1 //串口打印相关的: driver\_l1\_cfg.h

//#define UART0\_TX\_IOB5\_\_RX\_IOB4 0x00000001

//#define UART0\_TX\_IOC12\_\_RX\_IOC13 0x00000002

//#define UART0\_TX\_IOD5\_\_RX\_IOD4 0x00000003

//#define UART1\_TX\_IOB7\_\_RX\_IOB6 0x00000004

//#define UART1\_TX\_IOC15\_\_RX\_IOC14 0x00000005

//#define UART1\_TX\_IOD9\_\_RX\_IOD8 0x00000006

//#define UART1\_TX\_IOD15\_\_RX\_IOD14 0x00000007

//#define UART\_TX\_NONE\_\_RX\_NONE 0x00000000

//#if DBG\_MESSAGE == CUSTOM\_ON

//#if 1//(GPDV\_BOARD\_VERSION == DVP\_V1\_0) // EVB

//#if PCB\_TYPE == TYPE\_1248

//#define UART\_TX\_RX\_POS UART1\_TX\_IOB7\_\_RX\_IOB6// UART1\_TX\_IOD9\_\_RX\_IOD8

//#else

//#define UART\_TX\_RX\_POS UART1\_TX\_IOC15\_\_RX\_IOC14//UART1\_TX\_IOC15\_\_RX\_IOC14//UART1\_TX\_IOB7\_\_RX\_IOB6

//#endif

//#else

//#define UART\_TX\_RX\_POS UART\_TX\_NONE\_\_RX\_NONE

//#endif

2 //打开串口调试: customer.h

//#define CUSTOM\_ON 1

//#define CUSTOM\_OFF 0

//#define DBG\_MESSAGE CUSTOM\_ON//CUSTOM\_OFF // UART debug message

3 //烧录工具: writer（D:\TOOL）

4 //工程中Tool的运用：

//运行ComBin.exe， F:\work\study\v001\programe\Tools\GPCV1248\_SPIFC\_Pack\_\GPCV1248\_SDRAM\_Header\_144M.bin，其中144的确定,

//{(CUSTOMER.H)

// CPU and system clock

// #define INIT\_MHZ 144//115//144 // MHz

//}

5 //调试程序的:

//#if 0

//#undef \_\_here\_\_

//#undef \_\_msg

//#define \_\_here\_\_ eLIBs\_printf("@L%d(%s)\n", \_\_LINE\_\_, \_\_FILE\_\_);

//#define \_\_msg(...) (eLIBs\_printf("MSG:L%d(%s):", \_\_LINE\_\_, \_\_FILE\_\_), \

eLIBs\_printf(\_\_VA\_ARGS\_\_) )

//#define DebugPrintf(...) (eLIBs\_printf("MSG:L%d(%s):", \_\_LINE\_\_, \_\_FILE\_\_), \

// eLIBs\_printf(\_\_VA\_ARGS\_\_) )

//#else

//#define DebugPrintf(...)

//#endif

6 //LED灯的控制，根据原理图看是哪个io口控制，然后把io设置成相应的电平值: (ap\_peripheral\_handling.c)

//case LED\_INIT:

// led\_green\_on();

/\* add begin by Zhibo, 2017-04-27 \*/

//led\_red\_on();

/\* add end by Zhibo, 2017-04-27 \*/

// DBG\_PRINT("led\_type = LED\_INIT\r\n");

// \_\_msg("led\_type = LED\_INIT\r\n");

// break;

//(customer.h)

//#define LED1 IO\_E3

//#define LED2 IO\_E2

7 // OSTaskCreate(state\_handling\_entry, (void \*) 0, &StateHandlingStack[STATE\_HANDLING\_STACK\_SIZE - 1], STATE\_HANDLING\_PRIORITY);

//系统的初始化

8 //sd卡或者usb： void state\_video\_record\_entry(void \*para, INT32U state)中，首先判断有没有sd或者usb，没有的时候，

