Contenido

Editorial

Cuando se oye hablar de estadísticas acerca de la investigación que se lleva a cabo en cada uno de los países del mundo, haciendo referencia a las disciplinas científicas y tecnológicas y considerando desde luego el número de investigadores participantes, se observa que estas cifras son manejadas principalmente con fines políticos. Esto muchas veces no tiene mayores alcances puesto que la mayoría de las personas que las aplican no se encuentran inmersas en los medios científicos, y por lo tanto su objetividad es dudosa.

Lo cierto es que si se comparan los niveles de investigación científica y tecnológica de los países con mejores condiciones económicas y sociales, con los de México, se aprecia que existe aún un largo camino por recorrer no sólo para disminuir la dependencia tecnológica, sino también para equipararse con los países del "primer mundo", por lo cual hay que tener muy presente que de alguna manera la dependencia tecnológica de un país refleja su falta de investigadores, calidad de la investigación y desde luego la falta de unificación a nivel nacional de criterios acerca de la investigación.

Así pues, hay que hacer frente al hecho de que en la actualidad las fronteras mexicanas, económicamente hablando, se sitúan hasta donde nuestra tecnología es capaz de sostener al país, es decir, la importación de la tecnología será aún más grande si no se establecen diferentes políticas y métodos para aumentar tanto los Centros de investigación nacional, como la calidad de ésta última. Algunas políticas a instrumentar pueden ser: el contar con un eficaz sistema nacional de intercambio científico; la disminución de políticas burocráticas en el otorgamiento de becas para la realización de estudios de posgrado, tanto a nivel nacional como internacional y en la asignación de créditos para la investigación; el aumento de la calidad de enseñanza a nivel superior, buscar convenios reales con la industria; así como reforzar el intercambio científico y cultural con otros países como Japón, Francia, Alemania, todo esto con la finalidad de que en un mediano plazo, México pueda dejar de ser un país maquilador y formador de técnicos.

Ahora bien, parecería repetitivo el hablar de este tema, pero es conveniente tomar muy en cuenta lo anterior en la elaboración de los planes de desarrollo económico, social y educativos principalmente, para el futuro inmediato, que permitan despertar un verdadero interés por la investigación. El Cintec es consciente de esto, y ante todo busca su superación no sólo desde la calidad de la educación que proporciona, sino también de los niveles correspondientes a su planta docente e infraestructura.

Desarrollo de Programas Residentes en base a un TSR genérico

Ing. Eduardo Vega Alvarado
Jefe del Departamento de Laboratorios Ligeros del CINTEC-IPN

El presente artículo inicia una serie de trabajos sobre tópicos relacionados con programación, incluyendo tanto al desarrollo de algoritmos o rutinas específicas como la interacción de estos programas con el sistema operativo.

Introducción

Cuando se desea ejecutar un programa en el entorno del sistema operativo **MS-DOS** (**M**icro**S**oft **D**isk **O**perating **S**ystem), generalmente el cargador de programas del sistema busca espacio libre en la memoria para colocar el código solicitado, transfiere control al mismo y al terminar el procesamiento libera el área de memoria asignada a dicho código **[3]**; sin embargo, existe cierta clase de programas cuya ejecución no incluye esta última etapa. En esencia, todo programa que permanece cargado en memoria después de que termina su ejecución (incluso aún cuando se inicie la ejecución de otro programa), se denomina **TSR** (**T**erminate and **S**tay **R**esident, termina y permanece residente) **[4]**.

TSR Genérico

El principal problema a enfrentar cuando se escribe un programa TSR es el proporcionar al usuario la forma de activarlo en el momento que lo desee; esto es obvio, dado que aunque al código originalmente en memoria se le adicione una sección extra, esta porción no se ejecutará si no existe forma de invocarla. No basta únicamente con asignar un área permanente en memoria para la colocación del TSR, sino que es necesario además asegurar algún medio para transferirle control. En forma genérica **[2] [4]**, un TSR consta de tres partes principales, a saber:

1. El programa o rutinas que se desean ejecutar.

2. El conjunto de instrucciones que coloca al TSR en memoria e instruye al sistema operativo para que lo considere residente (Inicialización).

3. El código que implementa la vía de acceso o de ejecución al TSR.

La distribución usual de estas partes en un programa real se muestra en la **figura 1.**

A continuación se desarrolla un programa ejemplo, con el cual se explicará el proceso de generación de un TSR, indicando además qué secciones del mismo corresponden al modelo genérico y cuáles son propias de este caso específico. La ejecución del programa genera un

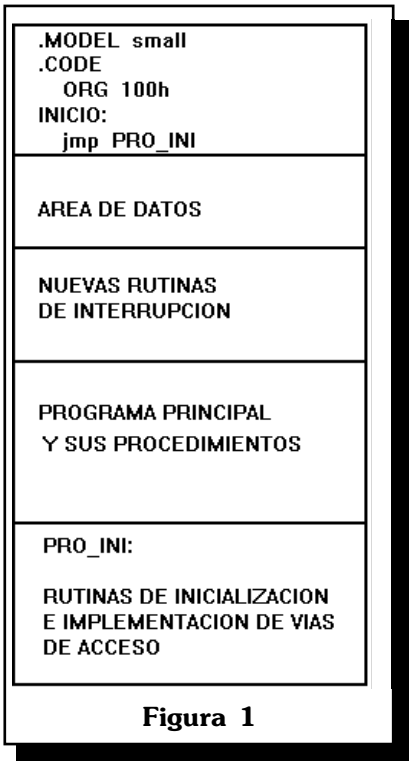


Figura 1

reloj, el cual aparece en la parte superior derecha de la pantalla desplegándose en forma digital; el listado se muestra en la anexo 1. Este programa ejemplo fue probado compilándolo con **Macro Assembler** ᴿ **[1]**, **MASM**, versión 5.1 de **Microsoft, Inc**., aunque puede también manejarse sin cambios con **Turbo Assembler** ᴿ, **TASM**, de **Borland International, Inc**.

Zona de Datos del Programa

Normalmente, los TSR comienzan con una instrucción de salto

```
1   ; TSR que Genera un Reloj en Pantalla
2   ; Programó: Ing. Eduardo Vega Alvarado
3   ;-------------------------------------------------------
4   .model   small
5   .code
6       org    100h
7
8   Inicio:
9       jmp    Instalar
10
11  ; Area para Datos del TSR
12  ;-------------------------
13  ; Vectores de interrupción originales
14      i1C             label   dword       ;int 1Ch
15      off_i1C     dw      ?
16      seg_i1C         dw      ?
17      i10             label   dword       ;int 10h
18      off_i10         dw      ?
19      seg_i10         dw      ?
20      i28             label   dword       ;int 28h
21      off_i28         dw      ?
22      seg_i28         dw ?
23
24      vid_act     db      0       ;bandera de int 10h activa
25      activo          db      0       ;tsr en ejecución
26      off_indos       dw      ?       ;segmento y offset de la bandera
27      seg_indos       dw      ?       ;de dos activo (indos)
28      err_act         dw      ?       ;offset de la bandera de error crítico
29      cursor1         dw      ?       ;posición y tamaño
30      cursor2         dw      ?       ;del cursor
31      sp_sist     dw      ?       ;sp y ss originales
32      ss_sist         dw      ?
33                      db  255 dup(?);pila propia
34      tsr_pil         db      ?       ;inicio de pila propia
35      horas10         db      ?       ;cadena para desplegar la hora
36      horas           db      ?
37                      db      ":"
38      mins10          db      ?
39      mins            db      ?
40                      db      ":"
41      segs10          db      ?
42      segs            db      ?
43
44  ; Rutina modificada para la int 1Ch
45  ;-----------------------------------
46  Sust1C      proc    near
47      mov         activo,18
48      pushf                       ;simula llamado a INT
49      call        i1C             ;invoca int 1Ch original
50      push        di
51      push        es
52
53  ; Checa si está activa int 10h, la bandera indos o se
54  ; atiende un error crítico. En caso afirmativo finaliza
55  ; rutina int 1Ch, en caso contrario ejecuta Reloj
56      cmp         vid_act,0       ;¿int 10h activa?
57      jne         Fin1C
```

hacia la zona de inicialización del programa, reservando en la parte inicial del segmento de código una zona para manejo de las estructuras de datos requeridas por el programa. En el programa ejemplo esta zona de almacenamiento abarca las líneas 11 a 42.

Inicialización del TSR y Acceso al Programa Residente

La sección correspondiente a la colocación en memoria del programa y a los cambios necesarios para instrumentar una vía de acceso al mismo se sitúa como la parte final del código, dado que su ejecución se realiza solo una vez; de esta forma, en el momento en que la rutina queda instalada esta área se descarta. En el listado anexo, la inicialización y colocación ocupa las líneas de código 222 a 275; como puede observarse, primeramente se interceptan o cambian los vectores de las interrupciones que sirvan como vía de acceso a la rutina residente **[2]**, o cuyo funcionamiento pudiera ser afectado por la misma (servicio 25h, int 21h). También se conservan las direcciones originales de las interrupciones empleadas (servicio 35h, int 21h), con el fin de permitir la invocación de los servicios correspondientes a esas rutinas, así como la desactivación del TSR en un momento dado.

Para el caso del ejemplo se manejan como vías de acceso las interrupciones 1Ch y 28h **[1] [2]**; ambas rutinas se producen automáticamente 18.2 veces por segundo, siendo generada la primera por la máquina y la segunda por el planificador **(scheduler)** del sistema operativo. Así mismo, se modifica el vector de la int 10h (manejo de video); esto debido a que la ejecución del programa en forma

```
58      mov     di,off_indos        ;¿bandera indos activa?
59      mov     es,seg_indos
60      cmp     byte ptr es:[di],0
61      jne     Fin1C
62      mov     di,err_act          ;¿atención a error crítico?
63      cmp     byte ptr es:[di],0
64      jne     Fin1C
65      pop     es
66      pop     di
67      call    Reloj               ;ejecuta la rutina principal del TSR
68      iret
69  Fin1C:
70      pop     es
71      pop     di
72      dec     activo
73      iret                        ;return to MS-DOS
74  Sust1C      endp
75
76  ; Rutina modificada para la int 10h
77  ;-------------------------------------
78  Sust10      proc    near
79      pushf                       ;simula llamado a INT
80      inc     vid_act
81      call    i10                 ;invoca int 10h original
82      dec     vid_act
83      iret
84  Sust10      endp
85
86  ; Rutina modificada para la int 28h
87  ;-------------------------------------
88  Sust28      proc    near
89      pushf                       ;simula llamado a INT
90      call    i28                 ;invoca int 28h original
91      cmp     activo,0
92      je      Fin28
93
94  ; Checa si está activa int 10h o se atiende un error
95  ; crítico. En caso afirmativo finaliza rutina int 28h
96  ; en caso contrario ejecuta Reloj
97      cmp     vid_act,0           ;int 10h activa?
98      jne     Fin28
99      push    di
100     push    es
101     mov     es,seg_indos        ;atención a error crítico?
102     mov     di,err_act
103     cmp     byte ptr es:[di],0
104     pop     es
105     pop     di
106     jne     Fin28
107     mov     activo,0
108     call    Reloj               ;ejecuta la rutina principal
109 Fin28:
110     iret
111 Sust28      endp
112
113 ; Rutina principal del TSR. Realiza el llamado a la lectura
114 ; de la hora y efectúa el despliegue de la misma
```

concurrente con la activación de la int 10h puede generar basura en el despliegue. Cabe mencionar que existe una interrupción, la 8h, que también se genera 18.2 veces por segundo; la diferencia entre esta interrupción y la 1Ch radica en las rutinas de servicio asociadas a cada una de ellas. La interrupción 8h maneja un determinado código para actualizar la hora del sistema (almacenada en el área de bios), mientras que la única instrucción de la interrupción 1Ch es el retorno de la misma, **iret**. Por esta razón se recomienda interceptar para aplicaciones de usuario la int 1Ch, evitando así el provocar disturbios en el manejo de la hora interna del sistema.

Posteriormente se leen los apuntadores a **indos** e **indos2** (servicios 34 y 5d-06, int 21h) **[3]**; estas banderas son manejadas por el sistema operativo para indicar si se está procesando una llamada a MS-DOS o si se ejecuta una rutina de atención por error crítico, respectivamente. Las banderas indos e indos2, al igual que los servicios de la int 21h necesarios para obtenerlas forman parte del llamado MS-DOS no documentado **[3]**; se ha denominado así a un conjunto de interrupciones, servicios y estructuras de datos que, pese a estar implementadas en el sistema, no han sido aceptadas como existentes por parte de Microsoft. Si bien su empleo es a riesgo del programador, su aplicación es de dominio casi público entre los diseñadores de software, especialmente en los casos de programas TSR o cuando se requiere un manejo especializado de dispositivos periféricos, siendo utilizadas inclusive en los programas mismos de Microsoft, tales como Windows ® y Excel ®.

