S32G How to Ping HSE with Linux Mem Tool

by John Li (nxa08200)

本文说明S32G在Linux中如何使用内存读写工具来发起一个HSE Server服务请求，以确认HSE是否正常工作。本说明的目的旨在在极端缺少Debug手段的情况下，确认HSE的状态。

历史	说明	作者
V1	● 创建本文	● John.Li

目录

1 背景说明与参考资料

1.1 背景说明

一般 S32G 的 HSE 应用会是：

- IVT 加入 HSE FW，加载 HSE FW，或使能 HSE secure boot。
- M 核使用 MCAL Crypto 驱动，通过其中一个 MU 访问 HSE。
- A 核使用 Linux HSE 相关 驱动，中间库等，通过另一个 MU，访问 HSE。

考虑到一些极端的调试手段缺乏的情况，比如说整车上，没有其它调试接口，只有 Linux 的串口可以使用的情况下，我们需要验证 HSE 是否正确工作，除了检查相应的 MU 寄存器外，还有一个办法是模拟出一次 HSE 服务来获得 HSE 的返回值，从而得知 HSE 是否还在工作，也就是 Ping 一下 HSE。

本文采用的测试环境是单纯 Linux 环境，正常情况下应该是 Secure boot+M7 crypto+Linux secure driver 的环境。本文不考虑使用 Linux 的 Crypto 驱动。

本文对 S32G3/G2 均可适用。

针对同一个 MU，系统一般在同步模式且不可抢占访问 HSE channel 的情况下，通常只能用到 Channel 1，所以为避免与正常 HSE 调用冲突，此实验可以使用 Channel 2。

1.2 参考资料

序号	资料	说明	如何获取
1	Linux BSP38	Linux BSP	www.nxp.com 个人帐号下载
2	rm649921-HSE-H&M Firmware Reference Manual(2.1)	HSE FW 手册	www.nxp.com/s32g secure folder，需要申请 secure 访问权限
3	S32G2RM.pdf S32G3RM.pdf	芯片手册	www.nxp.com/s32g 查看 Message Unit 一章
4	AN13495：S32G PKCS Compile and Test Procedure	G2 Linux HSE 应用手册	www.nxp.com/s32g
5	AN14072：Accessing the HSE Using	G3 Linux	www.nxp.com/

		HSE 应用手册	搜索 AN14072
6	S32G - HSE - Guidelines For processing HSE Fatal Error	HSE Log 说明	请与 NXP 技术支持确认如何获得

2 启动包含 HSE 的 Linux 镜像

根据文档《S32G2/3_LinuxBSP_38.0_User_Manual.pdf》使用 yocto 编译出支持 HSE 的镜像，其中，G3 的修改 sample 如下：

```
pwd
~/BSP38/fsl-auto-yocto-bsp/build_s32g399ardb3
vi conf/local.conf
DISTRO_FEATURES:append = "hse"
NXP_FIRMWARE_LOCAL_DIR = "/opt/samba/nxa08200/S32G/BSP38/S32G_FW"
HSE_VERSION = "0_2_22_0"
HSE_SOC_REV = "rev1.1"
```

然后将 HSE_FW_S32G3XX_0_2_22_0 整个放在：/opt/samba/nxa08200/S32G/BSP38/S32G_FW 中。编译生成*.sdcard。

启动后的打印：

```
root@s32g399ardb3:~# dmesg |grep hse
[    0.893724] hse-uio 40210000.mu0b: standard firmware, version 2.22.0
[    0.900170] hse-uio 40210000.mu0b: successfully registered device
[    1.045822] hse 40211000.mu1b: interface mu1b not active
```

3 HSE 服务代码逻辑与寄存器状态

3.1 HSE Demo 示例

为了简化 HSE 驱动流程，我们不以 MCAL Crypto 驱动为例，而是以 HSE Demo 示例为例，只考虑同步方式如下：

```
Demo_app\services\src\hse_get_version.c
HSE_GetVersion_Example
|->hseSrvDescriptor_t* pHseSrvDesc;
typedef struct
```

{

/** @brief The service ID of the HSE message */

hseSrvId_t srvId; //typedef uint32_t hseSrvId_t; 四字节

/** @brief The service metadata (e.g. priority)*/

hseSrvMetaData_t srvMetaData; //typedef struct{uint8_t reserved[4]; } hseSrvMetaData_t; 四字节

/** @brief The service ID will identify a service in the following union */

union

{…

hseGetAttrSrv_t getAttrReq; /**< @brief Request to get a HSE attribute */

| |-> typedef struct

{

/** @brief INPUT: Specifies the HSE attribute ID.*/

hseAttrId_t attrId; // typedef uint16_t hseAttrId_t; #define HSE_FW_VERSION_ATTR_ID ((hseAttrId_t)1U) //两字节低位，0x01 表示为读 FW 版本

uint8_t reserved[2]; //两字节高位，未用。

/** @brief INPUT: Specifies the attribute length (in bytes).The size of the memory location

* must be bigger than or equal to the length of attribute structure */

uint32_t attrLen; //attrID

/** @brief OUTPUT: The address where the attribute will be stored.

