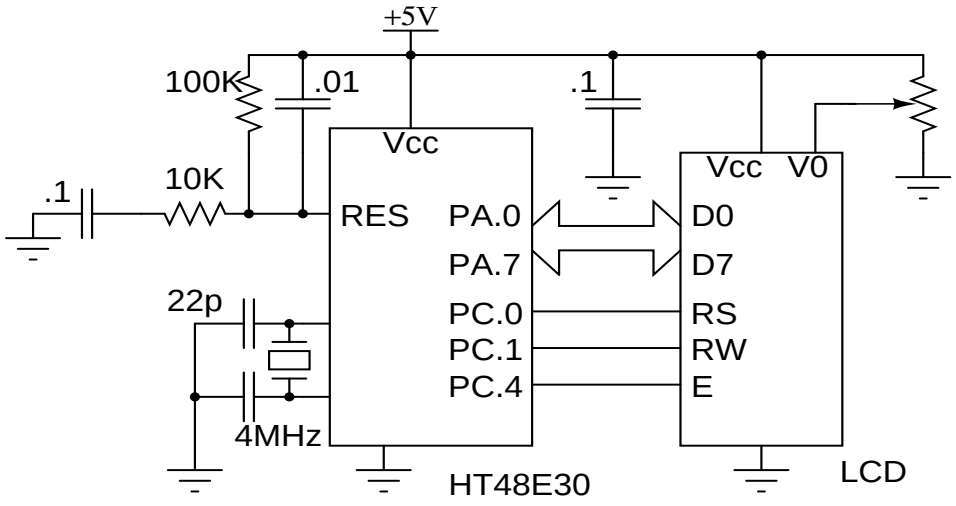
Nota de Aplicación: CAN-074
Título: **Manejo de displays LCD alfanuméricos inteligentes con Holtek**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	16/03/07	

La presente nota de aplicación desarrolla un esquema de control para displays LCD alfanuméricos, basada en la serie HT48E de Holtek. El código desarrollado es assembler, pero se proveen ejemplos de uso del mismo tanto desde C como desde assembler. Aquellos lectores no interesados en el funcionamiento interno del código, pueden saltear dicha parte y acceder directamente a los ejemplos de uso. Para una explicación detallada de los requisitos a cumplir para utilizar C y assembler simultáneamente, se recomienda la lectura del tutorial CTU-010.

Hardware

Utilizaremos el port A 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 48E30 en adelante.



Software

Las rutinas básicas de control del display son las siguientes:

```
_send_cmd:
                call busy_chk
                mov a,send_cmd0
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:
                call busy_chk
                mov a,send_char0
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
```

La inicialización y limpieza del display se realiza mediante las siguientes rutinas:

```
_init_LCD:
                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
_cls_LCD:
                call busy_chk
                mov    a,lcm_cls             ; clear screen
                call snd_cmd
                call busy_chk
                mov    a,cursor_home
                jmp snd_cmd
```

Proveeremos ahora un par de rutinas para imprimir strings. Debido a que la arquitectura de Holtek es Harvard, dispondremos de una rutina para textos dinámicamente generados, es decir, en RAM, y otra para textos fijos, en ROM. Los strings deben ubicarse a continuación de la subrutina, debido a la utilización de la instrucción TABRDC, que lee una tabla en la misma página de flash. Si el espacio no alcanza, deberá modificarse el código para realizar un call en lugar de esta instrucción[1]; sin embargo, esto adiciona un nivel de utilización del stack y preferimos evitarlo.

```
_send_string_RAM:
                mov A,send_string_RAM0       ; dirección del texto en RAM
                mov MP0,A                    ; puntero
ssloop:         call busy_chk
                mov A,IAR0                   ; acceso indirecto
                sz ACC                       ; 0 => fin de string
                jmp ssl1
                ret
ssl1:           call write_char              ; imprime
                inc MP0                      ; siguiente
                jmp ssloop


ssrom   .section INPAGE 'CODE'
_send_string_ROM:
                mov A,send_string_ROM0       ; número de string en la tabla
                call setstring               ; obtiene offset en página de ROM
                mov TBLP,A                   ; puntero en ROM
ss2lop:         call busy_chk
                tabrdc ACC                   ; lee
                sz ACC                       ; 0 => fin de string
                jmp ss2l1
```

1 Encontrará un ejemplo de esto en las rutinas de impresión de íconos en la nota de aplicación sobre displays gráficos, CAN-075.

```
                ret
ss2l1:          call write_char             ; imprime
                inc TBLP                    ; siguiente
                jmp ss2lop

; strings

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

ss              .section INPAGE 'CODE'
setstring:
                addm a,PCL                  ; traduce de string# a offset
                ret a,OFFSET txt1
                ret a,OFFSET txt2
                ;
```

Utilización desde C

Incluimos un header file, lcd.h, de modo que el compilador pueda chequear la sintaxis y el usuario conozca las funciones que debe llamar.

