ESCUELA POLITECNICA NACIONAL
FACULTAD DE INGENIERIA ELECTRICA

DISEÑO Y CONSTRUCCION DE UN SISTEMA DE CONTROL DE TIEMPOS DE RECORRIDO EN LOS BUSES DE TRANSPORTE PUBLICO

LISTADO DE PROGRAMAS EN ASEMBLER Y QBASIC

A N E X O

MARIO A. DELGADO R.

QUITO NOVIEMBRE DE 2000

LISTADO DEL PROGRAMA EN ASEMBLER USADO EN EL MICROCONTROLADOR DS2250T.

```
;*****************************************************************************

;***     Programa para control de sistema de control horario

;***     Septiembre de 1999

;***     Programa para microcontrolador DS2250T

;***     Compilado con CYS8051

;***     Mario Delgado

;*****************************************************************************

;       El sistema controla:

;       1.-     1 display de dos líneas de 16 caracteres cada una

;       2.-     1 Lector de tarjetas con banda magnética track 2

;       3.-     1 Interface RS-232 infrarroja
;
;*****************************************************************************
;><><><><><><><><><><.....><><><><><><><><><><><>
;*****************************************************************************
;       A C T U A L I Z A C I O N E S
;
;
;*****************************************************************************
;><><><><><><><><><><.....><><><><><><><><><><><>
;*****************************************************************************
;
;
;       AREA DESTINADA A ETIQUETAS
;
;
```

```
port_dis        equ     P2              ;Bus de datos general de display
ena             equ     P0.5            ;Control ENABLE display
r_w             equ     P0.6            ;Control READ/WRITE display
rs              equ     P0.7            ;Control RESET display

tarjeta_si      equ     INT1            ;Indica que hay tarjeta en el lector
lector_ena      equ     P1.0            ;Puerto para habilitar el lector

;
;
;
;       ETIQUETAS CONSTANTES
;
MCON            equ     0C6H
TA              equ     0C7H
stack           equ     2FH
num_car         equ     16              ;Numero de caracteres que controla el
                                        ;monitor del display
kte_mcon        equ     38H             ;selecciona dirección de inicio de RAM en
                                        ;1800H y 32 K total en el chip
                                        ;No mas de 4096 en programa
car_ret         equ     13
;
;
;*******************************************************************************
;
;       ETIQUETAS DE LOCALIZACION DE RAM
;
clave_dig       equ     5EH
clave_low       equ     5FH
clave_med       equ     60H
clave_med1      equ     61H
clave_med2      equ     62H
clave_med3      equ     63H
clave_hig       equ     64H
seg_dat         equ     65H             ;dirección de registro de segundos
min_dat         equ     66H             ;dirección de registro de minutos
hora_dat        equ     67H             ;dirección de registro de horas
dias_dat        equ     68H             ;dirección de registro de día de la semana
```

```
diam_dat        equ     69H             ;dirección de registro de día del mes
mese_dat        equ     6AH             ;dirección de registro del mes
anio_dat        equ     6BH             ;dirección de registro del año
data_dis        equ     6CH
dato_leido      equ     6DH
control         equ     6EH


inicio_d        equ     70H                     ;Area de RAM para manejo de display
; hasta         7FH

;
;*****************************************************************************
;
;*****************************************************************************;
;       Dirección de inicio del programa con HARDWARE - RESET
        org     0H
h_reset:        ljmp    programa
;
;
;*****************************************************************************
;
;       Area de entrada de interrupciones generales
;
        org     13H                     ;Origen de interrupción del lector
        ljmp    ver_lector              ;Va a rutina de atención al lector

        org     23H
        ljmp    serial                  ;interrupción de comunicación serial
;
;
;*****************************************************************************
;
;


        org     100H
programa:
        mov     SP,#stack               ;Inicializa el SP con STACK(30)
        mov     TA,#0AAH                ;Inicialización para acceso temporizado
        mov     TA,#55H
```

```
        mov     PCON,#0                 ;pone en reset todos los parámetros de
                                        ;control

        mov     MCON,#kte_mcon
        mov     IE,#0                   ;Inicializa control de interrupciones
                                        ;Se utiliza un cristal de 7.3728 MHz
        mov     TMOD,#21H               ;Inicializa los timers 1 para la generación
                                        ;de baud rate y 0 para los retardos
                                        ;Debe aumentarse 1 al resultado final para
                                        ;obtener el complemento a 2
;;;;;   mov     TH1,#0FCH               ;       9600 BAUD
;;;;;   mov     TH1,#0F8H               ;       4800 BAUD
        mov     TH1,#0F0H               ;       2400 BAUD
;;;;;   mov     TH1,#0E0H               ;       1200 BAUD
;;;;;   mov     TH1,#80H                ;        300 BAUD

        orl     PCON,#88H               ;Pone en doble baud_rate
                                        ;       NO PARITY
                                        ;       1 stop bit
        mov     SCON,#50H               ;Pone la puerta serial en modo 0
        mov     TCON,#40H

        lcall   init_display

        lcall   escuela_dis             ;Muestra en display  ESCUELA POLITECNICA
        lcall   display_lin1
        lcall   del_20mili
        lcall   nacional_dis            ;Muestra en display     NACIONAL
        lcall   display_lin2
        lcall   del_1seg
        setb    EA
;
;*******************************************************************************
;
main:
        mov     SP,#stack               ;Inicializa el SP con STACK(30)
        setb    ES                      ;habilita la interrupción serial
        setb    EX1                     ;habilita la interrupción externa 1
        setb    lector_ena
        mov     control,#'C'
```

```
        lcall   reloj                   ;Va a rutina de lectura del reloj
        lcall   display_lin1
        lcall   del_20mili
        lcall   estacion_dis
        lcall   display_lin2
        lcall   del_1seg
        lcall   reloj                   ;Va a rutina de lectura del reloj
        mov     R0,#(inicio_d+12)
        mov     @R0,#' '
        lcall   display_lin1
        lcall   del_1seg
        ljmp    main
;
;*****************************************************************************
;
ver_lector:
        clr     lector_ena              ;Habilita lectura de datos desde el lector
        mov     control,#'T'            ;Pone bandera de lectura en tarjeta

        push    ACC
        mov     R5,#100                 ;Rutina de espera de caracteres desde el
