

SG/426

SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO



No. 14-060246- -00002-0000

Fecha: 2014-04-16 09:39:20 Dep. 2020 DIR.NUEVASOR
Tra. 3 MODELO Eve: 1 REGDEPOSITO
Act. 444 RESPUESTAREQUE Folios: 63

Pereira, martes 15 de abril de 2014.

Doctor
JOSE LUIS SALAZAR LOPEZ
Director de Nuevas Creaciones
SUPERINTENDENCIA DE INDUSTRIA Y COMERCIO.

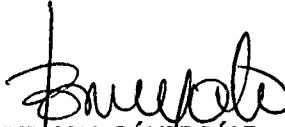
Asunto: Respuesta a Oficio No. Radicado 14-060246- -00001-000 - 2014-04-02.

Respetado Doctor Salazar,

Me permito dar respuesta al oficio RAD: 14-060246-00001-000 del 02 de abril de 2014, mediante el cual solicitan las correcciones referentes al registro de Patente por Modelo de Utilidad del Hardware "MICROPUBLIK", con numero de radicado No. 14-060246-00000-000 del 20 de marzo de 2014, ante cualquier requerimiento favor comunicarse al teléfono 3377500 ext. 603.

Correo electrónico: bgalvez@publik.co

Atentamente



BIBIANA GÁMEZ DÍAZ
Líder Gestión Legal
PUBLIK S.A.S

REIVINDICACIÓN

REIVINDICACIÓN 1

PREAMBULO

La tarjeta MicroPublik hace parte del sector de la electrónica, y desde su contexto técnico permite llevar a cabo la transmisión de mensajes de texto, temperatura, hora, fecha y datos a un display electrónico, cuya etapa de potencia esté constituida por las tarjetas o las Board Driver de los informadores o equipos incandescentes. Ésta tarjeta permite visualizar mensajes con cierta variedad de efectos que son: Rotate, Hold, Flash, Scroll, WipeUp. Estos efectos pueden ser programados con 10 velocidades distintas de visualización, permitiendo también la visualización de temperatura, hora en 2 formatos, fecha en tres formatos y texto en negrilla. La programación de los mensajes y las funciones mencionadas, se realizan vía puerto serial RS485 ó RS232 desde un software de programación.

CARACTERISTICAS

Antes de la creación de la tarjeta MicroPublik en el sector electrónico, no existía una tarjeta electrónica que conectara un mensaje realizado en un PC y a su vez registre medidas del mundo real como es la hora y la temperatura desde un informador de luz incandescente.

De manera, que lo que se pretende proteger, es la tarjeta en su integridad, al igual que cada uno de los componentes que la integran ya que cumplen una función primordial, con el fin de lograr el objetivo para el cual fue creada.

La Tarjeta MicroPublik, según los nombres dispuestos en el diseño (Esquemático e Impreso), está caracterizada por:

- H1, H2, H3, H4: Hoyos de fijación para la Tarjeta MicroPublik.
- P\$1, P\$2, P\$3: Conector de Entrada DC Jack.
- X2: Bornera de 2 puestos.
- VR1: Regulador de Voltaje de 12 Voltios.
- VR2: Regulador de Voltaje de 5 Voltios.
- D3, D4: Diodos de señal 1N4004.
- C22: Condensador electrolítico de 2200uF a 25V.
- C21: Condensador electrolítico de 1000uF a 16V.
- C17, C13, C19, C14, C16, C15, C11: Condensador de cerámica 104 (0,1uF o 100nF).
- C8: Condensador electrolítico de 470uF a 16V.
- C20: Condensador de tantalio de 1uF.
- U8: Circuito Integrado 7416, el cual tiene 4 compuertas negadoras.
- R13, R14, R15, R16: Resistencia de 470 Ohm a 1/4W.
- X1: Bornera de 6 puestos.
- RS485: Bornera de 3 puestos.
- U6: Circuito Integrado 75176: Es un circuito integrado diseñado para la comunicación bidireccional en líneas de transmisión multipunto. Está diseñado para líneas de transmisión balanceadas bajo el estándar EIA/TIA-422-B. Este chip posibilita la conversión entre los niveles TTL del micro-controlador y las señales del tipo diferencial que se utilizan el bus RS-485. Este integrado tiene dos líneas de control que dan 3 funciones diferentes al chip: transmisor, receptor y generador de alta impedancia, evitando así conflictos en el bus de datos.
- U\$1: Conector RJ-22: Sirve para las transmisiones de datos a través de RS485.
- R4, R5: Resistencias de 3.3KOhm a 1/4W.
- D6, D7, D8, D9, D10, D11, D12, D13, D14, D15: Diodos de señal 1N4148.
- C3, C4, C5, C6: Condensadores electrolíticos de 1uF a 25V.
- U\$2: Conector serial DB-9 hembra.
- U7: Circuito Integrado Max232: Es un conversor de niveles de voltaje TTL-RS232.
- C18, C7, C12: Condensador cerámico 103 (0,01uF o 10nF).
- T2, T3: Transistor 2N3906.
- R6: Resistencia de 330 Ohm a 1/4W.
- DL1, DL2, DL3: Diodos Led de encendido, de recepción de datos y envío respectivamente.
- R8, R10: Resistencia de 220 Ohm a 1/4W.
- R7, R9: Resistencia de 2.7 KOhm a 1/4W.
- C1, C2: Condensadores Ceramicos de 22pF.
- R11: Resistencia de 10 KOhm a 1/4W.
- R12: Resistencia de 1.5 KOhm a 1/4W.
- Q1, Q2: Cristal Oscilador de 32.768 KHz.
- R2, R3: Resistencia de 1 KOhm a 1/4W.

- R1: Resistencia de 6.8 KOhm a 1/4W.
- SL1, SL2: Conectores para el sensor de temperatura DS1620: Este es un integrado de 8 pines, encargado de enviar la información de temperatura, al microcontrolador.
- U2: Circuito Integrado DS1307: Es un circuito integrado IC de 8 pines, el cual cumple la función de llevar la noción del tiempo en años, meses, semanas, horas, minutos y segundos y va conectado directamente al microcontrolador. Este IC debe contar con una batería auxiliar (B1), que le permite continuar con su funcionamiento normal en ausencia de energía eléctrica. Cabe destacar, que si la batería está descargada o no está conectada al integrado, éste no funciona correctamente aún si éste cuenta con el suministro normal de energía.
- G2: Pila de 3V.
- IC7: Microcontrolador Pic16f877A.
- IC4, IC5: Circuito Integrado 4094: Registro de desplazamiento de 8 bits, para mover los datos de la Memoria SRam DS1230Y a la etapa de potencia del microcontrolador.
- IC3: Memoria SRam DS1230Y: Es una memoria no volátil donde se almacenan los datos a mostrar.

Dado el diseño con tantos componentes, se divide este en 5 hojas, cada hoja del diseño corresponde a una etapa del producto. En el caso del diseño del impreso, se muestran diferentes capas del mismo para ver como quedara finalmente.

- Diagrama 1. Esquemático de la Tarjeta MicroPublik - Etapa de Control
- Diagrama 2. Esquemático de la Tarjeta MicroPublik - Etapa de Comunicación
- Diagrama 3. Esquemático de la Tarjeta MicroPublik - Etapa a la Salida de Potencia
- Diagrama 4. Esquemático de la Tarjeta MicroPublik - Fuentes de Alimentación
- Diagrama 5. Esquemático de la Tarjeta MicroPublik - Hoyos para fijación de Tarjeta
- Grafica 1. Impreso de la Tarjeta MicroPublik - Componentes
- Grafica 2. Impreso de la Tarjeta MicroPublik - Cara Superior
- Grafica 3. Impreso de la Tarjeta MicroPublik - Cara Inferior
- Grafica 4. Impreso de la Tarjeta MicroPublik - Componentes y ambas caras
- Grafica 5. Tarjeta MicroPublik v2.0

A continuación se reseña el Código Fuente de MicroPublik en Asembler, para que se pueda llevar a cabo su utilización en puesta:

```
list p=16F877A
include<p16f877A.inc>
include <tabla_car_norm.h>
;Area FSR reservada para almacenar datos temporales para escritura/lectura en EEPROM de datos y
Reloj Serial:
;20h - 2Fh
;Definicion de registros de propósito general (Variables) en Banco 0
cblock 0x28
;está funcionando bien, leyendo mensajes desde la flash del micro, sin la ram
;ajustar el scroll ;STATUS
;IRP RP1 RP0 TO# PD# Z DC C
RETARD ;PARA USO GENERAL DE LOS RETARDOS
COLMNASH ;PARA CONTAR EL NUMERO DE COLUMNAS ENVIADAS AL INFORMADOR EN LA
FRASE
COLMNASL
COLMNASHH ;PARA EL CONTEO DE LAS COLUMNAS DE ESPACIOS EN BLANCO
COLMNASLL
TPO_DESP
tiempo_desp
RETA_FLASH ;retardos
RETA_HOLD
RET_SCROLL
RET_WIPE
C_RETA_EFEC ;contadores de retardos
ROTA_SCROLL
ROTA_WIPE_UP
COL_SCROLL
```

4

```

COL_WIPE
COL_AUX      ;VARIABLE AUXILIAR PARA GUARDAR EL ESTADO DE LAS COLUMNAS
CARAUX2      ;CARACTER AUXILIAR PARA EL MANEJO DE LAS TABLAS
EFFECT_AUX   ;VARIABLE AUXILIAR PARA CADA EFECTO
VECES        ;NUMERO DE VECES QUE EJECUTA CADA EFECTO
NCOLUMNAS_H  ;CONTADOR DE COLUMNAS
NCOLUMNAS_L
bandera0
bandera1
bandeMod
bandera2
contcolenvL
contcolenvH
contbits
DATOCOL
DIR_RAM_H
caracter
DIRH
DIRL
datox
DirIniH
DirIniL
NCVH
NCVL
NCTH
NCTL
cursor
cursor ini
NCSEP
an_o
NCSEPAux
DESPLAZL
DESPLAZH
caraux
PCLATHtemp
DespOffsetL
DespOffsetH
cuenta
DIRECCIONR
RECEP
EEPROM
TRANSM
DATOPC
mes
dia_mes
dia_sem
hora
minutos
DIRVERHR_H
DIRVERHR_L
DIRVERFE_H
DIRVERFE_L
contesp
codascii
CODE_INFO
cuantos
cont_seg
segundos
posic_car
CUATRO      ;77
Wtemp
DIR_RAM_HL
DIRH_HT
DIRL_HT
rx_retardl
rx_retardh
rx_retardhh
temp_l
temp_h
STATUS_TEMP

```

3

endc

cblock 0xA0

DIRH_TMP_b2
DIRL_TMP_b2
DATORX_b2
DECENA_b2
UNIDAD_b2
DIRVERTMP_H_b2
dirvertmp_l_b2
conta4_b2
datotx_b2
dato_b2
endc

;Señales de control de los registros de desplazamiento (4094) del PANEL:

#DEFINE DATAS PORTA,0 ;TRAMA DE BITS DE DATOS
#DEFINE CLOCK PORTA,1 ;ÉSTE, CON UN FLANCO DE SUBIDA LEE U NUEVO BIT DE LA TRAMA DE BITS DE DATOS DE LA ENTRADA
#DEFINE STROBE PORTA,2 ;CARGA EN EL 4094 LO QUE HAY EN SU REGITRO DE ENTRADA
#DEFINE OE PORTC,2 ;HABILITA A LA SALIDA DEL 4094 EL DATO CARGADO EN ÉL (PWM)
;Señales de control de los registros de desplazamiento (4094) para enviar Dirección de la RAM (DIRH_DIRL):

#define STR94 PORTD,7
#define DAT94 PORTD,6
#define CLK94 PORTD,5
#define OE94 PORTD,4

;Señales de control de memoria RAM:

#define WE PORTD,1 ;Selección Lectura/Escritura de memoria RAM
#define OER PORTD,2 ;Habilita búfer de salida de datos de memoria RAM
#define CS PORTD,3 ;Selección (habilitación) de memoria RAM

;Señal de control para habilitar/deshabilitar transmisión serial al PC [RS485]:

#define HABTX PORTC,5

;Señales de control del Reloj de Tiempo Real DS1307:

#define SCL PORTC,3
#define SDA PORTC,4

;Señales de control del termómetro digital DS1620

#define RSTT PORTC,0 ;pin 3 ds1620 pin4 gnd
#define CLKT PORTC,1 ;pin 2 pin8 vdd
#define DQT PORTD,0 ;pin 1
#define TRIS_DQT TRISD,0

;Bits del registro EEPROM asociados con el bus I2C

BIT_IN EQU 7

BIT_OUT EQU 6

#define fuente2 bandera2,0
#DEFINE CODEVELO BANDERA2,1
#DEFINE HORA_AMPM BANDERA2,2
#define hora_militar bandera2,3
#DEFINE sirx_retard BANDERA2,4
#DEFINE TEMPE_HR_FE BANDERA2,5
#define fin_digit bandera2,6

;/#define wide_hold bandera2,7

#define one_second bandera1,0 ;indica que ha transcurrido 1 segundo

#define time_desp bandera1,1 ;Indica si se ha cumplido tiempo de desplazamiento horizontal.

#define fuente bandera1,2 ;Bandera modo de texto: 0 para NORMAL y 1 para NEGRITA.

#define fuente_ini bandera1,3 ;Indica estado fuente de texto al iniciar el refresco

#define fin_car bandera1,4 ;Fin de caracter enviado [sigue separación en blanco].

#define fin_car_ini bandera1,5 ;Fin de caracter inicial [sigue siguiente caracter]

#define aborto bandera1,6 ;Indica si se completó Número de Columnas Total del Panel

#define colvacias bandera1,7 ;Indica si existen columnas vacías o no en la rutina de refresco.

;/#DEFINE modowide BANDEM0D,0

#define modoRotate bandeMod,1 ;Modo sistema: ROTATE.

