UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO DE EQUIPO ELECTRÓNICO E INTERFACES DE CÓMPUTO PARA LA INVESTIGACIÓN EN NEUROCIENCIAS

T E S I S

QUE PARA OBTENER EL TÍTULO DE
INGENIERO ELÉCTRICO ELECTRÓNICO
PRESENTAN:

VÍCTOR ADORNO DE LA ROSA
MARTÍN SABAS CRUZ JIMÉNEZ

DIRECTOR: **ING. ADRIÁN HERNÁNDEZ ALVA**

MÉXICO, D.F. **2008**

AGRADECIMIENTOS

Al Ing. Adrián Hernández Alva por ser, antes de un gran director de tesis, un gran amigo y una gran persona, que siempre nos brindó su confianza, y que sin su apoyo este trabajo no se hubiera terminado con gran satisfacción.

Al Dr. Ranulfo Romo Trujillo por darnos la oportunidad de desarrollar nuestro trabajo de tesis en su Laboratorio y por la gran cordialidad, respeto y atención que siempre nos brindo.

A todos los que tienen mucho que ver con el Laboratorio del departamento de Neurociencias por aguantarnos tanto. Gracias por que pasamos momentos muy agradables: Víctor, Lemus, Rogelio, Antonio, Manuel y Sergio. A todos ellos, gracias por la gran amistad que nos brindaron y qué seguirá firme aunque los años pasen.

A nuestro jurado, por sus valiosas observaciones y revisiones a nuestro trabajo:
M.I. Antonio Salvá Calleja.
M.I. Juan Manuel Gómez González.
M.I. José Castillo Hernández.
M.I. Antonio Martín Garcés Madrigal.

A la máxima casa de estudios (UNAM), a la gran Facultad de Ingeniería y a los profesores por su valiosa enseñanza.

Con una palabra dicha de corazón, **¡¡¡¡¡¡¡GRACIAS!!!!!!!**

DEDICATORIA

A las personas más especiales en mi vida, que llenaron el espacio que cualquier hijo necesita y supieron estar conmigo todo el tiempo y bajo cualquier circunstancia, que independientemente del estado de ánimo en que me encontrara, siempre conté con su apoyo incondicional.

Mis padres.

Agradezco a mi hermana Leticia y mis sobrinos Edgar Antonio, Giovanni y Fredy porque además de conformar mi hogar nunca han perdido la confianza en mí.

Mi familia.

A Isabel que amo muchísimo y que en cada momento me dio ánimos para no desmayar y seguir adelante.

Mi novia.

Por ser un gran tutor, guía, compañero y sobre todo un gran amigo, que nos dio confianza y nos tuvo paciencia a pesar de los tropiezos.

Ing. Adrián Hernández Alva.

Víctor Adorno de la Rosa.

Índice

I. Introducción

En el laboratorio del Dr. Ranulfo Romo Trujillo, del departamento de Neurociencias en el Instituto de Fisiología Celular de la UNAM, se investiga cómo es codificada la información en el sistema nervioso central (SNC) de monos *rhesus*. Estos estudios son importantes para entender la percepción, la memoria y el aprendizaje.

En el campo de las neurociencias se agrupan conocimientos y técnicas de áreas tan diversas como la medicina, psicología, biología e ingeniería, entre otras. La ingeniería, es valiosa en cuanto a que provee las herramientas técnicas necesarias para el desarrollo de las neurociencias. Por ejemplo: (1) dispositivos para la generación de señales que son presentadas como estímulos, y (2) dispositivos para la adquisición y el procesamiento de las señales neuronales que representan esos estímulos en el sistema nervioso.

Las neurociencias utilizan diversas metodologías. Entre las más importantes se encuentra el estudio de las lesiones que trata de estudiar el deterioro de una función cognitiva con reportes clínicos de daños en un área específica del cerebro. Otra es el registro unitario extracelular que mediante la introducción de microelectrodos (80 a 100 micras de diámetro) permite el registro de la actividad eléctrica de una neurona para posteriormente correlacionarla con alguna función cognitiva.

En el laboratorio del Dr. Romo, se lleva acabo el registro unitario extracelular para ello entrenan monos *rhesus* para que ejecuten diversas tareas; entendiendo como tarea: una secuencia donde se presentan estímulos (táctiles, auditivos o visuales) el sujeto debe responder a esos estímulos mediante movimientos previamente establecidos. Cuando el sujeto domina una tarea, se registran las respuestas neuronales del sujeto para poder establecer las relaciones estimulo-actividad neuronal-respuestas motoras.

Una vez que el sujeto ha aprendido a ejecutar una tarea, se le practica una cirugía en el cráneo y se insertan cámaras esterilizadas que permiten el acceso a la corteza cerebral del mono. En cada cámara se introducen siete microelectrodos que atraviesan la protectora duramadre craneal y llegan a la corteza cerebral del mono (el microelectrodo consiste en un filamento de aleación de platino y tungsteno forrado por cuarzo). Este sistema de microelectrodos lleva la actividad eléctrica de las neuronas del mono a una unidad de registro donde se acondiciona para poder ser adquirida y procesada.

También se ha estudiado en el laboratorio la forma en que las neuronas representan al estímulo, y como se utiliza esta representación para agrupar y diferenciar estos estímulos, es decir, la relación entre esta representación y la decisión tomada por un sujeto para indicar su detección y categorización.

La capacidad de los monos para realizar este tipo de tareas es prácticamente idéntica a la de los humanos, lo cual demuestra la utilidad de este modelo animal para investigar las bases neuronales de la percepción.

Para el investigador, es importante contar con las herramientas necesarias para poder definir las características del experimento y llevar el control del mismo. Además, debe llevar el registro de los estímulos aplicados y de las respuestas motoras (aciertos, tiempos de reacción, etc.), esto es útil para el investigador ya que así puede determinar la capacidad del sujeto para percibir la magnitud de un estímulo. Al tener acceso a la actividad neuronal del sujeto, se pueden encontrar correlaciones entre la actividad de una población de neuronas con los parámetros de los estímulos, con los parámetros del movimiento o finalmente con los procesos mentales que los relacionan. Este tipo de estudios requiere de

sistemas precisos y confiables que creen los estímulos y registren las respuestas neuronales motoras.

Para poder llevar acabo las investigaciones el laboratorio cuenta con dispositivos electrónicos y un Sistema computacional, el cual controla a estos dispositivos que generan los estímulos y el almacenamiento de los datos que se obtienen en una sesión experimental.

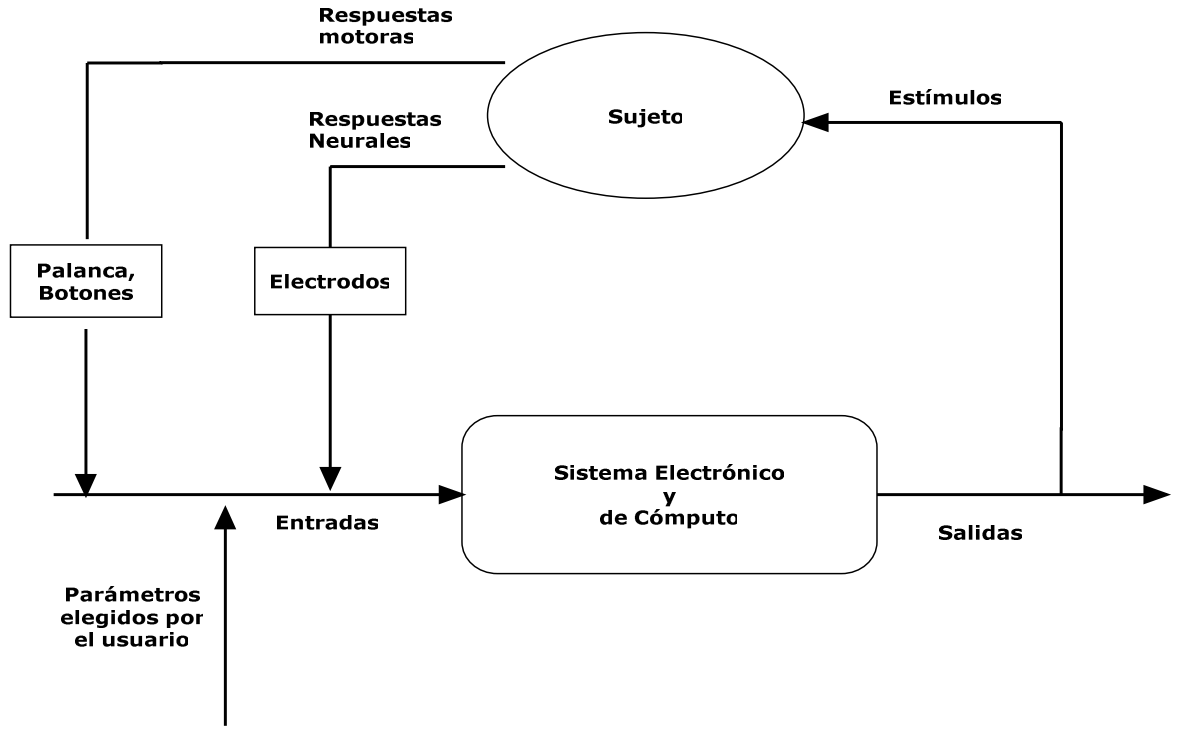


Diagrama del Sistema Electrónico y de Cómputo.

II. Definición del problema

Diseñar un sistema que permita al investigador del área de neurociencias, asignar diferentes tareas a un sujeto bajo estudio en una sola computadora. El investigador contará con una interfaz para definir los parámetros de los estímulos y el sistema los aplicará al sujeto de forma controlada. La forma en que el sujeto responda a dichos estímulos será registrada para informar al investigador la conducta (aciertos y errores), tiempos de reacción y tiempos de movimiento entre otros.

II.1. Objetivo General

Desarrollo de un sistema basado en equipo electrónico e interfaces de cómputo para la investigación en Neurofisiología usando monos rhesus. El sistema estará orientado tanto al entrenamiento de los primates como al desarrollo de los experimentos de psicofísica.

II.2. Objetivos particulares

Desarrollar un sistema que permita al investigador establecer las características del experimento usando monos rhesus, así como su ejecución y control. El sistema contendrá una interfaz gráfica de usuario.

Los datos de origen correspondientes a los parámetros de cada experimento deberán estar contenidos en un archivo de Excel.

Concentrar el sistema de definición y generación del experimento en una computadora.

Ofrecer un sistema flexible en cuanto al diseño de las tareas existentes y con la posibilidad de definir nuevas tareas.

Proveer sincronización con el sistema colector de datos electrofisiológicos.

Diseñar los sistemas electrónicos necesarios para el control y la generación de las señales necesarias para el experimento.

Elaborar la documentación necesaria para el sistema.

II.3. Justificación.

Dada la complejidad y la gran cantidad de preguntas alrededor de este tipo de experimentos y considerando que para obtener resultados se requiere de largos periodos de tiempo (dos o cuatro años) el Dr. Romo solicitó la construcción de un nuevo laboratorio.
El diseño del sistema que actualmente se utiliza es de gran ayuda para los experimentos, sin embargo, requiere de 3 computadoras para la definición y generación del experimento. Por lo que implica:

- Una computadora para la aplicación que va a interactuar con el usuario para construir, dar inicio y controlar la ejecución de una tarea.
- Una computadora para las respuestas motoras y otra para la generación de estímulos.

II.4. Restricciones.

Para el desarrollo de esta tesis sólo nos correspondió implementar la parte relacionada con la ejecución de un gran número de tareas y el registro de las respuestas motoras. La aplicación de pulsos de corriente en la corteza cerebral y el registro de respuestas neuronales serán los temas de otro proyecto.

III. Método en espiral

Basándonos en el método en espiral, el cual es un modelo de proceso de desarrollo de sistemas, conjugamos la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial. En este modelo, los sistemas se desarrollan en una serie de versiones incrementales. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o un prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del sistema diseñado.

El modelo en espiral se divide en un número de actividades de marco de trabajo que comprende regiones de tareas. Para nuestro trabajo, utilizamos las regiones siguientes:

III.1. Análisis

En esta etapa se hace una investigación exhaustiva de los requerimientos del Sistema. Para ello se hace un estudio preliminar y un análisis detallado.

III.2. Planificación

En esta parte se establecieron las tareas requeridas para definir recursos, el tiempo y otra información relacionadas con el proyecto. Como ya contábamos con los recursos, mencionados más adelante en la definición del entorno técnico, sólo fue necesario identificar y asignar el funcionamiento a dichos dispositivos.

III.3. Análisis de riesgos

Fueron las tareas requeridas para evaluar los riesgos técnicos y de gestión. Esta etapa fue dinámica puesto que conforme avanzaba el proyecto, nos enfrentábamos a especificaciones cambiantes que hacían que rediseñáramos o planteáramos nuevamente la forma en la cual se desarrollarían ciertas partes del sistema. Los requisitos nunca cambiaron, lo que cambió fue el cómo se solucionarían estos problemas.

III.4. Ingeniería

Las tareas requeridas para construir, probar e instalar una o más representaciones de la aplicación. Esta parte esta desarrollada en su mayoría en el capítulo “Diseño y construcción del sistema”.

III.5. Evaluación

En esta etapa se realizaron las pruebas al sistema para obtener una valoración del producto obtenido del área de ingeniería.

IV. Análisis

Estudio preliminar

En neurociencias, una pregunta clave es cómo el cerebro es capaz de transformar simples entradas sensoriales (registradas por los órganos de los sentidos) en percepciones, es decir, experiencias integradas que el individuo puede identificar. Para ello se usan macacos y su sentido del tacto. Los monos se entrenan y luego se les aplican estímulos vibratorios en las yemas de los dedos con el objetivo de poder correlacionar la actividad eléctrica de la neurona o de una población de ellas con los parámetros de los estímulos que se le presentan al sujeto.

Los investigadores en el área de neurociencias han diseñado un gran número de tareas pero en general se pueden clasificar en tres grupos:

Detección: el objetivo de esta tarea es determinar la capacidad del sujeto (humano o mono) para percibir la presencia del estímulo.

Categorización: La capacidad de un sujeto para clasificar una serie de estímulos, normalmente en dos categorías, por ejemplo, la velocidad de un estimulo táctil (velocidades altas o bajas).

Discriminación: En esta tarea se presentan en cada ensayo dos estímulos (que pueden ser táctiles, visuales o de otra modalidad sensorial) al sujeto y él debe compararlos para indicar cual de los dos es de mayor magnitud, o en su caso si esos estímulos son iguales o diferentes.

La actividad de las neuronas en diferentes regiones de la corteza cerebral del sujeto, que se encuentra resolviendo cierta tarea, es registrada con microelectrodos. Tanto los parámetros del estímulo como las respuestas motoras y las descargas neuronales son colectadas en tiempo real.

IV.1. Sistema actual

El sistema actual esta constituido por tres computadoras y dispositivos electrónicos.

Computadoras:

- La máquina de interfaz de usuario (IU)
  Interactúa con el usuario. Acepta los parámetros requeridos para el experimento presenta y almacena los datos electrofisiológicos.
- La máquina control (Controlador)
  Se encarga de la secuencia de la tarea activando los dispositivos que generan los estímulos (mecánicos y visuales) además de evaluar las respuestas motoras del sujeto.
- La máquina generadora de estímulos(generador)
  Produce las funciones rampa y a través del sintetizador la vibración los cuales son enviados al motor lineal.

Dispositivos:

- Sintetizador
  Es un generador de señales con la capacidad de presentar formas de onda senoidal, cuadrada y triangular con las siguientes especificaciones:
  Amplitud: rango 0 a ± 20 V, resolución 1mV.
  Frecuencia: rango 0 a 600 kHz, resolución 0.1 Hz.
  Fase: rango de 0 a 359.9 grados, resolución de 0.1 grados.

- Motor lineal
  Esté dispositivo es usado para aplicar las vibraciones mecánicas sobre la yema de alguno de los dedos del sujeto experimental. La punta de la flecha es de 2 milímetros es de cerámica y redondeada a fin de no provocar daño sobre la piel. Su desplazamiento es mediante una relación entrada/salida la cual especifica que, por cada 5mV que se apliquen a la entrada, la punta del motor lineal se desplazará una micra.
- Registro de respuestas motoras
  Este dispositivo electrónico permite identificar las respuestas motoras del sujeto que pueden ser retirar o colocar su mano sobre una palanca inmóvil o bien presionar uno de los cuatro botones usados para manifestar su decisión, además de generar una solicitud de interrupción de hardware para que se registre el instante en el cuál esos eventos suceden.

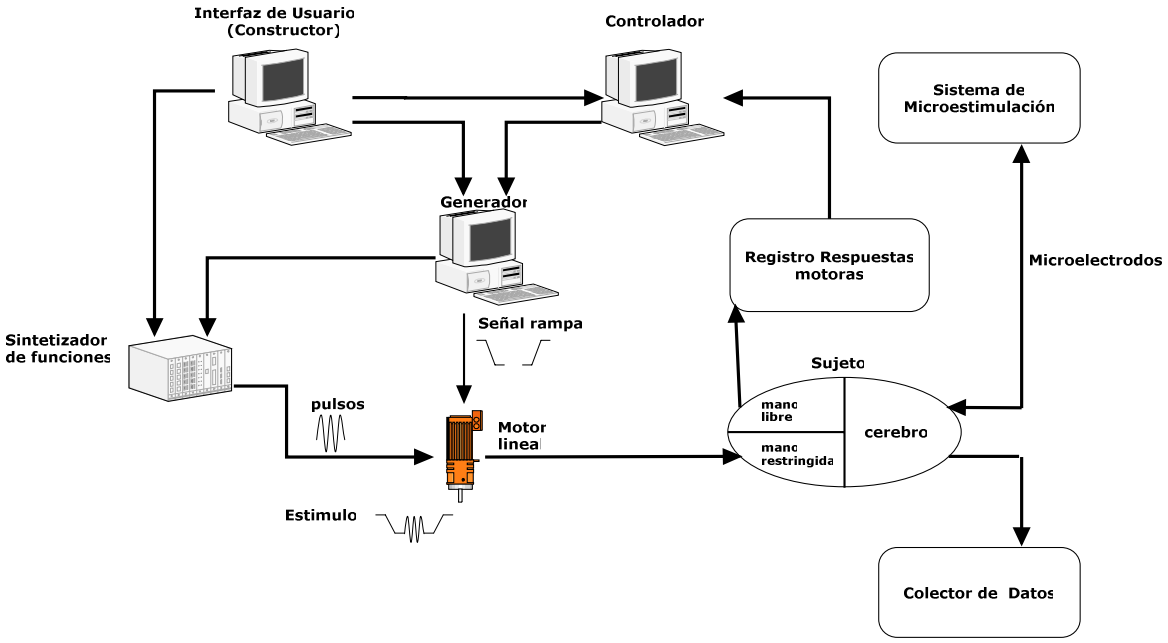


Diagrama conceptual Sistema actual

Para dar inicio a un experimento se debe cargar en el Controlador el programa que lleva la secuencia del experimento, y en la Interfaz de usuario los archivos que contienen los parámetros que requiere el experimento. La ejecución de un experimento se puede dividir en dos etapas; en la primera la IU verifica que sus archivos sean los correctos para el experimento que le fue cargado al Controlador. Si así fue la IU envía información al Micro estimulador y al Controlador. A su vez, el Controlador manda los datos correspondientes al Generador y al Colector de datos; y del Generador al sintetizador. En la segunda etapa una vez que la información ha sido distribuida a cada una de las máquinas el Controlador lleva

la secuencia de la tarea, dependiendo de las respuestas del sujeto, indicando en que momento se debe generar un estimulo, una micro estimulación o colectar datos. Cuando finaliza la tarea el Controlador se lo indica a la IU quien pide sus datos al control de micro electrodos y al colector de datos para finalmente presentar la gráfica correspondiente, y almacenar esa información en un archivo.

Análisis detallado

Se decidió tratar de identificar aquellos elementos que fuesen comunes a todas las tareas a fin de generar una estructura basada en bloques funcionales, es decir elementos básicos con características bien definidas a partir de las cuales se pudiesen construir estructuras de mayor agilidad para ello analizamos la tarea de discriminación de frecuencias de estímulos vibro táctiles, a partir da la cual se explicaran las acciones llevadas a cabo por un mono entrenado.

Para empezar, el mono es dispuesto en una silla especial en la cual se restringe su movimiento. Una de las partes del cuerpo del mono que es fijada es la cabeza, ya que es importante que los microelectrodos no se muevan, ya que se introducirían señales de ruido, además de que el animal podría lesionarse. Los estímulos táctiles son aplicados a la punta de uno de los dedos de una de sus manos, la cual queda también inmóvil; tales vibraciones son aplicadas con un motor lineal, cuya una flecha (de 2mm de longitud y de punta redondeada) realiza movimientos verticales sobre la yema del dedo estimulado. Al principio y al final de cada repetición de la tarea. La punta es colocada a aproximadamente un milímetro por encima de la piel de dicho dedo. Por otro lado, el único movimiento que puede realizar el sujeto lo hace con la mano que le queda libre, la cual utiliza para presionar uno de dos botones e indicar así cuál fue su percepción de los estímulos.

Los estímulos, que consisten en vibraciones táctiles de frecuencia, amplitud, fase, duración y forma de onda controladas, son aplicados directamente sobre la piel del dedo en cuestión. Así, el mono tiene que comparar si el segundo estímulo que recibió fue de mayor o menor frecuencia que el primero, para lo cual tiene que presionar el botón adecuado y, en caso de acertar, ser recompensado. De esta manera, el mono lleva a cabo una de las tareas de percepción y cognición básicas: comparar dos cantidades.

Para realizar lo anterior, se deben cumplir ciertas condiciones que permitan al investigador eliminar algunas variables que pudieran interferir en la investigación. Algunas de ellas son los factores de distracción alrededor del mono. Por ello, el mono es colocado en un espacio en el que se aísla del sonido externo y en el que la luz es adecuada para no distraerlo. Otro de los puntos importantes, es que la mano que queda libre también representa un factor de distracción para el mono, ya que con ella intenta tocarse la cara o algún objeto cercano, es por eso que se coloca también un dispositivo conocido como ***palanca*** sobre la cual el mono tiene que colocar su mano y evitar moverla; al ejecutar la tarea, el mono y el sistema generan una serie ordenada de eventos que son explicados a continuación con ayuda de la Figura IV-1 (la figura no está a escala).

Al inicio, marcado como S (START), la punta se encuentra por encima de la mano fija del mono y permanece allí por un periodo definido, en el cual la palanca debe estar libre y se espera que el mono no realice movimiento alguno.

La punta comenzará a bajar hasta tocar la yema del dedo del mono a partir de PD (PROBE DOWN), un tiempo típico para este movimiento es de 100ms y la distancia típica que recorre, en dirección vertical, es de 1.5mm. Es durante este intervalo de tiempo cuando el mono siente la presión que la punta ejerce sobre la piel de su dedo y es lo que le indica que debe colocar su mano sobre la palanca, este evento es conocido como KH (KEY

HOLD). El tiempo que transcurre desde que la punta comienza a bajar hasta que el mono ocupa la palanca es conocido como el primer tiempo de respuesta TR1.

Después de que se ha presentado el evento KH, la punta permanece estática durante un intervalo de tiempo variable hasta que el primer estímulo se presente. En este período, el mono no debe realizar movimiento alguno.

El primer estímulo es presentado y la punta vibra a la frecuencia f1 durante un tiempo típico de 500ms. Después de que ha terminado el primer estímulo, se presenta un período

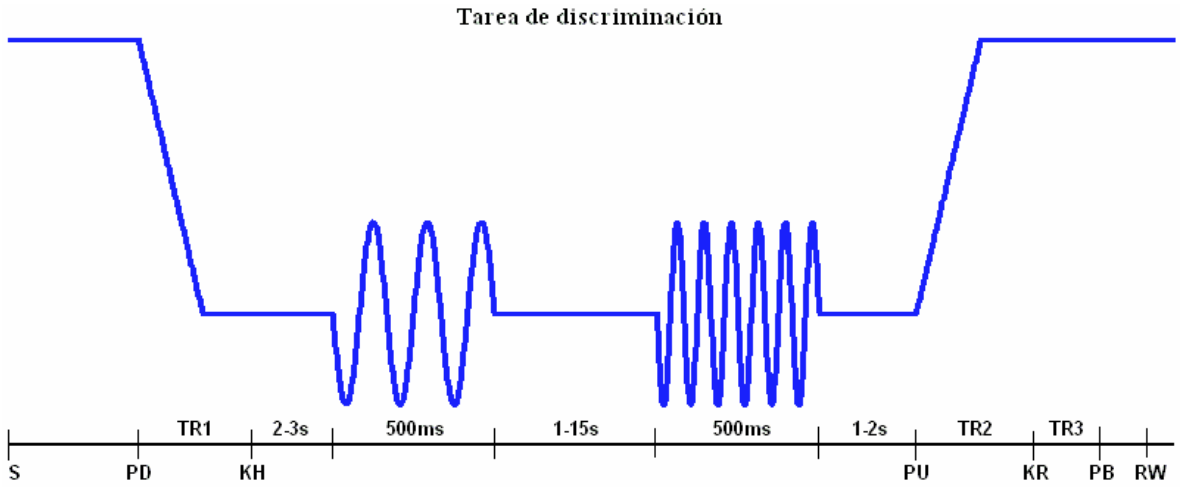


Figura IV-1. Tarea de discriminación con sus posibles eventos

ínter estímulo, el cual también es variable, y es seguido por el segundo estímulo a la frecuencia f2.

Una vez terminado el último estímulo, se presenta otro período en el que la punta está estática, nuevamente, este tiempo es variable. La punta es retirada del dedo estimulado en PU (PROBE UP) y lo hace de manera similar a PD: se desplaza linealmente 1.5mm en 100ms; y esto le señala al mono que debe quitar su mano de la palanca, lo cual ocurre en KR (KEY RELEASE) y el tiempo que toma el mono para hacerlo es TR2.

Cuando el mono ha liberado la palanca, debe presionar uno de dos botones para indicar cuál fue su percepción de los estímulos recibidos, por ejemplo: el botón a su izquierda si consideró que f2>f1 o el botón a su derecha si f1>f2. El tiempo de respuesta del mono desde que quita su mano de la palanca hasta que presiona el botón es TR3.

El mono es recompensado en RW (REWARD), si su comparación de las frecuencias de los estímulos fue correcta.

Lo anterior es un relato de lo que ocurriría si el mono está llevando a cabo correctamente la tarea, pero dada la característica repetitiva de ésta y la naturaleza inquieta del animal, es común que el mono realice acciones que provoquen él reinicio del ensayo. Dentro de tales acciones se tienen las siguientes:

El mono no puede ocupar la palanca sino hasta que perciba que la punta baja al tocar su dedo. De igual forma, el mono no debería quitar su mano de la palanca sino hasta que perciba que la punta se retira de su dedo. Por ello, el mono debe estar totalmente dedicado a percibir los estímulos, pero en caso de no ser así (lo cual se manifiesta al realizar

anticipadamente o tardíamente, los movimientos de la mano libre), se debe comenzar nuevamente ese ensayo.

IV.2. Definición de términos utilizados

IV.2.1. Estado

Se trata de un período de tiempo durante el cual el sistema mantiene una condición particular para sus salidas, por ejemplo: estímulo táctil (frecuencia, amplitud, forma de onda, etc.), estimulo visual (luz1 encendida o apagada, luz2 encendida o apagada, etc.), recompensa activada o desactivada. La transición a otro estado depende de los eventos o entradas al sistema que señalan: (1) algún movimiento del brazo del mono o (2) una señal que indica el fin de la duración del estado. Para ejecutar una tarea, es necesario que el sistema pase por una secuencia de estados. En la Figura IV-1, se tienen diez estados que son S-PD, PD-KH, el estado previo al primer estímulo, el primer estímulo, el estado ínter estímulo, el segundo estímulo, el estado posterior al segundo estímulo, PU-KR, KR-PB, PB-RW.

IV.2.2. Clase

Consiste en la enumeración de los estados que se presentarán en un ensayo, pero definiendo valores específicos para los atributos de los estímulos (frecuencia, duración o amplitud). Cada clase tiene el mismo número y tipo de estados. Por ejemplo, en la Figura IV-1, la clase asociada especificaría 10 estados, los tipos de estado, los tiempos de respuesta, los parámetros de cada estado, los tipos de movimiento asociados a cada estado, el botón que se espera que presione el mono, etc. Dicha definición siempre se hace antes de comenzar el experimento. Durante un experimento, se presentan varias clases, esto permite estudiar la sensibilidad de un sujeto ante los cambios de un atributo del estímulo (variable).

IV.2.3. Ensayo

Es la presentación de la secuencia de estados de una clase. Por lo tanto un ensayo es una repetición de la tarea. El ensayo debe ser abortado si el sujeto realiza movimientos anticipados o tardíos.

IV.2.4. Experimento

Tomando como base las definiciones anteriores, un experimento es la presentación de los ensayos correspondientes a cada una de las clases listadas en el diseño del experimento. El investigador define previamente cuántas clases forman el experimento y cuántos ensayos por clase se presentarán. También se debe definir si se llevará a cabo el registro unitario extracelular, el cual colecta la información de la actividad eléctrica neuronal mediante el sistema de microelectrodos anteriormente mencionado.

IV.3. Características del experimento

IV.3.1. Condiciones previas al experimento

El investigador debe definir cuántas clases contiene un experimento, cuántos ensayos por clase se presentarán en éste, el orden de presentación de los ensayos (secuencial o aleatorio) y el tipo de control que se hará. El número de ensayos por clase especifica el número de

veces que se llevará a cabo la ejecución de una clase en todo el experimento, esto quiere decir que si existen 10 clases y el número de ensayos por clase es 10, entonces se tendrán 100 ensayos en ese experimento. Ahora bien, la manera en la cual se presentan los ensayos de cada clase debe estar sujeta a un cierto orden: siguiendo con el ejemplo anterior, el ensayo correspondiente a una clase no puede presentarse hasta que no se hayan presentado los otros 9 ensayos de las clases restantes, es decir, los ensayos correspondientes a cada clase se presentarían en grupos de 10, cuando se ha completado un grupo de ensayos de todas las clases se tiene una ***corrida***. El orden de presentación de los ensayos indica cómo se presentarán los ensayos en cada corrida, siendo secuencial cuando, en una corrida, se presentan los ensayos en el mismo orden de las clases, y aleatorio cuando la presentación de cada clase es tomada de manera aleatoria, en el mismo ejemplo, en una corrida es posible que el primer ensayo corresponda a la sexta clase, el segundo a la cuarta, etc. Por otro lado, es posible que el investigador quiera aplicar un control pasivo, en el cual se tiene una condición experimental donde se presentan los estímulos pero el sujeto no debe realizar movimientos.

IV.3.2. Inicio del experimento

Cuando el usuario decide iniciar el experimento se presenta un ensayo cuyo fin puede ser satisfactorio o no, lo cual depende de si el mono realizó todo lo que se esperaba. Cuando un ensayo termina, se da paso al siguiente y así sucesivamente.

IV.3.3. Fin del experimento

Termina cuando se han presentado todos los ensayos por clase o si el usuario así lo decide anticipadamente.

IV.3.4. Eventos producidos por el usuario

La secuencia descrita en los dos puntos anteriores puede ser interrumpida si el final de un ensayo no es satisfactorio, en cuyo caso se aborta y se vuelve a presentar dicho ensayo. El usuario también puede intervenir abortando por cuenta propia el ensayo actual, lo cual provocará que se repita, o bien, aplicar una pausa al experimento, o repitiendo únicamente un ensayo en particular, que quiere decir que dicho ensayo estará hasta que el usuario así lo decida y continúe la secuencia normal.

IV.3.5. Presentación del ensayo

Un ensayo comienza cuando el sistema establece las condiciones del primer estado y se espera hasta la aparición de un evento que indica el fin del mismo, entonces se evalúa qué estado se presentara a continuación. La ejecución normal de los estados es secuencial y está sujeta a la presentación correcta de los eventos esperados: movimiento del mono sobre la palanca, los botones o bien el fin de la duración del estado.

Los eventos pueden ocurrir en cualquier instante pero algunos son considerados inapropiados, por ejemplo, el mono no puede quitar su mano de la palanca (KR) cuando se está aplicando el estímulo, o bien, si el mono quita su mano de la palanca pero no presiona un botón (PB). En caso de presentarse un evento no permitido, el ensayo es abortado y repetido nuevamente.

IV.4. Identificación de requisitos

Nuestro trabajo consistió en diseñar un sistema que permitiera al investigador asignar diferentes tareas a un sujeto bajo prueba. En esta relación, el investigador define los parámetros de las tareas y las ejecuta de manera controlada. El sujeto, en cambio, responde llevando a cabo ciertas acciones para las cuales ha sido entrenado. Además, el investigador es informado de la manera en la cual se lleva a cabo el experimento y la forma en la cual está respondiendo el sujeto. Como primer acercamiento al sistema nos basamos en el modelo entrada- salida mostrado en la Figura IV-2.

Además del enfoque entrada-proceso-salida, se agregaron las partes que involucran la interfaz de usuario y la de mantenimiento y autocomprobación. De esta forma, pudimos crear un modelo de componentes de sistema que estableció el fundamento para posteriores análisis y etapas de diseño. Por otro lado, también identificamos a los actores principales que intervenían en el sistema completo: el investigador, el sistema de micro estimulación, el sistema de adquisición de datos y el mono. De estos actores, los que jugarían un papel determinante en el desarrollo del sistema serían el investigador y el mono; los otros actores, en principio, son independientes del sistema puesto que sólo se sincronizan con el sistema a desarrollar.

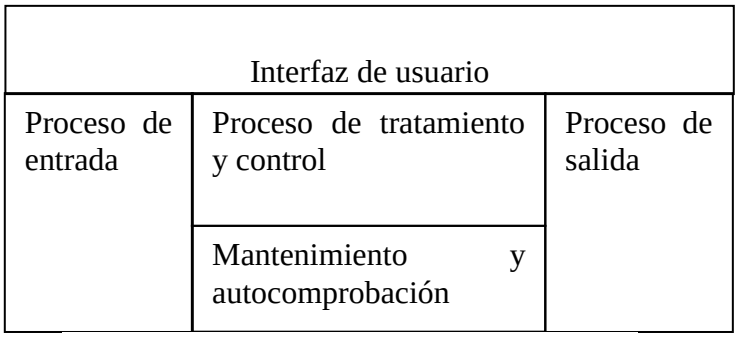


Figura IV-2. Modelo entrada salida

IV.4.1. Requisitos de interfaz de usuario

El usuario debía contar con una interfaz gráfica (GUI) en la cual pueda definir sus experimentos y llevarlos a cabo, teniendo control en todo momento sobre dichas pruebas. De lo anterior, y basados en las necesidades requeridas por el cliente, enumeramos los requisitos siguientes:

1. La definición de los datos del experimento se debía realizar a partir de una hoja de Excel.
2. El usuario podría comenzar, detener, interrumpir o abortar el desarrollo de su experimento.
3. La información generada durante el desarrollo del experimento, debe estar disponible al investigador.

IV.4.2. Requisitos de Entrada

El sistema debería contemplar las entradas procedentes de los agentes principales como las siguientes:

1. Respondería a las peticiones del usuario, quien tendría el control del experimento.
2. Tomaría en cuenta las acciones realizadas por el mono como son movimientos sobre la palanca y los botones.
3. Responder a los eventos generados por el propio sistema como el que indica el fin de la duración de un estado.

IV.4.3. Requisitos de Salida

En este aspecto, el sistema debería generar las señales adecuadas para manejar los dispositivos involucrados en el sistema:

1. Generación de señales analógicas que controlan el estimulador vibro-táctil para estimular al mono.
2. Proporcionaría las señales adecuadas para recompensar al mono.
3. Presentaría la información respecto a la ejecución del experimento en una interfaz útil para el usuario.
4. Señales digitales para estímulos visuales.

IV.4.4. Requisitos de Tratamiento y Control

El sistema procesaría las entradas de manera que éstas se convirtieran en la información y señales adecuadas para todos los demás dispositivos:

1. Atendería las peticiones del usuario cuando éste decidiera modificar la ejecución del experimento, interviniendo dispositivos, cambiando los estados del sistema, etc.
2. Ante los eventos producidos por las acciones del mono, como serían movimiento y decisiones, el sistema se debería comportar de acuerdo a las especificaciones dadas por el investigador.

IV.4.5. Requisitos de Mantenimiento y Autocomprobación

Esta parte concentraría las tareas requeridas para manejar los posibles errores que se presentarían, y la manera en la cual se identificarían problemas relacionados a los dispositivos.

1. La especificación de los parámetros debería ser correcta en la mayoría de los casos, sin embargo, el sistema debería informar al usuario acerca de cualquier posible error en dicha definición de datos.
2. Ante cualquier error, el sistema debería saber cómo manejarlo, tratándose de dispositivos, de datos o de programas.

IV.5. Definición del entorno técnico

Como el laboratorio ya contaba con sistemas que realizaban las tareas requeridas, nos apegamos al equipo disponible, el cual se menciona a continuación.

IV.5.1. Tarjetas de expansión

- **Tarjeta GPIB**

Esta tarjeta utiliza un bus PCI para establecer el protocolo de comunicaciones GPIB (General Purpose Interface Bus). Con este protocolo es posible comunicarse con un dispositivo de manera remota y automática. Esta tarjeta fue utilizada en los sistemas anteriores para la comunicación entre las computadoras de generación y control, y para manejar el instrumento que genera los estímulos, el cual será descrito más adelante.

- **Tarjeta PC-TIO 10**

Esta tarjeta cuenta con contadores digitales de 16 bits y un reloj interno cuya frecuencia puede ser dividida para proveer diferentes tiempos base (de 200ns hasta 10ms), con los cuales es posible contar diferentes tiempos. También se cuenta con dos puertos digitales de Entrada/Salida (E/S) de 8 bits cada uno, y con dos entradas de interrupción externa que se conectan a una de las líneas de interrupción de la PC que la contiene. Esta tarjeta se inserta en una de las ranuras ISA de la computadora.

- **Tarjeta LAB-PC+**

Se trata de una tarjeta multifunción que ofrece E/S digital y analógica, además de manejo de tiempos. Al conectarse al bus ISA de la PC, uno cuenta con 8 canales de entrada analógica que usan, cada uno, un convertidor analógico a digital (CAD) de aproximaciones sucesivas y 12 bits. Tenemos también dos canales de salida analógica que usan convertidores digitales a analógicos (CDA) de 12 bits con un rango de 10V. Además, los tres puertos digitales de E/S con que cuenta expanden la posibilidad del manejo de otras señales digitales.

IV.5.2. Sintetizador de señales

Es un generador de señales con la capacidad de presentar formas de onda senoidal, cuadrada, rampa, triangular, CD y ruido. Los parámetros de dichas señales pueden ser variados no sólo en su forma de onda, amplitud y frecuencia sino en la fase inicial de la señal. Este instrumento puede ser programado desde una computadora vía el protocolo GPIB, para ello cuenta con una interfaz de dicho tipo. Dentro de las características avanzadas de este instrumento se cuentan las siguientes:

Tiene 32 registros de memoria, cada uno de los cuales cuenta con 16 registros de cambio rápido de señal; ambos tipos de registro conservan configuraciones y parámetros de diferentes señales. Las restricciones aquí son que cada registro de memoria puede guardar un solo tipo de forma de onda y los registros de cambio rápido de señal pueden guardar los parámetros de hasta 16 señales diferentes (excepto por la forma de onda).

El cambio rápido de señal permite pasar de una señal con una frecuencia, amplitud y fase dada a otra cuyos parámetros sean diferentes, con una velocidad de cambio de hasta 8us. Simplemente hay que cambiar la dirección del registro de cambio de señal, lo cual se hace con el puerto digital de entrada con el que cuenta este sintetizador.

También es posible proveer un pulso de sincronización de fase para este dispositivo mediante una línea de Phase Reset, la cual controla la fase inicial de la señal.
La amplitud puede ser variada en un rango de 0 a +/- 20V, con una resolución de 1mV. El rango de frecuencia es de 0 a 600kHz y resolución de 0.1 Hz. La fase inicial puede ir de 0 hasta 359.9 grados, con 0.1 grados de resolución.

IV.5.3. Estimulador vibro-táctil

Consiste en un módulo compuesto por un motor lineal y su controlador. Su funcionamiento se basa en el principio de la fuerza de Lorentz y se compone de una bobina móvil que se encuentra dentro de un cilindro con un campo magnético permanente. Al hacer circular una corriente a través de la bobina se produce el desplazamiento de la misma y de la flecha a la cual se encuentra unida, la punta de esa flecha es de dos milímetros. El controlador de desplazamiento del motor lineal se encarga de mantener una relación lineal de Entrada/Salida la cual especifica que, por cada 5mV que se apliquen a la entrada, la punta del motor lineal se desplazará una micra. Este dispositivo tiene dos entradas analógicas que se suman internamente para formar la señal que la punta estimuladora debe seguir.

IV.5.4. Sistema operativo Windows 98 y Visual C++

El sistema operativo utilizado en el laboratorio es Windows 98 por lo que fue necesario que el sistema a desarrollar funcionara bajo esta plataforma. El laboratorio contaba también con el entorno de desarrollo de Visual C++ (VC++) versión 5, con el cual podemos desarrollar aplicaciones para Windows en 32 bits. Por otro lado, las tarjetas electrónicas antes descritas cuentan con controladores y librerías que permiten programarlas en el lenguaje C++.

IV.5.5. Dispositivos adicionales

Existían también dispositivos que son indispensables y que ya habían sido diseñados anteriormente, pero cuya funcionalidad los hizo muy abiertos a modificaciones o diseños un tanto diferentes.

- **Palanca**

Es un dispositivo basado en sensores infrarrojos que indica los movimientos que realiza el mono, los cuales representan eventos que deben ser notificados al sistema. El sujeto debe colocar su mano sobre la palanca para indicar que está dispuesto a realizar otro ensayo.

- **Botones**

Es una serie de cuatro botones de diferente color, de los cuales el sujeto debe presionar alguno e indicar una respuesta de acuerdo a la tarea asignada. Estos botones también cuentan con una lámpara de neón que ilumina la cara frontal del botón en cuestión. El investigador puede cambiar la luminosidad de dicha lámpara mediante un potenciómetro y dicha luminosidad se aplica a todos los botones.

IV.5.6. Asignación de funciones

Cada uno de los dispositivos arriba mencionados cumple ciertas funciones de las cuales se mencionan las siguientes:

La tarjeta GPIB permite configurar el sintetizador de señales de manera remota desde una PC.

La tarjeta PC-TIO 10 indicaría cuánto tiempo dura cada estado, contaría los tiempos de respuesta, serviría como interfaz para procesar los eventos externos y participaría en la generación de los estímulos.

La tarjeta LAB-PC+ generaría la señal analógica correspondiente al estímulo rampa.

El sintetizador de señales sirve para generar los estímulos aplicados al mono. Estas señales son de tipo senoidal y tienen establecidas una duración, ciclos de señal y frecuencia dadas.

El estimulador vibro-táctil recibe las señales provenientes del sintetizador y la tarjeta LAB-PC+, las cuales suma en una sola salida para que la flecha del motor lineal siga la trayectoria especificada por dicha suma de señales.

La palanca sirve como indicador

La computadora sobre la cual se desarrollaría el sistema debería ser la encargada de la definición de datos, del manejo de las tarjetas electrónicas, de la interfaz de usuario y el control del experimento.

