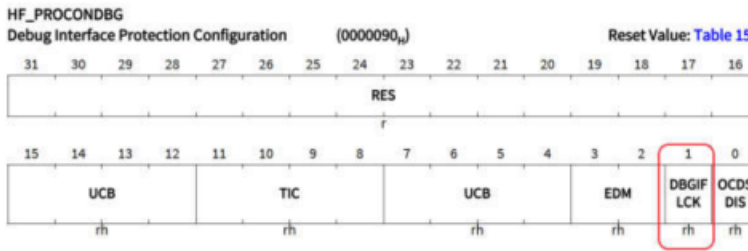
TC3xx 系列 MCU 提供了一种调试加密的方法，一旦调试加密使能，则调试器将不能连接 MCU，但是只要知道密码，**在 UDE 界面输入密码，则调试器就又可以正常连接 MCU 了，这个特点是本文讨论的重点。**

1. Tricore 调试加密以及加密情况下调试方法

1.1 设置调试加密

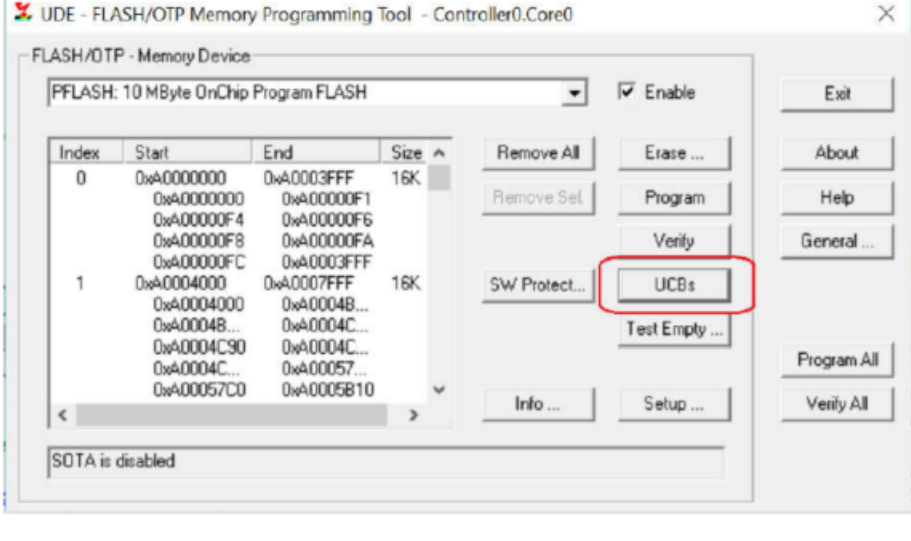
设置调试加密实际上就是下面这个位，注意这个位会在 SSW 自动从 UCB_DBG 中装载到下面寄存器，前提是 UCB_DBG 的 CONFIRMATION 状态会 UNLOCK 或者 CONFIRMED



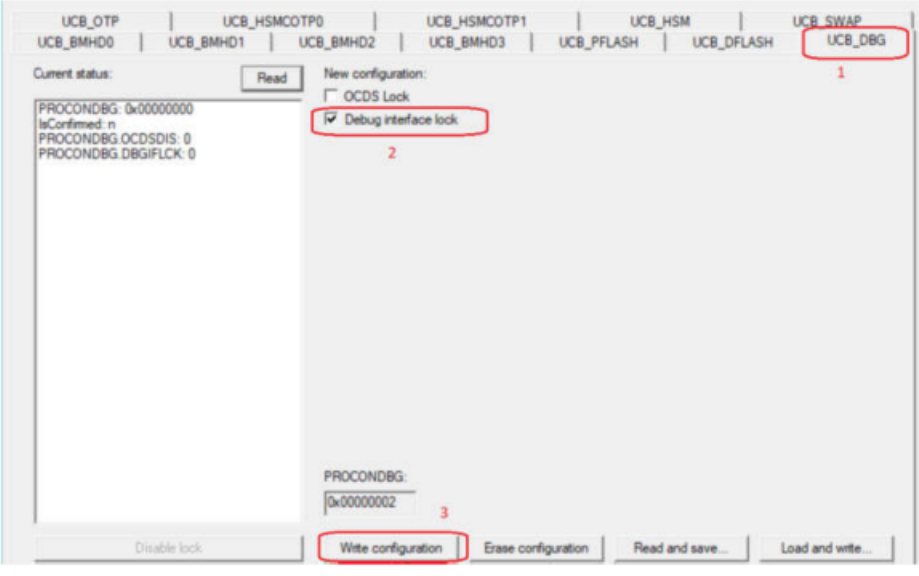
设置调试加密的有三种方法，分别是在 UDE 界面上设置，在程序中用数据定义 UCB 区域内容，在代码中直接用 Flash 命令操作 UCB。

