CAN Bus-Off Recovery Mechanism

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example is an application code for CAN bus, which describes how to recover and let a device re-enter the bus for packet transmission and reception when a bus off occurs at the device node.
BSP Version	M480 Series BSP CMSIS V3.05.003
Hardware	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. Overview

This example is an application code for CAN bus. In order to prevent a device from being unable to send and receive data correctly due to its own problems (such as hardware damage) or external interference and thus affecting other normal node communication, CAN bus has a hardware error diagnosis function.

The CAN General Specification states that each CAN controller has one transmit error counter and one receive error counter. Depending on the counters, the node will be in different error states, as shown in Figure 1-1.

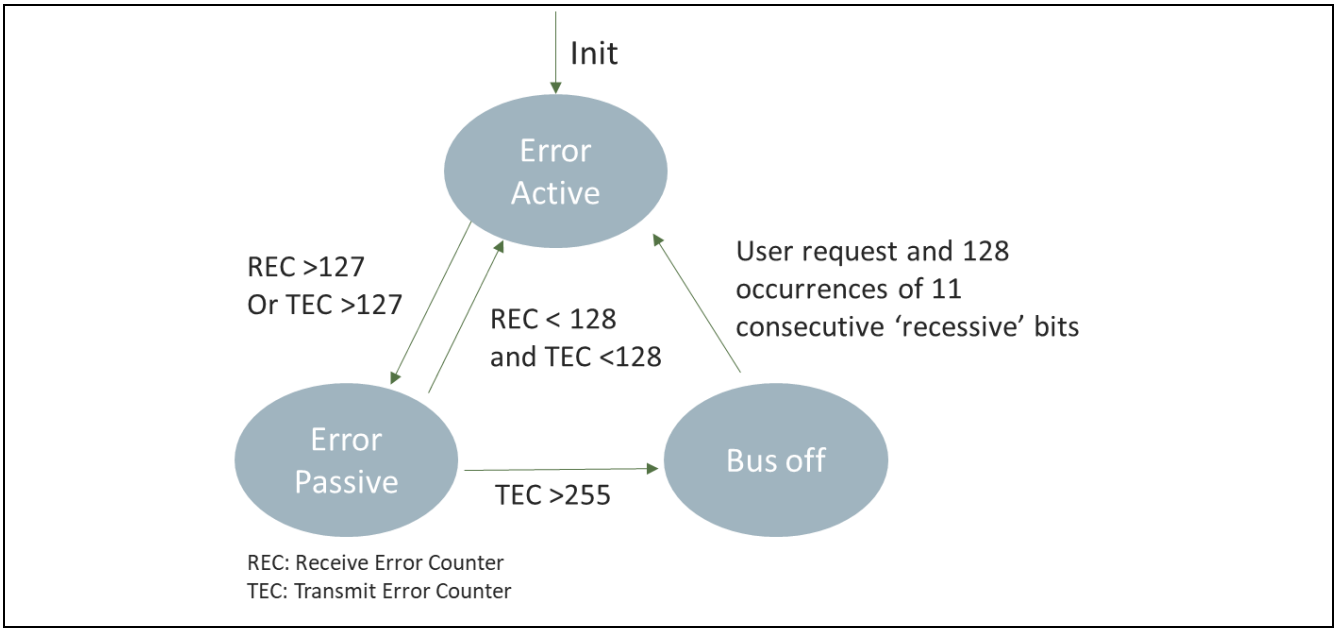

Figure 1-1 CAN Errors Management and Bus-off Recovery

Usually, the MCU does not need additional software to handle it in both Error Active and Error Passive states. The node automatically detects the number of error counters and restores the previous state when the counter is below the specified value.

If the error count is greater than a specified value, the node enters bus off. This state indicates that the node is no longer capable of automatic recovery and cannot transmit or receive data on the bus. According to CAN Specification Rev. 2.0, after the node enters the bus off state, communication can be resumed when the CAN controller detects 128 consecutive recessive bits 11 times. For example, with CAN bus at a baud rate of 500K, the time is only a few milliseconds. It is unlikely that the node has ruled out the cause of the bus off. Therefore, reinitializing the node and resuming communication may just be re-entering the bus-off state. This example provides a handler when the device goes to bus off. When the MCU detects that the node has entered the bus-off state, it will not immediately resume the node's communication. Wait until the status that caused the error is eliminated before proceeding with the bus-off recovery procedure.

