel);
            client.contador++;
            fseek(cfPtr, (client.codigo-1)*sizeof(struct clientData),
SEEK_SET);
            fwrite(&client, sizeof(struct clientData), 1, cfPtr);
            printf("Introduzca codigo\n? ");
            scanf("%d", &client.codigo);
        }

        fclose(cfPtr);
    }
    return 0;
}

void visualizar()
{
    int y;
    FILE *cfPtr;
    struct clientData client;

    if((cfPtr=fopen("LECTOR.txt", "r"))==NULL)
        printf("No way\n");
    else{
        printf(" %s%-11s%-11s%-9s %s\n", "Codigo", "Nombre", "Apellido", "Acceso", "Nivel", "Ingresos");

        while(!feof(cfPtr)) {
            fread(&client, sizeof(struct clientData), 1, cfPtr);

            if(client.codigo!=0)

```

```

        printf("%d-%d-%d-%d-%d-%d\n", client.codigo,
client.lastName, client.firstName, client.DirCla, client.nivel,
client.contaabr);

    }

    fclose(cfPtr);
    getch();
    return 0;
}

void actualizar()
{
    int i, pseudocodigo, cambio;
    struct clientData client;
    FILE *cfPtr;

    cfPtr=fopen("LECTOR.txt", "r");
    printf("Introduzca codigo del nuevo usuario: ");
    scanf("%d", &pseudocodigo);
    fseek(cfPtr, (pseudocodigo-1)*sizeof(struct clientData), SEEK_SET);
    fread(&client, sizeof(struct clientData), 1, cfPtr);

    if(client.codigo != 0)
        printf("El codigo %d posee informacion.\n", pseudocodigo);
    else
        printf("Introduzca Apellido, Nombre, Acceso Directo o por clave y
el Tipo de Nivel\n");
        scanf("%s %s %d", &client.lastName, &client.firstName,
&client.DirCla, &client.nivel);
        client.codigo=pseudocodigo;
        printf("%d-%d-%d-%d-%d-%d\n", client.codigo,
client.lastName, client.firstName, client.DirCla, client.nivel,
client.contaabr);
        fseek(cfPtr, (pseudocodigo-1)*sizeof(struct clientData), SEEK_SET);
        fwrite(&client, sizeof(struct clientData), 1, cfPtr);
        fclose(cfPtr);
    }

    return 0;
}

void TextFile()
{
    FILE *cfPtr, *wrtPtr;

    struct clientData client;

    cfPtr=fopen("LECTOR.txt", "r");

    if(!wrtPtr=fopen("Base.txt", "w"))
        printf("El archivo no puede ser abierto\n");

```



```

        else{
            rewind(cfPtr);
            fprintf(writePtr, "%-9s%-16s%-11s\n", "Codigo", "Apellido",
"Hombre");

            while(!feof(cfPtr)){
                fread(&client, sizeof(struct clientData), 1, cfPtr);

                if(client.codigo!=0)
                    fprintf(writePtr, "%-9s%-16s%-11s\n", client.codigo,
client.lastName, client.firstName);
            }
            fclose(writePtr);
            fclose(cfPtr);
            return 0;
        }

void Borrarcodigo()
{
    struct clientData client, blankClient={0, "", ""};
    int pseudocodigo;
    FILE *cfPtr;
    cfPtr=fopen("LECTOR.txt", "r");
    printf("Introduzca el codigo del usuario a eliminar\n");
    scanf("%d", &pseudocodigo);
    fseek(cfPtr, (pseudocodigo-1)*sizeof(struct clientData), SEEK_SET);
    fread(&client, sizeof(struct clientData), 1, cfPtr);

    if(client.codigo==0)
        printf("El usuario con codigo %d no existe\n", pseudocodigo);
    else{
        fseek(cfPtr, (pseudocodigo-1)*sizeof(struct clientData), SEEK_SET);
        fwrite(&blankClient, sizeof(struct clientData), 1, cfPtr);

        fclose(cfPtr);
    }
    return 0;
}

void Entradas()
{
    int codigo, i, j;
    char s[80];
    FILE *cfPtr;
    time(&t);
    clrscr();
    if((cfPtr=fopen("Entrada.txt", "r"))==NULL)
        printf("No se puede abrir");
    else{
        printf("%-10s\n", "\\t\\t\\t\\t");
        printf("CODIGO"
            "\\t\\t\\t\\t HORA Y FECHA DE ENTRADA\\n");
        fscanf(cfPtr, "%d", &codigo);
        fscanf(cfPtr, "%s", &s);
        while(!feof(cfPtr)){

```

