Revisión	Fecha	Comentario	Autor
0	11/12/2008		Ulises Bigliati

ÍNDICE

Introducción **2**

Objetivos **2**

Implementación **2**

Descripción del hardware **2**
Selección del pin-out del microprocesador 2
Circuito eléctrico 3

Descripción del software involucrado **3**
Introducción 4
Estructura de archivos del proyecto 4
Archivos del proyecto 4
Archivos fuente del proyecto 4
Módulos de software del proyecto 5
Módulo keyboard.lib 5
Módulo principal 6

Introducción

Conectar físicamente un teclado matricial a un microprocesador o microcontrolador y realizar el código para su lectura es una tarea bastante sencilla, sin embargo, si alguien lo ha hecho antes y nos facilita el código, esto se vuelve más sencillo aún, y nos reserva más tiempo para destinarle a nuestra aplicación.
Siendo que desde la óptica del programa principal se observa al teclado como un elemento accesorio y secundario, para el desarrollador es normalmente deseable no destinar demasiado tiempo para el diseño, implementación y prueba de este periférico.

Objetivos

Nos proponemos realizar en forma modular, la implementación de la conexión y lectura de un teclado matricial estándar de 4 filas x 4 columnas.
Para esto construiremos un módulo de software que podrá ser reutilizado en cualquier aplicación de forma muy simple.
En forma complementaria al manejo del teclado, agregaremos al presente ejemplo una señalización sonora emitida por un buzzer, que indica cuando una tecla es presionada.

Implementación

En esta nota de aplicación utilizamos un módulo Rabbit RCM4300 en conjunto con un teclado matricial estándar de membrana[1] de 4 filas x 4 columnas.

La implementación de este ejemplo se llevó a cabo, aprovechando la placa de prototipos que está presente en cualquier Kit de desarrollo de Rabbit. La placa de prototipos nos provee los 3.3Volts para la alimentación del módulo RCM4300 y nos provee también de 5 Volts que aprovechamos para alimentar el buzzer.

En cuanto al software de demostración desarrollado para realizar esta aplicación práctica se utilizó el entorno de desarrollo Dynamic C en su versión 10.40. El software utilizado se describirá en la sección correspondiente en esta misma nota.

Nota:
Si bien en este ejemplo se utilizó un módulo RCM4300, el ejemplo es perfectamente aplicable a cualquier otro módulo Rabbit, y conceptualmente también lo es para cualquier otro micro,

Descripción del hardware

Selección del pin-out del microprocesador

A continuación comenzaremos a definir las entradas y salidas que en nuestro microprocesador cumplirán con las funciones necesarias para controlar y leer el teclado.

Complementariamente, también vamos a seleccionar un pin del microprocesador destinado a la activación de un buzzer que podrá ser usado como estímulo sonoro como respuesta a la presión de una tecla. De esta forma habrá mayores posibilidades de que el usuario modere el instinto que lo impulsa a destrozar la membrana del teclado suponiendo que la presión ejercida sobre la tecla nunca es suficiente.

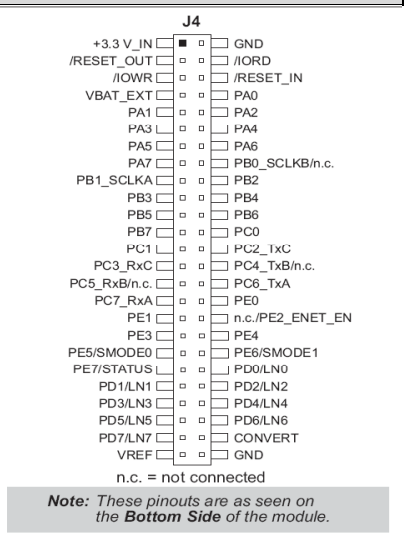
Utilizamos entonces el nibble más significativo del puerto B, configurando sus cuatro pines como entradas con la intención de conectarlos directamente a las filas del teclado.

Luego elegimos el nibble de menor peso del puerto C que debe ser configurado como salida en su totalidad, para así actuar también en forma directa sobre las columnas del teclado.