La tarjeta PC-TIO 10 se encargaría del temporizado de cada estado, esto lo haría gracias a los diferentes tiempos de conteo con los que cuenta. También manejaría otras señales a través de sus puertos digitales, como son los eventos producidos por la palanca y los botones, o bien, para proveer recompensa al sujeto.

Las partes de la tarea en las que la punta estimuladora baja y sube de manera lineal, se conocen como estados de tipo rampa, dichos estados se generarían con la tarjeta LAB-PC+. También utilizaríamos esta tarjeta para saber la posición actual de la punta del estimulador.

El sintetizador de señales es utilizado para la generación de los estímulos, los cuales consisten en pulsos senoidales de duración, ciclos de señal y frecuencia dadas.

Sistema propuesto

La aplicación que va a interactuar con el usuario para construir, dar inicio y controlar la ejecución de una tarea así como la generación de estímulos y registro de respuestas motoras correrá en una sola computadora, en la cual también se graficaran los resultados de los experimentos llevados acabo, como lo muestra el diagrama conceptual.

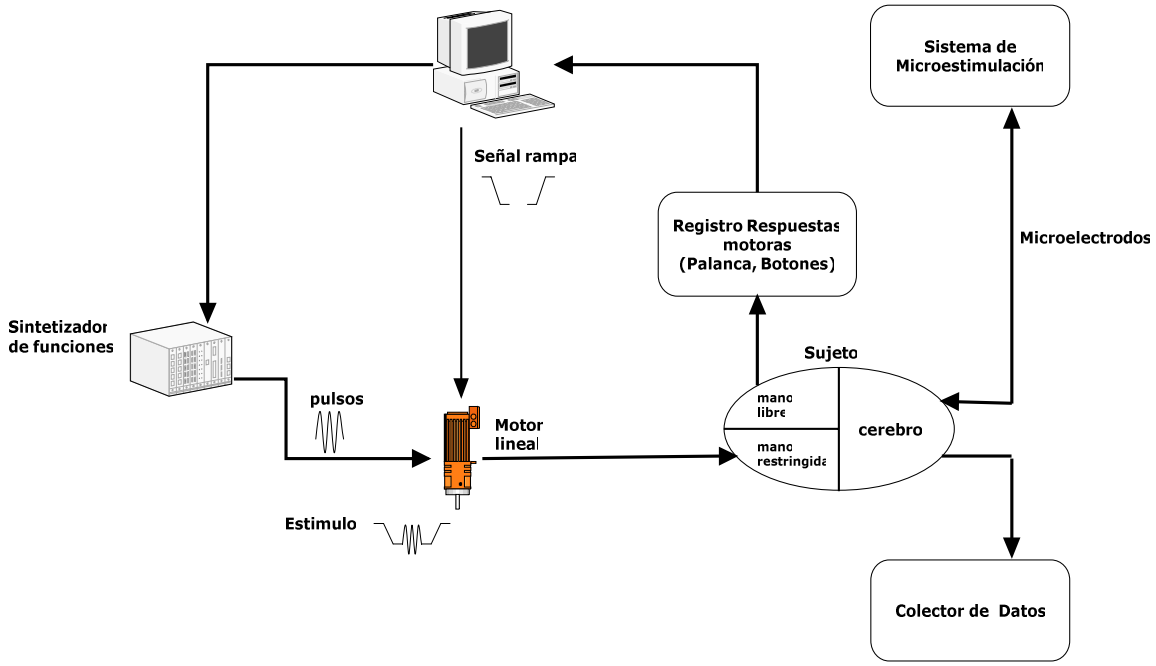


Diagrama conceptual Sistema propuesto

Adorno de la Rosa Víctor
Cruz Jiménez Martín Sabas

V. Ingeniería

El sistema se puede dividir en dos partes principales: ***Interfaz de Usuario*** y ***Control y Generación del Experimento***. Ambas partes a su vez estarían contenidas en la misma aplicación del sistema. Anteriormente se habló de la manera en la cual se lleva a cabo un experimento, por lo cual pudimos establecer un protocolo para seguir una secuencia en el programa, por ello primero atendimos la manera en la cual se controlaría y generaría el experimento y después a la interfaz de usuario.

V.1. Control y generación del experimento

Con la Figura V-1 mostramos los estados que componen la tarea de discriminación de frecuencias, asociados al estímulo táctil. En la figura se observa que los eventos, marcan la transición de estados y que al inicio de cada nuevo estado el sistema debe llevar a cabo alguna acción. Las señales que sigue el estimulador se generan con la tarjeta LAB-PC+ (rampas) y con el sintetizador de señales (vibraciones).

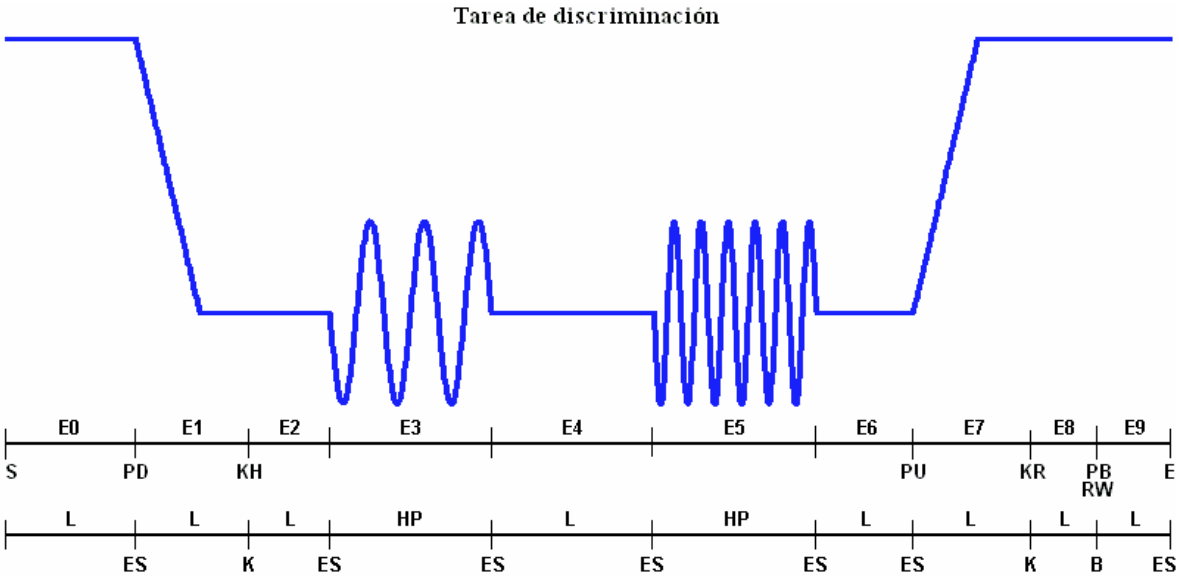


Figura V-1. Los eventos definen la transición de estados

La línea de acotación superior muestra los eventos y los 10 estados que comprenden a esta tarea, tal como se había mencionado antes. En la segunda línea de acotación se puede observar que los estados que son generados por la tarjeta LAB-PC+ son aquellos señalados con una L, y los estados generados por el sintetizador son señalados con HP. También se ve que los eventos involucrados se deben ya sea a un fin de estado (ES), un movimiento en la palanca (K) o la presión de un botón (B). Debemos tomar en cuenta también que cada estado tiene una duración específica definida por el investigador. Por lo tanto es necesario (1) registrar los eventos (acciones del mono sobre la palanca o los botones, así como la señal que indica el fin de la duración de cada estado). Y (2) generar las señales de salida que producen los diferentes estímulos (táctiles, visuales y de recompensa).

V.1.1. Temporizado de los estados

Cada estado esta definido, entre otras cosas, por su duración. En la Figura V-1, se puede ver que los eventos ES se deben a que el sistema indica el fin de la duración del estado, y los eventos K y B indican un final de estado debido a un movimiento del mono. Los estados cuyo fin es de tipo ES tienen una duración que se conoce antes de iniciar el ensayo en cuestión, mientras que la duración de los estados que involucran movimiento no puede ser determinada previamente.

Como mencionamos antes, la tarjeta PC-TIO 10 cuenta con 10 contadores digitales de 16 bits que pueden ser configurados de muchas maneras de acuerdo al propósito en particular. Dichos contadores cuentan con dos terminales de entrada, una de ellas sirve para establecer una señal de reloj diferente a las disponibles; la otra terminal es de habilitación y sirve para activar el funcionamiento del contador ya sea por nivel o por flanco. La única salida existente en cada contador es una señal digital. Existe además un reloj interno de 5 MHz que puede ser escalado para proveer frecuencias de reloj de 5 MHz, 1 MHz, 100 kHz, 10 kHz, 1 kHz y 100 Hz, es posible incluso proveer una señal externa que sirva como reloj para la generación de la señal. Con ello se logran resoluciones de tiempo de 200 ns, 1 μs, 10 μs, 100 μs, 1 ms y 10 ms. De las configuraciones existentes usamos dos: una que permite generar un pulso de ancho y retardo variable, y otra que es capaz de contar eventos.

La configuración de pulso retrasado se puede ver en la Figura V-2. Después de que el contador a ser utilizado es configurado para funcionar de esta manera, sólo hay que indicar al contador los parámetros del tiempo base, del retraso y del ancho del pulso para que genere la señal.

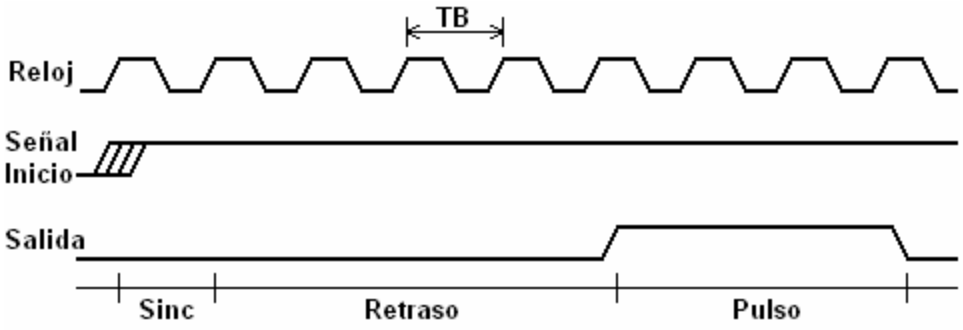


Figura V-2. Pulso retardado generado por la tarjeta PC-TIO 10

El retardo y el ancho del pulso están en unidades del tiempo base utilizado, por ejemplo, si se utiliza un tiempo base de 100 μs y se especifican un retardo y un ancho de pulso de 23 y 2, respectivamente, entonces la duración del retardo será de 23×100 μs = 2.3 ms y el pulso durará 200 μs; en total, esta señal durará 2.5 ms. Tenemos que mencionar también que, al ser contadores de 16 bits, la cuenta máxima alcanzada, comenzando en 0, es de 65,535, lo cual significa un manejo adecuado de tiempos base ya que, por ejemplo, un retardo de 100 ms no se podría llevar a cabo con un tiempo base de 1 μs. Un punto importante a señalar es que existe un período de incertidumbre en el que el contador se sincroniza con el reloj, lo cual se debe a que, una vez dada la señal de inicio, el contador comienza a funcionar cuando se presenta el primer flanco de reloj; no obstante, esta sincronización está limitada a un ciclo del tiempo base utilizado. La señal de inicio de

generación del pulso puede ser dada mediante un comando desde la computadora, o bien, mediante una señal externa.

La configuración que cuenta eventos la utilizamos para los estados que involucran movimiento por parte del mono. En este caso, el contador lleva a cabo una cuenta libre de los ciclos de reloj del tiempo base especificada, es decir, cuenta de uno en uno hasta alcanzar su cuenta máxima (65,535), después de la cual vuelve a contar desde cero, o bien, se detiene y enciende una bandera que indica que se ha desbordado el contador. Cuando comienza la generación de los estados en donde se esperan movimientos, se pone a funcionar otro contador cuya cuenta es leída cuando se produce el evento correspondiente al movimiento. Así, sólo leemos la cuenta de dicho contador y la multiplicamos por el tiempo base utilizado para obtener el tiempo de respuesta del mono; dicho tiempo de respuesta es entonces relativo al inicio del estado en cuestión.

Estas configuraciones, así como la generación de las señales, son totalmente programables y también pueden ser controladas por señales externas. Los tipos de datos utilizados por las funciones que manejan estas señales son de diferentes tipos, pero en su mayoría son enteros de diferentes tamaños por lo que fue necesario convertir los datos provistos por el investigador a los tipos de datos que utilizan, no sólo este sino, todos los demás dispositivos. En caso de existir algún tipo de error en el funcionamiento de la tarjeta o en la llamada a una función con parámetros equivocados, las funciones correspondientes regresan un error el cual debe ser manejado adecuadamente.

En cuanto al diseño, la manera en la cual temporizamos cada estado es mediante pulsos retardados como los anteriores. Cada estado tiene una duración establecida por el investigador por lo que no es posible asignar un solo tiempo base a todos los estados, por ello tomamos la decisión de asignar un tiempo base para cada estado; de igual forma, cada estado cuenta con un retardo y un ancho de pulso. Dichos parámetros son cargados cada vez que comienza la generación de un estado, y cambian conforme cambian los estados. En total, utilizamos 5 contadores: el contador 1 (CTR1) fue usado para el temporizado de cada estado, CTR3 para contar los tiempos de respuesta, CTR5 para proveer el pulso de recompensa, y CTR8 y CTR10 fueron utilizados en la generación de los estímulos (discutida más adelante). Fue necesario usar CTR5 porque la unidad que provee la recompensa, necesita un pulso de entrada para abrir, por un cierto tiempo, una válvula que deja pasar el líquido que es llevado a la boca del mono.

Hasta aquí, sabíamos cómo manejar el tiempo de cada estado y que el contador CTR1 producía eventos que indicaban que un final de estado había llegado; sin embargo, la palanca y los botones también producen eventos que tienen que ser atendidos. Las secciones siguientes discuten cómo se diseñaron la palanca y los botones y después pasamos al manejo asíncrono de los eventos.

V.1.2. Diseño de la palanca

La palanca tiene como objetivo evitar que el sujeto esté realizando movimientos mientras se lleva a cabo el ensayo, con ello se logra que toda la atención del mono se centre en los estímulos que va a recibir y, en cuanto a las señales de la actividad neuronal se refiere, evitar que la actividad relacionada al movimiento se mezcle con la de la percepción de los estímulos táctiles.

En un ensayo, la punta del estimulador baja y se posa sobre uno de los dedos del mono, esto es señal de que coloque su mano libre en la palanca y la deje ahí hasta que la punta del estimulador haya subido, entonces deberá quitar su mano. Además, el mono debe

llevar a cabo lo anterior dentro de ciertos tiempos establecidos por el investigador, por lo que no es válido continuar con el ensayo si el sujeto no ha puesto (o quitado) su mano en (de) la palanca; de ser así, el ensayo comenzará nuevamente.

Este dispositivo debe indicar los movimientos que realiza el sujeto, dichos movimientos sólo son dos, a saber, ocupar y liberar la palanca. Mecánicamente hablando, la palanca es una estructura como la que se muestra en la Figura V-3. Cuenta con un cilindro sostenido horizontalmente por dos paredes verticales las cuales, además de restringir el movimiento de la mano del mono al espacio limitado por ellas y la pared trasera, contienen los fototransistores (receptores) y leds infrarrojos (emisores) que detectan la presencia de la mano.

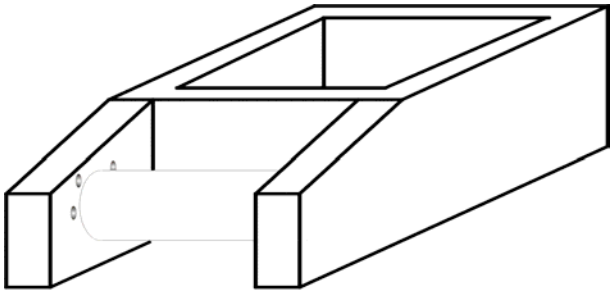


Figura V-3. Forma física de la palanca

V.1.2.1. Problemas a resolver en el diseño de la palanca

Este dispositivo se comunica con nuestro sistema para detectar el movimiento de la mano del mono, quien debe colocar su mano dentro de tiempos definidos por el usuario. Como se dijo anteriormente, el tiempo que le toma al mono realizar algún movimiento no se puede conocer previamente, por ello, la señal que obtenemos de la palanca es totalmente asíncrona. No obstante, el tiempo de respuesta del mono debe ser cuantificado en el momento que ocurra. Así, la palanca se diseñó teniendo en cuenta que debía enviar una señal en el momento en que ocurriese algún movimiento por parte del mono; además, también debía indicar qué tipo de movimiento (ocupación o liberación) se había producido.

Otro problema fue que, al utilizar los dispositivos opto electrónicos, se presentaban rebotes cuando la señal cambiaba de estado, lo cual no es favorable ya que producirían múltiples señales de movimiento que no indicarían un movimiento sino varios, lo cual no era real ya que se había producido un solo tipo de movimiento. Como el rayo de cada par emisor-receptor es lineal, sólo abarcábamos una pequeña parte de toda la región en la cual se movería la mano del mono, lo que hicimos fue barrer una región más amplia.

V.1.2.2. Electrónica digital usada en el diseño de la palanca

Usando tres pares emisor-receptor dispuestos adecuadamente, pudimos resolver el problema de la región abarcada por dichos sensores. Después, sólo tuvimos que combinar la salida de cada par emisor-receptor para obtener una sola salida. En la Figura V-4 se puede ver el circuito final. A la salida de cada fototransistor hay conectado en cascada, un transistor para dar potencia a la señal; por otro lado, las salidas de éstos últimos transistores van dirigidas a una compuerta NAND cuya salida presenta un nivel bajo cuando los rayos

de los pares emisor-receptor no son interferidos, esto es, cuando la palanca está libre. Por otro lado, si el mono coloca su mano en la palanca, alguno será interferido provocando que la salida de dicha compuerta esté en un nivel alto.

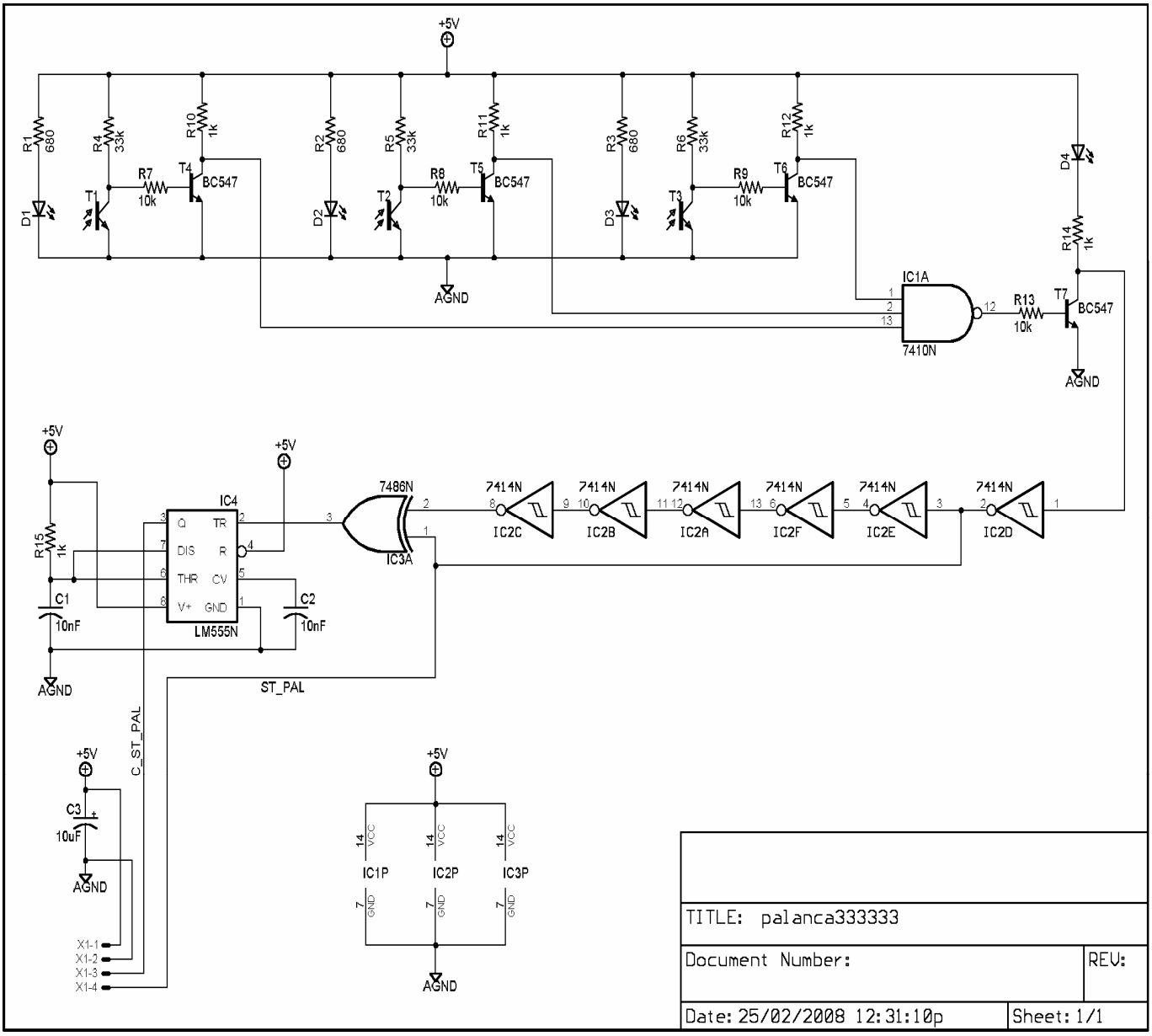


Figura V-4. Circuito de la palanca

El problema surge, precisamente, cuando la palanca cambia de estado, es decir, cuando el mono ocupa o libera la palanca ya que es en ése momento cuando se presentan los rebotes mencionados anteriormente. Por ello, utilizamos un inversor Schmitt trigger (7414); dicho circuito elimina los rebotes y hace que una señal que cambia lentamente lo haga más rápido. La salida del primer inversor nos provee la señal que deseamos para saber el estado de la palanca: nivel alto cuando la palanca está ocupada, nivel bajo cuando la palanca está liberada. La señal mencionada está etiquetada como ST_PAL.

Ahora bien, tener una señal que nos indicara el estado de la palanca no fue suficiente puesto que sólo indicaba un nivel (el estado de la palanca) y no un evento. El resto del circuito, genera un pulso positivo de 10μs cada vez que ST_PAL cambia de estado. Consiste en un circuito formado por un temporizador 555 monoestable disparado por un pulso negativo cuando ST_PAL cambia. Las entradas de la compuerta XOR (7486) son ST_PAL y el complemento de la misma pero con un leve retraso, provisto por los inversores restantes. La salida de la compuerta XOR cambia de alto a bajo cuando sus entradas son iguales, lo anterior sucede cuando ST_PAL cambia de estado: si no cambia, las entradas de la compuerta son diferentes y la salida está en alto; cuando ST_PAL cambia, las entradas de la compuerta son iguales durante un período breve (el retraso provisto por los inversores), entonces la salida cambia a bajo en dicho período y tenemos el pulso negativo que dispara al monoestable. La señal que forma la salida del monoestable se etiqueta como C_ST_PAL.

Más adelante se verá cómo se comunican las dos señales de salida con la aplicación del sistema final para la lectura de eventos.

V.1.3. Descripción de los botones

En la realización de un ensayo existe la necesidad de esperar una respuesta del sujeto en alguno de los botones, (de hasta cuatro que pueden ser elegidos anticipadamente por el investigador). El sistema reconoce qué botón fue presionado y conforme a los parámetros almacenados en el sistema se realizará la siguiente ejecución. La señal del botón presionado debe ser almacenada en un control de respuestas que básicamente es un registro, en el mismo instante la señal del botón presionado será detectada y comunicada al sistema.

Circuito de los botones. Consta de cuatro botones, de los cuales cada botón está formado por entradas normalmente abiertas y normalmente cerradas el circuito se divide en dos etapas:

- La primera etapa elimina los rebotes.

Los botones son elementos mecánicos por lo que al cerrar y abrir el circuito existe una generación de transiciones de voltaje (rebotes) que son detectados en la entrada de interrupción externa de la PC-TIO 10 ocasionando varias interrupciones con solo presionar una vez cualquier botón y lo que se quiere es detectar una interrupción cada vez que un botón sea presionado.

El circuito empleado es un circuito RC y compuertas lógicas de la familia TTL, con el retardo de propagación de compuertas y la constante de tiempo del circuito RC se logra el retardo necesario para eliminar los transitorios no deseados.

- La segunda etapa detecta el flanco de subida.

Al presionar un botón y soltarlo se produce un pulso teniendo por lo tanto un flanco de subida y un flanco de bajada, este pulso sirve para ser detectado en el control de respuestas, pero este pulso no es útil para generar una interrupción externa ya que la interrupción externa se genera con un flanco de bajada por lo que la interrupción se generaría hasta dejar de presionar el botón. Con lógica combinacional y configurando en forma monoestable el 555 TTL, se logra detectar en un pulso el flanco de subida.

V.1.4. Manejo de eventos asíncronos en Windows

Dada la naturaleza asíncrona de los eventos producidos por los dispositivos antes discutidos, la computadora encargada de la generación de señales de los sistemas anteriores detectaba y manejaba tales eventos mediante interrupciones. El programa que se encargaba de esto era estructurado y había sido escrito en Lenguaje C, además, corría bajo MS-DOS e invocaba una función que atendía la petición de interrupción cada vez que ésta ocurría; para ello, utilizaba la palabra clave *interrupt*, provista por muchos de los antiguos compiladores de C/C++. Esta computadora estaba dedicada completamente a generar las señales para el estimulador interpretando los comandos que la computadora constructora le enviaba.

Como nuestra tarea era crear una aplicación que fuera totalmente para Windows, tuvimos que investigar la forma en que Windows trabaja: un programa para Windows básicamente registra, crea, muestra y actualiza una ventana para después entrar en un ciclo de mensajes. Dentro de dicho ciclo, espera a que el sistema operativo le envíe un mensaje, si esta ventana es la destinataria de algún mensaje, entonces lo procesa y lo despacha; esta última acción se refiere a que, dependiendo del mensaje recibido, el flujo del programa se encamina a su correspondiente manejador de mensaje, que es dónde se realiza alguna acción, un ejemplo de este proceso es el movimiento del ratón o cuando se presiona una tecla. El problema fue que bajo Visual C++ (el entorno usado en el desarrollo de nuestra aplicación) la palabra clave *interrupt* no está disponible, por lo que no fue posible crear una rutina de atención de interrupción como se haría en MS-DOS. En Windows, el enfoque para el manejo de interrupciones es diferente: el encargado de manejar dicha interrupción es un dispositivo virtual que después pasa la información al sistema operativo; posteriormente, el sistema operativo envía un mensaje a la aplicación destinataria para que sea procesada la interrupción. Así, la interrupción no es atendida inmediatamente sino que es puesta en una cola de mensajes.

Nuestro sistema procesaría los eventos asíncronos provenientes de la palanca, los botones y los contadores de la tarjeta PC-TIO 10, así que nos enfrentamos al problema de manejar eventos asíncronos mediante interrupciones, pero sin interferir con el funcionamiento del sistema operativo.

Familiarizándonos con las tarjetas disponibles, nos dimos cuenta que la tarjeta PC-TIO 10 contaba con dos líneas externas de interrupción, las cuales van dirigidas a una de las líneas de interrupción de la computadora donde está instalada dicha tarjeta. El fabricante de estos dispositivos provee el software necesario para instalarlos y programarlos; además, dentro del paquete que acompaña al producto, se encuentran librerías que enlazamos con nuestro programa. Incluida en dichas librerías, encontramos una función que configura y habilita el recibo de mensajes provenientes de eventos en las tarjetas como, por ejemplo, las líneas de interrupción externa. De esta forma, había que indicar qué mensaje se configuraría y a qué programa (ventana) se le enviaría para que lo procesara.

Ya en nuestra aplicación, escribimos un manejador de mensajes que atendiera el mensaje recibido. El siguiente paso fue conocer la arquitectura de manejo de mensajes que tiene Windows: consta de una función que recupera los mensajes y otra que los despacha, esta última entra en una sentencia de opción múltiple (switch, en C++) en la cual se atiende el mensaje enviado. No obstante, como usamos VC++ nos encontramos con que la colección de clases de Microsoft (MFC, por sus siglas en Inglés) provee una forma distinta en la que se hace uso de macros y que lleva por nombre Mapa de Mensajes, con estas estructuras se facilita y se hace más eficiente la escritura del manejo de mensajes.

Al final, la escritura de los manejadores de mensajes consistió en los pasos básicos siguientes:

1. Asignar el mensaje a ser enviado a nuestra aplicación, lo cual se logró asignando un número de 32 bits al ID del mensaje que queríamos configurar. Como Windows ocupa parte de los primeros mensajes para uso propio, dicho ID debe estar en el rango WM_USER - 0x7FFF. Se puede definir más de un mensaje por dispositivo, dependiendo del tipo de evento a ser configurado.
2. Declarar y definir el manejador de mensaje del ID antes asignado. Esto se hizo en los archivos de código fuente, cabe mencionar que estas acciones se apegan a ciertas reglas establecidas por MFC.
3. Utilizar la macro ON_MESSAGE para notificar a la ventana de nuestra aplicación que, cuando recibiera el mensaje antes mencionado, se encaminara al manejador especificado dentro de dicha macro.

Dentro de la definición del manejador de mensaje, se dio el tratamiento al evento que produjo la interrupción. Para utilizar esta característica de envío de mensajes hicimos lo siguiente:

1. Configuramos y habilitamos el mensaje con el cual se deseaba trabajar.
2. Se inició una operación asíncrona, esperando que ésta produjera una interrupción.
3. Cuando la operación anterior notificara a la aplicación enviándole un mensaje, éste sería atendido por el manejador correspondiente.
4. Dependiendo de la aplicación, decidimos si deshabilitábamos el envío de mensajes o no, esto último es muy importante ya que tratamos con rutinas re-entrantes.

V.1.5. Restricciones en el uso de interrupciones

El problema que presenta una aplicación Windows es que coloca los mensajes en una cola y los va despachando conforme van llegando. De tal manera que el mensaje proveniente de algún dispositivo es atendido hasta que los otros mensajes hayan sido procesados; no obstante este problema, las acciones llevadas a cabo por los sujetos bajo estudio comparadas con la velocidad de procesamiento de la computadora utilizada, hacen que dicho problema no sea tan grave, excepto en la generación de los estímulos (véase la parte correspondiente a los estímulos). En nuestro caso, la computadora que usamos tiene un procesador Pentium II a 266MHz, 128MB en RAM y utiliza Windows98 SE. Con la ayuda de un osciloscopio y un programa de prueba, obtuvimos que el evento es atendido en un tiempo promedio de aproximadamente 1ms.

Otro problema fue que si el mensaje se perdía por cuestiones del sistema operativo, entonces la siguiente acción ya no es llevada a cabo puesto que la finalización de un evento

es la que hace que comience el siguiente. Nuevamente, fue en la generación de los estímulos donde observamos que cuando los tiempos entre interrupciones son menores que alrededor de 2 o 3 ms, los mensajes se pierden, en consecuencia, el estímulo generado es incompleto y el ensayo se queda parado en ése punto. Esta es la característica principal por la cual el tiempo muerto mínimo está restringido.

V.1.6. Ampliación del número de líneas de interrupción en la PC-TIO 10

Anteriormente, mencionamos que la palanca, los botones y los contadores de la PC-TIO 10 indican eventos externos que requieren ser procesados. Por otro lado, sólo contábamos con dos entradas de interrupción externa en la PC-TIO 10, tuvimos entonces que diseñar un circuito que pudiera extender tales entradas.

Para ser precisos, las señales asíncronas que manejamos son las siguientes:

La señal proveniente de la palanca.
La salida del contador CTR1 de la PC-TIO 10 que temporiza la duración de cada uno de los estados.
La señal que se obtiene cuando uno de los botones ha sido presionado.
La salida de los contadores CTR8 y CTR10 de la PC-TIO 10, los cuales temporizan las duraciones de los pulsos y los tiempos muertos en la generación de los estímulos.

Como el mono tiene una mano fija, en la cual es estimulado, y una mano libre con la que hace los movimientos en la palanca y en los botones, es improbable que realice un movimiento en tales dispositivos al mismo tiempo. Combinamos entonces ésas interrupciones en una sola; más aún, es poco probable que ocurra un movimiento justo cuando un contador esté terminando. En base a lo anterior, la configuración usada fue la mostrada en la Figura V-5:

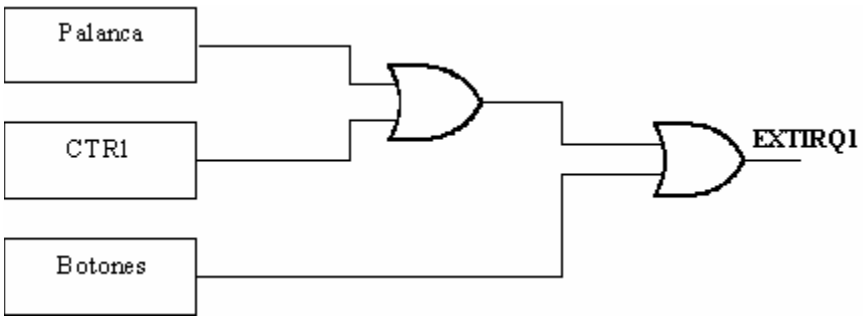


Figura V-5. Ampliación del número de líneas de interrupción

Con esta configuración, sabíamos cuándo se generaba una interrupción debida a un movimiento o a la finalización de un estado pero no sabíamos qué dispositivo la había ocasionado. Haciendo uso de un registro determinamos el origen de la interrupción leyendo la salida de dicho registro. Conectamos entonces las dos salidas de la palanca, la salida del contador CTR1 y las señales de los botones a la entrada del registro. Como es sabido, un registro necesita una señal de reloj para pasar los datos de la entrada a la salida, el problema es que como los eventos que manejamos son pulsos asíncronos y de breve duración, tuvimos que proveer una señal de estrobo más que de reloj. Necesitábamos usar el registro

como un seguro (latch) más que como un registro y además proveer una señal de estrobo que actualizara el registro sólo cuando se presentara la interrupción EXTIRQ1. Hubiéramos usado esta última señal como estrobo pero debido a los retrasos presentes en los circuitos digitales, no estábamos seguros si los datos llegarían primero a la entrada del registro o si lo

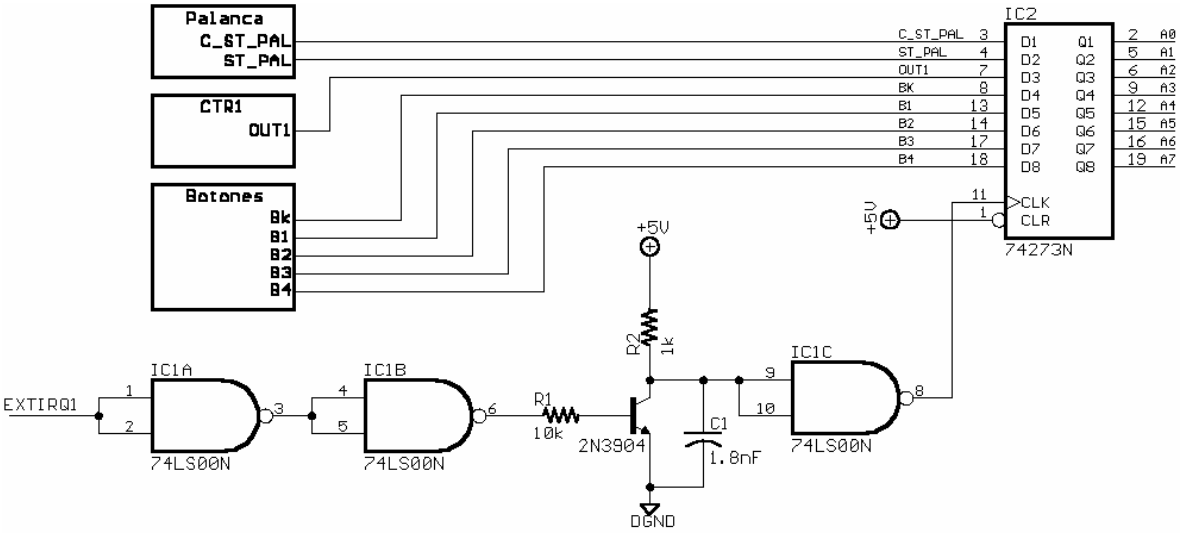


Figura V-6. Generación de la señal de estrobo a partir de EXTIRQ1

haría primero la señal EXTIRQ1, la cual es una combinación de algunas de las señales de entrada; estábamos expuestos a una posible pérdida de datos. Para asegurar que los datos a la entrada del registro llegaran primero y después se aplicara el pulso de estrobo, simplemente retrasamos la señal EXTIRQ1 en aproximadamente 300ns. Esta configuración se puede ver en la Figura V-6:

Se puede observar que la señal EXTIRQ1 se retrasó para que pudiera ser aplicada a la entrada de reloj del registro, cuya salida se conectó directamente al puerto digital de entrada A de la PC-TIO 10.

Ahora bien, en su flanco positivo, la señal EXTIRQ1 funciona como estrobo para el registro, pero cuando dicha señal presenta su flanco negativo, dispara una interrupción que es atendida por el sistema operativo. Si a lo anterior agregamos que EXTIRQ1 representa un pulso positivo, tenemos el siguiente panorama:

Cuando ocurre un evento asíncrono (movimiento o fin de estado), la entrada del registro recibe los estados de todos los dispositivos.

El flanco positivo de la versión retrasada de EXTIRQ1 actualiza la salida del registro pasando los datos que hay en la entrada, asegurándolos ahí hasta que se presente otro evento.

En su flanco negativo, la señal EXTIRQ1 dispara una interrupción, la cual es procesada por el sistema operativo y encaminada a la aplicación adecuada.

Cuando la interrupción es atendida, leemos el puerto A de la PC-TIO 10, el cual contiene los datos asegurados del registro, para saber cuál dispositivo produjo la interrupción.

V.1.7. Identificación del dispositivo causante de la interrupción

Una vez dentro del manejador del mensaje que atiende el evento asíncrono, leímos el puerto A de la PC-TIO 10 y determinamos qué acción seguir de acuerdo a dicha lectura; para ello, primero analizamos las lecturas que eran más probables de ocurrir, éstas se muestran en la Tabla V-1. En las primeras 5 columnas se muestra la lectura de los 4 bits menos significativos del puerto A, en hexadecimal y en binario. En realidad, leemos todos los 8 bits del puerto, pero en principio solamente utilizamos los 4 bits menos significativos, esto es así porque sólo teníamos tres tipos de dispositivos causantes de interrupción: el contador CTR1, la palanca y los botones. En la Figura V-6, se puede ver que a dichos bits menos significativos llegan las señales C_ST_PAL, ST_PAL, OUT1 y BK. Las primeras dos señales indicarían que hubo un cambio en el estado de la palanca y el estado actual de la misma. OUT1 indicaría que ha llegado el final de un estado. La señal BK solamente indicaría que un botón ha sido presionado, lo cual es suficiente ya que si éste fuera el caso, bastaría con leer los cuatro bits más significativos, que es a donde llegan las señales de los cuatro botones, véase la Figura V-6. Así, también se redujo el número de lecturas sobre las cuáles decidiríamos qué hacer ya que sólo tenemos que considerar 16 posibles lecturas, de las cuales descartamos también algunas, como se explica a continuación.

La columna **Motivo** muestra el tipo de evento por el cual se produjo la interrupción y la última columna indica el dispositivo que produjo la interrupción. Las lecturas donde **A3**, **A2** y **A0** son cero, no son posibles ya que implicaría que ninguno de los dispositivos provocó la interrupción, éstas lecturas son 0x0 y 0x2. Además, es imposible que el mono lleve a cabo un movimiento en la palanca y en los botones (**A3** = **A0** = 1) de manera simultánea, por lo tanto, las lecturas 0x9, 0xB, 0xD y 0xF tampoco son posibles. Las lecturas 0xA y 0xE tampoco son posibles porque el sujeto no puede estar presionando un botón y tener su mano sobre la palanca al mismo tiempo.

PuertoA Hex	**A3** Bk	**A2** OUT1	**A1** ST_PAL	**A0** C_ST_PAL	**Motivo**	**Origen**
0x0	0	0	0	0		
0x1	0	0	0	1	liberación reciente	Palanca
0x2	0	0	1	0		
0x3	0	0	1	1	ocupación reciente	Palanca
0x4	0	1	0	0	fin de estado, palanca desocupada	CTR1
0x5	0	1	0	1	fin de estado, liberación reciente	CTR1, palanca
0x6	0	1	1	0	fin de estado, palanca ocupada	CTR1
0x7	0	1	1	1	fin de estado, ocupación reciente	CTR1, palanca
0x8	1	0	0	0	botón, palanca desocupada	Botón
0x9	1	0	0	1		
0xA	1	0	1	0		
0xB	1	0	1	1		
0xC	1	1	0	0	botón, fin de estado	botón, CTR1
0xD	1	1	0	1		
0xE	1	1	1	0		
0xF	1	1	1	1		

Tabla V-1. Lectura de los 4 bits menos significativos del puerto A de PC-TIO 10

Por otro lado, aunque es poco probable que el sujeto lleve a cabo un movimiento al mismo tiempo que finaliza un estado, se contempla dicha posibilidad en las lecturas donde **A3** o **A0** y **A2** son iguales a 1: 0x5, 0x7 y 0xC.

La palanca puede ser ocupada o liberada (**A0 = 1**), lo cual determinamos basándonos en el bit que indica el estado de la palanca (**A1**), sólo hay que tener en cuenta que si es cero, significa que la palanca está actualmente liberada y cambió de estado, es decir, fue liberada. El caso de cuando la palanca es ocupada es semejante: **A0 = 1** y **A1 = 1**.

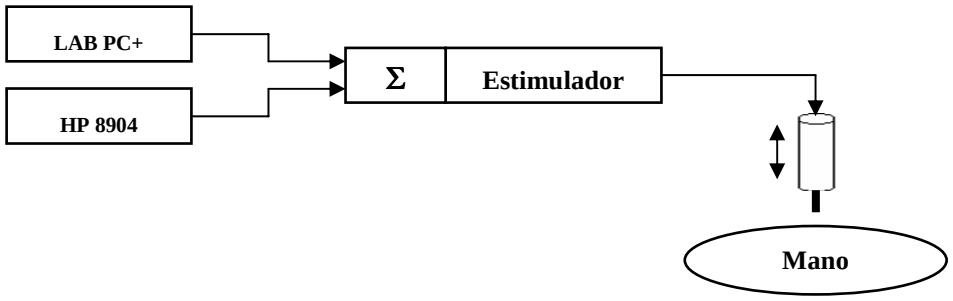
A diferencia de la palanca, el caso de los botones fue más sencillo ya que sólo uno de ellos puede ser presionado y sólo nos interesaba saber cuando había sido presionado y no cuando fuera soltado, es decir, no nos importaba saber cuándo el mono dejaba de presionar el botón. Cuando este evento se presentara, haría que **A3 = 1** y tendríamos entonces que un botón había sido presionado; las únicas posibilidades viables que abarcarían tal situación son: 0x8 y 0xC, si éstas ocurrieran tomaríamos en cuenta los 4 bits más significativos para determinar qué botón fue presionado, véase la Figura V-6.

