# AVR_BootLoader 设计及实现

由于最近日本 TOM 公司的一个项目中,有要求在超终终端参数设置中能更新程序,于是,就把以前编写的 boot_loader 代码重新整理了下,从代码上略做了些修改。借此发到网上来,以供像我当初对 boot 认识不深的朋友们阅读，希望能给大家带来帮助。

MCU：ATMEGA128
编译环境：WinAVR

## 准备工作

## 1、通讯协议选择及上位机：

由于为了通用及简化，选择了超级终端及 Xmodem 协议(此协议不在此细说，望读者能直接去查阅相关信息)

## 2、页写函数：

此函数在 boot 区内被 boot_loader 代码调用，将收到的数据即二进制应用代码写到 application flash section(应用代码区)。此项工作，查阅了下 boot.h，直接 copy 出了内部示例代码，函数全称为：void boot_program_page (uint32_t page, uint8_t *buf) 此代码在 boot.h 是被屏蔽了的。

## 3、boot_loader 代码的定位：

由于 avr 的 boot 区根据熔丝位的不同选择不同大小的 boot 区,所以 boot_loader 代码要烧写到相应的位置，以确保代码的正确运行。本例选择的是最大 boot 区:boot flash section size=4096 words boot start address=$F000;[BOOTSZ=00]。注意：此处所标起始地址的单位是字，要左移一位转化单位为字节。即$F000 words=0x1e000 byte
并写入到 Makefile 相应位置处。

```
#---------------- Linker Options ----------------
#  -Wl,...:     tell GCC to pass this to linker.
#    -Map:      create map file
#    --cref:    add cross reference to  map file
LDFLAGS = -Wl,-Map=$(TARGET).map,--cref
LDFLAGS += $(EXTMEMOPTS)
LDFLAGS += $(patsubst %,-L%,$(EXTRALIBDIRS))
LDFLAGS += $(PRINTF_LIB) $(SCANF_LIB) $(MATH_LIB)
#LDFLAGS += -T linker_script.x
#添加处：
LDFLAGS += -Wl,--section-start=.text=0x1e000
```

msn:brotherwyz@hotmail.com Email:eric540@163.com

在应用程序中执行下列语句，即可执行 boot_loader 代码
Asm volatile(“jmp 0x1e000”);

# 软件思想

## 1、启动

启动时开启定时器发送连接信号，每隔一秒发一次，如果发送了十次都没收到信息，则退出到应用代码区（如果 flash 里有应用代码，如果没有，PC 指针会依次递增指到 boot 区，重新运行 boot_loader 代码）。
如果在这十次连接信号中收到数据，则退出连接，进入收包，检包，装包，烧写主要代码区。

## 2、收包

收包:采用 Xmodem 协议,CRC 校验。Baudrate=9600bps,stop_bit=1,data_len=8,parity=NONE Flow_control=NONE。将中断向量定位在 boot 区，以中断方式接收和发送。
接收到 CAN 中止信号，打印相关提示信息，退出 boot 程序，进入 application code
接收到 EOT 文件结束信号，判断装包情况(请看装包思想)，打印提示信息，跳转到应用程序。
无上述信号，收到包长 133 字节时，通知主程序已接收到一包信息。
133 字节=包头(1byte)+包号(1byte)+反包号(1byte)+包数据(128byte)+crc 校验(2byte)

