M451 CAN ISP

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	This example code uses M451 CAN interface to upgrade firmware for APROM.
BSP Version	M451 Series BSP CMSIS V3.01.001
Hardware	NuMaker-PFM-M453

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 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

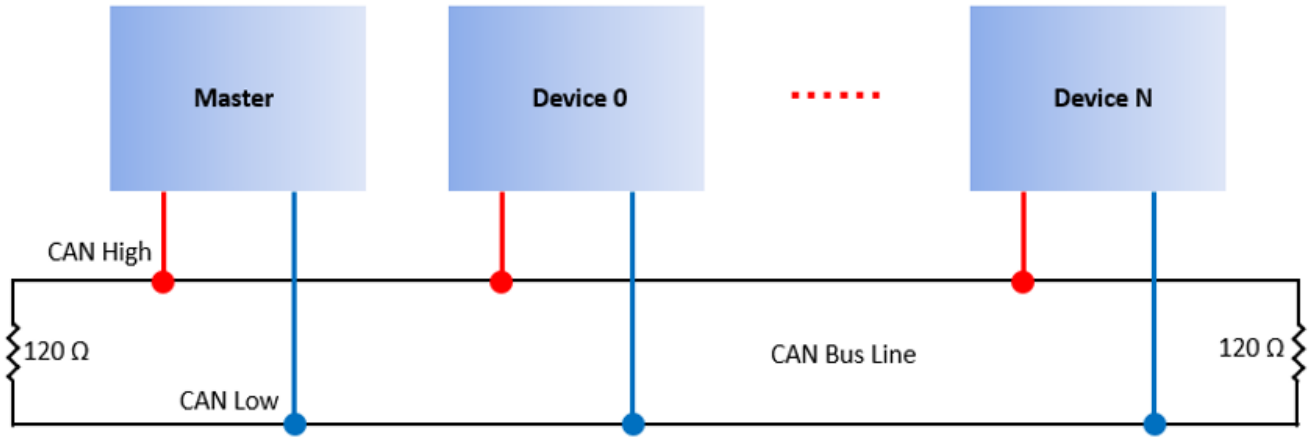
1 Function Description

1.1 Introduction

The C_CAN consists of the CAN Core, Message RAM, Message Handler, Control Registers and Module Interface. The CAN Core performs communication according to the CAN protocol version 2.0 part A and B. The bit rate can be programmed to values up to 1MBit/s. For the connection to the physical layer, additional transceiver hardware is required.
This example code uses M451 CAN interface to upgrade firmware on the NuMaker-PFM-M453 board.

