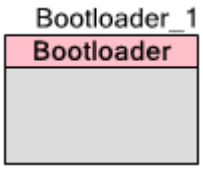
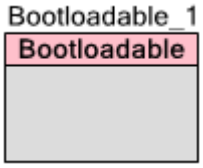
Bootloader 和 Bootloadable

1.20





性能

- 独立的 Bootloader 及 Bootloadable 组件
- 支持指令的可配置集
- 灵活的组件配置

概述

Bootloader 系统管理使用新应用代码和/或数据来更新器件闪存存储器。为了使流程生效，我们使用以下组件：

- Bootloader 项目 — 含有 Bootloader 组件和通信组件
- Bootloadable 项目 — 含有 Bootloadable 组件，用于创建代码

Bootloader 组件

通过 Bootloader 组件您可以使用新代码更新器件闪存存储器。Bootloader 接收并执行相应指令，然后将这些指令的响应回送给通信组件。Bootloader 收集并整理接收到的数据，并通过一个简单的指令/状态寄存器接口对闪存的实际写入操作进行管理。

项目的应用类型需要与原理图上的组件相匹配。例如，对于 Bootloader 项目，在 Build Settings 项下将 **Application Type**（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，请参考“PSoC Creator 帮助”部分的内容。

通信组件

通过管理通信协议，通信组件可以接收来自外部系统的指令，然后将这些指令传递 Bootloader。它还将 Bootloader 的命令响应传递回片外系统。

只有 USB 和 I^2C 是受 Bootloader 官方支持的两种通信方法。有关通信方法的详情，请参见 USBFS 或 I^2C 组件数据手册。还可使用 Custom Interface（自定义接口）选项向任何现有通信组件添加 Bootloader 支持。

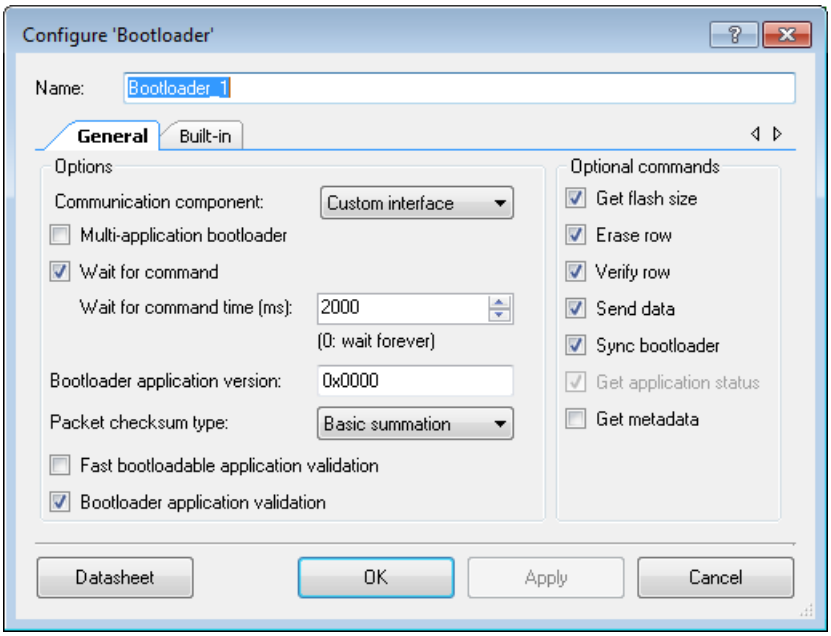
您还可以创建自己的 Bootloader 通信组件，其中支持任意数量的通信方法。有关如何执行此操作的信息和指令，请参考《组件创建指南》（Component Author Guide）中介绍的信息。

Bootloadable 组件

通过 Bootloadable 组件可以指定使用于 Bootloadable 项目的附加参数。

Bootloader 组件参数

将 Bootloader 组件拖入到您的设计窗口上，然后双击它将打开 **Configure** 对话框。



Bootloader 组件具有以下参数：

Communication Component（通信组件）

通过使用通信组件，Bootloader 可以接收指令并传递其响应。只能选择唯一的通信组件。该属性是原理图上受 Bootloader 支持的可用通信协议列表。在任何情况下，无论原理图上的内容如何，都有一个“Custom interface”（自定义接口）选项用于直接实现 Bootloader 函数。

如果原理图上没有通信组件，则将选择 Custom Interface（自定义接口）选项。这样能够以任何方式实现通信。

Multi-application bootloader（多应用的 Bootloader）

该选项允许闪存中有两个 Bootloadable 应用。它对于要保证始终有一个可运行的有效应用的设计非常有用。此保证有一个限制，即每个应用只能使用“标准 Bootloader”项目可用闪存空间的一半。

Wait for command（等待指令）

在器件复位时，Bootloader 可以等待来自 Bootloader 主机的指令，或者立即跳转到应用代码。如果此选项有效，Bootloader 等待来自主机的指令，直到由 **Wait for command time** （等待指令时间）参数所指定的超时时长结束为止。如果超过了所设置的时间，Bootloader 还没有收到主机的指令，这时将执行闪存中的活动 Bootloadable 项目。

Wait for command time（等待指令的时间）

器件复位后，Bootloader 要等待一个指令后才能开始加载新的 Bootloadable 应用。等待指令的时间就是启动现有 Bootloadable 应用之前所等待的时间。当选中了 **Wait for command** 参数时，该选项才可用；否则它将被忽略并显示为灰色（无效状态）。如果该项的值等于“0”，则表示永久等待。默认值为 2 秒超时。

Bootloader application version（Bootloader 应用版本）

该参数中的 2 字节数字表示 Bootloader 的版本。其默认值为 0x0000。

Packet checksum type（数据包校验和类型）

该参数提供了两个校验和类型选项，用于在主机和 Bootloader 之间传输数据包。默认设置为 **Basic summation**（基本求和）。

校验和的基本求和方式是：将所有字节（不包含校验和值）加起来，然后进行计算二进制补码。另一个选项是 CRC-16CCITT（即 16 位 CRC 使用 CCITT 算法）。