1.1 Software Architecture

The software flowchart is shown in Figure 1-2. Use three NuMaker-PFM-M487 boards with M487 Advance board, connect the CAN bus interface on the board, press reset, and the sample program will run. At first, the CAN controller is initialized, the baud rate is set, the interrupt can be set, and the RX buffer is set initially. After that, each time the CAN packet is sent through the TX buffer, it will be checked whether the bus-off flag is raised (the CAN controller will issue an interrupt after detecting it). When the software detects that the bus-off flag is raised, the system can no longer receive data on the bus. The software also stops sending packets to the bus and waits for the recovery signal.

The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting the **Init** bit. If the device goes into the bus-off state, it will set **Init** of its own accord, stopping all bus activities. Once **Init** has been cleared by the CPU, the device will then wait for 129 occurrences of bus Idle (129 * 11 consecutive recessive bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters will be reset. During the waiting time after resetting **Init**, each time a sequence of 11 recessive bits has been monitored, a **Bit0Error** code is written to the Status Register, enabling the CPU to readily check whether the CAN bus is not stuck at dominant or continuously disturbed and to monitor the proceeding of the bus-off recovery sequence.

In the example, the software is notified by pressing the SW1 button on the M487 Advance board. When it is confirmed that the fault causing bus-off has been eliminated, SW1 can be pressed and the software can complete the operation of restoring the CAN node.

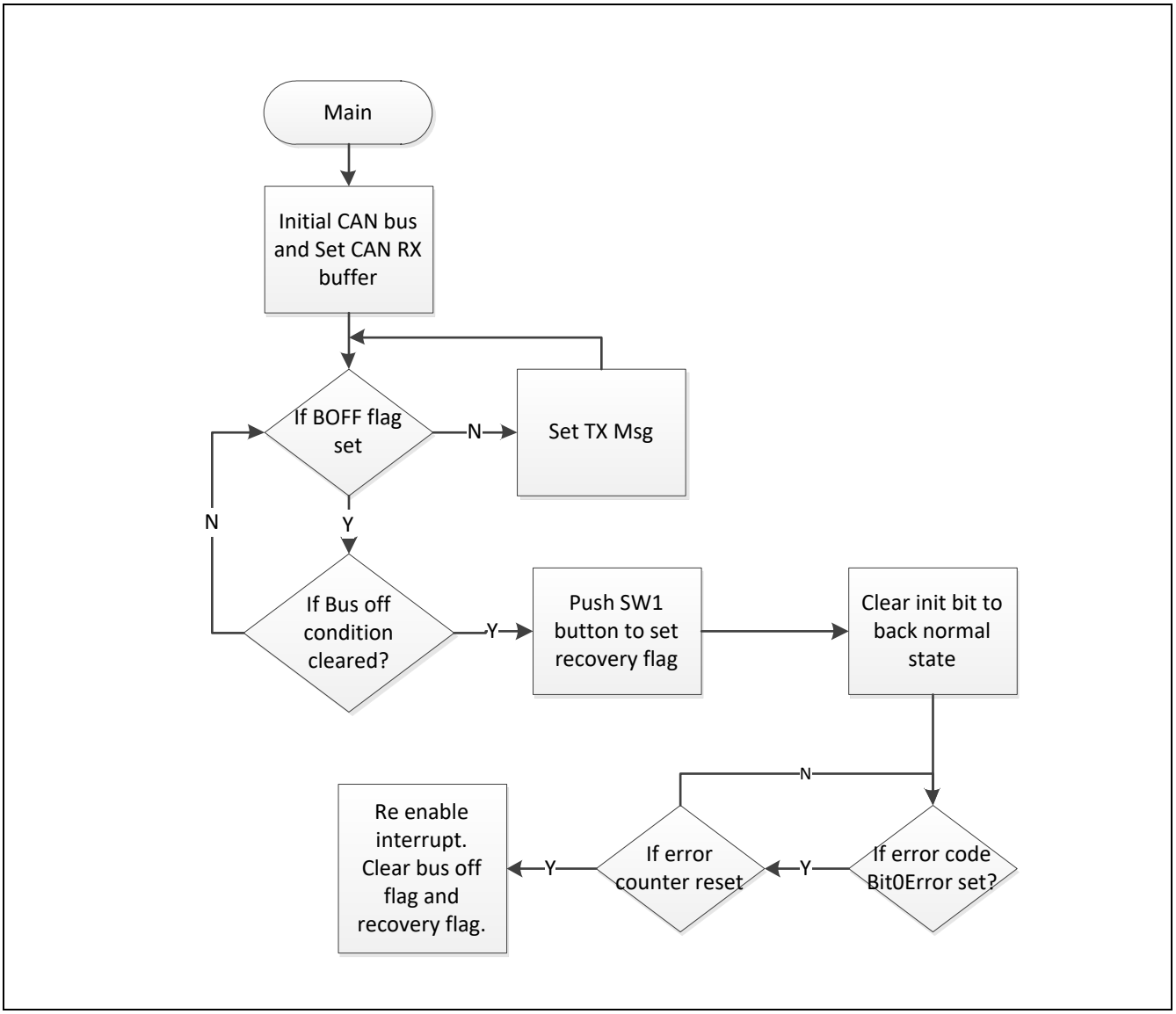

Figure 1-2 Software Flowchart

1.2 Function Description

1. Connect the CAN bus interface on the three NuMaker-PFM-M487 boards to the breadboard. The packet transmission is shown in Figure 1-3.

