UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Solución de cartas ASM por medio de técnicas de direccionamiento

Guía de prácticas

MATERIAL DIDÁCTICO

Que para obtener el título de

Ingeniero Mecatrónico

P R E S E N T A

Gustavo Mújica Pérez

ASESOR DE MATERIAL DIDÁCTICO

M.I. Ulises Martín Peñuelas Rivas

Ciudad Universitaria, CDMX, 2022

AGRADECIMIENTOS

Gracias a Dios, por todas las bendiciones, por permitirme tener y disfrutar a mi familia, tu amor y tu bondad no tienen fin.

A mi madre, porque has trabajado incasablemente por darnos lo mejor siempre, por todo tu amor y apoyo, por el ánimo para terminar este trabajo, eres mi motor y mi pilar en todo lo que hago, me has educado y enseñado todo, confío en Dios nos de vida y para poder regresarte lo que me has dado.

A mi hermana, por todo tu amor y apoyo en todo lo que hago, al igual que mi mamá eres un motor y pilar en mi vida.

A mi abuelita Conchita y a mi abuelito Neto, porque sin su apoyo en mi vida nada de esto hubiera sido posible. Los quiero infinitamente.

A mi Padre y a mi abuelita Fina, por todo el gran apoyo que me han dado.

A mi tía Marthy, a mi tía Rosy y a mi primo Neto, por todo el enorme apoyo. Les agradezco y los quiero muchísimo.

Al M.I. Ulises Martín Peñuelas Rivas, por su apoyo, orientación y consejos en la realización de este trabajo, al igual que en mi etapa académica y laboral. Muchas gracias.

A todos mis amigos, por todos los buenos momentos que pasamos juntos.

A la UNAM, por darme educación, aventuras y experiencias inolvidables, al igual que excelentes amigos. Muchas gracias.

CONTENIDO

CAPÍTULO 1 INTRODUCCIÓN

Este trabajo tiene como finalidad reforzar los conocimientos acerca de las técnicas para implementar una carta ASM (máquina de estado algorítmica por sus siglas en inglés), mediante la realización de prácticas basadas en tres maquetas didácticas: un estacionamiento, un sistema robotizado clasificador de paquetes y un elevador. Estás prácticas aportarán beneficios a los estudiantes de ingeniería que estén cursando asignaturas relacionadas con el tema, dándole un enfoque teórico-práctico. Estas prácticas serán de gran utilidad ya que no existe en internet o libros aplicaciones similares utilizando cartas ASM.

Se utilizarán tres técnicas para la implementación de cartas ASM: direccionamiento por trayectoria, direccionamiento entrada-estado y direccionamiento implícito, para entender la importancia de utilizar una u otra, así como sus ventajas y desventajas.

1.1 Objetivo general

Diseñar prácticas para la solución de problemas de diseño de circuitos secuenciales con múltiples entradas y salidas, utilizando máquinas de estado algorítmicas, así como diferentes técnicas para implementarlas.

1.2 Objetivos específicos

- Demostrar mediante maquetas didácticas la aplicación práctica de cartas ASM
- Reforzar los conocimientos acerca de las tres técnicas para implementar cartas ASM

1.3 Impacto en el proceso de enseñanza – aprendizaje

Estas prácticas tienen como finalidad proporcionar material didáctico que ayude a los profesores y alumnos de Circuitos Digitales, a reafirmar y mejorar el proceso enseñanza – aprendizaje. Pueden ser utilizadas en una clase a distancia o presencial, porque se podrá accesar fácilmente al contenido. Las prácticas pueden ser utilizadas para planes de estudios anteriores, actuales y futuros donde se enseñen cartas ASM.

1.4 Asignatura a la que se apoya

Circuitos Digitales:

- **Clave**: 1996
- **Plan de estudios**: Ingeniería Mecatrónica 2016
- **Licenciaturas para las que se imparte**: Ingeniería Mecatrónica e Ingeniería en Sistemas Biomédicos.

1.5 Material didáctico que se plantea elaborar

Prácticas basadas en maquetas didácticas, utilizando diferentes técnicas para implementar máquinas de estado algorítmicas.

1.6 Organización del material

Está trabajo está dividido en cuatro capítulos:

- **Capitulo uno**: se menciona la motivación para realizar este trabajo así como la importancia de proveer a los estudiantes de aplicaciones utilizando cartas ASM. Se plantean los objetivos generales que resumen la idea central y finalidad de este trabajo, así como los objetivos específicos que reflejan las metas concretas para alcanzar el objetivo general, se menciona la asignatura que se apoyará y el material didáctico que se plantea elaborar.

- **Capítulo dos**: se describen las características de las maquetas en las que se basarán las prácticas así como su funcionamiento. Cada maqueta se dividirá en subsistemas para facilitar la comprensión del diseño de las cartas ASM, posteriormente las cartas ASM de estos subsistemas se unirán, formando el sistema completo de la maqueta.

- **Capítulo tres**: se describen brevemente los conocimientos y métodos necesarios para poder realizar las prácticas y para poder hablar en un lenguaje común cuando se esté explicando el funcionamiento de estas. Además se habla brevemente acerca de la historia de las cartas ASM, así como las reglas para utilizarlas y las diferentes técnicas para implementarlas.

- **Capítulo cuatro**: se muestra el desarrollo de las prácticas donde se expone el diseño y los pasos para la realización de éstas. En primer lugar la maqueta del estacionamiento con once prácticas, en segundo lugar el sistema robotizado clasificador de paquetes con cinco prácticas y en último lugar el elevador con seis prácticas.

CAPÍTULO 2 DESCRIPCIÓN DE LAS MAQUETAS

A continuación, se describirán las maquetas didácticas que se utilizaron para elaborar las prácticas. Estas maquetas ayudaran a los interesados en conocer aplicaciones de las cartas ASM utilizando diferentes técnicas de direccionamiento con memorias. Estas maquetas son de gran utilidad porque no hay aplicaciones similares utilizando cartas ASM en internet o libros. Las maquetas se dividieron en subsistemas para facilitar su comprensión y el diseño de las prácticas, la última practica de cada maqueta une todos los subsistemas para formar el sistema completo.

2.1 Estacionamiento

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura 2.1).

Figura 2.1 Maqueta de estacionamiento con 10 lugares.

Sistema de entrada al estacionamiento

El sistema de entrada está compuesto por el sistema de emisión/retiro de boletos y el de la pluma para estacionamiento. En la entrada se debe adquirir un boleto para poder ingresar al estacionamiento. Se entrega un boleto de forma automática al presionar un botón, el boleto sale a través de una ranura. Una vez retirado el boleto, se levanta la pluma para permitir el paso de un vehículo y, cuando éste haya pasado completamente, se baja la pluma (ver figura 2.2).

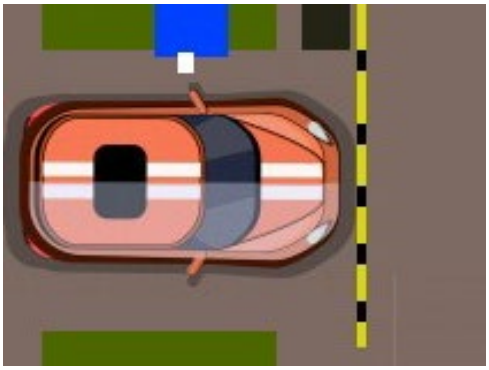


Figura 2.2 Entrada al estacionamiento.

Sistema de salida del estacionamiento

Este sistema está compuesto por el sistema de recolección de boletos y el de la pluma para estacionamiento. A la salida se debe recolectar el boleto para poder abandonar el estacionamiento. Se recolecta el boleto de forma automática al introducirlo en una ranura. Una vez ingresado el boleto completamente, se levanta la pluma para permitir el paso de un vehículo y, cuando éste haya pasado completamente, se baja la pluma (ver figura 2.3).

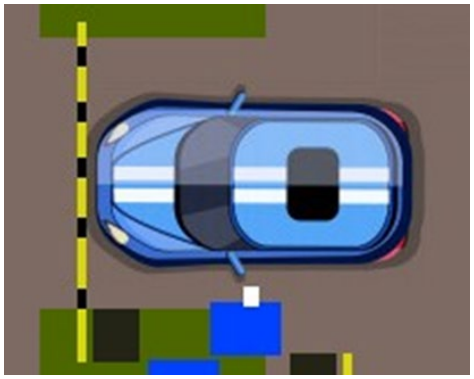


Figura 2.3 Salida del estacionamiento.

2.2 Sistema robotizado clasificador de paquetes

Para este sistema, también se simuló su funcionamiento con base en una maqueta. Este es un sistema robotizado que clasifica paquetes o cajas por colores y debe ser capaz de reconocer tres tipos de paquetes por su color colocados en una banda transportadora.

Los elementos principales del sistema son: un robot, un sensor de color y una banda transportadora. Inicialmente el sistema está en reposo, para activar el sistema se debe presionar un botón, al presionarlo la banda transportadora empezará a desplazar un paquete, un sensor detectará cuando el paquete esté en la posición adecuada para evaluar su color, por lo que la banda se detendrá.

El robot sujeta los paquetes y los deposita en el contenedor correspondiente. Los colores serán rojo, azul y verde. Los paquetes que no sean de estos colores, los dejará pasar (ver figura 2.4).

El robot se mueve a lo largo del eje horizontal por medio de un motor DC, para desplazarse en el eje vertical lo hace por medio de un motor paso a paso. El robot cuenta con un gripper, el cual es el encargado de sujetar los paquetes. El gripper está conformado por un motor DC.

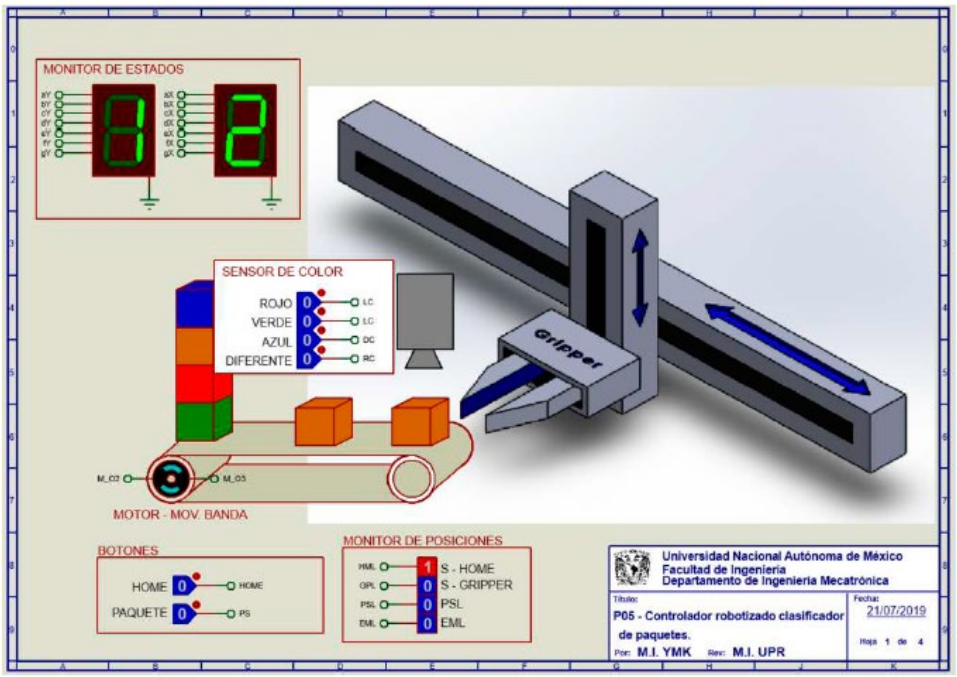


Figura 2.4 Sistema robotizado para clasificar paquetes por colores.

El sistema tiene un botón de emergencia, cuando éste sea oprimido el sistema quedará inmóvil, hasta que se oprima otro botón para dirigirse a su posición inicial. Se consideró la posición inicial del sistema cuando la articulación horizontal está en su extremo izquierdo, la articulación vertical en su posición superior y con el gripper abierto.

El sistema se dividió en 2 partes para facilitar su comprensión. El sistema de posición inicial controla cuando el gripper debe moverse en su posición inicial y el sistema clasificador, controla la banda y la clasificación de los paquetes por colores. La unión de los dos sistemas mencionados anteriormente forma el sistema robotizado clasificador de paquetes.

2.3 Elevador

Se simuló el funcionamiento de un elevador de un edificio de tres pisos por medio de una maqueta que está dividida en tres partes, las cuales se describen a continuación.

Cubo del elevador

El cubo tiene cuatro niveles, los cuales son: planta baja, piso 1, piso 2 y piso 3. Donde un carro sube y baja simulando el movimiento del elevador. Cada uno de los pisos tiene controles a un costado de la puerta para subir o bajar de piso, así como sensores de presencia que indican en donde está el carro (ver figura 2.5).

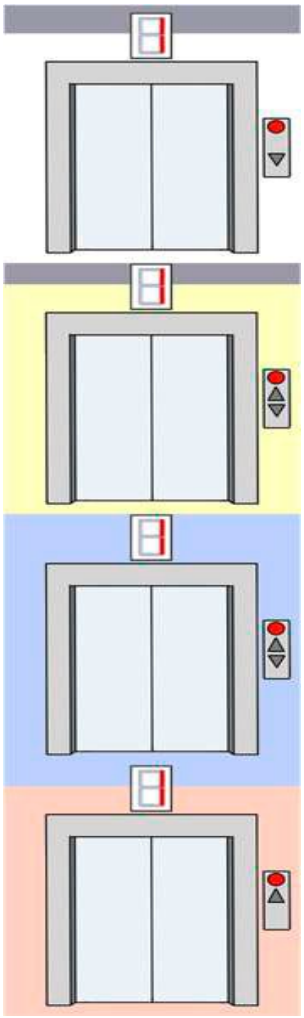


Figura 2.5 Elevador en torre de 4 niveles.

Puerta que abre y cierra

La maqueta cuenta con una caja estática que sirve de monitor de estados para visualizar el abrir y cerrar de la puerta. La puerta se abre por medio de un motor cuando el carro del elevador llegue al piso seleccionado. También se debe abrir cuando se presionen los botones de abrir o cerrar puerta mientras el carro este estático en un piso. Además, si se obstruye el paso de la puerta mientras la puerta se esté cerrando, se debe abrir inmediatamente (ver figura 2.6).

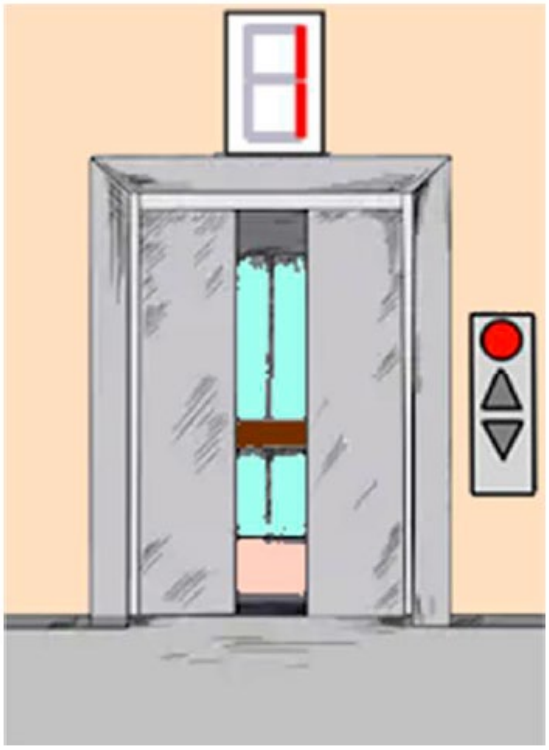


Figura 2.6 Puerta que abre y cierra.

Tablero de control

El tablero de control simula el control interno del elevador. Tiene botones para: dirigirse a los 4 niveles del elevador, detener elevador, abrir y cerrar la puerta, así como un indicador del piso en el que se encuentra el elevador (ver figura 2.7).

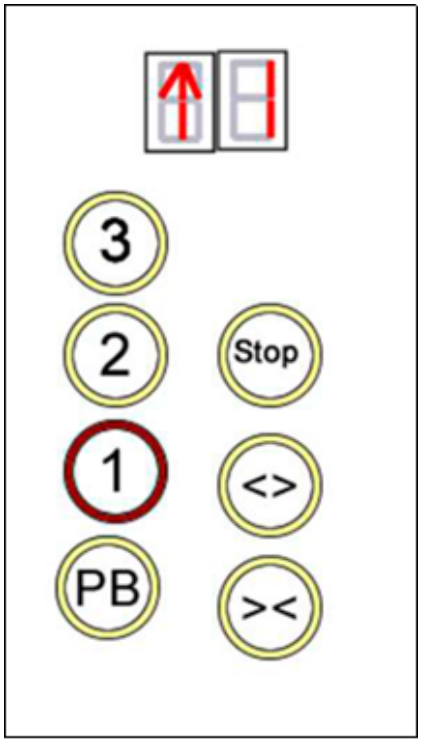


Figura 2.7 Tablero de control interno del cubo del elevador.

CAPÍTULO 3 MARCO TEÓRICO

Se describirán los conceptos necesarios para poder entender el diseño y la solución de las prácticas, para el diseño de estas se utilizó máquinas de estados algorítmicas, así como tres diferentes técnicas para implementarlas. Para la solución de las prácticas también se utilizaron conceptos de la asignatura de Circuito Digitales, estos se explican a continuación de manera breve, porque son necesarios para poder entender las soluciones propuestas.

3.1 Circuito combinacional

Los circuitos lógicos se pueden clasificar en dos grupos: combinacionales o secuenciales. Un circuito combinacional consiste en una conexión de compuertas lógicas, cuyas salidas en cualquier momento están determinadas por combinaciones de las variables de entrada. Los circuitos combinacionales realizan operaciones que se pueden representar mediante funciones booleanas (ver figura 3.1). Las funciones booleanas pueden ser determinadas mediante tablas de verdad y mapas de Karnaugh.

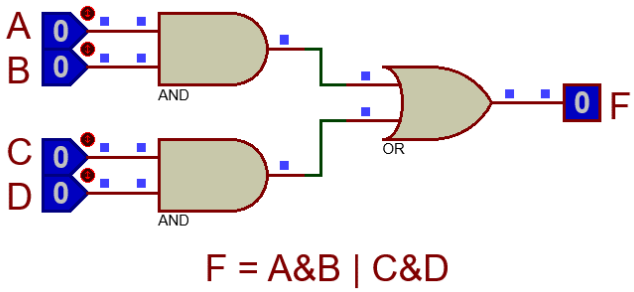


Figura 3.1 Circuito combinacional.

3.2 Decodificador

Un decodificador es un circuito combinacional que detecta un conjunto de bits en su entrada y activa sólo la salida que corresponde a este conjunto, todas las demás salidas permanecen inactivas. Como cada una de las N entradas puede ser '0' o '1', puede haber como máximo 2^N salidas (ver figura 3.2).

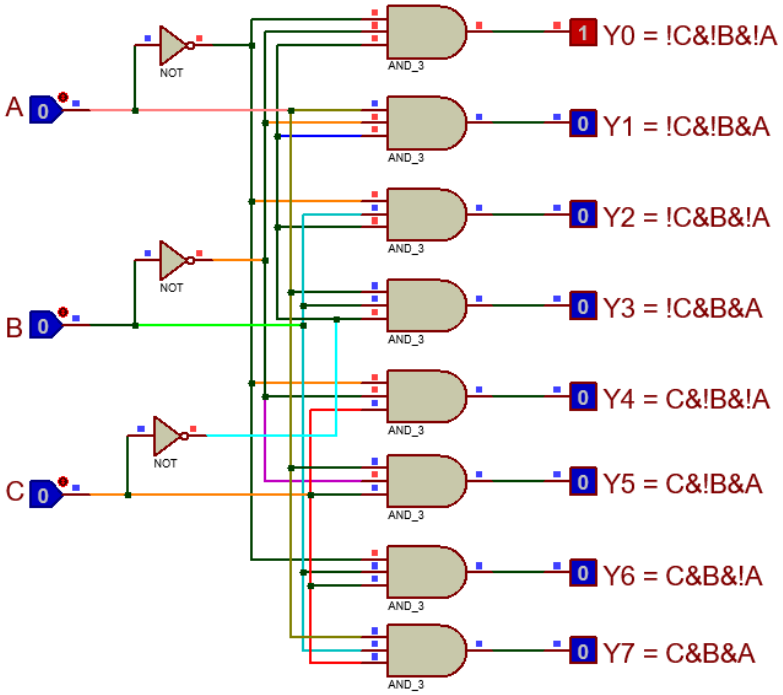


Figura 3.2 Decodificador de tres a ocho líneas. [1]

Se puede entender mejor el funcionamiento del decodificador con la tabla de verdad que se muestra en la tabla 3.1. Para cada posible combinación de entrada, existen siete salidas iguales a '0' y sólo una igual '1' [1].

Tabla 3.1 Tabla de verdad de un decodificador de tres a ocho líneas.

ENTRADAS			SALIDAS							
C	**B**	**A**	**Y0**	**Y1**	**Y2**	**Y3**	**Y4**	**Y5**	**Y6**	**Y7**
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

3.3 Multiplexor

Un multiplexor (MUX) es un circuito combinacional que permite dirigir la información binaria procedente de varias líneas de entrada a una única línea de salida. Tiene líneas de selección de datos, que permiten controlar los bits provenientes de las entradas hacia la línea de salida (ver figura 3.3). A los multiplexores también se les conoce como selectores de datos.

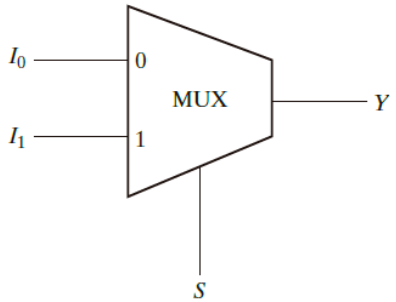


Figura 3.3 Diagrama de bloques de un multiplexor de dos líneas a una [1].

Usualmente hay 2^N líneas de entrada y N líneas para seleccionar datos. En la figura 3.4 se muestra un multiplexor de cuatro líneas a una. Dado que tiene 2 líneas para seleccionar datos, puede dirigir la información binaria de una de las cuatro líneas de entrada a una línea de salida (ver figura 3.5).

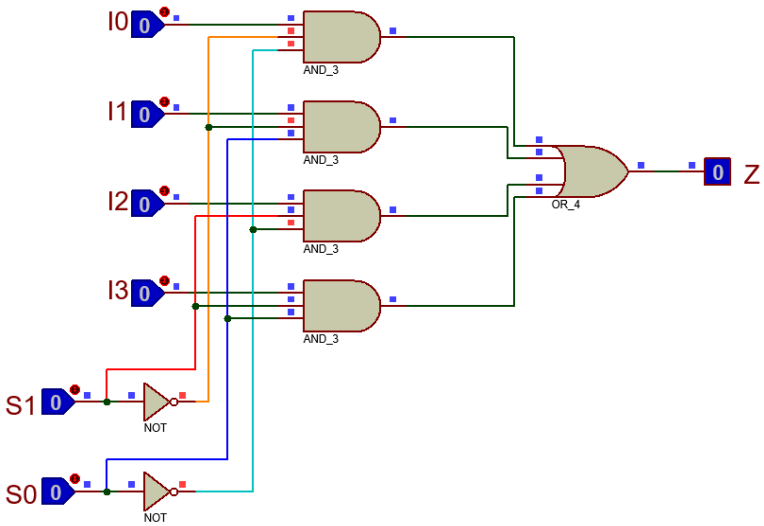


Figura 3.4 Multiplexor de cuatro líneas a una [1].

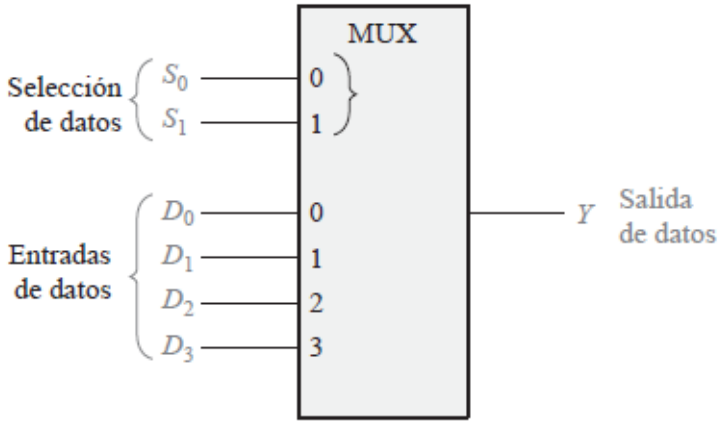


Figura 3.5 Símbolo lógico para un multiplexor de cuatro líneas a una [2].

La tabla de funcionamiento de un multiplexor (tabla 3.2) indica la relación entre las entradas y salidas dependiendo de las líneas de selección de datos para un multiplexor de cuatro líneas a una [1].

Tabla 3.2 Tabla de funcionamiento para un multiplexor de cuatro líneas a una.

S1	S0	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Los multiplexores se pueden combinar, por ejemplo, en la figura 3.6 se muestra un circuito que tiene cuatro multiplexores de un bit, cada uno puede seleccionar una de dos líneas de entrada y dirigir la información binaria seleccionada a una línea de salida. Dicho de otra manera, se puede entender como un multiplexor que selecciona información entre dos conjuntos de 4 bits (ver tabla 3.3).

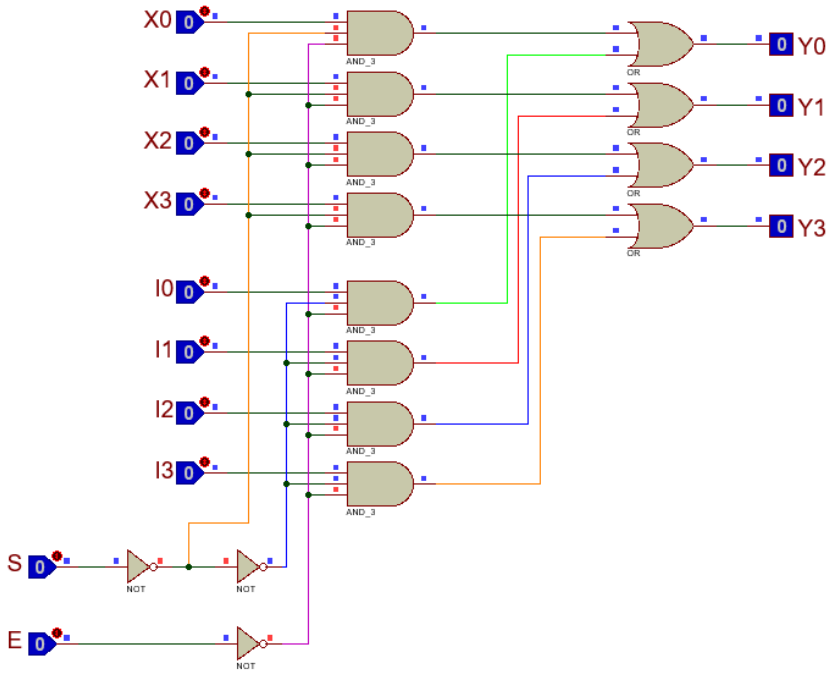


Figura 3.6 Multiplexor cuádruple de dos líneas a una [1].

Tabla 3.3 Tabla de funcionamiento para un multiplexor cuádruple de dos líneas a una.

E (HABILITAR)	S (SELECCIONAR)	Y
1	*	0
0	0	SELECCIONA "X"
0	1	SELECCIONA "I"

3.4 Demultiplexor

Un demultiplexor (DEMUX) o distribuidor de datos realiza la operación contraria a un multiplexor, es decir, recibe el dato de una línea de entrada y lo dirige a una de entre varias líneas de salida. El código del selector de datos determina cuál salida transmitirá la información (ver figura 3.7).

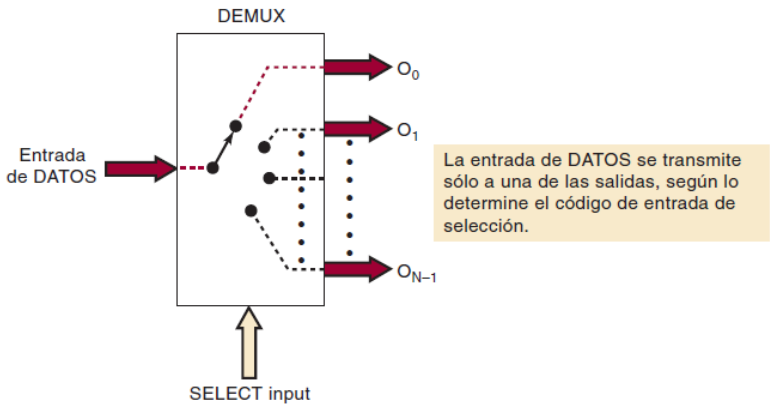


Figura 3.7 Demultiplexor general [3].

En la figura 3.8 se muestra un ejemplo de demultiplexor, donde hay 1 línea de entrada de datos, 2 líneas de selección de datos y 4 líneas de salida que dependen de la selección de datos.

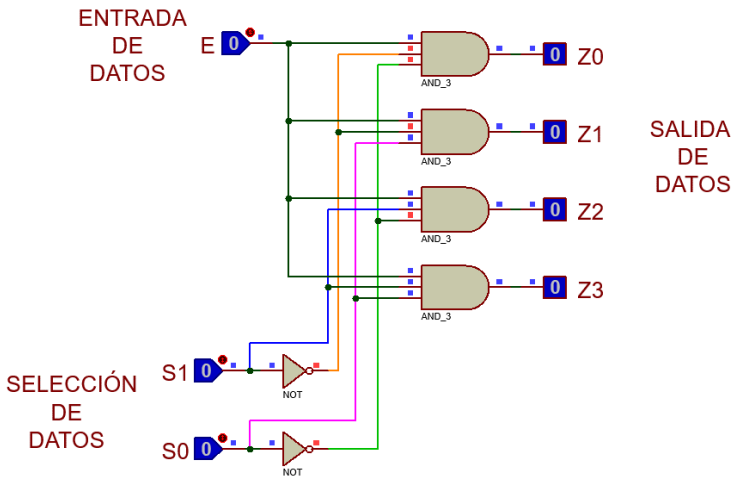


Figura 3.8 Demultiplexor de 1 a 4 líneas [1].

En la tabla 3.4 se muestra el funcionamiento para un multiplexor de 1 a 4 líneas, donde E es la entrada de datos.

Tabla 3.4 Tabla de funcionamiento para un demultiplexor de 1 a 4 líneas.

SELECCIÓN		**SALIDAS**			
S1	**S0**	**Z0**	**Z1**	**Z2**	**Z3**
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

3.5 Circuito secuencial

A diferencia de los circuitos combinacionales cuyas salidas sólo dependen de las entradas presentes, los circuitos secuenciales dependen de las entradas presentes, las salidas pasadas y de una señal cuadrada de periodo constante comúnmente conocida como señal de reloj, para poder almacenar las salidas pasadas estos circuitos cuentan con memoria. Es decir, un circuito secuencial está compuesto por un circuito combinacional al que se conectan elementos de memoria para formar una trayectoria de retroalimentación [1] (ver figura 3.9).

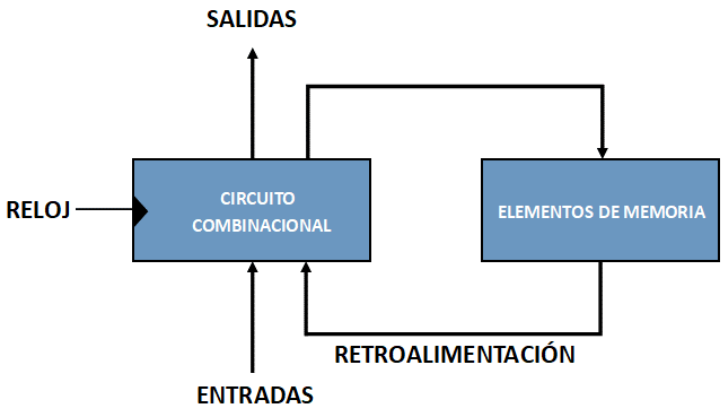


Figura 3.9 Diagrama de bloques de un circuito secuencial.

Los circuitos secuenciales se clasifican en dos grupos síncronos y asíncronos. Un circuito secuencial síncrono depende de pulsos de reloj sincronizados, sus elementos de memoria son flip-flops que operan con un reloj. Un circuito secuencial asíncrono depende de cambios en sus variables de entrada, sus elementos de memoria son flip-flops sin reloj o elementos de retardo.

3.6 Flip – flop

Los elementos de memoria más utilizados son los flip-flops, en adelante FF, están formados por varias compuertas lógicas interconectadas de cierta forma que pueden almacenar un bit de información. Los cambios en la salida de un flip-flop dependen de una transición de reloj (CLK) y puede asumir uno de dos estados estables '0' o '1'. El nombre técnico en español para un flip-flop es multivibrador biestable.

Flip-Flop tipo D

Este flip-flop es el más económico y eficiente, debido a que requiere un número mínimo de compuertas para su construcción. Funciona de la siguiente manera: Q cambiará al mismo estado que esté presente en la entrada D cuando ocurra una transición en el reloj (ver figura 3.10 y tabla 3.5).

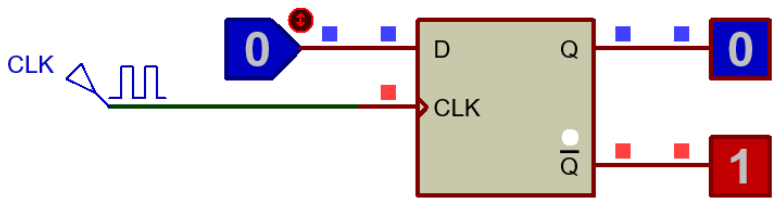


Figura 3.10 Flip-flop D.

Tabla 3.5 Tabla de verdad para un Flip-flop tipo D [1].

ENTRADAS		SALIDAS	
D	**CLK**	**Q**	**$\overline{Q}$**
1	↑	1	0
0	↑	0	1
↑ = Transición de reloj de bajo a alto			

Flip- flop JK

Dos tipos de flip-flop menos utilizados son los tipos JK y T. Un flip-flop JK dependiendo de sus dos entradas puede dar como salida ‘1’ o ‘0’, mantener o complementar su salida (ver figura 3.11 y tabla 3.6).

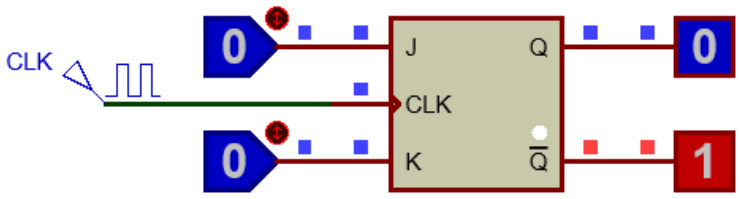


Figura 3.11 Flip-flop JK.

Tabla 3.6 Tabla de verdad para un Flip-flop JK [3].

ENTRADAS			SALIDAS
J	K	CLK	Q
0	0	↑	Q (Ningún cambio)
1	0	↑	1
0	1	↑	0
1	1	↑	!Q (Complemento)
↑ = Transición de reloj de BAJO a ALTO			

3.7 Registro

Un registro es un conjunto de flip-flops que son usados para almacenar datos y que son sincronizados por una señal de reloj común. Un ejemplo es el registro de cuatro bits, este consiste en cuatro flip-flops tipo D (ver figura 3.12). Para este registro, la señal de reloj común activa a todos los flip-flops luego de la transición de cada pulso y los datos disponibles en las entradas se transfieren a las salidas del registro. Mientras no haya una transición del pulso de reloj, los datos a la entrada del registro no afectarán las salidas de este.

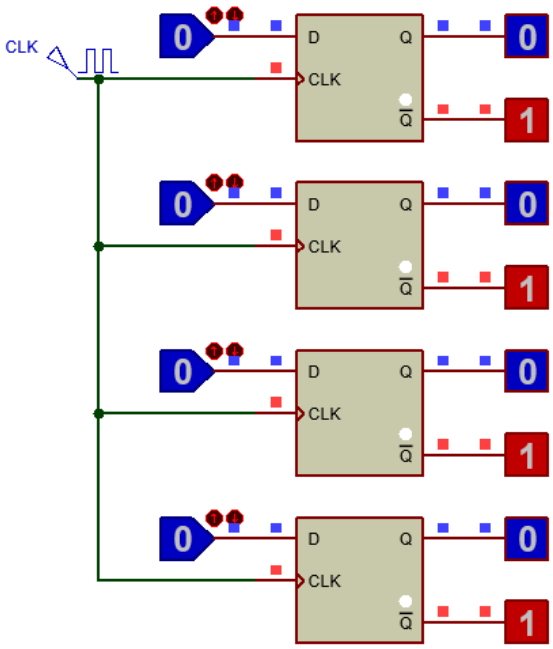


Figura 3.12 Registro de cuatro bits.

3.8 Contador

Los flip-flops se pueden conectar entre sí para realizar conteos; dependiendo de la señal de reloj los contadores se clasifican en dos categorías: asíncronos y síncronos.

Contadores asíncronos

Los contadores asíncronos normalmente llamados contadores con propagación (*ripple counters*) o contadores de rizo, cuentan con una señal externa de reloj que es aplicada al primer flip-flop y posteriormente la señal de salida del primer flip-flop es aplicada como señal de reloj al siguiente flip-flop, este proceso se repite para todos los flip-flops del contador.

Un ejemplo se muestra en la figura 3.13, donde se muestra un contador de cuatro bits que cambia cuando hay una transición negativa del reloj (de alto a bajo). En este contador todas las entradas J y K de los flip-flops son iguales a uno, las salidas D, C, B y A representan un número binario, donde D es el MSB (bit más significativo) [3].

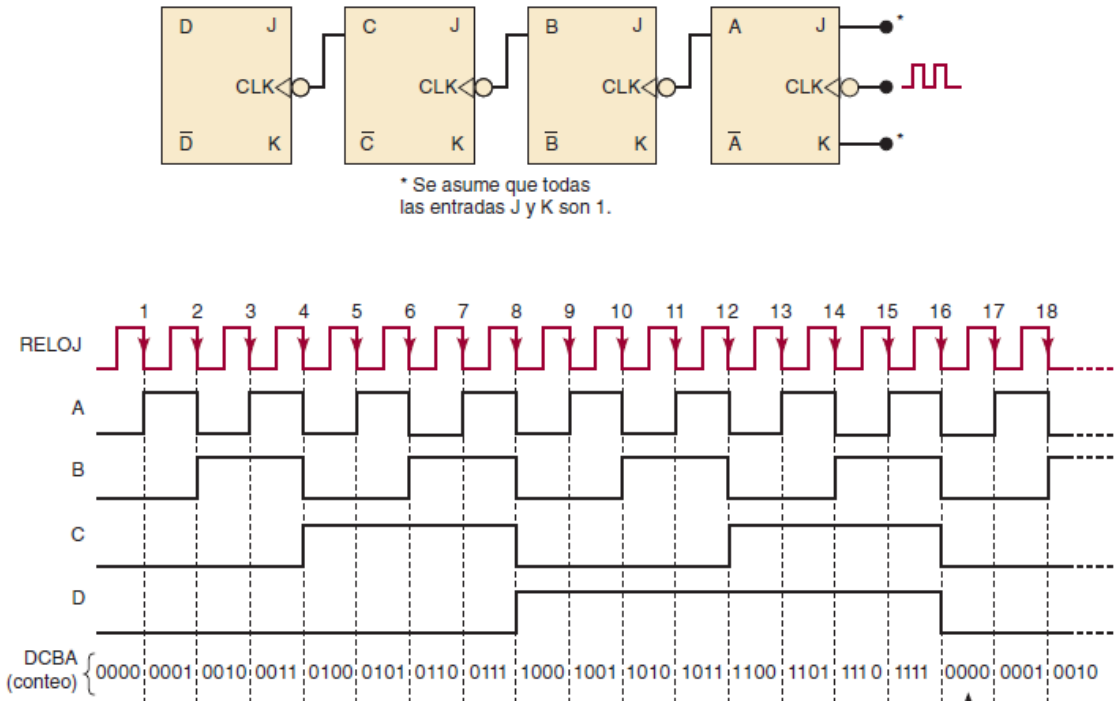


Figura 3.13 Contador asíncrono de cuatro bits [3].

División de frecuencia

Un contador asíncrono también puede ser usado para dividir o reducir la frecuencia de una señal de reloj. Un divisor de frecuencia puede utilizar flip-flops JK, las entradas J y K deben ser iguales a '1', de esta forma la salida del primer flip-flop que se utilice será la mitad de la frecuencia utilizada en la señal de reloj. Para obtener una frecuencia donde N es el número de flip-flops y 2^N es el divisor de la frecuencia, se debe conectar la salida del primer flip-flop a la entrada de reloj del segundo flip-flop y así sucesivamente hasta lograr la división de frecuencia deseada (ver figura 3.14). La señal de salida del último FF (MSB) será la que proporcione la división de la frecuencia. Por ejemplo, si se requiere dividir una frecuencia de 64 Hz a 1 Hz se necesitarán 6 flip-flops, porque $2^6 = 64$ y $\frac{64\ Hz}{64} = 1\ Hz$.

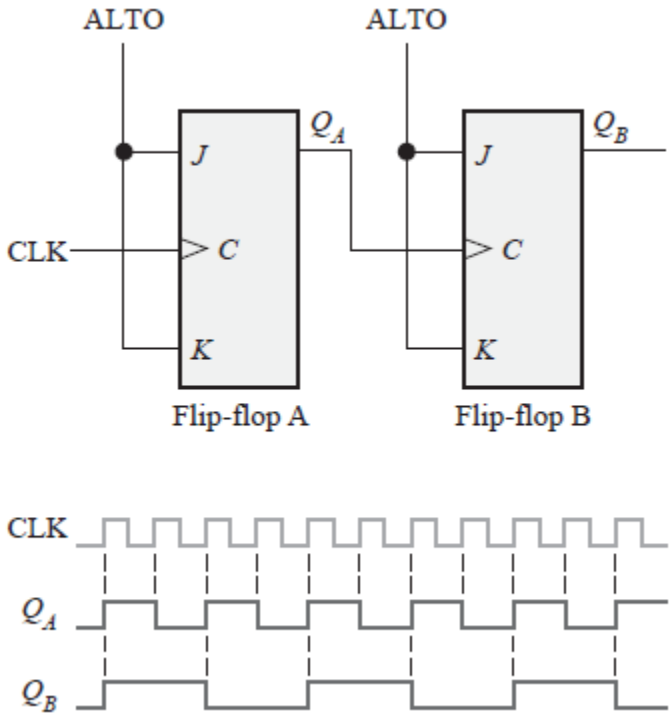


Figura 3.14 Divisor de frecuencia para obtener una frecuencia de reloj dividida por 4 [2].

Contadores síncronos

Los contadores síncronos utilizan la misma señal de reloj para controlar a todos los flip-flops. Un ejemplo se muestra en la figura 3.15, junto con su diagrama de tiempos.

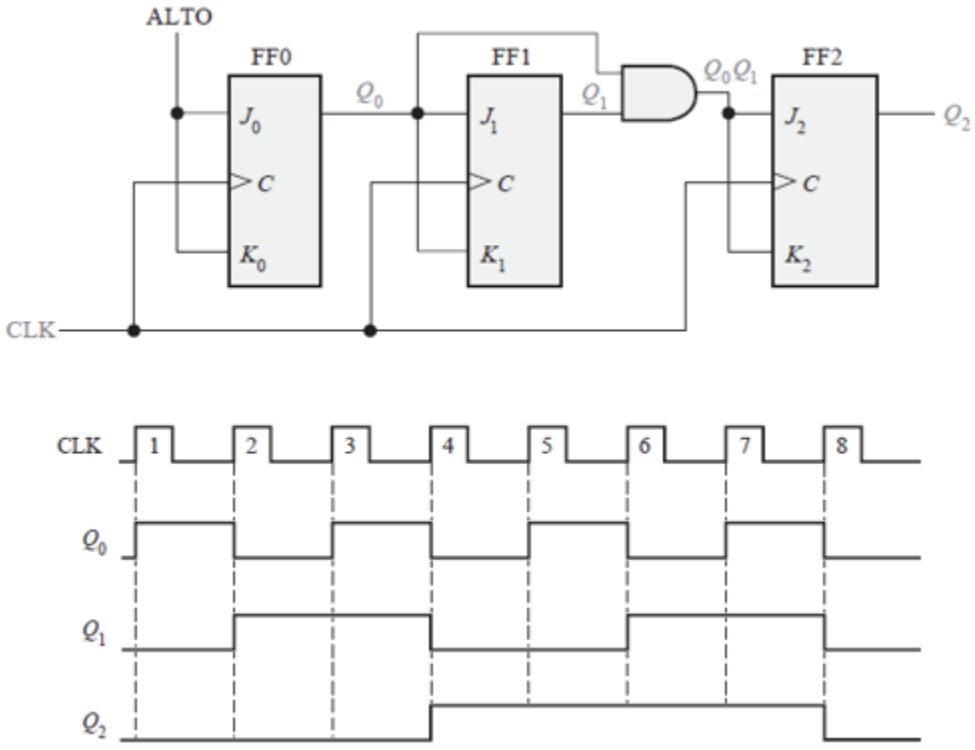


Figura 3.15 Contador binario síncrono de tres bits [2].

Contador con carga paralela

Un contador con carga paralela también se puede entender como un registro el cual, dependiendo de la operación a realizar, puede transferir datos de las entradas de los flip-flops o contar.

En la tabla 3.7 se resume el funcionamiento de un contador binario de cuatro bits con carga paralela mostrado en la figura 3.16. El contador tiene cuatro entradas *Clear*, CLK, *Load* y *Count*. Con las entradas *Load* y *Count* en 0 las salidas no cambian, incluso cuando se aplican pulsos de reloj. Si la entrada de carga es igual a '1', se transfieren los datos de las entradas I_0-I_3 a las salidas A_0-A_3 durante la transición del pulso del reloj. Los datos se transfieren sin importar qué valor tenga la entrada de conteo, porque ésta se inhibe cuando la entrada de carga está habilitada. La entrada de carga debe ser 0 para que la entrada de conteo controle el funcionamiento del contador. Cuando la entrada de conteo es '1' y la de carga es '0', se habilita la operación de contar [1].

Tabla 3.7 Tabla de funcionamiento para un contador de cuatro bits con carga paralela [1].

Clear	CLK	Carga	Conteo	Función
0	X	X	X	Poner en ceros
1	↑	1	*X*	Cargar entradas
1	↑	0	1	Contar al siguiente estado binario
1	↑	0	0	Sin cambio

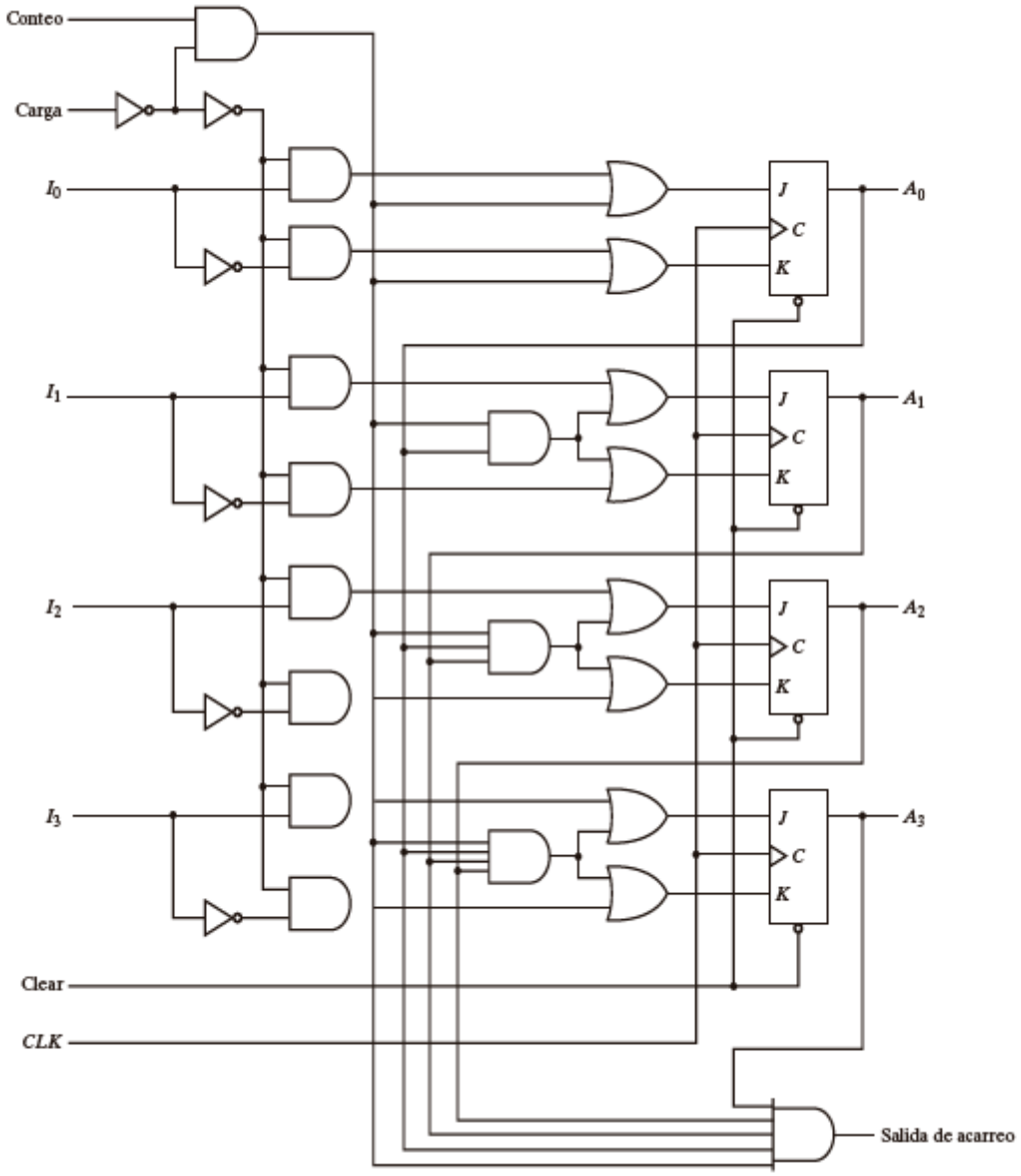


Figura 3.16 Contador binario de cuatro bits con carga paralela [1].

3.9 Memoria ROM

Las memorias de sólo lectura (ROM, por sus siglas en inglés) almacenan datos permanentes o que no cambien con frecuencia. Cuando se está utilizando una memoria ROM no pueden escribirse datos en ella, pero sí se pueden leer. Existen los siguientes tipos de memorias ROM: ROM de máscara, **PROM** (ROM programable), **EPROM** (PROM borrable), **UV EPROM** (EPROM mediante luz ultravioleta) y **EEPROM** o **E²PROM** (PROM borrable eléctricamente).

Para que una memoria ROM pueda ser leída se debe proporcionar un conjunto de bits como entrada (dirección), dependiendo de la dirección se proporcionará una salida de bits. La salida de bits dependerá de la información que almacena la memoria. En la figura 3.17 se muestra un ejemplo de los elementos de una memoria ROM.

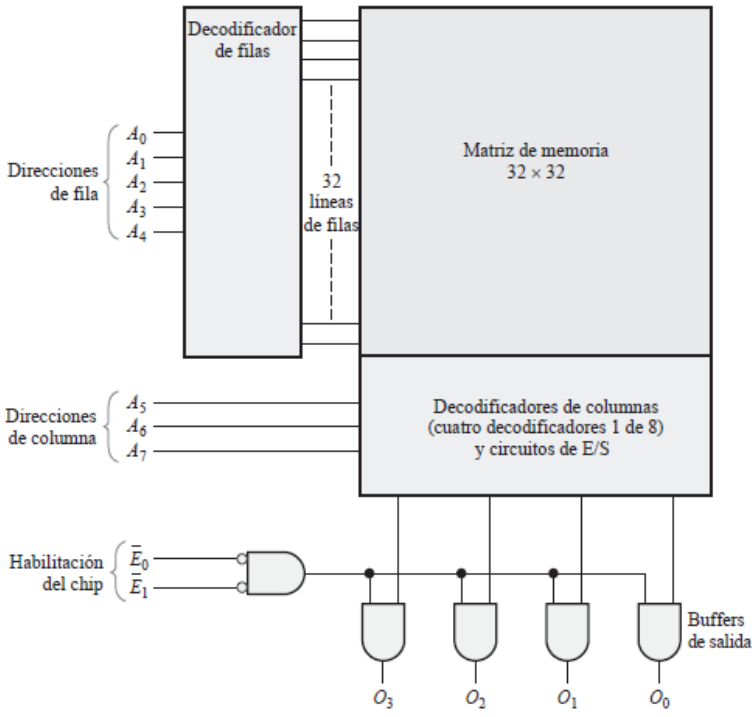


Figura 3.17 ROM de 1024 bits con una organización de 256 x 4 basada en una matriz de 32 × 32 [2].

Un ejemplo es una ROM de 32 x 8 mostrada en la figura 3.18, hay cinco líneas de entrada que forman los números binarios del 0 al 31 para la dirección. Las cinco entradas se decodifican a 32 salidas distintas con un decodificador de 5 x 32 [1].

Cada salida del decodificador representa una dirección de memoria. Las 32 salidas del decodificador se conectan a cada una de las ocho compuertas OR [1]. Cada compuerta OR tiene 32 entradas. Cada salida del decodificador se conecta a una de las entradas de cada compuerta OR. Puesto que cada compuerta OR tiene 32 conexiones de entrada y hay ocho compuertas OR, la ROM contiene 256 conexiones internas (32 x 8) [1].

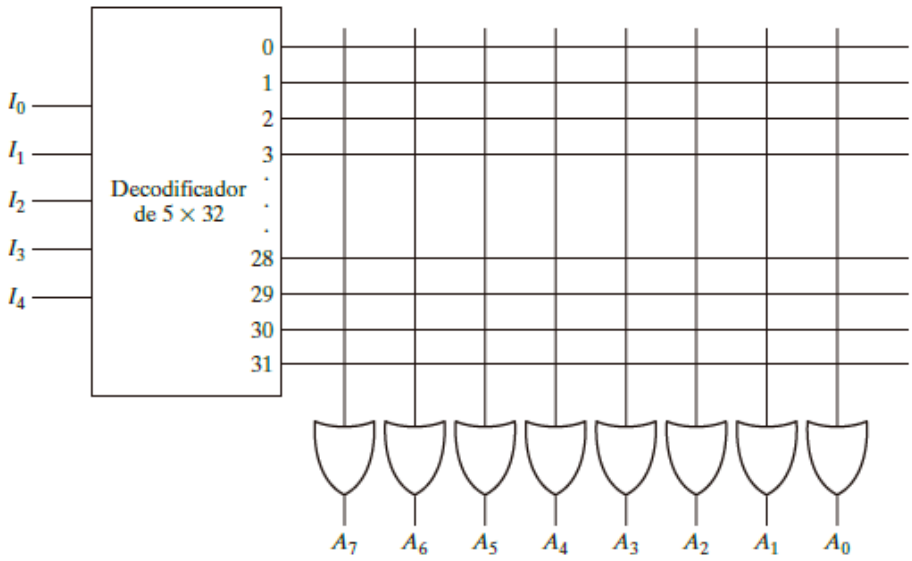


Figura 3.18 Lógica interna de una ROM de 32 x 8 [1].

Las 256 interconexiones de la figura 3.18 son programables. Una conexión programable entre dos líneas equivale a un interruptor que puede estar cerrado (las dos líneas están conectadas) o abierto (las dos líneas están desconectadas). La intersección programable entre dos líneas se conoce como punto de cruce. Se usan diversos dispositivos físicos para implementar interruptores de punto de cruce. Una de las tecnologías más sencillas utiliza un fusible que normalmente conecta los dos puntos, pero que se abre o “quema” con un pulso de alto voltaje [1].

El procedimiento en hardware que programa la ROM hace que se quemen fusibles según una tabla de verdad dada. Por ejemplo, la programación de la ROM según la tabla de verdad de la tabla 3.8 produce la configuración que se aprecia en la figura 3.19. Cada ‘0’ de la tabla de verdad especifica una ausencia de conexión, y cada ‘1’, especifica una trayectoria que se obtiene con una conexión. Por ejemplo, la tabla especifica la palabra de ocho bits 10110010 que se almacenará permanentemente en la dirección 3 [1].

Los cuatro ceros de la palabra se programan quemando los fusibles entre la salida 3 del decodificador y las entradas de las compuertas OR asociadas a las salidas A_6, A_3, A_2 y A_0. Los cuatro unos de la palabra se han marcado en el diagrama con una X para denotar una conexión, en lugar del punto que se usa para indicar una conexión permanente en los diagramas lógicos. Cuando la entrada de la ROM es 00011, todas las salidas del decodificador son '0' excepto la salida 3, que es '1' lógico. La señal equivalente a '1' lógico en la salida 3 del decodificador se propaga por las conexiones hasta las salidas de compuerta OR A_7, A_5, A_4 y A_1. Las otras cuatro salidas siguen en 0. El resultado es que la palabra almacenada 10110010 se aplica a las ocho salidas de datos [1].

Tabla 3.8 Tabla de verdad (parcial) para una ROM [1].

Entradas					Salidas							
I4	**I3**	**I2**	**I1**	**I0**	**A7**	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮					⋮					
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

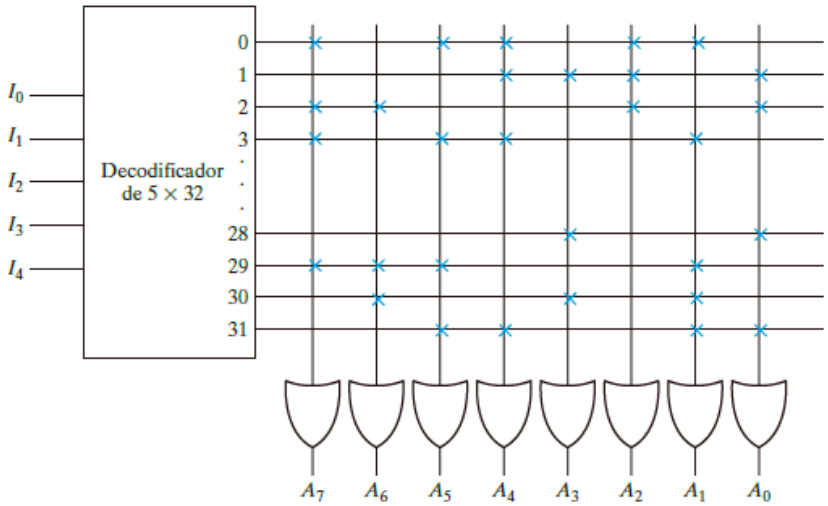


Figura 3.19 Programación de una ROM de acuerdo con la Tabla 3.8 [1].

3.10 Dispositivo lógico programable

Los dispositivos lógico-programables (PLD, por sus siglas en inglés) son circuitos integrados con matrices de compuertas AND y OR programables o fijas, que sirven para implementar circuitos digitales (ver figura 3.20). Los PLD se clasifican de la siguiente manera:

- **SPLD** (dispositivos lógicos programables simples): **EEPROM** (PROM borrable eléctricamente), **PLA** (matriz lógica programable), **PAL** (lógica de matriz programable) y **GAL** (lógica de matriz genérica).
- **CPLD** (dispositivos lógicos programables complejos).
- **FPGA** (matrices de compuertas lógicas programables).

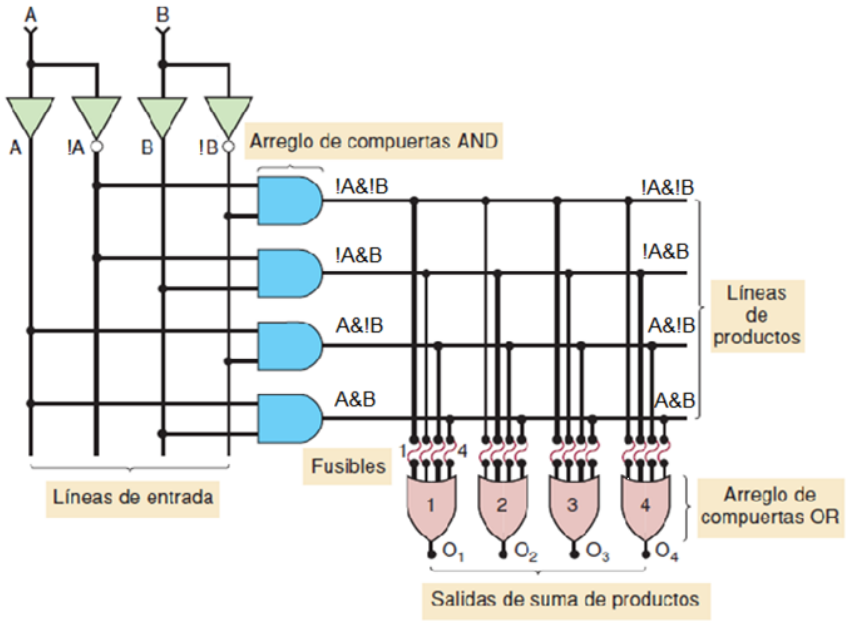


Figura 3.20 Ejemplo de un PLD (imagen modificada) [3].

PAL/GAL

La estructura básica de los dispositivos PAL/GAL es una matriz programable de compuertas AND junto con una matriz fija de compuertas OR; un ejemplo se muestra en la figura 3.21.

Una PAL sólo se puede programar una vez (ver figura 3.22), mientras que una GAL es reprogramable. El término GAL fue utilizado por primera vez por Lattice Semiconductor.

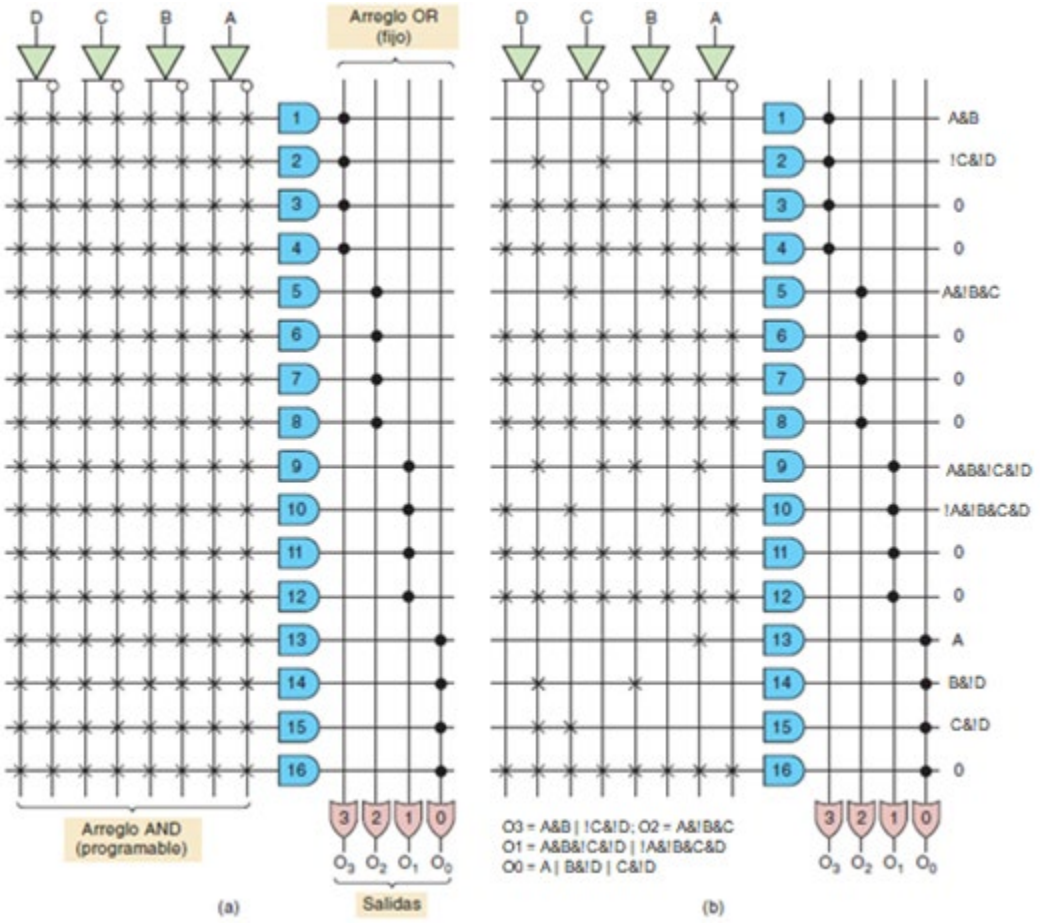


Figura 3.21 Arquitectura de PAL típica y la misma PAL programada con las funciones dadas (imagen modificada) [3].

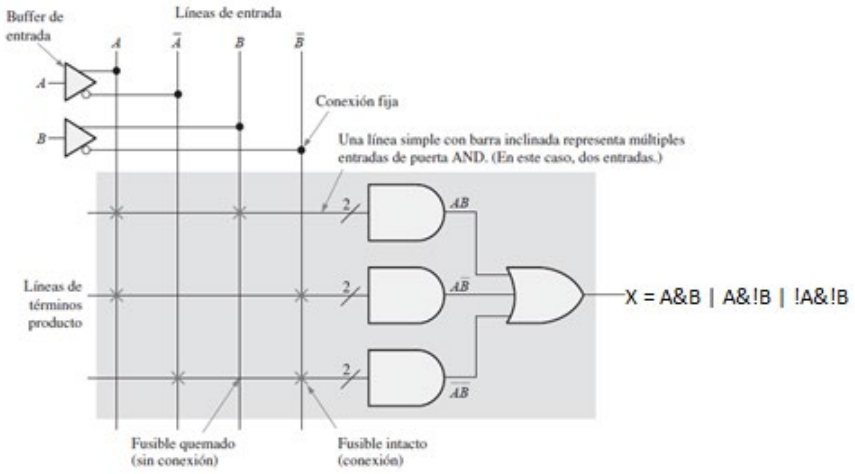


Figura 3.22 Parte de una PAL/GAL programada (imagen modificada) [2].

3.11 Máquina de estado algorítmica

Antes de empezar a describir conceptos, es necesario hablar del creador del diagrama ASM. El creador del diagrama o carta ASM fue el **Dr. Christopher R. Clare**, gerente de proyectos de laboratorio en el Laboratorio de Investigación Electrónica en los Laboratorios de Hewlett-Packard. Este método fue publicado por primera vez en 1973, en el libro "*Designing logic systems using state machines*". Este método gráfico fue creado por el autor para documentar un diseño digital [4], [5].

Un **algoritmo** es un procedimiento que especifica un conjunto de paso ordenados y finitos, que se deben seguir para la solución de un problema. En otras palabras, es como una receta de cocina, pero definida muy cuidadosamente.

La carta o diagrama ASM es una descripción esquemática, usada como ayuda en el diseño de máquinas de estados para implementar un algoritmo y posteriormente formar parte de la documentación de un diseño digital. El diagrama ASM es importante porque describe un algoritmo y una máquina de estados simultáneamente [4].

La carta ASM este compuesta por tres elementos básicos: cuadro de estado, cuadro de decisión y cuadro de salida condicional.

Un **estado** (cuadro de estado) es representado por un rectángulo. En el extremo superior izquierdo debe estar el nombre del estado, en el extremo superior derecho debe estar el código binario del estado y dentro del rectángulo deben estar las salidas del estado en cuestión (ver figura 3.23).

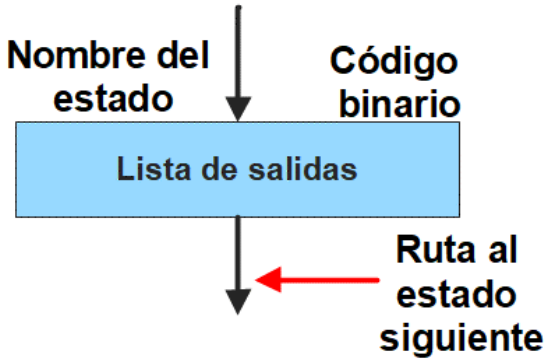


Figura 3.23 Representación de un estado.

Una **decisión** (cuadro de decisión) está representado por un rombo. Describe y evalúa las entradas a la máquina de estados (ver figura 3.24). El cuadro de decisión tiene dos rutas de salida, una ruta es utilizada cuando la condición es verdadera (indicada por un '1') y la otra cuando la condición es falsa (indicada por un '0').

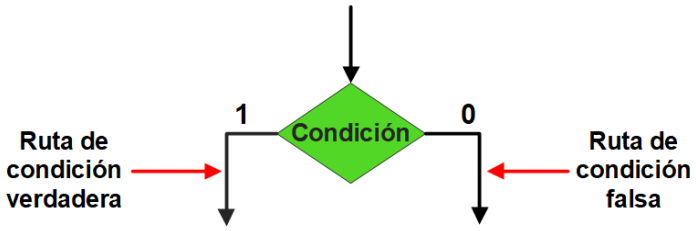


Figura 3.24 Representación de una decisión.

Las salidas en una carta ASM pueden ser condicionales o no condicionales. Las **salidas condicionales** (óvalo de salida condicional) describen salidas que dependen del estado presente y del valor de las señales de entrada. Las salidas condicionales son esencialmente salidas de una máquina de estados Mealy (ver figura 3.25) [6].

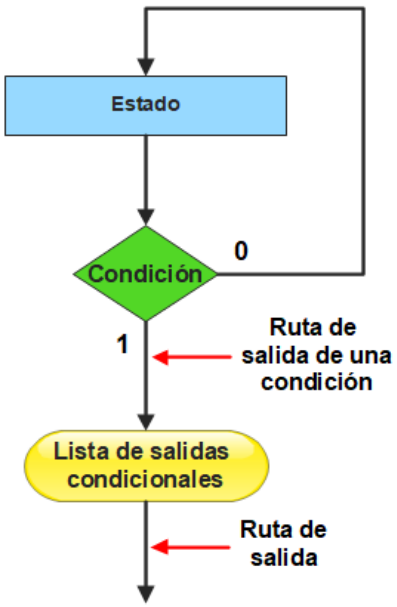


Figura 3.25 Representación de una salida condicional.

Las salidas **no condicionales** dependen sólo del estado presente, no dependen de las señales de entrada. Estás se escriben dentro del estado como se puede ver en la figura 3.23. Las salidas no condicionales son esencialmente salidas de una máquina de estados Moore [6].

Un bloque ASM contiene un cuadro de estado y todos los cuadros de decisión y salidas condicionales asociados a este estado. Un bloque ASM contiene una ruta de entrada y una o más rutas de salida (figura 3.26). Una carta ASM está conformada por uno o más bloques ASM. Cada bloque ASM describe las operaciones que se realizarán en un pulso de reloj.

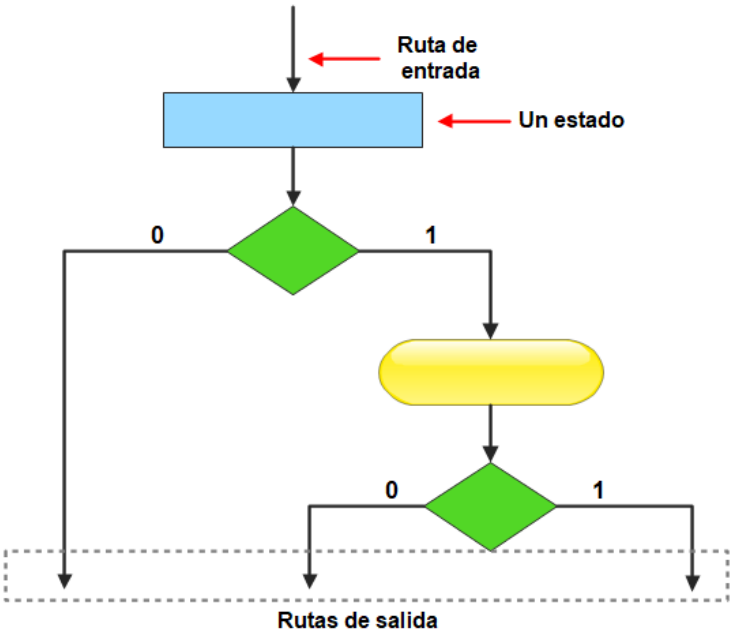


Figura 3.26 Bloque ASM.

Para la construcción de cartas ASM hay algunas consideraciones que deben hacerse. Por ejemplo, la figura 3.27 es una mala representación de la transición de un estado a otro, porque la ruta de salida del primer estado indica que el algoritmo se debe dirigir a dos estados siguientes simultáneamente.

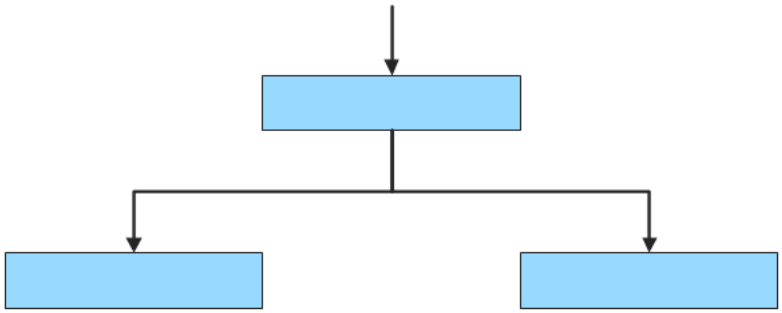


Figura 3.27 Representación incorrecta de una transición de estado.

En la figura 3.28 se muestra una representación incorrecta de una ruta de transición de estados, porque si la Entrada_1 y la Entrada_2 son iguales a uno, el algoritmo debe avanzar a dos estados al mismo tiempo y esto no es posible.

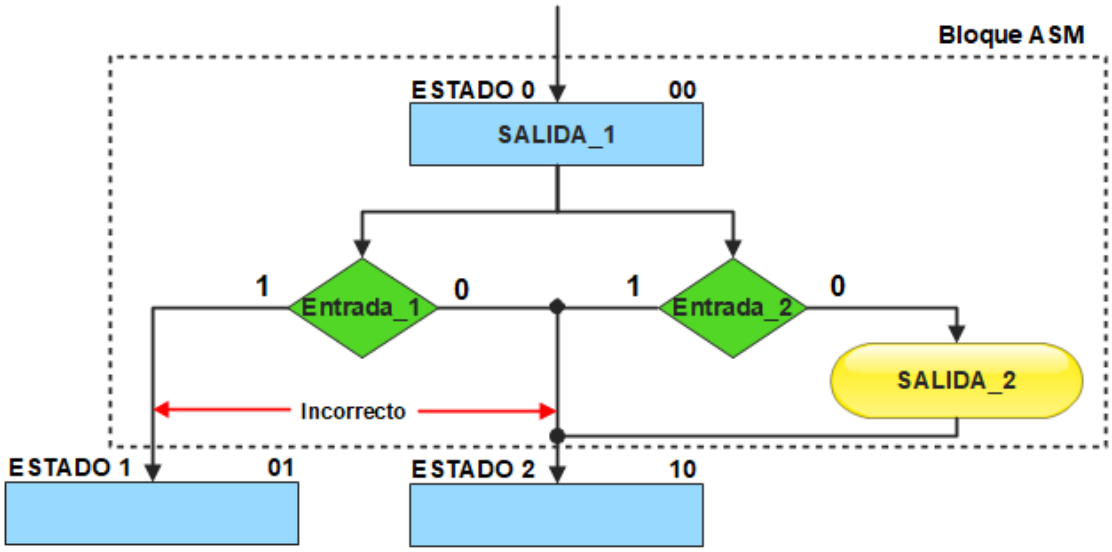


Figura 3.28 Representación indefinida de una ruta de transición de estados.

En la figura 3.29 se muestran dos representaciones de rutas de transición de estados las cuales son similares a la figura 3.28 pero correctamente definidas para permitir el avance del algoritmo a un sólo estado siguiente.

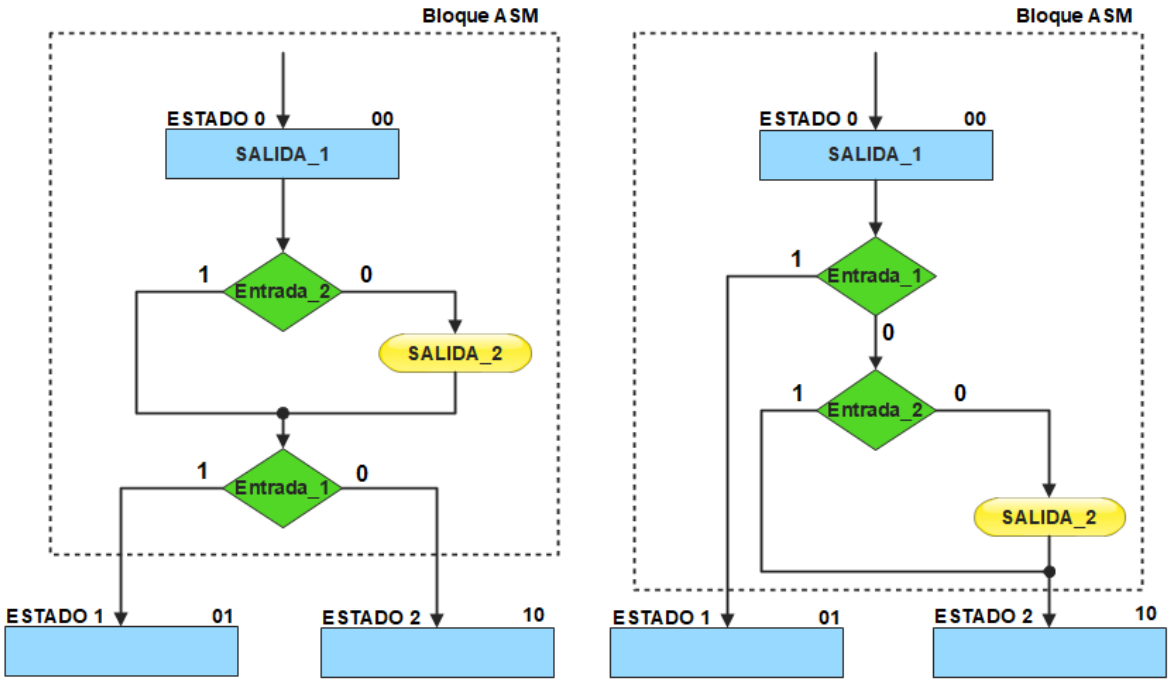


Figura 3.29 Dos posibles representaciones de rutas de transición de estados.

La figura 3.30 muestra, a la izquierda un bloque ASM con retroalimentación interna incorrecto. En la misma figura, al lado derecho se muestra una representación correcta para representar un bloque ASM con retroalimentación. Por lo tanto, se establece que toda combinación de entradas debe conducir a un sólo estado siguiente.

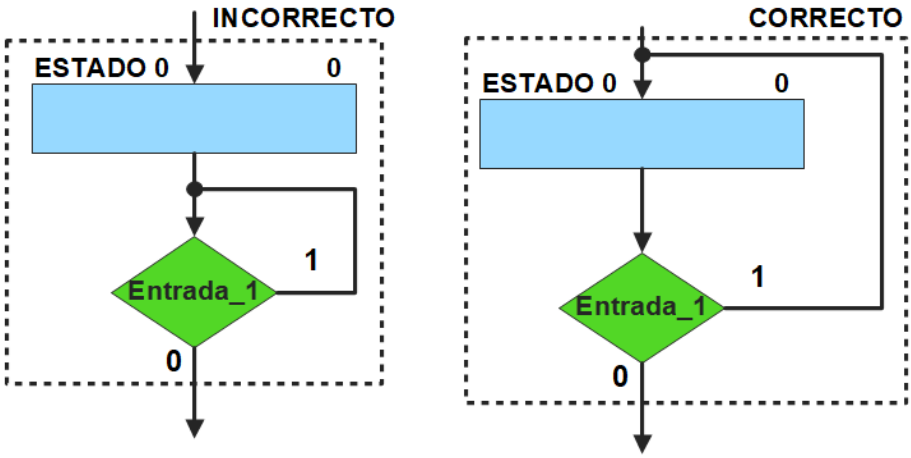


Figura 3.30 Representación incorrecta y correcta un bloque ASM con retroalimentación.

La figura 3.31 muestra la representación de entradas en paralelo y en serie. Ambas representaciones producen la misma operación, es decir son equivalentes. Un bloque ASM puede tener múltiples rutas que dirijan a una misma ruta de salida, más de una ruta puede activarse al mismo tiempo. Por ejemplo, en el bloque ASM en paralelo, si todas las entradas son igual a 1, todas las salidas se activarán y las salidas permanecerán activadas hasta que haya un nuevo pulso de reloj. De la misma manera ocurre con la representación en serie, si todas las entradas son iguales a '1', todas las salidas se activarán y las salidas permanecerán activadas hasta que haya un nuevo pulso de reloj. Cabe mencionar que la representación en paralelo sólo se puede implentar con una de las tres técnicas que se detallarán más adelante.

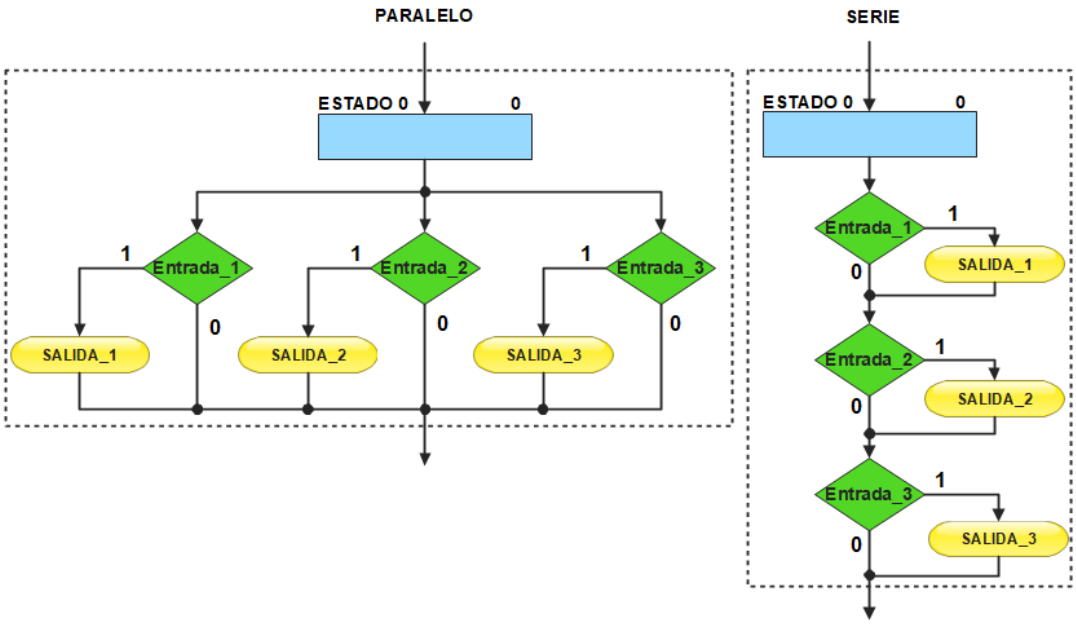


Figura 3.31 Dos representaciones equivalentes de un bloque ASM.

Los cuadros de condición de un bloque ASM también pueden relacionarse entre sí, como se muestra en la figura 3.32, sin embargo, esto no simplifica el algoritmo, sólo ahorra espacio en la escritura.

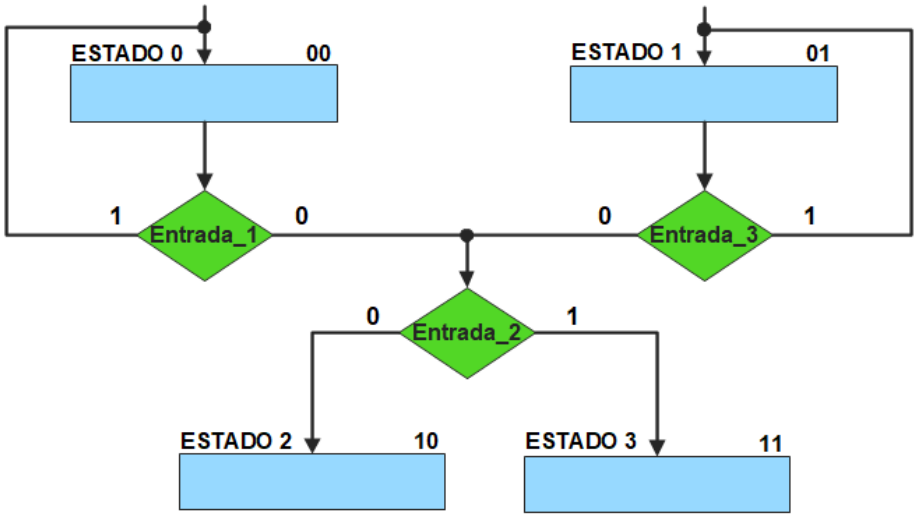


Figura 3.32 Cuadros de condición compartidos.

En la figura 3.33 se muestra una representación de cuadros de decisión equivalentes, en ambos casos, si la función de variables de entrada (A | B&C) es igual a uno, se activa una salida condicional (SALIDA_1).

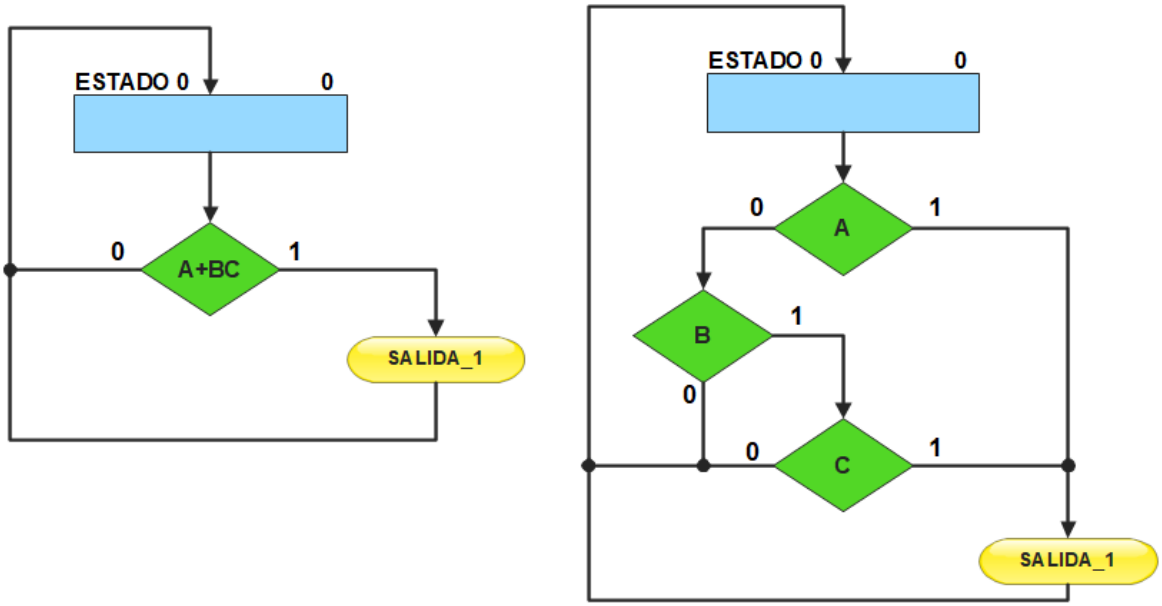


Figura 3.33 Representaciones equivalentes de cuadros de decisión.

Cuando se relacionan dos máquinas de estados, se dice que están enlazadas. Un ejemplo se muestra en la figura 3.34. Cuando se activa la salida OUT_X del estado EST_Y_1, hace que la entrada IN_X sea igual a uno, permitiendo al algoritmo avanzar al siguiente estado. Cuando se activa la salida OUT_Y del estado EST_X_2, hace que la entrada IN_Y sea igual a uno, permitiendo al algoritmo avanzar al siguiente estado. Para este ejemplo se asume que ambas máquinas de estados tienen la misma señal de reloj.

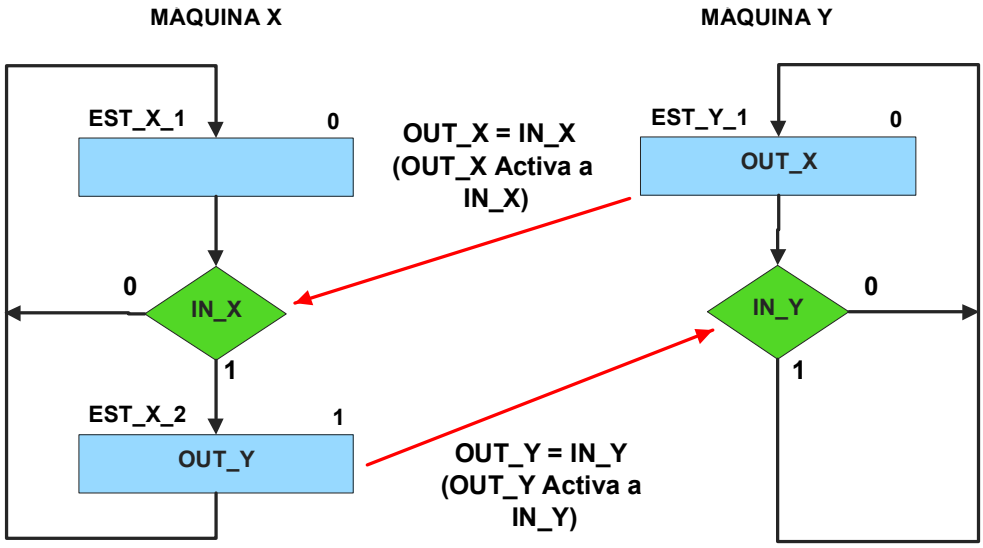


Figura 3.34 Representación de dos máquinas de estados enlazadas.

3.12 Técnicas para implementar cartas ASM

Método tradicional

El primer método para implementar una carta ASM consiste en:

1. Diseñar carta ASM.
2. Tabla de transición de estados y salidas a partir de carta ASM.
3. Mapas de Karnaugh a partir de la tabla.
4. Ecuaciones resultantes de mapas de Karnaugh.

Sin embargo, para este trabajo la implementación del método tradicional será a partir del **método de la variable suscrita** [7], para simplificar los puntos 2, 3 y 4 mencionados anteriormente.

Para entender el funcionamiento de este método se proporcionará un ejemplo.

Se hará la implementación del control de un motor paso a paso utilizando el método de la variable suscrita. Se considera un motor de imán permanente en el rotor como se muestra en la figura 3.35, de modo que se pueda invertir la corriente que circula en sus bobinas.

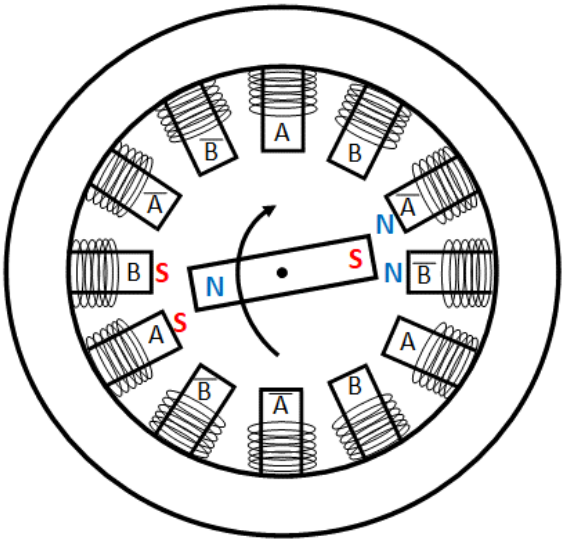


Figura 3.35 Representación de los campos magnéticos de un motor paso a paso en sentido horario, para este caso A es igual a 1 y B es igual a 1.

El patrón de pulsos para este motor paso a paso en sentido horario es el siguiente:

A = 1 1 0 0 1 1 0 0

B = 0 1 1 0 0 1 1 0

El patrón de pulsos para el motor paso a paso en sentido antihorario es el siguiente:

A = 1 0 0 1 1 0 0 1

B = 0 0 1 1 0 0 1 1

A partir de lo descrito anteriormente se diseña una carta ASM como se muestra en la figura 3.36, considerando que la señal M_S es la que controla el sentido de giro del motor y el código binario de los estados representa el patrón de pulsos de las bobinas A y B respectivamente.

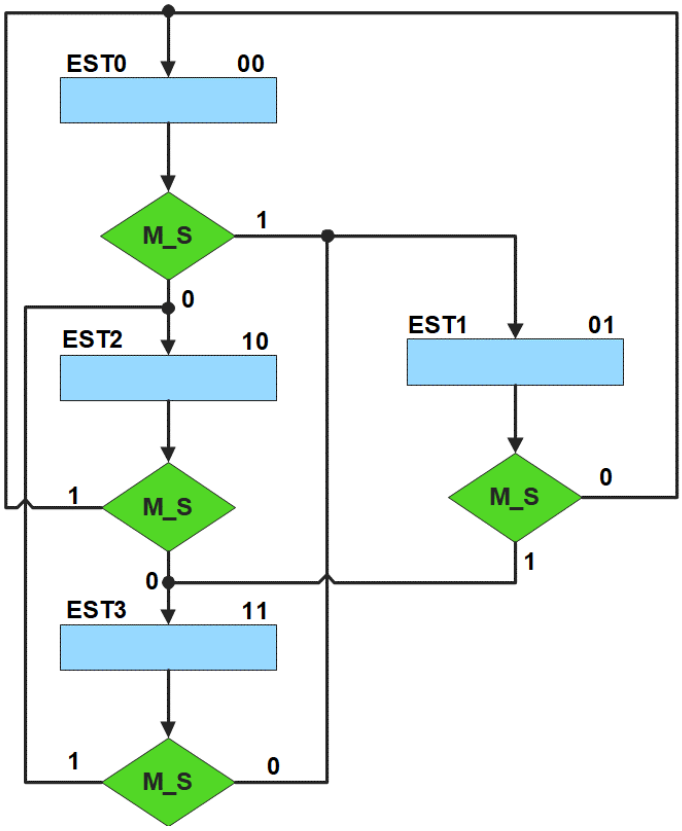


Figura 3.36 Carta ASM de un paso a paso.

A partir de la carta ASM se obtendrán las expresiones lógicas siguiendo el método de la variable suscrita como se muestra en la tabla 3.9.

Tabla 3.9 Mapa de Karnaugh general de transición de estados para el motor paso a paso.

		B	
		0	1
A	0	EST0 - 00	EST1 - 01
		M_S - 01 !M_S - 10	M_S - 11 !M_S - 00
	1	EST2 - 10	EST3 - 11
		M_S - 00 !M_S - 11	M_S - 10 !M_S - 01

Los mapas de Karnaugh particulares para los flip-flops tipo D de los bits A y B son:

A \ B	0	1
0	!M_S	M_S
1	!M_S	M_S

```
DA = !B&!M_Sn | B&M_Sn;
```

A \ B	0	1
0	M_S	M_S
1	!M_S	!M_S

```
DB = !A&M_Sn | A&!M_Sn = A^M_Sn;
```

En la figura 3.37 se muestra la implementación del motor paso a paso por medio del método de la variable suscrita.

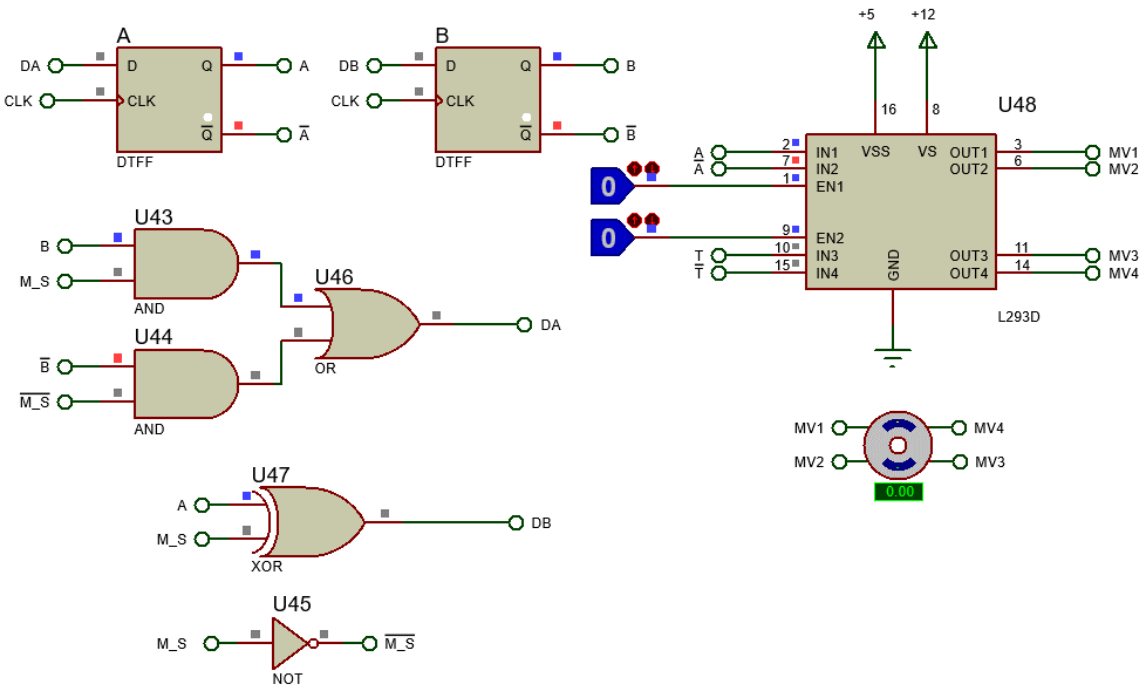


Figura 3.37 Implementación de un motor paso a paso por el método de la variable suscrita.

Microprogramación

La microprogramación es una técnica para implementar la unidad de control de un sistema digital. Consiste en construir una computadora especial para describir el control de un sistema, de manera que sea sencillo pasar de una descripción de un sistema (por ejemplo, una carta ASM) a un programa de computadora (por ejemplo, utilizando instrucciones *if then else*) [8].

La microprogramación puede ser implementada de diferentes maneras. La idea general es almacenar una palabra de control que corresponda a cada estado. La palabra de control también es llamada **microinstrucción** [8]**.**

A continuación, se describirán tres técnicas diferentes para implementar cartas ASM. Ejemplos de estas técnicas se muestran en el Capítulo 4 de este trabajo.

Diseño con memorias y direccionamiento por trayectoria

El diseño con memorias y direccionamiento por trayectoria guarda el estado siguiente de la salida de cada estado de la carta ASM en una localidad de memoria. La porción de la memoria que indica el estado siguiente es llamada “liga”, mientras que la porción que indica las salidas se llama “la parte de las salidas” [9].

La arquitectura de un diseño con memorias y direccionamiento por trayectoria se muestra en la figura 3.38.

Donde:

A = Estado siguiente

B = Entradas del estado siguiente

C = Entradas del estado presente

D = Estado presente

E = Dirección del estado siguiente

F = Salidas del estado presente

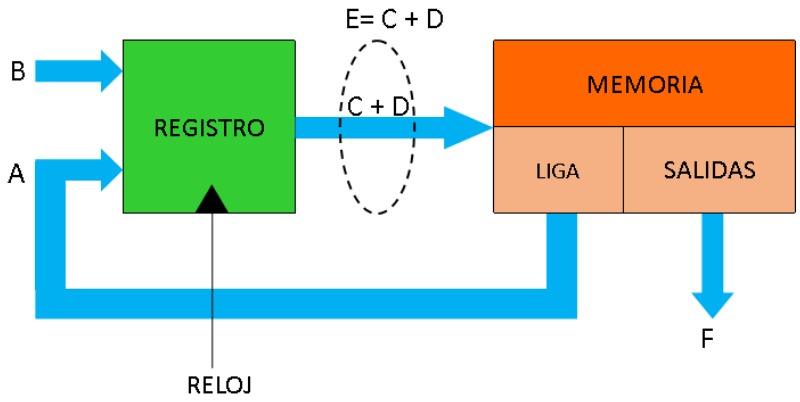


Figura 3.38 Arquitectura de un diseño con direccionamiento por trayectoria.

En este tipo de direccionamiento, todas las salidas de la carta ASM deberán depender del estado presente y de los valores de entrada.

Diseño con memorias y direccionamiento entrada-estado

El diseño con memorias y direccionamiento entrada-estado restringe las cartas ASM a una sola entrada por estado. Una nueva porción de la palabra de memoria contiene una representación binaria de la entrada a probar en cada estado, esta parte es llamada “la parte de prueba”. Con esta representación binaria un selector de entrada elige una de las variables de entrada (ver figura 3.39) [9].

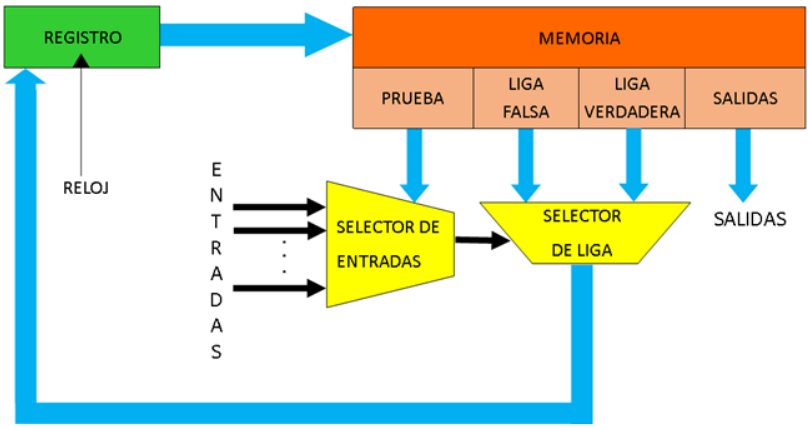


Figura 3.39 Arquitectura de un diseño con direccionamiento entrada-estado.

La parte de liga tiene dos estados siguientes, escogiéndose uno por el selector de liga, con base en la entrada seleccionada por la parte de prueba. Si el valor de la entrada seleccionada por el selector de entradas es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera (ver figura 3.40) [9].

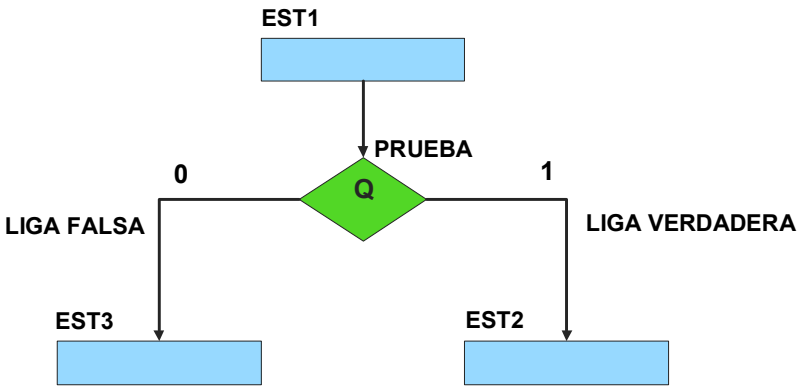


Figura 3.40 La liga falsa es el estado EST3, mientras que la liga verdadera es el estado EST2.

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará también una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista, se probará la variable auxiliar, la cual tiene un valor preestablecido de cero o uno [9].

Diseño con memorias y direccionamiento implícito

El diseño con memorias y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura 3.41). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura 3.42 muestra la arquitectura de este método.

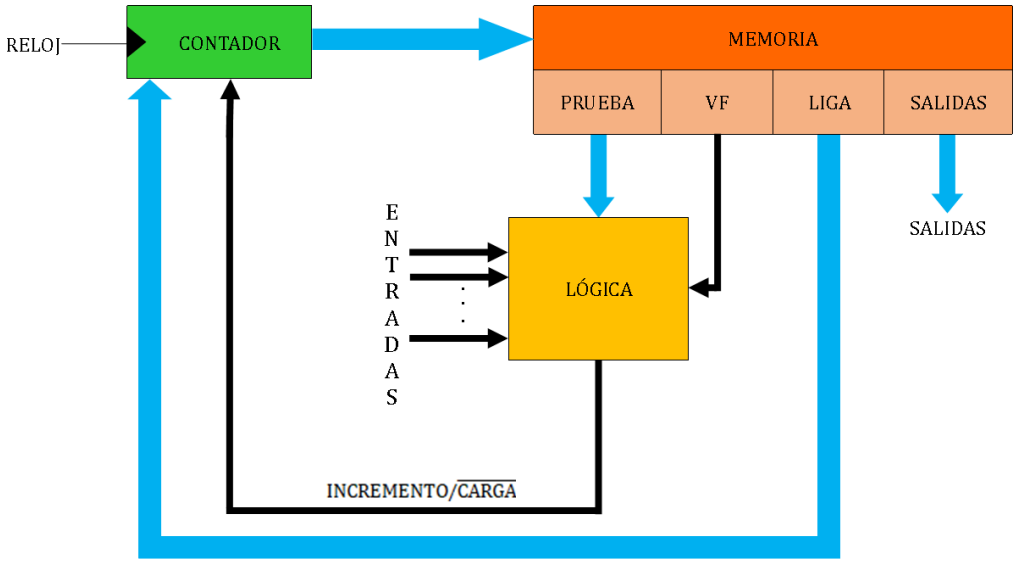


Figura 3.41 Arquitectura de un diseño con direccionamiento implícito.

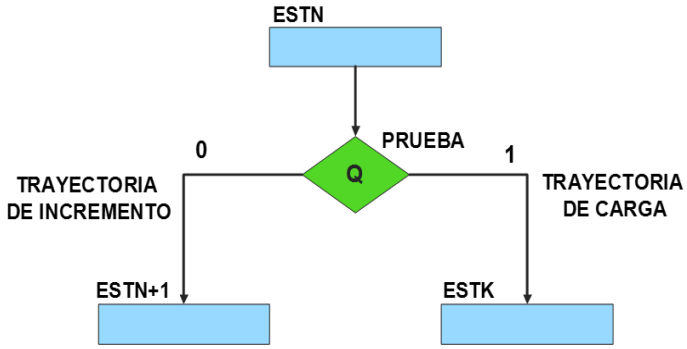


Figura 3.42 Trayectoria de incremento y carga.

La tabla 3.10 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que valor de entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un ‘1’ se hace un incremento, cuando el resultado de la función da como resultado un ‘0’ se hace una carga.

Tabla 3.10 Relación entre VF, variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incremento o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es ‘0’, se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es ‘1’, se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura 3.43).

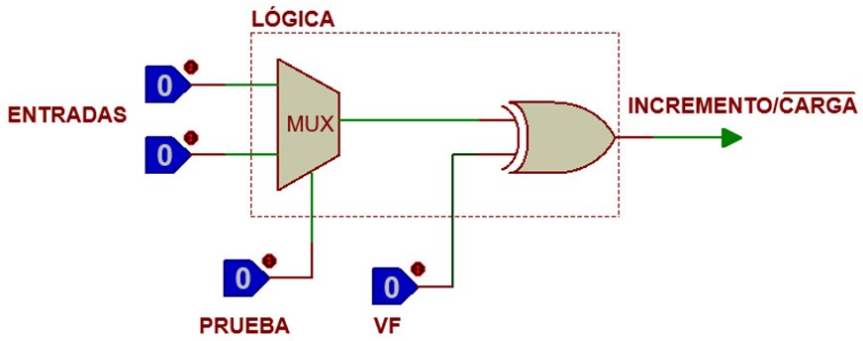


Figura 3.43 Bloque de lógica de incremento/carga para el direccionamiento implícito [9].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

CAPÍTULO 4 MANUAL DE PRÁCTICAS

Se presenta un conjunto de prácticas basadas en tres maquetas didácticas: un estacionamiento, un sistema robotizado clasificador de paquetes y un elevador. Las prácticas fueron realizadas utilizando tres diferentes técnicas para implementar una carta ASM con memorias, con la finalidad de mejorar la comprensión de estas técnicas.

Las prácticas están diseñadas para que estas se comprendan de una manera fácil, utilizando imágenes representativas y describiendo a detalle el funcionamiento de cada práctica.

Las maquetas se dividieron en subsistemas para facilitar la comprensión de su funcionamiento. Las prácticas están basadas en estos subsistemas. La última práctica de cada maqueta engloba todos los subsistemas, formando el sistema de la maqueta en cuestión. La dificultad de las prácticas tendrá un orden ascendente.

