AN3388

针对 MPLAB® Harmony v2 用户的 MPLAB Harmony v3 应用程序开发指南

简介

本文档旨在指导 MPLAB® Harmony v2 用户如何使用 MPLAB Harmony v3 开发应用程序。

MPLAB Harmony 是一个软件框架，由兼容且可互操作的模块组成，例如外设库（PLIB）、驱动程序、系统服务、中间件和第三方库。MPLAB Harmony v3 具有与 MPLAB Harmony v2 相同的基本原理；但此版本新增了功能并进行了改进。

© 2020 Microchip Technology Inc.

目录

1. 工具与安装

MPLAB Harmony v2 打包为 zip 文件，其中包含开发应用程序所需的所有软件组件或模块，例如 MPLAB Harmony 配置器（MPLAB Harmony Configurator，MHC）文件、PLIB、驱动程序、多个中间件库和第三方库。

相比之下，MPLAB Harmony v3 具有模块化下载功能。MPLAB Harmony v3 组件分组为不同的软件包，用户只需下载项目所需的软件包。可以从 GitHub 下载 MPLAB Harmony v3 软件包。有关 MPLAB Harmony v3 环境设置的更多信息，请参见文档如何设置 MPLAB Harmony v3 软件开发框架。

注： 在本文档中，术语“模块”和“组件”可通篇互换使用。

2. MPLAB Harmony 配置器 GUI

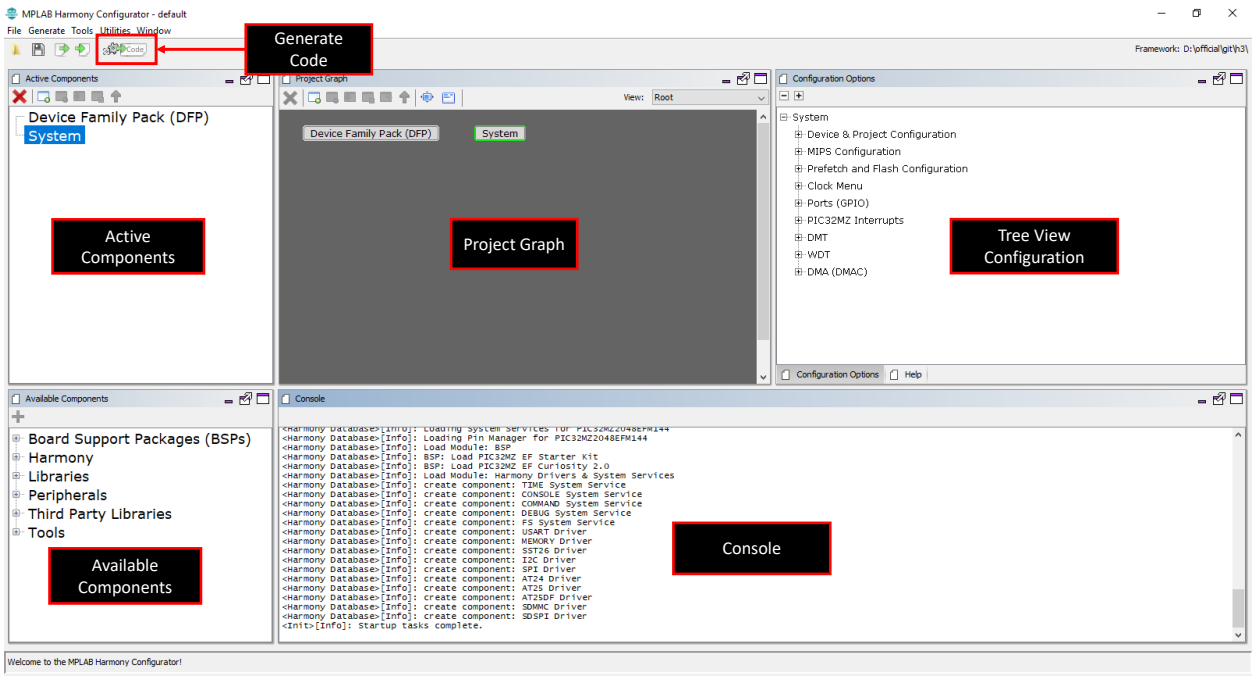
MPLAB Harmony 配置器（MHC）是一款基于 GUI 的工具，可轻松使能、配置和生成代码，适用于各种 MPLAB Harmony 模块。MHC 是 MPLAB X IDE 的插件。

要下载适用于 MLPAB Harmony v3 的 MHC 插件，请按照以下步骤操作：

1. 打开 MPLAB X IDE。
2. 选择 *Tools* > *Plug-ins Download*（工具 > 插件下载），然后单击 **Go to MPLAB X Plug-in Manager**（转到 MPLAB X 插件管理器）。随即将显示 Available Plugins（可用插件）窗口。
3. 在 Available Plugins 窗口中，选择 MPLAB Harmony Configurator v3（MPLAB Harmony 配置器 v3），然后单击 **Install**（安装）并按照插件安装程序的提示进行操作。
4. 选择 **Restart Now**（立即重启）。

在 MPLAB Harmony Configurator v2（MPLAB Harmony 配置器 v2）中，所有模块都以树形结构排列，可通过展开“+”号进行配置。但是，在 MPLAB Harmony Configurator v3 中，可以使用多个窗口更轻松、直观地进行配置。下图所示为 MPLAB Harmony Configurator v3 窗口。

图 2-1. MPLAB Harmony 配置器窗口



2.1 可用组件

此窗口中列出了已下载相应软件包的所有模块。Board Support Package (BSP)（板级支持包）和 Peripherals（外设）中列出的模块适用于所建项目的器件。列出的其余组件适用于所有器件。**Harmony** 组件下列出了 MPLAB Harmony 驱动程序和系统服务。

2.2 活动组件

此窗口列出了项目中使用的所有模块。要使用某个模块，用户需要从 Available Components（可用组件）列表中双击该模块。Active Components（活动组件）列表中默认添加了一些模块，例如 Device Family Pack (DFP)（器件系列包）和 System（系统）。要从所使用的模块或 Active Components 列表中移除某个模块，请选择该模块，然后单击 Active Components 窗口左上方出现的按钮（红色叉号）。

注： 活动组件列表中必须有以下 MPLAB Harmony v3 模块：

- System
- DFP
- CMSIS（仅在使用 SAM 器件时出现）

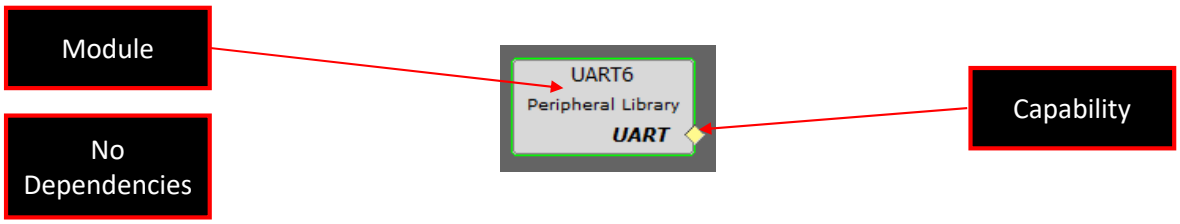
2.3 项目图

Project Graph（项目图）给出了项目中正在使用的所有模块，并提供了更多详细信息。在 MPLAB Harmony 软件框架中，一些模块与单片机硬件紧密关联（例如外设库），另有一些模块不直接与硬件相关联，而是依赖于与硬件相关的模块。例如，MPLAB Harmony 驱动程序依赖于外设库。此外，还有一些依赖于 MPLAB Harmony 驱动程序的软件模块，例如文件系统服务和中间件库。

由于 MPLAB Harmony 具有各种存在依赖关系结构的软件模块，因此 MPLAB Harmony v2 中的整体配置树有时会变得难以管理和配置。MPLAB Harmony v3 中的 Project Graph 窗口解决了这一问题。在 Project Graph 窗口中，每个模块以矩形框的形式显示。在这些矩形框的左边缘，显示了一些小框（依赖关系框），用于指示模块的依赖关系。具有依赖关系意味着模块依赖于其他模块来确保功能正常。例如，USART 驱动程序模块依赖于 UART（或 USART）PLIB。同样，在矩形框的右边缘，也将显示小框（功能框），用于指示模块的功能。具有功能意味着模块会将其功能公开给其他模块。例如，UART PLIB 模块具有 UART 的功能，SERCOM PLIB 模块具有 UART、I^2C 和 SPI 的功能。下图给出了模块、其依赖关系和功能的示例：

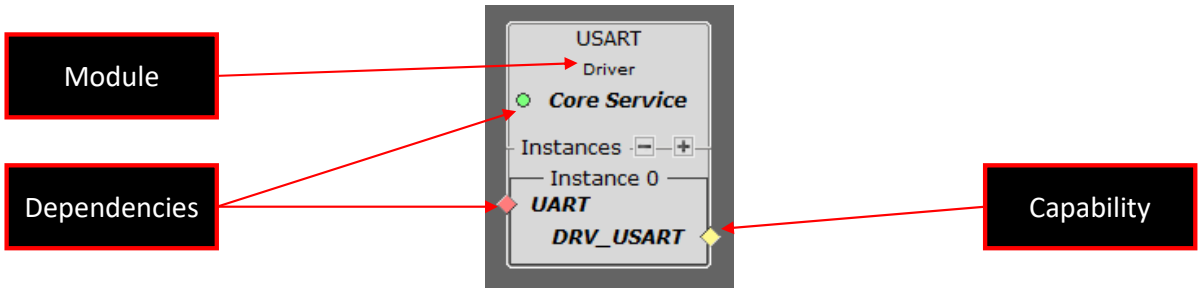
注： 少数模块不具有任何依赖关系或功能，例如，PLIB 模块没有任何依赖关系。系统模块既没有依赖关系也不具有功能。

