M030G I²C Slave 400 kbps

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	本範例程式碼使用 M030G 實現 I²C Slave 模式支援 400 kbps
BSP 版本	M029G_M030G_M031G_Series_BSP_CMSIS_V3.04.000
開發平台	NuMaker-M030GTD V1.1 NuMaker-M031GTD V1.0

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

本範例程式碼使用 M030G 實現 I²C Slave 模式下支援 400 kbps 速率。

I²C 匯流排只需要兩個訊號線分別是 SDA（串列資料線）和 SCL（串列時鐘線），都是雙向 I/O 線。介面電路為開漏輸出，需通過上拉電阻接到電源 VCC。協定規定標準模式下可達 100 kbps，快速模式下可達 400 kbps，高速模式下可達 3.4 Mbps。本參考程式碼介紹基於 M030G 如何實現 I²C Slave 支持 400 kbps。

1.1 原理

M030G 支援 2 個 I²C 裝置，都可以設置為 Master 或 Slave。各自有一個獨立的中斷向量和中斷狀態暫存器。

當 I²C 裝置設置為 Slave，如果 I²C 匯流排有資料在傳輸，I²C 裝置一旦檢測匯流排上狀態發生變化，就會產生 I²C 中斷。固件需要根據 I²C 協定，對不同的狀態進行相應的處理。在 MCU 主頻固定的情況，I²C 中斷對各狀態處理的更快，I²C 速率也就會更高。

當產生 I²C 中斷的時候，I²C 中斷標識 SI 將置 1。在將 SI 清零之前，匯流排由 I²C 裝置控制，中斷狀態暫存器一直保持不變。SI 清零後，I²C 裝置立即釋放匯流排，開始後續的傳輸。因此可以在全部獲取當前有效狀態後，儘早將 SI 清零，以提高傳輸速率。

針對某些應用例如光模組，I²C 從機有多個設備地址，而且資料結構較複雜。當從機處於發送資料狀態，如果此時需要通過查找資料表找到待發送資料，這樣會比較耗時，進而影響 I²C 通信速率。如果可以在其他狀態下提前準備待發送資料，當從機處於發送資料狀態，直接將提前準備好的資料發送出去，也可以提高傳輸速率。

1.2 執行結果

本範例程式碼正確編譯後刻錄到評估板 M030G 作從機；為了主機能輸出更高速率時鐘信號，將 *I2C_Master_400_kbps.bin* 刻錄到另一塊評估板 M031G 實現主機。通過設置主、從機板指撥開關啟用其各自虛擬 COM，然後使用 2 條 USB 線將主、從機連接到同一 PC。在 PC 上打開 2 個對應的 COM 調試助手，然後根據主機 COM 提示，輸入資料進行測試，如圖 1-1 所示。

```
CPU @ 72000000Hz

Enter any a data to start write. OxA0[0x00~0x03] = 0x00~0x03.
Write OxA0[0x00~0x03] finished.

Enter any a data to start current read. OxA0[0x04~0x07] = 0x04~0x07.
->A0[0x4] is 0x4
->A0[0x5] is 0x5
->A0[0x6] is 0x6
->A0[0x7] is 0x7
Current read A0[0x04~0x07] successful.

Enter any a data to start random read. OxA0[0x20~0x23] = 0x20~0x23.
->A0[0x20] is 0x20
->A0[0x21] is 0x21
->A0[0x22] is 0x22
->A0[0x23] is 0x23
Random read A0[0x20~0x23] successful.
```

圖 1-1 主機 COM 調試信息

I²C Master 向從機設備地址為 0xA0 暫存器起始地址為 0x00，連續寫 0x00、0x01、0x02、0x03，波形如圖 1-2 所示。

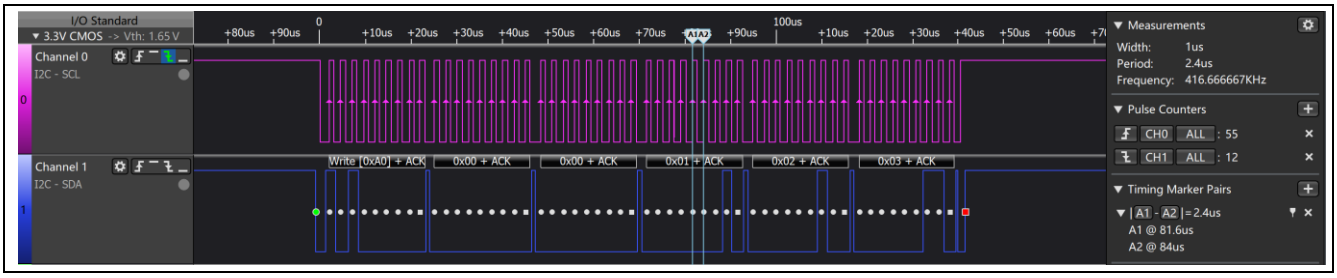

圖 1-2 連續寫

之後 I²C Master 向從機設備地址為 0xA0，執行當前連續讀 4 個位元組，所讀數據為 0x04、0x05、0x06、0x07，波形如圖 1-3 所示。