Práctica 1 Sistema de emisión/retiro de boletos; diseño con memoria y direccionamiento por trayectoria

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P1.1).

Figura P1.1 Maqueta de estacionamiento con 10 lugares.

En la entrada se debe adquirir un boleto para poder ingresar al estacionamiento. El sistema de emisión/retiro de boletos entrega un boleto de forma automática al presionar un botón, el boleto debe salir a través de una ranura (ver figura P1.2).

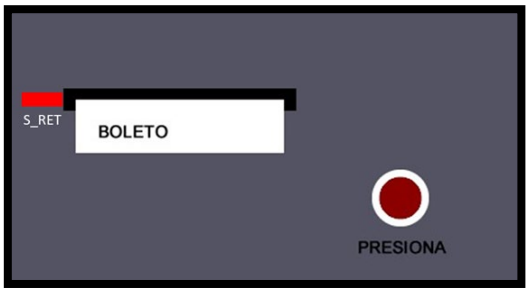


Figura P1.2 Sección de la maqueta en donde se emite el boleto.

El diseño con memoria y direccionamiento por trayectoria guarda el estado siguiente según la salida de cada estado de la carta ASM en una localidad de memoria. La porción de la memoria que indica el estado siguiente es llamada “liga”, mientras que la porción que indica las salidas se llama “la parte de las salidas” [1].

La arquitectura de un controlador con memoria y direccionamiento por trayectoria se muestra en la figura P1.3, donde:

A = estado siguiente

B = entradas del estado siguiente

C = entradas del estado presente

D = estado presente

E = dirección del estado siguiente

F = salidas del estado presente.

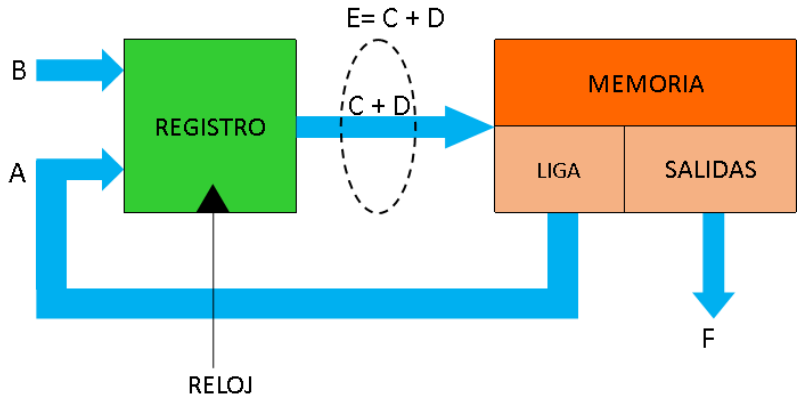


Figura P1.3 Arquitectura de un controlador con memoria y direccionamiento por trayectoria.

En este tipo de diseño, todas las salidas de la carta ASM deberán depender del estado presente y de los valores de entrada.

Objetivo

Diseñar un controlador para el sistema de emisión/retiro de boletos, por medio del método de diseño con memoria y direccionamiento por trayectoria.

Descripción

Primero se diseña una carta ASM para el sistema de emisión/retiro de boletos por medio del método de diseño con memoria y direccionamiento por trayectoria. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P1.1 se muestran los detalles de las entradas y salidas de este controlador.

Para la emisión de boletos se necesita señales de entrada:

- un sensor detecta la presencia de vehículo
- un botón detecta la solicitud de emisión de boleto
- un sensor verifica la emisión y, posteriormente, el retiro del boleto (ver figura P1.2).

Como salida se requiere de las siguientes señales:

- activación del motor de emisión del boleto.

Tabla P1.1 Entradas y salidas para el sistema de emisión/retiro de boletos.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
S_E1	D0 del registro	I	Sensor de entrada que detecta cuando cuándo un vehículo llega a la entrada del estacionamiento
BOT	D1 del registro	I	Botón para obtener un boleto
S_RET	D2 del registro	I	Sensor de emisión y retiro de boleto
M_EBO	D0 de la memoria	O	Motor que emitirá un boleto

Notas de diseño

a) El sensor S_E1 está ubicado antes del cruce de la pluma.
b) El sensor S_RET tiene dos funciones: detectar cuándo el boleto ha sido emitido y cuándo el boleto ha sido retirado.

Reglas de funcionamiento

- S_E1: sensor de entrada
  1 = detecta vehículo
  0 = no detecta vehículo
- BOT: botón para obtener un boleto
  1 = se presionó el botón
  0 = no se presionó el botón
- S_RET: sensor de emisión y retiro de boleto
  1 = detecta que se ha emitido el boleto o detecta que el boleto no ha sido retirado
  0 = detecta que no se ha emitido el boleto o detecta que el boleto ha sido retirado.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado. El algoritmo de la máquina de estados se puede ver en la figura P1.4.

Estado ‘00’ – ENTRADA

Entrada del vehículo. En el primer estado, el controlador de la emisión de boletos se encuentra en espera. El sensor (S_E1), al detectar un vehículo, permite al sistema pasar al Estado ‘01’. De lo contrario, permanece en el Estado ‘00’.

Estado ‘01’ – EBOT

En este estado se espera a que se presione el botón (BOT) para la emisión del boleto. Al ser presionado el botón, el sistema avanza al Estado ‘10’ para emitir el boleto. De lo contrario, regresa al Estado ‘00’ para verificar si aún hay un vehículo presente.

Estado ‘10’ – EBOL

Emisión de boleto. En este estado se activa el motor correspondiente (M_EBO) para que se emita un boleto. Cuando el sensor (S_RET) detecta que el boleto ya se encuentra listo para retirarse, el sistema pasa al Estado ‘11’ donde se desactiva al motor y se espera a que se retire el boleto. Si el sensor aún no detecta el boleto, permanece en el Estado ‘10’.

Estado ‘11’ – ERET

En este estado se espera a que el boleto sea retirado. Cuando el sensor (S_RET) detecta que el boleto se ha retirado, regresa al Estado ‘00’, para iniciar nuevamente el proceso de emisión/retiro de boleto. De lo contrario, permanece en el Estado ‘11’, en espera a que el boleto sea retirado.

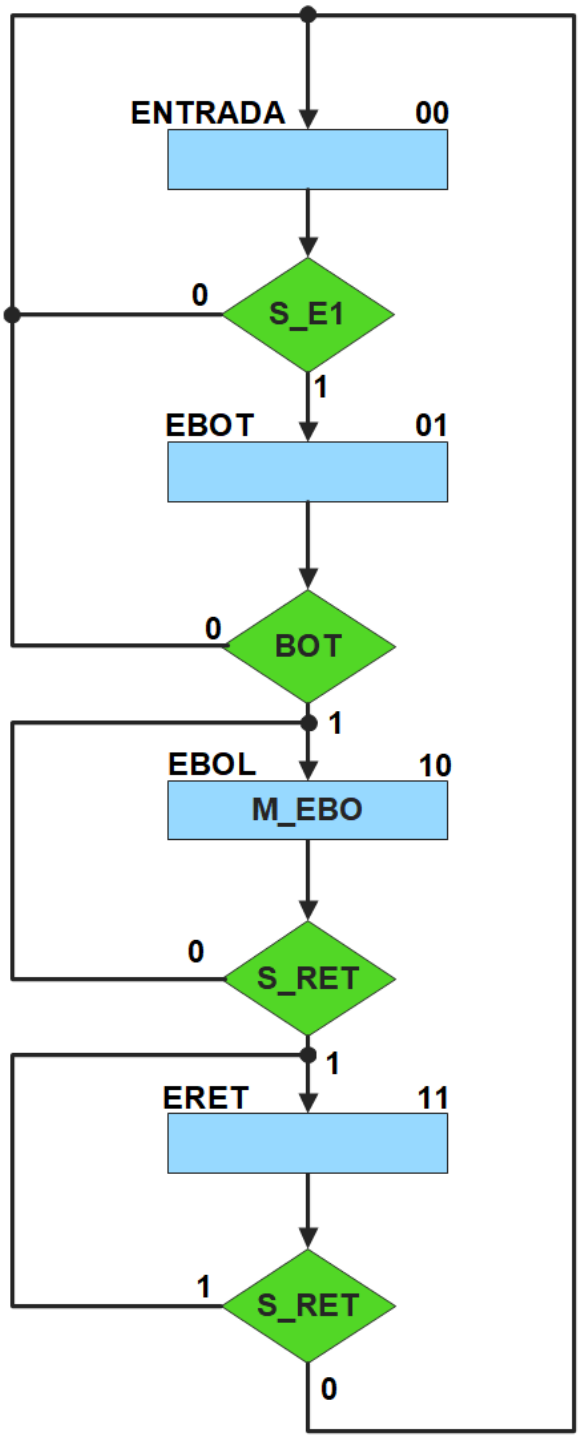


Figura P1.4 Carta ASM del sistema de emisión/retiro de boletos.

Solución

Se debe llenar la tabla P1.2 con base en la información de la carta ASM de la figura P1.4, usando el método de diseño con memoria y direccionamiento por trayectoria. Para cada estado es necesario considerar todas las posibles combinaciones de las variables de entrada, aun cuando algunas de ellas no se utilicen [1].

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘00’.

Debido a que hay tres variables de entrada se deben considerar 8 posibles combinaciones de estas para cada estado. Si en el Estado ‘00’ la variable S_E1 es igual a ‘1’, el estado siguiente será el Estado ‘01’ independientemente de los valores de las otras variables. De lo contrario el estado siguiente será el Estado ‘00’. En el Estado ‘00’ el motor M_EBO no está activado, por lo que se coloca un ‘0’ en la parte de salidas de todo el estado.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P1.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P1.2 Contenido de la memoria para el sistema de emisión/retiro de boletos.

	Dirección de memoria					Contenido de memoria			
	Estado presente		Entradas			Liga		Salida	HEX
	QA	QB	S_E1	BOT	S_RET	QA	QB	M_EBO	
Estado 00	0	0	0	0	0	0	0	0	00
	0	0	0	0	1	0	0	0	00
	0	0	0	1	0	0	0	0	00
	0	0	0	1	1	0	0	0	00
	0	0	1	0	0	0	1	0	02
	0	0	1	0	1	0	1	0	02
	0	0	1	1	0	0	1	0	02
	0	0	1	1	1	0	1	0	02
Estado 01	0	1	0	0	0	0	0	0	00
	0	1	0	0	1	0	0	0	00
	0	1	0	1	0	1	0	0	04
	0	1	0	1	1	1	0	0	04
	0	1	1	0	0	0	0	0	00
	0	1	1	0	1	0	0	0	00
	0	1	1	1	0	1	0	0	04
	0	1	1	1	1	1	0	0	04
Estado 10	1	0	0	0	0	1	0	1	05
	1	0	0	0	1	1	1	1	07
	1	0	0	1	0	1	0	1	05
	1	0	0	1	1	1	1	1	07
	1	0	1	0	0	1	0	1	05
	1	0	1	0	1	1	1	1	07
	1	0	1	1	0	1	0	1	05
	1	0	1	1	1	1	1	1	07
Estado 11	1	1	0	0	0	0	0	0	00
	1	1	0	0	1	1	1	0	06
	1	1	0	1	0	0	0	0	00
	1	1	0	1	1	1	1	0	06
	1	1	1	0	0	0	0	0	00
	1	1	1	0	1	1	1	0	06
	1	1	1	1	0	0	0	0	00
	1	1	1	1	1	1	1	0	06

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P1.5,figura P1.6). Se carga en el controlador el archivo con extensión “HEX” de la memoria. Se debe seguir la descripción de la carta ASM para probar la implementación de esta práctica.

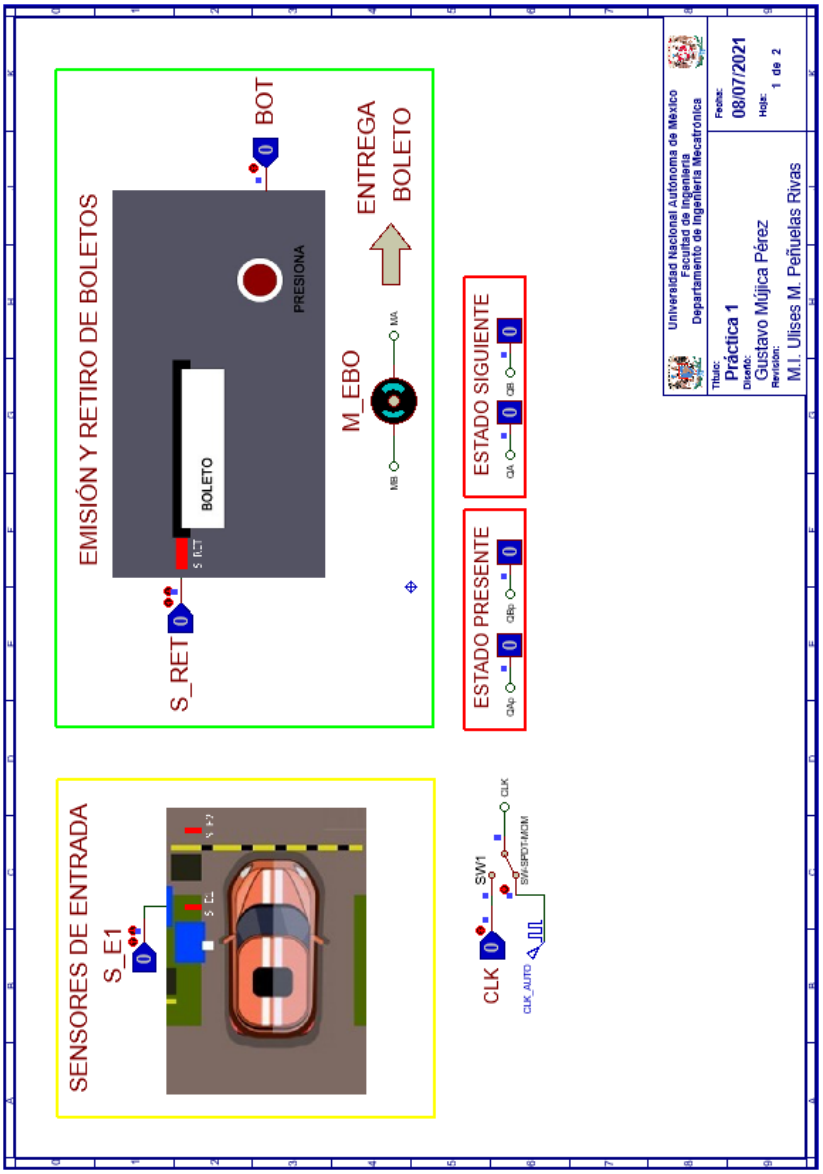


Figura P1.5 Interfaz hombre-máquina para el controlador de la Práctica 1 hoja 1/2.

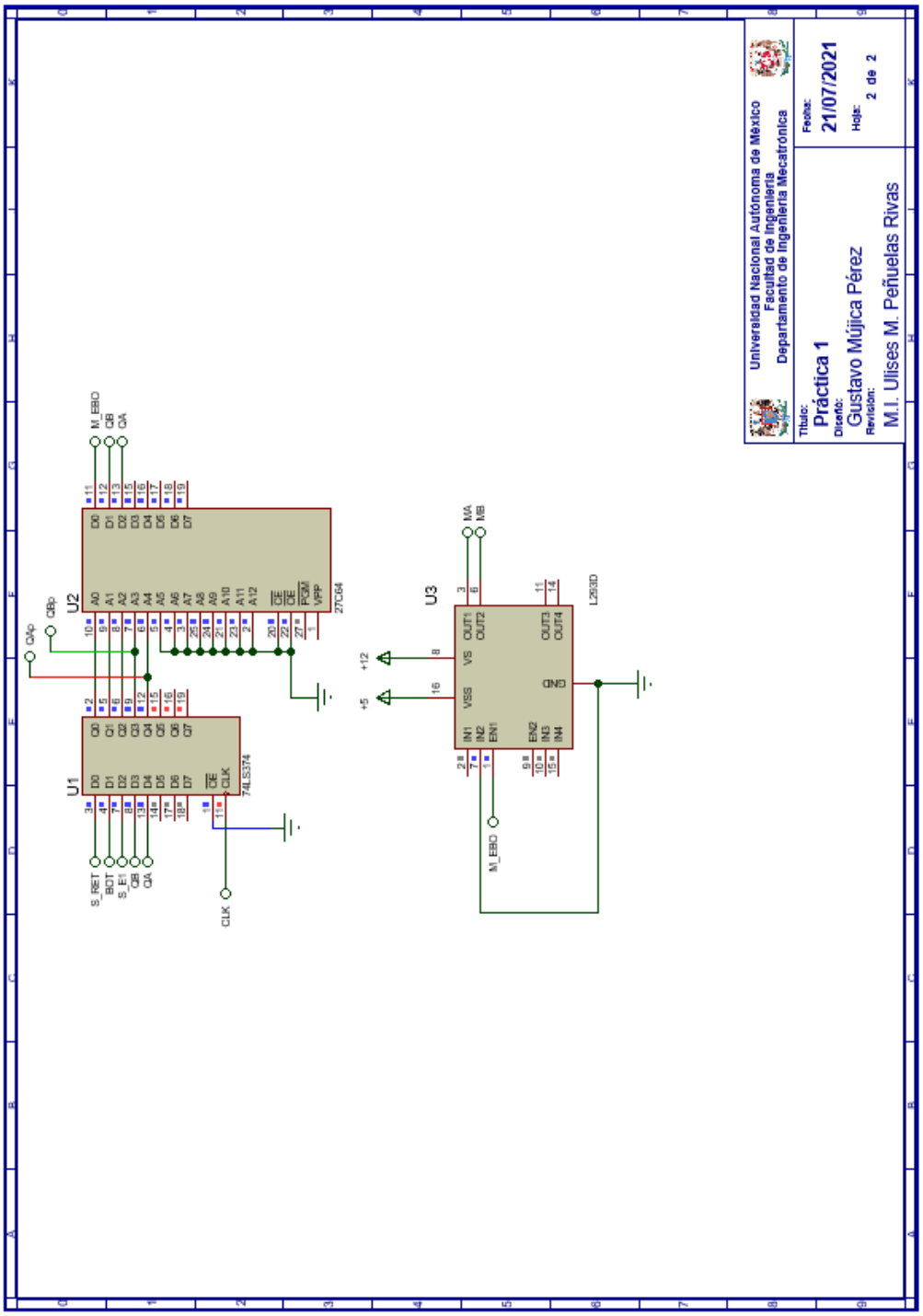


Figura P1.6 Esquema electrónico para el controlador de la Práctica 1 hoja 2/2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 2 Sistema de pluma para estacionamiento; diseño con memoria y direccionamiento por trayectoria

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P2.1).

Figura P2.1 Maqueta de estacionamiento con 10 lugares.

Se usan plumas que impiden el avance del vehículo a la entrada y salida del estacionamiento. Se levanta la pluma para permitir el paso de un vehículo y cuando éste haya pasado completamente, se baja la pluma (ver figura P2.2, figura P2.3).

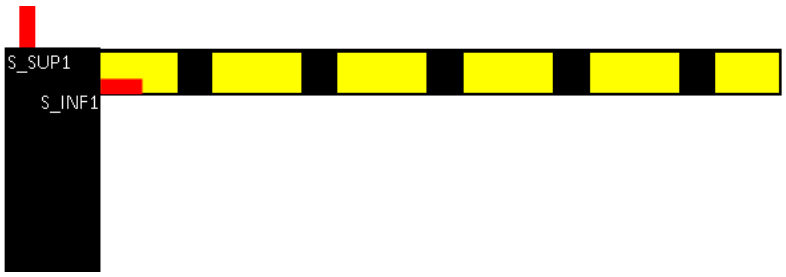


Figura P2.2 Sensores de posición superior e inferior en la pluma.

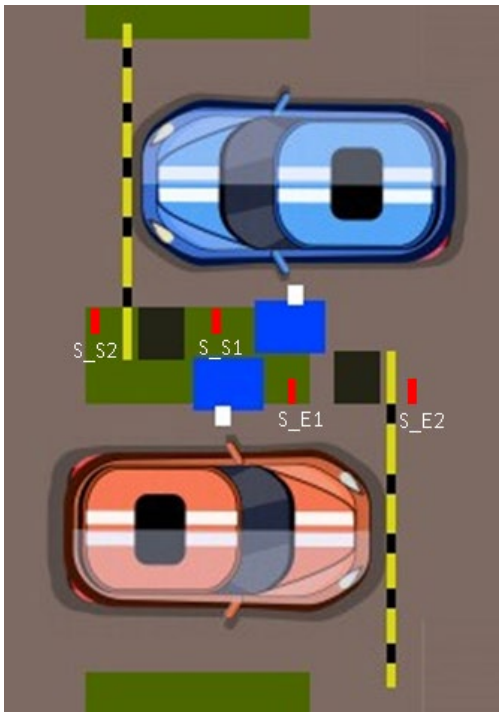


Figura P2.3 Sensores a la entrada y salida del estacionamiento.

El diseño con memoria y direccionamiento por trayectoria guarda el estado siguiente de la salida de cada estado de la carta ASM en una localidad de memoria. La porción de la memoria que indica el estado siguiente es llamada “liga”, mientras que la porción que indica las salidas se llama “la parte de las salidas” [1].

La representación del diseño con memoria y direccionamiento por trayectoria se muestra en la figura P2.4, donde:

A = estado siguiente

B = entradas del estado siguiente

C = entradas del estado presente

D = estado presente

E = dirección del estado siguiente

F = salidas del estado presente.

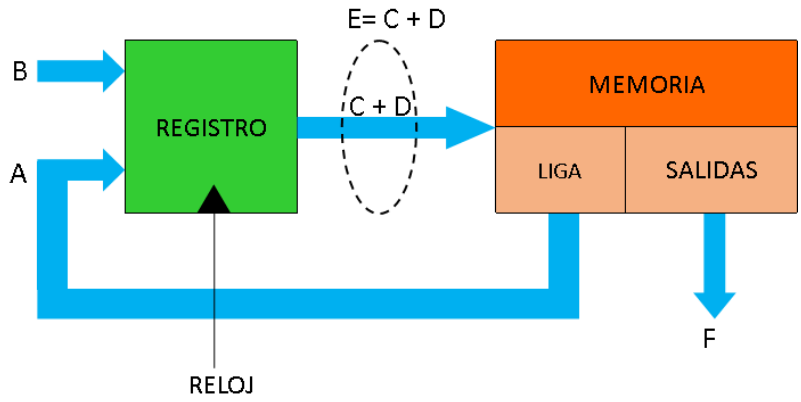


Figura P2.4 Arquitectura de un controlador con memoria y direccionamiento por trayectoria.

En este tipo de diseño todas las salidas de la carta ASM deberán depender del estado presente y de los valores de entrada.

Objetivo

Diseñar un controlador para el sistema de una pluma para estacionamiento, por medio del método de diseño con memoria y direccionamiento por trayectoria.

Descripción

Primero se diseña una carta ASM para el sistema de una pluma para estacionamiento por medio del método de diseño con memoria y direccionamiento por trayectoria. Posteriormente se propone una solución para implementar el sistema.

Nota: Para este diseño se usarán los sensores de la pluma de entrada, sin embargo, el proceso es el mismo para la pluma de entrada como para la pluma de salida.

Tabla de entradas y salidas

En la tabla P2. 1 se muestran los detalles de las entradas y salidas de este controlador.

Para la pluma del estacionamiento se necesitan señales de entrada:

- un sensor detecta cuándo la pluma se encuentra en la posición superior y otro cuándo se encuentra en la posición inferior (ver figura P2.2)
- un sensor detecta cuándo un vehículo está a la entrada del estacionamiento y otro cuando se esté ingresando a éste (ver figura P2.3).

Como salida se requiere de las siguientes señales:

- una señal activa el motor de la pluma
- una señal se activa para que la pluma suba y otra para que la pluma baje.

Tabla P2. 1 Entradas y salidas para el sistema de pluma para estacionamiento.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
S_E1	D0 del registro	I	Sensor de entrada que detecta cuándo un vehículo llega a la entrada del estacionamiento
S_E2	D1 del registro	I	Sensor de ingreso que detecta cuándo un vehículo está ingresando al estacionamiento
S_SUP1	D2 del registro	I	Sensor de posición superior que detecta cuándo la pluma se ha elevado en su totalidad
S_INF1	D3 del registro	I	Sensor de posición inferior que detecta cuándo la pluma se ha bajado en su totalidad
MOT_1	D2 de la memoria	O	Motor de la pluma
MX_1	D1 de la memoria	O	Señal para que la pluma suba
nMX_1	D0 de la memoria	O	Señal para que la pluma baje.

Notas de diseño

a) El sensor S_E1 está ubicado antes del cruce de la pluma

b) El sensor S_E2 está ubicado después del cruce de la pluma.

Reglas de funcionamiento

- S_E1: sensor de entrada
  1 = detecta vehículo
  0 = no detecta vehículo
- S_E2: sensor de ingreso
  1 = detecta vehículo ingresando
  0 = no detecta vehículo ingresando
- S_SUP1: sensor de posición superior
  1 = detecta pluma completamente elevada
  0 = no detecta pluma completamente elevada
- S_INF1: sensor de posición inferior
  1 = detecta que la pluma ha bajado completamente
  0 = no detecta que la pluma ha bajado completamente.

Descripción de la Carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado. El algoritmo de la máquina de estados se puede ver en la figura P2.5.

Estado '00' – ELEVA

En este estado se activa el motor de la pluma (MOT_1) y la señal (MX_1) para que la pluma suba, permitiendo el ingreso del vehículo al estacionamiento. Si el sensor (S_SUP1) detecta que la pluma ha sido elevada completamente, el sistema avanza al Estado '01' para esperar a que pase el vehículo. De lo contrario se verifica si aún hay un vehículo presente a la entrada del estacionamiento.

Si el sensor (S_E1) no detecta un vehículo a la entrada, el sistema avanza a verificar si hay uno ingresando al estacionamiento. Cuando se detecta un vehículo a la entrada, permanece en el Estado ´00´.

Si el sensor (S_E2) detecta un vehículo ingresando, el sistema avanza al Estado '10' para bajar la pluma, porque el vehículo ha pasado antes de que la pluma se eleve completamente. Si se detecta un vehículo ingresando, permanece en el Estado '01'.

Estado '01' – ESPERA

En este estado la pluma esta elevada, esperando a que el vehículo pase completamente. Si el sensor (S_E1) no detecta un vehículo a la entrada del estacionamiento, el sistema avanza a revisar si hay un vehículo ingresando. De lo contrario permanece en el Estado '01'.

Cuando el sensor (S_E2) no detecta un vehículo ingresando, el sistema pasa al Estado '10' para que se baje la pluma ya que el vehículo ha pasado completamente. De lo contrario permanece en el Estado '01'.

Estado '10' – BAJA

En este estado, se activa el motor de la pluma (MOT_1) y la señal (nMX_1) para que la pluma baje. Si el sensor (S_INF1) detecta que la pluma ha bajado completamente, el sistema regresa al Estado '00' para elevar la pluma nuevamente. Cuando se detecta que la pluma no ha bajado en su totalidad, avanza a revisar si hay un auto a la entrada del estacionamiento.

Cuando el sensor (S_E1) detecta un vehículo a la entrada, el sistema regresa al Estado ‘00’ para subir la pluma porque ésta puede dañar al vehículo. De lo contrario avanza a revisar si hay un vehículo ingresando.

Si el sensor (S_E2) detecta un vehículo ingresando, el sistema regresa al Estado ‘00’ para subir la pluma porque ésta puede dañar al vehículo. De lo contrario permanece en el Estado ‘10’.

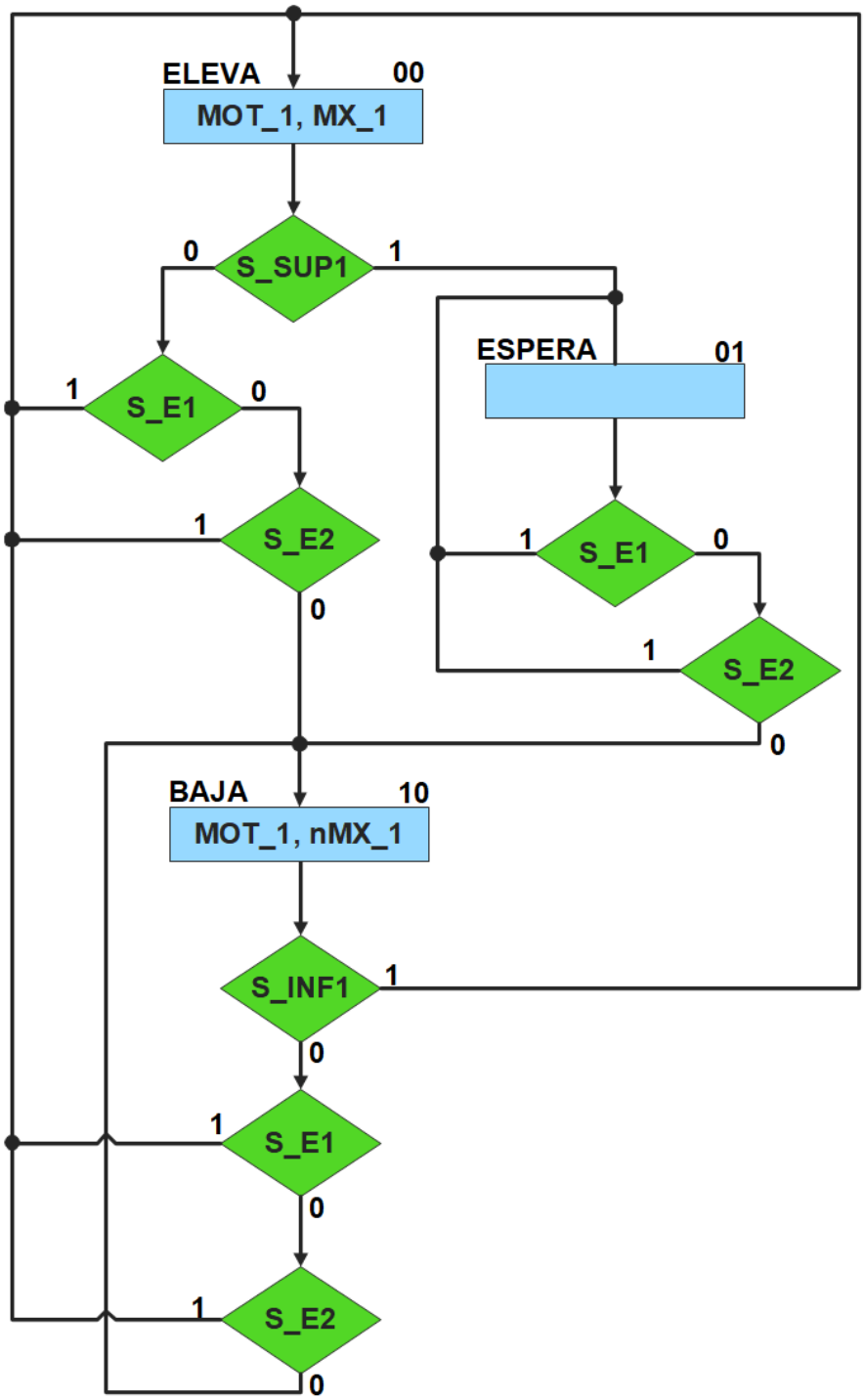


Figura P2.5 Carta ASM del sistema de pluma para estacionamiento.

Solución

Se debe llenar las tablas tabla P2.2 y tabla P2.3 con base en la información de la carta ASM de la figura P2.5, usando el método de diseño con memoria y direccionamiento por trayectoria. Para cada estado es necesario considerar todas las posibles combinaciones de las variables de entrada aun cuando algunas de ellas no se utilicen [1].

A continuación, se describe como llenar los campos de la memoria para el Estado ‘00’.

Debido a que hay cuatro variables de entrada se deben considerar 16 posibles combinaciones de éstas para cada estado. Si en el Estado ‘00’ la variable S_SUP1 es igual a ‘1’, el estado siguiente será el Estado ‘01’ independientemente de los valores de las otras variables. En el Estado ‘00’ el motor MOT_1 y la señal MX_1 están activados, por lo que se coloca un ‘1’ en la parte de salidas de todo el estado.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P2.4). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P2.2 Contenido de la memoria para el sistema de una pluma para estacionamiento.

	Dirección de memoria						Contenido de memoria					
	Estado presente		Entradas				Liga		Salidas			HEX
	QA	QB	S_E1	S_E2	S_SUP1	S_INF1	QA	QB	MOT_1	MX_1	nMX_1	
Estado 00	0	0	0	0	0	0	1	0	1	1	0	16
	0	0	0	0	0	1	1	0	1	1	0	16
	0	0	0	0	1	0	0	1	1	1	0	0E
	0	0	0	0	1	1	0	1	1	1	0	0E
	0	0	0	1	0	0	0	0	1	1	0	06
	0	0	0	1	0	1	0	0	1	1	0	06
	0	0	0	1	1	0	0	1	1	1	0	0E
	0	0	0	1	1	1	0	1	1	1	0	0E
	0	0	1	0	0	0	0	0	1	1	0	06
	0	0	1	0	0	1	0	0	1	1	0	06
	0	0	1	0	1	0	0	1	1	1	0	0E
	0	0	1	0	1	1	0	1	1	1	0	0E
	0	0	1	1	0	0	0	0	1	1	0	06
	0	0	1	1	0	1	0	0	1	1	0	06
	0	0	1	1	1	0	0	1	1	1	0	0E
	0	0	1	1	1	1	0	1	1	1	0	0E
Estado 01	0	1	0	0	0	0	1	0	0	0	0	10
	0	1	0	0	0	1	1	0	0	0	0	10
	0	1	0	0	1	0	1	0	0	0	0	10
	0	1	0	0	1	1	1	0	0	0	0	10
	0	1	0	1	0	0	0	1	0	0	0	08
	0	1	0	1	0	1	0	1	0	0	0	08
	0	1	0	1	1	0	0	1	0	0	0	08
	0	1	0	1	1	1	0	1	0	0	0	08
	0	1	1	0	0	0	0	1	0	0	0	08
	0	1	1	0	0	1	0	1	0	0	0	08
	0	1	1	0	1	0	0	1	0	0	0	08
	0	1	1	0	1	1	0	1	0	0	0	08
	0	1	1	1	0	0	0	1	0	0	0	08
	0	1	1	1	0	1	0	1	0	0	0	08
	0	1	1	1	1	0	0	1	0	0	0	08
	0	1	1	1	1	1	0	1	0	0	0	08

Tabla P2.3 Contenido de la memoria para el sistema de una pluma para estacionamiento.

	Dirección de memoria						Contenido de memoria					
	Estado presente		Entradas				Liga		Salidas			HEX
	QA	QB	S_E1	S_E2	S_SUP1	S_INF1	QA	QB	MOT_1	MX_1	nMX_1	
Estado 10	1	0	0	0	0	0	1	0	1	0	1	15
	1	0	0	0	0	1	0	0	1	0	1	05
	1	0	0	0	1	0	1	0	1	0	1	15
	1	0	0	0	1	1	0	0	1	0	1	05
	1	0	0	1	0	0	0	0	1	0	1	05
	1	0	0	1	0	1	0	0	1	0	1	05
	1	0	0	1	1	0	0	0	1	0	1	05
	1	0	0	1	1	1	0	0	1	0	1	05
	1	0	1	0	0	0	0	0	1	0	1	05
	1	0	1	0	0	1	0	0	1	0	1	05
	1	0	1	0	1	0	0	0	1	0	1	05
	1	0	1	0	1	1	0	0	1	0	1	05
	1	0	1	1	0	0	0	0	1	0	1	05
	1	0	1	1	0	1	0	0	1	0	1	05
	1	0	1	1	1	0	0	0	1	0	1	05
	1	0	1	1	1	1	0	0	1	0	1	05

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P2.6,figura P2.7). Se carga en el controlador el archivo con extensión “HEX” de la memoria. Se debe seguir la descripción de la carta ASM para probar la implementación de esta práctica.

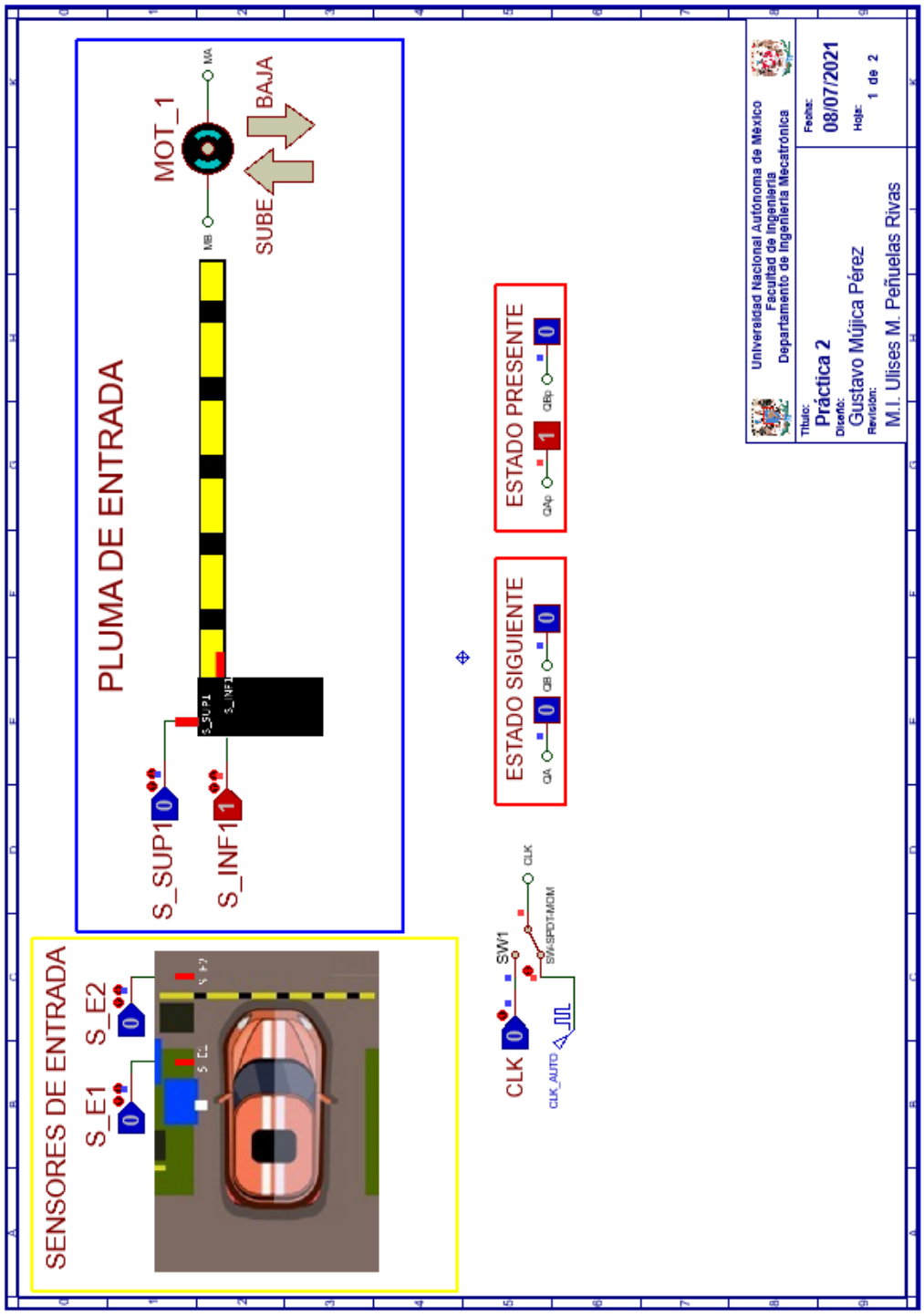


Figura P2.6 Interfaz hombre-máquina para el controlador de la Práctica 2 hoja 1/2.

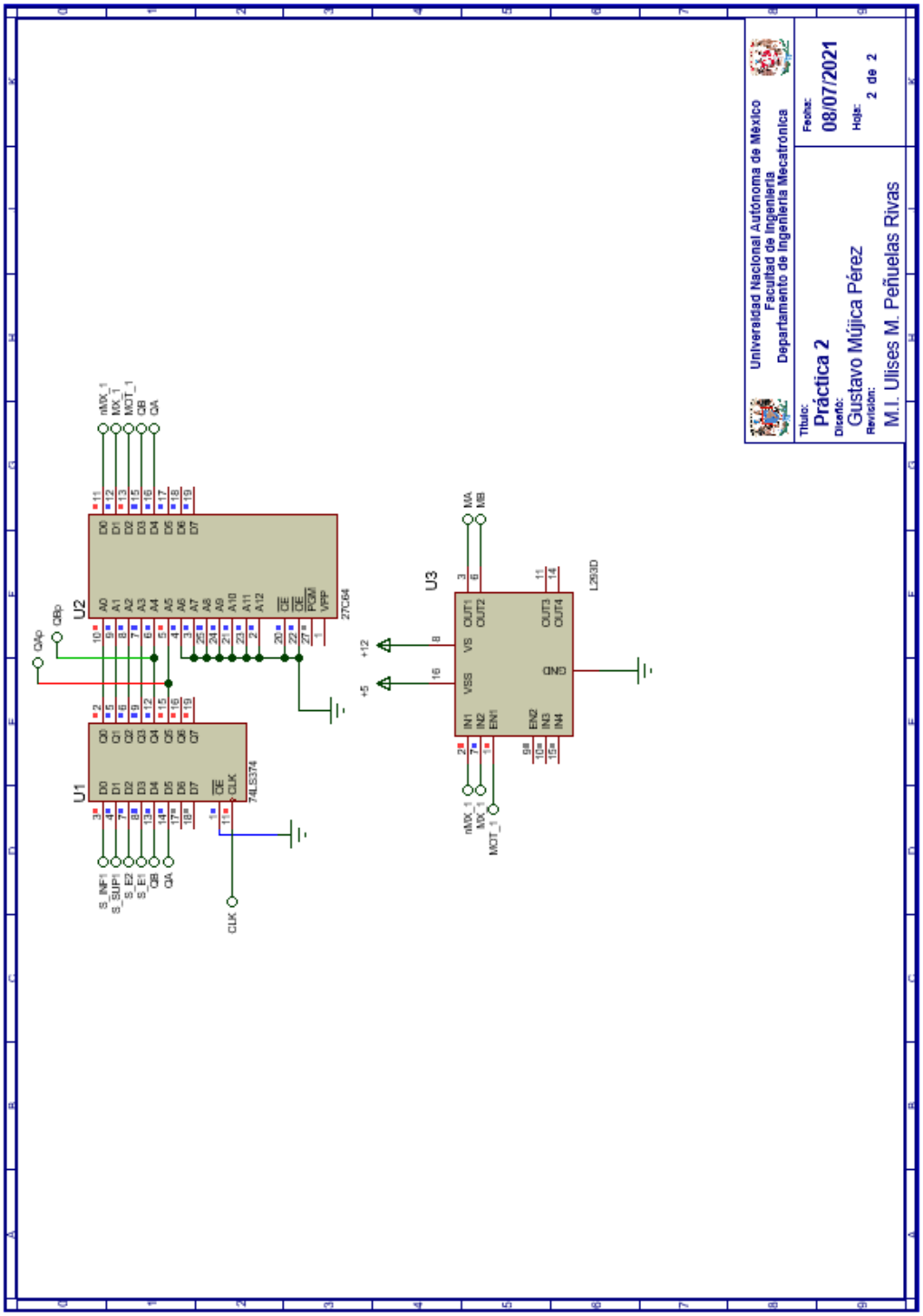


Figura P2.7 Esquema electrónico para el controlador de la Práctica 2 hoja 2/2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 3 Sistema de recolección de boletos; diseño con memoria y direccionamiento por trayectoria

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P3.1).

Figura P3.1 Maqueta de estacionamiento con 10 lugares.

A la salida se debe recolectar el boleto para poder abandonar el estacionamiento. Este sistema recolecta el boleto de forma automática al introducirlo en una ranura (ver figura P3.2).

Figura P3.2 Sección de la maqueta en donde se recolectará el boleto.

El diseño con memoria y direccionamiento por trayectoria guarda el estado siguiente de la salida de cada estado de la carta ASM en una localidad de memoria. La porción de la memoria que indica el estado siguiente es llamada “liga”, mientras que la porción que indica las salidas se llama “la parte de las salidas” [1].

La arquitectura del diseño con memoria y direccionamiento por trayectoria se muestra en la figura P3.3, donde:

A = estado siguiente

B = entradas del estado siguiente

C = entradas del estado presente

D = estado presente

E = dirección del estado siguiente

F = salidas del estado presente.

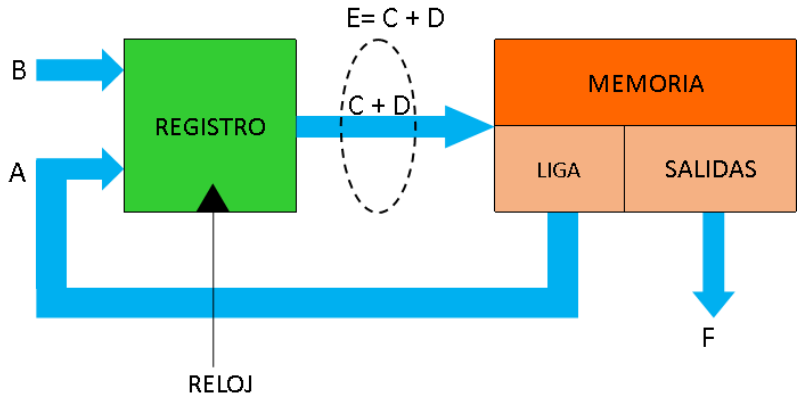


Figura P3.3 Arquitectura de un controlador con memoria y direccionamiento por trayectoria.

En este tipo de diseño con memoria y direccionamiento por trayectoria, todas las salidas de la carta ASM deberán depender del estado presente y de los valores de entrada.

Objetivo

Diseñar un controlador para el sistema de recolección de boletos, por medio del método de diseño con memoria y direccionamiento por trayectoria.

Descripción

Primero se diseña una carta ASM para el sistema de recolección de boletos por medio del método de diseño con memoria y direccionamiento por trayectoria. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P3.1 se muestran los detalles de las entradas y salidas de este controlador.

Para la recolección de boletos se necesitan señales de entrada:

- un sensor detecta la presencia de vehículo
- un sensor indica cuándo se esté ingresando un boleto y cuándo este haya ingresado totalmente (ver figura P3.2).

Como salida se requiere de las siguientes señales:

- activación del motor que recolecta el boleto.

Tabla P3.1 Entradas y salidas para el sistema de recolección de boletos.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
S_S1	D1 del registro	I	Sensor de salida que detecta cuándo un vehículo llega a la salida del estacionamiento.
S_ING	D0 del registro	I	Sensor de ingreso y recolección de boleto.
M_RBO	D0 de la memoria	O	Motor que recolectará el boleto.

Notas de diseño

a) El sensor S_S1 está ubicado antes del cruce de la pluma.

b) El sensor S_ING tiene dos funciones: detectar cuándo se está ingresando un boleto y cuándo el boleto ha sido recolectado totalmente.

Reglas de funcionamiento

- S_S1: sensor de salida

  1 = detecta vehículo

  0 = no detecta vehículo

- S_ING: sensor de ingreso

  1 = detecta ingreso de boleto o detecta boleto no recolectado completamente

  0 = no detecta ingreso de boleto o detecta boleto recolectado completamente.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado. El algoritmo de la máquina de estados se puede ver en la figura P3.4.

Estado '00' – SALIDA

En el primer estado, el controlador de la recolección de boletos se encuentra en espera. El sensor (S_S1), al detectar un vehículo, permite al sistema pasar al Estado '01'. De lo contrario, permanece en el Estado '00'.

Estado ‘01’ – INTB

En este estado, se espera a que un boleto sea introducido. Si el sensor (S_ING) detecta que se introduce un boleto, el sistema avanza al Estado ‘10’ para recolectar el boleto. De lo contrario, regresa al Estado ‘00’ para verificar si aún hay un vehículo presente.

Estado ‘10’ – RBOL

En este estado se activa el motor (M_RBO) para recolectar un boleto. Cuando el sensor (S_ING) detecta que el boleto ha sido recolectado en su totalidad, el sistema regresa al Estado ‘00’, para iniciar nuevamente el proceso de recolección del boleto. De lo contrario, permanece en el Estado ‘10’ recolectando el boleto.

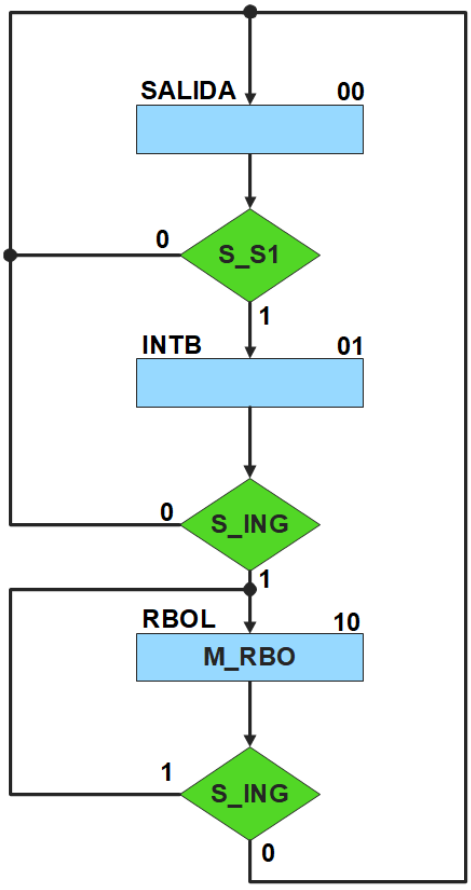


Figura P3.4 Carta ASM del sistema de recolección de boletos.

Solución

Se debe llenar la tabla P3.2 con base en la información de la carta ASM de la figura P3.4, usando el método de diseño con memoria y direccionamiento por trayectoria. Para cada estado es necesario considerar todas las posibles combinaciones de las variables de entrada aun cuando algunas de ellas no se utilicen [1].

A continuación, se describe como llenar los campos de la memoria para el Estado ‘00’.

Debido a que hay dos variables de entrada, se deben considerar 4 posibles combinaciones de éstas para cada estado. Si en el Estado ‘00’ la variable S_S1 es igual a ‘1’, el estado siguiente será el Estado ‘01’ independientemente de los valores de las otras variables. En el Estado ‘00’ el motor M_RBO no está activado, por lo que se coloca un ‘0’ en la parte de salidas de todo el estado.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P3.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P3.2 Contenido de la memoria para el sistema de recolección de boletos.

	Dirección de memoria				Contenido de memoria			
	Estado presente		Entradas		Liga		Salida	HEX
	QA	QB	S_S1	S_ING	QA	QB	M_RBO	
Estado 00	0	0	0	0	0	0	0	00
	0	0	0	1	0	0	0	00
	0	0	1	0	0	1	0	02
	0	0	1	1	0	1	0	02
Estado 01	0	1	0	0	0	0	0	00
	0	1	0	1	1	0	0	04
	0	1	1	0	0	0	0	00
	0	1	1	1	1	0	0	04
Estado 10	1	0	0	0	0	0	1	01
	1	0	0	1	1	0	1	05
	1	0	1	0	0	0	1	01
	1	0	1	1	1	0	1	05

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P3.5,figura P3.6). Se carga en el controlador el archivo con extensión “HEX” de la memoria. Se debe seguir la descripción de la carta ASM para probar la implementación de esta práctica.

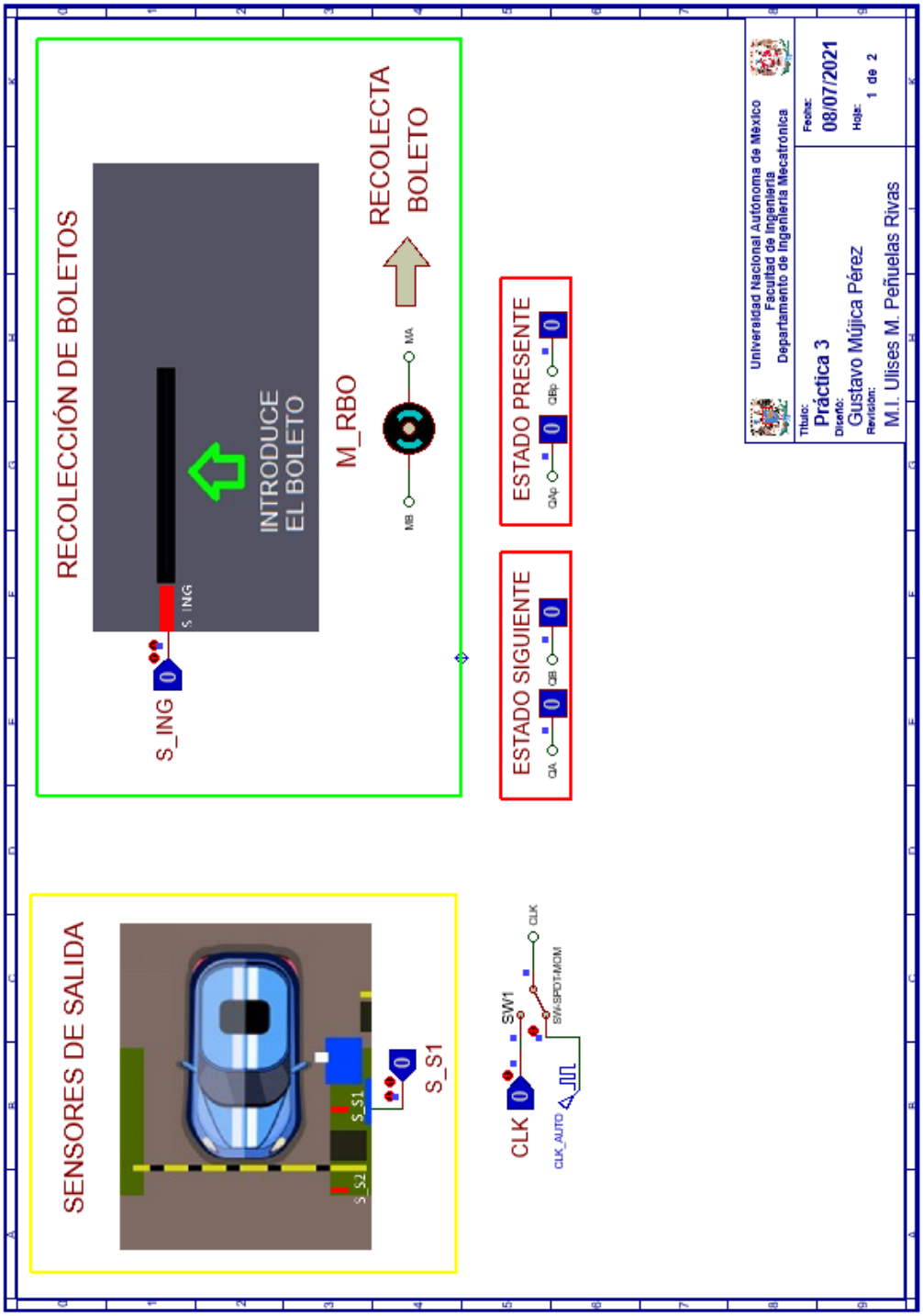


Figura P3.5 Interfaz hombre-máquina para el controlador de la Práctica 3 hoja 1/2.

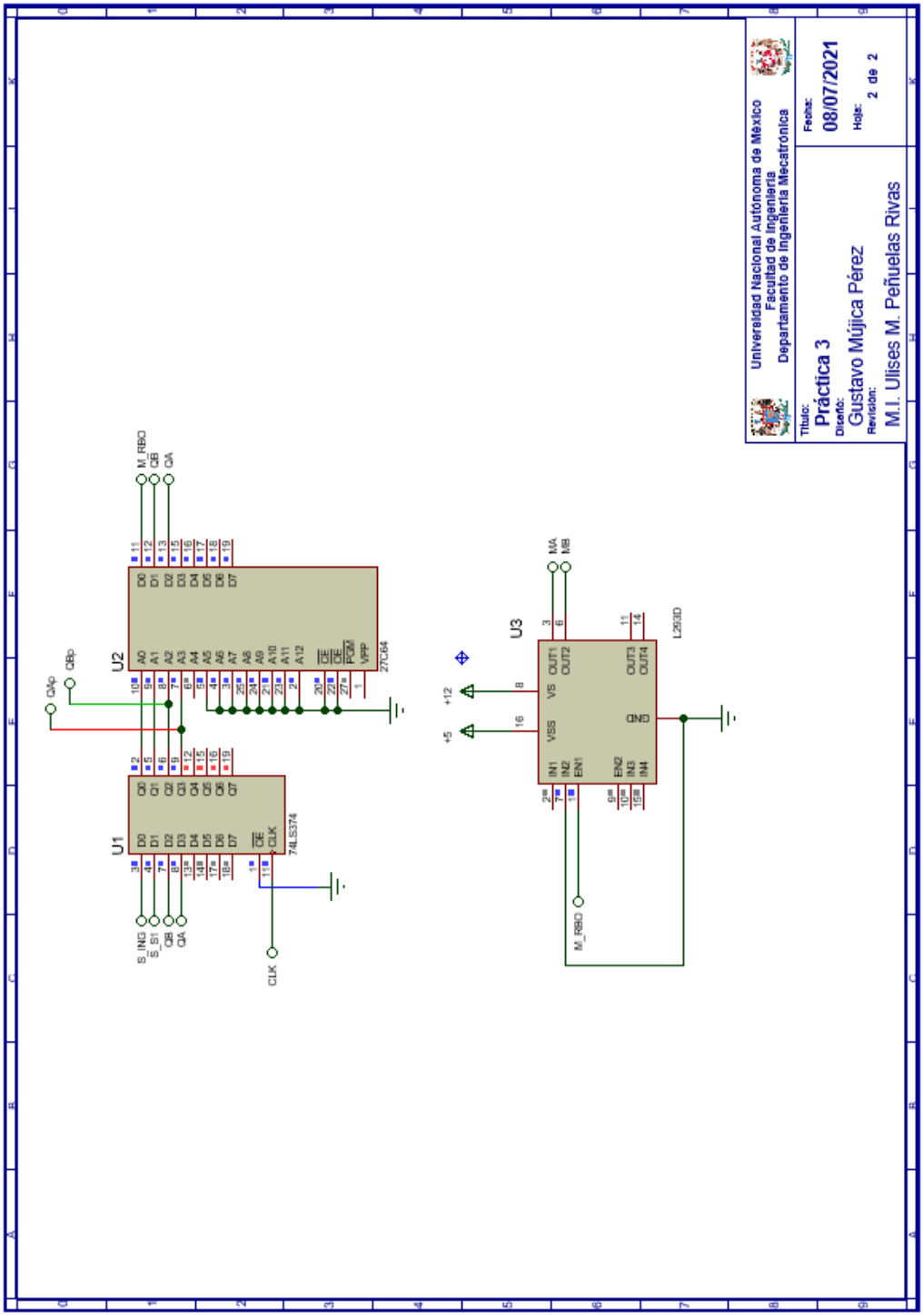


Figura P3.6 Esquema electrónico para el controlador de la Práctica 3 hoja 2/2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 4 Sistema de emisión/retiro de boletos; diseño con memoria y direccionamiento entrada-estado

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P4.1).

Figura P4.1 Maqueta de estacionamiento con 10 lugares.

En la entrada se debe adquirir un boleto para poder ingresar al estacionamiento. El sistema de emisión/retiro de boletos entrega un boleto de forma automática al presionar un botón, el boleto debe salir a través de una ranura (ver figura P4.2).

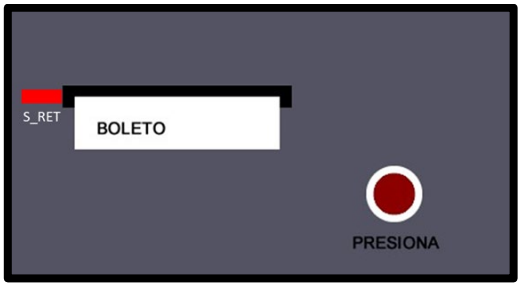


Figura P4.2 Sección de la maqueta en donde se emite el boleto.

El diseño con memoria y direccionamiento entrada-estado restringe las cartas ASM a una sola entrada por estado. Una nueva porción de la palabra de memoria contiene una representación binaria de la entrada a probar en cada estado, esta parte es llamada “la parte de prueba”. Con esta representación binaria un selector de entrada elige una de las variables de entrada (ver figura P4.3) [1].

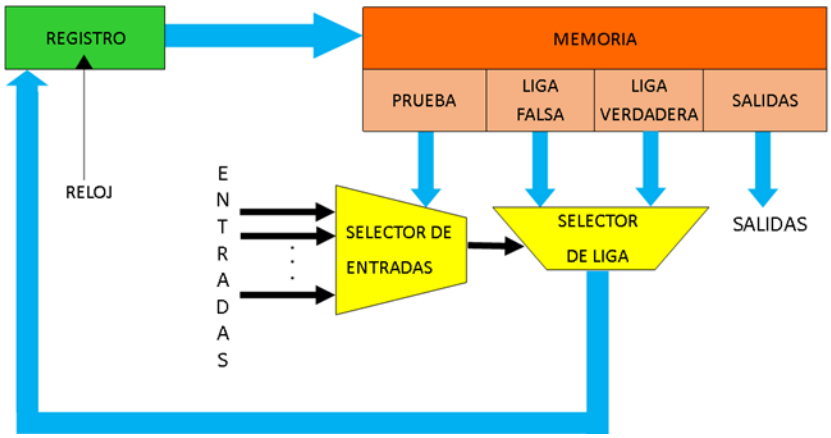


Figura P4.3 Arquitectura de un controlador con memoria y direccionamiento de entrada-estado.

La parte de liga tiene dos estados siguientes, escogiéndose uno por el selector de liga, con base en la entrada seleccionada por la parte de prueba. Si el valor de la entrada seleccionada por el selector de entradas es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera (ver figura P4.4) [1].

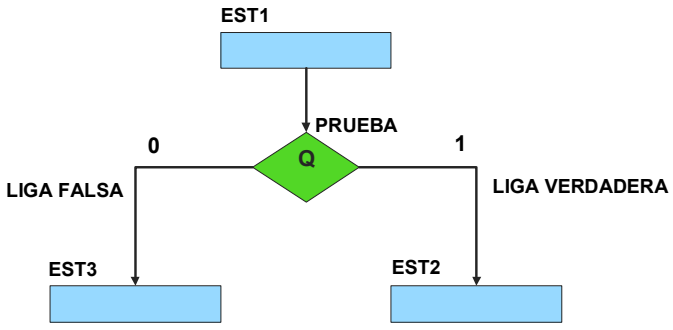


Figura P4.4 La liga falsa es el estado EST3, mientras que la liga verdadera es el estado EST2.

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará también una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista, se probará la variable auxiliar, la cual tiene un valor preestablecido de cero o uno [1].

Objetivo

Diseñar un controlador para el sistema de emisión/retiro de boletos, por medio del método de diseño con memoria y direccionamiento entrada-estado.

Descripción

Primero se diseña una carta ASM para el sistema de emisión/retiro de boletos por medio del método de diseño con memoria y direccionamiento entrada-estado. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P4.1 se muestran los detalles de las entradas y salidas de este controlador.

Para la emisión de boletos se necesita señales de entrada:

- un sensor detecta la presencia de vehículo
- un botón detecta la solicitud de emisión de boleto
- un sensor verifica la emisión y, posteriormente, el retiro del boleto (ver figura P4.2).

Como salida se requiere de las siguientes señales:

- activación del motor de emisión del boleto.

Tabla P4.1 Entradas y salidas del sistema de emisión/retiro de boletos.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
S_E1	A3 del PIC	I	Sensor de entrada que detecta cuándo un vehículo llega a la entrada del estacionamiento
BOT	A4 del PIC	I	Botón para obtener un boleto
S_RET	A5 del PIC	I	Sensor de emisión y retiro de boleto
M_EBO	D0 de la memoria	O	Motor que emitirá un boleto.

Notas de diseño

a) El sensor S_E1 está ubicado antes del cruce de la pluma

b) El sensor S_RET tiene dos funciones: detectar cuándo el boleto ha sido emitido y cuándo el boleto ha sido retirado.

Reglas de funcionamiento

- S_E1: sensor de entrada
  1 = detecta vehículo
  0 = no detecta vehículo
- BOT: botón para obtener un boleto
  1 = se presionó el botón
  0 = no se presionó el botón
- S_RET: sensor de emisión y retiro de boleto
  1 = detecta que se ha emitido el boleto o detecta que el boleto no ha sido retirado
  0 = detecta que no se ha emitido el boleto o detecta que el boleto ha sido retirado.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Si el valor de la entrada es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera. El algoritmo de la máquina de estados se puede ver en la figura P4.5.

Estado ‘00’ – ENTRADA

Entrada del vehículo. En el primer estado, el controlador de la emisión de boletos se encuentra en espera. El sensor (S_E1), al detectar un vehículo, permite al sistema pasar al Estado ‘01’. De lo contrario, permanece en el Estado ‘00’.

Estado ‘01’ – EBOT

En este estado se espera a que se presione el botón (BOT) para la emisión del boleto. Al ser presionado el botón, el sistema avanza al Estado ‘10’ para emitir el boleto. De lo contrario, regresa al Estado ‘00’ para verificar si aún hay un vehículo presente.

Estado ‘10’ – EBOL

Emisión de boleto. En este estado se activa el motor correspondiente (M_EBO) para que se emita un boleto. Cuando el sensor (S_RET) detecta que el boleto ya se encuentra listo para retirarse, el sistema pasa al Estado ‘11’ donde se desactiva al motor y se espera a que se retire el boleto. Si el sensor aún no detecta el boleto, permanece en el Estado ‘10’.

Estado ‘11’ – ERET

En este estado se espera a que el boleto sea retirado. Cuando el sensor (S_RET) detecta que el boleto se ha retirado, regresa al Estado ‘00’, para iniciar nuevamente el proceso de emisión/retiro de boleto. De lo contrario, permanece en el Estado ‘11’, en espera a que el boleto sea retirado.

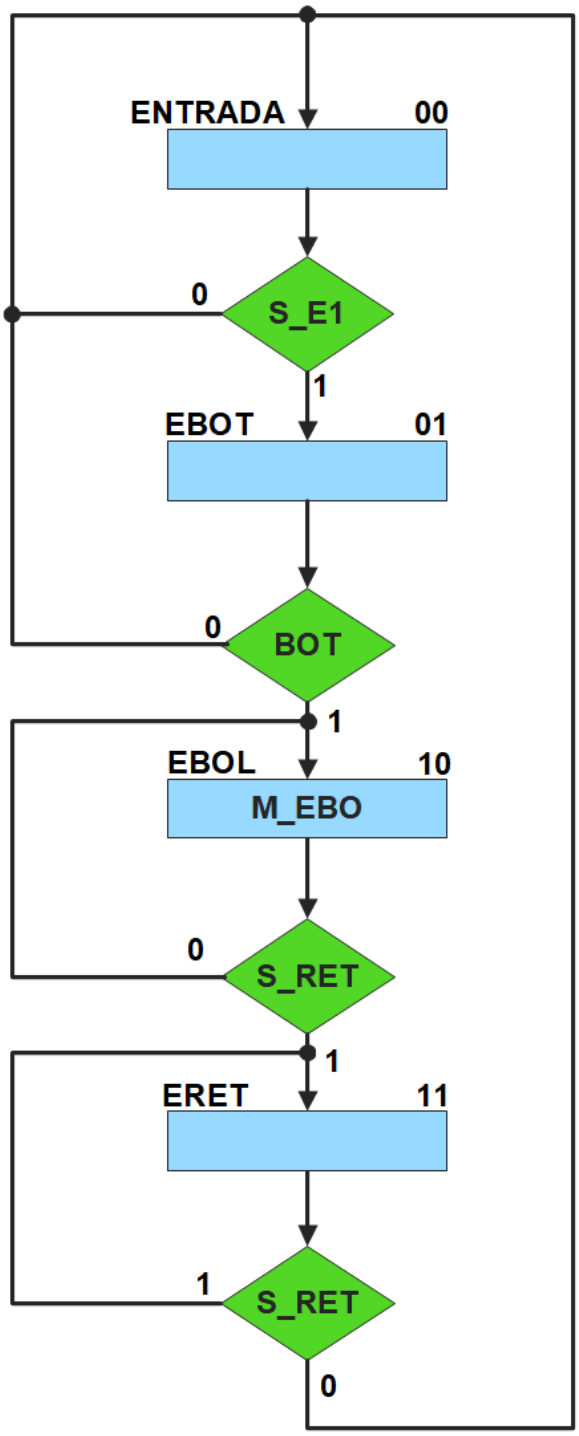


Figura P4.5 Carta ASM del sistema de emisión/retiro de boletos.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P4.2).

Tabla P4.2 Representación binaria de entradas del sistema de emisión/retiro de boletos.

Entrada	Prueba
S_E1	0 0
BOT	0 1
S_RET	1 0

Se debe llenar la tabla P4.3 con base en la información de la carta ASM de la figura P4.5, usando el método de diseño con memoria y direccionamiento entrada-estado.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘00’.

En el Estado ‘00’ se selecciona la entrada S_E1, por lo tanto, se coloca en el campo de prueba de la memoria su representación binaria, es decir, ‘00’. Si S_E1 es igual a cero, el estado siguiente es el Estado ‘00’, su representación binaria ‘00’ es colocada en el campo de la liga falsa. Si S_E1 es igual a uno, el estado siguiente es el Estado ‘01’, su representación binaria ‘01’ es colocada en el campo de la liga verdadera. En el Estado ‘00’ el motor M_EBO no está activado, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P4.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P4.3 Contenido de la memoria para el sistema de emisión/retiro de boletos.

Dirección de memoria		Contenido de memoria							
Estado presente		Prueba		Liga Falsa		Liga Verdadera		Salida	HEX
QA	QB	I1	I0	LF1	LF0	LV1	LV0	M_EBO	
0	0	0	0	0	0	0	1	0	02
0	1	0	1	0	0	1	0	0	24
1	0	1	0	1	0	1	1	1	57
1	1	1	0	0	0	1	1	0	46

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de cuatro líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de éste (ver tabla P4.4). Se puede comprobar realizando el circuito de la figura P4.6.

Tabla P4.4 Tabla de funcionamiento del selector de entradas del sistema de emisión/retiro de boletos.

Prueba		SI
I1	I0	
0	0	S_E1
0	1	BOT
1	0	S_RET
1	1	*

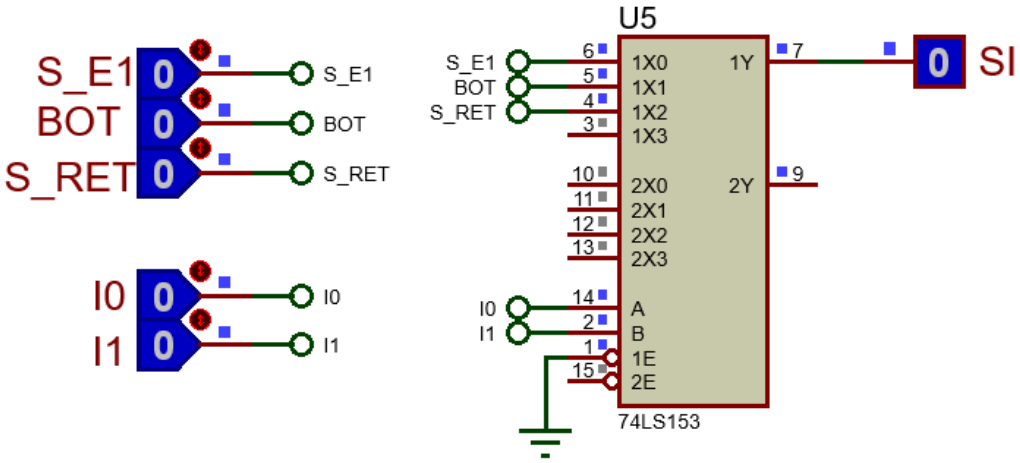


Figura P4.6 Multiplexor 74LS133 para el selector de entradas del sistema de emisión/retiro de boletos.

La función booleana del selector de entradas queda:

```
SI = S_E1&!I1&!I0 | BOT&!I1&I0 | S_RET&I1&!I0;
```

El selector de liga es un multiplexor doble de dos líneas a una, éste es implementado por el PIC16F1939. Si el selector entradas es igual a ‘1’, se selecciona la información binaria de la liga verdadera. Si el selector entradas es igual a ‘0’, se selecciona la información binaria de la liga falsa. Se puede comprobar realizando el circuito de la figura P4.7.

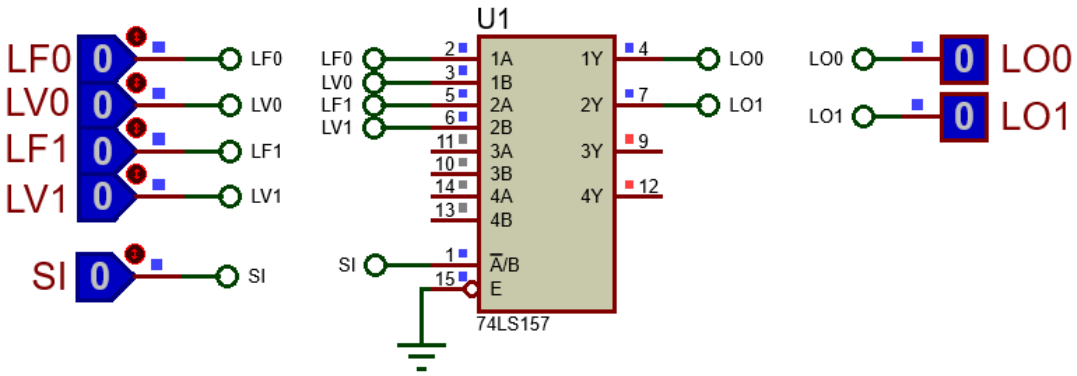


Figura P4.7 Multiplexor 74LS157 para el selector de liga del sistema de emisión/retiro de boletos.

Las funciones booleanas del selector de liga quedan:

```
LO0 = !SI&LF0 | SI&LV0;

LO1 = !SI&LF1 | SI&LV1;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P4.8,figura P4.9). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939.

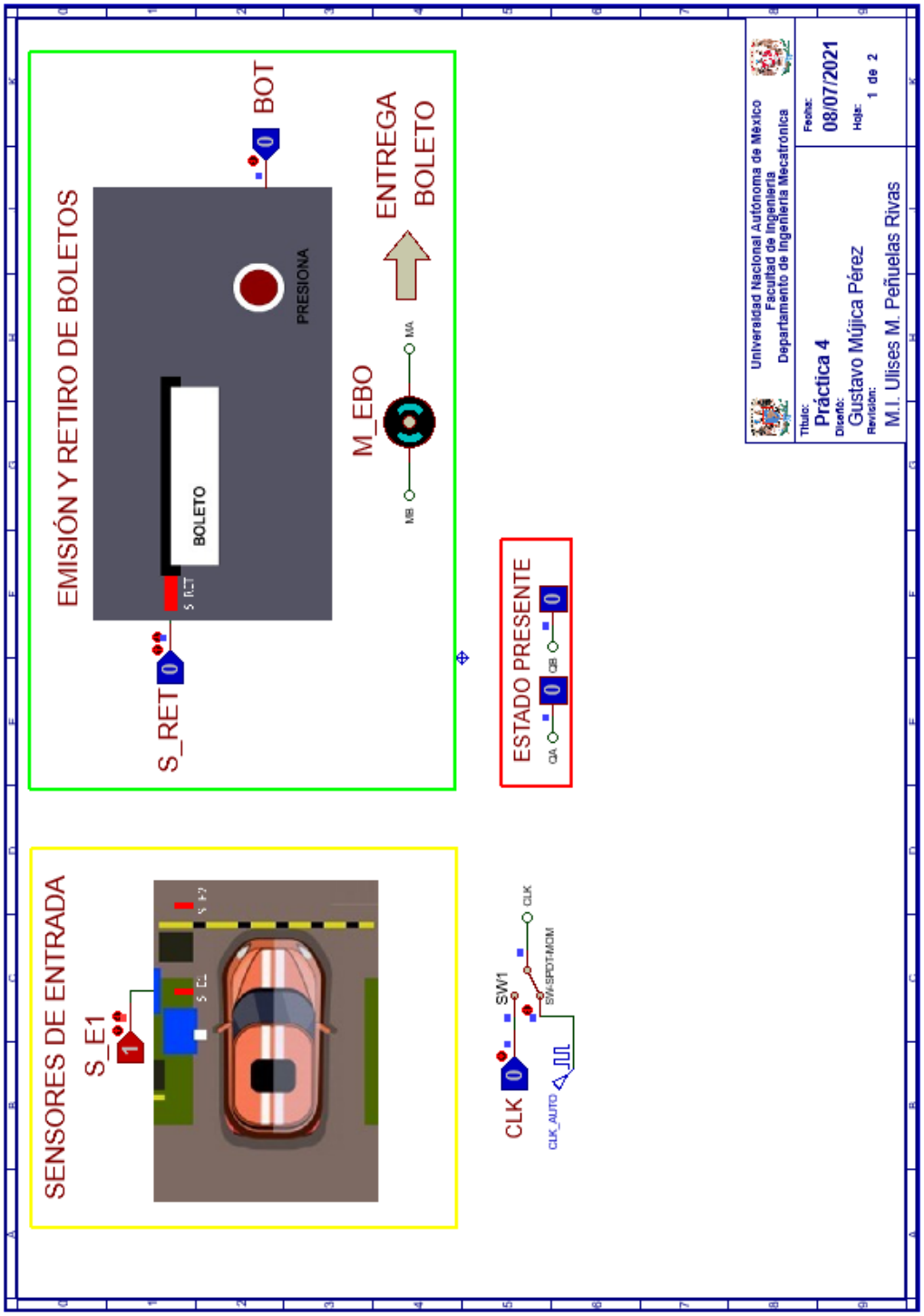


Figura P4.8 Interfaz hombre-máquina para el controlador de la Práctica 4 hoja 1/2.

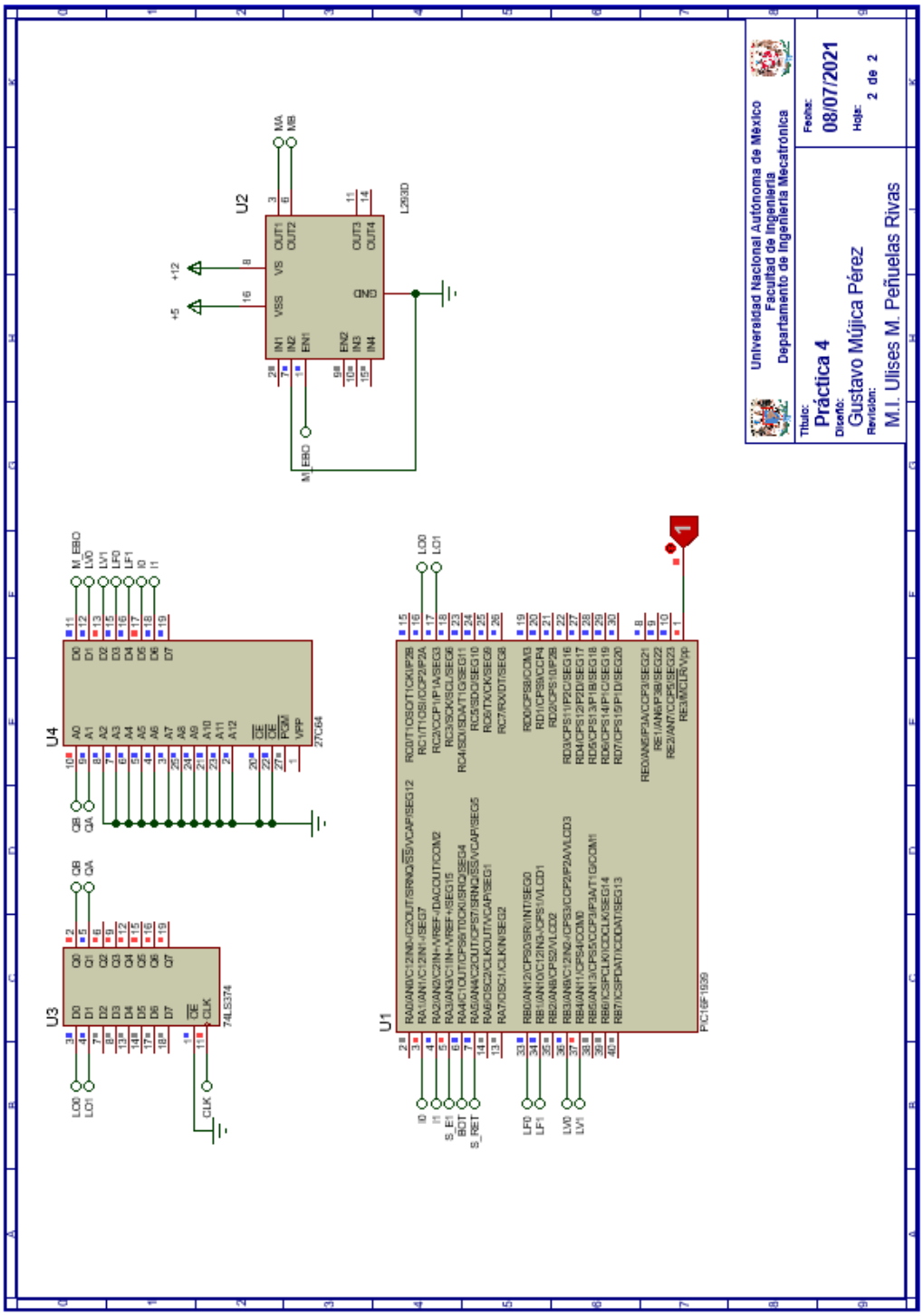


Figura P4.9 Esquema electrónico para el controlador de la Práctica 4 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P4.10) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     // Carga biblioteca PLD.h

//***ENTRADAS***

#define I0 A1 //PRUEBA
#define I1 A2 //PRUEBA

#define S_E1  A3 //ENTRADA
#define BOT   A4 //ENTRADA
#define S_RET A5 //ENTRADA


//LIGAS FALSAS
#define LF0 B0
#define LF1 B1

//LIGAS VERDADERAS
#define LV0 B3
#define LV1 B4

//***SALIDAS***

#define LO0 C1
#define LO1 C2

//***VARIABLES INTERMEDIAS***

short SI=0; //SELECTOR DE ENTRADA

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD


//LOOP INFINITO
   while(1)
   {

   //SELECTOR DE ENTRADAS
   SI= S_E1&!I1&!I0 | BOT&!I1&I0 | S_RET&I1&!I0 ;

   //SELECTOR DE LIGA
   LO0= !SI&LF0 | SI&LV0;
   LO1= !SI&LF1 | SI&LV1;

   }

}
```

Figura P4.10 Código para el controlador de la Práctica 4.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 5 Sistema de pluma para estacionamiento; diseño con memoria y direccionamiento entrada-estado

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P5.1).

Figura P5.1 Maqueta de estacionamiento con 10 lugares.

Se usan plumas que impiden el avance del vehículo a la entrada y salida del estacionamiento. Se levanta la pluma para permitir el paso de un vehículo y cuando éste haya pasado completamente, se baja la pluma (ver figura P5.2 y figura P5.3).

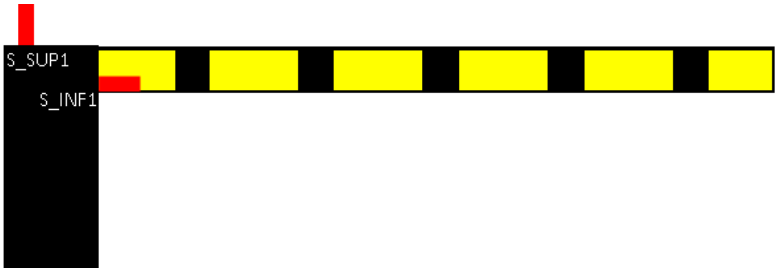


Figura P5.2 Sensores de posición superior e inferior en la pluma.

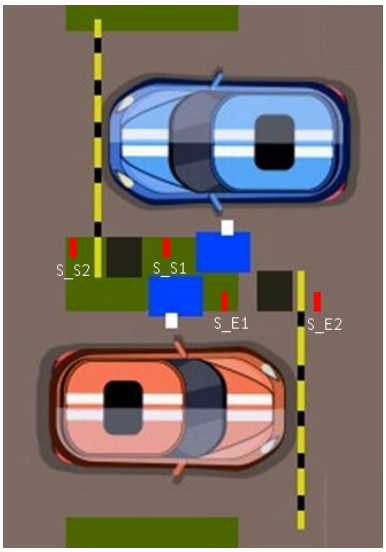


Figura P5.3 Sensores a la entrada y salida del estacionamiento.

El diseño con memoria y direccionamiento entrada-estado restringe las cartas ASM a una sola entrada por estado. Una nueva porción de la palabra de memoria contiene una representación binaria de la entrada a probar en cada estado, esta parte es llamada “la parte de prueba”. Con esta representación binaria un selector de entrada elige una de las variables de entrada (ver figura P5.4) [1].

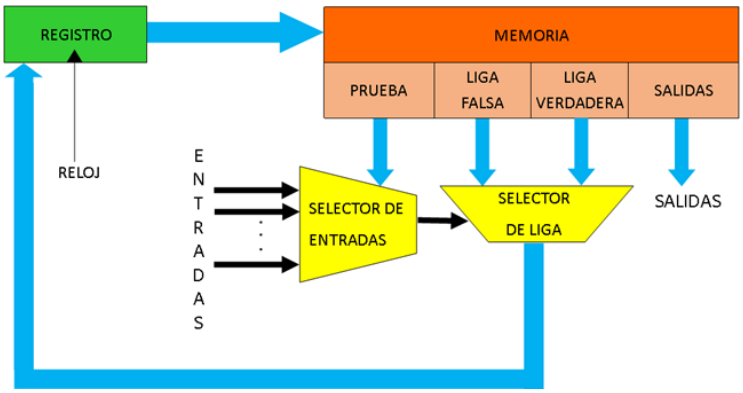


Figura P5.4 Arquitectura de un controlador con memoria y direccionamiento entrada-estado.

La parte de liga tiene dos estados siguientes, escogiéndose uno por el selector de liga, con base en la entrada seleccionada por la parte de prueba. Si el valor de la entrada seleccionada por el selector de entradas es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera (ver figura P5.5) [1].

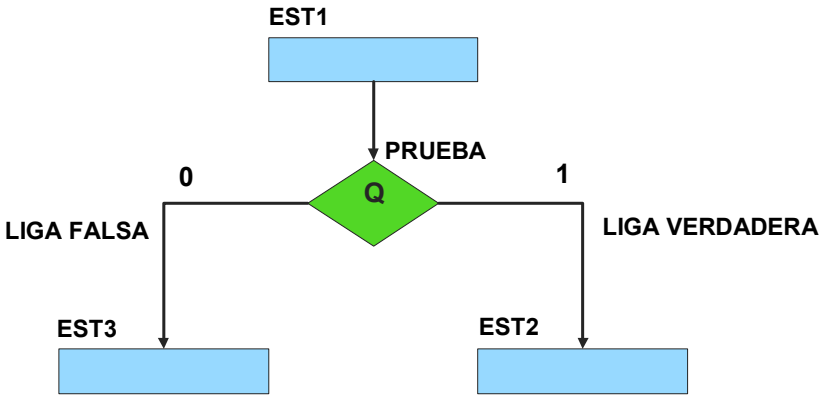


Figura P5.5 La liga falsa es el estado EST3, mientras que la liga verdadera es el estado EST2.

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará también una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista, se probará la variable auxiliar, la cual tiene un valor preestablecido de cero o uno [1].

Objetivo

Diseñar un controlador para el sistema de pluma para estacionamiento, por medio del método de diseño con memoria y direccionamiento entrada-estado.

Descripción

Primero se diseña una carta ASM para el sistema pluma para estacionamiento por medio del método de diseño con memoria y direccionamiento entrada-estado. Posteriormente se propone una solución para implementar el sistema.

Nota: Para este diseño se usarán los sensores de la pluma de entrada, sin embargo, el proceso es el mismo para la pluma de entrada como para la pluma de salida.

Tabla de entradas y salidas

En la tabla P5.1 se muestran los detalles de las entradas y salidas de este controlador.

Para la pluma del estacionamiento se necesitan señales de entrada:

- un sensor detecta cuándo la pluma se encuentra en la posición superior y otro cuando se encuentra en la posición inferior (ver figura P5.2)
- un sensor detecta cuándo un vehículo está a la entrada del estacionamiento y otro cuando se esté ingresando a éste (ver figura P5.3).

Como salida se requiere de las siguientes señales:

- una señal activa el motor de la pluma
- una señal se activa para que la pluma suba y otra para que la pluma baje.

Tabla P5.1 Entradas y salidas para el sistema de pluma para estacionamiento.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
S_E1	A0 del PIC	I	Sensor de entrada que detecta cuándo un vehículo llega a la entrada del estacionamiento
S_E2	A1 del PIC	I	Sensor de ingreso que detecta cuándo un vehículo está ingresando al estacionamiento
S_SUP1	A2 del PIC	I	Sensor de posición superior que detecta cuándo la pluma se ha elevado en su totalidad
S_INF1	A3 del PIC	I	Sensor de posición inferior detecta cuándo la pluma se ha bajado en su totalidad
MOT_1	D2 de la memoria	O	Motor de la pluma
MX_1	D1 de la memoria	O	Señal para que la pluma suba
nMX_1	D0 de la memoria	O	Señal para que la pluma baje.

Notas de diseño

a) El sensor S_E1 está ubicado antes del cruce de la pluma

b) El sensor S_E2 está ubicado después del cruce de la pluma.

Reglas de funcionamiento

- S_E1: sensor de entrada
  1 = detecta vehículo
  0 = no detecta vehículo

- S_E2: sensor de ingreso
  1 = detecta vehículo ingresando
  0 = no detecta vehículo ingresando
- S_SUP1: sensor de posición superior
  1 = detecta pluma completamente elevada
  0 = no detecta pluma completamente elevada
- S_INF1: sensor de posición inferior
  1 = detecta que la pluma ha bajado completamente
  0 = no detecta que la pluma ha bajado completamente.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Si el valor de la entrada es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera. El algoritmo de la máquina de estados se puede ver en la figura P5.6.

Estado ‘000’ – ELEVA

En el primer estado, se activa el motor (MOT_1) y la señal (MX_1) para que la pluma suba, permitiendo el ingreso. Si el sensor (S_SUP1) detecta que la pluma ha sido elevada completamente, el sistema avanza al Estado ‘001’ para esperar a que pase el vehículo. De lo contrario avanza al Estado ‘011’ para verificar si hay un vehículo a la entrada.

Estado ‘001’ – ESPERA

En este estado, la pluma esta elevada, esperando a que el vehículo pase completamente. Si el sensor (S_E1) no detecta un vehículo a la entrada, el sistema avanza al Estado ‘010’ para revisar si un vehículo está ingresando. De lo contrario, permanece en el estado actual.

Estado ‘010’ – AUX1

Si el sensor (S_E2) no detecta un vehículo ingresando, avanza al Estado ‘101’ para que se baje la pluma ya que el vehículo ha pasado completamente. De lo contrario, el sistema regresa al Estado ‘001’ para esperar a que el vehículo pase completamente.

Estado ‘011’ – AUX2

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (MX_1) para que la pluma suba. Cuando el sensor (S_E1) no detecta vehículo a la entrada, el sistema avanza al Estado ‘100’ para revisar si hay un vehículo ingresando. De lo contrario, regresa al Estado ‘000’ para seguir elevando la pluma.

Estado ‘100’ – AUX3

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (MX_1) para que la pluma suba. Si el sensor (S_E2) no detecta un vehículo ingresando, el sistema avanza al Estado ‘101’ para bajar la pluma, debido a que el auto ha pasado antes de que la pluma se eleve completamente. De lo contrario, regresa al Estado ‘000’ para seguir elevando la pluma.

Estado ‘101’ – BAJA

En este estado se activa el motor (MOT_1) y la señal (nMX_1) para que la pluma baje. Si el sensor (S_INF1) detecta que la pluma no ha bajado en su totalidad, avanza al Estado ‘110’ para revisar si hay un vehículo a la entrada. De lo contrario, regresa al Estado ‘000’ para elevar la pluma nuevamente.

Estado ‘110’ – AUX4

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (nMX_1) para que la pluma siga bajando. Si el sensor (S_E1) no detecta un vehículo a la entrada, el sistema avanza al Estado ‘111’ para revisar si hay un auto ingresando. De lo contrario, regresa al Estado ’000’ para subir la pluma y que ésta no dañe al vehículo.

Estado ‘111’ – AUX5

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (nMX_1) para que la pluma siga bajando. Si el sensor (S_E2) no detecta un vehículo, el sistema regresa al Estado ‘101’ para seguir bajando la pluma. De lo contrario, regresa al Estado ‘000’ para subir la pluma y que ésta no dañe al vehículo.

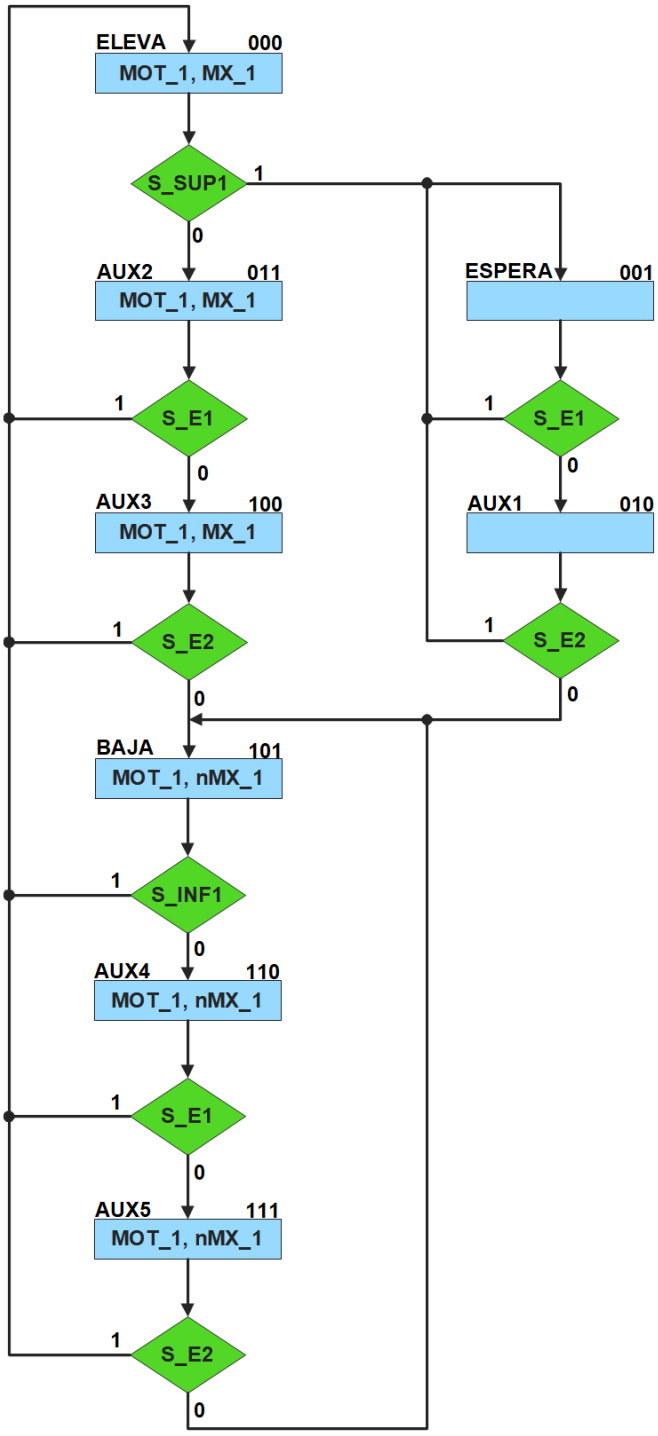


Figura P5.6 Carta ASM del sistema de pluma para estacionamiento.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P5.2).

Tabla P5.2 Representación binaria de entradas del sistema de pluma para estacionamiento.

Entrada	Prueba
S_E1	0 0
S_E2	0 1
S_SUP1	1 0
S_INF1	1 1

Se debe llenar la tabla P5.3 con base en la información de la carta ASM de la figura P5.6, usando el método de diseño con memoria y direccionamiento entrada-estado.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘000’.

En el Estado ‘000’ se selección la entrada S_SUP1, por lo tanto, se coloca en el campo de prueba de la memoria su representación binaria, es decir, ‘00’. Si S_E1 es igual a cero, el estado siguiente es el Estado ‘011’, su representación binaria ‘011’ es colocada en el campo de la liga falsa. Si S_E1 es igual a uno, el estado siguiente es el Estado ‘001’, su representación binaria ‘001’ es colocada en el campo de la liga verdadera. En el Estado ‘000’ las señales de salida del motor MOT_1 y la señal MX_1 para que la pluma suba están activadas, por lo que se coloca un ‘1’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P5.4). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P5.3 Contenido de la memoria del sistema de pluma para estacionamiento.

Dirección de memoria			Contenido de memoria												
Estado presente			Prueba		Liga Falsa			Liga Verdadera			Salidas			HEX 1	HEX 2
QA	QB	QC	I1	I0	LF2	LF1	LF0	LV2	LV1	LV0	MOT_1	MX_1	!M_X1		
0	0	0	1	0	0	1	1	0	0	1	1	1	0	99	06
0	0	1	0	0	0	1	0	0	0	1	0	0	0	11	00
0	1	0	0	1	1	0	1	0	0	1	0	0	0	69	00
0	1	1	0	0	1	0	0	0	0	0	1	1	0	20	06
1	0	0	0	1	1	0	1	0	0	0	1	1	0	68	06
1	0	1	1	1	1	1	0	0	0	0	1	0	1	F0	05
1	1	0	0	0	1	1	1	0	0	0	1	0	1	38	05
1	1	1	0	1	1	0	1	0	0	0	1	0	1	68	05

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de cuatro líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P5.4). Se puede comprobar realizando el circuito de la figura P5.7.

Tabla P5.4 Tabla de funcionamiento del selector de entradas del sistema de pluma para estacionamiento.

Prueba		SI
I1	I0	
0	0	S_E1
0	1	BOT
1	0	S_RET
1	1	S_INF1

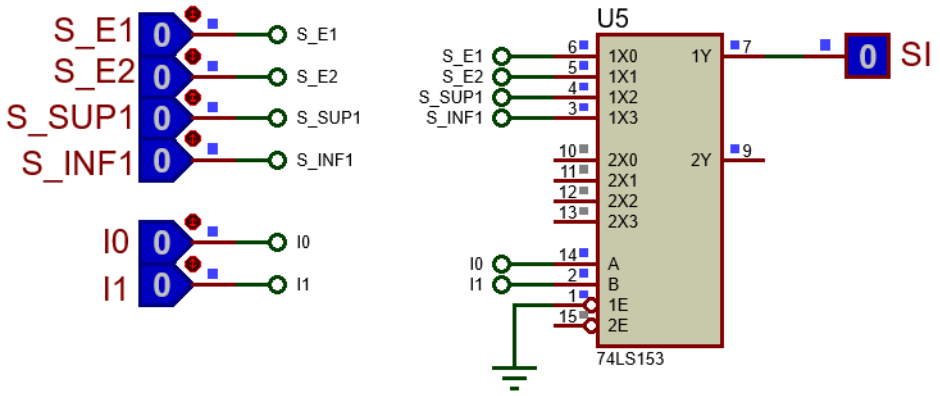


Figura P5.7 Multiplexor 74LS153 para el selector de entradas del sistema de pluma para estacionamiento.

La función booleana del selector de entradas queda:

```
SI = S_E1&!I1&!I0 | S_E2&!I1&I0 | S_SUP1&I1&!I0 | S_INF1&I1&I0;
```

El selector de liga es un multiplexor triple de dos líneas a una, éste es implementado por el PIC16F1939. Si el selector entradas es igual a ‘1’, se selecciona la información binaria de la liga verdadera. Si el selector entradas es igual a ‘0’, se selecciona la información binaria de la liga falsa. Se puede comprobar realizando el circuito de la figura P5.8.

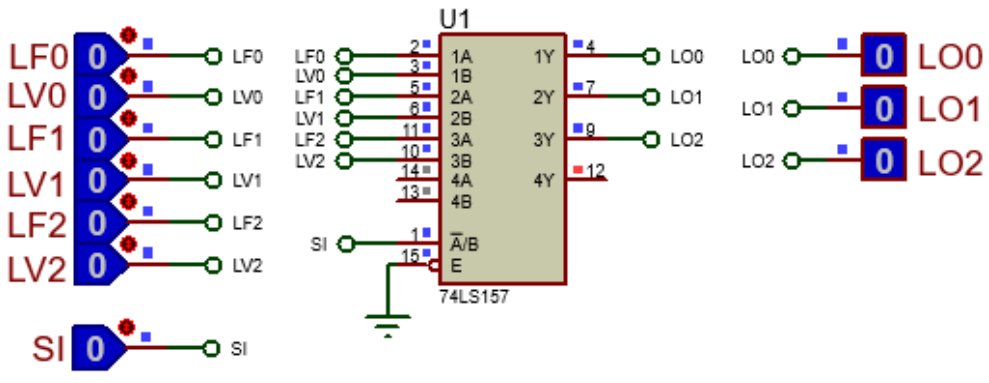


Figura P5.8 Multiplexor 74LS157 para el selector de liga del sistema de pluma para estacionamiento.

Las funciones booleanas del selector de liga quedan:

```
LO0 = !SI&LF0 | SI&LV0;

LO1 = !SI&LF1 | SI&LV1;

LO2 = !SI&LF2 | SI&LV2;
```

Diagrama de conexiones

Se verídica el funcionamiento del controlador por medio de un simulador (ver figura P5.9, figura P5.10). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939.

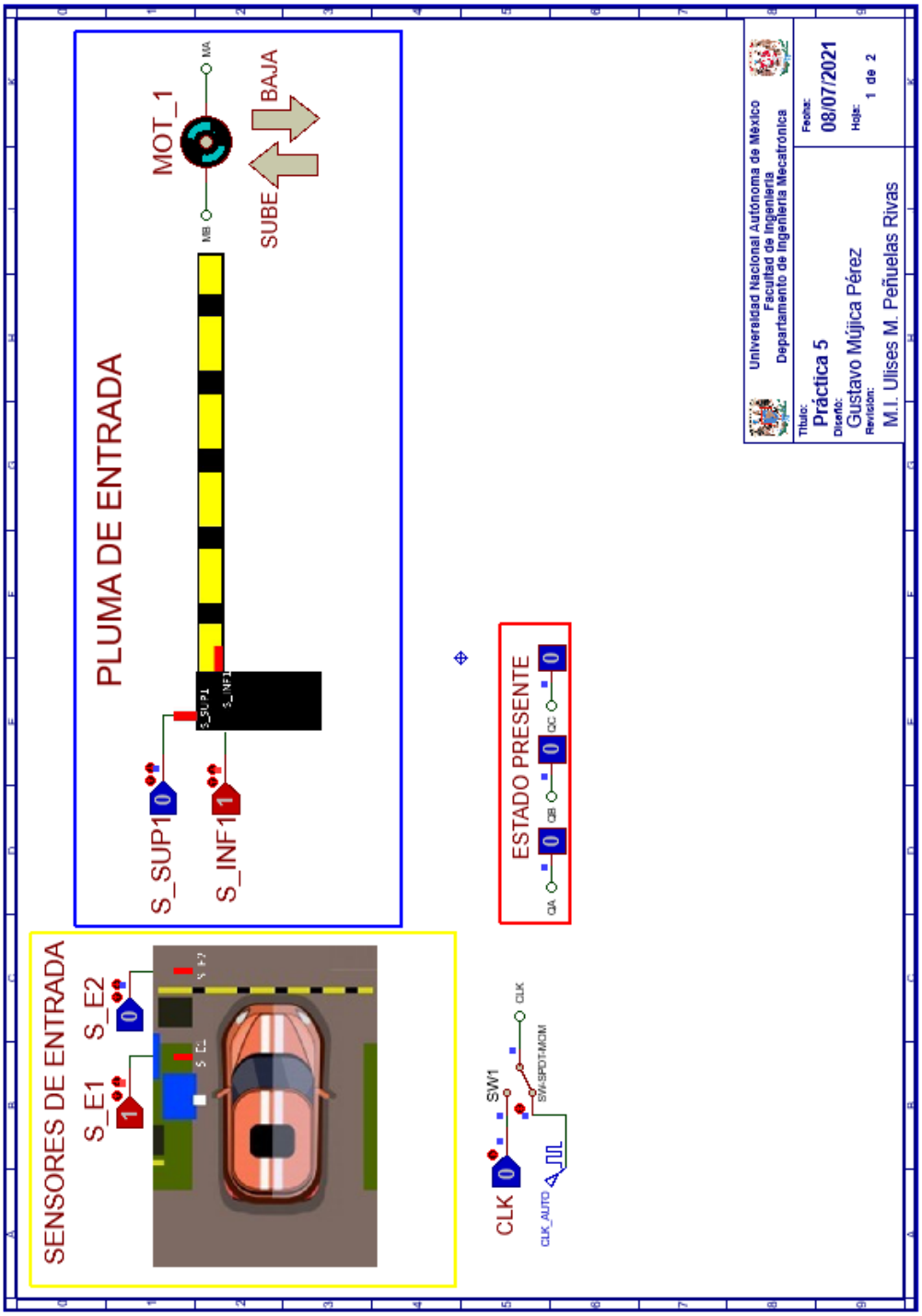


Figura P5.9 Interfaz hombre-máquina para el controlador de la Práctica 5 hoja 1/2.

