# 使用 MPLAB® Harmony v3 和 MCC 为 PIC32CZ CA 和 SAM E70/S70/V7x MCU 上的外设添加 DMA 支持

**TB3370**

产品页链接

## 简介

在现代计算系统中，高效的数据传输机制是保持高性能和快速响应的关键，尤其在需处理大量数据的应用中。直接存储器访问（Direct Memory Access，DMA）正是这样一种机制，在优化数据移动方面发挥着关键作用。通过允许硬件子系统独立于 CPU 访问系统，DMA 显著减少了处理瓶颈。此外，借助专用 DMA 控制器，可确保无缝管理数据传输，从而进一步提高系统效率和性能。

本文档说明如何使用 MPLAB® Harmony v3 和 MCC 配置 DMA 并将 DMA 与外设（例如 PIC32CZ CA 和 SAM E/S/V 系列器件上的 SPI）结合使用。

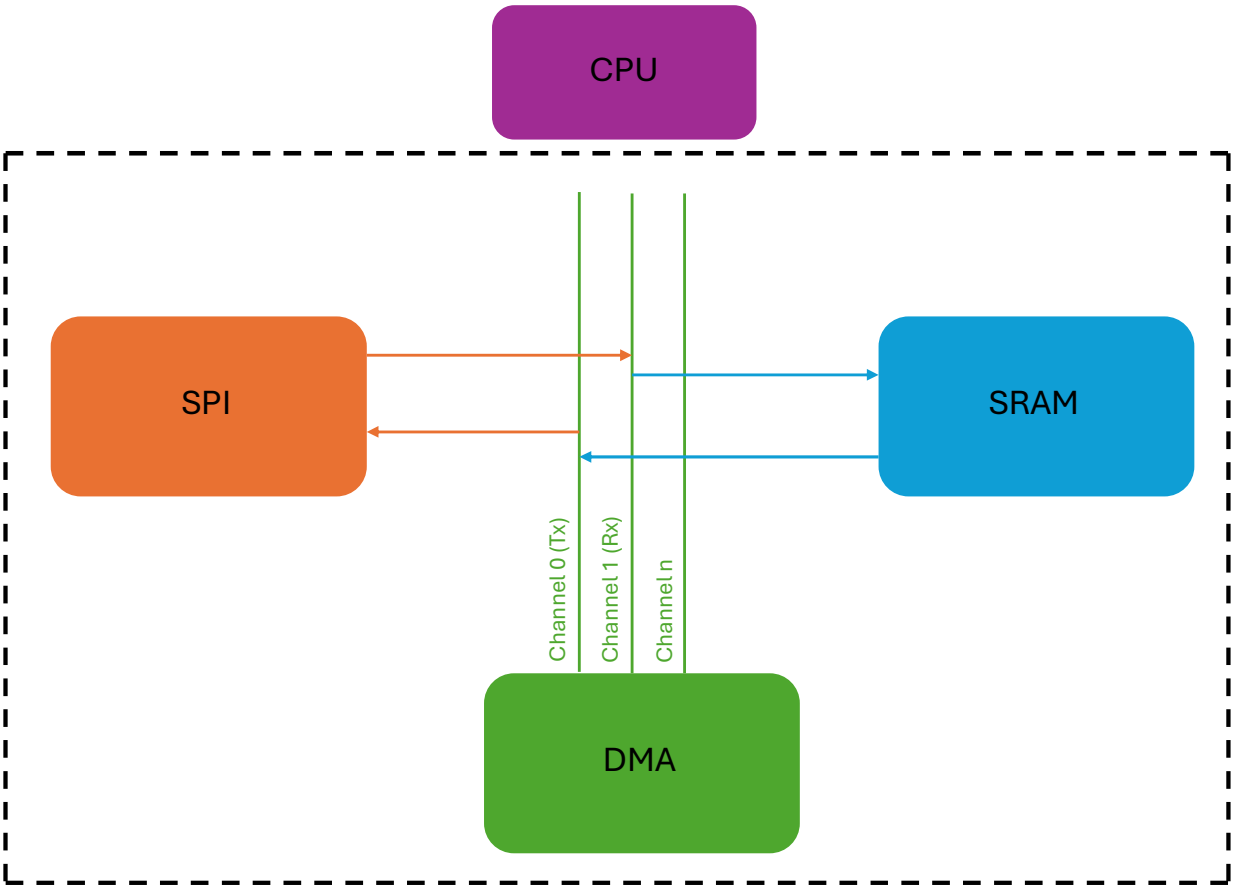
# 目录

# 1. 什么是 DMA？

直接存储器访问（DMA）是一项重要功能，它允许特定硬件子系统独立于中央处理器（Central Processing Unit，CPU）访问系统存储器（RAM）。此功能显著提高数据传输的效率和性能，尤其是在需要快速移动大量数据的应用中，例如多媒体、网络和存储系统。

DMA 过程由名为 DMA 控制器的专用硬件组件管理，该组件既可集成到 CPU 中，也可作为单独的芯片存在。该控制器通常包含多个通道，每个通道能够处理单独的数据传输操作，并通过寄存器来存放存储器地址、传输计数和控制信息。DMA 的主要优势包括效率提升、数据传输速率加快和 CPU 开销降低，这使其成为音频和视频流以及实时嵌入式系统等应用中不可或缺的组件。

**图 1-1.** DMA 框图




## 1.1. DMA 工作原理

只有检测到 DMA 传输请求时，才能启动 DMA 传输。传输请求可以由软件、外设或事件发起。DMA 操作的流程为：首先由 CPU 初始化 DMA 控制器，随后由外设发起传输请求；然后，DMA 控制器会在有多个请求时（如有必要）进行仲裁，并控制系统总线，以直接在存储器与外设之间进行数据传输，从而释放 CPU 来执行其他任务。传输完成后，DMA 控制器会通过中断向 CPU 发送通知。

## 1.2. DMA 中的数据传输类型

DMA 数据传输能够实现不同组件之间的高效通信，对于优化系统性能至关重要。根据数据的源位置和目标位置，这类数据传输可分为以下四类：

- 外设到存储器传输

- 存储器到外设传输
- 外设到外设传输
- 存储器到存储器传输

**表 1-1.** DMA 中的数据传输类型

| 类型 | 源位置 | 目标位置 |
|---|---|---|
| **外设到存储器** | 外设 | 存储器 |
| **存储器到外设** | 存储器 | 外设 |
| **外设到外设** | 外设 | 外设 |
| **存储器到存储器** | 存储器 | 存储器 |

**注：**SAM E/S/V 系列器件不支持外设到外设类型的数据传输。

根据组件传输的数据大小，DMA 传输模式还可分为以下几类：

- **节拍传输：**单次数据总线传输的数据大小。
- **块传输：**单个传输描述符所能传输的数据量。
- **突发传输：**无需 CPU 干预的背靠背 DMA 传输。
- **DMA 事务：**完整传输描述符链表中所有数据的过程。
- **周期窃取：**DMA 控制器在每个周期后中断 CPU 来传输数据。

# 2. 使用 MPLAB Harmony v3 和 MCC 创建应用程序

本演示使用以下软件和硬件工具：

- MPLAB X IDE v6.20
- MPLAB 代码配置器插件 v5.5.1
- MPLAB XC32 编译器 v4.45
- csp v3.20.0
- PIC32CZ CA90 Curiosity Ultra 开发板
- PIC32CZ CA80 Curiosity Ultra 开发板
- SAM E70 Xplained Ultra 评估工具包
- SAM V71 Xplained Ultra 评估工具包

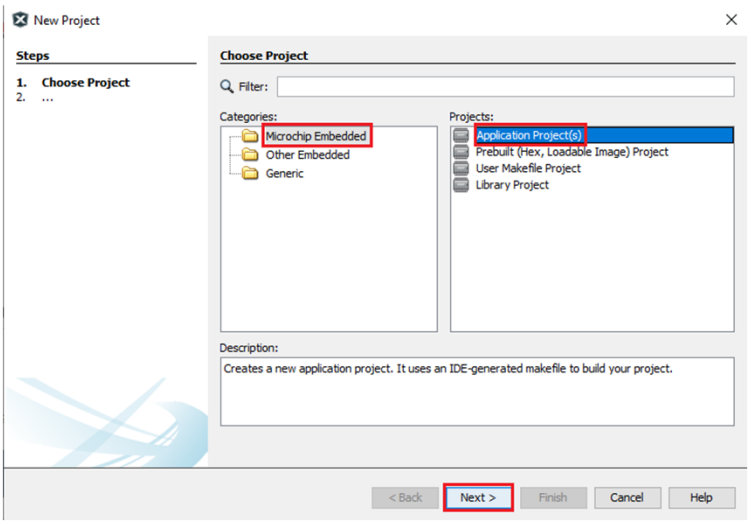
**注：** 上述工具的更新版本也可用于创建应用程序，用户无需局限于旧版本。此外，创建应用程序只需选用上述开发板中的一种即可。

## 2.1. 使用 PIC32CZ CA90 Curiosity Ultra 开发板创建演示应用程序

要创建基于 MPLAB Harmony v3 的项目，请按照以下步骤操作：

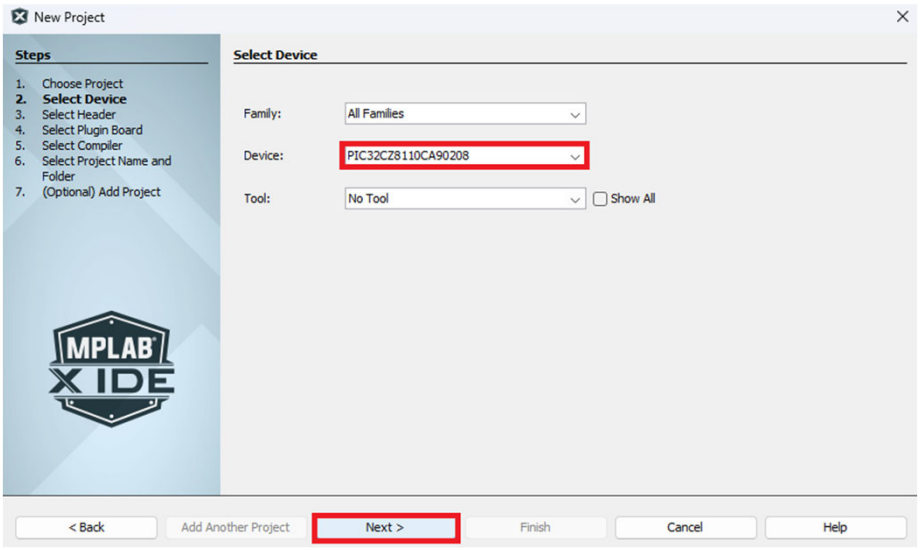
1. 从 **Start**（开始）菜单启动 MPLAB X IDE。
2. 在 **File**（文件）菜单上，单击 **New Project**（新建项目）或单击 *New Project* 图标。
3. 此时将显示 **New Project** 窗口。在 **Steps**（步骤）导航窗格中，单击 **Choose Project**（选择项目）。
4. 在右侧的 **Choose Project** 属性页面中：
   a. “Categories”（类别）：选择 **Microchip Embedded**（Microchip 嵌入式）。
   b. “Projects”（项目）：选择 **Application Project(s)**（应用程序项目）。
5. 单击 **Next**（下一步）。

