Cika Electrónica

Nota de Aplicación: CAN-041
Título: **Conversión de tipografías para LCD gráficos con HD61202**
Autor: Sergio R. Caprile, Senior Engineer

Revisiones	Fecha	Comentarios
0	16/09/05	

A pedido del público, publicamos el pequeño programita Q&D[1] desarrollado oportunamente para convertir tipografías "standard" al formato de memoria utilizado por los chips controladores compatibles con el HD61202, de Hitachi, y su clon: el KS0108, de Samsung; presentes en los módulos Powertip PG12864, de 128x64 pixels.

Hemos ya presentado la estructura de memoria de estos displays en la CAN-003, CAN-008 y CAN022, por respeto a la redundancia sugerimos a los lectores interesados remitirse a dichas notas.

Según comentáramos en la CAN-008, para generar los sets de caracteres tomamos tipografías de dominio público disponibles en Internet y las rotamos al formato de los controladores utilizados. Generalmente, los sets de caracteres se hallan definidos a un byte por línea horizontal, n líneas de arriba a abajo, caracter por caracter. La tarea a realizar consiste en transferir esa información, caracter por caracter, al formato del HD61202: un byte por línea vertical, n líneas de izquierda a derecha, caracter por caracter.

A continuación desarrollamos código en C para realizar la función correspondiente al algoritmo descripto, dejando totl y absolutamente de lado la eficiencia en favor de la claridad. Obtendremos la salida por standard output, de donde se puede incluir en nuestro código mediante copy/paste o redireccionando la salida a un archivo. En nuestro caso hemos utilizado como compilador *gcc* bajo Linux, sin embargo el código debería compilar sin mayores inconvenientes bajo cualquier otro compilador y plataforma, dado que no se han utilizado funciones no standard.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "font_pepito8x8.c"


#define CHARS 256

main()
{
int i,j,k;
unsigned char *font;

  if(!(font=(unsigned char *)calloc(8*CHARS,sizeof(char))))
    exit(0);
  for(i=0;i<CHARS;i++) {
    for(j=0;j<8;j++) {
      for(k=0;k<8;k++)
        font[8*i+j]|=(((font_pepito8x8[8*i+k]>>(7-j))&1)<<k);
    }
  }
  puts("unsigned char font8x8[]={");
  for(i=0;i<CHARS;i++) {
    for(j=0;j<8;j++)
      printf("0x%02X,",font[8*i+j]);
    printf("\t// 0x%0X '%c'\n",i,((i>0x1f)&&(i!=0x7F))? i:' ');
  }
  puts("};\n");
```

1 Quick & Dirty

```
  free(font);
}
```

Obsérvese como los datos de la tipografía están en el array *font_pepito8x8[]*, el mismo es de tipo *unsigned char* y debe ser declarado y tener los datos correspondientes dentro del archivo *font_pepito8x8.c*, que se incluye al principio del listado. Este método es suficientemente rápido y práctico, aunque debemos recompilar el programa cada vez que queremos convertir una tipografía (cosa que no ocurre muy a menudo) pero permite utilizar muchas tipografías distribuidas como "código en C" en la internet. Es necesario modificar el programa si la tipografía tiene un tamaño diferente, lo cual no resultará muy difícil dado que ya se tiene un esquema funcionando. La intención de esta nota es solamente mostrar un método rápido y simple para resolver la generación de tipografías para los displays basados en HD61202.

Si este método resulta un inconveniente, disponemos de otra versión con una interfaz para leer la tipografía directamente del disco y alojarla en un array, siempre y cuando disponga de la misma en formato compatible con este programa (8 bytes por caracter: 1 byte por línea horizontal, 8 líneas consecutivas), que *curiosamente* es el mismo empleado por computadoras de 8-bits como la Sinclair Spectrum.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>



main(int argc, char *argv[])
{
int i,j,k,CHARS,BASE;
unsigned char *font,*spectrum,*filename;
FILE *fin;

  if(argc!=4){
    printf("Usage: %s font maxchar# basechar#\n",argv[0]);
    exit(1);
  }
  else {
    filename=argv[1];
    CHARS=atoi(argv[2]);
    BASE=atoi(argv[3]);
  }
  if(!(font=(unsigned char *)calloc(8*CHARS,sizeof(char))))
    exit(0);
  if(!(spectrum=(unsigned char *)calloc(8*CHARS,sizeof(char))))
    exit(0);

  if(fin=fopen(filename,"r")) {
    fread(spectrum,1,768,fin);
    for(i=BASE;i<CHARS;i++) {
      for(j=0;j<8;j++) {
        for(k=0;k<8;k++)
          font[8*i+j]|=(((spectrum[8*(i-BASE)+k]>>(7-j))&1)<<k);
      }
    }
    fclose(fin);
  }
  else {
       printf("Can't open %s\n",filename);
  }

  printf("unsigned char font_%s[]={\n",filename);
  for(i=BASE;i<CHARS;i++) {
    for(j=0;j<8;j++) {
      printf("0x%02X,",font[8*i+j]);
    }
    printf("\t// 0x%0X '%c'\n",i,((i>0x1f)&&(i!=0x7F))? i:' ');
```

```
  }
  puts("};\n");
  free(font);
  free(spectrum);
}
```

Para llamar al programa anterior, necesitamos conocer el caracter más alto (generalmente 127) y el primero que aparece en la tipografía (generalmente 32). Llamamos entonces al programa como:

genfont8x8file fontfile 128 32

donde *128* es 1+ el caracter más alto y *32* el caracter más bajo contenidos dentro de la tipografía. Debido a que esto no es un programa comercial sino un *quick hack* para resolver un problema, no se ha invertido tiempo en chequear la integridad de los parámetros, queda como responsabilidad del usuario el introducir los mismos de manera correcta.