

COMENTARIO TECNICO

Como manejar un Display Inteligente LCD por medio de un Microcontrolador de la Familia HC908..

Por Ing. Daniel Di Lella
D.F.A.E For Motorola Products
Depto. Técnico Electrocomponentes S.A.
Fae@electrocom.com.ar

Hoy en día, es una necesidad cada vez más frecuente, la visualización de comandos y estados de una gran variedad de equipos con el usuario final. Es evidente que, al ir mejorando las prestaciones generales de un sistema o equipo, se hace necesario disponer de un medio práctico de comunicar las diferentes informaciones "Usuario - Máquina".

Uno de los elementos más usados en este sentido, es el display LCD de tipo "inteligente". Estos módulos, son la forma más eficiente de comunicación alfanumérica con el mundo exterior. Los hay de distintos tamaños, cantidad de caracteres y cantidad de filas. Por ejemplo, la empresa **WINTEK Corporation** (Electrocomponentes, es distribuidor exclusivo en la Argentina), posee módulos desde 8 caracteres x 1 línea (osea una línea de 8 caracteres) , hasta módulos de 40 caracteres x 4 líneas. Un opcional que mejora la visibilidad en cualquier condición es el "Back - Light" o luz de fondo que puede ser del tipo a LED (array de LEDs uniformemente distribuidos) o del tipo Electroluminiscente.

El propósito del siguiente artículo, no es entrar en detalles constructivos o de presentación, sino aportar datos útiles para poder utilizar uno de estos display con un microcontrolador en un proyecto determinado.

Introducción:

Los módulos de display inteligentes, son denominados así, ya que poseen la "inteligencia" necesaria como para efectuar en forma totalmente automática y transparente para el usuario, las complejas operaciones de control, presentación y mantenimiento de los caracteres en pantalla. Para ello, todos cuentan con controladores dedicados, integrados a la estructura de los mismos. Estos controladores integrados, aunque de distintos fabricantes, presentan repertorios o sets de instrucciones (lista de comandos) hacia el mundo exterior muy similares entre sí, por lo cuál, en la mayoría de los casos, no importa la marca del display a emplear, sino el hecho que utilicen controladores integrados compatibles, haciendo de esta forma "genérico" el uso de un display LCD inteligente.

Líneas de Conexión:

Los display inteligentes poseen líneas de conexión con el mundo exterior que pueden dividirse en :

- Líneas de Datos / Comandos.
- Líneas de Control

Las líneas de Datos / Comandos son las utilizadas por el display para recibir Datos o Comandos y eventualmente pueden utilizarse para conocer el estado de ocupación del controlador interno (display "Busy"). Según el modo empleado, pueden necesitarse 8 o 4 líneas de Datos / Comandos.

Si se utiliza el modo de 8 líneas de Datos, se tiene la ventaja de un manejo más sencillo a la hora de implementar el software en el microcontrolador elegido, pero se paga el precio de tener que disponer más puerto I/O del MCU para el envío de datos o comandos al display.

En cuanto a las líneas de Control, son necesarias para coordinar las distintas operaciones que puede realizar un display LCD , a saber:

- Escritura de Datos al display (Data Word).
- Escritura de Comandos al display (Control Word).
- Lectura de estados del display (Busy, Address Counter, DD RAM, CG RAM)

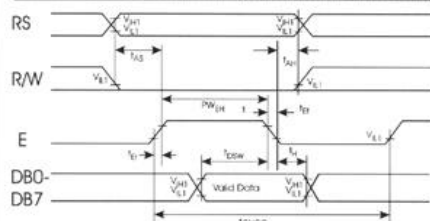
Solo son necesarias 3 líneas de control para efectuar la tarea, RS , E , y R/W.

Cuando se quiere trabajar con alguno de estos displays, se pueden utilizar varias formas de control de los mismos. Por ejemplo se puede trabajar con **8 bits** o con **4 bits de palabra de datos**. También puede utilizarse el método de demora fija entre envío de caracter y caracter, o emplear el método de lectura del estado "Busy" que proporciona el controlador integrado en el display.

Los diagramas de tiempos correspondientes a las distintas señales de control se observan en las siguientes figuras:

Timing Control

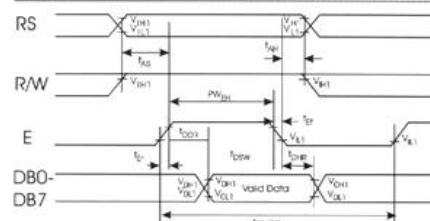
1. Write Operation



Writing data from MPU to LCM

Item	Symbol	Limit (min.)	Limit (Max.)	Unit
Enable Cycle Time	t_{cycle}	666	-	nS
Enable Pules Width (High level)	PW_{EH}	300	-	nS
Enable Rise/Fall Time	t_{er}, t_{ef}	-	25	nS
Address Set-Up Time (RS,R/W,E)	t_{AS}	100	-	nS
Address Hold Time	t_{AH}	10	-	nS
Data Set-Up Time	t_{DSW}	100	-	nS
Data Hold Time	t_{DH}	10	-	nS

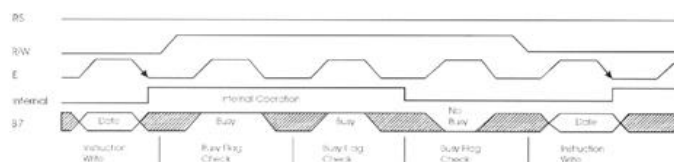
2. Read Operation



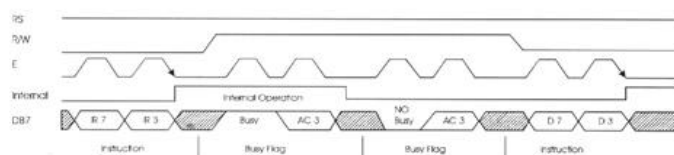
Reading data from LCM to MPU

Item	Symbol	Limit (min.)	Limit (Max.)	Unit
Enable Cycle Time	t_{cycle}	666	-	nS
Enable Pules Width (High level)	PW_{EH}	300	-	nS
Enable Rise/Fall Time	t_{er}, t_{ef}	-	25	nS
Address Set-Up Time (RS,R/W,E)	t_{AS}	100	-	nS
Address Hold Time	t_{AH}	10	-	nS
Data Delay Time	t_{DDR}	-	190	nS
Data Hold Time	t_{DHR}	20	-	nS

