Cika Electrónica

Nota de Aplicación: CAN-051
Título: **Manejo de displays LCD alfanuméricos inteligentes con Holtek**
Autor: Sergio R. Caprile, Senior Engineer

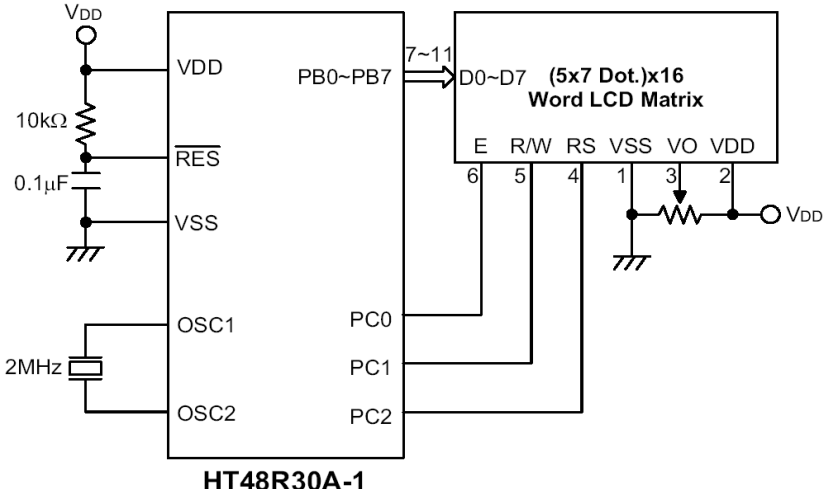
Revisiones	Fecha	Comentarios
0	31/05/06	

La presente nota de aplicación está basada en la nota de aplicación HA0013E de Holtek. El objetivo es interiorizarnos en el desarrollo de una interfaz para conectar un módulo LCD alfanumérico inteligente a un microcontrolador Holtek. Analizaremos más tarde el software de control y un simple programa demostración.

La interfaz que utilizamos es de 8 bits con lectura y escritura del controlador LCD. La nota de aplicación original es algo más completa, permitiendo compilado condicional para 4-bits ú 8-bits. A los fines de hacer de ésta una nota introductoria, decidimos simplificarla.

Hardware

Utilizaremos el port B completo para los datos, y algunos pines del port C para las señales de control. Como la gran mayoría de los microcontroladores, los ports de I/O son bidireccionales. Ambos puertos están presentes con todas las líneas necesarias desde 48R10/48E10 en adelante.



Software

Comenzamos con las rutinas básicas de control del display, aprovechamos para comentar las características del entorno de desarrollo (IDE)

Incluimos los archivos que tienen definiciones de los registros del procesador, los pines utilizados, etc:

```
include ht48r30a-1.inc

include lcmglob.inc
include command.inc
```

Luego declaramos las subrutinas y variables que serán accedidas desde otros módulos, como por ejemplo el programa principal:

```
public delay
public send_char
public send_cmd
```

```
public send_string
public initLCD
public clsLCD
```

Luego, declaramos el área de variables en RAM y reservamos espacio para ellas:

```
dsec    .section 'data'

dtmp    db ?
dtmp2   db ?
tmp     db ?
```

El signo *?* luego de la directiva *db* indica que se trata de un byte y no se lo inicializa a ningún valor

A continuación, declaramos el área de código y escribimos las rutinas correspondientes. De ser necesario, se puede volver a definir otro tipo de área diferente en cualquier parte del archivo. Al nodefinir dirección, el área es relocalizable, y el linker decide qué direcciones utilizar.

```
scode   .section 'code'

send_cmd:       mov tmp,a
                call busy_chk
                mov a,tmp
snd_cmd:        mov lcm_data,a              ; pone comando en bus
                clr lcm_ctrl.rw             ; RW=W
                clr lcm_ctrl.rs             ; RS=0 (comando)
                set lcm_ctrl.e              ; pulso en E
                clr lcm_ctrl.e
                ret

busy_chk:       set lcm_data_ctrl           ; pone bus como entradas
                clr lcm_ctrl.rs             ; RS=0 (comando)
                set lcm_ctrl.rw             ; RW=R
                set lcm_ctrl.e              ; sube E
                mov a,lcm_data              ; lee bus
                clr lcm_ctrl.e              ; baja E
                sz acc.7                    ; bit 7 = 0 ?
                jmp busy_chk                ; no, loop
                clr lcm_ctrl.rw             ; sí, RW=W
                clr lcm_data_ctrl           ; pone bus como salidas
                ret

send_char:      mov tmp,a
                call busy_chk
                mov a,tmp
write_char:     mov lcm_data,a              ; pone dato en bus
                clr lcm_ctrl.rw             ; RW=W
                set lcm_ctrl.rs             ; RS=1 (dato)
                set lcm_ctrl.e              ; pulso en E
                clr lcm_ctrl.e
                ret


initLCD:        mov a,38h                   ;  8 bit, 2 lineas, caracteres 5x7
                call snd_cmd

                call busy_chk
                mov a,06h                   ; inc address, desplaza cursor
                call snd_cmd
                call busy_chk
                mov a,0Ch                   ; sin cursor, ni blink
                call snd_cmd

clsLCD:         call busy_chk
                mov a,lcm_cls               ; clear screen
                call snd_cmd
                call busy_chk
                mov a,cursor_home
                jmp snd_cmd

; Llamar con dirección del string (dentro del bloque de strings en última página) en A
send_string:    mov TBLP,A                  ; inicializa puntero para lectura desde ROM
```

```
ssloop:         call busy_chk
                tabrdl ACC              ; lee un caracter en el acumulador (mapeado como RAM)
                sz ACC                          ; fin de string ? (ACC = 0?)
                jmp ssl1
                ret                             ; sí
ssl1:           call write_char                 ; no, lo escribe
                inc TBLP                        ; incrementa puntero
                jmp ssloop                      ; repite


delay:          mov dtmp,a
                set dtmp2
drep:           sdz dtmp2
                jmp drep
                sdz dtmp
                jmp drep
                ret
```

