M030G 模擬MDIO 波型與解析封包

NuMicro® 32位系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例代碼使用 M030G I²C 介面接收操作命令並使用 SPI 介面模擬 MDIO 波形用以控制乙太網路 PHY(IP101GR)。
BSP 版本	M029G_M030G_M031G_Series_BSP_CMSIS_V3.03.000
開發平台	NuMaker-M030GTD V1.1

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

本範例代碼使用 M030G I²C 介面來接收操作指令，並使用 SPI 介面來模擬 MDIO 波形用以控制乙太網路 PHY。

大多數 NuMicro® 家族微控制器不支援 MDIO 介面，本範例代碼提供一種存取乙太網路 PHY 的解決方案，可廣泛應用於 NuMicro® 家族微控制器。

本範例代碼需另接外部硬體模組，包含一 I²C 主機與乙太網路 PHY。I²C 主機發送讀取/寫入指令來觸發 SPI 介面模擬 MDIO 讀取/寫入波形用以存取 IC+ 的乙太網路 PHY(IP101GR)。

1.1 原理

本範例代碼可以分為兩個部分，I²C 從機用來接收指令，MDIO 主機用來存取乙太網路 PHY。以下將重點介紹 I²C 封包格式與說明如何透過 SPI 介面來模擬 MDIO 波形。

1.1.1 I²C 封包格式

圖 1-1 與圖 1-2 分別表示 I²C 讀取/寫入指令的封包格式。封包格式由五個部分組成:

- 開始 (STA) 或者重複開始(Repeated STA) 信號
  - I²C 主機透過發送 STA 信號或 Repeated STA 信號來發起傳輸。STA 訊號表示新資料傳輸的開始。接著發送 I²C 從機位址和 R/W 位元。
- 從機位址和讀/寫位元傳輸
  - 在 STA 信號或 Repeated STA 信號後，I²C 主機發送 7 位元從機位址來識別通訊的目標裝置。I²C 從機確認接收後會傳送一 ACK 訊號。
- 乙太網路 PHY 的暫存器位址傳輸
  - 傳送 I²C 從機位址後，I²C 主機傳送 2 位元組乙太網路 PHY 的暫存器位址。當 I²C 從機向 I²C 主機傳送 ACK 訊號後，I²C 主機接著傳送資料。
- 資料傳輸

- 寫操作: I²C 主機傳送 2 位元組資料給 I²C 從機。
- 讀操作: I²C 主機從 I²C 從機接收 2 位元組資料。

- 停止信號(STO)
  - I²C 主機可以透過產生停止信號(STO) 信號來結束通訊。

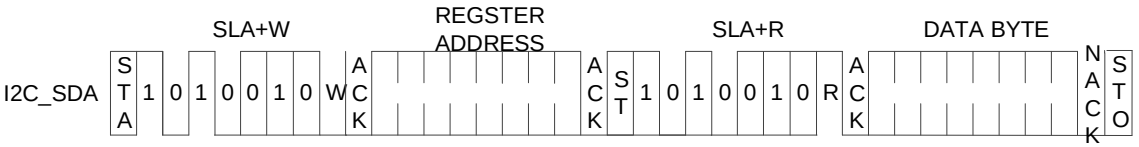

圖 1-1 I²C 讀取指令封包格式

圖 1-2 I²C 寫入指令封包格式

1.1.2 MDIO 資料格式

MDIO 資料格式如圖 1-3 所示，欄位說明如表 1-1 所示。

Preamble	ST	OP	PHY_ADR	REG_ADR	TA	DATA
32-bit	2-bit	2-bit	5-bit	5-bit	2-bit	16-bit

圖 1-3 MDIO 資料格式

欄位	資料寬度	說明
Preamble	32-bit	32 個連續 2'1
ST	2-bit	幀開始標誌 2'01
OP	2-bit	操作碼

		01 = 寫操作 10 = 讀操作
PHY_ADR	5-bit	PHY 位址
REG_ADR	5-bit	暫存器位址
TA	2-bit	狀態轉換時間
DATA	16-bit	MDIO 資料

表 1-1 MDIO 資料欄位介紹

由於 MDIO 是雙向資料線，故需將 SPI_MOSI 腳位、SPI_MISO 腳位和 MDIO 腳位連接在一起以實現此功能。要注意的是，在讀操作中，程式必須先將 SPI_MOSI 腳位配置為 GPIO 輸入模式，才能讀取資料。**Error! Reference source not found.** 圖 1-4 與圖 1-5 分別描述 SPI 介面模擬 MDIO 寫操作與讀操作的程式流程。

