CAN Bus Off 恢復機制

NuMicro® 32位系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例程式為 CAN Bus 之應用，當節點發生 Bus off 情形時如何進行恢復，讓節點重新進入 Bus 上進行封包的傳送跟接收。
BSP 版本	M480 Series BSP CMSIS V3.05.003
開發平台	NuMaker-PFM-M487 V3.0

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

本範例為 CAN Bus 相關應用，為了避免某個設備因為自身原因（例如硬體損壞）或外部干擾，導致無法正確收發資料，從而影響總線上其它正常的節點，提出一恢復節點設備的參考設計。CAN Bus 具備硬體錯誤診斷功能，CAN 通用規範中規定每個 CAN 控制器中有一個發送錯誤計數器，和一個接收錯誤計數器，根據計數值不同，節點會處於不同的錯誤狀態，並根據計數值的變化進行狀態轉換，如下圖所示:

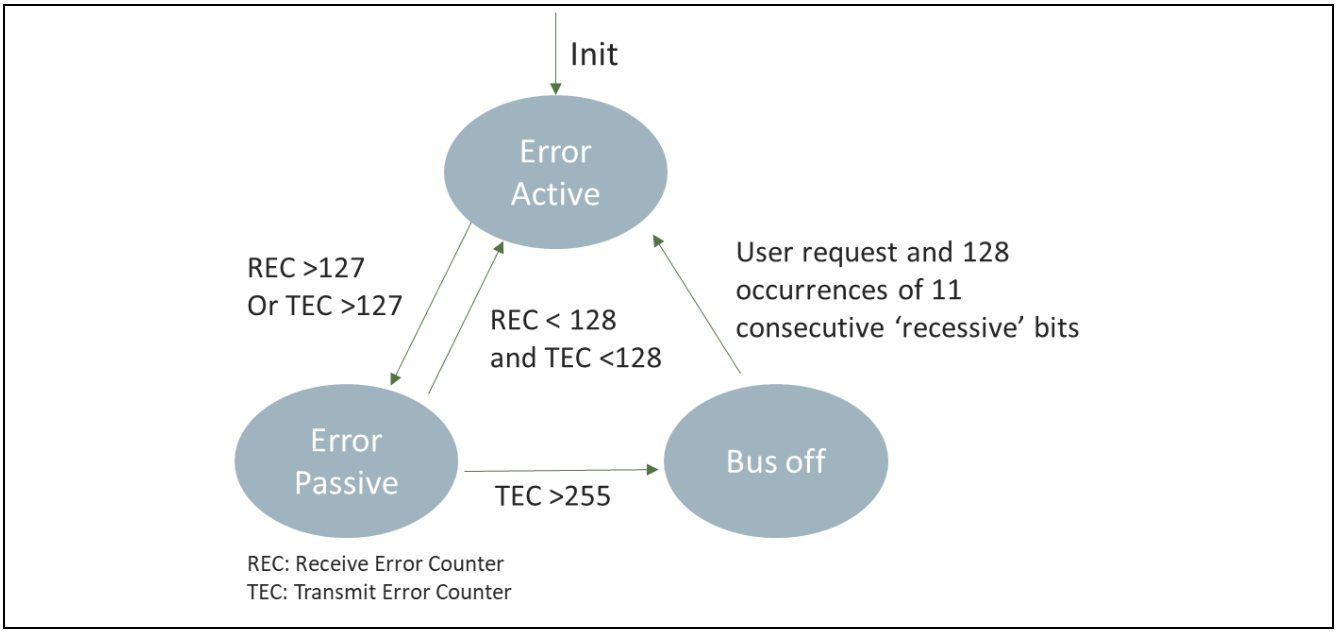

圖 1-1 CAN 錯誤及 bus off 恢復機制

通常在 Error Active 以及 Error Passive 兩種狀態下的節點，MCU 不需要額外的軟體去介入處理，節點會自動偵測錯誤的計數量並恢復至前一狀態其中。若錯誤計數大於一定數值，節點進入 bus off，在該狀態下代表節點已無法進行自動恢復，並且不能傳送或接收總線上的數據。根據 CAN Specification Rev. 2.0，節點進入到 bus off 的狀態後，當 CAN controller 檢測到 128 次 11 個連續的隱性位後即可恢復通信，若以 baud rate 設定 500K 的 CAN Bus 來看，時間僅僅為幾毫秒，通常該節點不太可能已經排除了造成 bus off 的實際原因，因此，貿然將該節點重新初始化恢復通訊，可能只是重新進到 bus off 的狀態。基於上述原因，本範例提供一 bus off 的處理方式，當軟體檢測到節點進到 bus off 狀態時不會立即恢復通訊，而是等到確認造成錯誤的狀態排除後，再進行 bus off recovery 的步驟，讓節點重新進入總線並進行資料的接收及發送。

