UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

instituto de astronomía
unam

Instituto de astronomía

Publicaciones Técnicas

“Reporte Técnico” **RT-2004-23**

TARJETA SECUENCIADORA PARA EL CONTROLADOR VERSATIL DE CCDs OAN.

S. Zazueta, E. Colorado.

TARJETA SECUECIADORAN PARA EL CONTROLADOR VERSATIL DE CCDs OAN.

Mayo de 2004.

SALVADOR ZAZUETA
ENRIQUE COLORADO ORTIZ

Departamento de Instrumentación Electrónica, Ensenada.
Instituto de Astronomía, OAN, UNAM.

TARJETA SECUECIADORAN PARA EL CONTROLADOR VERSATIL DE CCDs OAN.

Versión 1.0, Mayo de 2004.

Salvador Zazueta, Enrique Colorado.

Resumen.

El presente trabajo describe las características generales del controlador versátil de CCDs y describe el diseño electrónico de la tarjeta de fases y la programación asociada a la tarjeta.

CONTENIDO

1.- DESCRIPCION GENERAL DEL SISTEMA.

El objetivo de este proyecto es desarrollar un controlador versátil para circuitos integrados CCD de tipo científico.

Las principales características del sistema son:

- Interfaz Ethernet con protocolo de comunicación TCP/IP.
- 16 bits de resolución.
- Bajo ruido de lectura de la electrónica de adquisición.
- Secuencia de lectura definida por programación.
- Expandible hasta 24 fases de reloj.
- La configuración de voltajes de los relojes del CCD serán configurados por programación.
- Máxima frecuencia de muestreo (FM) de 300 Khz. La FM será programable.
- Los voltajes máximos de operación de las fases de reloj serán: +/- 15 volts.

Las características mencionadas nos permiten adecuar el sistema propuesto al control de los siguientes detectores:

- Familia SiTe 1kx1k, 2kx2k.
- Familia Thomson 1kx1k .
- Familia Kodak KAF0400 (768x512) hasta KAF 02300 2kx2k.

El sistema de adquisición de imágenes es controlado usando una computadora industrial con sistema operativo Linux a la cual se le conecta circuitería adicional para el control y

lectura del CCD. En la Figura 1, se muestra el esquema general del sistema, se utiliza otra computadora personal, la cual es la interfaz hacia el usuario donde se podrá visualizar y almacenar las imágenes obtenidas. Toda la operación del sistema será remota vía red Ethernet.

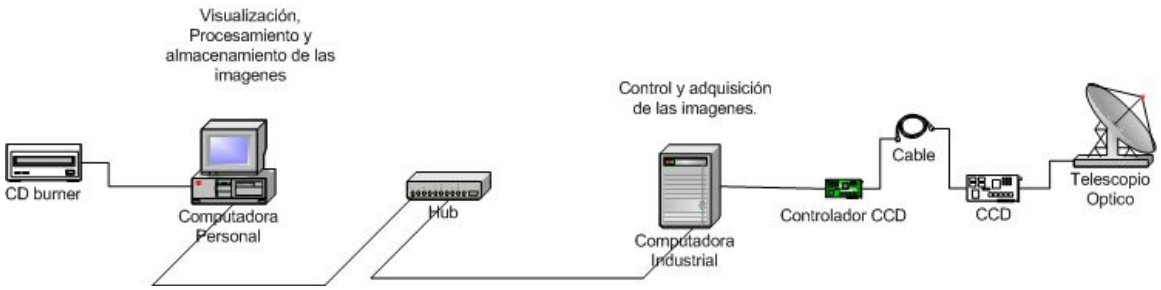


Figura 1.- Esquema general del sistema de adquisición.

2.- COMPONENTES DEL SISTEMA.

Para adquirir imágenes controlando el CCD, el sistema cuenta con 4 tarjetas con circuitería electrónica, unidas por un ducto común (*Euro Bus*) y controladas por una computadora de uso dedicado, como se muestra en la Figura 2.

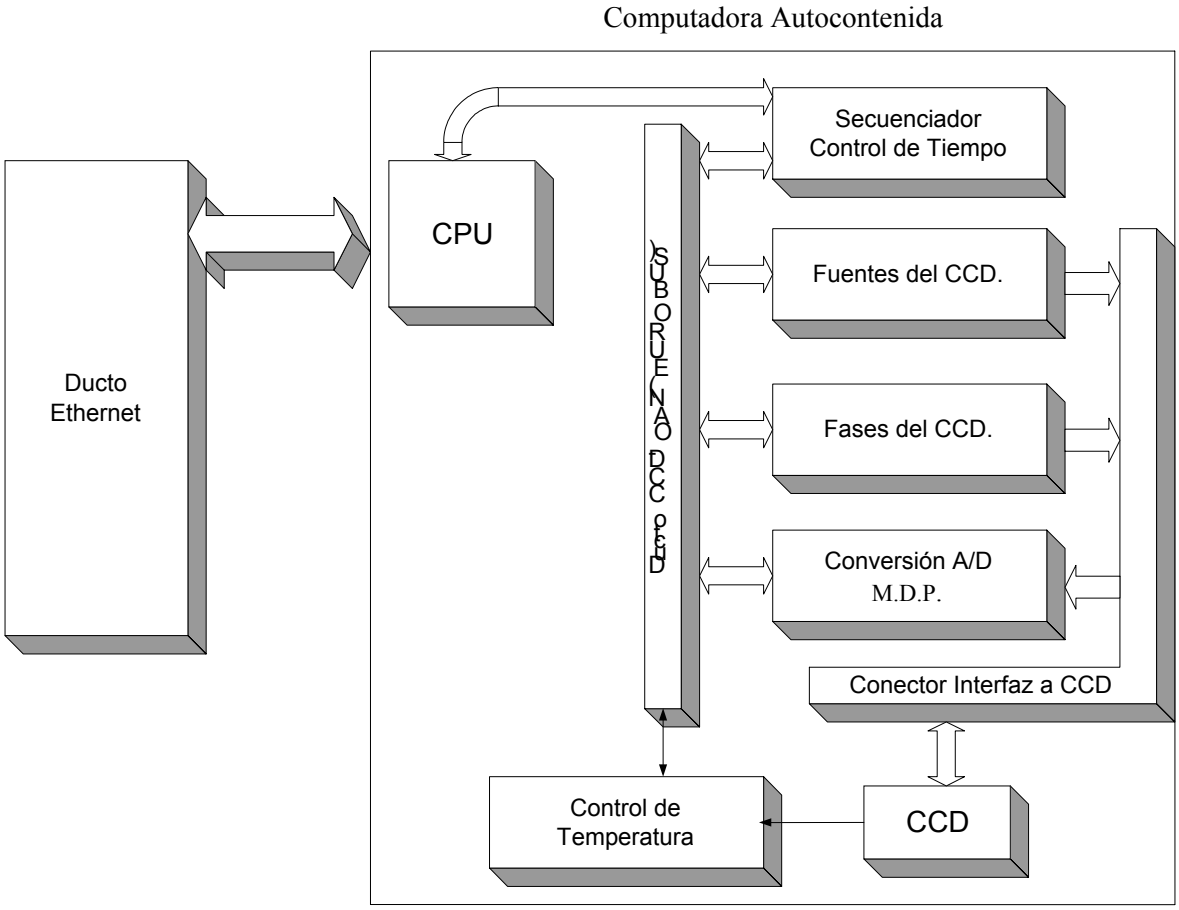


Figura 2.- Esquema general de la electrónica del controlador.

El CPU se encarga de la comunicación y del control en general del sistema de adquisición. Se utiliza una PC dedicada tipo industrial con sistema operativo Linux utilizando una memoria flash como disco duro.

- La tarjeta secuenciadora sirve como interfaz entre la PC y las demás tarjetas; es la tarjeta encargada de la generación de las señales TTL de control y temporización necesarias para la operación de las tarjetas de fases, fuentes y de conversión analógico a digital.

- La tarjeta de fuentes es la encargada de generar los voltajes de polarización del CCD que no estén variando, e.g. VDD, etc. Se puede generar hasta 8 voltajes, los cuales pueden ser programados independientemente para que su valor se encuentre en el intervalo de -30 a +30 volts. Se utiliza un convertidor digital a analógico de 12 bits con el cual obtenemos una resolución de 0.0146 volts.

- La tarjeta de fases se encarga de generar los relojes o fases del sistema. La tarjeta puede generar hasta 8 fases, es posible programar el voltaje de salida de cada fase para que su valor se encuentre en el intervalo de -15 a +15 volts con una resolución de 0.007 volts.

- La tarjeta del convertidor analógico a digital, es la encargada de convertir la señal del vídeo del CCD a un valor digital de 16 bits usando la técnica del muestreo doble pendiente para incrementar la precisión de la medición.

Cada una de las tarjetas tiene una dirección asignada y es posible añadir más tarjetas de fuentes, convertidores y/o fases si es necesario.

3.- FUNCIONAMIENTO DE LA TARJETA SECUENCIADORA.

Esta tarjeta contiene un micro-controlador, el cual es el encargado de recibir las instrucciones de la PC industrial, generar las secuencias necesarias para leer los pixeles del

CCD, programar los niveles de voltajes fijos de polarización del CCD, programar los niveles de voltaje de las fases del CCD y controlar la conversión analógica a digital.

La comunicación entre la PC y la tarjeta secuenciadora se lleva a cabo usando el puerto de interfaz IDE 0x1F0. El diagrama a bloques de la tarjeta secuenciadora se muestra en la Figura 3.