图 2-2. PLIB 模块的功能



- UART6 外设库（PLIB）是模块
- UART6 PLIB 没有依赖关系
- UART6 PLIB 具备一种功能：UART

图 2-3. 驱动程序模块的依赖关系和功能

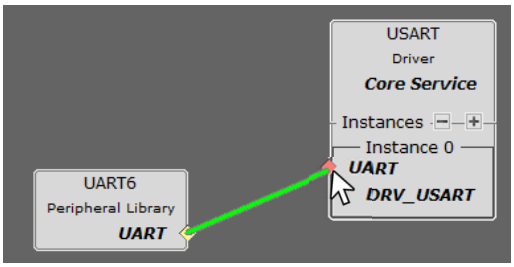


- USART 驱动程序是模块
- USART 驱动程序上有两种依赖关系：内核服务和 UART
- USART 驱动程序具备一种功能：DRV_USART

在项目图上，依赖模块可以连接到相应的功能模块，以正确使用依赖模块。连接两个模块以使用依赖模块的这一步骤称为满足依赖关系。可以通过以下三种不同方式来满足依赖关系：

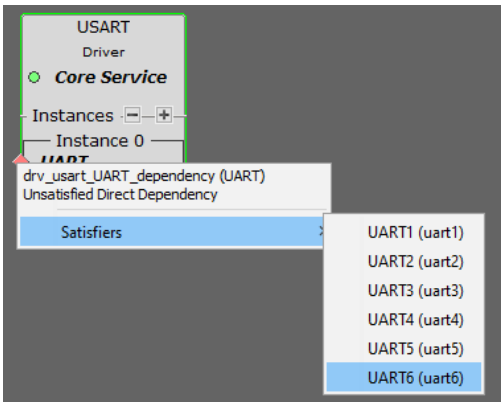
1. 单击左键并按住，在依赖关系框和功能框之间拖动鼠标指针，如下图所示：

图 2-4. 满足依赖关系 1



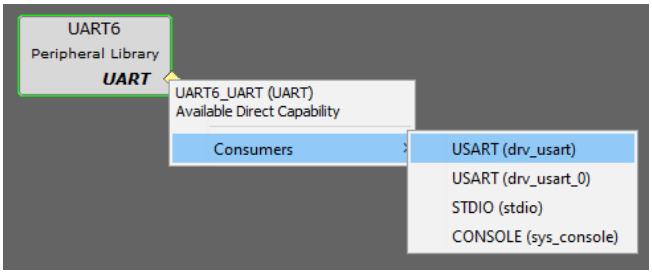
2. 右键单击依赖关系框，然后选择适当的满足条件，如下图所示：

图 2-5. 满足依赖关系 2



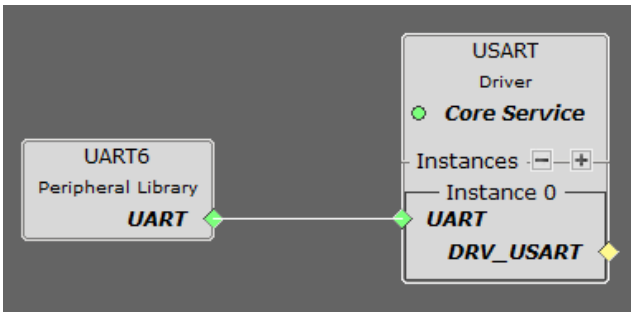
3. 右键单击功能框，然后选择适当的使用者，如下图所示：

图 2-6. 满足依赖关系 3



在配置模块和生成代码之前，请确保满足模块的所有依赖关系。下图所示为模块满足依赖关系后的外观：

图 2-7. 已连接模块



此外，用户还可以通过移动 Project Graph 内的模块框（单击、按住并移动）来适当地排列它们，以便使项目图更有条理。

2.3.1 依赖关系

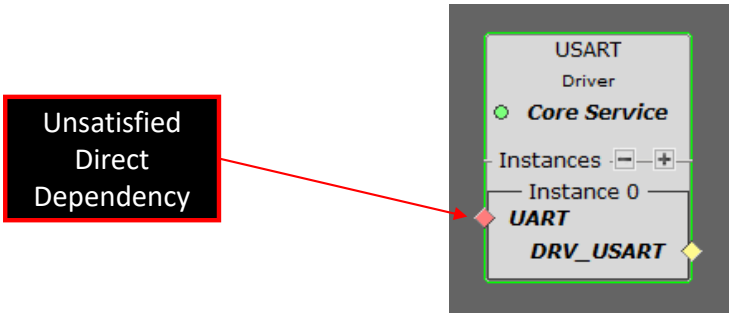
如果模块具有任何依赖关系，则会在 Project Graph 中的模块左侧以小框（圆形、菱形或正方形）的形式显示：

- 绿色框表示已满足依赖关系。
- 粉色框表示需要满足依赖关系。
- 黄色框表示可以选择满足此依赖关系。

以下是三种依赖关系：

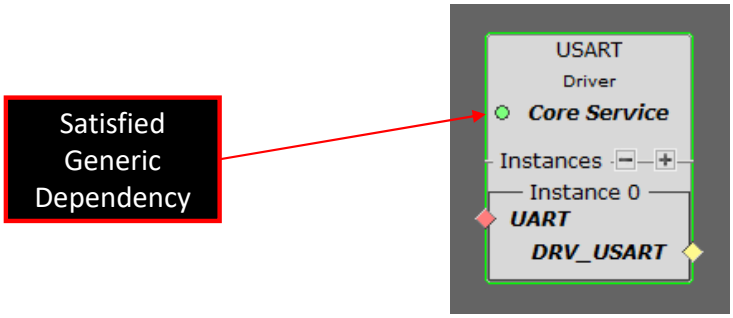
- **直接依赖关系：**需要手动连接才能满足它们。它们以小菱形符号表示。

  图 2-8. 直接依赖关系



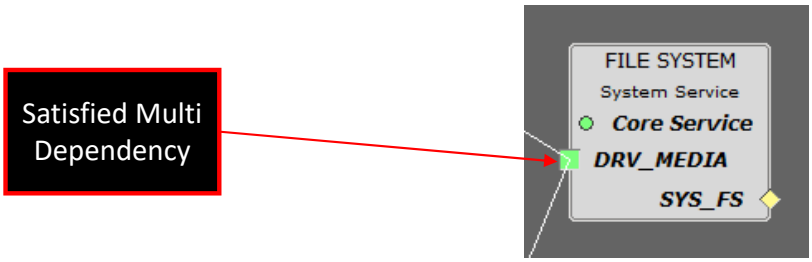
- **通用依赖关系：**不需要任何连接即可满足它们。当项目图中存在提供功能的任何其他模块时，它们会自动变为绿色（满足条件）。它们以小圆形符号表示。

  图 2-9. 通用依赖关系



- **多重依赖关系：**允许建立多个连接来满足它们。它们以小方形符号表示。

图 2-10. 多重依赖关系



2.3.2 功能

如果模块具有任何功能，则会在 Project Graph 中的模块右侧以小框（圆形、菱形或正方形）的形式显示：

- 绿色框表示已满足功能。
- 粉色框表示需要满足依赖关系。
- 黄色框表示可以选择满足此依赖关系。

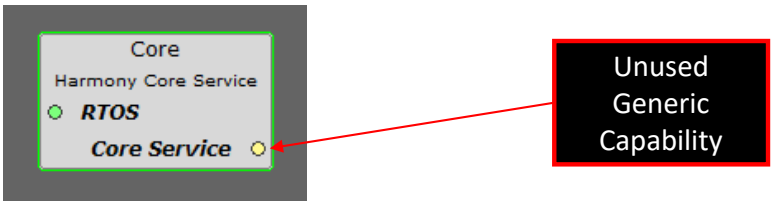
以下是三种功能：

- **直接功能：**需要手动连接才能使用它们。它们以小菱形符号表示。

图 2-11. 直接功能

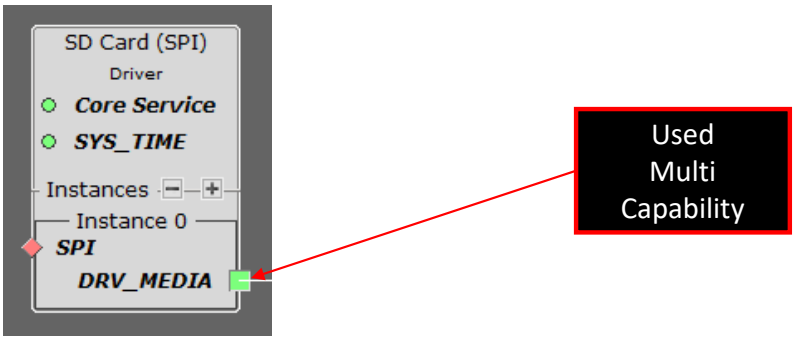
- **通用功能：**无需任何连接即可使用它们。当项目图中存在依赖它们的其他模块时，它们会自动变为绿色（已使用）。它们以小圆形符号表示。

图 2-12. 通用功能



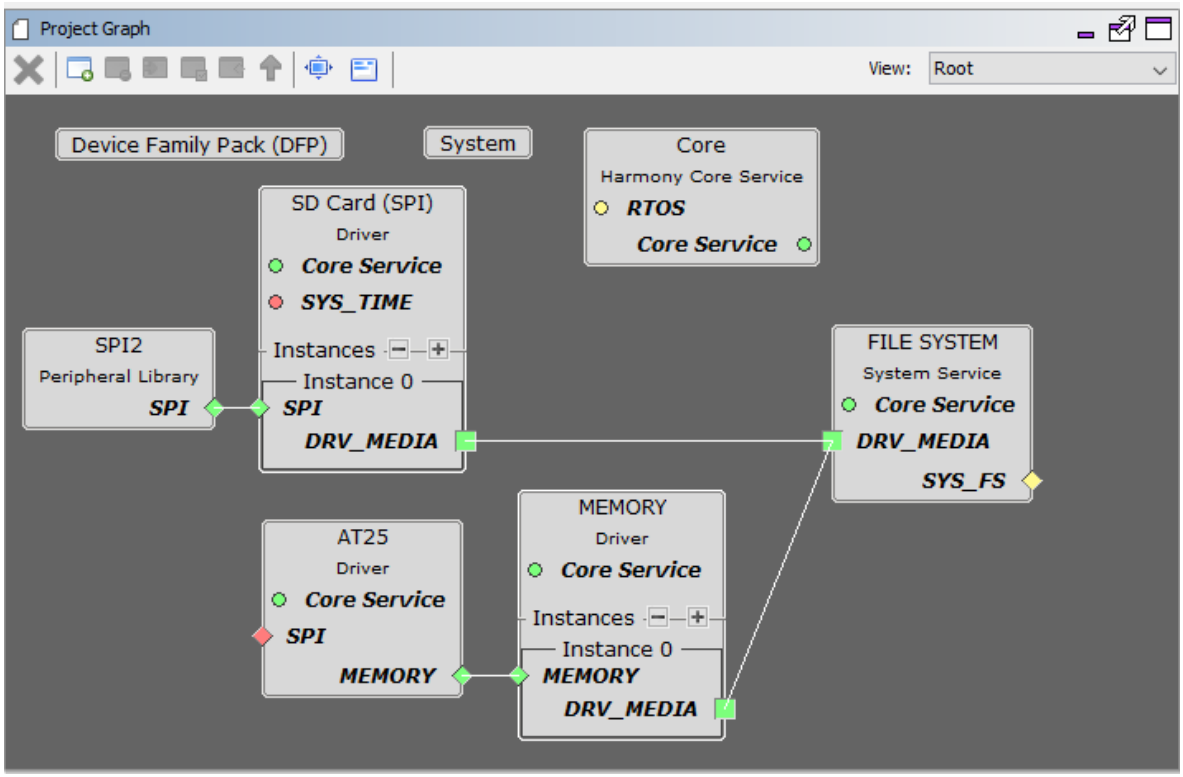
- **多重功能：**必须使用以满足模块的多个依赖关系。它们以小方形符号表示。

图 2-13. 已使用的多重功能



下图给出了 MHC 项目图模块的依赖关系和功能状态：

图 2-14. MHC 项目图



- 器件系列包和系统模块不具备任何功能或依赖关系。
- 内核模块对 RTOS 具有可选的通用依赖性。由于项目图上没有模块具备 RTOS 功能，因此始终无法满足这种依赖关系。但是由于这是一个可选的依赖关系，因此不要紧。
- 内核模块提供内核服务的通用功能。因为它是通用功能，因此 SD Card Driver、AT25 Driver、MEMORY Driver 和 FILE SYSTEM 模块无需连接即可使用它。
- SD Card Driver 对 SYS_TIME 具有通用依赖性，由于项目图中没有提供 SYS_TIME 功能的模块，因此 SD Card Driver 以粉色小圆圈显示这种依赖关系。用户必须添加时间系统服务模块来满足这种依赖关系，这样才能成功生成代码。
- SPI2 外设库具有直接功能，此功能用于满足 SD Card Driver Instance 0 的依赖关系。

- AT25 Driver 对 SPI 具有直接依赖性。SD Card Driver 使用具有 SPI 功能的模块。因此，AT25 Driver 的这种依赖关系保持未得到满足的状态，AT25 Driver 框左侧的粉色小菱形表示了这种依赖关系。为进一步操作，用户必须在项目图中添加其他 SPI 外设库组件，并且必须满足这种依赖关系。
- 文件系统具有多个依赖关系，可通过 SD Card Driver 和 MEMORY Driver 的多重功能来满足。

2.4 树形视图配置

与 MPLAB Harmony v2 不同，MPLAB Harmony v3 为每个模块提供一个单独的配置树。一旦用户在项目图中或从可用组件列表中选择了一个模块，此窗口中就会显示相应模块的所有配置选项。

2.5 控制台

此窗口类似于 MPLAB Harmony v2 的 MHC Output（输出）窗口。它用于显示 MHC 相关消息。例如，如果在不满足某种直接依赖关系时尝试生成代码，则控制台窗口会提示错误消息。

2.6 帮助

MPLAB Harmony v2 在 MHC 的右侧有一个“Help”（帮助）窗口，可帮助用户配置模块。MPLAB Harmony v3 未实现此窗口，而是采用高级树形视图。树形视图中的几个配置项都有一个工具提示，将鼠标悬停在相应项上即可看到工具提示。

2.7 生成代码

MPLAB Harmony v3 中的代码生成步骤与 MPLAB Harmony v2 中的相同。添加并配置所有需要的模块之后，用户可以单击 Generate Code（生成代码）按钮来生成 MPLAB Harmony v3 代码并将其添加到项目中。

注： 与 MPLAB Harmony v2 不同，MPLAB Harmony v3 项目不引用 MPLAB Harmony v3 资源库中的 MPLAB Harmony v3 框架文件（外设库、驱动程序和系统服务的源文件）。相反，它会将所有需要的源代码从 MPLAB Harmony v3 资源库复制到本地项目目录。这意味着 MPLAB Harmony v3 项目可以轻松地从一台计算机移植到另一台计算机。