* The attribute must be stored in the format of the corresponding attribute Id

* (see the attributes definition). */

HOST_ADDR pAttr; //#define HOST_ADDR uint64_t 读出的信息的地址指针，八个字节

} hseGetAttrSrv_t;

|->hseSrvResponse_t srvResponse = HSE_SRV_RSP_GENERAL_ERROR; #define HSE_SRV_RSP_GENERAL_ERROR ((hseSrvResponse_t)0x33D6D4F1UL)

//希望返回值：#define HSE_SRV_RSP_OK ((hseSrvResponse_t)0x55A5AA33UL)

|->HSE_AllocateFreeChannel

/* Fill the service descriptor */

pHseSrvDesc->srvId = HSE_SRV_ID_GET_ATTR;

#define HSE_SRV_ID_GET_ATTR ((hseSrvId_t)(HSE_SRV_VER_0 | 0x00A50002UL)) /**< @brief Get HSE attribute. */

pHseSrvDesc->hseSrv.getAttrReq.attrId = HSE_FW_VERSION_ATTR_ID;

pHseSrvDesc->hseSrv.getAttrReq.attrLen = sizeof(hseAttrFwVersion_t);

| |->typedef struct

{

uint8_t reserved; /**< @brief For HSE-B, it is used for OTA Config: 0 = Full Mem Config; 1 = AB Swap Config.

For other SOC type: Reserved, expected to be 0 */

uint8_t socTypeId; /**< @brief Identifies the SoC Type ID; same as HSE_PLATFORM from hse_target.h */

uint16_t fwTypeId; /**< @brief Identifies the FW type:

- 0 - Standard FW targeting all customers

- 1 - Premium FW targeting all customers

- 2-7 - Reserved

- 8 >= Custom1, Custom2... etc */

uint8_t majorVersion; /**< @brief Major revision

- 0 - Pre-stabilization releases

- 1 - at first stable interface release, and increased later if breaking changes were introduced */

uint8_t minorVersion; /**< @brief Minor revision, bumped on new compatible changes added;

reset to 0 on majorVersion bump, if majorVersion > 0 */

uint16_t patchVersion; /**< @brief Hotfix release (patch version, bug fix releases).

After majorVersion > 0, reset to 0 on majorVersion or minorVersion bump. */

} hseAttrFwVersion_t;//总共 8 字节

pHseSrvDesc->hseSrv.getAttrReq.pAttr = PTR_TO_HOST_ADDR(&gHseFwVersion);

所以填写好后的格式应该是：

hseSrvDescriptor_t	Byte3	Byte2	Byte1	Byte0
hseSrvId_t(u32)	0x00A50002			
hseSrvMetaData_t(uint8_t reserved[4])	0x00	0x00	0x00	0x00
uint8_t reserved[2]+hseAttrId_t(u16)	0x00	0x00	0x00	0x01
uint32_t attrLen	0x00	0x00	0x00	0x08
Uint64_t pATTR	输出信息地址，注意需要在 nocachable 地址			

|->srvResponse = HSE_Send(MU0, u8MuChannel, gSyncTxOptions, pHseSrvDesc);

| |->HSE_MU_SEND_NON_BLOCKING(u8MuInstance, u8MuChannel, (uintptr_t)pHseSrvDesc);

#define HSE_MU_SEND_NON_BLOCKING(u8MuIf, u8Channel, value) (MU_REG_WRITE32 ((u32BaseAddr[u8MuIf] + MU_TR_OFFSET + (uint32_t)(u8Channel << 2U)), (value))) //向相应的 TR 里写入 hseSrvDescriptor_t 的地址。

| |-> srvResponse = HSE_MU_ReceiveResponseBlocking(u8MuInstance, u8MuChannel);

| | |->HSE_MU_IsResponseReady(u8MuInstance, u8Channel)

| | | |->HSE_MU_READ_RECEIVE_STATUS_REGISTER(u8MuInstance), u8Channel)

