恩智浦半导体
应用笔记

文档编号：AN12218
第 1 版，2018年10月

S32K1xx 引导加载程序

作者：恩智浦半导体

目录

1. 简介

以下文档介绍了 S32K1 MCU 中 S32K1xx 引导加载程序的架构和用法。

此引导加载程序支持通用异步接收器/发送器 (UART) 作为通信接口，并且可以轻松修改以支持其他类型的通信接口。

2. 架构说明

引导加载程序分为三层：

- 引导加载程序 — 负责启动用户应用程序并轮询传入数据。
- 通信处理/内存处理 — 负责处理接收到的数据并处理对非易失性存储器的写入。
- 微控制器驱动程序 — 负责处理与微控制器上可用的实际外设的所有低级通信。

下图展示了引导加载程序的架构图：

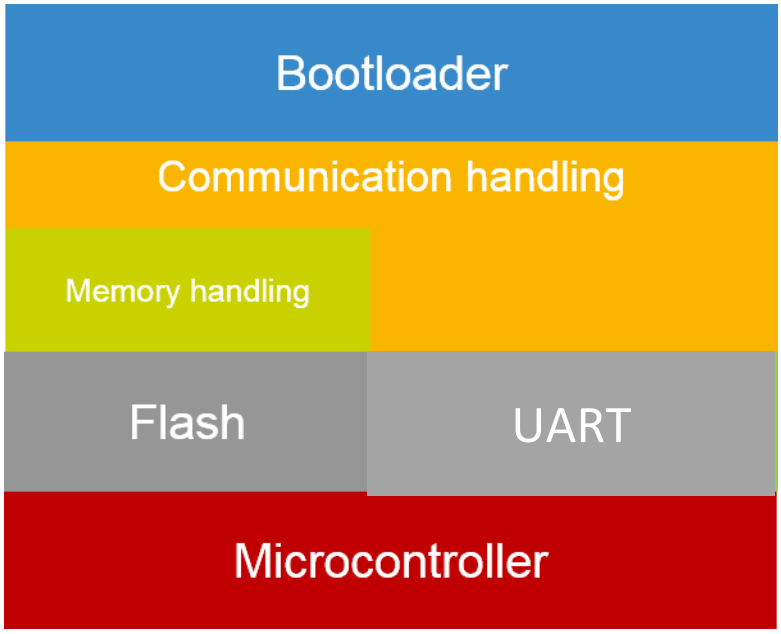


图1. 引导加载程序架构

引导加载程序位于 D-Flash 部分。应用程序应该放在 P-Flash 部分。将引导加载程序存储到 D-Flash 部分背后的想法是允许应用程序使用整个 P-Flash，而无需为引导加载程序保留一个部分。下图展示了引导加载程序具有且应用程序必须遵循的内存布局。