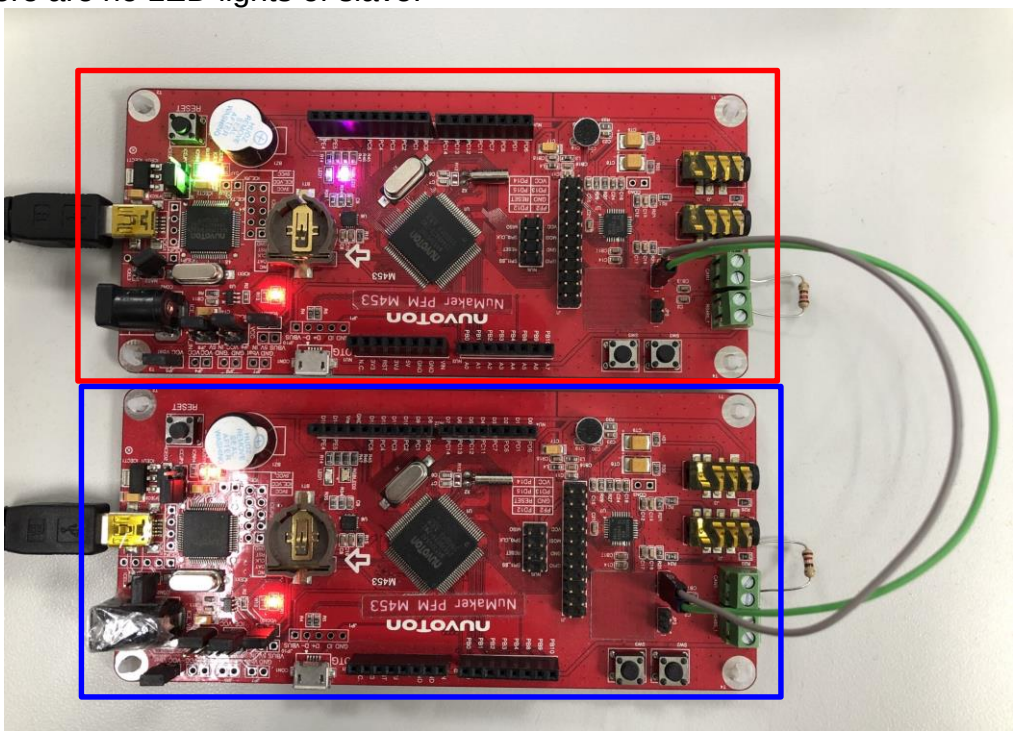
1.2 Principle

As shown in the picture below, the CAN master can transmit data to the CAN slave through the CAN bus line to achieve the firmware update function.

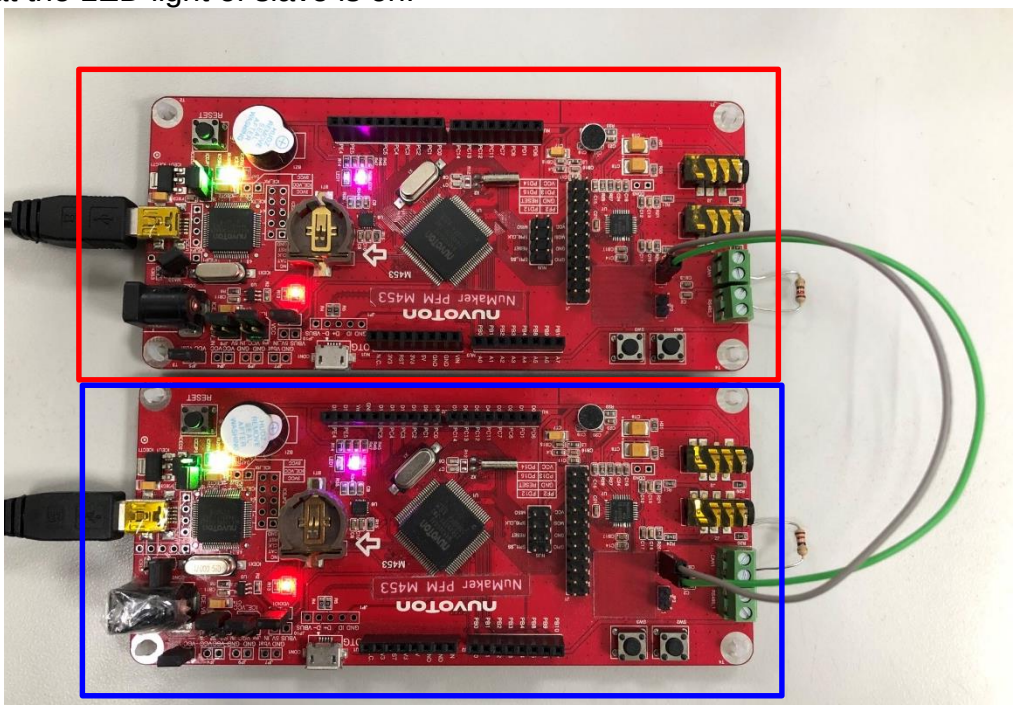


1.3 Demo Result

After executed this example code, user will be seeing below picture. The master board is red rectangle; the slave board is blue rectangle.
The first photo is that the firmware has not been updated through the CAN interface. You can see that there are no LED lights of slave.



The second photo shows that the firmware has been updated through the CAN interface. You can see that the LED light of slave is on.



2 Code Description

The master main() function is used to initialize platform for CAN interface and FMC. After that, it will send commend to slave for upgrading slave firmware.

```
#define PLL_CLOCK                       48000000
#define CAN_BAUD_RATE                   250000
#define Master_ISP_ID                   0x800
#define Device0_ISP_ID                  0x802
#define CAN_ISP_DtatLength              0x08
#define CAN_FailTime                    0xff
#define StartProgramAddress             0
int main(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();
    /* Init System, IP clock and multi-function I/O */
    SYS_Init();
    /* Init CAN port */
    can_init();
    /* Enable FMC ISP function */
    FMC_Open();
    if(APROM_detect()>0)
    {
        /* stat update program */
        if(CAN_ISP_Update(CAN0)>0)
        {
            goto _APROM;
        }
    }
    while(1);
    _APROM:
    /*clear bit*/
    outpw(&SYS->RSTSTS, 3);
    outpw(&FMC->ISPCTL, inpw(&FMC->ISPCTL) & 0xFFFFFFFC);
    outpw(&SCB->AIRCR, (V6M_AIRCR_VECTKEY_DATA | V6M_AIRCR_SYSRESETREQ));
    /* Trap the CPU */
    while(1);
}
```