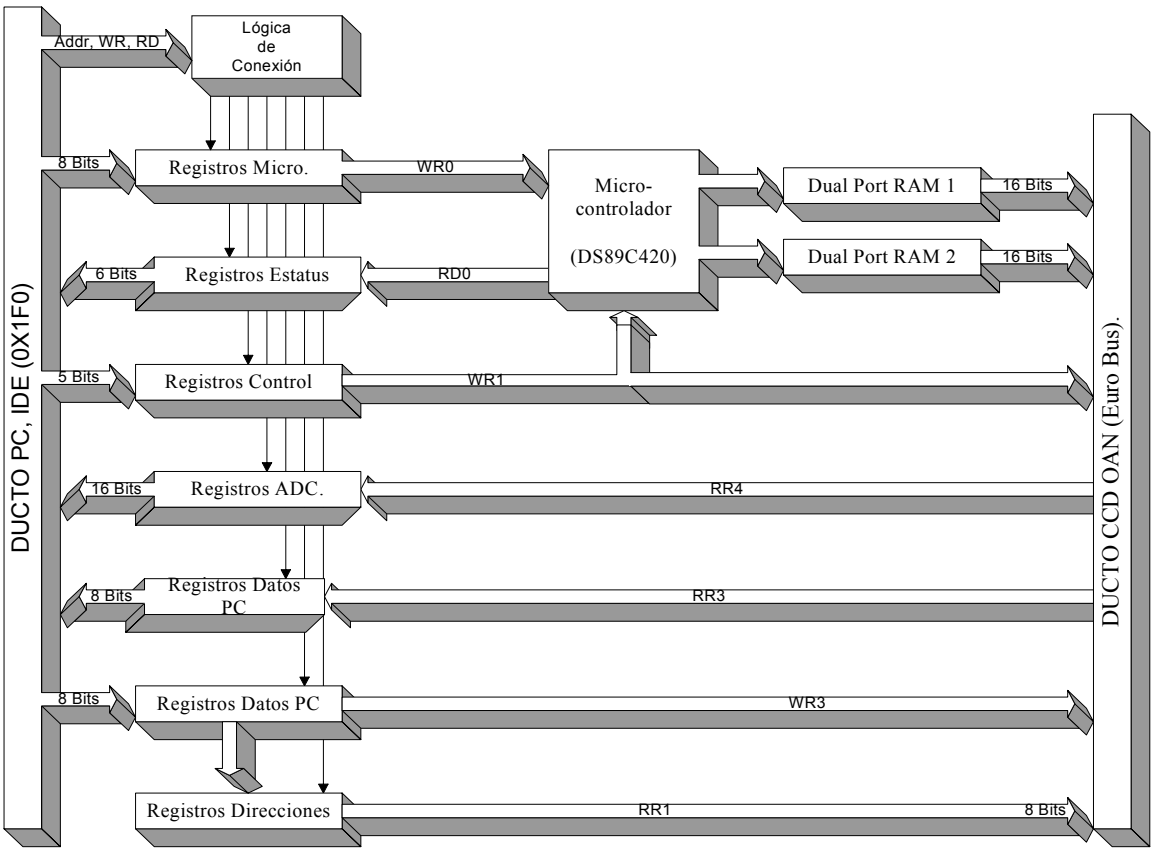


Figura 3.- Diagrama a bloques de la tarjeta secuenciadora.

La tarjeta tiene 2 modos de operación:

- Modo de instrucciones.
- Modo de secuencias.

- Modo de instrucciones:

Al inicio (Reset) del micro-controlador, este despierta en modo de espera de una instrucción de la PC.

Las instrucciones que espera el micro-controlador se enlistan a continuación:

INSTRUCCIÓN	CÓDIGO
Carga RAM	1
A Secuencias	2
Retardo mueve línea	3
Num. Pasos digitaliza píxel	4
Num. Pasos mueve píxel	5
Num. Pasos mueve línea	6
Retardo digitaliza píxel	7
Retardo mueve píxel	8
Num. Pasos mueve línea tipo 1	9
Num. Pasos mueve línea tipo 2	10
Num. Pasos mueve línea tipo 3	11
Mueve línea por siempre	12
Mueve y llena RAM por siempre	13

Tabla I.-Instrucciones de PC a tarjeta secuenciadora.

Los mensajes entre la PC y la tarjeta secuenciadora se componen de un número fijo de caracteres ASCII.

La longitud del mensaje es de 16 caracteres y están definidos por los siguientes campos:

NÚMERO DE BYTE	VALOR DEL BYTE	USO
1	#	Inicio del mensaje
2 – 15	A hasta P	Carga o paquete del mensaje (*Nota)
16	$	Fin de mensaje

Tabla II.-Formato del mensaje a secuenciadora.

*** Nota:** Los números son codificados en nibbles, de la siguiente forma:

NÚMERO	CÓDIGO	NÚMERO	CÓDIGO
0	A	8	I
1	B	9	J
2	C	10 (A HEX)	K
3	D	11 (BHEX)	L
4	E	12 (C HEX)	M
5	F	13 (D HEX)	N
6	G	14 (E HEX)	O
7	H	15 (F HEX)	P

Tabla III.- Codificación de los números del mensaje.

La secuencia que sigue la PC para mandar un byte al micro-controlador es la siguiente:

1. Se verifica que bit 0 (*status*) de registro de estado no se encuentre ocupado (0=OK, 1=Busy).
2. Se escribe el byte a mandar en el registro *micro* (R0).
3. Se sube *LEEM* (bit 0) del registro de control.
4. Se espera a que el bit *status* cambie a 1.
5. Se limpia el registro de datos.
6. Se baja el bit *LEEM.*
7. Se espera a que el bit *Status* baje.

Al finalizar de transmitir el mensaje (16 bytes) el micro-controlador pondrá el bit 2, *EXTRA*, del registro de estado en 1, si el mensaje mandado fue valido, si hubo un problema o error, el estado será 0.

En el apéndice A se describe detalladamente las funciones de la tabla I correspondiente a la tarjeta secuenciadora.

- Modo de secuencias:

Las memorias bi-puerto son utilizadas para programar y almacenar las secuencias de: digitaliza píxel, mueve píxel y mueve línea. Cada secuencia tendrá un máximo de 32 pasos.

Las localidades en RAM donde se almacenan las secuencias son las siguientes:

TIPO DE SECUENCIA	LOCALIDAD EN RAM
Digitaliza píxel	0 - 31
Mueve píxel	32 – 63
Mueve línea	64 - 91
Mueve línea tipo 1	96 - 127
Mueve línea tipo 2	128 - 159
Mueve línea tipo 3	160 - 191

Tabla IV.-Localidades en RAM de las secuencias.

Para que el micro-controlador genere las secuencias, la PC debe de escribir el tipo de secuencia deseada en el registro *R0*, después se debe mandar la instrucción 2 (a secuencias).

En la tabla V, se presenta la asignación de cada secuencia y su bit correspondiente.

TIPO DE SECUENCIA	BIT (REGISTRO 0)
Digitaliza píxel	0
Mueve píxel	1

Mueve línea	2
Mueve línea tipo 1	3
Mueve línea tipo 2	4
Mueve línea tipo 3	5

Tabla V.-Identificación de los Bits de secuencia.

Para terminar el modo de secuencias y regresar al modo de instrucciones se debe seguir los siguientes pasos:

1. Mandar a ejecutar secuencia mueve píxel.
2. Esperar a que termine la secuencia.
3. Poner todos los bits del registro a cero.
4. Bajar el bit *LEEM* para que el micro-controlador vuelva a leer los registros.
5. El micro-controlador se pone en espera de una instrucción.

El programa del microcontrolador fue escrito en el lenguaje C y se utilizo un compilador para generar el código en ensamblador del 8051. El código se presenta el en apéndice G y su diagrama de flujo en el apéndice F.

4.-REFERENCIAS BIBLIOGRAFÍCAS.

Referencias computacionales:

1. Richard Stones Neil Matthew
   Beginning linux programming
   Segunda Edición.
   Wrox

2. Alessandro Rubini.
   Linux Device Drivers.
   O´Reilly
   Primera edición, 1998.

3. Bao Ha, Tina Nguyen
   Linux Slackware Unleashed
   Sams
   Primera edición 1999.

Referencias de electrónica:

4. Holst Gerald.
   CCD arrays Cameras and Displays.
   JDC Publishing.
   1996.

5. Bair Stephan.
   CCD Imaging Systems.
   Burr Brown.

6. Kristian,J, Blouke, M.
   Charge Coupled Devices in Astronomy.
   Scientific American.
   Octubre, 1992.

7. Ronald J. Tocci, Neal S. Widmer.
   Digital systems: principles and applications.

Octava edición.
Prentice Hall.

8. James Janesick.
Electronics Design of High Performance Digital CCD Cameras.
The international Society for optical engineering.
Short Course Notes.
Enero, 2000.

9. James Janesick, Blouke, M.
Past and Present of Scientific CCD's.
Optics and Photonics News.
Abril, 1995.

10. Steve B. Howel.
Handbook of CCD Astronomy.
Cambridge University Press.
Primera edición, 2002.

11. James R. Janesick.
Scientific Charge Coupled Devices.
SPIE Press.
2001.

12. Thomas H. Ebben.
Low Noise Electronics
The international society for optical engineering (SPIE).
Short Course Notes.

13. **The ccd image sensor.**
Thomson composants militaires ET spatiux

Referencias de Internet:

14. Apogee Instruments, Inc.

http://www.ccd.com/ccdu.html

15. Scientific imaging technologies, Inc.

http://www.site-inc.com/tutorial1.htm

"An introduction to Scientific imaging Charge-Couple Devices"
http://www.site-inc.com/product-1024x1024-ccd.htm

http://www.site-inc.com/pdf/003datsh.pdf

CD Interactivos

16. Rouper Scientific CD.

CCD Imaging Fundamentals.

17. James, Janesick

Introduction to charged coupled devices

APÉNDICE A.

INSTRUCCIONES DE LA TARJETA SECUENCIADORA.

#	Instrucción	Uso
01	Carga RAM	Cargar secuencia a RAM.
Mensaje:		
#		Inicio
01		Numero de instrucción
NN		Dirección en RAM
NNNNNNNNN		Secuencia a almacenar
$		Fin del mensaje

Tabla VI.- Instrucción carga a RAM.

#	Instrucción	Uso
02	A secuencias	Pone al micro-controlador en modo de secuencias
Mensaje:		
#		Inicio
02		Numero de instrucción
xx		No se usa
xxxxxxxxxx		No se usa
$		Fin del mensaje

Tabla VII.- Instrucción a secuencias.

#	Instrucción	Uso
03	Retardo mueve línea	Retardo de mueve línea
Mensaje:		
#		Inicio
03		Numero de instrucción
xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del retardo
$		Fin del mensaje

Tabla VIII.- Instrucción Retardo mueve línea.

