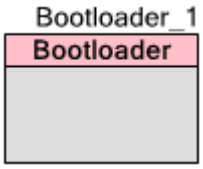
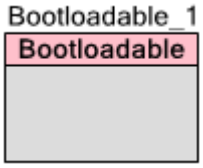
Bootloader 和 Bootloadable

1.10





特性

- 单独的 Bootloader 和 Bootloadable 组件
- 支持可配置的命令组
- 灵活的组件配置

概述

Bootloader 系统使用新应用代码和/或数据管理组件闪存的更新流程。为了使流程生效，我们使用以下组件：

- Bootloader 工程 - 包含 Bootloader 和通信组件的工程
- Bootloadable 工程- 包含用于创建代码的 Bootloadable 组件的工程

Bootloader 组件

Bootloader 组件允许用户对组件闪存进行代码更新。Bootloader 接受并执行命令，然后将这些命令的响应传回通信组件。Bootloader 收集并整理接收到的数据，并通过一个简单的命令/状态寄存接口管理对闪存的写入操作。

工程应用类型需与原理图上放置的组件匹配。例如，对于 Bootloader 工程，将应用类型设置为 Bootloader 并将 Bootloader 组件放置在原理图上。有关应用类型的信息，请参见 PSoC Creator 帮助。

通信组件

通信组件管理通信协议从外部系统接收命令，然后将这些命令传递到 Bootloader 。它还将 Bootloader 的命令响应传递回片外系统。

Bootloader 仅正式支持 USB 和 I^2C 这两种通信方法。有关相关通信方法的详情，请参见 USBFS 或 I^2C 组件数据手册。还可使用 Custom Interface（自定义界面）选项向任何现有通信组件（例如 UART，SPI 等）添加 Bootloader 支持。

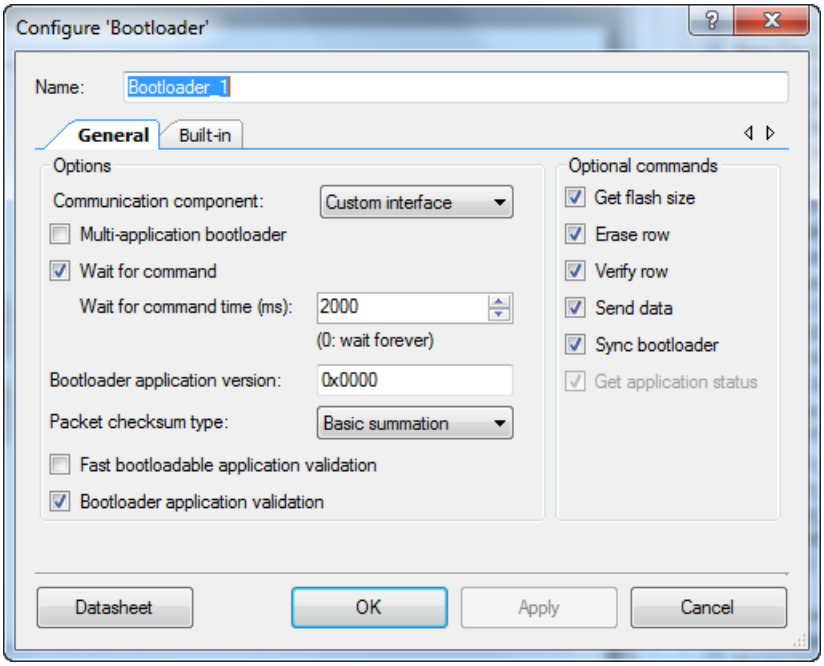
您还可以为任意的通信接口创建自己的 Bootloader 组件。有关如何执行此操作的信息和指令，请参见《组件作者指南》(Component Author Guide)。

Bootloadable 组件

使用 Bootloadable 组件时，可以为 Bootloadable 项目指定附加参数。

Bootloader 组件参数

将 Bootloader 组件拖入设计中，双击该组件，打开 **Configure**（配置）对话框。



Bootloader 组件具有以下参数：

通信组件

这是 Bootloader 用于接收命令并发送响应的通信组件。必须并且只能选择一个通信组件。此属性是原理图上具有 Bootloader 支持的可用通信协议列表。在任何情况下，无论原理图上有何内容，Custom Interface（自定义界面）选项都可用于直接实现 Bootloader 函数。

如果原理图上没有通信组件，则将选择 Custom Interface（自定义界面）选项。这可以以任何方式实现通信。

多应用 Bootloader

此选项允许闪存中驻留两个 Bootloadable 应用。这对于要保证始终有可运行的有效应用的设计非常有用。该方法有一个限制，即每个应用只可用“标准” Bootloader 工程闪存的一半。

等待命令

组件复位时，Bootloader 可等待来自 Bootloader 主机的命令或立即切换至应用代码。如果启用此选项，Bootloader 将等待来自主机的命令，直到经过 **Wait for command time**（等待命令的时间）参数指定的超时时间。如果 Bootloader 未在指定的超时间隔内接收到此命令，将在超时后执行闪存中处于活动状态的 Bootloadable。

Wait for command time（等待命令的时间）

如果 Bootloader 等待命令以便在复位之后开始加载新的 Bootloadable 应用，等待命令的时间即为启动现有 Bootloadable 应用之前等待的时间。此选项仅在**等待命令**启用后有效，否则将被忽略并变为灰色。值为零意味着永久等待。默认值为 2 秒超时。