lazo_dec1:                              ;lector. Si no viene dato sale
        mov     A,#100
lazo_dec:
        dec     A
        jb      RI,si_car               ;Si recibe un caracter por la puerta serial
        jnz     lazo_dec                ;va a si_car
        djnz    R5,lazo_dec1
        pop     ACC
        reti
;
;*****************************************************************************
;
si_car: pop     ACC

serial:
        clr     es
        clr     ex1
        mov     A,control
        cjne    A,#'C',con_tarjeta
```

```
con_computador:
        dec     SP
        dec     SP
        mov     DPTR,#computador
        push    DPL
        push    DPH
        reti
;
;*****************************************************************
;
con_tarjeta:
        dec     SP
        dec     SP
        mov     DPTR,#ver_tarjeta
        push    DPL
        push    DPH
        reti

ver_tarjeta:
        clr     ES
        clr     EX1
;
;******************
;
;       Borra el espacio del display
        mov     R0,#inicio_d
        mov     R6,#16
lazo_borra:
        mov     @R0,#' '
        inc     R0
        djnz    R6,lazo_borra

;
;*****************************************************************************
;
;       Rutina para leer la tarjeta
;
tarjeta:
        mov     R0,#inicio_d
        clr     RI
```

```
        mov     A,SBUF                  ;Los datos leídos se ponen en el buffer del
                                        ;display
        mov     @R0,A                   ;Carga el primer caracter
        inc     R0
        mov     R5,#1                   ;Registro para contar cantidad de carac-
                                        ;teres recibidos
        mov     R6,#0
        mov     R4,#2
lazo_tar:       jb      RI,OTRO_CAR     ;Espera recibir un caracter
        mov     A,#30                   ;Inicia un lazo de 255 x 30 veces
lazo_esp:       dec     A               ;para terminar automáticamente la
        jnz     lazo_esp                ;lectura de la tarjeta
        inc     R6                      ;No se puede trabajar con TARJETA_SI
        cjne    R6,#255,lazo_tar        ;porque la señal de la finalización
                                        ;de la tarjeta sucede mucho antes de que
                                        ;el control del lector termine de transmitir
                                        ;todos los datos por la línea señal
        djnz    R4,lazo_tar

ljmp    verificar_tarjeta               ;Termina la lectura de la tarjeta
;
;*****************************************************************************
;
otro_car:       clr     RI              ;Carga los datos en el área de display
        mov     A,SBUF                  ;sucesivamente
        mov     @R0,A
        inc     R0
        inc     R5                      ;Suma registro de control de cantidad de
        ljmp    lazo_tar                ;caracteres
;
;
;*********************************************
;
;       Verifica que no se hayan recibido ni mas ni menos que 11
;       caracteres, para evitar que puedan ser leídas tarjetas de
;       crédito o cualquier otro tipo de tarjeta
;
verificar_tarjeta:
        mov     A,R5
        cjne    A,#11,noes_tarjeta
```

```
        lcall   conv_dis_cla
        lcall   correcta_dis
        lcall   display_lin2
lcall   grabar                          ;Va a rutina de guardar transacción
        lcall   del_1seg
        ljmp    main
;
;*****************************************************************
;
;       Regresa a rutina principal
;
;
noes_tarjeta:
        lcall   display_lin2
        lcall   del_5seg
        ljmp    main
;
;
;***********************************************
;
;       Rutina para convertir datos ascii en BCD comprimido de los datos
;       leídos desde la tarjeta
;
conv_dis_cla:
        mov     R0,#inicio_d
        mov     A,@R0
        anl     A,#0FH
        swap    A
        mov     clave_hig,A
        inc     R0
        mov     A,@R0
        anl     A,#0FH
        add     A,clave_hig
        mov     clave_hig,A
        inc     R0

        mov     A,@R0
        anl     A,#0FH
        swap    A
        mov     clave_med1,A
```

```
inc     R0
mov     A,@R0
anl     A,#0FH
add     A,clave_med1
mov     clave_med1,A
inc     R0

mov     A,@R0
anl     A,#0FH
swap    A
mov     clave_med2,A
inc     R0
mov     A,@R0
anl     A,#0FH
add     A,clave_med2
mov     clave_med2,A
inc     R0

mov     A,@R0
anl     A,#0FH
swap    A
mov     clave_med3,A
inc     R0
mov     A,@R0
anl     A,#0FH
add     A,clave_med3
mov     clave_med3,A
inc     R0

mov     A,@R0
anl     A,#0FH
swap    A
mov     clave_low,A
inc     R0
mov     A,@R0
anl     A,#0FH
add     A,clave_low
mov     clave_low,A

inc     R0
```

```
        mov     A,@R0
        anl     A,#0FH
        swap    A
        mov     CLAVE_DIG,A

        RET
;
;*****************************************************************
;
;       Rutina para grabar los registros en la NVRAM
;
grabar:
        mov     DPTR,#direcc_low
        movx    A,@DPTR                 ;Actualiza la dirección del DPTR
        push    ACC
        mov     DPTR,#direcc_hig
        movx    A,@DPTR
        mov     DPH,A
        pop     ACC
        mov     DPL,A
        mov     A,anio_dat              ;Guarda el año
        movx    @DPTR,A
        mov     A,mese_dat              ;Guarda el mes
        inc     DPTR
        movx    @DPTR,A
        mov     A,diam_dat              ;Guarda el día
        inc     DPTR
        movx    @DPTR,A
        mov     A,hora_dat              ;Guarda la hora
        inc     DPTR
        movx    @DPTR,A
        mov     A,min_dat               ;Guarda el minuto
        inc     DPTR
        movx    @DPTR,A
        mov     A,clave_hig             ;Guarda el nro de tarjeta
        inc     DPTR
        movx    @DPTR,A
        mov     A,clave_med1            ;Guarda el nro de tarjeta
        inc     DPTR
        movx    @DPTR,A
```

```
        mov     A,clave_med2            ;Guarda el nro de tarjeta
        inc     DPTR
        movx    @DPTR,A
        mov     A,clave_med3            ;Guarda el nro de tarjeta
        inc     DPTR
        movx    @DPTR,A
        mov     A,clave_low             ;Guarda el nro de tarjeta
        inc     DPTR
        movx    @DPTR,A
        mov     A,clave_dig             ;Guarda el nro de tarjeta
        inc     DPTR
        movx    @DPTR,A
        inc     DPTR
        mov     A,#'Z'                  ;Este valor indica el fin del
        movx    @DPTR,A                 ;del archivo de datos

        mov     A,DPL
        push    ACC
        mov     A,DPH
        mov     DPTR,#direcc_hig        ;Actualiza los pointers
        movx    @DPTR,A
        pop     ACC
        mov     DPTR,#direcc_low
        movx    @DPTR,A
;
;*************************************************************************
;
;       Rutina para verificar que no se excedan las grabaciones de la
;       dirección 7Fxxh que es la máxima capacidad del controlador.
