Transferring CAN Message Periodically by NUC131

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This sample code uses NUC131SD2AEU to transfer CAN messages periodically and debug CAN interface.
BSP Version	NUC131 BSP CMSIS V3.00.005
Hardware	NuMaker-NUC131U 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 Overview

When configuring four messages, two messages of them will be updated every 2 seconds and the other two messages will not be updated. Only one of the four messages will be transferred actively every 2 seconds and the other three messages always wait for a remote frame.

1.1 Demo Result

CAN interface will transfer message of ID 188 every 2 seconds while executing this sample code on NUC131. After NUC131 receives a remote frame of ID=6789,128 and 12, the corresponding data message will be transferred out as shown in Figure 1-1.

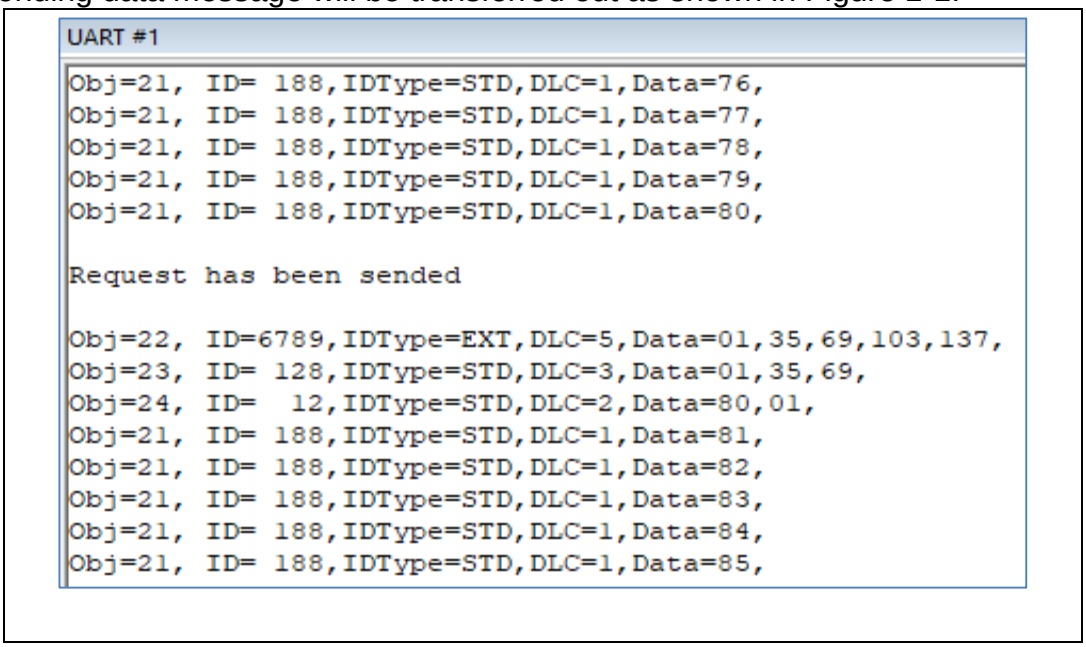

Figure 1-1 Messages Displayed on PC

If you have no other CAN transceiver tool, you can use NUC131 project *NUC131_CAN_Rx_Display_Message* provided as the CAN message receiver to receive CAN messages.

2 Code Description

First, initialize the NUC131 in main file, and initialize clock, GPIO, Timer1, and CAN interface. Then, after configuring message-objects of NUC131, NUC131 will transfer messages every 2 seconds.

```
int main(void)
{
    uint16_t item = 0;

    SYS_UnlockReg();
    SYS_Init();                            // Init System, IP clock and multi-function I/O
    SYS_LockReg();
    Timer1_Init();                        // Continuous mode for delay
                          // division factor of clock= 4,So CAN clock = 48M/4=12M
    CAN_Init(4,20,5);     // Clock number of one bit = 20, so baud rate = 12M/20= 600Kbps

    GPIO_SetMode(PB, BIT14, GPIO_PMD_OUTPUT);              // PB4 is only for indication
    GPIO_SetMode(PC, BIT15, GPIO_PMD_OUTPUT);
    PC->DOUT &= ~0x8000 ;                                  // Enable CAN transceiver

     /* This 2 data frames won't transfer actively. They always wait for remote frame */
    CAN_WrtObj(0x128,  3, 0x67452301, 0xefcdab89, 3) ;       //Data frame, length = 3byte
    CAN_WrtObj(0x6789, 5, 0x67452301, 0xefcdab89, 4) ;       //Data frame, length = 5byte

    while (1)
    {
        Delayus(2*1000*1000) ;                          // Delay 2 seconds
        PB->DOUT ^= 0x4000 ;                            // Indication
        CAN_WrtObj(0x188, 1, ++item, 0xefcdab89, 10) ; // update data in object 10, 1 byte
        CAN_WrtObj(0x12,  2,   item, 0xefcdab89, 12) ; // update data in object 12, 2 byte

   /* Only Message 10 will be transfered actively. Other message wait for remote frame */
        CAN_TransObj(10) ;
    };
}
```

3 Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - NUC131_Series_BSP_CMSIS_V3.00.005
- IDE version
  - Keil uVersion 5.26

3.2 Hardware Requirements

- Circuit components
  - NuMaker-NUC131U V1.0
- Pin Connect

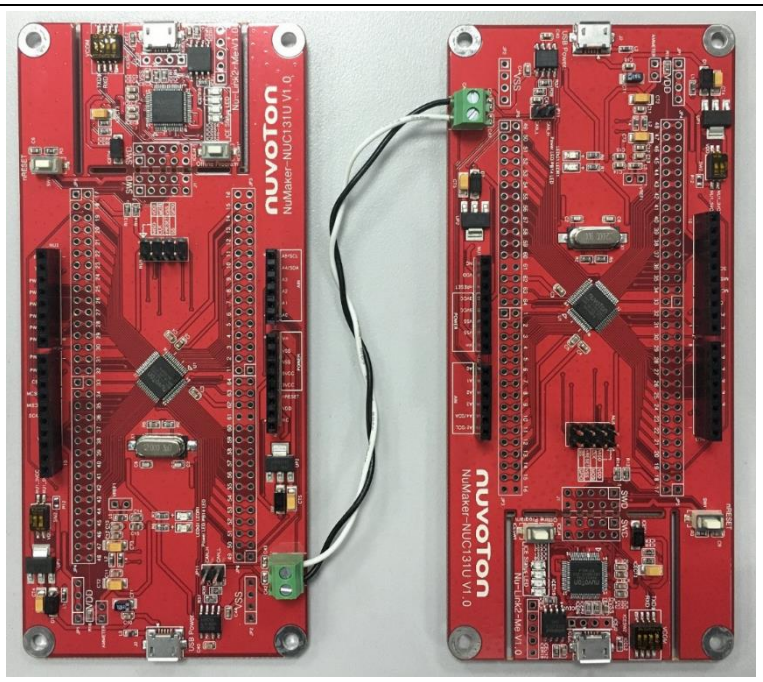

Figure 3-1 Pin Connect

4 Directory Information

The directory structure is shown below.

EC_NUC131_Transfer_CAN_Message Periodically_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

Figure 4-1 Directory Structure

5 Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *NUC131_Periodically_transfer_Message.uvproj*
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6 Revision History

Date	Revision	Description
2021.09.28	1.00	1. Initially issued.

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.