引导加载应用程序版本

此参数提供一个代表 Bootloader 应用程序版本的 2 字节数字。默认值为 0x0000。

数据包校验类型

此参数具有两个校验类型选项，在主机和 Bootloader 之间传输数据包时使用这两个选项。默认值为 **Basic summation**（基本求和）。

所有字节（校验和除外）相加，然后取 2 的补码计算得出校验和。另一个选项是 CRC-16CCITT，即使用 CCITT 算法的 16 位 CRC(循环冗余校验)。

校验和针对完整数据包进行计算，但 Checksum（校验和）与 End of Packet（结束包）字段除外。

快速 Bootloadable 应用验证

此选项控制 Bootloader 如何验证应用数据。如果禁用此选项，Bootloader 将在每次启用选项之前计算 Bootloadable 应用的校验和。如果启用此选项，Bootloader 仅在第一次启动时计算校验和，并假定以后来每次启动时该值均有效。

Bootloader 应用程序验证

如果启用此选项，Bootloader 将通过计算校验和并将其与原数据中驻留的已保存校验和进行比较，以此进行自我验证。如果未通过验证，组件将停止。如果禁用此选项，Bootloader 即使损坏了也会执行。这可能导致不可预知的结果。

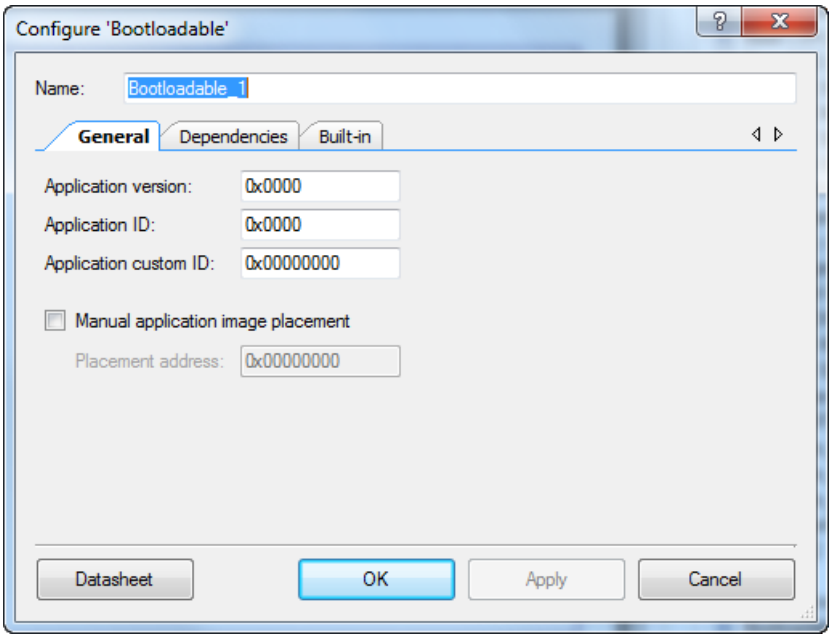
可选命令

此选项组可确定 Bootloader 是否支持相应的命令。如果启用此选项，将支持相应的命令。默认情况下支持所有可选命令。

赛普拉斯 Bootloader 主机工具需要 **Get flash size**（获取闪存大小）、**Send data**（发送数据）和 **Verify row**（验证行）命令。用户定制的 Bootloader 主机工具可能不会使用这些命令。

Bootloadable 组件参数

将 Bootloadable 组件拖入设计中，双击该组件，打开 **Configure**（配置）对话框。



Bootloadable 组件的 General（通用）选项卡中包含以下参数：

应用版本

此参数提供一个代表 Bootloadable 应用版本的 2 字节数字。默认值为 0x0000。

应用 ID

此参数提供一个代表 Bootloadable 应用 ID 的 2 字节数字。默认值为 0x0000。

应用自定义 ID

此参数提供一个代表 Bootloadable 应用中任何内容的 4 字节自定义 ID 号。默认值为 0x00000000。

手动应用映像放置

如果启用此选项，PSoC Creator 会将 Bootloadable 应用映像放置在 **Placement address**（放置地址）选项指定的位置。此外还会根据下面的 **Bootloadable** 章节概述的规则放置。

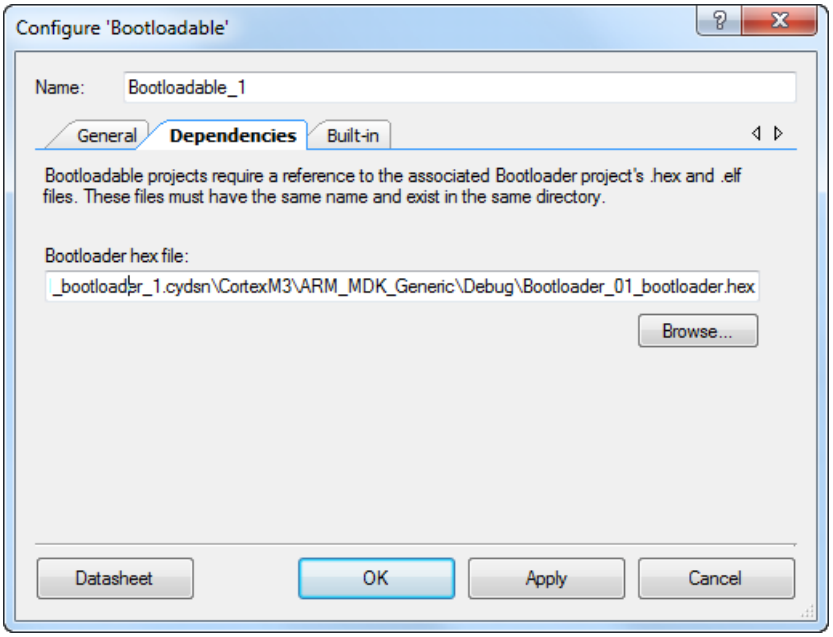
如果两个 Bootloadable 应用均参考**多应用 Bootloader** 应用，则针对每个 Bootloadable 应用独立使用此选项。

