XC2287M HOT
Solution CAN
Serial Communication using the CAN

Device: XC2287M-104F80
Compiler: Tasking Viper 2.4r1
Code Generator: DAvE 2.1
Serial Communication using the CAN

Let’s get started now!
 XC2287M HOT Exercise CAN
Interaction of Development Tools

DAvE

IDE

Compiler
Assembler
Linker
Locator

Debugger
int main()
char a;
long b;
...

Programming Tool

Debugger/Emulator

Target

SW

HW
HOT Exercise CAN

Objective

■ Use a terminal program to send ASCII characters via the serial port
■ Receive these ASCII characters using ASC0
■ Loop back the characters to ASC0 via CAN Node 0/1
■ Transmit the received ASCII characters back to the PC
HOT Exercise CAN
RS232 & CAN settings

■ **RS232 Settings**
  - Baud Rate = 19200 Baud
  - 8 bit data, 1 stop bit, no parity
  - Receive interrupt
  - RXD:P7.4 TXD: P7.3

■ **CAN Settings**
  - Baud Rate = 500 kBaud
  - Message Object 2 : CAN node 0, transmit, 1 byte, 11-bit ID = 0x11
  - Message Object 5 : CAN node 1, receive, 1 byte, 11-bit ID = 0x11
    RX-interrupt, level 5
  - RX:2.6, TX: 2.5 Boards Rev1.1
  - RX:2.4, TX: 2.2 Boards Rev1.1
HOT Exercise CAN Block Diagram

PC

RS 232 via USB
- 19200 Baud
- 8 n 1
- RX interrupt

Transfer by
Interrupt handler

Receive Buffer

Transmit Buffer

USIC 0 – ASC 0

U0C0 Receive IRQ
Priority 3

CAN Node 0

CAN MSG OBJ 2

ID=0x11

CAN_L

CAN_H

CAN Node 1

CAN MSG OBJ 5

MO5 Receive IRQ
Priority 5

ID=0x11

CAN_L

CAN_H

Ext. CAN Bus

RX: P2.6
TX: P2.5

RXD: P7.4
TXD: P7.3

Transfer by Interrupt handler

• 19200 Baud
• 8 n 1
• RX interrupt
HOT Exercise CAN Flow Chart 1/2

- Init CAN Module
  - Module enable
  - Fractional Divider
  - EINIT write protected register

- Init CAN Node
  - Enable node and interrupts
  - Configure pin connections
  - Set baud rate
  - Assign Message Objects to node

- Init Message Objects (MO)
  - RX/TX mode
  - data length
  - INT pointer and enable
  - Identifier
  - Data content
  - Enable MO

- Start Node on CAN Bus
HOT Exercise CAN Flow Chart 2/2

- USIC 0 Channel 0 (ASC) Interrupt
  - Clear ASC status register
  - Received Character -> MOData2
  - Sent MOData2
  - Exit

- CAN 1 Message Object 5 Interrupt
  - Clear CAN status register
  - Received Character -> U0C0 ASC TX
  - Sent U0C0 ASC
  - Exit
HOT Exercise CAN - DAVeE Configurations
Start DAVeE

- Start DAVeE
  - Click on the DAVeE

- Create a new project (Startup Dialog pop up automatically)
  - Click on ‘Create a new project’ or select File -> New
  - Select microcontroller: ‘XC2287M’
HOT Exercise CAN
Start DAvE (cont.)

