VS Code NuCodeGen 代码生成

NuMicro® 32 位系列微控制器范例代码介绍

文件信息

应用简述	本范例介绍如何使用VS Code扩展功能NuCodeGen来生成M55M1代码
BSP 版本	M55M1_Series_BSP_CMSIS_V3.01.003
开发平台	NuMaker-X-M55M1

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

这篇文章说明如何使用 VS Code 扩展功能 NuCodeGen，通过大型语言模型（Large Language Model）的能力自动生成完整代码，以提升开发效率并降低人工编码成本。内容首先通过 UART 示例说明 NuCodeGen 的基本操作，接着以 I²C 示例示范完整的代码生成步骤，其中 I²C 示例会附上对话记录和最终代码，以便清楚展现代码生成的各个阶段。

本示例以 VS Code 为开发环境，搭配 Nuvoton NuMicro Cortex-M Pack 开发工具套件（内含 NuCodeGen）进行操作。在开始实作前，请先于 VS Code 中完成该套件的安装。

虽然本篇以 NuMaker-X-M55M1 作为示范，但内容介绍的流程和方法同样适用于所有 Nuvoton NuMicro 系列微控制器，包含 M0、M23、M4 等系列产品。

更多的范例可以参考 NuCodegen 范例提示词。

1.1 原理

NuCodeGen 是一款基于 GitHub Copilot 的 AI Agent，具备以下核心特性：

- **自然语言生成代码**：使用者可直接以自然语言描述开发需求，NuCodeGen 即可自动生成可编译的代码，大幅提升开发效率。
- **快速查询技术资讯**：能够即时回答外设 IP 介绍、API 列表与使用方式等技术问题，无需额外查阅标准驱动程序库文件。
- **自动修正编译错误**：可解析编译阶段的错误信息并直接修改代码，提高调试效率。
- **整合式 AI 开发闭环**：结合 GitHub Copilot 与 NuMicro 开发工具，支持一站式 Build、Load、Debug，并可由 AI 自动执行命令，形成完整的闭环式开发流程。

通过整合代码生成、技术查询与错误处理能力，NuCodeGen 能显著提升 NuMicro 项目的开发效率，并有效降低人力投入。

1.2 扩展功能设置

1.2.1 安装扩展

安装 Nuvoton NuMicro Cortex-M Pack 开发工具套件（内含 NuCodeGen）需通过 VS Code 的网络机制进行安装与资源加载。若处于受控网络环境（如企业防火墙或 Proxy），请确保网络访问权限至少包含以下核心域名，以维持扩展功能的基本运行：

- **marketplace.visualstudio.com**：用于搜索扩展功能并获取版本信息。
- ***.gallerycdn.vsassets.io**：用于下载扩展功能安装包（.vsix）。
- ***.vscode-cdn.net**：用于加载 VS Code 相关的 CDN 资源。

所有联机皆必须通过 HTTPS（TCP 埠 443）进行。

实际连接需求可能会因 VS Code 版本或环境配置不同而有所差异。有关更完整的网络需求、Proxy 与防火墙设置说明，请参阅 VS Code 官方文档：VS Code 官方文档：Network Setup。

1.2.2 启用扩展

NuCodeGen 在一般情况下不会在 VS Code 中自动启用，仅在 CMSIS-Csolution 项目中才会生效，例如 BSP 项目。若为新建项目，需先完成 CMSIS-Csolution 的初始化流程，NuCodeGen 的相关功能才会被启用。

1.3 支援 BSP 结构介绍

用户可自行从 OpenNuvoton 下载 BSP，并在 VS Code 中打开项目，具体打开方式请参考范例程序执行的章节。需特别注意，必须依照范例程序执行的章节所述流程，打开与 *main.c* 位于同一层级的 *VSCode* 文件夹。

NuCodeGen 支持图 1-1 所示结构的 BSP，在执行期间会自动會自動从目前打开的工程路径向上层搜索 *Library* 文件夹中的标准驱动程序库文件（Standard Driver Library Files），因此，当开启任一 BSP 所属的工程时，系统将自动使用该 BSP 目录下的 *Library* 文件夹。

BSP	
Library	Sample code header and source files
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	

图 1-1 NuCodeGen 支持的 BSP 结构

由于 NuCodeGen 是在本地端 BSP 环境中运行，若 BSP 发布新版，请先至 OpenNuvoton 下载并更新至最新版本后再使用。

1.4 操作介绍

在 GitHub Copilot Chat 的使用界面中，请点击 Agent 图示，如图 1-2 所示，请选择 NuCodeGen 这个自定义 Agent，以使用 NuCodeGen 扩展功能，如图 1-3 所示。本章节范例使用 Claude Haiku 4.5 模型进行操作。

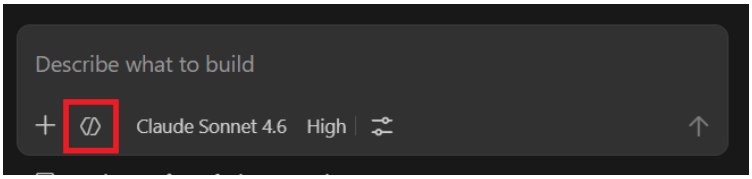

图 1-2 Agent 图示

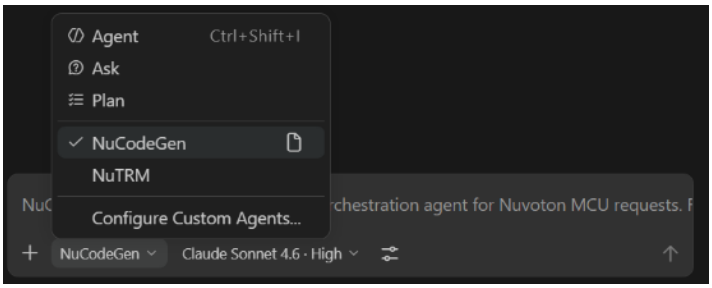

图 1-3 NuCodeGen 使用

1.4.1 提示词生成

提示词（prompt）的质量会直接影响生成代码的正确性。若提示内容过于简略，可能导致关键配置被忽略，甚至产生错误设置，进而造成程序运行问题并增加调试时间。

因此，NuCodeGen 提供 **/<外设名称>-settings** 这个指令，如图 1-4 所示，用于列出特定外设的相关设定项目。当使用者输入此指令时，NuCodeGen 会根据目标微控制器的标准驱动程序库，自动生成该外设可设置的配置清单和示例提示词，如图 1-5 和图 1-6 所示，协助开发者依据需求客制化更完善的提示内容。