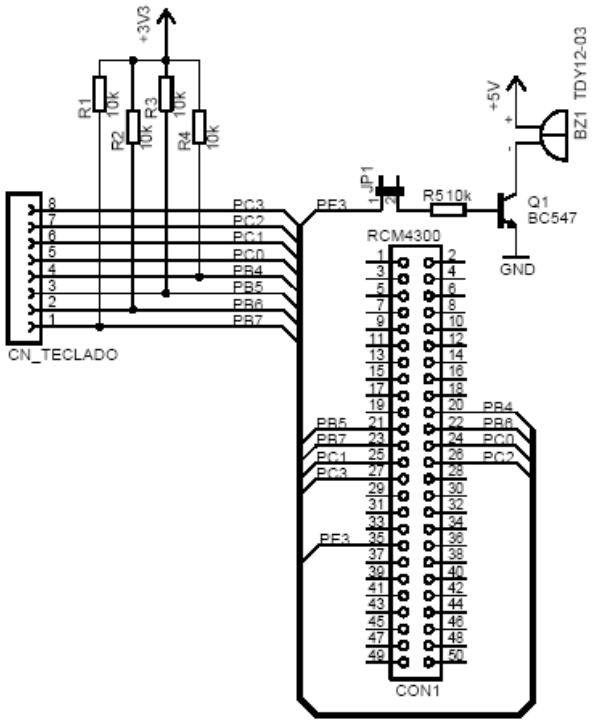
[1] El producto utilizado es fabricado por Microteclados

Pin-out				
Pin	Nombre	Sentido	Destino/función	
20	PB4	← In	Teclado	Pin 1 (fila 0)
21	PB5	← In		Pin 2 (fila 1)
22	PB6	← In		Pin 3 (fila 2)
23	PB7	← In		Pin 4 (fila 3)
24	PC0	Out →		Pin 5 (columna 0)
25	PC1	Out →		Pin 6 (columna 1)
26	PC2	Out →		Pin 7 (columna 2)
27	PC3	Out →		Pin 8 (columna 3)
35	PE3	Out →	Buzzer	

En la tabla podemos apreciar la designación de pines decretada para esta implementación, por supuesto estos pines podrán ser otros de acuerdo a las necesidades del proyecto real que pudiera implementar el presente ejemplo. A la derecha se presenta un vista desde abajo del pin-out del conector del módulo Rabbit RCM4300 que utilizamos para el ejemplo.



Circuito eléctrico



Una vez definidos los pines que serán destinados para controlar y leer el teclado, y en forma accesoria, también el pin que actuará sobre el buzzer nos dedicaremos al diseño del circuito eléctrico.
Como puede apreciarse en el esquemático, el teclado se conecta en forma directa a los pines del microprocesador según hemos definido más arriba. Colocamos resistencias de pull-up en los pines que funcionan como entrada para leer el estado de los contactos de la matriz (PB[4:7]).
Por otra parte, vemos el circuito del buzzer (que pose oscilador incorporado) que es conmutado mediante un transistor de uso general, el jumper JP1 está puesto por si se desea eliminar manualmente el sonido del buzzer. Y esto es todo en cuanto a la circuitería eléctrica

Nota 1:
Cabe destacar que el buzzer está alimentado con 5Volts, mientras que las resistencias de pull-up están conectadas a un terminal de 3.3Volts.

Nota 2:
Se omitieron los contactos de alimentación correspondientes al pin-out del módulo Rabbit, ya que se asume que la realización de está aplicación de ejemplo fue realizada sobre la placa de prototipos incluida en el kit de desarrollo del RCM4300.

Descripción del software involucrado

Introducción

Para utilizar el teclado mediante el circuito que acabamos de exponer escribimos un sencillo programa de demostración en DC10.40 que muestra a través del standard output cual fue la tecla presionada, y si así lo definimos, también se oirá el beep correspondiente de parte del buzzer.

Se trabajó con el concepto de modularidad en mente, ya que se estructuró el código generando una library que proporciona un grado de portabilidad importante para facilitar así la tarea del desarrollador que desee implementar este ejemplo en sus propios proyectos con mínimas o ninguna modificación.