;       Se dejan 255 bytes libres para efectos de seguridad
;
        mov     A,DPH
        cjne    A,#7FH,no_lleno
        lcall   memoria_full
        lcall   display_lin1
        sjmp    $
;
;*************************************************************************
;
no_lleno:
```

```
        ret
;
;*****************************************************************************
;
computador:
        clr     EX1
        clr     ES
        lcall   datos_cpu               ;Va a rutina de lectura de datos desde
        clr     RI                      ;el computador
        clr     TI
        setb    EX1
        setb    ES
        ljmp    main
;
;*****************************************************************************
;
;       datos_cpu       Rutina par tomar datos desde computador.
;               La cpu debe mandar un header luego datos y fin de archivo
;               El header puede ser 'I' para igualar el reloj y 'L' para
;               lectura de datos.
;               Para igualar reloj el cpu debe mandar la información de
;               fecha y hora así:
;               IMM-DD-AAAAHH:MM:SS
;
datos_cpu:
        push    ACC
        push    PSW
        push    DPL
        push    DPH
        clr     RI
        mov     A,SBUF
        mov     B,A
        cjne    A,#'I',ver_dato1
        lcall   igualar                 ;Va a rutina de igualar reloj
        ljmp    salir_cpu
ver_dato1:
        cjne    A,#'L',ver_dato2
        lcall   leer                    ;Va a rutina de leer datos del CPU
        ljmp    salir_cpu
ver_dato2:
```

```
        cjne    A,#'Z',ver_dato3
        lcall   encerar                 ;Encerar registros
        ljmp    salir_cpu
ver_dato3:
        cjne    A,#'M',ver_dato4
        lcall   mandar_datos            ;Mandar datos al cpu
        ljmp    salir_cpu
;
;*****************************************************************************
;
ver_dato4:
salir_cpu:
        pop     DPH
        pop     DPL
        pop     PSW
        pop     ACC
        clr     RI
        clr     TI
        ret
;
;*****************************************************************************
;
;       Rutina para mandar datos al computador.
;       Esta rutina envía todos los datos almacenados desde el ultimo
;       encerado.
;
mandar_datos:
        mov     A,#'M'                  ;Devuelve caracter para indicar que lo recibió
        lcall   send_car
        mov     A,#car_ret
        lcall   send_car

        mov     DPTR,#numero
        movx    A,@DPTR
        mov     R7,A                    ;Guarda el numero mas significativo de
estación
        inc     DPTR
        movx    A,@DPTR
        mov     R6,A                    ;Guarda el numero menos significativo de
estación
```

```
        mov     DPTR,#datos
        movx    A,@DPTR                 ;Verifica si esta vacía la base de datos
        cjne    A,#'Z',lazo_mandar
        ljmp    fin_mandar
;
;*****************************************************************************
;
lazo_mandar:
        mov     A,R7                    ;Recupera el numero de estación
        lcall   send_car                ;Manda los dos caracteres del numero de
                                        ;estación
        mov     A,R6                    ;Recupera el numero de estación
        lcall   send_car
        mov     A,#','
        lcall   send_car

        movx    A,@DPTR
        lcall   send_dat                ;Manda los dos caracteres del año
        mov     A,#','
        lcall   send_car

        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los dos caracteres del mes
        mov     A,#','
        lcall   send_car

        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los dos caracteres del día
        mov     A,#','
        lcall   send_car

        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los dos caracteres de la hora
        mov     A,#','
        lcall   send_car
```

```
        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los dos caracteres del minuto
        mov     A,#','
        lcall   send_car

        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los caracteres de la tarjeta
        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los caracteres de la tarjeta
        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los caracteres de la tarjeta
        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los caracteres de la tarjeta
        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los caracteres de la tarjeta
        inc     DPTR
        movx    A,@DPTR
        lcall   send_dat                ;Manda los caracteres de la tarjeta

        mov     A,#car_ret              ;Manda retorno de carro por cada registro
        lcall   send_car                ;completo de datos

        inc     DPTR                    ;Hace el lazo hasta que se terminen los
        movx    A,@DPTR                 ;datos
        cjne    A,#'Z',lazo_mandar
fin_mandar:
        mov     A,#26                   ;Para finalizar el envio de todos los
        lcall   send_car                ;registros manda un cntrl_z

        mov     A,#car_ret
        lcall   send_car

        ret
;
```

```
;*************************************************************************
;
send_dat:
        mov     B,A                     ;Guarda temporal
        swap    A
        anl     A,#0FH                  ;Lo transforma en ASCII
        orl     A,#30H
        clr     TI
        lcall   manda_rs
        mov     A,B                     ;Recupera el datos original
        anl     A,#0FH                  ;Lo transforma en ASCII
        orl     A,#30H