3. 8-bit busy flag check timing



4. 4-bit busy check timing



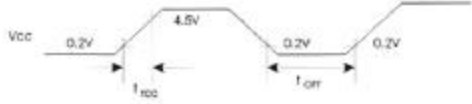
(None) R 7, R 3: Instruction 7th bit; AC3: Address Counter 3rd bit

La inicialización de estos displays , puede efectuarse en forma automática por medio del propio controlador integrado, al detectar este un Power - On Reset (P.O.R) , osea durante el encendido del sistema, o bien por medio de comandos de software enviados al controlador para efectuar esta tarea de inicialización. Muchas veces puede ocurrir, que la pendiente de subida de la fuente de alimentación, no es lo suficientemente abrupta como para garantizar que el controlador integrado detecte la condición de P.O.R. , por lo que es conveniente tener contemplado el uso de una rutina de inicialización por software en nuestra aplicación.

Initialization of LCM

The LCM automatically initializes (reset) when power is turned on using the internal reset circuit. If the power supply conditions for correctly operation of the internal reset circuit are not met, initialization by instruction is required. Use the procedure in next page for initialization.

Valid Power Supply Condition



(Note 1) $0.1 \text{ ms} \geq t_{\text{rise}} \geq 10 \text{ ms}$, $t_{\text{off}} \geq 1 \text{ ms}$

(Note 2) t_{off} stipulates the time of power OFF for momentary power supply dip or when power supply cycles ON and OFF.

Item	Sym	Test condition	limit (Min.)	limit (Max.)	Unit
Power supply rise time	t_{rise}	-	0.1	10	ms
Power supply OFF time	t_{off}	-	1	-	ms

INITIALIZATION BY INSTRUCTION

8-bits

Power On

Wait more than 15 ms
Vcc rises to 4.5V

↓ BF cannot be checked before this

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	X	X	X	X

Function set : interface is 8-bit long

Wait more than 4.1ms

↓ BF cannot be checked before this

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	X	X	X	X

Function set

Wait more than 0.1ms

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	X	X	X	X

Function set

↓ BF cannot be checked at this time

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	1	N	F	X	X

Function set:DL=1, 8bit interface data.

↓ Check for not busy

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	0	0	0

Display off

↓ Check for not busy

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

Clears all display and return cursor to home position

↓ Check for not busy

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	I/O S

Entry mode set

↓

End of initialization

4-bits

Power On

Wait more than 15 ms
Vcc rises to 4.5V

↓ BF cannot be checked before this

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	1

Function set:DL=1,8 bit interface data.

Wait more than 4.1ms

↓ BF cannot be checked before this

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	1

Function set:DL=1,8 bit interface data.

Wait more than 0.1ms

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	0

Function set:DL=0, 4bit interface data.

↓ BF cannot be checked at this time

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	0

Function set: DL=0, 4 bit interface data.

↓ Check for not busy

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	0	0

Display off

↓ Check for not busy

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	0	0

Display on

↓ Check for not busy

RS	R/W	DB7	DB6	DB5	DB4
0	0	0	0	1	I/D S

Entry mode set

↓

End of initialization

Una vez efectuado el "Reset" del display, el mismo se encuentra en condiciones de recibir palabras de comando que personalizan la forma de presentar los sucesivos caracteres

enviados a este. Por ejemplo se pueden enviar comandos de "Clear Display" para limpiar la presentación en pantalla, "Return Home" para posicionar al cursor en la posición "Home" o primer carácter superior izquierdo, etc, etc. En las siguientes figuras, pueden verse la lista completa de los mismos y algunos ejemplos de uso de estos.

Instruction Operation

Function	R S	R W	D 7	D 6	D 5	D 4	D 3	D 2	D 1	D 0	Description	Execu. Time* (Max.)
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and returns the cursor to home position (address 0).	1.64mS
Return Home	0	0	0	0	0	0	0	0	1	X	Return the cursor to the home position. Also returns the display being shifts to the original position. DD RAM contents remain unchanged. Set DD RAM address to zero.	1.64mS
Entry mode set	0	0	0	0	0	0	0	0	1	I / S D	Set cursor move directly and specifies or not to shift the display. These operations are performed during data write/read of DD RAM/CG RAM. For normal operation, sets S to zero. I/D=1; increment; I/D=0; decrement S=1; accompanies display shift when data is written, for normal operation, set to zero.	40µS
Display ON/OFF control	0	0	0	0	0	0	1	D	C	B	Set ON/OFF all display (D), cursor ON/OFF(C), and blink of cursor position character (B). D=1: ON display, D=0:OFF display. C=1: ON Cursor, C=0: OFF cursor, B=1: ON Blink Cursor, B=0, OFF Blink Cursor	40µS
Cursor or Display shift	0	0	0	0	0	1	/	/	X	X C L	Move the crusor and shift the display without changing DD RAM contents. S/C=1: Display Shift, S/C=0: Cursor move, R/L=1: shift to right, R/L=0:shift to left.	40µS
Function Set	0	0	0	0	1	D	N	F	X	X L	Set the interface data length (DL), number of display lines (N) and character font (F). DL=1: 8 bits, DL=0: 4bits N=1: 2 lines, N=0:2 lines, N=0:1 lines, F=1: 5x10 dots, F=0: 5x7 dots	40µS
Set CG RAM Addr.	0	0	0	1	ACG						Set CG RAM address. CG RAM data is sent and received after this setting.	40µS
Set DD RAM Addr.	0	0	1	ADD							Set DD RAM address. DD RAM data is sent and received after this setting.	40µS
Read busy flag & Addr	0	1	B F	AC							Reads BUSY FLAG (BF) indicating internal operation is beginning perfmed and reads address counter contents. BF=1:internally operating. BF=0: can accept instruction.	0µS
Write Data to CG RAM	1	0	WRITE DATA								Write data into DD RAM or CG RAM.	40µS**
Read Data from CG/DD RAM	1	1	READ DATA								Read data from DD RAM or CG RAM	40µS**

