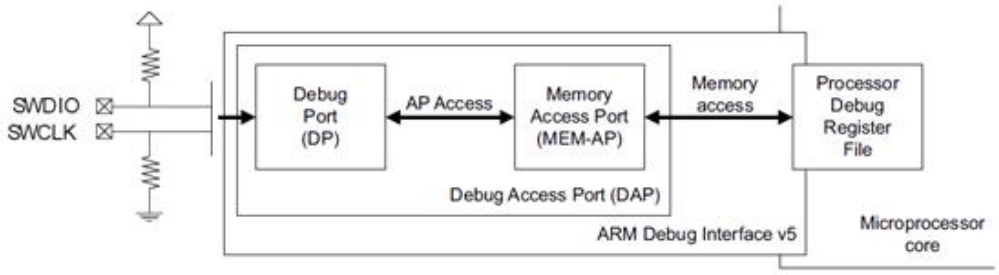
一.　　SW 协议框图。



SW 烧录协议围绕上图的 Debug　Access Port（DAP）来实现。

时钟信号由 SWCLK 管脚输入，数据信号从 SWDIO 管脚输入输出，操作 Debug Access Port（DAP）的寄存器，进而能够访问 MCU 内部的各种资源。

Debug Access Port（DAP）由两部分组成：

1.Debug Port(DP),

2.Memory Access Port（MEM-AP）（简称 AP）

二.　　SW 通讯要操作的寄存器。

烧录器要操作的 Debug Port（DP）寄存器如下：

6.2 Debug Port (DP) register descriptions
- 6.2.1 The AP Abort Register, ABORT
- 6.2.2 The Identification Code Register, IDCODE
- 6.2.3 The Control/Status Register, CTRL/STAT
- 6.2.4 The AP Select Register, SELECT
- 6.2.5 The Read Buffer, RDBUFF
- 6.2.6 The Wire Control Register, WCR (SW-DP only)
- 6.2.7 The Read Resend Register, RESEND (SW-DP only)

即：Debug Port（DP）总共包含 7 个寄存器，分别是 ABORT 寄存器， IDCODE 寄存器，CTRL/STAT 寄存器， SELECT 寄存器， RDBUFF 寄存器， WCR 寄存器， RESEND 寄存器。共 7 个。

烧录器要操作的 Memory Access Port（MEM-AP）寄存器如下：

即：Memory Access Port（MEM-AP）总共包含 9 个寄存器，分别是 CSW 寄存器，TAR 寄存器，DRW 寄存器， BD0 寄存器，BD1 寄存器，BD2 寄存器， BD3 寄存器，CFG 寄存器， BASE 寄存器，共 9 个。