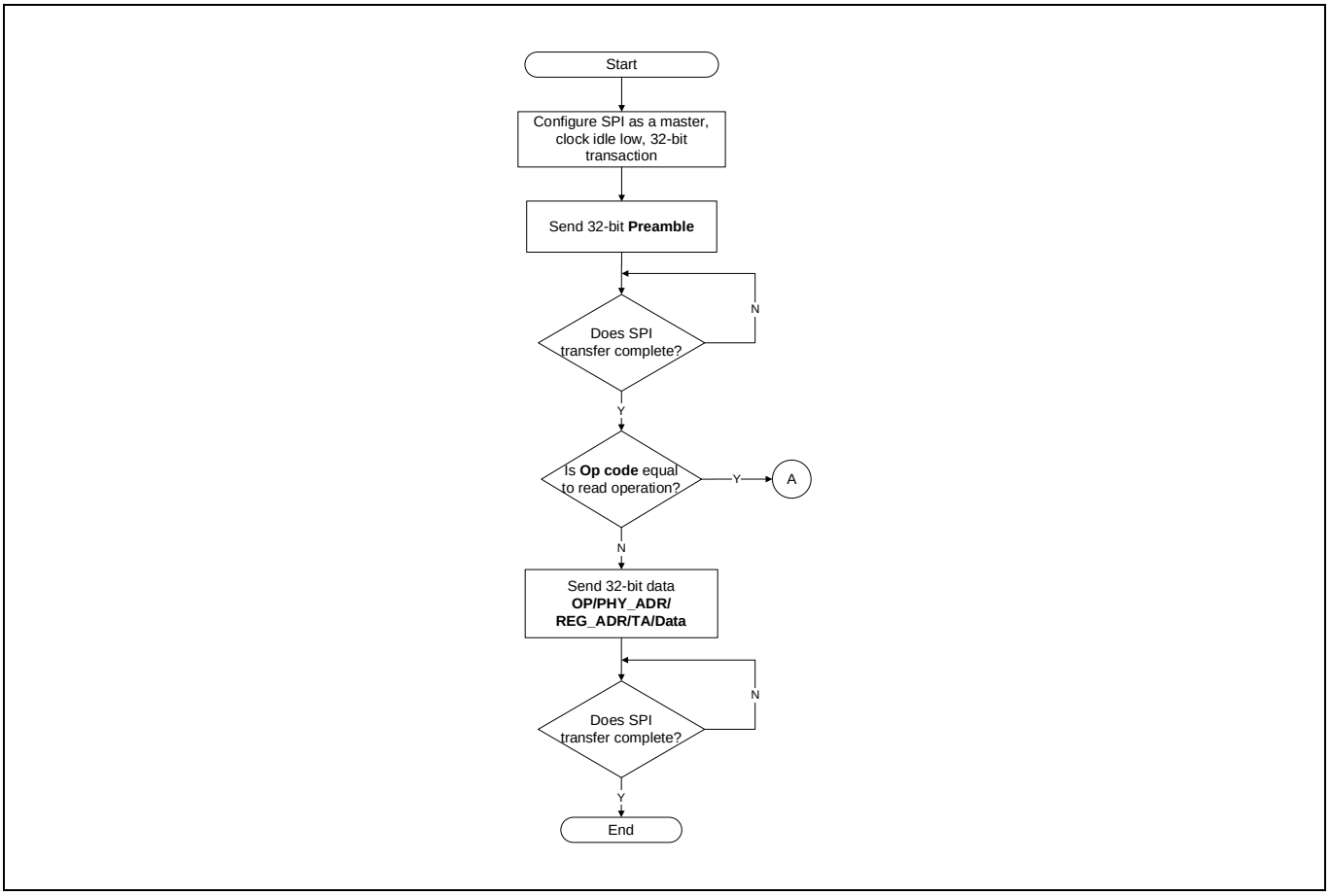

圖 1-4 SPI 模擬 MDIO 寫操作流程圖

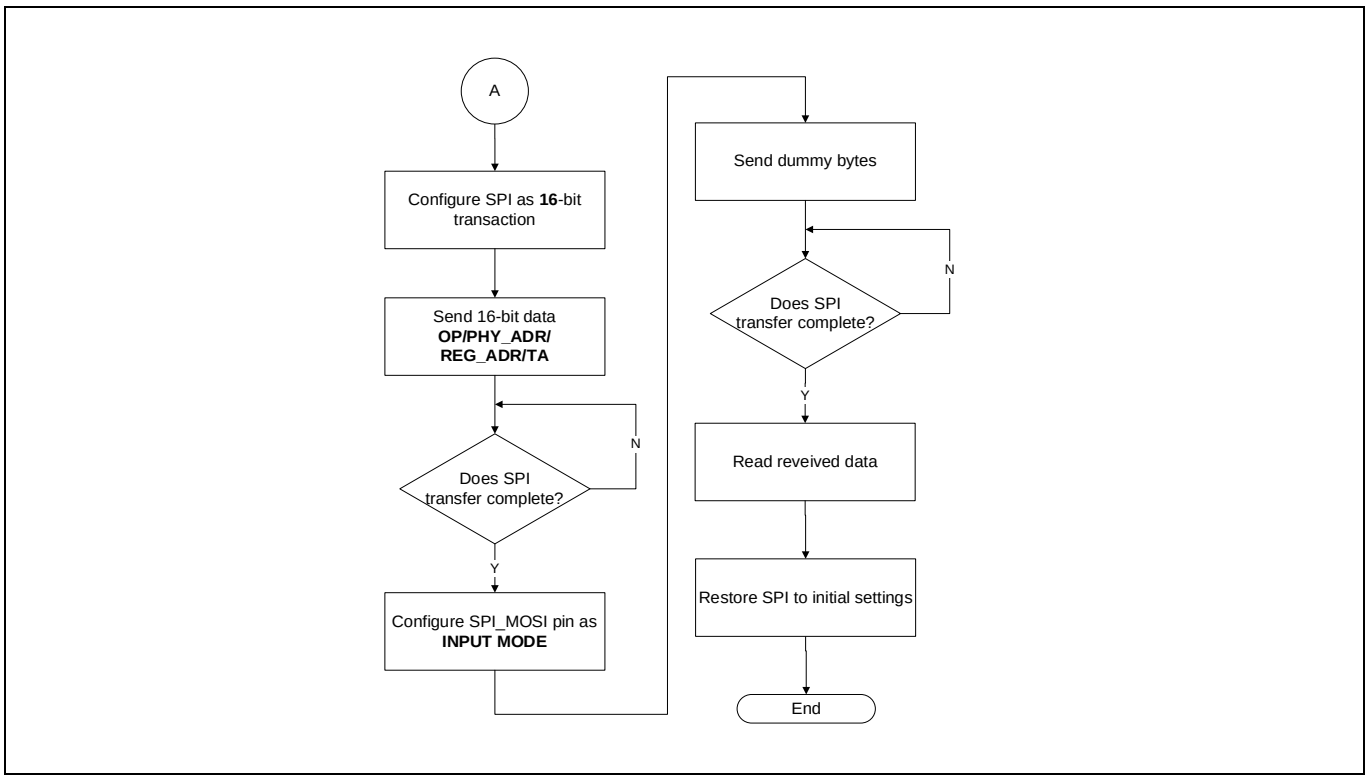

圖 1-5 SPI 模擬 MDIO 讀操作流程圖

1.2 執行結果

本範例代碼執行編譯結果如圖 1-6 與圖 1-7 所示，分別對應 MDIO 寫操作和讀操作波型。演示將暫存器 0x04 配置為 0x0001 並進行驗證。

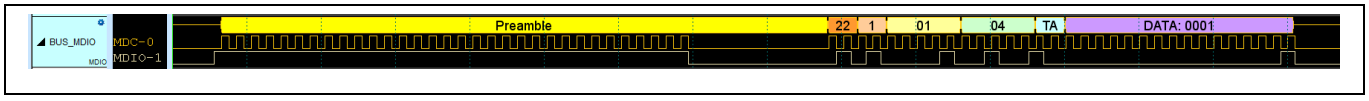

圖 1-6 MDIO 寫操作波型

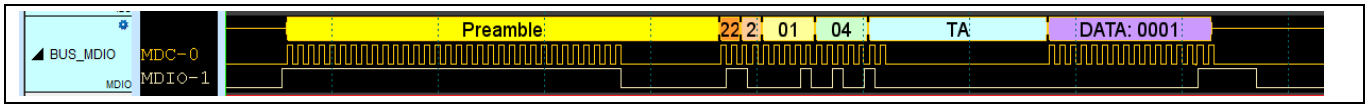

圖 1-7 MDIO 讀操作波型

2. 代碼介紹

初始化 SPI 介面，代碼位於 *spi_transfer.c*。

```
void SPI_Init(void)
{
    /* Set IP clock divider. SPI clock rate = 1MHz */
    SPI_Open(SPI0, SPI_MASTER, SPI_MODE_0, 32, SPI_CLK_FREQ);

    /* Enable the automatic hardware slave select function. Select the SS pin and
       configure as low-active. */
    SPI_EnableAutoSS(SPI0, SPI_SS, SPI_SS_ACTIVE_LOW);
}
```