Placement Address（放置地址）

此选项允许您在储存器中指定放置 Bootloadable 应用的地址。此选项仅在 **Manual application image placement**（手动应用映像放置）选项启用时有效，否则将变为灰色。您需要将地址指定在 Bootloader 映像上方和 metadata 数据区下方。

用闪存行数乘以闪存行大小并将结果添加至闪存基地址，即可计算出放置地址。使放置地址与闪存行大小一致。有关闪存存储器组织的详细信息，请参见*系统参考指南*的*闪存和 EEPROM* 章节。

禁用 **Manual application image placement**（手动应用映像放置）选项或可由 Get Flash Size（获取闪存大小）命令进行报告时，您将从相关的 cyacd 文件获取 Bootloadable 应用可用的第一行。



Bootloadable 组件的 Dependencies（依赖关系）选项卡中包含以下参数：

Bootloader HEX 文件

此选项可用于使 Bootloadable 与 Bootloader 工程相关联。此操作是必要的，因为这样可在构建 Bootloadable 时获取有关 Bootloader 工程的信息（例如，正确计算出其在存储器中的位置）。

应用程序编程 (API)接口

应用程序编程接口 (API) 库例程允许您使用软件配置组件。下表列出了每个函数的接口并进行了说明。以下各节将更详细地介绍每个函数。

默认情况下，PSoC Creator 将实例名称“Bootloade _1”分配给指定设计中的 Bootloader 组件的第一个实例，将“Bootloadable_1”分配给 Bootloadable 组件的第一个实例。您可以将该实例重命名为符合标识符语法规则的任意唯一值。实例名称会成为每个全局函数名称、变量和常量符号的前缀。出于可读性考虑，下表中使用的实例名称为“Bootloader” 和“Bootloadable”。

Bootloader 和 Bootloadable 的函数

函数	说明
Bootloade_Start()	一旦调用，就会执行软件复位，然后Bootloader 会接管 CPU。
Bootloadable_Load()	更新 Bootloader 的 metadata 数据区以便在组件复位时启动并复位组件。

void Bootloader_Start(void)

说明： 一旦调用，就会执行软件复位，然后Bootloader 应用会接管 CPU。相关通信组件作为 Bootloader 应用初始化的一部分启动，但不会执行 Bootloadable应用代码，包括中断处理程序。

根据 Bootloader 组件的配置，应用会等待来自 Bootloader 主机的命令或切换为应用代码。

参数： None

返回值： 无。传输完成后处理器复位。

副作用： None

void Bootloadable_Load(void)

说明： 更新 Bootloader 的 metadata 数据区以便在组件复位时启动并复位组件。

参数： None

返回值： 无。处理器在执行函数时复位。

副作用： None

MISRA 合规性

本节介绍了本组件与 MISRA-C:2004 的合规和偏差情况。定义了两种类型的偏差：

- 工程偏差 - 适用于所有 PSoC Creator 组件的偏差
- 特定偏差 - 仅适用于此组件的偏差

本节提供了有关组件特定偏差的信息。*系统参考指南*的“MISRA 合规性”章节中介绍工程偏差以及有关 MISRA 合规性验证环境的信息。

Bootloader / Bootloadable 组件尚未进行 MISRA-C:2004 编码指南合规性的验证。

固件源代码示例

PSoC Creator 在“查找示例工程”对话框中提供了很多包括原理图和代码示例的示例工程。要获取组件特定的示例，请打开组件目录中的对话框或原理图中的组件实例。要获取通用的示例，请打开 Start Page（开始页）或 **File**（文件）菜单中的对话框。根据需要，使用对话框中的 **Filter Options**（筛选选项）可缩小可选工程的列表。

