

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

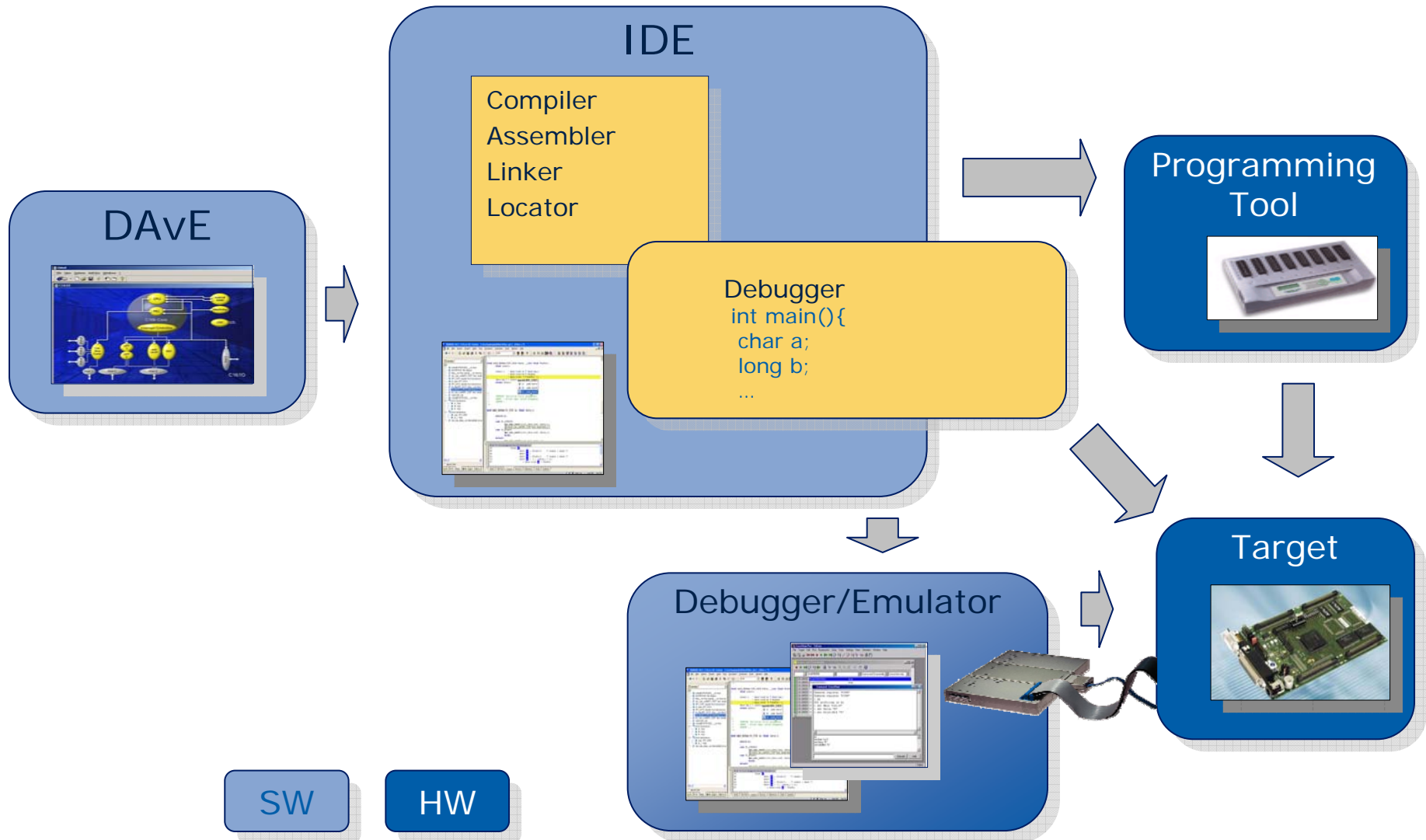
Code Generator: DAvE 2.1



Never stop thinking

XC2287M HOT Exercise CAN_3

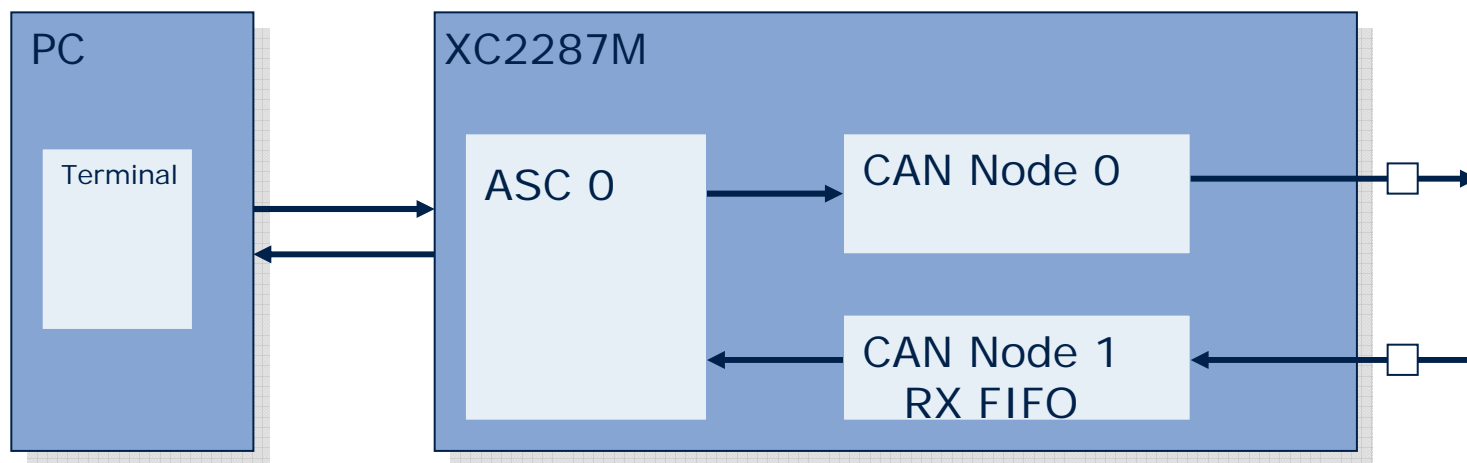
Interaction of Development Tools



HOT Exercise CAN_3

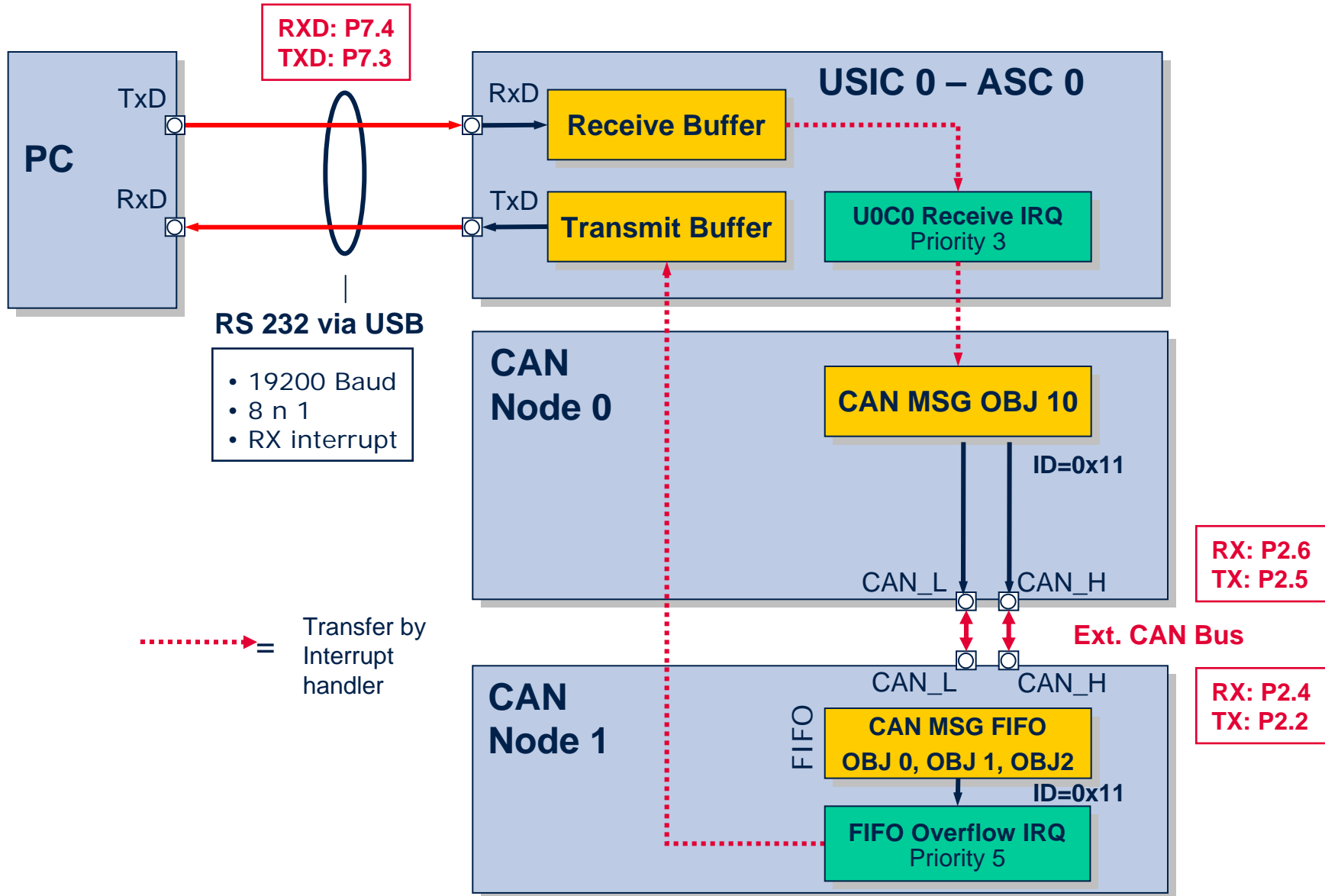
Objective

- 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



■ Start DAVe

- Click on the



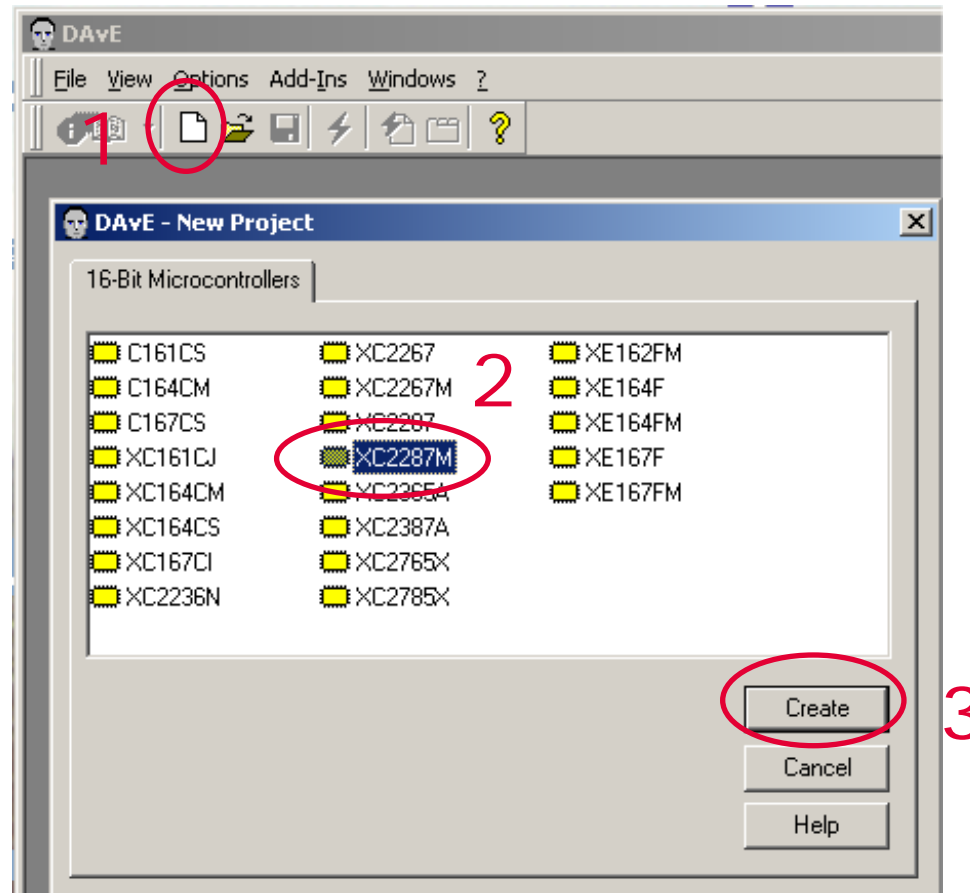
■ Create a new project (Startup Dialog pop up automatically)

- Click on 'Create a new project'
- Select microcontroller: 'XC2287M'



HOT Exercise CAN_3

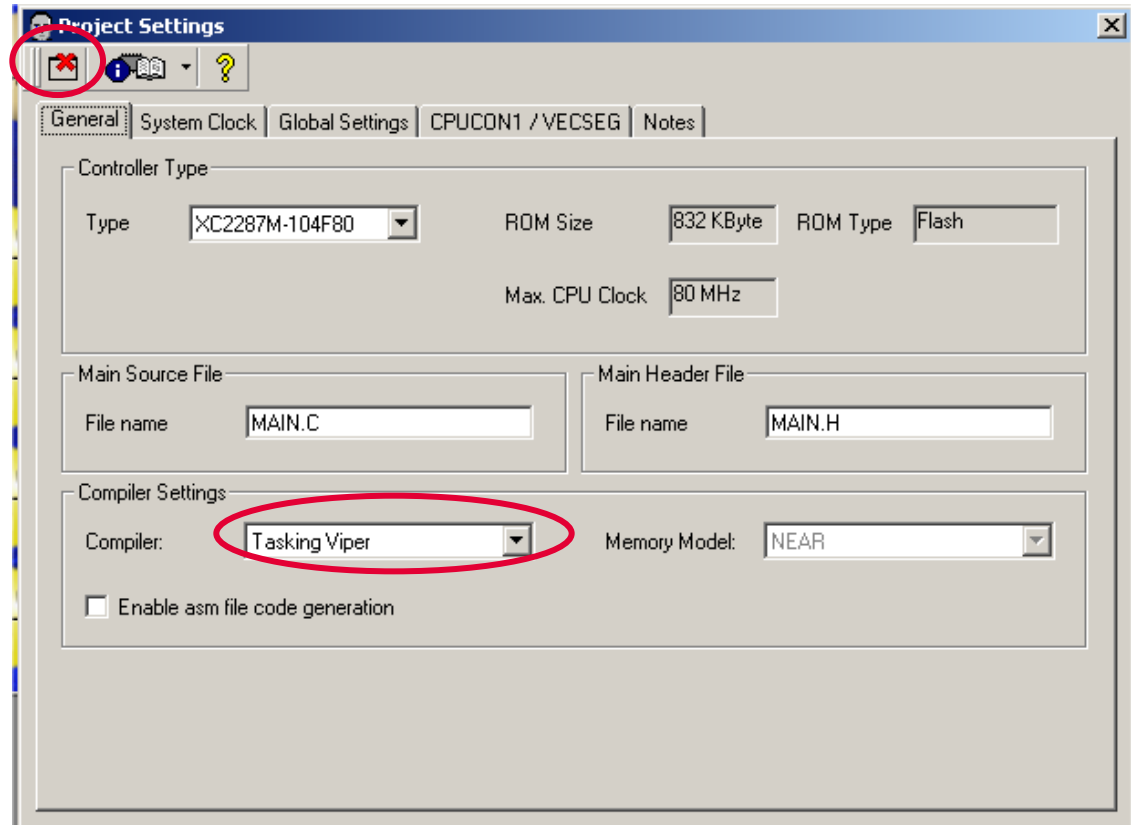
Start DAVe (cont.)



HOT Exercise CAN_3- DAVe Configurations

Project Settings

- Project Settings
- Close the window



HOT Exercise CAN_3 - DAVe Configurations

Save DAVe Project

■ Save your DAVe project



□ Path:

C:\IFX_HOT\XC2287M\Examples\CAN_3

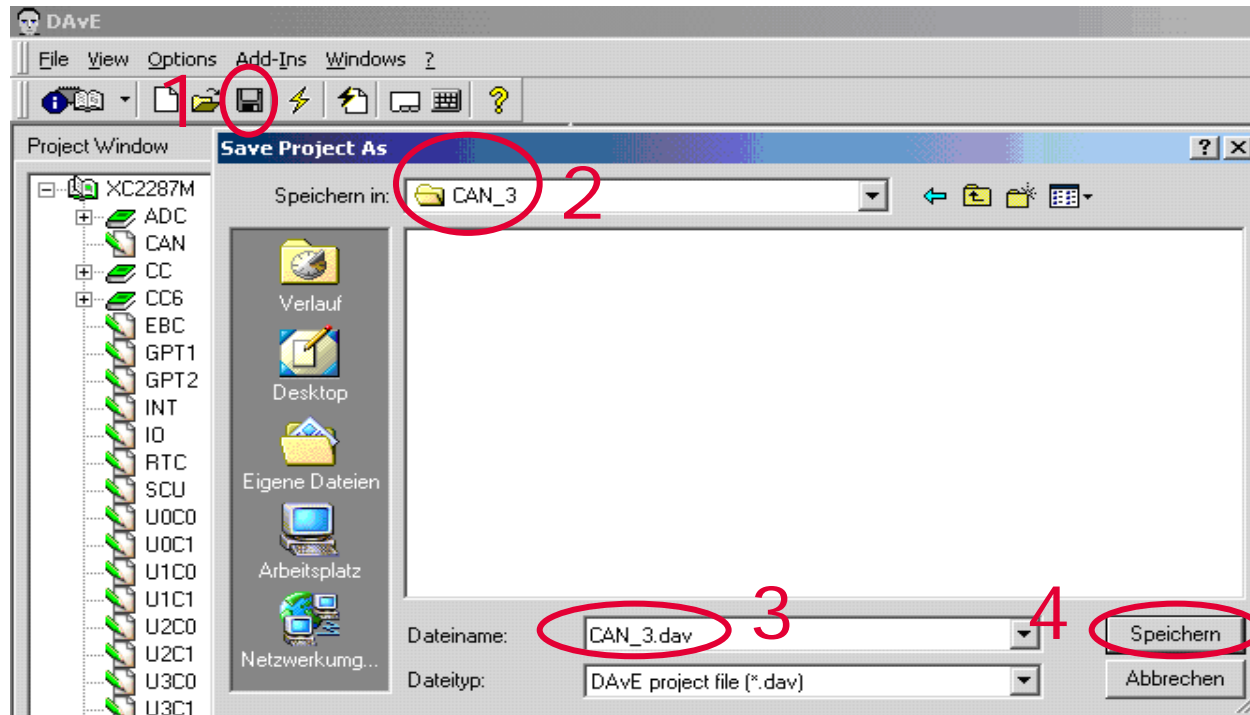
□ Project name:

CAN_3\CAN_3.dav

HOT Exercise CAN_3 - DAVe Configurations

Save DAVe Project

- 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

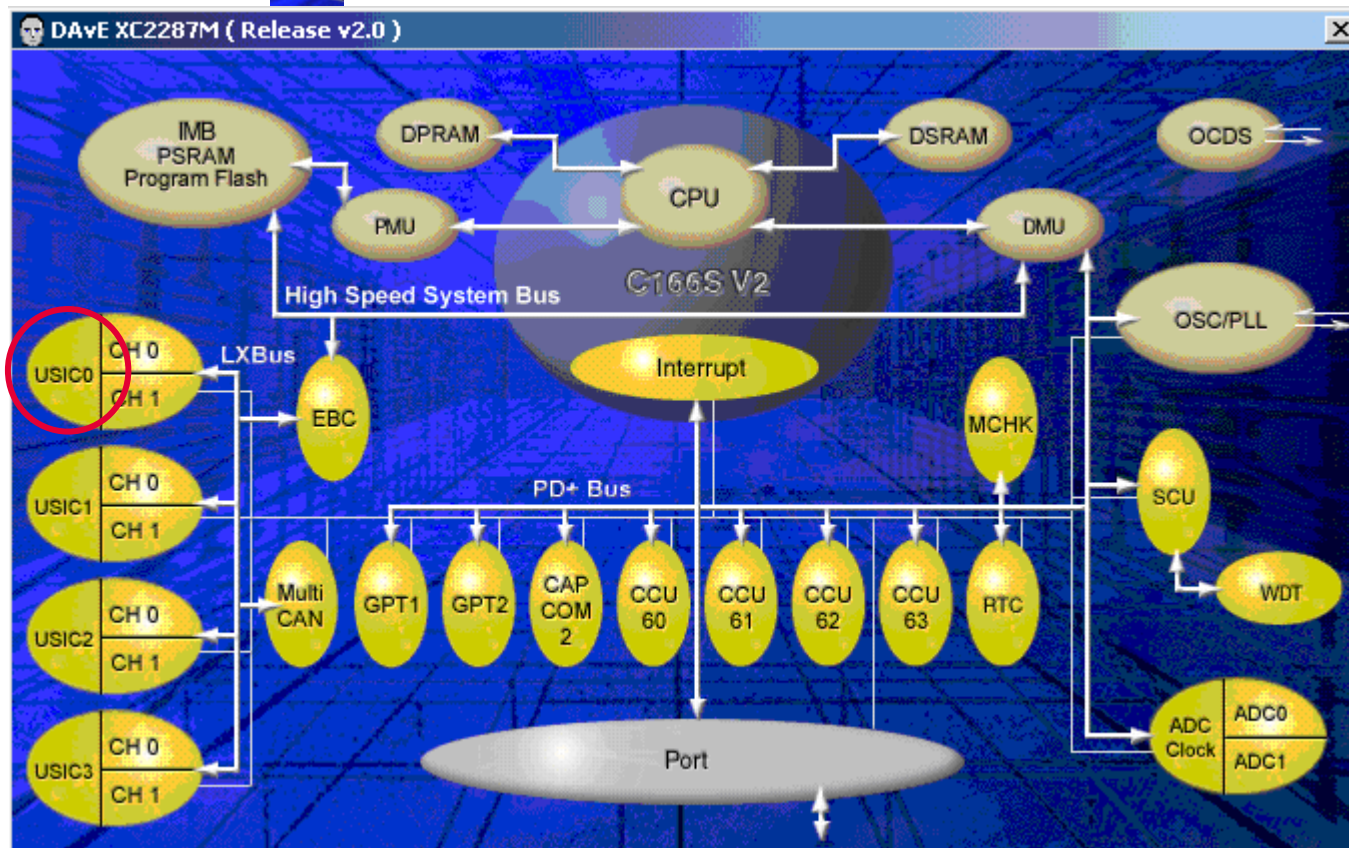
HOT Exercise CAN_3 - DAVe Configurations

ASC settings

■ XC2287M

□ USIC0 :

→ Click on the




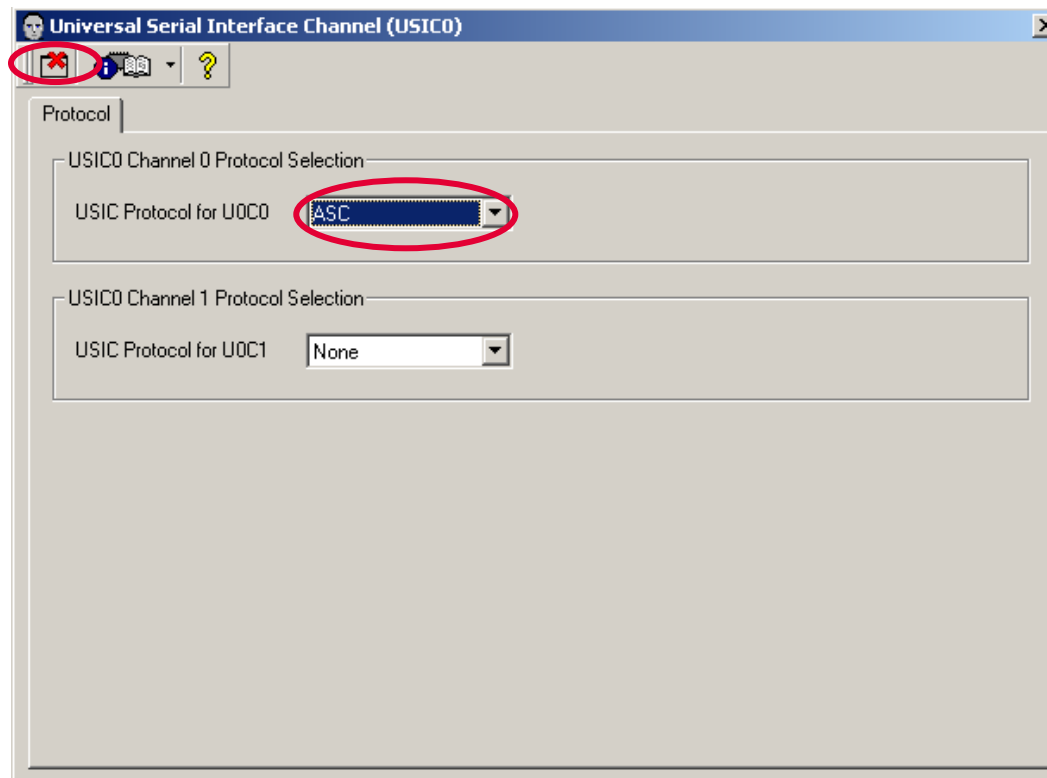
HOT Exercise CAN_3 - DAVe Configurations

ASC settings

■ XC2287M

□ USIC0 :

- Select ASC for U0C0 protocol
- Click  to exit



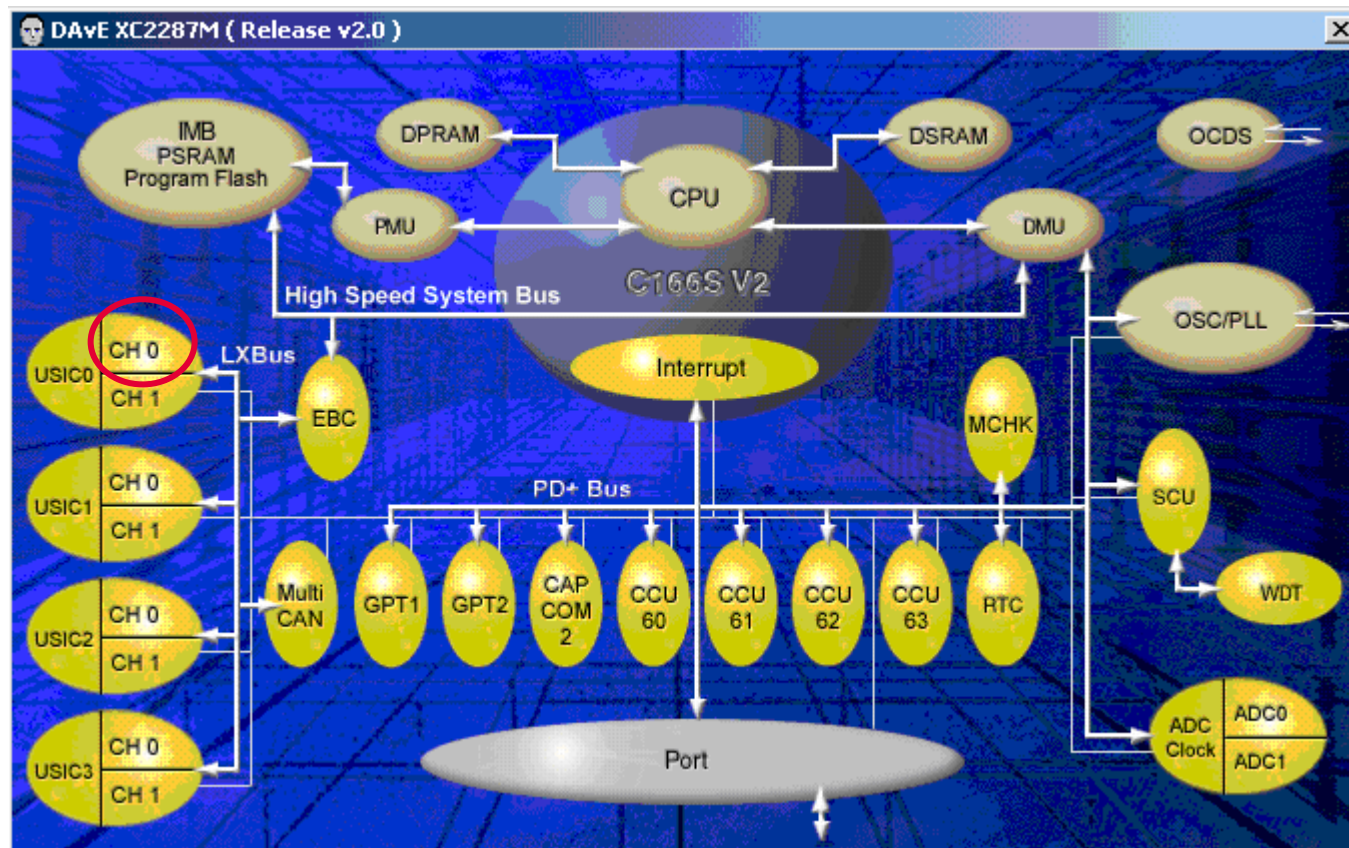
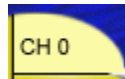
HOT Exercise CAN_3 - DAVe Configurations

ASC settings

■ XC2287M

□ USIC0, CH0 :

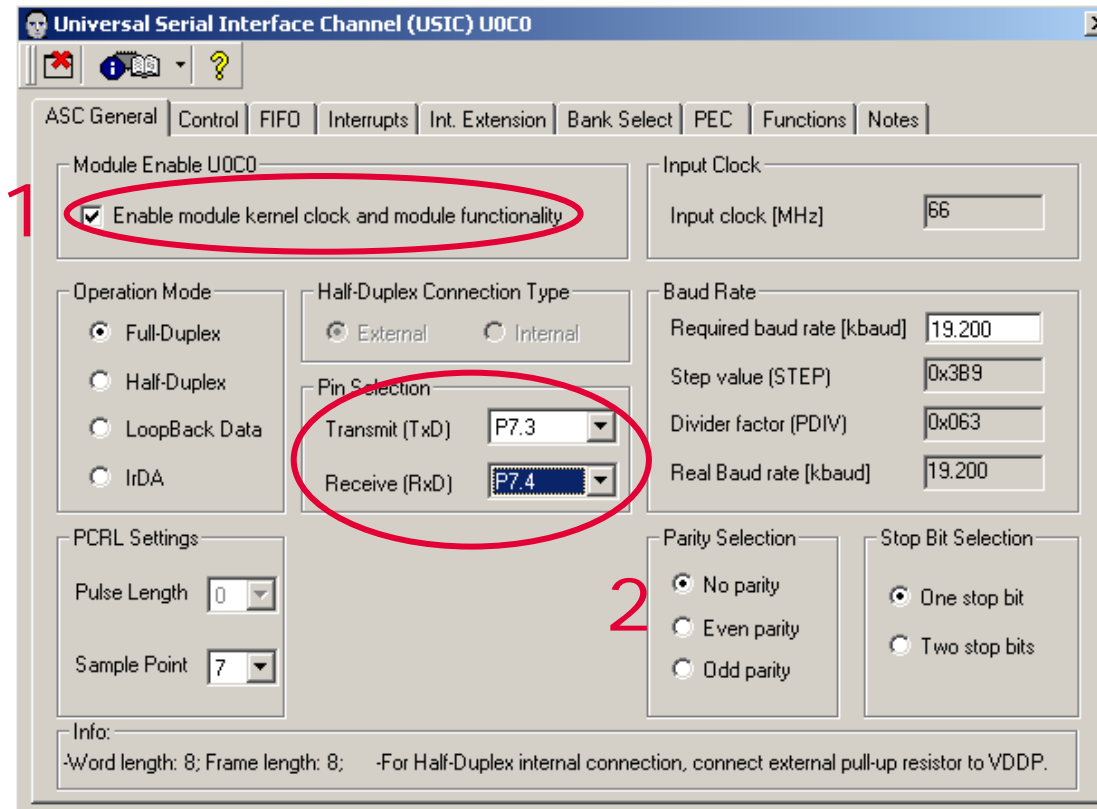
→ Click on the



■ Configure CH0 ASC

□ ASC General:

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



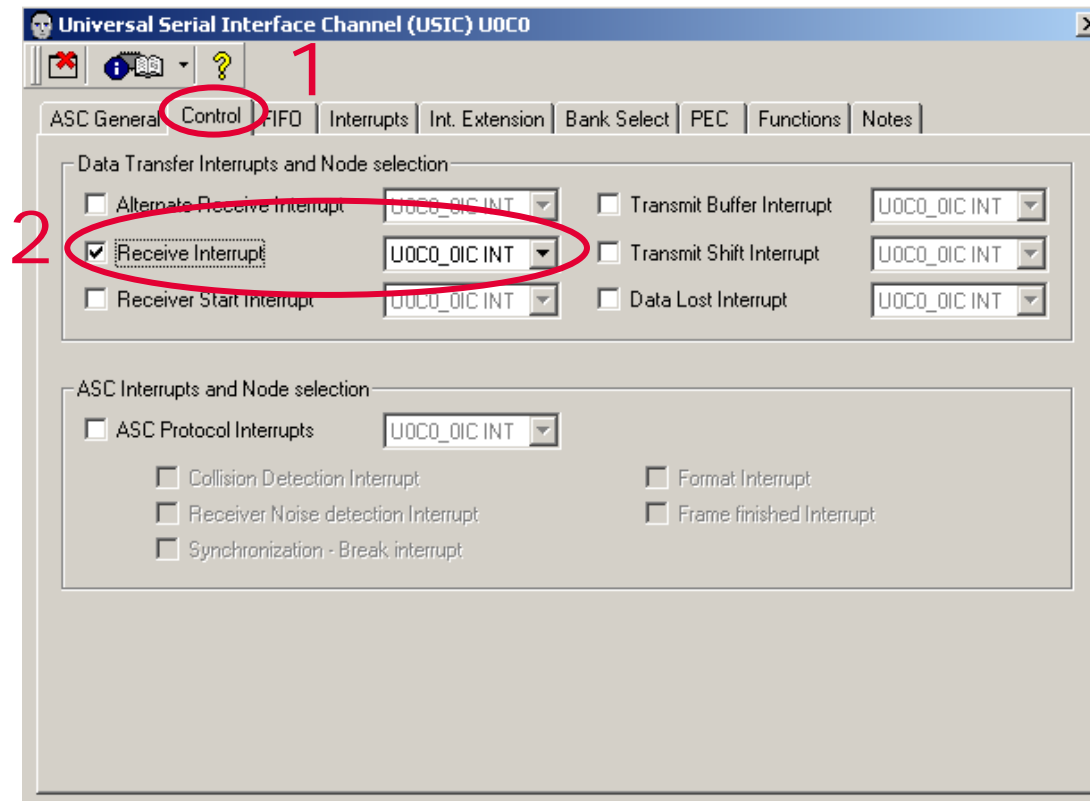
HOT Exercise CAN_3 - DAVe Configurations

ASC settings

■ Configure CH0 ASC

□ Control:

- Enable Receive Interrupt



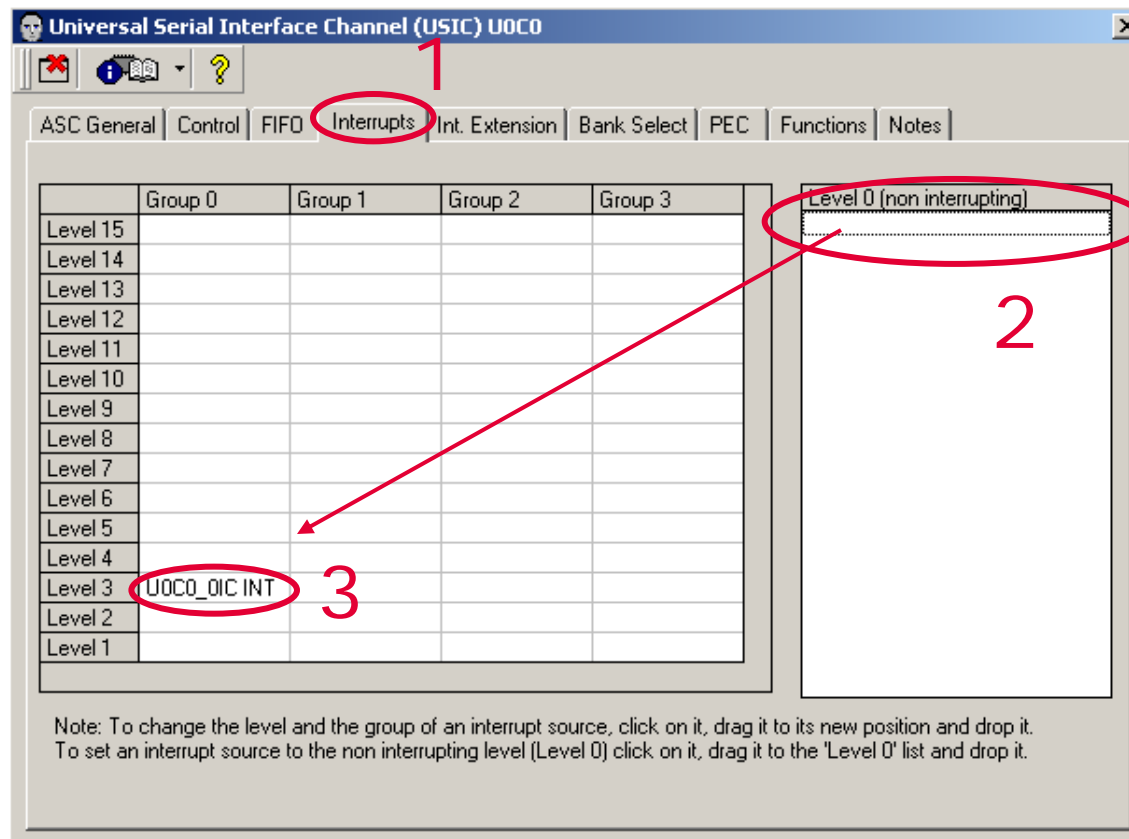
HOT Exercise CAN_3 - DAVe Configurations

ASC settings

■ Configure CH0 ASC

□ Interrupts:

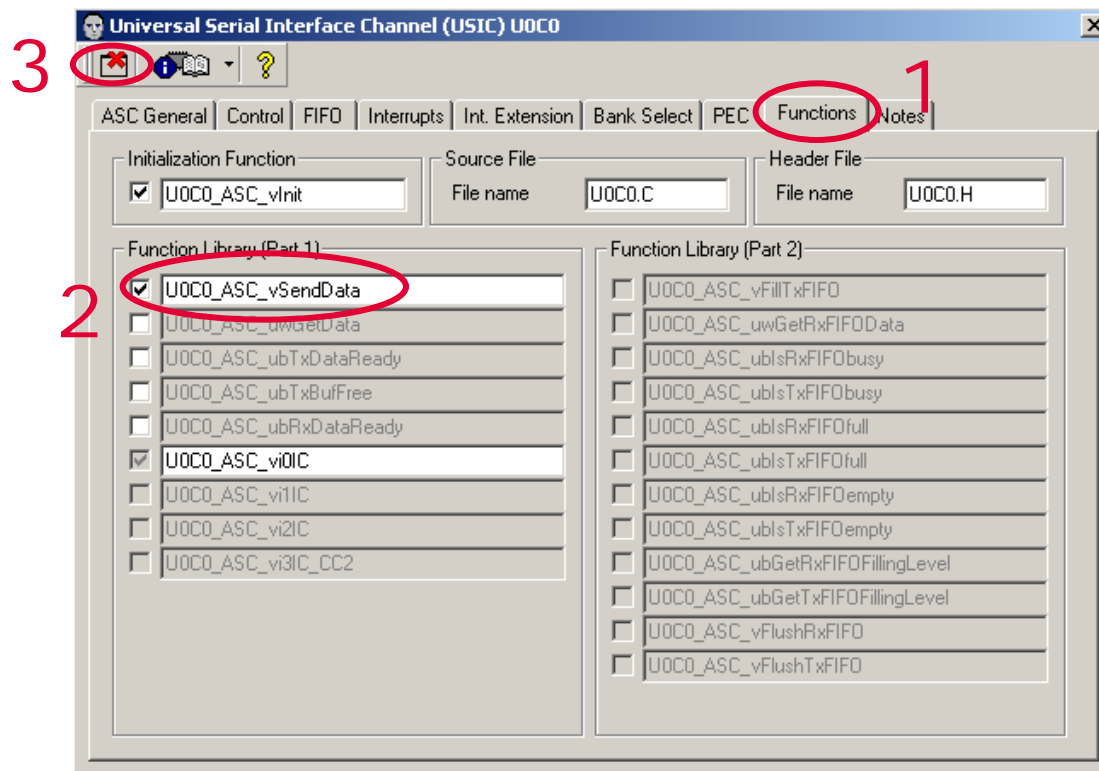
- Drag 'UOCO_OIC INT' and drop it to Interrupt Level 3, Group 0



