VS Code NuCodeGen 程式碼生成

NuMicro® 32 位元系列微控制器範例程式碼介紹

文件資訊

應用簡述	本範例介紹如何使用 VS Code 擴充功能 NuCodeGen 來生成 M55M1 程式碼
BSP 版本	M55M1_Series_BSP_CMSIS_V3.01.003
開發平台	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

這篇文章說明如何使用 VS Code 擴充功能 NuCodeGen，透過大型語言模型（Large language model）的能力自動生成完整程式碼，以提升開發效率並降低人工編碼成本。內容首先透過 UART 範例說明 NuCodeGen 的基本操作，接著以 I2C 範例示範完整的程式碼生成步驟，其中 I2C 範例會附上對話紀錄和最終程式碼，以便清楚展現程式碼生成的各個階段。

本範例以 VS Code 為開發環境，搭配 Nuvoton NuMicro Cortex-M Pack 開發工具套件（內含 NuCodeGen）進行操作。在開始實作前，請先於 VS Code 中完成該套件的安裝。

雖然本篇以 NuMaker-X-M55M1 作為示範。但內容介紹的流程和方法同樣適用於所有 Nuvoton NuMicro 系列微控制器，包含 M0、M23、M4 等系列產品。

更多的範例可以參考 NuCodegen 範例提示詞。

1.1 原理

NuCodeGen 是一款基於 GitHub Copilot 的 AI agent，具備以下核心特性：

- **自然語言生成程式碼**：使用者可直接以自然語言描述開發需求，NuCodeGen 即可自動產生可編譯的程式碼，提升開發效率。
- **快速查詢技術資訊**：可即時回應外設 IP 介紹、API 列表與使用方式等技術問題，無需額外查閱標準驅動程式庫的檔案。
- **自動修正編譯錯誤**：能解析編譯階段的錯誤訊息並直接修正程式碼，提高除錯效率。
- **整合式 AI 開發閉環**：結合 GitHub Copilot 與 NuMicro 開發工具，支援一站式 Build、Load、Debug，並可由 AI 自動執行指令，形成完整的閉環式開發流程。

透過整合程式碼生成、技術查詢與錯誤處理能力，NuCodeGen 可大幅提升 NuMicro 專案的開發效率，並有效降低人力投入。

1.2 擴充功能設定

1.2.1 安裝擴充

安裝 Nuvoton NuMicro Cortex-M Pack 開發工具套件（內含 NuCodeGen）需透過 VS Code 的網路機制進行安裝與資源加載。若處於受控網路環境（如企業防火牆、Proxy），請確保網路存取權限至少包含以下核心網域，以維持擴充功能的基礎運作：

- **marketplace.visualstudio.com**：用於搜尋擴充功能與獲取版本資訊。
- ***.gallerycdn.vsassets.io**：用於下載擴充功能安裝包 (.vsix)。
- ***.vscode-cdn.net**：用於載入 VS Code 相關的 CDN 資源。

所有連線皆必須透過 HTTPS（TCP 埠 443）進行。

實際連線需求可能因 VS Code 版本或環境設定而異。更完整之網路需求、Proxy 與防火牆設定說明，請參考 VS Code 官方文件： VS Code 官方文件：Network Setup。

1.2.2 啟用擴充

NuCodeGen 在一般情況下不會於 VS Code 中自動啟用，僅在 CMSIS-Csolution 專案中才會生效，例如 BSP 專案。若為全新建立的專案，需先完成 CMSIS-Csolution 的初始化流程，NuCodeGen 的相關功能才會啟用。

1.3 支援 BSP 結構介紹

使用者可自 OpenNuvoton 下載 BSP，並於 VS Code 中開啟專案，開啟方式請參考範例程式執行的章節。需特別注意，必須依範例程式執行的章節所述流程，開啟與 *main.c* 位於同一層級的 *VSCode* 資料夾。

NuCodeGen 支援圖 1-1 所示結構的 BSP，在執行時間會自動從目前打開的專案路徑向上層搜尋 *Library* 資料夾中的標準驅動程式庫檔案（Standard Driver Library Files），因此，當開啟任一 BSP 所屬的專案時，系統將自動使用該 BSP 目錄下的 *Library* 資料夾。