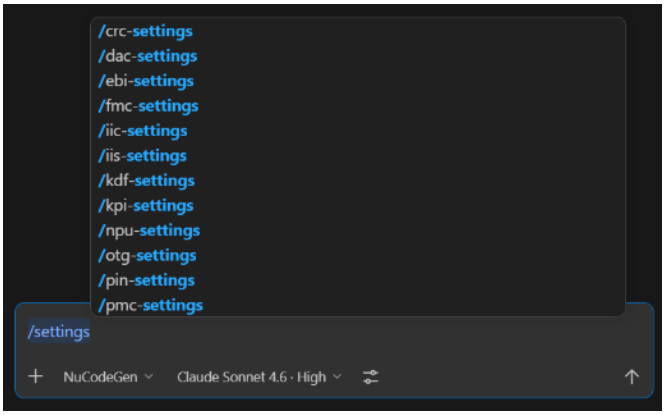

图 1-4 <外设名称>-settings 指令

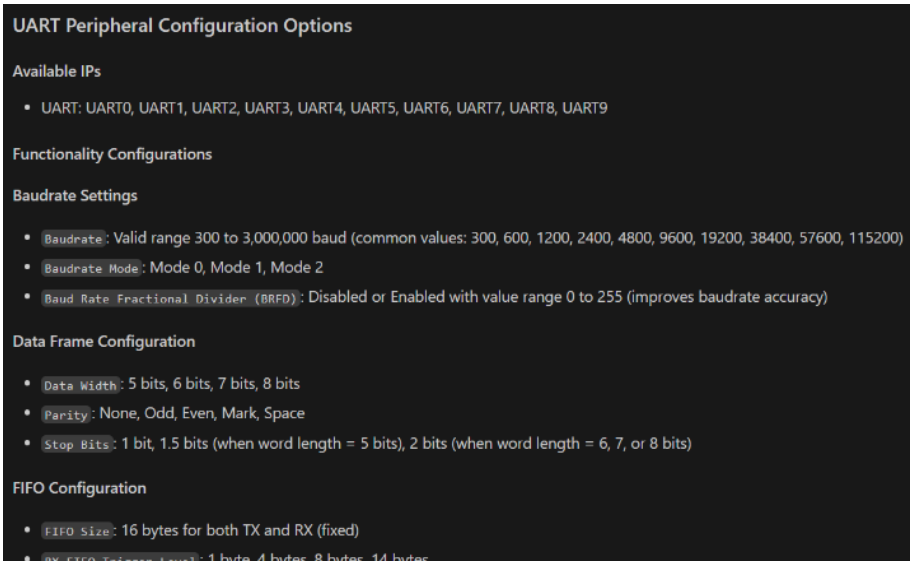

图 1-5 UART 可设定的配置

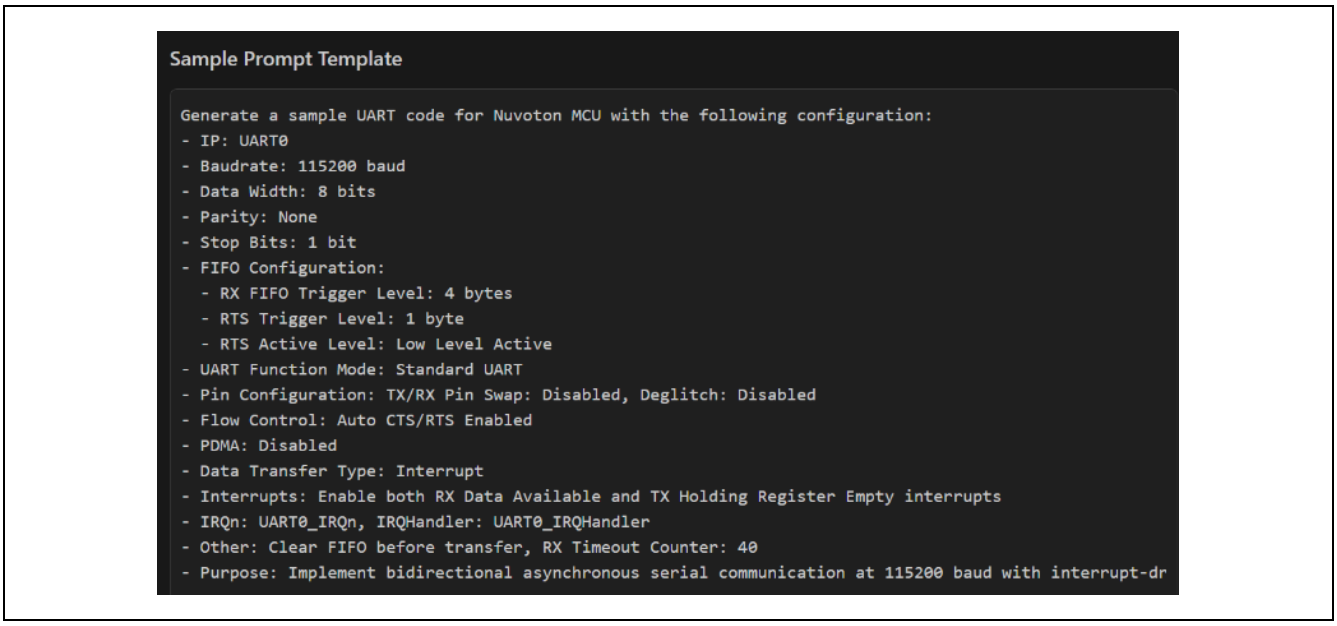

图 1-6 UART 示例提示词

1.4.2 代码生成

当提示词生成后，即可用于生成对应的程序代码。NuCodeGen 会提供一组工具供 AI 使用，使 AI 能够自行判断并选择合适的工具，从标准驱动程序库（其导入机制如支援 BSP 结构介绍的章节所述）中获取所需信息，例如函数名称或宏定义，并根据提示内容生成程序代码。程序代码生成后，NuCodeGen 会自动执行编译，并根据编译结果进行必要的修正，直至编译成功。