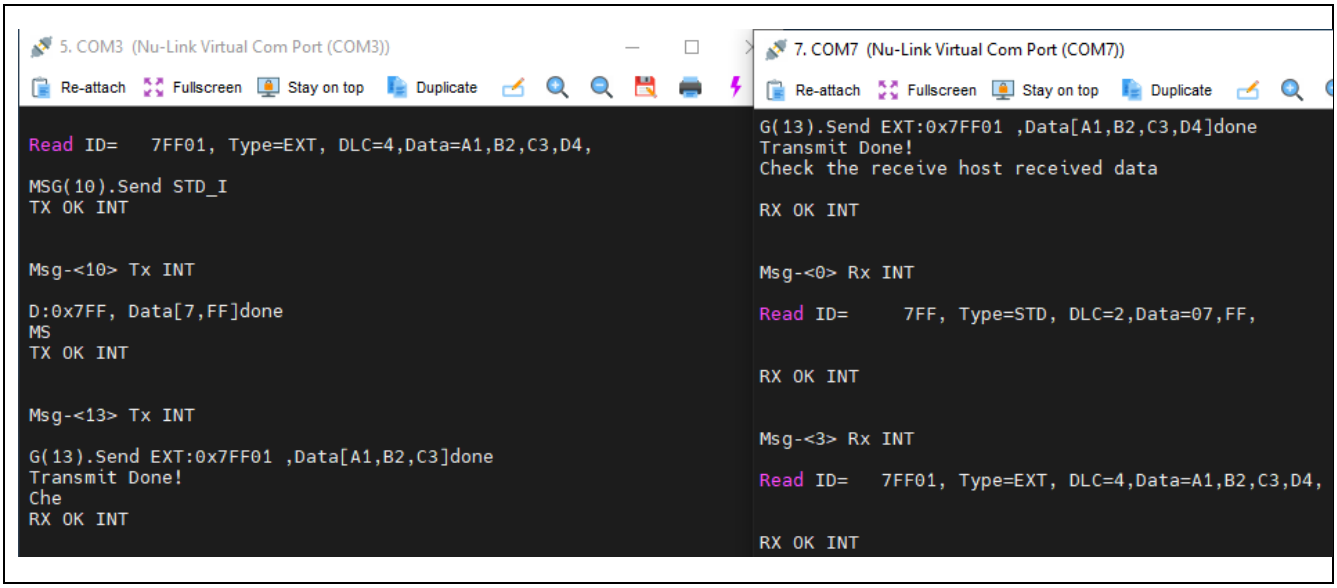

Figure 1-3 Normal Packet Transmission and Reception

2. Generate a bus off by shorting CAN_H/CAN_L on the CAN transceiver as shown in Figure 1-4. After removing the short circuit, press SW1 to restore as shown in Figure 1-3.