**图 2-1.** 新建项目窗口



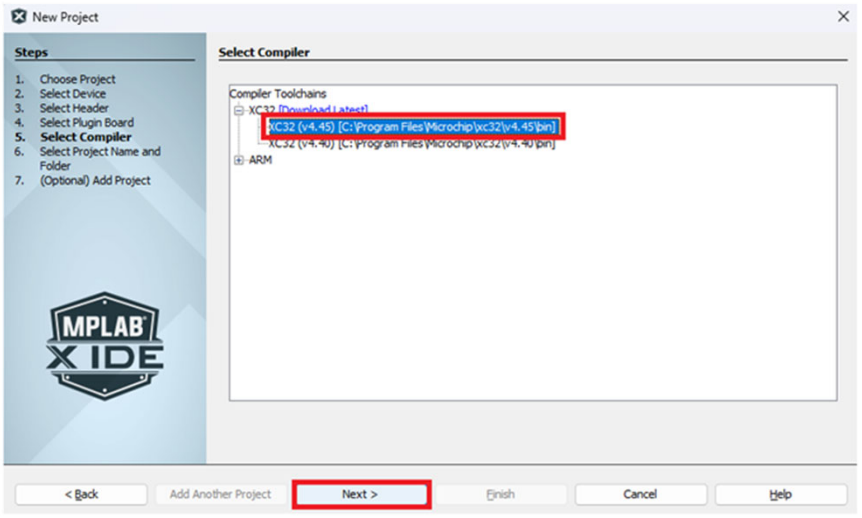
6. 单击 **Select Device**（选择器件），并在右侧的 **Select Device** 属性页面中，为 **Device**（器件）选择 **PIC32CZ8110CA90208**，以便在 PIC32CZ CA90 Curiosity Ultra 开发板上创建项目（所选器件将显示在“Target Device”（目标器件）下）。

**图 2-2.** 器件选择




7. 单击 **Next**。
8. 单击 **Select Compiler**（选择编译器），并在右侧的 **Select Compiler** 属性页面的 **Compiler Toolchains**（编译器工具链）下，单击并展开 **XC32**，然后选择 **XC32（v4.45）**。

**图 2-3.** 选择编译器

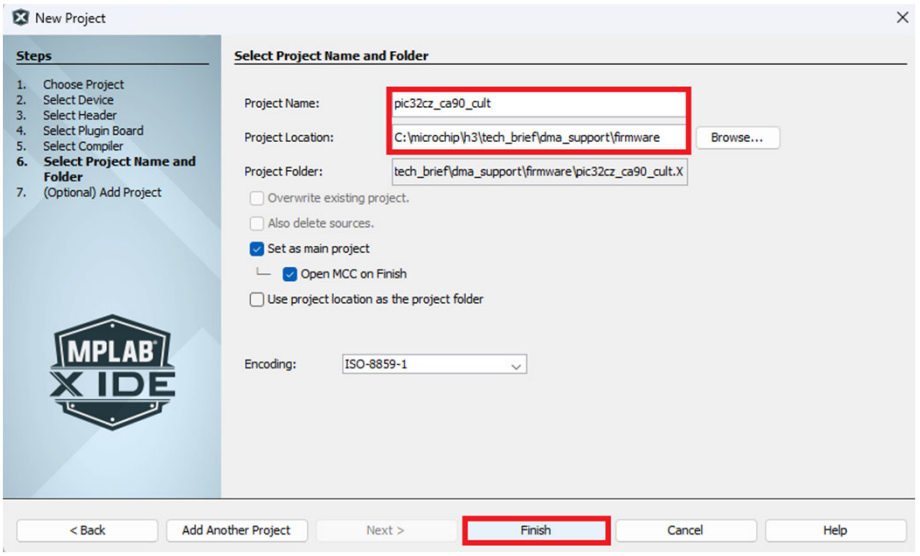



9. 单击 **Next**。
10. 单击 **Select Project Name and Folder**（选择项目名称和文件夹），并在右侧的 **Select Project Name and Folder** 属性页面中：
    a. **Project Name**（项目名称）：输入 *pic32cz_ca90_cult*（表示项目的名称，它将显示在 MPLAB X IDE 中，用于设置项目的名称）。

b. **Project Location**（项目位置）：输入 *C:\microchip\h3\tech_brief\dma_support\firmware*（表示新项目根文件夹的路径。所有项目文件都将存放在此文件夹中。项目位置可以是任何有效的路径）。

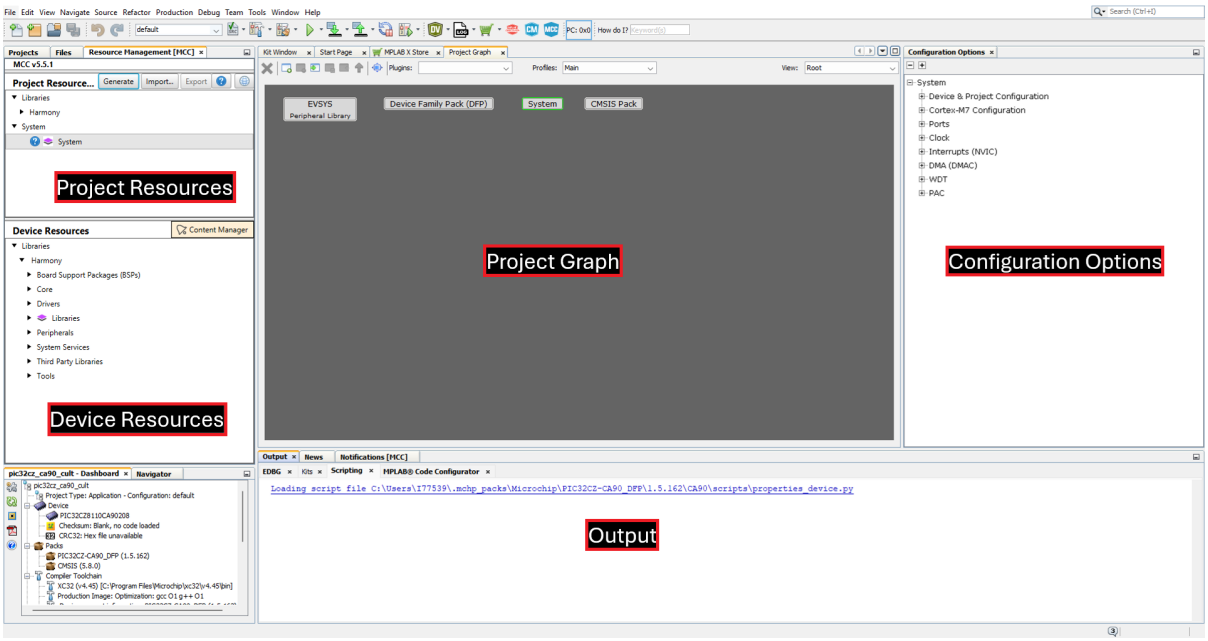
c. **Project Folder**（项目文件夹）：只读内容（当用户更改上述条目时，会自动更新）。