■ Configure CH0 ASC

□ Functions:

- Include 'U0C0_ASC_vInit'
- Include 'U0C0_ASC_vSendData'



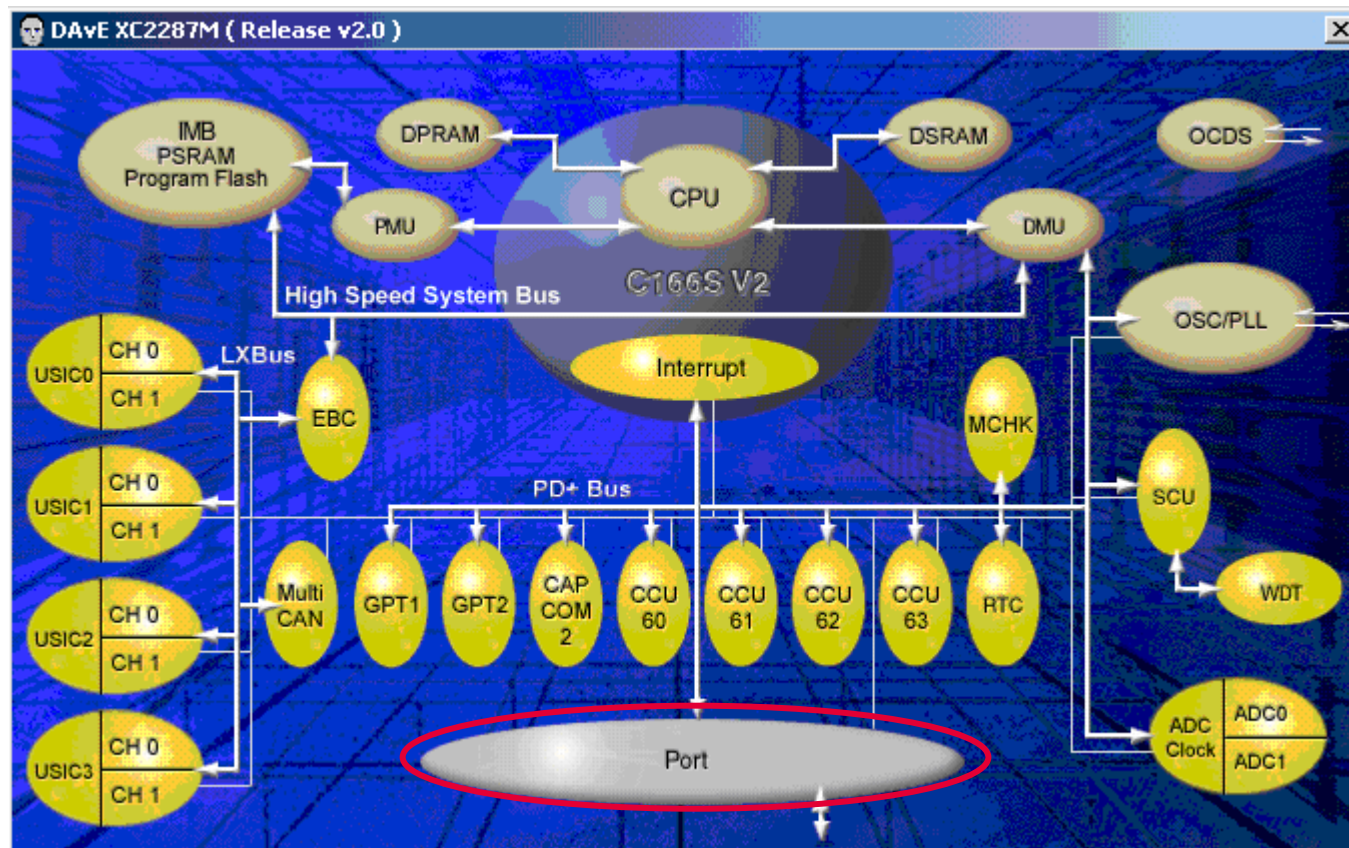
HOT Exercise CAN_3 - DAVe Configurations

Port settings

■ XC2287M

□ Port:

→ Click on the



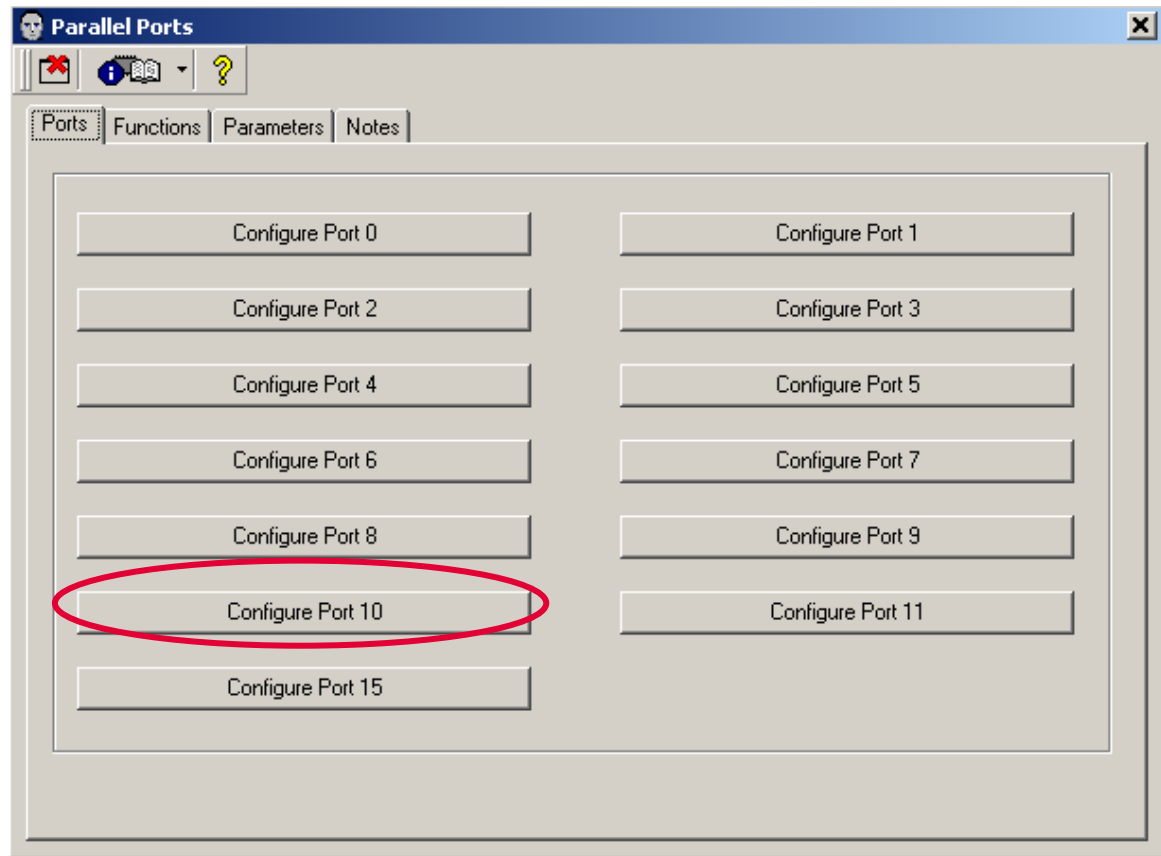
HOT Exercise CAN_3 - DAVe Configurations

Port settings

■ Parallel Ports

□ Ports:

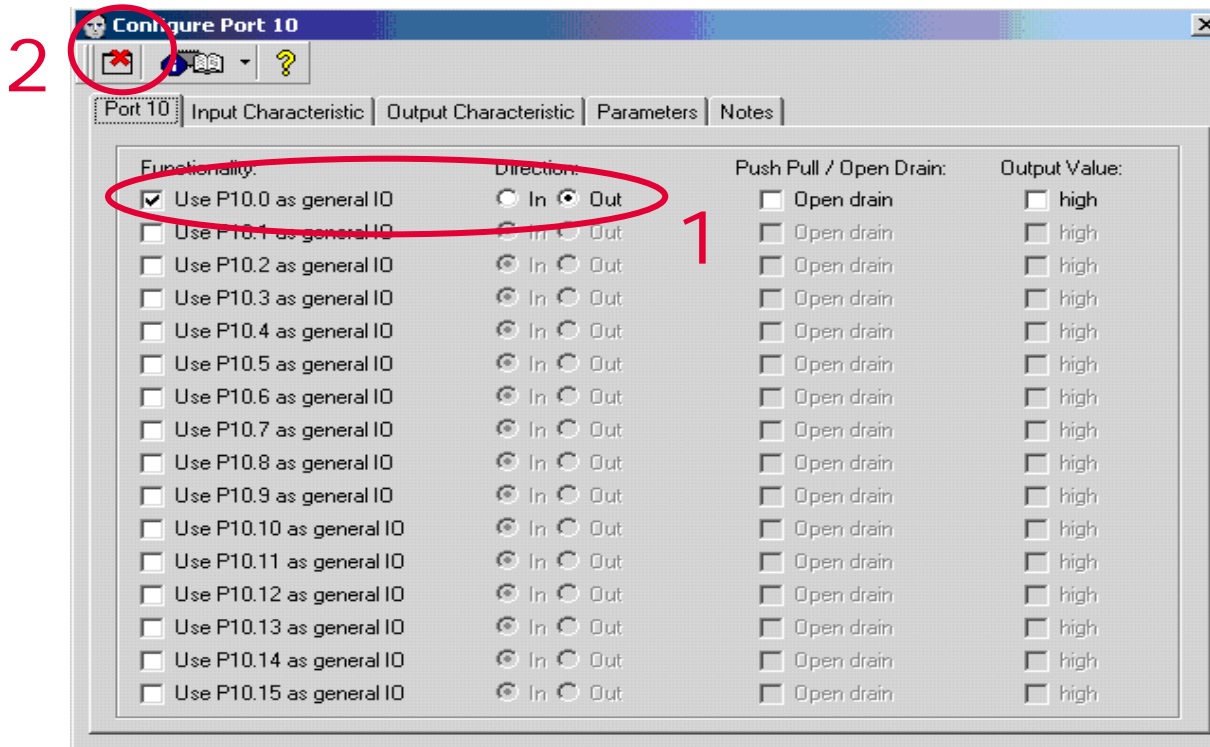
- Configure Port 10



■ Configure Port 10

□ Port 10:

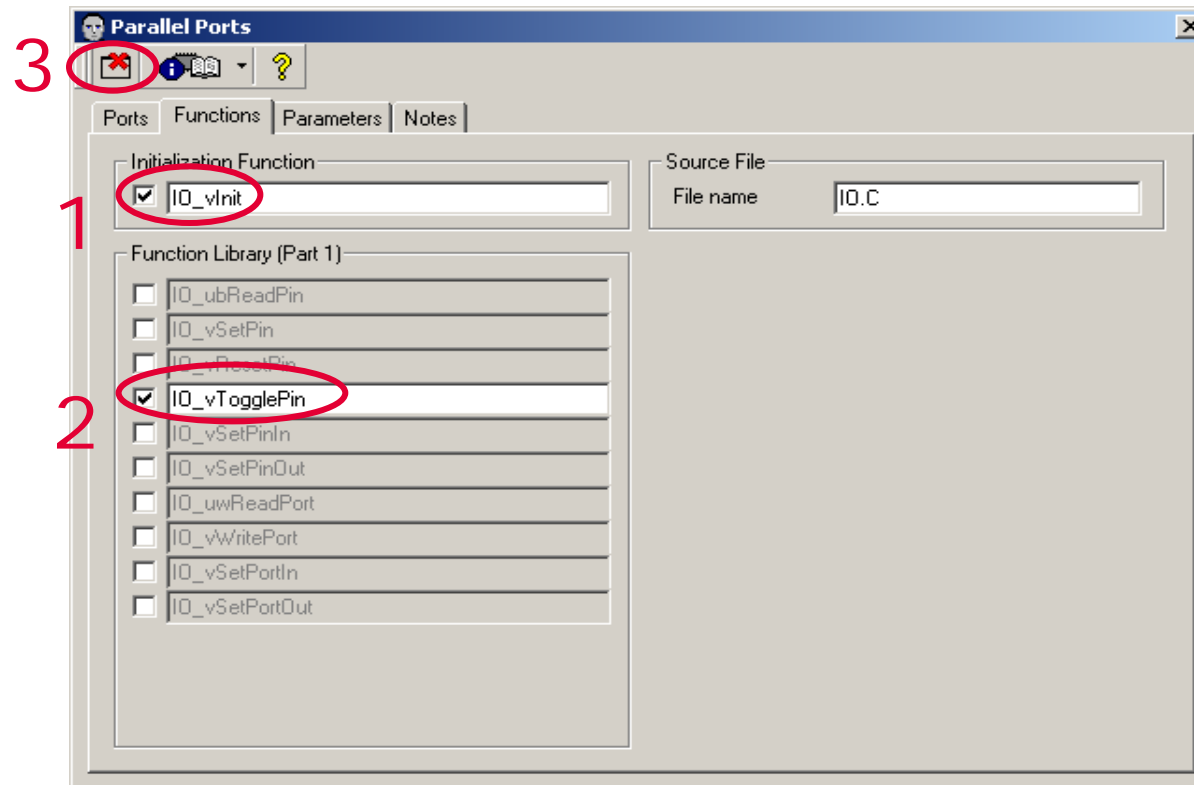
- Use P10.0 as general IO
- Set Direction to Out
- Close the window



■ Parallel Ports

□ Functions:

- Include 'IO_vInit'
- Include 'IO_vTogglePin'



■ 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,

□ MultiCAN :



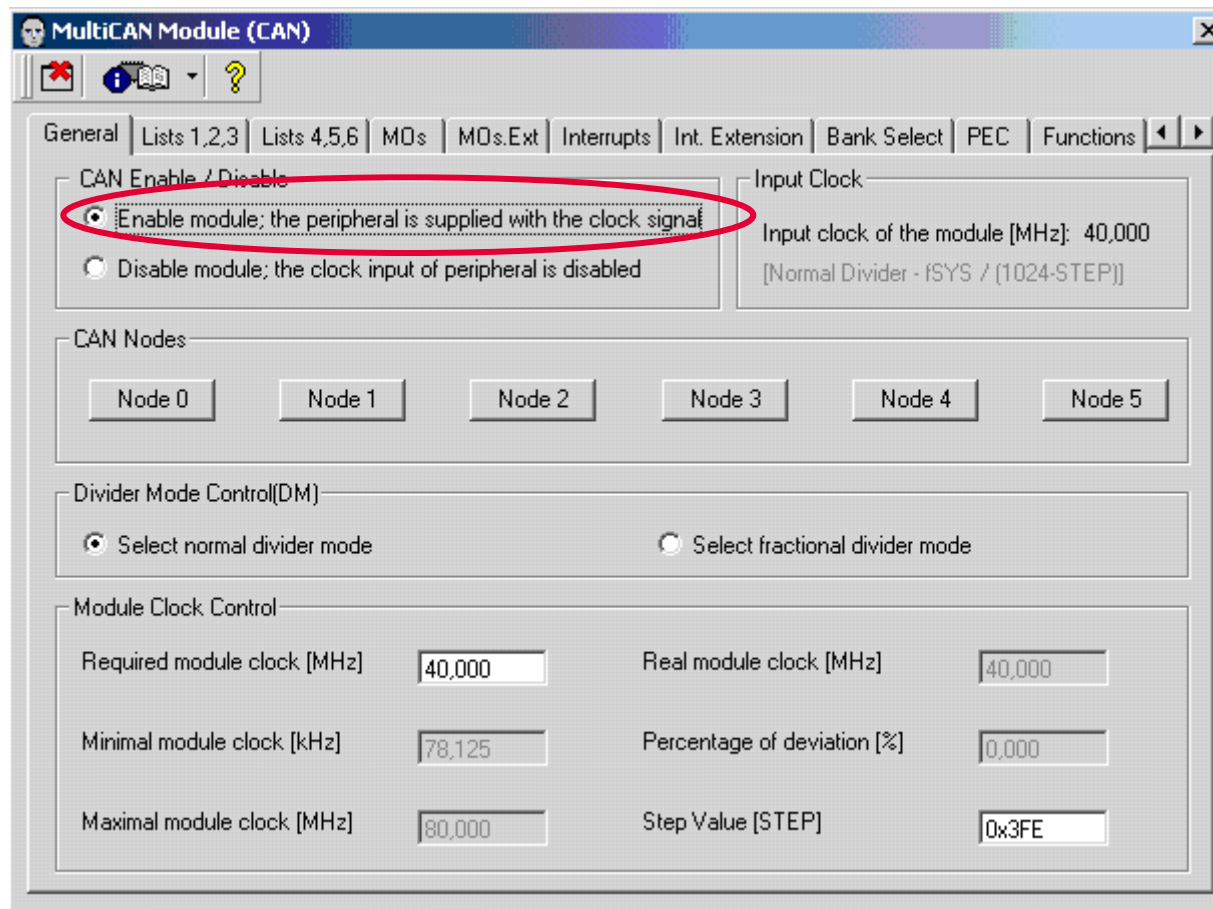
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure MultiCAN

□ General:

- Enable module



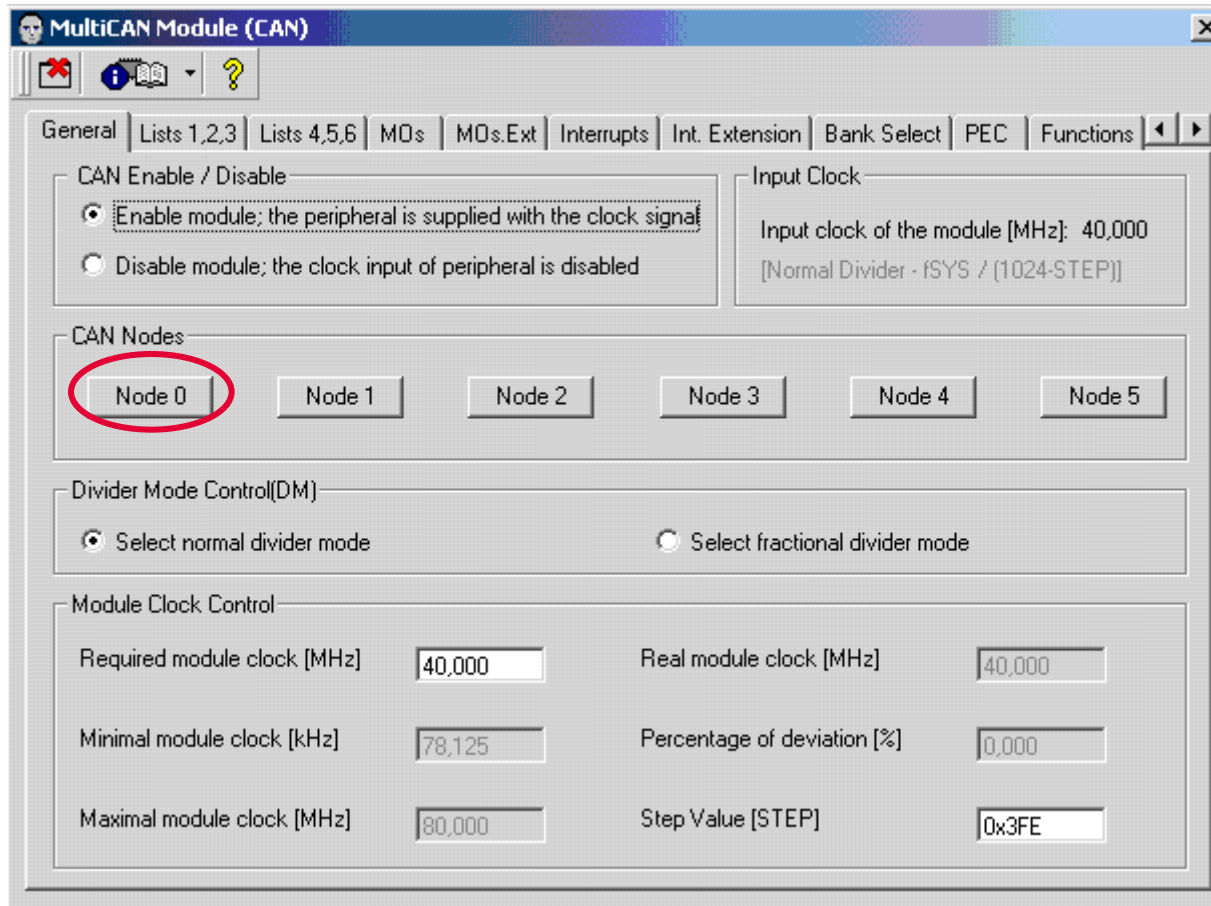
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure MultiCAN

