**“路基”实时操作系统 内核手册**

**初稿**

V1.10R Build 1651

目录

1 总览

2 进程调度单元(目前只支持轮转时间片调度,属于弱实时,但即将支持优先级调度)

3 内存管理单元

4 信号管理单元

5 资源管理单元

6 文件系统管理单元

7 系统调用

8 命令行界面

9 图形用户界面

10 中断管理系统(未实现)

11 错误管理单元(部分实现)

12 随机数产生单元与安全单元(未实现)

(13 用户应用程序)

14说明:

本OS使用RVMDK(Keil Uvision)进行开发.这是一个完整的工程,本版本为Cortex-M3的移植版本.

本OS目前可以运行在Cortex-M系列的处理器上,包括最新的Cortex-M4.

目前标准开发板未决定,欢迎大家多提些建议,采用哪个开发平台最好.

1.总览

1.1系统特性

1.1.1设计的原则:简单就是美.

此操作系统的大部分设计都被大大简化.

1.1.2通用性强:

通过编译选项的设置可以使系统针对性的适用于多种CPU.

1.1.3要求极低:

最小内核可以在1KB ROM 和128B RAM之上运行.

1.1.4界面友好:

提供可选的命令行界面与图形化界面.

1.1.5完全自由:

遵循GPLV2许可证的规定.

1.2系统架构

[APPLICATIONS]

==================================================================

[RKGUI] [RKSH] (完成,但不很完善)

-------------------------------------------------------------------------------------------------------------

[FILESYSTEM] [INTERNET] (此层完成中)

------------------------------------------------------------------------------------------------------------

[SIGNAL] [RESOURCE] [MEMORY] (此层完成)

==================================================================

[CORE/INIT] (此层完成)

1.3文件组织

所有的内核源代码均以模块的方式命名,文件名一律小写并采用单数形式.

文件的组织图如下:

<xxx.h>头文件,里面存放:

->文件属性信息

->预处理器控制指令

->要包含的其他头文件

->本模块中用到的私有函数(static)

<xxx.c>c语言源代码,里面存放:

->文件属性信息

->与本模块相关的c语言函数

<defines.h>宏定义专用文件,里面存放

->操作系统中所使用的全部的宏定义

->所有的编译选项设置

<globals.h>全局定义专用文件,里面存放

->所有全局变量的定义与初始化

->所有全局函数的定义与初始化

<externs.h>全局声明专用文件,里面包括:

->所有全局变量的声明

->所有全局函数的声明

<typedefs.h>typedef和struct定义专用文件,里面包括:

->所有的struct定义

->所有的typedef定义

<arch.s>汇编语言程序合集文件,里面包括:

->所有的与架构相关的汇编语言程序

1.4注意事项

系统调用都以”Sys\_”开头.用户应用程序中不应有以”Sys\_”开头的子函数,否则会造成混淆.以”\_Sys\_”开头的是操作系统内用函数.绝不应该在应用程序中直接调用它们.一般来讲,这些函数没有原子操作保护,用户直接调用有可能在原子操作中导致中断,从而导致死机等问题.

[IN]表示传入的参数是给函数传入参数的.

[OUT]表示传入的参数是用来接收函数返回值的,通常是一个指针(函数的参数区),或者函数的返回值

1.5版权声明

此操作系统的版权归pry所有.

2 进程/线程调度单元

2.1模块名称<kernel.c>,头文件<kernel.h>

2.2模块作用:

加载系统的内核.

2.3工作原理

2.3.1操作系统的启动

操作系统启动后,先将内核加载,再次第加载要开机启动的进程.之后初始化定时器和应用程序可用内存,然后调用函数\_Sys\_Start()开始任务轮转,完成操作系统的启动.

2.3.2任务的轮转

由Systick定时器中断触发,调用Systick\_Handler()完成任务切换,任务列表在每次大轮转前由Init进程确定.

2.4关键函数介绍

2.4.1 void \_Sys\_Kernel\_Load(u32 SP,u32 Ticks)

系统关键函数,加载系统的0号进程(Init).用户不应调用.

@入口参数:

[IN]u32 SP:本进程的堆栈绝对地址.

[IN]u32 Ticks:给Init进程初始化时分配的时间片.单位是定时器的滴答数.因为Init进程很特殊,在一个大轮转中只会完整的运行一次,故这参数实际无意义.

@出口参数:无

@用户调用:禁止

2.4.2 void \_Sys\_Proc\_Load(void\* Task,u32 SP,u8 PID,u32 Ticks)

系统关键函数,加载一个进程(任务).用户初始化时参与运行的进程都必须在这里加载,并合理确定其堆栈位置和时间片.时间片不能过大,否则定时器无法容纳,也不能太小,否则会导致频繁的上下文切换.

@入口参数:

[IN]void\* Task:任务的入口地址,即函数名.

[IN]u32 SP:任务的堆栈绝对地址.

[IN]u8 PID:任务的进程号.

[IN]u32 Ticks:给任务分配的时间片,单位是系统计时器滴答数.

@出口参数:无

@用户调用:禁止

2.4.3 u8 Sys\_Start\_Process(void\* Task,u32 SP,u32 Ticks)

可供用户调用的任务加载函数,谨慎使用并合理确定其堆栈位置.

@入口参数:

[IN]void\* Task:任务函数的绝对地址.

[IN]u32 SP:堆栈的绝对地址

[IN]u32 Ticks: 给任务分配的时间片,单位是系统计时器滴答数.

@出口参数:

[OUT]u8 : 新开始进程的PID.如果失败(任务槽不够用),会返回”ENOPID”,即0x00.

@用户调用:允许

2.4.4 void \_Sys\_Systick\_Init(u32 Ticks)

系统定时器的初始化函数,初始化并启动定时器

@入口参数:

[IN] u32 Ticks:初始的定时器值,一般取最大值.

@出口参数:无

@用户调用:禁止

2.4.5 void SysTick\_Handler(void)

中断处理函数,也是任务切换函数,这个函数绝不允许在用户应用程序中调用.

@入口参数:无

@出口参数:无

@用户调用:禁止

2.4.6 void \_Sys\_Memory\_Init(void)

应用程序可用内存初始化函数,不能在应用程序中调用.

@入口参数:无

@出口参数:无

@用户调用:禁止

2.4.7 void \_Sys\_Init(void)

零号进程Init,专门负责进程调度.不能在应用程序中调用.他也是唯一系统中的一次运行进程,也即在时间片用完之前自行申请调度.

@入口参数:无

@出口参数:无

@用户调用:禁止

2.4.8 void \_Sys\_Arch(void)

一号进程Arch,专门负责系统服务.不能在应用程序中调用.

@入口参数:无

@出口参数:无

@用户调用:禁止

2.4.9 u8 Sys\_Get\_PID(void)

获得当前进程的PID的函数.

@入口参数:无

@出口参数:

[OUT]u8 :返回的当前正在运行进程的PID.

@用户调用:允许

2.4.10 void Sys\_Switch\_Now(void)

立即进行任务切换的函数.使得当前主调进程放弃未用完的时间片,并且接受系统的立即调度.

@入口参数:无

@出口参数:无

@用户调用:允许

2.4.11 int main(void)

操作系统入口函数,不能在用户应用程序中调用.

@入口参数:无

@出口参数:无(int是假返回值)

@用户调用:禁止

3 内存管理单元

3.1模块名称<memory.c>,头文件<memory.h>

3.2模块作用:

系统的内存管理单元.

3.3工作原理

3.3.1分配内存

当分配内存函数被调用时,它会到内存分配表中去找一块空出来的内存,登记,并且将指针传回用户应用程序.如果找不到合适的内存,返回一个空指针.

3.3.2申请共享内存

当共享内存申请启动时,系统先去找一块空出来的内存,登记在一号进程Arch的名下,而且将申请共享内存的进程的进程号登记在共享内存表中.

当有另外的进程申请加入这块共享内存时,它的进程号也将会被写入表中.

3.3.3释放内存

当内存被释放时,操作系统先回卷指针,然后开始释放指针指向的内存.注意,释放内存时所用的指针指向的可以是这块内存中的任意位置,不一定要是内存的开头,即分配函数返回的指针.

3.3.4释放共享内存

当共享内存被释放时,系统先去删除该进程在共享内存使用表中的登记,然后看是否还有进程在使用共享内存.如果有,则不做其他动作;如果没有,则旋即释放这块内存.

3.4关键函数介绍

3.4.1 void\* Sys\_Malloc(u32 Bytes)

内存分配函数.系统的最基本函数之一.

@入口参数:

[IN]u32 Bytes:欲分配的内存大小,单位Byte.

@出口参数:

[OUT]void\*:指向这块内存开头的指针.如果失败会返回”ENOMEM”(0x00).

@用户调用:允许.

注意:由于内存分配有粒度,最小分配的内存单位由宏定义NUMBER\_OF\_MEM\_PAGES决定,一次性分配大块内存比多次分配小块内存要好,内存利用率高.

3.4.2 void\* \_Sys\_Malloc(u32 Bytes)

操作系统内部用内存分配函数,用户不要调用,无原子操作保护.

@入口参数:

[IN]u32 Bytes:欲分配的内存大小,单位Byte.

@出口参数:

[OUT]void\*:指向这块内存开头的指针.如果失败会返回”ENOMEM”(0x00).

@用户调用:禁止

3.4.3 void Sys\_Free\_Memory(void\* Mem\_Ptr)

内存释放函数,最基本函数之一.

@入口参数:

[IN]void\* Mem\_Ptr:欲释放内存的地址.不一定要是起始地址,可以是任意位置.

@出口参数:无(如果失败也不返回值).

@用户调用:允许

3.4.4 void \_Sys\_Free\_Memory(void\* Mem\_Ptr)

操作系统内用函数,无原子操作保护.

@入口参数:

[IN]void\* Mem\_Ptr:欲释放内存的地址.不一定要是起始地址,可以是任意位置.

@出口参数:无(如果失败也不返回值).

@用户调用:禁止

3.4.5 void Sys\_Free\_All\_Memory(void)

释放本进程申请的所有内存.

共享内存由于登记在一号进程Arch名下而不会释放.这个函数也不会释放这个进程占用的共享内存,要想释放共享内存要使用其它函数.

@入口参数:无

@出口参数:无(如果失败也不返回值).

@用户调用:允许

3.4.6 void \_Sys\_Free\_All\_Memory(u8 PID)

操作系统内用函数,无原子操作保护.

@入口参数:

[IN]u8 PID:欲释放所有内存的进程号

@出口参数:无(如果失败也不返回值).