Por último, con auxilio de la etiqueta **final** se determina el nú-

```
115 ;-------------------------------------------------------------------
116 Reloj       proc     near
117 ;establece el stack local y salva registros del DOS
118     cli
119     mov      sp_sist,sp          ;guarda apuntadores a
120     mov      ss_sist,ss          ;la pila del sistema y
121     push     cs                  ;establece pila propia
122     pop      ss
123     mov      sp,offset tsr_pil
124     sti
125     push     ax                  ;salva registros
126     push     bx                  ;del sistema
127     push     cx
128     push     dx
129     push     si
130     push     di
131     push     ds
132     push     es
133     push     bp
134     mov      ah,0fh              ;¿trabajando en modo gráfico?
135     int      10h
136     cmp      al,3
137     jbe      Pos_cursor
138     cmp      al,7
139     je       Pos_cursor
140 Termino:
141     pop      bp                  ;modo gráfico activo, por
142     pop      es                  ;lo que termina rutina
143     pop      ds
144     pop      di
145     pop      si
146     pop      dx
147     pop      cx
148     pop      bx
149     pop      ax
150     cli
151     mov      ss,ss_sist
152     mov      sp,sp_sist
153     sti
154     ret
155 Pos_cursor:                      ;modo texto, continúa despliegue
156     mov      ah,03               ;lee posición original del cursor
157     int      10h                 ;y la almacena en área de datos
158     mov      cursor1,dx
159     mov      cursor2,cx
160     call     Tiempo              ;lee hora del sistema
161     push     cs
162     pop      ds
163     mov      ah,016              ;cambia atributo de cursor
164     mov      cx,1000h
165     int      10h
166     mov      ah,02               ;coloca cursor en la esquina
167     mov      dh,0                ;superior derecha de la pantalla,
168     mov      dl,70               ;en (0,70)
169     int      10h
170     mov      si,offset horas10   ;apuntador a cadena
171     mov      cx,8                ;para desplegar
```

mero de párrafos que conforman al TSR, dejándose a éste permanente en memoria (servicio 31, int 21h). Como ya se había mencionado antes, en el código residente no se incluye el área de inicialización.

Declaración y Manejo de las Nuevas Rutinas de Interrupción

Uno de los principales detalles que debe respetar la nueva rutina de interrupción es lo que se conoce como encadenamiento **(chaining) [2] [4]**. El encadenamiento consiste en que el procedimiento nuevo invoque a la interrupción original, ya que puede darse el caso de que se hubieran montado anteriormente otros programas residentes que utilicen dicha interrupción. Dado que el llamado de la rutina antigua es por medio de **call [1]**, es necesario eje-cutar primero la instrucción **pushf** para simular una invocación por interrupción y compensar así el regreso de la rutina, el cual está marcado por **iret**. En el listado anexo, las nuevas rutinas de interrupción ocupan las líneas 44 a 111.

Rutina Principal

Se denomina rutina principal al código que genera la acción que se desea realice el TSR (por ejemplo, desplegar la hora en pantalla, como en el listado anexo). Independientemente de su función, esta rutina debe cumplir algunas reglas no escritas, entre las que destaca el conservar los registros en su estado inicial, almacenándolos en la pila **(stack)**. Lo anterior obedece a que en esos registros se conserva parte del estado del programa que se suspendió para dar paso al TSR; en la medida de lo posible es conveniente implementar una zona de

```
Desplegar:
    lodsb                       ;muestra cadena en pantalla,
    mov     ah,0ah              ;escribiendo un carácter a la vez
    push    cx
    mov     cx,1
    int     10h
    pop     cx
    inc     dl
    mov     ah,02               ;avanza cursor una posición
    int     10h
    loop    Desplegar
    mov     ah,18               ;regresa cursor a posición original
    mov     cx,cursor2          ;y restablece registros del sistema
    int     10h
    mov     ah,2
    mov     dx,cursor1
    int     10h
    jmp     Termino
Reloj   endp

; Rutina para leer la hora del sistema, convertirla
; a código ASCII y almacenarla en el área de datos
; del TSR
;-------------------------------------------------
Tiempo  proc    near
    mov     ah,2ch
    int     21h                 ;lectura de la hora
    mov     bl,10
    xor     ah,ah
    mov     al,ch               ;horas en ch
    div     bl
    or      ax,3030h
    mov     horas10,al
    mov     horas,ah
    xor     ah,ah
    mov     al,cl               ;minutos en cl
    div     bl
    or      ax,3030h
    mov     mins10,al
    mov     mins,ah
    xor     ah,ah
    mov     al,dh               ;segundos en dh
    div     bl
    or      ax,3030h
    mov     segs10,al
    mov     segs,ah
    ret
Tiempo  endp
final   db  "$"     ;marca para fin de TSR

; Rutina de instalación en memoria del TSR. También
; modifica los vectores de interrupción utilizados
; o afectados por el TSR, para apuntar a los nuevos
; manejadores.
;-------------------------------------------------
Implementa  proc    near
assume  ds:@code        ;variables en este segmento
```

pila local en el área de datos del TSR, para evitar conflictos tales como desbordamiento si se emplea el stack convencional del sistema. En consecuencia, previo a la terminación de la rutina principal se deben restaurar los registros, siguiendo la operación inversa.

La rutina principal, al igual que las nuevas rutinas de interrupción, deben respetar la no reentrancia **(reentry) [2]** del sistema operativo. La no reentrancia consiste en que no se puede efectuar una llamada al sistema en tanto no finalice la llamada anterior, ya que de lo contrario se pueden generar fallas ocasionadas por el frágil manejo de MS-DOS para preservar su estado actual al momento de una interrupción. Este problema se maneja en el programa ejemplo mediante el uso de la bandera indos para detectar si existe una llamada al sistema activa y no ejecutar el TSR en tal caso. Por esta misma razón para el despliegue de la cadena generada con la hora se utiliza la int 10h servicio 0ah, que muestra un carácter a la vez, para no emplear la int 21h servicio 9h, que permite desplegar la cadena completa.

Como consecuencia de la no reentrancia del sistema operativo, se debe evitar activar el TSR cuando se presenta una condición de error crítico; estos errores se producen cuando el DOS necesita utilizar un dispositivo periférico pero este no se encuentra listo o disponible, activándose entonces la int 24h **[3]**, conocida como manejador de errores críticos. En el listado anexo se comprueba esta condición por medio de la bandera indos2, permitiéndose o no la ejecución del TSR, dependiendo del valor de esa bandera.

La rutina principal del programa ejemplo se explica línea por

```
229
230 Instalar:
231     mov     ah,34h              ;localiza bandera indos
232     int     21h
233     mov     off_indos,bx        ;offset y segmento
234     mov     seg_indos,es        ;de indos
235     mov     ah,5dh              ;localiza bandera de atención
236     mov     al,6h               ;a error crítico y copia apuntador
237     int     21h                 ;en el área de datos
238     push    cs
239     pop     ds
240     mov     err_act,si
241
242  ; Sustitución de vectores de interrupción afectados,
243  ; conservando los apuntadores originales para efectos
244  ; de restauración por terminación del TSR
245     mov     ax,351Ch            ; int 1Ch
246     int     21h
247     mov     off_i1C,bx
248     mov     seg_i1C,es
249     mov     ax,251Ch
250     mov     dx,offset Sust1C
251     int     21h
252     mov     ax,3510h            ; int 10h
253     int     21h
254     mov     off_i10,bx
255     mov     seg_i10,es
256     mov     ax,2510h
257     mov     dx, offset Sust10
258     int     21h
259     mov     ax,3528h            ; int 28h
260     int     21h
261     mov     off_i28,bx
262     mov     seg_i28,es
263     mov     ax,2528h
264     mov     dx,offset Sust28
265     int     21h
266
267  ; Cálculo de longitud del TSR y asignación de
268  ; espacio permanente en memoria
269     mov     dx,(offset final-@code+15)    ; No. de bytes
270     mov     cl,4h
271     shr     dx,cl                         ; No. de párrafos
272     mov     ax,3100h
273     int     21h
274  Implementa  endp
275     end Inicio
```

línea en el listado anexo (líneas 113 a 218); no se hace mayor hincapié en la misma puesto que puede ser sustituida en el TSR genérico descrito por cualquier otro procedimiento que se desee.

Consideraciones Finales

Dentro de las consideraciones finales se encuentran las correspondientes a la desactivación del TSR. La desactivación incluye dos tareas principales, la primera es interceptar los vectores de las interrupciones ocupadas de tal forma que tomen los apuntadores originales (los cuales previamente se guardaron en el área de datos del TSR) ocupando para ello la int 21h servicio 25h. La segunda tarea presenta mayor complejidad, y consiste en liberar el área de memoria ocupada por el TSR, después de su desactivación; debido a esta complejidad implícita, se considera conveniente manejar este tópico en un artículo posterior.

Finalmente, debe recordarse que si bien existen características del sistema no documentadas cuya utilidad es manifiesta, su uso se debe restringir a aquellas situaciones en las que constituyan la única opción dado que, por ser información no oficial, no se puede asegurar su no modificación o incluso su conservación en versiones posteriores del sistema operativo MS-DOS, lo que puede ocasionar problemas futuros de compatibilidad.

Bibliografía

[1] Barkakati, Nabajyoti & Hyde, Randall. *"Microsoft Macro Assembler Bible"*. SAMS. 1992.

[2] Holzner, Steven. *"Advanced Assembly Language"*. Peter Norton Computing, Inc. 1991.

[3] Schulman et Al. *"Undocumented DOS"*. Addison Wesley Publishing Company, Inc. 1990.

[4] Simrin, Steven. *"MS-DOS Bible"*. Howard W. Sams & Company. 1989.

Aplicaciones del procesamiento gráfico y de imágenes: La realidad virtual. (Introducción)

M. en C. Héctor S. García Salas
Profesor e Investigador del CINTEC-IPN

Con esta publicación se desea involucrar al lector en uno de los últimos campos de aplicación del procesamiento gráfico y de imágenes, siendo éste el primero de una serie de artículos en donde se pretende presentar algunos de los diferentes aspectos técnicos y problemas que aparecen a lo largo de estos procesos.

Glosario

Objeto sintético.- Conjunto de datos calculados, cuya estructura permite obtener una representación gráfica con una forma y una textura. Estos objetos pueden ser una representación de algo real o imaginario.

Objeto virtual.- Objeto sintético, que ocupa un lugar en un mundo virtual (posición tridimensional, etc.).

Mundo virtual.- Escenario gráfico donde se encuentran los objetos sintéticos. En este ambiente, se llevan a cabo las interacciones entre los objetos, desarrollando así una secuencia de escenas que dan vida a la realidad virtual.

Antecedentes

El procesamiento gráfico y de imágenes representa hoy en día una de las aplicaciones más importantes en el campo computacional. Las nuevas tecnologías, como por ejemplo: arquitecturas adecuadas para el tratamiento de datos, semiconductores con tiempos de respuesta más cortas, conductores con resistencias menos importantes al paso de los electrones, mayores escalas de integración en los C.I., dispositivos de memoria con mayor capacidad de almacenamiento, nuevas técnicas de procesamiento de datos, etc., han permitido que los procesos para tratar imágenes disminuyan los tiempos de ejecución. Todo esto ha tenido como resultado el desarrollo de poderosos programas de aplicación en esta área, conduciendo así a una explosión en cuanto a las aplicaciones del tratamiento gráfico y de imágenes, en una serie de campos tan diversos como sorprendentes.

Ahora bien, los avances de estas técnicas se han llevado a cabo de una manera muy compleja. Las necesidades económicas han sido, desde luego, las que han influido (y seguirán haciéndolo) de manera determinante en la dirección que han tomado los desarrollos de los campos de investigación.

Así, una de las aplicaciones que más ha contribuido al desarrollo de estas técnicas ha sido el campo cinematográfico. Las necesidades de este campo; en la producción y tratamiento de imágenes y gráficos han permitido que se dediquen grandes presupuestos a la investigación en el área, y los resultados y técnicas de estos procedimientos, se han difundido a otras disciplinas. Así, la realidad virtual ha sido una de las consecuencias de este tipo de avances.

El término de realidad virtual puede parecer incongruente; sin embargo, la referencia es aplicada debido a la interacción de objetos reales con objetos sintéticos. Los nuevos dispositivos electrónicos y las técnicas del procesamiento gráfico y de imágenes permiten crear objetos gráficos sintéticos en un escenario gráfico de tres dimensiones, equivalentes a los objetos reales. Las acciones efectuadas por el objeto real son transmitidas a los objetos sintéticos modificando de esta manera el escenario gráfico, produciendo así una realidad virtual (representación matemática de un universo espacio-temporal).

Así, las interacciones se llevan a cabo entre los objetos virtuales que representan a los objetos reales, con objetos virtuales imaginarios o

con otros objetos virtuales interpretados, es decir, que si existen físicamente pero que se encuentran en otro emplazamiento.

En la actualidad los "mundos virtuales" son poco detallados debido a la complejidad de los objetos y a los problemas de cálculo de las imágenes de los objetos virtuales. No obstante, éstos permiten ya una sensibilidad (percepción de los objetos en tres dimensiones y en perspectiva así como la modificación en tiempo real del escenario), de tal manera que el ejecutante puede introducirse en ellos e interactuar con los otros objetos de estos mundos, es decir, manipularlos o bien llevar a cabo acciones que no podrían hacerse en el mundo real. Estos son de hecho algunos de los objetivos de la tecnología de la realidad virtual.

Objetivos

Como puede suponerse, los objetivos de esta tecnología son tantos como aplicaciones se efectúen; sin embargo, el elemento principal consiste en permitir desarrollar acciones en terrenos donde lo imposible puede cuestionarse.

Esto puede parecer a primera vista superfluo. Sin embargo, actualmente la solución de problemas, en cualesquiera de las disciplinas científicas e inclusive en otras áreas, pasa primero por la simulación computacional, donde las condiciones y los límites son extrapolados, de tal manera que puedan obtenerse resultados que en condiciones normales, físicamente serían difícilmente realizables.

Ahora bien, la creación de condiciones ficticias, puede no ser en realidad sin consecuencias. Con esto se quiere decir que existen situaciones a las cuales no nos hemos enfrentado debido a las condiciones en las cuales vivimos. Es dentro de esta óptica que la realidad virtual puede ser comprendida más claramente.