#define modoRotate2 bandeMod,2 ;Modo sistema:

#define modohold bandeMod,3 ;Modo sistema:

#define modowipe_up bandeMod,4 ;Modo sistema:

#define modoscroll bandeMod,5 ;Modo sistema:

#define modoflash bandeMod,6 ;Modo sistema:

#DEFINE MODOHOLDW BANDEM0D,7 ;

#define fuente_fe bandera0,0 ;iniciador de negrilla en los efectos

#define fuente_fe2 bandera0,1 ;iniciador de negrilla 2

5

4

[illegible]

7

```

    goto continuar
;Ahora busca dirección RAM donde se encuentra el primer caracter ascii para empezar [DirIni]:
call BUSCADIRINI
; call SETUPRAM ;No se encontró código de fin de mensaje, entonces graba mensaje por defecto.
CONTINUAR
    BSF MODOROTATE ;MODO ROTATE POR DEFECTO
clrf TMR1H
clrf TMR1L
bsf T1CON,TMR1ON
bsf INTCON,PEIE
bsf INTCON,GIE ;Habilita todas las interrupciones
;Configurar USART para comunicación serial generando interrupción.
clrf RCSTA
bsf RCSTA,SPEN ;Configura RC7(RX) y RC6(TX) como pines del puerto serial.
bsf STATUS,RP0
bsf PIE1,RCIE ;Habilita interrupción para recepción de USART
bcf TXSTA,SYNC ;Selecciona modo ASINCRONO de la USART.
bcf TXSTA,TX9 ;Dato de transmisión de 8 bits.
bsf TXSTA,BRGH ;Escoge velocidad de comunicación ALTA.
BSF TXSTA,TXEN ;HABILITA TRANSMISIÓN
movlw .129 ;Establece velocidad de comunicación (baudios).
movwf SPBRG ;Para un cristal de 20 MHz.
; movlw .77 ;Establece velocidad de comunicación (baudios).
; movwf SPBRG ;Para un cristal de 12 MHz.PRUEBABA
bcf STATUS,RP0
bsf RCSTA,CREN ;Habilita recepción continua.
bcf RCSTA,RX9 ;Dato de recepción de 8 bits.
;***** prueba trasmisión
; movlw 0x31
; bcf PIR1,TXIF ;Restaura el flag del transmisor
; movwf TXREG ;Almacena el byte a transmitir
; bsf STATUS,RP0
;Tx_Dato_Waitx1
; btfss TXSTA,TRMT ;Byte transmitido ??
; goto Tx_Dato_Waitx1 ;No, esperar
; bcf STATUS,RP0
; movlw 0x32
; bcf PIR1,TXIF ;Restaura el flag del transmisor
; movwf TXREG ;Almacena el byte a transmitir
; bsf STATUS,RP0
;Tx_Dato_Waitx2
; btfss TXSTA,TRMT ;Byte transmitido ??
; goto Tx_Dato_Waitx2 ;No, esperar
; bcf STATUS,RP0
; bsf pclath,3
; bsf pclath,2
; call respuesta
; CLRF PCLATH
;*****espera de dos segundos
espera clrwdt
MOVLW 0xFF
XORWF TMR1H,W
BTFSS STATUS,Z
goto espera
esp_ov ;ESPERA EL OVERFLOW
CLRWDT
MOVLW 0x00
XORWF TMR1H,W
BTFSS STATUS,Z
goto esp_ov
incf cont_seg,1
movlw .76
xorwf cont_seg,W
btfss STATUS,Z
goto espera
clrf cont_seg
;Uso del PWM
; bsf STATUS,RP0 ;switch bank 1
; movlw periodo ; cargamos el valor del período en PR2
; movwf PR2
; bcf STATUS,RP0 ;switch bank 0

```

6

```

; movlw duty ; cargamos valor del duty en CCPR1L
; movwf CCPR1L
pwmon
; bsf T2CON,2 ;TMR2 activado
; movlw 0x0c ;cargamos setup módulo
; movwf CCP1CON ;PWM
; Estas líneas para desactivar la salida
;pwmon clrf CCP1CON
; bcf T2CON,2 ;Desactivamos TMR2
; bcf PWM ;esta línea en caso que al finalizar la emisión el pin quede a 1.
BSF OE
goto ciclo

```

```

-----
ACTUALZR_HT

```

```

CALL ACTUALIZA_H_T

```

```

BCF TEMPE_HR_FE

```

```

*****
*****
*****
*****
*****

```

```

CICLO

```

```

BTFSC TEMPE_HR_FE

```

```

GOTO ACTUALZR_HT

```

```

; movlw duty ; recargamos valor del duty en CCPR1L

```

```

; movwf CCPR1L

```

```

movf tmr1h,w ;control del timer overflow

```

```

addlw .80

```

```

movwf tmr1h

```

```

WIT_OVRFW0 ;ESPERA EL OVERFLOW

```

```

clrwdt

```

```

MOVLW 0xFF

```

```

XORWF TMR1H,W

```

```

BTFSS STATUS,Z

```

```

goto WIT_OVRFW0

```

```

WIT_OVRFW

```

```

CLRWDW

```

```

MOVLW 0x00

```

```

XORWF TMR1H,W

```

```

BTFSS STATUS,Z

```

```

goto WIT_OVRFW

```

```

*****
*****

```

```

TIMER_OVERFLOW

```

```

bsf INTCON,PEIE

```

```

bsf INTCON,GIE ;Habilita todas las interrupciones

```

```

clrwdt

```

```

bcf status,z

```

```

btfsc modorotate2

```

```

bsf modorotate

```

```

;Se pregunta si está en modo Rotate ???

```

```

incf tiempo_desp,1

```

```

movf TPO_DESP,W

```

```

xorwf tiempo_desp,W

```

```

btfss STATUS,Z

```

```

goto CHSEG

```

```

bsf time_desp ;Cumplido tiempo de desplazamiento.

```

```

clrf tiempo_desp

```

```

CHSEG

```

```

incf cont_seg,1

```

```

movlw .38 ;OJO, con preescal de Timer 1 en 1:2 (tpo_rebase = 13,1 ms). Entonces con preescala de 1:2,

```

```

xorwf cont_seg,W ;la interrupción ocurre cada: 2*tpo_rebase = 26,2 ms. Para un tiempo transcurrido de un segundo,

```

```

btfss STATUS,Z ;el número de veces que se debe detectar la interrupciones de 38 ! [38*26.2ms = 997ms]

```

```

goto SIGREF_XX

```

```

;Aquí se lee la hora/fecha y se almacena en RAM si existe el atributo:

```

```

BSF ONE_SECOND

```

```

clrf cont_seg

```

```

bcf status,z

```

```

incf segundos,1

```

4

```

movlw .15
xorwf segundos,w
btfss status,z
goto SIGREF_XX
clrf segundos
bsf INTCON,PEIE
bsf INTCON,GIE ;Habilita todas las interrupciones
;Configurar USART para comunicación serial generando interrupción.
clrf RCSTA
bsf RCSTA,SPEN ;Configura RC7(RX) y RC6(TX) como pines del puerto serial.
bsf STATUS,RP0
bsf PIE1,RCIE ;Habilita interrupción para recepción de USART
bcf TXSTA,SYNC ;Selecciona modo ASINCRONO de la USART.
bcf TXSTA,TX9 ;Dato de transmisión de 8 bits.
bsf TXSTA,BRGH ;Escoge velocidad de comunicación ALTA.
BSF TXSTA,TXEN
movlw .129 ;Establece velocidad de comunicación (baudios).
movwf SPBRG ;Para un cristal de 20 MHz.
; movlw .77 ;Establece velocidad de comunicación (baudios).
; movwf SPBRG ;Para un cristal de 12 MHz.PRUEBABA
bcf STATUS,RP0
bsf RCSTA,CREN ;Habilita recepción continua.
bcf RCSTA,RX9 ;Dato de recepción de 8 bits.
bcf status,rp1
btfss PIR1,RCIF ;PARA DESCARGAR EL BUFFER
GOTO SIGREF_XX
bsf pclath,3
bsf pclath,2
call RutinaINT
; bsf STATUS,RP0 ;HABILITA RX SERIAL POR SILAS
; bsf PIE1,RCIE
; bcf STATUS,RP0
SIGREF_XX
; btfss modoRotate ;El sistema se encuentra en modo ROTATE?
; goto sigref ;No
; btfsc time_desp ;Cumplido tiempo de desplazamiento?
; goto SIGREF ;SI.
NOVRFLW ;ESPERA PARA QUE SE REINICIE EL TIMER
clrwdt
MOVLW 0xFF
XORWF TMR1H,W
BTFSC STATUS,Z
goto NOVRFLW ;No. SE HA COMPLETADO EL TIEMPO DE ESPERA DEL TIMER
; goto FIN_TIM
SIGREF
call REFRESCA
; bcf PIR1,TMR1IF
btfss modoRotate ;El sistema se encuentra en modo ROTATE?
goto FIN_TIM ;No
MODROT
clrwdt
btfss time_desp ;Cumplido tiempo de desplazamiento?
goto FIN_TIM ;No.
bcf time_desp
btfsc colvacías ;Existen columnas vacías? prueba debe ser activado
goto INCCURSORINI ;NO!, entonces se incrementa valor del cursor inicial (incluye separación de
caracteres)
;Sí, entonces decrementa el número de columnas vacías:
clrwdt
bcf STATUS,C
movlw .1
subwf NCVL,1
btfss STATUS,Z
goto MIRANEG
movlw .0
xorwf NCVH,W
btfss STATUS,Z
goto FIN_TIM
; clrf cursor_ini
bsf colvacías ;Ya no hay columnas vacías!!
; call BUSCADIRINI

```

8..

12

```

; CLRF CURSOR_INI
goto FIN_TIM
MIRANEG
    btfsc STATUS,C ;resultado de resta anterior negativo?
    goto FIN_TIM ;No!
    decf NCVH,1
    goto FIN_TIM
INCCURSORINI ;cursor ini es la posicio inicial de la columna del caracter inicial.
;Al final de un caracter inicial, se incrementa Dirección Inicial RAM y Cursor Inicial = 0.
;Lee dato de columna en posición cursor_ini de caracter inicial [en DirIniH_DirIniL] para
;determinar si es fin de caracter inicial:
clrwdt
    movf DirIniH,W
    movwf DIRH
    movf DirIniL,W
    movwf DIRL
    call LEE_RAM
call calcula_offset ;CALCULA LA POSICION DEL CARACTER EN LA TABLA DE CARACTERES QUE
ESTÁ AL FINAL
    movf DespOffsetL,W ;Y QUEDA GUARDADA EN DespOffsetH y DespOffsetL
    movwf DESPLAZL
    movf DespOffsetH,W
    movwf DESPLAZH
    movf cursor_ini,W
    addwf DESPLAZL,1
btfsc STATUS,C
    incf DESPLAZH,1
bcf fuente
    btfsc fuente_ini
    bsf fuente
    call BUSCACOL
;Chequea si es fin de caracter [separación] para incrementar Dirección Inicial de RAM
    btfsc DATOCOL,0
    goto FINCARINI ;Sí es fin de caracter inicial.
    incf cursor_ini,1 ;No es fin de caracter inicial.
    goto FIN_TIM
FINCARINI
    clrwdt
    clrf cursor_ini
    incf DirIniL,1
    btfsc STATUS,Z
    incf DirIniH,1
MARCAFMSGE
    movf DirIniH,W
    movwf DIRH
    movwf DIRH_HT
    movf DirIniL,W
    movwf DIRL
    movwf DIRL_HT
    call LEE_RAM
    movlw CODE_TNORM ;Código de texto NORMAL
    xorwf character,W
    btfss STATUS,Z
    goto salxx1
    CALL code_tnorm_cod
    goto FINCARINI
salxx1
    clrwdt
    movlw CODE_TNEG ;Código de texto NEGRITA
    xorwf character,W
    btfss STATUS,Z
    goto SALXX1X
    call code_tneg_cod
    goto FINCARINI
SALXX1X
    movlw CODE_TNEG2 ;Código de texto NEGRITA2
    xorwf character,W
    btfss STATUS,Z
    goto salxx2
    call code_tneg2_cod
    goto FINCARINI

```

-9-

```

salxx2
    clrwdt
    movlw CODE_HORA1    ;Código de HORA1
xorwf caracter,W
    btfss STATUS,Z
goto salxx3
    call code_hora1_cod
    goto FINCARINI
salxx3
    clrwdt
    movlw CODE_HORA2    ;Código para mostrar la HORA2
xorwf caracter,W
    btfss STATUS,Z
goto salxx4
    call code_hora2_cod
    goto FINCARINI
salxx4
    clrwdt
    CALL BUSCA_FECHACODE
    BTFSC FECHACODE
goto FINCARINI
salxxx4
    movlw temperatura
    xorwf caracter,W
    btfss STATUS,Z
goto salxx5
    call temperatura_cod
    goto FINCARINI
salxx5
    movlw CODE_FINEFECTO ;Código para FIN de mensajes!
xorwf caracter,W
    btfss STATUS,Z
    goto salxx6
    goto FINCARINI
salxx6
    clrwdt
    movlw CAR_ROTATE
    xorwf caracter,W
    btfss STATUS,Z
    goto salxx611
    call car_rotate_cod
goto fincarini
salxx611
    call code_velo      ;codigo de velocidad
btfss codevelo
    GOTO salxx7
    CALL set_codevelo_rot
goto fincarini
salxx7
    clrwdt
    movlw CODE_FINMENSAJE ;Código para FIN de mensajes!
xorwf caracter,W
    btfss STATUS,Z
goto FIN_TIM          ;Es código ASCII de caracter!
FINALMSG
    clrwdt ;FIN de mensajes!
    call BUSCADIRINI
    ;Establece número de columnas vacías para separación de mensajes:
    clrf tiempo_desp
FIN_TIM
    bsf STATUS,RP0
    bsf PIE1,RCIE
    bcf STATUS,RP0
    BCF ONE_SECOND
    goto CICLO
;*****
;*****
;*****
; AQUÍ SE ENVÍA TODA LA TRAMA COMPLETA PARA LLENAR EL INFORMADOR, ES DECIR, SE
; ACTUALIZA.
;*****

```

```

;*****
REFRESCA      ;
  clrwdt
  clrf contcolenvL ;contador columnas enviadas
  clrf contcolenvH
  bcf STROBE
  bcf status,z
  movlw .0
  xorwf bandemod,w
  BTFSC status,z
GOTO SIGA_ROTAR ;
  btfss modorotate ;El sistema se encuentra en modo ROTATE?
  GOTO SIGARAM ;si no hay modo rotate activado salte a la lectura de la ram
siga_rotar
  btfsc colvacias ;Existen columnas vacías para enviar inicialmente?
  goto SIGARAM ;NO!
ENVCOLVAC
  clrwdt ;SI!, todavía.
  clrf DATOCOL
  call ENVIA7
  incf contcolenvL,1
  btfsc STATUS,Z
  incf contcolenvH,1
  ; [Cont column enviadas] = [Número de columnas vacías existentes]?
  movf NCVH,W
  xorwf contcolenvH,W
btfss STATUS,Z
  goto ENVCOLVAC
  movf NCVL,W
  xorwf contcolenvL,W
  btfss STATUS,Z
  goto ENVCOLVAC
  clrwdt
SIGARAM
  clrwdt
  movf cursor_ini,W
  movwf cursor
  movf DirIniH,W
  movwf DIRH
  movf DirIniL,W
  movwf DIRL
  bcf fuente
  btfsc fuente_ini
  bsf fuente
CICLECT
  clrwdt
  call LEE_RAM ;Lee código de la posición de memoria DIRH_DIRL de RAM. El dato leído se
almacena en 'caracter'
  movf dirl,w
  movwf dirl_ht
  movf dirh,w
  movwf dirh_ht
  movlw code_tnorm
  xorwf caracter,W
btfsc STATUS,Z
  goto TEXTONORM ;Código de texto NORMAL
  movlw code_tneg
  xorwf caracter,W
btfsc STATUS,Z
  goto TEXTONEGR ;Código de texto NEGRITA
  movlw code_tneg2
  xorwf caracter,W
btfsc STATUS,Z
  goto TEXTONEGR2 ;Código de texto NEGRITA
  movlw code_hora1
  xorwf caracter,W
  btfsc STATUS,Z
  goto MOSTHORA ;Código para mostrar la HORA
  movlw code_hora2
  xorwf caracter,W
  btfsc STATUS,Z