通过上述两个步骤，开发者即可持续生成所需代码，直到完成所有外设、时钟与脚位的设定。图 1-7 示范使用 Claude Haiku 4.5 模型生成 UART 回路测试程序后的烧录结果，图 1-8 则呈现完整的代码生成流程。这个范例在产生完代码后，直接进行烧录，没有任何额外的代码修改。

关于 NuCodeGen 的其他详细说明与使用方式，请参考 NuCodeGen Overview 的章节。

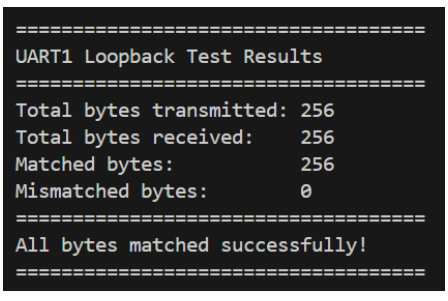

图 1-7 UART 回路测试烧录结果

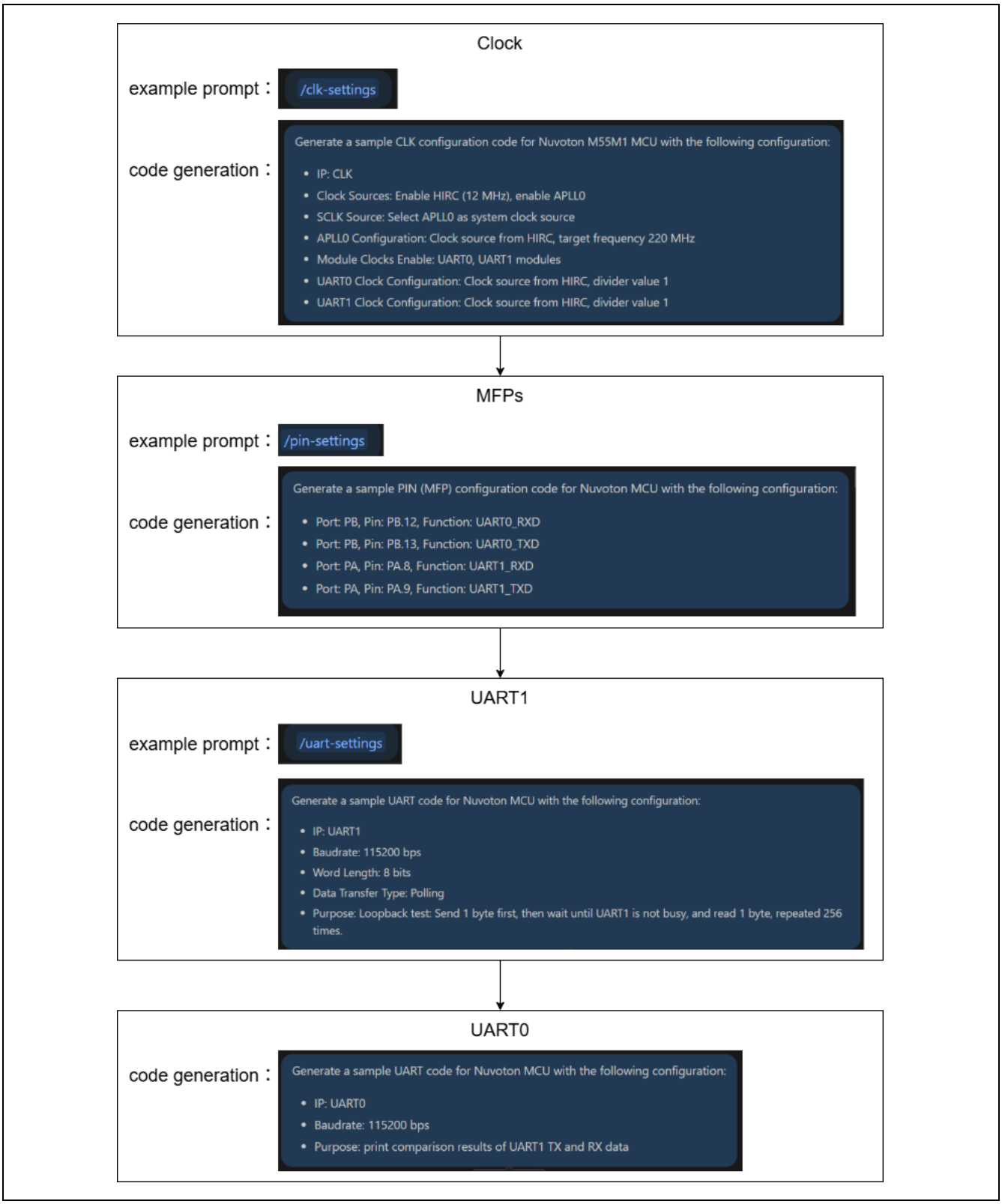

图 1-8 UART 回路测试代码生成流程

1.5 模型选择

GitHub Copilot 各方案每月可使用的 premium requests 数量有所差异，例如目前免费方案每月提供 50 次、Pro 方案提供 300 次。建议开发者在使用时适度留意用量。

免费方案中，目前各模型的使用量计算方式相同，如图 1-9 所示，因此建议在生成外设、时钟与脚位相关程序代码时优先选用 Claude Haiku 4.5。

此外，使用 NuCodeGen Agent 发送请求时，premium requests 的计算方式与 GitHub Copilot 内建 Agent 相同。目前每次请求仅会消耗一次 premium request；若未来 Copilot 的计算方式有所调整，NuCodeGen 也将同步受到影响。

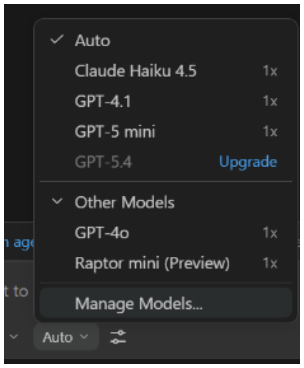

图 1-9 免费方案支持的模型

对于 GitHub Copilot 付费方案的开发者，可以根据需求的功能复杂度选择合适的模型。例如，在进行外设时钟、脚位设定或仅需通过 UART0 打印调试信息时，可选择 premium requests 较低的模型，如 Grok、Gemini 3 Flash 或 Claude Haiku 4.5。若需要产生外设功能代码，则可使用 premium requests ×1 的模型，例如 Claude Sonnet 4.6、Gemini 3.1 Pro 或 GPT 5.3 Cortex。对于更高复杂度的功能，也可以选择 premium requests 更高的模型，以生成更精准并符合需求的程式代码。

