STM32 串口 ISP 攻略

作者：ronger
原文链接：https://ld246.com/article/1639620542536
来源网站：链滴

1 STM32串口ISP原理

STM32单片机系统存储器中有一段Bootloader代码，他的主要作用为通过串口下载程序代码到单片内部Flash中。该过程称为串口ISP，也就是说串口与Bootloader进行通信，就可以实现代码下载功能，下面详细讲解与Bootloader的通信规则。

2 Bootloader通信规则

2.1 启动bootloader

与Bootloader通信前，需要先启动他，也就是需要让单片机进入到Bootloader到代码中执行。通过SM32的两个引脚可以实现配置。如下图所示：

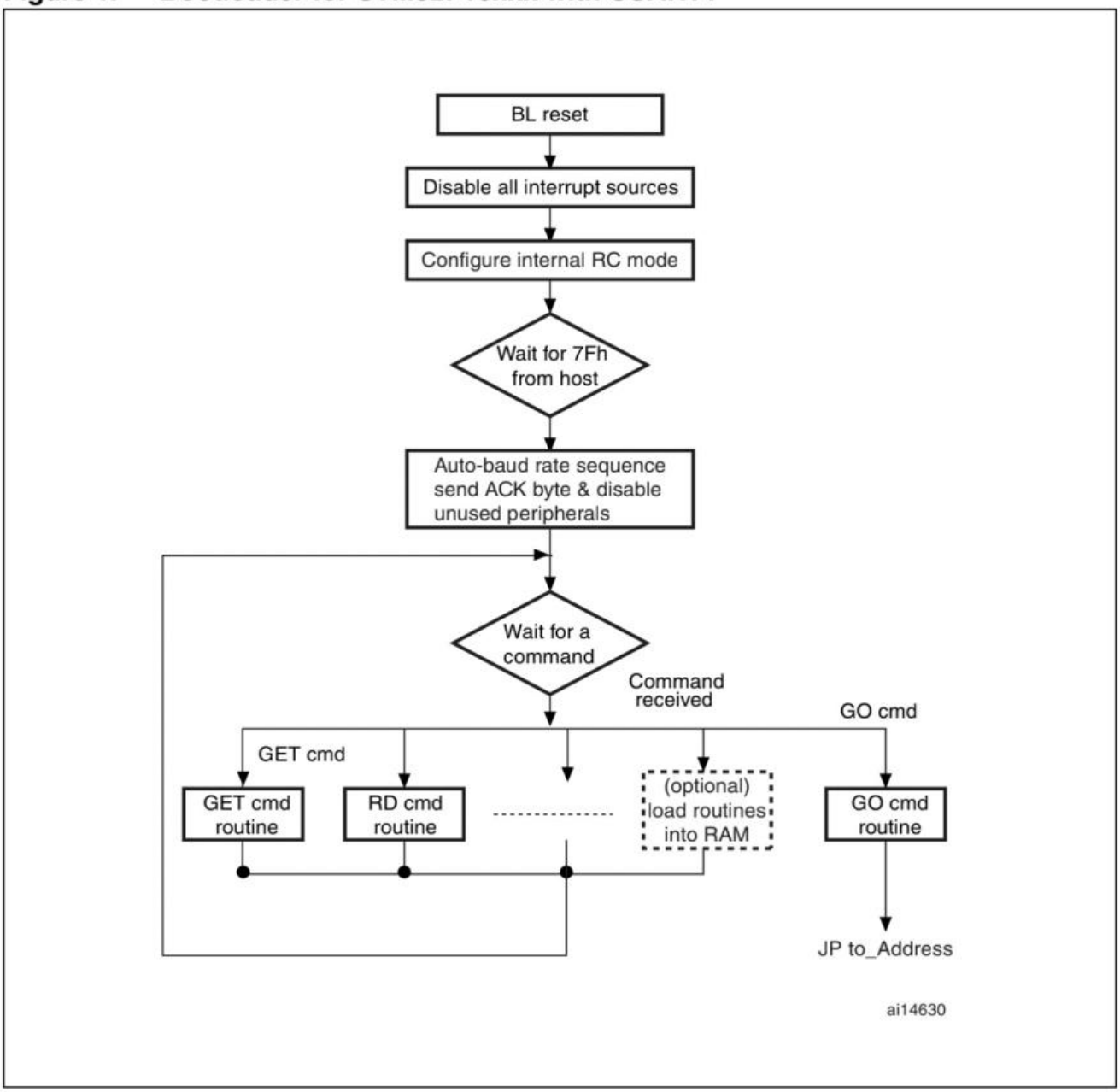
Table 1. Boot pin configuration

Boot mode selection pins		Boot mode	Aliasing
BOOT1	BOOT0		
X	0	User Flash memory	User Flash memory is selected as the boot space
0	1	System memory	System memory is selected as the boot space
1	1	Embedded SRAM	Embedded SRAM is selected as the boot space

如上图所示，可以配置成三种模式，第二种为启动Bootloader方式，即BOOT0=1，BOOT1=0。

2.2 Bootloader编码序列

Figure 1. Bootloader for STM32F10xxx with USART1



当STM32F10xx系列单片机进入Bootloader程序后，单片机开始检测PA10(串口1接收引脚)是否收到口数据0x7F，并且要求串口数据格式为1bit起始+8bit数据+1bit偶校验+1bit停止位。

