调试备忘录-SWD协议解析

目录--点击可快速直达

目录

写在前面

最近由于公司需要，所以就做了个基于SWD协议的离线烧写器。由于过程中参考了很多大神的文章，因此就想写个随笔记录下。整个烧写器由三个部分组成分别为：

- SWD协议读写芯片内部寄存器、RAM、Flash.
- STM32 + SPI + Flash + USB + FAFTS,通过USB虚拟出一个U盘，将要烧写的BIN文件放到这个U盘中，就可以烧写了.
- OLED + 按键，实现简单的操作页面.

本章先对SWD协议进行解析，整个文章主要分了三个部分，第一部分是对SWD协议进行简介；第二部分是对SWD协议进行物理层上的解析；

1 SWD协议简介

SWD的全称应该是**The Serial Wire Debug Port**(SW-DP),也就是串行调试端口，是**ARM**目前支持的两种调试端口之一，另一个调试端口叫做**JTAG Debug Port**，也就是我们常用的J-link上面的调试端口（JTAG模式下）。

基于ARM CoreSight调试构架，SWD可以通过传输数据包来读写芯片的寄存器。

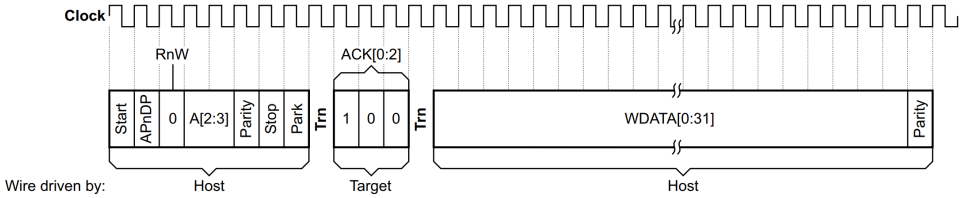
2 SWD物理层协议解析

SWD需要三根线与目标的MCU连接，分别为SWDIO、SWDCLK和GND.后面的内容中，HOST为主机，就是我们提供的SCK的一方；TARGET为目标MCU。

- **SWDIO**为双向Data线，主机读写目标芯片数据。
- **SWDCLK**为时钟线，类似于SPI需要由主机提供时钟。同时，数据都是在时钟下降沿读取，上升沿进行数据翻转。
- **GND**为双向Data口，主机读写目标芯片数据。

2.1 SWD通信时序分析

首先放个写操作成功的时序图。



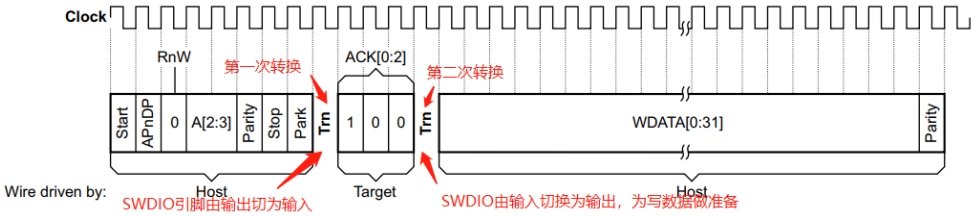
从图中可知，整个流程大致分为三部分：

- 第一部分：HOST -> TARGET 主机发送读写命令，这部分指定了读写操作，指定了要访问AP还是DP,还指定了ADDR。
- 第二部分：TARGET -> HOST 目标MCU回复Ack，通知主机是否可以继续操作。
- 第三部分：HOST -> TARGET 主机写数据