Problemática

El problema fundamental en el tratamiento de datos (en cualesquiera de las disciplinas del procesado gráfico y de imágenes), ha consistido en desarrollar los métodos y técnicas para la adquisición, el análisis, la modulación, la interpretación, etc., de los datos, ya que la cantidad de información que se maneja es demasiado voluminosa.

Vamos a tratar de separar, si es posible, los diferentes problemas a los que se enfrentan a los que quieren aventurarse en esta área.

La interpretación de un objeto real pasa a través de una serie de acciones las cuales pueden ser (ya que existen muy variadas técnicas):

1. Adquisición de los datos, normalmente pasa por una tecnología de digitalización de imágenes.

2. Tratamiento y modelización de los datos adquiridos; en general puede ser un tratamiento gráfico (análisis de contornos, de los histogramas, variancia del entorno, filtrado , correlación, convolución, etc.). La modelización pasa por el tratamiento a base de métodos matemáticos que permiten obtener una representación del objeto real.

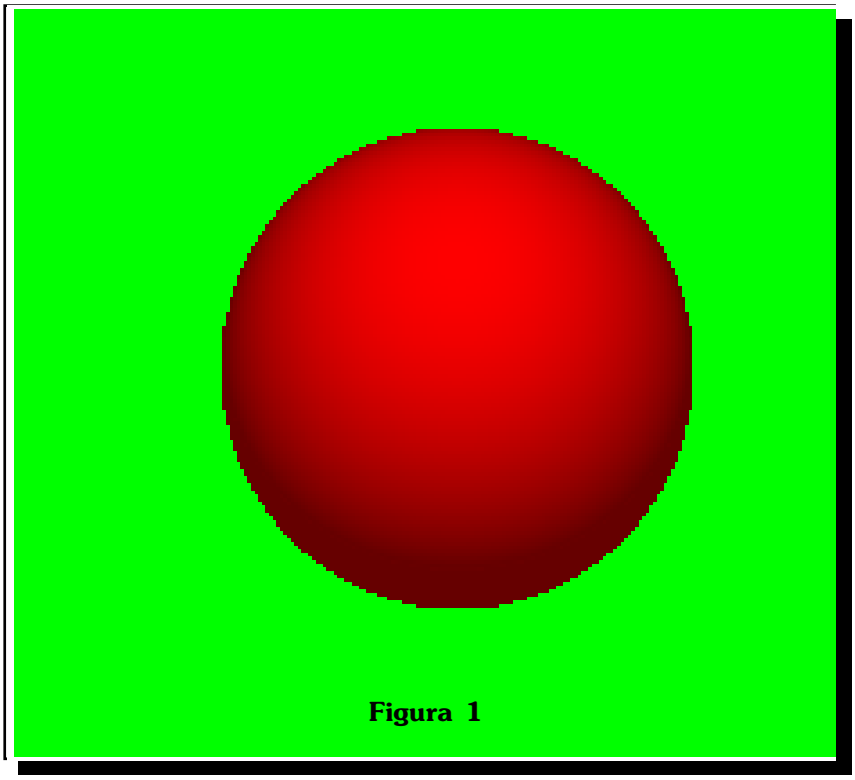


Figura 1

3. La interpretación del conjunto de datos en un ambiente gráfico.

La creación de un objeto sintético imaginario pasa por los dos últimos pasos del procedimiento

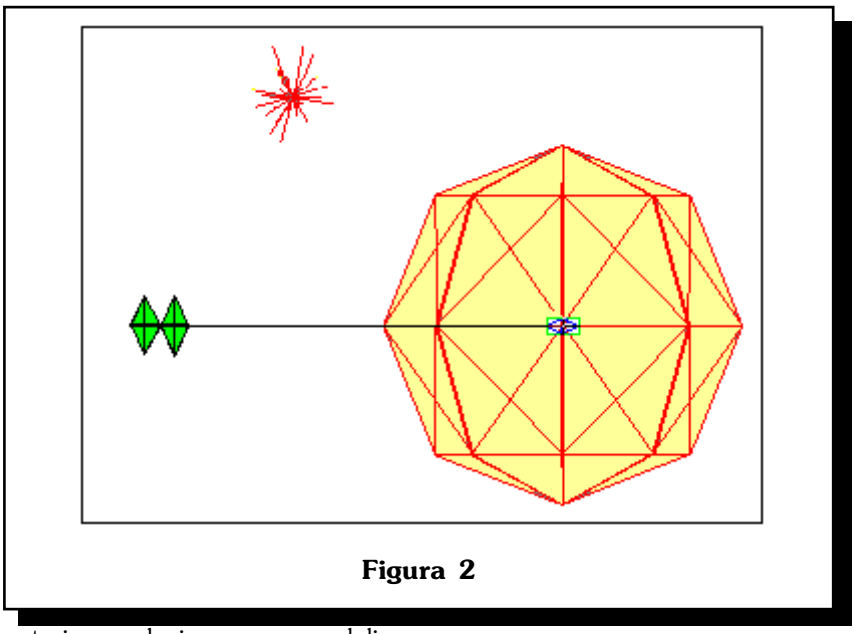


Figura 2

anterior, es decir, una vez modelizado y tratado, el objeto es interpretado. Por ejemplo, la **figura 1** muestra un objeto esférico, el cual ha sido calculado a partir de una construcción geométrica tridimensional llamada "hilos", **figura 2**.

Después de calcular la superficie en función de los parámetros: Textura, fuentes luminosas, punto de observación, etc. y de considerar los objetos que van a intervenir en el escenario, es necesario se diseñe este último teniendo en cuenta las posiciones y formas de los objetos. Esto permitirá más adelante, que la interacción pueda ser llevada a cabo en una buena armonía.

Bibliografía

[1] Joseph Graderick, "Réalité Virtuelle", SYBEX, París, 1994.

[2] Zahid Hussain, "Digital Signal Processing", ELLIS HORWOOD, England, 1991.

[3] Robert Sedgewick, "Algorithms in C", ADDISON-WESLEY, New York,1990.

[4] Ludovic Lecoint, "PV3D un modeleur d'image de synthèse", París, 1993.

Temporizador Programable en base a un dispositivo MAPL128.

Ing. Aquilino Cervantes Avila
Alumno de la Maestría del CINTEC-IPN

El trabajo con Dispositivos Lógicos Programables (PLD'S) se ha desarrollado ampliamente en este centro de estudios, por lo cual el objetivo de este trabajo es, por medio de un problema relativamente simple, introducir un dispositivo programable de gran potencial.

El trabajo con un Arreglo Múltiple de Lógica Programable ("Múltiple Array Programable Logic", MAPL), ofrece la ventaja de sustituir a múltiples dispositivos de lógica tradicional o de lógica programable; debido a esto, disminuye el área física en tarjeta, además de reducir el consumo de corriente de la fuente de poder empleada.

Hasta el momento se tiene noticia de 4 de estos dispositivos, siendo: MAPL's 128, 144, 244 y 268, fabricados por National Semiconductor. Si bien poseen ligeras variaciones en su configuración interna, la diferencia real la establece el número de compuertas que pueden sustituir y, por ende, el tamaño de su presentación física.

Para este proyecto se aplicará el dispositivo más sencillo de la serie, el MAPL 128. Para ello se realizará una descripción general de este dispositivo.

Características de los dispositivos MAPL128/144.

El MAPL128/144 pertenece a la tecnología EECMOS y posee una arquitectura de arreglos lógicos programables integrando múltiples FPLAS ("Field Programable Logic Array", Arreglo Lógico Programable en Campo), lo que permite facilitar el diseño de máquinas de estado, controladores, secuenciadores de microinstrucciones, interfaces y en general cualquier diseño lógico secuencial. Las características que posee son las siguientes:

Características generales de la serie MAPL128/144:

- Alta densidad, con arquitectura PLA.
- Velocidad de operación de 33,40 y 45 MHz dependiendo de la terminación en el dispositivo.
- Bajo consumo de potencia Icc = 110 mA máximo.
- Borrable eléctricamente.
  - * 100 % de funcionalidad de prueba.
  - * Lógica instantáneamente reconfigurable
  - * Capacidad Mínima de 100 ciclos programación - borrado.
- 27 macroceldas con registros tipo DE, JK, RS o T.
- Precarga asíncrona, y capacidad de reinicio.
- Reemplazo de múltiples dispositivos pal/pla.
- Soportado por OPAL y por software popular dedicado al tema.
- Celdas de seguridad que previenen el copiado.

La arquitectura MAPL es funcionalmente equivalente a un FPLA grande, teniendo un total de 128 productos términos. En realidad, el MAPL128 tiene ocho planos o páginas FPLAS, consistiendo cada página de un arreglo AND y un arreglo OR, ambos programables.

Cada arreglo AND posee 58 entradas y 16 términos producto, los cuales pueden conectarse a cualquiera de las 54 sumas de términos, por lo que cada página tiene una configuración de 58 X 16 X 54.

Paginación

Tres registros internos manejan la paginación, teniendo solo una página activa en un momento dado. Al igual que los demás registros internos, estos pueden usarse también para almacenar bits.

Las páginas se seleccionan en el momento de la aplicación; tal como las salidas de las macroceldas

Características del MAPL128

Terminales de alimentación	2
Entrada de reloj	1
Entrada de habilitación	1
Terminales de entrada	9
Terminales de salida	4
Terminales de entrada/salida	12
Total	28
Registros de página	3
Registros Internos*	8
Registros de salida	16
Total	27
Total de arreglos AND (entradas)	32
Total de arreglos OR (salidas)	54
Términos de transición lógicos	128
Control de términos	4
Suma de términos	432
Total de términos producto y suma	564

* Registros internos se refiere a que con estos se puede lograr la conexión interna, es decir, un nodo interno.

son controladas por reloj, así también lo son los registros de página. Lo anterior tiene el efecto de preseleccionar la página donde se encuentra el siguiente estado lógico. Debido a que el dispositivo posee arquitectura PIPELINE, la siguiente página no acumula retardo, con lo que la paginación aparece transparente al usuario.

La arquitectura que presenta el MAPL128/144 se muestra en la figura 1 .

Macroceldas

En general, los dispositivos MAPL incluyen una estructura de macrocelda flexible, configurable por el usuario. La misma macrocelda registrada se emplea para las funciones de salida, retroalimentación o entrada/salida. Cada macrocelda puede ser configurada como registro JK o D; ambos tipos se implementan en hardware y no requieren transformaciones por software o por lógica adicional. Todas las macroceldas se sincronizan por el flanco de subida de la señal de reloj.

Las macroceldas de salida presentan una señal de habilitación (OE) que permite llevar al dispositivo a un estado de alta impedancia. Es posible utilizar esta terminal en la función indicada o como entrada al arreglo AND; además, se tiene que las macroceldas de salida pueden ser configuradas como activas en alto o en bajo.

Las arquitecturas que presentan las macroceldas del MAPL 128 se muestran en las figuras 2 y 3.

NOTA: La macrocelda paginada es exactamente igual a la macrocelda interna. Por su parte, la macrocelda de solo salida, es igual a la de entrada/salida, excepto por la línea de retroalimentación.

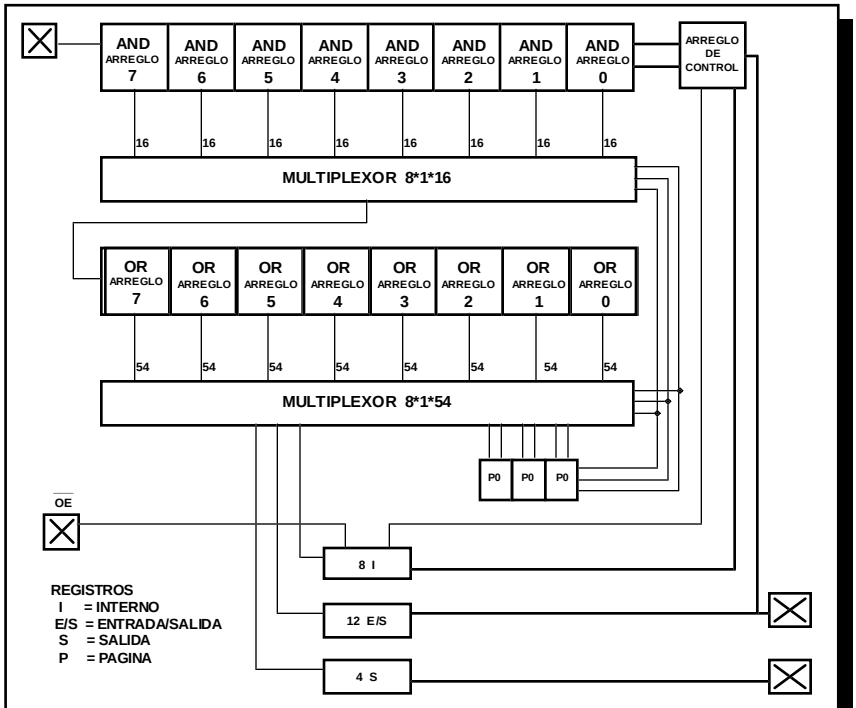
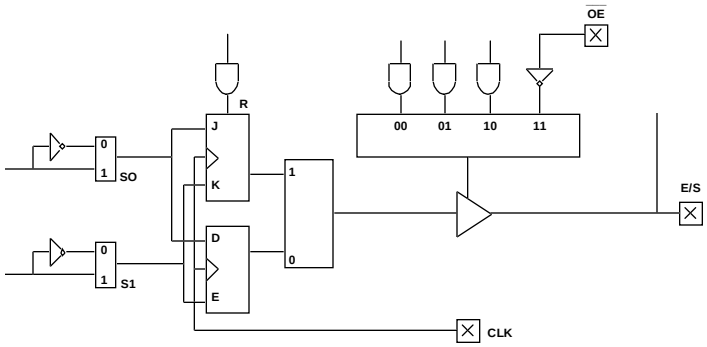
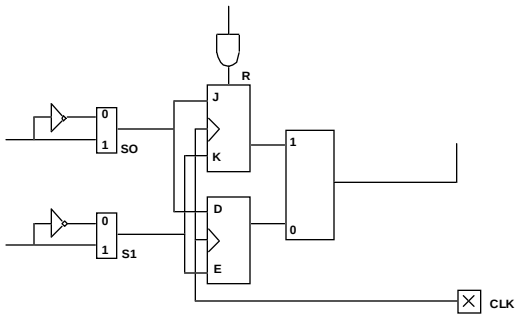


Figura 1



MACROCELDA DE ENTRADA/SALIDA DEL MAPL 128

Figura 2



MACROCELDA DE INTERNA DEL MAPL 128

Figura 3

Descripción General del temporizador implementado

El circuito consta de: 4 contadores tipo ascendente - descendente y una unidad de control; 3 contadores trabajan en módulo 10 y el restante en módulo 6. La unidad de control se encarga de conectar los contadores en cascada para la forma descendente y de forma individual para la opción ascendente; el orden en que la unidad de control conecta los contadores, da como resultado el número 9:59:9 para cuenta máxima, generando así el mayor retardo. La programación de cada contador se realiza en forma individual; para esto se elige alguno de ellos y se lleva a la cuenta deseada (cuenta en forma ascendente), una vez que la cuenta ha sido seleccionada, se marca el inicio de cuenta y los contadores inician su actividad en forma descendente.