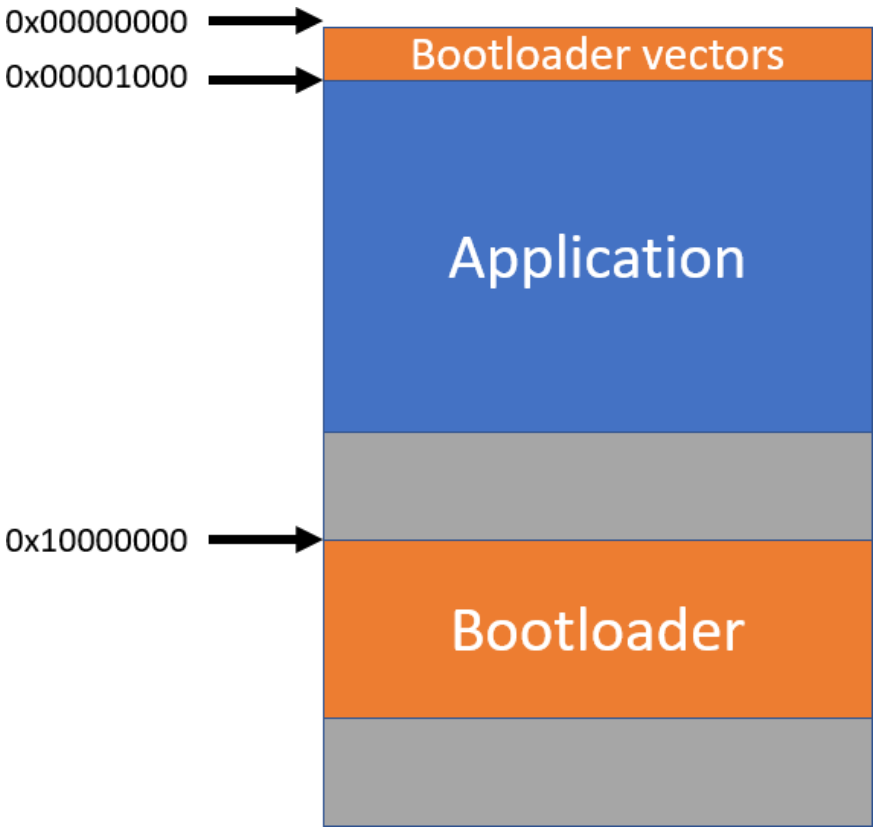


图2. 内存布局

2.1. 引导加载程序工作流程概述

引导加载程序工作流程可以在下图中观察到。

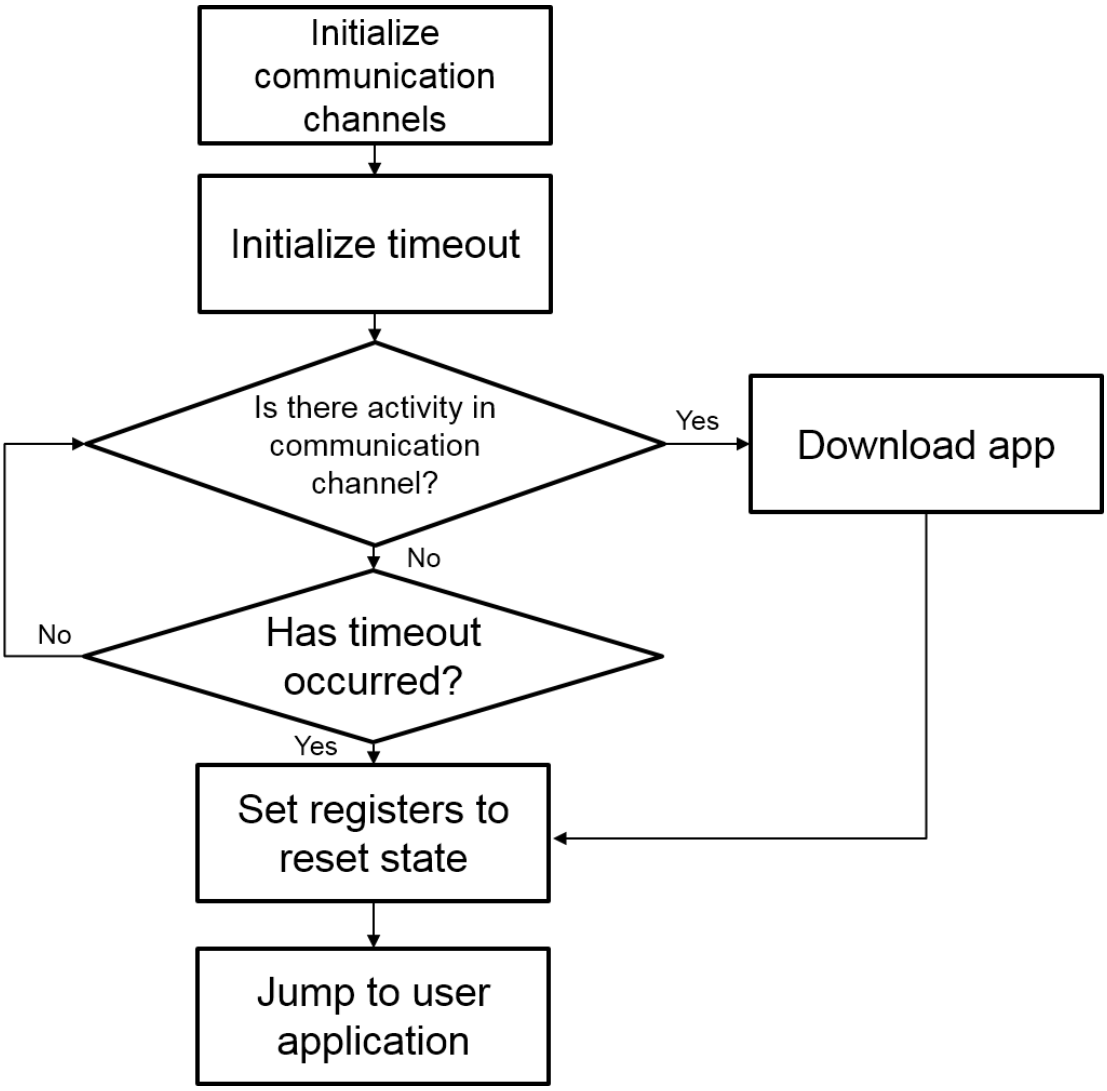


图3. 引导加载程序工作流程

第一步是初始化可用的通信通道，在这种情况下仅可用，但如果需要另一个通信通道，则应在此处调用其初始化例程。

要选择要使用的通信通道，只需修改第 11 行中的“sources/drivers/inc/comm.h”以选择要使用的通信接口。将预处理器指令设置为 0 会禁用通信接口，将其设置为 1 会启用它。

```
/* Define communication interfaces to use, 0-> Disable 1-> Enable */
#define UART_COMM                1
```