#define HSE_MU_READ_RECEIVE_STATUS_REGISTER(u8MuIf) (MU_REG_READ32 (u32BaseAddr[u8MuIf] + MU_RSR_OFFSET))//如果相应的 RSR 置位。

| | | |->u32HseMuResponse = HSE_MU_ReceiveResponse(u8MuInstance, u8Channel);

| | | | |-> HSE_MU_RECEIVE_NON_BLOCKING(u8MuInstance, u8Channel);

#define HSE_MU_RECEIVE_NON_BLOCKING(u8MuIf, u8Channel) (MU_REG_READ32 (u32BaseAddr[u8MuIf] + MU_RR_OFFSET + (uint32_t)(u8Channel << 2U))) //从对应的 RR 中读出 u32HseMuResponse,正确值应该是：#define HSE_SRV_RSP_OK ((hseSrvResponse_t)0x55A5AA33UL)

| | | | |->#define HSE_MU_READ_FLAG_STATUS_REGISTER(u8MuIf) (MU_REG_READ32 (u32BaseAddr[u8MuIf] + MU_FSR_OFFSET))直到 HSE 将对应 FSR 清位，表示 HSE 已经释放对应 Channel.

所以可以总结出使用同步方式读取 HSE 版本的方法如下：

1. 在 no-cachable 的一段内存中准备一个 hseSrvDescriptor_t 数据结构，如上 get attribute 的数据结构大小是 4x6=24 字节。注意此数据结构的输出信息指针也需要在 non-cachable 地址。
2. 将相应的数据结构地址写到对应 Channel 的 TR 寄存器中。
3. 等一段时间读取对应 Channel 的 RSR 寄存器，如果为 1，则证明已经收到了 HSE 的回复。
4. 从对应 Channel 的 RR 寄存器中读出 HSE 的返回值，看看是不是 0x55A5AA33UL，如果是则证明返回正确。
5. 这个时候就可以从 hseSrvDescriptor_t 的 pAttr 成员中获得返回信息指针，到此地址可以看到返回值。
6. 如果要继续进行下一步操作，则需要先确认 FSR 的相应 Channel 是否已经释放。

注意：

- 一般来讲，同步模式下，系统绝大部分时间只会用到 MU 的 Channel 1。
- HSE Demo 中的获得空闲 Channel 是使用软件标注的办法，Crypto 驱动中是使用检查 FSR/RSR/TSR 寄存器的办法。
- 如果考虑到一般正常驱动会使用 Channel 1 的情况，则此测试方法可以直接使用 Channel 2，这样就不会影响到正常驱动的使用。

3.2 IDEL 情况下 MU 寄存器状态

1. FSR:

62.3.2.1 MUB memory map

MU0.MUB base address: 4021_0000h

104	Flag Status Register (FSR)	32	RO	0000_0000

FSR 的详细定义在 HSE RM 中：

Table 8: HSE global status bits in FSR

Bit #	Description
31	HSE_STATUS_BACKUP_SYS_IMAGE; when set to 1, indicates that the backup SYS-IMG (vs. main) has been loaded by the HSE
30	HSE_STATUS_PRIMARY_SYS_IMAGE; when set to 1, indicates that the main SYS-IMG (vs. backup) has been loaded by the HSE
29	HSE_STATUS_PUBLISH_SYS_IMAGE; when set to 1, indicates that SYS-IMG must be written (updated) in application NVM
28	HSE_STATUS_OEM_SUPER_USER; when set to 1, indicates that SU rights are granted to OWNER_OEM
27	HSE_STATUS_CUST_SUPER_USER; when set to 1, indicates that SU rights are granted to OWNER_CUST
26	HSE_STATUS_BOOT_OK; set to 1 when the secure boot conditions (pre-boot phase) defined in the HSE successfully pass
25	HSE_STATUS_INSTALL_OK; set to 1 once the key catalogs have been successfully formatted; when cleared to 0, indicates to the host that the key catalogs must be formatted
24	HSE_STATUS_INIT_OK; set to 1 when the HSE initialization is completed; when cleared to 0, no service request can be made to the HSE (MU disabled)
23	HSE_STATUS_HSE_DEBUGGER_ACTIVE; set to 1 when a HSE debug session is active
22	HSE_STATUS_HOST_DEBUGGER_ACTIVE; set to 1 when a host debug session is active
21	HSE_STATUS_RNG_INIT_OK; set to 1 when the RNG initialization is complete; when cleared to 0, any services using random number is unavailable to the host
20	HSE_SHE_STATUS_SECURE_BOOT_OK; set to 1 when SMR #0 successfully verified against BOOT_MAC
19	HSE_SHE_STATUS_SECURE_BOOT_FINISHED; set to 1 when SMR #0 was not successfully verified
18	HSE_SHE_STATUS_SECURE_BOOT_INIT; set to 1 when SMR #0 has been installed and authenticated with BOOT_MAC_KEY
17	HSE_SHE_STATUS_SECURE_BOOT; set to 1 when SMR #0 has been installed and BOOT_SEQ equals 1
16	RFU
15	Set to 1 when service channel #15 execution is in progress
14	Set to 1 when service channel #14 execution is in progress
...	...
2	Set to 1 when service channel #2 execution is in progress
1	Set to 1 when service channel #1 execution is in progress
0	Set to 1 when service channel #0 execution is in progress