可以在 VS Code 的状态栏中查看 GitHub Copilot 的使用状况，如图 1-10 所示。未来各方案若有推出更新或更高性能的模型，亦可依需求调整使用。

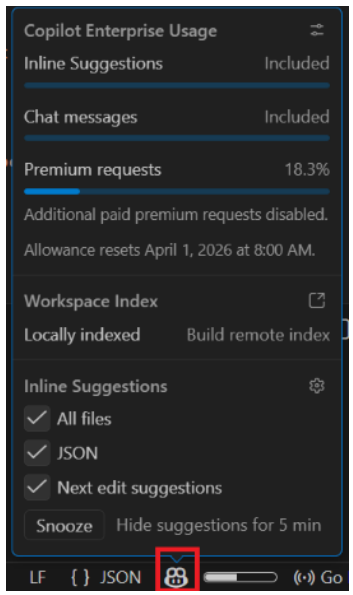

图 1-10 GitHub Copilot 使用状况

GitHub Copilot 各方案的费用、可使用的 premium requests 数量与支持的模型可能会随时调整。如需查询最新资讯，可参考官方文件 GitHub Copilot Plans & pricing。

2. 代码介绍

本章节以 I²C 示例说明完整的开发流程，以及各阶段提示词所产生的代码结果，所使用的模型为 Claude Sonnet 4.6。示例中设定 I2C0 为主机端、I2C1 为从机端，并进行两项测试：第一项测试为主机端传送 128 bytes 数据至从机地址 0x16；第二项测试则是主机端自从机地址 0x16 接收 128 bytes 数据。

完整的代码生成流程如图 2-1 所示。可先设置好需要使用的时钟与引脚，避免在生成 I²C、UART 功能程序代码时，AI 可能会自动补齐的设置与预期使用的时钟或引脚不一致。