Existen dos líneas de control, PROG Y CHG: la primera se utiliza para lograr la programación, mientras que la segunda para la selección del contador deseado. Una vez que el sistema se programa con ambas señales, se marca el inicio de cuenta descendente, a partir del valor preestablecido en la programación. Esto se resume en la siguiente tabla.

Se presenta además una señal de entrada que permite la inhibición de una nueva cuenta descendente (NEW), o el abandono de la cuenta actual; si esta señal se aterriza, después que el temporizador llega al fin de su cuenta descendente (0000) el circuito continúa con una nueva cuenta descendente a partir de 9 : 59 : 9, y así en forma sucesiva. Si se desea tener solo una cuenta, entonces la señal de ENDCOUNT se conecta en forma directa a la terminal NEW. De esta forma, la cuenta se detiene en cero y permite programar al sistema nuevamente, si así se desea, o reiniciar la cuenta en que se encontraba el temporizador en el momento en que se detuvo.

El diagrama a bloques del temporizador se muestra en la figura 4.

Programa de aplicación

Para este proyecto se empleó la herramienta de programación OPAL, proporcionado por National Semiconductor. El método que se usó para la solución del problema es el de tablas de verdad. Dado que el OPAL no soporta dos tipos de tablas diferentes (si se desea verificar el ejemplo), es necesario trabajar con métodos alternativos, o compilar en forma parcial las diferentes tablas, y utilizar los módulos EQN.

PROG	CHG	FUNCION
0	0	Inicio de cuenta descendente
0	1	Cuenta ascendente para el contador elegido
1	0	Cambio de contador
1	1	Cuenta descendente

Una vez que la cuenta termina, una línea señaliza el fin de la cuenta ENDCOUNT.

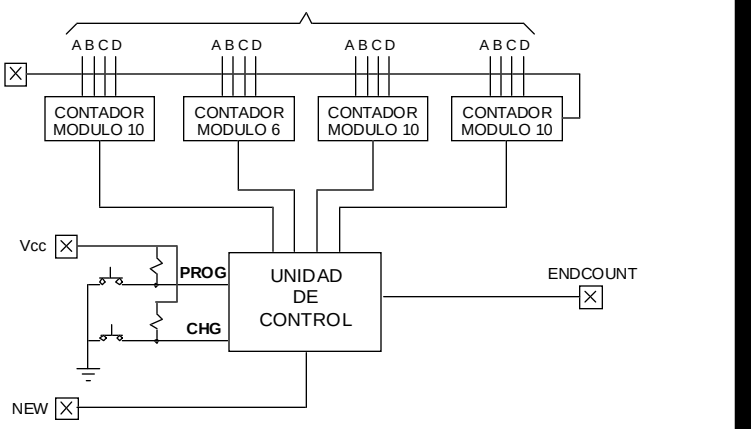


Figura 4

En esta aplicación se maneja paginación manual para lograr un mejor control sobre el sistema. Si en el sistema se deja que el OPAL realice la paginación automática, es probable que se marque errores en la compilación , esto es debido a que el OPAL intenta aprovechar al máximo el espacio por página, por lo que es posible que alguna ecuación sobrepase el número máximo de términos producto y no sea posible crear el archivo JED para el programa. A continuación se tiene una explicación del programa, y en cuadros anexos el listado del mismo.

Explicación general del programa

El programa ocupa las 8 páginas del MAPL, esto es, cada contador ocupa dos páginas, una de ellas se destina para la cuenta ascendente y otra más para la cuenta descendente. La unidad de control se organiza dentro de las 8 páginas y va ligada en forma directa con el control de la paginación.

Si se observa en forma detenida la manera en que se logra la conexión de los contadores en cascada, se podrá ver que ésta se da gracias precisamente a que las páginas se encuentran multiplexadas, así la conmutación de páginas es la que realiza la conexión indicada y debido a que todas las salidas son registradas, se observa el efecto de captura (HOLD).

También es necesario indicar que el MAPL 128 posee únicamente 12 salidas configurables también como entradas; si consideramos que el temporizador emplea 4 contadores con un total de 15 salidas, se aprecia la necesidad de obtener la retroalimentación de alguna manera para compensar las tres salidas faltantes. Para evitar un aca-

```
BEGIN HEADER
                                IPN
        CENTRO DE INVESTIGACION TECNOLOGICA EN COMPUTACION
              EDIFICIO DE GRADUADOS DE UPIICSA
        CALLE TE No. 950  COL. GRANJAS MEXICO   MEXICO D.F.  CP. 08400
              ALUMNO: AQUILINO CERVANTES AVILA

.DESCRIPCION:
        El siguiente programa implementa un temporizador programable en un MAPL 128

ENDHEADER

BEGIN DEFINITION

    Device MAPL128;

    Inputs    clk,chg,prog,new;              {señales de control }

    Outputs   (jk,hold) endcount,al0,al1,al2;  {señales solo salida }

    Feedbacks (jk,hold) sl0,sl1,sl2,sl3;          {forman al contador menos significativo}
    Feedbacks (jk,hold) ml0,ml1,ml2,ml3;  {forman al contador que sigue al menos significativo}
    Feedbacks (jk,hold,buried) l0,l1,l2;          {forman al contador que sigue al más significativo}
    Feedbacks  (jk,hold) s0,s1,s2,s3;             {forman al contador más significativo}
    Feedbacks  (buried) p0,p1,p2;                 {registros internos de paginación}

END DEFINITIONS

BEGIN EQUATIONS

    al0 = l0;                  { se ocupan  tres registros internos ( l0, l1 y  l2) para el contador }
    al1 = l1;                  { módulo 6, y se conectan a tres salidas (al0, al1 y al2)   }
    al2 = l2;
                               {se  garantiza que ENDCOUNT está activa solo al  término   }
                               {de la cuenta descendente.                                 }
    aux 1= sl0 * /p0 * /p1 * p2;
    aux2 = /sl1 * /sl2 * /sl3 * /ml0 * /ml1 * /ml2 * /ml3 ;

    endcount = aux1 * aux2 * /l0 * /l1 * /l2 * /s0 * /s1 * /s2 * /s3 ;

END EQUATIONS

BEGIN TRUTH_TABLE          {  tabla de verdad para el contador ascendente   }
                                {   menos significativo                                 }

    TTIN chg,prog,sl3,sl2,sl1,sl0,p2,p1,p0;
    TTOUT sl3,sl2,sl1,sl0,p2,p1,p0;


    01   ——    000  ???? 001     { si chg,prog están en 01 o en 11, sin importar el  }
    11   ——    000  ???? 000     { contenido de sl3-sl2, captura su valor y hace un  }
```

```
    00  ——   000   ???? 100 { cambio de página a 001  o 000 respectivamente }
                              { La combinación chg,prog = 00 provoca un inicio }
                          { de cuenta descendente     }
    10  0000 000 0001 000
    10  0001 000 0010 000
    10  0010 000 0011 000 { la combinación chg,prog = 10 permite al contador }
    10  0011 000 0100 000{ continuar con su cuenta                          }
    10  0100 000 0101 000
    10  0101 000 0110 000
    10  0110 000 0111 000
    10  0111 000 1000 000{ si se efectúa el cambio de página a 001, se busca }
    10  1000 000 1001 000{ el siguiente contador ascendente por programar   }
    10  1001 000 0000 000

ENDTRUTH_TABLE

BEGINTRUTH_TABLE              { tabla de verdad para el contador descendente }
                                  { menos significativo                     }

    TTIN new,chg,prog,sl3,sl2,sl1,sl0,p2,p1,p0;
    TTOUT sl3,sl2,sl1,sl0,p2,p1,p0;

    111  —— 100   ???? 000 { si  new,chg y prog se encuentran en 1, se    }
                              { captura el contenido de sl3-sl0 y salta a la  }
                               { página 000 (donde es posible la precarga) }
    011  0000 100      1001 101
    011  1001 100      1000 100
    011  1000 100      0111 100
    011  0111 100      0110 100{ para que el contador trabaje es necesario la    }
    011  0110 100      0101 100{ combinación de bits 011 dada por new, chg    }
    011  0101 100      0100 100{  y prog en forma respectiva. Este contador se  }
    011  0100 100      0011 100{localiza en la página 100                      }
    011  0011 100      0010 100
    011  0010 100      0001 100
    011  0001 100      0000 100

ENDTRUTH_TABLE

BEGINTRUTH_TABLE              { tabla de verdad para el contador ascendente- }
                                  { descendente que sigue al menos significativo  }

    TTIN chg,prog,ml3,ml2,ml1,ml0,p2,p1,p0;
    TTOUT ml3,ml2,ml1,ml0,p2,p1,p0;

    01  ——   001    ???? 011          { si chg,prog están en 01 o en 11, sin importar el   }
    11  ——   001    ???? 001          { contenido de sl3-sl2, captura su valor y hace un  }
    00  ——   001    ???? 100     { cambio de página a 011  o 001 respectivamente }
                              { La combinación chg,prog = 00 provoca un inicio }
                          { de cuenta descendente     }
    10  0000 001  0001 001
    10  0001 001  0010 001
    10  0010 001  0011 001
    10  0011 001  0100 001          { el contador ascendente se encuentra en la página }
    10  0100 001  0101 001          { 001 y trabaja con la combinación 01 dada por chg }
    10  0101 001  0110 001          {y prog                                            }
    10  0110 001  0111 001
    10  0111 001  1000 001
    10  1000 001  1001 001
    10  1001 001  0000 001

    11  0000 101  1001 110
    11  1001 101  1000 100
    11  1000 101  0111 100
    11  0111 101  0110 100      { el contador descendente se encuentra en la página  }
    11  0110 101  0101 100      {101 actualiza registros y regresa al contador descen- }
    11  0101 101  0100 100      {dente menos significativo. Si  existe un cambio de     }
    11  0100 101  0011 100      {estado 0000 a 1001, se busca el contador siguiente  }
    11  0011 101  0010 100      {en la página 110                                    }
    11  0010 101  0001 100
    11  0001 101  0000 100

ENDTRUTH_TABLE
```

rreo externo, y además el uso probable de lógica adicional, el contador módulo 6 se efectúa por medio de 3 registros internos conectados, también en forma interna, a 3 registros de sólo salida, solucionando con esto el problema.

De esta forma, si el MAPL cuenta con 16 terminales de salida, resta indicar que la última salida se utiliza para marcar el fin de cuenta.

Conclusiones

El trabajo con el MAPL 128 sirve como introducción para el manejo de otros dispositivos más complejos, tales como GATE ARRAY's, FPGA's, ASIC's, etc. Sin embargo, aunque en este ejemplo no se utiliza, es necesario usar la técnica de "pruebabilidad", que permite verificar si un proyecto dentro de un dispositivo programable se encuentra trabajando en forma adecuada.