#	Instrucción	Uso
04	Numero de pasos DigPix	Numero de pasos de la secuencia de digitaliza píxel.
Mensaje:		
#		Inicio
04		Numero de instrucción
xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del número de pasos.
$		Fin del mensaje

Tabla IX.- Instrucción Numero de pasos DigPix.

#	Instrucción	Uso
05	Numero de pasos MuevePIX	Numero de pasos de la secuencia mueve píxel.
Mensaje:		
#		Inicio
05		Numero de instrucción
xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del número de pasos.
$		Fin del mensaje

Tabla X.- Instrucción numero de pasos MuevePIX.

#	Instrucción	Uso
06	Numero de pasos Mueve línea	Numero de pasos de la secuencia mueve línea.
Mensaje:		
#		Inicio
06		Numero de instrucción
xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del número de pasos.
$		Fin del mensaje

Tabla XI.- Instrucción numero de pasos mueve línea.

#	Instrucción	Uso
07	Retardo digitaliza píxel	Valor de retardo de digitaliza píxel
Mensaje:		
#		Inicio
07		Numero de instrucción
xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del retardo.
$		Fin del mensaje

Tabla XII.- Instrucción retardo digitaliza píxel.

#	Instrucción	Uso
08	Retardo mueve Píxel	Valor de retardo de mueve píxel
Mensaje:		
#		Inicio
08		Numero de instrucción
Xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del retardo.
$		Fin del mensaje

Tabla XIII.- Instrucción retardo mueve píxel.

#	Instrucción	Uso
09	Numero de pasos mueve línea 1	Numero de pasos de la secuencia mueve línea tipo 1.
Mensaje:		
#		Inicio
09		Numero de instrucción
xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del número de pasos.
$		Fin del mensaje

Tabla XIV.- Instrucción para fijar el numero de pasos de mueve línea 1.

#	Instrucción	Uso
10	Numero de pasos mueve línea 2	Numero de pasos de la secuencia mueve línea tipo 2.
Mensaje:		
#		Inicio
10		Numero de instrucción
xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del número de pasos.
$		Fin del mensaje

Tabla XV.- Instrucción para fijar el numero de pasos de mueve línea 2.

#	Instrucción	Uso
11	Numero de pasos mueve línea 3	Numero de pasos de la secuencia mueve línea tipo 3.
Mensaje:		
#		Inicio
11		Numero de instrucción
xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del número de pasos.
$		Fin del mensaje

Tabla XVI.- Instrucción para fijar el numero de pasos de mueve línea 3.

#	Instrucción	Uso
12	Mueve línea por siempre	Ciclo infinito de mueve línea.
Mensaje:		
#		Inicio
12		Numero de instrucción
xx		No se usa
Xxxxxxxxxx		No se usa
$		Fin del mensaje

Tabla XVII.- Instrucción de ciclo infinito de secuencia mueve línea.

#	Instrucción	Uso
13	Mueve línea y llena RAM	Ciclo infinito de mueve línea y llena la RAM con un dato.
Mensaje:		
#		Inicio
13		Numero de instrucción
xx		No se usa
xxxxxxxxNN		El byte menos significativo contiene el valor del número a grabar en toda la RAM.
$		Fin del mensaje

Tabla XVIII.- Instrucción de ciclo infinito de mueve línea y llena RAM.

APÉNDICE B.
PROTOCOLO DE COMUNICACIÓN DE LAS TARJETAS DEL SISTEMA.

El protocolo para escribir caracteres a los microcontroladores de las tarjetas de fuentes, fases y secuenciadora es el mismo. A continuación se describen los pasos para escribir un carácter:

1. La computadora de control (IPC) escribe el carácter en el registro de datos de la tarjeta deseada.
2. La IPC escribe un “1” en el bit menos significativo del registro de estado.
3. La IPC lee el dato del registro de estado y espera a que el bit menos significativo de este registro tenga el valor de “1”.
4. La IPC escribe un “0” en el bit menos significativo del registro de estado.
5. La IPC lee el dato del registro de estado y espera a que el bit menos significativo de este registro tenga el valor de “0”.
6. Este proceso se repite para cada carácter a enviar.

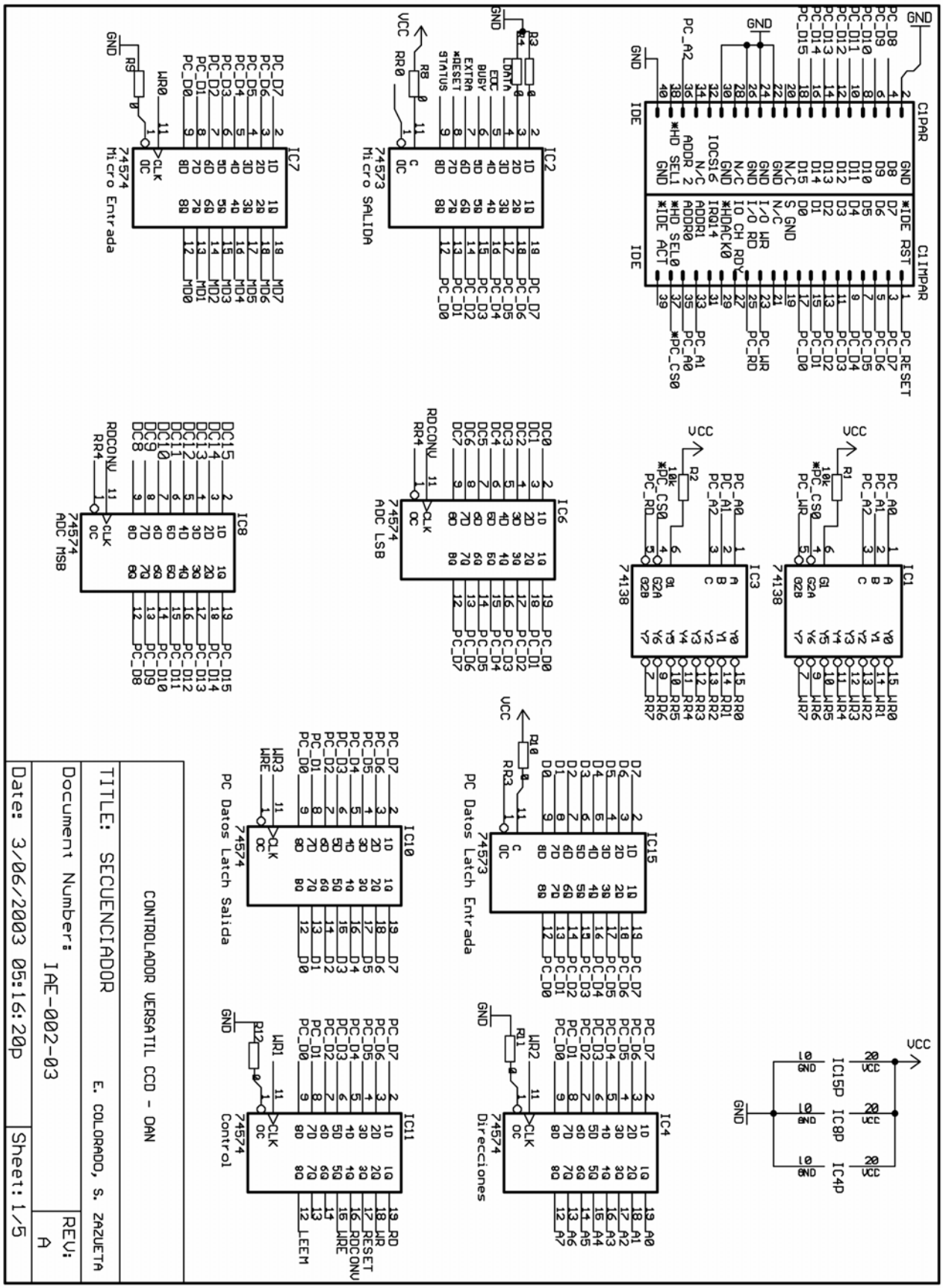
APÉNDICE C.
CÁLCULO DEL CHECK SUM.

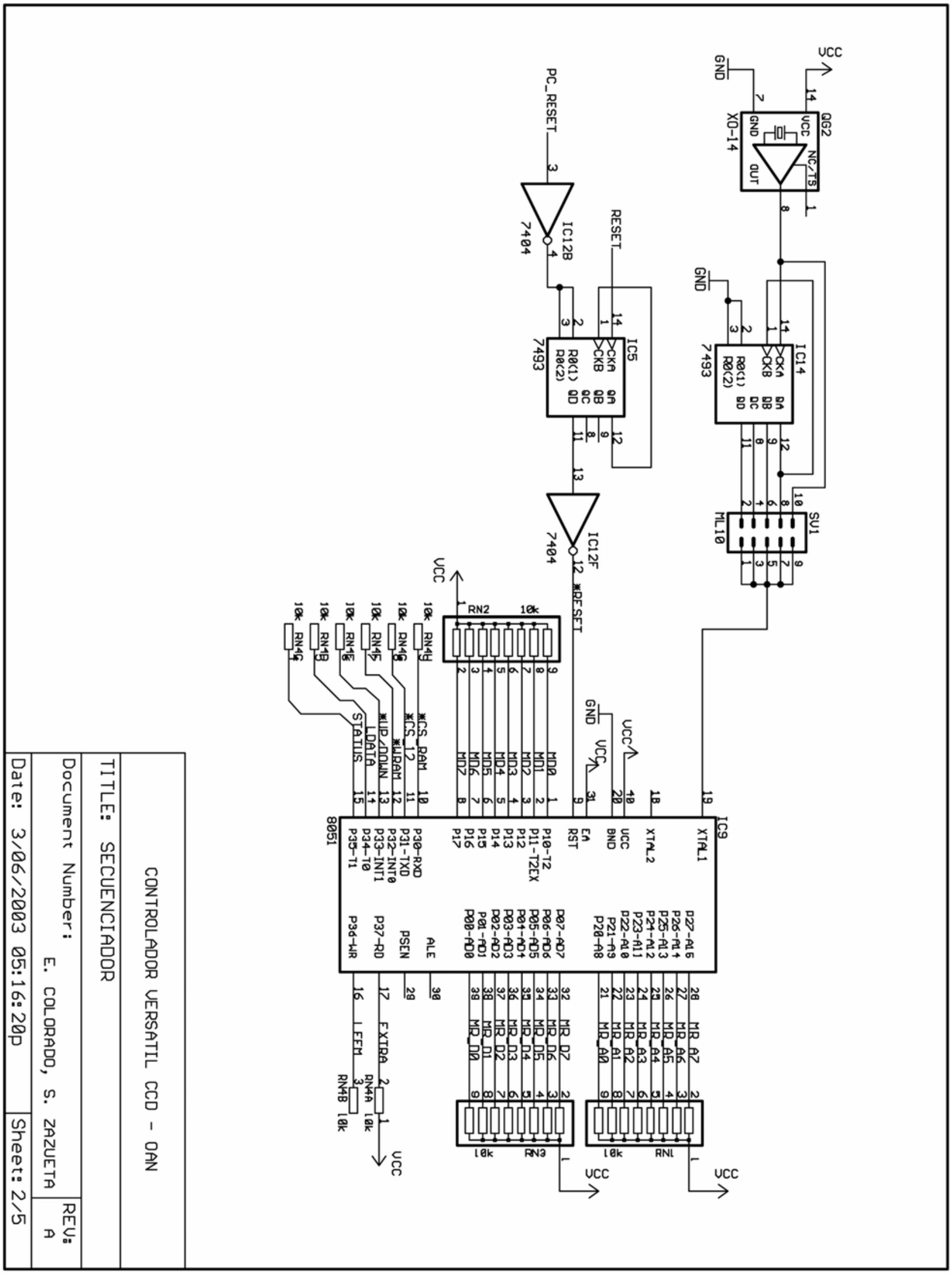
El cálculo del “check sum” utilizado para comunizarnos con las tarjetas del sistema se realiza de la siguiente manera:

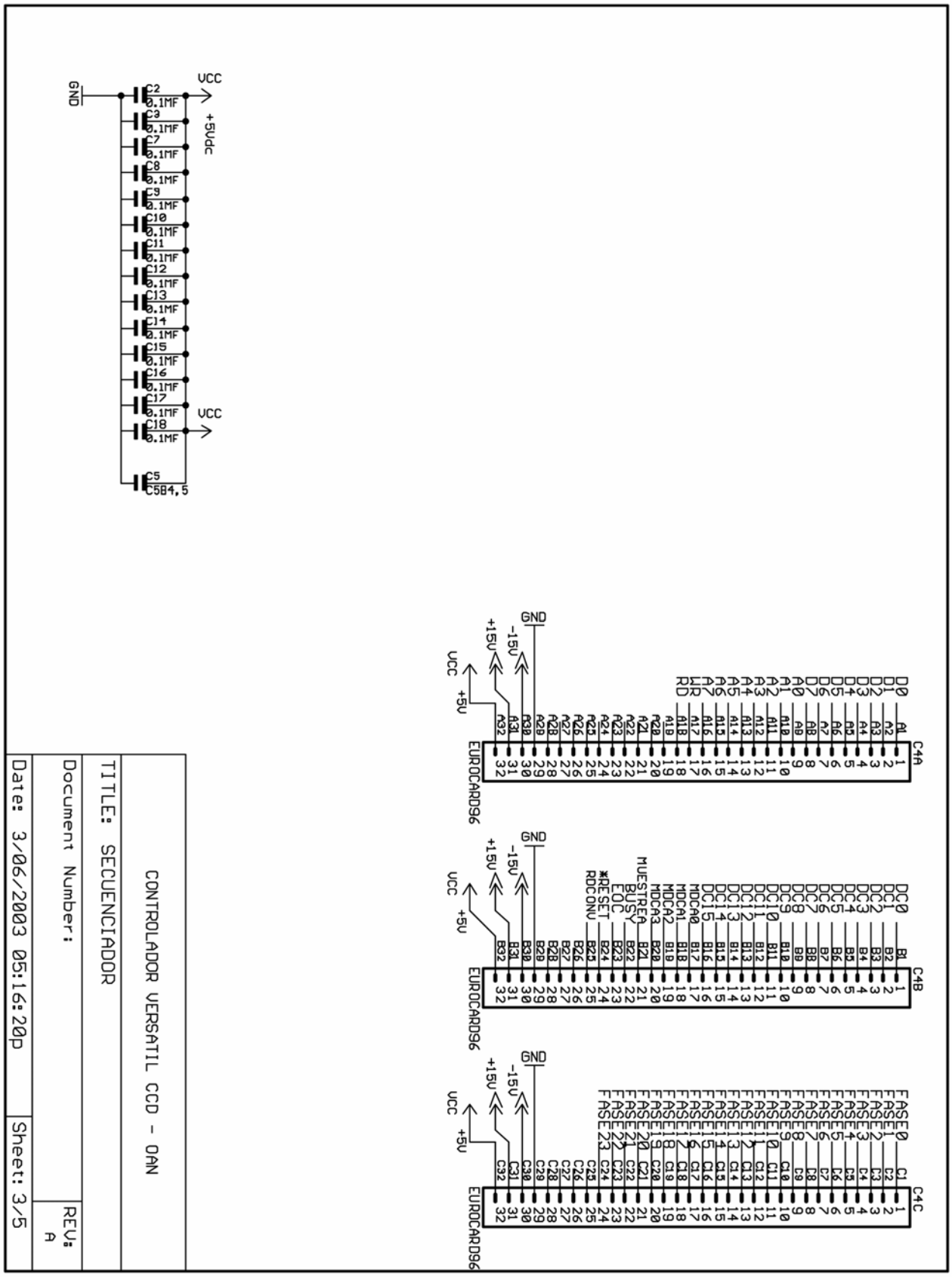
- Se realiza la suma de todos los caracteres sin tomar en cuenta el carácter de inicio “#” ni el carácter final “$”.
- Al resultado de la suma se le hace una negación BIT a BIT.
- Se aplica una operación AND al resultado con el valor 0FH.
- El resultado se codifica de acuerdo al código mostrado en la tabla 21.

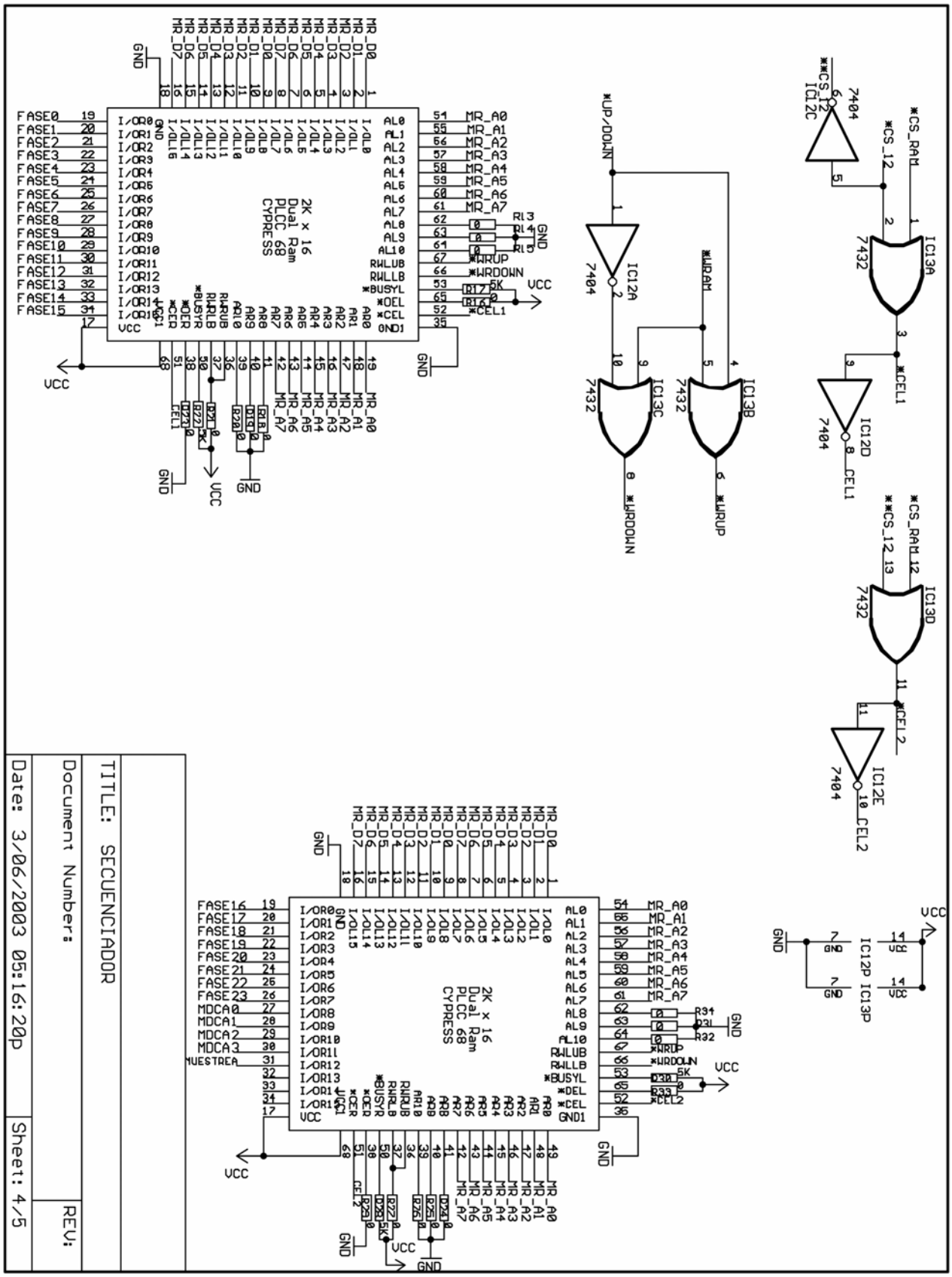
APÉNDICE D.
DIAGRAMA ELECTRÓNICO

A continuación se presenta las cuatro hojas del diagrama electrónica de la tarjeta secuenciadora.