@用户调用:禁止

3.4.7 void\* Sys\_Shared\_Malloc(u8\* SMID,u32 Bytes)

申请分配一块共享内存.进程交流的基本方法之一

@入口参数:

[OUT]u8\* SMID:共享内存的编号,用来传出值.

[IN]u32 Bytes:欲申请的共享内存大小.

@出口参数:

[OUT]void\*:申请的共享内存的起始地址.如果内存不够而引起失败,则会返回”ENOSHM”(0x00).

@用户调用:允许

3.4.8 void\* Sys\_Join\_Shared\_Malloc(u8 SMID, u32\* Bytes)

申请加入一块共享内存编号为SMID的共享内存.Bytes是用来返回值的,返回这块共享内存的大小,返回的void\*指针指向共享内存的起始地址.Bytes所存的返回值很可能与共享内存申请者申请的共享内存大小不同(会偏大,这也是这块共享内存的实际大小),是因为内存分配有最小粒度导致的.

@入口参数:

[IN]u8 SMID:欲加入的共享内存的编号.

[OUT]u32 Bytes:欲申请加入的申请的共享内存大小,用来传出值.

@出口参数:

[OUT]void\*:申请加入的的共享内存的起始地址.如果失败会返回”ENOSHM”(0x00),并且Bytes也将变为”ENOSHM”(0x00).

@用户调用:允许

3.4.9 void Sys\_Free\_Shared\_Memory(u8 SMID)

使现有进程放弃对编号SMID的共享内存的使用权.

@入口参数:

[IN]u8 SMID:欲放弃使用的共享内存的编号.

@出口参数:无(如果失败也不返回值).

@用户调用:允许

3.4.10 void Sys\_Free\_All\_Shared\_Memory(void)

使现有进程放弃对所有共享内存的使用权.

@入口参数:无

@出口参数:无(如果失败也不返回值).

@用户调用:允许

3.4.11 void \_Sys\_Free\_All\_Shared\_Memory(u8 PID)

操作系统内用函数,使进程号为PID的进程放弃对所有共享内存的使用权.

@入口参数:

[IN]u8 SMID:共享内存的编号,用来传出值.

@出口参数:无(如果失败也不返回值).

@用户调用:禁止

4信号管理单元

4.1模块名称<signal.c>,头文件<signal.h>

4.2模块作用:

系统的信号管理单元,管理系统的信号传递机制.

4.3工作原理

4.3.1发出信号

当信号被发出后,在下一次Init进程给轮转分配时间片时,会考虑这个选项并进行相关处理.只有OCCUPY状态且不睡眠和僵尸化的进程会被分配时间片.

4.3.2处理信号

当任务切换进行到最后时,调用”void \_Sys\_Signal\_Handler(u8 PID)”进行进程信号检测,如果检测到了信号则进行相关处理.

4.3.3睡眠处理

当检测到其他进程给它发睡眠(SIGSLEEP)信号时,调用睡眠处理函数.具体操作是将该进程的状态表改成SLEEP状态,这样Init进程在进行下一次轮转时间分配时便不会给它分配时间片,使该进程睡眠

4.3.4 醒来处理

当检测到其他进程给它发醒来(SIGWAKEUP)信号时,调用睡眠处理函数.具体操作是将该进程的状态表改成OCCUPY状态,这样Init进程在进行下一次轮转时便会继续给它分配时间片,使该进程醒来.

4.3.5杀死处理

当检测到其他进程给它发杀死(SIGKILL)信号时,调用杀死处理函数.具体操作是释放该进程占用的所有内存,解除它对所有的共享内存的使用权,并且释放它使用的所有资源,最后将该进程在系统中的所有登记除去,彻底突然停止该进程的执行.

4.3.6客制化处理

当检测到其他进程给它发客制化处理(SIGCUSTOM)信号时,调用客制化处理处理函数.具体操作是将客制化处理函数从其所在数组中提取出来,并进行执行.

4.4关键函数介绍

4.4.1 void \_Sys\_Signal\_Handler(u8 PID)

系统的信号处理函数,它会在每次任务切换时使用,然后判断该进程是否收到信号,如果收到,则进行相关动作,见下面.

@入口参数:

[IN]u8 PID:将要被进行信号处理的进程号.

@出口参数:无

@用户调用:禁止.

4.4.2 void \_Sys\_Kill\_The\_Process(u8 PID)

操作系统内部用杀死函数.

@入口参数:

[IN]u8 PID:欲杀死的进程的进程号.

@出口参数:无

@用户调用:禁止

4.4.3 void \_Sys\_Sleep\_The\_Process(u8 PID)

操作系统内部用睡眠函数.

@入口参数:

[IN] u8 PID:欲使之进入睡眠状态的进程的进程号.

@出口参数:无

@用户调用:禁止

4.4.4 void \_Sys\_Wakeup\_The\_Process(u8 PID)

操作系统内部用睡眠函数.

@入口参数:

[IN] u8 PID:欲唤醒(使之退出睡眠状态)的进程的进程号.

@出口参数:无

@用户调用:禁止

4.4.5 void Sys\_Send\_Signal(u8 PID,u8 Signal)

信号发出函数.向一指定的进程发出信号.

@入口参数:

[IN] u8 PID:欲对之发出信号进程的进程号.

[IN]u8 Signal:发出的信号内容,可以是以下几个之一:

NOSIG [0x00] 一个空信号

SIGKILL [0x01] 立即杀死进程,对Init与Arch无效.

SIGSLEEP [0x02] 睡眠信号

SIGWAKEUP [0x04] 唤醒信号

SIGCUSTOM [0x08] 客制化信号

SIGNAL\_(x) [X] 自用信号(在自己的应用程序中处理, 操作系统不干涉.)

@出口参数:无

@用户调用:允许

4.4.6 void Sys\_Process\_Quit(u8 End\_Val)

进程退出函数,由进程调用来使自己退出,可以返回一个四位的二进制返回值.同时将PCB\_PID\_STATUS的前四位置成ZOMBIE状态.这个进程放弃了所有的私有内存空间,共享内存使用权以及资源,但仍未放弃它在进程登记表中的位置,等待其他的进程去取返回值.

@入口参数:

[IN]u8 End\_Val:欲给出的返回值,后四位有效,这个返回值会被存储在PCB\_PID\_STATUS表中

@出口参数:无

@用户调用:允许

4.4.7 u8 Sys\_Get\_The\_Process\_Retval(u8 PID)

取一个进程的返回值,即调用函数4.4.6后返回的值,然后将进程从进程登记表中彻底除去.

@入口参数:

[IN]u8 PID:欲取返回值的自动结束的进程的进程号.

@出口参数:

[OUT]u8:取回的返回值,如果失败,取回的是0xFF,即256.

@用户调用:允许

4.4.8 void Sys\_Register\_SIGCUSTOM\_Handler(void (\*Func)(void))

在系统中给本进程自身登记SIGCUSTOM的处理函数

@入口参数:

[IN] void (\*Func)(void) [OUT]u32 Bytes:欲登记的处理函数,必须无返无参.

@出口参数:无

@用户调用:允许

5 资源管理单元

5.1模块名称<resource.c>,头文件<resource.h>

操作系统的资源管理单元.这里”资源”的定义包括硬件外设和信号量.

5.2模块作用:

系统的资源管理单元,管理系统的信号量(排他性资源)的使用与管理机制.

5.3工作原理

5.3.1登记资源

当登记资源时,系统先去找一个未用过的资源登记块,然后将资源的相关信息写入其中.资源的数目必须大于0,其属性的可取值见后.返回的是该资源的系统内登记号,这个是操作资源的唯一根据.

5.3.2占用资源

先判断要占用的资源数(不能为0,否则这个系统调用就没意义了);然后这个资源必须是系统中已经存在的,而且数目足够用.之后将该进程号登记在资源占用列表里(如果是第一次占用的话),而且从资源的总数里除去占用数.

5.3.3除去资源

系统先判断该资源在系统中是否存在(不存在则出错退出),然后再判断该资源是否被占用.如果是不能从系统中除去,如果不是则除去它.

5.3.4 释放资源

先判断该资源在系统中是否存在;如果存在,该进程是否有资格释放(他在历史上是否占用过该资源,占用数目是否多于释放数目).如果否,出错退出,如果是则释放资源.若该进程此次释放了它占用的全部资源,那么退出时它将会从资源占用列表里消失.

5.3.5释放全部资源

扫描资源登记表,并将该进程占用的全部资源释放.

5.3.6询问资源数量

传入资源的登记号,返回值是询问的该资源的现存数量.

5.4关键函数介绍

5.4.1 u8 Sys\_Register\_Resource(u8 Resource\_Type,u8 Number\_Of\_Resources)

在操作系统中登记一个资源,类型为Resource\_Type,其可占用数目为Number\_Of\_Resources.这个资源可以是实际的有具体数目的外设(比如定时器等等),也可以是一个”空”资源,此时它是一个信号量.本函数返回值是Resource\_ID,即在系统中登记的资源号.如果此次登记不成功,他会返回一个大于MAX\_NUMBER\_OF\_RES(系统中最大允许资源数)的值(为其+1)

失败的原因可能是:

在系统中登记的资源数目已达到最大值.此时会返回失败信息.

@入口参数:

[IN]u8 Resource\_Type:将要进行登记的资源类型,各个位的意义如下:

7 6 5 4 3 2 1 0

EXISTING READ WRITE CHIP\_BOARD Device Class[3 : 0]

EXIST [0x80] 标志这个资源被登记.

DEV\_R [0x40] 标志此设备可读

DEV\_W [0x20] 标志此设备可写

ON\_CHIP [0x10] 标志此设备在片上

ON\_BOARD [0x00] 标志此设备在板上

DUMMY [0x00] 标志此设备不是一个真的设备

SRAM [0x01] 标志此设备是一个SRAM

DRAM [0x02] 标志此设备是一个DRAM

SERIAL\_FLASH [0x03] 标志此设备是一个串行Flash

NOR\_FLASH [0x04] 标志此设备是一个Nor Flash

NAND\_FLASH [0x05] 标志此设备是一个Nand Flash

INDICATOR [0x06] 标志此设备是一个指示设备,如灯.

MISC\_INPUT [0x07] 标志此设备是一个输入设备,比如键盘,传感器,触摸屏,DAC等等.

MISC\_OUTPUT [0x08] 标志此设备是一个输出设备,比如逻辑IC,串口与ADC等等.

LCD\_TFT [0x09] TFTLCD液晶屏.

LCD\_12864 [0x0A] 12864液晶屏.