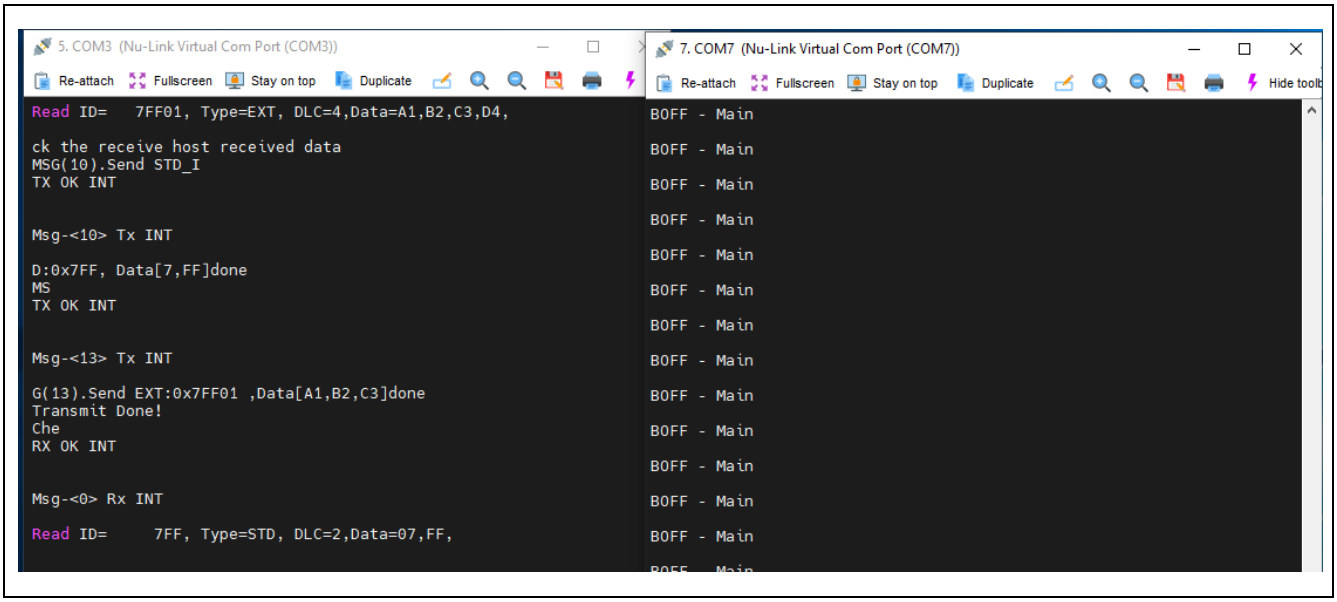

Figure 1-4 CAN Bus Off

2. Code Description

The code below realizes the architecture described in section 1.1.

```
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. Software and Hardware Requirements

3.1 Software Requirements

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

3.2 Hardware Requirements

- NuMaker-PFM-M487 V3.0
- M487 Advance board Ver. 4.0
- Dupont line, Breadboard
- Pin Connect
  - Connect the CAN_H (PA.5) pin and CAN_L (PA.4) pin to the bus (Breadboard).

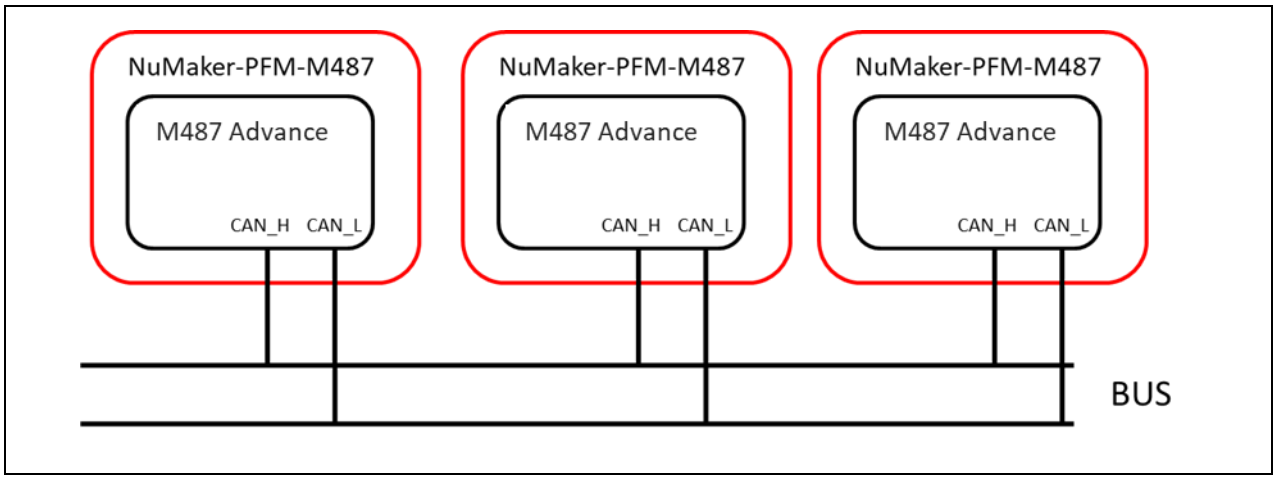

Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

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

Figure4-1 Directory Structure

5. Example Code Execution

1. Open the Keil Project.
2. Build the example code and program into the NuMaker-PFM-M487.
3. Connect to the CAN bus interface on the M487 Advance board as shown in Figure 3-1.
4. Press reset to start the CAN bus packet transmission.
5. Remove the CAN_H/CAN_L of one of the boards from the breadboard and short-circuit the CAN_H/CAN_L to generate bus off.
6. Remove the short circuit and reattach the CAN_H/CAN_L of the NuMaker-PFM-M487 to the breadboard. At this time, the other two pieces continue to receive and receive the data packet normally.
7. Press SW1 button of the M487 Advance board to perform node recovery on boards reconnected to bus.
8. The node reinitializes and begins packet transmission.

6. Revision History

Date	Revision	Description
2022.11.28	1.00	Initial version.

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.