初始化系統後，透過 SPI 介面模擬 MDIO 波型。代碼位於 *spi_transfer.c*。

寫操作

- 傳送 32 位元 Preamble。
- 傳送 MDIO_temp，包含 ST (2 位元)、opcode(2 位元)、PHY 位址(5 位元)、暫存器位址(5 位元)、TA (2 位元) 與 Data(16 位元) 。

讀操作

- 傳送 32 位元 Preamble。
- 設定 SPI 介面資料寬度為 16 位元與傳送 MDIO_temp，包含 ST (2 位元)、opcode(2 位元)、PHY 位址(5 位元)、暫存器位址(5 位元)、TA (2 位元) 與 Data(16 位元)。
- 設定 SPI_MOSI 腳位為 GPIO input 模式來接收資料。
- 接收完畢後，需重新設定 SPI 介面為初始設定。

```
MDIO_RES SPI_MDIO_RW(unsigned char rw, unsigned char phy_addr, unsigned char reg_addr,
unsigned int w_data, unsigned int *r_data)
{
    unsigned int MDIO_temp = 0;
    /* Checking parameters validation */
    if (rw > 0x02)
        return RW_ERROR;
    if (phy_addr > MDIO_PHY_MAX)
        return phy_ERROR;
    if (reg_addr > MDIO_REG_MAX)
        return reg_ERROR;
    if ((rw == MDIO_OP_W) && (w_data > W_DATA_MAX))
        return w_data_ERROR;
```

```
    if ((rw == MDIO_OP_R) && (w_data != W_DATA_MAX))
        return r_data_ERROR;

    /* Clear SPI FIFO */
    SPI_ClearTxFIFO(SPI0);
    SPI_ClearRxFIFO(SPI0);

    /* Prepare the MDIO package with 32-bit data, including 2-bit ST, 2-bit Op code,
     5-bit PHY address, 5-bit Register address, 2-bit TA and 16-bit Data */
    MDIO_temp = (MDIO_ST_CAUSE_22 << MDIO_ST_OFFSET) | (rw << MDIO_OP_OFFSET) |
                (phy_addr << MDIO_PHY_OFFSET) | (reg_addr << MDIO_REG_OFFSET) |
                (MDIO_TA << MDIO_TA_OFFSET) | w_data;

    /* READ operation with 16-bit data, including 2-bit ST, 2-bit Op code,
    5-bit PHY address and 5-bit Register address */
    if (rw == MDIO_OP_R)
        MDIO_temp = (MDIO_temp >> 16) & 0xFFFF;

    /* Send 32-bit preamble */
    SPI_WRITE_TX(SPI0, 0xffffffff);
    /* Check SPI0 busy status */
    while (SPI_IS_BUSY(SPI0));

    /* READ operation with 16-bit data, including 2-bit ST, 2-bit Op code,
    5-bit PHY address, 5-bit Register address and 2-bit TA*/
    /* WRITE operation with 32-bit data, including 2-bit ST, 2-bit Op code,
    5-bit PHY address, 5-bit Register address, 2-bit TA and 16-bit Data */