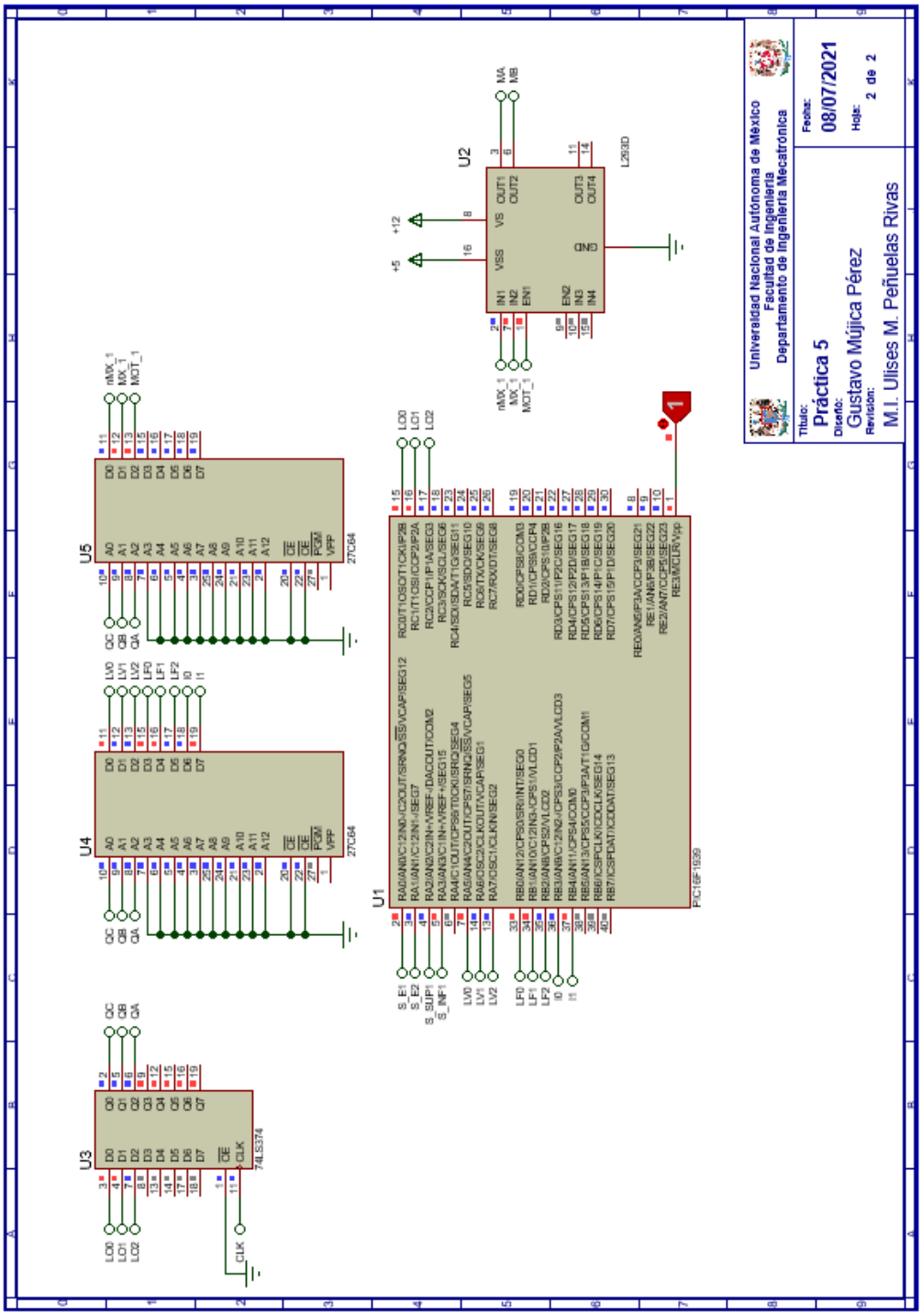


Figura P5.10 Esquema electrónico para el controlador de la Práctica 5 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P5.11) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     //Carga biblioteca PLD.h

//***ENTRADAS***

#define S_E1   A0 //ENTRADA
#define S_E2   A1 //ENTRADA
#define S_SUP1 A2 //ENTRADA
#define S_INF1 A3 //ENTRADA

//LIGAS VERDADERAS
#define LV0 A5
#define LV1 A6
#define LV2 A7

//LIGAS FALSAS
#define LF0 B0
#define LF1 B1
#define LF2 B2

//PRUEBAS
#define I0 B3 //PRUEBA
#define I1 B4 //PRUEBA

//***SALIDAS***

#define LO0 C0
#define LO1 C1
#define LO2 C2

//***VARIABLES INTERMEDIAS***

short SI; //SELECTOR DE ENTRADA

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD


//LOOP INFINITO
   while(1)
   {

   //SELECTOR DE ENTRADAS
   SI= S_E1&!I1&!I0 | S_E2&!I1&I0 | S_SUP1&I1&!I0 | S_INF1&I1&I0;

   //SELECTOR DE LIGA
   LO0= !SI&LF0 | SI&LV0;
   LO1= !SI&LF1 | SI&LV1;
   LO2= !SI&LF2 | SI&LV2;
   }

}
```

Figura P5.11 Código para el controlador de la Práctica 5.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 6 Sistema de recolección de boletos; diseño con memoria y direccionamiento entrada-estado

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P6.1).

Figura P6.1 Maqueta de estacionamiento con 10 lugares.

A la salida se debe recolectar el boleto para poder abandonar el estacionamiento. Este sistema recolecta el boleto de forma automática al introducirlo en una ranura (ver figura P6.2).

Figura P6.2 Sección de la maqueta en donde se recolectará el boleto.

El diseño con memoria y direccionamiento entrada-estado restringe las cartas ASM a una sola entrada por estado. Una nueva porción de la palabra de memoria contiene una representación binaria de la entrada a probar en cada estado, esta parte es llamada “la parte de prueba”. Con esta representación binaria un selector de entrada elige una de las variables de entrada (ver figura P6.3) [1].

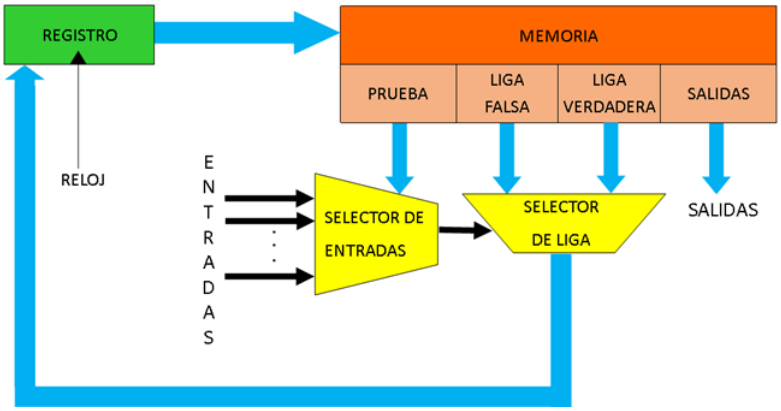


Figura P6.3 Arquitectura de un controlador con memoria y direccionamiento entrada-estado.

La parte de liga tiene dos estados siguientes, escogiéndose uno por el selector de liga, con base en la entrada seleccionada por la parte de prueba. Si el valor de la entrada seleccionada por el selector de entradas es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera (ver figura P6.4) [1].

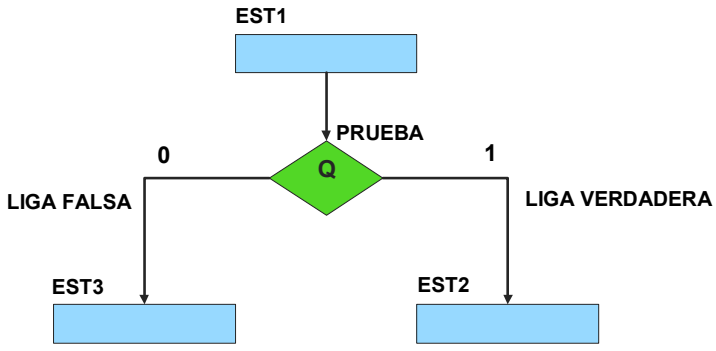


Figura P6.4 La liga falsa es el estado EST3, mientras que la liga verdadera es el estado EST2.

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará también una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista, se probará la variable auxiliar, la cual tiene un valor preestablecido de cero o uno [1].

Objetivo

Diseñar un controlador para el sistema de recolección de boletos por medio del método de diseño con memoria y direccionamiento entrada-estado.

Descripción

Primero se diseña una carta ASM para el sistema de recolección de boletos por medio del método de diseño con memoria y direccionamiento entrada-estado. Posteriormente se propondrá una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P6.1 se muestran los detalles de las entradas y salidas de este controlador.

Para la recolección de boletos se necesita señales de entrada:

- un sensor detecta la presencia de vehículo
- un sensor indica cuando se esté ingresando un boleto y cuando este haya ingresado totalmente (ver figura P6.2).

Como salida se requiere de las siguientes señales:

- Activación del motor que recolecta el boleto.

Tabla P6.1 Entradas y salidas para el sistema de recolección de boletos.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
S_S1	A0 del PIC	I	Sensor de salida que detecta cuándo un vehículo llega a la salida del estacionamiento.
S_ING	A1 del PIC	I	Sensor de ingreso y recolección de boleto.
M_RBO	D0 de la memoria	O	Motor que recolectará el boleto.

Notas de diseño

a) El sensor S_S1 está ubicado antes del cruce de la pluma

b) El sensor S_ING tiene dos funciones: detectar cuándo se está ingresando un boleto y cuándo el boleto ha sido recolectado totalmente.

Reglas de funcionamiento

- S_S1: sensor de salida

  1 = detecta vehículo

  0 = no detecta vehículo

- S_ING: sensor de ingreso

  1 = detecta ingreso de boleto o detecta boleto no recolectado completamente

  0 = no detecta ingreso de boleto o detecta boleto recolectado completamente.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Si el valor de la entrada es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera. El algoritmo de la máquina de estados se puede ver en la figura P6.5.

Estado ‘00’ – SALIDA

En el primer estado, el controlador de la recolección de boletos se encuentra en espera. El sensor (S_S1), al detectar un vehículo, permite al sistema pasar al Estado ‘01’. De lo contrario, permanece en el Estado ‘00’.

Estado ‘01’ – INTB

En este estado, se espera a que un boleto sea introducido. Si el sensor (S_ING) detecta que se introduce un boleto, el sistema avanza al Estado ‘10’ para recolectar el boleto. De lo contrario, regresa al Estado ‘00’ para verificar si aún hay un vehículo presente.

Estado ‘10’ – RBOL

En este estado se activa el motor (M_RBO) para recolectar un boleto. Cuando el sensor (S_ING) detecta que el boleto ha sido recolectado en su totalidad, el sistema regresa al Estado ‘00’, para iniciar nuevamente el proceso de recolección del boleto. De lo contrario, permanece en el Estado ‘10’ recolectando el boleto.

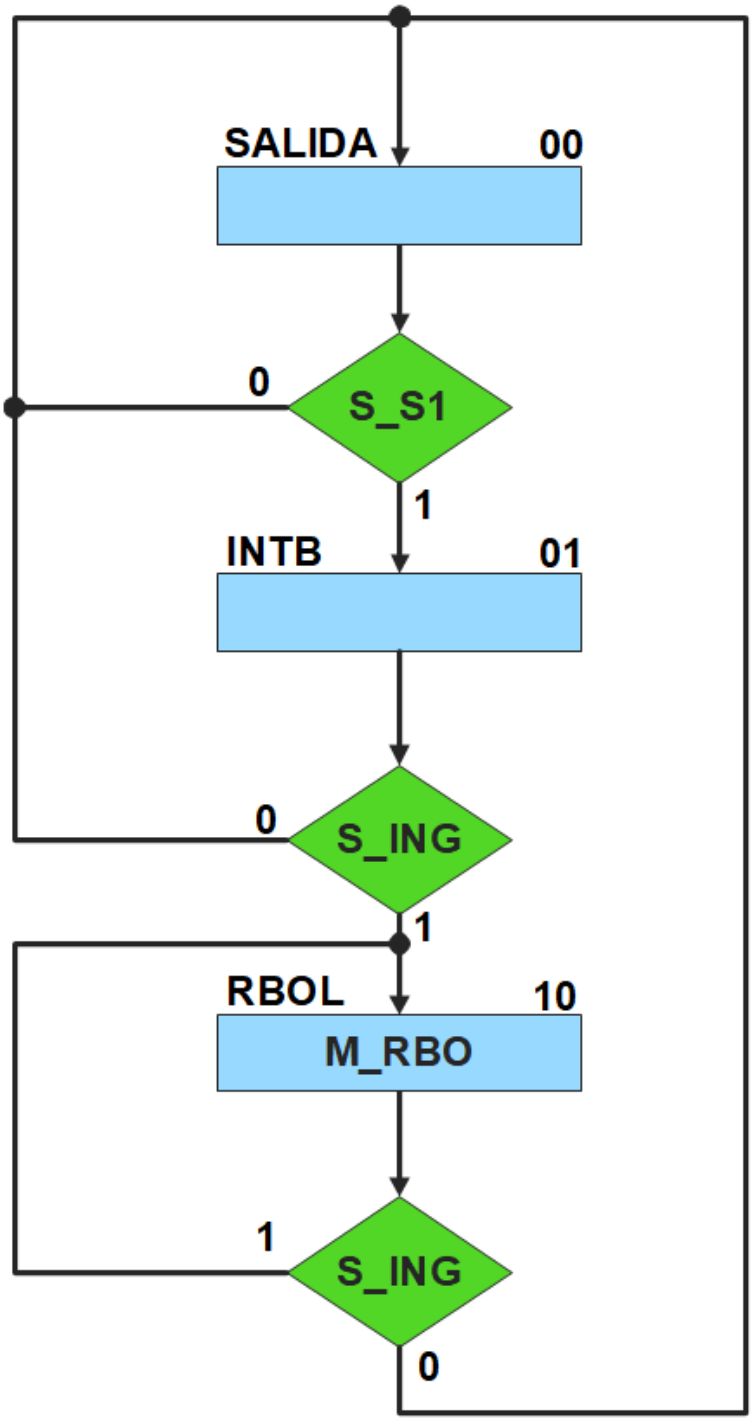


Figura P6.5 Carta ASM del sistema de recolección de boletos.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P6.2).

Tabla P6.2 Representación binaria de entradas para el sistema de recolección de boletos.

Entrada	Prueba
S_S1	0
S_ING	1

Se debe llenar la tabla P6.3 con base en la información de la carta ASM de la figura P6.5, usando el método de diseño con memoria y direccionamiento entrada-estado.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘00’.

En el Estado ‘00’ se selección la entrada S_S1, por lo tanto, se coloca en el campo de prueba de la memoria su representación binaria, es decir, ‘0’. Si S_S1 es igual a cero, el estado siguiente es el Estado ‘00’, su representación binaria ‘00’ es colocada en el campo de la liga falsa. Si S_S1 es igual a uno, el estado siguiente es el Estado ‘01’, su representación binaria ‘01’ es colocada en el campo de la liga verdadera. En el Estado ‘00’ la señal de salida del motor M_RBO no están activada, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P6.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P6.3 Contenido de la memoria para el sistema de recolección de boletos.

Dirección de memoria		Contenido de memoria						
Estado presente		Prueba	Liga Falsa		Liga Verdadera		Salida	Hex
QA	QB	I0	LF1	LF0	LV1	LV0	M_RBO	
0	0	0	0	0	0	1	0	02
0	1	1	0	0	1	0	0	24
1	0	1	0	0	1	0	1	25

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de dos líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de éste (ver tabla P6.4). Se puede comprobar realizando el circuito de la figura P6.6.

Tabla P6.4 Tabla de funcionamiento del selector de entradas para el sistema de recolección de boletos.

Prueba	SI
I0	
0	S_S1
1	S_ING

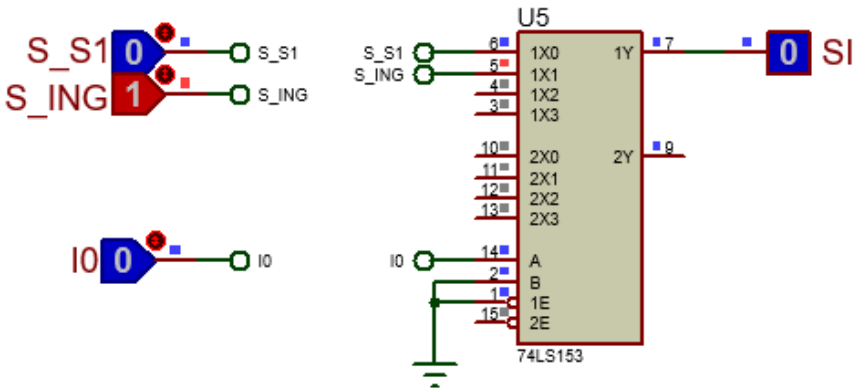


Figura P6.6 Multiplexor 74LS153 para el selector de entradas del sistema de recolección de boletos.

La función booleana del selector de entradas queda:

```
SI = S_S1&!I0 | S_ING&I0;
```

El selector de liga es un multiplexor doble de dos líneas a una, éste es implementado por el PIC16F1939. Si el selector entradas es igual a '1', se selecciona la información binaria de la liga verdadera. Si el selector entradas es igual a '0', se selecciona la información binaria de la liga falsa. Se puede comprobar realizando el circuito de la figura *P6.7*.

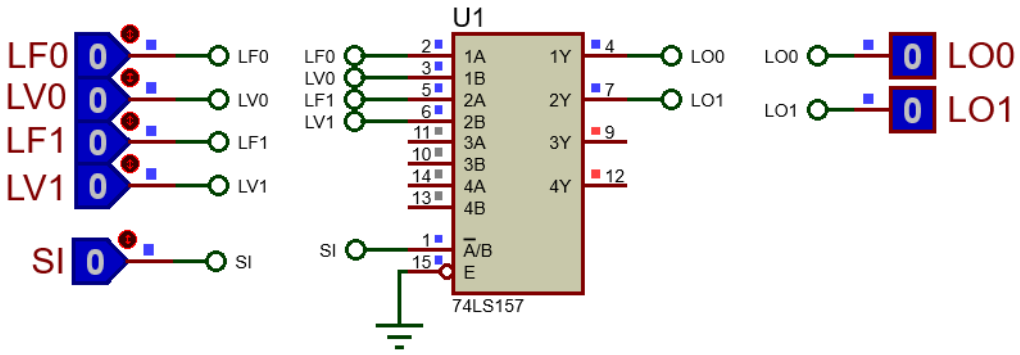


Figura P6.7 Multiplexor 74LS157 para el selector de liga del sistema de recolección de boletos.

Las funciones booleanas del selector de liga quedan:

```
LO0 = !SI&LF0 | SI&LV0;

LO1 = !SI&LF1 | SI&LV1;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P6.8, figura P6.9). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939.

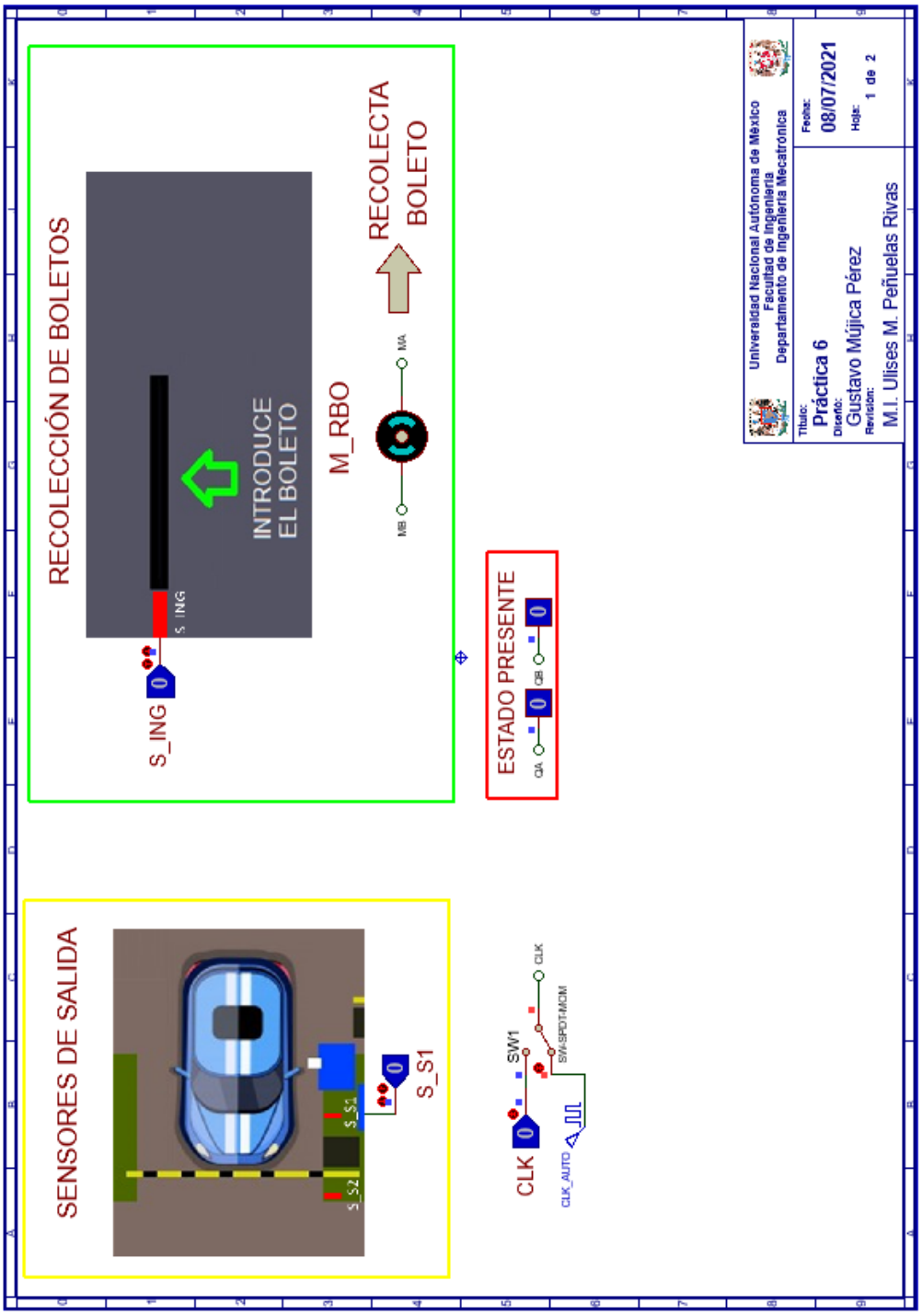


Figura P6.8 Interfaz hombre-máquina para el controlador de la Práctica 6 hoja 1/2.

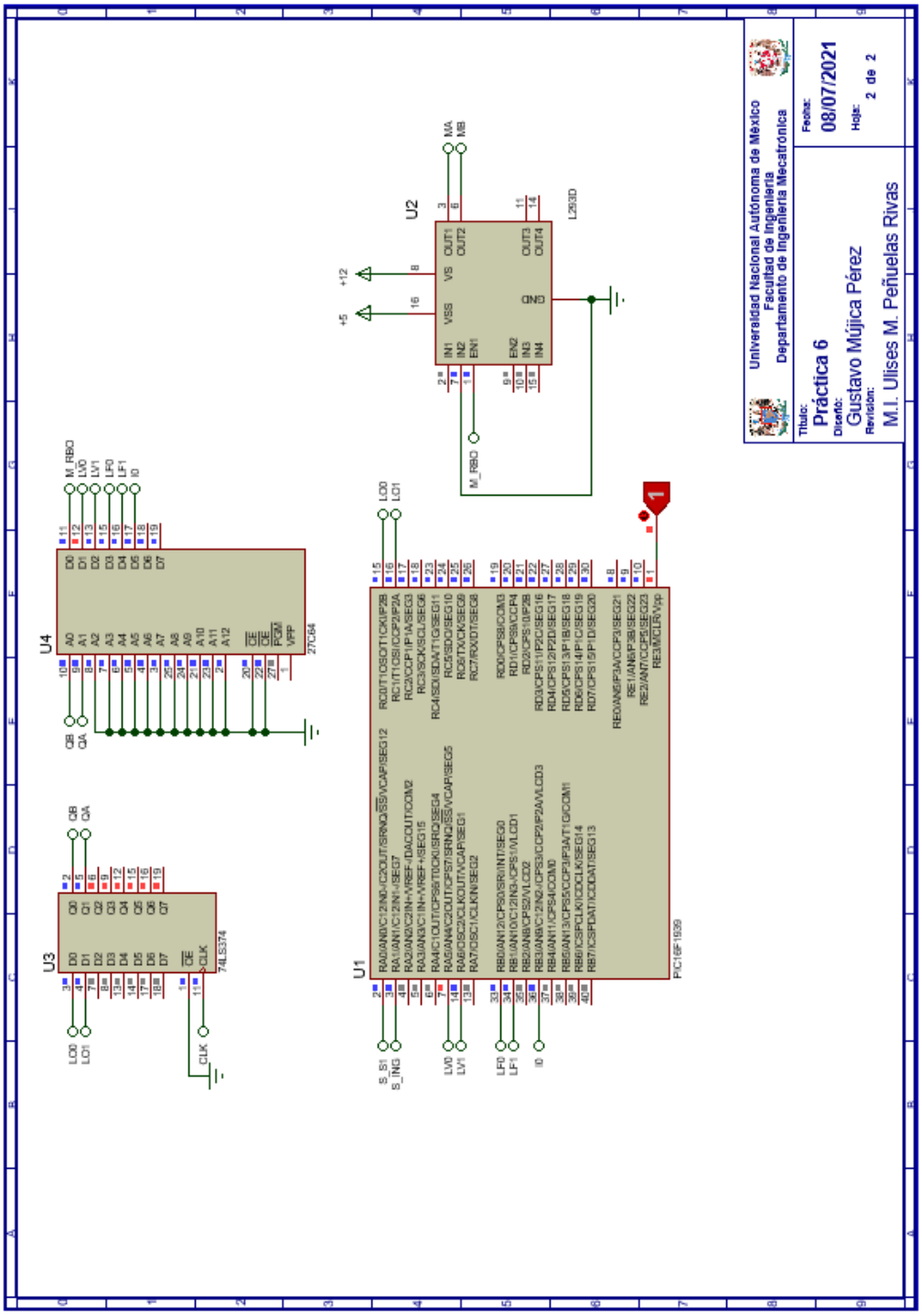


Figura P6.9 Esquema electrónico para el controlador de la Práctica 6 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P6.10) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     //Carga biblioteca PLD.h

//***ENTRADAS***

#define S_S1   A0 //ENTRADA
#define S_ING  A1 //ENTRADA

//LIGAS VERDADERAS
#define LV0 A5
#define LV1 A6

//LIGAS FALSAS
#define LF0 B0
#define LF1 B1

//PRUEBAS
#define I0 B3 //PRUEBA


//***SALIDAS***

#define LO0 C0
#define LO1 C1

//***VARIABLES INTERMEDIAS***

short SI; //SELECTOR DE ENTRADA

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD


//LOOP INFINITO
   while(1)
    {

    //SELECTOR DE ENTRADAS
    SI= S_S1&!I0 | S_ING&I0 ;

    //SELECTOR DE LIGA
   LO0= !SI&LF0 | SI&LV0;
   LO1= !SI&LF1 | SI&LV1;

    }

}
```

Figura P6.10 Código de la Práctica 6.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 7 Sistema de emisión/retiro de boletos; diseño con memoria y direccionamiento implícito

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P7.1).

Figura P7.1 Maqueta de estacionamiento con 10 lugares.

En la entrada se debe adquirir un boleto para poder ingresar al estacionamiento. El sistema de emisión/retiro de boletos entrega un boleto de forma automática al presionar un botón, el boleto debe salir a través de una ranura (ver figura P7.2).

Figura P7.2 Sección de la maqueta en donde se emite el boleto.

El diseño con memoria y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura P7.3). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura P7.4 muestra la arquitectura de este método.

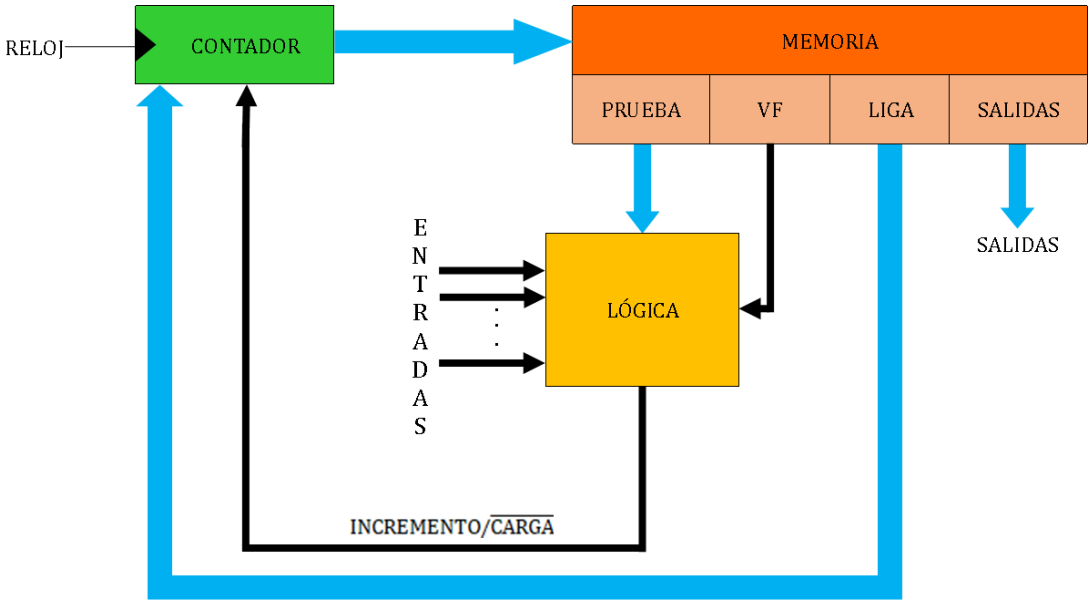


Figura P7.3 Arquitectura de un controlador con memoria y direccionamiento implícito.

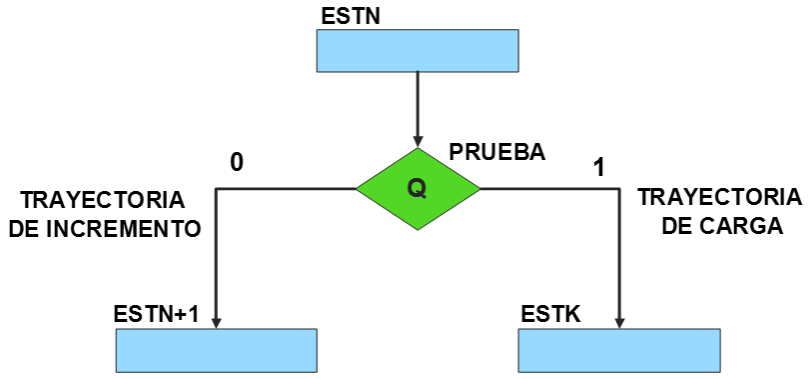


Figura P7.4 Trayectoria de incremento y carga.

La tabla P7.1 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que valor de entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un '1' se hace un incremento, cuando el resultado de la función da como resultado un '0' se hace una carga.

Tabla P7.1 Relación entre VF, la variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incremento o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es '0', se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es '1', se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura P7.5).

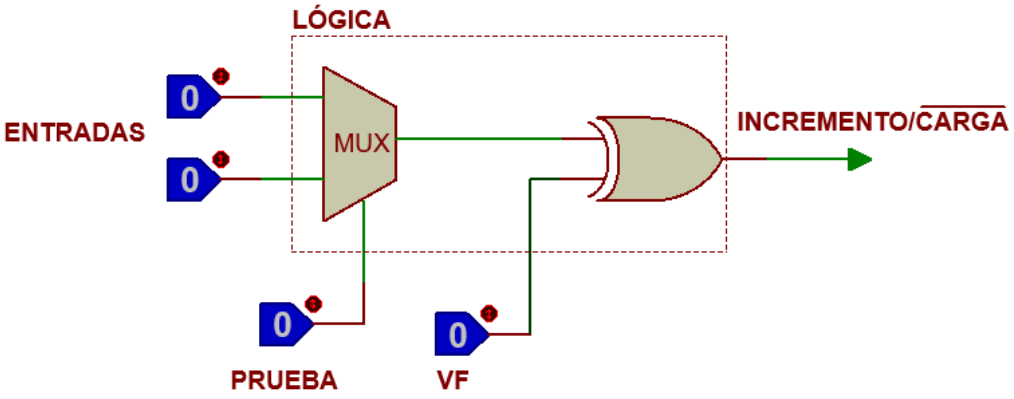


Figura P7.5 Bloque de lógica incremento/carga para el direccionamiento implícito [1].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

Objetivo

Diseñar un controlador para el sistema de emisión/retiro de boletos, por medio del método de diseño con memoria y direccionamiento implícito.

Descripción

Primero se diseña una carta ASM para el sistema de emisión/retiro de boletos por medio del método de diseño con memoria y direccionamiento implícito. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P7.1 se muestran los detalles de las entradas y salidas de este controlador.

Para la emisión de boletos se necesita señales de entrada:

- un sensor detecta la presencia de vehículo
- un botón detecta la solicitud de emisión de boleto
- un sensor verifica la emisión y, posteriormente, el retiro del boleto (ver Figura P7.2).

Como salida se requiere de las siguientes señales:

- activación del motor de emisión del boleto.

Tabla P7.2 Entradas y salidas del sistema de emisión/retiro de boletos.

Identificador	Ubicación	Tipo	Descripción
CLK	A0 del PIC	I	Señal de reloj
S_E1	A2 del PIC	I	Sensor de entrada que detecta cuándo un vehículo llega a la entrada del estacionamiento.
BOT	A3 del PIC	I	Botón para obtener un boleto.
S_RET	A4 del PIC	I	Sensor de emisión y retiro de boleto.
M_EBO	D0 de la memoria	O	Motor que emitirá el boleto.

Notas de diseño

a) El sensor S_E1 está ubicado antes del cruce de la pluma

b) El sensor S_RET tiene dos funciones: detectar cuándo el boleto ha sido emitido y cuándo el boleto ha sido retirado.

Reglas de funcionamiento

- S_E1: sensor de entrada

  1 = detecta vehículo

  0 = no detecta vehículo
- BOT: botón para obtener un boleto

  1 = se presionó el botón

  0 = no se presionó el botón
- S_RET: sensor de emisión y retiro de boleto

  1 = detecta que se ha emitido el boleto o detecta que el boleto no ha sido retirado

  0 = detecta que no se ha emitido el boleto o detecta que el boleto ha sido retirado.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Se hace un incremento cuando el valor del estado siguiente aumenta en una unidad, de lo contrario se hace una carga (ver figura P7.4). El algoritmo de la máquina de estados se puede ver en la figura P7.6.

Estado ‘000’ – ENTRADA

Entrada del vehículo. En el primer estado, el controlador de la emisión de boletos se encuentra en espera. El sensor (S_E1), al detectar un vehículo, permite al sistema pasar al Estado ‘001’. De lo contrario, permanece en el Estado ‘000’.

Estado ‘001’ – EBOT

En este estado se espera a que se presione el botón (BOT) para la emisión del boleto. Al ser presionado el botón, el sistema avanza al Estado ‘010’ para emitir el boleto. De lo contrario, regresa al Estado ‘000’ para verificar si aún hay un vehículo presente.

Estado ‘010’ – EBOL

Emisión de boleto. En este estado se activa el motor correspondiente (M_EBO) para que se emita un boleto. Cuando el sensor (S_RET) detecta que el boleto ya se encuentra listo para retirarse, el sistema pasa al Estado ‘011’ donde se desactiva al motor y se espera a que se retire el boleto. Si el sensor aún no detecta el boleto, permanece en el Estado ‘010’.

Estado ‘011’ – ERET

En este estado se espera a que el boleto sea retirado. Cuando el sensor (S_RET) detecta que el boleto se ha retirado, avanza al Estado ‘100’, para volver a iniciar el proceso de emisión y retiro de boleto. De lo contrario, permanece en el Estado ‘011’, en espera a que el boleto sea retirado.

Estado ‘100’ – AUX1

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada avanzar al Estado '000', haciendo una carga, para iniciar nuevamente el proceso de emisión y retiro del boleto.

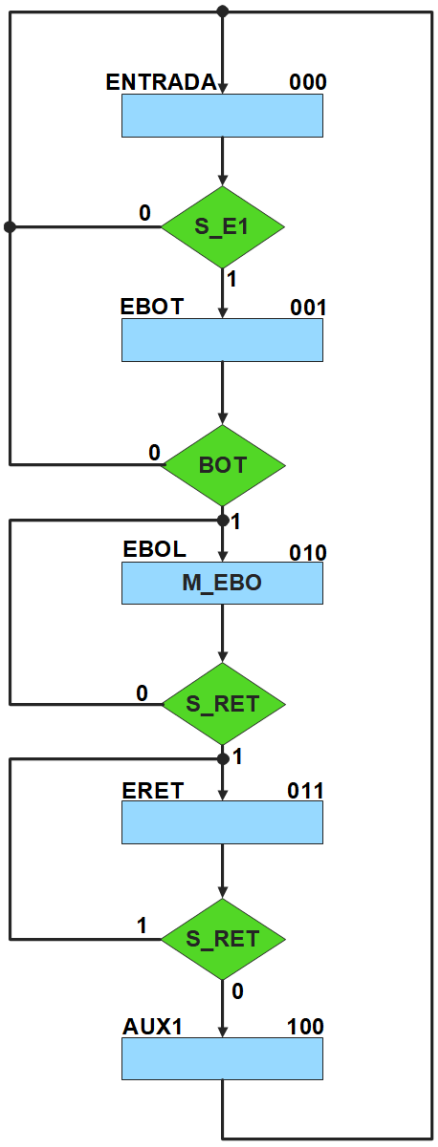


Figura P7.6 Carta ASM del sistema de emisión/retiro de boletos.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P7.3).

Tabla P7.3 Representación binaria de entradas para el sistema de emisión/retiro de boletos.

Entrada	Prueba
AUX	0 0
S_E1	0 1
BOT	1 0
S_RET	1 1

Se debe llenar la tabla P7.4 con base en la información de la carta ASM de la figura P7.6, usando el método de diseño con memoria y direccionamiento implícito.

A continuación, se describe como llenar los campos de la memoria para el Estado ‘000’.

En el Estado ‘000’ se selecciona la entrada S_E1, por lo tanto, se coloca en el campo de prueba su representación binaria, es decir, ‘01’. Si S_E1 es igual a cero, entonces el estado siguiente es el Estado ‘000’, su representación binaria ‘000’ es colocada en el campo de la liga, ya que se requiere hace una carga. El campo VF es igual cero, ya que, para hacer una carga en el contador, el valor de la entrada y de VF deben ser iguales. En el Estado ‘00’ el motor M_EBO no está activado, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P7.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P7.4 Contenido de la memoria para el sistema de emisión/retiro de boletos.

Dirección de memoria			Contenido de memoria							
Estado presente			Prueba		VF	Liga			Salida	Hex
QA	QB	QC	I1	I0		L2	L1	L0	M_EBO	
0	0	0	0	1	0	0	0	0	0	20
0	0	1	1	0	0	0	0	0	0	40
0	1	0	1	1	0	0	1	0	1	65
0	1	1	1	1	1	0	1	1	0	76
1	0	0	0	0	1	0	0	0	0	10

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de cuatro líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P7.5). Se puede comprobar realizando el circuito de la figura P7.7.

Tabla P7.5 Tabla de funcionamiento del selector de entradas para el sistema de emisión/retiro de boletos.

Prueba		SI
I1	I0	
0	0	AUX
0	1	S_E1
1	0	BOT
1	1	S_RET

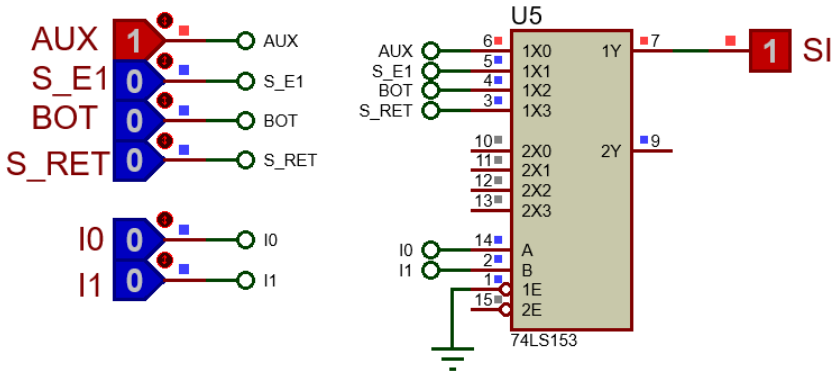


Figura P7.7 Multiplexor 74LS153 para el selector de entradas del sistema de emisión/retiro de boletos.

La función booleana del selector de entradas queda:

```
SI = AUX&!I1&!I0 | S_E1&!I1&I0 | BOT&I1&!I0 | S_RET&I1&I0;
```

Para obtener el valor de la lógica, se debe hacer la operación XOR entre el selector de entradas y el valor de VF (ver figura P7.5).

Por lo tanto, la función booleana de la lógica queda:

```
SL = VF ^ (AUX&!I1&!I0 | S_E1&!I1&I0 | BOT&I1&!I0 | S_RET&I1&I0);
```

Se utiliza un contador con carga paralela que indica el estado siguiente. El contador con carga paralela es implementado por el PIC16F1939. Si el valor a la salida de la lógica es igual a ‘1’, el contador carga la información binaria de la liga a la memoria. Si el valor de la lógica es igual a ‘0’, se cuenta al siguiente estado binario. A continuación, se obtiene las expresiones lógicas para un contador de tres bits por el método de variable suscrita (ver tabla P7.6).

Tabla P7.6 Mapa de Karnaugh general de transición de estados para un contador de tres bits.

		B C			
		0 0	0 1	1 1	1 0
A	0	000	001	011	010
		001	010	100	011
	1	100	101	111	110
		101	110	000	111

Posteriormente se procede a obtener el mapa de Karnaugh particular para cada uno de los bits de estado y sus expresiones lógicas:

Para el bit A:

		B C			
		00	01	11	10
A	0	0	0	1	0
	1	1	1	0	1

```
FFA = A&!B | !A&B&C | A&!C;
```

Para el bit B:

		B C			
		00	01	11	10
A	0	0	1	0	1
	1	0	1	0	1

```
FFB = !B&C | B&!C = B^C;
```

Para el bit C:

		B C			
		00	01	11	10
A	0	1	0	0	1
	1	1	0	0	1

```
FFC = !C;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P7.8,figura P7.9). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939.

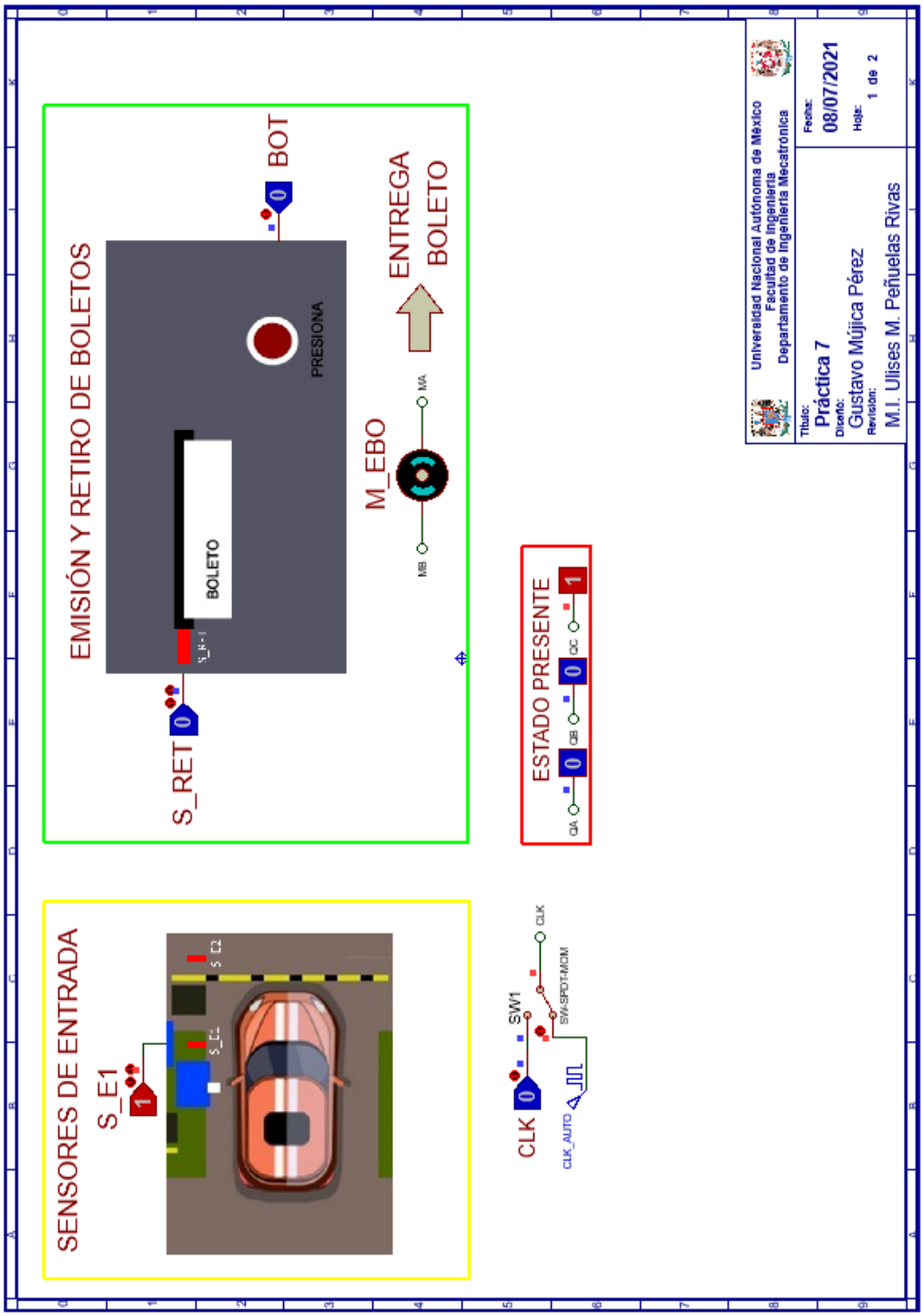


Figura P7.8 Interfaz hombre-máquina para el controlador de la Práctica 7 hoja 1/2.

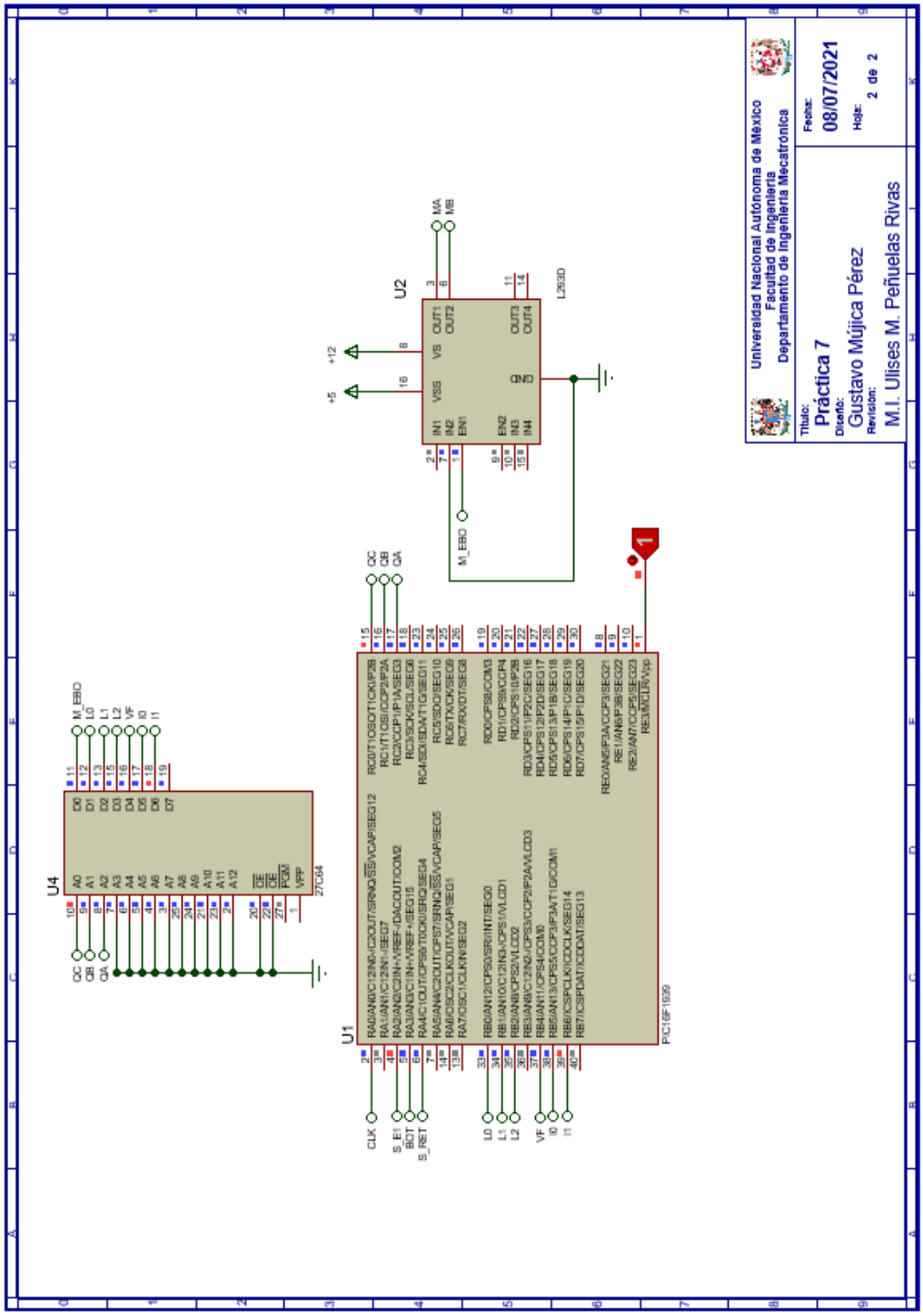


Figura P7.9 Esquema electrónico para el controlador de la Práctica 7 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P7.10, figura P7.11) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     // Carga biblioteca PLD.h

//***CONTADOR***

#define CLK A0 //RELOJ

#define A C2 //FFA
#define B C1 //FFB
#define C C0 //FFC

//LIGA
#define L0 B0
#define L1 B1
#define L2 B2


//***LOGICA***

//ENTRADAS
#define S_E1  A2
#define BOT   A3
#define S_RET A4

//VF
#define VF B4

//PRUEBAS
#define I0 B5
#define I1 B6

//VARIABLES INTERMEDIAS
short SL=0;  //Salida lógica
short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
short At=0, Bt=0, Ct=0; //Variables inntetrnas de los FF
short LD2,LD1,LD0; //Variables intermedias de la liga

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD


   //LOOP INFINITO
   while(1)
   {
   //CIRCUITO COMBINACIONAL
   LD2=L2; LD1=L1; LD0=L0;     /*ALMACENA DATOS HASTA EL CAMBIO DEL RELOJ,
                                 SIMULANDO UN REGISTRO, EVITANDO ASÍ
                                 QUE SE MODIFIQUEN LOS VALORES DE LA MEMORIA*/

```

Figura P7.10 Código de la Práctica 7 parte 1.

```
50:
51:     //SALIDA LÓGICA
52:      SL= VF ^ (AUX&!I1&!I0 | S_E1&!I1&I0 | BOT&I1&!I0 | S_RET&I1&I0);
53:
54:     //CIRCUITO SECUENCIAL (CONTADOR DE CARGA PARALELA)
55:
56:     if (!CLK)
57:     {//SECCIÓN DE OPERACIONES DEL CONTADOR
58:     At= A&!B | !A&B&C | A&!C;
59:     Bt= B^C;
60:     Ct= !C;
61:     }
62:
63:     else
64:     {//CUENTA CON SL=1 Y CARGA CON SL=0
65:     A= SL&At | !SL&LD2;
66:     B= SL&Bt | !SL&LD1;
67:     C= SL&Ct | !SL&LD0;
68:     while(clk){/*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
69:                 POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
70:                 DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
71:                 LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ*/}
72:     }
73:
74:     }
75:
76: }
```

Figura P7.11 Código de la Práctica 7 parte 2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 8 Sistema de pluma para estacionamiento; diseño con memoria y direccionamiento implícito

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P8.1).

Figura P8.1 Maqueta de estacionamiento con 10 lugares.

Se usan plumas que impiden el avance del vehículo a la entrada y salida del estacionamiento. Se levanta la pluma para permitir el paso de un vehículo y cuando éste haya pasado completamente, se baja la pluma (ver figura P8.2 y figura P8.3).

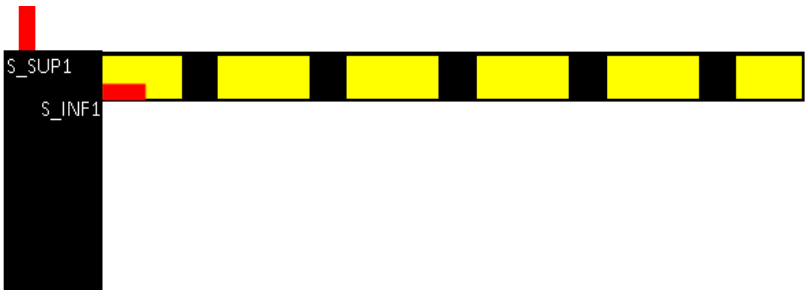


Figura P8.2 Sensores de posición superior e inferior en la pluma.

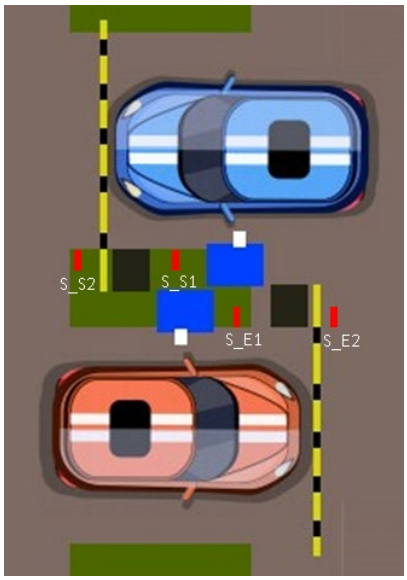


Figura P8.3 Sensores a la entrada y salida del estacionamiento.

El diseño con memoria y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura P8.4). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura P8.5 muestra la arquitectura de este método.

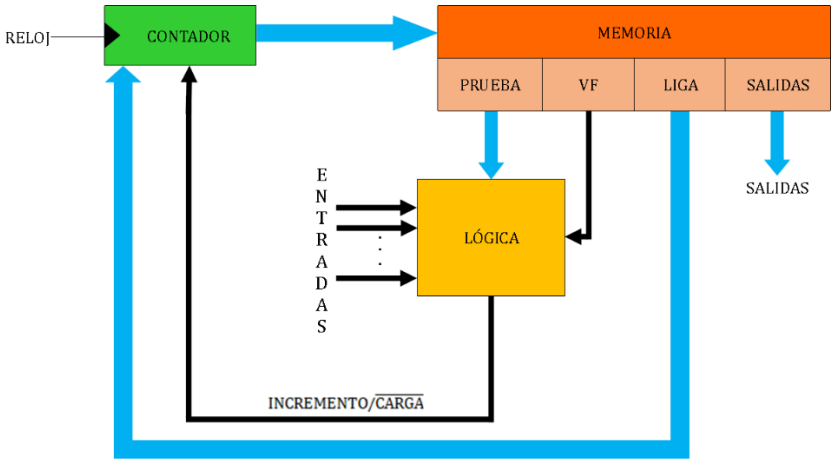


Figura P8.4 Arquitectura de un controlador con memoria y direccionamiento implícito.

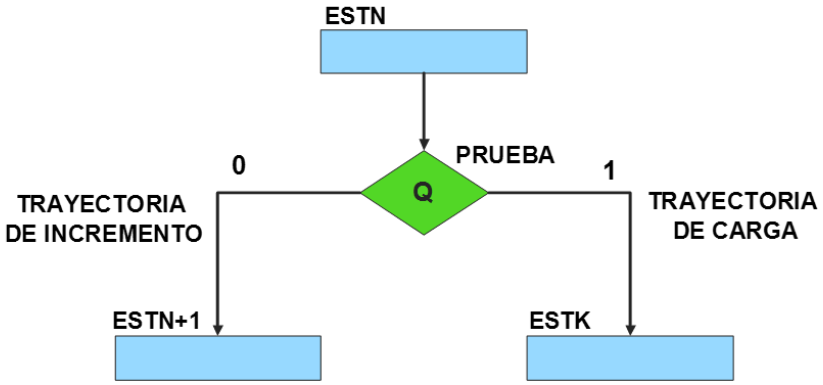


Figura P8.5 Trayectoria de incremento y carga.

La tabla P8.1 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que valor de entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un '1' se hace un incremento, cuando el resultado de la función da como resultado un '0' se hace una carga.

Tabla P8.1 Relación entre VF, la variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incremento o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es '0', se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es '1', se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura P8.6).

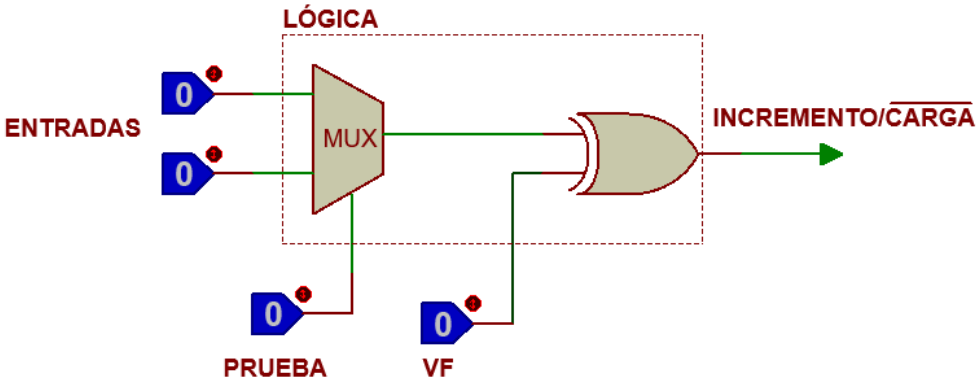


Figura P8.6 Bloque de lógica de incremento/carga para el direccionamiento implícito [1].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

Objetivo

Diseñar un controlador para el sistema de pluma para estacionamiento, por medio del método de diseño con memoria y direccionamiento implícito.

Descripción

Primero se diseña una carta ASM para el sistema de pluma para estacionamiento por medio del método de diseño con memoria y direccionamiento implícito. Posteriormente se propone una solución para implementar el sistema.

Nota: Para este diseño se usarán los sensores de la pluma de entrada, sin embargo, el proceso es el mismo para la pluma de entrada como para la pluma de salida.

Tabla de entradas y salidas

En la tabla P8.2 se muestran los detalles de las entradas y salidas de este controlador.

Para la pluma del estacionamiento se necesitan señales de entrada:

- un sensor detecta cuándo la pluma se encuentra en la posición superior y otro cuando se encuentra en la posición inferior (ver figura P8.2)
- un sensor detecta cuándo un vehículo está a la entrada del estacionamiento y otro cuando se esté ingresando a éste (ver figura P8.3).

Como salida se requiere de las siguientes señales:

- una señal activa el motor de la pluma
- una señal se activa para que la pluma suba y otra para que la pluma baje.

Tabla P8.2 Entradas y salidas para el sistema de pluma para estacionamiento.

Identificador	Ubicación	Tipo	Descripción
CLK	A0 del PIC	I	Señal de reloj
S_E1	A3 del PIC	I	Sensor de entrada que detecta cuándo un vehículo llega a la entrada del estacionamiento
S_E2	A4 del PIC	I	Sensor de ingreso que detecta cuándo un vehículo está ingresando al estacionamiento
S_SUP1	A5 del PIC	I	Sensor de posición superior que detecta cuándo la pluma se ha elevado en su totalidad
S_INF1	A6 del PIC	I	Sensor de posición inferior que detecta cuándo la pluma se ha bajado en su totalidad
MOT_1	D2 de la memoria 2	O	Motor de la pluma
MX_1	D1 de la memoria 2	O	Señal para que la pluma suba
nMX_1	D0 de la memoria 2	O	Señal para que la pluma baje.

Notas de diseño

a) El sensor S_E1 está ubicado antes del cruce de la pluma

b) El sensor S_E2 está ubicado después del cruce de la pluma.

Reglas de funcionamiento

- S_E1: sensor de entrada
  1 = detecta vehículo
  0 = no detecta vehículo
- S_E2: sensor de ingreso
  1 = detecta vehículo ingresando
  0 = no detecta vehículo ingresando
- S_SUP1: sensor de posición superior
  1 = detecta pluma completamente elevada
  0 = no detecta pluma completamente elevada
- S_INF1: sensor de posición inferior
  1 = detecta que la pluma ha bajado completamente
  0 = no detecta que la pluma ha bajado completamente.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Se hace un incremento cuando el valor del estado siguiente aumenta en una unidad, de lo contrario se hace una carga (ver figura P8.5). El algoritmo de la máquina de estados se puede ver en la figura P8.7.

Estado '0000' – ELEVA

En el primer estado, se activa el motor (MOT_1) y la señal (MX_1) para que la pluma suba, permitiendo el ingreso. Si el sensor (S_SUP1) detecta que la pluma ha sido elevada completamente, el sistema avanza al Estado '0001' para esperar a que pase el vehículo. De lo contrario avanza al Estado '0100' para verificar si hay un vehículo a la entrada.

Estado '0001' – ESPERA

En este estado, la pluma esta elevada, esperando a que el vehículo pase completamente. Si el sensor (S_E1) no detecta un vehículo a la entrada, el sistema avanza al Estado '0010' para revisar si un vehículo está ingresando. De lo contrario, permanece en el Estado '0001'.

Estado '0010' – AUX1

Si el sensor (S_E2) no detecta un vehículo ingresando, avanza al Estado '0011' para que posteriormente se baje la pluma ya que el vehículo ha pasado completamente. De lo contrario, el sistema regresa al Estado '0001' para esperar a que el vehículo pase completamente.

Estado '0011' – AUX2

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a avanzar al Estado '0110', haciendo una carga, para que se baje la pluma ya que el vehículo ha pasado completamente.

Estado '0100' – AUX3

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (MX_1) para que la pluma suba. Cuando el sensor (S_E1) no detecta vehículo a la entrada, el sistema avanza al Estado '0101' para revisar si hay un vehículo ingresando. De lo contrario, regresa al Estado '0000' para seguir elevando la pluma.

Estado '0101' – AUX4

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (MX_1) para que la pluma suba.

Si el sensor (S_E2) no detecta un vehículo ingresando, el sistema avanza al Estado ‘0110’ para bajar la pluma, debido a que el auto ha pasado antes de que la pluma se eleve completamente. De lo contrario, regresa al Estado ‘0000’ para seguir elevando la pluma.

Estado ‘0110’ – BAJA

En este estado se activa el motor (MOT_1) y la señal (nMX_1) para que la pluma baje. Si el sensor (S_INF1) detecta que la pluma no ha bajado en su totalidad, avanza al Estado ‘0111’ para revisar si hay un vehículo a la entrada. De lo contrario, regresa al Estado ‘0000’ para elevar la pluma nuevamente.

Estado ‘0111’ – AUX5

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (nMX_1) para que la pluma siga bajando. Si el sensor (S_E1) no detecta un vehículo a la entrada, el sistema avanza al Estado ‘1000’ para revisar si hay un auto ingresando. De lo contrario, regresa al Estado ’0000’ para subir la pluma y que ésta no dañe al vehículo.

Estado ‘1000’ – AUX6

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (nMX_1) para que la pluma siga bajando. Si el sensor (S_E2) no detecta un vehículo, el sistema regresa al Estado ‘0110’ para seguir bajando la pluma. De lo contrario, avanza al Estado ‘1001’ para subir la pluma y que ésta no dañe al vehículo.

Estado ‘1001’ – AUX7

En este estado se activa la señal del motor (MOT_1) y la señal (MX_1) para que la pluma suba y no dañe al vehículo. En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada avanzar al Estado ‘0000’, haciendo una carga, para que la pluma siga subiendo y no dañe al vehículo.

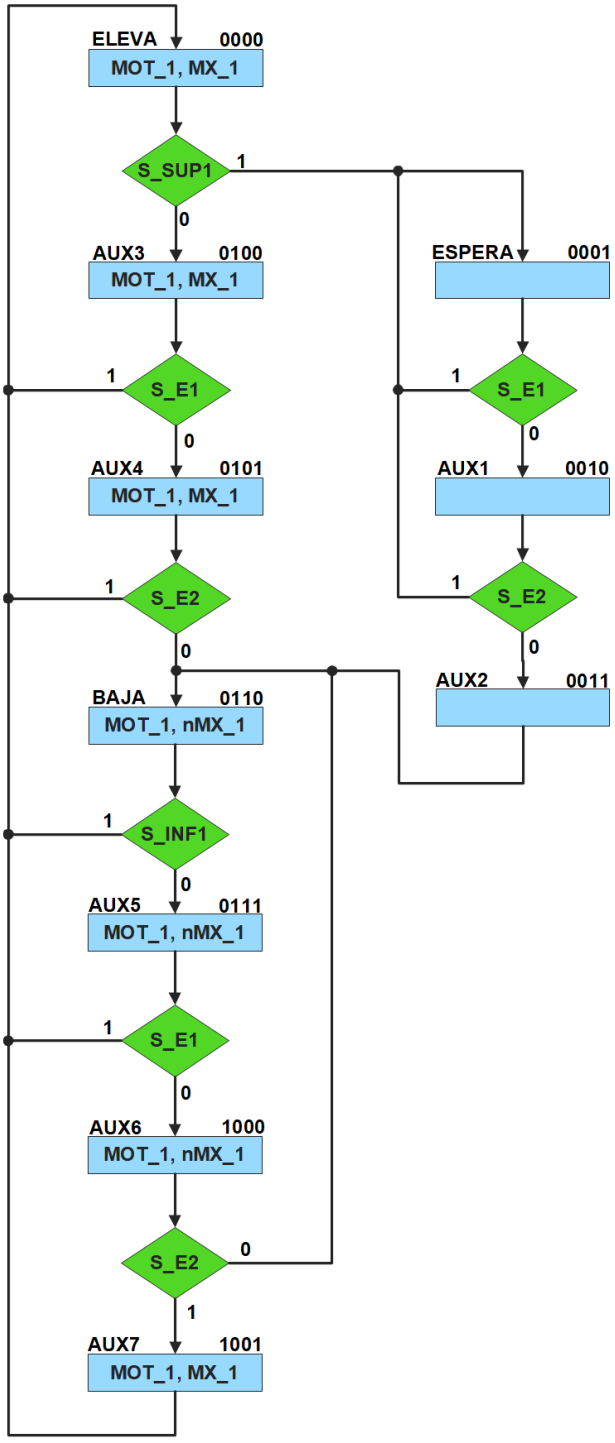


Figura P8.7 Carta ASM del sistema de pluma para estacionamiento.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P8.3).

Tabla P8.3 Representación binaria de entradas para el sistema de pluma para estacionamiento.

Entrada	Prueba
AUX	0 0 0
S_E1	0 0 1
S_E2	0 1 0
S_SUP1	0 1 1
S_INF1	1 0 0

Se debe llenar la tabla P8.4 con base en la información de la carta ASM de la figura P8.7, usando el método de diseño con memoria y direccionamiento implícito.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘0000’.

En el Estado ‘0000’ se selecciona la entrada S_SUP1, por lo tanto, se coloca en el campo de prueba su representación binaria, es decir, ‘011’. Si S_SUP1 es igual a cero, entonces el estado siguiente es el Estado ‘0100’, su representación binaria ‘0100’ es colocada en el campo de la liga, ya que se requiere hace una carga. El campo VF es igual cero, ya que, para hacer una carga en el contador, el valor de la entrada y de VF deben ser iguales. En el Estado ‘0000’ esta activado el motor MOT_1 y la señal MX_1 para que la pluma suba, por lo que se coloca un ‘1’ en la parte de salidas de estas señales.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P8.4). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P8.4 Contenido de la memoria para el sistema de pluma para estacionamiento.

Dirección de memoria				Contenido de memoria												
Estado presente				Prueba			VF	Liga				Salidas			Hex 1	Hex 2
QA	QB	QC	QD	I2	I1	I0		L3	L2	L1	L0	MOT_1	MX_1	nMX_1		
0	0	0	0	0	1	1	0	0	1	0	0	1	1	0	64	06
0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	31	00
0	0	1	0	0	1	0	1	0	0	0	1	0	0	0	51	00
0	0	1	1	0	0	0	1	0	1	1	0	0	0	0	16	00
0	1	0	0	0	0	1	1	0	0	0	0	1	1	0	30	06
0	1	0	1	0	1	0	1	0	0	0	0	1	1	0	50	06
0	1	1	0	1	0	0	1	0	0	0	0	1	0	1	90	05
0	1	1	1	0	0	1	1	0	0	0	0	1	0	1	30	05
1	0	0	0	0	1	0	0	0	1	1	0	1	0	1	46	05
1	0	0	1	0	0	0	1	0	0	0	0	1	1	0	10	06

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de ocho líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P8.5). Se puede comprobar realizando el circuito de la figura P8.8.

Tabla P8.5 Tabla de funcionamiento del selector de entradas para el sistema de pluma para estacionamiento.

Prueba			SI
I2	I1	I0	
0	0	0	AUX
0	0	1	S_E1
0	1	0	S_E2
0	1	1	S_SUP1
1	0	0	S_INF1

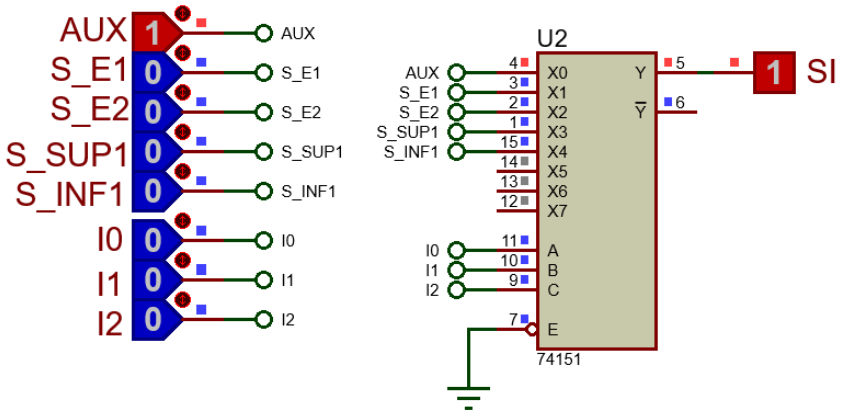


Figura P8.8 Multiplexor 74151 para el selector de entradas del sistema de pluma para estacionamiento.

Por lo tanto, la función booleana del selector de entradas queda:

```
SI = AUX&!I2&!I1&!I0 | S_E1&!I2&!I1&I0 | S_E2&!I2&I1&!I0 |
     S_SUP1&!I2&I1&I0 | S_INF1& I2&!I1&!I0;
```

Para obtener el valor de la lógica, se debe hacer la operación XOR entre el selector de entradas y el valor de VF (ver figura P8.6).

Por lo tanto, la función booleana de la lógica queda:

```
SL = VF ^ (AUX&!I2&!I1&!I0 | S_E1&!I2&!I1&I0 | S_E2&!I2&I1&!I0 |
     S_SUP1!I2&I1&I0 | S_INF1& I2&!I1&!I0);
```

Se utiliza un contador con carga paralela que indica el estado siguiente. El contador con carga paralela será implementado por el PIC16F1939. Si el valor a la salida de la lógica es igual a '1', el contador carga la información binaria de la liga a la memoria. Si el valor de la lógica es igual a '0', se cuenta al siguiente estado binario. A continuación, se obtiene las expresiones lógicas para un contador de cuatro bits por el método de variable suscrita (ver tabla P8.6).

Tabla P8.6 Mapa de Karnaugh general de transición de estados para un contador de cuatro bits.

A B \ C D	0 0	0 1	1 1	1 0
0 0	0000	0001	0011	0010
0 1	0001	0010	0100	0011
1 1	0100	0101	0111	0110
1 0	0101	0110	1000	0111
	1100	1101	1111	1110
	1101	1110	0000	1111
	1000	1001	1011	1010
	1001	1010	1100	1011

Para el bit A:

A B \ C D	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	1	0	1
10	1	1	1	1

```
FFA = A&!C | A&!B | A&!D |
      !A&B&C&D;
```

Para el bit B:

A B \ C D	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	1	1	0	1
10	0	0	1	0

```
FFB = B&!C | B&!D | !B&C&D;
```

Para el bit C:

A B \ C D	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

```
FFC = !C&D | C&!D = C^D;
```

Para el bit D:

A B \ C D	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

```
FFD = !D;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P8.9, figura P8.10). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939.

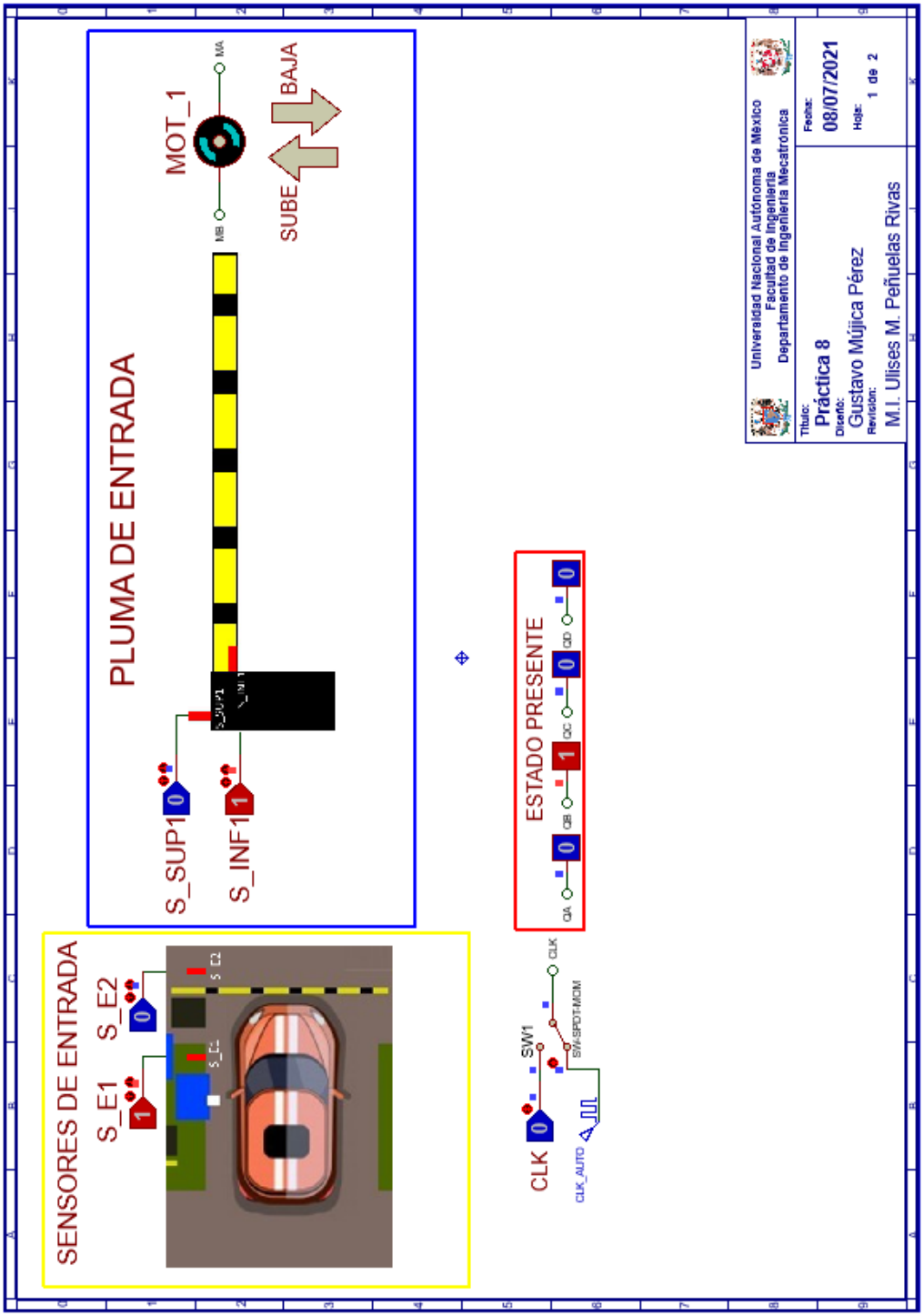


Figura P8.9 Interfaz hombre-máquina para el controlador de la Práctica 8 hoja 1/2.

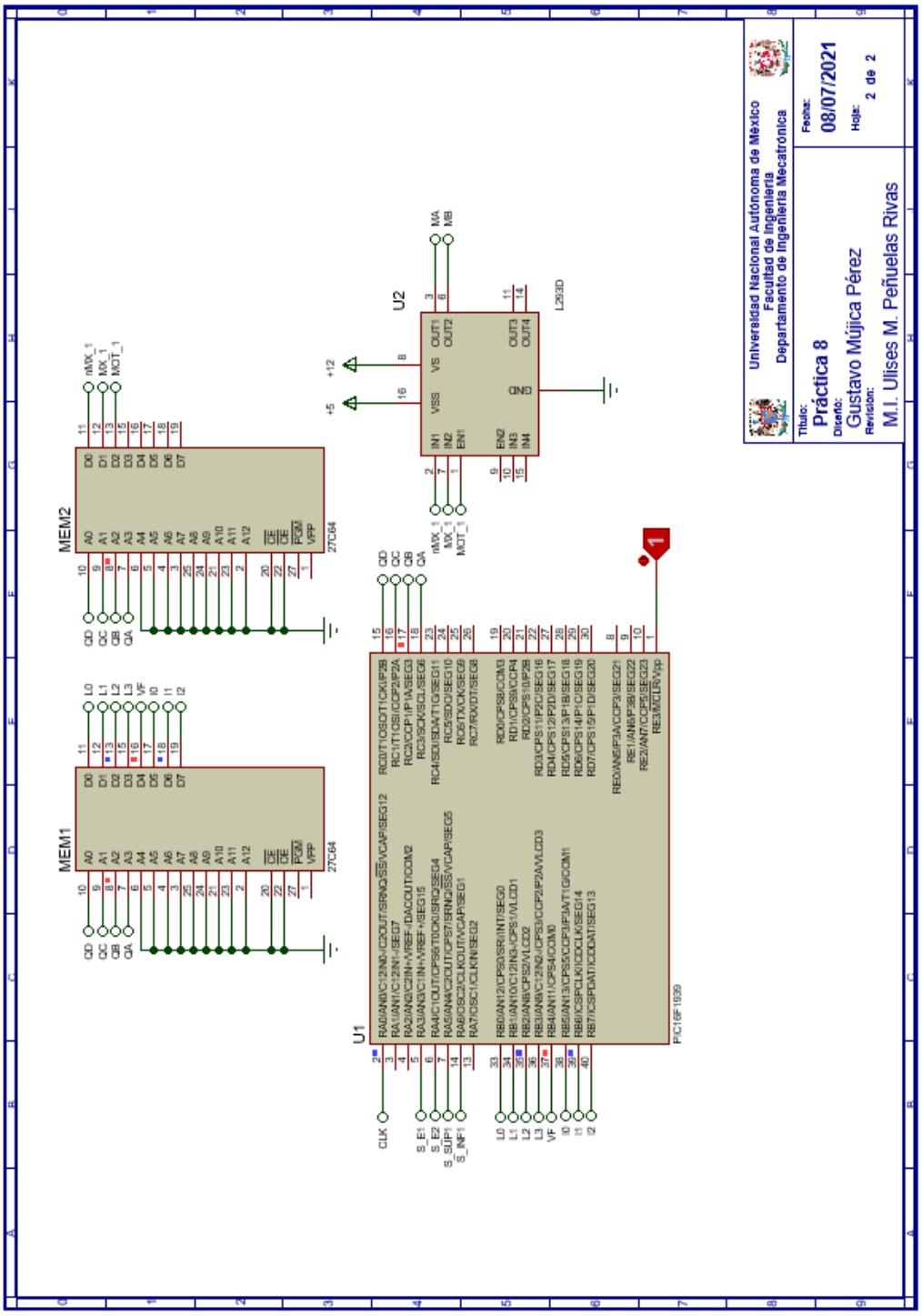


Figura P8.10 Esquema electrónico para el controlador de la Práctica 8 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P8.11, figura P8.12) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     // Carga biblioteca PLD.h

//***CONTADOR***

#define CLK A0 //RELOJ
//ENTRADAS

//LIGA
#define L0 B0
#define L1 B1
#define L2 B2
#define L3 B3

//SALIDAS
#define A C3 //FFA
#define B C2 //FFB
#define C C1 //FFC
#define D C0 //FFD


//***LOGICA***

//ENTRADAS
#define S_E1   A3
#define S_E2   A4
#define S_SUP1 A5
#define S_INF1 A6

//VF
#define VF B4

//PRUEBAS
#define I0 B5
#define I1 B6
#define I2 B7

//VARIABLES INTERMEDIAS
short SL=0;  //Salida lógica
short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
short At=0, Bt=0, Ct=0, Dt=0; //Variables inntetrnas de los FF
short LD3,LD2,LD1,LD0; //Variables intermedias de la liga

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD


   //LOOP INFINITO
   while(1)
```

Figura P8.11 Código de la Práctica 8 parte 1.

```
50:     while(1)
51:     {
52:     //CIRCUITO COMBINACIONAL
53:     LD3=L3; LD2=L2; LD1=L1; LD0=L0;      /*ALMACENA DATOS HASTA EL CAMBIO DEL RELOJ,
54:                                           SIMULANDO UN REGISTRO, EVITANDO ASÍ
55:                                           QUE SE MODIFIQUEN LOS VALORES DE LA MEMORIA*/
56:
57:     //SALIDA LÓGICA
58:     SL= VF ^ (AUX&!I2 & !I1 & !I0 | S_E1&!I2 & !I1 & I0 |
59:     S_E2&!I2 & I1 & !I0 | S_SUP1&!I2 & I1 & I0 | S_INF1& I2 & !I1 & !I0);
60:
61:     //CIRCUITO SECUENCIAL (CONTADOR DE CARGA PARALELA)
62:
63:     if (!CLK)
64:     {//SECCIÓN DE OPERACIONES DEL CONTADOR
65:     At= A&!C | A&!B | A&!D | !A&B&C&D;
66:     Bt= B&!C | B&!D | !B&C&D;
67:     Ct= C^D;
68:     Dt= !D;
69:
70:     }
71:
72:     else
73:     {//CUENTA CON SL=1 Y CARGA CON SL=0
74:     A= SL&At | !SL&LD3;
75:     B= SL&Bt | !SL&LD2;
76:     C= SL&Ct | !SL&LD1;
77:     D= SL&Dt | !SL&LD0;
78:     while(clk){/*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
79:                 POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
80:                 DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
81:                 LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ*/}
82:     }
83:
84:     }
85:
86: }
```

Figura P8.12 Código de la Práctica 8 parte 2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, CDMX: UNAM, Facultad de Ingeniería, 2015.

Práctica 9 Sistema de recolección de boletos; diseño con memoria y direccionamiento implícito

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P9.1).

Figura P9.1 Maqueta de estacionamiento con 10 lugares.

A la salida se debe recolectar el boleto para poder abandonar el estacionamiento. Este sistema recolecta el boleto de forma automática al introducirlo en una ranura (ver figura P9.2).

Figura P9.2 Sección de la maqueta en donde se recolectará el boleto.

El diseño con memoria y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura P9.3). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura P9.4 muestra la arquitectura de este método.

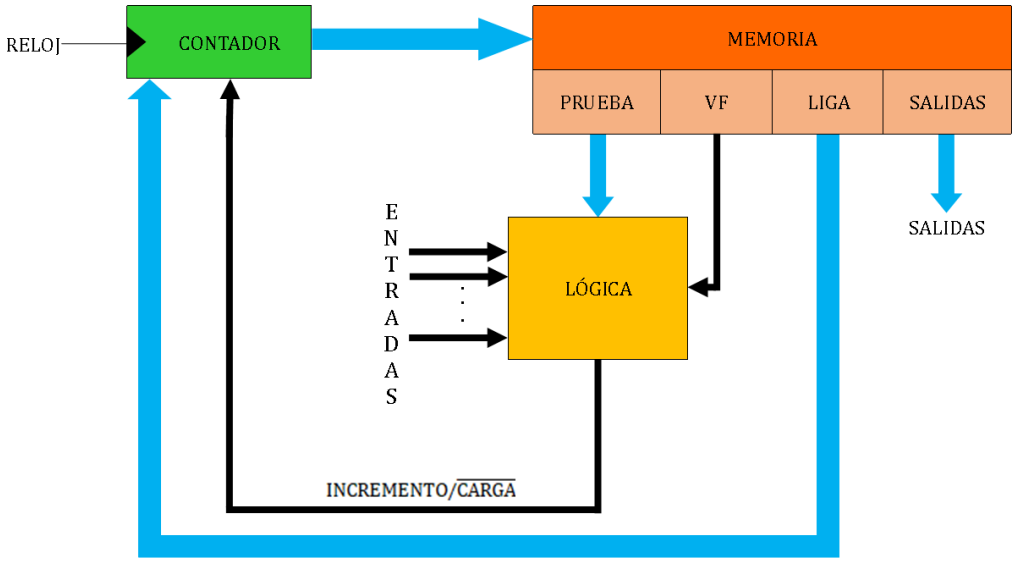


Figura P9.3 Arquitectura de un controlador con memoria y direccionamiento implícito.

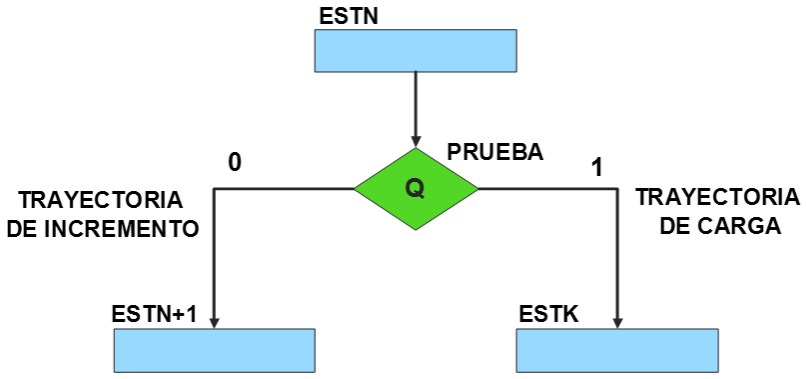


Figura P9.4 Trayectoria de incremento y carga.

La tabla P9.1 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que valor de la entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un '1' se hace un incremento, cuando el resultado de la función da como resultado un '0' se hace una carga.

Tabla P9.1 Relación entre VF, la variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incremento o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es '0', se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es '1', se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura P9.5).

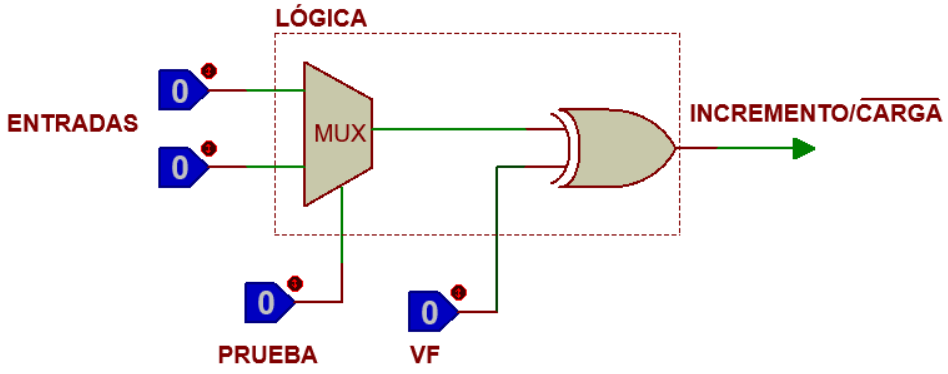


Figura P9.5 Bloque de lógica de incremento/carga para el direccionamiento implícito [1].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

Objetivo

Diseñar un controlador para el sistema de recolección de boletos por medio del método de diseño con memoria y direccionamiento implícito.

Descripción

Primero se diseña una carta ASM para el sistema de recolección de boletos por medio del método de diseño con memoria y direccionamiento implícito. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P9.2 se muestran los detalles de las entradas y salidas de este controlador.

Para la recolección de boletos se necesita señales de entrada:

- un sensor detecta la presencia de vehículo
- un sensor indica cuándo se esté ingresando un boleto y cuando este haya ingresado totalmente (ver figura P9.2).

Como salida se requiere de las siguientes señales:

- activación del motor que recolecta el boleto.

Tabla P9.2 Entradas y salidas para el sistema de recolección de boletos.

Identificador	Ubicación	Tipo	Descripción
CLK	A0 del PIC	I	Señal de reloj
S_S1	A2 del PIC	I	Sensor de salida que detecta cuando un vehículo llega a la salida del estacionamiento
S_ING	A3 del PIC	I	Sensor de ingreso y recolección de boleto
M_RBO	D0 de la memoria	O	Motor que recolectará el boleto.

Notas de diseño

a) El sensor S_S1 está ubicado antes del cruce de la pluma

b) El sensor S_ING tiene dos funciones: detectar cuándo se está ingresando un boleto y cuándo el boleto ha sido recolectado totalmente.

Reglas de funcionamiento

- S_S1: sensor de salida

  1 = detecta vehículo

  0 = no detecta vehículo

- S_ING: sensor de ingreso

  1 = detecta ingreso de boleto o detecta boleto no recolectado completamente

  0 = no detecta ingreso de boleto o detecta boleto recolectado completamente.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Se hace un incremento cuando el valor del estado siguiente aumenta en una unidad, de lo contrario se hace una carga (ver figura P9.4). El algoritmo de la máquina de estados se puede ver en la figura P9.6.

Estado '00' – SALIDA

En el primer estado, el controlador de la recolección de boletos se encuentra en espera. El sensor (S_S1), al detectar un vehículo, permite al sistema pasar al Estado '01'. De lo contrario, permanece en el Estado '00'.

Estado '01' – INTB

En este estado, se espera a que un boleto sea introducido. Si el sensor (S_ING) detecta que se introduce un boleto, el sistema avanza al Estado '10' para recolectar el boleto. De lo contrario, regresa al Estado '00' para verificar si aún hay un vehículo presente.

Estado '10' – RBOL

En este estado se activa el motor (M_RBO) para recolectar un boleto. Cuando el sensor (S_ING) detecta que el boleto ha sido recolectado en su totalidad, el sistema regresa al Estado '11', para posteriormente iniciar nuevamente el proceso de recolección del boleto. De lo contrario, permanece en el Estado '10' recolectando el boleto.

Estado '11' – AUX1

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada avanzar al Estado '00', haciendo una carga, para iniciar nuevamente el proceso de recolección de boleto.

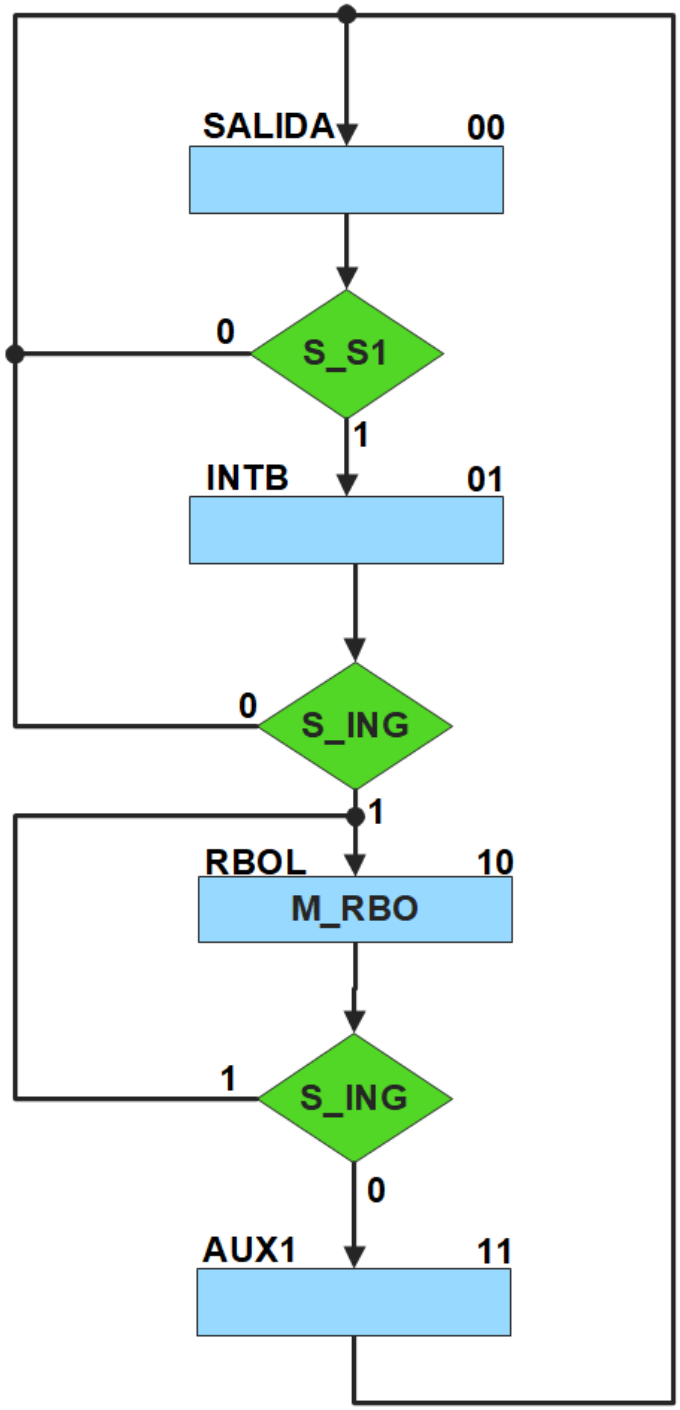


Figura P9.6 Carta ASM del sistema de recolección de boletos.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P9.3).

Tabla P9.3 Representación binaria de entradas para el sistema de recolección de boletos.

Entrada	Prueba
AUX	0 0
S_S1	0 1
S_ING	1 0

Se debe llenar la tabla P9.4 con base en la información de la carta ASM de la figura P9.6, usando el método de diseño con memoria y direccionamiento implícito.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘00’.

En el Estado ‘00’ se selecciona la entrada S_S1, por lo tanto, se coloca en el campo de prueba su representación binaria, es decir, ‘01’. Si S_S1 es igual a cero, entonces el estado siguiente es el Estado ‘00’, su representación binaria ‘00’ es colocada en el campo de la liga, ya que se requiere hace una carga. El campo VF es igual cero, ya que, para hacer una carga en el contador, el valor de la entrada y de VF deben ser iguales. En el Estado ‘00’ el motor M_RBO no está activado, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P9.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P9.4 Contenido de la memoria para el sistema de recolección de boletos.

Dirección de memoria		Contenido de memoria						
Estado presente		Prueba		VF	Liga		Salidas	Hex
QA	QB	I1	I0		L1	L0	M_RBO	
0	0	0	1	0	0	0	0	10
0	1	1	0	0	0	0	0	20
1	0	1	0	1	1	0	1	2D
1	1	0	0	1	0	0	0	08

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de cuatro líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P9.5). Se puede comprobar realizando el circuito de la figura P9.7.

Tabla P9.5 Tabla de funcionamiento del selector de entradas para el sistema de recolección de boletos.

Prueba		SI
I1	I0	
0	0	AUX
0	1	S_S1
1	0	S_ING

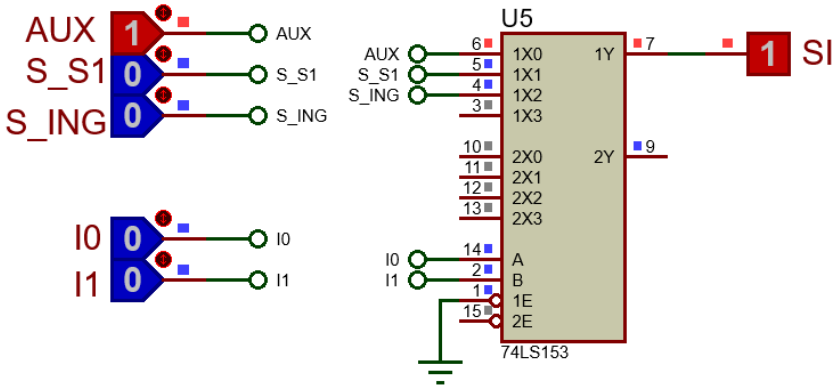


Figura P9.7 Multiplexor 74LS153 para el selector de entradas del sistema de recolección de boletos.

Por lo tanto, la función booleana del selector de entradas queda:

```
SI = AUX&!I1&!I0 | S_S1&!I1&I0 | S_ING&I1&!I0;
```

Para obtener el valor de la lógica, se debe hacer la operación XOR entre el selector de entradas y el valor de VF (ver figura P9.5).

Por lo tanto, la función booleana de la lógica queda:

```
SL = VF ^ (AUX&!I1&!I0 | S_S1&!I1&I0 | S_ING&I1&!I0);
```

Se utiliza un contador con carga paralela que indica el estado siguiente. El contador con carga paralela será implementado por el PIC16F1939. Si el valor a la salida de la lógica es igual a ‘1’, el contador carga la información binaria de la liga a la memoria. Si el valor de la lógica es igual a ‘0’, se cuenta al siguiente estado binario. A continuación, se obtienen las expresiones lógicas para un contador de dos bits por el método de variable suscrita (ver tabla P9.6).

Tabla P9.6 Mapa de Karnaugh general de transición de estados para un contador de dos bits.

A \ B	0	1
0	00	01
1	01	10
0	10	11
1	11	00

Para el bit A:

A \ B	0	1
0	0	1
1	1	0

```
FFA = A&!B | !A&B = A^B;
```

Para el bit B:

A \ B	0	1
0	1	0
1	1	0

```
FFB = !B;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P9.8,figura P9.9). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939.

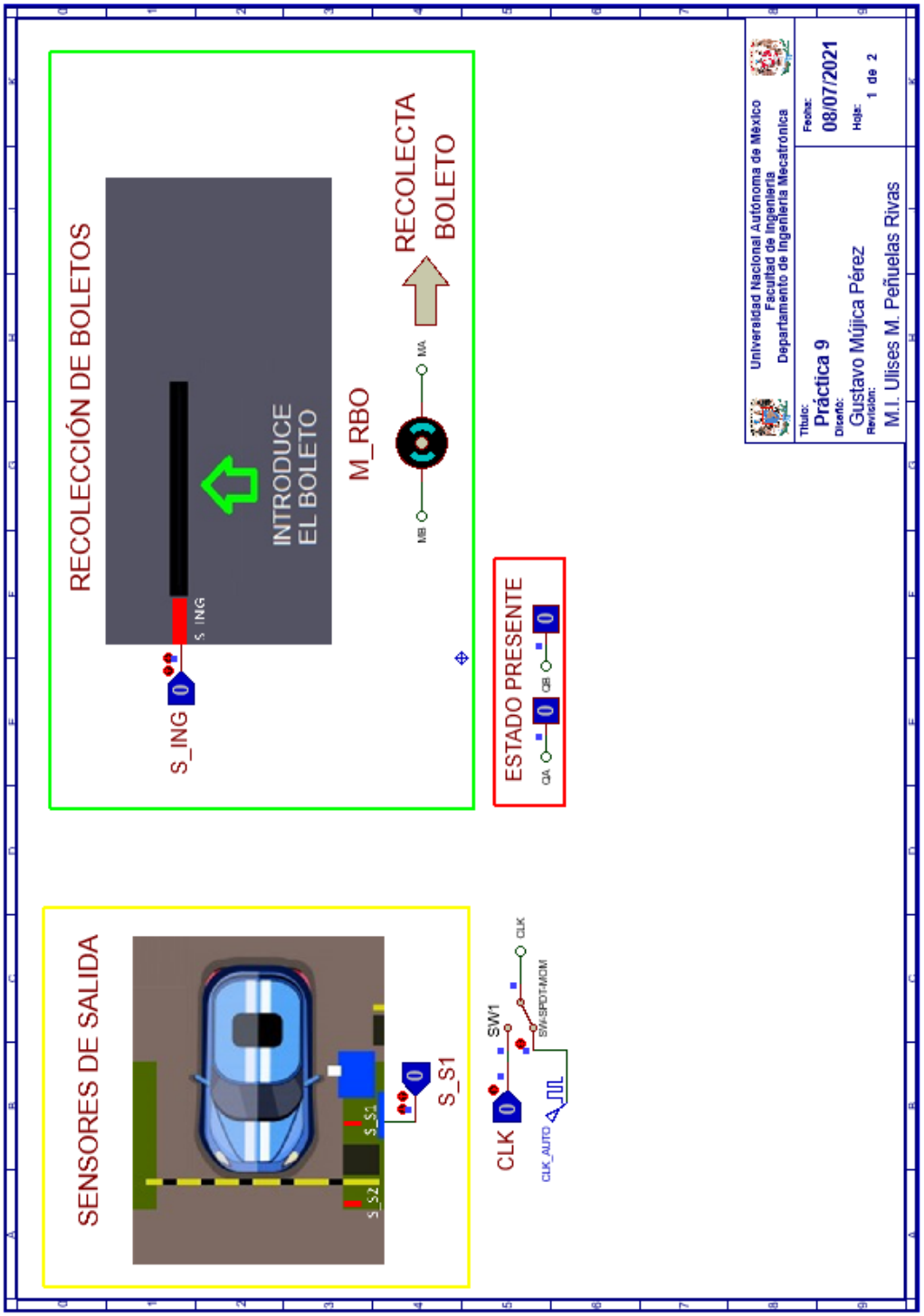


Figura P9.8 Interfaz hombre-máquina para el controlador de la Práctica 9 hoja 1/2.

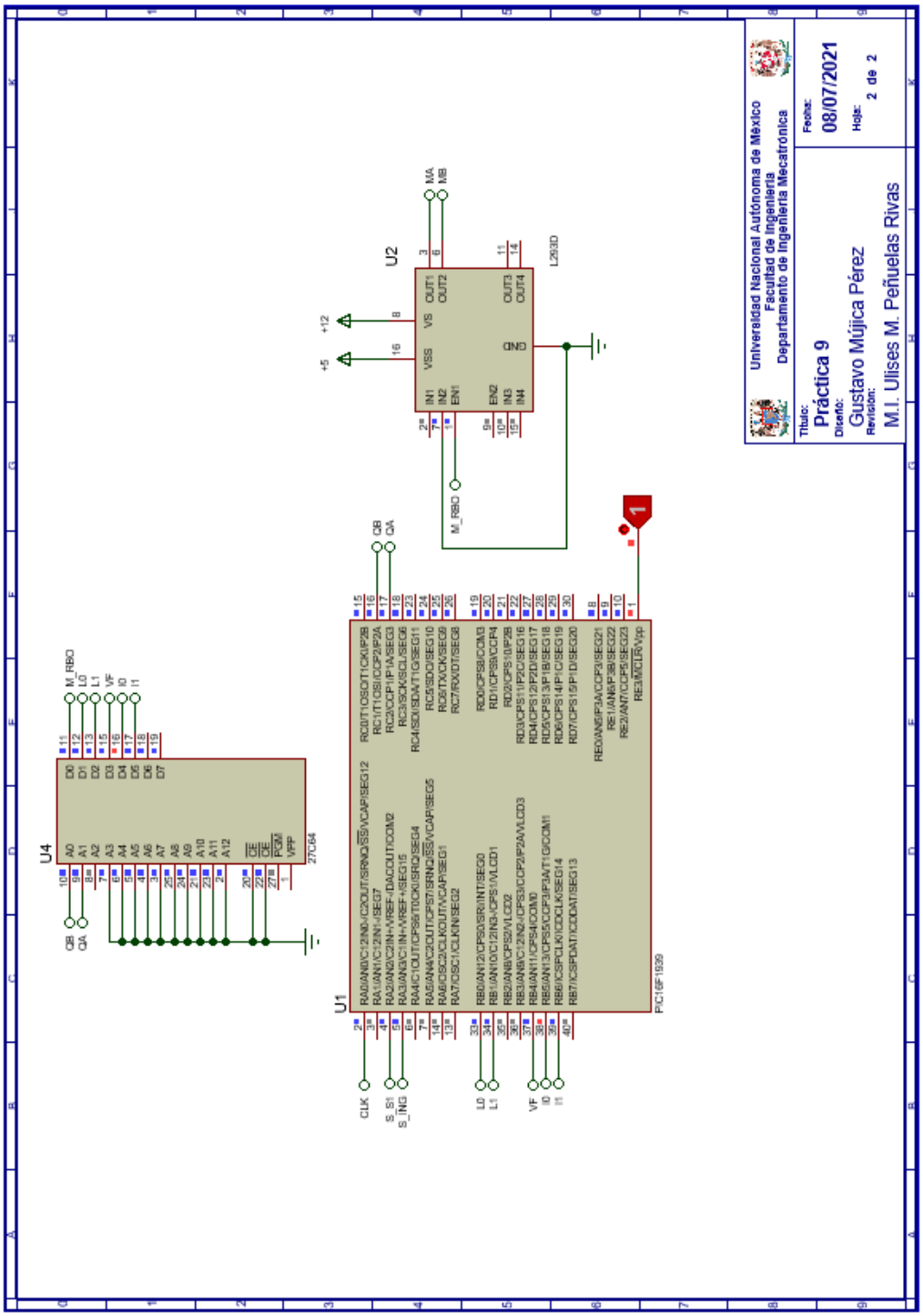


Figura P9.9 Esquema electrónico para el controlador de la Práctica 9 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P9.10, figura P9.11) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     // Carga biblioteca PLD.h

//***CONTADOR***

#define CLK A0 //RELOJ

#define A C2 //FFA
#define B C1 //FFB

//LIGA
#define L0 B0
#define L1 B1

//***LOGICA***

//ENTRADAS
#define S_S1  A2
#define S_ING A3

//VF
#define VF B4

//PRUEBAS
#define I0 B5
#define I1 B6

//VARIABLES INTERMEDIAS
short SL=0;  //Salida lógica
short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
short At=0, Bt=0; //Variables inntetrnas de los FF
short LD1,LD0; //Variables intermedias de la liga

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD


   //LOOP INFINITO
   while(1)
   {
   //CIRCUITO COMBINACIONAL
   LD1=L1; LD0=L0;              /*ALMACENA DATOS HASTA EL CAMBIO DEL RELOJ,
                                SIMULANDO UN REGISTRO, EVITANDO ASÍ
                                QUE SE MODIFIQUEN LOS VALORES DE LA MEMORIA*/

   //SALIDA LÓGICA
    SL= VF ^ (AUX&!I1&!I0 | S_S1&!I1&I0 | S_ING&I1&!I0);

   //CIRCUITO SECUENCIAL (CONTADOR DE CARGA PARALELA)
```

Figura P9.10 Código de la Práctica 9 parte 1.