Lo mas interesante de este ejemplo es, sin duda, el manejo de la paginación. Las páginas dentro del MAPL se encuentran multiplexadas, por lo cual solo un plano está activo a la vez. Debido a esta característica es posible insertar los decodificadores de BCD a 7 segmentos dentro del propio MAPL. Sin embargo, debido a que se necesita multiplexar las líneas del decodificador (si se desean tener líneas independientes, no son suficientes las salidas del dispositivo), sería necesario el aumentar un latch para cada contador y agregar al menos dos señales (con las que se tienen cuatro combinaciones) de control para indicar en que momento se encuentran válidos los datos de qué contador. El trabajo, finalmente, resulta en agregar circuitos tipo latch o colocar decodificadores, optándose por lo segundo. El ejem-

```
BEGIN TRUTH_TABLE              { tabla de verdad para el contador ascendente- }
                                    { descendente que sigue al más significativo    }

    TTIN chg,prog,I2,I1,I0,p2,p1,p0;
    TTOUT I2,I1,I0,p2,p1,p0;

    01   ——   010    ???? 011          { si chg,prog están en 01 o en 11, sin importar el  }
    11   ——   010    ???? 010          { contenido de I2 - I0, captura su valor y hace un  }
    00   ——   010    ???? 100      { cambio de página a 011  o 010 respectivamente }
                                    { La combinación chg,prog = 00 provoca un inicio }
                                { de cuenta descendente     }

    10  0000 010  0001 010
    10  0001 010  0010 010
    10  0010 010  0011 010
    10  0011 010  0100 010          { el contador ascendente se encuentra en la página }
    10  0100 010  0101 010          { 010 y trabaja con la combinación 01 dada por chg }
    10  0101 010  0110 010          {y prog                                            }
    10  0110 010  0111 010
    10  0111 010  1000 010
    10  1000 010  1001 010
    10  1001 010  0000 010

    11  0000 110  1001 111
    11  1001 110  1000 100
    11  1000 110  0111 100
    11  0111 110  0110 100      { el contador descendente se encuentra en la página  }
    11  0110 110  0101 100      {110 actualiza registros y regresa al contador descen- }
    11  0101 110  0100 100      {dente menos significativo. Si  existe un cambio de     }
    11  0100 110  0011 100      {estado 0000 a 1001, se busca el contador siguiente  }
    11  0011 110  0010 100      {en la página 111                                    }
    11  0010 110  0001 100
    11  0001 110  0000 100

END TRUTH_TABLE

BEGIN TRUTH_TABLE              { tabla de verdad para el contador ascendente- }
                                    { descendente que sigue al más significativo     }

    TTIN chg,prog,s3,s2,s1,s0,p2,p1,p0;
    TTOUT s3,s2,s1,s0,p2, p1,p0;

    01   ——   011    ???? 000          { si chg,prog están en 01 o en 11, sin importar el  }
    11   ——   011    ???? 011          { contenido de I2 - I0, captura su valor y hace un  }
    00   ——   011    ???? 100          { cambio de página a 011  o 010 respectivamente }
                                    { La combinación chg,prog = 00 provoca un inicio }
                                { de cuenta descendente     }

    10  0000 011  0001 011
    10  0001 011  0010 011
    10  0010 011  0011 011
    10  0011 011  0100 011      { el contador ascendente se encuentra en la página }
    10  0100 011  0101 011      { 011 y trabaja con la combinación 01 dada por chg }
    10  0101 011  0110 011      {y prog.                                           }
    10  0110 011  0111 011
    10  0111 011  1000 011
    10  1000 011  1001 011
    10  1001 011  0000 011

    11  0000 111  1001 100
    11  1001 111  1000 100
    11  1000 111  0111 100
    11  0111 111  0110 100
    11  0111 111  0110 100      { el contador descendente se encuentra en la página   }
    11  0110 111  0101 100      {110 actualiza registros y regresa al contador descen-  }
    11  0101 111  0100 100      {dente menos significativo. Este contador es el más     }
    11  0100 111  0011 100      {significativo y no necesita llevar acarreo a otro contador}
    11  0011 111  0010 100
    11  0010 111  0001 100
    11  0001 111  0000 100

END TRUTH_TABLE
```

plo descrito, con los decodificadores integrados, también se desarrolló y está a disposición de quien lo solicite. El paquete de OPAL, se trabajo en una computadora 80386DX a 40 MHz.

Bibliografía

[1] Programable Logic Devices Data Book and Design Guide. National Semiconductor, 1993.

[2] OPAL TM. Manual de Usuario.

Análisis, Diseño y Construcción de un Control Digital Directo usando la Computadora IPN E-32

M. en C. Romeo Urbieta Parrazales
Profesor e investigador del CINTEC-IPN
M. en C. Teodoro Alvarez Sánchez
Jefe del Departamento de Programación del CINTEC-IPN

Martín S. Domínguez González
Claudia Lara Vivas
Manuel Sáenz Sánchez
Raymundo Téllez Cortéz
Alumnos de la Maestría del CINTEC-IPN

Este artículo tiene tres grandes bloques de trabajo, **análisis, diseño, y construcción** de un sistema de control digital directo de un motor de C.D. En el análisis y diseño del sistema se emplearon la transformaciones ***s***, ***z*** y ***w***, para representar los modelos del motor, controlador, convertidores y tacómetro; el algoritmo empleado para este proceso es uno de tipo Proporcional Integral y Derivativo (P.I.D. por sus iniciales). Los criterios de estabilidad usados fueron: Jury, Nyquist, Bode, eliminación de polos y respuesta en el tiempo, para obtener el rango de valores de las ganancias del controlador. En la construcción se empleó una computadora personal IPN-E32 para instalar el algoritmo de control y para monitorear evoluciones de variables de interés en tiempo real, dicho algoritmo fue estructurado en lenguaje Turbo C; además se manejó una interface electrónica entre la computadora y el motor de C.D., conteniendo un puerto paralelo, convertidores A/D, D/A, F/V, amplificador de corriente; y un motor de C.D. de 24V, 1A y 3000 rpm.

Introducción

Durante la última década, el empleo de una computadora digital como dispositivo compensador a aumentado, ya que su precio y confiabilidad han mejorado notablemente. Debido esto, actualmente los sistemas de control digital se emplean en múltiples aplicaciones: para máquinas herramienta, procesos metalúrgicos, procesos químicos, control de aviones, etc.

Las ventajas de usar control digital se reflejan en una mayor sensibilidad y una gran inmunidad a la distorsión de las señales, básicamente. La sensibilidad se consigue gracias al empleo de señales de baja energía, mientras que la inmunidad a la distorsión causada por el ruido y a las no linealidades, se obtiene gracias al buen acoplamiento de estas señales en los dispositivos digitales empleados.

La principal desventaja de usar control digital es que la precisión de un computador y de los convertidores asociados está limitada por una longitud de palabra finita; así mismo, los convertidores (A/D y D/A) introducen un error de cuantificación de amplitud. Cuando el error de cuantificación y el error debido al tamaño finito de palabra del computador son pequeños en relación con la amplitud de la señal, entonces el sistema es suficientemente preciso y se pueden ignorar estas limitaciones. [1,2]

Análisis y Diseño de Control

El sistema cuenta con dos partes principales: el computador y la interface de potencia del motor. El computador ocupa una tarjeta convertidora A/D - D/A [6] controlada mediante un programa, el cual toma un valor analógico en la tarjeta de potencia (voltaje obtenido por el tacómetro), lo procesa y entrega un valor analógico, el cual excita a la etapa de potencia del motor [4,5]. La interface de potencia cuenta con un amplificador de potencia, y un convertidor de frecuencia a voltaje (F/V), éste último dará el voltaje que se introducirá al A/D, en relación a la velocidad del motor. El amplificador tiene una ganancia que producirá el voltaje necesario para el motor (0-24V), a partir del voltaje que proporciona el D/A (0-5V), [4,5]. En la **figura 1** se observa un diagrama general del lazo de control.

Función de Transferencia del Sistema

El sistema analizado se describe en la **figura 2**, la cual contempla todas y cada una de las partes y las funciones de control que involucra cada sección. Para efectuar el análisis es necesario encontrar la fun-

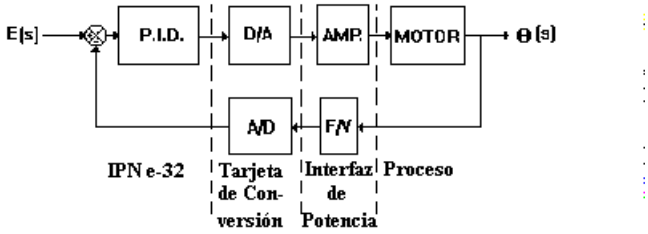


Figura 1

ción de transferencia del sistema, sin controlador P.I.D., y en lazo abierto. Empleando K1=4.8, Km=2105, a=80 y T=1 mseg, de acuerdo a la referencia [5], se obtienen las ecuaciones de la **figura 3**.

Observando la ecuación se aprecia un polo en z=1; por lo tanto, el error en estado estacionario a lazo cerrado será igual a cero. El siguiente paso será calcular los valores de las constantes Kp, Ki y Kd, para lo cual se igualará el numerador de la función de transferencia del control P.I.D. con el denominador de la función a lazo abierto [3]. Esto puede apreciarse en la **figura 4**.

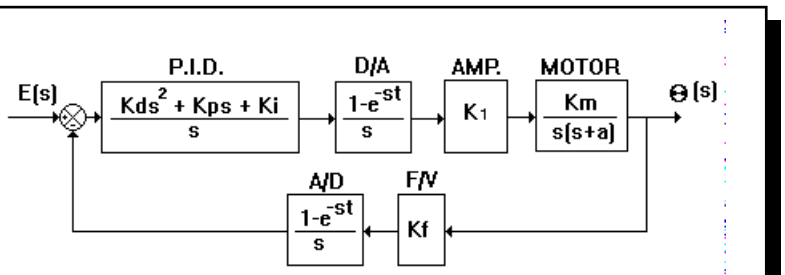


Figura 2

$$G(s)=(1-e^{-st})\frac{K1\ Km}{s^2(s+a)} \quad -(1)$$

$$G(s)=(1-e^{-st})\frac{10104}{s^2(s+80)} \quad -(2)$$

$$G(z)=\frac{0.0049(z+0.974)}{(z-1)(z-0.923)} \quad -(3)$$

Figura 3

De la ecuación [7], se obtienen los valores para Kp, Ki y Kd, **figura 5**.

Una vez obtenidos los valores, la función de transferencia del sistema compensado a lazo cerrado se obtiene sustituyendo dichos datos en las ecuaciones [11], [12], [13] y [14], como se observa en la **figura 6**.

A esta ecuación se aplican los criterios de estabilidad. Variando los valores de Kp, Ki y Kd, se tienen ecuaciones iguales en estructura a la ecuación 14, pero con diferentes gráficas de respuesta. Al final de las pruebas, se obtienen los valores siguientes:

$$\frac{\overline{\Theta[z]}}{E[z]}=\frac{0.5(z+0.974)}{z-0.9963z+0.0036} \quad -(15)$$

Kp = 7.854
Ki = 0
Kd = 0.094

Prueba de estabilidad de Nyquist, Bode y Jury

El análisis de sistemas de control muestreados se realiza bajo el dominio del tiempo y la frecuencia. Los métodos más apropiados para el análisis de estos sistemas son:

1. **Dominio de la frecuencia**.- Método de transformada ***s*** (Transformada de Laplace), respuesta en frecuencia, transformada ***z*** y transformada ***w*** (transformación bilineal).

$$G(s)=\frac{Kds^2+Kps+Ki}{s} \quad -(4)$$

$$G(z)=\frac{Kd\,(z-1)}{T(z)}+Kp+\frac{Ki\,T\,(z+1)}{2(z-1)} \quad -(5)$$

$$G(z)=\frac{(KiT^2+2KpT+2Kd)z^2+(KiT^2-2KpT-4Kd)z+2Kd}{2Tz(z-1)} \quad -(6)$$

$$(KiT^2+2KpT+2Kd)z^2+(KiT^2-2KpT-4Kd)z+2Kd=(z-1)(z-0.923) \quad -(7)$$

Figura 4

$$KiT^2 + 2KpT + 2Kd = 1 \quad \text{- (8)}$$

$$KiT^2 - 2KpT - 4Kd = -1.923 \quad \text{- (9)}$$

$$2Kd = 0.923 \quad \text{- (10)}$$

$$Kd = 0.462$$
$$Kp = 38.5$$
$$Ki = 0$$

Figura 5

2. **Dominio del tiempo**.- Método de ecuación de diferencias y respuesta impulso.

Estos métodos se utilizan **para analizar el comportamiento del sistema** en el dominio de la frecuencia y determinar si éste cumple con las **condiciones necesarias de estabilidad.**

Un sistema de control con retroalimentación lineal continuo es estable si todos los polos de la función de transferencia de circuito cerrado están en la mitad izquierda del **plano *s***; el **plano z** está relacionado con el **plano s** por la transformación:

$$z = e^{sT}$$

$$z = e^{(s + jw)T}$$

Esta relación se puede escribir también como:

$$|z| = e^{sT}$$

$$z = wT$$

En la parte izquierda del plano ***s*** se tiene $s < 0$, por lo tanto, la magnitud correspondiente de ***z*** varía de 0 a 1. Por esta razón, el eje imaginario del plano s corresponde al círculo unitario en el plano z y la parte interna del mismo círculo corresponde a la mitad izquierda del plano s. En consecuencia, se puede establecer que un sistema muestreado es estable si todos los polos de la función de transferencia de circuito cerrado están dentro del círculo unitario **(figura 7)**.

$$G'[z] = \frac{2.45(z+0.974)}{z(z-1)} \quad \text{- (11)}$$

$$H[z] = 0.0074 \quad \text{- (12)}$$

$$\frac{\Theta[z]}{E[z]} = \frac{G'[z]}{1+G'[z]H[z]} \quad \text{- (13)}$$

$$\frac{\Theta[z]}{E[z]} = \frac{2.45\,(z+0.974)}{z - 0.982z + 0.0175} \quad \text{- (14)}$$

Figura 6

El método de estabilidad de Nyquist establece que al graficar la función de transferencia en el plano z, el lugar geométrico resultante no debe encerrar al punto -1,0, ya que esto implica la existencia de polos o ceros en el lado derecho del plano ***s***, esto es, inestabilidad del sistema.

El diagrama de Nyquist se muestra en la **figura 8**; una manera de entenderlo mejor es por medio del diagrama de Bode, el cual nos muestra el margen de fase y el margen de ganancia del sistema. Para nuestro caso, el margen de fase es aproximadamente 39 grados, y el margen de ganancia es 6.4 dB; los parámetros anteriores nos dan una idea de la estabilidad relativa del sistema. El diagrama de Bode se muestra en la **figura 9**.

Por último se realiza la prueba de estabilidad de Jury, que por medios algebraicos muestra si existe o no estabilidad en el sistema.

