CAN Bus Off 恢复机制

NuMicro® 32位系列微控制器范例代码介绍

文件信息

应用简述	本范例程序为 CAN Bus 之应用，当该节点发生 Bus off 情形时如何进行恢复，让装置重新进入 Bus 上进行数据包的传送跟接收。
BSP 版本	M480 Series BSP CMSIS V3.05.003
开发平台	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

本范例为 CAN Bus 相关应用，为了避免某个设备因为自身原因（例如硬件损坏）或外部干扰，导致无法正确收发数据，从而影响总线其它正常的节点，提出一恢复节点设备的参考设计。CAN Bus 具备硬件错误诊断功能，CAN 通用规范中规定每个 CAN 控制器中有一个发送错误计数器，和一个接收错误计数器，根据计数值不同，节点会处于不同的错误状态，并根据计数值的变化进行状态转换，如下图所示:

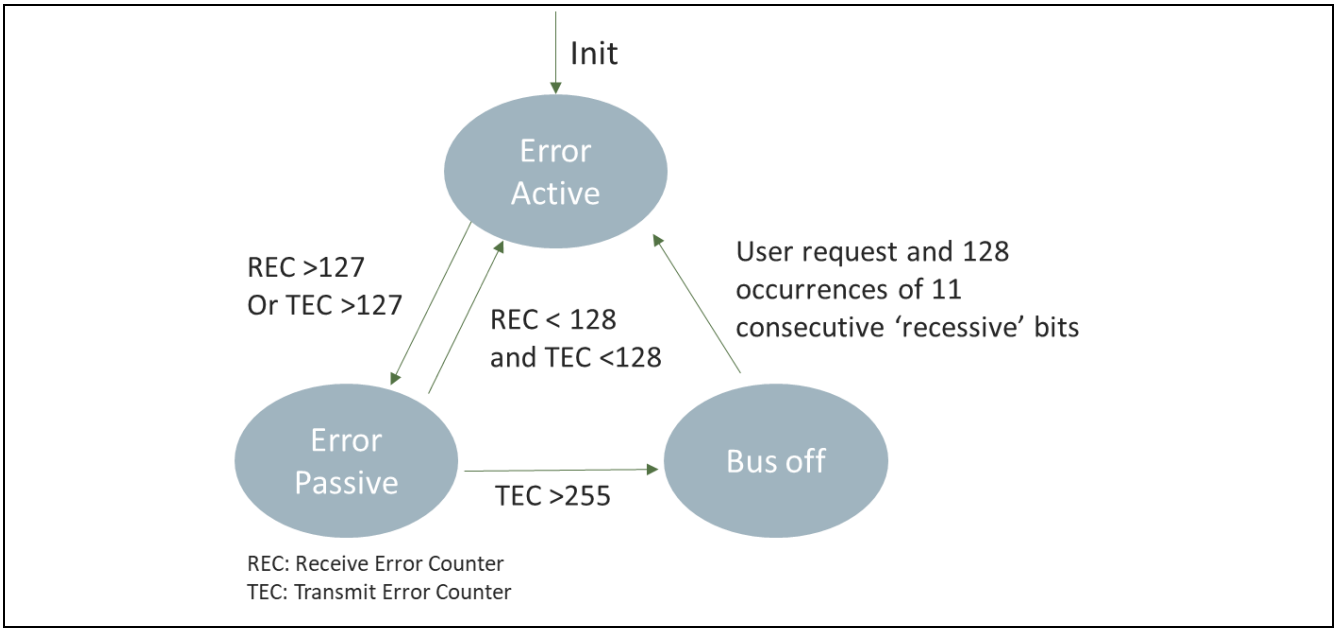

图 1-1 CAN 错误及 bus off 恢复机制

通常在 Error Active 以及 Error Passive 两种状态下的节点，MCU 不需要额外的软件去介入处理，节点会自动侦测错误的计数量并恢复至前一状态中。若错误计数大于一定数值，节点进入 bus off，在该状态下代表节点已无法进行自动恢复，并且不能传送或接收总线的数据。根据 CAN Specification Rev. 2.0，节点进入到 bus off 的状态后，当 CAN controller 检测到 128 次 11 个连续的隐性位后即可恢复通信，若以 baud rate 设定 500K 的 CAN Bus 来看，时间仅仅为几毫秒，通常该节点不太可能已经排除了造成 bus off 的实际原因，因此，贸然将该节点重新初始化恢复通讯，可能只是重新进到 bus off 的状态。