LCD\_1602 [0x0B] 1602 液晶屏

CPU [0x0C] 另一个CPU.必须是与之对等的CPU(如果

一个CPU被用在键盘上,专门用来扫描键盘,则不属于对等地位,应放在MISC\_INPUT类中.)

CPLD [0x0D] 标志此设备是一个CPLD

FPGA [0x0E] 标志此设备是一个FPGA

ANALOG [0x0F] 标志此设备是一个模拟设备.

[IN]u8 Number\_Of\_Resources

@出口参数:

[OUT]u8 如果成功,返回资源号;如果不成功,返回”ENORCB”,即一个比系统中可能的最大资源号大1的数.

@用户调用:允许.

5.4.2 u8 Sys\_Remove\_Resource(u8 Resource\_ID)

在操作系统中去除一个资源,其资源号为Resource\_ID.返回值是0则表示成功,返回值是”ENORES” (0x01)则表示失败.

失败的原因可能是:

试图释放的资源不存在;

试图释放的资源有别的进程(也许就是这个进程自身)占用.

@入口参数:

[IN]u8 Resource\_ID:欲去除的资源的资源号.

@出口参数:

[OUT]u8 操作是否成功,返回0表示成功,返回1表示失败.

@用户调用:允许

5.4.3 u8 Sys\_Occupy\_Resource(u8 Resource\_ID,u8 Number\_To\_Occupy)

占用一种指定资源,数量为Number\_To\_Occupy.占用失败返回”ENORES” (0x01).

失败的原因可能是:

欲占用资源数为0(这个调用整个是一个恶搞?!)

欲占用的资源系统中不存在

系统中资源不够

@入口参数:

[IN] u8 Resource\_ID:欲占用的资源的资源号.

[IN]u8 Number\_To\_Occupy:占用资源的个数.

@出口参数:

[OUT]u8 操作是否成功,如果成功返回0,否则为1.

@用户调用:允许

还有一个系统专用的\_Sys\_Occupy\_Resource,用户不允许调用.

5.4.4 u8 Sys\_Release\_Resource(u8 Resource\_ID,u8 Number\_To\_Release)

释放一种指定资源,数量为Number\_To\_Occupy.释放失败返回”ENORES” (0x01).

失败的原因可能是:

欲释放资源数为0(这个调用整个是一个恶搞?!)

欲释放的资源系统中不存在

这个进程没有释放的必要或者权限(它从未申请过该资源的使用)

@入口参数:

[IN] u8 Resource\_ID:欲释放的资源的资源号.

[IN]u8 Number\_To\_Release:释放资源的个数.

@出口参数:

[OUT]u8 操作是否成功,如果成功返回0,否则为1.

@用户调用:允许

还有一个系统专用的\_Sys\_Release\_Resource,用户不允许调用.

5.4.5 void Sys\_Release\_All\_Resources(void)

释放当前进程占有的所有资源.

@入口参数:无

@出口参数:无

@用户调用:允许

5.4.6 void \_Sys\_Release\_All\_Resources(u8 PID)

操作系统内用函数,释放进程号为PID的进程占有的所有资源.

@入口参数:

[IN]u8 PID:欲释放所有资源的进程的进程号.

@出口参数:无

@用户调用:禁止

5.4.7 u8 Sys\_Query\_Amount(u8 Resource\_ID)

询问某一资源的剩余数量.这个函数没有也不需要原子操作保护.

@入口参数:

[IN]u8 Resource\_ID:欲询问的资源的资源号

@出口参数:

[OUT]u8 :资源的剩余数量.若该资源在系统中不存在,则返回值也是0.

@用户调用:允许

5.4.8 void Sys\_Register\_SIGCUSTOM\_Handler(void (\*Func)(void))

在系统中给本进程自身登记SIGCUSTOM的处理函数

@入口参数:

[IN] void (\*Func)(void) [OUT]u32 Bytes:欲登记的处理函数,必须无返无参.

@出口参数:无

@用户调用:允许

6文件系统管理单元(仍不完善)

6.1模块名称<filesystem.c>,头文件<filesystem.h>

6.2模块作用:

系统的文件系统管理单元.提供文件系统管理功能,也提供文件的管理/操作功能.

特性:

6.2.1文件系统可以自由挂载和卸载

6.2.2一个进程可以使用多个文件系统,多个进程也可以使用一个文件系统,但不允许同时读写同一个文件.

6.2.3同一时刻一个盘符下只能挂载一个文件系统.如需换掉,必须先解挂.

6.3工作原理

6.3.1挂载文件系统

当一个文件系统被挂载时,首先由用户输入欲挂载的根目录,文件系统在介质上的绝对起始地址,文件系统在资源管理系统中的资源号,以及文件系统的各个操作函数(由用户Sys\_Malloc分配一块内存之后,填充结构,并且把指针交给它).系统会先去判断是否有文件系统挂载在同一个盘符下,如果是,则出错退出;然后再看有没有管理块可以挂载该文件系统,如果无则退出,有则将各个函数填充在该管理块的指定区域内.至此挂载完成.

挂载时系统会自动占用一个资源,不要在挂载前手动占用这个资源.

6.3.2打开文件相关的(即涉及路径的系统调用)

当打开文件被进行时,先调用”Sys\_Fopen”,传入打开文件的路径和模式.系统先判断是否有文件系统挂载在该盘符下,如果没有,则出错退出.然后将打开文件的进程的进程号储存在该文件系统的管理块内.即使该进程已经打开了一个文件(即它的进程号已在使用清单中被登记),还要再登记一次,因为这个文件被打开了同一个文件在同一个时间只能被同一个进程打开一次.这些特性中后几个必须由文件系统提供的读写函数完成.

之后会去找一个IO\_Buffer,即缓冲区控制块,若找不到可用控制块(也即打开的文件数目已达上限,不能再打开),也出错退出.接下来调用该文件系统对应的”\_Sys\_XXFS\_Fopen”函数,将任务交给文件系统提供的”\_Sys\_XXFS\_Fopen”.他负责按Sys\_Fopen中传给他的参数u8 Root, void\* Start\_Address,u8 Resource\_ID与struct File\_Operations\* Operations进行操作,并且如果打开成功还要为文件系IO\_Buffer分配内存.

在打开文件时,由系统提供的服务包括:

1. 根据路径中的盘符找到对应的FSCB(如果找不到将会出错退出),并将Absolute\_Start\_Address提供给用户函数;
2. 找到一个未用过的IOBCB位置并提供给用户函数,填充由用户函数完成;
3. 调用对应的用户提供的函数,并且如果成功,将用户的PID填进登记表中;
4. 根据用户函数调用是否成功决定返回值.

其余的功能(与文件系统内部结构相关的)必须由用户的程序自行提供.

6.3.3操作文件相关的(即操作打开的文件的)

当其他文件操作被进行时,系统会根据IOBCB中的盘符信息找到相应的FSCB,然后再将Absolute\_Start\_Address与IOBCB编号一起提供给用户函数.相应的用户函数完成剩下的操作.

系统提供的服务包括:

1. 根据IOBCB中的盘符找到相应的FSCB;
2. 将Absolute\_Start\_Address与IOBCB编号和其他参数一起在调用时传给用户函数.
3. 根据用户函数调用是否成功决定返回值.

6.3.4 卸载文件系统

系统首先根据” Sys\_Inner\_Fopen”是否存在判断该文件系统是否被挂载,如果是,再判断是否有进程在使用这个文件系统;如果没有,系统自动释放资源占用列表里的对应资源,并且将整个对应的FSCB清空,卸载完成.

6.4关键函数介绍

6.4.1 s8 Sys\_Mount\_FS(u8 Root,void\* Start\_Address,u8 Resource\_ID,struct

File\_Operations\* Operations)

文件系统挂载函数,挂载一个文件系统..

@入口参数:

[IN]u8 Root:该文件系统将要被挂载到的根目录名.只能是一个字符.

[IN]u32 Start\_Address:该文件系统在其存储介质上的绝对起始地址.

[IN]u8 Resource\_ID:该文件系统在资源管理单元中申请的资源号

[IN] struct File\_Operations\* Operations:文件系统的操作函数集,由结构体传入.

该结构体包含的要传入的参数(函数指针)如下.这些函数指针应该由挂载文件系统者提供:

这些函数在文件系统设计时必须符合如下特性:

6.4.1.1 u8 (\*Sys\_This\_FS\_Fopen)(u8\* Path,u8 Mode,u8 IO\_Buffer, void\*

Absolute\_Start\_Address)

这个文件系统的Fopen函数.

@入口参数:

[IN]u8\* Path:欲打开的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

[IN]u8 Mode:打开文件的模式,它包括以下四种:

READ [0x01]

WRITE [0x02]

ADD [0x04]

CREATE [0x08]

几种模式可以用”|”相接表示这两种模式同时被选中,如READ|WRITE表示既可读又可写.

[IN]u8 IO\_Buffer:表示Sys\_Fopen找到的将要交给它的流缓冲区编号.

[IN] void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,否则返回1.

用户调用:禁止

是否需要原子操作保护:不需要(见上面的文件系统工作原理)

6.4.1.2 u8(\*Sys\_This\_FS\_Fread)(void\* Buffer,u8 Size,u32 Count,u8 IO\_Buffer, void\* Absolute\_Start\_Address)

这个文件系统的Fread函数.

@入口参数:

[OUT]void\* Buffer:指向所分配的内存缓冲区的指针.是接收数据的地址.

[IN]u8 Size:要读取的单个元素的大小.单位字节.

[IN]u32 Count:要读取的元素个数.

[IN]u8 IO\_Buffer:打开的文件的文件缓冲区号

[IN]void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,不成功1.

用户调用:禁止

是否需要原子操作保护:不需要.

6.4.1.3 u8(\*Sys\_This\_FS\_Fwrite)(void\* Buffer,u8 Size,u32 Count,u8 IO\_Buffer, void\* Absolute\_Start\_Address)

这个文件系统的Fwrite函数.

@入口参数:

[IN]void\* Buffer:指向所分配的内存缓冲区的指针.是将要写入文件的数据的地址.

[IN]u8 Size:要写入的单个元素的大小.单位字节.

[IN]u32 Count:要顺序写入的元素个数.

[IN]u8 IO\_Buffer:打开的文件的文件缓冲区号

[IN]void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,不成功1.

用户调用:禁止

是否需要原子操作保护:不需要.

6.4.1.4 u8 (\*Sys\_This\_FS\_Fclose)(u8 IO\_Buffer,void\*

Absolute\_Start\_Address)

这个文件系统的Fclose函数.

