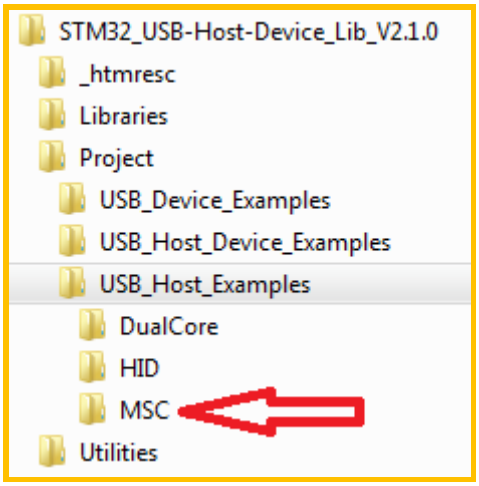
OTG 做 U 盘主机的兼容性提高

问题：

某客户使用 STM32F2 的 OTG 库中的 U 盘主机例程在连接 U 盘时，有些 U 盘不能识别，甚至出现操作死机的情况。现就针对版本 V2.1.0 的 USB 主机库中的 MSC Host 例程做一些修改，以能够兼容更 U 盘。

Part Number	Version	Marketing Status	Order From ST
STSW-STM32046	2.1.0	Active	Download

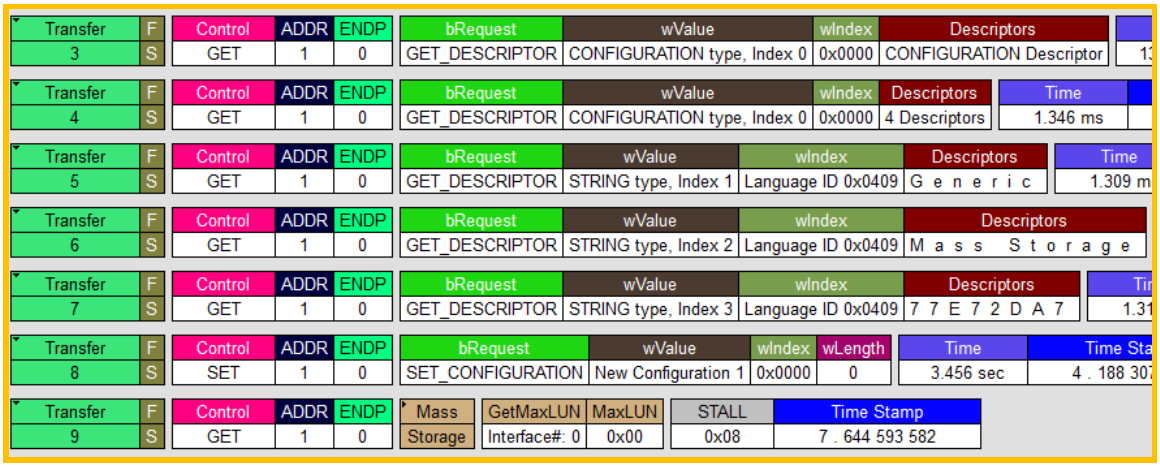


调研：

1、**有些 U 盘在收到 BOT_RESET 这个 MSC 类相关命令时，就不再有反应了：设备一直对后续主机发 来的 IN 令牌回复 NAK。**通过对 BOT_RESET 命令的调查得知，USB 规范为大容量设备定义了两种 类型的复位：USB 端口复位和大容量设备 BOT 复位。USB 规范并未规定这两种复位对应到 SCSI 命令中是做什么操作，因此也没有规定在收到这些命令后 SCSI 的操作。通常设备把 USB 端口复 位映射成 SCSI 的硬复位；有些设备把 BOT 复位映射到 hard reset，有的仅是映射成逻辑单元复位。该条命令通常用于主机对在 BOT 通信中出错的设备进行复位恢复。但是需要指出的是，有些 U 盘并没有完全遵守大容量规范，比如就不实现对 BOT_RESET 命令的支持。在这种情况下，设备一般会回复 STALL，那么主机需要发送 Set Port Feature（PORT_RESET）来复位设备所连的 Hub 端口。但是这个 U 盘并没有回复 STALL，且 U 盘是直接连在 STM32 USB 主机上的，直接 并没有 Hub。于是，参照该 U 盘在 Window 下的连接操作的过程文件（tracer file），发现在 枚举完成后，Windows 并没有发送 BOT_RESET 命令，而是直接就发送 GET_Max_Lun 命令来获取 U 盘的逻辑盘符个数。于是在 STM32 的 USB Host 例程中注释掉 BOT_RESET 命令，果然就可以正确操作了。修改代码如下：

```
<usbh_msc_core.c>
USBH_MSC_Handle()
{
switch(USBH_MSC_BOTXferParam.MSCState)
{
case   USBH_MSC_BOT_INIT_STATE:
USBH_MSC_Init(pdev);
//USBH_MSC_BOTXferParam.MSCState = USBH_MSC_BOT_RESET;
USBH_MSC_BOTXferParam.MSCState = USBH_MSC_GET_MAX_LUN;
break;
case...
```

