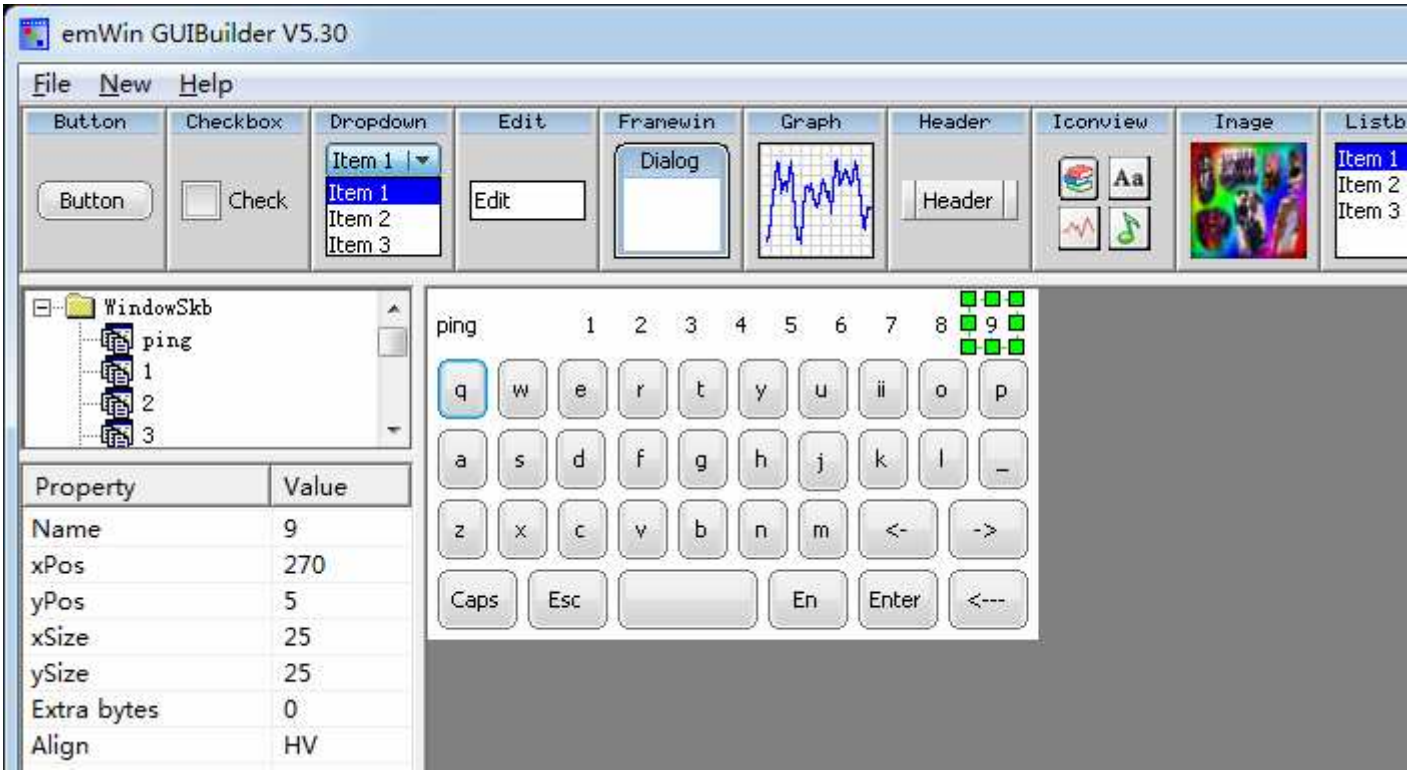
emwin 2-day quick tutorial 010_全键盘中文汉字拼音输入法

这个 emWin 的中文拼音输入法已经做的很完善,也经过比较详细的测试,可以直接拿来当做模板使用(根据需要修改键盘的大小);实现的程序也非常简洁,同时也使用了 emWin 的一些知识点,值得阅读学习;若你还不会用 emWin,可以看先看前面的教程.