Hasta este punto, hemos discutido la forma en la cual manejamos los eventos producidos por los diferentes dispositivos, así como la manera en la cual temporizamos cada estado. A manera de resumen, cada vez que se presenta un evento se produce un pulso en la terminal EXTIRQ1 de la PC-TIO 10, dicho pulso dispara una interrupción, la cual envía un mensaje a la ventana de nuestra aplicación, donde el manejador de mensaje correspondiente lleva a cabo alguna acción dependiendo del dispositivo que creó tal interrupción. Si pudiéramos ver la señal en EXTIRQ1, veríamos varios pulsos en donde cada uno de ellos correspondería ya sea a un final de estado proveniente del contador CTR1, o bien, debido a un movimiento por parte del mono.

Por otro lado, la siguiente etapa fue la de generar las señales correspondientes de cada estado. Dichas señales se alimentarían directamente al estimulador vibro-táctil para que la punta estimuladora las siguiera y realizara así los desplazamientos sobre la punta del dedo del mono.

V.1.8. Generación de las señales de cada estado

De la Figura V-1, podemos ver que, para este tipo de tarea, existen diez estados cada uno de los cuales involucra diferentes señales que son generadas ya sea por la tarjeta LAB-PC+ o por el sintetizador. Por ejemplo, los estados E0, E2, E4, E6, E8 y E9 están formados por una señal constante y son generados por la tarjeta LAB-PC+; los estados E1 y E7 se conocen como estados de tipo rampa y también son generados por la tarjeta analógica antes mencionada; por último, los estímulos E3 y E5 son generados por el sintetizador de señales. Ya antes se había comentado que el estimulador vibro-táctil cuenta con dos entradas de comando las cuales se suman internamente para formar la señal de salida.



Varias son las razones por las cuales se utilizan dos dispositivos en la generación de la señal de salida del motor:

La tarjeta LAB-PC+ ofrece gran flexibilidad en la generación de diferentes señales ya que cuenta con un búfer en donde se almacena la señal a ser generada. De este modo, se puede generar cualquier señal que se desee cargando el búfer con los datos deseados o, en caso de una señal constante, sólo escribiendo el valor correspondiente sin llenar el búfer. Sin embargo, esta tarjeta tiene una tasa de muestreo y un tamaño de búfer algo limitados. Además, toma cierto tiempo cargar el búfer asociado con la señal. Lo anterior quiere decir que para generar todo un ensayo sólo con ésta tarjeta, tendríamos muchos puntos a ser convertidos y estaríamos limitados a la frecuencia de muestreo de este dispositivo.

El sintetizador puede generar muy bien señales cuyas amplitudes y frecuencias, respectivamente, están en los rangos siguientes: 0 a ±20 V y 0 a 600 kHz. El problema fue que las señales a generar sólo pueden ser de un mismo tipo, es decir, no podemos generar una señal senoidal primero y después una triangular. Por otro lado, si guardamos dichas señales en diferentes registros de memoria, tenemos que recuperar, en tiempo de ejecución, el registro correspondiente lo cual toma un tiempo que no está bajo nuestro control. Por ello, el sintetizador sólo se utiliza para la generación de los estímulos.

Los estados de tipo rampa se generaron con la tarjeta analógica porque el diseño del búfer no es complicado y porque la frecuencia de muestreo para generar esta señal no es muy grande. Además, la tarjeta analógica cuenta con una resolución de 2.44 mV por cada paso digital, mientras que el sintetizador tiene una resolución de 1 mV. Teniendo en cuenta que los estímulos representan señales pequeñas, usamos la tarjeta LAB-PC+ para realizar los movimientos gruesos y el sintetizador para los estímulos.

Dado que la generación de los estímulos es más compleja que la de los otros estados, ésta se trata en una sección aparte. A continuación se discute cómo se generaron los otros estados.

V.1.8.1. Generación de una señal constante

La tarjeta LAB-PC+ provee dos canales de salida D/A de 12 bits. Cada canal puede ser configurado como unipolar o bipolar; en el primero, tenemos un rango de 0.0000 V a +9.9976 V, y en el segundo es de -5.0000 V a +4.9976 V, con una resolución de 2.44 mV por paso digital. La resolución es el resultado de la razón del rango total de 10 V y el número de pasos digitales que se tienen con 12 bits que es de 4096 (2^{12}). La configuración que nosotros usamos fue la unipolar ya que nos interesaba hacer un recorrido desde 0 V hasta 10 V, lo cual corresponde a un rango de 0 micras hasta 1999 micras, que es el recorrido que puede realizar la punta estimuladora con una resolución de 1 micra por cada 5 mV aplicados a la entrada.

Para generar una señal de tipo constante sólo hay que escribir el valor del voltaje deseado a la salida del canal analógico de salida. Nosotros utilizamos el canal de salida 0 (DAC0OUT) para generar las señales constantes y de tipo rampa. En el caso de una señal constante, hicimos uso de una función que permite escribir un valor a la salida de este canal, la ventaja es que la tarjeta mantiene el valor del último voltaje escrito, es decir, basta escribir un solo punto para que ése valor de voltaje se mantenga ahí.

V.1.8.2. Generación de la señal correspondiente a la rampa

En este caso, no bastó con una sola muestra (punto) para la señal de salida sino que tuvimos que hacer uso de funciones generadoras de forma de onda, las cuales utilizan un búfer almacenado en memoria. Para ello, primero tenemos que calcular la forma de onda y almacenarla en un arreglo de datos en memoria. Dicho cálculo involucró determinar la tasa de muestreo que sería utilizada a la salida del canal analógico, teniendo en cuenta el número de puntos y la duración de la forma de onda. Además, calculamos la señal punto por punto, lo cual significó aplicar la ecuación de una recta para cada punto; ésta ecuación fue determinada a partir de la posición inicial (Pi) de donde comenzaría a desplazarse la punta estimuladora, la posición final (Pf) a donde llegaría la misma, y el tiempo (D) que tomaría dicho desplazamiento, lo cual se observa en la Figura V-7. Se puede observar también que existe un tiempo adicional después de que la punta ha bajado totalmente. Para la generación de la rampa, utilizamos un búfer de tamaño fijo de 200 puntos, en el cual guardamos únicamente los puntos que representan la señal, esto es, después de que el búfer completo se ha generado, la tarjeta analógica mantiene en su salida el último valor escrito.

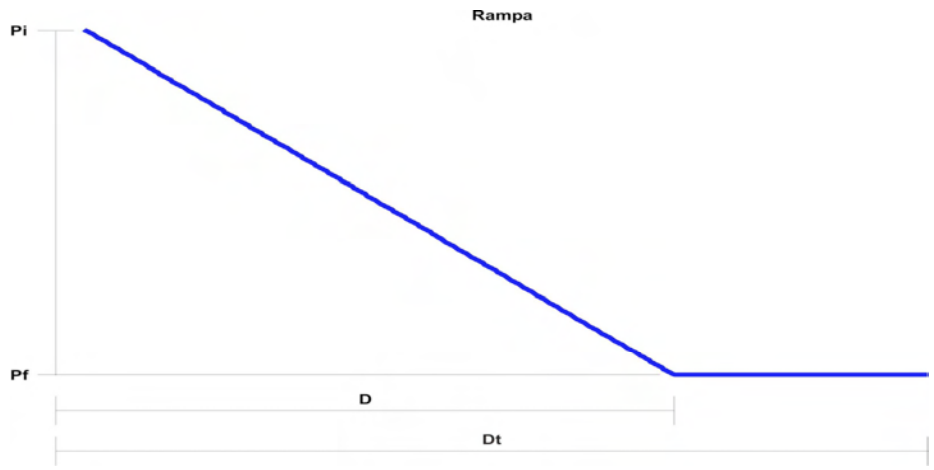


Figura V-7. Rampa generada como una línea recta

La frecuencia a la cual se actualiza la salida del canal analógico representa la razón del número de puntos a generar y el tiempo en el cual se llevará a cabo está operación. Una de las razones por las cuales decidimos mantener fijo el tamaño del búfer fue precisamente para reducir el número de variables, de tal manera que sólo calculamos la frecuencia a partir del número de puntos y de la duración (D) de la rampa. Cabe mencionar que después de varias pruebas y tomando en cuenta el número de puntos utilizados y la computadora que usamos, determinamos que el número máximo de puntos por segundo a generar, cuando la generación es asíncrona, era de alrededor de 4585. Al hablar de generación asíncrona, nos referimos a que la función que se encarga de la generación regresa el control

al programa principal, con lo cual el programa puede atender otros procesos o eventos y no tiene que esperar a que la generación termine para después continuar. Con esta frecuencia de actualización, pudimos generar satisfactoriamente las señales de la rampa, con una duración mínima de rampa de aproximadamente 43 ms. Obviamente, con una generación síncrona se puede generar una rampa de duración menor pero se pierde el control del programa durante la generación. La computadora que utilizamos fue aquélla que finalmente contendría la aplicación principal del sistema, siendo una máquina Pentium II a 166 MHz con 128 MB en RAM y bajo Windows 98. Finalmente, aplicamos una conversión de valores reales a valores binarios que son los que utiliza la tarjeta LAB-PC+.

V.1.8.3. Tipos de estímulo

En la Figura V-1 se puede ver que los estados E3 y E5 son de tipo estímulo. En este caso, estos estímulos representan señales senoidales de parámetros conocidos; sin embargo, no todos los estímulos son de este tipo. Tras años de investigación en el laboratorio del Dr. Ranulfo Romo, se han llegado a diseñar diferentes tipos de estímulos de acuerdo al tipo de investigación que se esté llevando a cabo. En lo que sigue, comentamos los diferentes tipos de estímulos utilizados y después pasamos a la generación de ellos. Antes, es necesario mencionar ciertos términos utilizados en la generación de estímulos.

V.1.8.3.1. La forma de onda

La forma de onda representa una señal determinística (excepto por la forma de onda de tipo ruido blanco), en este caso, se trata de alguna de las señales que provee el sintetizador HP 8904A: senoidal, rampa, cuadrada, ruido y de componente directa (CD). Para las señales que son periódicas se pueden establecer la frecuencia, la amplitud y la fase. En el caso de las últimas dos señales listadas anteriormente, sólo se puede fijar la amplitud. El estado E3, antes mencionado, es un estímulo con un tono senoidal de amplitud y frecuencia dadas.

V.1.8.3.2. El pulso

El pulso es una señal basada en un ciclo completo de alguna de las formas de onda provistas por el sintetizador. Cabe aclarar que un pulso es una señal sin discontinuidades en un intervalo dado. Como ejemplo, los estímulos E3 y E5 constan, cada una de de 3 y 6 ciclos de forma de onda senoidal, respectivamente.

V.1.8.3.3. El tiempo muerto

El tiempo muerto es una señal constante de valor cero y de duración definida que puede presentarse entre pulsos.

V.1.8.3.4. El estímulo

Un estímulo es una señal formada por pulsos y puede o no contener tiempos muertos. Dentro de las características principales de un estímulo se tienen las siguientes:

1. La forma de onda de los pulsos del estímulo es la misma para todo el estímulo.
2. Los pulsos del estímulo están formados por ciclos enteros de la forma de onda, excepto cuando el estímulo es de tipo bipulso, en cuyo caso los pulsos están formados por medio ciclo de la forma de onda, más adelante se discuten los tipos de estímulo.

3. La frecuencia del estímulo está dada por el número de pulsos en un segundo. Para el caso en que la duración del estímulo no sea de un segundo, se debe mantener la proporción, lo cual quiere decir que si, por ejemplo, la frecuencia del estímulo es de 40 pulsos por segundo, entonces se presentarán 20 pulsos en 500ms, 10 en 250ms, etc.
4. El número de pulsos presentados es entero. No puede especificarse una duración de 100ms para una frecuencia de estímulo de 32 pulsos por segundo ya que implicaría presentar 3.2 pulsos. El programa tampoco redondea esta última cantidad para hacerla entera.
5. Pueden o no tener tiempos muertos. En caso de tenerlos, el número de tiempos muertos siempre es uno menos que el número de pulsos.

Los siguientes, son los tipos de estímulos utilizados en el laboratorio. Cada tipo de estímulo maneja parámetros semejantes aunque éstos son calculados de manera diferente.

V.1.8.3.5. Estímulo periódico más un pulso final

Este estímulo presenta un cierto número de pulsos dado por el producto de la frecuencia (F_e) y la duración (D_e) del estímulo. Este producto debe ser entero dado que sólo se presentan pulsos enteros. La duración de cada pulso está dada por el recíproco de la frecuencia del tono (F_t). Este estímulo puede tener tiempos muertos lo cual se determina a partir de las frecuencias de estímulo y del tono, las cuales deben mantener cierta relación para que se cumpla el requisito de tiempo muerto mínimo, 3ms en nuestro caso. Si el estímulo presenta tiempos muertos, todos ellos son de la misma duración con lo cual tenemos una señal periódica.

Si $F_e > F_t$, el estímulo no puede ser calculado ya que en este tipo de estímulos se tiene un ciclo completo del tono por cada pulso presentado. Lo anterior quiere decir que un pulso duraría más que el tiempo máximo permitido, en este caso, dicho tiempo máximo lo da el recíproco de F_e, donde F_e es la frecuencia de estímulo en pulsos por segundo, y F_t es la frecuencia del tono. Si este es el caso, se debe reportar una incoherencia en los datos y reportar una advertencia al usuario para que corrija los datos.

Si $F_e = F_t$, entonces no existen tiempos muertos en el estímulo, no se agrega un pulso adicional y la duración del estímulo es la que se especifica en los datos de origen. La forma de onda resultante es continua en el tiempo que dura el estímulo y se presentan el mismo número de ciclos de tono que el número de pulsos que se obtendría del producto de F_e con D_e, donde D_e es la duración dada del estímulo. La Figura V-8 muestra un estímulo con las características siguientes: 10 pulsos por segundo, 500ms de duración, 10Hz de frecuencia de tono y 10um de amplitud de tono; por lo tanto, tenemos un estímulo de $10 \cdot 0.5 = 5$ pulsos sin tiempos muertos.

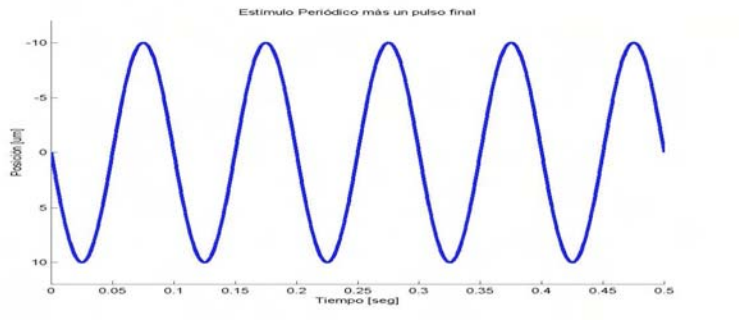


Figura V-8. Estímulo periódico sin tiempos muertos ni pulso final

Si $F_e < F_t < F_{Tmin}$, el estímulo no podrá ser generado satisfactoriamente ya que la señal de interrupción que genera el pulso y el tiempo muerto no podrá ser procesada por el sistema operativo. F_{Tmin} es la frecuencia de tono mínima que garantiza que un estímulo con tiempos muertos se generará adecuadamente, dicha frecuencia está dada por la Ecuación V-1:

$$F_{T\min} = \frac{F_e}{1 - D_{TM\min} \cdot F_e}$$

Ecuación V-1

Donde D_{TMmin} es la duración mínima de tiempo muerto y es igual a 3ms.

Si $F_t \geq F_{Tmin}$, el estímulo contendrá tiempos muertos y se agregará un pulso adicional al final del estímulo. Esto quiere decir que la duración real del estímulo será la suma de la duración dada del estímulo más la duración del pulso adicional. Por ejemplo, si en el ejemplo anterior cambiamos la frecuencia del tono por una frecuencia de 20Hz, tenemos un estímulo con 5+1=6 pulsos, 5 tiempos muertos y una duración real de 500ms más 1/20Hz=0.05s, lo cual da un total de 550ms, la duración de todos los tiempos muertos es la misma. El estímulo resultante se muestra en la Figura V-9:

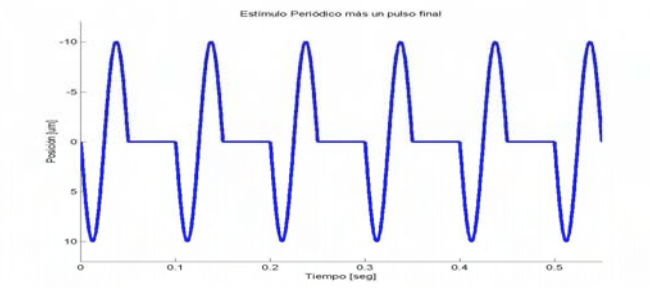


Figura V-9. Estímulo periódico con tiempos muertos y pulso adicional

V.1.8.3.6. Estímulo aperiódico más un pulso final

Este estímulo presenta N_p pulsos de señal dados por el producto de la frecuencia y la duración del estímulo, dicho producto debe ser un número entero. La duración de cada pulso está dada por el recíproco de la frecuencia del tono. Este tipo de estímulo tiene $N_p - 1$ tiempos muertos, cuya duración mínima es de 3ms. En este caso la señal generada no es periódica, pero se mantiene la proporción de pulsos en el tiempo. La duración de cada uno de los tiempos muertos se genera de manera aleatoria. En este caso, también se agrega un pulso adicional al final del estímulo.

Si $F_{tono} < F_{Tmin}$, la señal no podrá ser calculada ya que no se cumplirá la duración mínima de los tiempos muertos y entonces la señal de interrupción no podrá ser procesada por el sistema operativo.

Si $F_{tono} \geq F_{Tmin}$, la señal generada puede ser calculada satisfactoriamente. Tomemos, por ejemplo, los mismos parámetros que el ejemplo anterior: 10 pulsos por segundo, 500ms de duración, 20Hz de frecuencia del tono y 10um de amplitud. Lo anterior da por resultado 10·0.5=5 pulsos, más el adicional, dando por resultado 6 pulsos en el estímulo; una duración real de 500ms más 1/20Hz=0.05s igual a 550ms, y la duración de cada tiempo muerto es aleatoria. Esta señal podría verse como la mostrada en la Figura V-10.

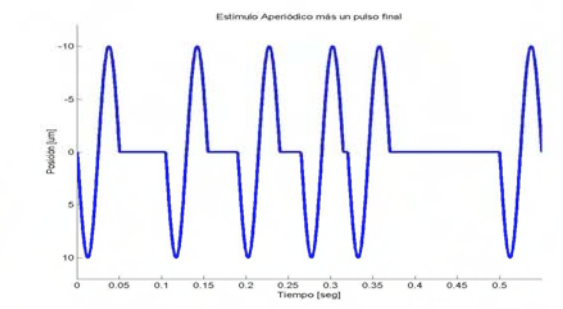


Figura V-10. Estímulo aperiódico con tiempos muertos aleatorios y pulso final adicional.

V.1.8.3.7. Estímulo basado en número de pulsos

Es prácticamente igual al estímulo periódico más un pulso final sólo que en este caso no se agrega el pulso final.

Si $F_e > F_t$, el estímulo no puede ser calculado ya que el pulso duraría más que el tiempo máximo permitido, en este caso, dicho tiempo máximo lo da el recíproco de F_e.

Si $F_e = F_t$, no existen tiempos muertos en el estímulo, la forma de onda resultante es continua en el tiempo que dura el estímulo y se presentan el número de pulsos resultante del producto de $F_e \cdot D_e$. La Figura V-8 muestra un estímulo con las características siguientes: 10 pulsos por segundo, 500ms de duración, 10Hz de frecuencia de tono y 10um de amplitud de tono; por lo tanto, tenemos un estímulo de 10·0.5=5 pulsos sin tiempos muertos.

Si $F_e < F_t \leq F_{Tmin}$, el estímulo no podrá ser generado satisfactoriamente ya que la señal de interrupción que genera el pulso y el tiempo muerto no podrá ser procesada por el sistema operativo.

Si $F_t \geq F_{Tmin}$, la señal generada puede ser calculada satisfactoriamente. La Figura V-11 muestra el estímulo cuyas características son: 10 pulsos por segundo, 500ms de duración, 20Hz de frecuencia del tono y 10um de amplitud. Lo anterior da por resultado 10·0.5=5 pulsos en el estímulo y una duración real de 500ms. Nótese la diferencia entre este estímulo y el presentado en la Figura V-9.

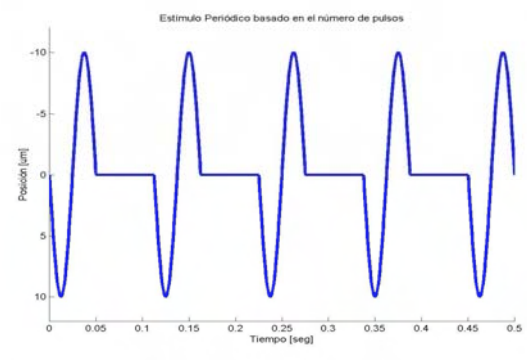


Figura V-11. Estímulo basado en el número de pulsos

Estímulo bipulso

En este tipo de estímulo sólo se presentan 2 pulsos conformados, cada uno, de medio ciclo del tono. En este caso, el usuario sólo especifica la duración del estímulo, la frecuencia del tono y la amplitud. Para que este estímulo se genere adecuadamente, la frecuencia del tono deberá estar dada por la Ecuación V-2:

Donde D_{TMmin} es la duración mínima de tiempo muerto y es igual a 3ms. Si la duración del estímulo es de 500ms, la frecuencia del tono de 10Hz y la amplitud es de

$$F_t \geq \frac{1}{D_e - D_{TM\,\min}}$$

Ecuación V-2

10um, el estímulo resultante sería como el que se muestra en la Figura V-12.

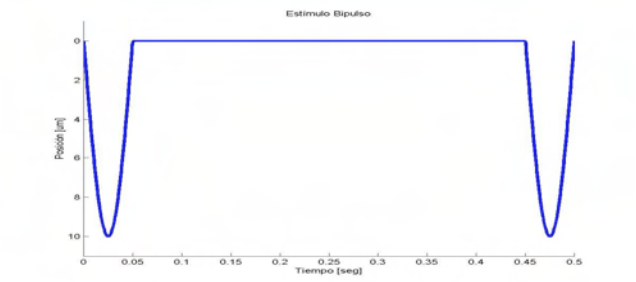


Figura V-12. Estímulo bipulso

V.1.8.3.8. Estímulo definido por el usuario

Con este tipo de estímulo el usuario puede definir la duración de cada tiempo muerto presente en el estímulo respetando, desde luego, el mínimo tiempo muerto ya antes mencionado. En este caso, el investigador tiene que especificar los parámetros del tono correspondientes a la frecuencia y la amplitud; con ello, se generan pulsos de un ciclo de tono, entre los cuales existe un tiempo muerto. Cabe mencionar que la frecuencia y la duración del estímulo se obtienen a partir de la frecuencia del tono y de la duración de cada tiempo muerto especificado por el usuario.

Por ejemplo, si el usuario especifica que la frecuencia y la amplitud del tono son 35Hz y 20um, respectivamente, y las duraciones de tiempo muerto siguientes: 20ms, 30ms, 40ms, 40ms, 30ms, 20ms, entonces el primer tiempo muerto tendrá una duración de 20ms, el segundo durará 30ms y así hasta llegar al último cuya duración será de 20ms, 6 tiempos muertos en total en el estímulo. Como el número de tiempos muertos siempre es uno menor que el número de pulsos, tenemos que el estímulo contendrá N_p = 6+1 = 7 pulsos de un ciclo de tono cada uno. La duración total del estímulo se obtiene sumando N_p / F_t más la suma total de tiempos muertos: 7 / 35 + (0.02 + 0.03 + 0.04 + 0.04 + 0.03 + 0.02) = 0.2 + 0.18 = 0.38s. Aunque la frecuencia de estímulo no representa un número entero en este caso, se puede estimar dividiendo el número de pulsos entre la duración total del estímulo: 7/0.38≈18 pulsos por segundo; sin embargo esta interpretación se deja al criterio del investigador. El estímulo resultante es mostrado en la Figura V-13.

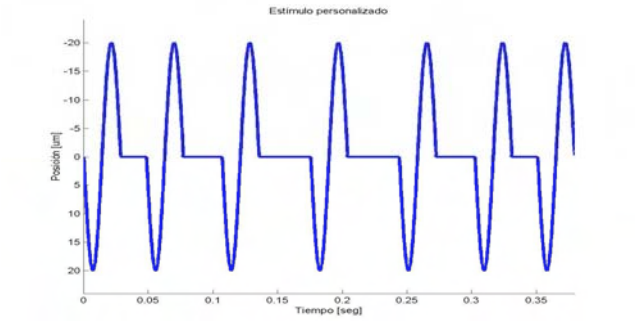


Figura V-13. La duración de cada tiempo muerto es definida por el usuario

V.1.8.4. Generación de los estímulos

Una vez entendidos los diferentes tipos de estímulo pasamos al problema de la generación de estímulos. Habíamos comentado antes que éstas señales eran generadas por el sintetizador de señales; sin embargo, a pesar de ser un dispositivo avanzado, no provee la generación de partes de señal como los pulsos utilizados en los estímulos antes vistos, lo que el sintetizador hace es generar una señal de manera continua. De la señal generada en cuestión, se puede cambiar rápidamente la amplitud, la frecuencia y la fase inicial pero no la forma de onda. Por otro lado, tenemos el problema de cómo generar pulsos de alguna señal y tiempos muertos, además de su temporizado. En primera instancia, pensamos en temporizar cada pulso y tiempo muerto de manera semejante a como lo hicimos con los estados, el problema fue el tiempo que tardaba en atenderse la interrupción: cuando se produjera una interrupción debida al final de un pulso de señal (o al final de un tiempo muerto), ésta debía de terminar la generación del pulso (o tiempo muerto) y comenzar a generar el siguiente tiempo muerto (o el siguiente pulso). Dada la incertidumbre en la atención de la interrupción, existía un tiempo adicional entre cada fin de pulso (o de tiempo muerto) que hacía que el tiempo de estímulo no fuera el indicado por el investigador. Además, también teníamos que controlar la fase inicial del pulso porque de lo contrario no tendríamos el control de la posición en la cual comenzaría la señal generada.

El establecimiento de los parámetros se realizó vía el protocolo GPIB, para lo cual contábamos con la tarjeta GPIB II PCI, instalada en la computadora, y el sintetizador, que contiene un puerto GPIB por el cual recibe comandos remotamente. Por lo tanto, dicha especificación de parámetros se realizó desde la computadora.

El temporizado de cada pulso (o tiempo muerto) se llevó a cabo de manera muy similar al temporizado de los estados, sólo que aquí utilizamos dos contadores, uno para los pulsos y otro para los tiempos muertos. La diferencia fue que estos contadores deberían comenzar su funcionamiento con una señal externa y no mediante un comando como sí lo hacen los contadores que temporizan a los estados. La idea fue hacer que el contador encargado de temporizar los pulsos (CTRp) hiciera funcionar al temporizador de tiempos muertos (CTRm) y viceversa. Para ello, los dos contadores se configuraron para que pudieran funcionar cuando la terminal de habilitación (GATE) de uno de ellos recibiera una señal formada por la salida del otro, es decir, cuando CTRp terminara su cuenta, activaría a CTRm y éste a su vez activaría nuevamente a CTRp, así sucesivamente hasta terminar de generar los pulsos y tiempos muertos que forman el estímulo. Lo anterior también deja ver que mientras un contador funciona el otro está detenido, diseñamos entonces un circuito que permitiera combinar las salidas de estos contadores junto con otra que indicaría qué señal generar: un pulso o un tiempo muerto. La Figura V-14 muestra las señales que nos permitirían generar un estímulo. Se puede ver un estímulo formado por tres pulsos y dos tiempos muertos. Se observan también las señales de salida y de habilitación de los contadores antes mencionados. Esta vez la señal de interrupción (EXTIRQ2) la formarían las salidas de los dos contadores, de tal manera que tendríamos una interrupción cada vez que alguno de los dos contadores terminara su operación. Podemos ver que la señal de habilitación GATEp coincide con los pulsos y, de manera semejante, GATEm habilita a CTRm sólo cuando existe un tiempo muerto.

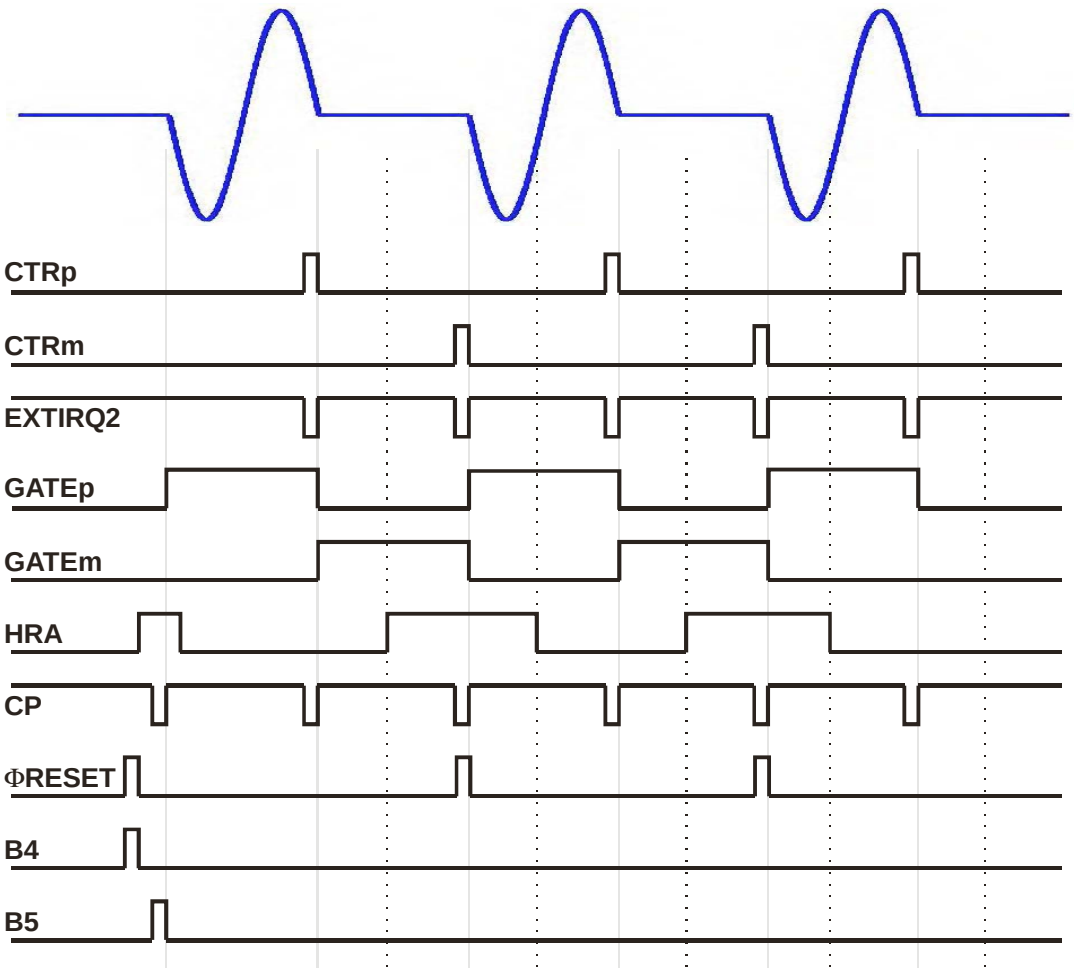


Figura V-14. Señales de control utilizadas en la generación de un estímulo

Por otro lado, el HP-8904A cuenta con 16 registros de salto de señal en los cuales se pueden tener hasta 16 señales totalmente diferentes (excepto la forma de onda). En cada registro, se especifica la frecuencia, amplitud y fase de cada señal (cada tono en nuestro caso). Además, el sintetizador también cuenta con un puerto digital con el que se puede cambiar la dirección del registro de salto, así es como podemos cambiar rápidamente la señal en la salida del sintetizador. Usando el puerto digital se puede cambiar la dirección de salto con tiempos mínimos de 8μs. La solución que encontramos fue la de predeterminar la dirección 0 del registro a una señal con amplitud cero y formar así nuestro tiempo muerto; de tal manera que las direcciones 01 a la 15 contendrían las direcciones de los pulsos correspondientes a los estímulos a utilizar.

En base a lo anterior, diseñamos un circuito que permitiera controlar los contadores y la dirección de registro de salto de señal del puerto digital del sintetizador. Este circuito es mostrado en la Figura V-15.

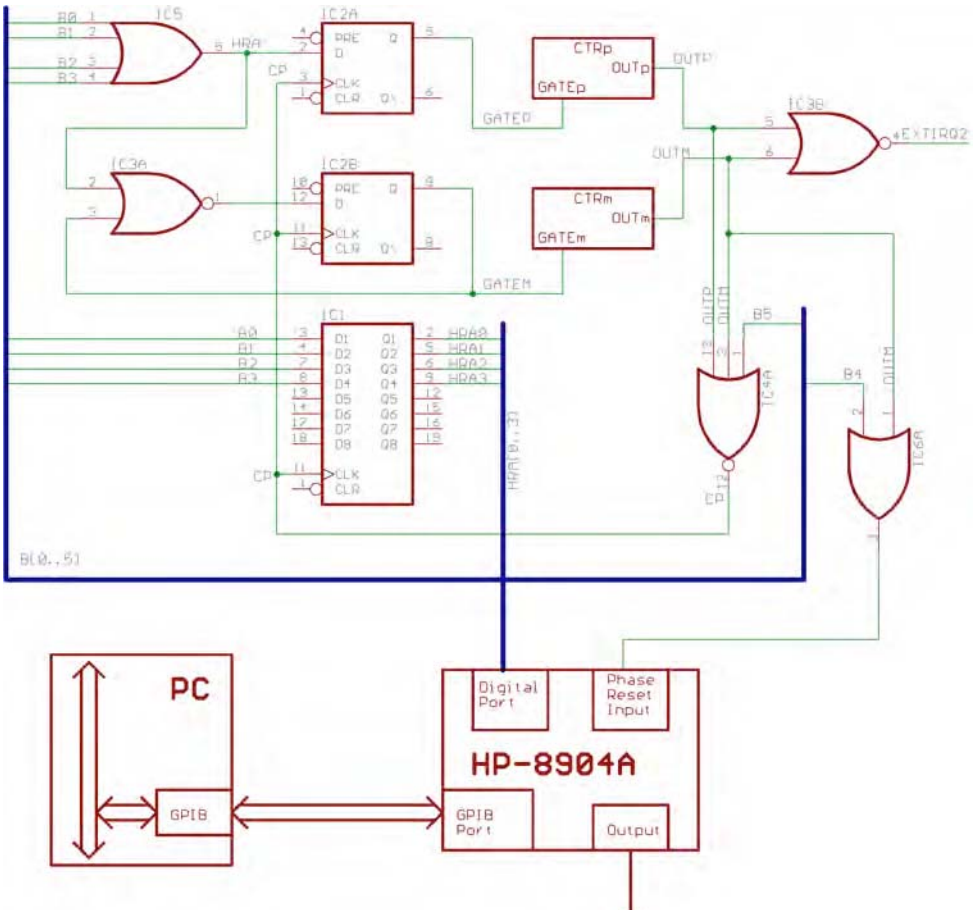


Figura V-15. Circuito usado para la generación del estímulo

En este diagrama, las señales B0, B1, B2, B3, B4 y B5 provienen del puerto digital B de la tarjeta PC-TIO 10; las primeras cuatro líneas contienen la dirección del registro del puerto digital del sintetizador, de este modo, cuando la dirección sea 0 el sintetizador generará un tiempo muerto, y cuando la dirección no sea 0 se generará un estímulo. Ahora bien, esta dirección se retiene en un registro y se deja pasar al puerto digital del HP-8904A hasta que alguno de los contadores provea una señal de reloj e indique que hay que cambiar de un pulso a un tiempo muerto o viceversa. De manera parecida, las líneas B0...B3 se combinan para formar la señal HRA, la cual sirve para determinar que contador se habilitara a continuación. La señal de reloj (CP) utilizada para actualizar la salida de los circuitos retenedores se forma a partir de las salidas de los dos contadores; sin embargo, al principio no hay ningún contador funcionando por lo que fue necesario usar la línea B5 para proveer un primer pulso de reloj. Por otro lado, debemos proveer un pulso de sincronización de fase al sintetizador para que el pulso generado comience en 0 grados, este pulso lo tomamos del contador CTRm ya que cada vez que éste termina sigue un pulso; no obstante, al principio no está funcionando éste contador por lo que hay que dar un primer pulso de sincronización (ΦRESET) mediante la línea B4. Las señales antes mencionadas se pueden ver en la Figura V-14. Basándonos en lo anterior, comenzamos a generar el estímulo de la forma siguiente:

1. Preparamos al contador de pulsos CTRp para que comience a funcionar cuando su terminal de habilitación GATEp sea activada.
2. Aplicamos un primer pulso de sincronización a la terminal Phase Reset del sintetizador mediante la línea B4.
3. Establecemos la dirección del registro de salto de señal, la cual contiene los parámetros que generarán la señal de pulsos del estímulo. Esta dirección es asegurada mediante el registro y mantenida ahí hasta que llega una señal de disparo CP provista por la cuenta final de alguno de los contadores.
4. Presentamos un pulso digital en B5 para formar una primera señal CP que haga que la dirección antes mencionada pase al puerto digital del sintetizador, el cual genera el pulso en ese momento.
5. Preparamos al contador CTRm para que funcione cuando su terminal GATEm tenga un nivel alto.
6. Llevamos la dirección del tiempo muerto al registro, de tal manera que cuando CTRp termine de contar genere la señal CP que pasará la dirección del tiempo muerto al puerto digital del sintetizador; cuando eso suceda, la salida del generador cambia a una señal de amplitud cero y se generará el tiempo muerto, si es que lo hay.
7. Cuando el primer pulso ha terminado, CTRp provoca que el dato en la entrada del registro (la dirección del tiempo muerto) pase al puerto digital del sintetizador y dicho dato queda asegurado ahí hasta que no haya otra señal CP, estamos generando entonces un tiempo muerto.

Una vez que se ha generado el primer pulso, la secuencia en la generación del estímulo es controlada mediante una señal de interrupción (EXTIRQ2) provocada por la salida de CTRp o CTRm. Cada vez que uno de estos contadores termina su cuenta se produce una interrupción cuya atención no es inmediata; en la Figura V-14 se puede ver que las líneas verticales punteadas representan el momento en que nuestro programa atiende la interrupción. El manejador de este mensaje hace lo siguiente:

1. Se determina, en base a una bandera, si la última interrupción fue provocada por un pulso o por un tiempo muerto.
2. Si la última interrupción fue provocada por un pulso, primero se determina si el número de pulsos se ha completado, en cuyo caso se deshabilita el envío de mensajes, se apagan los contadores y se termina el estímulo. Si el número de pulsos no se ha completado, se carga la dirección del tono en el registro y se prepara el contador CTRp. Lo anterior hace que cuando el tiempo muerto haya terminado, la salida de CTRm forme una señal CP que pasa la dirección del tono al puerto digital del sintetizador y además active a CTRp. La bandera antes mencionada es actualizada para que indique que la última interrupción se debió a un pulso.
3. Si la última interrupción se debió a un tiempo muerto, debemos cargar la dirección 0 al registro IC9 y preparar al contador CTRm. Así, cuando CTRp termine, su salida provocará que la dirección del tiempo muerto pase al puerto digital y se habilite a CTRm. La bandera de la fuente de última interrupción es actualizada a falso.

Observaciones

Como la comunicación por GPIB no es muy precisa y tiene un valor de alrededor de medio segundo, no podemos cargar los parámetros del tono que forma la señal del pulso justo antes de que ésta se genere.

Un punto importante a señalar, es que siempre terminamos la generación del estímulo con un pulso, nunca con un tiempo muerto, por eso, el número de tiempos muertos es uno menor que el de pulsos.

Los tiempos muertos están representados por un arreglo de N_m tiempos muertos, cada vez que se entra al manejador de mensajes y se tienen que establecer los parámetros del siguiente tiempo muerto a generar, se toma el elemento siguiente de dicho arreglo.

Cuando se tiene un estímulo de tipo aperiódico, los elementos del arreglo de tiempos muertos son calculados sólo al principio, en las corridas siguientes dichos elementos son permutados aleatoriamente.

V.2. Interfaz de Usuario

El investigador debe contar con una herramienta que le permita definir los datos del experimento que llevará a cabo; asimismo, debe tener el control de la ejecución de dicho experimento y guardar los datos que surjan del mismo.

Con lo anterior en mente, primero entendimos la manera final en la cual sería usado el sistema, posteriormente, atacamos cada parte de acuerdo a los requisitos identificados y recursos disponibles. Esto implicó una descripción en alto nivel que indicara las operaciones básicas disponibles al usuario, y una descripción de nivel más bajo que se encargara de la generación y control de las señales involucradas en el experimento.

V.2.1. Desarrollo de aplicaciones para Windows usando Visual C++

La plataforma sobre la cual debía funcionar nuestro sistema es Windows 98; este popular sistema operativo ofrece interfaces gráficas amigables al usuario que hacen más intuitivo el manejo de programas. Por otro lado, la programación en Windows no es fácil dado que implica conocer a fondo las técnicas utilizadas en el manejo de varios procesos, el dibujo en pantalla, la creación de ventanas, etc. Por ello es que la programación ha tenido que evolucionar para que el programador se concentre más en su aplicación que en la forma en la cual funciona el sistema operativo. Para este efecto, se han creado entornos de desarrollo que provean los componentes mínimos de una aplicación, uno de ellos es Visual C++. Junto con este poderoso sistema de desarrollo, se ofrecen un conjunto de librerías que implementan varias operaciones relacionadas a tareas básicas y de propósito general, tal como la librería de clases fundamentales de Microsoft (MFC). Por último, Microsoft también desarrolla la idea de aplicaciones orientadas a objetos que separen las diferentes tareas llevadas a cabo. En este tenor, surge la arquitectura documento-vista en la cual los datos son separados de la manera en la cual se visualizan; así, se pueden tener distintas vistas de los datos en cuestión.

De manera más detallada, Visual C++ provee asistentes de creación de aplicaciones que recolectan la información necesaria para crear un esqueleto básico para una aplicación Windows, de tal manera que el programador añade comandos y controles según sus necesidades. En nuestro caso, identificamos los 4 objetos básicos que necesitaríamos en base al tipo de programa: la aplicación, la ventana principal, la vista y el documento. Aquí, la aplicación crea a los otros tres objetos y se encarga de establecer las configuraciones e información necesarias para el sistema operativo; la ventana principal se encarga de proveer un marco de trabajo general para los otros dos últimos objetos, además, contiene a la vista; esta última se comunica con el documento para recuperar los datos y representarlos en la pantalla; el documento se encarga del manejo y almacenamiento de los datos.

Así, comenzamos a trabajar de manera aislada cada objeto que creímos conveniente debía existir en la aplicación. Trabajar de esta manera fue importante ya que nos permitió concentrarnos en pequeños módulos que a final de cuentas integrarían a todo el sistema. Una vez que las partes principales estaban trabajando en su forma más básica, procedimos a trabajar en la parte visual de la interfaz gráfica que el usuario utilizaría, es decir, en proveer controles y comandos comunes de Windows que un usuario común reconoce.

V.2.2. Comandos disponibles al usuario

Las operaciones básicas que el investigador necesitaba fueron:

Establecer los parámetros del experimento a partir de un archivo de Excel
: Implicaba establecer los comandos y funciones necesarias para que el investigador seleccionara el archivo que requería para la ejecución del experimento. Los datos del archivo de Excel definirían por completo la información del experimento.