**图 2-4.** 项目名称和文件夹设置




11. 单击 **Finish**（完成）以启动 MCC。
12. MCC 插件将在新窗口中打开，如下图所示：

**图 2-5.** MPLAB 代码配置器窗口

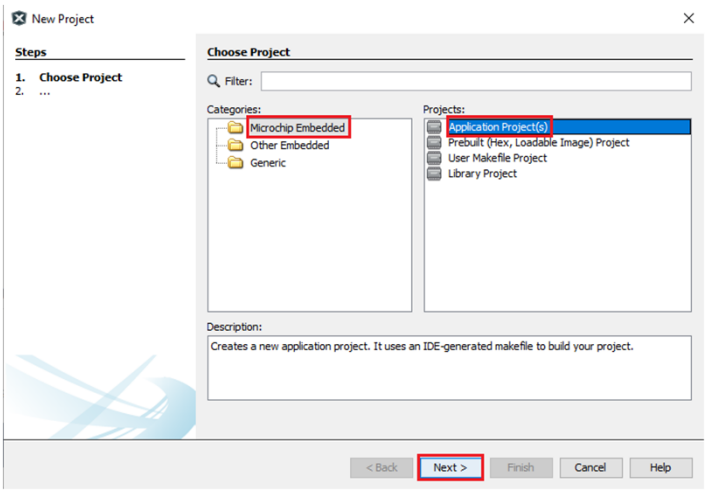


## 2.2. 使用 SAM E70 Xplained Ultra 评估工具包创建演示应用程序

要创建基于 MPLAB Harmony v3 的项目，请按照以下步骤操作：

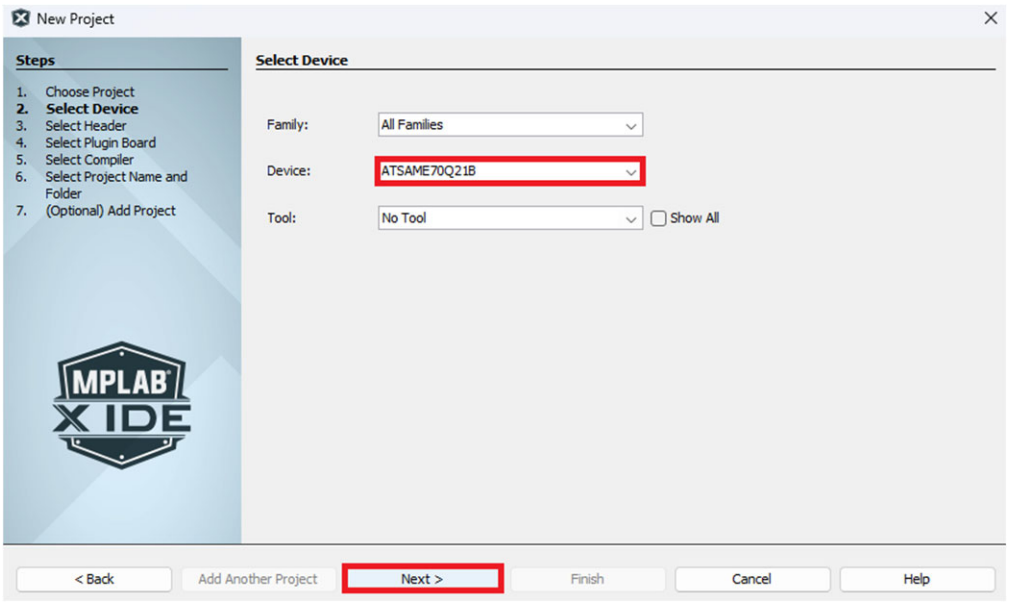
1. 从 **Start** 菜单启动 MPLAB X IDE。
2. 在 **File** 菜单上，单击 **New Project** 或单击 *New Project* 图标。
3. 此时将显示 **New Project** 窗口。在 **Steps** 导航窗格中，单击 **Choose Project**。
4. 在右侧的 **Choose Project** 属性页面中：
   a. “Categories”：选择 **Microchip Embedded**。
   b. “Projects”：选择 **Application Project(s)**。
5. 单击 **Next**。

**图 2-6.** 新建项目窗口



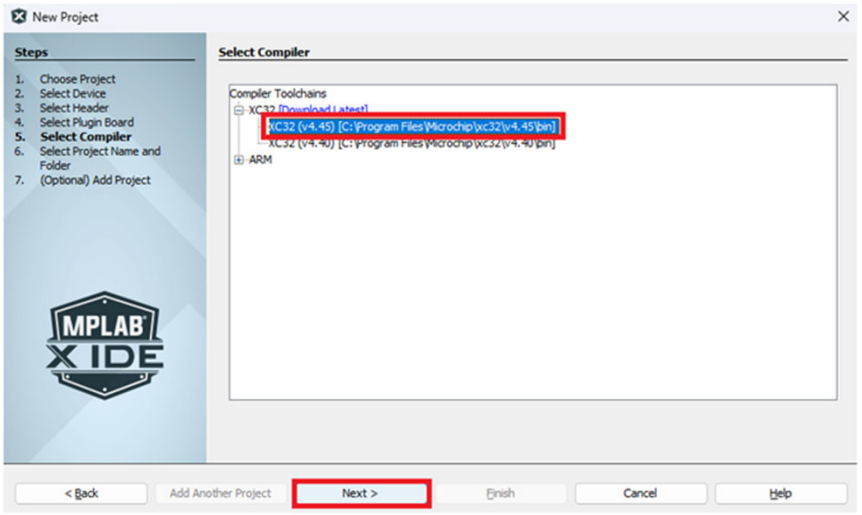

6. 单击 **Select Device**，并在右侧的 **Select Device** 属性页面中，为 **Device** 选择 **ATSAME70Q21B**，以便在 SAM E70 Xplained Ultra 评估工具包上创建项目（器件条目将显示在“Target Device”下）。

**图 2-7.** 器件选择




7. 单击 **Next**。
8. 单击 **Select Compiler**，并在右侧的 **Select Compiler** 属性页面的“Compiler Toolchains”下，单击并展开 **XC32**，然后选择 **XC32（v4.45）**。

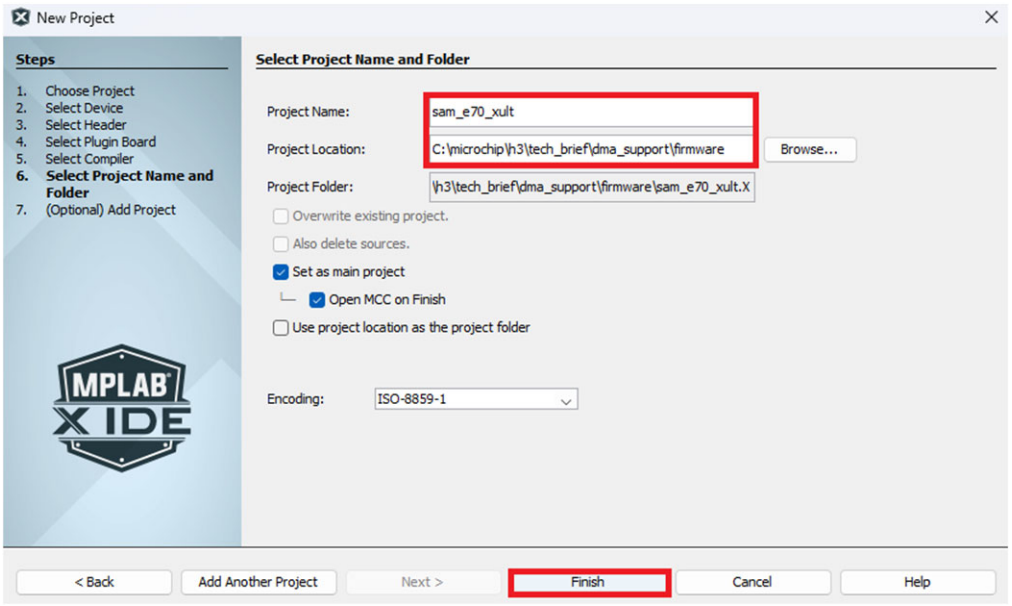
**图 2-8.** 选择编译器




9. 单击 **Next**。
10. 单击 **Select Project Name and Folder**，并在右侧的 **Select Project Name and Folder** 属性页面中：
    a. **Project Name**：输入 *sam_e70_xult*（表示项目的名称，它将显示在 MPLAB X IDE 中，用于设置项目的名称）。
    b. **Project Location**：输入 *C:\microchip\h3\tech_brief\dma_support\firmware*（表示新项目根文件夹的路径。所有项目文件都将存放在此文件夹中。项目位置可以是任何有效的路径）。

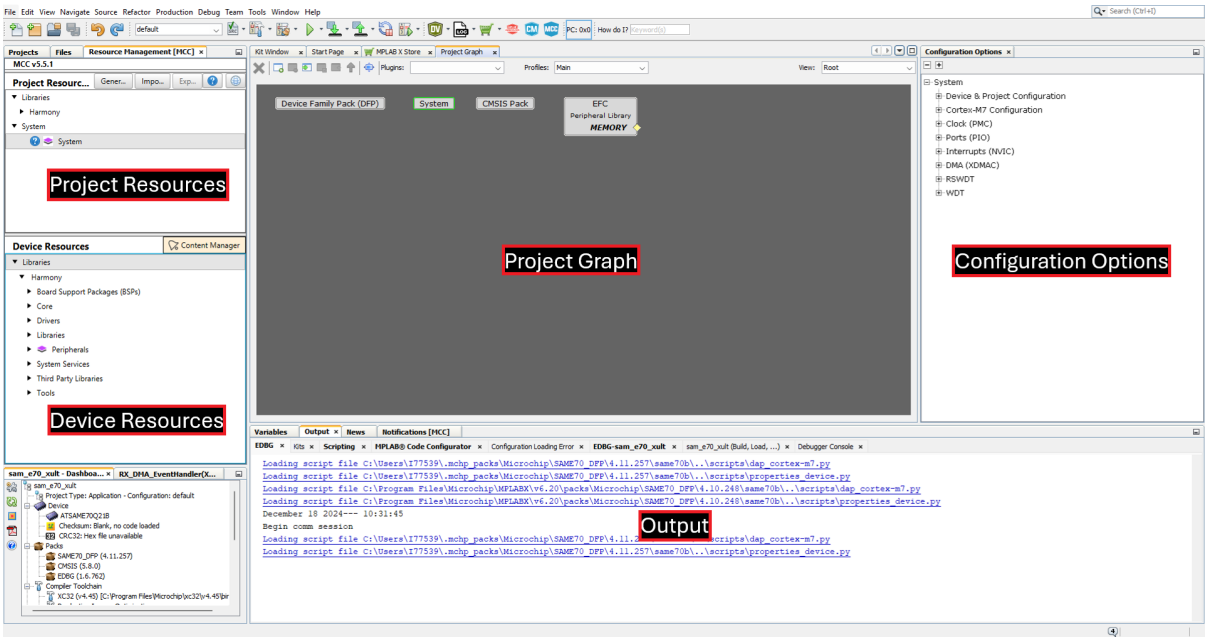
c. **Project Folder**：只读内容（当用户更改上述条目时，会自动更新）。