2.8 MHC 实用程序或插件

MPLAB Harmony v3 与 MPLAB Harmony v2 类似，因为两者都具有实用程序，可用于配置一些在树形视图中难以配置的模块，例如时钟配置、引脚配置和 ADC 配置。与 MPLAB Harmony v2 不同，在 MPLAB Harmony v3 中，默认情况下不会启动这些实用程序。需要通过 MPLAB X IDE 的 *MHC>Tools*（MHC>工具）菜单选项（或在独立模式下从 MPLAB Harmony 配置器窗口的 Tools（工具）菜单）手动启动它们。此外，MPLAB Harmony v3 还有一些其他实用程序，例如 DMA 配置和中断（EVIC）配置。

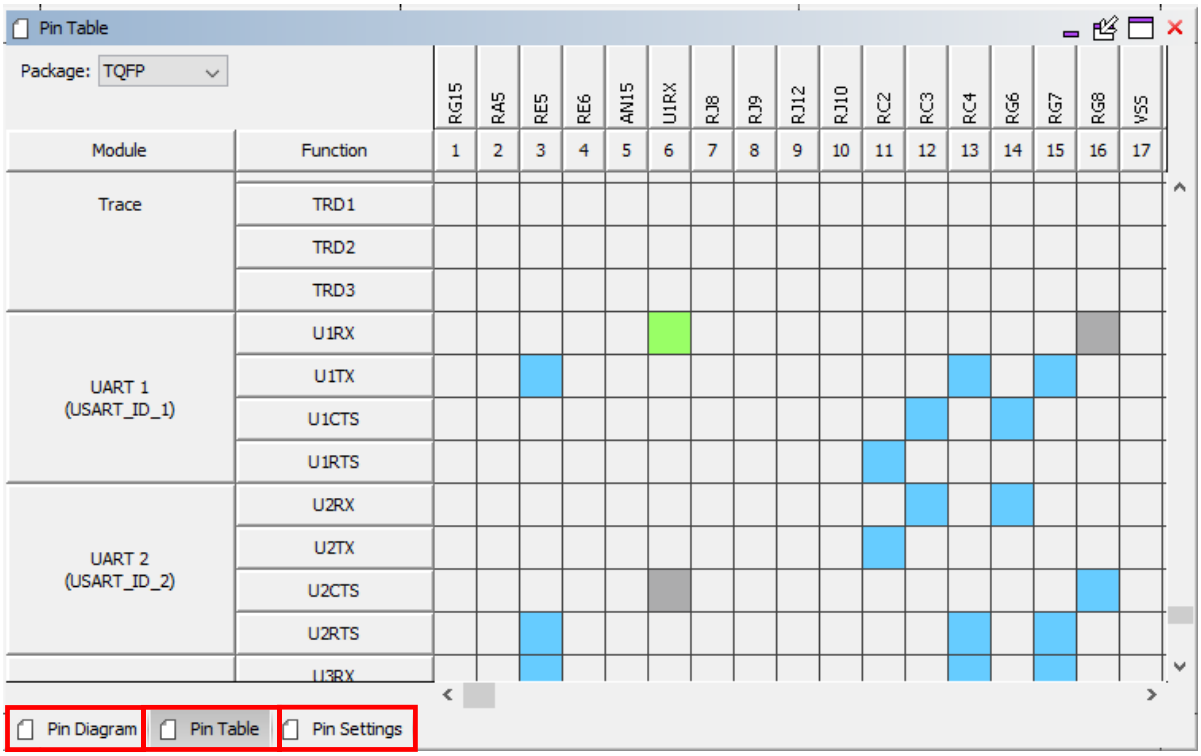
2.8.1 时钟配置

MPLAB Harmony v3 中的时钟配置实用程序与 MPLAB Harmony v2 中的相同，但是包含一些改进和缺陷修正。

2.8.2 引脚配置

MPLAB Harmony v3 中的引脚配置实用程序与 MPLAB Harmony v2 中的相同，但是包含一些改进和缺陷修正。在 MPLAB Harmony v2 中，引脚配置实用程序的 Pin Table（引脚表）窗口与 MPLAB Harmony 配置器的 Output 窗口一起移至底部。但是，在 MPLAB Harmony v3 中，三个引脚配置窗口（Pin Diagram（引脚图）、Pin Table 和 Pin Settings（引脚设置））全都位于同一区域，如下图所示：

图 2-15. MHC 引脚配置



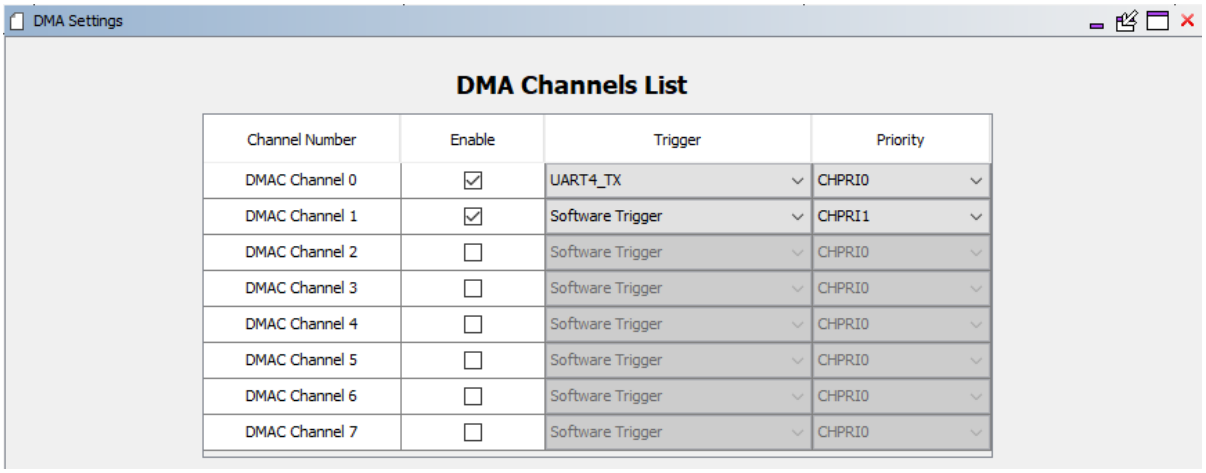
2.8.3 ADC 配置

仅在将 ADC PLIB 添加到项目图上之后，MPLAB Harmony v3 中的 ADC 配置实用程序才会在 MPLAB X IDE 的 *MHC>Tools* 菜单选项（或独立模式下的 MPLAB Harmony 配置器窗口的 Tools 菜单）下列出。MPLAB Harmony v3 和 MPLAB Harmony v2 中的 ADC 配置实用程序相同。

2.8.4 DMA 配置

MPLAB Harmony v3 的 DMA 配置实用程序可用于为不同的事务分配 DMA 通道，如下图所示。

图 2-16. DMA 配置



如果要将 DMA 与 PLIB 模块（例如 UART4 PLIB）一起使用，则必须在 DMA 配置器实用程序中为 UART4 TX 和 RX 配置 DMA 通道，这样才能生成代码。在应用程序中，可以调用 DMA（PLIB 或系统服务）传输 API 执行传输操作。

如果要将 DMA 与驱动程序模块一起使用，则必须在相应的驱动程序配置中选择 DMA 模式，随后驱动程序将自行配置所需的 DMA 通道。用户无需配置 DMA 通道，也无需调用任何 DMA API。

2.8.5 增强型向量中断控制器（EVIC）配置

MPLAB Harmony v3 实用程序提供以下配置选项：

- **优先级和子优先级**：当多个 PLIB 模块与一个中断一起使用时，配置这些中断的优先级和子优先级十分重要。必须使用此窗口完成配置，随后 `plib_evic.c` 的 `EVIC_Initialize` 函数中会生成相应的代码。
- **生成 ISR**：EVIC 配置实用程序的“Use”（使用）列用于为相应的中断生成 ISR。默认情况下，应取消选中所有中断的 *Use* 选项。在中断模式下使用 PLIB 模块时，MHC 会在 EVIC 配置器中自动选中相应的 *Use* 选项，随后 `interrupt.c` 文件中会生成 ISR。如果用户希望在不使用外设的 PLIB 模块的情况下生成该外设的 ISR，则可以手动选中 *Use* 选项，这样便可生成代码。可以使用“Handler Name”（处理程序名称）列配置 ISR 名称。下图给出了为 UART2_RX 和 UART_TX 配置的中断：

图 2-17. 中断配置

EVIC Settings

Vector Number	Vector	Use	Priority (1 = Lowest)	SubPriority	Handler Name
144	SPI2_TX (SPI2 Transmit Done)	☐	1	0	SPI2_TX_Handler
145	UART2_FAULT (UART2 Fault)	☐	1	0	UART2_FAULT_Handler
146	UART2_RX (UART2 Receive Done)	☑	3	0	UART2_RX_Handler
147	UART2_TX (UART2 Transmit Done)	☐	1	0	UART2_TX_Handler
148	I2C2_BUS (I2C2 Bus Collision Event)	☐	1	0	I2C2_BUS_Handler
149	I2C2_SLAVE (I2C2 Slave Event)	☐	1	0	I2C2_SLAVE_Handler
150	I2C2_MASTER (I2C2 Master Event)	☐	1	0	I2C2_MASTER_Handler
151	CAN1 (Control Area Network 1)	☐	1	0	CAN1_Handler
152	CAN2 (Control Area Network 2)	☐	1	0	CAN2_Handler
153	ETHERNET (Ethernet)	☐	1	0	ETHERNET_Handler
154	SPI3_FAULT (SPI3 Fault)	☐	1	0	SPI3_FAULT_Handler
155	SPI3_RX (SPI3 Receive Done)	☐	1	0	SPI3_RX_Handler
156	SPI3_TX (SPI3 Transmit Done)	☐	1	0	SPI3_TX_Handler
157	UART3_FAULT (UART3 Fault)	☐	1	0	UART3_FAULT_Handler
158	UART3_RX (UART3 Receive Done)	☐	1	0	UART3_RX_Handler
159	UART3_TX (UART3 Transmit Done)	☑	1	2	UART3_TX_Handler
160	I2C3_BUS (I2C3 Bus Collision Event)	☐	1	0	I2C3_BUS_Handler
161	I2C3_SLAVE (I2C3 Slave Event)	☐	1	0	I2C3_SLAVE_Handler
162	I2C3_MASTER (I2C3 Master Event)	☐	1	0	I2C3_MASTER_Handler

注： 如果在 MHC 中选择了中断的“Use”列，并且用户尝试覆盖实用程序中的相应设置，则 MHC 会允许此类操作，但可能会导致不可预测的行为或不良行为。

3. 外设库（PLIB）

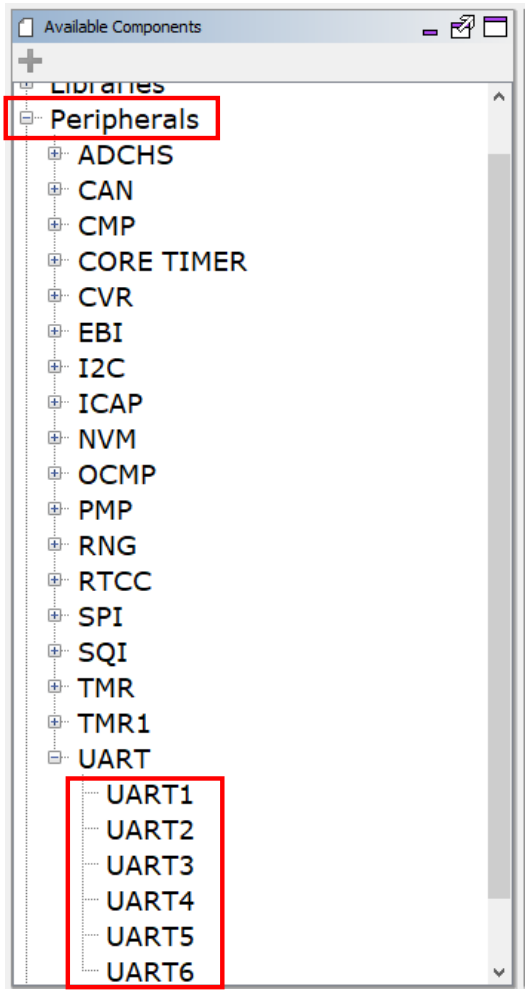
在 MPLAB Harmony v2 和 MPLAB Harmony v3 中，PLIB 与 Microchip 的单片机硬件交互。但是，在 MPLAB Harmony v3 中，对外设库进行了许多更改，以使其对用户更友好。