三. SW 操作 DAP 寄存器的时序。

3.1 一次成功的写操作流程（指烧录器往 DP 或 MEM-AP 寄存器的写操作）

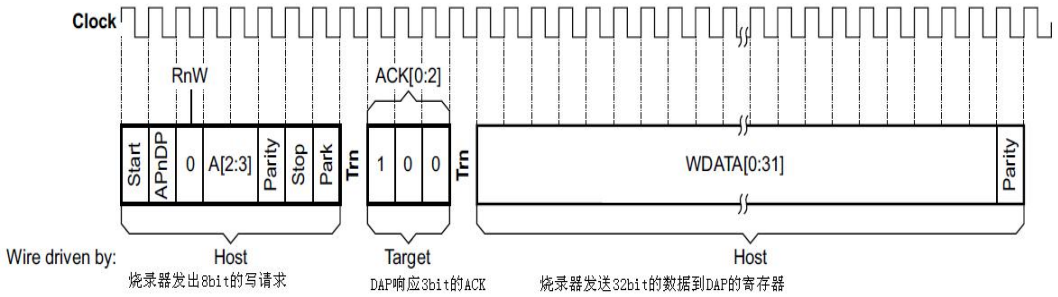


Figure 5-1 Serial Wire Debug successful write operation

3.2 一次成功的读操作流程（指烧录器读取 DP 或 MEM-AP 寄存器的数据）

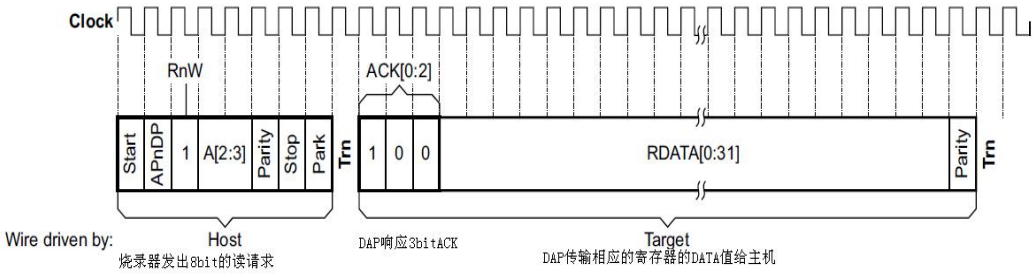


Figure 5-2 Serial Wire Debug successful read operation

3.3 DAP 没准备好， 响应 WAIT 给 HOST，此次通讯结束。

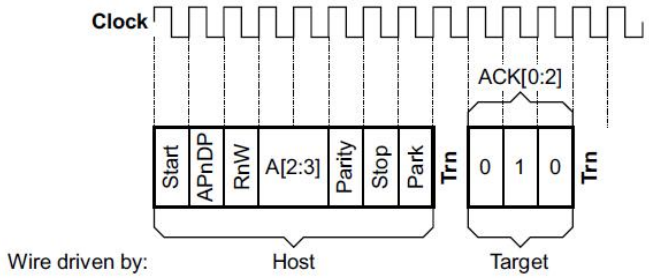


Figure 5-3 Serial Wire Debug WAIT response to a packet request

3.4 DAP 出错，响应 FAULT 给 HOST，此次通讯结束。

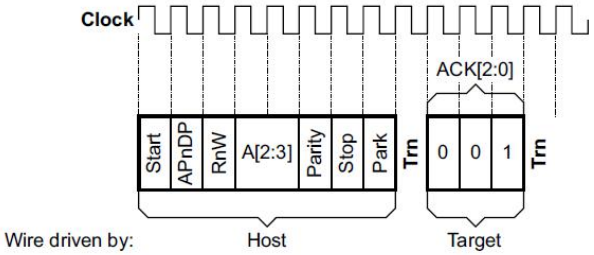


Figure 5-4 Serial Wire Debug FAULT response to a packet request

其中，主机 HOST 发出的请求由 8bit 组成，其含义如下，

Bit0： Start，其值固定为 1.

Bit1： APnDP，值为 0 时，代表请求访问 DP 寄存器。
值为 1 时，代表请求访问 MEM-AP 寄存器。

Bit2： RnW， 值为 0 时，代表写请求。
值为 1 时，代表读请求。

Bit[4:3]: A[3:2], 地址值，由于 DP 共有 7 个寄存器，AP 共有 9 个寄存器由{A[3：2],RnW, APnDP} 共同决定读/写访问 DP/MEM-AP 寄存器中的哪一个寄存器（详见下面表格）。

Bit[5]： Parity, 奇偶校验值，用于指示由{APnDP，RnW, A[2:3]}组成这四位数的奇偶总个数。

Bit[6]: Stop, 其值为固定为 0.

Bit[7]: Park, Not driven by the host, Must be read as”1” by the target Because of the pull-up

SW 协议的 ACK 由 3bit 组成，总共有三类 ACK，其值如下：

ACK[0:2]==001,代表 FAULT response,

ACK[0:2]==010,代表 WAIT response,

ACK[0:2]==100,代表 OK response,

数据由 32bit 有效数据 + 1bit 数据的奇偶校验位给成。

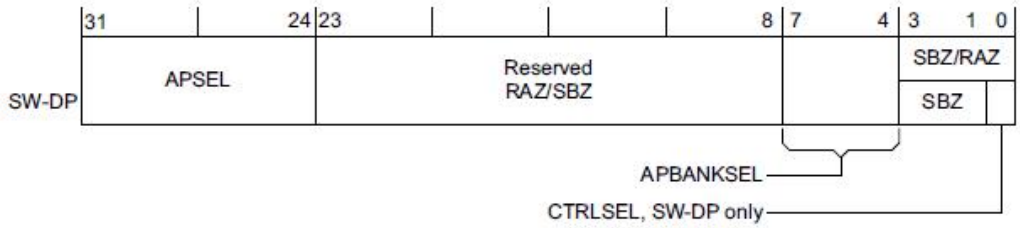
Trn， Turnaround, This is a period during which the line is not driven
and the state of line is undefined.
Trn 的周期个数可在 WCR（wire control register）中配置
（WCR 属于 DP 寄存器）