基于上述原因，本范例提供一 bus off 的处理方式，当软件检测到节点进到 bus off 状态时不会立即恢复通讯，而是等到确认造成错误的状态排除后，再进行 bus off recovery 的步骤，让节点重新进入总线并进行数据的接收及发送。

1.1 软件架构

软件流程如图 1-2 所示， 使用三片带有 M487 Advance 上板的 NuMaker-PFM-M487 ，将板上 CAN bus 接口接上后，按下 reset 键范例程序便开始运行。一开始会先初始化 CAN controller、设定 baud rate、打开中断以及设定 rx buffer。之后每次透过 tx buffer 传送 CAN 数据包前，都会先确认 bus off 的标志是否置位(CAN controller 侦测到后会发出中断)。当软件检测到 bus off 标志置位后，系统便无法再接收 bus 上的信息，软件也会停止发送数据包到 bus 上，并等待恢复节点的信号产生。

Bus off 恢复机制不能通过设置或重置 Init bit 来缩短。如果设备进入 bus off，M487 将自行设置 Init bit，停止所有总线活动。一旦 Init 被 CPU 清除，设备将等待 129 次的 idle 信号（129 * 11 个连续的隐性位）出现，然后恢复正常操作。在总线关闭恢复机制结束时，错误管理计数器将被重置。在重置 Init bit 后的等待期间，每当系统侦测到 11 个隐性位序列时，都会将 Bit0Error 设为 1 写入状态寄存器，使系统能够及时检查 CAN bus 是否没有停留在显性状态，或者持续受到干扰。而在范例中是透过按下 M487 Advance 上板的 SW1 按键，来通知软件进行节点恢复程序，因此，当确认造成 bus off 的故障已排除后，便可按下 SW1 交由软件完成操作。

