Cika Electrónica

Nota de Aplicación: CAN-014
Título: **Utilización de Touch Screens con Dynamic C 8**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	23/10/03	

Analizamos las bibliotecas de funciones con soporte para pantallas sensibles al tacto o touch screens, que incluye Dynamic C en su versión 8, y la forma de aprovechar la mayor cantidad posible de este código para acortar nuestros tiempos de desarrollo.

A partir de la versión 8, Dynamic C provee como standard muchas de las prestaciones que antes estaban reservadas para la distribución Premier. Entre estas encontramos muchas bibliotecas de funciones desarrolladas como demo de las placas de Z-World; pero fundamentalmente queremos destacar un set de bibliotecas de funciones orientadas a la utilización de touch screen.

No vamos a detallar cada una de las funciones que cada biblioteca incluye, pero sí haremos una somera enumeración, antes de analizar la forma de desarrollar drivers para el controlador que vamos a utilizar.

Bibliotecas de funciones

GLTOUCHSCREEN.LIB

Esta biblioteca provee funciones para operar con botones sobre una pantalla sensible al tacto (touch screen). Los botones pueden contener un texto o un gráfico en su interior, que los identifica, y pueden agruparse en niveles, de modo de poder ser desplegados de a grupos. Cada botón posee sus propias características. Esta biblioteca utiliza funciones de bajo nivel provistas por otra biblioteca de funciones, para la lectura de la pantalla en sí. Incorpora además soporte de buzzer, emitiendo un breve sonido cuando un botón es presionado, si se habilita esta opción. La rutina de control del buzzer en sí deberá ser provista por el usuario, bajo el nombre de *buzzer()*.

El soporte gráfico está provisto por *GRAPHIC.LIB*, cuya integración desarrollamos en la CAN-013. También ha sido desarrollada para OP7200, pero resulta un excelente esqueleto para nuestras aplicaciones.

Entre las características principales de esta biblioteca de funciones encontramos:

- ✔ Armado e inicialización de las estructuras que forman y definen a los botones: *btnCreatebitmap(), btnCreateText(), btnAttributes()*
- ✔ Despliegue de los botones en pantalla: *btnDisplay(), btnClear(), btnDisplayLevel(), btnClearLevel()*
- ✔ Detección de la presión sobre un botón: *btnGet()*

TS_R4096.LIB

Se trata de una colección de funciones de bajo nivel para lectura de pantallas sensibles al tacto. Fue desarrollada para el hardware del OP7200, pero puede servir como esqueleto para desarrollar el soporte para el hardware que utilicemos.

Las funciones principales de esta biblioteca, accedidas externamente son:

- ✔ Generar la matriz de calibración de la touchscreen: *TSCalib()*
- ✔ Salvar y recuperar la matriz de calibración en memoria no volátil: *TsCalibEEWr(), TsCalibEERd()*
- ✔ Leer la coordenada actual de la touchscreen: *TsXYvector()*
- ✔ Leer el estado actual de la touchscreen: *TsActive()*
- ✔ Procesar el estado de la touchscreen, obteniendo coordenada actual o indicación de que no hay presión sobre la touchscreen: *TsScanState()*. Esta función también maneja el anti-rebote (debouncing) mediante una máquina de estados finitos (finite state machine)
- ✔ Obtener el resultado de la ejecución de *TsScanState()* anterior: *TsXYBuffer()*

Las funciones de más bajo nivel son:

➔ *_adcTouchScreen()*: es la que realiza la lectura del chip controlador de la touchscreen (ADS7843)

➔ *TsActive()*: es la que monitorea la presión sobre la touchscreen, aprovechando el pin de interrupción del chip controlador.

Desarrollo de drivers

Para poder utilizar las funciones de alto nivel de *GLTOUCHSCREEN.LIB* que nos interesan, deberemos proveer las funciones de bajo nivel acorde al hardware que estemos utilizando. En general, deberemos portar las funciones de driver del controlador de touchscreen. Como comentáramos anteriormente, poseemos un excelente ejemplo: *TS_R4096.LIB*. La mejor opción es copiar esta biblioteca y analizar sus funciones para luego portar las partes donde nuestro hardware difiere. No deje de consultar las notas de aplicación, es posible que haya una para su controlador específico.

