**第五期：emWin5.20汉字显示方法一（Font&#160;&#160;Converter的使用）|**

**http://bbs.armfly.com/read.php?tid=1580**

**http://bbs.armfly.com/read.php?tid=1604**

**特别说明：原创教程，未经许可禁止转载，教程采用回复可见的形式，谢谢大家的支持。**  
**armfly-x2,x3,v2,v3,v5开发板裸机和带系统的emWin工程已经全部建立，链接如下：**  
[**http://bbs.armfly.com/read.php?tid=1830**](http://bbs.armfly.com/read.php?tid=1830)  
       本期主要讲如何将字库放置到外部存储器的方法，这里以放到SD卡为例，放到其他存储器是一样的。  
其实用官方提供的XBF方式，将字库放到外部字库是最合适的，而且能够使用抗锯齿，暂时还没有调试出  
来，后面有机会了再做尝试，这里用的外置字库的方法还是UCGUI时代遗留下来的。现在贴个图让大家看  
一下实际的显示效果，有一点在这里必须的说明一下，外置字库到SD卡显示大字体的时候会非常的卡。  


图片:0.PNG

本期分为三个小节：  
6. 1  移植方法  
6. 2  点阵字体的说明  
6. 3  将生成的字体移植到armfly - v5开发板上面  
6. 4  实验总结

本部分设定了隐藏，您已回复过了，以下是隐藏的内容

**6. 1  移植方法**  
   6.1.1  要添加两个文件，一个是 GUI\_UC\_EncodeNone.c 用于解码非unicode编码的字符，程序如下

[复制代码](javascript:)

1. #include "GUI\_Private.h"
2. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
3. \*
4. \*       Static code
5. \*
6. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
7. \*/
8. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
9. \*
10. \*       \_GetCharCode
11. \*
12. \* Purpose:
13. \*   Return the UNICODE character code of the current character.
14. \*/
15. static U16 \_GetCharCode(const char GUI\_UNI\_PTR \* s) {
16. if((\*s) > 0xA0)
17. {
18. return \*(const U16 GUI\_UNI\_PTR \*)s;
19. }
20. return \*(const U8 GUI\_UNI\_PTR \*)s;
21. }
22. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
23. \*
24. \*       \_GetCharSize
25. \*
26. \* Purpose:
27. \*   Return the number of bytes of the current character.
28. \*/
29. static int \_GetCharSize(const char GUI\_UNI\_PTR \* s) {
30. GUI\_USE\_PARA(s);
31. if((\*s) > 0xA0)
32. {
33. return 2;
34. }
35. return 1;
36. }
37. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
38. \*
39. \*       \_CalcSizeOfChar
40. \*
41. \* Purpose:
42. \*   Return the number of bytes needed for the given character.
43. \*/
44. static int \_CalcSizeOfChar(U16 Char) {
45. GUI\_USE\_PARA(Char);
46. if(Char > 0xA0A0)
47. {
48. return 2;
49. }
50. return 1;
51. }
52. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
53. \*
54. \*       \_Encode
55. \*
56. \* Purpose:
57. \*   Encode character into 1/2/3 bytes.
58. \*/
59. static int \_Encode(char \*s, U16 Char) {
60. if(Char > 0xA0A0)
61. {
62. \*((U16 \*)s) = (U16)(Char);
63. return 2;
64. }
65. \*s = (U8)(Char);
66. return 1;
67. }
68. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
69. \*
70. \*       Static data
71. \*
72. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
73. \*/
74. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
75. \*
76. \*       \_API\_Table
77. \*/
78. const GUI\_UC\_ENC\_APILIST GUI\_\_API\_TableNone = {
79. \_GetCharCode,     /\*  return character code as U16 \*/
80. \_GetCharSize,     /\*  return size of character: 1 \*/
81. \_CalcSizeOfChar,  /\*  return size of character: 1 \*/
82. \_Encode           /\*  Encode character \*/
83. };
84. const GUI\_UC\_ENC\_APILIST GUI\_UC\_None = {
85. \_GetCharCode,     /\*  return character code as U16 \*/
86. \_GetCharSize,     /\*  return size of character: 1 \*/
87. \_CalcSizeOfChar,  /\*  return size of character: 1 \*/
88. \_Encode           /\*  Encode character \*/
89. };