CONTROLADOR VERSATIL CCD - OAN
TITLE: SECUENCIADOR
E. COLORADO, S. ZAZUETA
Document Number: IAE-002-03
REV: A
Date: 3/06/2003 05:16:20p
Sheet: 1/5


CONTROLADOR VERSATIL CCD - OAN
TITLE: SECUENCIADOR
Document Number:
E. COLORADO, S. ZAZUETA
REV: A
Date: 3/06/2003 05:16:20p
Sheet: 2/5
VCC
GND
QG2
XO-14
PC_RESET
RESET
IC12B
7404
IC12F
*RESET
IC5
7493
IC14
SU1
ML10
RN2
10k
RN1
RN3
IC9
8051
XTAL1
XTAL2
VCC
GND
EA
RST
P10-T2
P11-T2EX
P12
P13
P14
P15
P16
P17
P30-RXD
P31-TXD
P32-INT0
P33-INT1
P34-T0
P35-T1
P27-A15
P26-A14
P25-A13
P24-A12
P23-A11
P22-A10
P21-A9
P20-A8
P07-AD7
P06-AD6
P05-AD5
P04-AD4
P03-AD3
P02-AD2
P01-AD1
P00-AD0
ALE
PSEN
P37-RD
P36-WR
MD0
MD1
MD2
MD3
MD4
MD5
MD6
MD7
*CS_RAM
*CS_12
*WRAM
*UP/DOWN
LDATA
STATUS
MR_A0
MR_A1
MR_A2
MR_A3
MR_A4
MR_A5
MR_A6
MR_A7
MR_D0
MR_D1
MR_D2
MR_D3
MR_D4
MR_D5
MR_D6
MR_D7
EXTRA
LFFM
RN4A 10k
RN4B 10k


CONTROLADOR VERSATIL CCD - OAN
TITLE: SECUENCIADOR
Document Number:
REV: A
Date: 3/06/2003 05:16:20p
Sheet: 3/5
EUROCARD96
C4A
C4B
C4C
GND
VCC
+5V
+15V
-15V
+5Vdc
C2 0.1MF
C3 0.1MF
C7 0.1MF
C8 0.1MF
C9 0.1MF
C10 0.1MF
C11 0.1MF
C12 0.1MF
C13 0.1MF
C14 0.1MF
C15 0.1MF
C16 0.1MF
C17 0.1MF
C18 0.1MF
C5 C584,5


TITLE: SECUENCIADOR
Document Number:
REV:
Date: 3/06/2003 05:16:20p
Sheet: 4/5
2K x 16
Dual Ram
PLCC 68
CYPRESS
IC13A
IC13B
IC13C
IC13D
IC12A
IC12C
IC12D
IC12E
IC12P
IC13P
7432
7404
*CS_RAM
*CS_12
*CEL1
*CEL2
*WRAM
*WRUP
*WRDOWN
*UP/DOWN
VCC
GND

APÉNDICE E.

DIAGRAMA DE LA TARJETA DEL CIRCUITO IMPRESO.

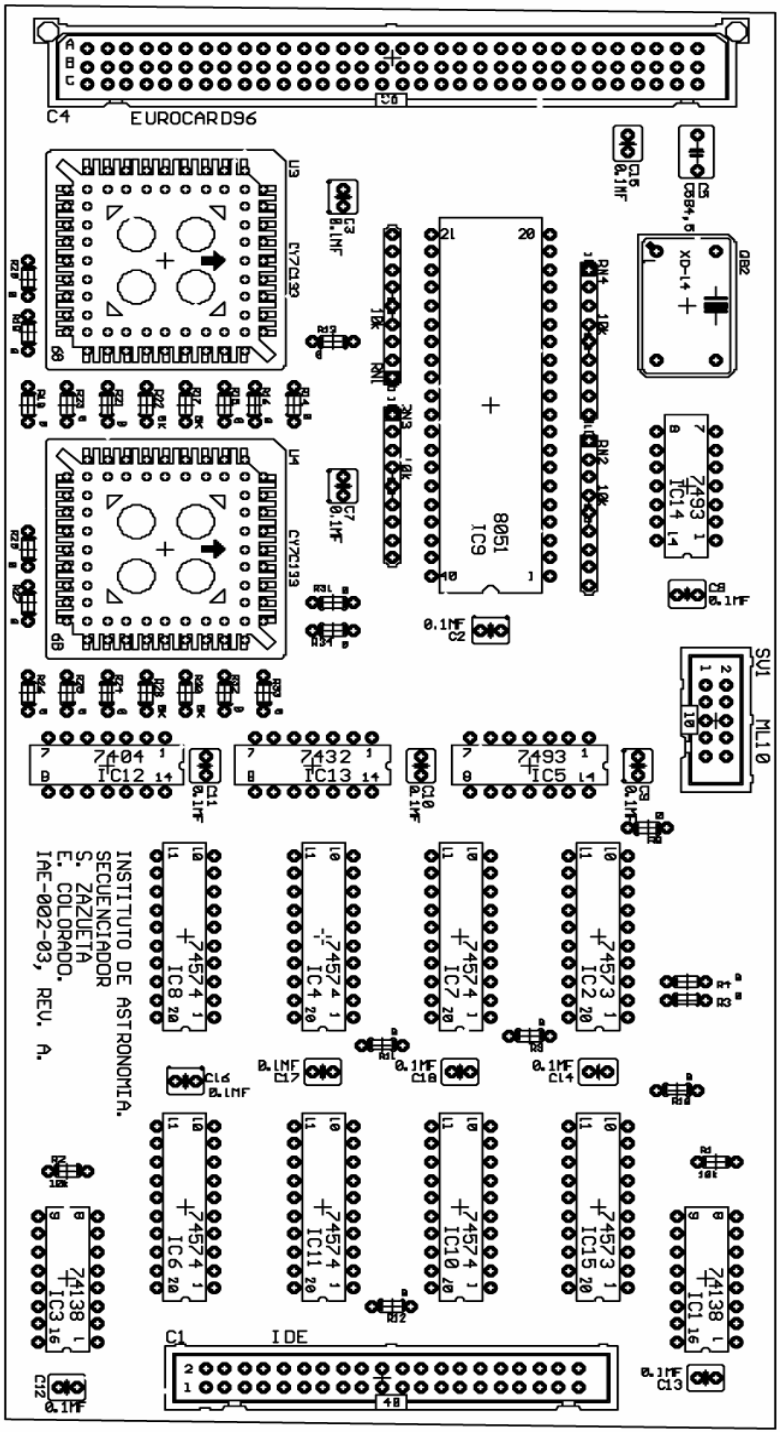


Figura E.1.- Diagrama de la posición de componentes.

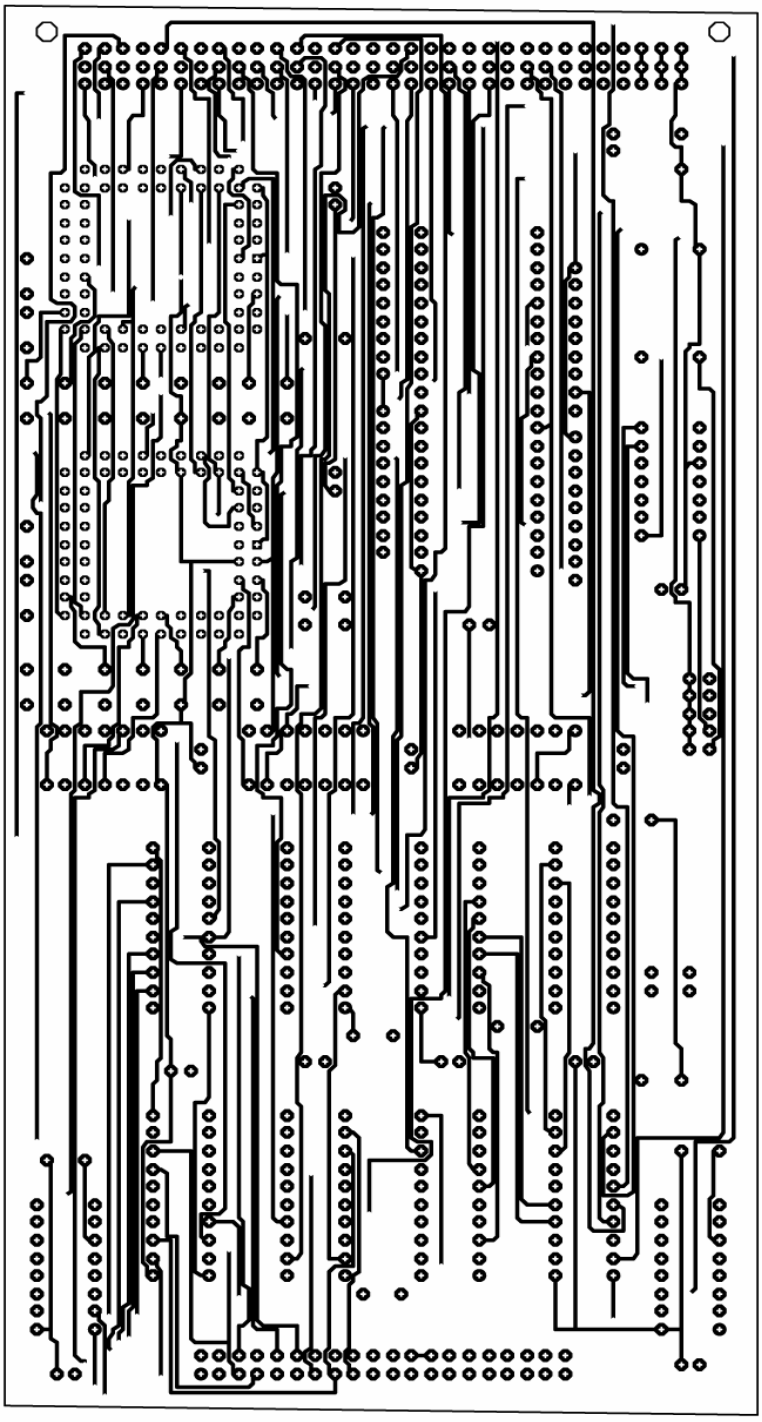


Figura E.2.- Capa superior del circuito impreso.