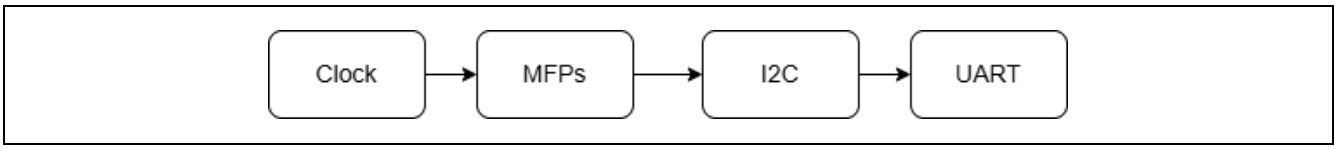

图 2-1 I²C 示例流程

最后，图 2-2 是在生成完整代码后直接烧录所得的结果。

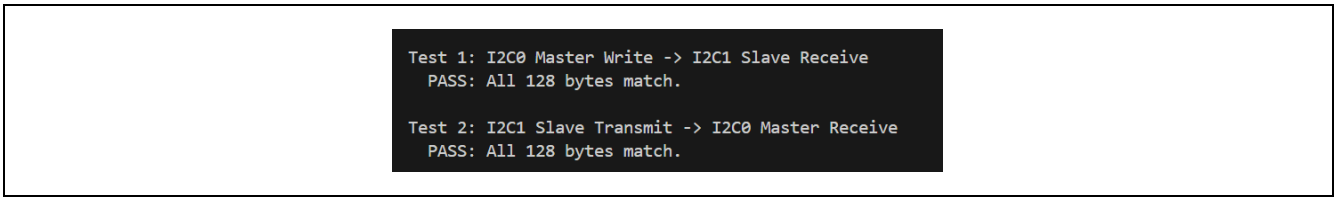

图 2-2 I²C 示例烧录结果

2.1 时钟代码生成

首先，根据图 2-3 所示的提示词生成系统时钟相关的程序代码。

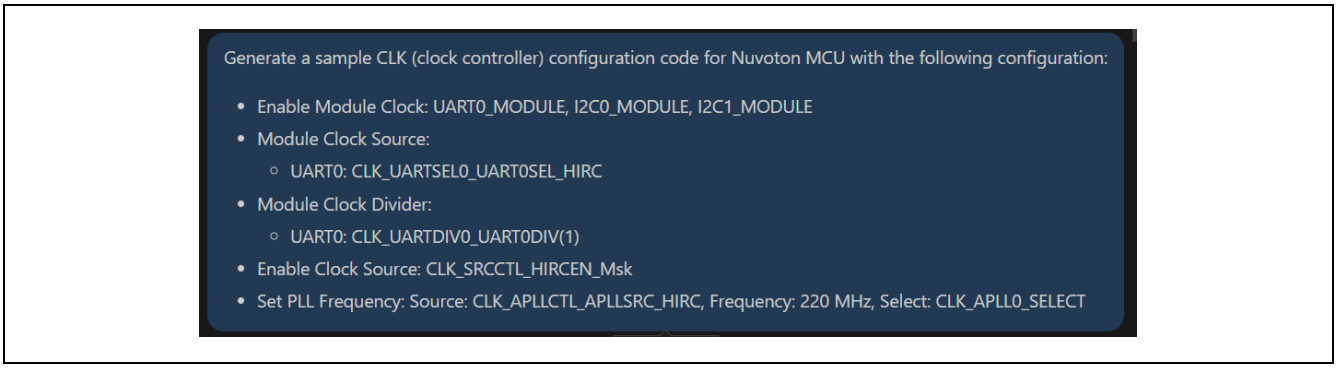

图 2-3 外设时钟配置提示词

```
void SYS_Init(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();
    /* Enable HIRC clock source */
    CLK_EnableXtalRC(CLK_SRCCTL_HIRCEN_Msk);
    /* Wait for HIRC clock source stable */
    CLK_WaitClockReady(CLK_STATUS_HIRCSTB_Msk);
    /* Set APLL0 clock source to HIRC and set frequency to 220 MHz */
    CLK_EnableAPLL(CLK_APLLCTL_APLLSRC_HIRC, 220000000UL, CLK_APLL0_SELECT);
    /* Wait for APLL0 clock source stable */
    CLK_WaitClockReady(CLK_STATUS_APLL0STB_Msk);
    /* Switch SCLK clock source to APLL0 */
    /* Power level and flash access cycle are set automatically */
    CLK_SetSCLK(CLK_SCLKSEL_SCLKSEL_APLL0);

    /* Update System Core Clock */
    SystemCoreClockUpdate();

    /* Set UART0 module clock source to HIRC and divider to 1 */
    CLK_SetModuleClock(UART0_MODULE, CLK_UARTSEL0_UART0SEL_HIRC,
CLK_UARTDIV0_UART0DIV(1));

    /* Enable UART0 module clock */
    CLK_EnableModuleClock(UART0_MODULE);

    /* Enable I2C0 module clock */
    CLK_EnableModuleClock(I2C0_MODULE);

    /* Enable I2C1 module clock */
    CLK_EnableModuleClock(I2C1_MODULE);

    /* Set PB.12 as UART0_RXD and PB.13 as UART0_TXD */
    /* Lock protected registers */
    SYS_LockReg();
}
```

2.2 引脚代码生成

接下来进行引脚配置，相关提示词如图 2-4 所示。

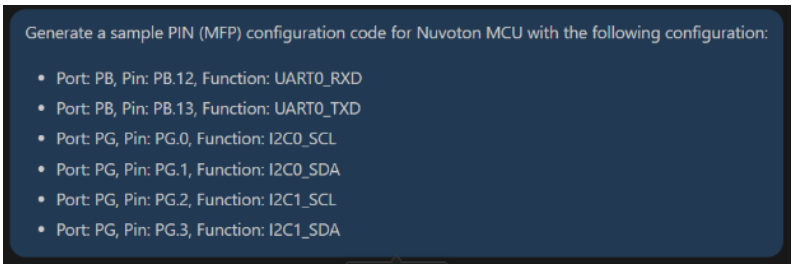

图 2-4 引脚配置提示词

```
void SYS_Init(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();
    /* Set PB.12 as UART0_RXD and PB.13 as UART0_TXD */
    SYS->GPB_MFP3 = (SYS->GPB_MFP3 & ~(SYS_GPB_MFP3_PB12MFP_Msk |
SYS_GPB_MFP3_PB13MFP_Msk)) |
                    (SYS_GPB_MFP3_PB12MFP_UART0_RXD | SYS_GPB_MFP3_PB13MFP_UART0_TXD);

    /* Set PG.0 as I2C0_SCL, PG.1 as I2C0_SDA, PG.2 as I2C1_SCL, PG.3 as I2C1_SDA */
    SYS->GPG_MFP0 = (SYS->GPG_MFP0 & ~(SYS_GPG_MFP0_PG0MFP_Msk | SYS_GPG_MFP0_PG1MFP_Msk |
                                       SYS_GPG_MFP0_PG2MFP_Msk |
SYS_GPG_MFP0_PG3MFP_Msk)) |
                    (SYS_GPG_MFP0_PG0MFP_I2C0_SCL | SYS_GPG_MFP0_PG1MFP_I2C0_SDA |
                     SYS_GPG_MFP0_PG2MFP_I2C1_SCL | SYS_GPG_MFP0_PG3MFP_I2C1_SDA);

    /* Lock protected registers */
    SYS_LockReg();
}
```

2.3 I²C 代码生成

首先，根据图 2-5 所示的提示词生成 I2C 相关代码，其中涵盖 I2C0 与 I2C1 的必要设置及其对应的功能实现。

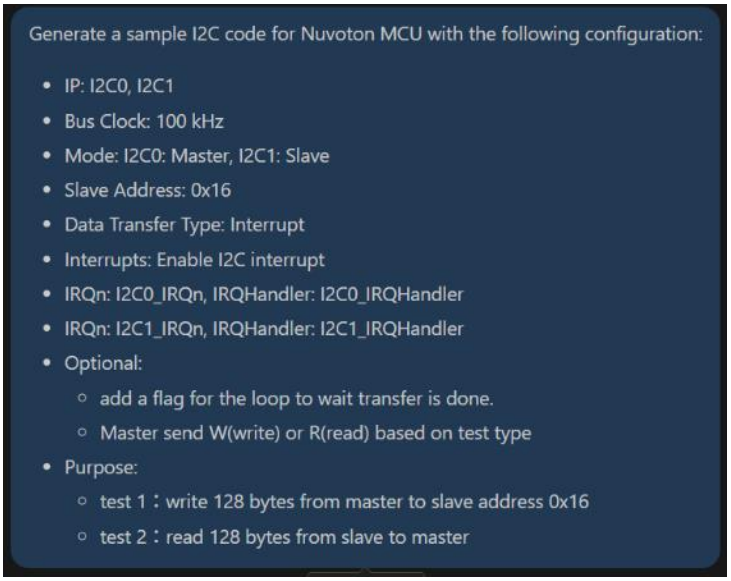

图 2-5 I²C 提示词

这段提示词生成的代码其中就包含：

- 代码中使用到的宏定义和变量。

```
/*---------------------------------------------------------------------------------------*/
```

```
/* Defines                                                                                */
/*----------------------------------------------------------------------------------------*/
#define TRANSFER_SIZE   128
#define SLAVE_ADDRESS   0x16

/*----------------------------------------------------------------------------------------*/
/* Global variables (volatile: shared between main and ISR)                               */
/*----------------------------------------------------------------------------------------*/
/* Test type: 0 = Master Write / Slave Receive, 1 = Master Read / Slave Transmit */
volatile uint8_t  g_u8TestType = 0;

/* Master TX/RX buffers and transfer state */
volatile uint8_t  g_au8MasterTxBuf[TRANSFER_SIZE];
volatile uint8_t  g_au8MasterRxBuf[TRANSFER_SIZE];
volatile uint32_t g_u32MasterDataLen = 0;
volatile uint8_t  g_u8MasterEndFlag  = 0;  /* Set by ISR when master transfer completes */

/* Slave TX/RX buffers and transfer state */
volatile uint8_t  g_au8SlaveTxBuf[TRANSFER_SIZE];
volatile uint8_t  g_au8SlaveRxBuf[TRANSFER_SIZE];
volatile uint32_t g_u32SlaveDataLen  = 0;
volatile uint8_t  g_u8SlaveEndFlag   = 0;  /* Set by ISR when slave transfer completes */
```