2、**还有些 U 盘在收到 Get_Max_Lun 命令返回 STALL，但是 U 盘主机就走不下去了。**抓到的 tracer 文件如图



Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	
3	S	GET	1	0	GET_DESCRIPTOR	CONFIGURATION type, Index 0	0x0000	CONFIGURATION Descriptor	1

Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time	
4	S	GET	1	0	GET_DESCRIPTOR	CONFIGURATION type, Index 0	0x0000	4 Descriptors	1.346 ms	

Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time
5	S	GET	1	0	GET_DESCRIPTOR	STRING type, Index 1	Language ID 0x0409	G e n e r i c	1.309 m

Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors
6	S	GET	1	0	GET_DESCRIPTOR	STRING type, Index 2	Language ID 0x0409	M a s s S t o r a g e

Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Ti
7	S	GET	1	0	GET_DESCRIPTOR	STRING type, Index 3	Language ID 0x0409	7 7 E 7 2 D A 7	1.31

Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	wLength	Time	Time Sta
8	S	SET	1	0	SET_CONFIGURATION	New Configuration 1	0x0000	0	3.456 sec	4 . 188 307

Transfer	F	Control	ADDR	ENDP	Mass	GetMaxLUN	MaxLUN	STALL	Time Stamp
9	S	GET	1	0	Storage	Interface#: 0	0x00	0x08	7 . 644 593 582

经查阅，有些只包含唯一一个逻辑盘符的 U 盘可以对该命令回复数值 0 或者直接回复 STALL。 那么对于 USB 主机来说，在收到了该条命令的 STALL 回复后就应该第一：认为该 U 盘仅包含 一个逻辑盘符；第二：对 STALL 应答进行处理，然后继续下面的命令流程。

通过 USB2.0 协议规范（章节 9.2.7），我们得知在控制传输（Control Transfer）过程中，当设 备收到的命令自己不支持或者不适合设备当前的设置，就认为是命令出错。那么设备通过在接 下来的数据阶段或者状态阶段回复 STALL 应答来告知主机这个错误。这种“协议 STALL”是 控制传输特有的；这样 STALL 的状态，会在下一个控制传输（Setup 令牌）的到来而解除。 我们看看 Window 对 U 盘这样的回复是怎么处理，tracer 文件抓图如下：主机通过 Clear Feature 命令，参数 EP_Halt（这是一个控制传输）来把设备方端点的 Halt feature 清除掉。

从代码里可以看到，例程是有做这方面的处理的：如果主机发送的 Get_Max_Lun 命令不被设 备支持，则将 MSCState 状态设定成 CTRL_ERROR_STATE，发送 Clear Feature 的命令，以及随后 的 Test_Unit_Ready 命令。

```
<usbh_msc_core.c>
USBH_MSC_Handle()
{
switch(USBH_MSC_BOTXferParam.MSCState)
{
case USBH_MSC_GET_MAX_LUN:
status = USBH_MSC_GETMaxLUN(pdev, phost);
if(status == USBH_OK )
{ ……,
USBH_MSC_BOTXferParam.MSCState = USBH_MSC_TEST_UNIT_READY;
}
if(status == USBH_NOT_SUPPORTED )
{ ……,
USBH_MSC_BOTXferParam.MSCStateBkp = USBH_MSC_TEST_UNIT_READY;
USBH_MSC_BOTXferParam.MSCState = USBH_MSC_CTRL_ERROR_STATE;}
break ;
case    USBH_MSC_CTRL_ERROR_STATE:
USBH_ClrFeature(pdev, phost, 0x00, pphost->Control.hc_num_out);
……
USBH_MSC_BOTXferParam.MSCState    =    USBH_MSC_BOTXferParam.MSCStateBkp;
……
```