1.1.1 通过 UDE 解密进行设置：

在连接 PLS，load elf 文件后会自动弹出下面对话框（或通过 Tools | Programming 打开），点击 UCBs

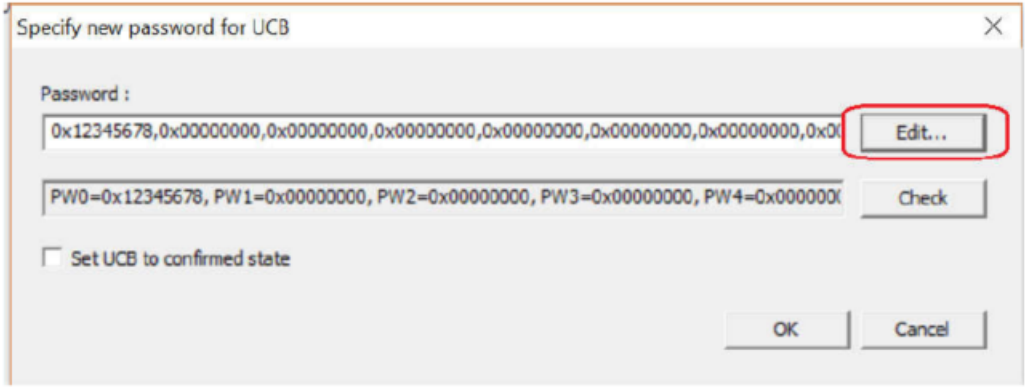


选择 UCB_DBG，勾选"Debug Interface Lock", 点击 Write configuration.



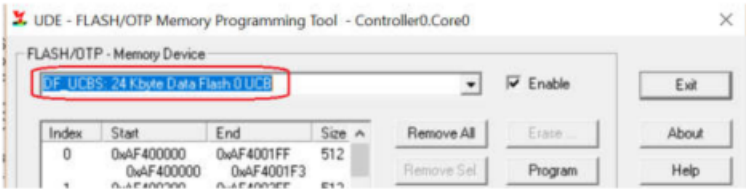
如果要解除保护，则可以点击"Erase configuration " (仅支持在 Unlocked 状态下擦除)

在弹出的下面对话框设置密码(0x12345678), 共有 8 个字 (32 字节)的密码可以设置。



在实际应用中，应该把"Set UCB to confirmed state" 勾选上，把 CONFIRMED 状态改成 CONFIRMED, 具体看第 3 节。(在调试和测试时可以先不勾选)

注：如果操作不成功，则重新连接和加载 elf，以及烧录一下 BMHD（例如下面），再试试。



1.1.2 在程序中用数组定义 UCB 区域内容:

在工厂生产时，UCB 部分内容不可能再使用调试器去设置，这时往往需要把 UCB 内容和程序一起烧录进去，然后重新上电后 UCB 的内容就有效了。可以用数组的方式定义 UCB 内容，来实现这种方式。

UCB_DBG 区域定义如下：

Table 217 Register Overview - UCB18 (ascending Offset Address)

Short Name	Long Name	Offset Address	Page Number
PROCONDBG	DMU_HF_PROCONDBG	2400_H	
PW0	UCB_DBG_ORIG_PW	2500_H	
PW1	UCB_DBG_ORIG_PW	2504_H	
PW2	UCB_DBG_ORIG_PW	2508_H	
PW3	UCB_DBG_ORIG_PW	$250C_H$	
PW4	UCB_DBG_ORIG_PW	2510_H	
PW5	UCB_DBG_ORIG_PW	2514_H	
PW6	UCB_DBG_ORIG_PW	2518_H	
PW7	UCB_DBG_ORIG_PW	$251C_H$	
CONFIRMATION	UCB_DBG_ORIG_CODE	$25F0_H$	