1.1 軟體架構

軟體流程如圖 1-2 所示， 使用三片帶有 M487 Advance 上板的 NuMaker-PFM-M487 ，將板上 CAN bus 介面接上後，按下 reset 鍵範例程式便開始運行。一開始會先初始化 CAN controller、設定 baud rate、打開中斷以及設定 rx buffer。之後每次透過 tx buffer 傳送 CAN 封包前，都會先確認 bus off 的旗標是否舉起(CAN controller 偵測到後會發出中斷)。當軟體檢測到 bus off 旗標舉起後，系統便無法再接收 bus 上的訊息，軟體也會停止發送封包到 bus 上，並等待恢復節點的訊號產生。

Bus off 恢復機制不能通過設置或重置 **Init** bit 來縮短。如果設備進入 bus off，M487 將自行設置 **Init** bit，停止所有總線活動。一旦 **Init** 被 CPU 清除，設備將等待 129 次的 idle 訊號（129 * 11 個連續的隱性位）出現，然後恢復正常操作。在總線關閉恢復機制結束時，錯誤管理計數器將被重置。在重置 **Init** bit 後的等待期間，每當系統偵測到 11 個隱性位序列時，都會將 **Bit0Error** 設為 1 寫入狀態寄存器，使系統能夠及時檢查 CAN bus 是否沒有停留在顯性狀態，或者持續受到干擾。而在範例中是透過按下 M487 Advance 上板的 SW1 按鍵，來通知軟體進行節點恢復程序，因此，當確認造成 bus off 的故障已排除後，便可按下 SW1 交由軟體完成操作。

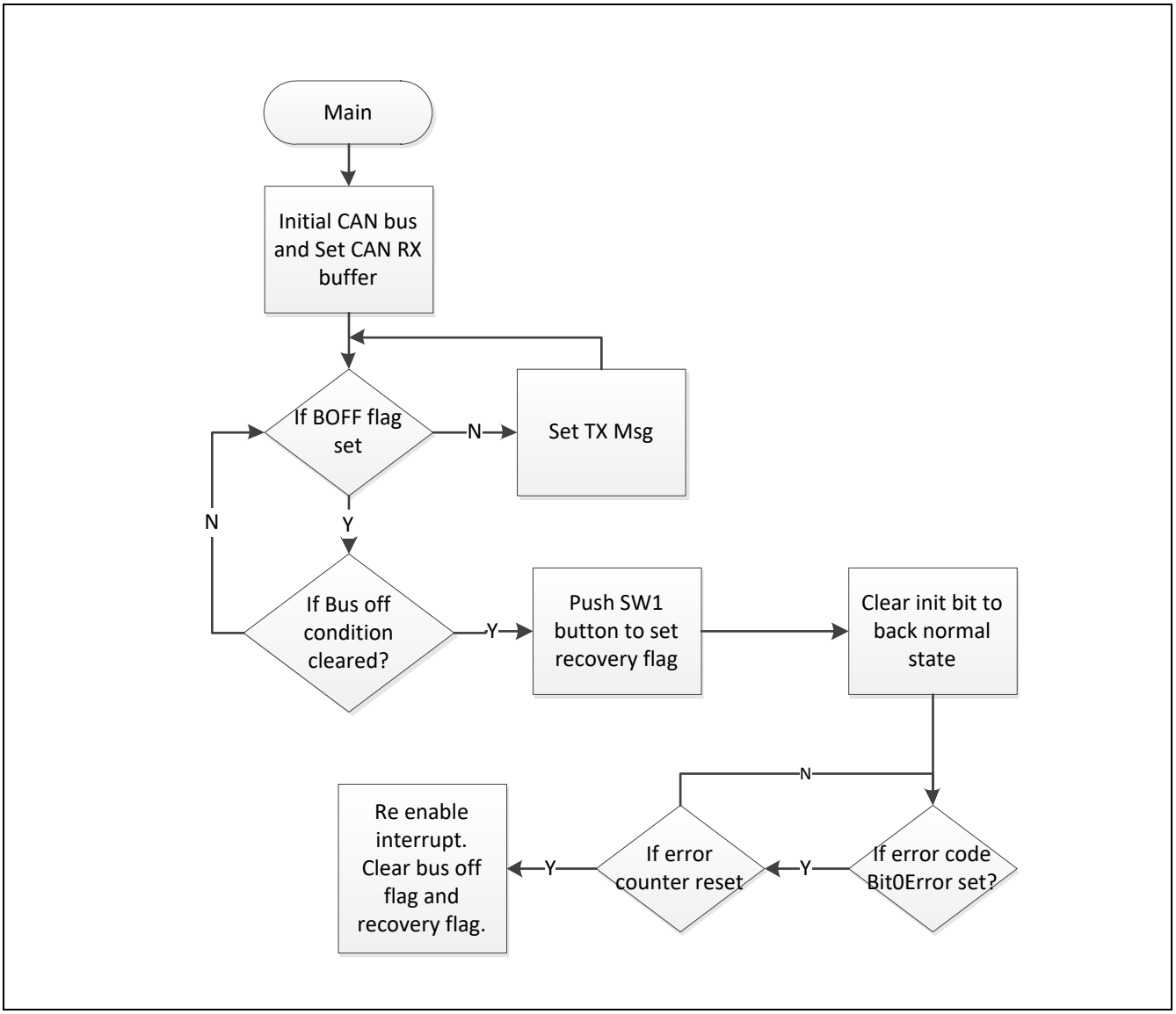
Main
Initial CAN bus and Set CAN RX buffer
If BOFF flag set
N
Set TX Msg
Y
N
If Bus off condition cleared?
Y
Push SW1 button to set recovery flag
Clear init bit to back normal state
N
If error code Bit0Error set?
Y
If error counter reset
Y
Re enable interrupt. Clear bus off flag and recovery flag.