下面，我们针对图中的一些名词做出解释：

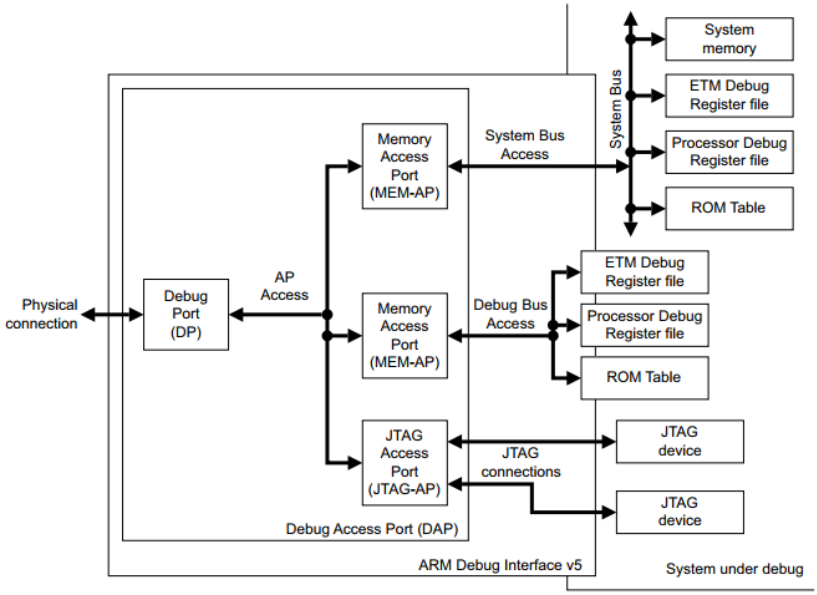
- Start: 一位起始位，值为1。
- APnDP: 一位，表示访问的是DP寄存器还是AP寄存器，0：DP,1: AP。
- RnW: 一位，表明是读操作还是写操作，0：写，1：读。
- ADDR[2:3]:两位，给出DP或者AP寄存器地址ADDR[3:2]地址区域，发送时低位在前（后面有具体说明）。
- Parity: 一位 为前面的数据包提供奇偶校验。在包头中的校验位校验了APnDP, RnW and A[2:3]，在数据中的校验位校验了32位的数据。
- Stop: 一位，停止位，值为0。
- Park: 一位，在传输该位时，主机不对SWDIO进行驱动，该线由硬件拉高（上拉），所以这位读作一。
- Trn: 调转周期，在该周期中，不对线进行前驱动，实际应用时，这个时候要将SWDIO引脚的输入输出状态翻转。这个周期的时间由TURNROUND控制（位于WCR寄存器中）。
- Ack[0:2]: 三位，目标MCU到主机的响应，低位在前。100：成功，010：等待，001：失败。
- WDATA[0:31]: 32位的写数据包，由主机发送给目标MCU。
- RDATA[0:31]: 32位的读数据包，由MCU发送给主机。

值的注意的是，Trn，调转周期，因为我们是单总线通信，一根线上既有写又有读，而这个Trn就是发生在写读切换的时候的一个延时。



2.2 SWD 寄存器简介

SWD通信的时候主要涉及的寄存器就两个，一个DP，一个AP。



如图所示，DP就是Debug port,AP 就是 Access port。如果要访问内核寄存器，就得先配置DP->AP->MEM-AP。

2.2.1 DP寄存器

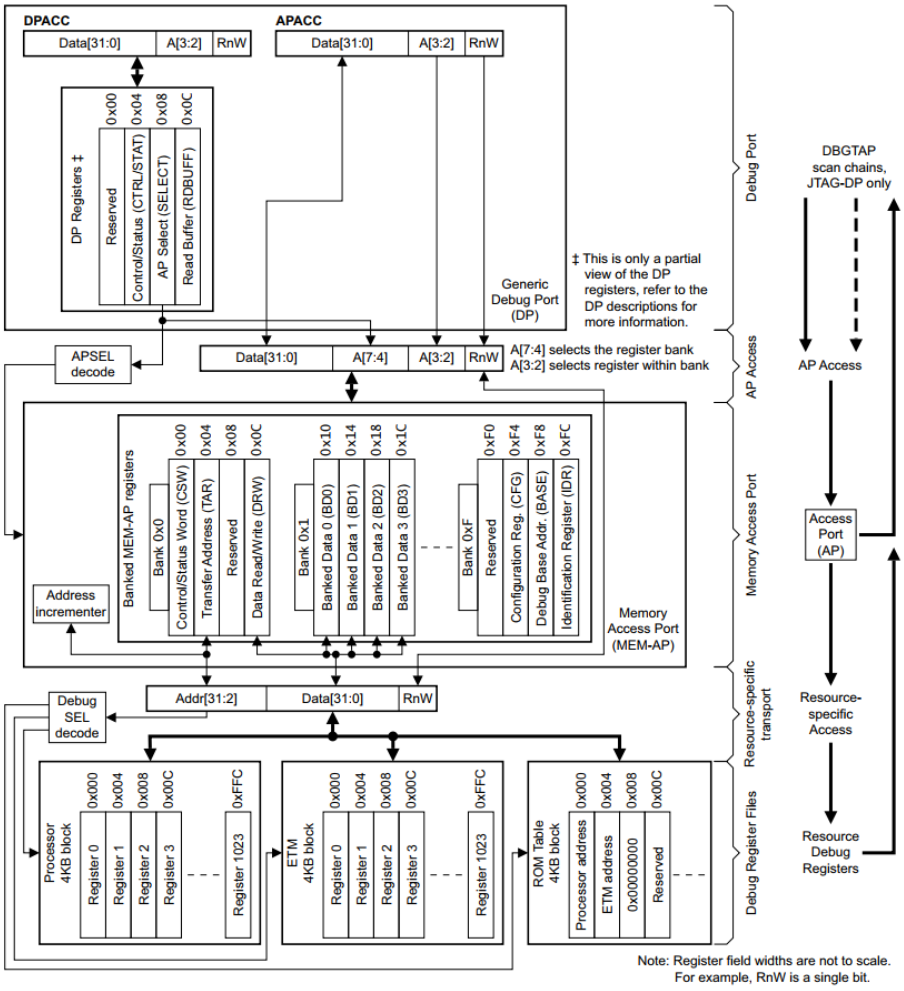
DP寄存器如下图，地址就是我们在包头中的ADDR[2:3]，这两位指示的地址。具体的寄存器实在是太多了，具体的请参考《ARM Debug Interface v5 Architecture Specification》。