```
50:     //CIRCUITO SECUENCIAL (CONTADOR DE CARGA PARALELA)
51:
52:     if (!CLK)
53:     {//SECCIÓN DE OPERACIONES DEL CONTADOR
54:     At= A^B;
55:     Bt= !B;
56:
57:     }
58:
59:     else
60:     {//CUENTA CON SL=1 Y CARGA CON SL=0
61:     A= SL&At | !SL&LD1;
62:     B= SL&Bt | !SL&LD0;
63:     while(clk){/*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
64:                 POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
65:                 DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
66:                 LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ*/}
67:     }
68:
69:     }
70:
71: }
```

Figura P9.11 Código de la Práctica 9 parte 2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 10 Sistema de entrada al estacionamiento; diseño con memoria y direccionamiento implícito

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P10.1).

Figura P10.1 Maqueta de estacionamiento con 10 lugares.

El sistema de entrada está compuesto por el sistema de emisión/retiro de boletos y el de la pluma para estacionamiento. En la entrada se debe adquirir un boleto para poder ingresar al estacionamiento. Se entrega un boleto de forma automática al presionar un botón, el boleto sale a través de una ranura. Una vez retirado el boleto, se levanta la pluma para permitir el paso de un vehículo y, cuando éste haya pasado completamente, se baja la pluma (ver figura P10.2).

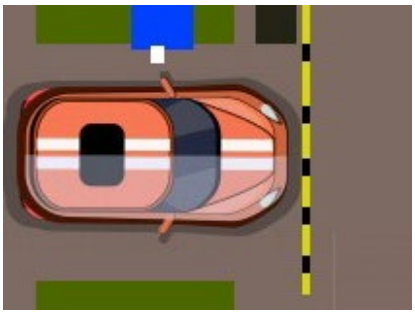


Figura P10.2 Entrada al estacionamiento.

El diseño con memorias y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura P10.3). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura P10.4 muestra la arquitectura de este método.

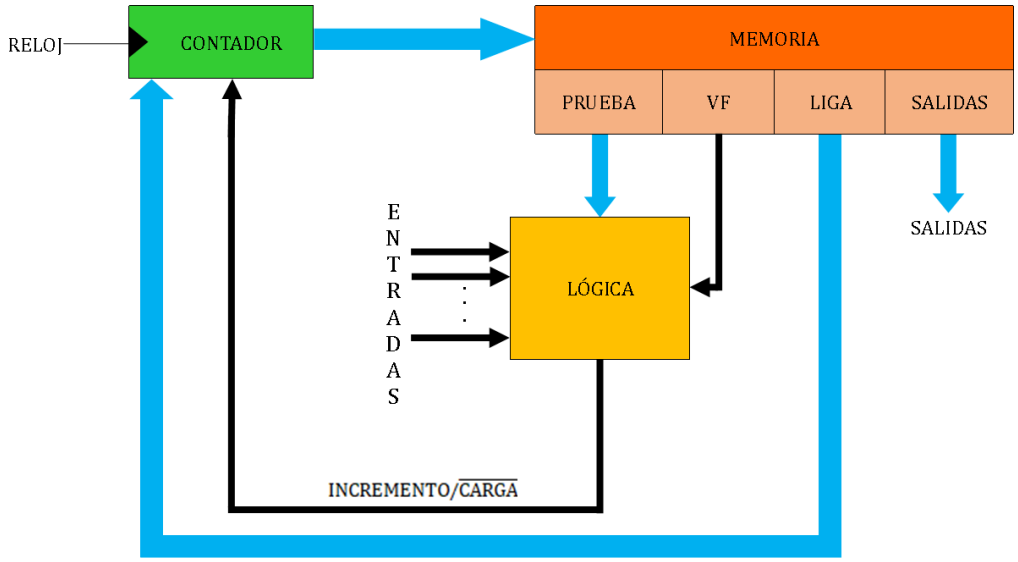


Figura P10.3 Arquitectura de un controlador con memoria y direccionamiento implícito.

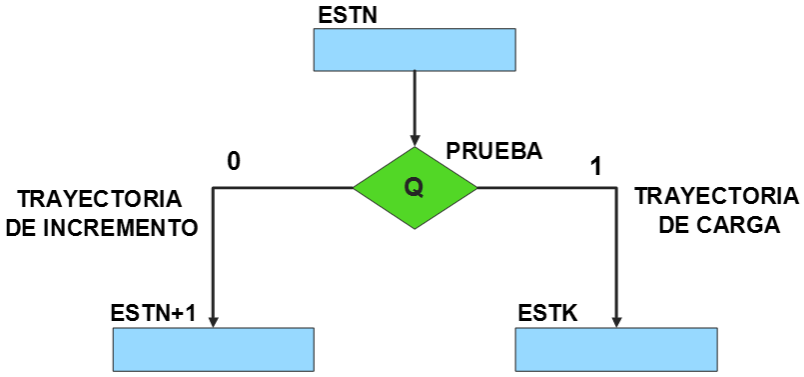


Figura P10.4 Trayectoria de incremento y carga.

La tabla P10.1 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que valor de entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un '1' se hace un incremento, cuando el resultado de la función da como resultado un '0' se hace una carga.

Tabla P10.1 Relación entre VF, la variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incremento o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es '0', se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es '1', se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura P10.5).

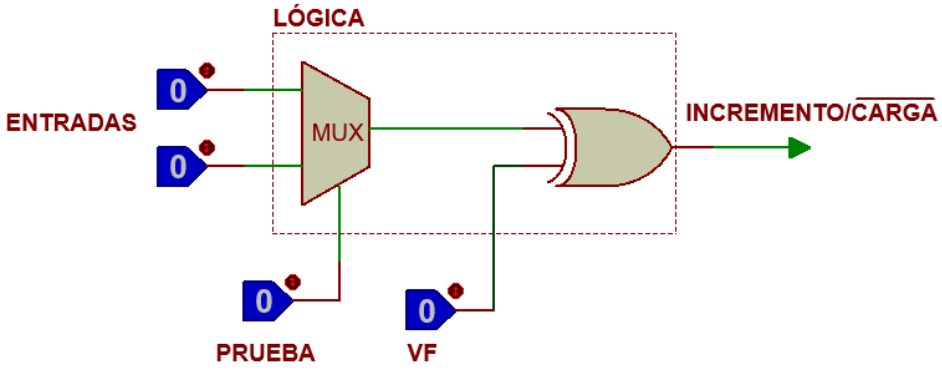


Figura P10.5 Bloque de lógica de incremento/carga para el direccionamiento implícito [1].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

Objetivo

Diseñar un controlador para el sistema de entrada por medio del método de diseño con memoria y direccionamiento implícito.

Descripción

Primero se diseña una carta ASM para el sistema de recolección de boletos por medio del método de diseño con memoria y direccionamiento implícito. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P10.2 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema de entrada se necesitan las siguientes señales:

- un sensor detecta la presencia de vehículo
- un botón detecta la solicitud de emisión de boleto
- un sensor verifica la emisión y, posteriormente, el retiro del boleto
- un sensor detecta cuándo la pluma se encuentra en la posición superior y otro cuando se encuentra en la posición inferior
- un sensor detecta cuándo un vehículo está a la entrada del estacionamiento y otro cuando se esté ingresando a éste.

Como salida se requiere de las siguientes señales:

- activación del motor de emisión del boleto
- una señal activa el motor de la pluma
- una señal se activa para que la pluma suba y otra para que la pluma baje.

Tabla P10.2 Entradas y salidas para el sistema de entrada al estacionamiento.

Identificador	Ubicación	Tipo	Descripción
CLK	A0 del PIC	I	Señal de reloj
S_E1	A1 del PIC	I	Sensor de entrada que detecta cuándo un vehículo llega a la entrada del estacionamiento
BOT	A2 del PIC	I	Botón para obtener un boleto
S_RET	A3 del PIC	I	Sensor de emisión y retiro de boleto
S_E2	A4 del PIC	I	Sensor de Ingreso que detecta cuándo un vehículo está ingresando al estacionamiento
S_SUP1	A5 del PIC	I	Sensor de posición superior que detecta cuándo la pluma se ha elevado en su totalidad
S_INF1	A6 del PIC	I	Sensor de posición inferior que detecta cuándo la pluma se ha bajado en su totalidad
M_EBO	D3 de la memoria 2	O	Motor que emitirá el boleto
MOT_1	D2 de la memoria 2	O	Motor de la pluma
MX_1	D1 de la memoria 2	O	Señal para que la pluma suba
nMX_1	D0 de la memoria 2	O	Señal para que la pluma baje.

Notas de diseño

a) El sensor S_E1 está ubicado antes del cruce de la pluma

b) El sensor S_RET tiene dos funciones: detectar cuándo el boleto ha sido emitido y cuándo ha sido retirado

c) El sensor S_E2 está ubicado después del cruce de la pluma.

Reglas de funcionamiento

- S_E1: sensor de entrada
  1 = detecta vehículo
  0 = no detecta vehículo
- BOT: botón para obtener un boleto
  1 = se presionó el botón
  0 = no se presionó el botón
- S_RET: sensor de emisión y retiro de boleto
  1 = detecta que se ha emitido el boleto o detecta que el boleto no ha sido retirado
  0 = detecta que no se ha emitido el boleto o detecta que el boleto ha sido retirado
- S_E2: sensor de ingreso
  1 = detecta vehículo ingresando
  0 = no detecta vehículo ingresando
- S_SUP1: sensor de posición superior
  1 = detecta pluma completamente elevada
  0 = no detecta pluma completamente elevada
- S_INF1: sensor de posición inferior
  1 = detecta que la pluma ha bajado completamente
  0 = no detecta que la pluma ha bajado completamente.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Se hace un incremento cuando el valor del estado siguiente aumenta en una unidad, de lo contrario se hace una carga (ver figura P10.4). El algoritmo de la máquina de estados se puede ver en la figura P10.6.

Estado ‘0000’ – ENTRADA

Entrada del vehículo. En el primer estado, el controlador de la emisión de boletos se encuentra en espera. El sensor (S_E1), al detectar un vehículo, permite al sistema pasar al Estado ‘0001’. De lo contrario, permanece en el Estado ‘0000’.

Estado ‘0001’ – EBOT

En este estado se espera a que se presione el botón (BOT) para la emisión del boleto. Al ser presionado el botón, el sistema avanza al Estado ‘0010’ para emitir el boleto. De lo contrario, regresa al Estado ‘0000’ para verificar si aún hay un vehículo presente.

Estado ‘0010’ – EBOL

Emisión de boleto. En este estado se activa el motor correspondiente (M_EBO) para que se emita un boleto. Cuando el sensor (S_RET) detecta que el boleto ya se encuentra listo para retirarse, el sistema pasa al Estado ‘0011’ donde se desactiva al motor y espera a que se retire el boleto. Si el sensor aún no detecta el boleto, permanece en el Estado ‘0010’.

Estado ‘0011’ – ERET

En este estado espera a que el boleto sea retirado. Cuando el sensor (S_RET) detecta que el boleto se ha retirado, avanza al Estado ‘0100’, para levantar la pluma permitiendo el ingreso del vehículo. De lo contrario, permanece en el estado actual, en espera a que el boleto sea retirado.

Estado ‘0100’ – ELEVA

En este estado, se activa el motor (MOT_1) y la señal (MX_1) para que la pluma suba, permitiendo el ingreso. Si el sensor (S_SUP1) detecta que la pluma ha sido elevada completamente, el sistema avanza al Estado ‘0101’ para esperar a que pase el vehículo. De lo contrario avanza al Estado ‘1000’ para verificar si hay un vehículo a la entrada.

Estado ‘0101’ – ESPERA

En este estado, la pluma está elevada, esperando a que el vehículo pase completamente. Si el sensor (S_E1) no detecta un vehículo a la entrada, el sistema avanza al Estado ‘0110’ para revisar si un vehículo está ingresando. De lo contrario, permanece en el Estado ‘0101’.

Estado ‘0110’ – AUX1

Si el sensor (S_E2) no detecta un vehículo ingresando, avanza al Estado ‘0111’ para que posteriormente baje la pluma ya que el vehículo ha pasado completamente. De lo contrario, el sistema regresa al Estado ‘0101’ para esperar a que el vehículo pase completamente.

Estado ‘0111’ – AUX2

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados sea forzada a avanzar al Estado ‘1010’, haciendo una carga, para que se baje la pluma, ya que el vehículo ha pasado completamente.

Estado ‘1000’ – AUX3

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (MX_1) para que la pluma suba. Cuando el sensor (S_E1) no detecta vehículo a la entrada, el sistema avanza al Estado ‘1001’ para revisar si hay un vehículo ingresando. De lo contrario, regresa al Estado ‘0000’ para seguir elevando la pluma.

Estado ‘1001’ – AUX4

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (MX_1) para que la pluma suba. Si el sensor (S_E2) no detecta un vehículo ingresando, el sistema avanza al Estado ‘1010’ para bajar la pluma, debido a que el auto ha pasado antes de que la pluma se eleve completamente. De lo contrario, regresa al Estado ‘0000’ para seguir elevando la pluma.

Estado ‘1010’ – BAJA

En este estado se activa el motor (MOT_1) y la señal (nMX_1) para que la pluma baje. Si el sensor (S_INF1) detecta que la pluma no ha bajado en su totalidad, avanza al Estado ‘1011’ para revisar si hay un vehículo a la entrada. De lo contrario, regresa al Estado ‘0000’ para verificar si algún vehículo requiere ingresar al estacionamiento.

Estado ‘1011’ – AUX5

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (nMX_1) para que la pluma siga bajando. Si el sensor (S_E1) no detecta un vehículo a la entrada, el sistema avanza al Estado ‘1100’ para revisar si hay un auto ingresando. De lo contrario, regresa al Estado ’0100’ para subir la pluma y que ésta no dañe al vehículo.

Estado ‘1100’ – AUX6

En este estado continúa estando activa la señal del motor (MOT_1) y la señal (nMX_1) para que la pluma siga bajando. Si el sensor (S_E2) no detecta un vehículo entrando, el sistema regresa al Estado ‘1010’ para seguir bajando la pluma. De lo contrario, avanza al Estado ‘1101’ para subir la pluma y que ésta no dañe al vehículo.

Estado ‘1101’ – AUX7

En este estado se activa la señal del motor (MOT_1) y la señal (MX_1) para que la pluma suba y no dañe al vehículo. En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados sea forzada avanzar al Estado ‘0100’, haciendo una carga, para que la pluma siga subiendo y no dañe al vehículo.

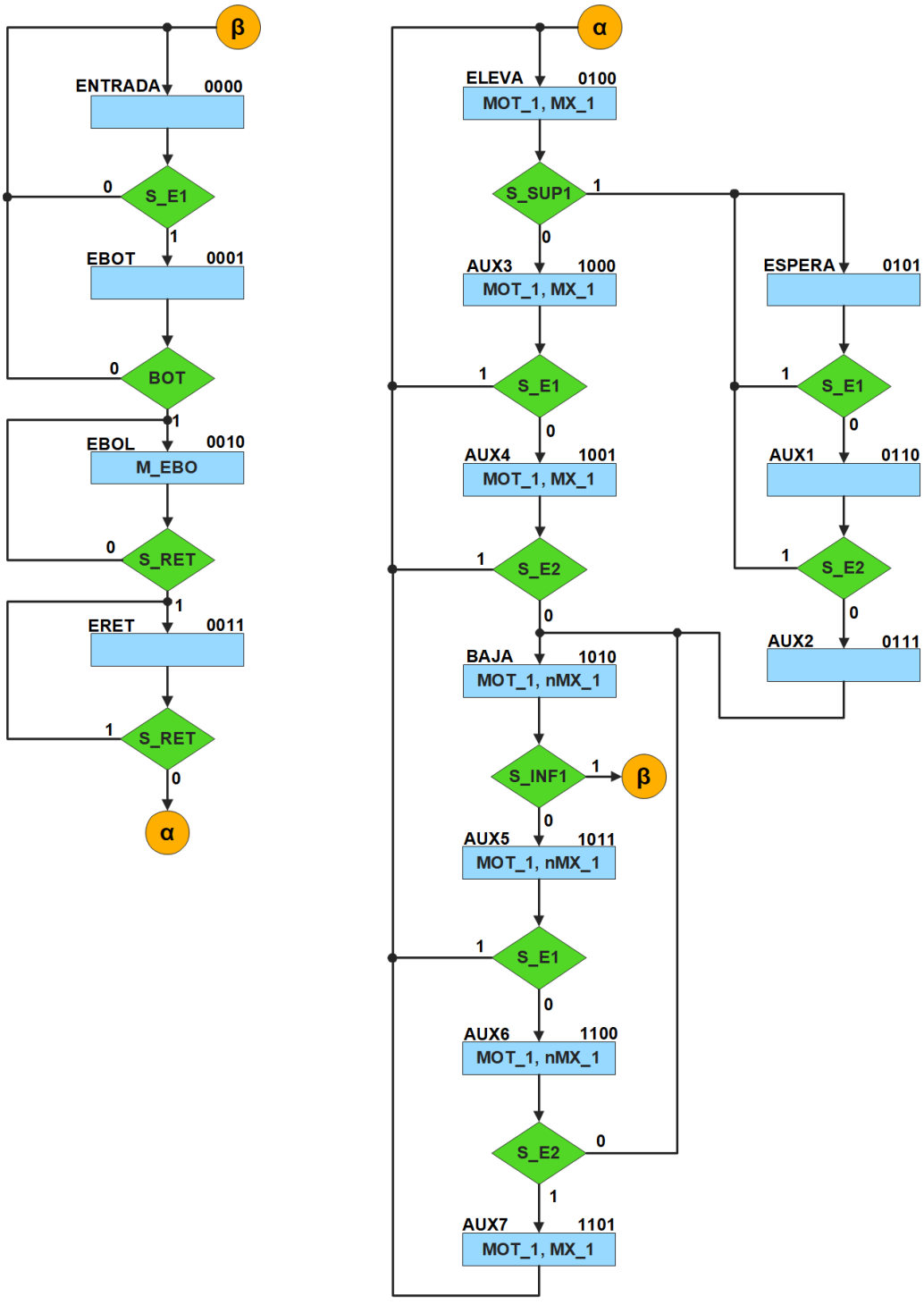


Figura P10.6 Carta ASM del sistema de entrada al estacionamiento.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P10.3).

Tabla P10.3 Representación binaria de entradas para el sistema de entrada al estacionamiento.

Entrada	Prueba
AUX	0 0 0
S_E1	0 0 1
BOT	0 1 0
S_RET	0 1 1
S_E2	1 0 0
S_SUP1	1 0 1
S_INF1	1 1 0

Se debe llenar la tabla P10.4 con base en la información de la carta ASM de la figura P10.6, usando el método de diseño con memoria y direccionamiento implícito.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘0000’.

En el Estado ‘0000’ se selecciona la entrada S_E1, por lo tanto, se coloca en el campo de prueba su representación binaria, es decir, ‘001’. Si S_E1 es igual a cero, entonces el estado siguiente es el Estado ‘0000’, su representación binaria ‘0000’ es colocada en el campo de la liga, ya que se requiere hacer una carga. El campo VF es igual cero, ya que, para hacer una carga en el contador, el valor de la entrada y de VF deben ser iguales. En el Estado ‘0000’ no hay salidas activadas, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria, ver figura P10.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P10.4 Contenido de la memoria para el sistema de entrada al estacionamiento.

Dirección de memoria				Contenido de memoria													
Estado presente				Prueba			VF	Liga				Salidas				Hex 1	Hex 2
QA	QB	QC	QD	I2	I1	I0		L3	L2	L1	L0	M_EBO	MOT_1	MX_1	nMX_1		
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	20	00
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	40	00
0	0	1	0	0	1	1	0	0	0	1	0	1	0	0	0	62	08
0	0	1	1	0	1	1	1	0	0	1	1	0	0	0	0	73	00
0	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	A8	06
0	1	0	1	0	0	1	1	0	1	0	1	0	0	0	0	35	00
0	1	1	0	1	0	0	1	0	1	0	1	0	0	0	0	95	00
0	1	1	1	0	0	0	1	1	0	1	0	0	0	0	0	1A	00
1	0	0	0	0	0	1	1	0	1	0	0	0	1	1	0	34	06
1	0	0	1	1	0	0	1	0	1	0	0	0	1	1	0	94	06
1	0	1	0	1	1	0	1	0	0	0	0	0	1	0	1	D0	05
1	0	1	1	0	0	1	1	0	1	0	0	0	1	0	1	34	05
1	1	0	0	1	0	0	0	1	0	1	0	0	1	0	1	8A	05
1	1	0	1	0	0	0	1	0	1	0	0	0	1	1	0	14	06

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de ocho líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P10.5). Se puede comprobar realizando el circuito de la figura P10.7.

Tabla P10.5 Tabla de funcionamiento del selector de entradas para el sistema de entrada al estacionamiento.

Prueba			SI
I2	I1	I0	
0	0	0	AUX
0	0	1	S_E1
0	1	0	BOT
0	1	1	S_RET
1	0	0	S_E2
1	0	1	S_SUP1
1	1	0	S_INF1

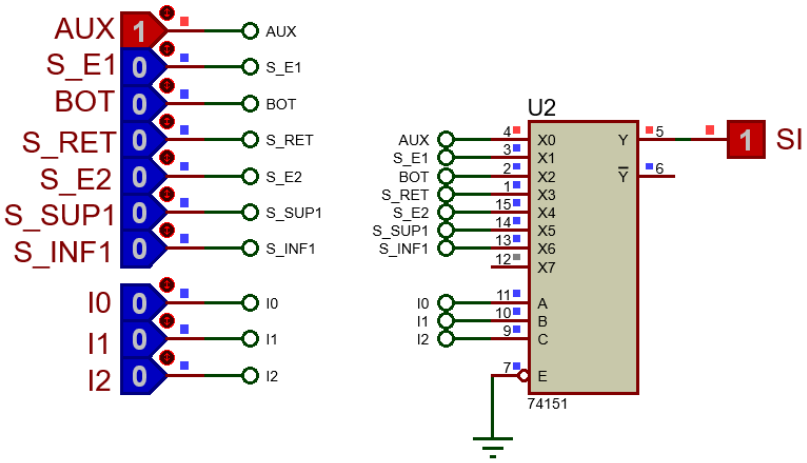


Figura P10.7 Multiplexor 74151 para el selector de entradas para el sistema de entrada al estacionamiento.

La función booleana del selector de entradas queda:

```
SI = AUX&!I2&!I1&!I0| S_E1&!I2&!I1&I0 | BOT&!I2&I1&!I0 | S_RET&!I2&I1&I0
    | S_E2&I2&!I1&!I0 | S_SUP1&I2&!I1&I0 | S_INF1&I2&I1&!I0;
```

Para obtener el valor de la lógica, se debe hacer la operación XOR entre el selector de entradas y el valor de VF (ver figura P10.5).

Por lo tanto, la función booleana de la lógica queda:

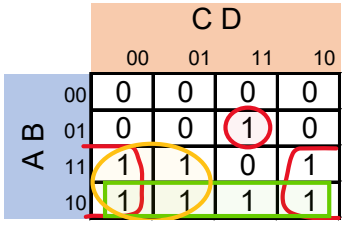
```
SL = VF ^ (AUX&!I2&!I1&!I0| S_E1&!I2&!I1&I0 | BOT&!I2&I1&!I0 |
     S_RET&!I2&I1&I0 | S_E2&I2&!I1&!I0 | S_SUP1&I2&!I1&I0 |
     S_INF1&I2&I1&!I0);
```

Se utiliza un contador con carga paralela que indica el estado siguiente. El contador con carga paralela será implementado por el PIC16F1939. Si el valor a la salida de la lógica es igual a '1', el contador carga la información binaria de la liga a la memoria. Si el valor de la lógica es igual a '0', se cuenta al siguiente estado binario. A continuación, se obtienen las expresiones lógicas para un contador de cuatro bits por el método de variable suscrita (ver tabla P10.6).

Tabla P10.6 Mapa de Karnaugh general de transición de estados para un contador de cuatro bits.

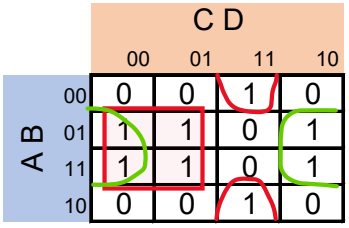
A B \ C D	0 0	0 1	1 1	1 0
0 0	0000	0001	0011	0010
	0001	0010	0100	0011
0 1	0100	0101	0111	0110
	0101	0110	1000	0111
1 1	1100	1101	1111	1110
	1101	1110	0000	1111
1 0	1000	1001	1011	1010
	1001	1010	1100	1011

Para el bit A:




```
FFA = A&!C | A&!B | A&!D |
      !A&B&C&D;
```

Para el bit B:




```
FFB = B&!C | B&!D | !B&C&D;
```

Para el bit C:

A B \ C D	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

```
FFC = !C&D | C&!D;
```

Para el bit D:

A B \ C D	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

```
FFD = !D;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P10.8, figura P10.9). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939.

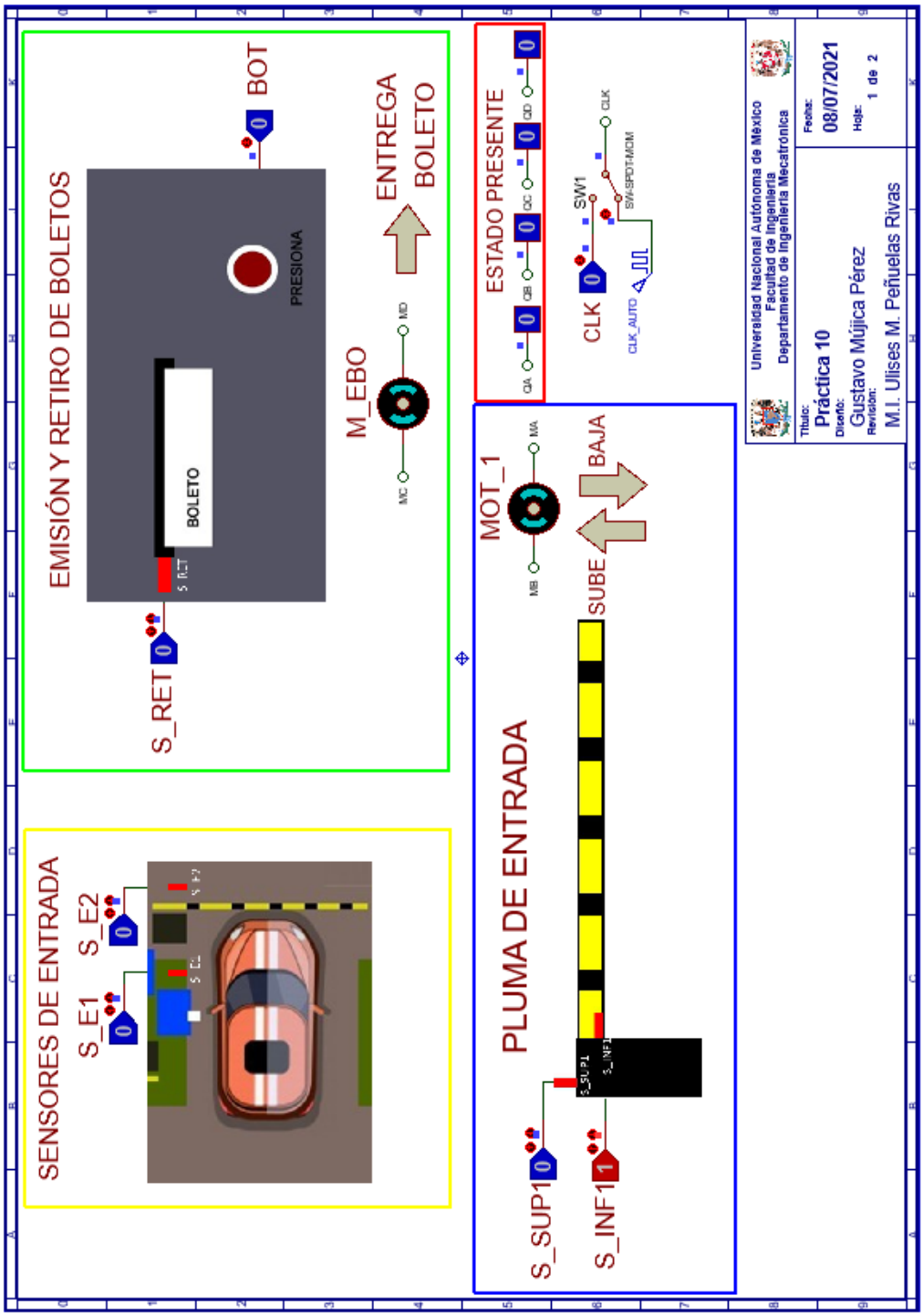


Figura P10.8 Interfaz hombre-máquina para el controlador de la Práctica 10 hoja 1/2.

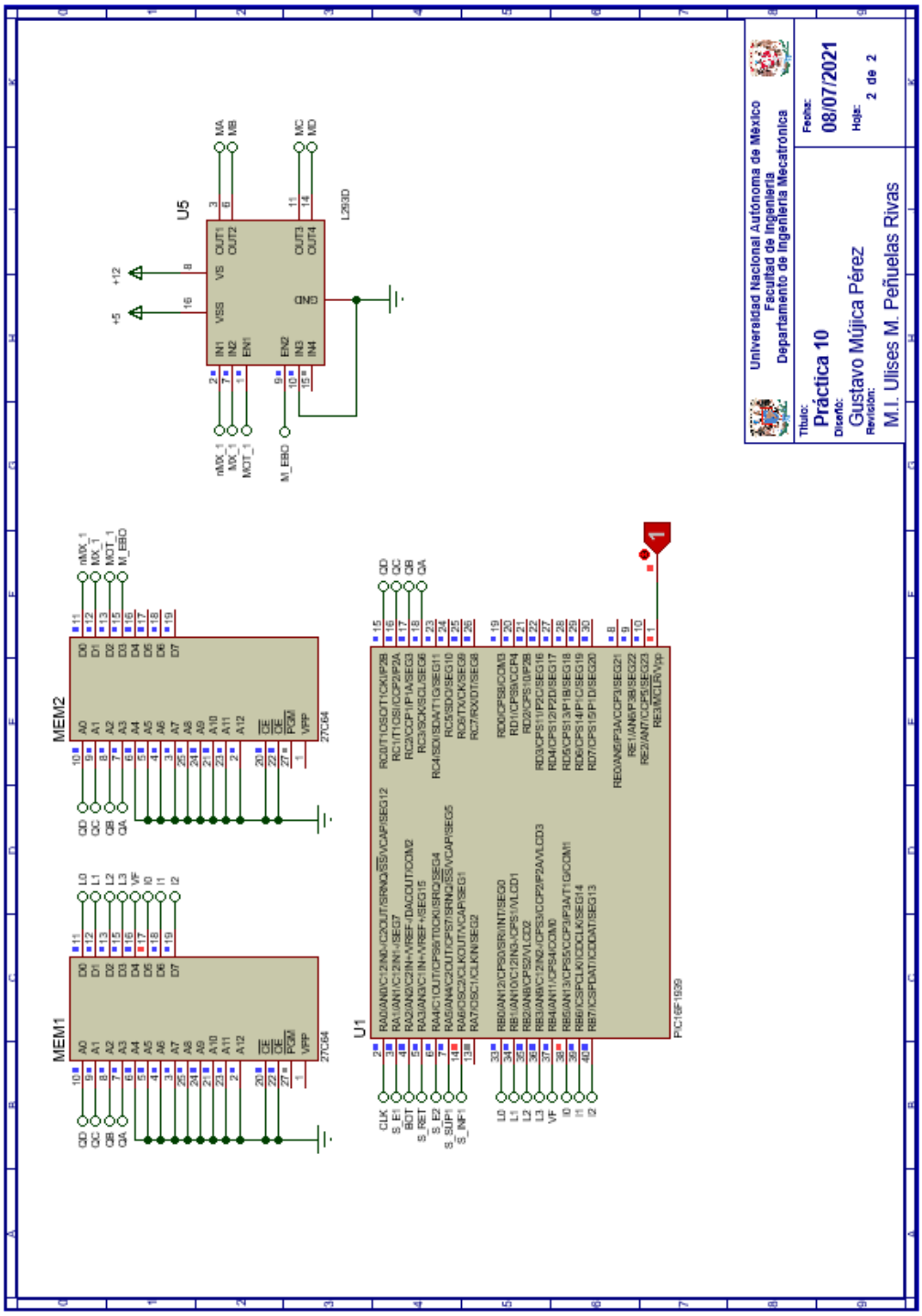


Figura P10.9 Esquema electrónico para el controlador de la Práctica 10 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P10.10,figura P10.11) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     // Carga biblioteca PLD.h

//***CONTADOR***

#define CLK A0 //RELOJ
//ENTRADAS

//LIGA
#define L0 B0
#define L1 B1
#define L2 B2
#define L3 B3

//SALIDAS
#define A C3 //FFA
#define B C2 //FFB
#define C C1 //FFC
#define D C0 //FFD

//***LOGICA***

//ENTRADAS
#define S_E1   A1
#define BOT    A2
#define S_RET  A3
#define S_E2   A4
#define S_SUP1 A5
#define S_INF1 A6

//VF
#define VF B4

//PRUEBAS
#define I0 B5
#define I1 B6
#define I2 B7

//VARIABLES INTERMEDIAS
short SL=0;  //Salida lógica
short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
short At=0, Bt=0, Ct=0, Dt=0; //Variables inntetrnas de los FF
short LD3,LD2,LD1,LD0; //Variables intermedias de la liga

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD
//pld_555(10);    // Genera señal cuadrada en Hz,
                  //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO

```

Figura P10.10 Código de la Práctica 10 parte 1.

```
50:
51:     //LOOP INFINITO
52:     while(1)
53:     {
54:     //CIRCUITO COMBINACIONAL
55:     LD3=L3; LD2=L2; LD1=L1; LD0=L0;      /*ALMACENA DATOS HASTA EL CAMBIO DEL RELOJ,
56:                                           SIMULANDO UN REGISTRO, EVITANDO ASÍ
57:                                           QUE SE MODIFIQUEN LOS VALORES DE LA MEMORIA*/
58:
59:     //SALIDA LÓGICA
60:     SL = VF ^ (AUX&!I2&!I1&!I0| S_E1&!I2&!I1&I0 | BOT&!I2&I1&!I0 | S_RET&!I2&I1&I0 |
61:          S_E2&I2&!I1&!I0 | S_SUP1&I2&!I1&I0 | S_INF1&I2&I1&!I0);
62:
63:     //CIRCUITO SECUENCIAL (CONTADOR DE CARGA PARALELA)
64:
65:       if (!CLK) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO
66:    // if (!out_555) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO,
67:                     //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
68:
69:     {//SECCIÓN DE OPERACIONES DEL CONTADOR
70:     At= A&!C | A&!B | A&!D | !A&B&C&D;
71:     Bt= B&!C | B&!D | !B&C&D;
72:     Ct= C^D;
73:     Dt= !D;
74:
75:     }
76:
77:     else
78:     {//SECCIÓN DE MEMORIZACIÓN
79:     //CUENTA CON SL=1 Y CARGA CON SL=0
80:     A= SL&At | !SL&LD3;
81:     B= SL&Bt | !SL&LD2;
82:     C= SL&Ct | !SL&LD1;
83:     D= SL&Dt | !SL&LD0;
84:
85:     while(clk){}/*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
86:                 POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
87:                 DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
88:                 LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ*/
89:
90:    /* while(out_555){} /*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
91:                     POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
92:                     DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
93:                     LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ,
94:                     EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO*/
95:     }
96:
97:     }
98:
99: }
```

Figura P10.11 Código de la Práctica 10 parte 2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 11 Sistema de salida del estacionamiento; diseño con memoria y direccionamiento implícito

Introducción

Por medio de una maqueta, se simuló el funcionamiento de un estacionamiento público que emite boletos, se desarrollaron sistemas que controlan la entrada y salida de éste (ver figura P11.1).

Figura P11.1 Maqueta de estacionamiento con 10 lugares.

Este sistema está compuesto por el sistema de recolección de boletos y el de una pluma para estacionamiento. A la salida se debe recolectar el boleto para poder abandonar el estacionamiento. Se recolecta el boleto de forma automática al introducirlo en una ranura. Una vez ingresado el boleto completamente, se levanta la pluma para permitir el paso de un vehículo y, cuando éste haya pasado completamente, se baja la pluma (ver figura P11.2).

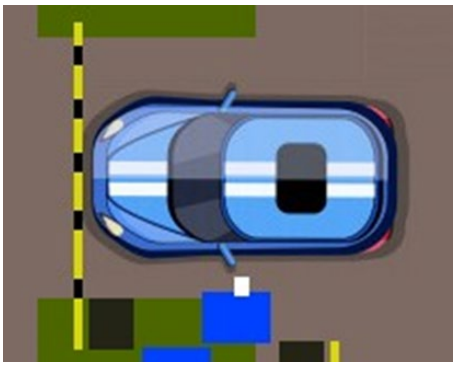


Figura P11.2 Salida del estacionamiento.

El diseño con memoria y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura P11.3). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura P11.4 muestra la arquitectura de este método.

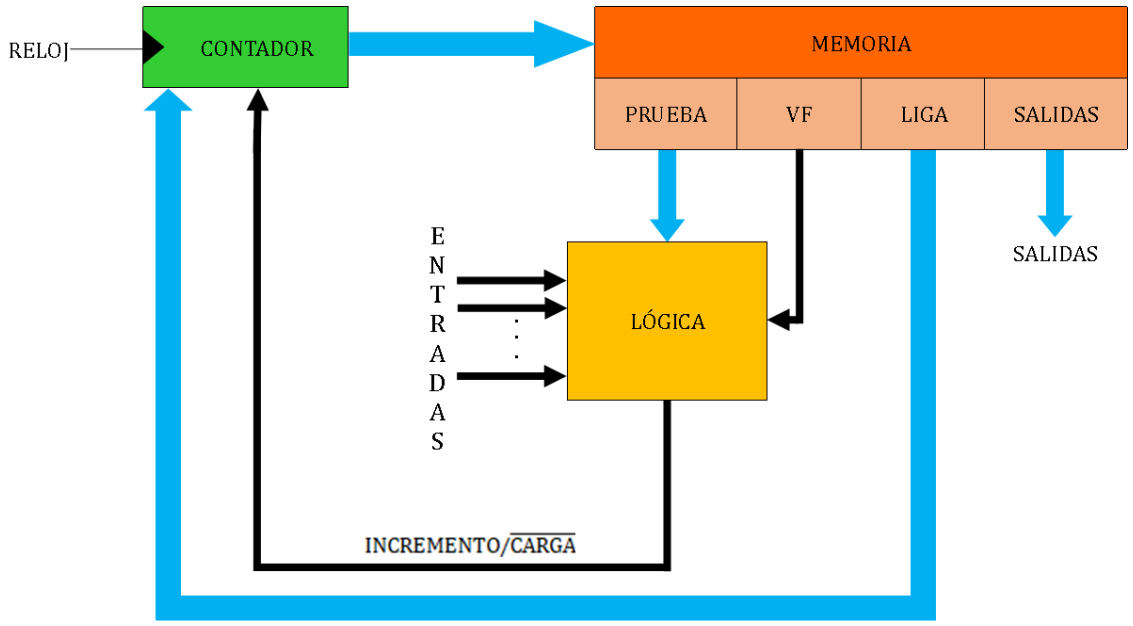


Figura P11.3 Arquitectura de un controlador con memoria y direccionamiento implícito.

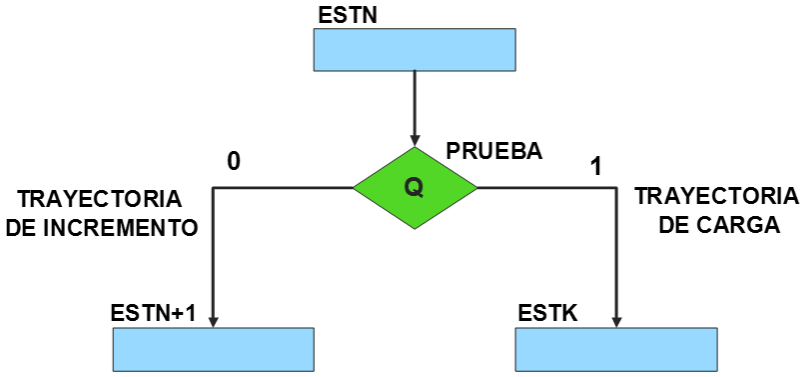


Figura P11.4 Trayectoria de incremento y carga.

La tabla P11.1 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que valor de entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un '1' se hace un incremento, cuando el resultado de la función da como resultado un '0' se hace una carga.

Tabla P11.1 Relación entre VF, variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incremento o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es '0', se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es '1', se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura P11.5).

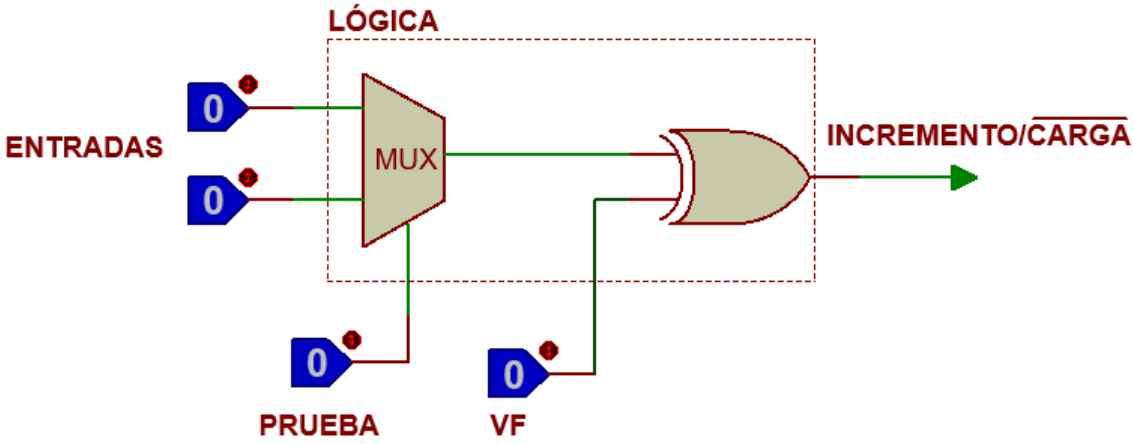


Figura P11.5 Bloque de lógica incremento/carga para el direccionamiento implícito [1].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

Objetivo

Diseñar un controlador para el sistema de salida por medio del método de diseño con memoria y direccionamiento implícito.

Descripción

Primero se diseña una carta ASM para el sistema de salida por medio del método de diseño con memoria y direccionamiento implícito. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P11.2 se muestran los detalles de las entradas y salidas de este controlador.

Para la pluma del estacionamiento se necesitan las siguientes señales de entrada:

- un sensor detecta la presencia de vehículo
- un sensor indica cuándo se esté ingresando un boleto y cuándo este haya ingresado totalmente
- un sensor detecta cuándo la pluma se encuentra en la posición superior y otro cuándo se encuentra en la posición inferior
- un sensor detecta cuándo un vehículo está a la entrada del estacionamiento y otro cuándo se esté ingresando a éste.

Como salida se requiere de las siguientes señales:

- activación del motor que recolecta el boleto
- una señal activa el motor de la pluma
- una señal se activa para que la pluma suba y otra para que la pluma baje.

Tabla P11.2 Entradas y salidas para el sistema de salida del estacionamiento.

Identificador	Ubicación	Tipo	Descripción
CLK	A0 del PIC	I	Señal de reloj
S_S1	A1 del PIC	I	Sensor de salida que detecta cuándo un vehículo llega a la salida del estacionamiento
S_ING	A2 del PIC	I	Sensor de ingreso
S_S2	A3 del PIC	I	Sensor de abandono que detecta cuándo un vehículo está abandonando al estacionamiento
S_SUP2	A4 del PIC	I	Sensor de posición superior que detecta cuándo la pluma se ha elevado en su totalidad
S_INF2	A5 del PIC	I	Sensor de posición inferior que detecta cuándo la pluma se ha bajado en su totalidad
M_RBO	D3 de la memoria 2	O	Motor que recolectará el boleto
MOT_2	D2 de la memoria 2	O	Motor de la pluma
MX_2	D1 de la memoria 2	O	Señal para que la pluma suba
nMX_2	D0 de la memoria 2	O	Señal para que la pluma baje.

Notas de diseño

a) El sensor S_S1 está ubicado antes del cruce de la pluma.

b) El sensor S_ING tendrá dos funciones: detectar cuando se está introduciendo un boleto y cuando el boleto haya sido ingresado totalmente.

c) El sensor S_S2 está ubicado después del cruce de la pluma.

Reglas de funcionamiento

- S_S1: sensor de salida

  1 = detecta vehículo

  0 = no detecta vehículo

- S_ING: sensor de ingreso

  1 = detecta ingreso de boleto o detecta boleto no recolectado completamente

  0 = no detecta ingreso de boleto o detecta boleto recolectado completamente

- S_E2: sensor de ingreso
  1 = detecta vehículo ingresando
  0 = no detecta vehículo ingresando
- S_SUP1: sensor de posición superior
  1 = detecta pluma completamente elevada
  0 = no detecta pluma completamente elevada
- S_INF1: sensor de posición inferior
  1 = detecta que la pluma ha bajado completamente
  0 = no detecta que la pluma ha bajado completamente.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Se hace un incremento cuando el valor del estado siguiente aumenta en una unidad, de lo contrario se hace una carga (ver figura P11.4). El algoritmo de la máquina de estados se puede ver en la figura P11.6.

Estado ‘0000’ – SALIDA

En el primer estado, el controlador de la recolección de boletos encuentra en espera. El sensor (S_S1), al detectar un vehículo, permite al sistema pasar al Estado ‘0001’. De lo contrario, permanece en el Estado ‘0000’.

Estado ‘0001’ – INTB

En este estado, se espera a que un boleto sea introducido. Si sensor (S_ING) detecta que se introduce un boleto, el sistema avanza al Estado ‘0010’ para recolectar el boleto. De lo contrario, regresa al Estado ‘0000’ para verificar si aún hay un vehículo presente.

Estado ‘0010’ – RBOL

En este estado se activa el motor (M_RBO) para recolectar un boleto. Cuando el sensor (S_ING) detecta que el boleto ha sido recolectado en su totalidad, el sistema avanza al Estado ‘0011’, para levantar la pluma permitiendo así la salida del vehículo. De lo contrario, permanece en el Estado ‘0010’ recolectando el boleto.

Estado ‘0011’ – ELEVA

En este estado, se activa el motor (MOT_2) y la señal (MX_2) para que la pluma suba, permitiendo la salida. Si el sensor (S_SUP2) detecta que la pluma ha sido elevada completamente, el sistema avanza al Estado ‘0100’ para esperar a que pase el vehículo. De lo contrario avanza al Estado ‘0111’ para verificar si hay un vehículo a la entrada.

Estado ‘0100’ – ESPERA

En este estado, la pluma está elevada, esperando a que el vehículo pase completamente. Si el sensor (S_S1) no detecta un vehículo a la salida, el sistema avanza al Estado ‘0110’ para revisar si un vehículo está saliendo. De lo contrario, permanece en el Estado ‘0100’.

Estado ‘0101’ – AUX1

Si el sensor (S_S2) no detecta un vehículo saliendo, avanza al Estado ‘0110’ para que posteriormente baje la pluma ya que el vehículo ha pasado completamente. De lo contrario, el sistema regresa al Estado ‘0100’ para esperar a que el vehículo pase completamente.

Estado ‘0110’ – AUX2

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados sea forzada a avanzar al Estado ‘1001’, haciendo una carga, para que se baje la pluma, ya que el vehículo ha pasado completamente.

Estado ‘0111’ – AUX3

En este estado continúa estando activa la señal del motor (MOT_2) y la señal (MX_2) para que la pluma suba. Cuando el sensor (S_S1) no detecta vehículo a la salida, el sistema avanza al Estado ‘1000’ para revisar si un vehículo está saliendo. De lo contrario, regresa al Estado ‘0011’ para seguir elevando la pluma.

Estado ‘1000’ – AUX4

En este estado continúa estando activa la señal del motor (MOT_2) y la señal (MX_2) para que la pluma suba. Si el sensor (S_S2) no detecta un vehículo saliendo, el sistema avanza al Estado ‘1001’ para bajar la pluma, debido a que el auto ha pasado antes de que la pluma se eleve completamente. De lo contrario, regresa al Estado ‘0011’ para seguir elevando la pluma.

Estado ‘1001’ – BAJA

En este estado se activa el motor (MOT_2) y la señal (nMX_2) para que la pluma baje. Si el sensor (S_INF2) detecta que la pluma no ha bajado en su totalidad, avanza al Estado ‘1010’ para revisar si hay un vehículo a la salida. De lo contrario, regresa al Estado ‘0000’ para verificar si algún vehículo requiere salir del estacionamiento.

Estado ‘1010’ – AUX5

En este estado continúa estando activa la señal del motor (MOT_2) y la señal (nMX_2) para que la pluma siga bajando. Si el sensor (S_S1) no detecta un vehículo a la salida, el sistema avanza al Estado ‘1011’ para revisar si hay un auto saliendo. De lo contrario, regresa al Estado ’0011 para subir la pluma y que ésta no dañe al vehículo.

Estado ‘1011’ – AUX6

En este estado continúa estando activa la señal del motor (MOT_2) y la señal (nMX_2) para que la pluma siga bajando. Si el sensor (S_S2) no detecta un vehículo saliendo, el sistema regresa al Estado ‘1001’ para seguir bajando la pluma. De lo contrario, avanza al Estado ‘1100’ para subir la pluma y que ésta no dañe al vehículo.

Estado ‘1100’ – AUX7

En este estado se activa la señal del motor (MOT_2) y la señal (MX_2) para que la pluma suba y no dañe al vehículo. En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados sea forzada avanzar al Estado ‘0011’, haciendo una carga, para que la pluma siga subiendo y no dañe al vehículo.

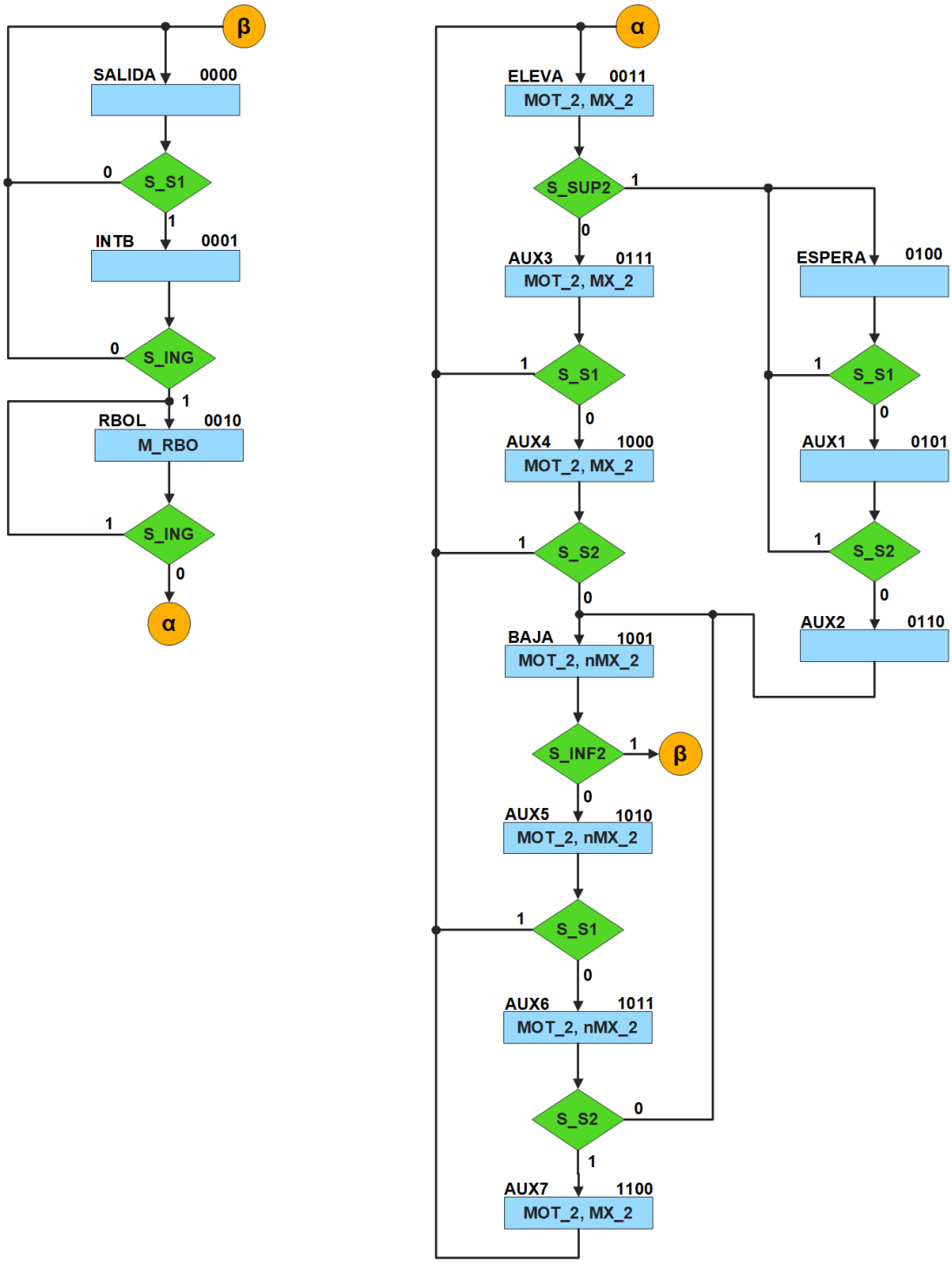


Figura P11.6 Carta ASM del sistema de salida del estacionamiento.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P11.3).

Tabla P11.3 Representación binaria de entradas para el sistema de salida del estacionamiento.

Entrada	Prueba
AUX	0 0 0
S_S1	0 0 1
S_ING	0 1 0
S_S2	0 1 1
S_SUP2	1 0 0
S_INF2	1 0 1

Se debe llenar la tabla P11.4 con base en la información de la carta ASM de la figura P11.6, usando el método de diseño con memoria y direccionamiento implícito.

A continuación, se describe como llenar los campos de la memoria para el Estado ‘0000’.

En el Estado ‘0000’ se selecciona la entrada S_S1, por lo tanto, se coloca en el campo de prueba su representación binaria, es decir, ‘001’. Si S_S1 es igual a cero, entonces el estado siguiente es el Estado ‘0000’, su representación binaria ‘0000’ es colocada en el campo de la liga, ya que se requiere hacer una carga. El campo VF es igual cero, ya que, para hacer una carga en el contador, el valor de la entrada y de VF deben ser iguales. En el Estado ‘0000’ no hay salidas activadas, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria, ver figura P11.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P11.4 Contenido de la memoria para el sistema de salida del estacionamiento.

Dirección de memoria				Contenido de memoria													
Estado presente				Prueba			VF	Liga				Salidas				Hex 1	Hex 2
QA	QB	QC	QD	I2	I1	I0		L3	L2	L1	L0	M_RBO	MOT_2	MX_2	nMX_2		
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	20	00
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	40	00
0	0	1	0	0	1	0	1	0	0	1	0	1	0	0	0	52	08
0	0	1	1	1	0	0	0	0	1	1	1	0	1	1	0	87	06
0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	0	34	00
0	1	0	1	0	1	1	1	0	1	0	0	0	0	0	0	74	00
0	1	1	0	0	0	0	1	1	0	0	1	0	0	0	0	19	00
0	1	1	1	0	0	1	1	0	0	1	1	0	1	1	0	33	06
1	0	0	0	0	1	1	1	0	0	1	1	0	1	1	0	73	06
1	0	0	1	1	0	1	1	0	0	0	0	0	1	0	1	B0	05
1	0	1	0	0	0	1	1	0	0	1	1	0	1	0	1	33	05
1	0	1	1	0	1	1	0	1	0	0	1	0	1	0	1	69	05
1	1	0	0	0	0	0	1	0	0	1	1	0	1	0	1	13	05

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de la biblioteca “PLD.H”.

El selector de entradas es un multiplexor de ocho líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P11.5). Se puede comprobar realizando el circuito de la figura P11.7.

Tabla P11.5 Tabla de funcionamiento del selector de entradas para el sistema de salida del estacionamiento.

Prueba			SI
I2	I1	I0	
0	0	0	AUX
0	0	1	S_S1
0	1	0	S_ING
0	1	1	S_S2
1	0	0	S_SUP2
1	0	1	S_INF2

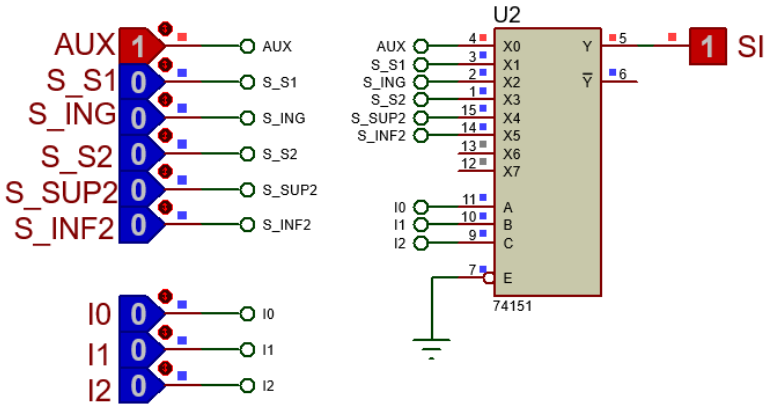


Figura P11.7 Multiplexor 74151 para el selector de entradas para el sistema de salida del estacionamiento.

La función booleana del selector de entradas queda:

```
SI = AUX&!I2&!I1&!I0| S_S1&!I2&!I1&I0 | S_ING&!I2&I1&!I0 | S_S2&!I2&I1&I0
     | S_SUP2&I2&!I1&!I0 | S_INF2&I2&!I1&I0;
```

Para obtener el valor de la lógica, se debe hacer la operación XOR entre el selector de entradas y el valor de VF (ver figura P11.5).

Por lo tanto, la función booleana de la lógica queda:

```
SL= VF ^ (AUX&!I2&!I1&!I0| S_S1&!I2&!I1&I0 | S_ING&!I2&I1&!I0 |
    S_S2&!I2&I1&I0 | S_SUP2&I2&!I1&!I0 | S_INF2&I2&!I1&I0);
```

Se utiliza un contador con carga paralela que indica el estado siguiente. El contador con carga paralela será implementado por el PIC16F1939. Si el valor a la salida de la lógica es igual a '1', el contador carga la información binaria de la liga a la memoria. Si el valor de la lógica es igual a '0', se cuenta al siguiente estado binario. A continuación, se obtienen las expresiones lógicas para un contador de cuatro bits por el método de variable suscrita (ver tabla P11.6).

Tabla P11.6 Mapa de Karnaugh general de transición de estados para un contador de cuatro bits.

A B \ C D	0 0	0 1	1 1	1 0
0 0	0000	0001	0011	0010
	0001	0010	0100	0011
0 1	0100	0101	0111	0110
	0101	0110	1000	0111
1 1	1100	1101	1111	1110
	1101	1110	0000	1111
1 0	1000	1001	1011	1010
	1001	1010	1100	1011

Para el bit A:

A B \ C D	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	1	0	1
10	1	1	1	1

```
FFA = A&!C | A&!B | A&!D |
      !A&B&C&D;
```

Para el bit B:

A B \ C D	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	1	1	0	1
10	0	0	1	0

```
FFB = B&!C | B&!D | !B&C&D;
```

Para el bit C:

A B \ C D	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

```
FFC = !C&D | C&!D = C^D;
```

Para el bit D:

A B \ C D	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

```
FFD = !D;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P11.8, figura P11.9). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939.

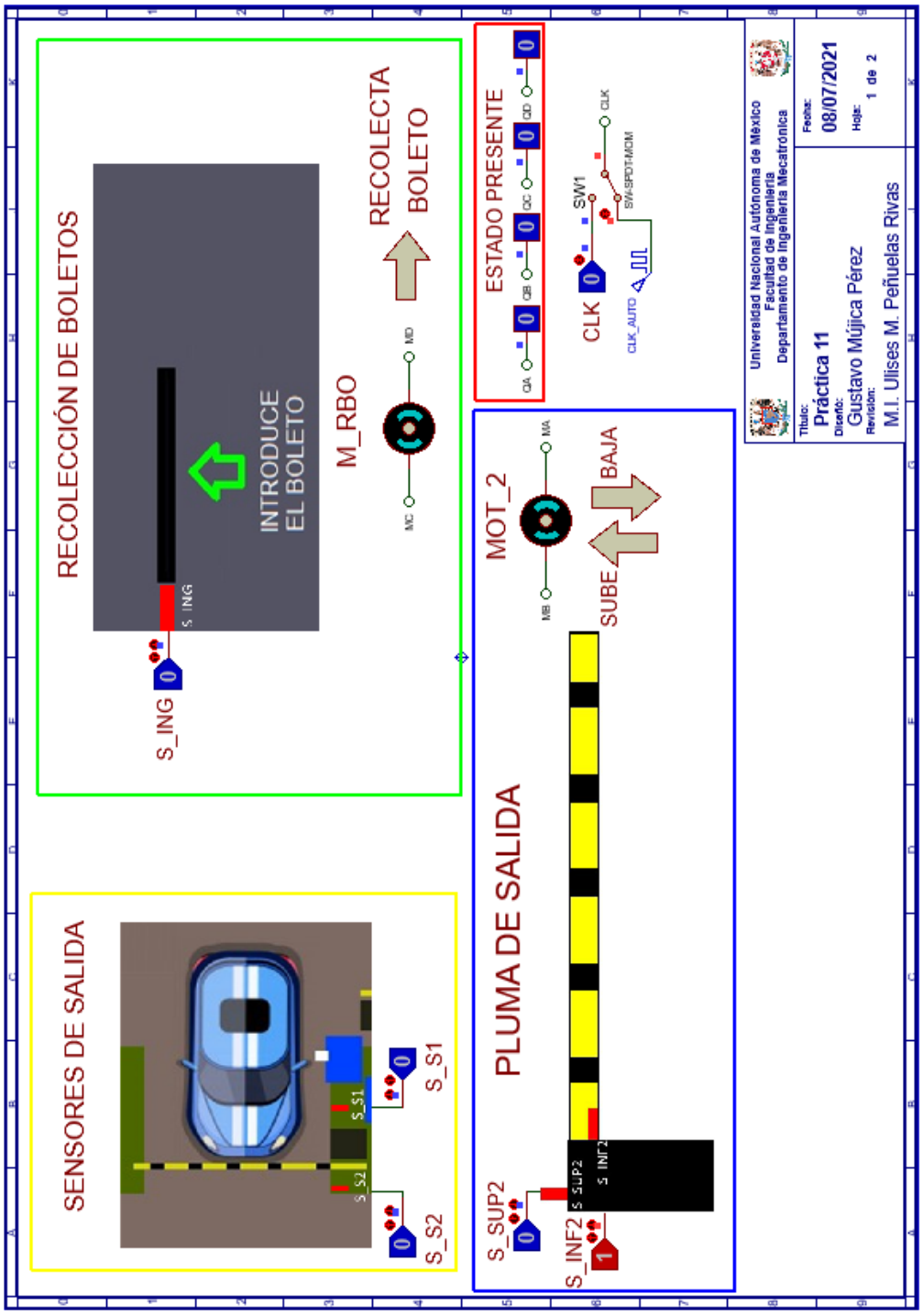


Figura P11.8 Interfaz hombre-máquina para el controlador de la Práctica 11 hoja 1/2.

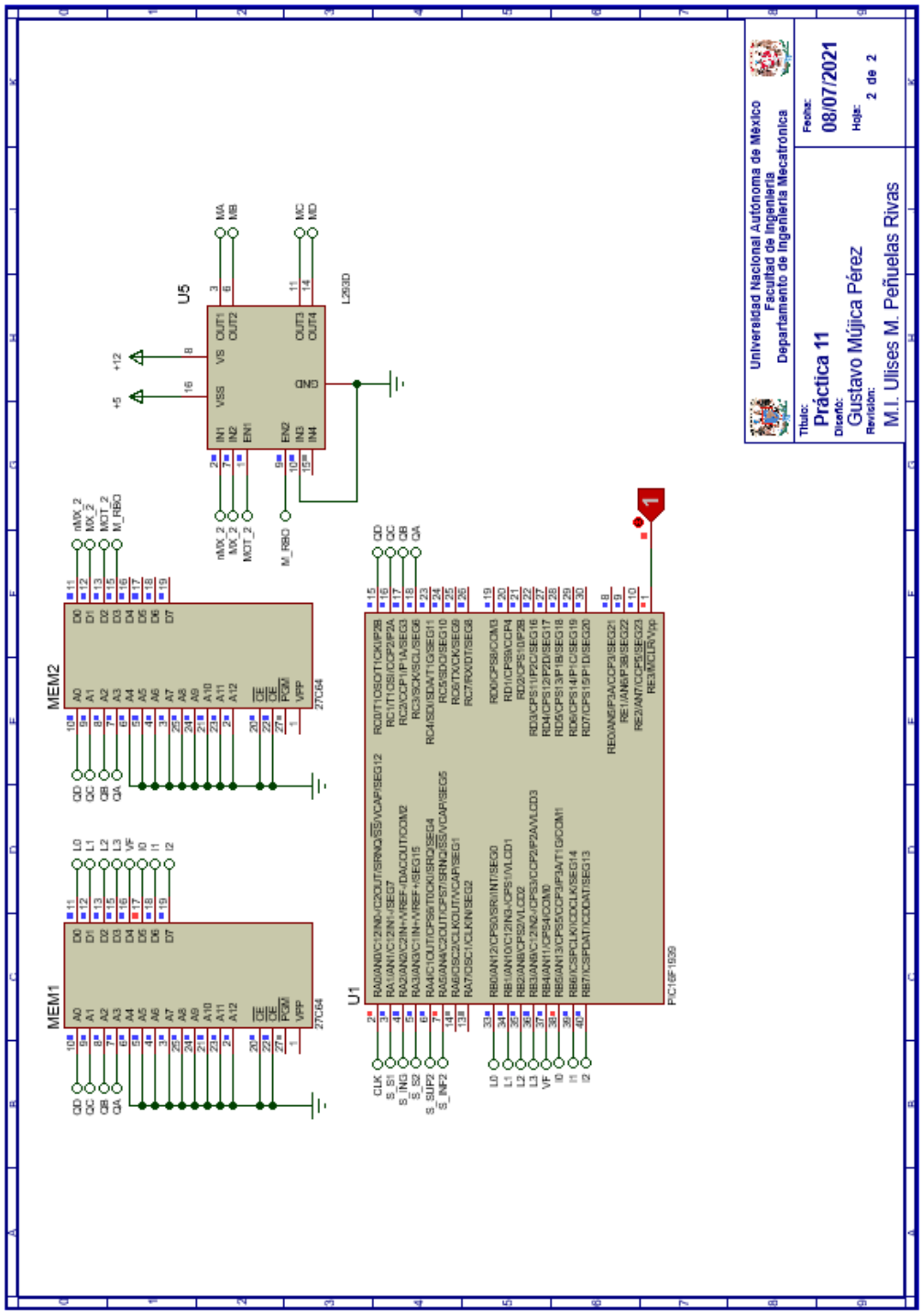


Figura P11.9 Esquema electrónico para el controlador de la Práctica 11 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P11.10,figura P11.11) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     // Carga biblioteca PLD.h

//***CONTADOR***

#define CLK A0 //RELOJ
//ENTRADAS

//LIGA
#define L0 B0
#define L1 B1
#define L2 B2
#define L3 B3

//SALIDAS
#define A C3 //FFA
#define B C2 //FFB
#define C C1 //FFC
#define D C0 //FFD

//***LOGICA***

//ENTRADAS
#define S_S1   A1
#define S_ING  A2
#define S_S2   A3
#define S_SUP2 A4
#define S_INF2 A5

//VF
#define VF B4

//PRUEBAS
#define I0 B5
#define I1 B6
#define I2 B7

//VARIABLES INTERMEDIAS
short SL=0;  //Salida lógica
short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
short At=0, Bt=0, Ct=0, Dt=0; //Variables inntetrnas de los FF
short LD3,LD2,LD1,LD0; //Variables intermedias de la liga

void main ()
{
   pld_ini();  // INICIALIZA AL PIC COMO PLD
  // pld_555(10);   // Genera señal cuadrada en Hz,
                    //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
    while(1) //LOOP INFINITO
    {
```

Figura P11.10 Código de la Práctica 11 parte 1.

```
50:     {
51:     //CIRCUITO COMBINACIONAL
52:     LD3=L3; LD2=L2; LD1=L1; LD0=L0;      /*ALMACENA DATOS HASTA EL CAMBIO DEL RELOJ,
53:                                           SIMULANDO UN REGISTRO, EVITANDO ASÍ
54:                                           QUE SE MODIFIQUEN LOS VALORES DE LA MEMORIA*/
55:
56:     //SALIDA LÓGICA
57:     SL= VF ^ (AUX&!I2&!I1&!I0| S_S1&!I2&!I1&I0 | S_ING&!I2&I1&!I0 |
58:         S_S2&!I2&I1&I0 | S_SUP2&I2&!I1&!I0 | S_INF2&I2&!I1&I0);
59:
60:     //CIRCUITO SECUENCIAL (CONTADOR DE CARGA PARALELA)
61:
62:       if (!CLK) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO
63:    // if (!out_555) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO,
64:                     //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
65:     {//SECCIÓN DE OPERACIONES DEL CONTADOR
66:     At= A&!C | A&!B | A&!D | !A&B&C&D;
67:     Bt= B&!C | B&!D | !B&C&D;
68:     Ct= C^D;
69:     Dt= !D;
70:
71:     }
72:
73:     else
74:     {//SECCIÓN DE MEMORIZACIÓN
75:     //CUENTA CON SL=1 Y CARGA CON SL=0
76:     A= SL&At | !SL&LD3;
77:     B= SL&Bt | !SL&LD2;
78:     C= SL&Ct | !SL&LD1;
79:     D= SL&Dt | !SL&LD0;
80:
81:     while(clk){}/*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
82:                  POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
83:                  DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
84:                  LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ*/
85:
86:    /* while(out_555){} /*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
87:                       POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
88:                       DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
89:                       LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ,
90:                       EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO*/
91:
92:     }
93:
94:     }
95:
96: }
```

Figura P11.11 Código de la Práctica 11 parte 2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 12 Sistema de posición inicial utilizando diseño con memoria y direccionamiento por trayectoria

Introducción

A través de una maqueta, se simuló el funcionamiento de un sistema robotizado que clasifica paquetes o cajas por colores y debe ser capaz de reconocer tres tipos de paquetes por su color colocados en una banda transportadora.

Los elementos principales del sistema son: un robot, un sensor de color y una banda transportadora. Inicialmente el sistema está en reposo, para activar el sistema se debe presionar un botón, al presionarlo la banda transportadora empezará a desplazar un paquete, un sensor detectará cuando el paquete esté en la posición adecuada para evaluar su color, por lo que la banda se detendrá.

El robot sujeta los paquetes y los deposita en el contenedor correspondiente. Los colores serán rojo, azul y verde. Los paquetes que no sean de estos colores, los dejará pasar (ver figura P12.1).

El robot se mueve a lo largo del eje horizontal por medio de un motor DC, para desplazarse en el eje vertical lo hace por medio de un motor paso a paso. El robot cuenta con un gripper, el cual es el encargado de sujetar los paquetes. El gripper está conformado por un motor DC.

El sistema tiene un botón de emergencia, cuando éste sea oprimido el sistema quedará inmóvil, hasta que se oprima otro botón para dirigirse a su posición inicial. Se consideró la posición inicial del sistema cuando la articulación horizontal está en su extremo izquierdo, la articulación vertical en su posición superior y con el gripper abierto.

El sistema se dividió en 2 partes para facilitar su comprensión. El sistema de posición inicial controla cuando el gripper debe moverse en su posición inicial y el sistema clasificador, controla la banda y la clasificación de los paquetes por colores. La unión de los dos sistemas mencionados anteriormente forma el sistema robotizado clasificador de paquetes.

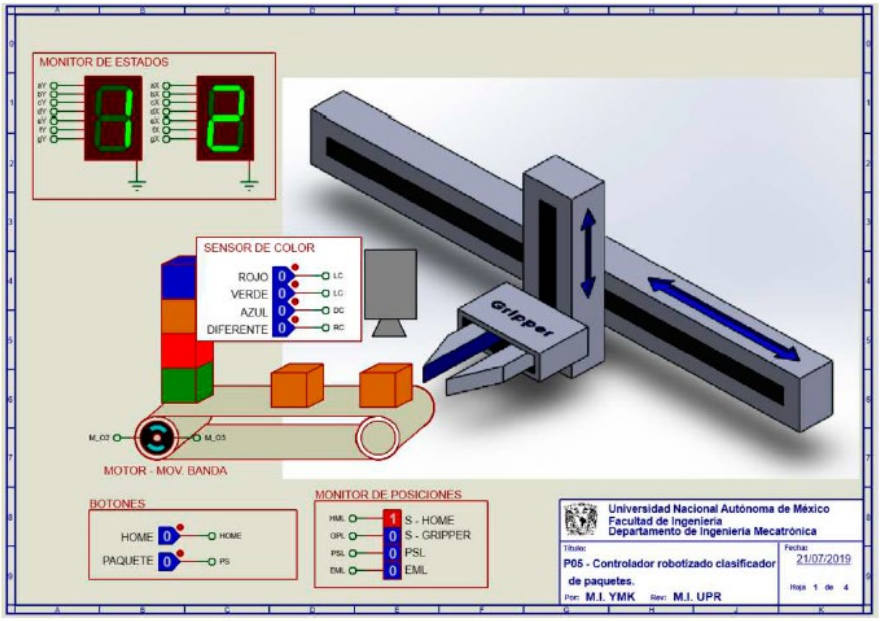


Figura P12.1 Sistema robotizado para clasificar paquetes por colores.

El diseño con memoria y direccionamiento por trayectoria guarda el estado siguiente según la salida de cada estado de la carta ASM en una localidad de memoria. La porción de la memoria que indica el estado siguiente es llamada “liga”, mientras que la porción que indica las salidas se llama “la parte de las salidas” [1].

La arquitectura de un diseño con memorias y direccionamiento por trayectoria se muestra en la figura P12.2, donde:

A = estado siguiente

B = entradas del estado siguiente

C = entradas del estado presente

D = estado presente

E = dirección del estado siguiente

F = salidas del estado presente.

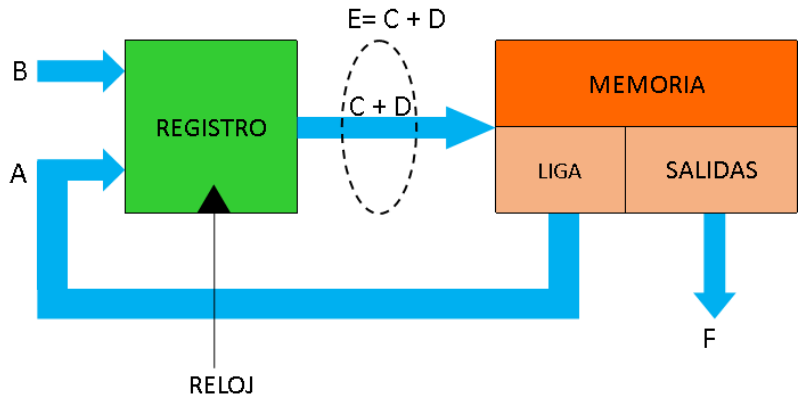


Figura P12.2 Arquitectura de un controlador con memoria y direccionamiento por trayectoria.

En este tipo de diseño todas las salidas de la carta ASM deberán depender del estado presente y de los valores de entrada.

Objetivo

Diseñar un controlador para el sistema de posición inicial por medio del método de diseño con memoria y direccionamiento por trayectoria.

Descripción

Primero se diseña una carta ASM para el sistema de posición inicial por medio del método de diseño con memoria y direccionamiento por trayectoria. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P12.1 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema de posición inicial se necesitan señales de entrada:

- un botón detecta la solicitud para que el sistema comience a funcionar o para regresar a la posición inicial de éste

- se necesitan sensores para detectar la posición inferior y superior del sistema
- se necesita un sensor de final de carrera en el extremo izquierdo del sistema
- un sensor verifica si el gripper está sujetando una caja.

Como salida se requiere de las siguientes señales:

- activación del motor para el movimiento vertical
- una señal se activa para mover el gripper a la posición superior del sistema
- activación del motor para el movimiento horizontal
- una señal se activa para mover el gripper hacia el extremo izquierdo del sistema
- activación del motor del gripper
- una señal se activa para que el gripper se abra y otra para que se cierre.

Tabla P12.1 Entradas y Salidas para el sistema de posición inicial.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
BH	D4 del registro	I	Botón de Home para ir a la posición inicial del sistema
SVS	D3 del registro	I	Sensor de posición superior
SVI	D2 del registro	I	Sensor de posición inferior
SHI	D1 del registro	I	Sensor de final de carrera izquierda
SG	D0 del registro	I	Sensor de gripper que indica cuando el gripper está sujetando una caja
MV	D4 de la memoria 1	O	Motor vertical paso a paso para el movimiento vertical
M_S	D3 de la memoria 1	O	Señal para hacer que el gripper suba
MH	D2 de la memoria 1	O	Motor horizontal de CD para el movimiento horizontal
M_I	D1 de la memoria 1	O	Señal para hacer que el gripper se mueva a la izquierda
MG	D0 de la memoria 1	O	Motorreductor del gripper
M_A	D0 de la memoria 2	O	Señal para abrir el gripper.

Notas de diseño

a) Para iniciar a mover la banda si se acaba de encender el sistema, se debe oprimir el botón de home
b) Para que funcione de nuevo el sistema si se oprimió el botón de paro, se debe oprimir el botón de home
c) Se considera la posición inicial del sistema cuando el gripper está abierto, en la posición superior y en el extremo izquierdo del sistema.

Reglas de funcionamiento

- BH: botón de home
  1 = se oprimió el botón de home
  0 = no se oprimió el botón de home
- SVS: sensor de posición superior
  1 = el gripper está en la posición superior
  0 = no está el gripper en la posición superior
- SVI: sensor de posición inferior
  1 = el gripper está en la posición inferior del sistema
  0 = no está el gripper en la posición inferior del sistema
- SHI: sensor de final de carrera izquierda
  1 = el gripper está en el extremo izquierdo del sistema
  0 = no está el gripper en el extremo izquierdo del sistema
- SG: sensor de gripper
  1 = el gripper está sujetando una caja o cerrado
  0 = no está el gripper sujetando una caja o abierto.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo.

Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado. El algoritmo de la máquina de estados se puede ver en la figura P12.3.

Estado '000' – INICIO

El sistema se dirige a estado, después de que se deja de presionar el botón de paro o cuando se acaba de encender. Cuando el botón (BH) es oprimido, el sistema avanza al Estado '001' para verificar la posición vertical del gripper. De lo contrario, permanece en el Estado '000', en espera de que sea oprimido.

Estado '001' – DETPV

En estado se verifica la posición vertical del sistema. Si el sensor (SVS), detecta al gripper en la posición vertical superior, el sistema avanza al Estado '011' para detectar si el gripper está en alguno de los extremos horizontales del sistema. De lo contrario, avanza al Estado '010' para subir el gripper.

Estado '010' – SUBIRG

En este estado se activa el motor paso a paso (MV) y la señal (M_S) para que el motor paso a paso suba el gripper a la posición superior del sistema. Cuando el sensor (SVS), detecte que el gripper ha llegado a la posición superior, el sistema avanza al Estado '011' para detectar si el gripper está en alguno de los extremos horizontales del sistema. De lo contrario, permanecerá en el Estado '010' subiendo el gripper.

Estado '011' – DETPH

En estado se verifica si el gripper está en alguno de los extremos horizontales del sistema. Si el sensor (SHI), detecta que el gripper se encuentra en el final de carrera izquierda, el sistema avanza al Estado '101' para detectar si el gripper está abierto o cerrado. De lo contrario, avanza al Estado '100', para mover el gripper al extremo izquierdo del sistema.

Estado '100' – MOVGI

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_I) para mover el gripper al extremo izquierdo del sistema.

Cuando el sensor (SHI), detecta que el gripper ha llegado al límite izquierdo del sistema, el sistema avanza al Estado ‘101’ para detectar si el gripper está abierto o cerrado. De lo contrario, permanece en el Estado ‘100’ desplazando el gripper hacia la izquierda.

Estado ‘101’ – DETG

En estado se verifica si el gripper está abierto o cerrado. Si el sensor (SG) detecta que el gripper está abierto, regresa al Estado ‘000’ para iniciar nuevamente el proceso. De lo contrario, el sistema avanza al Estado ‘110’ para bajar el gripper a la posición inferior y soltar la caja que está sujetando.

Estado ‘110’ – BAJARG

En este estado se activa el motor paso a paso (MV) y debido a que no se activa la señal (M_S) para subir el gripper, el gripper desciende a la posición inferior del sistema. Cuando el sensor (SVI), detecta que el gripper ha llegado a la posición inferior, el sistema avanza al Estado ‘111’ para abrir el gripper y por lo tanto soltar la caja. De lo contrario, permanece en el Estado ‘110’ bajando el gripper.

Estado ‘111’ – ABRIRG

En este estado se activa el motor del gripper (MG) y la señal (M_A) para que el gripper se abra y suelte la caja. Cuando el sensor del gripper (SG), detecte que se ha abierto el gripper completamente, el sistema regresa al Estado ‘010’ para desplazar el gripper a su posición inicial. De lo contrario, permanece en el Estado ‘111’ abriendo el gripper.

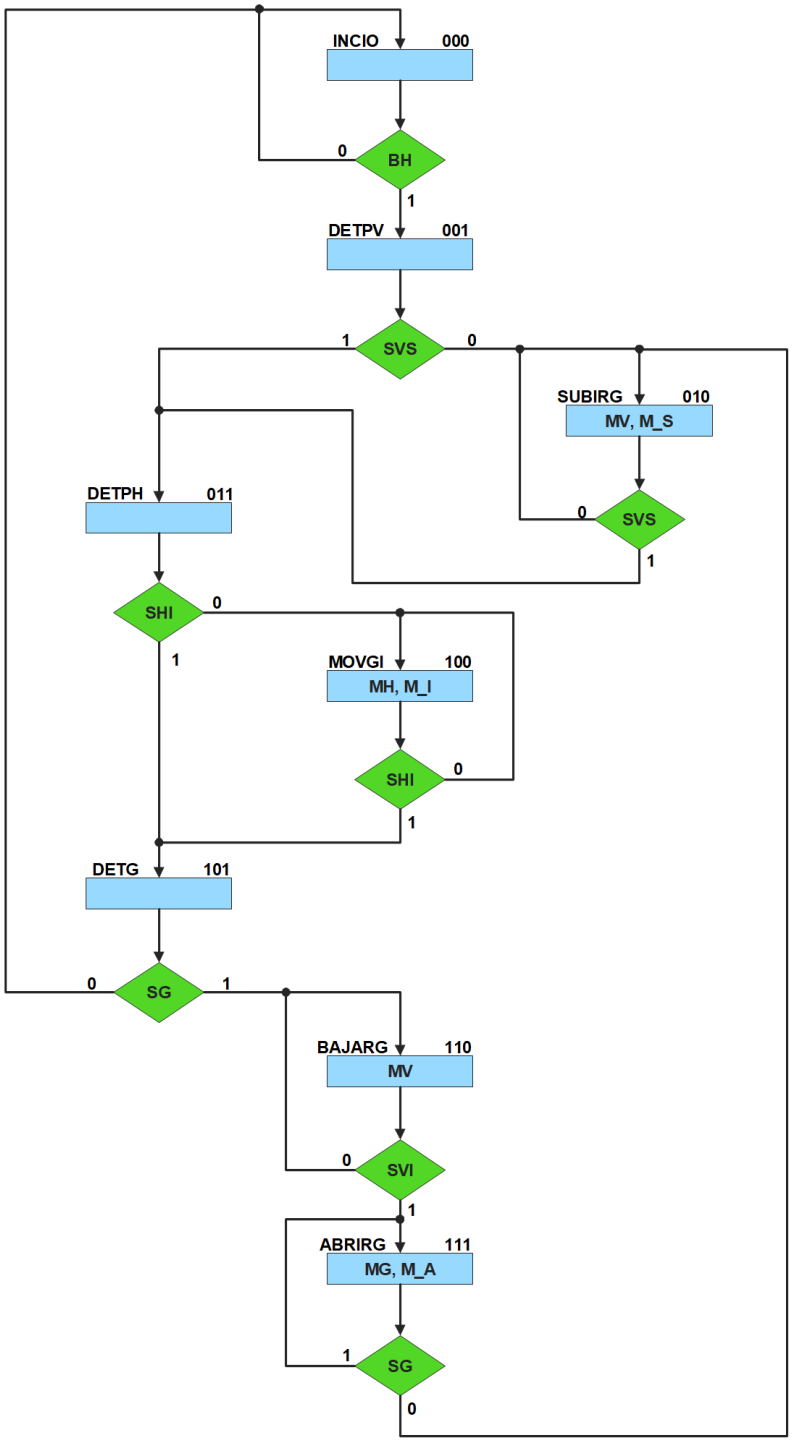


Figura P12.3 Carta ASM de sistema de posición inicial.

Solución

Se debe llenar de la tabla P12.2 a la tabla P12.9, usando el método de diseño con memoria y direccionamiento por trayectoria. Para cada estado es necesario considerar todas las posibles combinaciones de las variables de entrada aun cuando algunas de ellas no se utilicen [1].

A continuación, se describe como llenar los campos de la memoria para el Estado ‘000’.

Debido a que hay cinco variables de entrada se deben considerar 32 posibles combinaciones de éstas para cada estado.

Si en el Estado ‘000’ B_H es igual a ‘0’, el estado siguiente será el Estado ‘000’ independientemente de los valores de las otras variables. En el Estado ‘000’ no hay señales de salida activadas, por lo que se coloca un ‘0’ en la parte de salidas de todo el estado.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P12.2). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P12.2 Contenido de la memoria para el sistema de posición inicial parte 1.

	Dirección de memoria								Contenido de memoria										
	Estado presente			Entradas					Liga			Salidas					Salidas	HEX 1	HEX 2
	QA	QB	QC	BH	SVS	SVI	SHI	SG	QA	QB	QC	MV	M_S	MH	M_I	MG	M_A		
Estado 000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	00	00
	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	0	1	0	1	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	0	1	1	1	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	1	1	0	1	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	20	00
	0	0	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0	20	00

Tabla P12.3 Contenido de la memoria para el sistema de posición inicial parte 2.

	Dirección de memoria								Contenido de memoria										
	Estado presente			Entradas					Liga			Salidas					Salidas	HEX 1	HEX 2
	QA	QB	QC	BH	SVS	SVI	SHI	SG	QA	QB	QC	MV	M_S	MH	M_I	MG	M_A		
Estado 001	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	0	1	0	0	1	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	0	1	0	1	0	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	0	1	0	1	1	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	0	1	1	0	0	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	0	1	1	1	0	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	0	1	1	1	1	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	1	0	1	0	1	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	1	0	1	1	1	0	1	0	0	0	0	0	0	0	40	00
	0	0	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	1	1	0	0	1	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	1	1	0	1	0	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	1	1	0	1	1	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	1	1	1	0	1	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	60	00
	0	0	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	60	00

Tabla P12.4 Contenido de la memoria para el sistema de posición inicial parte 3.

	Dirección de memoria								Contenido de memoria										
	Estado presente			Entradas					Liga			Salidas					Salidas	HEX 1	HEX 2
	QA	QB	QC	BH	SVS	SVI	SHI	SG	QA	QB	QC	MV	M_S	MH	M_I	MG	M_A		
Estado 010	0	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	0	0	0	0	1	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	0	0	0	1	0	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	0	0	0	1	1	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	0	0	1	0	0	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	0	0	1	0	1	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	0	0	1	1	0	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	0	0	1	1	1	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	0	1	0	0	0	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	0	1	0	0	1	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	0	1	0	1	0	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	0	1	0	1	1	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	0	1	1	0	0	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	0	1	1	0	1	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	0	1	1	1	0	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	0	1	1	1	1	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	1	0	0	0	0	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	1	0	0	0	1	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	1	0	0	1	0	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	1	0	0	1	1	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	1	0	1	0	0	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	1	0	1	0	1	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	1	0	1	1	0	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	1	0	1	1	1	0	1	0	1	1	0	0	0	0	58	00
	0	1	0	1	1	0	0	0	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	1	1	0	0	1	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	1	1	0	1	0	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	1	1	0	1	1	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	1	1	1	0	0	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	1	1	1	0	1	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	1	1	1	1	0	0	1	1	1	1	0	0	0	0	78	00
	0	1	0	1	1	1	1	1	0	1	1	1	1	0	0	0	0	78	00

Tabla P12.5 Contenido de la memoria para el sistema de posición inicial parte 4.

	Dirección de memoria								Contenido de memoria										
	Estado presente			Entradas					Liga			Salidas					Salidas	HEX 1	HEX 2
	QA	QB	QC	BH	SVS	SVI	SHI	SG	QA	QB	QC	MV	M_S	MH	M_I	MG	M_A		
Estado 011	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	0	0	1	1	0	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	0	0	1	1	1	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	0	1	0	1	1	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	0	1	1	1	1	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	1	0	0	1	0	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	1	0	0	1	1	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	1	0	1	1	0	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	1	0	1	1	1	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	1	1	0	1	0	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	1	1	0	1	1	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	80	00
	0	1	1	1	1	1	1	0	1	0	1	0	0	0	0	0	0	A0	00
	0	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	A0	00

Tabla P12.6 Contenido de la memoria para el sistema de posición inicial parte 5.

	Dirección de memoria								Contenido de memoria										
	Estado presente			Entradas					Liga			Salidas					Salidas	HEX 1	HEX 2
	QA	QB	QC	BH	SVS	SVI	SHI	SG	QA	QB	QC	MV	M_S	MH	M_I	MG	M_A		
Estado 100	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	0	0	0	0	1	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	0	0	0	1	0	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	0	0	0	1	1	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	0	0	1	0	1	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	0	0	1	1	0	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	0	0	1	1	1	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	0	1	0	0	1	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	0	1	0	1	0	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	0	1	0	1	1	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	0	1	1	0	0	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	0	1	1	0	1	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	0	1	1	1	0	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	0	1	1	1	1	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	1	0	0	0	1	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	1	0	0	1	0	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	1	0	0	1	1	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	1	0	1	0	0	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	1	0	1	0	1	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	1	0	1	1	0	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	1	0	1	1	1	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	1	1	0	0	0	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	1	1	0	0	1	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	1	1	0	1	0	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	1	1	0	1	1	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	1	1	1	0	0	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	1	1	1	0	1	1	0	0	0	0	1	1	0	0	86	00
	1	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	0	A6	00
	1	0	0	1	1	1	1	1	1	0	1	0	0	1	1	0	0	A6	00

Tabla P12.7 Contenido de la memoria para el sistema de posición inicial parte 6.

	Dirección de memoria								Contenido de memoria										
	Estado presente			Entradas					Liga			Salidas					Salidas	HEX 1	HEX 2
	QA	QB	QC	BH	SVS	SVI	SHI	SG	QA	QB	QC	MV	M_S	MH	M_I	MG	M_A		
Estado 101	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	0	0	1	0	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	1	0	0	1	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	1	0	1	0	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	C0	00
	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	00	00
	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	C0	00

Tabla P12.8 Contenido de la memoria para el sistema de posición inicial parte 7.

	Dirección de memoria								Contenido de memoria										
	Estado presente			Entradas					Liga			Salidas					Salidas	HEX 1	HEX 2
	QA	QB	QC	BH	SVS	SVI	SHI	SG	QA	QB	QC	MV	M_S	MH	M_I	MG	M_A		
Estado 110	1	1	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	0	0	0	0	1	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	0	0	0	1	0	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	0	0	0	1	1	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	0	0	1	0	1	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	0	0	1	1	0	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	0	1	0	0	1	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	0	1	0	1	0	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	0	1	0	1	1	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	0	1	1	0	0	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	0	1	1	0	1	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	0	1	1	1	0	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	1	0	0	0	0	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	1	0	0	0	1	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	1	0	0	1	0	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	1	0	0	1	1	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	1	0	1	0	0	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	1	0	1	1	0	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	1	0	1	1	1	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	1	1	0	0	0	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	1	1	0	0	1	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	1	1	0	1	0	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	1	1	0	1	1	1	1	0	1	0	0	0	0	0	D0	00
	1	1	0	1	1	1	0	0	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	1	1	1	0	1	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	1	1	1	1	0	1	1	1	1	0	0	0	0	0	F0	00
	1	1	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	F0	00

Tabla P12.9 Contenido de la memoria para el sistema de posición inicial parte 8.

	Dirección de memoria								Contenido de memoria										
	Estado presente			Entradas					Liga			Salidas					Salidas	HEX 1	HEX 2
	QA	QB	QC	BH	SVS	SVI	SHI	SG	QA	QB	QC	MV	M_S	MH	M_I	MG	M_A		
Estado 111	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	0	0	0	1	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	0	0	0	1	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	0	0	1	0	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	0	0	1	1	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	0	0	1	1	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	0	1	0	0	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	0	1	0	0	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	0	1	0	1	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	0	1	0	1	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	0	1	1	0	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	0	1	1	0	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	0	1	1	1	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	1	0	0	1	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	1	0	0	1	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	1	0	1	0	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	1	0	1	1	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	1	1	0	0	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	1	1	0	1	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	1	1	0	1	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	1	1	1	0	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	1	1	1	0	1	1	1	1	0	0	0	0	1	1	E1	01
	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	1	1	41	01
	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	E1	01

Para controlar el motor paso a paso se utilizarán Flip-Flops tipo D, de manera que sea posible controlar su sentido de giro con la señal de salida M_S de la memoria. Si M_S es igual a '1', el motor girará hacia la derecha para subir el gripper a la posición superior del sistema. Cuando M_S sea igual a'0'el motor girará hacia la izquierda para bajar el gripper a la posición inferior del sistema. A continuación, se muestra la carta ASM para el motor paso a paso (ver figura P12.4).

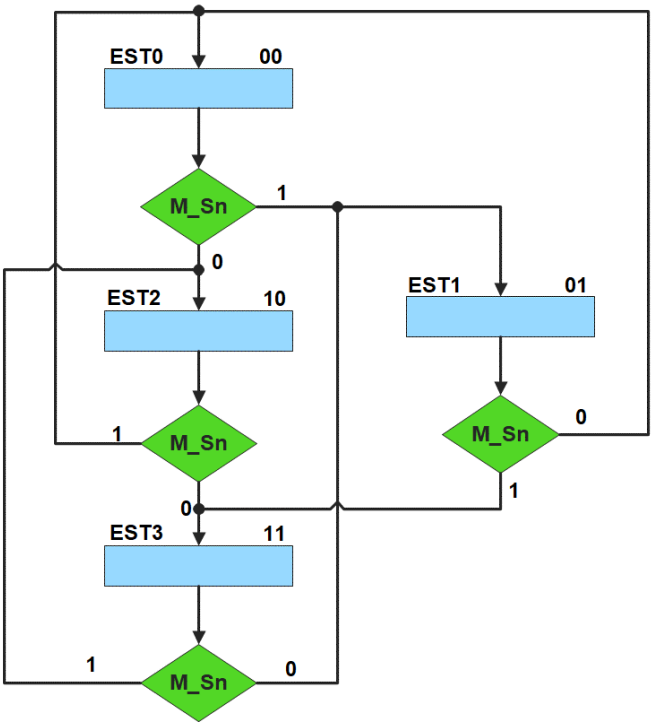


Figura P12.4 Carta ASM del motor paso a paso.

Los bits de los estados de la Carta ASM del motor paso a paso serán el patrón de pulsos de entrada a éste para hacerlo girar, dependiendo de la señal M_Sn. La señal de salida M_S de la máquina de estados del sistema de posición inicial, será una entrada (M_Sn) a la máquina de estados del motor pasos, de esta manera se relacionan las máquinas de estados.

La máquina de estados para controlar las entradas al motor paso a paso se resolverá por el método de variable suscrita. El mapa de Karnaugh general de transición de estados para el motor paso a paso se muestra en la tabla P12.10.

Tabla P12.10 Mapa de Karnaugh general de transición de estados para el motor paso a paso.

		T = 0	T = 1
S	0	EST0 - 00	EST1 - 01
		M_Sn - 01 !M_Sn - 10	M_Sn - 11 !M_Sn - 00
	1	EST2 - 10	EST3 - 11
		M_Sn - 00 !M_Sn - 11	M_Sn - 10 !M_Sn - 01

Mapas de Karnaugh particulares para los Flip-Flops tipo D de los bits S y T son:

S \ T	0	1
0	!M_Sn	M_Sn
1	!M_Sn	M_Sn

```
DS = !T&!M_Sn | T&M_S n;
```

S \ T	0	1
0	M_Sn	M_Sn
1	!M_Sn	!M_Sn

```
DT = !S&M_Sn | S&!M_Sn = S^M_Sn;
```

Las expresiones de los Flip-Flops harán la operación AND con la señal MV para que sólo cambien de estado cuando este activado el motor paso a paso.

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P12.5, figura P12.6). Se carga en el controlador el archivo con extensión “HEX” de la memoria. Se debe seguir la descripción de la carta ASM para probar la implementación de esta práctica. Para poder visualizar de manera más rápida el movimiento del motor paso a paso, se debe utilizar una frecuencia de 30 Hz y el ángulo en el que gira el motor paso a paso es de 3.6°.

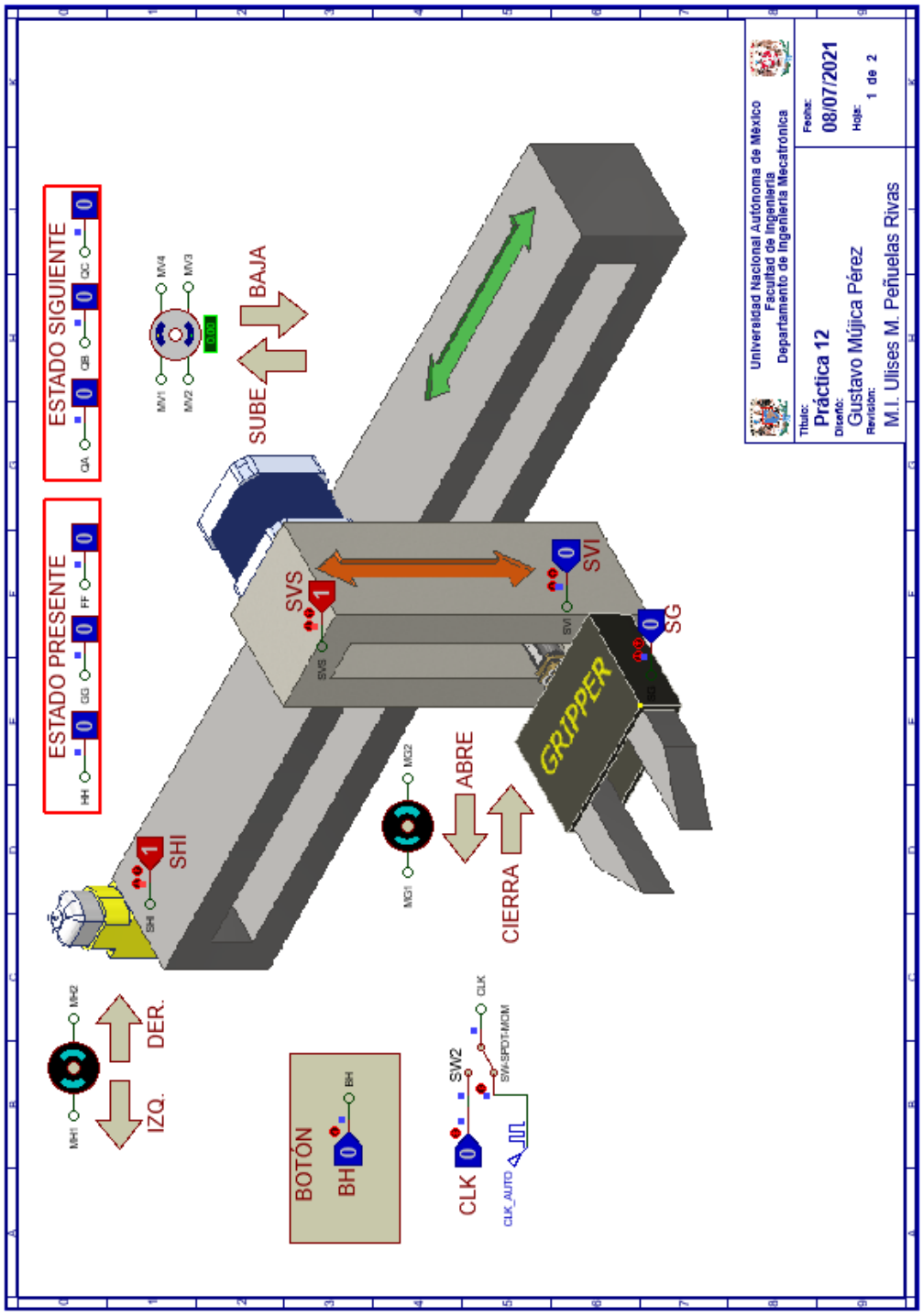


Figura P12.5 Interfaz hombre-máquina para el controlador de la Práctica 12 hoja 1/2.

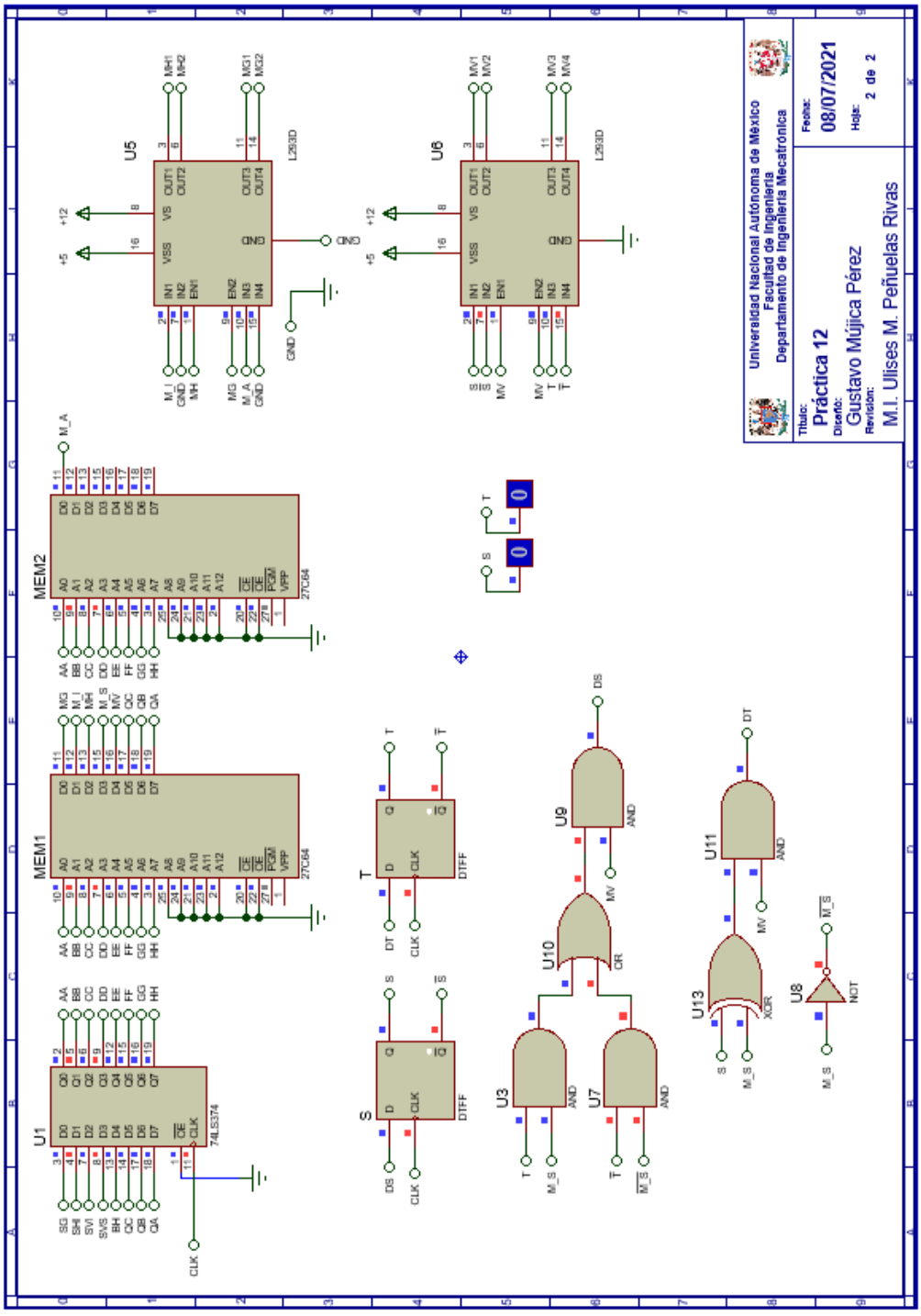


Figura P12.6 Esquema electrónico para el controlador de la Práctica 12 hoja 2/2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 13 Sistema de posición inicial; diseño con memoria y direccionamiento entrada-estado

Introducción

A través de una maqueta, se simuló el funcionamiento de un sistema robotizado que clasifica paquetes o cajas por colores y debe ser capaz de reconocer tres tipos de paquetes por su color colocados en una banda transportadora.

Los elementos principales del sistema son: un robot, un sensor de color y una banda transportadora. Inicialmente el sistema está en reposo, para activar el sistema se debe presionar un botón, al presionarlo la banda transportadora empezará a desplazar un paquete, un sensor detectará cuando el paquete esté en la posición adecuada para evaluar su color, por lo que la banda se detendrá.

El robot sujeta los paquetes y los deposita en el contenedor correspondiente. Los colores serán rojo, azul y verde. Los paquetes que no sean de estos colores, los dejará pasar (ver figura P13.1).

El robot se mueve a lo largo del eje horizontal por medio de un motor DC, para desplazarse en el eje vertical lo hace por medio de un motor paso a paso. El robot cuenta con un gripper, el cual es el encargado de sujetar los paquetes. El gripper está conformado por un motor DC.

El sistema tiene un botón de emergencia, cuando éste sea oprimido el sistema quedará inmóvil, hasta que se oprima otro botón para dirigirse a su posición inicial. Se consideró la posición inicial del sistema cuando la articulación horizontal está en su extremo izquierdo, la articulación vertical en su posición superior y con el gripper abierto.

El sistema se dividió en 2 partes para facilitar su comprensión. El sistema de posición inicial controla cuando el gripper debe moverse en su posición inicial y el sistema clasificador, controla la banda y la clasificación de los paquetes por colores. La unión de los dos sistemas mencionados anteriormente forma el sistema robotizado clasificador de paquetes.

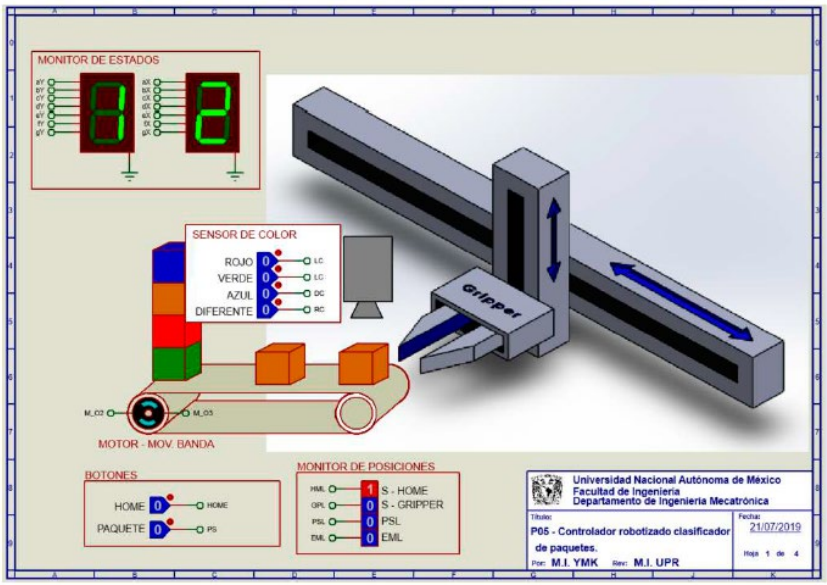


Figura P13.1 Sistema robotizado para clasificar paquetes por colores.

El diseño con memoria y direccionamiento entrada-estado restringe las cartas ASM a una sola entrada por estado. Una nueva porción de la palabra de memoria contiene una representación binaria de la entrada a probar en cada estado, esta parte es llamada “la parte de prueba”. Con esta representación binaria un selector de entrada elige una de las variables de entrada (ver figura P13.2) [1].

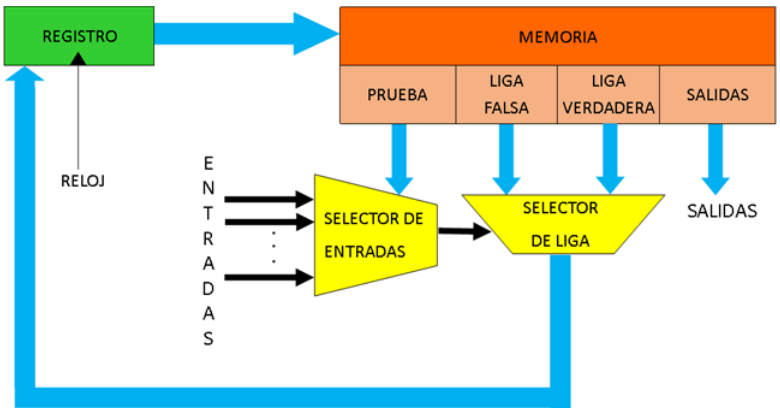


Figura P13.2 Arquitectura de un controlador con memoria y direccionamiento entrada-estado.

La parte de liga tiene dos estados siguientes, escogiéndose uno por el selector de liga, con base en la entrada seleccionada por la parte de prueba. Si el valor de la entrada seleccionada por el selector de entradas es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera (ver figura P13.3) [1].

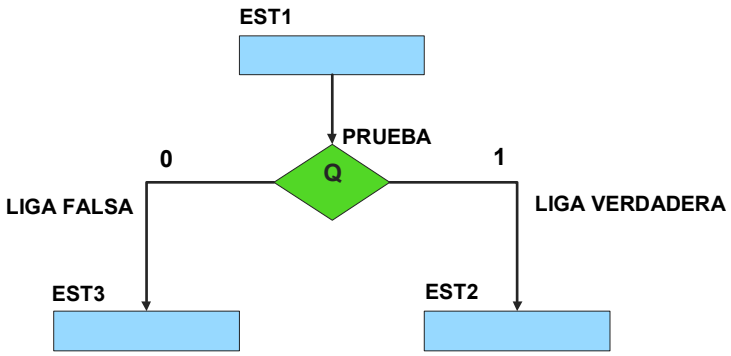


Figura P13.3 La liga falsa es el estado EST3, mientras que la liga verdadera es el estado EST2.

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará también una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista, se probará la variable auxiliar, la cual tiene un valor preestablecido de cero o uno [1].

Objetivo

Diseñar un controlador para el sistema de posición inicial por medio del método de diseño con memoria y direccionamiento de entrada-estado.

Descripción

Primero se diseña una carta ASM para el sistema de posición inicial por medio del método de diseño con memoria y direccionamiento entrada-estado. Posteriormente se propondrá una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P13.1 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema de posición inicial se necesitan las siguientes señales de entrada:

- un botón detecta la solicitud para que el sistema comience a funcionar o para regresar a la posición inicial de éste
- se necesitan sensores para detectar la posición inferior y superior del sistema
- se necesita un sensor de final de carrera en el extremo izquierdo del sistema
- un sensor verifica si el gripper está sujetando una caja.

Como salida se requieren de las siguientes señales:

- activación del motor para el movimiento vertical
- una señal se activa para mover el gripper a la posición superior del sistema
- activación del motor para el movimiento horizontal
- una señal se activa para mover el gripper hacia el extremo izquierdo del sistema
- activación del motor del gripper
- una señal se activa para que el gripper se abra y otra para que se cierre.

Tabla P13.1 Entradas y salidas para el sistema de posición inicial.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
BH	A0 del PIC	I	Botón de Home para ir a la posición inicial del sistema
SVS	A1 del PIC	I	Sensor de posición superior
SVI	A2 del PIC	I	Sensor de posición inferior
SHI	A3 del PIC	I	Sensor de final de carrera izquierda
SG	A4 del PIC	I	Sensor de gripper que indica cuándo el gripper está sujetando una caja
MV	D0 de la memoria 2	O	Motor vertical paso a paso para el movimiento vertical
M_S	D1 de la memoria 2	O	Señal para hacer que el gripper suba
MH	D2 de la memoria 2	O	Motor horizontal de CD movimiento horizontal
M_I	D3 de la memoria 2	O	Señal para hacer que el gripper se mueva a la izquierda
MG	D4 de la memoria 2	O	Motorreductor del gripper
M_A	D5 de la memoria 2	O	Señal para abrir el gripper.

Notas de diseño

a) Para iniciar iniciar el movimiento de la banda si se acaba de encender el sistema, se debe oprimir el botón de home.
b) Para que funcione de nuevo el sistema si se oprimió el botón de paro, se debe oprimir el botón de home.
c) Se considera la posición inicial del sistema cuando el gripper está abierto, en la posición superior y en el extremo izquierdo del sistema.

Reglas de funcionamiento

- BH: botón de home
  1 = se oprimió el botón de home
  0 = no se oprimió el botón de home
- SVS: sensor de posición superior
  1 = el gripper está en la posición superior
  0 = no está el gripper en la posición superior
- SVI: sensor de posición inferior
  1 = el gripper está en la posición inferior del sistema
  0 = no está el gripper en la posición inferior del sistema
- SHI: sensor de final de carrera izquierda
  1 = el gripper está en el extremo izquierdo del sistema
  0 = no está el gripper en el extremo izquierdo del sistema
- SG: sensor de gripper
  1 = el gripper está sujetando una caja o cerrado
  0 = no está el gripper sujetando una caja o abierto.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Si el valor de la entrada es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera. El algoritmo de la máquina de estados se puede ver en la figura P13.4.

Estado '000' – INICIO

El sistema se dirige a estado, después de que se deja de presionar el botón de paro o cuando se acaba de encender. Cuando el botón (BH) es oprimido, el sistema avanza al Estado '001' para verificar la posición vertical del gripper. De lo contrario, permanece en el Estado '000', en espera de que sea oprimido.

Estado '001' – DETPV

En estado se verifica la posición vertical del sistema. Si el sensor (SVS), detecta al gripper en la posición vertical superior, el sistema avanza al Estado '011' para detectar si el gripper está en alguno de los extremos horizontales del sistema. De lo contrario, avanza al Estado '010' para subir el gripper.

Estado '010' – SUBIRG

En este estado se activa el motor paso a paso (MV) y la señal (M_S) para que dicho motor suba el gripper a la posición superior del sistema. Cuando el sensor (SVS), detecte que el gripper ha llegado a la posición superior, el sistema avanza al Estado '011' para detectar si el gripper está en alguno de los extremos horizontales del sistema. De lo contrario, permanece en el Estado '010' subiendo el gripper.

Estado '011' – DETPH

En estado se verifica si el gripper está en alguno de los extremos horizontales del sistema. Si el sensor (SHI), detecta que el gripper se encuentra en el final de carrera izquierda, el sistema avanza al Estado '101' para detectar si el gripper está abierto o cerrado. De lo contrario, avanza al Estado '100', para mover el gripper al extremo izquierdo del sistema.

Estado '100' – MOVGI

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_I) para mover el gripper al extremo izquierdo del sistema. Cuando el sensor (SHI), detecta que el gripper ha llegado al límite izquierdo del sistema, el sistema avanza al Estado '101' para detectar si el gripper está abierto o cerrado. De lo contrario, permanece en el Estado '100' desplazando el gripper hacia la izquierda.

Estado ‘101’ – DETG

En estado se verifica si el gripper está abierto o cerrado. Si el sensor (SG) detecta que el gripper está abierto, regresa al Estado ‘000’ para iniciar nuevamente el proceso. De lo contrario, el sistema avanza al Estado ‘110’ para bajar el gripper a la posición inferior y soltar la caja que está sujetando.

Estado ‘110’ – BAJARG

En este estado se activa el motor paso a paso (MV) y debido a que no se activa la señal (M_S) para subir el gripper, éste desciende a la posición inferior del sistema. Cuando el sensor (SVI), detecta que el gripper ha llegado a la posición inferior, el sistema avanza al Estado ‘111’ para abrir el gripper y por lo tanto soltar la caja. De lo contrario, permanece en el Estado ‘110’ bajando el gripper.

Estado ‘111’ – ABRIRG

En este estado se activa el motor del gripper (MG) y la señal (M_A) para que el gripper se abra y suelte la caja. Cuando el sensor del gripper (SG), detecte que se ha abierto el gripper completamente, el sistema regresa al Estado ‘010’ para desplazar el gripper a su posición inicial. De lo contrario, permanece en el Estado ‘111’ abriendo el gripper.

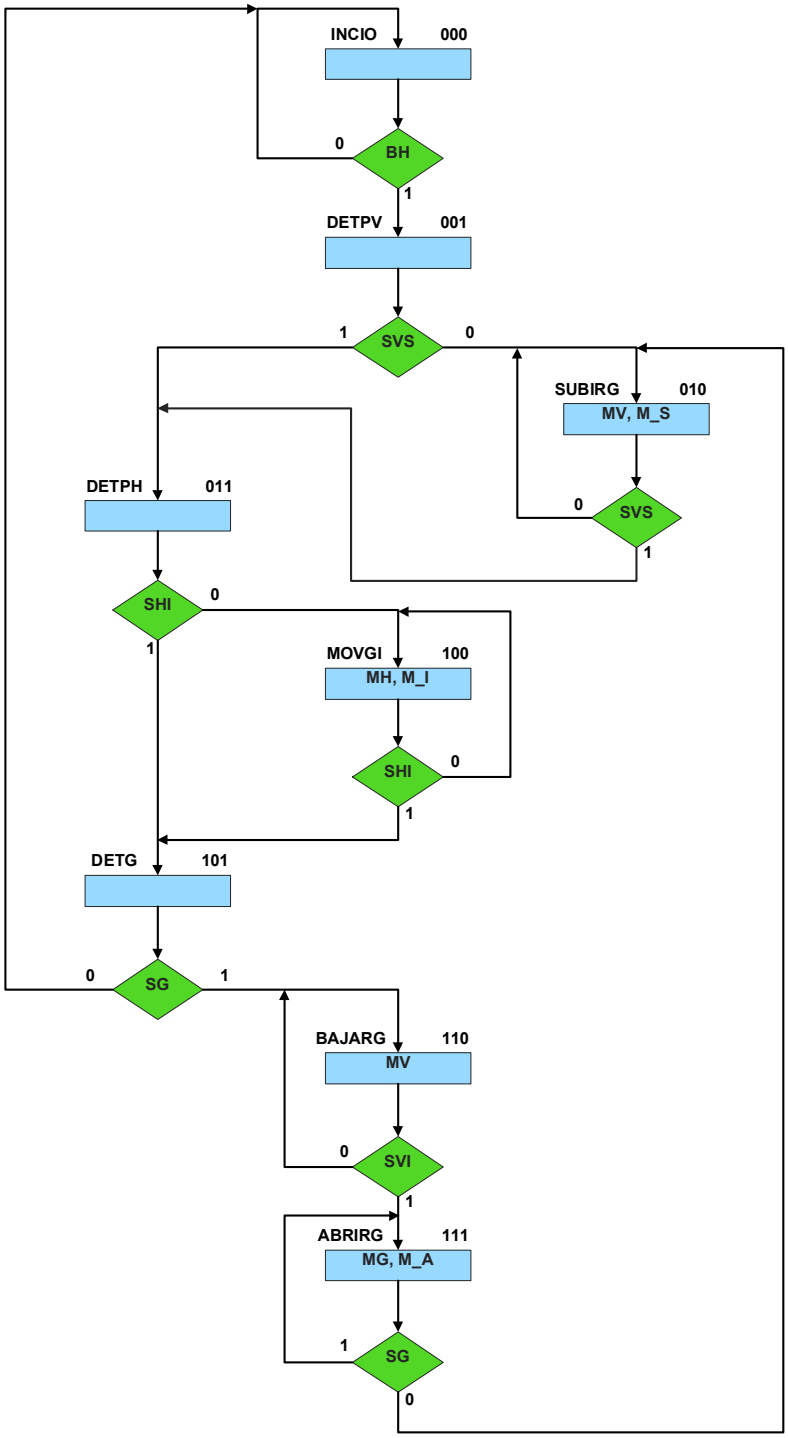


Figura P13.4 Carta ASM del sistema de posición inicial.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P13.2).

Tabla P13.2 Representación binaria de entradas para el sistema de posición inicial.

Entrada	Prueba
BH	0 0 0
SVS	0 0 1
SVI	0 1 0
SHI	0 1 1
SG	1 0 0

Se debe llenar la tabla P13.3 con base en la información de la carta ASM de la figura P13.4, usando el método de diseño con memoria y direccionamiento de entrada-estado.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘000’.

En el Estado ‘000’ se selecciona la entrada BH, por lo tanto, se coloca en el campo de prueba de la memoria su representación binaria, es decir, ‘000’. Si BH es igual a cero, el estado siguiente es el Estado ‘000’, su representación binaria ‘0000’ es colocada en el campo de la liga falsa. Si BH es igual a uno, el estado siguiente es el Estado ‘001’, su representación binaria ‘001’ es colocada en el campo de la liga verdadera. En el Estado ‘000’ no se activa ninguna señal de salida, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria, ver figura P13.2). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P13.3 Contenido de la memoria para el sistema de posición inicial.

Dirección de memoria			Contenido de memoria																
Estado presente			Prueba			Liga Falsa			Liga Verdadera		Liga Verdadera	Salidas						Hex 1	Hex 2
QA	QB	QC	I2	I1	I0	LF2	LF1	LF0	LV2	LV1	LV0	M_A	MG	M_I	MH	M_S	MV		
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	00	40
0	0	1	0	0	1	0	1	0	0	1	1	0	0	0	0	0	0	29	40
0	1	0	0	0	1	0	1	0	0	1	1	0	0	0	0	1	1	29	43
0	1	1	0	1	1	1	0	0	1	0	1	0	0	0	0	0	0	72	40
1	0	0	0	1	1	1	0	0	1	0	1	0	0	1	1	0	0	72	4C
1	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	83	00
1	1	0	0	1	0	1	1	0	1	1	1	0	0	0	0	0	1	5B	41
1	1	1	1	0	0	0	1	0	1	1	1	1	1	0	0	0	0	8B	70

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de ocho líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de éste (ver tabla P13.4). Se puede comprobar realizando el circuito de la figura P13.5.

Tabla P13.4 Tabla de funcionamiento del selector de entradas para el sistema de posición inicial.

Prueba			SI
I1	I1	I0	
0	0	0	BH
0	0	1	SVS
0	1	0	SVI
0	1	1	SHI
1	0	0	SG

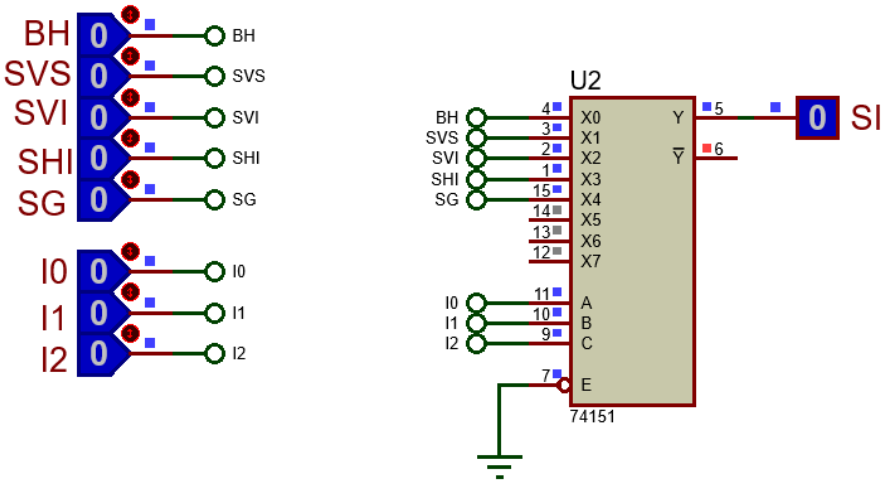


Figura P13.5 Multiplexor 74151 para selector de entradas del sistema de posición inicial.

La función booleana del selector de entradas queda:

```
SI = BH&!I2&!I1&!I0 | SVS&!I2&!I1&I0 | SVI&!I2&I1&!I0 | SHI&!I2&I1&I0 |
     SG&I2&!I1&!I0;
```

El selector de liga es un multiplexor triple de dos líneas a una, éste es implementado por el PIC16F1939. Si el selector entradas es igual a ‘1’, se selecciona la información binaria de la liga verdadera. Si el selector entradas es igual a ‘0’, se selecciona la información binaria de la liga falsa. Se puede comprobar realizando el circuito de la figura P13.6.

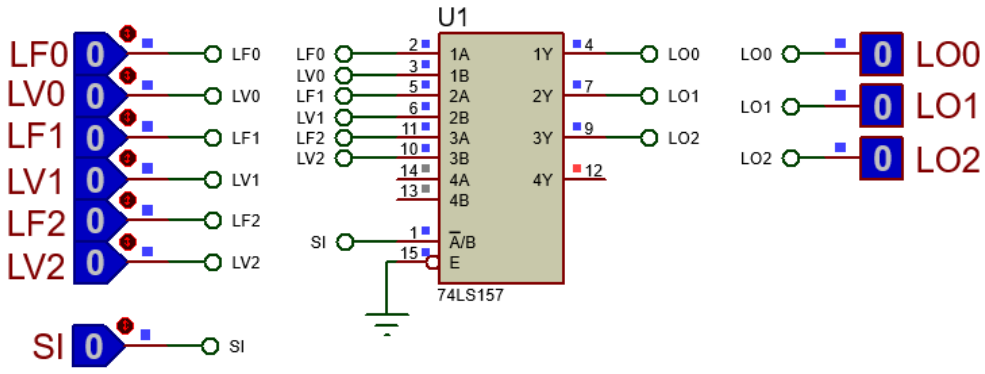


Figura P13.6 Multiplexor 74LS157 para el selector de liga para el sistema de posición inicial.

Las funciones booleanas del selector de liga quedan:

```
LO0 = !SI&LF0 | SI&LV0;

LO1 = !SI&LF1 | SI&LV1;

LO2 = !SI&LF2 | SI&LV2;
```

La explicación para obtener las expresiones que controlan el motor paso a paso se encuentra en la Práctica 12.

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P13.7,figura P13.8). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939. Para poder visualizar de manera más rápida el movimiento del motor paso a paso, se deberá utilizar una frecuencia de 30 Hz y el ángulo en el que gira dicho motor será de 3.6°.

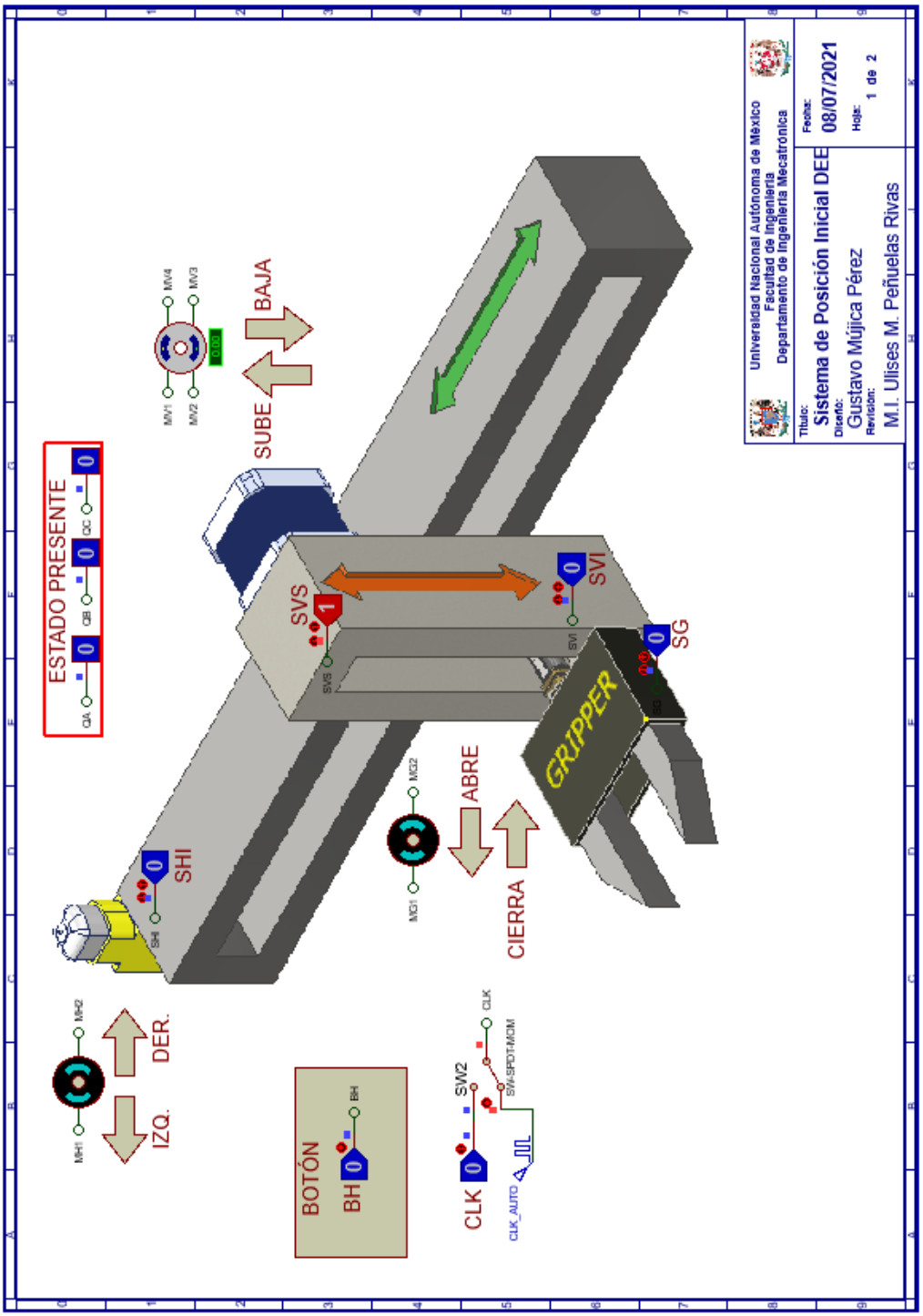


Figura P13.7 Interfaz hombre-máquina para el controlador de la Práctica 13 hoja 1/2.

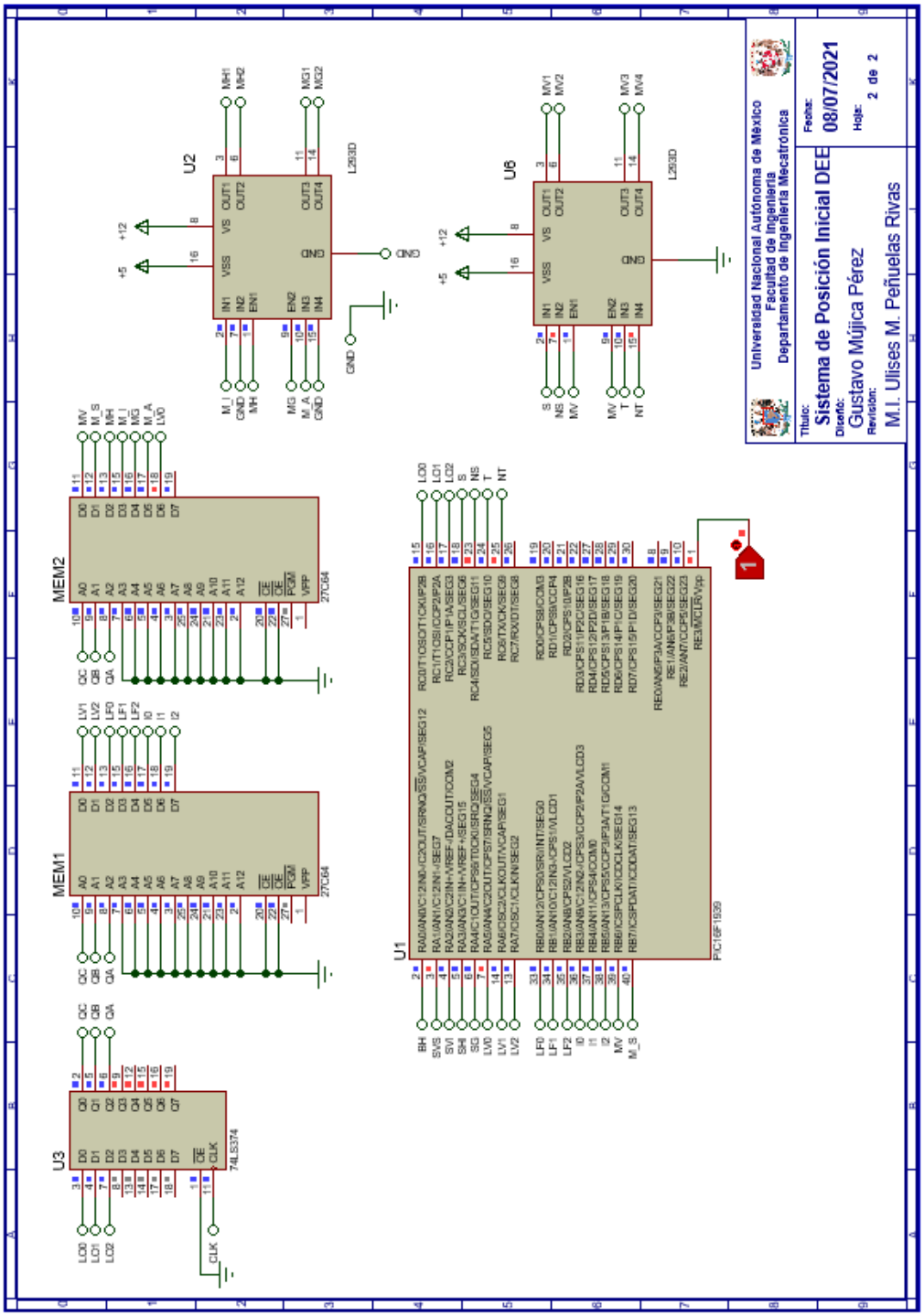


Figura P13.8 Esquema electrónico para el controlador de la Práctica 13 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P13.9, figura P13.10) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     //Carga biblioteca PLD.h

//***ENTRADAS***

#define BH  A0 //ENTRADA
#define SVS A1 //ENTRADA
#define SVI A2 //ENTRADA
#define SHI A3 //ENTRADA
#define SG  A4 //ENTRADA

//LIGAS VERDADERAS
#define LV0 A5
#define LV1 A6
#define LV2 A7

//LIGAS FALSAS
#define LF0 B0
#define LF1 B1
#define LF2 B2

//PRUEBAS
#define I0 B3 //PRUEBA
#define I1 B4 //PRUEBA
#define I2 B5 //PRUEBA

//***SALIDAS***
#define LO0 C0
#define LO1 C1
#define LO2 C2

//****MOTOR A PASOS****
//ENTRADAS
#define MVn  B6 //MV
#define M_Sn B7 //M_S

//SALIDAS
#define S  C3
#define NS C4
#define T  C5
#define NT C6

//***VARIABLES INTERMEDIAS***

short SI; //SELECTOR DE ENTRADA
short Sn, Tn; // Variables intermedias motor a pasos

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD
```

Figura P13.9 Código de la Práctica 13 parte 1.

```
50: pld_ini();  // INICIALIZA AL PIC COMO PLD
51: pld_555(30);   // Genera señal cuadrada en Hz,
52:                  //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
53:
54: //LOOP INFINITO
55:    while(1)
56:    {
57:
58:    //SELECTOR DE ENTRADAS
59:   SI = BH&!I2&!I1&!I0 | SVS&!I2&!I1&I0 | SVI&!I2&I1&!I0 |
60:        SHI&!I2&I1&I0 | SG&I2&!I1&!I0;
61:
62:    //SELECTOR DE LIGA
63:    LO0= !SI&LF0 | SI&LV0;
64:    LO1= !SI&LF1 | SI&LV1;
65:    LO2= !SI&LF2 | SI&LV2;
66:
67:     //****CIRCUITO SECUENCIAL ****
68:
69:    if (!out_555) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO,
70:                    //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
71:
72:    {
73:
74:    //MOTOR A PASOS
75:    Sn= (!T&!M_Sn | T&M_Sn)&MVn;
76:    Tn= (!S&M_Sn | S&!M_Sn)&MVn;
77:    }
78:
79:    else
80:    {//SECCIÓN DE MEMORIZACIÓN
81:    S=Sn;
82:    T=Tn;
83:
84:    NS=!S;
85:    NT=!T;
86:
87:    }
88:
89:    }
90: }
```

Figura P13.10 Código de la Práctica 13 parte 2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 14 Sistema clasificador; diseño con memoria y direccionamiento entrada-estado

Introducción

A través de una maqueta, se simuló el funcionamiento de un sistema robotizado que clasifica paquetes o cajas por colores y debe ser capaz de reconocer tres tipos de paquetes por su color colocados en una banda transportadora.

Los elementos principales del sistema son: un robot, un sensor de color y una banda transportadora. Inicialmente el sistema está en reposo, para activar el sistema se debe presionar un botón, al presionarlo la banda transportadora empezará a desplazar un paquete, un sensor detectará cuando el paquete esté en la posición adecuada para evaluar su color, por lo que la banda se detendrá.

El robot sujeta los paquetes y los deposita en el contenedor correspondiente. Los colores serán rojo, azul y verde. Los paquetes que no sean de estos colores, los dejará pasar (ver figura P14.1).

El robot se mueve a lo largo del eje horizontal por medio de un motor DC, para desplazarse en el eje vertical lo hace por medio de un motor paso a paso. El robot cuenta con un gripper, el cual es el encargado de sujetar los paquetes. El gripper está conformado por un motor DC.

El sistema tiene un botón de emergencia, cuando éste sea oprimido el sistema quedará inmóvil, hasta que se oprima otro botón para dirigirse a su posición inicial. Se consideró la posición inicial del sistema cuando la articulación horizontal está en su extremo izquierdo, la articulación vertical en su posición superior y con el gripper abierto.

El sistema se dividió en 2 partes para facilitar su comprensión. El sistema de posición inicial controla cuando el gripper debe moverse en su posición inicial y el sistema clasificador, controla la banda y la clasificación de los paquetes por colores. La unión de los dos sistemas mencionados anteriormente forma el sistema robotizado clasificador de paquetes.

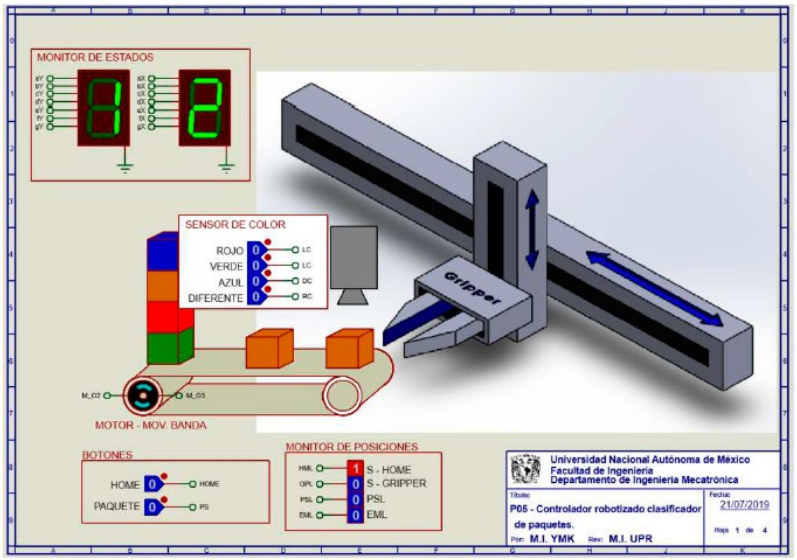


Figura P14.1 Sistema robotizado para clasificar paquetes por colores.

El diseño con memoria y direccionamiento entrada-estado restringe las cartas ASM a una sola entrada por estado. Una nueva porción de la palabra de memoria contiene una representación binaria de la entrada a probar en cada estado, esta parte es llamada “la parte de prueba”. Con esta representación binaria un selector de entrada elige una de las variables de entrada (ver figura P14.2) [1].

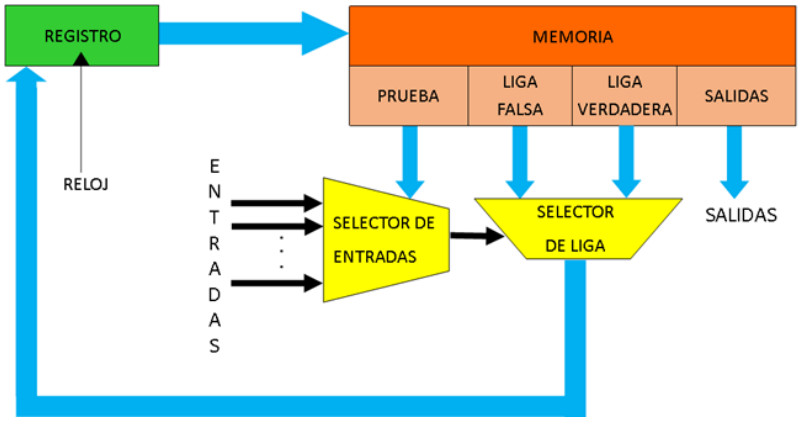


Figura P14.2 Arquitectura de un controlador con memoria y direccionamiento entrada-estado.

La parte de liga tiene dos estados siguientes, escogiéndose uno por el selector de liga, con base en la entrada seleccionada por la parte de prueba. Si el valor de la entrada seleccionada por el selector de entradas es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera (ver figura P14.3) [1].

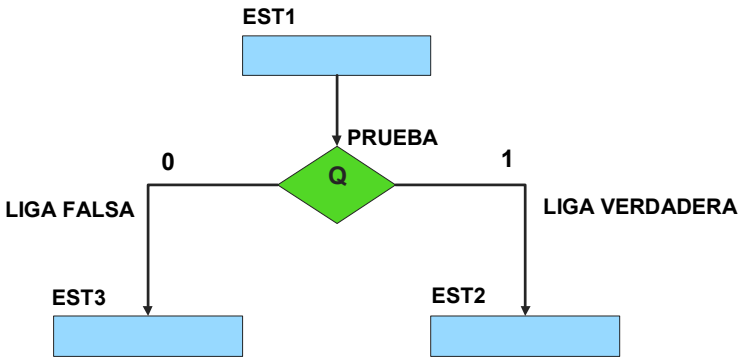


Figura P14.3 La liga falsa es el estado EST3, mientras que la liga verdadera es el estado EST2.

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará también una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista, se probará la variable auxiliar, la cual tiene un valor preestablecido de cero o uno [1].

Objetivo

Diseñar un controlador para el sistema clasificador por medio del método de diseño con memoria y direccionamiento entrada-estado.

Descripción

Primero se diseña una carta ASM para el sistema clasificador por medio del método de diseño con memoria y direccionamiento entrada-estado. Posteriormente se propone una solución para guardar un registro si se detecta una caja de color verde o rojo. Luego se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P14.1 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema clasificador se necesitan señales de entrada:

- sensores para detectar la posición inferior y superior del sistema
- un sensor para detectar el final de carrera derecho
- un sensor indica la posición donde se debe colocar la caja verde
- un sensor verifica si se cerró el gripper para tomar una caja
- un sensor detecta una caja en la banda
- un sensor le indica al sistema donde se encuentra la banda
- un sensor detecta el color de la caja.

Como salida se requieren las siguientes señales:

- activación del motor de la banda
- activación del motor para el movimiento vertical
- una señal se activa para mover el gripper a la posición superior del sistema
- activación del motor para el movimiento horizontal
- una señal se activa para mover el gripper hacia el extremo derecho del sistema y otra para el extremo izquierdo
- activación del motor del gripper
- una señal se activa para que el gripper se cierre.

Tabla P14.1 Entradas y salidas para el sistema clasificador.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
SVS	A0 del PIC	I	Sensor de posición superior
SVI	A1 del PIC	I	Sensor de posición inferior
SHD	A2 del PIC	I	Sensor de final de carrera derecha
SPCV	A3 del PIC	I	Sensor que indica la posición donde se debe colocar la caja verde
SG	A4 del PIC	I	Sensor de gripper que indica cuando el gripper está sujetando una caja
SP	A5 del PIC	I	Sensor de caja que la detecta y detiene la banda
SPB	A6 del PIC	I	Sensor de posición banda que le indica al sistema dónde está la banda, para detenerse y dejar de desplazarse hacia la derecha
SC-R	A7 del PIC	I	Señal del sensor de color para detectar una caja de color rojo
SC-V	B0 del PIC	I	Señal del sensor de color para detectar una caja de color verde
SC-A	B1 del PIC	I	Señal del sensor de color para detectar una caja de color azul
SC-O	B2 del PIC	I	Señal del sensor de color para detectar una caja de otro color
BCR	Interna	I	Señal indicadora que se detectó una caja color roja
BCV	Interna	I	Señal indicadora que se detectó una caja color verde.
M_I	D0 de la memoria 3	O	Señal para hacer que el gripper se mueva a la izquierda
MB	D1 de la memoria 3	O	Motor de la banda
MV	D2 de la memoria 3	O	Motor vertical paso a paso del movimiento vertical
M_S	D3 de la memoria 3	O	Señal para hacer que el gripper suba
MH	D0 de la memoria 2	O	Motor horizontal de CD del movimiento horizontal
M_D	D1 de la memoria 2	O	Señal para hacer que el gripper se mueva a la derecha
MG	D2 de la memoria 2	O	Motorreductor del gripper
M_C	D3 de la memoria 2	O	Señal para cerrar el gripper.

Notas de diseño

a) Se considera la posición inicial del sistema cuando el gripper está abierto, en la posición superior y en el extremo izquierdo del sistema.
b) Se considera que el sensor de color sólo podrá detectar colores cuando el gripper éste en la posición superior del sistema, arriba de la caja que va a detectar.
c) Las cajas rojas serán llevadas al extremo derecho del sistema.
d) Las cajas verdes serán llevadas a una posición intermedia del sistema, antes de llegar al extremo izquierdo.
e) Las cajas azules serán llevadas al extremo izquierdo del sistema.
f) El gripper sólo podrá desplazar una caja cuando éste se encuentre en la parte superior del sistema.

Reglas de funcionamiento

- SVS: sensor de posición superior
  1 = el gripper está en la posición superior
  0 = no está el gripper en la posición superior
- SVI: sensor de posición inferior
  1 = el gripper está en la posición inferior del sistema
  0 = no está el gripper en la posición inferior del sistema
- SHD: sensor de final de carrera derecha
  1 = el gripper está en el extremo derecho del sistema
  0 = no está el gripper en el extremo derecho del sistema
- SPCV: sensor posición caja verde
  1 = el gripper está en la posición para soltar la caja verde
  0 = no está el gripper en la posición para soltar la caja verde
- SG: sensor de gripper
  1 = el gripper está sujetando una caja o cerrado
  0 = no está el gripper sujetando una caja o abierto

- SP: sensor de caja
  1 = se detecta una caja en la banda
  0 = no se detecta una caja en la banda
- SPB: sensor de posición banda
  1 = se detecta que el gripper está alineado con la banda
  0 = no se detecta que el gripper está alineado con la banda
- SC-R: caja de color rojo
  1 = se detecta una caja de color rojo
  0 = no se detecta una caja de color rojo
- SC-V: caja de color verde
  1 = se detecta una caja de color verde
  0 = no se detecta una caja de color verde
- SC-A: caja de color azul
  1 = se detecta una caja de color azul
  0 = no se detecta una caja de color azul
- SC-O: caja de otro color
  1 = se detecta una caja de otro color
  0 = no se detecta una caja de otro color
- BCR: registro de caja color rojo
  1 = se detectó una caja color rojo
  0 = no se detectó una caja color rojo
- BCV: registro de caja color verde
  1 = se detectó una caja color verde
  0 = no se detectó una caja color verde

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Si el valor de la entrada es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera. El algoritmo de la máquina de estados se puede ver en la figura P14.4.

Estado '0000' – MB

En este estado se activa el motor de la banda (M_B). Si en este estado el sensor (SP) no detecta una caja, el sistema avanza al Estado '0001' para continuar moviendo la banda hasta que se detecte una. De lo contrario, permanece en el Estado '0000' moviendo la banda, para que la caja no se tome en cuenta en la clasificación ya que es de un color distinto a los establecidos.

Estado '0001' – MBANDA

En este estado se activa el motor de la banda (M_B). Si el sensor (SP), detecta una caja avanza al Estado '0010' para detener la banda e iniciar el proceso de clasificación. De lo contrario, permanece en el Estado '0001' moviendo la banda.

Estado '0010' – MOVGD

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_D) para mover el gripper a la derecha. Cuando el sensor de posición de la banda (SPB), detecta que el sistema está alineado con la banda, avanza al Estado '0011' para detener el desplazamiento del gripper y detectar el color de la caja. De lo contrario, permanece en el Estado '0010' moviendo el gripper a la derecha.

Estado '0011' – SCOLOR

En este estado el gripper está en espera de que el sensor detecte el color de la caja. Cuando el sensor detecta el color de la caja, avanza al Estado '0100' para mover el gripper dependiendo del color. De lo contrario, permanece en el Estado '0011' en espera a que el sensor detecte el color de la caja.

Estado '0100' – DCOLOR

En este estado se determina a que posición se lleva la caja dependiendo el color de esta. Si el sensor de color detecta una caja roja, verde o azul el sistema avanza al Estado '0101' para recogerla (se guardará un registro si la caja es de color rojo o verde).

De lo contrario, regresa al Estado ‘0000’ para iniciar nuevamente el proceso de clasificación ya que la caja es de un color distinto a los establecidos.

Estado ‘0101’ – BAJARG

En este estado se activa el motor paso a paso (MV) y debido a que no se activa la señal (M_S) el motor paso a paso baja el gripper a la posición inferior del sistema. Cuando el sensor (SVI), detecta que el gripper ha llegado a la posición inferior del sistema, el sistema avanza al Estado ‘0110’ para cerrar el gripper y por lo tanto sujetar la caja. De lo contrario, permanece en el Estado ‘0101’ bajando el gripper.

Estado ‘0110’ – CERRARG

En este estado se activa el motor del gripper (MG) y la señal (M_C) para que el gripper se cierre. Cuando el sensor del gripper (SG), detecta que se ha cerrado el gripper completamente, el sistema avanza al Estado ‘0111’ para desplazar el gripper a la posición superior del sistema. De lo contrario, permanece en el Estado ‘0110’ cerrando el gripper.

Estado ‘0111’ – SUBIRG

En este estado se activa el motor paso a paso (MV) y la señal (M_S) para que el motor paso a paso suba el gripper a la posición superior del sistema. Si el sensor (SVS), detecta que el gripper ha llegado a la posición superior, el sistema avanza al Estado ‘1000’ para determinar si se debe mover el gripper a la derecha o a la izquierda. De lo contrario, permanece en el Estado ‘0111’ subiendo el gripper.

Estado ‘1000’ – MOVDOI

En este estado se determina a que posición mover el gripper. Si el sensor de color detectó una caja de color rojo, avanza al Estado ‘1001’ para mover el gripper a la derecha.

De lo contrario, el sistema avanza al Estado ‘1010’ para verificar si se detectó una caja color verde.

Estado ‘1001’ – MOVGD

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_D) para mover el gripper al extremo derecho del sistema. Si el sensor (SHD), detecta que el gripper ha llegado al límite derecho del sistema, avanza al Estado ‘0000’ para iniciar el proceso nuevamente. De lo contrario, permanece en el Estado ‘1001’ desplazando el gripper hacia la derecha.

Estado ‘1010’ – DETCV

En este estado si el sensor de color detectó una caja color verde, el sistema avanza al Estado ‘1011’ para mover el gripper hacia una posición intermedia antes de llegar al extremo izquierdo. De lo contrario, la caja es color azul, el sistema regresa al Estado ‘0000’ para empezar nuevamente el proceso de clasificación.

Estado ‘1011’ – MOVCV

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_I) para mover la caja verde a una posición intermedia antes de llegar al extremo izquierdo del sistema. Cuando el sensor (SPCV), detecta que el gripper ha llegado a la posición intermedia, el sistema regresa al Estado ‘0000’ iniciando el proceso de clasificación nuevamente. De lo contrario, permanece en el Estado ‘1011’ desplazando el gripper hacia la izquierda.

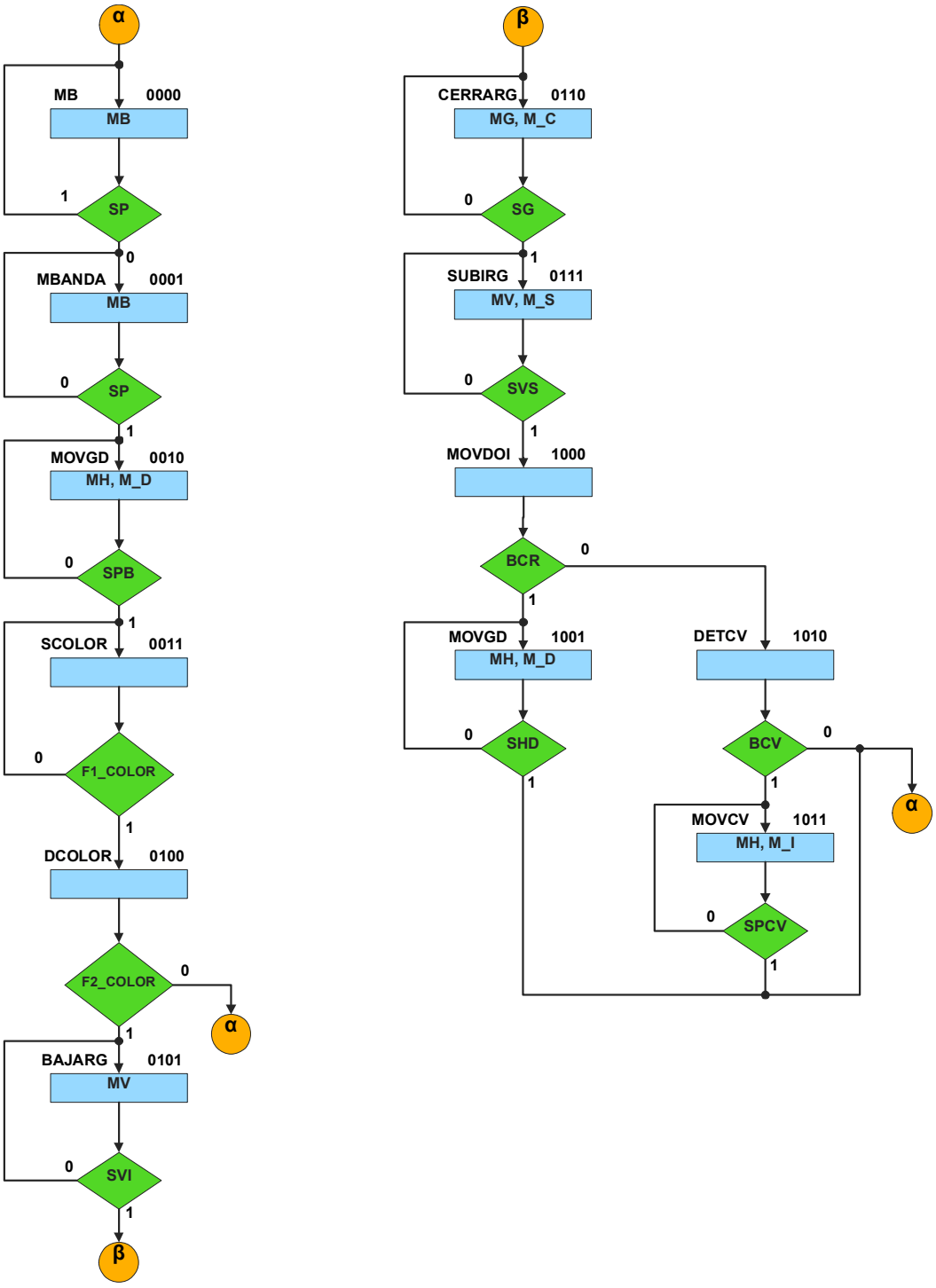


Figura P14.4 Carta ASM del sistema clasificador de paquetes.

Solución

Primero se resuelve por el método de variable suscrita, la máquina de estados para guardar un registro si se detecta una caja de color rojo o verde. Mientras no se detecte una caja color rojo o color verde, ambas máquinas permanecerán en el Estado '0'. Cuando se detecte una caja color rojo o verde, la máquina correspondiente avanza al Estado '1'. Cuando se detecte que el gripper ha llegado al extremo derecho del sistema o a la posición intermedia donde se colocará la caja verde, la máquina correspondiente regresa al Estado '0'. Se debe tener en cuenta que BCR= J y BCV=K, donde J y K son los bits de las máquinas de estados para guardar un registro si se detectó una caja roja o verde respectivamente. De esta manera se pueden relacionar las máquinas de estados (ver figura P14.5).

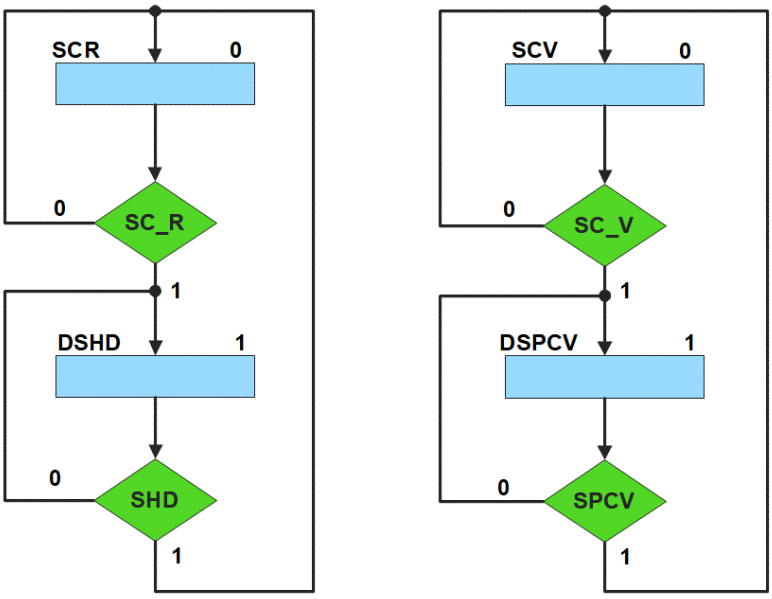


Figura P14.5 Cartas ASM para guardar un registro si se detecta una caja roja o verde.

El mapa de Karnaugh general de transición de estados para guardar un registro si se detecta una caja roja o verde se muestra en la tabla P14.2 y tabla P14.3.

Tabla P14.2 Mapa de Karnaugh general de transición de estados para guardar un registro si se detecta una caja roja.

	J	
	0	1
0	0	1
1	SC_R	SHD 0

Mapa de Karnaugh particular para el flip-flop del bit J es:

	J	
	0	1
	SC_R	!SHD

```
FFJ = !J&SC_R | J&!SHD;
```

Tabla P14.3 Mapa de Karnaugh general de transición de estados para guardar un registro si se detecta una caja verde.

	K	
	0	1
0	0	1
1	SC_V	SPCV 0

Mapa de Karnaugh particular para el flip-flop del bit K es:

	K	
	0	1
	SC_V	!SPCV

```
FFJ = !K&SC_V | K&!SPCV;
```

Para el sistema clasificador se debe asignar una representación binaria a cada variable de entrada (ver tabla P14.4).

Donde:

```
F1_COLOR = SC_R | SC_V | SC_A | SC_O;

F2_COLOR = SC_R | SC_V | SC_A;
```

Tabla P14.4 Representación binaria de entradas para el sistema clasificador.

Entrada	Prueba
SVS	0 0 0 0
SVI	0 0 0 1
SHD	0 0 1 0
SPCV	0 0 1 1
SG	0 1 0 0
SP	0 1 0 1
SPB	0 1 1 0
F1_COLOR	0 1 1 1
F2_COLOR	1 0 0 0
BCR	1 0 0 1
BCV	1 0 1 0

Se debe llenar la tabla P14.5 con base en la información de la carta ASM de la figura P14.4, usando el método de diseño con memoria y direccionamiento entrada-estado.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘0000’.

En el Estado ‘0000’ se selecciona la entrada SP y, por lo tanto, se coloca en el campo de prueba de la memoria su representación binaria, es decir, ‘0101’. Si SP es igual a cero, el estado siguiente es el Estado ‘0001’, su representación binaria ‘0001’ es colocada en el campo de la liga falsa. Si SP es igual a uno, el estado siguiente es el Estado ‘0000’, su representación binaria ‘0000’ es colocada en el campo de la liga verdadera. En el Estado ‘0000’ se activa la señal de salida MB, por lo que se coloca un ‘1’ en la parte de salidas de MB.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria, ver figura P14.2). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P14.5 Contenido de la memoria para el sistema clasificador.

Dirección de memoria				Contenido de memoria																						
Estado presente				Prueba				Liga Falsa				Liga Verdadera				Salidas				Salidas				Hex 1	Hex 2	Hex 3
QA	QB	QC	QD	I3	I2	I1	I0	LF3	LF2	LF1	LF0	LV3	LV2	LV1	LV0	M_C	MG	M_D	MH	M_S	MV	MB	M_I			
0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	51	00	02
0	0	0	1	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	51	20	02
0	0	1	0	0	1	1	0	0	0	1	0	0	0	1	1	0	0	1	1	0	0	0	0	62	33	00
0	0	1	1	0	1	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	73	40	00
0	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	80	50	00
0	1	0	1	0	0	0	1	0	1	0	1	0	1	1	0	0	0	0	0	0	1	0	0	15	60	04
0	1	1	0	0	1	0	0	0	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	46	7C	00
0	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1	1	0	0	07	80	0C
1	0	0	0	1	0	0	1	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	9A	90	00
1	0	0	1	0	0	1	0	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	29	03	00
1	0	1	0	1	0	1	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	A0	B0	00
1	0	1	1	0	0	1	1	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	1	3B	01	01

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de dieciséis líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P14.6). Se puede comprobar realizando el circuito de la figura P14.6.

Tabla P14.6 Tabla de funcionamiento del selector de entradas para el sistema clasificador.

Prueba				SI
I3	I2	I1	I0	
0	0	0	0	SVS
0	0	0	1	SVI
0	0	1	0	SHD
0	0	1	1	SPCV
0	1	0	0	SG
0	1	0	1	SP
0	1	1	0	SPB
0	1	1	1	F1_COLOR
1	0	0	0	F2_COLOR
1	0	0	1	BCR
1	0	1	0	BCV

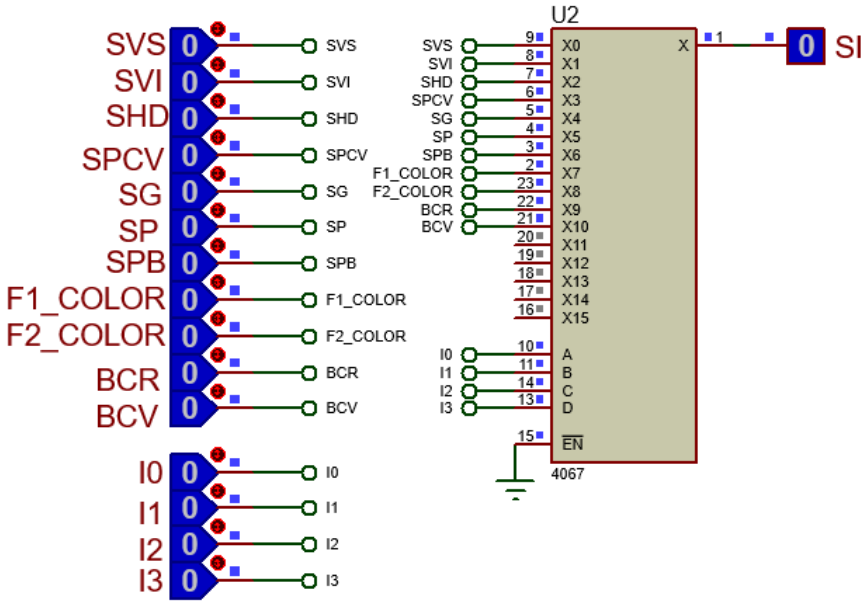


Figura P14.6 Multiplexor 4067 para el selector de entradas del sistema clasificador.

Teniendo en cuenta que MGD= J, donde J es el bit de la máquina de estados para guardar un registro si se detectó una caja roja o verde, la función booleana del selector de entradas queda:

```
SI = SVS&!I3&!I2&!I1&!I0 | SVI&!I3&!I2&!I1&I0 | SHD&!I3&!I2&I1&!I0 |
     SPCV&!I3&!I2&I1&I0 | SG&!I3&I2&!I1&!I0 | SP&!I3&I2&!I1&I0 |
     SPB&!I3&I2&I1&!I0 | F1_COLOR&!I3&I2&I1&I0 | F2_COLOR&I3&!I2&!I1&!I0
     | J&I3&!I2&!I1&I0 | K&I3&!I2&I1&!I0;
```

El selector de liga es un multiplexor cuádruple de dos líneas a una, éste es implementado por el PIC16F1939. Si el selector entradas es igual a '1', se selecciona la información binaria de la liga verdadera. Si el selector entradas es igual a '0', se selecciona la información binaria de la liga falsa. Se puede comprobar realizando el circuito de la figura P14.7.

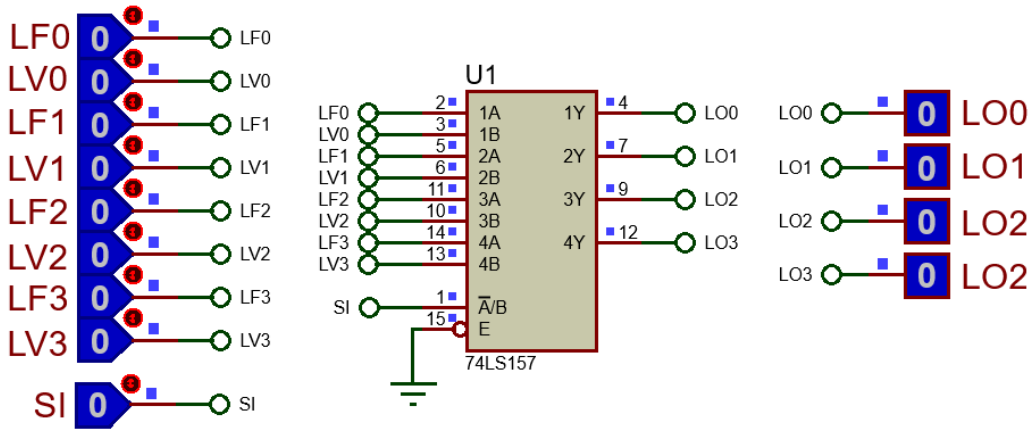


Figura P14.7 Multiplexor 74LS157 para el selector de liga del sistema clasificador.

Las funciones booleanas del selector de liga quedan:

```
LO0 = !SI&LF0 | SI&LV0;

LO1 = !SI&LF1 | SI&LV1;

LO2 = !SI&LF2 | SI&LV2;

LO3 = !SI&LF3 | SI&LV3;
```

La explicación para obtener las expresiones que controlan el motor paso a paso se encuentra en la Práctica 12.

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P14.8, figura P14.9). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939. Para poder visualizar de manera más rápida el movimiento del motor paso a paso, se debe utilizar una frecuencia de 30 Hz y el ángulo en el que gira el motor paso a paso es de 3.6°. Se modificó la biblioteca PLD.h para tener más puertos de entradas en el PIC.

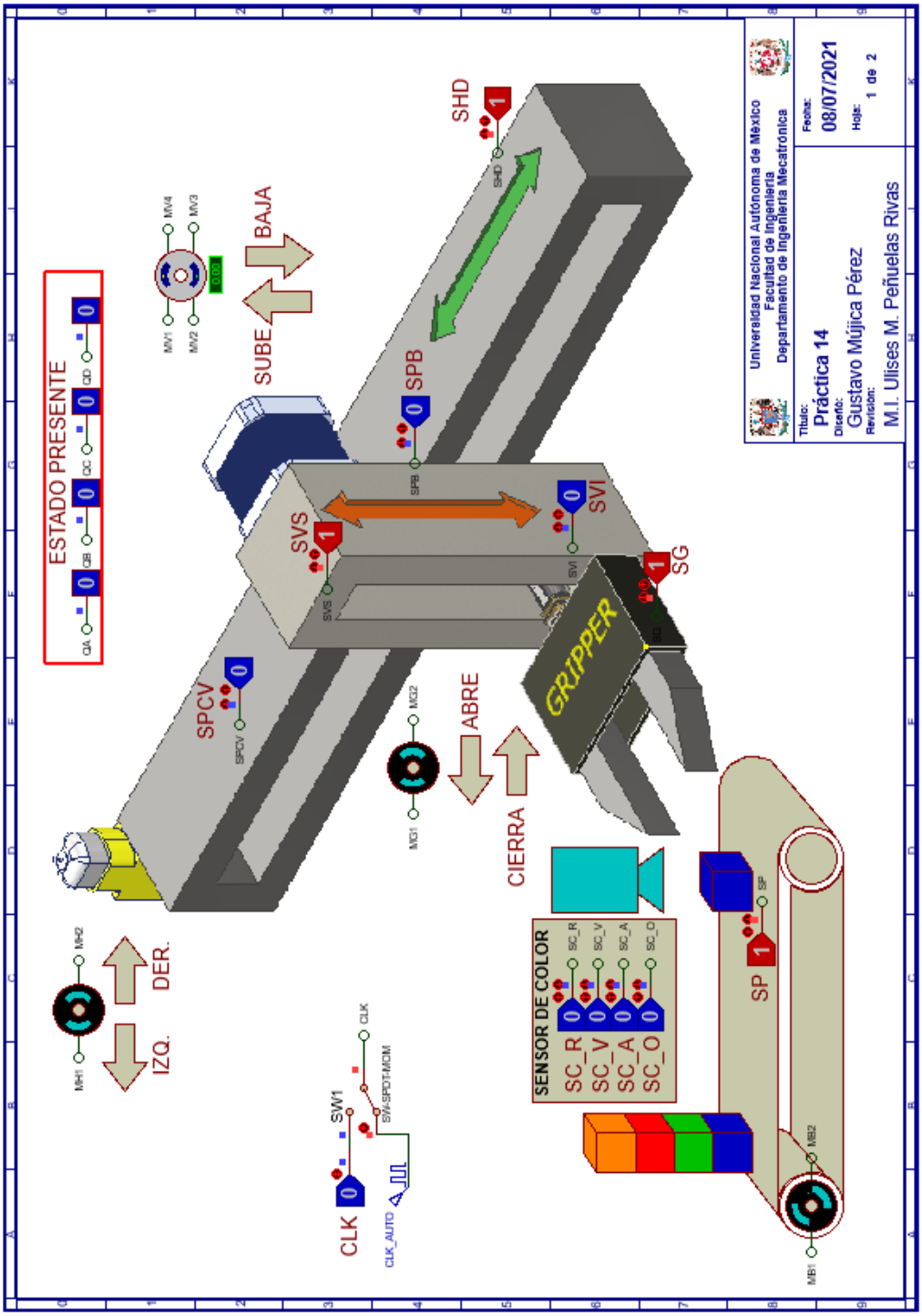


Figura P14.8 Interfaz hombre-máquina para el controlador de la Práctica 14 hoja 1/2.

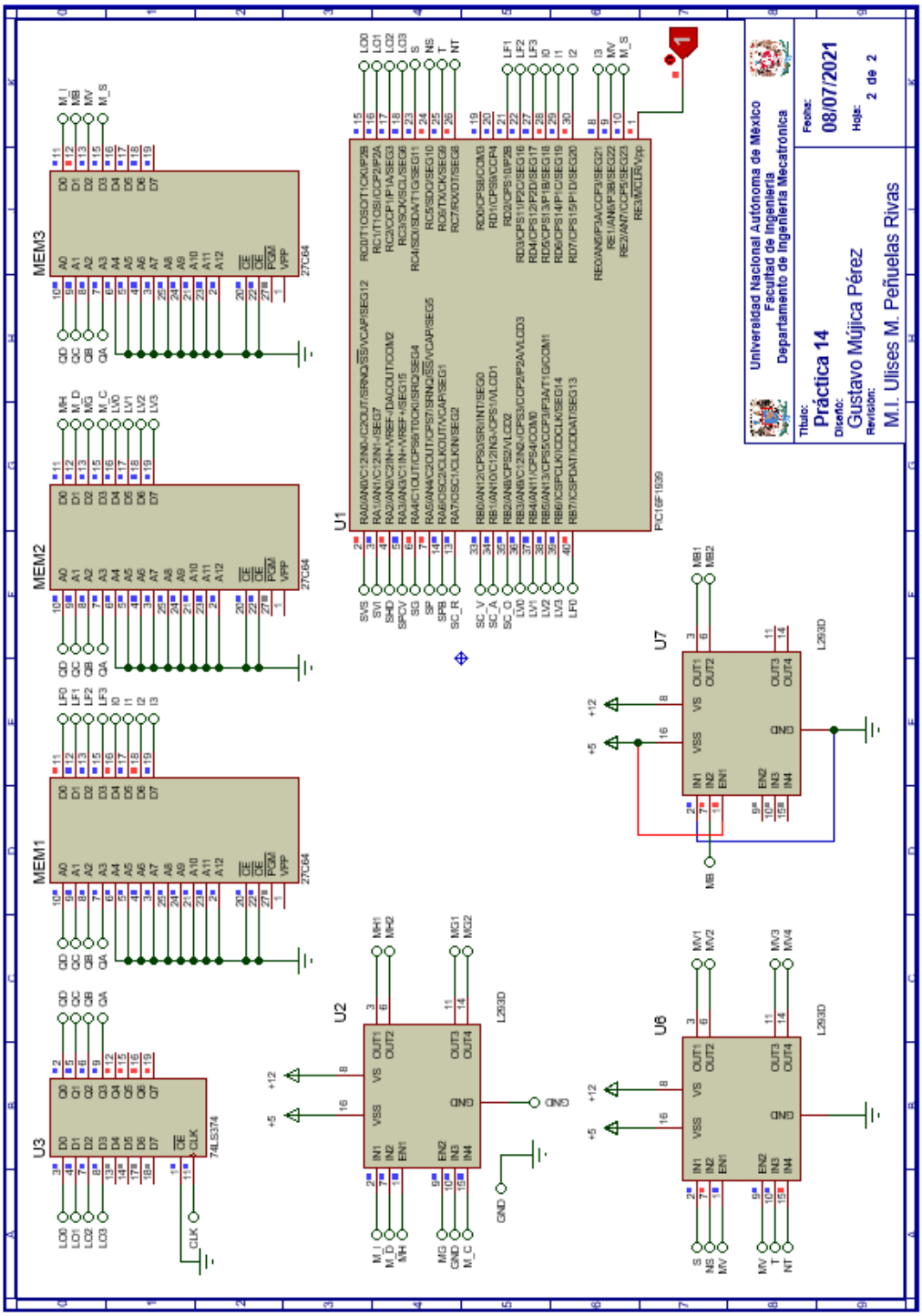


Figura P14.9 Esquema electrónico para el controlador de la Práctica 14 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P14.10 a figura P14.12) y se obtendrán archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD_SROBOTIZADO.h> // Carga biblioteca PLD_SROBOTIZADO.h

//***ENTRADAS***

#define SVS   A0
#define SVI   A1
#define SHD   A2
#define SPCV  A3
#define SG    A4
#define SP    A5
#define SPB   A6
#define SC_R  A7

#define SC_V  B0
#define SC_A  B1
#define SC_O  B2

//LIGAS VERDADERAS
#define LV0 B3
#define LV1 B4
#define LV2 B5
#define LV3 B6

//LIGAS FALSAS
#define LF0 B7

#define LF1 D2
#define LF2 D3
#define LF3 D4

//PRUEBAS
#define I0 D5
#define I1 D6
#define I2 D7

#define I3 E0

//***SALIDAS***
#define LO0 C0
#define LO1 C1
#define LO2 C2
#define LO3 C3

//****MOTOR A PASOS****
```

Figura P14.10 Código de la Práctica 14 parte 1.

```
45: //****MOTOR A PASOS****
46: //ENTRADAS
47: #define MVn  E1 //MV
48: #define M_Sn E2 //M_S
49:
50: //SALIDAS
51: #define S  C4
52: #define NS C5
53: #define T  C6
54: #define NT C7
55:
56: //***VARIABLES INTERMEDIAS***
57:
58: short SI; //SELECTOR DE ENTRADA
59: short Sn, Tn; // Variables intermedias motor a pasos
60: short J,K,Jn,Kn; // Variable auxiliar para registrar si se detectó una caja color verde o roja
61: short F1_COLOR, F2_COLOR; //Funciones
62:
63: void main ()
64: {
65: pld_ini();  // INICIALIZA AL PIC COMO PLD
66: pld_555(30);   // Genera señal cuadrada en Hz,
67:                  //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
68:
69: //LOOP INFINITO
70:    while(1)
71:    {
72:    //****CIRCUITO COMBINACIONAL ****
73:
74:    //FUNCIONES
75:    F1_COLOR= SC_R | SC_V | SC_A | SC_O;
76:    F2_COLOR= SC_R | SC_V | SC_A;
77:
78:    //SELECTOR DE ENTRADAS
79:    SI = SVS&!I3&!I2&!I1&!I0 | SVI&!I3&!I2&!I1&I0 | SHD&!I3&!I2&I1&!I0 | SPCV&!I3&!I2&I1&I0 |
80:         SG&!I3&I2&!I1&!I0 | SP&!I3&I2&!I1&I0 | SPB&!I3&I2&I1&!I0 | F1_COLOR&!I3&I2&I1&I0 |
81:         F2_COLOR&I3&!I2&!I1&!I0 | J&I3&!I2&!I1&I0 | K&I3&!I2&I1&!I0;
82:
83:    //SELECTOR DE LIGA
84:    LO0= !SI&LF0 | SI&LV0;
85:    LO1= !SI&LF1 | SI&LV1;
86:    LO2= !SI&LF2 | SI&LV2;
87:    LO3= !SI&LF3 | SI&LV3;
88:
89:     //****CIRCUITO SECUENCIAL ****
```

Figura P14.11 Código de la Práctica 14 parte 2.

```
 89:     //****CIRCUITO SECUENCIAL ****
 90:
 91:    if (!out_555) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO,
 92:                  //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
 93:
 94:    {
 95:
 96:    //MOTOR A PASOS
 97:    Sn= (!T&!M_Sn | T&M_Sn)&MVn;
 98:    Tn= (!S&M_Sn | S&!M_Sn)&MVn;
 99:    //REGISTRO CAJA COLOR ROJO O VERDE
100:    Jn= !J&SC_R | J&!SHD;
101:    Kn= !K&SC_V | K&!SPCV;
102:    }
103:
104:    else
105:    {//SECCIÓN DE MEMORIZACIÓN
106:    S=Sn;
107:    T=Tn;
108:
109:    NS=!S;
110:    NT=!T;
111:
112:    J=Jn;
113:    K=Kn;
114:    }
115:    }
116: }
```

Figura P14.12 Código de la Práctica 14 parte 3.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 15 Sistema clasificador; diseño con memoria y direccionamiento implícito

Introducción

A través de una maqueta, se simuló el funcionamiento de un sistema robotizado que clasifica paquetes o cajas por colores y debe ser capaz de reconocer tres tipos de paquetes por su color colocados en una banda transportadora.

Los elementos principales del sistema son: un robot, un sensor de color y una banda transportadora. Inicialmente el sistema está en reposo, para activar el sistema se debe presionar un botón, al presionarlo la banda transportadora empezará a desplazar un paquete, un sensor detectará cuando el paquete esté en la posición adecuada para evaluar su color, por lo que la banda se detendrá.

El robot sujeta los paquetes y los deposita en el contenedor correspondiente. Los colores serán rojo, azul y verde. Los paquetes que no sean de estos colores, los dejará pasar (ver figura P15.1).

El robot se mueve a lo largo del eje horizontal por medio de un motor DC, para desplazarse en el eje vertical lo hace por medio de un motor paso a paso. El robot cuenta con un gripper, el cual es el encargado de sujetar los paquetes. El gripper está conformado por un motor DC.

El sistema tiene un botón de emergencia, cuando éste sea oprimido el sistema quedará inmóvil, hasta que se oprima otro botón para dirigirse a su posición inicial. Se consideró la posición inicial del sistema cuando la articulación horizontal está en su extremo izquierdo, la articulación vertical en su posición superior y con el gripper abierto.

El sistema se dividió en 2 partes para facilitar su comprensión. El sistema de posición inicial controla cuando el gripper debe moverse en su posición inicial y el sistema clasificador, controla la banda y la clasificación de los paquetes por colores. La unión de los dos sistemas mencionados anteriormente forma el sistema robotizado clasificador de paquetes.

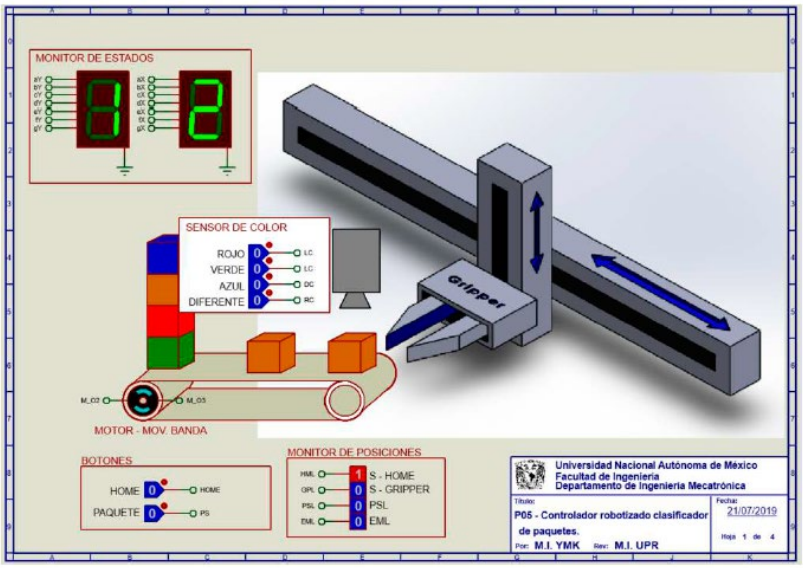


Figura P15.1 Sistema robotizado para clasificar paquetes por colores.

El diseño con memorias y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura P15.2). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura P15.3 muestra la arquitectura de este método.

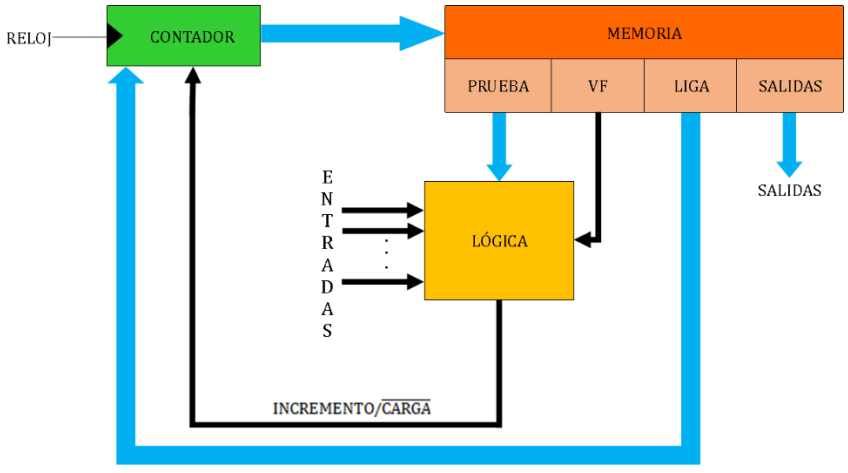


Figura P15.2 Arquitectura de un controlador con memoria y direccionamiento implícito.

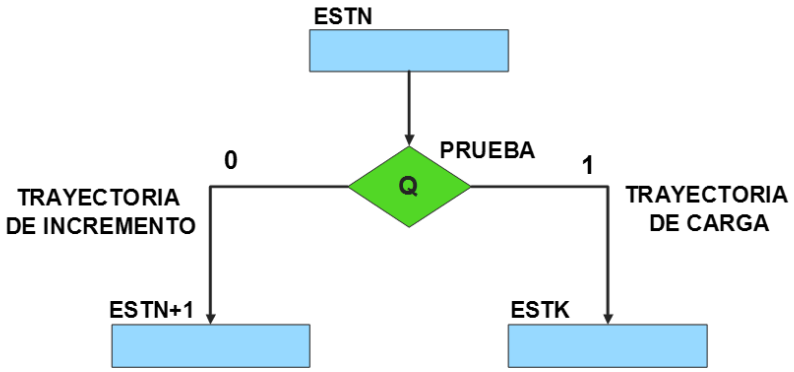


Figura P15.3 Trayectoria de incremento y carga.

La tabla P15.1 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que valor de entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un '1' se hace un incremento, cuando el resultado de la función da como resultado un '0' se hace una carga.

Tabla P15.1 Relación entre VF, variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incremento o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es '0', se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es '1', se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura P15.4).

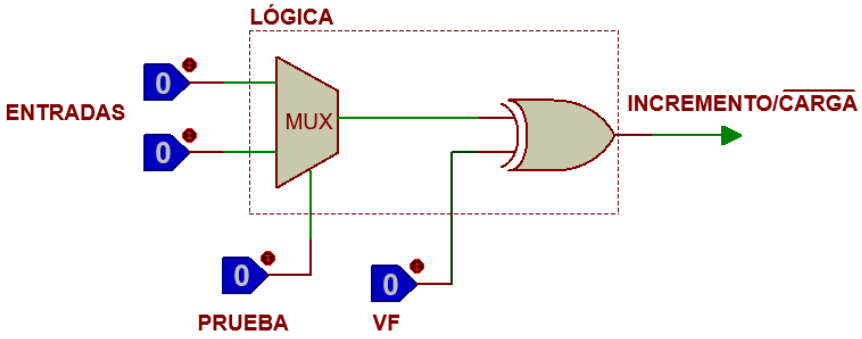


Figura P15.4 Bloque de lógica de incremento/carga para el direccionamiento implícito [1].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

Objetivo

Diseñar un controlador para el sistema clasificador por medio del método de diseño con memoria y direccionamiento implícito.

Descripción

Primero se diseña una carta ASM para el sistema clasificador por medio del diseño con memoria y direccionamiento implícito. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P15.2 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema clasificador se necesitan las siguientes señales de entrada:

- sensores para detectar la posición inferior y superior del sistema
- un sensor para detectar el final de carrera derecho
- un sensor indica la posición donde se debe colocar la caja verde
- un sensor verifica si se cerró el gripper para tomar una caja
- un sensor detecta una caja en la banda
- un sensor le indica al sistema donde se encuentra la banda
- un sensor detecta el color de la caja.

Como salida se requiere de las siguientes señales:

- activación del motor de la banda
- activación del motor para el movimiento vertical
- una señal se activa para mover el gripper a la posición superior del sistema
- activación del motor para el movimiento horizontal
- una señal se activa para mover el gripper hacia el extremo derecho del sistema y otra para el extremo izquierdo
- activación del motor del gripper
- una señal se activa para que el gripper se cierre.

Tabla P15.2 Entradas y salidas para el sistema clasificador.

Identificador	Ubicación	Tipo	Descripción
SVS	A0 del PIC	I	Sensor de posición superior
SVI	A1 del PIC	I	Sensor de posición inferior
SHD	A2 del PIC	I	Sensor de final de carrera derecha
SPCV	A3 del PIC	I	Sensor que indica la posición donde se debe colocar la caja verde
SG	A4 del PIC	I	Sensor de gripper que indica cuándo el gripper está sujetando una caja
SP	A5 del PIC	I	Sensor de caja que la detecta y detiene la banda
SPB	A6 del PIC	I	Sensor de posición banda que le indica al sistema dónde está la banda, para detenerse y dejar de desplazarse hacia la derecha
SC-R	A7 del PIC	I	Señal del sensor de color para detectar una caja de color rojo
SC-V	B0 del PIC	I	Señal del sensor de color para detectar una caja de color verde
SC-A	B1 del PIC	I	Señal del sensor de color para detectar una caja de color azul
SC-O	B2 del PIC	I	Señal del sensor de color para detectar una caja de otro color
BCR	Interna	I	Señal indicadora que se detectó una caja color roja
BCV	Interna	I	Señal indicadora que se detectó una caja color verde
MB	D0 de la memoria 2	O	Motor de la banda
MV	D1 de la memoria 2	O	Motor vertical paso a paso del movimiento vertical
M_S	D2 de la memoria 2	O	Señal para hacer que el gripper suba
MH	D3 de la memoria 2	O	Motor horizontal de CD del movimiento horizontal
M_D	D4 de la memoria 2	O	Señal para hacer que el gripper se mueva a la derecha
MG	D5 de la memoria 2	O	Motorreductor del gripper
M_C	D6 de la memoria 2	O	Señal para cerrar el gripper
M_I	D0 de la memoria 3	O	Señal para hacer que el gripper se mueva a la izquierda.

Notas de diseño

a) Se considera la posición inicial del sistema cuando el gripper está abierto, en la posición superior y en el extremo izquierdo del sistema.
b) Se considera que el sensor de color sólo podrá detectar colores cuando el gripper estÉ en la posición superior del sistema, arriba de la caja que va a detectar.
c) Las cajas rojas serán llevadas al extremo derecho del sistema.
d) Las cajas verdes serán llevadas a una posición intermedia del sistema, antes de llegar al extremo izquierdo.
e) Las cajas azules serán llevadas al extremo izquierdo del sistema.
f) El gripper sólo podrá desplazar una caja cuando esté se encuentre en la parte superior del sistema.

Reglas de funcionamiento

- SVS: sensor de posición superior
  1 = el gripper está en la posición superior
  0 = no está el gripper en la posición superior
- SVI: sensor de posición inferior
  1 = el gripper está en la posición inferior del sistema
  0 = no está el gripper en la posición inferior del sistema
- SHD: sensor de final de carrera derecha
  1 = el gripper está en el extremo derecho del sistema
  0 = no está el gripper en el extremo derecho del sistema
- SPCV: sensor posición caja verde
  1 = el gripper está en la posición para soltar la caja verde
  0 = no está el gripper en la posición para soltar la caja verde
- SG: sensor de gripper
  1 = el gripper está sujetando una caja o cerrado
  0 = no está el gripper sujetando una caja o abierto

- SP: sensor de caja
  1 = se detecta una caja en la banda
  0 = no se detecta una caja en la banda
- SPB: sensor de posición de la banda
  1 = se detecta que el gripper está alineado con la banda
  0 = no se detecta que el gripper está alineado con la banda
- SC-R: caja de color rojo
  1 = se detecta una caja de color rojo
  0 = no se detecta una caja de color rojo
- SC-V: caja de color verde
  1 = se detecta una caja de color verde
  0 = no se detecta una caja de color verde
- SC-A: caja de color azul
  1 = se detecta una caja de color azul
  0 = no se detecta una caja de color azul
- SC-O: caja de otro color
  1 = se detecta una caja de otro color
  0 = no se detecta una caja de otro color
- BCR: registro de caja color rojo
  1 = se detectó una caja color rojo
  0 = no se detectó una caja color rojo
- BCV: registro de caja color verde
  1 = se detectó una caja color verde
  0 = no se detectó una caja color verde.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Se hace un incremento cuando el valor del estado siguiente aumenta en una unidad, de lo contrario se hace una carga (ver figura P15.3). El algoritmo de la máquina de estados se puede ver en la figura P15.5.

Estado '0000' – MB

En este estado se activa el motor de la banda (M_B). Si en este estado, el sensor (SP) no detecta una caja, el sistema avanza al Estado '0001' para continuar moviendo la banda hasta que se detecte una. De lo contrario, permanece en el Estado '0000' moviendo la banda, para que la caja no se tome en cuenta en la clasificación ya que es de un color distinto a los establecidos.

Estado '0001' – MBANDA

En este estado se activa el motor de la banda (M_B). Si el sensor (SP), detecta una caja avanza al Estado '0010' para detener la banda e iniciar el proceso de clasificación. De lo contrario, permanece en el Estado '0001' moviendo la banda.

Estado '0010' – MOVGD

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_D) para mover el gripper a la derecha. Cuando el sensor de posición de la banda (SPB), detecta que el sistema está alineado con la banda, avanza al Estado '0011' para detener el desplazamiento del gripper y detectar el color de la caja. De lo contrario, permanece en el Estado '0010' moviendo el gripper a la derecha.

Estado '0011' – SCOLOR

En este estado el gripper está en espera de que el sensor detecte el color de la caja. Cuando el sensor detecta el color de la caja, avanza al Estado '0100' para mover el gripper dependiendo del color. De lo contrario, permanece en el Estado '0011' en espera a que el sensor detecte el color de la caja.

Estado '0100' – DCOLOR

En este estado se determina a que posición se lleva la caja dependiendo el color de esta. Si el sensor de color detecta una caja roja, verde o azul el sistema avanza al Estado '0101' para recogerla (se guardará un registro si la caja es de color rojo o verde).

De lo contrario, regresa al Estado ‘0000’ para iniciar nuevamente el proceso de clasificación ya que la caja es de un color distinto a los establecidos.

Estado ‘0101’ – BAJARG

En este estado se activa el motor paso a paso (MV) y debido a que no se activa la señal (M_S) el motor paso a paso baja el gripper a la posición inferior del sistema. Cuando el sensor (SVI), detecta que el gripper ha llegado a la posición inferior del sistema, el sistema avanza al Estado ‘0110’ para cerrar el gripper y por lo tanto sujetar la caja. De lo contrario, permanece en el Estado ‘0101’ bajando el gripper.

Estado ‘0110’ – CERRARG

En este estado se activa el motor del gripper (MG) y la señal (M_C) para que el gripper se cierre. Cuando el sensor del gripper (SG), detecta que se ha cerrado el gripper completamente, el sistema avanza al Estado ‘0111’ para desplazar el gripper a la posición superior del sistema. De lo contrario, permanece en el Estado ‘0110’ cerrando el gripper.

Estado ‘0111’ – SUBIRG

En este estado se activa el motor paso a paso (MV) y la señal (M_S) para que el motor paso a paso suba el gripper a la posición superior del sistema. Si el sensor (SVS), detecta que el gripper ha llegado a la posición superior, el sistema avanza al Estado ‘1000’ para determinar si se debe mover el gripper a la derecha o a la izquierda. De lo contrario, permanece en el Estado ‘0111’ subiendo el gripper.

Estado ‘1000’ – MOVDOI

En este estado se determina a que posición mover el gripper. Si el sensor de color detectó una caja de color rojo, avanza al Estado ‘1001’ para mover el gripper a la derecha. De lo contrario, el sistema avanza al Estado ‘1011’ para verificar si se detectó una caja color verde.

Estado ‘1001’ – MOVGD

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_D) para mover el gripper al extremo derecho del sistema.

Si el sensor (SHD), detecta que el gripper ha llegado al límite derecho del sistema, avanza al Estado ‘1010’ para posteriormente iniciar el proceso nuevamente. De lo contrario, permanece en el Estado ‘1001’ desplazando el gripper hacia la derecha.

Estado ‘1010’ – AUX1

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘0000’, haciendo una carga y así iniciar el proceso nuevamente.

Estado ‘1011’ – DETCV

En este estado si el sensor de color detectó una caja color verde, el sistema avanza al Estado ‘1100’ para mover el gripper hacia una posición intermedia antes de llegar al extremo izquierdo. De lo contrario, la caja es color azul, el sistema regresa al Estado ‘0000’ para empezar nuevamente el proceso de clasificación.

Estado ‘1100’ – MOVCV

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_I) para mover la caja verde a una posición intermedia antes de llegar al extremo izquierdo del sistema. Cuando el sensor (SPCV), detecta que el gripper ha llegado a la posición intermedia, el sistema avanza al Estado ‘1101’ para posteriormente poder avanzar al Estado ‘0000’. De lo contrario, permanece en el Estado ‘1100’ desplazando el gripper hacia la izquierda.

Estado ‘1101’ – AUX2

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘0000’, haciendo una carga y así iniciar el proceso nuevamente.

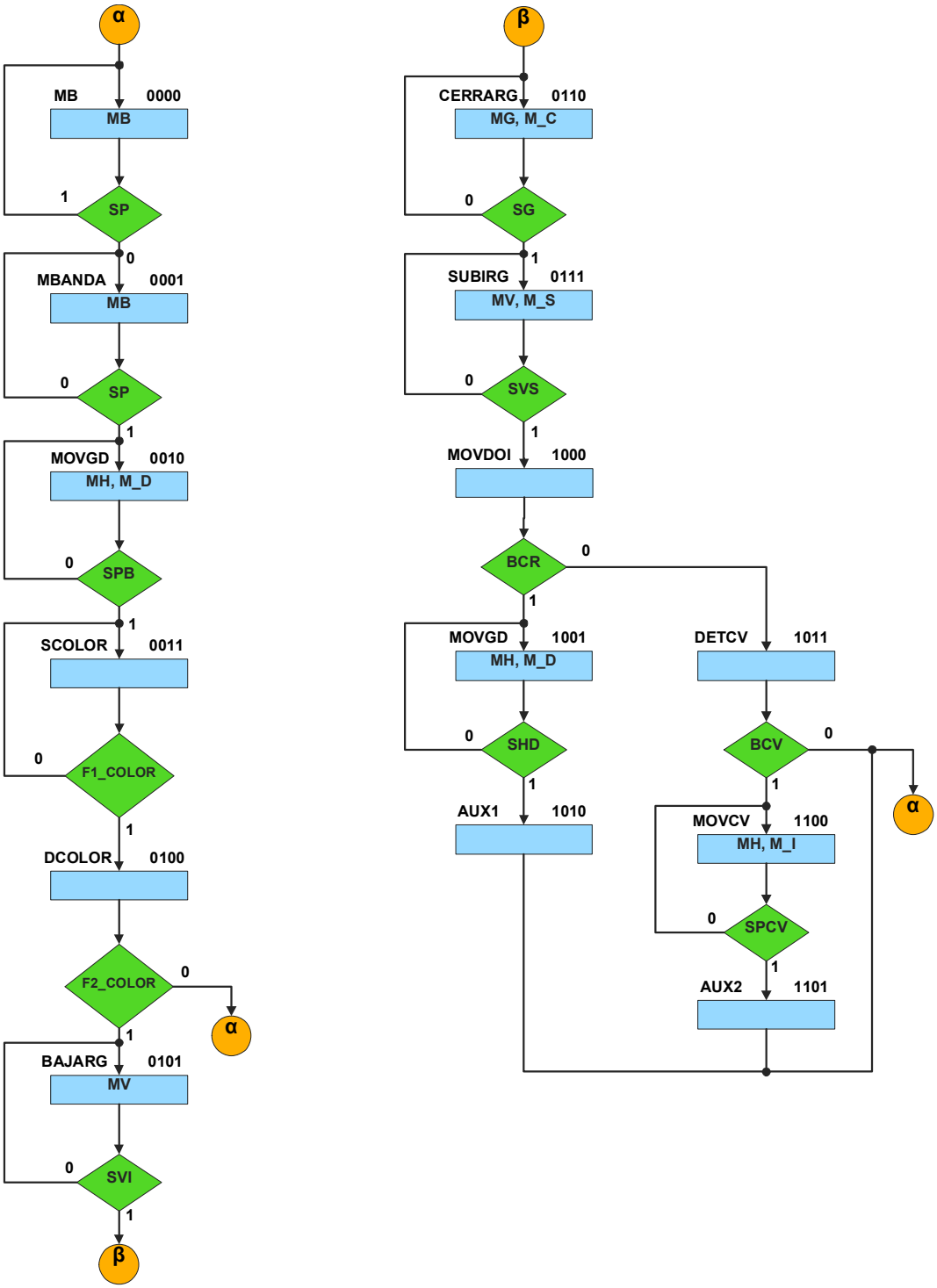


Figura P15.5 Carta ASM del sistema clasificador de paquetes.

Solución

La explicación de la máquina de estados para guardar un registro si se detectó una caja de color rojo o verde se encuentra en la Práctica 14.

Para el sistema clasificador se debe asignar una representación binaria a cada variable de entrada (ver tabla P15.3).

Donde:

```
F1_COLOR = SC_R | SC_V | SC_A | SC_O;

F2_COLOR = SC_R | SC_V | SC_A ;
```

Tabla P15.3 Representación binaria de entradas para el sistema clasificador.

Entrada	Prueba
AUX	0 0 0 0
SVS	0 0 0 1
SVI	0 0 1 0
SHD	0 0 1 1
SPCV	0 1 0 0
SG	0 1 0 1
SP	0 1 1 0
SPB	0 1 1 1
F1_COLOR	1 0 0 0
F2_COLOR	1 0 0 1
BCR	1 0 1 0
BCV	1 0 1 1

Se debe llenar la tabla P15.4 con base en la información de la carta ASM de la figura P15.5, usando el método de diseño con memoria y direccionamiento implícito.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘0000’.

En el Estado ‘0000’ se selecciona la entrada SP, por lo tanto, se coloca en el campo de prueba su representación binaria, es decir, ‘0110’.

Si SP es igual a uno, entonces el estado siguiente es el Estado ‘0000’, su representación binaria ‘0000’ es colocada en el campo de la liga, ya que se requiere hacer una carga.

El campo VF es igual uno, ya que, para hacer una carga en el contador, el valor de la entrada y de VF deben ser iguales. En el Estado ‘0000’ la señal MB está activada, por lo que se coloca un ‘1’ en la parte de salidas de MB.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P15.2). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P15.4 Contenido de la memoria para el sistema clasificador.

Dirección de memoria				Contenido de memoria																			
Estado presente				Prueba				VF	Liga			Liga	Salidas							Salidas	Hex 1	Hex 2	Hex 3
QA	QB	QC	QD	I3	I2	I1	I0		L3	L2	L1	L0	M_C	MG	M_D	MH	M_S	MV	MB	MI			
0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	68	01	00
0	0	0	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	60	81	00
0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	1	1	0	0	0	0	71	18	00
0	0	1	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	81	80	00
0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	90	00	00
0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	22	82	00
0	1	1	0	0	1	0	1	0	0	1	1	0	1	1	0	0	0	0	0	0	53	60	00
0	1	1	1	0	0	0	1	0	0	1	1	1	0	0	0	0	1	1	0	0	13	86	00
1	0	0	0	1	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0	A5	80	00
1	0	0	1	0	0	1	1	0	1	0	0	1	0	0	1	1	0	0	0	0	34	98	00
1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	08	00	00
1	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	B0	00	00
1	1	0	0	0	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	46	08	01
1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	08	00	00

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de dieciséis líneas a una, éste es implementado por el PIC16F1939.

El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P15.5). Se puede comprobar realizando el circuito de la figura P15.6.

Tabla P15.5 Tabla de funcionamiento del selector de entradas para el sistema clasificador.

Prueba				SI
I3	I2	I1	I0	
0	0	0	0	AUX
0	0	0	1	SVS
0	0	1	0	SVI
0	0	1	1	SHD
0	1	0	0	SPCV
0	1	0	1	SG
0	1	1	0	SP
0	1	1	1	SPB
1	0	0	0	F1_COLOR
1	0	0	1	F2_COLOR
1	0	1	0	BCR
1	0	1	1	BCV

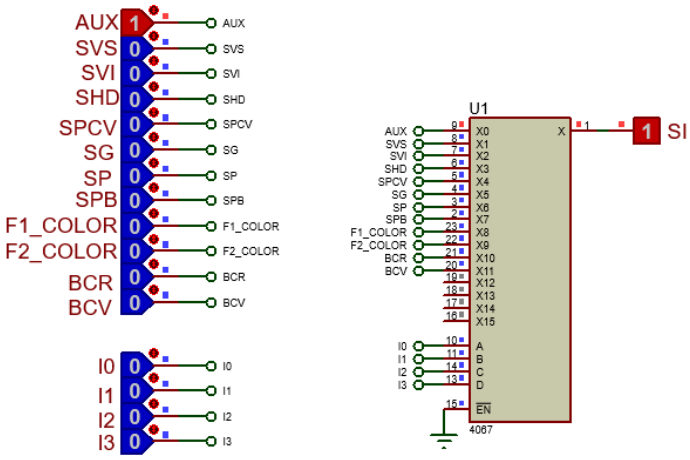


Figura P15.6 Multiplexor 4067 para el selector de entradas del sistema clasificador.

Teniendo en cuenta que `BCR = J` y `BCV = K`, donde J y K son los bits de las máquinas de estados para guardar un registro si se detectó una caja roja o verde, la función booleana del selector de entradas queda:

```
SI = AUX&!I3&!I2&!I1&!I0 | SVS&!I3&!I2&!I1&I0 | SVI&!I3&!I2&I1&!I0 |
     SHD&!I3&!I2&I1&I0 | SPCV&!I3&I2&!I1&!I0 | SG&!I3&I2&!I1&I0 |
     SP&!I3&I2&I1&!I0 | SPB&!I3&I2&I1&I0 | F1_COLOR&I3&!I2&!I1&!I0 |
     F2_COLOR&I3&!I2&!I1&I0 | J&I3&!I2&I1&!I0 | K&I3&!I2&I1&I0;
```

Para obtener el valor de la lógica, se debe hacer la operación XOR entre el selector de entradas y el valor de VF (ver figura P15.4).

Por lo tanto, la función booleana de la lógica queda:

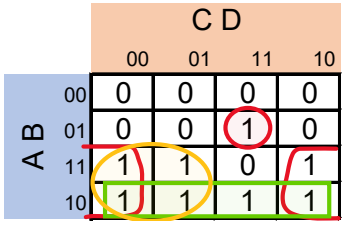
```
SL = VF ^ (AUX&!I3&!I2&!I1&!I0 | SVS&!I3&!I2&!I1&I0 | SVI&!I3&!I2&I1&!I0
     | SHD&!I3&!I2&I1&I0 | SPCV&!I3&I2&!I1&!I0 | SG&!I3&I2&!I1&I0 |
     SP&!I3&I2&I1&!I0 | SPB&!I3&I2&I1&I0 | F1_COLOR&I3&!I2&!I1&!I0 |
     F2_COLOR&I3&!I2&!I1&I0 | J&I3&!I2&I1&!I0 | K&I3&!I2&I1&I0);
```

Se utiliza un contador con carga paralela que indica que estado es el siguiente. El contador con carga paralela es implementado por el PIC16F1939. Si el valor a la salida de la lógica es igual a ‘1’, el contador carga la información binaria de la liga a la memoria. Si el valor de la lógica es igual a ‘0’, se cuenta al siguiente estado binario. A continuación, se obtienen las expresiones lógicas para un contador de cuatro bits por el método de variable suscrita (ver tabla P15.6).

Tabla P15.6 Mapa de Karnaugh general de transición de estados para un contador de cuatro bits.

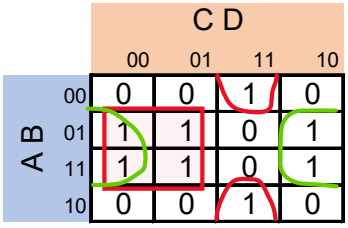
		C D			
		0 0	0 1	1 1	1 0
A B	0 0	0000	0001	0011	0010
		0001	0010	0100	0011
	0 1	0100	0101	0111	0110
		0101	0110	1000	0111
	1 1	1100	1101	1111	1110
		1101	1110	0000	1111
	1 0	1000	1001	1011	1010
		1001	1010	1100	1011

Para el bit A:



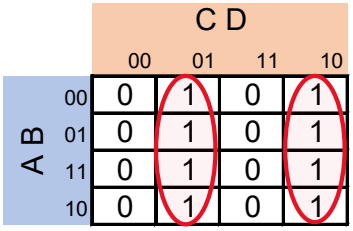

```
FFA = A&!C | A&!B | A&!D |
!A&B&C&D;
```

Para el bit B:



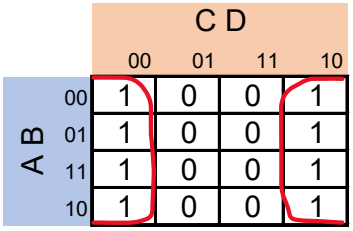

```
FFB = B&!C | B&!D | !B&C&D;
```

Para el bit C:




```
FFC = !C&D | C&!D = C^D;
```

Para el bit D:




```
FFD = !D;
```

La explicación para obtener las expresiones que controlan el motor paso a paso se encuentra en la Práctica 12.

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P15.7, figura P15.8). Se carga en el controlador el archivo con extensión “HEX” de la memoria y los archivos “COF” o “HEX” del PIC16F1939. Para poder visualizar de manera más rápida el movimiento del motor paso a paso, se debe utilizar una frecuencia de 30 Hz y el ángulo en el que gira el motor paso a paso debe ser de 3.6°. Se modificó la biblioteca PLD.h para tener más puertos de entradas en el PIC.

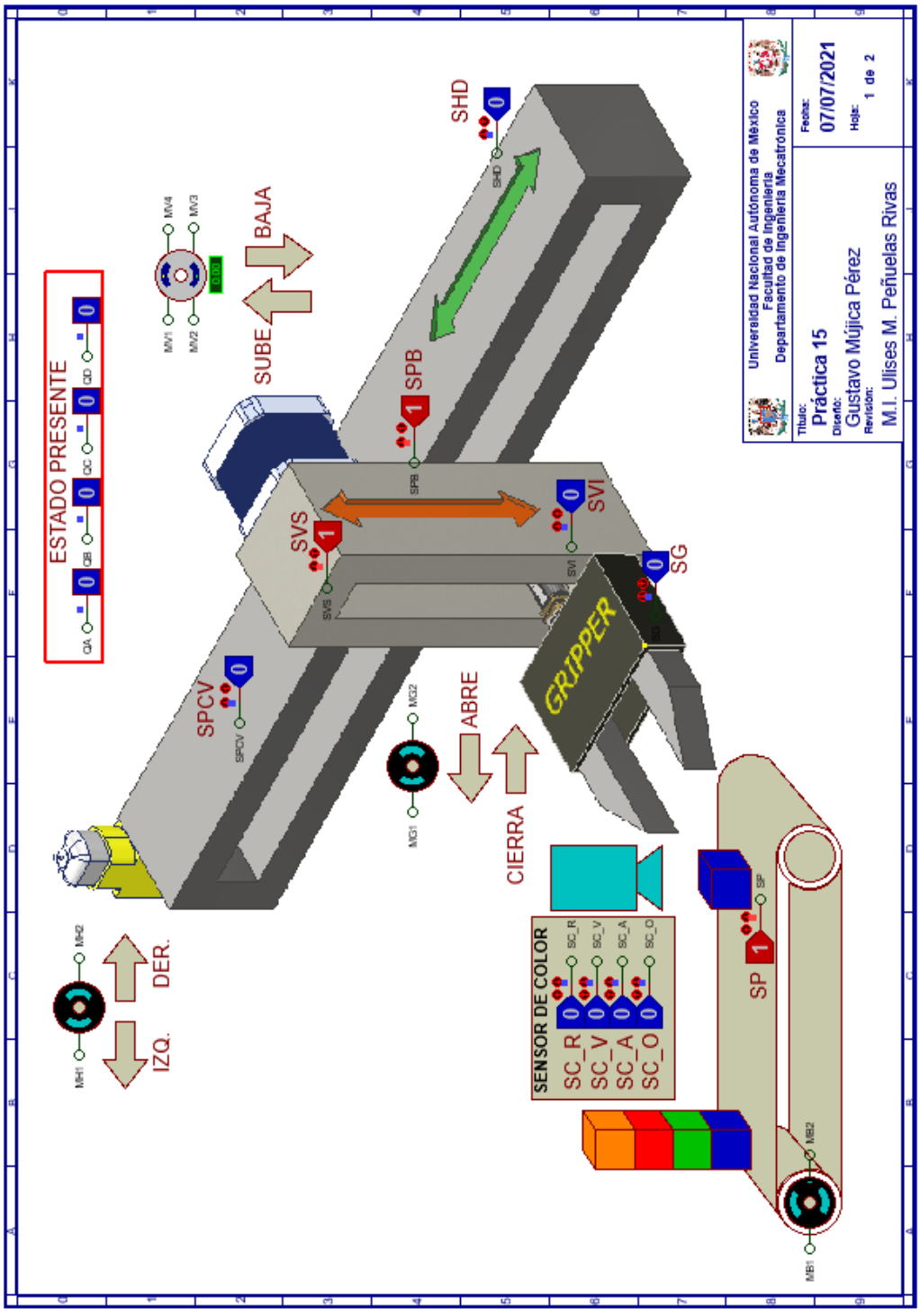


Figura P15.7 Interfaz hombre-máquina para el controlador de la Práctica 15 hoja 1/2.

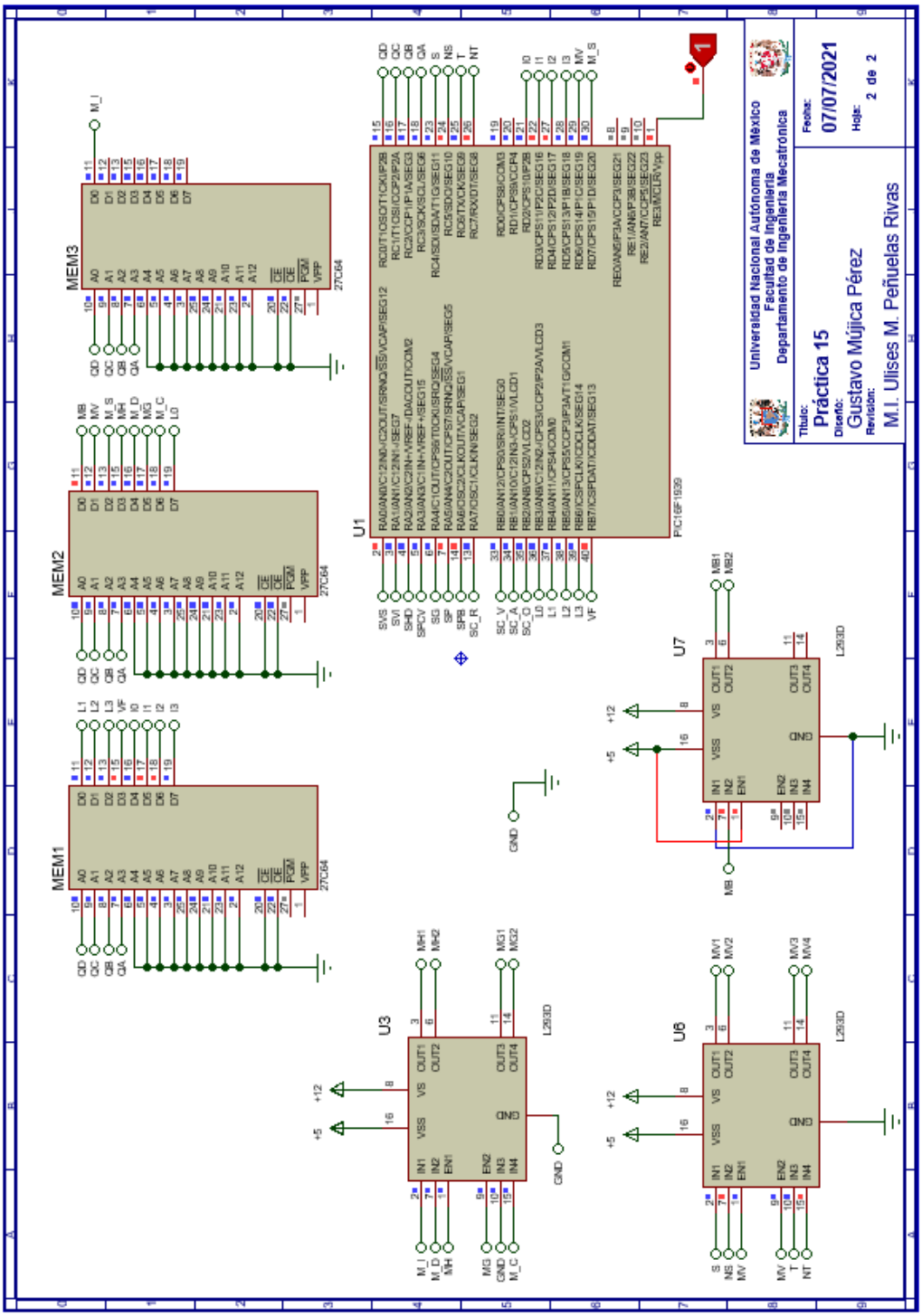


Figura P15.8 Esquema electrónico para el controlador de la Práctica 15 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P15.10 a figura P15.11) y se obtendrán archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD_SROBOTIZADO.h> // Carga biblioteca PLD_SROBOTIZADO.h

//***CONTADOR***

//#define CLK E0 //RELOJ
//LIGA
#define L0 B3
#define L1 B4
#define L2 B5
#define L3 B6

//SALIDAS
#define A C3 //FFA
#define B C2 //FFB
#define C C1 //FFC
#define D C0 //FFD

//***LOGICA***

//ENTRADAS
#define SVS   A0
#define SVI   A1
#define SHD   A2
#define SPCV  A3
#define SG    A4
#define SP    A5
#define SPB   A6
#define SC_R  A7

#define SC_V  B0
#define SC_A  B1
#define SC_O  B2
//VF
#define VF B7
//PRUEBAS
#define I0 D2
#define I1 D3
#define I2 D4
#define I3 D5

//****MOTOR A PASOS****
//ENTRADAS
#define MVn  D6 //MV
#define M Sn D7 //M S
```

Figura P15.9 Código de la Práctica 15 parte 1.

```
45: #define M_Sn D7 //M_S
46: //SALIDAS
47: #define S  C4
48: #define NS C5
49: #define T  C6
50: #define NT C7
51:
52: //***VARIABLES INTERMEDIAS***
53:
54: short SL=0;  //Salida lógica
55: short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
56: short Sn, Tn; // Variables intermedias motor a pasos
57: short At=0, Bt=0, Ct=0, Dt=0; //Variables intermedias de los FF
58: short J,K,Jn,Kn; // Variable auxiliar para registrar si se detectó una caja color verde o roja
59: short LD3,LD2,LD1,LD0; //Variables intermedias de la liga
60: short F1_COLOR, F2_COLOR; //Funciones
61:
62: void main ()
63: {
64: pld_ini();  // INICIALIZA AL PIC COMO PLD
65: pld_555(30);   // Genera señal cuadrada en Hz,
66:                  //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
67:
68: //LOOP INFINITO
69:    while(1)
70:    {
71:    //CIRCUITO COMBINACIONAL
72:    LD3=L3; LD2=L2; LD1=L1; LD0=L0;     /*ALMACENA DATOS HASTA EL CAMBIO DEL RELOJ,
73:                                         SIMULANDO UN REGISTRO, EVITANDO ASÍ
74:                                         QUE SE MODIFIQUEN LOS VALORES DE LA MEMORIA*/
75:
76:    //FUNCIONES
77:    F1_COLOR= SC_R | SC_V | SC_A | SC_O;
78:    F2_COLOR= SC_R | SC_V | SC_A;
79:
80:    //SALIDA LÓGICA
81:     SL = VF ^ (AUX&!I3&!I2&!I1&!I0 | SVS&!I3&!I2&!I1&I0 | SVI&!I3&!I2&I1&!I0 | SHD&!I3&!I2&I1&I0 |
82:          SPCV&!I3&I2&!I1&!I0 | SG&!I3&I2&!I1&I0 | SP&!I3&I2&I1&!I0 | SPB&!I3&I2&I1&I0 |
83:          F1_COLOR&I3&!I2&!I1&!I0 | F2_COLOR&I3&!I2&!I1&I0 | J&I3&!I2&I1&!I0 | K&I3&!I2&I1&I0);
84:
85:        //****CIRCUITO SECUENCIAL ****
86:  //  if (!CLK) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO
87:    if (!out_555) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO,
88:                  //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
89:
```

Figura P15.10 Código de la Práctica 15 parte 2.

```
 89:
 90:     {//SECCIÓN DE OPERACIONES DEL CONTADOR CON CARGA PARALELA
 91:     At= A&!C | A&!B | A&!D | !A&B&C&D;
 92:     Bt= B&!C | B&!D | !B&C&D;
 93:     Ct= C^D;
 94:     Dt= !D;
 95:
 96:     //MOTOR A PASOS
 97:     Sn= (!T&!M_Sn | T&M_Sn)&MVn;
 98:     Tn= (!S&M_Sn | S&!M_Sn)&MVn;
 99:     //REGISTRO CAJA COLOR ROJO O VERDE
100:     Jn= !J&SC_R | J&!SHD;
101:     Kn= !K&SC_V | K&!SPCV;
102:     }
103:
104:     else
105:     {
106:     //SECCIÓN DE MEMORIZACIÓN
107:     //CUENTA CON SL=1 Y CARGA CON SL=0
108:     A= SL&At | !SL&LD3;
109:     B= SL&Bt | !SL&LD2;
110:     C= SL&Ct | !SL&LD1;
111:     D= SL&Dt | !SL&LD0;
112:
113:     S=Sn;
114:     T=Tn;
115:     NS=!S;
116:     NT=!T;
117:
118:     J=Jn;
119:     K=Kn;
120:   /*  while(clk){}/*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
121:                 POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
122:                 DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
123:                 LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ*/
124:
125:     while(out_555){} /*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
126:                     POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
127:                     DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
128:                     LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ,
129:                     EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO*/
130:
131:     }
132:     }
133: }
```

Figura P15.11 Código de la Práctica 15 parte 3.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 16 Sistema robotizado para clasificar paquetes por colores; diseño con memoria y direccionamiento implícito

Introducción

A través de una maqueta, se simuló el funcionamiento de un sistema robotizado que clasifica paquetes o cajas por colores y debe ser capaz de reconocer tres tipos de paquetes por su color colocados en una banda transportadora.