收到0x7F后，单片机将内部串口的波特率设置和发送数据方相同，并且给主机回复应答信号0x79。方便描述，我们这里将发送数据方称为主机，STM32单片机方称为从机。回复完应答信号0x79后，示从机已经准备好接收主机的指令了。

2.3 波特率选择

为保证数据传输的正确性，一般选择在1200bps至115200bps之间。

2.4 Bootloader指令集

完整指令集如下图，但并不是单片机同时支持所有的指令，后续将逐一介绍。

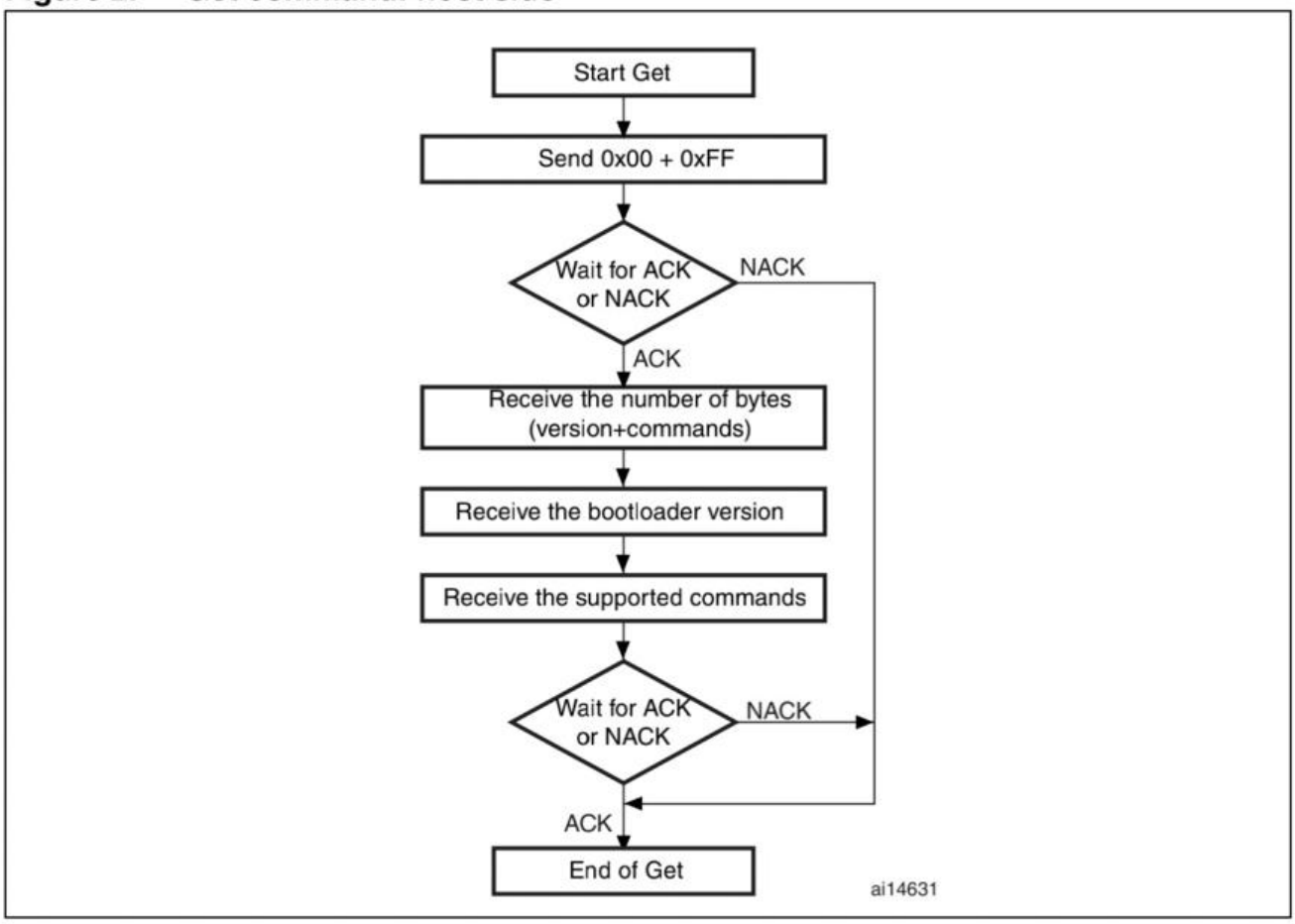
Table 2. USART bootloader commands

Command(1)	Command code	Command description
Get(2)	0x00	Gets the version and the allowed commands supported by the current version of the bootloader
Get Version & Read Protection Status(2)	0x01	Gets the bootloader version and the Read Protection status of the Flash memory
Get ID(2)	0x02	Gets the chip ID
Read Memory(3)	0x11	Reads up to 256 bytes of memory starting from an address specified by the application
Go(3)	0x21	Jumps to user application code located in the internal Flash memory or in SRAM
Write Memory(3)	0x31	Writes up to 256 bytes to the RAM or Flash memory starting from an address specified by the application
Erase(3)(4)	0x43	Erases from one to all the Flash memory pages
Extended Erase(3)(4)	0x44	Erases from one to all the Flash memory pages using two byte addressing mode (available only for v3.0 usart bootloader versions and above).
Write Protect	0x63	Enables the write protection for some sectors
Write Unprotect	0x73	Disables the write protection for all Flash memory sectors
Readout Protect	0x82	Enables the read protection
Readout Unprotect(2)	0x92	Disables the read protection