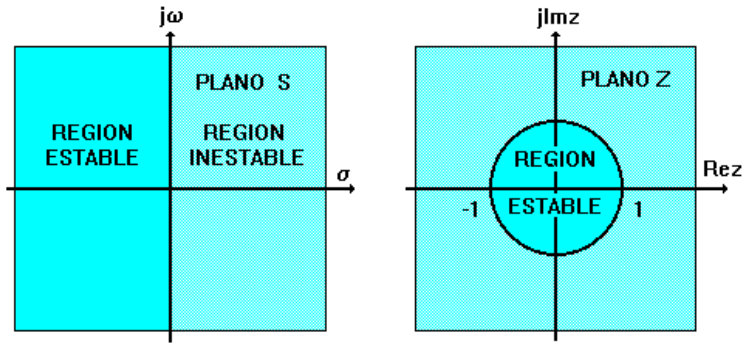


Figura 7

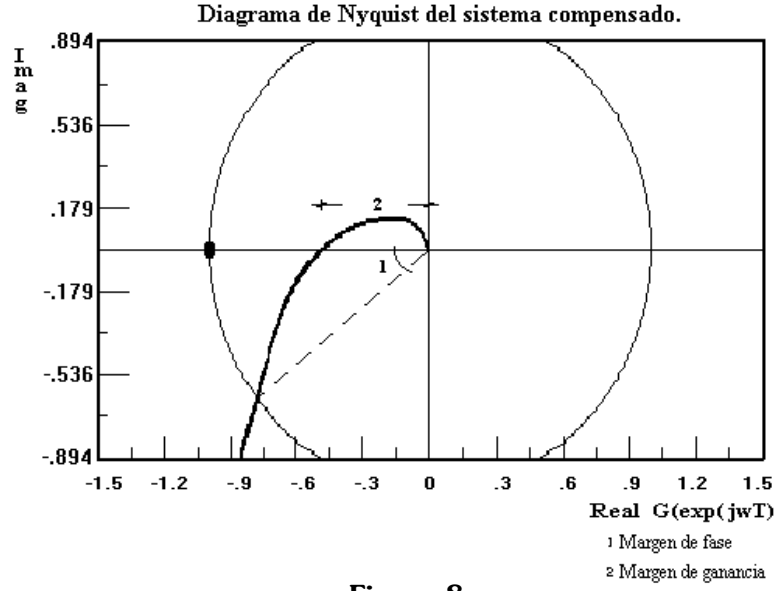


Figura 8

De la Ec. [15] se deduce la ecuación característica :

$F(z) = z^2 - 0.9963z + 0.0036$

1a. Prueba de Jury

$F(1) > 0$

$(1)^2 - 0.9963(1) + 0.0036 > 0$

$0.0073 > 0$

2a. Prueba de Jury

$(-1)^n F(-1) > 0$ {n = grado de la ecuación}

$(-1)^2 [(-1)^2 - 0.9963(-1) + 0.0036] > 0$

$1.9999 > 0$

3a. Prueba de Jury

z^0	z^1	z^2
0.0036	-0.9963	1

$|a_0| < a_2$

$|0.0036| < 1$

Por lo tanto, según el criterio de Jury, el sistema controlado por el P.I.D. es estable. La gráfica de respuesta en el tiempo del sistema incluyendo el controlador P.I.D. se muestra en la **figura 10**. Jugando con los valores de las constantes se puede obtener otro tipo de respuesta, ya sea con mayor o menor amortiguación

Diseño y Construcción

El sistema controlador se basa en el empleo de la computadora IPN-E32 con un algoritmo de control P.I.D., para conexión con el motor, recepción y envío de datos con convertidores A/D, D/A, F/V; también cuenta con un amplificador de corriente y un motor de C.D. de 24V, 1 A.

Los convertidores A/D, D/A se encuentran conectados en la tarjeta instalada en el bus de la computadora; mientras que el F/V (convertidor frecuencia-voltaje) con el amplificador están en la tarjeta de interface de potencia [4], ver **figura 11**. La descripción de la tarjeta "interface de potencia", diagrama esquemático y el programa realizado para realizar el control se detallan a continuación.

Descripción del Circuito

El circuito consta de una fuente de alimentación propia, que pro-

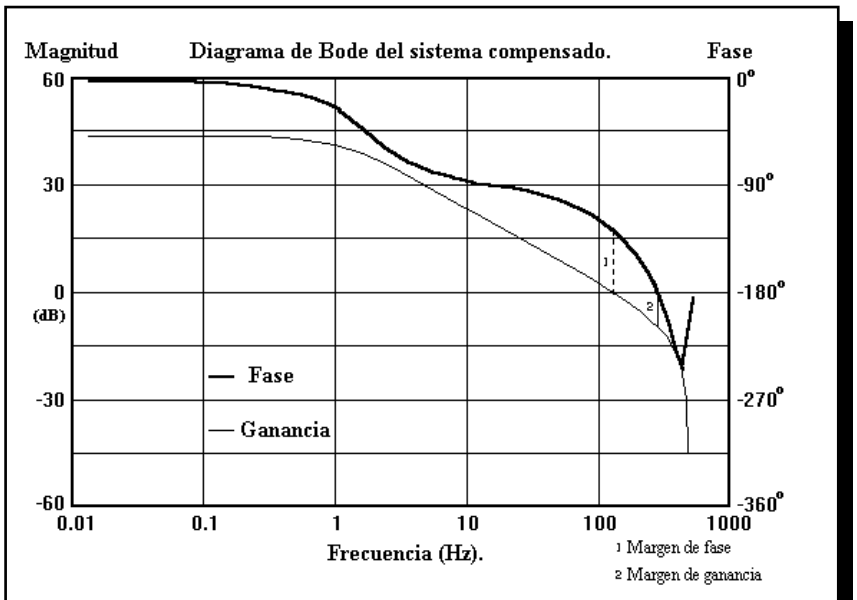


Figura 9

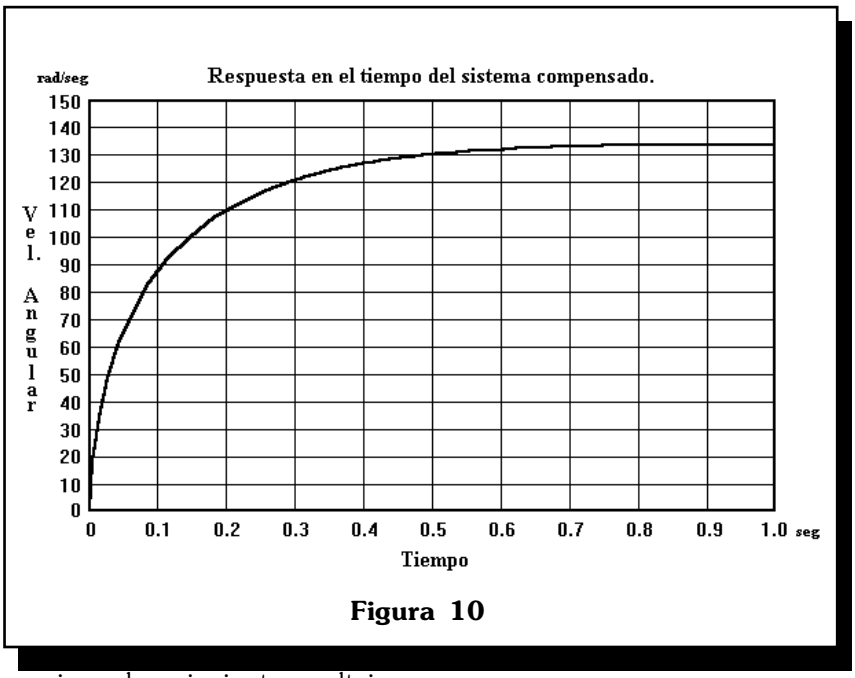


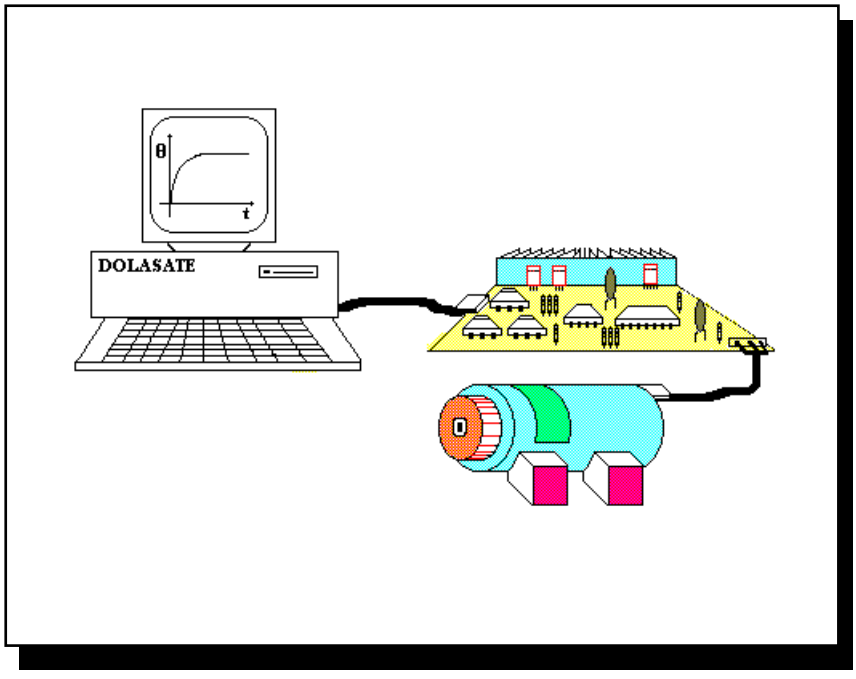
Figura 10

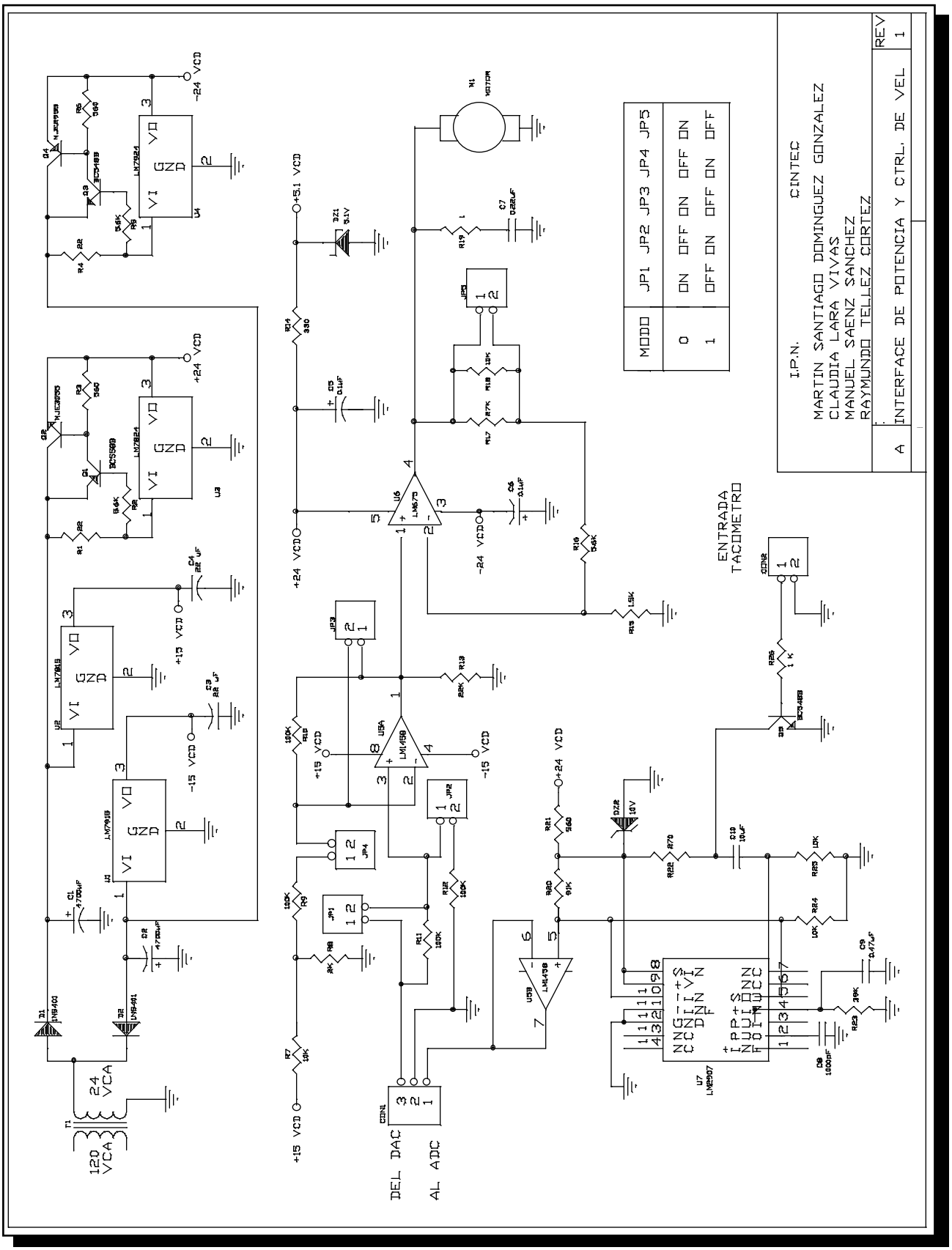
porciona los siguientes voltajes: +24V, -24V, +15V, -15V, +10V y +5V; con el fin de hacer el sistema autónomo los dos primeros proporcionan hasta 3A, que alimentan a un amplificador operacional de potencia (LM675); los dos siguientes a un amplificador operacional (LM1458), mientras que con los 10V se alimenta al convertidor de frecuencia a voltaje (LM2907) y 5V al circuito que está integrado al servomotor. La fuente de alimentación está basada en los reguladores 78XX, 79XX, diodos zener y arreglo de transistores, como se aprecia en el diagrama esquemático.