//向MSG\_PERIPHERAL\_TASK\_LED\_SET队列中发送消息。如果有则进行判别是连接电脑还是连接usb或者sd。

//ap\_state\_handling\_connect\_to\_pc(STATE\_VIDEO\_RECORD);摄像状态

9. //有电池没有usb连接时，五秒自动关机，msgQSend(ApQ, MSG\_APQ\_POWER\_KEY\_ACTIVE, NULL, NULL, MSG\_PRI\_NORMAL);

//

//if(shutoff\_cnt){

// shutoff\_cnt--;

// if(usb\_state\_get() != 0)

// {

//led\_green\_off();

// shutoff\_cnt = 0;

// DBG\_PRINT("fankun5:%d\r\n", usb\_state\_get());

// }

// if(!shutoff\_cnt)

// {

// if(usb\_state\_get() == 0)

// {

// DBG\_PRINT("fankun1:%d\r\n", usb\_state\_get());

// msgQSend(ApQ, MSG\_APQ\_POWER\_KEY\_ACTIVE, NULL, NULL, MSG\_PRI\_NORMAL);

// }

// }

// else

// {

// if(shutoff\_cnt % 20 == 0)

// {

// if(t\_flash\_flag == 0)

// {

// t\_flash\_flag = 1;

// moto\_vibration\_on(20);

// led\_green\_on();

// led\_red\_on();

// }

// else

//{

//led\_red\_off();

// led\_green\_off();

//t\_flash\_flag = 0;

// }

// }

// }

// }

//--->case MSG\_APQ\_POWER\_KEY\_ACTIVE:

// if ((ap\_video\_record\_sts\_get() & VIDEO\_RECORD\_BUSY))

// {

// if(ap\_video\_record\_func\_key\_active(msg\_id))

// {

// break;

// }

//}

// DBG\_PRINT("dfdfffdf\r\n");

//msgQSend(PeripheralTaskQ, MSG\_PERIPHERAL\_TASK\_ZD, &type, sizeof(INT8U), MSG\_PRI\_NORMAL);

//OSTimeDly(5);

//video\_encode\_exit();

//video\_calculate\_left\_recording\_time\_disable();

//ap\_state\_handling\_power\_off();

//break;

//--->

10. //进入摄像头状态,msgQSend(ApQ, MSG\_APQ\_VIDEO\_RECORD\_ACTIVE, NULL, NULL, MSG\_PRI\_NORMAL);

11. //摄像循环间隔： ap\_state\_config\_record\_time\_set(void); 宏:define C\_CYCLIC\_VIDEO\_RECORD on/off

12. //格式化TF卡命令：msgQSend(StorageServiceQ, MSG\_STORAGE\_SERVICE\_FORMAT\_REQ, NULL, NULL, MSG\_PRI\_NORMAL);

13. //移动侦测： msgQSend(ApQ, MSG\_APQ\_MOTION\_DETECT\_ACTIVE, NULL, NULL, MSG\_PRI\_NORMAL);

//ap\_state\_config\_md\_set(status);status 要设置为1，不然停止的时候，会出问题。

14. //chdir(const char \*path);更改当前工作目录。Path 目标目录，可以是绝对目录或相对目录。成功返回0 ，失败返回-1

15. // int unlink(const char\* pathname);unlink从文件系统中中删除一个名字，若这个名字是指向这个文件的最后一个链接，

//并且没有进程处于打开这个文件的状态，则删除这个文件，释放这个文件占用的空间。

//如果这个名字是指向这个文件的最后一个链接，但有某个进程处于打开这个文件的状态，则暂时不删除这个文件，要等到打开

//这个文件的进程关闭这个文件的文件描述符后才删除这个文件。

//如果这个名字指向一个符号链接，则删除这个符号链接。

//如果这个名字指向一个socket、fifo或者一个设备，则这个socket、fifo、设备的名字被删除，当时打开这些socke、fifo、

//设备的进程仍然可以使用它们。

//返回值：调用成功返回0，不成功返回-1.

16. //setfattr: The setfattr command associates a new value with an extended attribute name for each specified file.