Los elementos principales del sistema son: un robot, un sensor de color y una banda transportadora. Inicialmente el sistema está en reposo, para activar el sistema se debe presionar un botón, al presionarlo la banda transportadora empezará a desplazar un paquete, un sensor detectará cuando el paquete esté en la posición adecuada para evaluar su color, por lo que la banda se detendrá.

El robot sujeta los paquetes y los deposita en el contenedor correspondiente. Los colores serán rojo, azul y verde. Los paquetes que no sean de estos colores, los dejará pasar (ver figura P16.1).

El robot se mueve a lo largo del eje horizontal por medio de un motor DC, para desplazarse en el eje vertical lo hace por medio de un motor paso a paso. El robot cuenta con un gripper, el cual es el encargado de sujetar los paquetes. El gripper está conformado por un motor DC.

El sistema tiene un botón de emergencia, cuando éste sea oprimido el sistema quedará inmóvil, hasta que se oprima otro botón para dirigirse a su posición inicial. Se consideró la posición inicial del sistema cuando la articulación horizontal está en su extremo izquierdo, la articulación vertical en su posición superior y con el gripper abierto.

El sistema se dividió en 2 partes para facilitar su comprensión. El sistema de posición inicial controla cuando el gripper debe moverse en su posición inicial y el sistema clasificador, controla la banda y la clasificación de los paquetes por colores. La unión de los dos sistemas mencionados anteriormente forma el sistema robotizado clasificador de paquetes.

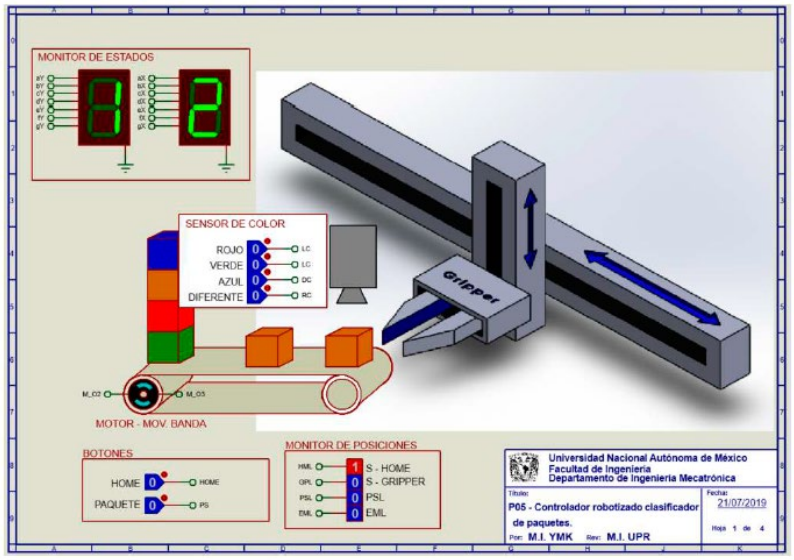


Figura P16.1 Sistema robotizado para clasificar paquetes por colores.

El diseño con memoria y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura P16.2). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura P16.3 muestra la arquitectura de este método.

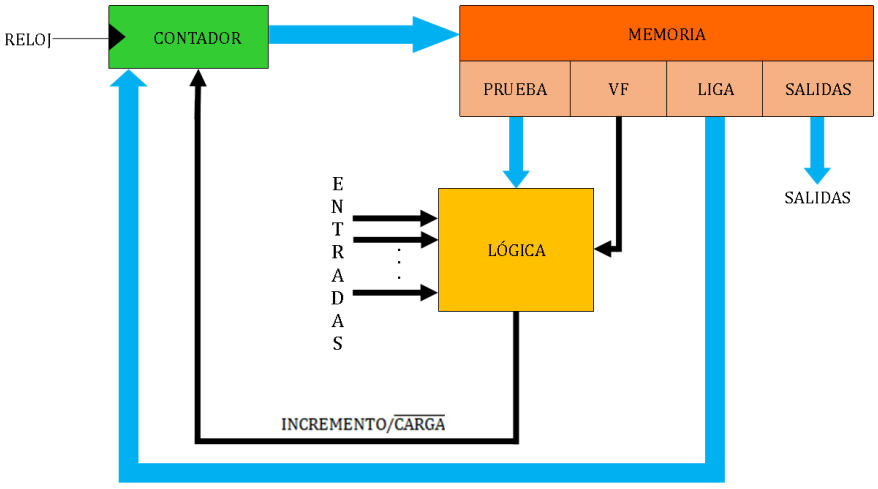


Figura P16.2 Arquitectura de un controlador con memoria y direccionamiento implícito.

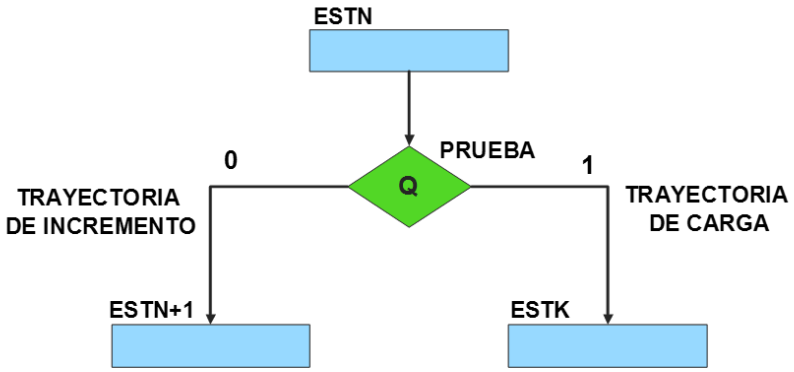


Figura P16.3 Trayectoria de incremento y carga.

La tabla P16.1 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que valor de entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un '1' se hace un incremento, cuando el resultado de la función da como resultado un '0' se hace una carga.

Tabla P16.1 Relación entre VF, variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incrementa o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es '0', se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es '1', se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura P16.4).

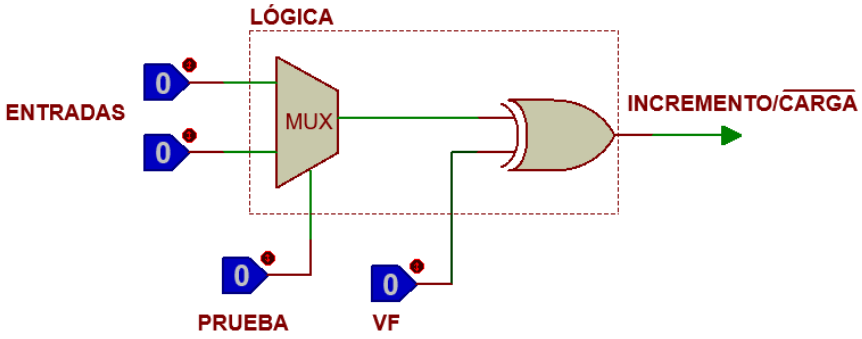


Figura P16.4 Bloque de lógica de incremento/carga para el direccionamiento implícito [1].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

Objetivo

Diseñar un controlador para el sistema robotizado para clasificar paquetes por colores por medio del método de diseño con memoria y direccionamiento implícito.

Descripción

Primero se diseña una carta ASM para el sistema robotizado para clasificar paquetes por colores por medio del método de diseño con memoria y direccionamiento implícito. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P16.2 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema robotizado clasificador se necesitan señales de entrada:

- un botón detecta la solicitud para que el sistema comience a funcionar o para regresar a la posición inicial de éste
- se necesitan sensores para detectar la posición inferior y superior del sistema
- se necesitan sensores para detectar el final de carrera derecho e izquierdo del sistema
- un sensor indica la posición donde se debe colocar la caja verde
- un sensor verifica si se cerró el gripper para tomar una caja
- un sensor detecta una caja en la banda
- un sensor le indica al sistema donde se encuentra la banda
- un sensor detecta el color de la caja.

Como salida se requiere de las siguientes señales:

- activación del motor de la banda
- activación del motor para el movimiento vertical
- una señal se activa para mover el gripper a la posición superior del sistema
- activación del motor para el movimiento horizontal
- una señal se activa para mover el gripper hacia el extremo derecho del sistema y otra para el extremo izquierdo
- activación del motor del gripper
- una señal se activa para que el gripper se cierre y otra para que se abra.

Tabla P16.2 Entradas y salidas para el sistema robotizado clasificador de paquetes.

Identificador	Ubicación	Tipo	Descripción
BH	A0 del PIC	I	Botón de Home. Botón para ir a la posición inicial del sistema
SVS	A1 del PIC	I	Sensor de posición superior
SVI	A2 del PIC	I	Sensor de posición inferior
SHI	A3 del PIC	I	Sensor de final de carrera izquierda
SHD	A4 del PIC	I	Sensor de final de carrera derecha
SPCV	A5 del PIC	I	Sensor que indica la posición donde se debe colocar la caja verde
SG	A6 del PIC	I	Sensor de gripper que indica cuándo el gripper está sujetando una caja
SP	A7 del PIC	I	Sensor de caja que la detecta y detiene la banda
SPB	B0 del PIC	I	Sensor de posición banda que le indica al sistema dónde está la banda, para detenerse y dejar de desplazarse hacia la derecha
SC-R	B1 del PIC	I	Señal del sensor de color para detectar una caja de color rojo
SC-V	B2 del PIC	I	Señal del sensor de color para detectar una caja de color verde
SC-A	B3 del PIC	I	Señal del sensor de color para detectar una caja de color azul
SC-O	B4 del PIC	I	Señal del sensor de color para detectar una caja de otro color
BCR	Interna	I	Señal indicadora que se detectó una caja color roja
BCV	Interna	I	Señal indicadora que se detectó una caja color verde
M_C	D0 de la memoria 3	O	Señal para cerrar el gripper
M_A	D1 de la memoria 3	O	Señal para abrir el gripper
MG	D2 de la memoria 3	O	Motorreductor del gripper
M_S	D0 de la memoria 2	O	Señal para hacer que el gripper suba
MV	D1 de la memoria 2	O	Motor vertical paso a paso del movimiento vertical
M_D	D2 de la memoria 2	O	Señal para hacer que el gripper se mueva a la derecha
M_I	D3 de la memoria 2	O	Señal para hacer que el gripper se mueva a la izquierda
MH	D4 de la memoria 2	O	Motor horizontal de CD para del movimiento horizontal
MB	D5 de la memoria 2	O	Motor de la banda.

Notas de diseño

a) Para iniciar a mover la banda si se acaba de encender el sistema, se debe oprimir el botón de home
b) Para que funcione de nuevo el sistema si se oprimió el botón de paro, se debe oprimir el botón de home
c) Se considera la posición inicial del sistema cuando el gripper está abierto, en la posición superior y en el extremo izquierdo del sistema
d) Si se requiere ir a la posición inicial en cualquier momento, se debe oprimir el botón de paro primero y después el botón home
e) Se considera que el sensor de color sólo podrá detectar colores cuando el gripper esté en la posición superior del sistema, antes de recoger la caja
f) Las cajas rojas serán llevadas al extremo derecho del sistema
g) Las cajas verdes serán llevadas a una posición intermedia del sistema, antes de llegar al extremo izquierdo del mismo
h) Las cajas azules serán llevadas al extremo izquierdo del sistema
i) El gripper sólo podrá llevar una caja al extremo izquierdo o derecho del sistema, sólo cuando éste se encuentre en la posición superior del sistema.

Reglas de funcionamiento

- BH: botón de home
  1 = se oprimió el botón de home
  0 = no se oprimió el botón de home
- SVS: sensor de posición superior
  1 = el gripper está en la posición superior
  0 = no está el gripper en la posición superior
- SVI: sensor de posición inferior
  1 = el gripper está en la posición inferior del sistema
  0 = no está el gripper en la posición inferior del sistema

- SHI: sensor de final de carrera izquierda
  1 = el gripper está en el extremo izquierdo del sistema
  0 = no está el gripper en el extremo izquierdo del sistema
- SHD: sensor de final de carrera derecha
  1 = el gripper está en el extremo derecho del sistema
  0 = no está el gripper en el extremo derecho del sistema
- SPCV: sensor posición caja verde
  0 = el gripper está en la posición para soltar la caja verde
  0 = no está el gripper en la posición para soltar la caja verde
- SG: sensor de gripper
  1 = el gripper está sujetando una caja o cerrado
  0 = no está el gripper sujetando una caja o abierto
- SP: sensor de caja
  1 = se detecta una caja en la banda
  0 = no se detecta una caja en la banda
- SPB: sensor de posición banda
  1 = se detecta que el gripper está alineado con la banda
  0 = no se detecta que el gripper está alineado con la banda
- SC-R: caja de color rojo
  1 = se detecta una caja de color rojo
  0 = no se detecta una caja de color rojo
- SC-V: caja de color verde
  1 = se detecta una caja de color verde
  0 = no se detecta una caja de color verde
- SC-A: caja de color azul
  1 = se detecta una caja de color azul
  0 = no se detecta una caja de color azul
- SC-O: caja de otro color
  1 = se detecta una caja de otro color
  0 = no se detecta una caja de otro color

- BCR: registro de caja color rojo
  1 = se detectó una caja color rojo
  0 = no se detectó una caja color rojo
- BCV: registro de caja color verde
  1 = se detectó una caja color verde
  0 = no se detectó una caja color verde

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Se hace un incremento cuando el valor del estado siguiente aumenta en una unidad, de lo contrario se hace una carga (ver figura P16.3). El algoritmo de la máquina de estados se puede ver en la figura P16.5 y en la figura P16.6.

Estado ‘00000’ – INICIO

El sistema se dirige a estado, después de que se deja de presionar el botón de paro o cuando se acaba de encender. Cuando el botón (BH) es oprimido, el sistema avanza al Estado ‘00001’ para verificar la posición vertical del gripper. De lo contrario, permanece en el Estado ‘00000’, en espera de que sea oprimido.

Estado ‘00001’ – DETPV

En estado se verifica la posición vertical del sistema. Si el sensor (SVS), detecta al gripper en la posición vertical superior, el sistema avanza al Estado ‘00011’ para detectar si el gripper está en alguno de los extremos horizontales del sistema. De lo contrario, avanza al Estado ‘00010’ para subir el gripper.

Estado ‘00010’ – SUBIRG

En este estado se activa el motor paso a paso (MV) y la señal (M_S) para que el motor paso a paso suba el gripper a la posición superior del sistema. Cuando el sensor (SVS), detecte que el gripper ha llegado a la posición superior, el sistema avanza al Estado ‘00011’ para detectar si el gripper está en alguno de los extremos horizontales del sistema. De lo contrario, permanece en el Estado ‘00010’ subiendo el gripper.

Estado ‘00011’ – DETPH

En estado se verifica si el gripper está en alguno de los extremos horizontales del sistema. Si el sensor (SHI), detecta que el gripper se encuentra en el final de carrera izquierda, el sistema avanza al Estado ‘00101’ para detectar si el gripper está abierto o cerrado. De lo contrario, avanza al Estado ‘00100’, para mover el gripper al extremo izquierdo del sistema.

Estado ‘00100’ – MOVGI

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_I) para mover el gripper al extremo izquierdo del sistema. Cuando el sensor (SHI), detecta que el gripper ha llegado al límite izquierdo del sistema, el sistema avanza al Estado ‘00101’ para detectar si el gripper está abierto o cerrado. De lo contrario, permanece en el Estado ‘00100’ desplazando el gripper hacia la izquierda.

Estado ‘00101’ – DETG

En estado se verifica si el gripper está abierto o cerrado. Si el sensor (SG) detecta que el gripper está abierto, el sistema avanza al Estado ‘01001’ para activar el motor de la banda y así empezar o continuar clasificando cajas. De lo contrario, el sistema avanza al Estado ‘00110’ para bajar el gripper a la posición inferior y soltar la caja que está sujetando.

Estado ‘00110’ – BAJARG

En este estado se activa el motor paso a paso (MV) y debido a que no se activa la señal (M_S) para subir el gripper, el gripper desciende a la posición inferior del sistema.

Cuando el sensor (SVI), detecta que el gripper ha llegado a la posición inferior, el sistema avanza al Estado '00111' para abrir el gripper y por lo tanto soltar la caja. De lo contrario, permanece en el Estado '00110' bajando el gripper.

Estado '00111' – ABRIRG

En este estado se activa el motor del gripper (MG) y la señal (M_A) para que el gripper se abra y suelte la caja. Cuando el sensor del gripper (SG), detecte que se ha abierto el gripper completamente, el sistema avanza al Estado '01000' para posteriormente desplazar el gripper a su posición inicial. De lo contrario, permanece en el Estado '00111' abriendo el gripper.

Estado '01000' – AUX1

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado '00010', haciendo una carga para poder subir el gripper. Posteriormente se regresará el gripper a su posición inicial y empezará el proceso nuevamente para clasificar otra caja.

Estado '01001' – MB

En este estado se activa el motor de la banda (M_B). Si en este estado, el sensor (SP) no detecta una caja, el sistema avanza al Estado '01010' para continuar moviendo la banda hasta que se detecte una. De lo contrario, permanece en el Estado '01001' moviendo la banda, para que la caja no se tome en cuenta en la clasificación ya que es de un color distinto a los establecidos.

Estado '01010' – MBANDA

En este estado se activa el motor de la banda (M_B). Si el sensor (SP), detecta una caja avanza al Estado '01011' para detener la banda e iniciar el proceso de clasificación. De lo contrario, permanece en el Estado '01010' moviendo la banda.

Estado '01011' – MOVGD

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_D) para mover el gripper a la derecha.

Cuando el sensor de posición de la banda (SPB), detecta que el sistema está alineado con la banda, avanza al Estado '01100' para detener el desplazamiento del gripper y detectar el color de la caja. De lo contrario, permanece en el Estado '01011' moviendo el gripper a la derecha.

Estado '01100' – SCOLOR

En este estado el gripper está en espera de que el sensor detecte el color de la caja. Cuando el sensor detecta el color de la caja, el sistema avanza al Estado '01101' para mover el gripper dependiendo del color. De lo contrario, permanece en el Estado '01100' en espera a que el sensor reconozca el color de la caja.

Estado '01101' – DCOLOR

En este estado se determina a que posición se lleva la caja dependiendo de su color. Si el sensor de color detecta una caja roja, verde o azul el sistema avanza al Estado '01110' para recogerla (se guardará un registro si la caja es de color rojo o verde). De lo contrario, regresa al Estado '00100' para volver a la posición inicial y dejar pasar la caja ya que es de un color distinto a los establecidos.

Estado '01110' – BAJARG

En este estado se activa el motor paso a paso (MV) y debido a que no se activa la señal (M_S) el motor paso a paso baja el gripper a la posición inferior del sistema. Cuando el sensor (SVI), detecta que el gripper ha llegado a la posición inferior del sistema, el sistema avanza al Estado '01111' para cerrar el gripper y por lo tanto sujetar la caja. De lo contrario, permanece en el Estado '01110' bajando el gripper.

Estado '01111' – CERRARG

En este estado se activa el motor del gripper (MG) y la señal (M_C) para que el gripper se cierre. Cuando el sensor del gripper (SG), detecta que se ha cerrado el gripper completamente, el sistema avanza al Estado '10000' para desplazar el gripper a la posición superior del sistema. De lo contrario, permanece en el Estado '01111' cerrando el gripper.

Estado ‘10000’ – SUBIRG

En este estado se activa el motor paso a paso (MV) y la señal (M_S) para que el motor paso a paso suba el gripper a la posición superior del sistema. Si el sensor (SVS), detecta que el gripper ha llegado a la posición superior, el sistema avanza al Estado ‘10001’ para determinar si se debe mover el gripper a la derecha o a la izquierda. De lo contrario, permanece en el Estado ‘10000’ subiendo el gripper.

Estado ‘10001’ – MOVDOI

En este estado se determina a que posición mover el gripper. Si el sensor de color detectó una caja de color rojo, avanza al Estado ‘10010’ para mover el gripper a la derecha. De lo contrario, el sistema avanza al Estado ‘10100’ para verificar si se detectó una caja color verde.

Estado ‘10010’ – MOVGD

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_D) para mover el gripper al extremo derecho del sistema. Si el sensor (SHD), detecta que el gripper ha llegado al límite derecho del sistema, avanza al Estado ‘10011’ para que posteriormente se pueda hacer una carga al Estado ‘00110’ y así poder bajar el gripper. De lo contrario, permanece en el Estado ‘10010’ desplazando el gripper hacia la derecha.

Estado ‘10011’ – AUX2

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘00110’, haciendo una carga para bajar el gripper. Posteriormente se regresará el gripper a su posición inicial y empezará el proceso nuevamente para clasificar otra caja.

Estado ‘10100’ – DETCV

En este estado si el sensor de color detectó una caja color verde, el sistema avanza al Estado ‘10101’ para mover el gripper hacia una posición intermedia antes de llegar al extremo izquierdo.

De lo contrario, la caja es color azul, el sistema regresa al Estado ‘00100’ para llevarla al extremo izquierdo del sistema y posteriormente empezar nuevamente el proceso para clasificar otra caja.

Estado ‘10101’ – MOVCV

En este estado se activa el motor del movimiento horizontal (MH) y la señal (M_I) para mover la caja verde a una posición intermedia antes de llegar al extremo izquierdo del sistema. Cuando el sensor (SPCV), detecta que el gripper ha llegado a la posición intermedia, el sistema avanza al Estado ‘10110’ para posteriormente poder avanzar al Estado ‘00110’ y así poder bajar el gripper. De lo contrario, permanece en el Estado ‘10101’ desplazando el gripper hacia la izquierda.

Estado ‘10110’ – AUX3

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘00110’, haciendo una carga para bajar el gripper. Posteriormente se regresará el gripper a su posición inicial y empezará el proceso nuevamente para clasificar otra caja.

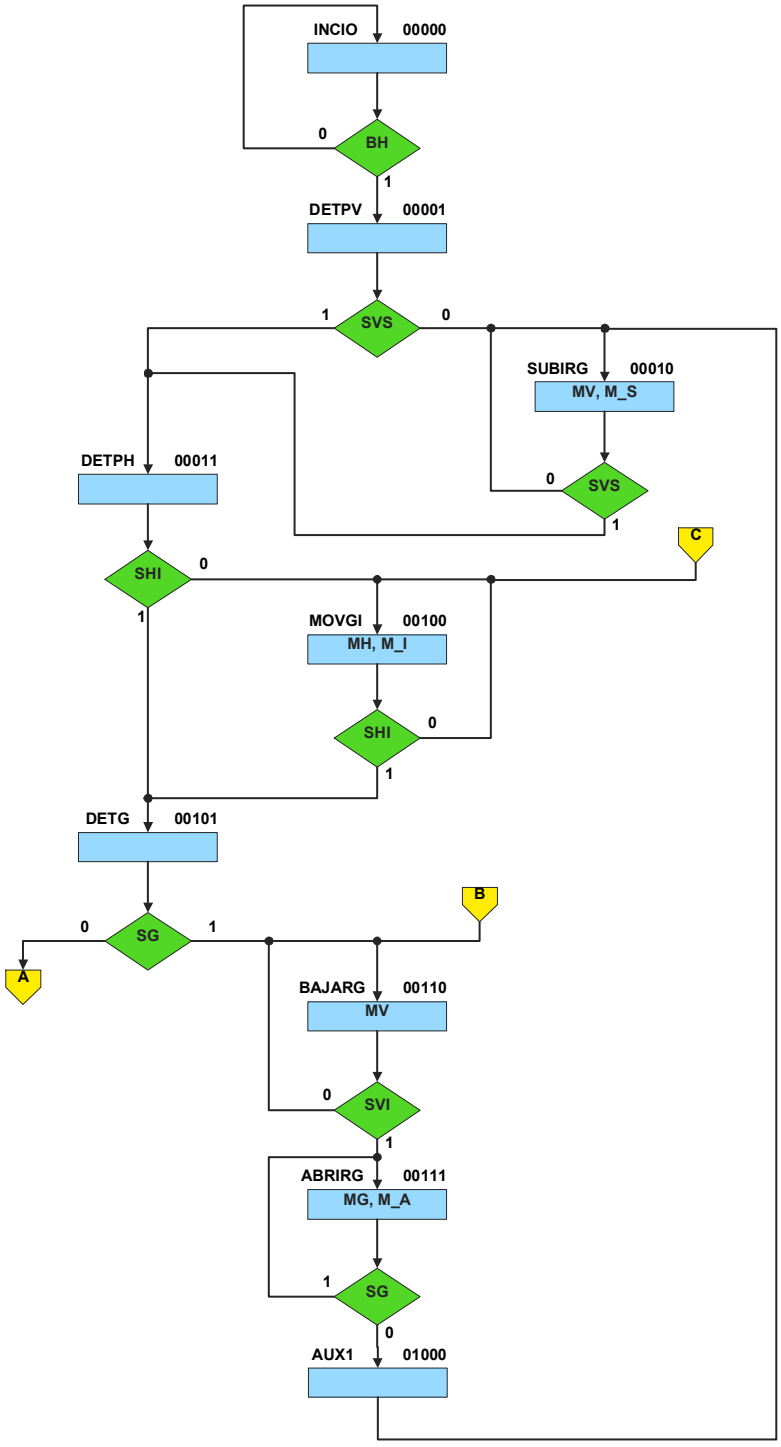


Figura P16.5 Carta ASM del sistema clasificador de paquetes parte 1.

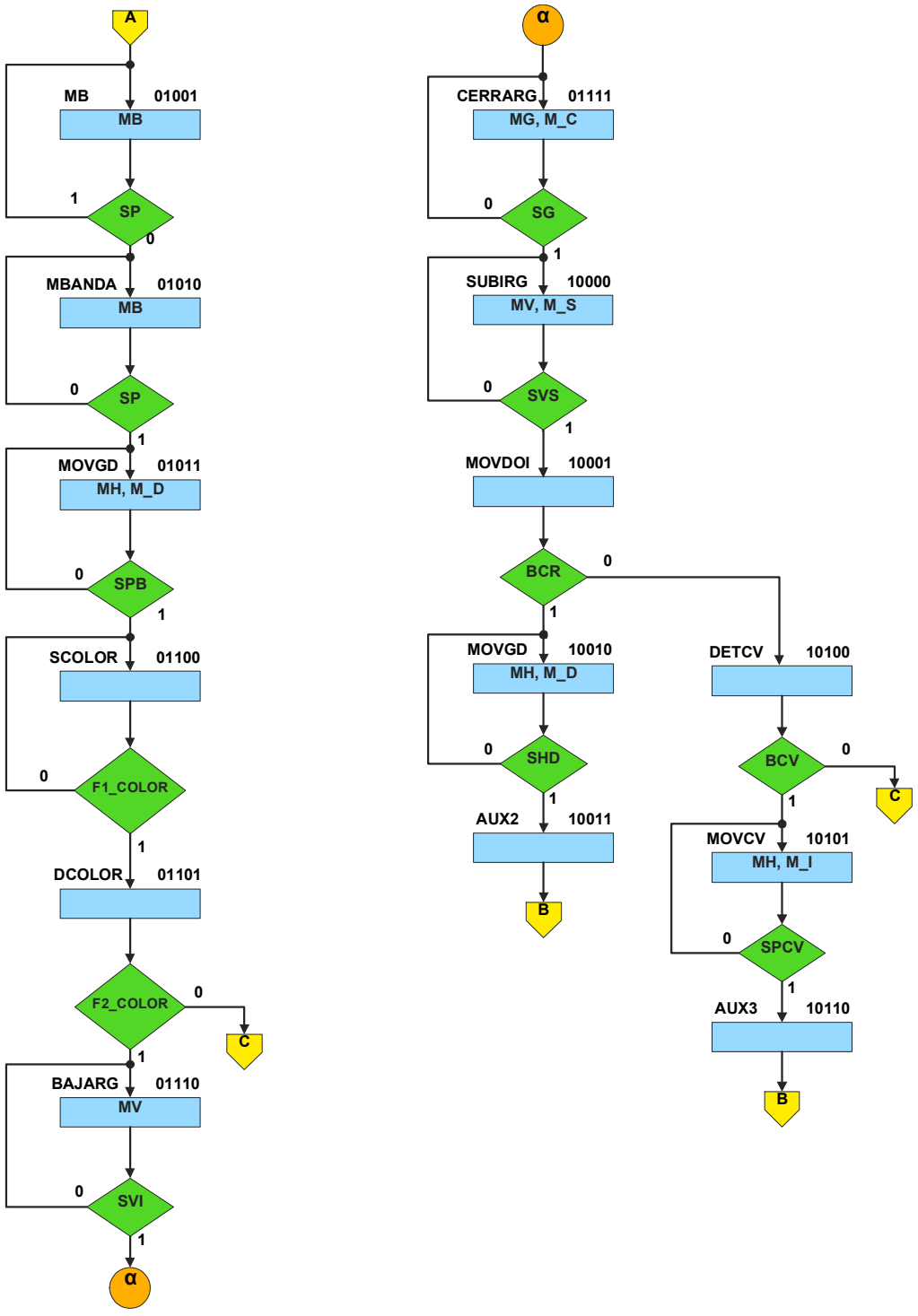


figura P16.6 Carta ASM del sistema clasificador de paquetes parte 2.

Solución

La explicación de la máquina de estados para guardar un registro si se detectó una caja de color rojo o verde se encuentra en la Práctica 14.

Para el sistema robotizado para clasificar paquetes por colores se debe asignar una representación binaria a cada variable de entrada (ver tabla P16.3).

Donde:

```
F1_COLOR = SC_R | SC_V | SC_A | SC_O;

F2_COLOR = SC_R | SC_V | SC_A;
```

Tabla P16.3 Representación binaria de entradas para el sistema robotizado clasificador de paquetes.

Entrada	Prueba
AUX	0 0 0 0
BH	0 0 0 1
SVS	0 0 1 0
SVI	0 0 1 1
SHI	0 1 0 0
SHD	0 1 0 1
SPCV	0 1 1 0
SG	0 1 1 1
SP	1 0 0 0
SPB	1 0 0 1
F1_COLOR	1 0 1 0
F2_COLOR	1 0 1 1
BCR	1 1 0 0
BCV	1 1 0 1

Se debe llenar la tabla P16.4 con base en la información de la carta ASM de la figura P16.5 y la figura P16.6, usando el método de diseño con memoria y direccionamiento implícito.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘00000’.

En el Estado ‘00000’ se selecciona la entrada BH, por lo tanto, se coloca en el campo de prueba su representación binaria, es decir, ‘0001’.

Si BH es igual a cero, entonces el estado siguiente es el Estado ‘00000’, su representación binaria ‘00000’ es colocada en el campo de la liga, ya que se requiere hacer una carga. El campo VF es igual cero, ya que, para hacer una carga en el contador, el valor de la entrada y de VF deben ser iguales. En el Estado ‘00000’ no hay señales de salida activadas, por lo que se coloca un ‘0’ en la parte de las salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria, ver figura P16.2). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P16.4 Contenido de la memoria para el sistema robotizado clasificador de paquetes.

Dirección de memoria					Contenido de memoria																				
Estado presente					Prueba				VF	Liga			Liga		Salidas						Salidas			Hex 1	Hex 2	Hex 3
QA	QB	QC	QD	QE	I3	I2	I1	I0		L4	L3	L2	L1	L0	MB	MH	M_I	M_D	MV	M_S	MG	M_A	M_C			
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	00	00
0	0	0	0	1	0	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	28	C0	00
0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	20	83	00
0	0	0	1	1	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	49	40	00
0	0	1	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	41	18	00
0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	72	40	00
0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	31	82	00
0	0	1	1	1	0	1	1	1	1	0	0	1	1	1	0	0	0	0	0	0	1	1	0	79	C0	06
0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	08	80	00
0	1	0	0	1	1	0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0	8A	60	00
0	1	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	82	A0	00
0	1	0	1	1	1	0	0	1	0	0	1	0	1	1	0	1	0	1	0	0	0	0	0	92	D4	00
0	1	1	0	0	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	A3	00	00
0	1	1	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	B1	00	00
0	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	0	0	0	1	0	0	0	0	33	82	00
0	1	1	1	1	0	1	1	1	0	0	1	1	1	1	0	0	0	0	0	0	1	0	1	73	C0	05
1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	24	03	00
1	0	0	0	1	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	C5	00	00
1	0	0	1	0	0	1	0	1	0	1	0	0	1	0	0	1	0	1	0	0	0	0	0	54	94	00
1	0	0	1	1	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	09	80	00
1	0	1	0	0	1	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	D1	00	00
1	0	1	0	1	0	1	1	0	0	1	0	1	0	1	0	1	1	0	0	0	0	0	0	65	58	00
1	0	1	1	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	09	80	00

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de dieciséis líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P16.5). Se puede comprobar realizando el circuito de la figura P16.7.

Tabla P16.5 Tabla de funcionamiento del selector de entradas para el sistema robotizado clasificador de paquetes.

Prueba				SI
I3	I2	I1	I0	
0	0	0	0	AUX
0	0	0	1	BH
0	0	1	0	SVS
0	0	1	1	SVI
0	1	0	0	SHI
0	1	0	1	SHD
0	1	1	0	SPCV
0	1	1	1	SG
1	0	0	0	SP
1	0	0	1	SPB
1	0	1	0	F1_COLOR
1	0	1	1	F2_COLOR
1	1	0	0	BCR
1	1	0	1	BCV

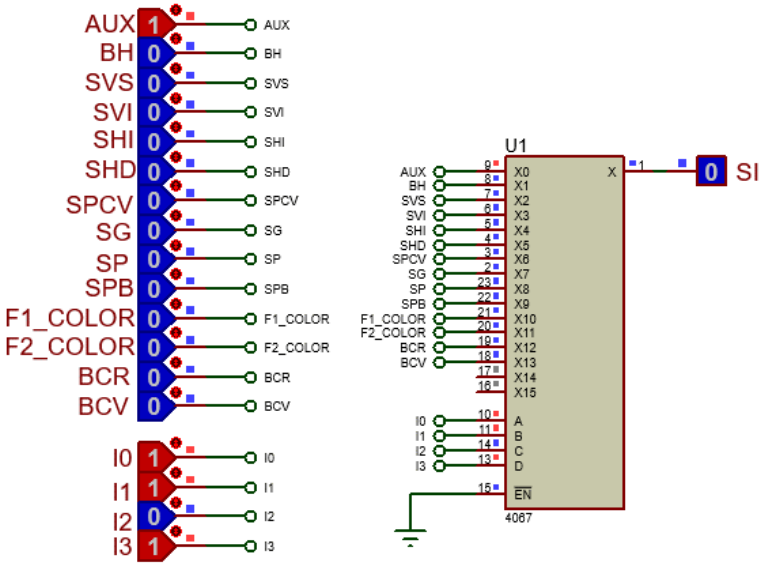


Figura P16.7 Multiplexor 4067 para el selector de entradas del sistema robotizado clasificador de paquetes.

Teniendo en cuenta que `BCR = J` y `BCV = K`, donde `J` y `K` son los bits de las máquinas de estados para guardar un registro si se detectó una caja roja o verde, la función booleana del selector de entradas queda:

```
SI = AUX&!I3&!I2&!I1&!I0 | BH&!I3&!I2&!I1&I0 | SVS&!I3&!I2&I1&!I0 |
     SVI&!I3&!I2&I1&I0 |SHI&!I3&I2&!I1&!I0 | SHD&!I3&I2&!I1&I0 |
     SPCV&!I3&I2&I1&!I0 | SG&!I3&I2&I1&I0 | SP&I3&!I2&!I1&!I0 |
     SPB&I3&!I2&!I1&I0 | F1_COLOR&I3&!I2&I1&!I0 | F2_COLOR&I3&!I2&I1&I0
     | J&I3&I2&!I1&!I0 | K&I3&I2&!I1&I0;
```

Para obtener el valor de la lógica, se debe hacer la operación XOR entre el selector de entradas y el valor de VF (ver figura P16.4).

Por lo tanto, la función booleana de la lógica queda:

```
SL = VF ^ (AUX&!I3&!I2&!I1&!I0 | BH&!I3&!I2&!I1&I0 |
     SVS&!I3&!I2&I1&!I0 | SVI&!I3&!I2&I1&I0 | SHI&!I3&I2&!I1&!I0 |
     SHD&!I3&I2&!I1&I0 | SPCV&!I3&I2&I1&!I0 | SG&!I3&I2&I1&I0 |
     SP&I3&!I2&!I1&!I0 | SPB&I3&!I2&!I1&I0 | F1_COLOR&I3&!I2&I1&!I0 |
     F2_COLOR&I3&!I2&I1&I0 | J&I3&I2&!I1&!I0 | K&I3&I2&!I1&I0);
```

Se utiliza un contador con carga paralela que indica el estado siguiente. El contador con carga paralela es implementado por el PIC16F1939. Si el valor a la salida de la lógica es igual a ‘1’, el contador carga la información binaria de la liga a la memoria. Si el valor de la lógica es igual a ‘0’, se cuenta al siguiente estado binario. A continuación, se obtienen las expresiones lógicas para un contador de cinco bits por el método de variable suscrita (ver tabla P16.6).

Tabla P16.6 Mapa de Karnaugh general de transición de estados para un contador de cinco bits.

A B \ C D E	0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0
0 0	00000	00001	00011	00010	00110	00111	00101	00100
	00001	00010	00100	00011	00111	01000	00110	00101
0 1	01000	01001	01011	01010	01110	01111	01101	01100
	01001	01010	01100	01011	01111	10000	01110	01101
1 1	11000	11001	11011	11010	11110	11111	11101	11100
	11001	11010	11100	11011	11111	00000	11110	11101
1 0	10000	10001	10011	10010	10110	10111	10101	10100
	10001	10010	10100	10011	10111	11000	10110	10101

A B \ C D E	000	001	011	010	110	111	101	100
00	0	0	0	0	0	0	0	0
01	0	0	0	0	0	1	0	0
11	1	1	1	1	1	0	1	1
10	1	1	1	1	1	1	1	1

```
FFA = A&!C&E | A&D&!E | A&!D |A&!B
    | !A&B&C&D&E;
```

A B \ C D E	000	001	011	010	110	111	101	100
00	0	0	1	0	1	0	1	1
01	0	0	1	0	1	0	1	1
11	0	0	1	0	1	0	1	1
10	0	0	1	0	1	0	1	1

```
FFC = !C&D&E | C&D&!E  |  C&!D;
```

A B \ C D E	000	001	011	010	110	111	101	100
00	0	0	0	0	0	1	0	0
01	1	1	1	1	1	0	1	1
11	1	1	1	1	1	0	1	1
10	0	0	0	0	0	1	0	0

```
FFB = B&!C&E | B&D&!E | B&!D |
    !B&C&D&E;
```

A B \ C D E	000	001	011	010	110	111	101	100
00	0	1	0	1	1	0	1	0
01	0	1	0	1	1	0	1	0
11	0	1	0	1	1	0	1	0
10	0	1	0	1	1	0	1	0

```
FFD = D&!E | !D&E = D^E;
```

A B \ C D E	000	001	011	010	110	111	101	100
00	1	0	0	1	1	0	0	1
01	1	0	0	1	1	0	0	1
11	1	0	0	1	1	0	0	1
10	1	0	0	1	1	0	0	1

```
FFE = !E;
```

La explicación para obtener las expresiones que controlan el motor paso a paso se encuentra en la Práctica 12.

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P16.8, figura P16.9). Se carga en el controlador los archivos con extensión “HEX” de las memorias y los archivos “COF” o “HEX” del PIC16F1939. Para poder visualizar de manera más rápida el movimiento del motor paso a paso, se debe utilizar una frecuencia de 30 Hz y el ángulo en el que gira el motor paso a paso debe ser de 3.6°. Se modificó la biblioteca PLD.h para tener más puertos de entradas en el PIC.

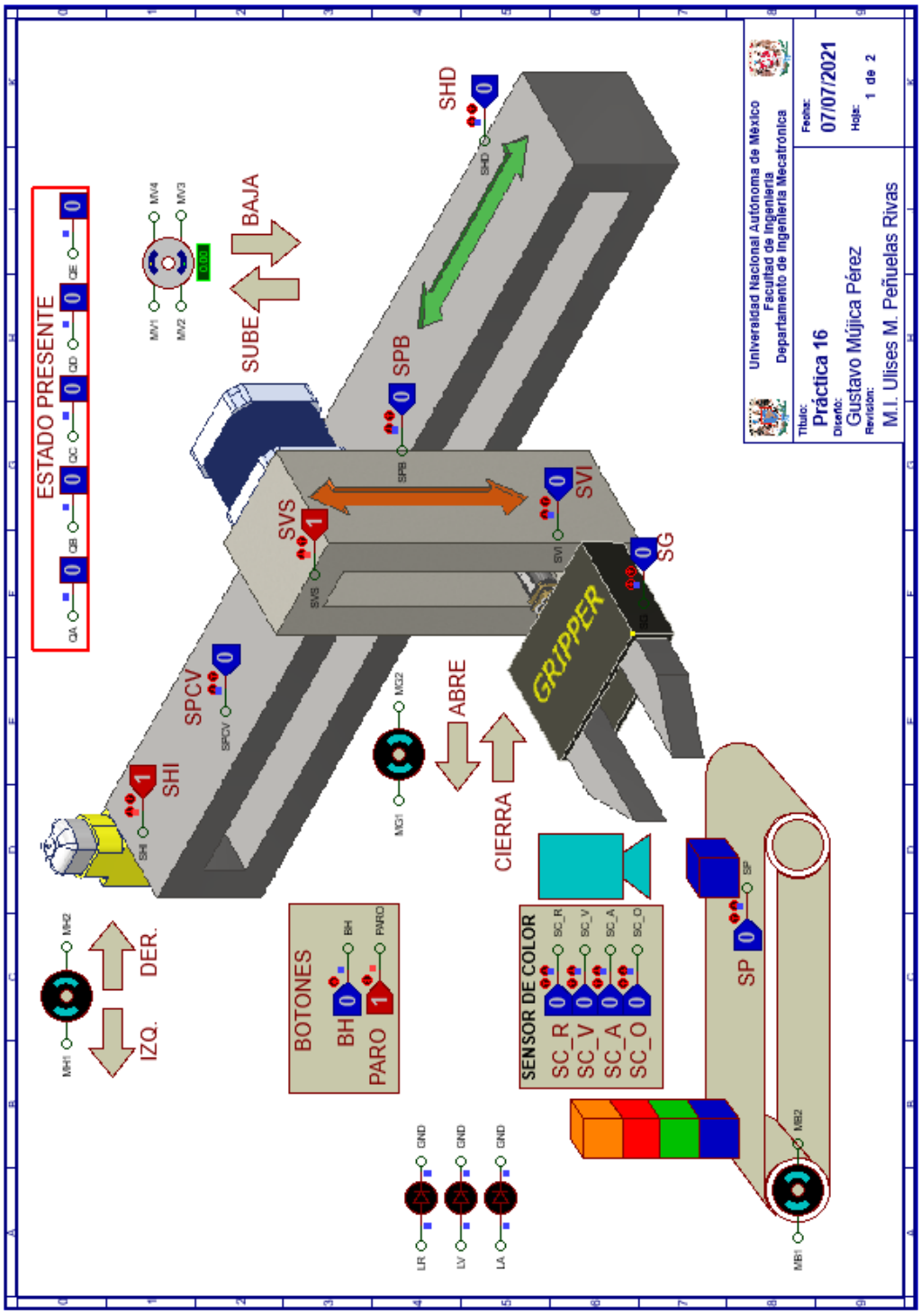


Figura P16.8 Interfaz hombre-máquina para el controlador de la Práctica 16 hoja 1/2.

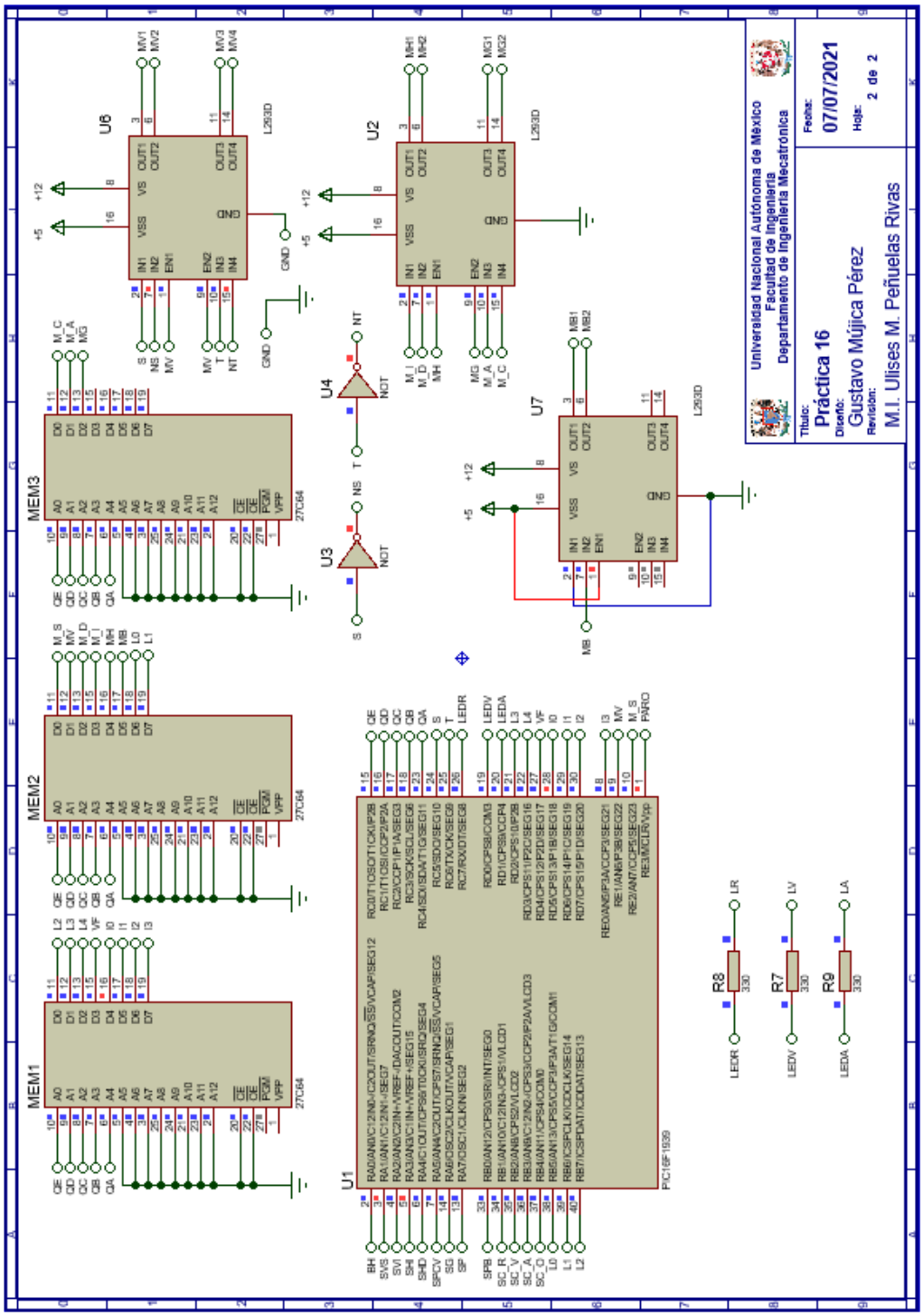


Figura P16.9 Esquema electrónico para el controlador de la Práctica 16 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P16.10 a figura P16.13) y se obtendrán archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD_SROBOTIZADO.h>     // Carga biblioteca PLD.h

//***CONTADOR CON CARGA PARALELA***

//ENTRADAS
//LIGA
#define L0 B5
#define L1 B6
#define L2 B7

#define L3 D2
#define L4 D3

//SALIDAS
#define A C4 //FFA
#define B C3 //FFB
#define C C2 //FFC
#define D C1 //FFD
#define E C0 //FFD

//***LOGICA***

//ENTRADAS
#define BH   A0
#define SVS  A1
#define SVI  A2
#define SHI  A3
#define SHD  A4
#define SPCV A5
#define SG   A6
#define SP   A7

#define SPB  B0
#define SC_R B1
#define SC_V B2
#define SC_A B3
#define SC_O B4

//VF
#define VF D4

//PRUEBAS
```

Figura P16.10 Código de la Práctica 16 parte 1.

```
43: //PRUEBAS
44: #define I0 D5
45: #define I1 D6
46: #define I2 D7
47:
48: #define I3 E0
49:
50: //****MOTOR A PASOS****
51: //ENTRADAS
52: #define MVn  E1
53: #define M_Sn E2
54:
55: //SALIDAS
56: #define S  C5
57: #define T  C6
58: //LEDS
59: #define LEDR  C7
60:
61: #define LEDV  D0
62: #define LEDA  D1
63:
64:
65: //***VARIABLES INTERMEDIAS***
66:
67: short SL=0;  //Salida lógica
68: short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
69: short At=0, Bt=0, Ct=0, Dt=0, Et=0; //Variables intermedias de los FF
70: short LD4,LD3,LD2,LD1,LD0; //Variables intermedias de la liga
71: short J,K,L,Jn,Kn,Ln; // Variables auxiliares para registrar
72:                         // si se detectó una caja color roja o verde respectivamente
73: short Sn,Tn; // Variables intermedias motor a pasos
74: short F1_COLOR, F2_COLOR; //Funciones
75:
76: void main ()
77: {
78: pld_ini();  // INICIALIZA AL PIC COMO PLD
79: pld_555(30);   // Genera señal cuadrada en Hz,
80:                   //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
81:
82:    //LOOP INFINITO
83:    while(1)
84:    {
```

Figura P16.11 Código de la Práctica 16 parte 2.

```
 84:     {
 85:     //****CIRCUITO COMBINACIONAL****
 86:     LD4=L4; LD3=L3; LD2=L2; LD1=L1; LD0=L0;  /*ALMACENA DATOS HASTA EL CAMBIO DEL RELOJ,
 87:                                                SIMULANDO UN REGISTRO, EVITANDO ASÍ
 88:                                             QUE SE MODIFIQUEN LOS VALORES DE LA MEMORIA*/
 89:     LEDR=J;//LED ROJO
 90:     LEDV=K;// LED VERDE
 91:     LEDA=L;//LED AZUL
 92:
 93:     //FUNCIONES
 94:     F1_COLOR = SC_R | SC_V | SC_A | SC_O;
 95:     F2_COLOR = SC_R | SC_V | SC_A;
 96:
 97:
 98:     //SALIDA LÓGICA
 99:    SL =  VF ^ (AUX&!I3&!I2&!I1&!I0 | BH&!I3&!I2&!I1&I0 | SVS&!I3&!I2&I1&!I0 |
100:          SVI&!I3&!I2&I1&I0 | SHI&!I3&I2&!I1&!I0 | SHD&!I3&I2&!I1&I0 |
101:          SPCV&!I3&I2&I1&!I0 | SG&!I3&I2&I1&I0 | SP&I3&!I2&!I1&!I0 |
102:          SPB&I3&!I2&!I1&I0 | F1_COLOR&I3&!I2&I1&!I0 |
103:          F2_COLOR&I3&!I2&I1&I0 | J&I3&I2&!I1&!I0 | K&I3&I2&!I1&I0);
104:
105:     //****CIRCUITO SECUENCIAL ****
106:
107:    //    if (!CLK) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO
108:     if (!out_555) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO,
109:                   //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
110:
111:     {//SECCIÓN DE OPERACIONES DEL CONTADOR CON CARGA PARALELA
112:     At = A&!C&E | A&D&!E | A&!D |A&!B | !A&B&C&D&E;
113:     Bt = B&!C&E | B&D&!E | B&!D | !B&C&D&E;
114:     Ct = !C&D&E | C&D&!E  |  C&!D;
115:     Dt = D^E;
116:     Et = !E;
117:
118:     //REGISTRO CAJA COLOR ROJO O VERDE
119:     Jn = !J&SC_R | J&!SHD;
120:     Kn = !K&SC_V | K&!SPCV;
121:     Ln = !L&SC_A | L&!SHI;
122:     //MOTOR A PASOS
123:     Sn = (!T&!M_Sn | T&M_Sn)&MVn;
124:     Tn = (!S&M_Sn | S&!M_Sn)&MVn;
125:     }
```

Figura P16.12 Código de la Práctica 16 parte 3.

```
125:     }
126:
127:     else
128:     {//SECCIÓN DE MEMORIZACIÓN
129:     //CUENTA CON SL=1 Y CARGA CON SL=0
130:     A = SL&At | !SL&LD4;
131:     B = SL&Bt | !SL&LD3;
132:     C = SL&Ct | !SL&LD2;
133:     D = SL&Dt | !SL&LD1;
134:     E = SL&Et | !SL&LD0;
135:
136:     J=Jn;
137:     K=Kn;
138:     L=Ln;
139:
140:     S=Sn;
141:     T=Tn;
142:
143:    /* while(clk){}/*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
144:                  POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
145:                  DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
146:                  LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ*/
147:
148:     while(out_555){} /*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
149:                       POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
150:                       DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
151:                       LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ,
152:                       EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO*/
153:     }
154:     }
155: }
```

Figura P16.13 Código de la Práctica 16 parte 4.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 17 Sistema de planta baja; diseño con memoria y direccionamiento por trayectoria

Introducción

A través de una maqueta, se simuló el funcionamiento de un elevador de un edificio de tres pisos por medio de una maqueta que está dividida en tres partes, las cuales son: cubo del elevador, puerta que abre y cierra, y el tablero de control.

El cubo tiene cuatro niveles, los cuales son: planta baja, piso 1, piso 2 y piso 3. Donde un carro sube y baja simulando el movimiento del elevador. Cada uno de los pisos tiene controles a un costado de la puerta para subir o bajar de piso, así como sensores de presencia que indican en donde está el carro.

El tablero de control simula el control interno del elevador. Tiene botones para: dirigirse a los 4 niveles del elevador, detener elevador, abrir y cerrar la puerta, así como un indicador del piso en el que se encuentra el elevador.

Para comprender mejor el funcionamiento del elevador este se dividió en cinco subsistemas, los cuales son: planta baja, piso 1, piso 2, piso 3 y el sistema de abrir/cerrar puerta.

En esta práctica se habla sobre el sistema de planta baja. Este sistema junto con el sistema de abrir/cerrar puerta, piso 1, piso 2 y piso 3 forman el sistema del elevador (ver figura P17.1, figura P17.2).

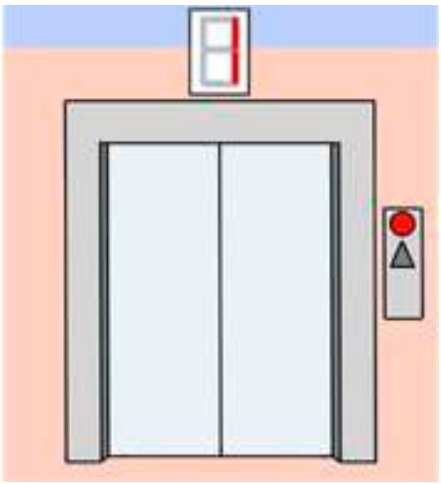


Figura P17.1 Planta baja de la maqueta.

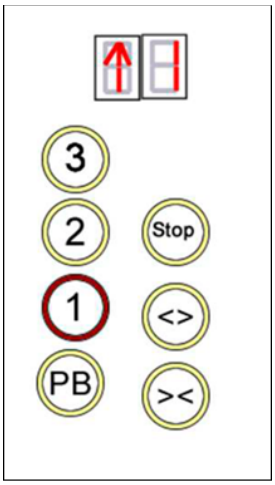


Figura P17.2 Tablero de control interno del cubo del elevador.

El diseño con memoria y direccionamiento por trayectoria guarda el estado siguiente de la salida de cada estado de la carta ASM en una localidad de memoria. La porción de la memoria que indica el estado siguiente es llamada “liga”, mientras que la porción que indica las salidas se llama “la parte de las salidas” [1].

La arquitectura de un diseño con memoria y direccionamiento por trayectoria se muestra en la figura P17.3, donde:

A = estado siguiente

B = entradas del estado siguiente

C = entradas del estado presente

D = estado presente

E = dirección del estado siguiente

F = salidas del estado presente.

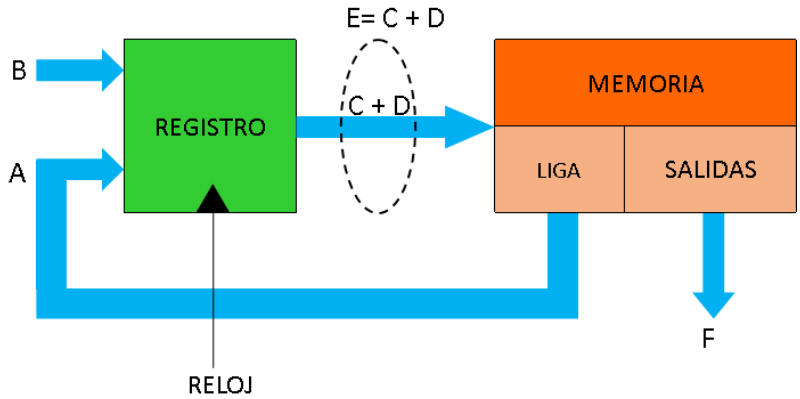


Figura P17.3 Arquitectura de un controlador con memoria y direccionamiento por trayectoria.

En este tipo de direccionamiento, todas las salidas de la carta ASM deberán depender del estado presente y de los valores de entrada.

Objetivo

Diseñar un controlador para el sistema de planta baja, por medio del método de diseño con memoria y direccionamiento por trayectoria.

Descripción

Primero se diseña una carta ASM para el sistema de planta baja por medio del método de diseño con memoria y direccionamiento por trayectoria. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P17.1 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema de planta baja se necesitan las siguientes señales de entrada:

- un botón detecta la solicitud para subir a otro piso desde la planta baja
- un botón dentro del elevador detecta la solicitud para ir hacia la planta baja
- un sensor indica que el elevador se encuentra en la planta baja
- un botón dentro del elevador detecta la solicitud para detener el elevador.

Como salida se requiere de las siguientes señales:

- una señal se activa para que el elevador baje hasta llegar a la planta baja.

Tabla P17.1 Entradas y salidas para el sistema de planta baja.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
B_STP	D0 del registro	I	Botón para detener el elevador
S_PB	D1 del registro	I	Sensor de planta baja
B_SPB	D2 del registro	I	Botón subir desde planta baja. Botón para subir desde planta baja
B_PB	D2 del registro	I	Botón de planta baja. Botón para ir a planta baja
M_B	D0 de la memoria	O	Señal para bajar el elevador.

Notas de diseño

a) El botón de planta baja y el botón de paro se encuentran en el tablero de control, los cuales simulan el control de los botones dentro del elevador.
b) El botón para subir desde planta baja se encuentra ubicado a un costado de la entrada del elevador en la planta baja.
c) Sólo se podrá detener elevador cuando se encuentre en movimiento.
d) El botón de paro debe mantenerse oprimido para que no se mueva el elevador.

Reglas de funcionamiento

- B_STP: botón de paro
  1 = se oprimió el botón de paro
  0 = no se oprimió el botón de paro
- S_PB: sensor de planta baja
  1 = el elevador está en la planta baja
  0 = no está el elevador en la planta baja
- B_SPB: botón subir desde planta baja
  1 = se oprimió el botón de subir desde planta baja
  0 = no se oprimió el botón de subir desde planta baja
- B_PB: botón de planta baja
  1 = se oprimió el botón de planta baja
  0 = no se oprimió el botón de planta baja.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado. El algoritmo de la máquina de estados se puede ver en la figura P17.4.

Estado ‘00’ – CONTROL

En el primer estado se está a la espera de que se oprima un botón para que el elevador se dirija a la planta baja. Si se oprime el botón de planta baja (B_PB) o se oprime el botón de subir desde planta baja (B_SPB), avanza al Estado ‘01’ para detectar en que piso se encuentra el elevador. De lo contrario, permanece en el Estado ‘00’ en espera de que se oprima un botón.

Estado ‘01’ – DETPB

En este estado se verifica si el elevador se encuentra en la planta baja. Cuando el sensor (S_PB), no detecta el elevador en la planta baja, avanza al Estado ‘10’ para bajar el elevador ya que se encuentra en alguno de los pisos superiores. De lo contrario, se regresará al Estado ‘00’ para iniciar el proceso nuevamente.

Estado ‘10’ – BAJAE

En este estado se activa la señal (M_B) para que el elevador baje. Si el sensor (S_PB), detecta que el elevador se encuentra en la planta baja, el sistema regresa al Estado ‘00’ para iniciar el proceso nuevamente. Cuando el sensor no detecta el elevador en la planta baja, avanza a revisar el botón que detiene al elevador. Si esta presionado el botón (B_STP), el sistema avanza al Estado ‘11’ para detener el elevador. De lo contrario, permanece en el Estado ‘10’ bajando el elevador.

Estado ‘11’ – STOP

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no está presionado, el sistema regresa al Estado ‘10’ para seguir bajando elevador. De lo contrario, el elevador permanece en el Estado ’11’ deteniendo el elevador.

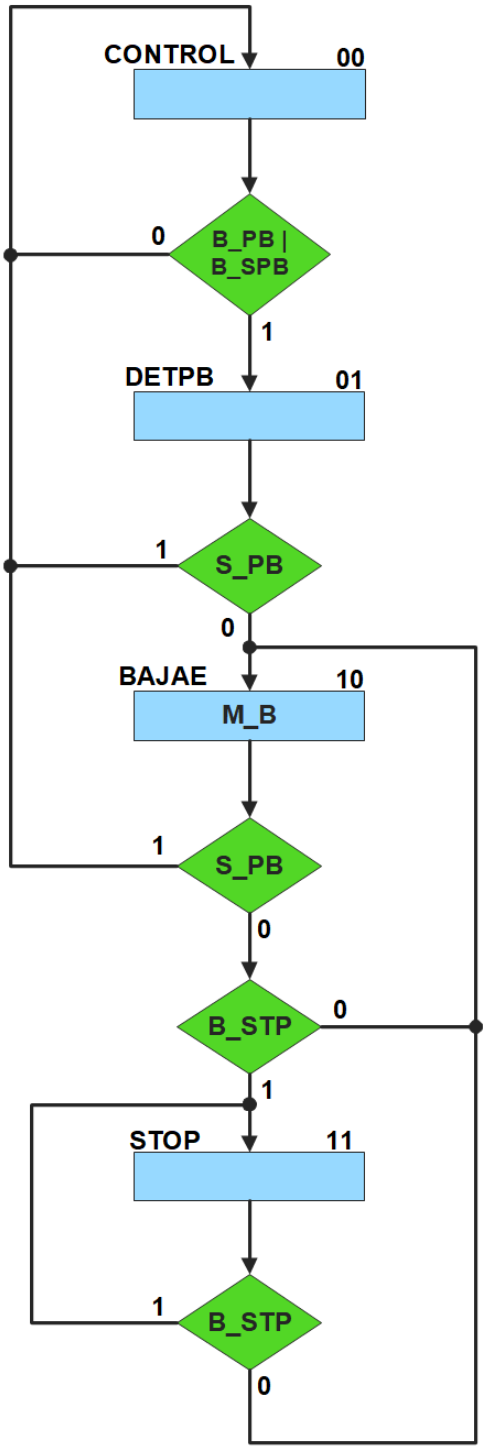


Figura P17.4 Carta ASM del sistema de planta baja.

Solución

Se debe llenar la tabla P17.2, donde se muestra el contenido de la memoria usando el método de diseño con memoria y direccionamiento por trayectoria. Para cada estado es necesario considerar todas las posibles combinaciones de las variables de entrada aun cuando algunas de ellas no se utilicen [1].

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘00’.

Debido a que hay tres variables de entrada, se deben considerar 8 posibles combinaciones de éstas para cada estado. Si en el Estado ‘00’ B_PB o B_SPB es igual a ‘1’, el estado siguiente será el Estado ‘01’ independientemente de los valores de las otras variables. En el Estado ‘00’ la señal M_B no está activada, por lo que se coloca un ‘0’ en la parte de salidas de todo el estado.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria, ver figura P17.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P17.2 Contenido de la memoria para el sistema de planta baja.

	Dirección de memoria					Contenido de memoria			
	Estado presente		Entradas			Liga		Salidas	HEX
	QA	QB	B_PB | B_SPB	S_PB	B_STP	QA	QB	M_B	
Estado 00	0	0	0	0	0	0	0	0	00
	0	0	0	0	1	0	0	0	00
	0	0	0	1	0	0	0	0	00
	0	0	0	1	1	0	0	0	00
	0	0	1	0	0	0	1	0	02
	0	0	1	0	1	0	1	0	02
	0	0	1	1	0	0	1	0	02
	0	0	1	1	1	0	1	0	02
Estado 01	0	1	0	0	0	1	0	0	04
	0	1	0	0	1	1	0	0	04
	0	1	0	1	0	0	0	0	00
	0	1	0	1	1	0	0	0	00
	0	1	1	0	0	1	0	0	04
	0	1	1	0	1	1	0	0	04
	0	1	1	1	0	0	0	0	00
	0	1	1	1	1	0	0	0	00
Estado 10	1	0	0	0	0	1	0	1	05
	1	0	0	0	1	1	1	1	07
	1	0	0	1	0	0	0	1	01
	1	0	0	1	1	0	0	1	01
	1	0	1	0	0	1	0	1	05
	1	0	1	0	1	1	1	1	07
	1	0	1	1	0	0	0	1	01
	1	0	1	1	1	0	0	1	01
Estado 11	1	1	0	0	0	1	0	0	04
	1	1	0	0	1	1	1	0	06
	1	1	0	1	0	1	0	0	04
	1	1	0	1	1	1	1	0	06
	1	1	1	0	0	1	0	0	04
	1	1	1	0	1	1	1	0	06
	1	1	1	1	0	1	0	0	04
	1	1	1	1	1	1	1	0	06

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P17.5, figura P17.6). Se carga en el controlador el archivo con extensión “HEX” de la memoria.

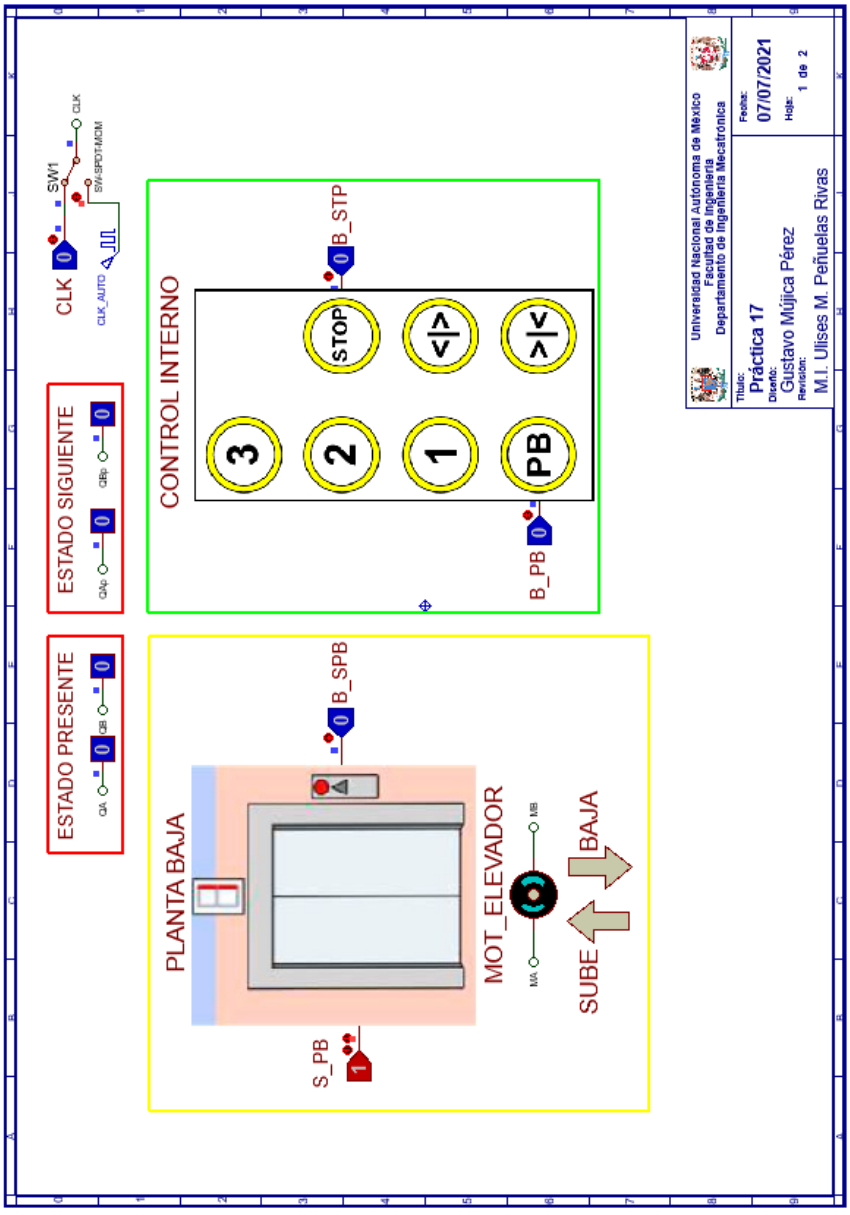


Figura P17.5 Interfaz hombre-máquina para el controlador de la Práctica 17 hoja 1/2.

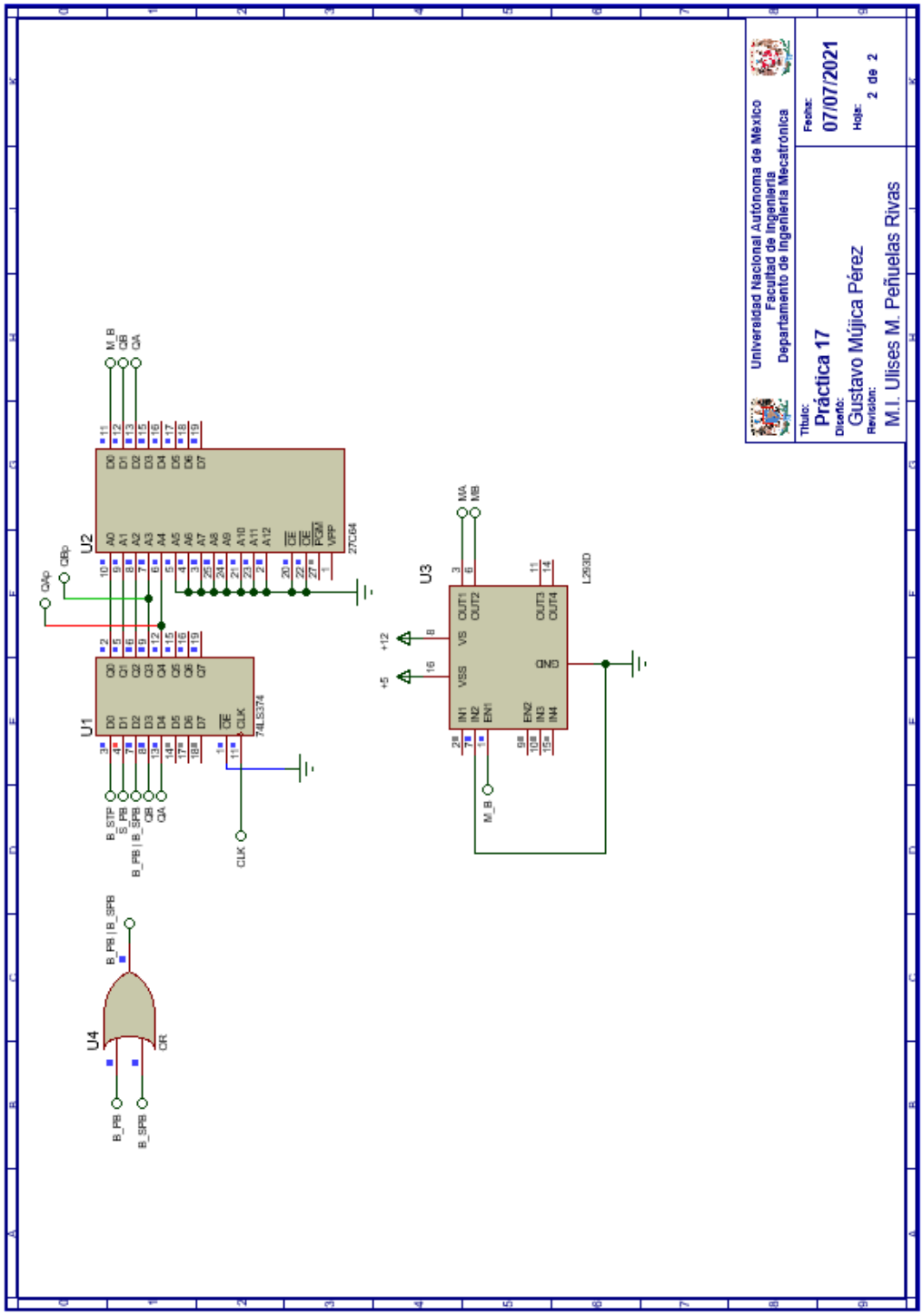


Figura P17.6 Esquema electrónico para el controlador de la Práctica 17 hoja 2/2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 18 Sistema de piso 1; diseño con memoria y direccionamiento por trayectoria

Introducción

A través de una maqueta, se simuló el funcionamiento de un elevador de un edificio de tres pisos por medio de una maqueta que está dividida en tres partes, las cuales son: cubo del elevador, puerta que abre y cierra, y el tablero de control.

El cubo tiene cuatro niveles, los cuales son: planta baja, piso 1, piso 2 y piso 3. Donde un carro sube y baja simulando el movimiento del elevador. Cada uno de los pisos tiene controles a un costado de la puerta para subir o bajar de piso, así como sensores de presencia que indican en donde está el carro.

El tablero de control simula el control interno del elevador. Tiene botones para: dirigirse a los 4 niveles del elevador, detener elevador, abrir y cerrar la puerta, así como un indicador del piso en el que se encuentra el elevador.

Para comprender mejor el funcionamiento del elevador este se dividió en cinco subsistemas, los cuales son: planta baja, piso 1, piso 2, piso 3 y el sistema de abrir/cerrar puerta.

En esta práctica se habla sobre el sistema de piso 1. Este sistema junto con el sistema de abrir/cerrar puerta, planta baja, piso 2 y piso 3 forman el sistema del elevador. (ver figura P18.1, figura P18.2).

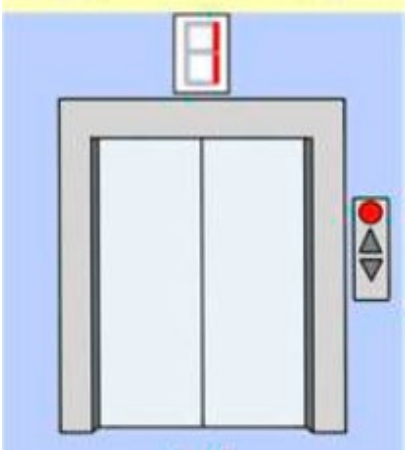


Figura P18.1 Piso 1 de la maqueta.

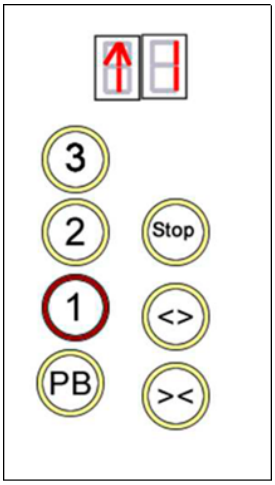


Figura P18.2 Tablero de control interno del cubo del elevador.

El diseño con memoria y direccionamiento por trayectoria guarda el estado siguiente de la salida de cada estado de la carta ASM en una localidad de memoria. La porción de la memoria que indica el estado siguiente es llamada “liga”, mientras que la porción que indica las salidas se llama “la parte de las salidas” [1].

La arquitectura de un diseño con direccionamiento por trayectoria se muestra en la figura P18.3, donde:

A = estado siguiente

B = entradas del estado siguiente

C = entradas del estado presente

D = estado presente

E = dirección del estado siguiente

F = salidas del estado presente.

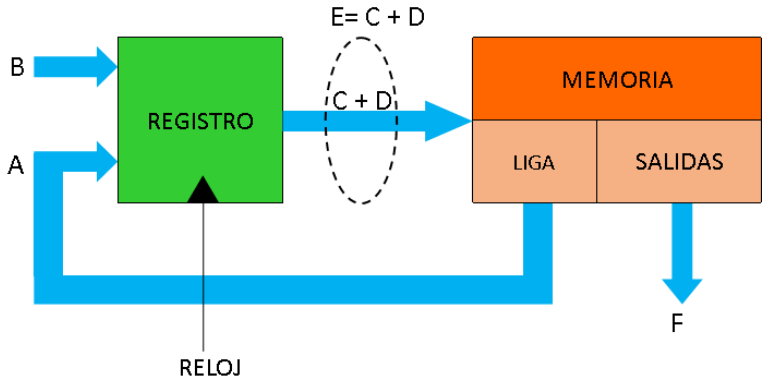


Figura P18.3 Arquitectura de un controlador con memoria y direccionamiento por trayectoria.

En este tipo de direccionamiento, todas las salidas de la carta ASM deberán depender del estado presente y de los valores de entrada.

Objetivo

Diseñar un controlador para el sistema de piso 1, por medio del método de direccionamiento por trayectoria.

Descripción

Primero se diseña una carta ASM para el sistema de piso 1 por medio del método de diseño con memoria y direccionamiento por trayectoria. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P18.1 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema de piso 1 se necesitan señales de entrada:

- un botón detecta la solicitud para subir a otro piso desde el piso 1 y otro para bajar desde el piso 1.
- un botón dentro del elevador detecta la solicitud para ir hacia el piso 1.
- un sensor indica que el elevador se encuentra en el piso 1 y otro indica que se encuentra en la planta baja.
- un botón dentro del elevador detecta la solicitud para detener el elevador.

Como salida se requiere de las siguientes señales:

- una señal se activa para que el elevador baje hasta llegar al piso 1 y otra se activa para que suba hasta llegar al piso 1.

Tabla P18.1 Entradas y salidas para el sistema de piso 1.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
B_STP	D0 del registro	I	Botón de paro para detener el elevador
S_PB	D1 del registro	I	Sensor de planta baja
S_P1	D2 del registro	I	Sensor de piso 1
B_P1	D3 del registro	I	Botón para ir al piso 1
B_SP1	D3 del registro	I	Botón para subir desde el piso 1
B_BP1	D3 del registro	I	Botón para bajar desde el piso 1
M_S	D0 de la memoria	O	Señal para subir el elevador
M_B	D1 de la memoria	O	Señal para bajar el elevador.

Notas de diseño

a) El botón de piso 1 y el botón de paro se encuentran en el tablero de control, el cual simula el control de los botones dentro del elevador
b) El botón para subir desde el piso 1 y el botón para bajar desde el piso 1 se encuentran ubicados a un costado de la entrada del elevador en el piso 1
c) Sólo se puede detener el elevador cuando éste se encuentre en movimiento
d) El botón de paro debe mantenerse presionado para que no se mueva el elevador.

Reglas de funcionamiento

- B_STP: Botón de paro
  1 = se oprimió el botón de paro
  0 = no se oprimió el botón de paro
- S_PB: sensor de planta baja
  1 = el elevador está en la planta baja
  0 = no está el elevador en la planta baja
- S_P1: sensor de piso 1
  1 = el elevador está en el piso 1
  0 = no está el elevador en el piso 1
- B_P1: botón de piso 1
  1 = se oprimió el botón de piso 1
  0 = no se oprimió el botón de piso 1
- B_SP1: botón subir desde piso 1
  1 = se oprimió el botón de subir desde el piso 1
  0 = no se oprimió el botón de subir desde el piso 1
- B_BP1: botón bajar desde piso 1
  1 = se oprimió el botón de bajar desde el piso 1
  0 = no se oprimió el botón de bajar desde el piso 1.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado. El algoritmo de la máquina de estados se puede ver en la figura P18.4.

Estado '000' – CONTROL

En el primer estado se está a la espera de que se oprima un botón para que el elevador se dirija al piso 1. Si se oprime el botón para subir (B_SP1) o el botón para bajar (B_BP1) desde el piso 1, o el botón de piso 1 (B_P1) del tablero, el sistema avanza al Estado '001' para detectar en que piso se encuentra el elevador. De lo contrario, permanece en el Estado '000' en espera de que se oprima un botón.

Estado '001' – DETP

En este estado se detecta en que piso se encuentra el elevador. Si el sensor (S_P1), detecta que el elevador se encuentra en el piso 1, el sistema regresa al Estado '000' para iniciar el proceso nuevamente. De lo contrario, avanza a revisar el sensor de planta baja. Si el sensor (S_PB), no detecta el elevador en la planta baja, avanza al Estado '010' para bajar el elevador al piso 1 ya que se encuentra en alguno de los pisos superiores. De lo contrario, avanza al Estado '100' para subir el elevador al piso 1 debido a que se encuentra en la planta baja.

Estado '010' – BAJAE

En este estado se activa la señal (M_B) para que el elevador baje. Si el sensor (S_P1) detecta que el elevador se encuentra en el piso 1, el sistema regresa al Estado '000' para iniciar el proceso nuevamente. De lo contrario, avanza a revisar el botón que detener el elevador. Si el botón (B_STP), no está presionado, permanece en el Estado '010' para seguir bajando el elevador. De lo contrario, avanza al Estado '011' para detener el elevador.

Estado ‘011’ – STOP1

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no está presionado, el sistema regresa al Estado ‘010’ para seguir bajando el elevador. De lo contrario, el elevador permanece detenido en el Estado ’011’.

Estado ‘100’ – SUBEE

En este estado se activa la señal (M_S), para que el elevador suba. Si el sensor (S_P1), detecta que el elevador se encuentra en el piso 1, el sistema regresa al Estado ‘000’ para iniciar el proceso nuevamente. De lo contrario, avanza a revisar el botón que detiene el elevador. Si el botón (B_STP), no está presionado, permanece en el Estado ‘100’ para seguir subiendo el elevador. De lo contrario, avanza al Estado ‘101’ para detener el elevador.

Estado ‘101’ – STOP2

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no esté presionado, el sistema regresa al Estado ‘100’ para seguir subiendo el elevador. De lo contrario, el elevador permanece detenido en el Estado ’101’.

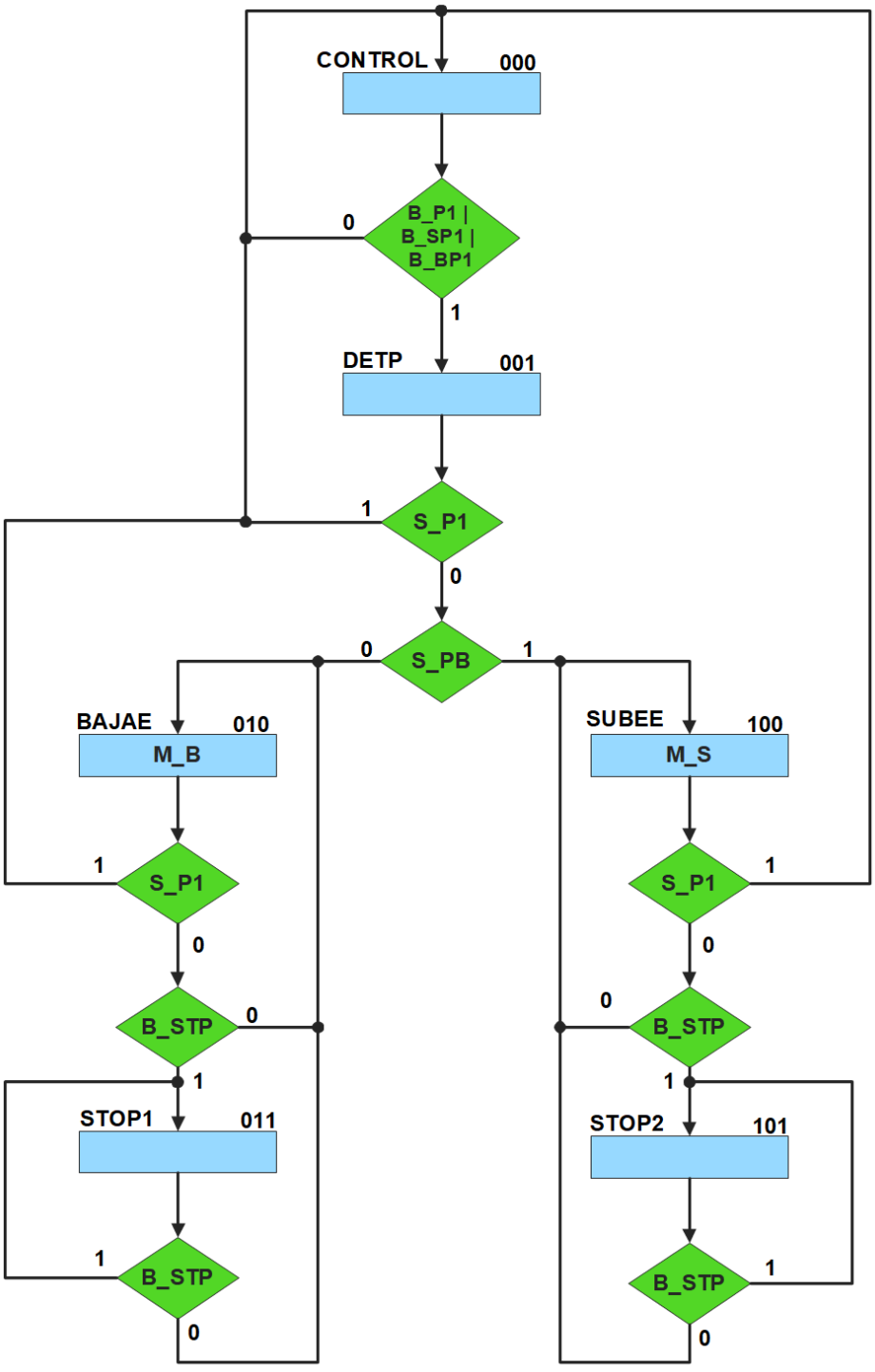


Figura P18.4 Carta ASM del sistema de piso 1.

Solución

Se debe llenar la tabla P18.2 a la tabla P18.4, en éstas se muestra el contenido de la memoria usando el método de diseño con memoria y direccionamiento por trayectoria. Para cada estado es necesario considerar todas las posibles combinaciones de las variables de entrada aun cuando algunas de ellas no se utilicen [1].

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘000’.

Debido a que hay cuatro variables de entrada, se deben considerar 16 posibles combinaciones para cada estado. Si en el Estado ‘000’ B_P1 o B_SP1 o B_BP1 es igual a ‘1’, el estado siguiente será el Estado ‘001’ independientemente de los valores de las otras variables. En el Estado ‘000’ las señales M_B y M_S no están activadas, por lo que se coloca un ‘0’ en la parte de salidas de todo el estado.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P18.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P18.2 Contenido de la memoria para el sistema de piso 1 parte 1.

	Dirección de memoria							Contenido de memoria					
	Estado presente			Entradas				Liga			Salidas		HEX
	QA	QB	QC	B_P1 | B_SP1 | B_BP1	S_P1	S_PB	B_STP	QA	QB	QC	M_B	M_S	
Estado 000	0	0	0	0	0	0	0	0	0	0	0	0	00
	0	0	0	0	0	0	1	0	0	0	0	0	00
	0	0	0	0	0	1	0	0	0	0	0	0	00
	0	0	0	0	0	1	1	0	0	0	0	0	00
	0	0	0	0	1	0	0	0	0	0	0	0	00
	0	0	0	0	1	0	1	0	0	0	0	0	00
	0	0	0	0	1	1	0	0	0	0	0	0	00
	0	0	0	0	1	1	1	0	0	0	0	0	00
	0	0	0	1	0	0	0	0	0	1	0	0	04
	0	0	0	1	0	0	1	0	0	1	0	0	04
	0	0	0	1	0	1	0	0	0	1	0	0	04
	0	0	0	1	0	1	1	0	0	1	0	0	04
	0	0	0	1	1	0	0	0	0	1	0	0	04
	0	0	0	1	1	0	1	0	0	1	0	0	04
	0	0	0	1	1	1	0	0	0	1	0	0	04
	0	0	0	1	1	1	1	0	0	1	0	0	04
Estado 001	0	0	1	0	0	0	0	0	1	0	0	0	08
	0	0	1	0	0	0	1	0	1	0	0	0	08
	0	0	1	0	0	1	0	1	0	0	0	0	10
	0	0	1	0	0	1	1	1	0	0	0	0	10
	0	0	1	0	1	0	0	0	0	0	0	0	00
	0	0	1	0	1	0	1	0	0	0	0	0	00
	0	0	1	0	1	1	0	0	0	0	0	0	00
	0	0	1	0	1	1	1	0	0	0	0	0	00
	0	0	1	1	0	0	0	0	1	0	0	0	08
	0	0	1	1	0	0	1	0	1	0	0	0	08
	0	0	1	1	0	1	0	1	0	0	0	0	10
	0	0	1	1	0	1	1	1	0	0	0	0	10
	0	0	1	1	1	0	0	0	0	0	0	0	00
	0	0	1	1	1	0	1	0	0	0	0	0	00
	0	0	1	1	1	1	0	0	0	0	0	0	00
	0	0	1	1	1	1	1	0	0	0	0	0	00

Tabla P18.3 Contenido de la memoria para el sistema de piso 1 parte 2.

	Dirección de memoria							Contenido de memoria					
	Estado presente			Entradas				Liga			Salidas		HEX
	QA	QB	QC	B_P1 | B_SP1 | B_BP1	S_P1	S_PB	B_STP	QA	QB	QC	M_B	M_S	
Estado 010	0	1	0	0	0	0	0	0	1	0	1	0	0A
	0	1	0	0	0	0	1	0	1	1	1	0	0E
	0	1	0	0	0	1	0	0	1	0	1	0	0A
	0	1	0	0	0	1	1	0	1	1	1	0	0E
	0	1	0	0	1	0	0	0	0	0	1	0	02
	0	1	0	0	1	0	1	0	0	0	1	0	02
	0	1	0	0	1	1	0	0	0	0	1	0	02
	0	1	0	0	1	1	1	0	0	0	1	0	02
	0	1	0	1	0	0	0	0	1	0	1	0	0A
	0	1	0	1	0	0	1	0	1	1	1	0	0E
	0	1	0	1	0	1	0	0	1	0	1	0	0A
	0	1	0	1	0	1	1	0	1	1	1	0	0E
	0	1	0	1	1	0	0	0	0	0	1	0	02
	0	1	0	1	1	0	1	0	0	0	1	0	02
	0	1	0	1	1	1	0	0	0	0	1	0	02
	0	1	0	1	1	1	1	0	0	0	1	0	02
Estado 011	0	1	1	0	0	0	0	0	1	0	0	0	08
	0	1	1	0	0	0	1	0	1	1	0	0	0C
	0	1	1	0	0	1	0	0	1	0	0	0	08
	0	1	1	0	0	1	1	0	1	1	0	0	0C
	0	1	1	0	1	0	0	0	1	0	0	0	08
	0	1	1	0	1	0	1	0	1	1	0	0	0C
	0	1	1	0	1	1	0	0	1	0	0	0	08
	0	1	1	0	1	1	1	0	1	1	0	0	0C
	0	1	1	1	0	0	0	0	1	0	0	0	08
	0	1	1	1	0	0	1	0	1	1	0	0	0C
	0	1	1	1	0	1	0	0	1	0	0	0	08
	0	1	1	1	0	1	1	0	1	1	0	0	0C
	0	1	1	1	1	0	0	0	1	0	0	0	08
	0	1	1	1	1	0	1	0	1	1	0	0	0C
	0	1	1	1	1	1	0	0	1	0	0	0	08
	0	1	1	1	1	1	1	0	1	1	0	0	0C

Tabla P18.4 Contenido de la memoria para el sistema de piso 1 parte 3.

	Dirección de memoria							Contenido de memoria					
	Estado presente			Entradas				Liga			Salidas		HEX
	QA	QB	QC	B_P1 | B_SP1 | B_BP1	S_P1	S_PB	B_STP	QA	QB	QC	M_B	M_S	
Estado 100	1	0	0	0	0	0	0	1	0	0	0	1	11
	1	0	0	0	0	0	1	1	0	1	0	1	15
	1	0	0	0	0	1	0	1	0	0	0	1	11
	1	0	0	0	0	1	1	1	0	1	0	1	15
	1	0	0	0	1	0	0	0	0	0	0	1	01
	1	0	0	0	1	0	1	0	0	0	0	1	01
	1	0	0	0	1	1	0	0	0	0	0	1	01
	1	0	0	0	1	1	1	0	0	0	0	1	01
	1	0	0	1	0	0	0	1	0	0	0	1	11
	1	0	0	1	0	0	1	1	0	1	0	1	15
	1	0	0	1	0	1	0	1	0	0	0	1	11
	1	0	0	1	0	1	1	1	0	1	0	1	15
	1	0	0	1	1	0	0	0	0	0	0	1	01
	1	0	0	1	1	0	1	0	0	0	0	1	01
	1	0	0	1	1	1	0	0	0	0	0	1	01
	1	0	0	1	1	1	1	0	0	0	0	1	01
Estado 101	1	0	1	0	0	0	0	1	0	0	0	0	10
	1	0	1	0	0	0	1	1	0	1	0	0	14
	1	0	1	0	0	1	0	1	0	0	0	0	10
	1	0	1	0	0	1	1	1	0	1	0	0	14
	1	0	1	0	1	0	0	1	0	0	0	0	10
	1	0	1	0	1	0	1	1	0	1	0	0	14
	1	0	1	0	1	1	0	1	0	0	0	0	10
	1	0	1	0	1	1	1	1	0	1	0	0	14
	1	0	1	1	0	0	0	1	0	0	0	0	10
	1	0	1	1	0	0	1	1	0	1	0	0	14
	1	0	1	1	0	1	0	1	0	0	0	0	10
	1	0	1	1	0	1	1	1	0	1	0	0	14
	1	0	1	1	1	0	0	1	0	0	0	0	10
	1	0	1	1	1	0	1	1	0	1	0	0	14
	1	0	1	1	1	1	0	1	0	0	0	0	10
	1	0	1	1	1	1	1	1	0	1	0	0	14

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P18.5,figura P18.6). Se carga en el controlador el archivo con extensión “HEX” de la memoria.

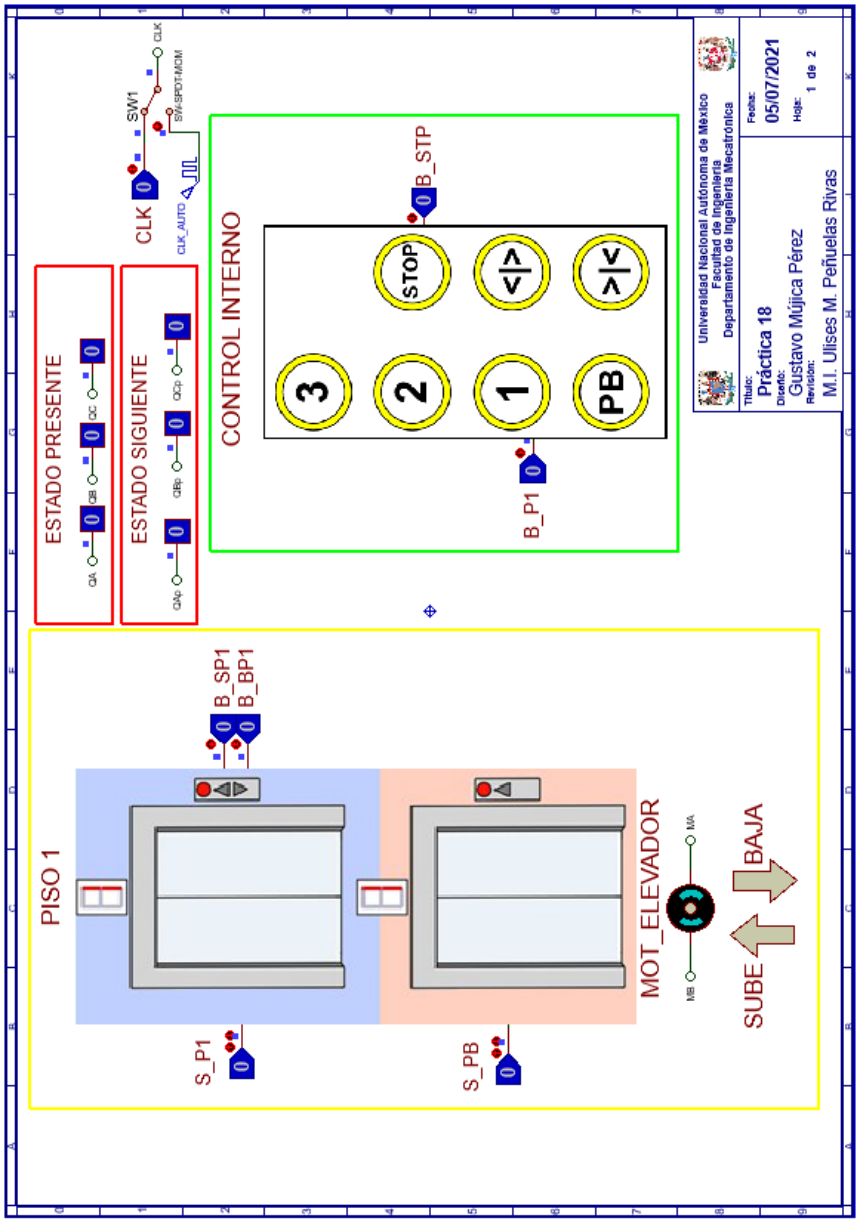


Figura P18.5 Interfaz hombre-máquina para el controlador de la Práctica 18 hoja 1/2.

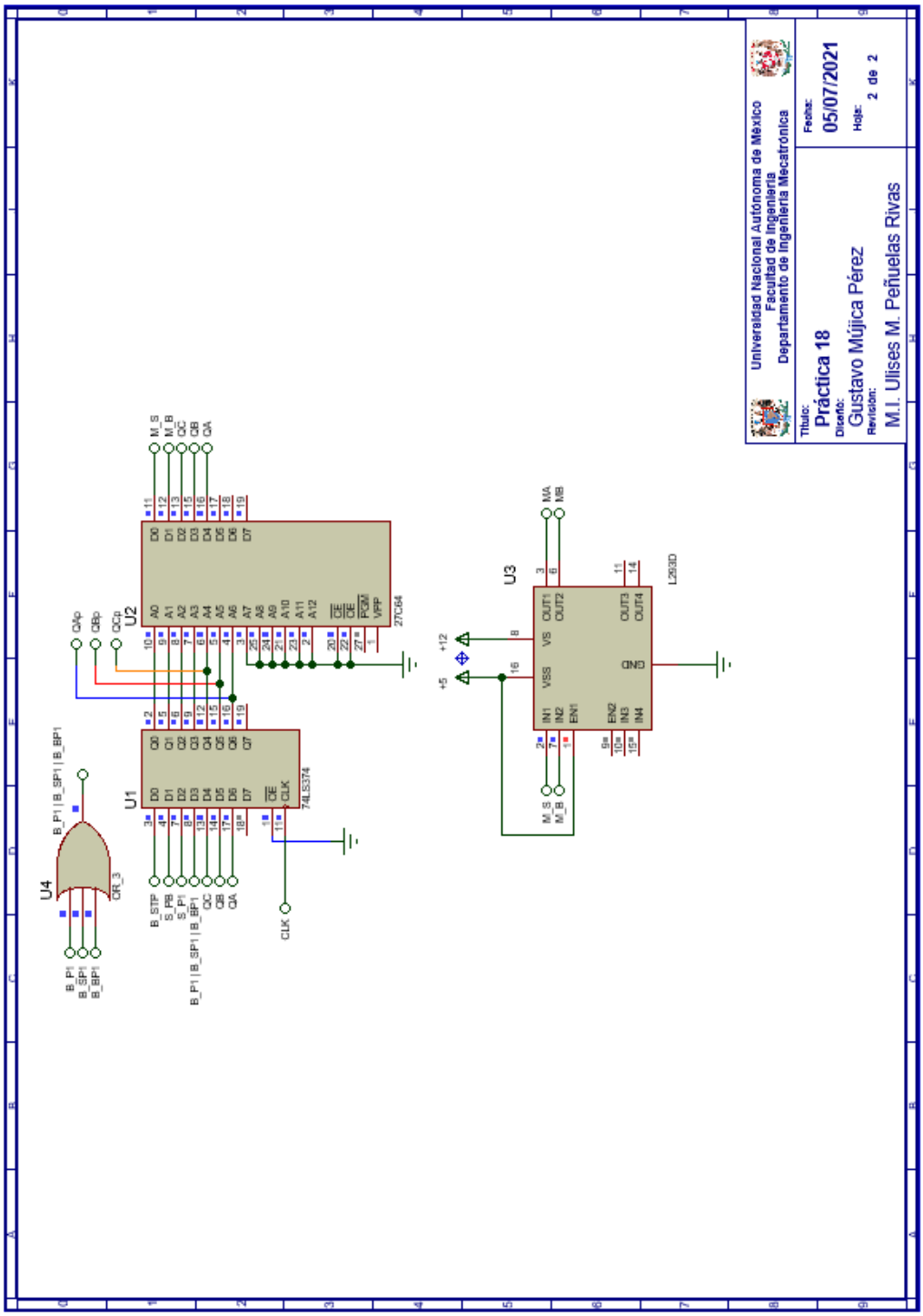


Figura P18.6 Esquema electrónico para el controlador de la Práctica 18 hoja 2/2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 19 Sistema de piso 2; diseño con memoria y direccionamiento entrada-estado

Introducción

A través de una maqueta, se simuló el funcionamiento de un elevador de un edificio de tres pisos por medio de una maqueta que está dividida en tres partes, las cuales son: cubo del elevador, puerta que abre y cierra, y el tablero de control.

El cubo tiene cuatro niveles, los cuales son: planta baja, piso 1, piso 2 y piso 3. Donde un carro sube y baja simulando el movimiento del elevador. Cada uno de los pisos tiene controles a un costado de la puerta para subir o bajar de piso, así como sensores de presencia que indican en donde está el carro.

El tablero de control simula el control interno del elevador. Tiene botones para: dirigirse a los 4 niveles del elevador, detener elevador, abrir y cerrar la puerta, así como un indicador del piso en el que se encuentra el elevador.

Para comprender mejor el funcionamiento del elevador este se dividió en cinco subsistemas, los cuales son: planta baja, piso 1, piso 2, piso 3 y el sistema de abrir/cerrar puerta.

En esta práctica se habla sobre el sistema de piso 2. Este sistema junto con el sistema de abrir/cerrar puerta, planta baja, piso 1 y piso 3 forman el sistema del elevador. (ver figura P19.1, figura P19.2).

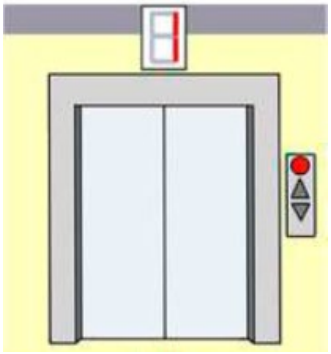


Figura P19.1 Piso 2 de la maqueta.

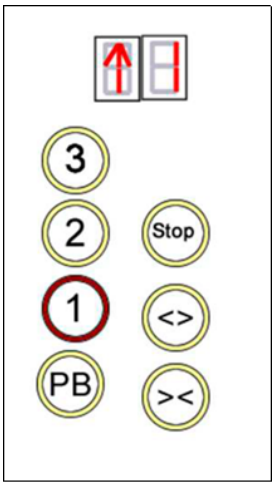


Figura P19.2 Tablero de control interno del cubo del elevador.

El diseño con memoria y direccionamiento entrada-estado restringe las cartas ASM a una sola entrada por estado. Una nueva porción de la palabra de memoria contiene una representación binaria de la entrada a probar en cada estado, esta parte es llamada “la parte de prueba”. Con esta representación binaria un selector de entrada elige una de las variables de entrada (ver figura P19.3) [1].

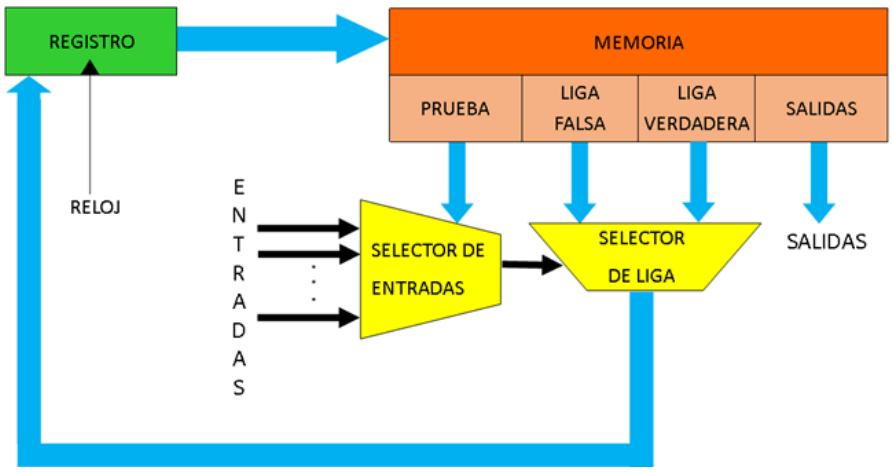


Figura P19.3 Arquitectura de un controlador con memoria y direccionamiento entrada-estado.

La parte de liga tiene dos estados siguientes, escogiéndose uno por el selector de liga, con base en la entrada seleccionada por la parte de prueba. Si el valor de la entrada seleccionada por el selector de entradas es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera (ver figura P19.4) [1].

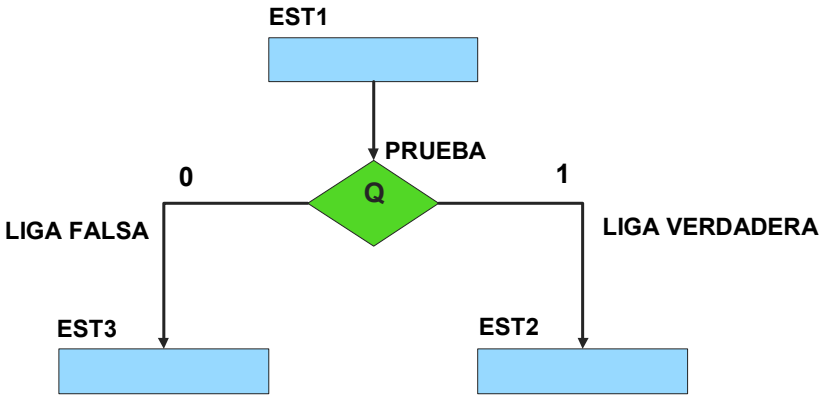


Figura P19.4 La liga falsa es el estado EST3, mientras que la liga verdadera es el estado EST2.

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará también una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista, se probará la variable auxiliar, la cual tiene un valor preestablecido de cero o uno [1].

Objetivo

Diseñar un controlador para el sistema de piso 2, por medio del método de diseño con memoria y direccionamiento entrada-estado.

Descripción

Primero se diseña una carta ASM para el sistema de piso 2 por medio del método de diseño con memoria y direccionamiento entrada-estado. Posteriormente se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P19.1 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema de piso 2 se necesitan las siguientes señales de entrada:

- un botón detecta la solicitud para subir a otro piso desde el piso 2 y otro para bajar desde el piso 2
- un botón dentro del elevador detecta la solicitud para ir hacia el piso 2
- un sensor indica que el elevador se encuentra en el piso 2 y otro indica que se encuentra en el piso 3
- un botón dentro del elevador detecta la solicitud para detener el elevador.

Como salida se requieren de las siguientes señales:

- una señal se activa para que el elevador baje hasta llegar al piso 2 y otra se activa para que suba hasta llegar al piso 2.

Tabla P19.1 Entradas y salidas para el sistema de piso 2.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
B_P2	A0 del PIC	I	Botón para ir al piso 2
B_SP2	A1 del PIC	I	Botón para subir desde el piso 2
B_BP2	A2 del PIC	I	Botón para bajar desde el piso 2
S_P2	A3 del PIC	I	Sensor de piso 2
S_P3	A4 del PIC	I	Sensor de piso 3
B_STP	A5 del PIC	I	Botón para detener el elevador
M_S	D0 de la memoria 2	O	Señal para subir el elevador
M_B	D1 de la memoria 2	O	Señal para bajar el elevador.

Notas de diseño

a) El botón de piso 2 y el botón de paro se encuentran en el tablero de control, el cual simula el control de los botones dentro del elevador
b) El botón para subir desde el piso 2 y el botón para bajar desde el piso 2 se encuentran ubicados a un costado de la entrada del elevador en el piso 2
c) Sólo se puede detener elevador cuando se encuentre en movimiento
d) El botón de paro debe mantenerse presionado para que no se mueva el elevador.

Reglas de funcionamiento

- B_P2: botón de piso 2
  1 = se oprimió el botón de piso 2
  0 = no se oprimió el botón de piso 2
- B_SP2: botón subir desde piso 2
  1 = se oprimió el botón de subir desde el piso 2
  0 = no se oprimió el botón de subir desde el piso 2
- B_BP2: botón bajar desde piso 2
  1 = se oprimió el botón de bajar desde el piso 2
  0 = no se oprimió el botón de bajar desde el piso 2
- S_P2: sensor de piso 2
  1 = el elevador está en el piso 2
  0 = no está el elevador en el piso 2
- S_P3: sensor de piso 3
  1 = el elevador está en el piso 3
  0 = no está el elevador en el piso 3
- B_STP: botón de paro
  1 = se oprimió el botón de paro
  0 = no se oprimió el botón de paro

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Si el valor de la entrada es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera. El algoritmo de la máquina de estados se puede ver en la figura P19.5.

Estado ‘0000’ – CONTROL

En el primer estado se está a la espera de que se oprima un botón para que el elevador se dirija al piso 2. Si se oprime el botón para subir (B_SP2) o el botón para bajar (B_BP2) desde el piso 2, o el botón de piso 2 (B_P2) del tablero, el sistema avanza al Estado ‘0001’ para detectar si el elevador se encuentra en el piso 2. De lo contrario, permanece en el Estado ‘0000’ en espera de que se oprima un botón.

Estado ‘0001’ – DETP2

En este estado se verifica si el elevador se encuentra en el piso 2. Si el sensor (S_P2), detecta que el elevador se encuentra en el piso 2, el sistema regresa al Estado ‘0000’ para iniciar el proceso nuevamente. De lo contrario, avanza al Estado ‘0010’ para detectar si el elevador se encuentra en el piso 3.

Estado ‘0010’ – DETP3

Si en este estado el sensor (S_P3), detecta al elevador en el piso 3, el sistema avanza al Estado ‘0011’ para bajar el elevador al piso 2 ya que se encuentra en el piso superior. De lo contrario, avanza al Estado ‘0110’ para subir el elevador al piso 2 debido a que se encuentra en alguno de los pisos inferiores.

Estado ‘0011’ – BAJAP2

En este estado se activa la señal (M_B), para que el elevador baje. Si el sensor (S_P2), detecta que el elevador se encuentra en el piso 2, el sistema regresa al Estado ‘0000’ para iniciar el proceso nuevamente. De lo contrario, avanza al Estado ‘0100’ para revisar el botón de paro del elevador.

Estado ‘0100’ – STPBP2

En este estado permanece activa la señal (M_B), para que el elevador baje. Si el botón (B_STP), no está presionado, el sistema regresa al Estado ‘0011’ para seguir bajando el elevador. De lo contrario, avanza al Estado ‘0101’ para detener el elevador.

Estado ‘0101’ – STOPBP2

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no está presionado, el sistema regresa al Estado ‘0011’ para seguir bajando el elevador. De lo contrario, el elevador permanece inmóvil en el Estado ’0101’.

Estado ‘0110’ – SUBEP2

En este estado se activa la señal (M_S) para que el elevador suba. Si el sensor (S_P2), detecta que el elevador se encuentra en el piso 2, el sistema regresa al Estado ‘0000’ para iniciar el proceso nuevamente. De lo contrario, avanza al Estado ‘0111’ para revisar el botón de paro del elevador.

Estado ‘0111’ – STPSP2

En este estado permanece activa la señal (M_S), para que el elevador suba. Si el botón (B_STP) no está presionado, el sistema regresa al Estado ‘0110’ para seguir subiendo el elevador. De lo contrario, avanza al Estado ‘1000’ para detener el elevador.

Estado ‘1000’ – STOPSP2

En este estado el elevador se encuentra detenido. Si el botón (B_STP), está presionado, el elevador permanece inmóvil en el Estado ’1000’. De lo contrario, regresa al Estado ‘0110’ para seguir subiendo el elevador.

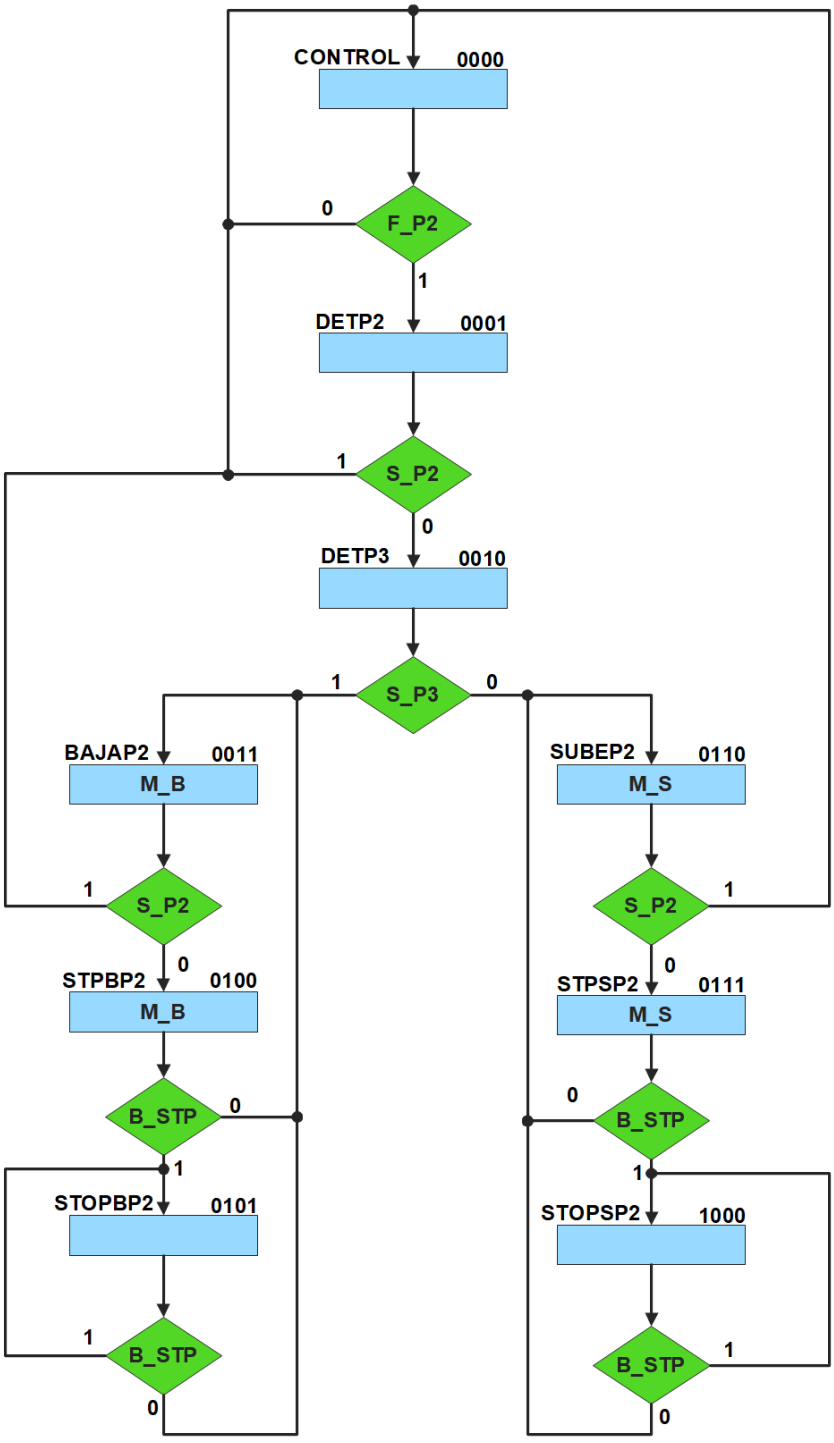


Figura P19.5 Carta ASM del sistema de piso 2.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P19.2).

Donde:

```
F_P2 = B_P2 | B_SP2 | B_BP2;
```

Tabla P19.2 Representación binaria de entradas para el sistema de piso 2.

Entrada	Prueba
F_P2	0 0
S_P2	0 1
S_P3	1 0
B_STP	1 1

Se debe llenar la tabla P19.3 con base en la información de la carta ASM de la figura P19.5, usando el método de direccionamiento de entrada-estado.

A continuación, se describe como llenar los campos de la memoria para el Estado ‘0000’.

En el Estado ‘0000’ se selecciona la entrada F_P2, por lo tanto, se coloca en el campo de prueba de la memoria su representación binaria, es decir, ‘00’. Si F_P2 es igual a cero, el estado siguiente es el Estado ‘0000’, su representación binaria ‘0000’ es colocada en el campo de la liga falsa. Si F_P2 es igual a uno, el estado siguiente es el Estado ‘0001’, su representación binaria ‘0001’ es colocada en el campo de la liga verdadera. En el Estado ‘0000’ no se activa ningún motor, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P19.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P19.3 Contenido de la memoria para el sistema de piso 2.

Dirección de memoria				Contenido de memoria													
Estado presente				Prueba		Liga Falsa				Liga Verdadera		Liga Verdadera		Salidas		Hex 1	Hex 2
QA	QB	QC	QD	I1	I0	LF3	LF2	LF1	LF0	LV3	LV2	LV1	LV0	M_B	M_S		
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	00	04
0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	48	00
0	0	1	0	1	0	0	1	1	0	0	0	1	1	0	0	98	0C
0	0	1	1	0	1	0	1	0	0	0	0	0	0	1	0	50	02
0	1	0	0	1	1	0	0	1	1	0	1	0	1	1	0	CD	06
0	1	0	1	1	1	0	0	1	1	0	1	0	1	0	0	CD	04
0	1	1	0	0	1	0	1	1	1	0	0	0	0	0	1	5C	01
0	1	1	1	1	1	0	1	1	0	1	0	0	0	0	1	DA	01
1	0	0	0	1	1	0	1	1	0	1	0	0	0	0	0	DA	00

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de cuatro líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P19.4). Se puede comprobar realizando el circuito de la figura P19.6.

Tabla P19.4 Tabla de funcionamiento del selector de entradas para el sistema de piso 2.

Prueba		SI
I1	I0	
0	0	F_P2
0	1	S_P2
1	0	S_P3
1	1	B_STP

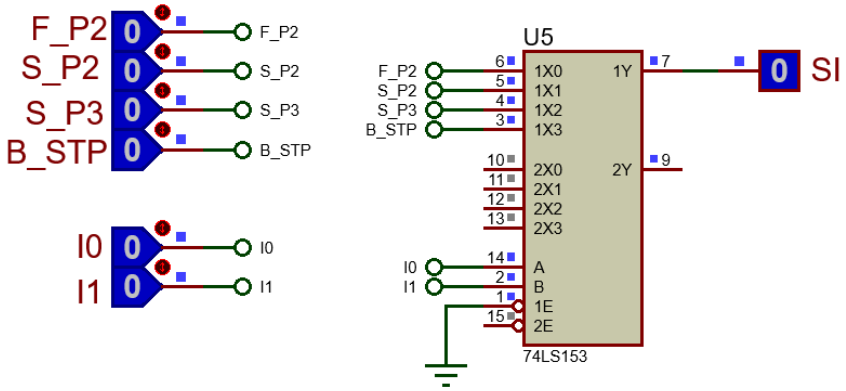


Figura P19.6 Multiplexor 74LS153 para el selector de entradas del sistema de piso 2.

La función booleana del selector de entradas queda:

```
SI = F_P2&!I1&!I0 | S_P2&!I1&I0 | S_P3&I1&!I0 | B_STP&I1&I0;
```

El selector de liga es un multiplexor cuádruple de dos líneas a una, éste es implementado por el PIC16F1939. Si el selector entradas es igual a ‘1’, se selecciona la información binaria de la liga verdadera. Si el selector entradas es igual a ‘0’, se selecciona la información binaria de la liga falsa. Se puede comprobar realizando el circuito de la figura P19.7.

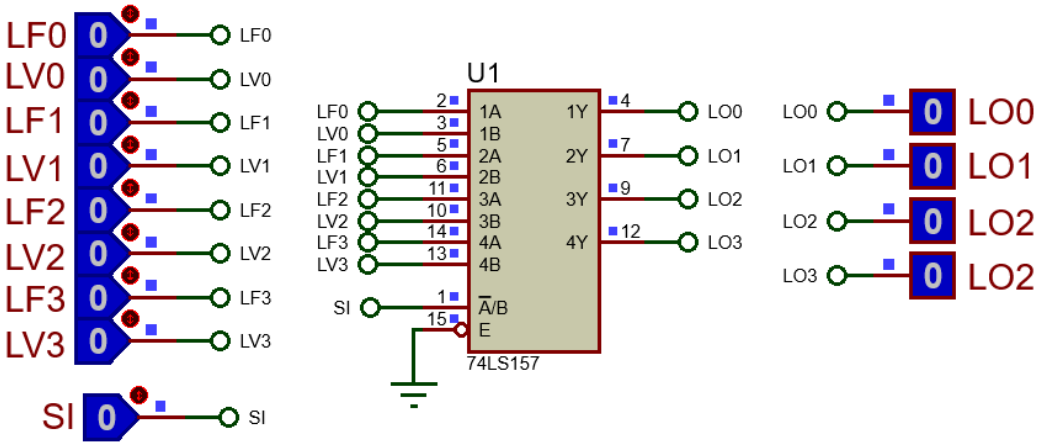


Figura P19.7 Multiplexor 74LS157 para el selector de liga para el sistema de piso 2.

Las funciones booleanas del selector de liga quedan:

```
LO0 = !SI&LF0 | SI&LV0;

LO1 = !SI&LF1 | SI&LV1;

LO2 = !SI&LF2 | SI&LV2;

LO3 = !SI&LF3 | SI&LV3;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P19.8,figura P19.9). Se carga en el controlador los archivos con extensión “HEX” de las memorias y los archivos “COF” o “HEX” del PIC16F1939.