2.4.1 Get 命令

该指令允许主机获取从机的Bootloader版本号以及支持哪些命令。当从机收到主机发来的Get命令后从机将Bootloader版本及所支持的命令发送给主机。发送Get命令的主机端时序如下图所示。

Figure 2. Get command: host side



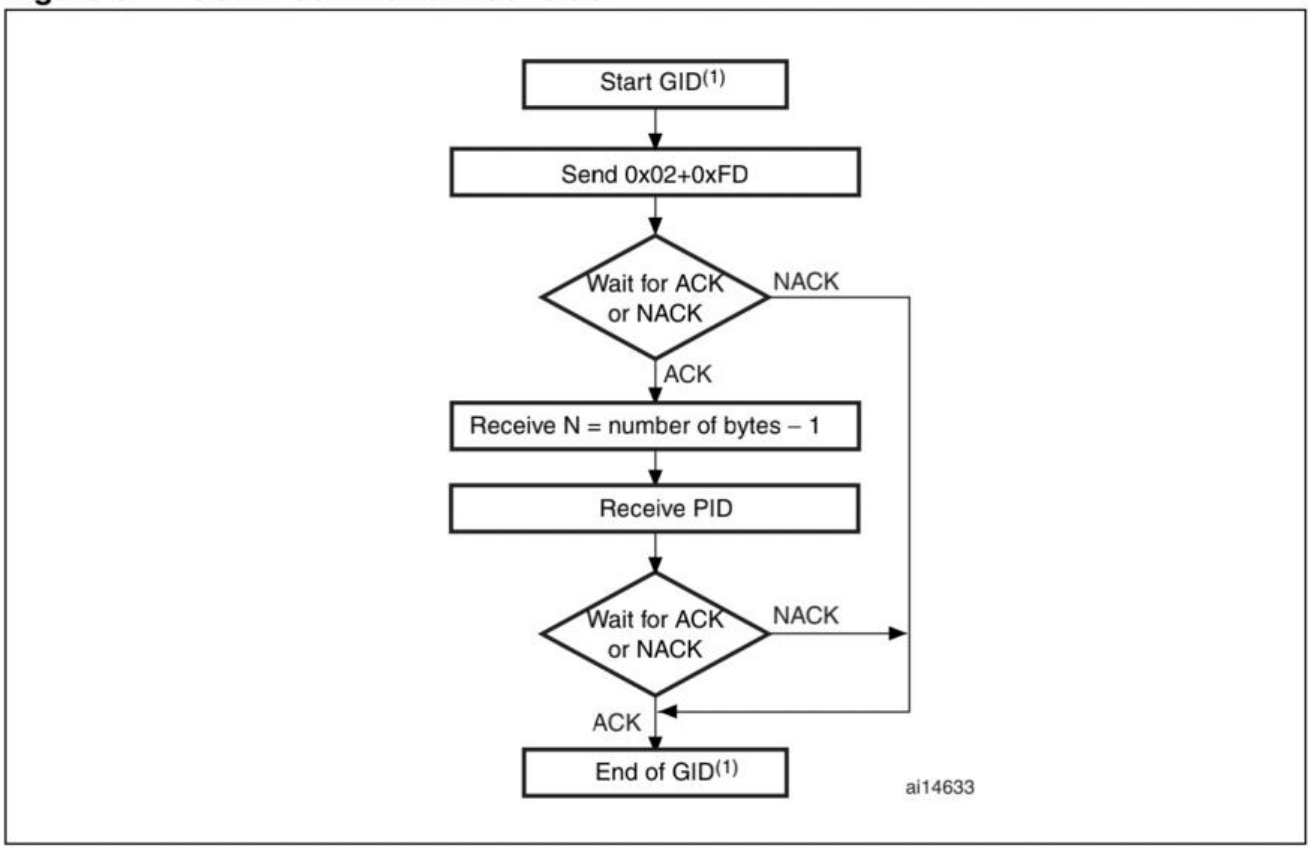
代码实现如下：

```
char GetCMD(DeviceInfo_t *DInfo)
{
  char error,len;

  //step1 发送指令0x00,0xFF,并等待ACK
  SendByte(0x00);
  SendByte(0xFF);
  error = ACK();if(!error) return 0;

  //step2 接收数据
  len = GetByte();  DInfo->cmd_count = len;//第一个字节为命令数量N
  DInfo->bootloaderversion = GetByte();//第二个字节为版本号
  for(int i =0;i<len;i++) DInfo->cmd[i] = GetByte();//获取CMD
  error = ACK();if(!error) return 0;

   //step3 处理数据，并打印版本号，命令等
   //打印数据，并清空串口缓存
   return 1;
}
```