**图 2-9.** 项目名称和文件夹设置




11. 单击 **Finish** 以启动 MCC。
12. MCC 插件将在新窗口中打开，如下图所示：

**图 2-10.** MPLAB 代码配置器窗口



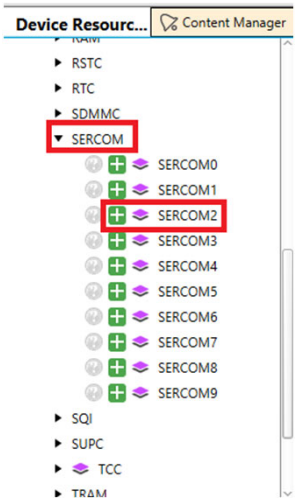
# 3. 添加和配置 MPLAB Harmony 组件

## 3.1. PIC32CZ CA90 Curiosity Ultra 开发板

要使用 MCC 添加和配置 MPLAB Harmony 组件，请按照以下步骤操作：

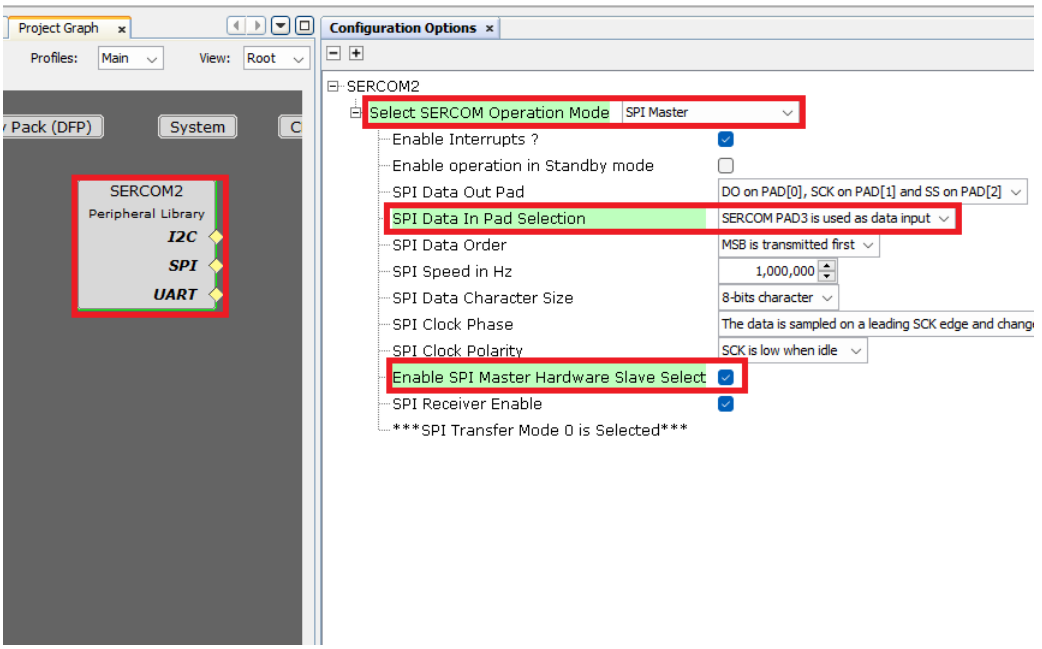
1. 在“MCC”窗口中，在“Device Resources”（器件资源）下，单击并展开 *Harmony* > *Peripherals* > *SERCOM*（Harmony > 外设 > SERCOM）选项列表。
2. 单击 **SERCOM2**，并确认“SERCOM2 Peripheral Library”（SERCOM2 外设库）模块已添加到“Project Graph”（项目图）部分。

**图 3-1.** SERCOM 模块



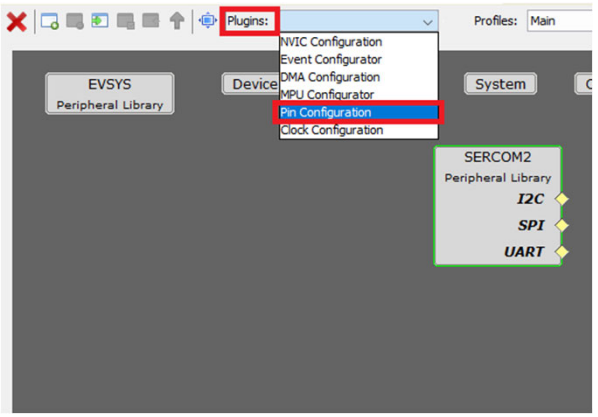

3. 单击 **SERCOM2**，并在 **Configuration Options**（配置选项）属性页面中，单击并展开 **SERCOM2**，然后将 SERCOM2 模块配置为 SPI 主器件，如下所示。

**图 3-2.** SERCOM2 配置



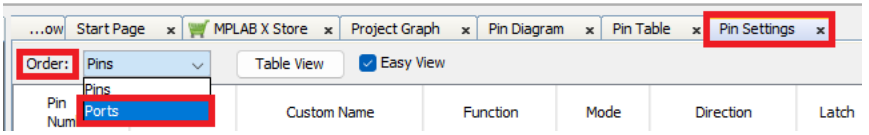

4. 在 **Plugins**（插件）下拉列表中，选择 **Pin Configuration**（引脚配置）。

**图 3-3.** 插件——引脚配置




5. 单击 **Pin Settings**（引脚设置），然后从“Order”（顺序）列表中选择 **Ports**（端口）对条目进行排序。

**图 3-4.** 从“Order”菜单中选择“Ports”



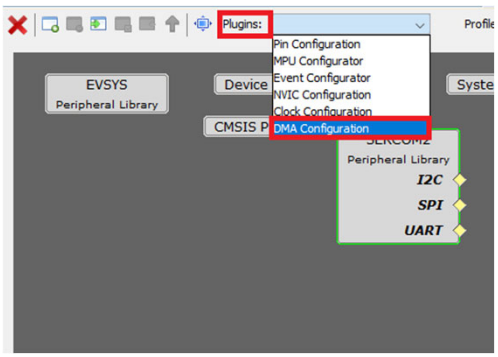

6. 按如下所示配置引脚：

**图 3-5.** 引脚配置

| Pin Number | Pin ID | Custom Name | Function | Mode | Direction | Latch | Pull Up | Pull Down | Open Drain | Slew Rate |
|---|---|---|---|---|---|---|---|---|---|---|
| V15 | PB31 | | Available | Digital | High Impedance | Low | | | | FAST |
| P15 | PC00 | | Available | Digital | High Impedance | Low | | | | FAST |
| T17 | PC01 | | Available | Digital | High Impedance | Low | | | | FAST |
| T18 | PC02 | | Available | Digital | High Impedance | Low | | | | FAST |
| R17 | PC03 | | Available | Digital | High Impedance | Low | | | | FAST |
| R18 | PC04 | | Available | Digital | High Impedance | Low | | | | FAST |
| N15 | PC05 | | Available | Digital | High Impedance | Low | | | | FAST |
| N17 | PC06 | | Available | Digital | High Impedance | Low | | | | FAST |
| N18 | PC07 | | Available | Digital | High Impedance | Low | | | | FAST |
| M13 | PC08 | | SERCOM2_PAD0 | Digital | High Impedance | n/a | | | | FAST |
| M18 | PC09 | | SERCOM2_PAD1 | Digital | High Impedance | n/a | | | | FAST |
| H17 | PC10 | | SERCOM2_PAD2 | Digital | High Impedance | n/a | | | | FAST |
| H15 | PC11 | | SERCOM2_PAD3 | Digital | High Impedance | n/a | | | | FAST |
| F18 | PC12 | | Available | Digital | High Impedance | Low | | | | FAST |
| F17 | PC13 | | Available | Digital | High Impedance | Low | | | | FAST |
| E17 | PC14 | | Available | Digital | High Impedance | Low | | | | FAST |

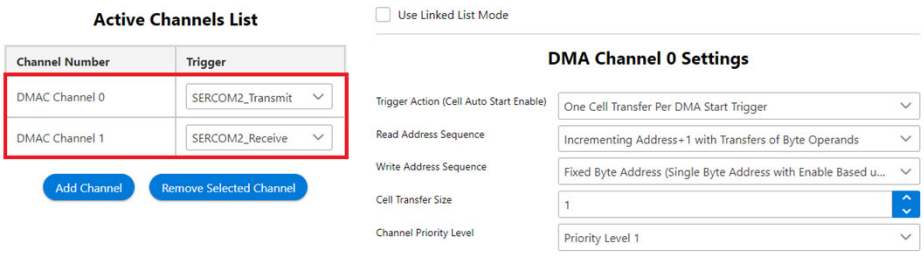
7. 在 **Plugins** 下拉列表中，选择 **DMA Configuration**（DMA 配置）。

**图 3-6.** 插件——DMA 配置



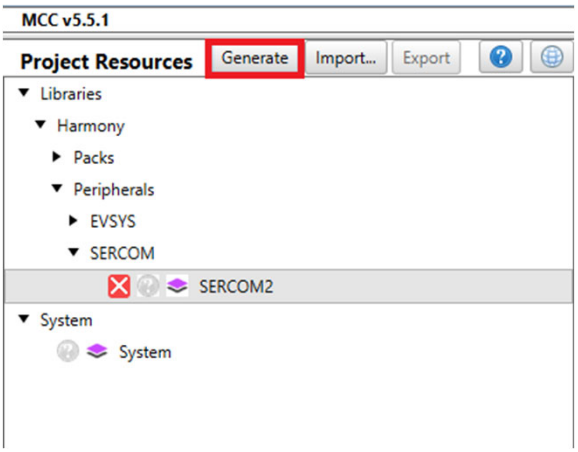

8. 在“DMA Configuration”中，单击 **Add Channel**（添加通道）以添加 DMAC 通道 0，并将触发器选择为 SERCOM2_Transmit。重复相同步骤，但对于 DMAC 通道 1，选择 SERCOM2_Receive 作为触发器。