其中低 16 位表示相应 Channel 是否是被占用。

devmem2 0x40210104

Read at address 0x40210104 (0x7f90d60104): 0x09200000
//HSE_STATUS_INITOK|HSE_STATUS_RNG_INIT_OK|HSE_STATUS_CUST_SUPER_USER

2. GSR:

118	General-purpose Status Register (GSR)	32	W1C	0000_0000

GSR 的详细定义在 HSE RM 中：

Table 105: HSE system events logging in GSR

Bit #	Description
31-16	The most significant 16 bits are reserved for **NXP internal errors**. These bits are set when a fatal error is triggered (one of the first 8 bits is set).
12-15	RFU
11	HSE_WA_PUBLISH_COUNTER_TBL; the application must publish and store the monotonic counter table.
10	HSE_WA_RNG_NOT_INIT; RNG is not initialized. RNG-based services may be delayed as the HSE attempts to re-initialize the RNG.
9	RFU
8	HSE_WA_SMR_PERIODIC_CHECK_FAILED; warning event, signaling that SMR periodic check failed. The application can read the HSE_SMR_CORE_BOOT_STATUS_ATTR_ID attribute to see what SMR failed.
1-7	RFU
0	HSE_ERR_GENERAL; set to 1 when a fatal error (e.g. multi-bit ECC error) or intrusion detection is detected in the HSE; this error cannot be recovered from, and a system reset should be triggered.

其中高 16 位表征了 NXP 内部错误码：所以如果此寄存器有值，则证明 HSE 发生了错误，则需要与 NXP 确认内部错误码表达为何种错误。

devmem2 0x40210118

Read at address 0x40210118 (0x7f892c3118): 0x00000000

3. TSR:

124	Transmit Status Register (TSR)	32	RO	0000_FFFF

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TE15	TE14	TE13	TE12	TE11	TE10	TE9	TE8	TE7	TE6	TE5	TE4	TE3	TE2	TE1	TE0
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fields

Field	Description
31-16 —	Reserved
15-0 TEn	MUB Transmit Register n Empty Indicates whether MUB Transmit Register n is empty. There are 16 Transmit Registers (n = 0 to 15). This field is set after the MUA_RRn register is read on the MUA side. Setting TEn to 1 informs the MUB side that the MUB_TRn register is ready to be written on the MUB side. If MUB_TCR[TIEn] = 1, a Transmit n interrupt is issued on the MUB side. This field is cleared after the MUB_TRn register is written on the MUB side. After this field is cleared, if MUB_TCR[TIEn] = 1, the Transmit n interrupt request is cleared on the MUB side. This field is set when MU resets.

Table continues on the next page

TSR 默认值是全 1，他表示对应 Channel 的 TR 是为空，当我们将 hseSrvDescriptor_t 地址写入对应 Channel 的 TR 后，则对应的 TSR bit 会被 MU 硬件自动设置为 0，而当 HSE 读取完 TR 后，MU 硬件又自动将 TSR 对应 bit 设置为 1，这个是 MU 硬件行为，HSE 和 Host 并不操作 TSR，不过通过读取 TSR 值是否为 1 可以得之对应 TR 是否为空。

devmem2 0x40210124

Read at address 0x40210124 (0x7fa2d1f124): 0x0000FFFF

4. RSR

12C	Receive Status Register (RSR)	32	RO	0000_0000

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RF15	RF14	RF13	RF12	RF11	RF10	RF9	RF8	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fields

Field	Description
31-16 —	Reserved
15-0 RFn	MUB Receive Register n Full Indicates whether Receive Register n is full. There are 16 Receive Registers (n = 0 to 15). This field becomes 1 when the MUA_TRn register is written on the MUA side. Setting this field to 1 informs the MUB side that new data in the MUB_RRn register is ready for MUB to read it. If MUB_RCR[RIEn] = 1, a Receive n interrupt is issued on the MUB side. This field becomes 0 when the MUB_RRn register is read, or when MU is reset. After this field becomes 0, if MUB_RCR[RIEn] = 1, the Receive n interrupt request is cleared on the MUB side. 0 - Not full 1 - MUB_RRn register has received data from MUA TRn register and is ready for MUB to read it.