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。

Fast Bootloadable application validation（ Bootloadable 应用的快速验证）

通过该选项可以控制 Bootloader 对应用数据的验证情况。如果禁止该选项，每次进行加载前 Bootloader 将计算 Bootloadable 应用的校验和。如果使能该选项，Bootloader 只计算第一次的校验和，并在以后每次启动时都将该值作为有效值。

Bootloader application validation（Bootloader 应用的验证）

如果该选项有效，Bootloader 应用的验证方法是：先计算校验和的值，然后将该值与存储在元数据区内的值进行比较。如果验证操作失败，则器件将被停止。如果禁用此选项，Bootloader 即使损坏了也会执行。这样可能会导致产生不可预测的结果。

Optional Commands（可选的指令）

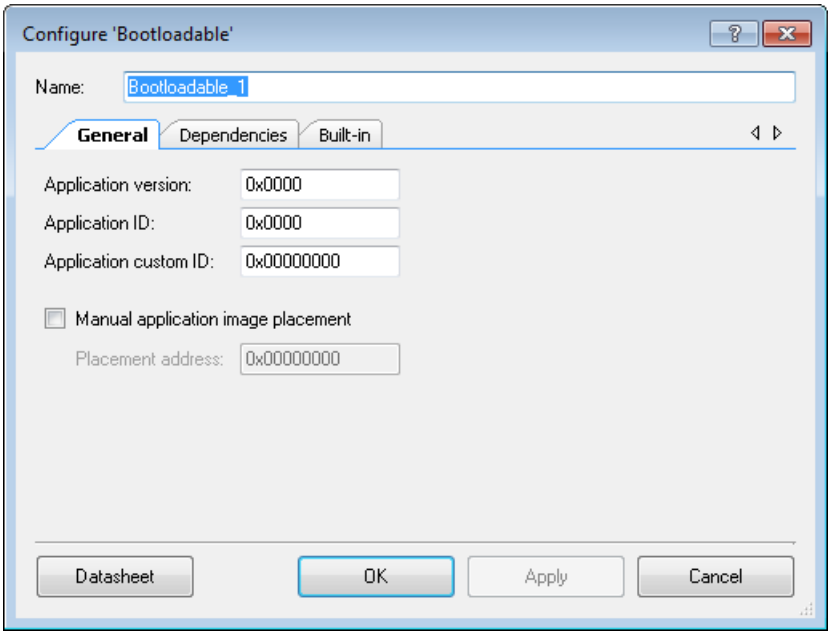
该选项组可以确定相应的指令是否受 Bootloader 支持。如果该选项有效，则支持相应的指令。默认情况下，所有可选指令均得到支持。

赛普拉斯 Bootloader 主机（Cypress Bootloader Host）工具需要使用 **Get flash size**、**Send data** 和 **Verify row** 指令。但在自定义 Bootloader 主机工具中可能不使用这些指令。

Bootloadable 组件参数

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

“General”选项卡



Bootloadable 组件的 General 选项卡下包括下列各参数：

Application version（应用版本）

该参数中的 2 字节数字表示的是 Bootloadable 应用的版本。其默认值为 0x0000。

Application ID（应用 ID）

该参数中的 2 字节数字表示的是 Bootloadable 应用的 ID。默认值为 0x0000。

Application custom ID（应用自定义 ID）

该参数中的 4 字节数字是自定义 ID 号码，表示 Bootloadable 应用的任何值。默认值为 0x00000000。

Manual application image placement（手动应用映像放置）

如果使能该选项，PSoC Creator 将把 Bootloadable 应用映像保存在 **Placement address** 选项所指定的位置内。也可以根据下面 **Bootloadable Project** 部分所介绍的规则放置 Bootloadable 应用映像。

如果两个 Bootloadable 应用都指向了 **Multiapplication bootloader** 应用，则这两个应用均可以独立使用该选项。

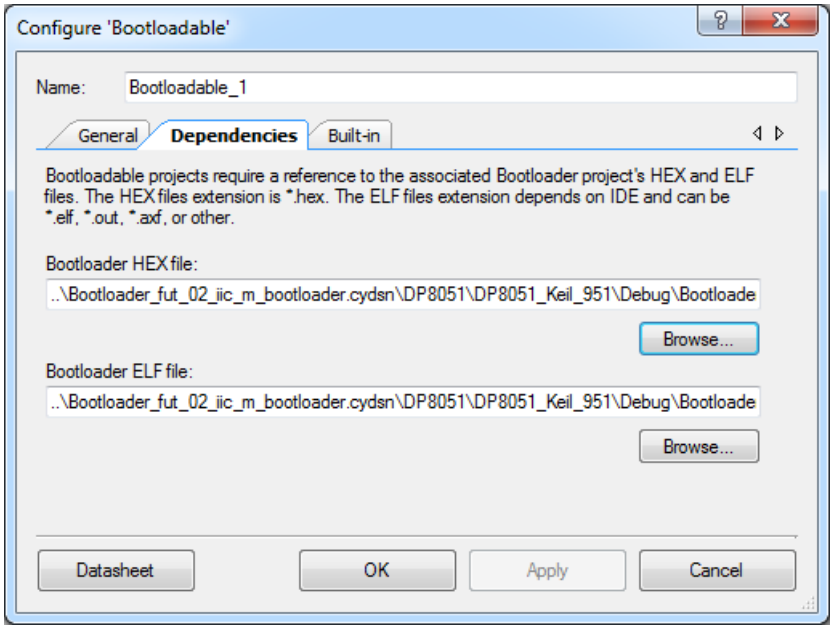
Placement Address（放置地址）

通过该项，用户可指定Bootloadable应用在存储器中的地址。只有使能了**Manual application image placement**项时，该项才有效；否则，该项将显示为灰色。另外需要指定位于Bootloader映像以上到元数据区以下的地址。