圖1-2 軟體流程圖

1.2 功能介紹

1. 將三片 M487 Advance 上板的 CAN 介面連接至麵包板上後，封包的傳輸情形如圖 1-3。

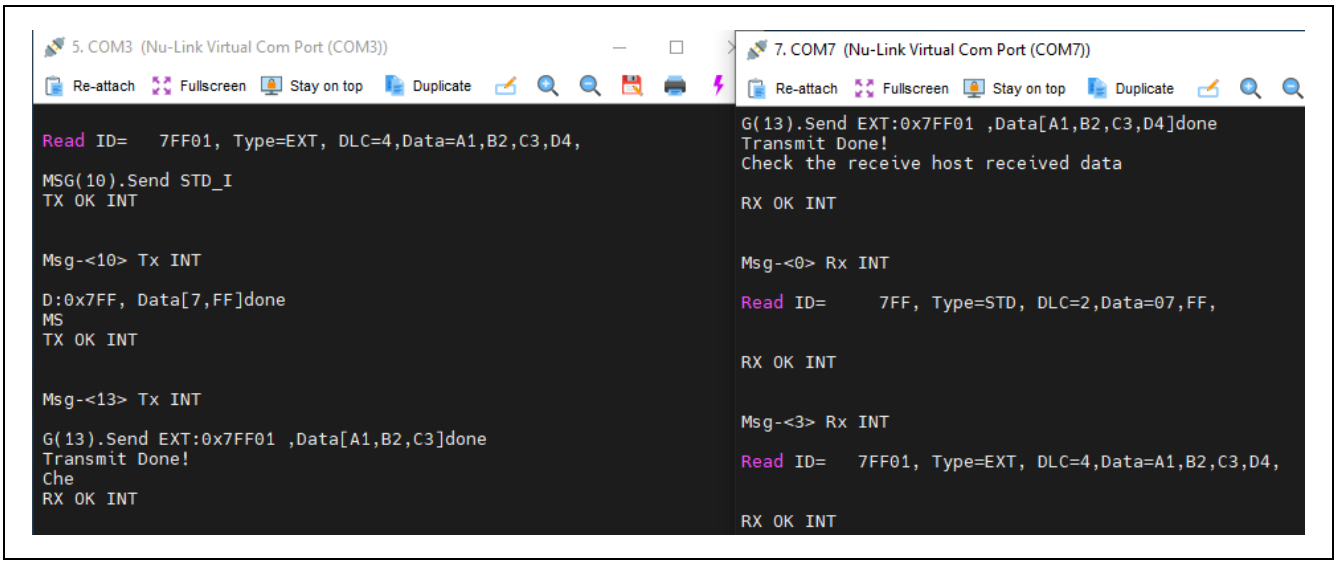

圖 1-3 CAN Bus 正常收送封包

2. 透過短路 CAN transceiver 上的 CAN_H/CAN_L 產生 bus off，如圖 1-4，移除短路後按下 SW1 即可恢復到圖 1-3。

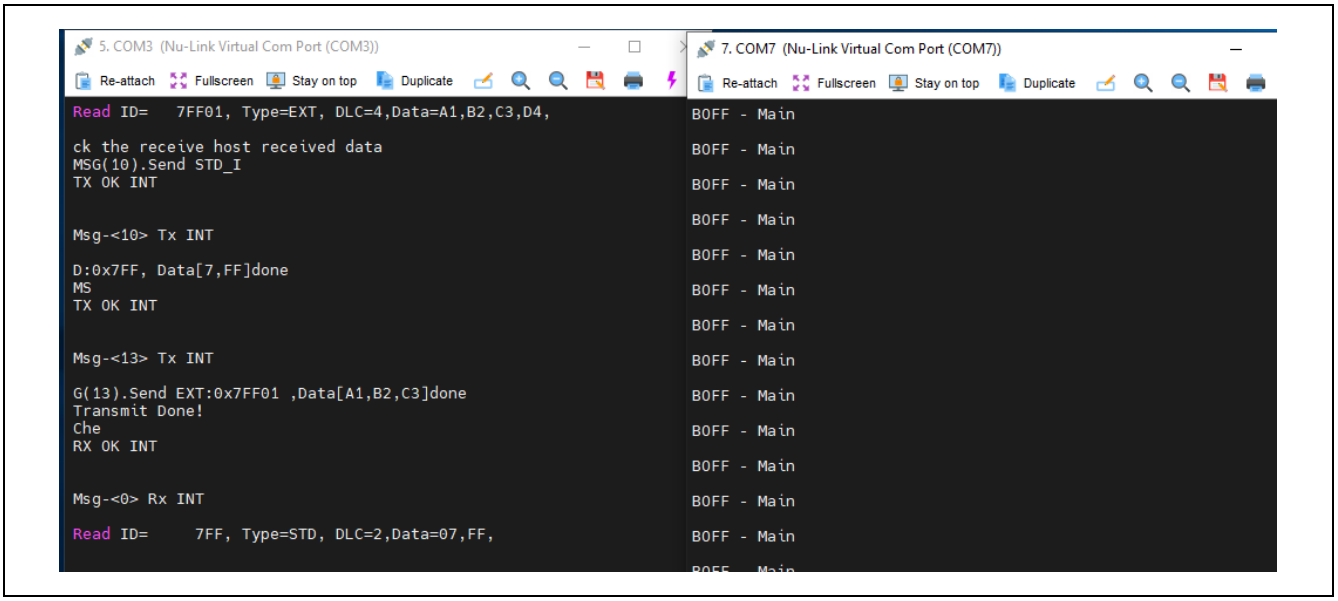

圖 1-4 CAN bus off 狀態

2. 程式介紹

以下程式實現了章節 1.1 的架構。

```
int main()
{

    SYS_Init();
    UART0_Init();

    /* Select CAN Multi-Function */
    CAN_Init(CAN0);
    Note_Configure();
    SelectCANSpeed(CAN0);

    printf("\n");
    printf("+------------------------------------------------------------------- +\n");
    printf("|  Nuvoton CAN BUS-off Recovery DEMO                                 |\n");
    printf("+--------------------------------------------------------------------+\n");
    printf("|  Receive/Transmit a message by normal mode (2)                     |\n");
    printf("+--------------------------------------------------------------------+\n");
    printf("|  When Bus off happened, push abvanced board SW1 button to notify   |\n");
    printf("|  software start bus recovery procedure                             |\n");
    printf("+--------------------------------------------------------------------+\n");

    printf("Press any key to continue ...\n\n");