2.4.2 Get ID命令

该命令用于获取单片机的的产品ID，主机端时序如下图所示。

Figure 6. Get ID command: host side



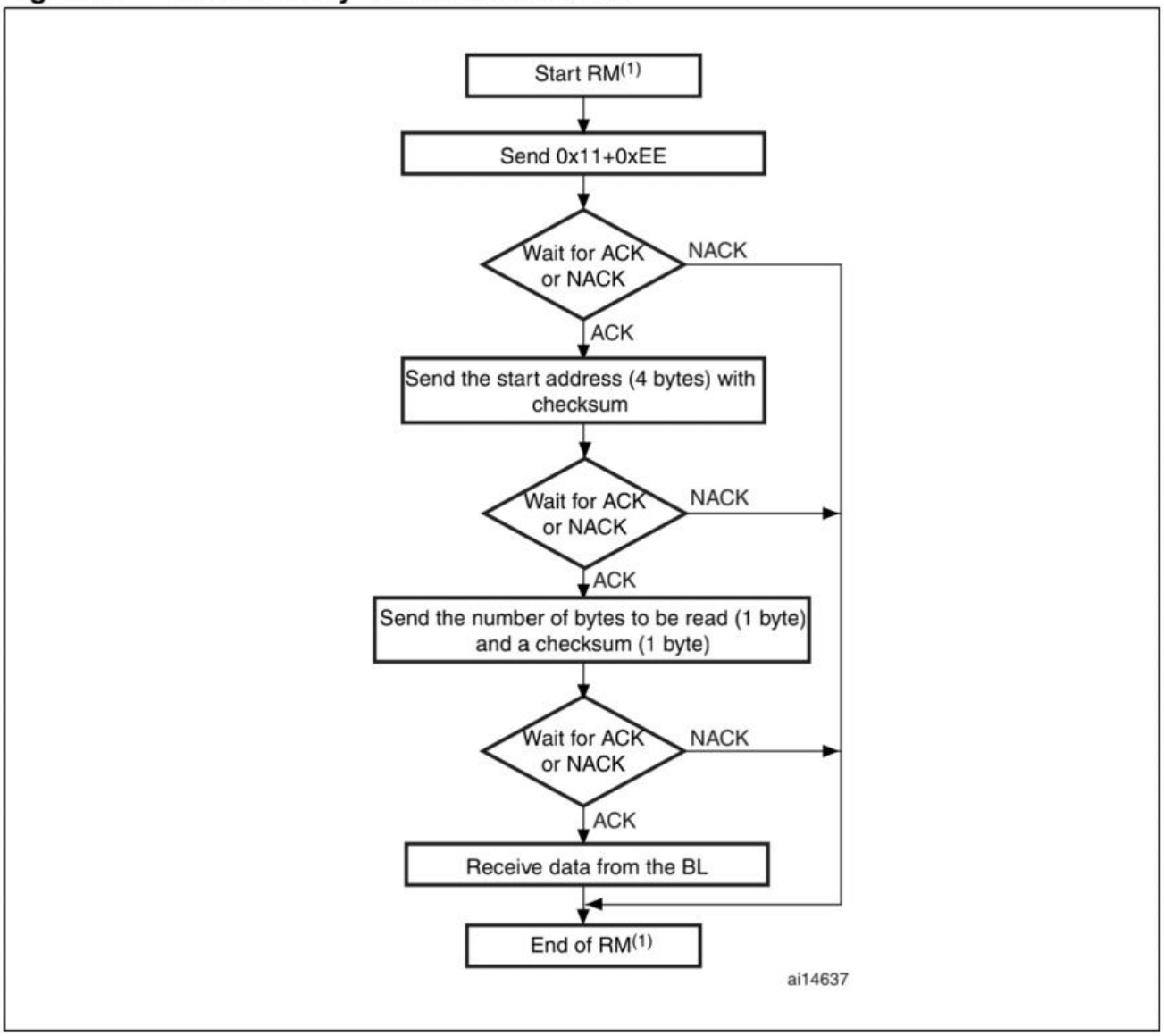
代码实现如下：

```
char GetID(DeviceInfo_t *DInfo)
{
   char len,error;

   //step1 发送序列0x02,0xfd，并等待ACK
   SendByte(0x02);
   SendByte(0xFD);
   error = ACK();if(!error) return 0;

   //step2 接收PID，并等待ACK
   len = GetByte();//第一个字节为ID长度N-1
   for(int i =0;i<=len;i++) DInfo->PID[i] = GetByte();//获取ID
   error = ACK();if(!error) return 0;

   return 1;
   //step3 清空接收缓存，并打印PID
}
```