将闪存行的数量（从放置映像的闪存行开始）乘以闪存行的大小，然后将得到的结果和闪存基本地址相加，这样可以计算出定位地址。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** 选项后，可以从相应的 cyacd 文件中查找 Bootloadable 应用的第一个有效行，或者通过执行 Get Flash Size 指令找到该行。

Dependencies 选项卡



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

Bootloader HEX file（Bootloader 的十六进制文件）

通过该项，您可以将某个 Bootloadable 项目同相应的 Bootloader 项目的 HEX 文件相连接。从而，Bootloadable 项目的编译能够包含有关 Bootloader 项目的信息（例如，可以正确计算出 Bootloader 项目在存储器中的位置）。

Bootloader ELF file（Bootloader 的 ELF 文件）

通过该项，您可以将某个 Bootloadable 项目同相应的 Bootloader 项目的 ELF 文件相连接。ELF 文件的扩展名取决于 IDE。例如，PSoC Creator 所生成的 ELF 文件具有“*.elf”扩展名，而其它 IDE 生成的文件的扩展名则为“*.elf”、“*.out”或“*.axf”。

如果“*.elf”文件所在的文件夹与特定 HEX 文件的相同，该选项便自动输入该文件的路径。您总是可以更新该选项并手动输入 ELF 文件的路径。

注意：必须在同一个编译过程中生成 HEX 和 ELF 文件，以确保它们的一致性。

应用编程接口

通过应用编程接口（API），您可以使用软件进行组件配置。该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，在给定的设计窗口中， PSoC Creator 将 Bootloader 组件的第一个实例命名为“Bootloader_1”，并将 Bootloadable 组件的第一个实例命名为“Bootloadable_1”。您可以将该实例重新命名为符合标识符语法规则的唯一一个任意值。该实例名称将成为各全局函数名称、变量和常数符号的前缀。为了提高可读性，下表中的实例名称分别为“Bootloader”和“Bootloadable”。

Bootloader 和 Bootloadable 的函数

函数	说明
Bootloade_Start()	调用该函数后，将立即执行软件复位，然后Bootloader应用将控制CPU。
Bootloadable_Load()	更新元数据区以便在进行器件复位时启动Bootloader，并复位器件。

void Bootloader_Start(void)

说明： 调用该函数后，将立即执行软件复位，然后Bootloader应用将控制CPU。在初始化Bootloader应用时，会启动相应的通信组件。包含各中断处理程序的Bootloadable应用代码不被执行。

根据Bootloader组件的配置，应用将等待Bootloader主机中的指令或跳转到应用代码。

参数： 无

返回值： 无。传输完成后，将复位处理器。

其他影响： 无

void Bootloadable_Load(void)

说明： 更新元数据区以便在进行器件复位时启动Bootloader，并复位器件。

参数： 无

返回值： 无。执行函数时，将复位处理器。

其他影响： 无

MISRA 合规性

本节介绍了 MISRA-C:2004 合规性和本组件的偏差情况。定义了两种类型的偏差：

- 项目偏差 — 适用于所有 PSoC Creator 组件的偏差
- 特定偏差 — 仅适用于该组件的偏差

本节介绍了有关组件特定偏差的信息。*《系统参考指南》*的“MISRA 合规性”章节中介绍项目偏差以及有关 MISRA 合规性验证环境的信息。

Bootloader 组件的特定偏差：

MISRA-C:2004 规则	规则类别（必须/建议）	规则说明	偏差描述
14.3	R	预处理前，空语句只能单独一行出现；可以在其后间隔一个空格添加注释。	空语句接近CyGlobalIntEnable宏；CyGlobalIntEnable宏后面是一个分号，而其实现自身已经包括了分号。可应用于PSoC 3/PSoC 5器件。
14.5	R	不会使用“continue”语句。	使用了两次“continue”语句以简化数据包处理。

MISRA-C:2004 规则	规则类别（必须/建议）	规则说明	偏差描述
14.7	R	函数应在结尾处有单一的退出点。	在验证Bootloadable应用的有效性的函数中使用了多个退出点。
19.7	A	使用时，函数应优先于类函数宏。	由于使用了函数宏以实现更高效的代码，所以出现了偏差。

Bootloadable 组件的特定偏差：

MISRA-C:2004 规则	规则类别（必须/建议）	规则说明	偏差说明
19.7	A	使用时，函数应优先于类函数宏。	由于使用了函数宏以实现更高效的代码，所以出现了偏差。

样例固件源代码

PSoC Creator 在“Find Example Project”对话框中提供了包括原理图和代码示例的许多示例项目。要查看特定组件实例，请打开“Component Catalog”中的对话框或者原理图中的组件样例。要查看通用示例，请打开“Start Page”或 **File** 菜单中的对话框。根据要求，可以通过使用对话框中的 **Filter Options** 选项来限定可选的项目列表。