通信接口可以同时工作，但由于引导加载程序针对大小进行了优化，因此必须修改引导加载程序的链接器文件以适应生成的代码。因此，建议一次只使用一种通讯方式。如果需要两个接口，第一个检测总线活动的接口将用于下载应用程序，默认情况下引导加载程序设置为仅使用UART通信。

第二步是初始化超时机制。复位后，微控制器将轮询选定的通信通道，如果在超时机制允许的时间内没有检测到活动，设备将尝试执行最后加载的应用程序，如果设备没有收到应用程序，它将卡住 在一个循环中。为了尝试下载应用程序，需要再次重置。

超时值是可配置的，默认设置为五秒。只能选择一秒倍数，为了更改超时值，只需在“sources/drivers/inc/timeout.h”第 14 行中设置所需的值。

```
/* Define timeout value, the base is 1s */
#define TIMEOUT_VAL             5
```

一旦超时机制被初始化，设备开始在超时值分配的时间内轮询通信信道中的活动。如果在通信通道中检测到活动，引导加载程序开始通过选定的通信通道（例如 UART）下载应用程序。

如果发生超时或应用程序闪存到设备，引导加载程序会禁用所有已修改的寄存器并将其设置为其重置状态，则需要执行此步骤以确保应用程序在接近退出重置状态的环境中开始执行。

一旦寄存器被设置为其复位状态，设备就会尝试跳转到用户应用程序。

2.2. 通信处理概述

通信处理例程执行的第一步是通过选定的通道获取SREC“段”。段只是SREC文件的一行。可以在下面找到SREC文件的两行（短语）：

```
S00F000068656C6C6F202020202000003C
S11F00007C0802A6900100049421FFF07C6C1B787C8C23783C600000386300002 6
```

Record type Byte count Address Data Checksum

SREC线路的结构如下图所示：

S	Type	Byte Count	Address	Data	Checksum

前两个字符以ASCII格式发送，'S' 和SREC类型（例如'0'、'1'…'9'），其余数据转换为其十六进制表示并发送（而不是发送 '0' 和 'F' 0x0F 被发送）。有关SREC格式的详细说明，请参阅 webpage。

该段被接收并存储在以下结构中：

```
typedef union
{
	uint8_t Byte[MAX_PHSIZE_BP];		/* Byte level access to the Phrase */
	struct
	{
		char PhraseType;	/* Type of received record (e.g. S0, S1, S5, S9...) */
		uint8_t PhraseSize; /* Phrase size (address + data + checksum) */
		/* Address, depending on the type of record it might vary */
		uint8_t PhraseAddress[MAX_ADDRESS_BP]__attribute__((aligned (32)));
		/* Maximum 32 data bytes */
		uint8_t PhraseData[MAX_DATA_BP]__attribute__((aligned (32)));
		uint8_t PhraseCRC; /* Checksum of size + address + data */
	}F;

}BootPhraseStruct;
```

该结构包含 SREC 段中提供的所有信息，例如记录类型、字节数、地址、数据和循环冗余校验（CRC）。

一旦结构被填充，就会检查它是否包含有效的记录类型（即在“0”和“9”之内），它的大小是否在 SREC 最大值内，并且 CRC 与接收到的数据一起计算并进行比较与收到的CRC。如果不满足这些条件中的任何一个，即无效的记录类型、无效的记录大小或 CRC 不匹配，则将 ERR_CRC (0x41) 信号发送回发送数据的设备。如果一切正常，接收到的数据将被处理并发送一个 ERR_OK (0x45) 信号作为确认。

