T80XXEWL双image备份实现说明

Luowutong

2014.7.15

# 1 产品说明

## 原系统分区

+-----------+----------+---------+-------------+-------------+

| Uboot | UsrCfg | RF | Kernel | disk |

+-----------+----------+---------+-------------+-------------+

|<--------0x50000----------->|<-------0x6B0000---->|

0x00000000-0x00800000 : "ALL" 8M

0x00000000-0x00030000 : "Bootloader" 192k=3\*64K

0x00030000-0x00040000 : "Config" 64k

0x00040000-0x00050000 : "Factory" 64k

0x00050000-0x00700000 : "Kernel" 6848k =107\*64K=6.6875M

0x00700000-0x00800000 : "disk" 1M

## 相关文件大小：

Bootloader ：uboot\_V1.0.bin 108K

Kernel：T80XXEWLV1.0.1.6.bin 3442k

## 其他

这里的Kernel包含了linux内核及文件系统。

# 2名词说明

app： linux内核加文件系统（kernel+sysfs），这里统称为app，即指软件T80XXEWLV1.0.1.6.bin。app1表示主分区，app2表示备份分区。

Bootflag：用于标识bootloader引导哪个app。0表示从主分区引导，1表示从备份分区引导。

Cnvenv：flash分区，用于存放bootflag，可扩展存取其他信息。

Eraseblock：flash最新擦除块大小，T80XXEWL是64k。所有分区大小必须是Eraseblock size 的整数倍。

# 3需求描述

基本需求：设备在升级过程中出现断电，上电后设备依然很启动。

高级需求：当前设备使用软件版本存在致命bug，通过修改bootflag免升级回退到上一版本。

# 4 实现描述

## 新分区描述

+---------+-------------+--------+----------+-----------+-----------------+-----------+

| Uboot| UsrCfg | RF | app1 | app2 | disk | cvnenv |

+---------+-------------+--------+----------+-----------+-----------------+-----------+

|<----0x50000--------------->|<-0x3A0000\*2-->|<-0x60000+0x10000->|

0x00000000-0x00800000 : "ALL" 8M

0x00000000-0x00030000 : "Bootloader" 192k=3\*64K

0x00030000-0x00040000 : "Config" 64k

0x00040000-0x00050000 : "Factory" 64k

0x00050000-0x003f0000 : "Kernel1" 3712K

0x003f0000-0x00790000 : "Kernel2" 3712K

0x00790000-0x007f0000 : "disk" 384k

0x007f0000-0x00800000 : "Cvnenv" 64k

说明：

1. 为不丢失已发货设备Config、Factory数据，以及减少app代码修改，前面3个分区保持不变。
2. Disk分区是jffs2文件系统，与已发货版本保持一致，jffs2至少要求5个eraseblock，因为所有分区必须是eraseblock size整数倍，考虑到其他分区情况这里分配6个eraseblock。
3. Kernel1指app1，Kernel2指app2，当前app软件大小达到3442k，剩余空间不是很多，后续合入其他特性时需要考虑flash空间。
4. 由于Config （实际是bootenv）存放了app的配置，考虑到数据安全，新建Cvnenv分区用于存放bootflag，可扩展存取其他信息。

## 代码修改说明

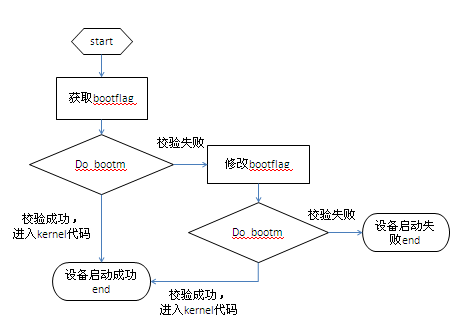
### 分区修改

内核创建的mtd分区信息是通过解析Kernel command line获得，由bootloader传入。修改文件source\vendors\Ralink\T80XXEWL\boot\_args.8M32M

Uboot没有类似内核那样的分区信息，所谓的分区实际只是定义相关宏。修改文件在uboot\include\configs\rt2880.h

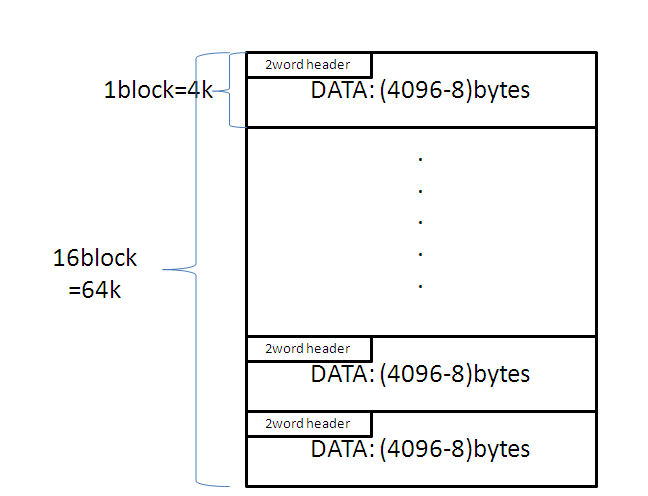
### Bootloader引导app过程修改

函数board\_init\_r引导app前先获取bootflag，根据bootflag获取相应flash分区起始地址。将该地址作为参数传给函数do\_bootm，do\_bootm会从flash中读取app并作校验，如果校验失败则回到函数board\_init\_r，修改bootflag，并再次执行do\_bootm。