La tarjeta cuenta con un F/V (LM2907) el cual se ajustó (según formulas del manual de National Semiconductor) para permitir en la entrada frecuencias de 1600 Hz. a 8140 Hz., proporcionando a la salida un rango de 0.96V a 4.8 V. La salida del F/V está conectada a un amplificador operacional LM 1458 como seguidor de voltaje, con el fin de presentar una impedancia de salida más baja (75 ohms) al A/D, que es implementado por el LM2907. Lo anterior permite que estos voltajes puedan introducirse al A/D; de esta manera se envía a la computadora un dato digital, el cual indica la velocidad a la que gira el motor.

Una vez procesado el dato por el algoritmo P.I.D. (como puede verse en el programa); en el D/A se tiene una señal analógica que entra a un amplificador operacional LM1458 como seguidor de voltaje, con el fin de presentar ante el D/A una impedancia de entrada alta. La salida del seguidor de voltaje se introduce al amplificador operacional de potencia LM675, el cual tiene un factor de amplificación de 4.74 (práctico), por lo que es posible alimentar al motor hasta con 21V aproximadamente (teóricamente sería hasta 24V, pero el amplificador operacional no puede proporcionar este voltaje a su salida, debido a que está siendo polarizado con 24V). El circuito está provisto para ser configurado de otra forma por medio de "puentes" y así ofrecer el giro en ambos sentidos. El LM1458 está configurado como sumador de dos voltajes: Vin y -2.5V; como Vin sólo puede variar de 0V a 5V, a la salida del sumador tendremos variaciones de -2.5V a 2.5V, los cuales permiten que el amplificador de potencia LM675 (con otro factor de amplificación: 9.6) pueda entregar voltajes positivos y negativos (± 24 volts) al motor.

Posteriormente, teniéndose una referencia (que será el valor espera-




MODO
JP1 JP2 JP3 JP4 JP5
0
ON OFF ON OFF ON
1
OFF ON OFF ON OFF
I.P.N.
CINTEC
MARTIN SANTIAGO DOMINGUEZ GONZALEZ
CLAUDIA LARA VIVAS
MANUEL SAENZ SANCHEZ
RAYMUNDO TELLEZ CORTEZ
INTERFACE DE POTENCIA Y CTRL. DE VEL
REV
1
ENTRADA TACOMETRO
DEL DAC
AL ADC
120 VCA
24 VCA
+24 VCD
-24 VCD
+15 VCD
-15 VCD
+5.1 VCD

do en la entrada del A/D) que indique la velocidad con que gira el motor (proporcionada por el F/V), se ejecuta el algoritmo P.I.D., y mediante el D/A se generará un voltaje que llegará al motor por medio de la interface de potencia, el cual alcanza y estabiliza la velocidad deseada, indicada en la referencia.

A continuación se tiene el listado del programa empleado en el sistema :

```
/*
                    INSTITUTO POLITECNICO NACIONAL
            CENTRO DE INVESTIGACION TECNOLOGICA EN COMPUTACION

    TEORIA DE CONTROL

    PROFESORES:
            M. EN C. ROMEO URBIETA PARRAZALEZ
            M. EN C. TEODORO ALVAREZ SANCHEZ
    ALUMNOS:
            MARTIN S. DOMINGUEZ GONZALEZ
            CLAUDIA LARA VIVAS
            MANUEL SAENZ SANCHEZ
            RAYMUNDO TELLEZ CORTEZ

                        DOLASATE INC.

    CONTROL P.I.D:

    Programa que completa el diagrama de flujo de un controlador empleando
    el algoritmo P.I.D.

    El programa acepta un nivel de referencia (velocidad del motor). El
    sistema muestra al usuario el nivel alcanzado realmente.

    El algoritmo P.I.D. trabaja en base a los valores predefinidos de las
    constantes Kp, Ki, y Kd.

*/

#include <STDIO.H>
#include <DOS.H>
#include <CONIO.H>
#include <GRAPHICS.H>

#define PUERTO_A  0X100     /* DIRECCION DEL PUERTO A DEL 8255            */
#define PUERTO_B  0X101     /* DIRECCION DEL PUERTO B DEL 8255            */
#define PUERTO_C  0X102     /* DIRECCION DEL PUERTO C DEL 8255            */
#define CONTROL   0X103     /* DIRECCION DEL PUERTO DE CONTROL DEL 8255 */

#define KP  0           /* CONSTANTE PROPORCIONAL                  */
#define KI  0.3462      /* CONSTANTE INTEGRAL                  */
#define KD  1.3849      /* CONSTANTE DERIVATIVA                    */

void portada();         /* SUBRUTINA DE PRESENTACION               */
void inicio();          /* SUBRUTINA DE INICIALIZACION DEL PROCESO     */
void pid();             /* SUBRUTINA DE CONTROL P.I.D.             */

charDATO;               /* VARIABLE PARA LEER UN DATO DEL ADC          */

int  TECLA;             /* VARIABLE QUE REGRESA EL CODIGO DE LA TECLA PRESIONADA
                           POR EL USUARIO       */
```

Prueba y Ajustes

Para lograr la implantación del sistema se probaron las partes que lo conforman de manera individual. Cabe resaltar que en la tarjeta de "interface de potencia" se realizaron algunas pruebas y ajustes, tomándose en cuenta las características de la tarjeta de los convertidores A/D y D/A, y las necesidades de alimentación del motor de C.D.; se sabe que a la salida del D/A se tienen voltajes de 0V a 5V, por lo que fue necesario aplicar un factor de ganancia de 4.8 al amplificador, para lograr alimentar al motor en un rango de 0V a 24V. Dependiendo de esta alimentación, el motor produce una frecuencia de giro, la cual, mediante el FVC, deberá producir voltajes de 0 a 5 volts que alimentan al ADC; esto último se logro configurando el circuito integrado LM2907 según fórmulas del manual de National Semiconductor.

Conclusiones

En el análisis y diseño del sistema de control se pudo representar a éste en una función de transferencia en transformadas ***s***, ***z***, ***w***; las pruebas (simulación) de estabilidad del sistema de control se aplicaron en conjunto para obtener un panorama más claro y confiable de funcionamiento (rangos de valores Kp, Ki, Kd). Aunque estas pruebas dieron un amplio rango de estabilidad, al llevarlos a la práctica (tiempo real) se tuvieron que realizar algunos ajustes (como por ejemplo, eliminar los sobretiros, y obtener un seguimiento del controlador igual a la del proceso, escogiendo un solo valor óptimo para cada Kp, Ki, y Kd) con lo cual se consiguió que el motor corriera perfectamente. También se pudo comprobar que algoritmo P.I.D. es ligeramente mejor que el P.I.

La construcción del sistema de control digital se llevó a cabo con tecnología elaborada y adquirida en el mercado nacional. Se consiguió capacitar recursos humanos así como utilizar materiales nacionales propios, propiciando con esto último ahorros económicos e independencia tecnológica en esta área. Finalmente, se pudo observar que

```
float     IN0  = 0,              /* VALOR DE LA ENTRADA EN UN INSTANTE t
     */
     IN1  = 0,              /* VALOR DE LA ENTRADA EN UN INSTANTE t-1
*/
     IN2  = 0,              /* VALOR DE LA ENTRADA EN UN INSTANTE t-2
*/

     OUT0 = 0,       /* VALOR DE LA SALIDA EN UN INSTANTE t          */
     OUT1 = 0,       /* VALOR DE LA SALIDA EN UN INSTANTE t-1            */

     REF  = 3,       /* VALOR DE LA REFERENCIA (EN VOLTS)            */
     PASO = 3,       /* VARIABLE DE PASO PARA LA REFERENCIA              */
     RETRO   ;       /* VALOR LEIDO DEL ADC                          */


     /* ————————————————     PROGRAMA PRINCIPAL     ———————————————— */
void  main(void)
{
portada();
inicio();
pid();
}

     /* ——————————————————— PRESENTACION ——————————————————— */
void  portada()
{
int gd,
    gm,
    tam_x;                          /* Tamaños de letras a utilizar */

/* Inicializamos gráficos */
gd=DETECT;
initgraph(&gd,&gm,"c:\\tc\\bgi");

/* Encuentra el tamaño óptimo para las letras */
tam_x=(getmaxx()/4)/25;

/* Escribe el título */
settextstyle(TRIPLEX_FONT,HORIZ_DIR,tam_x);
outtextxy(getmaxx()/3,getmaxy()/4,"CONTROL");
settextstyle(TRIPLEX_FONT,HORIZ_DIR,tam_x);
outtextxy((15*getmaxx())/36,getmaxy()/2,"P.I.D.");
settextstyle(TRIPLEX_FONT,HORIZ_DIR,2);
outtextxy(getmaxx()-190,getmaxy()-30,"by  DOLASATE  INC.");

/* Cerramos gráficos */
getch();
}

     /* ——————————————————— INICIALIZACION ——————————————————— */
void  inicio()
{
/*
     CONFIGURAMOS EL 8255:
     PUERTO A   SALIDA
     PUERTO B   ENTRADA
     PUERTO C   SALIDA
*/
outportb(CONTROL,0X82);

/* MANDAMOS 0 VOLTS AL DAC0 (FRENAMOS EL MOTOR) */
outportb(PUERTO_A,0x00);

/*
     CAPTURAMOS EL VALOR:
     PC5    1 LOGICO
     PC6    1 LOGICO
*/

outportb(CONTROL,0X0B);
outportb(CONTROL,0X0D);

/*
     DESACTIVAMOS EL DAC0
```

el circuito realizado "interface de potencia", puede tomarse como base para implantar otros algoritmos de control y comprobar los desarrollos teóricos, definiendo así si los sistemas lineales son estables, con perturbaciones suaves y bruscas en la carga del motor

La prueba del sistema de control digital se llevó a cabo mediante el programa estructurado en lenguaje Turbo C, dicho programa, de creación propia, fue instalado en la memoria de una PC (IPN-E32). Aquí, la computadora fue utilizada como elemento fundamental de control y como una poderosa herramienta de monitoreo.

Bibliografía

[1] DORF, Richard. *Sistemas Modernos de Control*. Addison-Wesley Iberoamericana, 2a. edición. México, 1989.

[2] HOUPIS, Constantine. *Digital Control Systems*. McGraw-Hill, 2a. edición. Singapore, 1992.

[3] KUO,Benjamin C. *Digital Control System*. Halt, Rinehart and Winston,Inc. 1980.

[4] ALVAREZ, Teodoro. *"Diseño y construcción de un sistema dedicado al control de motores de C.D. basado en el microcontrolador 8031 de INTEL."*. POLIBITS. Año IV, Vol.1, Num.10, Octubre-Diciembre 1992.

[5] PARRAZALES, Romeo. *"Simulación de la posición de un motor de C.D. de 1/15 de H.P controlado por un*

```
    PC5   0 LOGICO
    PC6   0 LOGICO
*/
outportb(CONTROL,0X0A);
outportb(CONTROL,0X0C);
getche();
closegraph();
clrscr();

gotoxy(8,5);
printf("VEL. REQUERIDA");
gotoxy(28,5);
printf("VEL. ALCANZADA");
gotoxy(13,10);
printf("CONTROL DE VELOCIDAD");
}

    /* ——————————————— ALGORITMO ——————————————— */
void pid()
{
for (;;){
    /*
        CONFIGURAMOS EL 8255:
        PUERTO A        SALIDA
        PUERTO B        ENTRADA
        PUERTO C        SALIDA
    */
    outportb(CONTROL,0X82);

    /*
        SE SELECCIONA LA INT0 DEL ADC COMO ENTRADA ANALOGICA:
        PC10 LOGICO
        PC20 LOGICO
        PC30 LOGICO
    */
    outportb(CONTROL,0X02);
    outportb(CONTROL,0X04);
    outportb(CONTROL,0X06);

    /*
        SE INICIA LA CONVERSION:
        PC41 LOGICO
        PC40 LOGICO
    */
    outportb(CONTROL,0X09);
    outportb(CONTROL,0X08);

    /*
        SE CONFIGURA 8255:
        PUERTO A
        PUERTO B
        PUERTO C
    */
    outportb(CONTROL,0X83);

    /* ESPERAMOS A QUE TERMINE LA CONVERSION */
    do{
        DATO=inportb(PUERTO_C);
        }while(DATO & 1);

    /* LEEMOS UN DATO DEL ADC-IN0 (RETROALIMENTACION) */
    DATO=inportb(PUERTO_B);

    RETRO=(5.0*DATO)/255;

    gotoxy(30,7);
    printf("%3.3f",(RETRO*657)/5);

    /*   REALIZAMOS EL CONTROL PID */
    IN0=REF-RETRO;

    OUT0=((KP+KI+KD)*IN0) - ((KP+(2*KD))*IN1) + (KD*IN2) + OUT1;
```

algoritmo PI usando la microcomputadora IPNe16-m." POLIBITS . Año IV, Vol.1,Num.9, Julio-Septiembre 1992.