    getchar();
    printf("Wait Msg\n");


    Test_NormalMode_Rx(CAN0);

    /* INT Mode for Tx & Rx*/
    CAN_EnableInt(CAN0, CAN_CON_IE_Msk|CAN_CON_SIE_Msk|CAN_CON_EIE_Msk);
    NVIC_SetPriority(CAN0_IRQn, (1<<__NVIC_PRIO_BITS) - 2);
    NVIC_EnableIRQ(CAN0_IRQn);


    while(1) {
        uint32_t tx_ount;
        for(tx_ount=1; tx_ount<20000000 ; tx_ount++) {}
```

```
        if(canbus_INT_flag != CAN_BOFF_INT) {
            Test_NormalMode_Tx(CAN0);
        }

        if(canbus_INT_flag == CAN_BOFF_INT) {
            printf("\nBOFF - Main\n");

            /*   bus-off recovery check*/
            if(PA0 == 0) {
                BUSrecovery = 1;
            }
            if(BUSrecovery != 0) {
                CAN0->CON &= (~(CAN_CON_INIT_Msk | CAN_CON_CCE_Msk));
                while(CAN0->CON & CAN_CON_INIT_Msk);
                while((CAN0->STATUS & CAN_STATUS_LEC_Msk) == 0x05) {
                    if(CAN0->ERR ==0) {
                        CAN_EnableInt(CAN0,
CAN_CON_IE_Msk|CAN_CON_SIE_Msk|CAN_CON_EIE_Msk);
                        canbus_INT_flag = 0x00;
                        BUSrecovery = 0;
                        break;
                    }
                }
            }
        }
    }

}
```

