AP16139

XE164

UConnect-CAN XE164 "Cookery Book" for a hello world application using Altium's TASKING classic C166 toolset (this toolset is available free of charge for one year)

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**).

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

Table of Contents **Page**

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.

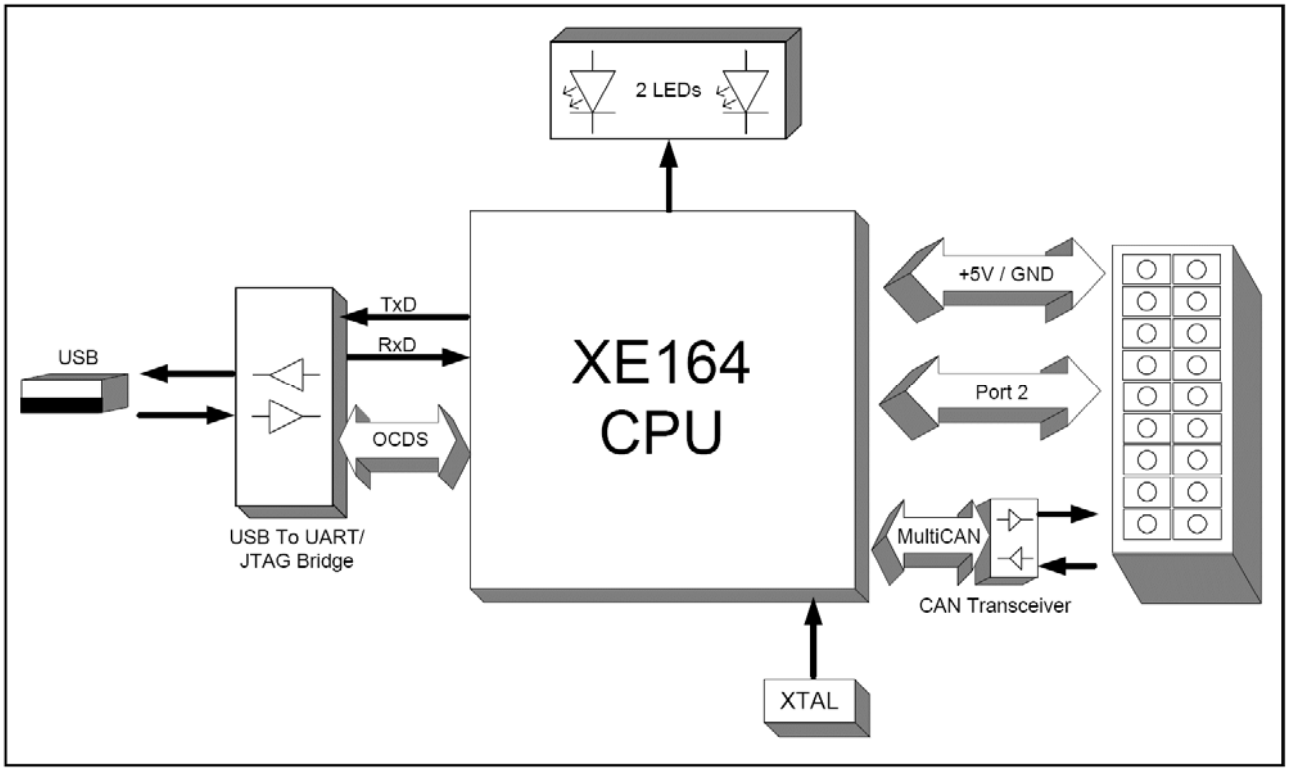
Note:
The style used in this document focuses on working through this material as fast and easily as possible. That means there are full screenshots instead of dialog-window-screenshots; extensive use of colours and page breaks; and listed source-code is not formatted to ease copy & paste.

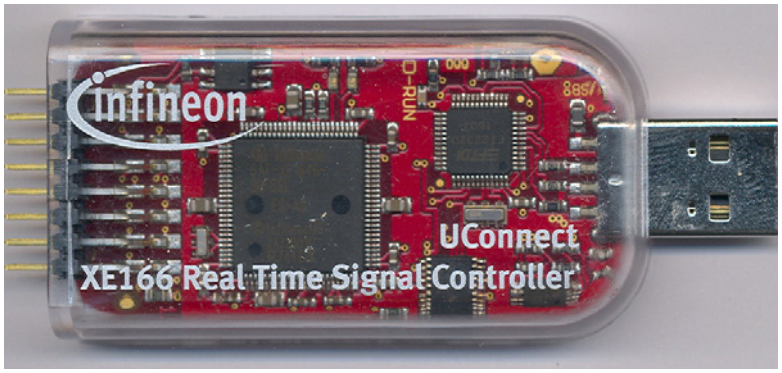
Have fun and enjoy the UConnect-CAN XE164!

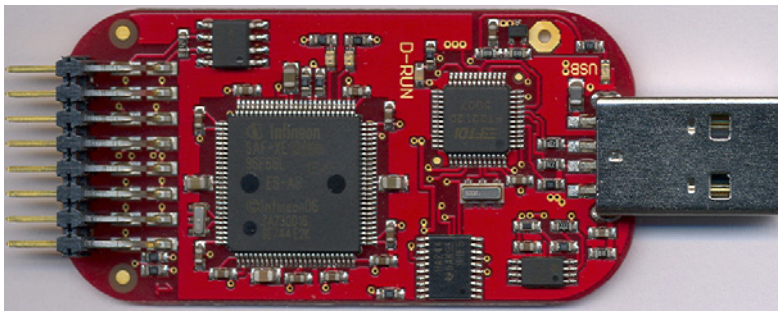
Programming Example

UConnect-CAN XE164

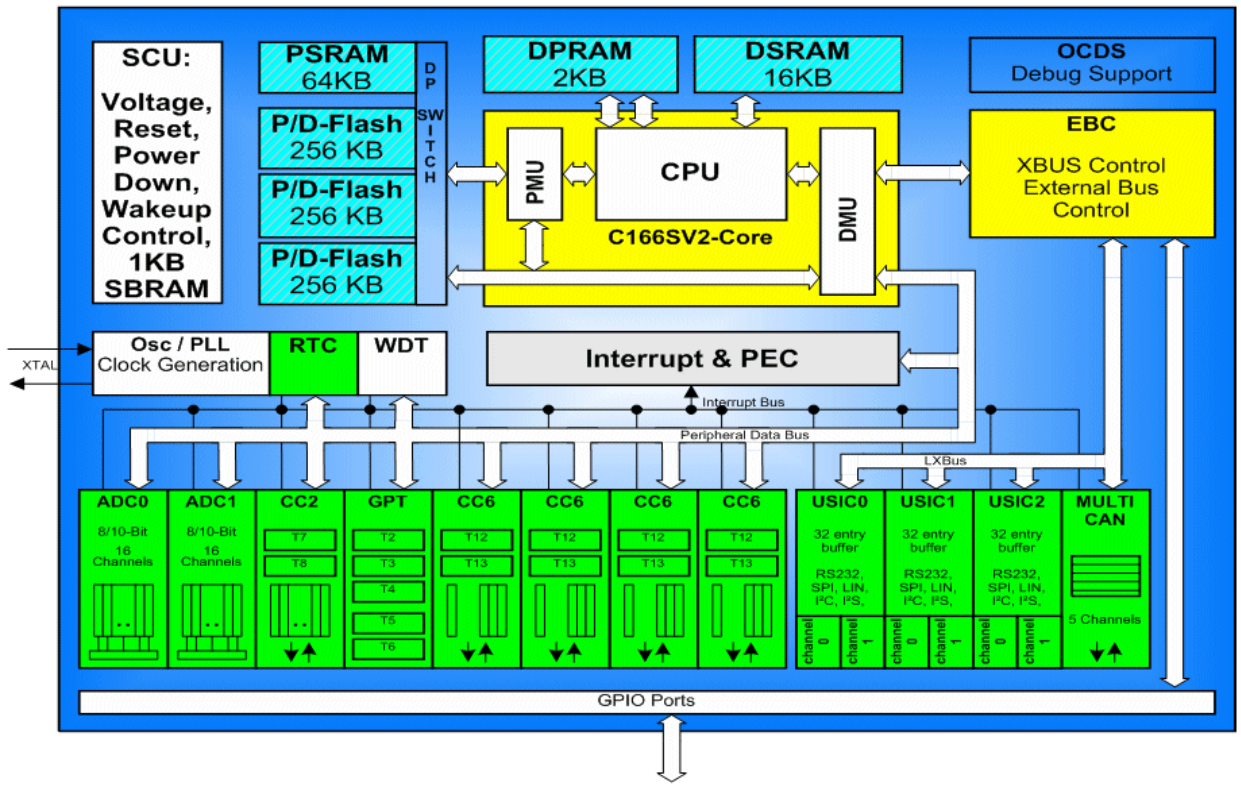
Block Diagram (Source: XE164 UConnect Manual)



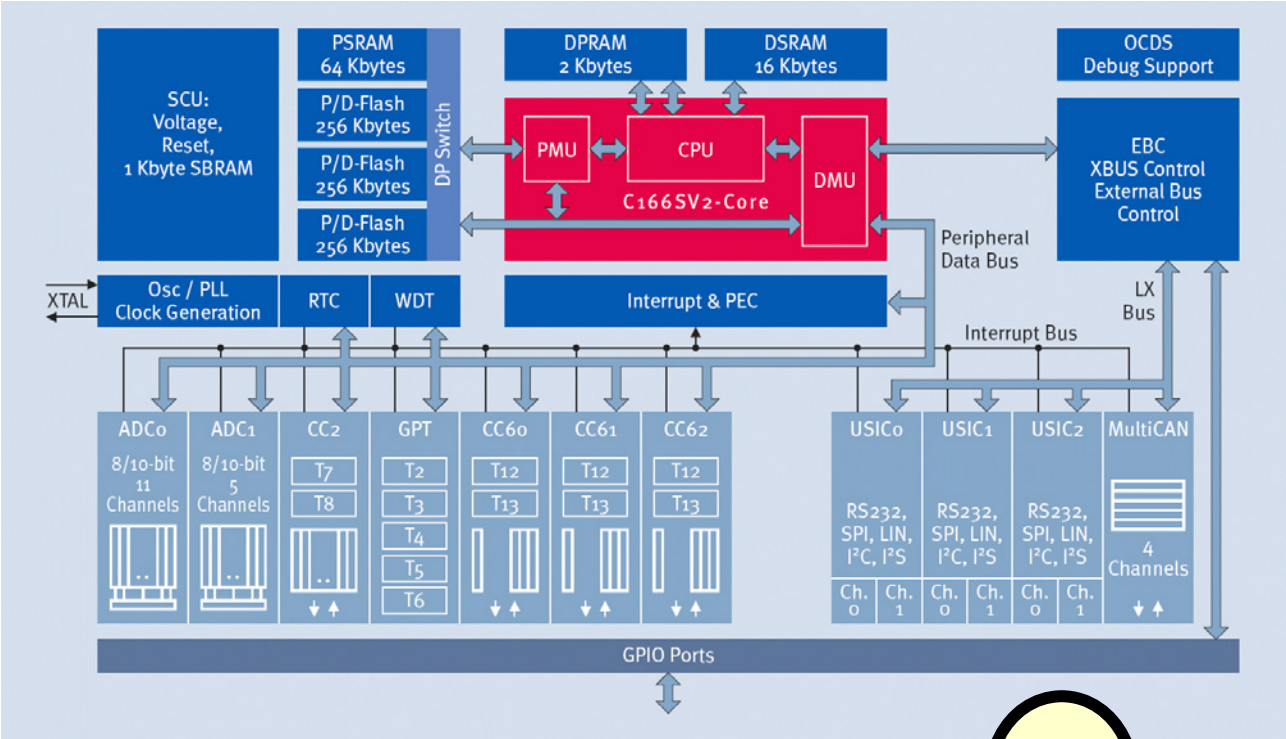




SAF-XE167F-96F66L Block Diagram (Source: Product Marketing)

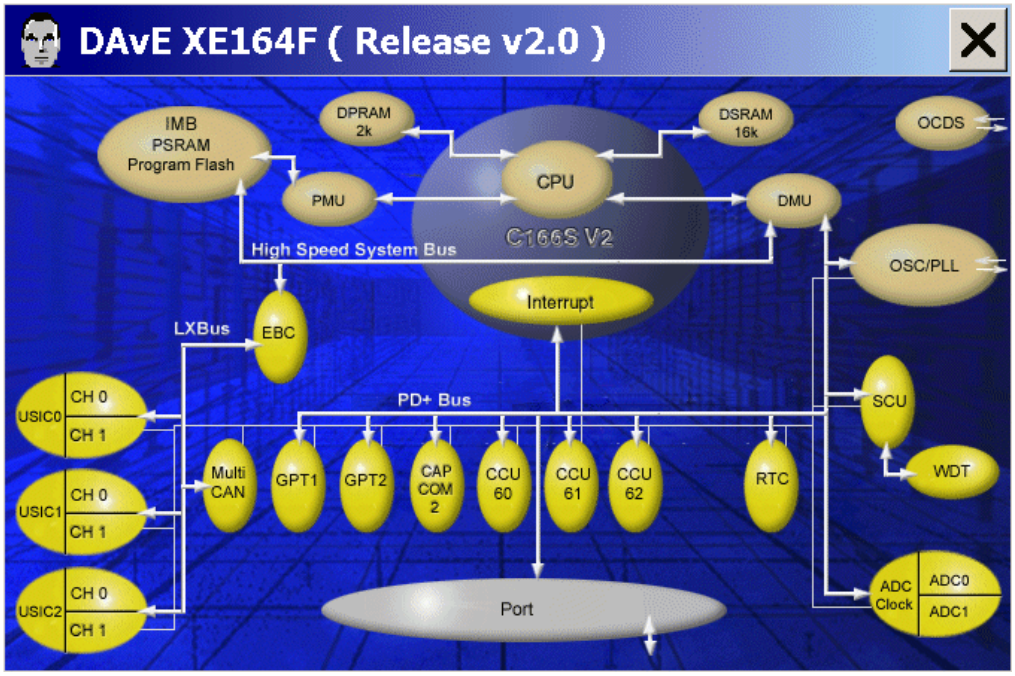


SAF-XE164F-96F66L Block Diagram (Source: Product Brief)

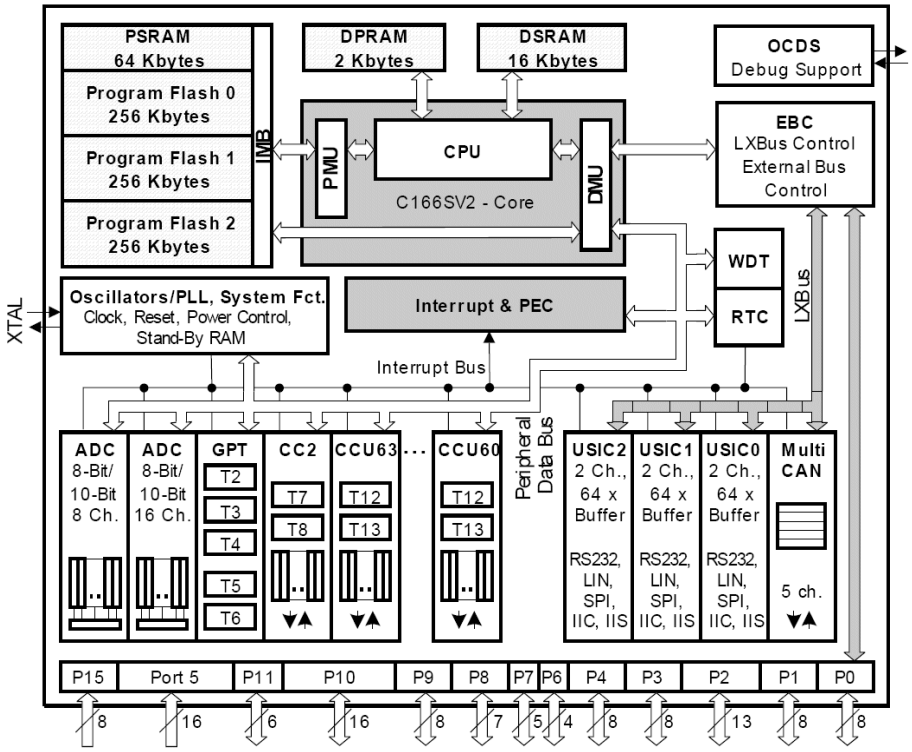


Note:
The XE164 microcontroller is a derivative of the XE167 microcontroller!

SAF-XE164F-96F66L Block Diagram (Source: DAvE)

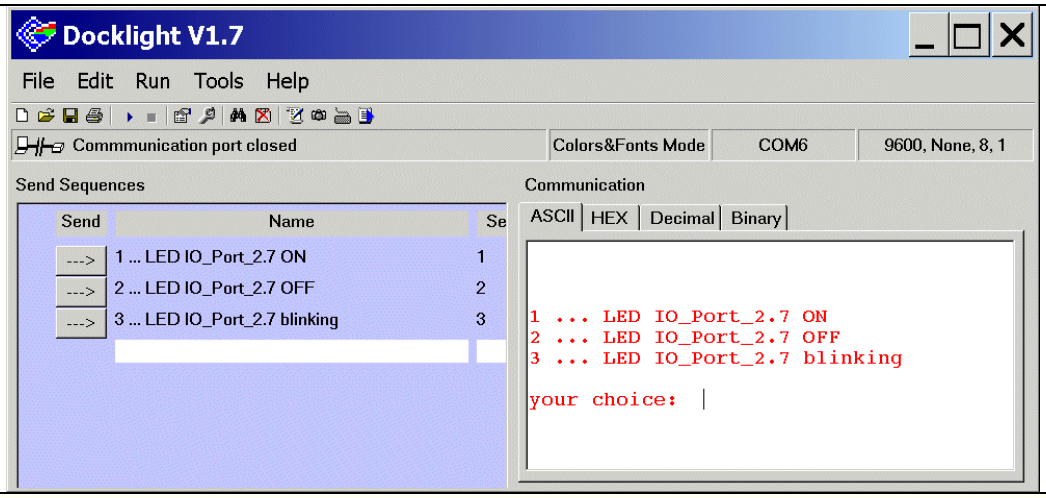


XE16x Block Diagram (Source: User's Manual)



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:

Your program:	
Chapter/ Step	***** Recipes *****
1.)	**DAS Installation + Connecting the UConnect-CAN XE164**
2.)	**DAvE (program generator)** **DAvE Installation (mothersystem) + DAvE Update Installation (XE16xx_Series.dip) for XE164**
3.)	**Using DAvE** **Microcontroller initialization for your programming example**
4.)	**Using the TASKING Development Tools (C-Compiler)** **Programming of your application (XE164) with Altium’s TASKING classic C166 toolset (v8.7r2)**
5.)	**Running your first programming example**

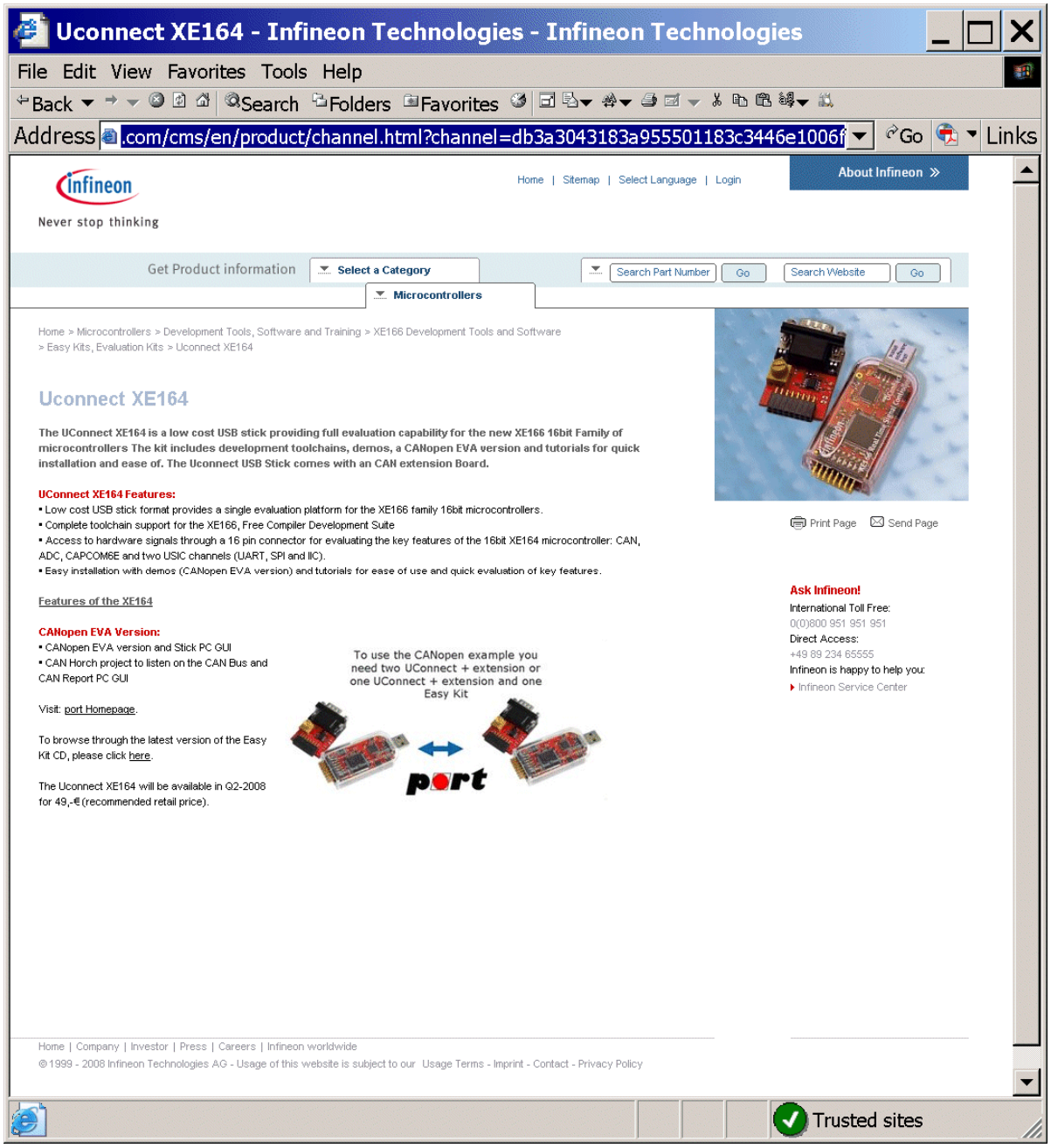
Feedback

6.)	**Feedback**

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



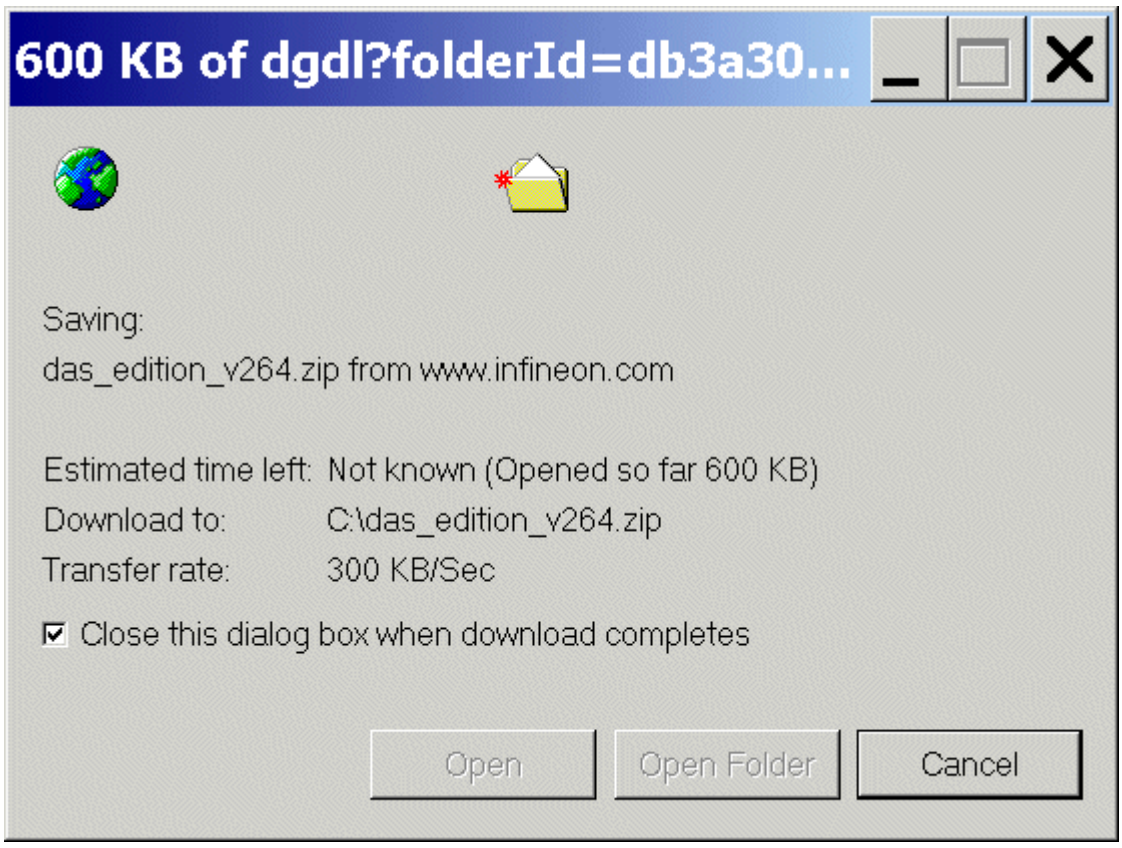
Note:
For further information, please refer to the XE164 UConnect Manual, V.1.0 .
For further information, please refer to the XE164 UConnect Manual, V.1.1 .

Install the Infineon DAS (D e v i c e A c c e s s S e r v e r) Server:

Go to www.infineon.com/DAS:

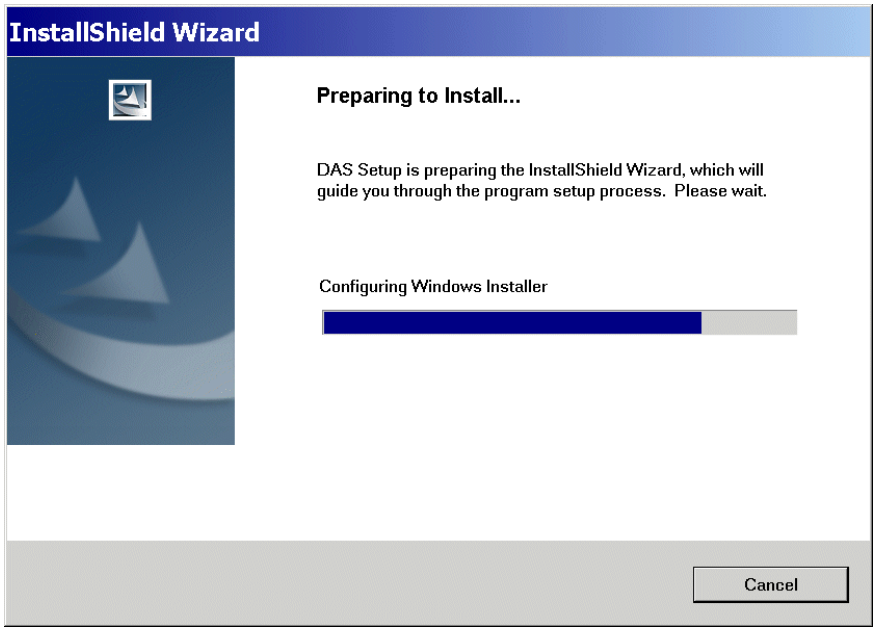
Note:
The DAS Server must be installed on your host computer!
The goal of the DAS software is to provide one single interface for all types of tools.
The USB Device driver communicates with the UConnect-CAN XE164 when connected to the host computer.
The USB Device driver for the UConnect-CAN XE164 USB interface is included in the DAS software.
A virtual COM port driver is also included.

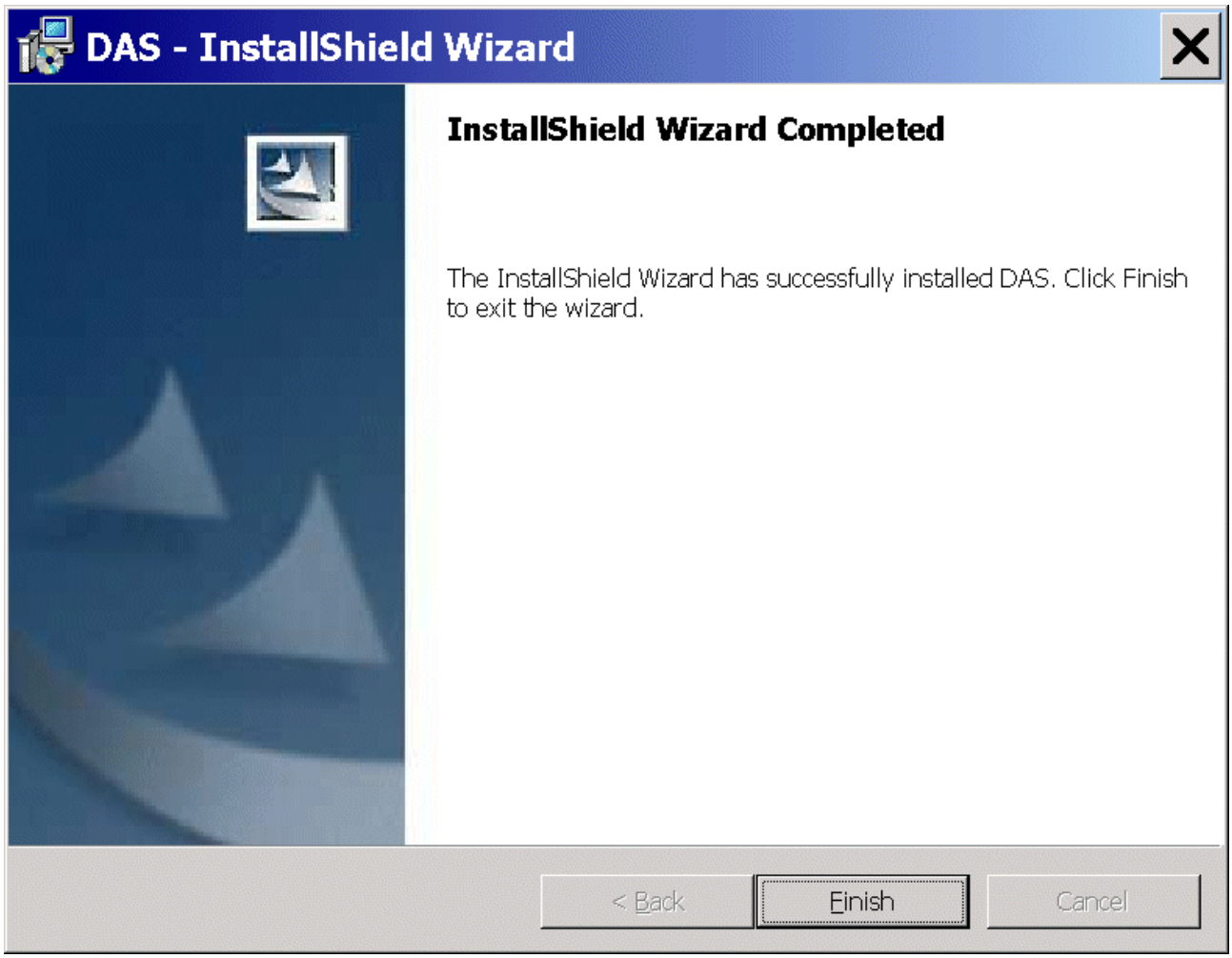
Download “The latest release version for download: DAS Edition v2.6.4”:



Unzip das_edition_v264.zip and

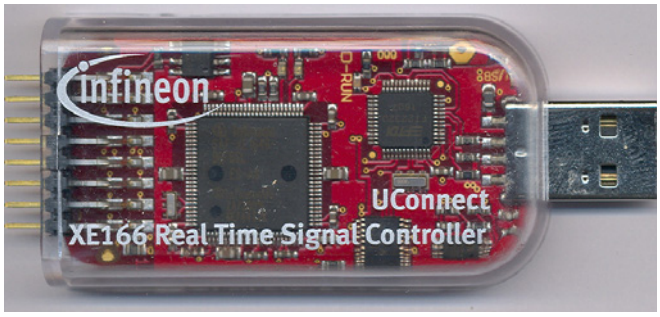
execute “DAS_v264_setup.exe” to install the DAS Server.

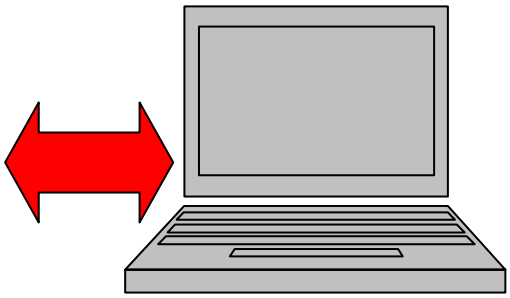




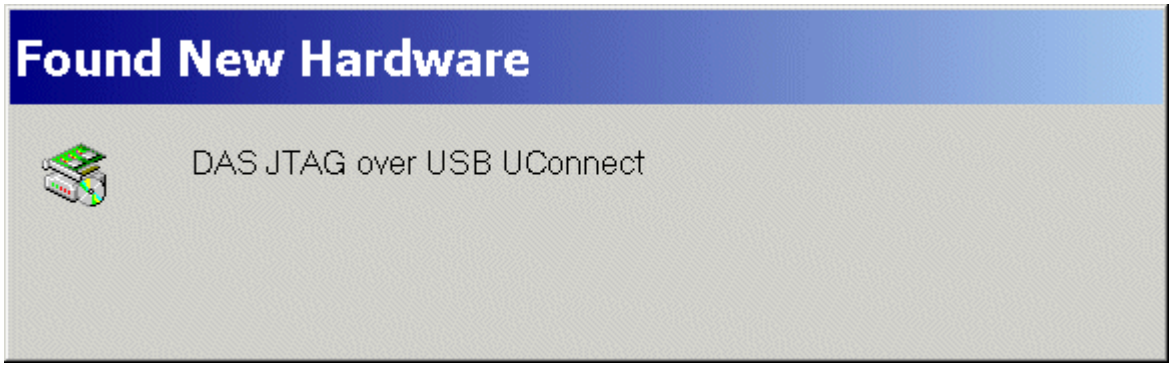
Click Finish

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.

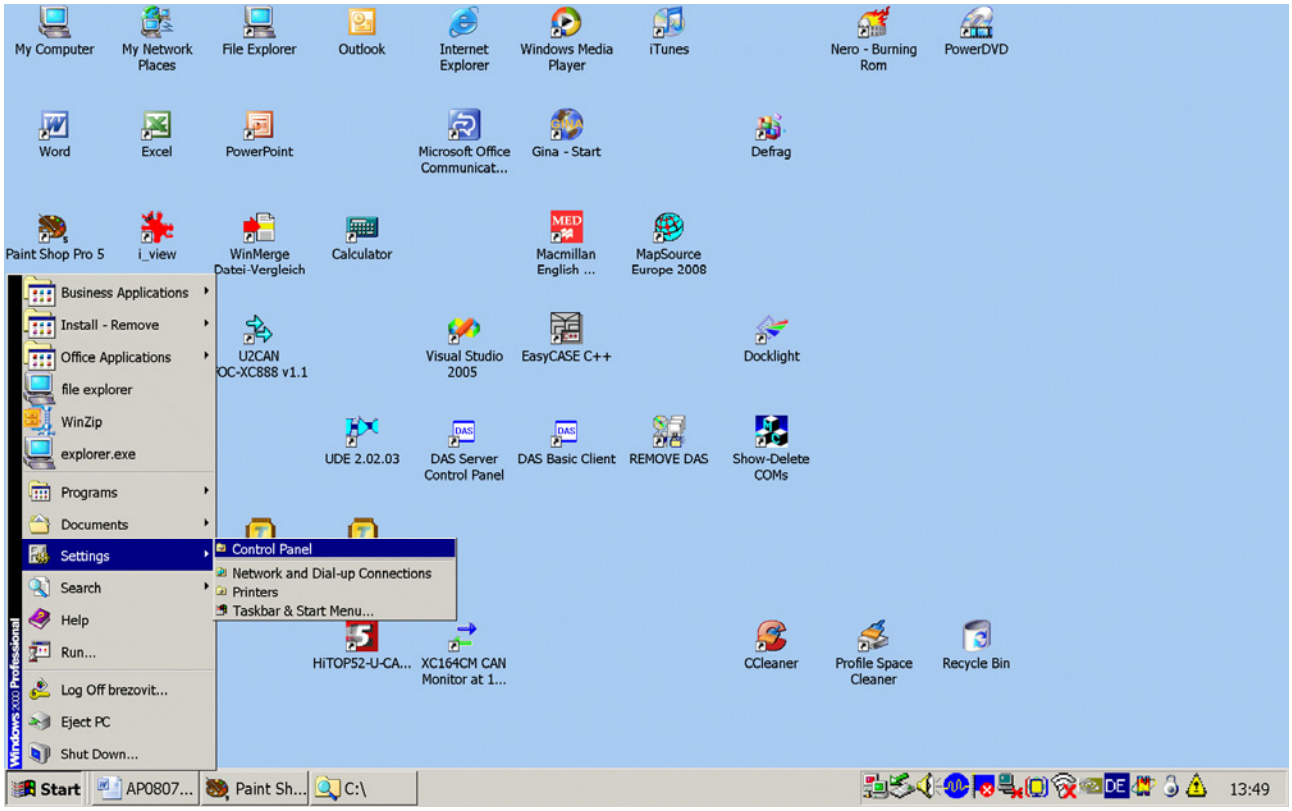


Note:
A USB driver is installed the first time while connecting the UConnect-CAN XE164 via USB to your host computer.

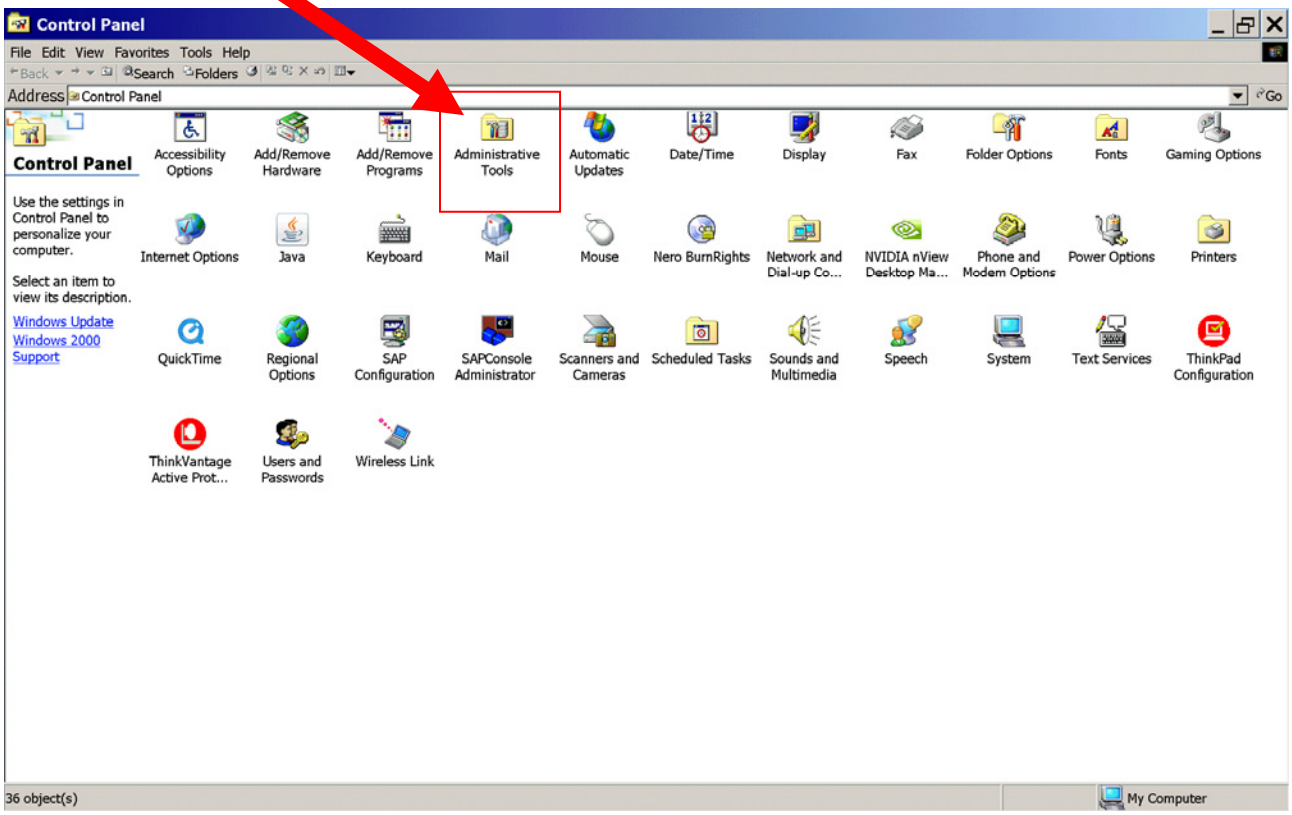
Note:
A default virtual COM Port is generated.

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:

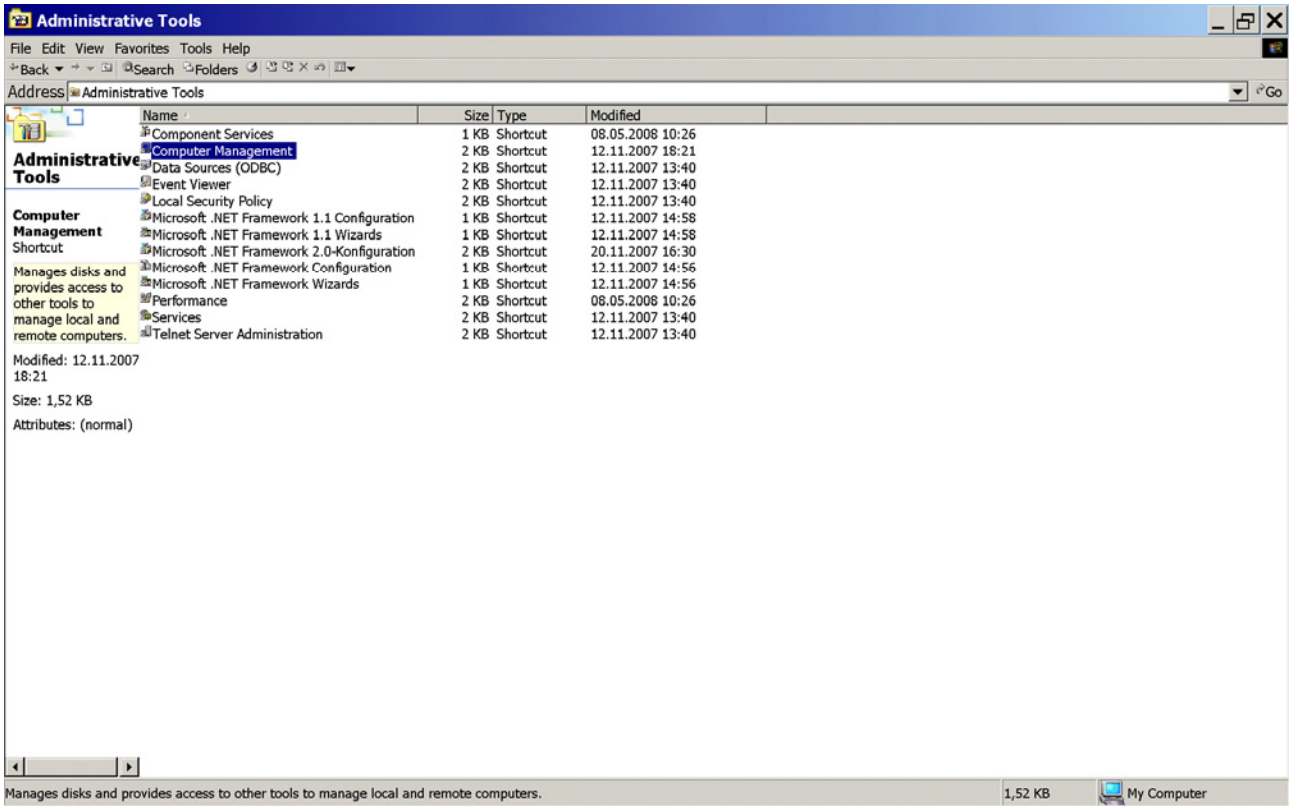
Start – Settings – Control Panel



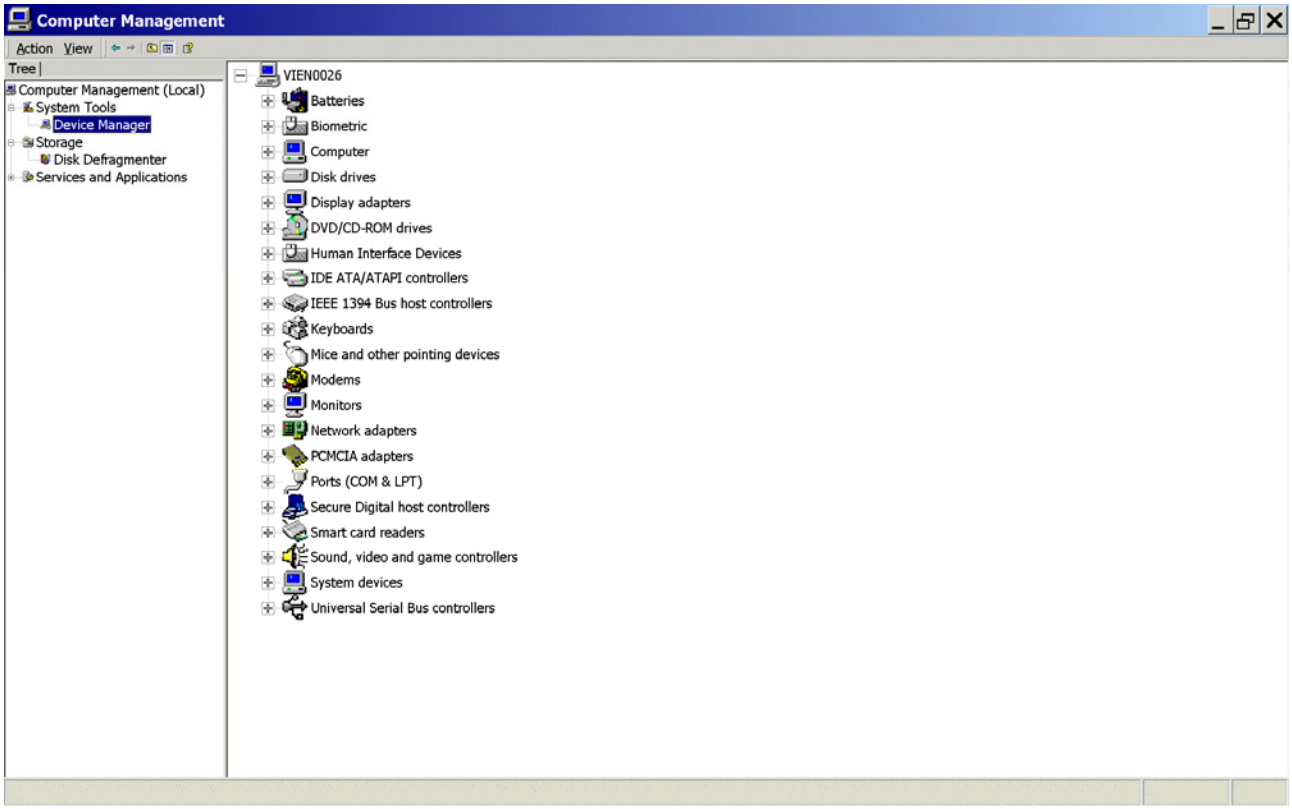
Double click: Administrative Tools



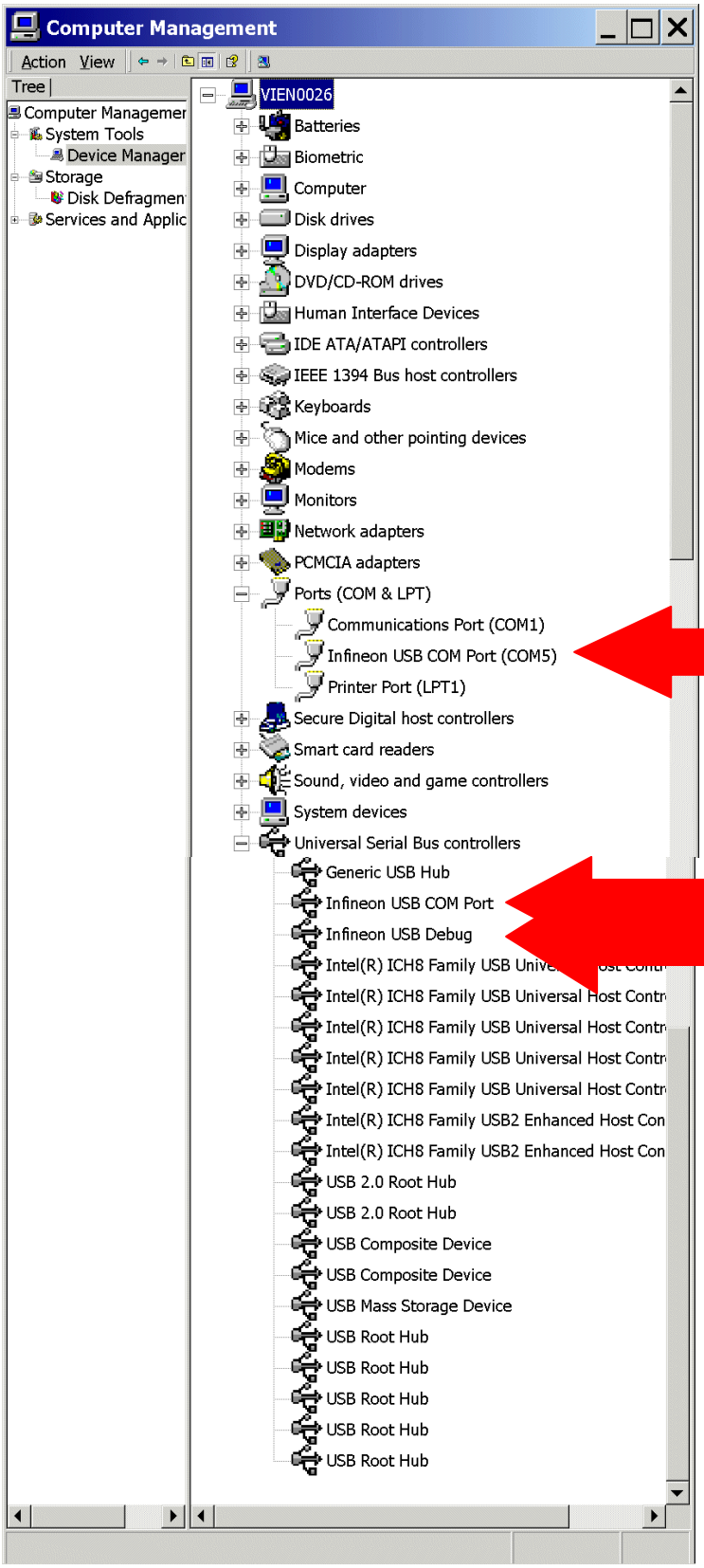
Double click: Computer Management



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
Tool Package			
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
DAvE home	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

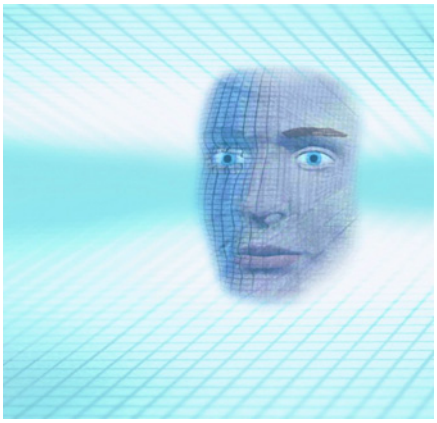
Development Tools

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.