* 1. When f_{cp} or f_{osc} is 250 KHz






2. Execution time changes when frequency changes. When f_{cp} or f_{osc} is 270KHz: $40\mu S \times \frac{250}{270} = 37\mu S$

** $t_{ADD} = 6\mu S$







Software Examples

8-bit operation (8-bits 2lines)

Function	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Display	Description
Power on delay												Initialization. No display appears.
Function set	0	0	0	0	1	1	0	0	X	X		Sets 8-bit operation and selects 2-line display and 5x7 dots character font. (Note: number of display lines and character fonts cannot be change after this.)
Display OFF	0	0	0	0	0	0	1	0	0	0		Turn off display.
Display ON	0	0	0	0	0	0	1	1	1	0		Turn on display and cursor.
Entry Mode Set	0	0	0	0	0	0	0	1	1	0		Set mode to increment the address by one and to shift the cursor to the right, at the time of write, to the DD/CG RAM Display is not shifted.

Write data to CG/DD RAM	1 0 0 1 0 1 0 1 1 1		Write "W". Cursor incremented by one and shift to right
Write data to CG/DD RAM	1 0 0 1 0 0 1 0 0 1		Write "I". Cursor incremented by one and shift to right
Write data to CG/DD RAM	• • •		Write "N", "T", "E", and "K".
Set DD address.	0 0 1 1 0 0 0 0 0 0		Set RAM address so that the cursor is propositioned at the head of the Second line.
Write data to CG/DD RAM	• •		Write "C", and "R".
Cursor or display shift	0 0 0 0 0 1 0 0 x x		Shift only the cursor position to the left.
Write data to CG/DD RAM	• • •		Write "O", "R", "P", "O", "R", "A", and "T".
Entry Mode Set	0 0 0 0 0 0 0 1 1 1		Set display mode shift at the time during writing operation.
Write data to CG/DD RAM	1 0 0 1 0 0 1 0 0 1		Write "I". Cursor incremented by one and shift to right. (The display move to left)
Write data to CG/DD RAM	• • •		Write other characters.
Return Home	0 0 0 0 0 0 0 0 1 0		Return both display and cursor to the original position. (Set address to zero.)

4-bit operation (4-bits 1line)

Function	RS RW D7 D6 D5 D4	Display	Description
power on delay			Initialization. No display appears.
Function set	0 0 0 0 1 0		Sets to 4-bit operation. In this case, operation is handled as 8-bits by initialization, and only this instruction completes with one write.
Function set	0 0 0 0 1 0 0 0 0 0 X X		Sets 4-bit operation and selects 1-line display and 5x7 dot character font on and resetting is needed. (number of display lines and character fonts cannot be changed hence after.)
Display ON/OFF Control	0 0 0 0 0 0 0 0 1 1 1 0		Turn on display and cursor.
Entry Mode Set	0 0 0 0 0 0 0 0 0 1 1 0		Set mode to increment the address by one and to shift the cursor to the right, at the time of write, to the DD/CG RAM display is not shifted.
Write data to CG/DD/RAM	1 0 0 0 0 0 1 0 0 1 1 1		Write "W". Cursor incremented by one and shift to right
same as 8-bit operation			

A continuación veremos una aplicación típica de manejo de display a 8 bits de datos, la forma de envío de los distintos caracteres es por el método de demora fija, utilizando para ello un microcontrolador de la línea FLASH HC908 de Motorola.

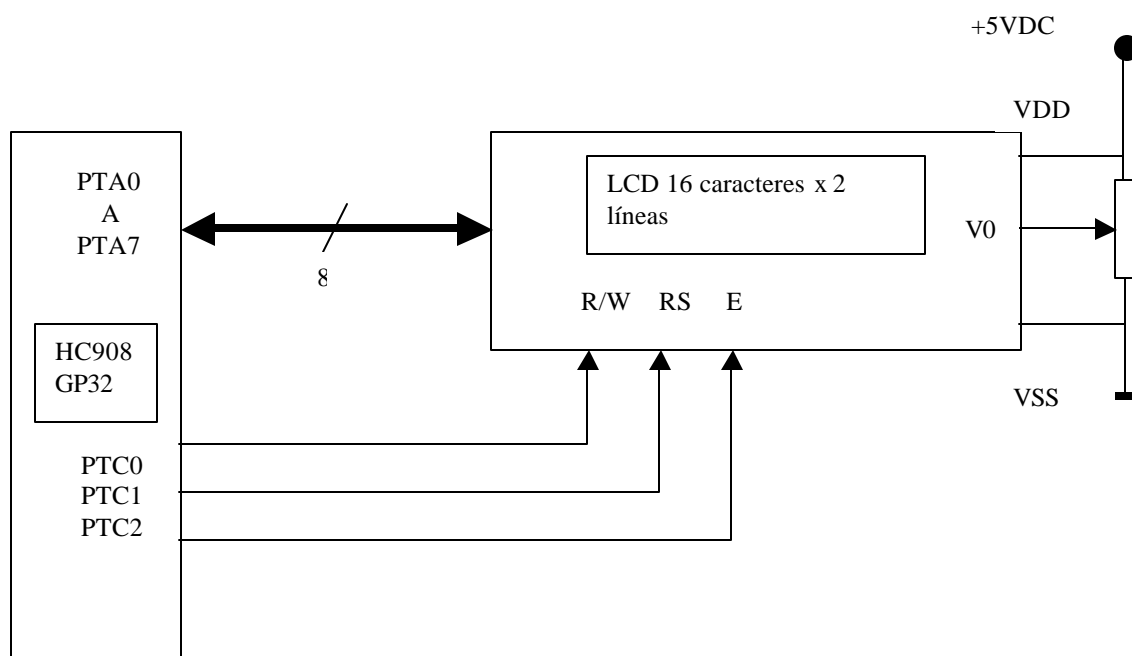
El MCU elegido es el MC68HC908GP32 pero la aplicación es valida para cualquier MCU de la familia.