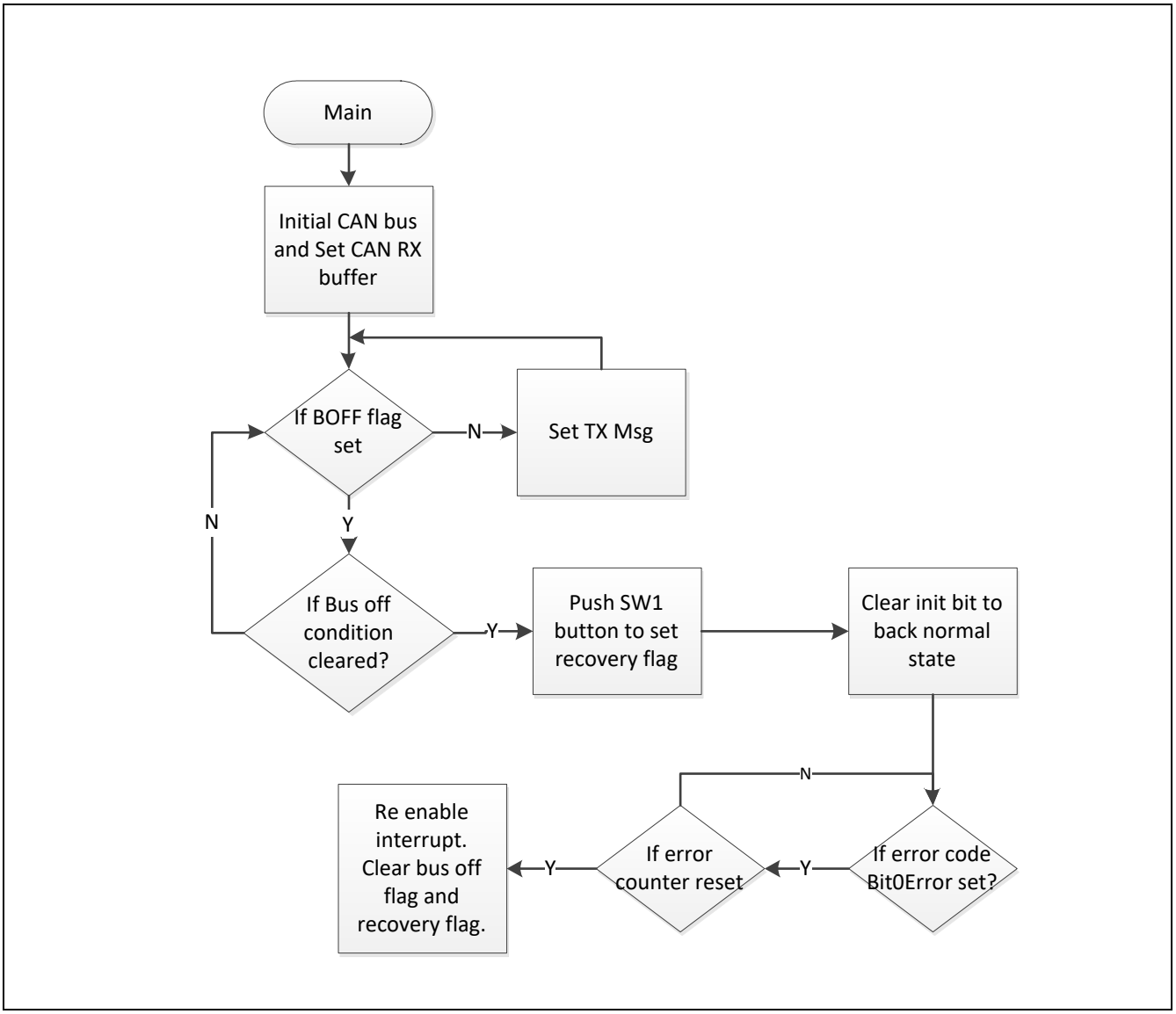
Main
Initial CAN bus and Set CAN RX buffer
If BOFF flag set
N
Set TX Msg
Y
N
If Bus off condition cleared?
Y
Push SW1 button to set recovery flag
Clear init bit to back normal state
N
If error code Bit0Error set?
Y
If error counter reset
Y
Re enable interrupt. Clear bus off flag and recovery flag.