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

圖 1-1 NuCodeGen 支援的 BSP 結構

由於NuCodeGen是在本地端BSP環境中運作，若 BSP 發布新版，請先至 OpenNuvoton 下載並更新至最新版本後再使用。

1.4 操作介紹

在 GitHub Copilot chat 使用介面上，請點擊 Agent 圖示，如圖 1-2 所示，並選擇 NuCodeGen 這個自訂 Agent，來使用NuCodeGen 擴充功能，如圖 1-3所示。本章節範例使用 Claude Haiku 4.5 模型進行操作。

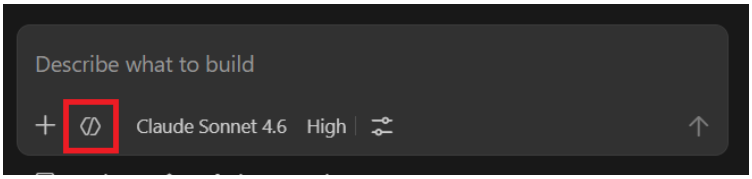

圖 1-2 agent 圖示

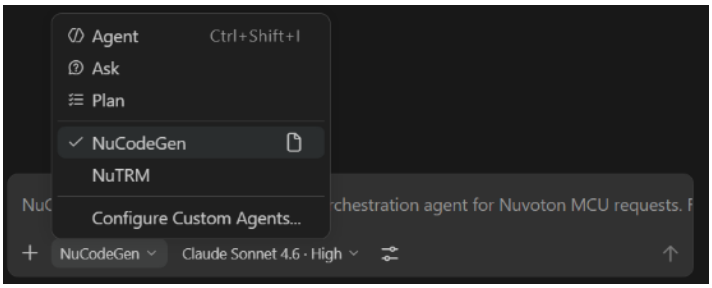

圖 1-3 NuCodeGen 使用

1.4.1 提示詞生成

提示詞（prompt）的品質會直接影響生成程式碼的正確性。若提示內容過於簡略，可能導致關鍵配置被忽略，甚至產生錯誤設定，進而造成程式運行問題並增加除錯時間。

因此，NuCodeGen 提供 **/<外設名稱>-settings** 這個指令，如圖 1-4 所示，用於列出特定外設的相關設定項目。當使用者輸入此指令時，NuCodeGen 會根據目標微控制器的標準驅動程式庫，自動生成該外設可設定的配置清單和範例提示詞，如圖 1-5 和圖 1-6 所示，協助開發者依需求客製化更完善的提示內容。