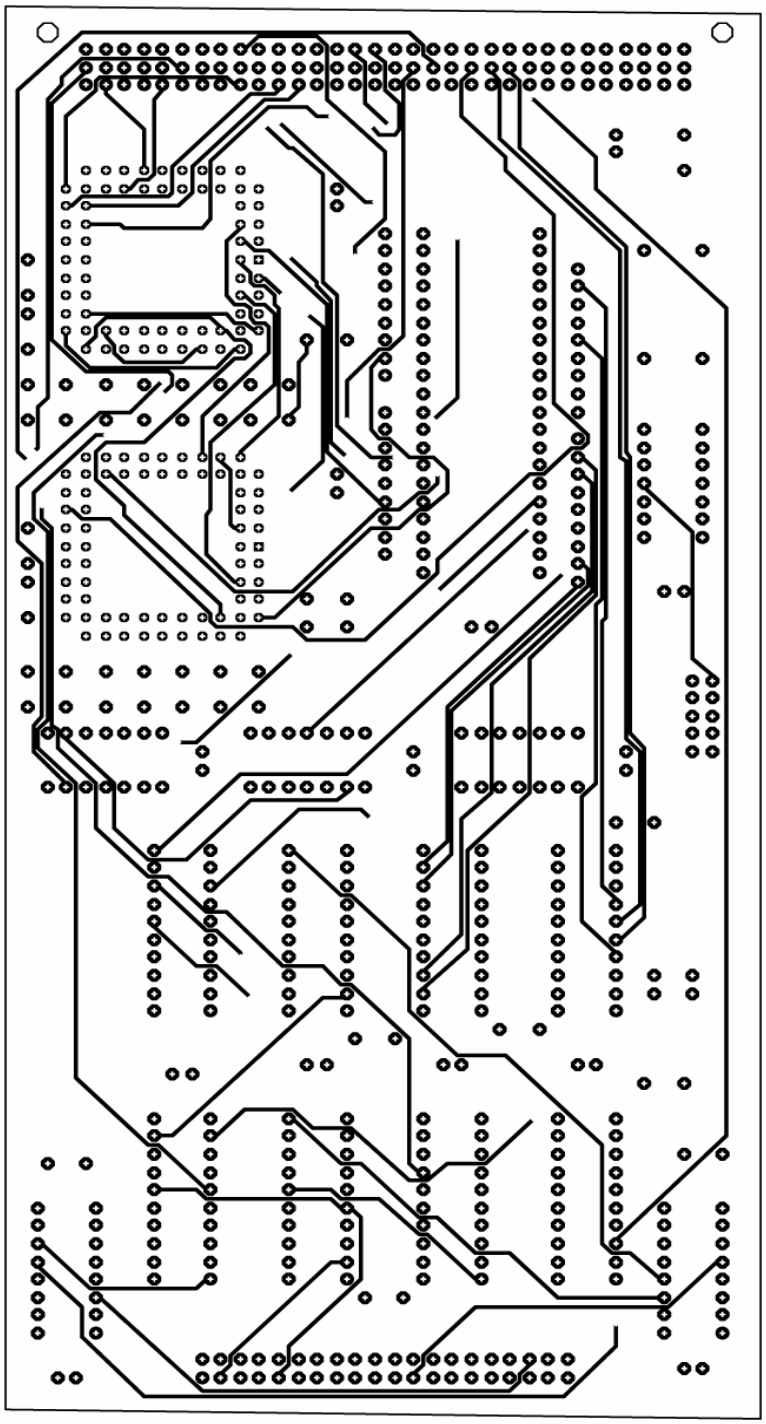


Figura E.3.- Capa 2 del circuito impreso.

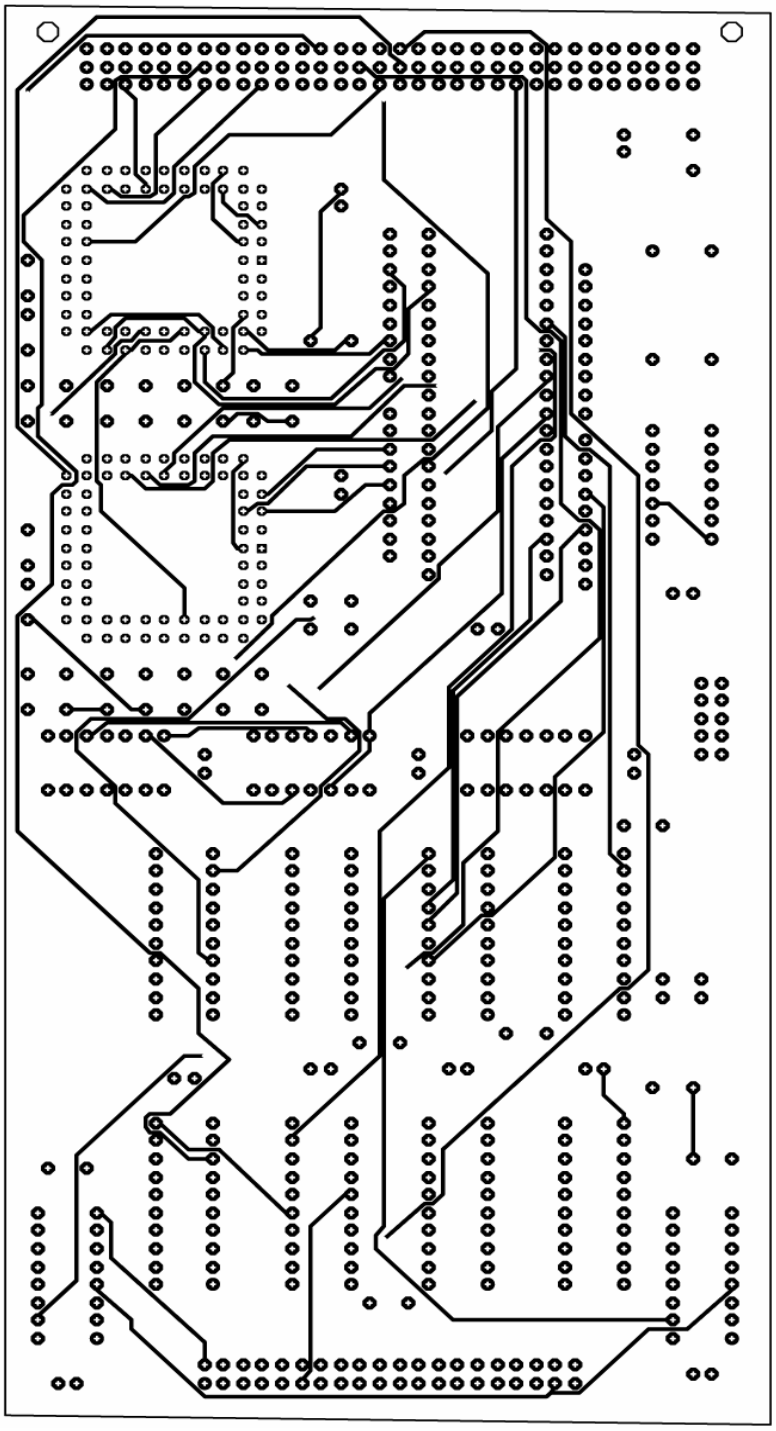


Figura E.4.- Capa 3 del circuito impreso.