Si bien el método de demora fija, es el menos eficiente en cuanto a velocidad de transferencia de información desde el MCU al display, es también el más sencillo de implementar por el usuario sin experiencia previa. Si el lector evaluara conveniente el uso

de un método más eficiente para una aplicación específica, sugerimos consultar la nota de aplicación de Motorola AN 1745 (e-www.motorola.com , sección "Microcontrollers") y la Información técnica disponible en el Web Site de WINTEK CORP (www.wintek.com.tw) .

A continuación veremos una aplicación típica de manejo de display a 8 bits de datos, la forma de envío de los distintos caracteres es por el método de demora fija.

El MCU elegido es el MC68HC908GP32 pero la aplicación es valida para cualquier MCU de la familia.



* DISP8BIT.ASM – PROGRAMA DE APLICACION PARA DISPLAY LCD *
* 16 CARACTERES X 2 LINEAS *

```
* El display aquí usado es marca WINTEK modelo WM-C1602Q1GLY      *
* pero puede aplicarse a cualquier tipo de display LCD inteligente  *
*                                                                    *
* Se utilizará un MC68HC908GP32CP para esta aplicación, con la siguiente config. *
* de puertos:                                                         *
*                                                                    *
* PORTA ---- PTA0 a PTA7 como salida de datos a conectar a DB0 a DB7 respec. *
*                                                                    *
* PORTC ---- PTC0 a PTC2 como salida líneas de control de la siguiente forma: *
*                                                                    *
* PTC0 ---- R/W                                                         *
* PTC1 ---- RS                                                           *
* PTC2 ---- E                                                            *
*****
```

```
base 10T                      ;Base Decimal por Default
include "gp32head.asm"        ;Equates grales. p/ GP32
```

```
*****
*   EQUATES   *
*****
```

```
RAMSPACE      EQU $0040      ;Comienzo de la RAM en el GP32
FLASHSPACE    EQU $8000      ;Comienzo de la FLASH en el GP32
```

```
*****
* VARIABLES RAM *
*****
```

```
                                ORG RAMSPACE      ;Comienzo de la RAM

TEMPA          RMB 1            ;Almacenamiento Temporal del ACC
TEMPX          RMB 1            ;Almacenamiento temporal de X
TEMPHX         RMB 2            ;Almacenamiento Temporal del H:X
PANTALLAS      RMB 1            ;Cdor. Tot. de pantallas a mostrar
DELAY1S        RMB 1            ;Cdor. Delay de 1 seg. (20 x 50mS)
```

```
;-----
;INICIALIZACIÓN DEL 908GP32
;Setup de los Configuration Registers
```



```
;-----  
  
START      MOV #$0B,CONFIG1      ;Set LVI5OR3 and COPD, enable STOP  
            MOV #$03,CONFIG2      ;Selecciono Fbus como SCI clock  
            NOP  
            NOP  
            MOV #$00,PORTA  
            MOV #$FF,DDRA          ;PORT A todo como SALIDA  
            MOV #00,PORTA  
            MOV #$00,PORTB  
            MOV #$00,DDRB          ;PORT B todo como entrada  
            MOV #$00,PORTB  
            MOV #$00,PORTC  
            MOV #$0F,DDRC          ;PORT C PTC0/3 ->OUT, PTC4/6 ->IN  
            MOV #$00,PORTC  
            MOV #$00,PORTD  
            MOV #$00,DDRD          ;PORT D todo como entrada  
            MOV #$00,PORTD  
  
            MOV #$00,PTAPUE        ;PORT A sin PULL UPS  
            MOV #$00,PTCPUE        ;PORT C sin PULL UPS  
            MOV #$00,PTDPUE        ;PORT D sin PULL UPS  
  
            SEI                    ; I MASK = 1 -> NO USO INTs  
                                    ; para esta aplicación
```

```
*****  
* PREPARO EL DISPLAY p/ POWER ON RESET *  
* del mismo (Ver gráficos aparte) y su *  
* inicialización de comandos           *  
*****
```

```
            JSR DLY50              ;DELAY 50MS DESPUES DEL POWER-ON  
            LDA #$38              ;FUNTION SET -8 BITS - 2LINE - 5X7  
            JSR WCTRL  
            JSR DLY50              ;DELAY 50MS  
            LDA #$38              ;FUNTION SET - 8BITS - 2LINE - 5X7  
            JSR WCTRL  
            JSR DLY50              ;FUNTION SET - 8BITS - 2LINE - 5X7  
            LDA #$38              ;FUNTION SET - 8BITS - 2LINE - 5X7  
            JSR WCTRL  
            LDA #$08              ;DISPLAY OFF  
            JSR WCTRL  
            LDA #$01              ;CLEAR DISPLAY  
            JSR WCTRL  
            LDA #$06              ;ENTRY MODE - INC ADDR - NO SHIFT  
            JSR WCTRL
```

```
*****  
* INICIALIZO EL LCD *  
*****
```

```
                LDA #$01                ;CLEAR DISPLAY
                JSR WCTRL
                LDA #$02                ;HOME CURSOR
                JSR WCTRL
                LDA #$38                ;FUNTION SET - 8BITS - 2LINE - 5X7
                JSR WCTRL
                LDA #$0C                ;DISPLAY ON , CURSOR OFF
                JSR WCTRL
                LDA #$06                ;ENTRY MODE - INC ADDR - NO SHIFT
                JSR WCTRL
```

```
*****
*  RUTINA PRESENTACION      *
*  PANTALLAS EN DISPLAY    *
*  CADA 1 SEG APROX.       *
*****
```

```
                CLRH                    ;H=$00, solo uso X como indice
AGAIN           CLRX                    ;APUNTO X AL COMIENZO DE TABLA
                MOV #5,PANTALLAS        ;5 PANTALLAS A MOSTRAR !!!
NEW_SCREEN      MOV #20,DELAY1S         ;CADA PANTALLA ESTARA 1 SEG
                JSR DISPLAY_L1
                JSR DISPLAY_L2
LOOP1S          JSR DLY50
                DEC DELAY1S
                BNE LOOP1S
                DEC PANTALLAS
                BNE NEW_SCREEN
                BRA AGAIN
```

```
*****
*  DISPLAY_L1/L2 - SUBROUTINAS DE PRESENTACION DE 1ERA*
*                  Y 2DA LINEA DISPLAY LCD CON MENSAJES *
*  CONTENIDOS EN UNA TABLA ( TDISPLAY )                *
*  SE ENTRA CON VALOR DE X ( COMIENZO DEL STRING )      *
*  Y SOLA LA RUTINA INCREMENTA X PARA MOSTRAR TODO     *
*  EL STRING EN PANTALLA                                *
*****
```

```
DISPLAY_L1      LDA #$02                ;POSICION HOME
                JSR WCTRL
                BRA DISPLOOP

DISPLAY_L2      LDA #$C0                ;POSICIONO COMIENZO 2DA LINEA
                JSR WCTRL

DISPLOOP        LDA TDISPLAY,X          ;ACC->1ER CARAC. DE TABLA TDISPLAY
                CMP #4                  ;SI CARACTER=$04 -> FIN 1ER STRING
                BEQ STE                  ;SINO SIGO MOSTRANDO OTRO CARACTER
                JSR WDAT                 ;ESCRIBO CARACTER EN LCD
                INCX                     ;INCR. PUNTERO AL PROX. CAR.
                BRA DISPLOOP            ;BUSCO OTRO CARACTER

STE             RTS                     ;***** RETORNO *****
```

```
*****
*  WCTRL - RUTINA DE ESCRITURA DE PALABRAS DE CONTROL *
*  EN EL LCD, ENTRO CON UN COMANDO EN EL ACC Y         *
*****
```