- I²C 的初始化

```
/*----------------------------------------------------------------------------------------*/
/* I2C Initialization: I2C0 (Master) and I2C1 (Slave at address 0x16)                     */
/*----------------------------------------------------------------------------------------*/
void I2C_Init(void)
{
    /* Open I2C0 with 100 kHz bus clock (Master) */
    I2C_Open(I2C0, 100000);

    /* Open I2C1 with 100 kHz bus clock (Slave) */
    I2C_Open(I2C1, 100000);

    /* Configure I2C1 slave address as 0x16, GC mode disabled */
    I2C_SetSlaveAddr(I2C1, 0, SLAVE_ADDRESS, I2C_GCMODE_DISABLE);

    /* Enable I2C0 interrupt and register NVIC */
    I2C_EnableInt(I2C0);
    NVIC_EnableIRQ(I2C0_IRQn);

    /* Enable I2C1 interrupt and register NVIC */
    I2C_EnableInt(I2C1);
    NVIC_EnableIRQ(I2C1_IRQn);

    /* Enter I2C1 (Slave) into non-addressed slave mode, ready to respond */
    I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
}
```

- I²C 的中断处理负责根据 I²C 主机端与从机端的不同状态，执行相应的流程并更新必要的控制信号。

```
/*----------------------------------------------------------------------------------------*/
/* I2C0 IRQ Handler (Master)                                                              */
/*----------------------------------------------------------------------------------------*/
void I2C0_IRQHandler(void)
{
```

```
    uint32_t u32Status = I2C_GET_STATUS(I2C0);

    if (g_u8TestType == 0)
    {
        /* Test 1: Master Write 128 bytes to slave */
        switch (u32Status)
        {
            case 0x08:  /* START transmitted */
                I2C_SET_DATA(I2C0, (uint8_t)((SLAVE_ADDRESS << 1) | 0x00));  /* SLA+W */
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI);
                break;

            case 0x18:  /* SLA+W transmitted, ACK received */
                I2C_SET_DATA(I2C0, g_au8MasterTxBuf[g_u32MasterDataLen++]);
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI);
                break;

            case 0x28:  /* Data byte transmitted, ACK received */
                if (g_u32MasterDataLen < TRANSFER_SIZE)
                {
                    I2C_SET_DATA(I2C0, g_au8MasterTxBuf[g_u32MasterDataLen++]);
                    I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI);
                }
                else
                {
                    /* All bytes sent, generate STOP */
                    I2C_SET_CONTROL_REG(I2C0, I2C_CTL_STO_SI);
                    g_u8MasterEndFlag = 1;
                }
                break;

            case 0x20:  /* SLA+W transmitted, NACK received */
            case 0x30:  /* Data byte transmitted, NACK received */
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_STO_SI);
                g_u8MasterEndFlag = 1;
                break;

            case 0x38:  /* Arbitration lost */
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI);
                break;

            default:
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_STO_SI);
                break;
        }
    }
    else
    {
        /* Test 2: Master Read 128 bytes from slave */
        switch (u32Status)
        {
            case 0x08:  /* START transmitted */
                I2C_SET_DATA(I2C0, (uint8_t)((SLAVE_ADDRESS << 1) | 0x01));  /* SLA+R */
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI);
                break;
```

```
            case 0x40:  /* SLA+R transmitted, ACK received */
                /* Set AA=1 to return ACK for the incoming bytes (more than 1 byte) */
                if (TRANSFER_SIZE > 1)
                    I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI_AA);
                else
                    I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI);
                break;

            case 0x50:  /* Data byte received, ACK returned */
                g_au8MasterRxBuf[g_u32MasterDataLen++] = (uint8_t)I2C_GET_DATA(I2C0);
                /* Return NACK for the last byte (index TRANSFER_SIZE-1) */
                if (g_u32MasterDataLen < (TRANSFER_SIZE - 1))
                    I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI_AA);
                else
                    I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI);  /* Next byte will be NACK'd */
                break;

            case 0x58:  /* Data byte received, NACK returned (last byte) */
                g_au8MasterRxBuf[g_u32MasterDataLen++] = (uint8_t)I2C_GET_DATA(I2C0);
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_STO_SI);
                g_u8MasterEndFlag = 1;
                break;

            case 0x48:  /* SLA+R transmitted, NACK received */
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_STO_SI);
                g_u8MasterEndFlag = 1;
                break;

            case 0x38:  /* Arbitration lost */
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI);
                break;

            default:
                I2C_SET_CONTROL_REG(I2C0, I2C_CTL_STO_SI);
                break;
        }
    }

    /* M55M1: Read status register again to force write-clear propagation through the
interconnect */
    u32Status = I2C_GET_STATUS(I2C0);
}

/*---------------------------------------------------------------------------------------*/
/* I2C1 IRQ Handler (Slave)                                                              */
/*---------------------------------------------------------------------------------------*/
void I2C1_IRQHandler(void)
{
    uint32_t u32Status = I2C_GET_STATUS(I2C1);

    switch (u32Status)
    {
        case 0x60:  /* Own SLA+W received, ACK returned */
            g_u32SlaveDataLen = 0;
            I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
            break;
```

```
        case 0x80:  /* Data received, ACK returned */
            g_au8SlaveRxBuf[g_u32SlaveDataLen++] = (uint8_t)I2C_GET_DATA(I2C1);
            if (g_u32SlaveDataLen < TRANSFER_SIZE)
                I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
            else
                I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI);
            break;

        case 0xA0:  /* STOP or Repeated START received */
            I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
            if (g_u8TestType == 0)
                g_u8SlaveEndFlag = 1;  /* Test 1: slave receive complete */
            break;

        case 0xA8:  /* Own SLA+R received, ACK returned */
            g_u32SlaveDataLen = 0;
            I2C_SET_DATA(I2C1, g_au8SlaveTxBuf[g_u32SlaveDataLen++]);
            I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
            break;

        case 0xB8:  /* Data transmitted, ACK received (master wants more) */
            if (g_u32SlaveDataLen < TRANSFER_SIZE)
            {
                I2C_SET_DATA(I2C1, g_au8SlaveTxBuf[g_u32SlaveDataLen++]);
                I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
            }
            else
            {
                /* All bytes loaded; master controls end via NACK */
                I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
            }
            break;

        case 0xC0:  /* Data transmitted, NACK received (master done reading) */
            I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
            g_u8SlaveEndFlag = 1;
            break;

        case 0xC8:  /* Last data transmitted (AA=0), ACK received */
            I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
            g_u8SlaveEndFlag = 1;
            break;

        default:
            I2C_SET_CONTROL_REG(I2C1, I2C_CTL_SI_AA);
            break;
    }

    /* M55M1: Read status register again to force write-clear propagation through the
interconnect */
    u32Status = I2C_GET_STATUS(I2C1);
}
```

