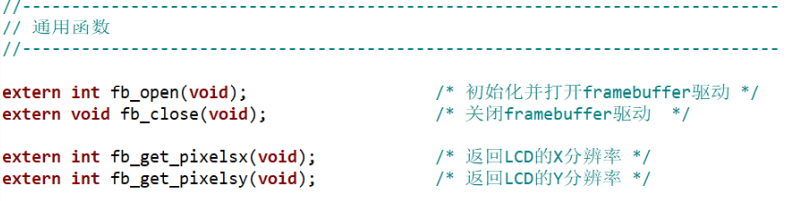
## GPIO驱动

LoongIDE集成开发环境下，已经配置好了GPIO的驱动，驱动文件在ls1x-drv/gpio中。通过调用GPIO相关驱动函数，将对应按键的GPIO端口配置为中断输入模式，再添加其对应的中断服务程序即可实现人机交互。

## LCD驱动

LoongIDE集成开发环境下，已经配置好了LCD显示屏的驱动，驱动文件在ls1x-drv/fb中，用户接口函数如下：



其中已经实现了设备初始化函数和绘图函数。再在绘图函数中，我们主要使用“指定颜色打印字符串”以及“使用指定颜色填充矩形”。

## 第一部分 硬件电路连接

|  |  |  |  |
| --- | --- | --- | --- |
| 按键 | GPIO端口 | 按键 | GPIO端口 |
| 玩家一（上） | GPIO36 | 玩家二（上） | GPIO43 |
| 玩家一（下） | GPIO34 | 玩家二（下） | GPIO45 |
| 玩家一（左） | GPIO37 | 玩家二（左） | GPIO46 |
| 玩家一（右） | GPIO35 | 玩家二（右） | GPIO47 |

## 第二部分 两种显示模式的构建

1. 屏幕设计问题分析

首先TFT分辨率要高很多，每个像素点都特别小，不能再用一个像素代表贪吃蛇的一格。所以，在显示贪吃蛇图像时需要对屏幕进行分块，每一块显示一格贪吃蛇。

其次，贪吃蛇行进过程中并不是每个点都在变化，如果每次刷新都要重新计算显示之后再向TFT发送数据，无疑对MCU资源造成了很大浪费，其刷新图像的速率（帧率）就特别低下，程序运行效率相当低。本设计中的贪吃蛇算法用到，去某个位置改变对应像素值的函数，以此高效率实现贪吃蛇平移。所以，构建一个图像显存，贪吃蛇运行逻辑改变图像显存的值即可，输出时根据显存的数据输出到屏幕。

最后，调用LoongIDE之中的字符串打印函数即可在对应位置显示对应字符。

对于屏幕显示的设计总结如下：字符显示API、图像模式显存、图像模式显示函数、更改图像显存、显存清除等。

1. 屏幕设计思路

屏幕划分：原屏幕分辨率800\*480，划分为40\*24块（块坐标如下图左），每一块分辨率20\*20（像素点坐标如下图右）。在代码中，“H”与“L”表示块坐标，用矩形框填充函数可不使用具体的像素坐标。

19行0列

19行19列

23行39列

23行0列

0行39列

0行0列

0行19列

0行0列

图像模式显存：对于贪吃蛇来说，该模式不需要显示很精细的图像，所以定义数组TCSK[40][24]来存储40行24列的图像模式显存。该数组中存储不同的整型数据，每个不同的数值代表对应的每个20\*20像素显示区域不同的颜色。

图像模式显示函数display\_TCS：先发送相关命令设置起始像素位置等进入显示模式，用“H”、“L”两层for循环搭配矩形框填充函数来实现图像显示，如果图像模式显存TCSK[H][L]为0则显示黑色，否则fb\_fillrect(L\*20,H\*20,(L+1)\*20,(H+1)\*20,TCSK[H][L]);即根据显存存储的数据改变该像素的值。