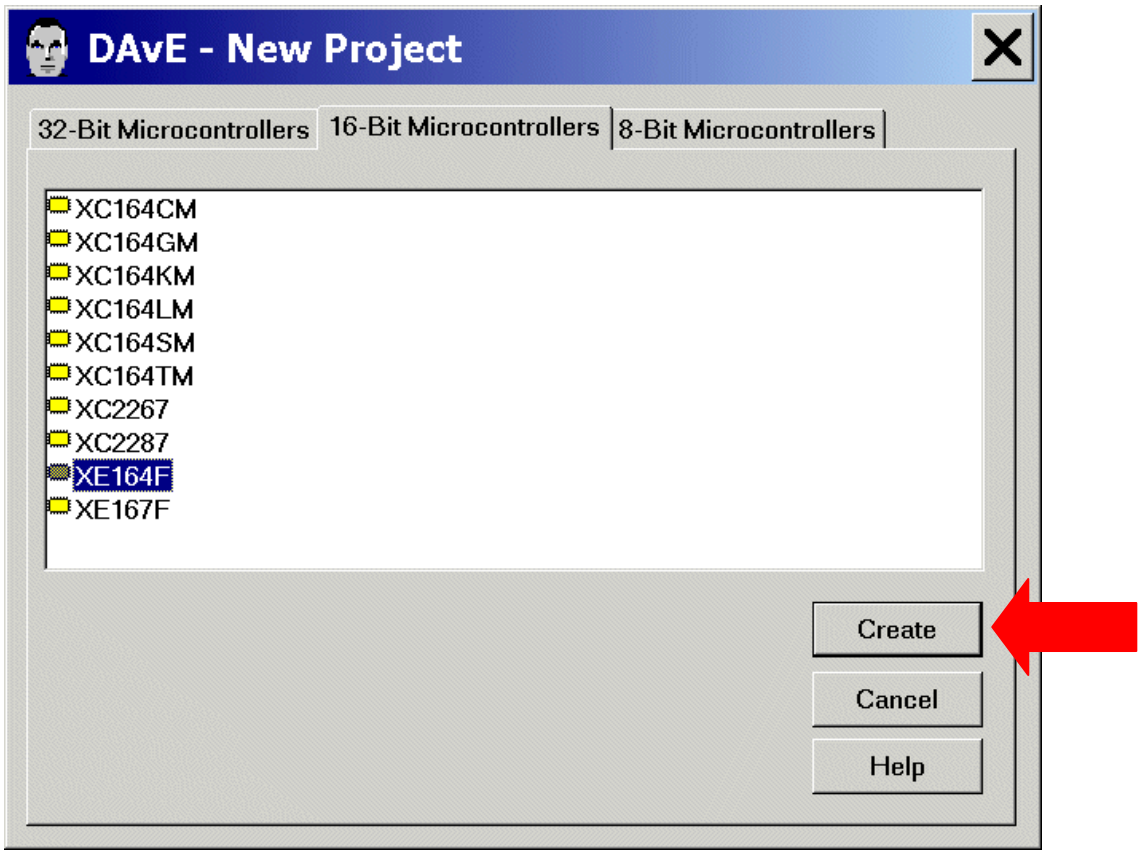
2.) Start DAvE - (click)
3.) View Setup Wizard Default: • Installation Forward> Select: • I want to install products from the DAvE's web site Forward> Select: C:\DAvE\XE16x-2008-05-29 Forward> Select: Available Products click ✔ XE16xx_Series Forward> Install End
4.) DAvE is now ready to generate code for the XE16x microcontrollers.

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



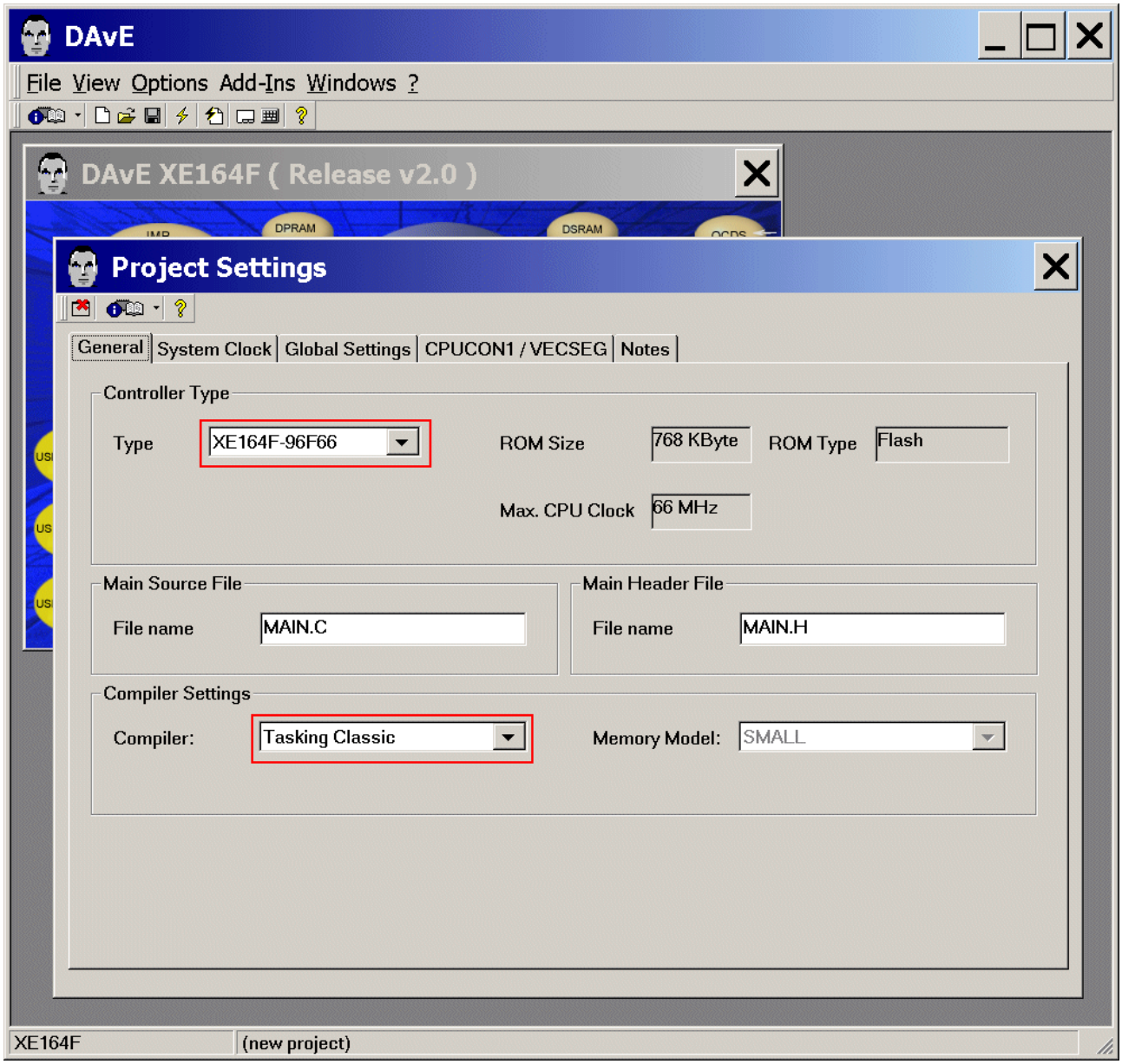
Start the program generator DAvE and select the XE164 microcontroller:

File
New
16-Bit Microcontrollers
Select XE164F
Create



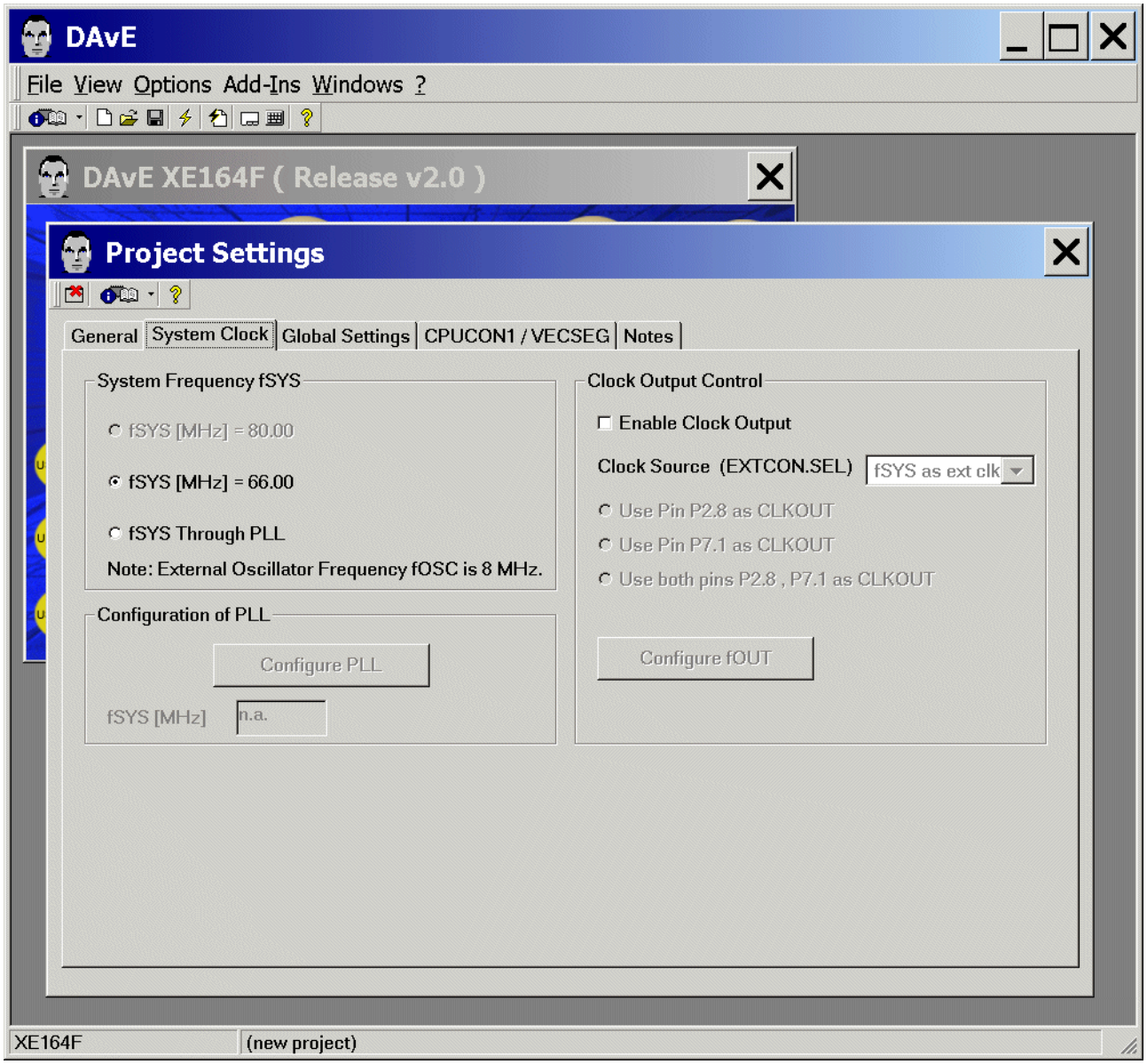
Choose the Project Settings as you can see in the following screenshots:

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



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

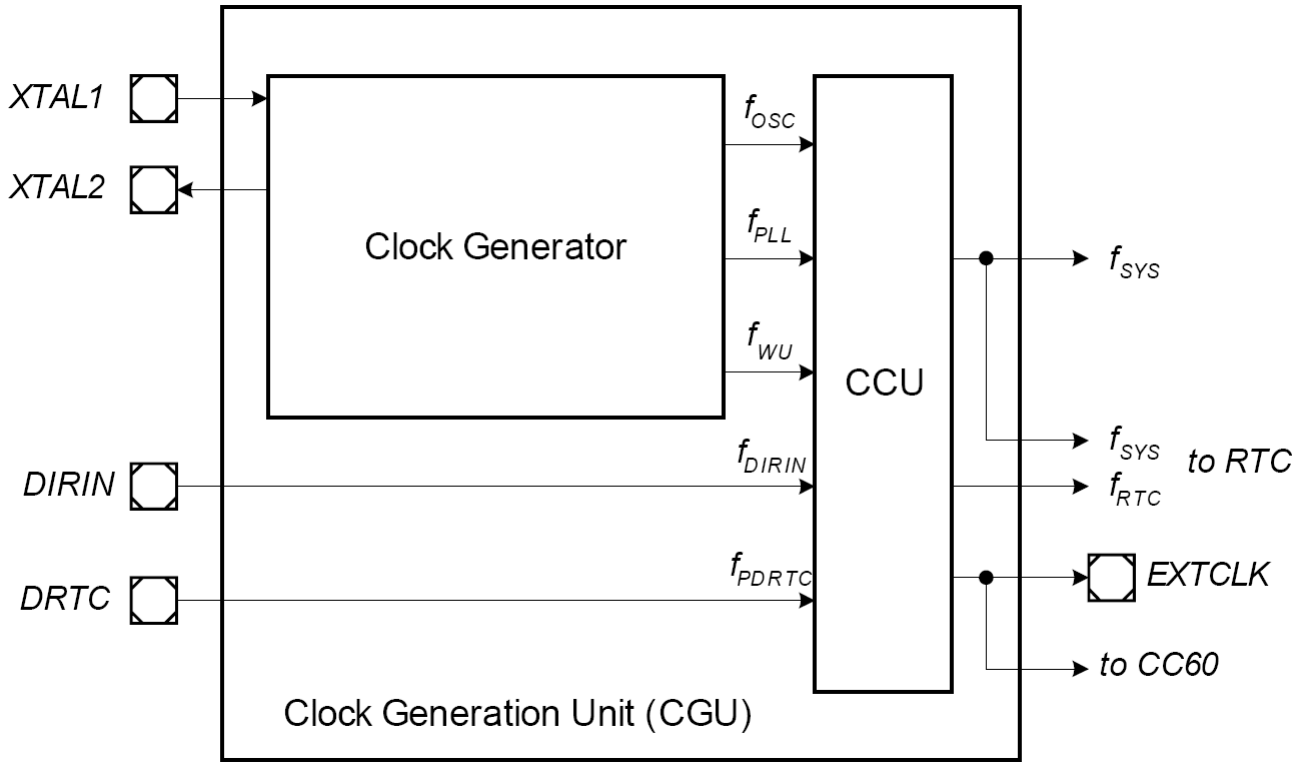
System Clock: (do nothing)



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):

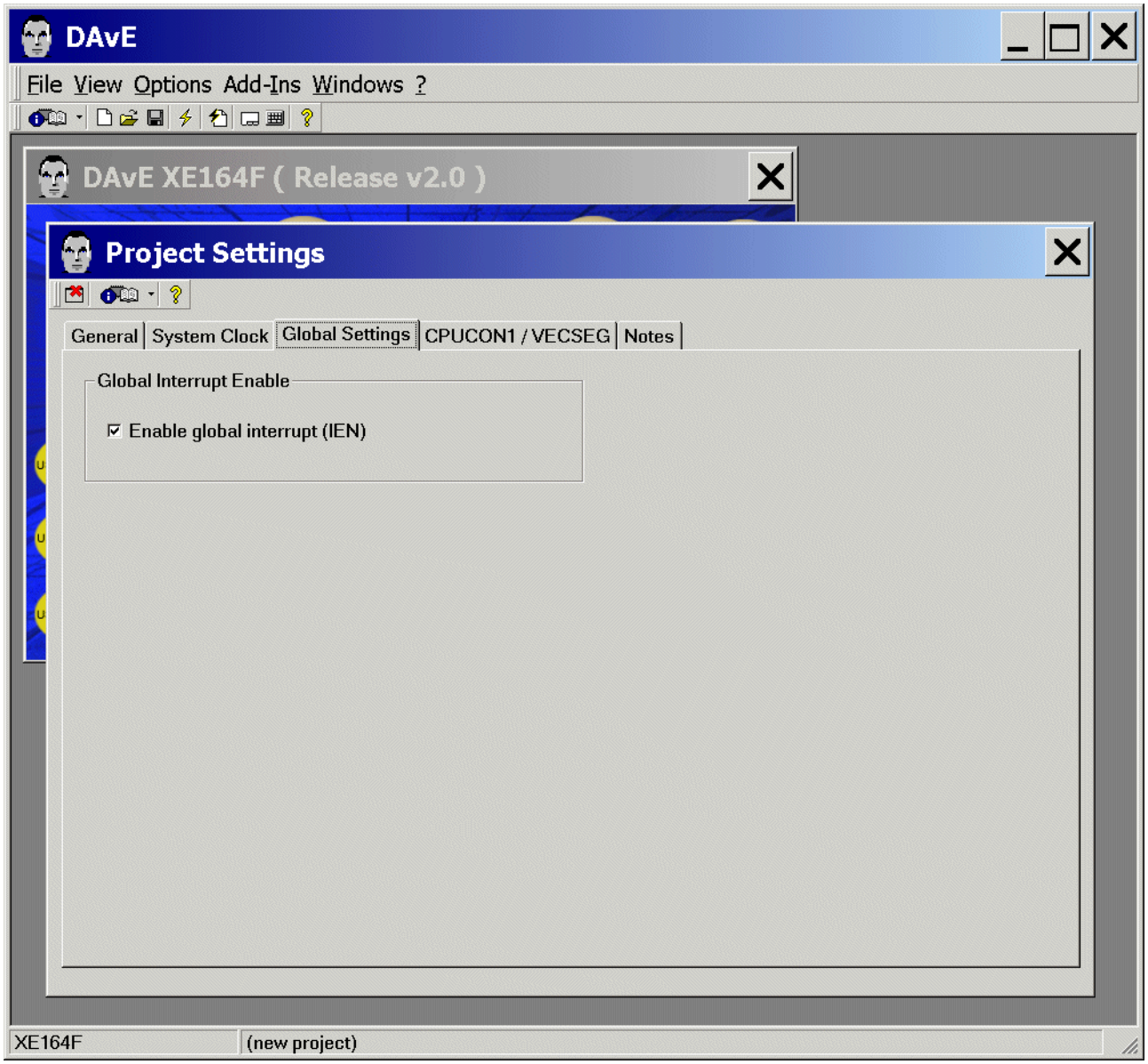
Clock Generation Unit (CGU) Block Diagram:



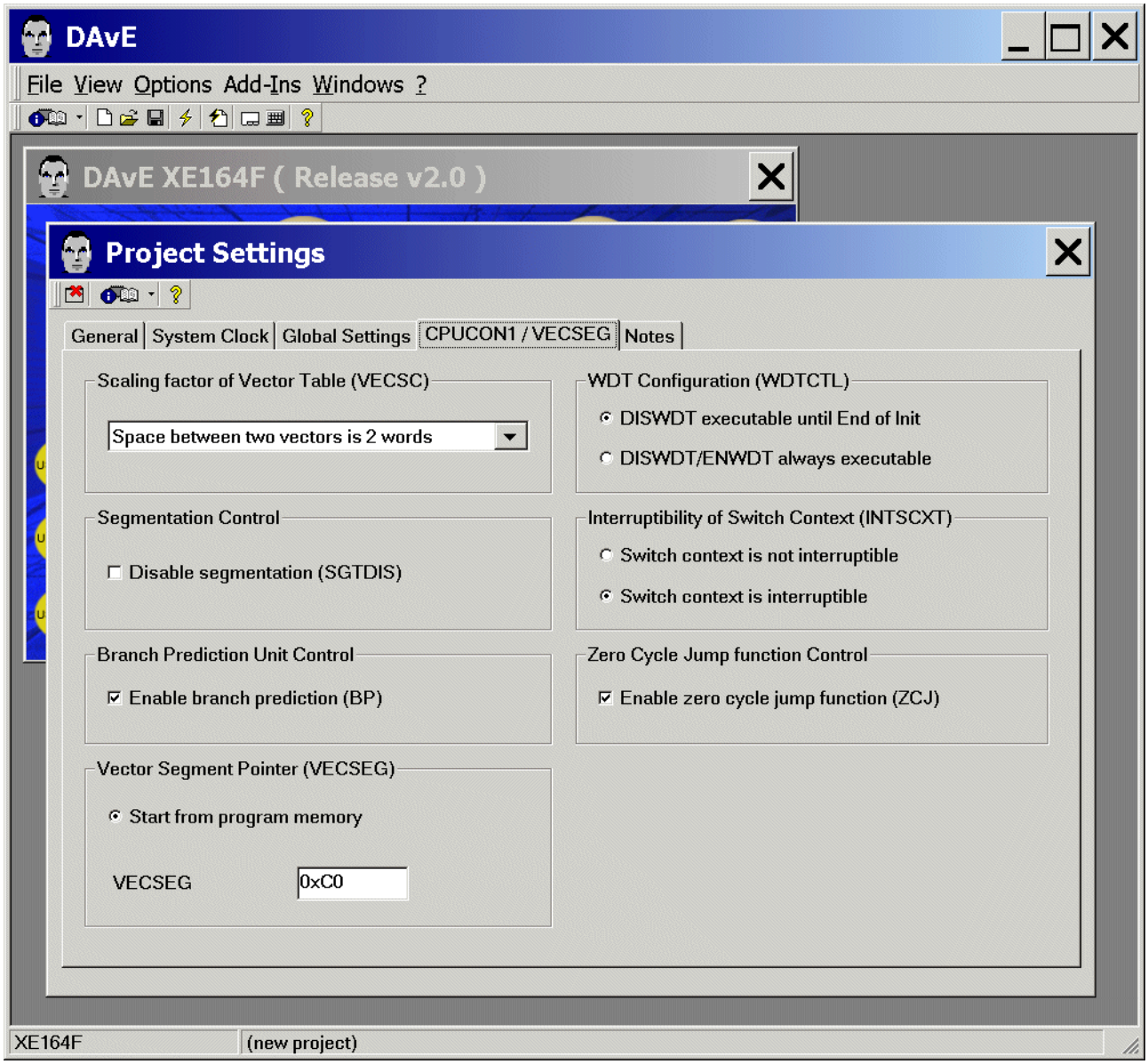
Note:
The CGU can convert a low-frequency external clock to a high-speed internal clock, or can create a high-speed internal clock without external input.

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}

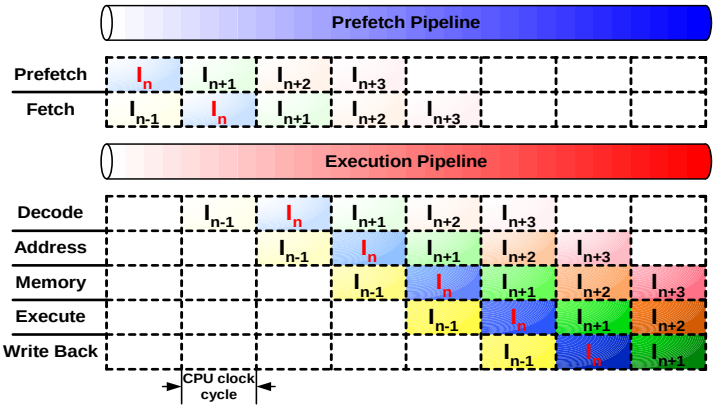
Global Settings: (do nothing. Do not change configuration)

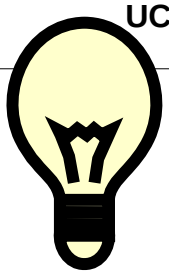


CPUCON1/VECSEG: (do nothing)

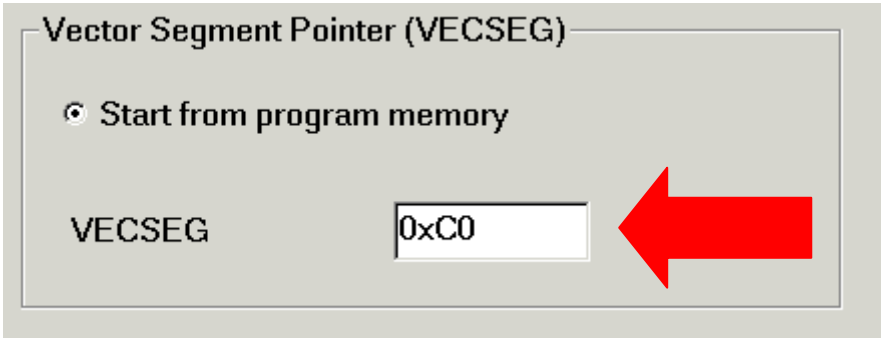


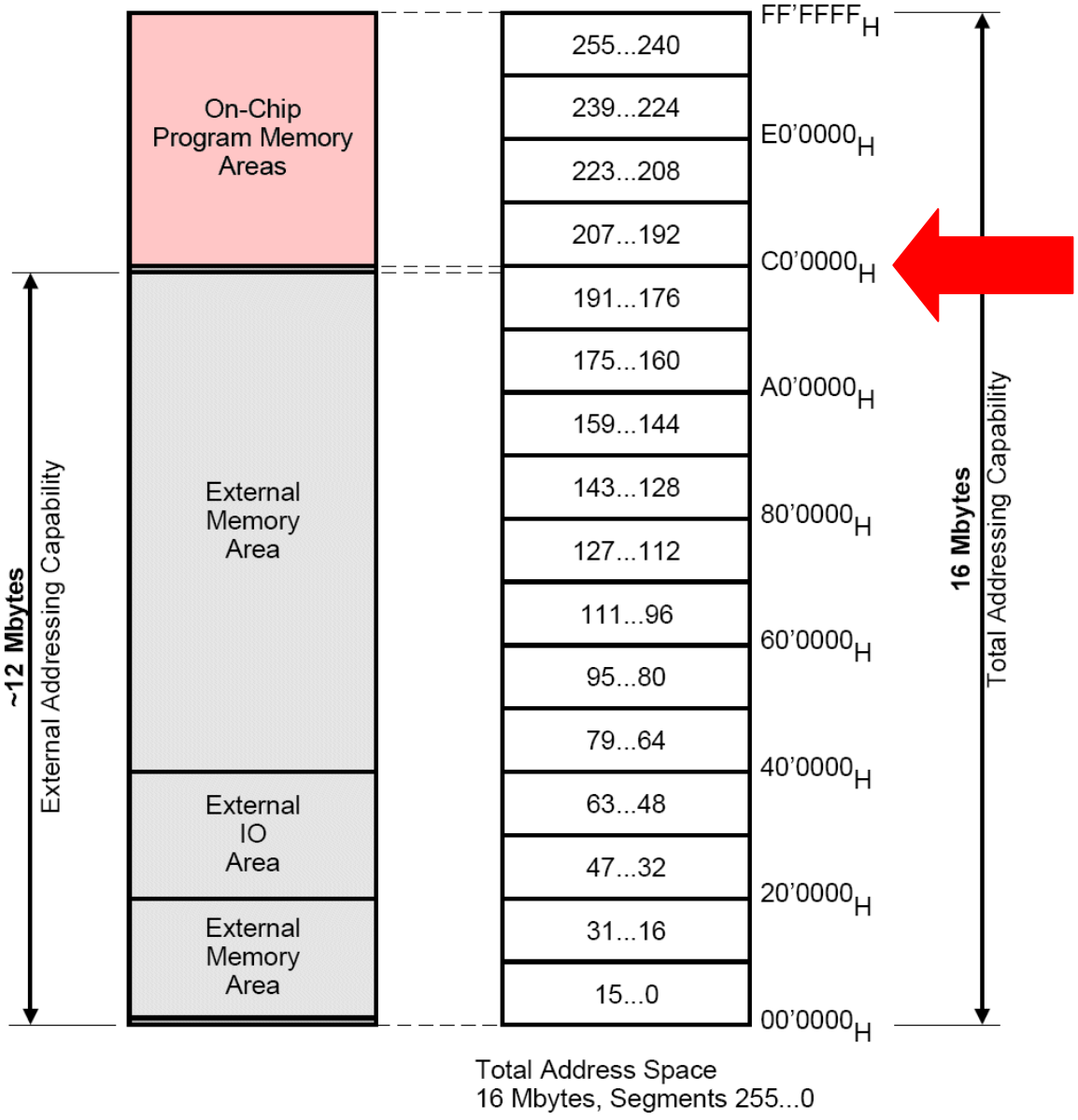
Note:
We should not change the pipeline behaviour.





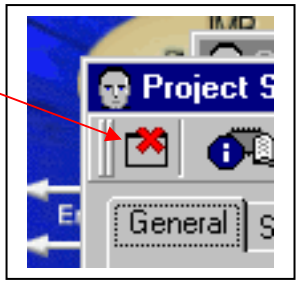
Additional information: Start from program memory (Source: User's Manual):





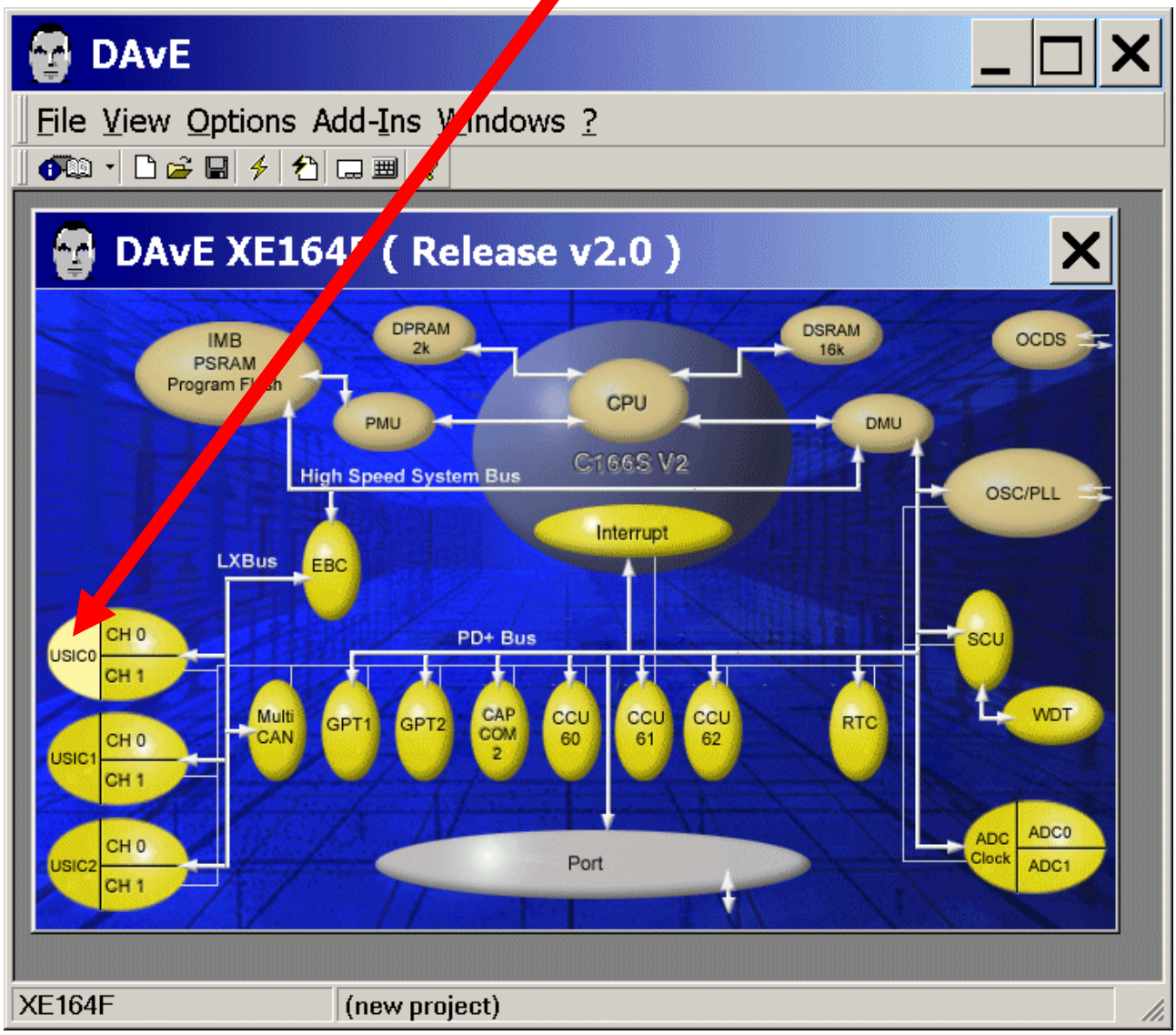
Notes: If you wish, you can insert your comments here.

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

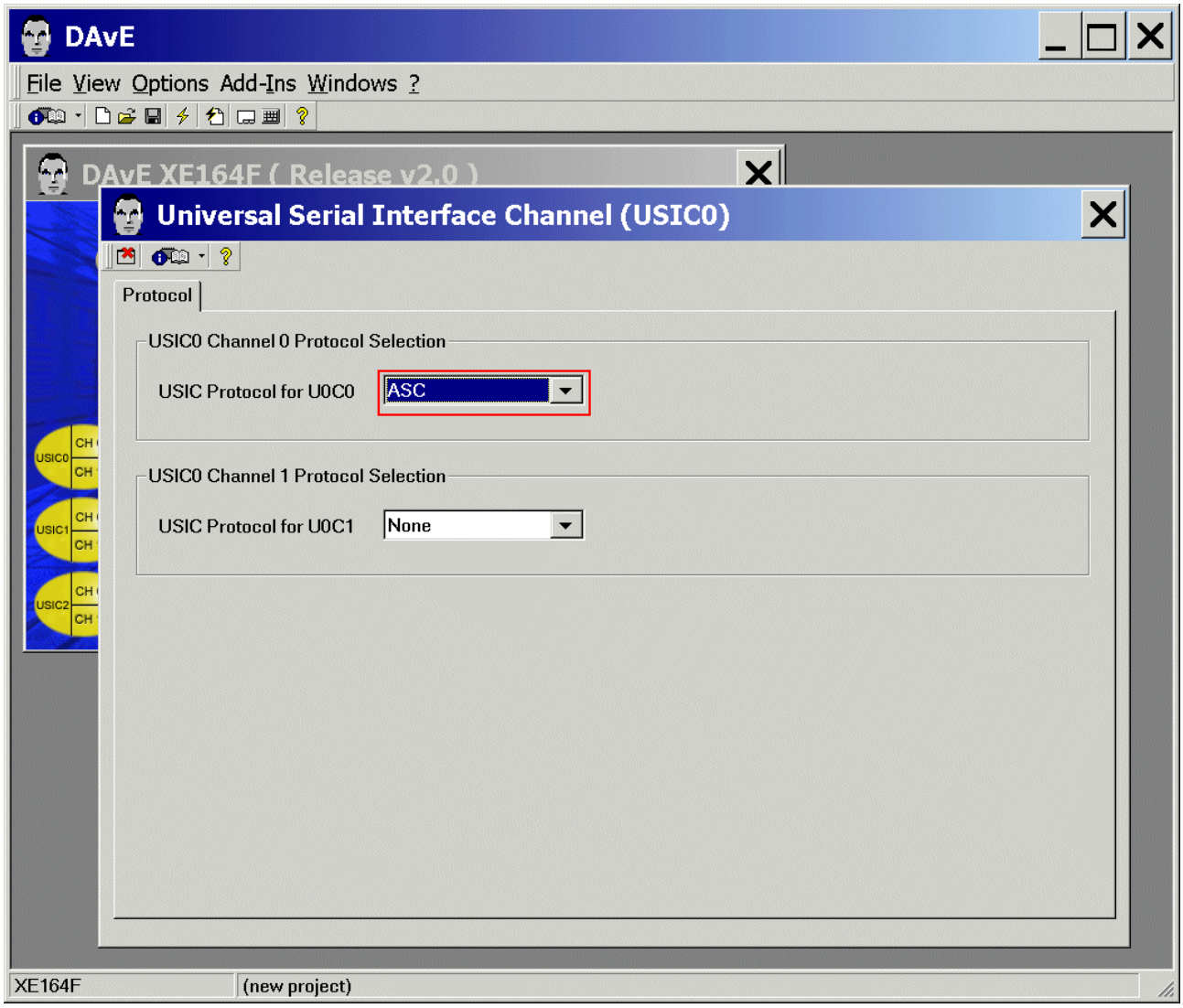


Configuration of the serial interface “ASC0” / UART / USIC0_CH0 / U0C0:

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



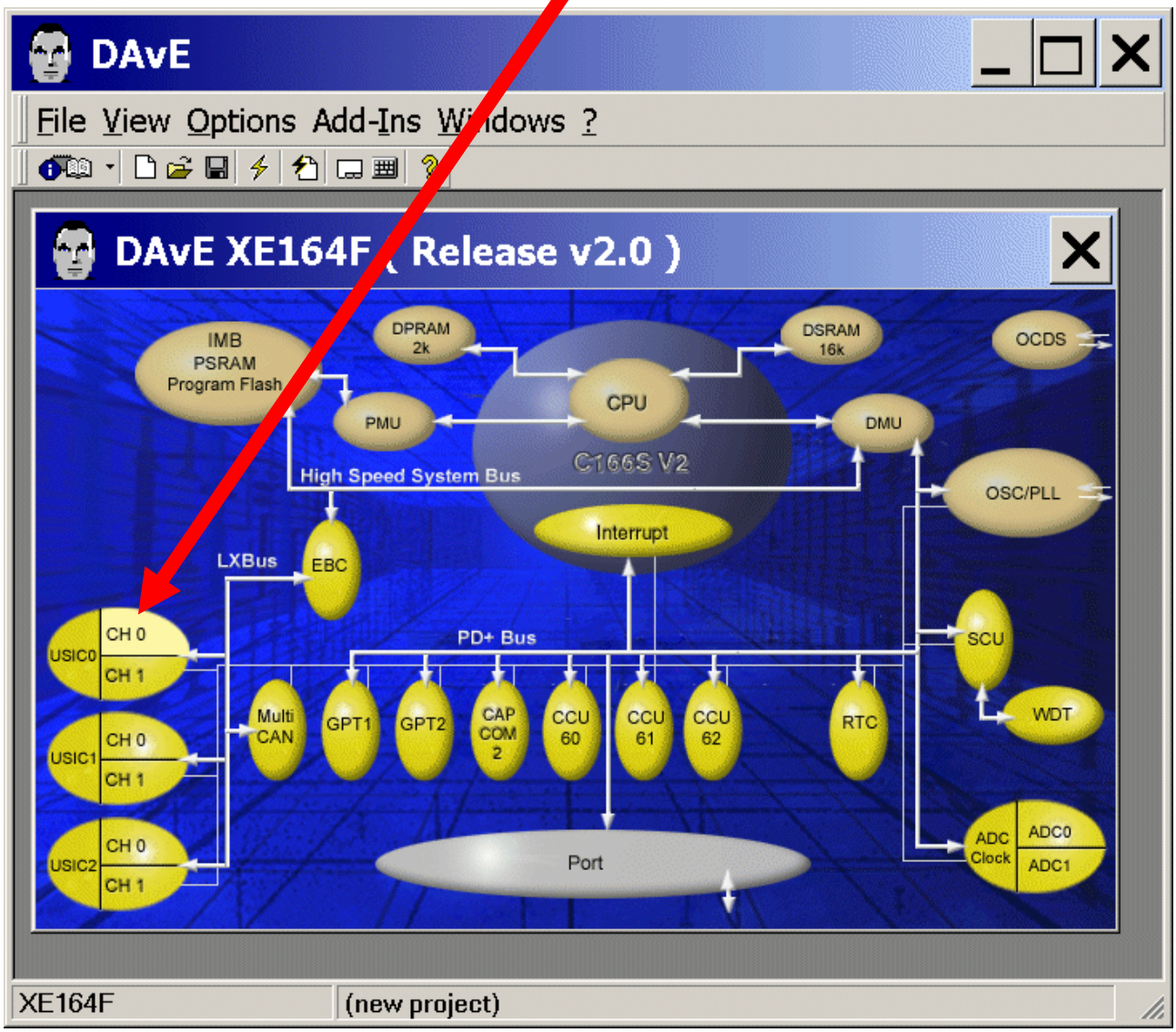
Protocol: USIC0 Channel 0 Protocol Selection: USIC Protocol for U0C0: select ASC



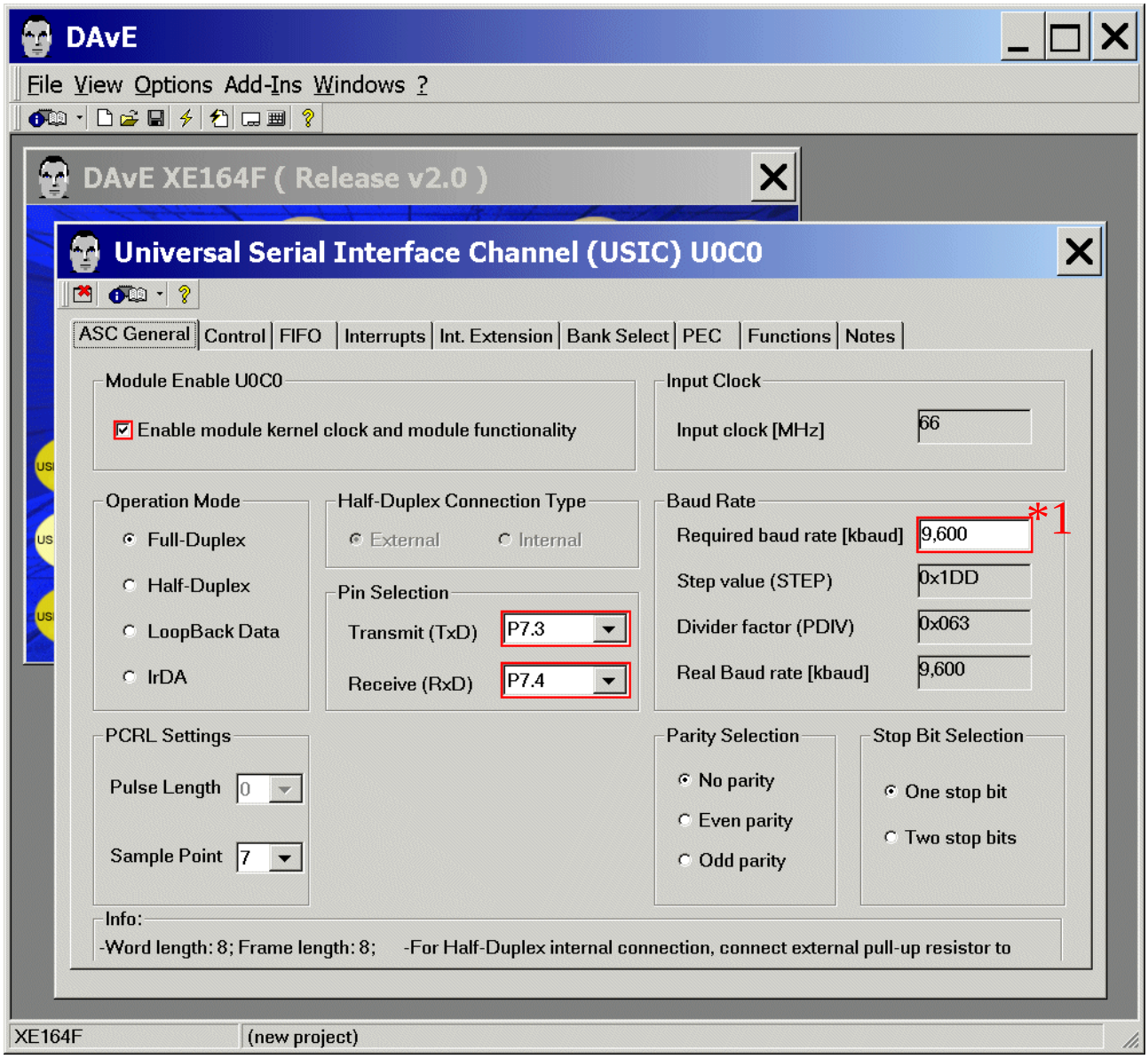
Exit and Save this dialog now by clicking the close button.