2.4.3 Read Memory命令

读内存命令用来读取从机单片机内部RAM，FLASH，信息块（包括系统存储区和选项字节）中的数。主机端时序如下所示。

Figure 8. Read Memory command: host side



1. RM = Read Memory.

代码实现如下：

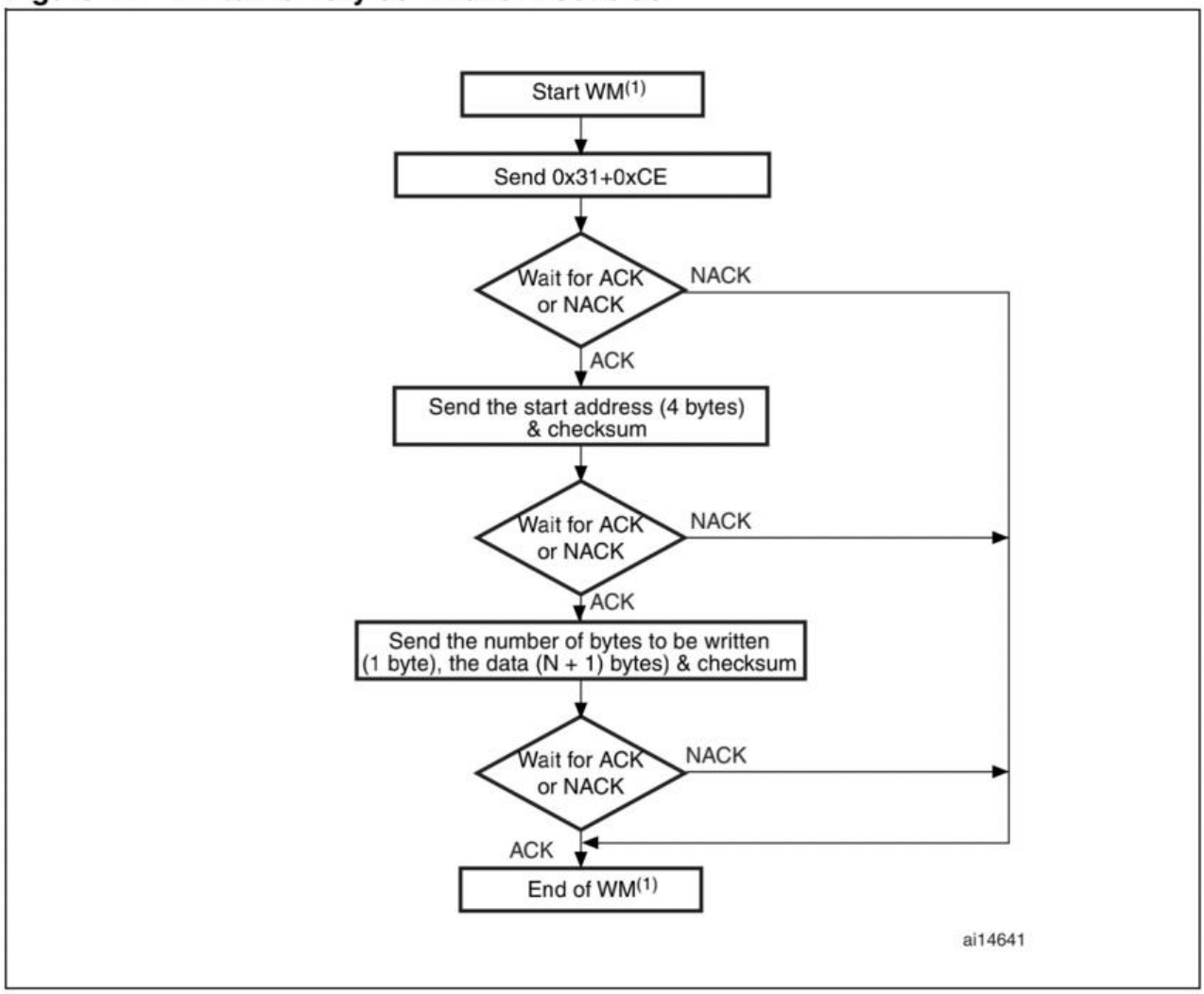
```
//////////////////////////////////////////////////
//
//读取数据
//与写数据步骤类似
//addr必须能被4整除，len发送数据长度-1，单次不能超过256B
//////////////////////////////////////////////////
char ReadMem(unsigned char *data, unsigned int addr, unsigned char len)
{
    unsigned char temp[4],error;      //保存addr的四个字节
    int i;
    temp[0] = ((addr>>24) & 0xFF);
    temp[1] = ((addr>>16) & 0xFF);
    temp[2] = ((addr>> 8) & 0xFF);
    temp[3] = ((addr    ) & 0xFF);
    //step1 发送序列0x11,0xEE，并等待ACK
    SendByte(0x11);
```

```
    SendByte(0xEE);
    error = ACK();if(!error) return 0;
    //step2 发送地址，先发高字节
    SendByte(temp[0]);
    SendByte(temp[1]);
    SendByte(temp[2]);
    SendByte(temp[3]);
    //step3 发送地址校验，并等待ACK
    SendByte(CheckSum(temp, 4));
    error = ACK();if(!error) return 0;
    //step4，发送len及校验，并等待ACK
    SendByte(len);
    SendByte(~len);
    error = ACK();if(!error) return 0;
    //step4 接收长度为len+1的数据
    for(i=0;i<=len;i++) data[i] = GetByte();

    return 1;
}
```

2.4.4 Write Memory命令

写内存命令用来往从机单片机内部RAM，FLASH，信息块（包括系统存储区和选项字节）中写入数，主机端时序如下所示。

Figure 12. Write Memory command: host side



1. WM = Write Memory.

If the start address is invalid, the command is NACKed by the device.