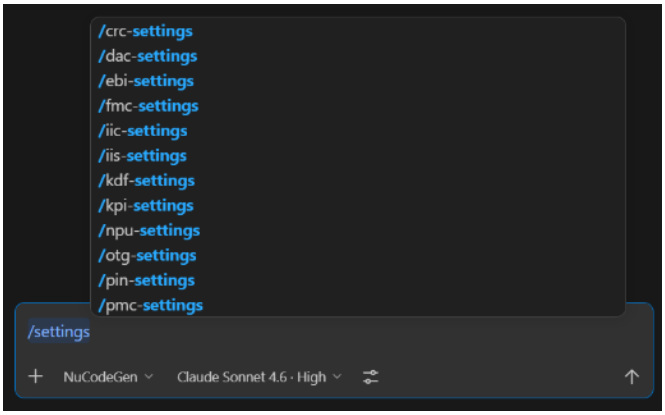

圖 1-4 <外設名稱>-settings 指令

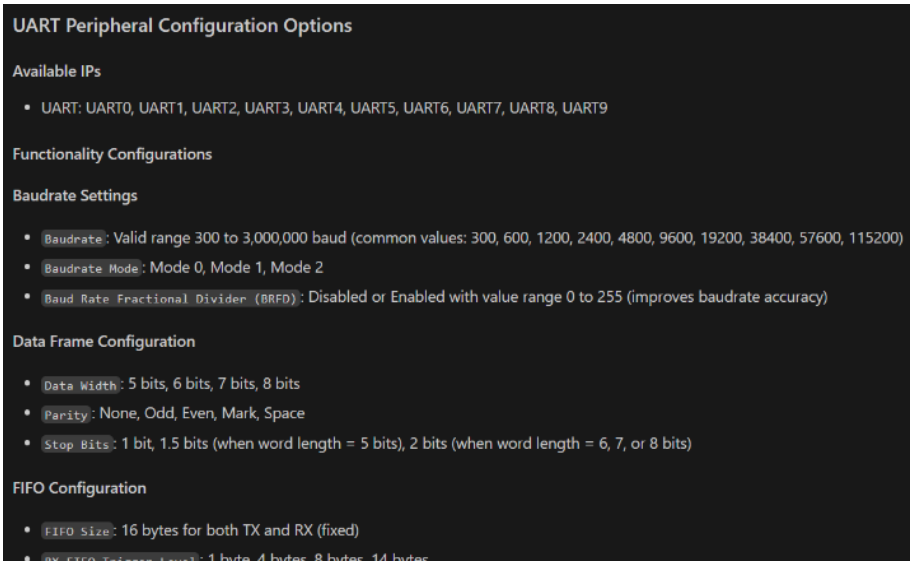

圖 1-5 UART 可設定的配置

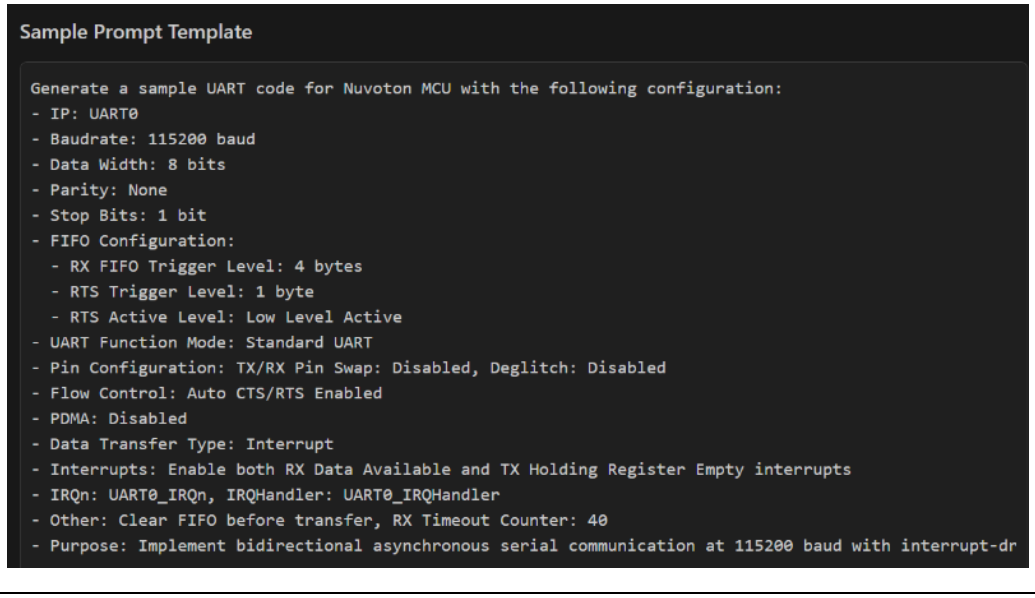

圖 1-6 UART 範例提示詞

1.4.2 程式碼生成

當提示詞生成後，即可用於產生對應的程式碼。NuCodeGen 會提供一組工具供 AI 使用，讓 AI 能夠自行判斷並選擇合適的工具，從標準驅動程式庫（其匯入機制如支援 BSP 結構介紹的章節所述）中取得所需資訊，例如函式名稱或巨集定義，並依提示內容產生程式碼。程式碼產生後，NuCodeGen 會自動執行編譯，並根據編譯結果進行必要的修正，直至編譯成功。

透過上述兩個步驟，開發者即可持續生成所需程式碼，直到完成所有外設、時鐘與腳位的設定。圖 1-7 示範使用 Claude Haiku 4.5 模型生成 UART 迴路測試程式後的燒錄結果，圖 1-8 則呈現完整的程式碼生成流程。這個範例在產生完程式碼後，直接進行燒錄，沒有任何額外的程式碼修改。