@入口参数:

[IN]u8 IO\_ Buffer: 将要关闭的,现在打开的文件的文件缓冲区号.

[IN]void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,不成功1.

用户调用:禁止

是否需要原子操作保护:不需要.

6.4.1.5 u8 (\*Sys\_This\_FS\_Fseek)(u8 IO\_Buffer,s32 Offset,u8 Position, void\*

Absolute\_Start\_Address)

这个文件系统的Fseek函数.

@入口参数:

[IN]u8 IO\_Buffer: 将被操作的打开的文件的文件缓冲区号.

[IN]s32 Offset:要移动文件读写指针的长度,单位字节.正数地址增加,负数地址减少.

[IN]u8 Position:从文件的哪个地方开始.是下列值之一:

SEEK\_SET [0x00] 从文件开头开始

SEEK\_CUR [0x01] 从文件指针现有位置开始.

SEEK\_END [0x02] 从文件结尾开始.

[IN]void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,不成功1.

用户调用:禁止

是否需要原子操作保护:不需要.

6.4.1.6 u8 (\*Sys\_This\_FS\_Frename)(u8\* Path,u8\* New\_Name,void\*

Absolute\_Start\_Address);

这个文件系统的Frename函数.

@入口参数:

[IN]u8\* Path:欲重命名的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

[IN]u8\* New\_Name: 欲重命名的文件的新名字.包括文件名但不包括盘符.它最长不应该超过系统中所允许的最长路径名-2(即LONGEST\_PATH宏-2)

[IN]void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,否则返回1.

用户调用:禁止

是否需要原子操作保护:不需要

6.4.1.7 u8 (\*Sys\_This\_FS\_Fremove)(u8\* Path,void\*

Absolute\_Start\_Address);

这个文件系统的Fremove函数.

@入口参数:

[IN]u8\* Path:欲删除的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

[IN]void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,否则返回1.

用户调用:禁止

是否需要原子操作保护:不需要

6.4.1.8 u8(\*Sys\_This\_FS\_Get\_Fproperty)(u8\* Path,struct THISFS\_File\_Property\* Property,void\* Absolute\_Start\_Address)

这个文件系统的文件属性读取函数.

@入口参数:

[IN]u8\* Path:欲读取属性的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

[OUT]struct THISFS\_File\_Property\* Property:文件的属性结构指针,传出值.

文件属性结构在各文件系统typedef文件的里面定义.

[IN]void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,否则返回1.

用户调用:禁止

是否需要原子操作保护:不需要

6.4.1.9 u8(\*Sys\_This\_FS\_Set\_Fproperty)(u8\* Path,struct THISFS\_File\_Property\* Property,void\* Absolute\_Start\_Address)

这个文件系统的文件属性设置函数.

@入口参数:

[IN]u8\* Path:欲重命名的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

[IN]struct THISFS\_File\_Property\* Property:文件的属性结构指针,传入值.

文件属性结构在各文件系统typedef文件的里面定义.

[IN]void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,否则返回1.

用户调用:禁止

是否需要原子操作保护:不需要

6.4.1.10 u8 (\*Sys\_This\_FS\_Format)(void\* Absolute\_Start\_Address)

这个文件系统的低级格式化函数.

@入口参数:

[IN]void\* Absolute\_Start\_Address:文件系统在介质上的绝对起始地址.

@出口参数:

[OUT]u8 :成功返回0,否则返回1.

用户调用:禁止

是否需要原子操作保护:不需要

@出口参数:

[OUT]s8 :如果成功,返回文件系统的识别码;如果不成功,返回-1.

@用户调用:允许.

6.4.2u8 Sys\_Unmount\_FS(u8 FS\_ID)

卸载一个文件系统.

@入口参数:

[IN]u8 FS\_ID:文件系统的识别码.

@出口参数:

[OUT]u8 :成功返回0,不成功返回1.

失败的原因可能是:

这个文件系统根本没有被挂载;

还有进程在使用这个文件系统.

@用户调用:允许.

6.4.3 u8 Sys\_Low\_Level\_Format\_FS(u8 Root)

低级格式化一个文件系统.

@入口参数:

[IN]u8 Root:盘符.这个文件系统必须已经被挂载.

@出口参数:

[OUT]u8 :成功返回0,不成功返回1.

@用户调用:允许.

6.4.4 u8 Sys\_Fopen(u8\* Path,u8 Mode)

打开一个文件.

@入口参数:

[IN]u8\* Path:欲重命名的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

[IN]u8 Mode:打开文件的方式.包括以下四种:

READ [0x01]

WRITE [0x02]

ADD [0x04]

CREATE [0x08]

几种模式可以用”|”相接表示这两种模式同时被选中,如READ|WRITE表示既可读又可写.

@出口参数:

[OUT]u8 :成功返回文件的流缓冲控制块代号,不成功返回比系统中允许打开的文件的个数大1的值.

@用户调用:允许.

6.4.5 u8 Sys\_Fread(void\* Buffer,u8 Size,u32 Count,u8 IO\_Buffer)

从文件中读出一些数据.

@入口参数:

[OUT]void\* Buffer:指向数据缓冲区的指针.用来接收数据.

[IN]u8 Size:欲顺序读出的单个数据块大小,单位字节.

[IN]u32 Count:欲读出的数据块个数

[IN]u8 IO\_Buffer:该打开的文件的文件缓冲区控制块编号.

@出口参数:

[OUT]u8 :0成功,1失败.

失败的原因可能是:

@用户调用:允许

6.4.6 u8 Sys\_Fwrite(void\* Buffer,u8 Size,u32 Count,u8 IO\_Buffer)

向文件中写入一些数据.

@入口参数:

[OUT]void\* Buffer:指向数据缓冲区的指针.用来存储准备写入的数据.

[IN]u8 Size:欲顺序写入的单个数据块大小,单位字节.

[IN]u32 Count:欲写入的数据块个数

[IN]u8 IO\_Buffer:该打开的文件的文件缓冲区控制块编号.

@出口参数:

[OUT]u8 :0成功,1失败.

失败的原因可能是:

@用户调用:允许

6.4.7 u8 Sys\_ Fclose(u8 IO\_Buffer)

关闭一个打开的文件.

@入口参数:

[IN]u8 IO\_ Buffer: 将要关闭的,现在打开的文件的文件缓冲区号.

@出口参数:

[OUT]u8 :成功返回0,不成功1.

用户调用:允许

6.4.8 u8 Sys\_Fseek(u8 IO\_Buffer,s32 Offset,u8 Position)

文件读写指针移动函数

@入口参数:

[OUT]u8 IO\_Buffer: 该打开的文件的文件缓冲区控制块编号.

[IN]s32 Offset:要移动文件读写指针的长度,单位字节.正数地址增加,负数地址减少.

[IN]u8 Position:从文件的哪个地方开始.是下列值之一:

SEEK\_SET [0x00] 从文件开头开始

SEEK\_CUR [0x01] 从文件指针现有位置开始.

SEEK\_END [0x02] 从文件结尾开始.

@出口参数:

[OUT]u8 :0成功,1失败.

失败的原因可能是:

@用户调用:允许

6.4.9 u8 Sys\_Frename(u8\* Path,u8\* New\_Name)

重命名一个文件.

@入口参数:

[IN]u8\* Path:欲重命名的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

[IN]u8\* New\_Name:新的文件名,不包括文件的盘符(路径)

@出口参数:

[OUT]u8 :0成功,1失败.

失败的原因可能是:

@用户调用:允许

6.4.10 u8 Sys\_Fremove(u8\* Path)

删除一个文件.

@入口参数:

[IN]u8\* Path:欲删除的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

@出口参数:

[OUT]u8 :0成功,1失败.

失败的原因可能是:

@用户调用:允许

6.4.11 u8 Sys\_Get\_Fproperty(u8\* Path,void\* Property)

文件属性读取函数.

@入口参数:

[IN]u8\* Path:欲读取属性的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

[OUT]void\* Property:文件的属性结构指针,传出值.

文件属性结构在各文件系统typedef文件的里面定义.这里统一用void\*代替.

@出口参数:

[OUT]u8 :成功返回0,否则返回1.

失败的原因可能是:

用户调用:允许

6.4.12 u8 Sys\_Set\_Fproperty(u8\* Path,void\* Property)

文件属性设置函数.

@入口参数:

[IN]u8\* Path:欲重命名的文件的路径.包括盘符和文件名.它最长不应该超过系统中所允许的最长路径名(即LONGEST\_PATH宏)

[IN]void\* Property:文件的属性结构指针,传入值.

文件属性结构在各文件系统typedef文件的里面定义. 这里统一用void\*代替.

@出口参数:

[OUT]u8 :成功返回0,否则返回1.

失败的原因可能是:

用户调用:允许

6.4.13 u8 Sys\_Low\_Level\_Format\_FS(u8 Root)

文件系统的低级格式化函数.

@入口参数:

[IN]u8 Root:文件系统挂载的根目录.

@出口参数:

[OUT]u8 :成功返回0,否则返回1.

用户调用:允许

6.4.14 u8 \_Sys\_Fclose(u8 IO\_Buffer)

特点与Sys\_Fclose相同,只不过没有原子操作保护,而且不能在用户程序中调用.

@入口参数:

[IN]u8 IO\_ Buffer: 将要关闭的,现在打开的文件的文件缓冲区号.

@出口参数:

[OUT]u8 :成功返回0,不成功1.

用户调用:禁止

6.4.15 void \_Sys\_Close\_All\_Files(u8 PID)

关闭一个进程打开的所有文件.操作系统内用函数,用户不能调用.

@入口参数:

[IN]u8 PID:欲关闭所有打开文件的进程的进程号.

@出口参数:无

用户调用:禁止

6.5自带的几个文件系统:(仍未完全完成)

6.5.1综述

本操作系统自带3个文件系统,分别为NOFS,RAMFS与ROMFS.

NOFS是最简化的文件系统(No File System),只支持文件的读写,文件没有属性,文件名为固定的数字,而且定长,相当于储存在一块介质上的大数组.

RAMFS是用于RAM的文件系统(Random Access Memory File System),充分考虑了RAM小的特性而设计,不支持长文件名与文件夹.但支持文件属性,文件变长.

ROMFS是用于EEPROM/小容量Flash的文件系统,能支持长文件名与(理论上)无限级文件夹,支持恢复记录和文件压缩存储(整个过程对用户透明).其附加功能是可以配置的,最基本的ROMFS与RAMFS基本相同.