当 APnDP= 0 时，
代表请求访问 DP 寄存器。由 A[3:2] 和 RnW 决定要访问哪一个 DP 寄存器。

Table 138. SW-DP registers

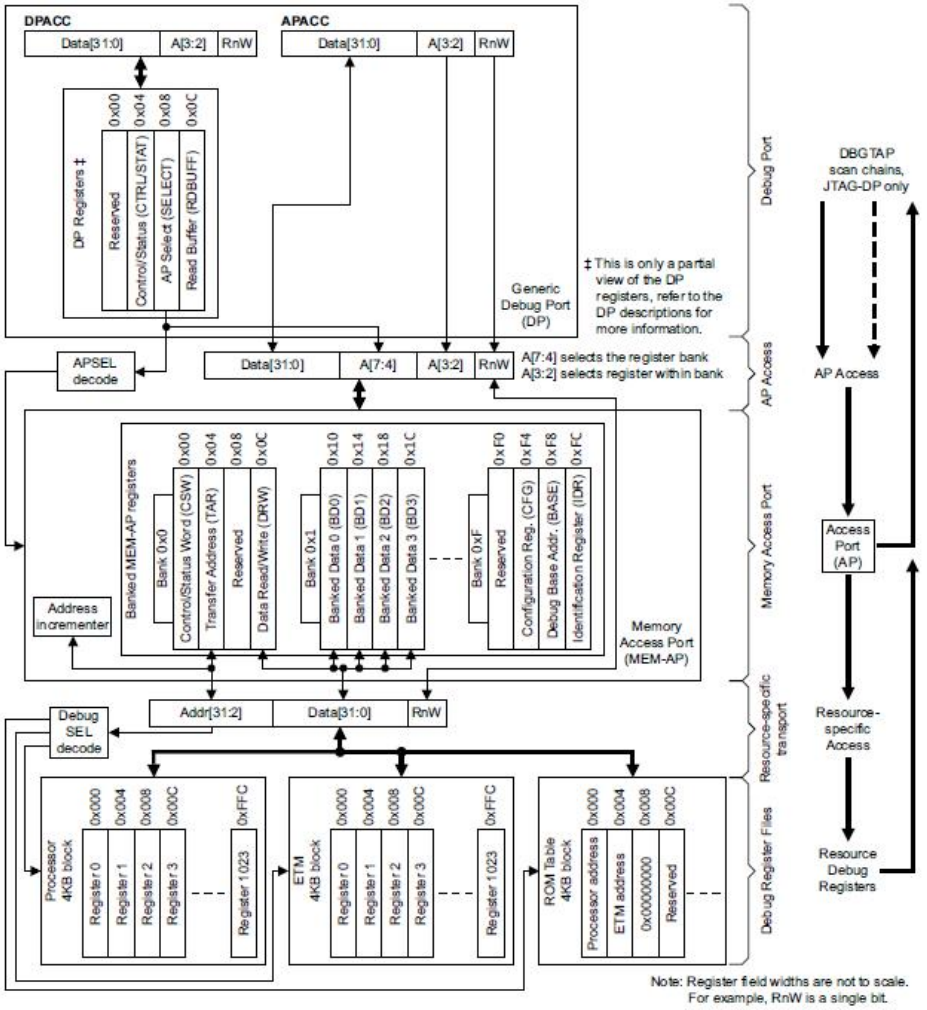
A[3:2]	R/W	CTRLSEL bit of SELECT register	Register	Notes
00	Read		IDCODE	The manufacturer code is set to the default ARM code for Cortex-M0: **0x0BB11477** (identifies the SW-DP)
00	Write		ABORT	
01	Read/Write	0	DP-CTRL/STAT	Purpose is to: – request a system or debug power-up – configure the transfer operation for AP accesses – control the pushed compare and pushed verify operations. – read some status flags (overrun, power-up acknowledges)
01	Read/Write	1	WIRE CONTROL	Purpose is to configure the physical serial port protocol (like the duration of the turnaround time)
10	Read		READ RESEND	Enables recovery of the read data from a corrupted debugger transfer, without repeating the original AP transfer.
10	Write		SELECT	The purpose is to select the current access port and the active 4-words register window
11	Read/Write		READ BUFFER	This read buffer is useful because AP accesses are posted (the result of a read AP request is available on the next AP transaction). This read buffer captures data from the AP, presented as the result of a previous read, without initiating a new transaction

当 APnDP=1 时，
代表请求访问 AP 寄存器。决定要访问哪一个 AP 寄存器由以下两个值决定，
A[3:2] + 当前的 SELECT 值（SELECT 是一个 DP 寄存器）
即，用户要访问 AP 寄存器，必须要先往 DP 的 SELECT 寄存器写相关的值。
原因如下：
SELECT 寄存器的定义如下：(SELECT 属于 DP，DP 决定访问哪个 AP 寄存器)



其中 SELECT[31：24]代表 APSEL，即当前选择的 AP，

由于只有一个 MEM-AP 所以 APSEL 一直保持为默认值 0 。烧录器没用到 APSEL
烧录器经常要写的地方是 SELECT[7:4],APBANKSEL 代表当前选择的 BANK。
AP 寄存器总共有 9 个寄存器，但被存放在不同的 BANK 里，总共有 16 个 BANK，由 SELECT 寄存器中的 APBANKSEL[7:4]决定哪个 BANK.