Funciones principales a portar de *TS_R4096.LIB*:

➔ *_adcTouchScreen()*: acorde a la forma de lectura de la touchscreen que utilicemos

➔ *TsActive()*: ídem

Adicionalmente, deberemos proveer una función que inicialice todo el hardware, *brdInit()*, la cual seteará determinados flags que serán chequeados por las otras bibliotecas, como el ejemplo. No olvidemos que debemos proveer la función *buzzer()*

Ejemplo de *brdInit()*:

```
nodebug
void brdInit (void)
{
        #GLOBAL_INIT {__brdInitFlag   = FALSE;}
// Insertar código de inicialización aquí
        __brdInitFlag = TRUE;
        __InitTSpower  = 0;
// Lee calibración del user block
        TsCalibEERd();
// Inicializa el controlador de la touchscreen
        _adcTouchScreen(0xD0);
        _adcTouchScreen(0x90);
        __anaPWRDelay = MS_TIMER + 250;
        __InitTSpower  = 1;
}
```

Ejemplos

En el siguiente ejemplo, deberemos reemplazar *TS_R4096.lib* por nuestra biblioteca correspondiente, y proveer una función *brdInit()* para inicializar nuestro hardware y leer las constantes de calibración de la touchscreen del user block.:

```
#memmap xmem
#use "sed1335F.lib"
#use "graphic.lib"
#use "glmenu.lib"
#use "gltouchscreen.lib"
#use "8x10l.lib"

#use "TS_R4096.lib"

fontInfo fi8x10;
windowFrame bodywin;
unsigned  long userX;         // Puntero a las estructuras de botones

main()
{
int btn;
```

```
        brdInit();                                  // Inicializa hardware
        glInit();                                   // Initializa driver gráfico
        glXFontInit(&fi8x10, 8, 10, 32, 127, Font8x10); // Define tipo de letra
        glBlankScreen();                            // Borra pantalla
        userX = btnInit( 3 );         // Reserva espacio para estructuras de botones
        btnCreateText(userX,1,50,100,80,50,0,1,&fi8x10,"Boton 1"); // crea botón cuadrado
        btnCreateText(userX,2,150,100,80,50,1,1,&fi8x10,"Boton 2"); // crea botón redondeado
        btnAttributes(userX,1,0,0,0,0);             // cambia atributos del botón 1
        while (!btnDisplayLevel(userX,1)); // Muestra todos los botones de nivel 1
        while(1){
                costate {                           // waitfor debe ir dentro de un costate
                        waitfor ( (    btn = btnGet(userX) ) >= 0 ); // espera botón apretado
                        switch (btn){
                                case 1:
                                        btnClear(userX,1);          // borra botón 1
                                        btnDisplay(userX,2);        // muestra botón 2
                                        break;
                                case 2:
                                        btnClear(userX,2);          // borra botón 2
                                        btnDisplay(userX,1);        // muestra botón 1
                                        break;
                        }
                }
        }
}
```

Para que la biblioteca compile correctamente, deberemos proveer una función buzzer(), aunque no es necesario que realice alguna función en particular.

```
void buzzer()
{
}
```

Para que este ejemplo se ejecute correctamente, deberemos primero calibrar la touchscreen, y guardar las constantes en el user block. La biblioteca *TS_R4096.LIB*, junto con *OP7200.LIB*, que controla e inicializa el hardware del OP7200, son un buen ejemplo de como inicializar el hardware y acceder al user block.

El user block es un espacio de flash reservado para que el usuario almacene particularmente constantes de calibración relativas al hardware en cuestión. Accedemos al mismo mediante dos funciones:

```
tc_cal _adcCalibTS;

        status = readUserBlock(&_adcCalibTS, CALIB_TS, sizeof(_adcCalibTS));
        status = writeUserBlock(CALIB_TS, &_adcCalibTS, sizeof(_adcCalibTS));
```

donde _adcCalibTS es la estructura que contiene las constantes a guardar y CALIB_TS es el offset dentro del user block.

Si recibimos algún error respecto a la existencia o estructura del user block, podemos generarlo mediante la utilidad *CHANGE_USERBLOCK.C*, que se adjunta. El offset dentro del user block para guardar nuestras constantes puede definirse de la siguiente forma:

```
#define CALIB_TS    (4096*GetIDBlockSize()-0x400)
```

La calibración de la touch screen se puede realizar con la utilidad *CAL_TOUCHSCREEN.C*, que viene incluída en la distribución de Dynamic C versión 8, en el directorio:
<RABBIT_PATH>\Samples\OP7200\LCD_TouchScreen.

Finalmente, no deje de chequear las notas de aplicación, puede que hayamos desarrollado una para el hardware que usted utiliza.