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

功能描述

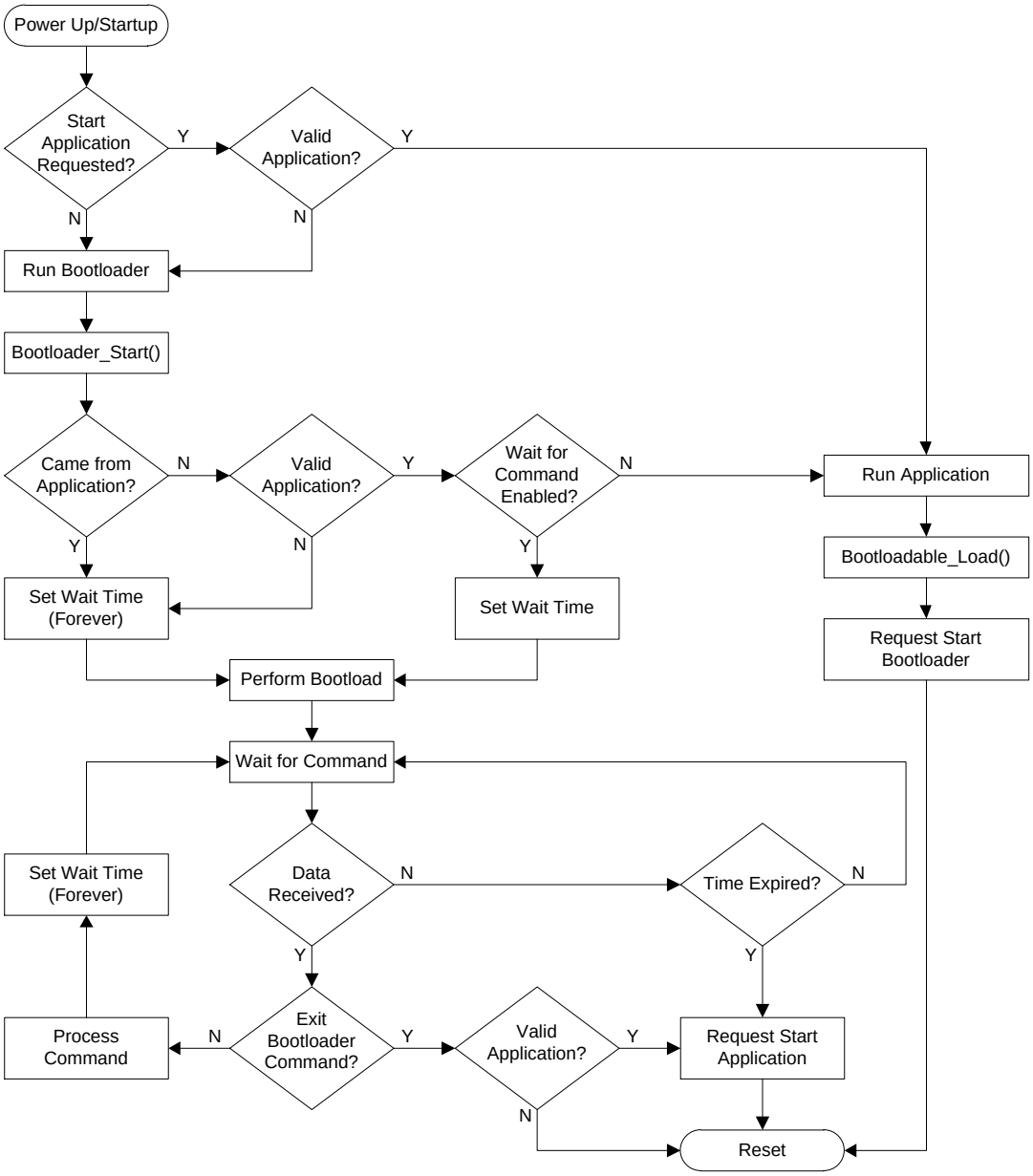
Bootloader 和 Bootloadable 项目的函数

Bootloader 项目通过它自己的通信组件将 Bootloadable 项目或新代码传输到闪存中。传输完成后，总会复位处理器。如果 Bootloadable 项目不存在或者已损坏，在复位时间内，Bootloader 项目将检测某些条件并可能会自动启动传输。

启动时，Bootloader 代码加载其自己的配置数据。此外还必须初始化堆栈、其他资源以及外设以便进行传输。传输完成后，通过软件复位将控制权传递到 Bootloadable 项目中。

然后， Bootloadable 项目加载自己的配置，并为它自己的函数重新初始化堆栈和资源及外设。为了切换到 Bootloader 应用， Bootloadable 的项目会调用 Bootloadable_Load()函数（这样将导致发生另一个软件复位）。

下图介绍的是 Bootloader 的运行方式。



Bootloader 应用

通常将 Bootloader 组件和通信组件拖动到原理图上，然后将输入/输出连接到引脚上，并设置时钟等操作来完成某个一 Bootloader 的设计项目。包含了 Bootloader 组件和通信组件的项目将执行基

本的 Bootloader 应用功能，即接收新代码并将该代码写入到闪存中。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 项目添加自定义函数。

Bootloadable 应用

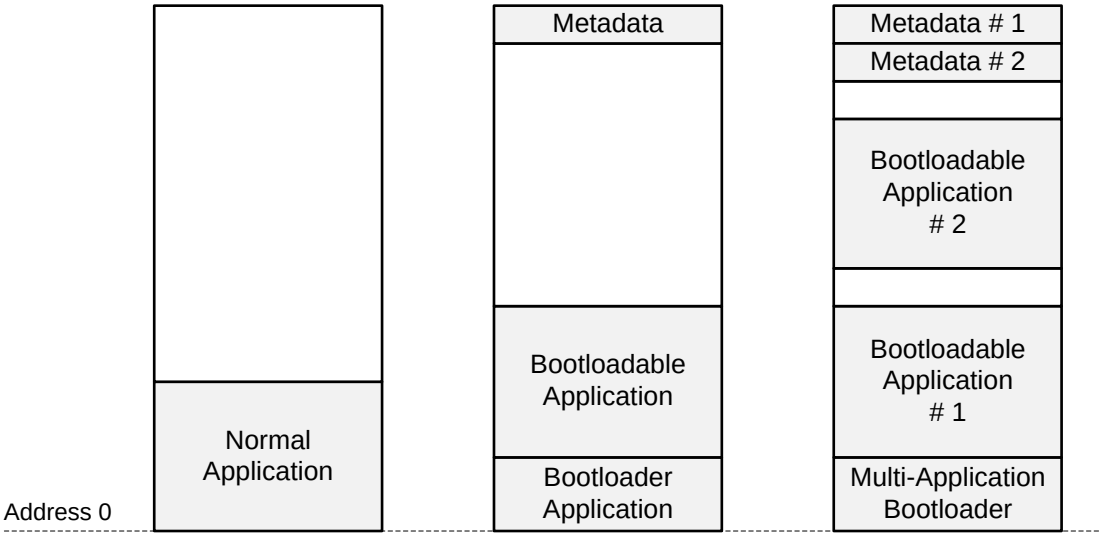
实际上，Bootloadable 应用就是工作工程的代码。该应用与普通的应用类型相类似。主要的区别是 Bootloadable 的应用通常与 Bootloader 应用相关联，而普通项目从不与 Bootloader 的应用相关联。

