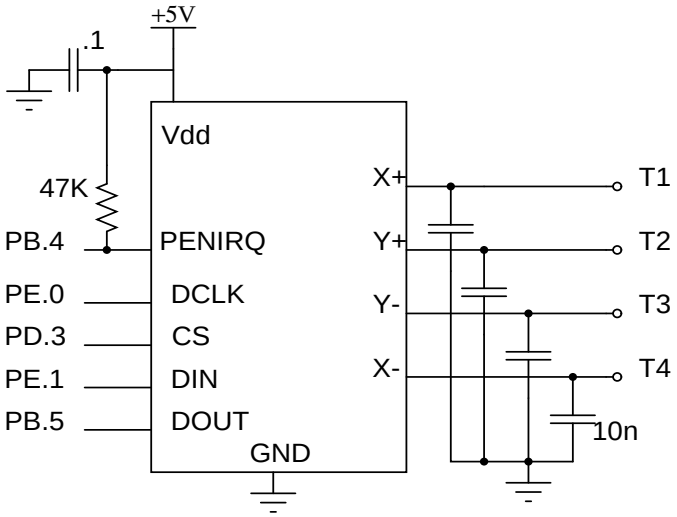
Cika Electrónica

Nota de Aplicación: CAN-021
Título: **Display PG320240FRST con ADS7846 y Dynamic C 8**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	12/12/03	

En la CAN-016, portamos las bibliotecas de funciones con soporte para displays gráficos y pantallas sensibles al tacto o touch screens, que incluye Dynamic C en su versión 8, para su utilización con el display PG320240FRST de Powertip. Aprovechamos el hardware de lectura de touch screen desarrollado en la CAN-020, y agregamos una opción a nuestra propia biblioteca de funciones: *Cika320240FRST.lib*, donde alojaremos las funciones para aprovechar el controlador ADS7846.

Hardware



El hardware es el desarrollado en la CAN-020, simplemente asignamos los pines de la interfaz serie a pines del Rabbit, e implementaremos la interfaz por software.

Desarrollo del driver

Como vimos en la CAN-014, debemos portar las funciones de más bajo nivel:
➔ *_adcTouchScreen()*: es la que realiza la lectura del chip controlador de la touchscreen
➔ *TsActive()*: es la que monitorea la presión sobre la touchscreen.

_adcTouchScreen()