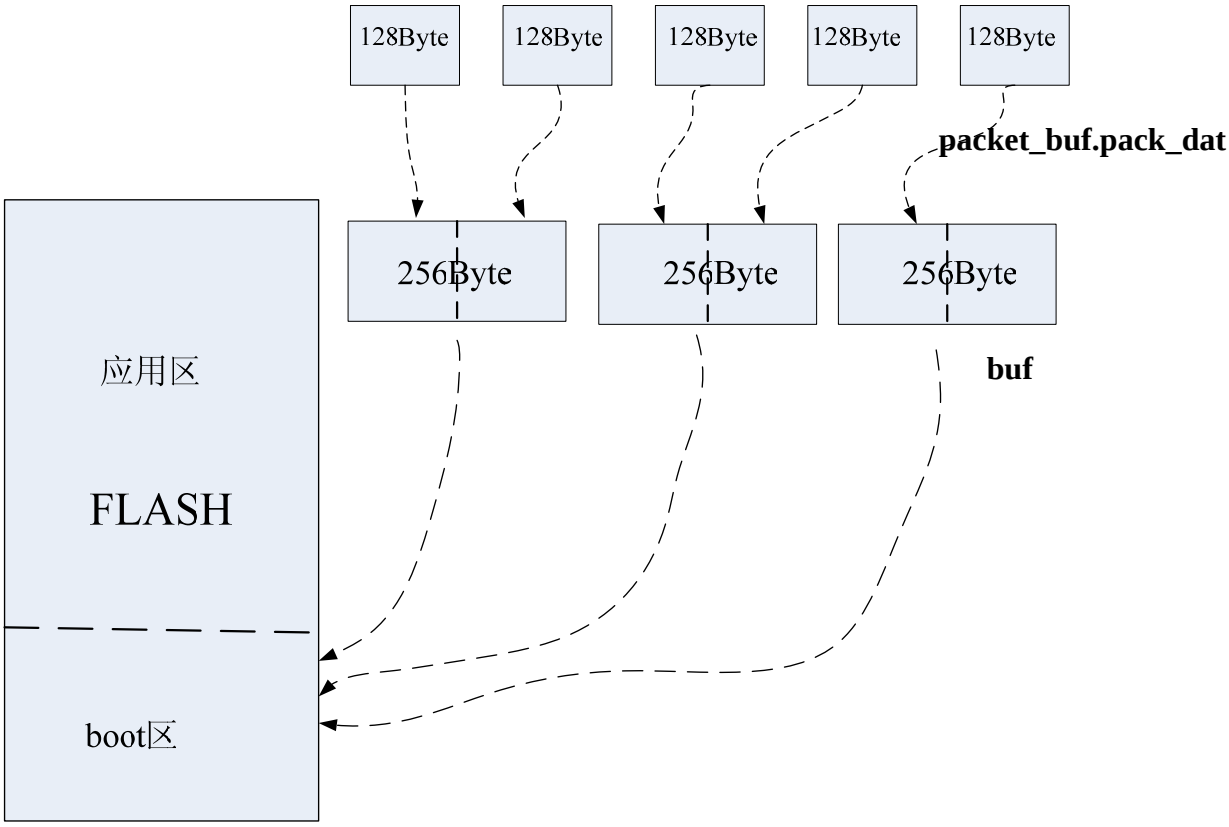
## 3、检包

检包：收到一包信息后开始检查包的正确性。
1、 包头数据是否等于包头值
2、 包号是否和预期包号相等。因为包号开始从 1 开始，到 255 时，再增则从 0 开始（以后都从 0 开始）。所以本例在程序开了一全局变量 uchar buf_num=1;//预期包号
3、 反包号与包号的和是否为 255
4、 crc 校验是否正确.
如果以上检测都正确表示正确接收一包信息，装包,发送 ACK 应答信号，请求主机发送下包。否则检包错误，即向主机发送 NAK 重发信号,如出现检包错误达 10 次，即向主机发送 CAN 中止信号。退出到应用代码区。

## 4、装包

装包:由于 flash 写函数的 buf 参数长度 SPM_PAGESIZE=256byte，而 Xmodem 协议数据域为 128 字节。所以每收到两包才能写 flash 一次,即调用 boot_program_page 函数一次。但如