□ General:

- Select Node 0



MultiCAN Module (CAN)

General | Lists 1,2,3 | Lists 4,5,6 | MOs | MOs.Ext | Interrupts | Int. Extension | Bank Select | PEC | Functions

CAN Enable / Disable

- ☒ Enable module; the peripheral is supplied with the clock signal
- ☐ Disable module; the clock input of peripheral is disabled

Input Clock

Input clock of the module [MHz]: 40,000
[Normal Divider - fSYS / (1024-STEP)]

CAN Nodes

Node 0 | Node 1 | Node 2 | Node 3 | Node 4 | Node 5

Divider Mode Control(DM)

- ☒ Select normal divider mode
- ☐ Select fractional divider mode

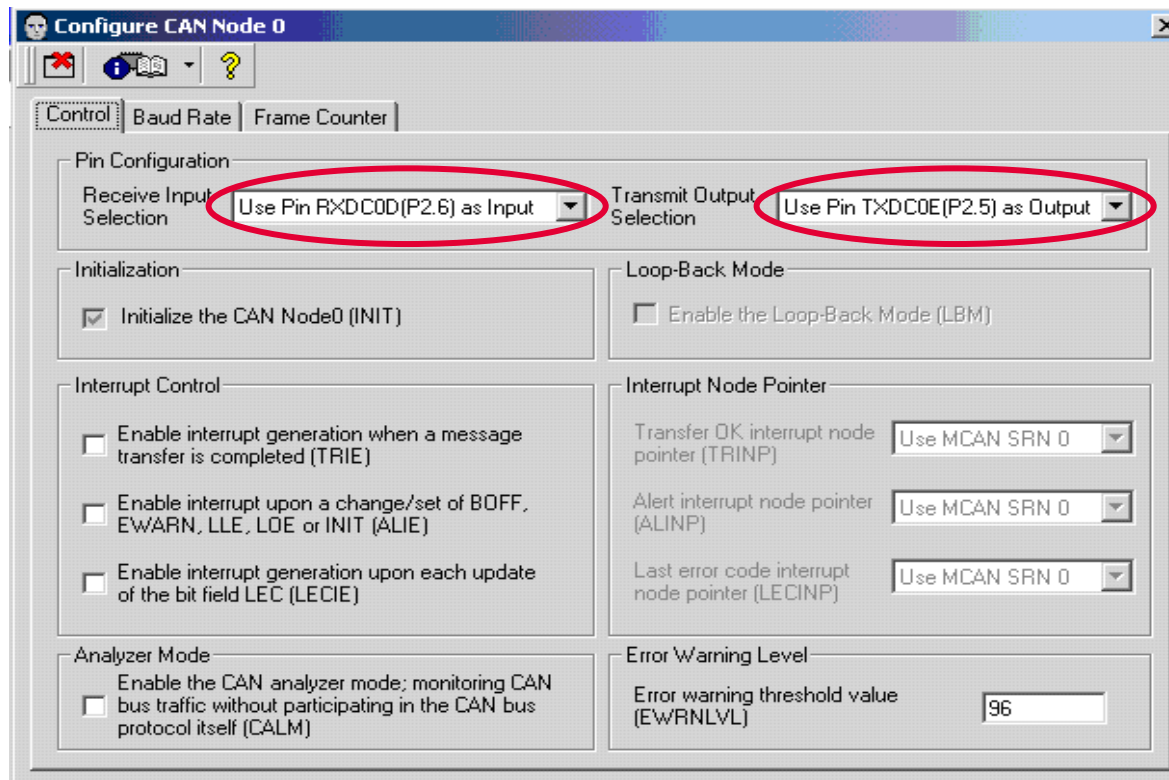
Module Clock Control

Required module clock [MHz]	40,000	Real module clock [MHz]	40,000
Minimal module clock [kHz]	78,125	Percentage of deviation [%]	0,000
Maximal module clock [MHz]	80,000	Step Value [STEP]	0x3FE

■ Configure CAN Node 0

□ General:

- Select P2.6 for Receive Input and P2.5 for Transmit Output
- Initialize the CAN node 0 automatically



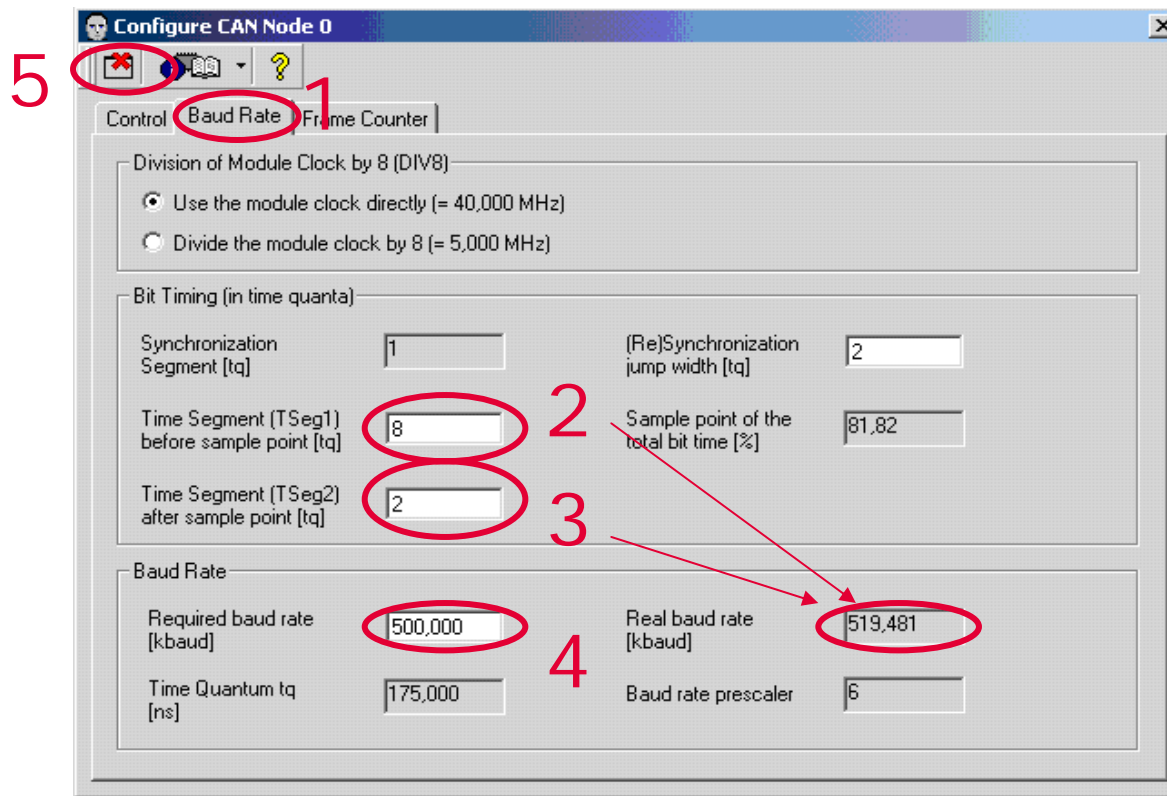
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure CAN Node 0

□ Baud Rate:

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



Configure CAN Node 0

Control Baud Rate Frame Counter

Division of Module Clock by 8 (DIV8)

☒ Use the module clock directly (= 40,000 MHz)

☐ Divide the module clock by 8 (= 5,000 MHz)

Bit Timing (in time quanta)

Synchronization Segment [tq] 1 (Re)Synchronization jump width [tq] 2

Time Segment (TSeg1) before sample point [tq] 8 Sample point of the total bit time [%] 81,82

Time Segment (TSeg2) after sample point [tq] 2

Baud Rate

Required baud rate [kbaud] 500,000 Real baud rate [kbaud] 519,481

Time Quantum tq [ns] 175,000 Baud rate prescaler 6

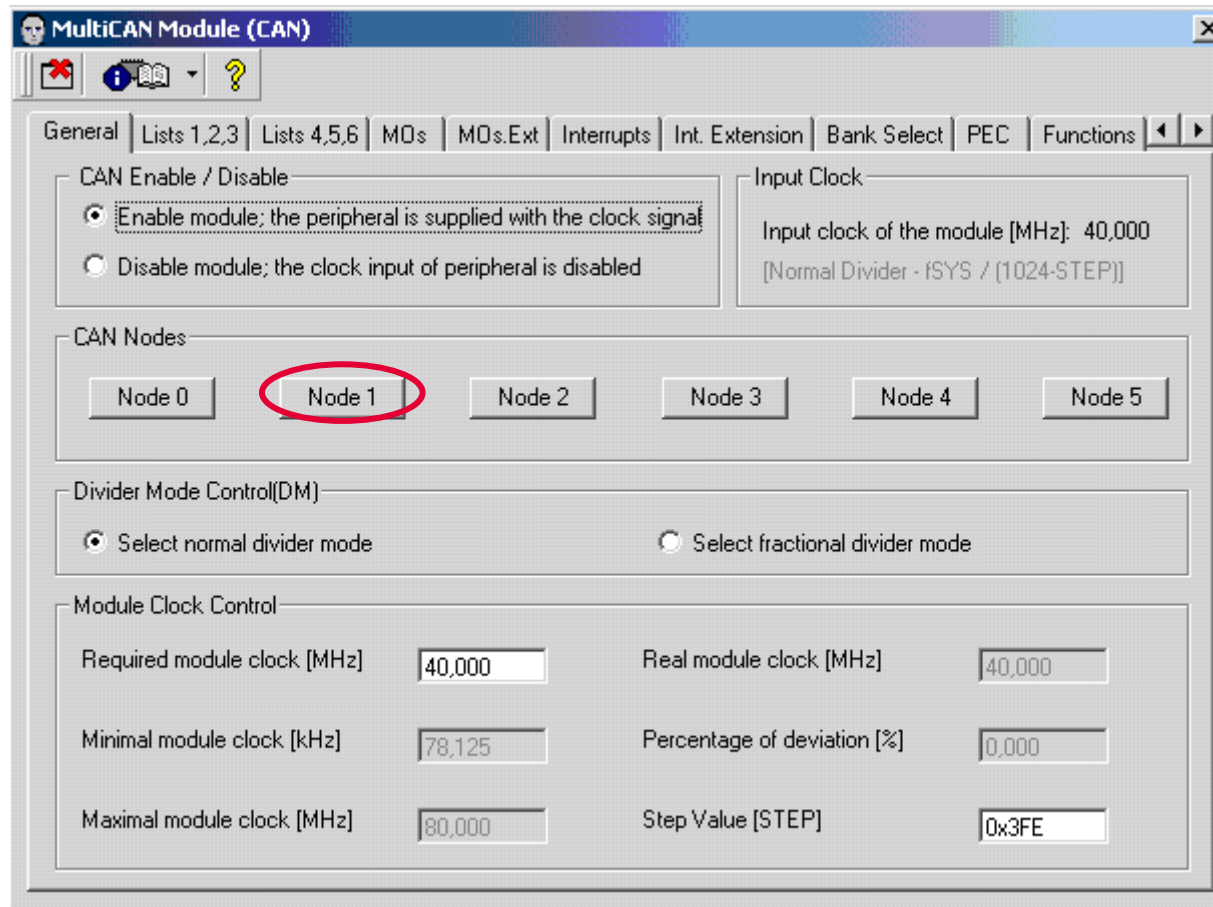
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure MultiCAN

□ General:

- Select Node 1



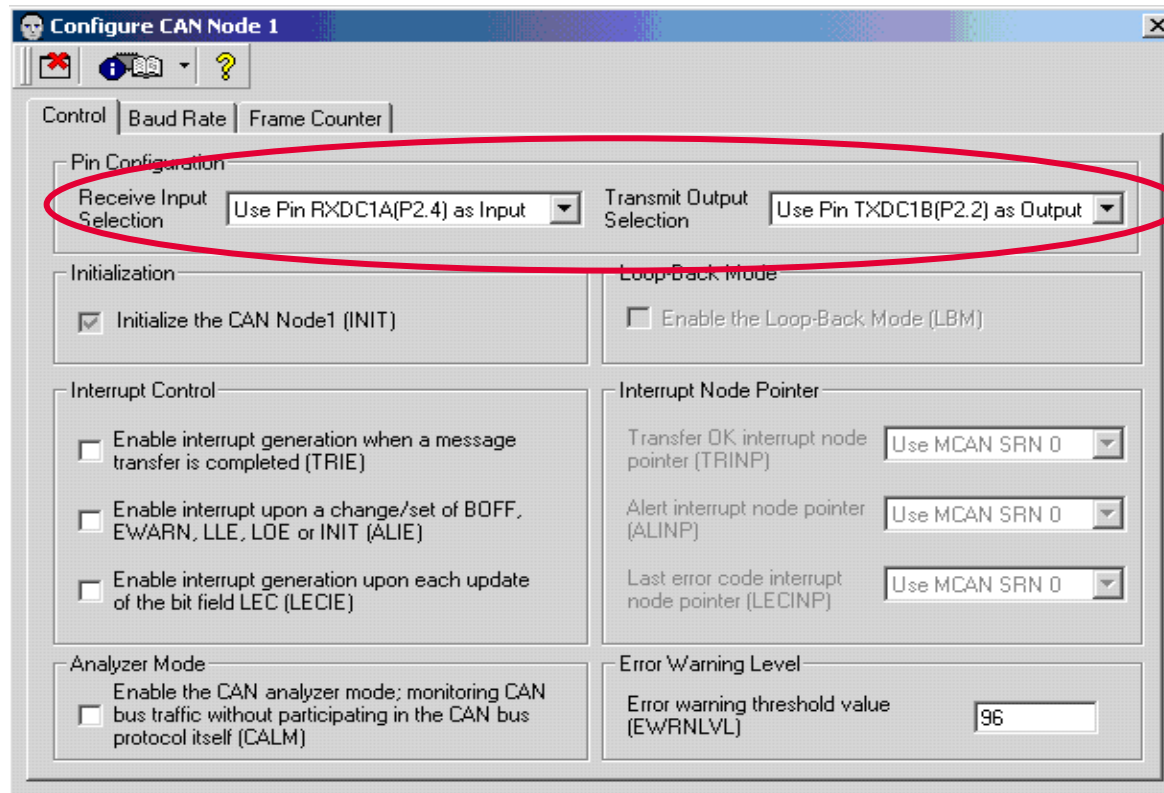
HOT Exercise CAN_3 - DAVe Configurations

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



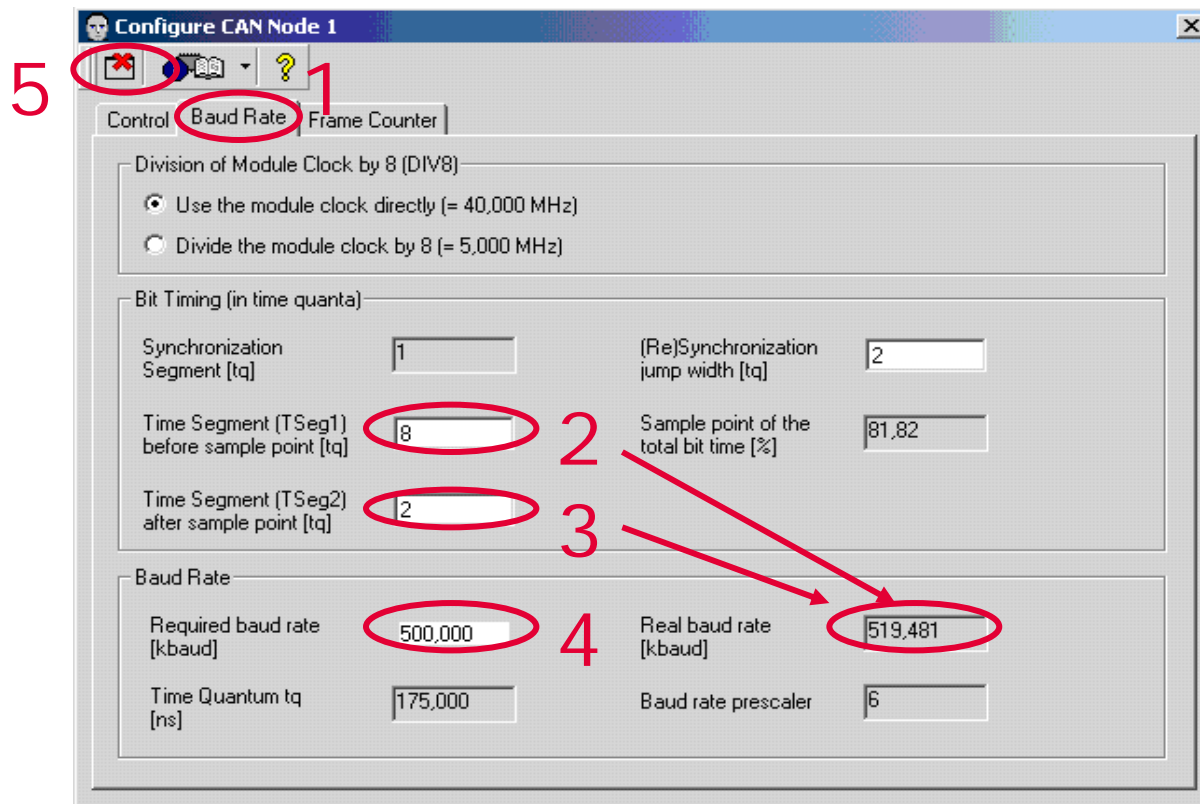
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure CAN Node 1

□ Baud Rate:

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



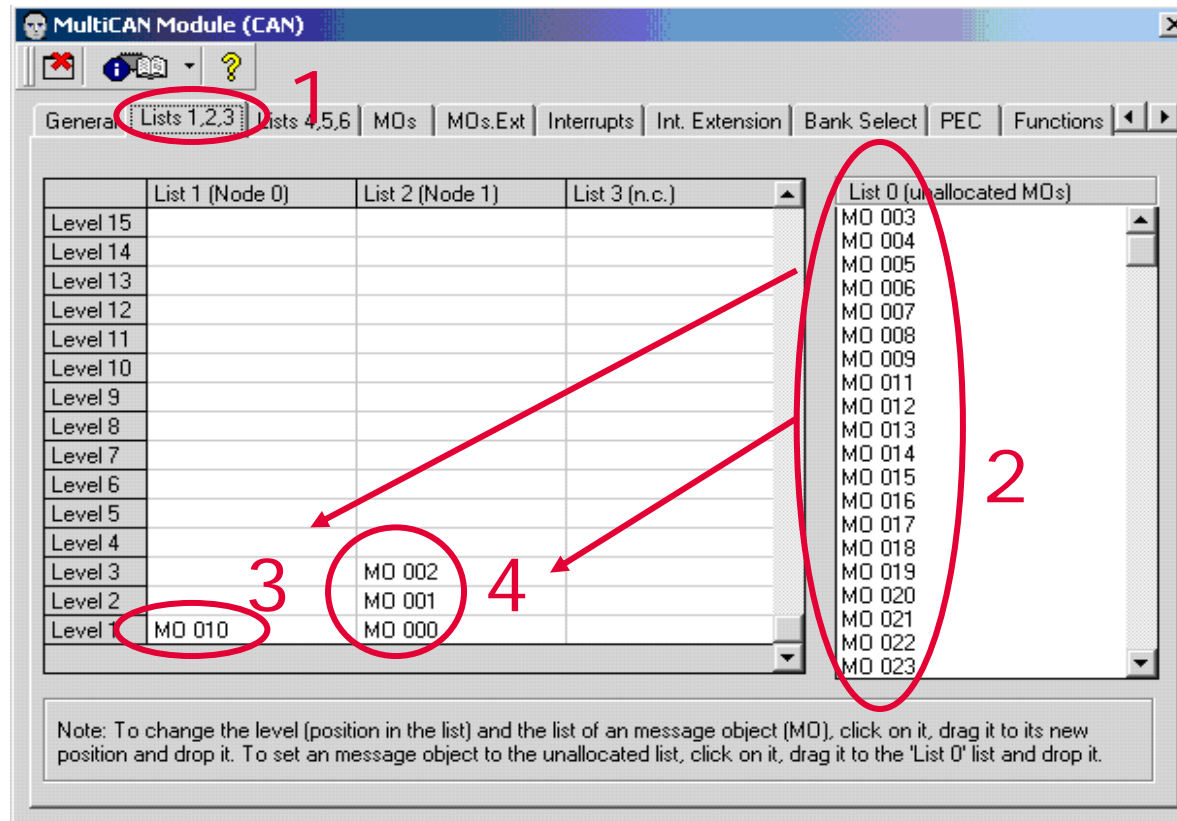
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ 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)



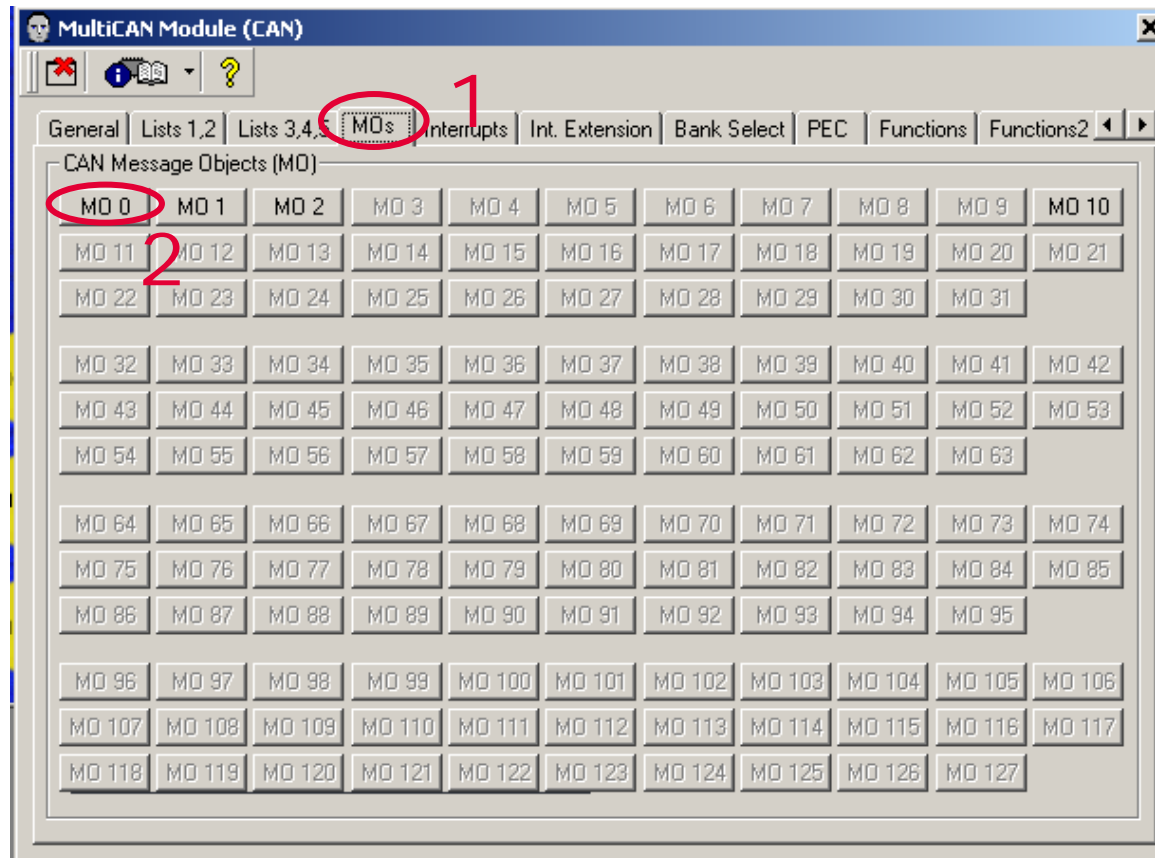
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure MultiCAN

□ MOs:

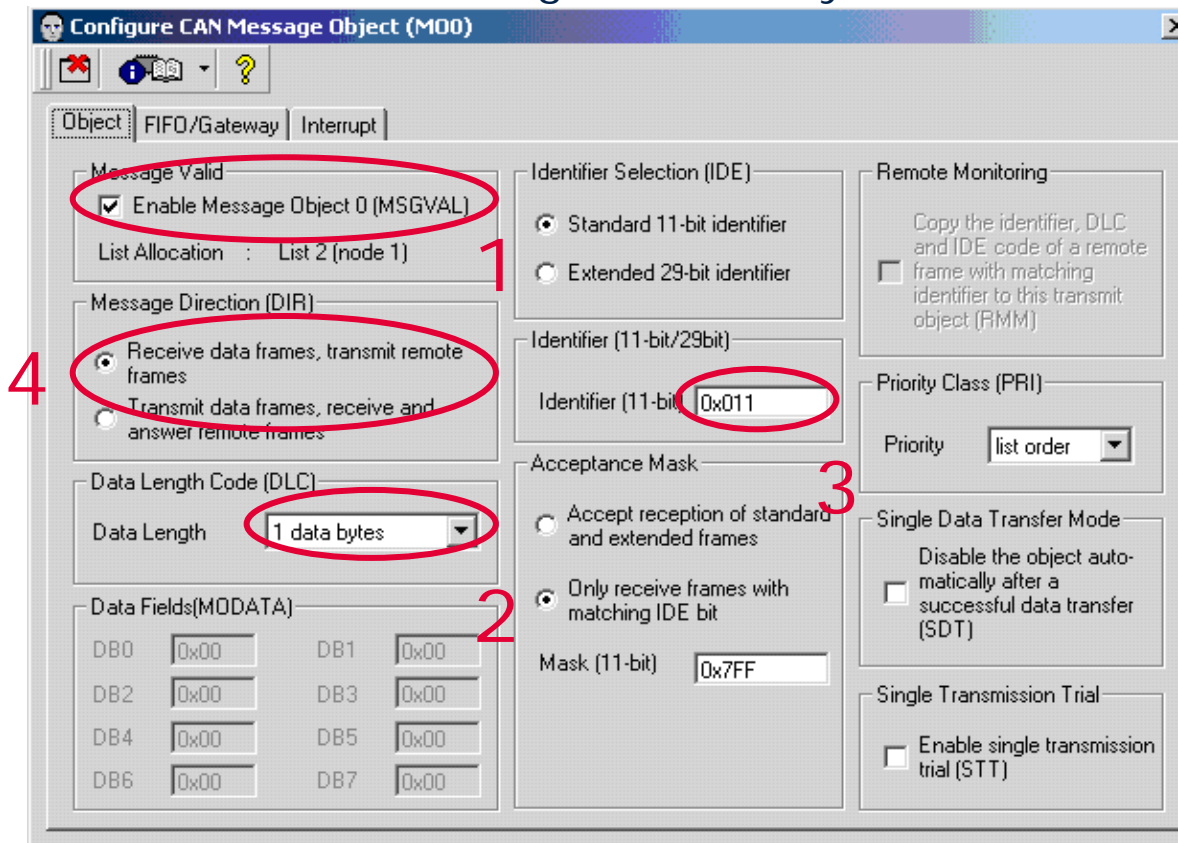
- Select MO0



■ Configure CAN Message Object (MO0)

□ Object:

- Enable MO0, Select Receive data frames
- Identifier: 0x011, Data Length: 1 data bytes



Configure CAN Message Object (MO0)

Object | FIFO/Gateway | Interrupt

Message Valid

☒ Enable Message Object 0 (MSGVAL)

List Allocation : List 2 (node 1)

Message Direction (DIR)

☒ Receive data frames, transmit remote frames

☐ Transmit data frames, receive and answer remote frames

Data Length Code (DLC)

Data Length: 1 data bytes

Data Fields (MODATA)

DB0	0x00	DB1	0x00
DB2	0x00	DB3	0x00
DB4	0x00	DB5	0x00
DB6	0x00	DB7	0x00

Identifier Selection (IDE)

☒ Standard 11-bit identifier

☐ Extended 29-bit identifier

Identifier (11-bit/29bit)

Identifier (11-bit): 0x011

Acceptance Mask

☐ Accept reception of standard and extended frames

☒ Only receive frames with matching IDE bit

Mask (11-bit): 0x7FF

Remote Monitoring

☐ Copy the identifier, DLC and IDE code of a remote frame with matching identifier to this transmit object (RMM)

Priority Class (PRI)

Priority: list order

Single Data Transfer Mode

☐ Disable the object automatically after a successful data transfer (SDT)

Single Transmission Trial

☐ Enable single transmission trial (STT)

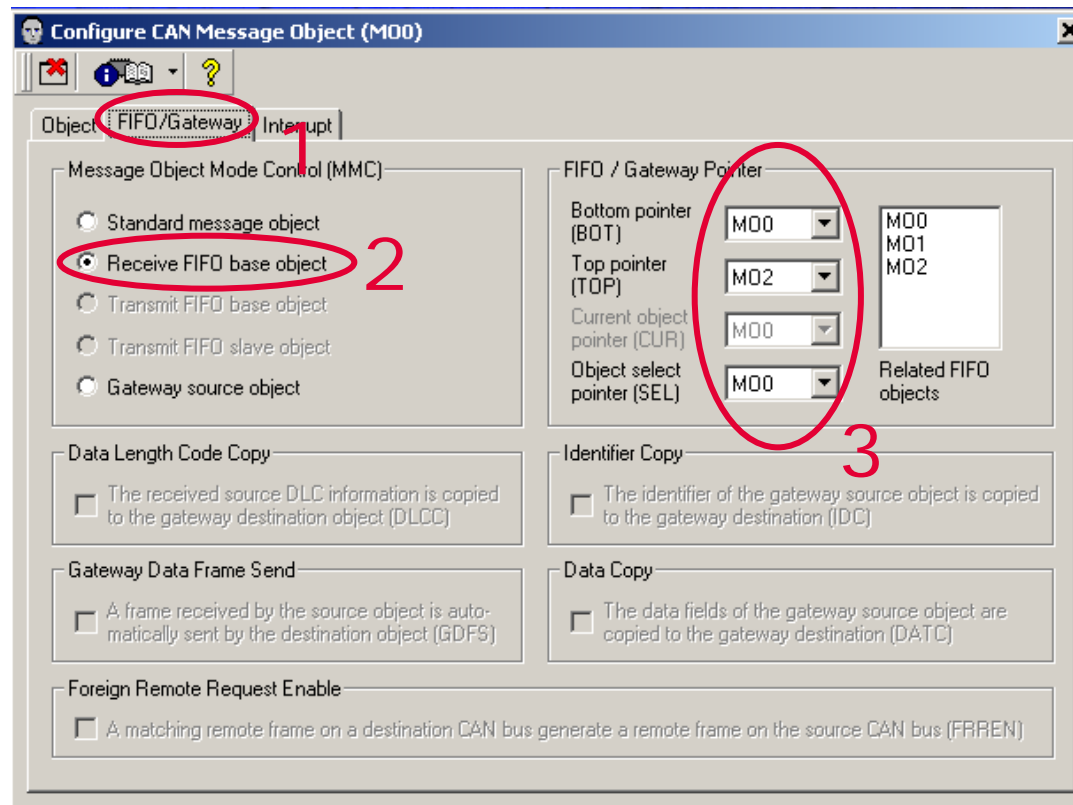
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure CAN Message Object (MO0)

□ FIFO/Gateway:

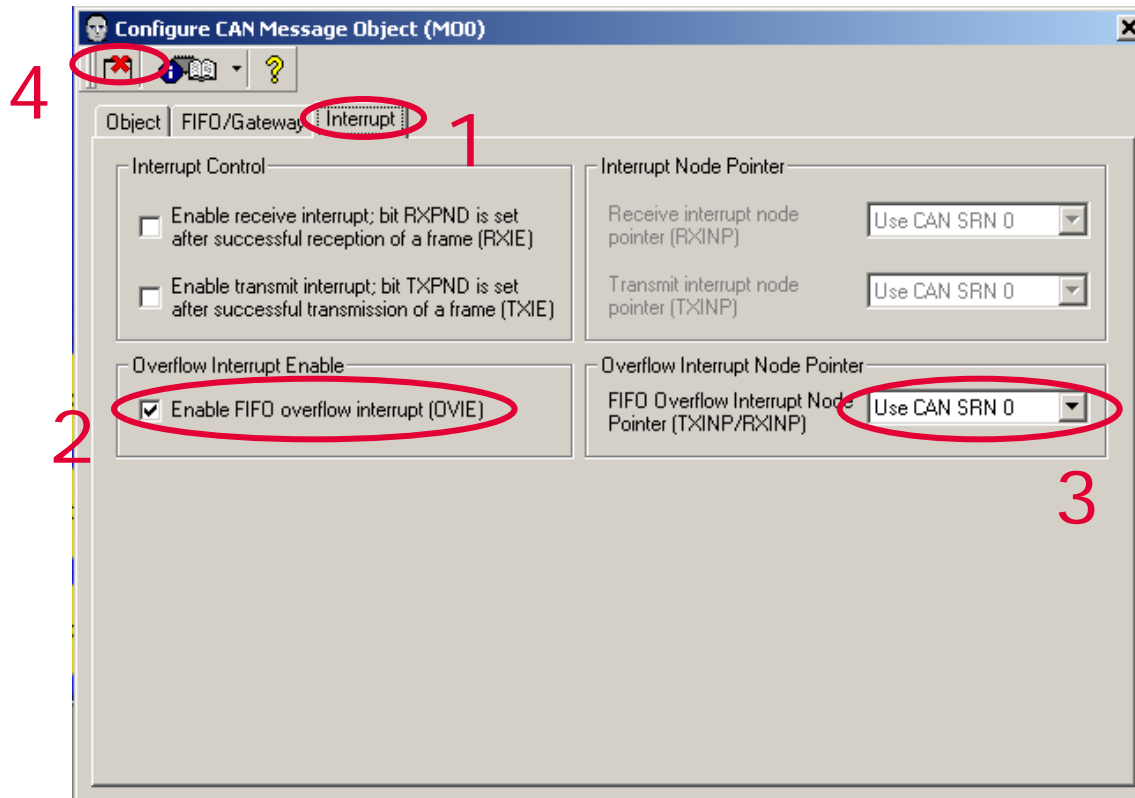
- Enable MO0, Select Receive FIFO base object
- FIFO Pointers: Bottom MO0, Top MO2, Object select MO0



■ Configure CAN Message Object (M00)

□ Interrupt:

- Enable FIFO overflow interrupt
- Use CAN SRN 0



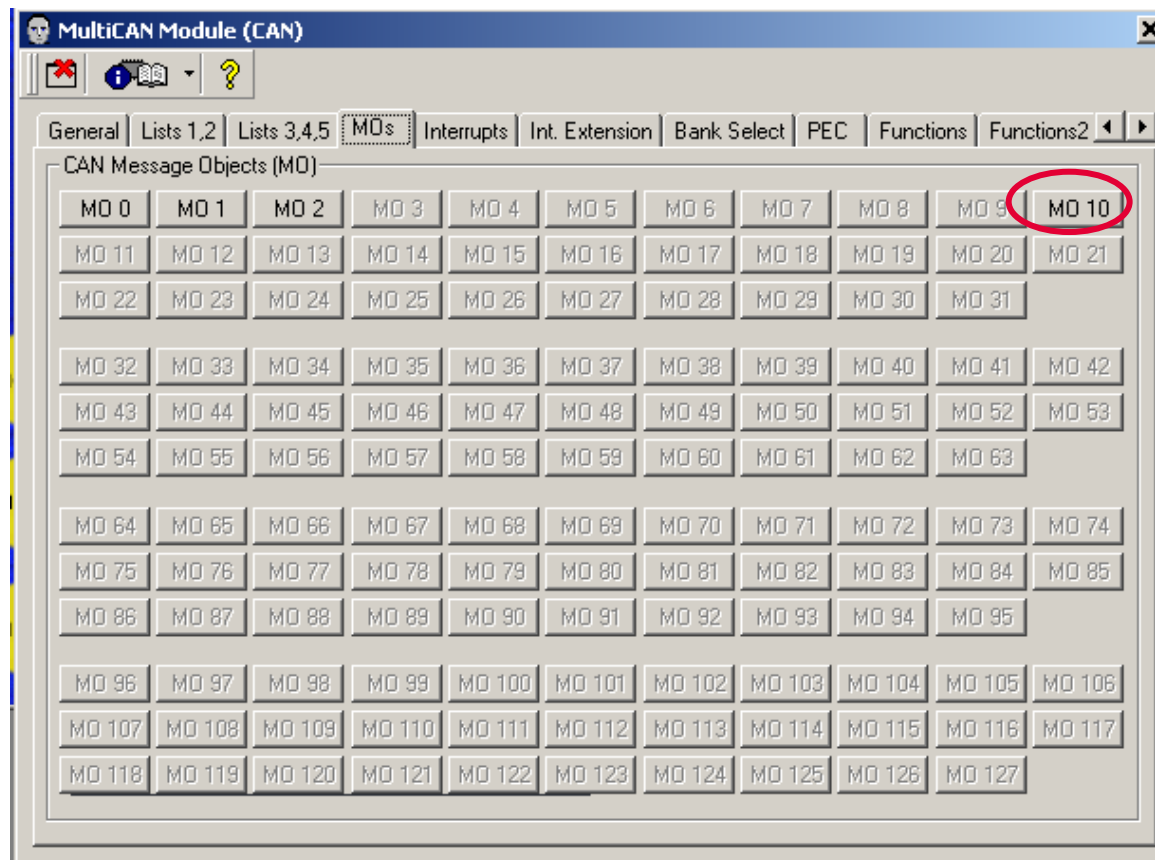
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure MultiCAN

□ MOs:

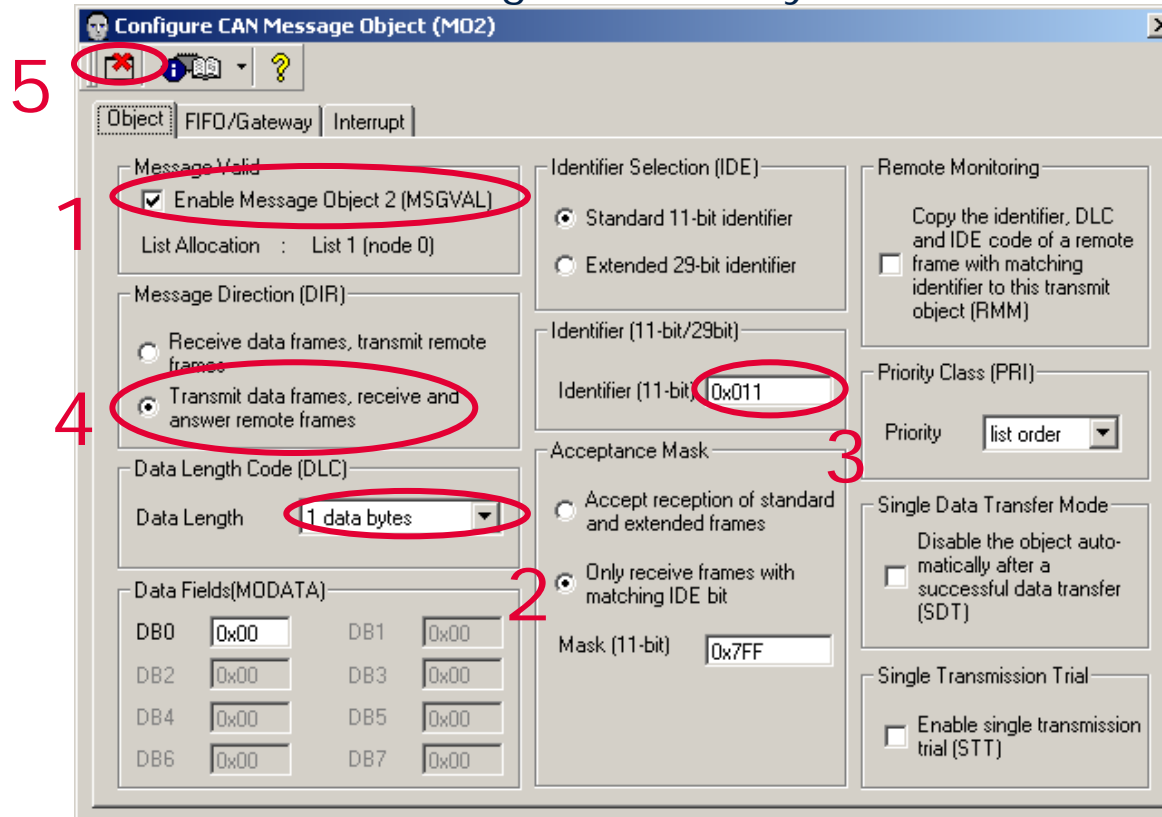
- Select MO10



■ Configure CAN Message Object (MO10)

□ Object:

- Enable MO10, Select Transmit data frames
- Identifier: 0x011, Data Length: 1 data bytes



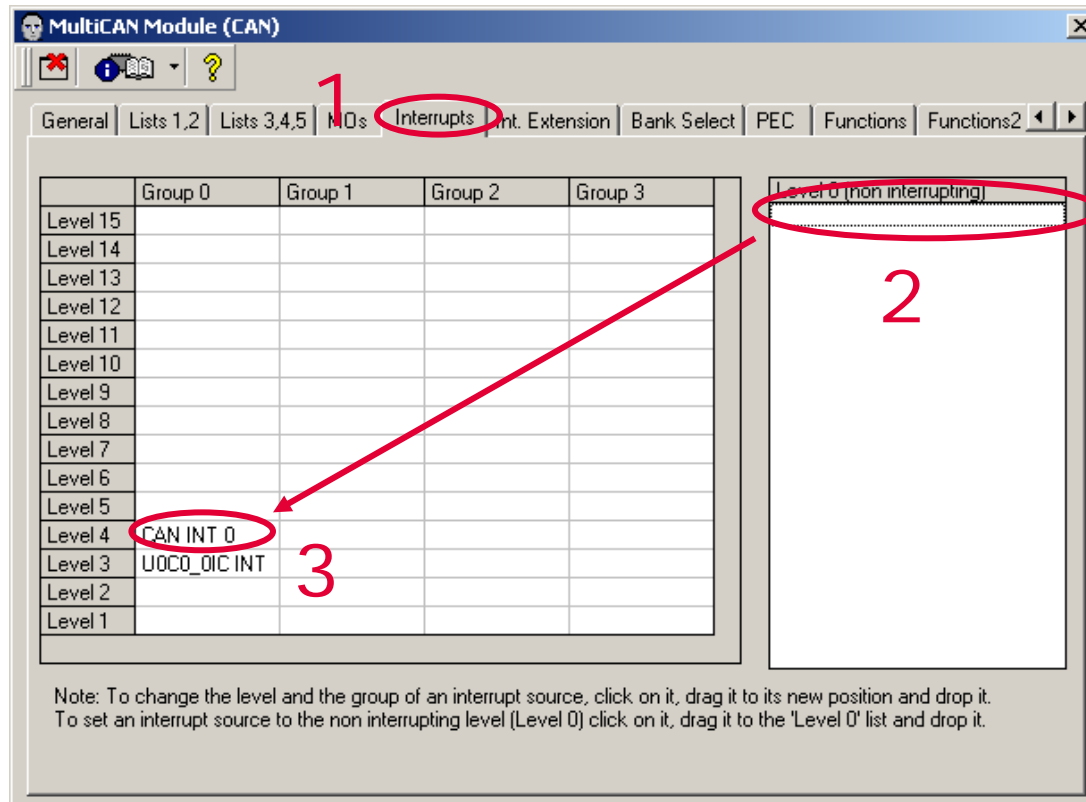
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure MultiCAN

□ Interrupts :

- Drag 'CAN INT 0' and drop it to Interrupt Level 4, Group 0



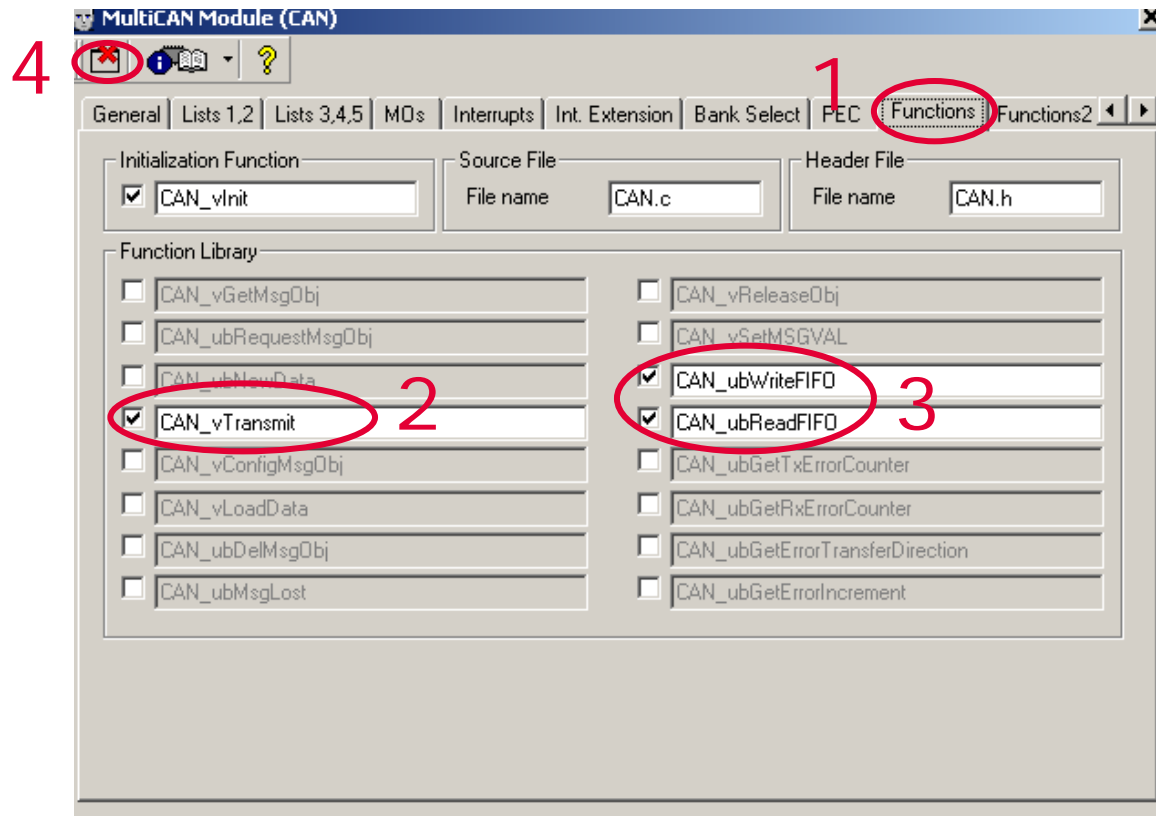
HOT Exercise CAN_3 - DAVe Configurations

MultiCAN settings

■ Configure MultiCAN

□ Functions:

- Include 'CAN_vInit'
- Include 'CAN_vTransmit', 'CAN_ubWriteFIFO, CAN_ubReadFIFO'



HOT Exercise CAN_3 - DAVe Configurations

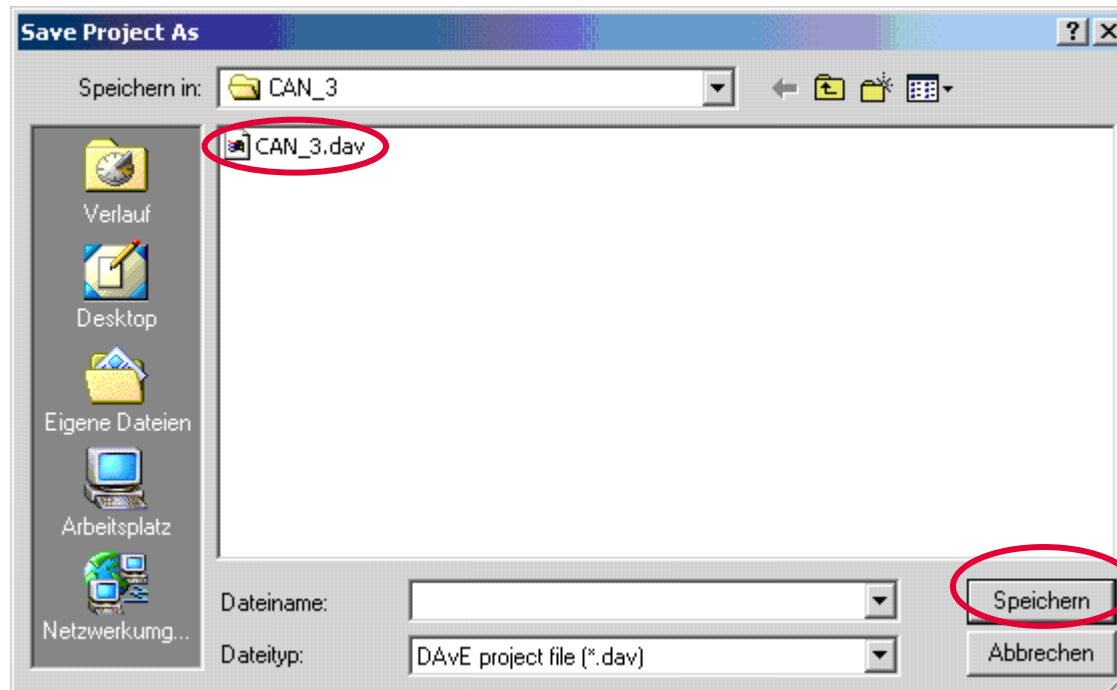
Save DAVe Project

■ 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

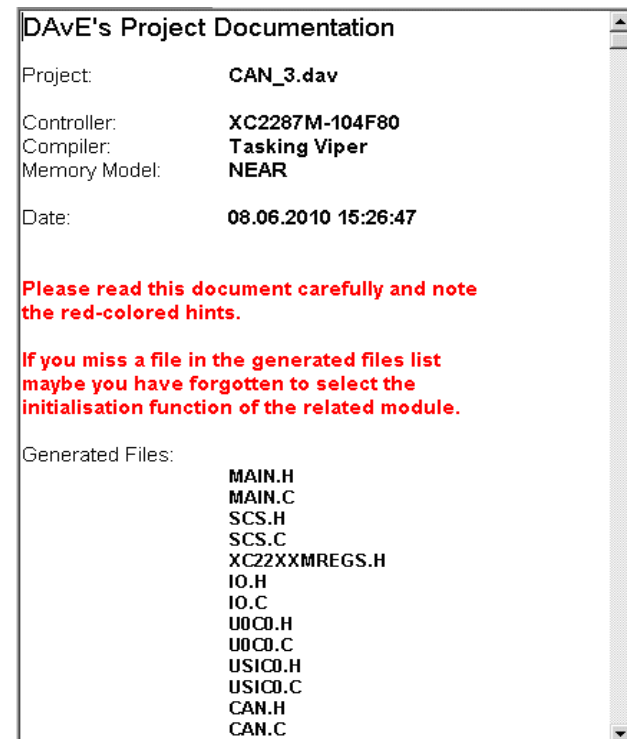


□ DAVe generated code files are

- '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

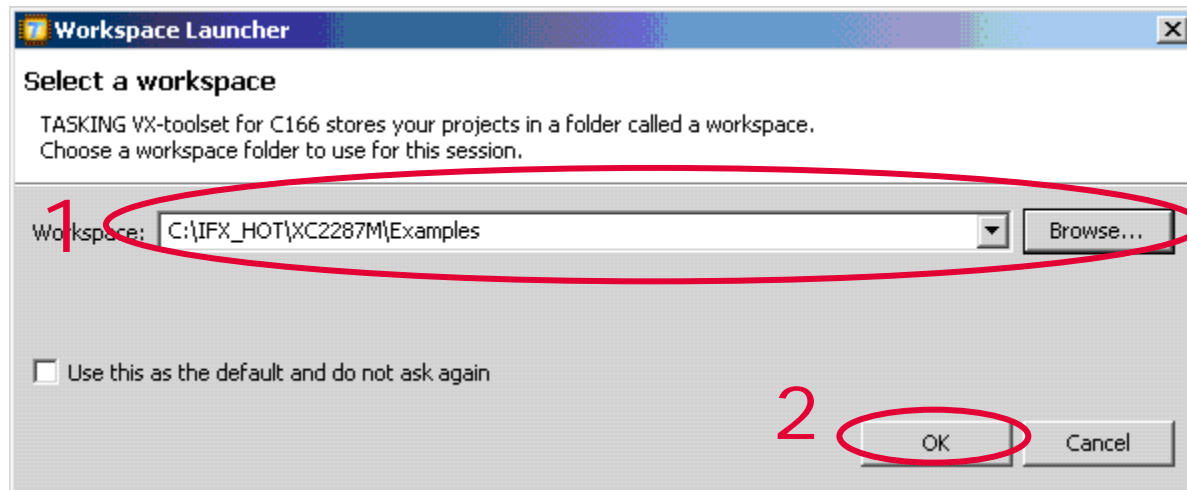


■ Open Project Work Space



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

□ Click ‘OK’



- Create New Project
- Click on Workbench (if not already there...)

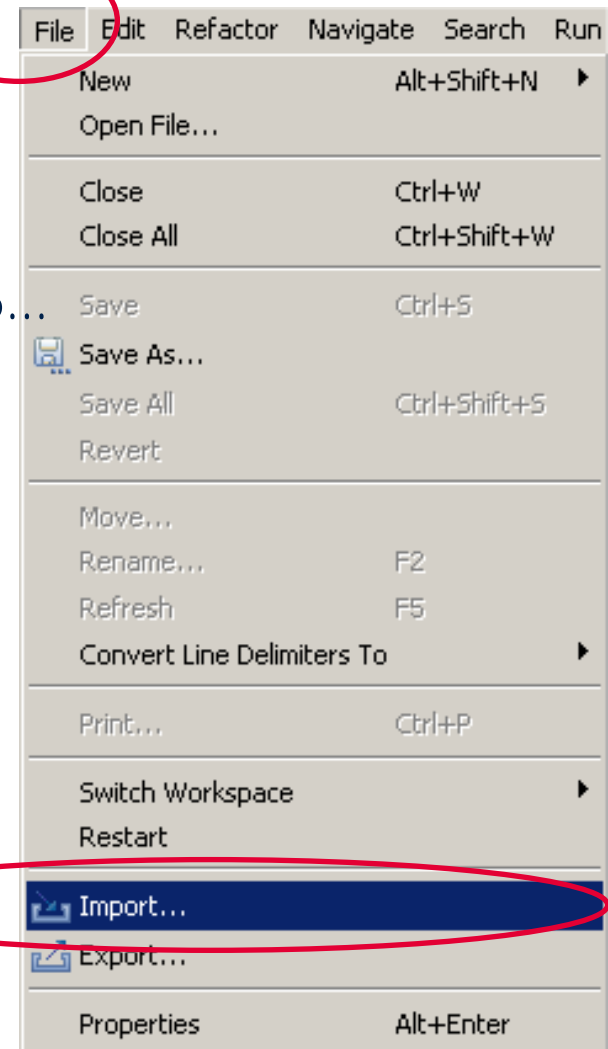


■ Import DAVE Project

- Click on File -> Import
- Select Tasking VX-toolset for C166...
- Click 'OK'

1

2

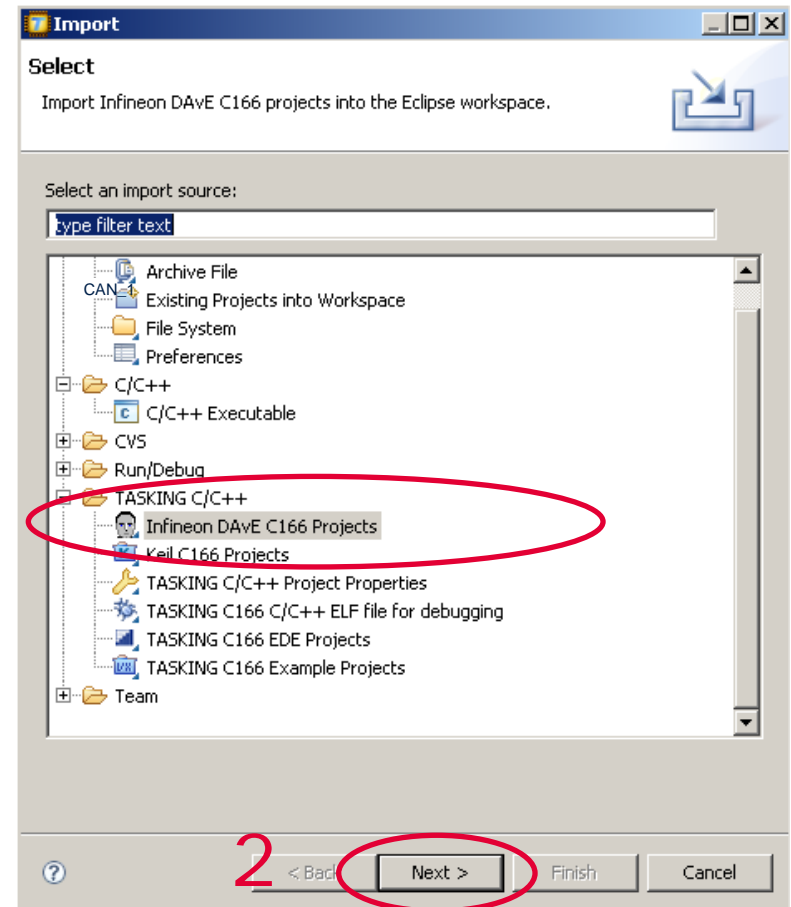


■ Import DAVE Project

□ Click `Infineon DAvE C166 Project`

□ Click 'Next'