地址	CTRLSEL	描述	访问	参考
b00	X	ID 代码寄存器	R	标识代码寄存器，IDCODE
		中止寄存器	W	中止寄存器，ABORT
b01	b0	DP 控制/状态寄存器	R/W	控制/状态寄存器，CTRL/STAT
	b1	线控制寄存器	R/W	线控制寄存器，WCR(只用于 SW-DP)
b10	X	读再发寄存器	R	读再发寄存器，RESEND(只用于 SW-DP)
		选择寄存器	W	AP 选择寄存器，SELECT
b11	X	读缓冲	R	读缓冲，RDBUFF
		-	-	-

值的说明的是，总线复位之后，然后进行JTAG和SWD的切换操作（发0X79，0XE7或者0X6D,0XB7），然后必须先读下IDDCODE,判断下MCU的类型，然后才能继续别的操作。

2.2.2 AP寄存器

AP寄存器如下路，AP寄存器相比较于DP则是复杂了很多。具体的寄存器实在是太多了，具体的请参考《ARM Debug Interface v5 Architecture Specification》。



2.3 SWD通信流程

2.3.1 SWD复位

如下图，ARM对SWD的复位有明确的说明，要求连续50个时钟周期的高电平，认为是SWD复位，同时复位之后必须读一次IDCODE（位于DP寄存器中）。

5.4.1 Connection and line reset sequence

The serial interface to the SW-DP must use a connection sequence, to ensure that hot-plugging the serial connection does not result in unintentional transfers. The connection sequence ensures that the SW-DP is synchronized correctly to the header that is used to signal a connection. It consists of a sequence of 50 clock cycles with data = 1, that is, with the serial data signal asserted HIGH by the debugger.

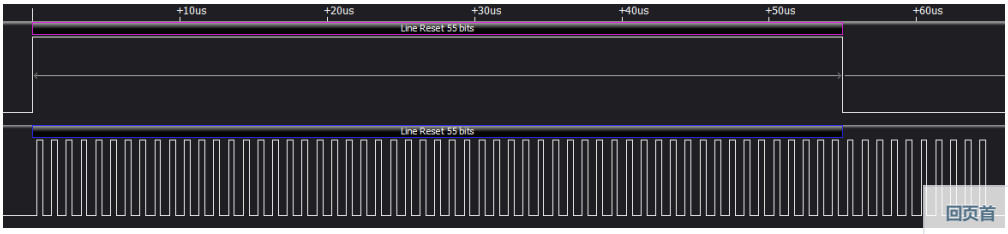
This connection sequence is also used as a line reset sequence, see *Protocol Error responses* on page 5-13. The protocol requires that any run of 50 consecutive 1s on the data input is detected as a line reset, regardless of the state of the protocol.

After the host has transmitted a line request sequence to the SW-DP, it must read the IDCODE register. The SW-DP returns an OK response to this read. For more information see:

- *The Identification Code Register, IDCODE* on page 6-8
- *Successful read operation (OK response)* on page 5-7.

The requirement that the host reads the IDCODE register to exit the training state gives confirmation that correct packet frame alignment has been achieved.