**6.1.2  另一个要添加的文件主要是用于从外部存储器读取字体点阵数据，并提供为emWin提供相应的接口函数**

[复制代码](javascript:)

1. #include <stddef.h>           /\* needed for definition of NULL \*/
2. #include "GUI\_Private.h"
3. #include "ff.h"
4. #include "bsp.h"
5. #include "MainTask.h"
6. /\* 字模数据的暂存数组,以单个字模的最大字节数为设定值 \*/
7. #define BYTES\_PER\_FONT      1024
8. static U8 GUI\_FontDataBuf[BYTES\_PER\_FONT];
9. FIL fsrc;         // 定义文件操作类
10. FRESULT res;      // 定义操作结果变量
11. UINT bw1;         // 定义读写数量变量
12. /\*---------------------------------------------------------------------------\*/
13. /\*字库外部函数部分-----------------------------------------------------------\*/
14. void GUI\_X\_GetFontData(char\* font, U32 oft, U8 \*ptr, U16 bytes)
15. {
16. res = f\_open(&fsrc, font, FA\_OPEN\_EXISTING | FA\_READ);   //打开字库文件
17. if(res != FR\_OK)
18. {
20. }
21. res = f\_lseek(&fsrc,oft); //找到首地址
22. res = f\_read(&fsrc, ptr, bytes, &bw1); //读取字库点阵数据
23. res = f\_close(&fsrc); //关闭字体
24. }
25. static void GUI\_GetDataFromMemory(const GUI\_FONT\_PROP GUI\_UNI\_PTR \*pProp, U16P c)
26. {
27. U16 BytesPerFont;
28. U32 oft;
29. char \*font = (char \*)pProp->paCharInfo->pData;
30. /\* 每个字模的数据字节数 \*/
31. BytesPerFont = GUI\_pContext->pAFont->YSize \* pProp->paCharInfo->BytesPerLine;
32. if (BytesPerFont > BYTES\_PER\_FONT)
33. {
34. BytesPerFont = BYTES\_PER\_FONT;
35. }
36. /\* 英文字符地址偏移算法 \*/
37. if (c < 0x80)
38. {
39. oft = (c-0x20) \* BytesPerFont;
40. }
41. else
42. {
43. /\* 用于读取12\*12 和 16\*16点阵字符 \*/
44. if(font[21] == '1')
45. {
46. /\* 中文字符地址偏移算法包括符号 \*/
47. oft = ((((c >> 8)-0xA1)) + ((c & 0xFF)-0xA1) \* 94)\* BytesPerFont;
48. }
49. /\* 用于读取24\*24 48\*48 \*/
50. else
51. {
52. /\* 中文字符地址偏移算法包括符号 \*/
53. oft = ((((c >> 8)-0xA1)) + ((c & 0xFF)-0xB0) \* 94)\* BytesPerFont;
54. }
55. }
56. GUI\_X\_GetFontData(font, oft, GUI\_FontDataBuf, BytesPerFont);
58. }
59. void GUIPROP\_X\_DispChar(U16P c)
60. {
61. int BytesPerLine;
62. GUI\_DRAWMODE DrawMode = GUI\_pContext->TextMode;
63. const GUI\_FONT\_PROP GUI\_UNI\_PTR \*pProp = GUI\_pContext->pAFont->p.pProp;
64. //搜索定位字库数据信息
65. for (; pProp; pProp = pProp->pNext)
66. {
67. if ((c >= pProp->First) && (c <= pProp->Last))break;
68. }
69. if (pProp)
70. {
71. GUI\_DRAWMODE OldDrawMode;
72. const GUI\_CHARINFO GUI\_UNI\_PTR \* pCharInfo = pProp->paCharInfo;
73. GUI\_GetDataFromMemory(pProp, c);//取出字模数据
74. BytesPerLine = pCharInfo->BytesPerLine;
75. OldDrawMode  = LCD\_SetDrawMode(DrawMode);
76. LCD\_DrawBitmap(GUI\_pContext->DispPosX, GUI\_pContext->DispPosY,
77. pCharInfo->XSize, GUI\_pContext->pAFont->YSize,
78. GUI\_pContext->pAFont->XMag, GUI\_pContext->pAFont->YMag,
79. 1,     /\* Bits per Pixel \*/
80. BytesPerLine,
81. &GUI\_FontDataBuf[0],
82. &LCD\_BKCOLORINDEX
83. );
84. /\* Fill empty pixel lines \*/
85. if (GUI\_pContext->pAFont->YDist > GUI\_pContext->pAFont->YSize)
86. {
87. int YMag = GUI\_pContext->pAFont->YMag;
88. int YDist = GUI\_pContext->pAFont->YDist \* YMag;
89. int YSize = GUI\_pContext->pAFont->YSize \* YMag;
90. if (DrawMode != LCD\_DRAWMODE\_TRANS)
91. {
92. LCD\_COLOR OldColor = GUI\_GetColor();
93. GUI\_SetColor(GUI\_GetBkColor());
94. LCD\_FillRect(GUI\_pContext->DispPosX, GUI\_pContext->DispPosY + YSize,
95. GUI\_pContext->DispPosX + pCharInfo->XSize,
96. GUI\_pContext->DispPosY + YDist);
97. GUI\_SetColor(OldColor);
98. }
99. }
100. LCD\_SetDrawMode(OldDrawMode); /\* Restore draw mode \*/
101. //      if (!GUI\_MoveRTL)
102. GUI\_pContext->DispPosX += pCharInfo->XDist \* GUI\_pContext->pAFont->XMag;
103. }
104. }
105. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
106. \*
107. \*       GUIPROP\_GetCharDistX
108. \*/
109. int GUIPROP\_X\_GetCharDistX(U16P c)
110. {
111. const GUI\_FONT\_PROP GUI\_UNI\_PTR \* pProp = GUI\_pContext->pAFont->p.pProp;
112. for (; pProp; pProp = pProp->pNext)
113. {
114. if ((c >= pProp->First) && (c <= pProp->Last))break;
115. }
116. return (pProp) ? (pProp->paCharInfo)->XSize \* GUI\_pContext->pAFont->XMag : 0;
117. }