果应用程序数据总量恰好为奇数个包，则在收到 EOT 信号时，要补一包数据到 buf 中，写入到 flash 中，防止最后一包没写入。



# 上位机操作

## 1、超级终端配置

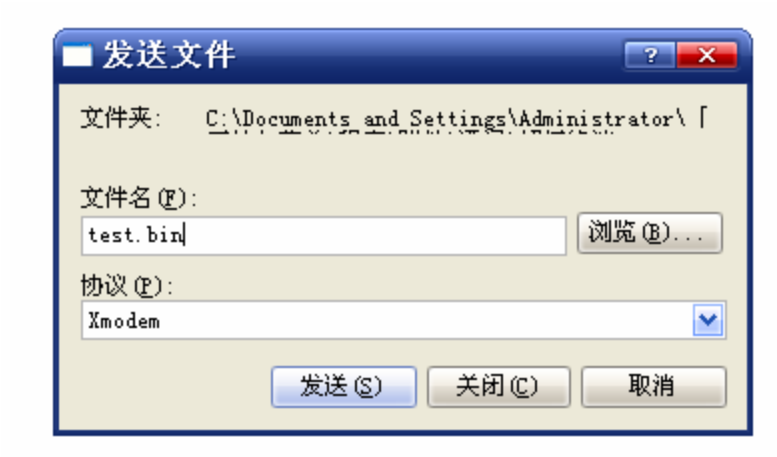

## 2、发送文件

点击传送，选择发送文件，载入纯二进制代码，协议选择 Xmodem。

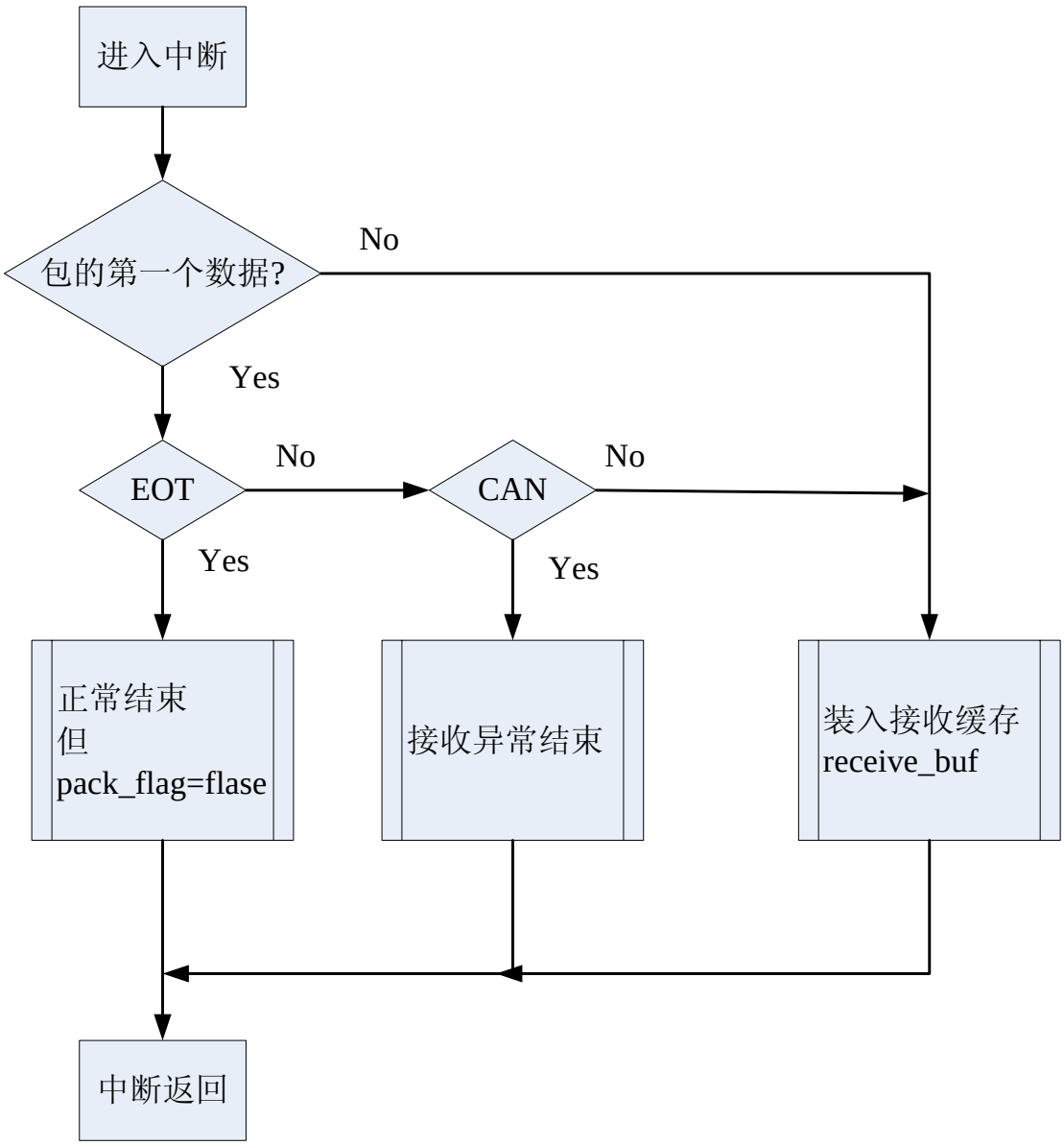
如果想用 WinAVR 生成二进制代码，即修改 Makefile 以下的地方

```
# Output format. (can be srec, ihex, binary)
FORMAT = binary
```