在 Tasking 的 link 文件中，定义一段 UCB 区域：

```
group (ordered)
{
        group  dbg_orig (run_addr=mem:ucb[0x2400])
        {
                select ".rodata.dbg_orig";
        }
        …
}
```

在程序中定义下面数组：

```
#pragma section farrom "dbg_orig"
const Ifx_Dbg_Config dbg_orig =
{
        0x00000002, /* procondbg, DBGIFLOK=1 */
        {
          0x00000000, /**< \brief 0x004: Reserved */
          0x00000000, /**< \brief 0x008: Reserved */
          0x00000000, /**< \brief 0x00C: Reserved */

          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x010: Reserved (0x010 - 0x01F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x020: Reserved (0x020 - 0x02F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x030: Reserved (0x030 - 0x03F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x040: Reserved (0x040 - 0x04F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x050: Reserved (0x050 - 0x05F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x060: Reserved (0x060 - 0x06F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x070: Reserved (0x070 - 0x07F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x080: Reserved (0x080 - 0x08F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x090: Reserved (0x090 - 0x09F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x0A0: Reserved (0x0A0 - 0x0AF) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x0B0: Reserved (0x0B0 - 0x0BF) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x0C0: Reserved (0x0C0 - 0x0CF) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x0D0: Reserved (0x0D0 - 0x0DF) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x0E0: Reserved (0x0E0 - 0x0EF) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x0F0: Reserved (0x0F0 - 0x0FF) */
        },
          0x12345678, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x100: PW0-PW3 */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x110: PE4-PW7 */

        {
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x120: Reserved (0x120 - 0x02F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x130: Reserved (0x130 - 0x03F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x140: Reserved (0x140 - 0x04F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x150: Reserved (0x150 - 0x05F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x160: Reserved (0x160 - 0x06F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x170: Reserved (0x170 - 0x07F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x180: Reserved (0x180 - 0x08F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x190: Reserved (0x190 - 0x09F) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x1A0: Reserved (0x1A0 - 0x0AF) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x1B0: Reserved (0x1B0 - 0x0BF) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x1C0: Reserved (0x1C0 - 0x0CF) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x1D0: Reserved (0x1D0 - 0x0DF) */
          0x00000000, 0x00000000, 0x00000000, 0x00000000, /**< \brief 0x1E0: Reserved (0x1E0 - 0x0EF) */
        },

        0x43211234,    /**< \brief 0x1F0: .confirmation: 32-bit CODE, (always same)*/
        {
          0x00000000, /* 0x004, Reserved */
          0x00000000, /* 0x008, Reserved */
          0x00000000, /* 0x00C, Reserved */
        }
};
```

1.1.3 在代码中直接用 Flash 命令操作 UCB:

可以用 Flash Command 直接写 UCB.

可以用 Erase Logical Sector Range 命令擦 UCB。

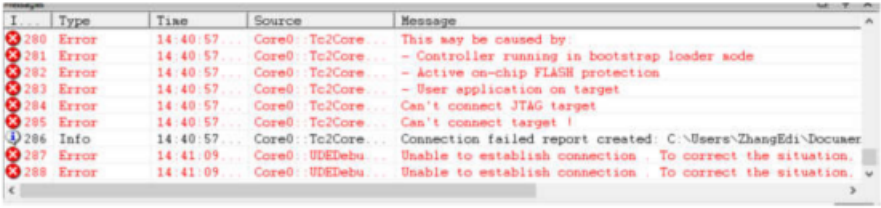
Erase Logical Sector Range	Address Data	.AA50 **SA**	.AA58 .xx**nn**	.AAA8 .xx**80**	.AAA8 .xx**50**		