[6] PARRAZALES, Romeo. *"Diseño y construcción de una tarjeta convertidora de 8 canales de A/D y 4 canales de D/A"*. POLIBITS, Año VI Vol. Num. 12. Enero-Marzo 1994.

```
    IN2=IN1;
    IN1=IN0;
    OUT1=OUT0;

    gotoxy(10,7);
    printf("%3.3f",(OUT0*657)/5);

    /* ESCRIBIMOS LA SALIDA DEL CONTROL AL DAC */
    outportb(PUERTO_A,(unsigned  char)  OUT0);

    outportb(CONTROL,0X0B);
    outportb(CONTROL,0X0D);

    outportb(CONTROL,0X0A);
    outportb(CONTROL,0X0C);

    /*
        ACEPTAMOS LAS TECLAS DE CURSOR
        ARRIBA AUMENTAMOS LA VARIABLE DE PASO
        ABAJO       DISMINUIMOS LA VARIABLE DE PASO
        ENTER       ACTUALIZAMOS EL VALOR DE LA REFERENCIA
        ESC     SALIMOS DEL PROGRAMA
    */
    if(kbhit())
        {
        switch(getch()){

            case 72:
                if(PASO<=4.78)    PASO+=0.02;
                gotoxy(20,12);
                printf("%1.3f",(PASO*675)/5);
                break;
            case 80:
                if(PASO>=1.02)    PASO-=0.02;
                gotoxy(20,12);
                printf("%1.3f",(PASO*675)/5);
                break;
            case 13:
                REF=PASO;
                break;
            case 27:
                /* PARAMOS EL MOTOR */
                outportb(PUERTO_A,0x00);

                outportb(CONTROL,0X0B);
                outportb(CONTROL,0X0D);

                outportb(CONTROL,0X0A);
                outportb(CONTROL,0X0C);
                exit();
            default:
                break;
            }
        }
}
```

El impacto de la nueva generación de Microprocesadores en la Ingeniería de Software

M. en C. Miguel A. Partida Tapia
Subdirector Académico y de Investigación del CINTEC-IPN.

La Ingeniería de Software ha tenido una evolución muy importante durante los últimos diez años; sin embargo, en relación al desarrollo del Hardware, se mantiene y en algunos casos se amplía un margen de retraso de por lo menos 5 años. Este desfasamiento se profundizó principalmente por la llegada de las computadoras personales y su alta demanda en las diversas áreas del conocimiento y por el tipo de usuarios.

Los principales avances del Hardware se han enfocado al desarrollo de microprocesadores tipo RISC ("Reduced Instruction Set Code", Conjunto Reducido de Instrucciones), Coprocesadores Gráficos y Almacenamiento Masivo de Datos. Un ejemplo de esto es el microprocesador i860 de Intel Corp., el cual integra una unidad de procesamiento tipo RISC, coprocesador numérico y coprocesador gráfico y, adicionalmente, capacidad para un alto ancho de banda de "*bus*" para transferencia de datos. Estas características de los microprocesadores han generado nuevas alternativas de procesamiento de datos y comunicaciones, principalmente el desarrollo de Multimedia, y en particular la capacidad de realizar comunicaciones de voz e imagen en tiempo real; como consecuencia, la comunicación vía satélite y la encriptación de la información en tiempo real es un hecho común en nuestros días.

La capacidad de estos microprocesadores ha quedado de manifiesto, sin lugar a dudas; sin embargo, todas estas aplicaciones son lo que se denomina, "Soluciones Integrales", donde los Ingenieros de Hardware y de Software no tienen más que aplicar esta solución y esperar los resultados. De este modo, la solución esperada estará acotada en las ventajas y limitaciones definidas por los diseñadores de estas "Soluciones Integrales". Para el presente análisis se partirá del caso en donde la arquitectura de la "Solución Integral" permita que diseñadores de Software desarrollen aplicaciones para resolver un tipo de problema en particular de la institución receptora.

A continuación se muestran algunas de las características que se considera limitan el desarrollo de Software:

a).- De la Arquitectura de Computadoras.

- Arquitectura del microprocesador.
- Tiempo de Acceso a memoria.
- Ancho de banda del "*bus*".
- Ubicación en la arquitectura de los periféricos.
- Características de los dispositivos periféricos.
- Capacidad de procesamiento de la unidad de punto flotante.
- Capacidad de manejo de memoria principal y virtual.

b).- Del Compilador.

- Capacidad de migración de código a otros sistemas.
- Capacidad de compactación de código.
- Capacidad de generación de código utilizando las características del microprocesador.

c).- Del Sistema Operativo.

- Capacidad de administración de recursos.
- Capacidad de administración de dispositivos periféricos.
- Velocidad de respuesta en sus rutinas de atención.

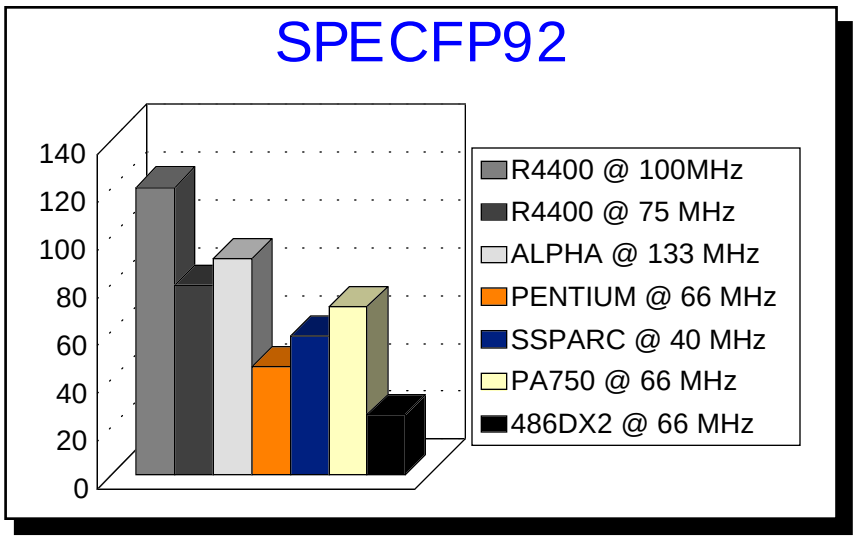
Con lo anterior se puede concluir que todo desarrollo de Software en una computadora específica se encuentra en dependencia directa de los alcances y limitaciones de su arquitectura, del compilador en que se desarrolló la aplicación y del sistema operativo en que trabajará dicho Software.

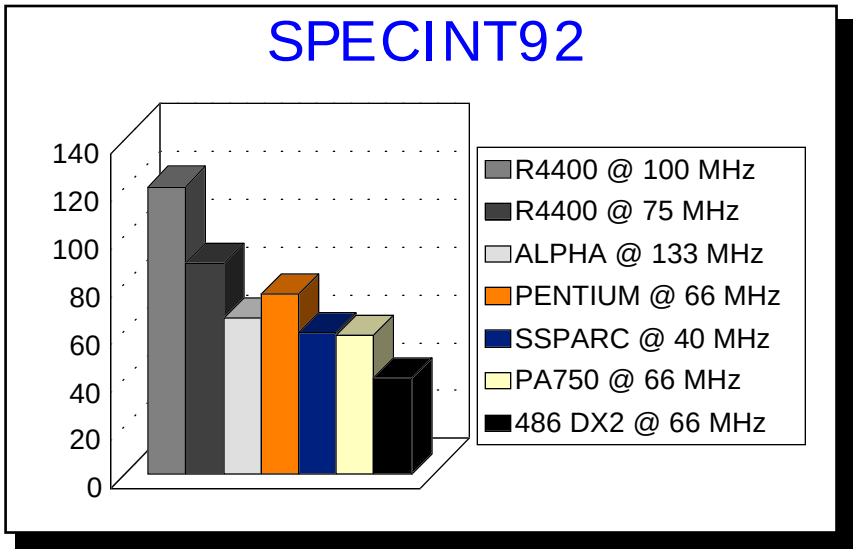
Un estandard en los diseñadores de Software se ha establecido en el lenguaje C de programación, mismo que se encuentra en el mer-

cado con diversas facetas, desde un seudo lenguaje de alto nivel (como originalmente fue propuesto), hasta aplicaciones en Programación Orientada a Objetos. Este lenguaje tiene su fundamento en la transportabilidad, sin embargo, solo ocurre en aquellas funciones que han sido normalizadas (ANSI). Dicha estandarización, en el mejor de los casos, alcanza hasta un 40% de las funciones que comprende el compilador, el resto se sujeta a funciones que cada fabricante ha definido por su cuenta y que tienen cierta similitud a otros compiladores que existen en el mercado, pero no son compatibles.

La mayoría de los compiladores mantienen un esquema de compatibilidad hacia "*atrás*" (versiones anteriores), obligando con ésto a manejar criterios de compilación y generación de código bajo este esquema. El compilador generará código para un 8088 igual que para un 80486, con solo cambiar una bandera en la solicitud de compilación, sin embargo, esto no basta, sino que se hace necesaria una optimización del mismo para no limitar el rendimiento ("*Performance*") del microprocesador. No obstante, hasta el momento no existen aún, en la línea Intel, compiladores que optimicen suficientemente el código como para poder establecer un máximo rendimiento de la nueva generación de microprocesadores.

Como anteriormente se expuso, la característica de los nuevos microprocesadores *RISC* establece la posibilidad de ejecutar la mayoría de sus instrucciones en uno o dos ciclos de reloj y realizar una o más instrucciones en este mismo ciclo; dicho de otra manera, se tiene la posibilidad de paralelizar ciertas operaciones de alto nivel. Lo anterior implica la necesidad de que, al diseñar el Software, sea analizado desde un punto de vista en el que la arquitectura del microprocesador es óptima, esto es, que cada procedimiento pudiese ser paralelizable en mayor medida. En las gráficas que se muestran a continuación, se establece un análisis de los diversos tipos de microprocesadores, donde se puede observar la relación entre precio/rendimiento. Dicha evaluación se efectuó con Software realizado por la *Standard Performance Evaluation Corp.*, el cual mide las capacidades de rendimiento en operaciones de punto flotante (**SPECFP92**) y enteros (**SPECINT92**). Además, se usó el sistema operativo **UNIX** de Silicon Graphics y la versión mas reciente de **NT WINDOWS**. En algunos casos, como los microprocesadores R4400 y ALPHA, sus resultados fueron caracterizados por emulación, dado que algunas rutinas, principalmente de **NT WINDOWS,** solo son nativas en microprocesadores de INTEL Corp. En este caso la pérdida de rendimiento fue ajustada para la tabla comparativa **(1)**.





De lo anteriormente descrito, se puede expresar que el rendimiento de un diseño de Software se en-

cuentra definido por la siguiente expresión:

$$\mathbf{R}_{SW} = 1 - (\mathbf{L}_{AC} + \mathbf{L}_{CMP} + \mathbf{L}_{SO}).$$

donde $\mathbf{R}_{SW}$ se define como el rendimiento de diseño de Software.

$\mathbf{L}_{AC}$ = Las limitaciones de la arquitectura de la computadora donde va a ser ejecutada la aplicación,

$\mathbf{L}_{CMP}$ = La limitación del compilador en obtener el máximo rendimiento en la generación de código y

$\mathbf{L}_{SO}$ = Las limitaciones del sistema operativo para entregar y administrar los recursos requeridos por la aplicación.

El cálculo de rendimiento para una arquitectura de computadora dada se podrá determinar con el siguiente ejemplo. Sea:

$\mathbf{T}_{AC}$ = El tiempo total en segundos de una rutina completa, medida en base a los tiempos reportados por instrucción a ejecutar y

$\mathbf{T'}_{AC}$ = El tiempo total en segundos de una rutina completa ejecutada, medida en tiempo real.

$$\mathbf{R}_{AC} = 1 - (\mathbf{T'}_{AC} - \mathbf{T}_{AC}) / \mathbf{T}_{AC}$$

donde:

$$\mathbf{L}_{AC} = 1 - \mathbf{R}_{AC} = (\mathbf{T'}_{AC} - \mathbf{T}_{AC}) / \mathbf{T}_{AC}$$

El rendimiento $\mathbf{R}_{SW}$ del Software, independientemente de las variables anteriormente expuestas, se verá afectado por variables intrínsecas al Ingeniero diseñador de Software, como son:

- Metodología de diseño empleada.
- Modularidad.
- Dependencia de datos.
- Verificación de transferencias entre módulos.
- Criterios de Auditoría Informática a establecer en el sistema.
- Requerimiento de uso de rutinas de "*tiempo real*"
- Requerimiento de almacenamiento masivo.
- Velocidad de las transferencias en el almacenamiento de datos.
- Menú de interface.
- etc.

De la misma forma pero en sentido contrario, estas variables definen los criterios particulares del tipo de arquitectura de la computadora, compilador y sistema operativo que habrá de seleccionarse.

Bibliografía

[1] Child, J. *"Pentium price/performance clouded by Software issues"*, Computer Design, May 1993, pp 28-30.

[2] Intel Corp. *"Procesador Pentium, Descripción General de Rendimiento"*, Notas Técnicas, Versión 2, Agosto 1993.

[3] White, B. *"Programming Techniques for Software Development"*, Stanford University, ISBN 0-442-29187-6, Edit. Van Nostrand Reinhold, 1989.

OBSERVACIONES

Si te interesa contribuir con algún comentario o sugerencia, el Consejo Editorial de la Revista Polibits agradecerá utilices este espacio.

Y lo remitas a :

Centro de Investigación Tecnológica en Computación
Consejo Editorial Revista POLIBITS
Av. Té No. 950, Edificio de Graduados 2o. piso.
Colonia Granjas México. C.P. 08400
México D. F.
☎ 6-54-39-32, 6-49-50-36, 6-49-03-66 ext. 327. Fax : 6-57-74-53.

Si tu contribución es en forma de un artículo, favor de anexar los siguientes datos: Nombre del autor, Grado Académico e Institución de procedencia.