

# AP16141

## XE164

UConnect-CAN XE164: "Cookery Book" for a hello world application using Altium's TASKING VX-toolset for C166 (Eclipse™ IDE, Viper compiler)

Microcontrollers



Never stop thinking

**Edition 2008-07-16**

**Published by  
Infineon Technologies AG  
81726 München, Germany**

**© Infineon Technologies AG 2008.  
All Rights Reserved.**

#### **LEGAL DISCLAIMER**

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

#### **Information**

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office ([www.infineon.com](http://www.infineon.com)).

#### **Warnings**

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

---

**AP08048**

**Revision History:** 2008-06 V2.0

Previous Version: none

Page	Subjects (major changes since last revision)

**We Listen to Your Comments**

Any information within this document that you feel is wrong, unclear or missing at all?  
Your feedback will help us to continuously improve the quality of this document.  
Please send your proposal (including a reference to this document) to:

[mcdocu.comments@infineon.com](mailto:mcdocu.comments@infineon.com)



**Note:** Table of Contents [see page 9](#).

### Introduction:

This “Appnote” is a Hands-On-Training / Cookery Book / step-by-step book.  
It will help inexperienced users to get an UConnect-CAN XE164 up and running.

With this step-by-step book you should be able to get your first useful program in less than 2 hours.

The purpose of this document is to gain know-how of the microcontroller and the tool-chain.  
Additionally, the "hello-world-example" can easily be expanded to suit your needs.  
You can connect either a part of - or your entire application to the UConnect-CAN XE164.  
You are also able to benchmark any of your algorithms to find out if the selected microcontroller fulfils all the required functions within the time frame needed.

**Note:**

The style used in this document focuses on working through this material as fast and easily as possible. That means there are full screenshots instead of dialog-window-screenshots; extensive use of colours and page breaks; and listed source-code is not formatted to ease copy & paste.

Have fun and enjoy the UConnect-CAN XE164!



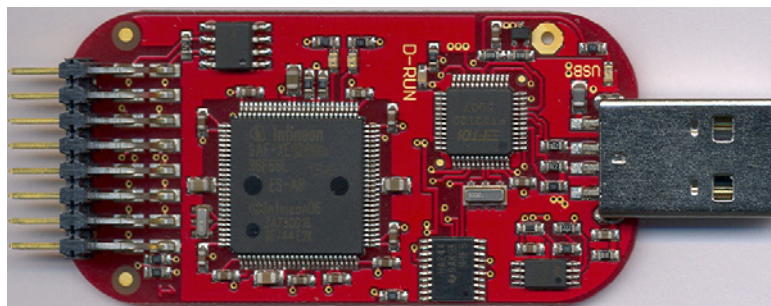
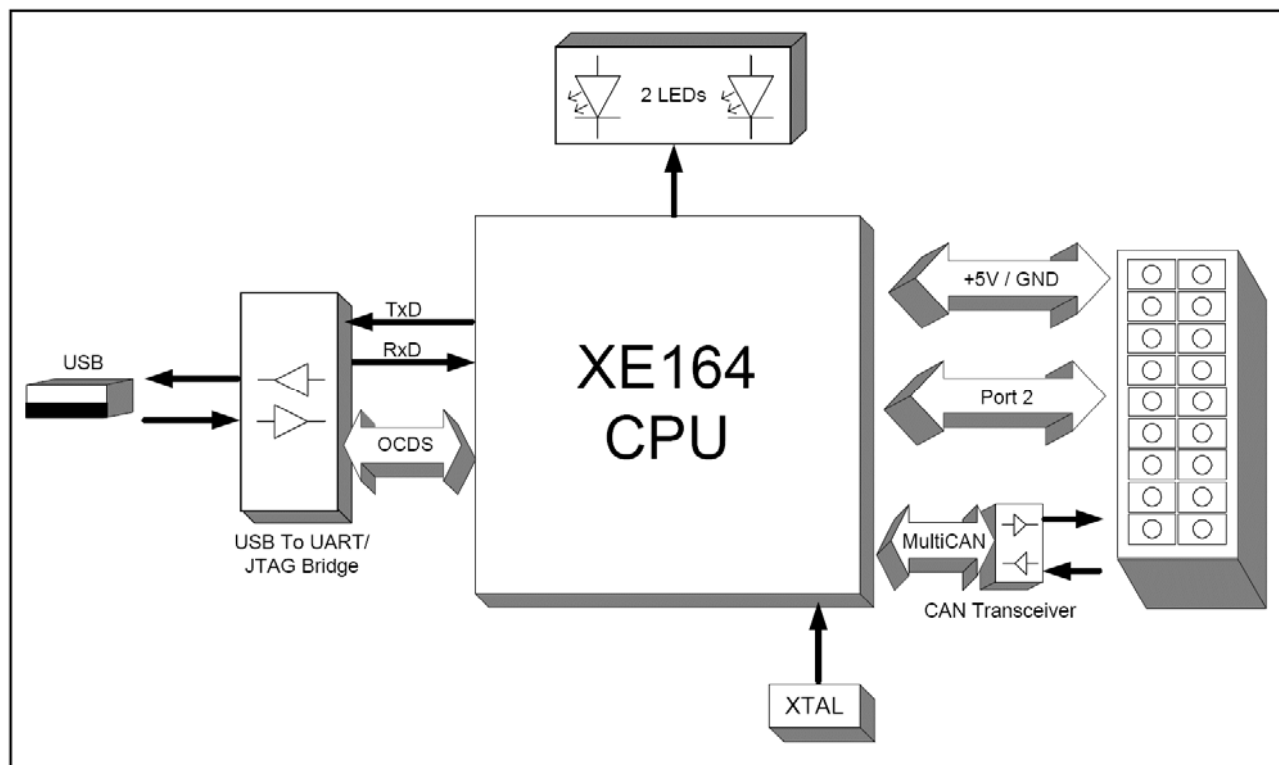


# Programming Example

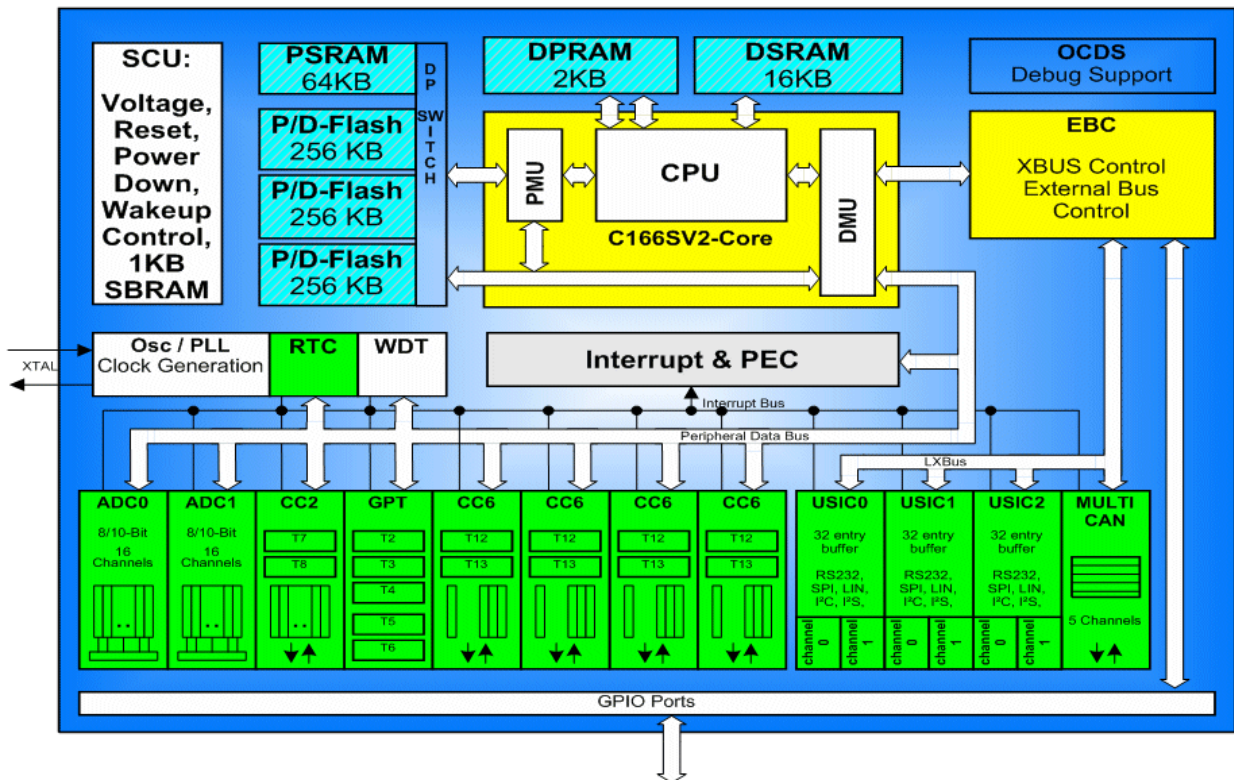
## UConnect-CAN XE164



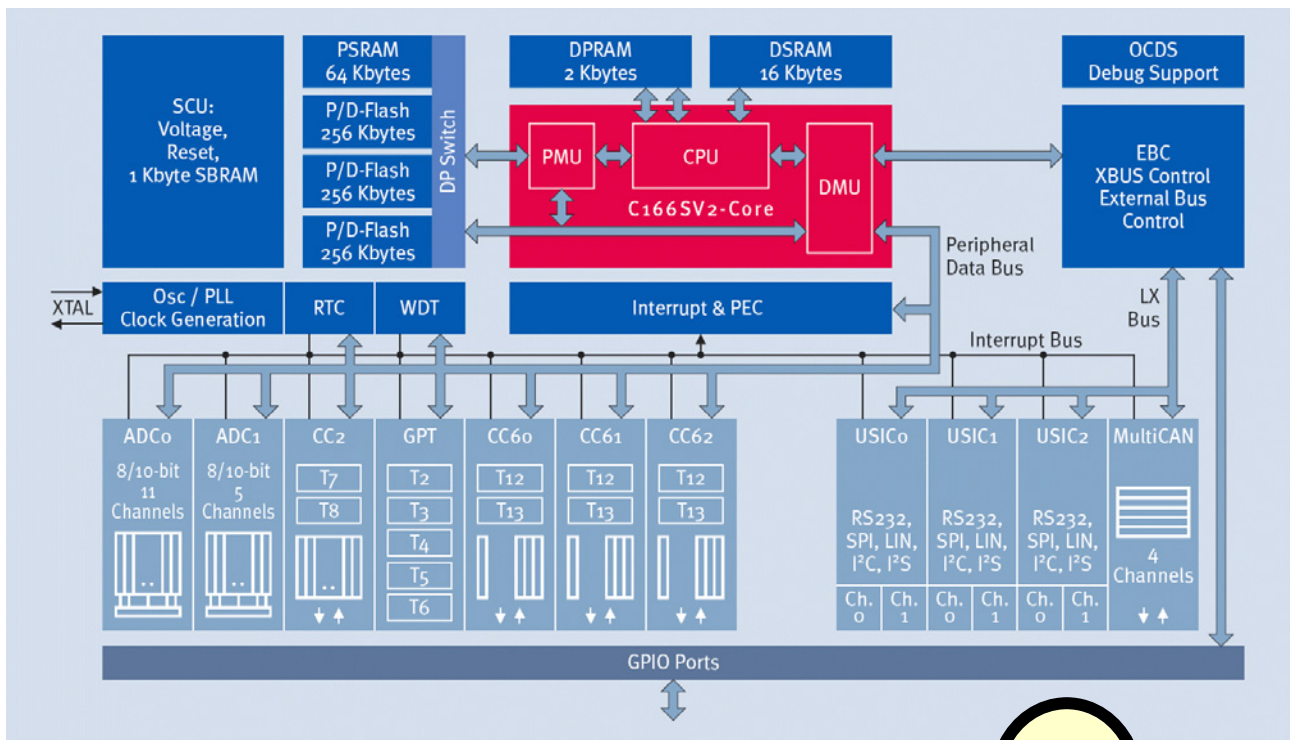
**Block Diagram** (Source: XE164 UConnect Manual)



**SAF-XE167F-96F66L Block Diagram** (Source: Product Marketing)

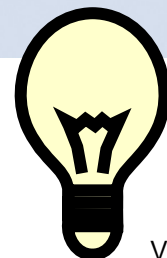


**SAF-XE164F-96F66L Block Diagram** (Source: Product Brief)



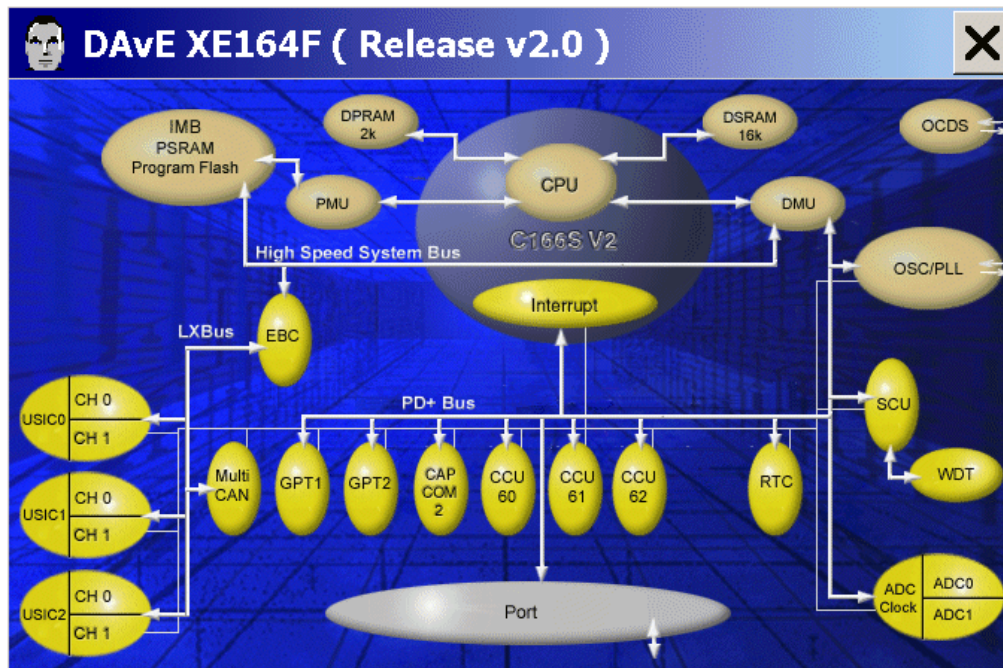
**Note:**

The XE164 microcontroller is a derivative of the XE167 microcontroller!

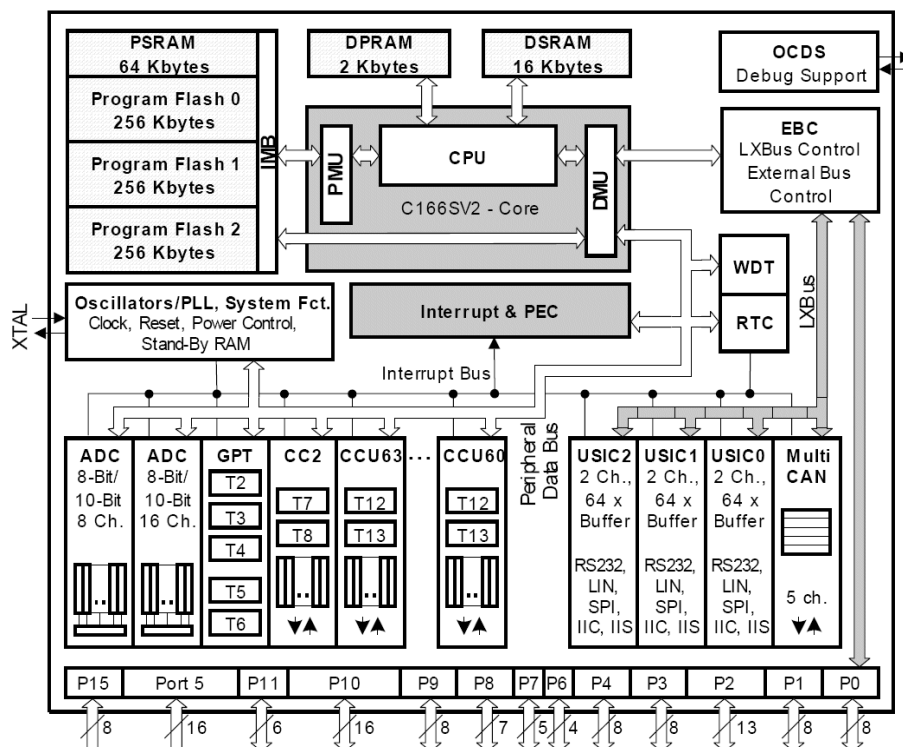




**SAF-XE164F-96F66L Block Diagram (Source: DAVe)**



**XE16x Block Diagram (Source: User's Manual)**



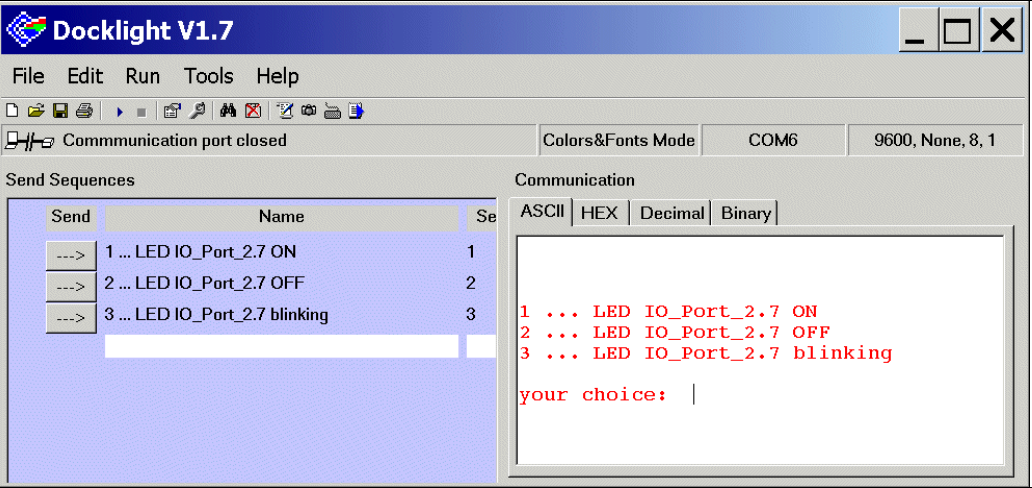
**Note:**

Just by comparing the different sources of block diagrams, you should be able to get a complete picture of the microcontroller and to answer some of your initial questions.



### “Cookery book“

For your first programming example for the UConnect-CAN XE164:

<b>Your program:</b>	
Chapter/ Step	<b>*** Recipes ***</b>
1.)	<a href="#"><u>DAS Installation + Connecting the UConnect-CAN XE164</u></a>
2.)	<a href="#"><u>DAvE (program generator)</u></a> <a href="#"><u>DAvE Installation (mothersystem) + DAVe Update Installation (XE16xx Series.dip) for XE164</u></a>
3.)	<a href="#"><u>Using DAVe</u></a> <a href="#"><u>Microcontroller initialization for your programming example</u></a>
4.)	<a href="#"><u>Using the TASKING Development Tools (C-Compiler)</u></a> <a href="#"><u>Programming of your application (XE164) with Altium's TASKING VX-toolset for C166 (v2.2r5)</u></a>
5.)	<a href="#"><u>Running your first programming example</u></a>

**Hint:**

6.)	<a href="#"><u>Hint: Profile Storage Space</u></a>
-----	--

**Feedback:**

7.)	<a href="#"><u>Feedback</u></a>
-----	---------------------------------

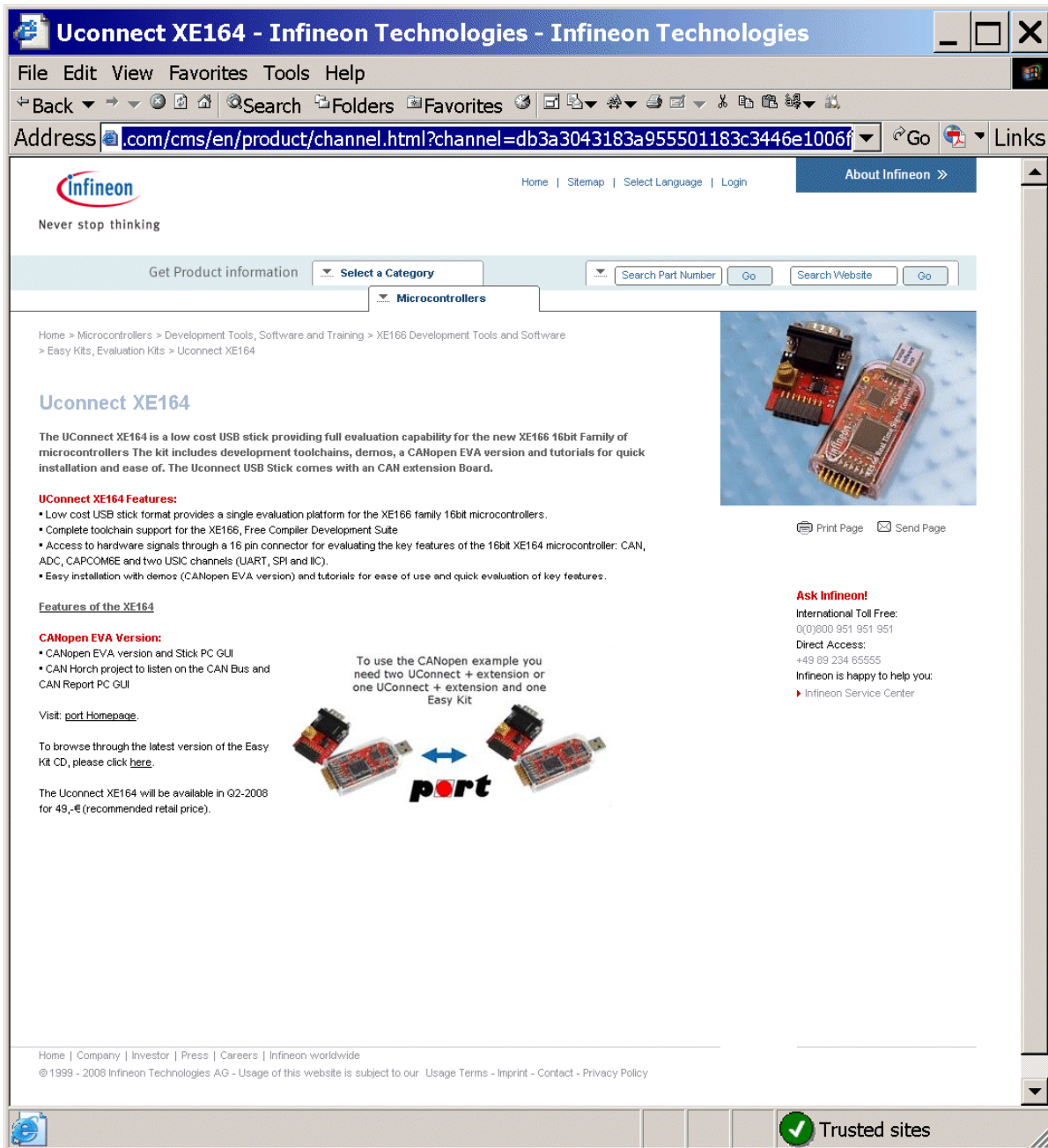


## 1.) DAS Installation + Connecting the UConnect-CAN XE164:



Screenshot of the UConnect-CAN XE164 Homepage:

<http://www.infineon.com/cms/en/product/channel.html?channel=db3a3043183a955501183c3446e1006f>



**Uconnect XE164 - Infineon Technologies - Infineon Technologies**

File Edit View Favorites Tools Help

Back Forward Stop Reload Home Search Folders Favorites Address [www.infineon.com/cms/en/product/channel.html?channel=db3a3043183a955501183c3446e1006f](http://www.infineon.com/cms/en/product/channel.html?channel=db3a3043183a955501183c3446e1006f) Go Links

infineon Never stop thinking Home | Sitemap | Select Language | Login About Infineon >>

Get Product information Select a Category Search Part Number Go Search Website Go

Microcontrollers

Home > Microcontrollers > Development Tools, Software and Training > XE166 Development Tools and Software > Easy Kits, Evaluation Kits > Uconnect XE164

### Uconnect XE164

The UConnect XE164 is a low cost USB stick providing full evaluation capability for the new XE166 16bit Family of microcontrollers. The kit includes development toolchains, demos, a CANopen EVA version and tutorials for quick installation and ease of use. The Uconnect USB Stick comes with an CAN extension Board.

**Uconnect XE164 Features:**

- Low cost USB stick format provides a single evaluation platform for the XE166 family 16bit microcontrollers.
- Complete toolchain support for the XE166, Free Compiler Development Suite
- Access to hardware signals through a 16 pin connector for evaluating the key features of the 16bit XE164 microcontroller: CAN, ADC, CAPCOM6E and two USIC channels (UART, SPI and IIC).
- Easy installation with demos (CANopen EVA version) and tutorials for ease of use and quick evaluation of key features.

**Features of the XE164**

**CANopen EVA Version:**

- CANopen EVA version and Stick PC GUI
- CAN Hirsch project to listen on the CAN Bus and CAN Report PC GUI

Visit: [port Homepage](#).

To browse through the latest version of the Easy Kit CD, please click [here](#).

The Uconnect XE164 will be available in Q2-2008 for 49,-€ (recommended retail price).

To use the CANopen example you need two UConnect + extension or one UConnect + extension and one Easy Kit

Print Page Send Page

**Ask Infineon!**  
International Toll Free: 0(0)800 951 951 951  
Direct Access: +49 89 234 65555  
Infineon is happy to help you:  
▶ Infineon Service Center

Home | Company | Investor | Press | Careers | Infineon worldwide  
© 1999 - 2008 Infineon Technologies AG - Usage of this website is subject to our Usage Terms - Imprint - Contact - Privacy Policy

Trusted sites

### Note:

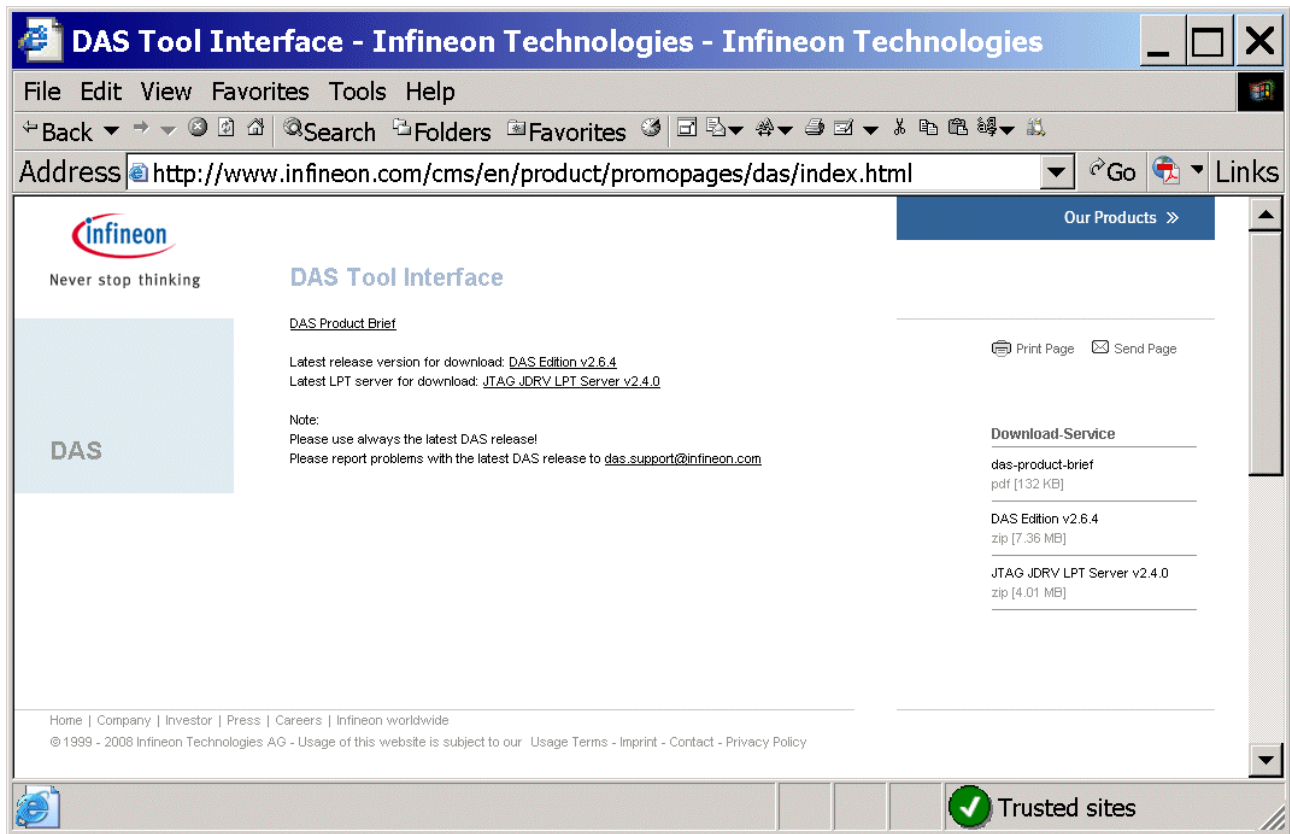
For further information, please refer to the [XE164 UConnect Manual, V.1.0](#) .  
For further information, please refer to the [XE164 UConnect Manual, V.1.1](#) .





Install the Infineon **DAS** (**D**evice **A**ccess **S**erver) Server:

Go to [www.infineon.com/DAS](http://www.infineon.com/DAS):



**Note:**

The DAS Server must be installed on your host computer!

The goal of the DAS software is to provide one single interface for all types of tools.

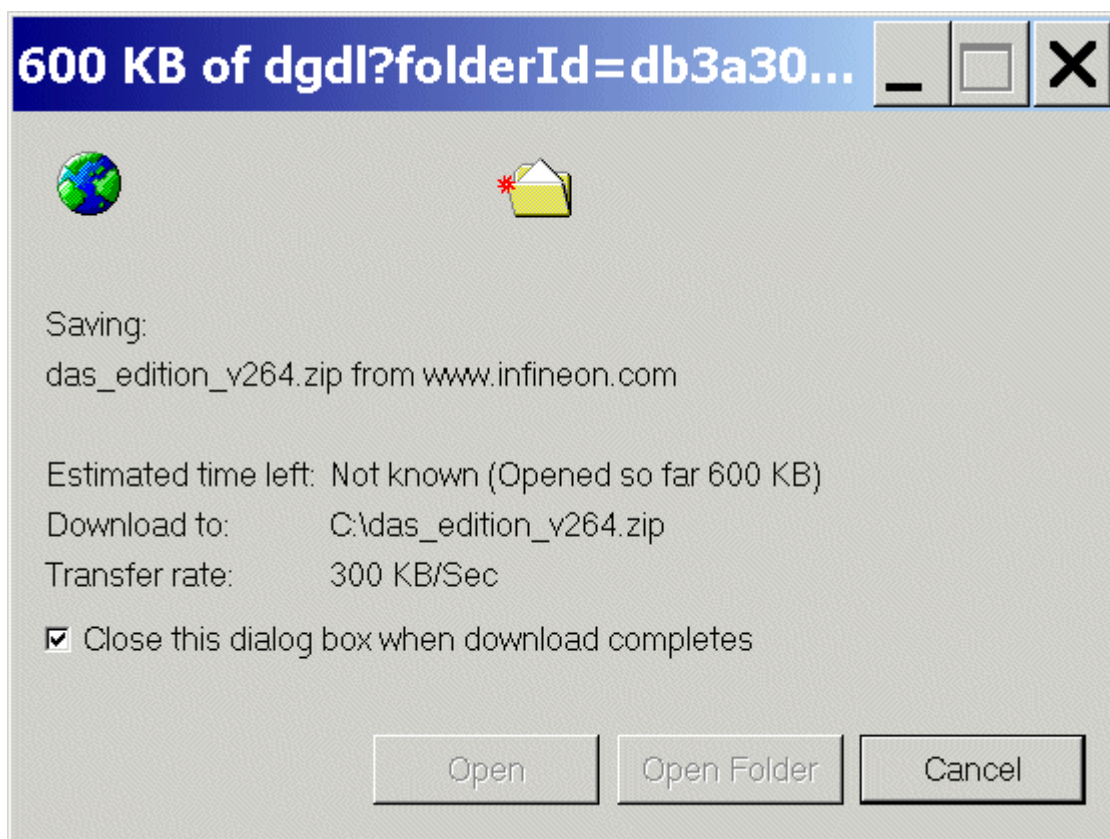
The USB Device driver communicates with the UConnect-CAN XE164 when connected to the host computer.

The USB Device driver for the UConnect-CAN XE164 USB interface is included in the DAS software.

A virtual COM port driver is also included.

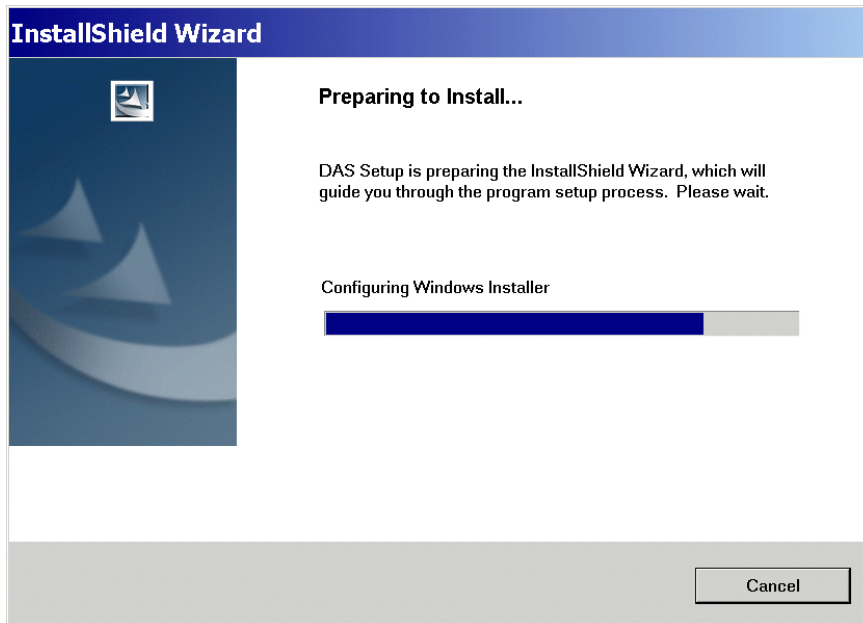


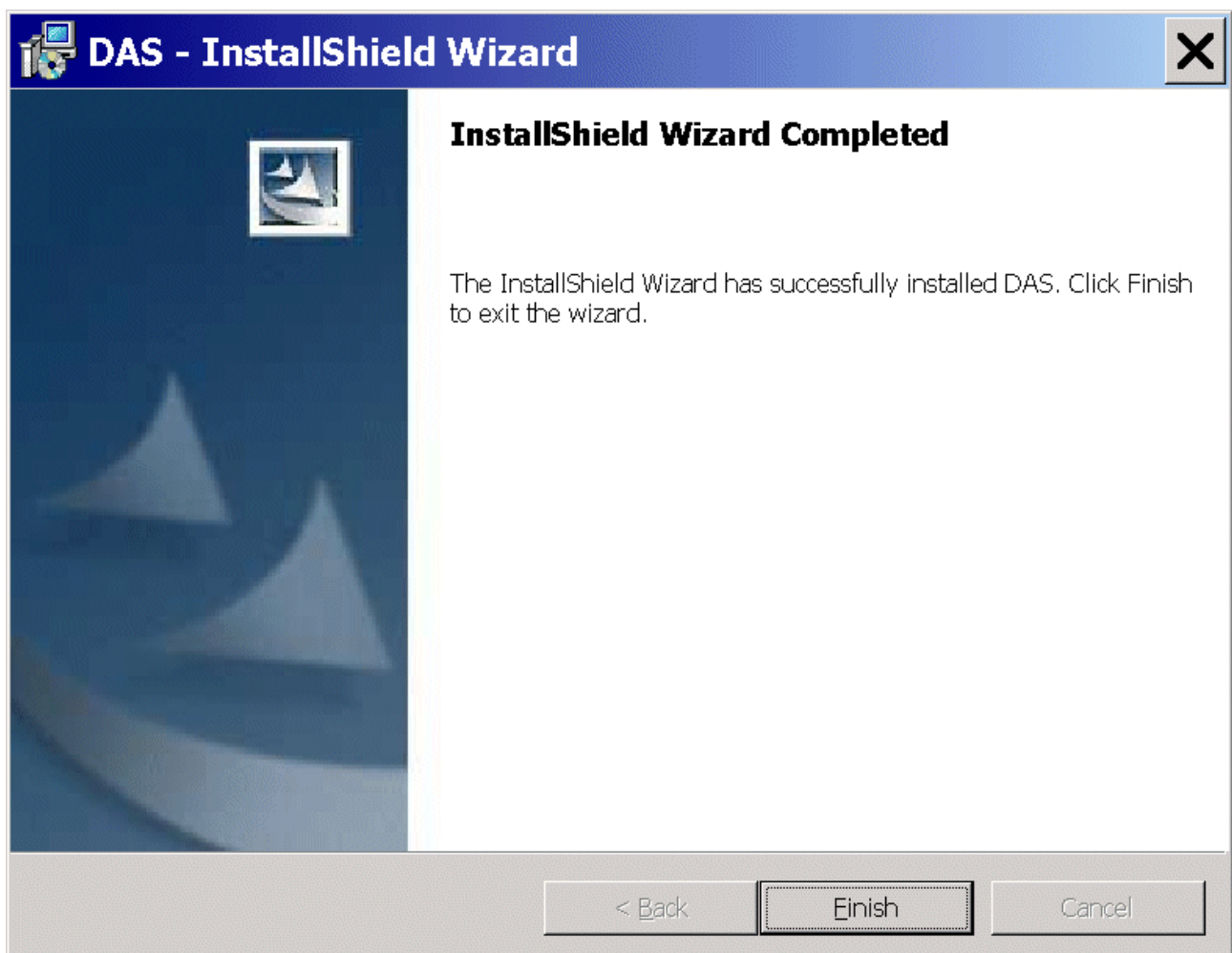
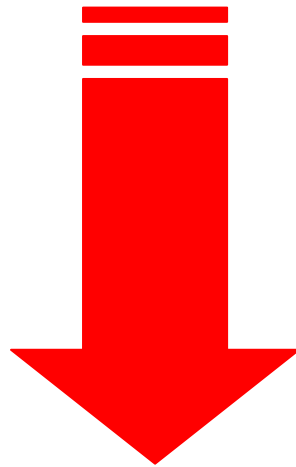
**Download** "The latest release version for download: DAS Edition v2.6.4":



**Unzip** [das\\_edition\\_v264.zip](#) and

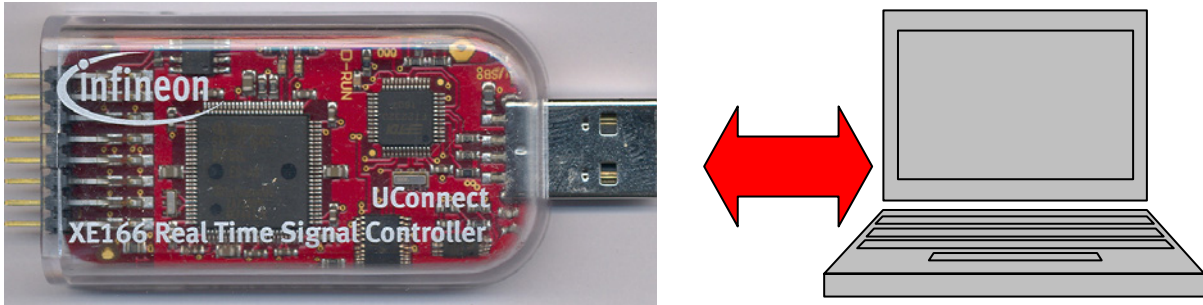
execute “DAS\_v264\_setup.exe” to install the DAS Server.





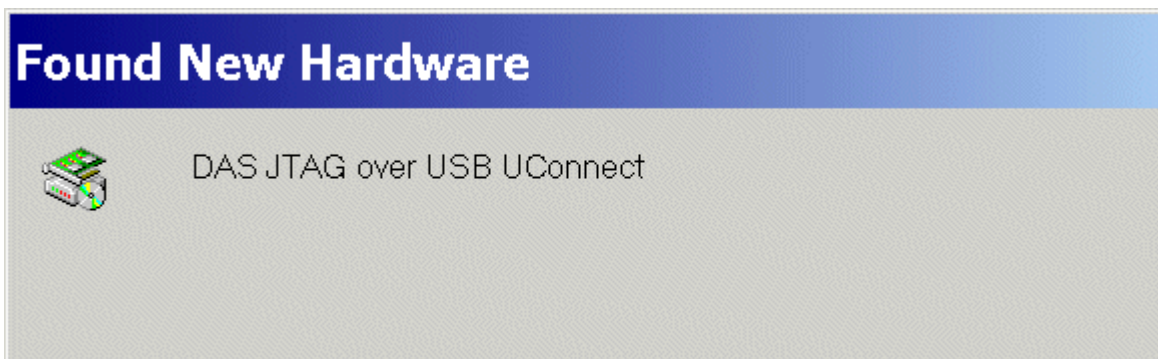
**Click** Finish

Connect the UConnect-CAN XE164 to the host computer:



#### USB Connection:

- .) used for: UART communication (the USIC0\_CH0/UART/RS232/serial interface is available via USB as a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
- .) used for: On-Chip-Flash-Programming and Debugging (first USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
- .) the USB connection works also as the power supply.



#### Note:

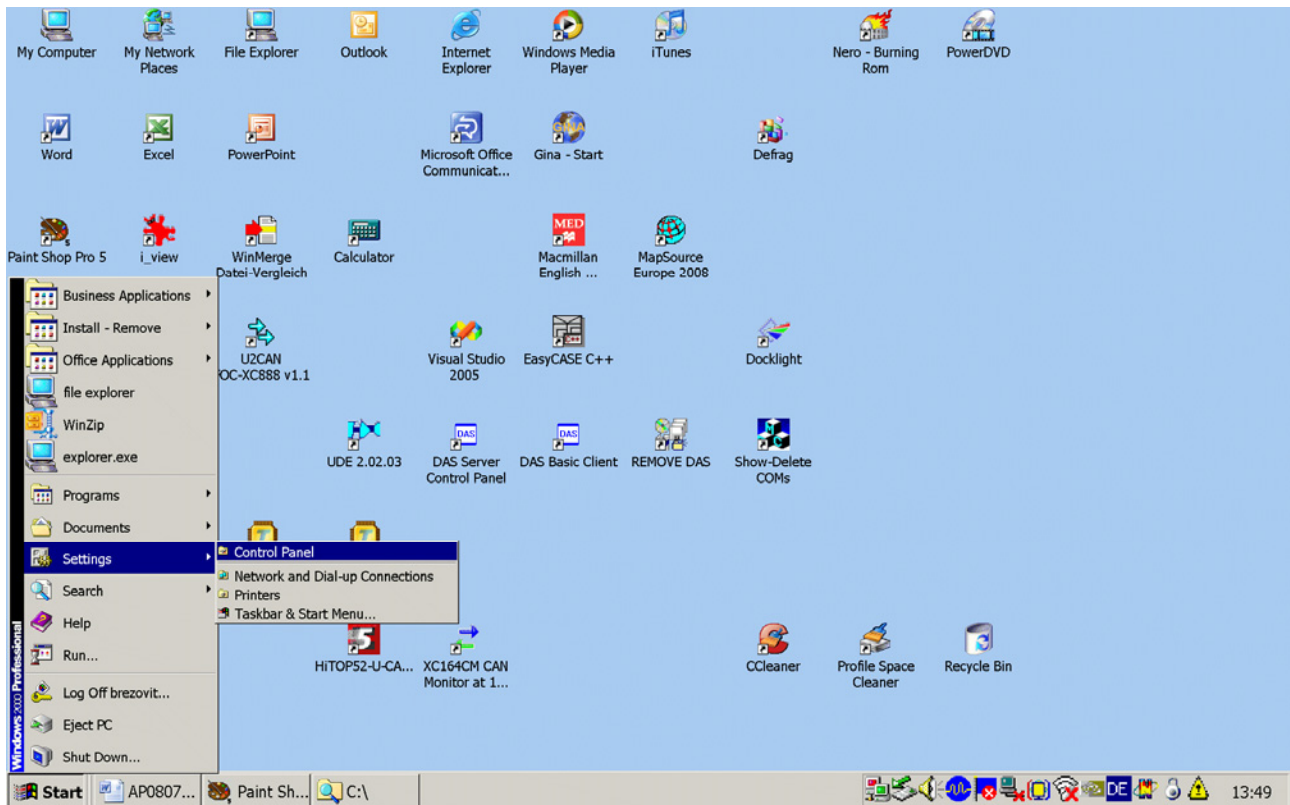
A USB driver is installed the first time while connecting the UConnect-CAN XE164 via USB to your host computer.

#### Note:

A default virtual COM Port is generated.

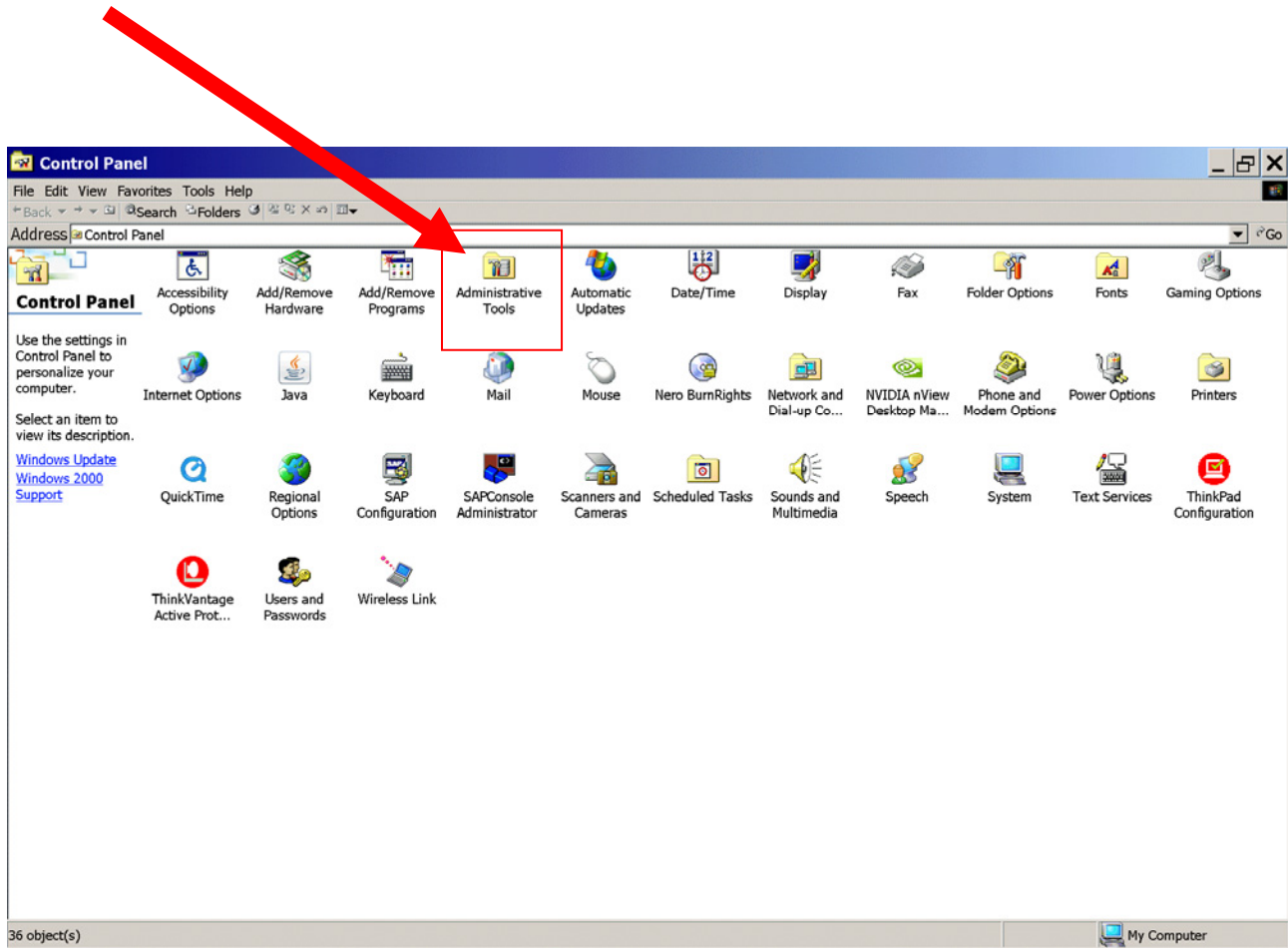
Using a Windows 2000 operating system, we are now going to search for the virtual COM Port which was generated after connecting our UConnect-CAN XE164:

### Start – Settings – Control Panel

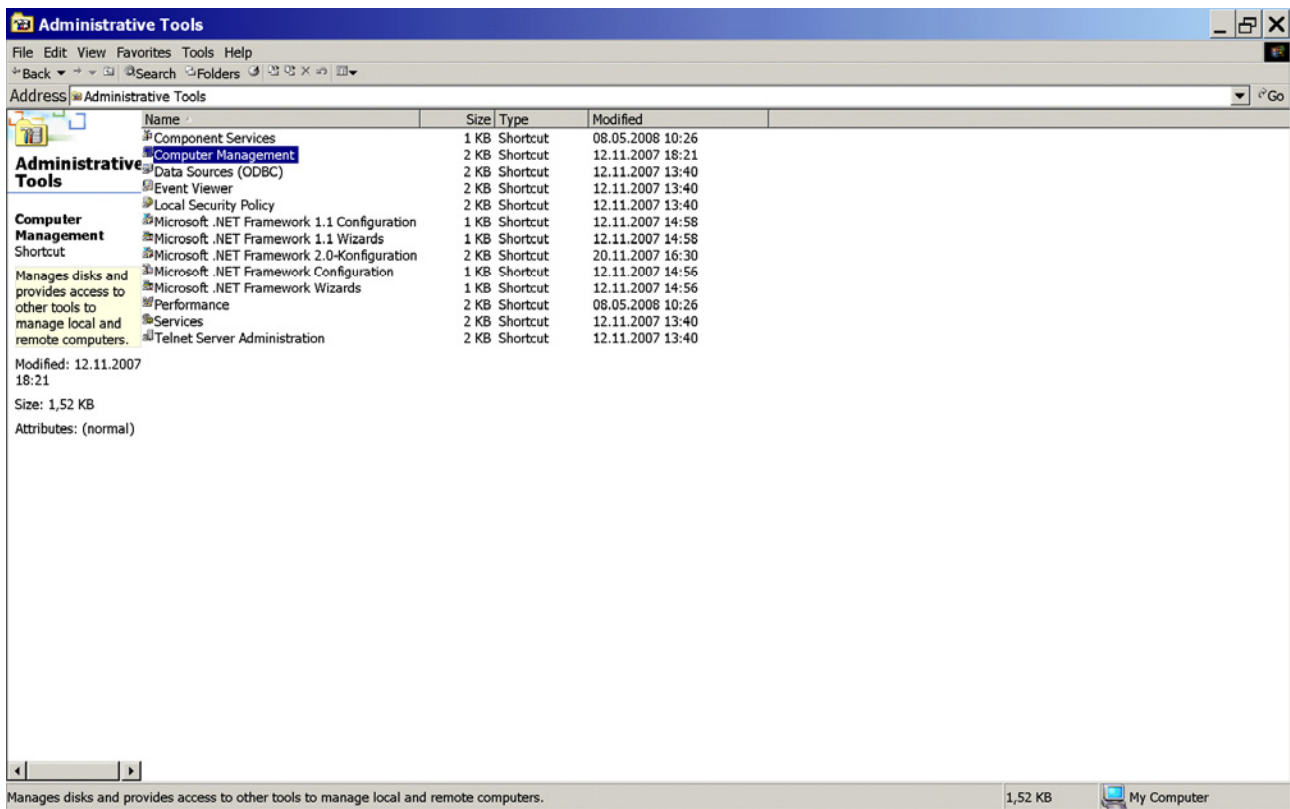




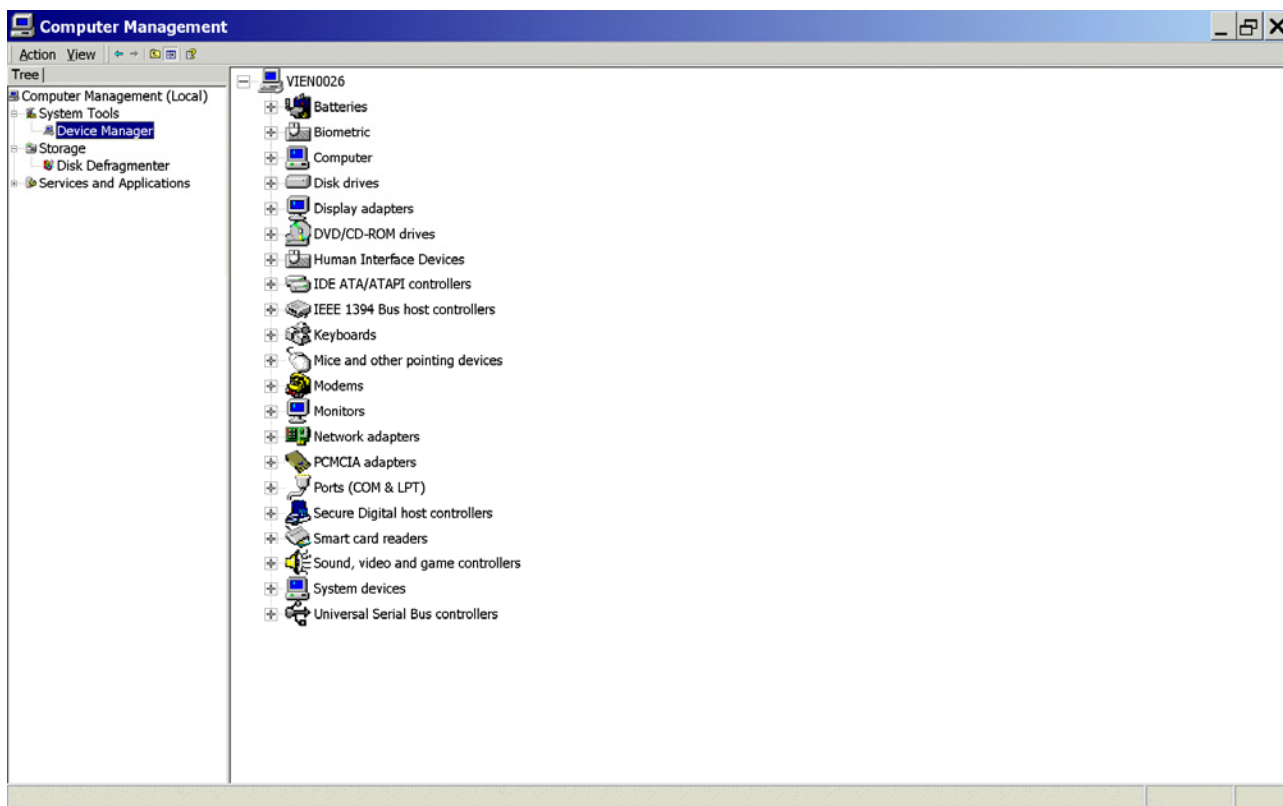
Double click: Administrative Tools



Double click: Computer Management



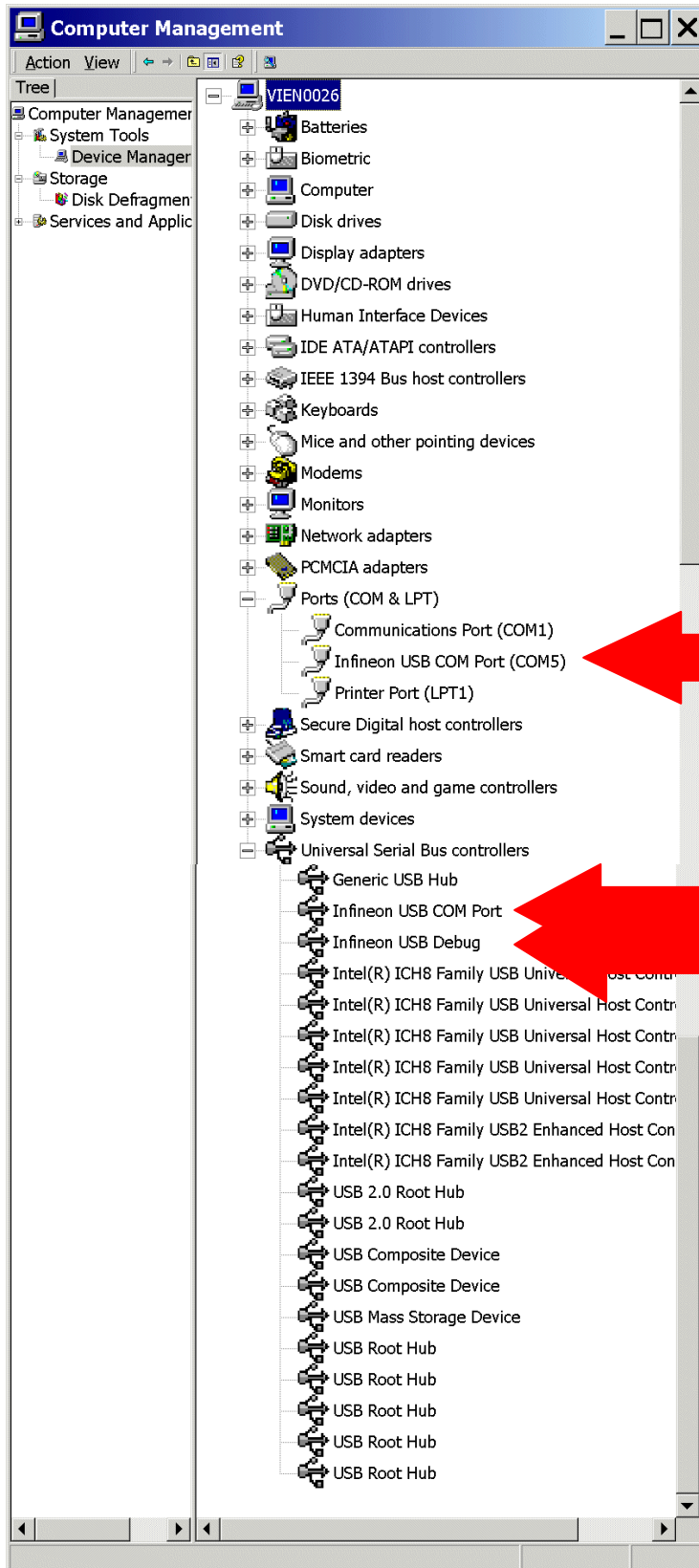
Click: Device Manager





Expand: Ports (COM & LPT):

Expand: Universal Serial Bus controllers:



**Note:**

As we can see:  
our virtual COM Port for  
UART/RS232 communication with the  
UConnect-CAN XE164 via USB is  
**COM5!**

**COM5**

## 2.) DAvE – Installation for XE16x microcontrollers:



### Install DAvE (mothersystem):

Download the DAvE-mothersystem **setup.exe** @ <http://www.infineon.com/DAvE>

Title	Date	Version	Size
<b>Tool Package</b>			
 DAvE - Mothersystem - latest version	05 Feb 2007	V2.1 r24	14.8 MB
 DAvE - Mothersystem	04 Jul 2006	V2.1 r23	15.1 MB

and execute **setup.exe** to install DAvE .

### **Note:**

Abort the installation of Acrobat Reader.



Install the XE164 microcontroller support/update (XE16xx\_Series.dip):

1.)

Download the DAvE-update-file (.DIP) for the required microcontroller


@ <http://www.infineon.com/DAvE>

## DAvE

### DAvE for the Infineon XE166 microcontroller Family

DAvE supports the 16-bit derivatives as DAvE Integration Package (DIP) files.

- All the latest DIPs are available for FREE download.

Company Name and Weblink	Product Name	XE167 Series	XE164 Series	Description
 <a href="#">DAvE home</a>	DAvE	x	x	DAvE stands for Digital Application Virtual Engineer and is Infineon Technologies' code generator for their range of 8, 16 and 32 Bit Microcontrollers. It provides initialization, configuration and driver code to ease programming for beginners as well as experts.

Documents

Contact us

### Document Types


▼ Development Tools

Title	Date	Version	Size
Development Tools ^			
 XE16xx-Series DIP file for DAvE (Microcontroller Configuration Tool) (XE16xx_Series_v2.0.zip)	20 May 2008	v2.0	4.2 MB

**Unzip** the zip-file “[XE16xx\\_Series\\_v2\[1\].0.zip](#)” and save “[XE16xx\\_Series.dip](#)”

@ e.g. C:\DAvE\XE16x-2008-05-29\XE16xx\_Series.dip.

2.)

Start DAVe - (  click DAVe )

3.)

View

Setup Wizard

Default: • Installation

Forward>

Select: • I want to install products from the DAVe's web site

Forward>

Select: C:\DAVe\XE16x-2008-05-29

Forward>

Select: Available Products

click ✓ XE16xx\_Series

Forward>

Install

End

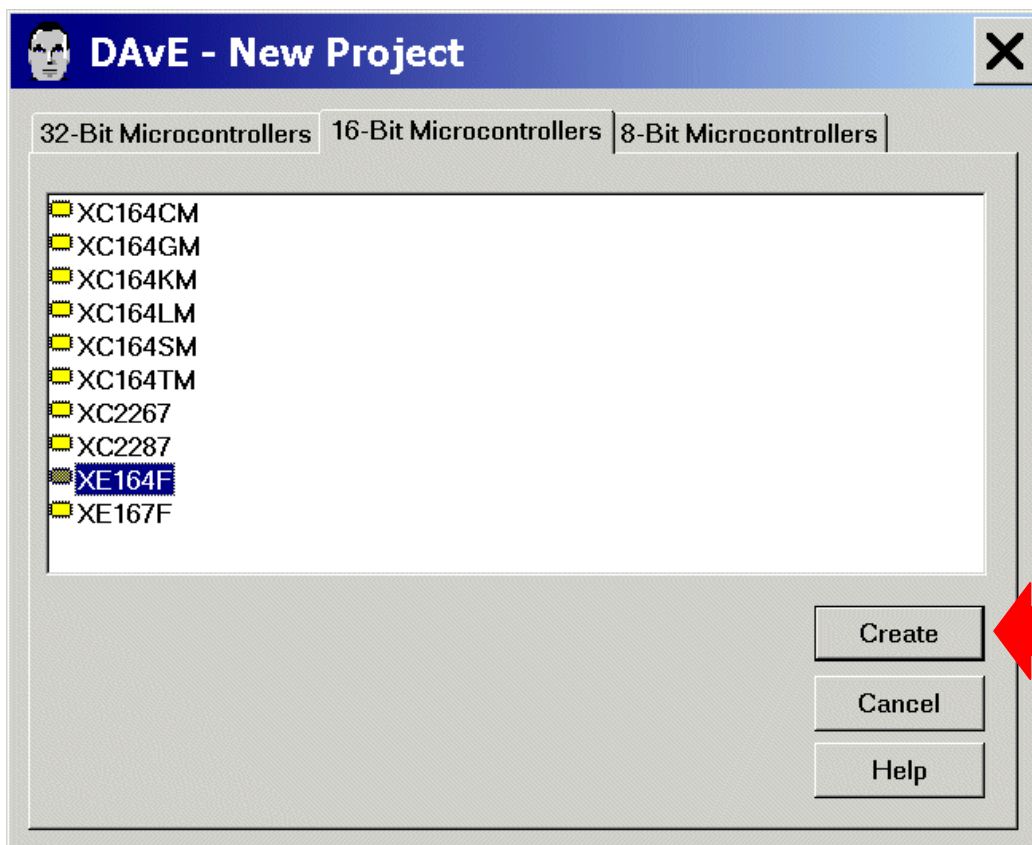
4.) DAVe is now ready to generate code for the XE16x microcontrollers.

### 3.) DAvE - Microcontroller Initialization after Power-On:



Start the program generator DAvE and select the XE164 microcontroller:

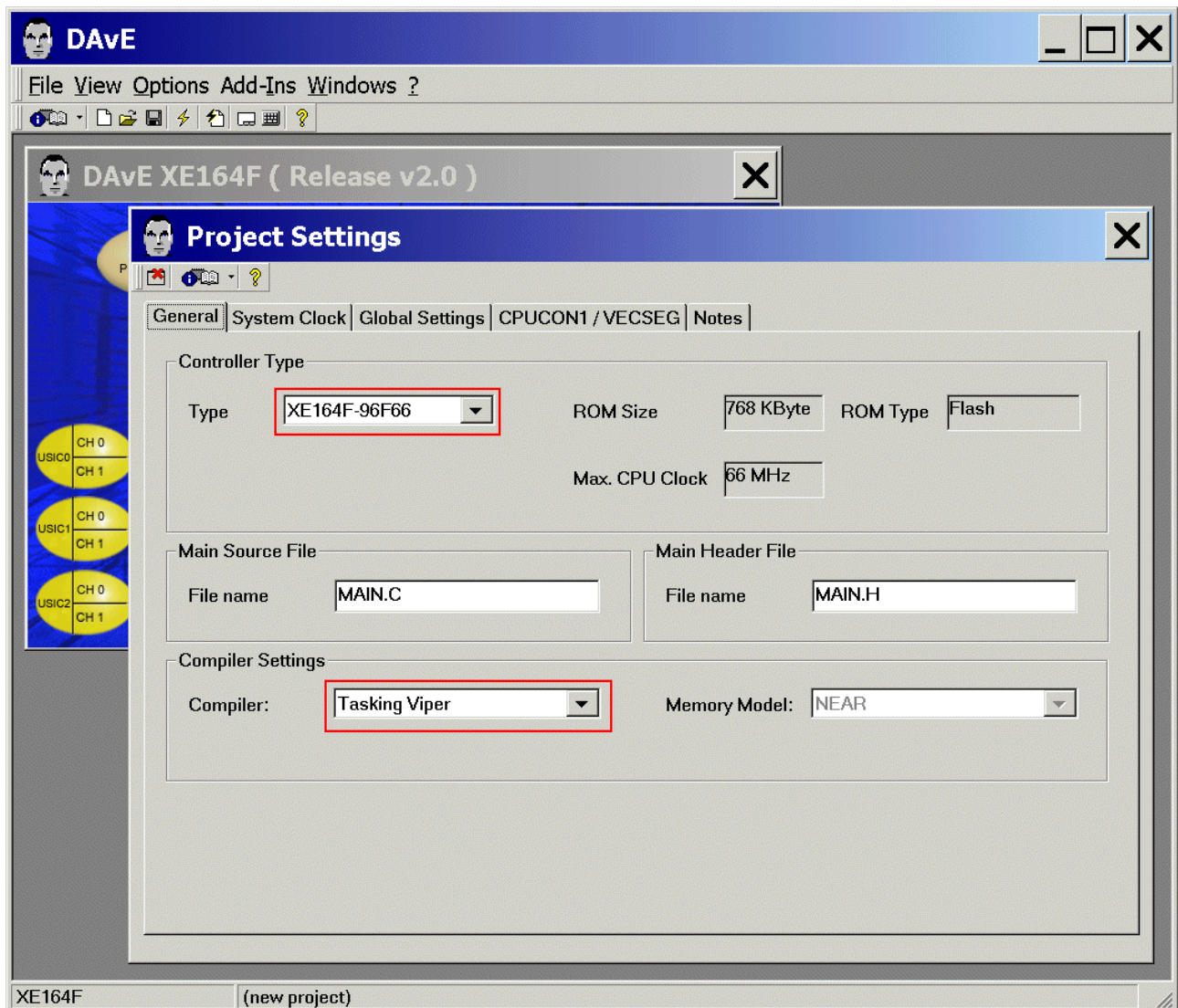
File  
New  
16-Bit Microcontrollers  
Select XE164F  
Create





Choose the Project Settings as you can see in the following screenshots:

General: Controller Type: Type: **check/select** XE164F-96F66  
General: Compiler Settings: **Compiler:** **select** Tasking Viper

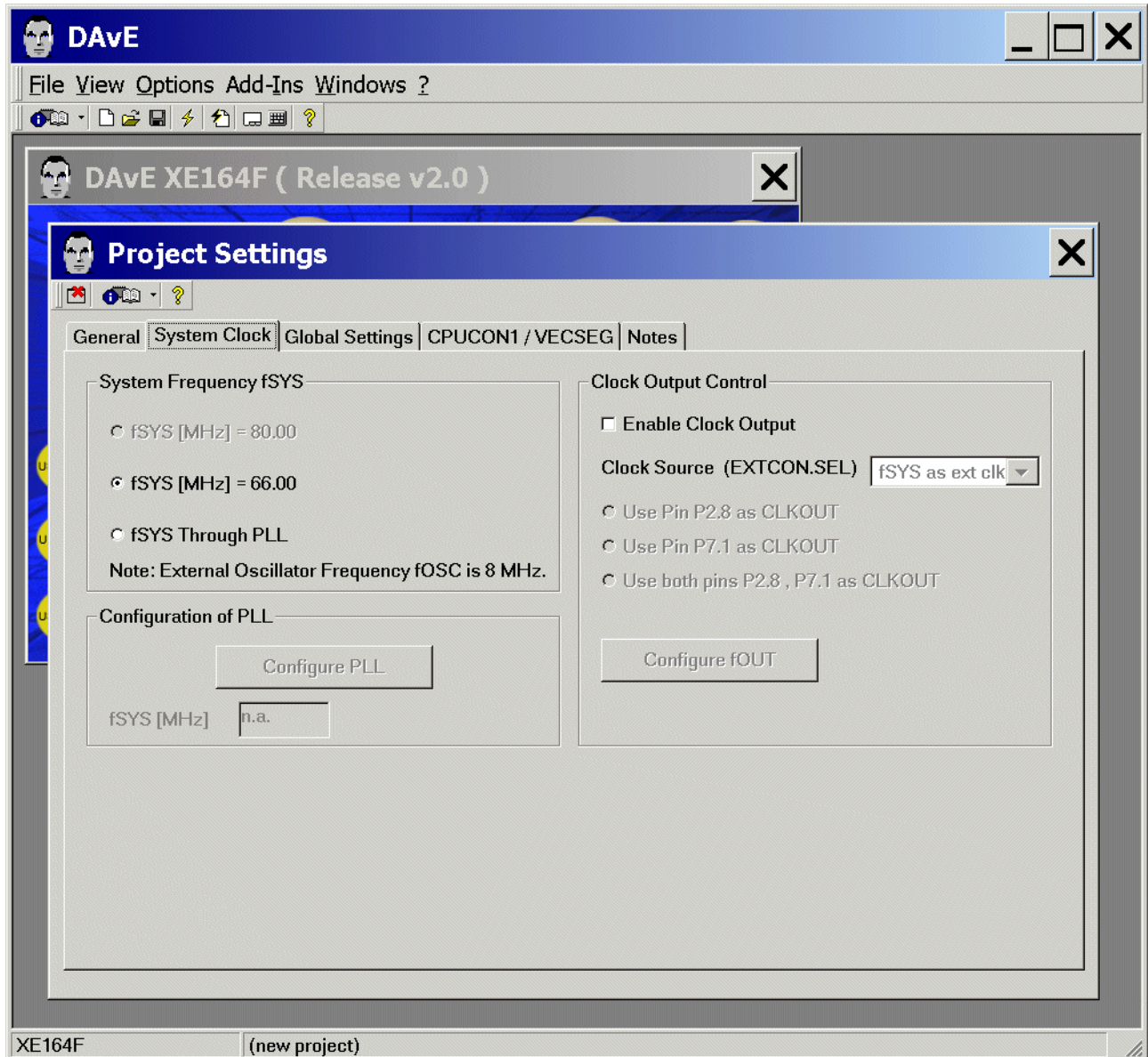


**Note:**

You can change file names (e.g. MAIN.C, MAIN.H) anytime.