3.1 Harmony v3 PLIB 使用入门

MPLAB Harmony v3 PLIB 是芯片支持包（Chip Support Package，CSP）的一部分，可按照“工具和安装”部分中所述从 MPLAB Harmony GitHub 页面下载（或克隆）csp 资源库来使用它们。用户需要按照以下步骤配置 MPLAB Harmony v3 PLIB：

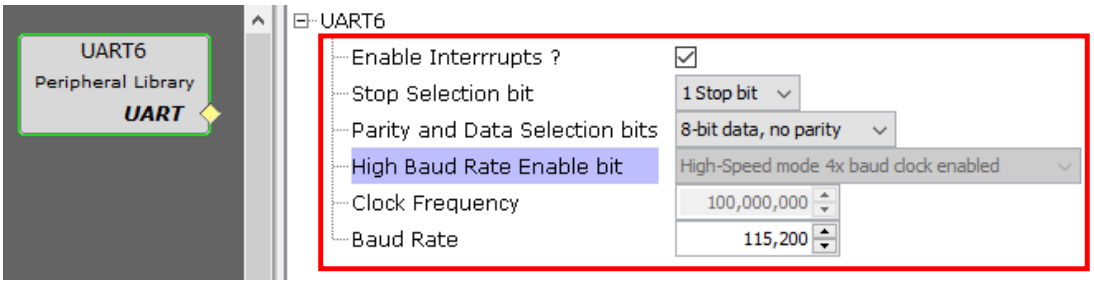
- 如上面的 MHC 部分所述，所有 MPLAB Harmony v3 组件（包括 PLIB）均在 Available Components 窗口中列出，如下图所示。PLIB 组件可能有多个实例，这些实例表示相应外设对应的多个硬件实例。例如，正在创建 MPLAB Harmony v3 项目的器件可能有六个 UART 外设，这些 UART 外设实例在 UART 内部列出，如下图所示。双击要使用 PLIB 的外设实例。此操作会将相应的 PLIB 组件置于项目图上。

  图 3-1. 可用组件——外设



- 将 PLIB 组件添加到项目图上后，单击选中此组件。此操作将使其配置选项显示在右侧的配置树中，如下图所示。根据应用的要求配置 PLIB。

图 3-2. 外设配置



- 完成 PLIB 配置后，将生成代码并开始使用 PLIB API 开发应用程序。已完成 PLIB 配置的代码在`<plib name>_Initilize` 函数内部生成，此函数是从 `initilization.c` 文件的 `SYS_Initialize()` 函数自动调用的。

3.2 了解 MPLAB Harmony v3 PLIB 生成的代码

MPLAB Harmony v3 项目中会生成以下文件和文件夹：

- **peripheral**：*peripheral* 文件夹中将列出项目中使用的所有外设的源文件和头文件。通常，每个外设实例将有一个 `.c` 文件和一个 `.h` PLIB 文件。某些外设及其文件始终默认添加到项目中，例如 `gpio`、`clock` 和 `evic`。
- **definitions.h**: 此文件类似于 MPLAB Harmony v2 的 `system_definitions.h` 文件，其中包含应用程序所需的所有头文件。
- **toolchain_specifics.h**：此文件定义了工具链特定的一些宏。
- **device.h**：此文件包含头文件，这些头文件具有所选器件特定的定义，例如 `xc.h`。
- **exception.c**：此文件与 MPLAB Harmony v2 的 `system_exceptions.c` 文件相同，用于定义应用程序的异常处理程序。
- **initialization.c**：此文件与 MPLAB Harmony v2 的 `system_init.c` 文件相同，用于初始化应用程序中使用的所有 MPLAB Harmony 模块。
- **interrupts.c**：此文件与 MPLAB Harmony v2 的 `system_interrupt.c` 文件相同，其中包含应用程序中使用的所有中断服务程序的定义。
- **stdio/xc32_monitor.c**：MPLAB Harmony v3 中添加了一项新功能，以支持标准 C 库的 `printf` 和 `scanf` 函数。当使用 STDIO 模块（在 Available Components 窗口的 Tools 中列出）时，MHC 将更新此文件中的函数定义，以便 `printf` 和 `scanf` 函数能够正常工作。
- **main.c**：此文件与 MPLAB Harmony v2 中的相同。但是，在 MPLAB Harmony v2 中，建议在 `app.c` 中开发应用程序。在仅使用 MPLAB Harmony v3 PLIB 的项目中，建议在 `main.c` 中进行此操作。仍然建议在 `app.c` 中开发基于 MPLAB Harmony v3 驱动程序的应用程序。

3.3 MPLAB Harmony v2 PLIB 和 MPLAB Harmony v3 PLIB 的差异

MPLAB Harmony v2 PLIB 和 MPLAB Harmony v3 PLIB 之间的主要差异如下：

- MPLAB Harmony v2 PLIB 没有任何 MHC 配置选项。PLIB 初始化代码必须手动编写。但是，MPLAB Harmony v3 为 PLIB 提供 MHC 配置选项，并根据配置生成 PLIB 初始化代码。
- MPLAB Harmony v2 PLIB 具有读取和写入外设模块几乎每个寄存器位域的 API，每个模块需要大量 API。尽管这种做法确保了灵活性，但它需要用户调用多个 API（有时甚至是 API 序列）来执行任务。相比之下，MPLAB Harmony v3 抽象了外设功能，并具有用于执行有意义任务的直接 API。它负责处理多个寄存器位域并在实现中进行排序，从而使应用变得简单快速。
- MPLAB Harmony v2 PLIB 针对不同外设实例提供相同 API。它提供了一个参数来区分实例，从而使 API 签名保持不变。但是，MPLAB Harmony v3 则针对不同实例提供专用 API。

- MPLAB Harmony v2 提供器件特定的预编译库文件（.a 形式）和头文件来使用其 PLIB。基于优化级别，将使用库文件实现或头文件内联实现。但是，MPLAB Harmony v3 为每个 PLIB 实例生成并添加一个.c 文件，其中包含所提供接口的定义。

3.4 使用 MPLAB Harmony v2 PLIB 的应用程序示例

Harmony v2 PLIB 和 Harmony v3 PLIB 之间存在显著差异，因此，无法直接从基于 MPLAB Harmony v2 PLIB 的应用程序移植到基于 MPLAB Harmony v3 PLIB 的应用程序。移植步骤将随应用程序使用的 Harmony 模块而异。

一个简单的示例是在计算机控制台上显示 *Hello World* 消息，目的是比较使用 MPLAB Harmony v2 PLIB API 与使用 MPLAB Harmony v3 PLIB API 的应用程序开发步骤。

要使用 MPLAB Harmony v2 PLIB 创建应用程序，请按照以下步骤操作：

1. 创建 MPLAB Harmony v2 项目。
2. 使用 MHC 进行配置。
   2.1. 配置 UART 引脚：转到 MHC 的 Pin Setting 窗口并配置 UART 发送引脚。
3. 使用 MHC 生成代码。
4. 更新 app.c。
   4.1. 以下代码给出了开发应用程序所需的初始化代码。它将初始化 UART 外设。

```
UART_Initialize ()
{
    uint32_t clockSource;

    /*禁止 USART 模块以对其进行配置*/
    PLIB_USART_Disable (USART_ID_6);

    /*设置线路控制模式*/
    PLIB_USART_LineControlModeSelect(USART_ID_6, USART_8N1);

    /*我们将接收中断模式设置为在 FIFO 不为空时接收中断*/

PLIB_USART_InitializeOperation(USART_ID_6,USART_RECEIVE_FIFO_ONE_CHAR,USART_TRANS
MIT_FIFO_IDLE,USART_ENABLE_TX_RX_USED);

    /*获取 USART 时钟源值*/
    clockSource = SYS_CLK_PeripheralFrequencyGet ( CLK_BUS_PERIPHERAL_1 );

    /*设置波特率并使能 USART*/
    PLIB_USART_BaudSetAndEnable(USART_ID_6, clockSource, 9600);  /*所需的波特率值*/
}

// *****************************************************************************
// *****************************************************************************
// 部分：应用程序初始化和状态机函数
// *****************************************************************************
// *****************************************************************************

/*******************************************************************************
  函数：
    void APP_Initialize ( void )

  备注：
    请参见 app.h 中的原型。
 */

void APP_Initialize ( void )
{
    UART_Initialize ();
}
```

注： 在 MPLAB Harmony v2 中，用于 UART 外设的 PLIB 被命名为 USART，而不是 UART。所有 API 的前缀都是 PLIB_USART，而不是 PLIB_UART。

4.2. 以下代码包含应用程序逻辑，可在控制台上显示消息。

```
uint8_t count = 0;
uint8_t consoleMsg[] = "Hello World\n\r";

void APP_Tasks ( void )
{
    if (count < sizeof(consoleMsg))
    {
        /*等待至 TX 缓冲区可用*/
        while(PLIB_USART_TransmitterBufferIsFull(USART_ID_6));
        /*发送一个字节*/
        PLIB_USART_TransmitterByteSend(USART_ID_6, consoleMsg[count]);
        count++;
    }
}
```

3.5 使用 MPLAB Harmony v3 PLIB 的应用程序示例

可以通过以下两种可能的方式，使用 MPLAB Harmony v3 PLIB 创建用于在控制台上显示消息的相同应用程序。

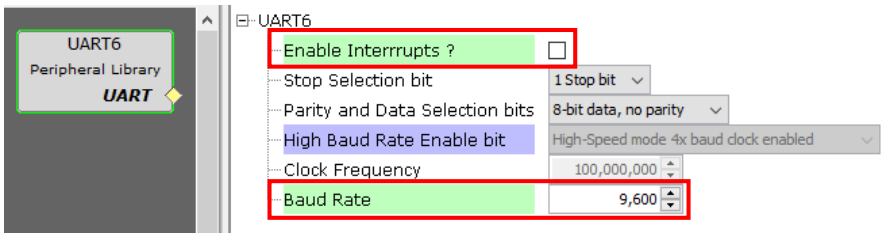
3.5.1 在阻塞模式下使用 MPLAB Harmony v3 PLIB 的应用程序示例

在阻塞模式下，PLIB 的中断被禁止。传输请求的 API 在阻塞模式下运行。由于 API 会一直阻塞到完成，因此不需要使用任何机制来检查传输状态。

在这种模式下，可以使用以下步骤创建应用程序：

1. 创建 MPLAB Harmony v3 项目。
2. 按照以下步骤使用 MHC 进行配置：
   2.1. 配置 UART 引脚：启动引脚管理器，转到 MHC 的 Pin Setting 窗口并配置 UART 发送引脚。
   2.2. 按照以下步骤配置 UART PLIB：
      2.2.1. 从 Available Components 列表中添加 UART6 外设库。
      2.2.2. 单击项目图上添加的 UART6 PLIB 框，配置选项将出现在右侧。
      2.2.3. 清除中断模式，并将波特率更改为 9600，如下图所示。

图 3-3. 配置 UART PLIB 1



3. 使用 MHC 生成代码。
4. 更新 `main.c` 文件。
   4.1. MHC 已根据上面步骤 2.2 中的配置生成了 UART 初始化代码，因此未编写代码来初始化 UART 外设。
   4.2. 以下以粗体显示的两行代码用于实现在控制台上显示消息的应用程序。

```
uint8_t consoleMsg[] = "Hello World\n\r";

int main ( void )
{
    /*初始化所有模块*/
    SYS_Initialize ( NULL );

    UART6_Write(&consoleMsg[0], sizeof(consoleMsg));

    while ( true )
    {
```

```
        /*维护所有已轮询 MPLAB Harmony 模块的状态机。*/
        SYS_Tasks ( );
    }

    /*正常运行期间不应执行*/

    return ( EXIT_FAILURE );
}
```

3.5.2 在非阻塞（中断）模式下使用 MPLAB Harmony v3 PLIB 的应用程序示例

在非阻塞模式下，将允许 PLIB 的中断，并且在传输完成之前提交传输请求的 API 不会阻塞。相反，会调用传输请求 API，此 API 触发传输过程，然后立即返回。传输在后台进行时，CPU 继续运行以下指令。但是，由于在此模式下允许了中断，因此在每次传输完成后，CPU 执行都会被中断（以执行中断服务程序），直到整个传输请求完成为止。在实现的中断模式下，可以通过以下两种方式检查传输状态。用户可以根据需要选择一种方法来检查传输状态。