- 两项测试项目

```
/* Initialize I2C0 (Master) and I2C1 (Slave) */
```

```
    I2C_Init();
/*---------------------------------------------------------------------------*/
/* Test 1: I2C0 Master writes 128 bytes to I2C1 Slave (addr 0x16)            */
/*---------------------------------------------------------------------------*/
/* Prepare Master TX buffer with incrementing data */
for (i = 0; i < TRANSFER_SIZE; i++)
    g_au8MasterTxBuf[i] = (uint8_t)i;

/* Reset transfer state for Test 1 */
g_u8TestType       = 0;
g_u32MasterDataLen = 0;
g_u32SlaveDataLen  = 0;
g_u8MasterEndFlag  = 0;
g_u8SlaveEndFlag   = 0;

/* Master sends START to initiate write transfer */
I2C_SET_CONTROL_REG(I2C0, I2C_CTL_STA);

/* Wait for Master to complete */
while (g_u8MasterEndFlag == 0);
/* Wait for Slave to complete */
while (g_u8SlaveEndFlag  == 0);

/*---------------------------------------------------------------------------*/
/* Test 2: I2C0 Master reads 128 bytes from I2C1 Slave (addr 0x16)           */
/*---------------------------------------------------------------------------*/
/* Prepare Slave TX buffer with offset data */
for (i = 0; i < TRANSFER_SIZE; i++)
    g_au8SlaveTxBuf[i] = (uint8_t)(i + 0x80);

/* Reset transfer state for Test 2 */
g_u8TestType       = 1;
g_u32MasterDataLen = 0;
g_u32SlaveDataLen  = 0;
g_u8MasterEndFlag  = 0;
g_u8SlaveEndFlag   = 0;

/* Master sends START to initiate read transfer */
I2C_SET_CONTROL_REG(I2C0, I2C_CTL_STA);

/* Wait for Master to complete */
while (g_u8MasterEndFlag == 0);
/* Wait for Slave to complete */
while (g_u8SlaveEndFlag  == 0);
```

2.4 UART 代码生成

最后，启用 UART0 以打印执行结果，相关提示词如图 2-6 所示。

图 2-6 UART 提示词

```
int main(void)
{
```

```
    uint32_t i;

    /* Open UART0 at 115200 bps for standard I/O retargeting (printf) */
    UART_Open(UART0, 115200);

    /*---------------------------------------------------------------------------*/
    /* Print comparison results via UART0                                        */
    /*---------------------------------------------------------------------------*/
    /* Test 1: g_au8MasterTxBuf vs g_au8SlaveRxBuf */
    printf("\r\nTest 1: I2C0 Master Write -> I2C1 Slave Receive\r\n");
    for (i = 0; i < TRANSFER_SIZE; i++)
    {
        if (g_au8MasterTxBuf[i] != g_au8SlaveRxBuf[i])
        {
            printf("  FAIL at byte[%u]: MasterTx=0x%02X, SlaveRx=0x%02X\r\n",
                   (unsigned int)i, g_au8MasterTxBuf[i], g_au8SlaveRxBuf[i]);
            break;
        }
    }
    if (i == TRANSFER_SIZE)
        printf("  PASS: All %d bytes match.\r\n", TRANSFER_SIZE);

    /* Test 2: g_au8SlaveTxBuf vs g_au8MasterRxBuf */
    printf("\r\nTest 2: I2C1 Slave Transmit -> I2C0 Master Receive\r\n");
    for (i = 0; i < TRANSFER_SIZE; i++)
    {
        if (g_au8SlaveTxBuf[i] != g_au8MasterRxBuf[i])
        {
            printf("  FAIL at byte[%u]: SlaveTx=0x%02X, MasterRx=0x%02X\r\n",
                   (unsigned int)i, g_au8SlaveTxBuf[i], g_au8MasterRxBuf[i]);
            break;
        }
    }
    if (i == TRANSFER_SIZE)
        printf("  PASS: All %d bytes match.\r\n", TRANSFER_SIZE);

    while (1);
}
```

3. 软件与硬件需求

3.1 软件需求

- BSP 版本
  - M55M1_Series_BSP_CMSIS_V3.01.003
- IDE 版本
  - VS Code 1.110.0

3.2 硬件需求

- 电路组件
  - NuMaker-X-M55M1
- 线路示意图
  - 将 I2C0 的 SCL（PG.0）脚位连接到 I2C1 的 SCL（PG.2）。
  - 将 I2C0 的 SDA（PG.1）脚位连接到 I2C1 的 SDA（PG.3）。
  - 将 Nu-Link2-Me 的 USB 与电脑的 USB 相连接，用于显示范例代码的执行结果。