```

        printf(" %-8d",codigo);

        for(j=0;j<4;j++){
            printf(" %-8s",s);
            fscanf(cfPtr, "%s", &s);
        }
        printf(" %-8s",s);
        printf("\n");
        fscanf(cfPtr, "%d", &codigo);
        fscanf(cfPtr,"%s", &s);

    }
    fclose(cfPtr);
}
getch();

return 0;
}

void Salidas()
{
    int codigo,t,j;
    char s[80];
    FILE *cfPtr;
    time(&t);
    clrscr();
    if((cfPtr=fopen("Salida.txt","r"))!=NULL)
        printf("NO SE PUEDE ABRIR");
    else{
        printf(" %-10s \n","\\(\\(\\n");
        printf("CODIGO"
            "\\(\\( HORA Y FECHA DE SALIDA\\n");
        fscanf(cfPtr, "%d", &codigo);
        fscanf(cfPtr,"%s", &s);
        while(!feof(cfPtr)){

            printf(" %-8d",codigo);

            for(j=0;j<4;j++){
                printf(" %-8s",s);
                fscanf(cfPtr, "%s", &s);
            }
            printf(" %-8s",s);
            printf("\n");
            fscanf(cfPtr, "%d", &codigo);
            fscanf(cfPtr,"%s", &s);

        }
        fclose(cfPtr);
    }
    getch();

    return 0;
}

```

```

void Arrancar()
{
    int cl,a,b,c,d,e,g,i,z[20];
    int L;
    FILE *cfPtr;
    struct clientData client;
    int y;
    struct ClavesData clont;
    clrscr();

    outp(0x303,0x80);/*Palabra de comando PPI#2 ontrolador de los displays
modo 0*/
    outp(0x307,0xbf);/*Palabra de comando PPI#1 lectores modo1*/
    q=0xd5;
    v=0xch;
    comienzo:
    clrscr();
    L=0;
    printf("INGRESAR TARJETA");
    outp(0x301,0x50);/*CLEAR PB0*/
    Retardo();
    outp(0x301,q);/*habilitacion y clear*/
    Retardo();
    outp(0x302,0x00);/*El puerto C controla A0 y A1*/
    outp(0x300,0x40);/* I */
    delay(300);

    outp(0x302,0x0);/*direcciones A0,A1);
    outp(0x300,0x20);

    outp(0x302,0x01);
    outp(0x300,0x40);/*1*/
    delay(300);

    outp(0x302,0x0);
    outp(0x300,0x40);/*N*/
    Retardo();

    outp(0x302,0x1);
    outp(0x300,0x20);

    outp(0x302,0x0);
    outp(0x300,0x20);

    outp(0x302,0x0);
    outp(0x300,0x40);/*1*/

    Retardo();
    outp(0x302,0x3);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x40);/*N*/

    Retardo();

```

```

outp(0x302,0x0);
outp(0x300,0x53);/*S*/
Potardo();

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

INSE();

outp(0x302,0x3);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x53);

outp(0x302,0x2);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x20);

outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
outp(0x302,0x0);
outp(0x300,0x40);
Potardo();

outp(0x301,q);
borron(); /*primer borron*/

NFER();

outp(0x301,v);/*segundo display*/
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,q);/*primero*/
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x302,0x1);
outp(0x300,0x40);

outp(0x301,0x0);
outp(0x300,0x20);

outp(0x302,0x1);

```

```

outp(0x300, 0x20);

outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x301, v);
outp(0x302, 0x1);
outp(0x300, 0x49);
Retardo();
outp(0x302, 0x0);
outp(0x300, 0x4e);
Retardo();

outp(0x301, q);
horron();