由上图可知，Debug Port（DP）处于图中的最上层。

Memory Access Port(MEM-AP)处于图中的中间层。

由 DP 中的 SELECT 寄存器，产生 APBANKSEL 信号，即 A[7:4]，决定。

例如，假设要访问 TAR 寄存器，由上图可见，TAR 处于 BANK0 的第二个寄存器。所以当要访问 TAR 时，要先往 DP 的 SELECT 寄存器写 0x0000_0000, 即 SELECT[7:4]＝0 选择 BANK0,先择好 BANK 后，再由 A[3:2]=2’b01,决定选择 BANK0 里的第二个寄存器，即 TAR 寄存器。

又例如，假设要访问 BASE 寄存器，由上图可见，CFG 处于 BANK 0xf 的第三个寄存器，所以当要访问 BASE 寄存器时，要先往 DP 的 SELECT 寄存器写 0x0000_00f0,即 SELECT[7:4]=0xf, 选择 BANK 0xf ， 先择好 BANK 0xf 后，再由 A[3:2]=2’b10,决定选择 BANK 0xf 里的第三个寄存器，即 BASE 寄存器。

四．**DAP 如何访问 MCU 内部资源**。

SW 烧录器的本质，实际上是在不断地操作 DAP 寄存器。

4.1 下面讨论，如何往 MCU 内部资源 SRAM（0x20000000）写一个数（0x55AA6699）。
为了实现这个简单功能，需要用到两个 AP 寄存器
TAR（The Transfer Address Register）
DRW(The Data Read/Write Register)
从字面上很容易理解，只要我们往地址寄存器 TAR 写入数值，0x20000000, 往数据寄存器 DRW 中写入数值 0x55AA6699，即可实现通过 DAP 访问 MCU 内部资源，实现往 SRAM（0x20000000）写一个数（0x55AA6699）的功能。

4.2 下面讨论，如何读取 MCU 内部资源 SRAM（0x2000000c）地址的数值。为了实现这个简单功能，需要用到两个 AP 寄存器
TAR（The Transfer Address Register）
DRW(The Data Read/Write Register)
从字面上很容易理解，只要我们往地址寄存器 TAR 写入数值，0x2000000c, 然后发起一条读请求，读取 DRW 的值， 即可实现通过 DAP 访问 MCU 内部资源，实现读取 SRAM（0x2000000c）地址的数值的功能。
这里需要注意一点的是：按照上面的方式，读回来的 DRW 的值，并不是正确的值。这是由于读取 AP 寄存器的特殊性所带来的。当读取 AP 寄存器时，返回的数据是上一次传输的值。
言下之意，有两种方式可以得到正确的 AP 寄存器的值，
第 1 种方式，发起两次读 AP 操作，真正有效的数据，在第二次读 AP 操作时给出。对于本例子，当 TAR 写入 0x2000000c 后，发起第一次读取 DRW 的值，读到的 DRW 是上一次 AP 访问内部资源的值，并不是我想要的值，再发起第二次读取 DRW 的值，此时读到的数值才是 0x2000000c 的值。
第 2 种方式，先发起第一次读 AP 操作，再发起读 RDBUFF 操作（注：RDBUFF 是一个 DP 寄存器）。对于本例子，当 TAR 写入 0x2000000c 后，发起第一次读取 DRW 的值，读到的 DRW 是上一次 AP 访问内部资源的值，再发起第二次读 RDBUFF 的值，读到的 RDBUFF 的值为 0x2000000c 的值。
JLINK 烧录器一般采用 RDBUFF 这种方式来读取 AP 寄存器。

总的来说，读写 DP 寄存器没有迟滞，不需要读写两次。
读 AP 寄存器，有迟滞，第一次读 AP，不是你想要的值，第二次读 AP 才是你想要的值。
写 AP 寄存器，没有迟滞。如 TAR，DRW 都是 AP 寄存器，只需要写一次即可写进去。

4.3 访问 MCU 内部资源的另外一种方式。
理论上，通过两个 AP 寄存器，
地址寄存器 TAR（The Transfer Adress Regisger），数据寄存器 DRW（The Data Read/Write Register）已经可以实现访问 MCU 内部资源，下面介绍 Jlink 烧录器，CMSIS-DAP 烧录器中用到的另外一种访问 MCU 内部资源的方式。
用到 5 个的 AP 寄存器：

TAR（The Transfer Adress Regisger）

BD0 (Banked Data Register 0）

BD1 (Banked Data Register 1)

BD2 (Banked Data Register 2)

BD3 (Banked Data Register 3)