System Clock: (do nothing)

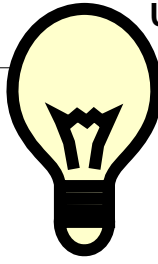


**Note (Source: DAVE):**

Configuration of the System Clock:

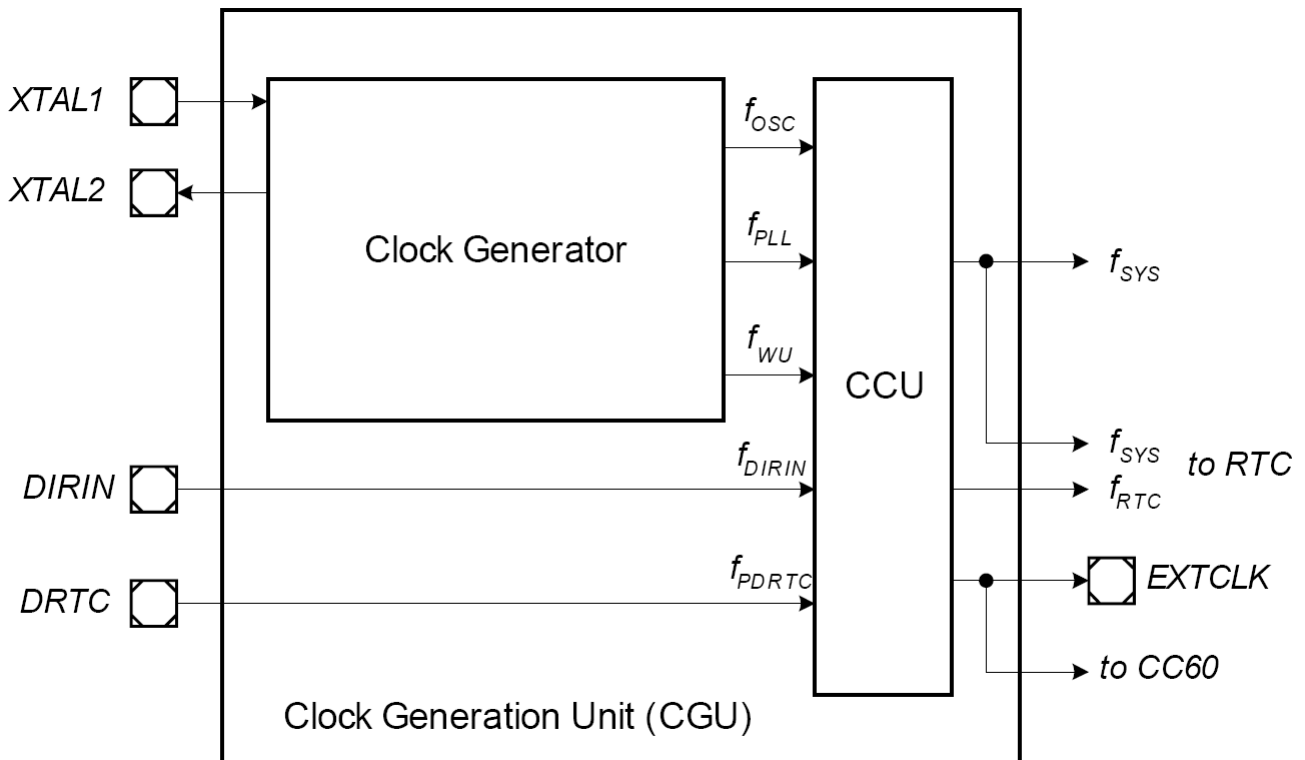
- VCO clock used, input clock is connected
- input frequency is 8,00 MHz (XTAL1)
- configured system frequency is 66,00 MHz
- system clock is 66.00 MHz





Additional information: Clock System (Source: User's Manual):

Clock Generation Unit (CGU) Block Diagram:



**Note:**

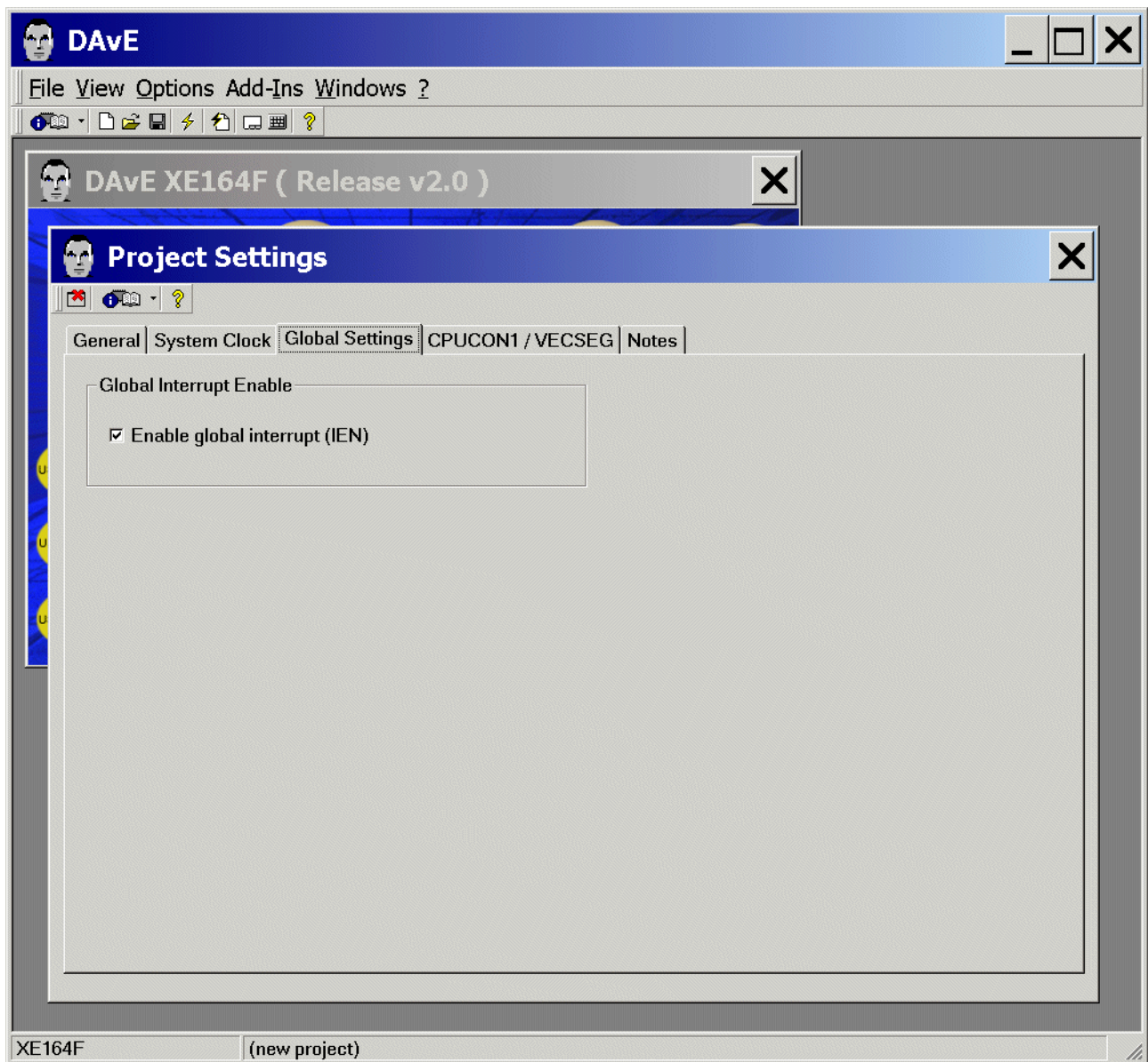
The CGU can convert a low-frequency external clock to a high-speed internal clock, or can create a high-speed internal clock without external input.

The system clock  $f_{SYS}$  is generated out of four selectable clocks:

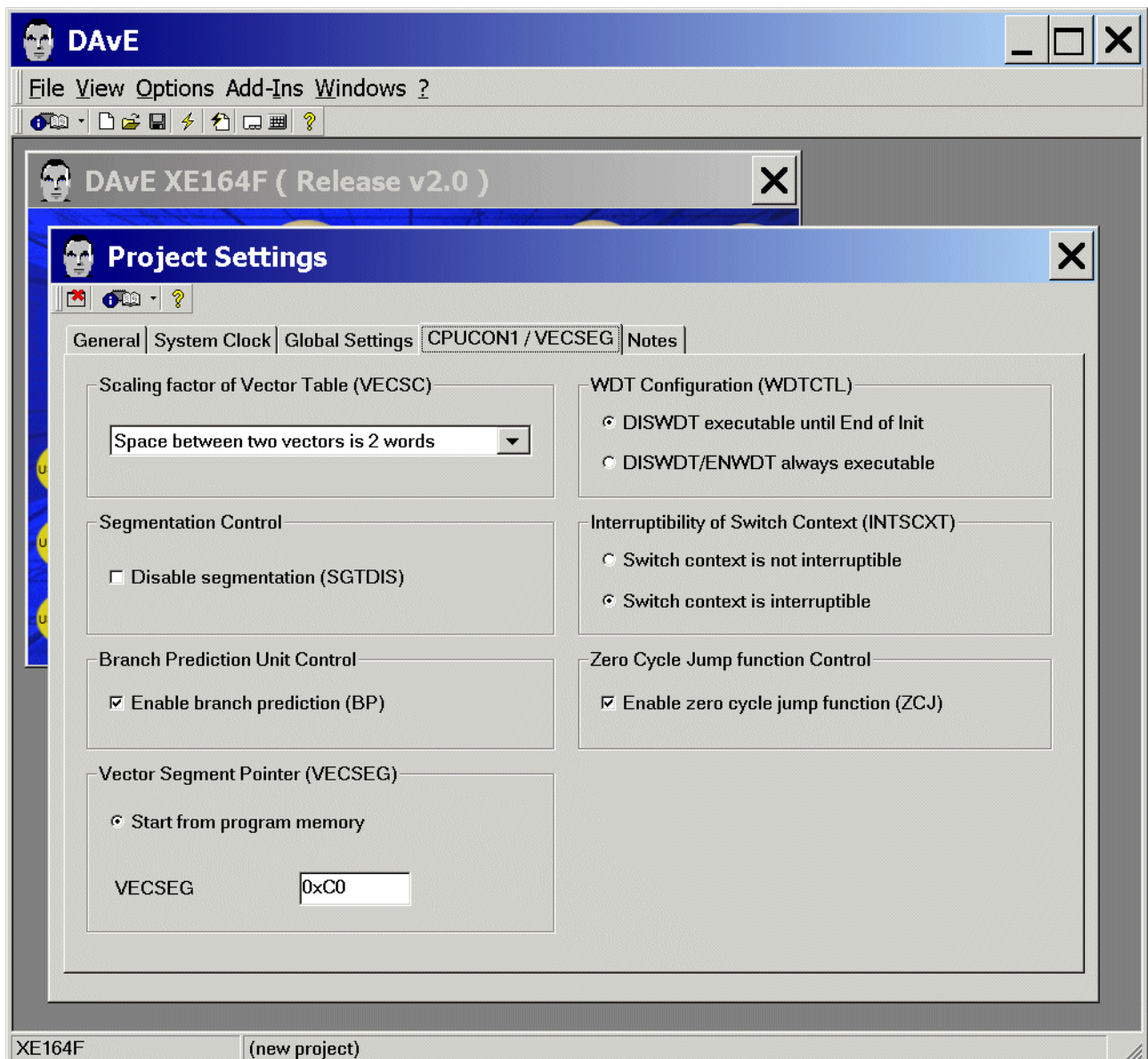
- PLL clock  $f_{PLL}$
- Wake-Up clock  $f_{WU}$
- The Direct Clock  $f_{OSC}$ , from pin XTAL1
- Input DIRIN as Direct Clock Input  $f_{DIR}$



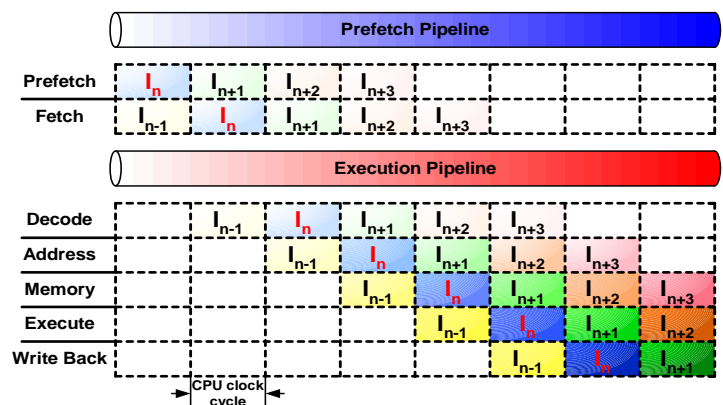
**Global Settings:** (do nothing. Do not change configuration)



CPUCON1/VECSEG: (do nothing)



**Note:**  
We should not change the pipeline behaviour.




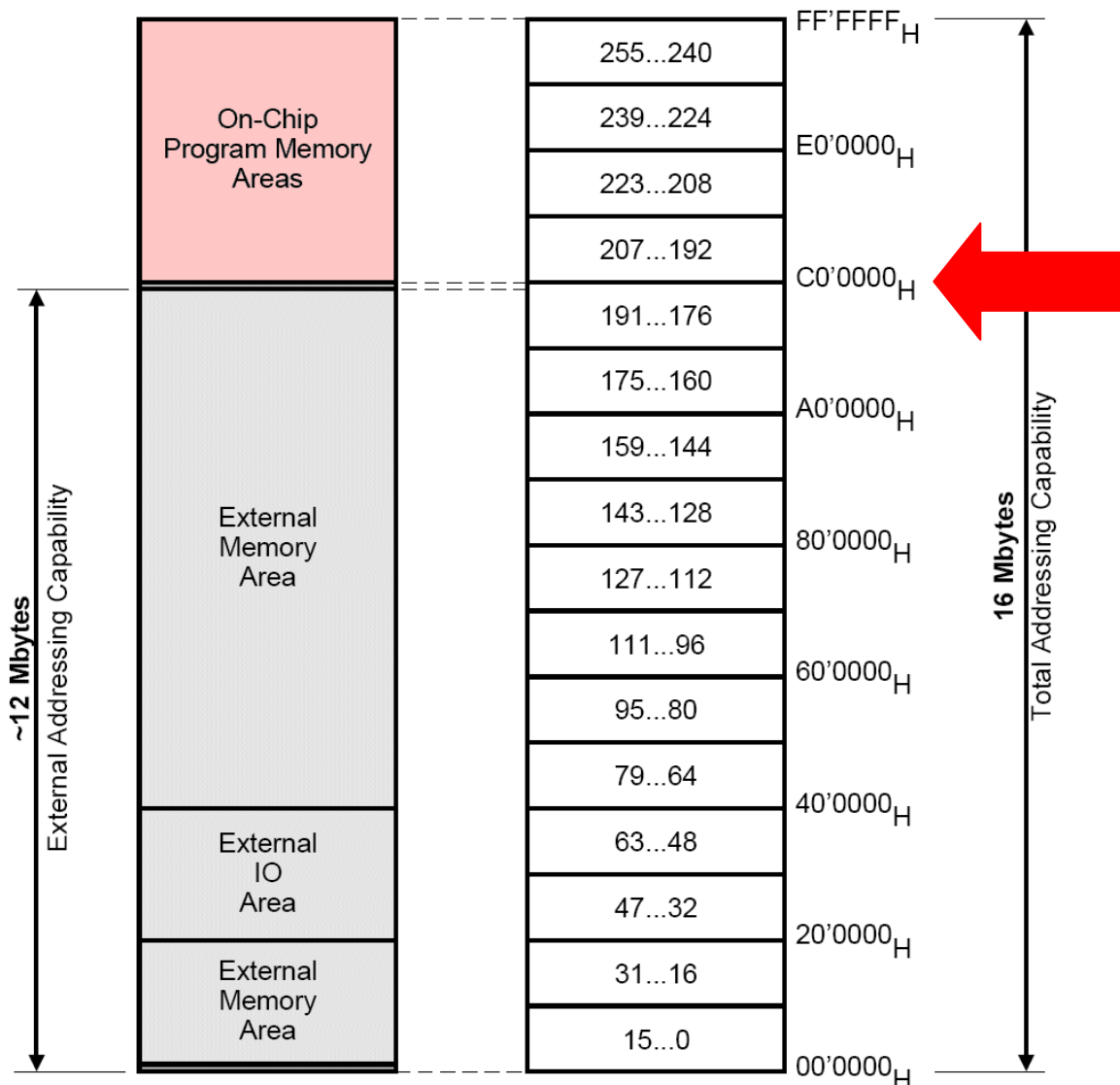


Additional information: Start from program memory (Source: User's Manual):

Vector Segment Pointer (VECSEG)


☒ Start from program memory

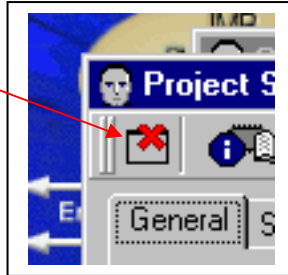
VECSEG  



Total Address Space  
16 Mbytes, Segments 255...0

**Notes:** If you wish, you can insert your comments here.

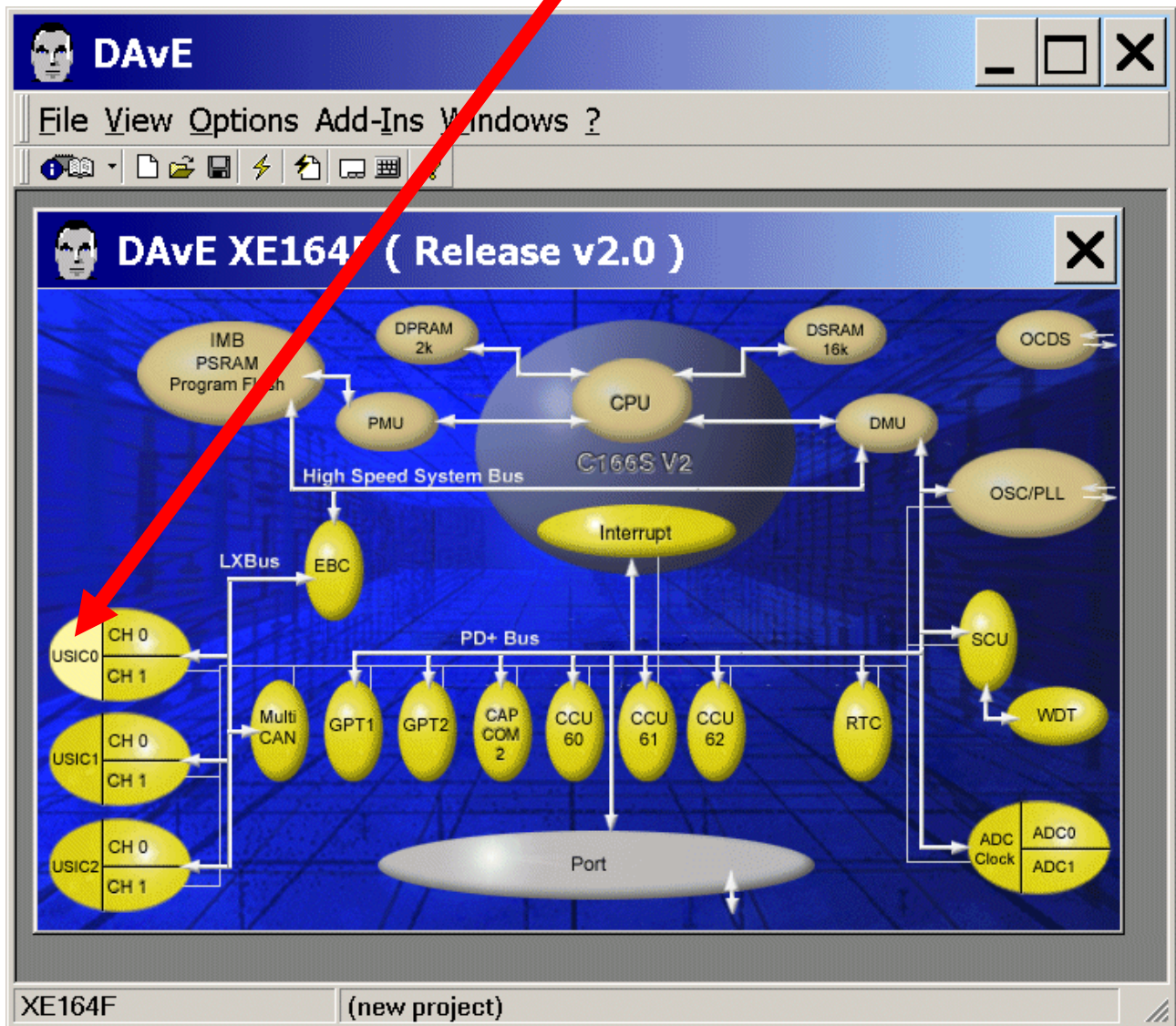
**Exit** and **Save** this dialog now by clicking  the close button:



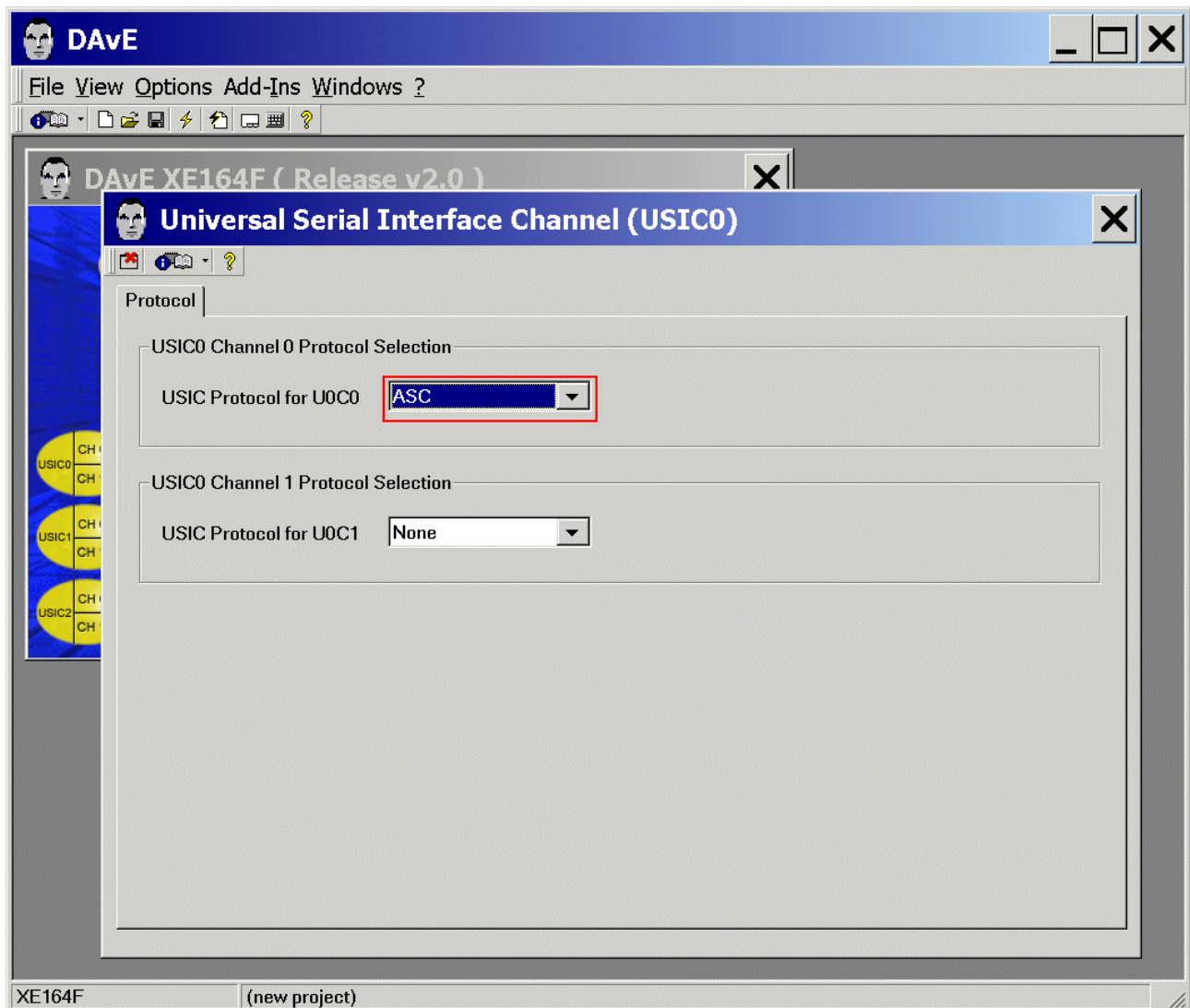



Configuration of the serial interface "ASC0" / UART / USIC0 CH0 / U0C0:

The configuration window/dialog can be opened by clicking the specific block/module (USIC0).



Protocol: USIC0 Channel 0 Protocol Selection: USIC Protocol for U0C0: select ASC

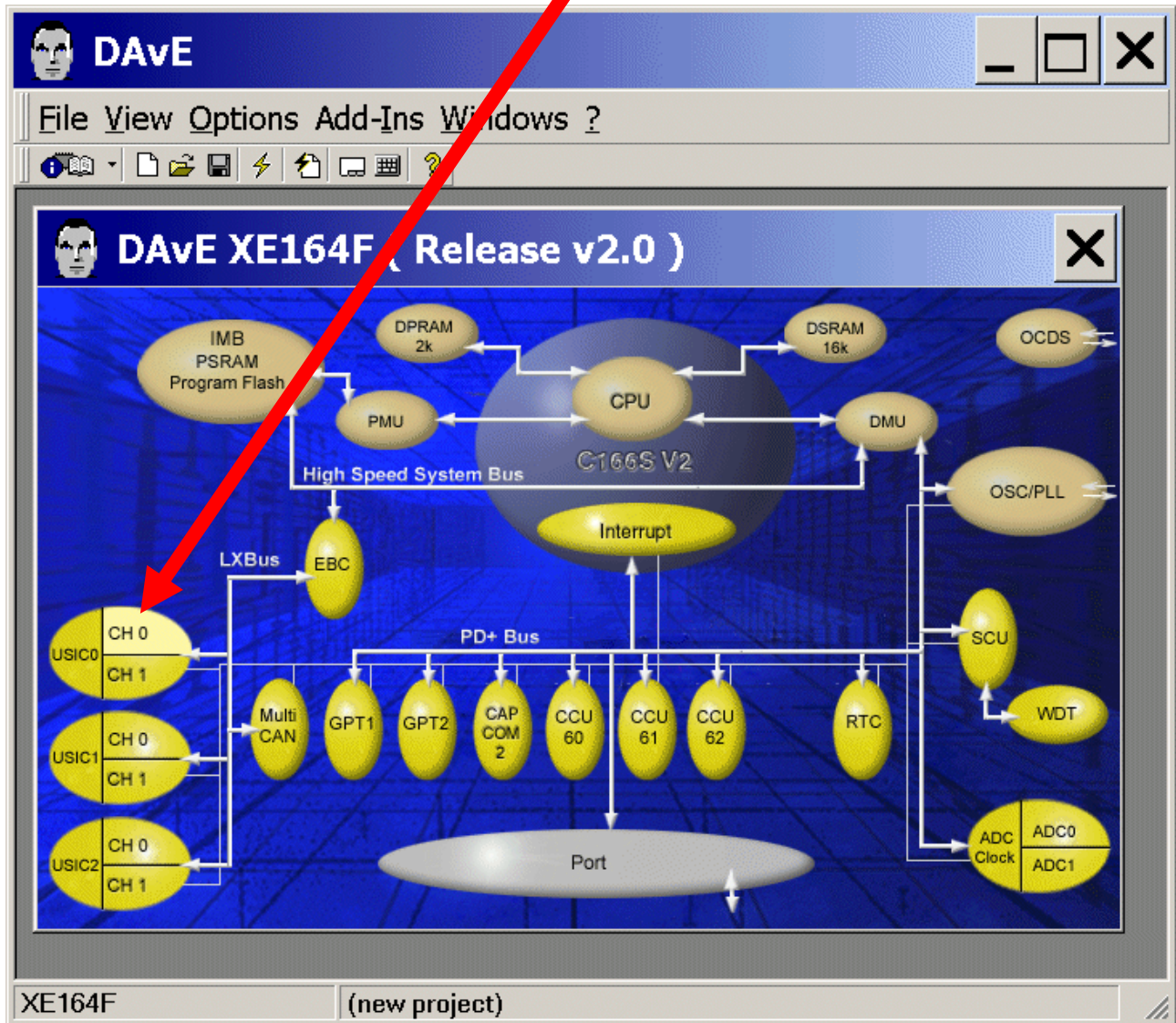


Exit and Save this dialog now by clicking  the close button.

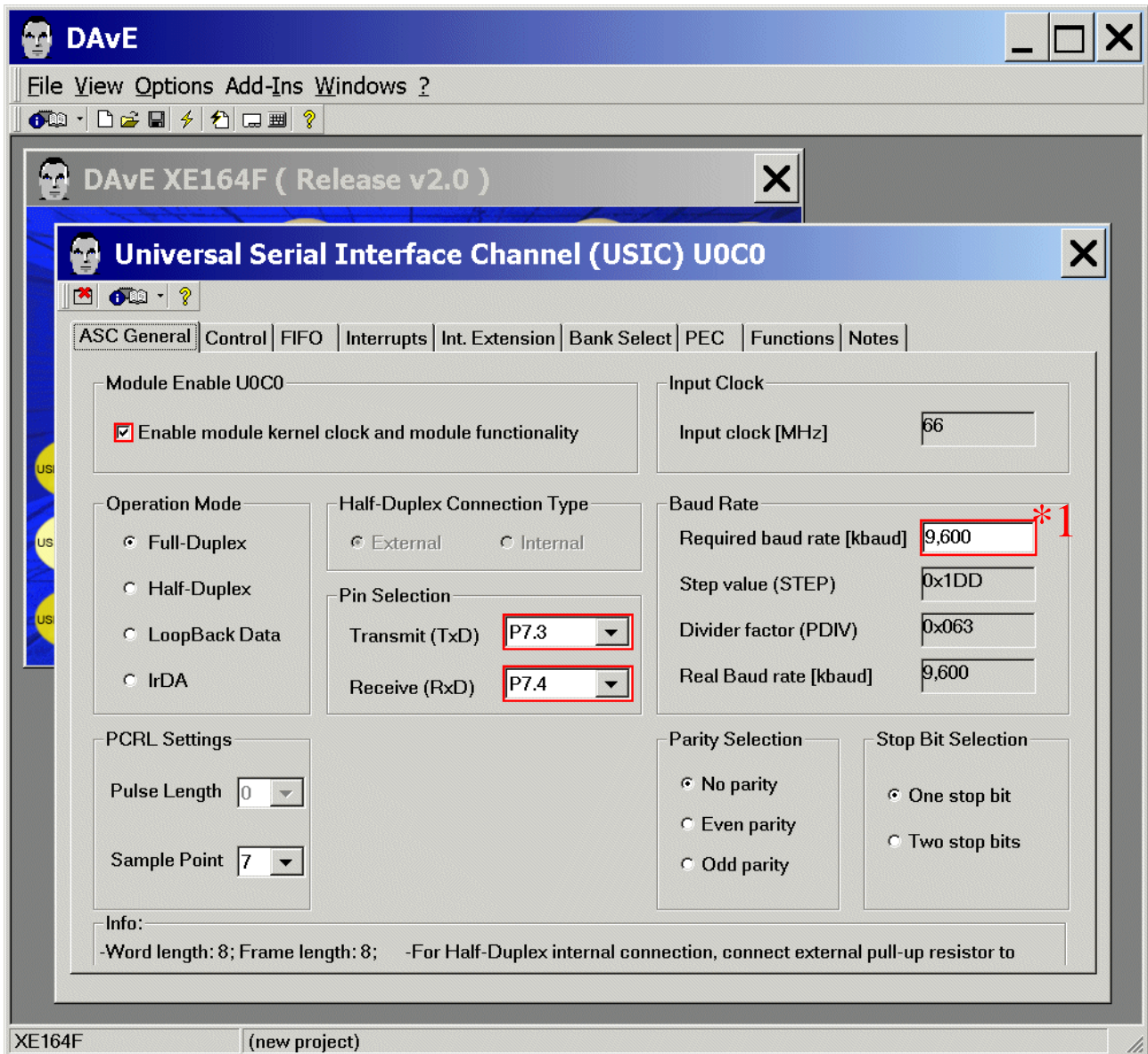


Configuration of the serial interface USIC0\_CH0 / U0C0:

The configuration window/dialog can be opened by clicking the specific block/module (CH 0).



ASC General: Module Enable U0C0: **click** ☒ Enable module kernel clock and module functionality  
 ASC General: Pin Selection: Transmit (TxD): **select** P7.3  
 ASC General: Pin Selection: Receive (RxD): **select** P7.4  
 ASC General: Baud Rate: Required baud rate [kbaud]: **insert** 9,600 <ENTER>



Note (\*1):  
Validate each alphanumeric entry by pressing <ENTER>.



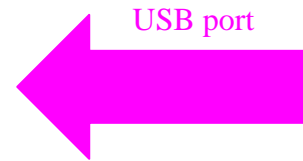




Additional information: RS232 serial interface:

**Note:**

The RS232 serial interface (USIC\_0\_Channel\_0 pins P7.3 and P7.4) is available via the [USB port](#) which converts the TTL-UART-signals to USB-signals (using a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).



USB port



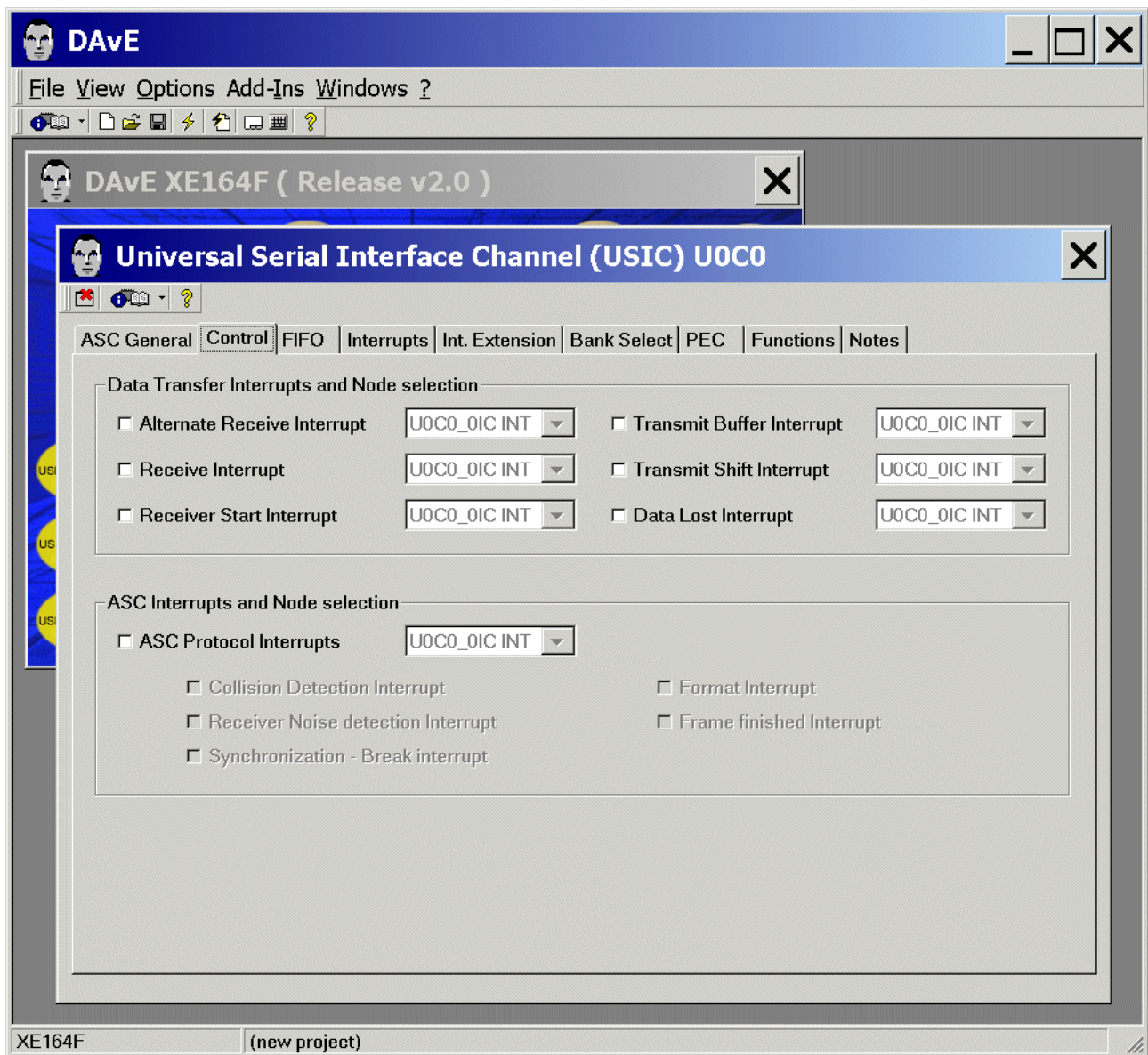
Additional information: Standard UART Pins (Source: User's Manual):

**Table 10-10 Configuration Data for Bootstrap Loader Modes**

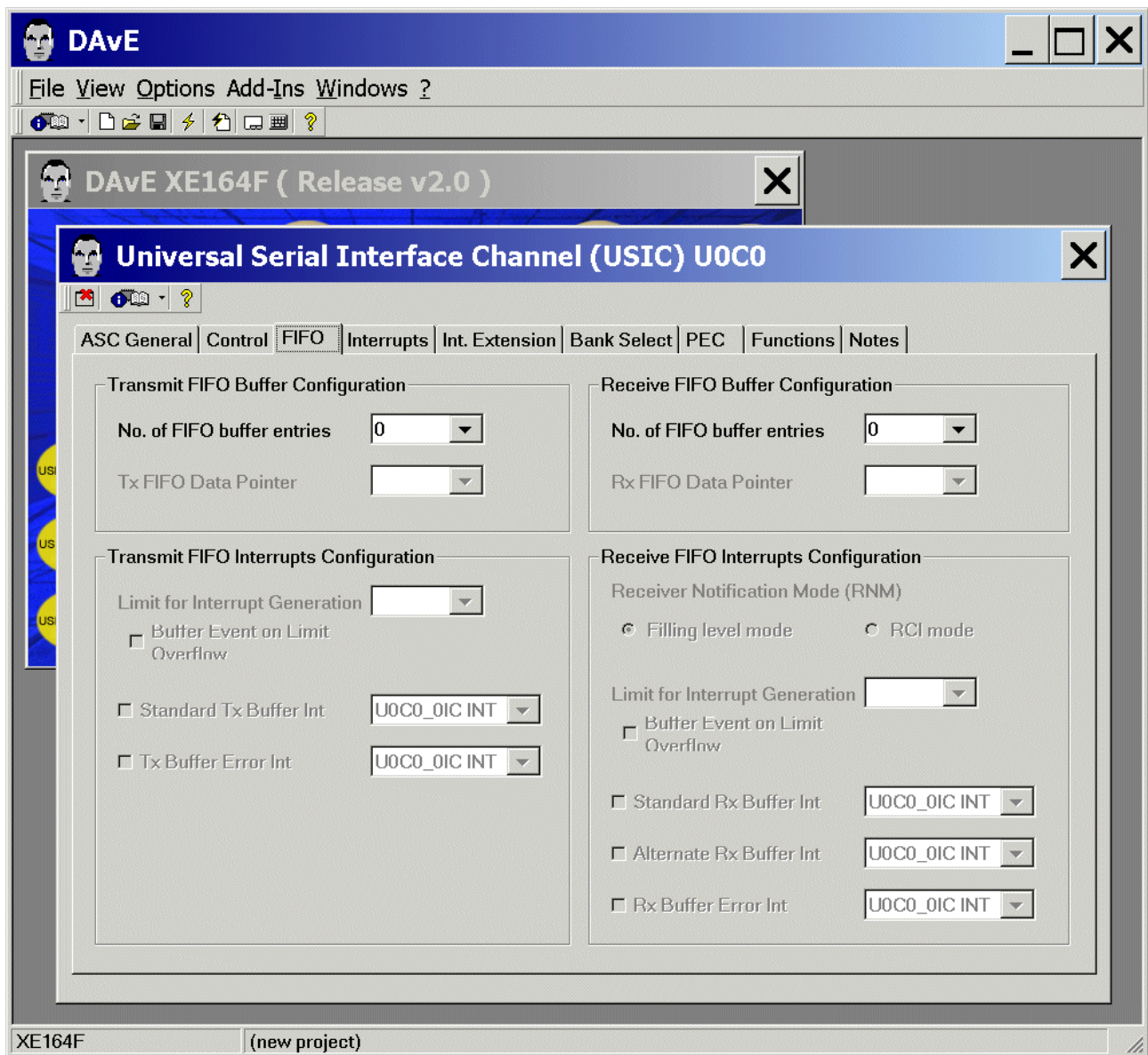
Bootstrap Loader Mode	Configuration on P10.3-0 <sup>1)</sup>	Receive Line from Host	Transmit Line to Host	Transferred Data
Standard UART	x110 <sub>B</sub>	RxD = P7.4	TxD = P7.3	32 Bytes
Sync. Serial	1001 <sub>B</sub>	MRST = P2.4	MTSR = P2.3 SCLK = P2.5 SLS = P2.6	n Bytes; 1 ... 65,280
MultiCAN	x101 <sub>B</sub>	RxDC0 = P2.6	TxDC0 = P2.5	8 × n Bytes

1) x means that the level on the corresponding pin is irrelevant.

Control: (do nothing)

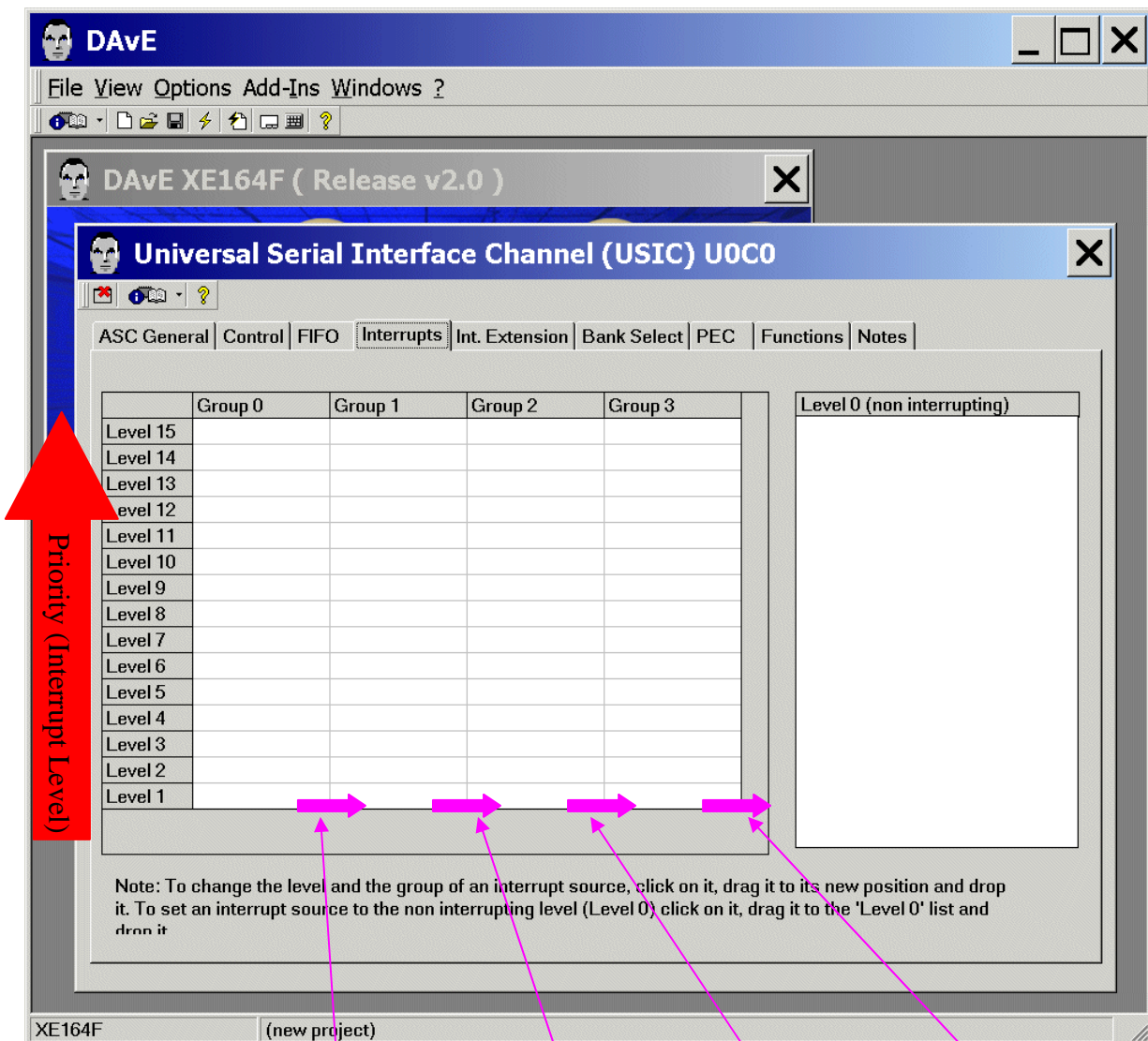


**FIFO:** (do nothing)





Interrupts: (do nothing)



Priority (Interrupt Level)

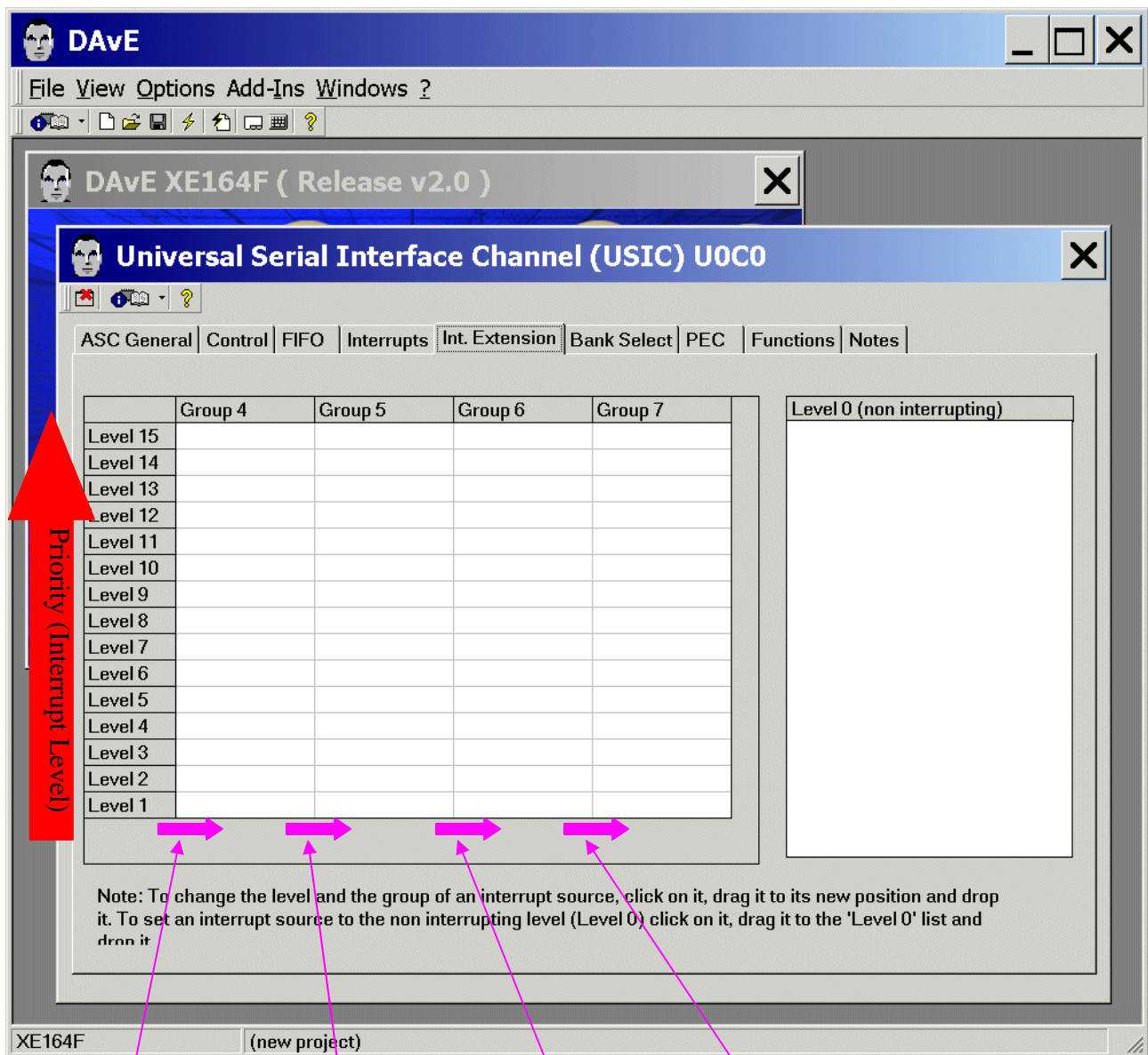
Group Priority 0 → Group Priority 1 → Group Priority 2 → Group Priority 3 →



**Note:**

For the serial communication with a terminal program (e.g. Docklight, [www.docklight.de](http://www.docklight.de)) running on your host computer the myprintf function is used. The myprintf function uses Software-Polling-Mode therefore we do not need to configure any interrupts.

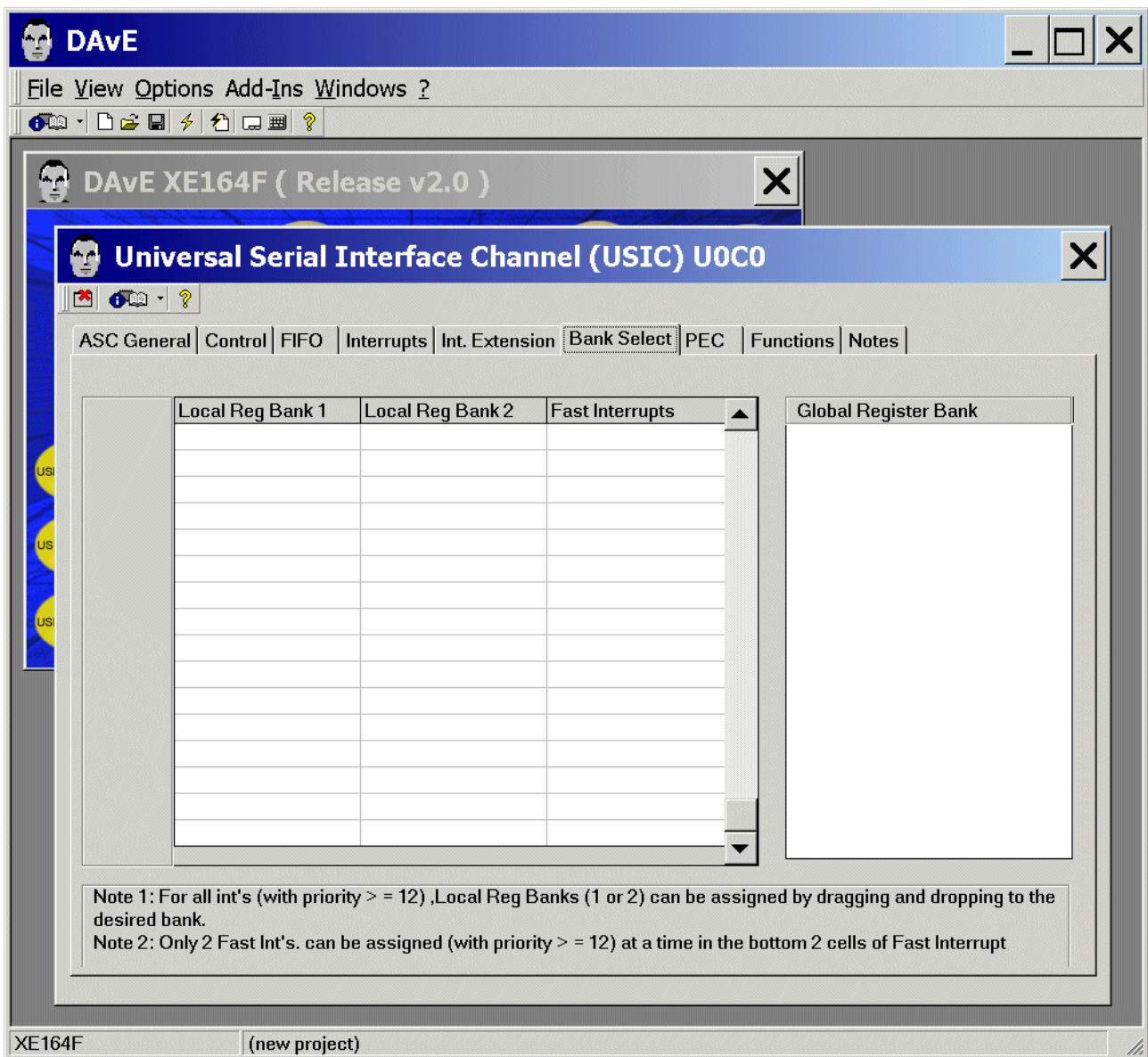
Int. Extension: (do nothing)



→ Group Priority 4 → Group Priority 5 → Group Priority 6 → Group Priority 7

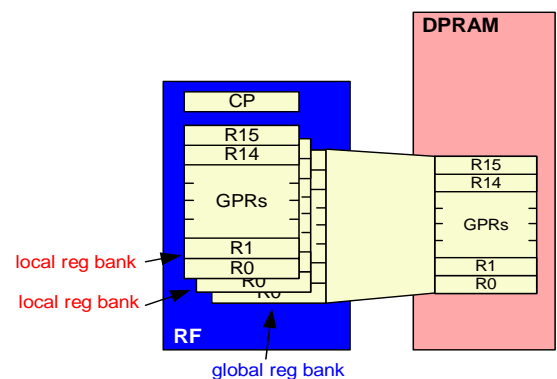


Bank Select: (do nothing)

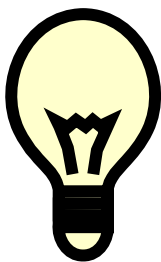
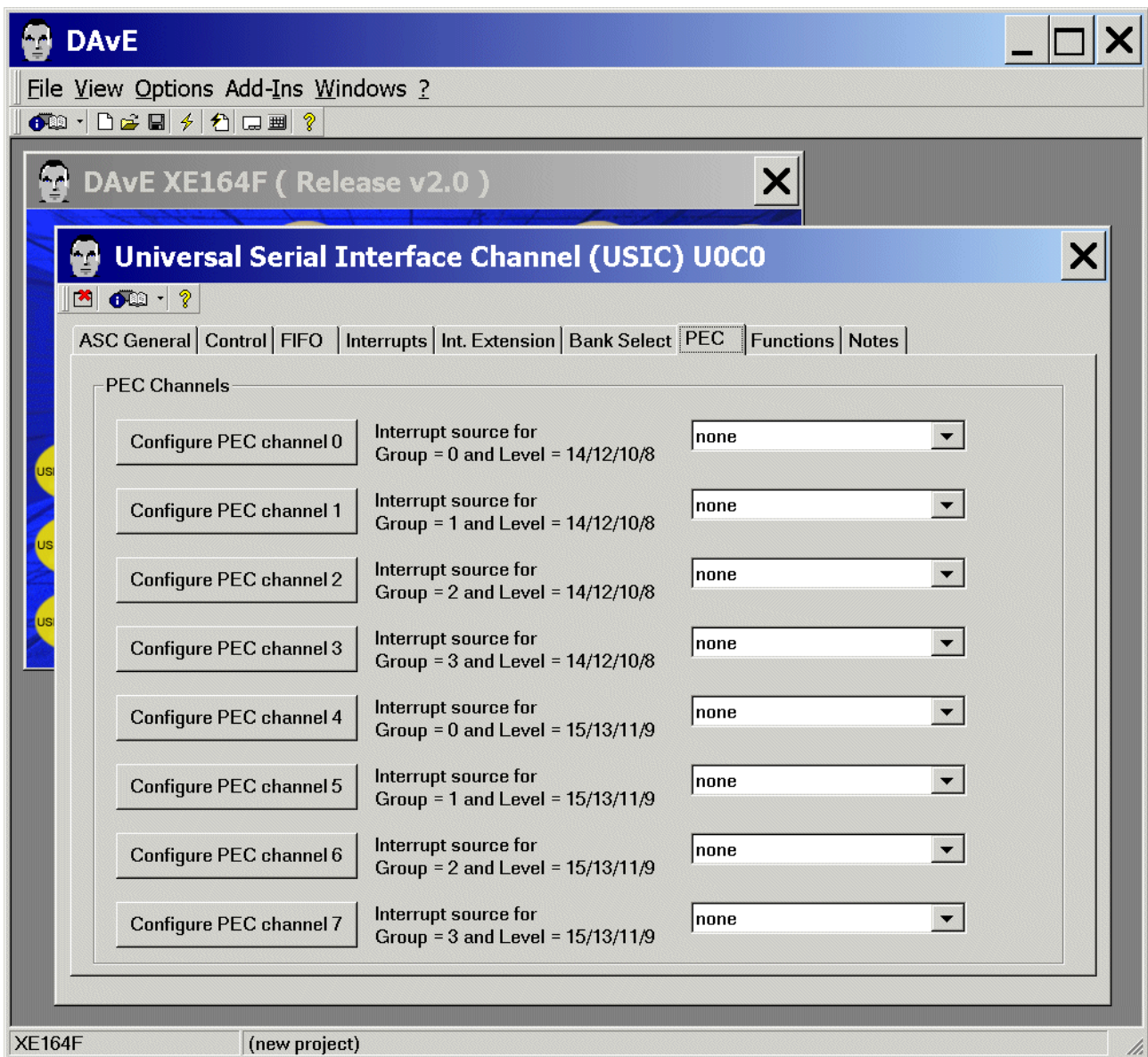


Note:

For our hello world program the 2 local register banks are not needed.

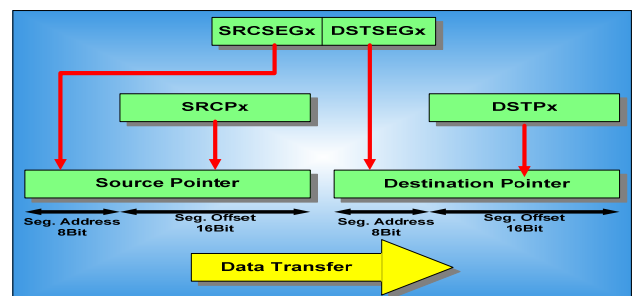


PEC: (do nothing)

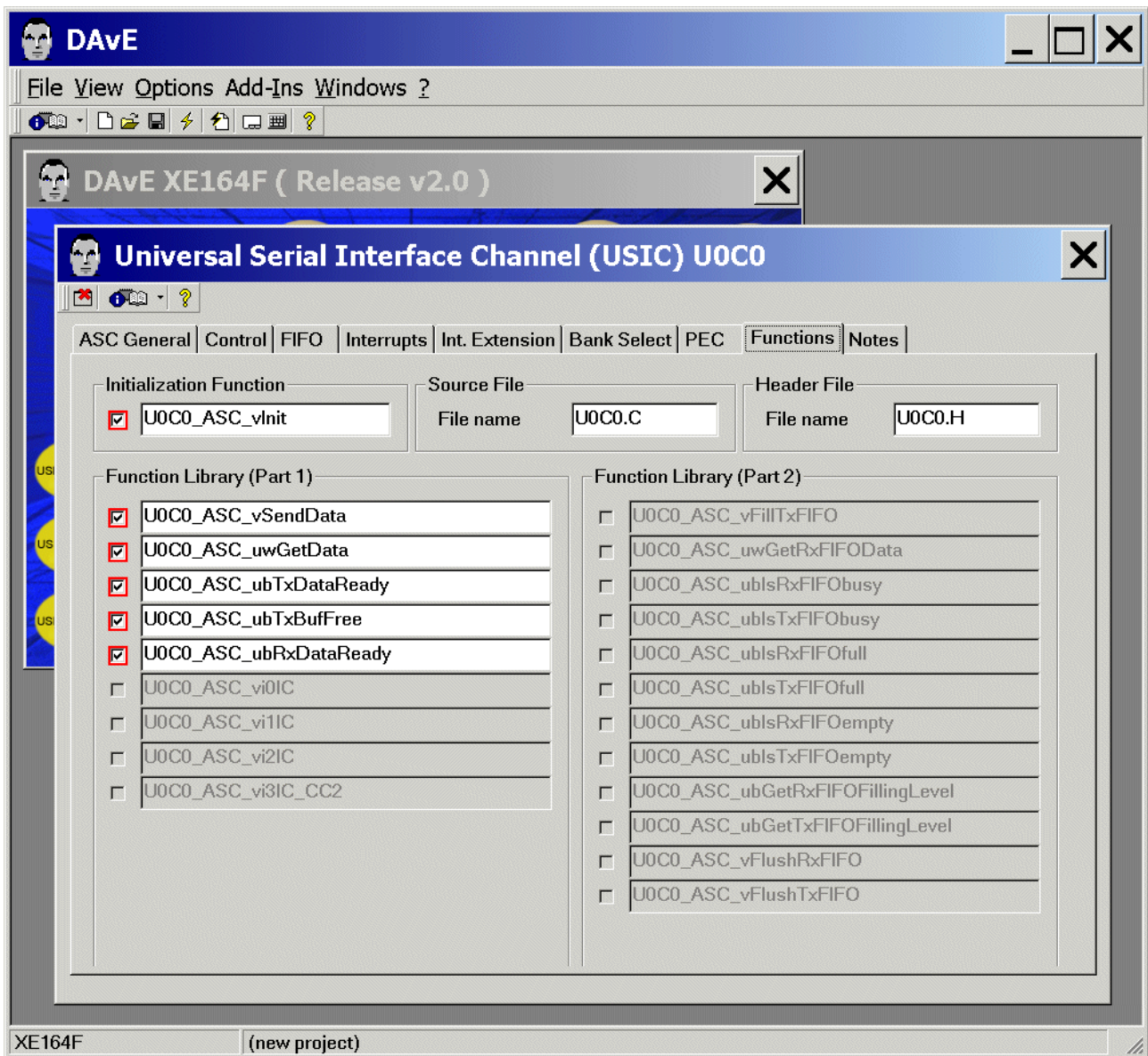


**Note:**

For our hello world program the 8 PEC Channels are not needed.



Functions: Initialization Function: **click** ☒ U0C0\_ASC\_vInit  
 Functions: Function Library (Part 1): **click** ☒ U0C0\_ASC\_vSendData  
 Functions: Function Library (Part 1): **click** ☒ U0C0\_ASC\_uwGetData  
 Functions: Function Library (Part 1): **click** ☒ U0C0\_ASC\_ubTxDataReady  
 Functions: Function Library (Part 1): **click** ☒ U0C0\_ASC\_ubTxBufFree  
 Functions: Function Library (Part 1): **click** ☒ U0C0\_ASC\_ubRxDataReady



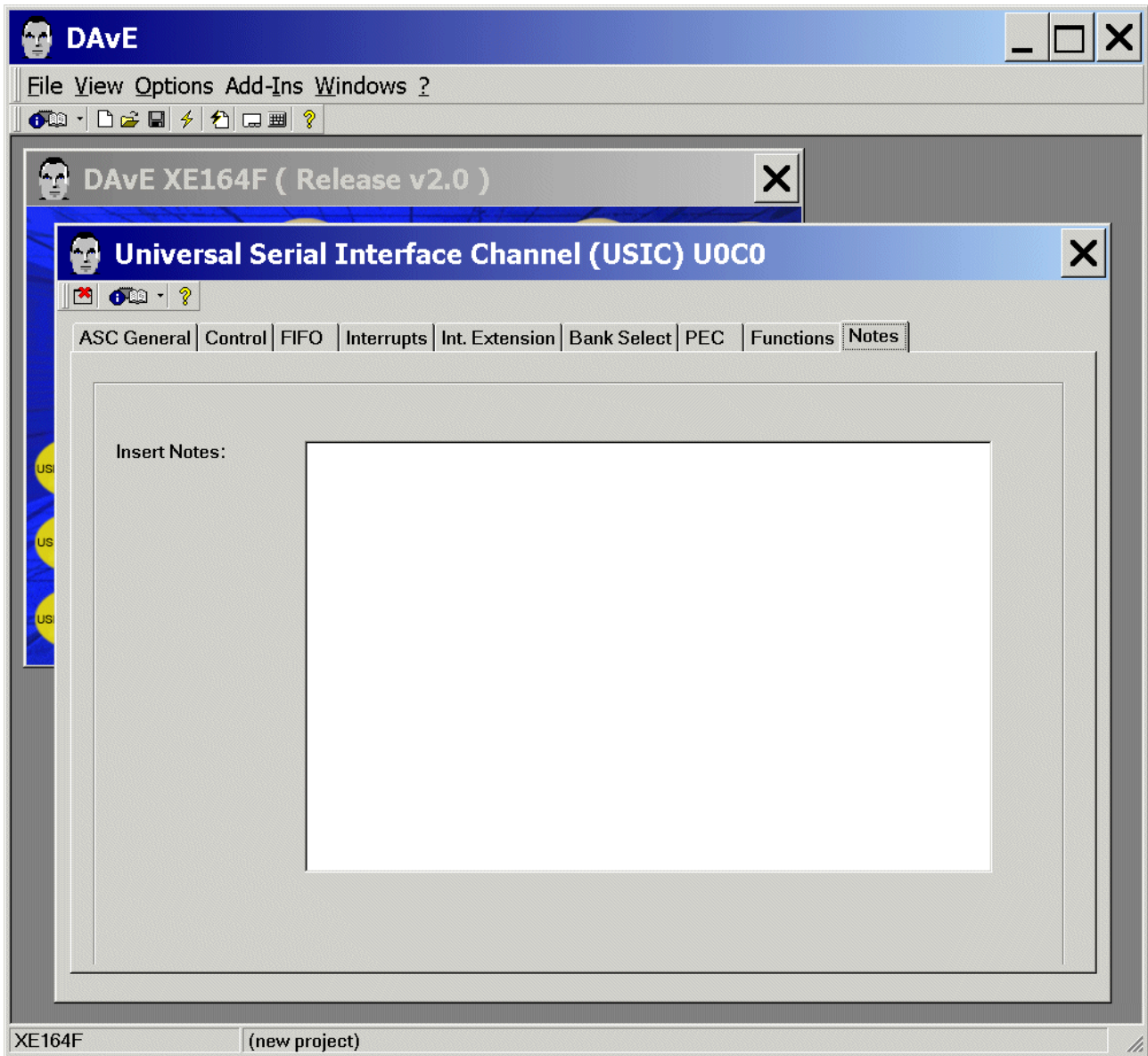
**Note:**

You can change function names (e.g. U0C0\_ASC\_vInit) and file names (e.g. U0C0.C, U0C0.H) anytime.