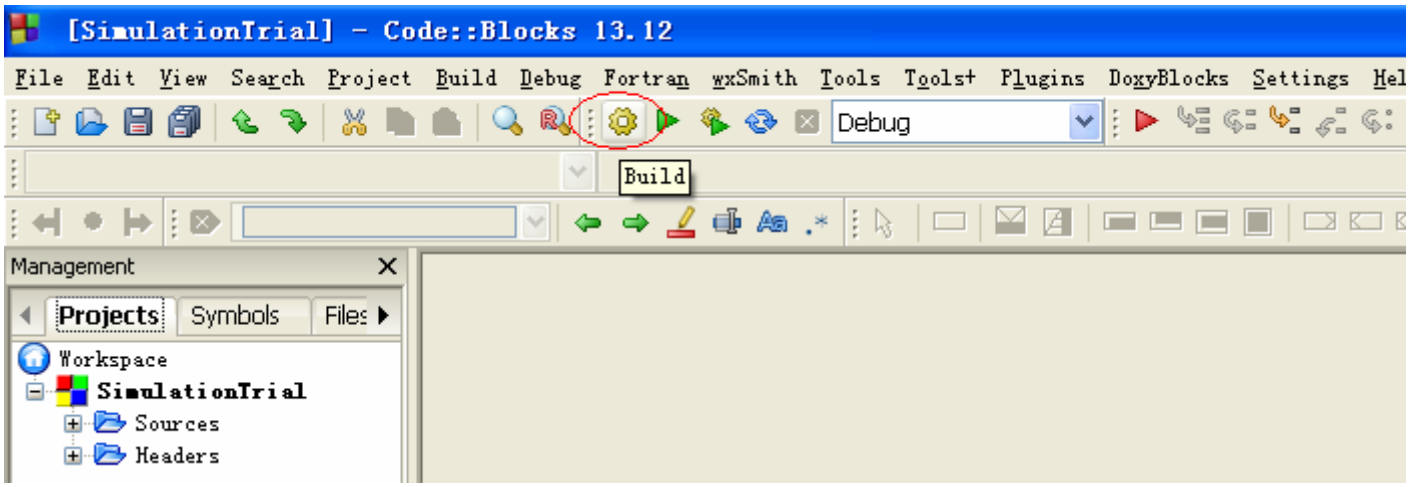


小技巧:放置 xxxx 控件之后不要用鼠标移动,用上下左右建移动更容易对齐(步进是 5).

备注:GUIBuilder 生成的 c 文件,用户代码最好加在"USER START"和"USER END"之间,其他地方除了数字以外,不要做任何修改,否则 GUIBuilder 将无法再次打开此 c 文件;另外, GUIBuilder 再次打开编辑并保存时,"USER START"和"USER END"之间的内容将不会被更改;还有 c 文件不要在 GUIBuilder 打开状态直接去修改 c 代码,否则点 GUIBuilder 保存之后修改的内容将会丢失.

使用例程:

用 emWin Code::Blocks(www.neqee.com 可下载)打开工程文件 SimulationTrial.cbp 然后编译工程:



运行 emWin 中文拼音输入法程序:

点击编辑框后自动弹出键盘的实现方法:

FramewinPage1DLG.c

```
81    // USER START (Optionally insert additional static code)
82    static void LOC_edit_0to3_action(WM_MESSAGE * pMsg)
83    {
84      char acBuffer[40];
85      int posiX, posiY, sizeX[2], sizeY[2];
86      //determine the keyboard position
87      sizeX[0] = WM_GetWindowSizeX(WM_GetClientWindow(pMsg->hWin));
88      sizeY[0] = WM_GetWindowSizeY(WM_GetClientWindow(pMsg->hWin));
89      sizeX[1] = WM_GetWindowSizeX(hPage[3]);
90      sizeY[1] = WM_GetWindowSizeY(hPage[3]);
91
92      if(sizeX[0] - (WM_GetWindowOrgX(pMsg->hWinSrc)-WM_GetWindowOrgX(WM_GetClientWindow(pMs
93        posiX = sizeX[0] - sizeX[1] + WM_GetWindowOrgX(WM_GetClientWindow(pMsg->hWin));
94      else posiX = WM_GetWindowOrgX(pMsg->hWinSrc);
95
96      if(WM_GetWindowOrgY(pMsg->hWinSrc) - WM_GetWindowOrgY(WM_GetClientWindow(pMsg->hWin))
97        posiY = WM_GetWindowOrgY(pMsg->hWinSrc) - sizeY[1];
98      else posiY = WM_GetWindowOrgY(pMsg->hWinSrc) + WM_GetWindowSizeY(pMsg->hWinSrc);
99
100     WM_MoveTo(hPage[3], posiX, posiY);
101     WM_ShowWindow(hPage[3]);
102   }
```