Como resultado de lo dicho en el párrafo precedente se generó un módulo de software con una interfaz de usuario muy simple compuesta únicamente por tres funciones que se enumeran a continuación:

```
1. void keyb_init();
2. char keyb_get_key();
3. char keyb_get_asckey(char keycode);
```

Estructura de archivos del proyecto

Físicamente este mini-proyecto está organizado según puede apreciarse en el siguiente esquema de árbol. Esta es una organización típica para cualquier proyecto en DC, sin embargo no deja de ser una cuestión de preferencia personal en cuanto a la forma de organización de los archivos de un proyecto.

```
myProyects___CoAN-006
                |___CoAN-006.c
                |___CoAN-006.dcp
                |___lib.dir
                |___myLibs
                        |___keyboard.lib
```

A continuación se explicará la naturaleza y funciones específicas de cada uno de los elementos del proyecto, para una mejor comprensión se separarán estos elementos en dos categorías:

Archivos del proyecto

- **myProyects:**
  Directorio de proyectos dentro del directorio de instalación del DC. Este es un directorio creado por el usuario (por el autor de esta nota) y podrá tomar cualquier nombre, sin embargo sería conveniente mantener el nombre provisto a fines de agilizar las pruebas de quien desee reproducir este proyecto en su computadora.
- **CoAN-006:**
  Directorio particular del proyecto actual.
- **myLibs:**
  Directorio conteniendo las libreries particulares creadas para este proyecto.
- **lib.dir:**
  Archivo de referencia de libraries para este proyecto.
- **CoAN-006.dcp**:
  Archivo de proyecto del DC. Este archivo mantiene los datos de configuración particulares para este proyecto (archivo principal, lib.dir, puerto de comunicación, archivo lib.dir, etc, etc).

Archivos fuente del proyecto

- **keyboard.lib:**
  Contiene las funciones de inicialización y control del teclado matricial.
- **CoAN-006.c**:
  Archivo de programa para prueba de la library.

Módulos de software del proyecto

Módulo keyboard.lib

Como ya se ha mencionado, esta es la library que implementa el código de inicialización y control del teclado. La interfaz que presenta esta library para el desarrollador se puede dividir en cuatro secciones:

A) Sección de definiciones de puertos:

Se encuentran en forma de directivas #define bajo las líneas de comentario:

```
//*****************
//*Port definition
//*****************
//COLs
      [ directivas #define para los registros de puerto utilizados
                    para las columnas]
//ROWs
      [ directivas #define para los registros de puerto utilizados
                    para las filas]
//*****************
//*Columns of array
//*****************
      [ directivas #define para la definicion de pines utilizados
                    para las columnas]
//*****************
//*Rows of array
//*****************
      [ directivas #define para la definicion de pines utilizados
                    para las filas]
```

Por supuesto, si nada se modifica en esta sección, el código original estará actuando de acuerdo a las configuraciones de hardware que describiéramos en el apartado correspondiente (Selección del pin-out...).

B) Función de inicialización:

Desde el programa principal, se deberá llamar a esta función en el inicio, lo cual provocará la configuración de los puertos para poder controlar y leer el teclado posteriormente:

```
void keyb_init();
```

C) Función de control:

Las siguiente función constituye la utilidad principal del módulo, ella realiza el acceso a los puertos que controlan y leen el teclado matricial, al ser llamada retorna el código “crudo” de la tecla presionada, o en su defecto, si no detectó ninguna tecla presionada, devuelve cero.

```
char keyb_get_key();
```

D)Función de utilidad:

Por último, incluimos en la library, una función adicional que podría categorizarse como una función de utilidad, ya que lo único que realiza es la traducción del código “en crudo” devuelto por la función anterior, al código ASCII correspondiente, según el teclado que estemos utilizando, la declaración de esta función es la que sigue:

```
char keyb_get_asckey(char keycode);
```