```

13

```

goto MOSTHORA2 ;Código para mostrar la HORA2
MOVLW TEMPERATURA
XORWF CHARACTER,W
btfsc STATUS,Z
GOTO COD_TEMPERATURA ;Código para TMP
CALL BUSCA_FECHACODE
BTFSC FECHACODE
GOTO CONTI_CICLECT
; movlw 0x09
; xorwf character,W
; btfsc STATUS,Z
; goto MOSTFECHA ;Código para mostrar la FECHA
;*****
;*****
MOVLW CAR_HOLD
xorwf character,W
btfsc STATUS,Z
goto EF_HOLD ;Código para EFECTO HOLD
MOVLW CAR_FLASH
xorwf character,W
btfsc STATUS,Z
goto EF_FLASH ;Código para EFECTO FLASH
MOVLW CAR_SCROLL
xorwf character,W
btfsc STATUS,Z
goto EF_SCROLL ;Código para EFECTO SCROLL
MOVLW CAR_WIPEUP
xorwf character,W
btfsc STATUS,Z
goto EF_WIPE_UP ;Código para EFECTO WIPE UP
; MOVLW CAR_WIDE
; xorwf character,W
; btfsc STATUS,Z
; goto EF_WIDE ;Código para EFECTO WIPE UP
MOVLW CAR_ROTATE
xorwf character,W
btfsc STATUS,Z
GOTO EF_ROTATE ;Código para EFECTO ROTATE
;*****
;*****

movlw code_finefecto ;Código para FIN de EFECTOS
xorwf character,W
btfsc STATUS,Z
GOTO CONTI_CICLECT
movlw code_finmensaje
xorwf character,W
btfsc STATUS,Z
goto FINALMSGXX ;Código para FIN de mensajes!
call code_velo ;busca codigo de velocidad
btfsc codevelo
goto inc_digi
GOTO MOSTCAR
INC_DIGI ;OTROS: 1 CAR_NORM; 3 CAR_NEG; 7 HORA; 6,8,9;FECHA
call set_codevelo_rot
CALL INC_DIRLH
GOTO CICLECT
;ENTONCES ES CÓDIGO ASCII DE CHARACTER!
;SÉ REINICIAN LOS REGISTROS QUE SE USAN EN LOS EFECTOS
MOSTCAR clrwdt ;MODO ROTATE
; movlw duty ; se incrementa el valor del duty en CCPR1L
; addlw .100 ;para mejorar la visualizacion del rotate
; MOVLW PERIODO ;PARA QUE EL DUTY SEA IGUAL AL PERIODO
; movwf CCPR1L
call ENVCOLCAR ;ENVIA COLUMNAS DEL CHARACTER CORRESPONDIENTE. SI SE
ACABARON LAS COLUMNAS => BSET aborto
btfsc aborto
goto SALIREF ;SE LLENÓ EL INFORMADOR (YA NO HAY COLUMNAS VACÍAS)
CALL INC_DIRLH
clrf cursor
goto CICLECT

```

```

CONTI_CICLECT
    CALL INC_DIRLH
    clrf cursor
goto CICLECT
;*****
FINALMSGXX
    clrwdt
;Se envían espacios en blanco, es decir, la separación entre el mensaje y la repetición del mismo.
    bcf fuente
    bcf fuente_fe
    movf NCSEP,W
    movwf NCSEPAux
    clrf DIRH
    clrf DIRL
    clrf cursor
    BTFSC MODOROTATE
    goto CICLECT
    CALL BUSCADIRINI
    BCF STATUS,Z
    INCF NCVL,1
    BTFSC STATUS,Z
    INCF NCVH,1
    RETURN
;*****
TEXTONORM
    clrwdt
    bcf fuente
    bcf fuente_fe
    BTFSS MODOROTATE
    bcf fuente_ini
    GOTO CONTI_CICLECT
;*****
TEXTONEGR
    clrwdt
    bsf fuente
    bsf fuente_fe
    GOTO CONTI_CICLECT
;*****
TEXTONEGR2
    clrwdt
    bsf fuente2
    bsf fuente_fe2
    GOTO CONTI_CICLECT
;*****
MOSTHORA
    CALL CODE_HORA1_COD
    GOTO CONTI_CICLECT ;No se ha terminado el refresco, entonces continúa leyendo datos de RAM!
;*****
MOSTHORA2
    CALL CODE_HORA2_COD
    GOTO CONTI_CICLECT ;No se ha terminado el refresco, entonces continúa leyendo datos de RAM!
;*****
COD_TEMPERATURA
    CALL TEMPERATURA_COD
    GOTO CONTI_CICLECT
;*****
EF_HOLD
    MOVF DIRL,W ;COLOCA EL PUNTERO DE LA FRASE EN EL INICIO DE LA FRASE CON EL EFECTO
    MOVWF DIRINIL
    MOVF DIRH,W
    MOVWF DIRINI
    BTFSC MODOHOLD
    GOTO SIG_HOLD0
    MOVF RETA_HOLD,W
    MOVWF C_RETA_EFEC
SIG_HOLD0
    clrwdt
    CALL LIMPIA_PANEL ;LIMPIA EL PANEL
    clrf cursor
    CLRF BANDEMODO

```

15

```

BSF  MODOHOLD
BCF  STATUS,C    ;BORRA EL BIT DE CARRY
CALL ENVIA_FRASE
BTFSC ONE_SECOND
DECFSZ C_RETA_EFEC,1    ;DECREMENTE Y SALTE SI CERO. DECREMENTA ERTA_HOLD Y
SI CERO, FIN_HOLD
GOTO SIG_HOLD
    GOTO FIN_HOLD    ;FIN DE CICLO HOLD
SIG_HOLD
NOP
NOP
NOP
BSF  STROBE
RETURN
FIN_HOLD
GOTO FIN_EFECTS
;*****
EF_FLASH
clrwdt
    MOVF  DIRL,W    ;COLOCA EL PUNTERO DE LA FRASE EN EL INICIO DE LA FRASE CON EL
EFFECTO
MOVWF DIRINIL
    MOVF  DIRH,W
    MOVWF DIRINIH
    BTFSC MODOFASH
    GOTO  SIG_FLASH0
    MOVF  RETA_FLASH,W
    MOVWF C_RETA_EFEC
SIG_FLASH0
    CLRF  BANDEMODO
    BSF  MODOFASH    ;MODO_FLASH
CALL  LIMPIA_PANEL    ;LIMPIA EL PANEL
    BCF  STATUS,Z
    DECFSZ C_RETA_EFEC,1
    GOTO  FOLLOW_FLASH
CAMBIO_FLASH
    BCF  STATUS,Z
    INCF  EFFECT_AUX,1
    MOVF  RETA_FLASH,W
    MOVWF C_RETA_EFEC
FOLLOW_FLASH
BTFSC EFFECT_AUX,0    ;TESTEA SI EL REGISTRO CONTADOR DE VECES FLASH ES PAR O
IMPAR.
GOTO  FLASH_IMPAR
FLASH_PAR
    clrf  cursor
call  ENVIA_FRASE    ;ENVIA LA FRASE
    MOVLW .10    ;PARA CONTRARESTAR EL TIEMPO DE MAS QUE SE GENERA EN EL MODO
PAR
    CALL  RETA    ;CON LIMPIA PANEL
    goto  flas_ya
FLASH_IMPAR
    call  ENVIA_FRASE    ;ENVIA LA FRASE
    CALL  LIMPIA_PANEL    ;LIMPIA EL PANEL
    goto  flas_ya
flas_ya
    BSF  STROBE
BCF  STATUS,Z
    MOVLW VECES_FLASH    ;CAMBIAR X CONSTANTE
    XORWF EFFECT_AUX,W
    BTFSC STATUS,Z
    GOTO  FIN_FLASH
    RETURN
FIN_FLASH
;  MOVLW TIME_FLASH
;  MOVWF RETA_FLASH
GOTO  FIN_EFECTS
;*****
EF_SCROLL
    MOVF  DIRL,W    ;COLOCA EL PUNTERO DE LA FRASE EN EL INICIO DE LA FRASE CON EL
EFFECTO

```

```

MOVWF DIRINIL
MOVF DIRH,W
MOVWF DIRINIH
BTFSC MODOScroll
GOTO SIG_SCROLL
MOVF RET_SCROLL,W
MOVWF C_RETA_EFEC
SIG_SCROLL
    clrwdt
    clrf cursor
    CLRF BANDEM0D
    BSF MODOScroll
    CALL LIMPIA_PANEL
SCRL
    BCF STATUS,C ;BORRA EL BIT DE CARRY
    BCF STROBE
    CALL ENVIA_FRASE
    MOVLW .90
    CALL RETA
    BCF STATUS,Z
    INCF EFECT_AUX,1
    MOVLW TIME_SCROLL
    XORWF EFECT_AUX,W
    BTFSS STATUS,Z
    GOTO SIG_SCROLL
    CLRF EFECT_AUX
    BCF STATUS,Z
    MOVLW .9
    XORWF ROTA_SCROLL,W
    BTFSC STATUS,Z
    GOTO SCRL_PAUSA
SCRL_SIGA2
    INCF COL_SCROLL,1
    DECFSZ ROTA_SCROLL,1 ;VERIFICA SI YA SE REALIZARON TODOS LO CORRIMIENTOS
    GOTO SIG_SCROLL
    GOTO FIN_SCROLL
SCRL_PAUSA
    DECFSZ C_RETA_EFEC,1
    GOTO SIG_SCROLL
    GOTO SCRL_SIGA2
SIG_SCROLL
    BSF STROBE
    RETURN
FIN_SCROLL
    MOVLW .16
    MOVWF ROTA_SCROLL
    MOVLW .1
    MOVWF COL_SCROLL ;GUARDA LA VARIABLE DE ROTACIÓN DE LAS COLUMNAS
    GOTO FIN_EFFECTS
;*****
EF_WIPE_UP
    MOVF DIRL,W ;COLOCA EL PUNTERO DE LA FRASE EN EL INICIO DE LA FRASE CON EL
    EFECTO
    MOVWF DIRINIL
    MOVF DIRH,W
    MOVWF DIRINIH
    BTFSC MODOHOLDW
    GOTO SIG_WIPEO
    MOVF RET_WIPE,W
    MOVWF C_RETA_EFEC
    SIG_WIPEO
    BTFSS MODOWIPE_UP
    GOTO WIPE_MODD
    BTFSC MODOHOLDW
    GOTO FIN_WIPE
WIPE_MODD
    clrwdt
    clrf cursor
    CLRF BANDEM0D
    BSF MODOWIPE_UP
    CALL LIMPIA_PANEL

```

17

```

WIPE_SCRL
  BCF STATUS,C ;BORRA EL BIT DE CARRY
  BCF STROBE
  CALL ENVIA_FRASE
  MOVLW .90
  CALL RETA
  INCF EFFECT_AUX,1
  MOVLW TIME_WIPE
  XORWF EFFECT_AUX,W
  BTFSS STATUS,Z
  GOTO SIGA_WIPE
  INCF COL_WIPE,1
  CLRF EFFECT_AUX
DECFSZ ROTA_WIPE_UP,1 ;VERIFICA SI YA SE REALIZARON TODOS LO CORRIMIENTOS
  GOTO SIGA_WIPE
  GOTO FIN_WIPP
SIGA_WIPE
  BSF STROBE
  RETURN
FIN_WIPP
  BSF STROBE
  BSF MODOHOLDW
  CLRF EFFECT_AUX
  RETURN
FIN_WIPE
  CALL LIMPIA_PANEL
clrf cursor
  BCF STATUS,C ;BORRA EL BIT DE CARRY
  CALL ENVIA_FRASE
  INCF EFFECT_AUX,1
MOVLW TIME_WIPE
  XORWF EFFECT_AUX,W
  BTFSS STATUS,Z
  GOTO SIG_HOLD_WIP
CLRF EFFECT_AUX
  DECFSZ C_RETA_EFEC,1 ;DECREMENTE Y SALTE SI CERO. DECREMENTA ERTA_HOLD Y SI
CERO, FIN_HOLD
  GOTO SIG_HOLD_WIP
  GOTO FIN_HOLD_WIP ;FIN DE CICLO HOLD
SIG_HOLD_WIP
  NOP
  NOP
  NOP
  BSF STROBE
  RETURN
FIN_HOLD_WIP
  MOVLW .7
  MOVWF ROTA_WIPE_UP
  MOVLW .1
  MOVWF COL_WIPE ;GUARDA LA VARIABLE DE ROTACIÓN DE LAS COLUMNAS
  GOTO FIN_EFECTS
;*****
;*****
FIN_EFECTS
  CLRF BANDEMODO ;LIMPIA MODOS DE EFECTOS
  CLRF EFFECT_AUX
  bcf retaefec
  BCF FUENTE_FE
  BTFSC FUENTE
  BSF FUENTE_FE
  BUSCA_CARFIN
  CLRWDI
  CALL LEE_RAM
  BSF PCLATH,3
  CALL BUS_CAR_ESP
  clrf pclath
  CALL INC_DIRLH
  BCF STATUS,Z ;PARA BUSCAR EL FINAL DE CARACTER. ULTIMA MODIFICACION.
  MOVLW 0xFE
  XORWF CARACTER,W
  BTFSS STATUS,Z

```

```

GOTO BUSCA_CARFIN
CARFIN
BCF STATUS,Z
; CALL INC_DIRLH
MOVF DIRL,W ;COLOCA EL PUNTERO DE LA FRASE EN EL INICIO DE LA FRASE CON EL
EFECTO
MOVWF DIRINIL
MOVF DIRH,W
MOVWF DIRINI
BSF STROBE
BSF OE
CALL BUSCADIRINI_ROTA
RETURN
;*****
EF_ROTATE
    clrwdt
    clrf tiempo_desp
CALL INC_DIRLH
CLRF CURSOR
    clrf cursor_ini
; BTFSC MODOROTATE
; GOTO CICLECT
    bcf fuente_ini
    btfsc fuente_fe
    bsf fuente_ini
    MOVF DIRL,W ;COLOCA EL PUNTERO DE LA FRASE EN EL INICIO DE LA FRASE CON EL
EFECTO
MOVWF DIRINIL
MOVF DIRH,W
MOVWF DIRINI
BSF MODOROTATE2
CALL BUSCADIRINI_ROTA
RETURN
;***** SALIREF clrwdt ;Se
completó el total de columnas!
    bsf STROBE ;VISUALIZA LO QUE SE ENVIÓ
    bsf OE
    return
;*****
;*****
ENVIA_FRASE
    bcf fuente
    btfsc fuente_fe
    bsf fuente
    CLRF COLMNASL ;CONTADORES DE COLMNAS ENVIADAS
    CLRF COLMNASH
    MOVF NCTL,W
    MOVWF COLMNASLL ;CONTADORES DE COLMNAS ENVIADAS
    MOVF NCTH,W
    MOVWF COLMNASHH ;CONTADORES DE COLMNAS ENVIADAS
    CALL INC_DIRLH ;PARA INICIAR EN EL SIGUIENTE CARACTER AL DE IDENTIFICADOR DE
EFECTO HOLD;
DIGITO
; BCF STATUS,C ;SE ENVIA UN ESPACIO
; CLRF DATOCOL
; CALL ENVIA7
    CALL LEE_RAM ;VARIABLE CARACTER = CARACTER GUARDADO EN RAM A SER ENVIADO
;DIRECCIONADO POR DIRL Y DIRH
    movlw code_tnorm
    xorwf caracter,W
    btfsc STATUS,Z
    goto FRA_TEXTONORM ;Código de texto NORMAL
    movlw code_tneg
    xorwf caracter,W
    btfsc STATUS,Z
    goto FRA_TEXTONEGR ;Código de texto NEGRITA
    movlw code_tneg2
    xorwf caracter,W
    btfsc STATUS,Z
    goto FRA_TEXTONEGR2 ;Código de texto NEGRITA
    movf dirh,w

```

```

movwf dirh_ht
movf dirl,w
movwf dirl_ht
movlw code_hora1
xorwf caracter,W
btfsc STATUS,Z
goto fra_mosthora1 ;Código de hora1
movlw code_hora2
xorwf caracter,W
btfsc STATUS,Z
goto FRA_MOSTHORA2 ;Código para mostrar la HORA
movlw temperatura
xorwf caracter,W
btfsc STATUS,Z
goto fra_mosttempe ;Código de temperatura
call busca_fechacode
btfsc fechacode
goto SIG_DIGITO_EF ;Código para mostrar la FECHA
MOVLW CODE_FINEFECTO
xorwf caracter,W
btfsc STATUS,Z
goto FN_DI ;Código para FIN de mensajes!
movlw code_finmensaje
xorwf caracter,W
btfsc STATUS,Z
goto FN_DI ;Código para FIN de mensajes!
call code_velo ;codigo de velocidad
btfss codevelo
goto conti_digi ;DIGITO ASCII. VISUALIZA EN EL EFECTO CORRESP
goto set_code_velo
; GOTO DIGITO
SIG_DIGITO_EF
CALL INC_DIRLH
BTFSC TEMPE_HR_FE
CALL ACTUALIZA_H_T ;ACTUALIZA
clrf cursor
GOTO DIGITO
;*****
set_code_velo
btfsc retaefec
goto fn_setvel
bsf retaefec
btfsc modohold
goto set_velhold
btfsc modoflash
goto set_velflash
btfsc modoscroll
goto set_velscroll
btfsc modowipe_up
goto set_velwipe
; btfsc modowide
; goto set_velwide
goto fn_setvel
set_velhold
movf codascii,w
MOVWF RETA_HOLD
MOVWF C_RETA_EFEC
goto fn_setvel
set_velflash
movf codascii,w
MOVWF RETA_FLASH
BCF STATUS,C
RLF RETA_FLASH,W ;MULTIPLICACION POR 2
MOVWF RETA_FLASH
MOVWF C_RETA_EFEC
goto fn_setvel
set_velscroll
movf codascii,w
MOVWF RET_SCRL
BCF STATUS,C
RLF RET_SCRL,W