存储器的使用情况

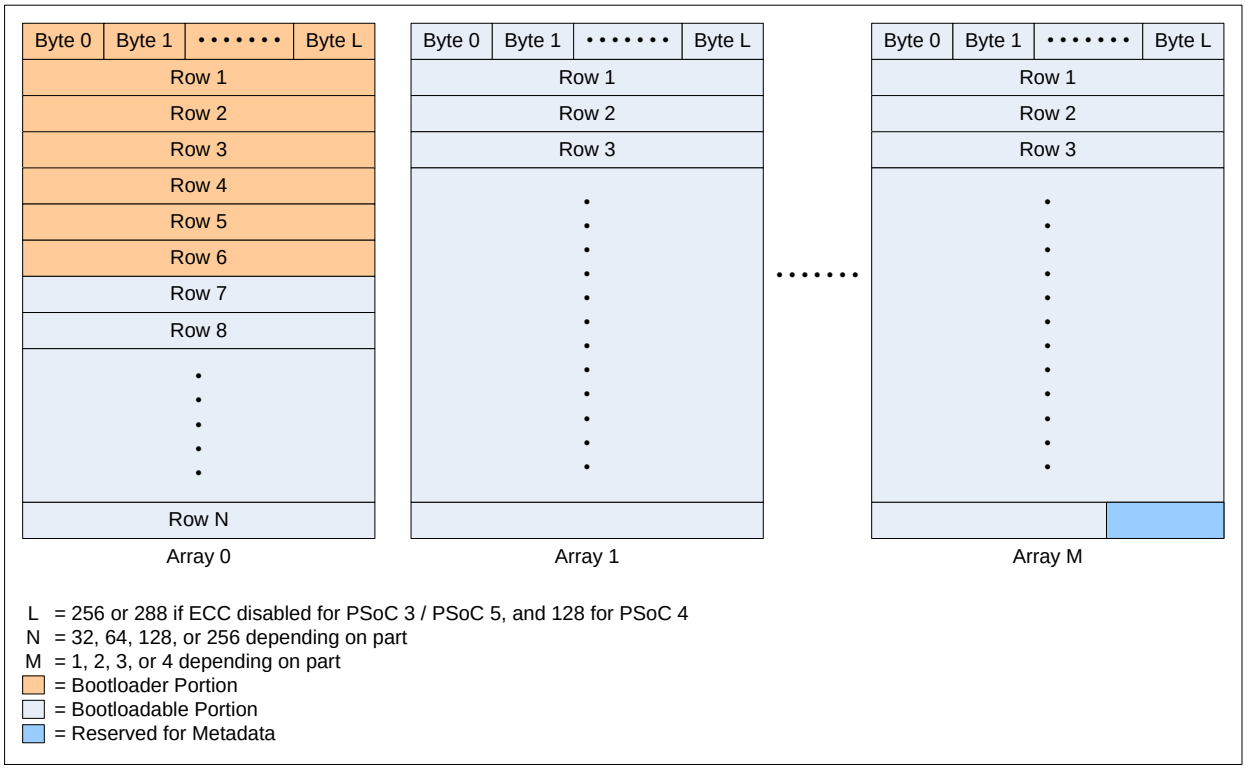
将普通应用和 Bootloader 应用存储在以地址 0 开始的闪存中。Bootloadable 应用使用从 Bootloader 应用的下一个空白行开始的闪存。对于多应用 Bootloader ，第一个 Bootloadable 应用位于 Bootloader 应用的上边。第二个 Bootloadable 应用占用闪存时，其起始地址为第一个 Bootloadable 应用开始和闪存末尾中间所在的行。

如果 Bootloadable 组件定制器中的 **Manual application image placement** 选项有效，则通过使用 **Placement address** 选项能够指定 Bootloadable 应用所在的位置。

下图（从左向右）依次显示的是普通应用、Bootloader 及 Bootloadable 应用、多应用 Bootloader 及两个 Bootloadable 应用使用存储器的情况：



下图显示了器件的闪存布局。



Bootloader 项目始终使用闪存底部的 X 行。对 X 进行设置，以便为以下内容提供足够的闪存：

- 该项目的矢量表，以地址 0 开始（PSoC 3 除外），
- Bootloader 项目的配置字节，
- Bootloader 项目的代码和数据，
- 闪存 Bootloader 部分的校验和

Bootloader 项目的配置字节始终存储在主闪存中，而不是存储在 ECC 闪存中。相关选项会从项目的 Design-wide Resource 文件中删除。

闪存的 Bootloader 应用部分应该在 Design-wide Resource 文件中 Flash Protection（闪存保护）选项卡下受到保护；只有通过 JTAG/SWD 进行下载才能将其覆盖。

Bootloadable 项目紧挨着 Bootloader，占用从首行边界开始的闪存，具体包括以下内容：

- 项目的矢量表（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，用以存储由 Bootloader 和 Bootloadable 项目共同使用的元数据。