图1-2 软件流程图

1.2 功能介绍

1. 将三片 M487 Advance 上板的 CAN 接口连接至面包板上后，数据包的传输情形如图 1-3。

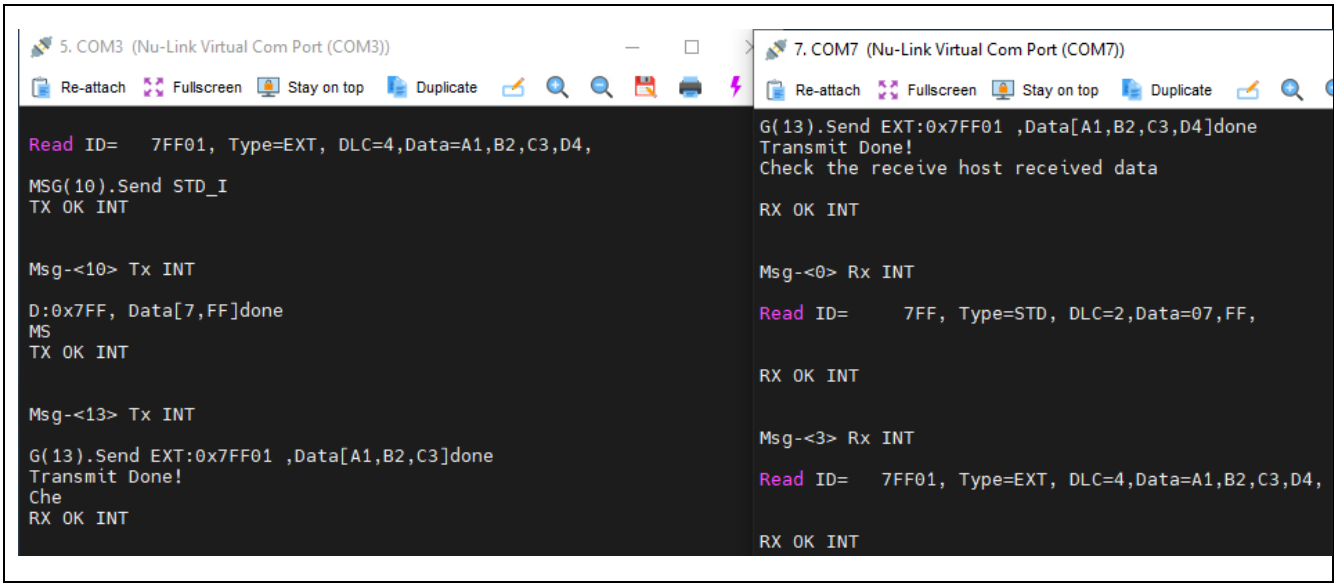

图 1-3 CAN Bus 正常收发数据包

2. 透过短路 CAN transceiver 上的 CAN_H/CAN_L 产生 bus off 如图 1-4，移除短路后按下 SW1 即可恢复到图 1-3。

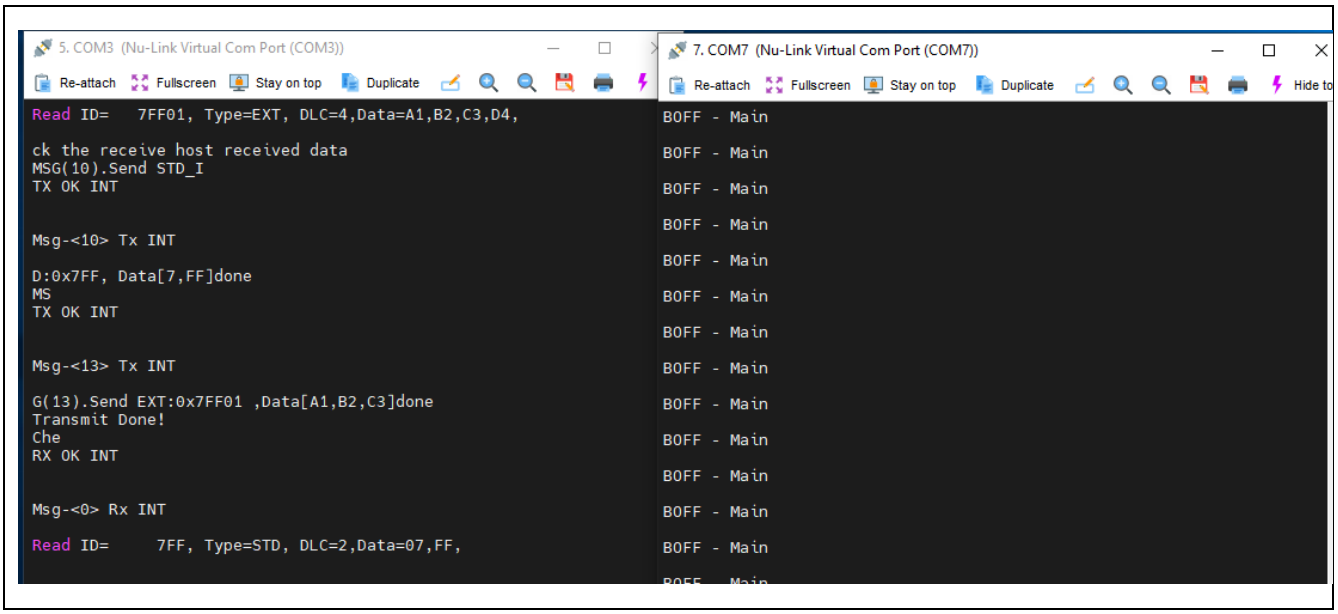

图 1-4 CAN bus off 状态

2. 程序介绍

以下程序实现了章节 1.1 的架构。

```
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. 软硬件需求

3.1 软件需求

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

3.2 硬件需求

- NuMaker-PFM-M487 v3.0
- M487 Advance board Ver. 4.0
- 杜邦线, 面包版
- 脚位连接示意图
  - 将 CAN_H (PA.5) pin 以及 CAN_L (PA.4) pin 脚连接到 bus (面包版)上。