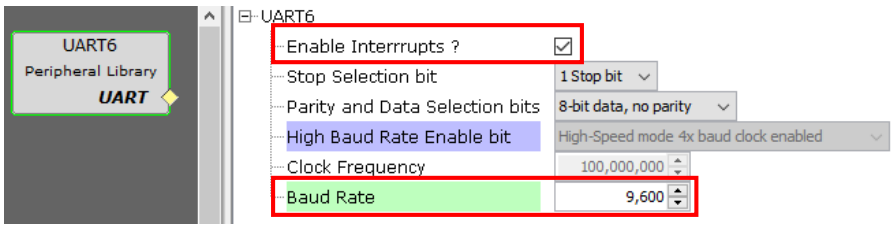
- 状态轮询：MPLAB Harmony v3 PLIB 提供一个 `IsBusy` API 来轮询传输状态。
- 回调：MPLAB Harmony v3 PLIB 提供一个回调注册 API 来注册回调。如果回调已注册，则在传输完成后 PLIB 会调用已注册的回调函数。

3.5.2.1 使用状态轮询的应用程序示例

请按照以下步骤创建示例应用程序，以使用状态轮询在非阻塞（中断）模式下通过 MPLAB Harmony v3 PLIB 在控制台上显示消息：

1. 创建 MPLAB Harmony v3 项目。
2. 使用 MHC 进行配置。
   2.1. 配置 UART 引脚：启动引脚管理器，转到 MHC 的 Pin Setting 窗口并配置 UART 发送引脚。
   2.2. 配置 UART PLIB：
      2.2.1. 从 Available Components 列表中添加 UART6 外设库。
      2.2.2. 单击项目图上添加的 UART6 PLIB 框，其配置选项将出现在窗口的右侧。
      2.2.3. 保持允许中断模式，并将波特率更改为 9600，如下图所示：

图 3-4. 配置 UART PLIB 2



3. 使用 MHC 生成代码。
4. 更新 `main.c` 文件。
   4.1. MHC 已根据上面步骤 2.2 中的配置生成了 UART 初始化代码，因此不需要代码来初始化 UART 外设。
   4.2. 使用以下代码更新 `main.c` 文件，以完成应用程序的实现过程：

```
uint8_t consoleMsg[] = "Hello World\n\r";

int main ( void )
{
    /*初始化所有模块*/
    SYS_Initialize ( NULL );

    UART6_Write(&consoleMsg[0], sizeof(consoleMsg));
    while (UART6_WriteIsBusy());

    while ( true )
    {
        /*维护所有已轮询 MPLAB Harmony 模块的状态机。*/
        SYS_Tasks ( );
    }
```

```
    /*正常运行期间不应执行*/

    return ( EXIT_FAILURE );
}
```

注： 非阻塞（中断）模式下的应用程序代码类似于阻塞模式下的应用程序代码。非阻塞模式还有一条额外的指令（在上文中以粗体显示），用于轮询传输状态。

3.5.2.2 使用回调的应用程序

在非阻塞方法中，还可以使用回调机制代替状态轮询来检查传输状态。按照以下步骤使用回调机制：

- 使用 PLIB 提供的专用 API 注册针对 PLIB 的回调函数。
- 定义已注册的函数。
- 提出传输请求。
- 执行其他应用程序任务。每当传输完成时，定义的已注册回调函数都将由 PLIB 调用。

由于回调是从中断现场调用的，因此在应用程序中定义回调函数时必须遵循以下准则：

- 处理方式必须与 ISR 相同
- 必须简短
- 禁止调用不具备中断安全性的应用程序函数
- 对于在回调函数内部和外部都可以访问的变量，请使用“易失性”关键字

未显示使用回调机制的代码示例，因为 MPLAB Harmony v2 PLIB 不具有此功能，因此没有可比较的内容。用户可以参考 `csp` 资源库中 `csp/apps` 文件夹下的 PLIB 演示来了解许多 PLIB 回调示例。

3.6 MPLAB Harmony v2 PLIB 示例和 MPLAB Harmony v3 PLIB 示例之间的比较

如果针对给定的示例比较 MPLAB Harmony v2 和 MPLAB Harmony v3 中的应用程序开发步骤，则会发现以下差异：

- MPLAB Harmony v3 多了一个 MHC 配置步骤 2.2
- MPLAB Harmony v3 少了一个编码步骤 4.1
- MPLAB Harmony v3 简化了应用程序逻辑步骤 4.2

前面的示例是一个非常简单的应用程序。随着应用程序复杂性的增加，与 MPLAB Harmony v2 相比，使用 MPLAB Harmony v3 PLIB 开发应用程序的差异将变得更加明显，而且会更加简单和用户友好。

4. 驱动程序

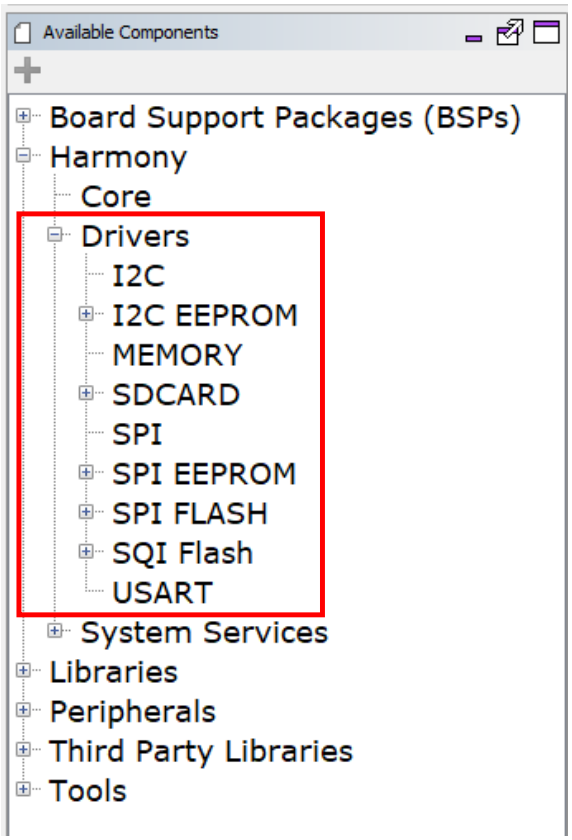
与 MPLAB Harmony v2 中一样，MPLAB Harmony v3 驱动程序还为外设和其他资源提供高度抽象的 C 语言接口。驱动程序的接口允许应用程序和其他客户端模块（即中间件库）与其控制的外设交互。

4.1 MPLAB Harmony v3 驱动程序使用入门

MPLAB Harmony v3 驱动程序是 MPLAB Harmony Core Service 的一部分，可以按照工具和安装部分中所述从 MPLAB Harmony GitHub 页面下载（克隆）core 资源库来使用它们。请按照以下步骤配置和使用 MPLAB Harmony v3 驱动程序：

- 如上面的 MHC 部分所述，所有 MPLAB Harmony v3 组件（包括驱动程序）都在 Available Components 窗口中列出，如下图所示。要添加驱动程序组件，请双击该驱动程序或将其拖放到 Project Graph 中。

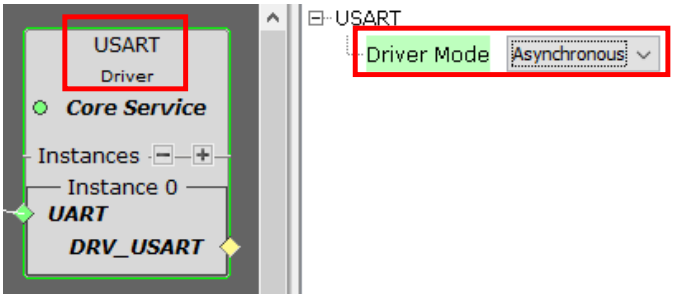
  图 4-1. 可用组件——驱动程序



- 如上面的 MHC 部分所述，MPLAB Harmony v3 驱动程序具有必须得到满足的依赖关系。这些依赖关系通常由 PLIB 或系统服务组件满足。应满足所有依赖关系。
- （在配置树中）配置用于满足依赖关系的所有组件。例如，USART 驱动程序将依赖于 UART PLIB，因此必须配置 UART PLIB。

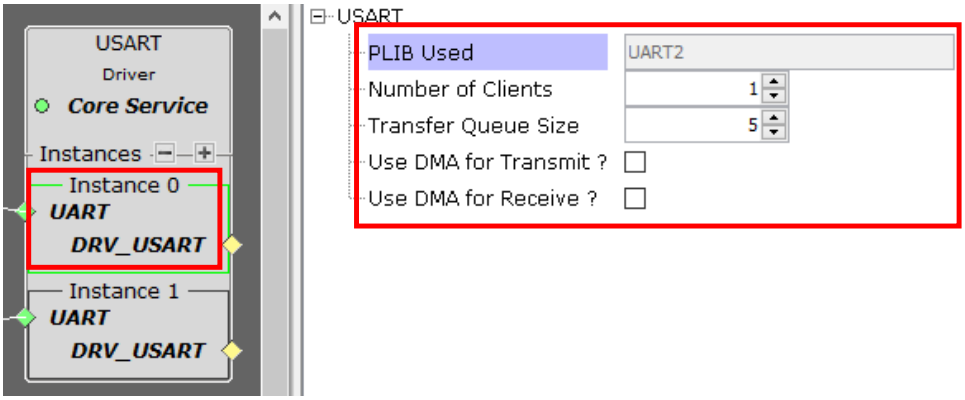
  注： 在 MPLAB Harmony（MPLAB Harmony v2 和 MPLAB Harmony v3）中，与 UART 和 USART 相对应的驱动程序统称为 USART 驱动程序。
- 驱动程序可以有多个实例。很少有配置选项通用于所有实例，同样，也很少有配置选项是实例特定的。因此，驱动程序有两个配置步骤：
  1. 通过单击项目图上驱动程序框的上部，在配置树中配置通用选项，如下图所示：

图 4-2. 驱动程序的通用配置



2. 默认情况下，驱动程序有一个实例（实例编号 0）。单击**+**号可以增加实例数，单击**-**号可以减少实例数。通过单击项目图上的相应实例框来配置实例特定的选项，如下图所示：

图 4-3. 驱动程序的实例特定配置



- 完成配置后，将生成代码并开始使用驱动程序 API 开发应用程序。

4.2 了解 MPLAB Harmony v3 驱动程序代码

与仅使用 PLIB 的项目相比，使用 MPLAB Harmony v3 驱动程序或系统服务时，会生成以下附加文件和文件夹：

- **driver**：驱动程序文件夹中将列出项目中使用的所有驱动程序的源文件和头文件。通常，使用的每个驱动程序将有一个`.c`文件和两个`.h`文件。
- **configuration.h**：此文件类似于 MPLAB Harmony v2 的`system_config.h`文件，用于定义驱动程序、系统服务和中间件库的配置宏。
- **user.h**：这是 MPLAB Harmony v3 中的一个新文件，默认情况下为空，用于定义应用程序特定的宏。在 MPLAB Harmony v2 中，通常在`system_config.h`文件中定义应用程序特定的宏。
- **app.h**：此文件与 MPLAB Harmony v2 中的文件相同，用于开发应用程序。
- **app.c**：此文件与 MPLAB Harmony v2 中的文件相同，用于开发应用程序。

4.3 MPLAB Harmony v2 和 MPLAB Harmony v3 驱动程序的相似之处

4.3.1 惟一接口

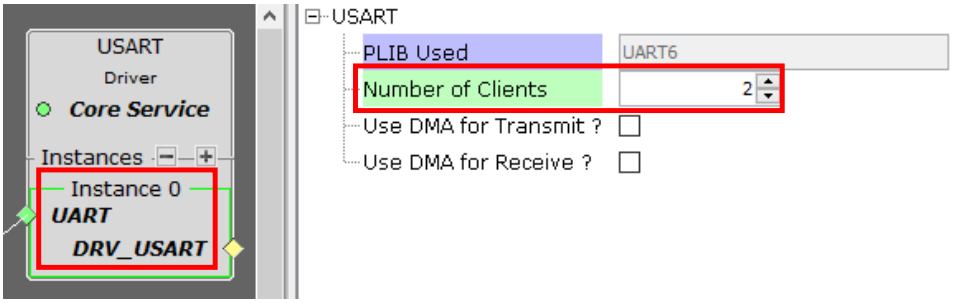
与 MPLAB Harmony v2 中一样，MPLAB Harmony v3 驱动程序也提供了惟一 API，以供 Microchip 32 位器件的外设使用。凭借 32 位 MCU 系列的这些惟一 API 接口，可以轻松地将使用 MPLAB Harmony 驱动程序开发的应用程序从一个器件移植到另一个器件。有关这些 API 的详细信息，请参见 core 资源库中的帮助文档。