Bootloadable 项目的配置字节和标准项目中的配置字节的存储方式相同，即存储在主闪存或 ECC 闪存中，具体情况取决于 Design-wide Resource 文件中的设置。

PSoC 3 的详细信息

在 PSoC 3 中，唯一的“异常矢量”是地址 0 处的 3 字节指令，在处理器复位时执行该指令。（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 5LP 和 PSoC 4 的详细信息

在 PSoC 5LP / PSoC 4 器件中，在地址 0 处必须放置异常矢量表（该表位于矢量表偏移寄存器指向的地址，该寄存器的地址为 0xE000ED08，其值在复位时被设置为 0）。Bootloader 代码则在该表后面。

该表包含 Bootloader 项目的初始堆栈指针（SP）值以及 Bootloader 项目代码的起始地址。另外它还包含了用于 Bootloader 的异常矢量及中断矢量。

Bootloadable 项目还有其自己的矢量表，该矢量表内包含了项目的起始 SP 值和第一个指令地址。完成代码传输后，作为将控制权传递到 Bootloadable 项目的一部分，矢量表偏移寄存器中的值将被更改为 Bootloadable 项目表中的地址。

元数据

元数据部分是闪存的 64 字节模块，作为 Bootloader 和 Bootloadable 应用的公共区域使用。对于 Bootloader 应用，元数据位于行 N-1；如果作为多应用 Bootloader ，第一个 Bootloadable 应用使用行 N-1，第二个应用使用行 N-2 进行存储其元数据，其中 N 是选定器件闪存行的总数量。

元数据的存储器映射

地址	PSoC 3	PSoC 4 / PSoC 5LP
0x00	Bootloadable应用的校验和	
0x01	预留	Bootloadable应用的启动程序地址
0x02		
0x03	Bootloadable应用的启动程序地址	
0x04		
0x05	预留	Bootloader的最后行
0x06		
0x07	Bootloader的最后行	预留
0x08		
0x09		

地址	PSoC 3	PSoC 4 / PSoC 5LP
0x0A	预留	Bootloadable应用长度
0x0B	Bootloadable应用长度	
0x0C		
0x0D	预留	
0x0E		
0x0F		
0x10	有效的Bootloadable应用	
0x11	Bootloadable应用的验证状态	
0x12	Bootloader程序版本	
0x13		
0x14	Bootloadable应用ID	
0x15		
0x16	Bootloadable应用版本	
0x17		
0x18	Bootloadable应用的自定义ID	
0x19		
0x1A		
0x1B		
0x1C- 0x3F	预留	

名称	说明
Bootloadable应用的校验和	这是基本的校验求和值，通过对Bootloadable应用映像图的所有字节（元数据部分除外）进行加法计算得到的。
Bootloadable应用的启动程序地址	Bootloadable应用的启动程序地址这是PSoC 3中的STARTUP1以及PSoC 4/PSoC 5中的Reset()函数。链接器可以将这些函数放置在应用的最小起始地址后面的任意位置。
Bootloader的最后闪存行	Bootloader应用映射图所在的最后闪存行的编号。 **注意：**对于多应用Bootloader中的第二个Bootloadable应用，该字段包含了第一个Bootloadable应用所占用的最后闪存行。
Bootloadable应用长度	Bootloadable应用的大小，单位为字节。

名称	说明
有效的Bootloadable应用	如果使能了**Multi-application bootloader**（多应用的Bootloader）项，该字段包含了有效 Bootloadable应用的相关信息。
Bootloadable应用的验证状态	如果使能了**Fast bootloadable application validation**（Bootloadable应用的快速验证）项，该字段包含了Bootloadable应用的验证状态：Bootloader只会计算校验和一次，后续的每一次启动将重新使用该值。
Bootloader应用版本	该字段包含了Bootloader的应用版本。可通过Bootloader组件的定制器指定该字段。
Bootloadable应用ID	该字段包含了Bootloadable应用的ID。可通过Bootloadable组件的定制器指定该字段。
Bootloadable应用版本	该字段包含了Bootloadable的应用版本。可通过Bootloadable组件的定制器指定该字段。
Bootloadable应用的自定义ID	该字段包含了Bootloadable应用的自定义ID。可通过Bootloadable组件的定制器指定该字段。

注意：按照处理器的顺序来保存所有字段：PSoC 3 中为低位优先，则 PSoC 4/PSoC 5LP 中为高位优先。

PSoC Creator 项目输出文件

在创建 Bootloader 项目或 Bootloadable 项目时，都会为相应项目创建输出文件。

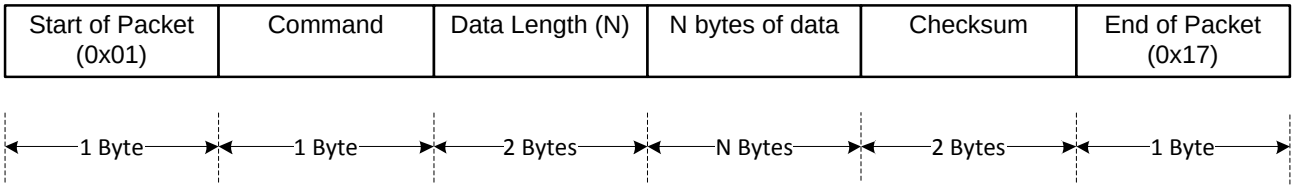
此外，在构建 Bootloadable 项目时，还会为这两种项目创建一个输出文件，即“组合”文件。该文件同时包含了 Bootloader 项目和 Bootloadable 项目。在生产环境中通常需要使用它，便于将这两种项目加载（通过 JTAG/SWD）到器件闪存中。

Bootloadable 项目的配置字节存储在主闪存或 ECC 闪存中。Bootloadable 项目输出文件的格式有以下特点：当器件上含有已被禁用的 ECC 字节时，可减少执行传输操作的时间。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该操作。Bootloader 通过对相关联的闪存行进行一次编程来利用该交错的结构，该闪存行同时包含了主闪存和 ECC 闪存的字节。