SERT();

outp(0x301, v);
outp(0x302, 0x01);
outp(0x300, 0x20);
outp(0x302, 0x00);
outp(0x300, 0x20);

outp(0x301, q);

outp(0x302, 0x3);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x52);

.

outp(0x302, 0x2);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x20);

outp(0x302, 0x0);
outp(0x300, 0x20);

outp(0x301, w);

outp(0x302, 0x2);
outp(0x300, 0x40); /*I*/

Potardo();
outp(0x302, 0x3);
outp(0x300, 0x20);

outp(0x302, 0x1);
outp(0x300, 0x40); /*N*/

Retardo();
outp(0x302, 0x0);
outp(0x300, 0x53); /*S*/

```

```

Retardo();

outp(0x301,q);
berron();
EkTA();

outp(0x301,v);
outp(0x302,0x02);
outp(0x300,0x20);
outp(0x302,0x01);
outp(0x300,0x20);
outp(0x302,0x00);
outp(0x300,0x20);

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x54);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x49); /*I*/

Retardo();
outp(0x302,0x2);
outp(0x300,0x10); /*H*/

Retardo();
outp(0x302,0x3);
outp(0x300,0x10); /*I*/

Retardo();
outp(0x302,0x1);
outp(0x300,0x10); /*S*/

Retardo();
outp(0x302,0x0);
outp(0x300,0x10); /*E*/
Retardo();

outp(0x301,q);
berron();

/*Fin*/

outp(0x302,0x0);
outp(0x302,0x00);
delay(10);

```

```

    outp(0x307,0x05);
    e--inp(0x306);
    delay(100);
    if (e&0x20;
    e<&0x02;
    if (e--0x2)
        {goto salida; }
    if (d--0x20)
        {goto entrada; }

    outp(0x301,v);/* BL, habilitar dispaly2*/
    outp(0x302,0x3);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x53);

    outp(0x302,0x2);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x20);

    outp(0x302,0x0);
    outp(0x300,0x20);

    outp(0x301,q);
    outp(0x302,0x3);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x41);

    outp(0x302,0x1);
    outp(0x300,0x20);

    outp(0x302,0x1);
    outp(0x300,0x20);

    outp(0x302,0x0);
    outp(0x300,0x20);/*aquilh*/

    outp(0x301,v);
    NOP();

    outp(0x301,q);
    NOP();

    NOP();

    outp(0x307,0x10);
    outp(0x307,0x09);
    delay(10);
    outp(0x307,0x05);

```

```

c=inp(0x306);
delay(100);
d=c&0x20;
e=c&0x02;
if(e==0x2)
    {goto salida;}
if(d==0x20)
    {goto entrada;}

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x45);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x52);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
SERF();

```

```

outp(0x301,q);
barron();

```

```

AR_T();
outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x52);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);

```



```

outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
ERTA();

```

```

outp(0x301,q);
borron();
RTA();

```

```

outp(0x307,0xbff);
outp(0x307,0x000);
delay(10);
outp(0x307,0x05);
c=inp(0x306);
delay(100);
d=c&0x20;
e=c&0x02;
if(e==0x2)
    {goto salida;}
if(d==0x20)
    {goto entrada;}

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x54);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x54);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);

```

```

BTAR();

outp(0x301,q);
berron();
_TAR();

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);/*aqui lh*/

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
_TAR();

outp(0x301,q);
berron();
_TAR();

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x50);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x52);
outp(0x302,0x2);
outp(0x300,0x20);

```

```

outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
AR_T();

```

```

outp(0x301,q);
borron();
ARJE();

```

```

outp(0x301,v);

```

```

outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x9);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x4a);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x0);

```

```

outp(0x301,v);
R_TA();
outp(0x301,q);
borron();
RJET();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x4a);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x45);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
_TAR();
outp(0x301,q);
borron();
DETA();

```