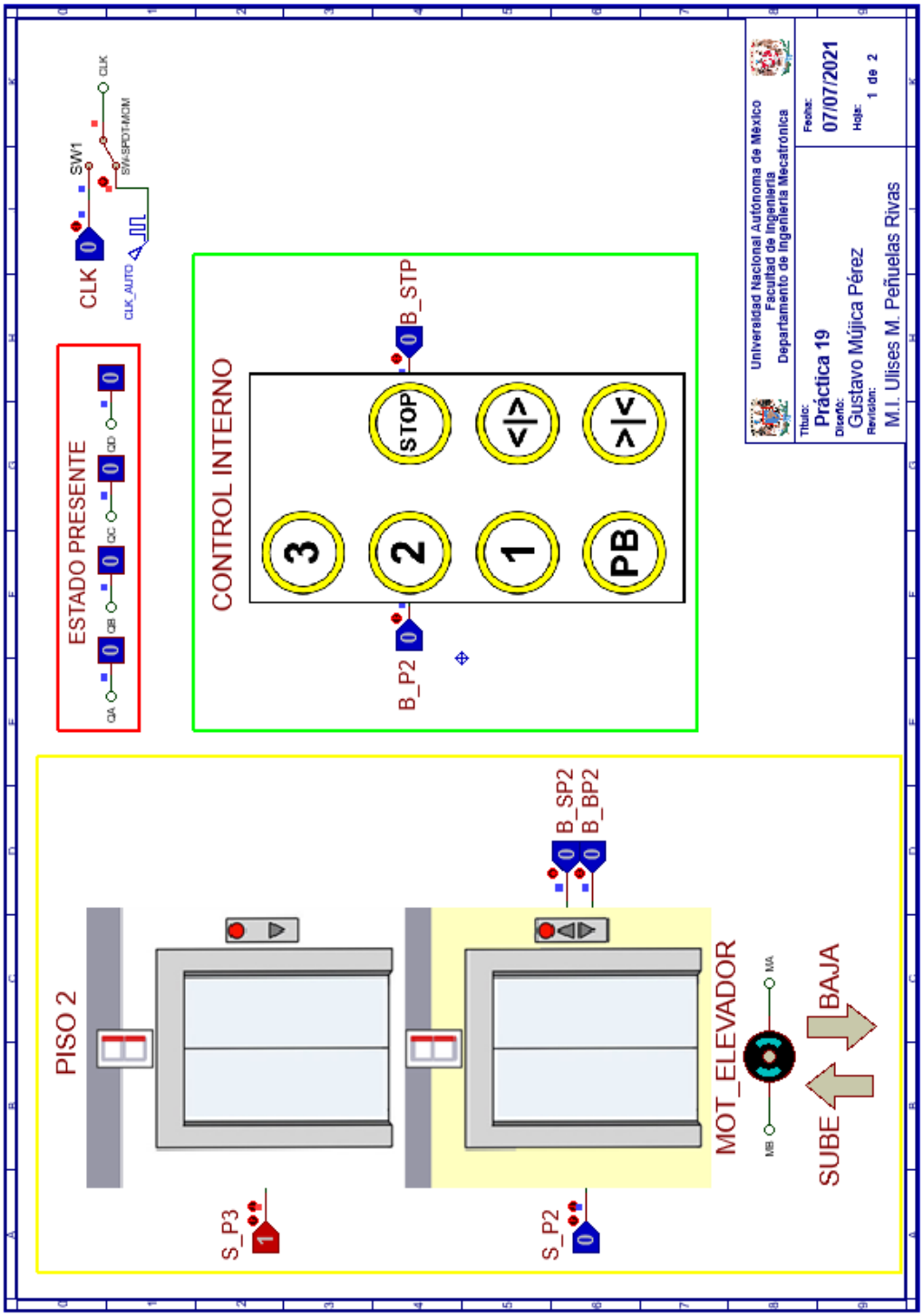


Figura P19.8 Interfaz hombre-máquina para el controlador de la Práctica 19 hoja 1/2.

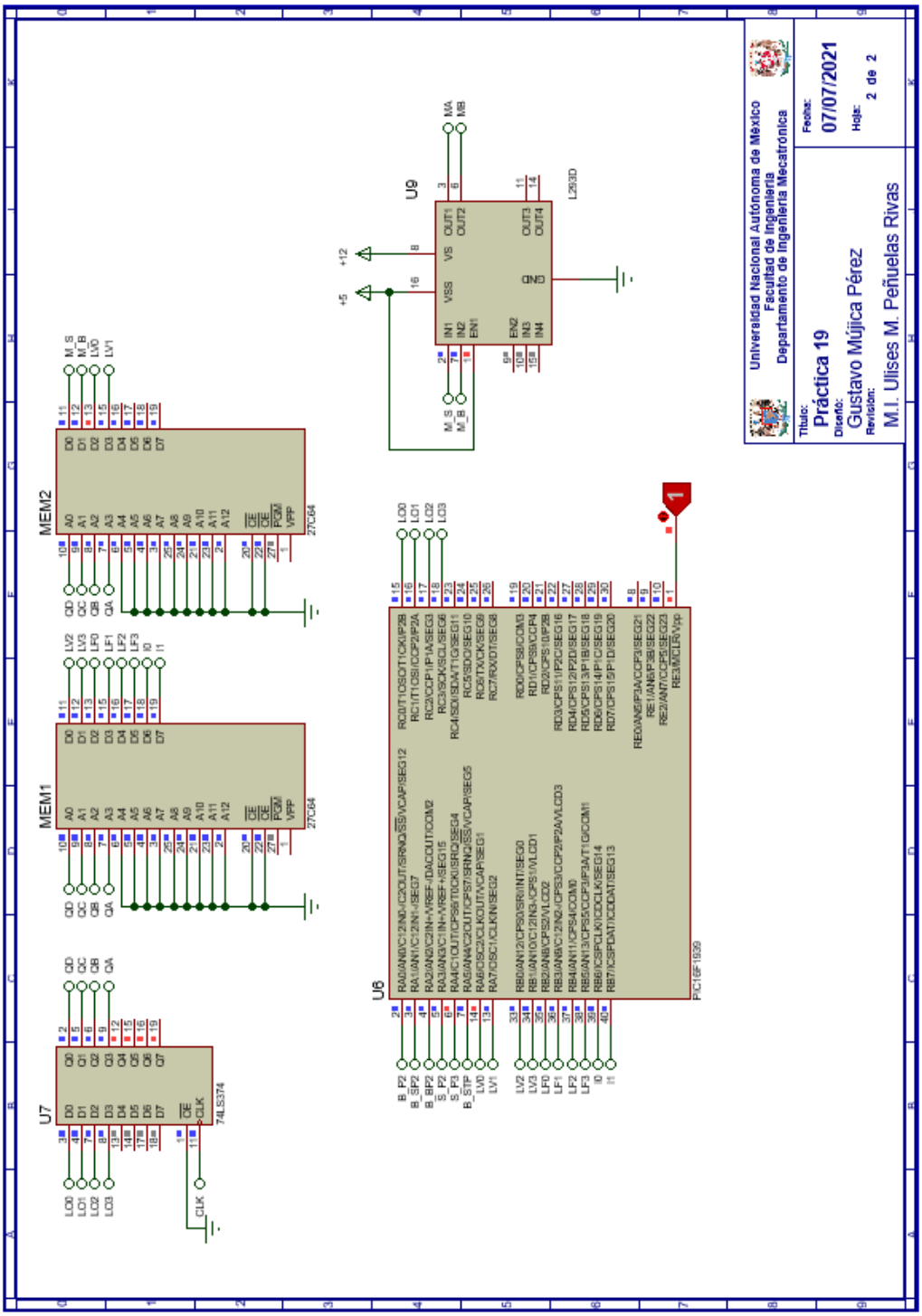


Figura P19.9 Esquema electrónico para el controlador de la Práctica 19 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P19.10, figura P19.11) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     //Carga biblioteca PLD.h

//***ENTRADAS***

#define B_P2  A0 //ENTRADA
#define B_SP2 A1 //ENTRADA
#define B_BP2 A2 //ENTRADA
#define S_P2  A3 //ENTRADA
#define S_P3  A4 //ENTRADA
#define B_STP A5 //ENTRADA

//LIGAS VERDADERAS
#define LV0 A6
#define LV1 A7
#define LV2 B0
#define LV3 B1

//LIGAS FALSAS
#define LF0 B2
#define LF1 B3
#define LF2 B4
#define LF3 B5

//PRUEBAS
#define I0 B6 //PRUEBA
#define I1 B7 //PRUEBA

//***SALIDAS***

#define LO0 C0
#define LO1 C1
#define LO2 C2
#define LO3 C3

//***VARIABLES INTERMEDIAS***

short SI; //SELECTOR DE ENTRADA
short F_P2; //Función

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD

   //LOOP INFINITO
   while(1)
   {
```

Figura P19.10 Código de la Práctica 19 parte 1.

```
47:     {
48:
49:      //Función
50:      F_P2= B_P2 | B_SP2 | B_BP2;
51:     //SELECTOR DE ENTRADAS
52:     SI = F_P2&!I1&!I0 | S_P2&!I1&I0 | S_P3&I1&!I0 | B_STP&I1&I0;
53:
54:     //SELECTOR DE LIGA
55:     LO0= !SI&LF0 | SI&LV0;
56:     LO1= !SI&LF1 | SI&LV1;
57:     LO2= !SI&LF2 | SI&LV2;
58:     LO3= !SI&LF3 | SI&LV3;
59:     }
60: }
```

Figura P19.11 Código de la Práctica 19 parte 2.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 20 Sistema de piso 3; diseño con memoria y direccionamiento entrada-estado

Introducción

A través de una maqueta, se simuló el funcionamiento de un elevador de un edificio de tres pisos por medio de una maqueta que está dividida en tres partes, las cuales son: cubo del elevador, puerta que abre y cierra, y el tablero de control.

El cubo tiene cuatro niveles, los cuales son: planta baja, piso 1, piso 2 y piso 3. Donde un carro sube y baja simulando el movimiento del elevador. Cada uno de los pisos tiene controles a un costado de la puerta para subir o bajar de piso, así como sensores de presencia que indican en donde está el carro.

El tablero de control simula el control interno del elevador. Tiene botones para: dirigirse a los 4 niveles del elevador, detener elevador, abrir y cerrar la puerta, así como un indicador del piso en el que se encuentra el elevador.

Para comprender mejor el funcionamiento del elevador este se dividió en cinco subsistemas, los cuales son: planta baja, piso 1, piso 2, piso 3 y el sistema de abrir/cerrar puerta.

En esta práctica se habla sobre el sistema de piso 3. Este sistema junto con el sistema de abrir/cerrar puerta, planta baja, piso 1 y piso 2 forman el sistema del elevador (ver figura P20.1, figura P20.2).

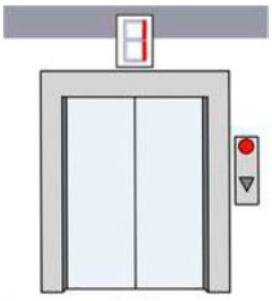


Figura P20.1 Piso 3 de la maqueta.

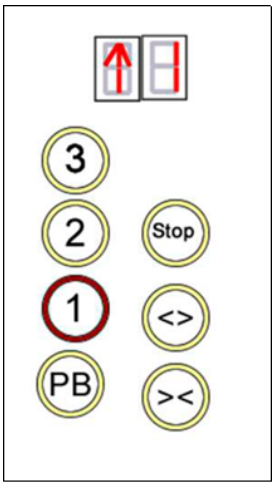


Figura P20.2 Tablero de control interno del cubo del elevador.

El diseño con memoria y direccionamiento entrada-estado restringe las cartas ASM a una sola entrada por estado. Una nueva porción de la palabra de memoria contiene una representación binaria de la entrada a probar en cada estado, esta parte es llamada “la parte de prueba”. Con esta representación binaria un selector de entrada elige una de las variables de entrada (ver figura P20.3) [1].

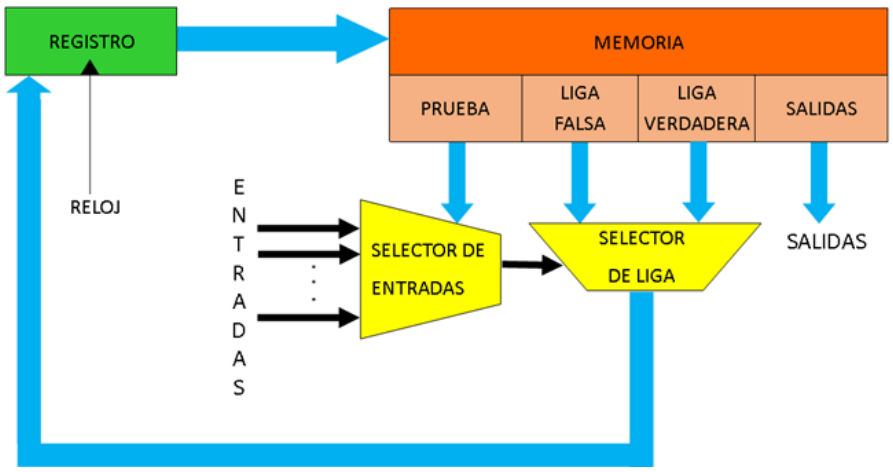


Figura P20.3 Arquitectura de un controlador con memoria y direccionamiento entrada-estado.

La parte de liga tiene dos estados siguientes, escogiéndose uno por el selector de liga, con base en la entrada seleccionada por la parte de prueba. Si el valor de la entrada seleccionada por el selector de entradas es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera (ver figura P20.4) [1].

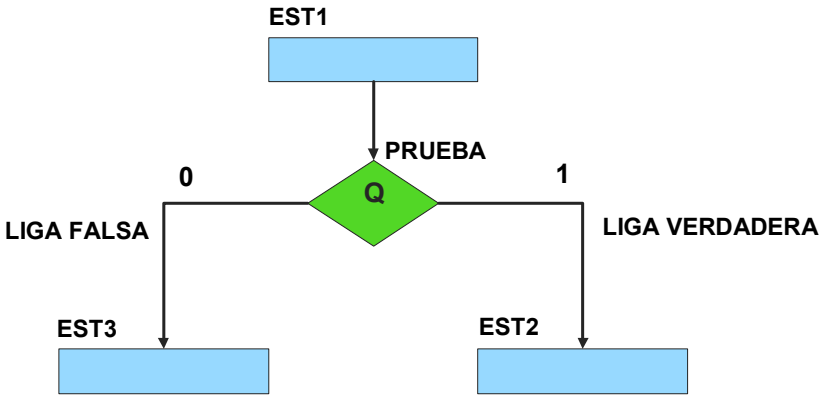


Figura P20.4 La liga falsa es el estado EST3, mientras que la liga verdadera es el estado EST2.

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará también una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista, se probará la variable auxiliar, la cual tiene un valor preestablecido de cero o uno [1].

Objetivo

Diseñar un controlador para el sistema de piso 3, por medio del método de diseño con memoria y direccionamiento entrada-estado.

Descripción

Primero se diseña una carta ASM para el sistema de piso 3 por medio del método de diseño con memoria y direccionamiento entrada-estado. Posteriormente se propondrá una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P20.1 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema de piso 3 se necesitan señales de entrada:

- un botón detecta la solicitud para subir a otro piso desde el piso 3
- un botón dentro del elevador detecta la solicitud para ir hacia el piso 3
- un sensor indica que el elevador se encuentra en el piso 3
- un botón dentro del elevador detecta la solicitud para detener el elevador.

Como salida se requiere de las siguientes señales:

- una señal se activa para que el elevador suba hasta llegar al piso 3.

Tabla P20.1 Entradas y salidas para el sistema de piso 3.

Identificador	Ubicación	Tipo	Descripción
CLK	CLK del registro	I	Señal de reloj
B_P3	A0 del PIC	I	Botón para ir al piso 3
B_BP3	A1 del PIC	I	Botón para bajar desde el piso 3
S_P3	A2 del PIC	I	Sensor de piso 3
B_STP	A3 del PIC	I	Botón para detener el elevador
M_S	D0 de la memoria 2	O	Señal para subir el elevador.

Notas de diseño

a) El botón de piso 3 y el botón de paro se encuentran en el tablero de control, el cual simula el control de los botones dentro del elevador
b) El botón para bajar desde el piso 3 se encuentra ubicado a un costado de la entrada del elevador en el piso 3
c) Sólo se puede detener elevador cuando se encuentre en movimiento
d) El botón de paro debe mantenerse oprimido para que no se mueva el elevador.

Reglas de funcionamiento

- B_P3: botón de piso 3
  1 = se oprimió el botón de piso 3
  0 = no se oprimió el botón de piso 3
- B_BP3: botón bajar desde piso 3
  1 = se oprimió el botón de bajar desde el piso 3
  0 = no se oprimió el botón de bajar desde el piso 3
- S_P3: sensor de piso 3
  1 = el elevador está en el piso 3
  0 = no está el elevador en el piso 3
- B_STP: botón de paro
  1 = se oprimió el botón de paro
  0 = no se oprimió el botón de paro

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Si el valor de la entrada es igual a cero, entonces el selector de liga elegirá la liga falsa, en caso contrario se elegirá la liga verdadera. El algoritmo de la máquina de estados se puede ver en la figura P20.5.

Estado '000' – CONTROL

En el primer estado se está a la espera de que se oprima un botón para que el elevador se dirija al piso 3. Si se presiona el botón de piso 3 del tablero o se oprime el botón para bajar a otro piso desde el piso 3, el sistema avanza al Estado '001' para detectar si el elevador se encuentra en el piso 3. De lo contrario, permanece en el Estado '000' en espera de que se oprima un botón.

Estado '001' – DETP3

En este estado se verifica si el elevador se encuentra en el piso 3. Si el sensor (S_P3), detecta que el elevador se encuentra en el piso 3, el sistema regresa al Estado '000' para iniciar el proceso nuevamente. De lo contrario, avanza al Estado '010' para subir el elevador ya que se encuentra en alguno de los pisos inferiores.

Estado '010' – SUBEP3

En este estado se activa la señal (M_S) para que el elevador suba al piso 3. Si el sensor (S_P3), detecta que el elevador se encuentra en el piso 3, el sistema regresa al Estado '000' para iniciar el proceso nuevamente. De lo contrario, avanza al Estado '011' para revisar el botón de paro del elevador.

Estado '011' – STPSP3

En este estado permanece activa la señal (M_S) para que el elevador suba. Si el botón (B_STP), no está oprimido, el sistema regresa al Estado '010' para seguir subiendo el elevador. De lo contrario, avanza al Estado '100' para detener el elevador.

Estado '100' – STOPSP3

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no está oprimido, el sistema regresa al Estado '010' para seguir subiendo elevador. De lo contrario, el elevador permanece inmóvil en el Estado '100'.

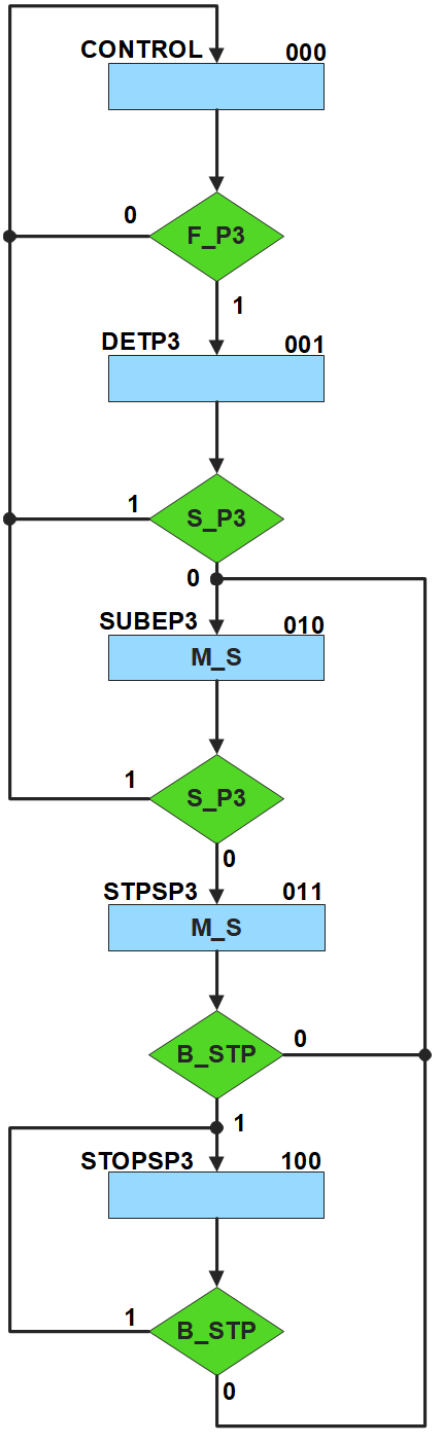


Figura P20.5 Carta ASM del sistema de piso 3.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P20.2).

Donde:

```
F_P3 = B_P3 | B_BP3;
```

Tabla P20.2 Representación binaria de entradas para el sistema de piso 3.

Entrada	Prueba
F_P3	0 0
S_P3	0 1
B_STP	1 0

Se debe llenar la tabla P20.3 con base en la información de la carta ASM de la figura P20.5, usando el método de diseño con memoria y direccionamiento de entrada-estado.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘000’.

En el Estado ‘000’ se selecciona la entrada F_P3, por lo tanto, se coloca en el campo de prueba de la memoria su representación binaria, es decir, ‘00’. F_P3 es igual a cero, el estado siguiente es el Estado ‘000’, su representación binaria ‘000’ es colocada en el campo de la liga falsa. Si F_P3 es igual a uno, el estado siguiente es el Estado ‘001’, su representación binaria ‘001’ es colocada en el campo de la liga verdadera. En el Estado ‘000’ no se activa ningún motor, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P20.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P20.3 Contenido de la memoria para el sistema de piso 3.

Dirección de memoria			Contenido de memoria										
Estado presente			Prueba		Liga Falsa			Liga Verdadera			Salida	Hex 1	Hex 2
QA	QB	QC	I1	I0	LF2	LF1	LF0	LV2	LV1	LV0	M_S		
0	0	0	0	0	0	0	0	0	0	1	0	01	00
0	0	1	0	1	0	1	0	0	0	0	0	50	00
0	1	0	0	1	0	1	1	0	0	0	1	58	01
0	1	1	1	0	0	1	0	1	0	0	1	94	01
1	0	0	1	0	0	1	0	1	0	0	0	94	00

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de cuatro líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P20.4). Se puede comprobar realizando el circuito de la figura P20.6.

Tabla P20.4 Tabla de funcionamiento del selector de entradas para el sistema de piso 3.

Prueba		SI
I1	I0	
0	0	F_P3
0	1	S_P3
1	0	B_STP

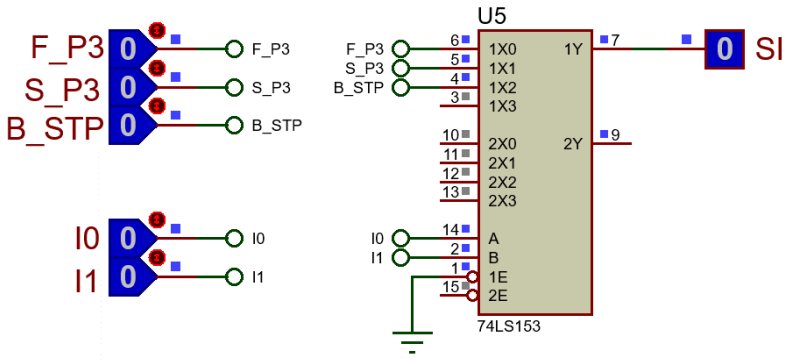


Figura P20.6 Multiplexor 74LS153 para el selector de entradas del sistema de piso 3.

La función booleana del selector de entradas queda:

```
SI = F_P3&!I1&!I0 | S_P3&!I1&I0 | B_STP&I1&!I0;
```

El selector de liga es un multiplexor triple de dos líneas a una, éste es implementado por el PIC16F1939. Si el selector entradas es igual a ‘1’, se selecciona la información binaria de la liga verdadera. Si el selector entradas es igual a ‘0’, se selecciona la información binaria de la liga falsa. Se puede comprobar realizando el circuito de la figura P20.7.

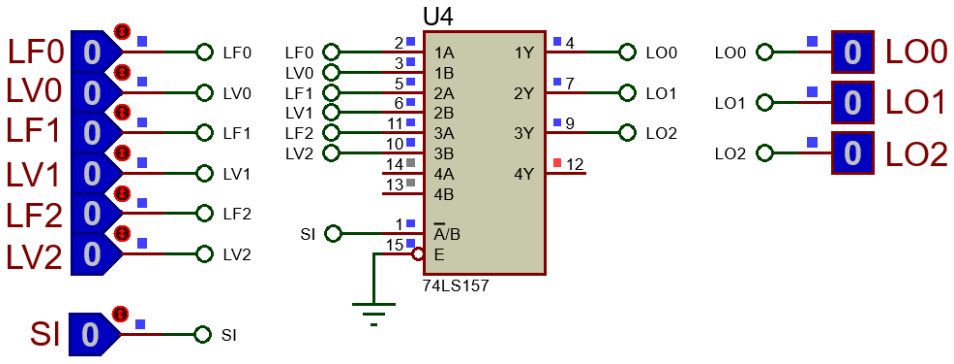


Figura P20.7 Multiplexor 74LS157 para el selector de liga del sistema de piso 3.

Las funciones booleanas del selector de liga quedan:

```
LO0 = !SI&LF0 | SI&LV0;

LO1 = !SI&LF1 | SI&LV1;

LO2 = !SI&LF2 | SI&LV2;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P20.8,figura P20.9). Se carga en el controlador los archivos con extensión “HEX” de las memorias y los archivos “COF” o “HEX” del PIC16F1939.

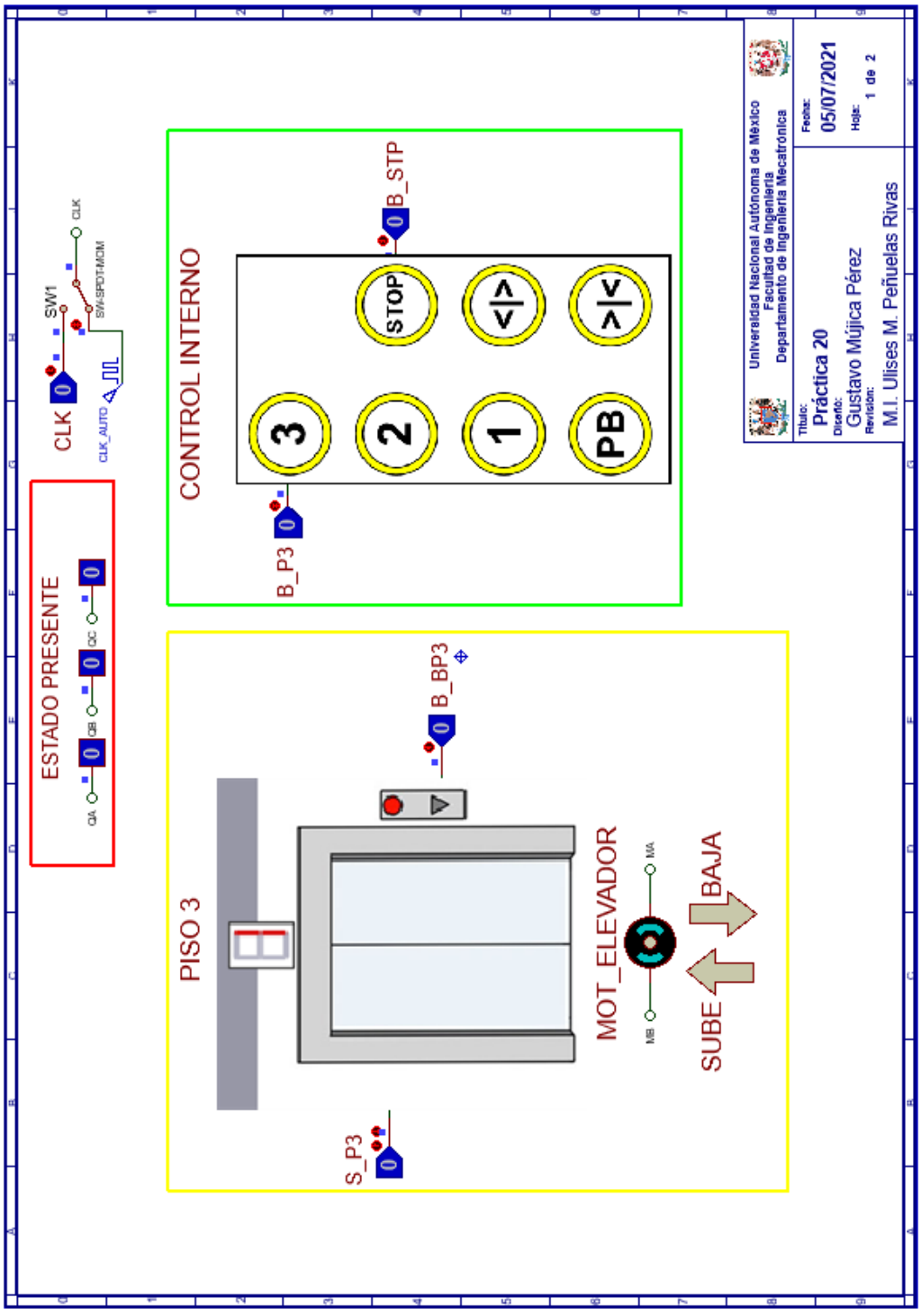


Figura P20.8 Interfaz hombre-máquina para el controlador de la Práctica 20 hoja 1/2.

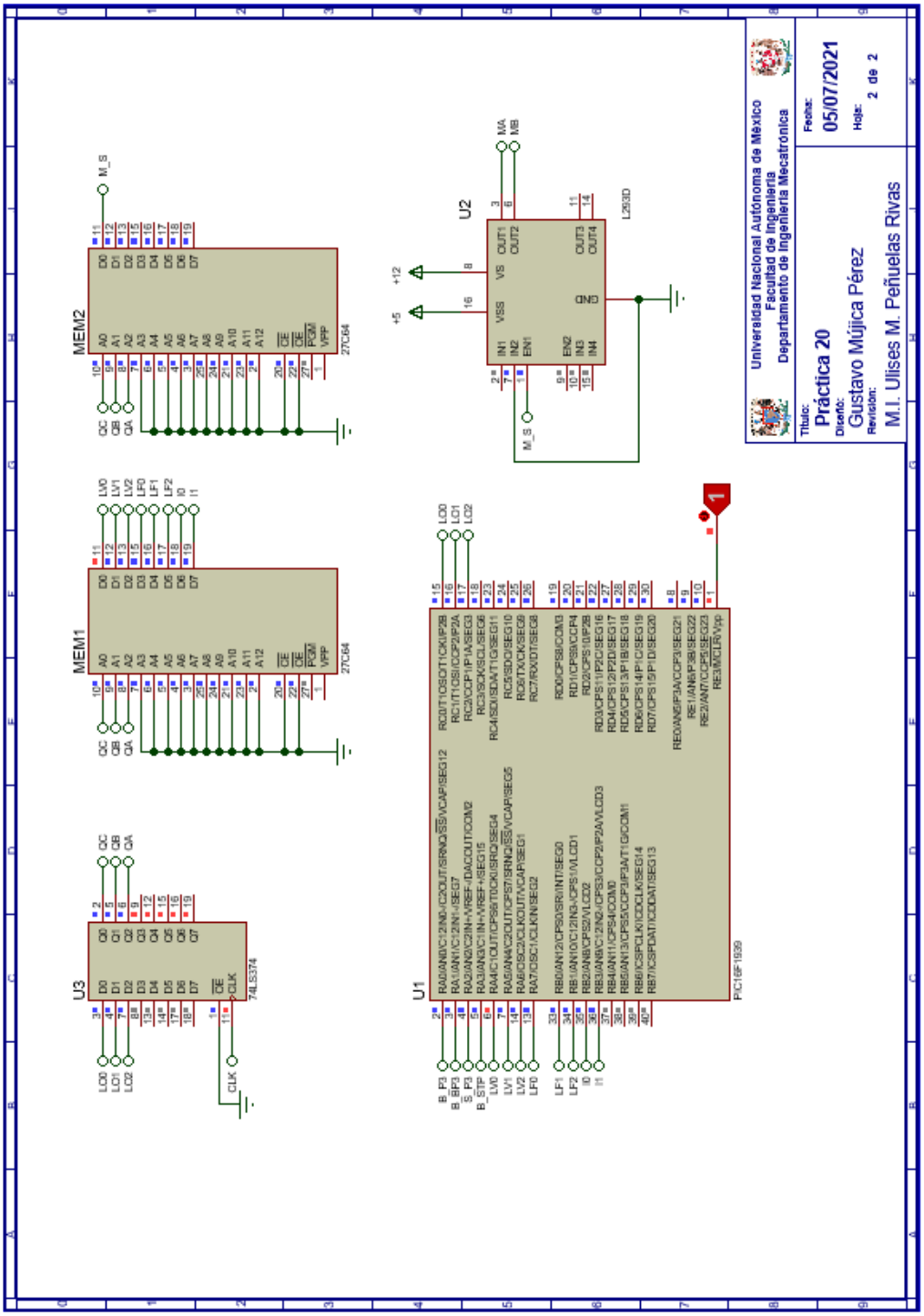


Figura P20.9 Esquema electrónico para el controlador de la Práctica 20 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P20.10) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     //Carga biblioteca PLD.h

//***ENTRADAS***
#define B_P3  A0 //ENTRADA
#define B_BP3 A1 //ENTRADA
#define S_P3  A2 //ENTRADA
#define B_STP A3 //ENTRADA
//LIGAS VERDADERAS
#define LV0 A4
#define LV1 A5
#define LV2 A6
//LIGAS FALSAS
#define LF0 A7
#define LF1 B0
#define LF2 B1
//PRUEBAS
#define I0 B2 //PRUEBA
#define I1 B3 //PRUEBA

//***SALIDAS***
#define LO0 C0
#define LO1 C1
#define LO2 C2

//***VARIABLES INTERMEDIAS***

short SI; //SELECTOR DE ENTRADA
short F_P3; //Función

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD

//LOOP INFINITO
   while(1)
   {
   //Función
   F_P3= B_P3 | B_BP3;
   //SELECTOR DE ENTRADAS
   SI= F_P3&!I1&!I0 | S_P3&!I1&I0 | B_STP&I1&!I0;

   //SELECTOR DE LIGA
   LO0= !SI&LF0 | SI&LV0;
   LO1= !SI&LF1 | SI&LV1;
   LO2= !SI&LF2 | SI&LV2;
   }
}
```

Figura P20.10 Código de la Práctica 20.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 21 Sistema de puerta del elevador; diseño con memoria y direccionamiento implícito

Introducción

A través de una maqueta, se simuló el funcionamiento de un elevador de un edificio de tres pisos por medio de una maqueta que está dividida en tres partes, las cuales son: cubo del elevador, puerta que abre y cierra, y el tablero de control.

El cubo tiene cuatro niveles, los cuales son: planta baja, piso 1, piso 2 y piso 3. Donde un carro sube y baja simulando el movimiento del elevador. Cada uno de los pisos tiene controles a un costado de la puerta para subir o bajar de piso, así como sensores de presencia que indican en donde está el carro.

La maqueta cuenta con una caja estática que sirve de monitor de estados para visualizar el abrir y cerrar de la puerta. La puerta se abre por medio de un motor cuando el carro del elevador llegue al piso seleccionado. También se debe abrir cuando se presionen los botones de abrir o cerrar puerta mientras el carro este estático en un piso. Además, si se obstruye el paso de la puerta mientras la puerta se esté cerrando, se debe abrir inmediatamente.

El tablero de control simula el control interno del elevador. Tiene botones para: dirigirse a los 4 niveles del elevador, detener elevador, abrir y cerrar la puerta, así como un indicador del piso en el que se encuentra el elevador.

Para comprender mejor el funcionamiento del elevador este se dividió en cinco subsistemas, los cuales son: planta baja, piso 1, piso 2, piso 3 y el sistema de abrir/cerrar puerta.

En esta práctica se habla sobre el sistema de abrir/cerrar puerta. Este sistema junto con el sistema de planta baja, piso 1, piso 2 y piso 3 forman el sistema del elevador (ver figura P21.1, figura P21.2).

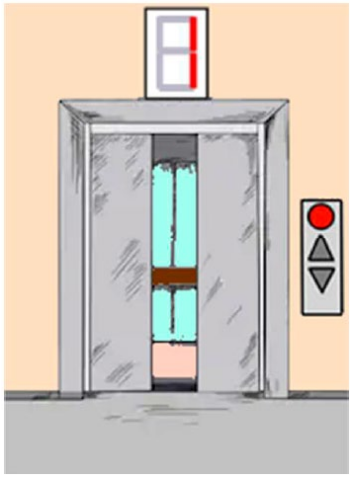


Figura P21.1 Puerta que abre y cierra.

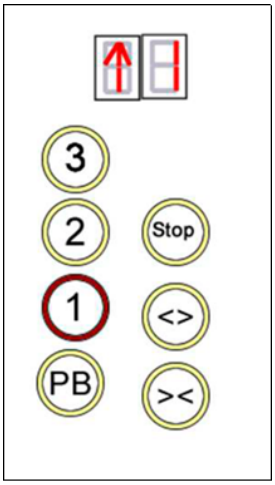


Figura P21.2 Tablero de control interno del cubo del elevador.

El diseño con memorias y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura P21.3). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura P21.4 muestra la arquitectura de este método.

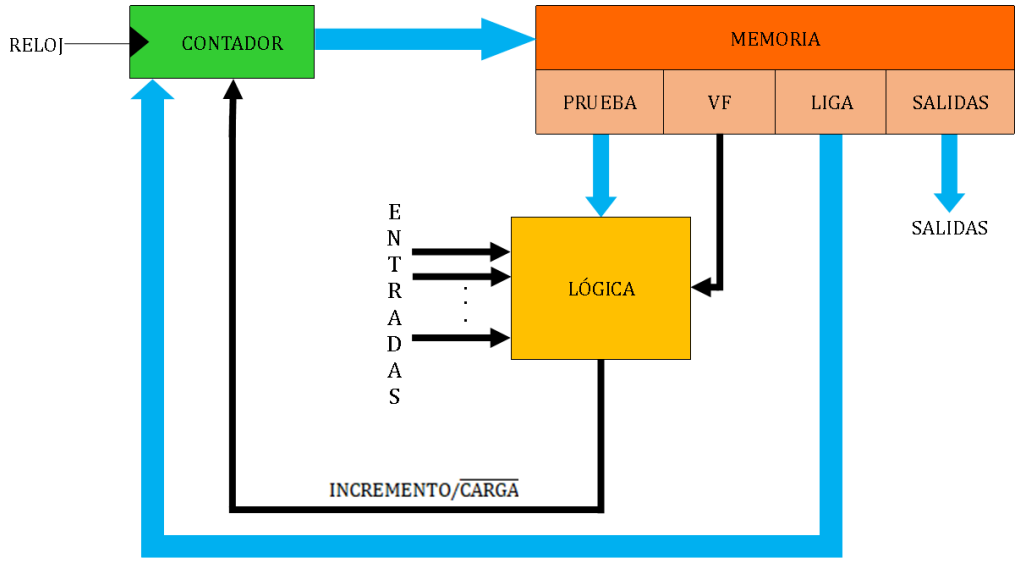


Figura P21.3 Arquitectura de un controlador con memoria y direccionamiento implícito.

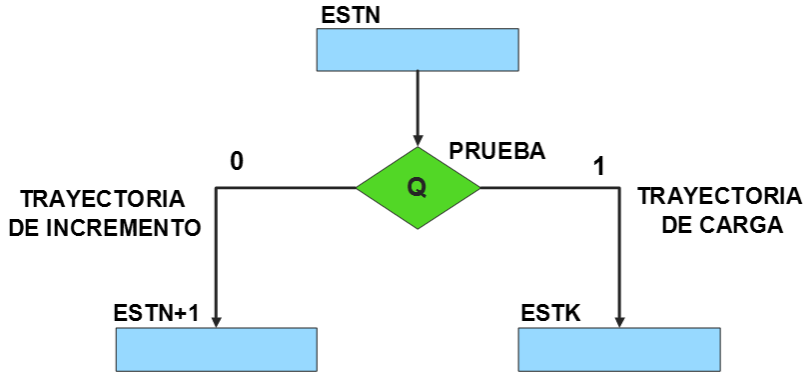


Figura P21.4 Trayectoria de incremento y carga.

La tabla P21.1 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que valor de entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un '1' se hace un incremento, cuando el resultado de la función da como resultado un '0' se hace una carga.

Tabla P21.1 Relación entre VF, variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incrementa o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es ‘0’, se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es ‘1’, se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura P21.5).

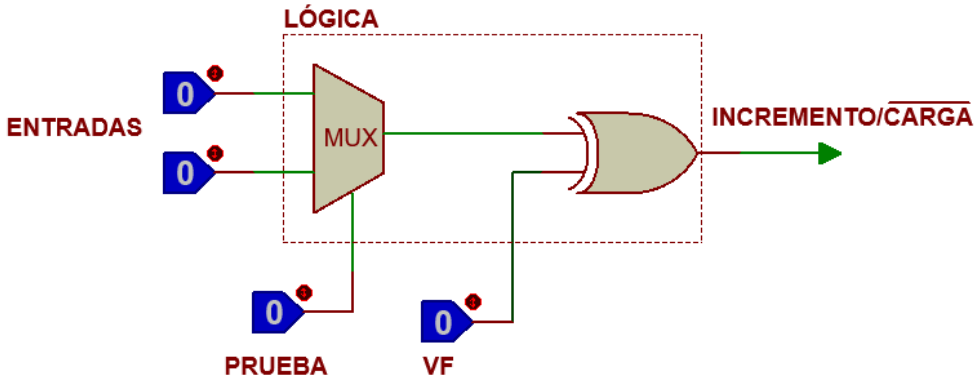


Figura P21.5 Bloque de lógica de incremento/carga para el direccionamiento implícito [1].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

Objetivo

Diseñar un controlador para el sistema de puerta del elevador, por medio del método de diseño con memoria y direccionamiento implícito.

Descripción

Se diseña una carta ASM para el sistema de puerta del elevador por medio del método de diseño con memoria y direccionamiento implícito. Posteriormente se propone una solución para el temporizador de la puerta del elevador. Después se propone una solución para implementar el sistema.

Tabla de entradas y salidas

En la tabla P21.2 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema de puerta del elevador se necesitan las siguientes señales de entrada:

- un sensor detecta si la puerta está abierta completamente y otro si está cerrada en su totalidad
- una señal indica que han transcurrido 5 segundos desde que se abrió la puerta completamente
- un sensor verifica si hay una obstrucción en la puerta
- una señal indica si se oprimió el botón para cerrar la puerta y otra para el botón que abre la puerta.

Como salida se requiere la siguiente señal:

- una señal se activa para abrir la puerta y otra para que se cierre.

Tabla P21.2 Entradas y salidas para el sistema de puerta del elevador.

Identificador	Ubicación	Tipo	Descripción
S_PA	A0 del PIC	I	Sensor de puerta abierta
S_PC	A1 del PIC	I	Sensor de puerta cerrada
T5	Interna	I	Señal que indica cuándo han transcurrido 5 segundos
S_OBS	A2 del PIC	I	Sensor de obstrucción
B_AP	A3 del PIC	I	Botón para abrir la puerta
B_CP	A4 del PIC	I	Botón para cerrar la puerta
M_A	D1 de la memoria 2	O	Señal para que se abra la puerta
M_C	D0 de la memoria 2	O	Señal para que se cierre la puerta.

Notas de diseño

a) El temporizador empieza a contar automáticamente sin necesidad de una señal de salida
b) El botón para abrir y cerrar la puerta se encuentra en el tablero de control, el cual simula el control de los botones dentro del elevador
c) Sólo se puede abrir y cerrar la puerta cuando el elevador no esté en movimiento.

Reglas de funcionamiento

- S_PA: sensor de puerta abierta
  1 = detecta puerta completamente abierta
  0 = no detecta puerta completamente abierta
- S_PC: sensor de puerta cerrada
  1 = detecta puerta completamente cerrada
  0 = no detecta puerta completamente cerrada
- T5: señal de 5 segundos
  1 = han transcurrido 5 segundos
  0 = no han transcurrido 5 segundos

- S_OBS: sensor de obstrucción
  1 = se está obstruyendo la puerta
  0 = no se está obstruyendo la puerta
- B_AP: botón para abrir puerta
  1 = se oprimió el botón para abrir puerta
  0 = no se oprimió el botón para abrir puerta
- B_CP: botón para cerrar puerta
  1 = se oprimió el botón para cerrar puerta
  0 = no se oprimió el botón para cerrar puerta.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Se hace un incremento cuando el valor del estado siguiente aumenta en una unidad, de lo contrario se hace una carga (ver figura P21.4). El algoritmo de la máquina de estados se puede ver en la figura P21.6.

Estado ‘0000’ – ABRE

En el primer estado se activa la señal (M_A), para que se abra la puerta, permitiendo el ingreso de personas. Cuando el sensor (S_PA), detecta que la puerta se ha abierto completamente, el sistema avanza al Estado ‘0001’ para esperar a que ingresen personas al elevador. De lo contrario, permanece en el Estado ‘0000’ abriendo la puerta.

Estado ‘0001’ – ESPERA

En este estado el temporizador empieza a contar o continúa contando 5 segundos sin necesidad de que sea activado por medio de una señal. La puerta se encuentra abierta, en espera de que ingresen las personas al elevador.

Cuando se detecta la señal del temporizador (T5), debido a que pasaron 5 segundos, el sistema avanza al Estado ’0010’ para cerrar la puerta. De lo contrario, avanza al Estado ‘0101’ para revisar el sensor de obstrucción.

Estado ‘0010’ – CIERRA

En este estado deja de contar el temporizador y se reinicia. Se activa la señal (M_C), para que se cierre la puerta. Si el sensor de puerta cerrada (S_PC), detecta que la puerta se ha cerrado completamente, el sistema regresa al Estado ‘0000’ para abrir la puerta nuevamente. De lo contrario, avanza al Estado ‘0011’ para revisar el sensor de obstrucción y el botón para abrir la puerta.

Estado ‘0011’ – BOTA

En este estado permanece activa la señal (M_C), para que la puerta siga cerrándose. Si el sensor (S_OBS), no detecta un obstáculo en el paso de la puerta y no está presionado el botón (B_AP) para abrir la puerta, el sistema regresa al Estado ‘0010’ para continuar cerrando la puerta. De lo contrario, avanza al Estado ‘0100’ para que posteriormente se pueda hacer una carga al Estado ‘0000’.

Estado ‘0100’ – AUX1

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘0000’, haciendo una carga y así iniciar el proceso nuevamente.

Estado ‘0101’ – OBST

En este estado el temporizador continúa contando 5 segundos y la puerta permanece abierta, en espera de que ingresen las personas al elevador. Cuando el sensor (S_OBS) detecta algo que obstruye el paso de la puerta, el sistema avanza al Estado ‘1000’ para reiniciar el temporizador. De lo contrario, avanza al Estado ‘0110’ para revisar si se oprimió el botón para cerrar la puerta.

Estado ‘0110’ – BOTC

En este estado, el temporizador continúa contando 5 segundos y la puerta permanece abierta en espera de que ingresen las personas al elevador.

Cuando se detecta que se oprimió el botón (B_CP), el sistema regresa al Estado ’0010’ para cerrar la puerta. De lo contrario, avanza al Estado ’0111’ para seguir contando 5 segundos en espera de que ingresen todas las personas al elevador.

Estado ‘0111’ – AUX2

En este estado el temporizador continúa contando 5 segundos y la puerta permanece abierta, en espera de que ingresen las personas al elevador. En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘0001’ para que el temporizador siga contando 5 segundos.

Estado ‘1000’ – RESET

En este estado se reinicia la cuenta del temporizador debido a que se detectó una obstrucción. En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada regresar al Estado ‘0001’ para que el temporizador se reinicie y empiece a contar 5 segundos nuevamente.

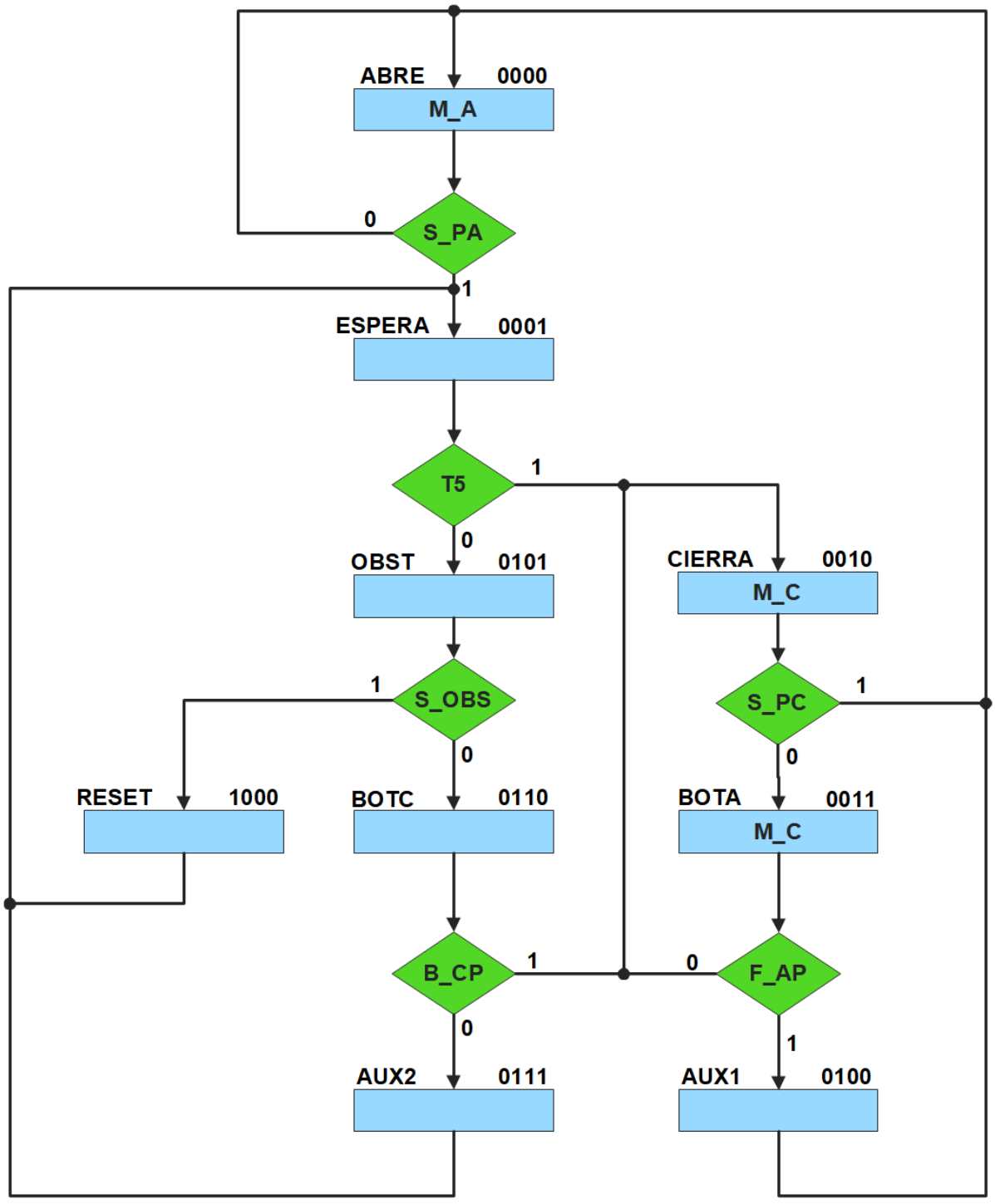


Figura P21.6 Carta ASM del sistema de puerta del elevador.

Solución

Se debe asignar una representación binaria a cada variable de entrada (ver tabla P21.3),

Donde:

```
F_AP = S_OBS | B_AP;
```

Tabla P21.3 Representación binaria de entradas para el sistema de puerta del elevador.

Entrada	Prueba
AUX	0 0 0
S_PA	0 0 1
S_PC	0 1 0
T5	0 1 1
S_OBS	1 0 0
F_AP	1 0 1
B_CP	1 1 0

Se debe llenar la tabla P21.4 con base en la información de la carta ASM de la figura P21.6, usando el método de diseño con memoria y direccionamiento implícito.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘0000’.

En el Estado ‘0000’ se selecciona la entrada S_PA, por lo tanto, se coloca en el campo de prueba su representación binaria, es decir, ‘001’. Si S_PA es igual a cero, entonces el estado siguiente es el Estado ‘0000’, su representación binaria ‘0000’ es colocada en el campo de la liga, ya que se requiere hacer una carga. El campo VF es igual cero, ya que, para hacer una carga en el contador, el valor de la entrada y de VF deben ser iguales. En el Estado ‘0000’ la señal M_A está activada, por lo que se coloca un ‘1’ en la parte de salidas de M_A, y ‘0’ en M_C.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria, ver figura P21.3). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits.

Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P21.4 Contenido de la memoria para el sistema de puerta del elevador.

Dirección de memoria				Contenido de memoria											
Estado presente				Prueba			VF	Liga				Salidas		Hex 1	Hex 2
QA	QB	QC	QD	I2	I1	I0		L3	L2	L1	L0	M_A	M_C		
0	0	0	0	0	0	1	0	0	0	0	0	1	0	20	02
0	0	0	1	0	1	1	0	0	1	0	1	0	0	65	00
0	0	1	0	0	1	0	1	0	0	0	0	0	1	50	01
0	0	1	1	1	0	1	0	0	0	1	0	0	1	A2	01
0	1	0	0	0	0	0	1	0	0	0	0	0	0	10	00
0	1	0	1	1	0	0	1	1	0	0	0	0	0	98	00
0	1	1	0	1	1	0	1	0	0	1	0	0	0	D2	00
0	1	1	1	0	0	0	1	0	0	0	1	0	0	11	00
1	0	0	0	0	0	0	1	0	0	0	1	0	0	11	00

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de ocho líneas a una, éste es implementado por el PIC16F1939. El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P21.5). Se puede comprobar realizando el circuito de la figura P21.7.

Tabla P21.5 Tabla de funcionamiento del selector de entradas para el sistema de puerta del elevador.

Prueba			SI
I2	I1	I0	
0	0	0	AUX
0	0	1	S_PA
0	1	0	S_PC
0	1	1	T5
1	0	0	S_OBS
1	0	1	F_AP
1	1	0	B_CP

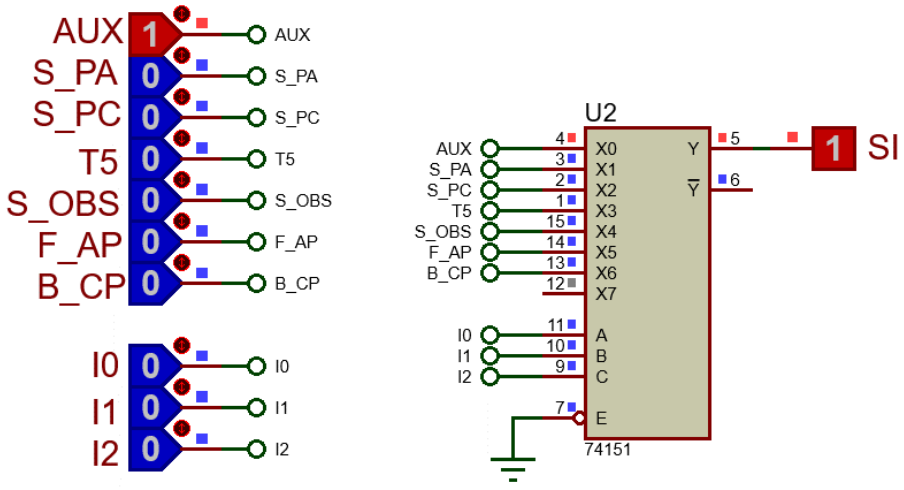


Figura P21.7 Multiplexor 74151 para el selector de entradas del sistema de puerta del elevador.

Teniendo en cuenta que `T5 = F3&!F2&F1&!F0`, donde F3, F2, F1 y F0 son los bits del temporizador. La función booleana del selector de entradas queda:

```
SI = AUX&!I2&!I1&!I0 | S_PA&!I2&!I1&I0 | S_PC&!I2&I1&!I0 |
     (F3&!F2&F1&!F0)&!I2&I1&I0 | S_OBS&I2&!I1&!I0 | F_AP&I2&!I1&I0 |
     B_CP&I2&I1&!I0;
```

Para obtener el valor de la lógica, se debe hacer la operación XOR entre el selector de entradas y el valor de VF (ver figura P21.5).

Por lo tanto, la función booleana de la lógica queda:

```
SL = VF ^ (AUX&!I2&!I1&!I0 | S_PA&!I2&!I1&I0 | S_PC&!I2&I1&!I0 |
     (F3&!F2&F1&!F0)&!I2&I1&I0 | S_OBS&I2&!I1&!I0 | F_AP&I2&!I1&I0 |
     B_CP&I2&I1&!I0);
```

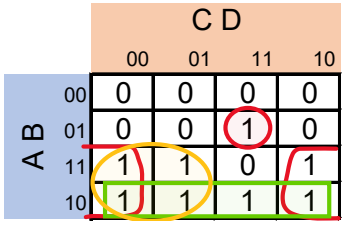
Se utiliza un contador con carga paralela que indica qué estado es el siguiente. El contador con carga paralela es implementado por el PIC16F1939. Si el valor a la salida de la lógica es igual a ‘1’, el contador carga la información binaria de la liga a la memoria.

Si el valor de la lógica es igual a ‘0’, cuenta al siguiente estado binario. A continuación, se obtienen las expresiones lógicas para un contador de cuatro bits por el método de variable suscrita (ver Tabla P21.6).

Tabla P21.6 Mapa de Karnaugh general de transición de estados para un contador de cuatro bits.

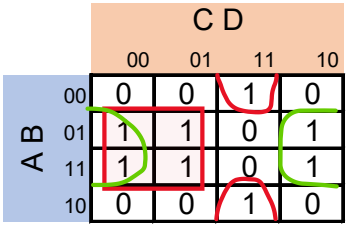
		C D			
		0 0	0 1	1 1	1 0
A B	0 0	0000	0001	0011	0010
		0001	0010	0100	0011
	0 1	0100	0101	0111	0110
		0101	0110	1000	0111
	1 1	1100	1101	1111	1110
		1101	1110	0000	1111
	1 0	1000	1001	1011	1010
		1001	1010	1100	1011

Para el bit A:



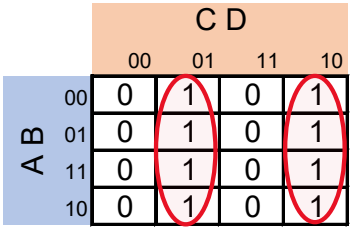

```
FFA = A&!C | A&!B | A&!D |
!A&B&C&D;
```

Para el bit B:



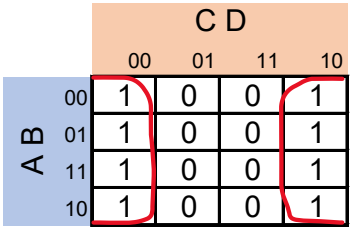

```
FFB = B&!C | B&!D | !B&C&D;
```

Para el bit C:




```
FFC = !C&D | C&!D = C^D;
```

Para el bit D:




```
FFD = !D;
```

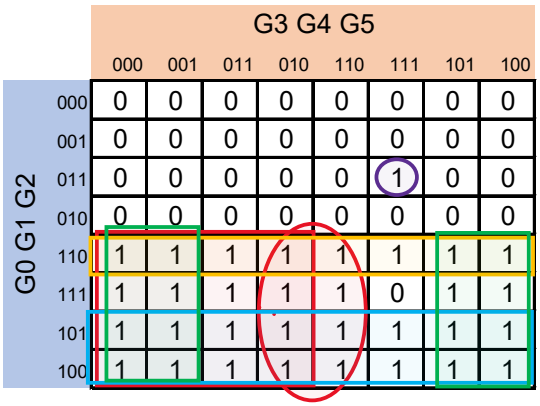
En función de lo planteado se propone una solución para el temporizador de 5 segundos del elevador. El temporizador es implementado de manera que se use una sola señal de reloj para todo el controlador. La señal de reloj para el PIC es de 64 Hz, por lo tanto, se necesita un divisor de frecuencia para poder obtener una señal de 1 Hz para el temporizador. Un contador binario de 6 bits es el divisor de frecuencia, debido a que con la salida del bit más significativo de los 6 flip-flops del contador se obtiene 1/64 de la frecuencia del reloj de entrada.

A continuación, se obtienen las expresiones lógicas para un contador de seis bits por el método de variable suscrita (ver tabla P21.7).

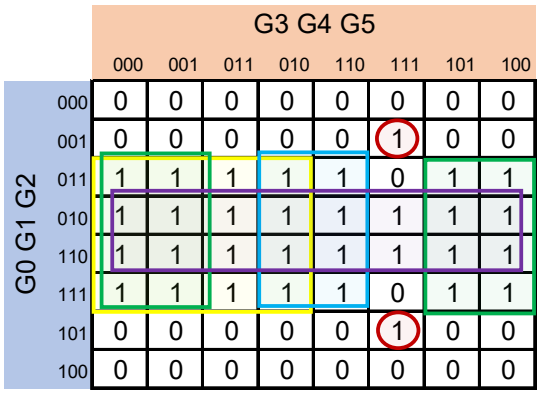
Tabla P21.7 Mapa de Karnaugh general de transición de estados para un contador de seis bits.

G0 G1 G2 \ G3 G4 G5	0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0
0 0 0	000000	000001	000011	000010	000110	000111	000101	000100
	000001	000010	000100	000011	000111	001000	000110	000101
0 0 1	001000	001001	001011	001010	001110	001111	001101	001100
	001001	001010	001100	001011	001111	010000	001110	001101
0 1 1	011000	011001	011011	011010	011110	011111	011101	011100
	011001	011010	011100	011011	011111	100000	011110	011101
0 1 0	010000	010001	010011	010010	010110	010111	010101	010100
	010001	010010	010100	010011	010111	011000	010110	010101
1 1 0	110000	110001	110011	110010	110110	110111	110101	110100
	110001	110010	110100	110011	110111	111000	110110	110101
1 1 1	111000	111001	111011	111010	111110	111111	111101	111100
	111001	111010	111100	111011	111111	000000	111110	111101
1 0 1	101000	101001	101011	101010	101110	101111	101101	101100
	001001	001010	001100	001011	001111	010000	001110	001101
1 0 0	100000	100001	100011	100010	100110	100111	100101	100100
	100001	100010	100100	100011	100111	101000	100110	100101

Mapas de Karnaugh particulares para los bits G0, G1, G2, G3, G4 y G5 respectivamente:




```
FFG0 = G0&G4&!G5 | G0&!G3 | G0&!G4 |
       G0&G1&!G2 | G0&!G1 |
       !G0&G1&G2&G3&G4&G5;
```




```
FFG1 = G1&!G3 | G1&!G4 | G1&G4&!G5 |
       G1&!G2 | !G1&G2&G3&G4&G5;
```

G0 G1 G2 \ G3 G4 G5	000	001	011	010	110	111	101	100
000	0	0	0	0	0	1	0	0
001	1	1	1	1	1	0	1	1
011	1	1	1	1	1	0	1	1
010	0	0	0	0	0	1	0	0
110	0	0	0	0	0	1	0	0
111	1	1	1	1	1	0	1	1
101	1	1	1	1	1	0	1	1
100	0	0	0	0	0	1	0	0

```
FFG2 = G2&!G3 | G2&G4&!G5 |
       G2&G3&!G4 |  G1&!G2&G3&G4&G5
       | !G1&!G2&G3&G4&G5;
```

G0 G1 G2 \ G3 G4 G5	000	001	011	010	110	111	101	100
000	0	1	0	1	1	0	1	0
001	0	1	0	1	1	0	1	0
011	0	1	0	1	1	0	1	0
010	0	1	0	1	1	0	1	0
110	0	1	0	1	1	0	1	0
111	0	1	0	1	1	0	1	0
101	0	1	0	1	1	0	1	0
100	0	1	0	1	1	0	1	0

```
FFG4 = G4&!G5 | !G4&G5 = G4^G5;
```

G0 G1 G2 \ G3 G4 G5	000	001	011	010	110	111	101	100
000	0	0	1	0	1	0	1	1
001	0	0	1	0	1	0	1	1
011	0	0	1	0	1	0	1	1
010	0	0	1	0	1	0	1	1
110	0	0	1	0	1	0	1	1
111	0	0	1	0	1	0	1	1
101	0	0	1	0	1	0	1	1
100	0	0	1	0	1	0	1	1

```
FFG3 = G3&G4&!G5 | !G3&G4&G5 |
       G3&!G4;
```

G0 G1 G2 \ G3 G4 G5	000	001	011	010	110	111	101	100
000	1	0	0	1	1	0	0	1
001	1	0	0	1	1	0	0	1
011	1	0	0	1	1	0	0	1
010	1	0	0	1	1	0	0	1
110	1	0	0	1	1	0	0	1
111	1	0	0	1	1	0	0	1
101	1	0	0	1	1	0	0	1
100	1	0	0	1	1	0	0	1

```
FFG5 = !G5;
```

Por lo tanto, la entrada de 1 Hz al temporizador es la expresión lógica del bit G0. El temporizador es implementado por el método de variable suscrita. El temporizador cuenta 5 segundos cambiando de estado. El cambio de estado depende de la entrada baja del pulso de reloj, con la entrada alta de esta misma señal, es decir la señal de G0. La carta ASM del temporizador se muestra en la figura P21.8.

La máquina de estados del temporizador y la de la puerta funciona conjuntamente. Es decir, las máquinas de estados de este sistema estarán relacionadas o vinculadas entre sí.

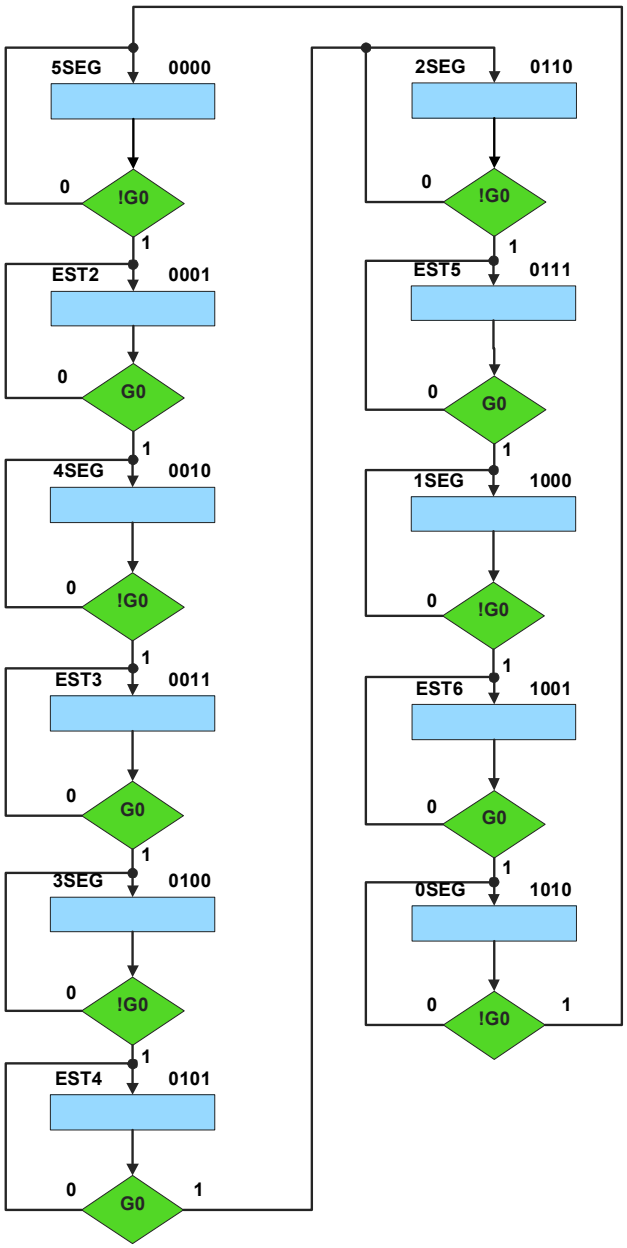


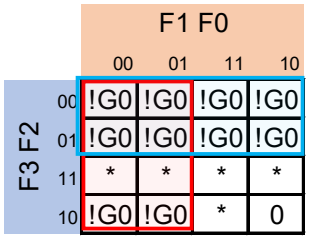
Figura P21.8 Carta ASM del temporizador de 5 segundos.

A partir de la carta ASM del temporizador de 5 segundos, se obtienen las expresiones lógicas del temporizador por el método de variable suscrita. A continuación, se proporciona el mapa de Karnaugh general de transición de estados (ver tabla P21.8).

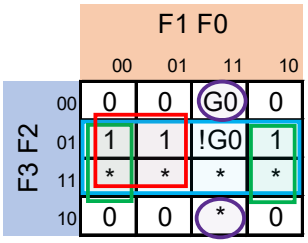
Tabla P21.8 Mapa de Karnaugh general de transición de estados para el temporizador de 5 segundos.

F3 F2 \ F1 F0	0 0	0 1	1 1	1 0
0 0	0000	0001	0011	0010
	!G0 0001	G0 0010	G0 0100	!G0 0011
0 1	0100	0101	0111	0110
	!G0 0101	G0 0110	G0 1000	!G0 0011
1 1	1100	1101	1111	1110
	*	*	*	*
1 0	1000	1001	1011	1010
	!G0 1001	G0 1010	*	!G0 0000

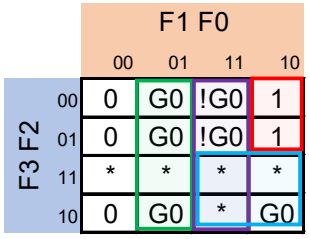
Mapas de Karnaugh particulares para los bits F0, F1, F2, F3 y F4 respectivamente:



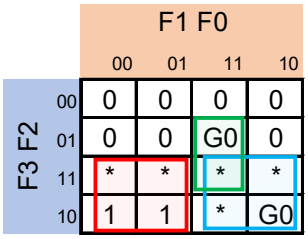

```
F0n = !F1&!G0 | !F3&!G0;
```




```
F2n = F2&!F1 | F2&!F0 | F2&!G0 |
!F2&F1&F0&G0;
```




```
F1n = !F3&F1&!F0 | !F1&F0&G0 |
      F3&F1&G0 | F1&F0&!G0;
```




```
F3n = F3&!F1 | F2&F1&F0&G0 |
      F3&F1&G0;
```

Todas las expresiones de los bits del temporizador de 5 segundos realizan la operación AND con los términos: `!A&!B&!C&D | !A&B&!C&D | !A&B&C&!D | !A&B&C&D`. De este modo el temporizador sólo cuenta en los estados indicados de la carta ASM del sistema de la puerta y en los demás estados el temporizador se reinicia y no cuenta.

Se utiliza un display de 7 segmentos para poder visualizar de una mejor manera el funcionamiento del temporizador. Por lo tanto, se utiliza un decodificador de BCD a 7 segmentos. La tabla de verdad para las entradas al decodificador se muestra en la tabla P21.9.

Tabla P21.9 Tabla de verdad para las entradas al decodificador.

Bits del temporizador				Entradas al decodificador		
F3	F2	F1	F0	Cp	Bp	Ap
0	0	0	0	1	0	1
0	0	1	0	1	0	0
0	1	0	0	0	1	1
0	1	1	0	0	1	0
1	0	0	0	0	0	1
1	0	1	0	0	0	0

Los mapas de Karnaugh para las entradas al decodificador se muestran a continuación.

F3 F2 \ F1 F0	00	01	11	10
00	1	*	*	0
01	1	*	*	0
11	*	*	*	*
10	1	*	*	0

`Ap = !F1;`

F3 F2 \ F1 F0	00	01	11	10
00	0	*	*	0
01	1	*	*	1
11	*	*	*	*
10	0	*	*	0

`Bp = F2;`

F3 F2 \ F1 F0	00	01	11	10
00	1	*	*	1
01	0	*	*	0
11	*	*	*	*
10	0	*	*	0

`Cp = !F3&!F2;`

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador (ver figura P21.9, P21.10). Se carga en el controlador los archivos con extensión “HEX” de las memorias y los archivos “COF” o “HEX” del PIC16F1939.

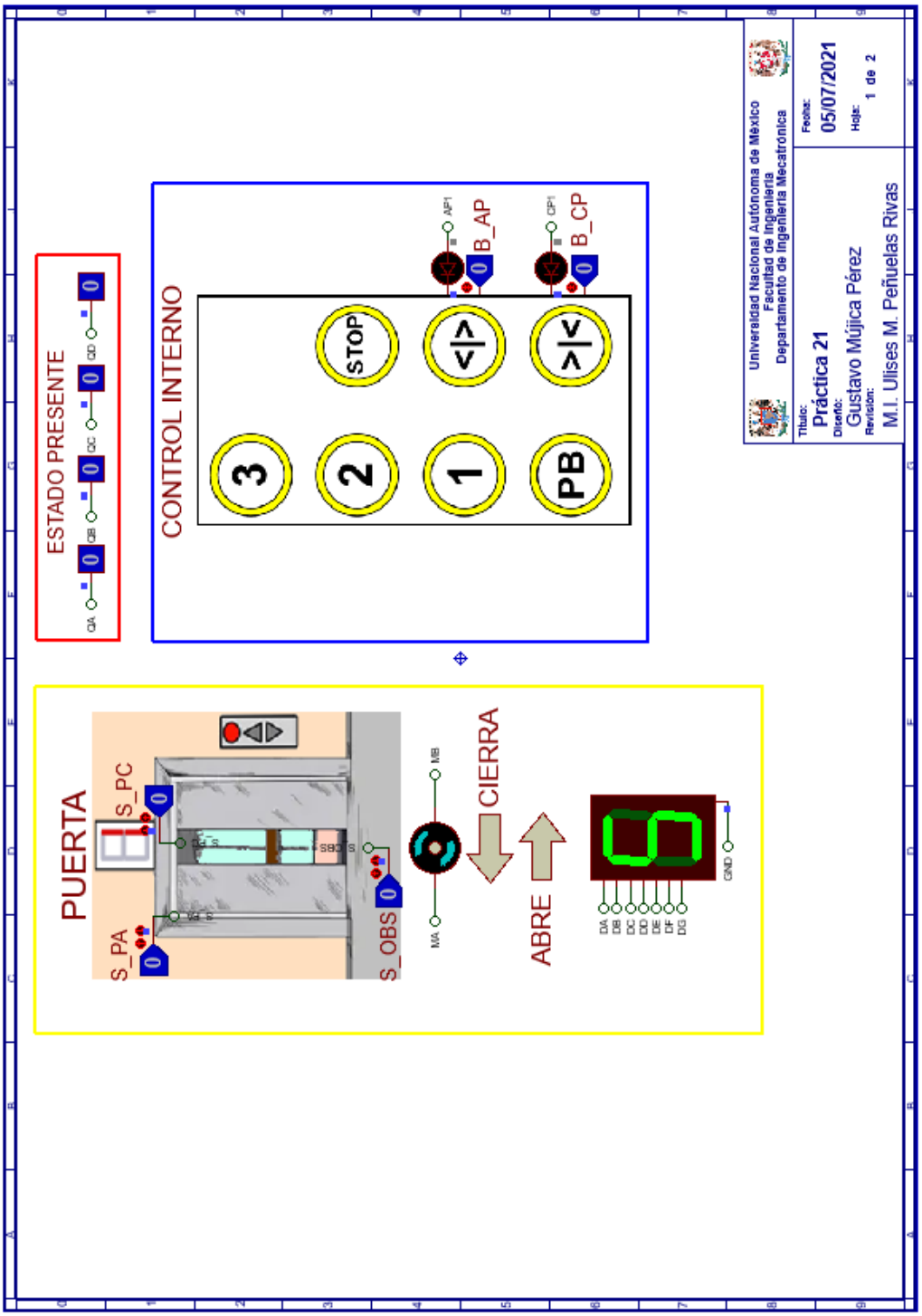


Figura P21.9 Interfaz hombre-máquina para el controlador de la Práctica 21 hoja 1/2.

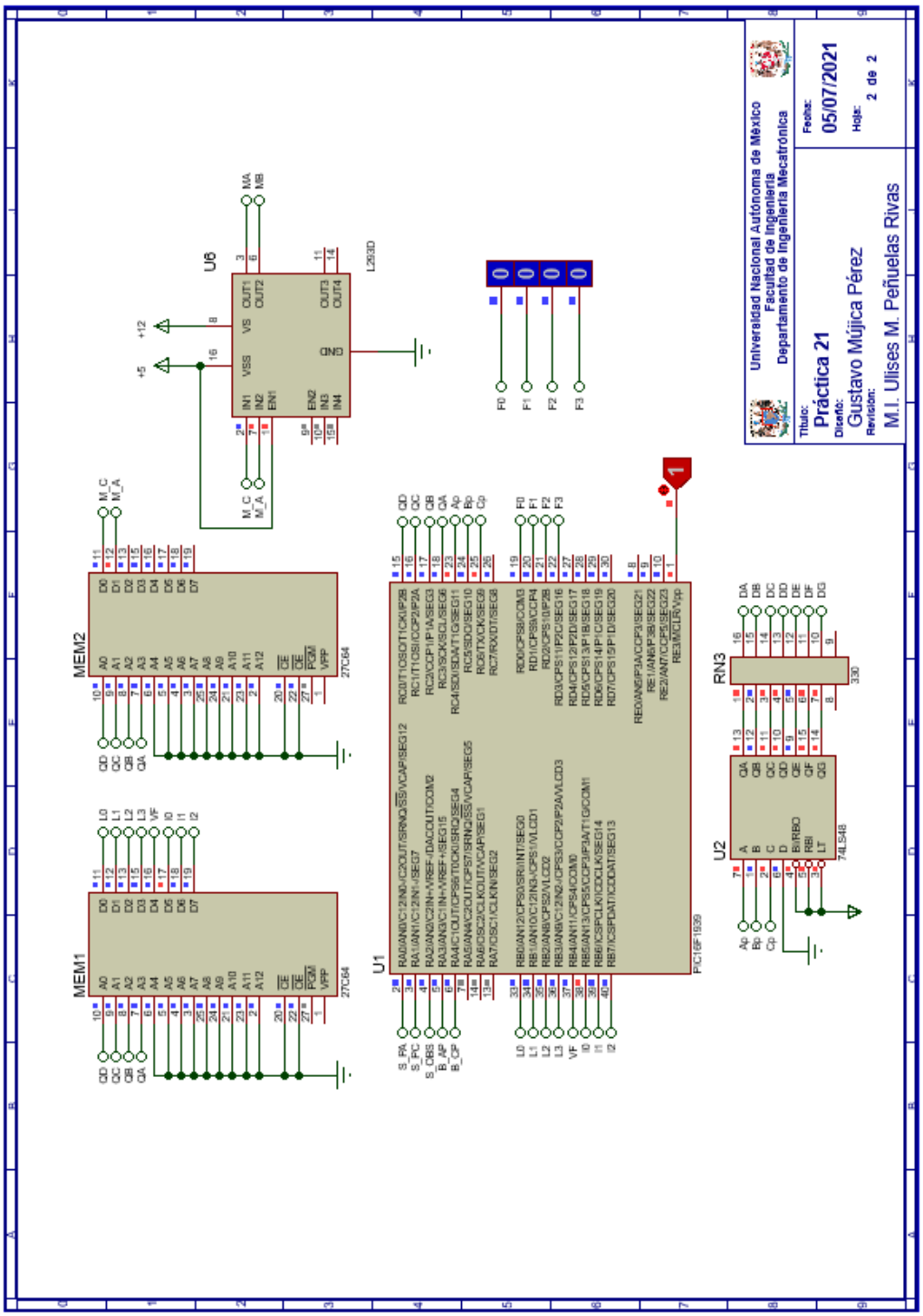


Figura P21.10 Esquema electrónico para el controlador de la Práctica 21 hoja 2/2.

Código

Para la programación del PIC16F1939 se utiliza un software para microcontroladores (ver figura P21.11, figura P21.12 y figura P21.13) y se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     // Carga biblioteca PLD.h

//***CONTADOR***
//#define CLK A0 //RELOJ

//ENTRADAS
//LIGA
#define L0 B0
#define L1 B1
#define L2 B2
#define L3 B3

//SALIDAS
#define A C3 //FFA
#define B C2 //FFB
#define C C1 //FFC
#define D C0 //FFD

//****TEMPORIZADOR****

#define Ap C4
#define Bp C5
#define Cp C6

#define F0 D0
#define F1 D1
#define F2 D2
#define F3 D3

//***LOGICA***

//ENTRADAS
#define S_PA   A0
#define S_PC   A1
#define S_OBS  A2
#define B_AP   A3
#define B_CP   A4

//VF
#define VF B4

//PRUEBAS
#define I0 B5
#define I1 B6
#define I2 B7

//***VARIABLES INTERMEDIAS***
short SL=0;  //Salida lógica
short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
```

Figura P21.11 Código de la Práctica 21 parte 1.

```
50: short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
51: short At=0, Bt=0, Ct=0, Dt=0; //Variables intermedias de los FF
52: short LD3,LD2,LD1,LD0; //Variables intermedias de la liga
53: short F0n, F1n, F2n, F3n;// Variables intermedias del temporizador
54: //Variables intermedias del divisor de frecuencia
55: short G0, G1, G2, G3, G4, G5, G0t, G1t, G2t, G3t, G4t, G5t;
56: short F_AP; //Función abrir puerta
57:
58: void main ()
59: {
60: pld_ini();  // INICIALIZA AL PIC COMO PLD
61: pld_555(64);   // Genera señal cuadrada en Hz,
62:                  //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
63:
64:    //LOOP INFINITO
65:    while(1)
66:    {
67:    //****CIRCUITO COMBINACIONAL****
68:    LD3=L3; LD2=L2; LD1=L1; LD0=L0;  /*ALMACENA DATOS HASTA EL CAMBIO DEL RELOJ,
69:                                       SIMULANDO UN REGISTRO, EVITANDO ASÍ
70:                                  QUE SE MODIFIQUEN LOS VALORES DE LA MEMORIA*/
71:    // Entradas al decodificador
72:     Ap =!F1;
73:     Bp =F2;
74:     Cp =!F3&!F2;
75:
76:    //Funcion
77:    F_AP=  S_OBS | B_AP;
78:    //SALIDA LÓGICA
79:    SL = VF ^ (AUX&!I2&!I1&!I0 | S_PA&!I2&!I1&I0 | S_PC&!I2&I1&!I0 |
80:         (F3&!F2&F1&!F0)&!I2&I1&I0 | S_OBS&I2&!I1&!I0 |
81:         F_AP&I2&!I1&I0 | B_CP&I2&I1&!I0);
82:
83:    //****CIRCUITO SECUENCIAL ****
84:
85:    // if (!CLK) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO
86:    if (!out_555) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO,
87:                     //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
88:
89:    {//SECCIÓN DE OPERACIONES DEL CONTADOR CON CARGA PARALELA
90:    At= A&!C | A&!B | A&!D | !A&B&C&D;
91:    Bt= B&!C | B&!D | !B&C&D;
92:    Ct= C^D;
93:    Dt= !D;
94:
95:    //DIVISOR DE FRECUENCIA PARA OBTENER 1/64 DE LA FRECUENCIA DEL RELOJ
96:    G0t= G0&G4&!G5 | G0&!G3 | G0&!G4 | G0&G1&!G2 | G0&!G1 | !G0&G1&G2&G3&G4&G5;
97:    G1t= G1&!G3 | G1&!G4 | G1&G4&!G5 | G1&!G2 | !G1&G2&G3&G4&G5;
98:    G2t= G2&!G3 | G2&G4&!G5 | G2&G3&!G4 |  G1&!G2&G3&G4&G5 | !G1&!G2&G3&G4&G5;
99:    G3t= G3&G4&!G5 | !G3&G4&G5 |  G3&!G4;
```

Figura P21.12 Código de la Práctica 21 parte 2.

```
 99:    G3t= G3&G4&!G5 | !G3&G4&G5 |  G3&!G4;
100:    G4t= G4^G5;
101:    G5t= !G5;
102:
103:    //EXPRESIONES LÓGICAS PARA EL TEMPORIZADOR DE 5 SEGUNDOS
104:    F0n= (!A&!B&!C&D | !A&B&!C&D | !A&B&C&!D | !A&B&C&D)&(!F1&!G0 | !F3&!G0);
105:    F1n= (!A&!B&!C&D | !A&B&!C&D | !A&B&C&!D | !A&B&C&D)&(!F3&F1&!F0 |!F1&F0&G0 | F3&F1&G0 | F1&F0&!G0);
106:    F2n= (!A&!B&!C&D | !A&B&!C&D | !A&B&C&!D | !A&B&C&D)&(F2&!F1 | F2&!F0 | F2&!G0 | !F2&F1&F0&G0);
107:    F3n= (!A&!B&!C&D | !A&B&!C&D | !A&B&C&!D | !A&B&C&D)&(F3&!F1 | F2&F1&F0&G0 | F3&F1&G0);
108:
109:    }
110:
111:    else
112:    {//SECCIÓN DE MEMORIZACIÓN
113:    //CUENTA CON SL=1 Y CARGA CON SL=0
114:    A= SL&At | !SL&LD3;
115:    B= SL&Bt | !SL&LD2;
116:    C= SL&Ct | !SL&LD1;
117:    D= SL&Dt | !SL&LD0;
118:
119:    G0 = G0t;
120:    G1 = G1t;
121:    G2 = G2t;
122:    G3 = G3t;
123:    G4 = G4t;
124:    G5 = G5t;
125:
126:    F0= F0n;
127:    F1= F1n;
128:    F2= F2n;
129:    F3= F3n;
130:
131:   /* while(clk){}/*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
132:                 POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
133:                 DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
134:                 LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ*/
135:
136:    while(out_555){} /*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
137:                    POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
138:                    DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
139:                    LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ,
140:                    EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO*/
141:    }
142:    }
143: }
```

Figura P21.13 Código de la Práctica 21 parte 3.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

Práctica 22 Sistema del elevador; diseño con memoria y direccionamiento implícito

Introducción

A través de una maqueta, se simuló el funcionamiento de un elevador de un edificio de tres pisos por medio de una maqueta que está dividida en tres partes, las cuales son: cubo del elevador, puerta que abre y cierra, y el tablero de control.

El cubo tiene cuatro niveles, los cuales son: planta baja, piso 1, piso 2 y piso 3. Donde un carro sube y baja simulando el movimiento del elevador. Cada uno de los pisos tiene controles a un costado de la puerta para subir o bajar de piso, así como sensores de presencia que indican en donde está el carro (ver figura P22.1).

La maqueta cuenta con una caja estática que sirve de monitor de estados para visualizar el abrir y cerrar de la puerta. La puerta se abre por medio de un motor cuando el carro del elevador llegue al piso seleccionado. También se debe abrir cuando se presionen los botones de abrir o cerrar puerta mientras el carro este estático en un piso. Además, si se obstruye el paso de la puerta mientras la puerta se esté cerrando, se debe abrir inmediatamente (ver figura P22.2).

El tablero de control simula el control interno del elevador. Tiene botones para: dirigirse a los 4 niveles del elevador, detener elevador, abrir y cerrar la puerta, así como un indicador del piso en el que se encuentra el elevador (ver figura P22.3).

Para comprender mejor el funcionamiento del elevador este se dividió en cinco subsistemas, los cuales son: planta baja, piso 1, piso 2, piso 3 y el sistema de abrir/cerrar puerta.

En esta práctica se unen los 5 subsistemas para formar el sistema del elevador, además se agregan las posibles combinaciones de botones que puede tener el elevador.

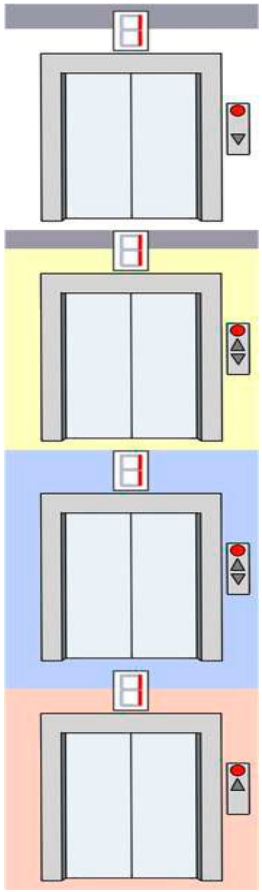


Figura P22.1 Elevador en torre de 4 niveles.

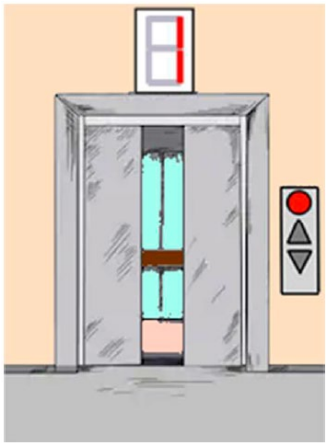


Figura P22.2 Puerta que abre y cierra.

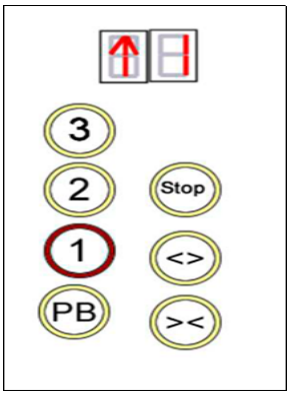


Figura P22.3 Tablero de control interno del cubo del elevador.

El diseño con memoria y direccionamiento implícito utiliza solamente un campo de liga. Se selecciona una variable de entrada por medio del campo de prueba (ver figura P22.4). El campo VF decide si se utiliza la dirección de liga (se carga el valor de liga) o no (se incrementa el valor del contador en una unidad). La figura P22.5 muestra la arquitectura de este método.

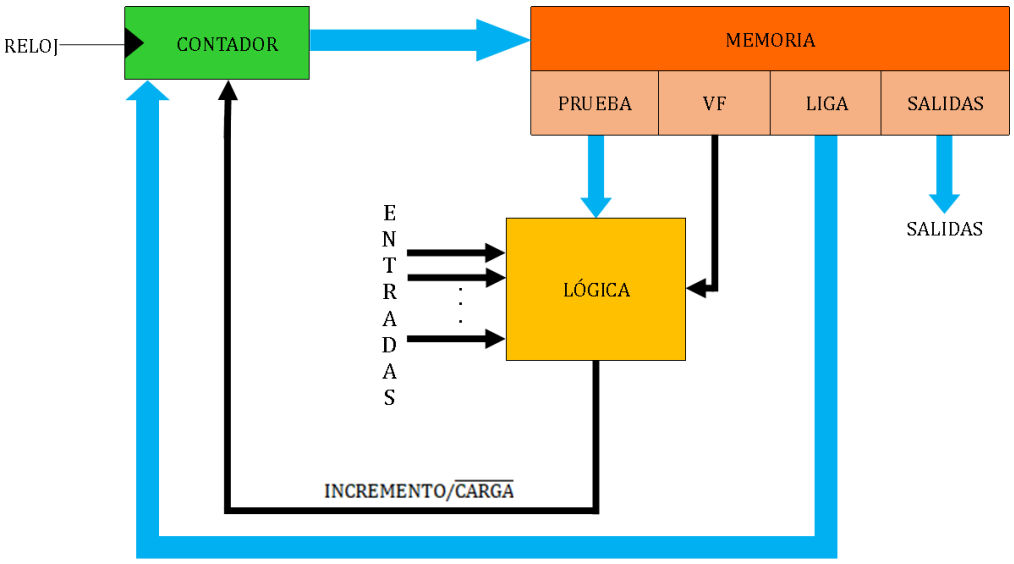


Figura P22.4 Arquitectura de un diseño con direccionamiento implícito.