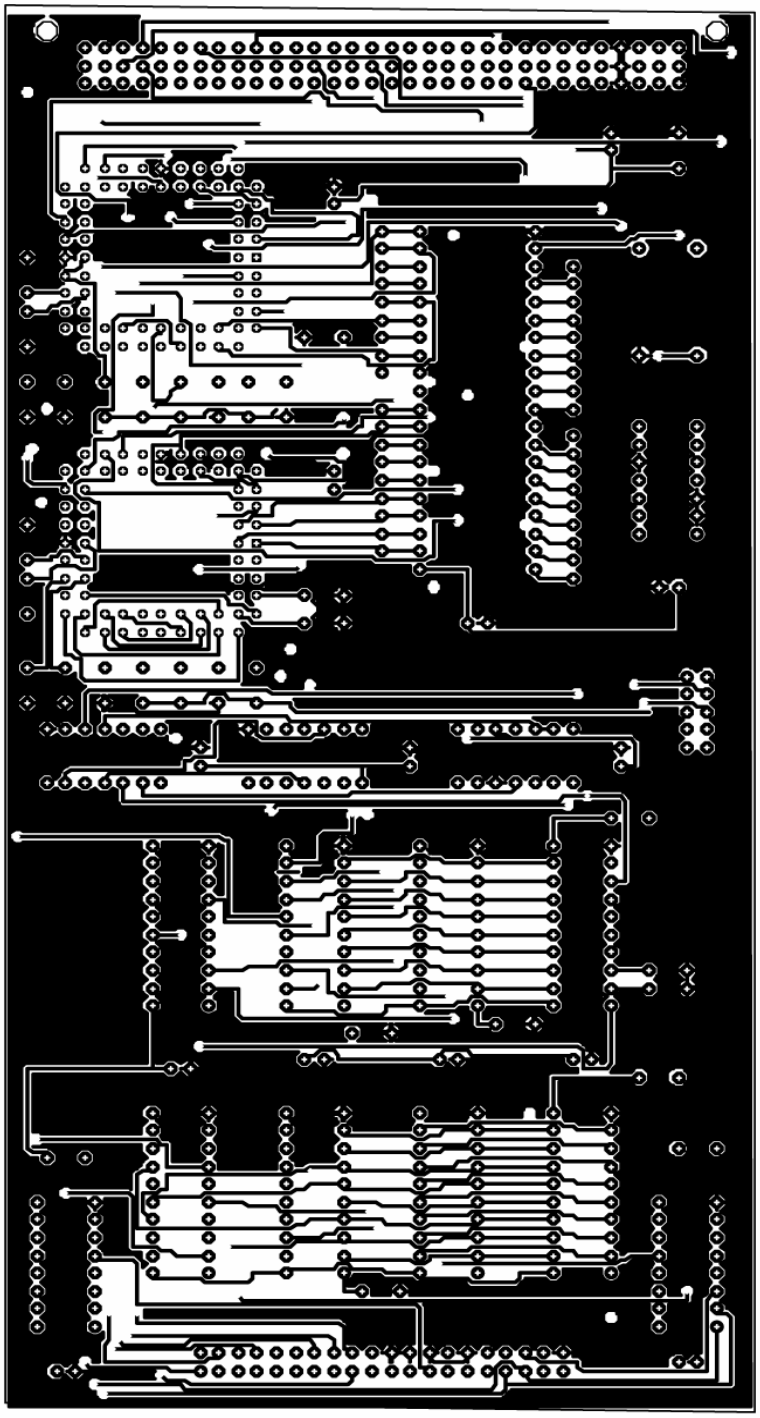
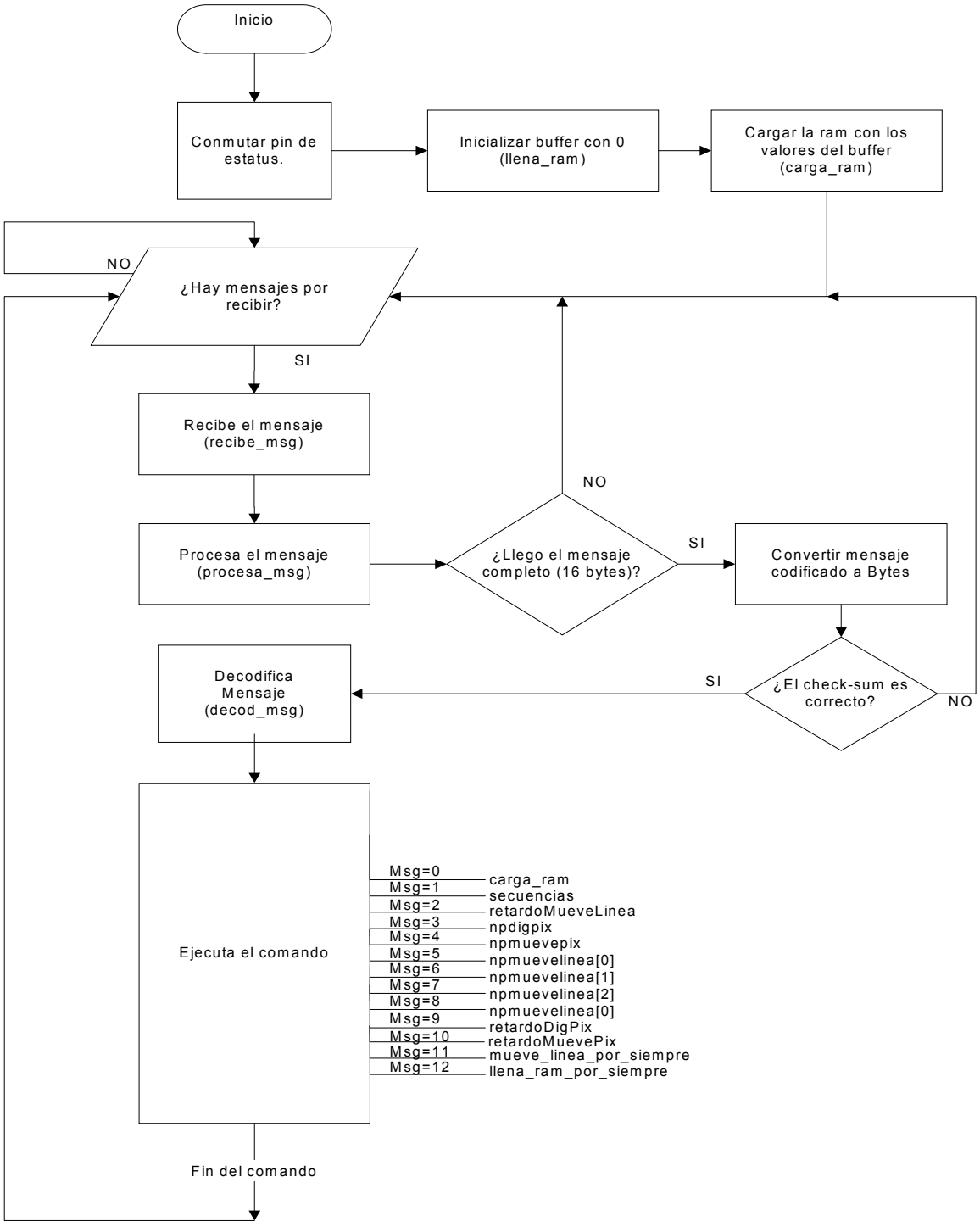


Figura E.5.- Capa inferior del circuito impreso.

APÉNDICE F.

DIAGRAMA DE FLUJO DEL PROGRAMA DEL MICROCONTROLADOR.



APÉNDICE G.