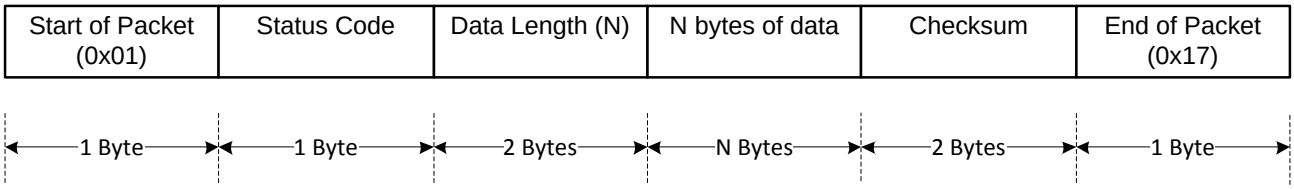
每个项目都有自己的校验和。在项目构建期间，校验和包含在输出文件中。

Bootloader 数据包结构

从主机发送到 Bootloader 的通信数据包的结构如下：



从 Bootloader 读取的响应数据包的结构如下：



状态/错误代码

Bootloader 中可能的状态/错误代码输出如下：

状态/错误代码	值	说明
CYRET_SUCCESS	0x00	成功接收并执行指令
BOOTLOADER_ERR_VERIFY	0x02	闪存未通过验证
BOOTLOADER_ERR_LENGTH	0x03	现有的数据量超出了预期范围
BOOTLOADER_ERR_DATA	0x04	数据格式错误
BOOTLOADER_ERR_CMD	0x05	无法识别指令
BOOTLOADER_ERR_DEVICE	0x06	预期器件与所检测到的器件不匹配。
BOOTLOADER_ERR_VERSION	0x07	检测到的Bootloader版本不受支持。
BOOTLOADER_ERR_CHECKSUM	0x08	数据包校验和与预期的值不匹配
BOOTLOADER_ERR_ARRAY	0x09	闪存阵列ID无效
BOOTLOADER_ERR_ROW	0x0A	闪存行编号无效
BOOTLOADER_ERR_APP	0x0C	应用无效，并且不可设置为有效
BOOTLOADER_ERR_ACTIVE	0x0D	当前应用被标记为有效状态
BOOTLOADER_ERR_UNK	0x0F	发生了未知错误

Bootloader 指令

Bootloader 支持这些指令。对于所有接收到的字节，如果开始部分不是该组指令字节中的某一个，则该字节会被丢弃，并且不会生成响应。所有多字节字段都是输出最低有效位优先。

注意：Bootloader 执行任何指令所需的时间取决于器件的配置情况。有些因素会影响时序，包括：

- 器件运行的时钟速度
- 用于构建项目的工具链
- 构建过程中使用的优化设置
- 后台运行的中断数

Bootloader指令名称（指令代码）			
数据字节（字节数量）	**响应数据包的状态代码**	**响应数据包的数据（字节数量）**	**说明**
进入Bootloader（0x38）			
不可用	成功 错误指令 错误数据 错误长度 错误校验和	4字节 — 芯片ID 1字节 — 芯片修订版 3字节 — 版本	Bootloader接收到此指令时，将以器件的信息以及Bootloader组件的版本进行回应。 这里的“版本”是指Bootloader组件的版本。
获取闪存大小（0x32）（可选）			
1字节 — 闪存阵列ID	成功 错误指令 错误数据 错误长度 错误校验和	2字节 — 第一个可用行 2字节 — 最后一个可用行	Bootloader接收到此指令时，将使用紧随Bootloader应用的第一整行（即Bootloadable应用的第一行）以及所选闪存阵列中最后一行进行响应。
编程行（0x39）			

Bootloader指令名称（指令代码）			
数据字节（字节数量）	**响应数据包的状态代码**	**响应数据包的数据（字节数量）**	**说明**
1字节 — 闪存阵列ID 2字节 — 闪存行数 n字节 — 要写入闪存行的数据	成功 错误指令 错误数据 错误长度 错误校验和 错误闪存行 错误活动	不可用	将一个闪存行的数据写入到器件内。 通过使用“Send Data”（发送数据）指令，可以使用多个数据包发送要写入闪存行的数据。 可同时传送该指令和最后数据模块，对闪存行进行编程。
擦除行（0x34）（可选）			
1字节 — 闪存阵列ID 2字节 — 闪存行数	成功 错误指令 错误数据 错误长度 错误校验和 错误闪存行 错误活动	不可用	擦除所提供的闪存行中的内容。
验证行（0x3A）（可选）			
1字节 — 闪存阵列ID 2字节 — 闪存行数	成功 错误指令 错误数据 错误长度 错误校验和	1字节 — 行校验和	获取所提供的闪存行内容的1字节校验和
验证校验和（0x31）			
不可用	成功 错误指令 错误数据 错误长度 错误校验和	1字节 — 校验和有效	返回值非零表示应用代码闪存的校验和与预期的校验和值相互匹配，即应用有效。 返回值0则表示校验和不匹配，即应用无效
发送数据（0x37）（可选）			