17. // OS\_EVENT \*OSQCreate(void \*\*start, INT16U size); 调用OSQcreate()函数创建消息队列，

18. /\*sprintf()函数用于将格式化的数据写入字符串，其原型为：

int sprintf(char \*str, char \* format [, argument, ...]);

【参数】str为要写入的字符串；format为格式化字符串，与printf()函数相同；argument为变量。

sprintf的作用是将一个格式化的字符串输出到一个目的字符串中，而printf是将一个格式化的字

符串输出到屏幕。sprintf的第一个参数应该是目的字符串，

如果不指定这个参数，执行过程中出现 "该程序产生非法操作,即将被关闭...."的提示.

sprintf()会根据参数format 字符串来转换并格式化数据，然后将结果复制到参数str 所指的字符串数组，

直到出现字符串结束('\0')为止。关于参数format 字符串的格式请参考printf()。

\*/

19. /\*

(\_\_asm)关键字启动内联汇编并且能写在任何C++合法语句之处。它不能单独出现，必须接汇编指令、

一组被大括号包含的指令或一对空括号。术语"\_\_asm块"在这里是任意一个指令或一组指令无论是否在括号内。

\*/

20. //MSG\_STORAGE\_SERVICE\_MOUNT:检测有卡发送的状态.

21. //#define SD\_POS SDC\_IOC4\_IOC5\_IOC6\_IOC7\_IOC8\_IOC9 //sd卡

22. //录像自动保存： time\_interval = 3 \* VIDEO\_RECORD\_CYCLE\_TIME\_INTERVAL + 112;

23. //stat函数讲解

//表头文件:

// #include <sys/stat.h>

// #include <unistd.h>

//定义函数:

//int stat(const char \*file\_name, struct stat \*buf);

//函数说明: 通过文件名filename获取文件信息，并保存在buf所指的结构体stat中

//返回值: 执行成功则返回0，失败返回-1，错误代码存于errno

24. //void FS\_OS\_GetTime(dostime\_t \*dt); //设置拍照录像显示时间与否

//void FS\_OS\_GetDate(dosdate\_t \*dd); //设置拍照录像显示时间与否

25. //APQ消息：只能在应用直接相互传递。

26. //延时过程中，按键的抬起动作可能被过滤，没有及时的清除，从而影响操作。可以不用延时操作函数。

27. //void ap\_video\_record\_md\_tick(INT8U \*md\_tick, INT32U state), 移动侦测到一定时间后，停止侦测，并保存。

28. //ap\_state\_config\_pic\_size\_set(8);设置图像大小，如：1280x720等

29. //ap\_state\_config\_video\_resolution\_set(2);设置视频大小，如：720P

30. //循环录像，sys\_set\_timer((void \*)msgQSend, (void \*)ApQ, MSG\_APQ\_CYCLIC\_VIDEO\_RECORD, \

//cyclic\_record\_timerid, time\_interval);启动循环录像定时器，当触发点后，进行循环录像。

31. //state\_handling\_init(void)创建了APQ消息队列，所有的应用消息都从APQ中接收消息。应用之间的切换管理的窗口就是

//state\_handling\_entry(void \*para);相当于一个总应用管理窗口。

32. //ap\_peripheral\_charge\_det();充电检测，配置好相应的IO口，如：#define CHARGE\_PIN IO\_D13//IO\_C14

33. //ap\_state\_handling\_connect\_to\_pc(INT32U prev\_state),key\_down = gpio\_read\_io(SWITCH\_MODE\_KEY);

//来检测连接PC的是pc\_cam模式还是usb模式。

33. //NVIC:嵌套中断向量控制器。

33. //定义函数：off\_t lseek(int fildes, off\_t offset, int whence);

//每一个已打开的文件都有一个读写位置, 当打开文件时通常其读写位置是指向文件开头, 若是以附加的方式打开文件(如O\_APPEND),

//则读写位置会指向文件尾. 当read()或write()时, 读写位置会随之增加,lseek()便是用来控制该文件的读写位置.

//参数fildes 为已打开的文件描述词, 参数offset 为根据参数whence来移动读写位置的位移数.

//参数 whence 为下列其中一种:

//SEEK\_SET 参数offset 即为新的读写位置.

//SEEK\_CUR 以目前的读写位置往后增加offset 个位移量.