The master CAN_ISP_Update () function is used to update slave firmware through CAN interface. Firstly, the function will send command to get slave ACK.

```
uint8_t CAN_ISP_Update(CAN_T *tCAN)
{
    STR_CANMSG_ISP CAN_buffer;

    uint8_t ErrorCnt=0, CompareFail_flag=1;
    memset(&CAN_buffer, 0, sizeof(CAN_buffer));
    CAN_buffer.Address = StartProgramAddress;

    while(1)
    {
        if(ErrorCnt++ == CAN_FailTime)              //ErrorCnt = 0xff CAN bus fail
        {
            return FALSE;
        }
        CAN_Package_Tx(CAN0, &CAN_buffer);          //send CAN ISP Package

        u8CAN_PackageFlag=0;
        TimeOut_flag=0;
        CLK_EnableSysTickINT(2000);                 // System Tick timer value us
        //Wait CAN ISP Package ACK
        while(u8CAN_PackageFlag == 0 && TimeOut_flag == 0);
        CLK_DisableSysTick();
        if(TimeOut_flag==0)
        {
            CompareFail_flag=1;
            //check CAN ISP Package
            if(((long)CAN_buffer.Address) != (*(long*)rrMsg.Data)) CompareFail_flag=0;
            if(CompareFail_flag)
            {
                //check CAN ISP EndCode
                if(CAN_buffer.Address == Chip_EndAddress)  return TRUE;
                ErrorCnt=0;
                CAN_buffer.Address += 4;            //for next address
            }
            else
            {
                if(CAN_buffer.Address < FMC_FLASH_PAGE_SIZE) CAN_buffer.Address =
StartProgramAddress;
                else CAN_buffer.Address = (CAN_buffer.Address / FMC_FLASH_PAGE_SIZE) *
FMC_FLASH_PAGE_SIZE;
            }
        }
    }
}
```

The slave main() function is used to initialize platform for CAN interface and FMC. After that, it will receive command from master.

```
#define PLL_CLOCK                       48000000
#define CAN_BAUD_RATE                   250000
#define Master_ISP_ID                   0x800
#define Device0_ISP_ID                  0x802
#define CAN_ISP_DtatLength              0x08
#define CAN_FailTime                    0xff
int main(void)
{
    /* Unlock protected registers */
    SYS_UnlockReg();
    /* Init System, IP clock and multi-function I/O */
    SYS_Init();
    /* Init CAN port */
    can_init();
    /* Enable FMC ISP function */
    FMC_Open();
    if(APROM_detect()>0)
    {
        /* stat update program */
        if(CAN_ISP_Update(CAN0)>0)
        {
            while(u8CAN_AckFlag);
            goto _APROM;
        }
    }

    while(1);
    _APROM:
    /*clear bit*/
    outpw(&SYS->RSTSTS, 3);
    outpw(&FMC->ISPCTL, inpw(&FMC->ISPCTL) & 0xFFFFFFFC);
    outpw(&SCB->AIRCR, (V6M_AIRCR_VECTKEY_DATA | V6M_AIRCR_SYSRESETREQ));
    /* Trap the CPU */
    while(1);
}
```