* RETORNO CON EL VALOR ORIGINAL DE X *

* DELAY 4,5 mS SI ACC=\$01 O \$02 SINO DE 120 MICROSEG *

```

WCTRL      STX TEMPX           ;SALVO X
            STHX TEMPHX        ;SALVO H:X
            STA PORTA          ;ESCRIBO PALABRA CONTROL EN LCD
            BSET 2,PORTC       ;E--->1 PULSO PARA ACTIVAR LCD
            NOP
            BCLR 2,PORTC       ;E--->0
            LDX #160           ;160*6*125nSEG = 120 MICROSEG
L120U      DECX                ;DELAY LOOP 120 MICROSEG
            NOP
            NOP
            BNE L120U          ;160,159.....0
            CMP #$02           ;SI COMANDOS = 01 o 02 EXTRA DELAY
            BHI ARN5M          ;SI COMANDOS >02 -> LO SALTO
            LDHX #2164
L5M         JSR ANRTS          ;JSR Y RTS TOMAN 9 CICLOS
            AIX #-1            ;2164x(9+2+3+3)x125Nseg = 4,6mS
            CPHX #0            ;
            BNE L5M
ARN5M      LDX TEMPX           ;RESTAURO X
            LDHX TEMPHX        ;RESTAURO H:X
ANRTS      RTS                ;*** RETORNO ****
    
```

* WDAT - RUTINA DE ESCRITURA DE DATOS EN EL LCD *

* ENTRO CON PALABRA DE DATOS EN EL ACC Y RETORNO *

* CON VALOR ORIGINAL DE X Y A *

```

WDAT      STX TEMPX           ;SALVO X
            STHX TEMPX        ;SALVO H:X
            STA TEMPX         ;SALVO A
            STA PORTA          ;ESCRIBO PALABRA DATOS EN EL LCD
            BSET 1,PORTC       ;RS -->1
            BSET 2,PORTC       ;E--->1 PULSO PARA LCD
            NOP
            BCLR 2,PORTC       ;E-->0
            BCLR 1,PORTC       ;RS-->0
            LDX #160           ;160*6*125nSEG = 120 MICROSEG
L120      DECX                ;DELAY LOOP 120 MICROSEG
            NOP
            NOP
            BNE L120          ;160,159...0
            LDA TEMPX         ;RESTAURO ACC
            LDX TEMPX         ;RESTAURO X
            LDHX TEMPHX        ;RESTAURO H:X
            RTS                ;*** RETORNO ***
    
```

* DLY50 *

* RUTINA DELAY 50 mS *

* con Xtal= 32 Mhz *
* T = 125 nSeg *

```
DLY50          STA TEMPA          ;Salvo ACC en RAM
                STHX TEMPXH        ;Salvo X:H en RAM
                LDHX #10000         ;H:X <-- 10000 para llegar a 10mS
                LDA #5              ;ACC <-- 5 para llegar a 50mS
OUTLP1          LDHX #10000        ;Loop interno de 10mS ya que:
INNRLP1         AIX #-1            ;10000 x8 ciclos x 125 nS = 10mS
                CPHX #0             ;
                BNE INNRLP1         ;
                DECA                ;ACC = 5,4,3...0
                BNE OUTLP1          ;5 x 10 mS = 50 mSeg
                LDA TEMPA           ;RECOBRO EL ACC DE LA RAM
                LDHX TEMPXH         ;RECOBRO X:H DE LA RAM
                RTS                 ;RETORNO
```

* TDISPLAY - TABLA DE STRINGS DE CARACTERES *
* PARA SUBROUTINA DISPLAY_L1 Y L2 *
* QUE SE UTILIZAN EN LA RUTINA "PRESENTACION"*

```
TDISPLAY       FCB 'BIENVENIDO !!!!!' ;
                FCB 'AL MUNDO DEL....' ;
                FCB $04                ;STE
                FCB 'DISPLAY INTELIG.' ;
                FCB 'LCD 16 X 2 LIN. ' ;
                FCB $04                ;STE
                FCB 'WINTEK CORP.....' ;
                FCB 'TIENE UN MODELO ' ;
                FCB $04                ;STE
                FCB 'PARA CADA UNA DE' ;
                FCB 'LAS APLICACIONES' ;
                FCB $04                ;STE
                FCB 'POSIBLES QUE UD.' ;
                FCB 'HABIA ESPERADO..' ;
                FCB $04                ;STE
```