总体思想是在 ID_EDIT_0～ID_EDIT_4 点击消息中用 WM_ShowWindow()函数显示出键盘,并在"Ese"按键的点击消息中用 WM_HideWindow()函数隐藏键盘. 当然,如果 RAM 空间有限的情况下,你也可以用 Create/Delete 方式做多页面切换. 其他代码的作用是判断并移动键盘到合适的位置. 总体的思想是判断键盘有没有超出屏幕范围,如果超出则移动键盘到合适

的位置,WM_GetWindowOrgX(pMsg->hWinSrc)是获取编辑框相对于桌面位置坐标的函数,如果不理解"pMsg->hWinSrc",可以看:教程 004_控件聚焦和用户消息的使用方法,其他 API 函数可以查阅 emWin 说明书,这些 API 函数是非常有用的,可以记住它们.

键盘处理程序:
这里只能简单的说说处理的思想,很多细节还需要耐心去看程序才行.
在初始化消息 WM_INIT_DIALOG 中,用下面程序禁止掉键盘里面所有按键的聚焦,因为如果不这样做的话,点击键盘之后,焦点光标就不会停留在编辑框了:

```
for(ii=ID_BUTTON_0; ii<=ID_BUTTON_34; ii++) BUTTON_SetFocussable(WM_GetDialogItem(pMsg->hWin, ii), 0);
```

在 ID_BUTTON_0～ID_BUTTON_26 的点击消息中,根据当前输入法做相应的处理,如果是英文或符号比较简单,获取按键的字符并用 GUI_StoreKeyMsg()函数向 emWin 内部储存键值即可,emWin 内部会自动处理并显示到编辑框里去;如果是中文,则将按键字符显示到 ID_TEXT_0 文本控件(显示拼音的那个),并用 WM_SendMessage()函数向 ID_TEXT_0 发送一个 WM_NOTIFICATION_USER+0 消息,通知 ID_TEXT_0 根据自己的内容去检索拼音并将汉字显示到 ID_TEXT_1～ID_TEXT_9 文本控件中;为什么要用发消息?为什么不直接在那里处理?因为很多地方都需要重新检索并更新 ID_TEXT_1～ID_TEXT_9 的内容,当然你也可以用调用函数的方式.

汉字选择:
在 ID_TEXT_1～ID_TEXT_9 文本控件的点击消息中,把自己的内容显示到当前聚焦的编辑框中去即可:

n\WindowSKB.c

```
  case ID_TEXT_6:
  case ID_TEXT_7:
  case ID_TEXT_8:
  case ID_TEXT_9:
    switch(NCode) {
    case WM_NOTIFICATION_CLICKED:
      break;
    case WM_NOTIFICATION_RELEASED:
      if(inputTYPE!=2) break;
      if(TEXT_GetText(WM_GetDialogItem(pMsg->hWin, Id), &tempBUFF[0], sizeof(tempBUFF)) == 0) break
      EDIT_GetText(hItem=WM_GetFocussedWindow(), &charBUFF[0], sizeof(charBUFF));
      strcat(charBUFF, tempBUFF);
      EDIT_SetText(hItem, &charBUFF[0]);
```

汉字检索翻页:

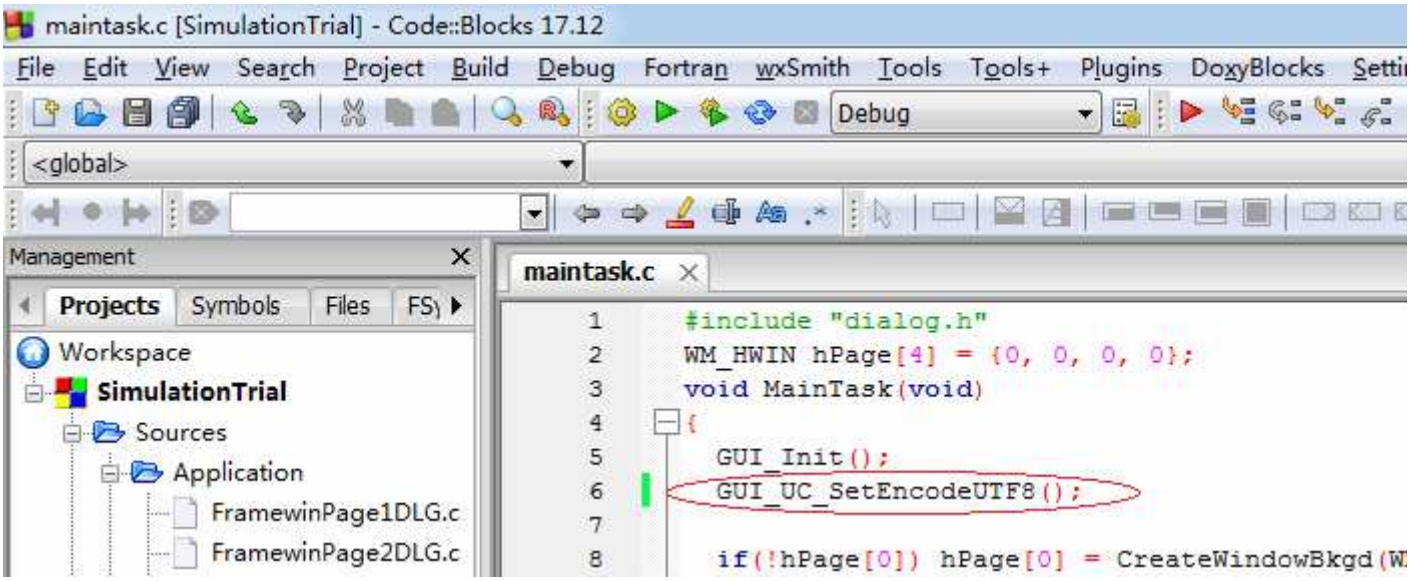
这是中文拼音输入法程序设计的难点,如果思路不好,程序会非常臃肿,笔者也是苦苦思索才非常简洁的实现这一点:

\WindowSKB.c

```
if(NCode == WM_NOTIFICATION_USER+0)
{
  TEXT_GetText(pMsg->hWinSrc, &charBUFF[0], sizeof(charBUFF));  //Get ID_TEXT_0
  for(i=0; i<sizeof(pbuffPY)/4; i++) pbuffPY[i] = 0;
  pstrPY = (char **)&pbuffPY[0];
  pstrTEMP = (char *)PYSearch(&charBUFF[0]);
  //每9个汉字储存一个指针到pbuffPY[i]
  for(i=0; pstrTEMP&&*pstrTEMP&&(i<sizeof(pbuffPY)/4-1); i++)
  {
    pbuffPY[i] = (int *)pstrTEMP;
    //pstrTEMP move 9 Chinese characters
    for(temp=ID_TEXT_1; temp<=ID_TEXT_9; temp++)
    {
      pstrTEMP = pstrTEMP + GUI_pUC_API->pfGetCharSize(pstrTEMP); //Using 'pfGet
      if(!pstrTEMP || !*pstrTEMP) break;
    }
  }
}
else if(NCode == WM_NOTIFICATION_USER+1)        //'<-'
{
  if(*(pstrPY-1) && (pstrPY != (char **)&pbuffPY[0])) pstrPY--;
}
else if(NCode == WM_NOTIFICATION_USER+2)        //'->'
{
  if(*(pstrPY+1) && (pstrPY!=(char **)&pbuffPY[sizeof(pbuffPY)/4-1])) pstrPY++;
}
```

基本思想是:在拼音检索时,把检索到的中文字符串分组,每组 9 个汉字(因为每次显示 9 个汉字 ID_TEXT_1～ID_TEXT_9),每一组对应一个指针并将指针存于 pbuffPY[]数组中,这样翻页时就知道怎么做了吧?

这里重点解释下 GUI_pUC_API->pfGetCharSize(),这个函数的作用是返回一个汉字占用几个字节,也许你会说:一个汉字不是占用两个字节吗?错!要看你的 emWin 用什么编码,如果是 GBK 或 GB2312 编码占两个字节没错,但如果是 UTF-8 编码呢?就不是占用两个字节了;是在哪里告诉 emWin 用什么编码呢?



另外,如果你用的是 GBK 或 GB2312 编码,emWin 是没有 GUI_UC_SetEncodeGBK()和 GUI_UC_SetEncodeGB2312()函数的,需要自己写,也许在德国人眼里,是没有 GBK 或 GB2312 概念吧.