        lcall   manda_rs
        ret
;
;*************************************************************************
;
manda_rs:
        mov     SBUF,A
        jnb     TI,$                    ;Espera al fin de transmisión
        clr     TI
        ret
;
;*************************************************************************
;
send_car:
        clr     TI
        lcall   manda_rs
        ret
;
;*************************************************************************
;
;       Rutina para encerar los registros de almacenamiento en la NVRAM
;
encerar:
        mov     DPTR,#datos
        mov     A,DPL
        mov     B,A
        mov     A,DPH
        mov     DPTR,#direcc_hig
```

```
        movx    @DPTR,A
        mov     A,B
        mov     DPTR,#direcc_low
        movx    @DPTR,A
        mov     DPTR,#datos
        mov     A,#'Z'
        movx    @DPTR,A

        mov     R0,#0FFH
        clr     A
lazo_encerar:
        inc     DPTR
        movx    @DPTR,A
        djnz    R0,lazo_encerar
        ret
;
;
;*************************************************************************
;
leer:
        mov     A,#'L'                  ;Devuelve caracter para indicar que lo recibió
        lcall   SEND_CAR
        lcall   FIN_STR
        mov     DPTR,#estacion
        mov     R7,#13                  ;Recibe máximo 13 caracteres de nombre
lazo_leer:
        jnb     RI,$
        clr     RI
        mov     A,SBUF
        movx    @DPTR,A
        inc     DPTR
        djnz    R7,lazo_leer

        mov     DPTR,#numero
        mov     R7,#2                   ;Recibe máximo 2 caracteres de numero
lazo_leer1:
        jnb     RI,$
        clr     RI
        mov     A,SBUF
        movx    @DPTR,A
```

```
        inc     DPTR
        djnz    R7,lazo_leer1

        lcall   estacion_dis
        lcall   display_lin2
        ret
;
;
;*******************************************************************************
;
igualar:
        mov     A,#'I'                  ;Devuelve caracter para indicar que lo recibió
        lcall   send_car
        lcall   fin_str

imes:   lcall   caracter
        mov     mese_dat,A
        jnb     RI,$
        clr     RI                      ;caracter de separación
diames:lcall    caracter
        mov     diam_dat,A
        jnb     RI,$
        clr     RI                      ;caracter de separación
        jnb     RI,$
        clr     RI                      ;Los dos caracteres son de decenas de año
        jnb     RI,$
        clr     RI                      ;Los dos caracteres son de decenas de año
anios:  lcall   caracter
        mov     anio_dat,A
horas:  lcall   caracter
        mov     hora_dat,A
        jnb     RI,$
        clr     RI                      ;caracter de separación
minutos:        lcall   caracter
        mov     min_dat,A
        jnb     RI,$
        clr     RI                      ;caracter de separación
segundos:       lcall   caracter
        mov     seg_dat,A
        jnb     RI,$                    ;Espera caracter de car_ret del basic
```

```
        clr     RI
        lcall   open
        mov     A,#0                    ;Pone en reloj .00 segundos
        lcall   grareg
        mov     A,seg_dat
        lcall   grareg
        mov     A,min_dat
        lcall   grareg
        mov     A,hora_dat
        lcall   grareg
        mov     A,#01H                  ;Pone en reloj el primer día
        lcall   grareg
        mov     A,diam_dat
        lcall   grareg
        mov     A,mese_dat
        lcall   grareg
        mov     A,anio_dat
        lcall   grareg
        clr     RI
        ret                             ;Termina rutina de igualación

;
;
;************************************
;
caracter:       jnb     RI,$            ;Espera próximo caracter
        clr     RI
        mov     A,SBUF                  ;Lee caracter en ASCII
        anl     A,#0FH                  ;Lo transforma en HEXA
        swap    A
        mov     B,A                     ;Guarda dato
        jnb     RI,$                    ;Espera próximo caracter
        clr     RI
        mov     A,SBUF                  ;Lee caracter en ASCII
        anl     A,#0FH                  ;Lo transforma en HEXA
        add     A,B
        ret
;
;************************************
;
```

```
fin_str: clr     TI
         mov     A,#car_ret
         mov     SBUF,A
         jnb     TI,$                    ;Espera al fin de transmisión
         clr     TI
         ret
;
;
;*************************************************************************
;
;     Rutina para leer el reloj y ponerlo en el display
;
reloj:   lcall   leer_clk                ;Pone en condición de leer

         mov     R0,#inicio_d
         mov     A,diam_dat              ;Muestra en display "DIA        "
         lcall   ascii_dis

         mov     @R0,#'-'                ;Muestra en display "DIA-       "
         inc     R0
         mov     A,mese_dat              ;Muestra en display "DIA-MES    "
         lcall   ascii_dis
         mov     @R0,#'-'                ;Muestra en display "DIA-MES-   "
         inc     R0

         mov     A,anio_dat              ;Muestra en display "DIA-MES-AÑO      "
         lcall   ascii_dis
         mov     @R0,#' '                ;Muestra en display "DIA-MES- AÑO "
         inc     R0
         mov     @R0,#' '                ;Muestra en display "DIA-MES- AÑO "
         inc     R0
         mov     A,hora_dat              ;Muestra en display "DIA-MES- AÑO HORA "
         lcall   ascii_dis
         mov     @R0,#':'                ;Muestra en display "DIA-MES- AÑO HORA: "
         inc     R0