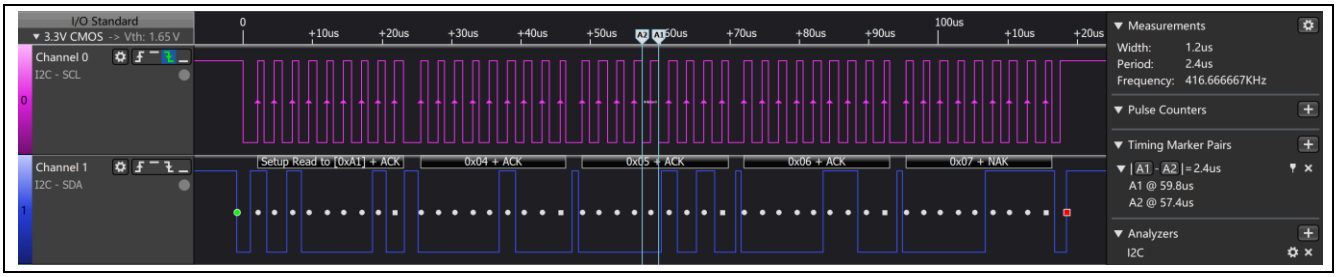

圖 1-3 當前地址連續讀

之後 I²C Master 向從機設備地址為 0xA0 暫存器起始地址為 0x20，連續讀 4 個位元組，所讀數據為 0x20，0x21，0x22，0x23，波形如圖 1-4 所示。

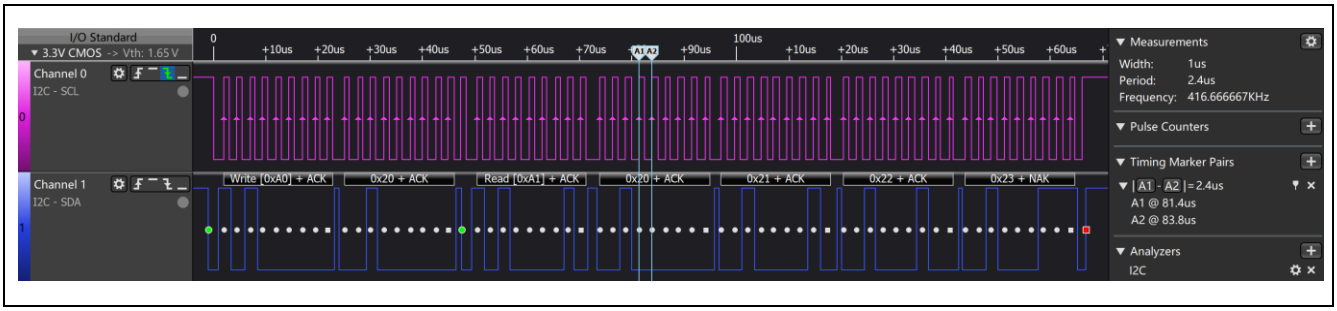

圖 1-4 隨機地址連續讀

2. 程式碼介紹

從機設備地址、Table 資料長度及 I²C 各中斷狀態巨集定義，程式碼位於 main()函式。

```
#define I2C_DEV_ADDR                        0xA0
#define TABALE_SIZE                         0x100
#define SLAVE_TRANSMIT_DATA_ACK             0xB8
#define SLAVE_TRANSMIT_ADDRESS_ACK          0xA8
#define SLAVE_TRANSMIT_REPEATSTART_OR_STOP  0xA0
#define SLAVE_RECEIVE_ADDRESS_ACK           0x60
#define SLAVE_RECEIVE_DATA_ACK              0x80
#define SLAVE_TRANSMIT_DATA_NACK            0xC0
#define SLAVE_RECEIVE_DATA_NACK             0x88
```

初始化各 Table 資料，程式碼位於 main()函式。

```
/* Initialize every table */
for (i = 0; i < TABALE_SIZE; i++)
{
    g_u8Tab_A0[i] = i;
    g_u8Tab_A2[i] = i + 1;
}
```