Como puede observarse, la sintaxis es clara y los mnemónicos son fácilmente memorizables, coincidentes con lo habitualmente utilizado en otros microcontroladores. La rutina de impresión de strings es clara y compacta, aprovechando la capacidad de lectura de tablas en ROM, mediante la cual un registro puntero permite extraer datos constantes de la memoria de programa. El hecho de disponer del acumulador mapeado en la RAM nos abre nuevas posibilidades y simplifica la operación.

Finalmente, el programa principal

```
include ht48r30a-1.inc

extern delay:near
extern send_char:near
extern send_cmd:near
extern send_string:near
extern initLCD:near
```

Las variables y rutinas globales las declaramos como *extern*, dado que son externas a este módulo, y el agregado *near* define que son rutinas y no variables.

El resto es similar. Al declarar el área de código, esta vez nos interesa que comience en la posición 0x0000 para definir el vector de reset, lo cual indicamos mediante

```
code    .section at 0 'code'
```

Declaramos con la directiva ORG el vector de reset y el comienzo del área de código, dejando espacio para vectores de interrupción en el futuro. También podríamos haberlos declarado en secciones diferentes

```
                ORG 00h
                jmp start

                ORG 0Ch
start:          clr pa
                set pac                         ; PA = entradas
                clr pc                          ; (E=0)
                clr pcc                         ; PC = salidas
                clr pb
                clr pbc                         ; PB = salidas
                mov a,0ffh
                call delay

                call initLCD

                mov a,txt1                      ; Esto carga la parte baja de la direción
                call send_string                ; que es el offset dentro del bloque y es
                                                ; justamente lo que queremos
                mov a,0C0h
                call send_cmd                   ; second line

                mov a,txt2
                call send_string                ; envía string al LCD

lp:             jmp lp
```

Por último, el área donde se alojan los textos. Como empleamos la instrucción tabrdl, dado que ésta lee desde la última página[1] de memoria de programa (256 bytes), deberemos alojarlos allí. Nos referiremos individualmente a cada uno de ellos mediante su offset dentro de dicha página, como observamos en la llamada a *send_string*, que carga el acumulador con dicho offset..

```
textos .section at 700h 'code'       ; última página de flash 48R30: 2K words -> 0000 a 07FF

txt1:   dw      67,105,107,97,32,69,108,101,99,116,114,111,110,105,99,97,0
txt2:   dw      72,111,108,116,101,107,32,76,67,68,32,100,101,109,111,0
```

IDE

La operación del entorno de desarrollo es similar a lo que se observa normalmente en otras marcas. Generamos un proyecto, para agregar estos archivos al proyecto seleccionamos en el menú *Project -> Edit*; y para ensamblar o compilar disponemos del clásico botón de *Build*. El IDE incluye un simulador para todos los micros OTP, y puede trabajar junto con un emulador en circuito muy económico para estos mismos micros o los MTP con EEPROM (serie 48E). Se incluye además, sin cargo extra, un compilador C basado en un subset del ANSI C standard, sin soporte para coma flotante, pero con soporte de bitfields y algunas extensiones interesantes.

Las opciones de hardware del micro como clock, función del pin PB0/BZ, las seleccionamos en *Tools -> Configuration options*. Para este caso, inhabilitamos BZ (pues usamos PB0) y el WDT.

1 No es que la memoria esté paginada, sino que la nomenclatura de Holtek se refiere de esa forma a los bloques o segmentos de 256 bytes.