**图 3-7.** DMA 配置




9. 单击 **Generate**（生成）以生成代码。

**图 3-8.** 生成代码

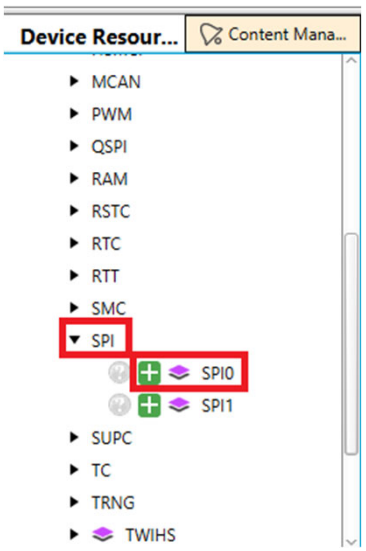


## 3.2. SAM E70 Xplained Ultra 评估工具包

要使用 MCC 添加和配置 MPLAB Harmony 组件，请按照以下步骤操作：

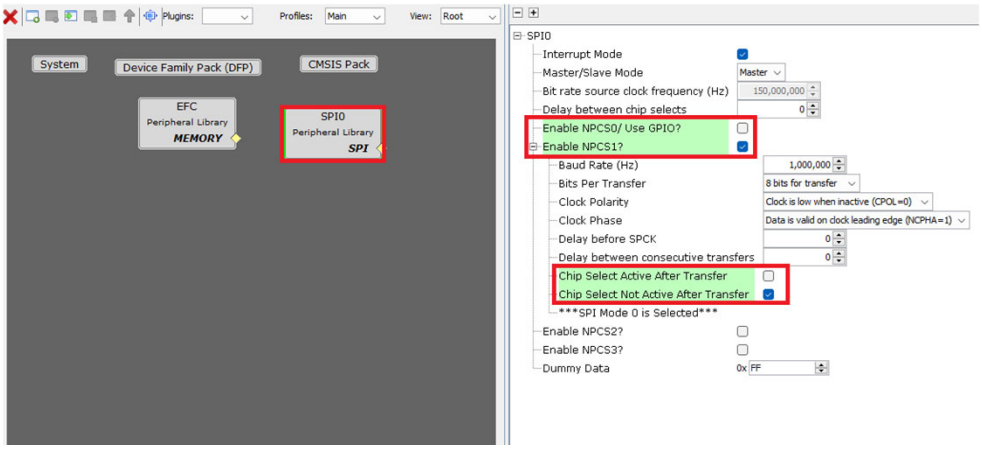
1. 在“MCC”窗口中，在“Device Resources”下，单击并展开 *Harmony* > *Peripherals* > *SPI*（Harmony > 外设 > SPI）选项列表。
2. 单击 **SPI0**，并确认“SPI0 Peripheral Library”模块已添加到“Project Graph”部分。

**图 3-9.** SPI 模块



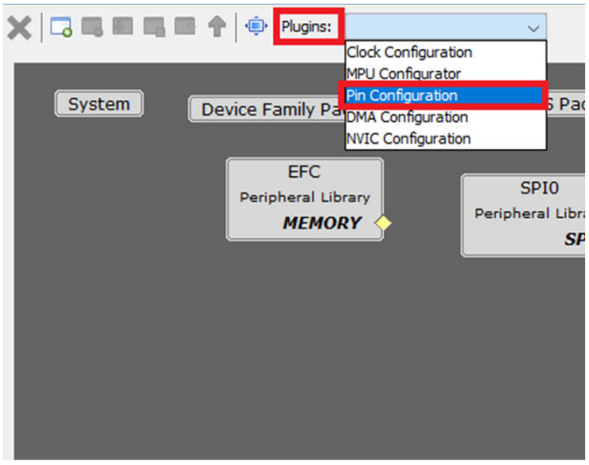

3. 单击“SPI0”，要配置 SPI0 模块，请单击并展开 SPI0，然后按如下所示进行配置。

**图 3-10.** SPI0 配置



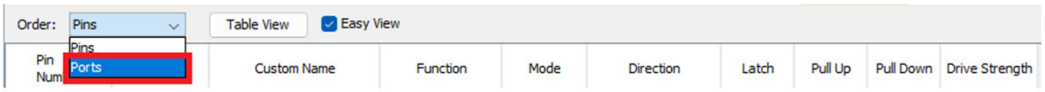

4. 在 **Plugins** 下拉列表中，选择 **Pin Configuration**。

**图 3-11.** 插件——引脚配置




5. 从 **Order** 下拉列表中，选择 **Ports** 对条目进行排序。

**图 3-12.** 从“Order”菜单中选择“Ports”




6. 从 **Plugins** 下拉列表中，选择 **DMA Configuration**。在 **DMA Configuration** 属性页面中，单击 **Add Channel** 以添加 DMAC 通道 0，并将触发器设置为 SPI0_Transmit。重复相同步骤，但对于 DMAC 通道 1，将触发器设置为 SPI0_Receive。

**图 3-13.** 插件——DMA 配置

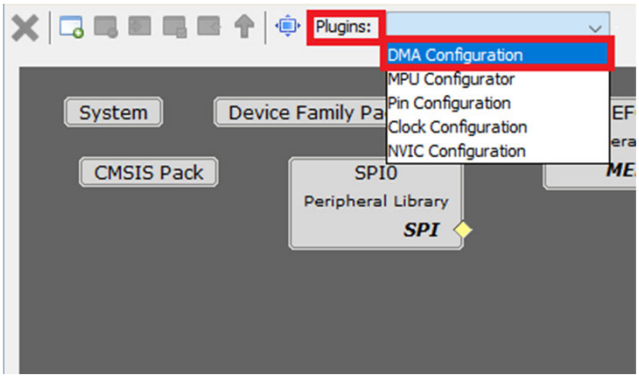
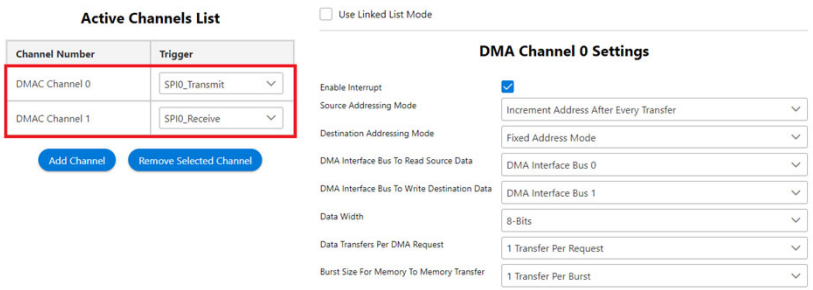


**图 3-14.** DMA 配置




7. 按如下所示配置引脚。

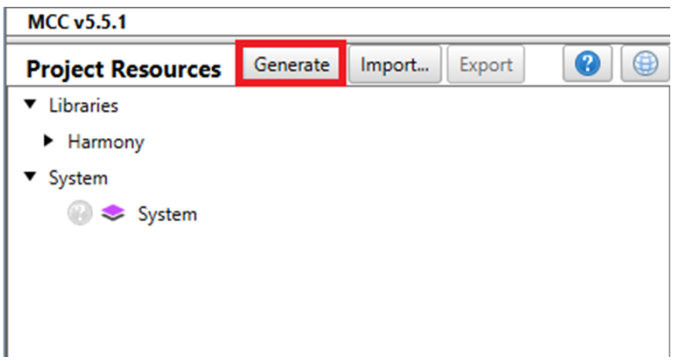
**图 3-15.** 引脚配置

Order: Ports  Table View

| Pin Number | Pin ID | Custom Name | Function | Direction | Latch | Open Drain | PIO Interrupt | Pull Up | Pull Down | Glitch/Debounce Filter | Drive |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 74 | PD17 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 69 | PD18 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 67 | PD19 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 65 | PD20 | | SPI0_MISO | n/a | n/a | | Disabled | | | Disabled | Low |
| 63 | PD21 | | SPI0_MOSI | n/a | n/a | | Disabled | | | Disabled | Low |
| 60 | PD22 | | SPI0_SPCK | n/a | n/a | | Disabled | | | Disabled | Low |
| 57 | PD23 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 55 | PD24 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 52 | PD25 | | SPI0_NPCS1 | n/a | n/a | | Disabled | | | Disabled | Low |
| 53 | PD26 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 47 | PD27 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 71 | PD28 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 108 | PD29 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 34 | PD30 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 2 | PD31 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 4 | PE0 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 6 | PE1 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 7 | PE2 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 10 | PE3 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 27 | PE4 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 28 | PE5 | | Available | In | n/a | | Disabled | | | Disabled | Low |
| 3 | VDDOUT | | | In | n/a | | | | | | Low |
| 5 | VDDIN | | | In | n/a | | | | | | Low |

8. 单击 **Generate** 以生成代码。

**图 3-16.** 生成项目



# 4. 向项目添加应用程序逻辑

## 4.1. PIC32CZ CA90 Curiosity Ultra 开发板

要开发并运行应用程序，请按照以下步骤操作：

1. 单击 **Projects**（项目），然后在“Source files”（源文件）下打开项目的 `main.c` 文件，并在 `main()` 函数外部添加所需的变量。

```
uint8_t __attribute__ ((aligned(32))) Tx[10] = {0x1F, 0x2F, 0x3F, 0x4F, 0x5F, 0x6F, 0x7F,
0x8F, 0x9F, 0xAF};
uint8_t __attribute__ ((aligned(32)))Rx[10];

/* transfer done flag */
volatile bool Rx_transfer_done = false;
volatile bool Tx_transfer_done = false;
```

2. 在 `main()` 函数外部，为 Tx 和 Rx 添加 DMA 事件处理程序。

```
/* This is called after transfer is done */
void Tx_DMA_EventHandler(DMA_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == DMA_TRANSFER_EVENT_BLOCK_TRANSFER_COMPLETE)
    {
        Tx_transfer_done = true;
    }
}

void Rx_DMA_EventHandler(DMA_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == DMA_TRANSFER_EVENT_BLOCK_TRANSFER_COMPLETE)
    {
        Rx_transfer_done = true;
    }
}
```

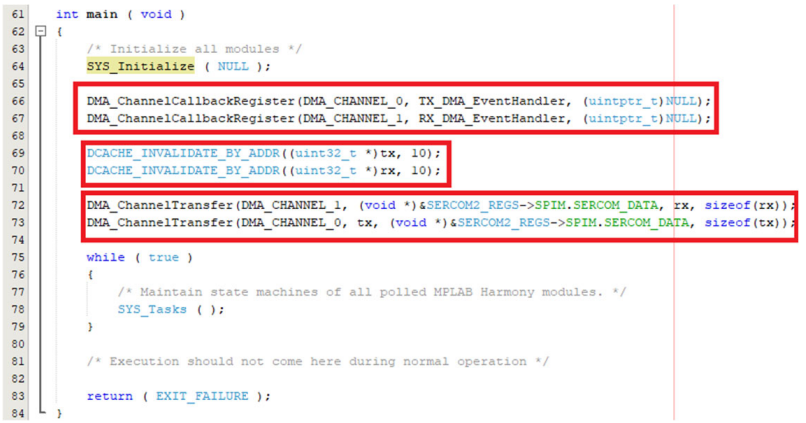
**图 4-1.** 添加宏、变量和事件处理程序

3. 添加 DMA 回调注册函数、高速缓存失效函数和 DMA 通道传递函数。

```
DMA_ChannelCallbackRegister(DMA_CHANNEL_0, Tx_DMA_EventHandler, (uintptr_t)NULL);
    DMA_ChannelCallbackRegister(DMA_CHANNEL_1, Rx_DMA_EventHandler, (uintptr_t)NULL);

    DCACHE_INVALIDATE_BY_ADDR((uint32_t *)Tx, 10);
    DCACHE_INVALIDATE_BY_ADDR((uint32_t *)Rx, 10);

    DMA_ChannelTransfer(DMA_CHANNEL_1, (void *)&SERCOM2_REGS->SPIM.SERCOM_DATA, Rx,
sizeof(Rx));
    DMA_ChannelTransfer(DMA_CHANNEL_0, Tx, (void *)&SERCOM2_REGS->SPIM.SERCOM_DATA,
sizeof(Tx));
```