```

20

```

    MOVWF RET_SCRL
    MOVWF C_RETA_EFEC
    goto fn_setvel
set_velwipe
    movf codascii,w
    MOVWF RET_WIPE
    BCF STATUS,C
    RLF RET_WIPE,W
    ADDWF RET_WIPE,W ;*3
    MOVWF RET_WIPE
    MOVWF C_RETA_EFEC
    goto fn_setvel
;set_velwide
;    movf codascii,w
;    MOVWF RET_WIDE
;    BCF STATUS,C
;    RLF RET_WIDE,W
;    ADDWF RET_WIDE,W ;*3
;    MOVWF RET_WIDE
;    MOVWF C_RETA_EFEC
;    goto fn_setvel
fn_setvel
    CALL INC_DIRLH
    CALL INC_DIRLH ;para eliminar el espacio que llega con el código de velocidad
goto digito
;*****
set_codevelo_rot
; MOVF CODASCII,W
; XORWF TPO_DESP,1
; BTFSS STATUS,Z
; CLRF TIEMPO_DESP
; MOVF CODASCII,W
; MOVWF TPO_DESP
    btfss time_desp
    return
    MOVF CODASCII,W
    MOVWF TPO_DESP
CLRF TIEMPO_DESP
    return
;+++++
FRA_TEXTONORM
call code_tnorm_cod ;
goto SIG_DIGITO_EF
;*****
FRA_TEXTONEGR
call code_tneg_cod
goto SIG_DIGITO_EF
;*****
FRA_TEXTONEGR2
call code_tneg2_cod
goto SIG_DIGITO_EF
;*****
FRA_MOSTHORA1
call CODE_HORA1_COD
goto SIG_DIGITO_EF ;No se ha terminado el refresco, entonces continúa leyendo datos de RAM!
;*****
FRA_MOSTHORA2
call CODE_HORA2_COD
goto SIG_DIGITO_EF ;No se ha terminado el refresco, entonces continúa leyendo datos de RAM!
;*****
fra_mosttempe
call temperatura_cod
goto SIG_DIGITO_EF
;*****
CONTI_DIGI
BCF STATUS,Z
CALL INC_DIRLH
CALL CALCULA_OFFSET ;CALCULA POSICIÓN OFFSET EN LA TABLA DE CARACTERES DEL
FINAL,PARA EL CARÁCTER 'CARACTER'
movf DespOffsetL,W
movwf DESPLAZL

```

21

```

movf  DespOffsetH,W
movwf DESPLAZH
movf  DespOffsetL,W
addwf cursor,W
movwf DESPLAZL
btfsc STATUS,C
incf  DESPLAZH,1
bcf   fin_digit
COL      ;CICLO DE BUSQUEDA DE LAS COLUMNAS DEL CARACTER
btfsc fin_digit
GOTO  DIGITO
clrwdt
INCF  COLMNASL,1      ;CONTADOR DE INCREMENTO DE COLUMNAS
BTFSC STATUS,Z
INCF  COLMNASH,1
BCF   STATUS,Z
MOVLW .0
XORWF COLMNASLL,w
BTFSC STATUS,Z
DECF  COLMNASHH,1
DECF  COLMNASLL,1      ;CONTADOR DE COLUMNAS SOBRANTES
MOVLW 0xFF             ;SE ACABARON LAS COLUMNAS?
XORWF COLMNASLL,W
BTFSS STATUS,Z
GOTO  SIGA_COLL
BCF   STATUS,Z
MOVLW 0xFF
XORWF COLMNASHH,W
BTFSC STATUS,Z
RETURN      ;SI YA NO HAY COLUMNAS DISPONIBLES RETORNAR
; GOTO  SIGA_COLL
SIGA_COLL
bcf   fin_digit
call  BUSCACOL      ;
BTFSC DATOCOL,0
bsf   fin_digit      ;indicador de fin de digito
btfsc fin_digit
clrf  datocol
; btfsc modowide
; goto  env_wide
BTFSC MODOSROLL      ;VERIFICAR SI ESTA ACTIVADO EL SCROLL
GOTO  ENV_SCROLL
BTFSC MODOWIPE_UP      ;VERIFICAR SI ESTA ACTIVADO EL
GOTO  ENV_WIPE
CALL  ENVI7          ;MODO FLASH O HOLD ACTIVADO
INCF  DESPLAZL,1
BTFSC STATUS,Z
INCF  DESPLAZH,1
BCF   STATUS,Z
GOTO  COL
ENV_SCROLL      ;MODO SCROLL
;REALIZAR CORRIMIENTO CON COL_AUX(PARA GUARDAR EL RESULTADO DE LA ROTACION) Y
COL_SCROLL(NUMERO DE ROTACIONES)
MOVF  COL_SCROLL,W
MOVWF CARAUX2
CLRF  COL_AUX
ROT_SCRO
BCF   STATUS,C
RRF   DATOCOL,1
RRF   COL_AUX,1
DECFSZ CARAUX2,1
GOTO  ROT_SCRO
MOVF  COL_AUX,W
MOVWF DATOCOL
CALL  ENVI7
INCF  DESPLAZL,1
BTFSC STATUS,Z
INCF  DESPLAZH,1
GOTO  COL
ENV_WIPE
MOVF  COL_WIPE,W

```

22

```

MOVWF CARAUX2
CLRF COL_AUX
ROT_WIPE
BSF STATUS,C
RRF COL_AUX,1
DECFSZ CARAUX2,1
GOTO ROT_WIPE
MOVF COL_AUX,W
ANDWF DATOCOL,W
MOVWF DATOCOL
CALL ENVIA7
INCF DESPLAZL,1
BTFSC STATUS,Z
INCF DESPLAZH,1
GOTO COL
;*****
;ENV_WIDE
; CALL ENVIA7
; INCF DESPLAZL,1
; BTFSC STATUS,Z
; INCF DESPLAZH,1
; GOTO COL
;*****
FN_DI
MOVWF COLMNASL ;PARA EL MODO FLASH,
BCF STATUS,C ;DETERMINA EL NUMERO DE COLUMNAS DE ESPACIO
RRF COLMNASHH,1 ;DIVISIÓN POR 2
RRF COLMNASLL,1 ;DIVISIÓN POR 2
BCF STATUS,Z
MOVF COLMNASLL,W
IORWF COLMNASHH,W
XORLW 0x00
BTFSC STATUS,Z
RETURN
;centrado
CALL ENV_ESPACIOS ;PRUEBABA
MOVLW TIME_REFRESH
CALL RETA
RETURN
;*****
ENV_ESPACIOS
BCF STATUS,C ;BORRA EL BIT DE CARRY
CLRF DATOCOL
ESPA ;ESPACIOS
CALL ENVIA7
DECFSZ COLMNASLL,1
GOTO ESPA
BCF STATUS,Z
MOVF COLMNASLL,W
IORWF COLMNASHH,W
BTFSC STATUS,Z ;SI YA NO HAY COLUMNAS DISPONIBLES RETORNAR
RETURN
DECF COLMNASHH,1
GOTO ESPA
;*****
LIMPIA_PANEL ;ENVIA COLUMNAS EN LANCO A TODO EL PANEL
MOVF NCTL,W
MOVWF NCOLUMNAS_L
MOVF NCTH,W
MOVWF NCOLUMNAS_H
CLRF DATOCOL
BCF STATUS,C ;BORRA EL BIT DE CARRY
LIM_PA
CALL ENVIA7
DECFSZ NCOLUMNAS_L,1
GOTO LIM_PA
BCF STATUS,Z
MOVLW .0
XORWF NCOLUMNAS_H,W
BTFSC STATUS,Z ;SALTE SI DIFERENTES
RETURN

```

23

```

DECF NCOLUMNAS_H,1
MOVLW 0xFF
MOVWF NCOLUMNAS_L
GOTO LIM_PA
;*****
ENVCOLCAR
    clrwdt
    bcf aborto
    call calcula_offset ;CALCULA POSICIÓN OFFSET EN LA TABLA DE CARACTERES DEL
FINAL, PARA EL CARACTER 'CARACTER'
movf DespOffsetL,W
movwf DESPLAZL
movf DespOffsetH,W
movwf DESPLAZH
movf DespOffsetL,W
addwf cursor,W
movwf DESPLAZL
btfsc STATUS,C
incf DESPLAZH,1
SigaEnvCol
    clrwdt
    call BUSCACOL
btfsc DATOCOL,0 ;Es fin de caracter?
bsf fin_car ;Sí, entonces se activa bandera de fin de caracter.
call ENVIA7 ;Envía columna apuntada por cursor del caracter ASCII correspondiente
btfss fin_car ;Está habilitado fin de caracter?
goto INCRCOLX ;No
bsf fin_car
incf contcolenvL,1 ;Se incrementa contador de columnas enviadas.
btfsc STATUS,Z
incf contcolenvH,1
; [Cont column enviadas] = [Total de columnas del panel para llenarlo]?
movf NCTH,W
xorwf contcolenvH,W
btfss STATUS,Z
goto SalEnvCar
    bcf status,z
movf NCTL,W
xorwf contcolenvL,W
btfss STATUS,Z
goto SalEnvCar
bsf aborto ;Se completó el total de columnas!
goto SalEnvCar
INCRCOLX
    clrwdt
    incf cursor,1
    movf DespOffsetL,W
    addwf cursor,W
    movwf DESPLAZL
btfsc STATUS,C
incf DESPLAZH,1
incf contcolenvL,1
btfsc STATUS,Z
incf contcolenvH,1
movf NCTH,W ;[Cont column enviadas] = [Total de columnas del panel para llenarlo]?
xorwf contcolenvH,W
btfss STATUS,Z
goto SigaEnvCol
movf NCTL,W
xorwf contcolenvL,W
btfss STATUS,Z
goto SigaEnvCol
bsf aborto ;Se completó el total de columnas!
SalEnvCar
    clrwdt
    return
;*****
;CALCULA POSICIÓN OFFSET EN LA TABLA:
calcula_offset
    clrwdt ;CARACTER=CARACTER
    clrf DespOffsetH

```

```

movlw ''
subwf caracter,W ;[código ascii del caracter - 32 (ascii de ESPACIO)]
movwf DespOffsetL
bcf STATUS,C
rlf DespOffsetL,1
rlf DespOffsetH,1
bcf STATUS,C
rlf DespOffsetL,1
rlf DespOffsetH,1
bcf STATUS,C
rlf DespOffsetL,1
rlf DespOffsetH,1 ;[código ascii del caracter - 32 ] * 8
movlw .32 ;{[código ascii del caracter - 32 ] * 8 } + 32
addwf DespOffsetL,1
btfsc STATUS,C
incf DespOffsetH,1
return
;*****
BUSCACOL
clrwdt
btfsc fuente ;Determina tipo de fuente: 0 >> Normal, 1 >> Negrilla.
goto BUSCANEG
BUSCANORM
clrwdt
movlw LOW CAR_NORM
addwf DESPLAZL,W
movwf caraux
movlw HIGH CAR_NORM
addwf DESPLAZH,W
movwf PCLATHtemp
movf PCLATHtemp,W
bsf PCLATH,4
call CAR_NORM ;Se encuentra en página 1 de memoria de programa.
clrf PCLATH
goto SIG555
BUSCANEG
clrwdt
movlw LOW CAR_NEG
addwf DESPLAZL,W
movwf caraux
movlw HIGH CAR_NEG
addwf DESPLAZH,W
movwf PCLATHtemp
movf PCLATHtemp,W
bsf PCLATH,4
bsf PCLATH,2
call CAR_NEG ;Se encuentra en página 2 de memoria de programa.
clrf PCLATH
SIG555 movwf DATOCOL
return
;*****
;*****
BUSCADIRINI
clrf DirIniH
clrf DirIniL
clrf cursor_ini
BCF FUENTE
BCF FUENTE_INI
BTFSC MODOROTATE
GOTO SEEKI
BUSCADIRINI_ROTA
bcf colvacias ;Sí existen columnas vacías.
bcf STATUS,C
movlw .1
subwf NCTL,W
movwf NCVL
btfss STATUS,Z
goto MIRANEGX2
movlw .0
xorwf NCTH,W
btfss STATUS,Z

```

25

```

goto SALRESTA12
  clrf NCVH
  bsf colvacias ;No hay columnas vacías!!
  goto SEEKDIRINI
MIRANEGX2
  btfs STATUS,C ;resultado de resta anterior negativo?
  goto SALRESTA12 ;No
  decf NCTH,W ;Sí
movwf NCVH
  goto SEEKDIRINI
SALRESTA12
  movf NCTH,W
  movwf NCVH
SEEKI
  clrf bandemod
; bsf modorotate
  bsf modorotate2
SEEKDIRINI
  clrwdt
  movf DirIniL,W
  movwf DIRL
  movwf DIRL_HT
  movf DirIniH,W
  movwf DIRH
  movwf DIRH_HT
  call LEE_RAM
movlw code_tnorm ;Código de texto NORMAL
xorwf caracter,W
  btfs STATUS,Z
  goto salx111
  call code_tnorm_cod
goto SIGABUSDIRINI
salx111
  clrwdt
  movlw code_tneg ;Código de texto NEGRITA
xorwf caracter,W
  btfs STATUS,Z
  goto salx211
  call code_tneg_cod
goto SIGABUSDIRINI
salx211
  clrwdt
  movlw code_tneg2 ;Código de texto NEGRITA2
xorwf caracter,W
  btfs STATUS,Z
  goto salx221
  call code_tneg2_cod
goto SIGABUSDIRINI
salx221
  clrwdt
  movlw code_hora1 ;Código para mostrar la HORA
xorwf caracter,W
  btfs STATUS,Z
  goto salx411
  call code_hora1_cod
goto SIGABUSDIRINI
salx411
  clrwdt
  movlw code_hora2 ;Código para mostrar la HORA2
xorwf caracter,W
  btfs STATUS,Z
  goto salx441
  call code_hora2_cod
goto SIGABUSDIRINI
salx441
  CALL BUSCA_FECHACODE
  BTFS FECHACODE
  goto SIGABUSDIRINI
  movlw temperatura ;Código para mostrar la HORA2
xorwf caracter,W
  btfs STATUS,Z