I²C 初始化函式，設置從機設備地址分別為 0xA0、0xA2，程式碼位於 I2C0_Init ()函式。

```
/* Open I2C module and set bus clock */
I2C_Open(I2C0, 100000);

/* Set I2C 4 Slave Addresses */
I2C_SetSlaveAddr(I2C0, 0, (I2C_DEV_ADDR >> 1), 0); /* Slave Address : 0xA0        */
I2C_SetSlaveAddr(I2C0, 1, 0x00, 0);                /* Slave Address : not be used */
I2C_SetSlaveAddr(I2C0, 2, 0x00, 0);                /* Slave Address : not be used */
I2C_SetSlaveAddr(I2C0, 3, 0x00, 0);                /* Slave Address : not be used */
I2C_SetSlaveAddrMask(I2C0, 0, 0x01);               /* 1:Mask Enabled.
                           Set bit0 for supporting two slave addresses(0xA0,0xA2) */

/* Enable I2C interrupt */
I2C_EnableInt(I2C0);
NVIC_EnableIRQ(I2C0_IRQn);
```

進入 I²C 中斷後，先讀取狀態暫存器及數據暫存器資料，然後清 SI 旗標。程式碼位於 I2C0_IRQHandler()函式。

```
    /* Read status register of I2C0 */
    u32Sta = (uint8_t)I2C_GET_STATUS(I2C0);

    /* Read data register of I2C0 */
    u8Dat0 = I2C_GET_DATA(I2C0);

    /* Clear SI,Set AA */
    I2C_SET_CONTROL_REG(I2C0, I2C_CTL_SI_AA);
```

產生中斷後如果狀態暫存器為 0xA0 即接收到 Restart 信號，此時準備待發送資料，並存儲於 u8PreDat_A0。程式碼位於 I2C0_IRQHandler()函式。

```
    else if (u32Sta == SLAVE_TRANSMIT_REPEATSTART_OR_STOP)  /* A STOP or repeated START
                                has been received while still addressed as Slave/Receiver*/
    {
        blData = FALSE;

        /* Prepare the first data to be sent later.*/
        if (blDevAddr_A0)
        {
            u8PreDat_A0 = g_u8Tab_A0[Data_Addr_A0++];
        }
        else
        {
            u8PreDat_A2 = g_u8Tab_A2[Data_Addr_A2++];
        }
    }
```

產生中斷後如果狀態暫存器為 0xA8 即接收到 SLA+R 信號，此時直接發送前面已準備資料 u8PreDat_A0，然後再準備下一個待發送資料，並存儲於 u8PreDat_A0。程式碼位於 I2C0_IRQHandler()函式。

```
        else if (u32Sta == SLAVE_TRANSMIT_ADDRESS_ACK)    /* Own SLA+R has been receive;
ACK has been return */
        {
            if (--u8Dat0 == I2C_DEV_ADDR)
            {
                blDevAddr_A0 = TRUE;
                I2C_SET_DATA(I2C0, u8PreDat_A0);          /* Send first data */
                u8PreDat_A0 = g_u8Tab_A0[Data_Addr_A0++]; /* Prepare second data */
            }
            else
            {
                blDevAddr_A0 = FALSE;
                I2C_SET_DATA(I2C0, u8PreDat_A2);
                u8PreDat_A2 = g_u8Tab_A2[Data_Addr_A2++];
            }
        }
```

產生中斷後如果狀態暫存器為 0xB8 即 I2CDAT 中資料已發送完成，此時直接發送前面已準備資料 u8PreDat_A0，然後再準備下一個待發送資料，並存儲於 u8PreDat_A0。程式碼位於 I2C0_IRQHandler()函式。

```
        if (u32Sta == SLAVE_TRANSMIT_DATA_ACK) /* Data byte in I2CDAT has been transmitted
ACK has been received */
        {
            if (blDevAddr_A0)
            {
                I2C_SET_DATA(I2C0, u8PreDat_A0);/* Send second data */
                u8PreDat_A0 = g_u8Tab_A0[Data_Addr_A0++];
```

```
            }
            else
            {
                I2C_SET_DATA(I2C0, u8PreDat_A2);
                u8PreDat_A2 = g_u8Tab_A2[Data_Addr_A2++];
            }
        }
```

3. 軟體與硬體需求

3.1 軟體需求

- BSP 版本
  - M029G_M030G_M031G_Series_BSP_CMSIS_V3.04.000
- IDE 版本
  - Keil uVision 5.36

3.2 硬體需求

- 電路元件
  - NuMaker-M030GTD V1.1
  - NuMaker-M031GTD V1.0
- 線路示意圖
  - 將從機 I2C0 SCL (PF.3) 、I2C SDA(PF.2) 、GND 引腳分別與主機 SCL (PB.5)、SDA (PB.4)、GND 相連。注意在從機上分別使用跳線將 PF.3 與 JP7 短接，實現 SCL 引腳上拉；將 PF.2 與 JP8 短接，實現 SDA 引腳上拉。