**注：**高速缓存失效对于维护数据的一致性和准确性至关重要，尤其是在数据频繁变化的动态系统中。通过使高速缓存失效，可确保下次请求数据时，系统会从原始源而不是从高速缓存中获取数据。

**图 4-2.** 应用程序逻辑




## 4.2. SAM E70 Xplained Ultra 评估工具包

要开发并运行应用程序，请按照以下步骤操作：

1. 打开项目的 `main.c` 文件，并在 `main()` 函数外部添加所需的变量。

```
uint8_t __attribute__ ((aligned(32))) Tx[10] = {0x1F, 0x2F, 0x3F, 0x4F, 0x5F, 0x6F, 0x7F,
0x8F, 0x9F, 0xAF};
uint8_t __attribute__ ((aligned(32)))Rx[10];

/* transfer done flag */
volatile bool Rx_transfer_done = false;
volatile bool Tx_transfer_done = false;
```

2. 在 `main()` 函数外部，为 Tx 和 Rx 添加 DMA 事件处理程序。

```
/* This is called after transfer is done */
void Tx_DMA_EventHandler(XDMAC_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == XDMAC_TRANSFER_COMPLETE)
    {
        Tx_transfer_done = true;
    }
}

void Rx_DMA_EventHandler(XDMAC_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == XDMAC_TRANSFER_COMPLETE)
    {
        Rx_transfer_done = true;
    }
}
```

**图 4-3.** 添加宏、变量和事件处理程序

```
#include <stddef.h>                     // Defines NULL
#include <stdbool.h>                    // Defines true
#include <stdlib.h>                     // Defines EXIT_FAILURE
#include "definitions.h"                // SYS function prototypes


// *****************************************************************************
// *****************************************************************************
// Section: Main Entry Point
// *****************************************************************************
// *****************************************************************************

uint8_t __attribute__ ((aligned(32))) tx[10] = {0x1F, 0x2F, 0x3F, 0x4F, 0x5F, 0x6F, 0x7F, 0x8F, 0x9F, 0xAF};
uint8_t __attribute__ ((aligned(32)))rx[10];

/* transfer done flag */
volatile bool rx_transfer_done = false;
volatile bool tx_transfer_done = false;

/* This is called after transfer is done */
void TX_DMA_EventHandler(XDMAC_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == XDMAC_TRANSFER_COMPLETE)
    {
        tx_transfer_done = true;
    }
}

void RX_DMA_EventHandler(XDMAC_TRANSFER_EVENT event, uintptr_t context)
{
    if (event == XDMAC_TRANSFER_COMPLETE)
    {
        rx_transfer_done = true;
    }
}
```

3. 添加 DMA 回调注册函数、高速缓存失效函数和 DMA 通道传递函数。

```
XDMAC_ChannelCallbackRegister(XDMAC_CHANNEL_0, Tx_DMA_EventHandler, (uintptr_t)NULL);
XDMAC_ChannelCallbackRegister(XDMAC_CHANNEL_1, Rx_DMA_EventHandler, (uintptr_t)NULL);

DCACHE_INVALIDATE_BY_ADDR((uint32_t *)Tx, 10);
DCACHE_INVALIDATE_BY_ADDR((uint32_t *)Rx, 10);

XDMAC_ChannelTransfer(XDMAC_CHANNEL_1, (void *)&SPI0_REGS->SPI_RDR, Rx, sizeof(Rx));
XDMAC_ChannelTransfer(XDMAC_CHANNEL_0, Tx, (void *)&SPI0_REGS->SPI_TDR, sizeof(Tx));
```

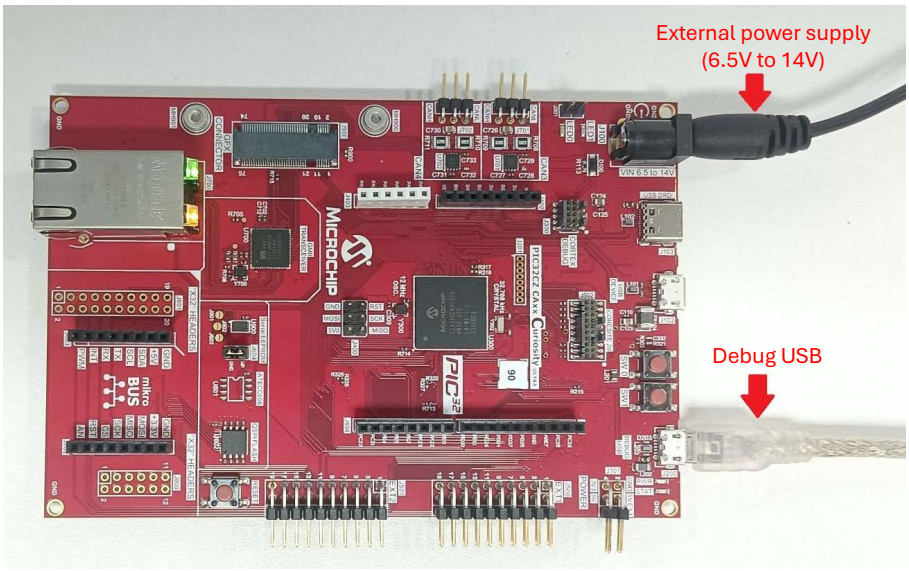
**图 4-4.** 应用程序逻辑

```
int main ( void )
{
    /* Initialize all modules */
    SYS_Initialize ( NULL );

    XDMAC_ChannelCallbackRegister(XDMAC_CHANNEL_0, TX_DMA_EventHandler, (uintptr_t)NULL);
    XDMAC_ChannelCallbackRegister(XDMAC_CHANNEL_1, RX_DMA_EventHandler, (uintptr_t)NULL);

    DCACHE_INVALIDATE_BY_ADDR((uint32_t *)tx, 10);
    DCACHE_INVALIDATE_BY_ADDR((uint32_t *)rx, 10);

    XDMAC_ChannelTransfer(XDMAC_CHANNEL_1, (void *)&SPI0_REGS->SPI_RDR, rx, sizeof(rx));
    XDMAC_ChannelTransfer(XDMAC_CHANNEL_0, tx, (void *)&SPI0_REGS->SPI_TDR, sizeof(tx));

    while ( true )
    {
        /* Maintain state machines of all polled MPLAB Harmony modules. */
        SYS_Tasks ( );
    }

    /* Execution should not come here during normal operation */

    return ( EXIT_FAILURE );
}
```

# 5. 编译和调试应用程序

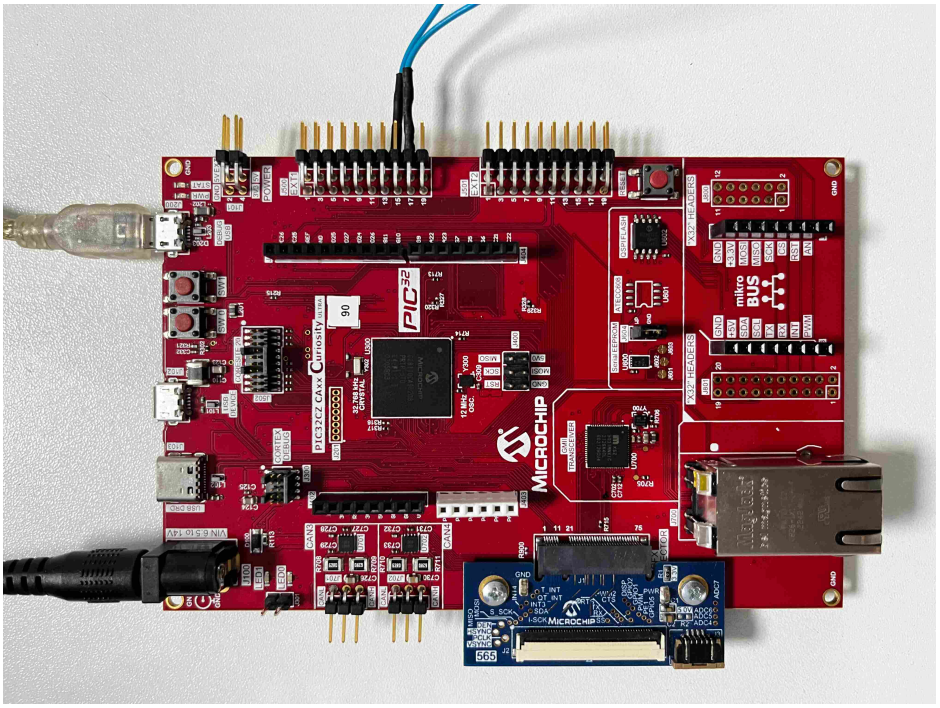
1. PIC32CZ CA90 Curiosity Ultra 开发板支持使用嵌入式调试器（Embedded Debugger，EDBG）进行调试。将 Type-A 公头转 micro-B USB 线缆连接到 PIC32CZ CA90 Curiosity Ultra 开发板的 micro-B USB 端口，并将 Type-A 公头连接到 PC。此外，连接外部电源（6.5V-14V）为开发板供电。