//SEEK\_END 将读写位置指向文件尾后再增加offset 个位移量. 当whence 值为SEEK\_CUR 或

//SEEK\_END 时, 参数offet 允许负值的出现.

//下列是教特别的使用方式:

//1) 欲将读写位置移到文件开头时:lseek(int fildes, 0, SEEK\_SET);

//2) 欲将读写位置移到文件尾时:lseek(int fildes, 0, SEEK\_END);

//3) 想要取得目前文件位置时:lseek(int fildes, 0, SEEK\_CUR);

34. //定义函数: int stat(const char \*file\_name, struct stat \*buf); 通过文件名filename获取文件信息，并保存在buf所指的结构体stat中

//返回值: 执行成功则返回0，失败返回-1，错误代码存于errno.

//ENOENT 参数file\_name指定的文件不存在

//ENOTDIR 路径中的目录存在但却非真正的目录

//ELOOP 欲打开的文件有过多符号连接问题，上限为16符号连接

//EFAULT 参数buf为无效指针，指向无法存在的内存空间

//EACCESS 存取文件时被拒绝

//ENOMEM 核心内存不足

//ENAMETOOLONG 参数file\_name的路径名称太长

35. //task\_state\_handling.c中,state\_handling\_entry(void)管理着各个应用的切换任务，比如: preview, record,

//audio, browse, setting and so on.

//每个应用的切换首先要经过这里，才能进入相应得应用处理对应的事件。

36. //开机logo, state\_startup\_init(void), 宏C\_LOGO所定义的部分.

37. //tft\_init();屏初始化相关的内容，点屏目涉及的参数时需要在这里调用。

38. //MAJOR(dev\_t dev), MINOR(dev\_t dev), MKDEV(int major, int minor);

39. //avi\_audio\_record\_entry: gp\_memset((INT8S \*)(audio\_frame[write\_audio\_idx].buffer\_addrs + 8), 0, (AVI\_AUDIO\_PCM\_SAMPLES << 1));

//录像声音相关的。

40. //void FS\_OS\_GetDate(dosdate\_t \*dd),void FS\_OS\_GetTime(dostime\_t \*dt)显示文件时间日期。

41.ARM基础知识：

R13 Sp 堆栈寄存器

R14 Lr 连接寄存器

R15 PC 程序计数器

42. Arm的工作模式以及相关寄存器设置：

1，用户模式（usr）      ［10000］：ARM处理器正常的程序执行状态

2，快速中断模式（fiq）  ［10001］：用于高速数据传输或通道处理

3，外部中断模式（irq）  ［10010］：用于通用的中断处理

4，管理模式（svc）      ［10011］：[操作系统](http://lib.csdn.net/base/operatingsystem)使用的保护模式

5，中止模式（abt）          ［10111］：当数据或指令预取终止时进入该模式， 用于虚拟存

储及存储保护

6，未定义指令模式（und）［11011］：当未定义的指令执行时进入该模式，用 于支持硬件

协处理器的软件仿真

7，系统模式（sys）      ［11111］：运行具有特权模式的操作系统任务

43. BOT\_DMA\_RAM\_BUF\_ORDER 7650修改了值， 由8改为3，换了usb-l2库文件。

44. AVI\_REC\_MAX\_BYTE\_SIZE: 录像的最大文件大小。

45. global: 就是相当于C语言中的Extern，声明此变量，并且告诉链接器此变量是全局的，外部可以访问.

46. 720P换成1080P，改动的地方：

a：ap\_state\_config\_video\_resolution\_set(0);

b: #define INIT\_MHZ 144

c:添加video\_codec\_dynamic\_jpeg\_Q.c到过程中；

d：sensor\_gc1004\_iqj.h中

/\*

0x0700, 0x0808: 帧率: 25(暗光)-30(亮光)

0x0700, 0x088f: 帧率: 25(暗光)-25(亮光)

0x0700, 0x08ff: 帧率: 22.8(暗光)-22.8(亮光)

0x0701, 0x080f: 帧率: 22.4(暗光)-22.4(亮光)

0x0701, 0x08ff: 帧率: 18.2(暗光)-18.2(亮光)

\*/

{0x07 ,0x00},

{0x08 ,0xff}

47. 改换pccam的帧率：void sensor\_set\_fps(INT32U fpsValue)

48. ap\_state\_resource\_char\_draw\_osd(0x2F...);2F 对应的是‘/’,视频显示的如"2017/06/30", 如果改成0x2d，则显示如："2017-06-30".