1. Select DAvE.
2. Create a new project.
3. Choose the XC2287M microcontroller.
HOT Exercise CAN - DAvE Configurations

- Project Settings
- Close the window
Save your DAvE project

- Path: 
  C:\IFX_HOT\XC2287M\Examples\CAN

- Project name: 
  CAN\CAN.dav
Save your DAvE Project File

1. Select "Save Project As" in the DAvE Project Window.
2. Choose the directory where you want to save your project.
3. Enter the file name "CAN.dav" in the "Dateiname:" field.
4. Click "Speichern" to save your project.
HOT Exercise CAN - DAvE Configurations

ASC settings

- **RS232 Settings**
  - Baud Rate = 19200 Baud
  - 8 bit data, 1 stop bit, no parity
  - Receive interrupt
  - RXD:P7.4 TXD: P7.3
HOT Exercise CAN - DAvE Configurations

ASC settings

- **XC2287M**
  - **USIC0**: Click on the

![Diagram](image-url)
XC2287M

- **USIC0**:  
  - Select ASC for U0C0 protocol  
  - Click to exit
HOT Exercise CAN - DAvE Configurations
ASC settings

- **XC2287M**

- **USIC0, CH0**:  
  - Click on the

![Diagram of XC2287M with ASC settings configuration](image)
Configure CH0 ASC

- **ASC General:**
  - Enable module clock
  - Select P7.3 for TxD and P7.4 for RxD
HOT Exercise CAN - DAvE Configurations

ASC settings

- **Configure CH0 ASC**
  - **Control:**
    - Enable Receive Interrupt

![Image of Universal Serial Interface Channel (USIC) configuration](image.png)
Configure CH0 ASC

- **Interrupts:**
  - Drag ‘U0C0_0IC INT’ and drop it to Interrupt Level 3, Group 0
HOT Exercise CAN - DAvE Configurations

Configure CH0 ASC

Functions:
- Include ‘U0C0_ASC_vInit’
- Include ‘U0C0_ASC_vSendData’
HOT Exercise CAN - DAvE Configurations Port settings

**XC2287M**

- **Port:**
  - Click on the

![Diagram of XC2287M Port configurations]
HOT Exercise CAN - DAvE Configurations Port settings

- Parallel Ports

- Ports:
  - Configure Port 10
Configure Port 10

- Use P10.0 as general IO
- Set Direction to Out
- Close the window

Port 10:

1. Select Use P10.0 as general IO
2. Set Direction to Out
3. Close the window
Parallel Ports

Functions:

- Include ‘IO_vInit’
- Include ‘IO_vTogglePin’
**CAN Settings**

- Baud Rate = 500 kBaud
- Message Object 2: CAN node 0, transmit, 1 byte, 11-bit ID = 0x11
- Message Object 5: CAN node 1, receive, 1 byte, 11-bit ID = 0x11
  - RX-interrupt, level 4
- CAN 0: RX: P2.6, TX P2.5
- CAN 1: RX: P2.4, TX P2.2
HOT Exercise CAN - DAvE Configurations
MultiCAN settings

- **XC2287M**
  - MultiCAN:
    - Click on the

![Diagram](image)
Configure MultiCAN

- General:
  - Enable module
HOT Exercise CAN - DAvE Configurations
MultiCAN settings

- Configure MultiCAN
- General:
  - Select Node 0

![MultiCAN interface screenshot]
HOT Exercise CAN - DAvE Configurations
MultiCAN settings

Configure CAN Node 0

- **General:**
  - Select P2.6 for Receive Input and P2.5 for Transmit Output
  - Initialize the CAN node 0 automatically
HOT Exercise CAN - DAvE Configurations
MultiCAN settings

Configure CAN Node 0

Baud Rate:

- Required baud rate: 500 Kbaud
- Modify TSeg1/TSeg2 to get Real baud rate at 500 Kbaud
Configure MultiCAN

- General:
  - Select Node 1
Configure CAN Node 1

- General:
  - Select P2.4 for Receive Input and P2.2 for Transmit Output
  - Initialize the CAN node 1 automatically
**Configure CAN Node 1**

- **Baud Rate:**
  - Required baud rate: 500 Kbaud
  - Modify TSeg1/TSeg2 to get Real baud rate at 500 Kbaud
Configure MultiCAN

- **List1, 2:**
  - Drag ‘M02’ and drop it to List 1 (Node 0)
  - Drag ‘M05’ and drop it to List 2 (Node 1)
HOT Exercise CAN - DAvE Configurations MultiCAN settings