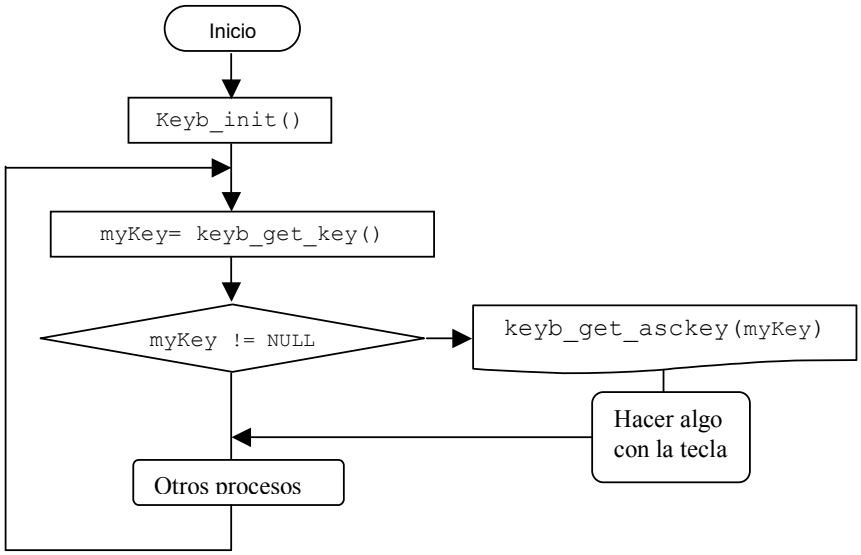
Y la definición de dicha función, es simplemente un switch, que reemplaza el código crudo por el ASCII que corresponda según nuestro teclado, como para nuestro ejemplo debemos responder a la siguiente disposición de teclas:

1	2	3	A

4	5	6	B
7	8	9	C
#	0	*	D

Módulo principal

Para finalizar esta nota mostraremos el contenido del archivo principal de programa (CoAN-006.c) que invoca a la library keyboard.lib. Como verán el método de utilización de la interfaz del módulo es muy simple, y responde básicamente al siguiente esquema:



Obviando algunas líneas del programa que no es necesario exponer, el código que nos interesa es el que se despliega y comentamos a continuación.

Las primeras dos directivas #define definen que el pin 3 del puerto PE, será una salida, (es la que se utilizará para el buzzer), esto queda declarado como una macro de inicialización.
Las dos restantes definen macros para activar y apagar cómodamente el buzzer.

```
//Buzzer defines
#define BUZZER 3
#define BUZZERINIT      BitWrPortI ( PEDDR,&PEDDRShadow,1,BUZZER )
#define BUZZERON        BitWrPortI ( PEDR, &PEDRShadow, 1,BUZZER )
#define BUZZEROFF       BitWrPortI ( PEDR, &PEDRShadow, 0,BUZZER )
```

La que sigue es una función de prueba con la finalidad de hacer algo con el carácter recibido. Simplemente lo muestra por el standard output.

```
//Test function to receive and show the pressed key
void showLastKey( char key ){
      printf("%c", key);
}
```

Comienza el programa y a continuación de las declaraciones de variables se invoca a la función de inicialización para el teclado. Seguidamente se configura el puerto de control del buzzer y se lo deja apagado.

```
main()
{
   unsigned char cKey;
   int result;

   //Keyboard initialization;
      keyb_init();
   //Buzzer start-up
   BUZZERINIT;
   BUZZEROFF;

      while(1)
      {/*****************************
        * Keyboard control task
        *****************************/
```

La pieza principal de este código reside dentro del costate siguiente, mediante el "`if`" nos enteramos si hay una tecla presionada o no, y en caso negativo, abortamos sin más trámites el costate para atender otras cuestiones.

Si el hilo de ejecución evitó el "`abort`" es porque detectamos una tecla pulsada y la tenemos disponible en la variable `cKey`, la cual le pasamos a la función traductora "`keyb_get_asckey()`" para que nos devuelva un código un poco mas humanamente interpretable, en la misma línea de programa le pasamos el código ASCII recibido a la función de prueba showLastKey(), quien lo imprime en el stdout del Dynamic C.
Para terminar, generamos un beep con la macro `BUZZERON`, agregamos el importante retardo anti-rebotes para evitar la detección de falsas repeticiones de teclas. Y finalmente silenciamos el beep del buzzer.

```
            costate
            {     // Look for key pressed
                  if (  !( cKey = keyb_get_key() ) )
                        abort;
                  //Send last key pressed(o NULL if no one there)
                  //to handler function
                  showLastKey( keyb_get_asckey( cKey ) );
                     BUZZERON;
                     waitfor(DelayMs(200));//prevent bounces
                     BUZZEROFF;
            }
      }
}
```