Nota de Aplicación: CAN-082
Título: **Teclado matricial con Holtek**
Autor: Sergio R. Caprile, Senior Engineer

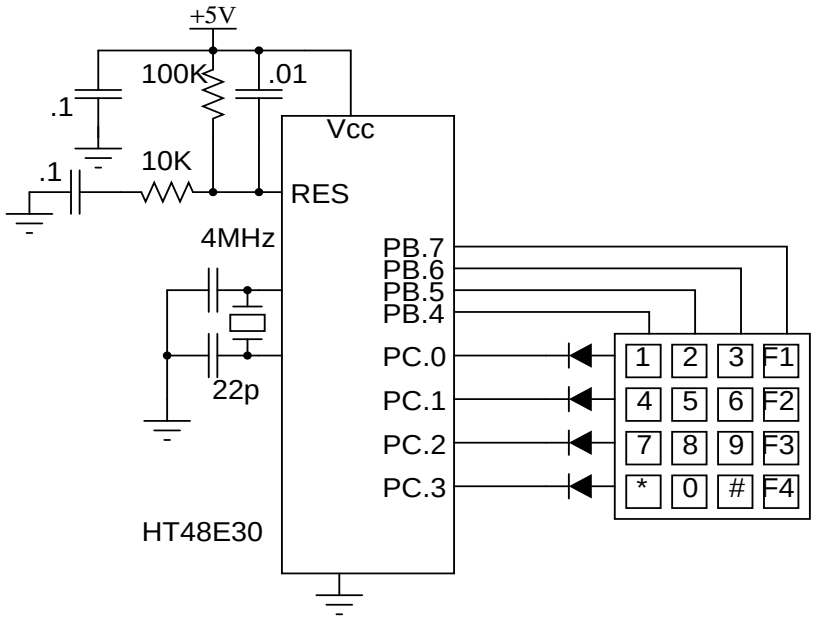
Revisiones	Fecha	Comentarios
0	15/08/07	

En la nota de aplicación CAN-079 desarrollamos un esquema de control para displays de 7 segmentos, que incluía lectura de un teclado matricial. Para aquellas aplicaciones en las cuales no se utilizan displays de 7-segmentos pero se requiere la lectura del teclado, presentamos aquí un pequeño esquema de scan que puede utilizarse a demanda (como muestra la nota de aplicación) o incluirse en un handler de interrupciones, si se toman los recaudos necesarios de stack como viéramos en CAN-079.

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 algunos pines del port C para la selección de las filas. La parte alta del port B la utilizaremos para retorno del teclado; el pull-up interno nos polariza convenientemente las entradas.



Software

La rutina básica de scan es la siguiente:

```
codek   .section 'CODE'
_kscan:         mov A,ROWS
                mov IDX,A
                mov A,LMOST
                mov ROW,A
                clr _LAST_K[1]
```

```
                jmp kl1                                                 ; 9 cycles
next:           rrc ROW
kl1:            mov A,ROWSOFF
                orm A,row_data
                mov A,ROW
                cpl ACC
                andm A,row_data
                mov A,kbd_data                          ; get returns
                cpl ACC
                and A,0F0h                                      ; only high nibble
                sz Z
                jmp nokey
                or A,ROW                                        ; add scan
                orm A,_LAST_K[1]                        ; store
nokey:  sdz IDX
                jmp next                                        ; 13/14 cycles (14 on key
pressed)
                mov A,_LAST_K[1]
                mov _LAST_K,A
                ret

;cycles: 11+(13/14)*ROWS-1+4 ~= 14+13*ROWS (+1 if key pressed)(including call+ret)
; 66 cycles for a 4x4 kbd
```

Para traducir el código de scan a un valor numérico más fácilmente utilizable, buscamos el código en una tabla, el offset dentro de la tabla es el valor buscado (la rutina es la misma desarrollada en CAN-079). Esto tiene además la ventaja de ignorar códigos de scan de múltiples teclas simultáneamente.

Utilización desde C

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

Inicialización

No existe rutina de inicialización, llamamos a la rutina de scan cuando deseamos leer el teclado. Para inicializar el hardware, realizamos lo siguiente:

```
        _pc=0;
        _pac=0;                         // PA0-7 = outputs
        _pbc=0xFF;                      // PB0-7 = inputs
        _pbc&=~(1<<0);                  // PB1 = output
        _pcc=0;                         // PC0-7 = outputs
```

Lectura del teclado

Para leer el teclado llamamos a la función *KSCAN()*, disponemos entonces del código de scan como valor devuelto por la función o en la variable global *LAST_K*. Para convertirlo a un código numérico correlativo, más manejable, podemos utilizar la función *FINDK()*:

```
        while(1) {
                if(i=KSCAN()){
                        if((i=FINDK(LAST_K))!=0xFF){  // 0xFF => ninguna o muchas, 0=>0, etc
                        // procesa
                        }
                }
        }
```

Utilización desde assembler

Inicialización

No existe rutina de inicialización, llamamos a la rutina de scan cuando deseamos leer el teclado. Para inicializar el hardware, realizamos lo siguiente:

```
                clr PC
                clr PAC                         ; PA0-7 = outputs
                set PBC                         ; PB0-7 = inputs
                clr PBC.1                       ; PB1 = output
```

```
                clr PCC                         ; PC0-7 = outputs
                set PGC                         ; PG0 = input
```

Lectura del teclado

Para leer el teclado llamamos a la subrutina *_kscan*, disponemos entonces del código de scan en el acumulador y en la variable *_LAST_K*. Para convertirlo a un código numérico correlativo, más manejable, podemos utilizar la subrutina *_findk*, a la que le pasamos el código de scan como parámetro en *findk0* y devuelve el resultado en A:

```
loop:           call _kscan                     ; lee teclado y devuelve _LAST_K en A
                mov findk0,A                    ; guarda como parámetro
                sz findk0                       ; espera una tecla (0 => no key)
                jmp gotk
                jmp loop
gotk:           call _findk                     ; devuelve 0 si está el 0 presionado, etc
                                                ; 0xFF si no hay ninguna (o demasiadas)
                                                ; tecla(s) apretada(s)
                ; procesa
                jmp loop
```