4.3.2 多客户端支持

与 MPLAB Harmony v2 驱动程序中一样，MPLAB Harmony v3 驱动程序也支持多个客户端。这样应用程序便可在不同的上下文中使用具有不同配置的外设实例，而不存在任何数据干扰。

在 MPLAB Harmony v3 中，可以通过 MHC 配置应用程序要使用的最大客户端数量。可以在 MPLAB Harmony 驱动程序的外设特定驱动程序实例的配置选项下完成此操作，如下图所示：

图 4-4. 多客户端配置

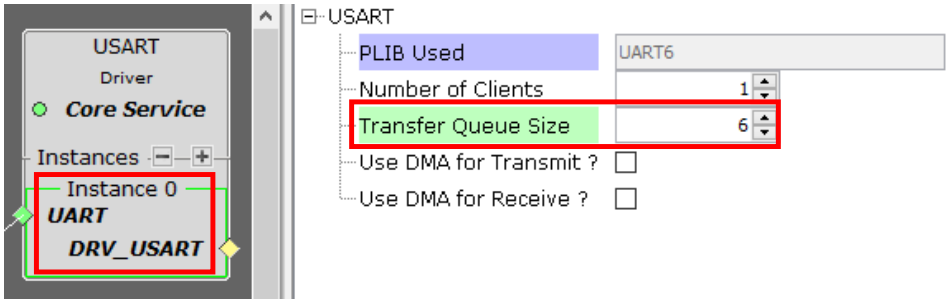


4.3.3 缓冲区队列支持

与 MPLAB Harmony v2 驱动程序中一样，MPLAB Harmony v3 驱动程序也支持缓冲区排队。驱动程序可以在处理旧请求过程中对客户端的新请求进行排队。在 MPLAB Harmony v3 中，只有驱动程序的异步模式支持排队。同步驱动程序本质上是阻塞的，因此不支持排队功能（下一节将讨论 MPLAB Harmony v3 中的同步和异步驱动程序）。

在 MPLAB Harmony v3 中，要配置缓冲区队列大小，先通过驱动程序的通用配置中所示驱动程序的通用配置将驱动程序配置为异步模式，然后通过外设特定的驱动程序实例配置来配置传输队列大小，如下图所示：

图 4-5. 配置队列大小



4.3.4 实时操作系统（RTOS）支持

与 MPLAB Harmony v2 驱动程序一样，MPLAB Harmony v3 驱动程序也支持多个 RTOS。

4.4 MPLAB Harmony v2 和 MPLAB Harmony v3 驱动程序的差异

以下是 MPLAB Harmony v2 和 MPLAB Harmony v3 驱动程序之间的主要差异：

4.4.1 API 兼容性

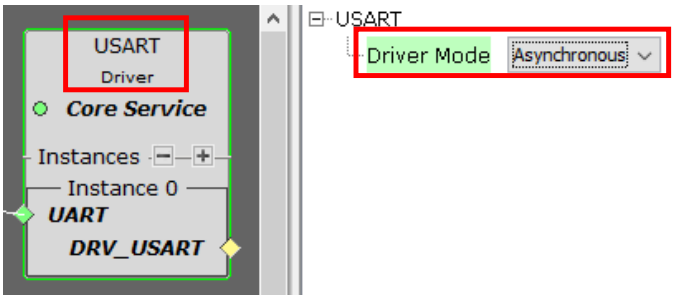
MPLAB Harmony v2 和 MPLAB Harmony v3 驱动程序 API 几乎没有差异。由于差异很小，因此基于 MPLAB Harmony v2 驱动程序的应用程序略微更改后便可移植到 MPLAB Harmony v3 应用程序。

4.4.2 同步和异步模型

同步驱动程序本质上是阻塞的，而异步驱动程序本质上是非阻塞的。建议将同步驱动程序与 RTOS 一起使用，而异步驱动程序则在裸机（Non-RTOS）环境中使用。尽管建议在基于非 RTOS 的应用程序环境中使用异步驱动程序，但也可以在基于 RTOS 的应用程序环境中使用异步驱动程序，在这种情况下，应用程序必须确保特定任务可交出控制权以允许相关应用程序任务正常运行。

MPLAB Harmony v3 提供同步和异步驱动程序，其中 MPLAB Harmony v2 仅提供驱动程序的异步模型。在 MPLAB Harmony v3 中，可以通过在项目图中单击驱动程序框的上半部分，在配置选项中完成此模式选择，如下图所示：

图 4-6. 配置驱动程序模式



4.4.3 中断和轮询模式

在驱动程序的中断模式下，驱动程序状态机（或任务程序）从中断服务程序运行。但是，在轮询模式下，驱动程序状态机从 `while(1)` 循环运行。驱动程序的中断模式提供最佳的响应能力，而轮询模式适合调试。

MPLAB Harmony v2 支持驱动程序的中断模式和轮询模式。但是，MPLAB Harmony v3 仅支持中断模式（一些与外设中断无关且从 `while(1)` 循环运行其状态机的驱动程序除外，例如存储器驱动程序和 SDSPI 驱动程序）。如果 MPLAB Harmony v2 应用程序使用驱动程序的轮询模式，则在 MPLAB Harmony v3 中必须使用驱动程序的中断模式。但是，从轮询模式切换为中断模式并不需要对应用程序进行任何更改。系统和中断文件中需要进行的任何更改均由 MHC 代码生成完成。

4.4.4 静态和动态模型

MPLAB Harmony v2 静态驱动程序针对单个外设实例实现，以减少驱动程序占用的存储空间。有时，驱动程序的其他功能（例如多客户端、缓冲区排队和 RTOS 支持）也会从静态驱动程序中删除，以使驱动程序更简洁。动态驱动程序是完备的驱动程序，支持多个外设实例、多客户端和 RTOS。

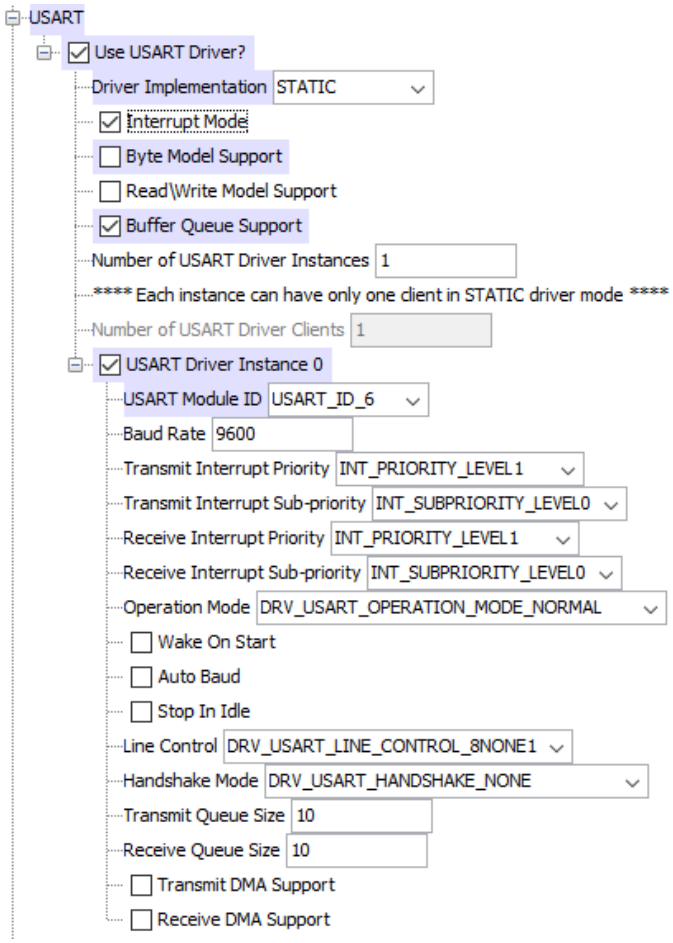
MPLAB Harmony v2 支持静态和动态驱动程序模型，而 MPLAB Harmony v3 仅具有动态驱动程序。但是，MPLAB Harmony v3 PLIB 提供 MPLAB Harmony v2 静态驱动程序的许多特性。使用 MPLAB Harmony v2 静态驱动程序开发的应用程序可以根据其要求切换到 MPLAB Harmony v3 PLIB 或 MPLAB Harmony v3 驱动程序（动态驱动程序）。如果应用程序需要多客户端、缓冲区排队和 RTOS 等功能，则可以使用 MPLAB Harmony v3 动态驱动程序；其他情况下，可以考虑使用 MPLAB Harmony v3 PLIB。

4.5 使用 MPLAB Harmony v2 驱动程序的应用程序示例

要使用 MPLAB Harmony v2 驱动程序创建在计算机控制台上显示消息的应用程序，请按照以下步骤操作：

1. 创建 MPLAB Harmony v2 项目。
2. 使用 MHC 进行配置。
   2.1. 配置 UART 引脚：转到 MHC 的 Pin Setting 窗口并配置 UART 发送引脚。
   2.2. 按下图所示配置 USART 驱动程序（注意，突出显示的选项已修改）：
   注： 在 MPLAB Harmony 中，与 UART 和 USART 相对应的驱动程序是 USART 驱动程序。

图 4-7. 配置 MPLAB Harmony v2 USART 驱动程序



3. 使用 MHC 生成代码。
4. 按照以下步骤更新应用程序：
   4.1. MHC 已根据步骤 2.2 中所述的配置生成了 UART 初始化代码，因此不需要添加代码。
   4.2. 使用应用程序逻辑更新 `app.c` 文件和 `app.h` 文件。下图给出了要在 `app.c` 文件中开发的应用程序逻辑，此逻辑具有以下三种状态：
      4.2.1. 打开驱动程序：对于支持多个客户端的驱动程序模型，此状态是必需的。例如，动态驱动程序。静态驱动程序可跳过此状态，因为它们支持单个客户端。
      4.2.2. 队列传输请求：此状态将传输请求添加到队列中。在此应用程序中，没有需要排队的连续请求，但使用了排队模型进行表示。
      4.2.3. 检查传输状态：此状态用于检查传输状态。可通过以下方法检查传输状态：
         4.2.3.1. 轮询：应用程序使用 API 连续轮询传输状态。本示例中使用了轮询方法。

4.2.3.2. 回调：可以使用可在传输完成后调用的专用 API 注册回调。

```
uint8_t consoleMsg[] = "Hello World\n\r";
void APP_Tasks ( void )
{

    /*检查应用程序的当前状态。*/
    switch ( appData.state )
    {
        /*应用程序的初始状态。*/
        case APP_STATE_OPEN_DRIVER:
        {
            appData.myUSARTHandle = DRV_USART_Open(DRV_USART_INDEX_0, DRV_IO_INTENT_READWRITE|
DRV_IO_INTENT_NONBLOCKING);
            if (appData.myUSARTHandle != DRV_HANDLE_INVALID)
            {
                appData.state = APP_STATE_ADD_REQUEST;
            }
            break;
        }
        case APP_STATE_ADD_REQUEST:
        {
            DRV_USART_BufferAddWrite(appData.myUSARTHandle, &appData.bufferHandle,
&consoleMsg[0], sizeof(consoleMsg));
            appData.state = APP_STATE_STATUS_CHECK;
            break;
        }
        case APP_STATE_STATUS_CHECK:
        {
            if (DRV_USART_TRANSFER_STATUS_TRANSMIT_EMPTY &
DRV_USART_TransferStatus(appData.myUSARTHandle))
            {
                //数据已发送，进入下一个状态
                appData.state = APP_STATE_COMPLETE;
            }
            break;
        }
```

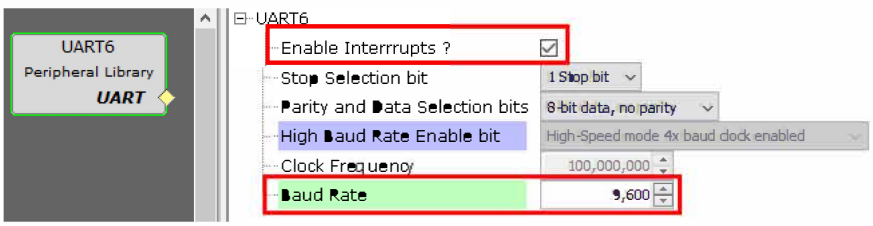
注： 如果用户要为此应用程序使用动态驱动程序或驱动程序的非中断模式，则 app.c 文件中的应用程序代码必须相同。惟一的更改是 MHC 配置更改，如步骤 2.2 中所述。