有关更多信息，请参见 PSoC Creator 帮助中的“Find Example Project（查找示例工程）”主题。

功能描述

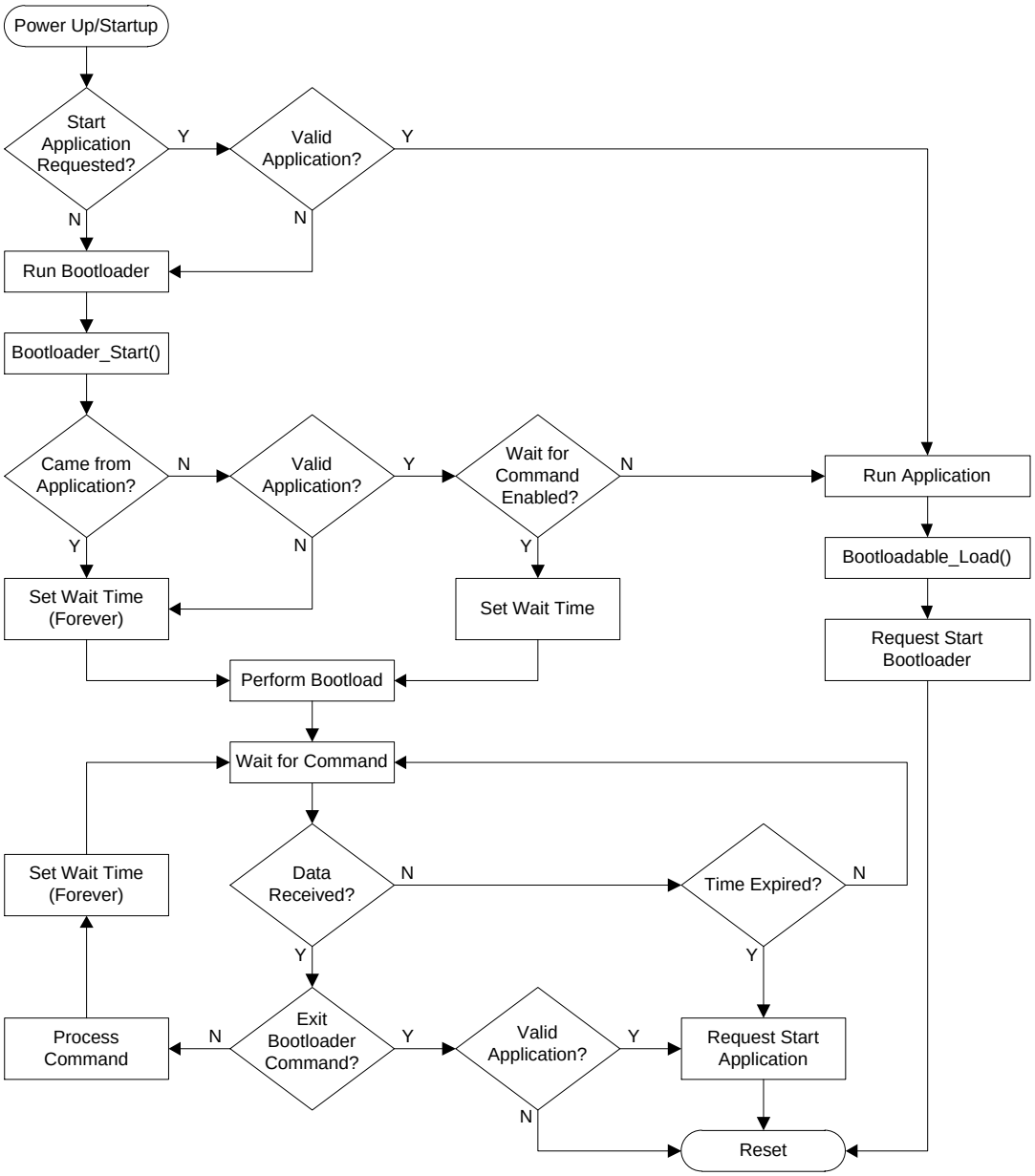
Bootloader 和 Bootloadable 的函数

Bootloader 工程通过 Bootloader 工程的通信组件可将 Bootloadable 工程（或新代码）整体烧写到闪存中。传输后，会始终对处理器进行复位。Bootloader 工程还负责在复位时测试特定条件，并有可能在 Bootloadable 工程不存在或损坏的情况下自动启动传输操作。

启动时，Bootloader 代码加载其自己配置的配置字节。此外还必须初始化堆栈和其他资源以及外设以便进行传输。传输完成后，控件通过软件复位传递到 Bootloadable 中。

然后， Bootloadable 加载其自己配置的配置字节，并对堆栈及其函数的其他资源和外设重新进行初始化。 Bootloadable 可调用其中的 Bootloadable_Load() 函数，以切换为引导加载应用程序（这会再次造成软件复位）。

下图显示了 Bootloader 的工作原理。



Bootloader 应用程序

通常，您可以通过将 Bootloader 组件和通信组件拖到原理图上、将 I/O 路由至引脚、设置时钟等操作来完成 Bootloader 设计工程。包含 Bootloader 组件和通信组件的工程实现基本引导加载应用程序函数的新代码接收并将其写入闪存。您可以通过将其他组件拖到原理图上或添加源代码来向基本的 Bootloader 工程添加自定义函数。