6.5.2 NOFS

此系统的文件定长,并且文件的名称和个数也固定.文件的数目不能超过10.

文件的数量由” FILES\_IN\_FS”指定,长度由” FILE\_LENGTH”指定.

文件名固定为文件序号,从0到9.

这个文件系统不支持文件属性,也不支持只读打开.

这个文件系统无法判断打开的文件是哪个进程打开的,而且文件一旦被打开多个进程都可以来对其进行读写,也可以由其他的进程来关闭这个文件.所以,这个文件系统仅仅推荐用于单个进程操作文件系统的情况.

6.5.3 RAMFS(未完成)

6.5.4.ROMFS(未开工)

7 杂项系统调用

7.1模块名称<miscellaneous.c>,头文件<miscellaneous.h>

7.2模块作用:

系统的杂项系统调用模块.提供一些最基本的所有操作系统都有的小功能,也提供一部分为本操作系统设计的C函数库函数(标准C函数库对嵌入式领域有时太臃肿.).

7.3工作原理

这一个模块的各个函数没有必然的联系,他们是分立的功能集合.这一块的大多数函数均可由用户调用.

8 命令行界面

8.1模块名称<shell.c>,<shell\_arch.c>头文件<shell.h>,<shell\_arch.h>

8.2模块作用:

系统的命令行界面.提供命令行功能,可以作为控制台使用.

8.3工作原理

对用户输入的命令进行解析,判断该命令的含义,并且进行执行.能解析的命令:

8.3.1 ls (list)

命令含义:列出当前文件夹下所有文件.

参数:无(目前)

8.3.2 pwd (print working directory)

命令含义:打印出当前的工作目录.

参数:无(目前)

8.3.3 du (disk usage)

命令含义:打印出磁盘的使用信息,包括已用空间,可用空间,等等.

参数: -[Disk] 该磁盘的盘符.

8.3.4 mkdir (make directory)

命令含义:在当前工作目录下创建一个子文件夹.

参数: [Name] 该新建的文件夹的名字

8.3.5 find (find)

命令含义:在全部磁盘文件中寻找一个文件夹或文件.

参数: [Name] 该要寻找的文件(夹)的名字.

8.3.6 cp (copy)

命令含义:复制磁盘文件.

参数: [Name1] [Name2]

[Name1]是源文件名(包含相对路径)

[Name2]是新的目标文件的名称(包含相对路径)

8.3.7 mv (move)

命令含义:转移磁盘文件.

参数: [Name1] [Name2]

[Name1]是源文件名(包含相对路径)

[Name2]是新的目标文件的名称(包含相对路径)

8.3.8 rm (remove)

命令含义:删除磁盘文件

参数: [Name] 是欲删除的文件名(包含相对路径)

8.3.9 edit (editor)

命令含义:打开文本编辑器.

参数: [Name] 是欲创建或编辑的文件的名字.

8.3.10 top ()

命令含义:查看系统的工作状况,类似于Windows的进程管理器.

参数:无

8.3.11 quit (quit)

命令含义:退出shell.

参数:无

8.4关键函数介绍:<shell\_arch.c>

8.5关键函数介绍:<shell.c>

9 用户图形界面

9.1模块名称<GUI.c>,<GUI\_arch.c>,头文件<GUI.h>,<GUI\_arch.h>,<GUI\_font.h>

其中GUI.c 存放与体系结构无关的代码,GUI\_arch.c存放与体系结构有关的代码,

GUI.h与GUI\_arch.h分别是他们的头文件. GUI\_font.h存放字模.

9.2模块作用:

系统的用户图形界面.可以作为控制台使用.

9.3工作原理:

9.3.1图形的绘制

绘制图形时,由GUI.c中的上层函数调用GUI\_arch.c中的基本函数之一

”Sys\_GUI\_LCD\_Set\_Point”,这个函数可以完成基本的画点动作.由这些点组成欲绘制的图案.GUI\_arch.c中能被GUI.c调用的函数是极其有限的.

9.3.2窗口的管理

窗口的属性由DCB(Dialog Box Control Block)记录和管理.DCB中的每一个元素代表一个窗口,该元素在DCB中的下标即为窗口的ID.窗口的叠加顺序(或说显示顺序)是由DCB内部组织成的双向循环链表(Rookie 内核链表,与Linux的完全相同)控制的.在链表中越靠近链表头就意味着整体刷新窗口时刷新顺序越靠后,这样便显示在越前端.(参见” Sys\_GUI\_Refresh\_All\_Dialog\_Boxes”).窗口的属性包括位置,大小,是否固定,是否置于最前端,等等.这些属性在绘制窗口时将传给绘制函数,由它们绘制窗口.

9.3.3控件的管理

控件的管理是分窗口进行的.在每个窗口对应的DCB元素中都有一个该窗口的CCB(Control Control Block).他们存放指向控件对应的结构中共有部分的指针.DCB中还记载这个窗口一共有多少个控件.

每个控件都必须有的属性记载在他们结构开头的的共有部分中,包括类别,所属的窗口ID,以及名称.各个控件的特有属性记载在它们各自结构的独特部分中,具体结构参见相关文件.

9.3.4输入的响应

在屏幕上点击输入给输入分析子系统的是一个坐标(Coord\_X,Coord\_Y).这个坐标在经过一级分析后得到所点击的窗口的ID.经二级分析后得到所点击的窗口中的控件ID.如果这个控件是一个选项卡,那么还要经三级分析判断是点到了选项卡的头,选项卡的空白地带还是选项卡中的控件,得到控件ID.然后这(二)三级ID被传给相关函数,执行早已存放在被点击控件的结构体中的动作函数,完成响应.(具体机制参见GUI.c.)重绘窗口/控件的工作交给该函数去做,系统不会自动完成.

在判断被点击到的窗口时,按链表的顺序从前向后搜索,取第一个被搜索到的值.这样,被搜索到的窗口必定是该位置上位于最上方的窗口.

9.4关键函数介绍:<GUI\_arch.c>

9.4.1 u16 Sys\_GUI\_LCD\_Init(void)

液晶屏的初始化函数,在使用液晶屏前至少需调用一次.这个函数会被选择性编译,具体编译哪一段由”CONTROLLER\_ID”宏决定.

@入口参数:无

@出口参数:

[OUT]u16 :返回液晶屏的型号,也即”CONTROLLER\_ID”宏的值.

用户调用:允许

9.4.2 void Sys\_GUI\_LCD\_Clear\_Screen(u16 Color)

液晶屏的快速清屏函数.这个函数没有使用”Sys\_GUI\_LCD\_Set\_Point”,而是根据液晶屏本身的特点编写,可以更快速的完成刷屏.

@入口参数:

[IN]u16 Color:清屏时使用的填充颜色.

@出口参数:无

用户调用:允许

9.4.3 void Sys\_GUI\_LCD\_Set\_Point(u16 Coord\_X,u16 Coord\_Y,u16 Color)

在液晶屏上画出一个像素点.

@入口参数:

[IN]u16 Coord\_X:该点的X坐标.

[IN]u16 Coord\_Y:该点的Y坐标.

[IN]u16 Color:该点的颜色值.

@出口参数:无

@用户调用:允许.

9.4.4 u16 Sys\_GUI\_LCD\_Get\_Point(u16 Coord\_X,u16 Coord\_Y)

得到液晶屏上某一像素点的颜色值.

@入口参数:

[IN]u16 Coord\_X:该点的X坐标.

[IN]u16 Coord\_Y:该点的Y坐标.

@出口参数:

[OUT]u16 :该点的颜色值.

@用户调用:允许.

9.4.5 void Sys\_GUI\_LCD\_Back\_Light(u8 Status)

开关液晶屏的背光.

@入口参数:

[IN]u8 Status:背光的状态.如果不为0则开灯,为0则关灯.

@出口参数:无

@用户调用:允许

9.5关键函数介绍:<GUI.c>

9.5.1void Sys\_GUI\_LCD\_Draw\_Line(u16 Start\_Coord\_X, u16 Start\_Coord\_Y,

u16 End\_Coord\_X, u16 End\_Coord\_Y,u16 Color)

在屏幕上划出一条线.

@入口参数:

[IN]u16 Start\_Coord\_X:起始点的X坐标

[IN]u16 Start\_Coord\_Y:起始点的Y坐标

[IN]u16 End\_Coord\_X:终了点的X坐标

[IN]u16 End\_Coord\_Y:终了点的Y坐标

[IN]u16 Color:该线的颜色.

@出口参数:无

@用户调用:允许

9.5.2 void Sys\_GUI\_LCD\_Print\_Char(u16 Coord\_X,u16 Coord\_Y,u8 Char,

u8 Size,u16 Char\_Color,u16 Background\_Color)

在液晶屏上画出一个字符.

@入口参数:

[IN]u16 Coord\_X:该字符的左上角X坐标.

[IN]u16 Coord\_Y:该字符的左上角Y坐标.

[IN]u8 Char:字符的ASCII码.

@出口参数:

[OUT]u8 :成功返回0,不成功1.

用户调用:允许

9.5.3 void Sys\_GUI\_LCD\_Print\_String(u16 Coord\_X,u16 Coord\_Y,u8\* String,u8 Size,

u16 Length,u16 Char\_Color,u16 Background\_Color)

在液晶屏上显示一行字符.字符的大小可以变化.

@入口参数:

[IN]u16 Coord\_X:该字符的左上角X坐标.

[IN]u16 Coord\_Y:该字符的左上角Y坐标.

[IN]u8\* String:该字符串.

[IN]u8 Size:字符的大小,可以选择”\_8X16\_”和”\_16X24\_”两种.

[IN]u16 Length:字符串的长度.

[IN]u16 Char\_Color:字符的颜色.

[IN]u16 Background\_Color:字符串的背景色.

@出口参数:无

@用户调用:允许

9.5.4 void Sys\_GUI\_LCD\_GUI\_Polygon(u16 Number\_Of\_Points,

u16 (\*Coord\_Of\_Dots)[2],u16 Color)

在屏幕上画连续的线,这些线可以组成一个多边形.

@入口参数:

[IN]u16 Number\_Of\_Points:点的个数

[IN]u16 (\*Coord\_Of\_Dots)[2]:点的坐标,以一个二维数组存储.

[IN]u16 Color:线的颜色

@出口参数:无

@用户调用:允许

9.5.5 void Sys\_GUI\_LCD\_GUI\_Filled\_Rectangle(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,u16 Border\_Color,u16 Fill\_Color)

画一个被填充完全的长方形.

@入口参数:

[IN]u16 Coord\_X:该字符的左上角X坐标.

