#include <braddress.h> //IRAM\_BASE = 0x70000000, BOOTROM\_STACK = (IRAM\_BASE + 0x10000)

#if defined(DEBUG)

#define ASSERT\_FATAL(bCond) BootAssertFatal(bCond)

#else // RELEASE

#define ASSERT\_FATAL(bCond)

#endif

**typedef** struct **{** // BootLoaderHeader

uint8\_t blSig**[**RSA\_SIG\_SIZE**];**

uint8\_t blHash**[**MD5\_HASH\_SIZE**];**

uint8\_t blKey**[**AES\_KEY\_SIZE**];**

uint32\_t blVersion**;**

uint32\_t blBlock**;**

uint32\_t blSize**;**

uint32\_t blEntryOffset**;**

uint8\_t \_pad**[**BLHDR\_AES\_PAD\_SIZE**];**

**}** BLHeader**;**

extern uint32\_t FuseGetVersion**(**void**);**

extern void BootAssertFatal**(**int bCond**);**

extern BootError FaultDetectRandomLengthDelay**(**void**);**

extern BootError ReadFromEmmc**(**uint32\_t nBlock**,** void **\***pDest**,** uint32\_t nSize**);**

// Compute hash digest and memcmp hash

extern BootError ValidateHash**(**const void **\***pHash**,** const void **\***pData**,** uint32\_t nSize**);**

// Compute signature digest and memcmp hash

extern BootError ValidateSig**(**const void **\***pSig**,** const void **\***pHash**);**

extern void AesEcbDecrypt**(**void **\***pKey**,** void **\***pData**,** uint32\_t nSize**);**

extern void \_asmJumpToAddress**(**void **\***pAddr**);**

#define BL\_BASE IRAM\_BASE

BootError LoadRunBootloader**(**void**)** **{**

BootError rc**;**

BLHeader **\***pHdr**;**

void **\***pEntry**;**

void **\***pBootloader**;**

rc **=** FaultDetectRandomLengthDelay**();**

**if** **(**rc **!=** BootError\_Success**)** **goto** \_error**;**

pHdr **=** **(**BLHeader**\*)**BL\_BASE**;**

rc **=** ReadFromEmmc**(0,** pHdr**,** **sizeof(\***pHdr**));**

**if** **(**rc **!=** BootError\_Success**)** **goto** \_error**;**

rc **=** FaultDetectRandomLengthDelay**();**

**if** **(**rc **!=** BootError\_Success**)** **goto** \_error**;**

ASSERT\_FATAL**(**FuseGetVersion**()** **==** pHdr**->**blVersion**);**

pBootloader **=** pHdr **+** **1;**

rc **=** ReadFromEmmc**(**pHdr**->**blBlock**,** pBootloader**,** pHdr**->**blSize**);**

**if** **(**rc **!=** BootError\_Success**)** **goto** \_error**;**

rc **=** ValidateHash**(**pHdr**->**blHash**,** pBootloader**,** pHdr**->**blSize**);**

**if** **(**rc **!=** BootError\_Success**)** **goto** \_error**;**

rc **=** ValidateSig**(**pHdr**->**blSig**,** pHdr**->**blHash**);**

**if** **(**rc **!=** BootError\_Success**)** **goto** \_error**;**

rc **=** FaultDetectRandomLengthDelay**();**

**if** **(**rc **!=** BootError\_Success**)** **goto** \_error**;**

AesEcbDecrypt**(**pHdr**->**blKey**,** pBootloader**,** pHdr**->**blSize**);**

pEntry **=** **(**uint8\_t**\*)**pBootloader **+** pHdr**->**blEntryOffset**;**

BootAssertFatal**(**

**((**uint32\_t**)**pEntry **>=** IRAM\_BASE**)** **&&**

**((**uint32\_t**)**pEntry **<** IRAM\_END**)**

**);**

\_asmJumpToAddress**(**pEntry**);**

\_error**:**

**return** rc**;**

**}**