Bootloadable 应用

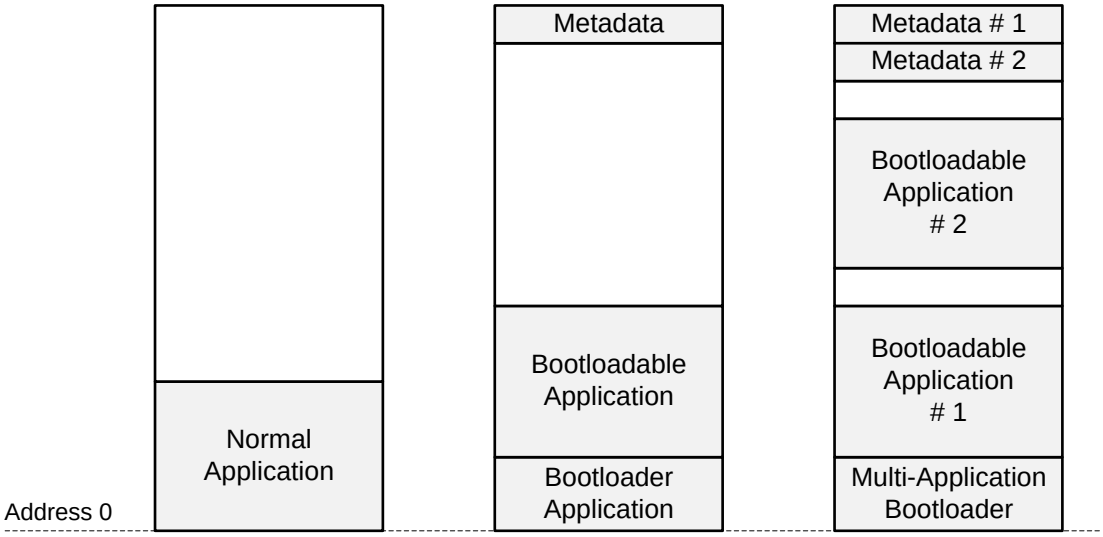
Bootloadable 应用实际上就是代码。它与常规应用类型非常相似。主要区别是，Bootloadable 应用始终与引导加载应用相关联，而常规工程从不会与引导加载应用关联。

存储器占用情况

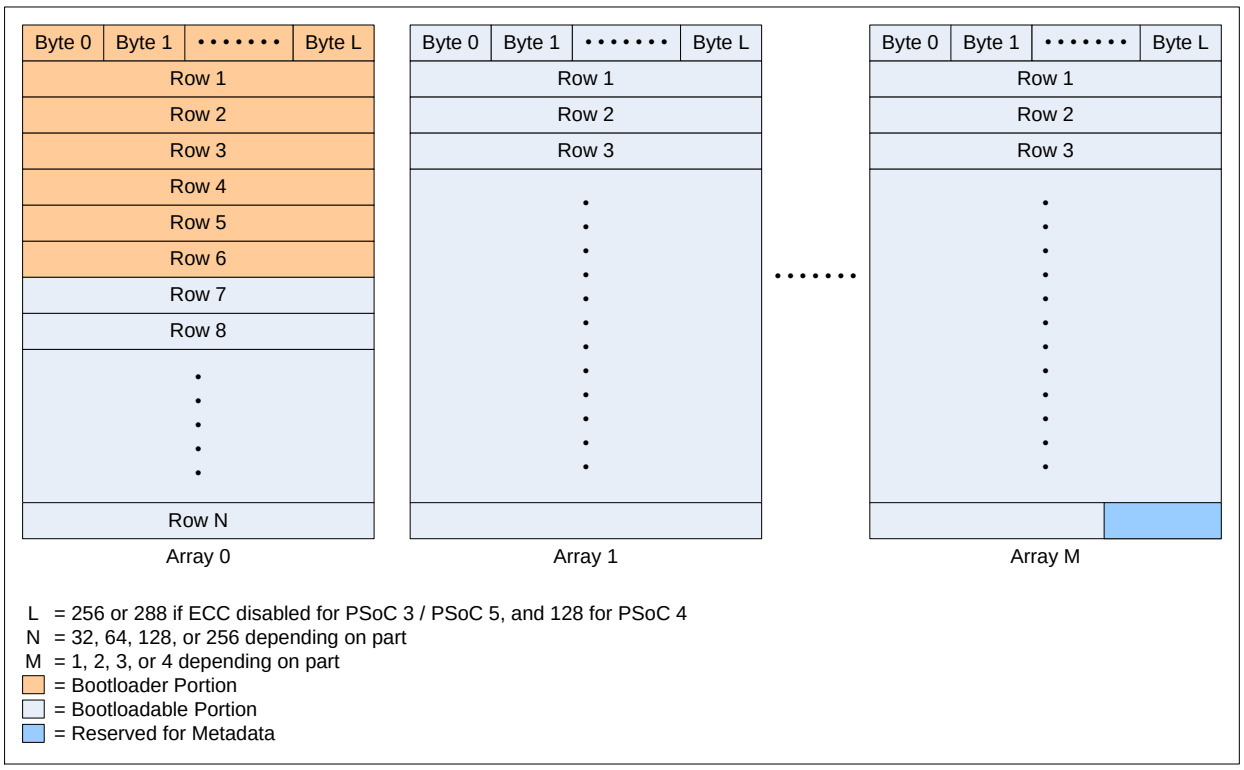
常规应用和引导加载应用驻留在以 0 地址开始的闪存中。Bootloadable 应用占据的闪存从下一空白闪存行开始到引导加载应用结束。对于多应用 Bootloader，第一个 Bootloadable 应用驻留在 Bootloader 应用之上。第二个 Bootloadable 应用占据的闪存以第一个 Bootloadable 应用开头和闪存末尾之间的中间行开始。