Fijar la manera en la cual se ejecutaría el experimento
: En esta parte, el investigador elegiría las clases a ejecutar, el número de ensayos por clase y el orden de ejecución (secuencial o aleatorio) de los ensayos.

Ejecutar el experimento
: Una vez que todos los datos y las configuraciones habían sido establecidas, el experimento estaba preparado para empezar, entonces, el investigador decidiría comenzar con la ejecución de los ensayos. Cuando los ensayos estuvieran ejecutándose, el investigador también podría decidir pausar el experimento, lo cual significaría suspender la ejecución de ensayos hasta que el investigador la reanudase nuevamente. Para fines de entrenamiento, también podría ser que el investigador repitiera indefinidamente un solo ensayo, para ello, debía proveerse un comando. Finalmente, el investigador debía poder abortar el ensayo actual, lo que significaría terminar en ése momento con dicho ensayo.

Almacenamiento de los datos
: Los datos resultantes del experimento deberían ser guardados en un archivo y ser expuestos al sistema colector de datos para que este llevara a cabo el análisis de los mismos.

Más adelante se verá cómo fueron creados estos controles, en tanto, se discute cómo se desarrollaron las partes principales que componen al programa.

V.2.3. Desarrollo de clases utilizadas en el programa

La programación orientada a objetos ofrece un conjunto de técnicas que hacen el desarrollo de sistemas más definido en cuanto a que un sistema muy grande se compone a su vez de sistemas más pequeños. Este tipo de programación organiza los programas de modo que reflejen la forma de organización de los sistemas en el mundo real. En este sentido, un sistema está compuesto de diferentes objetos que colaboran entre sí para llevar a cabo una tarea dada. Además, cada objeto sabe como comportarse y comunicarse con otros sin necesidad de que se conozca su funcionamiento interno, es decir, la manera en la cual lleva a cabo su función queda oculta (encapsulada). Siguiendo este enfoque, nos dedicamos a diseñar el sistema identificando los diferentes objetos que lo componen. Como se mencionó antes, el experimento está compuesto por ensayos y éstos a su vez por estados. Con estas

divisiones bien marcadas, empezamos a desarrollar las clases esenciales que utilizaríamos. Para empezar, un objeto experimento llevaría el control de los ensayos actuando sobre éstos y administrándolos; a su vez, cada ensayo debería encargarse de sus estados, y cada estado ser responsable de manejarse a sí mismo.

V.2.3.1. Desarrollo de la clase CExperimento

Esta clase es fundamental ya que se encargaría de todo el manejo y control de señales del experimento, interactuar con los ensayos, los estados y los objetos propios de la interfaz de Windows. En primer lugar, hicimos que poseyera los atributos necesarios para funcionar, dentro de los cuales se contaban procedimientos de administración de datos, de iniciación y de control en la ejecución del experimento.

Atributos de la clase

1. **Número de estados.** Es importante saber cuántos estados componen a los ensayos a presentar en el experimento, de esta forma se puede asignar la memoria suficiente para ellos.
2. **Número de clases.** Este atributo determina el número de clase que se tienen en el experimento.
3. **Número de ensayos por clase.** También es importante conocer cuántas corridas se presentarían en el experimento lo que permitió preparar a la vista para recibir la cantidad de datos necesaria.
4. **Lista de ensayos.** Representa un arreglo de datos que sirve para recuperar un ensayo en particular, de este modo podemos trabajar con cualquier ensayo en cualquier momento. En este caso, un ensayo es la ejecución de una clase que representa los estados que componen una tarea en particular; de tal forma que una instancia de dicha clase representa un ensayo.
5. **Arreglo de tipos de estado.** Una vez que sabíamos el número de estados, fue necesario conocer también el tipo de estados que teníamos.
6. **Lista de ejecución.** Cuando el orden de ejecución es secuencial no teníamos ningún problema, sólo teníamos que hacer referencia a cada elemento de la lista de clases en orden ascendente. El problema surgió cuando el orden de ejecución fue aleatorio, la solución que encontramos fue una lista de ejecución cuyos elementos representaran los índices de la lista de clases; así, si el orden de ejecución era aleatorio se llevaría a cabo una permutación aleatoria de ésta lista, en caso de que el orden fuera secuencial, arreglamos nuevamente esta lista en orden ascendente. Finalmente, recuperamos cada índice de ésta lista para saber que ensayo debíamos ejecutar.
7. Variable que indica el tipo de **orden de ejecución.** Con esta variable conocemos si el orden de ejecución es aleatorio o no.
8. Variable que represente si el **control** es **pasivo** o no. En caso de que el investigador quiera ejercer un control de tipo pasivo, esta variable sería ajustada adecuadamente y tomada en cuenta en los eventos.
9. **Comunicación con la vista.** Siguiendo el principio de envío de mensajes, incorporamos una variable que hiciera referencia a la vista, a la cual enviamos mensajes en base a los eventos que ocurrieran.
10. **Índice de ensayo.** Variable utilizada para hacer referencia a una clase en particular de la lista de clases.

11. **Índice de selección.** Variable utilizada para recuperar un índice de ensayo de la lista de ejecución.
12. **Ensayo actual.** Cuando el experimento está llevándose a cabo, esta variable apunta al ensayo actual en cuestión.
13. **Variable de control para abortar.** Con el experimento en ejecución, esta variable es ajustada cuando el usuario decide abortar.
14. Número de **ensayos ejecutados** actualmente.
15. Número de **corridas ejecutadas** actualmente.
16. **Número de aciertos** que el mono ha tenido hasta el momento.

Comandos de administración de datos

1. El **constructor** de la clase se dedica a iniciar las variables antes descritas adecuadamente. Por el contrario, el **destructor** se encarga de llevar a cabo la limpieza de la memoria que fue asignada durante el experimento.
2. **Limpiar el experimento.** Necesario cada vez que el usuario decida detener y abandonar el experimento actual para redefinir otro experimento basado en otro archivo de Excel, o bien cuando el programa se cierre.
3. **Agregar tipos de estado, borrar tipos de estado.** Comandos necesarios ya que la asignación de memoria se lleva a cabo dinámicamente.
4. **Agregar** ensayo, **eliminar** ensayos.

Comandos de Ejecución

1. Es necesario **iniciar** adecuadamente las variables antes descritas para que el experimento se encuentre en un estado conocido.
2. **Ejecutar ensayo.** Esta función sirve para ejecutar un ensayo de la lista de clases.
3. **Repetir ensayo actual.** Comando necesario en caso de que el ensayo actual tenga que ser repetido.
4. **Despacho de eventos.** Aunque la recepción de eventos se lleva a cabo en la clase vista de nuestra aplicación, ésta delega la responsabilidad del manejo de eventos a la clase experimento.
5. **Orden de ejecución.** Esta función se encarga de permutar o arreglar la lista de ejecución en cada final de corrida.
6. **Ensayo siguiente.** Cada vez que un ensayo finaliza, este procedimiento es llamado para ejecutar el ensayo siguiente.
7. **Fin de corrida.** Cuando se ha completado una corrida es necesario llevar a cabo ciertos ajustes y cambiar nuevamente los elementos de la lista de ejecución para la siguiente corrida. Además, se tiene que notificar a la clase vista de la aplicación para que actualice los datos en pantalla correspondientes a la corrida.
8. **Terminar experimento.** Simplemente notifica que el experimento se ha completado.
9. **A cero.** Esta función se llama cada vez que se aborta un ensayo y, básicamente, se encarga de llevar la punta desde la posición actual hasta la posición de 0 micras.

En cuanto a la dinámica del experimento es importante observar que una vez que el experimento está preparado ocurre lo siguiente:

1. Se ejecuta el ensayo recuperado de la lista de clases y se espera a que ocurra un evento producido por el mono o por un fin de estado.

2. Cuando se produce el evento mencionado, éste es enviado a la vista quien cede la responsabilidad a la clase CExperimento para que despache dicho evento. Aquí es donde se determina si el ensayo actual debe continuar con su ejecución normal o se tiene que abortar.
3. En caso de que el ensayo que se estaba ejecutando haya finalizado correctamente, y todavía haya ensayos por ejecutar, se ejecuta el ensayo siguiente, el cual fue recuperado de la lista de ejecución. Si la lista de ejecución no contiene más elementos a ejecutar, quiere decir que se ha llegado al final de una corrida y es necesario comenzar otra.
4. En caso de que el ensayo actual tenga que abortarse, primero se deja terminar el estado actual y después se aborta el ensayo.
5. Una vez terminados todos los ensayos y corridas, se reporta el final del experimento y se actualizan los datos en pantalla.

Por otro lado, las funciones de administración de datos, se llaman cada vez que el experimento se cambia, es decir, cuando el usuario abandona el experimento actual para ejecutar otro. Entonces se lleva a cabo una limpieza de la memoria y se asignan nuevamente los valores adecuados.

V.2.3.2. Desarrollo de la clase CEnsayo

Un experimento esta compuesto de ensayos, los cuales representan ejecuciones de clases que definen una tarea en particular. En términos de programación orientada a objetos, la clase CExperimento tendría una asociación de 1 : N con la clase CEnsayo, es decir, la primera contiene múltiples instancias de la segunda, y ésta última pertenece a la primera.

Aclarado lo anterior, pasamos al desarrollo de la clase que representa un ensayo. En primer lugar, una tarea determinada está definida por las clases que la componen y, durante la ejecución del experimento, cada vez que una clase se ejecuta tenemos un ensayo. Por otro lado, un ensayo debe ser responsable de ejecutar los diferentes estados que lo componen y de administrar sus datos para que reflejen su estado y la información que surgió de dicho ensayo.

Así, los atributos principales de esta clase serían los siguientes:

Atributos de la clase

1. **Apuntador al objeto padre** que representa la clase CExperimento. Con esto se crea la asociación de esta clase con el objeto que la posee. Establecemos así una comunicación entre objetos.
2. Una variable que **identifique a este objeto**. Recuérdese que el objeto padre posee una colección de objetos ensayo, por lo cual es necesario asignar un número que identifique a este ensayo de los otros.
3. Una variable que indique **cuántos estados posee este ensayo**. Es importante conocer esta variable al momento de crear los estados que componen este ensayo.
4. **Lista de estados**. Representa una colección de estados que serán ejecutados durante el ensayo.
5. Una variable que apunte al **estado actual** en ejecución.
6. Una variable que indique el número de registro de memoria del sintetizador de señales en donde se alojarán los datos de los estímulos de este ensayo. En caso de que el ensayo contenga estados de tipo estímulo, estos serán guardados en los

registros de salto de señal del sintetizador, dichos registros pertenecen a un registro de memoria de los 32 disponibles.

7. Variables que indiquen si **la respuesta del mono** en este ensayo fue acertada o no, así como la causa por la cual el ensayo terminó (fin satisfactorio, el usuario abortó, etc.)

Comandos de administración de datos

1. De manera semejante a la clase CExperimento, la colección de estados se crea dinámicamente cuando se elige un archivo de experimento. Para hacer lo anterior primero se debe desalojar la memoria previamente asignada.

Comandos de ejecución

1. **Comenzar ejecución de ensayo**. Cuando un ensayo inicia su ejecución, es necesario determinar si contiene estímulos (lo cual es muy probable), esto se debe a que el tiempo que le toma al sintetizador establecer los parámetros de las señales no esta bajo nuestro control. Entonces, tenemos que pasar dichos parámetros justo antes de empezar a generarse el primer estado, el cual es recuperado de la lista de estados. También envía un mensaje a la vista para indicarle que el ensayo está a punto de iniciar.
2. **Recomenzar ensayo**. Si por alguna razón se tiene que ejecutar nuevamente el ensayo actual, se recupera el primer estado de la lista de estados y se invoca la función que genera al primer estado. De esta forma, no es necesario pasar los parámetros nuevamente al sintetizador de señales. La vista es informada de que el ensayo será reiniciado.
3. **Ejecutar estado siguiente**. Este método recupera el siguiente estado de la lista de estados, dicho estado comienza a generarse. En caso de que ya no existan estados, se finaliza el ensayo. Aquí también se determina si el ensayo tiene que abortarse por orden del usuario, es decir, un ensayo se aborta cuando el estado actual se terminó de generar.
4. **Abortar ensayo**. Esta función detiene la ejecución del estado actual y lleva la punta a la posición de 0 micras. Esta función es invocada cuando el ensayo tiene que ser abortado por un evento no producido por el usuario. Se envía un mensaje a la vista para informar que el ensayo ha sido abortado.
5. **Finalizar ensayo**. Se detienen los dispositivos, se deshabilita el envío de mensajes de interrupción y se envía un mensaje a la vista para indicarle que el ensayo ha terminado.

Dado que en un ensayo la ejecución de los estados es meramente secuencial, la dinámica asociada es más simple:

1. Empezamos ejecutando el primer estado y esperamos a que el objeto padre indique la siguiente acción.
2. El objeto padre determina la siguiente acción, en base a la información de los objetos hijos: generar estado siguiente, repetir ensayo, abortar o finalizar.
3. Finalizar el ensayo cuando no haya más estados que generar.
4. Es importante señalar que un ensayo puede llegar a su fin por las razones siguientes: fin satisfactorio, por un movimiento no permitido por parte del mono o porque no se llevó a cabo un movimiento esperado.

5. Este objeto también envía mensajes a la vista para notificarle la manera en la cual se está llevando a cabo el ensayo, de tal manera que la vista actualiza y notifica al usuario de dichos eventos.

V.2.3.3. Desarrollo de la clase CEstado

De manera semejante al experimento, un ensayo está compuesto por estados que generan señales y están definidos por eventos. En este caso, la clase CEnsayo guarda una asociación de 1 : N objetos de tipo CEstado; por lo tanto, la clase CEstado es hija de la clase CEnsayo, la cual contiene una lista de estados. En el desarrollo de la clase CExperimento, se mencionó que ésta contiene una lista de ensayos, los cuales son responsables de manejarse a sí mismos. El punto es que en el caso de la clase CEnsayo, es de esperarse que todos los objetos sean del mismo tipo: están compuestos de estados; sin embargo, en el caso de los estados no podemos esperar que sean de un mismo tipo; esto significa que existen diferentes tipos de estado dependiendo de la tarea en cuestión, lo cual podemos constatar de la Figura V-1: los estados E0, E2, E4, E6, E8 y E9 son del mismo tipo; los estados E1 y E7 son de tipo rampa; y los estados E3 y E5 son estímulos. Como vimos en la sección V.1.8, la forma en que cada señal es generada difiere de las otras; no obstante, cada estado tiene características en común. Esto nos llevó a aplicar otra de las técnicas de programación orientada a objetos conocida como generalización de una clase. En esta técnica, se crea una clase base dotada de las características comunes a todas las otras, después, a partir de dicha clase, se derivan otras que se encargarán de especializarse en una tarea en particular. Por ello, en esta parte discutimos las funciones básicas que realiza un estado y después tratamos la generalización de esta clase.

Antes de continuar, debemos hacer mención de un concepto hasta ahora no tocado: la existencia de ***ventanas de tiempo***, las cuales imponen un rango de tiempo en el cual el mono debe realizar un tipo de movimiento, de no ser así el ensayo se reinicia. Esto se hace con el fin de evitar que el mono se adelante (anticipe) o retrase al realizar un movimiento o expresar alguna respuesta.

La Figura V-16 muestra parte de una tarea en la cual el mono debe colocar su mano sobre la palanca en el tiempo especificado por la ventana de tiempo; si el mono coloca su mano antes de T_{min} ocurre un evento llamado ***Early***, y si la coloca después de T_{max} se tiene un evento ***Late***. En caso de presentarse cualquiera de estos eventos, el ensayo actual se aborta y empieza de nuevo. De tal manera que el evento KH debe presentarse en este rango de tiempo para que el ensayo pueda continuar con sus demás estados. Una ventana de tiempo aparece cuando el mono debe llevar a cabo un movimiento, es decir, cuando la punta baja (PD) y le señala al mono que coloque su mano sobre la palanca (KH), cuando la punta se retira (PU) e indica al mono quitar su mano de la palanca (KR), y cuando tiene que presionar un botón (PB). Tendríamos entonces tres ventanas de tiempo. En caso de que el evento correspondiente ocurra dentro de la ventana de tiempo, se da inicio al estado siguiente.

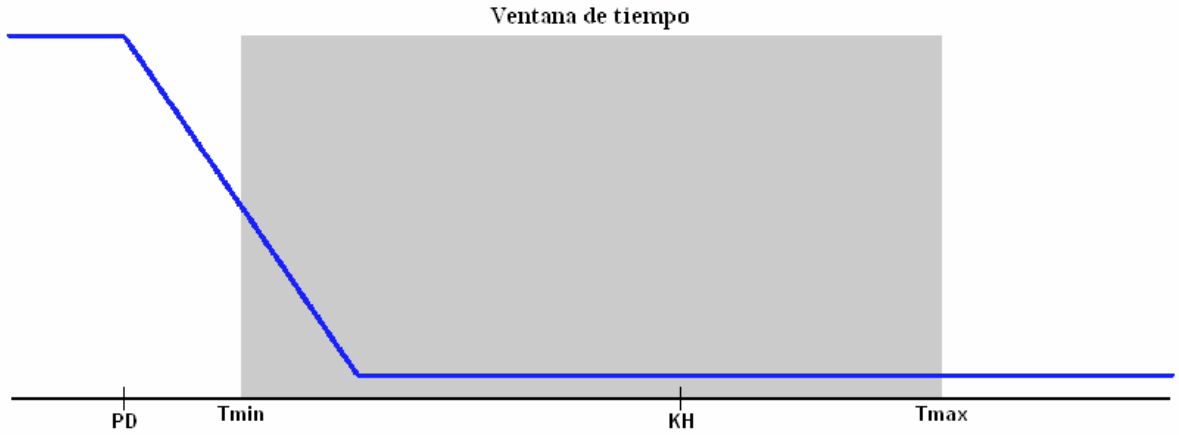


Figura V-16. La ventana de tiempo aparece en gris

Conviene recordar también que ciertos estados pueden cambiar aleatoriamente su duración de ensayo a ensayo, esto es, el usuario especificaría que la duración de un estado estaría limitada a un tiempo mínimo y máximo, pero dicha duración sería aleatoria. Por otro lado, el investigador también define un parámetro que cuantiza dicha duración; por ejemplo, si el investigador así lo establece, un estado tendría una duración mínima y máxima de 0.5 s y 1.5 s, respectivamente, y un ***Quantum*** de 100 ms, lo cual daría como resultado que este estado en particular pudiera tener cualquiera de las siguientes duraciones de ensayo a ensayo: 0.5, 0.6, 0.7, ... , 1.3, 1.4 o 1.5 segundos, determinadas éstas de manera aleatoria.

Tomando en cuenta que esta clase es la que trabajará directamente con los dispositivos electrónicos, y la que se encargará de decidir la acción a tomar en base al tipo de evento que llegase a ocurrir, contamos ahora con la información suficiente para desarrollar esta clase base:

Atributos de la clase

1. Un **apuntador al objeto padre** CEnsayo que posee esta clase. Establecemos entonces la asociación de objetos para que puedan comunicarse. Nótese que, mediante este atributo, éste objeto puede comunicarse incluso con el objeto experimento.
2. Un **identificador** que distinga a este objeto de los demás que posee el objeto ensayo en su lista de estados.
3. Variables que representen las **duraciones** de estado **mínima** y **máxima**, el **quantum** y, para los dispositivos, las variables cuyo **tipo de dato** es el adecuado.
4. Dado que los movimientos pueden presentarse en cualquier momento, cada estado debe saber si el **movimiento** es permitido o no. Por ello, es necesario asignar variables que indiquen si el movimiento está permitido o no.
5. En caso de permitirse el movimiento en este estado en particular, es necesario contar con variable de estado que indiquen dicho **movimiento** tanto en la **palanca** como en los **botones**.

6. Tal como se dijo antes, el movimiento también está restringido a **ventanas de tiempo**, en este caso, necesitamos variables que indiquen los tiempos mínimo y máximo de dicha ventana.
7. También es muy importante saber los **tiempos de respuesta** del mono en caso de un movimiento permitido.
8. Por último, una **variable que indique la manera en cómo finalizó este estado** en particular ya que servirá para comunicárselo al usuario.

Administración de datos

1. Necesitamos llevar a un estado conocido a cada estado antes de que este se ejecute. Aquí no necesitamos asignar memoria por lo cual no necesitamos de muchas funciones.

Manejo de eventos

1. Manejar el **movimiento de la palanca**. El objeto experimento determina el tipo de movimiento que se realizó, en caso de que éste provenga de la palanca, el estado determina primero si el tipo de control es pasivo, después verifica que el movimiento esté permitido, a continuación prueba que el movimiento se haya dado dentro de la ventana de tiempo y si fue adecuado. Finalmente, modifica la variable de fin de estado.
2. Manejar el **movimiento de los botones**. Muy parecido al anterior, sólo que aquí se revisa si el botón presionado fue el adecuado.
3. **Tiempo de estado cumplido**. Cuando la duración de un estado ha terminado, se produce un mensaje que a final de cuentas será tratado en esta parte. Lo único que hay que hacer es revisar cómo terminó este estado, si se esperaba un movimiento se revisa que éste se haya dado.

Funciones a ser especializadas

1. Una función que regrese el tipo de estado actual.
2. Funciones que calculen los datos de cada tipo de estado.
3. Funciones de ejecución que generen y terminen cada tipo de estado.
4. Funciones que reporten el final de cada estado.

Esta última parte de funciones especializadas es para indicar que serán desarrolladas en las clases que se deriven de CEstado, en términos de programación orientada a objetos, representan funciones virtuales que serán llamadas polimórficamente dependiendo del tipo de estado al cual se haga referencia.

En cuanto a la dinámica de un estado, tenemos lo siguiente:

1. Se **genera el estado** correspondiente. Es decir se llaman las funciones encargadas de manejar los dispositivos para que se generen las señales.
2. Si se presentan **eventos**, manejarlos adecuadamente. Aquí, el estado está consciente de lo que debe ocurrir y sabe cómo manejarlo.
3. Cuando el **estado finaliza**, el ensayo padre debe ser informado sobre cómo se generó este estado y si terminó adecuadamente o no.

V.2.3.4. Generalización de la clase CEstado

Una vez establecidas las operaciones básicas de cada estado, procedimos a derivar otras clases que se encargarían de cada tipo de estado. Dichas clases derivadas, heredan las características de la clase base pero incorporan nuevas características. Aquí es en donde insertamos el código desarrollado en las secciones V.1.1, V.1.8 y V.1.8.3. Con ello logramos que cada estado supiera cómo generar sus señales, cómo terminar su generación, informar al ensayo padre de su estado final y de calcular sus propios datos. Por ello, sólo mencionamos las clases que se generalizaron a partir de CEstado.

CEstadoEstatico
Esta clase implementaría los estados en donde la punta permanece estática, como los estados E0, E2, E4, E6 y E9 de la Figura V-1. Esta clase simplemente incorpora un dato que representa la posición en la cual será colocada la punta. Para la generación del estado hace uso del material desarrollado en la sección V.1.8.1.

CEstimulo
Los estados E3 y E5 de la Figura V-1 son de tipo estímulo, de tal manera que esta clase, la más compleja y especializada de todas, se encarga de incorporar todo lo visto en la secciones V.1.8.3 y V.1.8.4.

CRampa
Los estados de tipo rampa son generados con esta clase, la cual agrega toma las ideas desarrolladas en la sección V.1.8.2. Así, se pueden generar estados como E1 y E7 en la Figura V-1.

CRecompensa
El estado E8 de la Figura V-1 representa un estado de tipo recompensa. Decidimos agregar este tipo de estado porque se debía tomar una decisión en base a las respuestas del mono, esto es, cuando el mono presionara el botón adecuado, se le daría recompensa y se debía pasar al estado siguiente; en caso de que el mono presionara el botón incorrecto, no se le proporcionaría recompensa, se castigaría con tiempo adicional y se continuaría con el estado siguiente. Sin embargo, si el mono no presionara botón alguno, entonces el ensayo debía comenzar nuevamente puesto que el mono no habría llevado a cabo su movimiento. Para solucionar este problema necesitábamos tomar una decisión en base a la acción llevada a cabo por el mono. Con este tipo de estado lo que hicimos fue que si el movimiento había sido el adecuado por parte del mono, entonces se suministraría recompensa al mono, el estado siguiente sería omitido y se continuaría con los estados siguientes. En caso de que el mono hubiera llevado a cabo el movimiento pero su respuesta fuera equivocada, no se le daría recompensa, se ejecutaría el estado siguiente y se continuaría con los demás estados. Sería como si el estado que está a continuación de un estado *recompensa*, fuera un estado de castigo el cual se ejecutaría si el mono no acertó en su respuesta. Por otro lado, si el mono no llevó a cabo el movimiento, entonces se aborta el ensayo actual y se repite nuevamente. Este estado incorpora una variable que indica dónde colocar la punta y otra variable para indicar si hay castigo o no. En cuanto a su señal, este estado es idéntico al de tipo estático.

V.2.4. Definición de los datos del experimento a partir de un archivo de Excel

Con este requisito, identificamos que nuestra aplicación también debía leer e interpretar la información contenida en un archivo de otra aplicación, Excel en particular, dicha información definiría por completo los datos del experimento en su totalidad. Sin embargo, al provenir de otra aplicación, la fuente de datos debía ser vista más como una base de datos que como un archivo local a nuestra aplicación; además, el enfoque cambiaría ya que en lugar de realizar una apertura de archivo, haríamos una transacción.

Visual C++ provee diferentes interfaces para trabajar con bases de datos como son ODBC, ODBC con MFC, OLE DB y ADO. ODBC (Conectividad Abierta a Bases de Datos) permite conectarse a diferentes orígenes de datos como son SQL Server, Acces, FoxPro, Excel, dBASE, Paradox e incluso archivos de texto; además, dichas conexiones no sólo pueden ser locales sino remotas y pueden ser independientes de la plataforma. Para conectarse a dichas fuentes de datos, ODBC utiliza SQL (Lenguaje de Consulta Estructurado); por otro lado, MFC incluye clases que proveen una interfaz más simple hacia bases de datos. Por lo anterior, decidimos usar las clases MFC que implementan la interfaz de ODBC, de esta manera, todos los detalles que tienen que ver con la conexión a la base de datos quedan ocultos (encapsulados) en dichas clases. Imaginemos lo anterior como una comunicación de diversas capas entre la base de datos y nuestra aplicación: una capa MFC sobre una capa de ODBC que está sobre SQL. Este complejo sistema de conexión es soportado principalmente por el controlador específico que implementa la parte sustancial de las funciones de la librería API de ODBC y es el que realiza la verdadera conexión con la base de datos. Visual C++ provee el controlador para archivos de Excel con lo cual estábamos casi completos para solventar los problemas siguientes:

V.2.4.1. Problemas a resolver utilizando las clases de MFC para ODBC

Hasta entonces, Microsoft no había mencionado porqué el archivo de Excel tenía que ser formateado de manera especial para que pudiera ser leído. Esto era importante porque de lo contrario la conexión no vería la fuente de datos, aún cuando el archivo de Excel tuviera datos.

Otro problema fue el del nombre de la fuente de datos (DSN), que, en términos de bases de datos, representa los datos mismos, la información requerida para accederlos y el lugar donde están ubicados. Visual C++ permite crear aplicaciones de bases de datos sólo desde el inicio del desarrollo de la aplicación, pero el problema es que el administrador de fuentes (orígenes) de datos debe haber registrado (instalado) previamente el nombre de la fuente de datos para que el asistente de Visual C++ pueda crear la aplicación. Pero eso no es todo, la aplicación creada únicamente se puede conectar a archivos cuyo DSN sea el mismo, lo cual significa que la aplicación tendría que ampliarse (agregarle código extra) por cada tipo de conexión (archivo de experimento) que se hiciera, con lo cual la estabilidad de la aplicación sería pobre. La documentación de Visual C++ no menciona algo al respecto.

El otro problema es que dada la forma en la cual está diseñado el controlador y las clases de MFC para ODBC, no se puede hacer la actualización del archivo, es decir, desde nuestra aplicación no podemos cambiar el contenido del archivo de Excel.

Una aplicación ODBC realiza los siguientes pasos para trabajar con la base de datos: reserva el entorno ODBC, reserva un controlador para la conexión, se conecta al origen de los datos, ejecuta instrucciones SQL, recupera los resultados de la consulta, se desconecta del origen de datos y libera el entorno ODBC. Haciendo uso de las clases de MFC, muchos de estos procesos quedan encapsulados y son trasparentes al programador. Otra ventaja es el control de los errores en las consultas ya que estas clases lanzan excepciones cada vez que ocurre un error que no amerita que la aplicación termine.

V.2.4.2. Solución de los problemas

De acuerdo a Microsoft, una hoja de Excel (versión 4.x y posteriores) únicamente puede ser leída por ODBC si el rango de la base de datos es definido. Desgraciadamente Microsoft no dice cómo hacer esto de manera precisa. Una forma de hacer que los datos sean vistos por ODBC es nombrar un rango de datos en la hoja. Como puede haber más de una tabla en una hoja de trabajo, esto quiere decir que una hoja no es necesariamente lo mismo que una tabla en una base de datos real. Debido a lo anterior el usuario tendrá que llenar sus datos en la hoja de Excel y por último insertar un nombre que defina el rango de datos.

Como el asistente de Visual C++ crea una aplicación de bases de datos cuyo DSN no puede ser cambiado, tenemos que trabajar directamente con las funciones de las clases y establecer ahí nuestros propios parámetros. Una de estas clases es CDatabase, que es la que establece la conexión al origen de datos y lo hace con un DSN que le es pasado en una cadena de caracteres. Nuestra tarea entonces fue formar un DSN personalizado; para ello, sólo recuperamos la ruta del archivo que contiene los datos del experimento y formateamos la cadena que representa al DSN con la ruta del archivo. Pasando este DSN a la función de la clase CDatabase que abre la base de datos podemos especificar cualquier origen de datos, es decir, el usuario sólo tiene que especificar el lugar donde se encuentra el archivo de experimento.

Desgraciadamente, las clases que encapsulan ODBC no son capaces de crear, modificar o actualizar la base de datos representada por un archivo de Excel, por lo que nuestro programa sólo leería la información contenida en dichos archivos. Esto trae como consecuencia que el usuario tendría que modificar directamente el archivo de Excel y después recuperar los datos desde el programa del sistema.

VI. Integración final del sistema y evaluación

En este punto contábamos con las herramientas necesarias para integrar el sistema. Por un lado, teníamos los dispositivos físicos encargados de la generación y control del experimento, lo cual se puede ver en la Figura VI-1. Se pueden identificar los sistemas principales que son el del manejo de eventos y el de la generación de estímulos, los cuales hacen uso de las tarjetas electrónicas, del sintetizador de señales y del estimulador vibro-táctil. También se puede ver la interacción entre el mono y el sistema: el mono lleva a cabo movimientos con su mano libre en base a los estímulos que recibe en uno de los dedos de su mano fija; además, también puede recibir una recompensa y otros tipos de estímulos. De esta forma, teníamos los bloques funcionales que servían para trabajar las tareas principales de detección y discriminación.

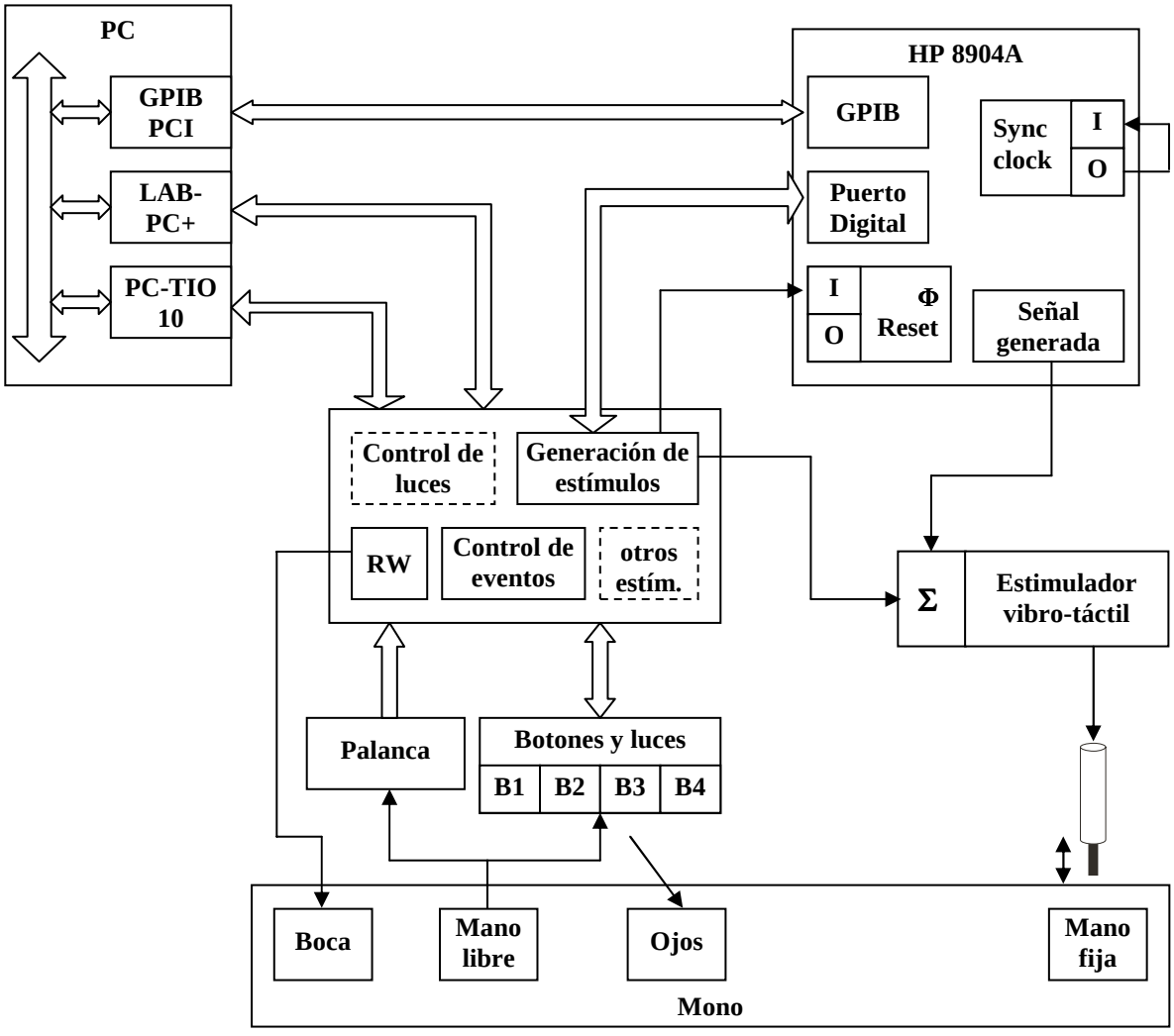


Figura VI-1. Integración de los diferentes dispositivos al sistema

Los cuadros punteados correspondientes a *Control de luces* y *otros estímulos*, representan módulos que no fueron integrados al sistema pero que se espera sean agregados

posteriormente. Es preciso decir que también diseñamos el subsistema que controla la intensidad de las luces integradas en los botones, el cual es un circuito digital que varía el ángulo de disparo de un triac para modificar la señal de CA que alimenta la lámpara de neón de los botones. El punto fue que por cuestiones de tiempo, no se integró esta parte al sistema. También se tiene pensado incluir otros tipos de estímulos, como son auditivos y visuales; además, el módulo donde se encuentran los botones llevará a cabo movimientos, de tal manera que el mono deberá presionar el botón adecuado cuando éste se esté moviendo.

En cuanto a la interfaz de usuario, contábamos con las clases básicas para la generación del experimento y restaba establecer los comandos necesarios para la ejecución del experimento, el cual se definió anteriormente en la clase CExperimento y cuya estructura básica sería como la que se muestra en la Figura VI-2.

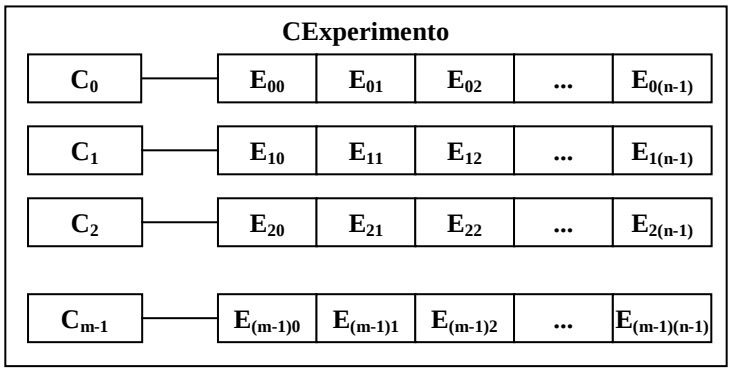


Figura VI-2. Experimento formado por ensayos y estados

La figura anterior representa una instancia de la clase CExperimento la cual contendría *m* clases a ejecutar y cada clase contiene *n* estados. Recuérdese también que la clase CExperimento es la encargada de administrar la ejecución del experimento, tenemos entonces que ésta clase provee la interfaz necesaria para integrar un objeto de este tipo en la aplicación que representa la interfaz de usuario.

Habíamos comentado anteriormente que una aplicación para Windows, desarrollada con Visual C++ y MFC, provee un esqueleto básico que consta de cuatro objetos básicos: la aplicación en sí, la ventana principal, el documento y la vista. Los dos últimos objetos establecen una arquitectura Documento-Vista en la que la que el documento se encarga de los datos y la vista se hace cargo de la visualización de ésos datos. Apegándonos a este tipo de arquitectura, el lugar más lógico para integrar un objeto de tipo experimento fue el objeto documento. La vista jugaría un papel muy importante ya que sería el objeto al cual llegarían los mensajes provocados por eventos externos (como los movimientos del mono) y por eventos propios de la aplicación (como la ejecución de comandos o la visualización de los datos). La Figura VI-3 muestra el esquema básico de la aplicación que diseñamos y la forma en la cual se comunican los objetos: el objeto App crea y administra a los demás objetos, representa el proceso principal de la aplicación; la instancia de MainFrame es la ventana principal de la aplicación y aloja a la vista, también provee la interfaz gráfica de usuario. El objeto Documento incorpora una instancia de la clase CExperimento.

En cuanto a la vista, ésta presenta la información resultante del experimento como una lista de resultados; de hecho, la vista está basada en una clase de MFC de tipo lista muy

parecida a la manera en que se presenta la información de una carpeta en el explorador de Windows cuando se presentan los detalles de cada archivo o carpeta. Elegimos este tipo de vista basados en la vista del anterior sistema, en el cual se tiene una vista semejante.

El usuario interactúa con el sistema a través de esta interfaz seleccionando los comandos disponibles para llevar a cabo el experimento. Dichos comandos de usuario hacen llamadas a las funciones miembro de cada clase, con ello se establece la interacción y colaboración de cada objeto para realizar las tareas adecuadas.

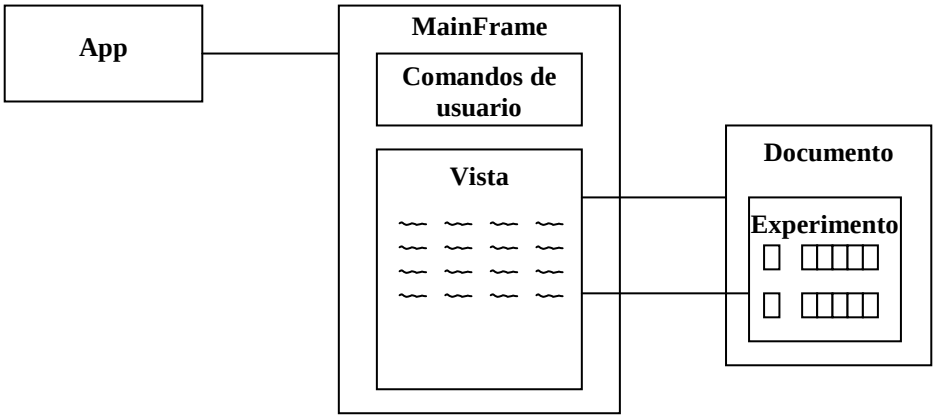


Figura VI-3. Interfaz de usuario

Es importante mencionar que cada clase se trabajó durante la realización del sistema para obtener el mejor rendimiento posible; así, por ejemplo, la clase que representa la aplicación (el objeto App) llama una función que cambia la prioridad del hilo que representa este proceso para que tenga una prioridad arriba de la normal, con ello logramos un pequeño incremento en cuanto al tiempo de respuesta a los eventos externos. El objeto MainFrame por su parte, fue modificado para que incorporara paneles en su barra de estado, dichos paneles muestran información importante como el número de corrida actual, la manera en la cual se desarrolla o termina el ensayo actual y el porcentaje de aciertos del mono. La clase Documento se encarga de establecer la conexión al archivo de Excel que contiene la información del experimento y de realizar las conversiones necesarias para preparar los datos del experimento; en este punto se hizo necesario proveer un control de errores en caso de que el archivo de Excel no estuviera formateado adecuadamente, para ello utilizamos las excepciones que son lanzadas por las clases de ODBC. También hicimos un control de errores basado en excepciones sobre la información contenida en el archivo, es decir, el usuario escribe los datos en el archivo de Excel de una manera que es representativa para él, pero que nada tiene que ver con los tipos de datos utilizados por los dispositivos; además, el usuario también puede escribir erróneamente un dato o no proveer la información suficiente para que el programa pueda determinar si se trata de un dato que representa un tiempo o una posición. Cada vez que existiera un error del tipo anterior, el sistema estaría preparado para indicar al usuario la causa del error y el lugar donde éste se encuentra.

Por otro lado, la clase vista está basada en la clase CListView de MFC, en este tipo de vista se puede mostrar información ordenada en columnas. Las funciones manejadoras de comandos de usuario son definidas en esta clase, lo cual quiere decir que los mensajes

de comandos son dirigidos a esta clase. Ahora bien, los comandos de usuario no pueden estar activos todo el tiempo ya que, por ejemplo, un experimento no puede empezar si no se ha elegido un archivo de Excel, por ello algunas funciones de actualización de comandos son definidas también dentro de la Vista. Cuando se ha establecido una conexión a un archivo de Excel, la clase Documento envía un mensaje a la clase Vista para que muestre al usuario un cuadro de diálogo donde se elegirán las clases a ejecutar y las opciones de ejecución; otro mensaje es enviado a la Vista cuando se ha obtenido la información necesaria para determinar qué tipo de información presentará la ventana, para este efecto, sólo se muestra la información correspondiente a los tiempos de respuesta, los parámetros de los estímulos aplicados y el tipo de respuesta del mono. Otra parte importantísima de la Vista es que recibe los mensajes provenientes de los dispositivos de manejo de eventos externos, de la generación de estímulos y de la generación de la rampa por parte de la tarjeta analógica. Pero eso no es todo, también recibe mensajes enviados por los objetos Ensayo y Experimento. Así, cada vez que un ensayo inicia, reinicia, es repetido, abortado o finaliza, la vista recibe el mensaje correspondiente para actualizar los datos que muestra en la ventana y delegar las responsabilidades correspondientes a dichos objetos dependiendo del mensaje recibido. En lo que al objeto Experimento se refiere, éste envía mensajes a la Vista cada vez que termina un ensayo y es necesario ejecutar el siguiente en la lista de ensayos; el objeto experimento también envía mensajes a la Vista cada vez que inicia o termina una corrida y cuando finaliza el experimento, dichos mensajes también actualizan la información mostrada en la ventana y en los paneles de la barra de estado de la ventana principal.