下图是我的复位时序，多写了5个周期的电平

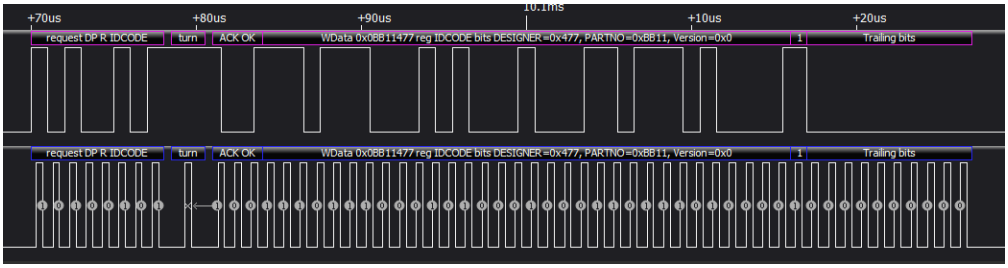


2.3.2 SWD读IDCODE

IDCODE位于SWD DP寄存器中，将地址设置为00b，读DP操作，就可以读到IDCODE了，根据内核型号不同，一般第一数字不一样，我的是0x0BB11477.

地址	CTRLSEL	描述	访问	参考
b00	X	ID 代码寄存器	R	标识代码寄存器，IDCODE
		中止寄存器	W	中止寄存器，ABORT
b01	b0	DP 控制/状态寄存器	R/W	控制/状态寄存器，CTRL/STAT
	b1	线控制寄存器	R/W	线控制寄存器，WCR(只用于 SW-DP)
b10	X	读再发寄存器	R	读再发寄存器，RESEND(只用于 SW-DP)
		选择寄存器	W	AP 选择寄存器，SELECT
b11	X	读缓冲	R	读缓冲，RDBUFF
		-	-	-

下图是我的读IDCODE时序



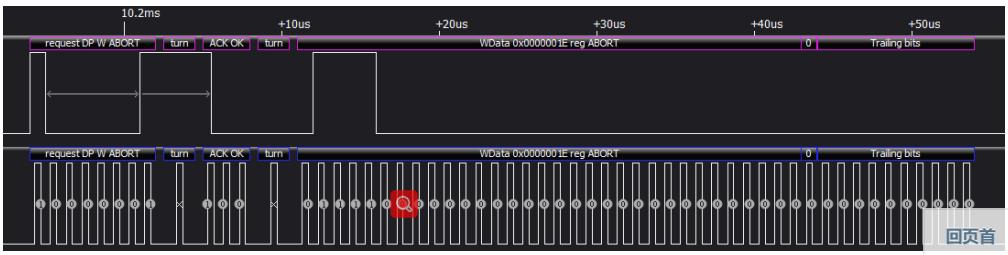
2.3.3 SWD清除错误标志位，并且使能AP调试

通过写DP中的APBORT[4:1]，来清除错误标志位。

Bits	Function	Description
[31:5]	-	Reserved, SBZ.
[4][a]	ORUNERRCLR[a]	Write 1 to this bit to clear the STICKYORUN overrun error flag[b] to 0.
[3][a]	WDERRCLR[a]	Write 1 to this bit to clear the WDATAERR write data error flag[b] to 0.

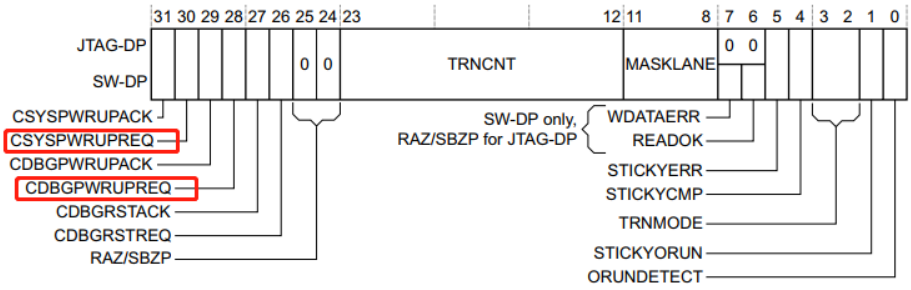
Table 6-4 AP Abort Register bit assignments (continued)

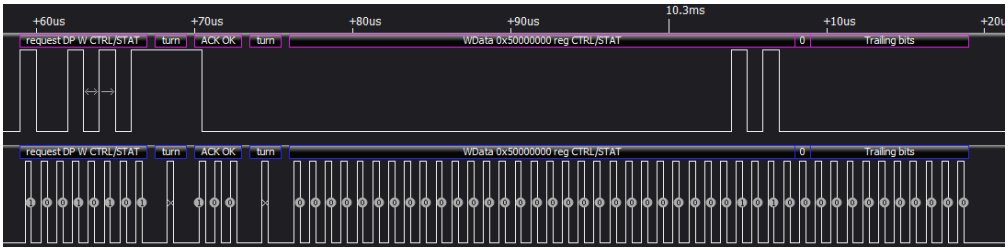
Bits	Function	Description
[2][a]	STKERRCLR[a]	Write 1 to this bit to clear the STICKYERR sticky error flag[b] to 0.
[1][a]	STKCMPCLR[a]	Write 1 to this bit to clear the STICKYCMP sticky compare flag[b] to 0.
[0]	DAPABORT	Write 1 to this bit to generate a DAP abort. This aborts the current AP transaction. Do this only if the debugger has received WAIT responses over an extended period.