如果在 Bootloadable 组件自定义程序中启用了 **Manual application image placement**（手动应用映像放置）选项， Bootloadable 应用将放置于 **Placement address**（放置地址）选项指定的地址。

下图显示了（从左到右）常规应用、引导加载应用和 Bootloadable 应用，以及多应用 Bootloader 和两个 Bootloadable 应用的存储器使用情况：



下图显示了组件的闪存布局。



Bootloader 工程始终占据底部的 X 个闪存行。将 X 设置为使以下内容具有足够的闪存：

- 此工程的矢量表，以地址 0 开始（PSoC 3 除外），
- Bootloader 工程配置字节，
- Bootloader 工程代码和数据，
- Bootloader 闪存部分的校验和。

Bootloader 工程的配置字节始终存储在主闪存中，从来不会在 ECC 闪存中。相关选项会从 Bootloader 工程设计范围资源文件中删除。

闪存的引导加载应用部分将保护在设计范围资源文件的 Flash Protection（闪存保护）选项卡中，只有通过 JTAG/SWD 进行下载才能将其覆盖。

Bootloadable 紧随 Bootloader，占据从首个闪存行大小边界开始的闪存，包含以下内容：

- 工程的矢量表（PSoC 3 除外），
- Bootloadable 项目的代码和数据，

- 64 字节数据保留在最后一个闪存阵列的末尾，以存储由 Bootloader 和 Bootloadable 的工程共同使用的 metadata 数据。

Bootloadable 的配置字节和标准工程中配置字节的存储方式相同，即存储在主闪存或 ECC 闪存中，具体取决于设计范围资源文件中的设置。

PSoC 3 详细信息

在 PSoC 3 中，唯一“异常矢量”是地址 0 处的 3 字节指令，在处理器复位时执行。（中断矢量不在闪存中，而是由中断控制器 [IC] 提供）。因此，复位时，PSoC 3 Bootloader 代码仅从闪存地址 0 开始执行。

PSoC 5 和 PSoC 4 详细信息

在 PSoC 5 / PSoC 4 组件中，地址 0 处必须存在异常矢量表。（该表位于矢量表偏移寄存器指向的地址 0xE000ED08，其值在复位时设置为 0。） Bootloader 代码紧跟着该表之后。

该表包含 Bootloader 工程的初始堆栈指针 (SP) 值以及 Bootloader 工程代码的起始地址。它还包含用于 Bootloader 的异常矢量和中断矢量。

Bootloadable 还有其自己的矢量表，其中包含工程的起始 SP 值和第一个指令地址。传输完成后，作为将控件传递到 Bootloadable 的一部分，矢量表偏移寄存器中的值更改为 Bootloadable 表的地址。

Metadata 数据

Metadata 数据部分是一个 64 位模块，可用作引导加载应用程序和 Bootloadable 应用的公共区域。在引导加载应用程序中，metadata 数据放置在第 N-1 行，对于多应用 Bootloader， Bootloadable 应用的数字 1 使用第 N-1 行，应用数字 2 使用第 N-2 行来存储 metadata 数据，其中 N 是选定组件的总行数。该模块存储了各种参数，包括：

- 引导加载应用程序版本
- Bootloadable 应用 ID
- Bootloadable 应用版本
- Bootloadable 自定义 ID

PSoC Creator 工程输出文件

在创建 Bootloader 工程或 Bootloadable 时，都会为该工程创建输出文件。

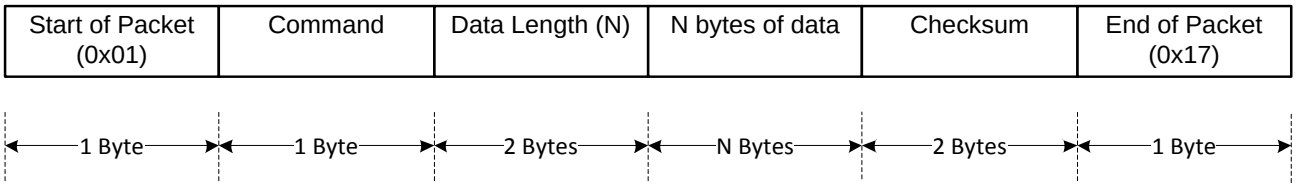
此外，在构建 Bootloadable 时，还会为这两种工程创建一个输出文件，即“组合”文件。该文件同时包含 Bootloader 工程和 Bootloadable。它通常用于方便在生产环境中将两种工程下载（通过 JTAG/SWD）到组件闪存中。

Bootloadable 的配置字节存储在主闪存或 ECC 闪存中。 Bootloadable 输出文件的格式有以下特点：当组件具有已被禁用的 ECC 字节时，执行传输操作的时间将减少。该操作通过交错 Bootloadable 的主闪存地址空间中的记录以及 ECC 闪存地址空间中的记录来完成。Bootloader 通过对关联的闪存行进行一次编程来利用此交错的结构，该闪存行同时包含主闪存和 ECC 闪存的字节。