         mov     A,min_dat               ;Muestra display "DIA-MES- AÑO HORA:MIN"
         lcall   ascii_dis

         mov     @R0,#' '                ;Borra el ultimo dígito derecho
```

```
        ret

;*************************************************************************
;
leer_clk:
        lcall   open                    ;Pone en condición de leer

next_lee:       lcall   leereg          ;Va a rutina de lectura del reloj
                                        ;no almacena décimas de segundo
        lcall   leereg                  ;Va a rutina de lectura del reloj
        mov     seg_dat,A               ;Guarda valor de segundos
        lcall   leereg                  ;Va a rutina de lectura del reloj
        mov     min_dat,A               ;Guarda valor de minutos
        lcall   leereg                  ;Va a rutina de lectura del reloj
        mov     hora_dat,A              ;Guarda valor de hora
        lcall   leereg                  ;Va a rutina de lectura del reloj
        mov     dias_dat,A              ;Guarda valor de día de la semana
        lcall   leereg                  ;Va a rutina de lectura del reloj
        mov     diam_dat,A              ;Guarda valor de día del mes
        lcall   leereg                  ;Va a rutina de lectura del reloj
        mov     mese_dat,A              ;Guarda valor de mes
        lcall   leereg                  ;Va a rutina de lectura del reloj
        mov     anio_dat,A              ;Guarda valor de año
        ret
;
;*************************************************************************
;
;       rutina para convertir en ascii valor hexa y guardar en
;       posiciones sucesivas de ram para display
;
ascii_dis:      mov     B,A
        swap    A
        anl     A,#0FH
        add     A,#30H                  ;Display requiere datos en ASCII
        mov     @R0,A
        inc     R0
        mov     A,B
        anl     A,#0FH
        add     A,#30H
        mov     @R0,A
```

```
        inc     R0
        ret
;*************************************************************************
;
;
;open   rutina para ejecutar la secuencia de lecturas y escrituras
;       necesarias para establecer la comunicación con el reloj.
;       La subrutina regresa dejando abierto el reloj para mantener
;       la comunicación, el ACC y el registro B quedan modificados.
;
open:   lcall   close                   ;Asegura que el reloj esta cerrado
        mov     B,#4                    ;Pone contador de lazos
        mov     A,#0C5H                 ;Pone el primer byte del patrón
openA:  lcall   grareg                  ;Escribe el byte en el reloj
        xrl     A,#0FFH                 ;Genera el siguiente byte del patrón
        lcall   grareg                  ;Pone el siguiente byte
        swap    A                       ;Genera el siguiente byte del patrón
        djnz    B,openA                 ;Repite hasta que se envíen los 8 Bytes
        ret                             ;Termina la rutina
;
;*************************************************************************
;
;close  rutina para asegurar que todos los registros del reloj
;       están cerrados.
;
close:  mov     B,#9                    ;realiza 9 lecturas
loop_cer:       lcall   leereg
        djnz    B,loop_cer
        ret
;
;*************************************************************************
;
;leereg rutina para leer datos del reloj , devuelve el valor leído
;       en el ACC
;
leereg: push    DPL                     ;guarda el DATA POINTER en el stack
        push    DPH
        push    MCON
        orl     MCON,#4                 ;Switch a CE2
        push    B                       ;Preserva el registro B
```

```
        mov     DPL,#4                  ;inicializa address de entrada
        mov     DPH,#0
        mov     B,#8                    ;inicializa el contador
loop_lee:       push    ACC             ;Preserva el ACC
        movx    A,@DPTR                 ;lee el dato del reloj
        rlc     A                       ;Pone el dato en el CARRY
        pop     ACC                     ;Recupera el ACC
        rrc     A                       ;Pone el dato leído en el ACC
        djnz    B,loop_lee              ;Repite el lazo por 8 veces
        pop     B                       ;Recupera el registro B
        pop     MCON                    ;Repone registro MCON
        pop     DPH                     ;Repone data pointer desde stack
        pop     DPL
        ret                             ;retorna a rutina invocante
;
;*************************************************************************
;
;grareg Rutina para grabar un registro desde el ACC en el reloj
;       El ACC trae valor a guardar
;
grareg: push    DPL                     ;Guarda el DATA POINTER en el stack
        push    DPH
        push    MCON                    ;Guarda el registro MCON
        ORL     MCON,#4                 ;Cambia para CE2
        push    B                       ;Guarda registro B
        mov     DPH,#0                  ;Pone dirección alta en 0
        mov     B,#8                    ;Pone en 8 contador de lazo
loop_gra:       push    ACC             ;Guarda el ACC
        anl     A,#1                    ;Setea bit para arrancar
        mov     DPL,A                   ;Debe poner dirección 01
        movx    @DPTR,A                 ;Saca valor al reloj
        pop     ACC                     ;Recupera el ACC
        rr      A                       ;Prepara para el siguiente bit
        djnz    B,loop_gra              ;Repite el lazo por B veces
        pop     B                       ;Repone registro B
        pop     MCON                    ;Repone registro MCON