可以用 Enter Page Mode, Load Page 和 Write Page 命令来写 UCB 区域。

Reset to Read	Address Data	.5554 .xx**F0**				
Enter Page Mode	Address Data	.5554 .xx**5y**				
Load Page	Address Data	.55F0 **WD**				
Write Page	Address Data	.AA50 **PA**	.AA58 .xx**00**	.AAA8 .xx**A0**	.AAA8 .xx**AA**	

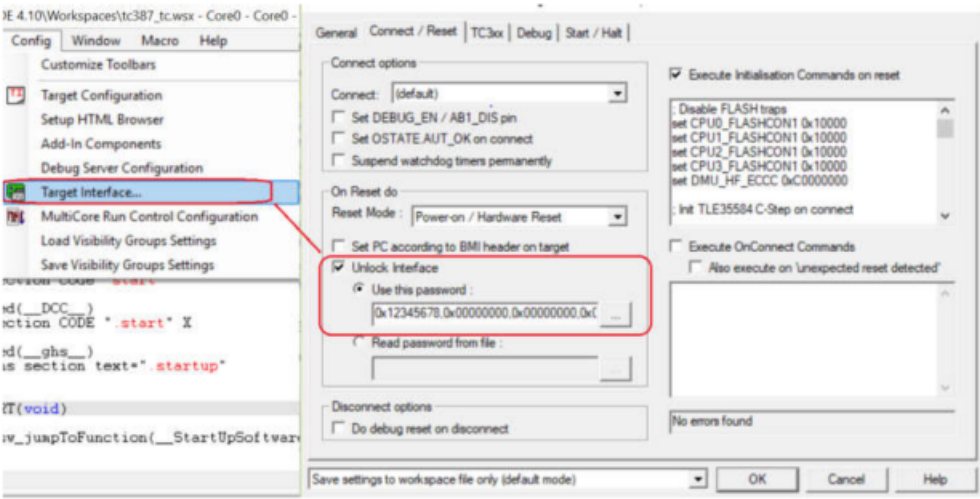
1.2 加密情况下调试

如果在调试加密的情况下，正常连接仿真器会连不上。



I...	Type	Time	Source	Message
280	Error	14:40:57...	Core0::Tc2Core...	This may be caused by:
281	Error	14:40:57...	Core0::Tc2Core...	- Controller running in bootstrap loader mode
282	Error	14:40:57...	Core0::Tc2Core...	- Active on-chip FLASH protection
283	Error	14:40:57...	Core0::Tc2Core...	- User application on target
284	Error	14:40:57...	Core0::Tc2Core...	Can't connect JTAG target
285	Error	14:40:57...	Core0::Tc2Core...	Can't connect target !
286	Info	14:40:57...	Core0::Tc2Core...	Connection failed report created: C:\Users\ZhangEdi\Docuser
287	Error	14:41:09...	Core0::UDEDebu...	Unable to establish connection . To correct the situation,
288	Error	14:41:09...	Core0::UDEDebu...	Unable to establish connection . To correct the situation,

这个时候在下面界面输入正确的密码，然后再次连接，就可以连上了。



下图显示连接成功：

Messages

I...	Type	Time	Source	Message
356	Info	14:45:15...	Core0::UDEMemtool	DF1: Normal sensing mode detected
357	Success	14:45:15...	Core0::UDEMemtool	FLASH programming for device '24 Kbyte Data Flash 0 UCB' re
358	Info	14:45:15...	UAD2CommDev	TriCore2 JTAG/OCDS Debug Protocol, V1.13.0, ID 2 opened
359	Info	14:45:15...	UAD2CommDev	TriCore2 JTAG/OCDS Debug Protocol, V1.13.0, ID 3 opened
360	Info	14:45:15...	UAD2CommDev	TriCore2 JTAG/OCDS Debug Protocol, V1.13.0, ID 4 opened
361	Success	14:45:16...	Core0::UDEDebu...	Connection to TC38xA target established: TriCore (Core0), 1
362	Success	14:45:16...	Core1::UDEDebu...	Connection to TC38xA target established: TriCore (Core1)
363	Success	14:45:16...	Core3::UDEDebu...	Connection to TC38xA target established: TriCore (Core3)
364	Success	14:45:16...	Core2::UDEDebu...	Connection to TC38xA target established: TriCore (Core2)