主函数中改变图像模式显存：此逻辑设计简单，只需要将原来去某个位置（x，y）改变对应显示的函数，替换为给显存TCSK[x][y]赋值即可。

字符显示，构建一个put\_text()函数其中嵌套字符串输出API即可。

显存清除函数的设计最容易，只需用for循环将图像显存二维数组二维数组清零即可。

1. 游戏逻辑设计

（1）数据存储：二维数组“flg[4][2]”存储蛇头位置变化（4个方向，2个坐标）；两个变量“m/n”存储可视化窗口大小；变量“tim”存储游戏执行延时（即游戏速度）；用结构体“tan”存储每条蛇每个坐标点的位置，结构体“food”记录两个食物的坐标；“tot”记录蛇的长度；“f”记录当前运行状态；食物存在点矩阵“vis”；记录得分、食物个数、背景颜色、游戏模式等。

（2）随机数的产生：那么怎么获取一个随机数呢？不停向前的时间可以帮助我们解决此问题，用get\_clock\_tick()函数来获取自系统启动后的毫秒数，取余运算后可以作为伪随机数来使用。每次get\_clock\_tick()返回的时间值不同，取余数之后产生的伪随机数就不同。

（3）模式选择模块：输出提示信息，根据按键改变全局变量“ms”的值来选择模式。

（4）边框显示模块：在电脑中一个方块字符“▇”占用一行两列，所以需要根据显示窗口大小计算每个边框点的坐标值。因边框固定，所以初始化的时候显示一下，之后就不需要再刷新了。

（5）小蛇数据初始化：根据对应模式，选择玩家一、玩家二、机器人一以及机器人二中的一条或者两条小蛇进行初始化（颜色与位置）。

（6）食物的随机投喂：判断如果其中一个食物不见了，那么生成两个随机数，根据窗口大小取余运算之后得到X和Y坐标，再检测投喂点是否为空，如果不是则生成新坐标，如果是则去对应位置投放食物，并更新与食物相关的数据。

（7）没有按键按下时小蛇自动运行：根据不同的小蛇设置显示颜色，如果是机器人则先判断按下一步前进方向自动寻路，如果是玩家则保持当前运行状态。根据此运行状态更新蛇头的位置。之后用for循环（蛇的长度为循环次数），更新蛇身每个坐标点的数据，最后擦除蛇尾。如果在此过程中，小蛇有吃掉一个食物则在尾巴处显示方块字符“▇”并更新蛇的长度与食物数据并判断是否得分足够胜利。判断：如果一方撞到蛇的身体或者墙壁则游戏结束。都没有撞到的话，显示蛇头（蛇头颜色与身子不同便于识别，故需要单独显示）。

（8）检查某个点是否为空（随机投喂食物用到）：设置布尔型变量“t”初始值为一，如果该点在边界外将t置为0，如果该点为蛇身体的一个点，也将t置为0，然后返回t的值。函数返回值为1即为空。

（9）机器人寻路模块：先根据蛇头与两个食物横纵坐标之差的绝对值，来判断距离哪个食物最近，然后根据坐标只差的正负确定8个方位，前方有障碍则运行other函数转变方向继续判断，根据上述过程计算出相应运行状态。

（10）游戏结果模块：游戏结束方式分为“一方积分足够”或者“一方撞到障碍物”，依据不同结束方式、游戏模式以及得分显示相应提示内容。

1. 寻路算法解析

寻路本质就是寻找合适的方向：计算蛇头与两个食物距离（坐标之差绝对值在求和|x1-x2|+|y1-y2|）。如果0号食物比1号食物近或者1号食物消失，则将目标设定为0号食物，否则设定为1号食物。For循环遍历4个方向，首先用if确保此方向不是朝向身后（剩下3个方向），其次用if确保蛇头按照此方向前进一格之后不是障碍物，如果都满足则计算蛇头按照此方向前进一格之后和目标食物的距离，如果此距离小于之前计算出的最小距离，则将此方向更改为新的前进方向，并更新最小距离。

该循环结束后，将会选择出距离目标食物距离最小的方向，并确保了不向后，且不主动撞障碍物。当蛇头3其他三个方向都是障碍物，进入死胡同的之后，遍历完四个方向则不会改变原来的前进方向，被迫无奈撞墙死亡（不会造成死循环）。

## 第三部分 构建工程

1. 在“bsp”文件中取消fb的注释以启用功能块
2. 在main.c下添加代码