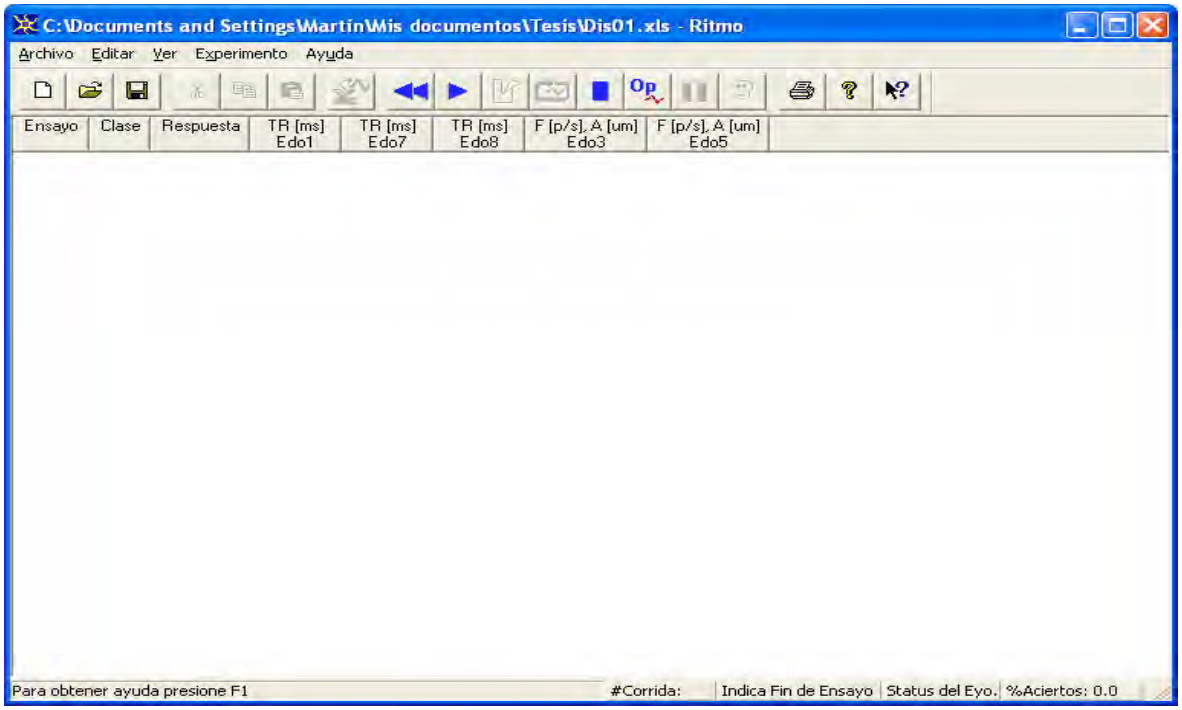


Figura VI-4. Aspecto de la interfaz de usuario

La Figura VI-4 muestra cómo se vería la interfaz de usuario cuando se realiza un experimento en el que el sujeto lleva a cabo una tarea de discriminación como la que se ha venido tratando a lo largo de este documento. Se pueden ver los comandos disponibles al usuario en la barra de herramientas y en los menús de programa, así como también la información presentada en la barra de estado, la cual corresponde al número de corrida, información acerca del desarrollo del ensayo y el porcentaje de aciertos del sujeto.

Por otro lado, podemos ver que la vista presenta, de manera general, la información correspondiente al ensayo actual, a la clase a la cual corresponde dicho ensayo y la respuesta del sujeto, mostradas éstas en las primeras tres columnas. De la Figura V-1 y de acuerdo a la tarea mencionada, podemos observar que los estados donde se presentan movimientos por parte del mono es en los estados 1, 7 y 8, entonces la información mostrada debajo de las siguientes tres columnas corresponde al tiempo de respuesta, en ms, del mono. Por último, las dos últimas columnas corresponden a los parámetros de los estímulos aplicados (estados 3 y 5), y se refieren a la frecuencia en pulsos por segundo y a la amplitud de los estímulos, en μm. No obstante, es importante mencionar que sólo aquellos estados donde se presente movimiento o sean estímulos tendrán una columna con la información correspondiente.

VI.1. Evaluación del sistema

Como habíamos mencionado anteriormente esta es una de las tareas del método en espiral donde se realizan pruebas para la valoración del sistema.

Durante la programación evaluamos rutinas. La prueba de los programas es una técnica de confirmación del sistema esta la debemos hacer antes de entregar el sistema al usuario. Las pruebas consisten en ejercitar el sistema utilizando datos similares a los reales y observar los resultados e interpretar estos para detectar errores o insuficiencias del sistema.

Es importante comprender que las pruebas nunca demuestran que un programa esta correcto; es posible que existan errores aun después de la prueba más completa.

Una vez terminado el sistema se hicieron varias pruebas para revisar que cumplieran con los requerimientos establecidos. Para ello establecimos diferentes tareas esto lo realizamos directamente sobre la maquina que conforma el sistema esto se documenta en los resultados del sistema.

VII. Resultados y Conclusiones

VII.1. Resultados

Para obtener los resultados correspondientes del sistema se realizó un número considerable de tareas (detección, categorización, discriminación, etc.); sin embargo, hay que señalar que las pruebas se realizaron con algunas restricciones.

Como se menciono anteriormente en esta tesis, nuestro trabajo consistió en diseñar un sistema que permitiera al investigador asignar diferentes tareas a un sujeto bajo prueba y proveer sincronización con el sistema colector de datos, es decir, nuestro sistema también debe generar las señales requeridas para el sistema colector de datos, las cuales están bien identificadas y definidas en los sistemas anteriores, por lo cual no es necesario interactuar directamente con el sistema colector de datos para hacer pruebas y evaluar el sistema.

Los resultados de las señales de sincronización fueron evaluados con la ayuda de un osciloscopio y obtuvimos los resultados esperados.

En cuanto a la interacción del sistema con el usuario y el sujeto, y a fin de ejemplificar los resultados, presentamos una tarea diseñada por el Dr. Romo y los resultados que se obtuvieron.

Estas pruebas fueron creadas con el nuevo sistema pero no se pudieron realizar con el sujeto (mono) ya que se necesitaba de una silla especial que se coloca en un lugar especifico en el laboratorio donde se llevan acabo los experimentos; además del tiempo prolongado y considerable para sustituir e implementar físicamente el nuevo sistema ya que esto implicaría la suspensión de experimentos, lo cual no es deseable ya que dichos experimentos arrojan resultados importantes para la investigación en periodos largos de tiempo. Sin embargo, las pruebas pueden ser realizadas por alguna persona con conocimiento o sin conocimiento de la tarea y obtener resultados confiables haciendo movimientos acertados o equivocados que haría el sujeto, teniendo también la posibilidad de efectuar movimientos complejos en el sistema que provocarían eventos externos que pudieron no ser contemplados en el diseño del sistema y poder corregir y rediseñar.

Siguiendo con la tarea de discriminación descrita en capítulos anteriores, y con el fin de ilustrar los resultados, a continuación mostramos la manera en la cual serían definidos los datos de dicha tarea a partir de un archivo de Excel (véase la Tabla VII-1).

Con este formato se puede definir un experimento compuesto de ***N*** clases conteniendo, cada una, ***K*** estados. Para nuestro ejemplo, tenemos dos clases ***E0*** y ***E1*** conteniendo cada una 10 estados. Este archivo debe tener dos tablas insertadas (o nombres) principales que son ***ensayo*** y ***experimento***, dichas tablas únicamente deben abarcar el número de celdas a ser consideradas en el experimento.

La tabla ***ensayo*** determina el número y tipos de estado de cada clase, para ello, se debe escribir “estado” debajo de la celda *ensayo* y “tipo” debajo de la celda *propiedades*. A continuación se deben agregar tantas columnas como estados se deseen tener.

La tabla ***experimento***, define las clases a presentar en el experimento también contiene las celdas *propiedades* y las de los *estados*, sólo que aquí se dan los parámetros de cada clase de manera independiente. Debajo de la celda experimento se listan las clases a presentar en el experimento.

La secuencia de las clases ***E0*** y ***E1*** es la mostrada en la Tabla VII-1. Como se puede observar, los estados "*estado3*" y "e*stado5*" de la tabla ***ensayo*** son de tipo estímulo, cuyos parámetros dependen de la tarea asignada por el investigador.

ensayo	propiedades	estado0	estado1	estado2	estado3	estado4	estado5	estado6	estado7	estado8	estado9	estado10
estado	tipo	retraso	rampa	retraso	estimulo	retraso	estimulo	retraso	rampa	recompensa	retraso	retraso
experimento	propiedades	estado0	estado1	estado2	estado3	estado4	estado5	estado6	estado7	estado8	estado9	estado10
e0	parametros	1s,0um	1s,0um,1.5mm,100ms	1s,1.5mm	p1,100,1s,100,20um	3s,1.5mm	p1,30,1s,30,50um	1s,1.5mm	1s,1.5mm,0um,100ms	3s,0um	2s,100um	1s,200um
e0	movimiento		ocupacion						liberacion	boton1		
e1	parametros	1s,0um	1s,0um,1.5mm,100ms	1s,1.5mm	p1,30,1s,30,50um	3s,1.5mm	p1,100,1s,100,20um	1s,1.5mm	1s,1.5mm,0um,100ms	3s,0um	2s,100um	1s,200um
e1	movimiento		ocupacion						liberacion	boton2		

Tabla VII-1. Formato del archivo de Excel que describe una tarea de discriminación

Se provoca que la punta del motor lineal haga contacto con la yema de uno de los dedos de la mano inmovilizada del sujeto (aplicando una rampa negativa en función del tiempo al control del motor) esto sucede en el "e*stado1*" de tipo rampa, durante este estado el sujeto tiene que sujetar la palanca al detectar la punta del motor en su dedo. En el archivo de Excel antes mostrado se puede ver claramente en el "*estado1*" de tipo rampa y el campo "*propiedades*" de la tabla experimento el registro "*movimiento*" asignando "*ocupación*" para el "*estado1*" por lo que el sistema debe de esperar un movimiento de ocupación en la palanca en el tiempo determinado durante el "*estado1*", si esto ocurre el sistema continua al siguiente estado del ensayo, en caso contrario el sistema aborta el ensayo y continua en el "*estado0*", es decir reinicia el ensayo.

Tras una serie de errores intencionales cometidos hasta Estado1, nuestro sistema funcionó adecuadamente cuando el sujeto (en este caso una persona), no colocaba la mano en la palanca en el tiempo establecido por los parámetros del estado rampa. Lo anterior quiere decir que el sistema reiniciaba el ensayo actual si el sujeto tomaba la palanca antes o después de lo permitido

Como explicamos anteriormente, si un estado es exitoso, es decir, cumple con los requisitos impuestos por sus parámetros y propiedades, el sistema continúa con el siguiente estado. Siguiendo con nuestra tarea, una vez terminado satisfactoriamente el "*estado1*", se daba paso al "*estado2*", de tipo retraso, en la tabla ensayo. En este estado el sistema no permite movimiento alguno en la palanca durante el tiempo establecido por el usuario, esto se puede observar en el archivo de Excel en la propiedad *movimiento* que está en blanco, es decir, no permite ocupación o liberación de la palanca. Los resultados de los parámetros y de la propiedad permitida "*cuantización*" fueron satisfactorios tras una serie de aciertos y equivocaciones intencionales.

El estado "*estado3*", de tipo estímulo, al igual que en los demás estados fue realizado, siempre y cuando el estado anterior también hubiera sido satisfactorio. En este estado se hicieron pruebas con los diferentes estímulos que el sistema es capaz de generar, en algunas de las pruebas que hicimos se establecieron parámetros no permitidos en el estado de tipo estímulo.

El sistema respondió satisfactoriamente enviando mensajes de error al usuario cuando se definían parámetros no permitidos. En cuanto a la ejecución de los estímulos, los resultados fueron los esperados.

De manera predeterminada, ninguno de los estados permite el movimiento en la palanca ni en los botones; no se proporciona recompensa y la duración del estado no está cuantizada. De esta forma, el usuario es responsable del diseño del experimento por lo que debe tener cuidado de seguir un orden lógico en cuanto a los estados y sus propiedades.

Con el fin de tener dos estímulos en la tarea para que el sujeto pueda compararlas y discriminar es necesario tener un tiempo de espera antes de generar el segundo estímulo esto se logra con un estado de tipo retraso y con los parámetros y propiedades permitidas que ya fueron explicadas, para nuestro ejemplo, como se puede ver en la Tabla VII-1 es el *estado4* de tipo retraso de 3s de duración y con la propiedad cuantización asignada. Los resultados por nuestro sistema fueron satisfactorios, recalcando que todos los estados anteriores fueron exitosos para poder llegar al estado en cuestión.

Una vez completado el tiempo de espera, se ejecuta el siguiente estado de tipo estimulo y con sus parámetros correspondientes, para después pasar a un tiempo de espera generado con un estado de tipo retraso antes de ejecutar el estado de tipo rampa "*estado7*".

El estado de tipo rampa es capaz de generar rampas con pendiente positiva o negativa. Dando seguimiento a nuestro ejemplo se pueden observar los parámetros del *estado7*: “2s,1.5mm,0um,100ms”. El segundo y tercer parámetro corresponden al punto inicial y punto final de la punta del motor. En este estado tenemos también la propiedad “*movimiento*” definida como “liberacion,0.2s,2.0s” esto significa que al retirar la punta del motor del dedo del sujeto donde fueron aplicados los estímulos, el sujeto debe quitar la mano de la palanca (liberarla) en un tiempo (ventana de tiempo) tal y como lo indican el segundo y tercer parámetro de la propiedad *movimiento*, esto es, entre 0.2s después de comenzar el *estado7* y 2.0s complementarios a 0.2s. Se realizaron pruebas con diferentes parámetros obteniendo resultados satisfactorios por el sistema.

La recompensa se ejecuta con el estado de tipo “*recompensa*”, el cual cuenta con las propiedades: *parámetros*, *movimiento* y *recompensa*. A continuación explicamos la propiedad recompensa, pero antes es necesario hacer énfasis en la tarea discriminación y lo que el sujeto debe realizar hasta ese momento. Como se mencionó antes, se aplican dos estímulos en el dedo de la mano sujeta del mono y en el momento que es retirada la punta del motor del dedo, la mano libre debe soltar la palanca en un tiempo establecido, si todo esto ocurre, entonces en el estado “*estado8*” el sujeto tiene un tiempo establecido por el usuario y definido en la propiedad *parametros* del *estado8* (2s para nuestro ejemplo) para comparar en igualdad o diferencia el segundo estimulo con el primero, para lo cual existen dos botones que están colocados frente al sujeto y que son asignados por el usuario en la propiedad *movimiento* del estado recompensa. El usuario hace una elección del botón que significara igual y, por lo tanto, el otro significara falso previa al llenado de las propiedad movimiento en base a los parámetros de los estímulos *estado3* y *estado5*. En nuestro ejemplo el ensayo *e0* los dos estímulos son iguales y el boton1 es asignado como verdadero por lo que si el sujeto oprime el boton1 será recompensado y si oprime el boton2 será castigado es decir no recibirá recompensa y pasará al estado final.

En el caso del ensayo *e1* la comparación del estimulo dos con el primer estimulo es diferente por lo que se asigna en la propiedad *movimiento* del estado recompensa el boton2 aquí sucede lo contrario si el mono oprime el boton1 será castigado ya que los estímulos son diferentes y si oprime el boton2 será recompensado y pasará al siguiente estado final.

Los resultados del sistema en la ejecución completa de la tarea fueron los esperados.

VII.2. Conclusiones

El sistema cumple con los objetivos establecidos. La flexibilidad en el diseño de sus tareas, así como el control y ejecución para la realización de experimentos. La creación y modificación de una tarea, así como la ejecución y control de ésta para el desarrollo de experimentos son realizadas en una computadora con una interfaz gráfica de usuario funcionando bajo la plataforma Windows 98. Los datos de origen correspondientes a los parámetros de cada experimento son contenidos en un archivo de Excel. El sistema cuenta con las señales necesarias para la sincronización con el sistema colector de datos.

El usuario puede establecer en su experimento diferentes tipos de estímulos desde un archivo de Excel:

Periódico más un pulso al final, aperiódicos más un pulso al final, estímulo basado en número de pulsos, bipulso y de usuario con esto el investigador tiene la posibilidad de crear y ejecutar diferentes tipos de estímulos y tener una versatilidad y flexibilidad para el diseño de las tareas existentes (detección, comparación y discriminación) así como la posibilidad de definir nuevas tareas por cualquiera de los investigadores del laboratorio sin requerir de ellos un conocimiento profundo del sistema lo cual acelera la obtención de resultados.

Logramos que el sistema permitiera configurar las opciones de ejecución del experimento, como decidir si la ejecución de ensayos es secuencial o aleatoria, si el mono juega un papel pasivo, controlar la ejecución del experimento, es decir, el investigador tiene la posibilidad mediante comandos en la interfaz grafica abortar y reiniciar el ensayo actual si el sujeto no está respondiendo como se esperaba o simple mente cuando lo decidiera, pausar , repetir, además de recabar información importante concerniente al comportamiento del sujeto(respuestas motoras) y mostrarla al investigador en tiempo real.

La ventaja de hacer la aplicación bajo la plataforma Windows es la flexibilidad para la interfaz grafica sin embargo el tiempo de atención a los eventos externos no es tan rápida como el sistema operativo MS-DOS esto debido a que Windows maneja eventos y no interrupciones. Windows coloca los eventos en una cola de mensajes y son despachados conforme fueron llegando pero tienen la posibilidad de manejar varios eventos simultáneamente por ser este sistema operativo multitarea. A este respecto, cabe mencionar que el tiempo mínimo logrado fue de 3ms para asegurar el control del sistema sobre la señal que genera los estímulos.

Al construir este Sistema nos dimos cuenta de la importancia de conocer y seguir una metodología de desarrollo de sistemas, ya que nos permitió establecer claramente los requerimientos y llevar un orden en cuanto al desarrollo de éste, evitando así ocupar tiempo importante en cosas no necesarias.

Conocimos instrumentación y equipos de uso común en la industria como la tarjeta GPIB, LAB-PC, etc.

VIII. Apéndice

HOJA DE ESPECIFICACIONES CIRCUITOS INTEGRADOS

National Semiconductor

April 1992

DM54LS273/DM74LS273
8-Bit Register with Clear

General Description

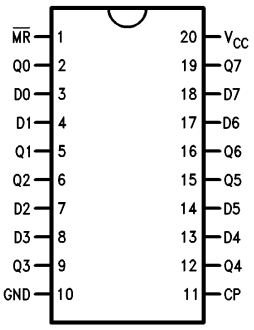
The 'LS273 is a high speed 8-bit register, consisting of eight D-type flip-flops with a common Clock and an asynchronous active LOW Master Reset. This device is supplied in a 20-pin package featuring 0.3 inch row spacing.

Features

- Edge-triggered
- 8-bit high speed register
- Parallel in and out
- Common clock and master reset

Connection Diagram

Dual-In-Line Package



TL/F/9825–1

Order Number DM54LS273E, DM54LS273J, DM54LS273W, DM74LS273M or DM74LS273N
See NS Package Number E20A, J20A, M20B, N20A or W20A

Pin Names	Description
CP	Clock Pulse Input (Active Rising Edge)
D0–D7	Data Inputs
$\overline{MR}$	Asynchronous Master Reset Input (Active LOW)
Q0–Q7	Flip-Flop Outputs

RRD-B30M115/Printed in U. S. A.

Absolute Maximum Ratings (Note)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	
DM54LS	−55°C to +125°C
DM74LS	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: *The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.*

Recommended Operating Conditions

Symbol	Parameter	DM54LS273			DM74LS273			Units
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			2			V
V_{IL}	Low Level Input Voltage			0.7			0.8	V
I_{OH}	High Level Output Current			−0.4			−0.4	mA
I_{OL}	Low Level Output Current			4			8	mA
T_A	Free Air Operating Temperature	−55		125	0		70	°C
$t_s(H)$ $t_s(L)$	Setup Time HIGH or LOW D_n to CP	15 15			15 15			ns
$t_h(H)$ $t_h(L)$	Hold Time HIGH or LOW D_n to CP	5 5			5 5			ns
$t_w(H)$ $t_w(L)$	CP Pulse Width HIGH or LOW	20 20			20 20			ns
$t_w(L)$	$\overline{MR}$ Pulse Width LOW	20			20			ns
t_{rec}	Recovery Time $\overline{MR}$ to CP	15			15			ns

Electrical Characteristics

Over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions		Min	Typ (Note 1)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max	DM54	2.5			V
			DM74	2.7	3.4		
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IH} = Min	DM54			0.4	V
			DM74		0.35	0.5	
		I_{OL} = 4 mA, V_{CC} = Min	DM74		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V V_I = 10V (DM54)				0.1	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V				20	μA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.4	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 2)	DM54	−20		−100	mA
			DM74	−20		−100	
I_{CC}	Supply Current	V_{CC} = Max				27	mA

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, and the duration should not exceed one second.

Switching Characteristics $V_{CC} = +5.0V, T_A = +25°C$

Symbol	Parameter	$C_L = 15$ pF				Units
		DM54LS		DM74LS		
				$R_L = 2\ k\Omega$		
		Min	Max	Min	Max	
f_{max}	Maximum Clock Frequency	30		30		MHz
t_{PLH} t_{PHL}	Propagation Delay CP to Q_n		24 24		24 24	ns
t_{PLH}	Propagation Delay $\overline{MR}$ to Q_n		27		27	ns

Functional Description

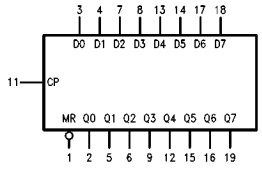
The 'LS273 is an 8-bit parallel register with a common Clock and common Master Reset. When the $\overline{MR}$ input is LOW, the Q outputs are LOW, independent of the other inputs. Information meeting the setup and hold time requirements of the D inputs is transferred to the Q outputs on the LOW-to-HIGH transition of the clock input.

Truth Table

Inputs			Outputs
MR	CP	D_n	Q_n
L	X	X	L
H	↑	H	H
H	↑	L	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

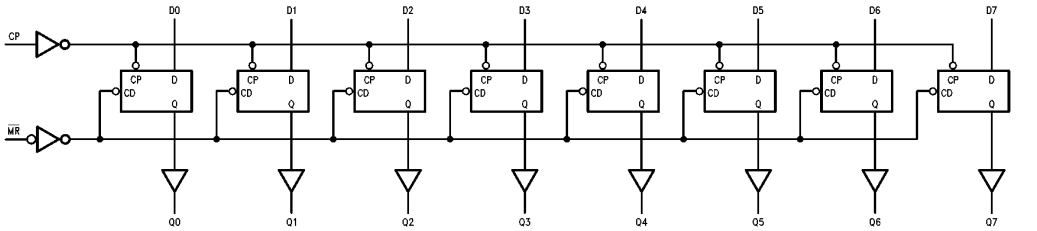
Logic Symbol



TL/F/9825–2

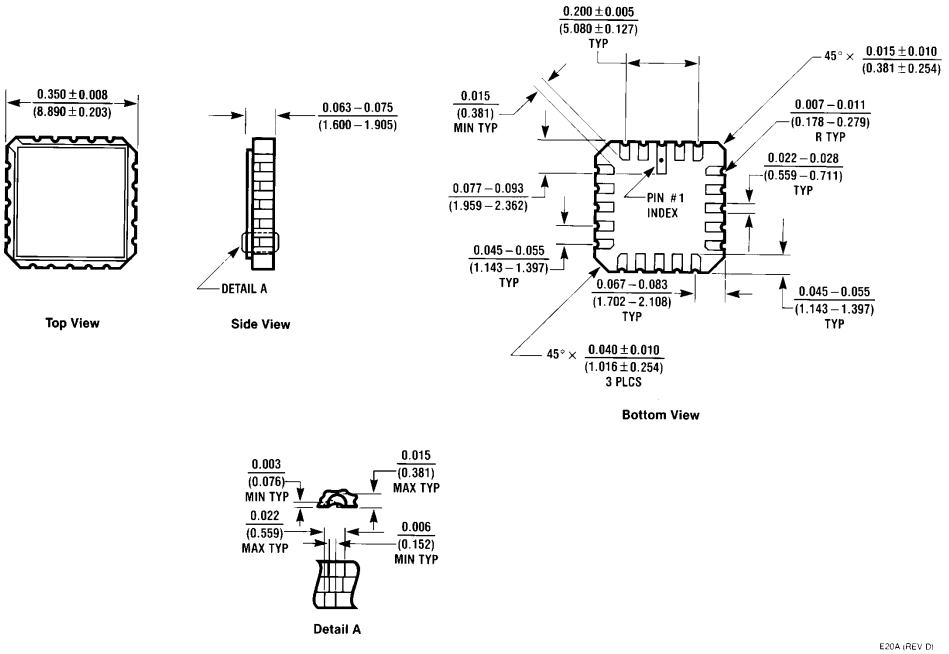
V_{CC} = Pin 20
GND = Pin 10

Logic Diagram

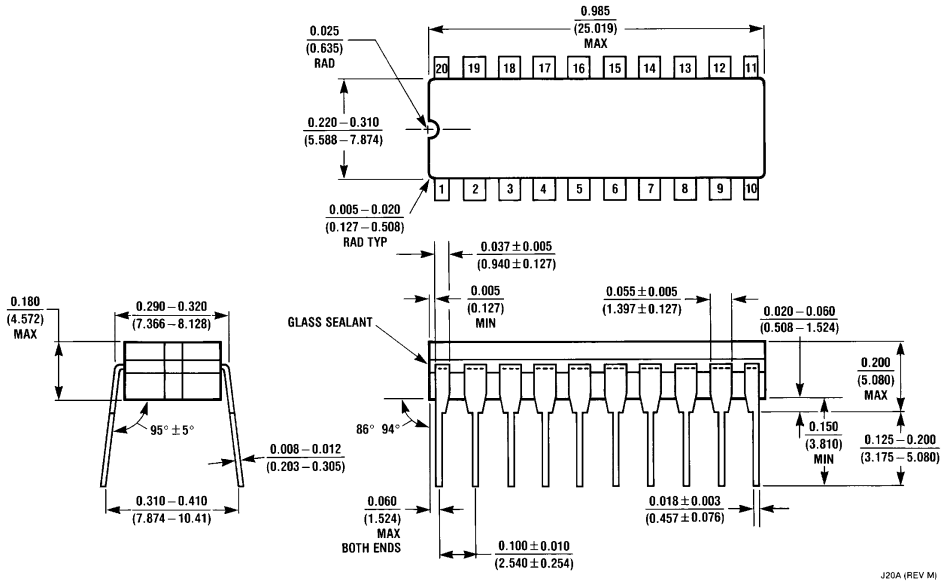


TL/F/9825–3

Physical Dimensions inches (millimeters)

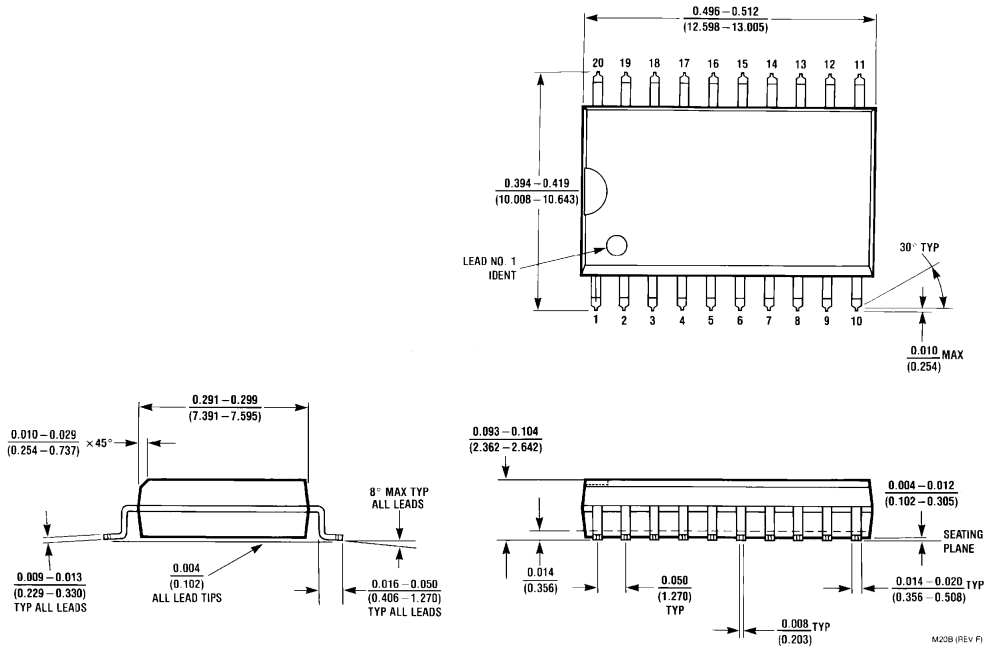


Ceramic Leadless Chip Carrier Package (E)
Order Number DM54LS273E
NS Package Number E20A

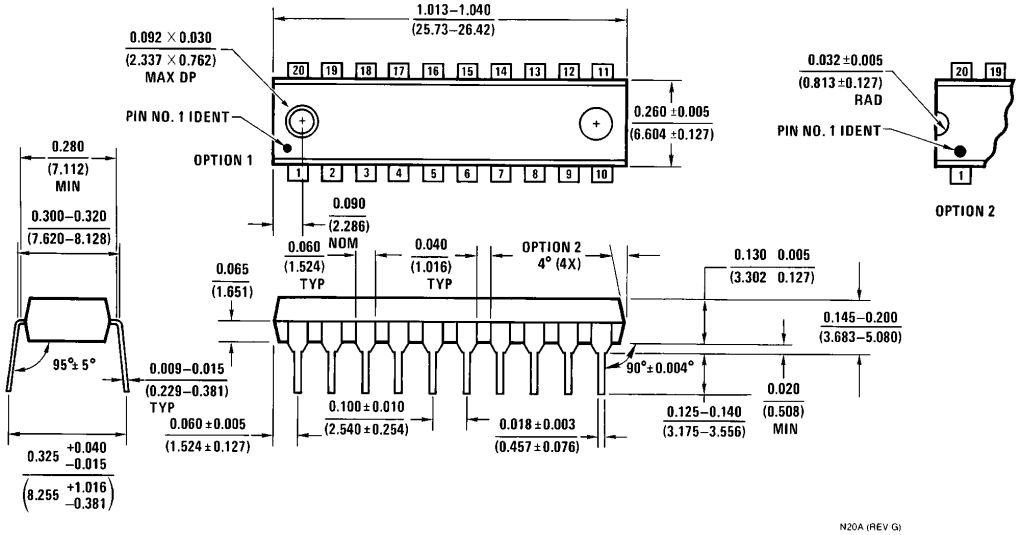


20-Lead Ceramic Dual-In-Line Package (J)
Order Number DM54LS273J
NS Package Number J20A

Physical Dimensions inches (millimeters) (Continued)

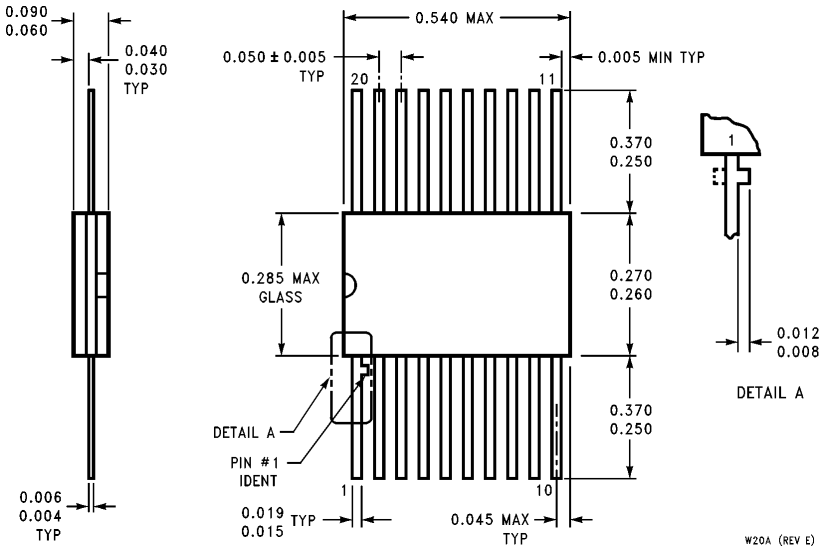


20-Lead Wide Small Outline Molded Package (M)
Order Number DM74LS273M
NS Package Number M20B



20-Lead Molded Dual-In-Line Package (N)
Order Number DM74LS273N
NS Package Number N20A

Physical Dimensions inches (millimeters) (Continued)



20-Lead Ceramic Flat Package (W)
Order Number DM54LS273W
NS Package Number W20A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

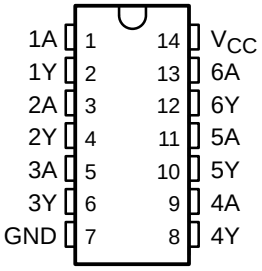
- **Operation From Very Slow Edges**
- **Improved Line-Receiving Characteristics**
- **High Noise Immunity**

description

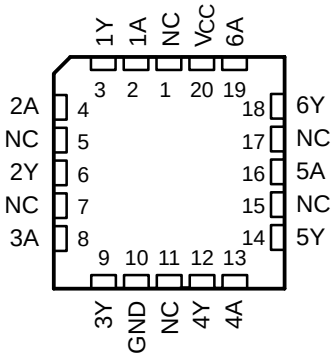
Each circuit functions as an inverter, but because of the Schmitt action, it has different input threshold levels for positive-going (V_{T+}) and negative-going (V_{T-}) signals.

These circuits are temperature compensated and can be triggered from the slowest of input ramps and still give clean, jitter-free output signals.

SN5414, SN54LS14 . . . J OR W PACKAGE
SN7414 . . . D, N, OR NS PACKAGE
SN74LS14 . . . D, DB, OR N PACKAGE
(TOP VIEW)



SN54LS14 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

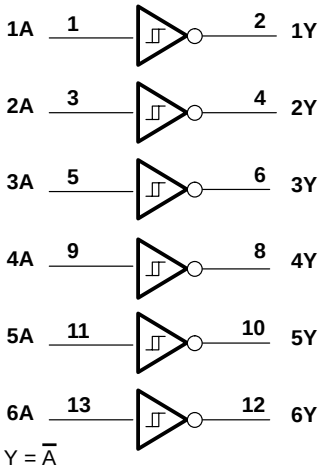
ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP – N	Tube	SN7414N	SN7414N
		Tube	SN74LS14N	SN74LS14N
	SOIC – D	Tube	SN7414D	7414
		Tape and reel	SN7414DR	
		Tube	SN74LS14D	LS14
		Tape and reel	SN74LS14DR	
	SOP – NS	Tape and reel	SN7414NSR	SN7414
	SSOP – DB	Tape and reel	SN74LS14DBR	LS14
–55°C to 125°C	CDIP – J	Tube	SN5414J	SN5414J
		Tube	SNJ5414J	SNJ5414J
		Tube	SN54LS14J	SN54LS14J
		Tube	SNJ54LS14J	SNJ54LS14J
	CFP – W	Tube	SNJ5414W	SNJ5414W
		Tube	SNJ54LS14W	SNJ54LS14W
	LCCC – FK	Tube	SNJ54LS14FK	SNJ54LS14FK

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

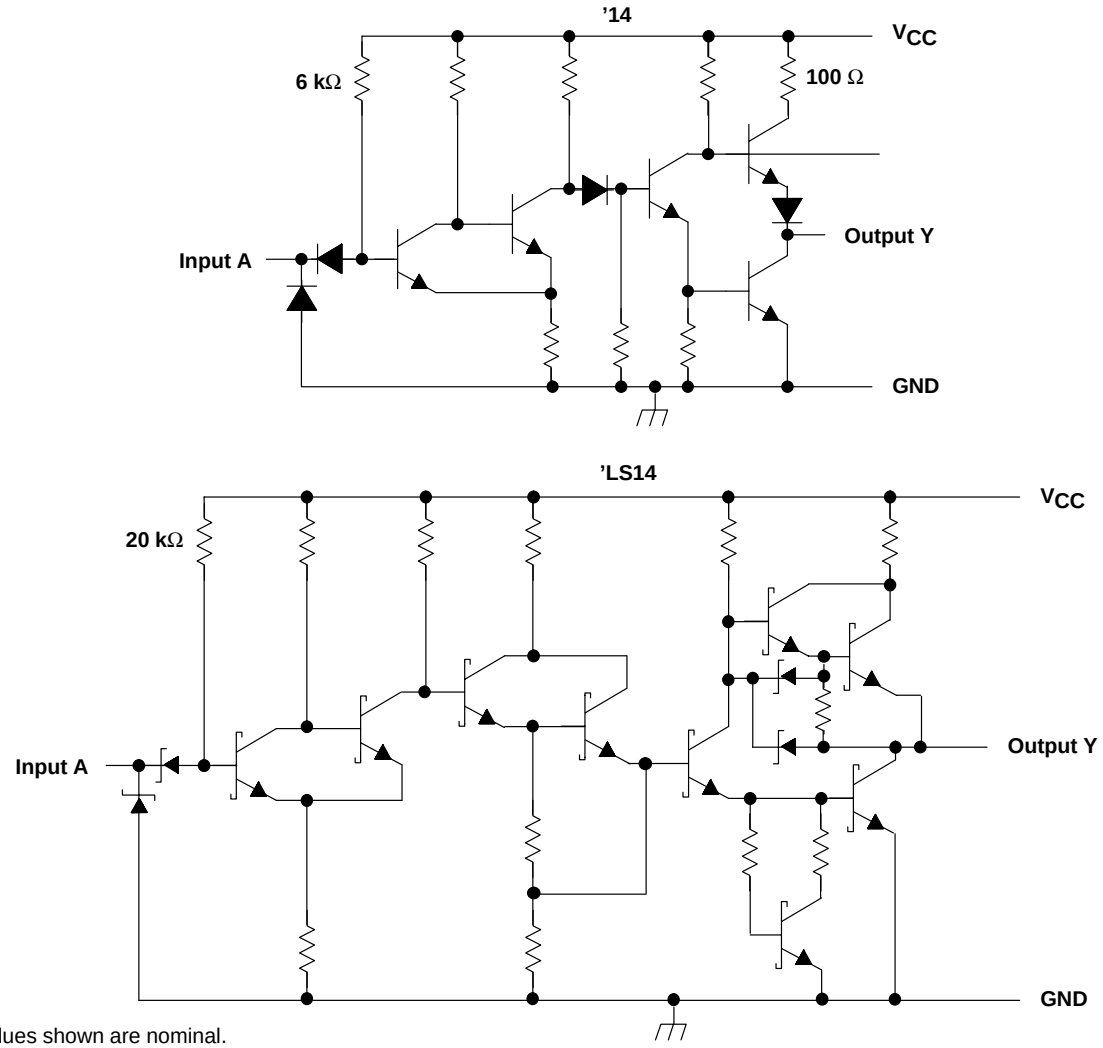
Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

logic diagram (positive logic)



Pin numbers shown are for the D, DB, J, N, NS, and W packages.

schematic



Resistor values shown are nominal.

absolute maximum ratings over operating free-air temperature (unless otherwise noted)†

Supply voltage, V_{CC} (see Note 1) . . . . . 7 V
Input voltage: ’14 . . . . . 5.5 V
’LS14 . . . . . 7 V
Package thermal impedance, θ_{JA} (see Note 2): D package . . . . . 86°C/W
DB package . . . . . 96°C/W
N package . . . . . 80°C/W
NS package . . . . . 76°C/W
Storage temperaturerange, T_{stg} . . . . . –65°C to 150°C

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. Voltage values are with respect to network ground terminal.
2. The package termal impedance is calculated in accordance with JESD 51-7

recommended operating conditions

		SN5414			SN7414			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
I_{OH}	High-level output current			–0.8			–0.8	mA
I_{OL}	Low-level output current			16			16	mA
T_A	Operating free-air temperature	–55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS‡			SN5414 SN7414			UNIT
				MIN	TYP§	MAX	
V_{T+}	V_{CC} = 5 V			1.5	1.7	2	V
$V_{T–}$	V_{CC} = 5 V			0.6	0.9	1.1	V
Hysteresis ($V_{T+} – V_{T–}$)	V_{CC} = 5 V			0.4	0.8		V
V_{IK}	V_{CC} = MIN,	I_I = –12 mA				–1.5	V
V_{OH}	V_{CC} = MIN,	V_I = 0.6 V,	I_{OH} = –0.8 mA	2.4	3.4		V
V_{OL}	V_{CC} = MIN,	V_I = 2 V,	I_{OL} = 16 mA		0.2	0.4	V
I_{T+}	V_{CC} = 5 V,	$V_I = V_{T+}$			–0.43		mA
$I_{T–}$	V_{CC} = 5 V,	$V_I = V_{T–}$			–0.56		mA
I_I	V_{CC} = MAX,	V_I = 5.5 V				1	mA
I_{IH}	V_{CC} = MAX,	V_{IH} = 2.4 V				40	µA
I_{IL}	V_{CC} = MAX,	V_{IL} = 0.4 V			–0.8	–1.2	mA
I_{OS}¶	V_{CC} = MAX			–18		–55	mA
I_{CCH}	V_{CC} = MAX				22	36	mA
I_{CCL}	V_{CC} = MAX				39	60	mA

‡ For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
§ All typical values are at V_{CC} = 5 V, T_A = 25°C.
¶ Not more than one output should be shorted at a time.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	SN5414 SN7414			UNIT
				MIN	TYP	MAX	
t_{PLH}	A	Y	R_L = 400 Ω, C_L = 15 pF		15	22	ns
t_{PHL}					15	22	

recommended operating conditions

		SN54LS14			SN74LS14			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
I_{OH}	High-level output current			–0.4			–0.4	mA
I_{OL}	Low-level output current			4			8	mA
T_A	Operating free-air temperature	–55		125	0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

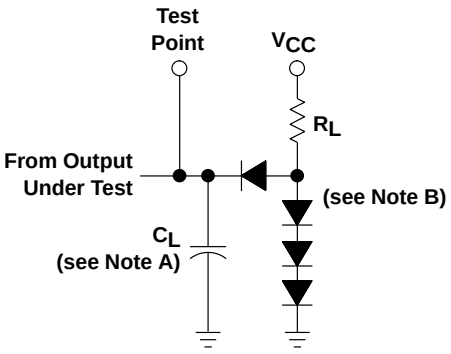
PARAMETER	TEST CONDITIONS†			SN54LS14			SN74LS14			UNIT
				MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{T+}	V_{CC} = 5 V			1.4	1.6	1.9	1.4	1.6	1.9	V
V_{T-}	V_{CC} = 5 V			0.5	0.8	1	0.5	0.8	1	V
Hysteresis ($V_{T+} - V_{T-}$)	V_{CC} = 5 V			0.4	0.8		0.4	0.8		V
V_{IK}	V_{CC} = MIN,	I_I = –18 mA				–1.5			–1.5	V
V_{OH}	V_{CC} = MIN,	V_I = 0.5 V,	I_{OH} = –0.4 mA	2.5	3.4		2.7	3.4		V
V_{OL}	V_{CC} = MIN,	V_I = –1.9 V	I_{OL} = 4 mA		0.25	0.4		0.25	0.4	V
			I_{OL} = 8 mA					0.35	0.5	
I_{T+}	V_{CC} = 5 V,	$V_I = V_{T+}$			–0.14			–0.14		mA
I_{T-}	V_{CC} = 5 V,	$V_I = V_{T-}$			–0.18			–0.18		mA
I_I	V_{CC} = MAX,	V_I = 7 V				0.1			0.1	mA
I_{IH}	V_{CC} = MAX,	V_{IH} = 2.7 V				20			20	μA
I_{IL}	V_{CC} = MAX,	V_{IL} = 0.4 V				–0.4			–0.4	mA
I_{OS}§	V_{CC} = MAX			–20		–100	–20		–100	mA
I_{CCH}	V_{CC} = MAX				8.6	16		8.6	16	mA
I_{CCL}	V_{CC} = MAX				12	21		12	21	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output should be shorted at a time, and duration of the short-circuit should not exceed one second.