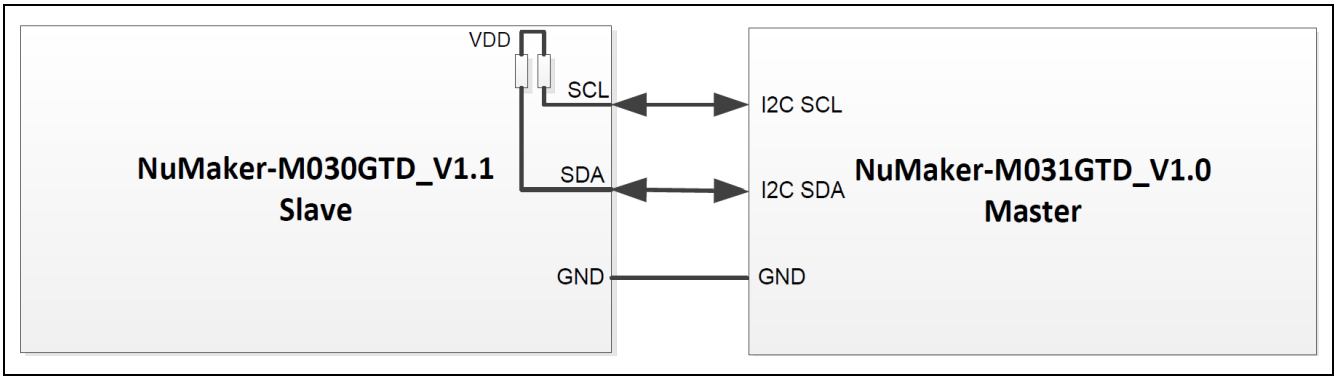

圖 3-1 線路示意圖

4. 目錄資訊

目錄結構如下圖所示。

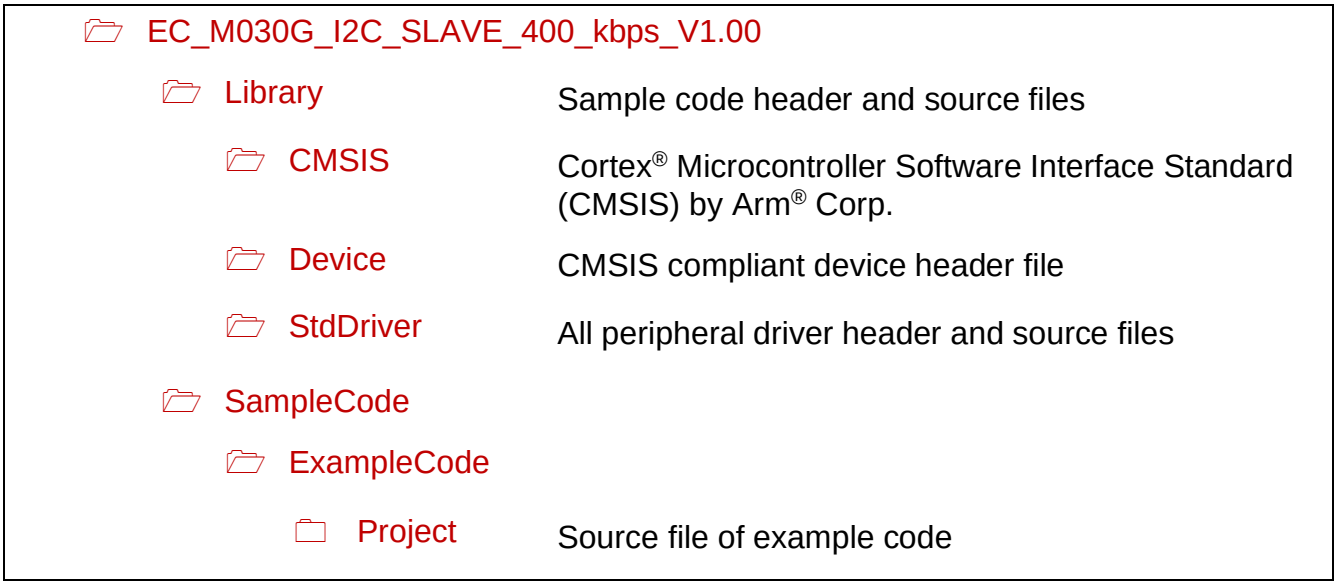

圖 4-1 目錄資訊

5. 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_SLAVE_400_kbps.uvproj*。
2. 進入編譯模式介面
   - 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

6. 修訂紀錄

Date	Revision	Description
2025.03.31	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.