Notes: (do nothing)



Note:

Notes: If you wish, you can insert your comments here.

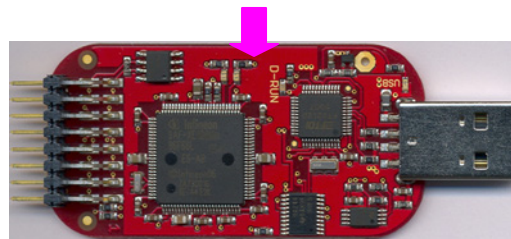
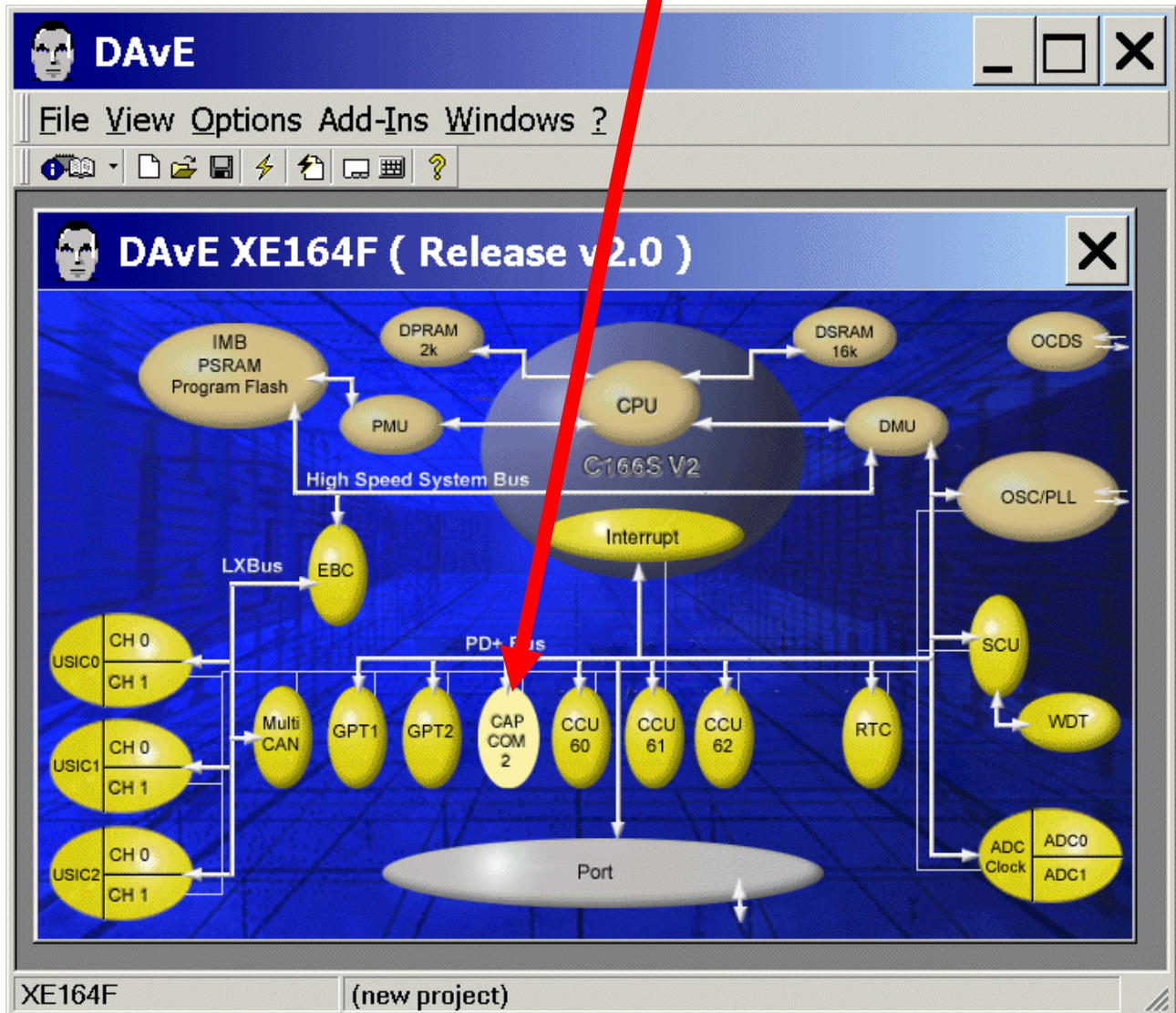


Exit and Save this dialog now by clicking  the close button.



Configure Timer T7 in the CAPCOM 2 module:

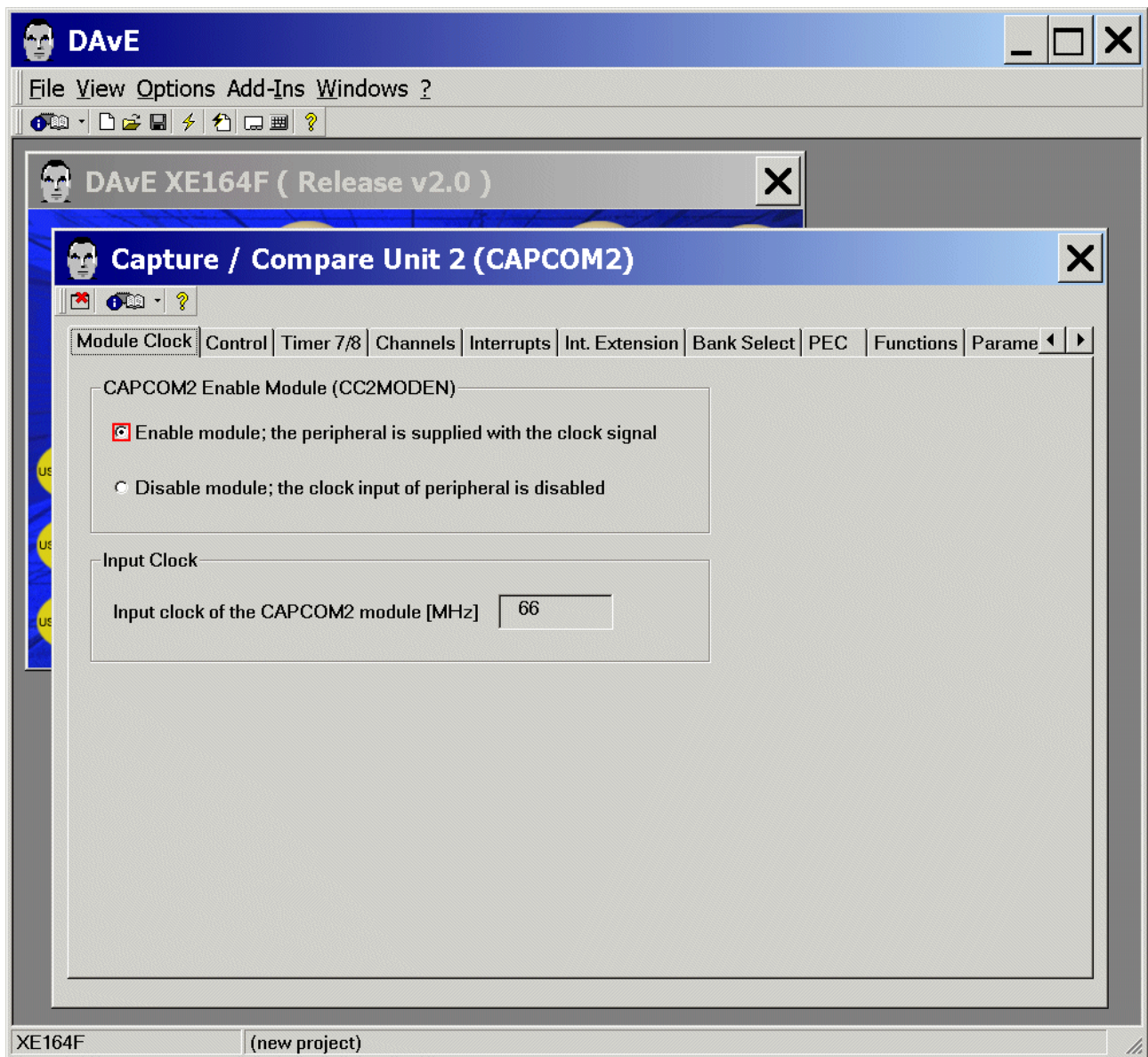
The configuration window/dialog can be opened by clicking the specific block/module (CAPCOM2).



**Note:**

The **LED on IO\_Port\_2.7** will be blinking (if selected in the main menu) with a frequency of about 1 second (done in the Timer\_7-Interrupt-Service-Routine). Therefore we have to configure Timer\_7.

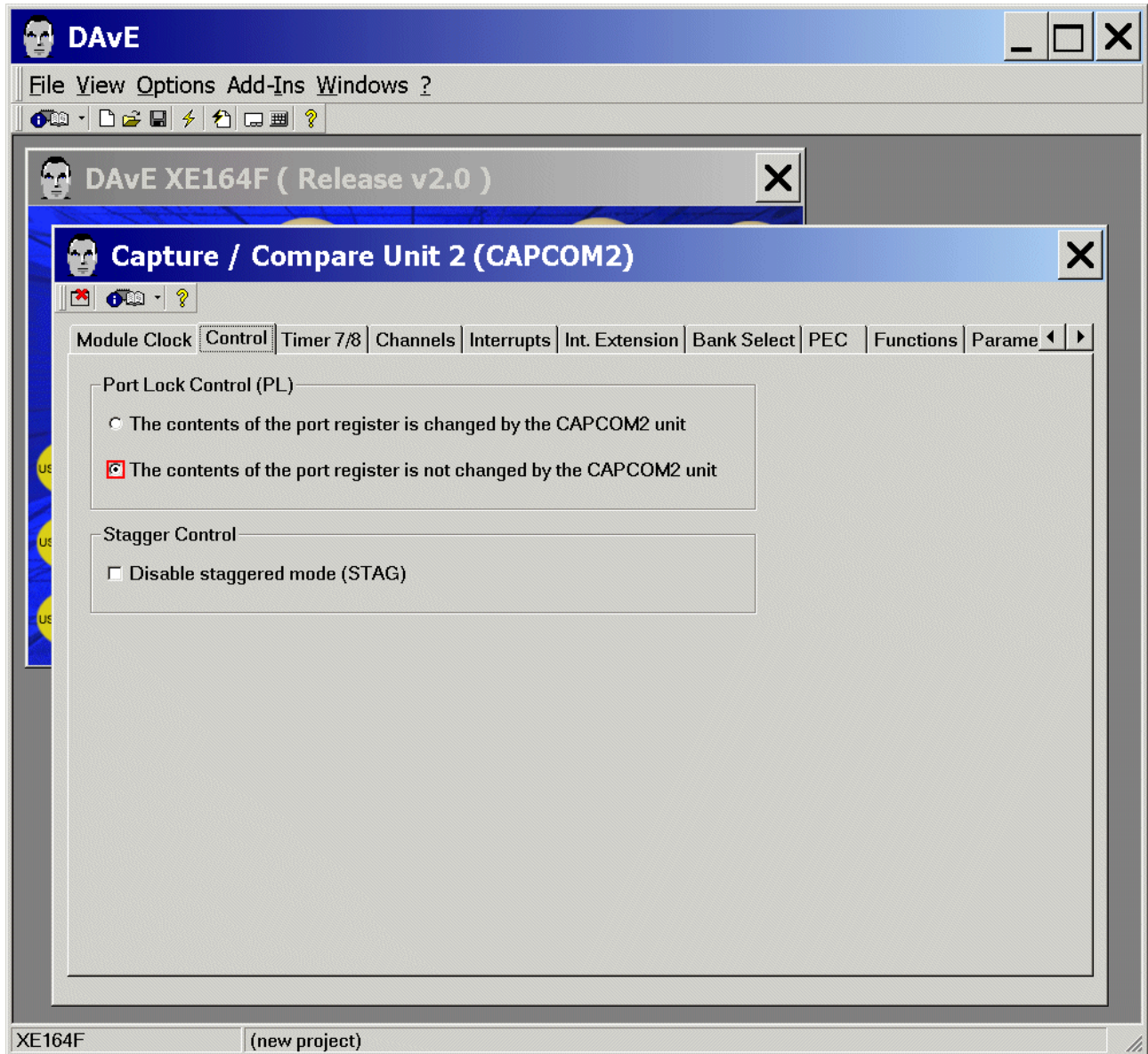
Module Clock: CAPCOM2 Enable Module: **click** ☒ Enable module



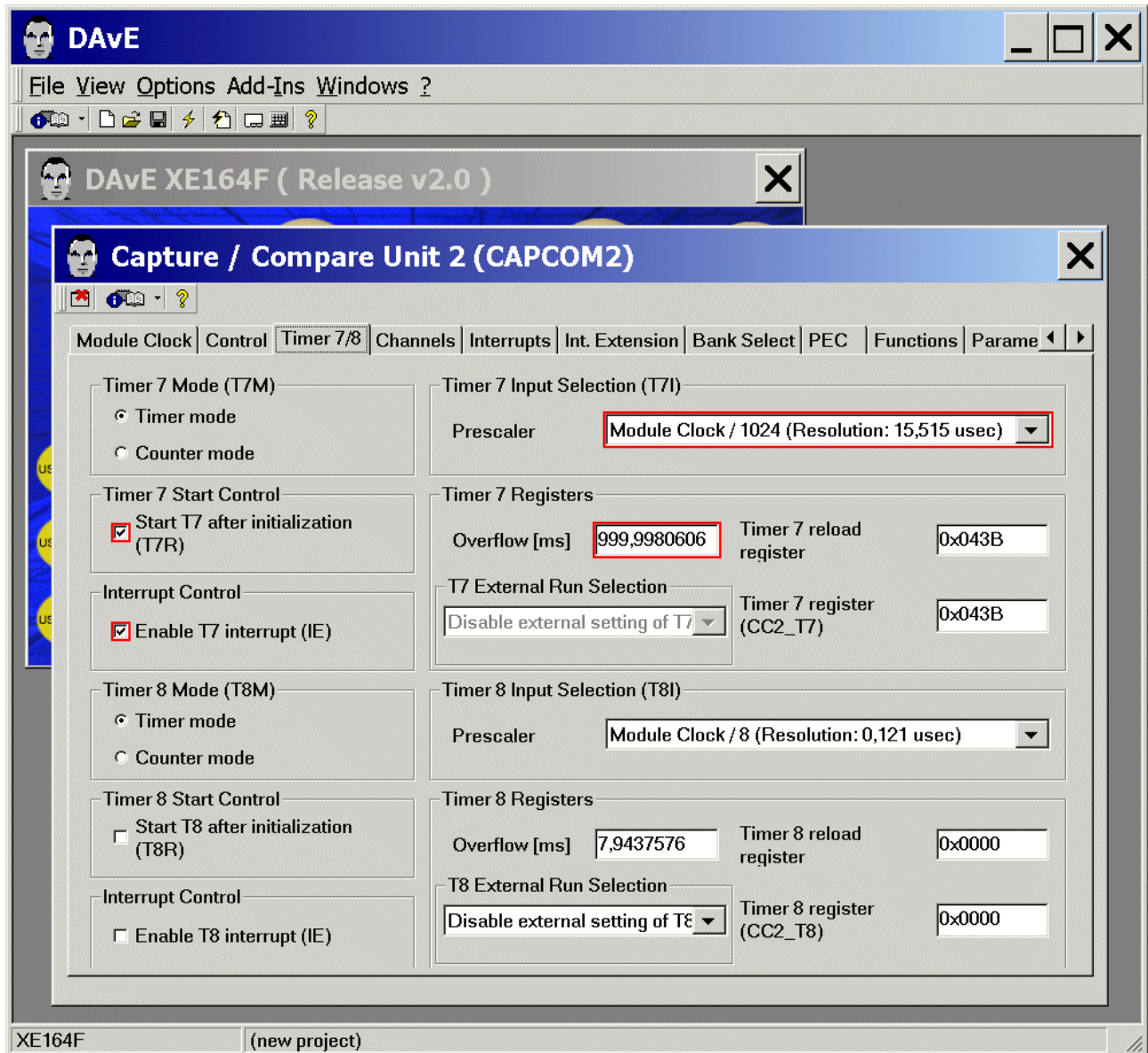


**Control: Port Lock Control:**

**click** ☒ The contents of the port register is not changed by the CAPCOM2 unit

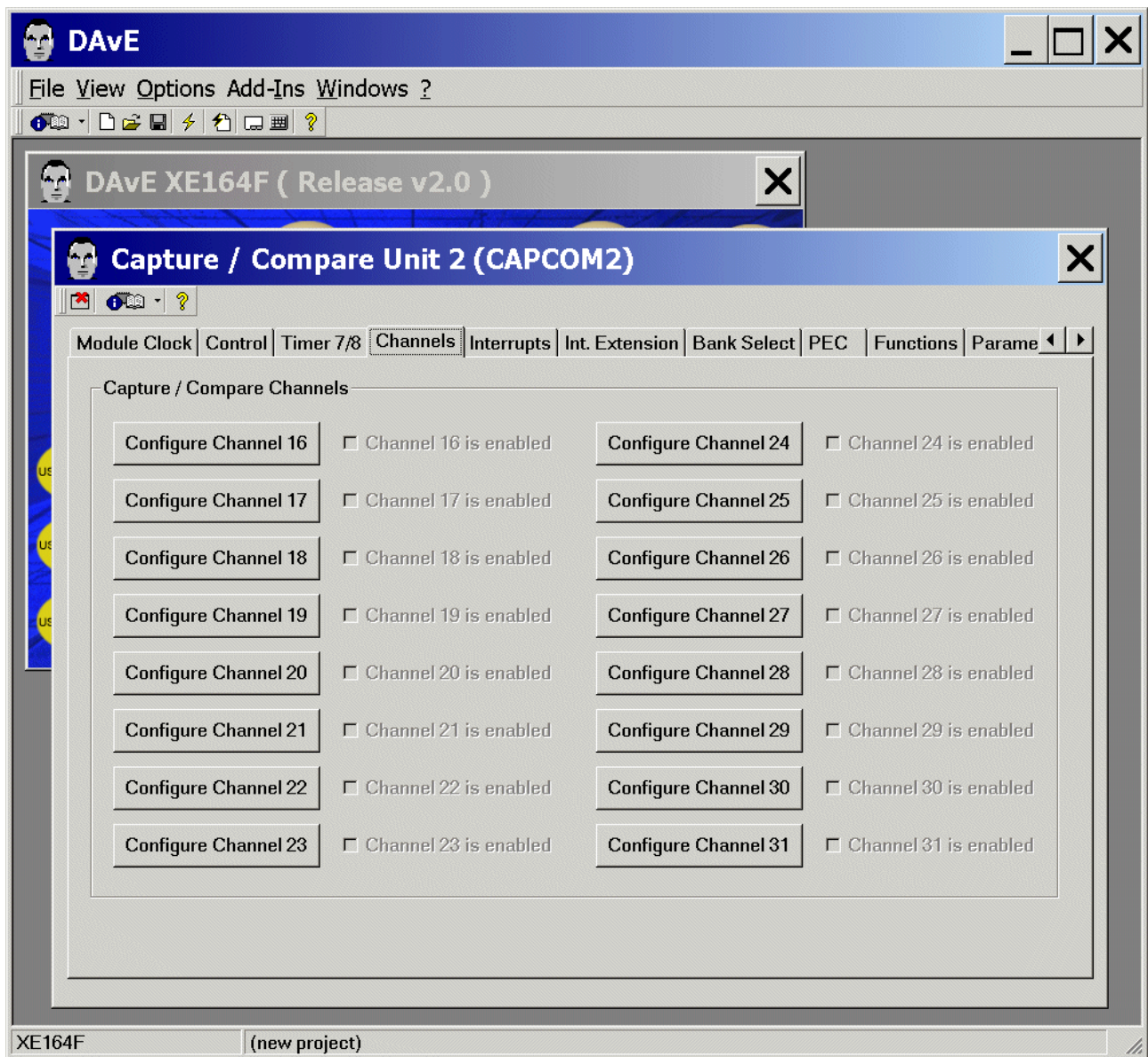


Timer 7/8: Timer 7 Start Control: **click** ✓ Start T7 after initialization (T7R)  
 Timer 7/8: Interrupt Control: **click** ✓ Enable T7 interrupt (IE)  
 Timer 7/8: Timer 7 Input Selection (T7I): **Prescaler**: **choose** Module Clock/1024  
 Timer 7/8: Timer 7 Registers: **Overflow** [s]: **insert** 1 <ENTER>

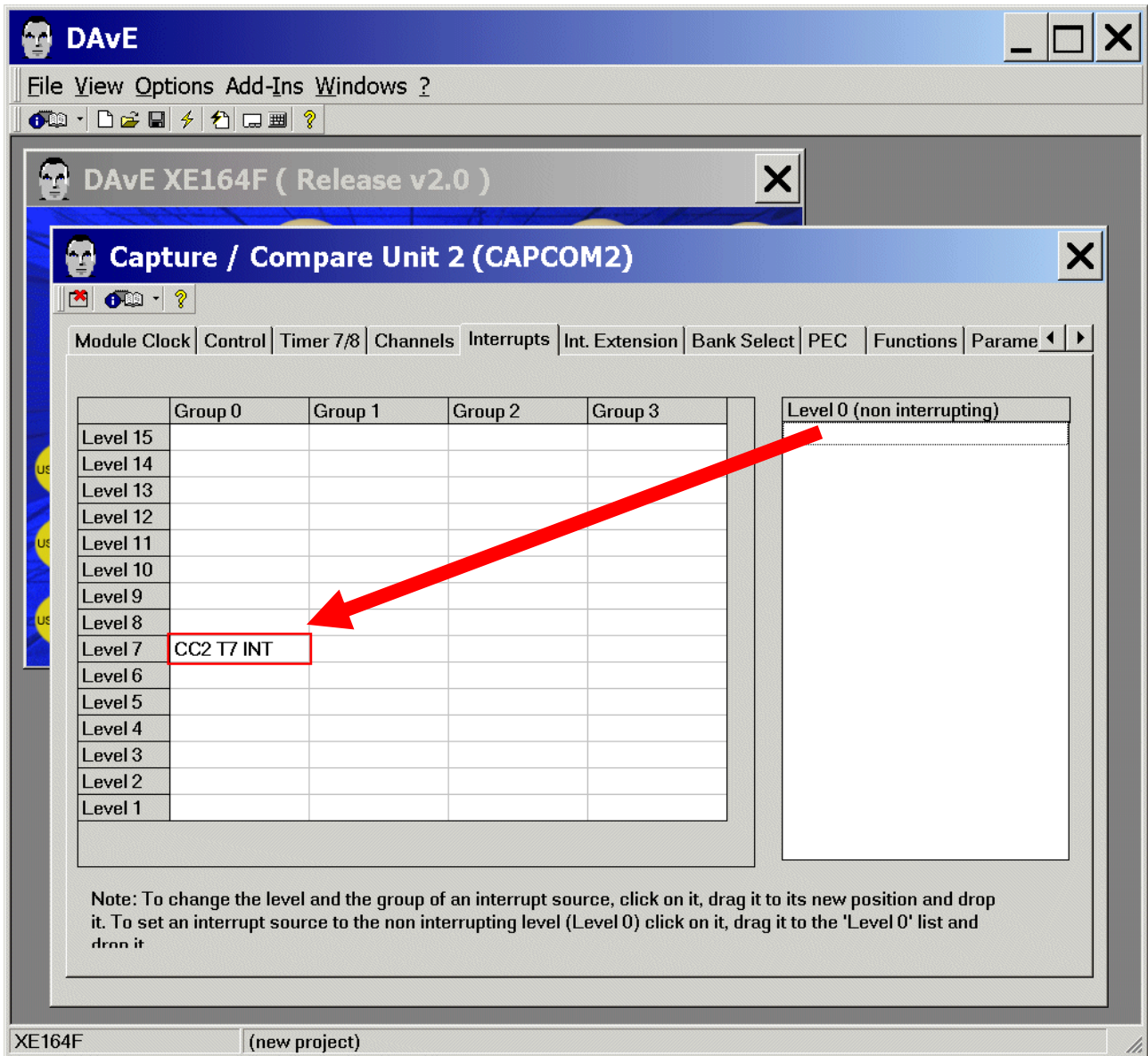




Channels: (do nothing)



Interrupts: drag and drop the CC2 T7 INT to Interrupt Level 7, Group 0

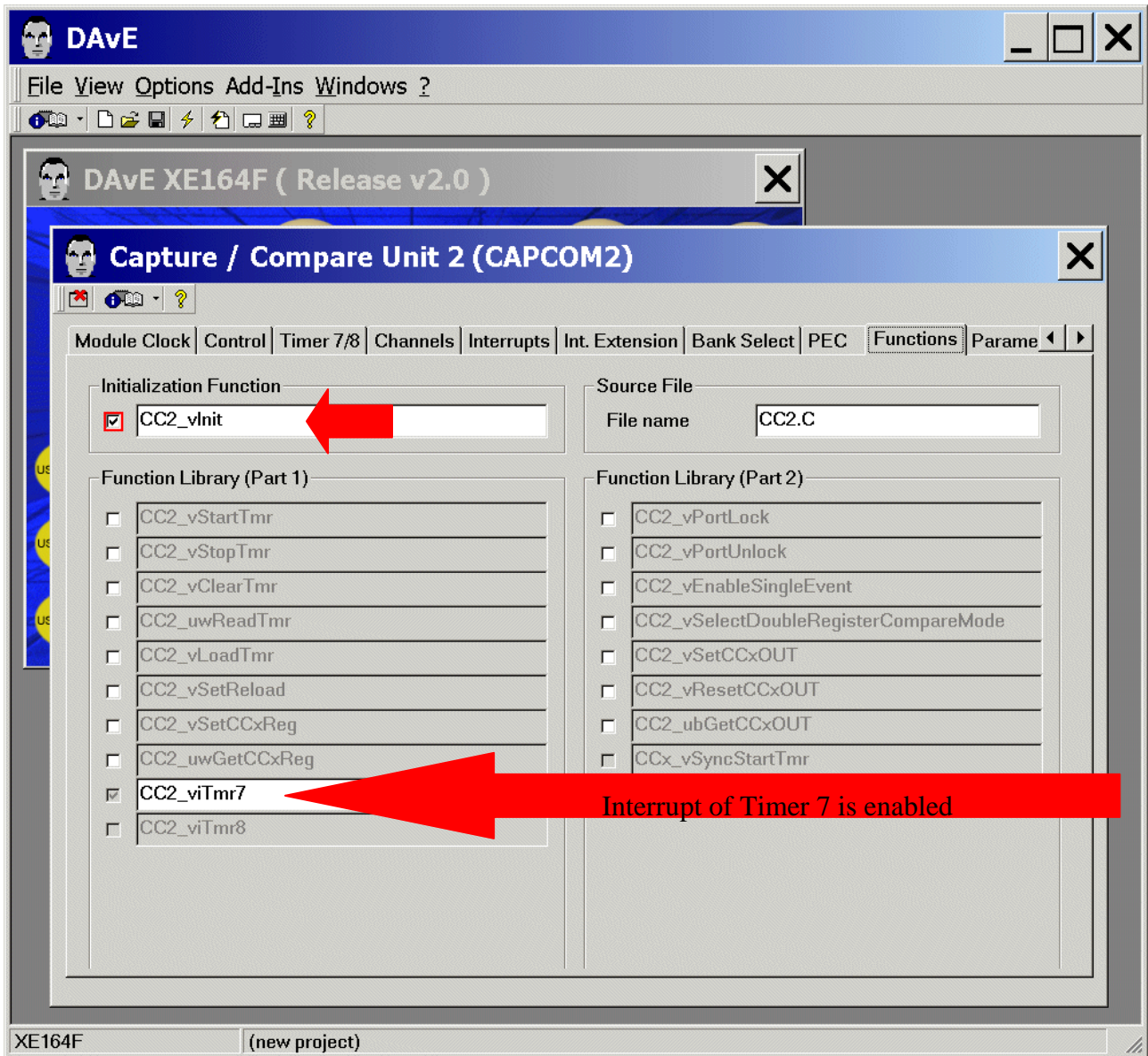


Int. Extension: (do nothing)

Bank Select: (do nothing)


PEC: (do nothing)

Functions: Initialization Function: click/check ☒ CC2\_vInit



Parameters: (do nothing)

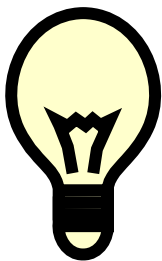
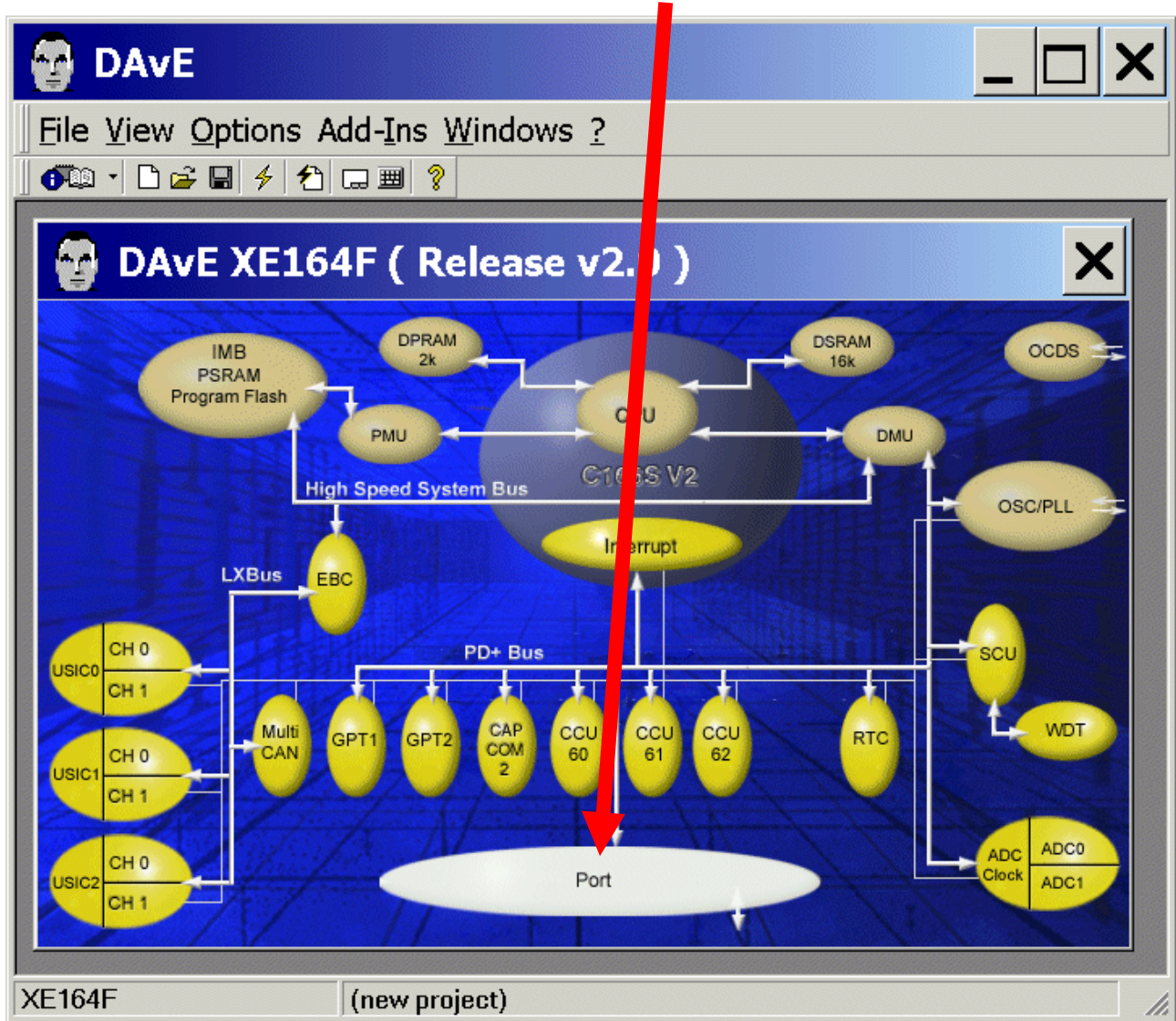
Notes: If you wish, you can insert your comments here.

Exit and Save this dialog now by clicking  the close button.

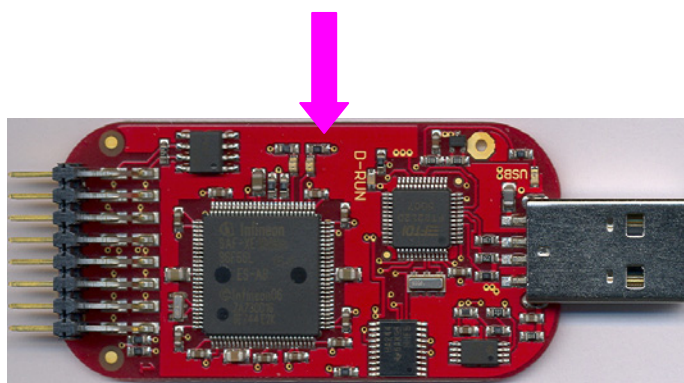


Configure Port 2 Pin 7 to Output:

The configuration window/dialog can be opened by [clicking](#) the specific block/module (Port).

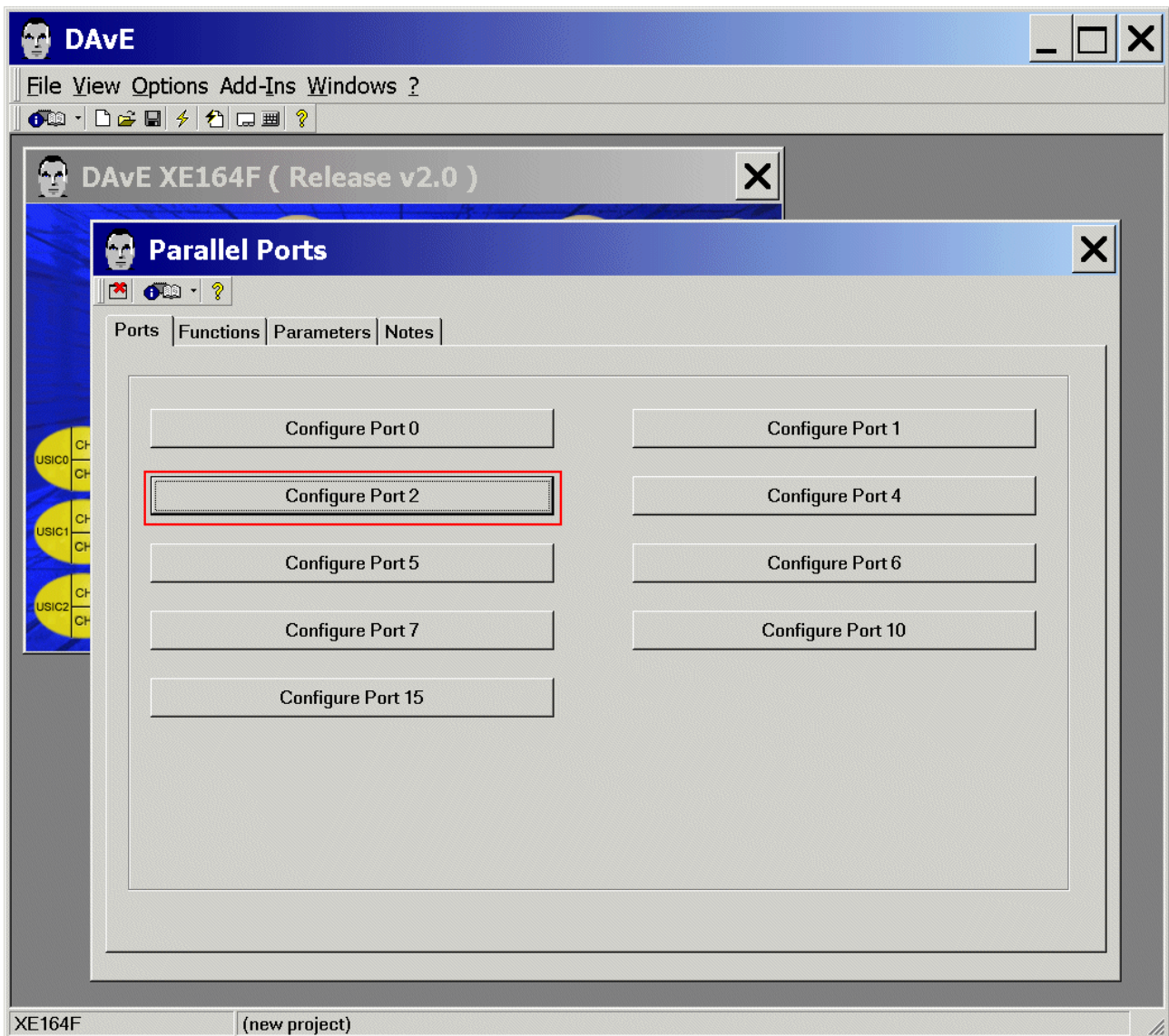


**Note:**  
The LED is connected to `IO_Port_2.7`

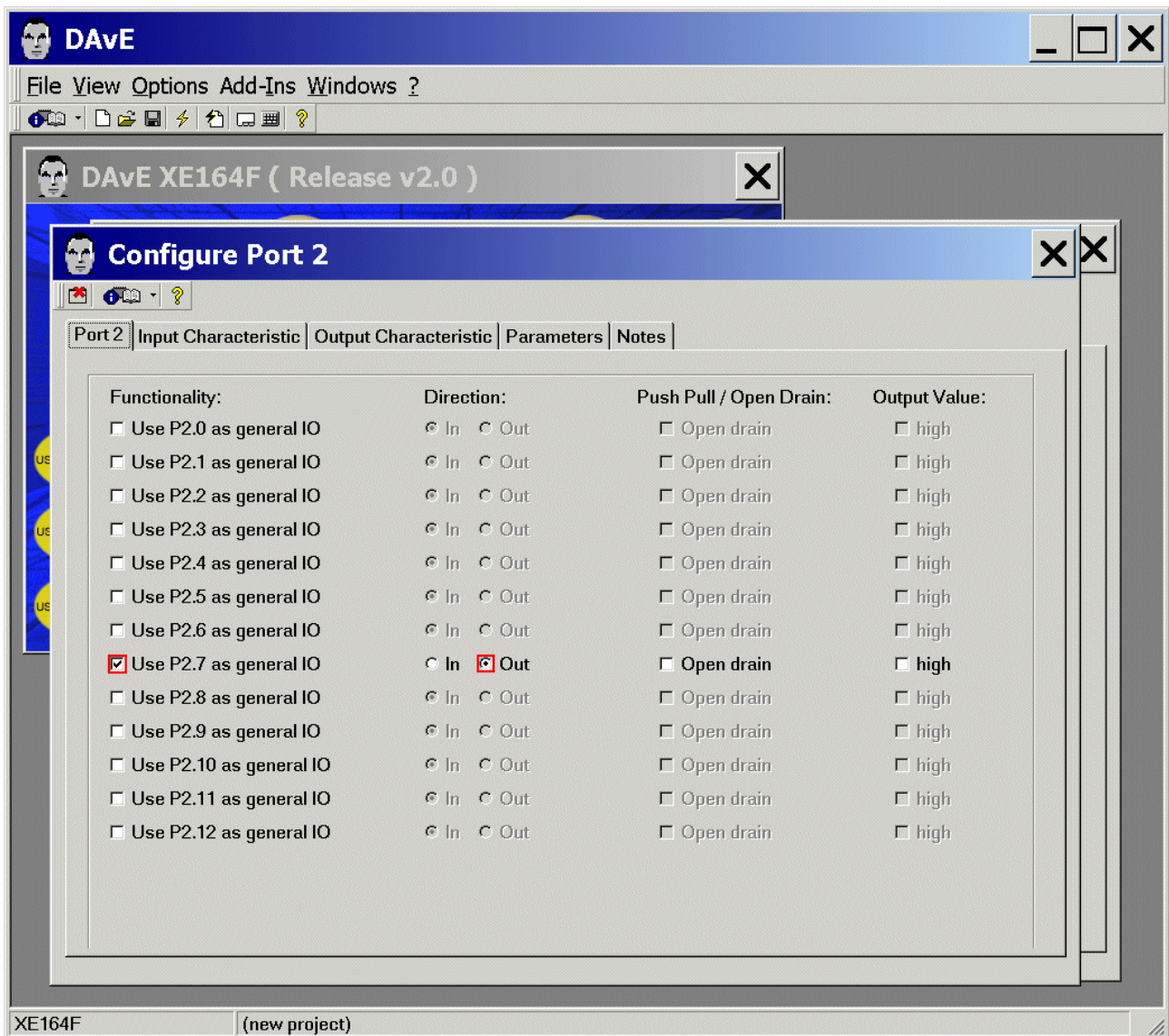




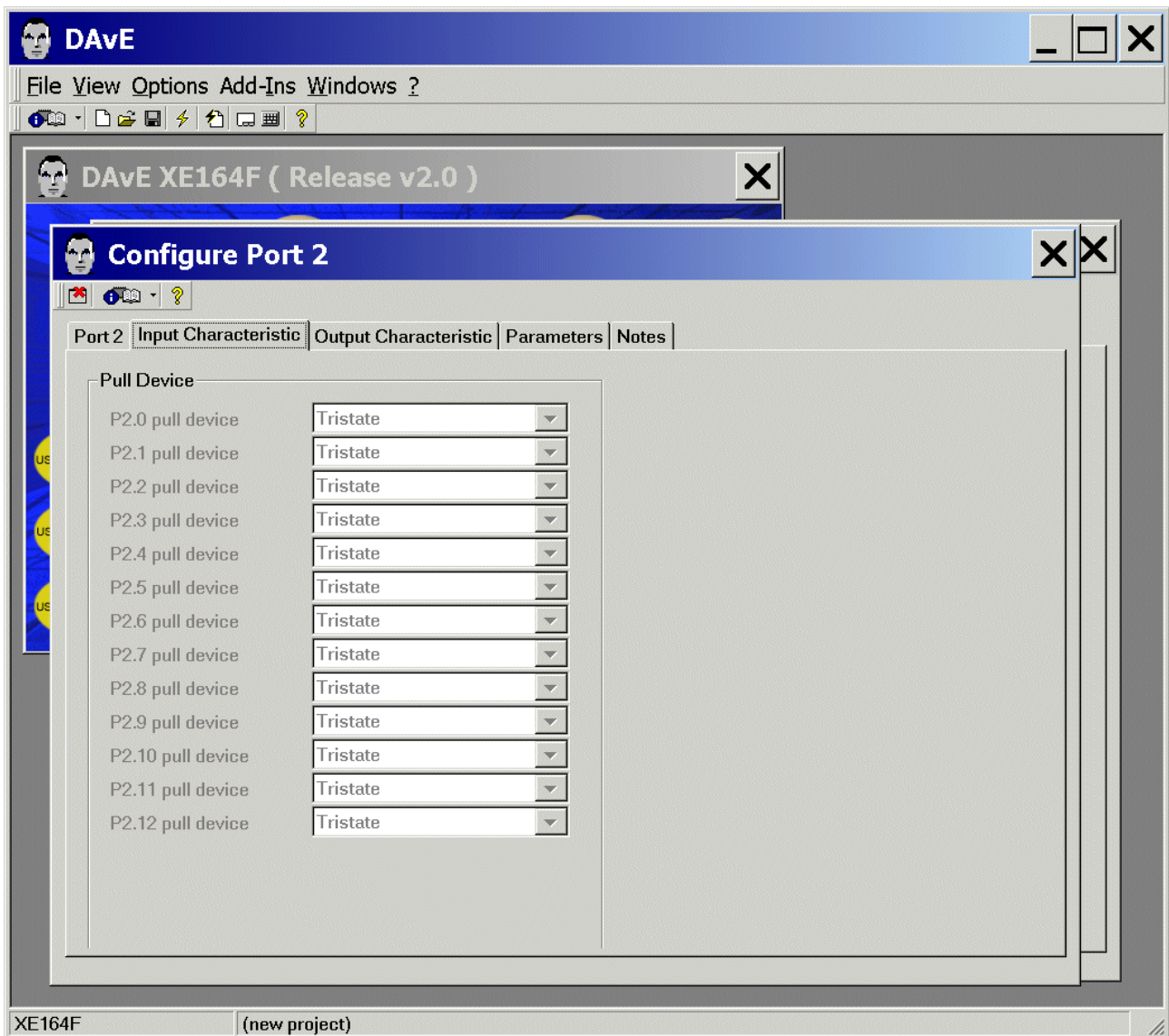
Ports: click "Configure Port 2"



Port 2: Functionality: **click** ☒ Use P2.7 as general IO - Direction: **click** ☒ Out

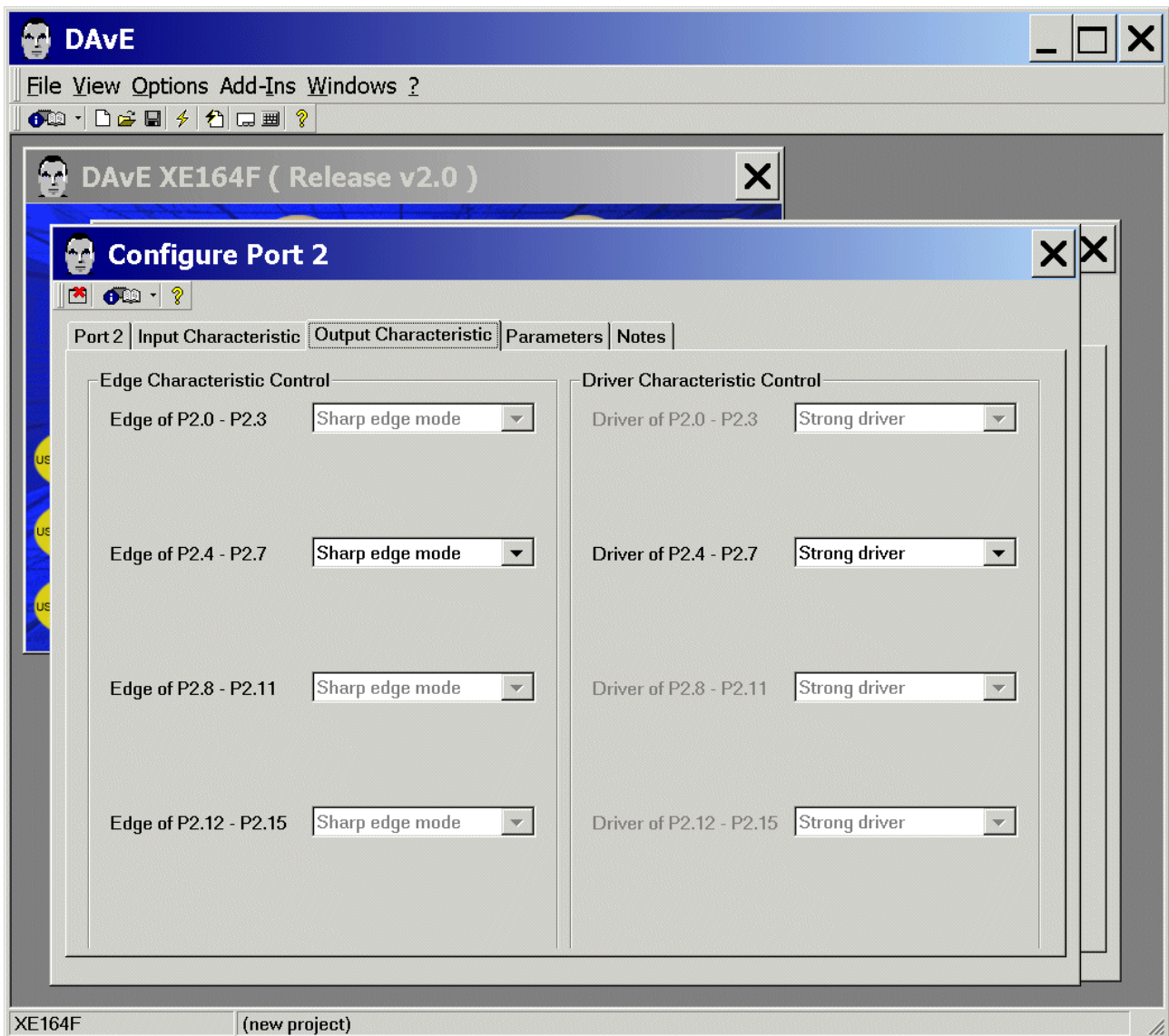


Input Characteristic: (do nothing)



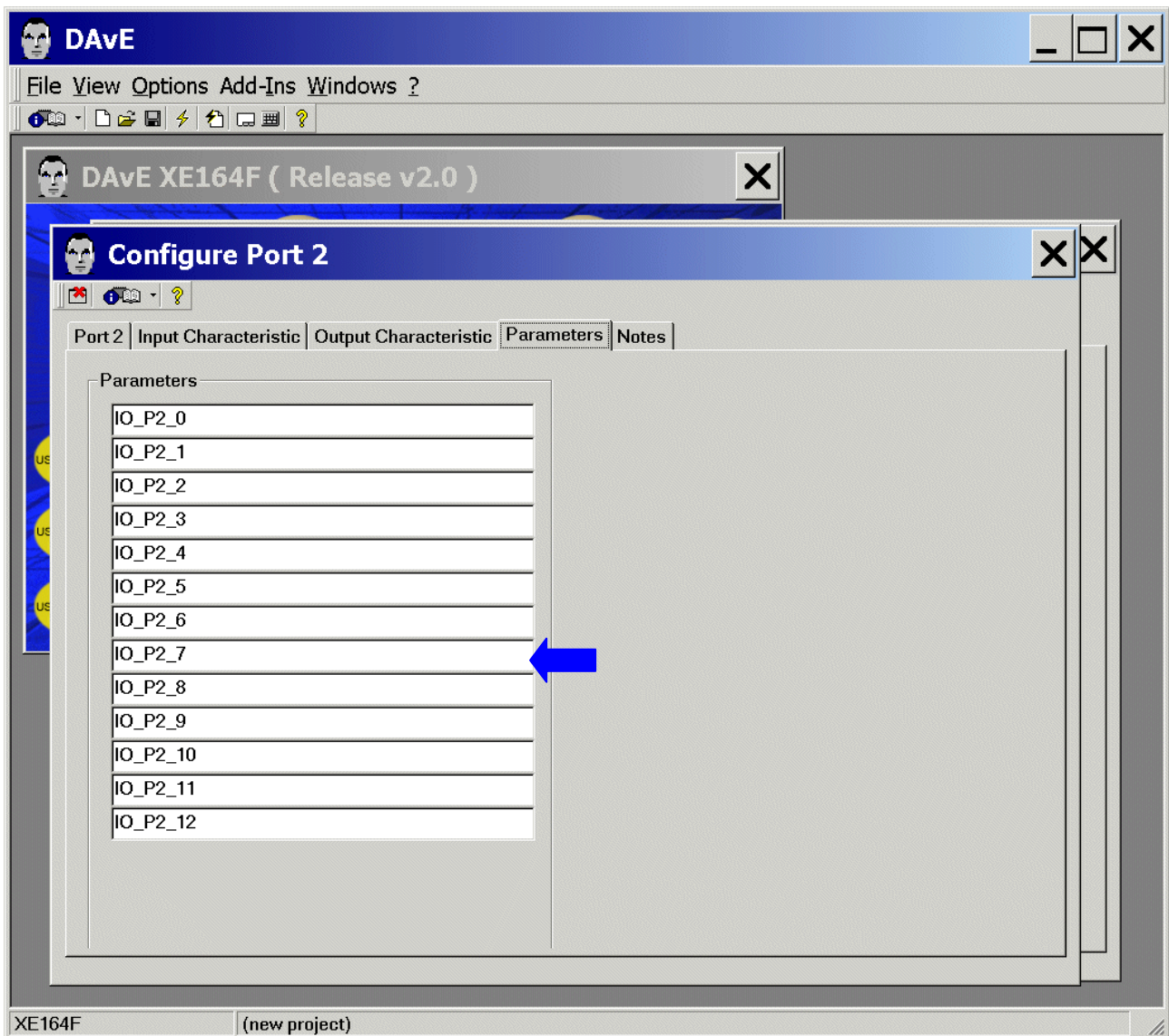


Output Characteristic: (do nothing)





Parameters: (do nothing)




**Note:**

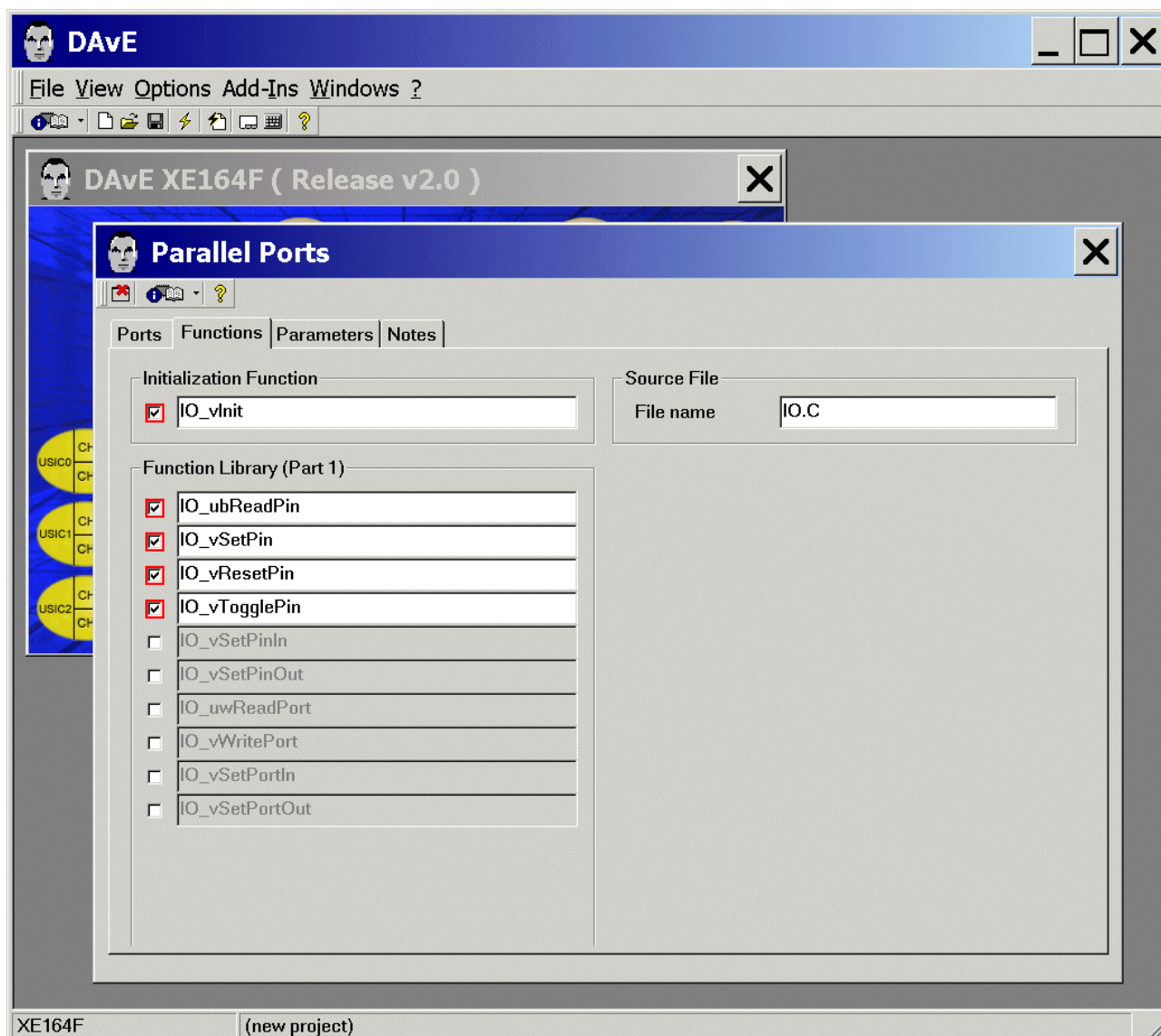
We will use the name **IO\_P2\_7** in application programming.



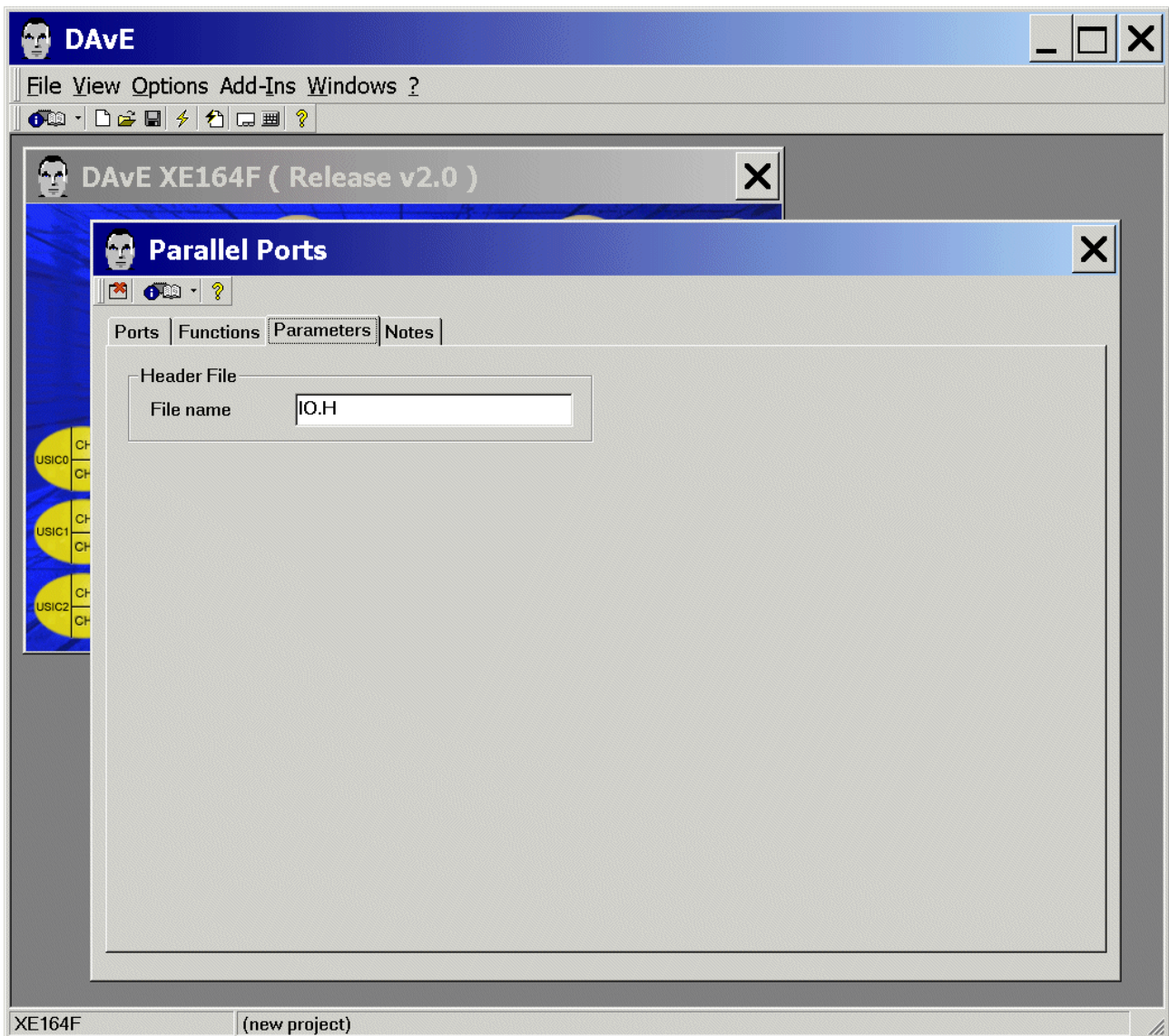
**Notes:** If you wish, you can insert your comments here.

**Exit** and **Save** this dialog now by clicking  the close button:


Functions: Initialization Functions: **click/check** ☒ IO\_vInit  
 Functions: Function Library (Part 1): **click** ☒ IO\_ubReadPin  
 Functions: Function Library (Part 1): **click** ☒ IO\_vSetPin  
 Functions: Function Library (Part 1): **click** ☒ IO\_vResetPin  
 Functions: Function Library (Part 1): **click** ☒ IO\_vTogglePin



Parameters: (do nothing)



Notes: If you wish, you can insert your comments here.

Exit and Save this dialog now by clicking  the close button.





**Note:**

Before we save the DAVe Project we are going to create a suitable directory structure with Windows File Explorer.