關於 NuCodeGen 的其他詳細說明與使用方式，請參考 NuCodeGen Overview 的章節。

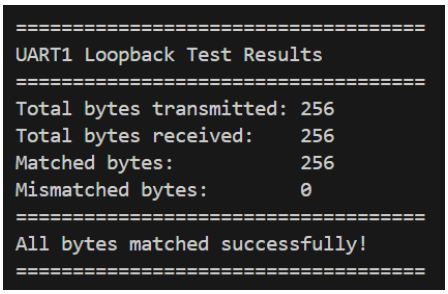

圖 1-7 UART 迴路測試燒錄結果

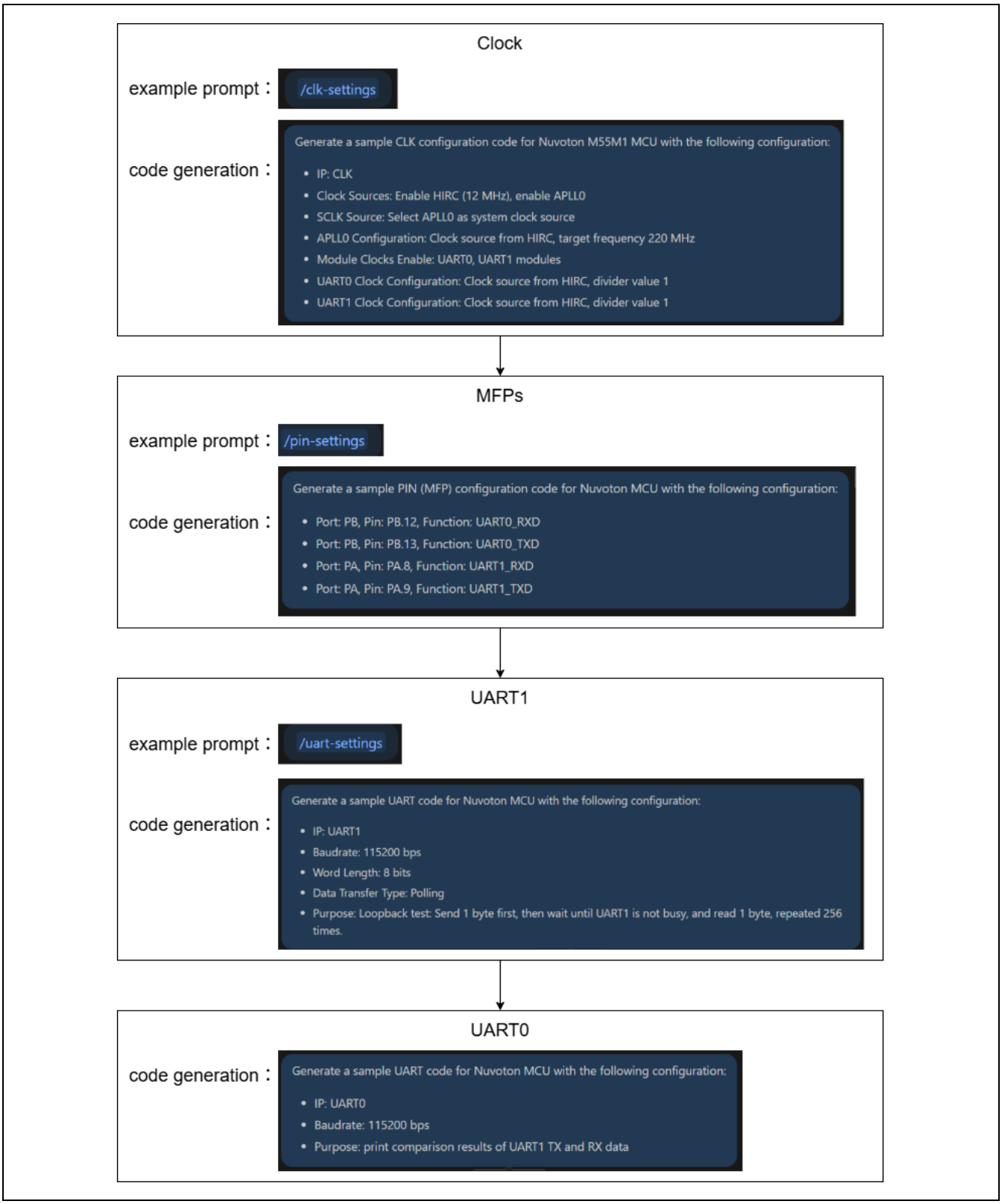

圖 1-8 UART 迴路測試程式碼生成流程

1.5 模型選擇

GitHub Copilot 各方案每月可使用的 premium requests 數量有所差異，例如目前免費方案每月提供 50 次、Pro 方案提供 300 次。建議開發者在使用時適度留意用量。