回页首
1

通过写DP中的CTRL/STAT[30]和CTRL/STAT[28]使能AP调试

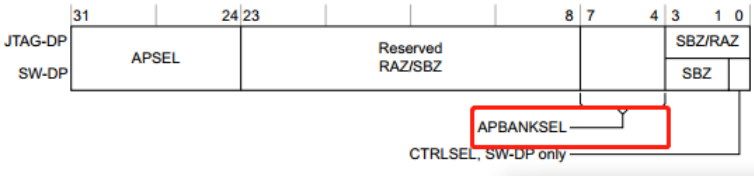




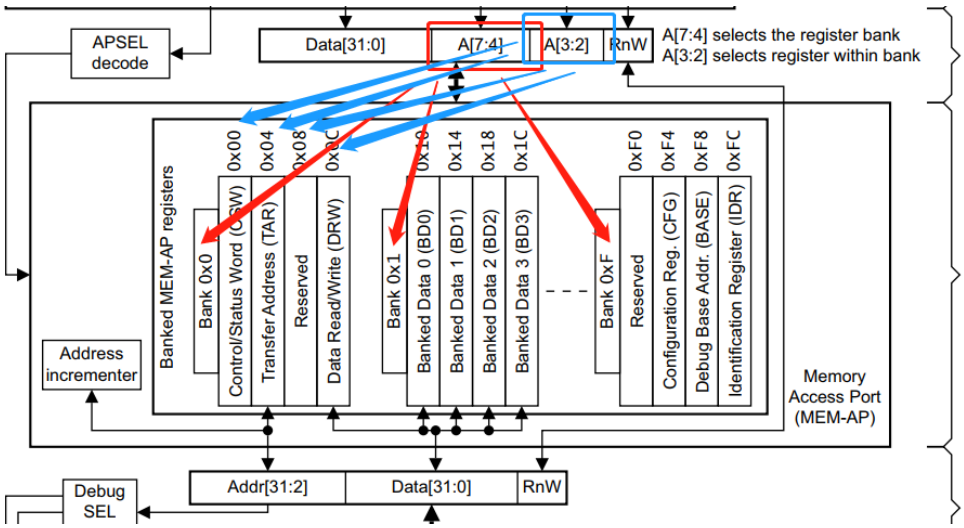
2.3.4 SWD读取AP IDR(也就是AP寄存器的ID CODE)

读取AP寄存器的步骤：

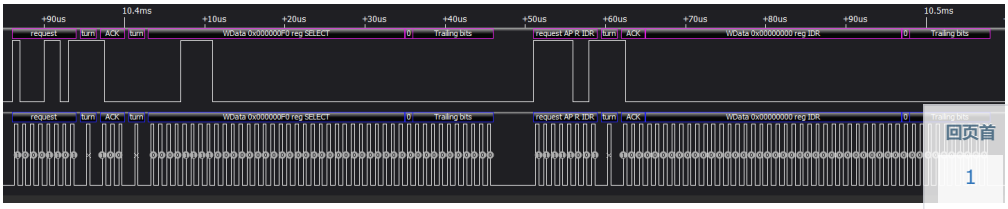
- 第一步通过写DP 中的SELECT寄存器，确定APBANK.
- 第二步通过AP命令包头的地址，确定APBANK内部的位置,发送读AP命令。如图DP SELECT寄存器。



如图，AP寄存器，APBANK决定的是BANK 0x0 -> BANK 0xF.ADDR[3:2]决定的是BANK内部的地址(ADDR[1:0]都是0)，0x00,0x04,0x08,0x0C,



所以，如果我要读取AP IDR的话，就先设置DP SELECT寄存器中的APBANKSEL为0xF,然后直接发读AP命令，ADDR[3:2]为11b.