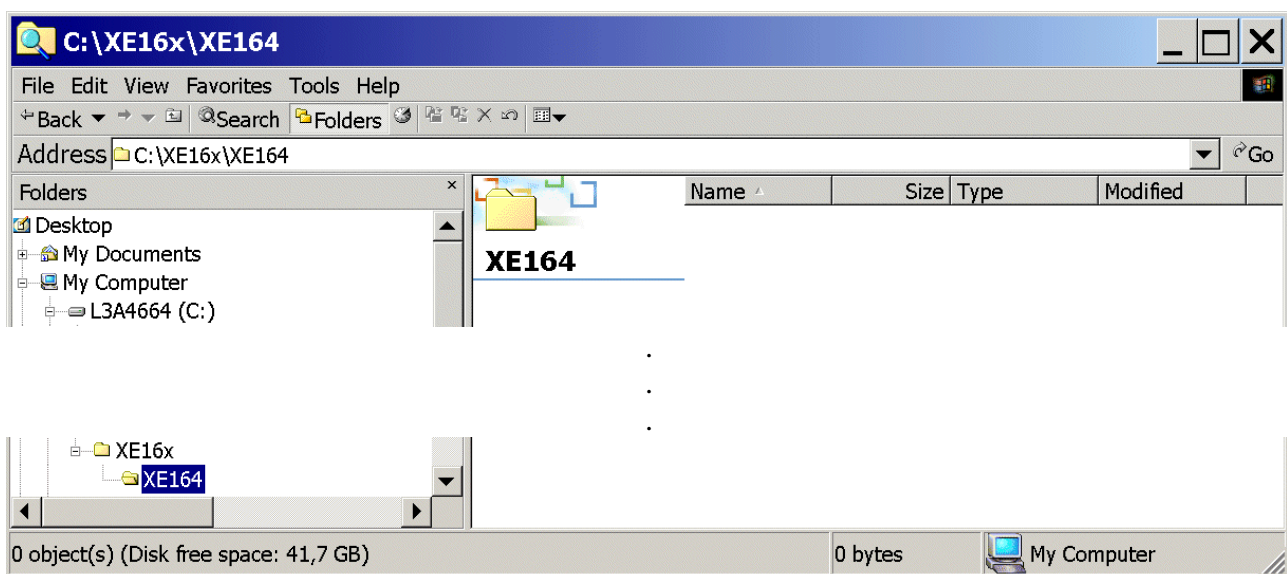
Create a suitable directory structure for Altium's TASKING VX-toolset (Eclipse IDE, Viper compiler):



**Start** Windows File Explorer

**Create directory/folder** C:\XE16x

**Create directory/folder** C:\XE16x\XE164

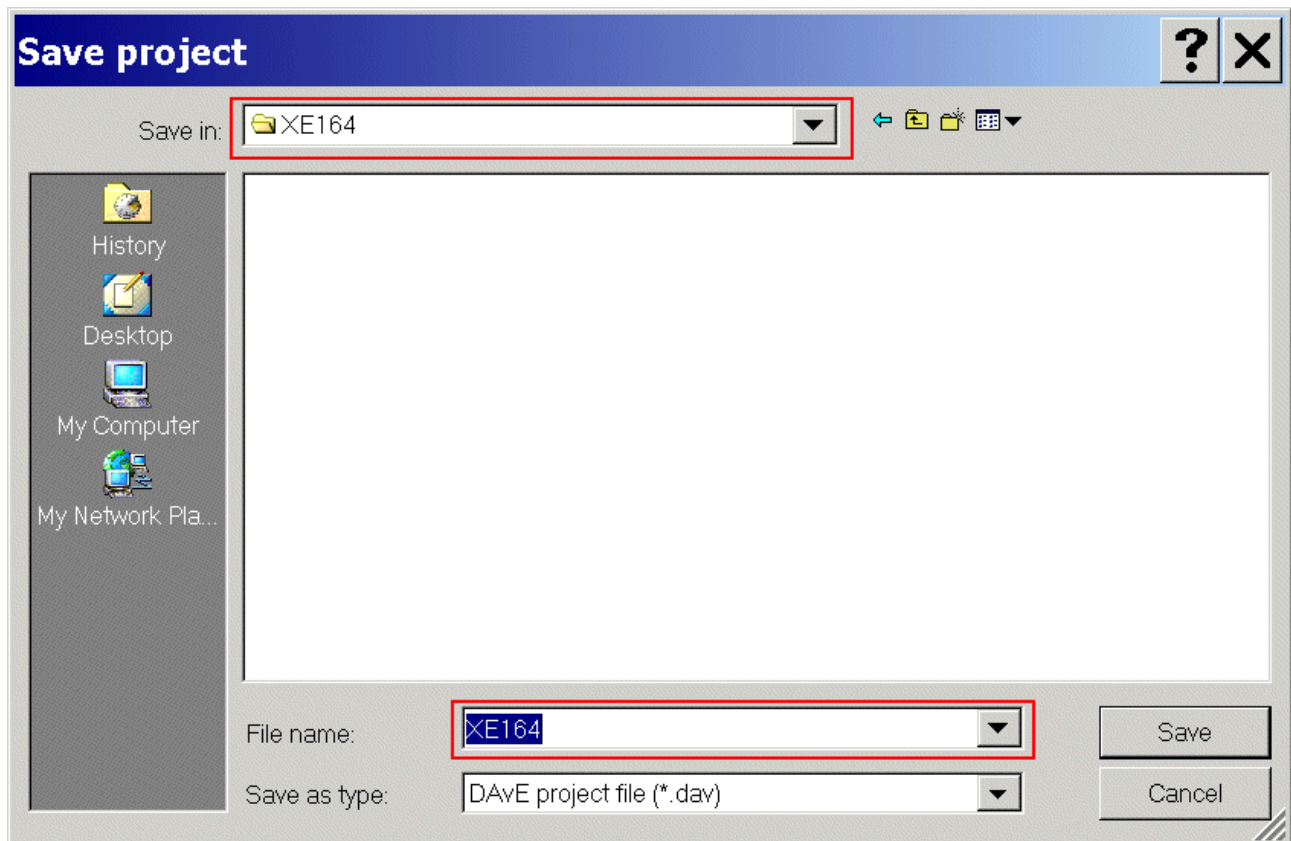




Save the project:


File  
Save

Save project: Save in C:\XE16x\XE164  
File name: XE164



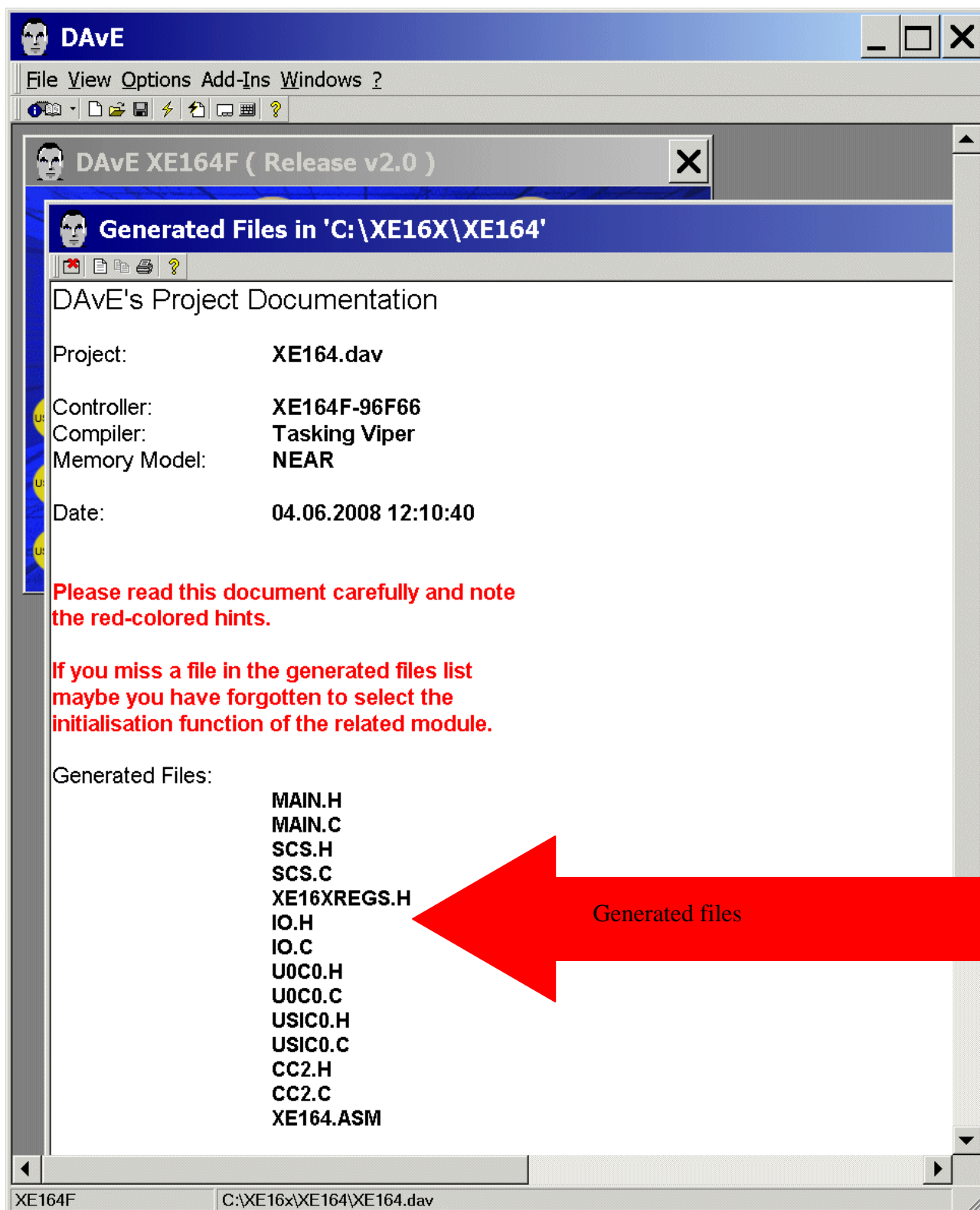
Save

Generate Code:

<p>File Generate Code</p>	<p>or      click </p>
-------------------------------	---



DAvE will show you all the files he has generated  
(File Viewer opens automatically):



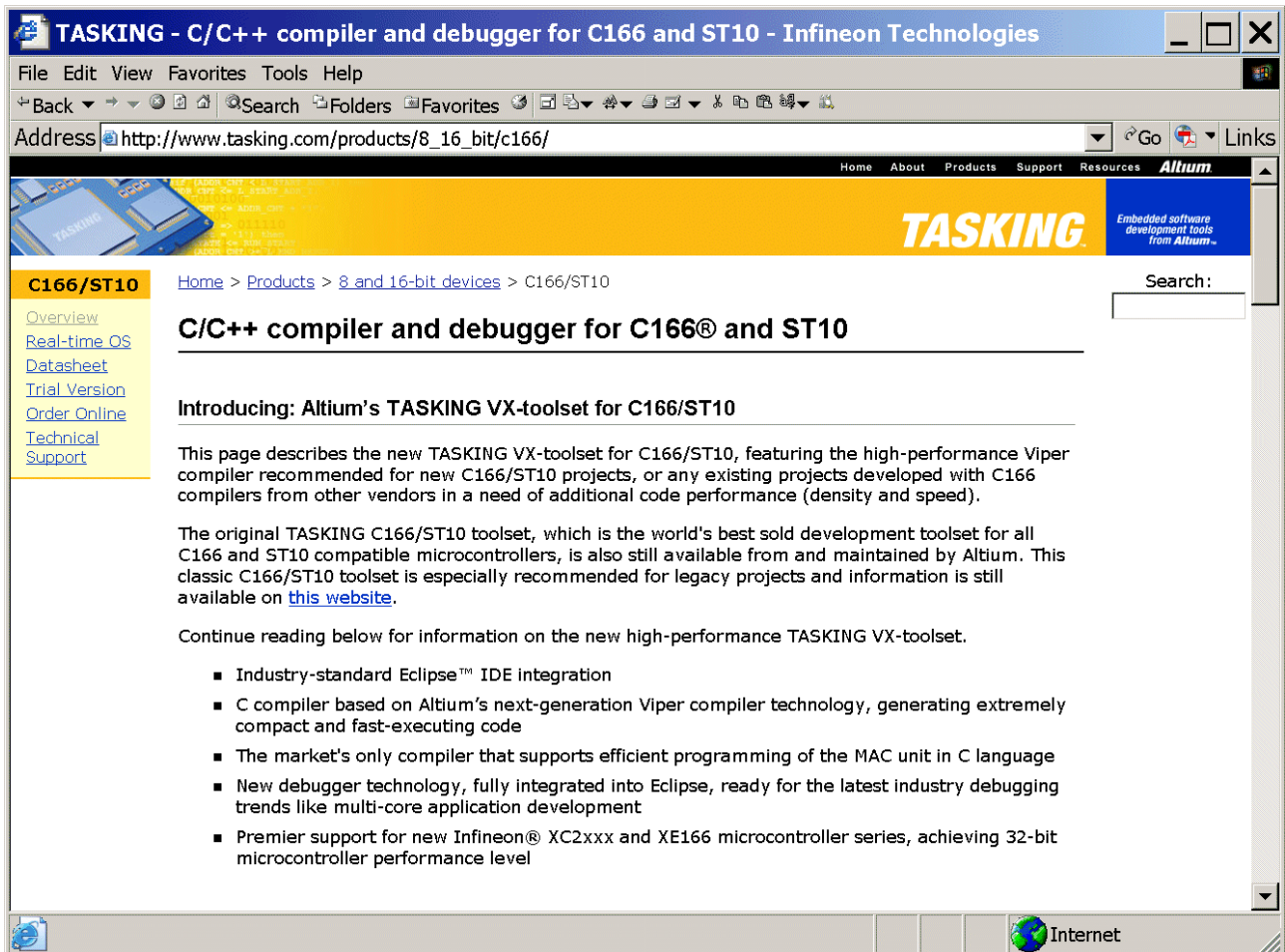
File - Exit

Save changes?

Click: **Yes**

#### 4.) Using the TASKING Development Tools (C-Compiler):

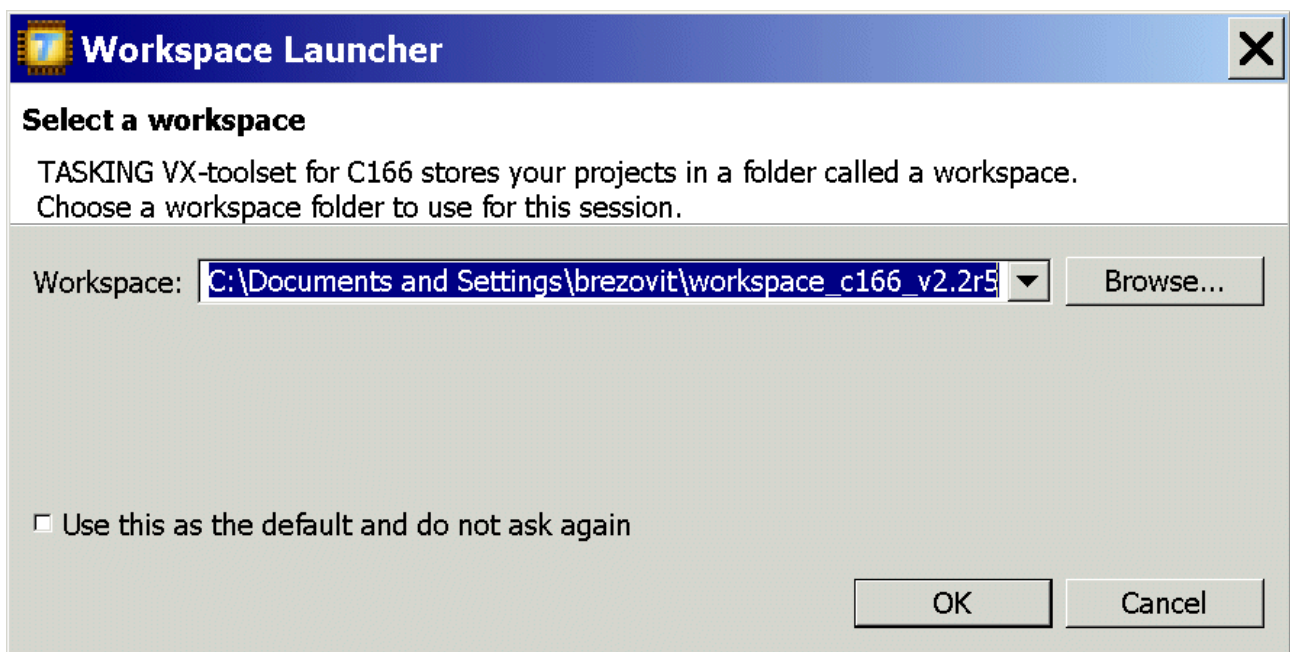
Purchase and install Altium's TASKING VX-toolset for C166/ST10 (<http://www.tasking.com>):

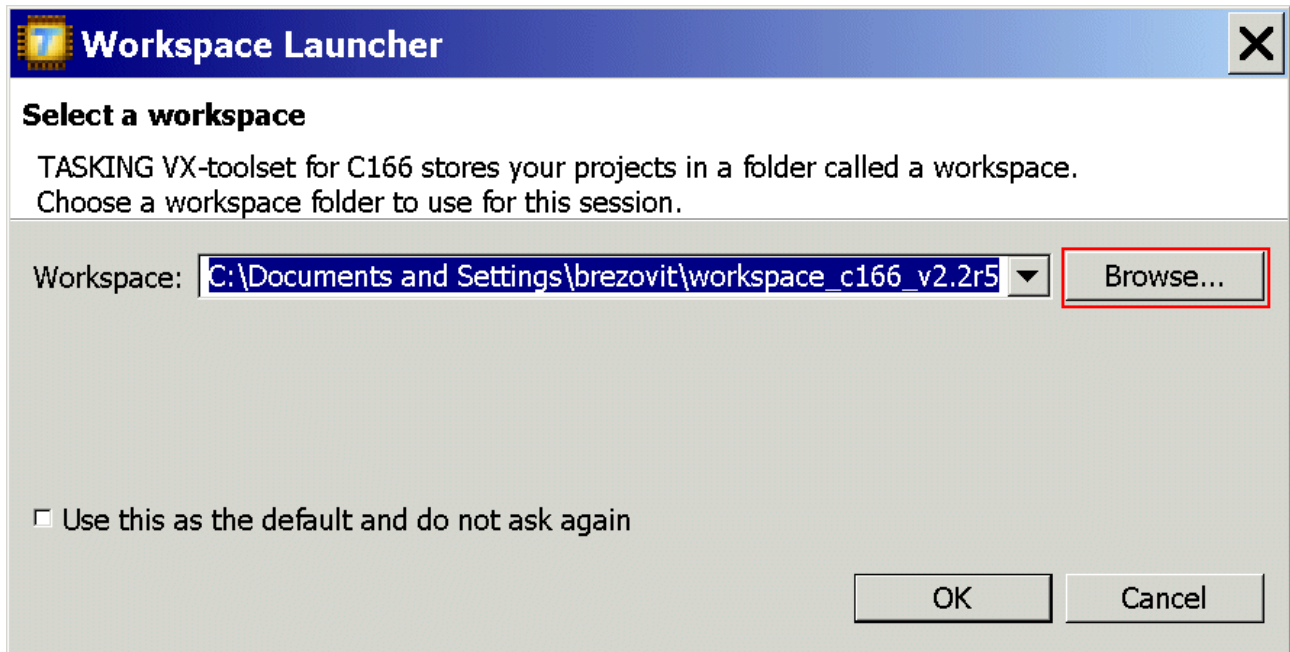






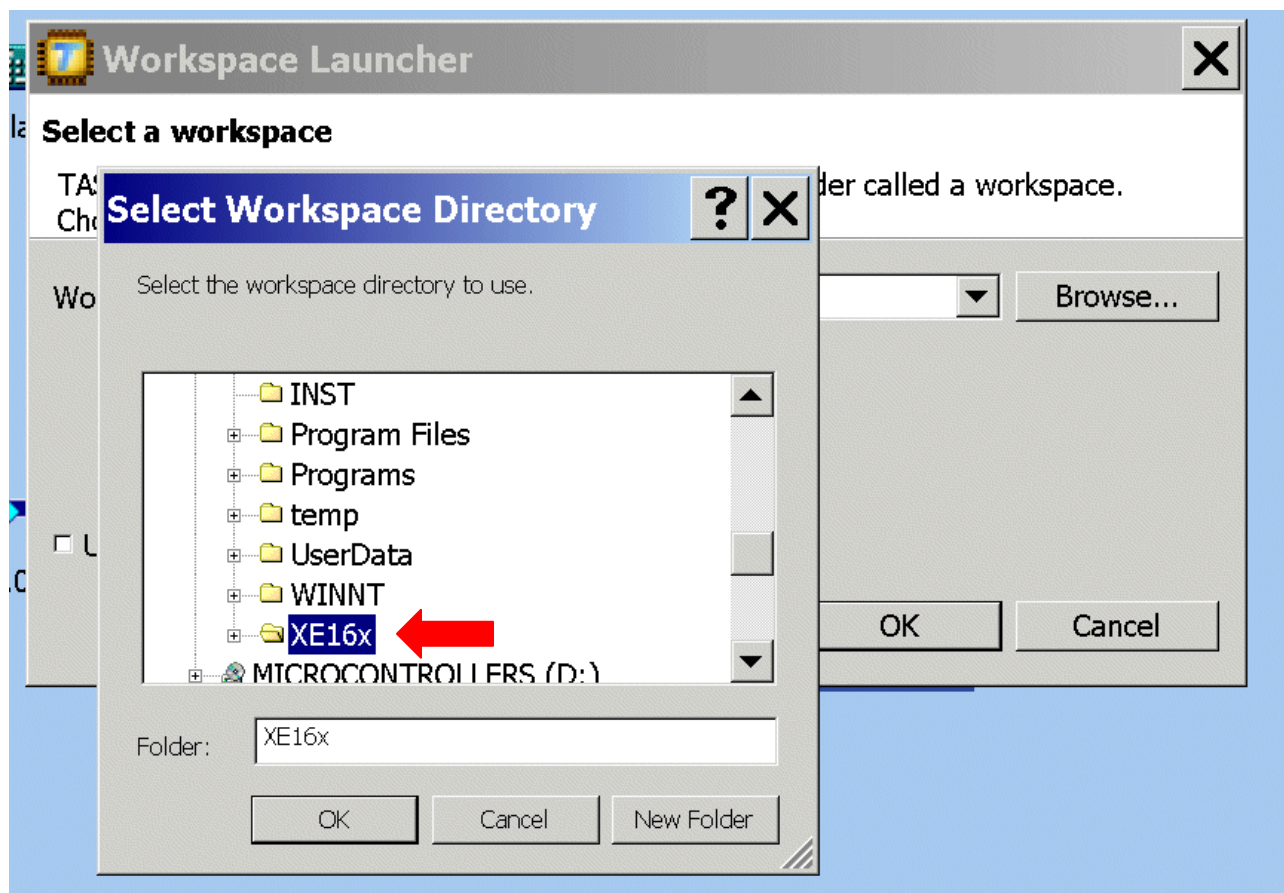
Start TASKING VX IDE and select the working directory:



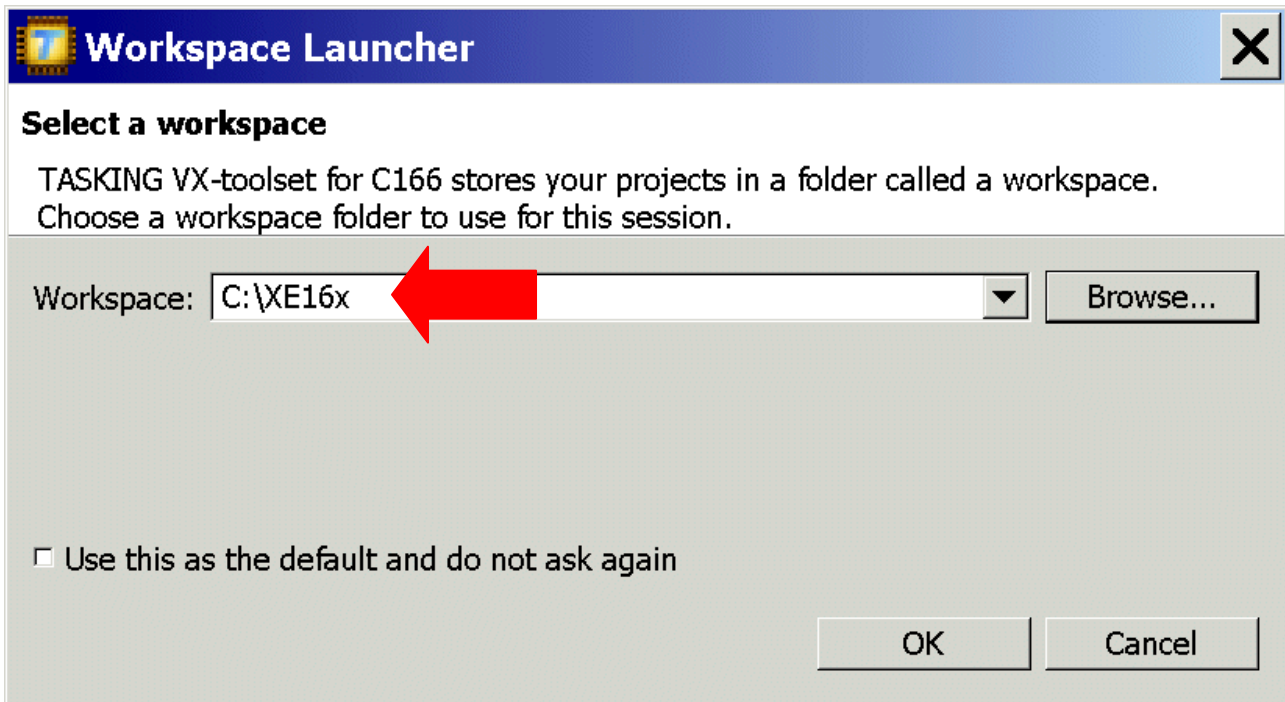


Click Browse

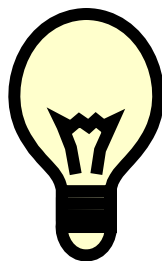
Select C:\XE16x



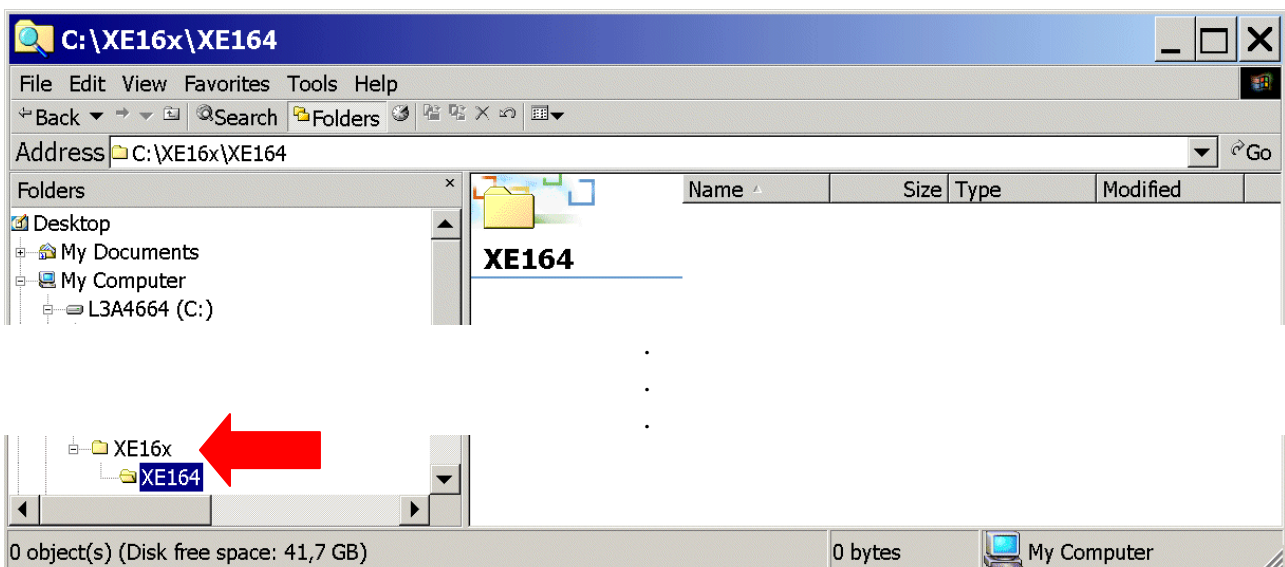
OK



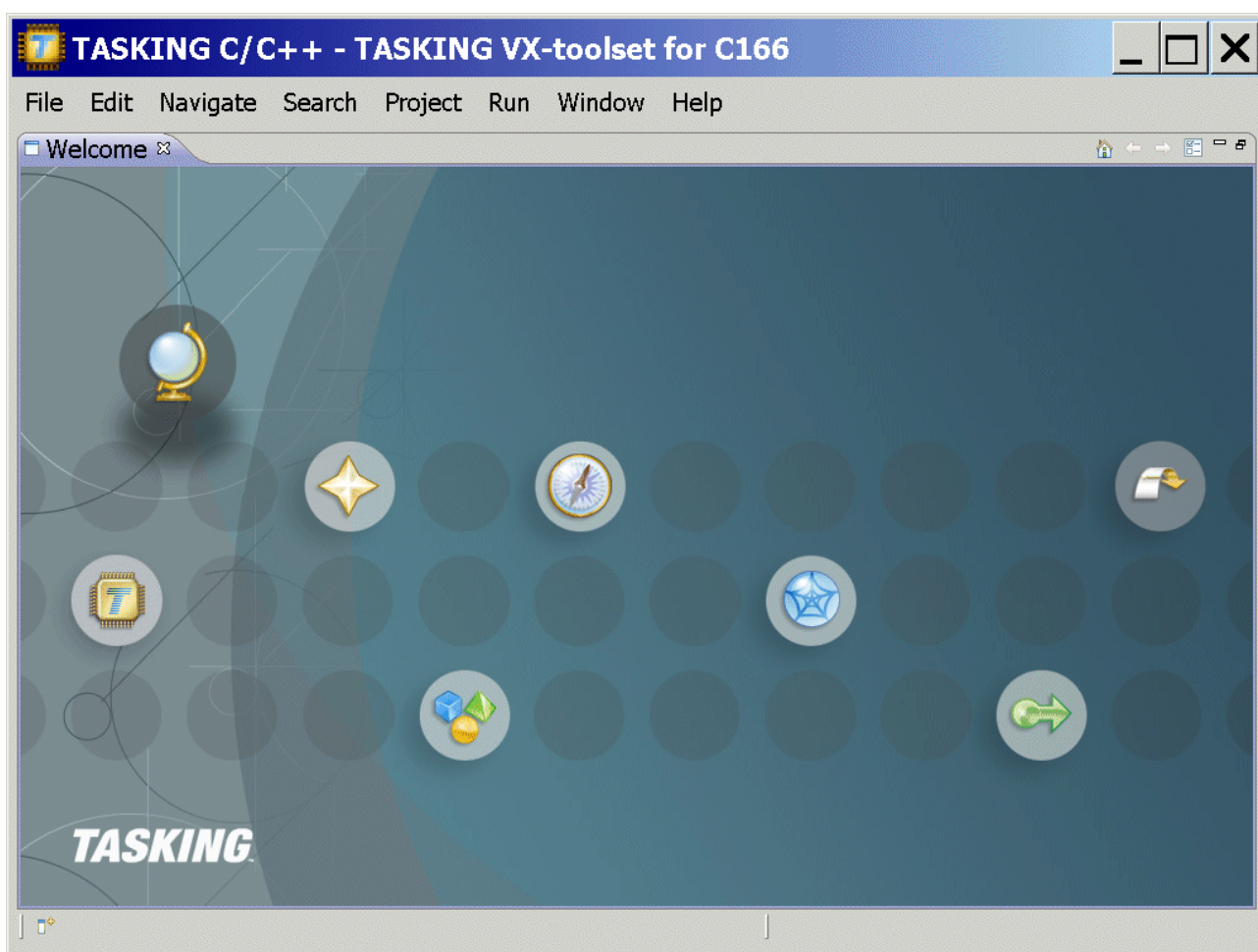
OK



Remember:

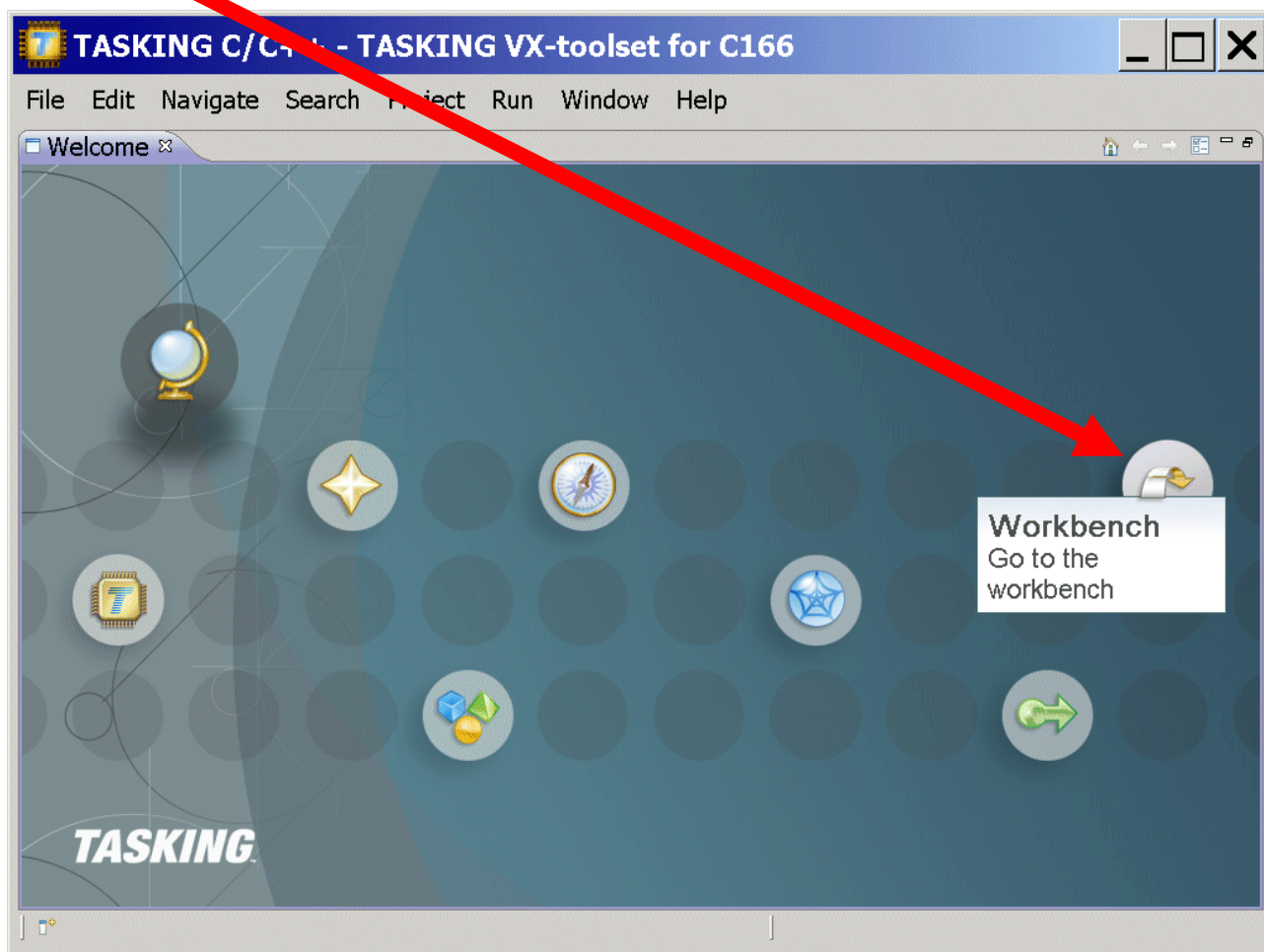




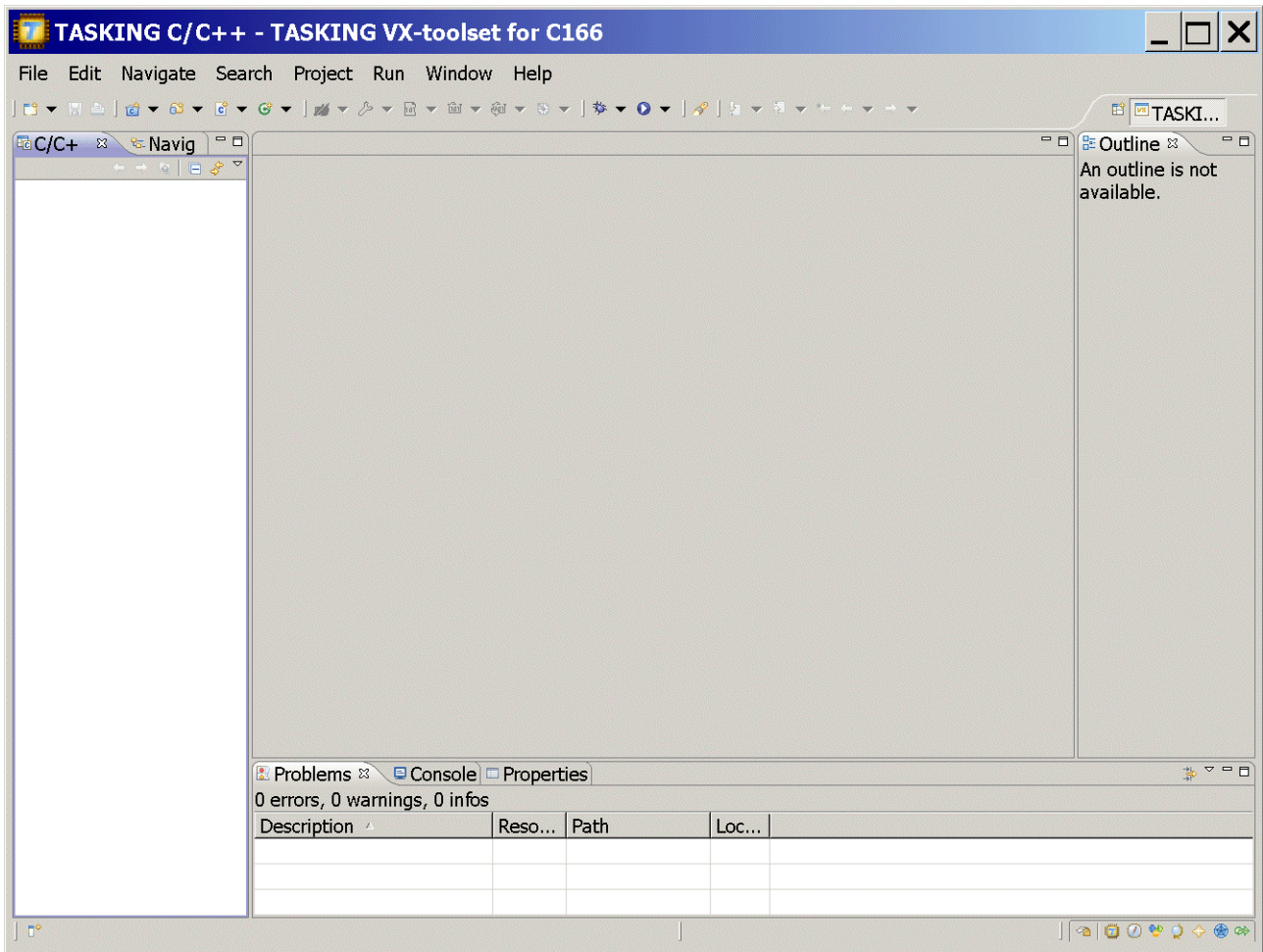


Create a new project and include the DAVE files:

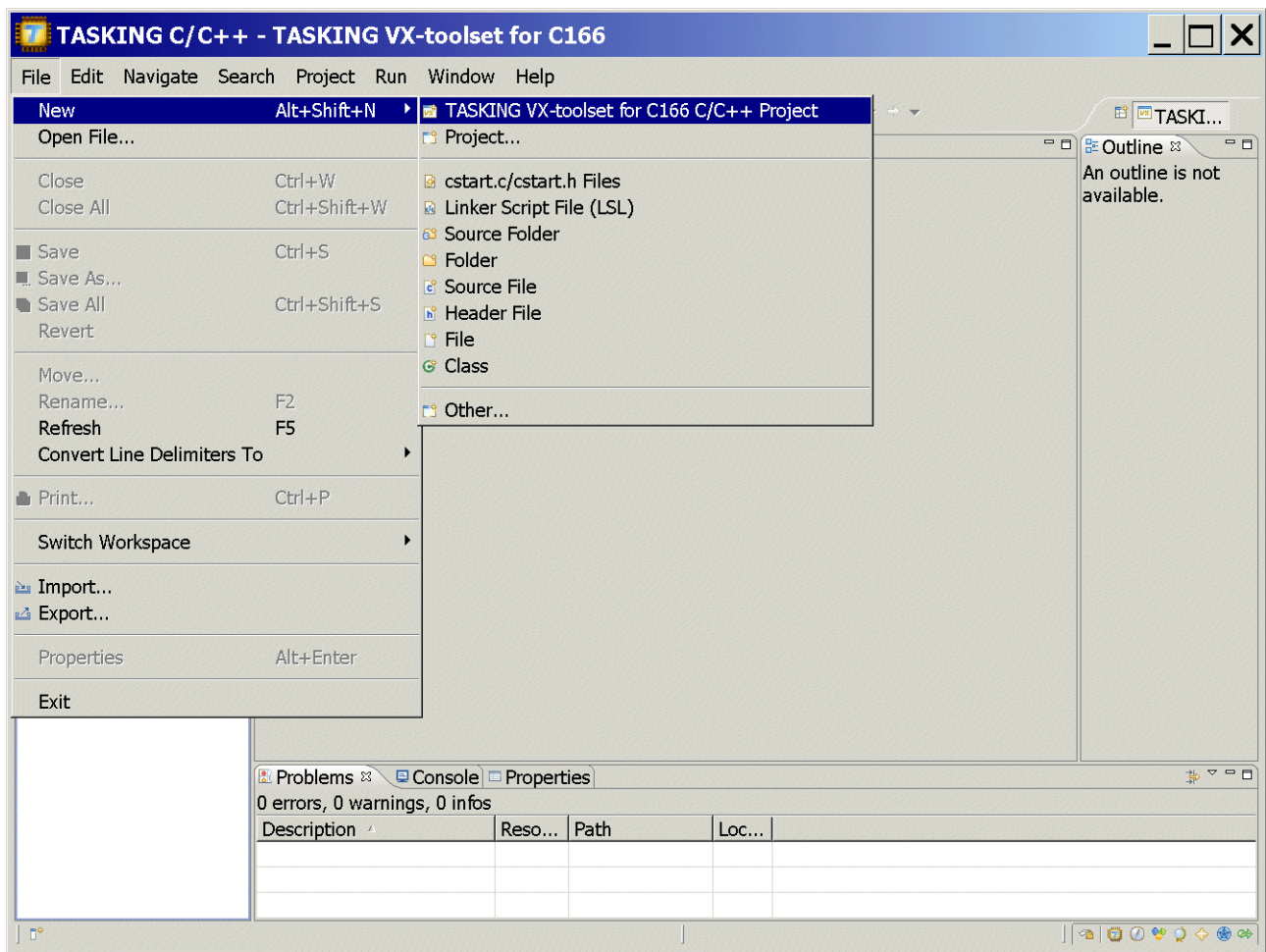
Click "Workbench Go to the workbench"



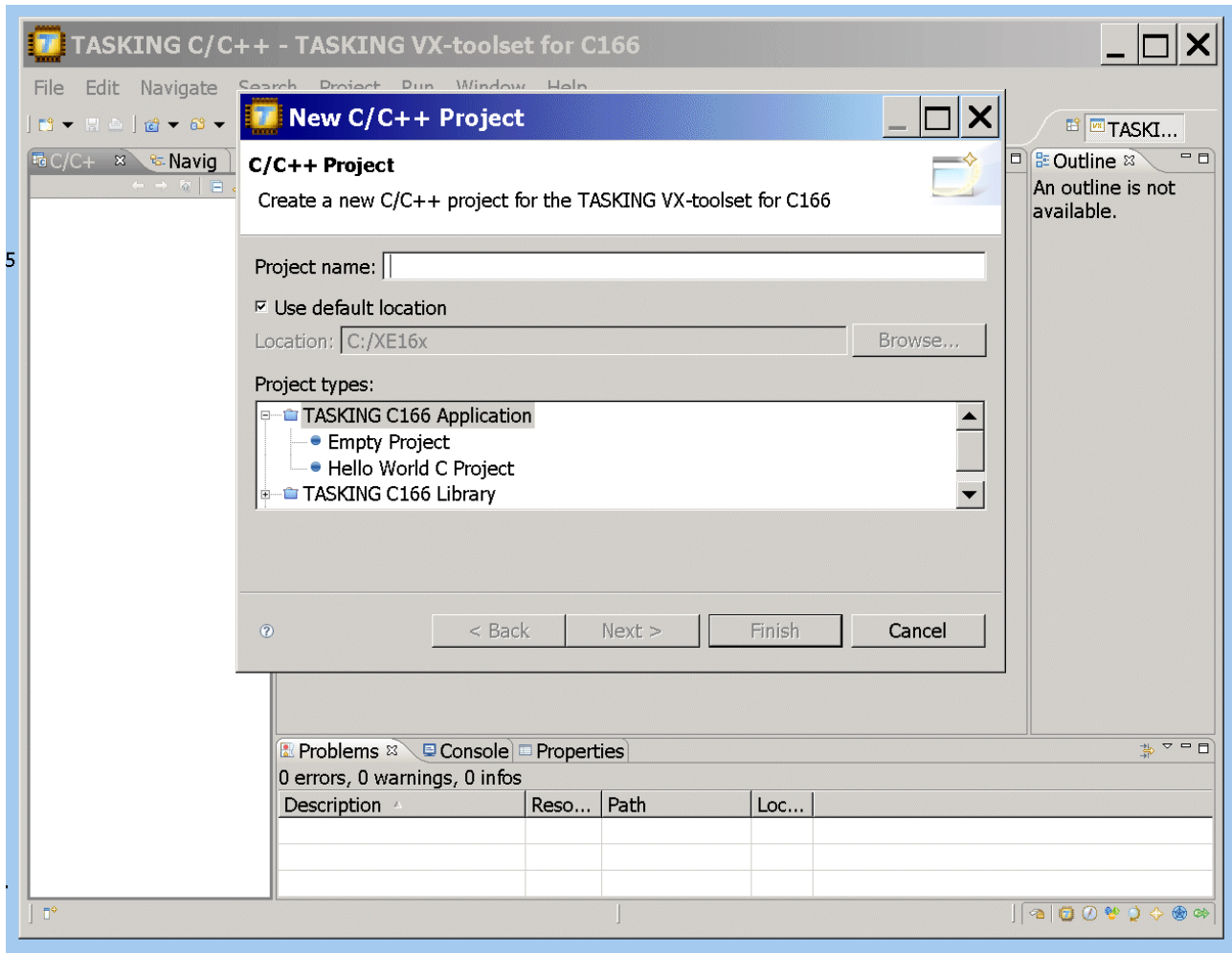




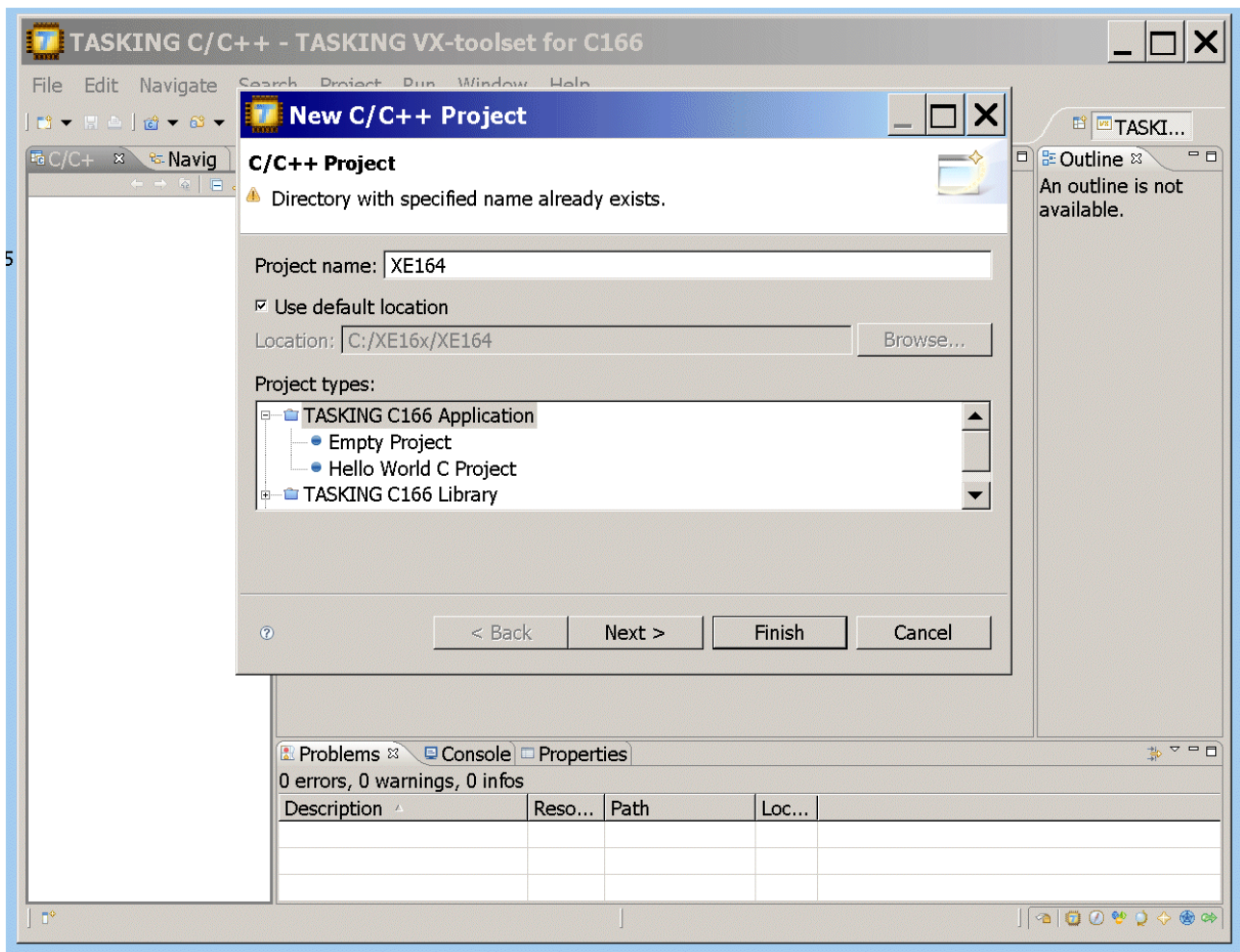
## File – New – TASKING VX-toolset for C166 C/C++ Project



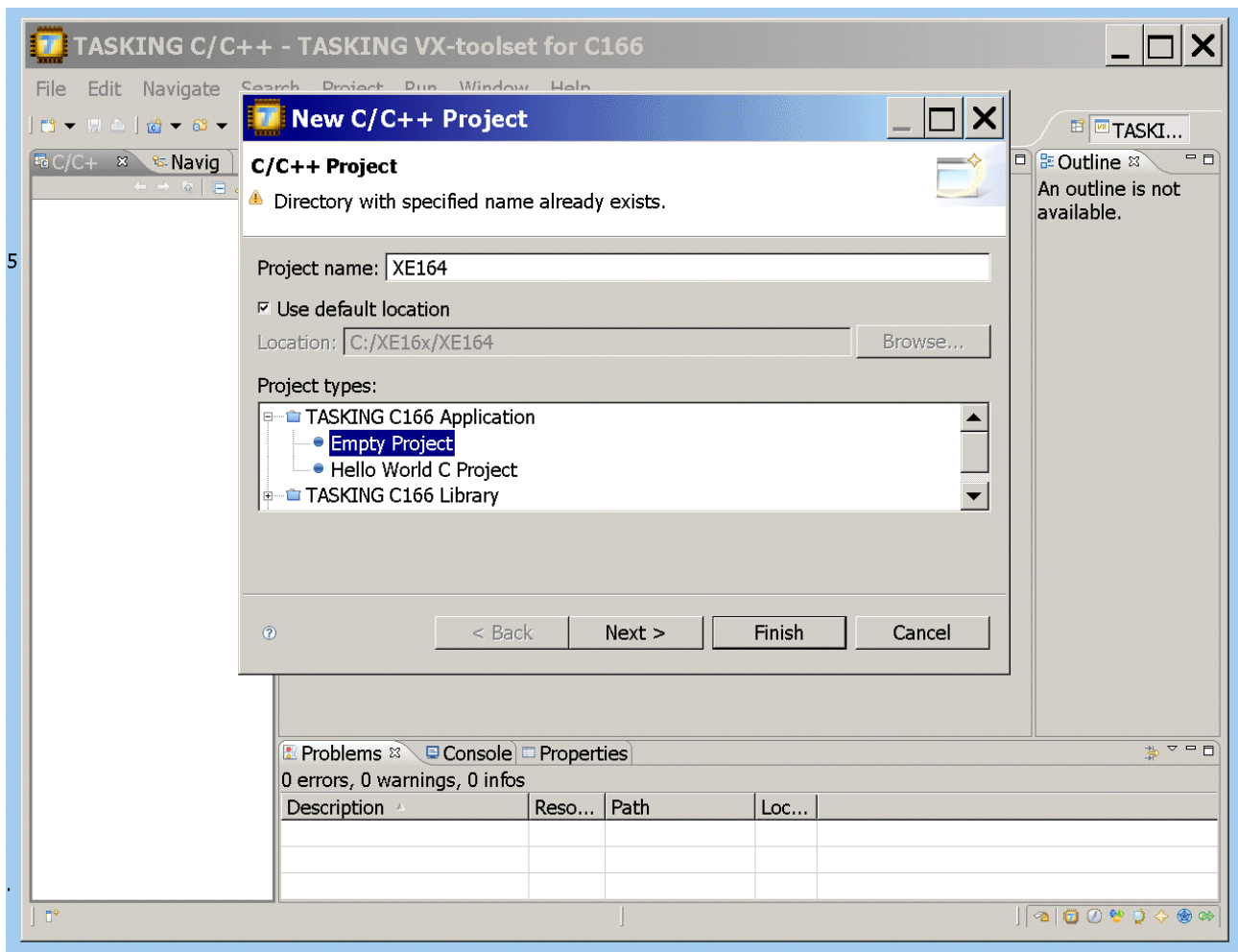




Project Name: insert XE164



Project types: select Empty Project

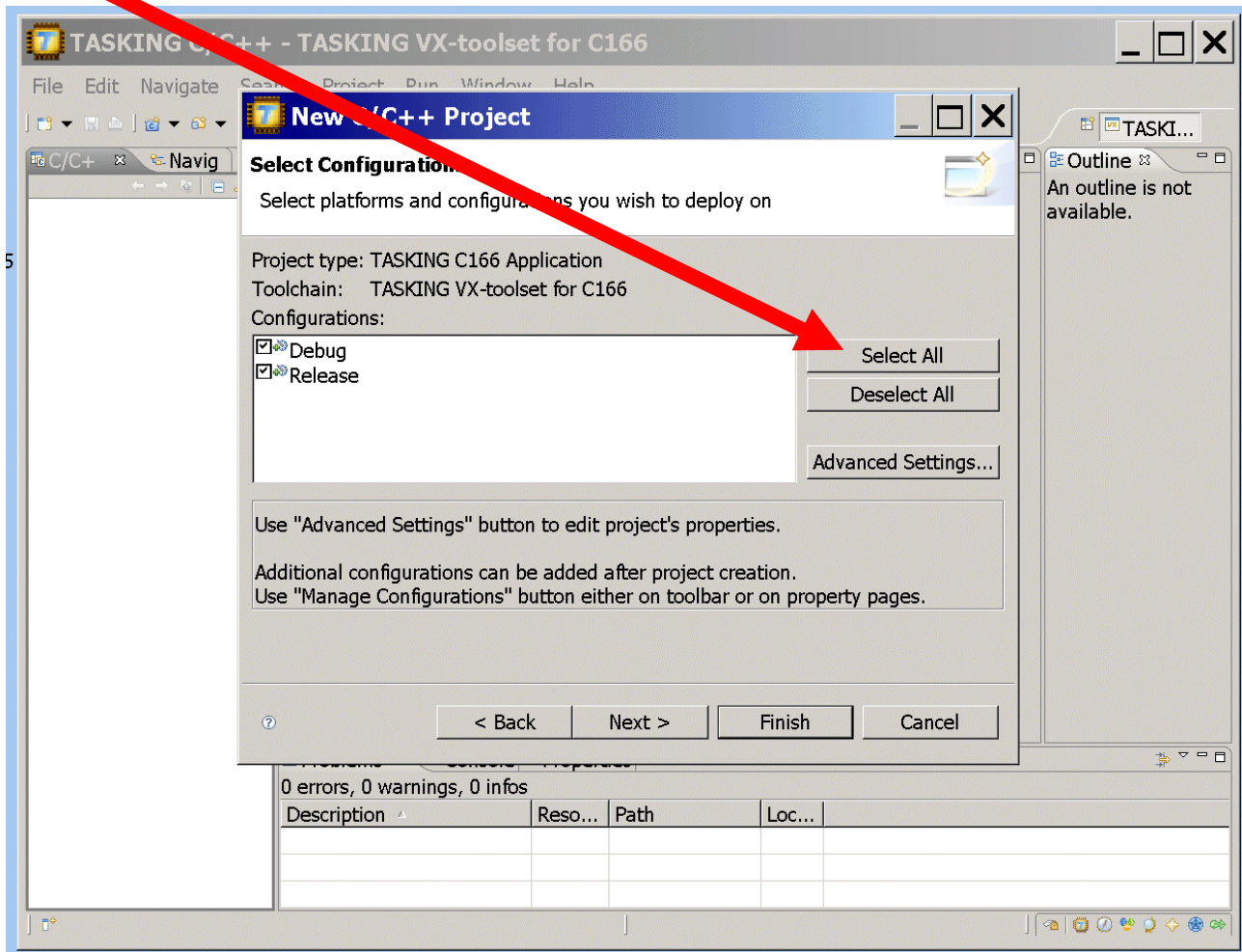


Click Next>



Configurations: check ✓ Debug  
Configurations: check ✓ Release

Or click Select All

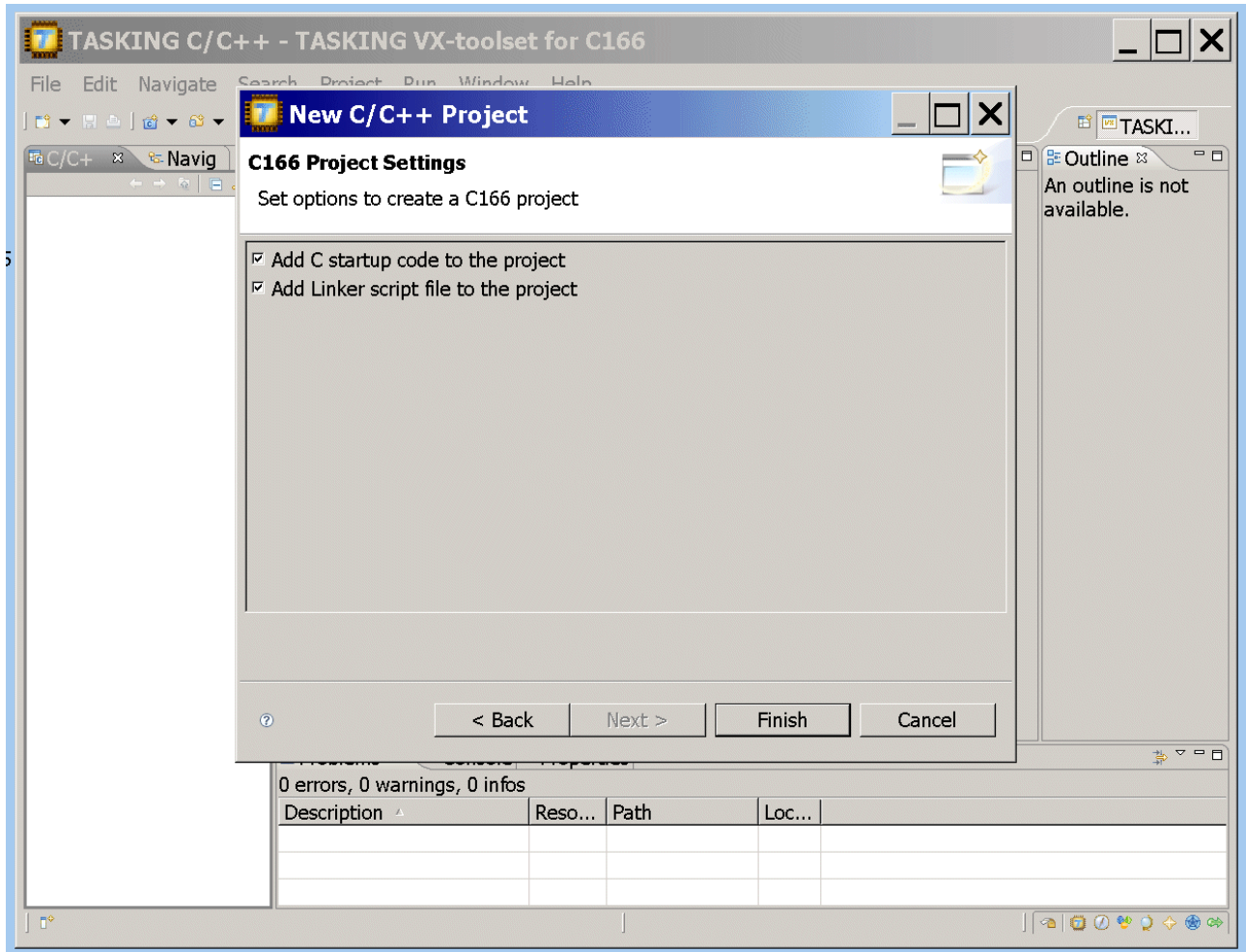


Click Next>

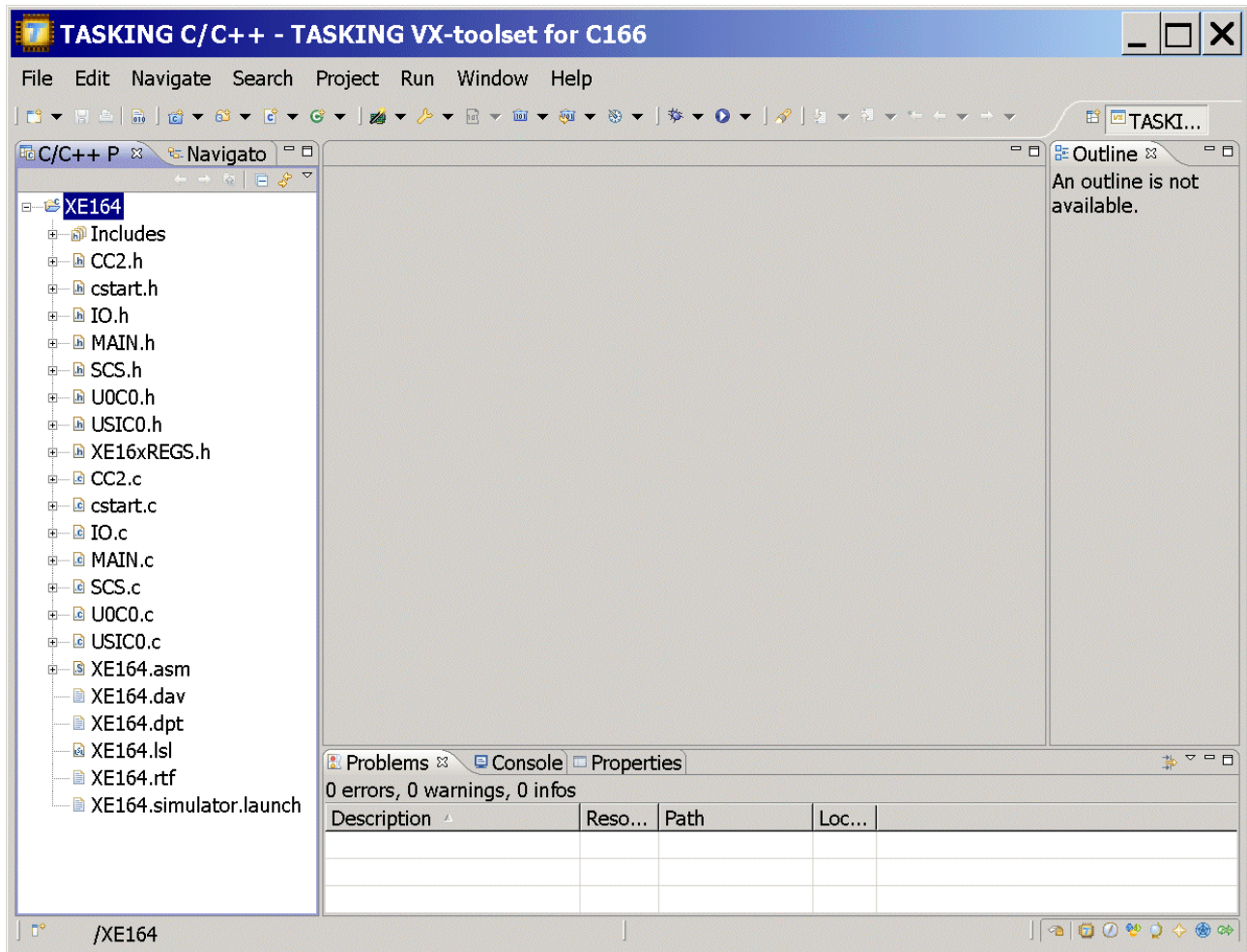


Set options to create a C166 project: check ☒ Add C startup code to the project

Set options to create a C166 project: check ☒ Add Linker script file to the project



Click Finish

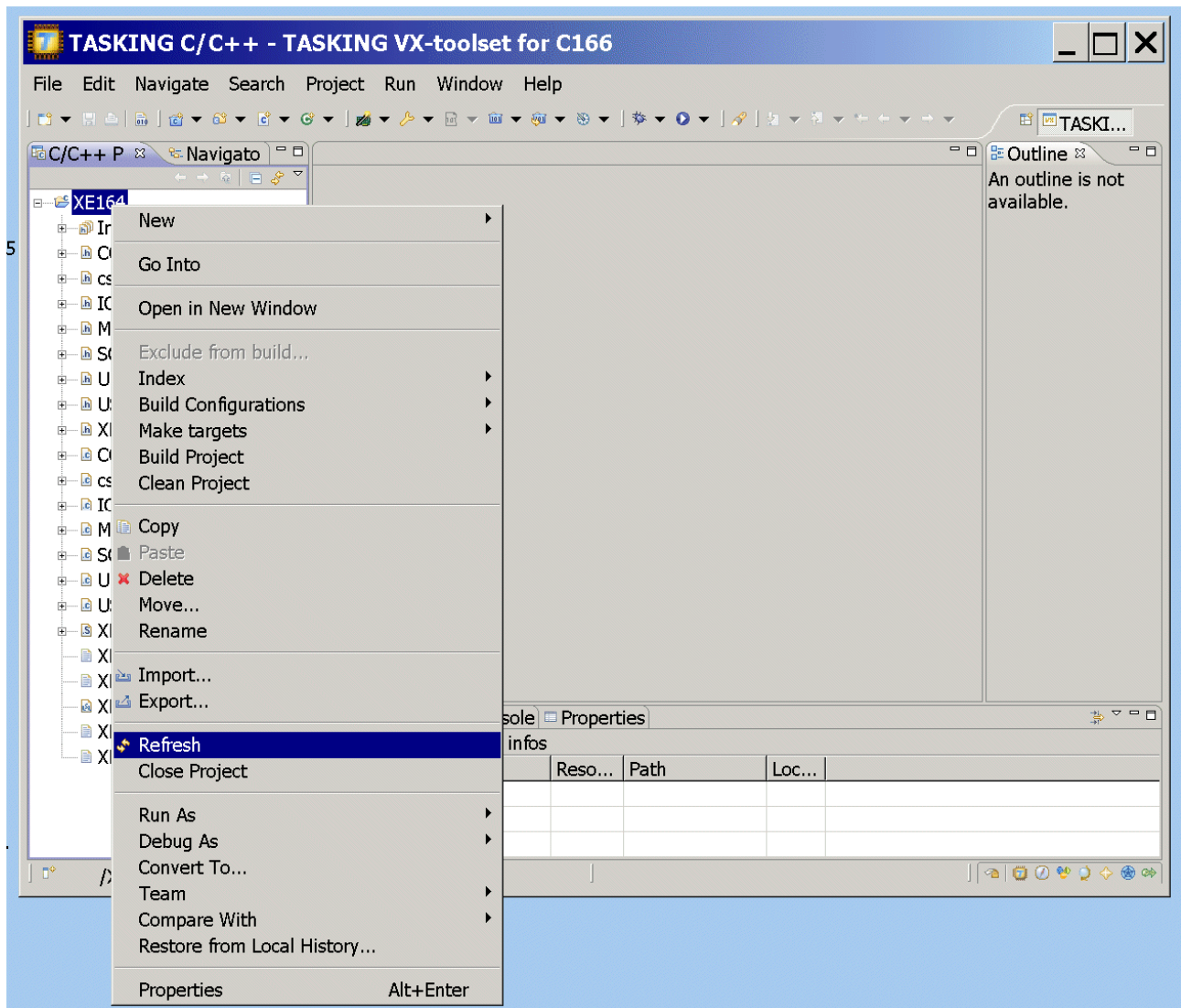


mouse position:

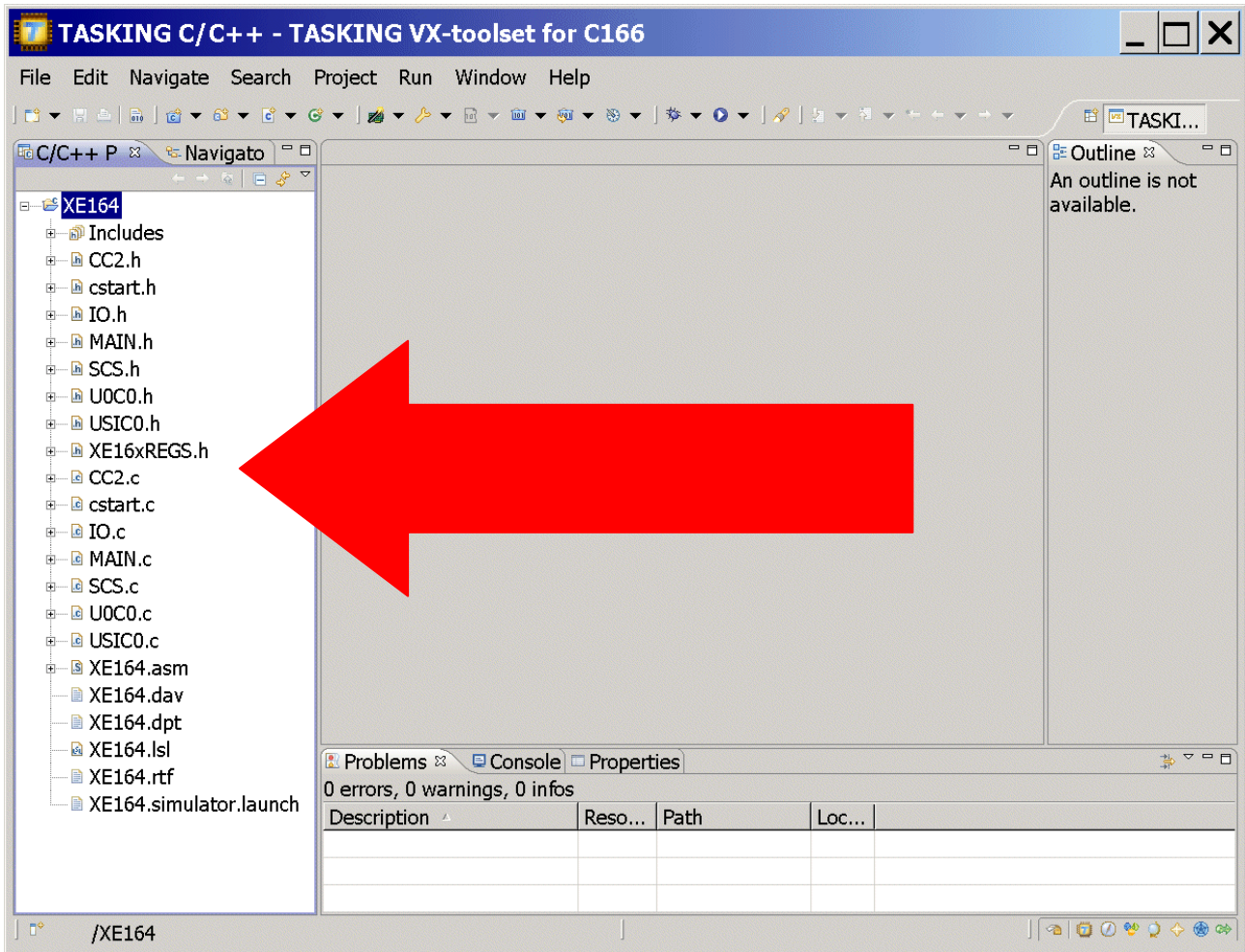
@ XE164

click right mouse button

click Refresh

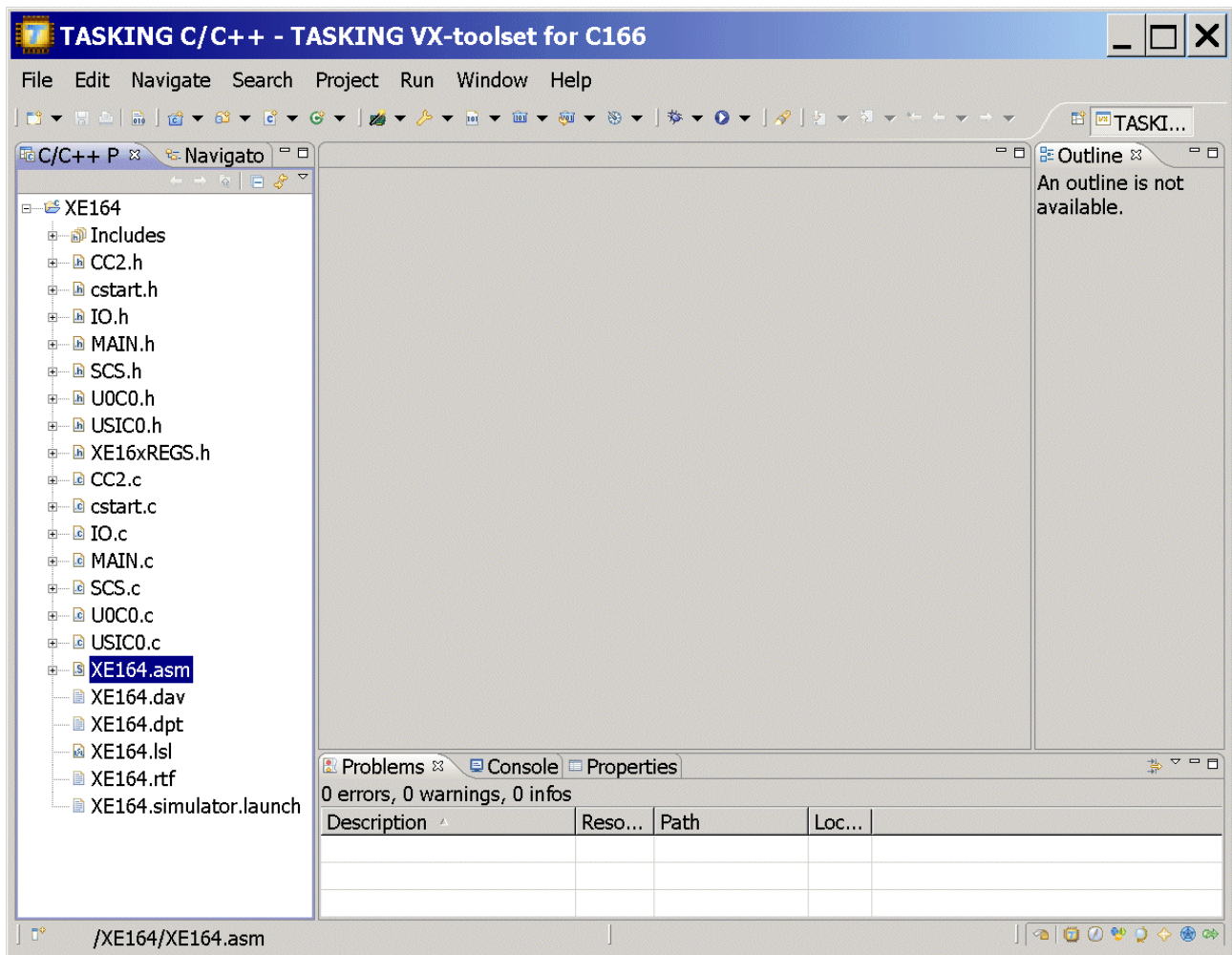




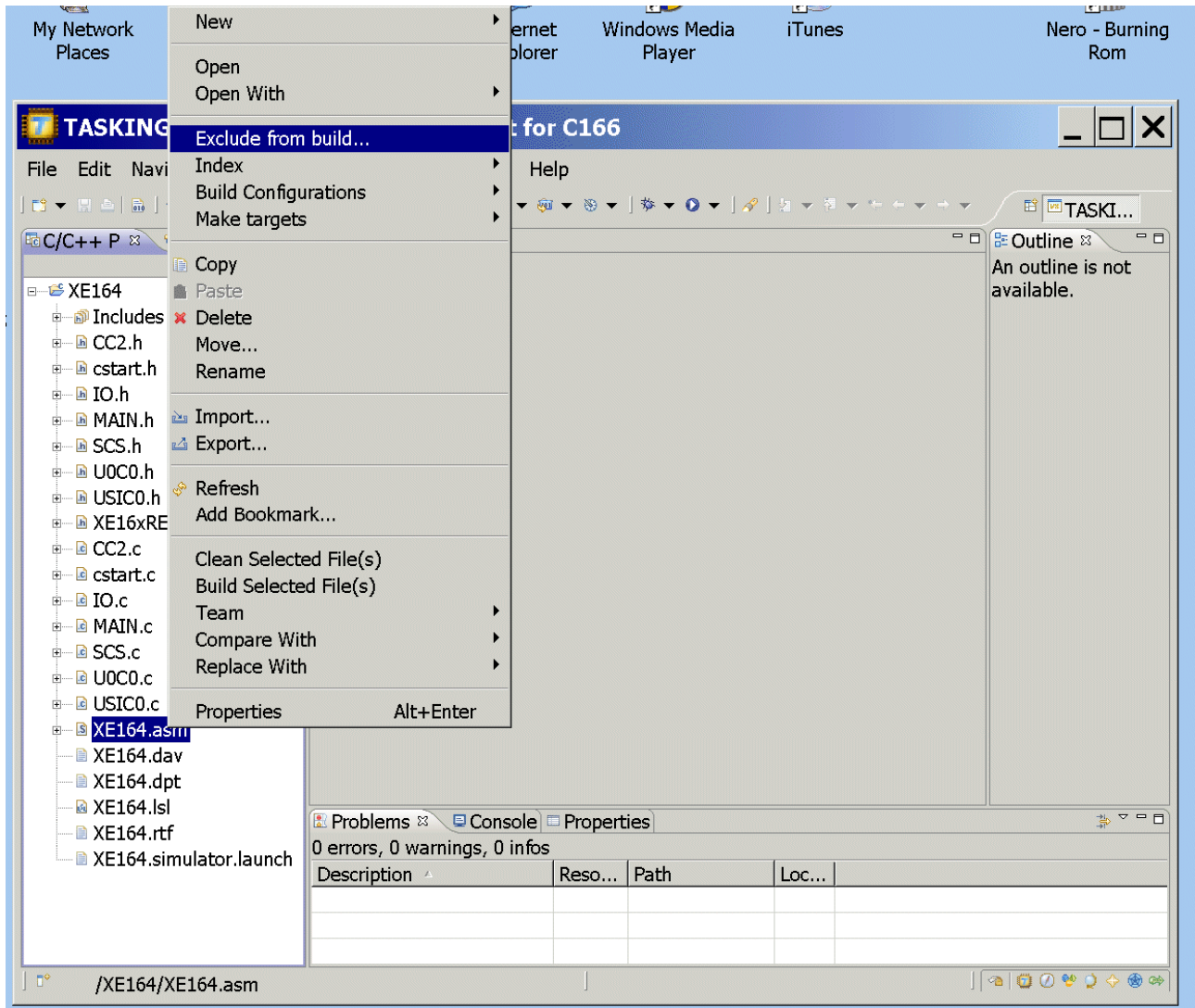


Exclude the (DAvE) XE164.asm file:

Click the right mouse button @ XE164.asm



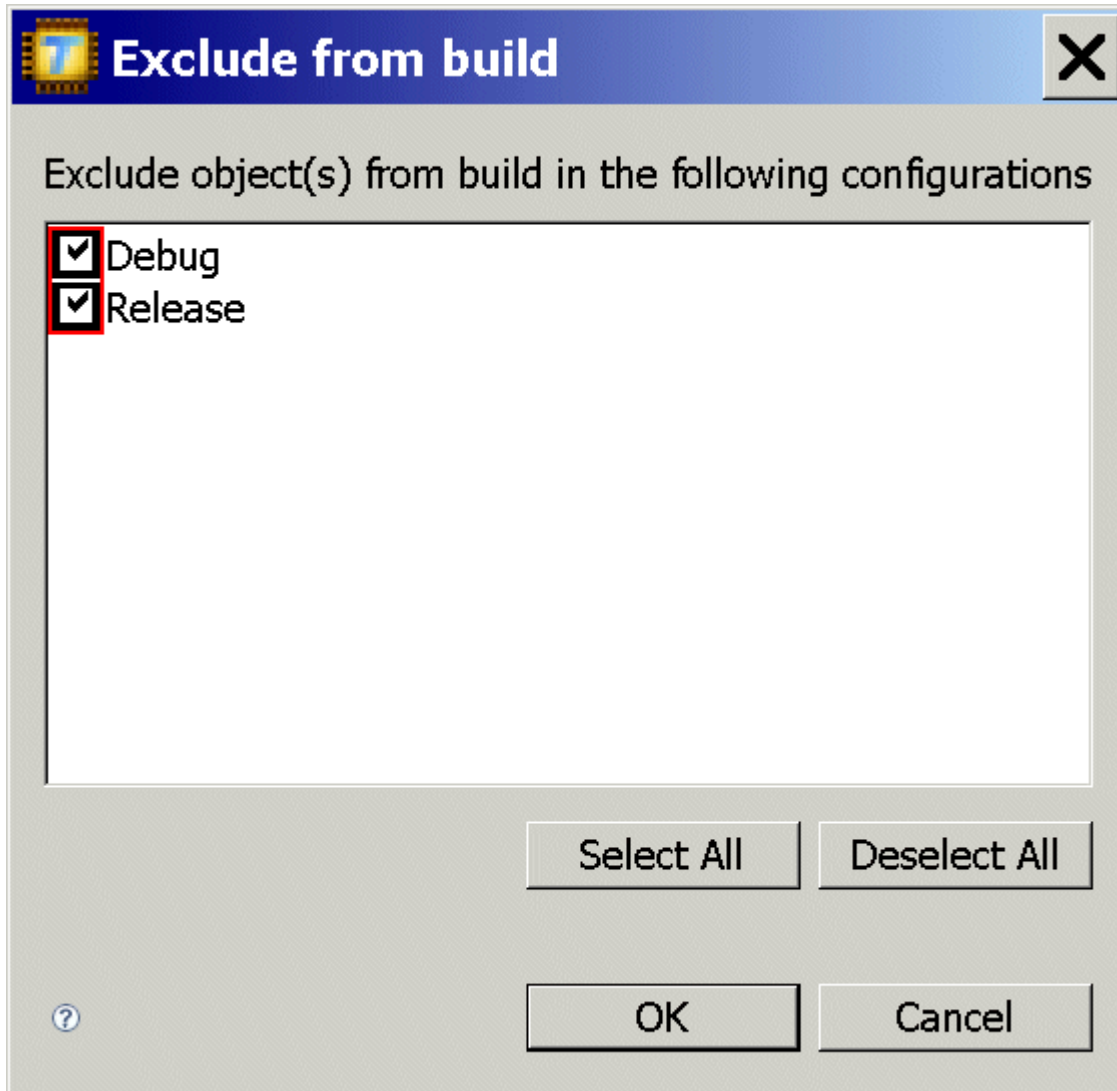




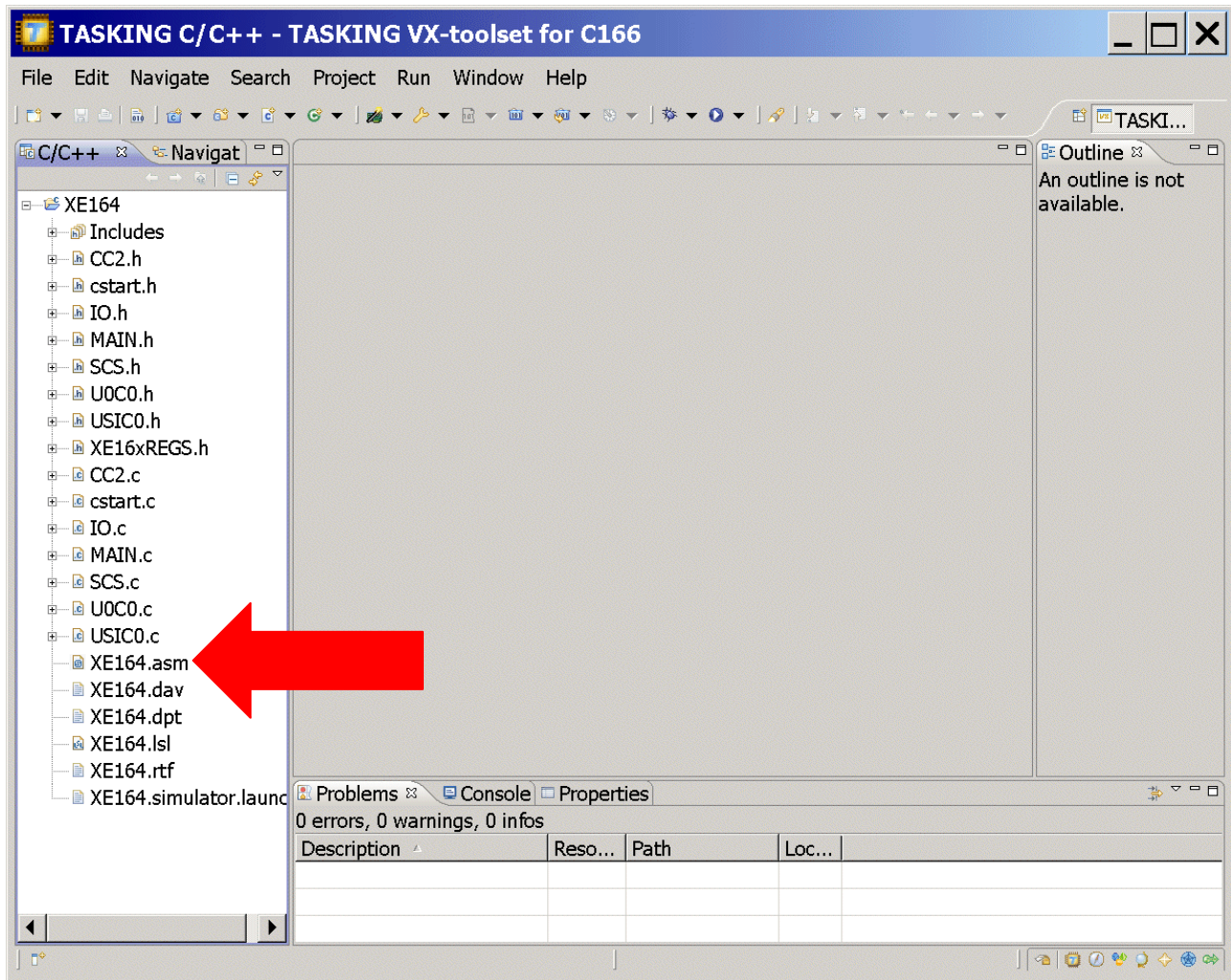
**Click** Exclude from build...



Click/tick ☒ Debug  
Click/tick ☒ Release

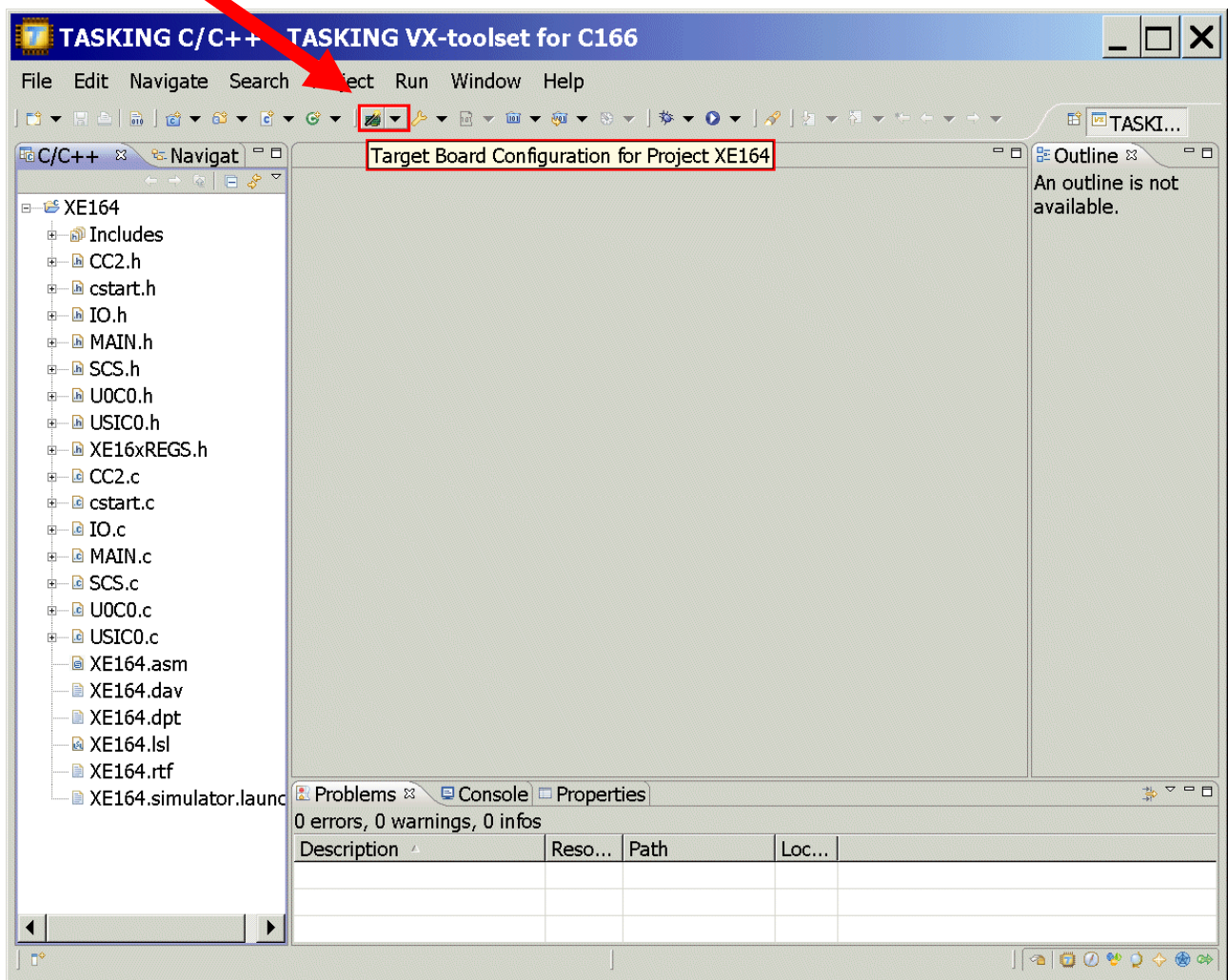


OK



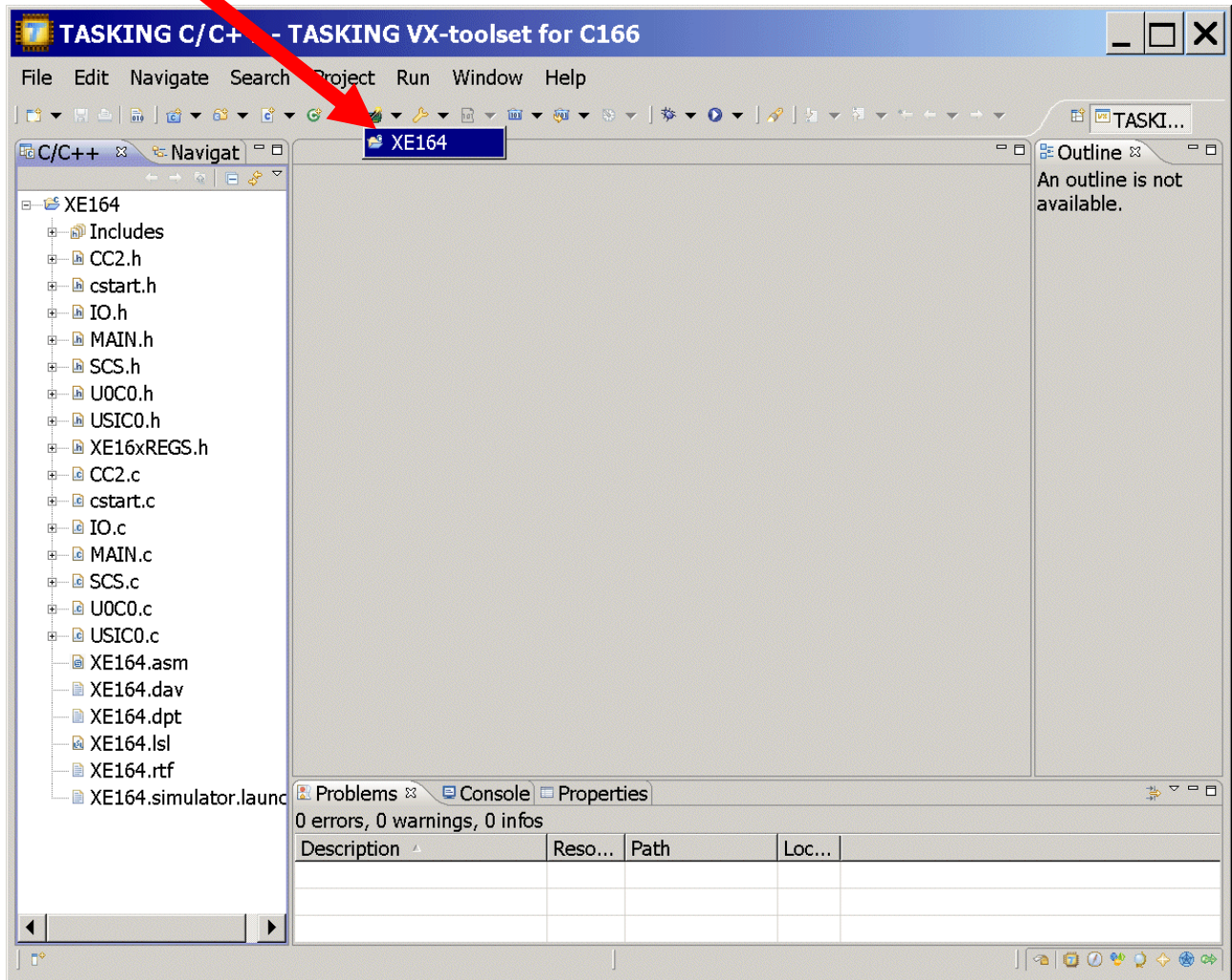
## Configure the Target Board:

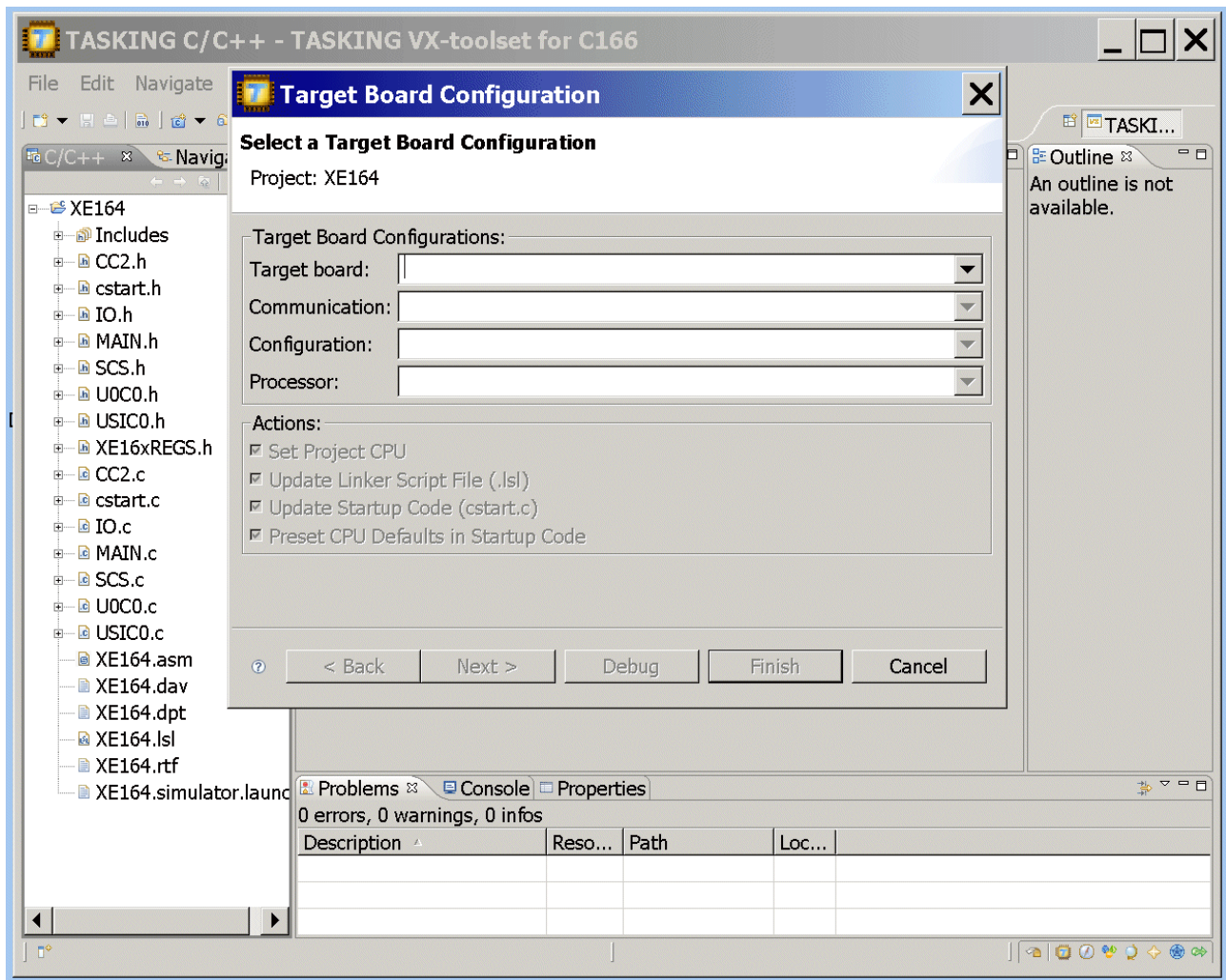
Click  Target Board Configuration for Project XE164





Click  XE164

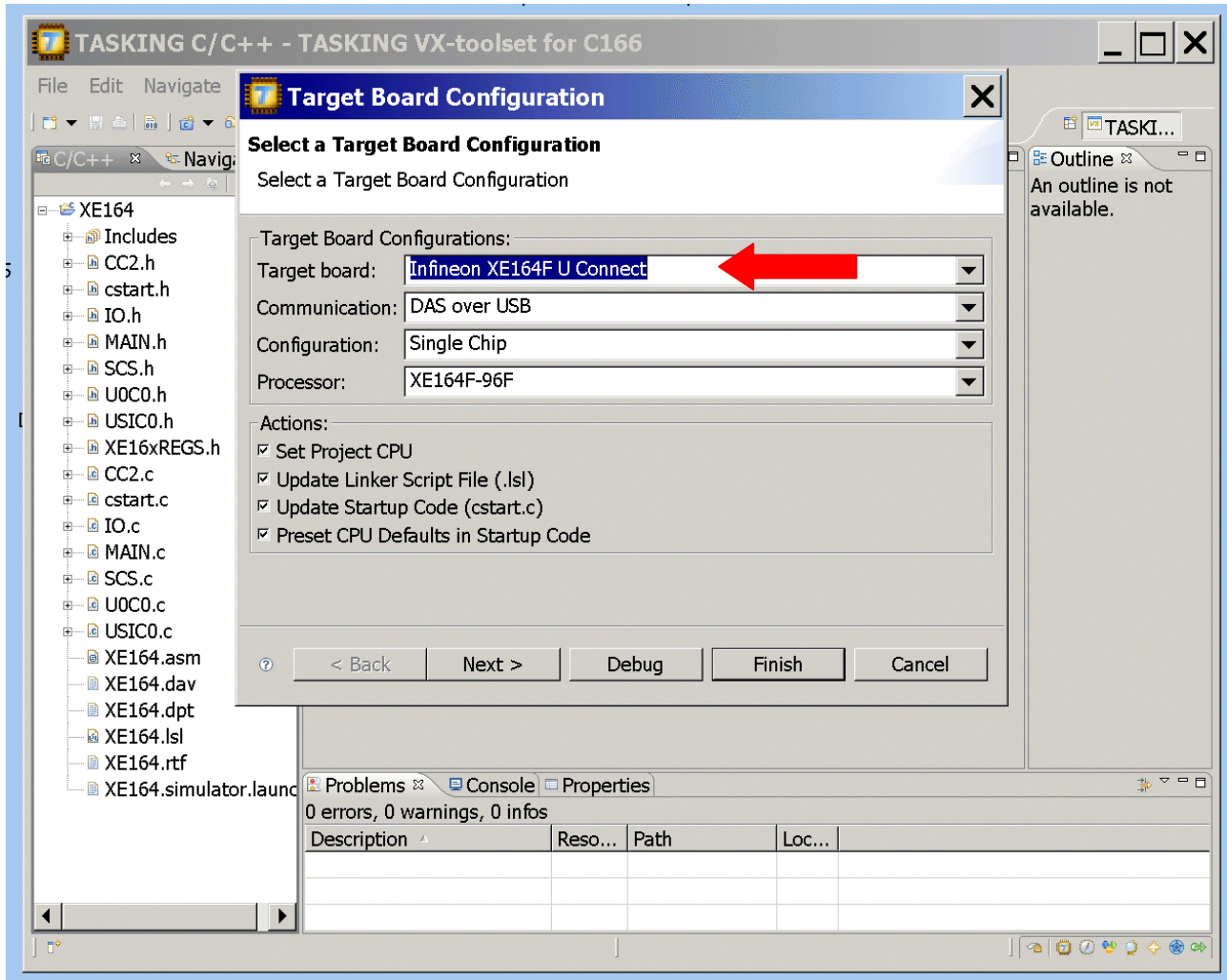






Select a Target Board Configuration:

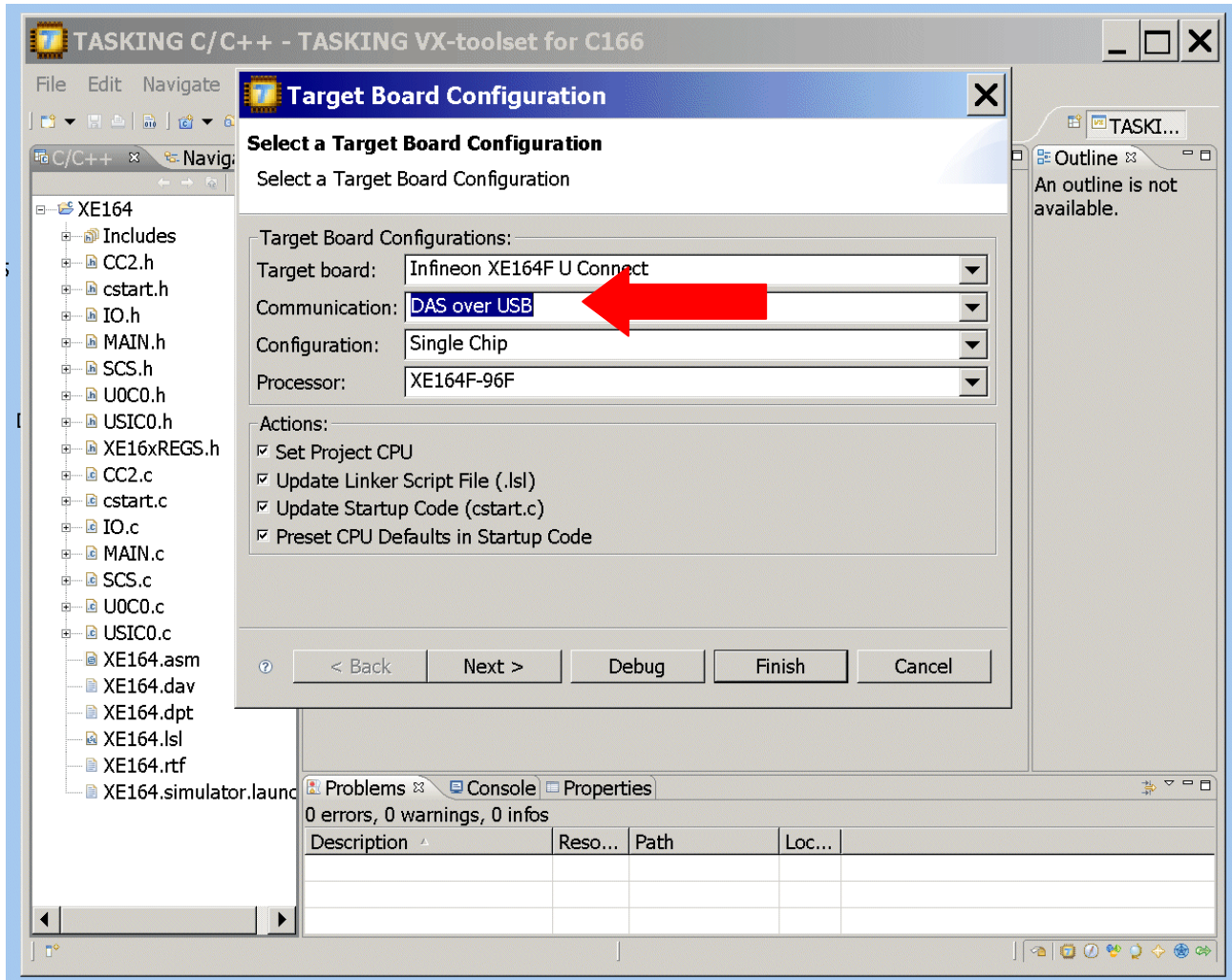
Target Board Configurations: Target board: select Infineon XE164F U Connect



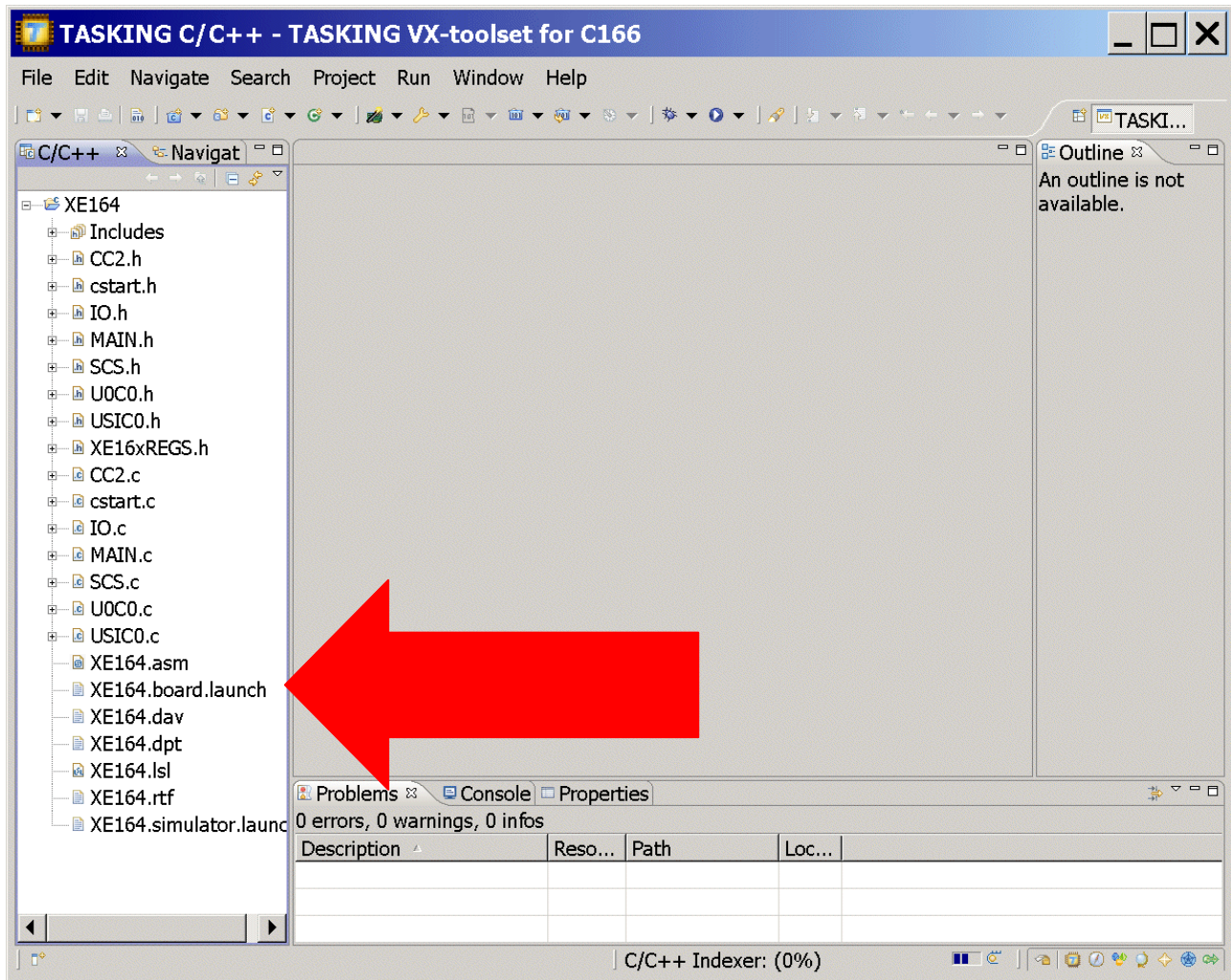


Select a Target Board Configuration:

Target Board Configurations: Communication: select/check DAS over USB



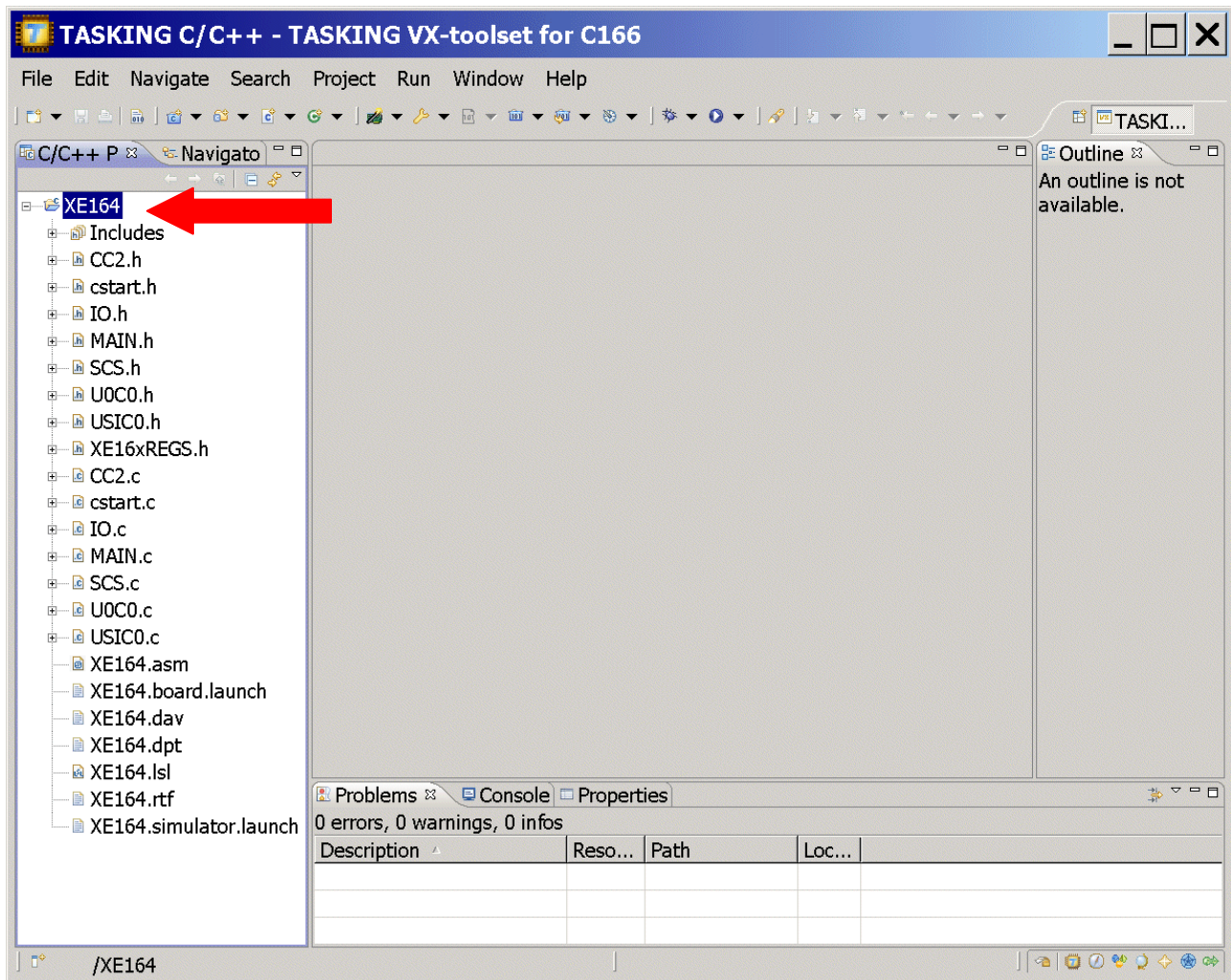
Click Finish





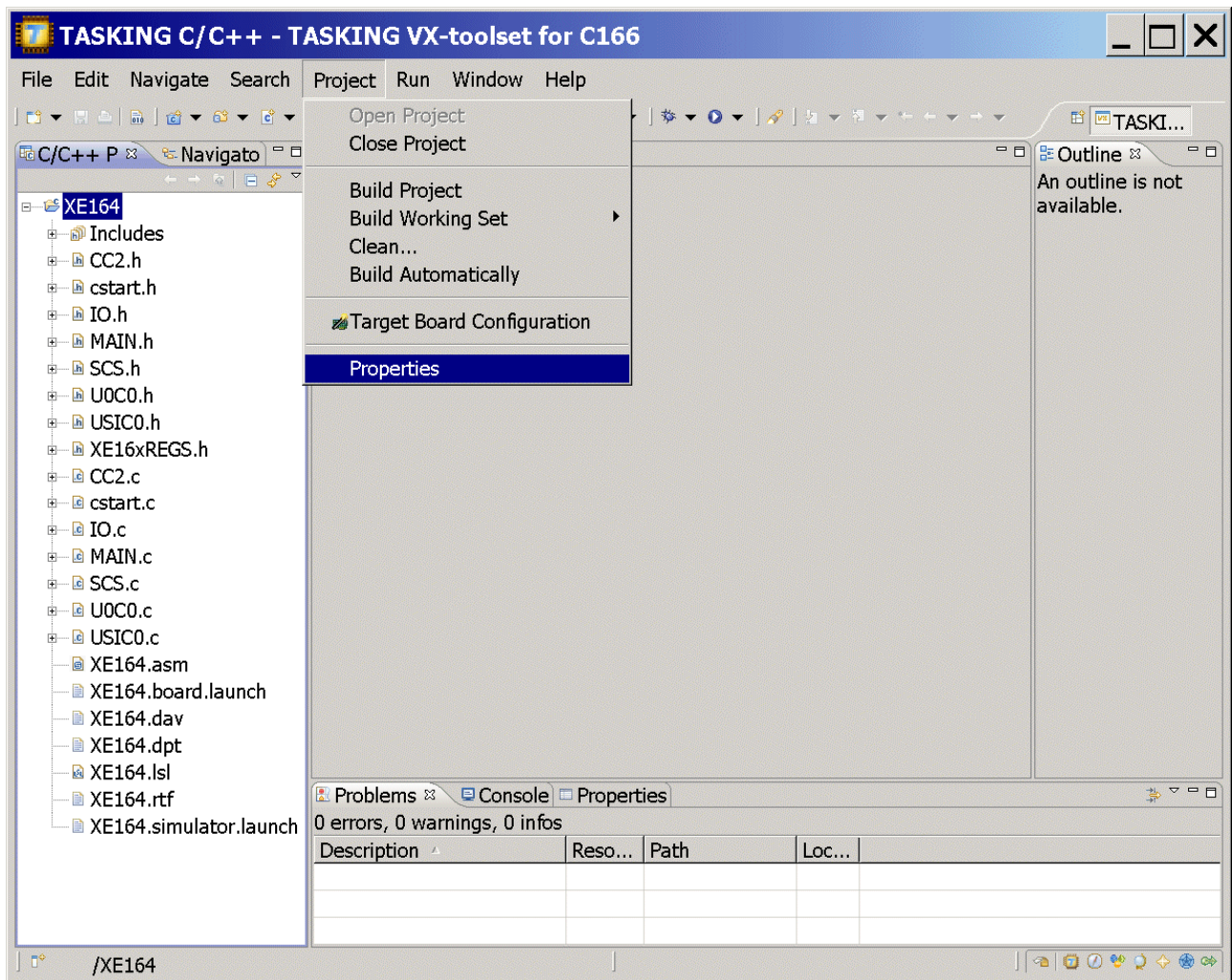
Configure the Project (unselect Automatic inclusion of .sfr file):

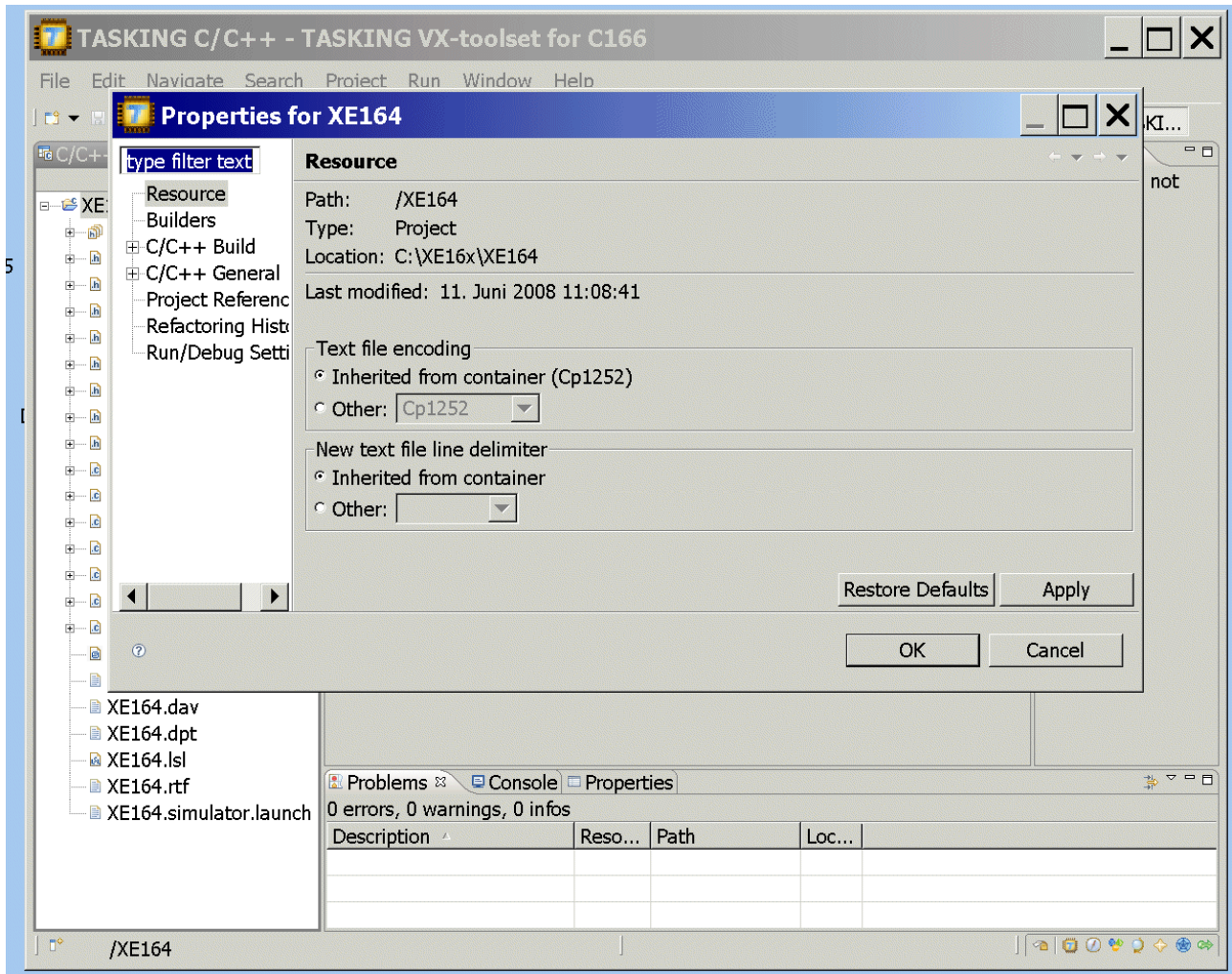
Click @ XE164





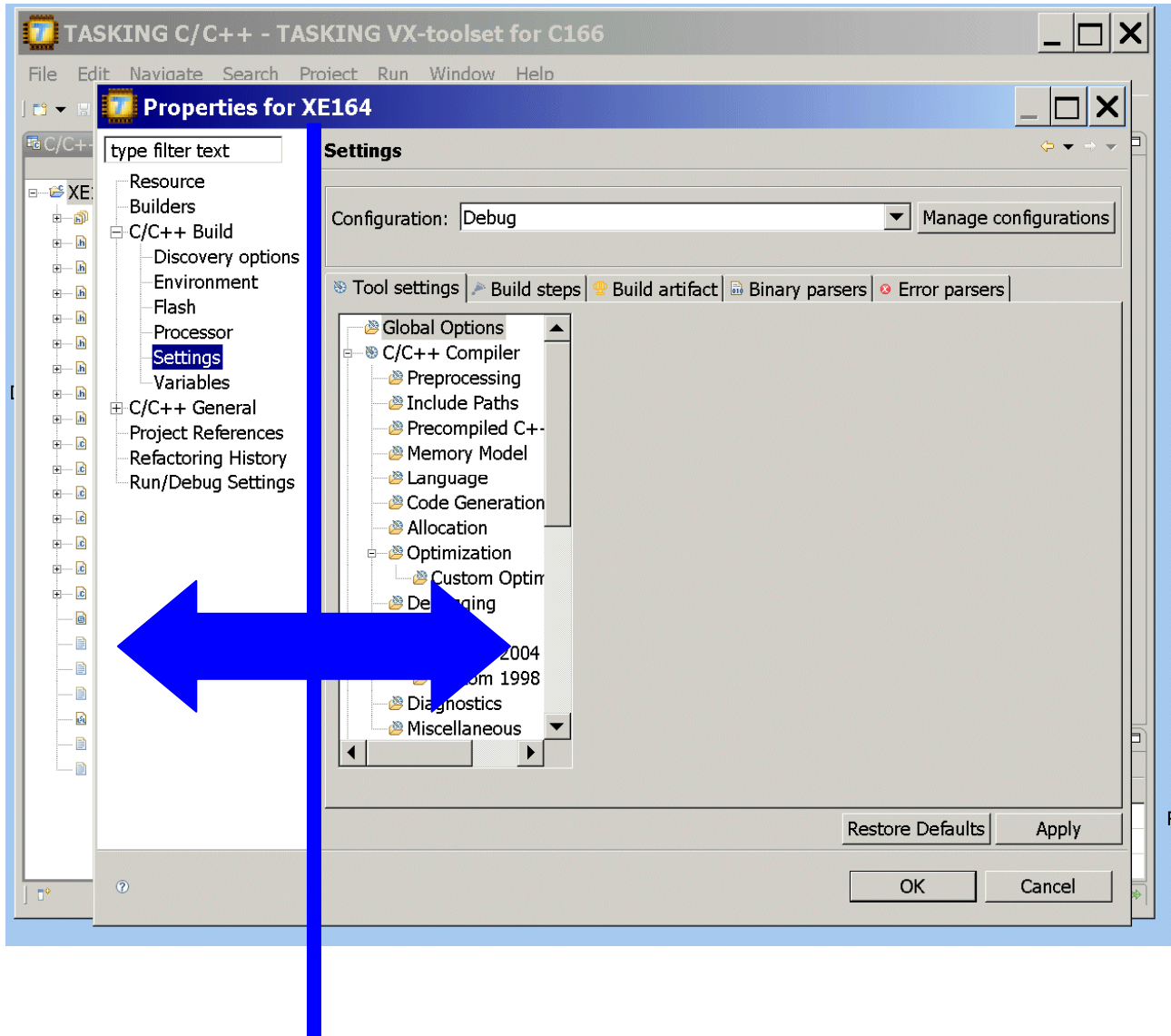
## Project - Properties







Expand C/C++ Build  
Click Settings

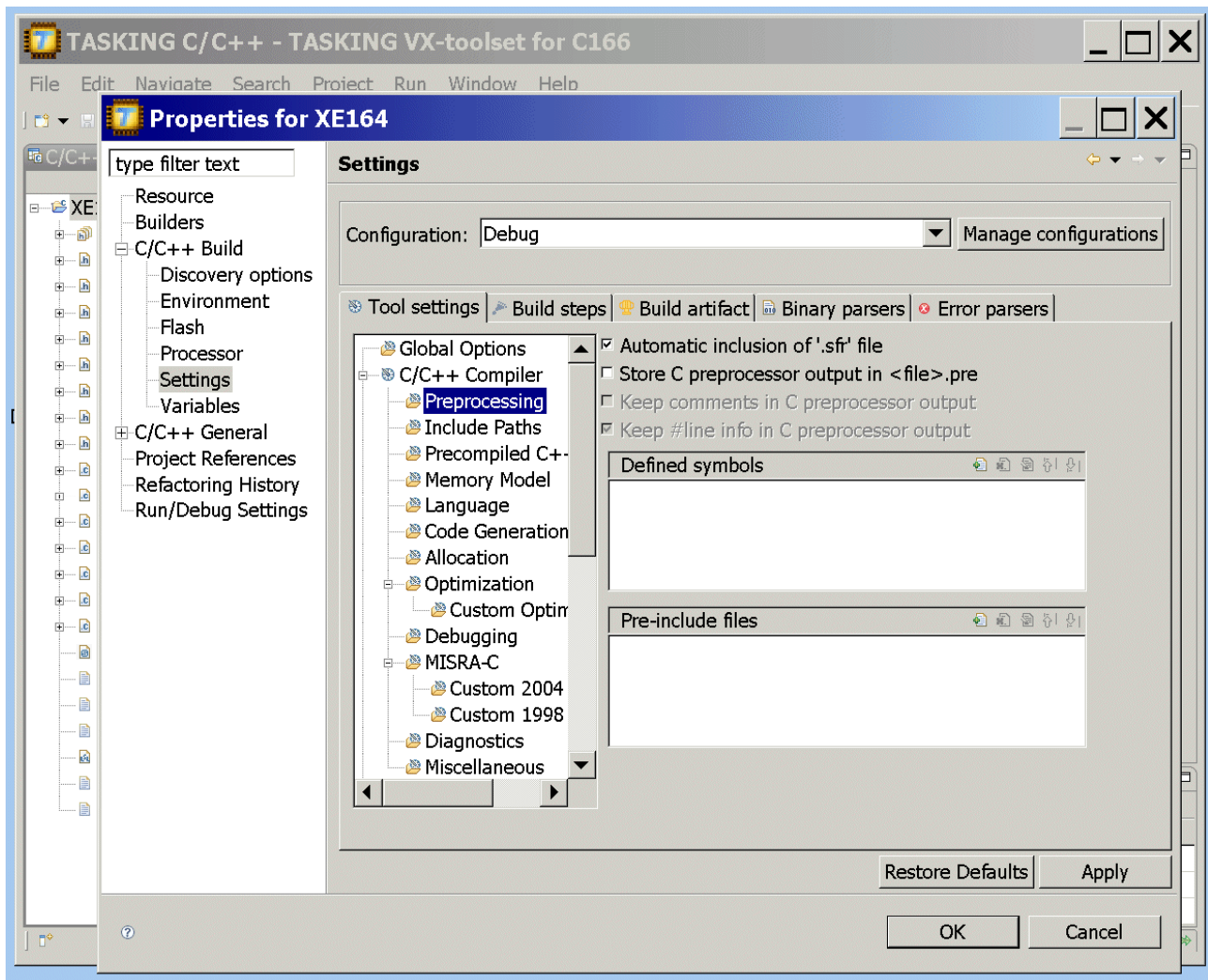


**Note:**

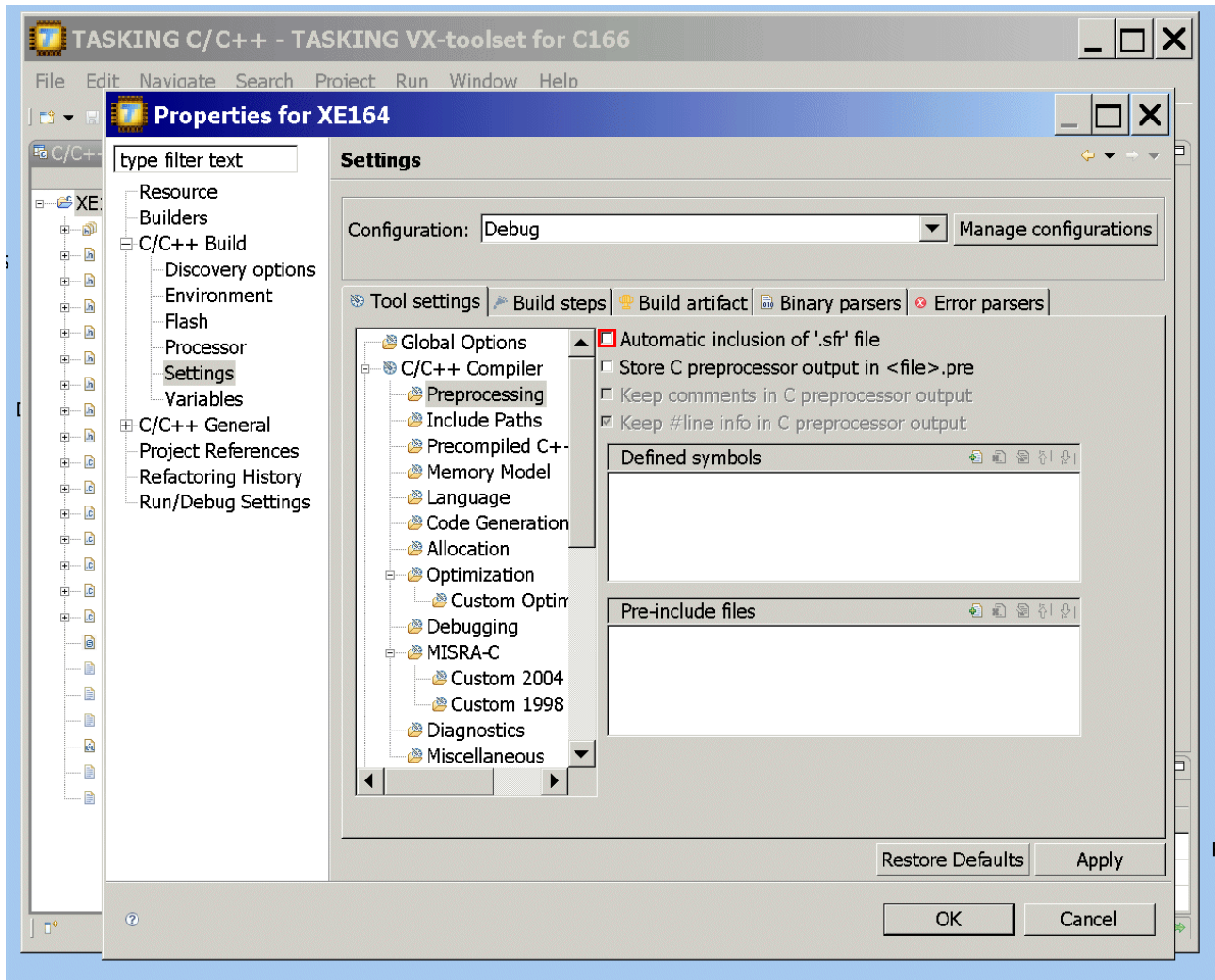
If you see nothing inside the **Tool settings** window, just move the window border (see **blue line**, see **blue arrows**) to the left or to the right.