免費方案中，目前各模型的使用量計算方式相同，如圖 1-9 所示，因此建議在生成外設、時鐘與腳位相關程式碼時優先選用 Claude Haiku 4.5。

此外，使用 NuCodeGen agent 發送請求時，premium requests 的計算方式與 GitHub Copilot 內建 Agent 相同。目前每次請求僅會消耗一次 premium request；若未來 Copilot 的計算方式有所調整，NuCodeGen 也將同步受到影響。

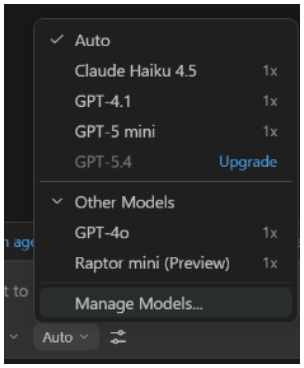

圖 1-9 免費方案支援的模型

對於 GitHub Copilot 付費方案的開發者，可以依據需求的功能複雜度選擇合適的模型。例如，在進行外設時鐘、腳位設定或僅需透過 UART0 輸出除錯訊息時，可選擇 premium requests 較低的模型，如 Grok、Gemini 3 Flash 或 Claude Haiku 4.5。若需要產生外設功能程式碼，則可使用 premium requests ×1 的模型，例如 Claude Sonnet 4.6、Gemini 3.1 Pro 或 GPT-5.3 Cortex。對於更高複雜度的功能，也可以選擇 premium requests 更高的模型，以生成更精準並符合需求的程式碼。

可以在 VS Code 的狀態欄中查看 GitHub Copilot 的使用狀況，如圖 1-10 所示。未來各方案若有推出更新或更高性能的模型，亦可依需求調整使用。

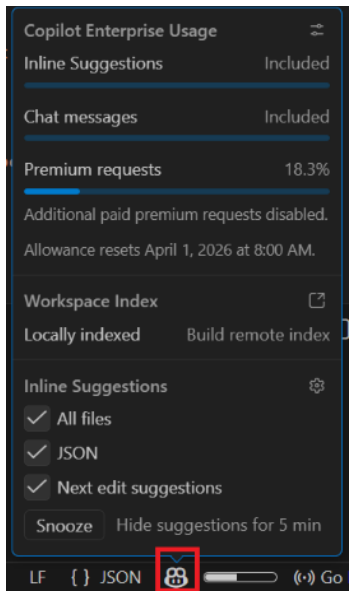

圖 1-10 GitHub Copilot 使用狀況

GitHub Copilot 各方案的費用、可使用的 premium requests 數量與支援的模型可能會隨時調整。如需查詢最新資訊，可參考官方文件 GitHub Copilot Plans & pricing。

2. 程式碼介紹

本章節以 I²C 範例說明完整的開發流程，以及各階段提示詞所產生的程式碼結果，所使用的模型為 Claude Sonnet 4.6。範例中設定 I2C0 為主機端、I2C1 為從機端，並實作兩項測試：第一項測試為主機端傳送 128 bytes 資料至從機位址 0x16；第二項測試則是主機端自從機位址 0x16 接收 128 bytes 資料。

整個程式碼生成的流程，如圖 2-1 所示。可先設定好需要使用的時鐘與腳位，避免在產生 I2C、UART 功能程式碼時，AI 可能會自動補齊的設定與預期使用的時鐘或腳位不一致。