```

26

```

goto salx4111
call temperatura_cod
goto SIGABUSDIRINI
salx4111
movlw car_rotate
xorwf character,W
btfss STATUS,Z
goto salxx511
call car_rotate_cod
goto SIGABUSDIRINI
salxx511
call code_velo ;codigo de velocidad
btfss codevelo
goto salxx5111
CALL set_codevelo_rot
goto SIGABUSDIRINI
salxx5111
movlw code_finefecto ;Código para FIN DE CHARACTER
xorwf character,W
btfsc STATUS,Z
goto SIGABUSDIRINI
movlw code_finmensaje ;Código para FIN de mensajes!
xorwf character,W
btfss STATUS,Z
goto SALECCC ;Entonces es caracter ascii ! ;[No se chequea si es código válido, puesto...
;que ya está garantizado que al menos es código de ESPACIO !]
bsf pclath,3
bsf pclath,2
call SETUPRAM2 ;No se encontró código de caracter ascii, entonces graba mensaje por defecto.
CLRF PCLATH
goto SALECCC ;Entonces es caracter ascii ! ;[No se chequea si es código válido, puesto...
;que ya está garantizado que al menos es código de ESPACIO !]
SIGABUSDIRINI
incf DirIniL,1
btfsc STATUS,Z
incf DirIniH,1
goto SEEKDIRINI
SALECCC return
;*****
CODE_TNORM_COD ;Código de texto NORMAL
bcf fuente
bcf fuente_ini
bcf fuente_ini2
RETURN
CODE_TNEG_COD ;Código de texto NEGRITA
bsf fuente
bcf fuente_ini2
bsf fuente_ini
RETURN
CODE_TNEG2_COD ;Código de texto NEGRITA2
bsf fuente2
bcf fuente_ini
bsf fuente_ini2
RETURN
CODE_HORA1_COD ;Código de HORA1
movf DIRH_HT,W
movwf DIRVERHR_H
movf DIRL_HT,W
movwf DIRVERHR_L
incf DIRVERHR_L,1
btfsc STATUS,Z
incf DIRVERHR_H,1
BCF HORA_MILITAR
BSF TEMPE_HR_FE
RETURN
CODE_HORA2_COD ;Código para mostrar la HORA2
movf DIRH_HT,W
movwf DIRVERHR_H
movf DIRL_HT,W
movwf DIRVERHR_L
incf DIRVERHR_L,1

```

2X

```

    btfsc STATUS,Z
    incf DIRVERHR_H,1
    BSF  HORA_MILITAR
    BSF  TEMPE_HR_FE
    RETURN
BUSCA_FECHACODE
    BCF  FECHACODE
    movlw code_fecha1 ;Código para mostrar la FECHA01
    xorwf character,W
    btfss STATUS,Z
    goto salx5110
    BCF  FECHA02
    BSF  FECHA01
    GOTO SALX5112
salx5110
    movlw code_fecha2 ;Código para mostrar la FECHA02
    xorwf character,W
    btfss STATUS,Z
    goto salx5111
    BCF  FECHA01
    BSF  FECHA02
    GOTO SALX5112
salx5111
    movlw code_fecha3 ;Código para mostrar la FECHA03
    xorwf character,W
    btfss STATUS,Z
    goto salx511
    BCF  FECHA01
    BCF  FECHA02
salx5112
    movf DIRH_HT,W
    movwf DIRVERFE_H
    movf DIRL_HT,W
    movwf DIRVERFE_L
    incf DIRVERFE_L,1
    btfsc STATUS,Z
    incf DIRVERFE_H,1
    BSF  FECHACODE
    BSF  TEMPE_HR_FE
SALX511
    RETURN
TEMPERATURA_COD
    movf DIRH_HT,W
    bsf  STATUS,RP0
    movwf DIRVERTMP_H_b2
    bcf  STATUS,RP0
    movf DIRL_HT,W
    bsf  STATUS,RP0
    movwf DIRVERTMP_L_b2
    incf DIRVERTMP_L_b2,1
    btfsc STATUS,Z
    incf DIRVERTMP_H_b2,1
    bcf  STATUS,RP0
    BSF  TEMPE_HR_FE
    RETURN
CAR_ROTATE_COD
    clrf bandemod
CLRF CURSOR
    clrf cursor_ini
    bsf  modorotate2 ;activa modo rotate
; CALL BUSCADIRINI_ROTA
    RETURN
;*****;rutina para buscar codigo de
velocidad
code_velo
    BCF  codevelo
    movlw code_velmin
    movwf codascii
codevel2
    CLRWDI
    movf codascii,W

```

26.

23

```

xorwf character,W
btfsc STATUS,Z
goto si_codevel
movlw code_velmax
xorwf codascii,W
btfsc STATUS,Z
goto fin_codevelo
incf codascii,1
goto codevel2
si_codevel
bsf codevelo
; incf codascii,1
movlw code_velmin
subwf codascii,w
sublw .10
movwf codascii
; CALL INC_DIRLH ;
; movlw 0x5B ;ESPACIO CHIQUITO QUE SOBREESCRIBE EL QUE ENVIA EL SOFTWARE
; movwf DATOX
; call ESC_RAM
; decf DIRL,1
; btfsc STATUS,Z
; incf DIRH,1
fin_codevelo
return
;*****
;SETUPRAM
; clrwdt
; bcf colvacias ;Sí hay columnas vacías.
; clrf DIRH
; movlw .1
; movwf DIRL
; movlw '!'
; movwf DATOX
; call ESC_RAM
; CALL INC_DIRLH
; movlw 0xFC
; movwf DATOX
; call ESC_RAM ; RAM 0001H = !
; ;Establece número de columnas vacías [NCT - 2]: 0002H = FF
; bcf STATUS,C
; movlw .5
; subwf NCTL,W
; movwf NCVL
; btfss STATUS,Z
; goto MIRANEGXX
; movlw .0
; xorwf NCTH,W
; btfss STATUS,Z
; goto SALRESTA22
; clrf NCVH
; bsf colvacias ;No hay columnas vacías!!
; goto ESTATICO
;MIRANEGXX
; btfsc STATUS,C ;resultado de resta anterior negativo?
; goto SALRESTA22 ;No
; decf NCTH,W ;Sí
; movwf NCVH
; goto ESTATICO
;SALRESTA22
; movf NCTH,W
; movwf NCVH
;ESTATICO
; bcf modoRotate
; clrf DirIniH
; clrf DirIniL
; clrf cursor_ini
; return
;*****
;*****
ACTUALIZA_H_T

```

29

CadenaMes

clrwdt

addwf PCL,1

dt "Ene"

dt 0x7B ;espacio corto

dt "Feb"

dt 0x7B

dt "Mar"

dt 0x7B

dt "Abr"

dt 0x7B

dt "May"

dt 0x7B

dt "Jun"

dt 0x7B

dt "Jul"

dt 0x7B

dt "Ago"

dt 0x7B

dt "Sep"

dt 0x7B

dt "Oct"

dt 0x7B

dt "Nov"

dt 0x7B

dt "Dic"

dt 0x7B

IntSerRout

MOVWF WTEMP

SWAPF STATUS,W

CLRF STATUS

MOVWF STATUS_TEMP

MOVF PCLATH,W

MOVWF PCLATHTEMP

; bcf INTCON,PEIE

; bcf intcon,gie

CLRF INTCON ;Deshabilita todas las interrupciones

btfss PIR1,RCIF

goto NO_INTSER ;NO HAY INTERRUPCIÓN VÁLIDA

clrwdt

bsf pclath,3

bsf pclath,2

call RutinaINT

clrwdt

NO_INTSER

; bcf STATUS,RP0

; BCF ONE_SECOND

; BCF PIR1,TMR1IF

bsf STATUS,RP0

bsf PIE1,RCIE ;Habilita interrupción para recepción de USART

BSF TXSTA,TXEN ;HABILITA TRANSMISIÓN

bsf STATUS,RP0

bsf RCSTA,SPEN

bsf RCSTA,CREN ;Habilita recepción continua.

bsf RCSTA,RX9 ;Dato de recepción de 8 bits.

clrf intcon ;dashabilita interrupciones y algunos flags

clrf PIR1 ;borra los demás flags de interrupcion

;SWAPF NO AFECTA EL REGISTRO STATUS

MOVF PCLATHTEMP, W ;Restore PCLATH

MOVWF PCLATH ;Move W into PCLATH

SWAPF STATUS_TEMP,W ;Swap STATUS_TEMP register into W

MOVWF STATUS ;Move W into STATUS register

SWAPF WTEMP,F ;Swap W_TEMP

SWAPF WTEMP,W ;Swap W_TEMP into W

retfie

INICIO

bsf pclath,3

bsf pclath,2

CALL INICIALIZACION

CLRF PCLATH

btfsc continuarr

30

```

; BCF  HORA_MILITAR
BCF  STATUS,Z
MOVF  HORA,W
    ANDLW  0x0F
    XORLW  0x02
    BTFSS  STATUS,Z
    GOTO  SIGXXXX1
    BTFSS  HORA,4
GOTO  SIGXXXX1
    BTFSC  HORA_AMPM
    GOTO  HORA_NORMAL
    GOTO  SIGXXXX11
SIGXXXX1
    BTFSS  HORA_AMPM
    GOTO  HORA_NORMAL
SIGXXXX11
    BCF  HORA,7
    INCF  HORA,1
    BCF  STATUS,Z
    MOVF  HORA,W
    ANDLW  0x0F
    XORLW  0x0A
    BTFSC  STATUS,Z
    CALL  PRINT2
    INCF  HORA,1
    BCF  STATUS,Z
    MOVF  HORA,W
    ANDLW  0x0F
    XORLW  0x0A
    BTFSS  STATUS,Z
    GOTO  SIGXXX2
    CALL  PRINT2
    GOTO  SIGXXX3
PRINT2
    movlw  '2'
    movwf  DATOX
    call  ESC_RAM
    MOVLW  0xF0
    ANDWF  HORA,1
BSF  HORA,7
    RETURN
SIGXXX2
    btfsc  hora,7
    GOTO  SIGXXX3
    btfss  hora,4
    GOTO  PRINT1
PRINT3
    BCF  STATUS,Z
    MOVF  HORA,W
    ANDLW  0x0F
    XORLW  0x04
    BTFSC  STATUS,Z
    goto  print00
    movlw  '2'
    movwf  DATOX
    call  ESC_RAM
    GOTO  SIGXXX3
PRINT1
    movlw  '1'
    movwf  DATOX
    call  ESC_RAM
    goto  SIGXXX3
print00
    movlw  '0'
    movwf  DATOX
    call  ESC_RAM
    call  inc_dirlh
    movlw  '0'
    movwf  DATOX
    call  ESC_RAM
    goto  sigue_mins

```

```

SIGXXX3
    CALL    INC_DIRLH
    movf    hora,W
    andlw   0x0F
    addlw   0x30
    movwf   DATOX
    call    ESC_RAM
SIGUE_MINS
    CALL    INC_DIRLH
    movlw   ':'
    movwf   DATOX
    call    ESC_RAM
    CALL    INC_DIRLH
    swapf   minutos,W
    andlw   0x0F
    addlw   0x30
    movwf   DATOX
    call    ESC_RAM
    CALL    INC_DIRLH
    movf    minutos,W
    andlw   0x0F
    addlw   0x30
    movwf   DATOX
    call    ESC_RAM
    clrwdt
SALACTHR
    clrwdt
    return
;*****
;*****
ACTUALIZFE
    clrwdt
    movf    DIRVERFE_H,W
    xorlw   0x00
    btfss   STATUS,Z
    goto    GRAFE
    movf    DIRVERFE_L,W
    xorlw   0x00
    btfsc   STATUS,Z
    RETURN    ;No existe dir RAM para insertar la fecha
GRAFE
    movf    DIRVERFE_H,W
    movwf   DIRH
    movf    DIRVERFE_L,W
    movwf   DIRL
BTFSC    FECHA01
    GOTO    FECHA_01
    BTFSC    FECHA02
    GOTO    FECHA_02
    GOTO    FECHA_03    ;POR DEFECTO
FECHA_01
    CALL    ACT_DIA
    CALL    ACT_DE
    CALL    ACT_MES_STR
    CALL    ACT_DE
    CALL    ACT_AÑO
;    goto    fin_fecha
    RETURN
FECHA_02
    CALL    ACT_MES_STR
    CALL    ACT_ESPA
    CALL    ACT_DIA
    CALL    ACT_DE
    CALL    ACT_AÑO
;    goto    fin_fecha
    RETURN
FECHA_03
    CALL    ACT_ESPA
    CALL    ACT_DIA
    CALL    ACT_SLASH
    CALL    ACT_MES_NUM

```

32

```

    CALL ACT_SLASH
    CALL ACT_AÑO2
;goto fin_fecha
    RETURN
fin_fecha
return
;*****
ACT_MES_NUM
    SWAPF MES,W
    ANDLW 0x01
    ADDLW 0x30
    MOVWF DATOX
    CALL ESC_RAM
    CALL INC_DIRLH
    MOVF MES,W
    ANDLW 0x0F
    ADDLW 0x30
    MOVWF DATOX
    CALL ESC_RAM
    CALL INC_DIRLH
    RETURN
;*****
ACT_MES_STR
    movlw 0x01
    xorwf mes,W
    btfsc STATUS,Z
goto ENERO
    movlw 0x02
    xorwf mes,W
    btfsc STATUS,Z
goto FEBRERO
    movlw 0x03
    xorwf mes,W
    btfsc STATUS,Z
goto MARZO
    movlw 0x04
    xorwf mes,W
    btfsc STATUS,Z
goto ABRIL
    movlw 0x05
    xorwf mes,W
    btfsc STATUS,Z
goto MAYO
    movlw 0x06
    xorwf mes,W
    btfsc STATUS,Z
goto JUNIO
    movlw 0x07
    xorwf mes,W
    btfsc STATUS,Z
goto JULIO
    movlw 0x08
    xorwf mes,W
    btfsc STATUS,Z
goto AGOSTO
    movlw 0x09
    xorwf mes,W
    btfsc STATUS,Z
goto SEPTIEMBRE
    movlw 0x10
    xorwf mes,W
    btfsc STATUS,Z
goto OCTUBRE
    movlw 0x11
    xorwf mes,W
    btfsc STATUS,Z
goto NOVIEMBRE
    movlw 0x12
    xorwf mes,W
    btfsc STATUS,Z
goto DICIEMBRE

```

```
ENERO movlw .0
goto ESCRIBEMES
FEBRERO movlw .4
goto ESCRIBEMES
MARZO movlw .8
goto ESCRIBEMES
ABRIL movlw .12
goto ESCRIBEMES
MAYO movlw .16
goto ESCRIBEMES
JUNIO movlw .20
goto ESCRIBEMES
JULIO movlw .24
goto ESCRIBEMES
AGOSTO movlw .28
goto ESCRIBEMES
SEPTIEMBRE
movlw .32
goto ESCRIBEMES
OCTUBRE movlw .36
goto ESCRIBEMES
NOVIEMBRE
movlw .40
goto ESCRIBEMES
DICIEMBRE
movlw .44
ESCRIBEMES
movwf posic_car
movlw .3
movwf CUATRO
CICGRABMES
movf posic_car,W
call CadenaMes
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
incf posic_car,1
decfsz CUATRO,1
goto CICGRABMES
RETURN
```

```
ACT_DIA ;Determina día del mes:
swapf dia_mes,W
andlw 0x03 ;Captura decenas de DIA_MES
addlw 0x30
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
movf dia_mes,W
andlw 0x0F ;Captura unidades de DIA_MES
addlw 0x30
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
RETURN
```

```
ACT_de
CALL ACT_ESPA
movlw 'd'
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
movlw 'e'
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
CALL ACT_ESPA
RETURN
```

```
ACT_ESPA
movlw 0x7B
```

```

movwf DATOX
call ESC_RAM
CALL INC_DIRLH
RETURN
;*****
ACT_SLASH
movlw '/'
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
RETURN
;*****
;Determina Año:
ACT_AÑO
movlw '2'
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
movlw '0'
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
ACT_AÑO2
swapf an_o,W
andlw 0x0F ;Captura decenas de AÑO
addlw 0x30
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
movf an_o,W
andlw 0x0F ;Captura unidades de AÑO
addlw 0x30
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
clrwdt
return
;*****
;*****
;*****
INC_DIRLH
incf DIRL,1
btfsc STATUS,Z
incf DIRH,1
RETURN
;*****
;*****
RETA
MOVWF RETARD
RERE
CLRWDT
; NOP
; NOP
; NOP
DECFSZ RETARD,1
GOTO RERE
RETURN
;*****
;Envía un dato de 8 bits (Direccion ALTA para RAM) al registro de desplazamiento 4094.
;ENVDIRH1 movwf DIR_RAM_H
; movlw .8
; movwf contbits
;TX4094b2 bcf CLK94
; clrwdt
; bcf DAT94 ;Coloca en cero línea de datos del 4094
; btfsc DIR_RAM_H,7
; bsf DAT94 ;Coloca en alto línea de datos del 4094
; nop
; clrwdt
; bsf CLK94 ;Pulso de reloj para los registros 4094
; rlf DIR_RAM_H,1 ;Rota byte que se está enviando (DATOCOL)
; decfsz contbits,1