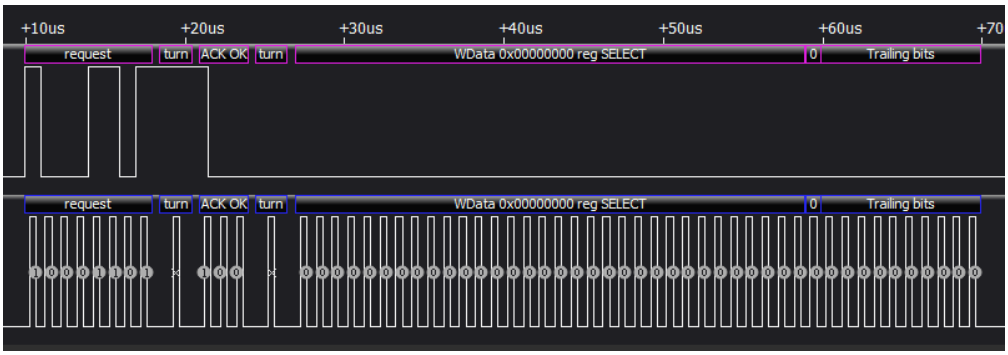


2.3.5 SWD读写MCU任意寄存器

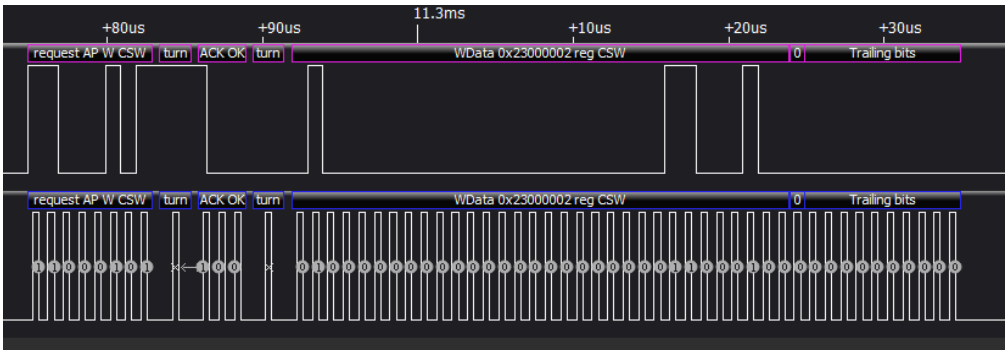
读取MCU任意寄存器的步骤：

- 第一步通过写DP 中的SELECT寄存器，确定APBANK为BANK0.
- 第二步通过AP命令包头的ADDR[3:2]地址，确定APBANK内部的地址为0x00(CSW),发送写AP命令。数据为0x23000002（该值可能有问题，此处请自行查看CSW寄存器描述，如果这个值写完之后依然读不出DRW，请联系我）.
- 第三步通过AP命令包头的ADDR[3:2]地址，确定APBANK内部的地址为0x04(TAR),发送写AP命令。数据为要读取的寄存器地址。
- 第四步通过AP命令包头的ADDR[3:2]地址，确定APBANK内部的地址为0x0C(DRW),发送读AP命令。一次读回来的数据无效。
- 第五步通过AP命令包头的ADDR[3:2]地址，确定APBANK内部的地址为0x0C(DRW),发送读AP命令。此时读回来的数据为正确数据。（这里读DP的RDBUFF也可以，详细请自行查阅寄存器文档）