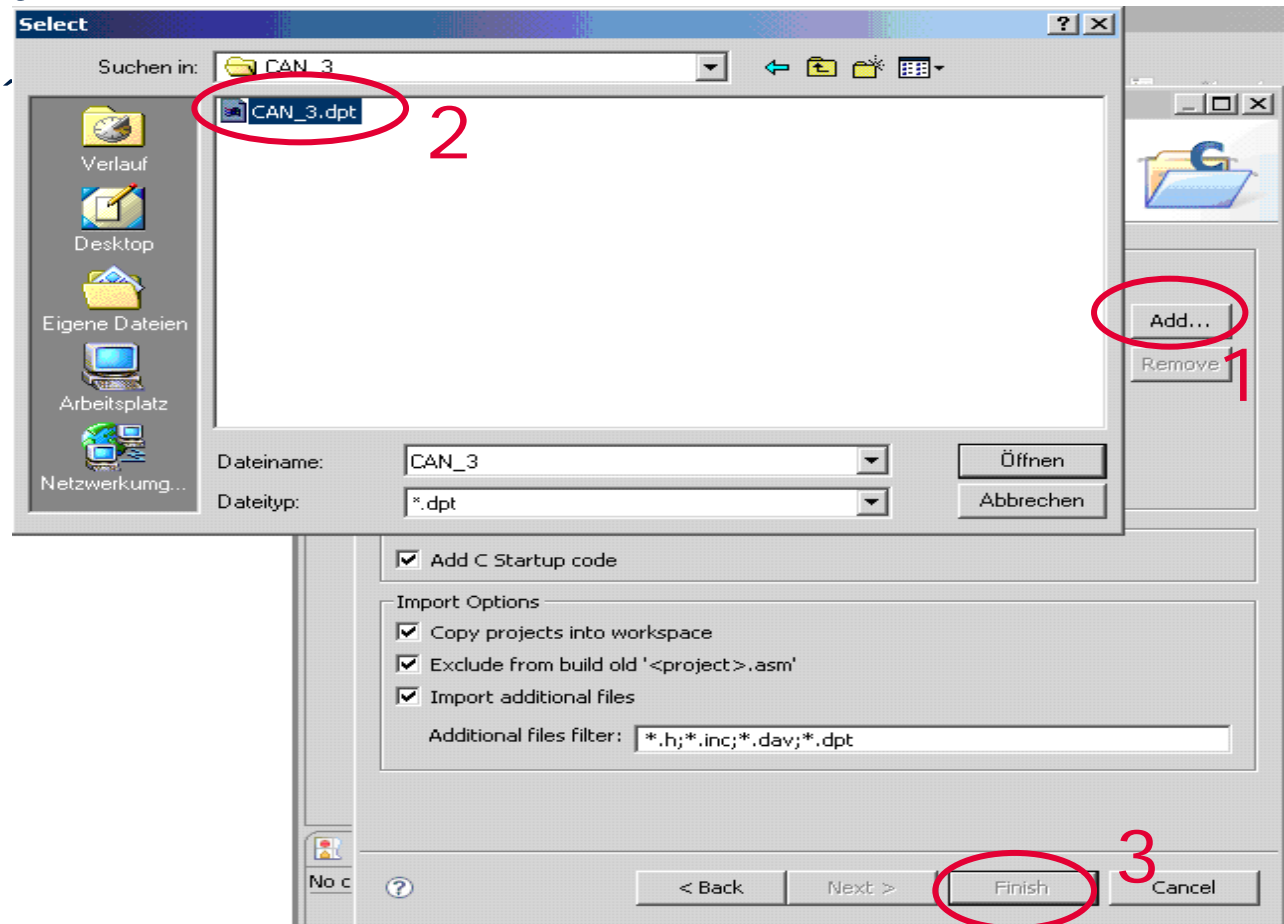
1



■ Import DAVe Project

□ Add Dave Project 'CAN_3'

□ Click 'Finish'

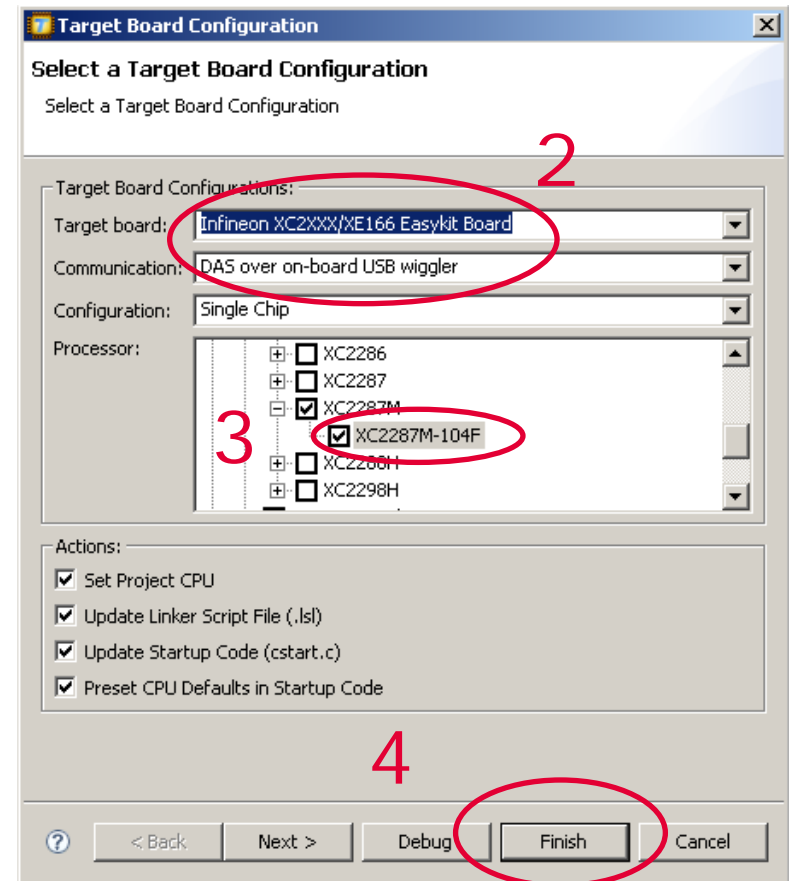
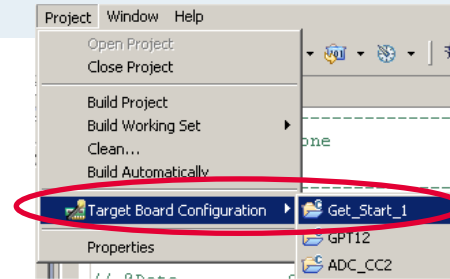


■ Configure Target Board



1

- Select the project in the navigator
- Select 'Project/Target Board Configuration'
- Select 'Infineon XC2000/XE166 Easykit Board'
- Select 'XC2287M-104F'
- Choose 'XC2287M-104F'
- 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:

```
while(1)
{
    // USER CODE BEGIN (Main,4)
```

```
    BlinkLED();
```

```
    // USER CODE END
}
```

- In the ISR function 'UOC0_ASC_vi0IC(void)' (almost at the end)

```
_interrupt(UOC0_OINT) void UOC0_ASC_vi0IC(void)
{
    // USER CODE BEGIN (ASCOIC,2)
    // USER CODE END

    if (UOC0_PSR & 0x4000)
    {
        // USER CODE BEGIN (ASCOIC,4)

        CAN_MODATA10LL = UOC0_RBUF; //store received character in MO10
        CAN_vTransmit(10);

        // USER CODE END

        UOC0_PSCR |= 0x4000;        // clear PSR_RIF
    }
    // USER CODE BEGIN (ASCOIC,15)
    // USER CODE END

} // End of function UOC0_ASC_vi0IC
```

HOT Exercise CAN_3 - Complete code

Edit File 'CAN.C'



- In the ISR function 'CAN_vISRNO(void)' (almost at the end)

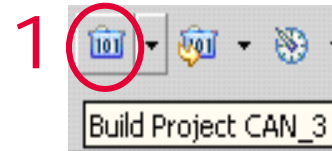
```
_interrupt(CAN_SRNOINT) void CAN_vISRNO(void)
{
    // USER CODE BEGIN (SRNO,2)

    IO_vTogglePin(IO_P10_0);
    U0C0_ASC_vSendData (CAN_MODATA0LL);
    U0C0_ASC_vSendData (CAN_MODATA1LL);
    U0C0_ASC_vSendData (CAN_MODATA2LL);

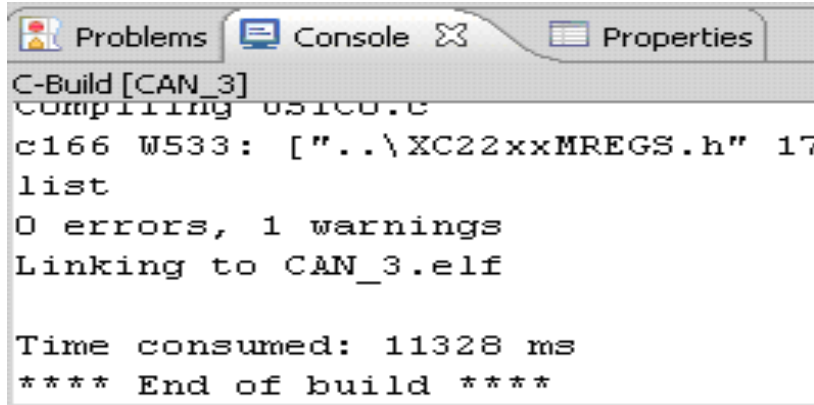
    // USER CODE END

:
} // End of function CAN_vISRNO
```

HOT Exercise CAN_3 – Tasking VX Toolset Build Project



- Click on 'Build Project CAN'

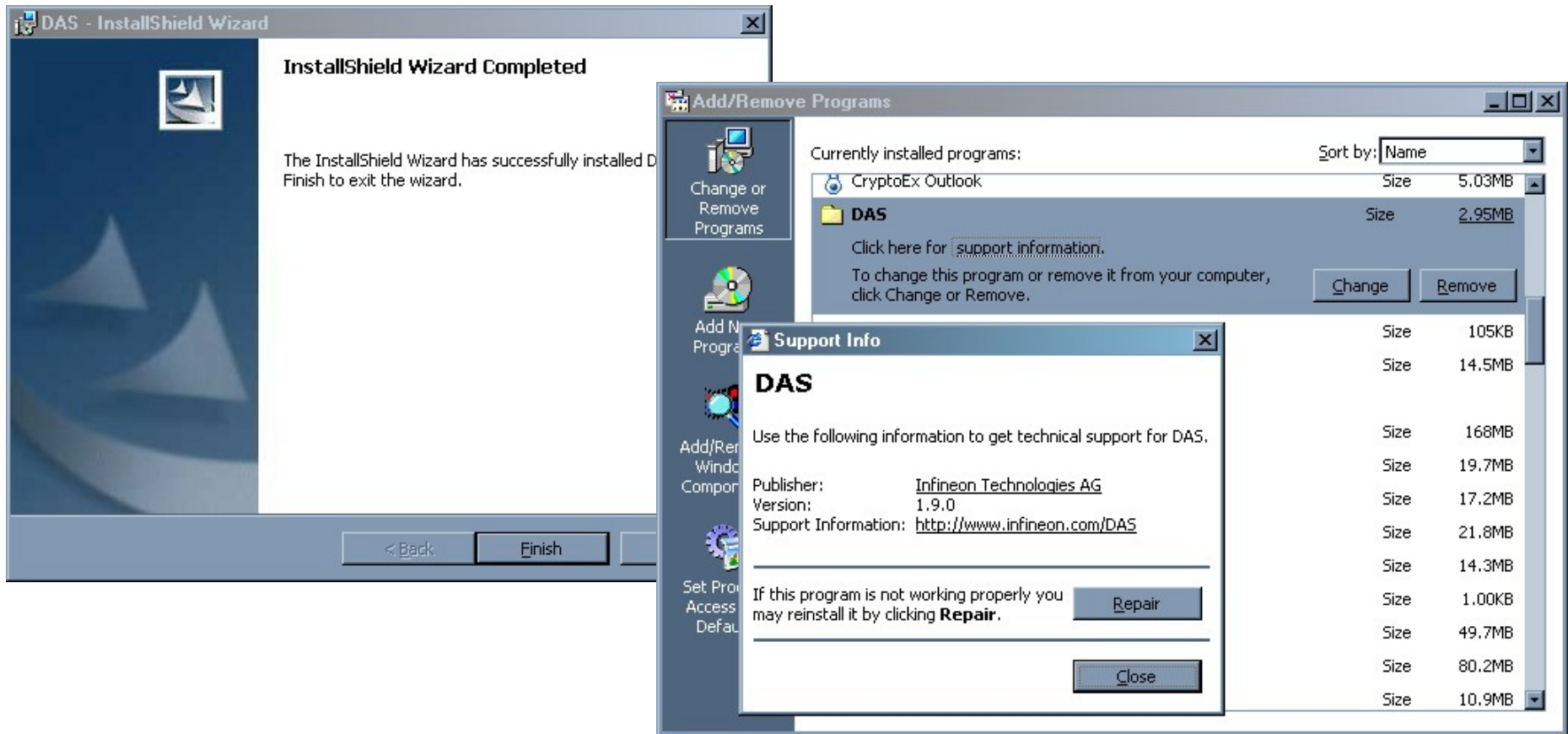


```
C-Build [CAN_3]
Compiling 05100.c
c166 W533: ["..\XC22xxMREGS.h" 17
list
0 errors, 1 warnings
Linking to CAN_3.elf

Time consumed: 11328 ms
**** End of build ****
```

HOT Exercise CAN_3 - Device Access Server

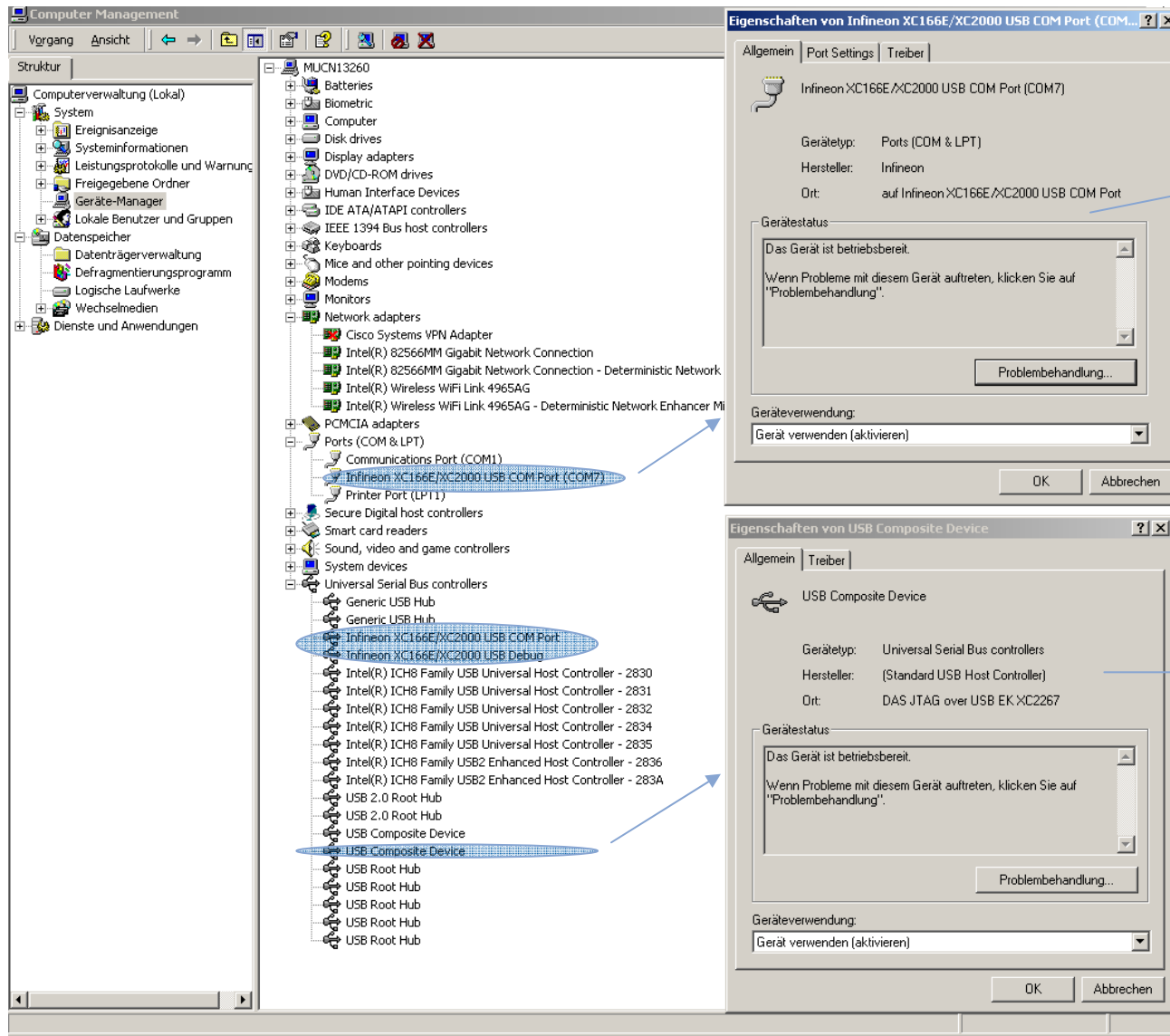
Check for the latest DAS version



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

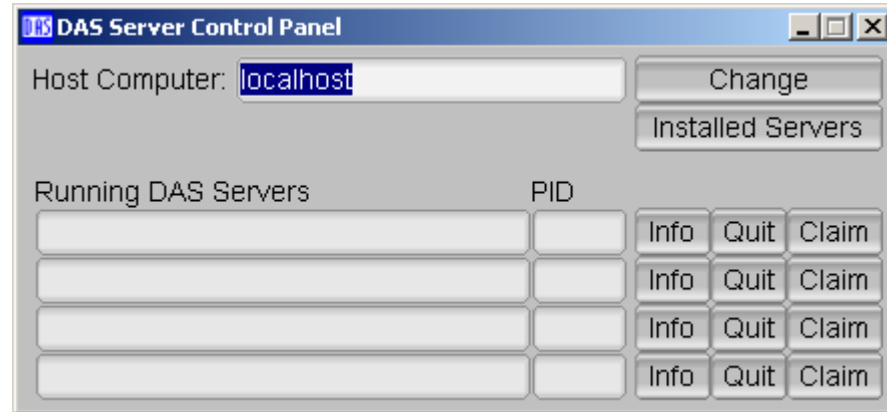
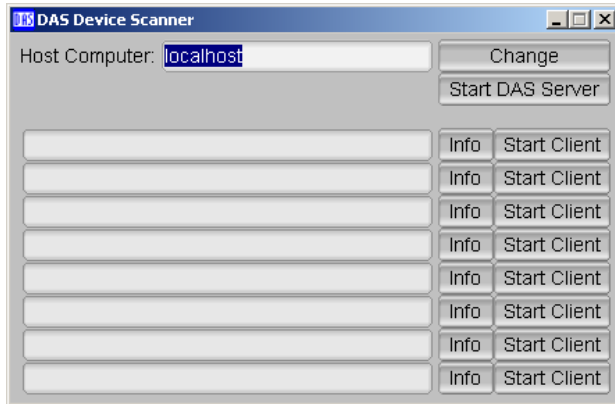
The DAS JTAG composite device gets identified

- 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

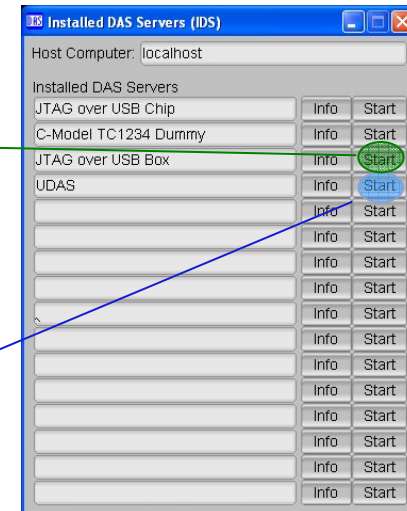
1. Start DAS device scanner
2. Start DAS Server Control panel



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

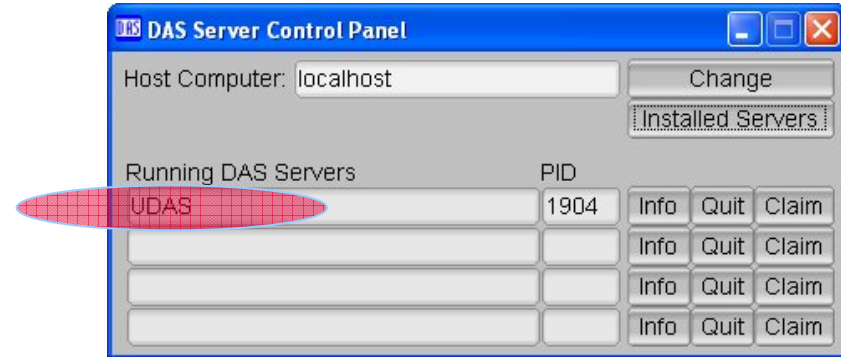
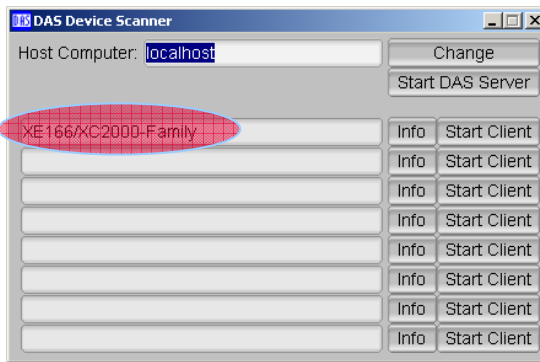
Incase you are connected via the USB Wiggler box,
then start „JTAG over USB Box“