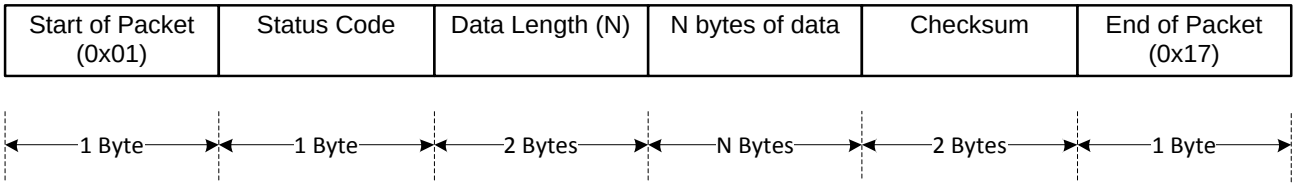
每个工程都有其自己的校验和。在工程构建期间，校验和包含在输出文件中。

Bootloader 数据包结构

从主机发送至 Bootloader 的通信数据包的结构如下：



从移动加载程序读取的响应包的结构如下：



状态/错误代码

Bootloader 中可能的状态/错误代码输出包括：

状态/错误代码	值	说明
CYRET_SUCCESS	0x00	成功接收并执行命令
BOOTLOADER_ERR_VERIFY	0x02	闪存验证失败
BOOTLOADER_ERR_LENGTH	0x03	可用数据量超出预期范围
BOOTLOADER_ERR_DATA	0x04	数据形式错误
BOOTLOADER_ERR_CMD	0x05	无法识别命令
BOOTLOADER_ERR_DEVICE	0x06	预期组件与检测到的组件不匹配。

状态/错误代码	值	说明
BOOTLOADER_ERR_VERSION	0x07	不支持检测到的 Bootloader 版本。
BOOTLOADER_ERR_CHECKSUM	0x08	数据包校验和与预期值不匹配
BOOTLOADER_ERR_ARRAY	0x09	闪存阵列 ID 无效
BOOTLOADER_ERR_ROW	0x0A	闪存行数无效
BOOTLOADER_ERR_APP	0x0C	应用无效，不能设置为活动状态
BOOTLOADER_ERR_ACTIVE	0x0D	应用当前标记为活动
BOOTLOADER_ERR_UNK	0x0F	发生未知错误

Bootloader 命令

Bootloader 支持这些命令。对于所有接收到的字节，如果没有以一组命令字节之一开始，则该字节会被丢弃，且不会生成响应。所有多字节字段都是输出最低有效位优先。

注意：Bootloader 执行任何命令所需的时间是以组件的设置为基础的。一些因素会影响时序，包括：

- 部件运行的时钟速度
- 用于构建工程的工具链
- 构建过程中使用的优化设置
- 后台运行的中断数

Bootloader 命令名称（命令代码）			
数据字节（字节数）	响应包状态代码	响应包数据（字节数）	说明
进入 Bootloader (0x38)			
无	成功 错误命令 错误数据 错误长度 错误校验和	芯片 ID (4) 芯片修订版 (1) 版本 (3)	Bootloader 通过组件信息和 Bootloader 组件版本响应此命令。 版本指 Bootloader 组件的版本。
获取闪存大小 (0x32)（可选）			

Bootloader 命令名称（命令代码）			
数据字节（字节数）	**响应包状态代码**	**响应包数据（字节数）**	**说明**
闪存阵列 (1)	成功 错误命令 错误数据 错误长度 错误校验和	第一个可用行 (2) 最后一个可用行 (2)	Bootloader 对该命令的响应包含引导加载应用程序之后的第一个整行（即可 Bootloader 应用的第一行）和选定闪存阵列中的最后一个闪存行。
编程行 (0x39)			
闪存阵列 (1) 闪存行数 (2) 写入数据 (n)	成功 错误命令 错误数据 错误长度 错误校验和 错误闪存行 错误活动	无	将一行闪存数据写入组件。 使用 Send Data（发送数据）命令可将写入闪存的数据发送至多个数据包。 此命令可随最后一个数据模块发送以便对此行进行编程。
擦除行 (0x34)（可选）			
闪存阵列 (1) 闪存行数 (2)	成功 错误命令 错误数据 错误长度 错误校验和 错误闪存行 错误活动	无	擦除所提供的闪存行的内容。
验证行 (0x3A)（可选）			
闪存阵列 (1) 闪存行数 (2)	成功 错误命令 错误数据 错误长度 错误校验和	行验证和 (1)	获取所提供的闪存行内容的 1 字节校验和。
验证校验和 (0x31)			

Bootloader 命令名称（命令代码）			
数据字节（字节数）	**响应包状态代码**	**响应包数据（字节数）**	**说明**
无	成功 错误命令 错误数据 错误长度 错误校验和	有效校验和 (1)	返回值非 0 表示应用代码闪存校验和与闪存中存储的预期值匹配，因此应用有效。 返回值 0 表示校验和不匹配，因此应用无效。
发送数据 (0x37)（可选）			
组件数据 (n)	成功 错误命令 错误数据 错误长度 错误校验和	无	将一个数据块发送到组件。 此数据将进行缓冲，等待另一个命令，该命令会告知 Bootloader 如何处理此数据。如果连续发出多个“发送数据”命令，则此数据将附在上一模块后。 此命令用于将大量的传输数据分为适当大小，以防止某些协议中出现总线枯竭的情况。
同步 Bootloader (0x35)（可选）			
无	无	无	将 Bootloader 复位至空白状态，准备接受新命令。 所有缓冲的数据都将被删除。只有主机和客户端之间不再同步时才需要使用此命令。
退出 Bootloader (0x3B)			
无	无	无	通过触发组件的软件复位退出 Bootloader 。 执行软件复位之前，验证 Bootloadable的应用。如果应用通过了验证，将在软件复位之后执行此应用。如果应用未通过验证，则将在软件复位之后再次用 Bootloader 执行此应用。
获取应用状态（仅限多应用 Bootloader ）(0x33)（可选）			
应用 # (1)	成功 错误长度 错误校验和 错误数据	应用 # 有效 (1) 应用 # 活动 (1)	返回指定应用状态。