简单来说，一种方式是 TAR+DRW， 一种方式是 TAR+BD(0~3). 这两种方式的区别如下：

对于 TAR+DRW：

对 DRW 寄存器读操作，实际上是指读取 MCU 内部 TAR[31:0]所指向的地址的数值。

对 DRW 寄存器写操作，实际上是指往 MCU 内部 TAR[31:0]所指向的地址写入数据。

对于 TAR+BD(0~3)

对 BD0 寄存器的读操作，实际上是指读取 MCU 内部{TAR[31:4],4'h0}所指向地址的数值

对 BD1 寄存器的读操作，实际上是指读取 MCU 内部{TAR[31:4],4'h4}所指向地址的数值

对 BD2 寄存器的读操作，实际上是指读取 MCU 内部{TAR[31:4],4'h8}所指向地址的数值

对 BD3 寄存器的读操作，实际上是指读取 MCU 内部{TAR[31:4],4'hc}所指向地址的数值

对 BD0 寄存器的写操作，实际上是指往 MCU 内部{TAR[31:4],4'b0}所指向地址写入数据

对 BD1 寄存器的写操作，实际上是指往 MCU 内部{TAR[31:4],4'b4}所指向地址写入数据

对 BD2 寄存器的写操作，实际上是指往 MCU 内部{TAR[31:4],4'b8}所指向地址写入数据

对 BD3 寄存器的写操作，实际上是指往 MCU 内部{TAR[31:4],4'bc}所指向地址写入数据

有以下几点补充说明，

4.3.1 要访问的 MCU 内部资源的地址，都是由 TAR 来决定。不管是 TAR+DRW 还是 TAR+BD(0~3).

4.3.2 TAR，DRW，BD0，BD1,BD2,BD3 都是 AP 寄存器，读 AP 寄存器要注意迟滞，即读两次。

4.3.3 SW 提供地址 TAR 自加 1 功能，即读/写一次 DRW/BD(0~3)后，TAR 自动加 1 ,指向下一个 TAR=TAR+ 1 的地址。

方法是配置 CSW（Control/Status Word Regisger）寄存器中的 Bit[5:4]==AddrInc.

AddrInc==2'b00, TAR 自动加 1 功能关闭（默认）

AddrInc==2'b01, TAR 自动加 1 功能打开

AddrInc==2'b10/2'b11, 保留，无定义。

CSW（Control/Status Word Regisger）寄存器定义如下。（另 CSW[2:0] Size 默认值为 010（代表 WORD（32bit））其它值如下 000(代表 BYTE（ 8 bit）) 001（代表 HalfWord(16bit)））

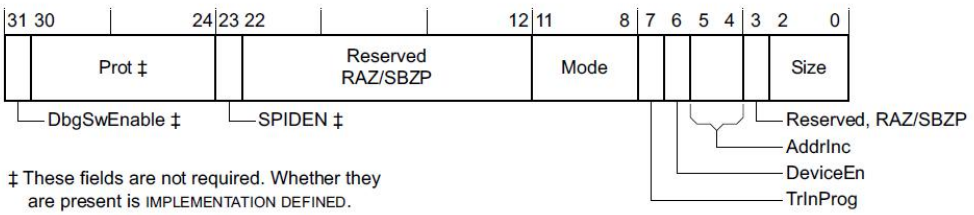


Figure 11-1 MEM-AP Control/Status Word Register, CSW, bit assignments

4.3.4 很明显，访问 DRW 跟访问 BD0，BD1，BD2,BD3 实现的功能都是类似的。它们都是 AP 寄存器，但要注意 DRW（The Data Read/Write Register）位于 AP 寄存器 16 个 BANK 中的 BANK0. 而 BD(0~3) （Banked Data Register 0~ 3 ）寄存器位于 AP 寄存器 16 个 BANK 中的 BANK1。

所以，当访问 DRW 寄存器（AP）时，需要先访问 SELECT 寄存器（DP），将 APBANKSEL 写 0 .

当访问 BD0~3 寄存器（AP）时，需要先访问 SELECT 寄存器（DP），将 APBANKSEL 写 1 .

4.3.5 分析 JLINK 烧录器，CMSIS-DAP 烧录器的流程发现。它们是既有用 TAR+DRW 方式访问 MCU 内部资源。 也有用 TAR+BD(0~3)方式访问 MCU 内部资源。

什么情况下用到了 TAR+BD(0~3）的方式更有优势?答案是 TAR 自增功能关闭，需要反复操作四个相邻的寄存器,反反复复，不断地读写，都是在操作这 4 个寄存器的情况，使用 TAR+BD(0~3)更有优势。

因为使用 TAR+BD(0~3)这种方式时。我只需要配置一个地址 TAR[31:0]. 操作 BD0 相当于操作{TAR[31:4],4'h0}， 操作 BD1 时相当于操作{TAR[31:4],4'h4}, 操作 BD2 相当于操作{TAR[31:4],4'h8}, 操作 BD3 时相当于操作{TAR[31:4],4'hc}