Configuration of the serial interface USIC0_CH0 / U0C0:

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



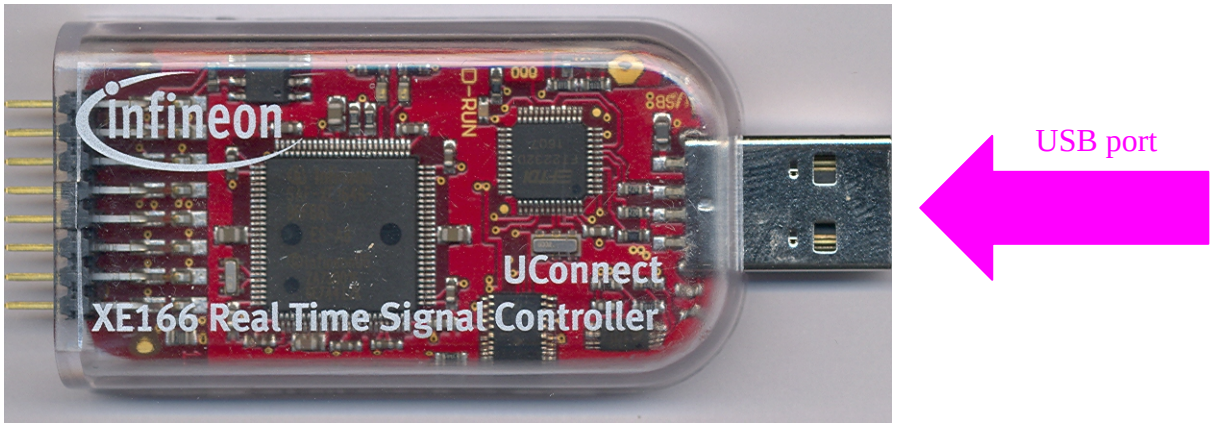
ASC General: Module Enable U0C0: click ☑ Enable module kernel clock and module functionality
ASC General: Pin Selection: Transmit (TxD): select P7.3
ASC General: Pin Selection: Receive (RxD): select P7.4
ASC General: Baud Rate: Required baud rate [kbaud]: insert 9,600 <ENTER>



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).



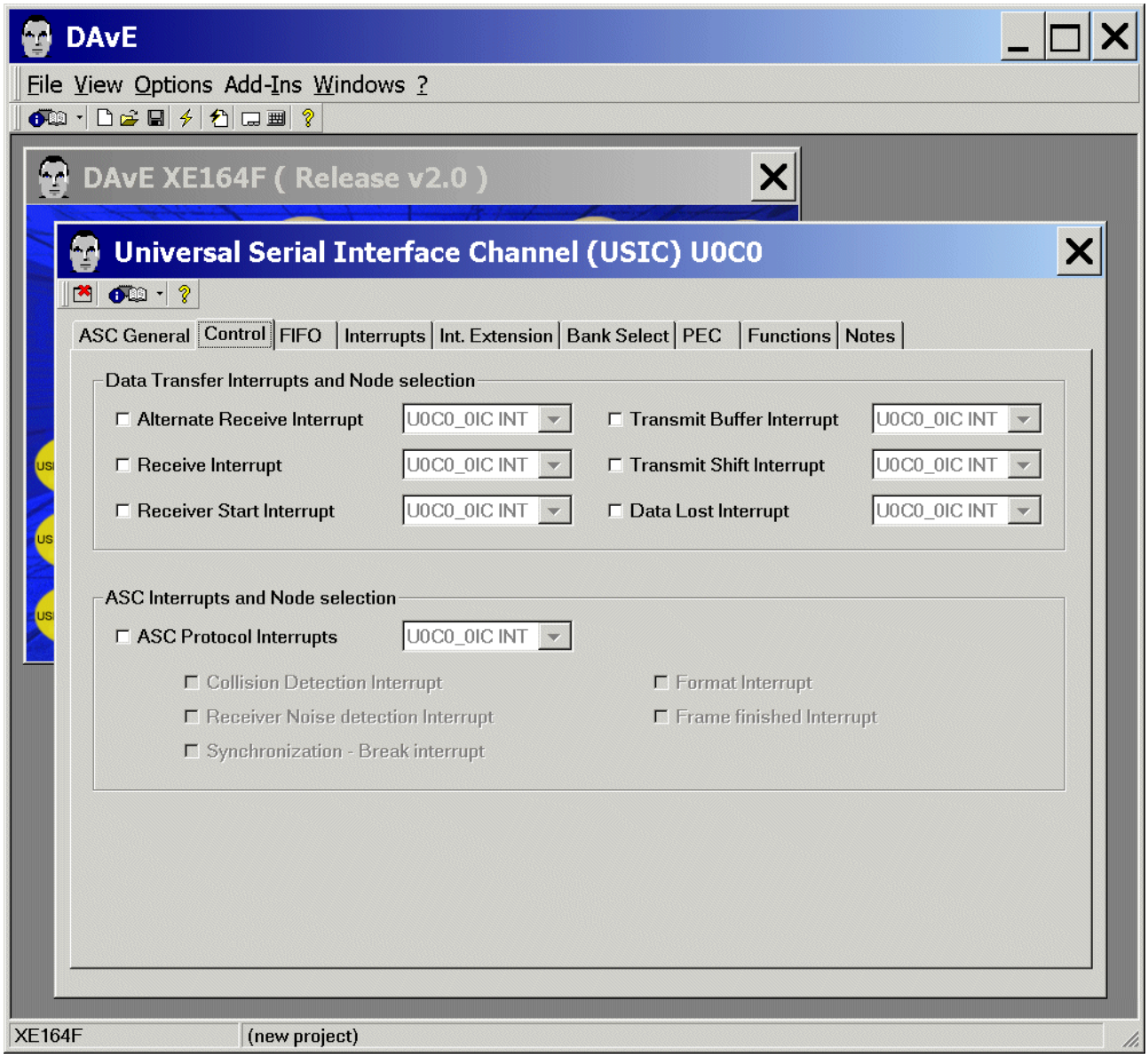
Additional information: Standard UART Pins (Source: User's Manual):

Table 10-10 Configuration Data for Bootstrap Loader Modes

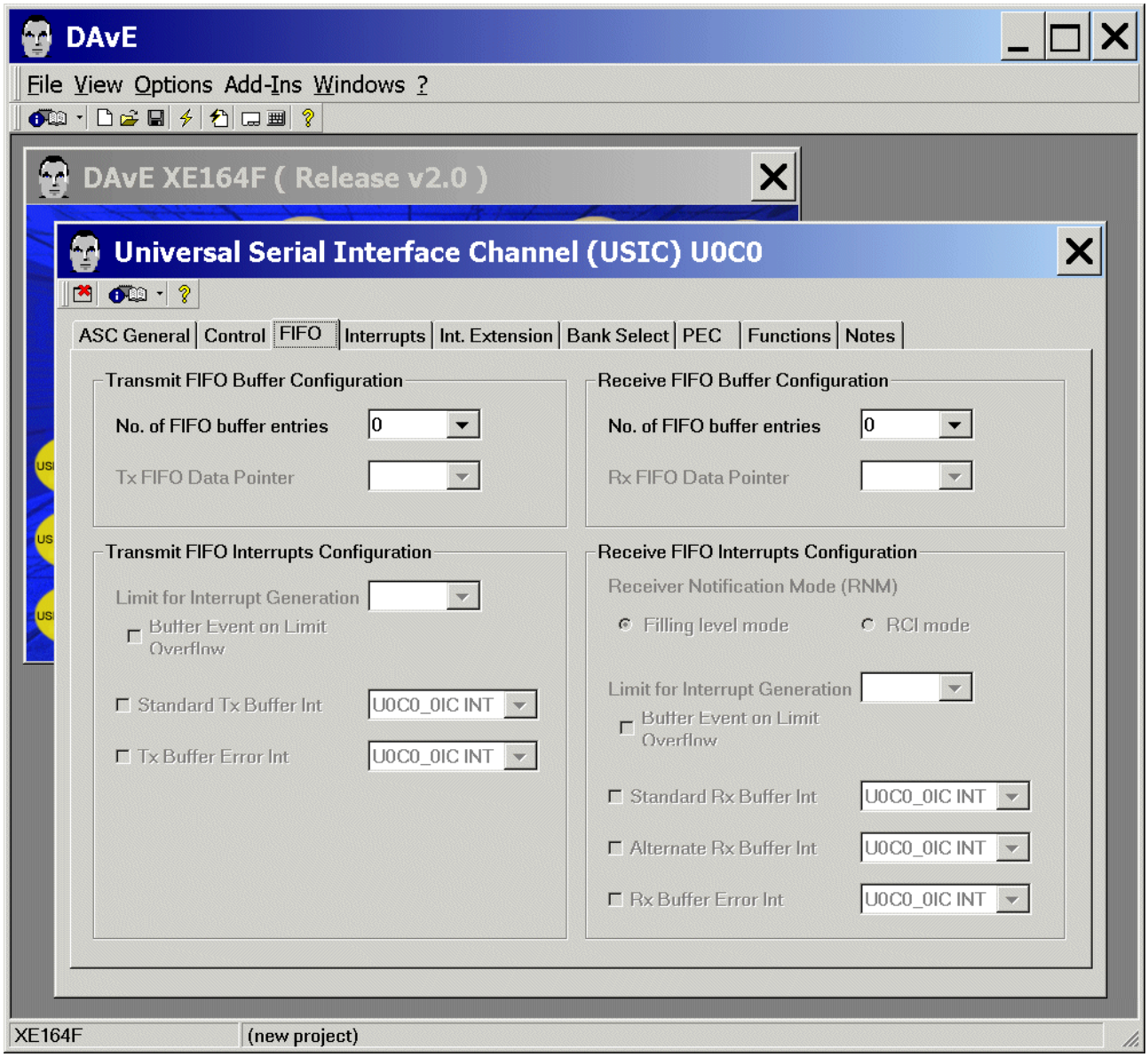
Bootstrap Loader Mode	Configuration on P10.3-0[1)]	Receive Line from Host	Transmit Line to Host	Transferred Data
Standard UART	$x110_B$	RxD = P7.4	TxD = P7.3	32 Bytes
Sync. Serial	1001_B	MRST = P2.4	MTSR = P2.3 SCLK = P2.5 SLS = P2.6	n Bytes; 1 … 65,280
MultiCAN	$x101_B$	RxDC0 = P2.6	TxDC0 = P2.5	8 × n Bytes

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

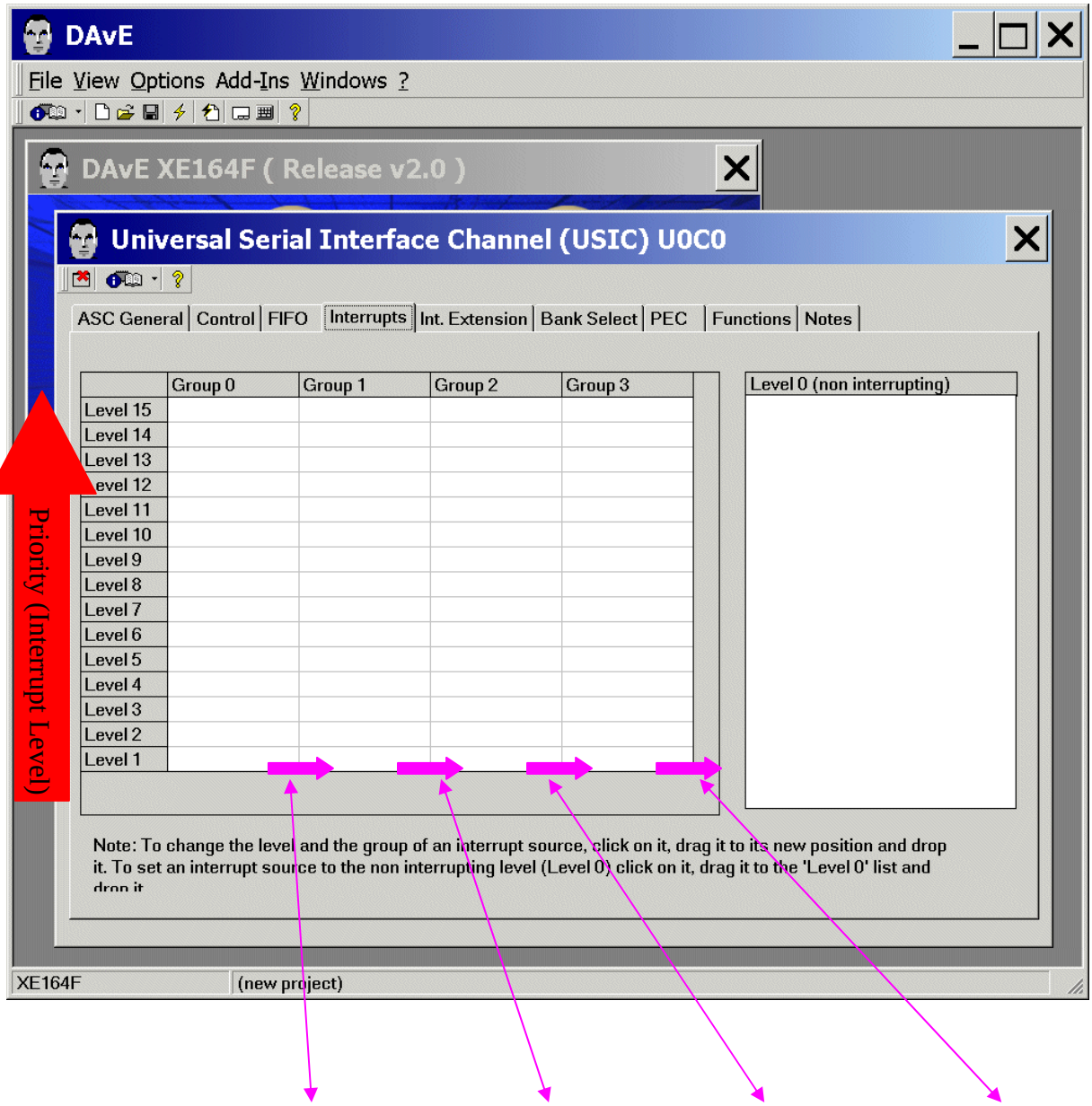
Control: (do nothing)



FIFO: (do nothing)



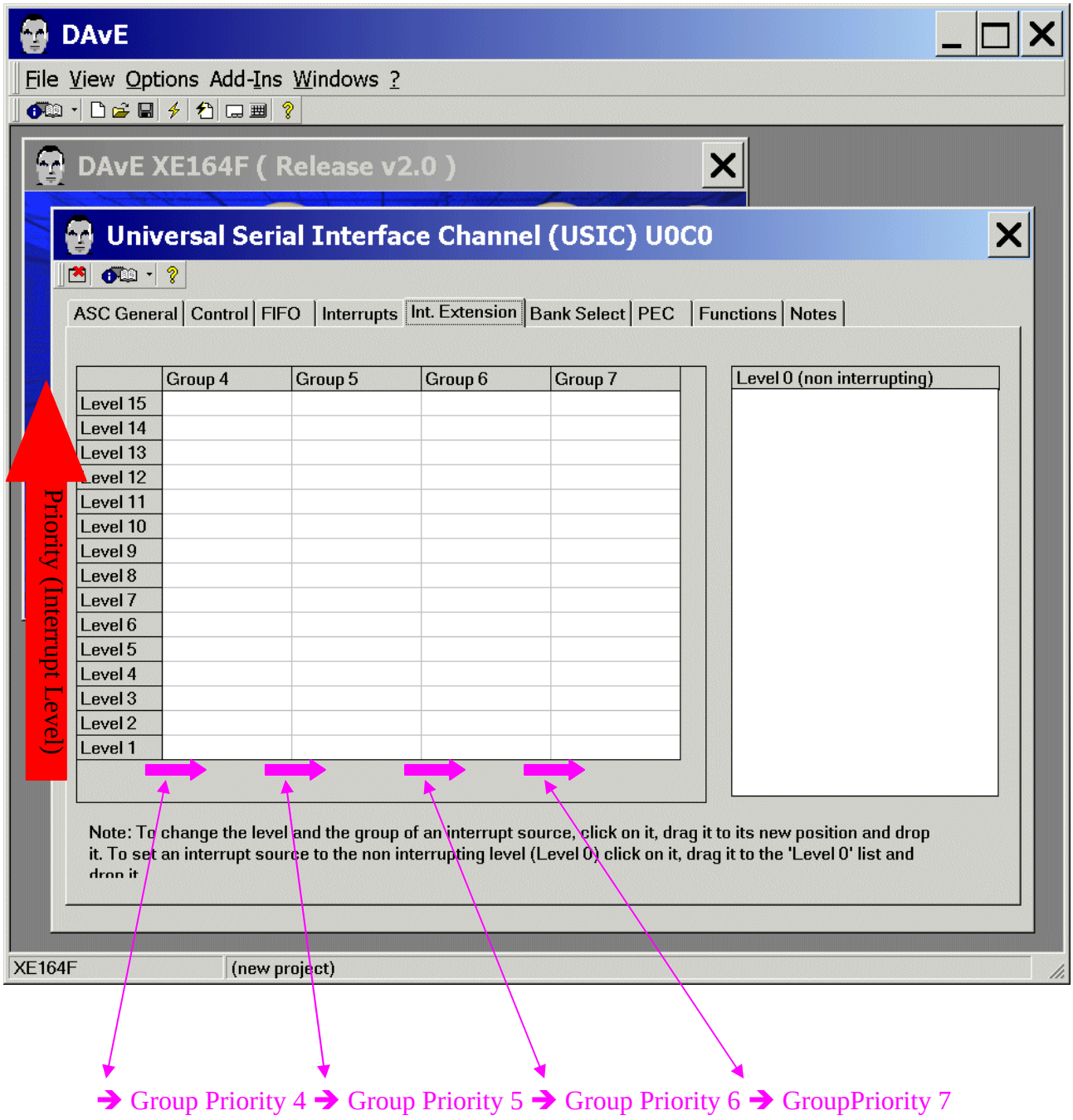
Interrupts: (do nothing)



Group Priority 0 ➔ Group Priority 1 ➔ Group Priority 2 ➔ GroupPriority 3 ➔

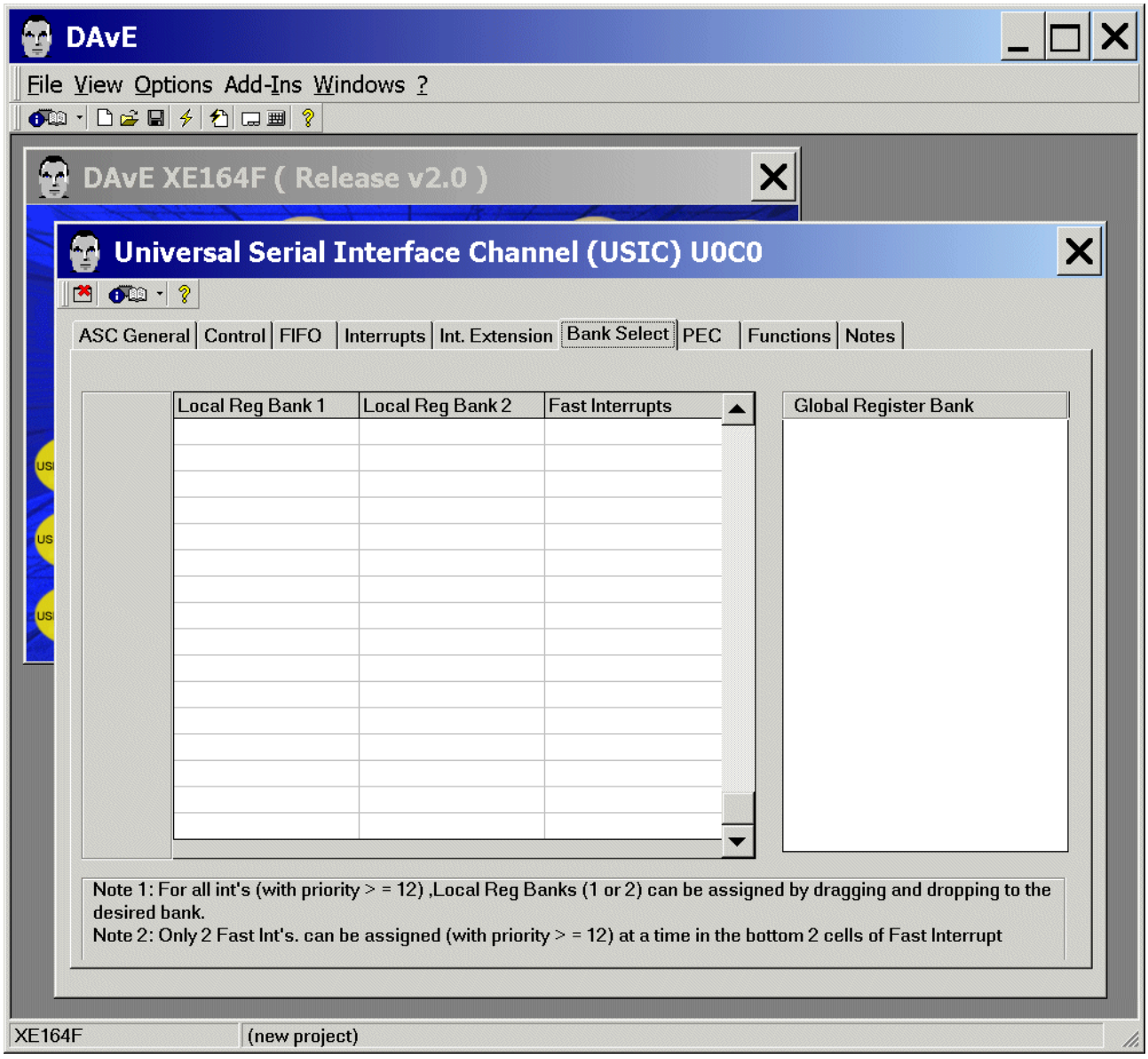
Note:
For the serial communication with a terminal program (e.g. Docklight, www.docklight.de) running on your host computer the myprintf function is used. The myprintf function uses Software-Polling-Mode therefore we do not need to configure any interrupts.

Int. Extension: (do nothing)

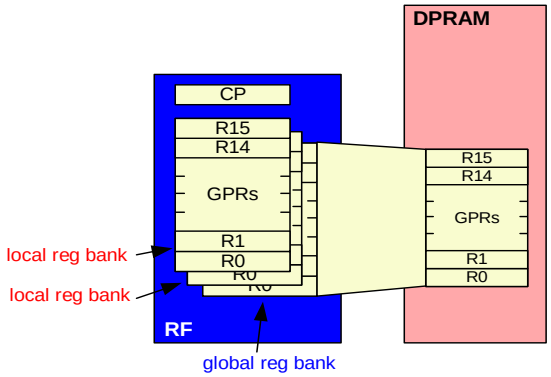


→ Group Priority 4 → Group Priority 5 → Group Priority 6 → GroupPriority 7

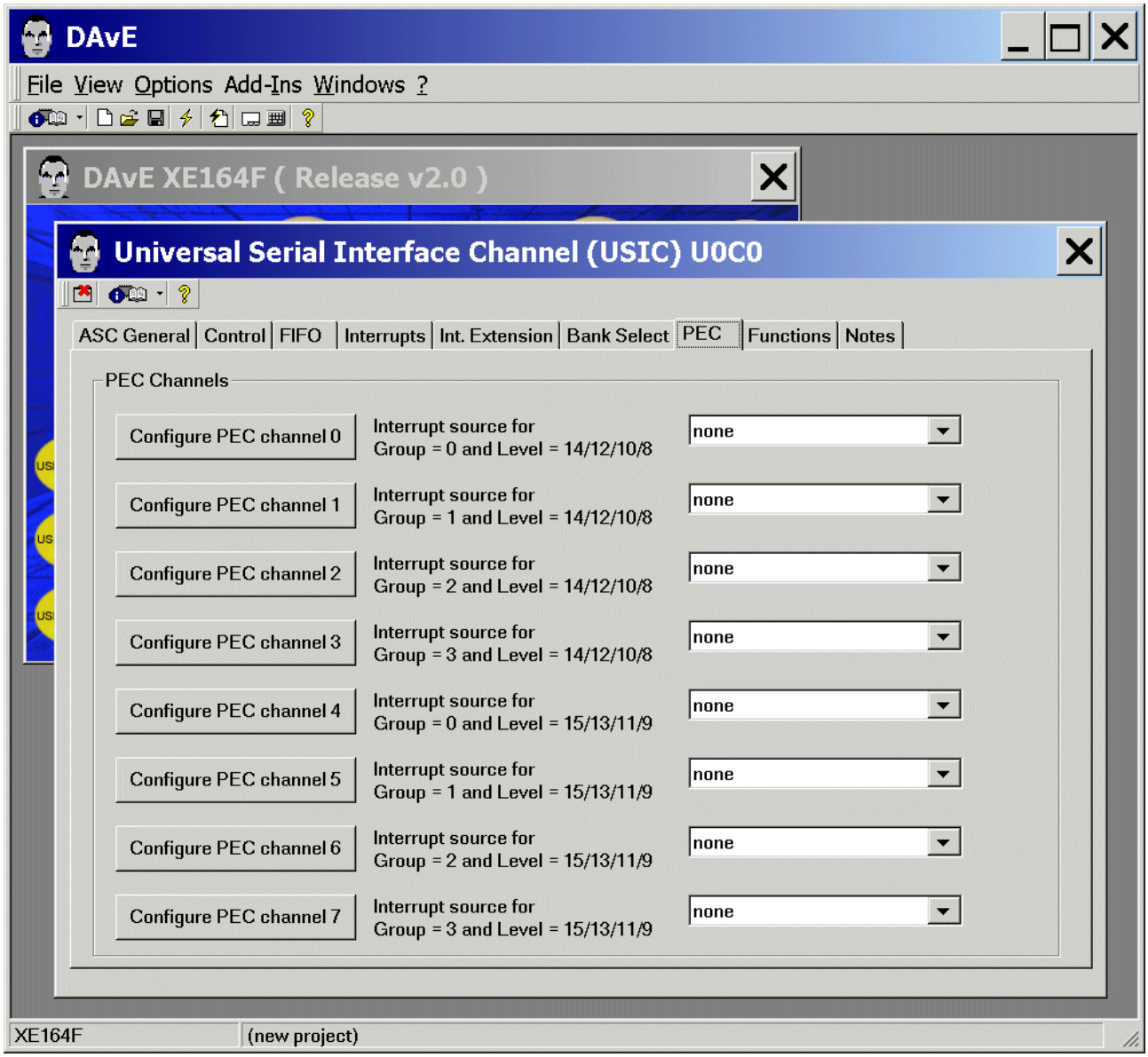
Bank Select: (do nothing)



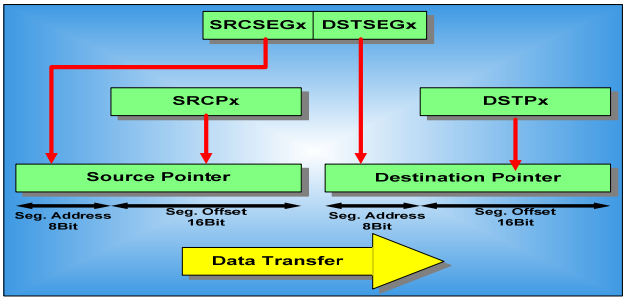
Note:
For our hello world program the 2 local register banks are not needed.



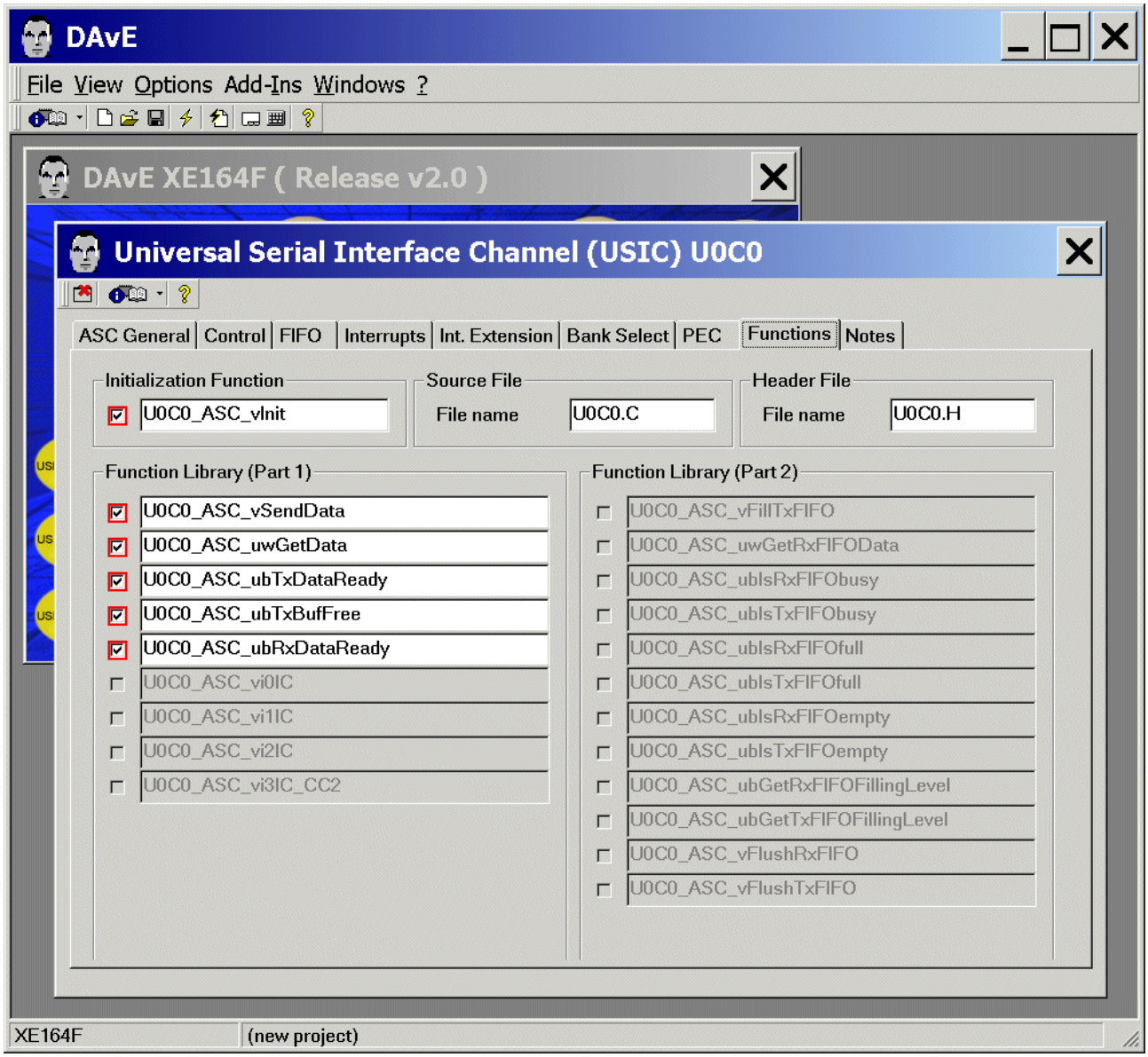
PEC: (do nothing)



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

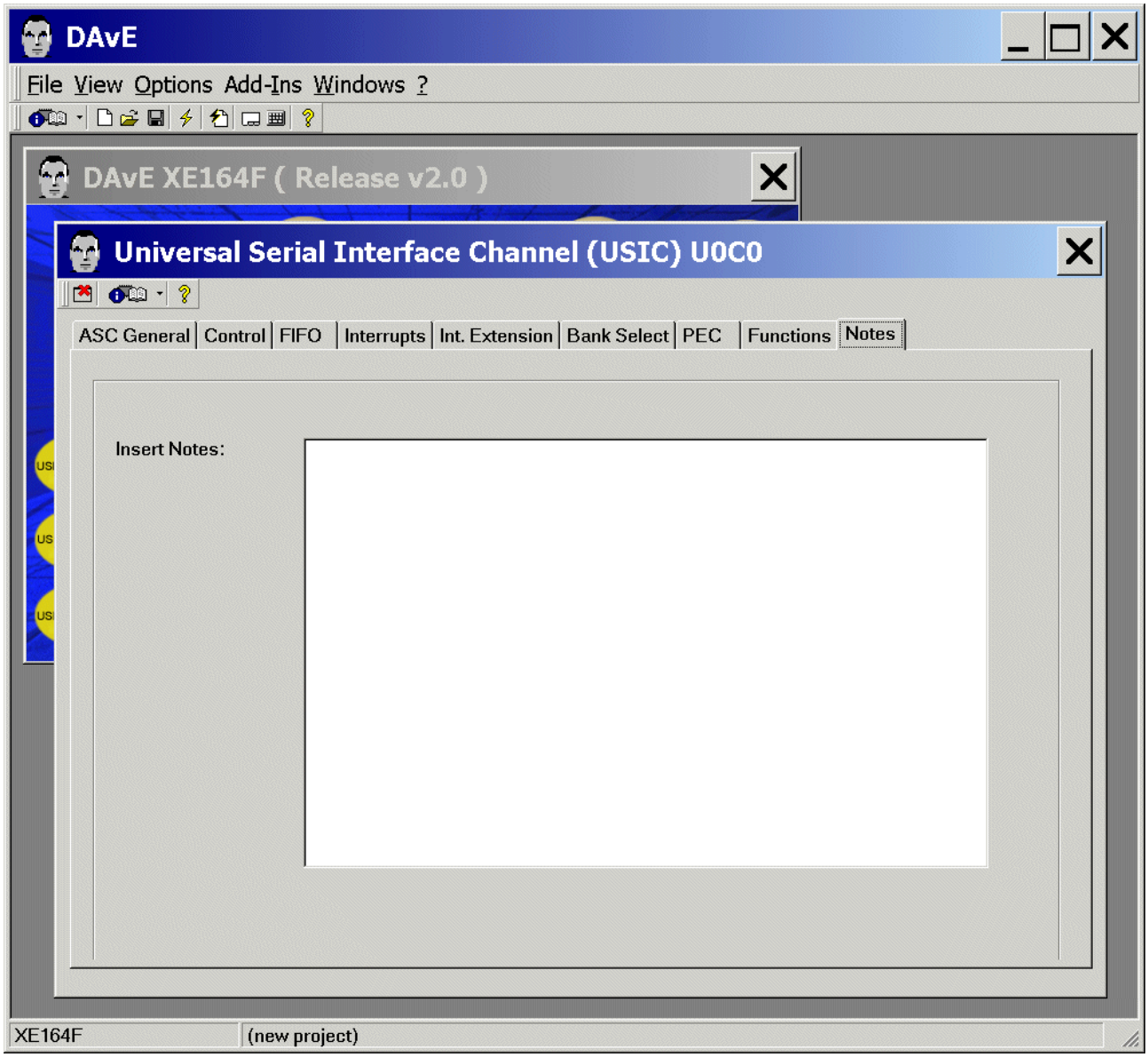


Functions: Initialization Function: click ☑ U0C0_ASC_vInit
Functions: Function Library (Part 1): click ☑ U0C0_ASC_vSendData
Functions: Function Library (Part 1): click ☑ U0C0_ASC_uwGetData
Functions: Function Library (Part 1): click ☑ U0C0_ASC_ubTxDataReady
Functions: Function Library (Part 1): click ☑ U0C0_ASC_ubTxBufFree
Functions: Function Library (Part 1): click ☑ U0C0_ASC_ubRxDataReady

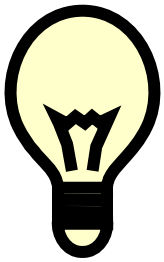


Note:
You can change function names (e.g. U0C0_ASC_vInit) and file names (e.g. U0C0.C, U0C0.H) anytime.

Notes: (do nothing)



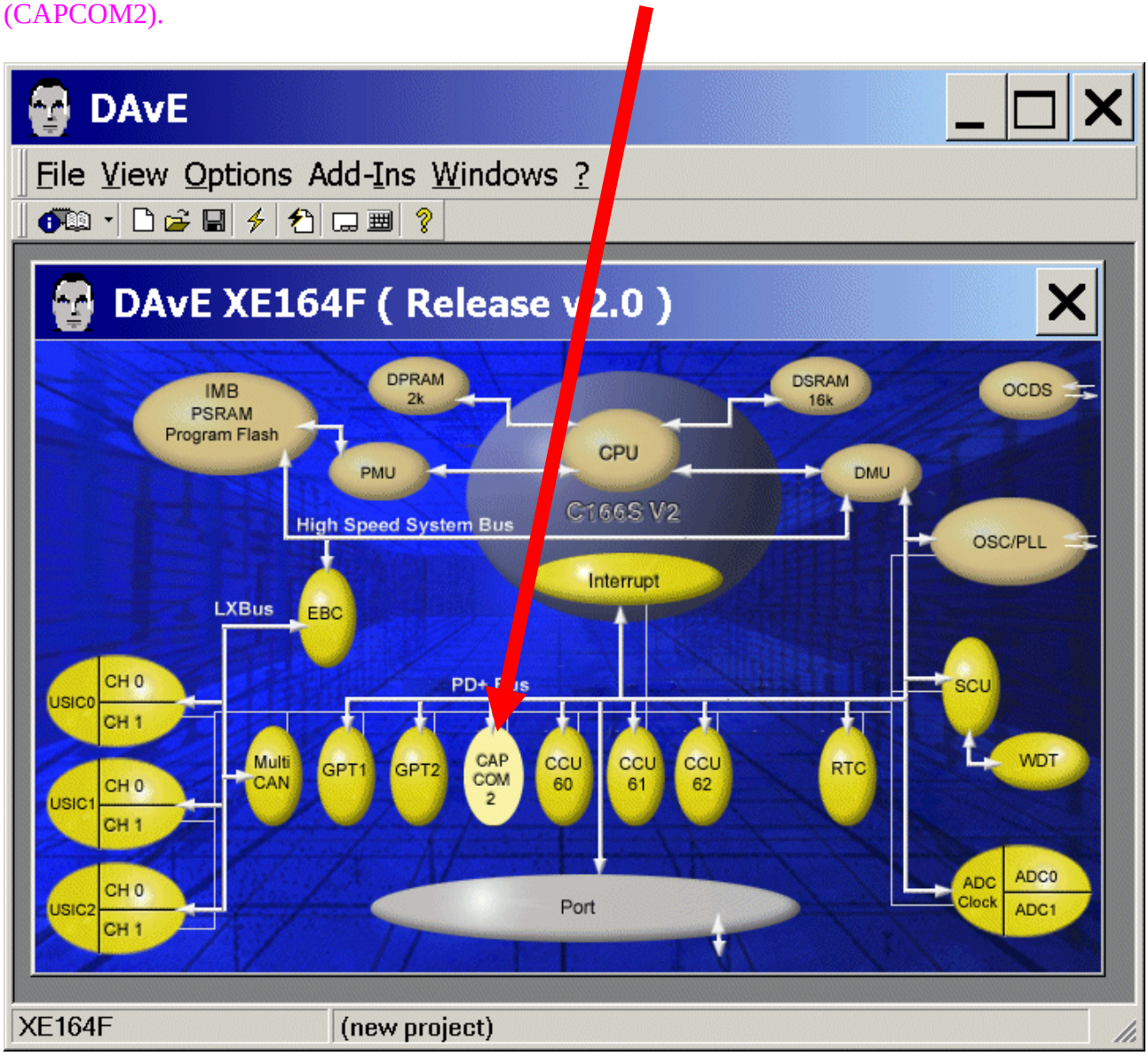
Note:
Notes: If you wish, you can insert your comments here.

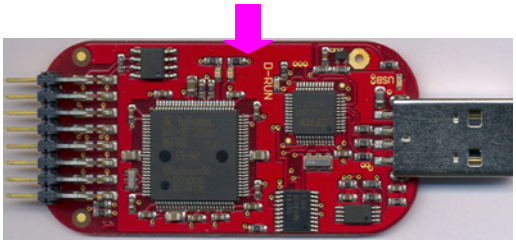


Exit and Save this dialog now by clicking the close button.

Configure Timer T7 in the CAPCOM 2 module:

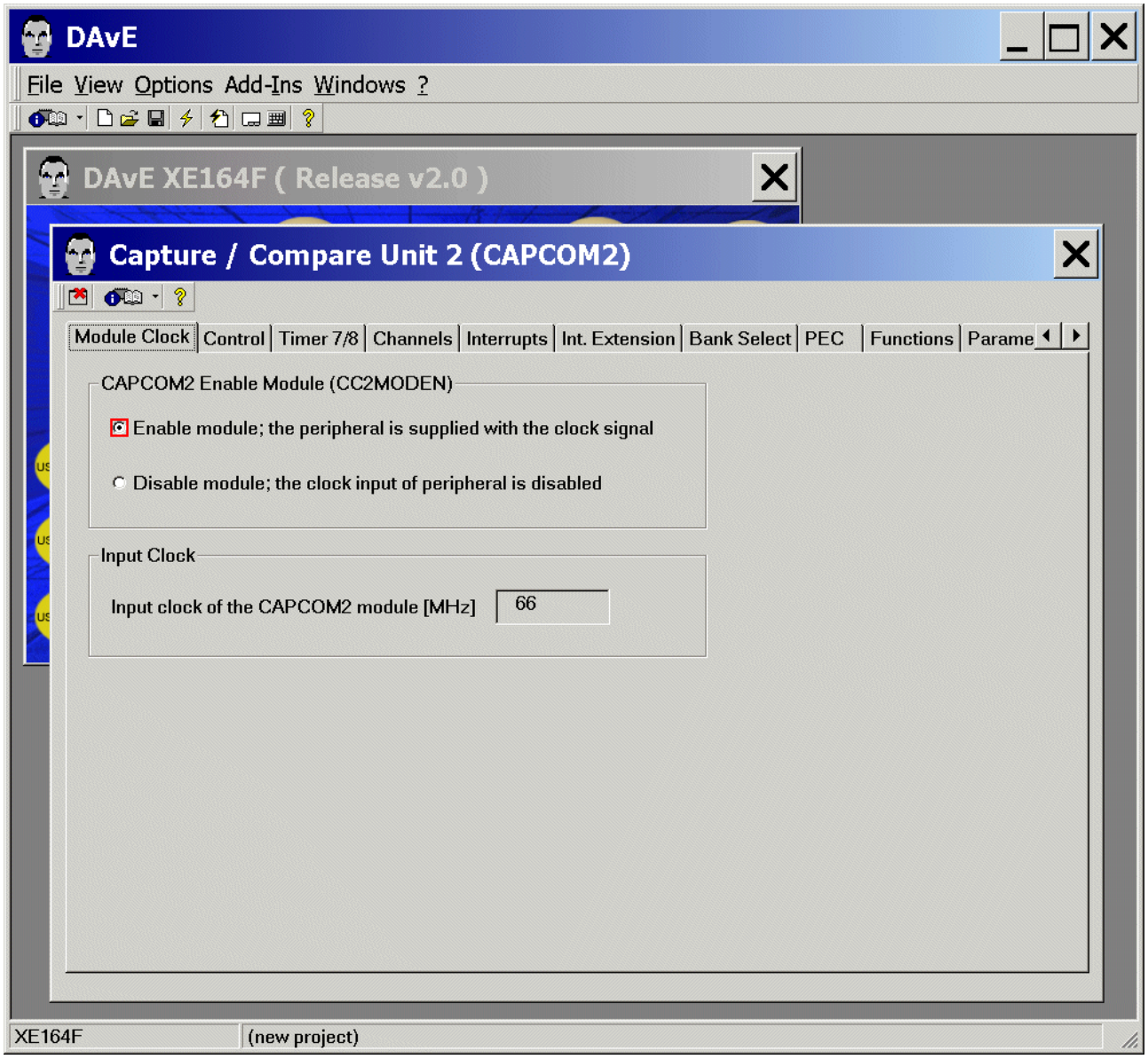
The configuration window/dialog can be opened by clicking the specific block/module (CAPCOM2).



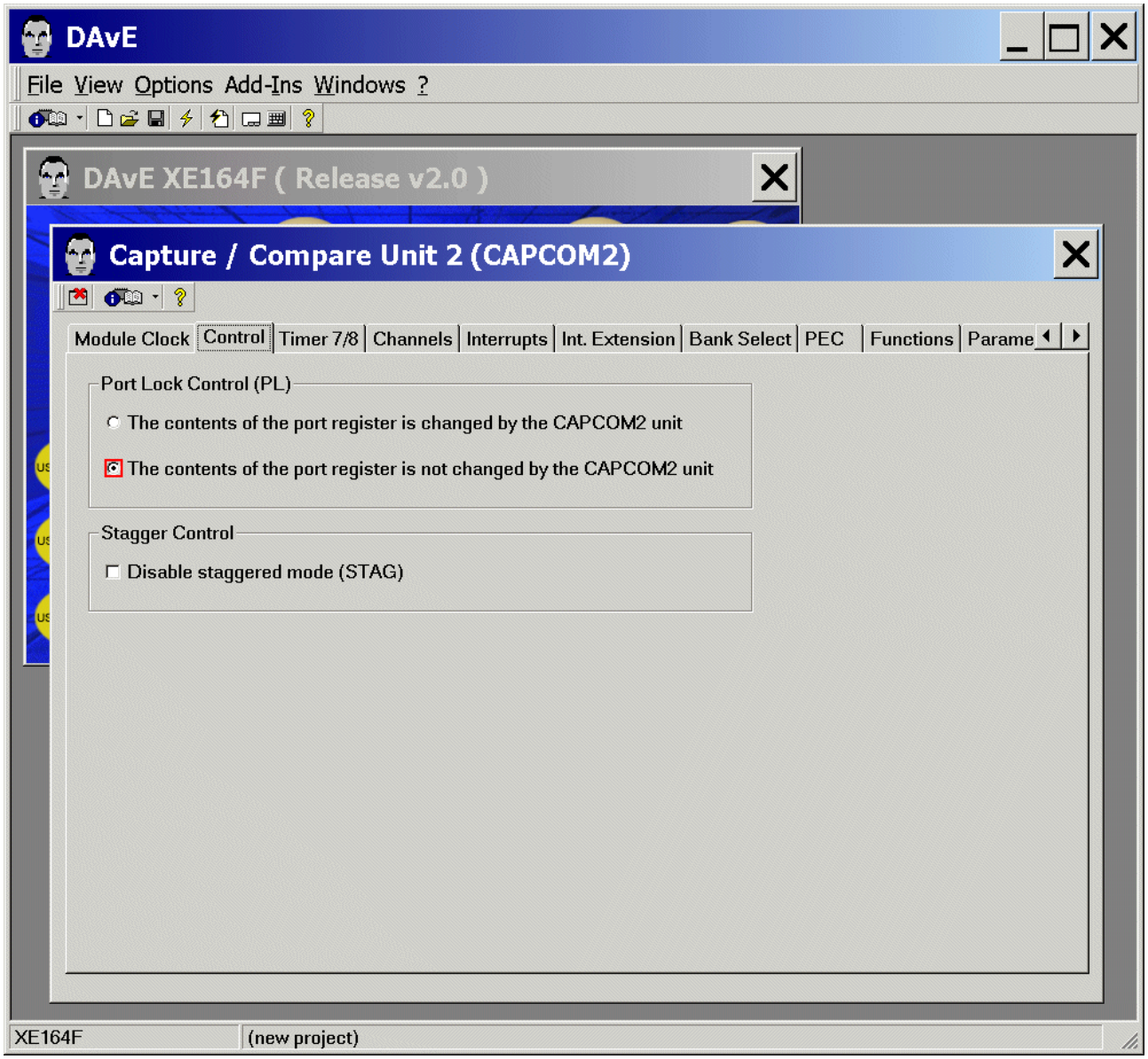


Note:
The LED on IO_Port_2.7 will be blinking (if selected in the main menu) with a frequency of about 1 second (done in the Timer_7-Interrupt-Service-Routine). Therefore we have to configure Timer_7.