    if (rw == MDIO_OP_R)
        SPI_SET_DATA_WIDTH(SPI0, 16);

    SPI_WRITE_TX(SPI0, MDIO_temp);
    /* Check SPI0 busy status */
    while (SPI_IS_BUSY(SPI0));

    /* Read data from Ethernet PHY */
    if (rw == MDIO_OP_R){
        /* Configure PA0(MOSI) as input mode for receiving data */
        SYS->GPA_MFPL = (SYS->GPA_MFPL & ~(SYS_GPA_MFPL_PA0MFP_Msk)) |
                        (SYS_GPA_MFPL_PA0MFP_GPIO);
        GPIO_SetMode(PA, BIT0, GPIO_MODE_INPUT);

        SPI_ClearRxFIFO(SPI0);
        /* Send dummy data for receiving data */
        SPI_WRITE_TX(SPI0, 0xFFFF);
        /* Check SPI0 busy status */
        while (SPI_IS_BUSY(SPI0));
        /* Read received data */
        r_data[reg_addr] = SPI_READ_RX(SPI0);

        /* Restore to initial settings */
        SYS->GPA_MFPL = (SYS->GPA_MFPL & ~(SYS_GPA_MFPL_PA0MFP_Msk)) |
                        (SYS_GPA_MFPL_PA0MFP_SPI0_MOSI);
        SPI_SET_DATA_WIDTH(SPI0, 32) ;
    }
    return PASS;
}
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M029G_M030G_M031G_Series_BSP_CMSIS_V3.03.000
- IDE 版本
  - Keil uVersion 5.38

3.2 硬體需求

- 電路元件
  - NuMaker-M030GTD V1.1
- 線路示意圖
  - 將 I2C_SCL (PF.3) pin 腳與 I2C_SDA (PF.2) pin 腳分別連接到 I²C 主機，用以接收操作指令。
  - 將 SPI_MOSI (PA.0) pin 腳與 SPI_MISO (PA.1) pin 腳連接到乙太網路 PHY 的 MDIO pin 腳; SPI_CLK (PA.2) pin 腳連接到乙太網路 PHY 的 MDC pin 腳用存取乙太網路 PHY 暫存器。
  - 分別連接 4.7 k 歐姆的上拉電阻於 I2C_SCL (PF.3) pin 腳、I2C_SDA (PF.2) pin 腳與 MDIO pin 腳。

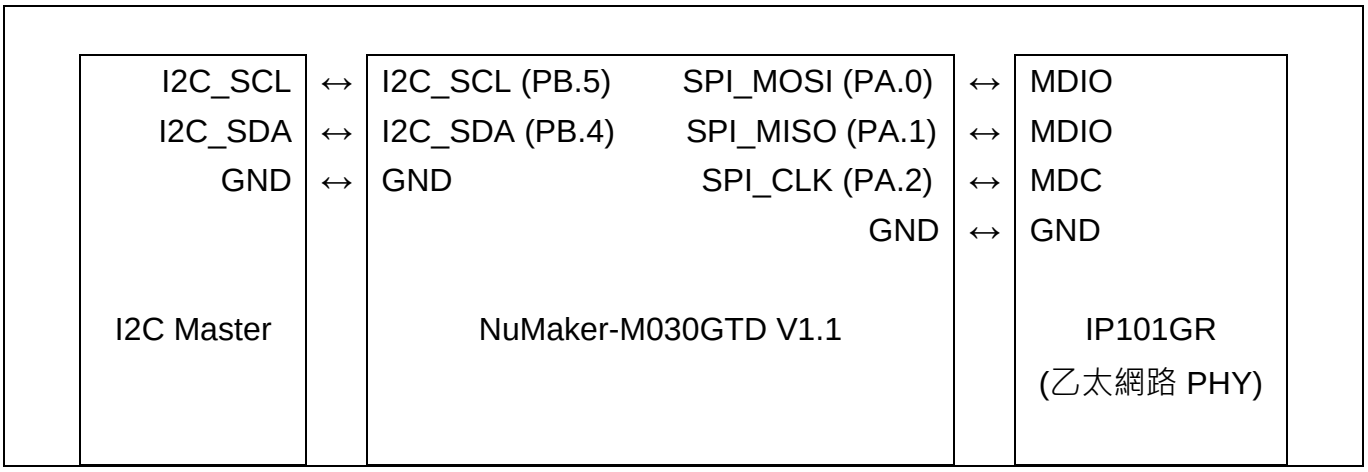

圖 3-1 腳位連接圖

4. 目錄資訊

EC_M030G_I2C_Parse_CMD_SPI_Emulate_MDIO _V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	
Project	Source file of example code

圖 4-1 目錄資訊

5. 範例程式執行

1. 根據目錄資訊章節進入 ExampleCode 路徑中的 KEIL 資料夾，雙擊 M030G_I2C_Parse_CMD_SPI_Emulate_MDIO.uvprojx。
2. 進入編譯模式介面
   - 編譯
   - 下載代碼至記憶體
   - 進入 / 離開除錯模式
3. 進入除錯模式介面
   - 執行代碼

6. 修訂紀錄

Date	Revision	Description
2024.05.16	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.