往选择字区写入数据时，开始地址必须为0x1FFFF800。

代码实现如下：

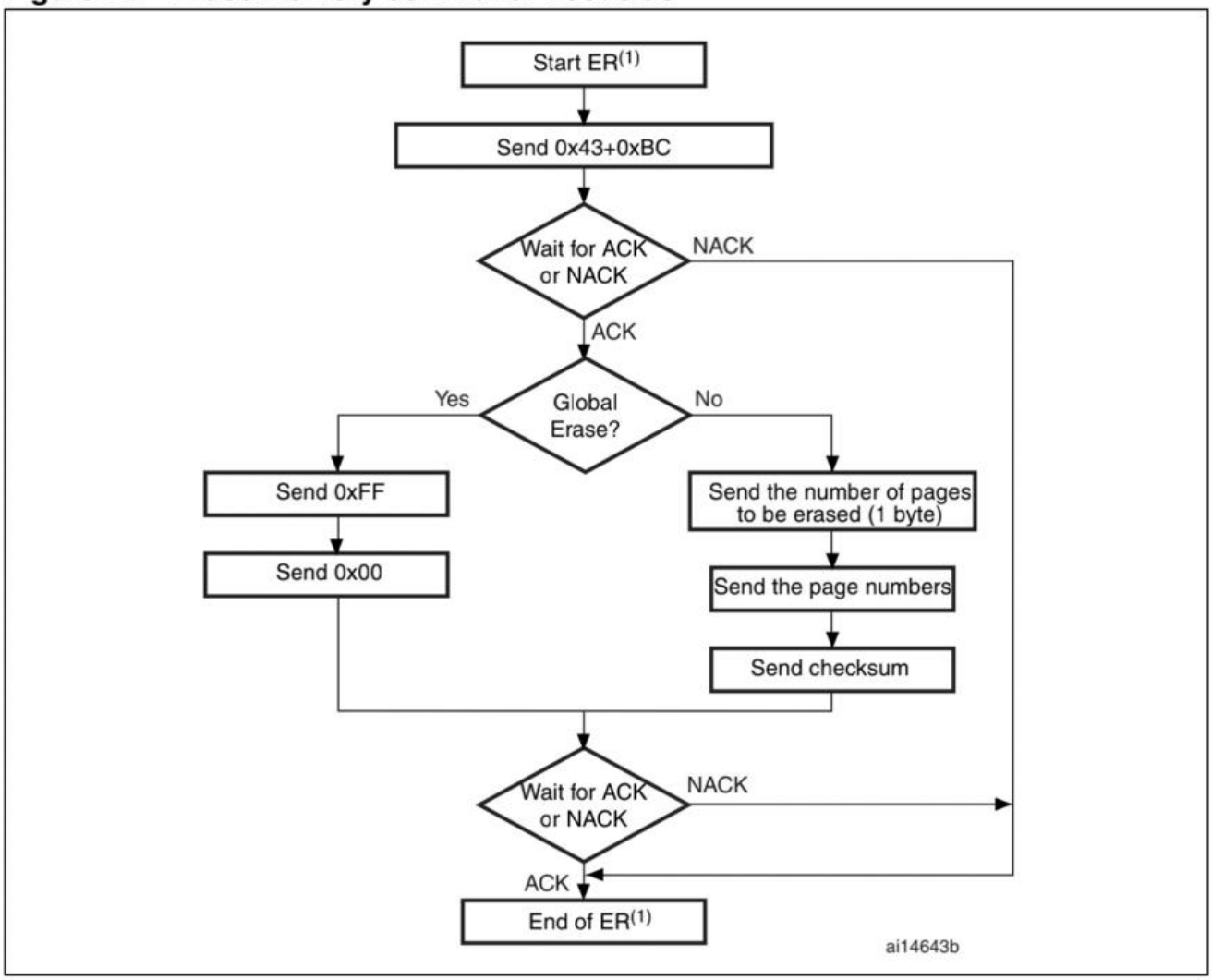
```
//////////////////////////////////////////////////
//
//写入数据块，从*data处，往stm32的addr处，写入len+1字节数据
//
//////////////////////////////////////////////////
char WriteMem(unsigned char *data, unsigned int addr, unsigned char len)

{
   unsigned char temp[4],error;     //保存addr的四个字节
   int i;
   temp[0] = ((addr>>24) & 0xFF);
   temp[1] = ((addr>>16) & 0xFF);
   temp[2] = ((addr>>8 ) & 0xFF);
   temp[3] = ((addr    ) & 0xFF);

   //step1 发送序列0x31,0xCE,并等待ACK
```

```
    SendByte(0x31);
    SendByte(0xCE);
    error = ACK();if(!error) return 0;
    //step2 发送地址，先发高字节
    SendByte(temp[0]);
    SendByte(temp[1]);
    SendByte(temp[2]);
    SendByte(temp[3]);
    //step3 发送地址校验，并等待ACK
    SendByte(CheckSum(temp, 4));
    error = ACK();if(!error) return 0;
    //step4 发送len
    SendByte(len);
    //step5 连续发送数据，最后字节为校验，并等待ACK
    for(i=0;i<=len;i++)
    {
        SendByte(data[i]);
    }
    SendByte(len ^ CheckSum(data, len+1));
    //delay_msec(3000);
    error = ACK();if(!error) return 0;

   //清空串口缓存，并延时1s.
    return 1;
}
```