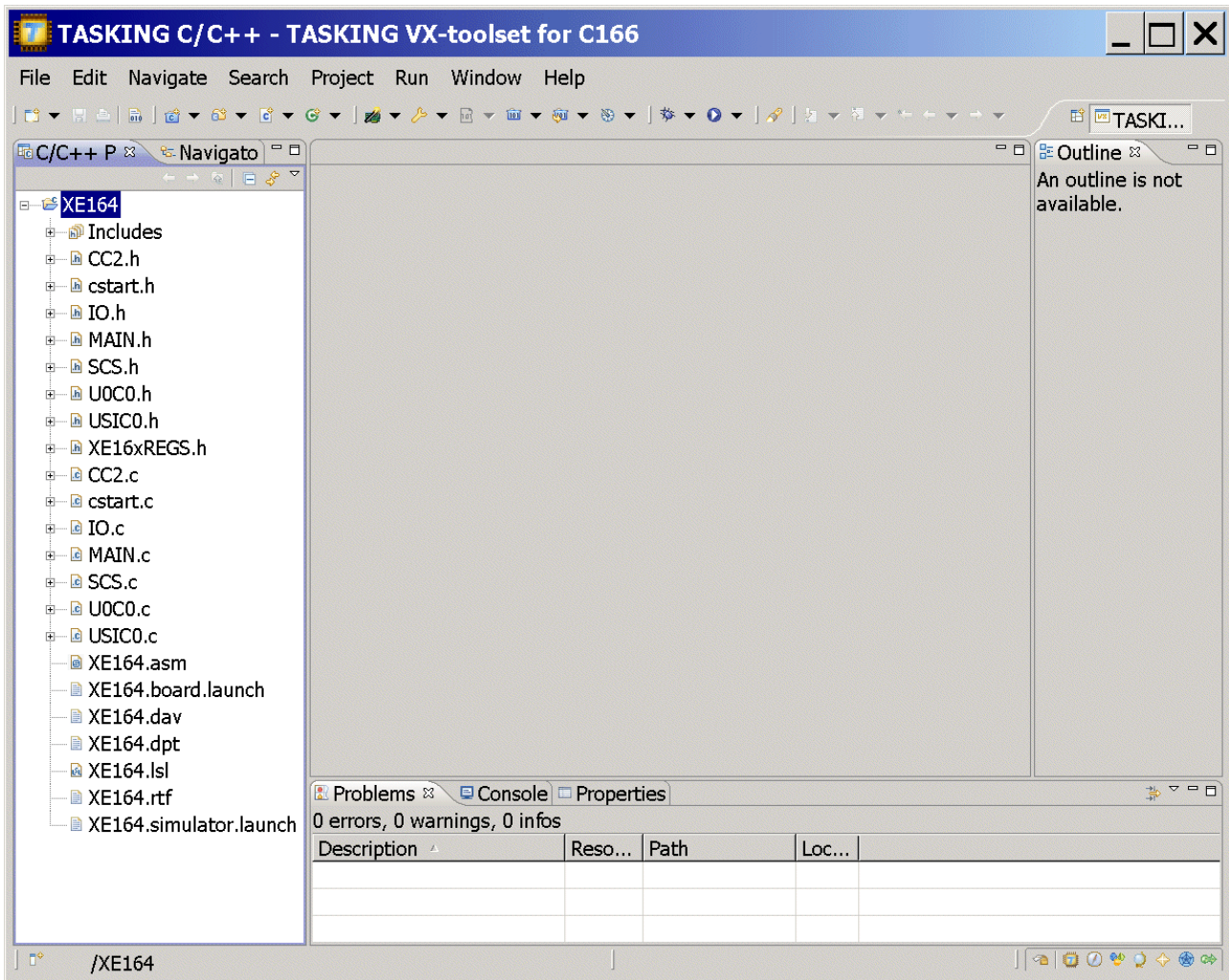
Click Preprocessing



Click to unselect ☐ Automatic inclusion of .sfr file



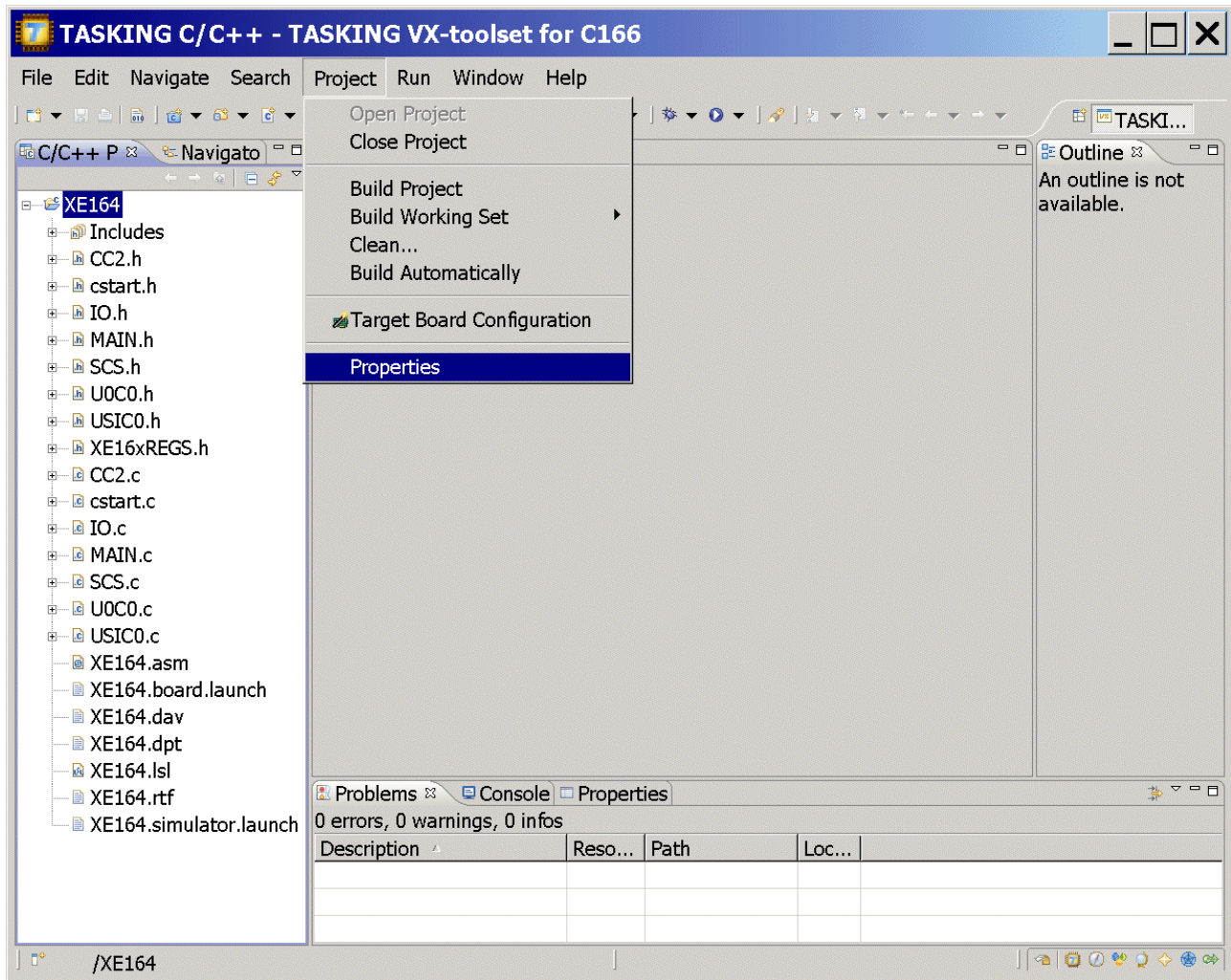
OK

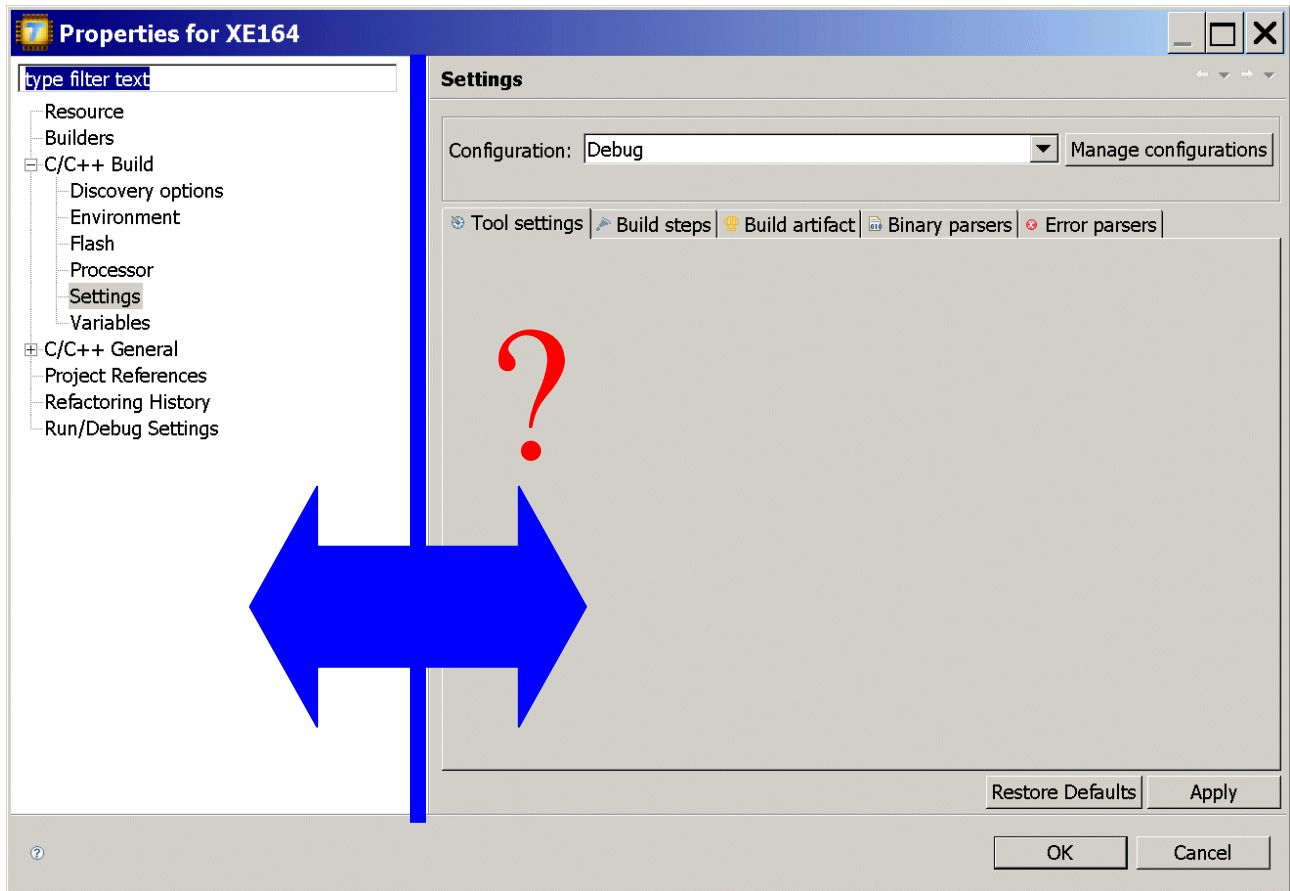




Configure the Project (delete the simulator use):

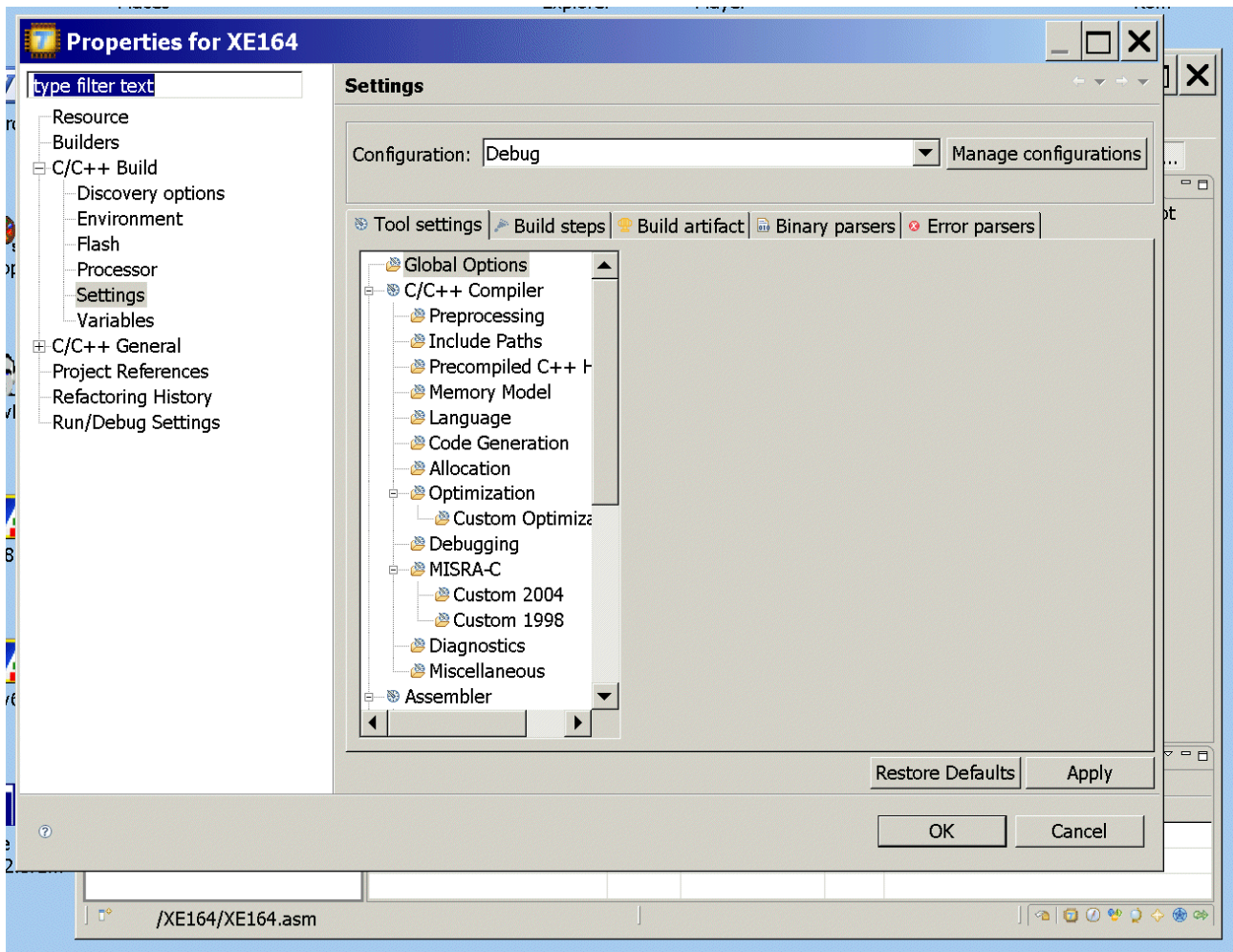
### Project - Properties





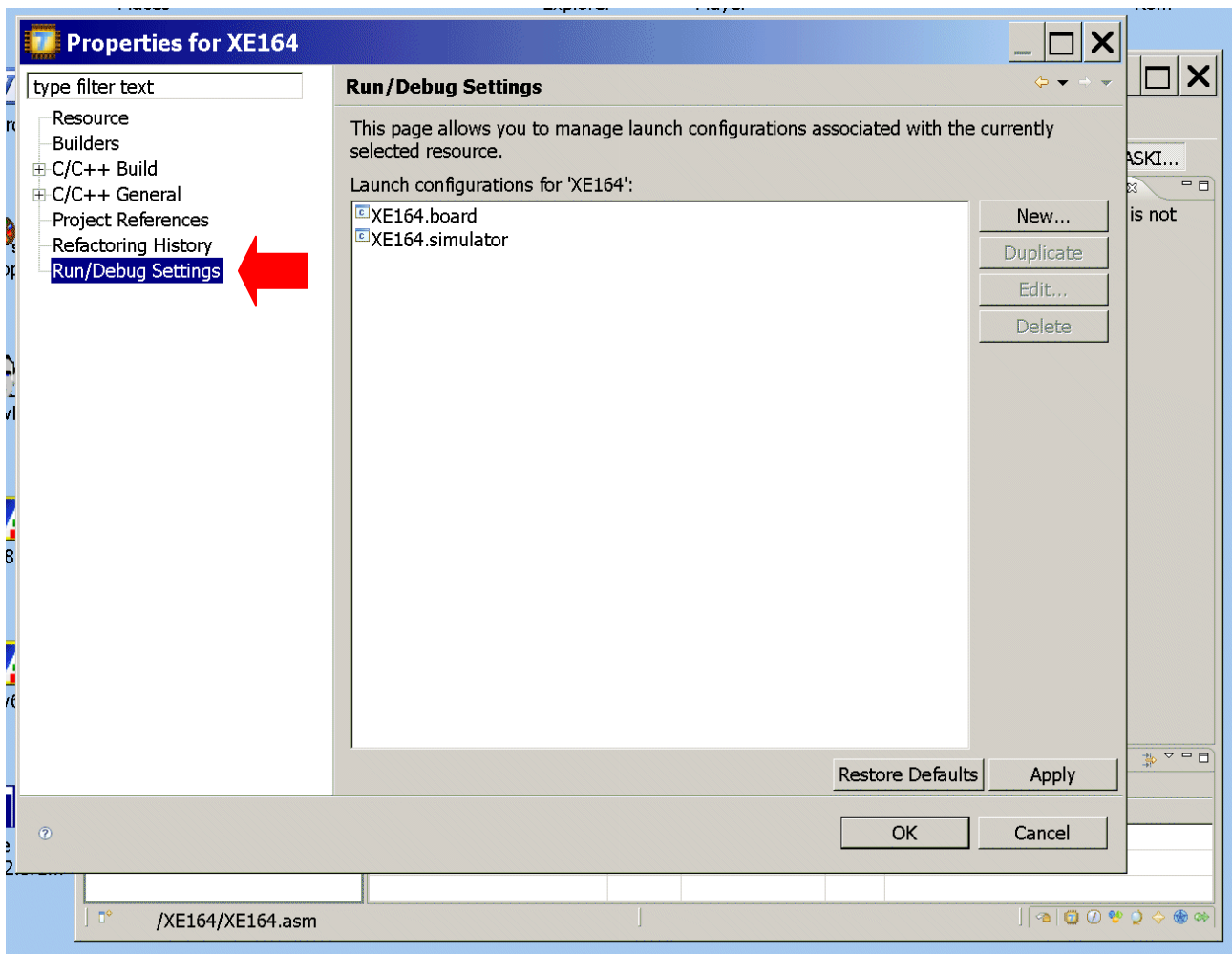
**Remember:**  
Blue line, blue arrows (if you see nothing inside the Tool settings window).



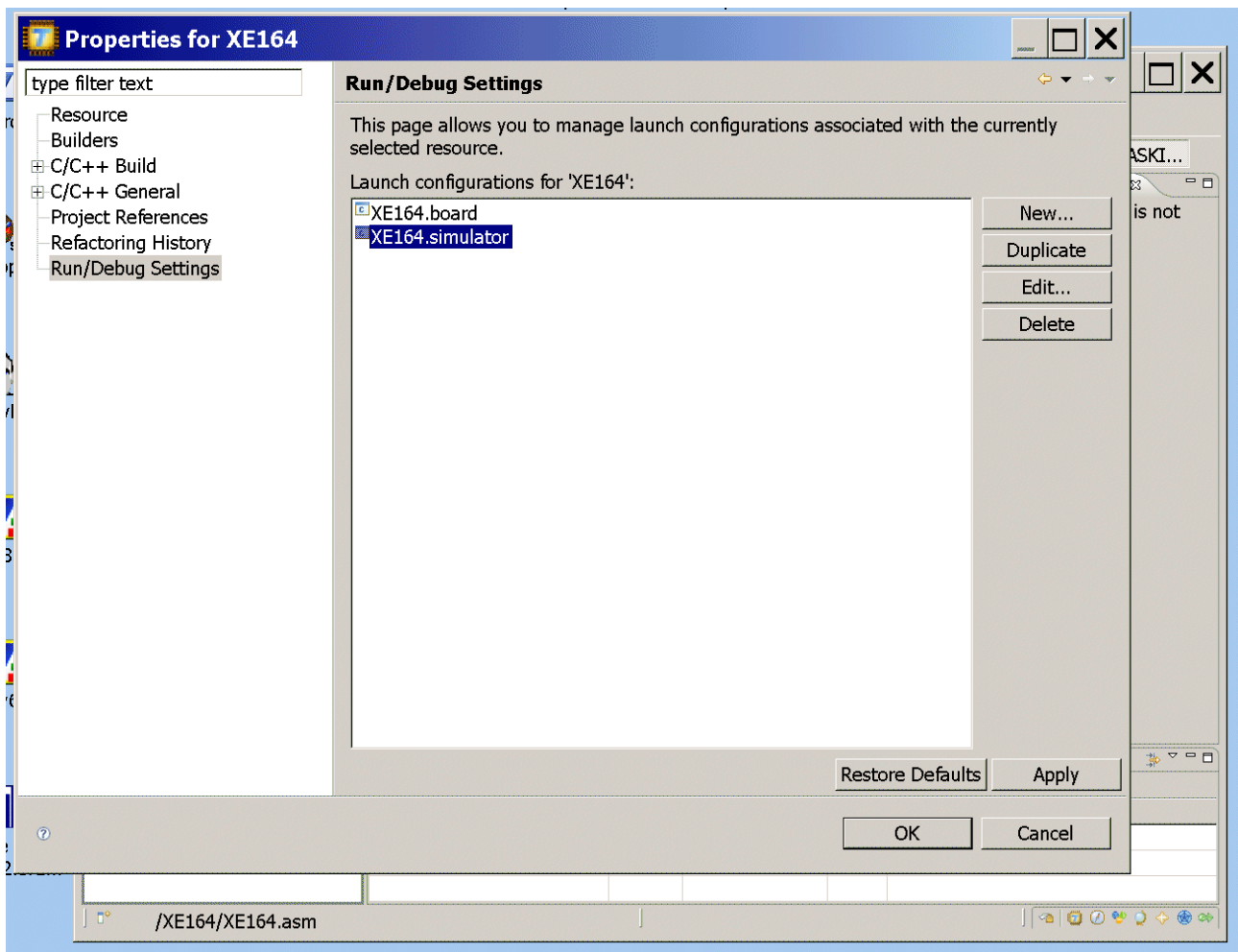




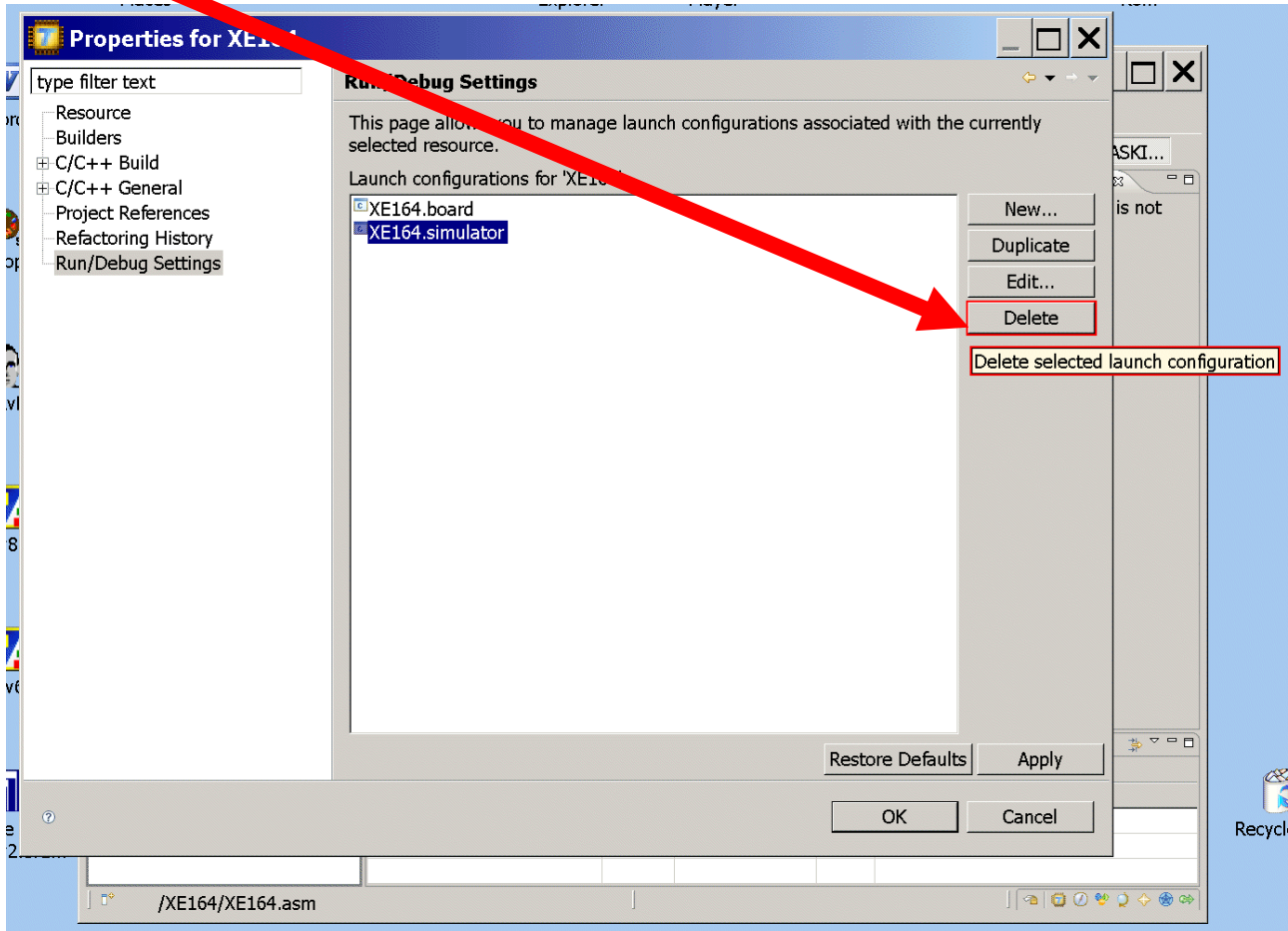
Click Run/Debug Settings



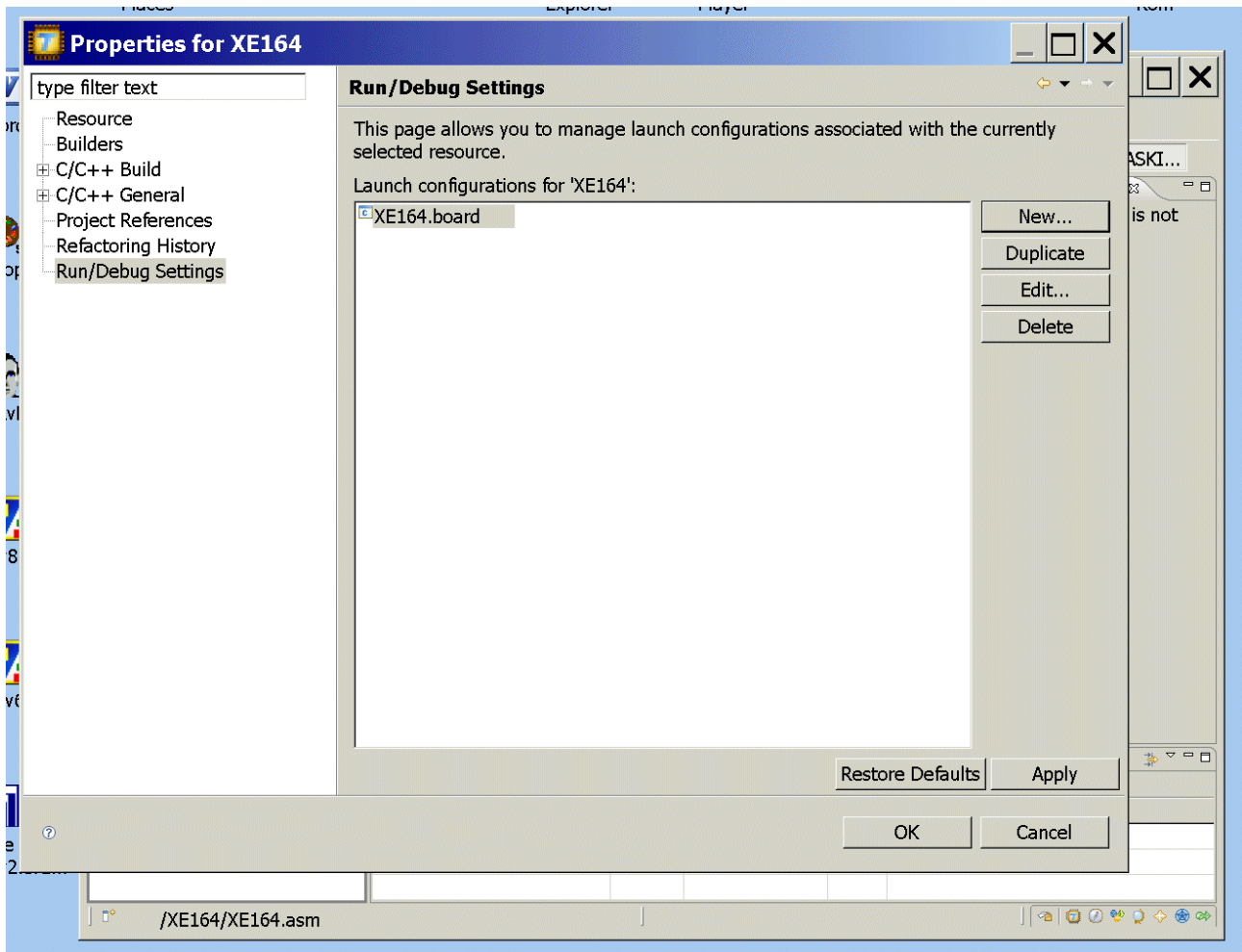
Click XE164.simulator



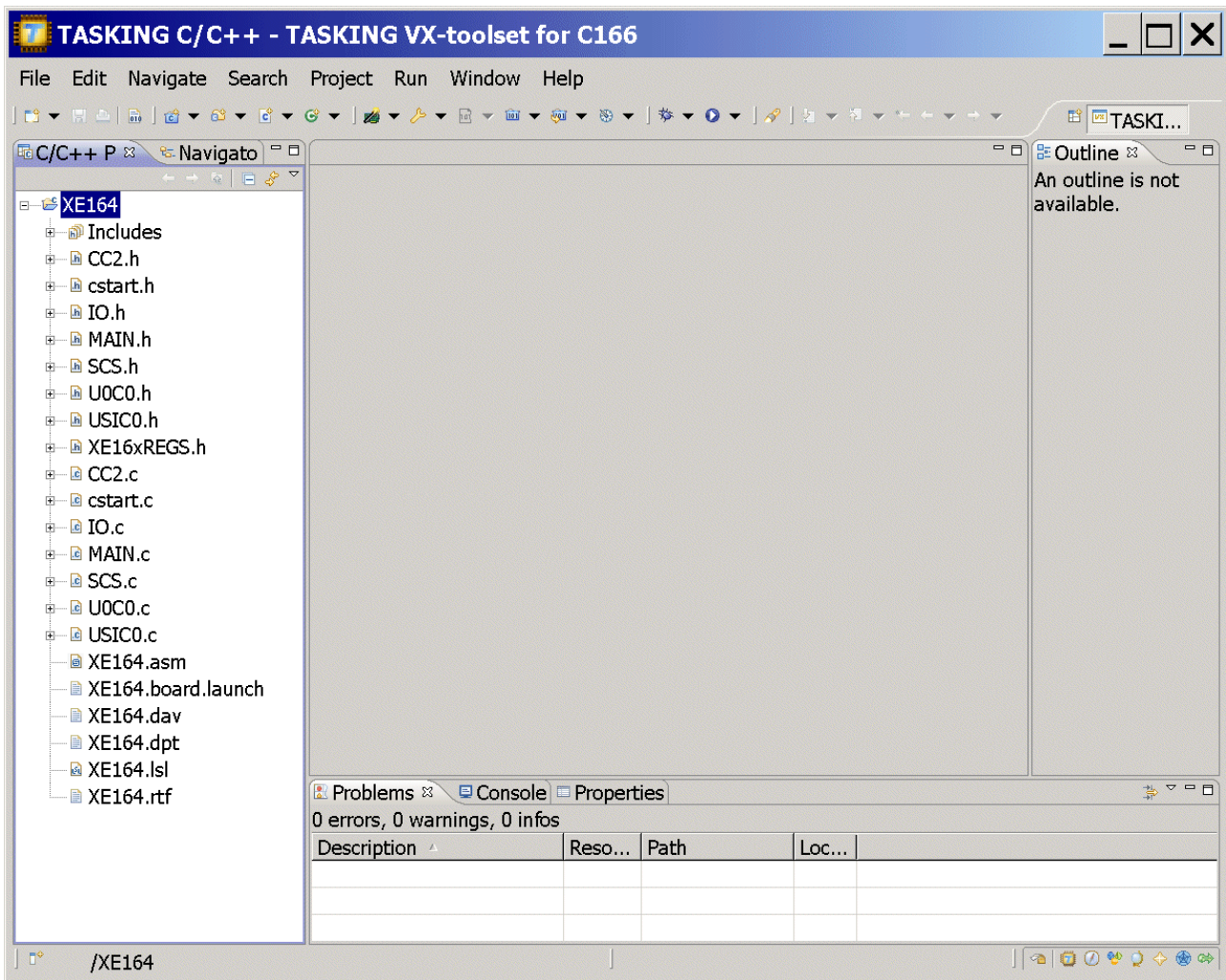
Click Delete





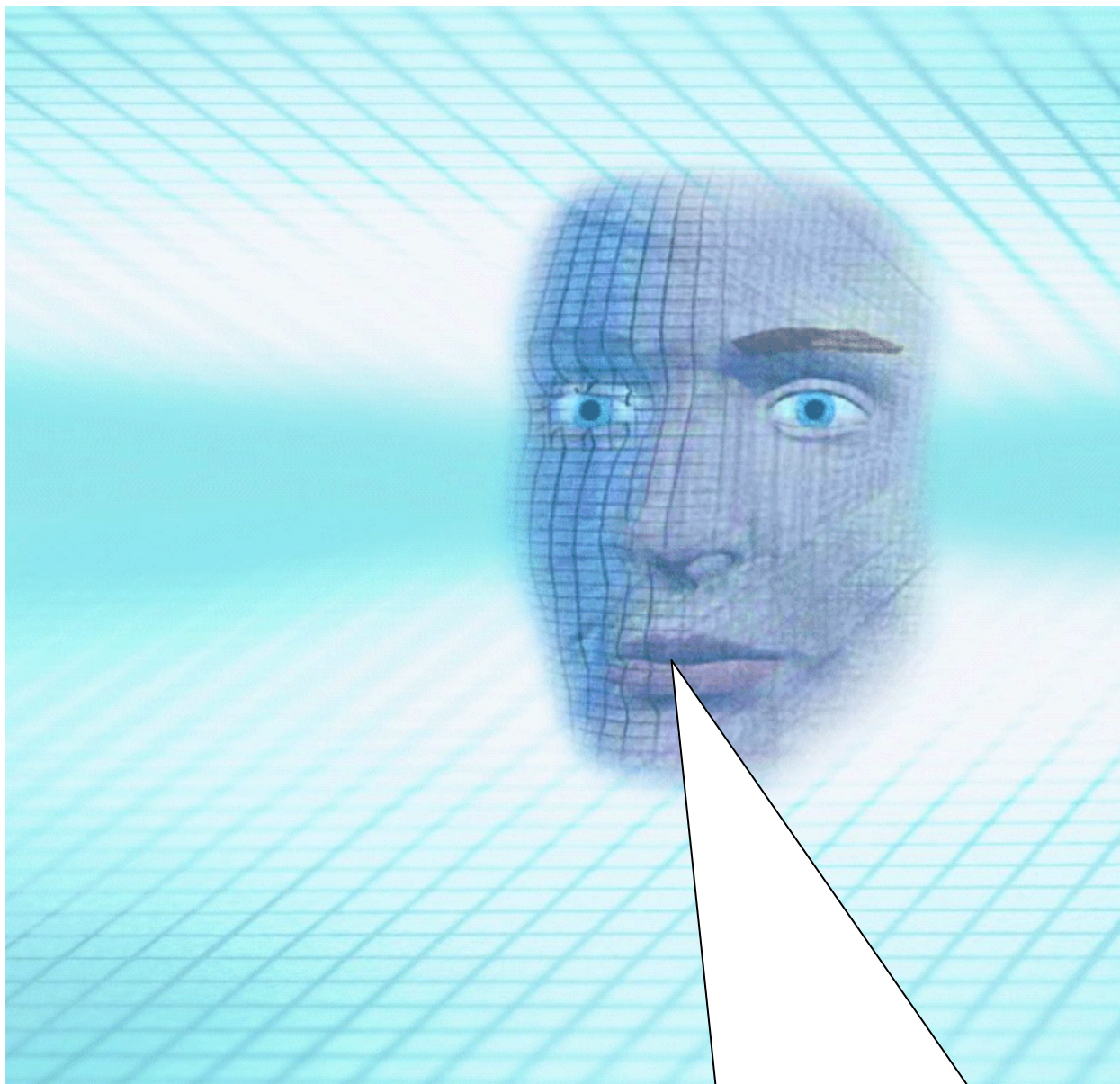


OK





Insert your application specific program:



**Note:**

DAvE doesn't change code which is inserted between '`// USER CODE BEGIN`' and '`// USER CODE END`'. Therefore, whenever adding code to DAvE's generated code, write it between '`// USER CODE BEGIN`' and '`// USER CODE END`'.

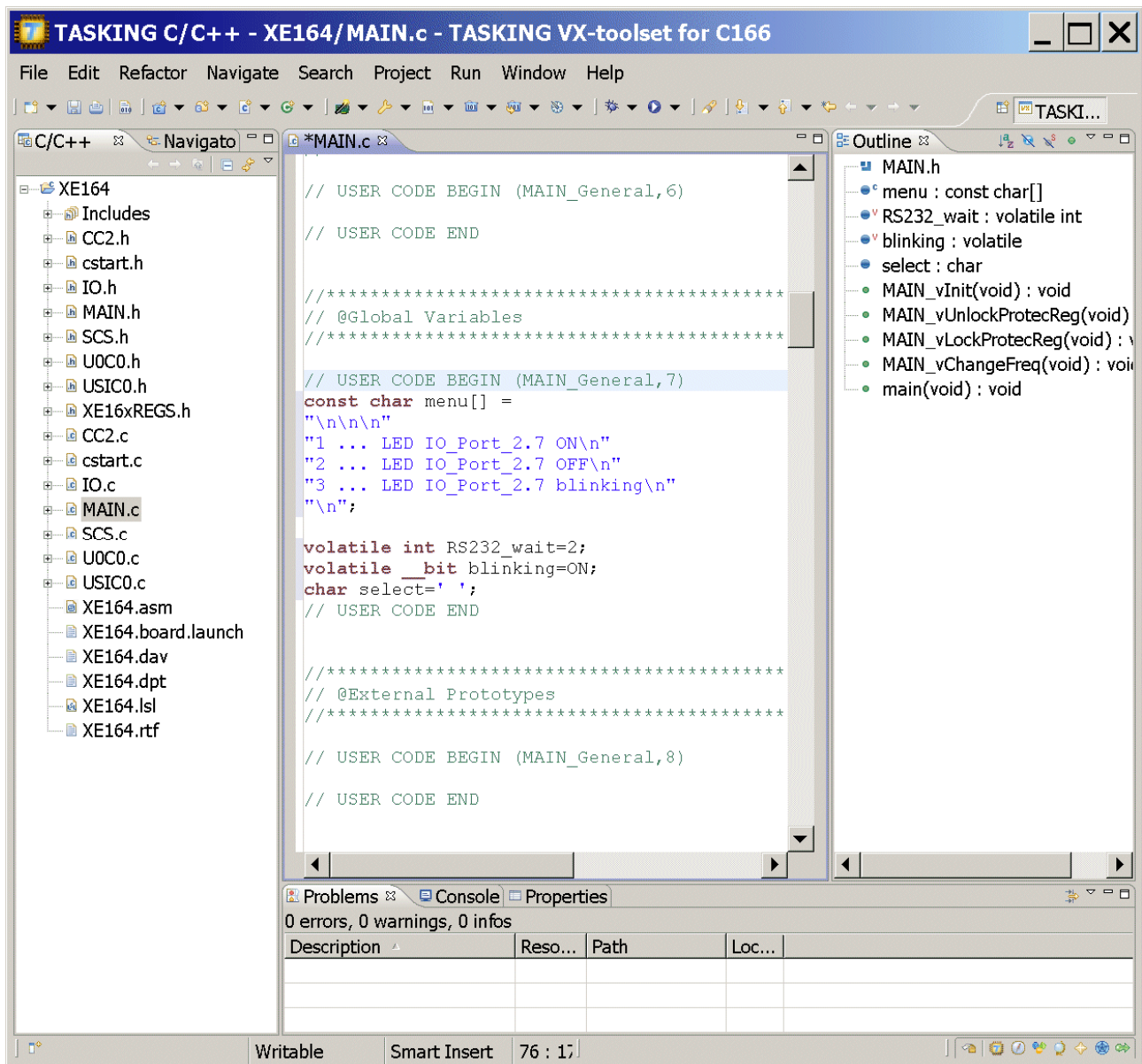
If you wish to change DAvE's generated code or add code outside these 'USER CODE' sections you will have to insert/modify your changes each time after letting DAvE regenerate code!



Double click **MAIN.C** and **insert** Global Variables:

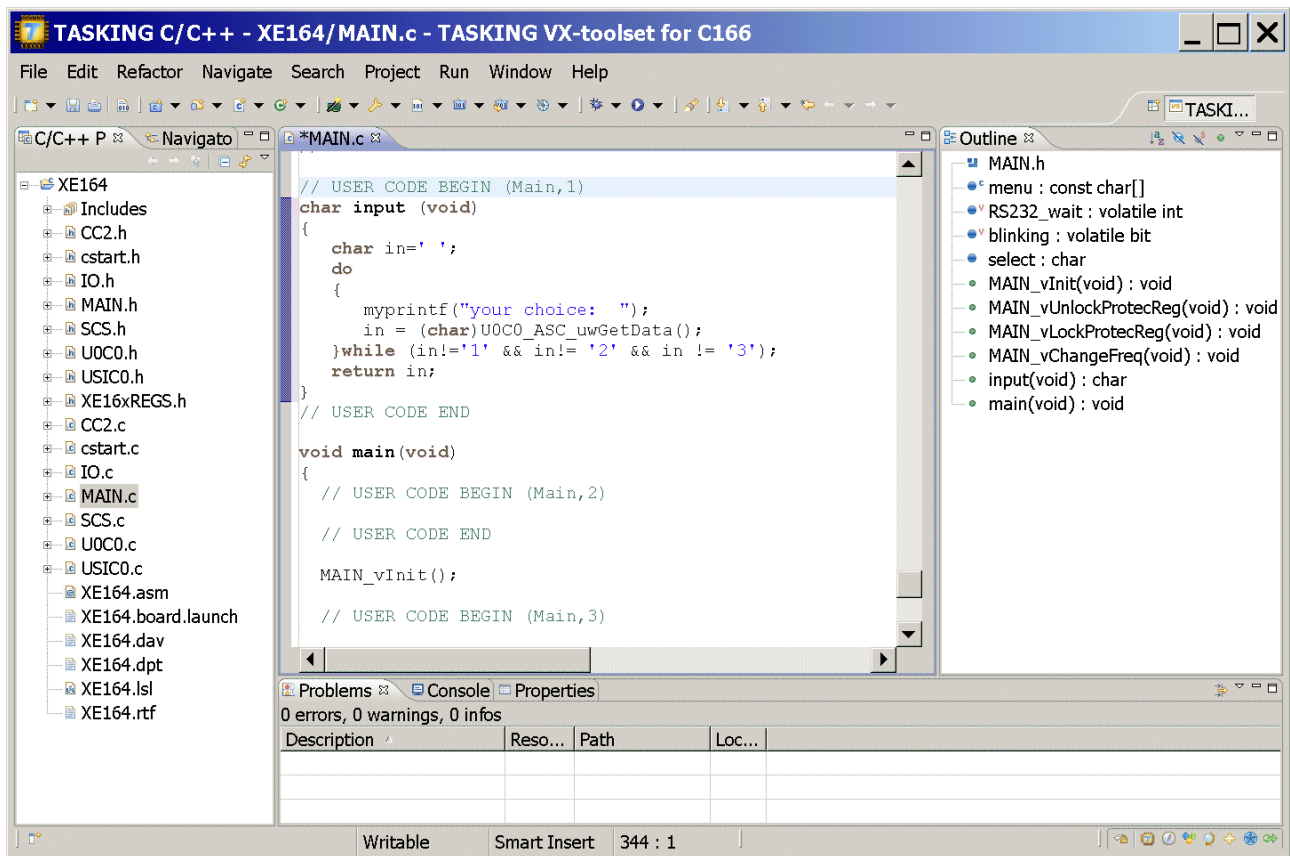
```
const char menu[] =
"\n\n\n"
"1 ... LED IO_Port_2.7 ON\n"
"2 ... LED IO_Port_2.7 OFF\n"
"3 ... LED IO_Port_2.7 blinking\n"
"\n";

volatile int RS232_wait=2;
volatile __bit blinking=ON;
char select=' ';
```



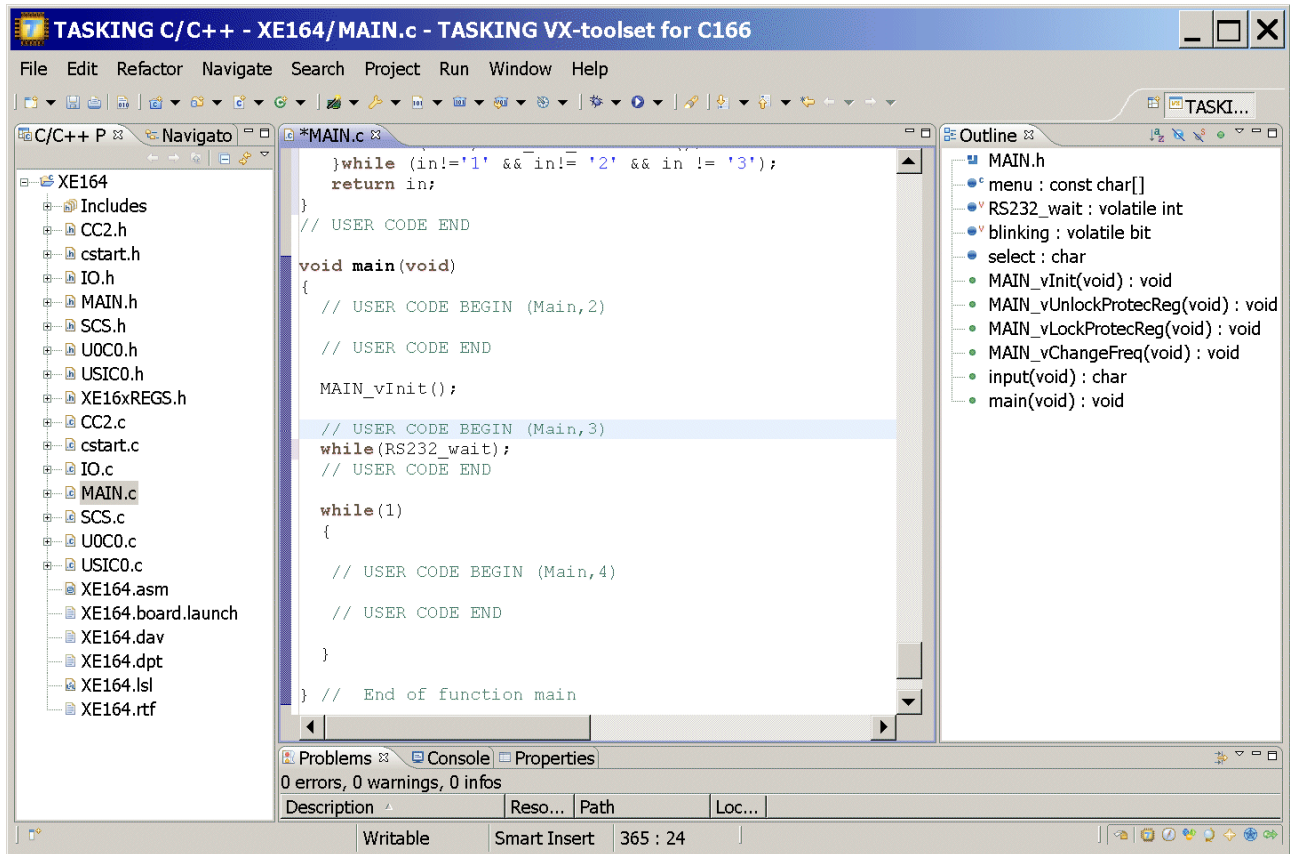
Double click **MAIN.C** and insert the function **input()**:

```
char input (void)
{
    char in=' ';
    do
    {
        myprintf("your choice: ");
        in = (char)U0C0_ASC_uwGetData();
    }while (in!='1' && in!= '2' && in != '3');
    return in;
}
```



Double click **MAIN.C** and insert the following code in the **main** function:

```
while(RS232_wait);
```

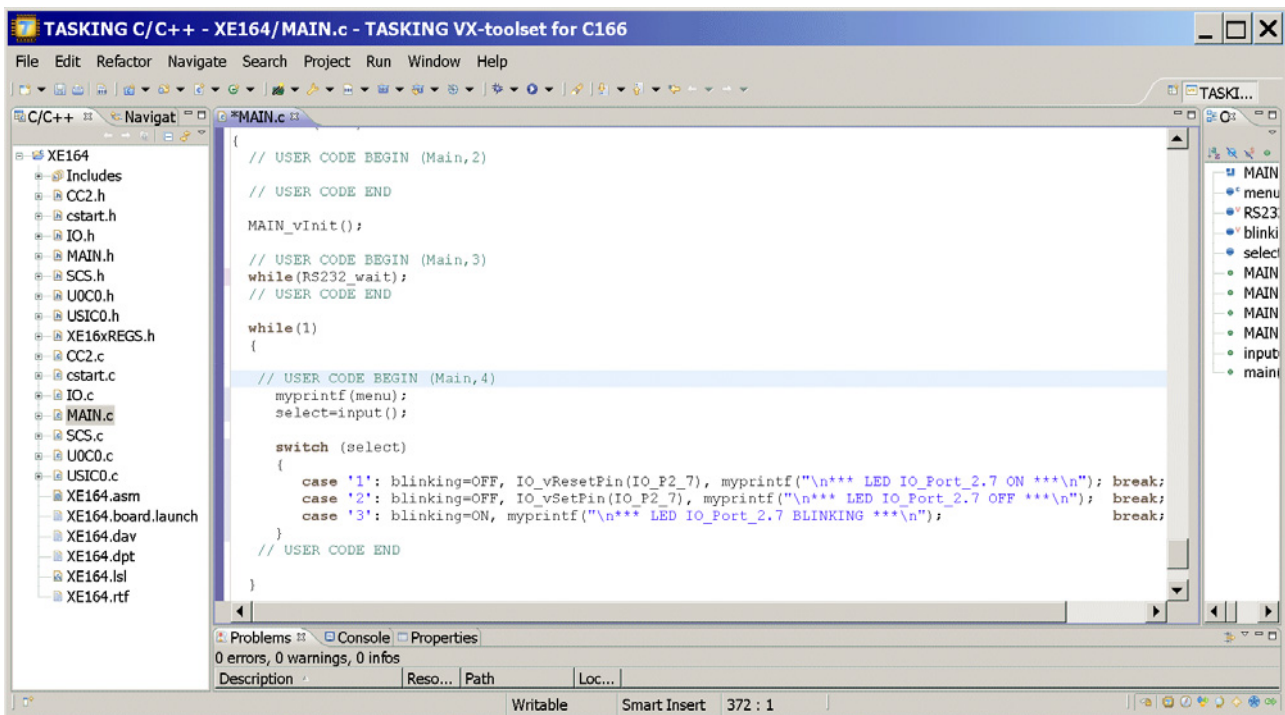




Double click **MAIN.C** and insert the following code in the **main** function into the **while(1)** loop:

```
myprintf(menu);
select=input();

switch (select)
{
    case '1': blinking=OFF, IO_vResetPin(IO_P2_7), myprintf("\n*** LED IO_Port_2.7 ON ***\n"); break;
    case '2': blinking=OFF, IO_vSetPin(IO_P2_7), myprintf("\n*** LED IO_Port_2.7 OFF ***\n"); break;
    case '3': blinking=ON, myprintf("\n*** LED IO_Port_2.7 BLINKING ***\n"); break;
}
```





Additional information: Port Output Modification Register (Source: User's Manual):

### Pn\_OMRL (n=6-11)

Port n Output Modification Register LowXSFR (E9C0<sub>H</sub>+4\*n) Reset Value: XXXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC 7	PC 6	PC 5	PC 4	PC 3	PC 2	PC 1	PC 0	PS 7	PS 6	PS 5	PS 4	PS 3	PS 2	PS 1	PS 0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Field	Bits	Type	Description
<b>PSx</b> (x = 0-7)	x	W	<b>Port Set Bit x</b> Setting this bit sets or toggles the corresponding bit in the port output register Pn_OUT (see <a href="#">Table 7-4</a> ). On a read access, this bit returns 0.
<b>PCx</b> (x = 0-7)	x + 8	W	<b>Port Clear Bit x</b> Setting this bit clears or toggles the corresponding bit in the port output register Pn_OUT. (see <a href="#">Table 7-4</a> ). On a read access, this bit returns 0.

### Function of the PCx and PSx bit fields

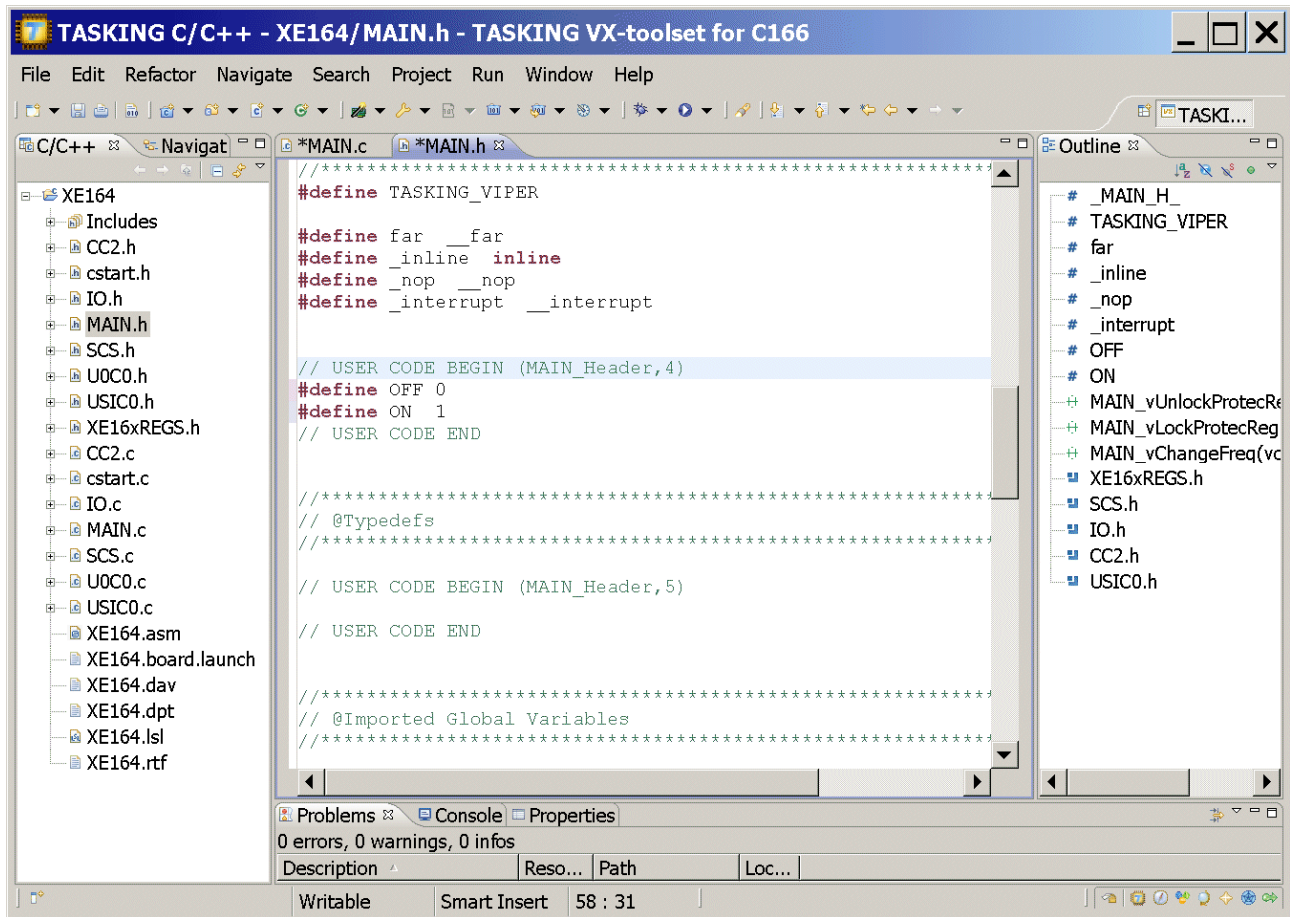
Table 7-4 Function of the Bits PCx and PSx

PCx	PSx	Function
0 or no write access	0 or no write access	Bit Pn_OUT.Px is not changed.
0 or no write access	1	Bit Pn_OUT.Px is set.
1	0 or no write access	Bit Pn_OUT.Px is cleared.
1	1	Bit Pn_OUT.Px is toggled.

*Note: If a bit position is not written (one out of two bytes not targeted by a byte write), the corresponding value is considered as 0. Toggling a bit requires one 16-bit write.*

Double click **Main.h** and insert the following Defines:

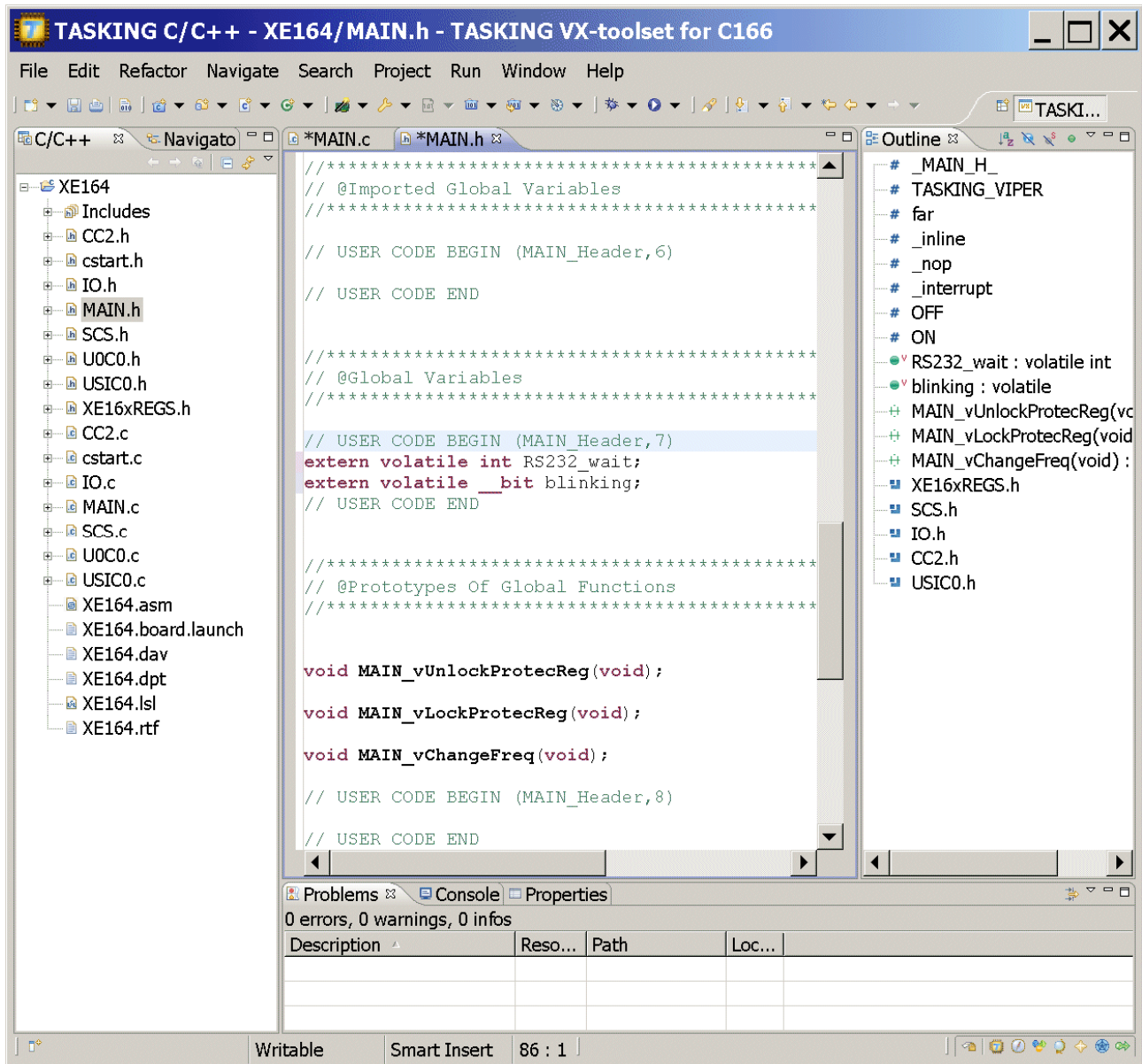
```
#define OFF 0
#define ON 1
```





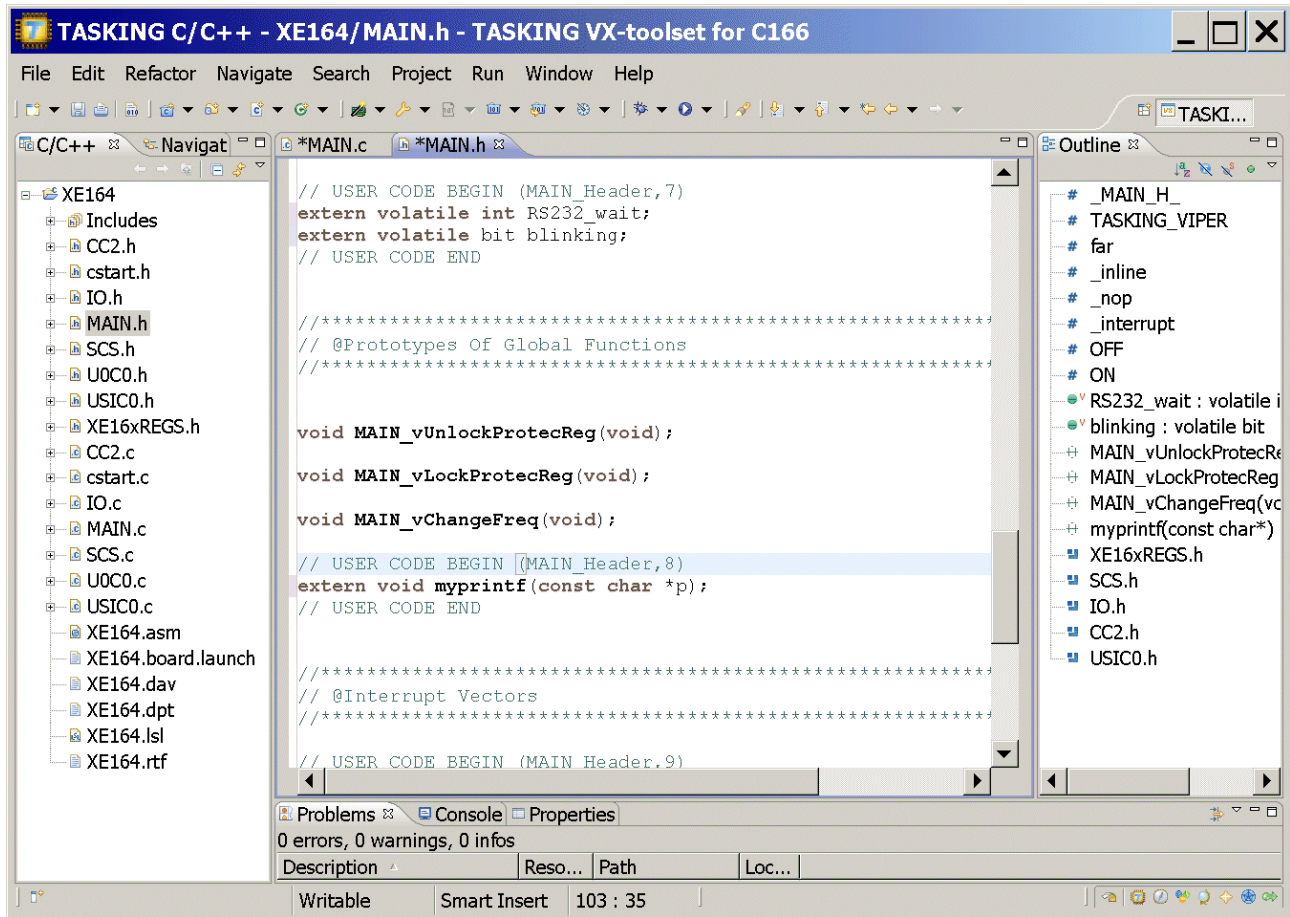
Double click **Main.h** and insert extern declarations "Global Variables":

```
extern volatile int RS232_wait;
extern volatile __bit blinking;
```



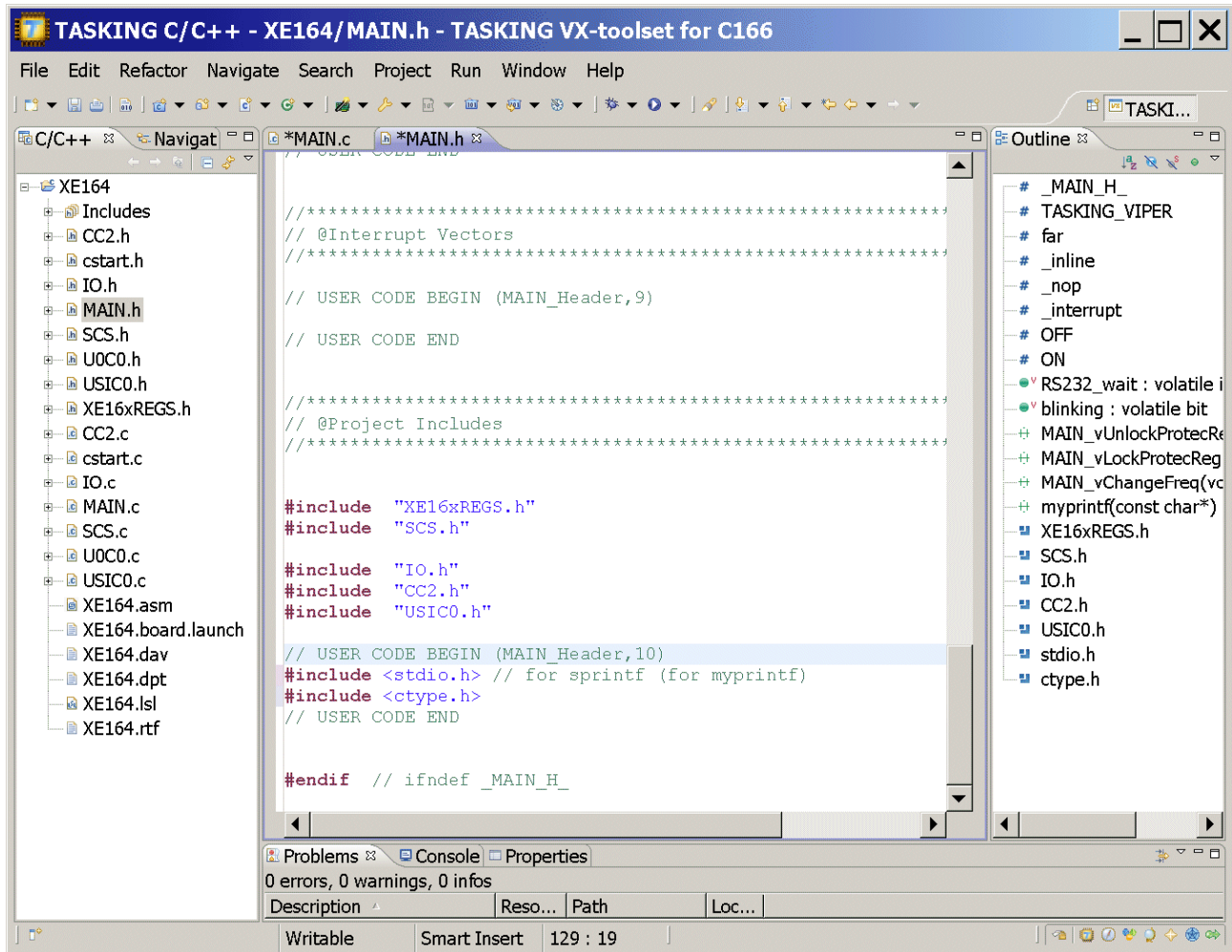
Double click **Main.h** and insert extern declarations "Global Functions":

```
extern void myprintf(const char *p);
```



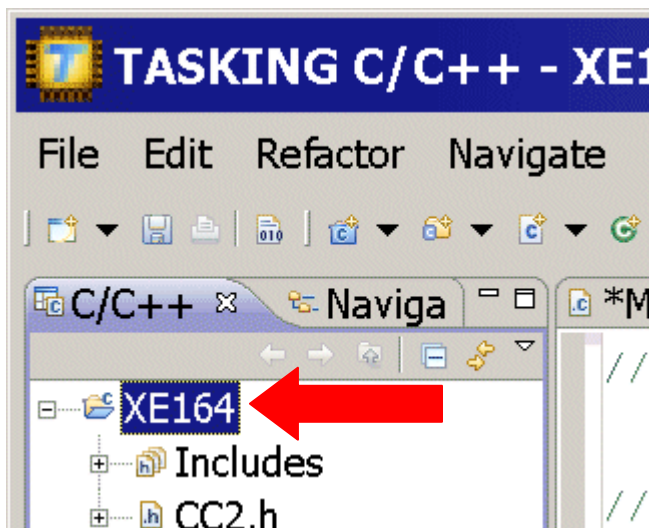
Double click **Main.h** and **insert** include files:

```
#include <stdio.h> // for sprintf (for myprintf)
#include <ctype.h>
```