如果接收到的记录类型携带要写入微控制器的数据（“1”、“2”或“3”），则数据由存储器处理层处理和写入。

重复此过程，直到接收到终止标志（‘7’、‘8’或‘9’），一旦接收到该标志，通信处理例程结束并返回到引导加载程序。

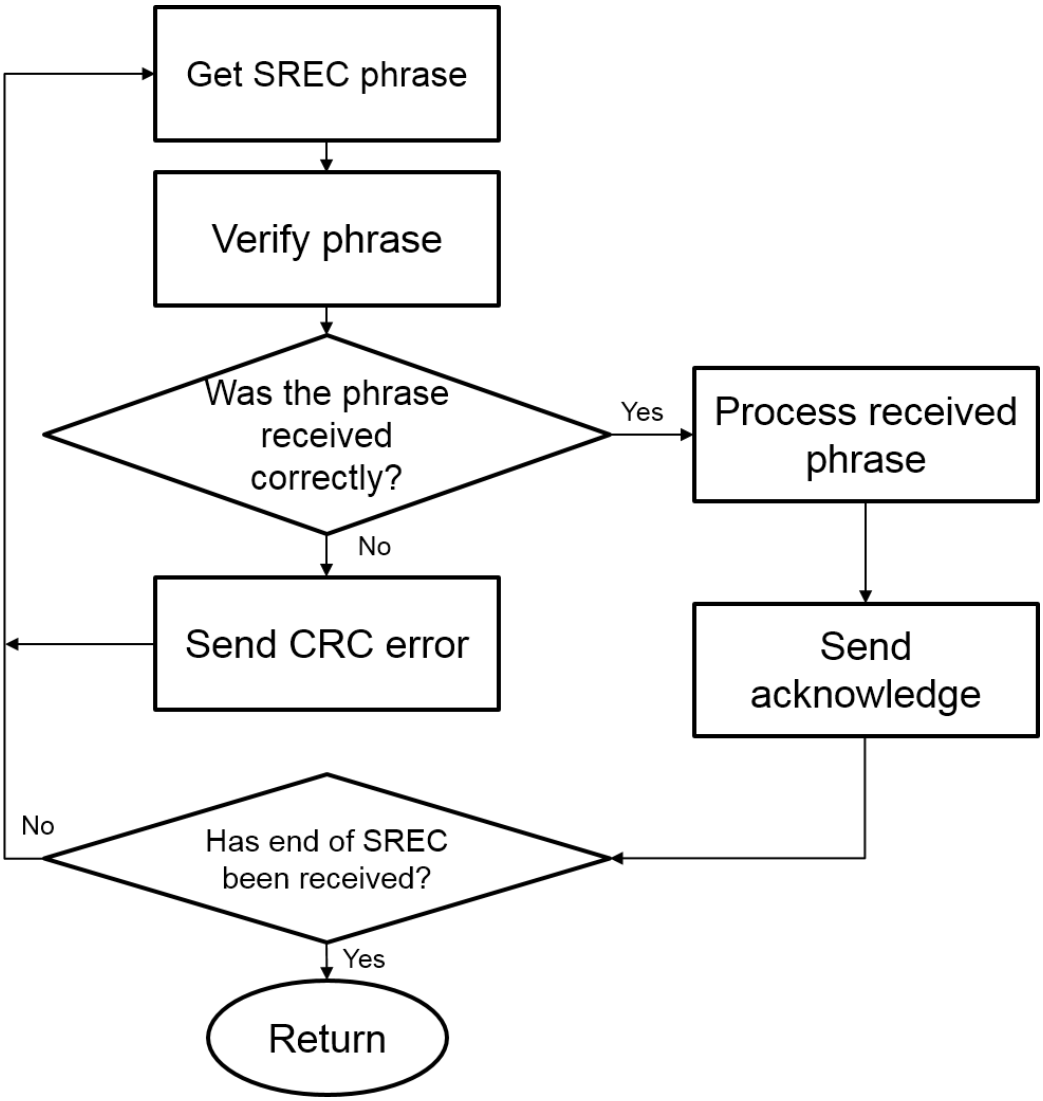


图4. 通信处理工作流

2.2.1. UART 通信

当使用UART通信接收时，数据流过程如下：

1. 首先必须发送一个“s”（0x53）来表示SREC行的开始。
2. 下一个预期字节表示要接收的SREC类型，也是以ASCII格式发送的，例如‘0’ (0x30)、‘1’ (0x31)、‘2’ (0x32)、……‘9’ (0x39)。
3. SREC行的其余部分（字节数、地址、数据和校验和）从 ASCII 转换为其十六进制表示，例如 从“AF”（0x41 和 0x46）到 0xAF。执行此操作后，数据将通过UART发送并且 get_phrase 例程停止，直到它收到 SREC行上的 “字节计数” 字段所指示的所有字节。