4.6 使用 MPLAB Harmony v3 驱动程序的应用程序示例

要使用 MPLAB Harmony v3 驱动程序创建在计算机控制台上显示消息的应用程序，请按照以下步骤操作：

1. 创建 MPLAB Harmony v3 项目。
2. 使用 MHC 进行配置。
   - 2.1. 配置 UART 引脚：启动引脚管理器，转到 MHC 的 Pin Setting 窗口并配置 UART 发送引脚。
   - 2.2. 按照以下步骤配置 UART PLIB：
     - 2.2.1. 从 Available Components 窗口中添加 UART6 外设库。
     - 2.2.2. 单击项目图上添加的 UART6 PLIB 框，其配置选项将出现在窗口的右侧。
     - 2.2.3. 保持允许中断模式，并将波特率更改为 9600，如下图所示：

图 4-8. 配置 UART PLIB



   - 2.3. 按照以下步骤配置 USART 驱动程序：

2.3.1. 从 Available Components 窗口中添加 USART 驱动程序。

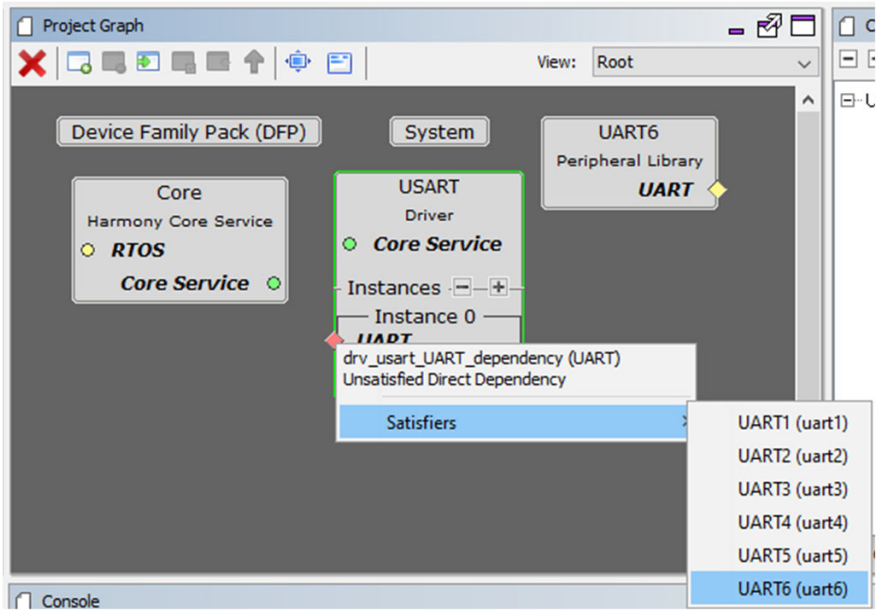
2.3.2. 在弹出窗口中询问时添加 Core（HarmonyCore）组件。

2.3.3. 请勿在弹出窗口中询问时添加 FreeRTOS 组件。

2.3.4. 右键单击 USART 驱动程序实例 0 依赖关系框，并使用 UART6 PLIB 满足依赖关系，如下图所示：

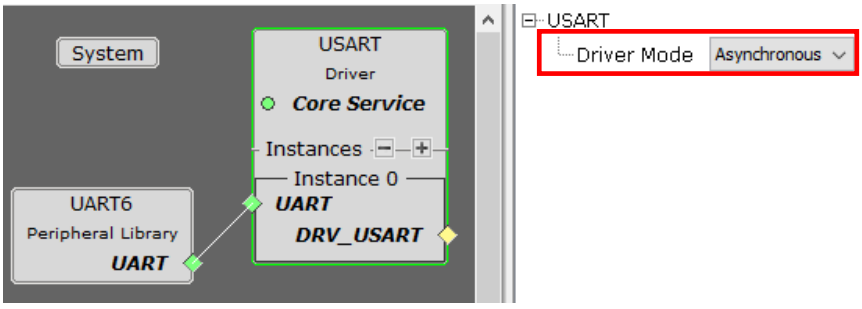
注： 在 MPLAB Harmony 中，与 UART 和 USART 相对应的驱动程序是 USART 驱动程序。

图 4-9. 配置 MPLAB Harmony v3 USART 驱动程序 1



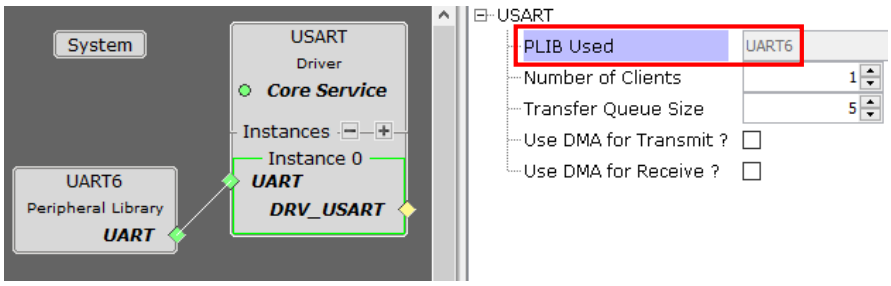
2.3.5. 单击项目图上 USART 驱动程序框的上部，USART 驱动程序通用配置选项将出现在窗口的右侧。将 Driver Mode（驱动程序模式）设置为 Asynchronous（异步），如下图所示：

图 4-10. 配置 MPLAB Harmony v3 USART 驱动程序 2



2.3.6. 单击项目图上 USART 驱动程序框的下部，USART 驱动程序实例 0 配置选项将出现在窗口的右侧。保持配置不变。请注意，“PLIB Used”（使用的 PLIB）自动配置为 UART6。如果未显示 UART6，则意味着 USART 驱动程序尚未与 UART6 PLIB 连接，需要建立此连接。

图 4-11. 配置 MPLAB Harmony v3 USART 驱动程序 3



3. 使用 MHC 生成代码。
4. 更新应用程序：
   - 4.1. MHC 已根据步骤 2.2 和 2.3 中所述的配置生成了 UART PLIB 和驱动程序初始化代码。
   - 4.2. 使用应用程序逻辑更新 `app.c` 文件和 `app.h` 文件。下图给出了要在 `app.c` 文件中开发的应用程序逻辑，此逻辑具有以下三种状态：
     - 4.2.1. 打开驱动程序：对于 MPLAB Harmony v3 驱动程序，此状态是必需的，因为 v3 驱动程序支持多客户端。这意味着即使驱动程序只有一个客户端，应用程序也需要在使用驱动程序之前将其打开。
     - 4.2.2. 队列传输请求：此状态将传输请求添加到队列中。
     - 4.2.3. 检查传输状态：此状态用于检查传输状态。与 MPLAB Harmony v2 一样，可以通过以下方法检查传输状态：
       - 4.2.3.1. 轮询：应用程序使用 API 连续轮询传输状态。本示例中使用了轮询方法。
       - 4.2.3.2. 回调：可以使用可在传输完成后调用的专用 API 注册回调。

```
uint8_t consoleMsg[] = "Hello World\n\r";
void APP_Tasks ( void )
{

    /*检查应用程序的当前状态。*/
    switch ( appData.state )
    {
        /*应用程序的初始状态。*/
        case APP_STATE_OPEN_DRIVER:
        {
            appData.myUSARTHandle = DRV_USART_Open(DRV_USART_INDEX_0, DRV_IO_INTENT_READWRITE|
DRV_IO_INTENT_NONBLOCKING);
            if (appData.myUSARTHandle != DRV_HANDLE_INVALID)
            {
                appData.state = APP_STATE_ADD_REQUEST;
            }
            break;
        }
        case APP_STATE_ADD_REQUEST:
        {
            DRV_USART_WriteBufferAdd(appData.myUSARTHandle, &consoleMsg[0],
sizeof(consoleMsg), &appData.bufferHandle);
            appData.state = APP_STATE_STATUS_CHECK;
            break;
        }
        case APP_STATE_STATUS_CHECK:
        {
            if (DRV_USART_BUFFER_EVENT_COMPLETE &
DRV_USART_BufferStatusGet(appData.bufferHandle))
            {
                //数据已发送，进入下一个状态
                appData.state = APP_STATE_COMPLETE;
            }
            break;
        }
```

4.6.1 MPLAB Harmony v2 驱动程序示例和 MPLAB Harmony v3 驱动程序示例之间的比较

MPLAB Harmony v2 和 MPLAB Harmony v3 中的 MHC 配置项相似。MPLAB Harmony v3 需要分两部分进行配置（如步骤 2.2 和 2.3 所示），但是在 MPLAB Harmony v2 中只需一步（2.2）即可配置相同的配置项。

MPLAB Harmony v2 和 MPLAB Harmony v3 具有相似的驱动程序使用模型和应用程序逻辑，但 MPLAB Harmony v3 中的 API 有所更改。前面的示例代码中突出显示了 API 更改。

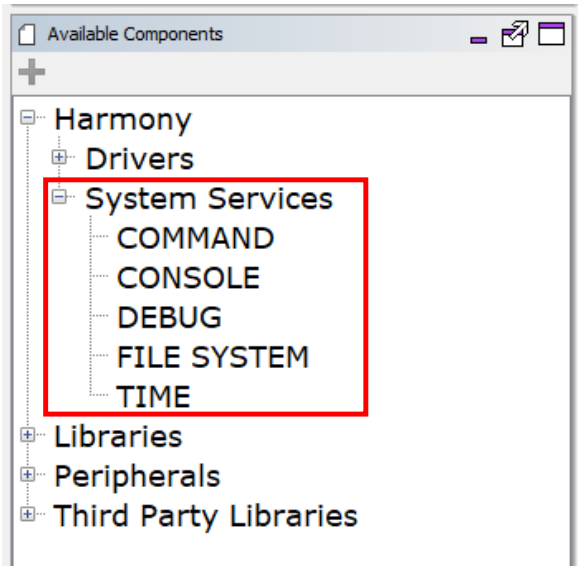
5. 系统服务

MPLAB Harmony 提供系统服务库来支持通用功能和管理多个驱动程序、库和其他模块共用的资源。MPLAB Harmony v3 系统服务类似于 MPLAB Harmony v2 系统服务。

5.1 MPLAB Harmony v3 系统服务使用入门

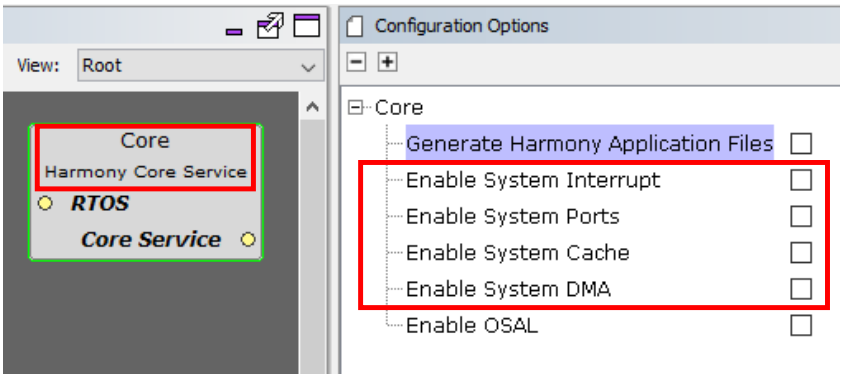
MPLAB Harmony v3 系统服务也是 Harmony Core Service 的一部分，可以按照“工具和安装”部分所述从 MPLAB Harmony GitHub 页面下载（克隆）core 资源库来使用它们。如上文的 MHC 部分所述，所有 MPLAB Harmony v3 组件（包括系统服务）都在 Available Components 窗口中列出，如下图所示。用户可以双击它们将其放置在项目图上以进行配置和使用。

图 5-1. 可用组件——系统服务



可以从 core 组件（Harmony Core Service）的配置树中配置驱动程序、中间件库和应用程序常用的一些系统服务（端口系统服务和中断系统服务等），如下图所示。如果 MPLAB Harmony v3 组件需要这些系统服务，则将自动选择这些选项。仅当默认情况下未选中而应用程序需要使用这些系统服务时，用户才需要手动选择它们。

图 5-2. 常用系统服务



6. 中间件库

MPLAB Harmony v3 具有与 MPLAB Harmony v2 相同的中间件库。中间件的实现已更新为使用最新的驱动程序和系统服务，但 API 将保持不变。要获得 MPLAB Harmony v3 Available Components 列表中列出的中间件，用户必须按照工具和安装部分中的说明从 GitHub 下载（克隆）相应的中间件资源库。