49. 读版本号：在sd卡根目录下创建"time.txt"文件, 把"VERSION\_NUMBER?????"拷贝到"time.txt"中，就可以读出来。

50. #define SENSOR\_FLIP 0or 1, 画面倒转控制。

51. sensor\_frame\_rate = 30,// default sensor frame rate,sensor 默认帧率

52. scaler: 定标器，计算器，换算。

53. 账号：zhibo@joyhonest.com，密码：zzb@2611212，网址：<https://qiye.163.com/login/>

54. git工具的使用技巧：

git init 初始化，增加一个.git目录(注意不要随意修改)

git add， git remote add origin git@github.com:yilanjueding123/lexinxingye.git增加对我们github上创建的lexinxingye仓库的管理

git pull git@github.com: yilanjueding123/lexinxingye.git在本地同步一下仓库的内容

git commit –m “这里写下你自己的记录本次提交内容的信息”

git push [git@github.com: yilanjueding123/lexinxingye.git](mailto:git@github.com:bxxfighting/NowToDo.git)推送到远程仓库上去

git push -u origin master把本地库的所有内容推送到远程库上

55. git学习

要关联一个远程库，使用命令git remote add origin git@server-name:path/repo-name.git；

关联后，使用命令git push -u origin master第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

要克隆一个仓库，首先必须知道仓库的地址，然后使用git clone命令克隆。

Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。

git clone git@github.com: yilanjueding123/gitskills.git

分支管理：

创建dev分支，然后切换到dev分支：git checkout -b dev

git checkout命令加上-b参数表示创建并切换，相当于以下两条命令

git branch dev

git checkout dev

git branch命令查看当前分支

git branch命令会列出所有分支，当前分支前面会标一个\*号

Git鼓励大量使用分支：

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

用git log --graph命令可以看到分支合并图。

Git分支十分强大，在团队开发中应该充分应用。

合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

56. 小卡录音正常，64g的录音没有文件：

INT8S ap\_audio\_record\_reply\_action(STOR\_SERV\_FILEINFO \* file\_info\_ptr)中：

INT32S free\_size = file\_info\_ptr->storage\_free\_size;(小卡是INT16S)。

(sd\_card\_csd[SD\_DEVICE\_NUM][2]>>16) & 0x0000FFFF；//支持64G卡。

57. MD\_STOP\_TIME,移动侦测等待时间

58. AWB:自动白平衡调节：

定义：要求在不同色温环境下，照白色的物体，屏幕中的图像应也是白色的。色温表示光谱成份，光的颜色。色温低表示长波光成分多。当色温改变时，光源中三基色(红、绿、蓝)的比例会发生变化，需要调节三基色的比例来达到彩色的平衡，这就是白平衡调节的实际。

59. 图像压缩方式：

JPEG：(joint photo graphicexpert group)静态图像压缩方式。一种有损图像的压缩方式。压缩比越大，图像质量也就越差。当图像精度要求不高存储空间有限时，可以选择这种格式。目前大部分数码相机都使用JPEG格式。

60. 彩色深度(色彩位数)

反映对色彩的识别能力和成像的色彩表现能力，实际就是A/D转换器的量化精度，是指将信号分成多少个等级。常用色彩位数(bit)表示。彩色深度越高，获得的影像色彩就越艳丽动人。现在市场上的摄像头均已达到24位，有的甚至是32位。