修改文件lib\_mips\board.c

### bootflag存储读写实现



1、Bootflag存放在cvnenv区，将其分成16个block，见上图。使用多次写、一次擦技术，data存放格式同uboot bootenv一致即：”item1=value1\0item2=vlaue2\0\0”，每隔item间用一个\0分割，\0\0表示数据结束。

2、结构体义如下

typedef struct cvn\_environment\_s {

unsigned long crc; /\* CRC32 over data bytes \*/

unsigned long env\_flag;

unsigned char data[CVN\_ENV\_DATA\_SIZE]; /\* Environment data \*/

} cvn\_env\_t;

3、读操作：data有效需要判断crc正确，env\_flag为0xAA55AA55。Data无效，或者data中找不到读取的item则返回默认值（如bootflag项，则返回0）。

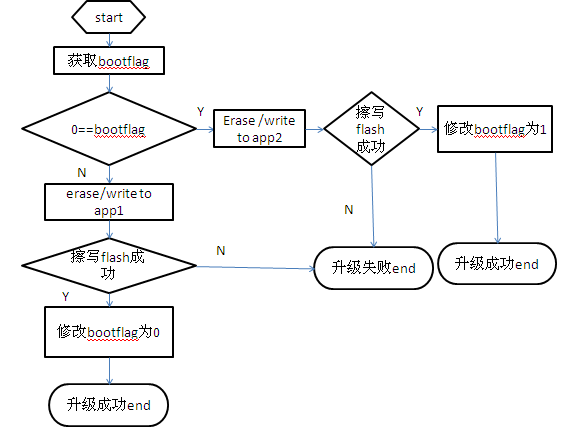
4、写操作：首先找到cvnenv中有效data，找不到则擦除cvnenv，将数据写入第一块；找到则将data数据拷贝到内存，添加或修改要写入的item，再将data写入下一个block。

5、代码修改

bootloader代码添加在uboot/common/spi/env\_spi.c

app代码添加在../source/lib/libnvram/bootflag\_api.c

### 升级app流程修改



Bootloader修改文件lib\_mips\board.c

App 修改文件比较零散，建议后面将所有升级app逻辑放在lib中。

# 5接口说明

int get\_bootflag(void); 获取bootflag值。

void set\_bootflag(int iflag);设置bootflag为指定值。

Bootloader下命令：

getbootflag：打印bootflag值

setbootflag：设置bootflag另一个app分区值。即set\_bootflag((cur\_bootflag + 1)%2)

后续app命令行也可以添加类似功能命令。

# 6单双image兼容说明

## 方案1

代码静态兼容，使用代码宏，不同产品各自编译。

主要修改如下，特点风险小。

int get\_bootflag(void)

{

#ifdef CONFIG\_DUAL\_IMAGE

char szValue[16] = {0};

if(0 != cvn\_getenv("bootflag", szValue, sizeof(szValue)))

{

return MAIN\_FLAG;

}

if('0' == szValue[0])

{

return MAIN\_FLAG;

}

else

{

return BAKCUP\_FLAG;

}

#else

return MAIN\_FLAG;

#endif

}

void set\_bootflag(int iflag)

{

#ifdef CONFIG\_DUAL\_IMAGE

if(MAIN\_FLAG == iflag)

{

cvn\_setenv("bootflag", "0");

}

else

{

cvn\_setenv("bootflag", "1");

}

#endif

}

#ifdef CONFIG\_DUAL\_IMAGE

#define MTDBLOCK4 "/dev/mtdblock4" /\* main image kernel \*/

#define MTDBLOCK5 "/dev/mtdblock5" /\* backup image kernel \*/

#define MTDBLOCK6 "/dev/mtdblock6"

#define MTDBLOCK7 "/dev/mtdblock7" /\* cvn\_env for bootflag \*/

#define MTD4 "/dev/mtd4" /\* main image kernel \*/

#define MTD5 "/dev/mtd5" /\* backup image kernel \*/

#define MTD6 "/dev/mtd6"

#define MTD7 "/dev/mtd7" /\* cvn\_env for bootflag \*/

#else

#define MTDBLOCK4 "/dev/mtdblock4" /\* main image kernel \*/

#define MTDBLOCK5 "/dev/mtdblock4" /\* backup image kernel \*/

#define MTDBLOCK6 "/dev/mtdblock5"

#define MTDBLOCK7 "/dev/mtdblock6" /\* cvn\_env for bootflag \*/

#define MTD4 "/dev/mtd4" /\* main image kernel \*/

#define MTD5 "/dev/mtd4" /\* backup image kernel \*/

#define MTD6 "/dev/mtd5"

#define MTD7 "/dev/mtd6" /\* cvn\_env for bootflag \*/

#endif

## 方案2

在cvnenv中添加daul\_image\_flag 项，读写存储与bootflag相同。为实现已发货T80XXEWL是dual image开启，如果在cvnenv中读取不到daul\_image\_flag，则认为daul\_image\_flag=1，及双image功能使能。

后期新产品需要关闭双image功能需要在制作img文件是添加daul\_image\_flag=0，或者生产时通过厂测命令添加daul\_image\_flag=0。

后期已发货产品需要关闭双image功能，先设置bootflag为0，再设置daul\_image\_flag=0。

说明：该方案能够动态修改双image功能使能，比较灵活，相比风险也较大。