- **Configure MultiCAN**
  - **M0s:**
    - Select M02
Configure CAN Message Object (M02)

- **Object:**
  - Enable M02, Select Transmit data frames
  - Identifier: 0x011, Data Length: 1 data bytes
HOT Exercise CAN - DAvE Configurations
MultiCAN settings

- **Configure MultiCAN**
  
- **M0s:**
  - Select M05
Configure CAN Message Object (M05)

- Enable MO5, Select Receive data frames
- Identifier: 0x011, Data Length: 1 data bytes
Configure CAN Message Object (M05)

- **Interrupt:**
  - Enable receive interrupt
  - Use CAN SRN 0
Configure MultiCAN

- Interrupts:
  - Drag ‘CAN INT 0’ and drop it to Interrupt Level 4, Group 0
Configure MultiCAN

- Include 'CAN_vInit'
- Include 'CAN_vTransmit' and 'CAN_vLoadData'
Save your DAvE Project File

- Go to **File → Save (or Save As)** or click on

- Filename entered previously:
  
  "c:\IFX_HOT\XC2287M\Examples\CAN\CAN.dav"
HOT Exercise CAN - DAvE Configurations

Code Generation

Let DAvE Generate Code for You

- Go to **File → generate Code** or click on

- DAvE generated code files are

  - ‘CAN.c’, ‘CAN.h’
  - ‘U0C0.c’, ‘U0C0.h’
  - ‘IO.c’, ‘IO.h’
  - ‘USIC0.c’, ‘USIC0.h’
  - ‘MAIN.c’, ‘MAIN.h’
  - ‘SCS.c’, ‘SCS.h’
  - ‘XC22xxREGS.h’

- In general:
  - if the included function is a macro it is included in the ‘.h’ file
  - if the included function is a function it is included in the ‘.c’ file

† In general:
- if the included function is a macro it is included in the ‘.h’ file
- if the included function is a function it is included in the ‘.c’ file
Open Project Work Space

- Click on

- Filename: browse to “c:\IFX_HOT\XC2287M\Examples”

- Click ‘OK’
- Create New Project

- Click on Workbench (if not already there...)

1

![Image of Workbench interface with highlight on the 'Workbench' button]
1. Import DAVE Project
   - Click on File -> Import
   - Select Tasking VX-toolset for C166...
   - Click ‘OK’
Import DAVE Project

- Click `Infineon DAvE C166 Project´
- Click ‘Next’
Import DAvE Project

- Add Dave Project ‘CAN’
- Click ‘Finish’
Configure Target Board

- Select the project in the navigator
- Select ‘Project/Target Board Configuration’
- Select ‘Infineon XC2000/XE166 Easykit Board’
- Choose `XC2287M-104F´
- Click `Finish´

HOT Exercise CAN – Tasking VX Toolset
Software Hint

DAvE doesn’t change code that is inserted in the ‘USER CODE’ sections if you let DAvE regenerate the code. Therefore, whenever adding code to the generated code, write it into a ‘USER CODE’ section.

The code you really have to add looks like this:

```c
while(1)
{
    // USER CODE BEGIN (Main,4)
    BlinkLED();
    // USER CODE END
}
```
In the ISR function ‘U0C0_ASC_vi0IC(void)’ (almost at the end)

```c
_interrupt(U0C0_0INT) void U0C0_ASC_vi0IC(void)
{
    // USER CODE BEGIN (ASC0IC,2)
    // USER CODE END

    if (U0C0_PSR & 0x4000)
    {
        // USER CODE BEGIN (ASC0IC,4)
        CAN_MODATA2LL = U0C0_RBUF; //store received character in MO2
        CAN_vTransmit(2);
        // USER CODE END

        U0C0_PSCR |= 0x4000; // clear PSR_RIF
    }
    // USER CODE BEGIN (ASC0IC,15)
    // USER CODE END

} // End of function U0C0_ASC_vi0IC
```
In the ISR function ‘CAN_viSRN0(void)’ (almost at the end)

```c
_interrupt(CAN_SRN0INT) void CAN_viSRN0(void)
{
    
    while (CAN_MSID0 != 0x0020)
    {
        switch(CAN_MSID0){
            case 5: // message object 5 interrupt
                uwSRN0ObjHandler = CAN_HWOBJ[5].uwMOCTRL;
                if(uwSRN0ObjHandler & MOSTAT_RXPND) // if message object 5 receive interrupt
                    {
                        // USER CODE BEGIN (SRN0_OBJ5,1)
                        IO_vTogglePin(IO_P10_0);
                        U0C0_ASC_vSendData (CAN_MODATA5LL);
                        // USER CODE END
                    }
            
        }
    }
}
```