        pop     DPH                     ;Repone data pointer desde el stack
        pop     DPL
        ret                             ;retorna a rutina invocante
```

```
;
;*******************************************************************
;
;       Rutina para intercambiar los valores del ACC de manera que el
;       bit0 sea el bit7, bit1 sea bit6, etc. para adaptar el puerto
;       P2 a la entrada del bus del display
;
cambia:
        mov     B,#0H
        jb      ACC.0,poner_7
ret_poner7:
        jb      ACC.1,poner_6
ret_poner6:
        jb      ACC.2,poner_5
ret_poner5:
        jb      ACC.3,poner_4
ret_poner4:
        jb      ACC.4,poner_3
ret_poner3:
        jb      ACC.5,poner_2
ret_poner2:
        jb      ACC.6,poner_1
ret_poner1:
        jb      ACC.7,poner_0
ret_poner0:
        mov     A,B
        ret
;
;*******************************************************************
;
poner_7:
        orl     B,#80H
        ljmp    ret_poner7
poner_6:
        orl     B,#40H
        ljmp    ret_poner6
poner_5:
        orl     B,#20H
        ljmp    ret_poner5
poner_4:
```

```
        orl     B,#10H
        ljmp    ret_poner4
poner_3:
        orl     B,#08H
        ljmp    ret_poner3
poner_2:
        orl     B,#04H
        ljmp    ret_poner2
poner_1:
        orl     B,#02H
        ljmp    ret_poner1
poner_0:
        orl     B,#01H
        ljmp    ret_poner0
;
;
;*****************************************************************************
;
;       Rutina de inicialización temporizada del display
;
init_display:
        lcall   del_1seg
        setb    ena
        mov     port_dis,#30H
        clr     ena
        lcall   del_20mili
        setb    ena
        mov     port_dis,#30H
        clr     ena
        lcall   del_20mili
        setb    ena
        mov     port_dis,#30H
        clr     ena

        lcall   clrD
        lcall   del_20mili
        mov     data_dis,#38H           ;Define dos líneas y 5x7 dots
        lcall   saca                    ;y 8 bits de interface
        lcall   clrD
        lcall   del_20mili
```

```
        mov     data_dis,#0AH           ;SET DISPLAY OFF CURSOR ON
        lcall   saca
        lcall   clrD
        mov     data_dis,#0EH           ;SET DISPLAY AND CURSOR ON
        lcall   saca
        lcall   clrD
        mov     data_dis,#06H           ;SET inc add 1,SHIFT CUR RIGHT
        lcall   saca
        lcall   clrDIS                  ;Borra el display
        lcall   del_20mili
        ret
;
;*************************************************************************
;
clrdis:
        lcall   clrd                    ;Rutina para borrar el display
        setb    ena
        mov     data_dis,#01H
        lcall   saca
        ret
;
;*************************************************************************
;
saca:
        mov     A,data_dis              ;Recupera el valor para el display
        lcall   cambia                  ;Va a rutina de intercambio de bits
        mov     data_dis,A
        lcall   del_150micro            ;Rutina para sacar datos que vienen en
        setb    ena
        setb    r_w                     ;Esta condición se obtiene si se sube el R_W
;       clr     rs
lazo_saca:
        mov     A,port_dis
libre:  clr     r_w
        mov     A,data_dis
        mov     port_dis,A
        lcall   del_100micro
        clr     ena
        ret
;
```

```
;*******************************************************************************
;
clrd:   clr     rs                      ;Baja las señales de control del display
        clr     r_w
        clr     ena
        ret
;
;*******************************************************************************
;
;       Area de definicion de letreros fijos para el display
;
escuela_dis:
        mov     R1,#(0*16)
        ljmp    sacle
nacional_dis:
        mov     R1,#(1*16)
        ljmp    sacle
memoria_full:
        mov     R1,#(2*16)
        ljmp    sacle
correcta_dis:
        mov     R1,#(3*16)
        ljmp    sacle

;
;*******************************************************************************
;

sacle:  mov     A,R1                    ;Dirección del letrero
add     A,#09H                          ;Constante del Program Counter
        mov     R1,A
        mov     R0,#inicio_d            ;Dirección inicial del buffer de display
l1:     mov     A,R1                    ;Pone la dirección del caracter
        movc    A,@A+PC                 ;Lee el caracter de la tabla
        mov     @R0,A                   ;Lo guarda en el buffer del display
        inc     R0
        inc     R1
        mov     A,R0
        xrl     A,#(inicio_d+16)        ;En total debe leer 16 caracteres
        jnz     l1
```

```
        ret
;
;*****************************
;
display_00:DB 'Escuela Politec.'
display_01:DB '   NACIONAL    '
display_02:DB 'Memoria *llena* '
display_03:DB 'Tarjeta correcta'

;
;*************************************************************************
;
display_lin1:
        clr     EA
        lcall   clrd
        mov     data_dis,#80H           ;SET address EN 00
        lcall   saca
        mov     R0,#inicio_d            ;INICIO DE RAM 30H
        mov     R6,#num_car             ;REG DE CONTROL
con_otro1:
        lcall   otro
        inc     R0
        djnz    R6,con_otro1
        clr     C                       ;Borra el carry para control de salida
        setb    EA
        ret
;
;*************************************************************************
;
display_lin2:
        clr     EA
        lcall   clrd
        mov     data_dis,#0C0H          ;SET add EN 40
        lcall   saca