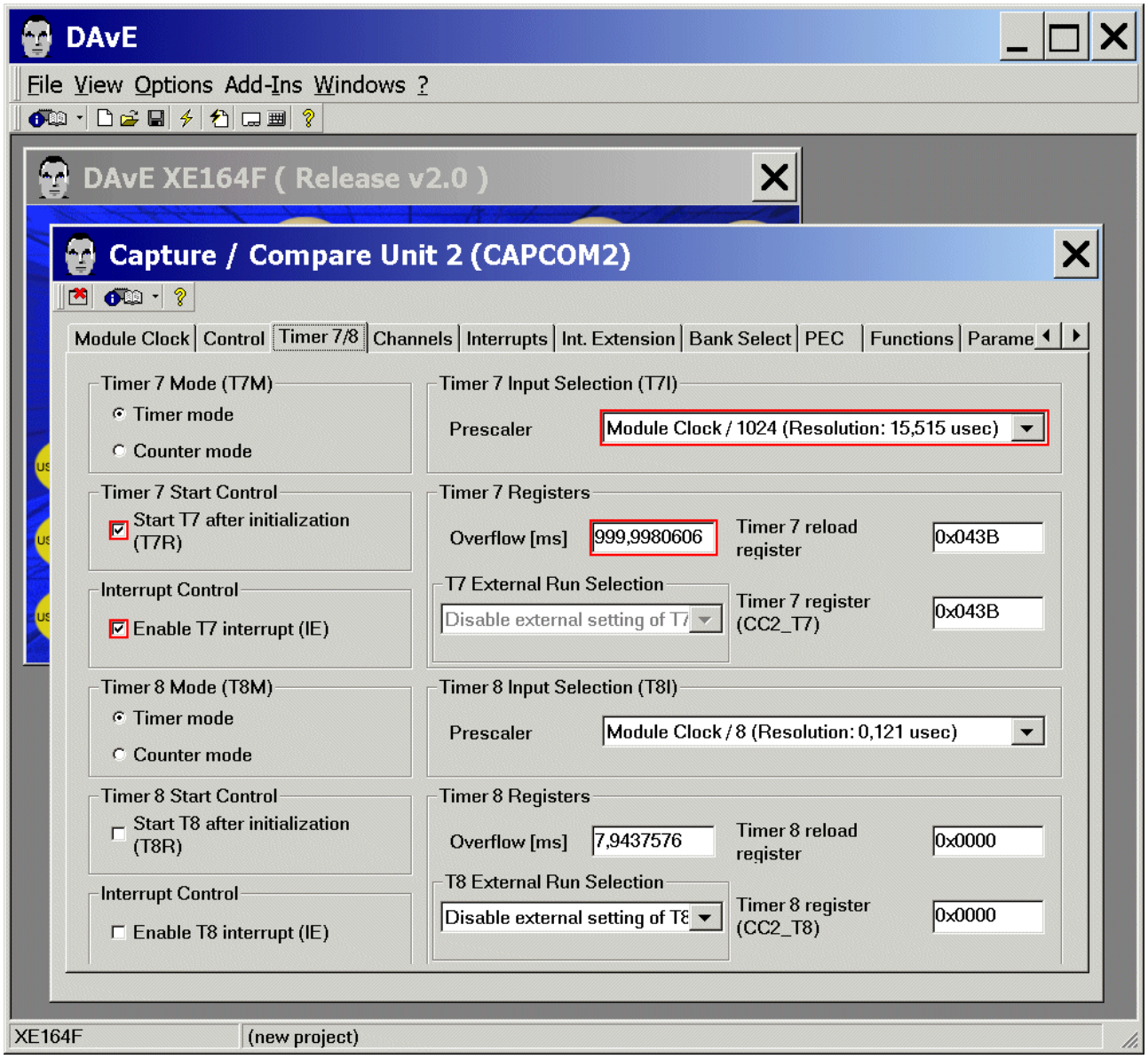
Module Clock: CAPCOM2 Enable Module: click ⊙ Enable module



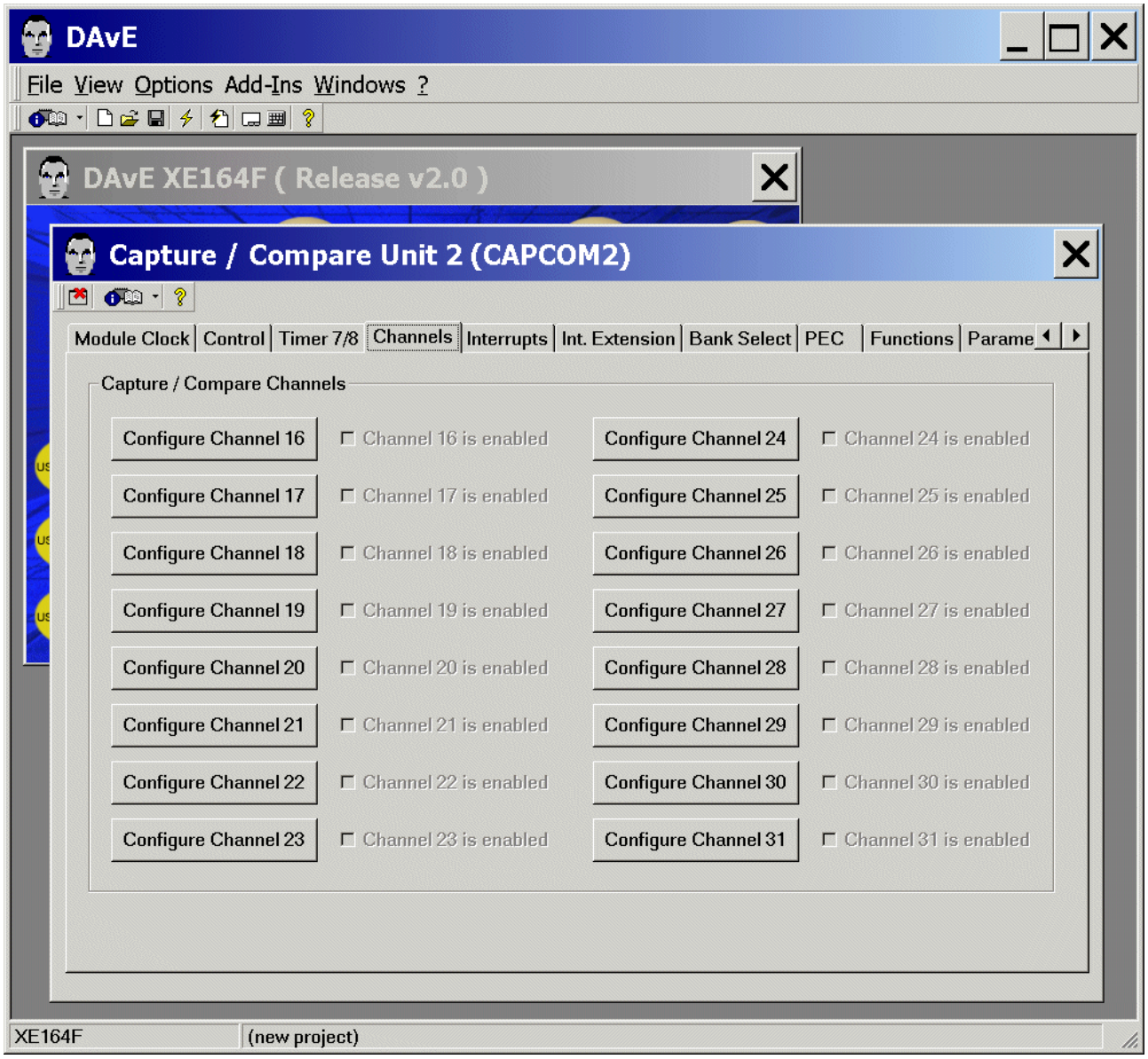
Control: Port Lock Control:
click ⊙ The contents of the port register is not changed by the CAPCOM2 unit



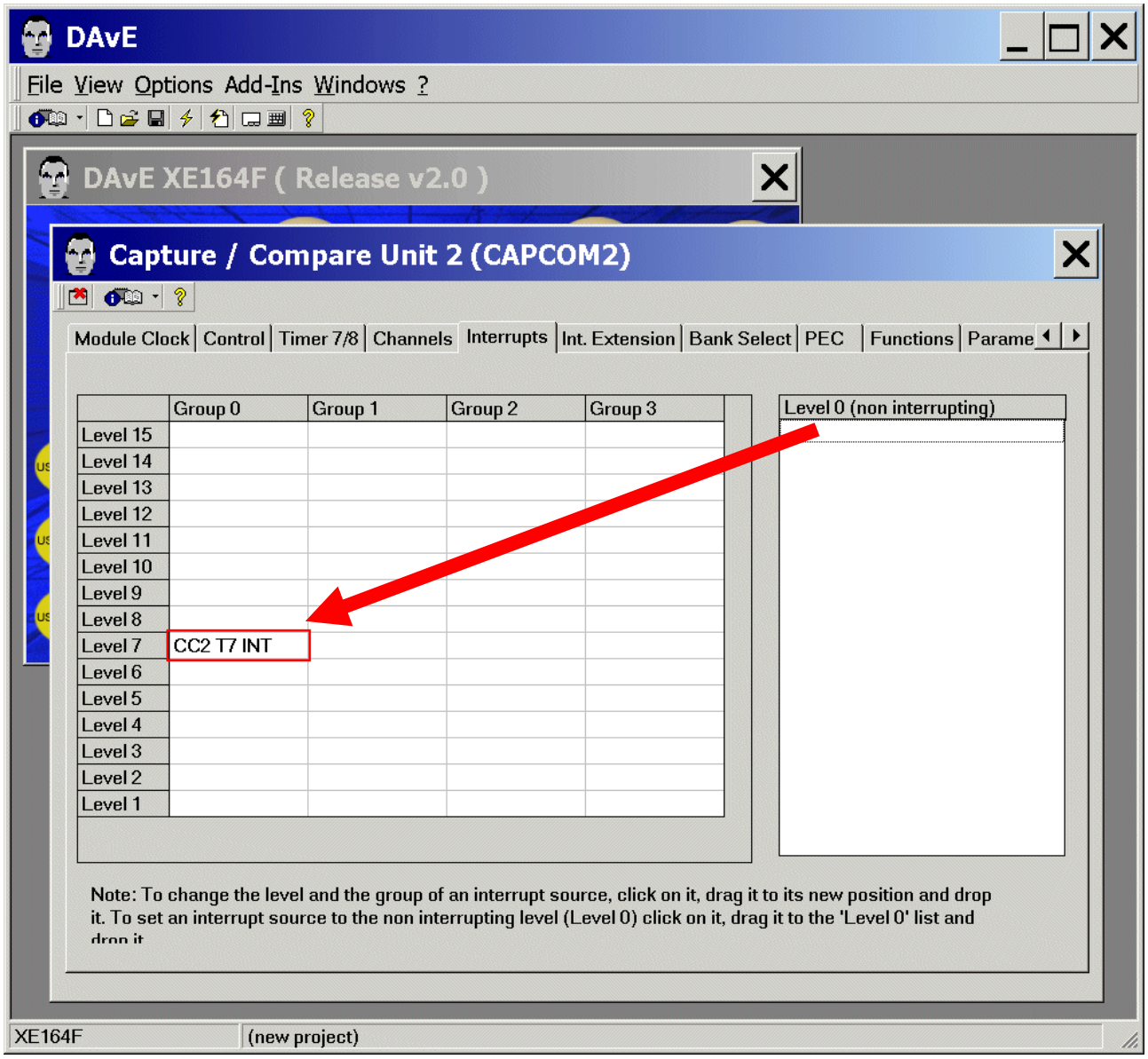
Timer 7/8: Timer 7 Start Control: click ✓ Start T7 after initialization (T7R)
Timer 7/8: Interrupt Control: click ✓ Enable T7 interrupt (IE)
Timer 7/8: Timer 7 Input Selection (T7I): Prescaler: choose Module Clock/1024
Timer 7/8: Timer 7 Registers: Overflow [s]: insert 1 <ENTER>



Channels: (do nothing)

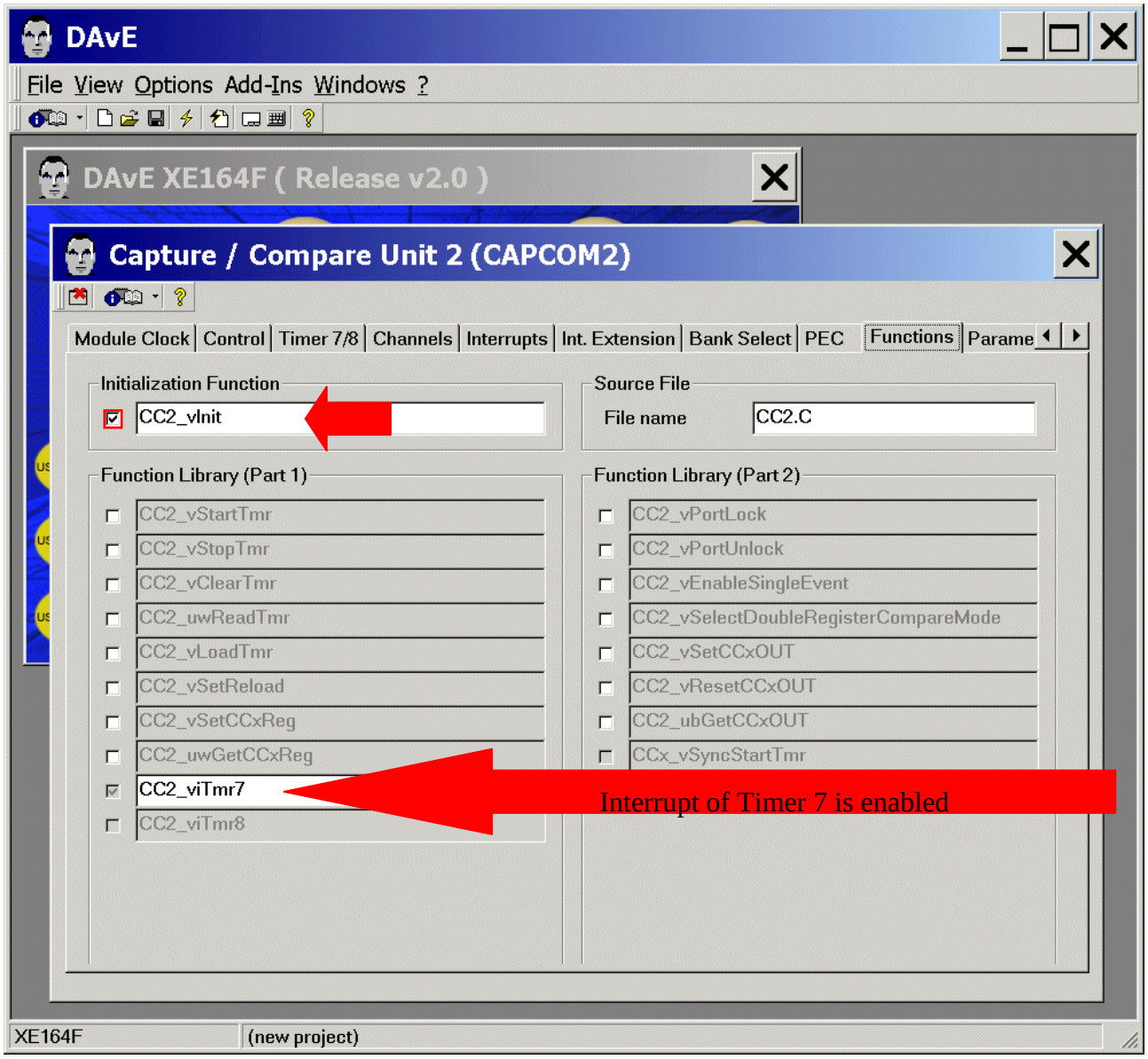


Interrupts: drag and drop the CC2 T7 INT to Interrupt Level 7, Group 0



Int. Extension: (do nothing)
Bank Select: (do nothing)
PEC: (do nothing)

Functions: Initialization Function: click/check ☑ CC2_vInit

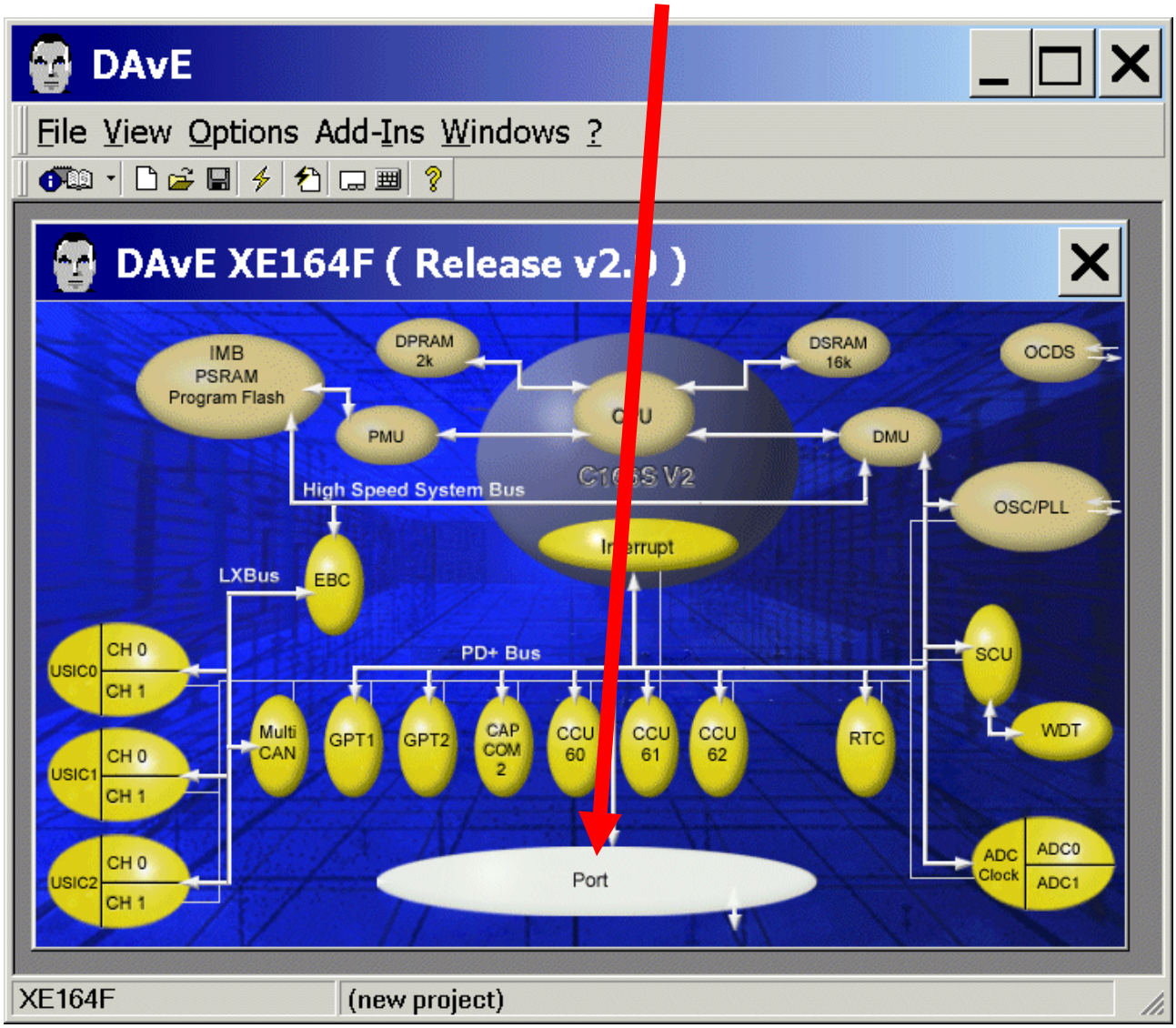


Parameters: (do nothing)
Notes: If you wish, you can insert your comments here.

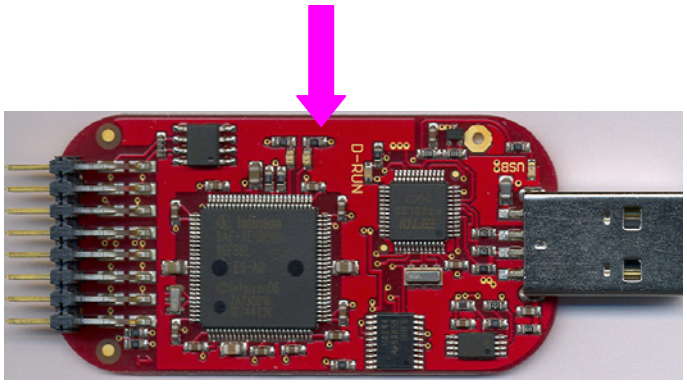
Exit and Save this dialog now by clicking the close button.

Configure Port 2 Pin 7 to Output :

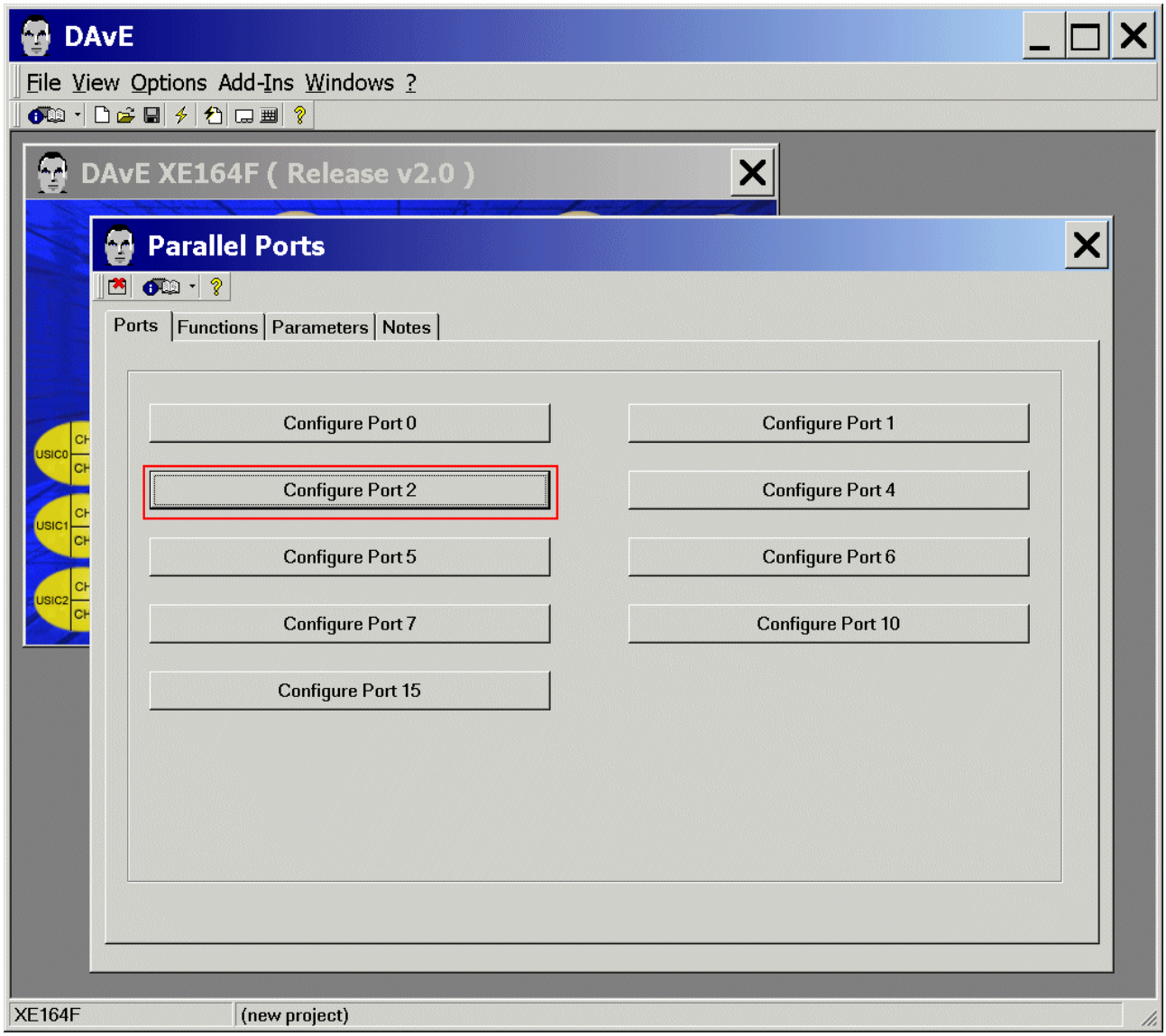
The configuration window/dialog can be opened by clicking the specific block/module (Port).



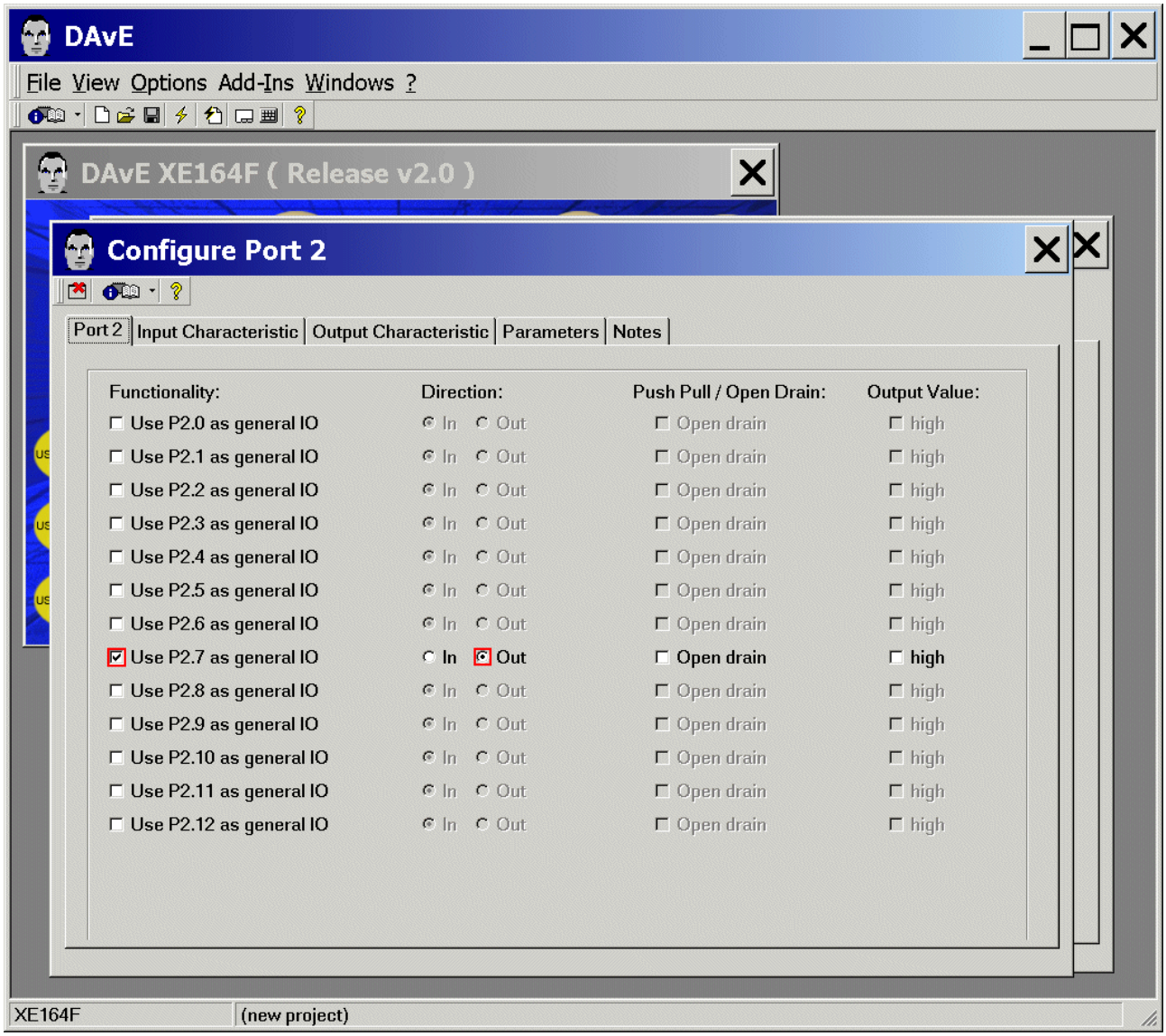
Note:
The LED is connected to IO_Port_2.7



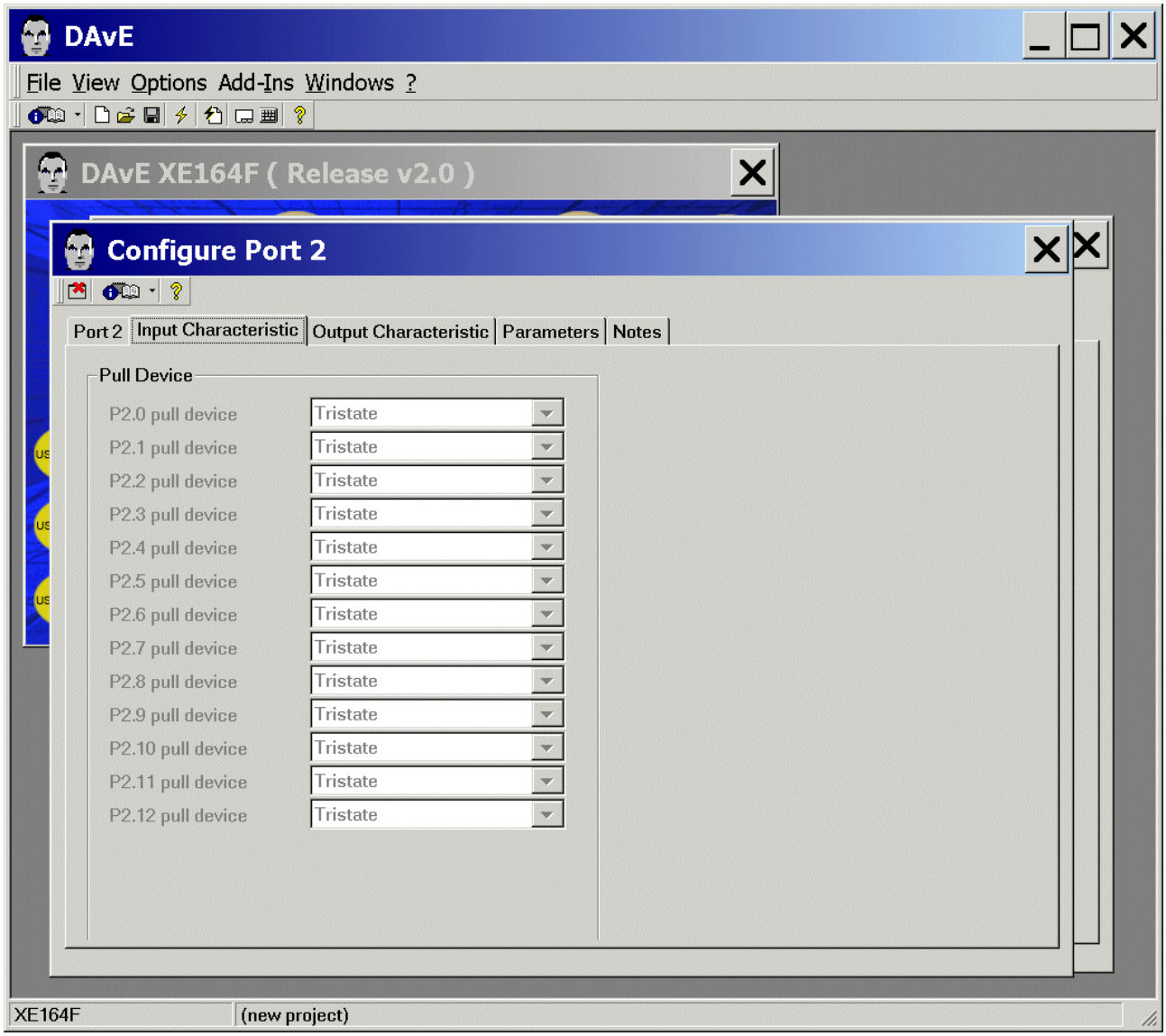
Ports: click “Configure Port 2”



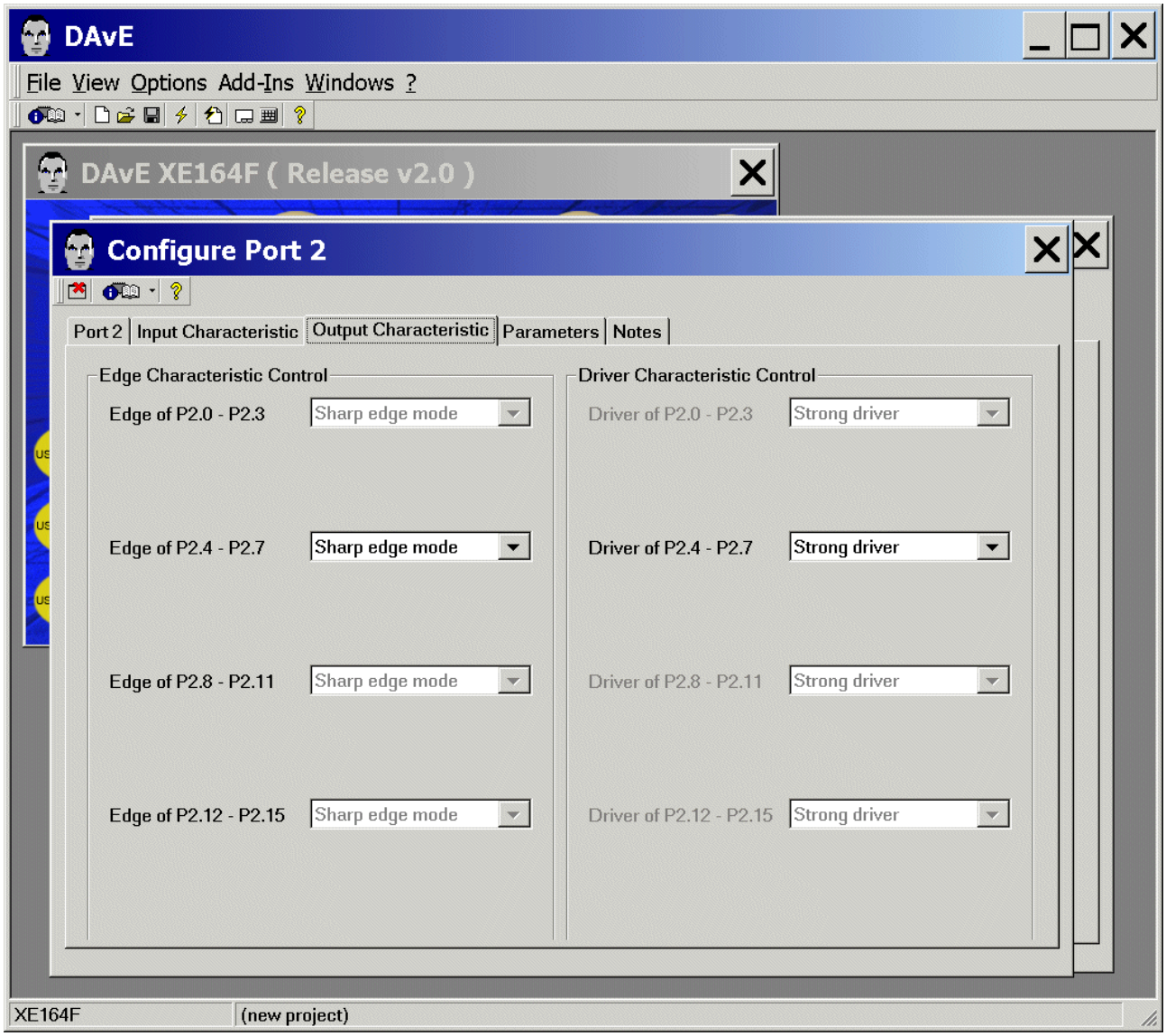
Port 2: Functionality: click ☑ Use P2.7 as general IO - Direction: click ⊙ Out



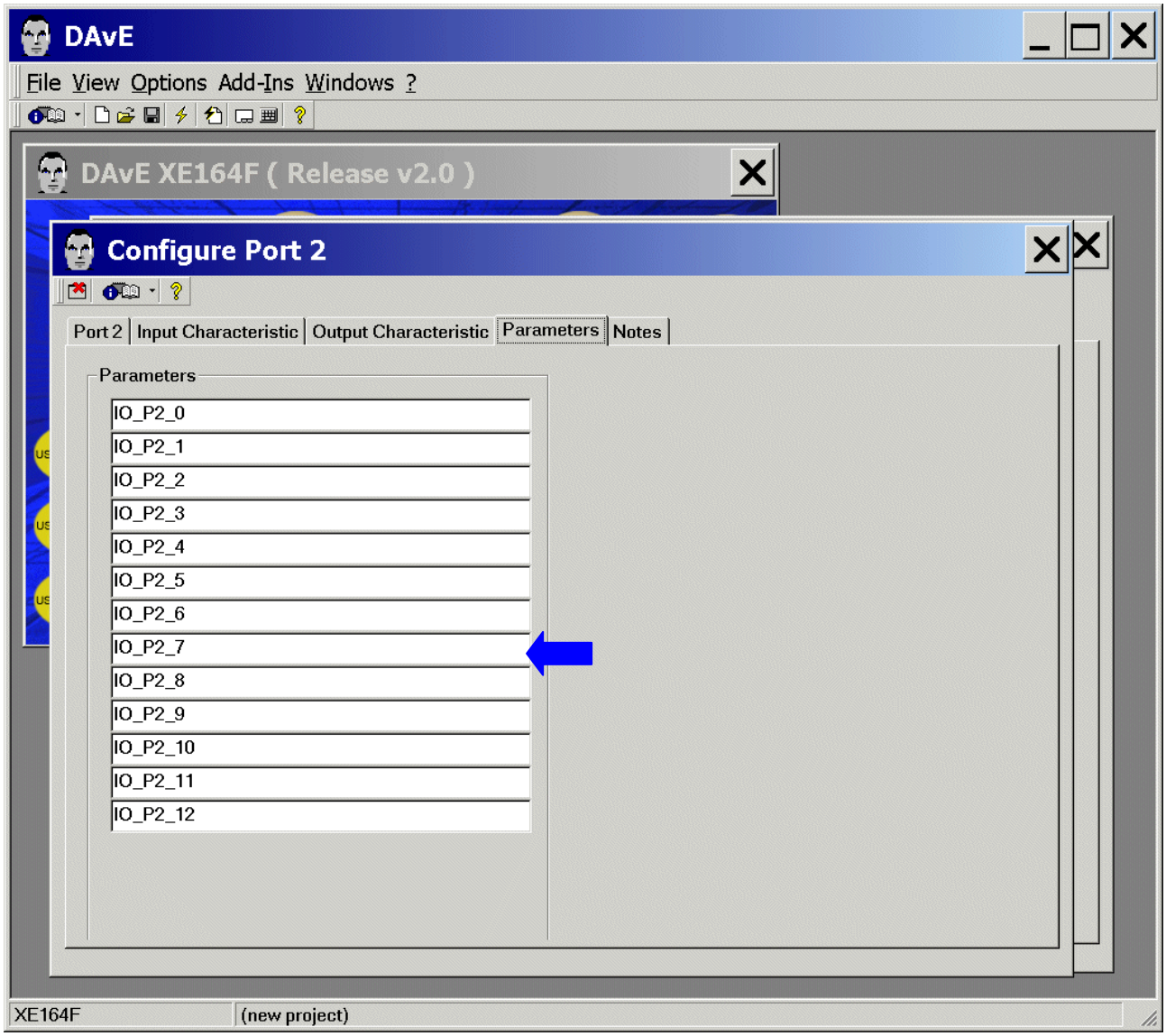
Input Characteristic: (do nothing)



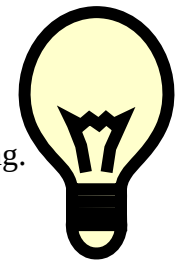
Output Characteristic: (do nothing)



Parameters: (do nothing)



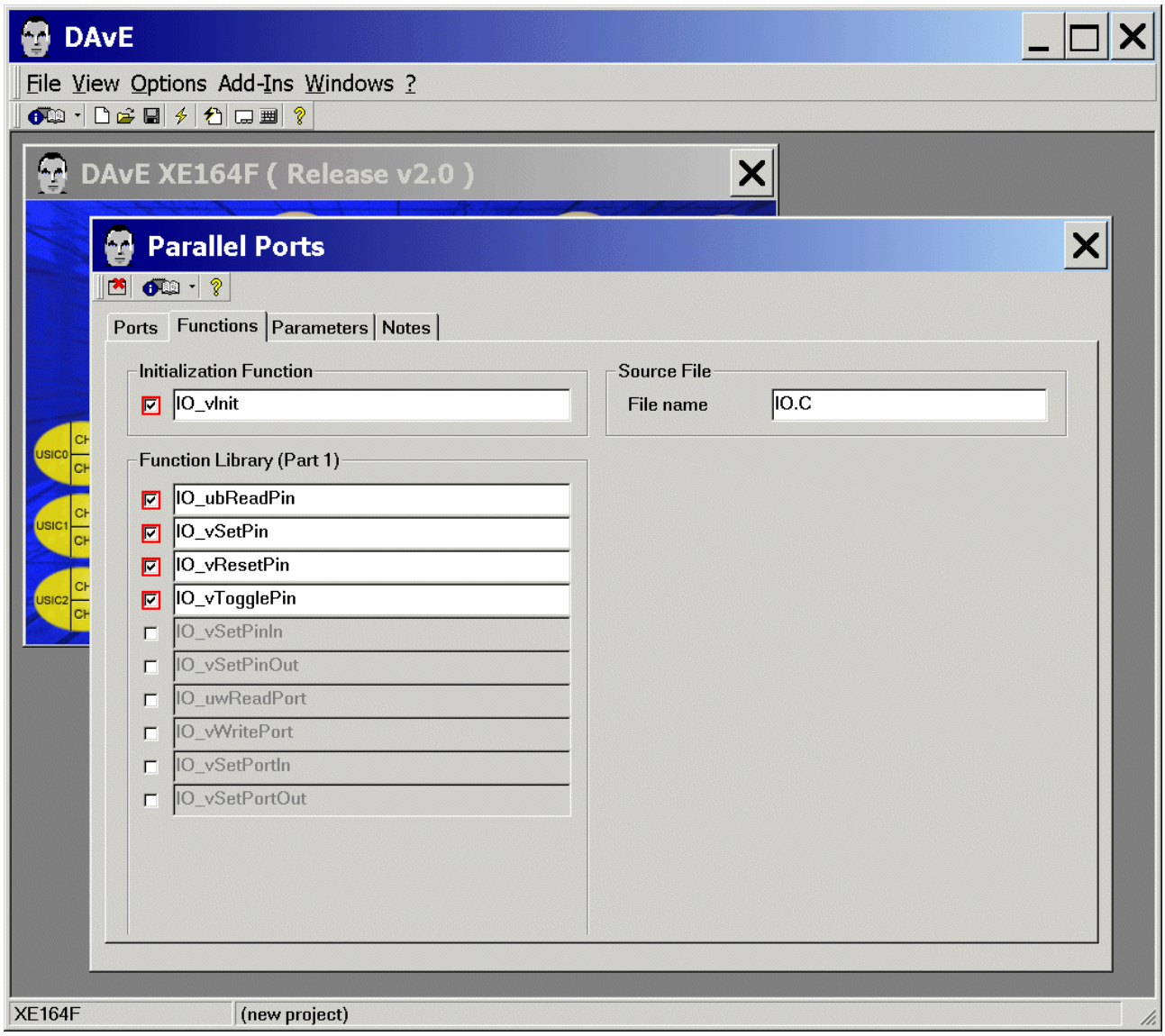
Note:
We will use the name IO_P2_7 in application programming.



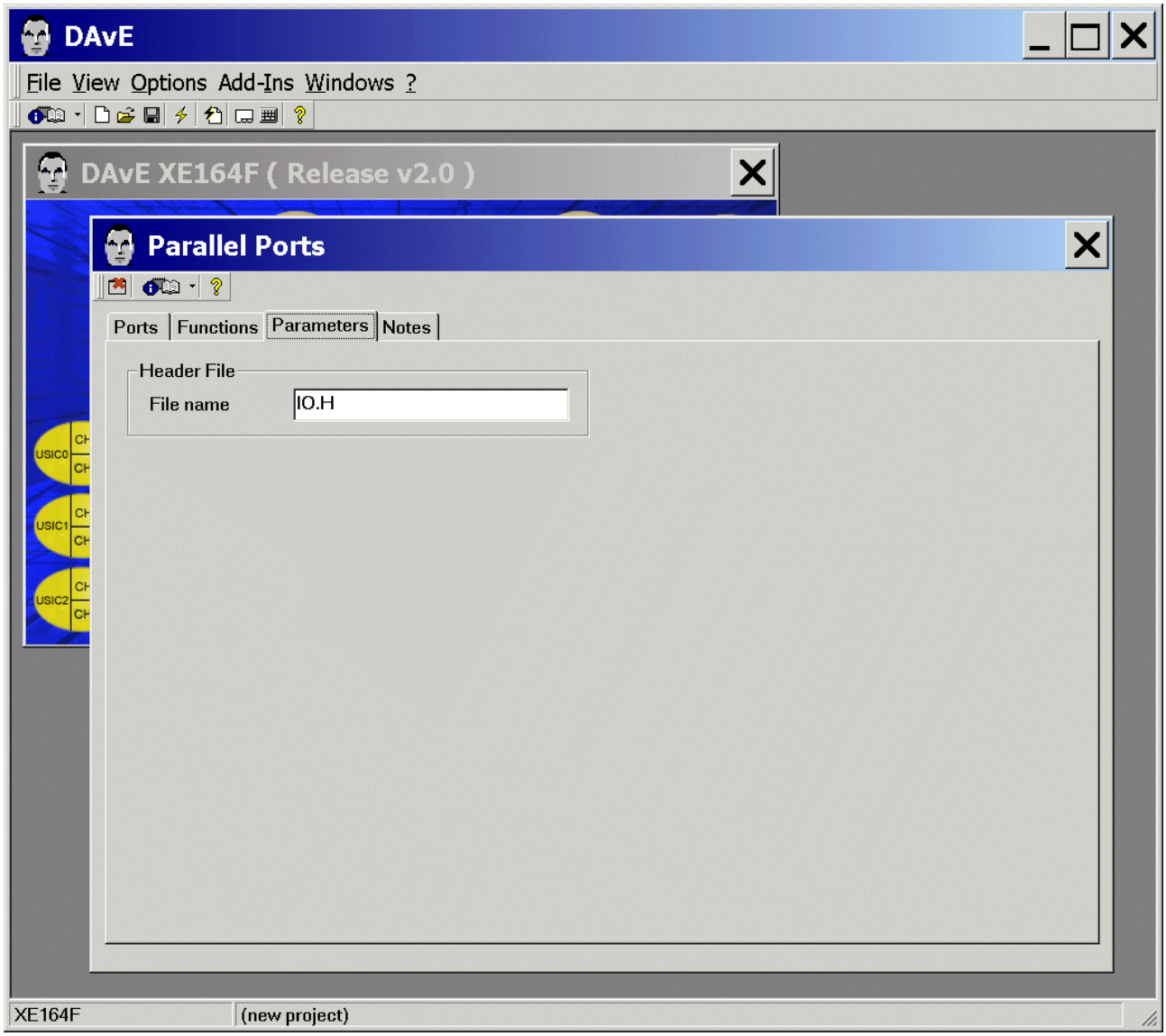
Notes: If you wish, you can insert your comments here.

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.

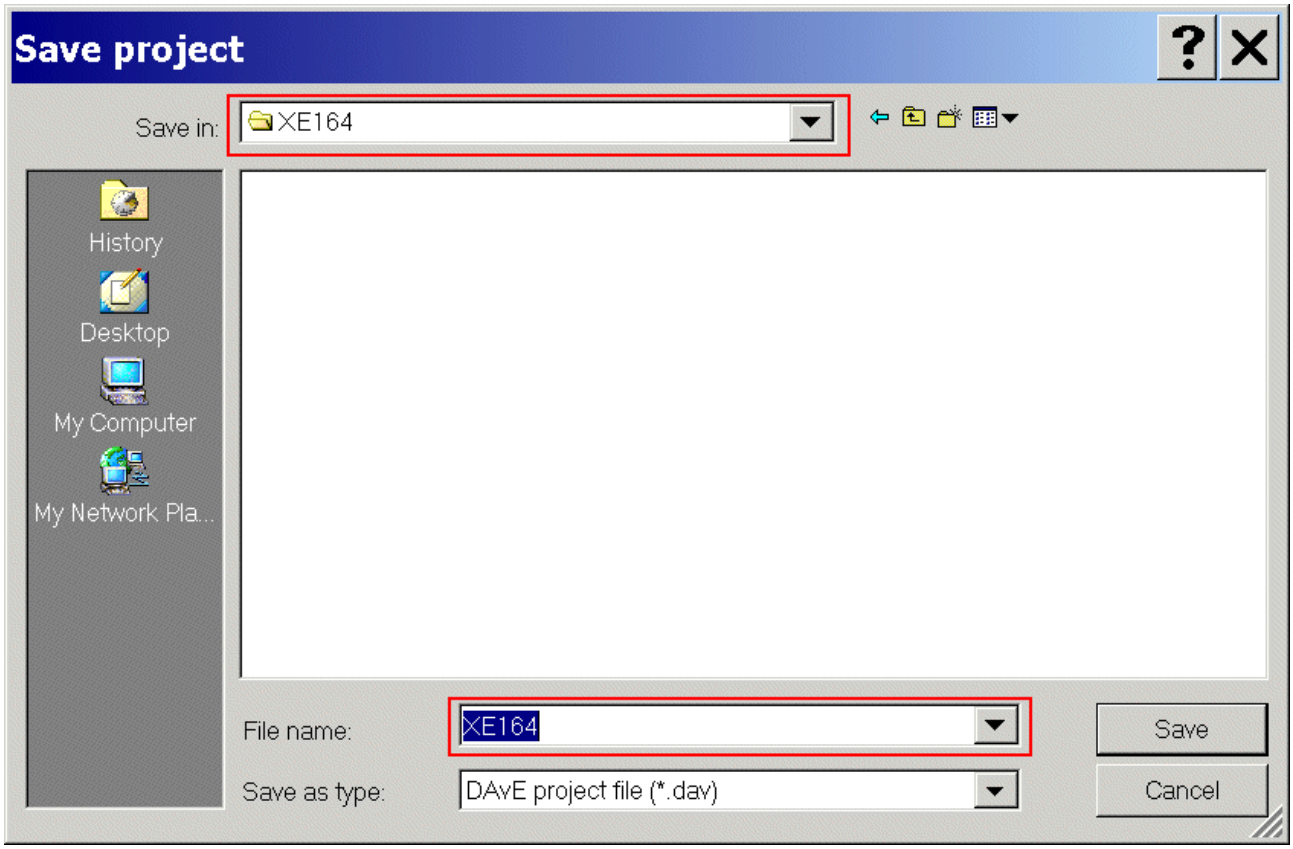
Exit and Save this dialog now by clicking the close button.