[IN]u16 Coord\_Y:该字符的左上角Y坐标.

[IN]u16 Length:长方形的长度

[IN]u16 Width:长方形的宽度

[IN]u16 Border\_Color:长方形边界的颜色.

[IN]u16 Fill\_Color:填充在长方形内部的颜色.

@出口参数:无.

@用户调用:允许

9.5.6 void Sys\_GUI\_LCD\_GUI\_Circle(u16 Center\_Coord\_X,u16 Center\_Coord\_Y,

u16 Radius,u16 Color)

在液晶屏上画一个圆.

@入口参数:

[IN]u16 Center\_Coord\_X:圆心的X坐标.

[IN]u16 Center\_Coord\_Y: 圆心的Y坐标.

[IN] u16 Radius:圆的半径.

[IN]u16 Color:圆外边的颜色

@出口参数:无

@用户调用:允许

9.5.7 void Sys\_GUI\_LCD\_Draw\_Picture(u16 Start\_Coord\_X,u16 Start\_Coord\_Y,

u16 Original\_Length,u16 Original\_Width,u16 \*Picture)

在屏幕上画一幅图片.图片的取模方向应该是从左到右,从上到下.

@入口参数:

[IN]u16 Start\_Coord\_X:左上角的X坐标.

[IN]u16 Start\_Coord\_Y:左上角的Y坐标.

[IN]u16 Original\_Length:原来图片的长度(单位:像素).

[IN]u16 Original\_Width:原图片的宽度(单位:像素).

[IN]u16\* Picture:指向图片的指针

@出口参数:无

用户调用:允许

9.5.8 void Sys\_GUI\_LCD\_Draw\_Dotted\_Line(u16 Start\_Coord\_X, u16 Start\_Coord\_Y,

u16 End\_Coord\_X, u16 End\_Coord\_Y,u16 Color)

在屏幕上画一条虚线.这条线是隔点画点.

@入口参数:

[IN]Start\_Coord\_X:起始点的X坐标

[IN]Start\_Coord\_Y:起始点的Y坐标

[IN]End\_Coord\_X:终了点的X坐标

[IN]End\_Coord\_Y:终了点的Y坐标

[IN]u16 Color:该虚线所画点的颜色.

@出口参数:无

@用户调用:允许

9.5.9 u16 Sys\_GUI\_Color\_Mixer(u16 Color\_1,u16 Color\_2,float Ratio)

系统内置的颜色混合器,可以将两种符合5R6G5B编码的颜色按比例进行混合.

@入口参数:

[IN]u16 Color\_1:参与混合的第一种颜色

[IN]u16 Color\_2:参与混合的第二种颜色

[IN]float Ratio:混合的比例.

@出口参数:无

@用户调用:允许

9.5.10 void \_Sys\_GUI\_Draw\_Dialog\_Title(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,void\* Title,void\* Icon)

画一个对话框的头部.

@入口参数:

[IN]u16 Coord\_X:对话框头部的左上角X坐标

[IN]u16 Coord\_Y:对话框头部的左上角Y坐标

[IN]u16 Length:对话框的头部长度

[IN]u16 Width:对话框头部的宽度

[IN]void\* Title:实际上是一个指向字符串的指针,该字符串是窗口的标题.

[IN]void\* Icon:对话框的图标,其取模方法与图片一致.

@出口参数:无

@用户调用:禁止

9.5.11 void \_Sys\_GUI\_Draw\_Dialog\_Box\_Without\_Title(u16 Coord\_X,u16 Coord\_Y,

u16 Length,u16 Width)

画出对话框的本体.不包含头.

@入口参数:

[IN]u16 Coord\_X:对话框本体左上角的X坐标.

[IN]u16 Coord\_Y:对话框本体左上角的Y坐标.

[IN]u16 Length:对话框的长度.

[IN]u16 Width:对话框的宽度.

@出口参数:无

用户调用:禁止

9.5.12 void Sys\_GUI\_Draw\_Dialog\_Box(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,void\* Title,void\* Icon)

对话框绘制函数.

@入口参数:

[IN]u16 Coord\_X:对话框左上角的X坐标.

[IN]u16 Coord\_Y:对话框左上角的Y坐标.

[IN]u16 Width:对话框的宽度.对话框头的宽度被包含在内.

[IN]void\* Title:对话框的标题

[IN]void\* Icon:对话框的图标

@出口参数:无

用户调用:允许

9.5.13 void Sys\_GUI\_Erase\_Dialog\_Box(u8 Dialog\_ID,u16 Color)

擦除屏幕上的对话框.

@入口参数:

[IN]u8 Dialog\_ID:对话框的ID.

[IN]u16 Color:擦出对话框后欲填充于空缺区域的颜色.

@出口参数:无

用户调用:允许

9.5.14 void Sys\_GUI\_Draw\_Command\_Button(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,u8 Status)

在屏幕上画一个命令按钮.

@入口参数:

[IN]u16 Coord\_X:按钮左上角的X坐标.

[IN]u16 Coord\_Y:按钮左上角的Y坐标.

[IN]u16 Length:按钮的长度.

[IN]u16 Width:按钮的宽度.

[IN]u8 Status:按钮的状态.

@出口参数:

[OUT]u8 :成功返回0,不成功1.

用户调用:允许

9.5.15 void Sys\_GUI\_Draw\_Check\_Box(u16 Coord\_X,u16 Coord\_Y,u8 Status)

在屏幕上画一个复选框.

@入口参数:

[IN]u16 Coord\_X:该复选框左上角的X坐标.

[IN]u16 Coord\_Y:该复选框左上角的Y坐标.

[IN]u8 Status:该复选框的状态.

@出口参数:无

用户调用:允许

9.5.16 void Sys\_GUI\_Draw\_Scroll\_Bar(u16 Coord\_X,u16 Coord\_Y,u16 Length,u16 Width)在屏幕上画一个卷动条.

@入口参数:

[IN]u16 Coord\_X:该卷动条左上角的X坐标.

[IN]u16 Coord\_Y:该卷动条左上角的Y坐标.

[IN]u16 Color:该线的颜色.

@出口参数:无

@用户调用:允许

9.5.17 void Sys\_GUI\_Draw\_Radio\_Button\_Group(u16 Coord\_X,u16 Coord\_Y,

u8 Number,u16\* Relative\_Centers,u8 Status)在屏幕上画一组互相相关的单选按钮组.

@入口参数:

[IN]u16 Coord\_X:该单选按钮组参考点的X坐标.

[IN]u16 Coord\_Y:该单选按钮组参考点的Y坐标.

[IN]u8 Number:该单选按钮组包含的单选按钮的数量.最大为8.

[IN]u16\* Relative\_Centers:该单选按钮组各按钮相对参考点的坐标.按X1,Y1,X2,Y2….Xn,Yn顺序排列.

[IN]u8 Status:该单选按钮组的状态.

@出口参数:无

用户调用:允许

9.5.18 void Sys\_GUI\_Draw\_Tab\_Group(u16 Coord\_X,u16 Coord\_Y, u16 Header\_Width,

u8 Number,u16\* Header\_Lengths,u16 Width\_Of\_Tabs,u8 Status) 在屏幕上画一组互相相关的选项卡组.

@入口参数:

[IN]u16 Coord\_X:该选项卡组的左上角X坐标.

[IN]u16 Coord\_Y:该选项卡组的左上角Y坐标.

[IN]u16 Header\_Width:该选项卡组的选项卡头宽度

[IN]u8 Number: 选项卡组包含的选项卡数目.

[IN]u16\* Header\_Lengths:各选项卡的头部(标签)的长度.选项卡的长度为它们的和.

[IN]u16 Width\_Of\_Tabs:选项卡空白地带的宽度,不包括头.

[IN] u8 Status:该选项卡组的状态.

@出口参数:无

@用户调用:允许

9.5.19 void Sys\_GUI\_Draw\_Mouse(u16 Coord\_X,u16 Coord\_Y,u8 Status)

在屏幕上画一个鼠标.

@入口参数:

[IN]u16 Coord\_X:该鼠标图案(光标)的左上角X坐标.

[IN]u16 Coord\_Y:该鼠标图案(光标)的左上角Y坐标.

[IN]u8 Status:该鼠标的鼠标图案(显示状态).

@出口参数:无

@用户调用:允许

9.5.20 void Sys\_GUI\_Draw\_Selected\_Bar(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,u8 Status)

在屏幕上画一个选项被选中后反白的选中行.

@入口参数:

[IN]u16 Coord\_X:该选中行的左上角X坐标.

[IN]u16 Coord\_Y:该选中行的左上角Y坐标.

[IN]u16 Length:该选中行的长度

[IN]u16 Width:该选中行的宽度

[IN]u8 Status:该选中行的状态

@出口参数:无.

@用户调用:允许

9.5.21 void Sys\_GUI\_Draw\_Progress\_Bar(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,u16 Front\_Beginning\_Color,u16 Front\_End\_Color,

u16 Back\_Color,double Progress)

在液晶屏上画一个进度条.

@入口参数:

[IN]u16 Coord\_X:该进度条的左上角X坐标.

[IN]u16 Coord\_Y:该进度条的左上角Y坐标.

[IN]u16 Length:该进度条的长度

[IN]u16 Width:该进度条的宽度

[IN]u16 Front\_Beginning\_Color:该进度条进度初始时的前景色.

[IN]u16 Front\_End\_Color: 该进度条进度终了时的前景色.

[IN]u16 Back\_Color:该进度条的背景色

[IN]double Progress:该进度条的进度

@出口参数:无

@用户调用:允许

9.5.22 u8 Sys\_GUI\_Create\_Dialog\_Box(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,u8 Status,void\* Title,void\* Icon)

创建一个对话框.这个对话框不会立即显示在屏幕上,要刷新后才会显示.

@入口参数:

[IN]u16 Coord\_X:对话框左上角的X坐标.

[IN]u16 Coord\_Y:对话框左上角的Y坐标.

[IN]u16 Length:对话框的长度(单位:像素).

[IN]u16 Width:对话框的宽度(单位:像素).

[IN]u8 Status:对话框的状态.

[IN]void\* Title:对话框的标题.

[IN]void\* Icon:对话框的图标.

@出口参数:

[OUT]u8 :如果成功会返回该对话框的对话框ID,否则返回”ENODIA”.

用户调用:允许

9.5.23 u8 Sys\_GUI\_Delete\_Dialog\_Box(u8 Dialog\_ID)

删除一个对话框.