Esta función es la encargada de leer el chip controlador, devolviendo un valor entre 0 y 4095. Dado que el ADS7846 es un conversor de 12-bits, muy similar al ADS7843, podemos aprovechar gran parte del software original que Rabbit desarrollara para el OP7200, salvando las leves diferencias de hardware:

```
int _adcTouchScreen( int cmd )
{
#asm
        ;--------------------------------------------------------------------
        ;       Send CMD to the touchscreen ADC to get the x,y coordinate's.
        ;--------------------------------------------------------------------
        ld    de, PDDR
```

```
        ld    a, (PDDRShadow)
        res     3,a                         ;Assert Chip Select
        ioi     ld (de),a                   ;
        ld    (PDDRShadow),a                ;Update shadow register

        push  ip                            ; save off IP state
        ipset   1                           ; set interrupt priority to level 1
        ld              b,8
        ld    l,(ix+cmd)
.cmdcontinue:
        ld    de, PEDR
        ld    a, (PEDRShadow)
        res     0,a                         ;set CLK bit low
        ioi     ld (de),a                   ;CLK SDI data bit into AD
        ld    (PEDRShadow),a

        ld              a,l
        and   a,0x80
        cp    a,0x80
        jr              nz,.cmdbitlow

.cmdbithigh:
        ld    a,(PEDRShadow)
        set   1,a
        ioi   ld (de),a
        ld    (PEDRShadow),a
        jr    .cmdCLK

.cmdbitlow:
        ld    a,(PEDRShadow)
        res   1,a
        ioi   ld (de),a
        ld    (PEDRShadow),a

.cmdCLK:
        ld    de, PEDR
        ld    a, (PEDRShadow)
        set     0,a                         ;set CLK bit High
        ioi     ld (de),a                   ;CLK SDI data bit into AD
        ld    (PEDRShadow),a
        sla   l
        djnz    .cmdcontinue
        pop     ip
        WAIT_50usX(1);


        ;---------------------------------------------------------------------
        ;       Get the x,y coordinate's from the touchscreen ADC.
        ;---------------------------------------------------------------------
        ld              de,PEDR
        ld    a,(PEDRShadow)
        res     1,a
        ioi     ld (de),a
        ld              b,12
        ld              iy,0x0000
        push  ip                            ; save off IP state
        ipset   1                           ; set interrupt priority to level 1

.rd_touchscreen:
        ld    de, PEDR
        ld    a, (PEDRShadow)
        res     0,a                         ;set CLK bit low
        ioi     ld (de),a                   ;CLK SDI data bit into AD
        ld    (PEDRShadow),a
        ld    de, PBDR
        ioi   ld a,(de)
```

```
        and   a,0x20
        rra
        rra
        rra
        rra
        rra
        ld    d,0
        ld    e,a
        ld    hl,iy
        or            hl,de
        ex            de,hl
        rl            de
        ex            de,hl
        ld    iy,hl
        ld    de,PEDR
        ld    a,(PEDRShadow)
        set     0,a                         ;set CLK bit low
        ioi     ld (de),a                   ;CLK SDI data bit into AD
        ld    (PEDRShadow),a
        djnz    .rd_touchscreen

        ;---------------------------------------------------------------------
        ;       Un-assert chip select and preset the CLK line low, for the next
        ;  conversion cycle.
        ;---------------------------------------------------------------------
        ld    de, PDDR
        ld    a, (PDDRShadow)
        set   3,a
        ioi     ld (de),a                   ;Un-assert Chip Select on ADC
        ld    (PDDRShadow),a                ;Update shadow register
        ld    de, PEDR
        ld    a, (PEDRShadow)
        res     0,a                         ;set CLK bit low
        ioi     ld (de),a                   ;CLK SDI data bit into AD
        ld    (PEDRShadow),a
        ld            hl,iy
        pop     ip
#endasm
}
```

La macro *WAIT_50usX()* que utilizamos pertenece al cuerpo de la biblioteca de funciones. En nuestro desarrollo utilizamos un loop, el usuario puede utilizar lo que más le convenga.

TsActive()

Esta función es la encargada de detectar la presencia de presión sobre la pantalla, entregando '0' mientras no hay presión y un valor diferente cuando la hay. Con un chip controlador como el ADS7846, esto es una tarea fácil, dado que dispone de un pin destinado específicamente a esta tarea. Simplemente deberemos invertir la señal *PENIRQ* por software, ya que esta señal toma el valor '0' al detectar presión.

```
int TsActive( void )
{
#asm
        ld   de,PBDR
        ioi  ld a,(de)
        ld   l,0x10
        and  l
        xor  l
        ld   l,a
        ld   h,0x00
#endasm
}
```

Como tenemos pensado hacer crecer esta biblioteca en el futuro, agregando diversas opciones de hardware de lectura de touch screen, utilizamos ensamblado condicional en varias partes del código. No lo incluímos en el ejemplo por claridad.

Inicialización

Debemos configurar los ports utilizados y mantener el CS en estado inactivo (1 lógico), para lo cual no necesitamos realizar modificaciones sobre el código original de la biblioteca.

Calibración

A fin de poder seguir re-utilizando la mayor cantidad posible de código y ejemplos, conservamos el esquema de *TS_R4096.LIB* para guardar y recuperar las constantes de calibración en el user block, descripto en la CAN-014, y tal como hiciéramos en la CAN-016.

El esquema de calibración utiliza la teoría desarrollada en la CAN-015 y CAN-020, con las constantes en una estructura:

```
typedef struct
{
	int x_offset;
	int y_offset;
	float x_gainfactor;
	float y_gainfactor;
} tc_cal;
```

1- Obtención de las constantes de calibración asociando dos puntos en pantalla del display $(X_1; Y_1)$ y $(X_2; Y_2)$, con las coordenadas devueltas por el sistema de touch screen $(x_1; y_1)$ y $(x_2; y_2)$, como se observa en *TsCalib()*:

```
	_adcCalibTS.x_offset = x1;
	_adcCalibTS.y_offset = y1;

	_adcCalibTS.x_gainfactor = (float)LCD_XS/(float)(x2-x1);
	_adcCalibTS.y_gainfactor = (float)LCD_YS/(float)(y2-y1);
```

2- Corrección de los datos al momento de usarlos, como se observa en TsXYvector():

```
		*xkey -= _adcCalibTS.x_offset;
		*ykey -= _adcCalibTS.y_offset;

		*xkey =  (int) (*xkey*_adcCalibTS.x_gainfactor);
		*ykey =  (int) (*ykey*_adcCalibTS.y_gainfactor);
```

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.