**图 5-1.** 硬件——PIC32CZ CA90 Curiosity Ultra 开发板



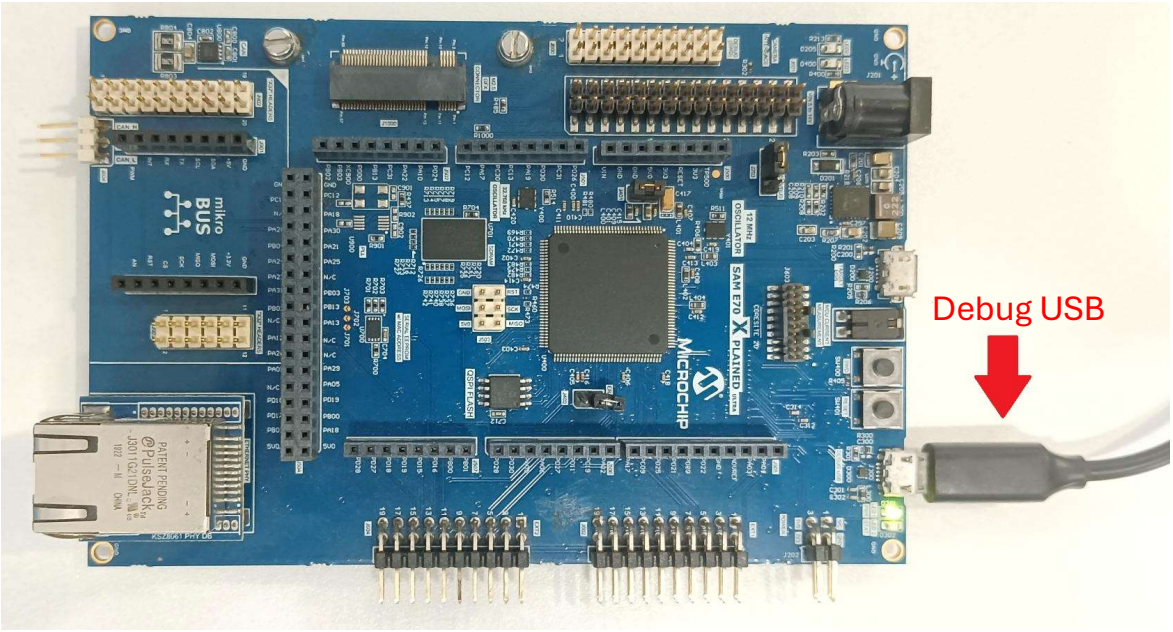

2. 使用导线短接 PIC32CZ CA90 Curiosity Ultra 开发板中 EXT1 的 MISO（PC11——引脚 17）和 MOSI（PC08——引脚 16）引脚。

**图 5-2.** 硬件设置——PIC32CZ CA90 Curiosity Ultra 开发板



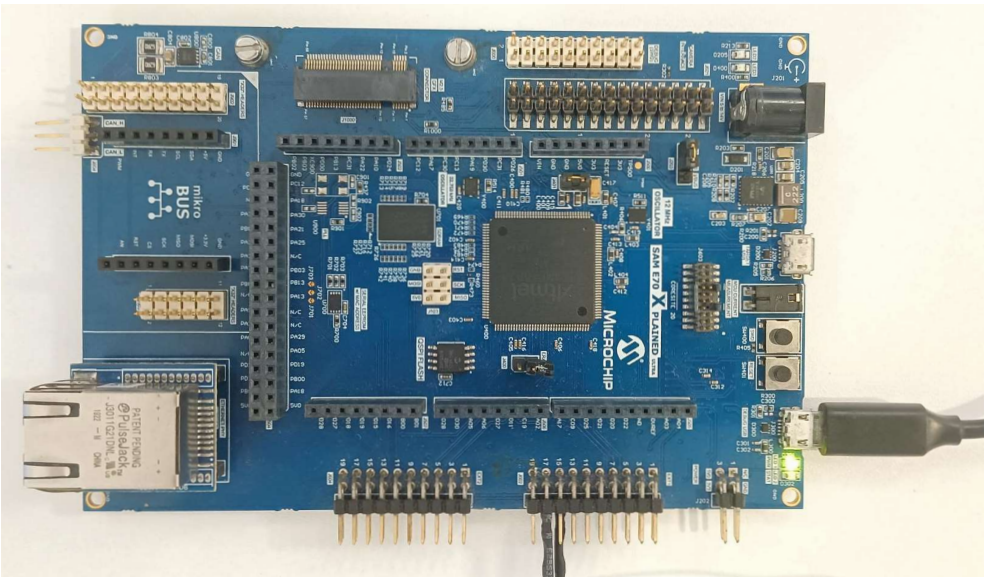
3. 将 Type-A 公头转 micro-B USB 线缆连接到 micro-B 调试 USB 端口，为 SAM E70 Xplained Ultra 评估工具包供电并进行调试。

**图 5-3.** 硬件——SAM E70 Xplained Ultra 评估工具包



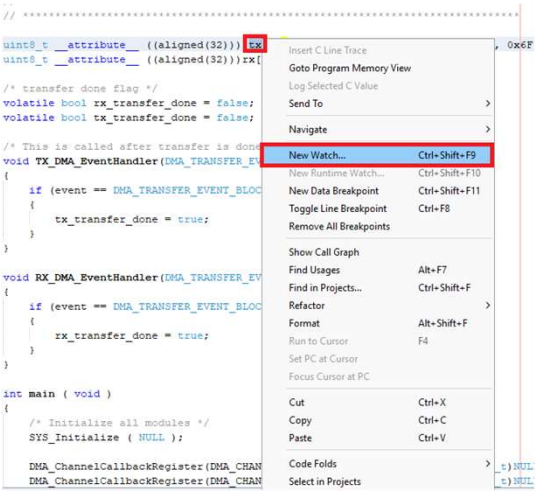

4. 使用导线短接 SAM E70 Xplained Ultra 评估工具包 EXT1 的 MISO（PD20——引脚 17）和 MOSI（PD21——引脚 16）引脚。

**图 5-4.** 硬件设置——SAM E70 Xplained Ultra 评估工具包



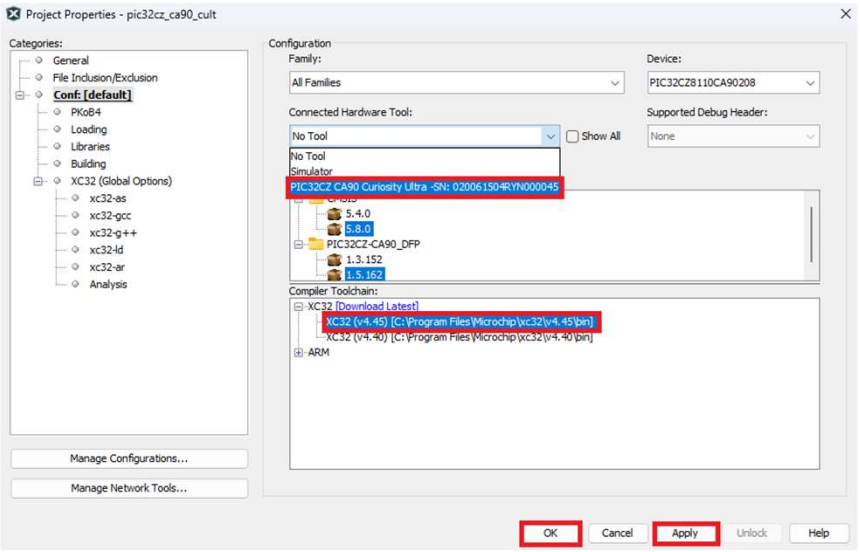
5. 在代码中选择 **Tx** 变量，右键单击所选文本，然后选择 **New Watch**（新建监视）。
   **注：**对 **Rx** 执行相同操作。

**图 5-5.** 添加 New Watch



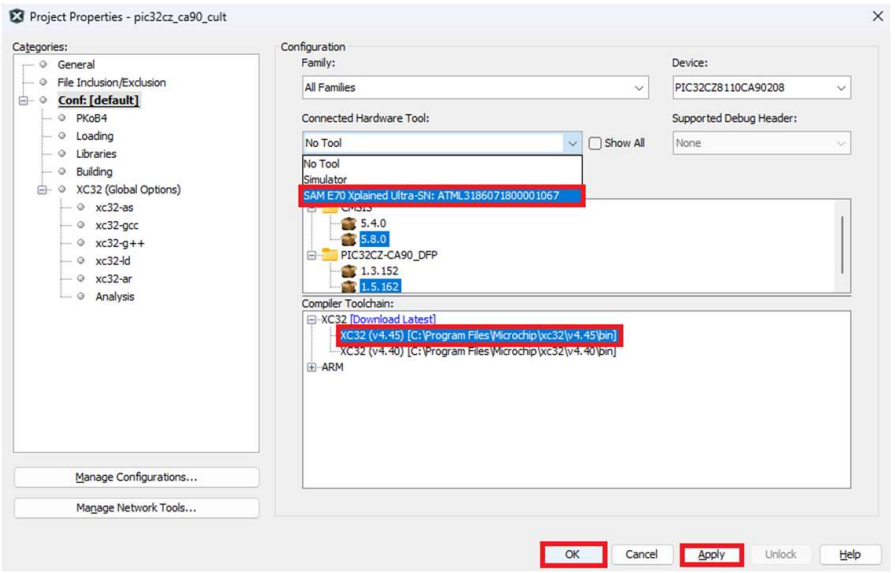

6. 在 MPLAB X IDE 的“Project Properties”（项目属性）窗口中，执行以下操作。
   a. 在左侧的“Categories”部分下，选择 **Conf: [default]**（配置：[默认]），并在右侧的 **Configuration**（配置）属性页面中，选择“Connected Hardware Tool and Compiler Toolchain”（已连接的硬件工具和编译器工具链），如下所示。

**图 5-6.** 项目属性——PIC32CZ CA90 Curiosity Ultra 开发板



7. 对于 SAM E70 Xplained Ultra 评估工具包，需要按如下所示更改硬件工具。

**图 5-7.** 项目属性——SAM E70 Xplained Ultra 评估工具包




**注：**以下步骤适用于两种板（PIC32CZ CA90 Curiosity Ultra 开发板和 SAM E70 Xplained Ultra 评估工具包）。

8. 单击 **Apply**（应用）然后单击 **OK**（确定）。
9. 单击下图所示的高亮位置以添加断点。
   **注：**在接收传输处理程序中添加端点，可实现接收数据的可视化查看。

