XC2287M HOT

Solution CAN_2

Serial Communication using the CAN with external CAN BUS

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

Let’s get started now!
XC2287M HOT Exercise CAN_2
Interaction of Development Tools

IDE
- Compiler
- Assembler
- Linker
- Locator

DAvE

Programming Tool

Target

IDE

Debugger
int main(){
  char a;
  long b;
  ...
HOT Exercise CAN_2
Objective

- Use a terminal program to send ASCII characters via the serial port
- Receive these ASCII characters using ASC0
- Forward to CAN Node 0 and transmit to CAN Node 1 via external CAN-Bus
- Forward from CAN Node 1 back to ASC0
- Transmit the received ASCII characters back to the PC
HOT Exercise CAN_2
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 Node 1

CAN_L CAN_H

CAN_L CAN_H

CAN_MSG OBJ 2

CAN_MSG OBJ 5

MO5 Receive IRQ
Priority 5

ID=0x11

ID=0x11

Ext. CAN Bus

RX: P2.6
TX: P2.5

RXD: P7.4
TXD: P7.3

RXD: P7.4
TXD: P7.3

RXD: P7.4
TXD: P7.3

RXD: P7.4
TXD: P7.3
HOT Exercise CAN_2
Flow Chart 1/2

- Init CAN Module
  - Module enable
  - Module Clock
  - 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_2
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_2 - 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’ or select File -> New
  - Select microcontroller: ‘XC2287M’
HOT Exercise CAN _2
Start DAvE (cont.)
HOT Exercise CAN_2- DAvE Configurations

- Project Settings
- Close the window

![Project Settings Window]

- Controller Type:
  - Type: XC2207M-1C4F30
  - ROM Size: 832 KByte
  - Max. CPU Clock: 80 MHz

- Main Source File:
  - File name: MAIN.C

- Main Header File:
  - File name: MAIN.H

- Compiler Settings:
  - Compiler: Tasking Viper
  - Memory Model: NEAR
HOT Exercise CAN_2 - DAvE Configurations
Save DAvE Project

- Save your DAvE project
  - Path: C:\IFX_HOT\XC2287M\Examples\CAN_2
  - Project name: CAN_2\CAN_2.dav
Save your DAvE Project File

1. Open the DAvE project window.
2. Navigate to the 'Save Project As' option.
3. Enter the file name in the 'Dateiname' field.
4. Select the file type as 'DAvE project file (*.dav)'.
5. Click on 'Speichern' to save the project.
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_2 - DAvE Configurations
ASC settings

- **XC2287M**

  - **USIC0:**
    - Click on the image of USIC0.
**HOT Exercise CAN_2 - DAvE Configurations**

**ASC settings**

- **XC2287M**
  - **USIC0 :**
    - Select ASC for U0C0 protocol
    - Click to exit
HOT Exercise CAN_2 - DAvE Configurations
ASC settings

- **XC2287M**
  - **USIC0, CH0 :**
    - Click on the
Configure CH0 ASC

- ASC General:
  - Enable module clock
  - Select P7.3 for TxD and P7.4 for RxD

![Image of configuration settings]

1. Enable module kernel clock and module functionality
2. Pin Selection:
   - Transmit (TxD): P7.3
   - Receive (RxD): P7.4
Configure CH0 ASC

- Control:
  - Enable Receive Interrupt

![Diagram showing configuration settings]

1. Select Receive Interrupt
2. Set Interrupt to U0C0_OIC_INT
HOT Exercise CAN_2 - DAvE Configurations

ASC settings

- Configure CH0 ASC

- **Interrupts:**
  - Drag ‘U0C0_0IC INT’ and drop it to Interrupt Level 3, Group 0
HOT Exercise CAN_2 - DAveE Configurations

Configure CH0 ASC

- **Functions:**
  - Include ‘U0C0_ASC_vInit’
  - Include ‘U0C0_ASC_vSendData’
HOT Exercise CAN_2 - DAvE Configurations
Port settings

- XC2287M
  - Port:
    - Click on the
Parallel Ports

- Ports:
  - Configure Port 10
Configure Port 10

- Use P10.0 as general IO
- Set Direction to Out
- Close the window
HOT Exercise CAN_2 - DAvE Configurations
Port settings

- **Parallel Ports**
  - **Functions:**
    - Include ‘IO_vInit’
    - Include ‘IO_vTogglePin’
HOT Exercise CAN_2 - DAvE Configurations
MultiCAN settings

- **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
  - CAN 0: RX: P2.6, TX P2.5
  - CAN 1: RX: P2.4, TX P2.2
HOT Exercise CAN_2 - DAvE Configurations