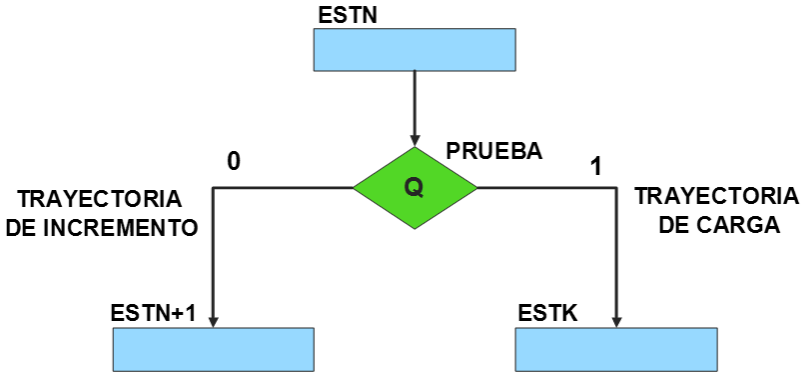


Figura P22.5 Trayectoria de incremento y carga.

La tabla P22.1 muestra la relación de VF y la variable de entrada con la señal de incremento o carga. La variable VF, que indica para que calor de entrada se hace la carga, y la variable de entrada se relacionan por medio de una función XOR, cuando el resultado de la función da como resultado un '1' se hace un incremento, cuando el resultado de la función da como resultado un '0' se hace una carga.

Tabla P22.1 Relación entre VF, variable de entrada y la señal de incremento o carga.

VF	VARIABLE DE ENTRADA	INCREMENTO/$\overline{\text{CARGA}}$
0	0	0
0	1	1
1	0	1
1	1	0

La señal de incrementa o carga ingresará a un contador con carga paralela. Si la señal que sale de la lógica es '0', se hará una carga, es decir, para hacer una carga el valor de la entrada y de VF deben ser iguales. Si la señal que sale de la lógica es '1', se hará un incremento, es decir, para hacer un incremento el valor de la variable de entrada y VF deben ser diferentes (ver figura P22.6).

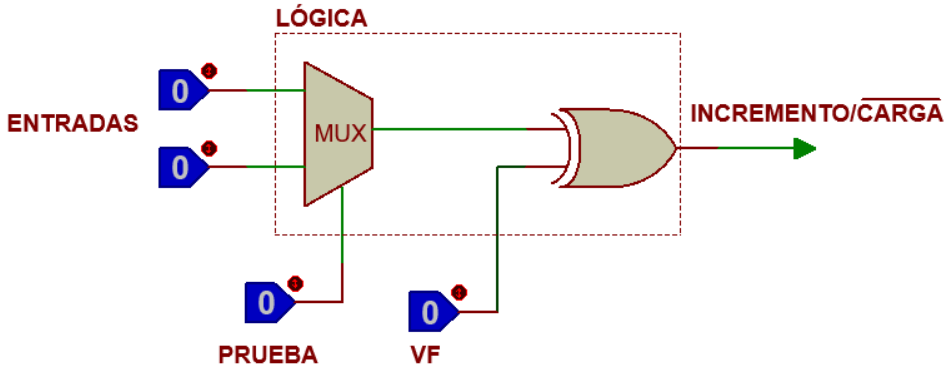


Figura P22.6 Bloque de lógica de incremento/carga para el direccionamiento implícito [1].

Además de asignar una representación binaria a cada estado, también a cada variable de entrada se le asignará una representación. Se utilizará una variable auxiliar que sirve para los estados que no tengan variable de entrada, de manera que cuando en un estado no exista variable de entrada se probará la variable auxiliar, la cual puede tener un valor de cero o uno, se prefiere utilizar el valor uno que presenta un nivel lógico alto.

Objetivo

Diseñar un controlador para el sistema del elevador, por medio del método de diseño con memoria y direccionamiento implícito.

Descripción

Se diseña una carta ASM y se propone una solución para el sistema del elevador por medio del método de diseño con memoria y direccionamiento de implícito. Teniendo en cuenta que, al ser el sistema del elevador completo, se propone en el diseño de la carta ASM combinaciones de botones para desplazarse a varios pisos. Posteriormente se propone una solución para guardar un registro de los botones presionados. También se propone una solución para mostrar en que piso se encuentra el elevador en todo momento. Por último, se propone una solución para implementar el sistema del elevador.

Tabla de entradas y salidas

En la tabla P22.2 se muestran los detalles de las entradas y salidas de este controlador.

Para el sistema del elevador se necesitan señales de entrada:

- un botón detecta la solicitud para subir a otro piso desde la planta baja
- botones detectan la solicitud para subir o bajar otro piso desde el piso 1 y el piso 2
- un botón detecta la solicitud para bajar a otro piso desde el piso 3
- sensores detectan en que piso se encuentra el elevador
- botones dentro del elevador detectan la solicitud para que para ir a: planta baja, piso 1, piso 2 y piso 3. así mismo, hay botones para: abrir puerta, cerrar puerta y para detener el elevador
- un sensor detecta si la puerta está abierta completamente y otro si está cerrada en su totalidad
- una señal indica que han transcurrido 5 segundos desde que se abrió la puerta completamente
- un sensor verifica si hay una obstrucción en la puerta.

Como salida se requiere de las siguientes señales:

- se usan señales de salida para indicarle al elevador cuando debe bajar o subir
- se usan señales de salida para indicarle a la puerta que debe abrirse o cerrarse.

Tabla P22.2 Entradas y salidas para el sistema del elevador.

Identificador	Ubicación	Tipo	Descripción
B_PBn	A0 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de planta baja
B_P1n	A1 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de piso 1
B_P2n	A2 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de piso 2
B_P3n	A3 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de piso 3
B_SPBn	A4 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de subir desde planta baja
B_SP1n	A5 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de subir desde piso 1
B_SP2n	A6 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de subir desde piso 2
B_BP3n	A7 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de bajar desde piso 3
B_BP2n	B0 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de bajar desde piso 2
B_BP1n	B1 del PIC 2	I	Señal de led que indica si está presionado o fue presionado el botón de bajar desde piso 1
S_PB	B2 del PIC 1 y PIC 2	I	Sensor de planta baja
S_P1	B3 del PIC 1 y PIC 2	I	Sensor de piso 1
S_P2	B4 del PIC 1 y PIC 2	I	Sensor de piso 2
S_P3	B5 del PIC 1 y PIC 2	I	Sensor de piso 3
S_PA	B6 del PIC 2	I	Sensor de puerta abierta
S_PC	B7 del PIC 2	I	Sensor de puerta cerrada
B_CP	D0 del PIC 2	I	Botón para cerrar puerta
B_AP	D1 del PIC 2	I	Botón para abrir puerta
S_OBS	D2 del PIC 2	I	Sensor de obstrucción
B_STP	D4 del PIC 2	I	Botón para detener el elevador
T5	D1 del PIC 3	I	Señal que indica cuando han transcurrido 5 segundos
M_B	D0 de la memoria 2	O	Señal para bajar el elevador
M_S	D1 de la memoria 2	O	Señal para subir el elevador
M_C	D2 de la memoria 2	O	Señal para que se cierre la puerta
M_A	D3 de la memoria 2	O	Señal para que se abra la puerta

Notas de diseño

a) Los botones de: planta baja, piso1, piso 2, piso 3, abrir puerta, cerrar puerta y paro del elevador se encuentran en el tablero de control, el cual simula el control de los botones dentro del elevador
b) A un costado de la entrada de cada piso se encuentran ubicados los botones de: subir desde planta baja, subir desde piso 1, bajar desde piso 1, subir desde piso 2, bajar desde piso 2 y bajar desde piso 3
c) Se guarda un registro de los botones presionados que estén relacionados con un piso, de manera que se puedan presionar varios botones
d) Para poder guardar un registro de los botones presionados se utilizan leds. De manera que, si el botón de un piso es presionado, se enciende un led asociado a éste
e) El botón de piso 3 es el de mayor prioridad
f) El temporizador empieza a contar automáticamente sin necesidad de una señal de salida
g) Sólo se puede abrir y cerrar la puerta cuando el elevador no esté en movimiento
h) Sólo se puede detener elevador cuando éste se encuentre en movimiento
i) El botón de paro debe mantenerse presionado para que no se mueva el elevador.

Reglas de funcionamiento

- B_PBn: led del botón de planta baja
  1 = led del botón de planta baja está encendido
  0 = led del botón de planta baja está apagado
- B_SPBn: led del botón para subir desde planta baja
  1 = led del botón para subir desde planta baja está encendido
  0 = led del botón para subir desde planta baja está apagado
- S_PB: sensor de planta baja
  1 = el elevador está en la planta baja
  0 = no está el elevador en la planta baja
- B_P1n: led del botón de piso 1
  1 = led del botón de piso 1 está encendido
  0 = led del botón de piso 1 está apagado

- B_SP1n: led del botón para subir desde piso 1
  1 = led del botón para subir desde el piso 1 está encendido
  0 = led del botón para subir desde el piso 1 está apagado
- B_BP1n: led del botón para bajar desde piso 1
  1 = led del botón para bajar desde el piso 1 está encendido
  0 = led del botón para bajar desde el piso 1 está apagado
- S_P1: sensor de piso 1
  1 = el elevador está en el piso 1
  0 = no está el elevador en el piso 1
- B_P2n: led del botón para de piso 2
  1 = led del botón de piso 2 está encendido
  0 = led del botón de piso 2 está apagado
- B_SP2n: led del botón para subir desde piso 2
  1 = led del botón para subir desde el piso 2 está encendido
  0 = led del botón para subir desde el piso 2 está apagado
- B_BP2n: led del botón para bajar desde piso 2
  1 = led del botón para bajar desde el piso 2 está encendido
  0 = led del botón para bajar desde el piso 2 está apagado
- S_P2: sensor de piso 2
  1 = el elevador está en el piso 2
  0 = no está el elevador en el piso 2
- B_P3n: led del botón de piso 3
  1 = led del botón de piso 3 está encendido
  0 = led del botón de piso 3 está apagado
- B_BP3n: led del botón para bajar desde piso 3
  1 = led del botón para bajar desde el piso 3 está encendido
  0 = led del botón para bajar desde el piso 3 está apagado
- S_P3: sensor de piso 3
  1 = el elevador está en el piso 3
  0 = no está el elevador en el piso 3

- S_PA: sensor de puerta abierta
  1 = detecta puerta completamente abierta
  0 = no detecta puerta completamente abierta
- S_PC: sensor de puerta cerrada
  1 = detecta puerta completamente cerrada
  0 = no detecta puerta completamente cerrada
- T5: señal de 5 segundos
  1 = han transcurrido 5 segundos
  0 = no han transcurrido 5 segundos
- S_OBS: sensor de obstrucción
  1 = se está obstruyendo la puerta
  0 = no se está obstruyendo la puerta
- B_AP: botón para abrir puerta
  1 = se oprimió el botón para abrir puerta
  0 = no se oprimió el botón para abrir puerta
- B_CP: botón para cerrar puerta
  1 = se oprimió el botón para cerrar puerta
  0 = no se oprimió el botón para cerrar puerta
- B_STP: botón de paro
  1 = se oprimió el botón de paro
  0 = no se oprimió el botón de paro.

Descripción de la carta ASM

Aquí se realiza una breve descripción de la secuencia de pasos que se deben seguir en el mundo real para lograr el objetivo de la práctica. Cada paso se denomina Estado y se numeran de acuerdo con su secuencia de aparición en el algoritmo. Además, en el título de cada estado, se agrega una etiqueta que resume la actividad que se realiza en dicho estado.

Se hace un incremento cuando el valor del estado siguiente aumenta en una unidad, de lo contrario se hace una carga (ver figura P22.5). El algoritmo de la máquina de estados se puede ver de la figura P22.7 a la figura P22.12.

Estado '000000' – BOTAP

En el primer estado, el elevador se encuentra inmóvil y se está a la espera de que se oprima un botón. Si se presiona el botón para abrir la puerta (B_AP), el sistema avanza al Estado '010100' para abrir la puerta. De lo contrario, avanza al Estado '000001' para revisar si está encendido el led del botón del piso 3.

Estado '000001' – BOTP3

En este estado, cuando no está encendido el led del botón de piso 3 (B_P3), el sistema avanza al Estado '000111' para revisar el led del botón de planta baja. De lo contrario, avanza al Estado '000010' para revisar el sensor de planta baja, dado que, si el elevador está en la planta baja, se puede ir al piso 1 antes de ir al piso 3.

Estado '000010' – BP3SPB

En este estado si el sensor (S_PB), detecta que el elevador está en la planta baja, el sistema avanza al Estado '000011' para verificar si se está encendido el led del botón de piso 1 o el led del botón para subir desde el piso 1. De lo contrario, avanza al Estado '000100' para revisar el sensor de piso 1, porque si el elevador está en el piso 1, se puede ir al piso 2 antes de ir al piso 3.

Estado '000011' – BP1BSP1

En este estado si se está encendido el led del botón del piso 1 (B_P1) o el led del botón para subir desde el piso 1 (B_SP1), el sistema avanza al Estado '101000' para subir el elevador al piso 1 antes de ir al piso 3. De lo contrario, avanza al Estado '000100' para revisar el sensor del piso 1, porque si el elevador está en el piso 1, se puede ir al piso 2 antes de ir al piso 3.

Estado ‘000100’ – BP3SP1

En este estado si el sensor (S_P1), detecta al elevador en el piso 1, el sistema avanza al Estado ‘000101’ para revisar si está encendido el led del botón de piso 2 o el led del botón para subir desde el piso 2. De lo contrario, avanza al Estado ‘110110’ para que elevador se dirija al piso 3 directamente.

Estado ‘000101’ – BP2BSP2

En este estado, si está encendido el led del botón de piso 2 (B_P2) o el led del botón para subir desde el piso 2 (B_SP2), el sistema avanza al Estado ‘110010’ para subir el elevador al piso 2 antes de ir al piso 3. De lo contrario, avanza al Estado ‘000110’ para poder hacer una carga y así poder subir el elevador al piso 3 directamente.

Estado ‘000110’ – AUX1

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a avanzar al Estado ‘110110’, haciendo una carga, para que elevador se dirija al piso 3 directamente.

Estado ‘000111’ – BOTPB

En este estado si está encendido el led del botón de planta baja (B_PB), el sistema avanza al Estado ‘001000’ para revisar el sensor del piso 3, dado que si el elevador está el piso 3 se puede ir al piso 2 antes de ir a la planta baja. De lo contrario, avanza al Estado ‘001101’ para revisar si está encendido el led del botón del piso 1.

Estado ‘001000’ – BPBSP3

En este estado si el sensor (S_P3), detecta que el elevador está en el piso 3, el sistema avanza al Estado ‘001001’ para verificar si está encendido el led del botón del piso 2 o el led del botón para bajar desde el piso 2. De lo contrario, avanza al Estado ‘001010’ para revisar el sensor del piso 2, porque si el elevador está en el piso 2, se puede ir al piso 1 antes de ir a la planta baja.

Estado ‘001001’ – BP2BBP2

En este estado si está encendido el led del botón del piso 2 (B_P2) o el led del botón para bajar desde el piso 2 (B_BP2), el sistema avanza al Estado ‘101110’ para bajar el elevador al piso 2 antes de ir a la planta baja. De lo contrario, avanza al Estado ‘001010’ para revisar el sensor del piso 2, porque si el elevador está en el piso 2, se puede ir al piso 1 antes de ir a la planta baja.

Estado ‘001010’ – BPBSP2

En este estado si el sensor (S_P2), detecta el elevador en el piso 2, el sistema avanza al Estado ‘001011’ para revisar si está encendido el led del botón del piso 1 o el led del botón para bajar desde el piso 1. De lo contrario, avanza al Estado ‘011101’ para ir a la planta baja directamente, ya que el elevador se encuentra en el piso 1 o en la planta baja.

Estado ‘001011’ – BP1BBP1

En este estado si está encendido el led del botón del piso 1 o el led del botón para bajar desde el piso 1, el sistema avanza al Estado ‘100100’ para bajar el elevador al piso 1 antes de ir a la planta baja. De lo contrario, avanza al Estado ‘001100’ para poder hacer una carga y así poder bajar el elevador a la planta baja directamente.

Estado ‘001100’ – AUX2

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a avanzar al Estado ‘011101’, haciendo una carga, para que elevador se dirija a la planta baja directamente.

Estado ‘001101’ – BOTP1

En este estado si está encendido el led del botón del piso 1 (B_P1), el sistema avanza al Estado ‘100010’ para que el elevador se desplace al piso 1. De lo contrario, avanza al Estado ‘001110’ para revisar el led del botón del piso 2.

Estado ‘001110’ – BOTP2

En este estado si está encendido el led del botón del piso 2 (B_P2), el sistema avanza al Estado ‘101100’ para que el elevador se desplace al piso 2. De lo contrario, avanza al Estado ‘001111’ para revisar el led del botón para subir desde planta baja.

Estado ‘001111’ – BOTSPB

En este estado si está encendido el led del botón para subir desde la planta baja (B_SPB), el sistema avanza al Estado ‘011101’ para que el elevador se desplace a la planta baja. De lo contrario, avanza al Estado ‘010000’ para revisar el led de los botones para bajar desde el piso 1 y para bajar desde el piso 1.

Estado ‘010000’ – SBP1

En este estado, si está encendido el led del botón para subir desde el piso 1 (B_SP1) o el led del botón para bajar desde el piso 1 (B_BP1), el sistema avanza al estado ‘100010’ para que el elevador se desplace al piso 1. De lo contrario, avanza al Estado ‘010001’ para revisar el led de los botones para subir desde el piso 2 o para bajar desde el piso 2.

Estado ‘010001’ – SBP2

En este estado, si está encendido el led del botón para subir desde el piso 2 (B_SP2) o el led del botón para bajar desde el piso 2 (B_BP2), el sistema avanza al Estado ‘101100’ para que el elevador se desplace al piso 2. De lo contrario, avanza al Estado ‘010010’ para revisar el led del botón para bajar desde el piso 3.

Estado ‘010010’ – BOTBP3

En este estado si está encendido el led del botón para bajar desde el piso 3 (B_BP3), el sistema avanza al estado ‘110110’ para que el elevador se desplace al piso 3. De lo contrario, avanza al Estado ‘010011’ para posteriormente hacer una carga y así esperar a que se oprima un botón o varios botones.

Estado ‘010011’ – Auxiliar 3

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘000000’, haciendo una carga, para estar a la espera de que se oprima un botón.

Estado ‘010100’ – Abre puerta

En este estado se activa la señal (M_A) para que se abra la puerta, permitiendo el ingreso de personas. Cuando el sensor (S_PA), detecta que la puerta se ha abierto completamente, el sistema avanza al Estado ‘010101’ para esperar a que ingresen las personas al elevador. De lo contrario, permanece en el Estado ‘010100’ abriendo la puerta.

Estado ‘010101’ – ESPERA

En este estado el temporizador empieza a contar o continúa contando 5 segundos sin necesidad de que sea activado por medio de una señal.

La puerta se encuentra abierta, en espera de que ingresen las personas al elevador. Cuando se detecta la señal T5 debido a que pasaron 5 segundos, avanza al Estado ’011010’ para cerrar la puerta. De lo contrario, avanza al Estado ‘010110’ para revisar el sensor de obstrucción.

Estado ‘010110’ – OBST

En este estado el temporizador continúa contando 5 segundos y la puerta permanece abierta, en espera de que ingresen personas al elevador. Cuando el sensor (S_OBS), detecta algo que obstruye el paso de la puerta, el sistema avanza al Estado ‘011001’ para reiniciar el temporizador. De lo contrario, avanza al Estado ‘010111’ para revisar si el botón para cerrar la puerta esta presionado.

Estado ‘010111’ – BOTC

En este estado el temporizador continúa contando 5 segundos y la puerta permanece abierta, en espera de que ingresen personas al elevador. Cuando se detecta que el botón (B_CP) esta presionado, el sistema avanza al Estado ’011010’ para cerrar la puerta. De lo contrario, avanza al Estado ’011000’ para posteriormente seguir contando 5 segundos, en espera de que ingresen todas las personas al elevador.

Estado ‘011000’ – AUX4

En este estado el temporizador continúa contando 5 segundos y la puerta permanece abierta, en espera de que ingresen las personas al elevador. En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘010101’ para que el temporizador siga contando 5 segundos.

Estado ‘011001’ – RESET

En este estado se reinicia la cuenta del temporizador debido a que se detectó una obstrucción. En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada regresar al Estado ‘010101’ para que el temporizador se reinicie y empiece a contar 5 segundos nuevamente.

Estado ‘011010’ – CIERRA

En este estado deja de contar el temporizador y se reinicia. Se activa la señal (M_C), para que se cierre la puerta. Si el sensor de puerta cerrada (S_PC), detecta que la puerta se ha cerrado completamente, el sistema regresa al Estado ‘000000’ para verificar si se presionó algún botón o hay algún led de un botón encendido. De lo contrario, avanza al Estado ‘011011’ para revisar el sensor de obstrucción y el botón para abrir la puerta del elevador.

Estado ‘011011’ – BOTOBS

En este estado permanece activa la señal (M_C) para que la puerta siga cerrándose. Si el sensor (S_OBS), no detecta un obstáculo en el paso de la puerta o no se presiona el botón (B_AP) para abrir la puerta, el sistema regresa al Estado ‘011010’ para continuar cerrando la puerta del elevador. De lo contrario, avanza al Estado ‘011100’ para que posteriormente se pueda hacer una carga al Estado ‘010100’ para abrir la puerta.

Estado ‘011100’ – AUX5

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘010100’, haciendo una carga para que se abra la puerta e inicie el proceso de la puerta nuevamente.

Estado ‘011101’ – DETPB

En este estado se verifica si el elevador se encuentra en la planta baja. Si el sensor (S_PB), detecta que el elevador se encuentra en la planta baja, el sistema regresa al Estado ‘010100’ para abrir la puerta. De lo contrario, avanza al Estado ‘011110’ para bajar el elevador ya que se encuentra en alguno de los pisos superiores.

Estado ‘011110’ – BAJAPB

En este estado se activa la señal (M_B) para que el elevador baje. Si el sensor (S_PB), detecta que el elevador se encuentra en la planta baja, el sistema regresa al Estado ‘010100’ para abrir la puerta. De lo contrario, avanza al Estado ‘011111’ para revisar el botón de paro del elevador.

Estado ‘011111’ – STPPB

En este estado se mantiene activa la señal (M_B) para que el elevador baje. Si el botón (B_STP) no está presionado, el sistema regresa al Estado ‘011110’ para continuar bajando el elevador. De lo contrario, avanza al Estado ‘100000’ para detener el elevador.

Estado ‘100000’ – STOPPB

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no está presionado, el sistema avanza al Estado ‘100001’ para que posteriormente se pueda hacer una carga al Estado ‘011110’ y por lo tanto seguir bajando el elevador. De lo contrario, el elevador permanece en el Estado ’100000’ detenido.

Estado ‘100001’ – AUX6

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘011110’, haciendo una carga para seguir bajando elevador.

Estado ‘100010’ – DETP1

En este estado se verifica si el elevador se encuentra en el piso 1. Si el sensor (S_P1), detecta que el elevador se encuentra en el piso 1, el sistema regresa al Estado ‘010100’ para abrir la puerta. De lo contrario, avanza al Estado ‘100011’ para revisar el sensor de planta baja.

Estado ‘100011’ – DETSPB

Si el sensor (S_PB), no detecta el elevador en la planta baja, el sistema avanza al Estado ‘100100’ para bajar el elevador al piso 1 ya que se encuentra en alguno de los pisos superiores. De lo contrario, avanza al Estado ‘101000’ para subir el elevador al piso 1 debido a que se encuentra en la planta baja.

Estado ‘100100’ – BAJAP1

En este estado se activa la señal (M_B) para que el elevador baje. Si el sensor (S_P1) detecta que el elevador se encuentra en el piso 1, el sistema regresa al Estado ‘010100’ para abrir la puerta.

De lo contrario, avanza al Estado ‘100101’ para revisar el botón de paro del elevador.

Estado ‘100101’ – STPBP1

En este estado mantiene activa la señal (M_B), para que el elevador baje. Si el botón (B_STP), no está presionado, el sistema regresa al Estado ‘100100’ para seguir bajando el elevador. De lo contrario, avanza al Estado ‘100110’ para detener el elevador.

Estado ‘100110’ – STOPBP1

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no está presionado, el sistema avanza al Estado ‘100111’ para posteriormente poder hacer una carga y así seguir bajando el elevador. De lo contrario, el elevador permanece detenido en el Estado ’100110’.

Estado ‘100111’ – AUX7

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘100100’, haciendo una carga para seguir bajando elevador.

Estado ‘101000’ – SUBEP1

En este estado se activa la señal (M_S) para que el elevador suba. Si el sensor (S_P1), detecta que el elevador se encuentra en el piso 1, el sistema regresa al Estado ‘010100’ para abrir la puerta. De lo contrario, avanza al Estado ‘101001’ para revisar el botón de paro del elevador.

Estado ‘101001’ – STPSP1

En este estado se mantiene activa la señal (M_S) para que el elevador suba. Si el botón (B_STP), no está presionado, el sistema regresa al Estado ‘101000’ para seguir subiendo el elevador. De lo contrario, avanza al Estado ‘101010’ para detener el elevador.

Estado ‘101010’ – STOPSP1

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no esté presionado, el sistema avanza al Estado ‘101011’ para posteriormente poder hacer una carga y así seguir subiendo el elevador. De lo contrario, el elevador permanece detenido en el Estado ‘101010’.

Estado ‘101011’ – AUX8

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘101000’, haciendo una carga para seguir subiendo elevador.

Estado ‘101100’ – DETP2

En este estado si el sensor (S_P2) detecta que el elevador se encuentra en el piso 2, el sistema regresa al Estado ‘010100’ para abrir la puerta. De lo contrario, avanza al Estado ‘101101’ para revisar el sensor del piso 3.

Estado ‘101101’ – DETSP3

En este estado si el sensor (S_P3), detecta el elevador en el piso 3, el sistema avanza al Estado ‘101110’ para bajar el elevador al piso 2 ya que se encuentra en el piso superior. De lo contrario, avanza al Estado ‘110010’ para subir el elevador al piso 2 debido a que se encuentra en alguno de los pisos inferiores.

Estado ‘101110’ – BAJAP2

En este estado se activa la señal (M_B) para que el elevador baje. Si el sensor (S_P2), detecta que el elevador se encuentra en el piso 2, el sistema regresa al Estado ‘010100’ para abrir la puerta. De lo contrario, avanza al Estado ‘101111’ para revisar el botón de paro del elevador.

Estado ‘101111’ – STPBP2

En este estado mantiene activa la señal (M_B) para que el elevador baje. Si el botón (B_STP), no está presionado, el sistema regresa al Estado ‘101110’ para seguir bajando el elevador. De lo contrario, avanza al Estado ‘110000’ para detener el elevador.

Estado ‘110000’ – STOPBP2

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no esta presionado, el sistema avanza al Estado ‘110001’ para posteriormente poder hacer una carga y así seguir bajando el elevador). De lo contrario, el elevador permanece detenido en el Estado ’110000’.

Estado ‘110001’ – AUX9

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a avanzar al Estado ‘101110’, haciendo una carga para seguir bajando elevador.

Estado ‘110010’ – SUBEP2

En este estado se activa la señal (M_S) para que el elevador suba. Si el sensor (S_P2), detecta que el elevador se encuentra en el piso 2, el sistema regresa al Estado ‘010100’ para abrir la puerta. De lo contrario, avanza al Estado ‘110011’ para revisar el botón de paro del elevador.

Estado ‘110011’ – STPSP2

En este estado se mantiene activa la señal (M_S) para que el elevador suba. Si el botón (B_STP) no está presionado, el sistema regresa al Estado ‘110010’ para seguir subiendo el elevador. De lo contrario, avanza al Estado ‘110100’ para detener el elevador.

Estado ‘110100’ – STOPSP2

En este estado el elevador se encuentra detenido. Si el botón (B_STP) no está presionado, el sistema avanza al Estado ‘110101’ para posteriormente poder hacer una carga y así seguir subiendo el elevador. De lo contrario, el elevador permanece detenido en el Estado ’110100’.

Estado ‘110101’ – Auxiliar 10

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘110010’, haciendo una carga para seguir subiendo elevado.

Estado ‘110110’ – DETP3

En este estado se verifica si el elevador se encuentra en el piso 3. Si el sensor (S_P3), detecta que el elevador se encuentra en el piso 3, el sistema regresa al Estado ‘010100’ para abrir la puerta. De lo contrario, avanza al Estado ‘110111’ para subir el elevador ya que se encuentra en alguno de los pisos inferiores.

Estado ‘110111’ – SUBEP3

En este estado se activa la señal (M_S) para que el elevador suba. Si el sensor (S_P3), detecta que el elevador se encuentra en el piso 3, el sistema regresa al Estado ‘010100’ para abrir la puerta. De lo contrario, avanza al Estado ‘111000’ para revisar el botón de paro del elevador.

Estado ‘111000’ – STPSP3

En este estado se mantiene activa la señal (M_S) para que el elevador suba. Si el botón (B_STP), no está presionado, el sistema regresa al Estado ‘110111’ para seguir subiendo el elevador. De lo contrario, avanza al Estado ‘111001’ para detener el elevador.

Estado ‘1111001’ – STOPSP3

En este estado el elevador se encuentra detenido. Si el botón (B_STP), no esté presionado, el sistema regresa al Estado ‘111010’ para posteriormente poder hacer una carga y así seguir subiendo el elevador. De lo contrario, el elevador permanece en el Estado ‘111001’ deteniendo el elevador.

Estado ‘111010’ – Auxiliar 11

En este estado no hay variable de entrada. Se utiliza la variable auxiliar AUX, de manera que la máquina de estados es forzada a regresar al Estado ‘110111’, haciendo una carga para seguir subiendo elevador.

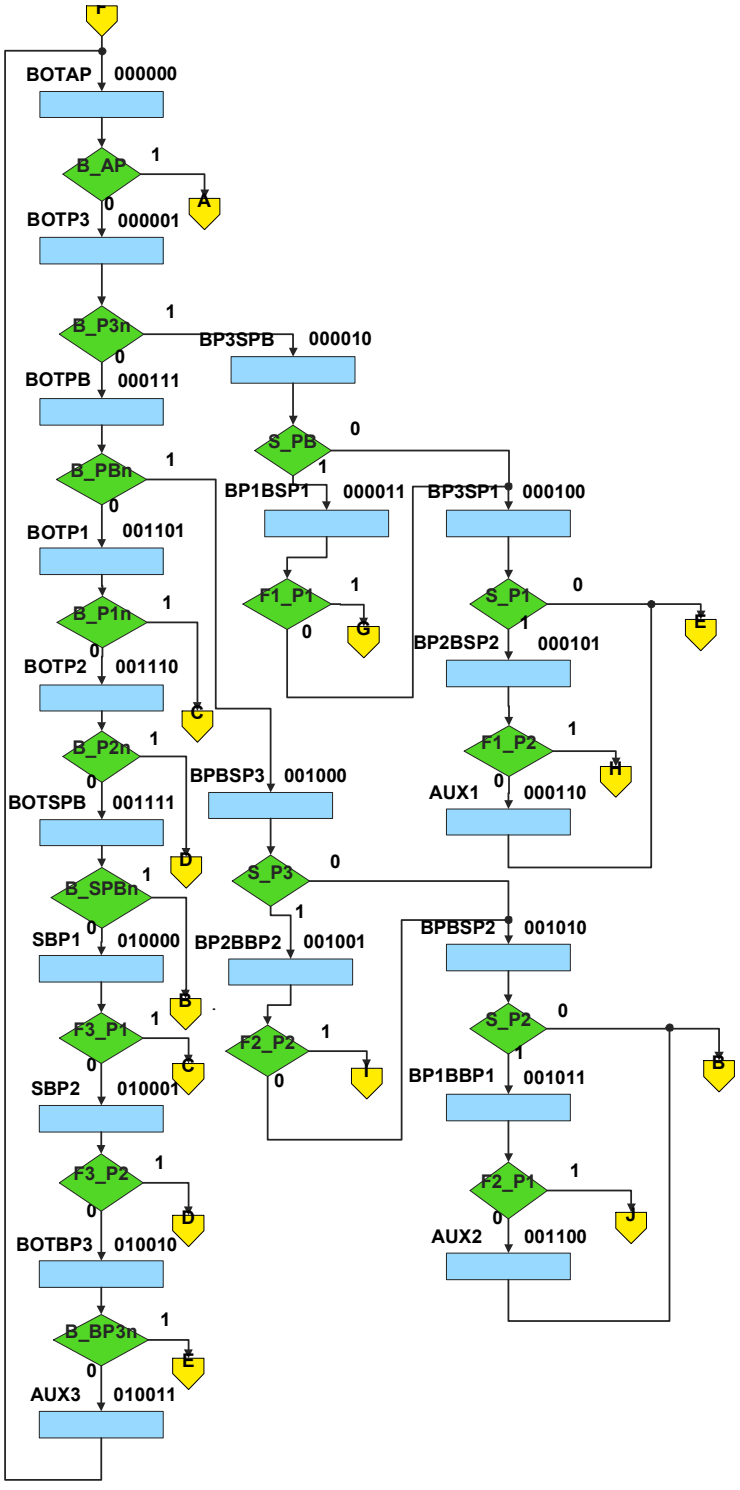


Figura P22.7 Carta ASM del sistema del elevador parte 1.

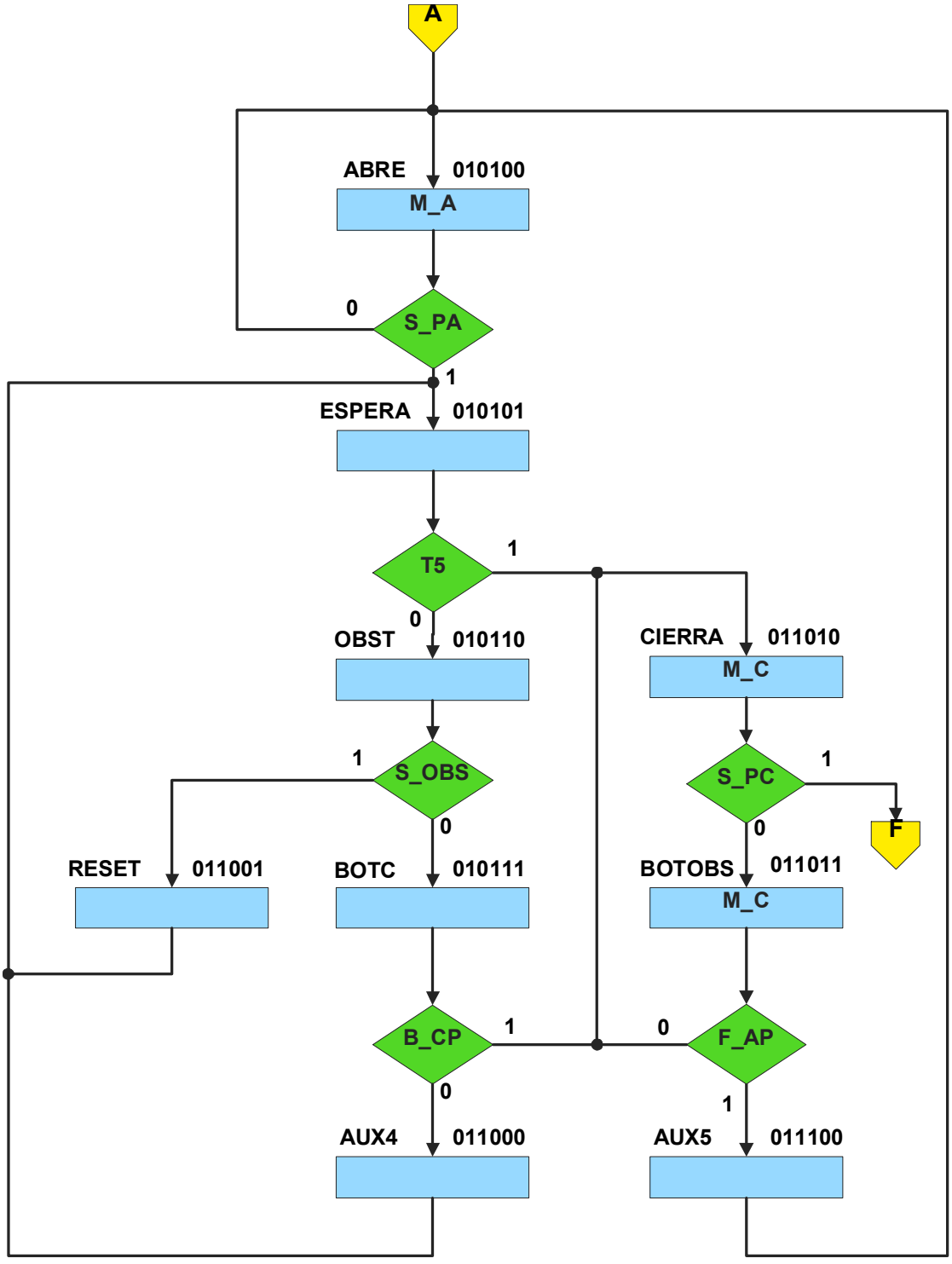


Figura P22.8 Carta ASM del sistema del elevador parte 2.

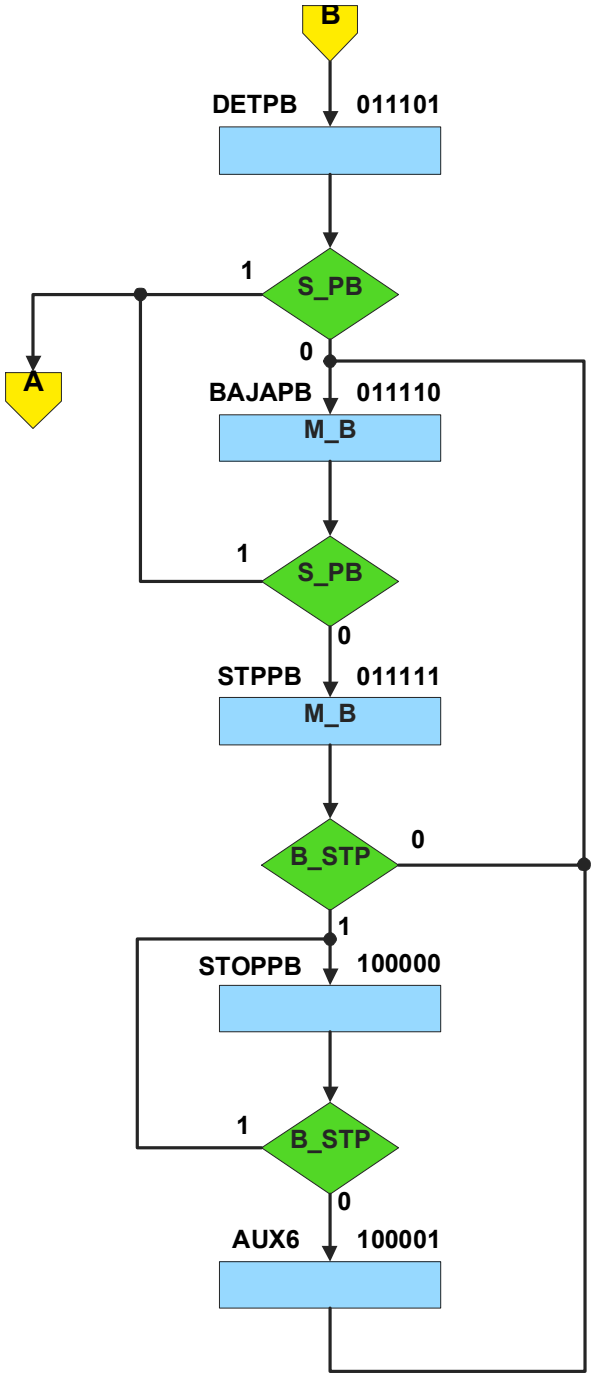


Figura P22.9 Carta ASM del sistema del elevador parte 3.

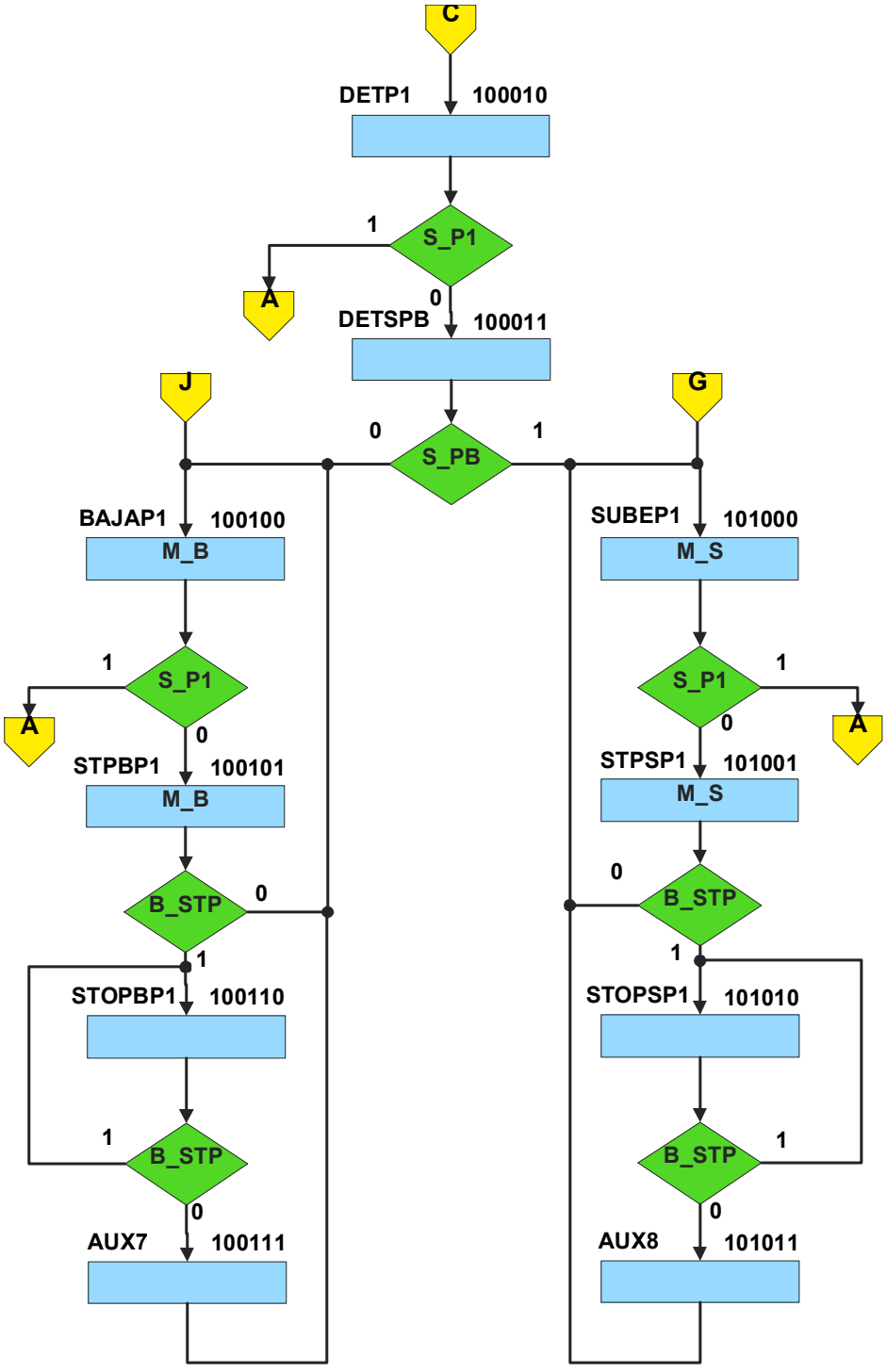


Figura P22.10 Carta ASM del sistema del elevador parte 4.

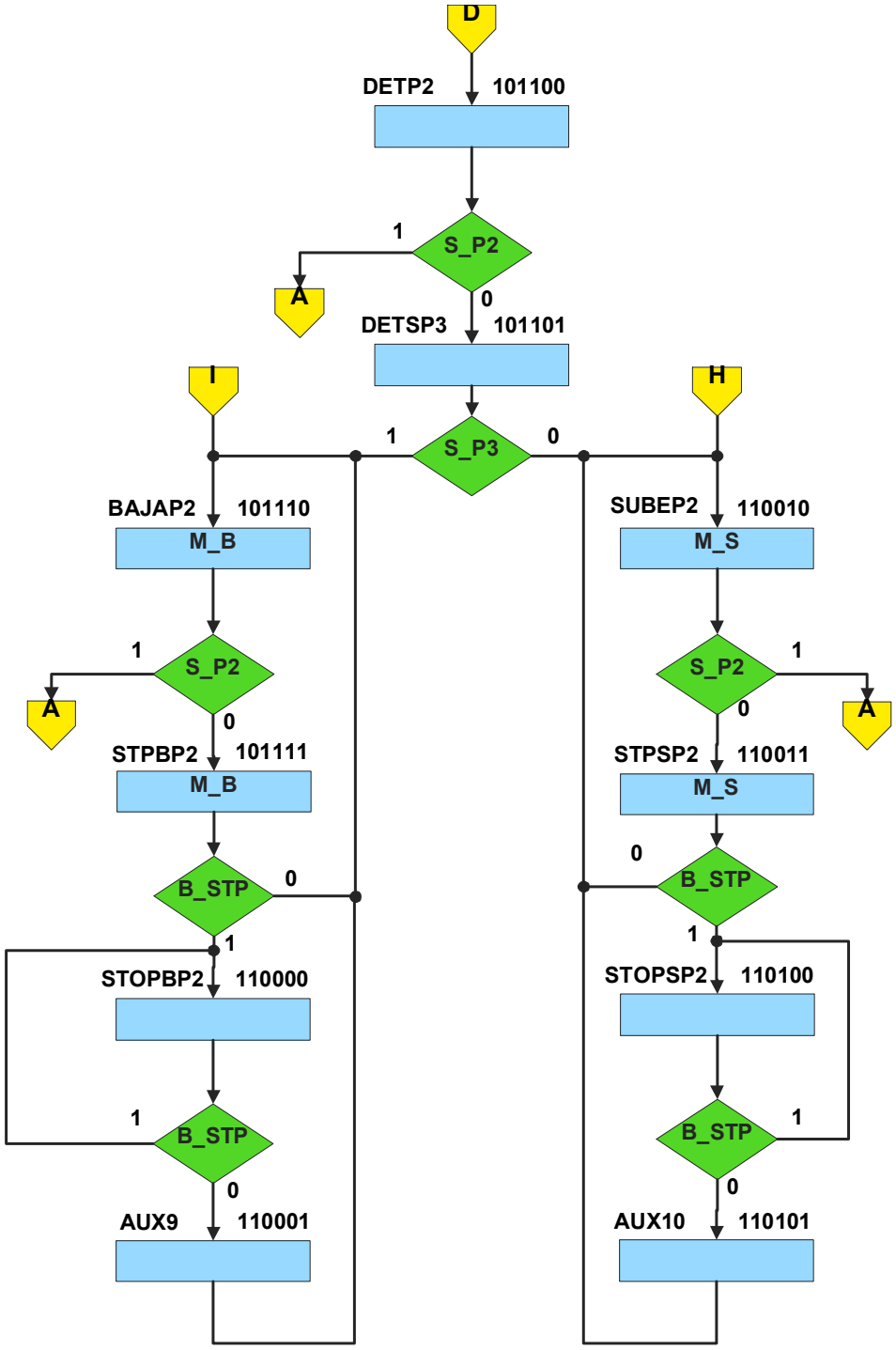


Figura P22.11 Carta ASM del sistema del elevador parte 5.

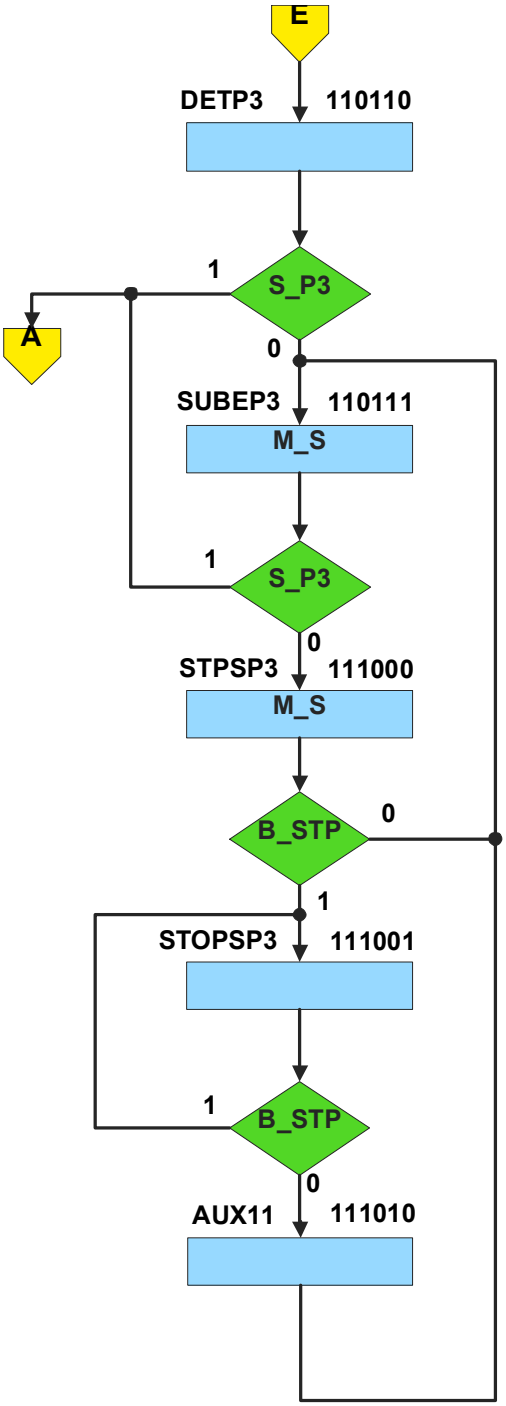


Figura P22.12 Carta ASM del sistema del elevador parte 6.

Solución

Se relacionan las máquinas de estado del elevador y del registro de los botones presionados, por medio de las señales de salida de los leds de los botones presionados.

Para poder guardar un registro de los botones presionados se utilizan leds. De manera que, si el botón de un piso es presionado, se enciende un led asociado a éste. El led sólo se apaga cuando llegué al piso que indicado por el botón y cuando se abre la puerta del elevador.

Se realiza una máquina de estados para cada botón asociado con un piso. Las máquinas de estados de los botones y del sistema del elevador funcionan conjuntamente. Es decir, las máquinas de estados de este sistema están relacionadas o vinculadas entre sí.

Las máquinas de estados se pueden relacionar por medio de las salidas, por lo tanto, la señal de salida de la puerta M_A del sistema del elevador, es igual a la señal de entrada M_An del control de los botones. De igual manera, se relacionan los leds del registro de botones, con el control del elevador. La carta ASM de cada botón se muestran en la figura P22.13.

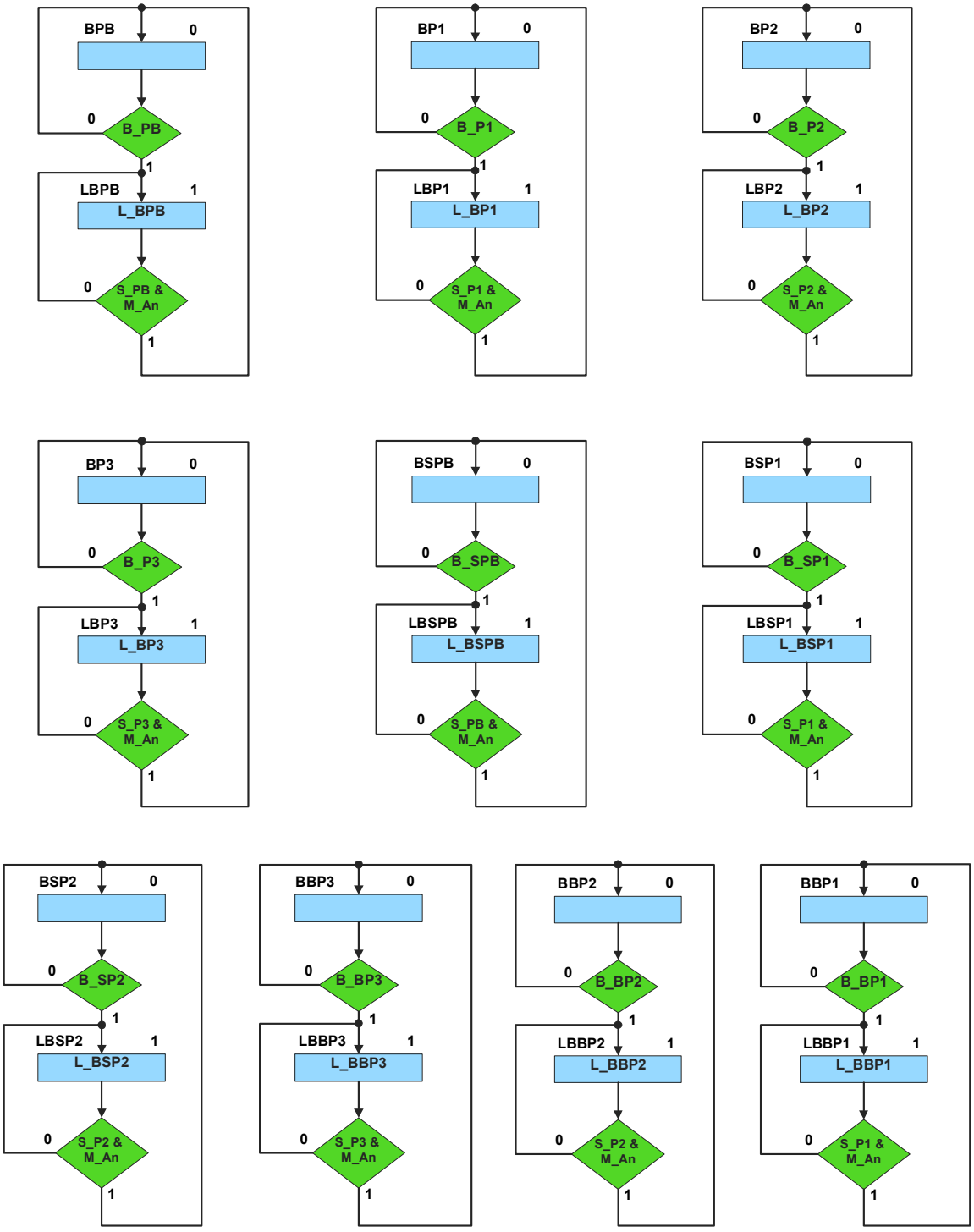


Figura P22.13 Cartas ASM para guardar registro de los botones presionados.

A continuación, se resuelve por el método de variable suscrita las máquinas de estado para el registro de los botones usando leds (ver figura P22.7). El mapa de Karnaugh general de transición de estados para el botón B_PB se muestra en la tabla P22.3.

Tabla P22.3 Mapa de Karnaugh general de transición de estados para el led del botón B_PB.

	J	
	0	1
B_PB	0	1
	1	S_PB&M_An 0

Mapa de Karnaugh particular para el bit J:

	J	
	0	1
	B_PB	!(S_PB&M_An)

```
Jn = !J&B_PB | J&!(S_PB&M_An);
```

Para el led del botón se utiliza el mismo valor del bit J, debido a que el led L_BPB sólo se activa cuando el bit J está en el Estado ‘1’.

Considerando que las cartas ASM para los demás botones son iguales. Las expresiones para los demás botones quedan de la siguiente manera:

```
Kn = !K&B_P1 | K&!(S_P1&M_An);

Ln = !L&B_P2 | L&!(S_P2&M_An);

Mn = !M&B_P3 | M&!(S_P3&M_An);

Nn = !N&B_SPB | N&!(S_PB&M_An);

Npn = !Np&B_SP1 | Np&!(S_P1&M_An);

On = !O&B_SP2 | O&!(S_P2&M_An);

Pn = !P&B_BP3 | P&!(S_P3&M_An);

Qn = !Q&B_BP2 | Q&!(S_P2&M_An);

Rn = !R&B_BP1 | R&!(S_P1&M_An);
```

Para el led de los botones se utiliza el mismo valor de los bits correspondientes, debido a que el led L_BPB sólo se activa cuando el bit que le corresponde a cada botón esté en el Estado ‘1’.

Para implementar el controlador del elevador por medio del direccionamiento implícito, se deben emplear las expresiones lógicas del temporizador de 5 segundos, obtenidas en la resolución de la Práctica 21.

Para el sistema del elevador se debe asignar una representación binaria a cada variable de entrada (ver tabla P22.4).

Donde:

```
F1_P1 =  B_P1n | B_SP1n;

F1_P2 = B_P2n | B_SP2n;

F2_P1 = B_P1n | B_BP1n;

F2_P2 = B_P2n | B_BP2n;

F3_P1 = B_SP1n | B_BP1n;

F3_P2 = B_SP2n | B_BP2n;

F_AP = S_OBS | B_AP;
```

Tabla P22.4 Representación binaria de entradas para el sistema del elevador.

Entrada	Prueba
AUX	00000
S_PB	00001
S_P1	00010
S_P2	00011
S_P3	00100
B_PBn	00101
B_P1n	00110
B_P2n	00111
B_P3n	01000
F1_P1	01001
F1_P2	01010
F2_P2	01011
F1_P1	01100
B_SPBn	01101
F3_P1	01110
F3_P2	01111
B_BP3n	10000
B_AP	10001
B_CP	10010
F_AP	10011
S_OBS	10100
T5	10101
S_PC	10110
S_PA	10111
B_STP	11000

Se debe llenar la tabla P22.5 y tabla P22.6 con base en la información de la carta ASM de la figura P22.7 a la figura P22.12 figura P21.6, usando el método de diseño con memoria y direccionamiento implícito.

A continuación, se describe cómo llenar los campos de la memoria para el Estado ‘000000’.

En el Estado ‘000000’ se selecciona la entrada B_AP, por lo tanto, se coloca en el campo de prueba su representación binaria, es decir, ‘10001’. Si B_AP es igual a uno, entonces el estado siguiente es el Estado ‘010100’, su representación binaria ‘010100’ es colocada en el campo de la liga, ya que se requiere hacer una carga.

El campo VF es igual uno, ya que, para hacer una carga en el contador, el valor de la entrada y de VF deben ser iguales. En el Estado ‘000000’ no hay señales de salida activadas, por lo que se coloca un ‘0’ en la parte de salidas.

Para los campos de los demás estados se procede de la misma manera.

De acuerdo con las entradas (dirección de memoria), la memoria proporciona salidas (contenido de memoria) (ver figura P22.4). Los valores hexadecimales indicados en la tabla son la conversión de base binaria a base hexadecimal para el contenido de una memoria de 8 bits. Con los valores hexadecimales se genera un archivo con extensión “HEX” por medio de un programa editor de memorias.

Tabla P22.5 Contenido de la memoria para el sistema del elevador parte 1.

Dirección de memoria						Contenido de memoria																	
Estado presente						Prueba					VF	Liga		Liga				Salidas				Hex 1	Hex 2
QA	QB	QC	QD	QE	QF	I4	I3	I2	I1	I0		L5	L4	L3	L2	L1	L0	M_A	M_C	M_S	M_B		
0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	8D	40
0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	40	70
0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	08	40
0	0	0	0	1	1	0	1	0	0	1	1	1	0	1	0	0	0	0	0	0	0	4E	80
0	0	0	1	0	0	0	0	0	1	0	0	1	1	0	1	1	0	0	0	0	0	13	60
0	0	0	1	0	1	0	1	0	1	0	1	1	1	0	0	1	0	0	0	0	0	57	20
0	0	0	1	1	0	0	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	07	60
0	0	0	1	1	1	0	0	1	0	1	0	0	0	1	1	0	1	0	0	0	0	28	D0
0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	20	A0
0	0	1	0	0	1	0	1	0	1	1	1	1	0	1	1	1	0	0	0	0	0	5E	E0
0	0	1	0	1	0	0	0	0	1	1	0	0	1	1	1	0	1	0	0	0	0	19	D0
0	0	1	0	1	1	0	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	66	40
0	0	1	1	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0	0	0	0	05	D0
0	0	1	1	0	1	0	0	1	1	0	1	1	0	0	0	1	0	0	0	0	0	36	20
0	0	1	1	1	0	0	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	3E	C0
0	0	1	1	1	1	0	1	1	0	1	1	0	1	1	1	0	1	0	0	0	0	6D	D0
0	1	0	0	0	0	0	1	1	1	0	1	1	0	0	0	1	0	0	0	0	0	76	20
0	1	0	0	0	1	0	1	1	1	1	1	1	0	1	1	0	0	0	0	0	0	7E	C0
0	1	0	0	1	0	1	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	87	60
0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	04	00
0	1	0	1	0	0	1	0	1	1	1	0	0	1	0	1	0	0	1	0	0	0	B9	48
0	1	0	1	0	1	1	0	1	0	1	1	0	1	1	0	1	0	0	0	0	0	AD	A0
0	1	0	1	1	0	1	0	1	0	0	1	0	1	1	0	0	1	0	0	0	0	A5	90
0	1	0	1	1	1	1	0	0	1	0	1	0	1	1	0	1	0	0	0	0	0	95	A0
0	1	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	05	50
0	1	1	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	05	50
0	1	1	0	1	0	1	0	1	1	0	1	0	0	0	0	0	0	0	1	0	0	B4	04
0	1	1	0	1	1	1	0	0	1	1	0	0	1	1	0	1	0	0	1	0	0	99	A4

Tabla P22.6 Contenido de la memoria para el sistema del elevador parte 2.

Dirección de memoria						Contenido de memoria																	
Estado presente						Prueba					VF	Liga		Liga				Salidas				Hex 1	Hex 2
QA	QB	QC	QD	QE	QF	I4	I3	I2	I1	I0		L5	L4	L3	L2	L1	L0	M_A	M_C	M_S	M_B		
0	1	1	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	05	40
0	1	1	1	0	1	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0D	40
0	1	1	1	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	1	0D	41
0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	C1	E1
1	0	0	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	C6	00
1	0	0	0	0	1	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0	0	05	E0
1	0	0	0	1	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	0	15	40
1	0	0	0	1	1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0E	80
1	0	0	1	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	0	1	15	41
1	0	0	1	0	1	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	1	C2	41
1	0	0	1	1	0	1	1	0	0	0	1	1	0	0	1	1	0	0	0	0	0	C6	60
1	0	0	1	1	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	06	40
1	0	1	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	0	1	0	15	42
1	0	1	0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	1	0	C2	82
1	0	1	0	1	0	1	1	0	0	0	1	1	0	1	0	1	0	0	0	0	0	C6	A0
1	0	1	0	1	1	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	06	80
1	0	1	1	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	0	1D	40
1	0	1	1	0	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	23	20
1	0	1	1	1	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	1	1D	41
1	0	1	1	1	1	1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	1	C2	E1
1	1	0	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	C7	00
1	1	0	0	0	1	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	06	E0
1	1	0	0	1	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0	1	0	1D	42
1	1	0	0	1	1	1	1	0	0	0	0	1	1	0	0	1	0	0	0	1	0	C3	22
1	1	0	1	0	0	1	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	C7	40
1	1	0	1	0	1	0	0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	07	20
1	1	0	1	1	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	25	40
1	1	0	1	1	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	1	0	25	42
1	1	1	0	0	0	1	1	0	0	0	0	1	1	0	1	1	1	0	0	1	0	C3	72
1	1	1	0	0	1	1	1	0	0	0	1	1	1	1	0	0	1	0	0	0	0	C7	90
1	1	1	0	1	0	0	0	0	0	0	1	1	1	0	1	1	1	0	0	0	0	07	70

Para poder utilizar las expresiones lógicas obtenidas en esta práctica, se debe usar el PIC16F1939 como PLD, esto se logra incluyendo en el programa el archivo de biblioteca “PLD.H”.

El selector de entradas es un multiplexor de treinta y dos líneas a una, éste es implementado por el PIC16F1939.

El multiplexor selecciona una de las entradas dependiendo de su representación binaria y dirige la información binaria de la entrada seleccionada a la salida de este (ver tabla P22.7).

Tabla P22.7 Tabla de funcionamiento del selector de entradas para la Práctica 22.

Prueba					SI
I4	I3	I2	I1	I0	
0	0	0	0	0	AUX
0	0	0	0	1	S_PB
0	0	0	1	0	S_P1
0	0	0	1	1	S_P2
0	0	1	0	0	S_P3
0	0	1	0	1	B_PBn
0	0	1	1	0	B_P1n
0	0	1	1	1	B_P2n
0	1	0	0	0	B_P3n
0	1	0	0	1	F1_P1
0	1	0	1	0	F1_P2
0	1	0	1	1	F2_P2
0	1	1	0	0	F1_P1
0	1	1	0	1	B_SPBn
0	1	1	1	0	F3_P1
0	1	1	1	1	F3_P2
1	0	0	0	0	B_BP3n
1	0	0	0	1	B_AP
1	0	0	1	0	B_CP
1	0	0	1	1	F_AP
1	0	1	0	0	S_OBS
1	0	1	0	1	T5
1	0	1	1	0	S_PC
1	0	1	1	1	S_PA
1	1	0	0	0	B_STP

Teniendo en cuenta que `T5 = F3&!F2&F1&!F0`, donde F3, F2, F1 y F0 son los bits del temporizador.

La función booleana del selector de entradas queda:

```
SI = AUX&!I4&!I3&!I2&!I1&!I0 | S_PB&!I4&!I3&!I2&!I1&I0 |
     S_P1&!I4&!I3&!I2&I1&!I0 | S_P2&!I4&!I3&!I2&I1&I0 |
     S_P3&!I4&!I3&I2&!I1&!I0 | B_PBn&!I4&!I3&I2&!I1&I0 |
     B_P1n&!I4&!I3&I2&I1&!I0 | B_P2n&!I4&!I3&I2&I1&I0 |
     B_P3n&!I4&I3&!I2&!I1&!I0 | F1_P1&!I4&I3&!I2&!I1&I0 |
     F1_P2&!I4&I3&!I2&I1&!I0 | F2_P2&!I4&I3&!I2&I1&I0 |
     F2_P1&!I4&I3&I2&!I1&!I0 | B_SPBn&!I4&I3&I2&!I1&I0 |
     F3_P1&!I4&I3&I2&I1&!I0 | F3_P2&!I4&I3&I2&I1&I0 |
     B_BP3n&I4&!I3&!I2&!I1&!I0 | B_AP&I4&!I3&!I2&!I1&I0 |
     B_CP&I4&!I3&!I2&I1&!I0 | F_AP&I4&!I3&!I2&I1&I0 |
     S_OBS&I4&!I3&I2&!I1&!I0 | T5&I4&!I3&I2&!I1&I0 | S_PC&I4&!I3&I2&I1&!I0 |
     S_PA&I4&!I3&I2&I1&I0 | B_STP&I4&I3&!I2&!I1&!I0;
```

Para obtener el valor de la lógica, se debe hacer la operación XOR entre el selector de entradas y el valor de VF (ver figura P22.6).

Por lo tanto, la función booleana de la lógica queda:

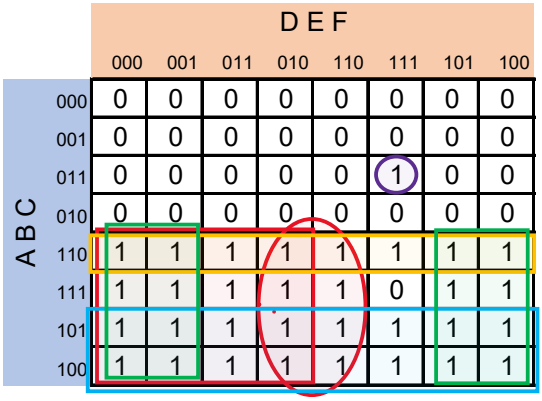
```
SL= VF ^ (AUX&!I4&!I3&!I2&!I1&!I0 | S_PB&!I4&!I3&!I2&!I1&I0 |
    S_P1&!I4&!I3&!I2&I1&!I0 | S_P2&!I4&!I3&!I2&I1&I0 |
    S_P3&!I4&!I3&I2&!I1&!I0 | B_PBn&!I4&!I3&I2&!I1&I0 |
    B_P1n&!I4&!I3&I2&I1&!I0 | B_P2n&!I4&!I3&I2&I1&I0 |
    B_P3n&!I4&I3&!I2&!I1&!I0 | F1_P1&!I4&I3&!I2&!I1&I0 |
    F1_P2&!I4&I3&!I2&I1&!I0 | F2_P2&!I4&I3&!I2&I1&I0 |
    F2_P1&!I4&I3&I2&!I1&!I0 | B_SPBn&!I4&I3&I2&!I1&I0 |
    F3_P1&!I4&I3&I2&I1&!I0 | F3_P2&!I4&I3&I2&I1&I0 |
    B_BP3n&I4&!I3&!I2&!I1&!I0 | B_AP&I4&!I3&!I2&!I1&I0 |
    B_CP&I4&!I3&!I2&I1&!I0 | F_AP&I4&!I3&!I2&I1&I0 |
    S_OBS&I4&!I3&I2&!I1&!I0 | T5&I4&!I3&I2&!I1&I0 | S_PC&I4&!I3&I2&I1&!I0 |
    S_PA&I4&!I3&I2&I1&I0 | B_STP&I4&I3&!I2&!I1&!I0);
```

Se utiliza un contador con carga paralela que indica que estado es el siguiente. El contador con carga paralela es implementado por el PIC16F1939. Si el valor a la salida de la lógica es igual a '1', el contador carga la información binaria de la liga a la memoria. Si el valor de la lógica es igual a '0', se cuenta al siguiente estado binario. A continuación, se obtienen las expresiones lógicas para un contador de seis bits por el método de variable suscrita (ver tabla P22.8).

Tabla P22.8 Mapa de Karnaugh general de transición de estados para un contador de seis bits.

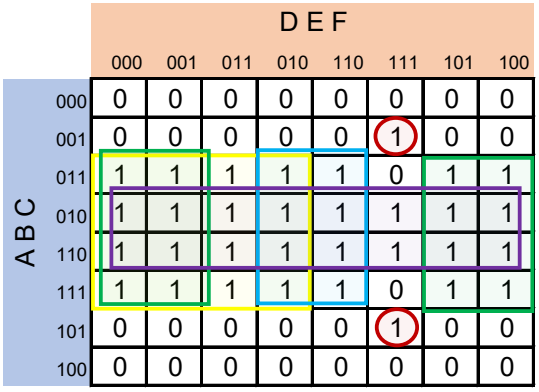
		D E F							
		0 0 0	0 0 1	0 1 1	0 1 0	1 1 0	1 1 1	1 0 1	1 0 0
A B C	0 0 0	000000	000001	000011	000010	000110	000111	000101	000100
		000001	000010	000100	000011	000111	001000	000110	000101
	0 0 1	001000	001001	001011	001010	001110	001111	001101	001100
		001001	001010	001100	001011	001111	010000	001110	001101
	0 1 1	011000	011001	011011	011010	011110	011111	011101	011100
		011001	011010	011100	011011	011111	100000	011110	011101
	0 1 0	010000	010001	010011	010010	010110	010111	010101	010100
		010001	010010	010100	010011	010111	011000	010110	010101
	1 1 0	110000	110001	110011	110010	110110	110111	110101	110100
		110001	110010	110100	110011	110111	111000	110110	110101
	1 1 1	111000	111001	111011	111010	111110	111111	111101	111100
		111001	111010	111100	111011	111111	000000	111110	111101
	1 0 1	101000	101001	101011	101010	101110	101111	101101	101100
		001001	001010	001100	001011	001111	010000	001110	001101
	1 0 0	100000	100001	100011	100010	100110	100111	100101	100100
		100001	100010	100100	100011	100111	101000	100110	100101

Mapas de Karnaugh particulares para los bits A, B, C, D, E y F respectivamente:



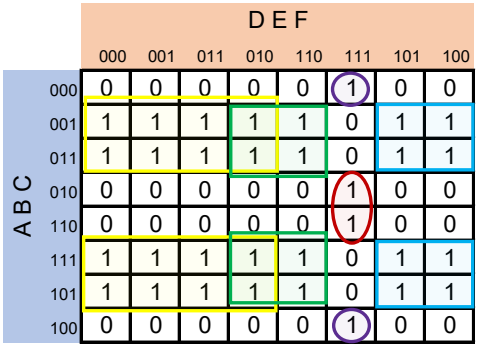
A B C \ D E F	000	001	011	010	110	111	101	100
000	0	0	0	0	0	0	0	0
001	0	0	0	0	0	0	0	0
011	0	0	0	0	0	1	0	0
010	0	0	0	0	0	0	0	0
110	1	1	1	1	1	1	1	1
111	1	1	1	1	1	0	1	1
101	1	1	1	1	1	1	1	1
100	1	1	1	1	1	1	1	1

```
At = A&E&!F | A&!D | A&!E | A&B&!C |
     A&!B | !A&B&C&D&E&F;
```



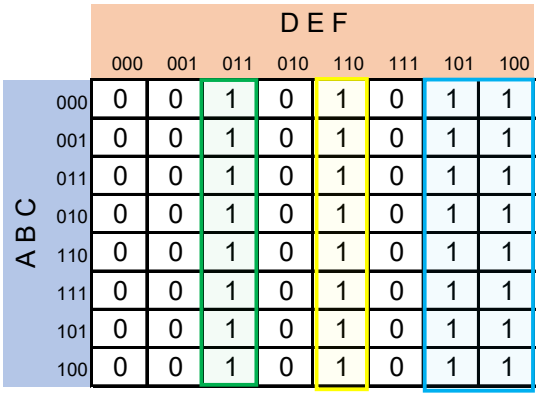
A B C \ D E F	000	001	011	010	110	111	101	100
000	0	0	0	0	0	0	0	0
001	0	0	0	0	0	1	0	0
011	1	1	1	1	1	0	1	1
010	1	1	1	1	1	1	1	1
110	1	1	1	1	1	1	1	1
111	1	1	1	1	1	0	1	1
101	0	0	0	0	0	1	0	0
100	0	0	0	0	0	0	0	0

```
Bt = B&!D | B&!E | B&E&!F | B&!C |
     !B&C&D&E&F;
```



A B C \ D E F	000	001	011	010	110	111	101	100
000	0	0	0	0	0	1	0	0
001	1	1	1	1	1	0	1	1
011	1	1	1	1	1	0	1	1
010	0	0	0	0	0	1	0	0
110	0	0	0	0	0	1	0	0
111	1	1	1	1	1	0	1	1
101	1	1	1	1	1	0	1	1
100	0	0	0	0	0	1	0	0

```
Ct = C&!D | C&E&!F | C&D&!E |
     B&!C&D&E&F | !B&!C&D&E&F;
```



A B C \ D E F	000	001	011	010	110	111	101	100
000	0	0	1	0	1	0	1	1
001	0	0	1	0	1	0	1	1
011	0	0	1	0	1	0	1	1
010	0	0	1	0	1	0	1	1
110	0	0	1	0	1	0	1	1
111	0	0	1	0	1	0	1	1
101	0	0	1	0	1	0	1	1
100	0	0	1	0	1	0	1	1

```
Dt = D&E&!F | !D&E&F |  D&!E;
```

ABC \ DEF	000	001	011	010	110	111	101	100
000	0	1	0	1	1	0	1	0
001	0	1	0	1	1	0	1	0
011	0	1	0	1	1	0	1	0
010	0	1	0	1	1	0	1	0
110	0	1	0	1	1	0	1	0
111	0	1	0	1	1	0	1	0
101	0	1	0	1	1	0	1	0
100	0	1	0	1	1	0	1	0

```
Et = E&!F | !E&F = E^F;
```

ABC \ DEF	000	001	011	010	110	111	101	100
000	1	0	0	1	1	0	0	1
001	1	0	0	1	1	0	0	1
011	1	0	0	1	1	0	0	1
010	1	0	0	1	1	0	0	1
110	1	0	0	1	1	0	0	1
111	1	0	0	1	1	0	0	1
101	1	0	0	1	1	0	0	1
100	1	0	0	1	1	0	0	1

```
Ft = !F;
```

Todas las expresiones de los bits del temporizador de 5 segundos realizán la operación AND con los términos: `!A&B&!C&D&!E&F | !A&B&!C&D&E&!F | !A&B&!C&D&E&F | !A&B&C&!D&!E&!F`. De este modo el temporizador sólo cuenta en los estados indicados de la carta ASM y en los demás estados el temporizador se reinicia y no cuenta.

Se utiliza un display de 7 segmentos para poder visualizar en que piso se encuentra el elevador. Por lo tanto, se utiliza un decodificador de BCD a 7 segmentos.

Se realiza una máquina de estados de manera que las entradas al decodificador son los bits de la máquina de estados. A continuación, se muestra la carta ASM para las entradas al decodificador (ver figura P22.14).

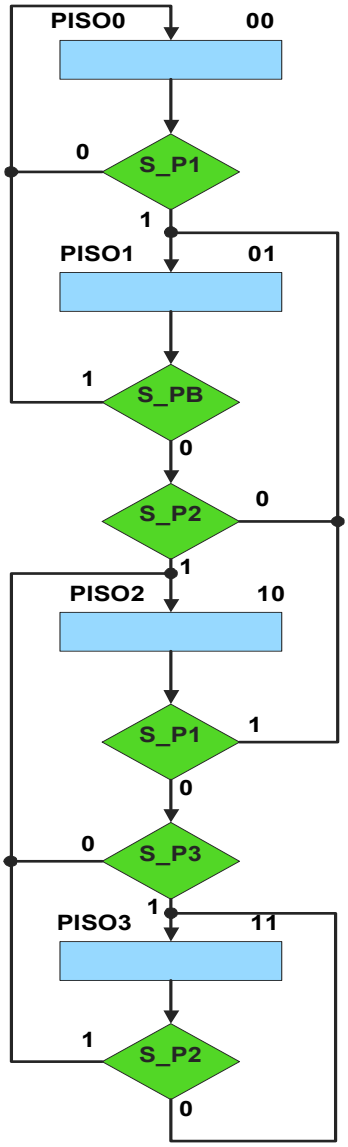


Figura P22.14 Cartas ASM para las entradas del decodificador del indicador de piso.

La máquina de estados para el indicador de pisos se resuelve por el método de variable suscrita. El mapa de Karnaugh general de transición de estados para las entradas al decodificador del indicador de piso se muestra en la tabla P22.9.

Tabla P22.9 Mapa de Karnaugh general de transición de estados para las entradas al decodificador del indicador de piso.

S \ T	0	1
	00	01
0	S_P1 01	S_PB 00 S_P2 10
	10	11
1	S_P1 01 S_P3 11	S_P2 10

Mapa de Karnaugh particular para el bit S y T respectivamente:

S \ T	0	1
0	0	S_P2
1	!S_P1	1

```
Sn = S&T | !S&T&S_P2 | S&!T&!S_P1;
```

S \ T	0	1
0	S_P1	!(S_P2|S_PB)
1	S_P3|S_P1	!S_P2

```
Tn = !S&!T&S_P1 | !S&T&!(S_P2|S_PB) |
     S&!T&(S_P3 | S_P1) | S&T&!S_P2;
```

Diagrama de conexiones

Se verifica el funcionamiento del controlador por medio de un simulador. Debido a la cantidad de entradas y salidas que se necesitan se utilizan tres PIC 16F1939 que están relacionados entre sí. En el PIC 1 está el programa encargado del registro de los botones presionados por medio de leds y las expresiones de las entradas al decodificador del indicador de piso. En el PIC 2 está el programa que controla la salida lógica del controlador. Para el PIC 2 se modificó la biblioteca PLD.h para poder tener más puertos de entradas.

En el PIC 3 está el programa que controla las principales expresiones lógicas para utilizar el direccionamiento implícito, las operaciones del temporizador y las expresiones de las entradas al decodificador del temporizador. Se carga en el controlador el archivo con extensión “HEX” de las memorias y los de extensión “COF” o “HEX” de los PIC 16F1939 (ver figura P22.15 a la figura P22.17).

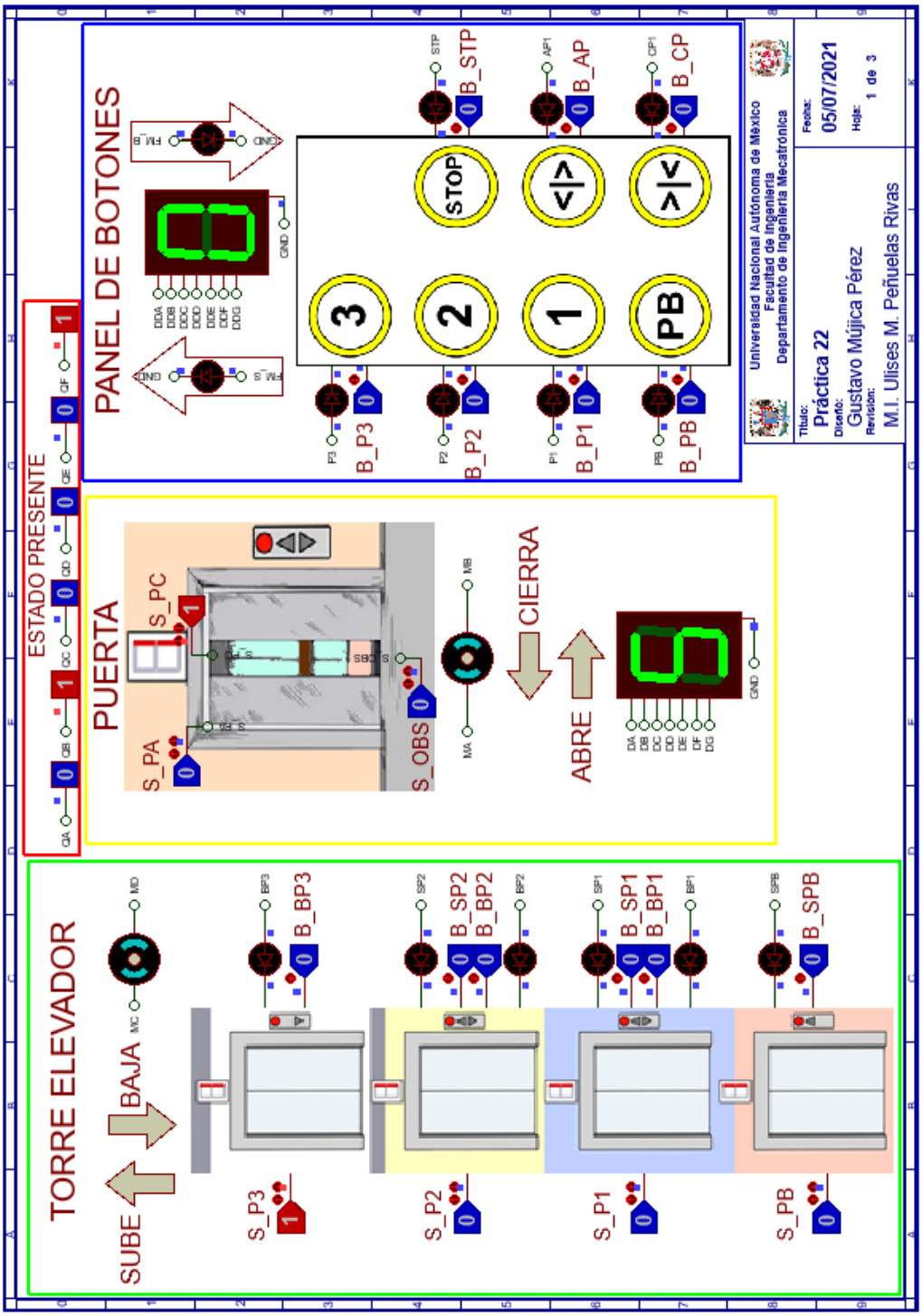


Figura P22.15 Interfaz hombre-máquina para el controlador de la Práctica 22 hoja 1/3.

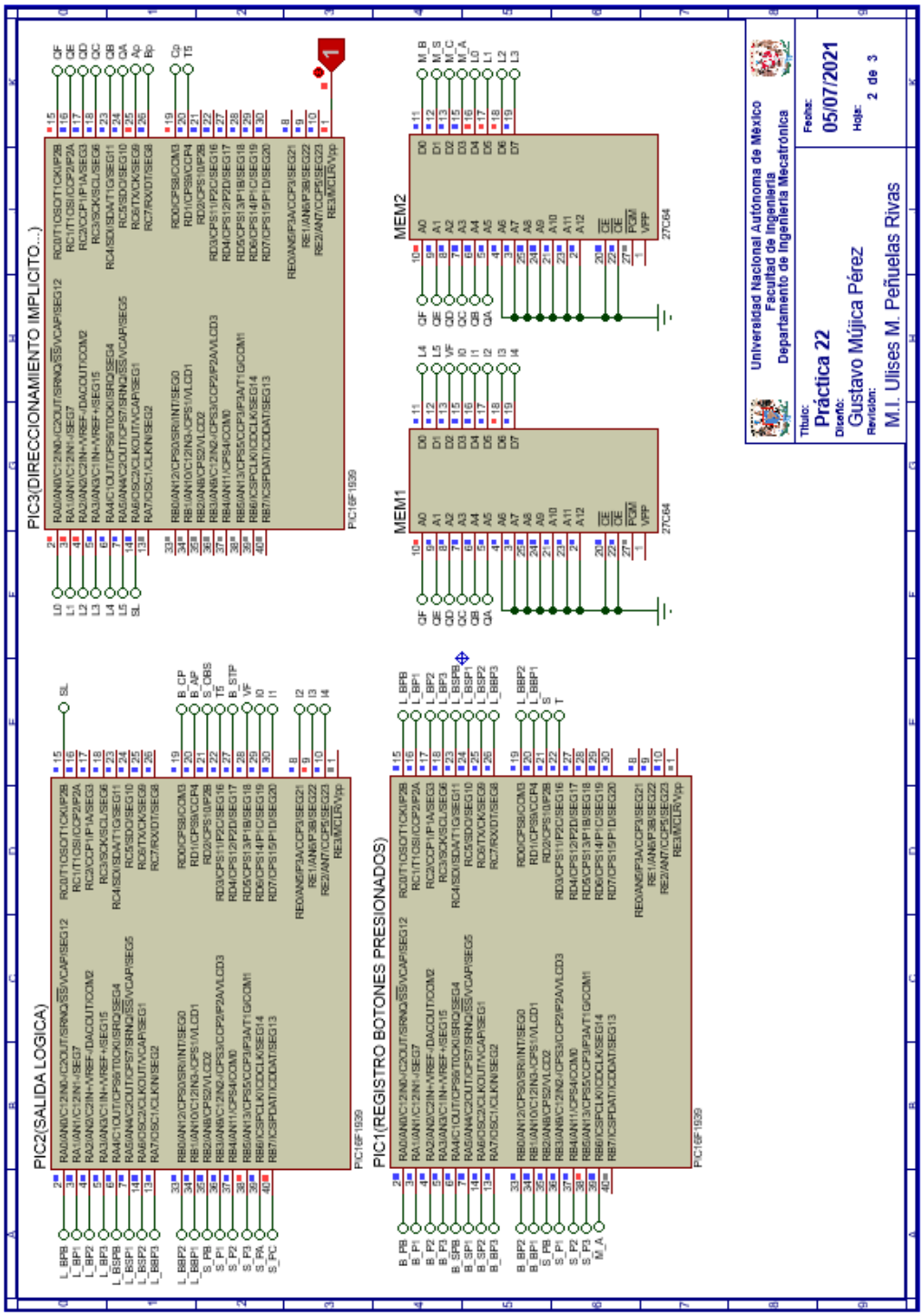


Figura P22.16 Esquema electrónico para el controlador de la Práctica 22 hoja 2/3.

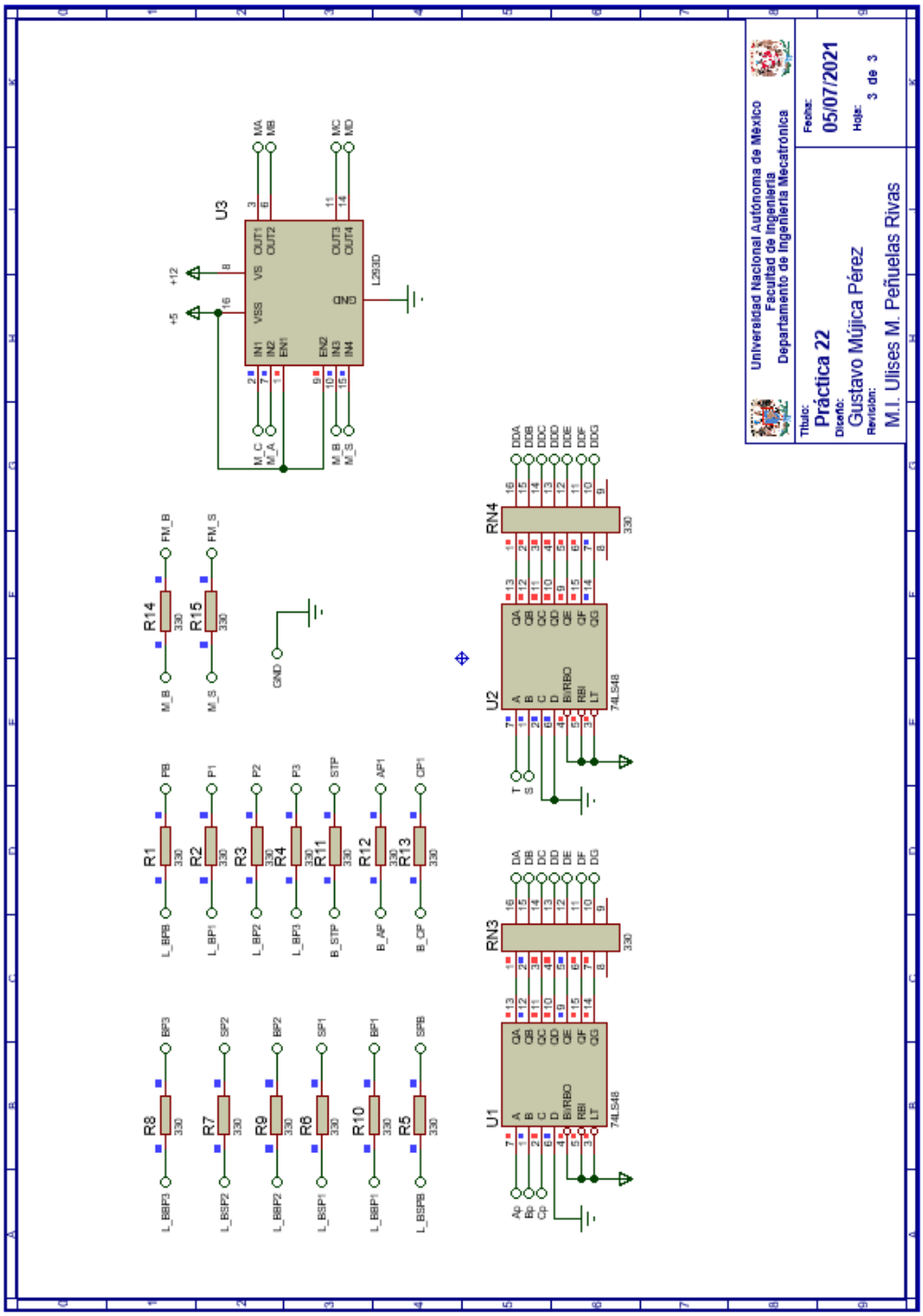


Figura P22.17 Esquema electrónico para el controlador de la Práctica 22 hoja 3/3.

Código

Para la programación de los PIC 16F1939 se utiliza un software para microcontroladores. Se muestran los códigos para el PIC 1, PIC 2 y PIC 3 respectivamente (ver figura P22.18 a la figura P22.24) se obtienen archivos con extensión “HEX” y “COF”.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     // Carga biblioteca PLD.h

//****ENTRADAS****
//BOTONES
#define B_PB   A0
#define B_P1   A1
#define B_P2   A2
#define B_P3   A3
#define B_SPB  A4
#define B_SP1  A5
#define B_SP2  A6
#define B_BP3  A7
#define B_BP2  B0
#define B_BP1  B1
//SENSORES
#define S_PB   B2
#define S_P1   B3
#define S_P2   B4
#define S_P3   B5
//SEÑAL PARA ABRIR PUERTA
#define M_An    B6
//LEDS DE LOS BOTONES
#define J  C0 //L_BPB
#define K  C1 //L_BP1
#define L  C2 //L_BP2
#define M  C3 //L_BP3
#define N  C4 //L_BSPB
#define Np  C5 //L_BSP1
#define O  C6 //L_BSP2
#define P  C7 //L_BBP3
#define Q  D0 //L_BBP2
#define R  D1 //L_BBP1
//ENTRADAS AL DECODIFICADOR DEL INDICADOR DE PISO
#define S   D2
#define T   D3

short Jn,Kn,Ln,Mn,Nn,Npn,On,Pn,Qn,Rn; //Variables inntetrnas de los FF
short Sn,Tn; //VARIABLES INTERNAS PARA EL DECODIFICADOR DEL DISPLAY INDICADOR DE PISO

void main ()
{
   pld_ini();  // INICIALIZA AL PIC COMO PLD
   pld_555(64);   // Genera señal cuadrada en Hz,
                    //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO

    while(1) //LOOP INFINITO
    {

  //if (!CLK) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO
```

Figura P22.18 Código PIC 1 de la Práctica 22 parte 1.

```
50:   //if (!CLK) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO
51:   if (!out_555) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO,
52:                   //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
53:    {
54:    //REGISTRO DE LOS BOTONES PRESIONADOS POR MEDIO DE LEDS
55:    Jn= !J&B_PB | J&!(S_PB&M_An);
56:    Kn= !K&B_P1 | K&!(S_P1&M_An);
57:    Ln= !L&B_P2 | L&!(S_P2&M_An);
58:    Mn= !M&B_P3 | M&!(S_P3&M_An);
59:    Nn= !N&B_SPB | N&!(S_PB&M_An);
60:    Npn= !Np&B_SP1 | Np&!(S_P1&M_An);
61:    On= !O&B_SP2 | O&!(S_P2&M_An);
62:    Pn= !P&B_BP3 | P&!(S_P3&M_An);
63:    Qn= !Q&B_BP2 | Q&!(S_P2&M_An);
64:    Rn= !R&B_BP1 | R&!(S_P1&M_An);
65:    // EXPRESIONES LOGICAS PARA LAS ENTRADAS A EL DECODIFICADOR DEL DISPLAY INDICADOR DE PISO
66:    Sn= S&T | !S&T&S_P2 | S&!T&!S_P1;
67:    Tn= !S&!T&S_P1 | !S&T&!(S_P2|S_PB) | S&!T&(S_P3 | S_P1) | S&T&!S_P2;
68:
69:     }
70:
71:    else
72:    {//****SECCIÓN DE MEMORIZACIÓN****
73:    //LEDS DE LOS BOTONES ENCIENDEN O APAGAN
74:     J= Jn;
75:     K= Kn;
76:     L= Ln;
77:     M= Mn;
78:     N= Nn;
79:     Np= Npn;
80:     O= On;
81:     P= Pn;
82:     Q= Qn;
83:     R= Rn;
84:     //ENTRADAS A EL DECODIFICADOR DEL DISPLAY INDICADOR DE PISO
85:     S= Sn;
86:     T= Tn;
87:    }
88:    }
89:
90: }
```

Figura P22.19 Código PIC 1 de la Práctica 22 parte 2.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD_CONTROL.h>    // Carga biblioteca PLD_CONTROL, PARA TENER MAS ENTRADAS EN EL PIC

//***LOGICA***

//ENTRADAS
#define B_PBn   A0
#define B_P1n   A1
#define B_P2n   A2
#define B_P3n   A3
#define B_SPBn  A4
#define B_SP1n  A5
#define B_SP2n  A6
#define B_BP3n  A7

#define B_BP2n  B0
#define B_BP1n  B1
#define S_PB    B2
#define S_P1    B3
#define S_P2    B4
#define S_P3    B5
#define S_PA    B6
#define S_PC    B7

#define B_CP    D0
#define B_AP    D1
#define S_OBS   D2
#define T5      D3
#define B_STP   D4
#define VF      D5
#define I0      D6
#define I1      D7

#define I2      E0
#define I3      E1
#define I4      E2

//SALIDAS
#define SL      C0

//VARIABLES INTERMEDIAS
short AUX=1; // Variable auxiliar se utiliza cuando no hay variable de entrada
short F1_P1, F1_P2, F2_P1, F2_P2, F3_P1, F3_P2, F_AP; // Funciones

void main ()
{
   pld_ini();  // INICIALIZA AL PIC COMO PLD

    while(1) //LOOP INFINITO
```

Figura P22.20 Código PIC 2 de la Práctica 22 parte 1.

```
49:      while(1) //LOOP INFINITO
50:     {
51:
52:     //FUNCIONES
53:     F1_P1=  B_P1n | B_SP1n;
54:     F1_P2= B_P2n | B_SP2n;
55:     F2_P1= B_P1n | B_BP1n;
56:     F2_P2= B_P2n | B_BP2n;
57:     F3_P1= B_SP1n | B_BP1n;
58:     F3_P2= B_SP2n | B_BP2n;
59:     F_AP= S_OBS | B_AP;
60:
61:     //****CIRCUITO COMBINACIONAL****
62:
63:     //SALIDA LÓGICA
64:     SL= VF ^ (AUX&!I4&!I3&!I2&!I1&!I0 | S_PB&!I4&!I3&!I2&!I1&I0 |
65:        S_P1&!I4&!I3&!I2&I1&!I0 | S_P2&!I4&!I3&!I2&I1&I0 |
66:        S_P3&!I4&!I3&I2&!I1&!I0 | B_PBn&!I4&!I3&I2&!I1&I0 |
67:        B_P1n&!I4&!I3&I2&I1&!I0 | B_P2n&!I4&!I3&I2&I1&I0 |
68:        B_P3n&!I4&I3&!I2&!I1&!I0 | F1_P1&!I4&I3&!I2&!I1&I0 |
69:        F1_P2&!I4&I3&!I2&I1&!I0 | F2_P2&!I4&I3&!I2&I1&I0 |
70:        F2_P1&!I4&I3&I2&!I1&!I0 | B_SPBn&!I4&I3&I2&!I1&I0 |
71:        F3_P1&!I4&I3&I2&I1&!I0 | F3_P2&!I4&I3&I2&I1&I0 |
72:        B_BP3n&I4&!I3&!I2&!I1&!I0 | B_AP&I4&!I3&!I2&!I1&I0 |
73:        B_CP&I4&!I3&!I2&I1&!I0 | F_AP&I4&!I3&!I2&I1&I0 |
74:        S_OBS&I4&!I3&I2&!I1&!I0 | T5&I4&!I3&I2&!I1&I0 |
75:        S_PC&I4&!I3&I2&I1&!I0 | S_PA&I4&!I3&I2&I1&I0 | B_STP&I4&I3&!I2&!I1&!I0);
76:
77:     }
78:     }
79:
80:
81:
```

Figura P22.21 Código PIC 2 de la Práctica 22 parte 2.

```
#include <16F1939.h> //Carga biblioteca del dispositivo
#include <PLD.h>     // Carga biblioteca PLD.h

//***CONTADOR***

//**ENTRADAS**
#define L0 A0
#define L1 A1
#define L2 A2
#define L3 A3
#define L4 A4
#define L5 A5

#define SL A6

//**SALIDAS**
#define A C5 //FFA
#define B C4 //FFB
#define C C3 //FFC
#define D C2 //FFD
#define E C1 //FFE
#define F C0 //FFF
//ENTRADAS AL DECODIFICADOR DEL DISPLAY DEL TEMPORIZADOR
#define Ap C6
#define Bp C7
#define Cp D0
//SEÑAL
#define T5 D1  //SEÑAL QUE INDICA QUE HAN TRANSCURRIDO 5 SEGUNDOS

//VARIABLES INTERMEDIAS
short At=0, Bt=0, Ct=0, Dt=0, Et=0, Ft=0; //Variables inntetrnas de los FF
short LD5,LD4,LD3,LD2,LD1,LD0; //Variables intermedias de la liga
short F0n, F1n, F2n, F3n, F0, F1, F2, F3;// Variables intermedias del temporizador
//Variables intermedias del divisor de frecuencia
short G0, G1, G2, G3, G4, G5, G0t, G1t, G2t, G3t, G4t, G5t;

void main ()
{
pld_ini();  // INICIALIZA AL PIC COMO PLD
pld_555(64);   // Genera señal cuadrada en Hz
                  //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO

   //LOOP INFINITO
   while(1)
   {
```

Figura P22.22 Código PIC 3 de la Práctica 22 parte 1.

```
45:    {
46:    //**CIRCUITO COMBINACIONAL**
47:    LD5=L5; LD4=L4; LD3=L3; LD2=L2; LD1=L1; LD0=L0;     /*ALMACENA DATOS HASTA EL CAMBIO DEL RELOJ,
48:                                        SIMULANDO UN REGISTRO, EVITANDO ASÍ
49:                                        QUE SE MODIFIQUEN LOS VALORES DE LA MEMORIA*/
50:
51:     T5=F3&!F2&F1&!F0; //SEÑAL QUE INDICA QUE HAN TRANSCURRIDO 5 SEGUNDOS
52:
53:     //ENTRADAS AL DECODIFICADOR DEL DISPLAY DEL TEMPORIZADOR
54:     Ap =!F1;
55:     Bp =F2;
56:     Cp =!F3&!F2;
57:
58:    //CIRCUITO SECUENCIAL (CONTADOR DE CARGA PARALELA)
59:
60:   //  if (!CLKK) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO
61:   if (!out_555) //PREGUNTA POR EL RELOJ  EN FLANCO BAJO,
62:                 //EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO
63:
64:    {//SECCIÓN DE OPERACIONES DEL CONTADOR
65:    At= A&E&!F | A&!D | A&!E | A&B&!C | A&!B | !A&B&C&D&E&F;
66:    Bt= B&!D | B&!E | B&E&!F | B&!C | !B&C&D&E&F;
67:    Ct= C&!D | C&E&!F | C&D&!E |  B&!C&D&E&F | !B&!C&D&E&F;
68:    Dt= D&E&!F | !D&E&F |  D&!E;
69:    Et= E^F;
70:    Ft= !F;
71:
72:      //DIVISOR DE FRECUENCIA PARA OBTENER 1/64 DE LA FRECUENCIA DEL RELOJ
73:    G0t= G0&G4&!G5 | G0&!G3 | G0&!G4 | G0&G1&!G2 | G0&!G1 | !G0&G1&G2&G3&G4&G5;//1 Hz
74:    G1t= G1&!G3 | G1&!G4 | G1&G4&!G5 | G1&!G2 | !G1&G2&G3&G4&G5;
75:    G2t= G2&!G3 | G2&G4&!G5 | G2&G3&!G4 |  G1&!G2&G3&G4&G5 | !G1&!G2&G3&G4&G5;
76:    G3t= G3&G4&!G5 | !G3&G4&G5 |  G3&!G4;
77:    G4t= G4^G5;
78:    G5t= !G5;
79:
80:    //EXPRESIONES LÓGICAS DEL TEMPORIZADOR
81:    F0n= (!A&B&!C&D&!E&F | !A&B&!C&D&E&!F | !A&B&!C&D&E&F | !A&B&C&!D&!E&!F)&(!F1&!G0 | !F3&!G0);
82:    F1n= (!A&B&!C&D&!E&F | !A&B&!C&D&E&!F | !A&B&!C&D&E&F | !A&B&C&!D&!E&!F)&(!F3&F1&!F0 |
83:          !F1&F0&G0 | F3&F1&G0 | F1&F0&!G0);
84:    F2n= (!A&B&!C&D&!E&F | !A&B&!C&D&E&!F | !A&B&!C&D&E&F | !A&B&C&!D&!E&!F)&(F2&!F1 | F2&!F0 |
85:          F2&!G0 | !F2&F1&F0&G0);
86:    F3n= (!A&B&!C&D&!E&F | !A&B&!C&D&E&!F | !A&B&!C&D&E&F | !A&B&C&!D&!E&!F)&(F3&!F1 | F2&F1&F0&G0 | F3&F1&G0);
87:    }
88:
89:    else
```

Figura P22.23 Código PIC 3 de la Práctica 22 parte 2.

```
 89:     else
 90:     {//SECCIÓN DE MEMORIZACIÓN
 91:     //CUENTA CON SL=1 Y CARGA CON SL=0
 92:     A= SL&At | !SL&LD5;
 93:     B= SL&Bt | !SL&LD4;
 94:     C= SL&Ct | !SL&LD3;
 95:     D= SL&Dt | !SL&LD2;
 96:     E= SL&Et | !SL&LD1;
 97:     F= SL&Ft | !SL&LD0;
 98:
 99:     //BITS DEL DIVISOR DE FRECUENCIA
100:     G0 = G0t;
101:     G1 = G1t;
102:     G2 = G2t;
103:     G3 = G3t;
104:     G4 = G4t;
105:     G5 = G5t;
106:
107:     //BITS DEL TEMPORIZADOR
108:     F0= F0n;
109:     F1= F1n;
110:     F2= F2n;
111:     F3= F3n;
112:
113:     while(out_555){} /*ESPERA A QUE EL RELOJ CAMBIA DE FLANCO,
114:                       POSTERIORMENTE SE USARÁN LOS VALORES INTERMEDIOS
115:                       DE LA LIGA, ES DECIR, SE SIMULA UN REGISTRO, DE OTRA MANERA
116:                       LOS VALORES CAMBIARÁN DURANTE EL PULSO DE RELOJ,
117:                       EL CAMBIO DE FLANCO DEL RELOJ ES AUTOMÁTICO*/
118:     }
119:
120:     }
121:
122: }
```

Figura P22.24 Código PIC 3 de la Práctica 22 parte 3.

Referencias

[1] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

CONCLUSIONES

Al realizar y comprender las prácticas de este trabajo el alumno será capaz de entender las diferentes técnicas para la implementación de cartas ASM, además de los diferentes conceptos que abarcan los primeros tres temas de la materia de Circuitos Digitales de los planes de estudio 2016 de Ingeniería Mecatrónica e Ingeniería en Sistemas Biomédicos.

El grado de dificultad de las prácticas propuestas tiene un orden ascendente, por consiguiente, el alumno que haya comprendido y realizado todas las prácticas podrá ser capaz de proponer nuevas o distintas soluciones para el diseño de cartas ASM de los modelos didácticos.

No hay mucha información en libros o páginas web acerca aplicaciones usando las diferentes técnicas para implementar la carta ASM, por lo que este trabajo podrá ayudar a los estudiantes universitarios que necesiten comprender mejor estas técnicas.

Si este material es utilizado para la enseñanza, un ejercicio propuesto es simplificar las prácticas 10 y 11, de manera que se reduzcan el número de estados utilizados y por lo tanto se facilite el llenado de la tabla del contenido de la memoria.

Además, se puede concluir que la técnica más eficiente para implementar cartas ASM es la de direccionamiento implícito. Es por esto por lo que, en la última práctica de las maquetas se utiliza este método.

Considerando la pandemia, este trabajo es un apoyo para las clases en línea, porque gracias a la interfaz HMI diseñada en el software de simulación, se logra dar un aspecto realístico a las prácticas.

REFERENCIAS

[1] M. M. Mano y M. D. Ciletti, Diseño digital, Pearson Educación, México, 2013.

[2] T. L. Floyd, FUNDAMENTOS DE SISTEMAS DIGITALES, Madrid: PEARSON EDUCACIÓN S.A., Madrid, 2006.

[3] R. J. , Tocci, Sistemas digitales. Principios y aplicaciones, Pearson Educación, México, 2007.

[4] C. R. Clare, Designing logic systems using state machines, McGraw-Hill, USA, 1973.

[5] C. R. Clare, *My Story at Hewlett-Packard and Before Hewlett-Packard Laboratories, USA.*

[6] H. Lam, J. O'Malley y A. A. Arroyo, Fundamentals of Computer Engineering: Logic Design and Microprocessors, Wiley, USA, 1988.

[7] U. M. Peñuelas Rivas y Y. Minami Koyama, «Método de la variable suscrita o implícita,» Universidad Nacional Autónoma de México, Facultad de Ingeniería, División de Ingeniería Mecánica e Industrial, México, 2014.

[8] C. H. Roth, L. Kurian Jhon y B. Kil Lee, Digital systems design using verilog, Cengage Learning, USA, 2016.

[9] J. Savage Carmona, G. J. Vázquez Torres y N. E. Chávez Rodríguez, Diseño de microprocesadores, UNAM, Facultad de Ingeniería, México, 2015.

[10] Y. Minami Koyama, «Ejercicio 8: Motor paso a paso», Circuitos Digitales, Séptimo semestre, UNAM, Facultad de Ingeniería, DIMEI, CDMX, 2018.