```

35

```

; goto TX4094b2
; return
;*****
;Envía un dato de 8 bits (Direccion ALTA para RAM) al registro de desplazamiento 4094.
ENVDIRH
;-----
;Envía un dato de 7 bits a los registros de desplazamiento 4094.
ENVIA7
; BSF STATUS,RP0
; movlw B'11111000' ;X X X RTC X STRB CLK DATA (PANEL)
; movwf TRISA
; BCF STATUS,RP0
; movlw .7
; movwf contbits
TX4094
; bcf CLOCK
; clrwdt
bcf DATAS ;Coloca en cero línea de datos del 4094
btfsc DATOCOL,7
bsf DATAS ;Coloca en alto línea de datos del 4094
nop
; clrwdt
bsf CLOCK ;Pulso de reloj para los registros 4094
rlf DATOCOL,1 ;Rota byte que se está enviando (DATOCOL)
decfsz contbits,1
goto TX4094
BCF CLOCK
CLRF DATOCOL
return
;*****
lee_raml
movlw LOW FRASE
addwf DIRL,W
movwf caraux
movlw HIGH FRASE
addwf DIRH,W
; movwf PCLATHtemp
; movf PCLATHtemp,W
bsf PCLATH,3
bsf PCLATH,4
call FRASE ;Se encuentra en página 2 de memoria de programa.
movwf CHARACTER
clrf PCLATH
return
LEE_RAM
clrwdt
bcf STR94
call ENVDIRHL1
bsf STR94
bsf STATUS,RP0
movlw 0xFF
movwf TRISB
bcf STATUS,RP0
bsf WE ;Lectura de RAM
bcf OER ;Salida de datos de RAM habilitada
bcf CS ;Selecciona RAM
nop
nop
movf PORTB,W
movwf CHARACTER
nop
bsf CS ;RAM inhabilitada
bsf OER ;Entrada de datos a RAM deshabilitada
BCF STR94 ;PARA BAJAR EL PULSO
return
;-----
;*****
;Envía un dato de 15 bits (Direccion ALTA para RAM) al registro de desplazamiento 4094.
ENVDIRHL1
MOVf DIRH,W
movwf DIR_RAM_HL

```

```

movlw .8
movwf contbits
TX4094BH1
bcf CLK94
clrwdt
bcf DAT94 ;Coloca en cero línea de datos del 4094
btfsc DIR_RAM_HL,7
bsf DAT94 ;Coloca en alto línea de datos del 4094
nop
clrwdt
bsf CLK94 ;Pulso de reloj para los registros 4094
rlf DIR_RAM_HL,1 ;Rota byte que se está enviando (DATOCOL)
decfsz contbits,1
goto TX4094bH1
MOVF DIRL,W
movwf DIR_RAM_HL
movlw .8
movwf contbits

```

```

TX4094BL1
bcf CLK94
clrwdt
bcf DAT94 ;Coloca en cero línea de datos del 4094
btfsc DIR_RAM_HL,7
bsf DAT94 ;Coloca en alto línea de datos del 4094
nop
clrwdt
bsf CLK94 ;Pulso de reloj para los registros 4094
rlf DIR_RAM_HL,1 ;Rota byte que se está enviando (DATOCOL)
decfsz contbits,1
goto TX4094bL1
NOP
NOP
NOP
BCF CLK94
BCF DAT94
return

```

```

;-----
Lee_EEP clrwdt
bsf STATUS,RP1
bsf STATUS,RP0 ;Banco 3 de RAM
bcf EECON1,EEPGD ;PROGRAM/DATA SELECT BIT; ACCESO A LA MEMORIA EEPROM DE
DATOS
bsf EECON1,RD ;INICIA LECTURA DE LA MEMORIA EEPROM DE DATOS
bcf STATUS,RP0 ;Banco 2 de RAM
movf EEDATA,w
bcf STATUS,RP1 ;Banco 0 de RAM
movwf INDF
bsf STATUS,RP1
bcf STATUS,RP0 ;Banco 2 de RAM
incf EEADR,1
bcf STATUS,RP1 ;Banco 0 de RAM
incf FSR,1
decfsz cuantos,1
goto Lee_EEP
return

```

```

*****
*****
ESC_RAM clrwdt ;ESCRIBE EN RAM EXTERNA 'DATOX'. PARTE ALTA 'DIRH' ES
DIRECCIONADA DE FORMA SERIAL POR UN 4094
bcf STR94 ;EN LA RUTINA ENVIDIRH Y LA PARTE BAJA 'DIRL' ES DIRECCIONADA POR
EL PUERTO D
; movf DIRH,W
call ENVDIRHL1
bsf STR94
nop
nop
nop
nop
nop
; movf DIRL,W
; movwf PORTD

```


38

[illegible]

37.


```

nop
nop
nop
nop
nop
nop
nop
nop
nop
nop
ORG    0800H
nop
nop
nop
nop
BUS_CAR_ESP
BUS_CAR_ESP_q
movlw  code_tnorm    ;Código de texto NORMAL
xorwf  character,W
btfss  STATUS,Z
goto   BUSX111
CLRF   PCLATH
call   code_tnorm_cod
BSF    PCLATH,3
bcf    fuente_fe
bcf    fuente_fe2
goto   BUSXx5111
BUSX111
clrwdt
movlw  code_tneg      ;Código de texto NEGRITA
xorwf  character,W
btfss  STATUS,Z
goto   BUSX211
CLRF   PCLATH
call   code_tneg_cod
BSF    PCLATH,3
goto   BUSXx5111
BUSX211
clrwdt
movlw  code_tneg2     ;Código de texto NEGRITA2
xorwf  character,W
btfss  STATUS,Z
goto   BUSX2211
CLRF   PCLATH
call   code_tneg2_cod
BSF    PCLATH,3
goto   BUSXx5111
BUSX2211
clrwdt
movlw  code_hora1     ;Código para mostrar la HORA
xorwf  character,W
btfss  STATUS,Z
goto   BUSX411
CLRF   PCLATH
call   code_hora1_cod
BSF    PCLATH,3
goto   BUSXx5111
BUSX411
clrwdt
movlw  code_hora2     ;Código para mostrar la HORA2
xorwf  character,W
btfss  STATUS,Z
goto   BUSX4411
CLRF   PCLATH
call   code_hora2_cod
BSF    PCLATH,3
goto   BUSXx5111
BUSX4411
CLRF   PCLATH
CALL   BUSCA_FECHACODE
BSF    PCLATH,3
BTFSC  FECHACODE

```

```

goto BUSXx5111
movlw temperatura ;Código para mostrar la TMP
xorwf caracter,W
btfss STATUS,Z
goto BUSX4111
CLRF PCLATH
call temperatura_cod
BSF PCLATH,3
goto BUSXx5111
BUSX4111
movlw car_rotate
xorwf caracter,W
btfss STATUS,Z
goto BUSXx511
CLRF PCLATH
call car_rotate_cod
BSF PCLATH,3
goto BUSXx5111
BUSXx511
clrf pclath
call code_velo ;codigo de velocidad
BSF PCLATH,3
btfss codevelo
goto BUSXx5111
clrf pclath
CALL set_codevelo_rot
BSF PCLATH,3
goto BUSXx5111
BUSXx5111
RETURN
ORG 0900H
ACTUALIZTMP
clrwdt
call ConfDS
bsf STATUS,RP0
clrf DatoRx_b2
bcf STATUS,RP0
call LeeT ;lee temperatura del ds1620 y la guarda en DatoRx_b2
bsf STATUS,RP0
movlw correccion_tmp
movwf Conta4_b2
correc_tmp
movf Conta4_b2,w
xorlw .0
btfsc status,z
goto contitmp1
decf Datorx_b2
decf Conta4_b2
goto correc_tmp
contitmp1
bcf status,rp0
call BinBCD
bsf STATUS,RP0
movf Decena_b2,w
addlw 0x30 ;convierte en acii
movwf Decena_b2

movf unidad_b2,w
addlw 0x30 ;convierte en acii
movwf unidad_b2
movf DIRVERTMP_H_b2,W
xorlw 0x00
btfss STATUS,Z
goto GRATMP
movf DIRVERTMP_L_b2,W
xorlw 0x00
btfsc STATUS,Z
GOTO FIN_TMP ;No existe dir RAM para insertar la fecha
GRATMP
clrwdt
bsf STATUS,RP0

```

43

```

movf DIRVERTMP_H_b2,W
bcf STATUS,RP0
movwf DIRH
bsf STATUS,RP0
movf DIRVERTMP_L_b2,W
bcf STATUS,RP0
movwf DIRL
;EN DATOX VIENE GURADADO EL VALOR DE LA
bsf STATUS,RP0
movf decena_b2,w
bcf STATUS,RP0
movwf datox
movf pclath,w
movwf pclathtemp
clrf pclath
call ESC_RAM
movf pclathtemp,w
movwf pclath
CALL INC_DIRLHX
bsf STATUS,RP0
movf unidad_b2,w
bcf STATUS,RP0
movwf DATOX
movf pclath,w
movwf pclathtemp
clrf pclath
call ESC_RAM
movf pclathtemp,w
movwf pclath
CALL INC_DIRLHX
movlw .125
movwf DATOX
movf pclath,w
movwf pclathtemp
clrf pclath
call ESC_RAM
movf pclathtemp,w
movwf pclath
CALL INC_DIRLHX
movlw 'C'
movwf DATOX
movf pclath,w
movwf pclathtemp
clrf pclath
call ESC_RAM
movf pclathtemp,w
movwf pclath
FIN_TMP
bcf STATUS,RP0
RETURN
;*****
;*****
;*****
; ConfDS configura el DS1620 para comunicación con CPU, y conversiones contínuas
; El byte de configuración es b'00001011', donde sólo los dos bits menos significativos
; son de interés. El bit 0 indica una sola conversión, y el 1 comunicación con CPU.
ConfDS
clrwdt
bsf RSTT
bsf STATUS,RP0
movlw .8
movwf Conta4_b2
movlw 0Ch ; Comando de escribir configuración
movwf DatoTx_b2
bcf STATUS,RP0
call OutBit

bsf STATUS,RP0
movlw .8
movwf Conta4_b2
movlw b'00001011' ; Registro de configuración

```

42.

```
movwf DatoTx_b2
    bcf STATUS,RP0
    call OutBit
    bcf RSTT
    retlw 0
```

; LeeT lee la temperatura desde el DS1620, con una resolución de 1°C. Envía el dato AAh
 ; (Leer temperatura), luego configura (DQ) como entrada, y recibe 9 bits enviados
 ; por el DS1620, dejando los 8 más significativos en DatoRx. Posteriormente vuelve a
 ; configurar DQ como salida.

```
*****
*****
```

```
LeeT
    CLRWDT
    bsf RSTT
    bsf STATUS,RP0
movlw .8
    movwf Conta4_b2
    movlw 0AAh ; Comando de leer temperatura
```

```
movwf DatoTx_b2
    bcf STATUS,RP0
    call OutBit
    bsf STATUS,RP0
    bsf TRIS_DQT
    bcf STATUS,RP0
    bsf STATUS,RP0
    movlw .9
    movwf Conta4_b2
    bcf STATUS,RP0
    call LeeBit
    bsf STATUS,RP0
    bcf TRIS_DQT
    bcf STATUS,RP0
    bcf RSTT
    nop
    nop
    nop
    nop
    nop
    clrwdt
    bsf RSTT
    bsf STATUS,RP0
    movlw .8
    movwf Conta4_b2
    movlw 0EEh ; Comando de inicar conversión
    movwf DatoTx_b2
    bcf STATUS,RP0
    call OutBit
    bcf RSTT
    nop
    nop
    nop
    retlw 40
```

```
*****
*****
```

; OutBit saca bits hacia el DS1620. Controla CLK y DQ. RST debe estar en bajo.
 ; Conta4 contiene el número de bits a sacar. DatoTx contiene el byte respectivo.
 ; Sale primero el LSB (Bit menos significativo).

```
OutBit
    bsf STATUS,RP0
    rrf DatoTx_b2,f
    bcf STATUS,RP0
    bcf DQT
    bcf CLKT
    btfsc STATUS,C
    bsf DQT
    nop
    nop
    nop
    nop
    nop
    clrwdt
```

45

```

bsf    CLKT
nop
nop
nop
    bsf    STATUS,RP0
decfsz Conta4_b2,f
goto   OutBit
bcf    STATUS,RP0
retlw  0
;*****
; LeeBit lee bits desde el DS1620. Controla CLK y DQ. RST debe estar alto.
; Conta4 contiene el número de bits a leer. El dato se guarda en DatoRx.
; Llega primero el LSB (Bit menos significativo).
; 4 (DQT) debe estar definido como entrada.
LeeBit
    bcf    STATUS,RP0
bcf    STATUS,C
bcf    CLKT
nop
nop
nop
    CLRWDT
btfsc  DQT
bsf    STATUS,C
    bsf    STATUS,RP0
rrf    DatoRx_b2,f
    bcf    STATUS,RP0
bsf    CLKT
nop
nop
nop
    bsf    STATUS,RP0
decfsz Conta4_b2,f
goto   LeeBit
    bcf    STATUS,RP0
retlw  0
; BinBCD fue tomado del "Embedded Control Handbook 1994/1995" - Microchip
; Convierte un byte (menor a 64H = 100 decimal) en dos dígitos BCD (Decena y Unidad)
; Se agregó la verificación del número, y la conversión del negativo.
; Se pueden mostrar números entre -9 y 99. Por encima de 99 sale en blanco.
; Por debajo de -9 se pierde el signo.
BinBCD
    CLRWDT
; bcf    INTCON,GIE
    bsf    STATUS,RP0
clrf   Decena_b2
movfw  DatoRx_b2
movwf  Unidad_b2
movlw  7F
subwf  Unidad_b2,w
btfsc  STATUS,C
call   Negativ
movlw  64
subwf  Unidad_b2,w
btfsc  STATUS,C
goto   Supera
gtenth
movlw  .10
subwf  Unidad_b2,w
btfss  STATUS,C
goto   over
movwf  Unidad_b2
incf   Decena_b2,f
goto   gtenth
over
    bsf    INTCON,GIE
    bcf    STATUS,RP0
retlw  0
Supera
movlw  0B
movwf  Unidad_b2

```

46

```

movwf Decena_b2
bsf  INTCON,GIE
    bcf  STATUS,RP0
retlw 0

```

Negativ

```

movlw 0FF
xorwf Unidad_b2,f
movlw 1
addwf Unidad_b2,f
movlw 0A

```

```

movwf Decena_b2
bsf  INTCON,GIE
retlw 0

```

```

*****
*****

```

INC_DIRLHX

```

clrwdt
incf DIRL,1
btfsc STATUS,Z
incf DIRH,1
RETURN

```

```

*****

```

ORG 0C00H

INICIALIZACION

```

bsf CS
bsf WE ;Lectura de RAM
bsf OER
bsf STATUS,RP0
movlw b'11111000'
movwf TRISC
bcf STATUS,RP0
bsf CS
bsf WE ;Lectura de RAM
bsf OER
clrf INTCON
bcf STATUS,RP1
bsf STATUS,RP0
movlw .7
movwf ADCON1
movlw B'11111000' ;X X X RTC X STRB CLK DATA (PANEL)

```

movwf TRISA

```

movlw b'11111111'
movwf TRISB ;Funciona como entrada (Leer RAM) y como salida (Escribir RAM).