@入口参数:

[IN]u8 Dialog\_ID:由” Sys\_GUI\_Create\_Dialog\_Box”返回的对话框ID

@出口参数:

[OUT]u8 :成功为0,失败为”ENODIA”.

@用户调用:允许

9.5.24 u8 \_Sys\_GUI\_Draw\_Tab\_Controls(struct Dialog\_Control\_Head\* Control)

画出一个选项卡中的控件.

@入口参数:

[IN]struct Dialog\_Control\_Head\* Control:指向该选项卡的公用头的指针.

@出口参数:

[OUT]u8 :成功返回1.若对话框存在而无法完成操作,返回”ENODOP”;对话框不存在返回”ENODIA”.

@用户调用:禁止

9.5.25 void \_Sys\_GUI\_Draw\_Dialog\_Title(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,void\* Title,void\* Icon)

画一个对话框的头部.

@入口参数:

[IN]u16 Coord\_X:对话框头部的左上角X坐标

[IN]u16 Coord\_Y:对话框头部的左上角Y坐标

[IN]u16 Length:对话框的头部长度

[IN]u16 Width:对话框头部的宽度

[IN]void\* Title:实际上是一个指向字符串的指针,该字符串是窗口的标题.

[IN]void\* Icon:对话框的图标,其取模方法与图片一致.

@出口参数:无

@用户调用:禁止

9.5.26 void \_Sys\_GUI\_Draw\_Dialog\_Box\_Without\_Title(u16 Coord\_X,u16 Coord\_Y,

u16 Length,u16 Width)

画出对话框的本体.不包含头.

@入口参数:

[IN]u16 Coord\_X:对话框本体左上角的X坐标.

[IN]u16 Coord\_Y:对话框本体左上角的Y坐标.

[IN]u16 Length:对话框的长度.

[IN]u16 Width:对话框的宽度.

@出口参数:无

用户调用:禁止

9.5.27 void Sys\_GUI\_Draw\_Dialog\_Box(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,void\* Title,void\* Icon)

对话框绘制函数.

@入口参数:

[IN]u16 Coord\_X:对话框左上角的X坐标.

[IN]u16 Coord\_Y:对话框左上角的Y坐标.

[IN]u16 Width:对话框的宽度,包括了对话框头宽度.

[IN]void\* Title:对话框的标题.

[IN]void\* Icon:对话框的图标.

@出口参数:无

用户调用:允许

9.5.28 void Sys\_GUI\_Erase\_Dialog\_Box(u8 Dialog\_ID,u16 Color)

擦除屏幕上的对话框.

@入口参数:

[IN]u8 Dialog\_ID:对话框的ID.

[IN]u16 Color:擦出对话框后欲填充于空缺区域的颜色.

@出口参数:无

用户调用:允许

9.5.29 void Sys\_GUI\_Draw\_Command\_Button(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,u8 Status)

在屏幕上画一个命令按钮.

@入口参数:

[IN]u16 Coord\_X:按钮左上角的X坐标.

[IN]u16 Coord\_Y:按钮左上角的Y坐标.

[IN]u16 Length:按钮的长度.

[IN]u16 Width:按钮的宽度.

[IN]u8 Status:按钮的状态.

@出口参数:

[OUT]u8 :成功返回0,不成功1.

用户调用:允许

9.5.30 void Sys\_GUI\_Draw\_Check\_Box(u16 Coord\_X,u16 Coord\_Y,u8 Status)

在屏幕上画一个复选框.

@入口参数:

[IN]u16 Coord\_X:该复选框的X坐标.

[IN]u16 Coord\_Y:该复选框的Y坐标.

[IN]u8 Status:该复选框的状态.

@出口参数:无

用户调用:允许

9.5.31 void Sys\_GUI\_Draw\_Scroll\_Bar(u16 Coord\_X,u16 Coord\_Y,u16 Length,u16 Width)

在屏幕上画一个卷动条.

@入口参数:

[IN]u16 Coord\_X:该卷动条左上角的X坐标

[IN]u16 Coord\_Y:该卷动条左上角的Y坐标

[IN]u16 Length:该卷动条的长度.

[IN]u16 Width:该卷动条的宽度.

@出口参数:无

@用户调用:允许

9.5.32 void Sys\_GUI\_Draw\_Radio\_Button\_Group(u16 Coord\_X,u16 Coord\_Y,

u8 Number,u16\* Relative\_Centers,u8 Status)画出一组单选框.

@入口参数:

[IN]u16 Coord\_X:该单选框组的左上角X坐标.

[IN]u16 Coord\_Y:该单选框组的左上角Y坐标.

[IN]u8 Number:该单选框组中单选框的个数.

[IN]u16\* Relative\_Centers:单选框中的各个单选框相对于参考点的位置.按{X1,Y1,X2,Y2,…,Xn,Yn}的方式存储.

[IN]u8 Status:单选框的状态.

@出口参数:无

用户调用:允许

9.5.33 void Sys\_GUI\_Draw\_Tab\_Group(u16 Coord\_X,u16 Coord\_Y, u16 Header\_Width,

u8 Number,u16\* Header\_Lengths,u16 Width\_Of\_Tabs,u8 Status)画出一组选项卡.

@入口参数:

[IN]u16 Coord\_X:该选项卡组左上角X坐标.

[IN]u16 Coord\_Y:该选项卡组左上角Y坐标.

[IN]u16 Header\_Width:该选项卡的头宽度.

[IN]u8 Number:该选项卡组包含的选项卡的个数.

[IN]u16\* Header\_Lengths:该选项卡组中各个选项卡的头长度.总的选项卡长度是它们的和.

[IN]u16 Width\_Of\_Tabs:选项卡宽度.不包括选项卡头.

[IN]u8 Status:该选项卡组的现有状态.

@出口参数:无

@用户调用:允许

9.5.34 void Sys\_GUI\_Draw\_Mouse(u16 Coord\_X,u16 Coord\_Y,u8 Status)

在屏幕上画一个鼠标光标.

@入口参数:

[IN]u16 Coord\_X:该鼠标光标左上角X坐标.

[IN]u16 Coord\_Y:该鼠标光标左上角Y坐标.

[IN]u8 Status:鼠标光标的样式.

@出口参数:无

@用户调用:允许

9.5.35 void Sys\_GUI\_Draw\_Selected\_Bar(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,u8 Status)

在屏幕上画一个选中的反色框.

@入口参数:

[IN]u16 Coord\_X:该反色框的左上角X坐标.

[IN]u16 Coord\_Y:该反色框的左上角Y坐标.

[IN]u16 Length:反色框的长度.

[IN]u16 Width:反色框的宽度.

[IN]u8 Status:反色框的显示状态.

@出口参数:无.

@用户调用:允许

9.5.36 void Sys\_GUI\_Draw\_Progress\_Bar(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,u16 Front\_Beginning\_Color,u16 Front\_End\_Color,

u16 Back\_Color,double Progress)

在屏幕上画一个进度条.

@入口参数:

[IN]u16 Coord\_X:该进度条的左上角X坐标.

[IN]u16 Coord\_Y:该进度条的左上角Y坐标.

[IN]u16 Length:进度条长度.

[IN]u16 Width:进度条宽度.

[IN]u16 Front\_Beginning\_Color:进度条的进度为零(0%)时的前景色.

[IN]u16 Front\_End\_Color:进度条的进度为满(100%)时的前景色.

[IN]u16 Back\_Color:进度条的背景色.

[IN]double Progress:进度条的进度.

@出口参数:无

@用户调用:允许

9.5.37 u8 Sys\_GUI\_Create\_Dialog\_Box(u16 Coord\_X,u16 Coord\_Y,u16 Length,

u16 Width,u8 Status,void\* Title,void\* Icon)

在系统中创建一个对话框.这个对话框并不会立即显示在屏幕上,而是在刷新该对话框后才会显示.

@入口参数:

[IN]u16 Coord\_X:该对话框左上角的X坐标.

[IN]u16 Coord\_Y:该对话框左上角的Y坐标.

[IN]u16 Length:该对话框长度.

[IN]u16 Width:该对话框宽度.

[IN]u8 Status:该对话框的状态.

[IN]void\* Title:该对话框的标题.

[IN]void\* Icon:该对话框的图标.

@出口参数:

[OUT]u8 :成功返回对话框ID,失败返回”ENODIA”.

用户调用:允许

9.5.38 u8 Sys\_GUI\_Delete\_Dialog\_Box(u8 Dialog\_ID)

删除一个对话框.

@入口参数:

[IN]u8 Dialog\_ID:欲删除的对话框的ID,即” Sys\_GUI\_Create\_Dialog\_Box”的返回值.

@出口参数:

[OUT]u8 :成功返回0,失败返回”ENODIA”.

@用户调用:允许

9.5.39 u8 \_Sys\_GUI\_Draw\_Tab\_Controls(struct Dialog\_Control\_Head\* Control)

画出一个选项卡上的所有控件.

@入口参数:

[IN] struct Dialog\_Control\_Head\* Control:指向该选项卡结构体中头的指针.

@出口参数:

[OUT]u8 :成功返回1;若(该选项卡所在的)对话框存在而操作失败,返 回”ENODOP”;如果对话框不存在,返回”ENODIA”.

@用户调用:禁止

9.5.40 \_Sys\_GUI\_Draw\_Any\_Control(u16 Coord\_X,u16 Coord\_Y,

struct Dialog\_Control\_Head\* Control) 画出一个在某对话框内的控件.

@入口参数:

[IN]u16 Coord\_X:该对话框头部的左上角X坐标

[IN]u16 Coord\_Y:该对话框头部的左上角Y坐标

[IN] struct Dialog\_Control\_Head\* Control:指向该控件结构体中头的指针.

@出口参数:

[OUT]u8 :成功返回1,失败返回”ENODOP”.

@用户调用:禁止

9.5.41 u8 \_Sys\_GUI\_Draw\_Any\_Control\_For\_Tabs(u16 Coord\_X,u16 Coord\_Y,

struct Dialog\_Control\_Head\* Control)

画出一个在某选项卡内的控件.之所以与上面的函数分开成两个是为了避免函数重入的问题.

@入口参数:

[IN]u16 Coord\_X:该选项卡的左上角X坐标

[IN]u16 Coord\_Y:该选项卡的左上角Y坐标

[IN] struct Dialog\_Control\_Head\* Control:指向该控件结构体中头的指针.

@出口参数:

[OUT]u8 :成功返回1,失败返回”ENODOP”.

@用户调用:禁止

9.5.42 u8 \_Sys\_GUI\_Draw\_Dialog\_Controls(u8 Dialog\_ID)

对话框控件绘制函数.