3. UCB CONFIRMATION 状态

UCB 的 CONFIRMATION 状态分为 UNLOCK 和 CONFIRMED，在 UNLOCK 状态时，UCB 可以重复擦写，当 CONFIRMED 状态时（相当于写保护），只有输入正确的密码，用"Disable Protection"命令对 UCB 临时解密，才能再操作 UCB。在 UDE 界面下，一旦 UCB 处在 CONFIRMED 状态，则就不能再改变 UCB 的状态了。

在量产的程序中，必须改为 CONFIRMED 状态，因为在 UNLOCK 状态下，UCB 的 Password 部分是可以读出来的（在调试器连接不上情况下，也还是可以通过 ASC Bootloader 读出来）。

在使能了调试保护后，当 UCB_DBG 的 CONFIRMATION 改为 CONFIRMED 后，只要知道调试密码，依然还是可以调试。

在使能了调试保护后，当 UCB_HSM 的 CONFIRMATION 改为 CONFIRMED 后，那么 HSM 的代码就永远不能调试了，除非在 HSM 的代码中加入一段后门代码，例如检测到一个信号或者命令后，把 UCB 中的调试保护解除。UCB_HSM 被 CONFIRMED 后，只有 HSM 代码才能操作这块 UCB。

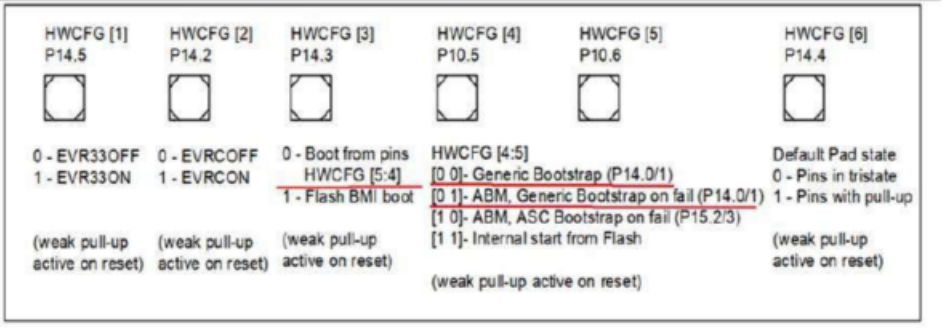
4. 实际应用的安全考虑

在实际应该中，即使 MCU 不能被调试，但是用外部工具通过和 Bootloader 通讯，还是可以把 Flash 程序读出来，有下面两种方式可以进入 Bootloader 模式。

Bootstrap mode set by external pin:

When PINDIS=0, and HWCFG[4:5]=01/00

Boot Mode Index - 16 bit	
PINDIS bit [0]	Mode selection by configuration pins: 0B Mode selection by HWCFG pins is enabled 1BMode selection by HWCFG pins is disabled



Bootstrap mode set by internal BMI:

When PINDIS=1, and HWCFG=100/011

Field Name	Subfield	Description
BMI	Boot Mode Index - 16 bit	
	PINDIS bit [0]	Mode selection by configuration pins: 0B Mode selection by HWCFG pins is enabled 1BMode selection by HWCFG pins is disabled
	HWCFG bits [3:1]	Start-up mode selection: 111B Internal start from Flash 110BAlternate Boot Mode (ABM) 100B Generic Bootstrap Loader Mode (ASC/CAN BSL) 011BASC Bootstrap Loader Mode (ASC BSL) elseinvalid

如果 PINDIS 为 1，而且 HWCFG] = 111. 那么程序一定会从内部 Flash 启动，而且不可能进入任何内部 bootloader。

因此结论是：**调试加密+BMHD 设置（PINDIS 为 1，且 HWCFG] = 111），能保证只有知道调试密码的外部工具，才能连接芯片，用其他手段无法读取 Flash 以及调试 MCU**

对于 HSM 的程序保护，需要把 HSM 程序空间设置为 HSM_Exclusive (操作 UCB_HSMCOTP0/1), 这样主核和外部工具就无法读取 HSM 程序空间了。