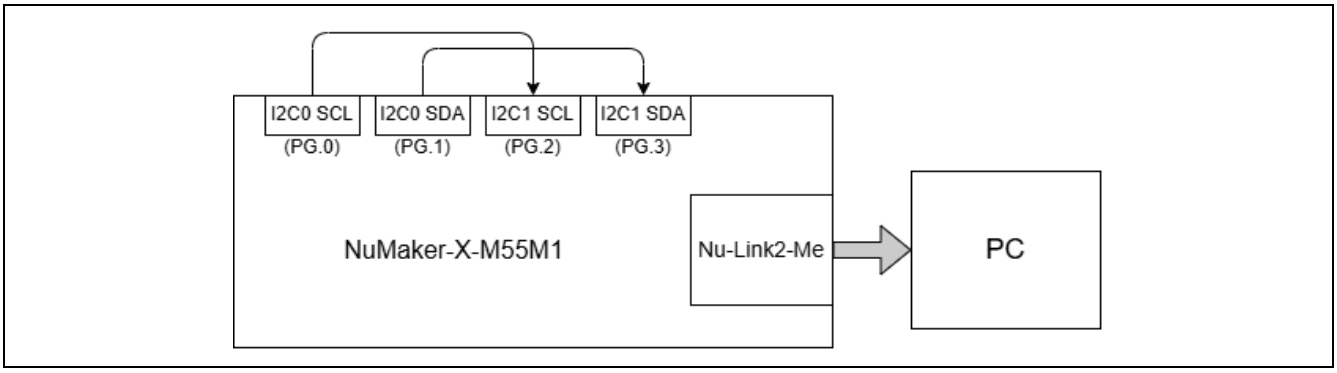

图 3-1 线路示意图

4. 目录信息

目录结构如下图所示。

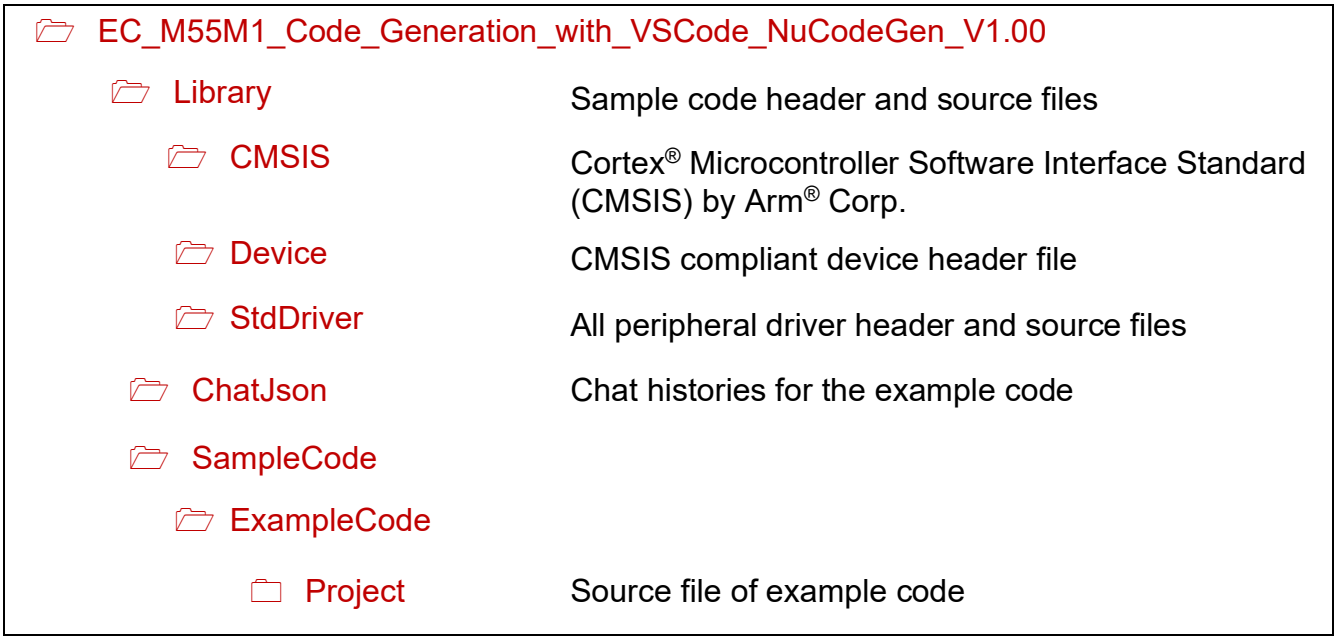

图 4-1 目录信息

5. Chat json 汇入

1. 在 VS Code 中，按下 **Ctrl+Shift+P**。

2. 搜索 Chat: Import Chat，如图 5-1 所示，选择要导入的 json 文件。

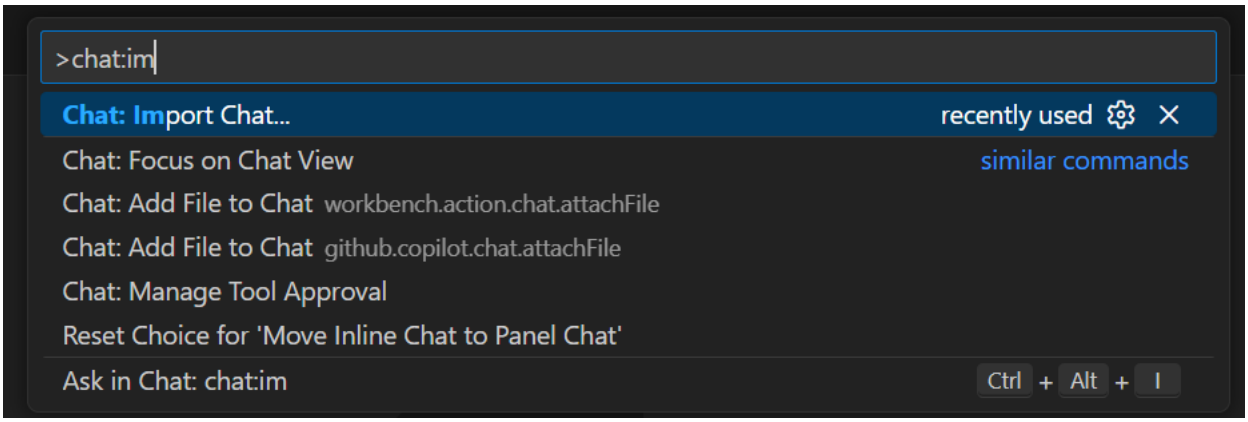

图 5-1 导入 json 文件

6. 范例程序执行

1. 根据目录信息章节，在 VS Code 中开启 *ExampleCode* 路径中的 *VSCode* 文件夹。
2. 点击 VS Code 窗口侧边栏的 CMSIS 图标，显示 CMSIS view
   - 编译
   - 下载代码至内存
   - 进入 / 离开仿真模式
3. 进入仿真模式界面
   - 执行代码

7. 修订纪录

Date	Revision	Description
2026.03.04	1.00	初始发布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.