Incase you are connected via the FTDI chip or mini wiggler,
then start „UDAS“

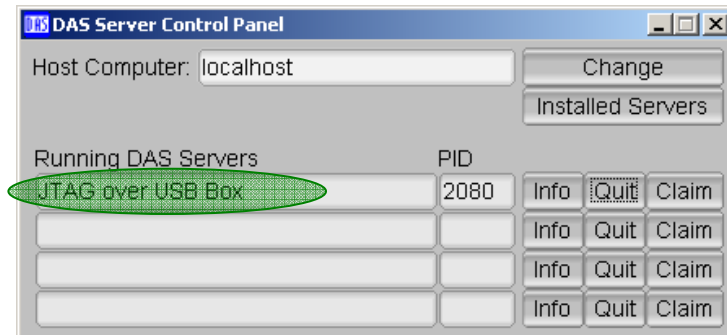
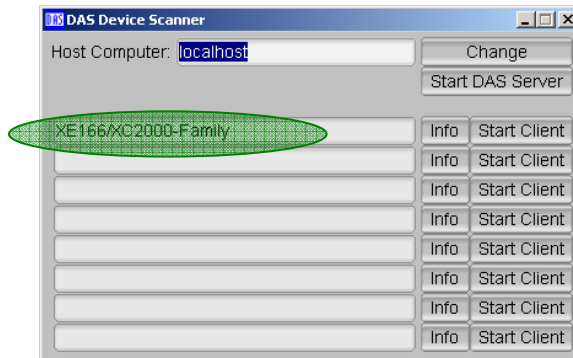


3.) Starting the servers manually

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



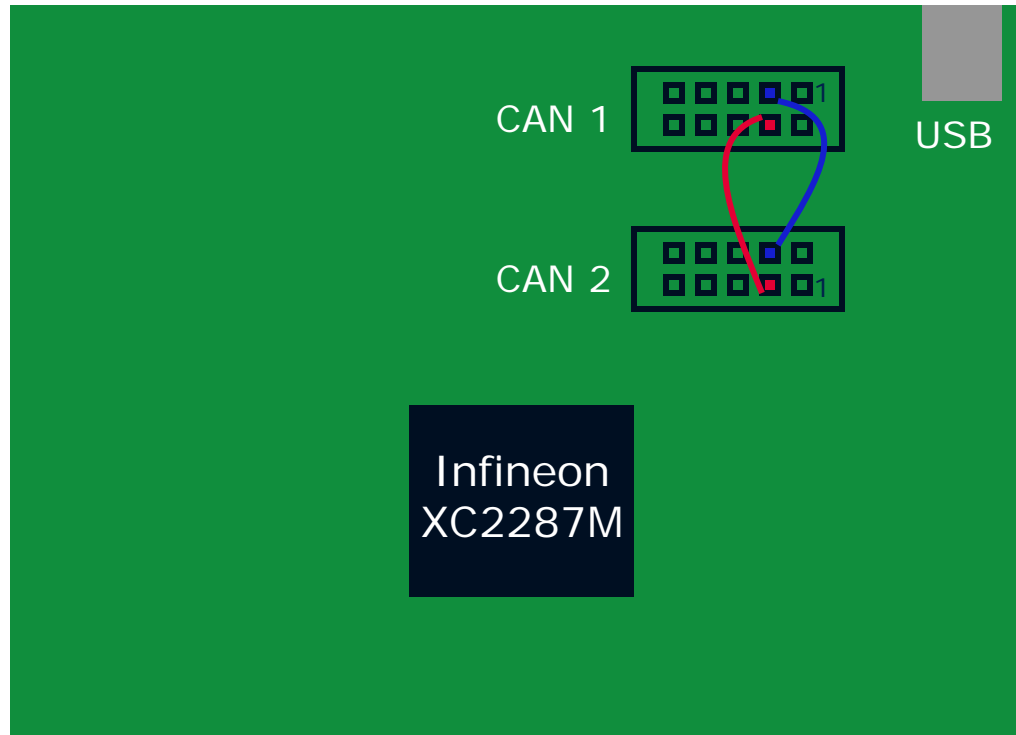
5. In case „JTAG over USB Box“ server is started and XC2000 starter kit is connected via Wiggler box, following status changes could be noted



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)

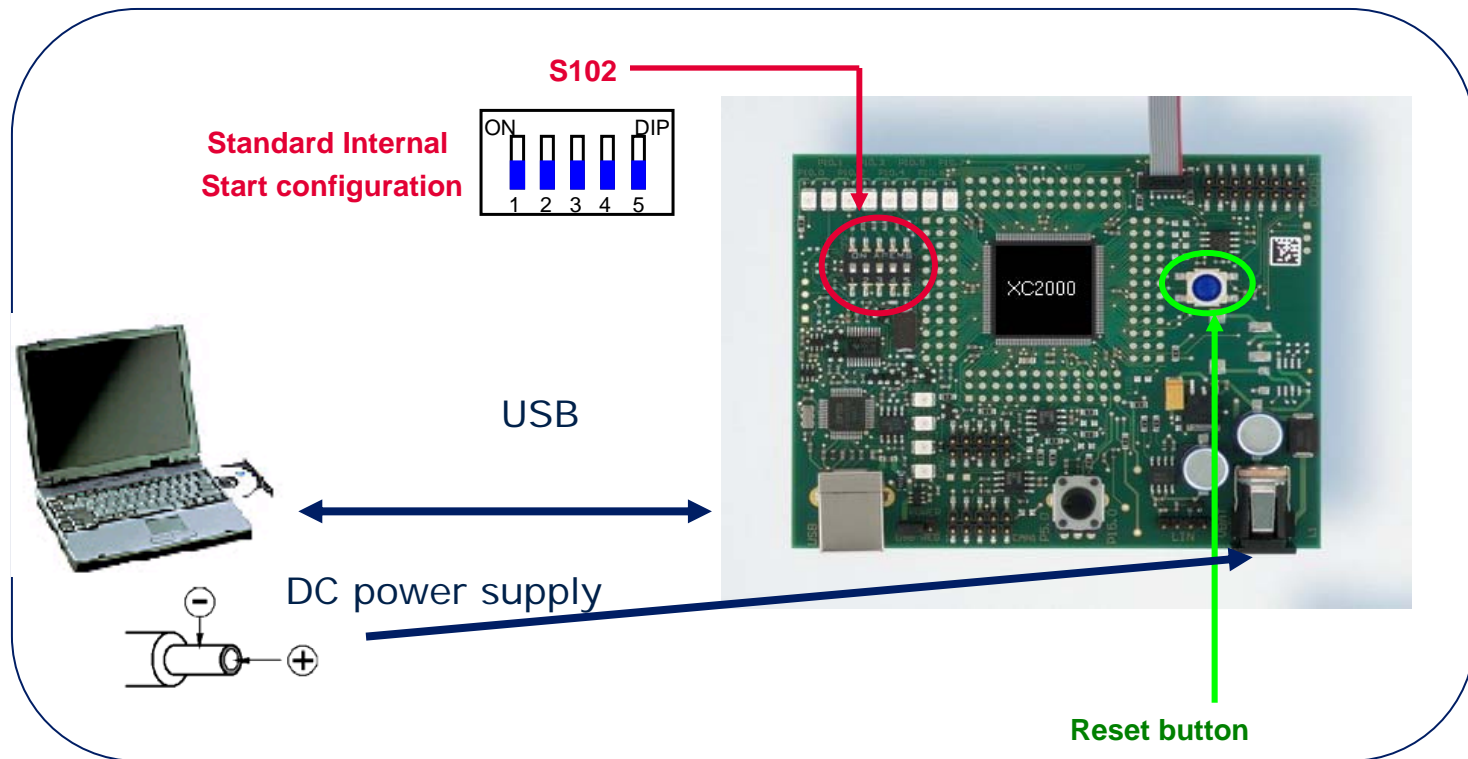


HOT Exercise CAN_3

Connect XC2287M Board

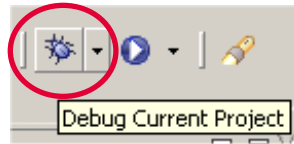
■ Internal start from Flash

- Connect XC2287M Board to PC
- Modify the DIP switch settings, S102: **OFF-OFF-OFF-OFF-OFF**
- Reset the board (press the reset button)



HOT Exercise CAN_3 – Tasking VX Toolset Run Debugger

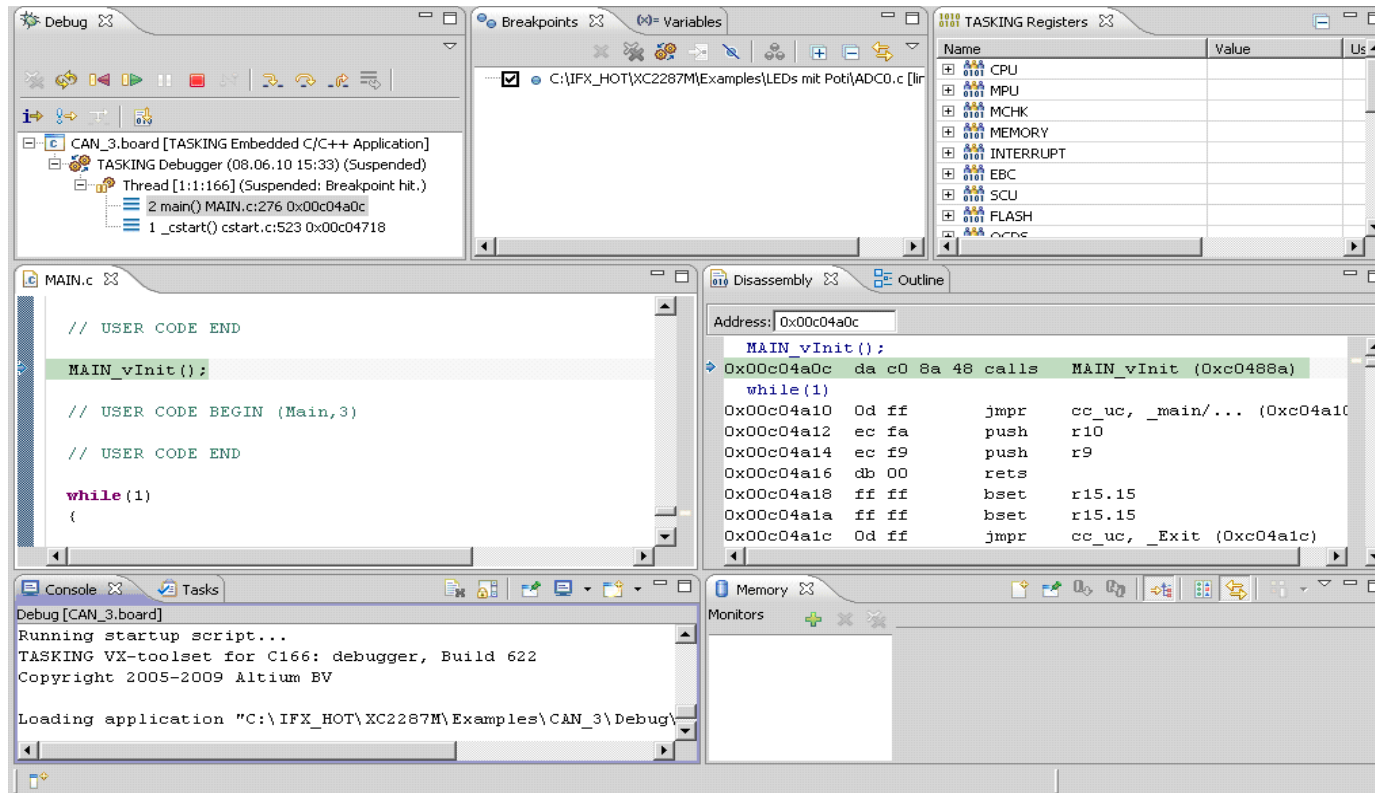
- 1
■ Click on



2



- Click on 'Resume' and start program

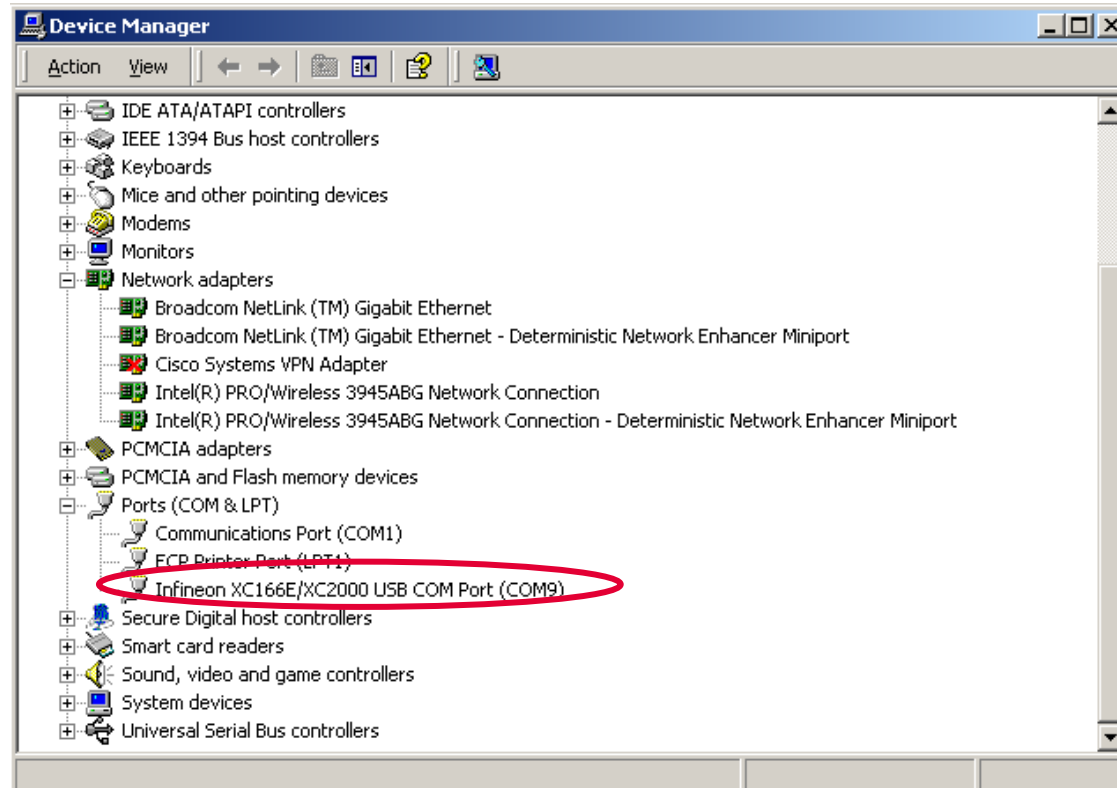


	Resume	F8
	Suspend	
	Terminate	Ctrl+F2
	Step Into	F5
	Step Over	F6
	Step Return	F7

HOT Exercise CAN_3

Start HyperTerminal

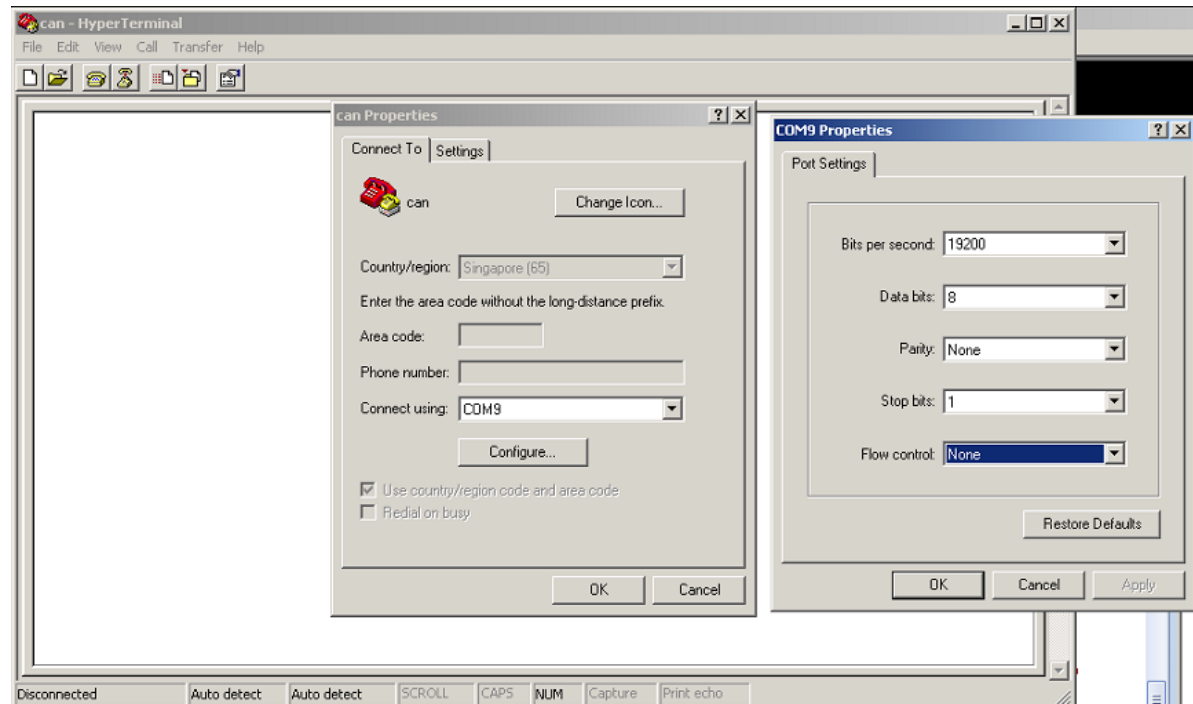
- With the FTDI chip on board, USB interface can be used for UART. FTDI device will convert the USB protocol to the ASCII protocol. Both USB and UART can be used at the same time.
- Open Device Manager and check which COM port is activated for the FTDI chip



HOT Exercise CAN_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!

HOT Exercise CAN_3

See Result

- The yellow LED will toggle when three characters are received in CAN Node1s FIFO

LED blinking



■ 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\

. . .

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

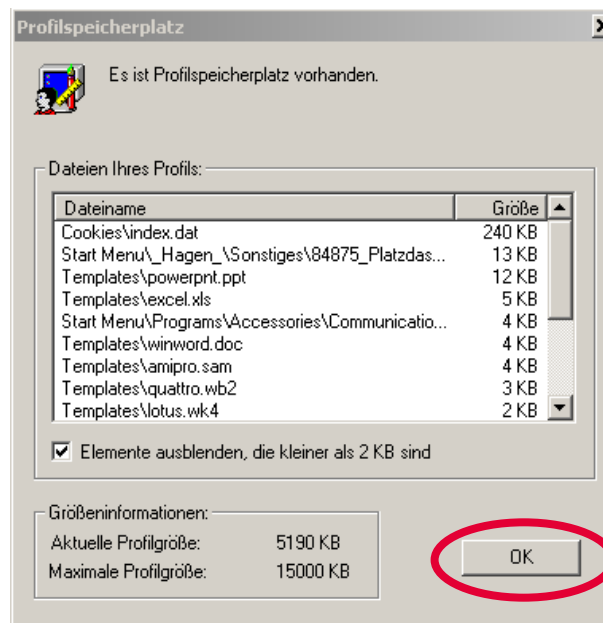
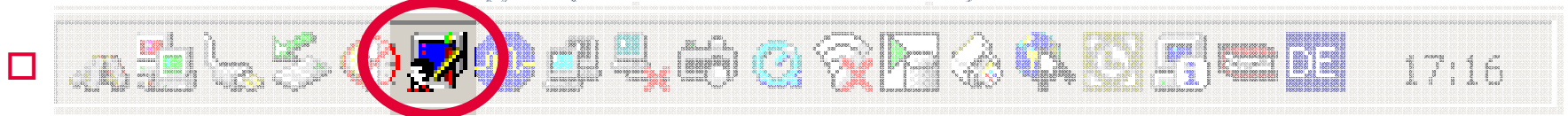
Tasking Viper

"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)



A person in a white lab coat and mask is working in a laboratory setting, holding a device. The background is a blurred laboratory environment with various equipment.

We commit.
We innovate.
We partner.
We create value.



Never stop thinking