Save the project:

File
Save

Save project: Save in C:\XE164 (create new directory

)
File name: XE164

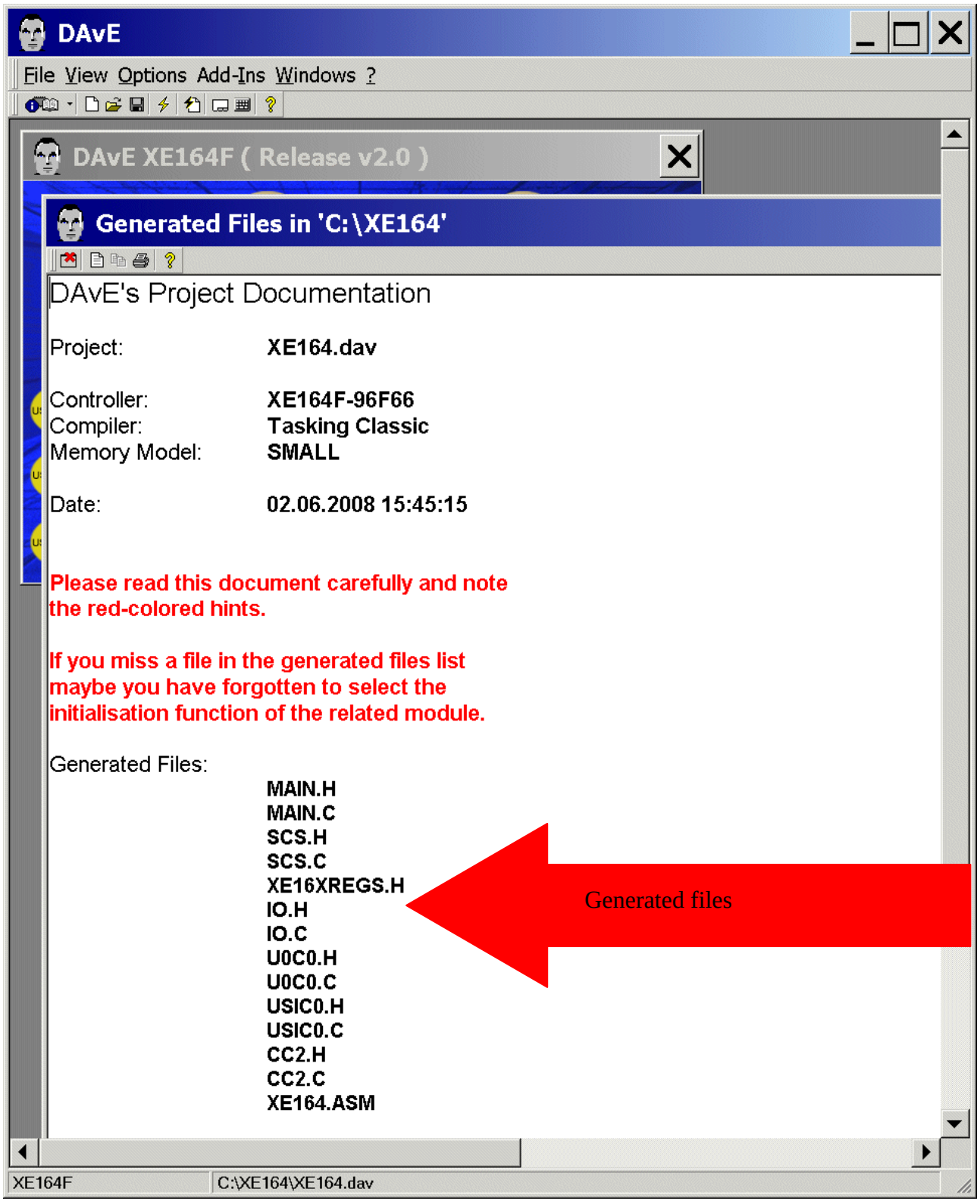


Save

Generate Code:

File Generate Code	or click

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



File - Exit

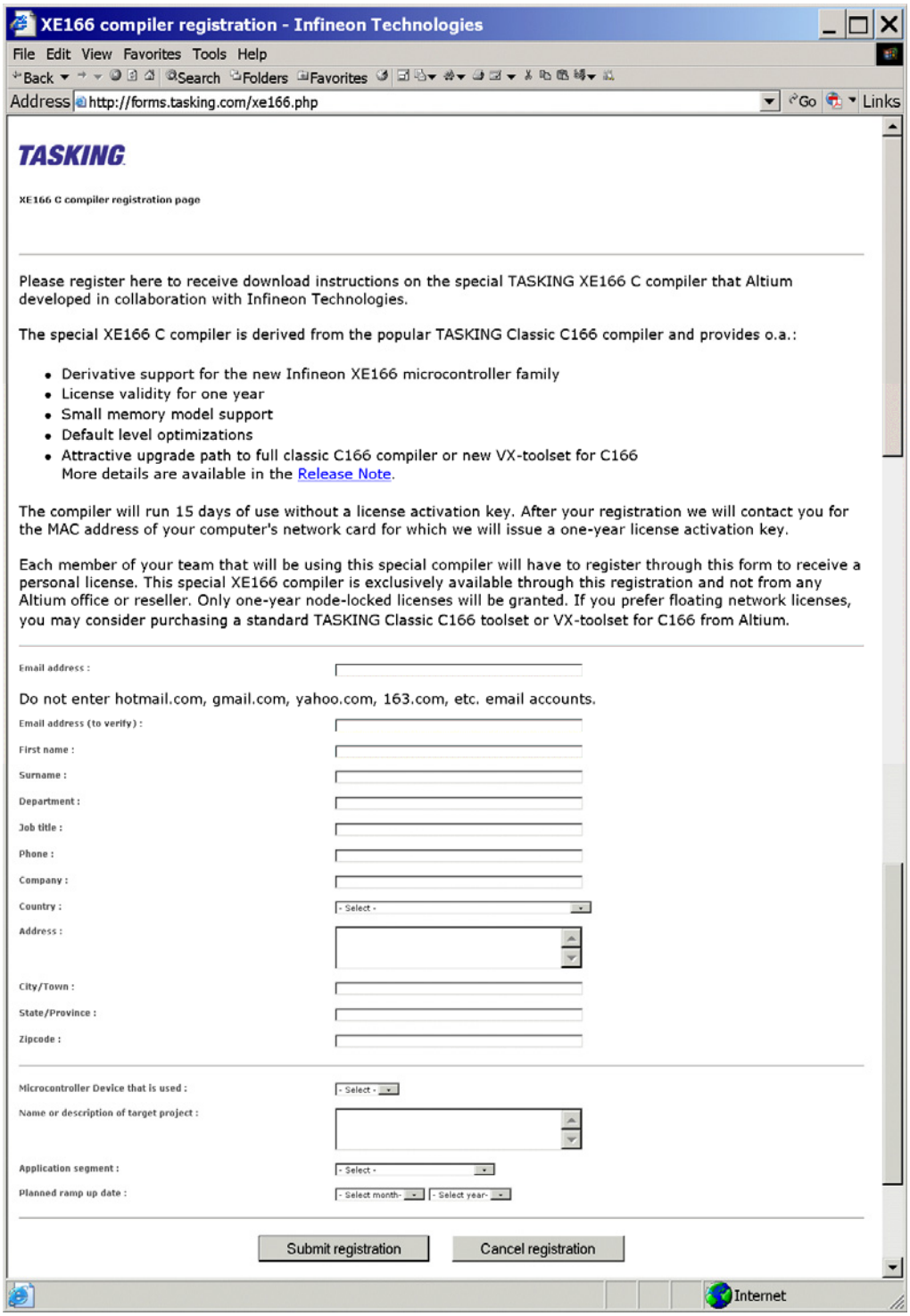
Save changes?

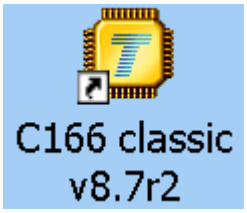
Click: Yes

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

Install the tool chain:
You can request the TASKING Development Tools @ http://forms.tasking.com/xe166.php :

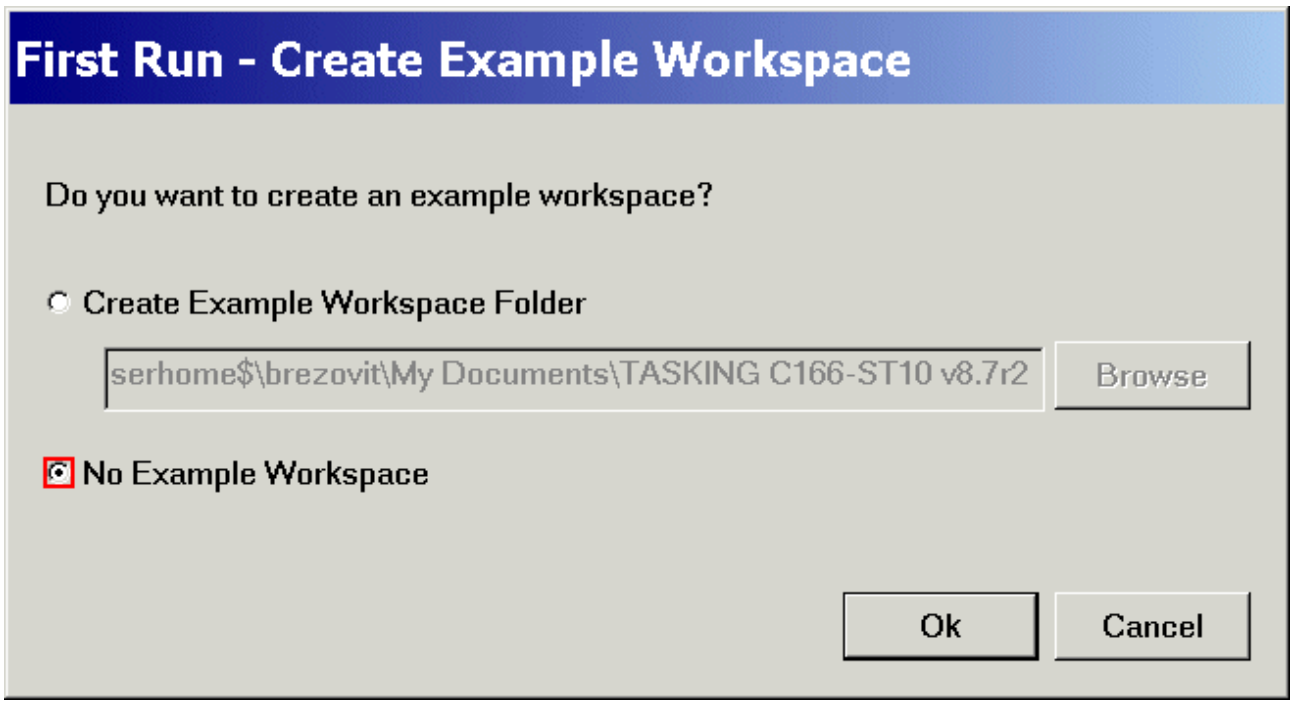
Note:
Altium's TASKING Classic Compiler for XE166 devices is free of charge for one year!





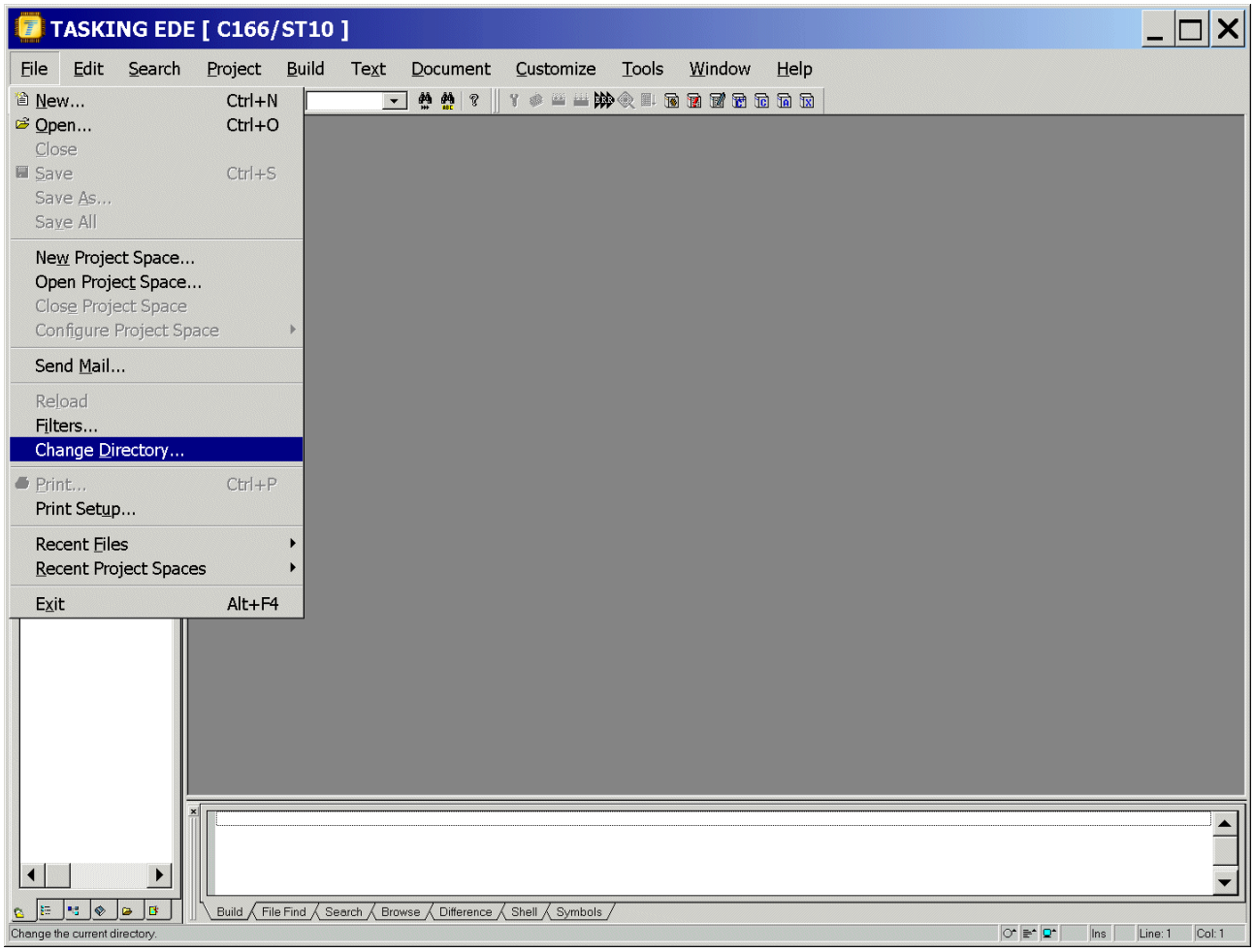
Start TASKING EDE Classic, select the working directory and include the DAvE Project:

First Run – Create Example Workspace: click ⊙ No Example Workspace

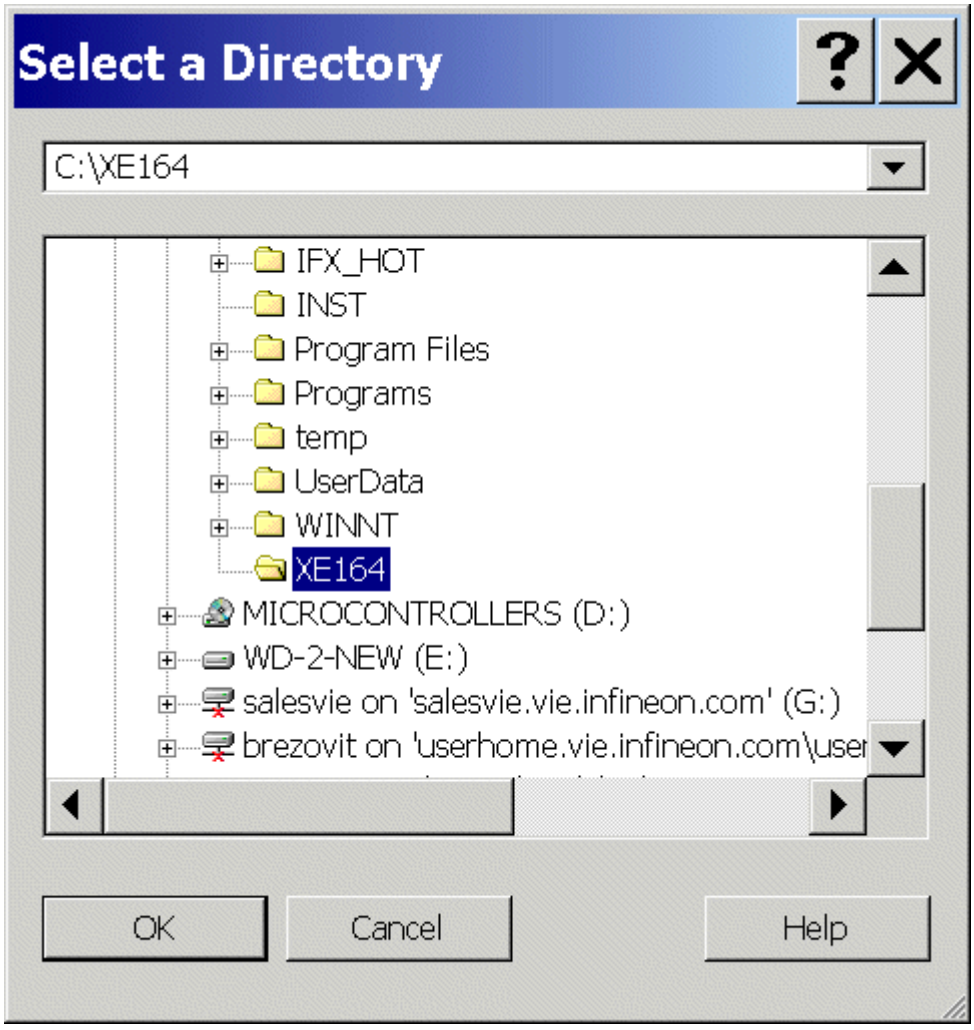


OK

File - Change Directory…

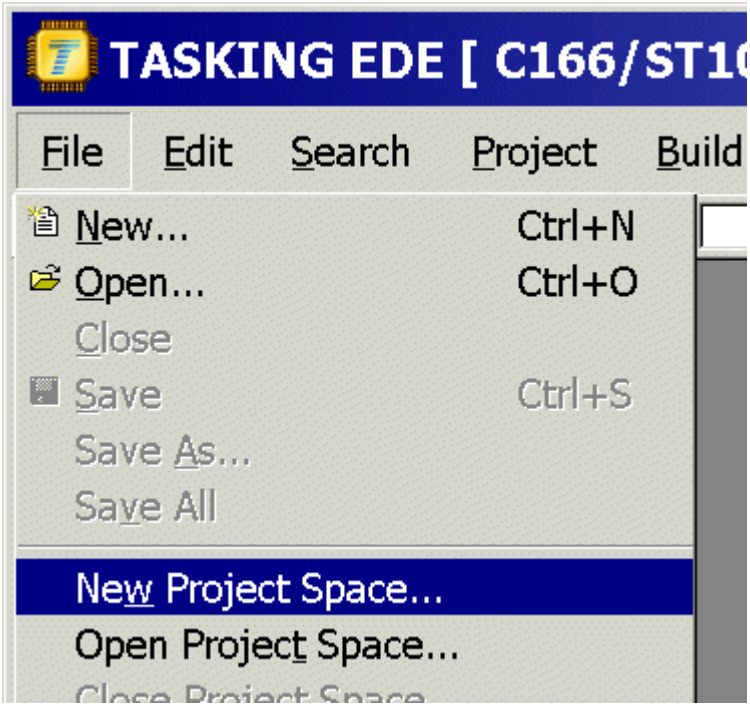


Choose C:\XE164

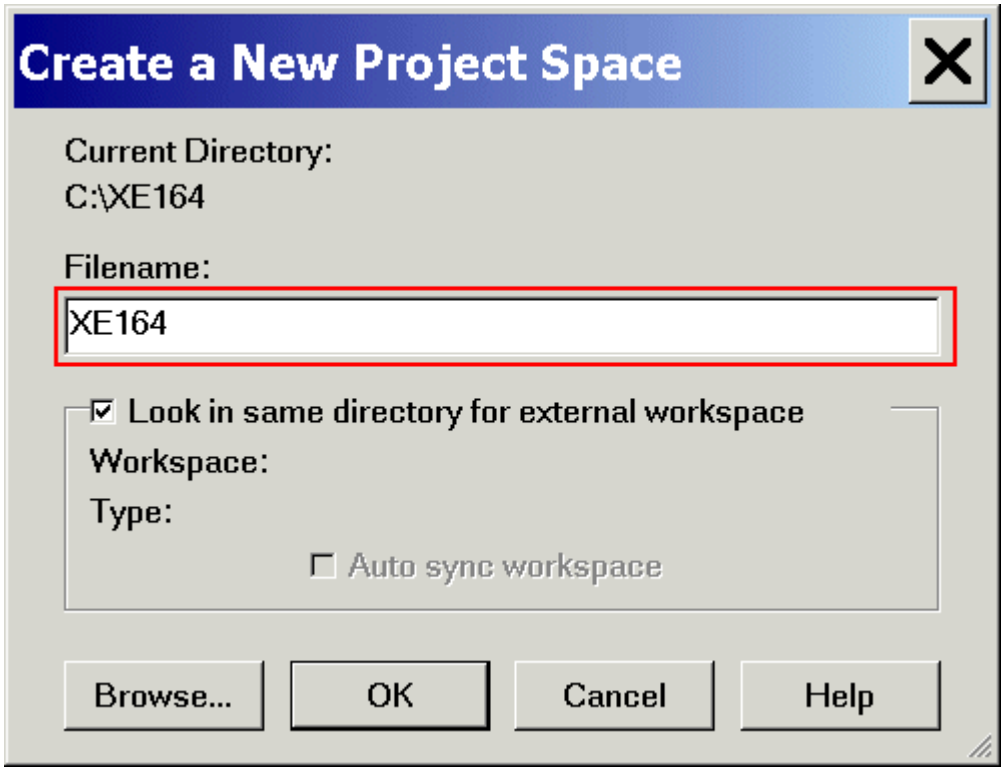


OK

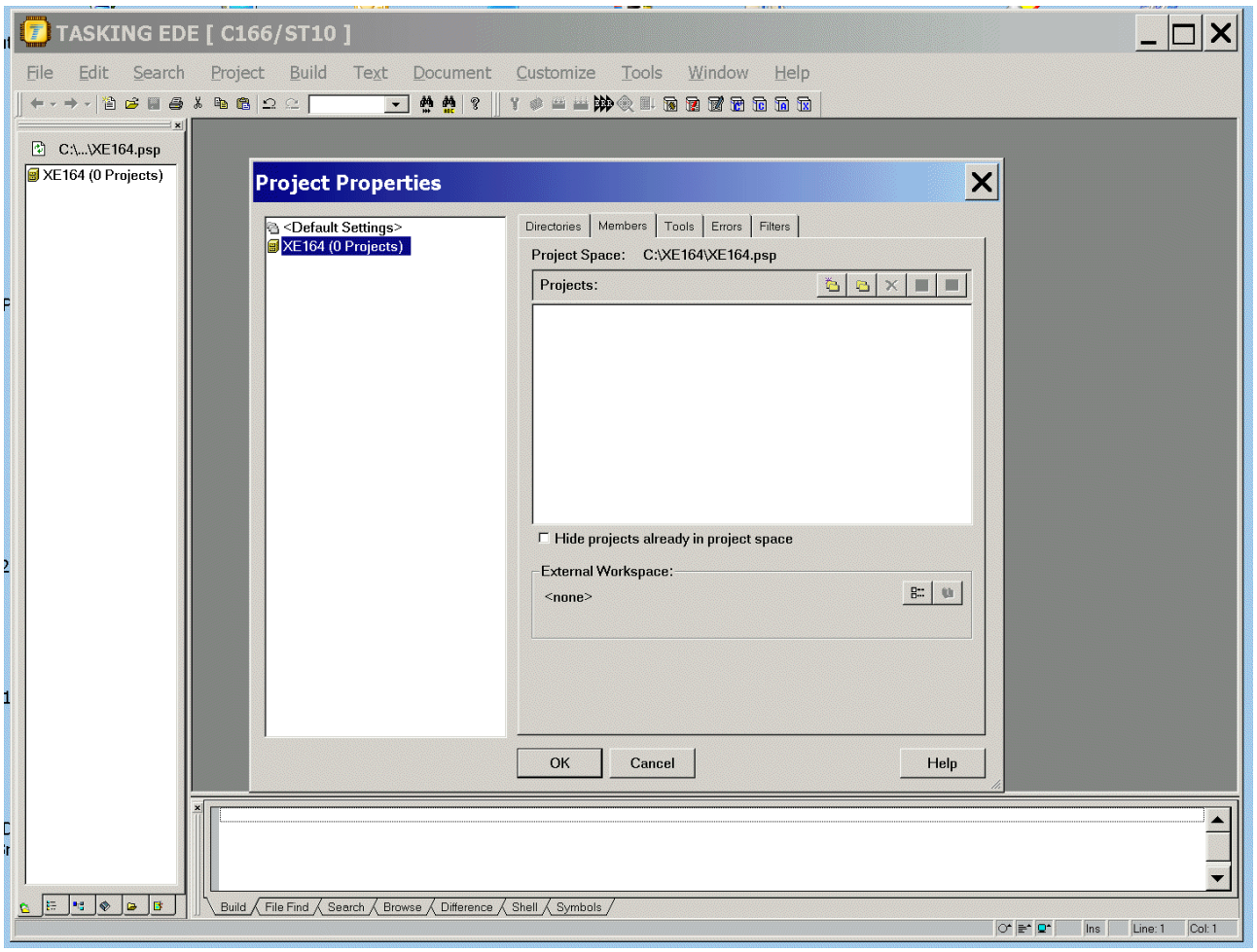
File - New Project Space



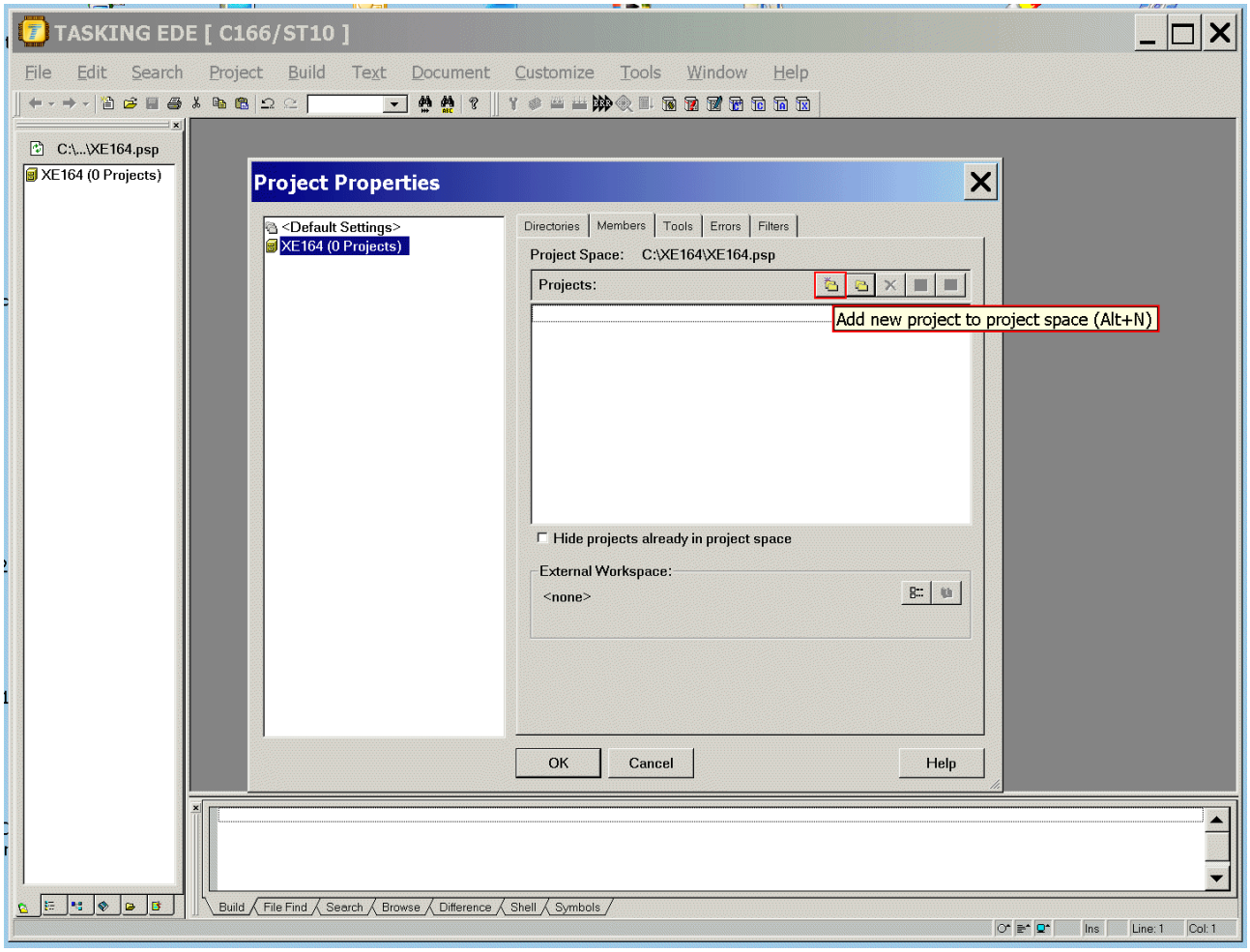
Insert XE164



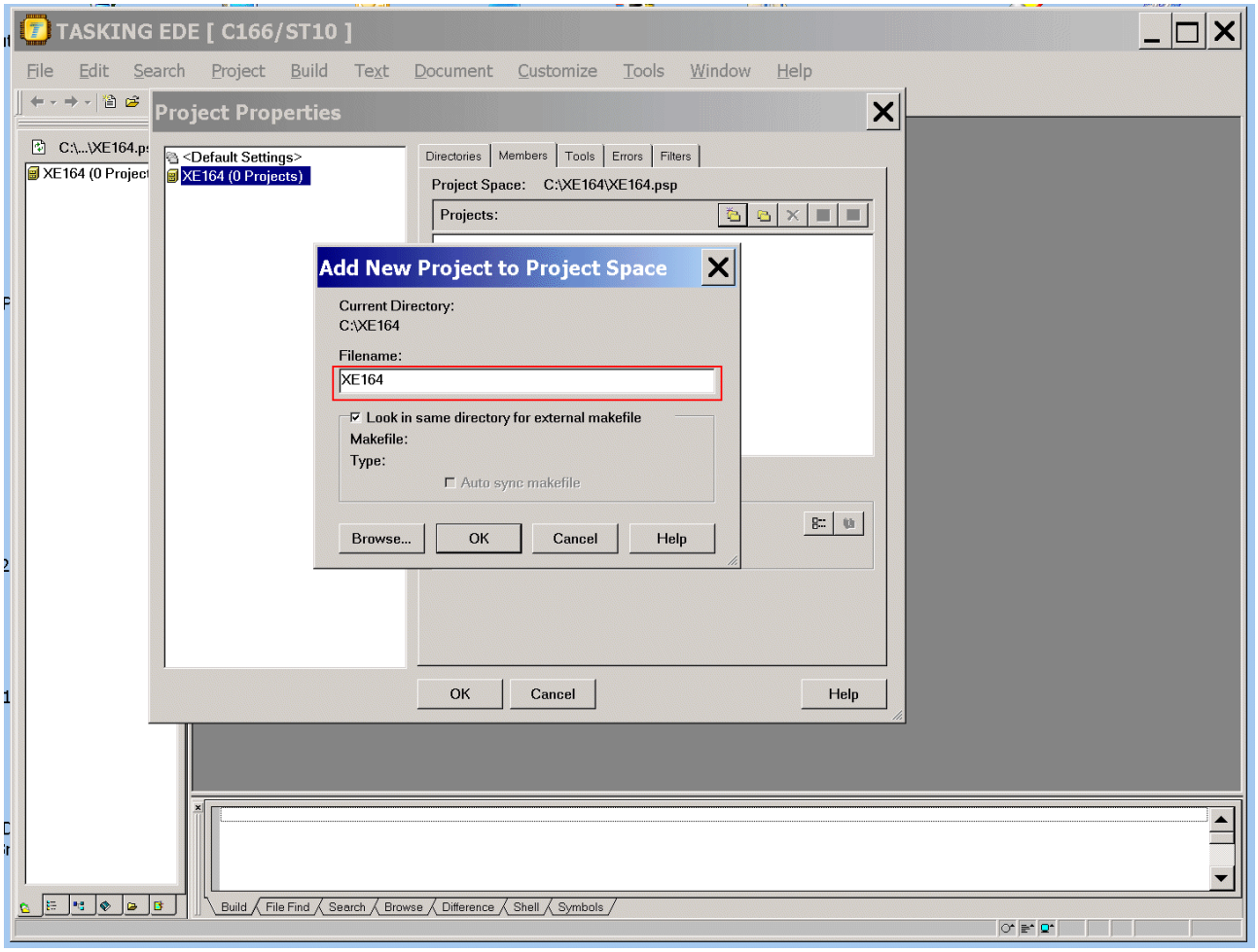
OK


TASKING EDE [C166/ST10]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\XE164.psp
XE164 (0 Projects)
Project Properties
<Default Settings>
XE164 (0 Projects)
Directories Members Tools Errors Filters
Project Space: C:\XE164\XE164.psp
Projects:
Hide projects already in project space
External Workspace:
<none>
OK Cancel Help
Build File Find Search Browse Difference Shell Symbols
Ins Line: 1 Col: 1

Click: “Add new project to project space”

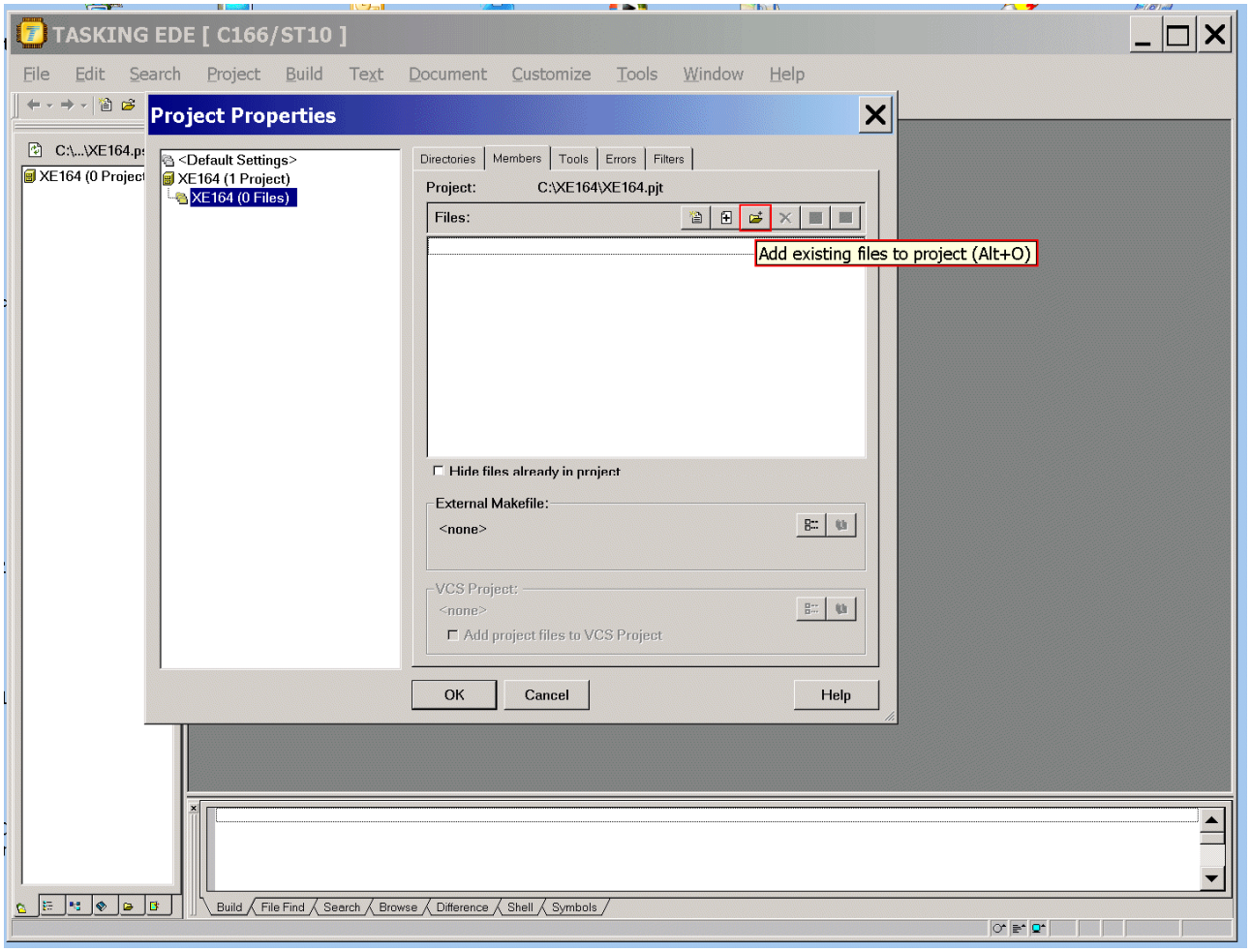


Insert XE164

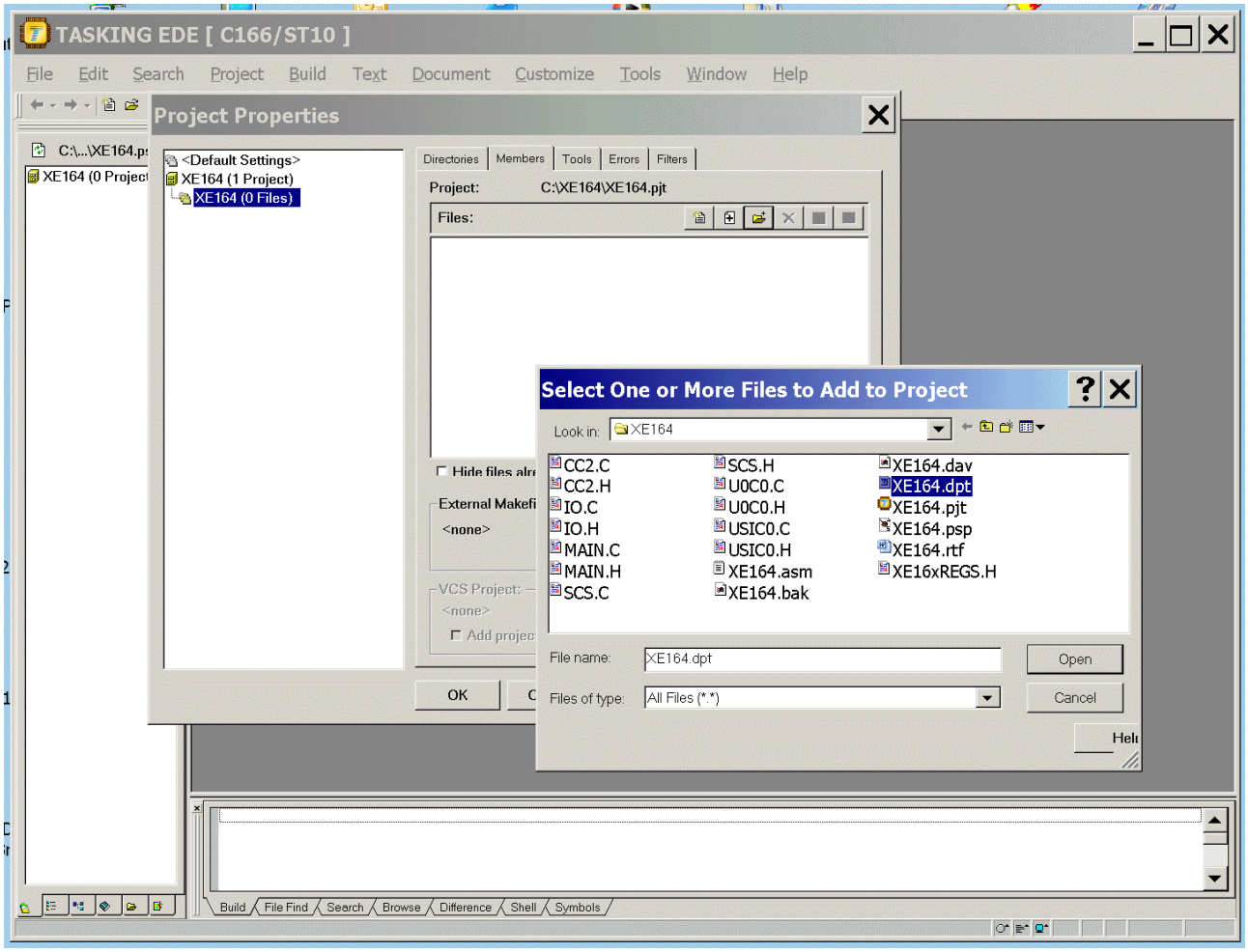


OK

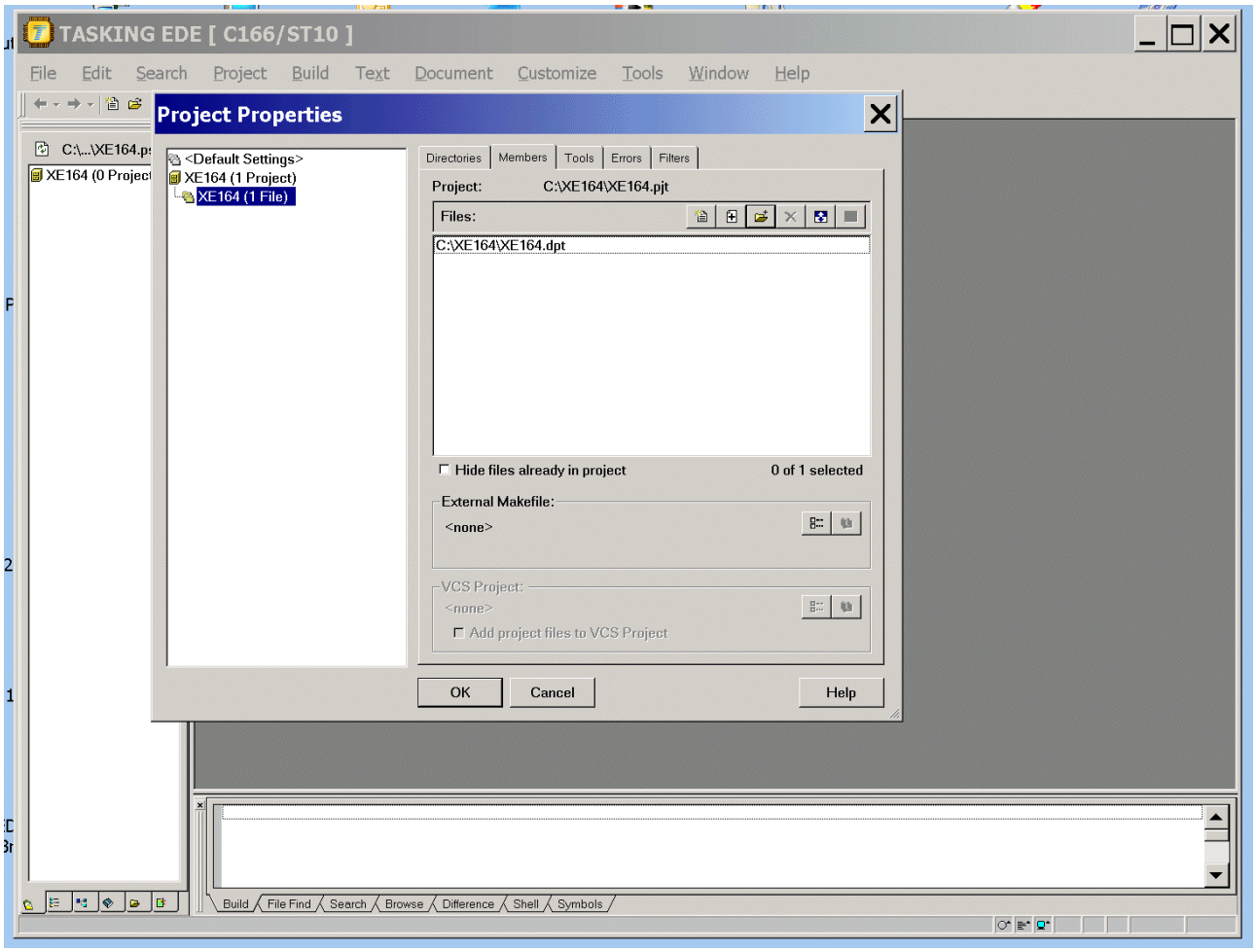
Click: “Add existing files to project”



Select XE164.dpt



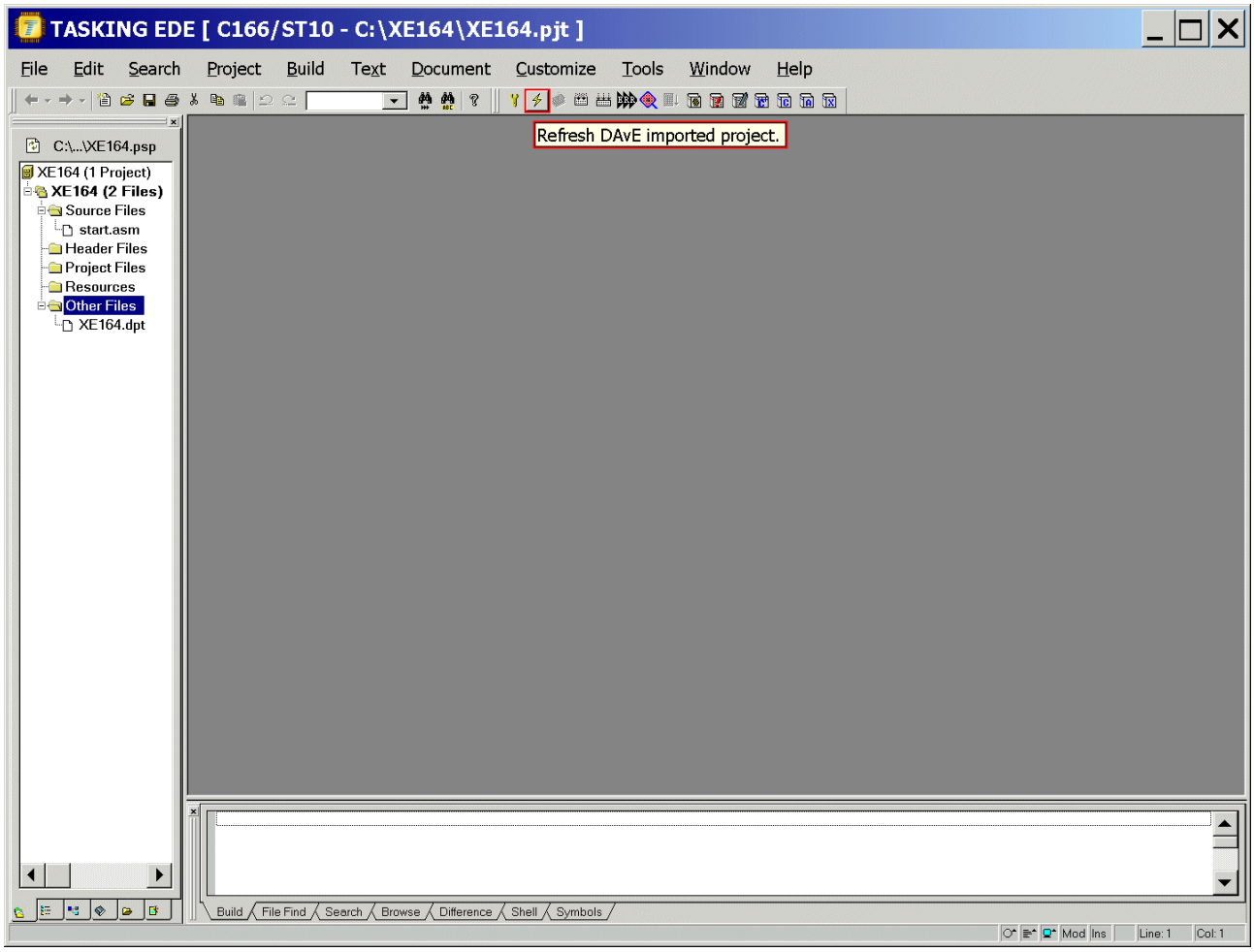
Open



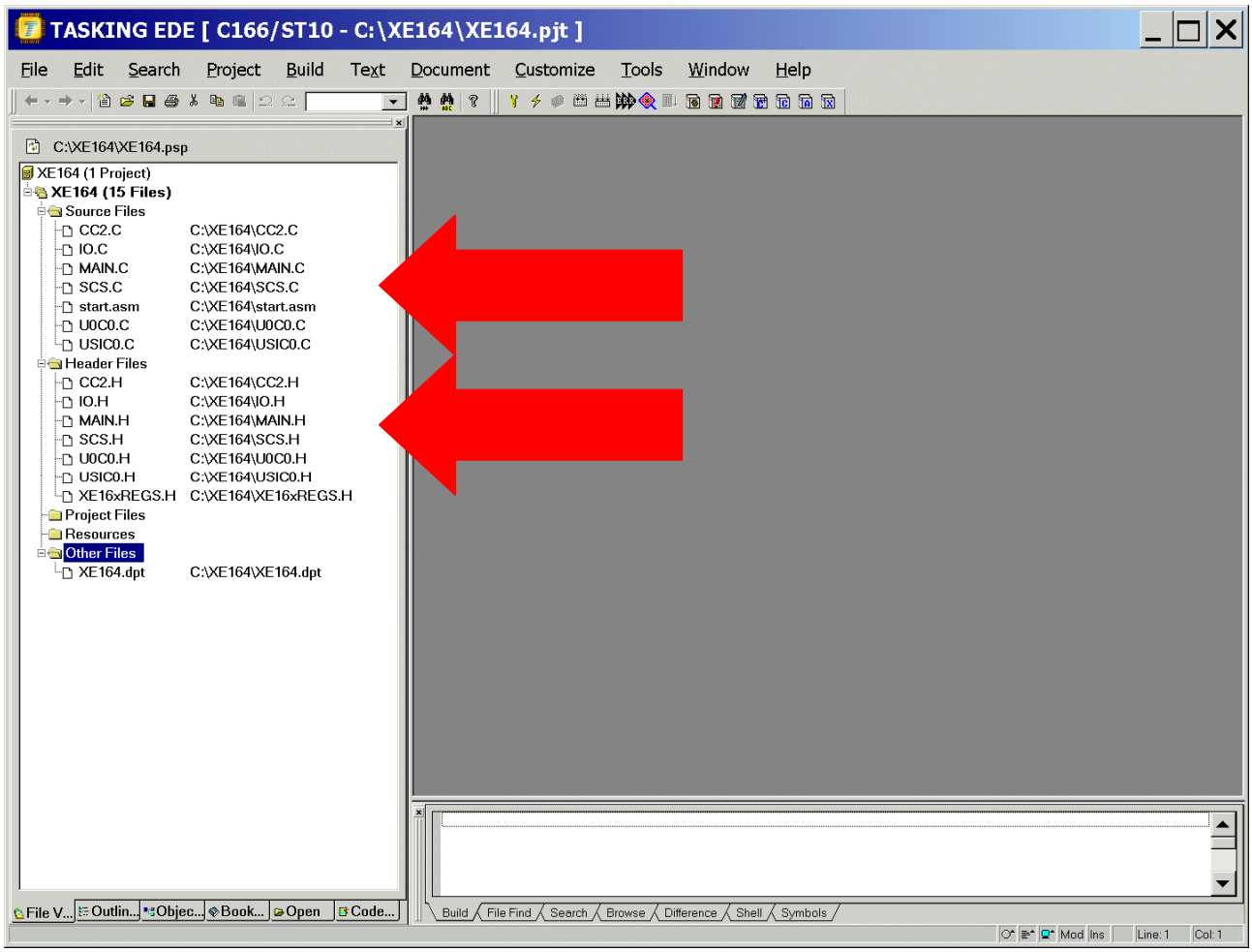
OK

Note:
This step includes the DAvE Project file.