HOT Exercise CAN - Complete code
Edit File ‘CAN.C’
HOT Exercise CAN – Tasking VX Toolset

Build Project

1. Click on ‘Build Project CAN’
Note: It is recommended to use the latest DAS version. Download the latest version at www.infineon.com\DAS
HOT Exercise CAN - Device Access Server

1.) Checking USB connections

This gets identified only when COM port is used
• Via the USB interface on the Easykit with FTDI chip

The DAS JTAG composite device gets identified
• When miniWiggler is connected
• When USB Wiggler Box is connected
• Via the USB interface on the Easykits with FTDI chip
HOT Exercise CAN - Device Access Server

2.) Check DAS status

1. Start DAS device scanner

2. Start DAS Server Control panel

3. If DAS device scanner does not show any device, start the appropriate DAS server

Incase you are connected via the USB Wiggler box, then start „JTAG over USB Box“

Incase you are connected via the FTDI chip or mini wiggler, then start „UDAS“
3.) Starting the servers manually

4. Incase „UDAS“ server is started and XC2000 easykit is connected via on-chip FTDI or via separate miniWiggler, following status changes could be noted

5. Incase „JTAG over USB Box“ server is started and XC2000 starter kit is connected via Wiggler box, following status changes could be noted
HOT Exercise CAN
Connect XC2287M Board

- Disconnect power supply from the board
- Connect CAN nodes: Connect wires from node A to node B (connect CAN1_L to CAN2_L and CAN1_H to CAN2_H)
**HOT Exercise CAN**
**Connect XC2287M Board**

- **Internal start from Flash**
  - Connect XC2287M Board to PC
  - Modify the DIP switch settings, S102: **OFF-OFF-OFF-OFF-OFF-OFF**
  - Reset the board (press the reset button)
HOT Exercise CAN – Tasking VX Toolset

Run Debugger

1. Click on "Debug Current Project"

2. Click on ‘Resume’ and start program
With the FTDI chip an on board, USB interface can be used for UART. FTDI device will converts the USB protocol the ASC protocol. Both USB and UART can be used at the same time.

Open Device Manager and check which COM port is activated for the FTDI chip.
HOT Exercise CAN
Start HyperTerminal

1. Start->Programs->Accessories->Communications->HyperTerminal
2. Enter any name and click ‘OK’
3. Connect using: COMx (COM port activated for the FTDI chip)
4. Click ‘Configure’ to enter Port settings
5. Select 19200 baud, no Parity, 8 Data Bits and 1 Stop Bit
6. Click ‘OK’
Start typing

- Enter ASCII characters in the HyperTerminal
- The characters you enter are sent to the XC2287M, through the CAN bus and back to the Terminal Program so that you can read them on the screen
- The characters are not sent directly from the keyboard to the screen!
The yellow LED will toggle when data is receive in CAN Node1
HOT Exercise CAN Verification

- Verification 1: (stop the program)
  - Go to Tasking debugger
  - Click on ‘Suspend’
  - Go back to the terminal program and start typing again:
    ⇒ you will no longer see what you are typing.

- Verification 2: (start the program)
  - Go to Tasking debugger and start the program.
  - Go back to the terminal program and start typing again:
  - Pull out the CAN cable
    ⇒ you will no longer see what you are typing
We commit.
We innovate.
We partner.
We create value.