XC2287M HOT
Solution CAN_3
Serial Communication using the CAN with external CAN Bus and RX FIFO

Device: XC2287M-104F80
Compiler: Tasking Viper 2.4r1
Code Generator: DAvE 2.1
Serial Communication using the CAN_2 with external CAN-BUS and RX FIFO

Let’s get started now!
XC2287M HOT Exercise CAN_3
Interaction of Development Tools

- IDE
  - Compiler
  - Assembler
  - Linker
  - Locator

- DAvE

- Programming Tool

- Debugger/Emulator

- Target

- SW
- HW

```c
int main()
{
    char a;
    long b;
    ...
```
HOT Exercise CAN_3

Objective

- Use a terminal program to send ASCII characters via the serial port to ASC0 and forward to CAN Node 0
- Transmit characters via external CAN-Bus to Node 1
- Node 1 uses a FIFO and sends back the received characters via ASC when the FIFO buffer is full. Forward from CAN Node 1 back to ASC0
- Transmit the received ASCII characters back to the PC
HOT Exercise CAN_3
Block Diagram

- **PC**
  - TxD
  - RxD
  - RS 232 via USB
    - 19200 Baud
    - 8 n 1
    - RX interrupt

- **USIC 0 – ASC 0**
  - Receive Buffer
  - Transmit Buffer
  - U0C0 Receive IRQ
    - Priority 3

- **CAN Node 0**
  - CAN MSG OBJ 10
    - ID=0x11
      - CAN_L
      - CAN_H

- **CAN Node 1**
  - CAN MSG FIFO
    - OBJ 0, OBJ 1, OBJ2
    - ID=0x11
  - FIFO Overflow IRQ
    - Priority 5

- **Transfer by Interrupt handler**

- **Ext. CAN Bus**
  - RX: P2.6
  - TX: P2.5
  - RX: P2.4
  - TX: P2.2

- **PC** connections:
  - RXD: P7.4
  - TXD: P7.3

- **CAN Node 0** connections:
  - RXD
  - TxD

- **CAN Node 1** connections:
  - RXD
  - TxD
HOT Exercise CAN_3 - DAvE Configurations
Start DAvE

- **Start DAvE**
  - Click on the DAvE

- **Create a new project** (Startup Dialog pop up automatically)
  - Click on ‘Create a new project’
  - Select microcontroller: ‘XC2287M’
HOT Exercise CAN_3
Start DAvE (cont.)

1. Open DAvE
2. Select XC2287M
3. Click Create
HOT Exercise CAN_3- DAvE Configurations
Project Settings

- Project Settings
- Close the window
Save your DAvE project

- Path:
  C:\IFX_HOT\XC2287M\Examples\CAN_3

- Project name:
  CAN_3\CAN_3.dav
Save your DAvE Project File
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_3 - DAvE Configurations ASC settings

- **XC2287M**
  - **USIC0:**
    - Click on the
HOT Exercise CAN_3 - DAvE Configurations
ASC settings

- **XC2287M**
  - **USIC0 :**
    - Select ASC for U0C0 protocol
    - Click to exit

![Universal Serial Interface Channel (USIC0)](image)
**XC2287M**

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

ASC General:
- Enable module clock
- Select P7.3 for TxD and P7.4 for RxD
## HOT Exercise CAN_3 - DAvE Configurations
### ASC settings

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

---

### ASC Control Settings

<table>
<thead>
<tr>
<th>Interrupt Type</th>
<th>U0C0_OIC_INT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate Receive Interrupt</td>
<td></td>
</tr>
<tr>
<td>Transmit Buffer Interrupt</td>
<td>U0C0_OIC_INT</td>
</tr>
<tr>
<td>Transmit Shift Interrupt</td>
<td>U0C0_OIC_INT</td>
</tr>
<tr>
<td>Data Lost Interrupt</td>
<td>U0C0_OIC_INT</td>
</tr>
</tbody>
</table>

- **Note:**
  - Enable the **Receive interrupt** to receive ASC interrupts.