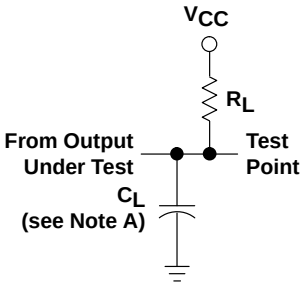
switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Figure 2)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PLH}	A	Y	R_L = 2 kΩ, C_L = 15 pF		15	22	ns
t_{PHL}					15	22	

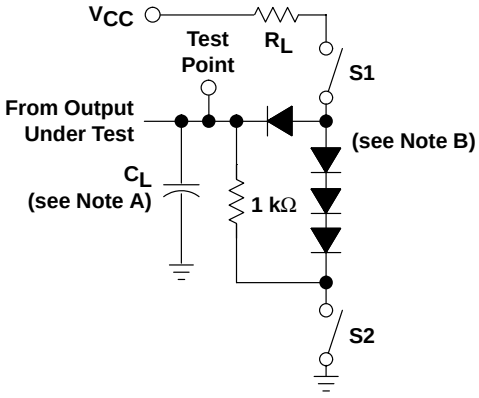
PARAMETER MEASUREMENT INFORMATION
SERIES 54/74 DEVICES



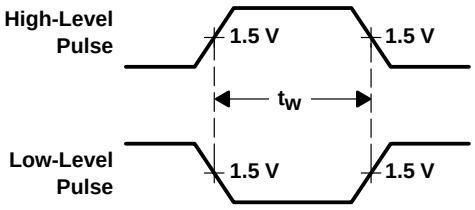
LOAD CIRCUIT
FOR 2-STATE TOTEM-POLE OUTPUTS



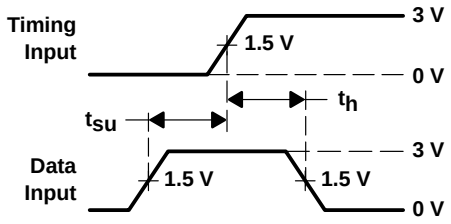
LOAD CIRCUIT
FOR OPEN-COLLECTOR OUTPUTS



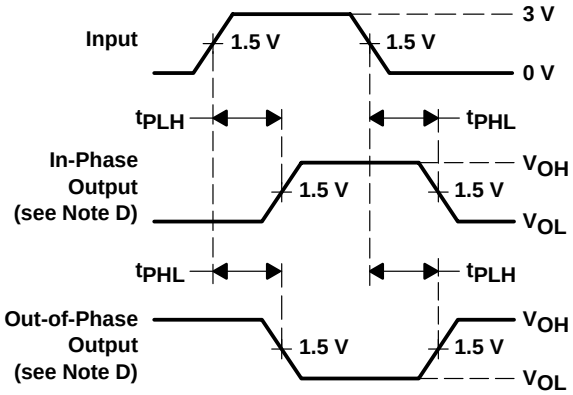
LOAD CIRCUIT
FOR 3-STATE OUTPUTS



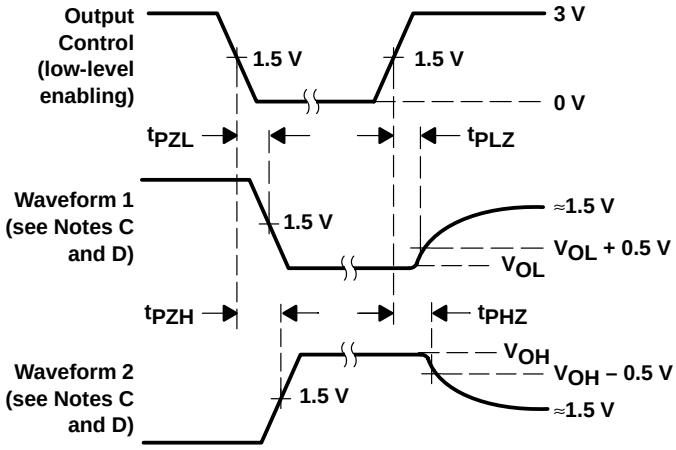
VOLTAGE WAVEFORMS
PULSE DURATIONS



VOLTAGE WAVEFORMS
SETUP AND HOLD TIMES



VOLTAGE WAVEFORMS
PROPAGATION DELAY TIMES

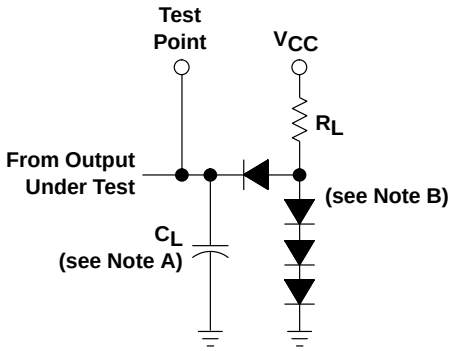


VOLTAGE WAVEFORMS
ENABLE AND DISABLE TIMES, 3-STATE OUTPUTS

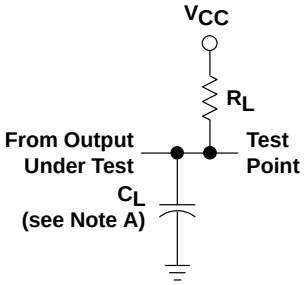
NOTES: A. C_L includes probe and jig capacitance.
B. All diodes are 1N3064 or equivalent.
C. Waveform 1 is for an output with internal conditions such that the output is low except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is high except when disabled by the output control.
D. S1 and S2 are closed for t_{PLH}, t_{PHL}, t_{PHZ}, and t_{PLZ}; S1 is open and S2 is closed for t_{PZH}; S1 is closed and S2 is open for t_{PZL}.
E. All input pulses are supplied by generators having the following characteristics: PRR ≤ 1 MHz, $Z_O \approx 50\ \Omega$; t_r and $t_f \leq 7$ ns for Series 54/74 devices and t_r and $t_f \leq 2.5$ ns for Series 54S/74S devices.
F. The outputs are measured one at a time with one input transition per measurement.

Figure 1. Load Circuits and Voltage Waveforms

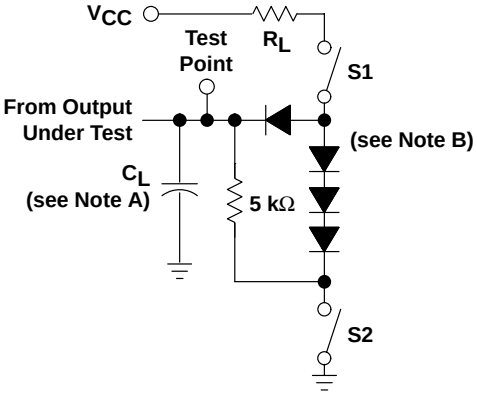
PARAMETER MEASUREMENT INFORMATION
SERIES 54LS/74LS DEVICES



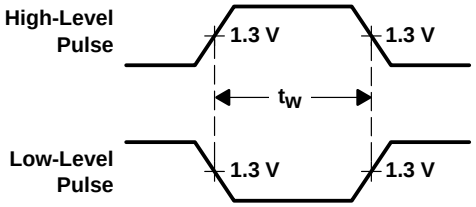
LOAD CIRCUIT
FOR 2-STATE TOTEM-POLE OUTPUTS



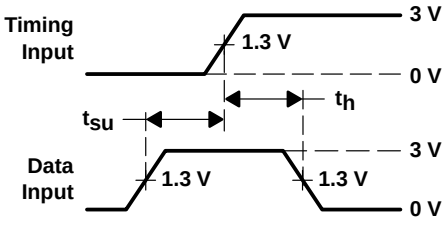
LOAD CIRCUIT
FOR OPEN-COLLECTOR OUTPUTS



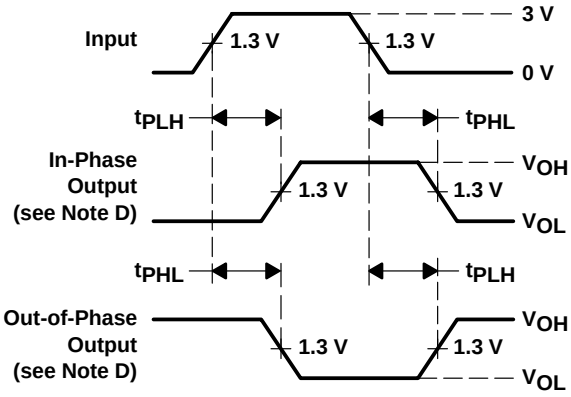
LOAD CIRCUIT
FOR 3-STATE OUTPUTS



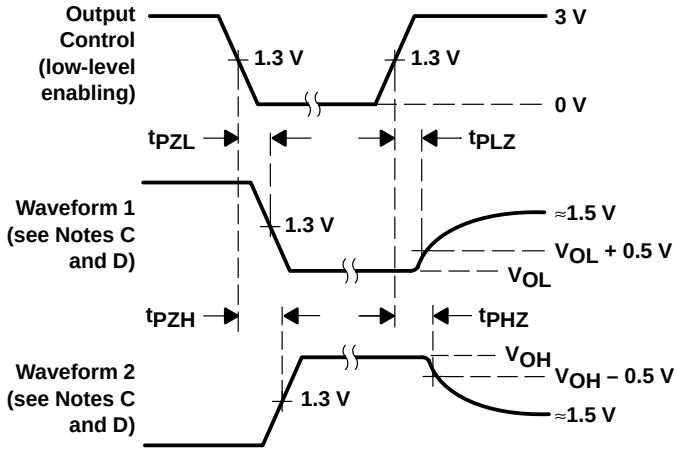
VOLTAGE WAVEFORMS
PULSE DURATIONS



VOLTAGE WAVEFORMS
SETUP AND HOLD TIMES



VOLTAGE WAVEFORMS
PROPAGATION DELAY TIMES



VOLTAGE WAVEFORMS
ENABLE AND DISABLE TIMES, 3-STATE OUTPUTS

NOTES: A. C_L includes probe and jig capacitance.
B. All diodes are 1N3064 or equivalent.
C. Waveform 1 is for an output with internal conditions such that the output is low except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is high except when disabled by the output control.
D. S1 and S2 are closed for t_{PLH}, t_{PHL}, t_{PHZ}, and t_{PLZ}; S1 is open and S2 is closed for t_{PZH}; S1 is closed and S2 is open for t_{PZL}.
E. Phase relationships between inputs and outputs have been chosen arbitrarily for these examples.
F. All input pulses are supplied by generators having the following characteristics: PRR ≤ 1 MHz, Z_O ≈ 50 Ω, t_r ≤ 1.5 ns, t_f ≤ 2.6 ns.
G. The outputs are measured one at a time with one input transition per measurement.

Figure 2. Load Circuits and Voltage Waveforms

TYPICAL CHARACTERISTICS OF '14 CIRCUITS†

POSITIVE-GOING THRESHOLD VOLTAGE vs FREE-AIR TEMPERATURE

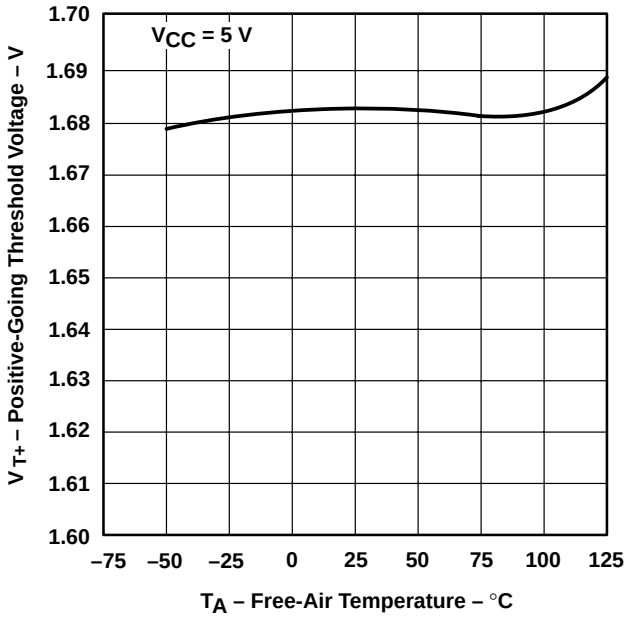


Figure 3

NEGATIVE-GOING THRESHOLD VOLTAGE vs FREE-AIR TEMPERATURE

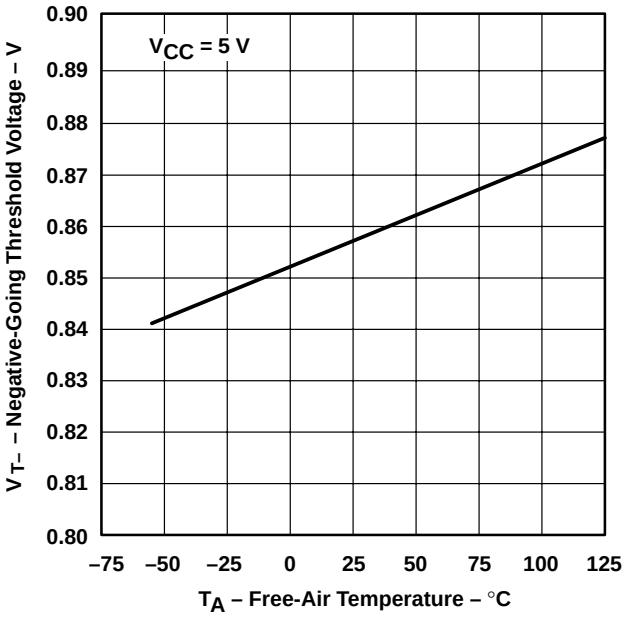


Figure 4

HYSTERESIS vs FREE-AIR TEMPERATURE

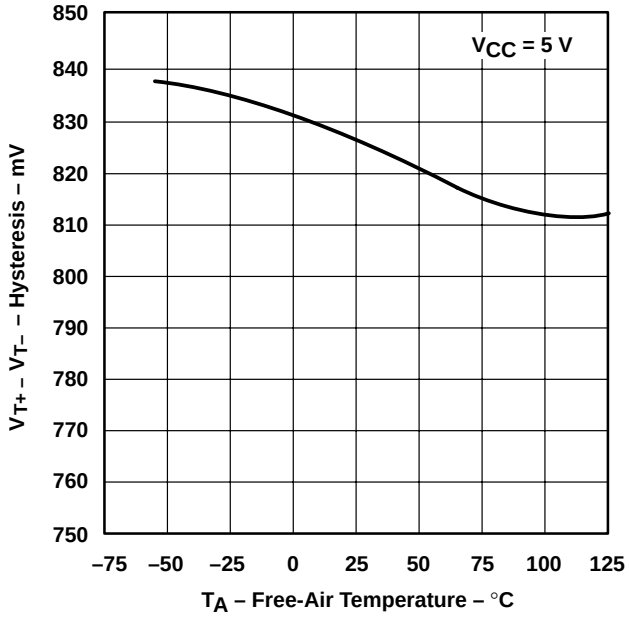


Figure 5

† Data for temperatures below 0°C and above 70°C and supply voltage below 4.75 V and above 5.25 V are applicable for SN5414 only.

TYPICAL CHARACTERISTICS OF ’14 CIRCUITS†

DISTRIBUTION OF UNITS FOR HYSTERESIS

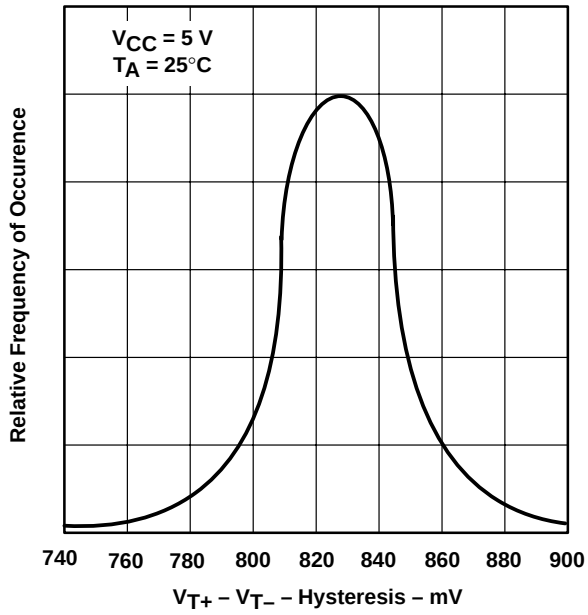


Figure 6

THRESHOLD VOLTAGES vs SUPPLY VOLTAGE

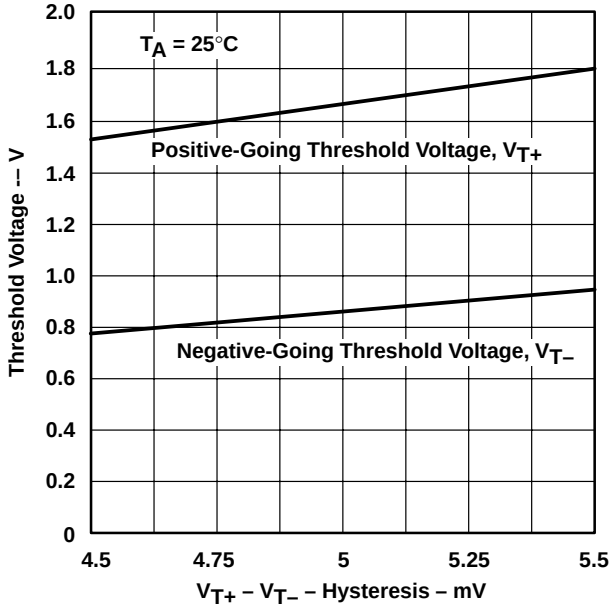


Figure 7

HYSTERESIS vs SUPPLY VOLTAGE

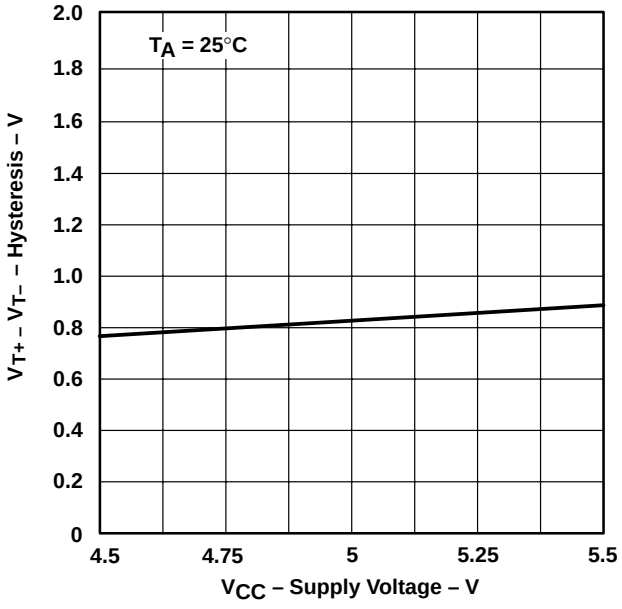


Figure 8

OUTPUT VOLTAGE vs INPUT VOLTAGE

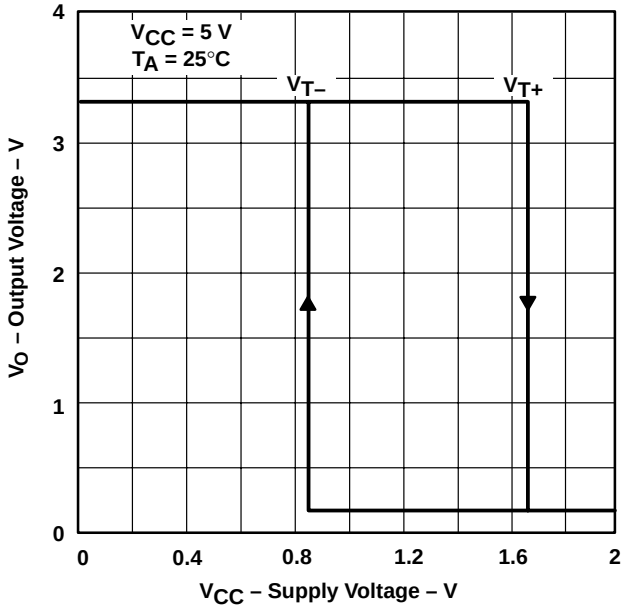


Figure 9

† Data for temperatures below 0°C and above 70°C and supply voltage below 4.75 V and above 5.25 V are applicable for SN5414 only.

TYPICAL CHARACTERISTICS OF 'LS14 CIRCUITS†

POSITIVE-GOING THRESHOLD VOLTAGE vs FREE-AIR TEMPERATURE

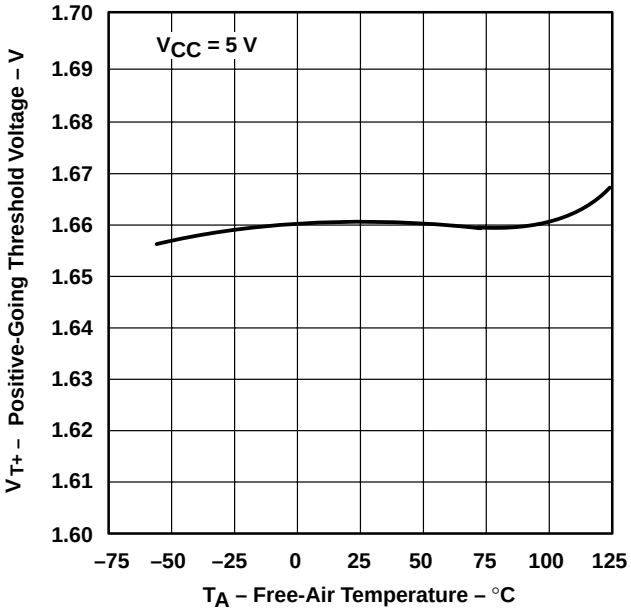


Figure 10

NEGATIVE-GOING THRESHOLD VOLTAGE vs FREE-AIR TEMPERATURE

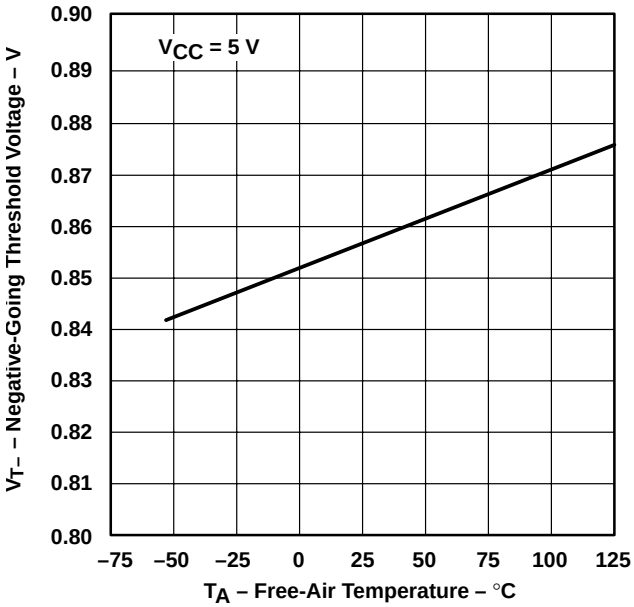


Figure 11

HYSTERESIS vs FREE-AIR TEMPERATURE

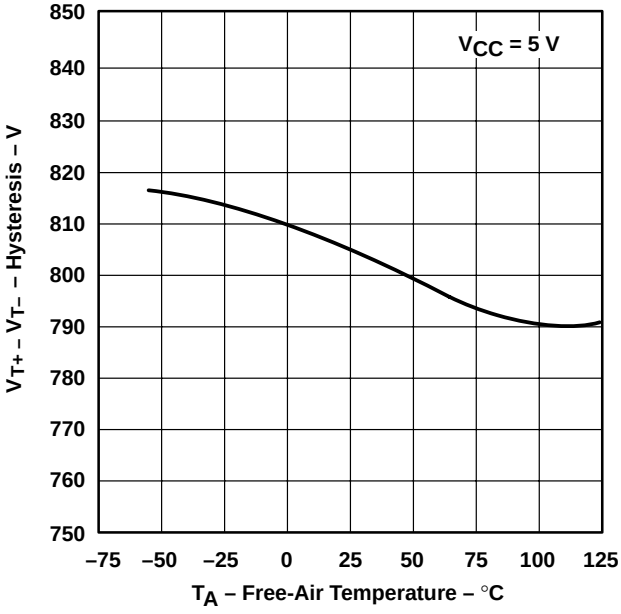


Figure 12

DISTRIBUTION OF UNITS FOR HYSTERESIS

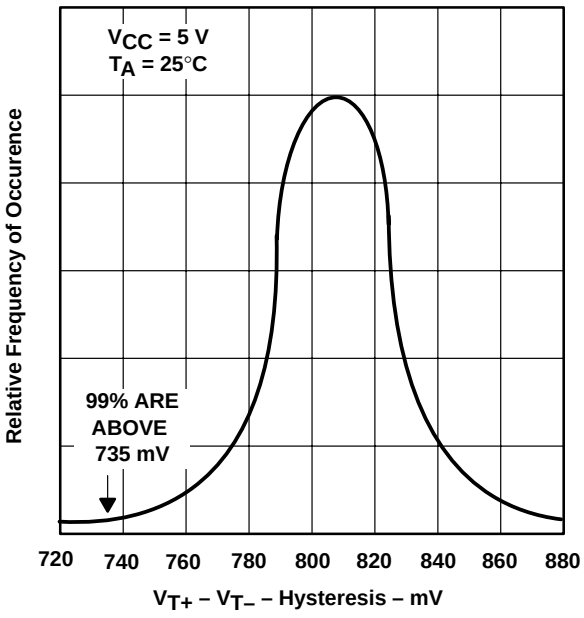


Figure 13

† Data for temperatures below 0°C and above 70°C and supply voltage below 4.75 V and above 5.25 V are applicable for SN5414 only.

TYPICAL CHARACTERISTICS OF 'LS14 CIRCUITS†

THRESHOLD VOLTAGES AND HYSTERESIS vs SUPPLY VOLTAGE

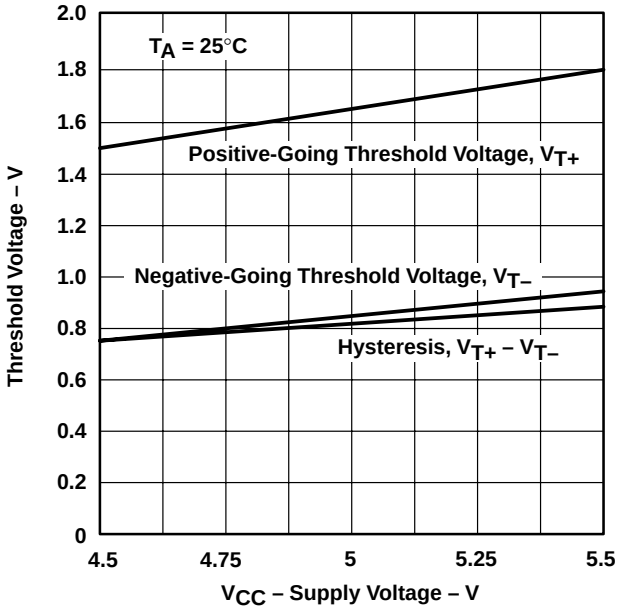


Figure 14

OUTPUT VOLTAGE vs INPUT VOLTAGE

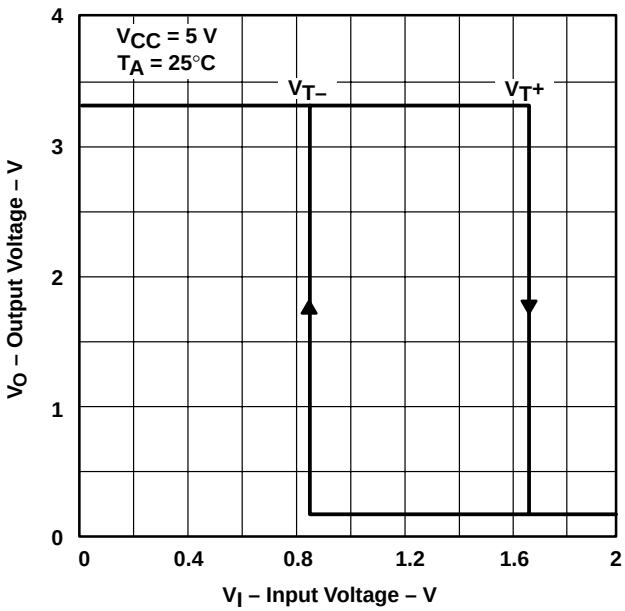
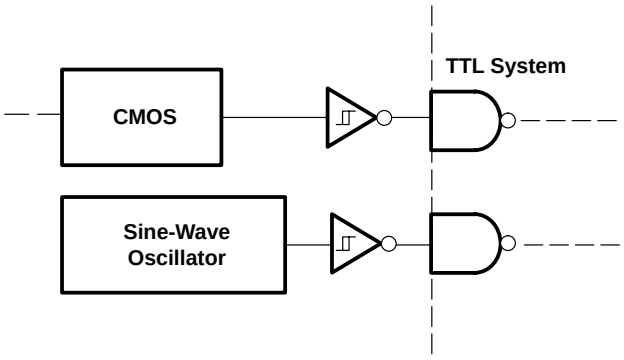


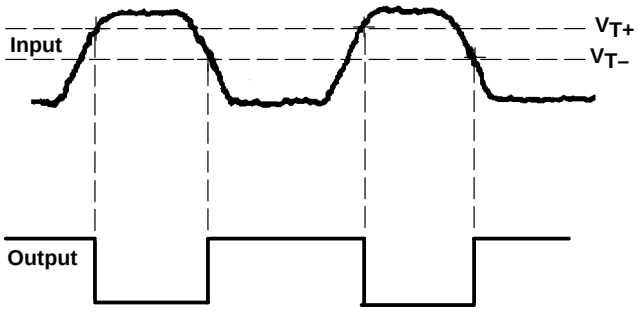
Figure 15

† Data for temperatures below 0°C and above 70°C and supply voltage below 4.75 V and above 5.25 V are applicable for SN5414 only.

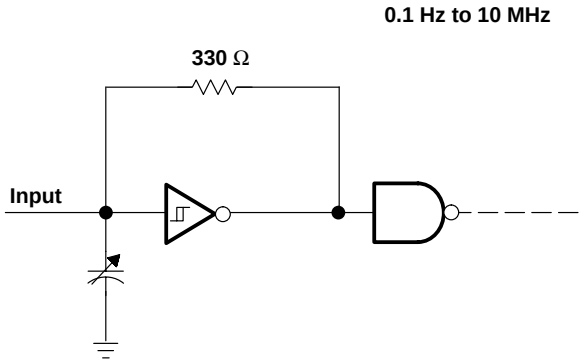
TYPICAL APPLICATION DATA



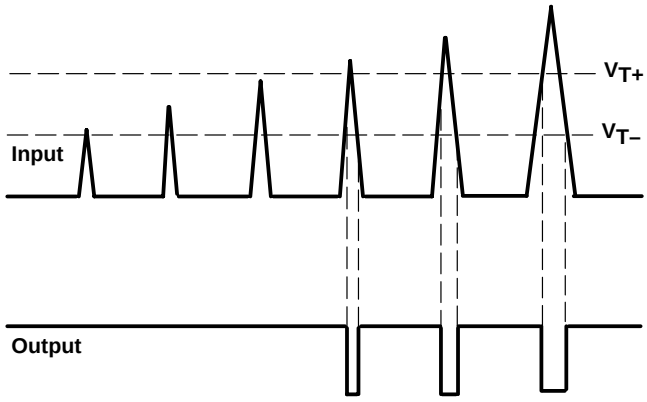
**TTL System Interface
for Slow Input Waveforms**



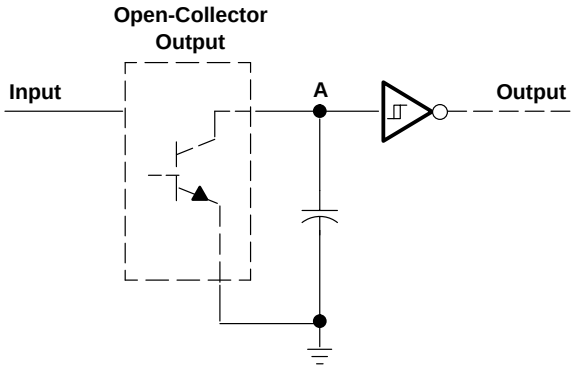
Pulse Shaper

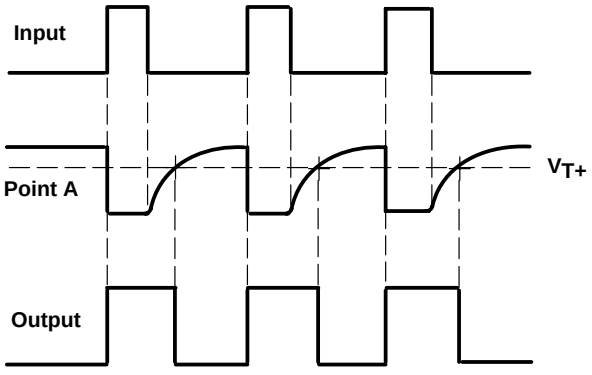


Multivibrator



Threshold Detector





Pulse Stretcher

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish	MSL Peak Temp (3)
5962-9665801Q2A	ACTIVE	LCCC	FK	20	1	TBD	POST-PLATE	N / A for Pkg Type
5962-9665801QCA	ACTIVE	CDIP	J	14	1	TBD	A42 SNPB	N / A for Pkg Type
5962-9665801QDA	ACTIVE	CFP	W	14	1	TBD	A42	N / A for Pkg Type
5962-9665801VCA	ACTIVE	CDIP	J	14	1	TBD	A42 SNPB	N / A for Pkg Type
5962-9665801VDA	ACTIVE	CFP	W	14	1	TBD	A42	N / A for Pkg Type
JM38510/31302BCA	ACTIVE	CDIP	J	14	1	TBD	A42 SNPB	N / A for Pkg Type
SN5414J	ACTIVE	CDIP	J	14	1	TBD	A42 SNPB	N / A for Pkg Type
SN54LS14J	ACTIVE	CDIP	J	14	1	TBD	A42 SNPB	N / A for Pkg Type
SN7414D	ACTIVE	SOIC	D	14	50	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN7414DE4	ACTIVE	SOIC	D	14	50	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN7414DG4	ACTIVE	SOIC	D	14	50	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN7414DR	ACTIVE	SOIC	D	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN7414DRE4	ACTIVE	SOIC	D	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN7414DRG4	ACTIVE	SOIC	D	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN7414N	ACTIVE	PDIP	N	14	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type
SN7414N3	OBSOLETE	PDIP	N	14		TBD	Call TI	Call TI
SN7414NE4	ACTIVE	PDIP	N	14	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type
SN7414NSR	ACTIVE	SO	NS	14	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN7414NSRE4	ACTIVE	SO	NS	14	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN7414NSRG4	ACTIVE	SO	NS	14	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14D	ACTIVE	SOIC	D	14	50	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14DBR	ACTIVE	SSOP	DB	14	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14DBRE4	ACTIVE	SSOP	DB	14	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14DBRG4	ACTIVE	SSOP	DB	14	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14DE4	ACTIVE	SOIC	D	14	50	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14DG4	ACTIVE	SOIC	D	14	50	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14DR	ACTIVE	SOIC	D	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14DRE4	ACTIVE	SOIC	D	14	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14DRG4	ACTIVE	SOIC	D	14	2500	Green (RoHS &	CU NIPDAU	Level-1-260C-UNLIM

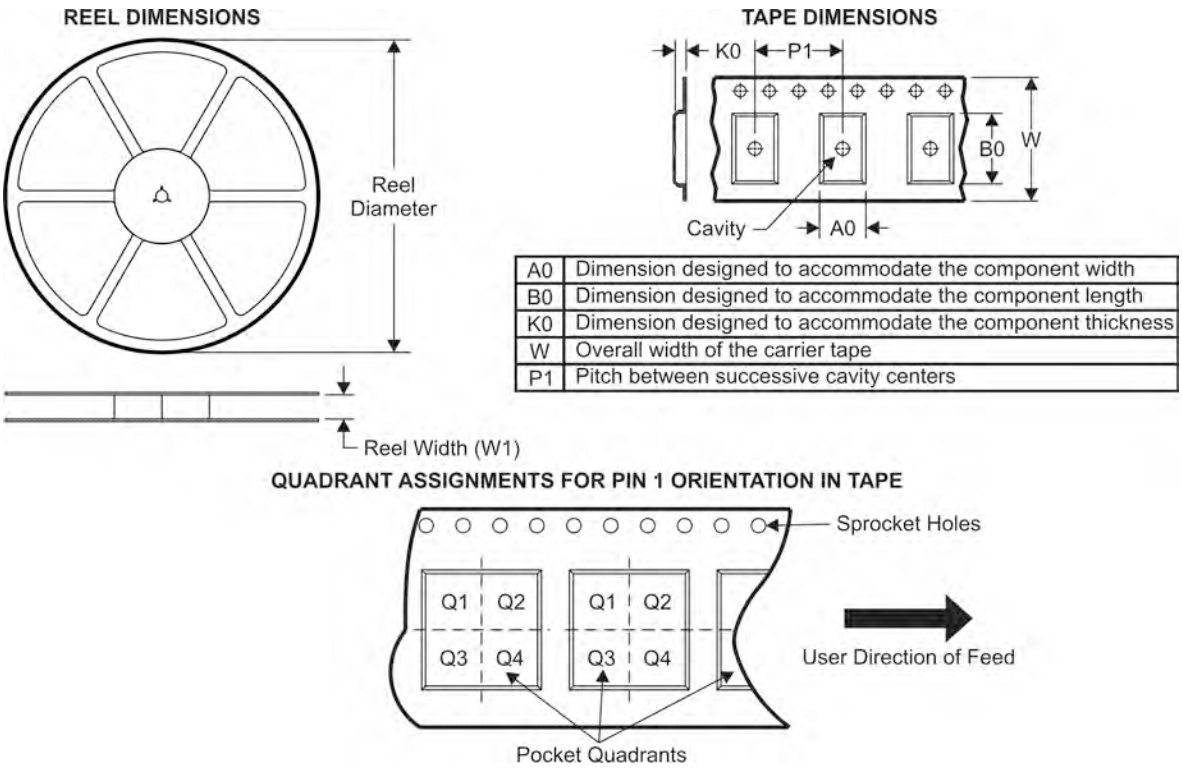
Orderable Device	Status [1]	Package Type	Package Drawing	Pins	Package Qty	Eco Plan [2]	Lead/Ball Finish	MSL Peak Temp [3]
						no Sb/Br)		
SN74LS14N	ACTIVE	PDIP	N	14	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type
SN74LS14N3	OBSOLETE	PDIP	N	14		TBD	Call TI	Call TI
SN74LS14NE4	ACTIVE	PDIP	N	14	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type
SN74LS14NSR	ACTIVE	SO	NS	14	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14NSRE4	ACTIVE	SO	NS	14	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SN74LS14NSRG4	ACTIVE	SO	NS	14	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM
SNJ5414J	ACTIVE	CDIP	J	14	1	TBD	A42 SNPB	N / A for Pkg Type
SNJ5414W	ACTIVE	CFP	W	14	1	TBD	A42	N / A for Pkg Type
SNJ54LS14FK	ACTIVE	LCCC	FK	20	1	TBD	POST-PLATE	N / A for Pkg Type
SNJ54LS14J	ACTIVE	CDIP	J	14	1	TBD	A42 SNPB	N / A for Pkg Type
SNJ54LS14W	ACTIVE	CFP	W	14	1	TBD	A42	N / A for Pkg Type

[1] The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

[2] Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

[3] MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

TAPE AND REEL INFORMATION

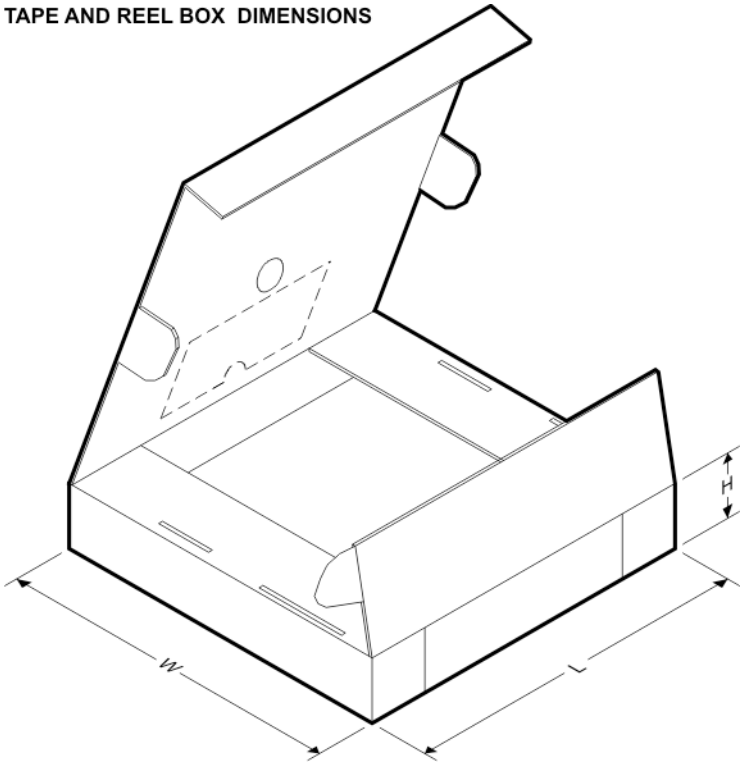


A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
SN7414DR	SOIC	D	14	2500	330.0	16.4	6.5	9.0	2.1	8.0	16.0	Q1
SN7414NSR	SO	NS	14	2000	330.0	16.4	8.2	10.5	2.5	12.0	16.0	Q1
SN74LS14DBR	SSOP	DB	14	2000	330.0	16.4	8.2	6.6	2.5	12.0	16.0	Q1
SN74LS14DR	SOIC	D	14	2500	330.0	16.4	6.5	9.0	2.1	8.0	16.0	Q1
SN74LS14NSR	SO	NS	14	2000	330.0	16.4	8.2	10.5	2.5	12.0	16.0	Q1