烧录器有这样一个工作需要,它确实需要反复地访问四个寄存器。具体的那 4 个寄存器如下：

Debug Halting Control and Status Register (DHCSR) address = 0xE000EDF0

Debug Core Register Selector Register (DCRSR) address = 0xE000EDF4

Debug Core Register Data Register (DCRDR) address = 0xE000EDF8

Debug Exception and Monitor Control Register (DEMCR) address = 0xE000EDFC

这四个寄存器, 主要是用来 halt 住 CPU，改写 CPU 的堆栈，PC，继续 RUN，又 halt 住 CPU，改写 PC，堆栈。。。。。反复地访问这四个寄存器。

Jlink 烧录器，CMSIS-DAP 烧录器在访问上述四存器时，会发现它们会先将 TAR 赋值为 0xE000EDF0. 即当 TAR=0xE00EDF0 时，访问 BD0 相当于访问{0xE000EDF,4'h0}=0xE000EDF0

访问 BD1 相当于访问{0xE000EDF,4'h4}=0xE000EDF4

访问 BD2 相当于访问{0xE000EDF,4'h8}=0xE000EDF8

访问 BD3 相当于访问{0xE000EDF,4'hc}=0xE000EDFC

总的来说，

Jlink 烧录器，CMSIS-DAP 烧录器在访问上述四存器时，会发现它们会先将 TAR 赋值为 0xE000EDF0. 所以， 烧录器访问 BD0 相当于访问 DHCSR

烧录器访问 BD1 相当于访问 DCRSR

烧录器访问 BD2 相当于访问 DCRDR

烧录器访问 BD3 相当于访问 DEMCR

当反反复复地来回访问这四个寄存器时，只需要设置 1 次 TAR，假如这种情况使用 TAR+DRW 这种方式，就需要不停地来回切换 TAR 的值。这就是 JLINK 在访问上述 4 个寄存器时使用 TAR+DB(0~3)方式的原因。

4.3.6 core debug registers (DHCSR, DCRSR,DCRDR,DEMCR)

Debug Halting Control and Status Register (0xE000EDF0)

Bits	Field	Type	Reset Value	Descriptions
31:16	DBGKEY (during write)	WO	—	Debug Key. During write, the value of 0xA05F must be used on the top 16-bit. Otherwise the write is ignored.
25	S_RESET_ST (during read)	RO	—	Reset status flag (sticky). Core has been reset or being reset; this bit is clear on read.
24	S_RETIRE_ST (during read)	RO	—	Instruction is completed since last read; this bit is clear on reset.

Bits	Field	Type	Reset Value	Descriptions
19	S_LOCKUP	RO	—	When this bit is 1, the core is in lockup state.
18	S_SLEEP	RO	—	When this bit is 1, the core is sleeping.
17	S_HALT (during read)	RO	—	When this bit is 1, the core is halted.
16	S_REGRDY_ST	RO	—	When this bit is 1, the core completed a register read or register write operation.
15:4	Reserved	—	—	Reserved.
3	C_MASKINTS	R/W	0	Mask exceptions while stepping (does not affect NMI and hard fault); valid only if C_DEBUGEN is set.
2	C_STEP	R/W	0	Single step control. Set this to 1 to carry out single step operation; valid only if C_DEBUGEN is set.
1	C_HALT	R/W	0	Halt control. This bit is only valid when C_DEBUGEN is set.
0	C_DEBUGEN	R/W	0	Debug enable. Set this bit to 1 to enable debug.

Debug Core Register Selector Register (0xE000EDF4)

Bits	Field	Type	Reset Value	Descriptions
31:17	Reserved	—	—	Reserved
16	REGWnR	WO	—	Set to 1 to write value to register Set to 0 to read value from register
15:5	Reserved	—	—	Reserved
4:0	REGSEL	WO	0	Register select

Debug Core Register Data Register (0xE000EDF8)

Bits	Field	Type	Reset Value	Descriptions
31:0	DBGTMP	RW	0	Data value for the core register transfer

Debug Exception and Monitor Control Register (0xE000EDFC)

Bits	Field	Type	Reset Value	Descriptions
31:25	Reserved	—	—	Reserved
24	DWTENA	RW	0	Data watchpoint unit enable
23:11	Reserved	—	—	Reserved

(*Continued*)

Bits	Field	Type	Reset Value	Descriptions
10	VC_HARDERR	RW	0	Debug trap at hard fault exception
9:1	Reserved	—	—	Reserved
0	VC_CORERESET	RW	0	Halt processor after system reset and before the first instruction executed