Inicialización

Para inicializar el LCD, realizamos lo siguiente:

```
        _pa=0;
        _pac=0;                             // PA = salidas
        _pc=0;                              // (E=0)
        _pcc=0;                             // PC = salidas
        _delay(10000);

        INIT_LCD();
```

Caracteres individuales

Podemos enviar caracteres al display llamando a la rutina correspondiente:

```
                SEND_CHAR('A');
```

Envío de comandos al display

Podemos enviar comandos al display llamando a la rutina correspondiente:

```
        SEND_CMD(0xC0);                     // ubica en segunda línea
```

Strings en RAM

Para imprimir un texto en RAM, llamamos a la rutina correspondiente con la dirección del texto como parámetro:

```
        SEND_STRING_RAM(buffer);
```

Strings en ROM

El ejemplo siguiente muestra la forma de imprimir un string en ROM. El string corresponde a los definidos en assembler en la sección anterior, y deberá cargarse en dicho archivo (o modificarlo para que los incluya.desde otro).

```
        SEND_STRING_ROM(0);
```

Si deseamos manejar los strings desde C, el tema es algo más complicado. Debido a que la arquitectura de Holtek es Harvard, no es simple pasar un puntero a una constante como parámetro. El compilador C que el fabricante provee no lo soporta, por lo que deberemos definir al string como un array y tratarlo como tal. Una posibilidad es escribir una pequeña rutina para cada string, que compilada ocupa unos pocos bytes:

```
const char str00[16]= "Cika Electronica";
const char str01[15]= "Holtek LCD demo";

        for(i=0;i<16;i++)
                SEND_CHAR(str01[i]);
```

```
SEND_CMD(0xC0);                         // ubica en segunda línea

for(i=0;i<15;i++)
        SEND_CHAR(str00[i]);
```

Otra alternativa es armar una tabla de strings y pasar a una subrutina el número de string, el inconveniente es que todos los strings de la tabla ocupan el mismo espacio, es decir, deben tener el mismo largo:

```
const char strs[2][16]= {
"Cika Electronica",
"Holtek LCD demo "
};

void ss(unsigned int string, unsigned int len)
{
unsigned int i;
        for(i=0;i<len;i++)
                SEND_CHAR(strs[string][i]);
}
```

Dicha subrutina la llamamos de la siguiente forma:

```
ss(0,16);
SEND_CMD(0xC0);
ss(1,16);
```

Utilización desde assembler

Si bien es posible evitar el paso de colocar el parámetro en RAM y llamar directamente a un punto de entrada más avanzado, preferimos mantener una uniformidad de llamado a las rutinas tanto desde C como assembler. Ante la imperiosa necesidad de espacio el developer puede recurrir al artilugio anterior si lo desea.

Inicialización

Para inicializar el LCD, realizamos lo siguiente:

```
clr pa
clr pac                                 ; PA = salidas
clr pc                                        ; (E=0)
clr pcc                                 ; PC = salidas

; delay

call _init_LCD
```

Caracteres individuales

Podemos enviar caracteres al display llamando a la rutina correspondiente:

```
mov a,041h
mov send_char0,A
call _send_char                         ; 'A'
```

Envío de comandos al display

Podemos enviar comandos al display llamando a la rutina correspondiente:

```
mov a,0C0h
mov send_cmd0,A
call _send_cmd                          ; segunda línea
```

Strings en RAM

Para imprimir un texto en RAM, llamamos a la rutina correspondiente con la dirección del texto como parámetro:

```
mov a,OFFSET buffer
mov send_string_RAM0,A
call _send_string_RAM
```

Strings en ROM

El ejemplo siguiente muestra la forma de imprimir un string en ROM. El string corresponde a los definidos en assembler en la sección anterior, y deberá cargarse en dicho archivo.

```
mov a,0
mov send_string_ROM0,A
call _send_string_ROM
```

Si bien el archivo provisto tiene los strings ahí mismo, es posible incluirlos desde otro, generado por una simple rutina que toma los strings de un archivo de texto y los convierte a las cadenas en decimal o hexa necesarias.