* -----
* ASIGNACION DE VECTORES
* -----

* LOS VECTORES NO UTILIZADOS POR LAS INT's APUNTARAN A "START"

* -----

```
ORG TBVEC
FDB START
ORG ADCVEC
FDB START
ORG KBIVEC
FDB START
ORG SCITXVEC
FDB START
ORG SCIRXVEC
FDB START
ORG SCIERVEC
FDB START
ORG SPITXVEC
FDB START
ORG SPIRXVEC
FDB START
ORG T2OFVEC
FDB START
ORG T2CH1VEC
FDB START
ORG T2CH0VEC
FDB START
ORG T1OFVEC
FDB START
ORG T1CH1VEC
FDB START
ORG T1CH0VEC
FDB START
ORG PLLVEC
FDB START
ORG IRQ1VEC
FDB START
ORG SWIVEC
FDB START
ORG RESETVEC
FDB START
```

Hasta aquí, se há mostrado la aplicación típica de control del display a 8 bits de datos, ahora veremos un ejemplo de control a 4 bits de datos, que como se dijo anteriormente, es más eficiente en el uso de líneas dedicadas para controlarlo (7 líneas total, contra 11 líneas del método anterior), pero paga el precio de ser menos "claro" la forma de manejo del mismo.

Para este ejemplo, se há utilizado nuevamente un MC68HC908GP32, pero este ejemplo puede aplicarse a cualquier otro MCU de la familia HC908 FLASH de Motorola.

* *****

*PROGRAMA APLICATIVO DEL USO DE UN DISPLAY LCD INTELIGENTE *

```
* POR MEDIO DE 4 BITS DE DATOS --> TOTAL 7 LINEAS DE CONTROL      *
*                                                                    *
* Se usará el método de demora fija entre envío y envío de los  distintos caracteres al  *
* display.....                                                    *
* -----*
*                                                                    *
*   MC68HC908GP32 Interface                                         *
*                                                                    *
* LCD module (1x16 display, 4-bit interface)                       *
* Conexiones hechas al PORT B                                       *
* PB4-PB7 = DB4-DB7                                                *
* PB1 = RS (register select)                                         *
* PB2 = RW (Read ! Write)                                           *
* PB3 = E_LCD (enable)                                              *
*                                                                    *
*****
```

```
base 10T                      ;Base Decimal como default
include "gp32head.asm"        ;Header file defines registers, etc.
```

```
* -----
* CHANGE THIS TO MATCH OPPORATING FREQUENCY (MHz)
* To allow adaptable delays.
* -----
```

```
BUSFREQ EQU 4
USCOUNT EQU $0E
USDELAY EQU BUSFREQ*USCOUNT
```

```
; Miscellaneous equates
```

```
DELAYVAL EQU $50 ;Default value for delays
PRESCALER2 EQU %00000010 ;Prescaler bits determine timer freq.
PRESCALER EQU %00000000 ;Prescaler bits determine timer freq.
```

```
; ASCII character definitions
```

```
SPACE EQU $20 ;ASCII space
EOT EQU $04 ;ASCII end-of-text
CR EQU $0D ;ASCII carriage return
LF EQU $0A ;ASCII linefeed
TABLEEND EQU $FF
```

```
;LCD Control lines, E = enable, RS = Register Select (Data vs. Instruction)
;R/W = Read ! Write; Referidas a los pines del PORT B
```

RS EQU 1
RW EQU 2
E_LCD EQU 3

MAXCHARS EQU \$10 ;Maxima cantidad de caracteres por linea del LCD
MAXLINE EQU \$08

TOFMAXCOUNT EQU 2 ;Changed for monitor mode debugging (/2)
;TOFMAXCOUNT EQU 1

* -----
* RAM variables
* -----

ORG RAMSPACE ;Start of RAM

Counter RMB 1 ;Counter variable for delay
TempVar RMB 1 ;Temporary 8-bit storage
TempWord RMB 2 ;Temporary 16-bit storage

*-----

Count RMB 1 ;Counter variable
MsgIndex RMB 2 ;Index counter variable
MsgStart RMB 2 ;Stores starting point of string
TOFCount RMB 1
EventCount RMB 1
TempPtr RMB 2

* -----
* Program code
* -----

ORG FLASHSPACE ;Start of FLASH memory

Start:

```
;-----  
;908GP32 Initialization  
;Setup configuration registers  
;-----
```

```
MOV #$0B,CONFIG1    ;Set LVI5OR3 and COPD, enable STOP  
MOV #$03,CONFIG2    ;Select bus as SCI clock source  
NOP  
NOP  
CLR PORTA  
CLR PORTB  
CLR PORTC  
CLR PORTD  
MOV #$FF,PTAPUE      ;PORT A PULLUP ENABLE REGISTER  
MOV #$6F,PTCPUE      ;PORT C PULLUP ENABLE REGISTER  
MOV #$FF,PTDPUE      ;PORT D PULLUP ENABLE REGISTER  
MOV #$FF,DDRB        ;PORT B SET TO OUTPUTS  
MOV #$10,DDRC        ;PORT C BIT4 AS OUTPUT FOR LED  
CLR EventCount
```

```
* -----  
* Main program code  
* -----
```

```
;-----  
;Port B setup for LCD module  
;-----
```

```
BCLR RS,PORTB        ;Preset control line output levels  
BCLR RW,PORTB  
BCLR E_LCD,PORTB  
MOV #$FE,DDRB        ;Setup PB1-PB7 as outputs for LCD  
BCLR RS,PORTB        ;Preset control line output levels  
BCLR RW,PORTB  
BCLR E_LCD,PORTB
```

```
CLI                  ;clear interrupt mask
```

```
;-----  
;LCD setup  
;-----
```

```
JSR LCDInit          ;Initialize LCD
```

```
* -----  
* Demo program 1
```


* Start listing information strings

* -----

InfoDemo:

LDHX #FeatureList ;Load offset of desired string

MainLoop:

```
JSR ShowString
LDA #$FA ;Number of mS delay between strings
JSR msDelay ;F0 = 240mS X 4 times = 1S Aprox
LDA #$FA
JSR msDelay
LDA #$FA
JSR msDelay
LDA #$FA
JSR msDelay
```

```
NextMsg LDA ,X
        CMP #EOT
        BEQ GotNext
        AIX #1
        BRA NextMsg
GotNext AIX #1
        LDA ,X
        CMP #TABLEEND
        BEQ InfoDemo
        BRA MainLoop
```

; "0123456789ABCDEF" Max Characters

FeatureList:

```
FCB "BIENVENIDOS !!! "
FCB EOT
FCB "AL INTERESANTE "
FCB EOT
FCB "MUNDO..... "
FCB EOT
FCB "DE LOS NUEVOS"
FCB EOT
FCB "DERIVATIVOS HC08"
FCB EOT
FCB "FLASH MOTOROLA"
FCB EOT
FCB "MC68HC908GR8/4"
```

FCB EOT
FCB "MC68HC908MR32/16"
FCB EOT
FCB "MC68HC908SR12"
FCB EOT
FCB "MC68HC908KX8/2"
FCB EOT
FCB "MC68HC908RK2"
FCB EOT
FCB "MC68HC908JB8"
FCB EOT
FCB "Y AL PORTFOLIO"
FCB EOT
FCB "DE HERRAMIENTAS"
FCB EOT
FCB "PARA DESARROLLO"
FCB EOT
FCB "EVAL08GP / JL"
FCB EOT
FCB "E-FLASH08 !!"
FCB EOT
FCB "Y MUCHO MAS !!"
FCB EOT
FCB "*** FIN *****"
FCB EOT
FCB TABLEEND

* -----
* LCD INITIALIZATION
* Called on startup to initialize the LCD into 4-bit mode
* Don't use the LCDControl subroutine at first, because initially
* the LCD is in 8-bit interface mode.
* -----

LCDInit:

```
;-----  
;8-bit interface at first  
;-----  
    LDA #$0F          ;Wait 15ms  
    JSR msDelay  
  
    LDA #$30          ;Function set  
    JSR LCD8Ctrl  
    LDA #5            ;Wait 5ms  
    JSR msDelay
```

LDA #\$30 ;Function set
JSR LCD8Ctrl

LDA #1 ;Wait 1ms
JSR msDelay

LDA #\$30 ;Function set
JSR LCD8Ctrl

LDA #\$20 ;Function set first set up 4-bit
JSR LCD8Ctrl

;-----
;4-bit operation
;-----

LDA #\$28 ;Second 4-bit function set
JSR LCDCtrl ;Also 2-line setup

LDA #\$06 ;Increment, no shift
JSR LCDCtrl

LDA #\$01 ;Clear display, cursor home
JSR LCDCtrl

LDA #\$0C ;Display on, cursor off, blink off
JSR LCDCtrl

RTS

* -----
* If the LCD is in 8-bit mode, just send the upper data bits through
* the lower port C bits
* Acc contains control byte
* -----

LCD8Ctrl:

AND #\$F0 ;Mask out lower data bits
PSHA ;Store value to send temporarily
LDA PORTB ;Don't change other PORTB pins
AND #\$0F ;Mask out UPPER nibble
ADD 1,SP ;Add value to send
STA PORTB ;Store to PORTB

BSET E_LCD,PORTB ;Toggle enable line

BCLR E_LCD,PORTB

JSR _40usDelay ;40us setup time

PULA ;Deallocate stack data
RTS

- * -----
- * Write LCD control byte in 4-bit mode
- * For 4-bit mode, need to send the nibbles one at a time, high nibble first
- * Acc contains control byte to send
- * This routine is also used for Address writes to LCD, when MSbit of
- * data byte is set. (i.e. \$80 for address 00)
- * -----

LCDCtrl:

```
;-----  
;Upper nibble  
;-----  
PSHA ;Store data on stack  
AND #$F0 ;Mask out lower nibble  
PSHA ;Store upper nibble on stack  
LDA PORTB ;Load PORTB contents  
AND #$0F ;Mask out UPPER nibble  
ADD 1,SP ;Add the data nibble  
STA PORTB ;Present upper nibble to LCD  
  
BSET E_LCD,PORTB ;Toggle Enable line  
BCLR E_LCD,PORTB
```

JSR _40usDelay ;40us setup time

```
;-----  
;Lower nibble  
;-----  
PULA ;Deallocate last temp storage  
PULA ;Get original data byte  
AND #$0F ;Mask out upper nibble  
NSA ;Put LOWER nibble in UPPER nibble  
PSHA ;Store onto stack  
LDA PORTB ;Get existing PORTB data  
AND #$0F ;Mask out lower nibble  
ADD 1,SP ;Add lower nibble of data byte  
STA PORTB ;Store to PORTB
```

```
BSET E_LCD,PORTB    ;Toggle Enable line
BCLR E_LCD,PORTB
```

```
PULA                ;Deallocate temp storage
CMP #$10             ;Longer delay for commands 1 or 2
BEQ LCLonger
CMP #$20
BEQ LCLonger
```

```
JSR _40usDelay      ;40us for any other command
RTS                  ;Return
```

LCLonger:

```
LDA #2
JSR msDelay
RTS
```

```
* -----
* Write data byte to LCD, using 4-bit mode
* Acc contains data byte to send
* -----
```