4. 作为最后一步，验证接收到的数据，如果数据正确接收，则将错误信号发送回信号，如果检测到错误，则发送ERR_CRC（0x41）信号。
    a. 如果主节点收到：
        i. 一个ERR_OK信号，发送下一个SREC短语。
        ii. 一个ERR_CRC信号，同样的SREC短语被再次发送，直到从机正确接受它。

在此过程之后，设备准备好接收下一个SREC短语，直到到达文件末尾。

3. 建立兼容的应用程序

应用程序应从Flash的0x1000（4kB）开始，其向量表应放置在该地址。

编译与此引导加载程序兼容的应用程序的一种简单快捷的方法是简单地向链接文件的存储器部分添加4kB的偏移量，下面显示了两个IDE的一些示例。

在 S32DS 上

```
/* Specify the memory areas */
MEMORY
{
  /* Flash */
  m_interrupts          (RX)  : ORIGIN = 0x00001000, LENGTH = 0x00000400
  m_flash_config        (RX)  : ORIGIN = 0x00001400, LENGTH = 0x00000010
  m_text                (RX)  : ORIGIN = 0x00001410, LENGTH = 0x0017EBF0

  /* SRAM_L */
  m_data                (RW)  : ORIGIN = 0x1FFE0000, LENGTH = 0x00020000

  /* SRAM_U */
  m_data_2              (RW)  : ORIGIN = 0x20000000, LENGTH = 0x0001F000
}
```

4. 使用引导加载程序

本应用笔记中包含的示例软件适用于S32K148 EVB。但是，它可以轻松迁移到其他S32K1xx芯片。引导加载程序期望以SREC格式加载文件，有关如何在S32DS上生成SREC文件的说明，请参阅Appendix A（附录A）。本应用笔记包中提供了预编译的示例SREC文件。这是一个简单的例程，可以在S32K148 EVB上运行。

注意：

某些IDE将项目名称放在第一个SREC行（S0）中，只要项目名称超过27个字符，这就会导致引导加载程序出现问题。每行的最大数据为32个字节，但IDE将字符串“.srec”附加到项目名称，因此最多允许27个字符。