7. 实时操作系统（RTOS）支持

与 MPLAB Harmony v2 中一样，MPLAB Harmony v3 驱动程序、系统服务和中间件通过操作系统抽象层（Operating System Abstraction Layer，OSAL）支持多个第三方 RTOS。OSAL 提供了一个一致的接口，支持兼容 MPLAB Harmony 的库在运行于 OS 环境下或独自运行时利用操作系统结构。MPLAB Harmony v3 中的 OSAL 层与 MPLAB Harmony v2 中的相同。

7.1 MPLAB Harmony v3 中的 RTOS 使用入门

要在 MPLAB Harmony v3 中使用任何 RTOS，需要两个资源库（对于每个 RTOS）：

- Harmony 配置资源库：此资源库将包含 RTOS 的 MPLAB Harmony 配置文件和 MPLAB Harmony 应用程序。它由 Microchip 提供，可以从 GitHub 下载，如“工具和安装”部分所述。
- RTOS 源代码资源库：此资源库将包含 RTOS 的源代码，必须从相应的第三方供应商处获取。

下表列出了当前支持的 RTOS 及其在 MPLAB Harmony v3 中使用时所必需的资源库：

表 7-1. 支持的 RTOS 和相应的资源库

RTOS 名称	MPLAB Harmony 配置资源库	RTOS 源代码资源库
FreeRTOS	core	FreeRTOS
Micrium ucos3	micrium_ucos3	需要从供应商处获取
Thread-X	expresslogic_threadx	需要从供应商处获取

8. 将 MPLAB Harmony v2 应用程序移植到 MPLAB Harmony v3

需要在 MPLAB Harmony v3 上开发的 MPLAB Harmony v2 应用程序可能正在使用多个组件，例如 PLIB、驱动程序和中间件。下面总结了在 MPLAB Harmony v3 中开发应用程序时使用的组件：

8.1 PLIB

如果应用程序使用 MPLAB Harmony v2 PLIB，则可以使用 MPLAB Harmony v3 PLIB 进行开发，但是不能直接使用，因为 MPLAB Harmony v3 PLIB 与 MPLAB Harmony v2 PLIB 不同。有关更多信息，请参见使用 MPLAB Harmony v2 PLIB 的应用程序示例和使用 MPLAB Harmony v3 PLIB 的应用程序示例。

8.2 静态驱动程序

如果 MPLAB Harmony v2 应用程序使用静态驱动程序，则可以通过以下两种方法在 MPLAB Harmony v3 上进行开发：

1. 可以使用 MPLAB Harmony v3 PLIB 代替 MPLAB Harmony v2 静态驱动程序。使用 MPLAB Harmony v3 PLIB 的应用程序示例给出了相应的示例。
2. 可以使用 MPLAB Harmony v3 动态驱动程序代替 MPLAB Harmony v2 静态驱动程序。使用 MPLAB Harmony v3 驱动程序的应用程序示例给出了相应的示例。

8.3 动态驱动程序

如果 MPLAB Harmony v2 应用程序使用动态驱动程序，则可以使用 MPLAB Harmony v3 动态驱动程序在 MPLAB Harmony v3 上对该应用程序进行开发。使用 MPLAB Harmony 3 驱动程序的应用程序示例给出了相应的示例。

8.4 系统服务

如果 MPLAB Harmony v2 应用程序使用系统服务，则可以使用 MPLAB Harmony v3 系统服务在 MPLAB Harmony v3 上对该应用程序进行开发。这里没有给出示例，因为这是一个简单的更改。有关 MPLAB Harmony v3 系统服务的详细信息，请参见系统服务。

8.5 中间件

MPLAB Harmony v3 中间件库与 MPLAB Harmony v2 中的相同。移植基于中间件的应用程序不需要任何 API 级别的更改。但是，MPLAB Harmony v3 中的 MHC 配置选项和样式发生了更改，因此必须注意。可以参考以下移植文档来开发基于 MPLAB Harmony v3 中间件的应用程序：

USB：

https://github.com/Microchip-MPLAB-Harmony/usb/wiki/MPLAB-Harmony-2-to-Harmony-3-USB-Application-Migration-Guide

TCP/IP：

https://github.com/Microchip-MPLAB-Harmony/net/wiki/H2-to-H3-Migration

图形：

https://github.com/Microchip-MPLAB-Harmony/gfx/wiki/Migrate-aria_quickstart-v2.06-to-3.04-pic32mz_ef_sk_meb2

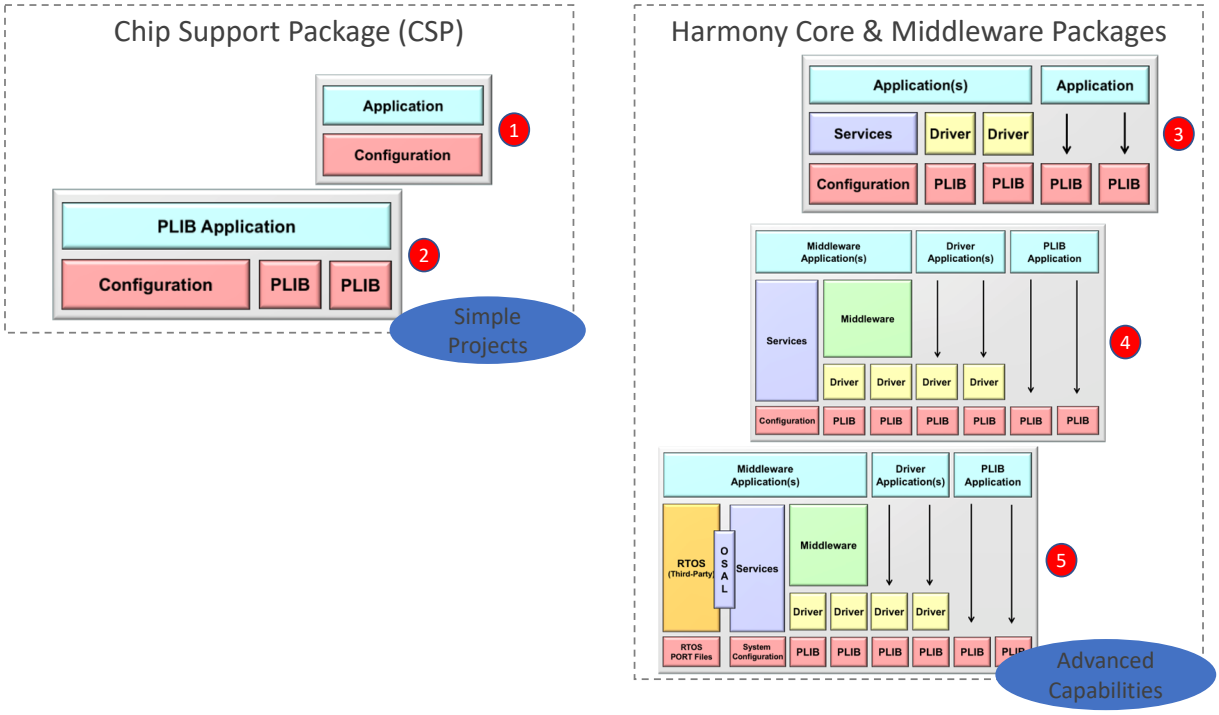
9. MPLAB Harmony 开发模型

MPLAB Harmony 架构可实现从小型实时应用程序到功能丰富的大型应用程序的各种应用程序。如前面的章节所述，可以使用各种 MPLAB Harmony 组件（模块）来开发这些应用程序。以下开发模型是基于应用程序开发中使用的 MPLAB Harmony 组件获得的：

- 使用 MHC 进行简单的器件配置和初始化。
- 基于外设库的应用程序。
- 基于驱动程序的强大、无冲突的应用程序。
- 需要 MPLAB Harmony 中间件的应用程序。
- 可实现最佳中央处理单元（Central Processing Unit，CPU）利用率的基于 RTOS 的应用程序。

下图给出了在不同开发模型中使用的 MPLAB Harmony 组件。前两个模型用于简单的应用程序，仅需要下载（克隆）`csp` 资源库（以及来自“dev_packs”资源库的器件包支持）。后三个模型适用于需要 `core` 和其他资源库的高级应用程序。

图 9-1. MPLAB Harmony v3 开发模型



MPLAB Harmony v2 和 MPLAB Harmony v3 均支持全部五个模型。但是，与 MPLAB Harmony v2 相比，在 MPLAB Harmony v3 中使用第二个模型开发应用程序更加容易。

10. 结论

MPLAB Harmony v3 通过 GitHub 提供模块化下载和更新，以更好地管理安装和配置。它使用 MPLAB Harmony 配置器（MHC）的 Project Graph 窗口来改善 MPLAB Harmony 模块的配置方式。MPLAB Harmony v3 提供了简单易用且经过优化的外设库，可快速开发应用程序。对于使用 MPLAB Harmony v2 驱动程序、系统服务和中间件库的应用程序，只需更改少量代码即可移植到 MPLAB Harmony v3。

11. 参考资料

- 有关 MPLAB Harmony 3 的更多信息，请访问 Microchip 网站：
  - https://www.microchip.com/mplab/mplab-harmony
  - https://microchipdeveloper.com/harmony3:start
- 如何设置 MPLAB Harmony v3 软件开发框架：
  - http://ww1.microchip.com/downloads/en/DeviceDoc/How_to%20_Setup_MPLAB_%20Harmonyv3_%20Software_%20Development_Framework_DS90003232A.pdf
- 有关各种 MPLAB Harmony 3 组件的详细文档，请参见相应资源库的文档文件夹。

Microchip 网站

Microchip 网站（www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术规范。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品非常安全。
- 目前，仍存在着用恶意、甚至是非法的方法来试图破坏代码保护功能的行为。我们确信，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这种试图破坏代码保护功能的行为极可能侵犯 Microchip 的知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

本出版物中提供的信息仅仅是为方便您使用 Microchip 产品或使用这些产品来进行设计。本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

MICROCHIP“按原样”提供这些信息。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对非侵权性、适销性和特定用途的适用性的暗示担保，或针对其使用情况、质量或性能的担保。

在任何情况下，对于因这些信息或使用这些信息而产生的任何间接的、特殊的、惩罚性的、偶然的或间接的损失、损害或任何类型的开销，MICROCHIP 概不承担任何责任，即使 MICROCHIP 已被告知可能发生损害或损害可以预见。在法律允许的最大范围内，对于因这些信息或使用这些信息而产生的所有索赔，MICROCHIP 在任何情况下所承担的全部责任均不超出您为获得这些信息向 MICROCHIP 直接支付的金额（如有）。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切损害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任。除非另外声明，在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、Microchip 徽标、Adaptec、AnyRate、AVR、AVR 徽标、AVR Freaks、BesTime、BitCloud、chipKIT、chipKIT 徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、HELDO、IGLOO、JukeBlox、KeeLoq、Kleer、LANCheck、LinkMD、maXStylus、maXTouch、MediaLB、megaAVR、Microsemi、Microsemi 徽标、MOST、MOST 徽标、MPLAB、OptoLyzer、PackeTime、PIC、picoPower、PICSTART、PIC32 徽标、PolarFire、Prochip Designer、QTouch、SAM-BA、SenGenuity、SpyNIC、SST、SST 徽标、SuperFlash、Symmetricom、SyncServer、Tachyon、TempTrackr、TimeSource、tinyAVR、UNI/O、Vectron 和 XMEGA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

APT、ClockWorks、The Embedded Control Solutions Company、EtherSynch、FlashTec、Hyper Speed Control、HyperLight Load、IntelliMOS、Libero、motorBench、mTouch、Powermite 3、Precision Edge、ProASIC、ProASIC Plus、ProASIC Plus 徽标、Quiet-Wire、SmartFusion、SyncWorld、Temux、TimeCesium、TimeHub、TimePictra、TimeProvider、Vite、WinPath 和 ZL 均为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BlueSky、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet 徽标、memBrain、Mindi、MiWi、MPASM、MPF、MPLAB Certified 徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA 和 ZENA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的商标。

SQTP 是 Microchip Technology Incorporated 在美国的服务标记。

Adaptec 徽标、Frequency on Demand、Silicon Storage Technology 和 Symmcom 为 Microchip Technology Inc.在其他国家或地区的注册商标。

GestIC 是 Microchip Technology Inc.的子公司 Microchip Technology Germany II GmbH & Co. KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2020，Microchip Technology Incorporated，版权所有。

ISBN： 978-1-5224-6688-8

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： www.microchip.com/support 网址： www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880- 3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4485-5910 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820