RSR 默认值是全 0，代表对应 Channel 的 RR 寄存器为空，当我们将对应 Channel 的 RR 寄存器的值读走后，对应的 RSR bit 会被 MU 硬件自动设置为 0，而当 HSE 将 response 写入对应 RR 寄存器后，MU 硬件又自动将 RSR 对应 bit 设置为 1，这个是 MU 硬件行为，HSE 和 Host 并不操作 RSR，不过通过读取 RSR 值是否为 0 可以得知对应 RR 是否为空。

```
devmem2 0x4021012C
Read at address  0x4021012C (0x7fa08e912c): 0x00000000
```

综上所述，如果要判断一个 Channel 是否可用，从寄存器来说，先读 FSR 对应 bit 是否为 0 看实际 HSE 内部的 Channel 通道是否为空闲，再读 RSR 对应 bit 是否为 0 看对应的 RR 是否被 Host 读走，最后读 TSR 对应 bit 是否为 1 看对应的 TR 是否被 HSE 读走。

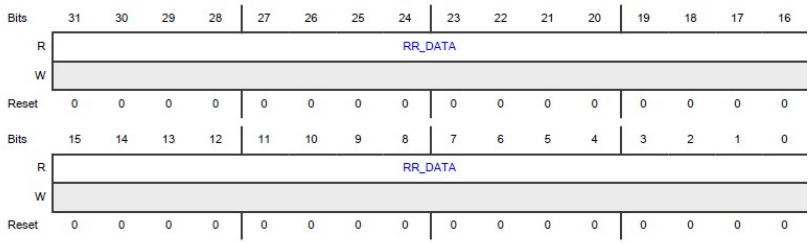
5. TR2

200 - 23C	Transmit Register (TR0 - TR15)	32	WORZ	0000_0000

TR 为只写寄存器，所以不用读取了。

6. RR2

280 - 2BC	Receive Register (RR0 - RR15)	32	RO	0000_0000



Fields

Field	Description
31-0 RR_DATA	MUB Receive Data Reflects the data written to MUA Transmit Register (TRn). Reading this register clears MUB_RSR[RFn] on the receiver side, and sets MUA_TSR[TEn] on the transmitter side. RRn register can be read only when MUA_RSR[RFn] = 1. Reading before MUA_RSR[RFn] = 1 may result in reading incorrect data. You must poll MUA_RSR[RFn] to confirm it is set before reading RRn. Writing to this register generates an error response to MUB.

RR 放在最后读取，因为读走 RR 后，对应的 RSR bit 会清 0，则 RSR 就不准确了，RR 中存放的是 HSE 返回码，正确值是：#define HSE_SRV_RSP_OK ((hseSrvResponse_t)0x55A5AA33UL)。由于我们打算使用 Channel 2 来访问 HSE，所以应该读取 RR2:

```
devmem2 0x40210288
Read at address  0x40210288 (0x7f9c90d288): 0x00000000
```

4 使用 Linux memtool 命令来访问 HSE

4.1 检查 HSE 状态

- 通过读取GSR，可以判断出HSE是否出错而ShutDown。
- 通过读取FSR, RSR, TSR可以判断出HSE是否处于Busy状态，可以多读几次，看看常用到的 Channel(如果确实有HSE服务调用存在的话)。
- 通过读取 RRn，可以获得上次 HSE 调用的返回值，从而判断出上次 HSE 是否报错。

4.2 准备 hseSrvDescriptor_t 数据结构

根据文件：C:\NXP\SW32G_RTD_4.4_4.0.2\eclipse\plugins\Platform_TS_T40D11M40I2R0\startup\include\core_specific.h：

```
3  HSE Shared RAM     0x22C00000  0x22C03FFF
7  Non-Cacheable RAM  0x34500000  0x345FFFFF
```