2.4.5 Erase Memory命令

擦除从机单片机内部FLASH命令，仅单片机Bootloader版本3.0以下支持该命令。主机端时序如下所。

Figure 14. Erase Memory command: host side



1. ER = Erase Memory.

全片擦除代码实现如下：

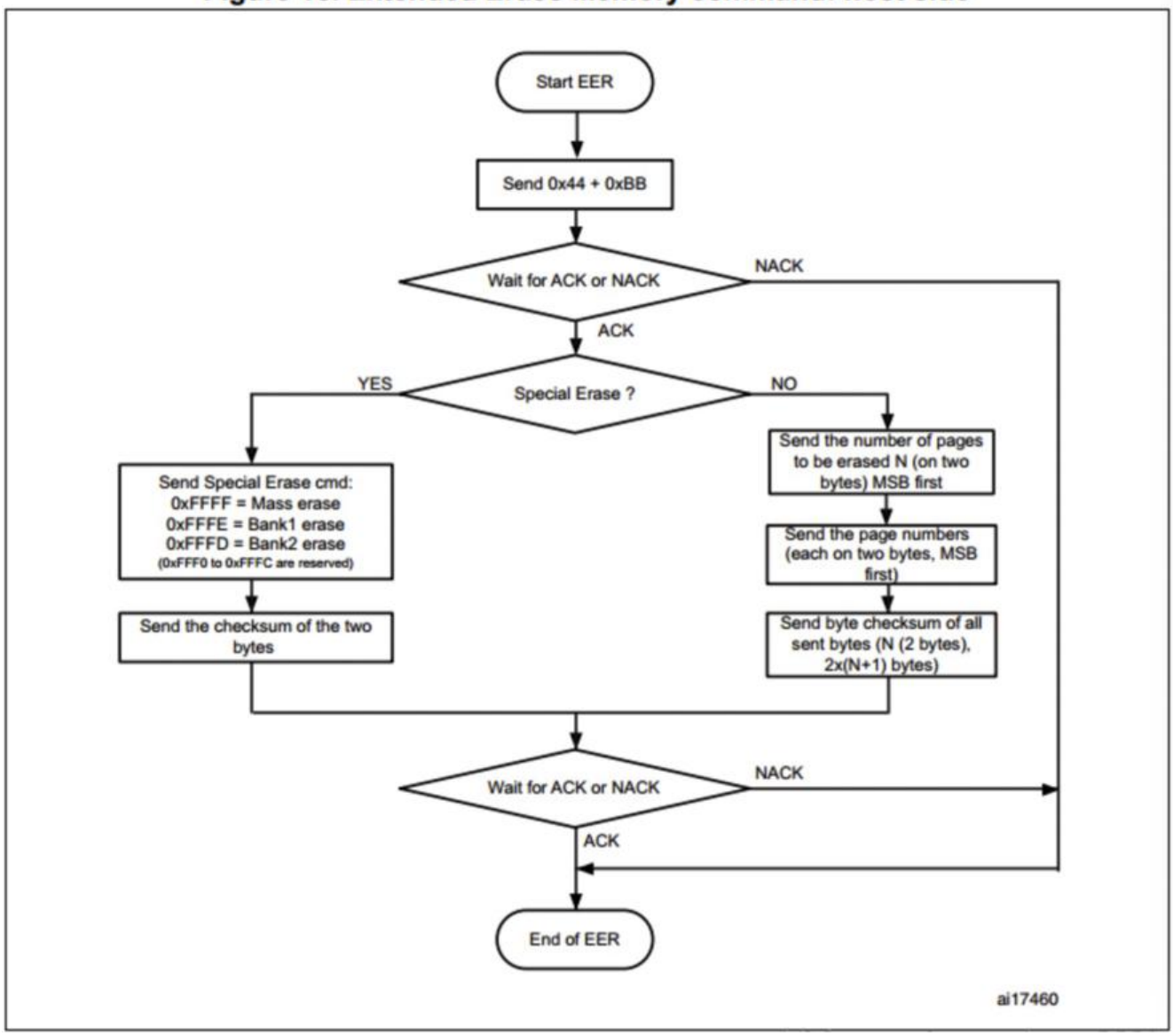
```
//////////////////////////////////////////////////
//
//全片擦除,bootloader V3.0以下有效
// step1 发送序列0x43,0xbc,并等待ACK
// step2 发送序列0xff,0x00,并等待ACK
// step3 清空串口接收缓冲
//////////////////////////////////////////////////
char EraseAll()
{
  char error;
   SendByte(0x43);//
   SendByte(0xBC);
   error = ACK();if(!error) return 0;
   SendByte(0xFF);
   SendByte(0x00);
   //不同的产品，全片擦除的时间长短不一，500ms的时间不一定够，
   //因此不能用现成的ACK函数，需重写如下：
   //error = ACK();if(!error) return 0;
   while(!MyComRevBUff.size())
```

```
    {
        delay_msec(1);//出让线程
    }
    if(0x79 == GetByte()) return 1;
    else                  return 0;
}
```

2.4.6 Extended Erase Memory命令

仅单片机Bootloader版本3.0及以上支持该命令。主机端时序如下所示，以Erase Memory命令相比该命令为双字节命令，即Erase cmd为双字节，例如0xFFFF。