```

outp(0x307,0xbf);
outp(0x307,0x09);
delay(10);
outp(0x307,0x05);
c=inp(0x306);
delay(100);
d=c&0x20;
e=c&0x02;
if(e==0x2)
    {goto salida;}
if(d==0x20)
    {goto entrada;}

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x54);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
TARJ();
outp(0x301,q);
horron();
ETA_();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x52);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x302,0x3);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
ARJE();
outp(0x301,q);
horron();
TA__();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x4a);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);

```

```

outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
RJFT();
outp(0x301,q);
Lorron();
A____();

```

```

outp(0x307,0xbf);
outp(0x307,0x09);
delay(10);
outp(0x307,0x05);
c=inp(0x306);
delay(100);
d=c&0x20;
e=c&0x02;
if(e==0x2)
    {goto salida;}
if(d==0x20)
    {goto entrada;}

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x45);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x01);
outp(0x300,0x20);

```

```

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

```

```

outp(0x301,v);
ETA();
outp(0x301,q);
Lorron();
____();

```

```

outp(0x301,v);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x54);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,q);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x301,v);
ETA_();
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x41);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

TA_();

outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
A_();

outp(0x301,0x1);
outp(0x302,0x2);
delay(10);
outp(0x300,0x1);
outp(0x300);

```

```

delay(100);
d=c&0x20;
e=c&0x02;
if (e ==0x2)
    {goto salida;}
if (d==0x20)
    {goto entrada;}

outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);
outp(0x302,0x3);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x2);
outp(0x300,0x20);
outp(0x302,0x1);
outp(0x300,0x20);
outp(0x302,0x0);
outp(0x300,0x20);

outp(0x302,0xf);
outp(0x302,0x09);
delay(10);
outp(0x302,0xf);
outp(0x302,0x09);
delay(100);
d=c&0x20;
e=c&0x02;
if (e==0x2)
    {goto salida;}
if (d==0x20)
    {goto entrada;}

goto comienzo;

entrada:
q=0x0f;
v=0x0f;
BIOYI();
q=0x00;
v=0x00;
clrscr();
printf("ESPERAR UN MOMENTO\n");
a=inp(0x304);

```



```

MOMENTO();
clrscr();

if((cfPtr=fopen("LECTOR.txt","r"))==NULL)
printf("Error no se puede abrir el archivo LECTOR\n");
else{
    while(!feof(cfPtr)){
        fread(&client, sizeof(struct clientData), 1, cfPtr);
        if(client.codigo==a)
        {
            switch(client.nivel)
            {
                case 0:
                    time(&segundos);
                    clrscr();
                    fh=localtime(&segundos);
                    /*printf("%d horas\n", fh->tm_hour);*/
                    if((fh->tm_hour>=7)&&(fh->tm_hour<=16))/*Diurno*/
                    {goto ESTE;}
                    else{ DENEGRAD();
                    fclose(cfPtr);
                    Arrancar();
                    }
                case 1:time(&segundos);
                    clrscr();
                    fh=localtime(&segundos);
                    /*printf("%d horas\n", fh->tm_hour);*/
                    if((fh->tm_hour>=17)&&(fh->tm_hour<=23))/*Nocturno*/
                    {goto ESTE;}
                    else{ DENEGRAD();
                    fclose(cfPtr);
                    Arrancar();
                    }
                case 2:goto ESTE;
                    fclose(cfPtr);
                    Arrancar();
            }
            ESTE:clrscr();
            goto xy(35,12);
            printf("%s-11s -11s\n", client.firstName, client.lastName);
            delay(1000);

            if(client.Dirección != 0){ for(i=0; i<4; i++)
            {
                printf("Indice de la dirección: ");
                for(i=0; i<4; i++)
                {
                    order:
                    a=getch();
                    b=a-48;
                    if(b>=0&&b<10)
                    {
                        z[i]=b;
                        printf("%s");
                    }
                }
            }
        }
    }
}

```