**6.1.3  比如这里要显示12\*12的点阵数据需要提供一个如下内容的文件**

[复制代码](javascript:)

1. #include "GUI.h"
2. extern void GUIPROP\_X\_DispChar(U16P c);
3. extern int GUIPROP\_X\_GetCharDistX(U16P c);
4. GUI\_CONST\_STORAGE GUI\_CHARINFO GUI\_FontHZ12\_CharInfo[2] =
5. {
6. {  6,    6,  1, (void \*)"0:/system/gui\_font/ASC6x12.bin"},
7. {  12,  12,  2, (void \*)"0:/system/gui\_font/HZ12x12.bin"},
8. };
9. GUI\_CONST\_STORAGE GUI\_FONT\_PROP GUI\_FontHZ12\_PropHZ= {
10. 0xA1A1,
11. 0xFEFE,
12. &GUI\_FontHZ12\_CharInfo[1],
13. (void \*)0,
14. };
15. GUI\_CONST\_STORAGE  GUI\_FONT\_PROP GUI\_FontHZ12\_PropASC= {
16. 0x0000,
17. 0x007F,
18. &GUI\_FontHZ12\_CharInfo[0],
19. (void GUI\_CONST\_STORAGE \*)&GUI\_FontHZ12\_PropHZ,
20. };
21. GUI\_CONST\_STORAGE  GUI\_FONT GUI\_FontHZ12 =
22. {
23. GUI\_FONTTYPE\_PROP\_USER,
24. 12,
25. 12,
26. 1,
27. 1,
28. (void GUI\_CONST\_STORAGE \*)&GUI\_FontHZ12\_PropASC
29. };
30. GUI\_CONST\_STORAGE  GUI\_FONT GUI\_FontHZ12x2 =
31. {
32. GUI\_FONTTYPE\_PROP\_USER,
33. 12,
34. 12,
35. 2,
36. 2,
37. (void GUI\_CONST\_STORAGE \*)&GUI\_FontHZ12\_PropASC
38. };