61. 图像噪音

指的是图像中的杂点干挠。表现为图像中有固定的彩色杂点。

62. 视角

与人的眼睛成像是相成原理，简单说就是成像范围。

63. 录像的时候自动覆盖：(1247)

ap\_storage\_service\_storage\_mount：//led\_type=LED\_CARD\_NO\_SPACE;

ap\_video\_record\_func\_key\_active: (disk\_free\_size < CARD\_FULL\_SIZE\_RECORD)

ap\_storage\_service\_freesize\_check\_switch: sys\_set\_timer((void\*)msgQSend, (void\*)StorageServiceQ, MSG\_STORAGE\_SERVICE\_FREESIZE\_CHECK, storage\_freesize\_timerid, STORAGE\_TIME\_INTERVAL\_FREESIZE);

ap\_video\_record\_reply\_action: bkground\_del\_disable(0);

64. R\_MIC\_READY |= 0x110;//0x10C;//降低mic声音

65. 帧率寄存器地址与值：(如GC1014\_MIPI\_720P[]中)

{0x07 ,0x01}, //18fps

{0x08 ,0xff}, //0x08

66. tiny\_counter\_get();//获取始终tick

66. 1247支持64G卡修改：

1. Drv\_l2\_sd.c中

c\_size = ((sd\_card\_csd[SD\_DEVICE\_NUM][2]>>16) & 0x0000FFFF)+((sd\_card\_csd[SD\_DEVICE\_NUM][1]&0x0000003F)<<16);

2. typedef struct

{

INT16S file\_handle;

#if GPS\_TXT

INT16S txt\_handle;

#endif

INT32S storage\_free\_size;

INT32U file\_path\_addr;

#if GPS\_TXT

INT32U txt\_path\_addr;

#endif

} STOR\_SERV\_FILEINFO;

3.格式化卡；

67. 摄像头阻抗匹配：

MipiIF\_Setting：(\*(volatile unsigned \*)0xD0F0000c) = 0x00012f07;

68. 修改MCLK：

1. timer\_pwm\_setup(TIMER\_B, (12000000), 50, PWM\_NRO\_OUTPUT);

2. {0xF8, 0x07}, // Pll mode 2 ， 摄像头寄存器倍频

**第二章 无线WiFi**

1. beacon:

信标,通信术语 。信标单位通常以[毫秒](https://baike.baidu.com/item/%E6%AF%AB%E7%A7%92)（ millsecond ）为单位( 1/1000 秒),一般默认值为100。信标间隔调低 ,可以加快wireless client 连结速度 。

2.

（\*）LAN：即局域网，是路由和主机组成的内部局域网，一般为有线网络。

（\*）WAN：即广域网，是外部一个更大的局域网。

        （\*）WLAN（Wireless LAN，即无线局域网）：前面我们说过LAN是局域网，其实大多数指的是有线网络中的局域网，无线网络中的局域网，一般用WLAN。

        （\*）AP（Access point的简称，即访问点，接入点）：是一个无线网络中的特殊节点，通过这个节点，无线网络中的其它类型节点可以和无线网络外部以及内部进行通信。这里，AP和无线路由都在一台设备上（即Cisco E3000）。

        （\*）Station（工作站）：表示连接到无线网络中的设备，这些设备通过AP，可以和内部其它设备或者无线网络外部通信。

        （\*）Assosiate：连接。如果一个Station想要加入到无线网络中，需要和这个无线网络中的AP关联（即Assosiate）。

        （\*）SSID：用来标识一个无线网络，后面会详细介绍，我们这里只需了解，每个无线网络都有它自己的SSID。

        （\*）BSSID：用来标识一个BSS，其格式和MAC地址一样，是48位的地址格式。一般来说，它就是所处的无线接入点的MAC地址。某种程度来说，它的作用和SSID类似，但是SSID是网络的名字，是给人看的，BSSID是给机器看的，BSSID类似MAC地址。

        （\*）BSS（Basic Service Set）：由一组相互通信的工作站组成，是802.11无线网络的基本组件。主要有两种类型的IBSS和基础结构型网络。IBSS又叫ADHOC，组网是临时的，通信方式为Station<->Station，这里不关注这种组网方式；我们关注的基础结构形网络，其通信方式是Station<->AP<->Station，也就是所有无线网络中的设备要想通信，都得经过AP。在无线网络的基础形网络中，最重要的两类设备：AP和Station。