```

movlw b'00000000'

```

movwf TRISD ;STR94 DAT94 CLK94 OE94 CS OER WE DQT
movlw b'11011000'; RX TX HAB485 SDA SCL OEPWM CLKT RSTT
movwf TRISC
movlw b'00000000'
movwf TRISE

```

```

; bsf PIE1,TMR1IE ;Habilita interrupción debida a Timer1 en modo temporizador.
BCF PIE1,TMR1IE ;DESHabilita interrupción debida a Timer1 en modo temporizador.

```

bcf STATUS,RP0

clrwdt

bcf HabTx ;Puerto RS 485 en modo RECEPCIÓN.

```

CLRF CURSOR
CLRF CURSOR_INI
CLRF NCVL
CLRF NCVH
CLRF NCSEP
CLRF NCSEPAUX
CLRF PCLATHTEMP
CLRF DESPOFFSETL
CLRF DESPOFFSETH
CLRF DIRECCIONR
CLRF DIRVERHR_H
CLRF DIRVERHR_L
CLRF POSIC_CAR

```

```

CLRF DESPLAZL
CLRF DESPLAZH
bcf aborto
clrf bandera0

```

45.

[illegible]

43

```

bcf EECON1,EEPGD ;Memoria de datos EEPROM.
bsf EECON1,RD
bcf STATUS,RP0 ;Banco 2 de RAM
movf EEDATA,w
bcf STATUS,RP1 ;Banco 0 de RAM
xorlw 0xFF
btfss STATUS,Z
goto LeeDatIni ;Si había valores en memoria EEPROM.
;No existen datos de configuración del Informador, entonces almacenar valores por defecto:
clrwdt
clrwdt
movlw 0x20
movwf FSR
movlw 0x04 ;Código de identificación del Informador en la red (por defecto: 04h).
movwf INDF
incf FSR,1
movlw N_COLS_NFO_H ;El número de columnas (verticales)
movwf INDF ;del Panel para el Informador es
incf FSR,1 ;de 400 [0190h]
movlw N_COLS_NFO_L ;
movwf INDF
incf FSR,1
MOVLW N_VERT_SEPA ;Número de verticales de separación entre el final del mensaje y su
repetición.
movwf INDF
incf FSR,1
movlw .4 ;Tiempo de desplazamiento de las verticales en modo Rotate [en decenas de
milisegundos].
movwf INDF
incf FSR,1
movlw .5 ;Tiempo de sostenimiento del mensaje en modo Hold [en segundos].
movwf INDF
movlw 0x20
movwf FSR
bsf STATUS,RP1
bcf STATUS,RP0 ;Banco 2 de RAM
clrf EEADR ;Posición inicial en memoria EEPROM para almacenar los datos.
bcf STATUS,RP1 ;Banco 0 de RAM
movlw .6 ;Número de datos a grabar.
movwf cuantos
call Esc_EEP
; Lectura de parámetros de configuración del sistema en la memoria EEPROM:
LeeDatIni
clrwdt
movlw 0x20
movwf FSR
bsf STATUS,RP1
bcf STATUS,RP0 ;Banco 2 de RAM
clrf EEADR ;Posición inicial de memoria EEPROM para leer los datos.
bcf STATUS,RP1 ;Banco 0 de RAM
movlw .6 ;Número de datos a leer.
movwf cuantos
call Lee_EEP2
movlw 0x20
movwf FSR
movf INDF,W
movwf CODE_INFO
incf FSR,1
movf INDF,W
movwf NCTH ;Parte alta de Número de columnas totales del Informador.
incf FSR,1
movf INDF,W
movwf NCTL ;Parte baja de Número de columnas totales del Informador.
incf FSR,1
movf INDF,W
movwf NCSEP ;Número de verticales de separación entre el final del mensaje y su repetición.
; incf FSR,1
; movf INDF,W
; movwf TPO_DESP ;Tiempo de desplazamiento de las verticales en modo Rotate [en decenas de
milisegundos].
; incf FSR,1

```

47

[illegible]

48.

50

```

movf pclath,w
movwf pclathtemp
clrf pclath
call LEE_RAM
movf pclathtemp,w
movwf pclath
movlw 0xFC
xorwf caracter,W
btfsc STATUS,Z
goto VERIFYDATA
;Chequea si Direccion actual (DIR) es Dirección final total de RAM (7FFFh):
movf DIRH,W
xorlw 0x7F
btfss STATUS,Z
goto CICLO_VERIF
movf DIRL,W
xorlw 0xFF
btfss STATUS,Z
goto CICLO_VERIF
call SETUPRAM2 ;No se encontró código de fin de mensaje, entonces graba mensaje por defecto.
VERIFYDATA
;Definido el intervalo de direcciones RAM donde se encuentran datos, verificar si estos datos
corresponden
;a códigos de caracteres ascii válidos o si son códigos de atributos. De no corresponder un dato con el
;código de un atributo o de un caracter ascii, se reemplaza el dato en dicha posición por el código ascii
;de un espacio en blanco.
clrf DIRH
clrf DIRL
CHEQDATA
clrwdt
movf pclath,w
movwf pclathtemp
clrf pclath
call LEE_RAM
movf pclathtemp,w
movwf pclath
CLRWDT
movlw 0xFE ;Código de texto NORMAL
xorwf caracter,W
btfss STATUS,Z
goto salatr11
goto SIGCHEQDATA
salatr11 clrwdt
movlw 0xFC ;Código de fin de mensaje
xorwf caracter,W
btfsc STATUS,Z
goto ACABOCHEQDATA
goto CHEQASCII2
SIGCHEQDATA
CALL INC_DIRLH2
goto CHEQDATA
CHEQASCII2
;Chequear si esta en el rango de caracteres válidos
movlw .32
movwf codascii
FINDASCII
CLRWDT
movf codascii,W
xorwf caracter,W
btfsc STATUS,Z
goto SIGCHEQDATA
movlw .156
xorwf codascii,W
btfsc STATUS,Z
goto INSERTA_SPACE
incf codascii,1
goto FINDASCII
INSERTA_SPACE
movlw 0x5B ;ESPACIO CORTO
movwf DATOX
movf pclath,w

```

49.

52

```

movf DIRH,W
bsf STATUS,RP0
movwf DIRH_TMP_b2
bcf STATUS,RP0
movf DIRL,W

bsf STATUS,RP0
movwf DIRL_TMP_b2
bcf STATUS,RP0
bsf pclath,3
bsf pclath,2
call LEEDS1307
call ACTUALIZHR
call ACTUALIZFE
bsf pclath,0
bsf pclath,3
CALL ACTUALIZTMP
CLRF PCLATH
bsf STATUS,RP0
movf DIRH_TMP_b2,W
bcf STATUS,RP0
movwf DIRH
bsf STATUS,RP0
movf DIRL_TMP_b2,W
bcf STATUS,RP0
movwf DIRL
BCF TEMPE_HR_FE ;DESACTIVE BANDERA DE ACTUALIZACION
clrwdt
RETURN
nop
nop
nop

```

```

ACTUALIZHR
clrwdt
movf DIRVERHR_H,W
xorlw 0x00
btfss STATUS,Z
goto GRAHR
movf DIRVERHR_L,W
xorlw 0x00
btfss STATUS,Z
goto GRAHR
goto SALACTHR ;No existe dir RAM para insertar la hora
GRAHR
movf DIRVERHR_H,W
movwf DIRH
movf DIRVERHR_L,W
movwf DIRL
BCF HORA_AMPM
BTFSC HORA,5
BSF HORA_AMPM ;HORA PM
BTFSC HORA_MILITAR
GOTO HORA_MILI
HORA_NORMAL
btfsc hora,4
goto DEC
movlw 0x5B
goto SIGXXX
DEC movlw '1'
SIGXXX
movwf DATOX
call ESC_RAM
CALL INC_DIRLH
movf hora,W
andlw 0x0F
addlw 0x30
movwf DATOX
call ESC_RAM
GOTO SIGUE_MINS
HORA_MILI

```

53

```

movlw 0x00
xorwf DATOPC,W
btfss STATUS,Z
GOTO return_INT
clrwdt
btfss PIR1,RCIF
goto $-1
bcf PIR1,RCIF
movf RCREG,W
movwf DATOPC
bcf STATUS,Z
movlw INFO_IDEN ; codigo del informador xorwf DATOPC,W
xorwf datopc,w
btfsc STATUS,Z
goto COMUNPC
bcf status,z
movlw 0x00
xorwf DATOPC,W
btfss STATUS,Z
GOTO return_INT
clrwdt
btfss PIR1,RCIF
goto $-1
bcf PIR1,RCIF
movf RCREG,W
movwf DATOPC
bcf STATUS,Z
movlw INFO_IDEN ; codigo del informador xorwf DATOPC,W
xorwf datopc,w
btfsc STATUS,Z
goto COMUNPC
bcf status,z
return_INT
CLRF PCLATH
RETURN
; BCF OE
;espera_rx
; nop
; btfss PIR1,RCIF
; goto $-1
; bcf PIR1,RCIF
; movf RCREG,W
; clrwdt
; goto espera_rx
; GOTO $
;Aquí se establece la comuincación con el PC si el código del Informador es correcto.
;Espera código de tarea arealizar:
COMUNPC
BCF OE
;Espera código de tarea:
bcf PIR1,RCIF
clrwdt
btfss PIR1,RCIF
goto $-1
bcf PIR1,RCIF
movf RCREG,W
movwf DATOPC
movlw 0x01 ;Código de tarea para grabar datos en RAM
xorwf DATOPC,W
btfsc STATUS,Z
goto GRABARAM
movlw 0x02 ;Código de tarea para configurar el reloj de tiempo real.
xorwf DATOPC,W
btfsc STATUS,Z
goto GRABARELOJ
movlw 0x03 ;Código de tarea para configurar el Informador.
xorwf DATOPC,W
btfsc STATUS,Z
goto CONFIG_INFO
NOP
BCF OE

```

```

espera_commpc
btfss PIR1,RCIF
    goto $-1
    bcf PIR1,RCIF
    movf RCREG,W
    clrwdt
    goto espera_commpc
GOTO $
CONFIG_INFO
    clrwdt
    movlw 0x20
    movwf FSR
    movlw .6
    movwf cuantos
RECICONFIG
    ;clrwdt
    btfss PIR1,RCIF
    goto $-1
    bcf PIR1,RCIF
    movf RCREG,W
    movwf INDF
    incf FSR,1
    decfsz cuantos,1
    goto RECICONFIG
    movlw 0x20
    movwf FSR
    bsf STATUS,RP1
    bcf STATUS,RP0 ;Banco 2 de RAM
    clrf EEADR ;Posición inicial en memoria EEPROM para almacenar los datos.
    bcf STATUS,RP1 ;Banco 0 de RAM
    movlw .6 ;Número de datos a grabar.
    movwf cuantos
    call Esc_EEP
    bcf OE
espera_config
    btfss PIR1,RCIF
    goto $-1
    bcf PIR1,RCIF
    movf RCREG,W
    clrwdt
    goto espera_config
GOTO $
;-----
;-----
GRABARELOJ
    clrwdt
    movlw 0x20
    movwf FSR
    movlw .7
    movwf cuantos
RECIDATRELOJ
    btfss PIR1,RCIF
    goto $-1
    bcf PIR1,RCIF
    movf RCREG,W
    movwf INDF
    incf FSR,1
    decfsz cuantos,1
    goto RECIDATRELOJ
    movlw 0x20
    movwf FSR
    movlw .7 ;Número de datos a grabar.
    movwf cuantos
    call ESCDS1307
    BCF OE
espera_rxrelog
    btfss PIR1,RCIF
    goto $-1
    bcf PIR1,RCIF
    movf RCREG,W
    clrwdt

```

SS

```

goto espera_rxrelog
GOTO $
;-----
GRABARAM
clrf DIRL
clrf DIRH
call clear_retard
bcf sirx_retard
clrwdt
SIGGRAB
incf rx_retardl,1 ;espera que da un tiempo límite para la recepcion del
NOP
NOP
NOP
btfss status,z ;siguiente dato. si se llega al limite rx_retardh=0
goto cont_siggrab ;entonces se graba un final de caracter 0xFC y se reinicia
incf rx_retardh,1
btfss status,z
goto cont_siggrab
incf rx_retardhh,1
btfss status,z
goto cont_siggrab
movlw 0xFC
bsf sirx_retard
goto cont_siggrab2
cont_siggrab
btfss PIR1,RCIF
goto SIGGRAB
bcf PIR1,RCIF
movf RCREG,W
cont_siggrab2
clrwdt
movwf DATOPC
movwf DATOX
clrwdt
call clear_retard
movf pclath,w
movwf pclathtemp
clrf pclath
call ESC_RAM
movf pclathtemp,w
movwf pclath
CALL INC_DIRLH2
movlw 0xFC
xorwf DATOPC,W
btfss STATUS,Z ;dato recibido es fin de mensaje?
goto SIGGRAB ;No!
call respuesta ;ENVIA RESPUESTA PARA CONFIRMACIÓN DE TX
btfsc sirx_retard
goto SalInterr
espera_fin
btfss PIR1,RCIF
goto $-1
bcf PIR1,RCIF
movf RCREG,W
btfss PIR1,RCIF
goto $-1
bcf PIR1,RCIF
movf RCREG,W
clrwdt
goto espera_fin
;fin_rxx
; bsf STATUS,RP0
; bsf PIE1,RCIE
; bcf STATUS,RP0
;return
; clrf pclath
; goto INICIO
SalInterr
GOTO $
; retfie

```

54.

56

```

; clrf pclath
; BSF STROBE
; return
clear_retard
  clrf rx_retardl
  clrf rx_retardh
  movlw .150
  movwf rx_retardhh
return
;*****
SETUPRAM2
  clrwdt
  bcf colvacias ;Sí hay columnas vacías.
clrf DIRH
  movlw .1
  movwf DIRL
  movlw '!'
  movwf DATOX
  movf pclath,w
  movwf pclathtemp
  clrf pclath
  call ESC_RAM
  movf pclathtemp,w
  movwf pclath
  CALL INC_DIRLH2
  movlw 0xFC
  movwf DATOX
  movf pclath,w
  movwf pclathtemp
  clrf pclath
  call ESC_RAM
  movf pclathtemp,w
  movwf pclath

; RAM 0001H = !
;Establece número de columnas vacías [NCT - 2]: 0002H = FF
bcf STATUS,C
  movlw .5
  subwf NCTL,W
  movwf NCVL
  btfs STATUS,Z
  goto MIRANEGXX3
  movlw .0
  xorwf NCTH,W
  btfs STATUS,Z
  goto SALRESTA223
  clrf NCVH
  bsf colvacias ;No hay columnas vacías!!
  goto ESTATICO3
MIRANEGXX3
  btfs STATUS,C ;resultado de resta anterior negativo?
  goto SALRESTA223 ;No
  decf NCTH,W ;Sí
  movwf NCVH
  goto ESTATICO3
SALRESTA223
  movf NCTH,W
  movwf NCVH
ESTATICO3
  bcf modoRotate
  clrf DirIniH
  clrf DirIniL
  clrf cursor_ini
  return
;*****
INC_DIRLH2
  incf DIRL,1
  btfs STATUS,Z
  incf DIRH,1
RETURN