引导加载程序支持一个通信通道：

- UART

4.1. UART 接口

使用UART接口时，只需打开位于“Java接口/”中的Java应用程序并按照以下步骤操作：

1. 选择通讯端口。
2. 选择波特率，默认波特率为19200。
3. 选择要发送的SREC文件。
4. 点击下载，SREC文件将会被逐行发送。

注意

在尝试使用该接口之前，必须安装Java JRE 8 32位版本。

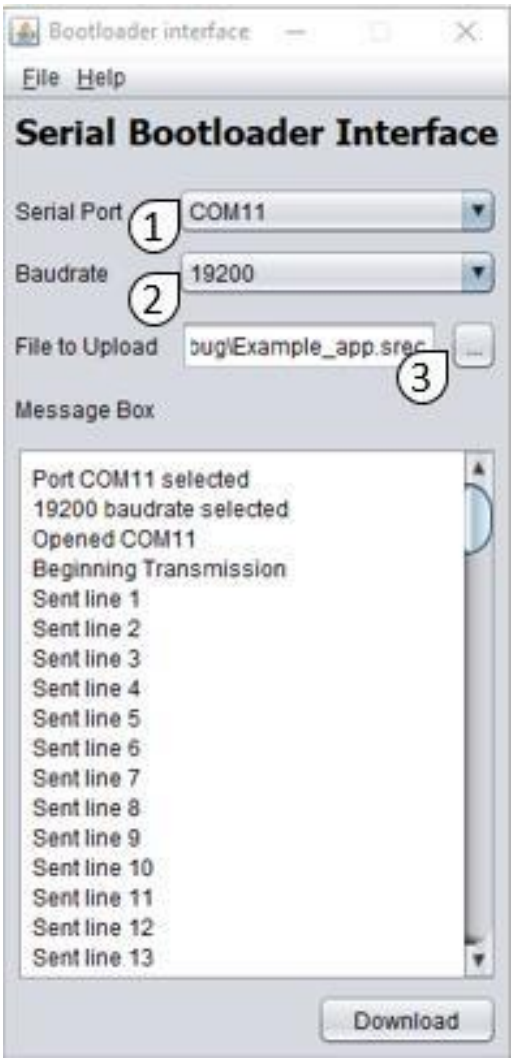


图5. UART引导加载程序接口步骤

一旦发送整个SREC文件后，java接口将关闭端口，应用程序应开始在目标板上执行。

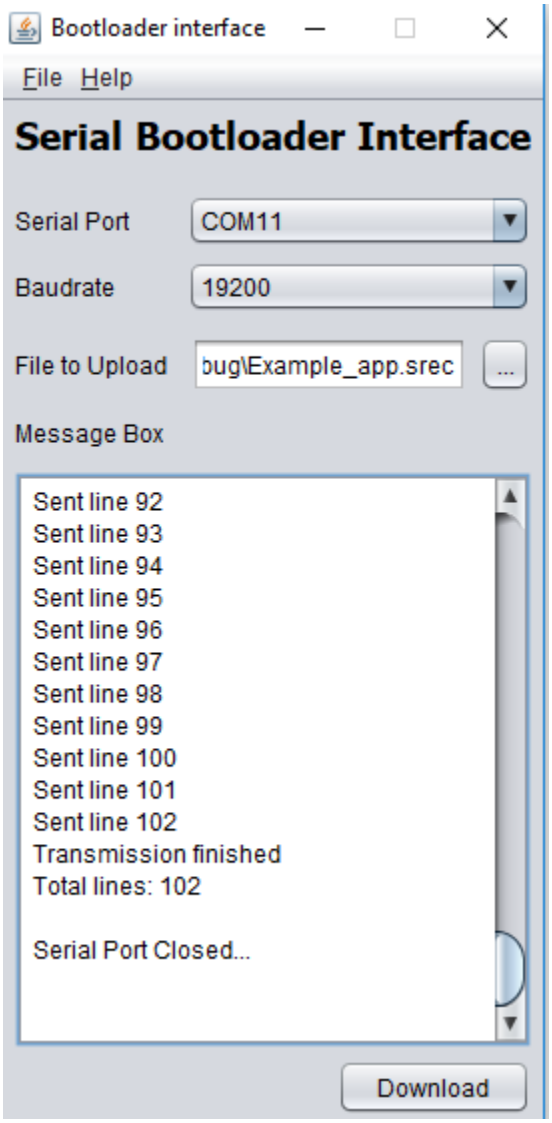
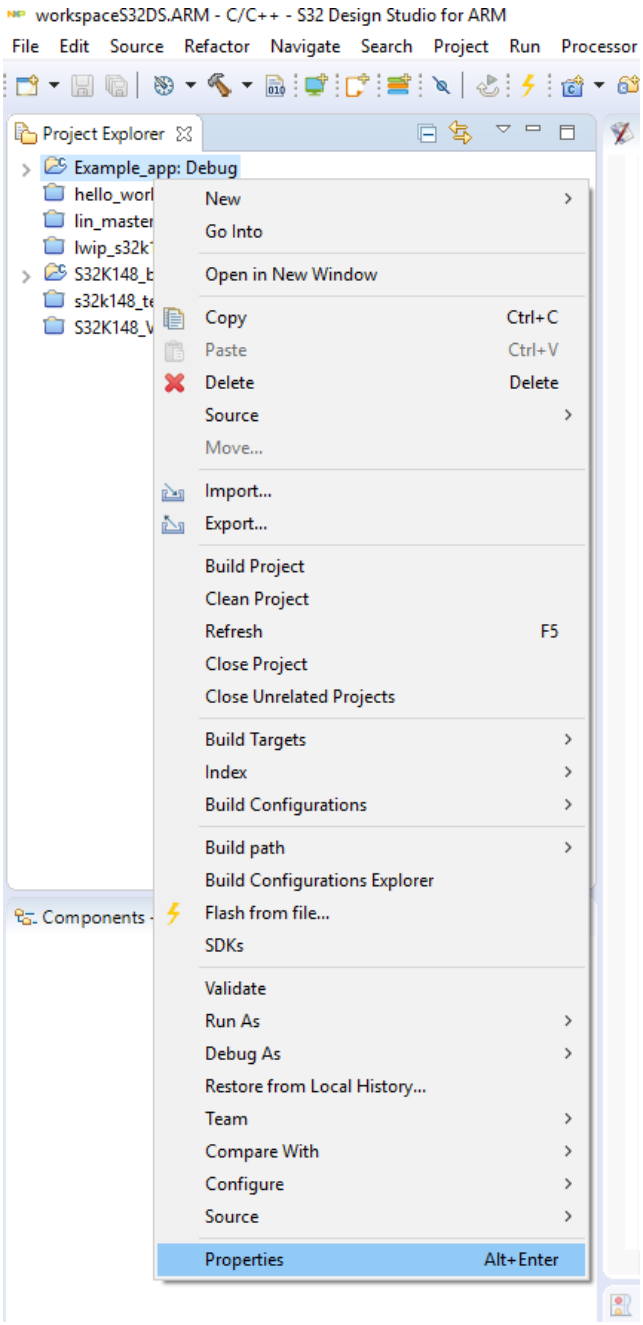
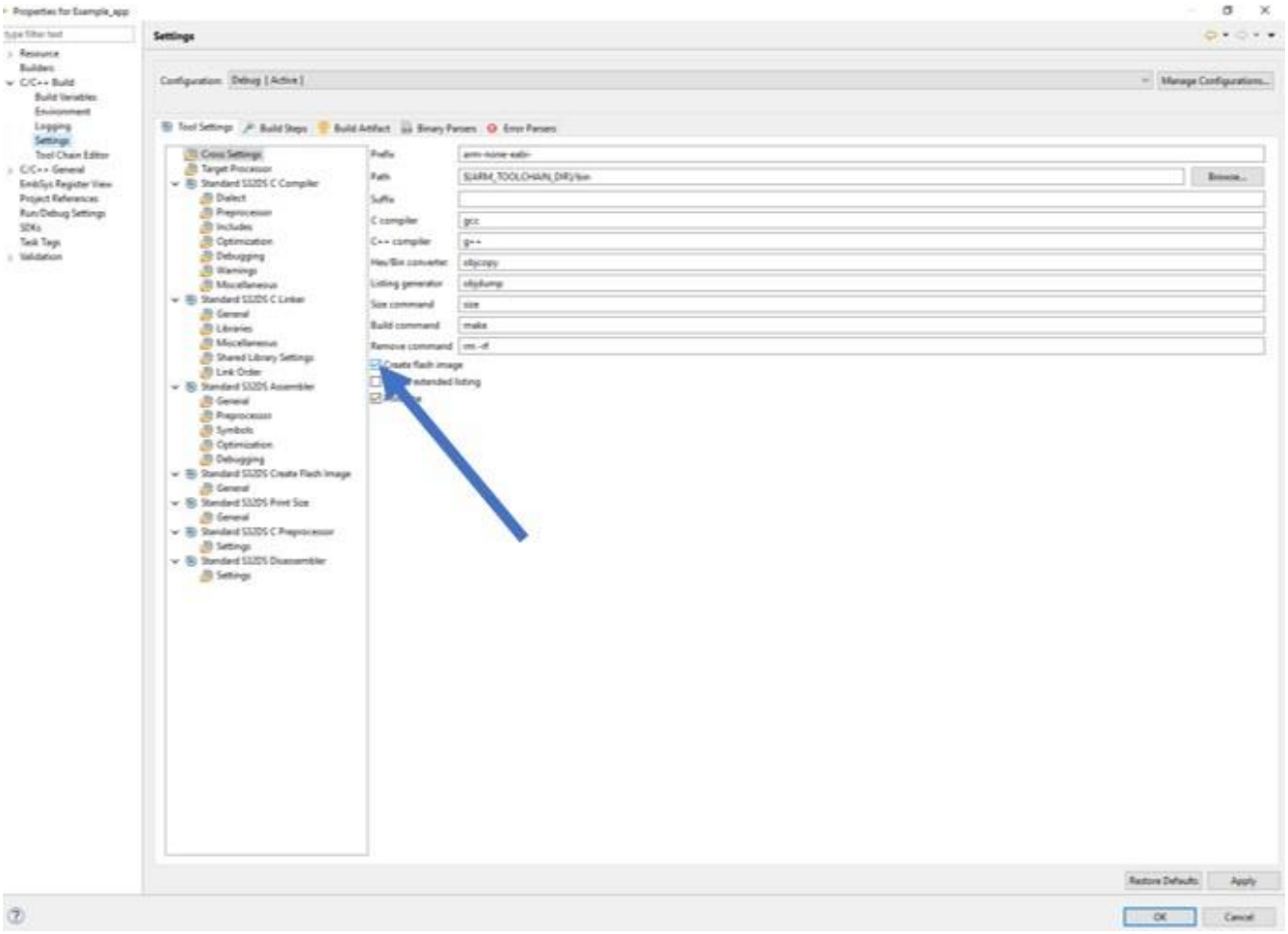