Bootloader指令名称（指令代码）			
数据字节（字节数量）	**响应数据包的状态代码**	**响应数据包的数据（字节数量）**	**说明**
n字节 — 要保存在器件中的数据	成功 错误指令 错误数据 错误长度 错误校验和	不可用	将一个数据模块发送到器件上。 缓冲该数据，等待另一个指令，该指令会告知Bootloader如何处理该数据。如果连续发出多个“发送数据”指令，则此数据将附加在上一个模块后。 该指令用于将大量的传输数据分为适当的大小，以防止某些协议中出现总线枯竭的情况。
同步Bootloader（0x35）（可选）			
不可用	不可用	不可用	将Bootloader复位至空白状态，准备接受新指令。 将删除所有缓冲中的数据。只有主机和客户端之间不再同步时才需要使用该指令。
退出Bootloader（0x3B）			
不可用	不可用	不可用	通过触发器件的软件复位退出Bootloader。 执行软件复位前，验证Bootloadable应用。如果应用通过了验证，将在软件复位后执行该应用。如果应用未通过验证，则将在软件复位后再次用Bootloader执行该应用。
获取元数据（0x03C）（可选）			
1字节 — 应用编号	成功 错误应用 错误长度 错误数据 错误校验和	56字节 — 元数据	报告所选应用的元数据的头56字节。更多有关元数据的信息，请查阅元数据一节的内容。
获取应用状态（仅适用于多应用Bootloader）(0x33)（可选）			
1字节 — 应用编号	成功 错误长度 错误校验和 错误数据	1字节 — 有效的应用编号 1字节 — 活动的应用编号	返回指定应用的状态。
设置活动的应用（仅适用于多应用Bootloader）（0x36）			

Bootloader指令名称（指令代码）			
数据字节（字节数量）	**响应数据包的状态代码**	**响应数据包的数据（字节数量）**	**说明**
1 字节——应用编号	成功 错误应用 错误长度 错误数据 错误校验和	不可用	指定的Bootloadable应用被设置为活动状态。此指令用于对两个Bootloadable应用进行切换。

Bootloader 应用和代码数据文件格式

Bootloader 应用和代码数据（.cyacd）文件格式可存储设计中 Bootloadable 的部分。该文件是由若干闪存数据行跟随的头文件。除标题外，“.cyacd”文件中的每一行均表示闪存数据的一整行。数据以高位优先（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]

标题中的校验和类型分别用于指示在 Bootloader 主机和 Bootloader 本身之间发送的数据包的校验和类型。数据记录中的校验和是基本求和，通过对所有字节求和（不包括校验和本身），然后计算其二进制补码可以得到。

Bootloader 主机工具

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

源代码

除了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 Bootloader 主机应用。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 Bootloader 主机所需的各种功能。根据所需的控制级别，部分或所有模块可用于开发自定义 Bootloader 主机应用。

cybtldr_command.c/h

该模块用于构建传送到 Bootloader 的数据包，并解析从 Bootloader 所收到的数据包。该模块具有两个单一函数，一个用于构建适合于 Bootloader 的各类数据包，另一个用于解析 Bootloader 返回的结果。

cybtldr_parse.c/h

该模块用于解析“*.cyacd”文件，该文件包含要发送至器件的 Bootloadable 映像。它包含多个函数，用于设置文件访问、读取标题、读取行数据和关闭文件各操作。

cybtldr_api.c/h

通过该行级 API 的模块，可以将一行的数据内容发送到使用了所提供通信机制的 Bootloader 内。它包含多个函数，用于设置引导加载操作、编程单一行、擦除单一行、验证单一行和结束引导加载各操作。

cybtldr_api2.c/h

该模块是级别更高的 API，用于处理整个引导加载流程。它包含的多个函数可用于进行编程器件、擦除器件、验证器件和中止当前操作。

资源

Bootloader 和 Bootloadable 项目使用以下器件资源：

- Bootloader 组件使用复位状态（RESET_SR0）寄存器上的通用位。当同 Bootloader 进行通信，并且通信内容与软件复位无关时，则需要使用这些位。
- 可以在相应的组件数据手册中查看通信组件使用的资源。

API 的内存使用量

根据编译器、器件、所使用的 API 数量以及组件的配置情况不同，组件的内存使用量也不一样。下表提供了给定组件配置中的所有 API 所使用存储空间。

通过释放模式中所配置的相应编译器来完成测量操作。在释放模式下，可以得到最优化的尺寸。有关特定的设计，可分析编译器生成的映射文件以确定存储器的大小。

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	2219	4	962	0	1036	0
Bootloader（应用于整个项目）[1]	6570	1559	3816	432[3]	4128	605[3]
Bootloadable（应用于整个项目）[2]	1770	95	4984	256[3]	5710	333[3]

注意：

1. 在该配置中，对整个 Bootloader 项目进行测量：基于固定功能的 I2C 作为通信组件使用，并且最小化 Bootloader 组件中的闪存大小。
2. 在该配置中，对整个可引导加载项目进行测量。
3. 所显示的 SRAM 大小不包括堆和栈的大小。

组件更改

本节列出了各版本的主要组件更改内容。

版本	更改说明	更改原因/影响
1.20	“Wait for command time”（等待指令的时间）选项的单位从10 ms改为100 ms。	**注意：**更新为版本1.20时，“Wait for command time”选项的值自动增加了10倍。
	添加了“Get Metadata”（获取元数据）指令。	报告所选应用的元数据的头56字节。
	Bootloader应用将忽略所有指令（除“Exit Bootloader”和“Sync Bootloader”指令外），直到接收到“Enter Bootloader”指令为止。	Bootloader应用只等待有效的流量（由“Enter Bootloader”标志），而不是所有流量。 如果接收到的流量不包含有效的“Enter Bootloader”指令，超过所指定的超时时间后，便启动Bootloadable应用。
	更新了**Dependencies**选项卡。	添加了指定Bootloader ELF文件的字段。
	更新了“MISRA 合规性”章节。	验证了Bootloader/Bootloadable组件的MISRA合规性，并描述了具体的偏差。
1.10	已添加MISRA合规性章节。	Bootloader/Bootloadable组件未经过MISRA合规性的验证。
	添加了PSoC 4器件支持。	新器件支持
	对数据手册进行了较小的编辑	

版本	更改说明	更改原因/影响
1.0.a	更正了数据手册	
1.0	初始组件版本	