@入口参数:

[IN]u8 Dialog\_ID:该对话框的ID.

@出口参数:

[OUT]u8 :成功返回该对话框的状态;失败返回”ENODIA”.

用户调用:允许

9.5.43 u8 Sys\_GUI\_Refresh\_Dialog\_Box(u8 Dialog\_ID)

刷新某一个对话框.

@入口参数:

[IN]u8 Dialog\_ID:对话框的ID.

@出口参数:

[OUT]u8 :成功返回该对话框的状态;失败返回”ENODOP”.

用户调用:允许

9.5.44 u8 Sys\_GUI\_Redefine\_Dialog\_Box(u8 Dialog\_ID,u16 Coord\_X,u16 Coord\_Y,

u16 Length,u16 Width,u8 Status,void\* Title,void\* Icon) 重定义对话框的部分属性.只有下面的这些属性可以用本函数进行重定义.

@入口参数:

[IN]u16 Coord\_X:对话框左上角的X坐标.

[IN]u16 Coord\_Y:对话框左上角的Y坐标.

[IN]u16 Length:对话框的长度.

[IN]u16 Width:对话框的宽度.

[IN]u8 Status:对话框的状态.

[IN]void\* Title:对话框的标题.

[IN]void\* Icon:对话框的图标.

@出口参数:

[OUT]u8 :成功返回该对话框的状态;失败返回”ENODOP”.

用户调用:允许

9.5.45 u8 Sys\_GUI\_Move\_Dialog\_Box(u16 Coord\_X,u16 Coord\_Y,u8 Dialog\_ID,u8 Mode) 移动对话框的函数,是”Sys\_GUI\_Redefine\_Dialog\_Box”的子集.

@入口参数:

[IN]u16 Coord\_X:该对话框新位置左上角的X坐标.

[IN]u16 Coord\_Y:该对话框新位置左上角的Y坐标.

[IN]u8 Dialog\_ID:该对话框的对话框号.

[IN]u8 Mode:移动模式.

@出口参数:

[OUT]u8 :成功返回对话框的新属性,失败则返回”ENODOP”.

用户调用:允许

9.5.46 u8 Sys\_GUI\_Refresh\_All\_Dialog\_Boxes(void)

刷新所有的对话框.刷新顺序是按照窗口叠加链表中的次序完成的,这样可以造成窗口叠加效果.

@入口参数:无

@出口参数:

[OUT]u8 :成功返回0,失败返回”ENODIA”.

@用户调用:允许

9.5.47 u8 Sys\_GUI\_Add\_Control\_To\_Dialog(u8 Dialog\_ID,

struct Dialog\_Control\_Head\* Control)向某一窗口添加一个控件.

@入口参数:

[IN]u8 Dialog\_ID:对话框的ID.

[IN]struct Dialog\_Control\_Head\* Control:指向控件结构体头部的指针.

@出口参数:

[OUT]u8 :成功返回1,失败返回” ENODOP”.

用户调用:允许

9.5.48 u8 Sys\_GUI\_Add\_Control\_To\_Tab(struct Dialog\_Control\_Head\* Tab,

u8 Tab\_ID,struct Dialog\_Control\_Head\* Control) 向某一选项卡添加一个控件.

@入口参数:

[IN]struct Dialog\_Control\_Head\* Tab:指向该选项卡组结构体头部的指针.

[IN]struct Dialog\_Control\_Head\* Control:指向控件结构体头部的指针.

@出口参数:

[OUT]u8 :成功返回1,失败返回”ENODOP”.

@用户调用:允许

9.5.49 u8 Sys\_GUI\_Delete\_Control\_From\_Tab(struct Dialog\_Control\_Head\* Tab,

u8 Tab\_ID,struct Dialog\_Control\_Head\* Control)从某一选项卡删除一个控件.

@入口参数:

[IN]struct Dialog\_Control\_Head\* Tab:指向该选项卡组结构体头部的指针.

[IN]u8 Tab\_ID:该选项卡的选项卡号(该选项卡在选项卡组中的位置).

[IN]struct Dialog\_Control\_Head\* Control:指向控件结构体头部的指针.

@出口参数:

[OUT]u8 :成功返回1,失败返回”ENODOP”.

@用户调用:允许

9.5.50 u8 Sys\_GUI\_Delete\_Control\_From\_Dialog(u8 Dialog\_ID,

struct Dialog\_Control\_Head\* Control) 从某一窗口删除一个控件.

@入口参数:

[IN]u8 Dialog\_ID:该对话框的ID.

[IN]struct Dialog\_Control\_Head\* Control:指向控件结构体头部的指针.

@出口参数:

[OUT]u8 : 成功返回1,失败返回”ENODOP”.

@用户调用:允许

9.5.51 u8 Sys\_GUI\_Get\_Clicked\_Dialog(u16 Coord\_X,u16 Coord\_Y)

根据输入的坐标,找到被点击的对话框.

@入口参数:

[IN]u16 Coord\_X:被点击处的X坐标.

[IN]u16 Coord\_Y:被点击处的Y坐标.

@出口参数:

[OUT]u8 :若点击处有对话框,返回最上方对话框的对话框ID,否则返回”ENODIA”

@用户调用:允许

9.5.52 u8 \_Sys\_GUI\_Search\_Clicked\_Dialog\_Condition(struct List\_Head\* Current\_Node,

void\* Coord\_Compound)

“Sys\_GUI\_Get\_Clicked\_Dialog”的内核链表搜索条件.

@入口参数:

[IN] struct List\_Head\* Current\_Node:指向现在正被搜索的链表节点的指针.

[IN]void\* Coord\_Compound:指向复合坐标的指针

@出口参数:

[OUT]u8 :满足条件(坐标在范围内)返回”END\_SEARCH”,否则返回0.

用户调用:禁止

9.5.53 u16 Sys\_GUI\_Get\_Clicked\_Control(u8 Dialog\_ID,u16 Coord\_X,u16 Coord\_Y)

根据” Sys\_GUI\_Get\_Clicked\_Dialog”提供的对话框ID,继续寻找被点击的控件,及点击的具体部分.

@入口参数:

[IN]u8 Dialog\_ID:由” Sys\_GUI\_Get\_Clicked\_Dialog”返回的对话框ID.

[IN]u16 Coord\_X:被点击处的X坐标.

[IN]u16 Coord\_Y:被点击处的Y坐标.

@出口参数:

[OUT]u8 :成功,前一个字节返回控件ID(在对话框的CCB中的位置),后一个字节返回控件点击位置的具体标识符,如果失败,在两个字节中都返回”ENODIA”.

@用户调用:允许

9.5.54 u16 Sys\_GUI\_Get\_Clicked\_Control\_In\_Tab(u8 Dialog\_ID,u16 Compound\_ID,

u16 Coord\_X,u16 Coord\_Y)

如果在一个对话框中被点击的控件是选项卡,还需要关心选项卡的那一部分被点击了.找到被点击的控件.

@入口参数:

[IN]u8 Dialog\_ID:对话框的ID.

[IN]u16 Compound\_ID:由”Sys\_GUI\_Get\_Clicked\_Control”返回的值.

[IN]u16 Coord\_X:点击处的X坐标.

[IN]u16 Coord\_Y:点击处的Y坐标.

@出口参数:

[OUT]u16 :一个混合的返回值.前一字节是控件ID(控件在Tab\_CCB中的位置).后一字节是控件被点击位置的具体标识符.如果失败,在两个字节中都返回”ENODIA”.

@用户调用:禁止

9.5.55 u8 Sys\_GUI\_Execute\_Clicked\_Control\_Action\_In\_Tab(u8 Dialog\_ID,

u16 Dialog\_Compound\_ID,u16 Tab\_Compound\_ID,u16 Coord\_X,u16 Coord\_Y)

执行选项卡中被点击控件的动作函数.

@入口参数:

[IN]u16 Dialog\_ID:选项卡所在对话框的ID.

[IN]u16 Dialog\_Compound\_ID:”Sys\_GUI\_Get\_Clicked\_Control”的返回值

[IN]u16 Tab\_Compound\_ID:”Sys\_GUI\_Get\_Clicked\_Control\_In\_Tab”的返回值

[IN]u16 Coord\_X:被点击处的X坐标.(其实无用,为了连续性而保留.)

[IN]u16 Coord\_Y:被点击处的Y坐标. (其实无用,为了连续性而保留.)

@出口参数:  
 [OUT]u8 :返回值与动作函数的相同.如果失败,返回”ENODOP”.

@用户调用:禁止

9.5.56 u8 Sys\_GUI\_Execute\_Clicked\_Control\_Action(u8 Dialog\_ID,u16 Compound,

u16 Coord\_X,u16 Coord\_Y)

执行对话框中被点击控件的动作函数.

@入口参数:

[IN]u16 Dialog\_ID:该控件所在对话框的ID.

[IN]u16 Compound:”Sys\_GUI\_Get\_Clicked\_Control”的返回值

[IN]u16 Coord\_X:被点击处的X坐标.(其实无用,为了连续性而保留.)

[IN]u16 Coord\_Y:被点击处的Y坐标. (其实无用,为了连续性而保留.)

@出口参数:无

用户调用:允许

10 中断管理单元(暂未实现)

10.1模块名称<interrupt.c>,头文件<interrupt.h>

10.2模块作用:

系统的中断管理单元.

10.3工作原理

1. 由系统处理,此时不允许中断嵌套.
2. 由任务自行处理

系统中的中断服务函数应尽可能地少,而且他们都必须在开头检测中断,如果定时器为开,则停下定时器中;如果定时器已停则不作操作.(写一个专用系统调用来做,不要让用户做).退出中断时,如果一开始没有做关定时器的操作,则也不执行开定时器的操作;如果一开始关了定时器,则之后要开定时器.(这是为了中断嵌套.)注意不是关中断,否则中断无法嵌套.

系统中的所有中断服务函数都不能当做进程去登记.中断服务函数的出口必须将值传给操作系统,再由操作系统将值传给进程.

11 错误管理单元

11.1 模块名称<error.c>,头文件<error.h>

11.2 模块作用

所有错误信息的控制模块.

11.3工作原理

11.3.1自动重启:

系统在发生HardFault等等异常中断时,会调用重启函数,完成系统重启.

11.3.2 错误号标识:

在<defines.h>中定义了所有可能的错误信息.

12 安全与随机数模块(尚未实现)

13 用户应用程序

13.1 模块名称<applications.c>,头文件<applications.h>

13.2 模块作用

所有用户应用程序的存放地点.