Una vez calibrada, cada vez que carguemos un nuevo ejemplo, la rutina de inicialización recuperará las constantes del user block, por lo que la pantalla seguirá funcionando. Si bien la calibración depende de las condiciones de temperatura, es lo suficientemente estable para aplicaciones de selección y para los alcances de esta nota de aplicación.

Si recibimos algún error respecto a la existencia o estructura del user block, podemos generarlo mediante la utilidad *CHANGE_USERBLOCK.C*, que se adjunta.

Ejemplos

Si bien la mayoría de los ejemplos que vienen para el OP7200 funcionan con menores modificaciones, hemos incluído el mismo utilizado en la CAN-016, mostrando gran parte de las posibilidades gráficas. Para que este y

otros ejemplos se ejecuten correctamente, deberemos primero calibrar la touchscreen, y guardar las constantes en el user block.

```
// Usamos el hardware de lectura de la touch screen basado en ADS7846
#define TSCONTROLLER  2
/* Esto incluye la biblioteca de funciones de Cika que soporta el display de 320x240 de
Powertip */
#use "Cika320240FRST.lib"

#memmap xmem

// íconos
#use "ledon_bmp.lib"
#use "ledoff_bmp.lib"
#use "button_bmp.lib"

int led1,led2;
fontInfo fi8x10,fi10x12,fi6x8;

void led1show(int state)
{
       if (state)
              glXPutBitmap(8, 8, 90,90, ledon_bmp);
       else
              glXPutBitmap(8, 8, 90,90, ledoff_bmp);
}

void led2show(int state)
{
       if (state)
              glXPutBitmap(208, 8, 90,90, ledon_bmp);
       else
              glXPutBitmap(208, 8, 90,90, ledoff_bmp);
}


unsigned  long userX;         // Área de botones

void main()
{
int btn;

       brdInit();             // Inicializa hardware
       glInit();              // Inicializa biblioteca de funciones gráficas

       glXFontInit ( &fi6x8,6,8,0x20,0x7E,Font6x8 );       // Inicializa tipografía

       glBlankScreen();                                    // borra pantalla
       userX = btnInit( 3 );
       btnCreateBitmap(userX,1,0,110,0,1,button_bmp,90,90);       // define botones
       btnCreateBitmap(userX,2,200,110,0,1,button_bmp,90,90);
       btnAttributes(userX,1,0,0,0,0);
       btnAttributes(userX,2,0,0,0,0);

       led1=1;
       led2=0;

       while (!btnDisplayLevel(userX,1)); // Muestra todos los botones de nivel 1
       led1show(led1==1);
       led2show(led2==1);
       glPrintf(4,230,&fi6x8,"Demo Cika320240FRST.lib, Cika Electronica S.R.L.");

          while (1) {

               costate {                    // waitfor debe ir dentro de un costate
                      waitfor ( ( btn = btnGet(userX) ) >= 0 );
```

```
                        switch (btn){
                                case 1:
                                   led1^=1;
                                   led1show(led1);
                                        break;
                                case 2:
                                   led2^=1;
                                   led2show(led2);
                                        break;
                        }
                }
           }
}
```

Las bibliotecas *led_on.lib*, *led_off.lib* y *button.lib* fueron generadas con la utilidad fbmcnvtr, provista con Dynamic C. Convierte fonts y bitmaps a formato library. Lo mismo que con *Cika320240FRST.lib*, deben estar listadas en el archivo índice de biblioteas de funciones (*LIB.DIR* o equivalente)

Para que algunos ejemplos compilen correctamente, deberemos proveer una función buzzer(), aunque no es necesario que realice alguna función en particular. Esto se debe a que las funciones de la biblioteca GLTOUCHSCREEN.LIB incluyen soporte para “keyclick”.

```
int buzzer()
{
}
```