TAPE AND REEL BOX DIMENSIONS



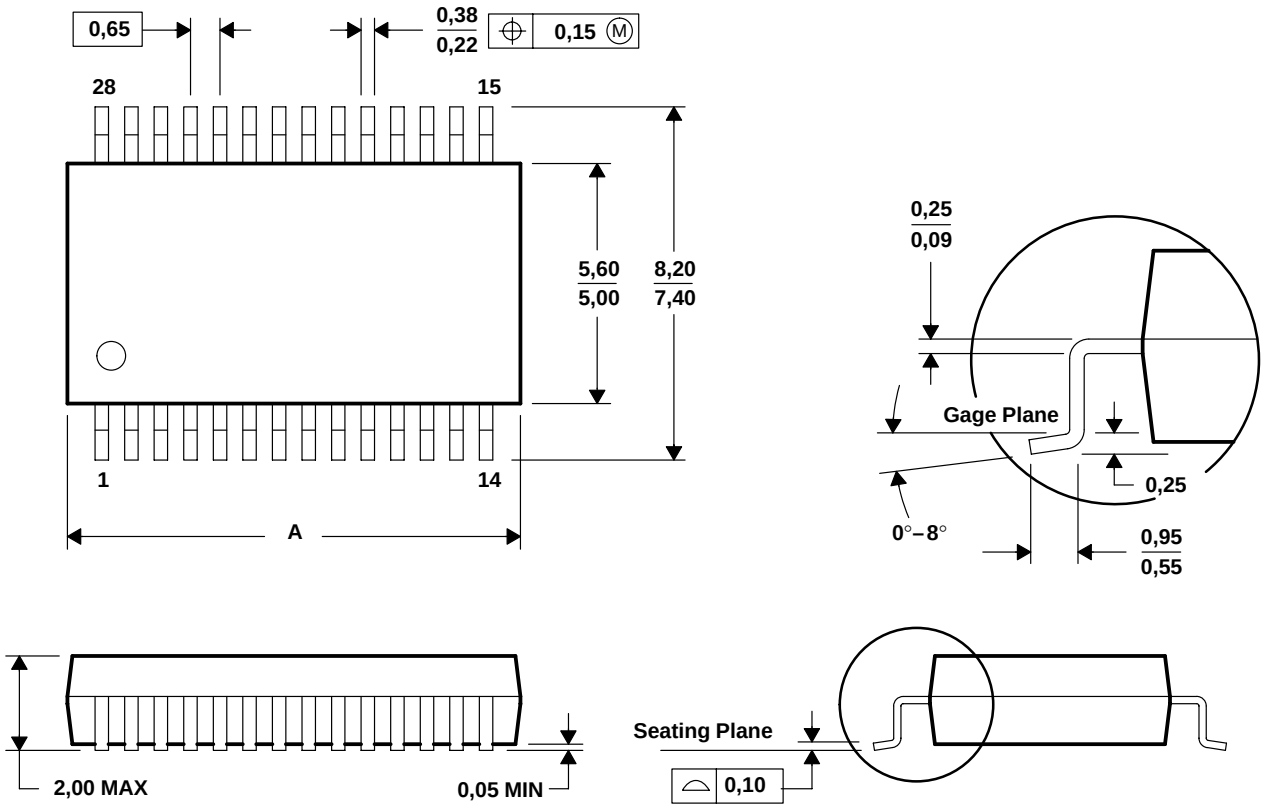
*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
SN7414DR	SOIC	D	14	2500	346.0	346.0	33.0
SN7414NSR	SO	NS	14	2000	346.0	346.0	33.0
SN74LS14DBR	SSOP	DB	14	2000	346.0	346.0	33.0
SN74LS14DR	SOIC	D	14	2500	346.0	346.0	33.0
SN74LS14NSR	SO	NS	14	2000	346.0	346.0	33.0

MSSO002E – JANUARY 1995 – REVISED DECEMBER 2001

DB (R-PDSO-G)** **PLASTIC SMALL-OUTLINE**

28 PINS SHOWN



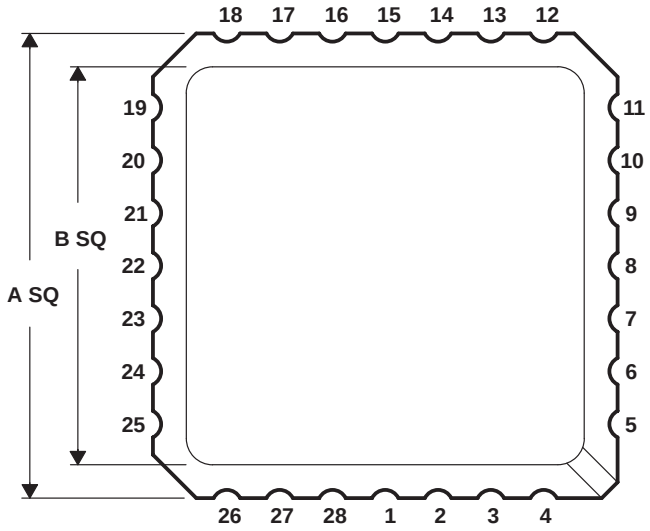
DIM \ PINS **	14	16	20	24	28	30	38
A MAX	6,50	6,50	7,50	8,50	10,50	10,50	12,90
A MIN	5,90	5,90	6,90	7,90	9,90	9,90	12,30

4040065 /E 12/01

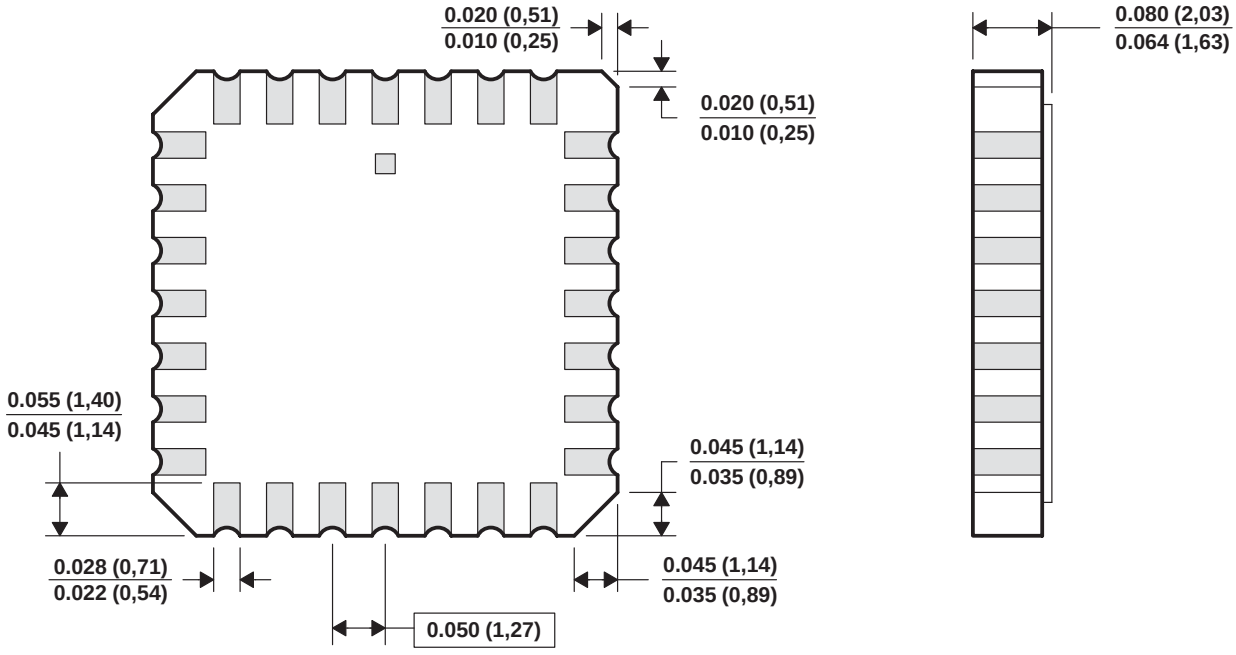
NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion not to exceed 0,15.
D. Falls within JEDEC MO-150

FK (S-CQCC-N)** **LEADLESS CERAMIC CHIP CARRIER**

28 TERMINAL SHOWN



NO. OF TERMINALS **	A		B	
	MIN	MAX	MIN	MAX
20	0.342 (8,69)	0.358 (9,09)	0.307 (7,80)	0.358 (9,09)
28	0.442 (11,23)	0.458 (11,63)	0.406 (10,31)	0.458 (11,63)
44	0.640 (16,26)	0.660 (16,76)	0.495 (12,58)	0.560 (14,22)
52	0.739 (18,78)	0.761 (19,32)	0.495 (12,58)	0.560 (14,22)
68	0.938 (23,83)	0.962 (24,43)	0.850 (21,6)	0.858 (21,8)
84	1.141 (28,99)	1.165 (29,59)	1.047 (26,6)	1.063 (27,0)



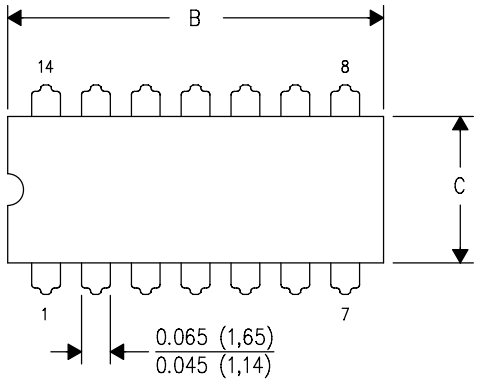
4040140/D 10/96

NOTES: A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. This package can be hermetically sealed with a metal lid.
D. The terminals are gold plated.
E. Falls within JEDEC MS-004

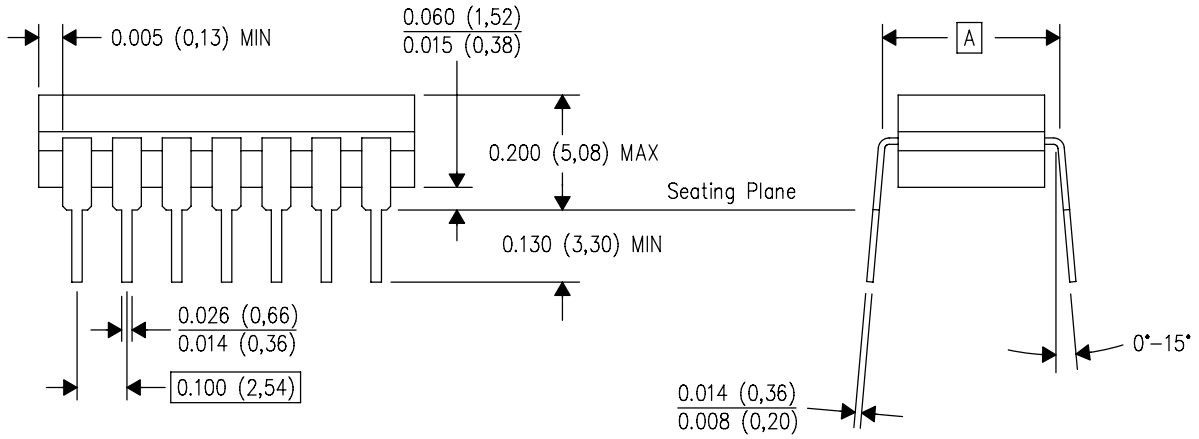
J (R–GDIP–T**)

14 LEADS SHOWN

CERAMIC DUAL IN–LINE PACKAGE



DIM \ PINS **	14	16	18	20
A	0.300 (7,62) BSC	0.300 (7,62) BSC	0.300 (7,62) BSC	0.300 (7,62) BSC
B MAX	0.785 (19,94)	.840 (21,34)	0.960 (24,38)	1.060 (26,92)
B MIN	—	—	—	—
C MAX	0.300 (7,62)	0.300 (7,62)	0.310 (7,87)	0.300 (7,62)
C MIN	0.245 (6,22)	0.245 (6,22)	0.220 (5,59)	0.245 (6,22)



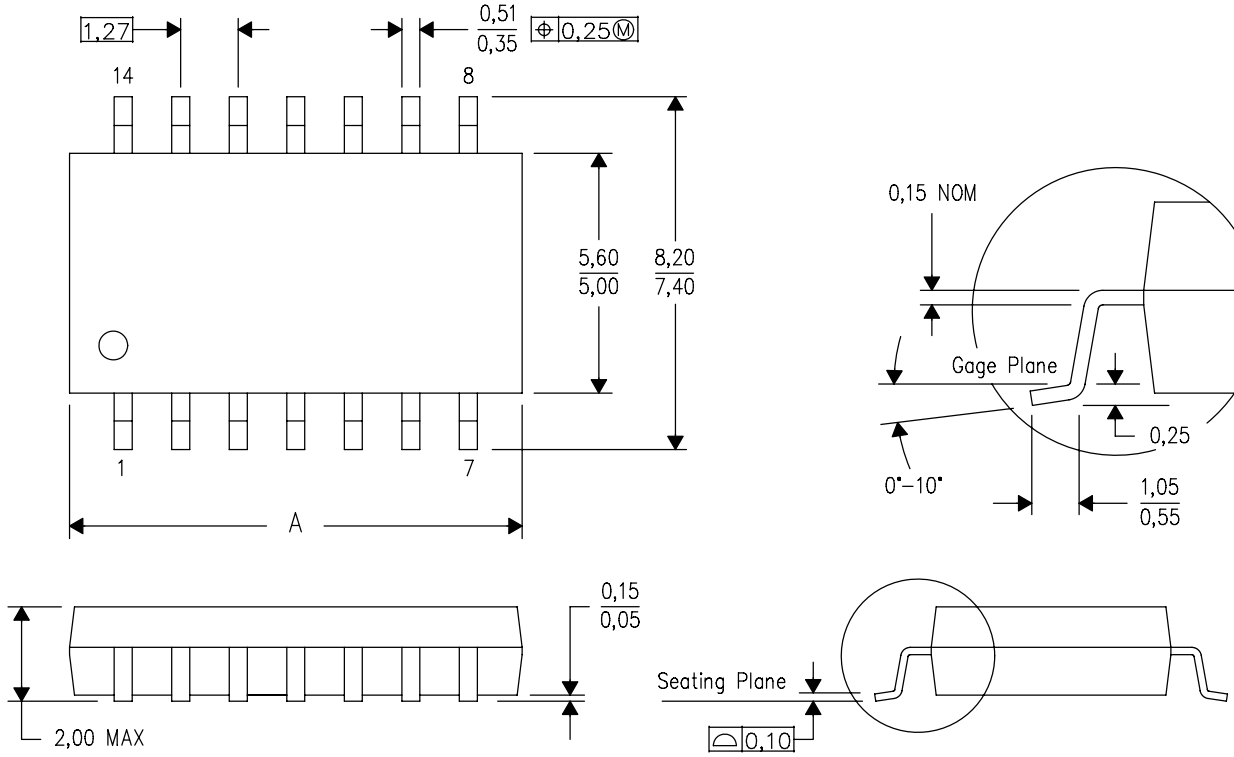
4040083/F 03/03

NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. This package is hermetically sealed with a ceramic lid using glass frit.
D. Index point is provided on cap for terminal identification only on press ceramic glass frit seal only.
E. Falls within MIL STD 1835 GDIP1–T14, GDIP1–T16, GDIP1–T18 and GDIP1–T20.

MECHANICAL DATA

NS (R-PDSO-G)** **PLASTIC SMALL-OUTLINE PACKAGE**

14-PINS SHOWN

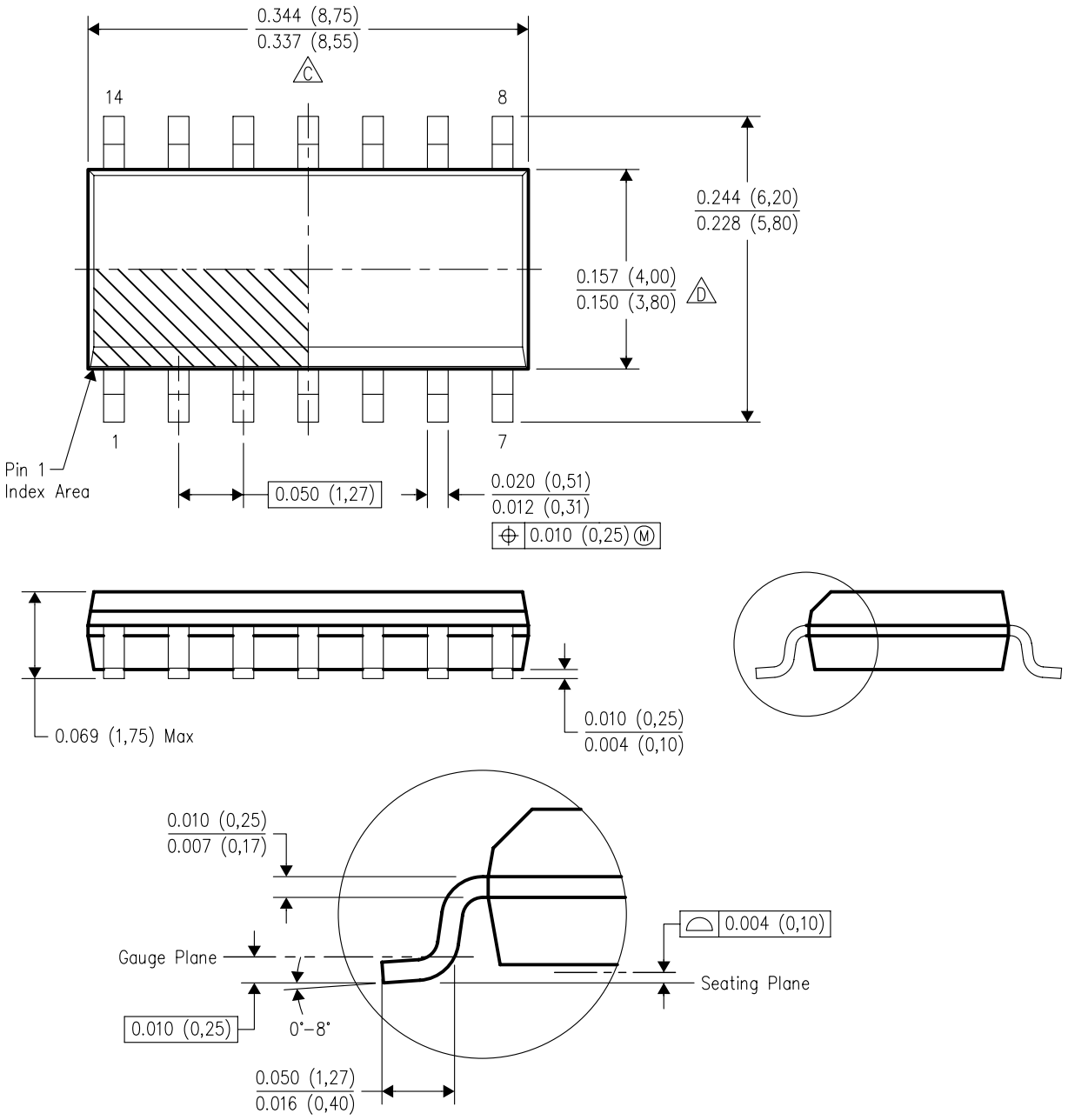


DIM \ PINS **	14	16	20	24
A MAX	10,50	10,50	12,90	15,30
A MIN	9,90	9,90	12,30	14,70

4040062/C 03/03

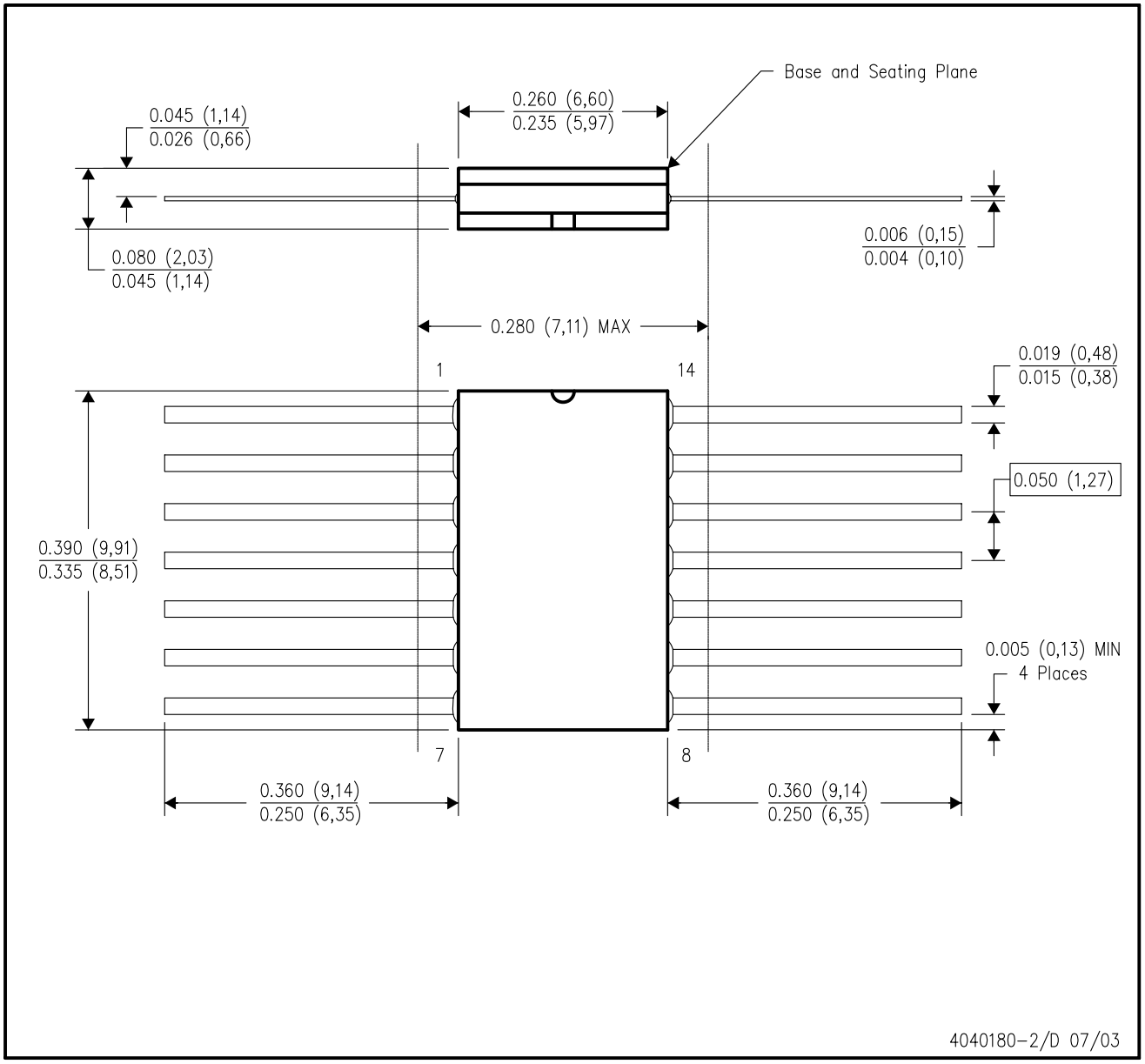
NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion, not to exceed 0,15.

D (R–PDSO–G14) PLASTIC SMALL–OUTLINE PACKAGE



NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Body length does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed .006 (0,15) per end.
D. Body width does not include interlead flash. Interlead flash shall not exceed .017 (0,43) per side.
E. Reference JEDEC MS–012 variation AB.

W (R–GDFP–F14) CERAMIC DUAL FLATPACK

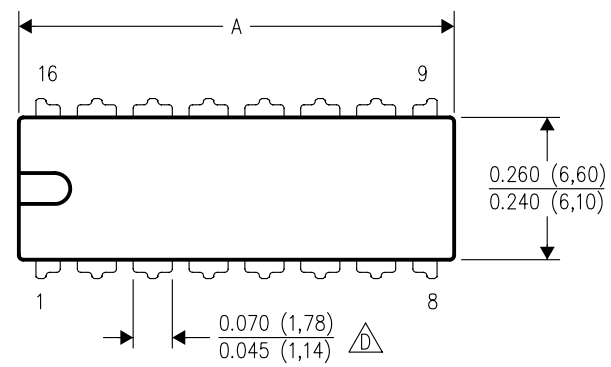


NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. This package can be hermetically sealed with a ceramic lid using glass frit.
D. Index point is provided on cap for terminal identification only.
E. Falls within MIL STD 1835 GDFP1–F14 and JEDEC MO–092AB

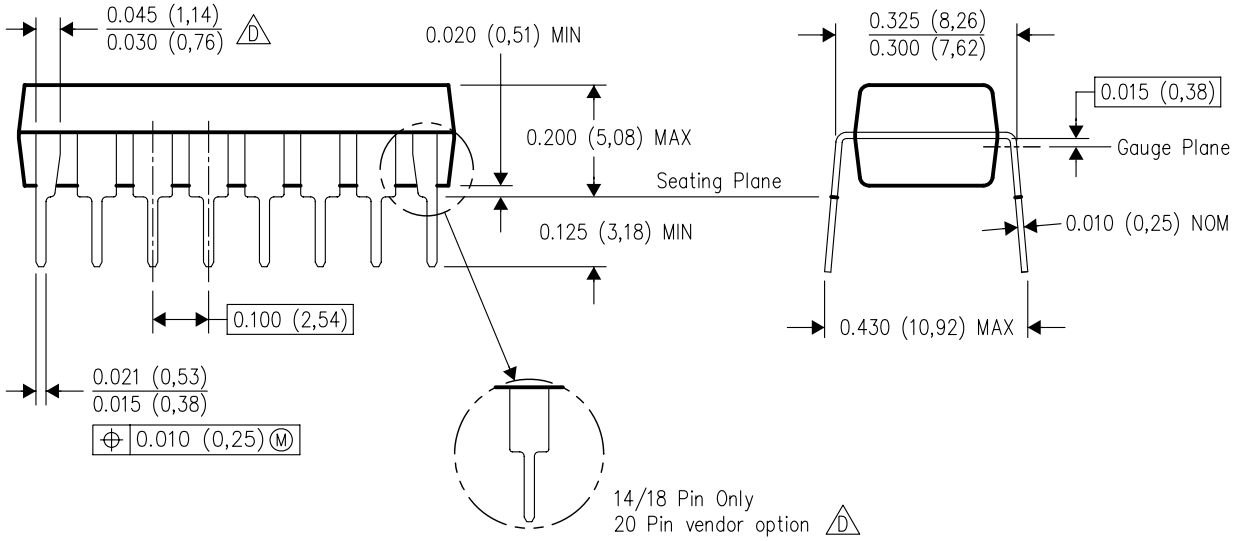
N (R–PDIP–T**)

PLASTIC DUAL–IN–LINE PACKAGE

16 PINS SHOWN



DIM \ PINS **	14	16	18	20
A MAX	0.775 (19,69)	0.775 (19,69)	0.920 (23,37)	1.060 (26,92)
A MIN	0.745 (18,92)	0.745 (18,92)	0.850 (21,59)	0.940 (23,88)
MS–001 VARIATION (C)	AA	BB	AC	AD



4040049/E 12/2002

NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS–001, except 18 and 20 pin minimum body length (Dim A).
D. The 20 pin end lead shoulder width is a vendor option, either half or full width.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Clocks and Timers	www.ti.com/clocks	Digital Control	www.ti.com/digitalcontrol
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

National Semiconductor

July 2006

LM555
Timer

General Description

The LM555 is a highly stable device for generating accurate time delays or oscillation. Additional terminals are provided for triggering or resetting if desired. In the time delay mode of operation, the time is precisely controlled by one external resistor and capacitor. For astable operation as an oscillator, the free running frequency and duty cycle are accurately controlled with two external resistors and one capacitor. The circuit may be triggered and reset on falling waveforms, and the output circuit can source or sink up to 200mA or drive TTL circuits.

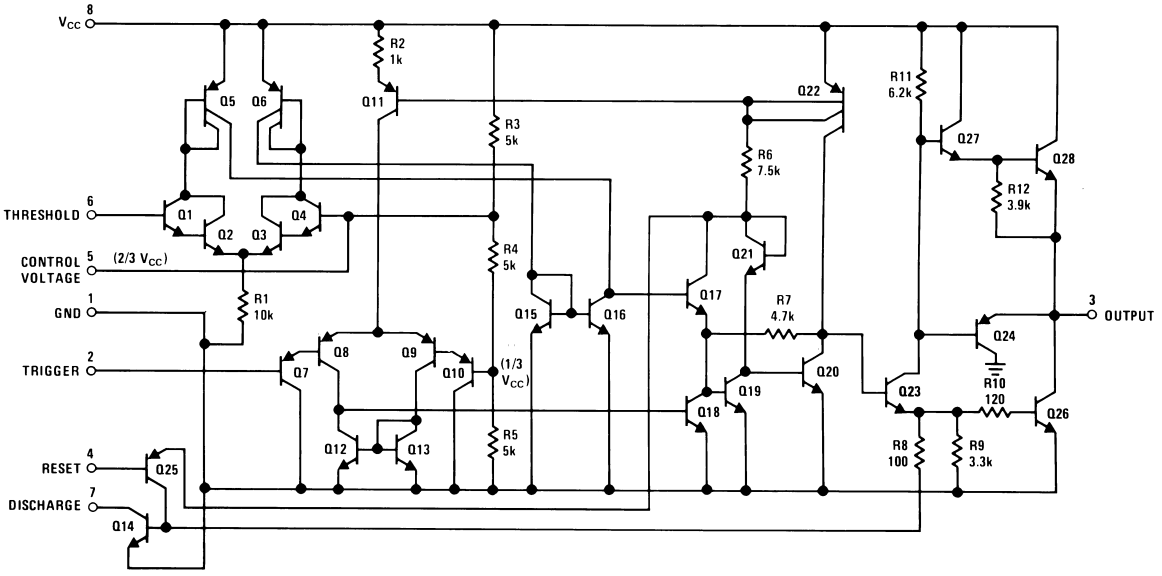
Features

- Direct replacement for SE555/NE555
- Timing from microseconds through hours
- Operates in both astable and monostable modes
- Adjustable duty cycle
- Output can source or sink 200 mA
- Output and supply TTL compatible
- Temperature stability better than 0.005% per ˚C
- Normally on and normally off output
- Available in 8-pin MSOP package

Applications

- Precision timing
- Pulse generation
- Sequential timing
- Time delay generation
- Pulse width modulation
- Pulse position modulation
- Linear ramp generator

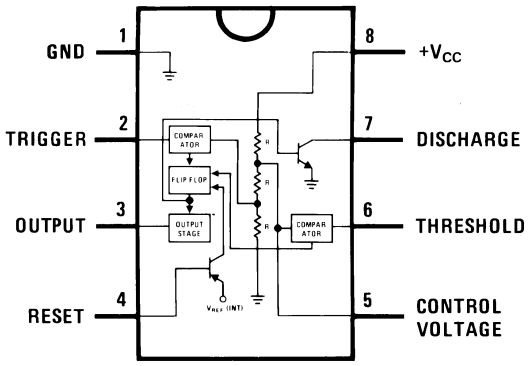
Schematic Diagram



00785101

Connection Diagram

Dual-In-Line, Small Outline and Molded Mini Small Outline Packages



00785103

Top View

Ordering Information

Package	Part Number	Package Marking	Media Transport	NSC Drawing
8-Pin SOIC	LM555CM	LM555CM	Rails	M08A
	LM555CMX	LM555CM	2.5k Units Tape and Reel	
8-Pin MSOP	LM555CMM	Z55	1k Units Tape and Reel	MUA08A
	LM555CMMX	Z55	3.5k Units Tape and Reel	
8-Pin MDIP	LM555CN	LM555CN	Rails	N08E

Absolute Maximum Ratings (Note 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+18V
Power Dissipation (Note 3)	
LM555CM, LM555CN	1180 mW
LM555CMM	613 mW
Operating Temperature Ranges	
LM555C	0˚C to +70˚C
Storage Temperature Range	−65˚C to +150˚C

Soldering Information

Dual-In-Line Package	
Soldering (10 Seconds)	260˚C
Small Outline Packages	
(SOIC and MSOP)	
Vapor Phase (60 Seconds)	215˚C
Infrared (15 Seconds)	220˚C

See AN-450 “Surface Mounting Methods and Their Effect on Product Reliability” for other methods of soldering surface mount devices.

Electrical Characteristics (Notes 1, 2)

(T_A = 25˚C, V_{CC} = +5V to +15V, unless othewise specified)

Parameter	Conditions	Limits			Units
		LM555C			
		Min	Typ	Max	
Supply Voltage		4.5		16	V
Supply Current	V_{CC} = 5V, R_L = ∞		3	6	
	V_{CC} = 15V, R_L = ∞ (Low State) (Note 4)		10	15	mA
Timing Error, Monostable					
Initial Accuracy			1		%
Drift with Temperature	R_A = 1k to 100kΩ, C = 0.1µF, (Note 5)		50		ppm/˚C
Accuracy over Temperature			1.5		%
Drift with Supply			0.1		%/V
Timing Error, Astable					
Initial Accuracy			2.25		%
Drift with Temperature	R_A, R_B = 1k to 100kΩ, C = 0.1µF, (Note 5)		150		ppm/˚C
Accuracy over Temperature			3.0		%
Drift with Supply			0.30		%/V
Threshold Voltage			0.667		x V_{CC}
Trigger Voltage	V_{CC} = 15V		5		V
	V_{CC} = 5V		1.67		V
Trigger Current			0.5	0.9	µA
Reset Voltage		0.4	0.5	1	V
Reset Current			0.1	0.4	mA
Threshold Current	(Note 6)		0.1	0.25	µA
Control Voltage Level	V_{CC} = 15V	9	10	11	V
	V_{CC} = 5V	2.6	3.33	4	
Pin 7 Leakage Output High			1	100	nA
Pin 7 Sat (Note 7)					
Output Low	V_{CC} = 15V, I_7 = 15mA		180		mV
Output Low	V_{CC} = 4.5V, I_7 = 4.5mA		80	200	mV

Electrical Characteristics (Notes 1, 2) (Continued)

(T_A = 25˚C, V_{CC} = +5V to +15V, unless othewise specified)

Parameter	Conditions	Limits			Units
		LM555C			
		Min	Typ	Max	
Output Voltage Drop (Low)	V_{CC} = 15V				
	I_{SINK} = 10mA		0.1	0.25	V
	I_{SINK} = 50mA		0.4	0.75	V
	I_{SINK} = 100mA		2	2.5	V
	I_{SINK} = 200mA		2.5		V
	V_{CC} = 5V				
	I_{SINK} = 8mA				V
	I_{SINK} = 5mA		0.25	0.35	V
Output Voltage Drop (High)	I_{SOURCE} = 200mA, V_{CC} = 15V		12.5		V
	I_{SOURCE} = 100mA, V_{CC} = 15V	12.75	13.3		V
	V_{CC} = 5V	2.75	3.3		V
Rise Time of Output			100		ns
Fall Time of Output			100		ns

Note 1: All voltages are measured with respect to the ground pin, unless otherwise specified.

Note 2: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not guarantee specific performance limits. Electrical Characteristics state DC and AC electrical specifications under particular test conditions which guarantee specific performance limits. This assumes that the device is within the Operating Ratings. Specifications are not guaranteed for parameters where no limit is given, however, the typical value is a good indication of device performance.

Note 3: For operating at elevated temperatures the device must be derated above 25˚C based on a +150˚C maximum junction temperature and a thermal resistance of 106˚C/W (DIP), 170˚C/W (S0-8), and 204˚C/W (MSOP) junction to ambient.

Note 4: Supply current when output high typically 1 mA less at V_{CC} = 5V.

Note 5: Tested at V_{CC} = 5V and V_{CC} = 15V.

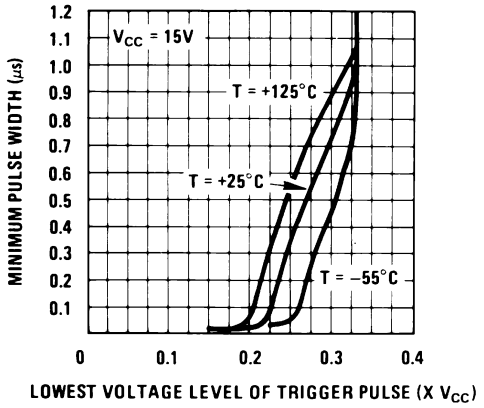
Note 6: This will determine the maximum value of $R_A + R_B$ for 15V operation. The maximum total ($R_A + R_B$) is 20MΩ.

Note 7: No protection against excessive pin 7 current is necessary providing the package dissipation rating will not be exceeded.

Note 8: Refer to RETS555X drawing of military LM555H and LM555J versions for specifications.

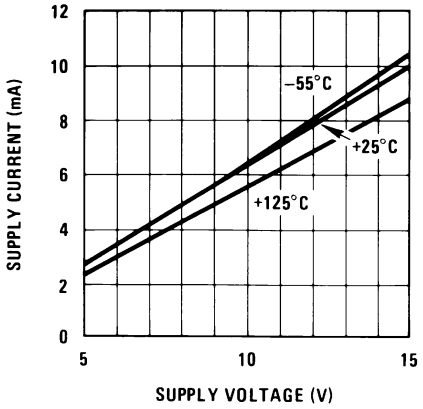
Typical Performance Characteristics

Minimuim Pulse Width Required for Triggering



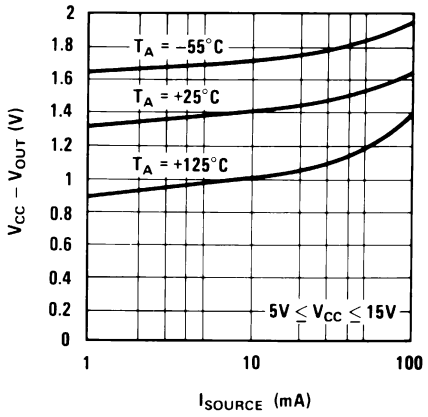
00785104

Supply Current vs. Supply Voltage



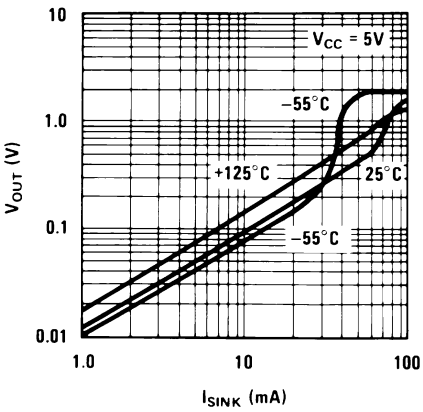
00785119

High Output Voltage vs. Output Source Current



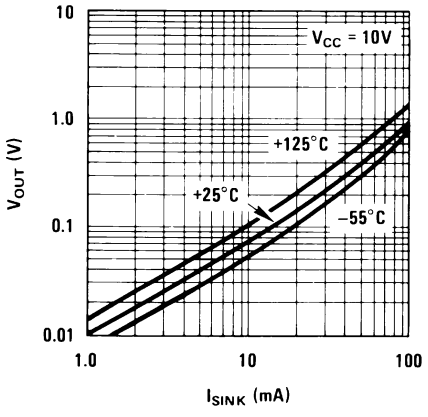
00785120

Low Output Voltage vs. Output Sink Current



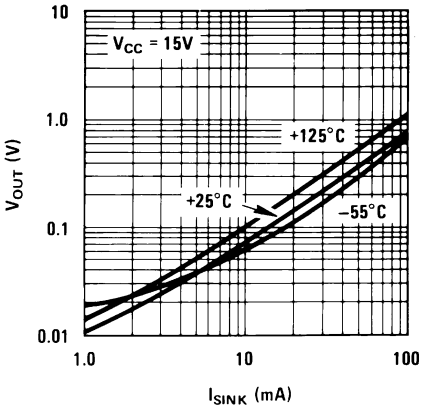
00785121

Low Output Voltage vs. Output Sink Current



00785122

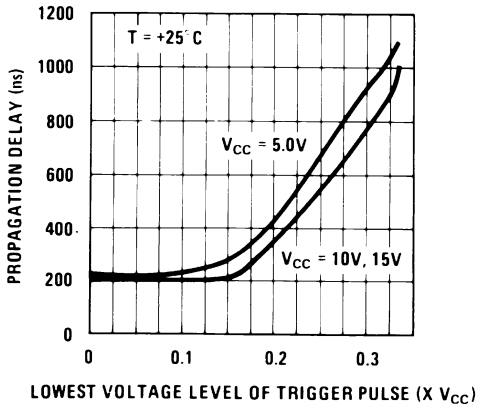
Low Output Voltage vs. Output Sink Current



00785123

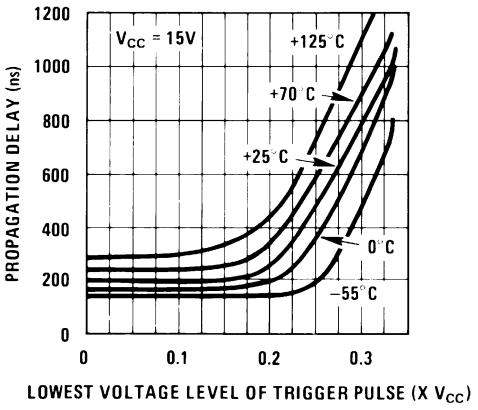
Typical Performance Characteristics (Continued)

Output Propagation Delay vs. Voltage Level of Trigger Pulse



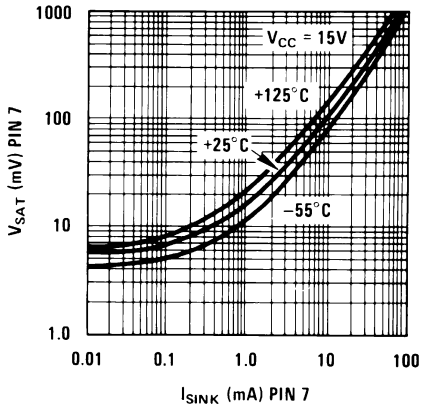
00785124

Output Propagation Delay vs. Voltage Level of Trigger Pulse



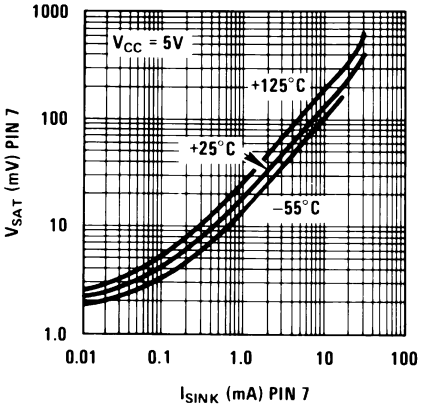
00785125

Discharge Transistor (Pin 7) Voltage vs. Sink Current



00785126

Discharge Transistor (Pin 7) Voltage vs. Sink Current



00785127

Applications Information

MONOSTABLE OPERATION

In this mode of operation, the timer functions as a one-shot (*Figure 1*). The external capacitor is initially held discharged by a transistor inside the timer. Upon application of a negative trigger pulse of less than 1/3 V_{CC} to pin 2, the flip-flop is set which both releases the short circuit across the capacitor and drives the output high.

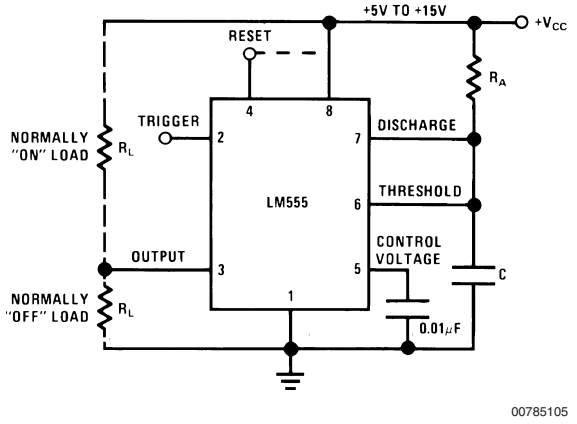
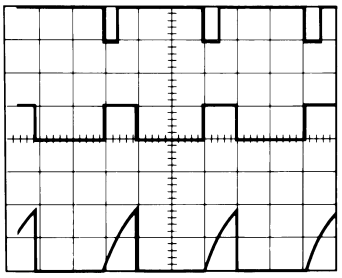


FIGURE 1. Monostable

The voltage across the capacitor then increases exponentially for a period of t = 1.1 R_A C, at the end of which time the voltage equals 2/3 V_{CC}. The comparator then resets the flip-flop which in turn discharges the capacitor and drives the output to its low state. *Figure 2* shows the waveforms generated in this mode of operation. Since the charge and the threshold level of the comparator are both directly proportional to supply voltage, the timing interval is independent of supply.