**6.1.4  还有一个重要的问题就是需要在GUI\_Typ.h文件里面添加如下的声明**  
  
#define GUI\_FONTTYPE\_PROP\_USER   \  
            GUIPROP\_X\_DispChar,            \  
            GUIPROP\_X\_GetCharDistX,        \  
            GUIMONO\_GetFontInfo,           \  
            GUIMONO\_IsInFont,              \  
           (GUI\_GETCHARINFO \*)0,          \  
           (tGUI\_ENC\_APIList\*)0               
  
**6. 2  点阵字体的说明**        
关于点阵字体，这里有一点需要特别的说明：  
12\*12点阵汉字和字符  使用的是UCDOS里面的，半角和全角字符显示都正常，汉字也正常  
                                      偏移地址计算：oft = ((((c >> 8)-0xA1)) + ((c & 0xFF)-0xA1) \* 94)\* BytesPerFont;   
16\*16点阵汉字和字符  使用的是UCDOS里面的，半角和全角字符显示都正常，汉字也正常  
                                      偏移地址计算：oft = ((((c >> 8)-0xA1)) + ((c & 0xFF)-0xA1) \* 94)\* BytesPerFont;   
24\*24点阵汉字和字符  使用的是UCDOS里面的，半角显示正常，字库里面全角字符，需要单独添加。  
                                      偏移地址计算：oft = ((((c >> 8)-0xA1)) + ((c & 0xFF)-0xB0) \* 94)\* BytesPerFont; 注意和前两个的不同  
                                      主要是因为这个里面只有汉字。最重要的是这个字体居然显示的时候是躺在的，而且上下镜像。  
                                      所以24\*24的点阵是用软件《字模3》生成的。  
  
48\*48点阵汉字和字符  使用的是UCDOS里面的，半角显示正常，字库里面全角字符，需要单独添加。  
                                      偏移地址计算：oft = ((((c >> 8)-0xA1)) + ((c & 0xFF)-0xB0) \* 94)\* BytesPerFont; 注意和前两个的不同  
                                      主要是因为这个里面只有汉字。  
提供一下我从网上下载的UCDOS字体  
ABC            英文字库文件  
ASC12         ASCII字库文件12X6  
ASC16         ASCII字库文件16X8  
ASC48         ASCII字库文件48X24  
Hzk12          汉字库宋体12X12  
Hzk16          汉字库宋体16X16  
hzk16F         汉字库仿宋16X16  
HZK24F        汉字库仿宋24X24  
HZK24H        汉字库黑体24X24  
HZK24K        汉字库楷体24X24  
HZK24S        汉字库宋体24X24  
HZK24T        全角字符库24X24  
HZK24Z        汉字库篆体24X24  
HZK40S        汉字库宋体40X40  
HZK40T        全角字符库40X40  
HZK48S        汉字库宋体48X48  
HZK48S        全角字符库48X48  
http://bbs.armfly.com/images/wind/file/zip.gif[字库UCDOS.zip](http://bbs.armfly.com/job.php?action=download&aid=1805) (3339 K) 下载次数:196   
  
  
**6. 3  将生成的字体移植到armfly - v5开发板上面**  
    根据上面的说明将其添加到工程中即可，通过截图，实际显示效果如下：  


图片:0.PNG

**6. 4  实验总结**  
        实际使用中，从SD卡读取大点阵的数据并显示，显示速度比较的慢，不能满足实际的应用，所以建议大字体存储到内部flash（当然是只生产部分字体）。  
还有一个比较郁闷的地方是，以前我用字模提取软件《字模3》非常的好用，字体能够在UCGUI上面正常的显示，现在用emWin，32\*32以上的点阵就是无法显示，  
下面这个图就是以前用UCGUI时的显示效果，软件《字模3》生成的汉字前面没有全角字符，这点一定要注意，也就是说里面只有汉字。  


图片:1.jpg

SD卡根目录要放的文件  
http://bbs.armfly.com/images/wind/file/zip.gif[system.zip](http://bbs.armfly.com/job.php?action=download&aid=1808) (1326 K) 下载次数:111   
  
程序下载  
http://bbs.armfly.com/images/wind/file/zip.gif[STemWin5.20+uCOS-III+FatFS.zip](http://bbs.armfly.com/job.php?action=download&aid=1809) (16357 K) 下载次数:222