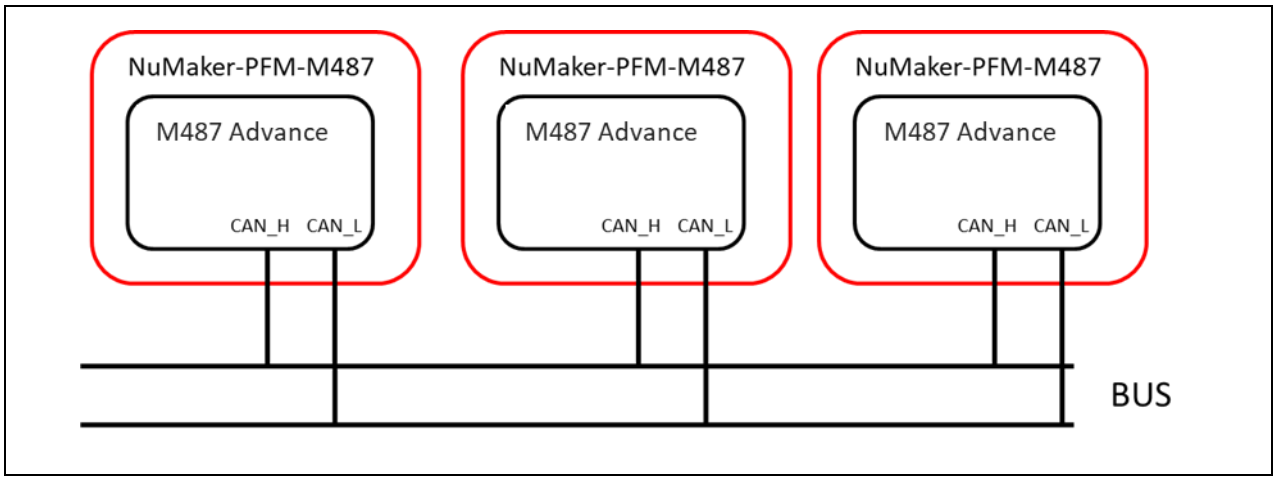

图 3-1 CAN bus 脚位连接示意图

4. 目录信息

目录架构如下所示。

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

图 4-1 目录信息

5. 范例程序执行

1. 打开 Keil Project
2. 编译范例程序后烧入至 NuMaker-PFM-M487。
3. 完成三片版子烧入后，将 M487 Advance 上板的 CAN bus 接口接至面包板。
4. 按下 reset 后开始进行 CAN bus 数据包对传。
5. 将其中一片板子的 CAN_H/CAN_L 从面包板上移除，并透过短路该 CAN_H/CAN_L 制造 bus off。
6. 移除短路，重新将该片 NuMaker-PFM-M487 的 CAN_H/CAN_L 接到面包板上，此时另外两片仍继续正常收发数据包。
7. 按下 M487 Advance 上板 SW1，对重新连接到 bus 上的 NuMaker-PFM-M487 进行节点恢复。
8. 该节点重新完成初始化，开始进行数据包收送。

6. 修订纪录

Date	Revision	Description
2022.11.28	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.