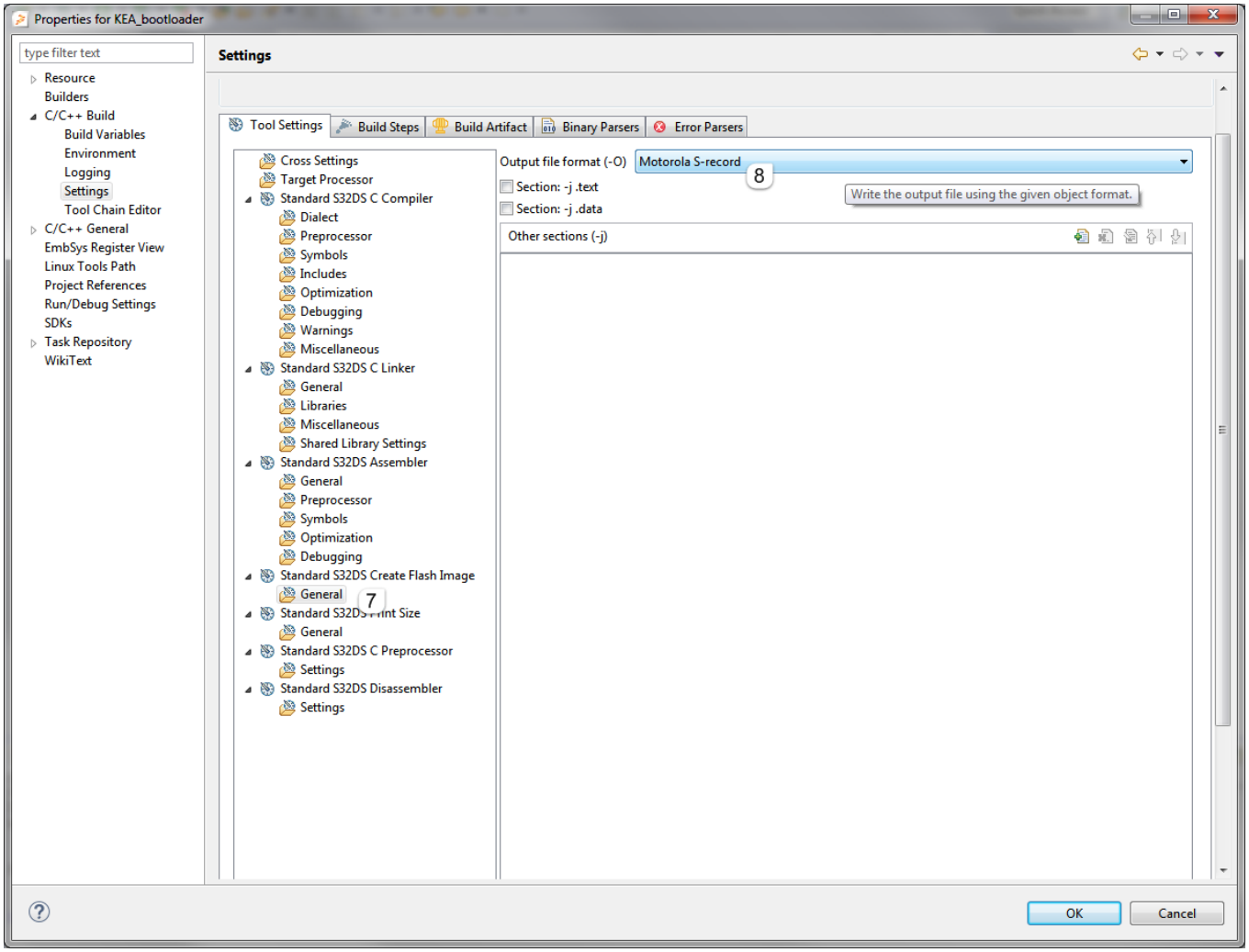
图6. UART引导加载程序接口完成

5. 附录 A

5.1. 在 S32DS上:




Properties for Example_app
Settings
Configuration: Debug [Active]
Manage Configurations...
Tool Settings
Build Steps
Build Artifact
Binary Parsers
Error Parsers
Cross Settings
Target Processor
Standard S32DS C Compiler
Dialect
Preprocessor
Includes
Optimization
Debugging
Warnings
Miscellaneous
Standard S32DS C Linker
General
Libraries
Miscellaneous
Shared Library Settings
Link Order
Standard S32DS Assembler
General
Preprocessor
Symbols
Optimization
Debugging
Standard S32DS Create Flash Image
General
Standard S32DS Print Size
General
Standard S32DS C Preprocessor
Settings
Standard S32DS Disassembler
Settings
Prefix
arm-none-eabi-
Path
${ARM_TOOLCHAIN_DIR}/bin
Browse...
Suffix
C compiler
gcc
C++ compiler
g++
Hex/Bin converter
objcopy
Listing generator
objdump
Size command
size
Build command
make
Remove command
rm -rf
Create flash image
Restore Defaults
Apply
OK
Cancel



6. 修订历史

修订号	日期	内容变化
0	07/2018	初版发布
1	10/2018	添加相关软件

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN12218
Rev. 1
10/2018