---

![ASC Control Settings](image-url)
Configure CH0 ASC

Interrupts:

- Drag ‘U0C0_0IC INT’ and drop it to Interrupt Level 3, Group 0
Configure CH0 ASC

- Functions:
  - Include ‘U0C0_ASC_vInit’
  - Include ‘U0C0_ASC_vSendData’
HOT Exercise CAN_3 - DAvE Configurations
Port settings

- **XC2287M**
  - **Port:**
    - Click on the
Parallel Ports

- Ports:
  - Configure Port 10
HOT Exercise CAN_3 - DAvE Configurations
Port settings

- **Configure Port 10**

- **Port 10:**
  - Use P10.0 as general IO
  - Set Direction to Out
  - Close the window
HOT Exercise CAN_3 - DAvE Configurations
Port settings

**Parallel Ports**

- **Functions:**
  - Include ‘IO_vInit’
  - Include ‘IO_vTogglePin’
HOT Exercise CAN_3 - DAvE Configurations
MultiCAN settings

- **CAN Settings**
  - Baud Rate = 500 kBaud
  - CAN 0:RX: P2.6, TX P2.5
  - CAN 1; RX: P2.4, TX P2.2
  - Message Object 10: CAN node 0, transmit, 1 byte, 11-bit ID = 0x11
  - Message Object 0: CAN node 1, receive FIFO base object, 1 byte, 11-bit ID = 0x11 FIFO Overflow-Interrupt, level 5
  - Message Object 1: CAN node 1, receive FIFO slave object,
  - Message Object 2: CAN node 1, receive FIFO slave object,
HOT Exercise CAN_3 - DAvE Configurations
MultiCAN settings

- **XC2287M**
  - **MultiCAN:**
    - Click on the MultiCAN node.
HOT Exercise CAN_3 - DAvE Configurations
MultiCAN settings

- **Configure MultiCAN**
  - **General:**
    - Enable module
Configure MultiCAN

- General:
  - Select Node 0
**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_3 - 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

![Configuration Interface]

1. Select Baud Rate
2. Set Time Segment (TSeg1) before sample point
3. Set Time Segment (TSeg2) after sample point
4. Enter Required baud rate
5. Calculate Real baud rate
Configure MultiCAN

- General:
  - Select Node 1
HOT Exercise CAN_3 - DAvE Configurations

MultiCAN settings

- **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 'MO10' and drop it to List 1 (Node 0)
  - Drag 'MO0', 'MO1', 'MO2' and drop it to List 2 (Node 1)
Configure MultiCAN

- M0s:
  - Select M00
Configure CAN Message Object (M00)

- **Object:**
  - Enable M00, Select Receive data frames
  - Identifier: 0x011, Data Length: 1 data bytes
Configure CAN Message Object (M00)

- **FIFO/Gateway:**
  - Enable MO0, Select Receive FIFO base object
  - FIFO Pointers: Bottom MO0, Top MO2, Object select MO0

![Diagram of CAN Message Object configuration](image-url)
Configure CAN Message Object (M00)

- **Interrupt:**
  - Enable FIFO overflow interrupt
  - Use CAN SRN 0
HOT Exercise CAN_3 - DAvE Configurations
MultiCAN settings

- **Configure MultiCAN**

- **M0s:**
  - Select MO10
HOT Exercise CAN_3 - DAvE Configurations MultiCAN settings

Configure CAN Message Object (MO10)

- Object:
  - Enable MO10, Select Transmit data frames
  - Identifier: 0x011, Data Length: 1 data byte
Configure MultiCAN

- **Interrupts**:
  - Drag ‘CAN INT 0’ and drop it to Interrupt Level 4, Group 0
HOT Exercise CAN_3 - DAvE Configurations