Bootloader 命令名称（命令代码）			
数据字节（字节数）	**响应包状态代码**	**响应包数据（字节数）**	**说明**
设置活动应用（仅限多应用 Bootloader）(0x36)			
应用 # (1)	成功 错误应用 错误长度 错误数据 错误校验和	无	指定的 Bootloadable 应用设置为活动状态。此命令用于在两个 Bootloadable 应用之间进行切换。

引导加载应用程序和代码数据文件格式

引导加载应用和代码数据 (.cyacd) 文件格式可存储设计中 Bootloadable 的部分。该文件是后面若干行闪存数据的数据头。除标题以外，.cyacd 文件中的每一行均表示闪存数据的一整行。数据以 Big Endian 的格式存储为 ASCII 数据。

标题记录格式为：

[4 字节芯片 ID][1 字节芯片版本][1 字节校验类型]

数据记录格式为：

[1 字节阵列 ID][2 字节行数][2 字节数据长度][N 字节数据][1 字节校验和]

标题中的校验类型指示用于在 Bootloader 主机和 Bootloader 自身之间发送的数据包的校验类型。数据记录中的校验和是基本求和，通过对所有字节求和（不包括校验和自身），然后取其二进制补码计算得到的。

Boot 工具

PSoC Creator 附带 Boot 工具 (bootloader_host.exe)，可用于对在 PSoC 芯片上运行的 Bootloader 进行测试。 Boot 工具是一个应用程序，它与 Bootloader 进行通信，以发送 Bootloadable 的新映像。提供的 Boot 工具只是一种开发和测试工具。

源代码

除本身可执行的主机外，还提供了所使用的大部分源代码。此源代码可用于创建您自己的引导加载主机应用程序。源代码位于下列目录中：

<Install Dir>\cybootloaderutils

默认情况下，此目录为：

C:\Program Files\Cypress\PSoC Creator\<Release Version>\PSoC Creator\cybootloaderutils

该源代码分为四个不同的模块。这些模块用于实现引导加载主机所需的各种功能。根据所需的控制级别，部分或所有这些模块可用于开发自定义引导加载主机应用程序。

cybtldr_command.c/h

该模块用于构建发送至 Bootloader 的数据包，并解析从 Bootloader 接收到的数据包。该模块具有构建 Bootloader 能够理解的各类数据包的独立函数，并具有解析 Bootloader 返回结果的独立函数。

cybtldr_parse.c/h

该模块用于解析 *.cyacd 文件，该文件包含要发送至组件的 Bootloadable 映像。该模块具有用于设置文件访问、读取标题、读取行数据和关闭文件的函数。

cybtldr_api.c/h

该模块是行级 API，允许每次向使用所提供的通信机制的 Bootloader 发送单行数据。该模块具有用于设置引导加载操作、编程行、擦除行、验证行和结束引导加载操作的函数。

cybtldr_api2.c/h

此模块是更高级别的 API，处理整个引导加载过程。该模块具有用于编程组件、擦除组件、验证组件和终止当前操作的函数。

资源

Bootloader 和 Bootloadable 使用以下组件资源：

- Bootloader 组件使用复位状态 (RESET_SR0) 寄存器的通用位。在软件复位边界进行 Bootloader 通信时需要这些位。
- 通信组件使用的资源在对应的组件数据手册中。

API 存储器使用

根据编译器、组件、所用 API 数量和组件配置的不同，组件内存使用会出现较大变化。下表提供指定组件配置中可用的 API 的存储器使用。

已利用释放模式中配置的相关编译器进行了测量，大小采用了优化设定。有关特定的设计，可分析编译器生成的映射文件以确定内存使用情况。

配置	PSoC 3 (Keil_PK51)		PSoC 4 (GCC)		PSoC 5LP (GCC)	
	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节	闪存 字节	SRAM 字节
Bootloader	1768	294	1090	8	1032	296
Bootloader （完整应用）[1]	5841	1560	4280	408	4384	580
Bootloadable（完整应用）[2]	1747	94	960	232	1544	276

注意：

1. 此配置的测量是针对整个 Bootloader 工程进行的，并将基于固定功能的 I²C 用作通信组件，对 Bootloader 组件进行配置以实现最低闪存消耗。
2. 此配置的测量是针对整个 Bootloader 工程进行的。

组件更改

本节介绍组件与以前版本相比的主要更改。

版本	更改说明	更改/影响原因
1.10	添加了 MISRA 合规性章节。	
	添加了 PSoC 4 组件支持。	新组件支持
	次要数据手册编辑	
1.0.a	数据手册纠正	
1.0	初始组件版本	

赛普拉斯半导体公司，2013-2016 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性质的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion徽标，及上述项目的组合，及PSoC、CapSense、EZ-USB、F-RAM和Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。