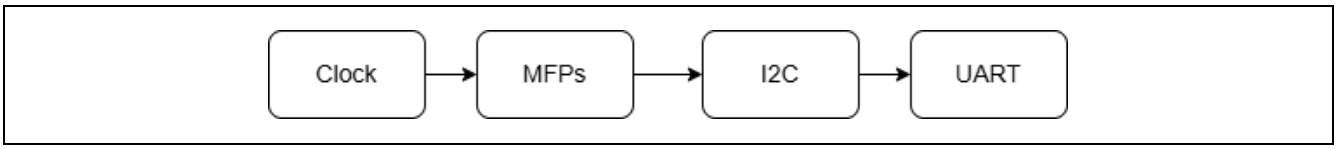

圖 2-1 I²C 範例流程

最後，圖 2-2是產生完整程式碼後直接燒錄後的結果。

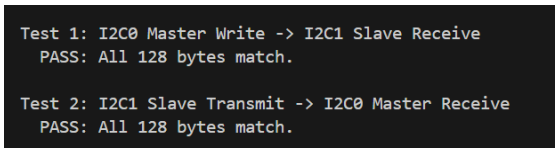

圖 2-2 I²C 範例燒錄結果

2.1 時鐘程式碼生成

首先，根據圖 2-3 所示的提示詞產生系統時鐘相關的程式碼。

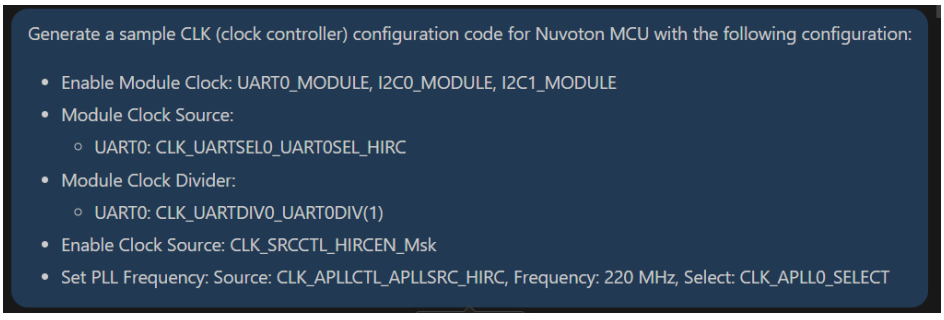

圖 2-3 外設時鐘配置提示詞

```
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 腳位程式碼生成

接下來進行腳位配置，相關提示詞如圖 2-4 所示。

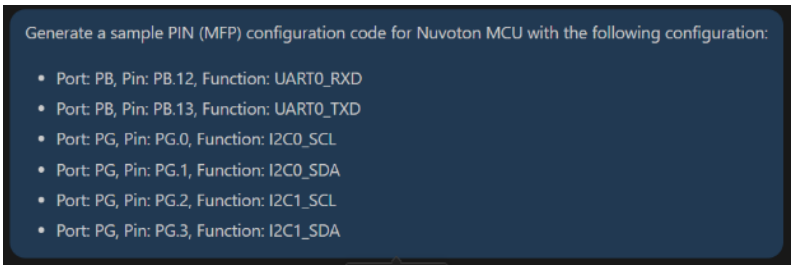

圖 2-4 腳位配置提示詞

```
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 程式碼生成

首先，根據圖 2-5 所示的提示詞生成 I²C 相關程式碼，其中涵蓋 I2C0 與 I2C1 的必要設定與其對應的功能實作。

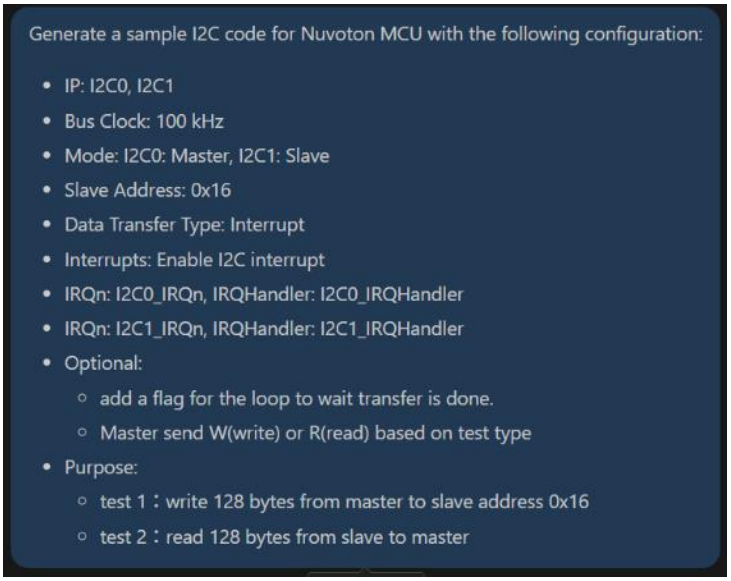

圖 2-5 I²C 提示詞

這段提示詞生成的程式碼其中就包含：

- 程式碼內使用到的巨集定義和變數。

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

```
/* 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 的中斷處理負責依據 I²C 主機端與從機端的不同狀態，執行相應的流程並更新必要的控制訊號。