3. 軟硬體需求

3.1 軟體需求

- BSP 版本
  - M480 Series BSP CMSIS V3.05.003
- IDE 版本
  - Keil uVersion 5.37

3.2 硬體需求

- NuMaker-PFM-M487 v3.0
- M487 Advance board Ver. 4.0
- 杜邦線, 麵包版
- 腳位連接示意圖
  - 將 CAN_H (PA.5) pin 以及 CAN_L (PA.4) pin 腳連接到 bus (麵包版)上。

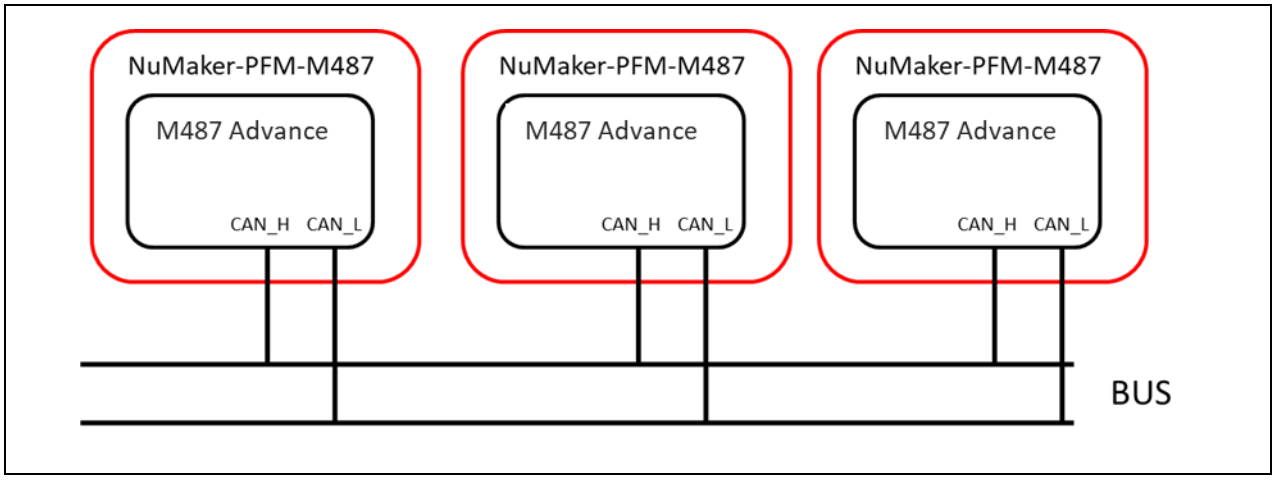

圖 3-1 CAN bus腳位連接示意圖

4. 目錄資訊

目錄架構如下所示。

EC_M480_CAN_Busoff_Recovery_Mechanism_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	Source file of example code

圖 4-1 目錄資訊

5. 範例程式執行

1. 打開 Keil Project
2. 編譯範例程式後燒入至 NuMaker-PFM-M487。
3. 完成三片版子燒入後，將 M487 Advance 上板的 CAN bus 介面接至麵包板。
4. 按下 reset 後開始進行 CAN bus 封包對傳。
5. 將其中一片板子的 CAN_H/CAN_L 從麵包板上移除，並透過短路該 CAN_H/CAN_L 製造 bus off。
6. 移除短路，重新將該 NuMaker-PFM-M487 的 CAN_H/CAN_L 接到麵包板上，此時另外兩片仍繼續正常收送封包。
7. 按下 M487 Advance 上板 SW1，對重新連接到 bus 上的 NuMaker-PFM-M487 進行節點恢復。
8. 該節點重新完成初始化，開始進行封包收送。

6. 修訂紀錄

Date	Revision	Description
2022.11.28	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.