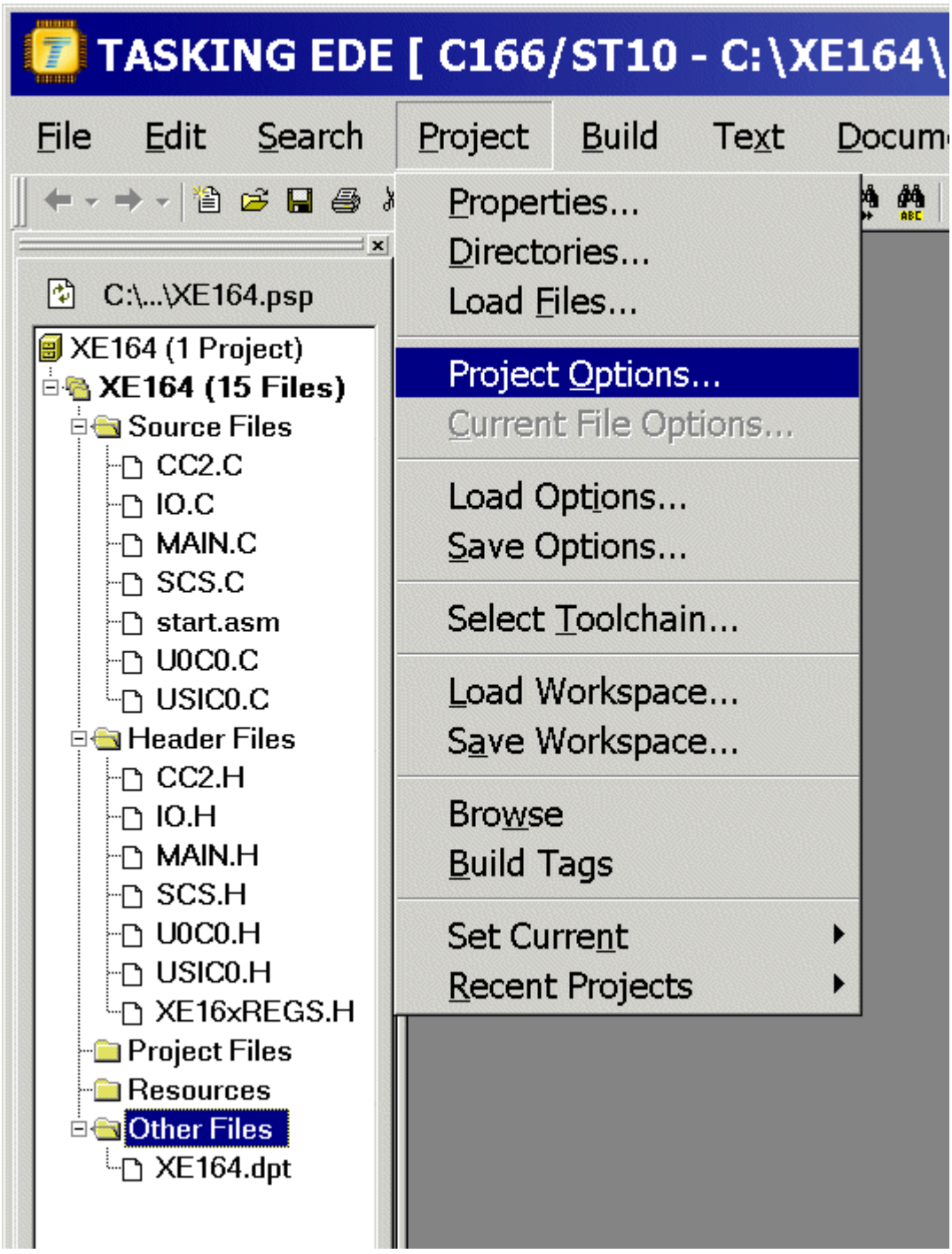
Click: “Refresh DAvE imported project”



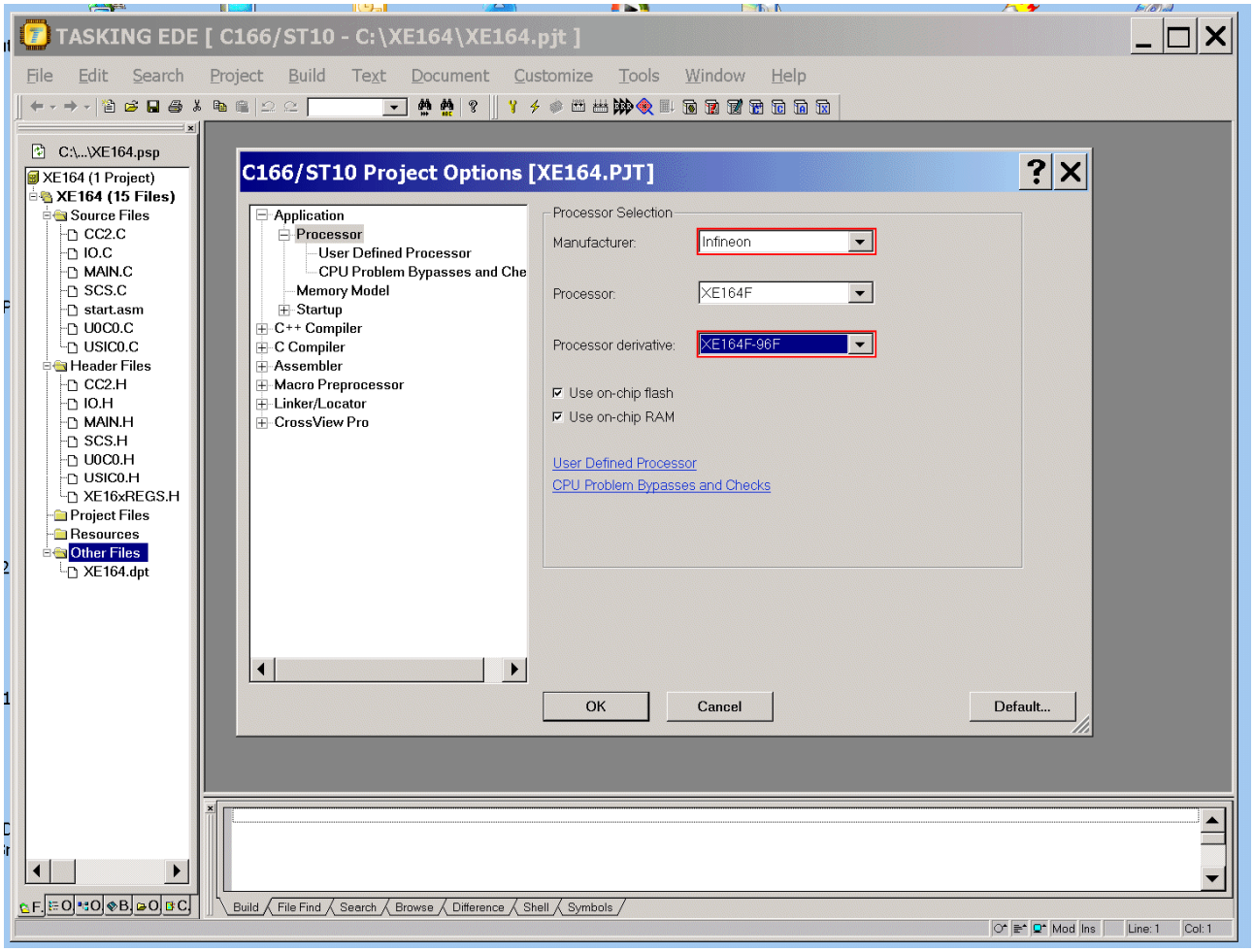
Note:
This step includes the Dave files.


TASKING EDE [C166/ST10 - C:\XE164\XE164.pjt]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\XE164\XE164.psp
XE164 (1 Project)
XE164 (15 Files)
Source Files
CC2.C C:\XE164\CC2.C
IO.C C:\XE164\IO.C
MAIN.C C:\XE164\MAIN.C
SCS.C C:\XE164\SCS.C
start.asm C:\XE164\start.asm
U0C0.C C:\XE164\U0C0.C
USIC0.C C:\XE164\USIC0.C
Header Files
CC2.H C:\XE164\CC2.H
IO.H C:\XE164\IO.H
MAIN.H C:\XE164\MAIN.H
SCS.H C:\XE164\SCS.H
U0C0.H C:\XE164\U0C0.H
USIC0.H C:\XE164\USIC0.H
XE16xREGS.H C:\XE164\XE16xREGS.H
Project Files
Resources
Other Files
XE164.dpt C:\XE164\XE164.dpt
File V... Outlin... Objec... Book... Open Code...
Build File Find Search Browse Difference Shell Symbols
Mod Ins Line: 1 Col: 1

Project – Project Options…

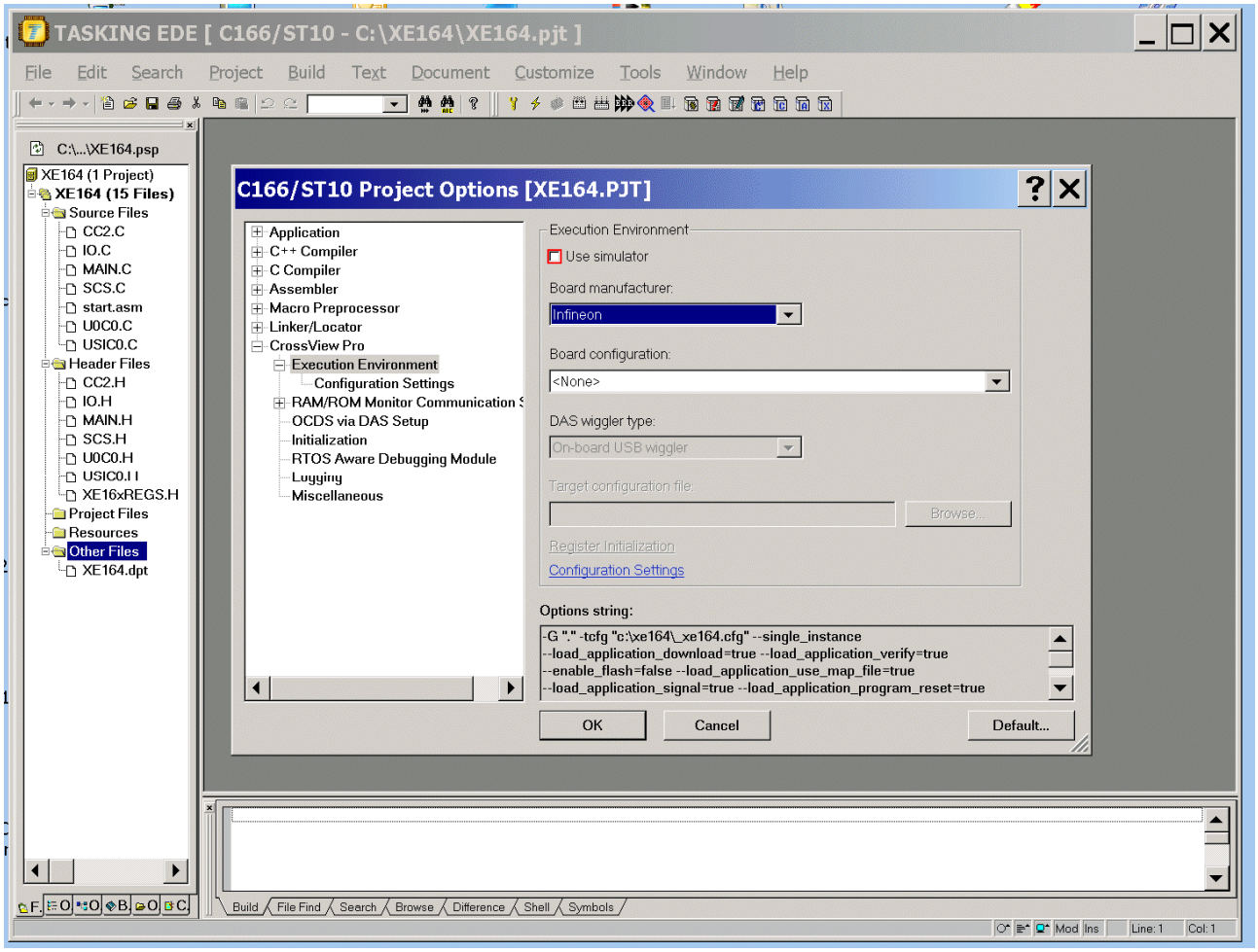


Application: Processor: Processor Selection: Manufacturer: select Infineon
Application: Processor: Processor Selection: Processor derivative: select XE164F-96F

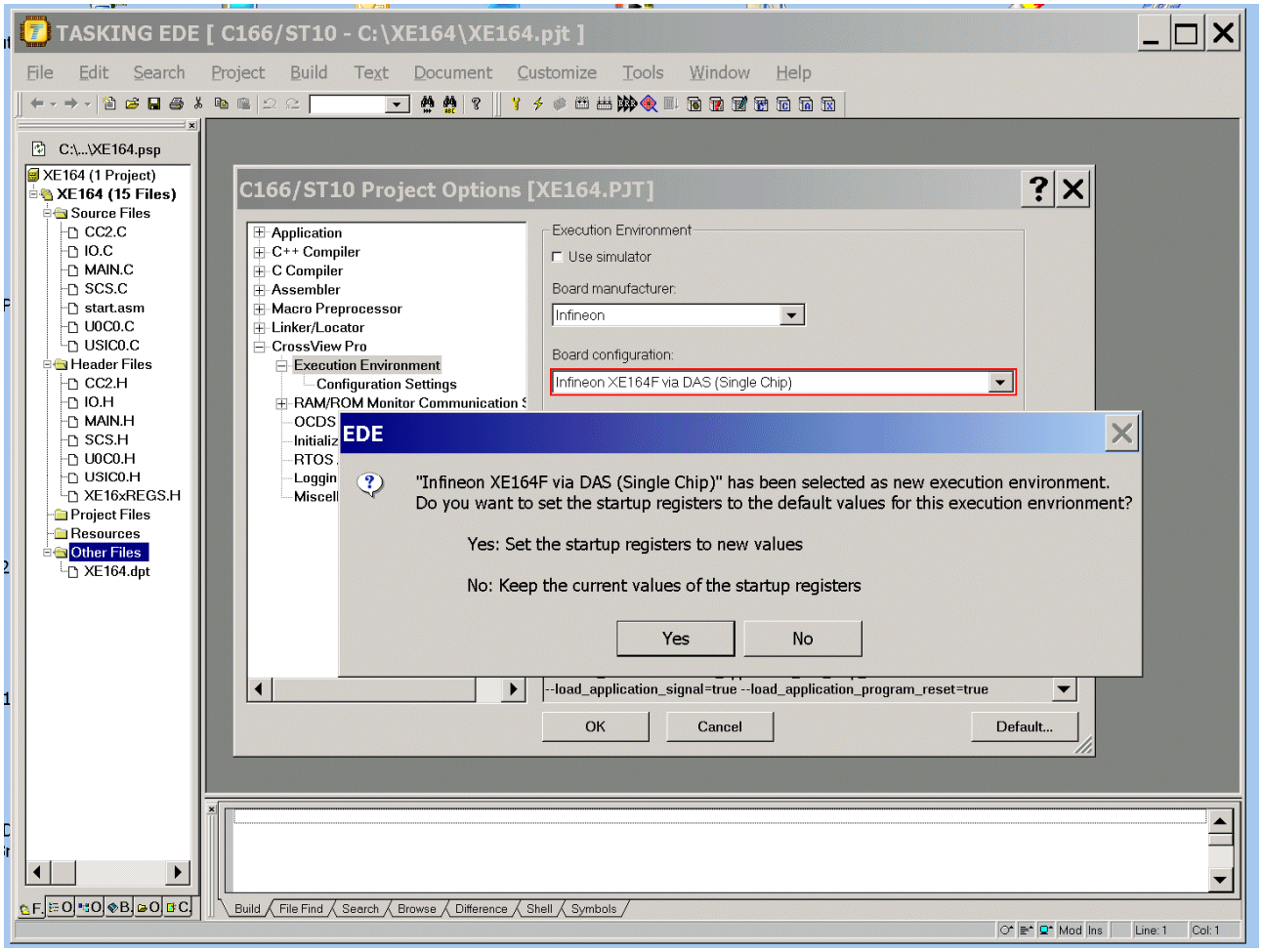


CrossView Pro:
Execution Environment: Execution Environment: click to unselect ☐ Use simulator

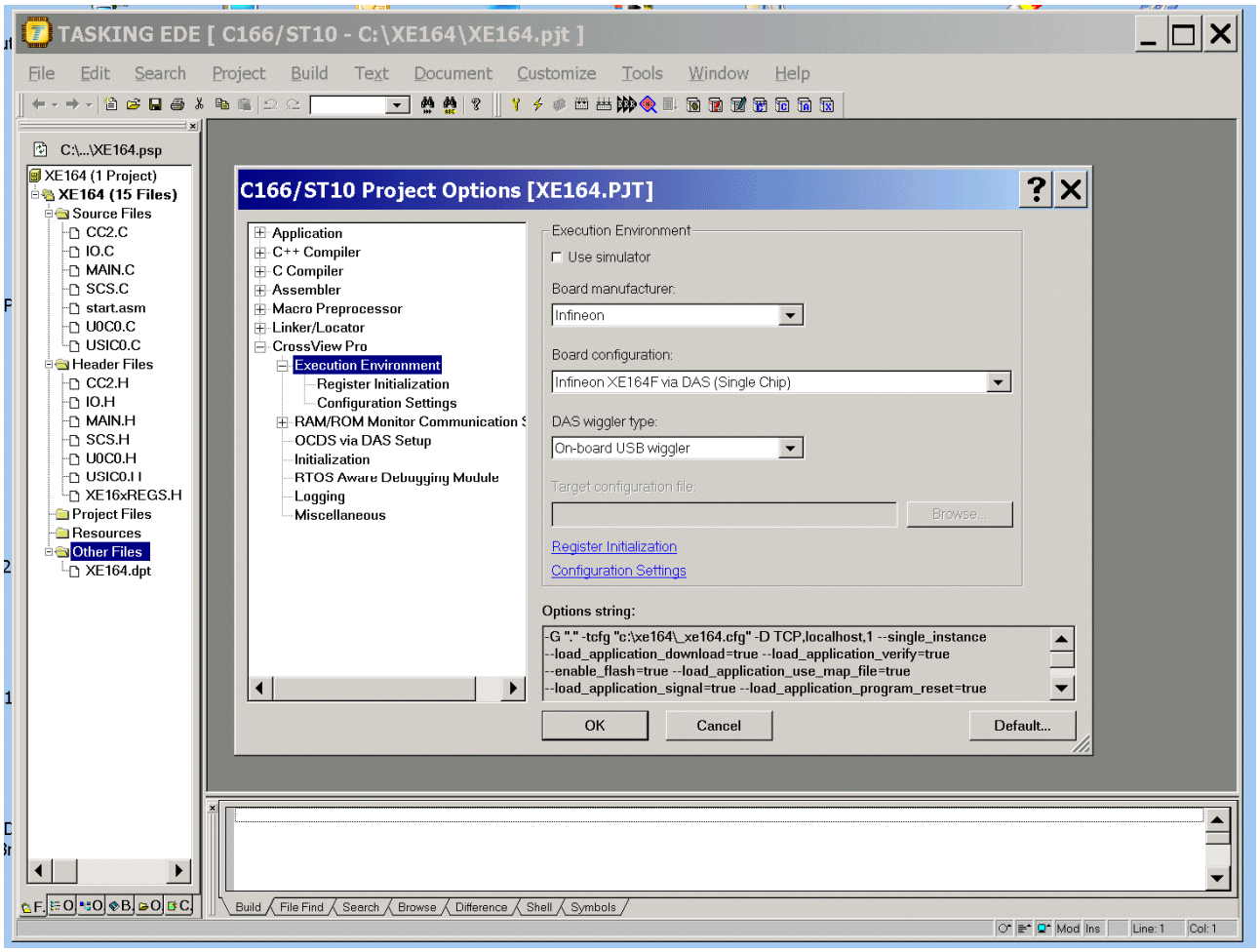
CrossView Pro:
Execution Environment: Board manufacturer: select Infineon



CrossView Pro:
Execution Environment: Board configuration: select Infineon XE164F via DAS (Single Chip)

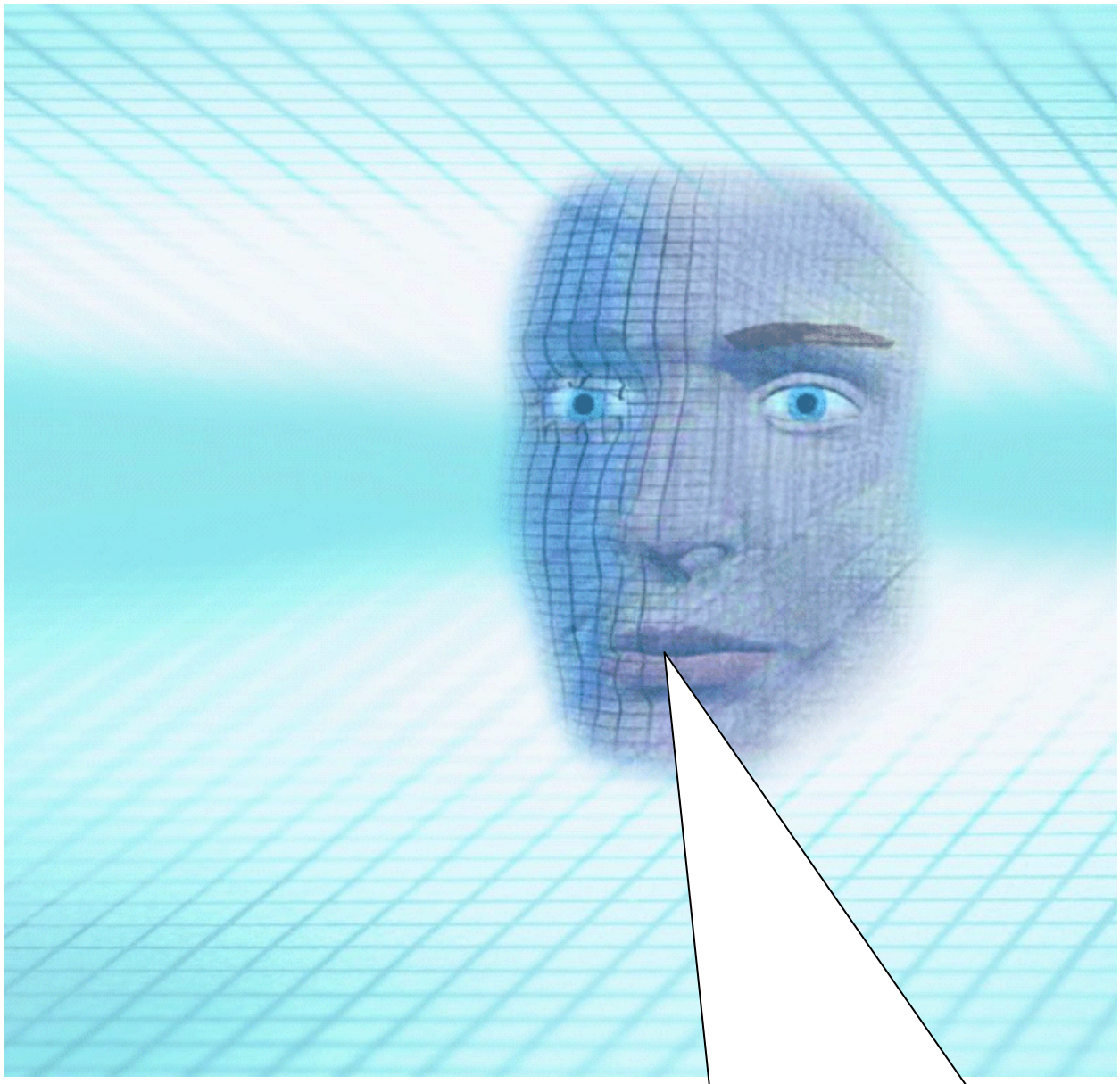


Click Yes


TASKING EDE [C166/ST10 - C:\XE164\XE164.pjt]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\XE164.psp
XE164 (1 Project)
XE164 (15 Files)
Source Files
CC2.C
IO.C
MAIN.C
SCS.C
start.asm
U0C0.C
USIC0.C
Header Files
CC2.H
IO.H
MAIN.H
SCS.H
U0C0.H
XE16xREGS.H
Project Files
Resources
Other Files
XE164.dpt
C166/ST10 Project Options [XE164.PJT]
Application
C++ Compiler
C Compiler
Assembler
Macro Preprocessor
Linker/Locator
CrossView Pro
Execution Environment
Register Initialization
Configuration Settings
RAM/ROM Monitor Communication
OCDS via DAS Setup
Initialization
RTOS Aware Debugging Module
Logging
Miscellaneous
Execution Environment
Use simulator
Board manufacturer:
Infineon
Board configuration:
Infineon XE164F via DAS (Single Chip)
DAS wiggler type:
On-board USB wiggler
Target configuration file:
Browse...
Register Initialization
Configuration Settings
Options string:
-G "." -tcfg "c:\xe164_xe164.cfg" -D TCP,localhost,1 --single_instance
--load_application_download=true --load_application_verify=true
--enable_flash=true --load_application_use_map_file=true
--load_application_signal=true --load_application_program_reset=true
OK
Cancel
Default...
Build
File Find
Search
Browse
Difference
Shell
Symbols
Mod
Ins
Line: 1
Col: 1

OK

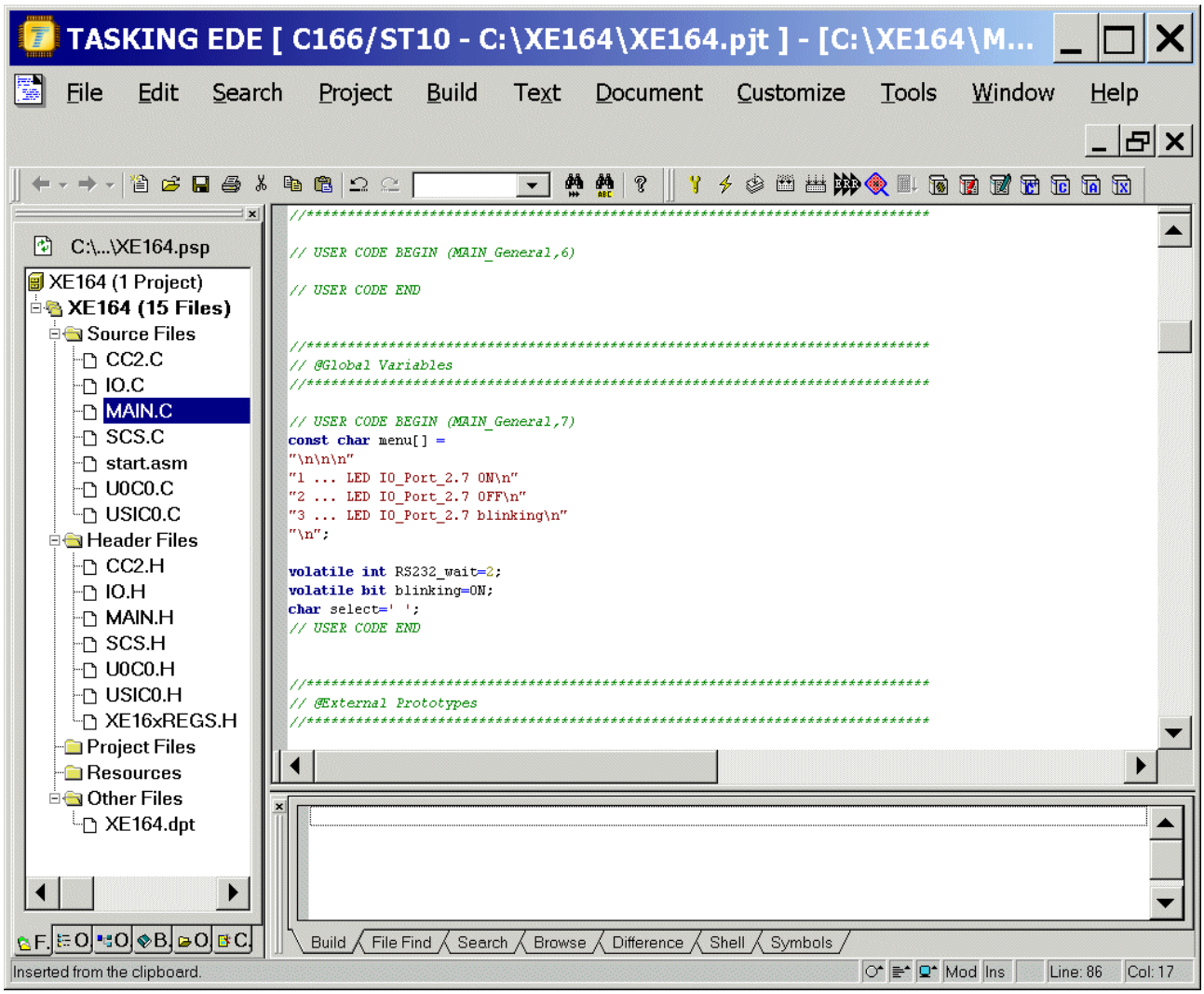
Insert your application specific program:



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!

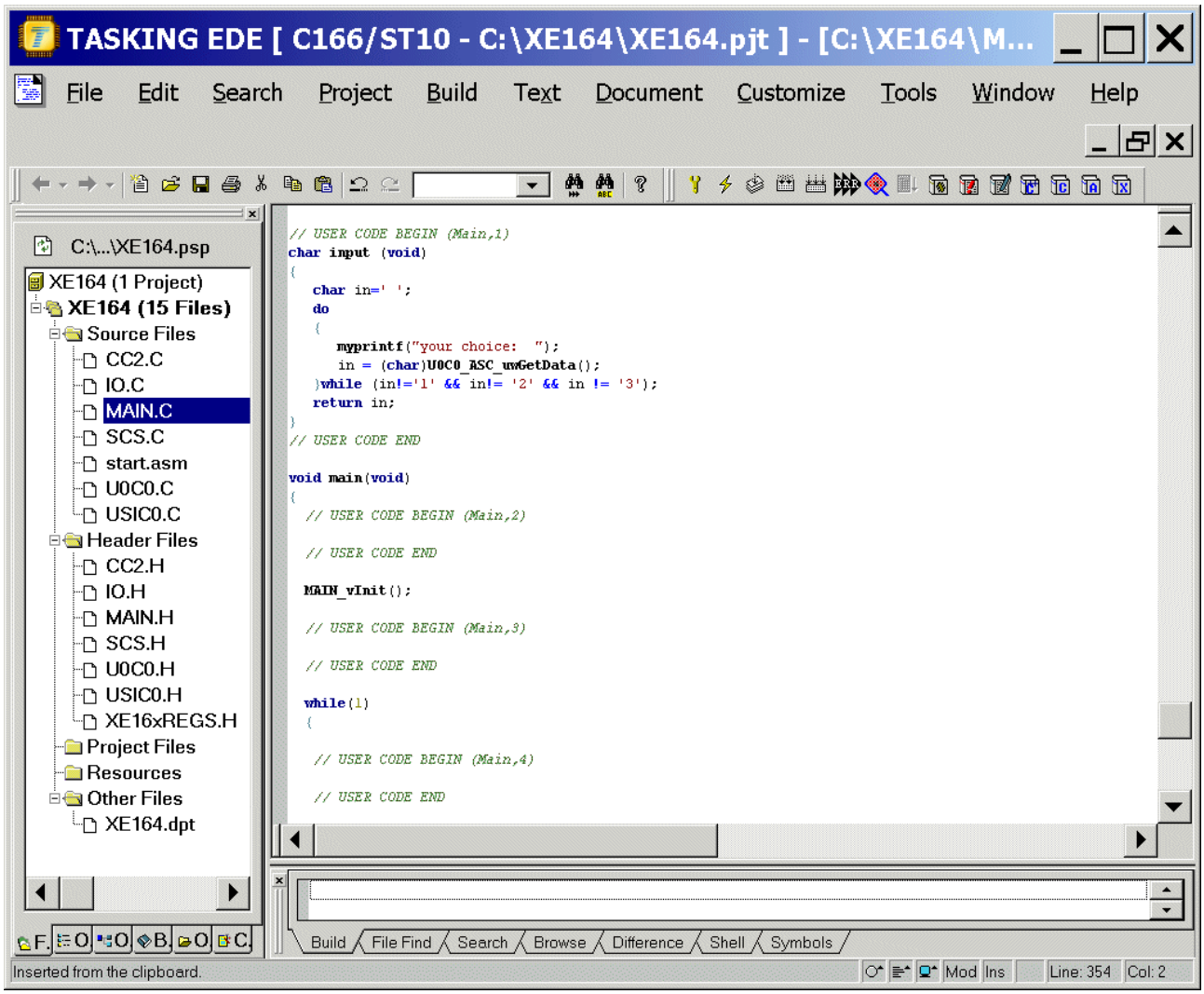
Double click MAIN.C and insert Global Variables:

```
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=' ';
```



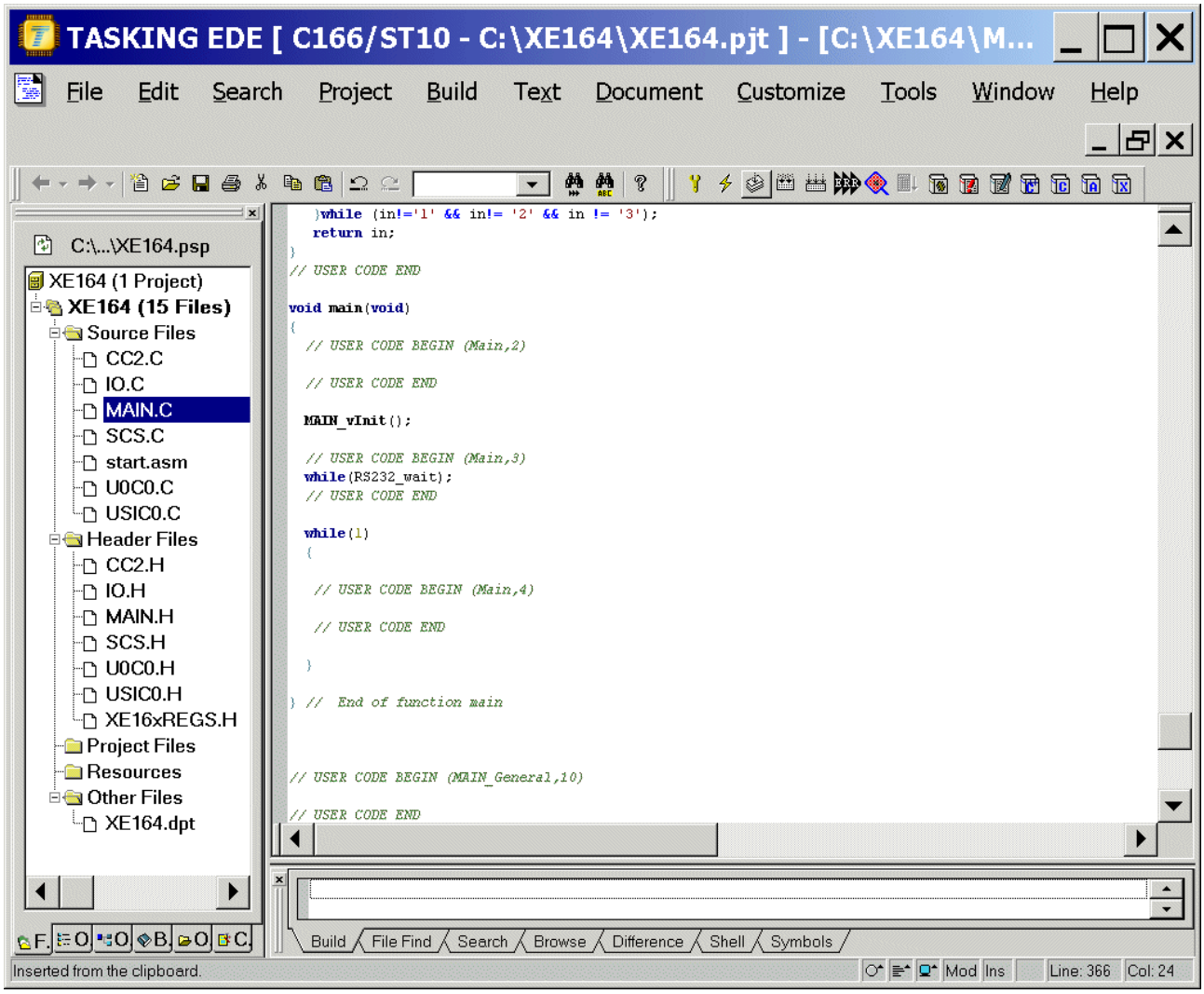
Double click MAIN.C and insert the function input():