```

57

```

*****
;ESC_RAM clrwdt          ;ESCRIBE EN RAM EXTERNA 'DATOX'. PARTE ALTA 'DIRH' ES
DIRECCIONADA DE FORMA SERIAL POR UN 4094
; bcf STR94              ;EN LA RUTINA ENVIDIRH Y LA PARTE BAJA 'DIRL' ES DIRECCIONADA POR
EL PUERTO D
; movf DIRH,W
; call ENVIDIRH
; bsf STR94
; movf DIRL,W
; movwf PORTD
; bsf STATUS,RP0
; movlw 0x00
; movwf TRISB
; bcf STATUS,RP0
; movf DATOX,W
; movwf PORTB
; bcf WE      ;escritura en RAM
; bcf OER     ;Entrada de datos a RAM habilitada
; bcf CS      ;Selecciona RAM
; nop
; nop
; bsf CS      ;RAM inhabilitada
; bsf WE      ;Lectura de RAM
; bsf OER     ;Entrada de datos a RAM deshabilitada
; return
*****
;lee_ram2
; movlw LOW FRASE
; addwf DIRL,W
; movwf caraux
; movlw HIGH FRASE
; addwf DIRH,W
; movwf PCLATHtemp
; movf PCLATHtemp,W
; clrf pclath
; bsf PCLATH,3
; bsf PCLATH,4
; call FRASE      ;Se encuentra en página 2 de memoria de programa.
; movwf CHARACTER
; clrf pclath
; bsf PCLATH,3
; bsf PCLATH,2
; return
;LEE_RAM2I
; clrwdt
; bcf STR94
; movf DIRH,W
; call ENVIDIRH
; bsf STR94
; movf DIRL,W
; movwf PORTD
; bsf STATUS,RP0
; movlw 0xFC
; movwf TRISB
; bcf STATUS,RP0
; bsf WE      ;Lectura de RAM
; bcf OER     ;Salida de datos de RAM habilitada
; bcf CS      ;Selecciona RAM
; movf PORTB,W
; movwf caracter
; nop
; bsf CS      ;RAM inhabilitada
; bsf OER     ;Entrada de datos a RAM deshabilitada
; return
;
*****
;ENVIDIRH
;movwf DIR_RAM_H
; movlw .8
; movwf contbits

```

AB

```
;TX4094b bcf CLK94
; clrwdt
; bcf DAT94 ;Coloca en cero línea de datos del 4094
; btfsc DIR_RAM_H,7
; bsf DAT94 ;Coloca en alto línea de datos del 4094
; nop
; clrwdt
; bsf CLK94 ;Pulso de reloj para los registros 4094
; rlf DIR_RAM_H,1 ;Rota byte que se está enviando (DATOCOL)
; decfsz contbits,1
; goto TX4094b
; return
```

```
*****
***** RUTINAS DEL BUS I2C *****
*****
```

```
-----
LEEDS1307 ;Lee la hora actual y la fecha del reloj.
```

```
clrwdt
movlw .6
movwf DIRECCIONR
call LEER_R
movf RECEP,W
movwf an_o
movlw .5
movwf DIRECCIONR
call LEER_R
movf RECEP,W
movwf mes
movlw .4
movwf DIRECCIONR
call LEER_R
movf RECEP,W
movwf dia_mes
movlw .3
movwf DIRECCIONR
call LEER_R
movf RECEP,W
movwf dia_sem
movlw .2
movwf DIRECCIONR
call LEER_R
movf RECEP,W
movwf hora
movlw .1
movwf DIRECCIONR
call LEER_R
movf RECEP,W
movwf minutos
clrwdt
clrf pclath
return
```

```
*****
```

```
LEER_R call BIT_INICIO ;Lee 1 byte del reloj serial DS1307.
```

```
nop
nop
movlw b'11010000' ;Código de dispositivo en el bus I2C. [bcf DISPTIVO,0]
movwf TRANSM
call ENVIA_BYTE
movf DIRECCIONR,W
movwf TRANSM
call ENVIA_BYTE
nop
nop
clrwdt
call BIT_INICIO
nop
nop
```

```
movlw b'11010001' ;Código de dispositivo en el bus I2C. [bsf DISPTIVO,0]>Lectura.
movwf TRANSM
clrwdt
call ENVIA_BYTE
```

54

```

nop
call RECIBE_BYTE
bsf   EEPROM,BIT_OUT
      clrwdt
call  SACA_BIT
call  BIT_PARO
return
;*****

```

BIT_INICIO ;Genera bit de inicio (START).

```

;clrwdt
bsf STATUS,RP0
movlw b'11000000'
movwf TRISC
bcf STATUS,RP0
bcf SCL
bsf SDA
nop
bsf SCL
nop
nop
nop
nop
nop
bcf SDA
nop
nop
nop
nop
nop
nop
bcf SCL
nop
nop
return
;*****

```

BIT_PARO ;Genera el bit de parada (STOP).

```

;clrwdt
bsf STATUS,RP0
movlw b'11000000'
movwf TRISC
bcf STATUS,RP0
bcf SDA
nop
nop
nop
bsf SCL
nop
nop
nop
nop
bsf SDA
nop
nop
bcf SCL
nop
nop
nop
return
;*****

```

ESCDS1307 ;Escribe la hora y la fecha al reloj.

```

movlw .6
movwf DIRECCIONR
NEXTESCDS1307
CLRWDT
movf INDF,W
bsf STATUS,RP0
movwf DATO_b2
bcf STATUS,RP0
call ESCRIBIR_R
decf DIRECCIONR,1
incf FSR,1
decfsz cuantos,1
goto NEXTESCDS1307

```

```

clrf DIRECCIONR
bsf STATUS,RP0
clrf DATO_b2
bcf STATUS,RP0
CLRWDT
call ESCRIBIR_R
return
;*****
;
; ESCRIBIR_R ;Escribe 1 byte en el reloj serial DS1307.
call BIT_INICIO
nop
nop
movlw b'11010000' ;Código de dispositivo en el bus I²C. [bcf DISPTIVO,0]
movwf TRANSM
call ENVIA_BYTE
movf DIRECCIONR,W
movwf TRANSM
call ENVIA_BYTE
nop
nop
;clrwdt
bsf STATUS,RP0
movf DATO_b2,W
bcf STATUS,RP0
movwf TRANSM
call ENVIA_BYTE
call BIT_PARO
return
;*****
;
; ENVIA_BYTE ;Envía un byte por el bus I²C al dispositivo esclavo.
movlw .8
movwf cuenta
TXLP ;clrwdt
bcf EEPROM,BIT_OUT
btfsc TRANSM,7
bsf EEPROM,BIT_OUT
call SACA_BIT
rlf TRANSM,1
decfsz cuenta,1
goto TXLP
call LEE_BIT ;lee (espera) reconocimiento del dispositivo.
return
;
; RECIBE_BYTE ;Recibe un byte por el bus I²C del dispositivo esclavo.
clrf RECEP
movlw .8
movwf cuenta
bcf STATUS,0
RXLP clrwdt
rlf RECEP,1
call LEE_BIT
btfsc EEPROM,BIT_IN
bsf RECEP,0
decfsz cuenta,1
goto RXLP
return
;*****
;
; LEE_BIT clrwdt
bsf EEPROM,BIT_IN ;Lee un bit en la línea de datos (SDA).
bsf STATUS,RP0
movlw b'11010000'
movwf TRISC
bcf STATUS,RP0
bsf SCL
nop
nop
nop
nop
nop
nop
nop
nop

```

61

```

nop
btfss SDA
bcf EEPROM,BIT_IN
bcf SCL
return
;*****
SACA_BIT ;Saca un bit por la línea de datos (SDA).
clrwdt
bsf STATUS,RP0
movlw b'11000000'
movwf TRISC
bcf STATUS,RP0
bsf SDA
btfss EEPROM,BIT_OUT
bcf SDA
nop
nop
bsf SCL
nop
nop
nop
nop
nop
bcf SCL
return
;*****
;-----
Lee_EEP2
clrwdt
bsf STATUS,RP1
bsf STATUS,RP0 ;Banco 3 de RAM
bcf EECON1,EEPGD ;PROGRAM/DATA SELECT BIT; ACCESO A LA MEMORIA EEPROM DE
DATOS
bsf EECON1,RD ;INICIA LECTURA DE LA MEMORIA EEPROM DE DATOS
bcf STATUS,RP0 ;Banco 2 de RAM
movf EEDATA,w
bcf STATUS,RP1 ;Banco 0 de RAM
movwf INDF
bsf STATUS,RP1
bcf STATUS,RP0 ;Banco 2 de RAM
incf EEADR,1
bcf STATUS,RP1 ;Banco 0 de RAM
incf FSR,1
decfsz cuantos,1
goto Lee_EEP2
return
;*****
;*****
respuesta
clrf temp_l ;Reg temporales guardan CRC temp ->CRC
clrf temp_h
clrf DIRH
clrf DIRL
clrwdt
call retardo_rxtx
bsf habtx
genera_crc
movf pclath,w
movwf pclathtemp
clrf pclath
call LEE_RAM
movf pclathtemp,w ;CRC =
movwf pclath
;PRUEBABABABA
; MOVF CHARACTER,W
; bcf PIR1,TXIF ;Restaura el flag del transmisor
; movwf TXREG ;Almacena el byte a transmitir
; bsf STATUS,RP0
pruebaba
; btfss TXSTA,TRMT ;Byte transmitido ??
; goto pruebaba ;No, esperar
; bcf STATUS,RP0

```

62

```

;HKJHKJHKJH
  swapf  caracter,w      ;w = caracter swaped
  andlw  0x0F
  addwf  temp_l,1        ;crc = crc + w
  btfsc  STATUS,C
  incf   temp_h,1
  CALL   INC_DIRLH2
  clrwdt
  movlw  0xFC
xorwf  caracter,W      ;verifica si es fin de caracter
btfss  STATUS,Z
  goto  genera_crc
envia_crc
  movlw  0x02            ;codigo que identifica la tarjeta microcinta
  bcf    PIR1,TXIF       ;Restaura el flag del transmisor
  movwf  TXREG           ;Almacena el byte a transmitir
bsf    STATUS,RP0
Tx_Dato_Wait0
btfss  TXSTA,TRMT       ;Byte transmitido ??
  goto  Tx_Dato_Wait0    ;No, esperar
bcf    STATUS,RP0
  movf   temp_l,w
bcf    PIR1,TXIF       ;Restaura el flag del transmisor
  movwf  TXREG           ;Almacena el byte a transmitir
bsf    STATUS,RP0
Tx_Dato_Wait1
btfss  TXSTA,TRMT       ;Byte transmitido ??
  goto  Tx_Dato_Wait1    ;No, esperar
bcf    STATUS,RP0
  movf   temp_h,w
bcf    PIR1,TXIF       ;Restaura el flag del transmisor
  movwf  TXREG           ;Almacena el byte a transmitir
bsf    STATUS,RP0
Tx_Dato_Wait2
btfss  TXSTA,TRMT       ;Byte transmitido ??
  goto  Tx_Dato_Wait2    ;No, esperar
bcf    STATUS,RP0
  clrwdt
envia_columnas
  movf   NCTL,w
  bcf    PIR1,TXIF       ;Restaura el flag del transmisor
  movwf  TXREG           ;Almacena el byte a transmitir
bsf    STATUS,RP0
Tx_Dato_Wait3
btfss  TXSTA,TRMT       ;Byte transmitido ??
  goto  Tx_Dato_Wait3    ;No, esperar
bcf    STATUS,RP0
  movf   NCTH,w
bcf    PIR1,TXIF       ;Restaura el flag del transmisor
  movwf  TXREG           ;Almacena el byte a transmitir
bsf    STATUS,RP0
Tx_Dato_Wait4
btfss  TXSTA,TRMT       ;Byte transmitido ??
  goto  Tx_Dato_Wait4    ;No, esperar
bcf    STATUS,RP0
fin_crc
  movlw  0xFC
bcf    PIR1,TXIF       ;Restaura el flag del transmisor
  movwf  TXREG           ;Almacena el byte a transmitir
bsf    STATUS,RP0
Tx_Dato_Wait5
btfss  TXSTA,TRMT       ;Byte transmitido ??
  goto  Tx_Dato_Wait5    ;No, esperar
bcf    STATUS,RP0
  bcf    habtx          ;deshabilita tx por RS485
  return
;*****
retardo_rtx
  clrf   rx_retardl
  clrf   rx_retardh
rt_rx

```

```

movlw .220 ;APROX 5 SEGUNDOS. A MENOR W MAYOR RETARDO
rt_wr
clrwdt
    addlw .1
    btfss status,z
    goto rt_wr
incf rx_retardl,1 ;espera que da un tiempo límite para la recepcion del
    btfss status,z ;siguiente dato. si se llega al limite rx_retardh=0
    goto rt_rx ;entonces se graba un final de caracter 0xFC y se reinicia
incf rx_retardh,1
    btfss status,z
    goto rt_rx
return
    ORG 1800H
FRASE
    movwf PCLATH
    movf caraux,W
    addwf PCL,1
; DT 0x5b,0x84
; DT "PUBLIK INFO"
; DT PARAVERIFICAR EL FUNCIONAMIENTO DEL
EFECTOHOLDCUANDOSEBLOQUEAPARAMENSAJESLARGOS"
;0x04,0x01,
; dt 0x5B,0xFE,0x9A,0x54,0x4D,0x50,0xFE,0x20,0xFC,0xFC,0xFC,0xFC
; dt 0x5B,0x68,0x6F,0x72,0x61,0xFE,0x98,0x48,0x4F,0x52,0x41,0x4E,0xFE
; dt
0x98,0x54,0x45,0x4D,0x50,0xFE,0x97,0x48,0x4F,0x52,0x41,0x4E,0xFE,0x20,0xFC,0xFC,0xFC
; dt 0x5B,0x68,0x6F,0x72,0x61,0xFE,0x98,0x86,0x48,0x4F,0x52,0x41,0x4E,0x85,0xFE
; dt 0x98,0x54,0x45,0x4D,0x50,0xFE,0x97,0x46,0x45,0x43,0x48,0x41,0x31,0x20,0x48
; dt 0x4F,0x52,0x41,0x4E,0xFE,0x9A,0x54,0x45,0x4D,0x50,0xFE,0x98,0x48,0x4F,0x52
; dt 0x41,0x4E,0xFE,0x20,0xFC,0xFC,0xFC
dt
0x5B,0xFE,0x9B,0x5B,0x8C,0x5B,0x5B,0x5B,0x5B,0x5B,0xfc,0x5B,0x5B,0x5B,0x20,0xFE,0x97,0x5B,0x8B
,0x5B,0x5B,0x5B
dt
0x5B,0x5B,0x5B,0x5B,0x5B,0x5B,0x5B,0x5B,0x5B,0x5B,0x5B,0x5B,0x20,0xFE,0x20,0xFC,0xFC,0xF
C
; dt 0x5B,0xFE,0x98,0x86,0x5B,0x88,0x5B,0x5B,0x5B,0x5B,0x20,0x85,0xFE,0x98
; dt 0x45,0x73,0x74,0x65,0xFE,0x98,0x61,0x7C,0x6F,0xFE,0x97,0x73,0x65,0x20,0x62
; dt 0x65,0x6E,0x65,0x66,0x69,0x63,0x69,0x61,0x72,0x60,0x6E,0x20,0x31,0x30,0x30
; dt 0x65,0x6E,0x74,0x61,0x64,0x6F,0x20,0x6C,0x61,0x20,0x63,0x6F,0x62,0x65,0x72
; dt 0x74,0x75,0x72,0x61,0x20,0x65,0x64,0x75,0x63,0x61,0x74,0x69,0x76,0x61,0xFE
; dt 0x98,0x65,0x6E,0xFE,0x98,0x32,0x2E,0x35,0x33,0x39,0x20,0xFE,0x97,0x6E,0x75
; dt 0x65,0x76,0x6F,0x73,0x20,0x61,0x6C,0x75,0x6D,0x6E,0x6F,0x73,0xFE,0x20,0xFC
; dt 0xFC,0xFC
; DT 0x88
; DT "023:23O"
; dt 0x9A ; ,0x09,0xFE,0xFC
; DT "Hora: "
; DT 0x89,0x45,0xFE,0xFC
DT "HO"
; DT 0xFE,0x17
; DT "TU"
; DT 0xFE,0xFC
; DT "a b c d e f g h i j k l m n"
; DT "o p q r s t u v w x y z a b c"
; DT "d e f g h i j k l m n o p q"
; DT "r s t u v w x y z"
END
```

62