        mov     R0,#inicio_d            ;INICIO DE RAM 30H
        mov     R6,#num_car             ;REG DE CONTROL
con_otro2:
        lcall   otro
        inc     R0
        djnz    R6,con_otro2
```

```
        clr     C                       ;Borra el carry para control de salida
        setb    EA
        ret
;
;*****************************************************************************
;
otro:
        setb    rs                      ;SET R
        clr     r_w                     ;clr R/W
        clr     ena                     ;clr E
        setb    ena                     ;SET E
        mov     A,@R0
        mov     data_dis,A              ;SACA LETRA
        lcall   saca
        clr     rs                      ;clr RS
        ret
;
;*****************************************************************************
;
estacion_dis:
        mov     R0,#inicio_d            ;Inicializa el área de display
        mov     DPTR,#estacion          ;Trae los valores almacenados en la NVRAM
                                        ;que contienen el nombre de la estación
        mov     R1,#13                  ;Debe cargar 13 caracteres
lazo_estacion:
        movx    A,@DPTR
        mov     @R0,A
        inc     DPTR
        inc     R0
        djnz    R1,lazo_estacion
        mov     @R0,#' '

        mov     R0,#(inicio_d+14)
        mov     DPTR,#numero            ;Trae los valores almacenados en la NVRAM
                                        ;que contienen el numero  de la estación
        mov     R1,#2                   ;Debe cargar 2 caracteres
lazo_estacion1:
        movx    A,@DPTR
        mov     @R0,A
```

```
        inc     DPTR
        inc     R0
        djnz    R1,lazo_estacion1
        ret

;
;*****************************************************************************
;
;       Calculo para 1 Segundo
;       1 CM=12/7.3728*10e6 = 1.63 * 10 e-6 = 1.63 micseg
;       total  1 / 1.63*10e-6
;       CMTOT 1seg = 614400 ==> 9 * 65536 + 24576
;       Complemento a 2 = 65536 - 24576 = 40960
;        75E4 h
del_1seg:
        mov     R7,#10
        mov     TH0,#0A0H
        mov     TL0,#0H
        setb    TR0
lazo_1seg:
        jnb     TF0,$
        clr     TF0
        djnz    R7,lazo_1seg
        clr     TR0
        ret
;
;*****************************************************************************
;
del_5seg:
        mov     R6,#5                   ;Hace el lazo por 5 segundos esperando
lazo_del5:                              ;que se presione otra tecla, si no se
        lcall   del_1seg                ;presiona sale por timeout
sigue_del:
        djnz    R6,lazo_del5
        ret
;
;*****************************************************************************
;
del_20mili:
```

```
        mov     TH0,#41H
        mov     TL0,#0E0H
        setb    TR0
        jnb     TF0,$
        clr     TF0
        clr     TR0
        ret
;
;*****************************************************************************
;
del_10mili:

        mov     TH0,#20H
        mov     TL0,#0F0H
        setb    TR0
        jnb     TF0,$
        clr     TF0
        clr     TR0
        ret
;
;*****************************************************************************
;
del_150micro:

        mov     TH0,#0FFH
        mov     TL0,#2AH
        setb    TR0
        jnb     TF0,$
        clr     TR0
        clr     TF0
        clr     TR0
        ret
;
;*****************************************************************************
;
del_100micro:

        mov     TH0,#0FFH
        mov     TL0,#6CH
        setb    TR0
```

```
        jnb     TF0,$
        clr     TR0
        clr     TF0
        clr     TR0
        ret
;
;
;*****************************************************************************
;
;       Origen del área NVRAM
;
        org     1800H
estación        equ     1800H
;hasta          1810H
numero equ      1811H
;hasta          1812H
direcc_low      equ     1813H
direcc_hig      equ     1814H


;
;
;*****************************************************************************
;
        org     1820H
datos:
;
;*****************************************************************************
;
;****************************
;   ***************
;      *******
;        *
;
         END
;
;        *
;      *******
;   ***************
;****************************
```

LISTADO DEL PROGRAMA EN QBASIC USADO EN LA COMUNICACIÓN ENTRE EL PROTOTIPO Y EL PC DE PROCESO.

```
'*************************************************************************
'
'       PROGRAMA DE MENU GENERAL DE CONTROL DE BUSES
'
'       BUSES.BAS
'
'*************************************************************************
'
INICIO:
SCREEN 12
CLS
GOSUB CUADROS
TECLA$ = ""
PORT$ = "COM1:2400,N,8,1,CS,CD,DS,RS"
COLOR 11
LOCATE 8, 35: PRINT "MENU DE OPCIONES"
LINE (190, 145)-(490, 145), 4
LINE (190, 148)-(490, 148), 4
COLOR 7
LOCATE 13, 30: PRINT " 1.-   Igualar el reloj  "
LOCATE 14, 30: PRINT " 2.-   Borrar registros  "
LOCATE 15, 30: PRINT " 3.-   Leer registros    "
LOCATE 16, 30: PRINT " 4.-   Cargar estacion   "
LOCATE 20, 30: PRINT " 9.-   Fin de  operaci¢n "
LINE (190, 360)-(490, 360), 4
LINE (190, 363)-(490, 363), 4
COLOR 15
LOCATE 25, 28: PRINT "Escoja su opci¢n : ===>  "
WHILE TECLA$ = ""
TECLA$ = INKEY$
COLOR 11
LOCATE 3, 3
PRINT DATE$
LOCATE 3, 70
PRINT TIME$
WEND
'
'************************************
'
ESCOGER:
IF TECLA$ = "1" THEN GOTO IGUALAR
IF TECLA$ = "2" THEN GOTO ENCERAR
IF TECLA$ = "3" THEN GOTO LEER
IF TECLA$ = "4" THEN GOTO CARGAR
IF TECLA$ = "9" THEN GOTO CDOS
'
'******************
'
FINALIZA:
CLOSE #1
CLOSE #2
```

```
GOTO INICIO
'