补充说明：

4.3.6.1 DCRSR(0xE000EDF4)中的 Bit[4:0] REGSEL 决定操作的 ARM 寄存器。ARM 寄存器定义如下:

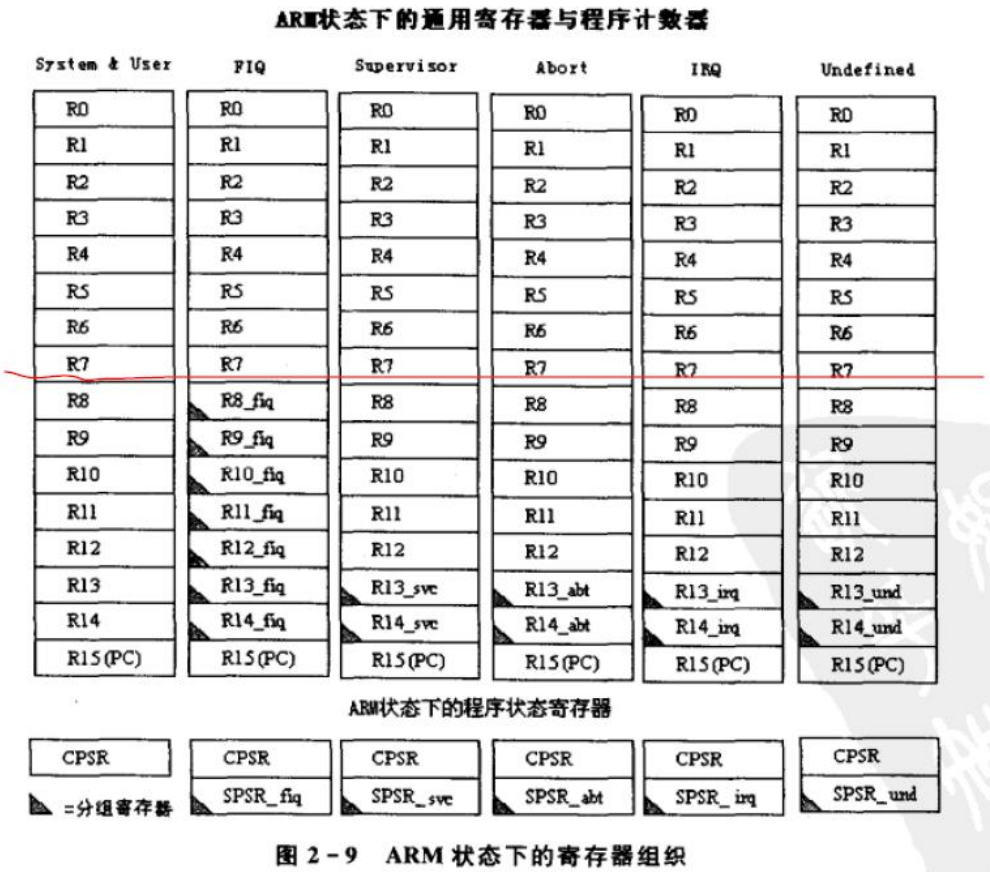


图 2-9 ARM 状态下的寄存器组织

举例：如果要访问 R15(PC),则将 DCRSR(0xE000EDF4)中的 Bit[4:0] REGSEL 配置为 0x0f,
如果要访问 CPSR，则将 DCRSR(0xE000EDF4)中的 Bit[4:0] REGSEL 配置为 0x10,
如果要访问 R3，则将 DCRSR(0xE000EDF4)中的 Bit[4:0] REGSEL 配置为 0x03

4.3.6.2 讨论更改 PC 值的步骤，使 PC 指向 SRAM 地址（0x2000_0000），目的是让程序从 0x2000_0000 开始执行。

步骤 1, 访问 AP 寄存器 TAR，TAR 赋值为 0xE000EDF0。

步骤 2， 访问 DP 寄存器 SELECT，SELECT 赋值为 0x0000_0010,选择为 BANK1

步骤 3, 访问 AP 寄存器 BD1（因为 TAR＝0xE000EDF0,所以 BD1 指向 0xE000EDF4 的寄存器，所以此时访问 BD1，相当于访问 DCRSR），往 BD1 中写 0x0001_000f.即 DCRSR 中写入 0x0001_000f.

相当于 REGWnR＝1,选择写操作

REGSEL＝0xf,选择 R15 寄存器，R15 寄存器即 PC

步骤 4, 访问 AP 寄存器 BD2（因为 TAR＝0xE000EDF0,所以 BD2 指向 0xE000EDF8 的寄存器，所以此时访问 BD2，相当于访问 DCRDR），往 BD2 中写 0x2000_0000 即 DCRDR 中写入 0x2000_0000.

