Cika Electrónica

Nota de Aplicación: CAN-013
Título: **Utilización de displays gráficos 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 displays gráficos 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 displays gráficos.

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 del display que vamos a utilizar.

Bibliotecas de funciones

GRAPHIC.LIB

Esta biblioteca reserva un buffer en *xmem* y realiza todas las operaciones gráficas sobre el mismo, actualizando luego el display, copiando el buffer en la memoria de éste. La actualización de la memoria del display puede demorarse hasta terminar una operación mediante la utilización conjunta de las funciones *glBuffLock()* y *glBuffUnlock()*, al principio y final de la operación en cuestión, respectivamente. Las funciones provistas pueden utilizarse sin modificaciones a la biblioteca. Si bien muchas de las funciones no son re-entrantes, su velocidad es interesante.

Entre las características principales de esta biblioteca de funciones encontramos:

- ✔ Borrado de pantalla o parte de ella: *glBlankScreen(), glBlankRegion()*
- ✔ Pintado de pantalla o parte de ella: *glFillScreen(), glBlock()*
- ✔ Scroll vertical u horizontal: *glVScroll(), glHScroll()*
- ✔ Dibujo de puntos en pantalla (plot): *glPlotDot()*
- ✔ Trazado de líneas: *glPlotLine()*
- ✔ Trazado de polígonos y círculos: *glPlotPolygon(), glPlotVPolygon(), glPlotCircle()*
- ✔ Pintado de polígonos y círculos: *glFillPolygon(), glFillVPolygon(), glFillCircle()*
- ✔ Armado de ventanas de texto: *TextWindowInit(), TextBorderInit(),TextBorder()*
- ✔ Manejo de diversos sets de caracteres: *glXFontInit()*
- ✔ Impresión de textos en el display: *glPrintf(), TextPrintf()*
- ✔ Despliegue de imágenes en pantalla: *glPutBitmap()*

Para utilizar esta biblioteca, desde un punto de vista de alto nivel, necesitamos inicializarla, lo cual se realiza llamando a la función *glInit()*. En todos los casos, el modo de dibujo puede elegirse mediante *glBrushType()* para setear, resetear, o invertir los pixels.

KEYPAD

Existen varias bibliotecas de soporte de keypads, las mismas fueron desarrolladas para distintos equipos de Z-World. Pueden utilizarse como esqueleto para desarrollar nuestro propio sistema. Esta biblioteca es la encargada de proveer las funciones *KeyGet(), KeyProcess()* y *KeyInit()*, que son utilizadas por *GLMENU.LIB* para la operación de menúes de selección.

GLMENU.LIB

Esta biblioteca provee funciones de despliegue de menúes en pantalla, utilizando funciones provistas por *GRAPHIC.LIB* para escribir en el display y funciones de *KEYPAD* para leer teclados. Podemos utilizarla como un esqueleto para nuestra propia biblioteca de funciones o aprovecharla en su totalidad.

SED1335F.LIB

Contiene las funciones de bajo nivel para el controlador de displays LCD SED1335. Provee las funciones básicas de inicialización, transferencia del buffer a la memoria del display, y escritura directa sobre la memoria del controlador, más algunas otras funciones de más alto nivel que son utilizadas por *GRAPHIC.LIB*. Es un excelente esqueleto para desarrollar soporte para otros controladores.

Las funciones principales de esta biblioteca son mayormente de bajo nivel, las accedidas externamente son:

- ✔ *_glSwapData()*: se encarga de copiar el buffer de SRAM a la memoria del display (bajo nivel)
- ✔ *_glInit()*: inicializa el display y su hardware asociado
- ✔ *_glPlotRealtime()*: dibuja directamente sobre el hardware del display (bajo nivel)
- ✔ *glRealtime()*: selecciona el modo de dibujo (copia de buffer o dibujo en tiempo real)
- ✔ *glBackLight()*: controla la operación del backlight
- ✔ *glSetContrast()*: controla el contraste del display
- ✔ *glDispOnOff()*: controla la aparición de la imagen en el display

Las funciones de uso interno y más bajo nivel son:

- ➔ *_glInitLCDChip()*: inicializa el controlador del display LCD
- ➔ *_glInitStrobe()*: inicializa el hardware asociado al controlador (ports del Rabbit)

Además, este driver en particular presenta un modo de animación, en el cual mediante una variable controla si la rutina de copia del buffer al display espera el sincronismo del display o no, a fin de minimizar el parpadeo (flicker) ante múltiples escrituras reiteradas, dada la condición particular de operación del SED1335.

Desarrollo de drivers

Para poder utilizar las funciones de alto nivel de las bibliotecas que más nos interesan: *GRAPHIC.LIB* y *GLMENU.LIB*, deberemos proveer las funciones de bajo nivel acorde al hardware que estemos utilizando. En general, deberemos portar las funciones de driver de display LCD y proveer funciones de lectura y procesamiento del teclado que utilicemos. Como comentáramos anteriormente, poseemos un excelente ejemplo: *SED1335F.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 *SED1335F.LIB*:

- ➔ *_glSwapData()*: mapeando la estructura lineal del buffer en SRAM al modo de trabajo del controlador que utilicemos
- ➔ *_glPlotRealtime()*: ídem, aunque esta función en particular puede omitirse, definiéndola como una función vacía y evitando modificar la variable global *realtime*.
- ➔ *_glInit()*: para inicializar el hardware que utilicemos. Podemos, si así lo queremos, utilizar un esquema similar y aprovechar parte de las rutinas *_glInitLCDChip()* y *_glInitStrobe()*.

Ejemplo de utilización de *GRAPHIC.LIB*

En el siguiente ejemplo, deberemos reemplazar *sed1335F.lib* por nuestra biblioteca correspondiente:

```
#memmap xmem
#use "sed1335F.lib"
#use "graphic.lib"
#use "glmenu.lib"
#use "6x8l.lib"

fontInfo fi6x8;
```

```
windowFrame bodywin;

main()
{
int x;
const static char Texto[]={"Esto es un texto cualquiera"};

        glInit();                                   // Inicializa driver gráfico
        glXFontInit(&fi6x8, 6, 8, 32, 127, Font6x8); // Define tipo de letra
        glBlankScreen();                            // Borra pantalla
        for(x=100;x<300;x++)
                glPlotLine(x,100,100,200);          // traza líneas
        glBlankScreen();                            // Borra pantalla
        glBuffLock();                               // Detiene actualización de pantalla
        for(x=100;x<300;x++)
                glPlotLine(x,100,100,200);          // traza líneas
        glBuffUnlock();                             // Reanuda actualización de pantalla
        glVScroll(0,0,320,240,-80);                 // Mueve pantalla hacia arriba 80 líneas
        glBlankScreen();                            // Borra pantalla
        glPrintf(0,0,&fi6x8,Texto);                 // Imprime texto
        glSetBrushType(PIXXOR);                     // Invierte pixels en un bloque
        glBlock(0,0,300,20);
        glSetBrushType(PIXBLACK);                   // Dibuja en negro
        TextWindowFrame(&bodywin,&fi6x8,100,50,100,80); // Define ventana de texto
        TextBorderInit(&bodywin,SINGLE_LINE,"Titulo"); // Define borde de ventana y título
        TextBorder(&bodywin);                       // Dibuja borde
        TextPrintf(&bodywin,Texto);                 // Acomoda texto dentro de la ventana
}
```