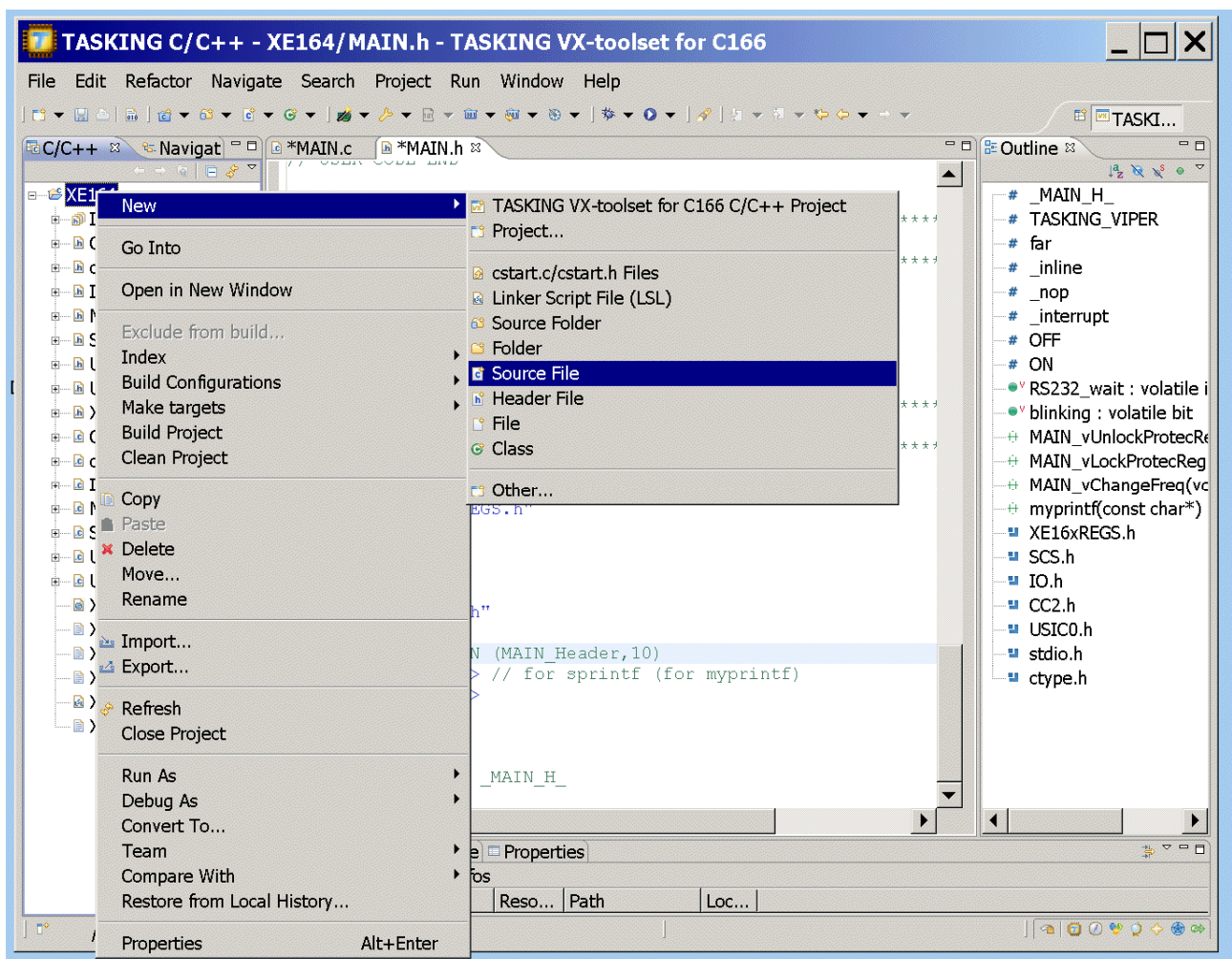




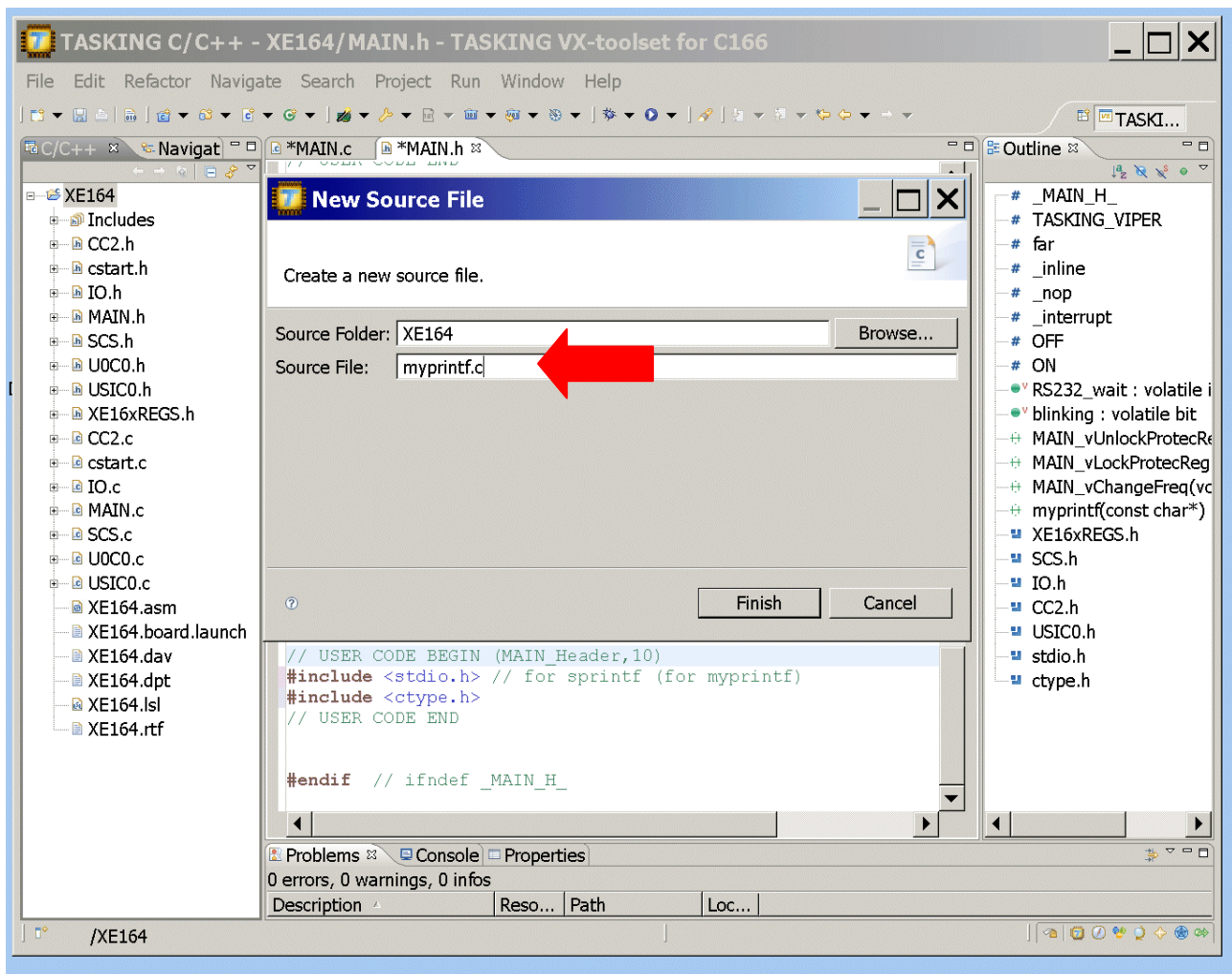
Click right mouse button @ XE164.asm



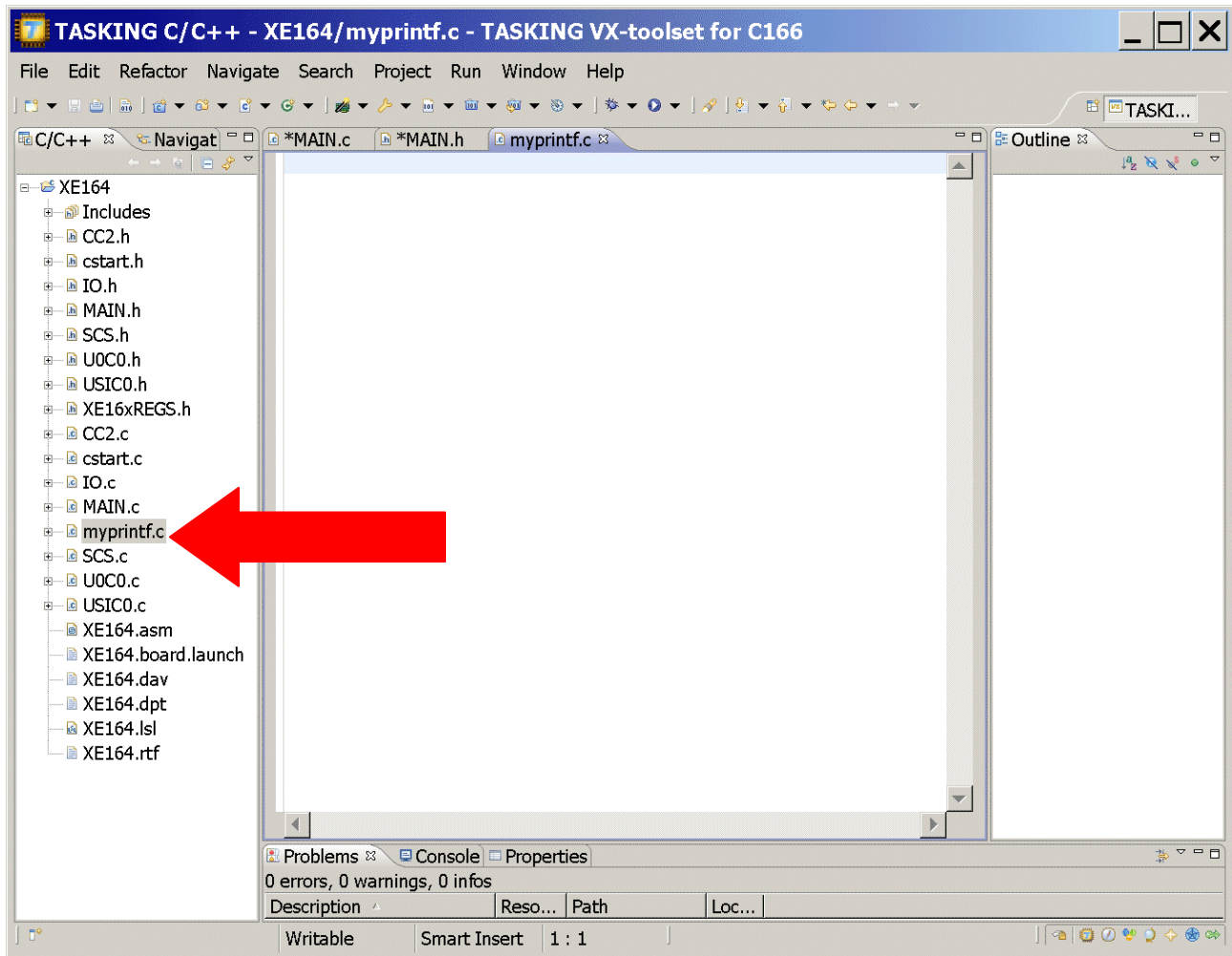
New – Source File



Create a new source file: Source File: insert myprintf.c



Click Finish





Insert:

```
#include "main.h"

void myprintf(const char *p)
{
    while(*p)
    {
        U0C0_ASC_vSendData(*p++);
    }
}

/*

// Example 1 (use of myprintf):
// =====

void main(void)
{
    myprintf("Hello World!\n");
}

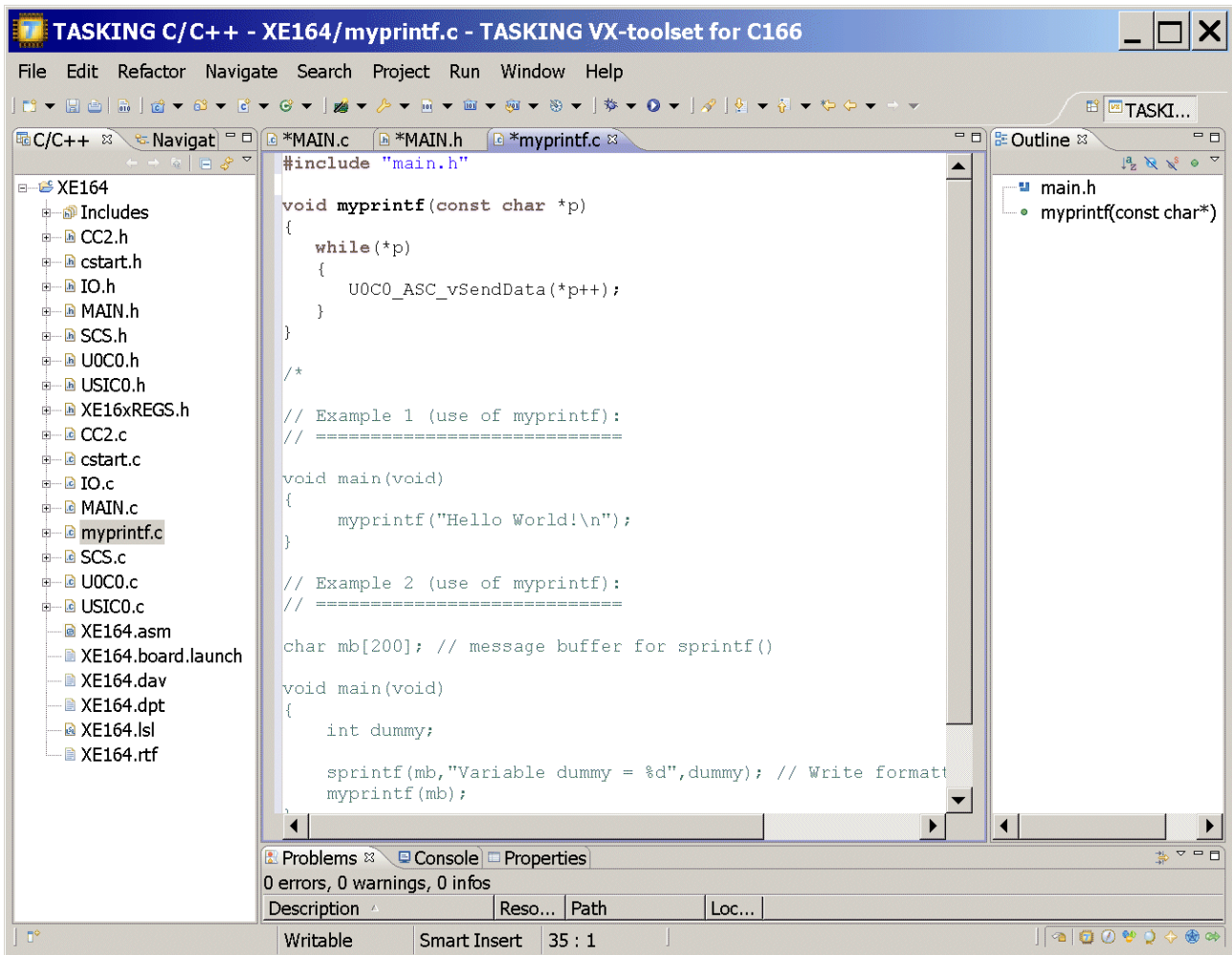
// Example 2 (use of myprintf):
// =====

char mb[200]; // message buffer for sprintf()

void main(void)
{
    int dummy;

    sprintf(mb,"Variable dummy = %d",dummy); // Write formatted data to string mb
    myprintf(mb);
}

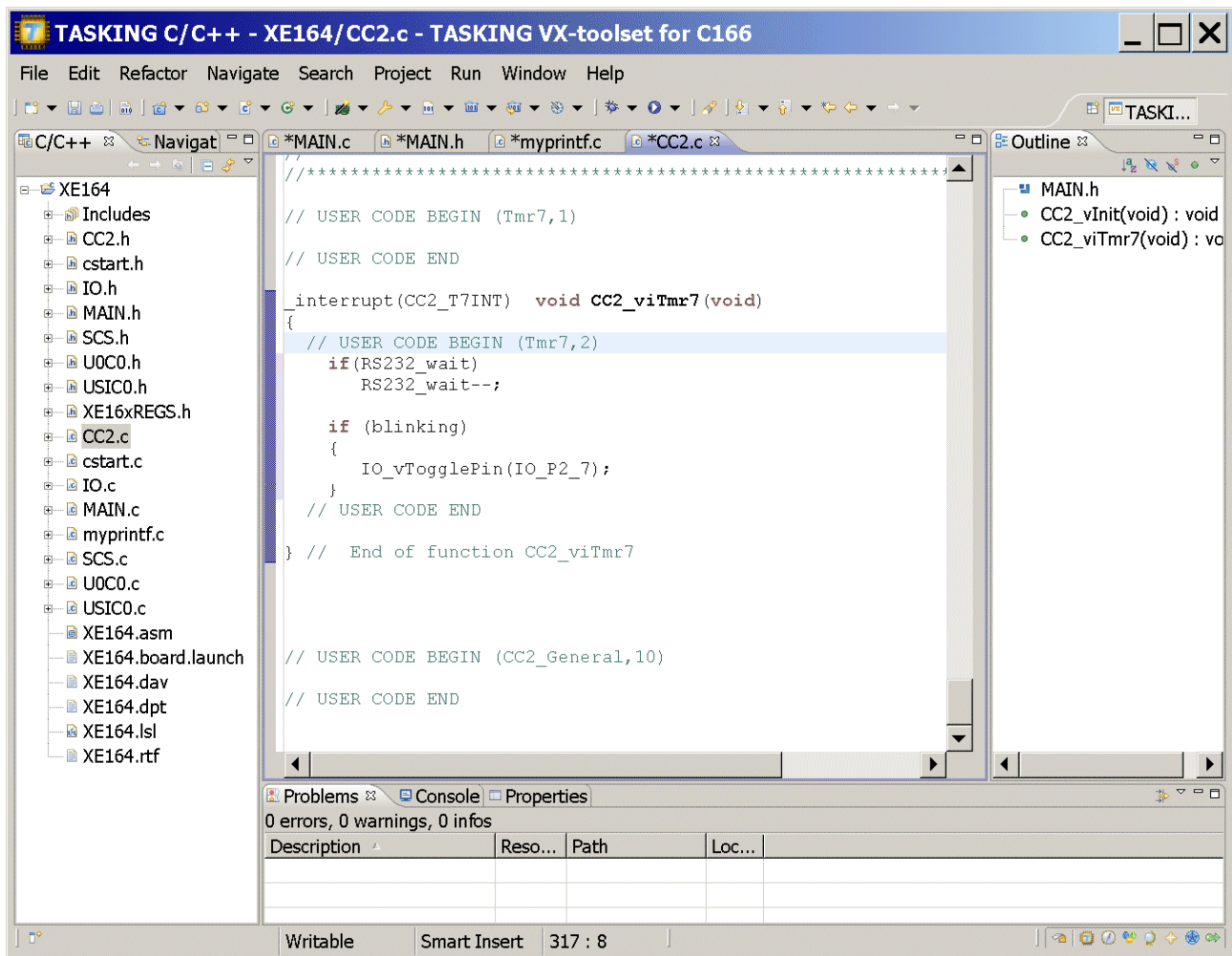
*/
```



Double click **CC2.C insert** Code (CAPCOM 2 Timer 7 Interrupt Service Routine):

```
if(RS232_wait)
    RS232_wait--;

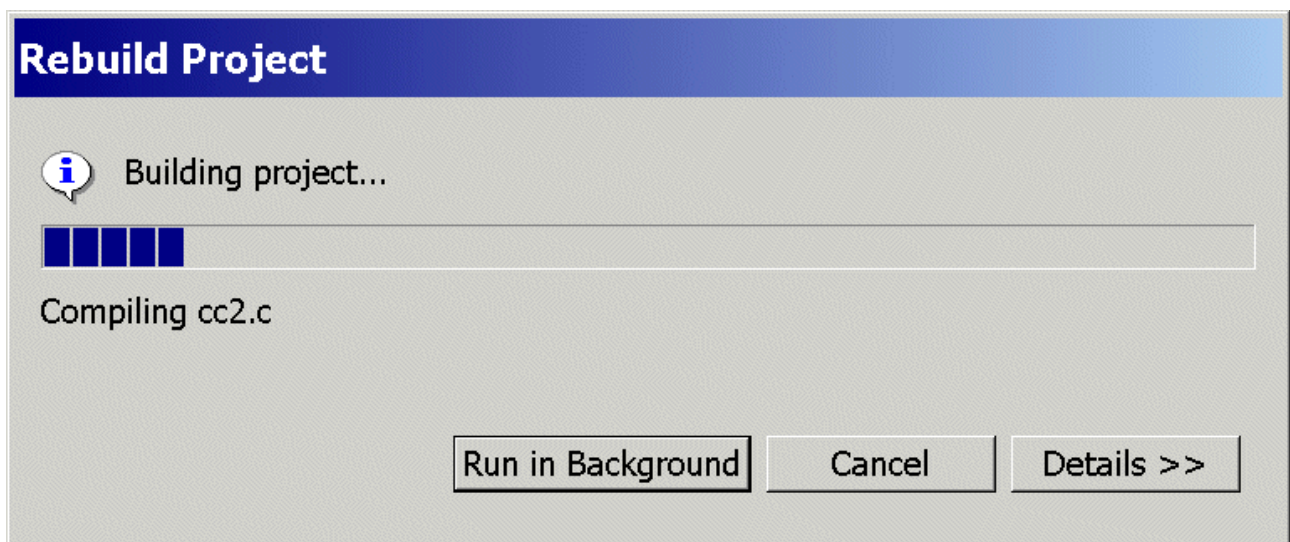
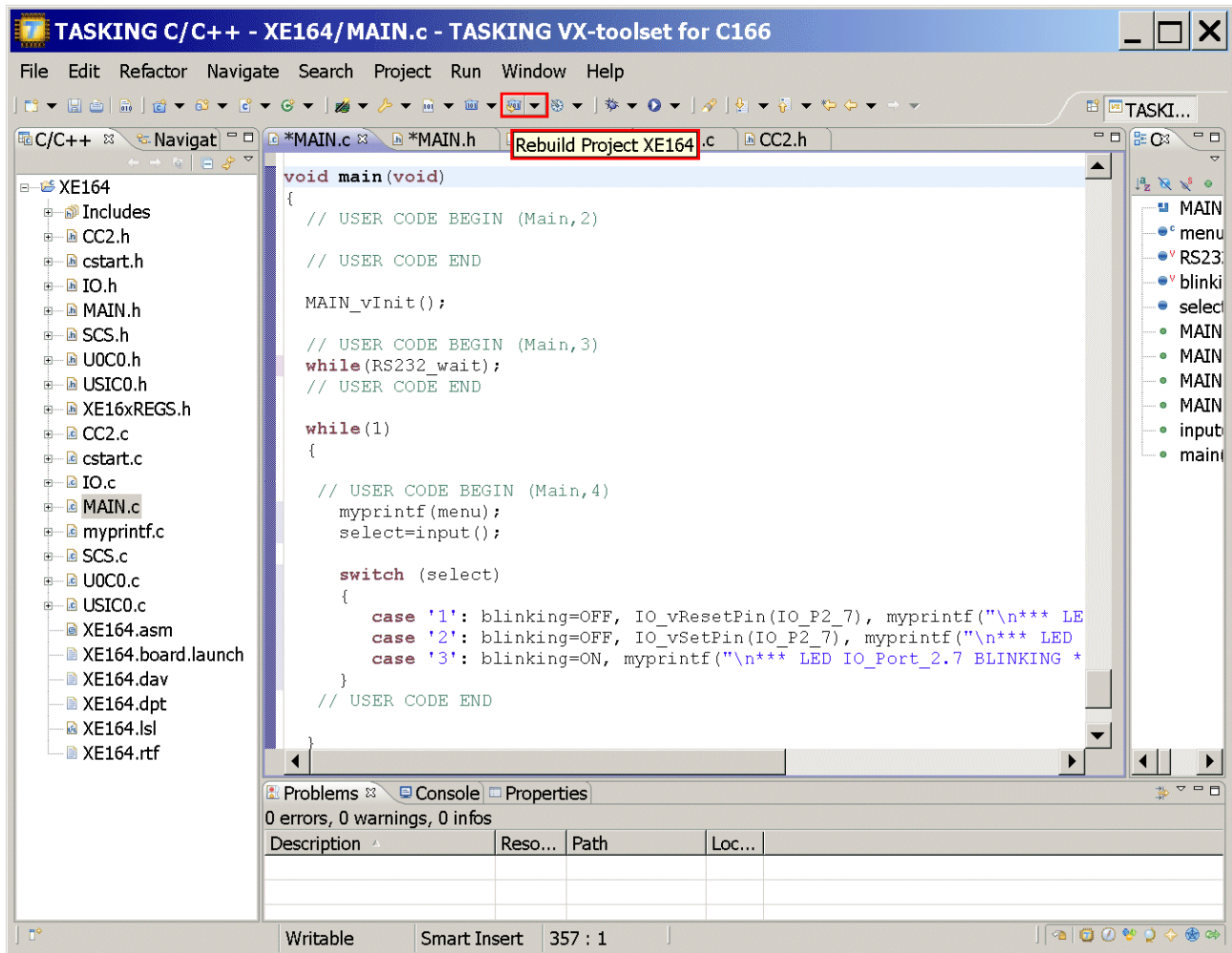
if (blinking)
{
    IO_vTogglePin(IO_P2_7);
}
```

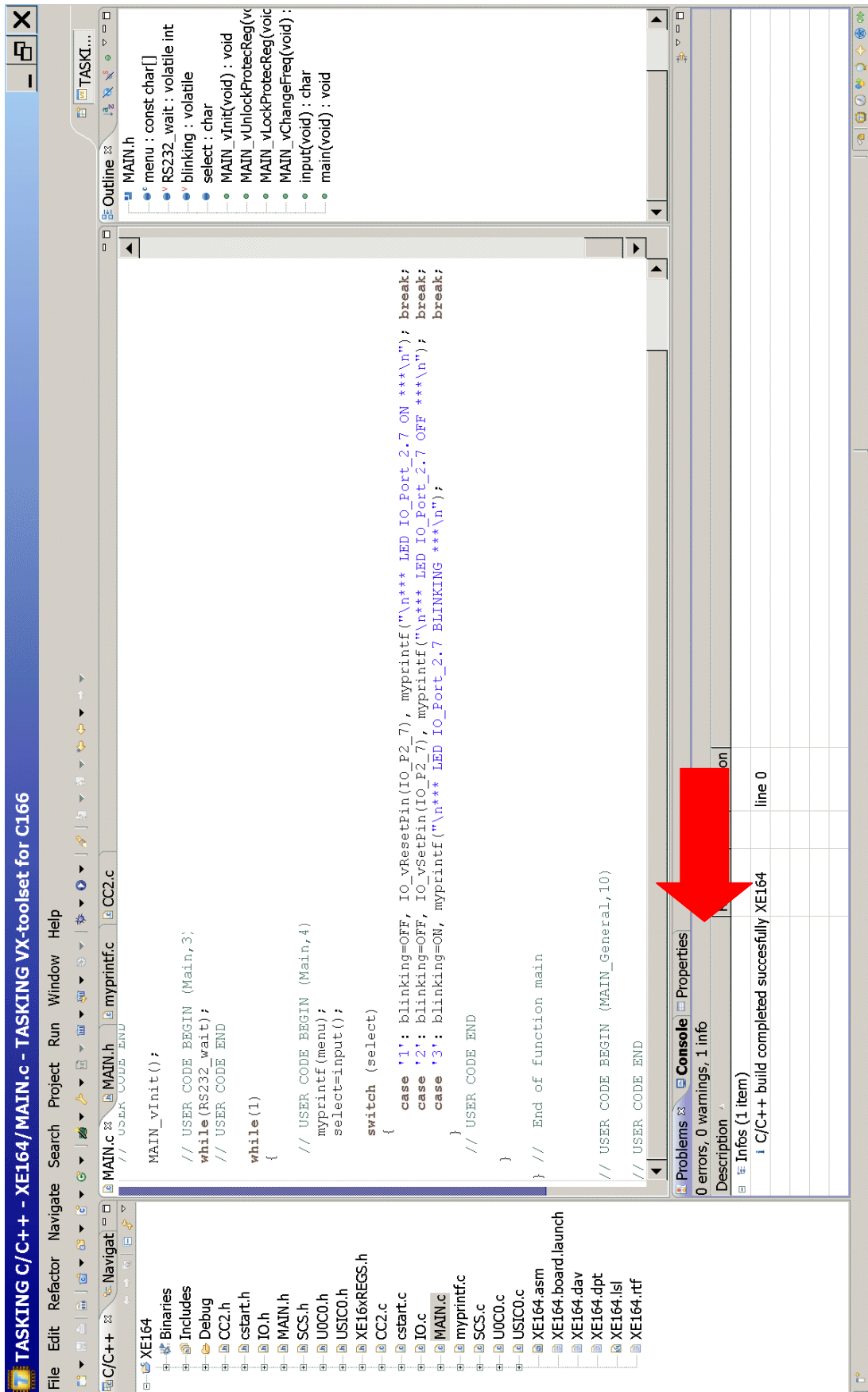




Generate your application program:

Click  Rebuild Project XE164





**TASKING C/C++ - XE164/MAIN.c - TASKING VX-toolset for C166**

File Edit Refactor Navigate Search Project Run Window Help

Navigation: MAIN.c, MAIN.h, myprintf.c, CC2.c

```

MAIN_vInit();
// USER CODE BEGIN (Main,3)
while (RS232_wait);
// USER CODE END

while (1)
{
    // USER CODE BEGIN (Main,4)
    myprintf(menu);
    select=input();

    switch (select)
    {
        case '1': blinking=OFF, IO_vResetPin(IO_P2_7), myprintf("\n*** LED IO_Port_2.7 ON ***\n"); break;
        case '2': blinking=OFF, IO_vSetPin(IO_P2_7), myprintf("\n*** LED IO_Port_2.7 OFF ***\n"); break;
        case '3': blinking=ON, myprintf("\n*** LED IO_Port_2.7 BLINKING ***\n"); break;
    }
    // USER CODE END
}

// End of function main

// USER CODE BEGIN (MAIN_General,10)
// USER CODE END
    
```

**Outline:**

- MAIN.h
- menu : const char[]
- RS232\_wait : volatile int
- blinking : volatile
- select : char
- MAIN\_vInit(void) : void
- MAIN\_vUnlockProtectReg(void) : void
- MAIN\_vLockProtectReg(void) : void
- MAIN\_vChangeFreq(void) : void
- input(void) : char
- main(void) : void

**Problems Console Properties**

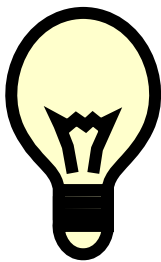
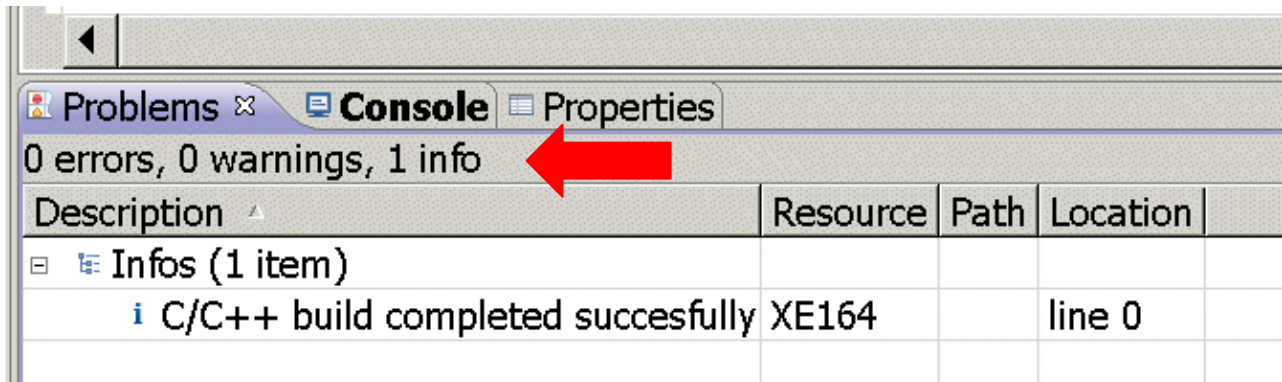
0 errors, 0 warnings, 1 info

Description

Infos (1 item)

Description	Line
C/C++ build completed successfully XE164	line 0

**Navigation:** XE164, Binaries, Includes, Debug, CC2.h, cstart.h, IO.h, MAIN.h, SCS.h, UOC0.h, USIC0.h, XE16xREGS.h, CC2.c, cstart.c, IO.c, MAIN.c, myprintf.c, SCS.c, UOC0.c, USIC0.c, XE164.asm, XE164.board launch, XE164.dav, XE164.dpt, XE164.lsl, XE164.rtf



**Note:**

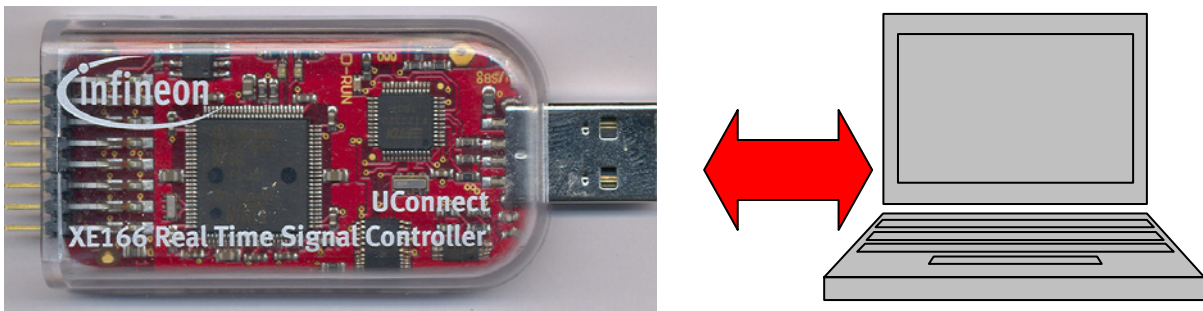
Programming is now complete.

Therefore we are going to **load** (On Chip Flash Programming) and **run** your program on the UConnect-CAN XE164 in the next chapter.



### 5.) Running your first programming example:

Make sure that the UConnect-CAN XE164 is still connected to the host computer:



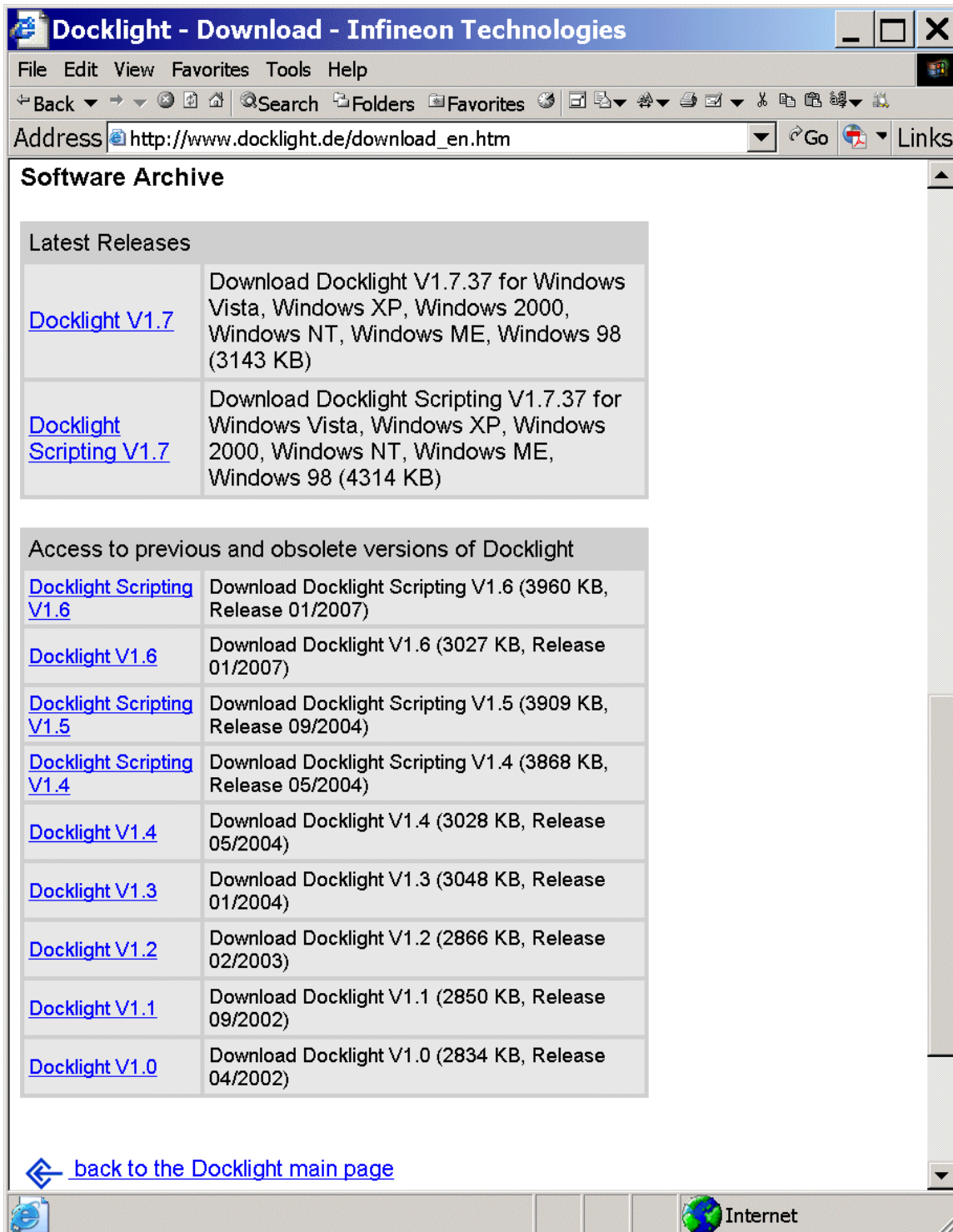
#### USB Connection:

- .) used for: UART communication (the USIC0\_CH0/UART/RS232/serial interface is available via USB as a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
- .) used for: On-Chip-Flash-Programming and Debugging (first USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
- .) the USB connection works also as the power supply.



**Note:**

Now we need a terminal program which is able to handle a virtual COM port (COM5)!  
As an example of “any terminal program” we are going to use Docklight.  
Docklight can be downloaded @ <http://www.docklight.de> :



The screenshot shows a web browser window titled "Docklight - Download - Infineon Technologies". The address bar shows the URL [http://www.docklight.de/download\\_en.htm](http://www.docklight.de/download_en.htm). The page content is titled "Software Archive" and lists the latest releases and previous versions of Docklight.

Latest Releases	
<a href="#">Docklight V1.7</a>	Download Docklight V1.7.37 for Windows Vista, Windows XP, Windows 2000, Windows NT, Windows ME, Windows 98 (3143 KB)
<a href="#">Docklight Scripting V1.7</a>	Download Docklight Scripting V1.7.37 for Windows Vista, Windows XP, Windows 2000, Windows NT, Windows ME, Windows 98 (4314 KB)

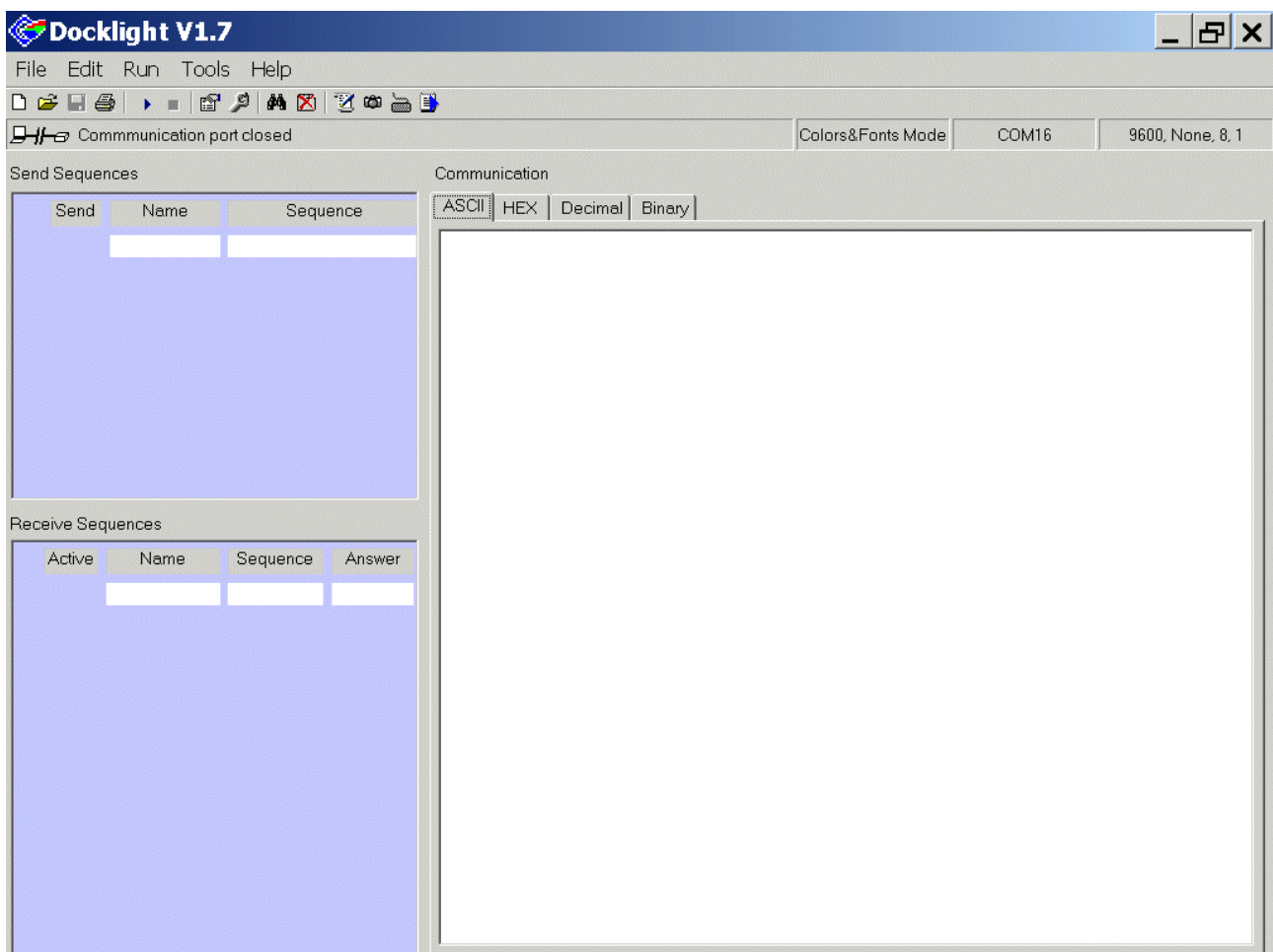
  

Access to previous and obsolete versions of Docklight	
<a href="#">Docklight Scripting V1.6</a>	Download Docklight Scripting V1.6 (3960 KB, Release 01/2007)
<a href="#">Docklight V1.6</a>	Download Docklight V1.6 (3027 KB, Release 01/2007)
<a href="#">Docklight Scripting V1.5</a>	Download Docklight Scripting V1.5 (3909 KB, Release 09/2004)
<a href="#">Docklight Scripting V1.4</a>	Download Docklight Scripting V1.4 (3868 KB, Release 05/2004)
<a href="#">Docklight V1.4</a>	Download Docklight V1.4 (3028 KB, Release 05/2004)
<a href="#">Docklight V1.3</a>	Download Docklight V1.3 (3048 KB, Release 01/2004)
<a href="#">Docklight V1.2</a>	Download Docklight V1.2 (2866 KB, Release 02/2003)
<a href="#">Docklight V1.1</a>	Download Docklight V1.1 (2850 KB, Release 09/2002)
<a href="#">Docklight V1.0</a>	Download Docklight V1.0 (2834 KB, Release 04/2002)

[back to the Docklight main page](#)

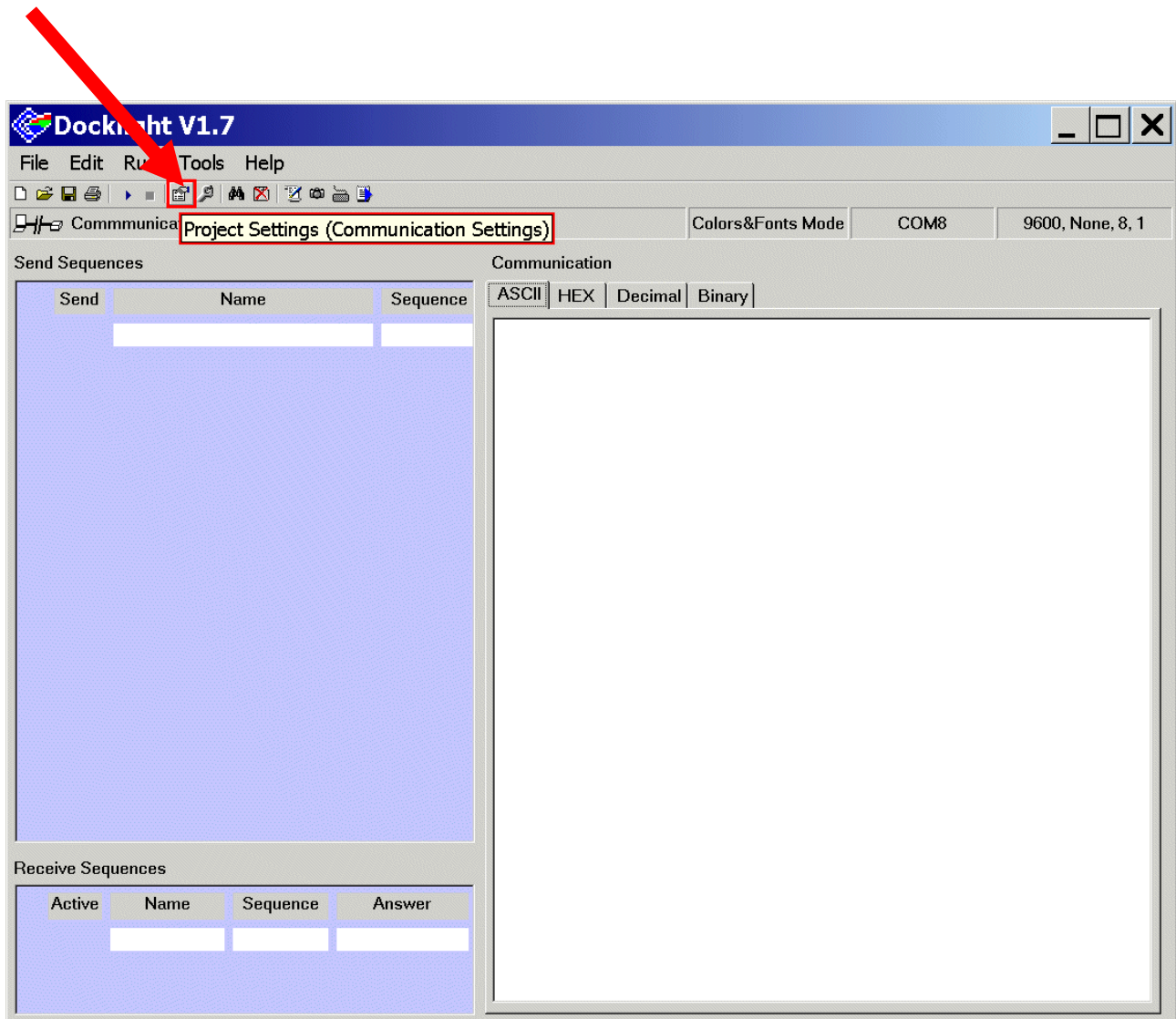


Now, **start** Docklight:





Click: Project Settings



Project Settings:

Communication: Communication Mode: **click** ☒ Send/Receive

Project Settings:

Communication: Communication Mode: Send/Receive on comm. channel: **select** COM5

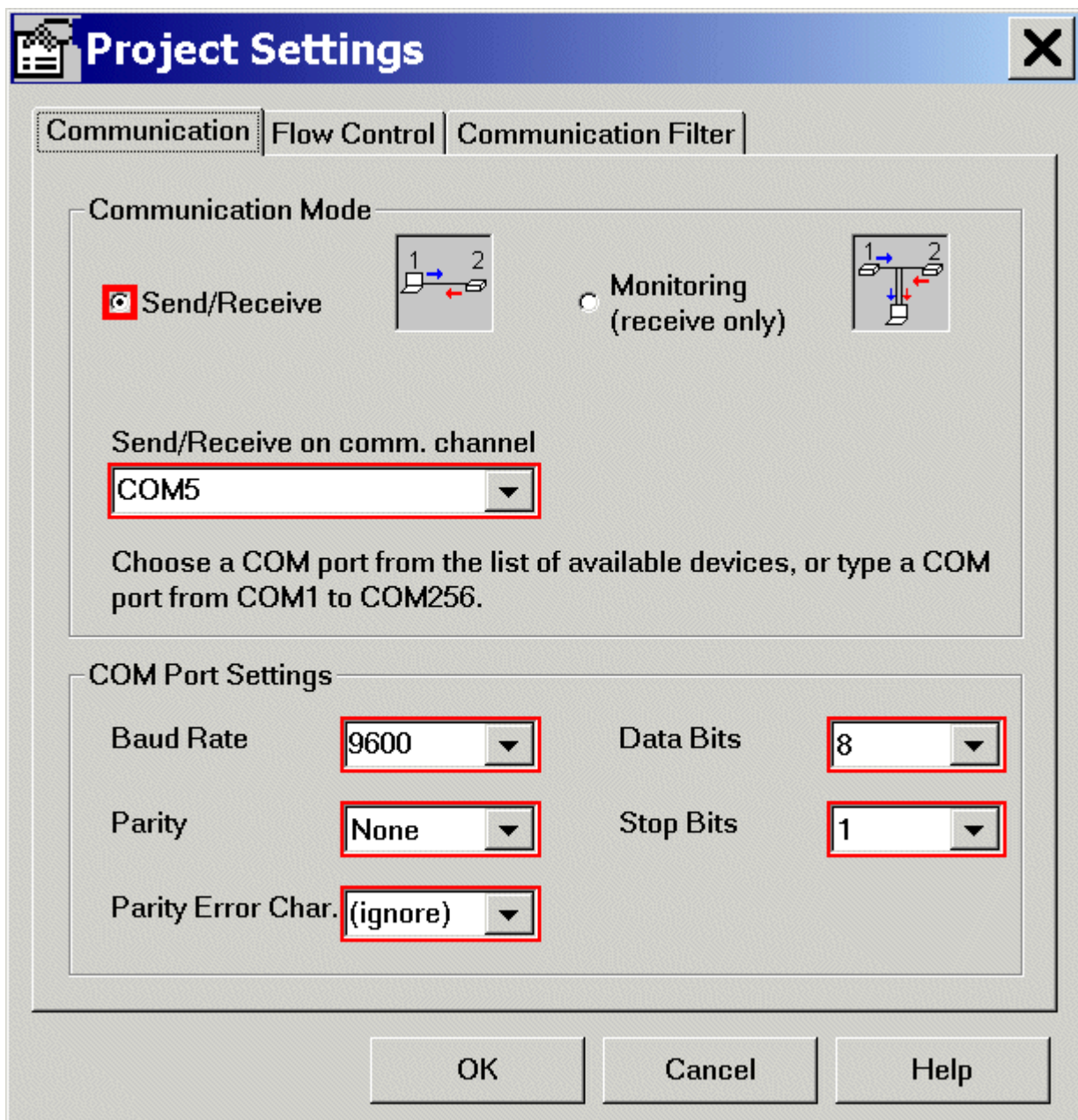
Project Settings: Communication: COM Port Settings: Baud Rate: **select** 9600

Project Settings: Communication: COM Port Settings: Parity: **select** None

Project Settings: Communication: COM Port Settings: Parity Error Char.: **select** (ignore)

Project Settings: Communication: COM Port Settings: Data Bits: **select** 8

Project Settings: Communication: COM Port Settings: Stop Bits: **select** 1



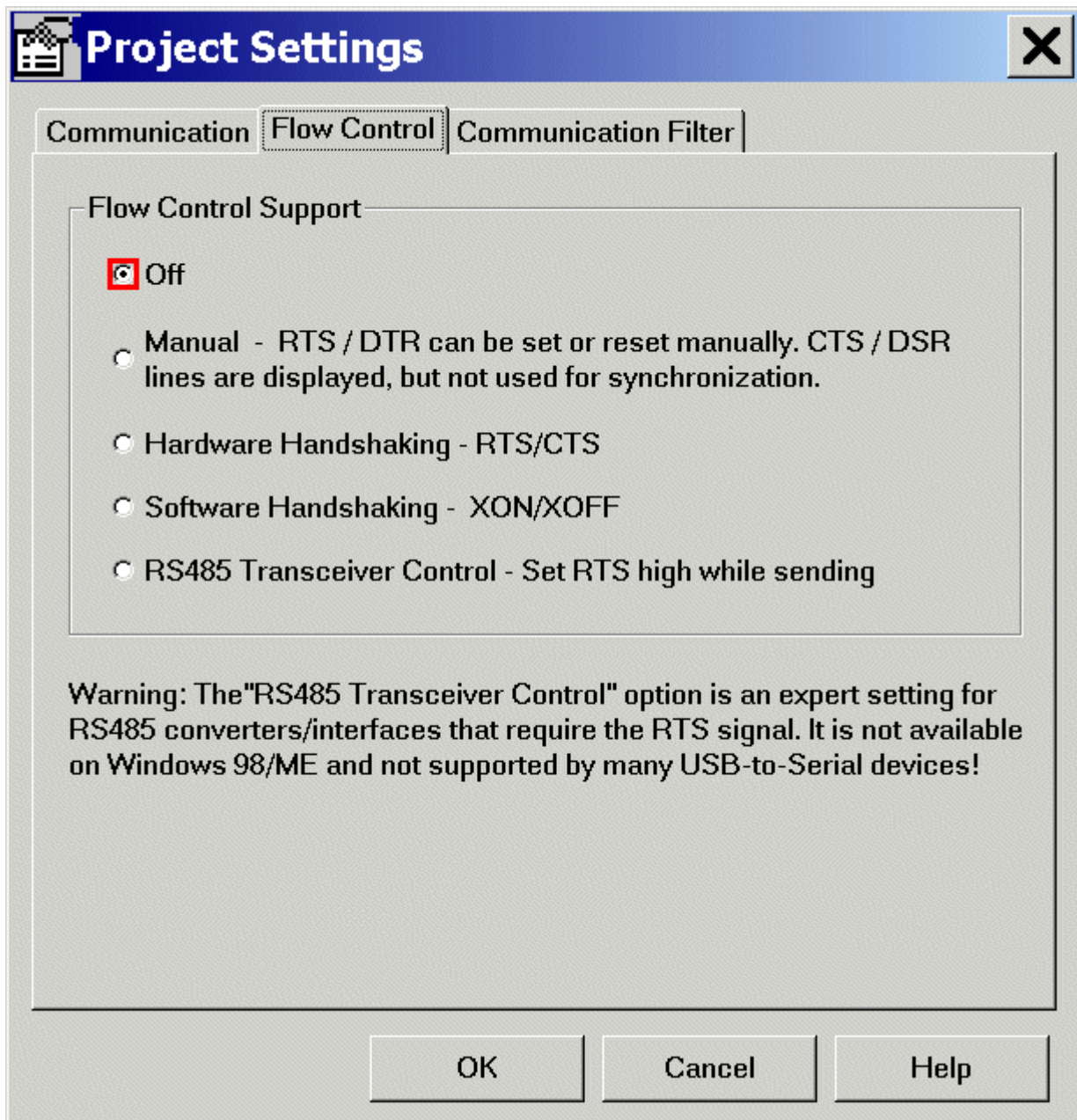
The image shows a 'Project Settings' dialog box with three tabs: 'Communication', 'Flow Control', and 'Communication Filter'. The 'Communication' tab is active. It contains two main sections: 'Communication Mode' and 'COM Port Settings'.

**Communication Mode:** This section has two options: 'Send/Receive' (selected with a checked radio button and a bidirectional arrow icon) and 'Monitoring (receive only)' (unselected with an unchecked radio button and a receive-only arrow icon). Below these is a dropdown menu labeled 'Send/Receive on comm. channel' with 'COM5' selected. A note below the dropdown states: 'Choose a COM port from the list of available devices, or type a COM port from COM1 to COM256.'

**COM Port Settings:** This section contains five dropdown menus, all of which are highlighted with red rectangles: 'Baud Rate' (9600), 'Data Bits' (8), 'Parity' (None), 'Stop Bits' (1), and 'Parity Error Char.' ((ignore)).

At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Help'.

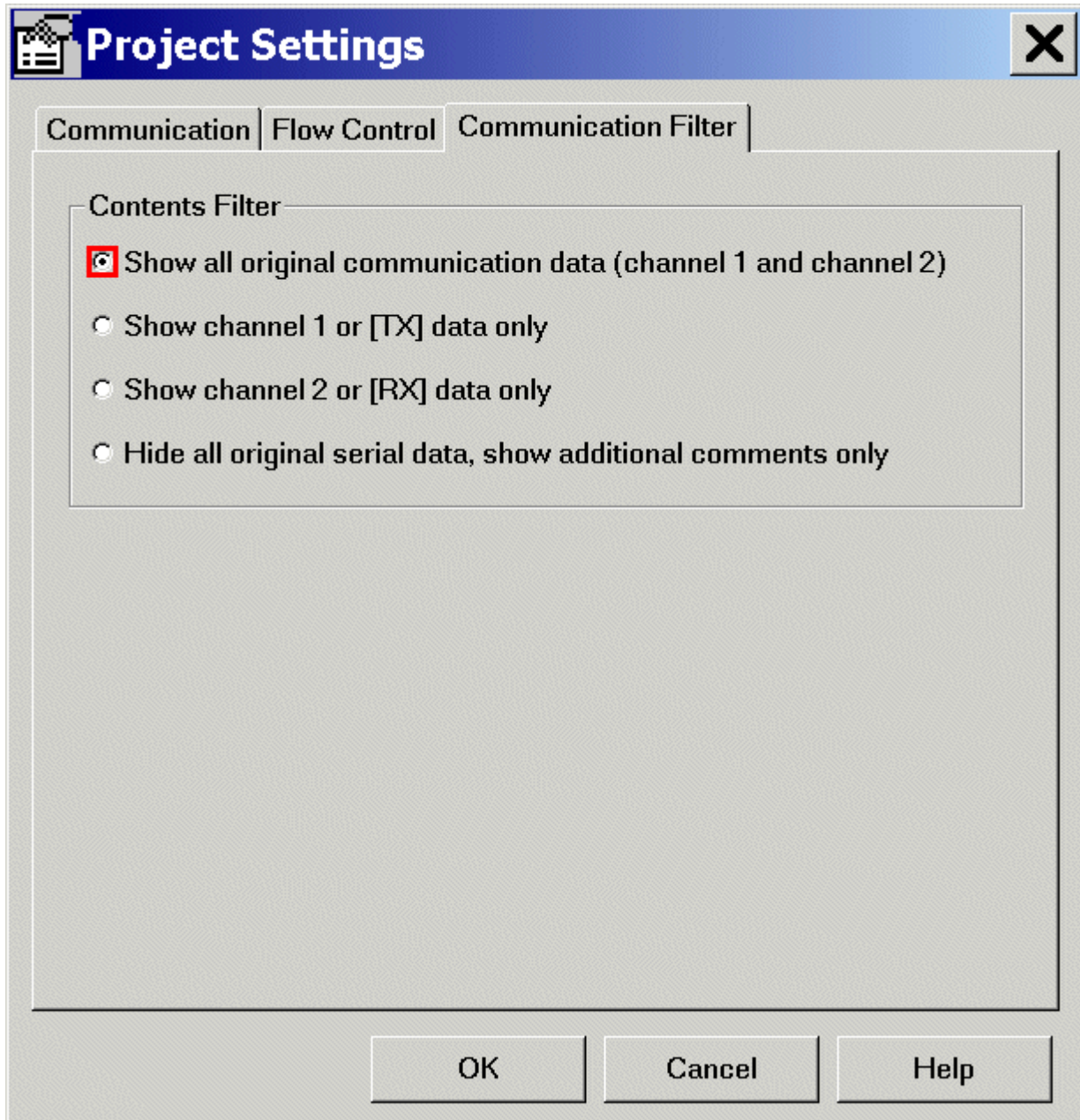
Project Settings: Flow Control: Flow Control Support: click  Off





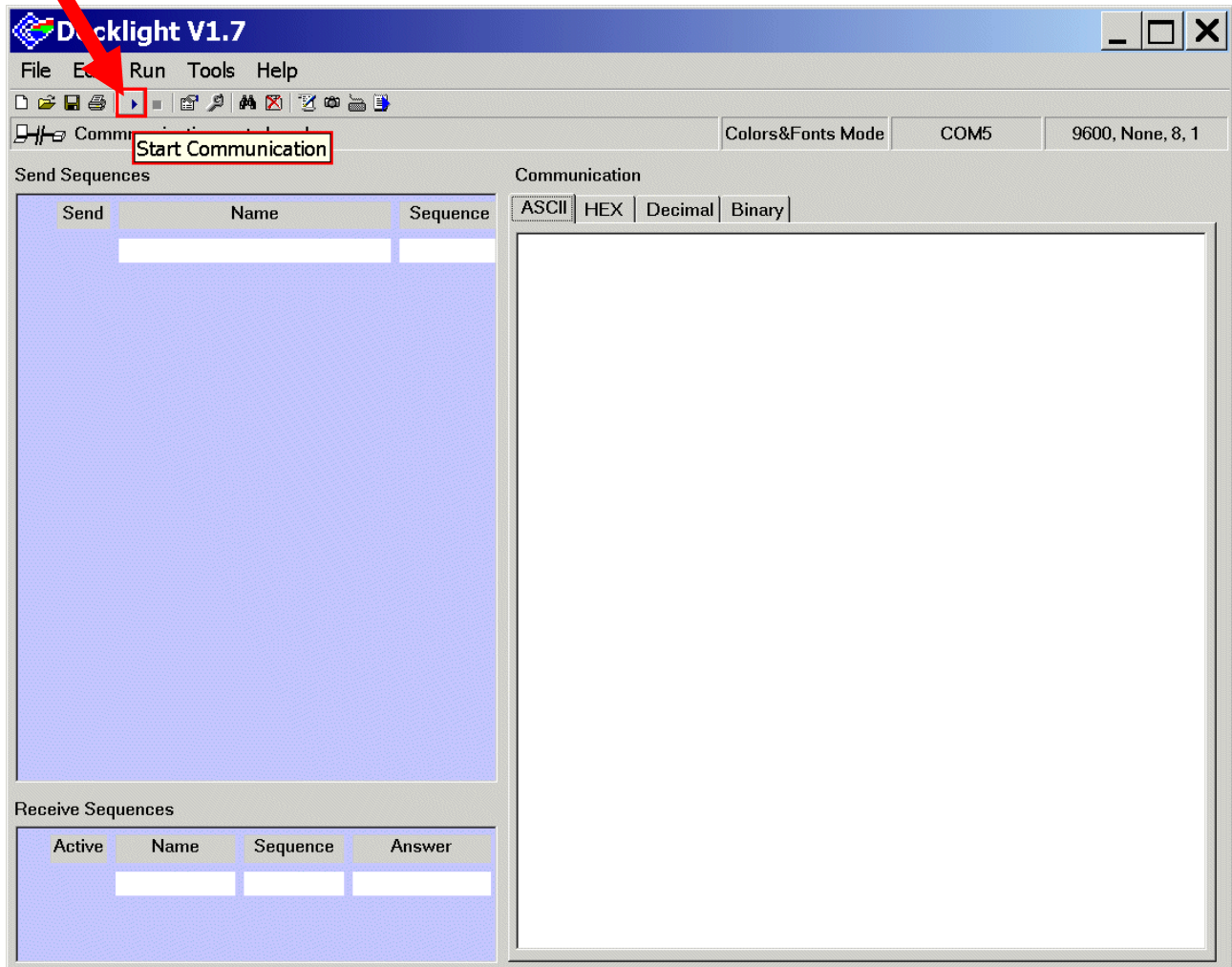
Project Settings:

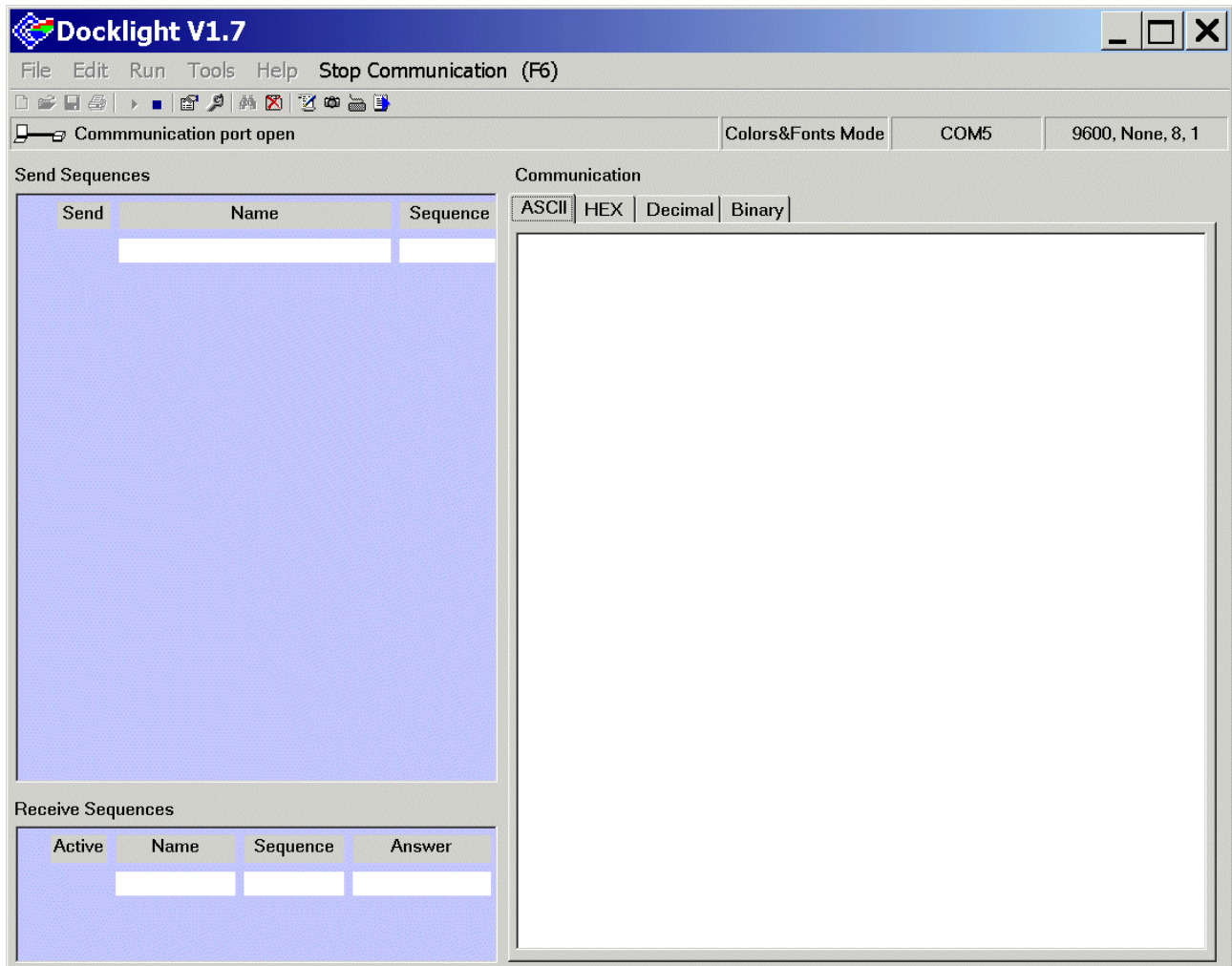
Communication Filter: Contents Filter: **click** ☒ Show all original communication data



OK

Click: 





**Note:**  
Docklight is now ready for serial communication!

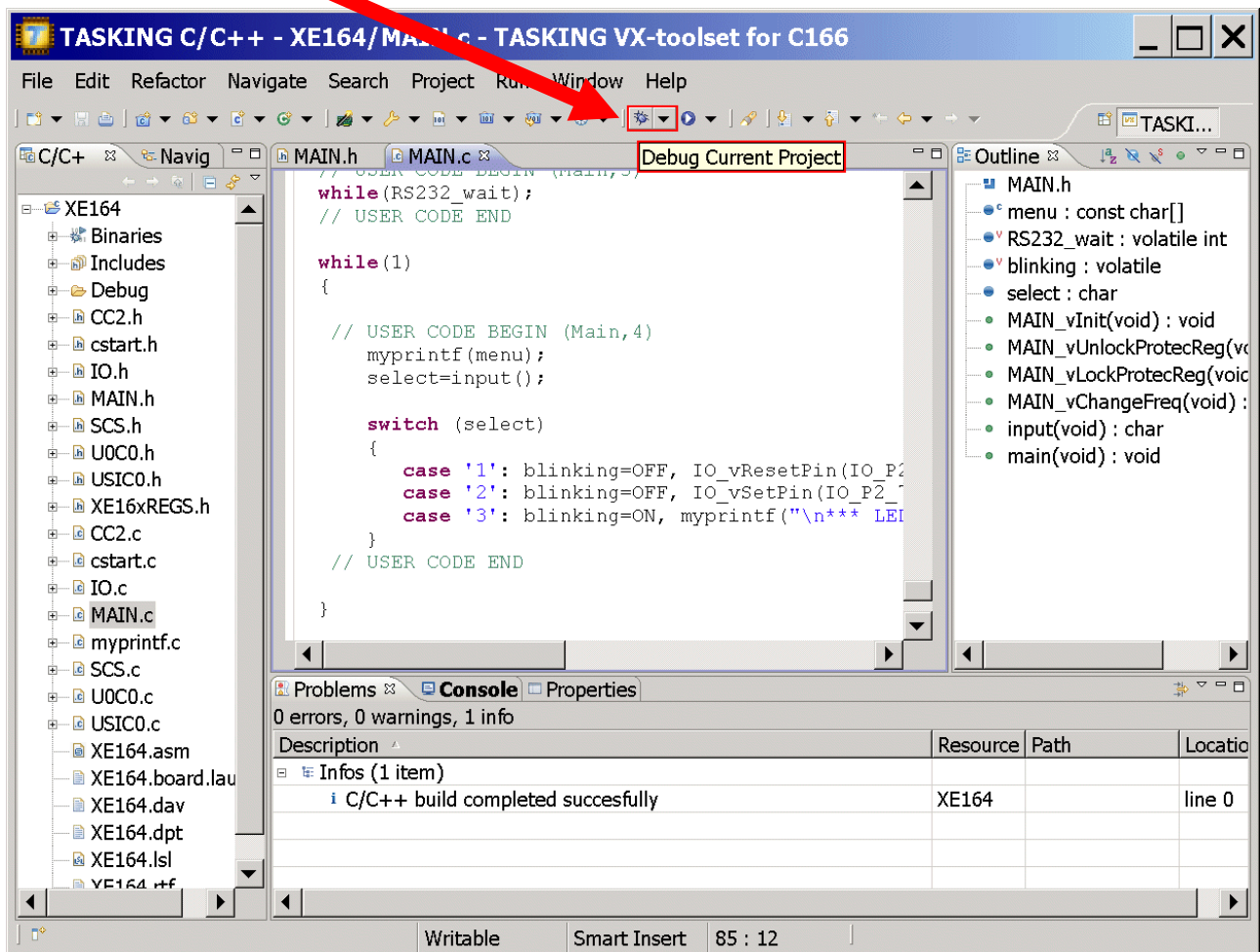


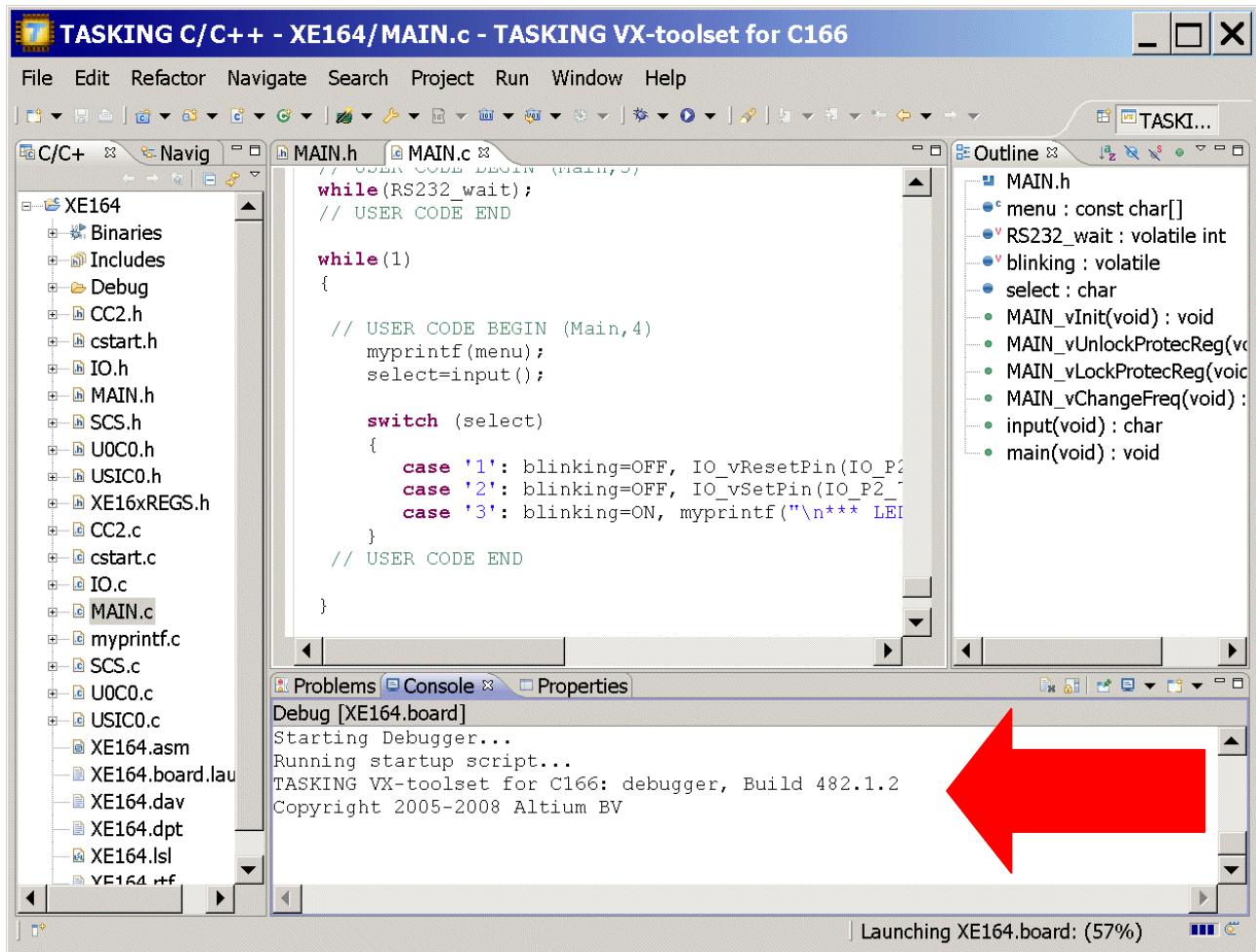


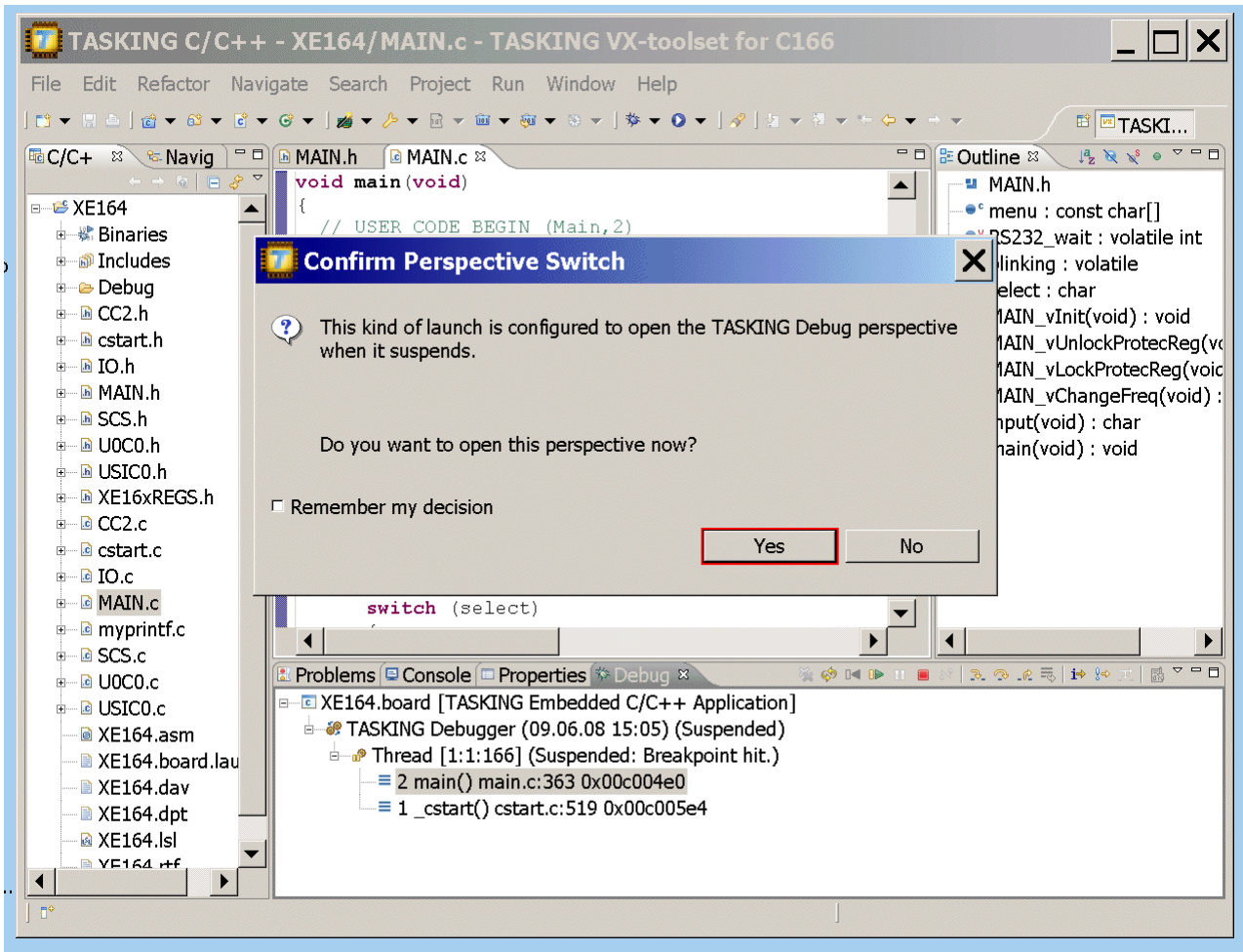


Go to Altium's TASKING VX-toolset for C166/ST10:

Click  Debug Current Project

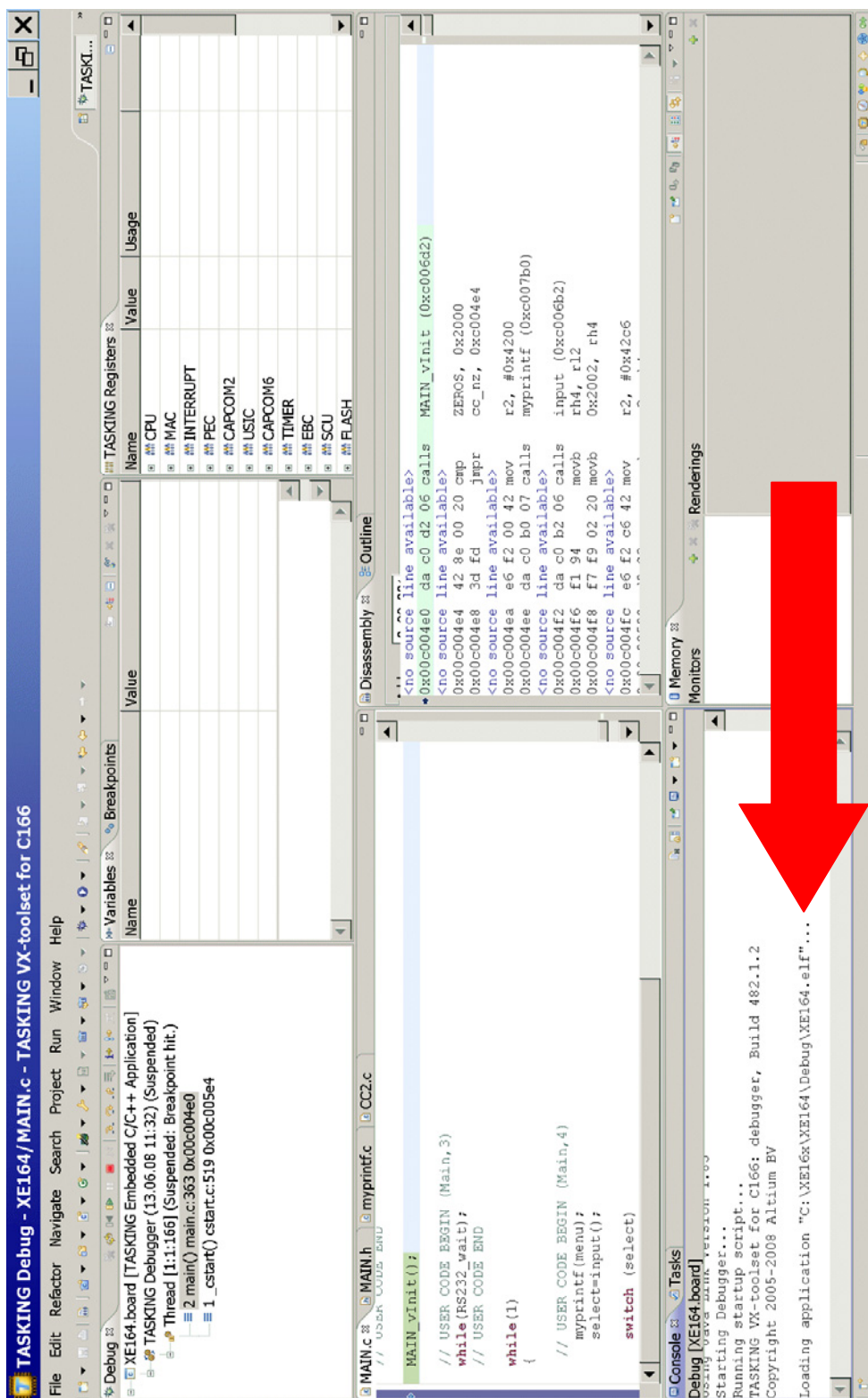






Click Yes




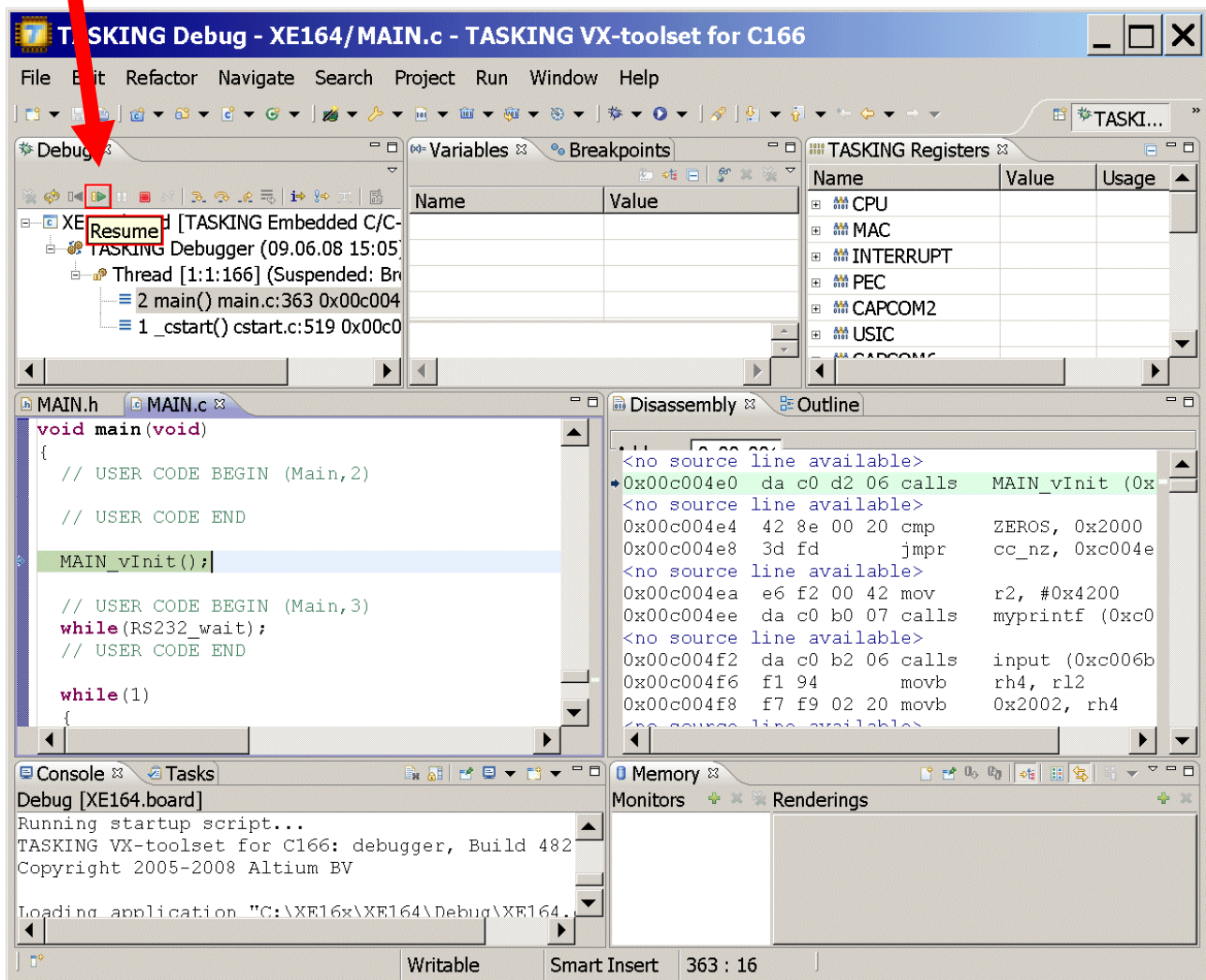


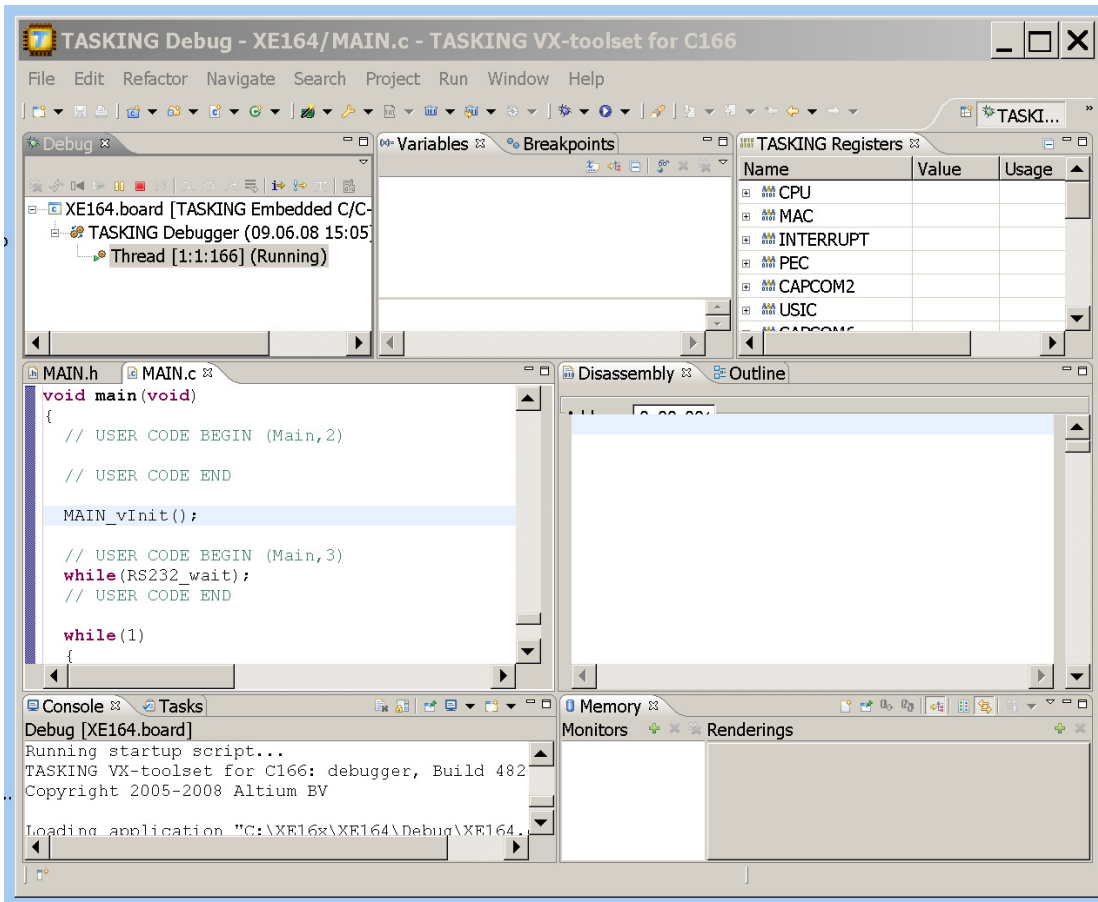
### Note:

As you can see in the screenshot above, the on chip program flash is automatically programmed when the debugger is launched.



Click  Resume (and start your program)





Note:

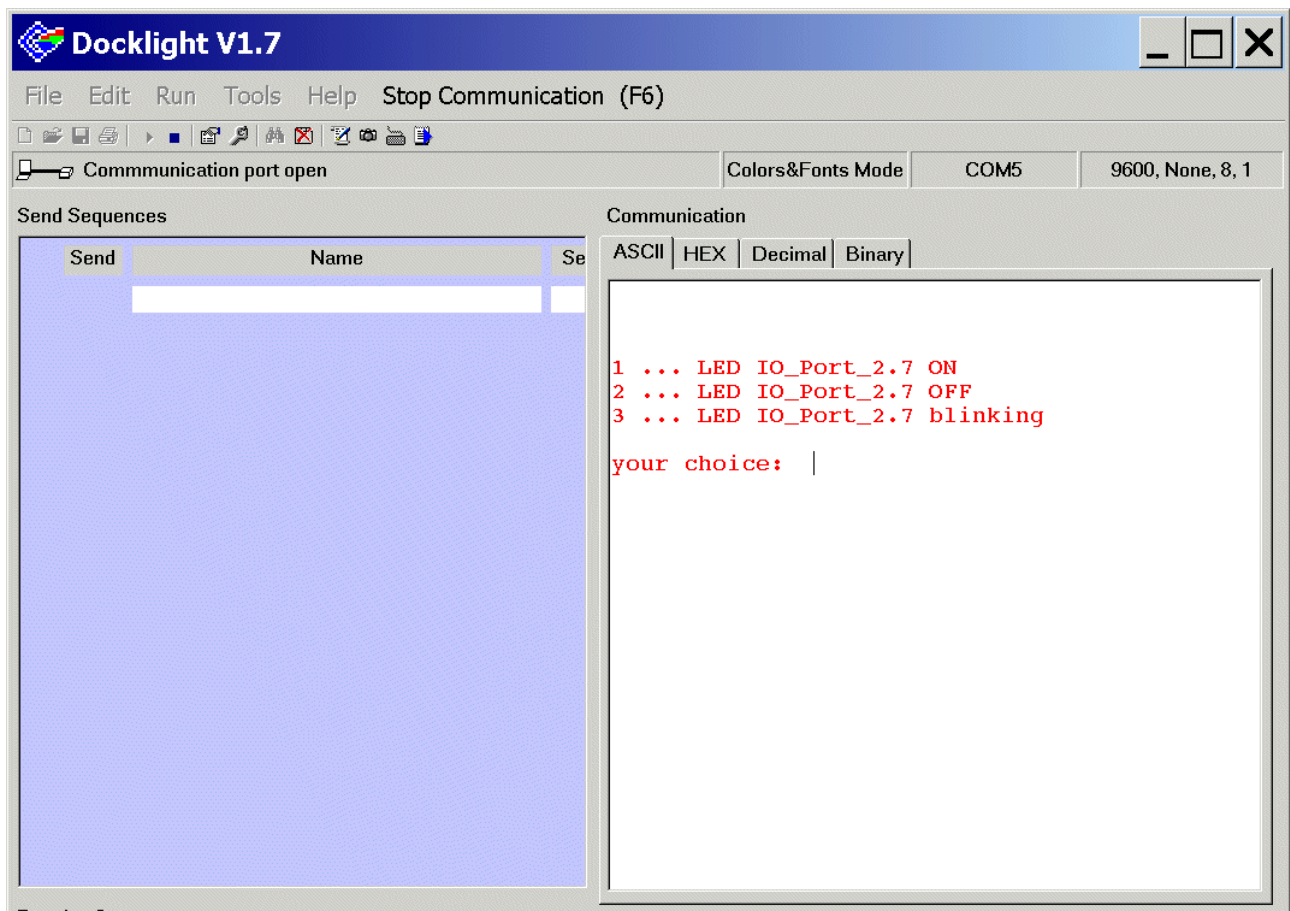
Your program is running!



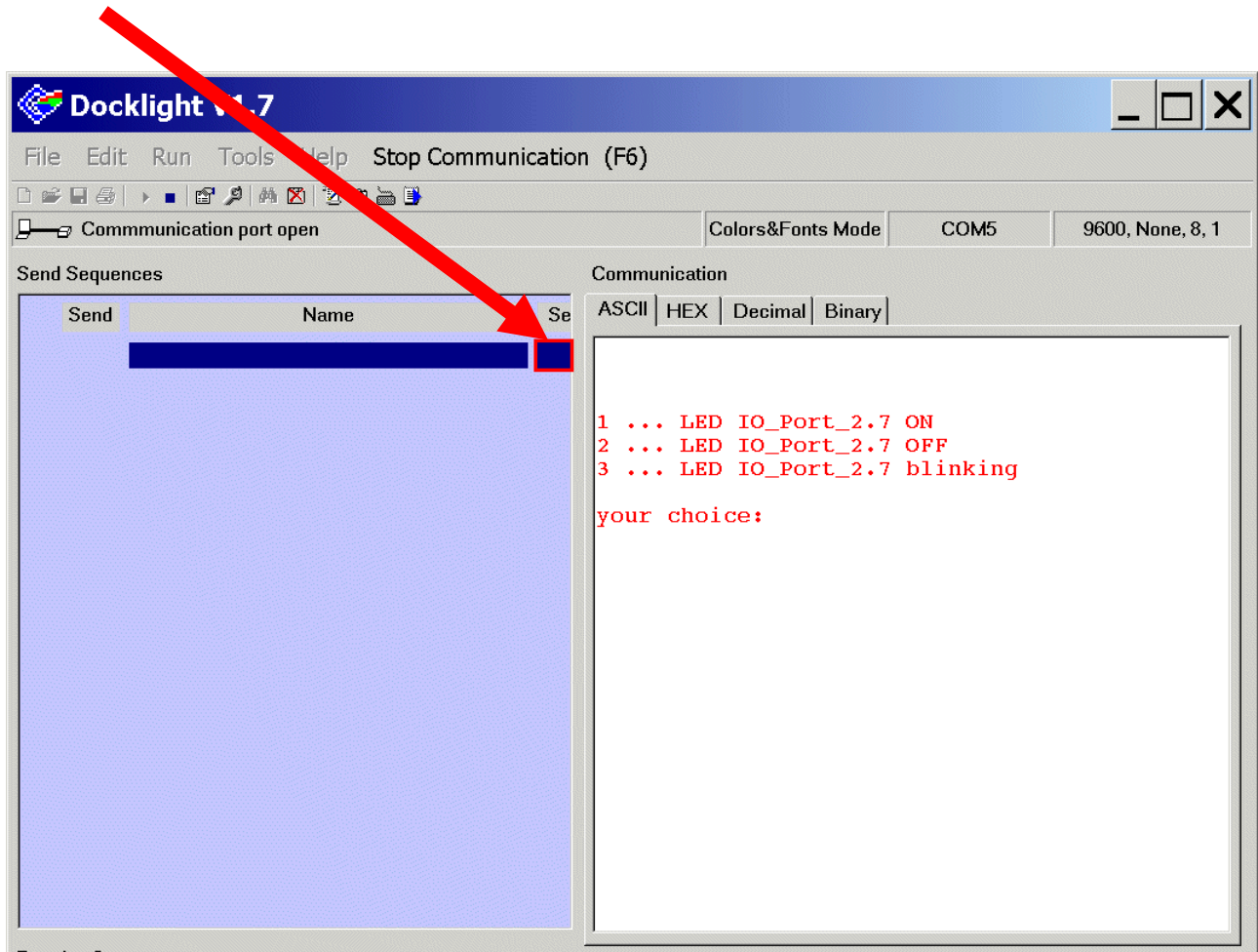




Go to Docklight and see the result:

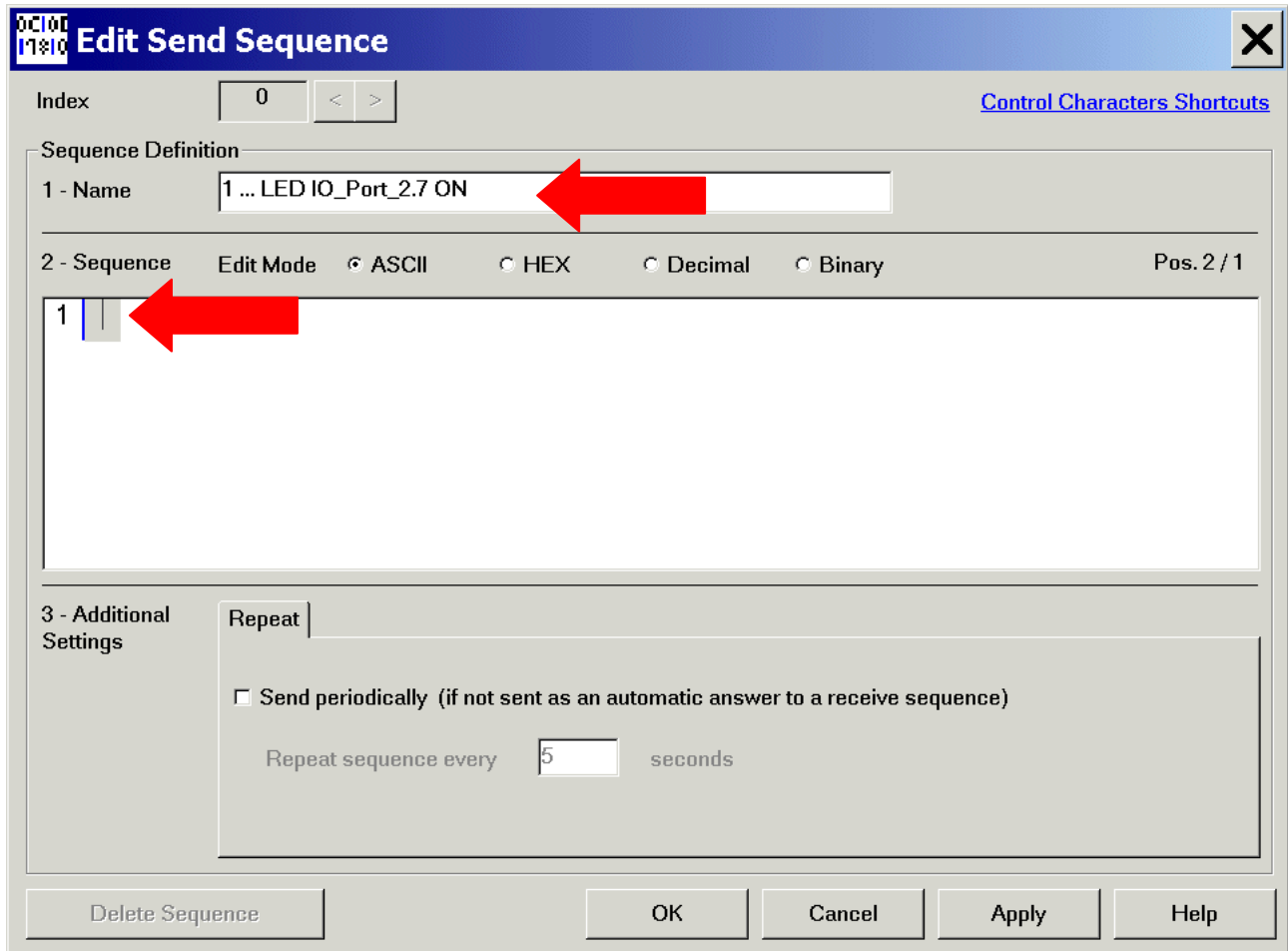


Double click inside the red box:



Edit Send Sequence: Sequence Definition: 1- Name: insert: 1 ... LED IO\_Port\_2.7 ON

Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 1



**Edit Send Sequence**

Index: 0

[Control Characters Shortcuts](#)

**Sequence Definition**

1 - Name: 1 ... LED IO\_Port\_2.7 ON

2 - Sequence: Edit Mode: ☒ ASCII ☐ HEX ☐ Decimal ☐ Binary Pos. 2 / 1

1

3 - Additional Settings

Repeat

☐ Send periodically (if not sent as an automatic answer to a receive sequence)

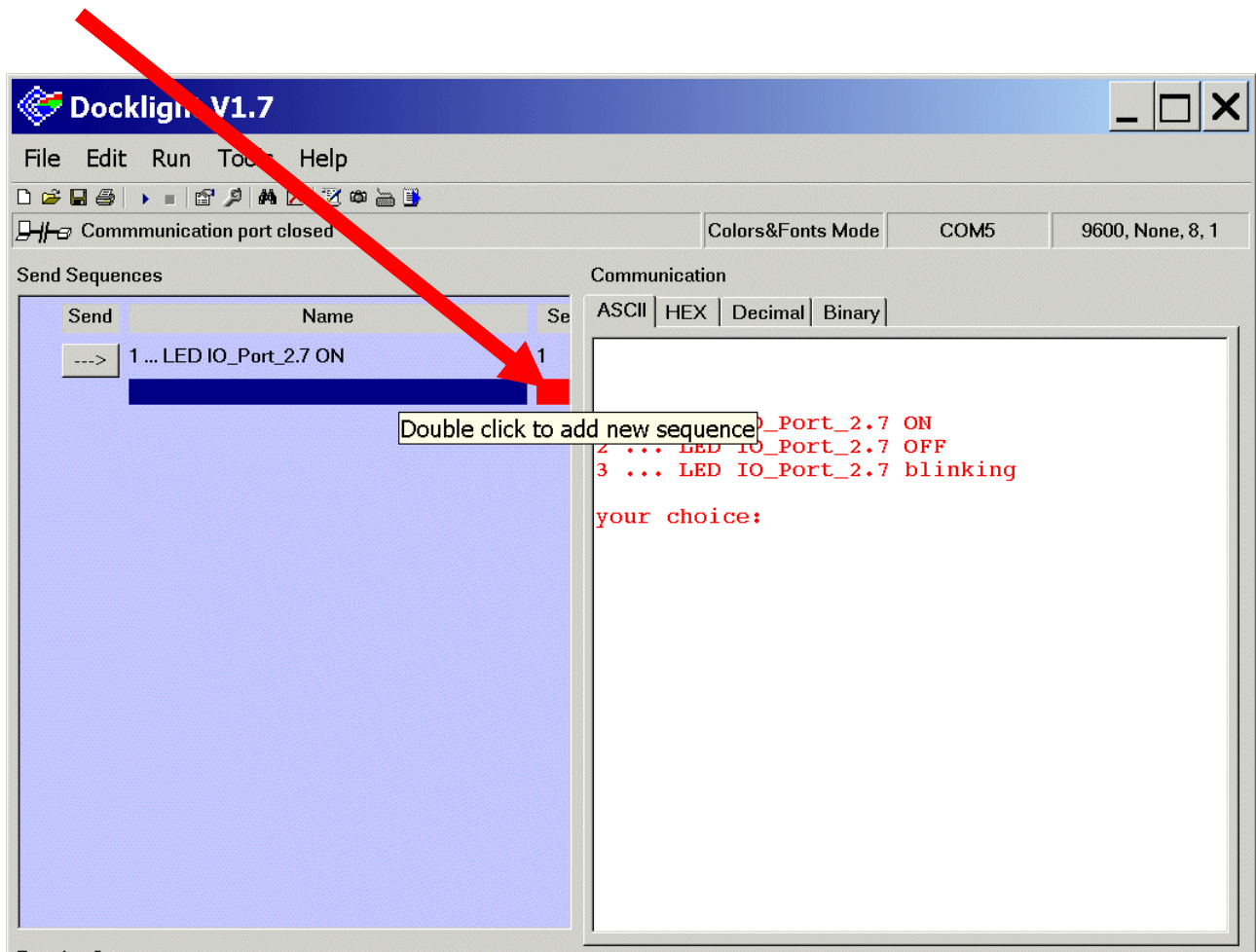
Repeat sequence every 5 seconds

Delete Sequence OK Cancel Apply Help

OK

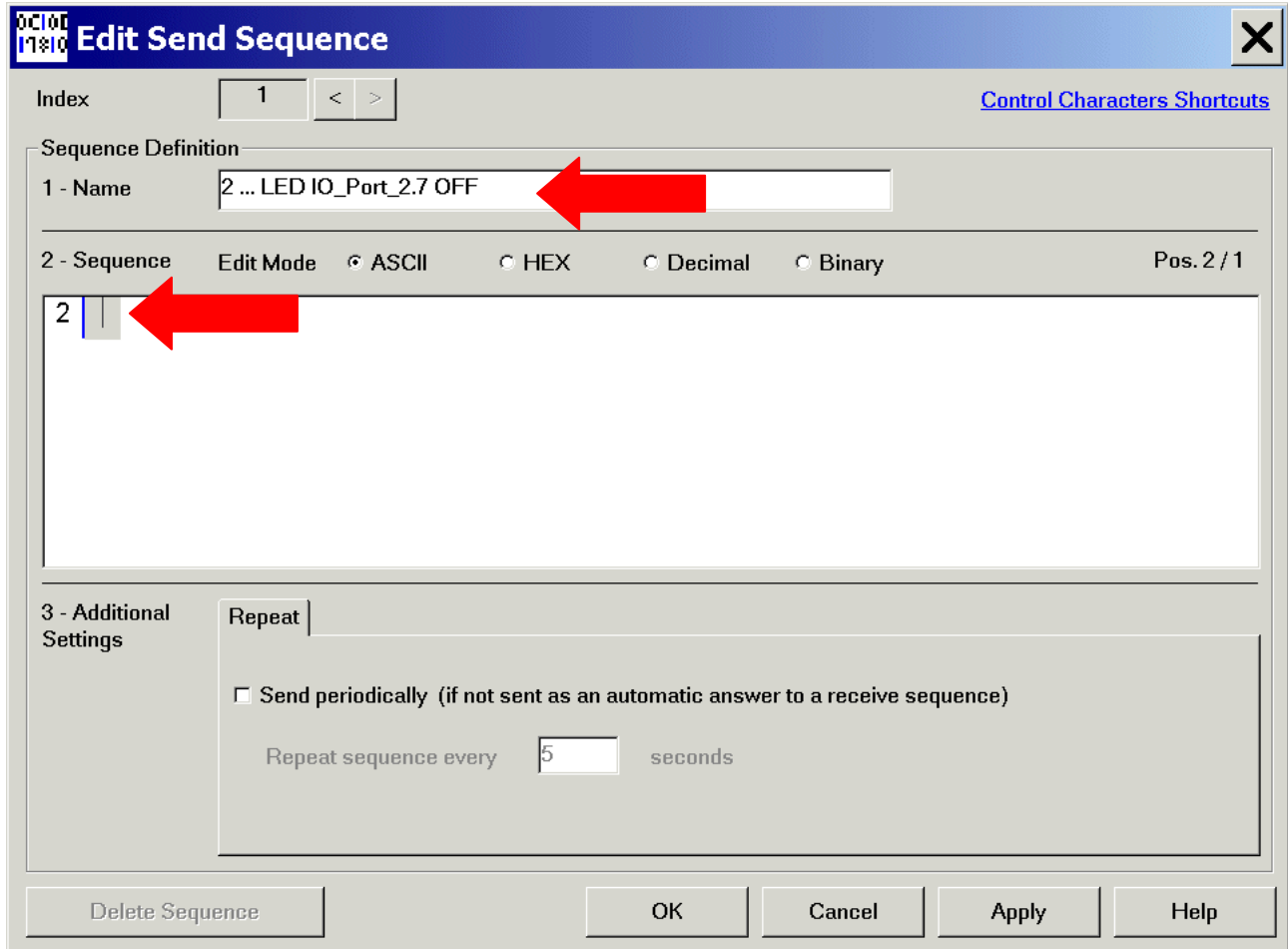


Double click inside the red box:



Edit Send Sequence: Sequence Definition: 1- Name: insert: 2 ... LED IO\_Port\_2.7 OFF

Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 2



**Edit Send Sequence**

Index: 1 < >

[Control Characters Shortcuts](#)

**Sequence Definition**

1 - Name: 2 ... LED IO\_Port\_2.7 OFF

2 - Sequence: Edit Mode (selected) | ASCII | HEX | Decimal | Binary | Pos. 2 / 1

2

**3 - Additional Settings**

Repeat

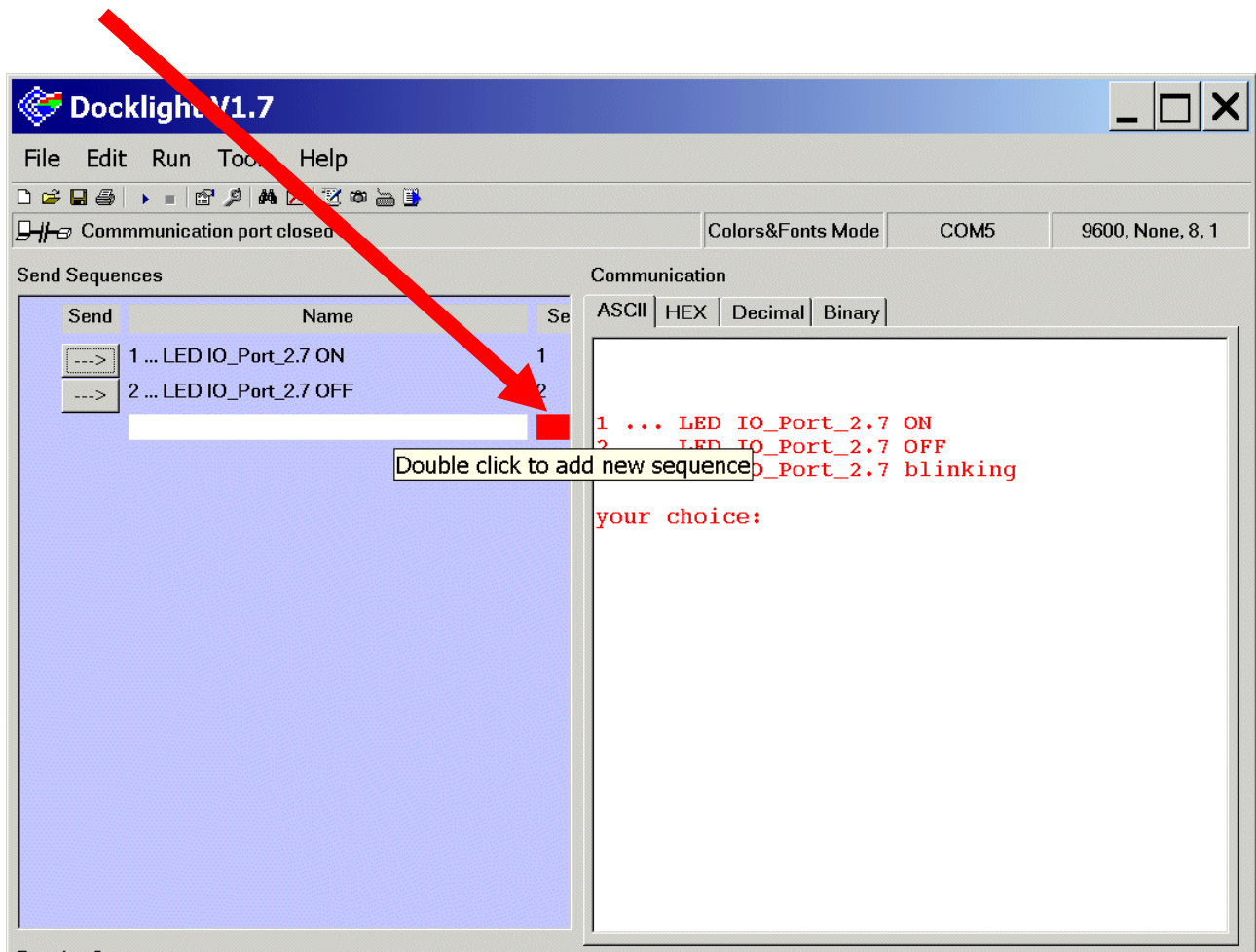
☐ Send periodically (if not sent as an automatic answer to a receive sequence)

Repeat sequence every 5 seconds

Delete Sequence OK Cancel Apply Help

OK

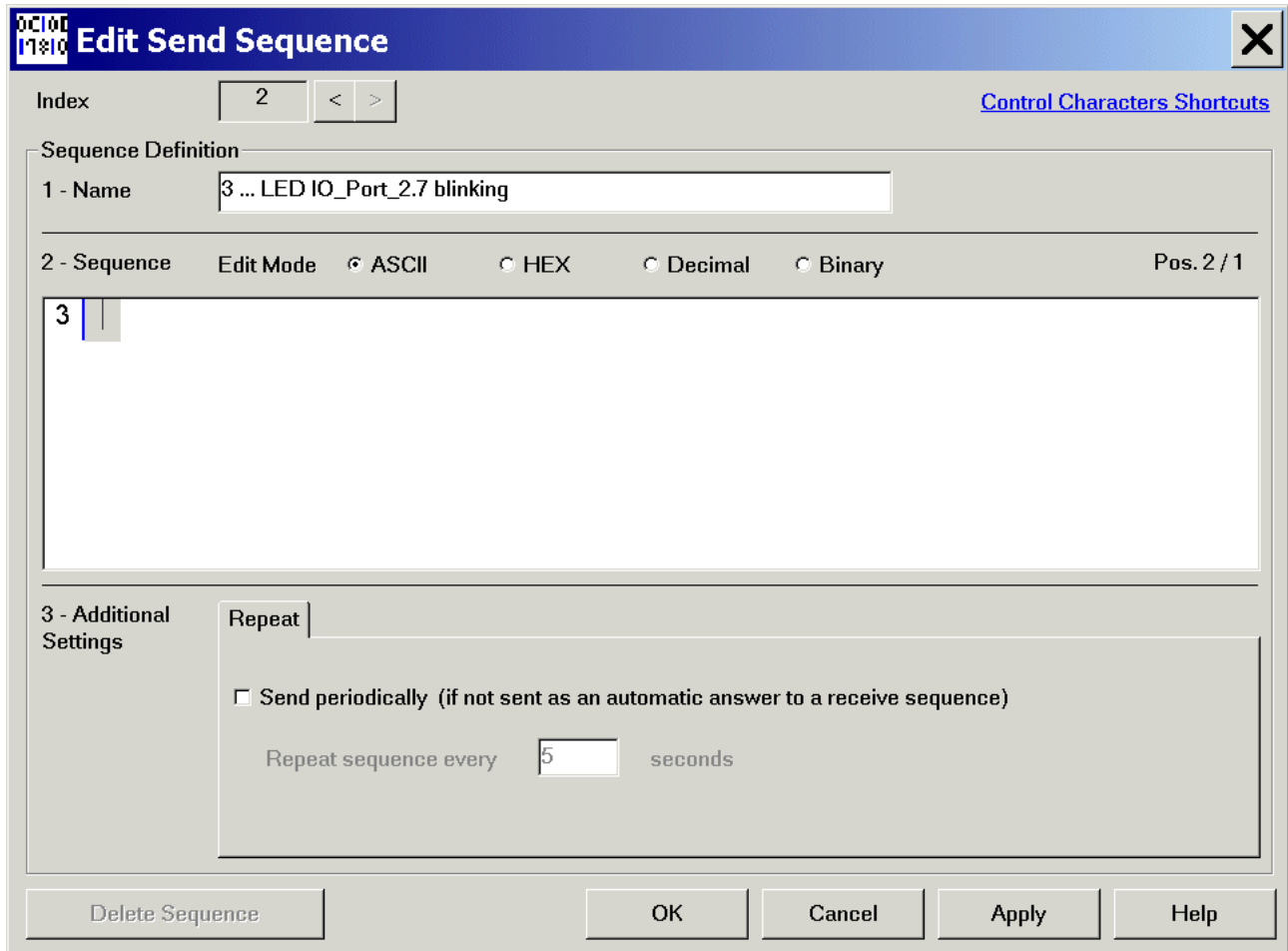
Double click inside the red box:





Edit Send Sequence: Sequence Definition: 1- Name: insert: 3 ... LED IO\_Port\_2.7 blinking

Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 3



**Edit Send Sequence**

Index: 2 < >

[Control Characters Shortcuts](#)

**Sequence Definition**

1 - Name: 3 ... LED IO\_Port\_2.7 blinking

2 - Sequence: Edit Mode: ☒ ASCII ☐ HEX ☐ Decimal ☐ Binary Pos. 2 / 1




3 - Additional Settings

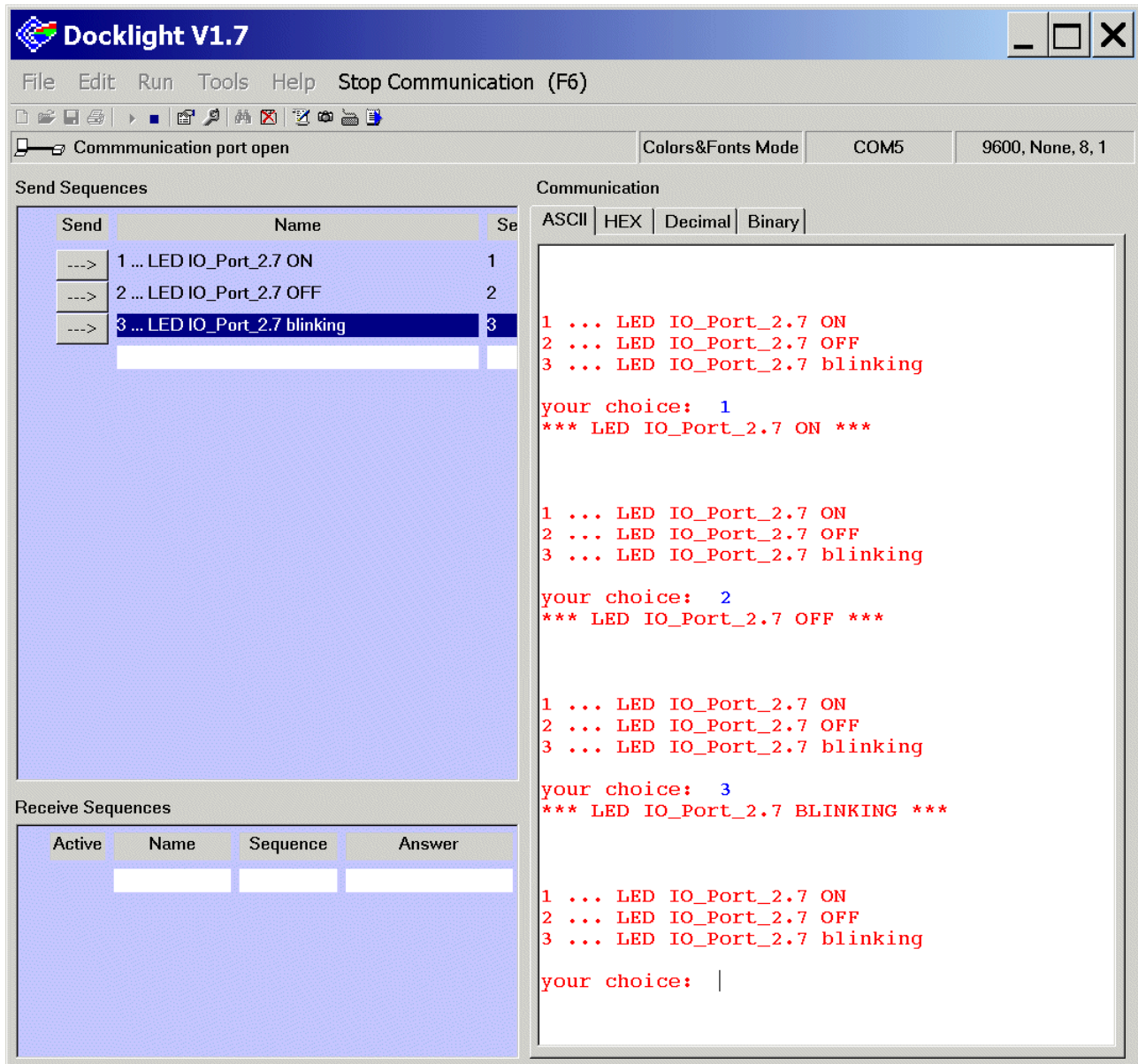
Repeat: ☐ Send periodically (if not sent as an automatic answer to a receive sequence)

Repeat sequence every 5 seconds

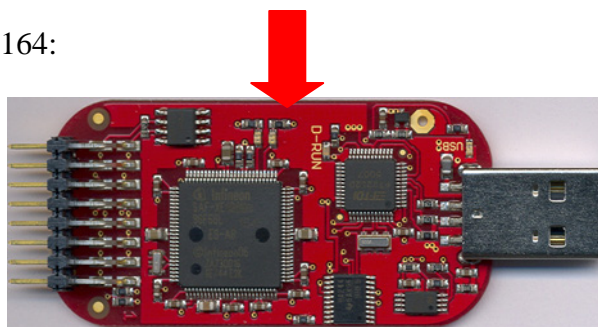
Buttons: Delete Sequence, OK, Cancel, Apply, Help

OK

- Click  1 ... LED IO\_Port\_2.7 ON or  
 Click  2 ... LED IO\_Port\_2.7 OFF or  
 Click  3 ... LED IO\_Port\_2.7 blinking

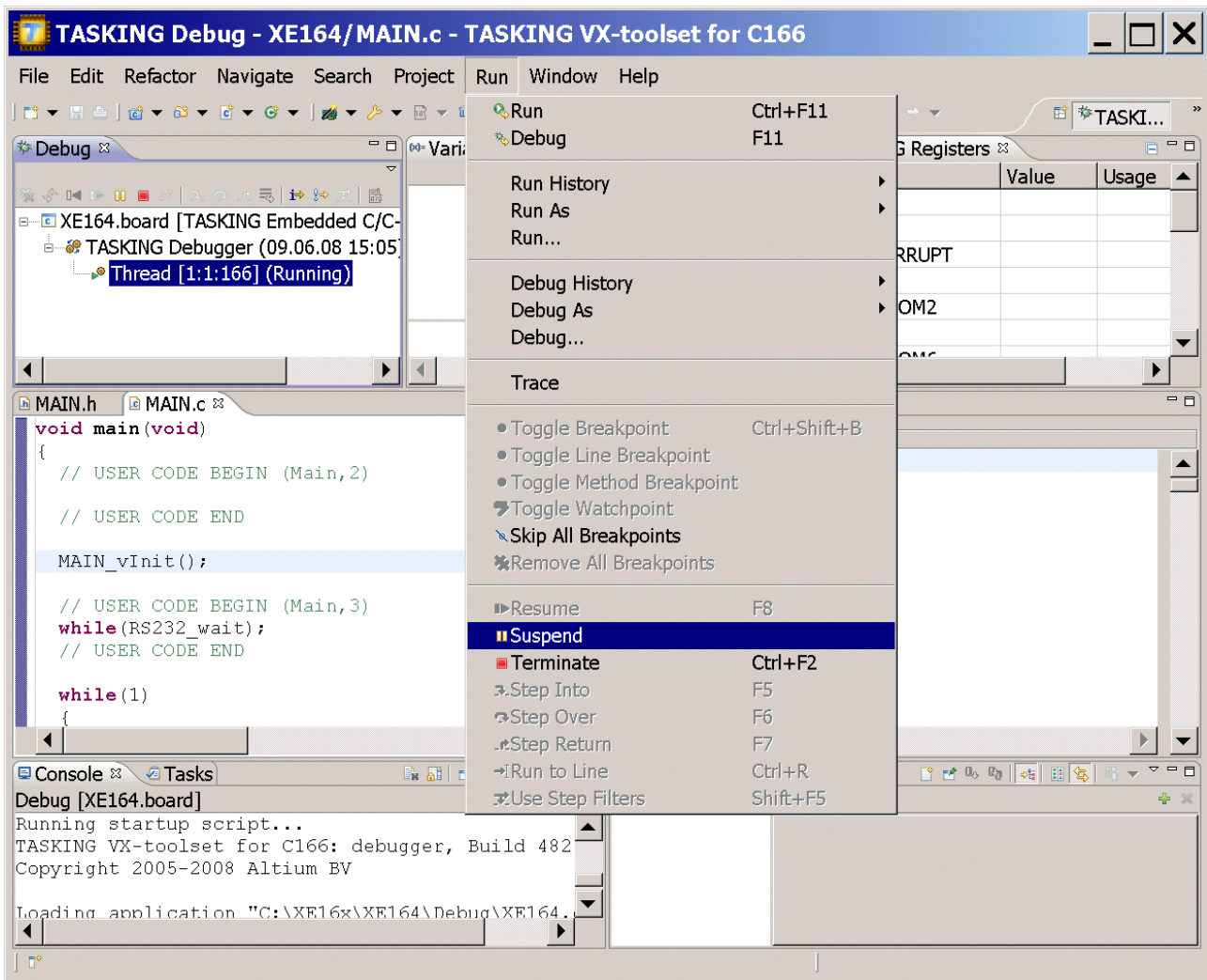


and **check** the results on your UConnect-CAN XE164:



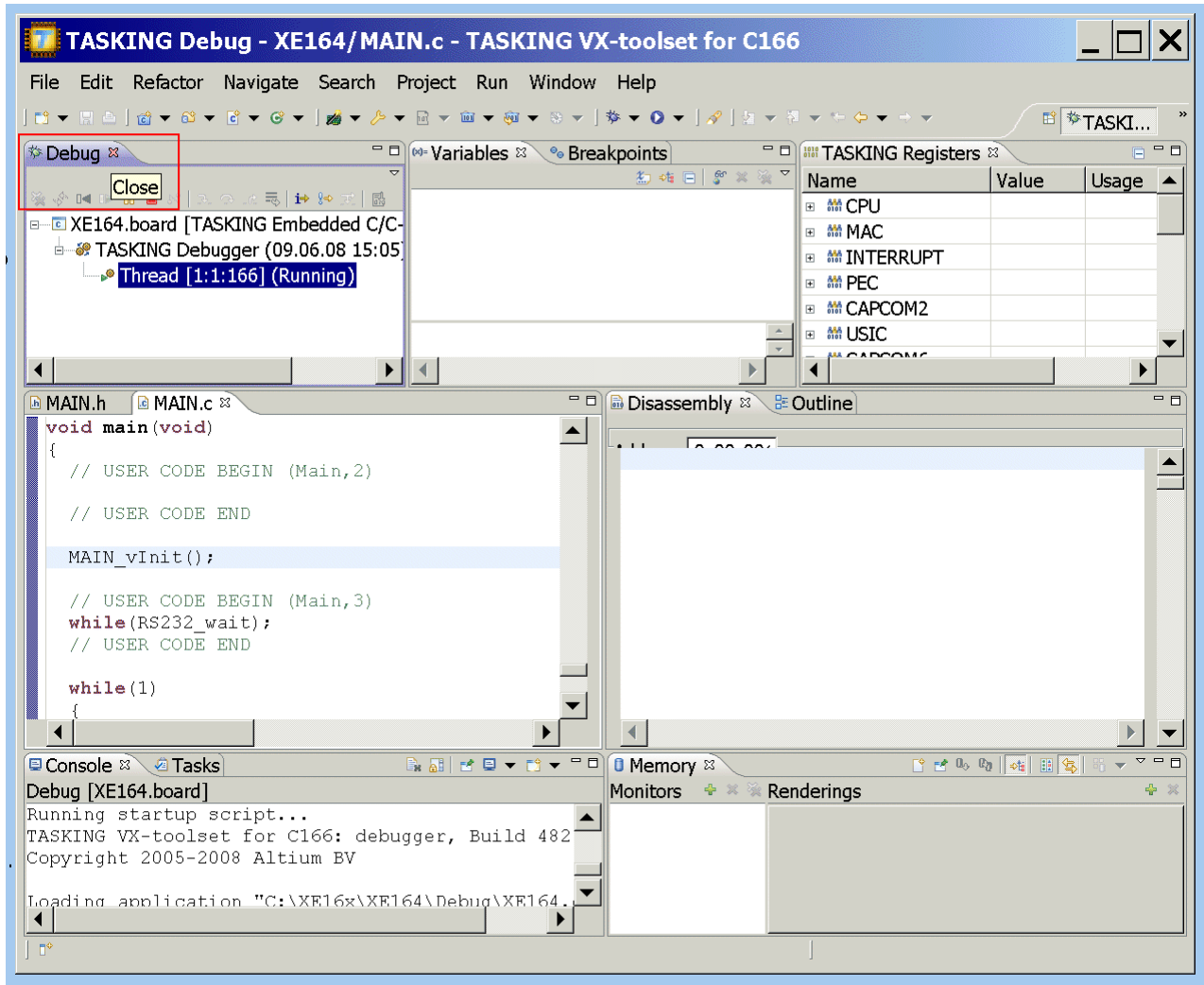
Now we close Altium's TASKING VX-toolset (Debugger):

**Run – Suspend**

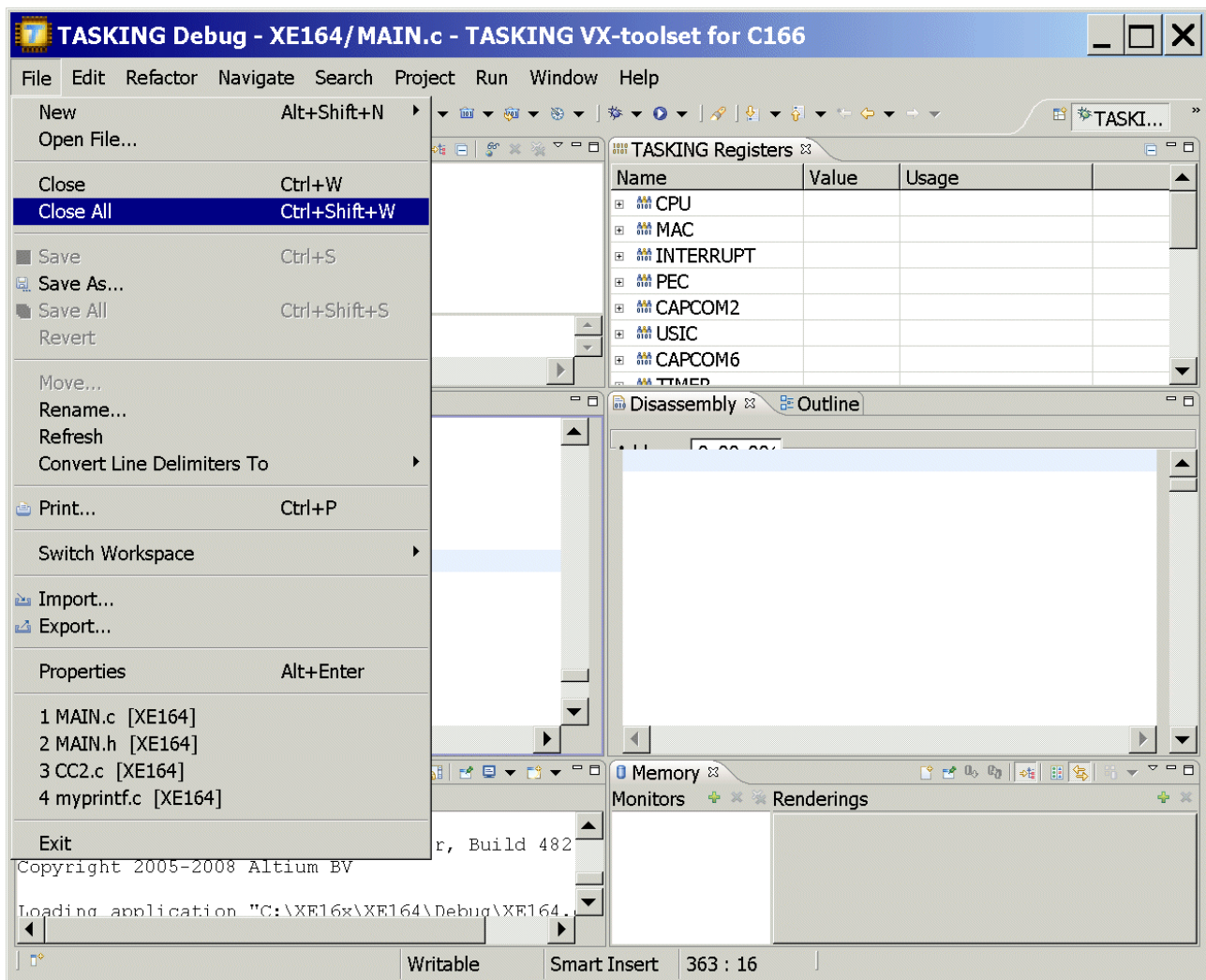




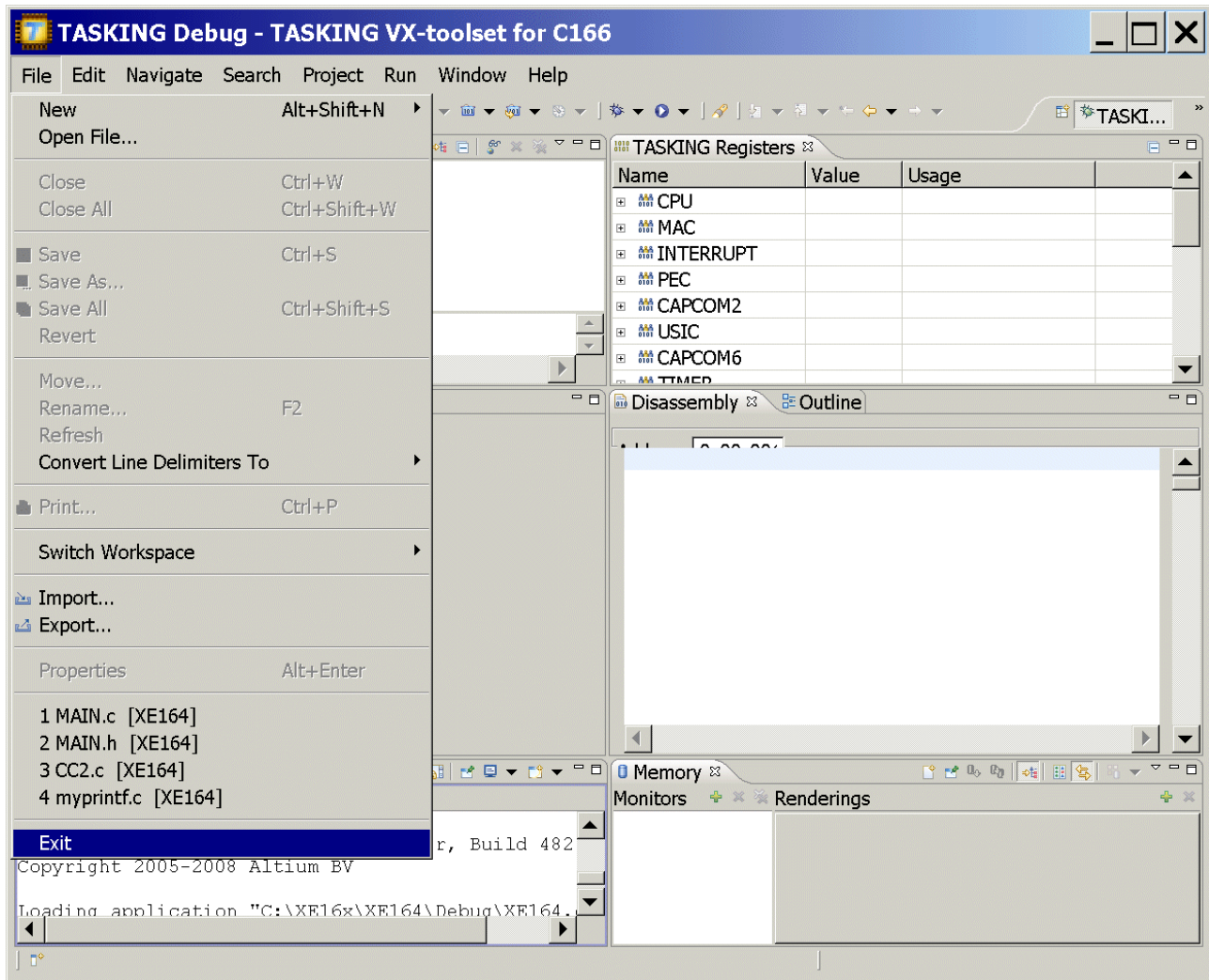
## Debug – Close



## File – Close All



## File – Exit







### Conclusion:

In this step-by-step book you have learned how to use the UConnect-CAN XE164 together with Altium's TASKING VX-toolset for C166/ST10.

Now you can easily expand your "hello world" program to suit your needs!

You can connect either a part of - or your entire application to the UConnect-CAN XE164.

You are also able to benchmark any of your algorithms to find out if the selected microcontroller fulfils all the required functions within the time frame needed.

Have fun and enjoy working with XE16x microcontrollers!

### Note:

There are step-by-step books for 8 bit microcontrollers (e.g. XC866 and XC888), 16 bit microcontrollers (e.g. C16x, XC16x and XE16x/XC2xxx) and 32 bit microcontrollers (e.g. TC1796 and TC1130).

All these step-by-step books use the same microcontroller resources and the same example code.

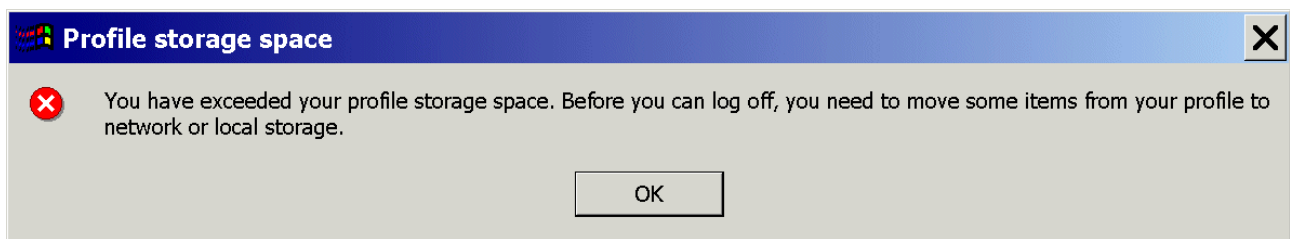
This means: configuration steps, function names and variable names are identical.

This should give you a good opportunity to get in touch with another Infineon microcontroller family or tool chain!

There are even more programming examples using the same style available [e.g. ADC-examples, CAPCOM6-examples (e.g. BLDC-Motor, playing music), Simulator-examples, C++ examples] based on these step-by-step books.

## 6.) Hint: Profile Storage Space:

To avoid:



Do what is explained on the following 2 slides:

## Tasking Viper "Profile Storage Space Exceeded"



### ■ Edit config.ini

□ C:\Program Files\TASKING\C166-VX v2.2r3\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_cl66_v2.2r3/config
osgi.configuration.area = C:/UserData/_login-name_/.eclipse_cl66_v2.2r3/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_cl66_v2.2r3
osgi.user.area = C:/UserData/_login-name_/.eclipse_cl66_v2.2r3

# The default workspace location
# osgi.instance.area.default = @user.home/workspace_cl66_v2.2r3
osgi.instance.area.default = C:/UserData/_login-name_/workspace_cl66_v2.2r3

# 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
```

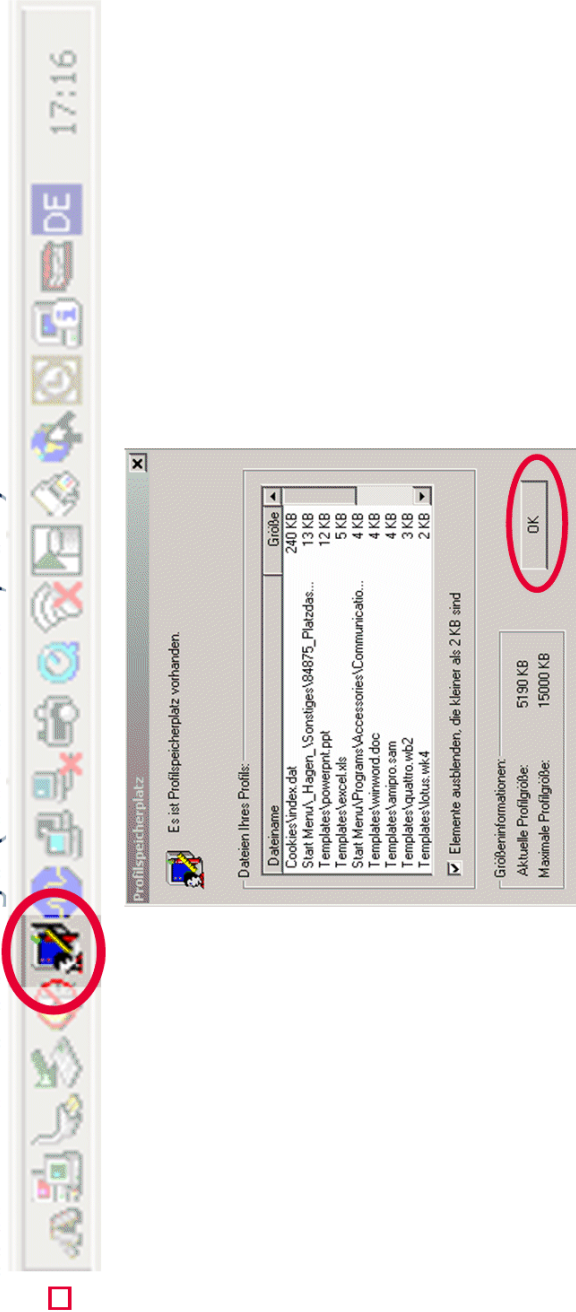


Page 1



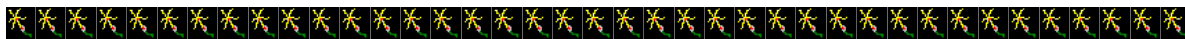
## Tasking Viper "Profile Storage Space Exceeded"

- Delete old directories in profile space
  - C:\Documents and Settings\login-name\.eclipse\_c166\_v2.2r3
- Rescan Profile Storage (double click; OK)



Page 2

**7.) Feedback (UConnect-CAN XE164, TASKING VX):**  
**Your opinion, suggestions and/or criticisms**



**Contact Details (this section may remain blank should you wish to offer feedback anonymously):**

---

---

---

If you have any suggestions please send this sheet back to:

**email:** [mcdocu.comments@infineon.com](mailto:mcdocu.comments@infineon.com)

**FAX:** +43 (0) 4242 3020 5783



**Your suggestions:**

---

---

---

---

---

---

---

---

---

<http://www.infineon.com>