由于步骤 3 中选定了 R15，并且选择写操作。所以往 DATA 寄存器 DCRDR 中写入 0x2000_0000，相当于往 R15（PC）中写入 0x2000_0000

至此，实现了往 PC 中写入 0x2000_0000 的简单功能。从以上步骤看出，实现更改 PC 的值，都是通过访问 AP 寄存器 TAR，BD1,BD2， DP 寄存器 TAR 来实现的。

4.3.6.3 讨论暂停 Halt 并复位 the CPU 的步骤

步骤 1, 访问 AP 寄存器 TAR，TAR 赋值为 0xE000EDF0。

步骤 2， 访问 DP 寄存器 SELECT，SELECT 赋值为 0x0000_0010,选择为 BANK1

步骤 3. 访问 AP 寄存器 BD0（因为 TAR＝0xE000EDF0,所以 BD0 指向 0xE000EDF0 的寄存器，所以此时访问 BD0，相当于访问 DHCSR），往 BD0 中写 0xA05F0003.即 DHCSR 中写入 0xA05F0003 往 DHCSR 寄存器写入 0xA05F0003，这会 halt the core,具体请查看 DHCSR 的定义

步骤 4. 访问 AP 寄存器 BD4（因为 TAR＝0xE000EDF0,所以 BD4 指向 0xE000EDFC 的寄存器，所以此时访问 BD4，相当于访问 DEMCR），往 BD4 中写 0x01000001. 即 DEMCR 中写入 0x01000001; 往 DEMCR 寄存器的 bit VC_CORERESET 写 1 ,使能 halt-on-reset.具体请查看 DEMCR 的定义

步骤 5. 访问 AP 寄存器 TAR，TAR 赋值为 0xE000ED0C

步骤 6. 访问 DP 寄存器 SELECT，SELECT 赋值为 0x0000_0000,选择为 BANK0 因为接下来要访问 DRW 寄存器，而 DRW 寄存器位于 BANK0.

步骤 7. 访问 AP 寄存器 DRW，往 DRW 中写入 0xFA050004,相于于往 AIRCR 写 0xFA050004，这会 reset the core，具体请查看 AIRCR 定义。

至此，CPU 会 halt 在第一条指令处，且所有的外设及寄存器都被复位。（除了 DEBUG 寄存器）。也看出了一点。不管烧录器要做什么，实现什么功能，都靠读写 AP 寄存器，读写 DP 寄存器来实现的。

Address	Name	CMSIS Symbol	Full Name
0xE000ED00	CPUID	SCB->CPUID	CPU ID (Identity) Base register
0xE000ED04	ICSR	SCB->ICSR	Interrupt Control State Register
0xE000ED0C	AIRCR	SCB->AIRCR	Application Interrupt and Reset Control Register

Application Interrupt and Control State Register (SCB -> AIRCR)

Bits	Field	Type	Reset value	Descriptions
31:16	VECTKEY (during write operation)	WO	—	Register access key. When writing to this register, the VECTKEY field need to be set to 0x05FA, otherwise the write operation would be ignored.
31:16	VECTKEYSTAT (during read operation)	RO	0xFA05	Read as 0xFA05.
15	ENDIANESS	RO	0 or 1	1 indicates the system is big endian. 0 indicates the system is little endian.
14:3	Reserved	—	—	Reserved.
2	SYSRESETREQ	WO	—	Write 1 to this bit cause the external signal SYSRESETREQ to be asserted.
1	VECTCLRACTIVE	WO	—	Write 1 to this bit causes: Exception active status to be cleared Processor return to Thread mode IPSR to be cleared This bit can be only be used by debugger.
0	Reserved	—	—	Reserved.

五．烧录器的实现思路。

通上以上的介绍，我们已经能够通过 DAP 访问 MCU 内部的资源，包括读写 SRAM，操作 MCU 片内的任意寄存器，操作 CPU 的 DEBUG 寄存器使 CPU 暂停 Halt,复位 reset。

显而易见地，有两种方式实现烧录。

实现方式一：

通过 SW 的 DAP 直接操作 FLASH_CR, FLASH_SR, FLASH_AR 等 embedded flash 寄存器，进 FLASH 进行编程。

实现方式二：

a. 通过 SW 的 DAP，先将 Flashloader 程序写入 SRAM(0x20000000)
b. Halt core
c. 强制更改 PC 指向 SRAM 指定地址，同时更改 MSP，
d. Unhalt core
e. Core 执行 sram 的 Flashloader 程序
f. 向 SRAM 指定区域写 flash image code
g. 步骤 e 和 f 按照某种算法交互
h. 烧写完毕。

补充说明，keil 使用的是方式二，在 keil 程序目录下，有个 flash 文件夹，每个不同芯片都有一段 flash 操作程序，即 Flashloader.