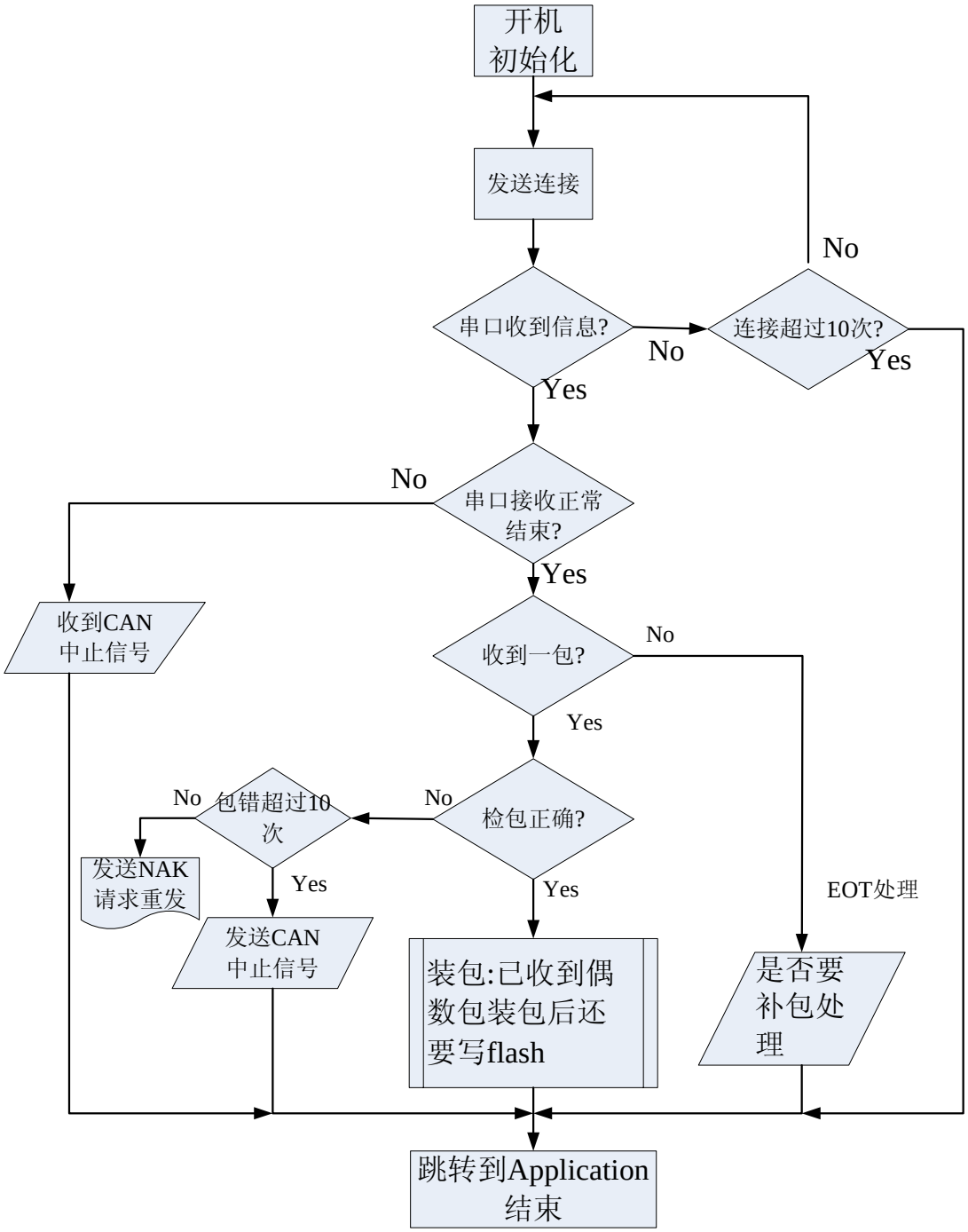
生成文件，虽然后缀为.hex，其实是二进制文件，把后缀改为.bin 即可。


发送文件
文件夹:
C:\Documents and Settings\Administrator\
文件名(F):
test.bin
浏览(B)...
协议(P):
Xmodem
发送(S)
关闭(C)
取消

## 串口接收 ISR 流程


进入中断
包的第一个数据?
No
Yes
EOT
No
CAN
No
Yes
Yes
正常结束
但
pack_flag=flase
接收异常结束
装入接收缓存
receive_buf
中断返回

# 主程序流程图




## 完整代码

将代码中的打印信息适量减少，或缩短，代码可缩小至 2k 内。