V_{CC} = 5V Top Trace: Input 5V/Div.
TIME = 0.1 ms/DIV. Middle Trace: Output 5V/Div.
R_A = 9.1kΩ Bottom Trace: Capacitor Voltage 2V/Div.
C = 0.01μF

FIGURE 2. Monostable Waveforms

During the timing cycle when the output is high, the further application of a trigger pulse will not effect the circuit so long as the trigger input is returned high at least 10μs before the end of the timing interval. However the circuit can be reset during this time by the application of a negative pulse to the reset terminal (pin 4). The output will then remain in the low state until a trigger pulse is again applied.

When the reset function is not in use, it is recommended that it be connected to V_{CC} to avoid any possibility of false triggering.

Figure 3 is a nomograph for easy determination of R, C values for various time delays.

NOTE: In monostable operation, the trigger should be driven high before the end of timing cycle.

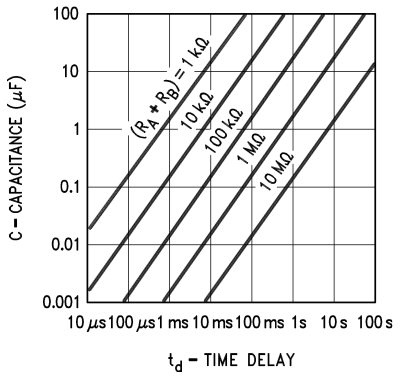


FIGURE 3. Time Delay

ASTABLE OPERATION

If the circuit is connected as shown in *Figure 4* (pins 2 and 6 connected) it will trigger itself and free run as a multivibrator. The external capacitor charges through R_A + R_B and discharges through R_B. Thus the duty cycle may be precisely set by the ratio of these two resistors.

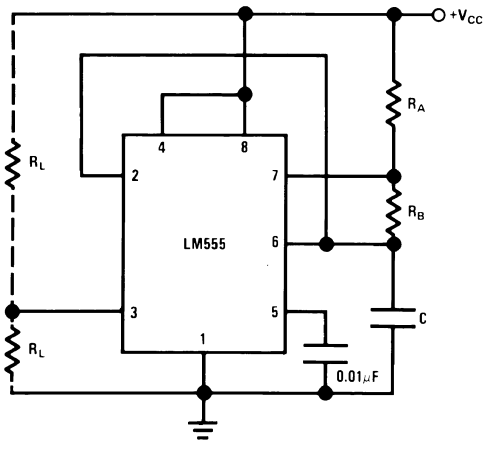
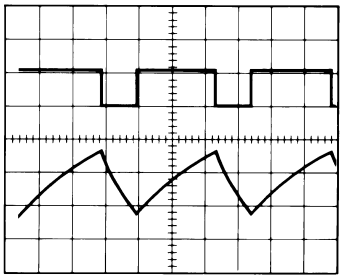


FIGURE 4. Astable

In this mode of operation, the capacitor charges and discharges between 1/3 V_{CC} and 2/3 V_{CC}. As in the triggered mode, the charge and discharge times, and therefore the frequency are independent of the supply voltage.

Applications Information (Continued)

Figure 5 shows the waveforms generated in this mode of operation.



00785109

V_{CC} = 5V Top Trace: Output 5V/Div.
TIME = 20µs/DIV. Bottom Trace: Capacitor Voltage 1V/Div.
R_A = 3.9kΩ
R_B = 3kΩ
C = 0.01µF

FIGURE 5. Astable Waveforms

The charge time (output high) is given by:

$$t_1 = 0.693\ (R_A + R_B)\ C$$

And the discharge time (output low) by:

$$t_2 = 0.693\ (R_B)\ C$$

Thus the total period is:

$$T = t_1 + t_2 = 0.693\ (R_A + 2R_B)\ C$$

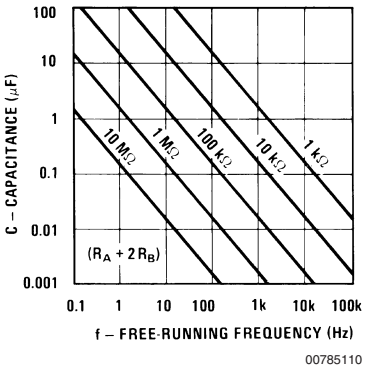
The frequency of oscillation is:

$$f = \frac{1}{T} = \frac{1.44}{(R_A + 2R_B)\ C}$$

Figure 6 may be used for quick determination of these RC values.

The duty cycle is:

$$D = \frac{R_B}{R_A + 2R_B}$$

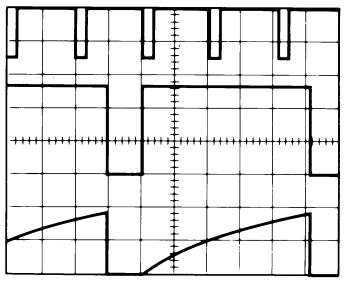


00785110

FIGURE 6. Free Running Frequency

FREQUENCY DIVIDER

The monostable circuit of *Figure 1* can be used as a frequency divider by adjusting the length of the timing cycle. *Figure 7* shows the waveforms generated in a divide by three circuit.



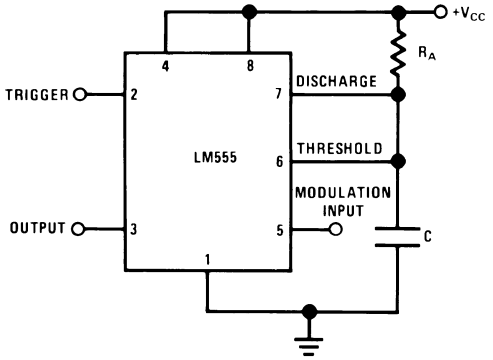
00785111

V_{CC} = 5V Top Trace: Input 4V/Div.
TIME = 20µs/DIV. Middle Trace: Output 2V/Div.
R_A = 9.1kΩ Bottom Trace: Capacitor 2V/Div.
C = 0.01µF

FIGURE 7. Frequency Divider

PULSE WIDTH MODULATOR

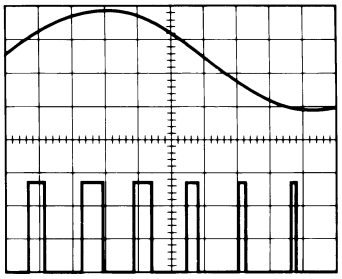
When the timer is connected in the monostable mode and triggered with a continuous pulse train, the output pulse width can be modulated by a signal applied to pin 5. *Figure 8* shows the circuit, and in *Figure 9* are some waveform examples.



00785112

FIGURE 8. Pulse Width Modulator

Applications Information (Continued)



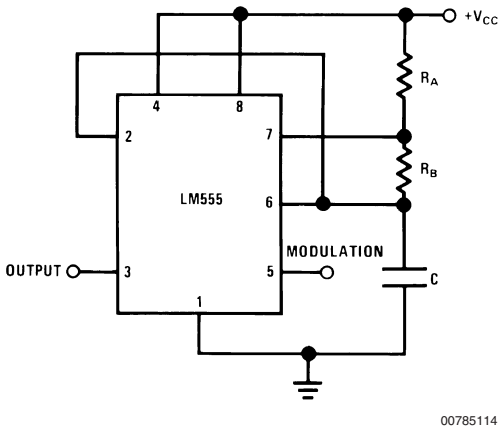
00785113

V_{CC} = 5V Top Trace: Modulation 1V/Div.
TIME = 0.2 ms/DIV. Bottom Trace: Output Voltage 2V/Div.
R_A = 9.1kΩ
C = 0.01µF

FIGURE 9. Pulse Width Modulator

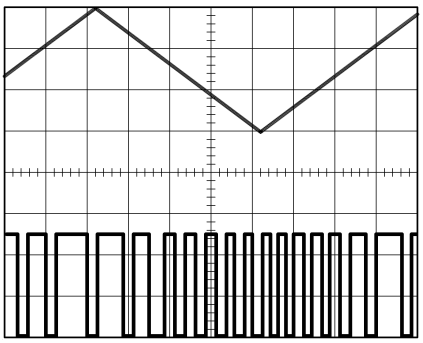
PULSE POSITION MODULATOR

This application uses the timer connected for astable operation, as in *Figure 10*, with a modulating signal again applied to the control voltage terminal. The pulse position varies with the modulating signal, since the threshold voltage and hence the time delay is varied. *Figure 11* shows the waveforms generated for a triangle wave modulation signal.



00785114

FIGURE 10. Pulse Position Modulator



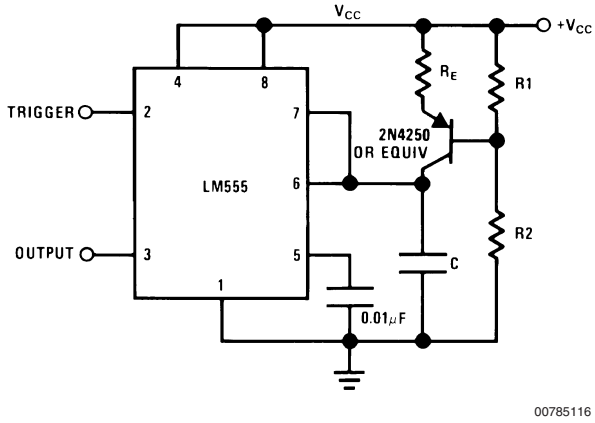
00785115

V_{CC} = 5V Top Trace: Modulation Input 1V/Div.
TIME = 0.1 ms/DIV. Bottom Trace: Output 2V/Div.
R_A = 3.9kΩ
R_B = 3kΩ
C = 0.01µF

FIGURE 11. Pulse Position Modulator

LINEAR RAMP

When the pullup resistor, R_A, in the monostable circuit is replaced by a constant current source, a linear ramp is generated. *Figure 12* shows a circuit configuration that will perform this function.



00785116

FIGURE 12.

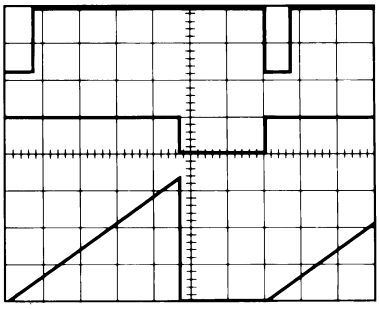
Figure 13 shows waveforms generated by the linear ramp. The time interval is given by:

$$T = \frac{2/3\ V_{CC}\ R_E\ (R_1 + R_2)\ C}{R_1\ V_{CC} - V_{BE}\ (R_1 + R_2)}$$

$$V_{BE} \cong 0.6V$$

$$V_{BE} \simeq 0.6V$$

Applications Information (Continued)



00785117

V_{CC} = 5V Top Trace: Input 3V/Div.
TIME = 20µs/DIV. Middle Trace: Output 5V/Div.
R_1 = 47kΩ Bottom Trace: Capacitor Voltage 1V/Div.
R_2 = 100kΩ
R_E = 2.7 kΩ
C = 0.01 µF

FIGURE 13. Linear Ramp

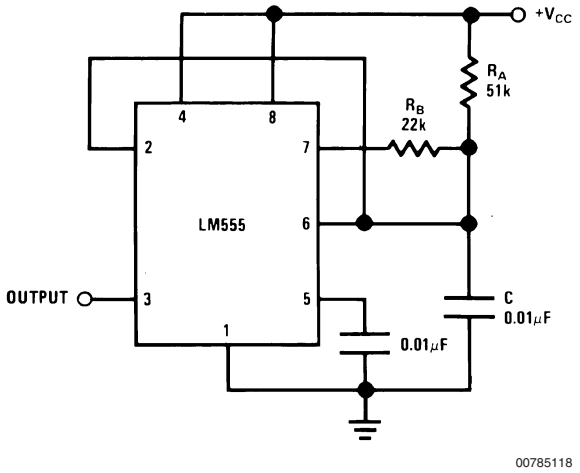
50% DUTY CYCLE OSCILLATOR

For a 50% duty cycle, the resistors R_A and R_B may be connected as in *Figure 14*. The time period for the output high is the same as previous, $t_1 = 0.693\ R_A\ C$. For the output low it is t_2 =

$$\left[(R_A\ R_B)/(R_A + R_B)\right] C\ \ell n \left[\frac{R_B - 2R_A}{2R_B - R_A}\right]$$

Thus the frequency of oscillation is

$$f = \frac{1}{t_1 + t_2}$$



00785118

FIGURE 14. 50% Duty Cycle Oscillator

Note that this circuit will not oscillate if R_B is greater than 1/2 R_A because the junction of R_A and R_B cannot bring pin 2 down to 1/3 V_{CC} and trigger the lower comparator.

ADDITIONAL INFORMATION

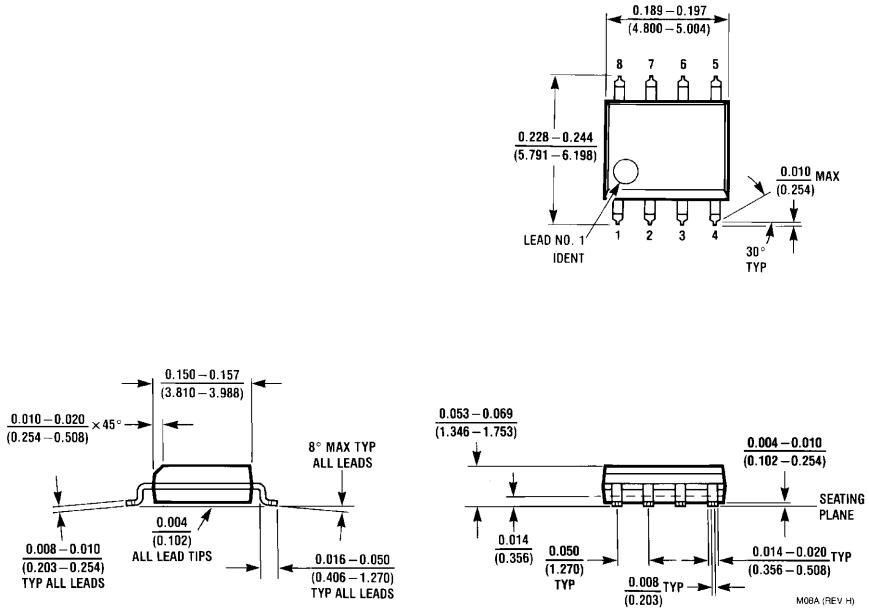
Adequate power supply bypassing is necessary to protect associated circuitry. Minimum recommended is 0.1µF in parallel with 1µF electrolytic.

Lower comparator storage time can be as long as 10µs when pin 2 is driven fully to ground for triggering. This limits the monostable pulse width to 10µs minimum.

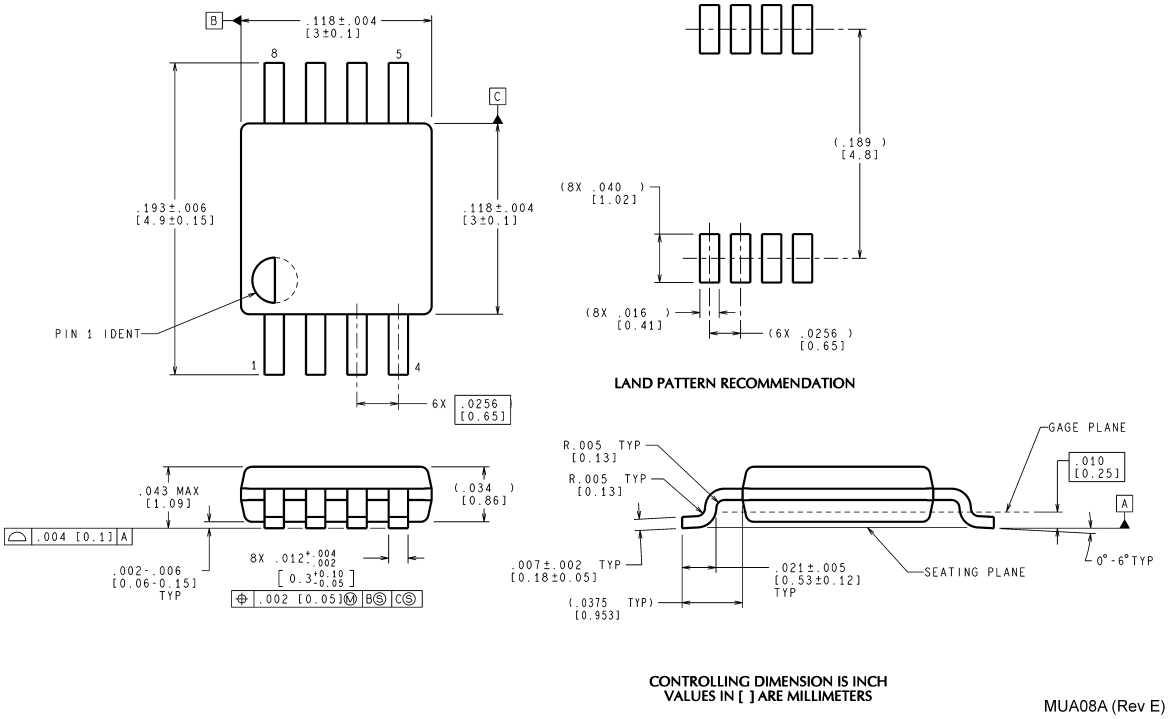
Delay time reset to output is 0.47µs typical. Minimum reset pulse width must be 0.3µs, typical.

Pin 7 current switches within 30ns of the output (pin 3) voltage.

Physical Dimensions inches (millimeters) unless otherwise noted

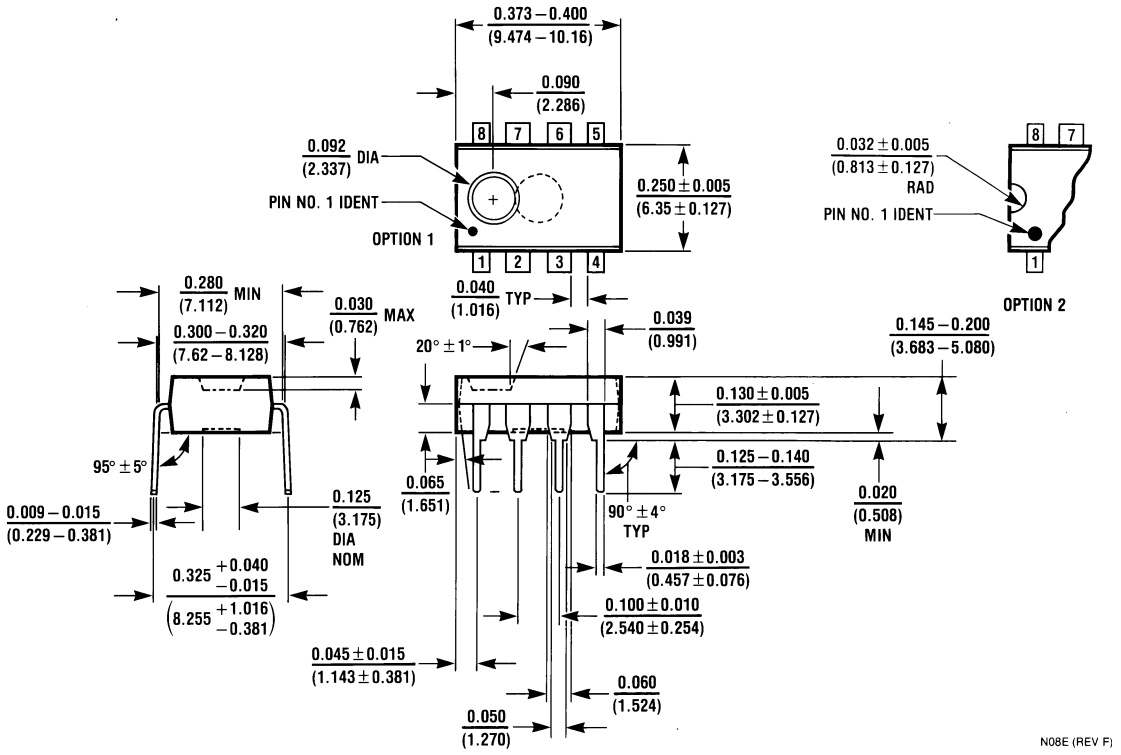


Small Outline Package (M)
NS Package Number M08A



8-Lead (0.118” Wide) Molded Mini Small Outline Package
NS Package Number MUA08A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



Molded Dual-In-Line Package (N)
NS Package Number N08E

National Semiconductor
Americas Customer
Support Center
Email: new.feedback@nsc.com
Tel: 1-800-272-9959

www.national.com

National Semiconductor
Europe Customer Support Center
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

National Semiconductor
Asia Pacific Customer
Support Center
Email: ap.support@nsc.com

National Semiconductor
Japan Customer Support Center
Fax: 81-3-5639-7507
Email: jpn.feedback@nsc.com
Tel: 81-3-5639-7560

National Semiconductor

June 1989

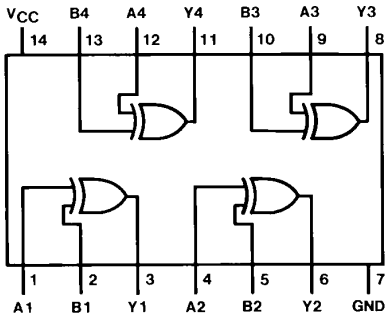
DM54LS86/DM74LS86
Quad 2-Input Exclusive-OR Gates

General Description

This device contains four independent gates each of which performs the logic exclusive-OR function.

Connection Diagram

Dual-In-Line Package



TL/F/6380–1

Order Number DM54LS86J, DM54LS86W, DM74LS86M or DM74LS86N
See NS Package Number J14A, M14A, N14A or W14B

Function Table

$$Y = A \oplus B = \overline{A}B + A\overline{B}$$

Inputs		Output
A	B	Y
L	L	L
L	H	H
H	L	H
H	H	L

H = High Logic Level
L = Low Logic Level

RRD-B30M105/Printed in U. S. A.

Absolute Maximum Ratings (Note)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	
DM54LS	−55°C to +125°C
DM74LS	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: *The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.*

Recommended Operating Conditions

Symbol	Parameter	DM54LS86			DM74LS86			Units
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			2			V
V_{IL}	Low Level Input Voltage			0.7			0.8	V
I_{OH}	High Level Output Current			−0.4			−0.4	mA
I_{OL}	Low Level Output Current			4			8	mA
T_A	Free Air Operating Temperature	−55		125	0		70	°C

Electrical Characteristics over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions		Min	Typ (Note 1)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max, V_{IH} = Min	DM54	2.5	3.4		V
			DM74	2.7	3.4		
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IL} = Max, V_{IH} = Min	DM54		0.25	0.4	V
			DM74		0.35	0.5	
		I_{OL} = 4 mA, V_{CC} = Min	DM74		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V				0.2	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V				40	μA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.6	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 2)	DM54	−20		−100	mA
			DM74	−20		−100	
I_{CCH}	Supply Current with Outputs High	V_{CC} = Max (Note 3)			6.1	10	mA
I_{CCL}	Supply Current with Outputs Low	V_{CC} = Max (Note 4)			9	15	mA

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, and the duration should not exceed one second.

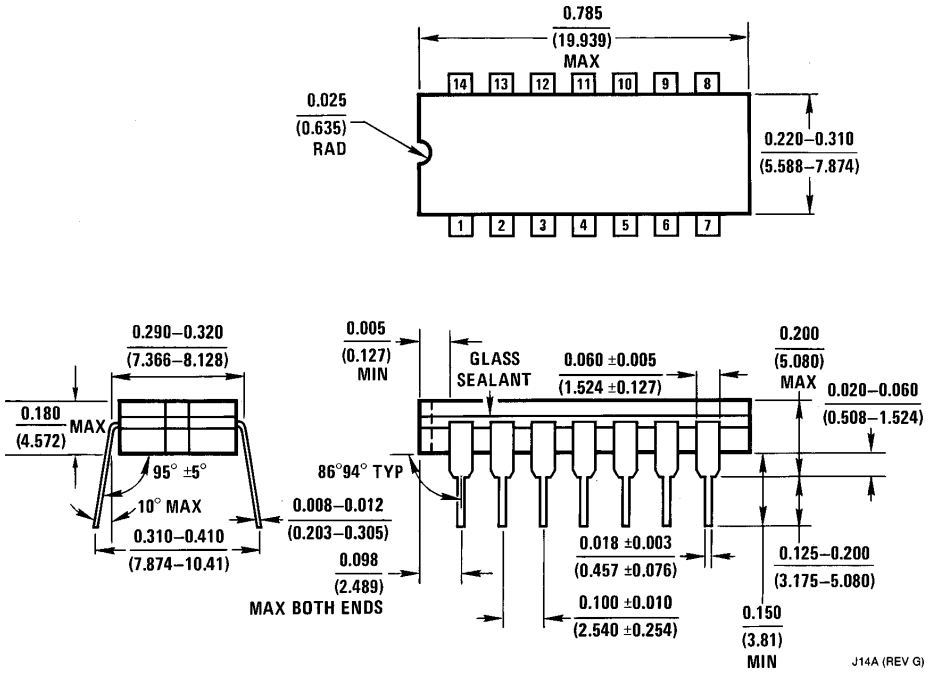
Note 3: I_{CCH} is measured with all outputs open, one input at each gate at 4.5V, and the other inputs grounded.

Note 4: I_{CCL} is measured with all outputs open and all inputs grounded.

Switching Characteristics at $V_{CC} = 5V$ and $T_A = 25°C$ (See Section 1 for Test Waveforms and Output Load)

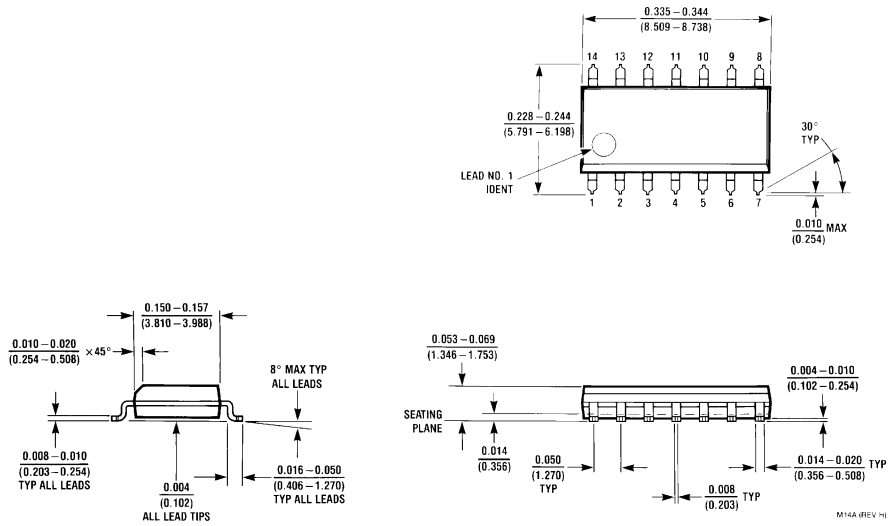
Symbol	Parameter	Conditions	$R_L = 2\ k\Omega$				Units
			$C_L = 15$ pF		$C_L = 50$ pF		
			Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time Low to High Level Output	Other Input Low		18		23	ns
t_{PHL}	Propagation Delay Time High to Low Level Output			17		21	ns
t_{PLH}	Propagation Delay Time Low to High Level Output	Other Input High		10		15	ns
t_{PHL}	Propagation Delay Time High to Low Level Output			12		15	ns

Physical Dimensions inches (millimeters)

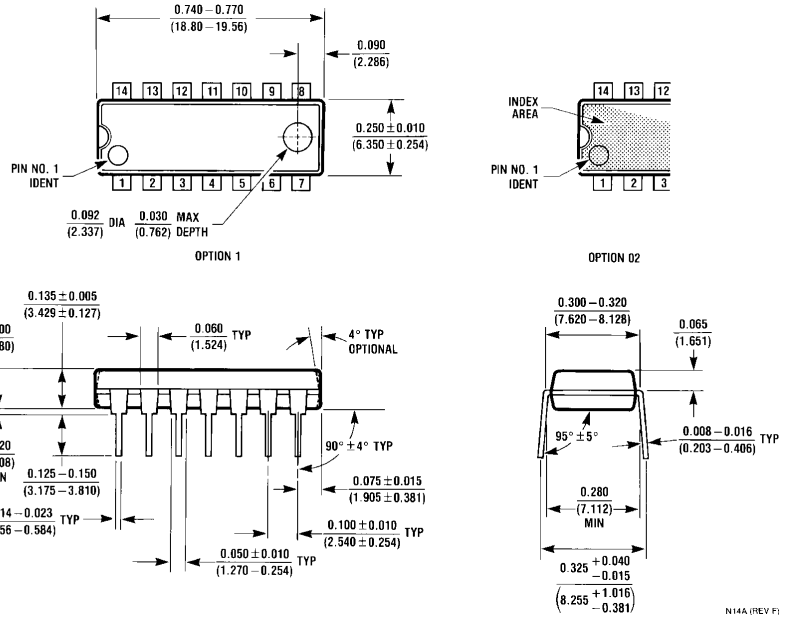


14-Lead Ceramic Dual-In-Line Package (J)
Order Number DM54LS86J
NS Package Number J14A

Physical Dimensions inches (millimeters) (Continued)

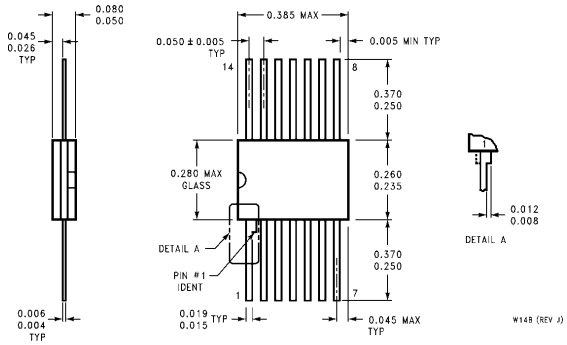


14-Lead Small Outline Molded Package (M)
Order Number DM74LS86M
NS Package Number M14A



14-Lead Molded Dual-In-Line Package (N)
Order Number DM74LS86N
NS Package Number N14A

Physical Dimensions inches (millimeters) (Continued)



14-Lead Ceramic Flat Package (W)
Order Number DM74LS86W
NS Package Number W14B

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

National Semiconductor

June 1989

54LS00/DM54LS00/DM74LS00 Quad 2-Input NAND Gates

General Description

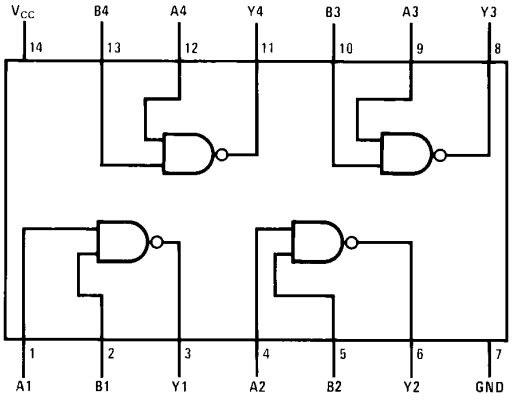
This device contains four independent gates each of which performs the logic NAND function.

Features

- Alternate Military/Aerospace device (54LS00) is available. Contact a National Semiconductor Sales Office/ Distributor for specifications.

Connection Diagram

Dual-In-Line Package



TL/F/6439–1

Order Number 54LS00DMQB, 54LS00FMQB, 54LS00LMQB, DM54LS00J, DM54LS00W, DM74LS00M or DM74LS00N
See NS Package Number E20A, J14A, M14A, N14A or W14B

Function Table

$Y = \overline{AB}$

Inputs		Output
A	B	Y
L	L	H
L	H	H
H	L	H
H	H	L

H = High Logic Level
L = Low Logic Level

RRD-B30M105/Printed in U. S. A.

Absolute Maximum Ratings (Note)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	7V
Input Voltage	7V
Operating Free Air Temperature Range	
DM54LS and 54LS	−55°C to +125°C
DM74LS	0°C to +70°C
Storage Temperature Range	−65°C to +150°C

Note: *The "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. The device should not be operated at these limits. The parametric values defined in the "Electrical Characteristics" table are not guaranteed at the absolute maximum ratings. The "Recommended Operating Conditions" table will define the conditions for actual device operation.*

Recommended Operating Conditions

Symbol	Parameter	DM54LS00			DM74LS00			Units
		Min	Nom	Max	Min	Nom	Max	
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High Level Input Voltage	2			2			V
V_{IL}	Low Level Input Voltage			0.7			0.8	V
I_{OH}	High Level Output Current			−0.4			−0.4	mA
I_{OL}	Low Level Output Current			4			8	mA
T_A	Free Air Operating Temperature	−55		125	0		70	°C

Electrical Characteristics over recommended operating free air temperature range (unless otherwise noted)

Symbol	Parameter	Conditions		Min	Typ (Note 1)	Max	Units
V_I	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA				−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, I_{OH} = Max, V_{IL} = Max	DM54	2.5	3.4		V
			DM74	2.7	3.4		
V_{OL}	Low Level Output Voltage	V_{CC} = Min, I_{OL} = Max, V_{IH} = Min	DM54		0.25	0.4	V
			DM74		0.35	0.5	
		I_{OL} = 4 mA, V_{CC} = Min	DM74		0.25	0.4	
I_I	Input Current @ Max Input Voltage	V_{CC} = Max, V_I = 7V				0.1	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.7V				20	μA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V				−0.36	mA
I_{OS}	Short Circuit Output Current	V_{CC} = Max (Note 2)	DM54	−20		−100	mA
			DM74	−20		−100	
I_{CCH}	Supply Current with Outputs High	V_{CC} = Max			0.8	1.6	mA
I_{CCL}	Supply Current with Outputs Low	V_{CC} = Max			2.4	4.4	mA

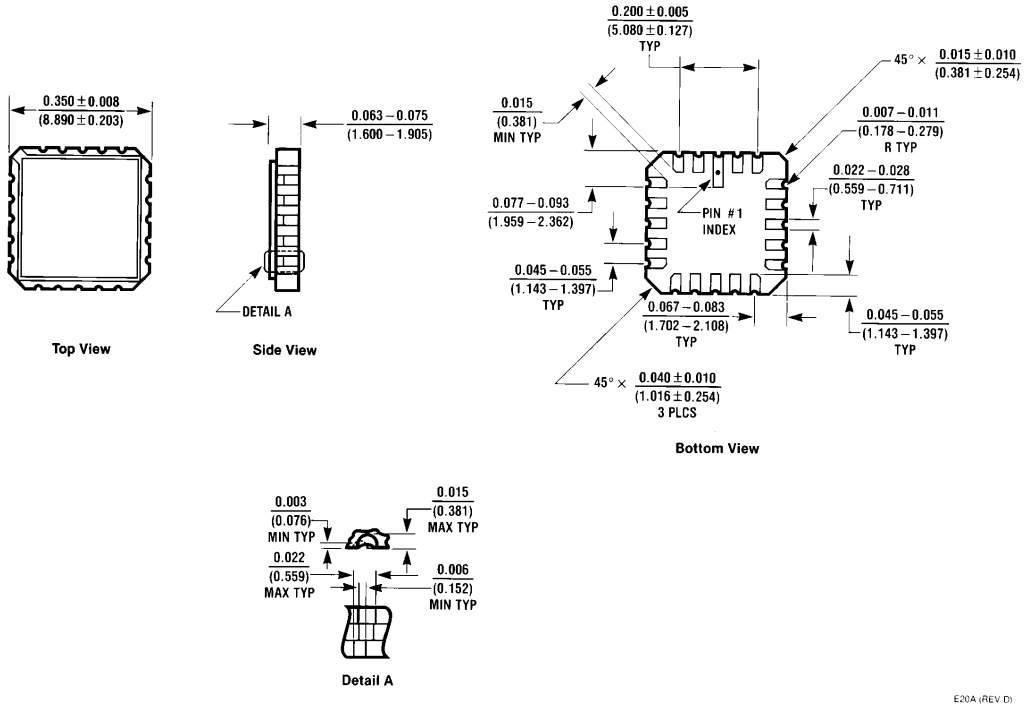
Switching Characteristics at V_{CC} = 5V and T_A = 25°C (See Section 1 for Test Waveforms and Output Load)

Symbol	Parameter	R_L = 2 kΩ				Units
		C_L = 15 pF		C_L = 50 pF		
		Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time Low to High Level Output	3	10	4	15	ns
t_{PHL}	Propagation Delay Time High to Low Level Output	3	10	4	15	ns

Note 1: All typicals are at V_{CC} = 5V, T_A = 25°C.

Note 2: Not more than one output should be shorted at a time, and the duration should not exceed one second.

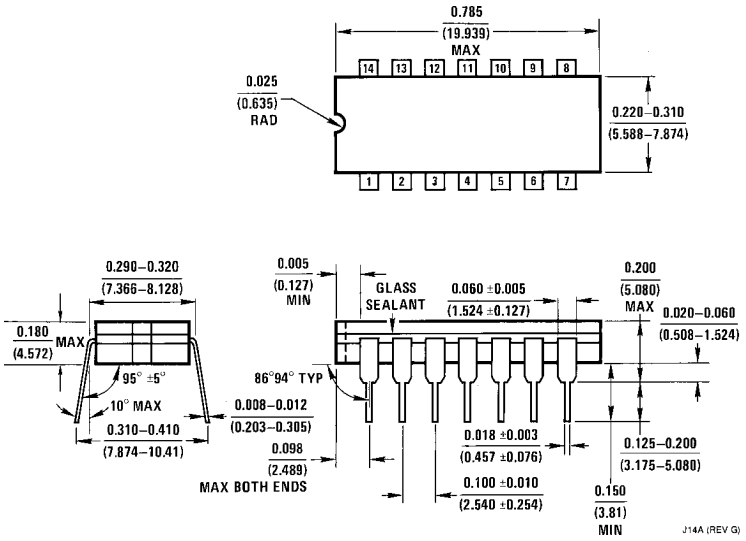
Physical Dimensions inches (millimeters)



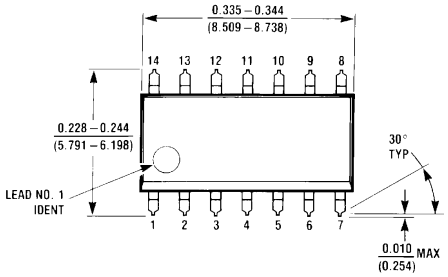
E20A (REV D)

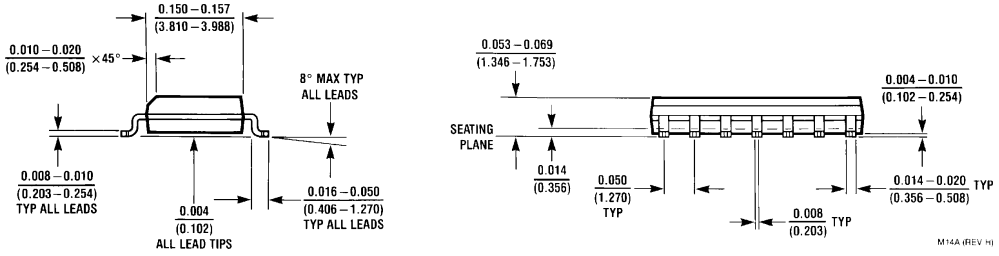
Ceramic Leadless Chip Carrier Package (E)
Order Number 54LS00LMQB
NS Package Number E20A

Physical Dimensions inches (millimeters)



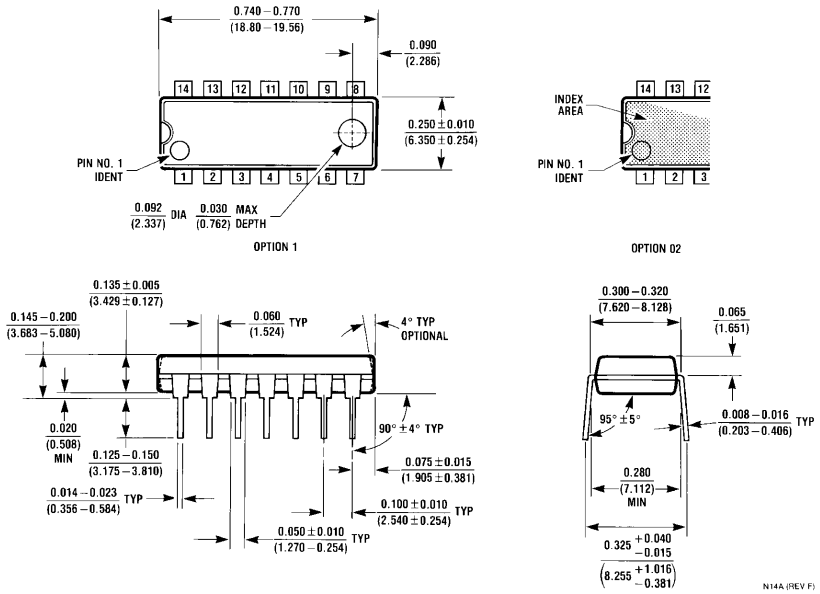
14-Lead Ceramic Dual-In-Line Package (J)
Order Number 54LS00DMQB or DM54LS00J
NS Package Number J14A





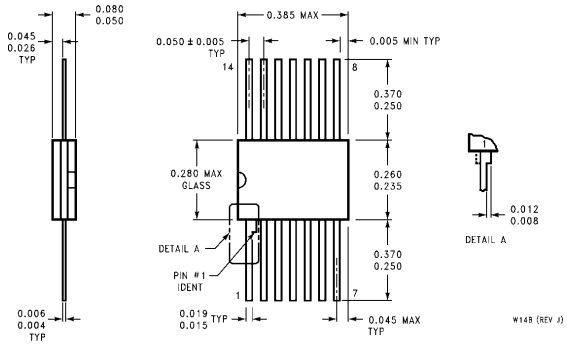
14-Lead Small Outline Molded Package (M)
Order Number DM74LS00M
NS Package Number M14A

Physical Dimensions inches (millimeters) (Continued)



14-Lead Molded Dual-In-Line Package (N)
Order Number DM74LS00N
NS Package Number N14A

Physical Dimensions inches (millimeters) (Continued)



14-Lead Ceramic Flat Package (W)
Order Number 54LS00FMQB or DM54LS00W
NS Package Number W14B

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408

IX. Bibliografía

- Documentación de GPIB (Archivo de ayuda gpib32.hlp)
- Documentación de PC-TIO 10 (Manuales)
- Lab-PC+ User Manual, Low-Cost Multifunction I/O Board for ISA
  National Instruments Corporation, 1996
- NI-DAQ ™ User Manual for PC Compatibles
  National Instruments Corporation, 2001
- DAQ Hardware Overview Guide
  National Instruments Corporation, 2000
- Documentación del sintetizador HP8904A
- Documentación del estimulador vibro-táctil
- Sphar, Chuck
  Aprenda Visual C++ 6.0 Ya
  Microsoft Press, 1999
- Mano, M. Morris
  Diseño Digital
  Prentice-Hall Hispanoamericana, S. A., 1987
- Application Note 086, Using DAQ Event Messaging under Windows NT/95/3.1 (AN086.pdf)
- Sadahiro, Ken
  National Instruments Corporation, May 1997
- Mikula, Alexander
  Reading Excel files using ODBC
  http://www.codeproject.com/database/excel_odbc.asp
- Press, William H. [et al.]
  Numerical recipes in C: the art of scientific computing
  Cambridge University Press, 2nd Edition