'*************************************************************************
'
LEER:
CHAIN "LEERBUS"
END
'
'*************************************************************************
'
CDOS:
CLOSE #1
END
'
'*************************************************************************
'
IGUALAR:
CLOSE #1
OPEN PORT$ FOR RANDOM AS #1
LOCATE 25, 28: INPUT "Si quiere IGUALAR digite <S> ", ESPE$
IF ESPE$ = "S" OR ESPE$ = "s" THEN GOTO siiguala
GOTO FINALIZA
'
'******************
'
siiguala:
PRINT #1, "I";
LINE INPUT #1, entrada$

FECHA$ = DATE$
HORA$ = TIME$
PRINT #1, DATE$; TIME$
GOTO FINALIZA
'
'*************************************************************************
'
CARGAR:
CLOSE #1
OPEN PORT$ FOR RANDOM AS #1
LOCATE 25, 28
INPUT "Si quiere CARGAR digite <S> ", ESPE$
IF ESPE$ = "S" OR ESPE$ = "s" THEN GOTO SICARGA
GOTO FINALIZA
'
'******************
'
SICARGA:
vacio$ = "              "
LOCATE 20, 27
PRINT "                    ++++++++++++"
LOCATE 20, 27: INPUT "Nombre estacion  : ", estacion$
estacion$ = estacion$ + vacio$

LOCATE 21, 30
vacio$ = "  "
PRINT "                         **"
LOCATE 21, 30: INPUT "Numero de estacion  : ", numero$
numero$ = numero$ + vacio$

PRINT #1, "L";
LINE INPUT #1, entrada$
```

```
PRINT #1, LEFT$(estacion$, 13);
PRINT #1, LEFT$(numero$, 2)

GOTO FINALIZA

'
'*********************************************************************
'
ENCERAR:
CLOSE #1
OPEN PORT$ FOR RANDOM AS #1
LOCATE 25, 28: INPUT "Si quiere ENCERAR digite <S> ", ESPE$
IF ESPE$ = "S" OR ESPE$ = "s" THEN GOTO SIENCERA
GOTO FINALIZA
'
'******************
'
SIENCERA:
PRINT #1, "Z";
GOTO FINALIZA

'
'*********************************************************************
'
CUADROS:
DRAW "BM0,0"                 'Mueve sin color al punto 0,0
DRAW "C9"                    'Dibuja en color lila
DRAW "R630 D450 L630 U450"   'Dibuja el recuadro
DRAW "BM5,5"                 'Mueve sin color al punto 5,5
DRAW "C2"                    'Dibuja en color verde
DRAW "R620 D53 L620 U53"     'Dibuja el recuadro
LINE (5, 50)-(625, 50), 2
LOCATE 2, 22
PRINT "PROGRAMA PARA CONTROL DE BUSES"
COLOR 11
LOCATE 3, 3
PRINT DATE$
LOCATE 3, 70
PRINT TIME$
DRAW "BM200,95"              'Mueve sin color al punto 200,95
DRAW "C8"                    'Dibuja en color 8
DRAW "R300 D300 L300 U300"   'Dibuja a partir del punto 200,95: 300 a
la
                             'derecha, 300 abajo, 300 izquierda, 300
                              'arriba
DRAW "BR10 BD10"             'Se ubica dentro del cuadro
DRAW "P3,8"                  'Pinta de color 6 hasta los limites del
color
                               '8

DRAW "BM190,105"
DRAW "C4"
DRAW "R300 D300 L300 U300"
DRAW "BR10 BD10"
DRAW "P7,4"
RETURN
'
'
'*********************************************************************
'
```

```
'*********************************************************************
'        PROGRAMA PARA LEER DATOS REGISTRADOS EN EL EQUIPO
'
'        LEER.BAS
'
'*********************************************************************
'
INICIO:
SCREEN 12
CLS
GOSUB CUADROS1
GOSUB CUADROS2
COLOR 11
LOCATE 8, 28: PRINT "LECTURA DE DATOS DEL REGISTRADOR"
LINE (190, 145)-(490, 145), 4
LINE (190, 148)-(490, 148), 4
COLOR 7
LOCATE 12, 28: PRINT " Para leer todos los datos <T>"
LOCATE 14, 28: PRINT " Para salir presione     <ENT>"
LOCATE 25, 28: INPUT "Escoja su opcion  : ", ESPE$

IF ESPE$ = "T" THEN GOTO LEERTODOS
IF ESPE$ = "t" THEN GOTO LEERTODOS
GOTO FIN
'
'************************************
'
LEERTODOS:
LOCATE 16, 28
ARCHIVO$ = "datos.txt"

CLOSE #1
CLOSE #2
PORT$ = "COM1:2400,N,8,1,CS,CD,DS,RS"    'Abre la puerta del reloj
OPEN PORT$ FOR RANDOM AS #1
OPEN ARCHIVO$ FOR APPEND AS #2

PRINT #1, "M";
LINE INPUT #1, entrada$

REGISTRO = 1
entrada$ = ""
        LINE INPUT #1, entrada$
        PRINT #2, entrada$

WHILE NOT entrada$ = CHR$(26)
ESPERAR1:
        ON ERROR GOTO ESPERAR1
        LINE INPUT #1, entrada$
        PRINT #2, entrada$
        LOCATE 20, 28: PRINT "REGISTRO  :  "; REGISTRO
        REGISTRO = REGISTRO + 1
WEND

FINARCHIVO:
LOCATE 25, 28
INPUT "Termino el archivo    <ENT>", ESPE$
GOTO INICIO
'
```

```
'************************************
'
FIN:
CLOSE #1
CLOSE #2
CHAIN "BUSES"
END
'
'*************************************************************************
'
CUADROS1:
DRAW "BM0,0"                        'Mueve sin color al punto 0,0
DRAW "C9"                           'Dibuja en color lila
DRAW "R630 D450 L630 U450"          'Dibuja el recuadro
DRAW "BM5,5"                        'Mueve sin color al punto 5,5
DRAW "C2"                           'Dibuja en color verde
DRAW "R620 D53 L620 U53"            'Dibuja el recuadro
LINE (5, 50)-(625, 50), 2
LOCATE 2, 27
COLOR 15
PRINT "PROGRAMA PARA CONTROL DE BUSES"
COLOR 11
LOCATE 3, 3
PRINT DATE$
LOCATE 3, 70
PRINT TIME$
RETURN
'
'************************************
'
CUADROS2:
DRAW "BM200,95"              'Mueve sin color al punto 200,95
DRAW "C8"                    'Dibuja en color 8
DRAW "R300 D300 L300 U300"   'Dibuja a partir del punto 200,95: 300 a
la
                             'derecha, 300 abajo, 300 izquierda, 300
                             'arriba
DRAW "BR10 BD10"             'Se ubica dentro del cuadro
DRAW "P3,8"                  'Pinta de color 6 hasta los limites del
color
                              '8
DRAW "BM190,105"
DRAW "C4"
DRAW "R300 D300 L300 U300"
DRAW "BR10 BD10"
DRAW "P7,4"
RETURN
'
'************************************
'
PIE1:
LINE (190, 360)-(490, 360), 4
LINE (190, 363)-(490, 363), 4
COLOR 15
LOCATE 25, 28: PRINT "Para terminar presione <ENT> "
RETURN
'
'*************************************************************************
```