MultiCAN settings

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

- **General:**
  - Enable module
HOT Exercise CAN_2 - DAvE Configurations
MultiCAN settings

- **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_2 - 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

### Baud Rate

<table>
<thead>
<tr>
<th><strong>Required baud rate [kbaud]</strong></th>
<th><strong>500,000</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time Quantum tq [ns]</strong></td>
<td><strong>175,000</strong></td>
</tr>
<tr>
<td><strong>Real baud rate [kbaud]</strong></td>
<td><strong>513,491</strong></td>
</tr>
<tr>
<td><strong>Baud rate prescaler</strong></td>
<td><strong>6</strong></td>
</tr>
</tbody>
</table>

![Configuration settings screenshot]

1. **Configure CAN Node 0**
2. **Baud Rate**
3. **TSeg1**
4. **TSeg2**
5. **Frame Counter**
Configure MultiCAN

- General:
  - Select Node 1
**HOT Exercise CAN_2 - 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 ‘M02’ and drop it to List 1 (Node 0)
  - Drag ‘M05’ and drop it to List 2 (Node 1)
HOT Exercise CAN_2 - DAvE Configurations
MultiCAN settings

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

- Enable M02, Select Transmit data frames
- Identifier: 0x011, Data Length: 1 data byte
HOT Exercise CAN_2 - DAvE Configurations
MultiCAN settings

- **Configure MultiCAN**
  - **M0s:**
    - Select M05
HOT Exercise CAN_2 - DAvE Configurations
MultiCAN settings

- Configure CAN Message Object (M05)

- **Object:**
  - 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_2\CAN_2.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_2 – Tasking VX Toolset

- Import DAVE Project
  - Click on File -> Import
  - Select Tasking VX-toolset for C166...
  - Click ‘OK’
HOT Exercise CAN_2 – Tasking VX Toolset

- Import DAVE Project
  - Click `Infineon DAvE C166 Project´
  - Click ‘Next’
HOT Exercise CAN_2 – Tasking VX Toolset

- Import DAve Project
  - Add Dave Project ‘CAN_2’
  - Click ‘Finish’
HOT Exercise CAN_2 – Tasking VX Toolset

- 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_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
                }
            }
        }
    }
}
```
Click on ‘Build Project CAN’
HOT Exercise CAN_2 - 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
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_2 - 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“
HOT Exercise CAN_2 - Device Access Server

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
- 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_2
Connect XC2287M Board

- Internal start from Flash
  - Connect XC2287M Board to PC
  - Modify the DIP switch settings, S102: OFF-OFF-OFF-OFF-OFF
  - Reset the board (press the reset button)
HOT Exercise CAN_2 – 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_2
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’
HOT Exercise CAN_2
Running the program

- **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!
HOT Exercise CAN_2
See Result

- The yellow LED will toggle when data is received in CAN Node1

LED blinking
HOT Exercise CAN_2
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
CAN Bonus Exercise 1 (Unidirectional)

PC 1

RS 232 via USB

PC 2

RS 232 via USB

CAN Node 0 EK-1

CAN Node 0 EK-2

CAN L CAN_H

CAN L CAN_H

CAN L CAN_H

CAN L CAN_H

RS 232

Rx D

Tx D

Rx D

Tx D

TxD

RxD

TxD

RxD

CAN Node 1 EK-1

CAN Node 1 EK-2
CAN Bonus Exercise 2 (Bidirectional)

PC 1
RxD

PC 2
TxD

RS 232
via USB

CAN Node 0
EK-1

TxD

RxD

USIC 0 – ASC 0

Receive Buffer

Transmit Buffer

U0C0 Receive IRQ
Priority 3

CAN MSG OBJ 2
ID=0x11

CAN_L
CAN_H

Ext. CAN Bus

CAN Node 1
EK-1

TxD

RxD

CAN MSG OBJ 5
ID=0x12

MO5 Receive IRQ
Priority 5

CAN_L
CAN_H

CAN Node 0
EK-2

TxD

RxD

USIC 0 – ASC 0

Receive Buffer

Transmit Buffer

U0C0 Receive IRQ
Priority 3

CAN MSG OBJ 2
ID=0x12

CAN_L
CAN_H

CAN Node 1
EK-2

TxD

RxD

USIC 0 – ASC 0

Receive Buffer

Transmit Buffer

U0C0 Receive IRQ
Priority 3

CAN MSG OBJ 5
ID=0x11

MO5 Receive IRQ
Priority 5

CAN_L
CAN_H

RS 232
via USB
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
```
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.