LCDDData:

```
;-----
;First nibble
;-----
PSHA                ;Store data on stack temporarily
AND #$F0            ;Mask out lower nibble
PSHA                ;Store upper nibble on stack
LDA PORTB           ;Load PORTB contents
AND #$0F            ;Mask out UPPER nibble
ADD 1,SP            ;Add the data nibble

STA PORTB           ;Store to PORTB

BSET RS,PORTB       ;Set RS for control

BSET E_LCD,PORTB    ;Toggle Enable line
BCLR E_LCD,PORTB
```

```
;-----
;Second nibble
```

```
;-----  
PULA          ;Deallocate temp storage  
PULA          ;Get original data  
AND #$0F      ;Mask out upper nibble  
NSA           ;Put LOWER nibble in UPPER port pins  
PSHA          ;Store onto stack  
LDA PORTB  
AND #$0F  
ADD 1,SP  
STA PORTB  
  
BSET E_LCD,PORTB    ;Toggle enable line  
BCLR E_LCD,PORTB  
  
JSR _40usDelay    ;40us setup time  
JSR _40usDelay    ;40us setup time  
  
BCLR RS,PORTB     ;Clear RS for data  
  
JSR _40usDelay    ;40us setup time  
JSR _40usDelay    ;40us setup time  
  
PULA          ;Deallocate temp storage  
RTS
```

```
* -----  
* Clear the LCD display by sending the appropriate command byte  
* -----  
LCDClear:
```

```
    LDA #$01      ;Clear display AND home cursor  
    JSR LCDCtrl  
    RTS
```

```
LCDHome    LDA #$02  
           JSR LCDCtrl  
           RTS
```

```
* -----  
* DELAY ROUTINES
```

* -----
* 1ms delay loop, causes _roughly_ 1.3ms delay @ fop = 8MHz
* uses constant for loop control
* cycles = 4 + X(6+7+1275) + 3 + 6
* cycles = 13 + X(1288)
* where X is value loaded into Acc
* Causes 1.3ms delay for BUSFREQ values of (1-8 integer values)
* -----

```
_1msDelay:
    PSHA                ;2 cycles
    LDA #BUSFREQ        ;2
DLLoop    DBNZA DLSub    ;3
          BRA DLDone     ;3
DLSub     MOV #$FF,Counter ;4
          DBNZ Counter,*  ;5
          BRA DLLoop     ;3
DLDone    PULA          ;2
          RTS            ;4
```

* -----
* Variable ms delay loop
* Calls DelayLoop Acc number of times
* -----
msDelay JSR _1msDelay
 DBNZA msDelay
 RTS

* -----
* 40usec delay routine. Important for LCD module, which requires 40us
* delay for many of its commands, including the "data setup time"
* 6+X(3)
* Where X = BUSFREQ*USCOUNT = USDELAY
* Provides _roughly_ 40us delay
* -----

```
_40usDelay:
    LDA #BUSFREQ*USCOUNT ;2
    DBNZA *                ;3
    RTS                    ;4
```

* -----
* Big delay loop

* Calls msDelay a predefined number of times

* -----

BigDelay:

PSHA
LDA #\$0A
JSR msDelay
PULA
RTS

* -----

* Bigger delay loop

* Calls msDelay a predefined number of times

* -----

BiggerDelay:

PSHA
LDA #\$FF
JSR msDelay
LDA #\$FF
JSR msDelay
LDA #\$FF
JSR msDelay
LDA #\$FF
JSR msDelay
PULA
RTS

* -----

* Vector trap to detect spurious exceptions. (no RTI)

* -----

Trap LDHX #ErrorMsg
 JSR ShowString
 BRA *

ErrorMsg:

FCB "Vector trapped"
FCB EOT

* -----

* Scroll subroutines

*-----
* Initialize the message variables for the desired output string
* Register A contains the offset of desired message.
*-----

```
LoadMsg      STHX MsgIndex      ;Setup the message index
              STHX MsgStart      ;Store the start of the message
              RTS                ;Return
```

*-----
* Update the LCD with current portion of string to be displayed
*-----

```
UpdateLCD:
              LDHX MsgIndex      ;Start at current index into message
              JSR ShowString      ;Show current portion of string
              LDHX MsgIndex
              CPHX MsgStart
              BNE ULgo
              LDA #$30
              JSR msDelay
ULgo          AIX #1
              STHX MsgIndex      ;Increment the index
              RTS                ;Return
```

*-----
* Show the current string portion on the display.
* When called, the X register contains the index offset.
*-----

```
ShowString:
              JSR LCDClear
              CLR Count           ;Clear the counter variable
              LDA #$80            ;Starting address
              JSR LCDCtrl
```

```
NextByte:
              LDA ,X              ;Load ASCII byte of string
              CMP #EOT            ;Check for end of string
              BEQ Padding         ;Last character reached
              JSR LCDDData
              AIX #1              ;Increment the index
              INC Count           ;Increment the counter
              LDA Count           ;Check the counter
              CMP #MAXCHARS       ;for LCD display length
              BEQ Done            ;End of display line reached
              CMP #MAXLINE
              BNE SSCont
              LDA #$C0            ;Set address for line 2
              JSR LCDCtrl
SSCont        BRA NextByte       ;Ready the next byte
```

Padding	LDA Count	;Pad the rest of the display with spaces
	CMP #\$00	;See if string has scrolled off display
	BEQ Reset	;Need to reset string
	CMP #MAXCHARS	;Check for end of display
	BEQ Done	;Finished displaying padding spaces
	INC Count	;Increment counter
	JSR BlankSpace	;Put space in current display position
	BRA Padding	;Repeat
Reset	JSR BlankSpace	;Show a final space in first position
	LDHX MsgStart	;Load start of message index
	AIX #-1	;Compensate for INCX in UpdateLCD after RTS
	STHX MsgIndex	;Record new message index
Done	RTS	;Return

* -----
* Sends an ASCII space character to the LCD
* -----

BlankSpace:

LDA #\$20
JSR LCDData
RTS

* -----
* VECTOR ASSIGNMENTS
* -----
* Trap unused vectors to indicate errors
* -----

ORG TBVEC
FDB Trap
ORG ADCVEC
FDB Trap
ORG KBIVEC
FDB Trap
ORG SCITXVEC
FDB Trap
ORG SCIRXVEC
FDB Trap
ORG SCIERVEC
FDB Trap
ORG SPITXVEC
FDB Trap
ORG SPIRXVEC
FDB Trap
ORG T2OFVEC
FDB Trap
ORG T2CH1VEC

FDB Trap
ORG T2CH0VEC
FDB Trap
ORG T1OFVEC
FDB Trap
ORG T1CH1VEC
FDB Trap
ORG T1CH0VEC
FDB Trap
ORG PLLVEC
FDB Trap
ORG IRQ1VEC
FDB Trap
ORG SWIVEC
FDB Trap
ORG RESETVEC
FDB Start