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



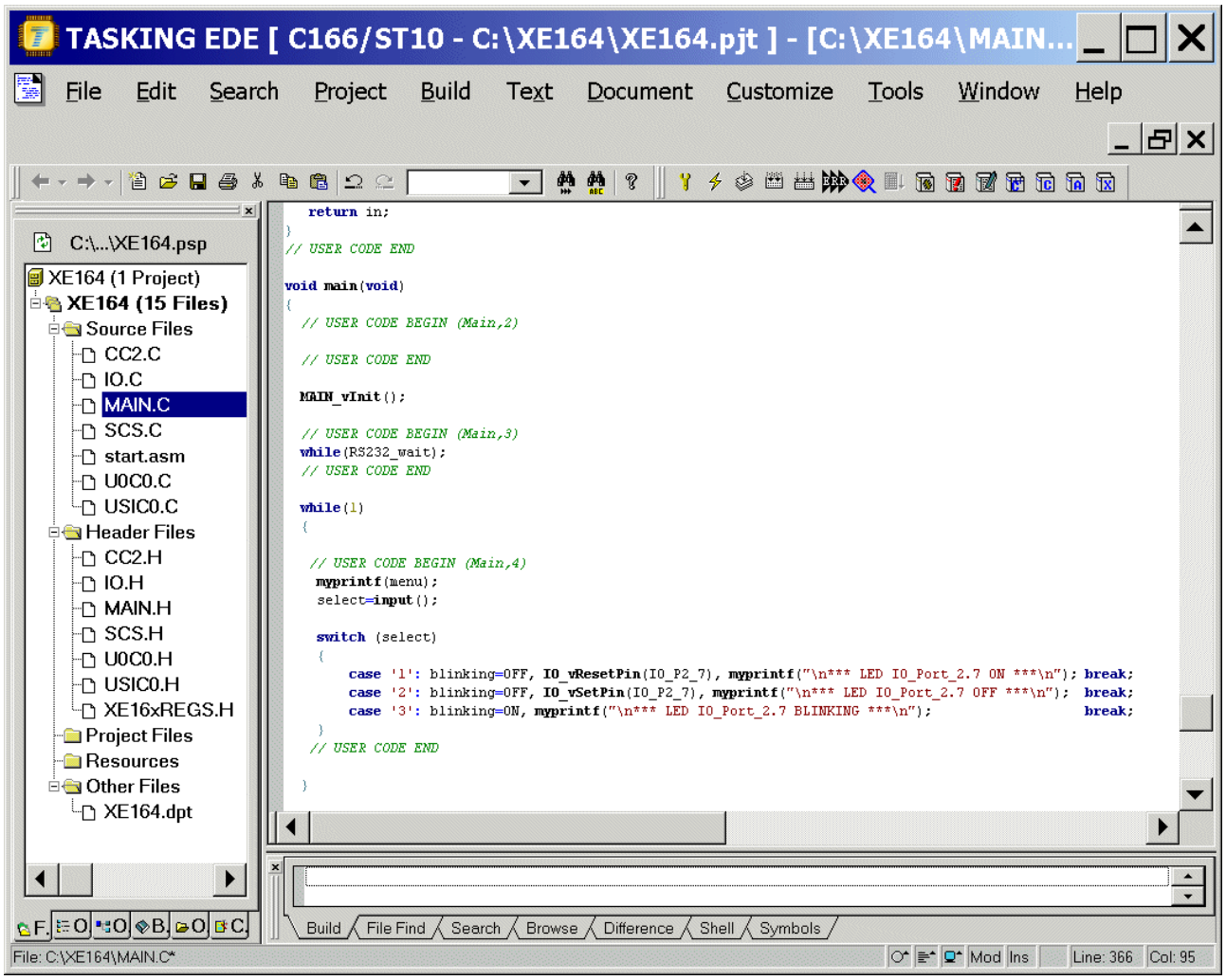
Double click MAIN.C and insert the following code in the main function:

```
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 ($E9C0_H$+4*n) Reset Value: $XXXX_H$

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
PSx (x = 0-7)	x	w	**Port Set Bit x** Setting this bit sets or toggles the corresponding bit in the port output register Pn_OUT (see Table 7-4). On a read access, this bit returns 0.
PCx (x = 0-7)	x + 8	w	**Port Clear Bit x** Setting this bit clears or toggles the corresponding bit in the port output register Pn_OUT. (see Table 7-4). On a read access, this bit returns 0.

Function of the PCx and PSx bit fields

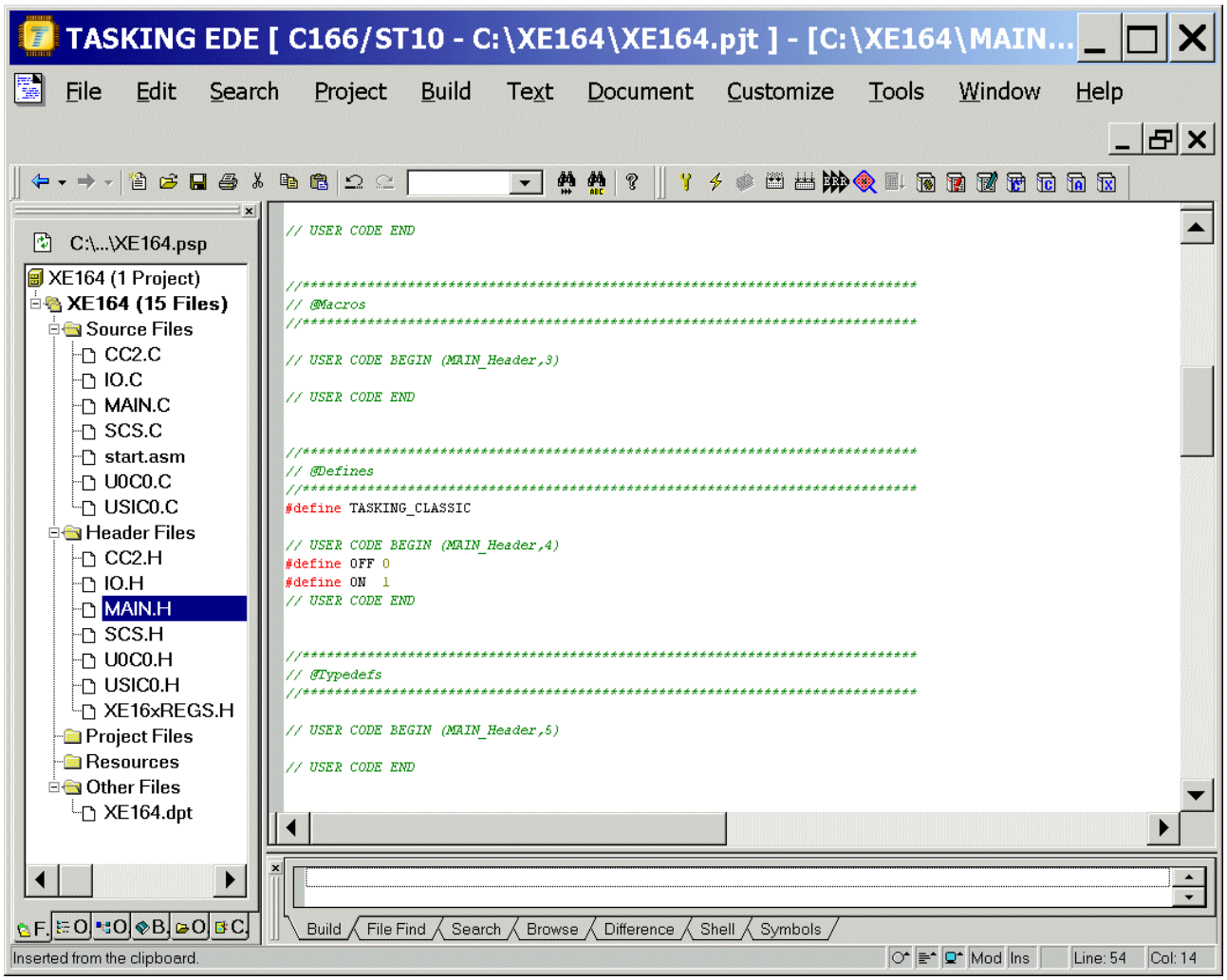
Table 7-4 Function of the Bits PCx and PSx

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.

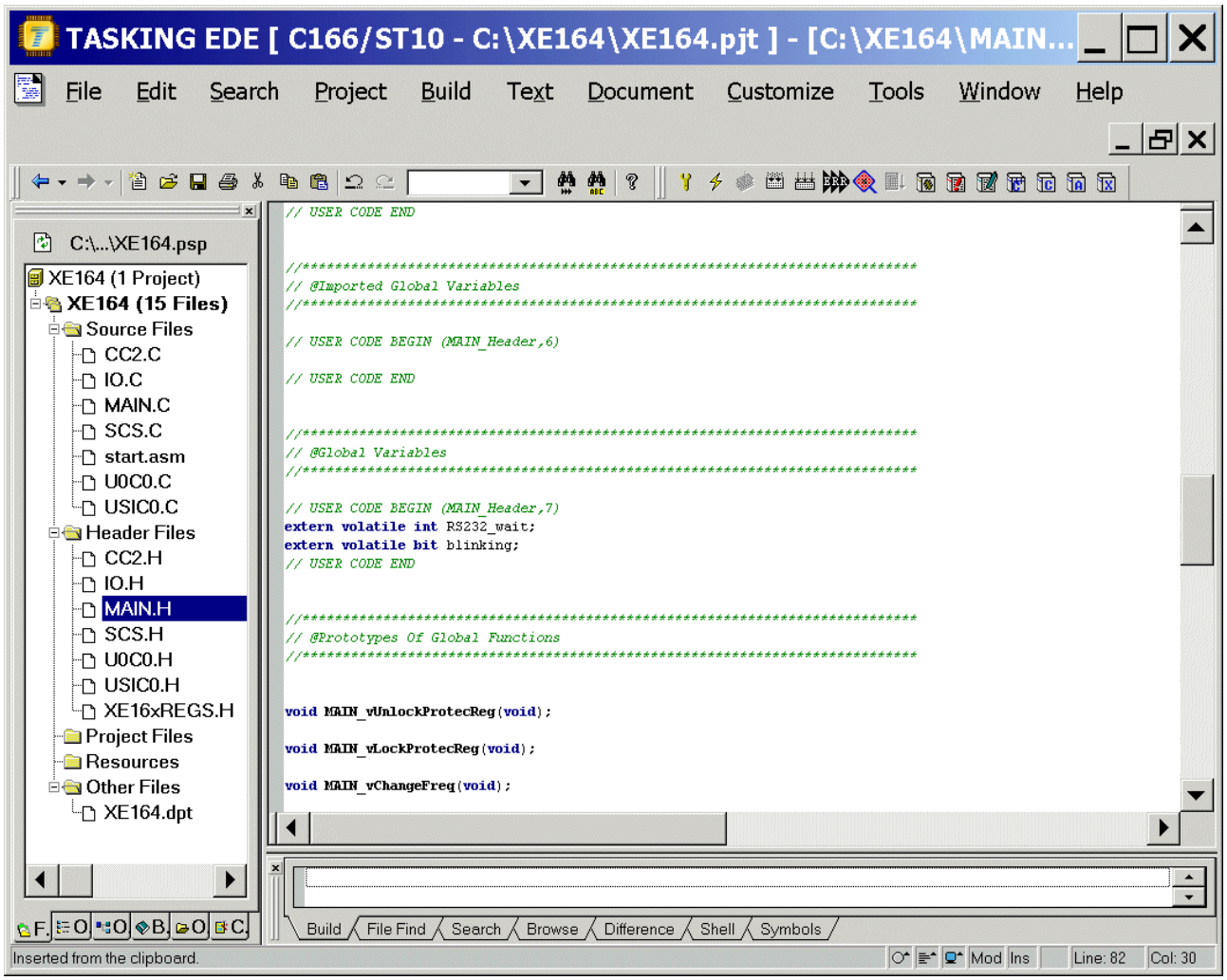
Double click Main.h and insert the following Defines:

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



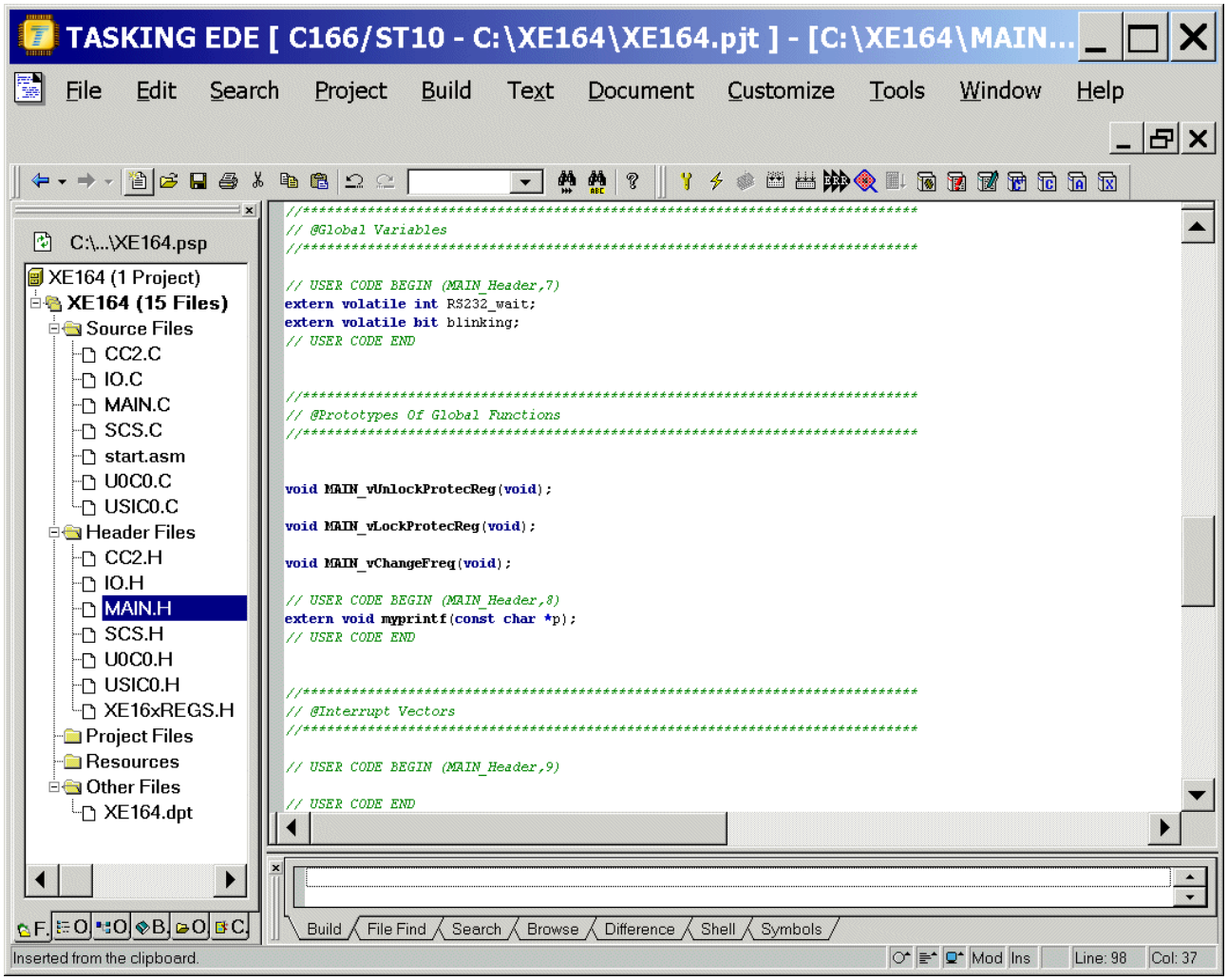
Double click Main.h and insert extern declarations “Global Variables”:

```
extern volatile int RS232_wait;
extern volatile bit blinking;
```



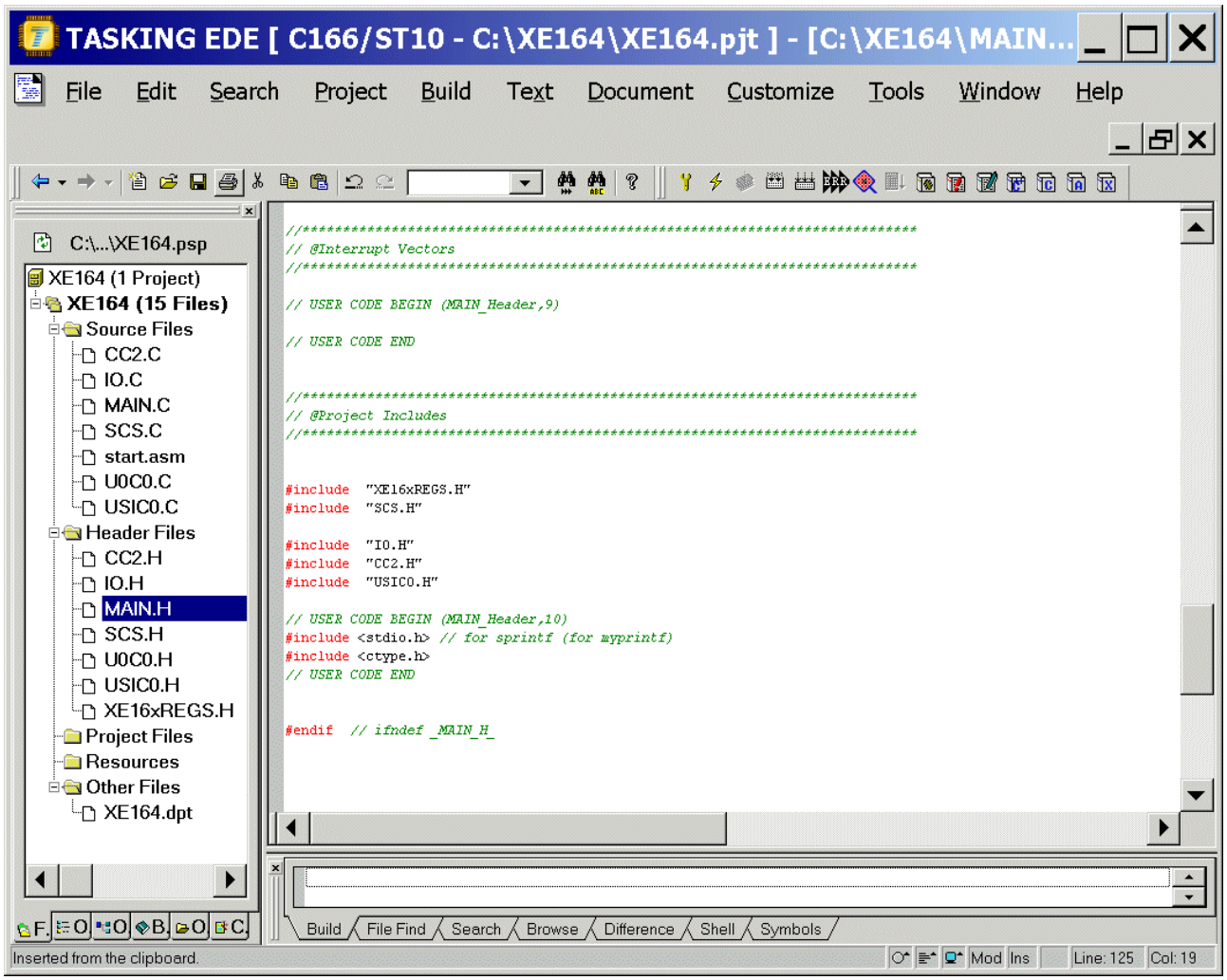
Double click Main.h and insert extern declarations “Global Functions”:

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

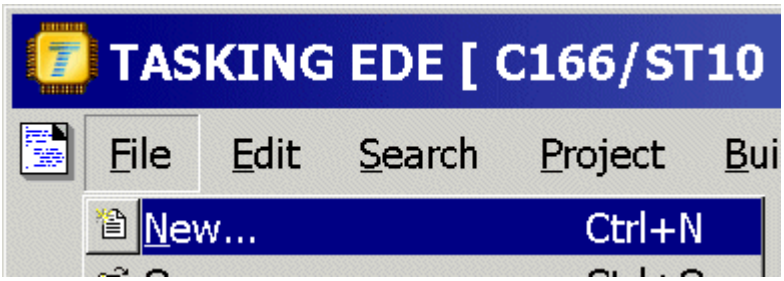


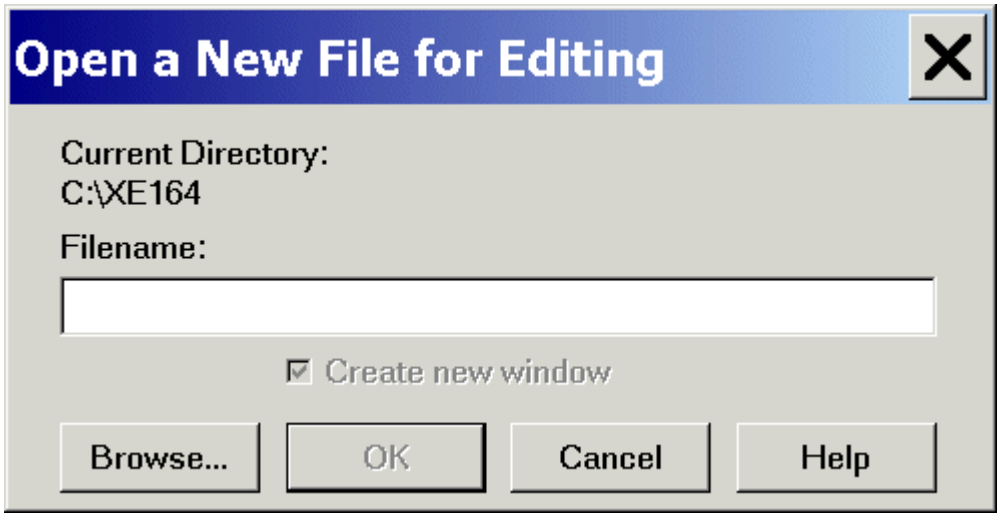
Double click Main.h and insert include files:

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

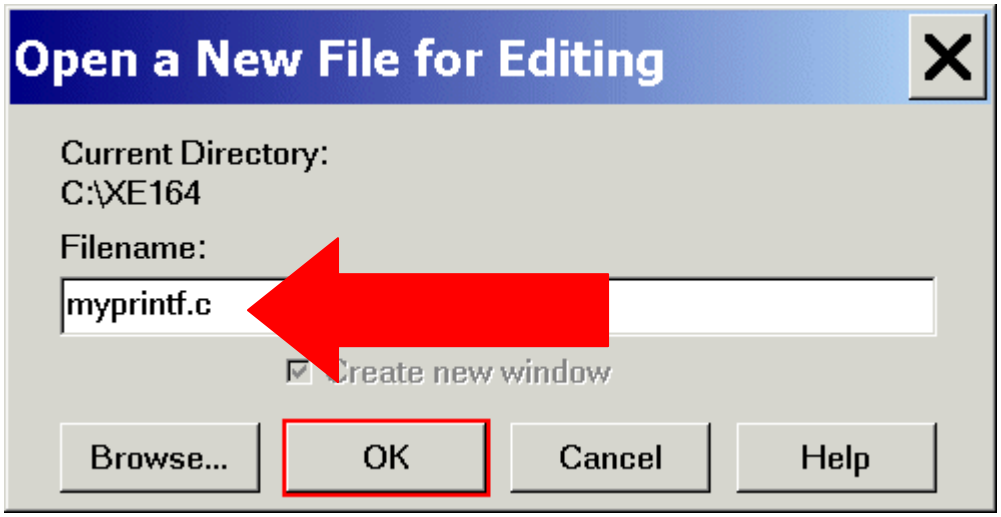


File – New





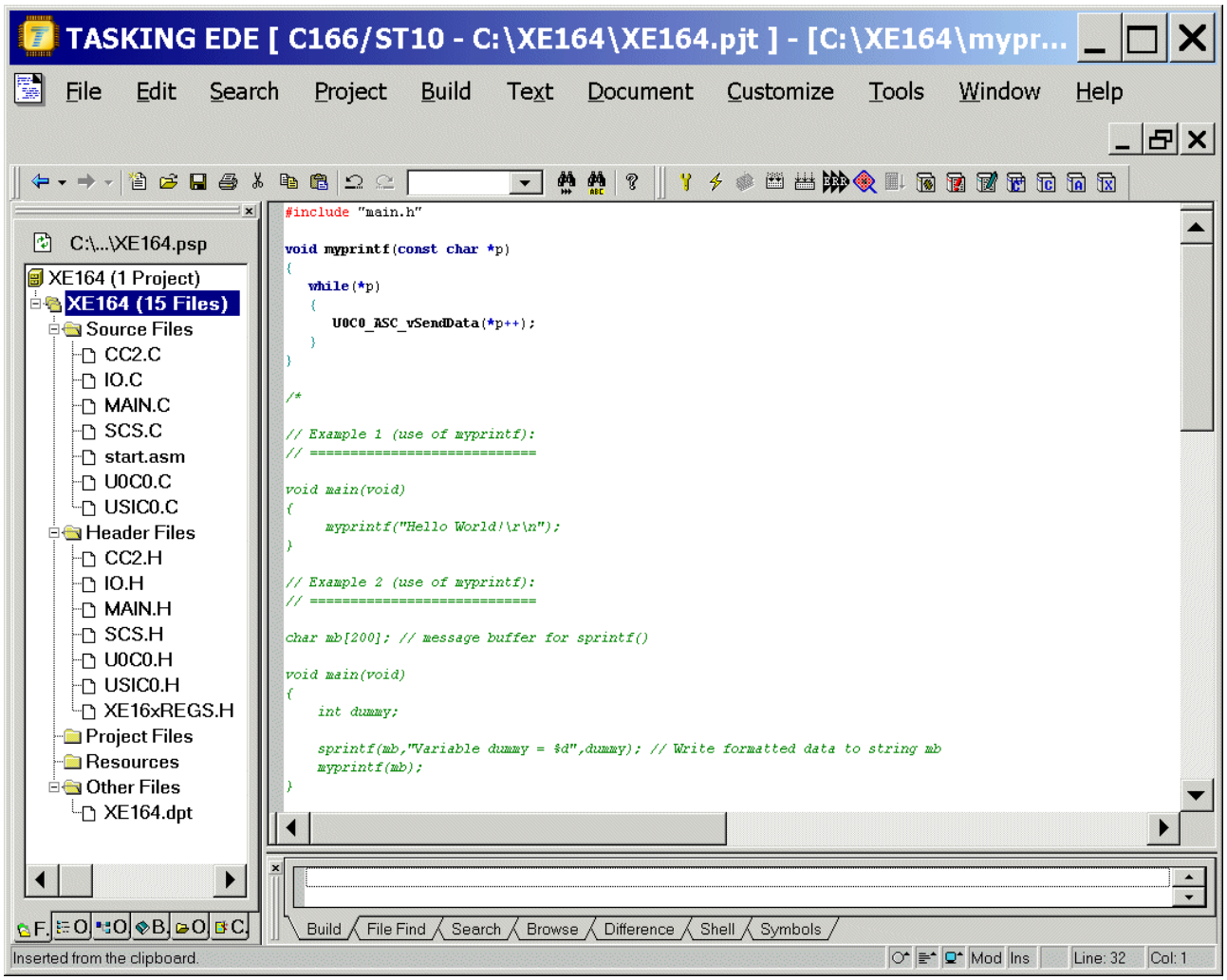
Insert: myprintf.c



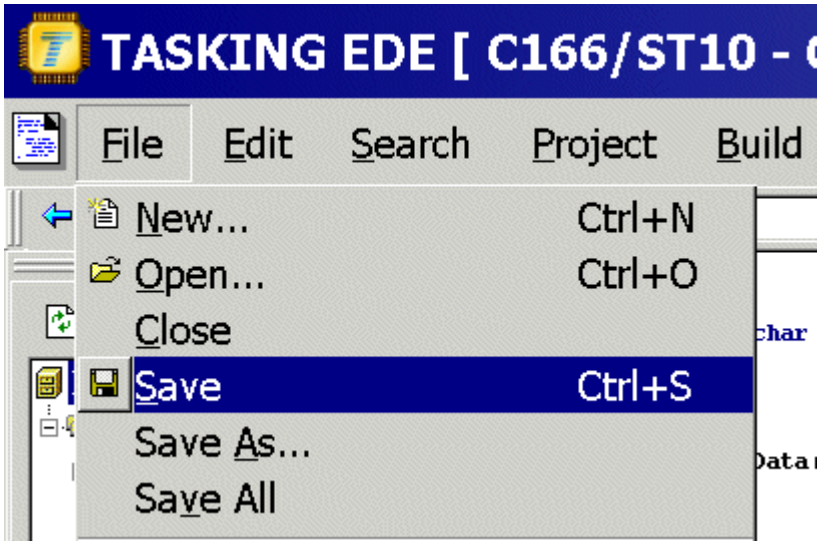
Click OK

Insert:

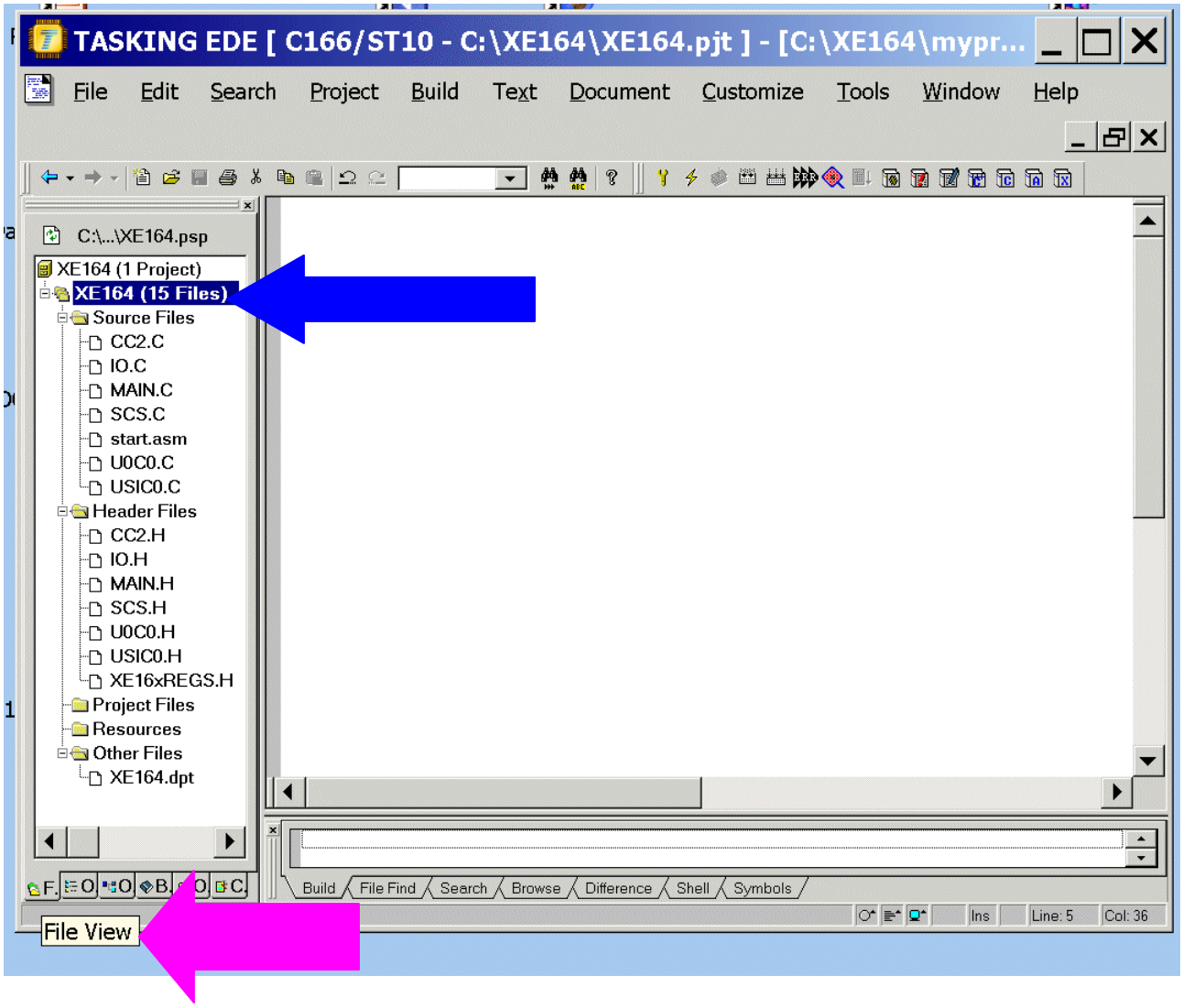
```
#include "main.h"

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

/*

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

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

*/
```


TASKING EDE [C166/ST10 - C:\XE164\XE164.pjt] - [C:\XE164\mypr...
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\XE164.psp
XE164 (1 Project)
XE164 (15 Files)
Source Files
CC2.C
IO.C
MAIN.C
SCS.C
start.asm
U0C0.C
USIC0.C
Header Files
CC2.H
IO.H
MAIN.H
SCS.H
U0C0.H
USIC0.H
XE16xREGS.H
Project Files
Resources
Other Files
XE164.dpt
#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!\r\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);
}
Build File Find Search Browse Difference Shell Symbols
Inserted from the clipboard.
Mod Ins Line: 32 Col: 1

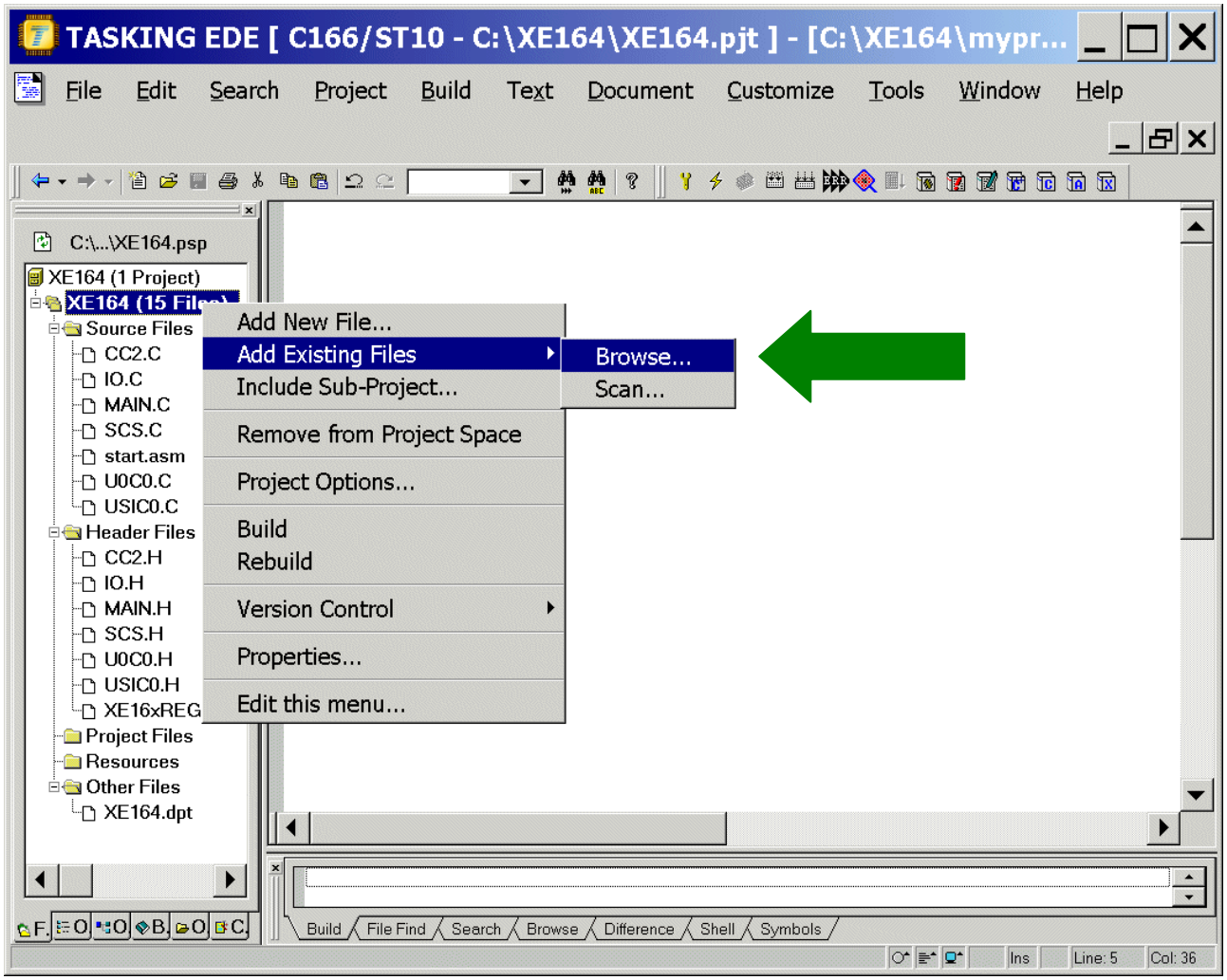
File – Save


TASKING EDE [C166/ST10 -
File
Edit
Search
Project
Build
New... Ctrl+N
Open... Ctrl+O
Close
Save Ctrl+S
Save As...
Save All

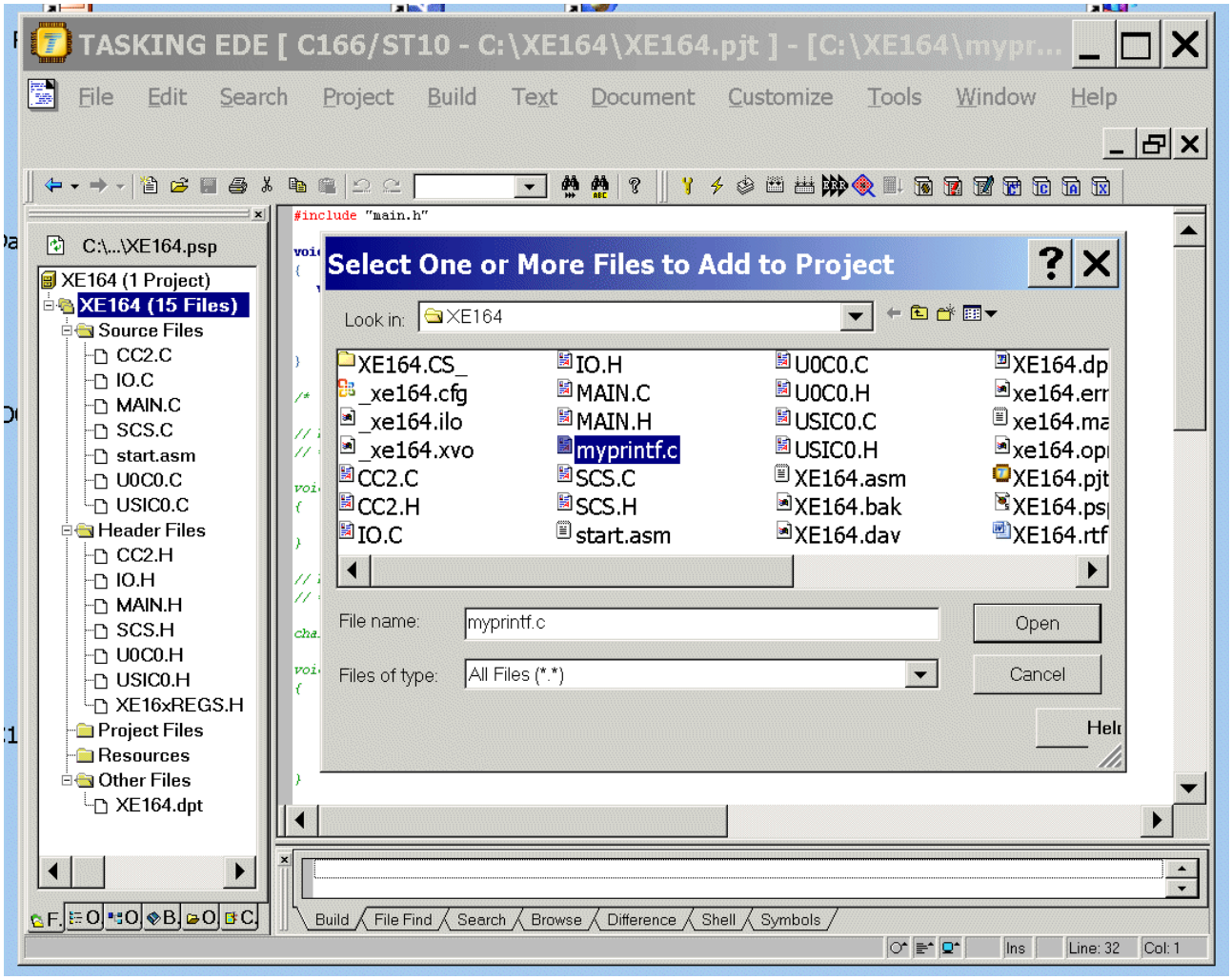
Mouse position: XE164 (15 Files), File View Window: click right mouse button



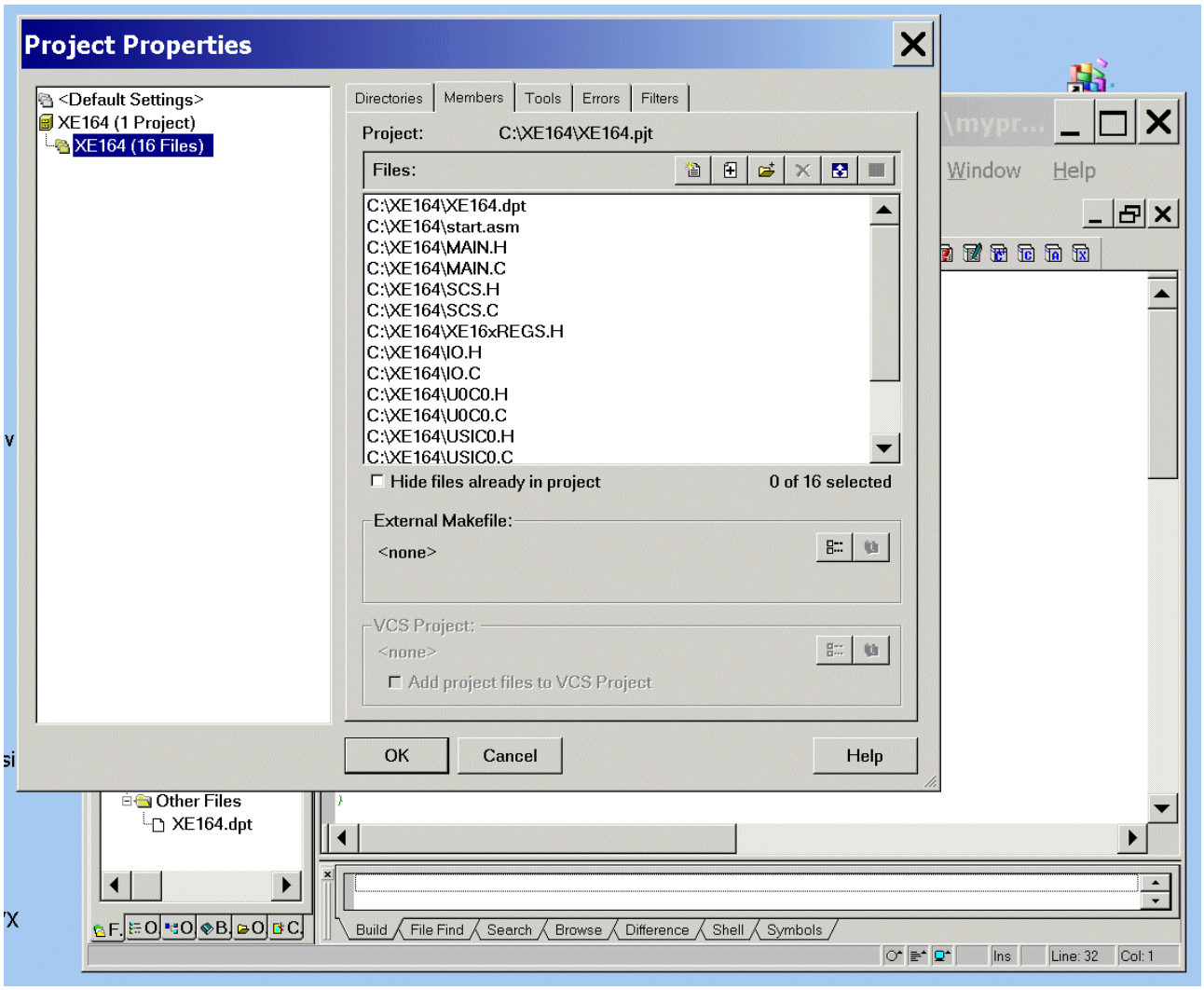
Mouse position: XE164 (15 Files), File View Window: click right mouse button
click Add Existing Files – Browse…



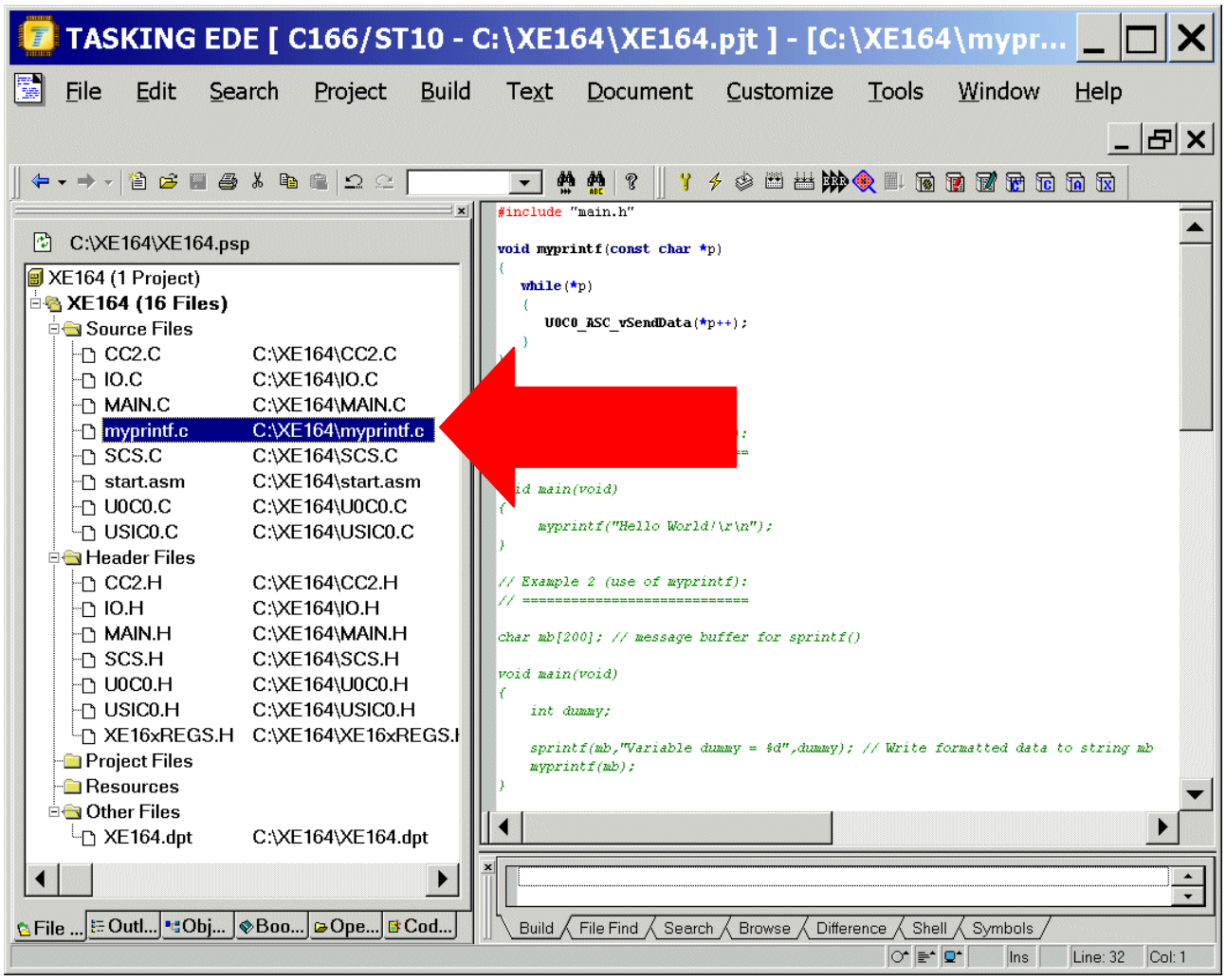
Click myprintf.c



Click Open

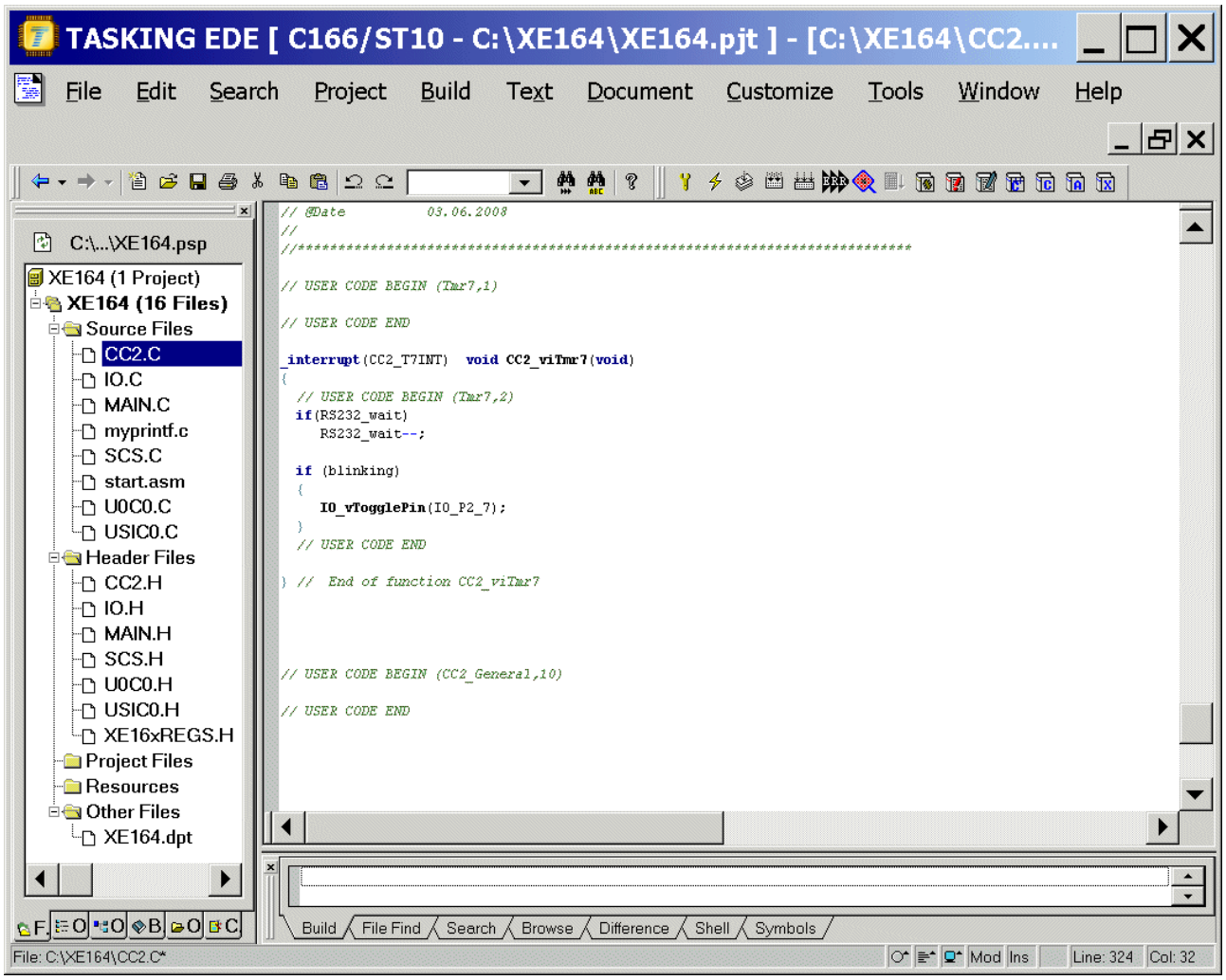

Project Properties
<Default Settings>
XE164 (1 Project)
XE164 (16 Files)
Directories
Members
Tools
Errors
Filters
Project:
C:\XE164\XE164.pjt
Files:
C:\XE164\XE164.dpt
C:\XE164\start.asm
C:\XE164\MAIN.H
C:\XE164\MAIN.C
C:\XE164\SCS.H
C:\XE164\SCS.C
C:\XE164\XE16xREGS.H
C:\XE164\IO.H
C:\XE164\IO.C
C:\XE164\U0C0.H
C:\XE164\U0C0.C
C:\XE164\USIC0.H
C:\XE164\USIC0.C
Hide files already in project
0 of 16 selected
External Makefile:
<none>
VCS Project:
<none>
Add project files to VCS Project
OK
Cancel
Help
Other Files
XE164.dpt
Window
Help
Build
File Find
Search
Browse
Difference
Shell
Symbols
Ins
Line: 32
Col: 1

Click OK

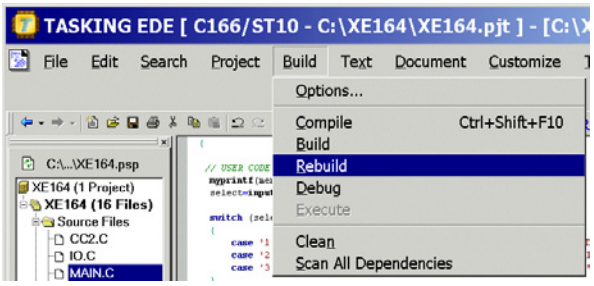
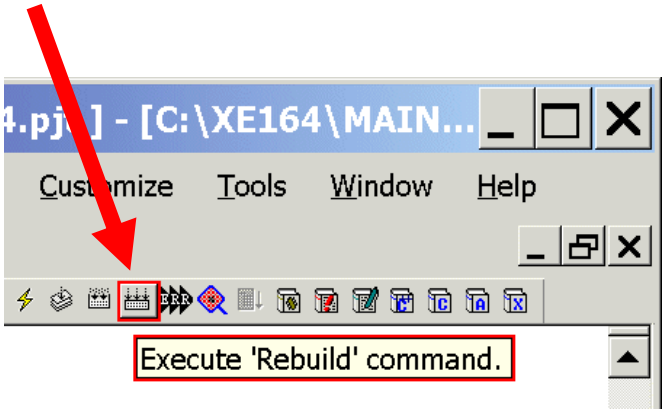

TASKING EDE [C166/ST10 - C:\XE164\XE164.pjt] - [C:\XE164\mypr...
File Edit Search Project Build Text Document Customize Tools Window Help
C:\XE164\XE164.psp
XE164 (1 Project)
XE164 (16 Files)
Source Files
CC2.C C:\XE164\CC2.C
IO.C C:\XE164\IO.C
MAIN.C C:\XE164\MAIN.C
myprintf.c C:\XE164\myprintf.c
SCS.C C:\XE164\SCS.C
start.asm C:\XE164\start.asm
U0C0.C C:\XE164\U0C0.C
USIC0.C C:\XE164\USIC0.C
Header Files
CC2.H C:\XE164\CC2.H
IO.H C:\XE164\IO.H
MAIN.H C:\XE164\MAIN.H
SCS.H C:\XE164\SCS.H
U0C0.H C:\XE164\U0C0.H
USIC0.H C:\XE164\USIC0.H
XE16xREGS.H C:\XE164\XE16xREGS.H
Project Files
Resources
Other Files
XE164.dpt C:\XE164\XE164.dpt
#include "main.h"
void myprintf(const char *p)
{
while(*p)
{
U0C0_ASC_vSendData(*p++);
}
}
id main(void)
{
myprintf("Hello World!\r\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);
}
File ... Outl... Obj... Boo... Ope... Cod...
Build File Find Search Browse Difference Shell Symbols
Ins Line: 32 Col: 1

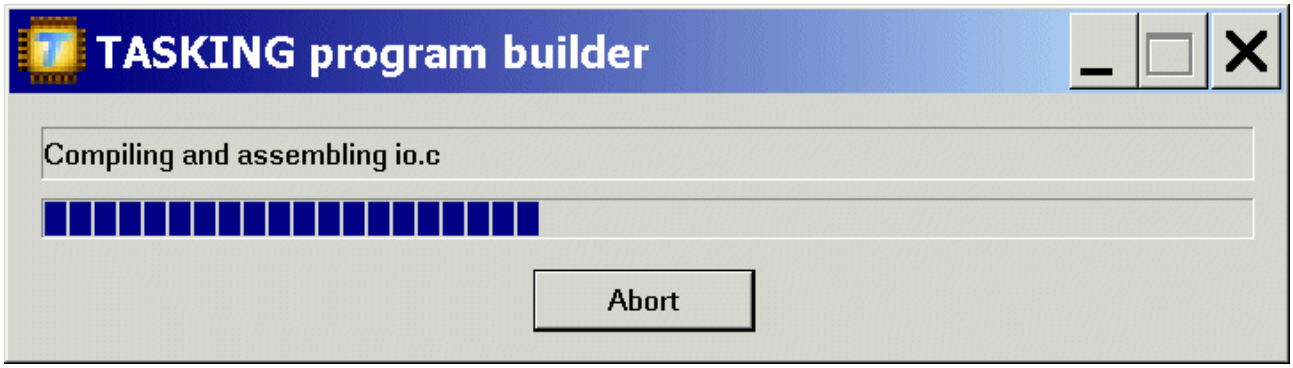
Double click CC2.C insert Code (CAPCOM 2 Timer 7 Interrupt Service Routine):

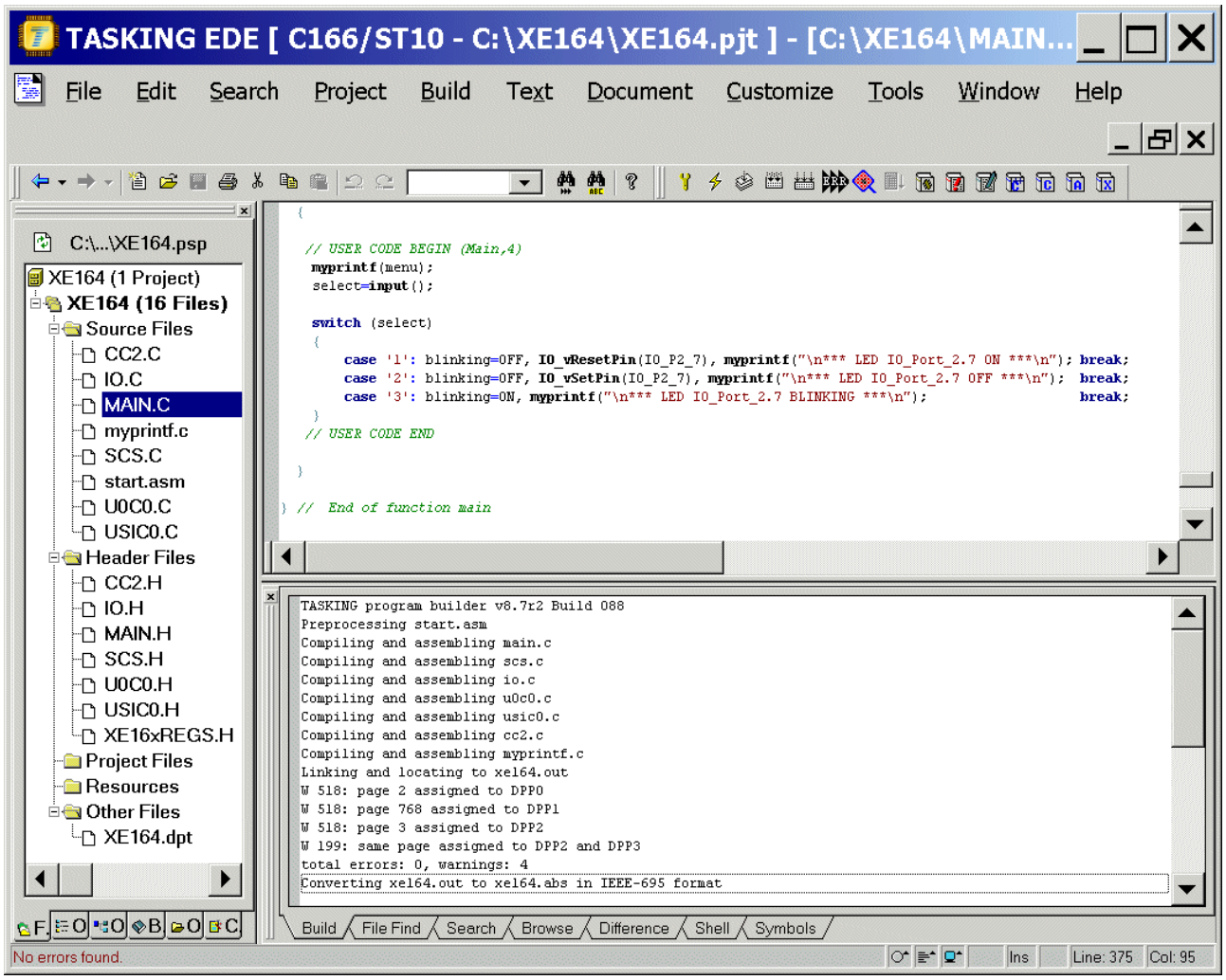
```
if(RS232_wait)
  RS232_wait--;