```
/*----------------------------------------------------------------------------------------*/
/* 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);
}
```

- 兩項測試項目

```
/* 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 程式碼生成

最後，啟用 UART0 以輸出執行結果，相關提示詞如圖 2-6 所示。

圖 2-6 UART 提示詞

```
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. 軟體與硬體需求

3.1 軟體需求

- BSP 版本

  - M55M1_Series_BSP_CMSIS_V3.01.003

- IDE 版本

  - VS Code 1.110.0

3.2 硬體需求

- 電路元件

  - NuMaker-X-M55M1

- 線路示意圖

  - 將 I2C0 的 SCL（PG.0）腳位連接到 I2C1 的 SCL（PG.2）。

  - 將 I2C0 的 SDA（PG.1）腳位連接到 I2C1 的 SDA（PG.3）。

  - 將 Nu-Link2-Me 的 USB 和電腦的 USB 對接，來顯示範例程式碼的執行結果。

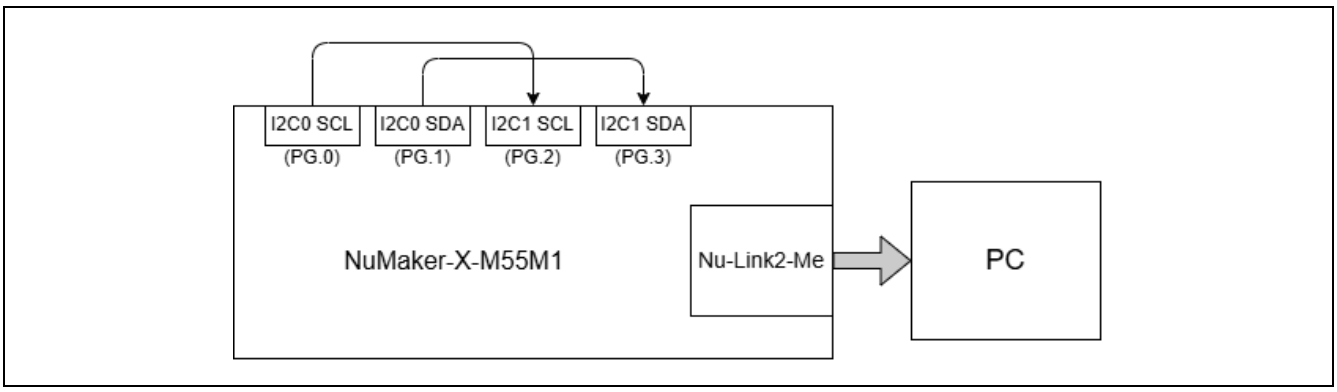

圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

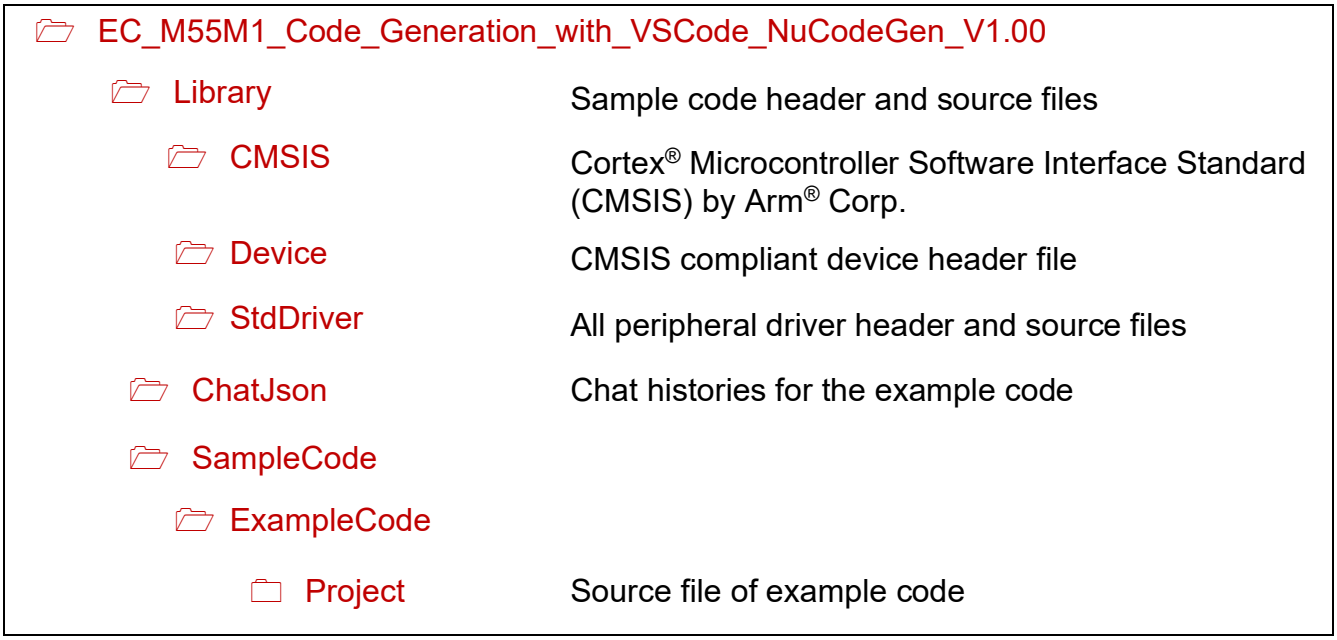

圖 4-1 目錄資訊

5. Chat json 匯入

1. 在 VS Code 中，按下 **Ctrl+Shift+P**。

2. 搜尋 Chat: Import Chat，如圖 5-1 所示，選擇要匯入的 json 檔。

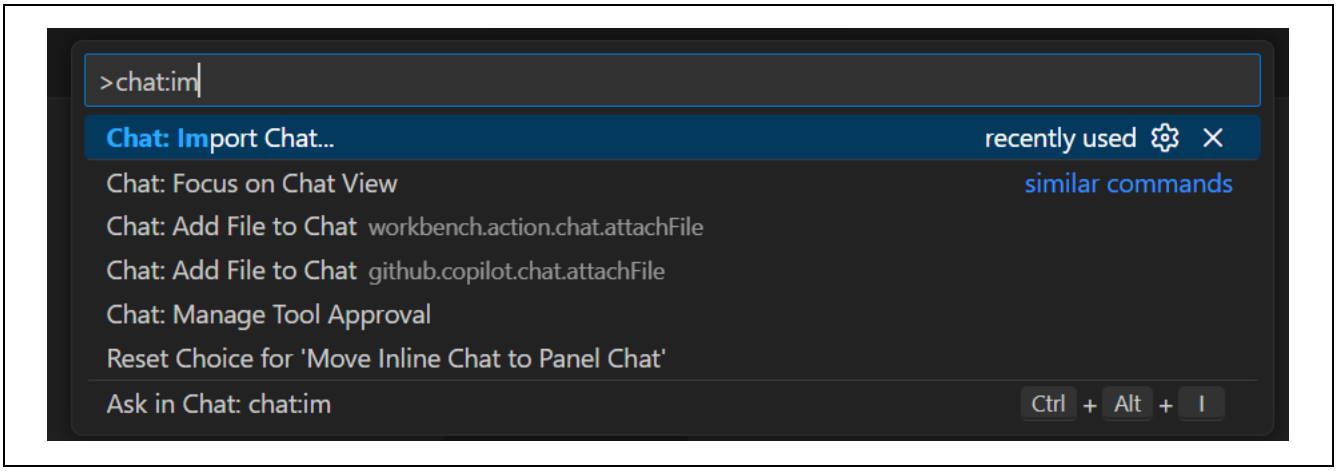

圖 5-1 匯入 json 檔

6. 範例程式執行

1. 根據目錄資訊章節，在 VS Code 中開啟 *ExampleCode* 路徑中的 *VSCode* 資料夾。
2. 點擊 VS Code 視窗側邊欄的 CMSIS 圖示，顯示 CMSIS view。
   - 編譯
   - 下載程式碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行程式碼

7. 修訂紀錄

Date	Revision	Description
2026.03.04	1.00	初始發布。

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.