Figure 16. Extended Erase Memory command: host side



全片擦除代码实现如下：

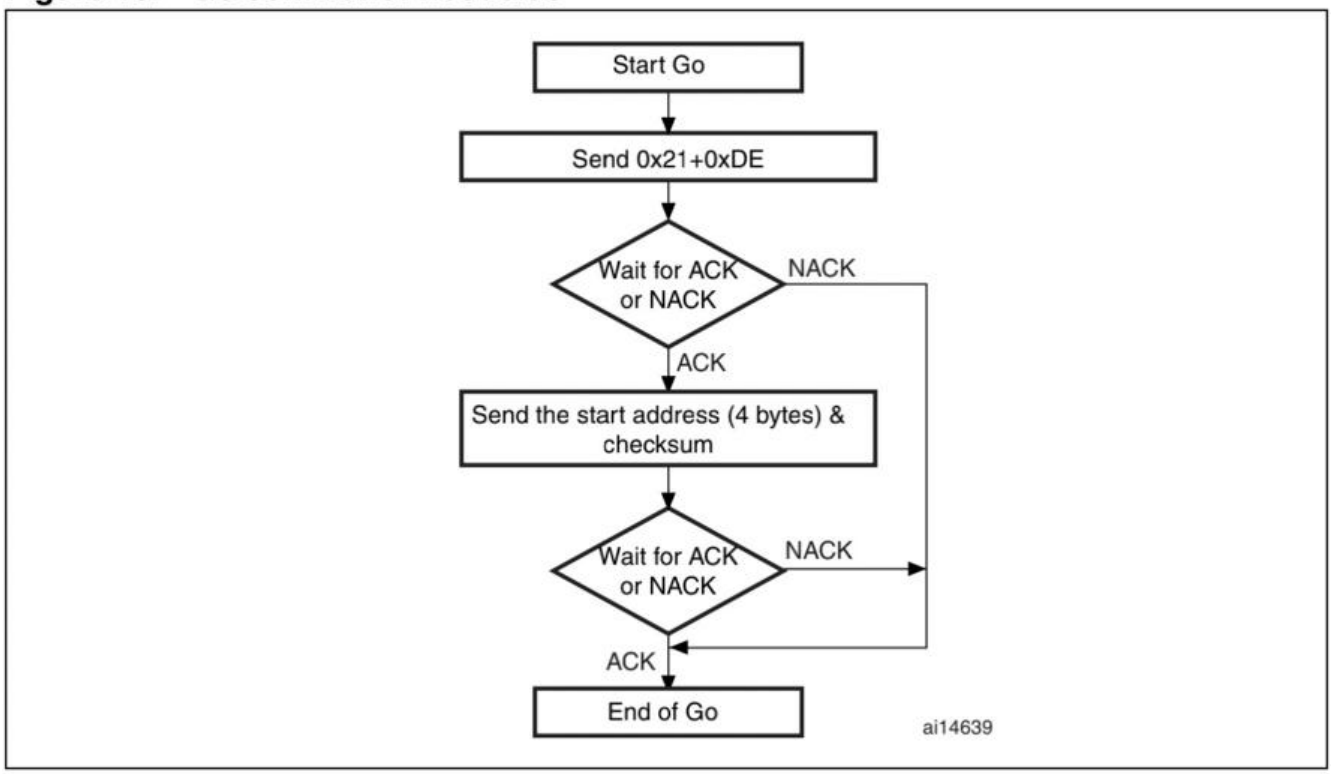
```
//////////////////////////////////////////////////
//
//全片擦除,bootloader V3.0及以上有效
// step1 发送序列0x43,0xbc,并等待ACK
```

```
// step2 发送序列0xff,0x00,并等待ACK
// step3 清空串口接收缓冲
//////////////////////////////////////////////////
char ExtendedEraseAll()
{
   char error;
    SendByte(0x44);//
    SendByte(0xBB);
    error = ACK();if(!error) return 0;

    unsigned char EraseCMD[2];
    EraseCMD[0] = 0xFF;
    EraseCMD[1] = 0xFF;

    SendByte(0xFF);//写入地址0xFFFF
    SendByte(0xFF);

    SendByte(CheckSum(EraseCMD, 2));//双字节校验

    //不同的产品，全片擦除的时间长短不一，500ms的时间不一定够，
    //因此不能用现成的ACK函数，需重写如下：
    //error = ACK();if(!error) return 0;
    while(!MyComRevBUff.size())
    {
        delay_msec(1);//出让线程
    }
    if(0x79 == GetByte()) return 1;
    else                  return 0;
}
```

2.4.7 GO命令

跳转到从机的指定地址开始执行程序。G0命令主机时序如下。

Figure 10. Go command: host side



代码实现如下：

```
////////////////////////////////////////////////
//
//跳转执行指令,下载完成后，跳转到RAM或内部FLASH执行
//
//
////////////////////////////////////////////////
char CMDGo(unsigned int addr)
{
    unsigned char temp[4],error;     //保存addr的四个字节
    int i;
    temp[0] = ((addr>>24) & 0xFF);
    temp[1] = ((addr>>16) & 0xFF);
    temp[2] = ((addr>>8 ) & 0xFF);
    temp[3] = ((addr    ) & 0xFF);

    //step1 发送序列0x21,0xDE,并等待ACK
    SendByte(0x21);
    SendByte(0xDE);
    error = ACK();if(!error) return 0;
    //step2 发送地址，先发高字节
    SendByte(temp[0]);
    SendByte(temp[1]);
    SendByte(temp[2]);
    SendByte(temp[3]);
    //step3 发送地址校验，并等待ACK
    SendByte(CheckSum(temp, 4));
    error = ACK();if(!error) return 0;
```

```
    return 1;
}
```

3 STM32串口ISP操作参考步骤

- step1 主机发送0x7F，启动从机并等待接收主机命令；
- step2 主机发送Get 命令，获取从机Bootloader版本号及所支持的命令等；
- step3 主机发送Get ID命令，获取从机产品ID；
- step4 主机发送擦除命令，擦除从机内部FLASH，为写数据做准备；
- step5 主机发送Write Memory命令，将代码写入从机器；
- step6 主机发送Read Memory命令，读取从机代码并与写入的数据进行对比校验；
- step7 主机发送Go命令，从机跳转至程序开始处执行代码，观察现象是否与程序一致。
- step8 结束。