**图 5-8.** 在接收完成中添加断点

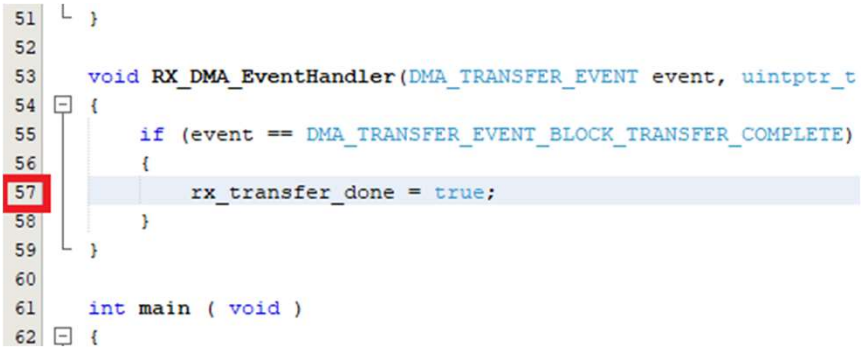
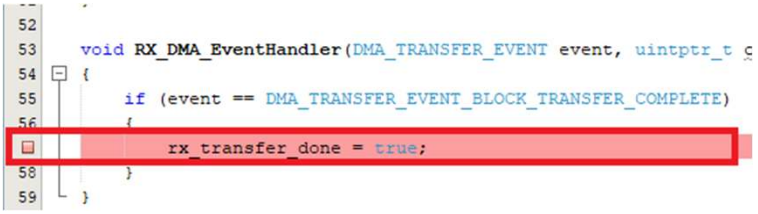



**图 5-9.** 接收完成回调处的断点



10. 单击 **Debug main Project**（调试主项目）按钮。

**图 5-10.** 调试主项目


**注：**如果“Variable”（变量）窗口未显示，请前往 *Window* > *Debugging* > *Variables*（窗口 > 调试 > 变量），打开“Variable”窗口。

**图 5-11.** “Window”（窗口）菜单详细信息

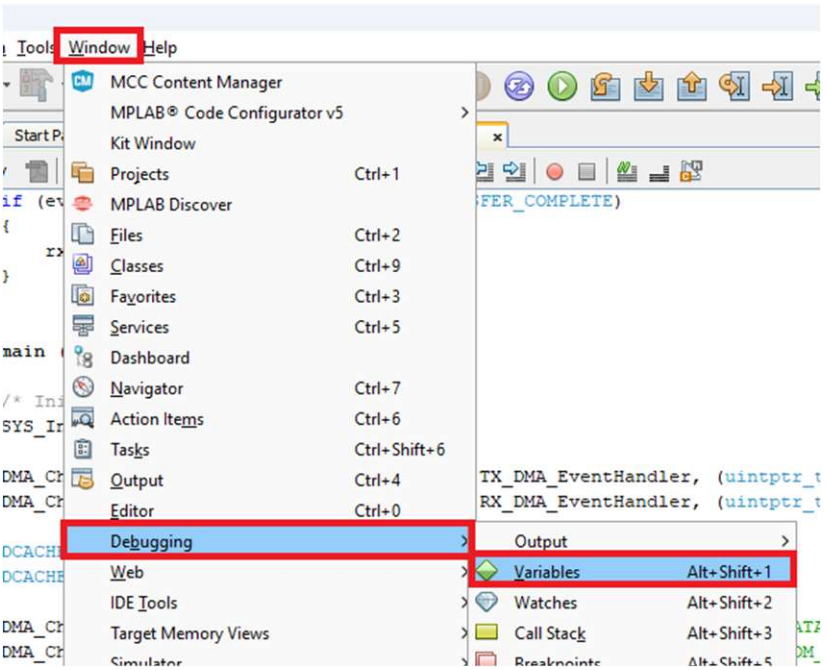
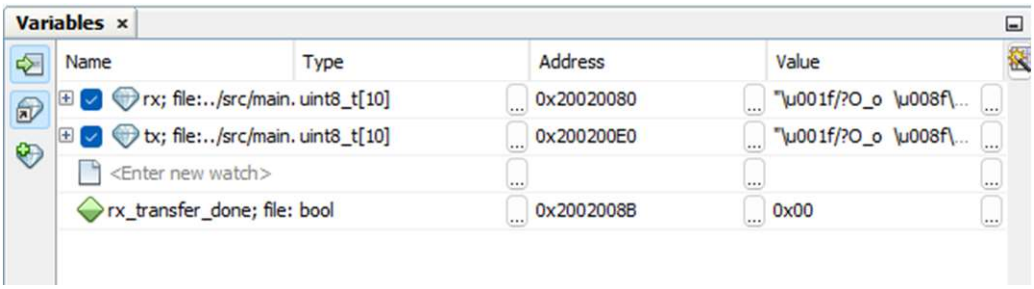
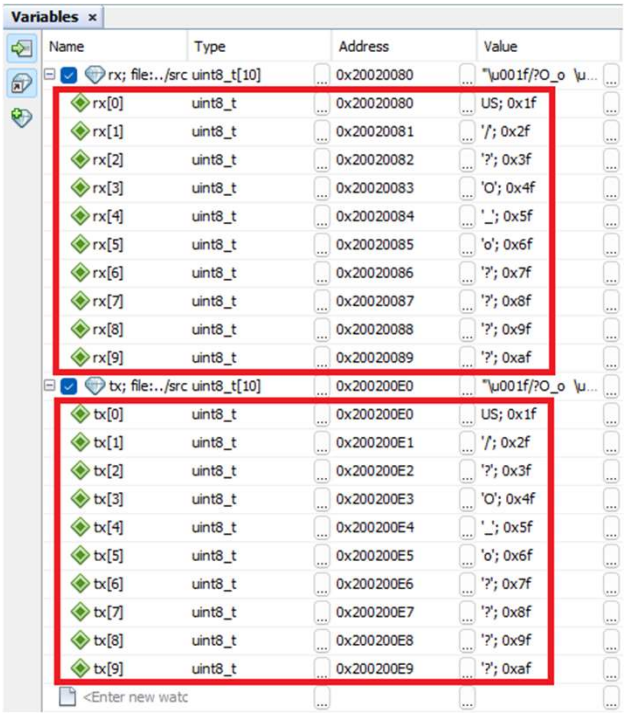



**图 5-12.** “Variable”窗口



11. 发送的数据（Tx）与接收的数据（Rx）一致。

**图 5-13.** 输出



# 6. 参考资料

- PIC32CZ CA90 Curiosity Ultra 开发板
- SAM E70 Xplained Ultra 评估工具包
- PIC32CZ CA80/CA90 Curiosity Ultra 用户指南（DS70005522）
- SAM E70 Xplained Ultra 用户指南（DS70005389B_CN）
- 使用 PIC32CZ CA90 Curiosity Ultra 评估板演示应用程序开发
- PIC32CZ CA90 Curiosity Ultra 开发板上的入门扩展应用
- 使用 MPLAB Harmony v3 通过 SAM E70 创建第一个项目
- SAM E70/S70/V70/V71 MCU 上的 MPLAB Harmony v3 驱动程序和系统服务使用入门
- 有关 32 位单片机相关资料和解决方案的更多信息，请参见：32 位单片机相关资料和解决方案参考指南（DS70005534）
- 有关 MPLAB Harmony v3 的更多信息，请参见 Microchip 网站：https://www.microchip.com/en-us/tools-resources/con igure/mplab-harmony 和 https://developerhelp.microchip.com/xwiki/bin/view/software-tools/harmony/
- 有关各种应用程序的更多信息，请参见：github.com/Microchip-MPLAB-Harmony/reference_apps
- 如需了解其他相关信息，请参见 Microchip 网站： www.microchip.com/

# 7. 版本历史

## 7.1. 版本 A——2025 年 1 月

这是本文档的初始版本。

# Microchip 信息

## 商标

“Microchip”的名称和徽标组合、“M”徽标及其他名称、徽标和品牌均为 Microchip Technology Incorporated 或其关联公司和/或子公司在美国和/或其他国家或地区的注册商标或商标（“Microchip 商标”）。有关 Microchip 商标的信息，可访问 https://www.microchip.com/en-us/about/legal-information/microchip-trademarks。

ISBN： 979-8-3371-1944-1

## 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。

本出版物及其提供的信息仅适用于 Microchip 产品，包括设计、测试以及将 Microchip 产品集成到您的应用中。以其他任何方式使用这些信息都将被视为违反条款。本出版物中的器件应用信息仅为您提供便利，将来可能会发生更新。您须自行确保应用符合您的规范。如需额外的支持，请联系当地的 Microchip 销售办事处，或访问 www.microchip.com/en-us/support/design-help/client-support-services。

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、偶然的或附带的损失、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。

## Microchip 器件代码保护功能

请注意以下有关 Microchip 产品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：在正常使用且符合工作规范的情况下，Microchip 系列产品非常安全。
- Microchip 注重并积极保护其知识产权。严禁任何试图破坏 Microchip 产品代码保护功能的行为，这种行为可能会违反《数字千年版权法案》（Digital Millennium Copyright Act）。
- 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。

# 产品页链接

ATSAME51G18A、ATSAME51G19A、ATSAME51J18A、ATSAME51J19A、ATSAME51J20A、ATSAME51N19A、ATSAME51N20A、ATSAME53J18A、ATSAME53J19A、ATSAME53J20A、ATSAMS70J19、ATSAMS70J20、ATSAMS70J21、ATSAMS70N19、ATSAMS70N20、ATSAMS70N21、ATSAMS70Q19、ATSAMS70Q20、ATSAMS70Q21、ATSAMV70J19、ATSAMV70J20、ATSAMV70N19、ATSAMV70N20、ATSAMV70Q19、ATSAMV70Q20、ATSAMV71J19、ATSAMV71J20、ATSAMV71J21、ATSAMV71N19、PIC32CZ2051CA70064、PIC32CZ2051CA70100、PIC32CZ2051CA70144、PIC32CZ2051CA80100、PIC32CZ2051CA80144、PIC32CZ2051CA80176、PIC32CZ2051CA80208、PIC32CZ2051CA90100、PIC32CZ2051CA90144 和 PIC32CZ2051CA90176