```

else goto eder;
}
d=z[0]*1000+z[1]*100+z[2]*10+z[3];
a=inp(0x304);

if((cfPtr=fopen("CLAVES.txt", "r"))==NULL)
    printf("No way\n");
else
{
    while(!feof(cfPtr))
    {
        fread(&clont,
sizeof(struct ClavesData), 1, cfPtr);

        if(clont.claves==d)
        {
            printf("LA
CLAVE ES VALIDA\n");
            ACTIVAR();
            Arrancar();
        }
        if(f>=2){
            DENEGAD2();
            Arrancar();
        }
        else goto Coro;}

    }

    ACTIVAR();
    fclose(cfPtr);
    Arrancar();
}

fclose(cfPtr);
ALERTA();
main();
}

```

```

salida:
    q=0xDD;
    v=0xDB;
    BUSY1();
    q=0xd6;
    v=0xcc;
    clrscr();
    a=inp(0x305);
    gotoxy(35,12);
    cprintf("ESPERAR UN MOMENTO\n");
    MOMENTO();

    if((cfPtr=fopen("LECTOR.txt","r"))!=NULL)
        printf("Error no se puede abrir el archivo LECTOR\n");
    else{
        /*printf("%-9s%-11s%-11s\n", "Codigo", "Nombre", "Apellido");*/

        while(!feof(cfPtr)){
            fread(&client, sizeof(struct clientData),
1, cfPtr);

            if(client.codigo==a)
            {

                switch(client.nivel)
                {
                    case 0:
                        time(&segundos);
                        clrscr();

                        ft= localtime(&segundos);
                        /*printf("%d
                        hours\n", ft->tm_hour);*/

                        /*printf("%d
                        hours\n", ft->tm_hour-16);*/ /*como*/

                        {
                            goto ESTE;
                        }
                        else{ DENEGAR();

                                fclose(cfPtr);
                                Arrancar();

                                }
                    case 1:time(&segundos);
                        clrscr();

                        /*printf("%d
                        hours\n", ft->tm_hour);*/

```

```

time_hour>=17)&&(fh->tm_hour<=23))/*Nocturno*/
    if((fh-
        {goto ESTE;}
        else{ DENEGAD();

            fclose(cfPtr);
            Arrancar();

        }
        case 2:goto ESTE1;
            fclose(cfPtr);
            Arrancar();

    }
    ESTE1:clrscr();
    gotoxy(35,12);
    cprintf("%s-11s%-11s\n",
client.firstName, client.lastName);
    delay(1000);

    if(client.DirCla==0){
        Corol:f++;
        {
            printf("Ingresar
clave\n");

            for(i=0;i<4;i++)
            {
                eder1:
                a=getch();
                b=a-48;
                if(b>=0&&b<10)
                {
                    z[i]=b;
                    printf("*");
                }
                else goto eder1;
            }

            d=z[0]*1000+z[1]*100+z[2]*10+z[3];

            a=inp(0x305);

            if((cfPtr=fopen("CLAVES.txt", "r"))!=NULL)
                printf("No way\n");
            else
            {
                while(!feof(cfPtr))
                {
                    fread(&clont,
sizeof(struct ClavesData), 1, cfPtr);

                    if(clont.clave==d)
                        {

```

```

clave es valida");

fclose(cfPtr);

fclose(cfPtr);

printf("la
ACTIVAR2();

Arrancar();
}

}
if(f>=2){
DENEGAD2();

Arrancar();}
else goto Corol;

}

}

ACTIVAR2();
fclose(cfPtr);
Arrancar();

}

}

fclose(cfPtr);
ALEPTA();
main();
}

}

int menu(void)
{
int menu;
clrscr();
printf("\n INTRODUCZA SU Opcion\n"
"1 - Crear archivo\n"
"2 - Leer Archivo de codigos\n"
"3 - Agregar usuario(s)\n"
"4 - Nuevo usuario\n"
"5 - Crear archivo de texto\n"
"6 - Eliminar Usuario\n"
"7 - Ver Entradas\n"
"8 - Ver Salidas\n"
"9 - Inicializar sistema\n"
"10 - Crear archivo de claves\n"
"11 - Agregar Claves\n"

```

```
        "12 - Visualizar Claves de usuarios\n"  
        "13 - Exit\n");  
scanf(" d",&menu);  
return menu;  
}
```