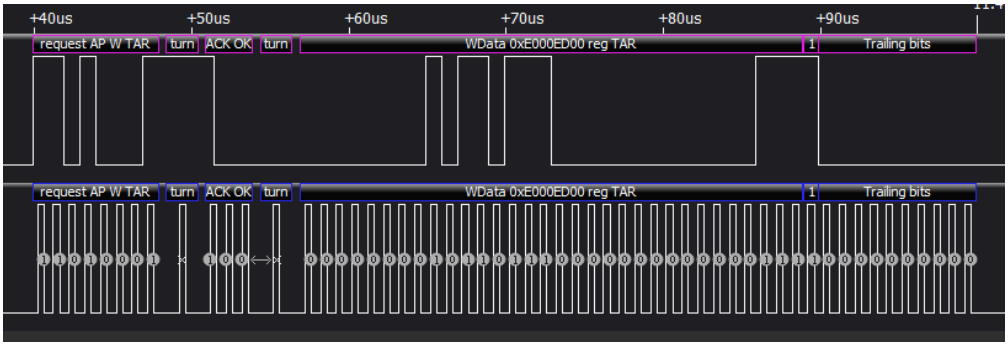
步骤一波形：



步骤二波形：

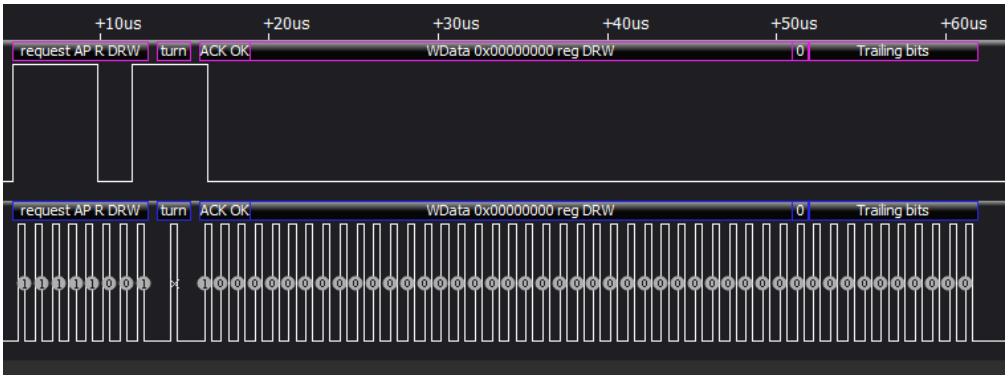


步骤三波形：

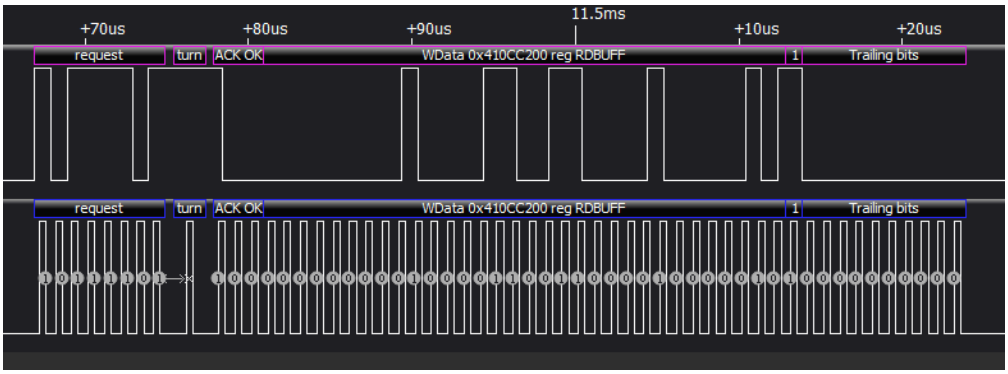


步骤四波形：

回页首
1



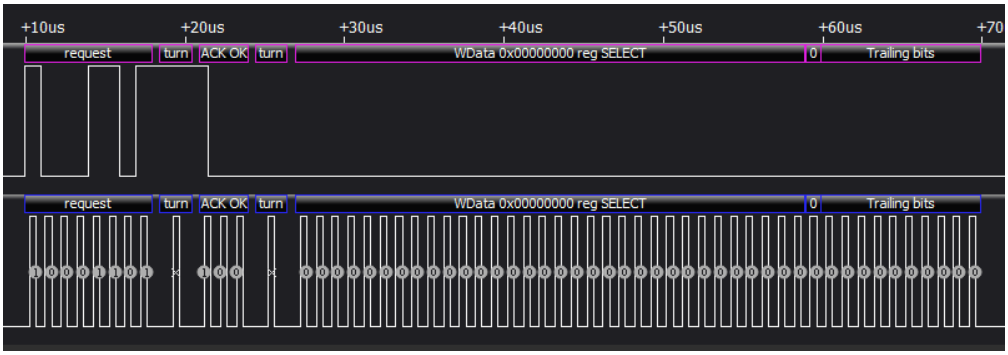
步骤五波形：



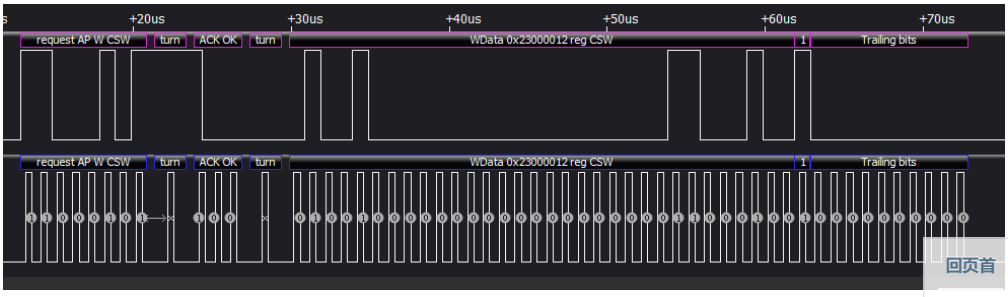
写入MCU任意寄存器的步骤：

- 第一步通过写DP 中的SELECT寄存器，确定APBANK为BANK0.
- 第二步通过AP命令包头的ADDR[3:2]地址，确定APBANK内部的地址为0x00(CSW),发送写AP命令。数据为0x23000012（该值可能有问题，此处请自行查看CSW寄存器描述，如果这个值写完之后依然不能写入DRW，请联系我）.
- 第三步通过AP命令包头的ADDR[3:2]地址，确定APBANK内部的地址为0x04(TAR),发送写AP命令。数据为要读取的寄存器地址。
- 第四步通过AP命令包头的ADDR[3:2]地址，确定APBANK内部的地址为0x0C(DRW),发送写AP命令。写入成功。

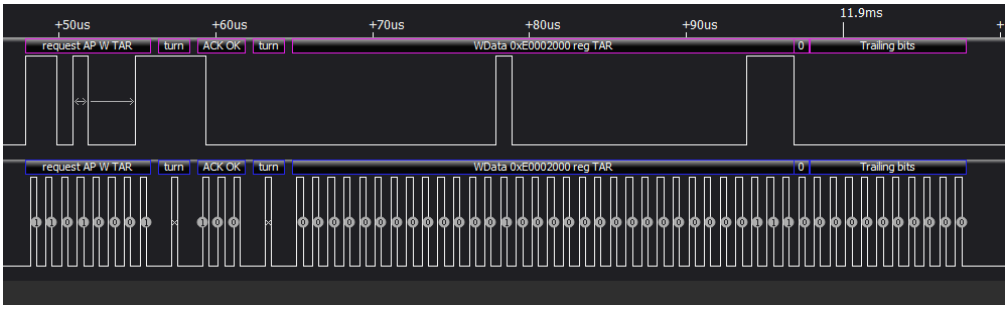