但是从抓到的 tracer 文件却看不到主机走到了发送 Test_Unit_Ready 的命令。经过调试、代码跟踪，终于定位在 USBH_HandleControl()中。对 CTRL_DATA_IN_WAIT 的处理，当收到 URB_STALL 的应答后，**没有给 phost 的 Control.state 赋值成 CTRL_STALLED！**

```
USBH_Status USBH_HandleControl (pdev, phost)
switch (phost->Control.state)
case CTRL_SETUP:（发出 SETUP token）跳到 CTRL_SETUP_WAIT
case CTRL_SETUP_WAIT: 可以跳到 CTRL_DATA_IN、CTRL_DATA_OUT、CTRL_
case CTRL_DATA_IN:（发出 IN token），跳到 CTRL_DATA_IN_WAIT
case CTRL_DATA_IN_WAIT:
        URB_Status:
        >> URB_DONE:phost->Control.state = CTRL_STATUS_OUT 跳到 CTRL_STATUS_OUT
        >> URB_STALL:phost->gState = phost->gStateBkp;(回到之前的状态机状态)，！！！！！
        >> URB_ERROR:phost->Control.state = CTRL_ERROR;
case CTRL_DATA_OUT:（发出 DATA0/1 TOKEN），跳到 CTRL_DATA_OUT_WAIT
case CTRL_DATA_OUT_WAIT:
        URB_Status:
        >> URB_DONE:phost->Control.state = CTRL_STATUS_IN;跳到 CTRL_STATUS_IN
        >> URB_STALL:phost->gState = phost->gStateBkp; phost->Control.state = CTRL_STALLED;
        >> URB_NOTREADY:phost->Control.state = CTRL_DATA_OUT;继续重发
        >> URB_ERROR:phost->Control.state = CTRL_ERROR;
case CTRL_STATUS_IN:（发出 0 字节的 IN 包），跳到 CTRL_STATUS_IN_WAIT
case CTRL_STATUS_IN_WAIT:
        URB_Status:
        >> URB_DONE:phost->gState =  phost->gStateBkp;phost->Control.state = CTRL_COMPLETE;
        >> URB_ERROR:phost->Control.state = CTRL_ERROR;
        >> URB_STALL:phost->gState = phost->gStateBkp; phost->Control.status = CTRL_STALL?;
                     status = USBH_NOT_SUPPORTED;
case CTRL_STATUS_OUT:（发出 0 字节的 OUT 包），跳到 CTRL_STATUS_OUT_WAIT
case CTRL_STATUS_OUT_WAIT:
        >> URB_DONE:phost->gState =  phost->gStateBkp;phost->Control.state = CTRL_COMPLETE;
        >> URB_NOTREADY:phost->Control.state = CTRL_STATUS_OUT;继续重发
        >> URB_ERROR:phost->Control.state = CTRL_ERROR;
case CTRL_ERROR
```

在这里 STALL，但是忘了把 Control.state = CTRL_STALLED

3、还有一点需要注意的是：本版的 U 盘主机例程，暂时不支持多盘符 U 盘。

重要通知 – 请仔细阅读

意法半导体公司及其子公司（“ST”）保留随时对ST 产品和/ 或本文档进行变更、更正、增强、修改和改进的权利，恕不另行通知。买方在订货之前应获取关于ST 产品的最新信息。ST 产品的销售依照订单确认时的相关ST 销售条款。

买方自行负责对ST 产品的选择和使用， ST 概不承担与应用协助或买方产品设计相关的任何责任。

ST 不对任何知识产权进行任何明示或默示的授权或许可。

转售的ST 产品如有不同于此处提供的信息的规定，将导致ST 针对该产品授予的任何保证失效。

ST 和ST 徽标是ST 的商标。所有其他产品或服务名称均为其各自所有者的财产。

本文档中的信息取代本文档所有早期版本中提供的信息。