MultiCAN settings

- **Configure MultiCAN**
  - **Functions:**
    - Include ‘CAN_vInit’
    - Include ‘CAN_vTransmit’, ‘CAN_ubWriteFIFO, CAN_ubReadFIFO’

![Diagram showing MultiCAN configuration options](image_url)
Save your DAvE Project File

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

- Filename entered previously:
  
  “c:\IFX_HOT\XC2287M\Examples\CAN_3\CAN_3.dav”
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
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...)
HOT Exercise CAN_3 – Tasking VX Toolset

- Import DAVE Project
  - Click on File -> Import
  - Select Tasking VX-toolset for C166...
  - Click ‘OK’
HOT Exercise CAN_3 – Tasking VX Toolset

- Import DAVE Project
  - Click `Infineon DAvE C166 Project`
  - Click ‘Next’
HOT Exercise CAN_3 – Tasking VX Toolset

- Import DAveE Project

  - Add Dave Project ‘CAN_3’

  - Click ‘Finish’
Configure Target Board

1. Select the project in the navigator
2. Select ‘Project/Target Board Configuration’
3. Select ‘Infineon XC2000/XE166 Easykit Board’
4. Choose `XC2287M-104F´
5. Click `Finish´
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_MODATA10LL = U0C0_RBUF; //store received character in MO10
        CAN_vTransmit(10);
        // 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)
{
    // USER CODE BEGIN (SRN0,2)
    IO_vTogglePin(IO_P10_0);
    U0C0_ASC_vSendData (CAN_MODATA0LL);
    U0C0_ASC_vSendData (CAN_MODATA1LL);
    U0C0_ASC_vSendData (CAN_MODATA2LL);
    // USER CODE END

    // End of function CAN_viSRN0
} // End of function CAN_viSRN0
```
Click on ‘Build Project CAN’
HOT Exercise CAN_3 - Device Access Server

Check for the latest DAS version

Note: It is recommended to use the latest DAS version. Download the latest version at www.infineon.com\DAS
HOT Exercise CAN_3 - 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_3 - 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

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

In case you are connected via the FTDI chip or mini wiggler, then start „UDAS“
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_3
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_3
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_3 – 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 Manger and check which COM port is activated for the FTDI chip.
HOT Exercise CAN_3
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

- At each third character you are typing the Interrupt routine sends back the received characters through the CAN bus 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 three characters are received in CAN Node1's FIFO.
HOT Exercise CAN_3
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
Tasking Viper
"Profile Storage Space Exceeded"

■ Edit config.ini

- C: \ Program Files \ TASKING \ C166-VX v2.4r1 \ eclipse \ configuration \...

```ini
# The default configuration location for this run of the platform. The configuration
# determines what plug-ins will run as well as various other system settings.
# osgi.configuration.area = @user.home/.eclipse_c166_v2.4r/config
osgi.configuration.area = C:/UserData/_login-name_/eclipse_c166_v2.4r/config

# The default location of the user area. The user area contains data (e.g., preferences)
# specific to the OS user and independent of any Eclipse install, configuration or instance.
# osgi.user.area = @user.home/.eclipse_c166_v2.4r
osgi.user.area = C:/UserData/_login-name_/eclipse_c166_v2.4r

# The default workspace location
# osgi.instance.area.default = @user.home/workspace_c166_v2.4r
osgi.instance.area.default = C:/UserData/_login-name_/workspace_c166_v2.4r

# TASKING plugins require at least Java runtime environment v1.5
osgi.requiredJavaVersion = 1.5.0

# The build identifier
eclipse.buildId=I20070625-1500

# End of file marker - must be here
eof=eof
```

04.08.2010
Tasking Viper
"Profile Storage Space Exceeded"

- Delete old directories in profile space
  - C:\Documents and Settings\_login-name_\.eclipse_c166_v2.4r

- Rescan Profile Storage (double click; OK)
We commit.
We innovate.
We partner.
We create value.