步骤一波形：



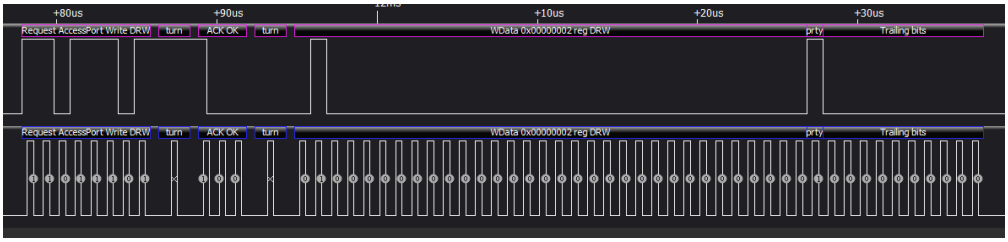
步骤二波形：



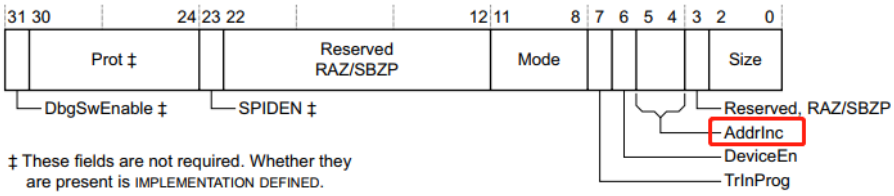
步骤三波形：



步骤四波形：



特殊说下，设置好CSW[5:4]之后，每次读写DRW之后，DRW的地址会自动+1，省去了需要每次重新设置地址的麻烦。



以上，本章的内容就结束了，通过本章的内容，可以实现SWD任意读取MCU寄存器，如果问题，请联系我~~~

参考文献：ARM Debug Interface

分类: 嵌入式常用通信协议

标签: 调试备忘录

洛神殇
关注 - 1
粉丝 - 4
+加关注

« 上一篇： Linux学习笔记 — 第五章 软件包管理

posted @ 2021-05-30 20:17 洛神殇 阅读(1145) 评论(5) 编辑 收藏 举报

· 探索 dotnet core 为何在 Windows7 系统需要补丁的原因
· 技术管理进阶——精要主义设计人生，对混乱的工作说不
· Dapper in .Net Core

最新新闻:
· 麻省理工学院物理学家观察超冷原子形成“量子龙卷”晶体
· Google本周末以人物涂鸦纪念霍金80周年冥诞
· 视频号还有机会吗?
· 元宇宙虚火加温，虚拟人物走进现实
· 资源砸进去，视频号直播带货今年能“起量”吗?
» 更多新闻...