为 Non-cacheable 地址，为了不影响其它 HSE 操作，不使用 HSE Share 地址，一般建议使用 SRAM 的 Non-cacheable 地址。注意，选择的地址要保证没有被用到，以避免误写，一个 hseSrvDescriptor_t 数据结构最大为 172 字节，本文使用的 get attribute 服务只有 24 字节。

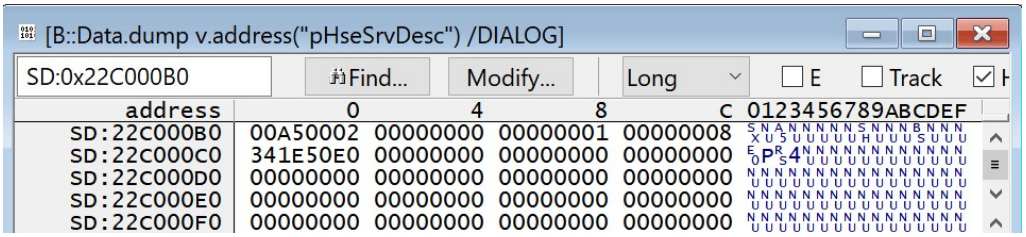
由于 Linux 环境中使用了 SRAM 地址，为了避免误写，本文采用 SSRAM 地址，地址如下：

4 Standby RAM 0x24000000 0x24007FFF

0x24000000 做为 hseSrvDescriptor_t 地址，0x24006000 做为输出数据地址。

hseSrvDescriptor_t	Byte3	Byte2	Byte1	Byte0
hseSrvId_t(u32)	0x00A50002 //HSE_SRV_ID_GET_ATTR			
hseSrvMetaData_t(uint8_t reserved[4])	0x00	0x00	0x00	0x00
uint8_t reserved[2]+hseAttrId_t(u16)	0x00	0x00	0x00	0x01
uint32_t attrLen	0x00	0x00	0x00	0x08
Uint64_t pATTR	输出信息地址，注意需要在 nocachable 地址			

相似示例如下：



devmem2 0x24000000 w 0x00A50002 //放入 Server ID 为：#define HSE_SRV_ID_GET_ATTR ((hseSrvId_t)(HSE_SRV_VER_0 | 0x00A50002UL))

devmem2 0x24000004 w 0x00 //放入 reserved word

devmem2 0x24000008 w 0x01 //放入 attribut ID 为：#define HSE_FW_VERSION_ATTR_ID ((hseAttrId_t)1U)

devmem2 0x2400000c w 0x08 //放入输出数据长度

devmem2 0x24000010 w 0x24006000//放入输出指针为 0x24006000

devmem2 0x24000014 w 0x0

devmem2 0x24006000 w 0x0 //清除目标内存

devmem2 0x24006004 w 0x0

echo 3 > /proc/sys/vm/drop_caches //将 cache 中内容同步到内存

4.3 申请 HSE 服务

devmem2 0x40210288 //读出 RR2，清除一下。

devmem2 0x40210208 w 0x24000000 //将 hseSrvDescriptor_t 地址放入 TR2

等待一段时间：

devmem2 0x40210288 //读出 RR2，看是不是 0x55A5AA33 (注意实际)

Read at address 0x40210288 (0x7f7fec6288): 0x55A5AA33

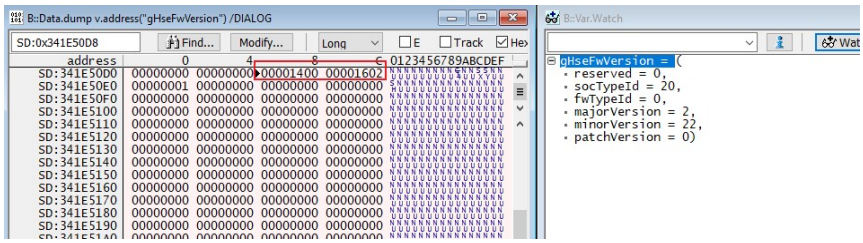
devmem2 0x24006000 //从输出指针读出返回值，此应该为 FW 版本号。

Read at address 0x24006000 (0x7f99acf000): 0x00001400

devmem2 0x24006004

Read at address 0x24006004 (0x7faa36f004): 0x00001602

对比如下：



HSE FW 0_2_22_0。

5 其它建议

- hseSrvDescriptor_t应编程为全局变量，则可以容易其描述符信息。
- 如果是编程实际hseSrvDescriptor_t的配置，建议在发送给TR2寄存器前执行语句DMB语句。
- 调节LOG的建议，请参考文档《S32G - HSE - Guidelines For processing HSE Fatal Error》 WangXuewei。