The slave CAN_ISP_Update () function is used to receive the latest data from master to update APROM firmware through CAN interface.

```
uint8_t CAN_ISP_Update(CAN_T *tCAN)
{
    STR_CANMSG_ISP CAN_buffer;
    uint32_t APROM_Data=0;
    uint8_t Verify_flag=1;
    memset(&CAN_buffer, 0, sizeof(CAN_buffer));
    FMC_EnableAPUpdate();

    while(1)
    {
        u32CANWaitCount = 0;
        u8CAN_PackageFlag=0;
        Verify_flag=1;
        CLK_EnableSysTickINT(10000);                          // System Tick timer value us
        while(u8CAN_PackageFlag == 0 && TimeOut_flag == 0); //receive CAN ISP Package
        if(TimeOut_flag)
        {
            CLK_DisableSysTick();
            FMC_DisableAPUpdate();
            return FALSE;
        }
        CLK_DisableSysTick();
        memcpy(&CAN_buffer.Address, &rrMsg.Data, 4);       //update address
        if(((CAN_buffer.Address % FMC_FLASH_PAGE_SIZE) == 0) || (CAN_buffer.Address == 0))
        {
            FMC_Erase(CAN_buffer.Address);
            if(VerifyData(CAN_buffer.Address, CAN_buffer.Address + FMC_FLASH_PAGE_SIZE,
0xFFFFFFFF) < 0) Verify_flag=0;
        }
        if(Verify_flag)
        {
            memcpy(&APROM_Data, &rrMsg.Data[4], 4);        //update data
            FMC_Write(CAN_buffer.Address, APROM_Data);     //program ROM
            u8CAN_AckFlag = 1;
            CAN_Package_ACK(CAN0);                         //send CAN ISP Package (ACK)
            if(CAN_buffer.Address == Chip_EndAddress)      //check CAN ISP EndCode
            {
                FMC_DisableAPUpdate();
                return TRUE;
            }
        }
    }
}
```

3 Software and Hardware Environment

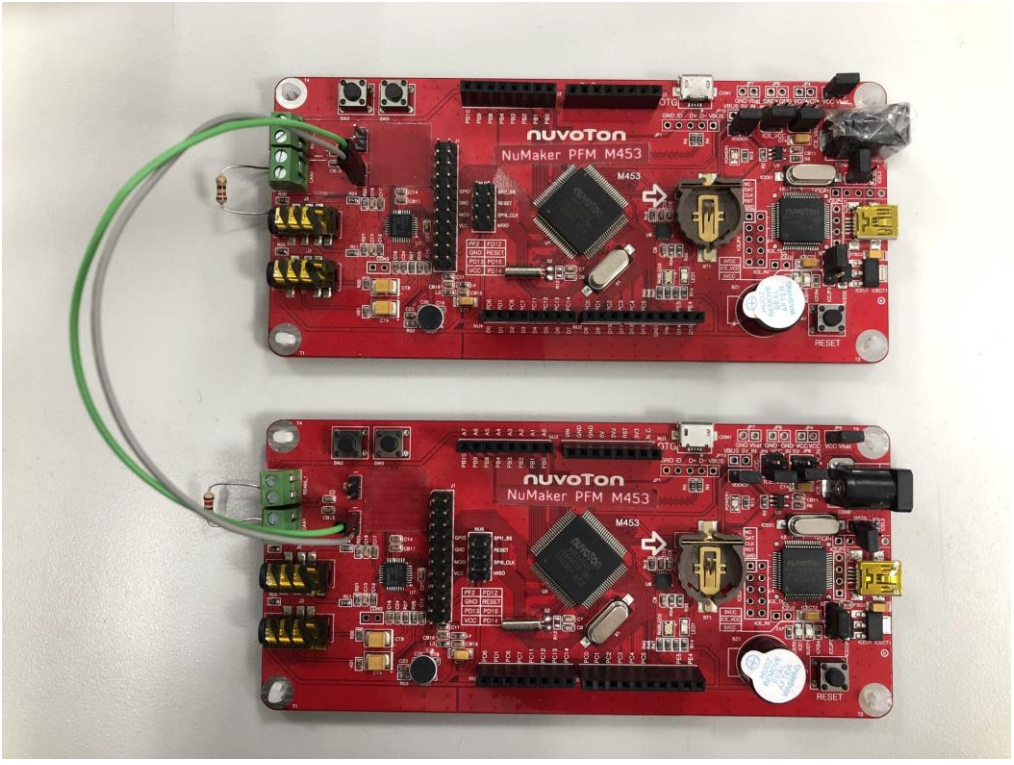
Software Environment

- BSP version
  - M451 Series BSP CMSIS V3.01.001
- IDE version
  - Keil uVersion 5.26

Hardware Environment

- Circuit components
  - NuMaker-PFM-M453 board
- Diagram

The below picture is to show how to connect between CAN master board and CAN slave board.



4 Directory Information

- EC_M451_CAN_ISP_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

5 How to Execute Example Code

1. Browsing into sample code folder by Directory Information (section 4) and double click CAN_ISP_Device.uvproj, CAN_ISP_Master.uvproj, GPIO_AP1.uvproj.
2. Enter Keil compile mode
   a. Build
   b. Download
   c. Start/Stop debug session
3. Enter debug mode
   a. Run

6 Revision History

Date	Revision	Description
Jul. 18, 2019	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.