LISTADO DEL PROGRAMA DEL MICROCONTROLADOR.

```
/*
  Programa de la tarjeta secuenciadora.

  Comunicacion con la tarjeta secuenciadora.
*/

#include <8051.h>

#define BIT_LEE_PC  P3_6
#define BYTE_PC     P1
#define BIT_BUSY    P3_5
#define DIR_RAM     P2
#define DATO_RAM    P0
#define CS_RAM       P3_0
#define CS_1o2       P3_1
#define CS_UPP_BYTE  P3_3
#define WR_DATO      P3_2
#define PIN_STATUS   P3_5
#define PIN_MSG_OK   P3_7
#define DIGPIX         P1_0
#define MUEVE_PIX      P1_1
#define MUEVE_LINEA    P1_2
#define MUEVE_LINEA_1  P1_3
#define MUEVE_LINEA_2  P1_4
#define MUEVE_LINEA_3  P1_5
#define BYTES_POR_MSG    14
/* Las instrucciones que acepta la secuenciadora */
#define INS_CARGA_RAM              1
#define INS_A_SECUENCIAS           2
#define INS_RETARDO_MUEVE_LINEA    3
#define INS_NUM_PASOS_DIGPIX       4
#define INS_NUM_PASOS_MUEVEPIX     5
#define INS_NP_MUEVELINEA          6
#define INS_RETARDO_DIGPIX         7
#define INS_RETARDO_MUEVEPIX       8
#define INS_NP_MUEVELINEA_1        9
#define INS_NP_MUEVELINEA_2        10
#define INS_NP_MUEVELINEA_3        11
#define INS_MUEVE_LINEA_POR_SIEMPRE 12
#define INS_MUEVE_LLENA_RAM_POR_SIEMPRE 13
#define DIR_RAM_MUEVEPIX       0
#define DIR_RAM_DIGPIX        32
#define DIR_RAM_MUEVELINEA    64
#define DIR_RAM_MUEVELINEA_1  96
#define DIR_RAM_MUEVELINEA_2  128
#define DIR_RAM_MUEVELINEA_3  160
#define INICIO_TX      '#'
#define FIN_TX         '$'

/* Prototipos de fns. */

char espera_byte_de_pc(void);
char procesa_msg(char c);
void carga_ram(void);
void secuencias(void);
```

```
void decod_msg(void);
char recibe_msg(void);
void llena_ram(void);
void mueve_linea(void);

/* Vars. Globales */
static unsigned char buf[BYTES_POR_MSG];  /* Buffer para los msgs */
static unsigned char npdigpix = 31;
static unsigned char npmuevepix = 31;
static unsigned char npmuevelinea[4] ={ 31,31,31,31};
static unsigned char retardoMueveLinea = 100;
static unsigned char retardoMuevePix = 0;
static unsigned char retardoDigPix = 0;

void main()
{
  char nbytes=0;
  char c;

  /* al inicio espera la instruccion de salir del prog. de
     cargar las rams.
  */
  CS_RAM = 1;
  BIT_BUSY = 0;
  WR_DATO = 1;

  /* un togle del PIN_STATUS p. indicar que el micro esta bien, solo se
     checa con el osciloscopio.
  */
  PIN_MSG_OK = 0;
  PIN_STATUS = 0;
  PIN_STATUS = 1;
  llena_ram();
  PIN_STATUS = 0;

  /*
  while(1){
    mueve_linea();
  }
  */



  nbytes=0;
  /* loop infinito */
  while(1){
    nbytes = recibe_msg();
    c = procesa_msg( nbytes );
    if( c == 1 ){
      decod_msg();
    }
  }
}

char espera_byte_de_pc(void)
{
  char b;

  /* espera a que suba el bit de LEE_PC */
  while( BIT_LEE_PC == 0 ){}
  b = BYTE_PC;
  /* sube el bit de busy y espera q' la PC baje su bit de LEE_PC */
```

```
  BIT_BUSY = 1;
  while( BIT_LEE_PC == 1 ){}
  /* Baja el bit de BUSY y regresa el byte que leyo del puerto */
  BIT_BUSY = 0;
  return(b);
}


char procesa_msg(char c)
{
  char i;
  char n;
  unsigned char cksum;

  /* Los msgs estan definidos por:
   # - inicio
   INSTRUCCION - 2 chars (1 byte)
   DIRECCION   - 2 chars (lugar donde se debe cargar el op)
   OPERANDO    - 8 chars 1 long
   CKSUM       - 2 (suma negada de los asciis )
   $ - fin
   En total - 16 bytes por paquete.
  */

  if( c != BYTES_POR_MSG ){
    return(0);  /* No llego el msg completo */
  }
  cksum = 0;
  for(i=0; i<BYTES_POR_MSG-2; i++){
    cksum = cksum + buf[i];
    buf[i] = buf[i] - 'A';        /* convierte todo a nibbles */
    buf[i] = buf[i] & 0xF;
  }
  buf[12] = (buf[12] - 'A') & 0x0F;
  buf[13] = (buf[13] - 'A') & 0x0F;

  for(i=0; i<BYTES_POR_MSG; i=i+2 ){
    n = (buf[i]<<4)  ;        /* convierte todo a bytes */
    n = n |  buf[i+1];
    buf[i] = n;
    buf[i+1] = 0;
  }
  buf[12] = ~buf[12];
  if( cksum == buf[12] ){
    PIN_MSG_OK = 1;
    return(1);
  }
  return(0);
}


void mueve_linea_por_siempre(void)
{
  for(;;){
    BIT_BUSY = 1;
    mueve_linea();
  }
}

void llena_ram_por_siempre(void)
{
  for(;;){
```

```
    BIT_BUSY = 1;
    llena_ram();
  }
}

void decod_msg(void)
{

  switch( buf[0] ){
  case INS_CARGA_RAM:
    carga_ram();
    break;
  case INS_A_SECUENCIAS:
    secuencias();
    break;
  case INS_RETARDO_MUEVE_LINEA:
    retardoMueveLinea = buf[10];
    break;
  case INS_NUM_PASOS_DIGPIX:
    npdigpix = buf[10] & 0x1F;   /* En el LSB del msg */
    break;
  case  INS_NUM_PASOS_MUEVEPIX:
    npmuevepix = buf[10] & 0x1F; /* No mas de 31 */
    break;
  case INS_NP_MUEVELINEA:
    npmuevelinea[0] = buf[10] & 0x1F; /* No mas de 31 */
    break;
  case INS_NP_MUEVELINEA_1:
    npmuevelinea[1] = buf[10] & 0x1F; /* No mas de 31 */
    break;
  case INS_NP_MUEVELINEA_2:
    npmuevelinea[2] = buf[10] & 0x1F; /* No mas de 31 */
    break;
  case INS_NP_MUEVELINEA_3:
    npmuevelinea[3] = buf[10] & 0x1F; /* No mas de 31 */
    break;
  case INS_RETARDO_DIGPIX :
    retardoDigPix =  buf[10] & 0x1F;
    break;
  case INS_RETARDO_MUEVEPIX:
    retardoMuevePix =  buf[10] & 0x1F;
    break;
  case INS_MUEVE_LINEA_POR_SIEMPRE:
    mueve_linea_por_siempre();
    break;
  case INS_MUEVE_LLENA_RAM_POR_SIEMPRE:
    llena_ram_por_siempre();
    break;
  }
}

void duerme_1u(void)
{
  unsigned char uc;

  uc = 20;

  while( uc != 0 ){
    uc = uc-1;
  }
}
```

```
void carga_ram(void)
{
  unsigned char ccc;


  WR_DATO = 1;
  ccc = buf[2];
  DIR_RAM = ccc;
  CS_1o2 = 1;          /* Chip 2 */
  CS_RAM = 0;         /* hab ram */

  CS_UPP_BYTE = 1;
  DIR_RAM = buf[2];
  DATO_RAM = buf[6];
  WR_DATO = 0;
  duerme_1u();
  WR_DATO = 1;

  CS_UPP_BYTE = 0;
  DATO_RAM = buf[4];  /* MSB  del long */
  WR_DATO = 0;
  duerme_1u();
  WR_DATO = 1;
  CS_RAM = 1;         /* deshab ram */
  CS_1o2 = 0;          /* Chip 1 */
  CS_RAM = 0;          /* hab. ram */
  CS_UPP_BYTE = 0;    /* El byte mas sig, */
  DIR_RAM = buf[2];
  DATO_RAM = buf[8];
  WR_DATO = 0;
  duerme_1u();
  WR_DATO = 1;

  CS_UPP_BYTE = 1;
  DIR_RAM = buf[2];
  DATO_RAM = buf[10]; /* EL LSB del long */
  WR_DATO = 0;
  duerme_1u();
  WR_DATO = 1;

  duerme_1u();
  CS_RAM = 1;         /* deshab. la RAM */
}

void mueve_pix(void)
{
  char c;
  char dir=DIR_RAM_MUEVEPIX;

  /* BIT_BUSY = 0; */
  for(c=0; c < npmuevepix; c++){
    DIR_RAM = dir;
    dir ++;
  }
  BIT_BUSY = 0;

}
```

```
Void dig_pix(void)
{

  /* char c;
  char dir=DIR_RAM_DIGPIX;
  */

  /*  BIT_BUSY = 1; */

  /* esta rut. tarda 133 us a 12 mhz. con los 32 pasos de la
     secuencia.
 */
_asm
  mov r2,_npdigpix
  /*  inc r2  */
  mov acc,#DIR_RAM_DIGPIX
$l1:
  mov P2,A
  inc A
  djnz r2,$l1

  mov r2, _retardoDigPix
  mov r3,A
  clr A
  add A,r2
  jz  $finPix
  inc r3
  mov P2,r3
$l2:
  djnz r2,$l2

$finPix:
_endasm ;

/* Esta rutina tarda  525 us a 12 mhz */
/*
  for(c=0; c < npdigpix; c++){
    DIR_RAM = dir;
    dir ++;
  }
*/

  BIT_BUSY = 0;
}


void mueve_linea(void)
{
  unsigned char c;
  unsigned char dir = DIR_RAM_MUEVELINEA;
  unsigned char retardo = retardoMueveLinea;


  /*  BIT_BUSY = 1; */
  for(c=0; c < npmuevelinea[0]; c++){
    DIR_RAM = dir;
    dir ++;

    retardo =  retardoMueveLinea;
    while( retardo != 0 ){
      retardo --;
```

```
    }

  }
  BIT_BUSY = 0;
}

void mueve_linea_1(void)
{
  unsigned char c;
  unsigned char dir = DIR_RAM_MUEVELINEA_1;
  unsigned char retardo = retardoMueveLinea;


  /*  BIT_BUSY = 1; */
  for(c=0; c < npmuevelinea[1]; c++){
    DIR_RAM = dir;
    dir ++;

    retardo =  retardoMueveLinea;
    while( retardo != 0 ){
      retardo --;
    }

  }
  BIT_BUSY = 0;
}


void mueve_linea_2(void)
{
  unsigned char c;
  unsigned char dir = DIR_RAM_MUEVELINEA_2;
  unsigned char retardo = retardoMueveLinea;


  /*  BIT_BUSY = 1; */
  for(c=0; c < npmuevelinea[2]; c++){
    DIR_RAM = dir;
    dir ++;

    retardo =  retardoMueveLinea;
    while( retardo != 0 ){
      retardo --;
    }

  }
  BIT_BUSY = 0;
}


void mueve_linea_3(void)
{
  unsigned char c;
  unsigned char dir = DIR_RAM_MUEVELINEA_3;
  unsigned char retardo = retardoMueveLinea;


  /*  BIT_BUSY = 1; */
  for(c=0; c < npmuevelinea[3]; c++){
    DIR_RAM = dir;
    dir ++;
```

```
    retardo =  retardoMueveLinea;
    while( retardo != 0 ){
      retardo --;
    }

  }
  BIT_BUSY = 0;
}


/* P. salir de la rutina de secuencias se debe generar lo sig.
   1 - se pone el bit de LEE_PC ( P3 )
   2 - se pone el bit de MUEVE_PIX
   3 - se baja el bit de MUEVE_PIX
   4 - se baja el bit de LEE_PC

   La PC puede verificar que se esta o no en el lazo de secuencias por
   medio del PIN_STATUS.
*/



void secuencias(void)
{
  /* El pin de status se levanta cuando se esta en la parte
     de secuencias.
  */
  PIN_STATUS = 1;
  BIT_BUSY = 1;
  while(1){

    if( DIGPIX == 1 ){
      dig_pix();
      /* espera a q' baje el bit de dig pix la PC */
      while( DIGPIX == 1 ){}
      BIT_BUSY = 1;
      continue;
    }

    if( MUEVE_PIX == 1 ){
      mueve_pix();
      /* Checa si regresar de esta fn al main */
      if( BIT_LEE_PC == 1 ){
       break;
      }
      /* espera a q' baje el bit de mueve pix la PC */
      while( MUEVE_PIX == 1){}
      BIT_BUSY = 1;
      continue;
    }
    if( MUEVE_LINEA == 1) {
      mueve_linea();
      while( MUEVE_LINEA == 1){}
      BIT_BUSY = 1;
    }

    if( MUEVE_LINEA_1 == 1) {
      mueve_linea_1();
      while( MUEVE_LINEA_1 == 1){}
      BIT_BUSY = 1;
    }
```

```
    if( MUEVE_LINEA_2 == 1) {
      mueve_linea_2();
      while( MUEVE_LINEA_2 == 1){}
      BIT_BUSY = 1;
    }

    if( MUEVE_LINEA_3 == 1) {
      mueve_linea_3();
      while( MUEVE_LINEA_3 == 1){}
      BIT_BUSY = 1;
    }


  } /* fin del while(1) */
  PIN_STATUS = 0;
  /* Esta aca hasta q' la pc baje su bit */
  while ( BIT_LEE_PC == 1 ){}
}


char recibe_msg(void)
{
  char nbytes = 0;
  char c;


  while (1){
    /* Se mantiene en este lazo hasta q' le llega el inicio de msg. */
    while( 1 ){
      c = espera_byte_de_pc();
      if( c == INICIO_TX ){
       nbytes = 0;
       PIN_MSG_OK = 0; /* baja el pin del cksum con el inicio de car */
       break;
      }
    }

    while(1){
      c = espera_byte_de_pc();
      if( c == FIN_TX ){
       return( nbytes );
      }
      if( (nbytes < BYTES_POR_MSG) &&
         (c >= 'A') &&
         (c <= 'P') ){
       buf[nbytes] = c;
       nbytes++;
      } else {
       /* Hubo un error en la transmision */
       nbytes = 0;
       break;
      }
    }
  }
  return(0);
}

void llena_ram(void)
{
  unsigned char c;
```

```
  for(c = 0; c<255; c++){
    buf[2] = c;
    buf[4] = 0x0 ; /* MSB */
    buf[6] = 0x0 ;
    buf[8] = 0x0;
    buf[10] = 0x0; /* LSB */
    carga_ram();
  }
}
```

APÉNDICE G.
HOJA DE DATOS.

A continuación se presentaran ligas a las hojas de información técnica de algunos componentes electrónicos.

- Microcontrolador DS89C420.
  http://www.astrosen.unam.mx/~colorado/data/DS89C420.pdf
- Ciprés dual port RAM.
  http://www.astrosen.unam.mx/~colorado/data/dual-port-ram.pdf
- 74LS574.
  http://www.astrosen.unam.mx/~colorado/data/74LS574.pdf
- 74LS573.
  http://www.astrosen.unam.mx/~colorado/data/74LS573.pdf
- 74LS93.
  http://www.astrosen.unam.mx/~colorado/data/74ls93.pdf

Comité Editorial de Publicaciones Técnicas
Instituto de Astronomía
UNAM

M.C. Urania Ceseña
Dr. Carlos Chavarria
M.C. Francisco Murillo

Observatorio Astronómico Nacional
Km. 103 Carretera Tijuana-Ensenada
22860 Ensenada B.C.
editorial@astrosen.unam.mx