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



Generate your application program:

Build – Rebuild	or	click
		

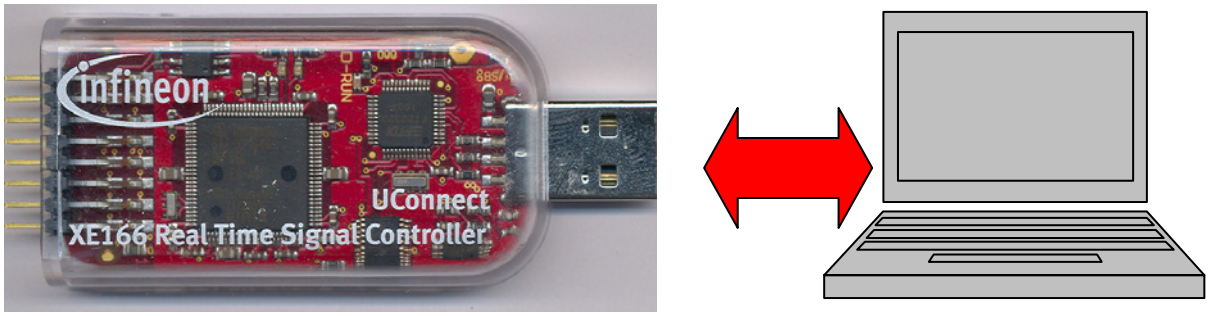




Note:
Programming is now complete.
Therefore we are going to load (On Chip Flash Programming) and run your program on the UConnect-CAN XE164 in the next chapter.

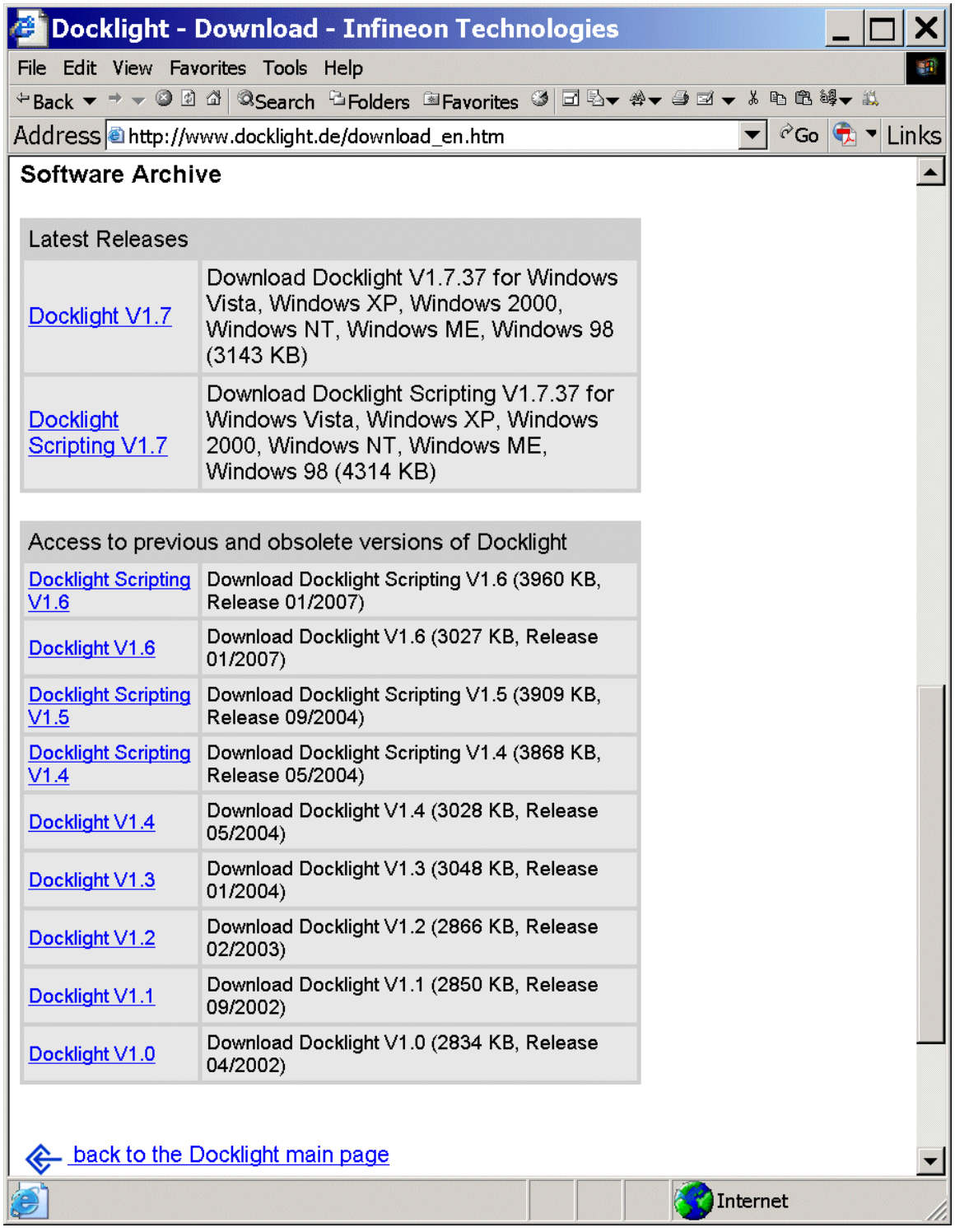
5.) Running your first programming example:

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

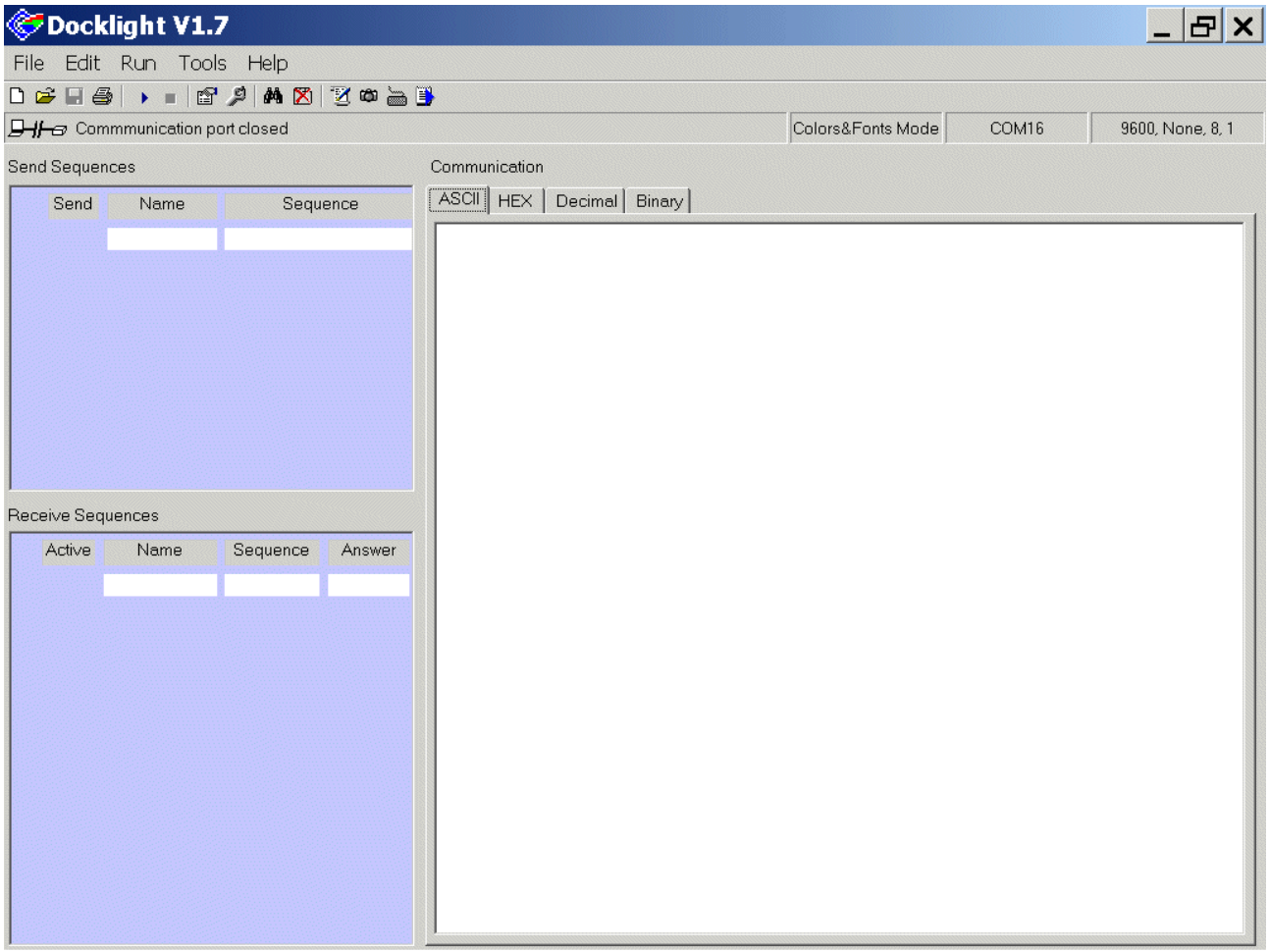


USB Connection:
.) used for: UART communication (the USIC0_CH0/UART/RS232/serial interface is available via USB as a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
.) used for: On-Chip-Flash-Programming and Debugging (first USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
.) the USB connection works also as the power supply.

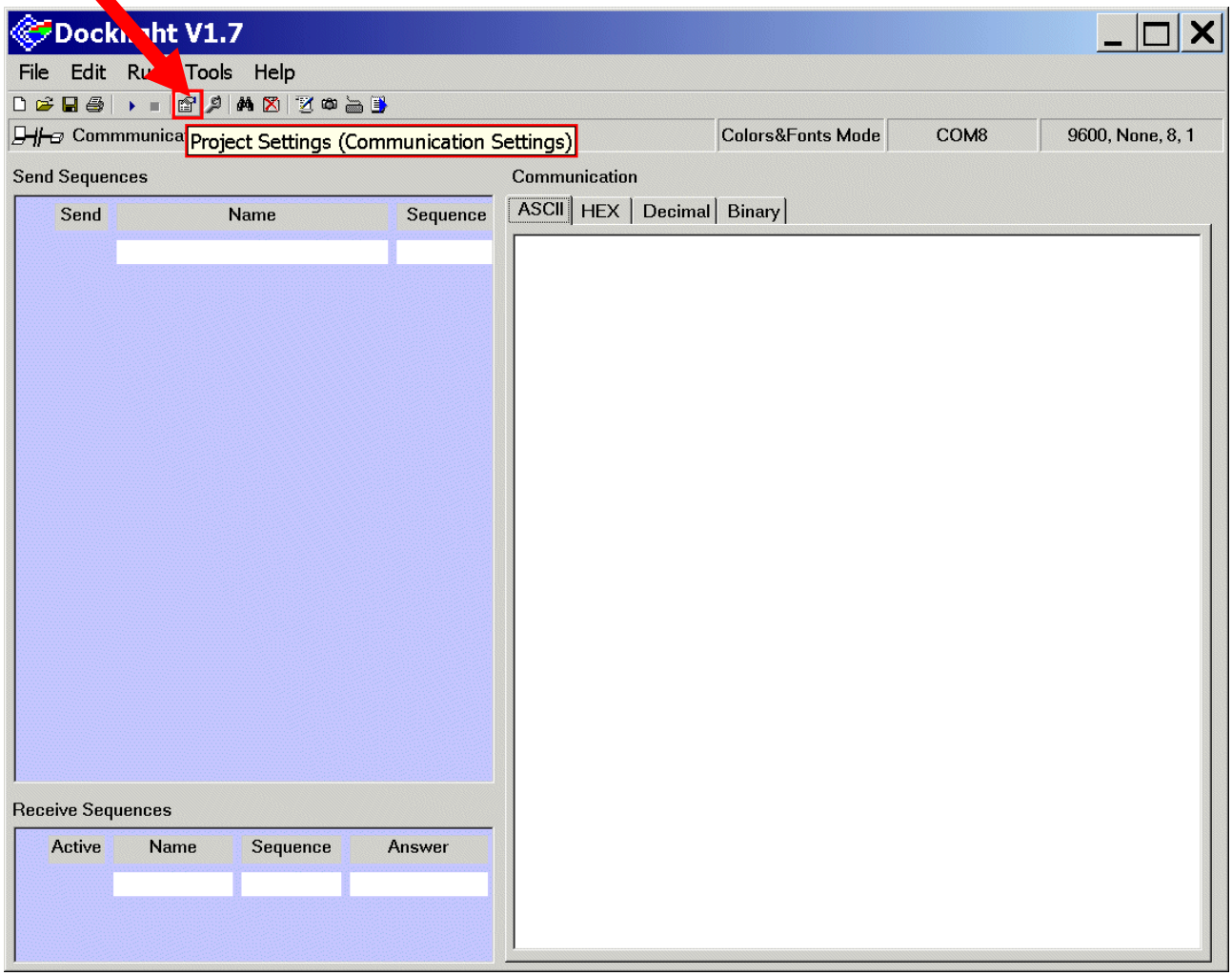
Note:
Now we need a terminal program which is able to handle a virtual COM port (COM5)!
As an example of “any terminal program” we are going to use Docklight.
Docklight can be downloaded @ http://www.docklight.de :



Now, start Docklight:



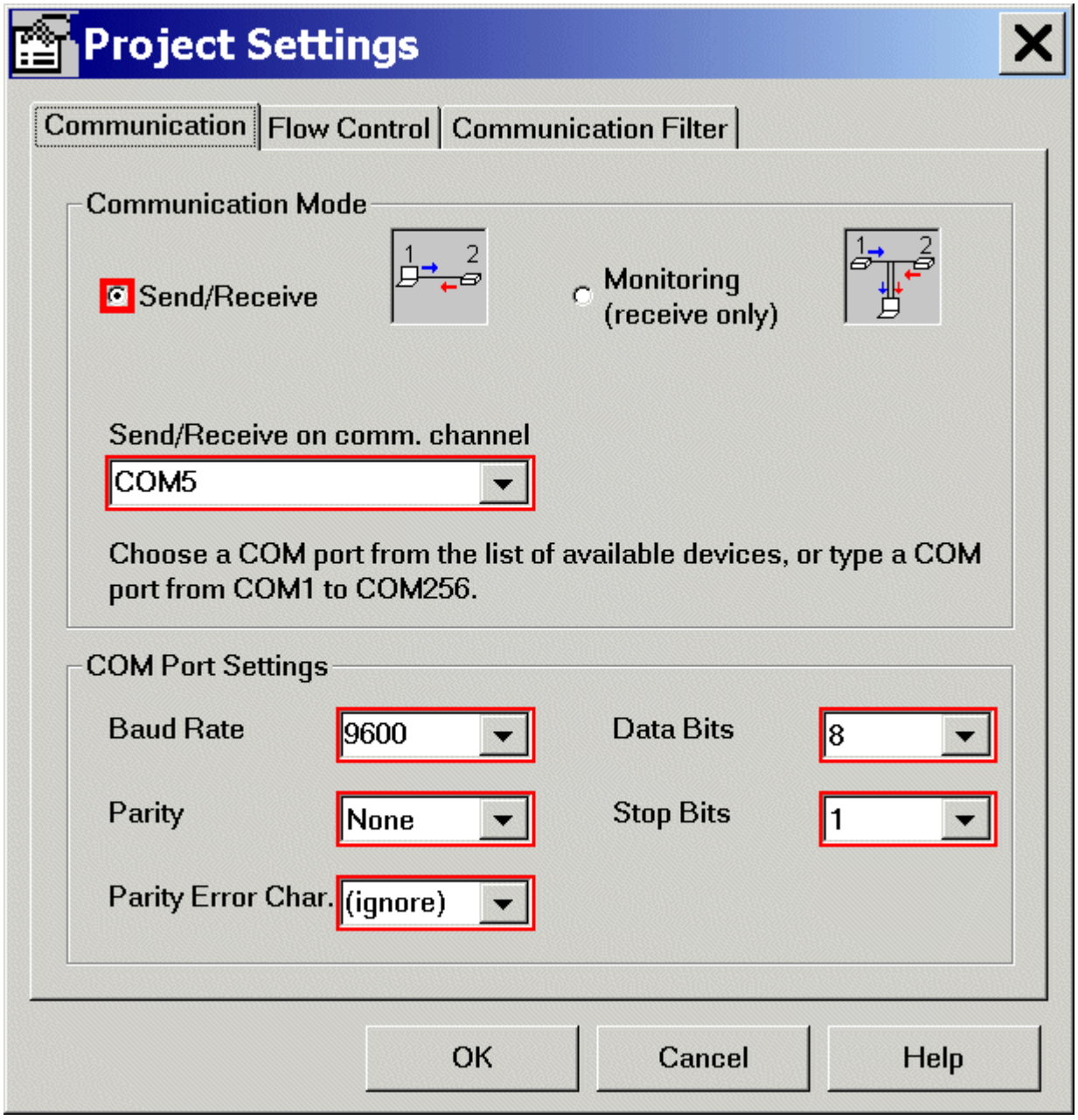
Click: Project Settings



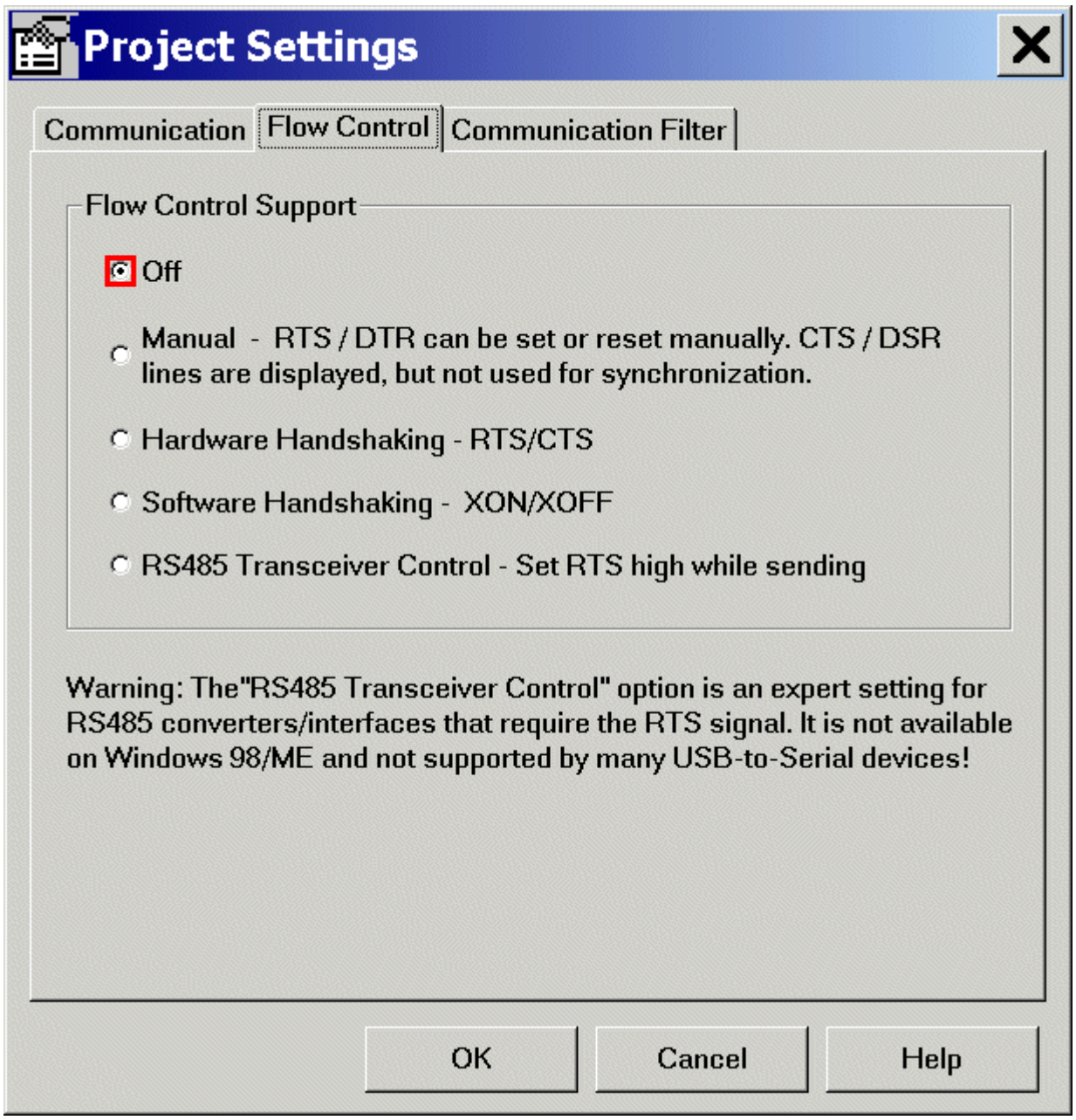
Project Settings:
Communication: Communication Mode: click ⊙ Send/Receive

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

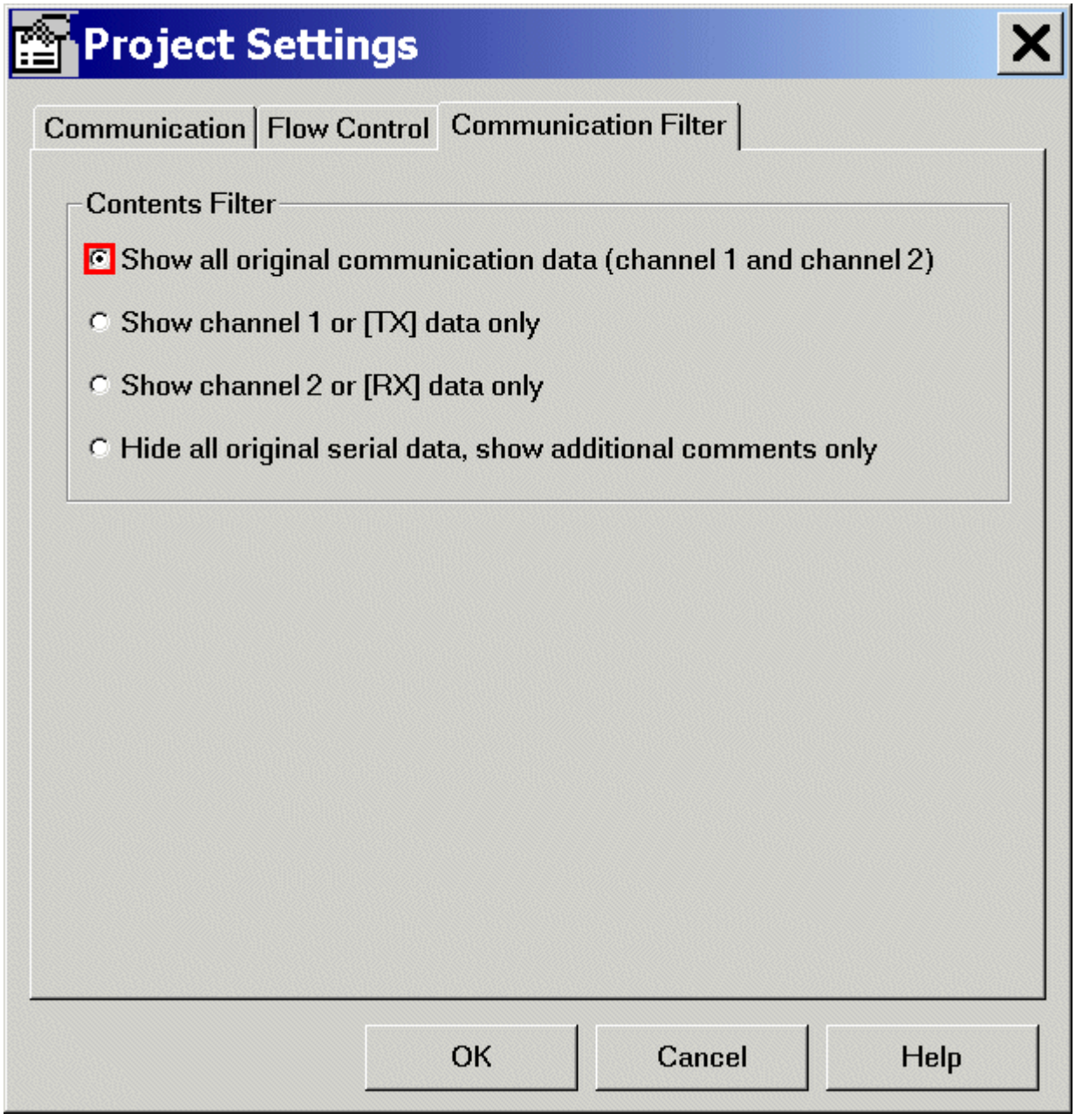
Project Settings: Communication: COM Port Settings: Baud Rate: select 9600
Project Settings: Communication: COM Port Settings: Parity: select None
Project Settings: Communication: COM Port Settings: Parity Error Char.: select (ignore)
Project Settings: Communication: COM Port Settings: Data Bits: select 8
Project Settings: Communication: COM Port Settings: Stop Bits: select 1



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

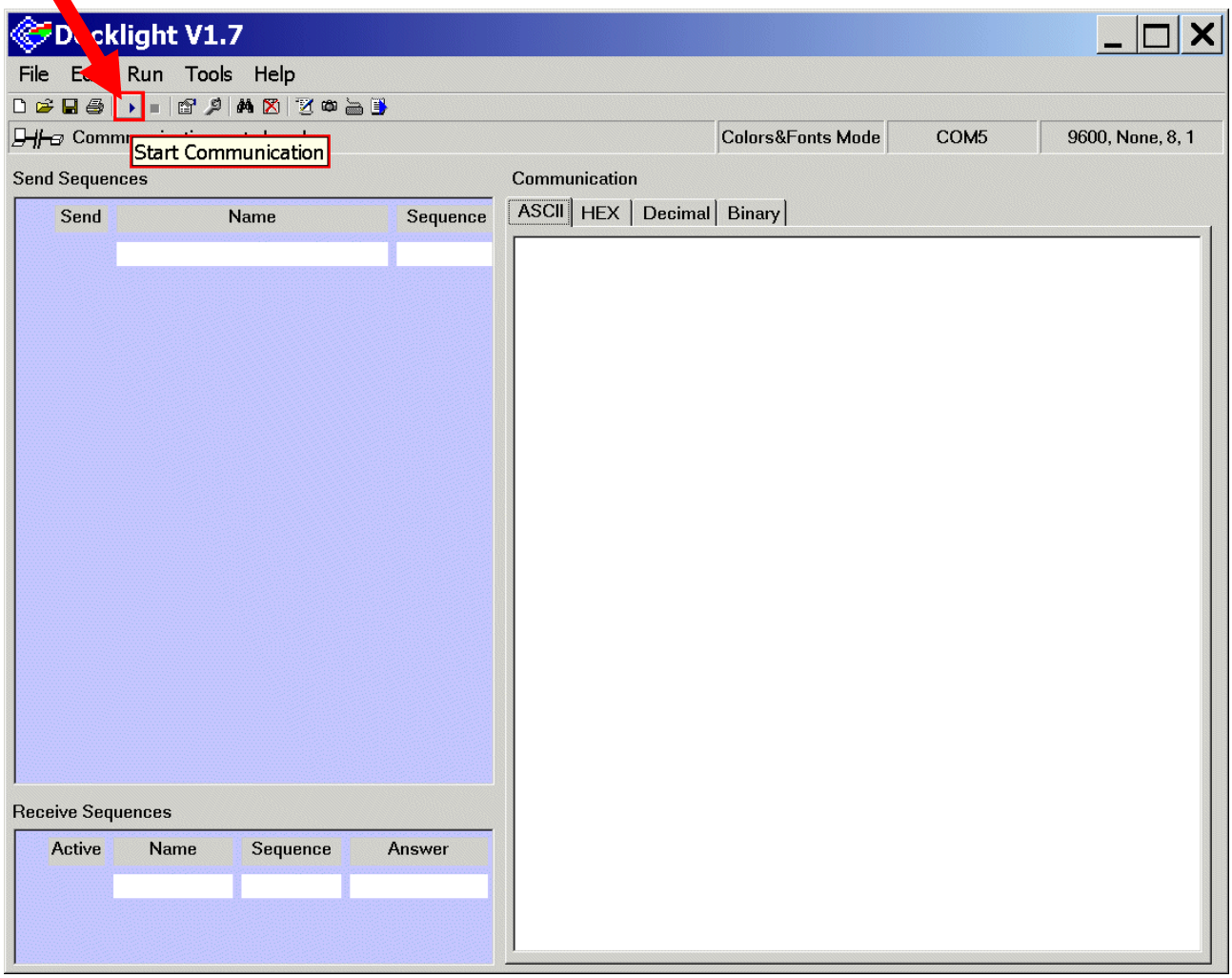


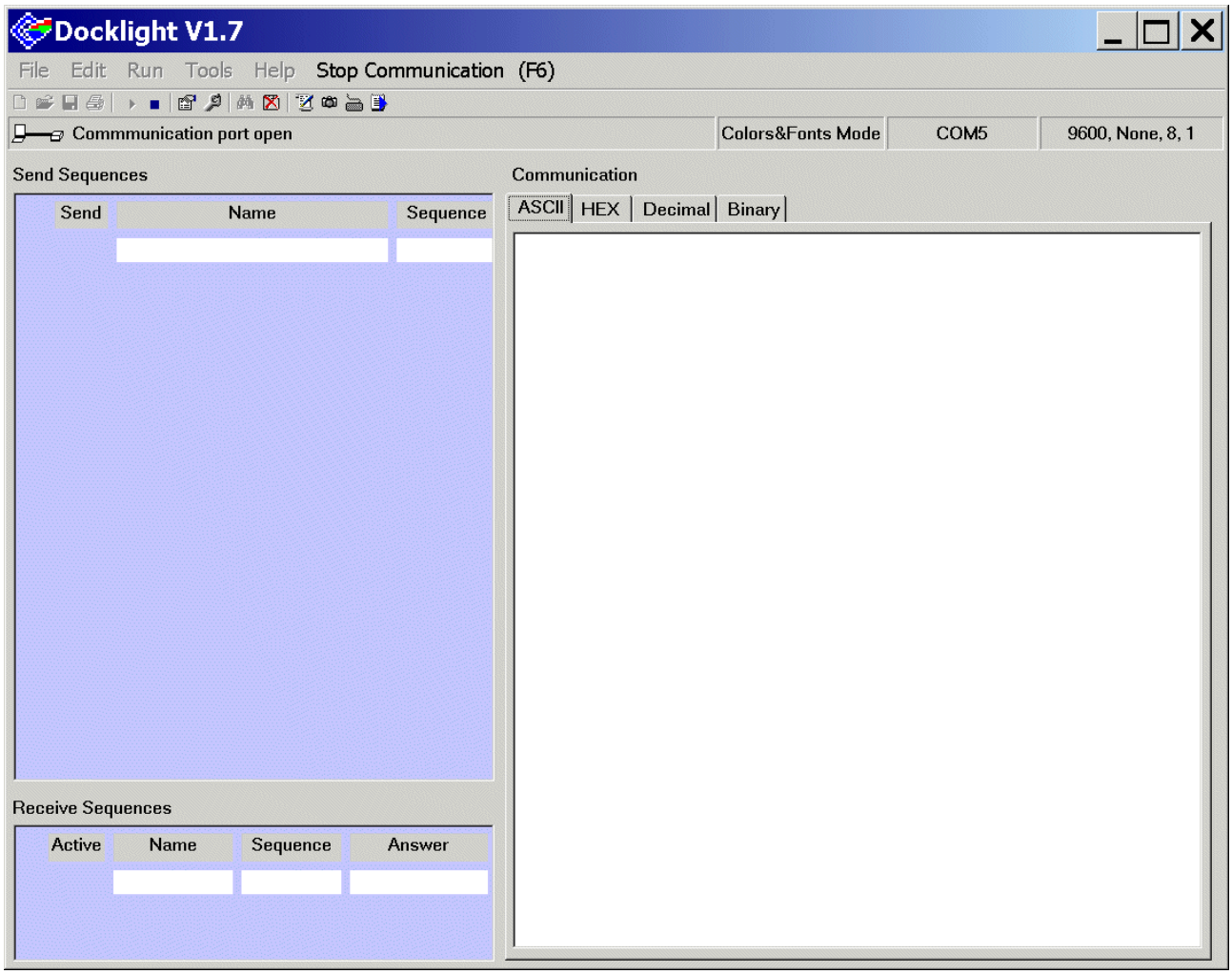
Project Settings:
Communication Filter: Contents Filter: click ⊙ Show all original communication data



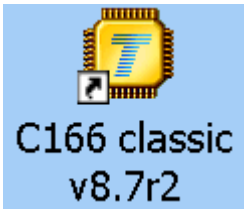
OK

Click:

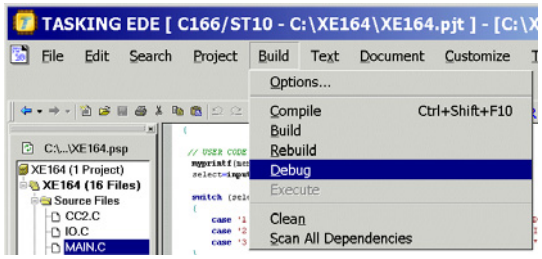
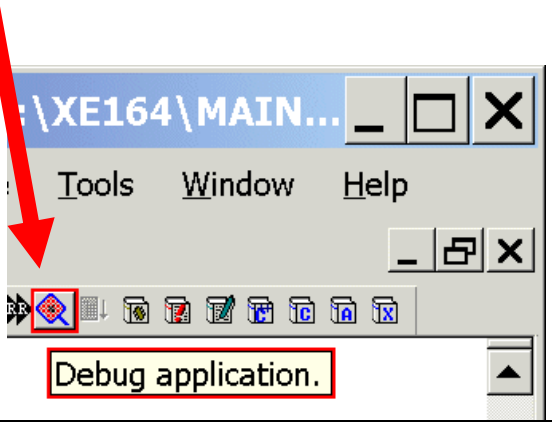

cklight V1.7
File
Run
Tools
Help
Start Communication
Colors&Fonts Mode
COM5
9600, None, 8, 1
Send Sequences
Send
Name
Sequence
Communication
ASCII
HEX
Decimal
Binary
Receive Sequences
Active
Name
Sequence
Answer

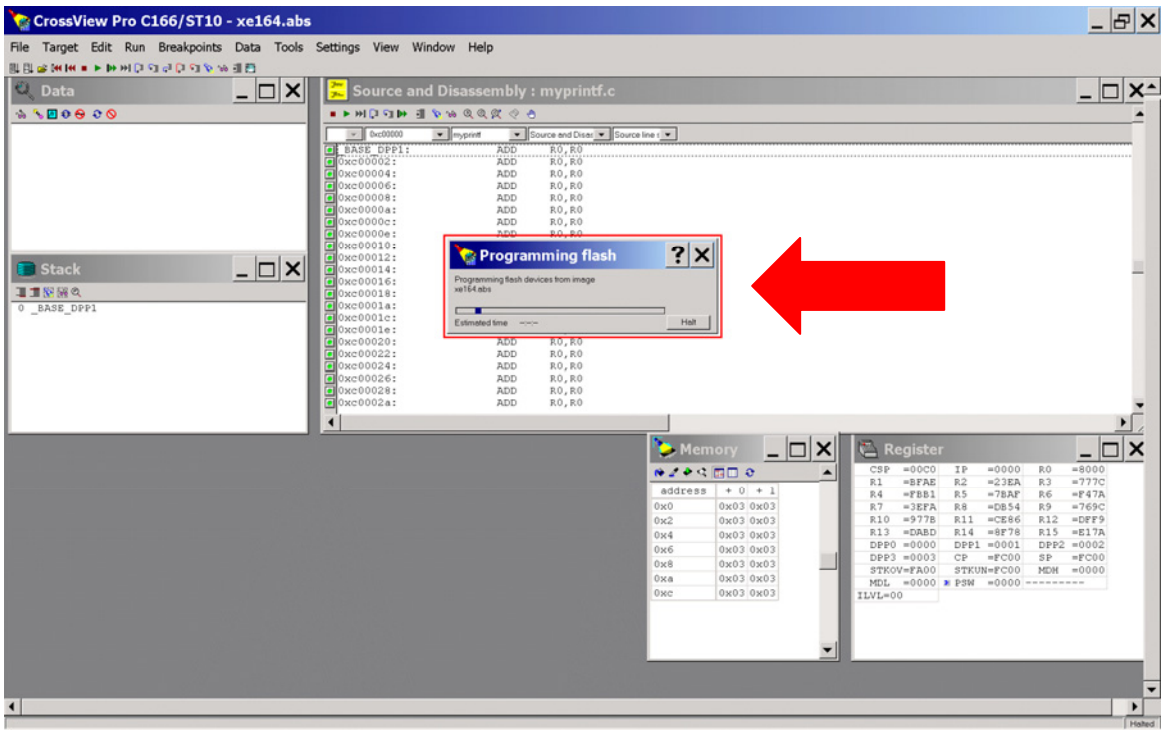


Note:
Docklight is now ready for serial communication!

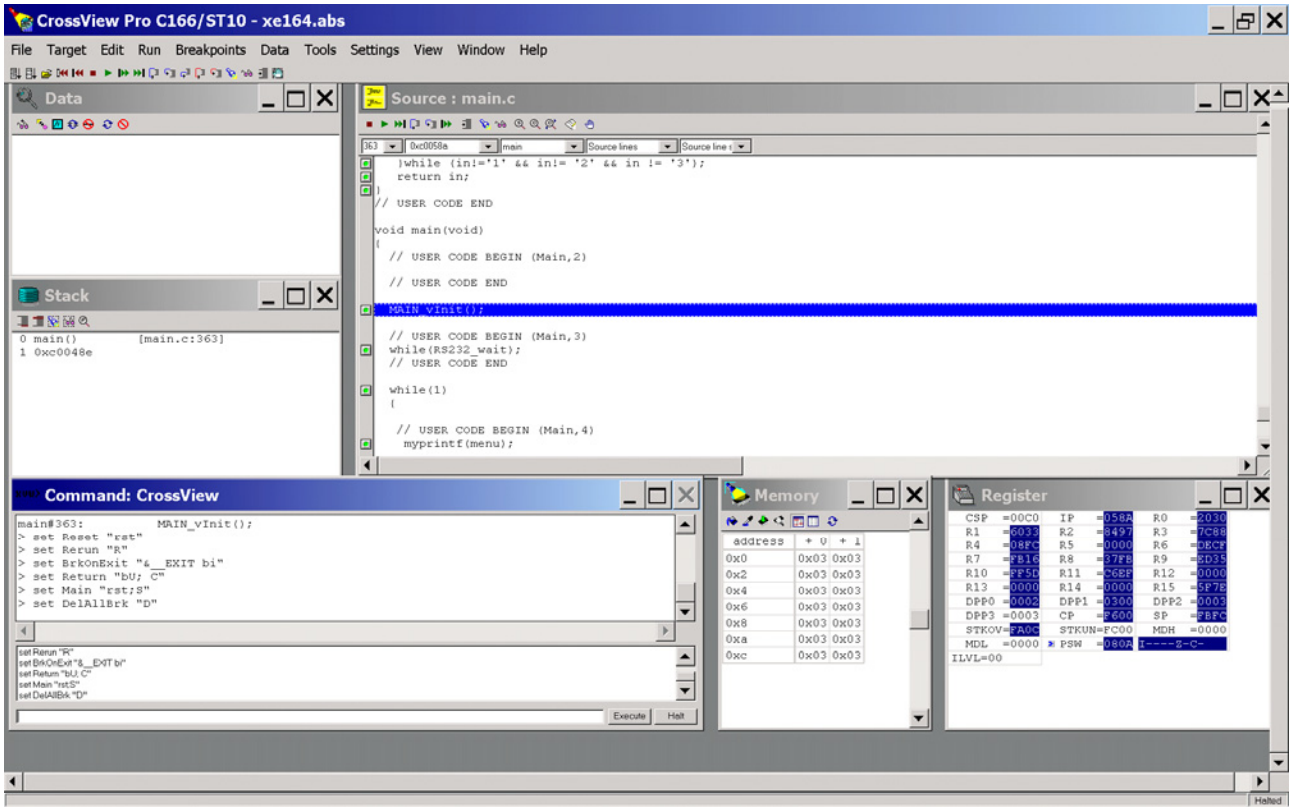


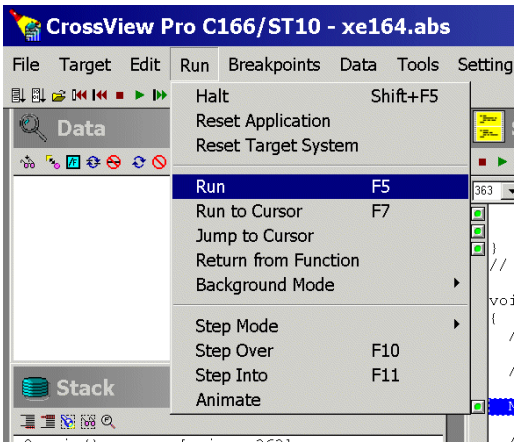
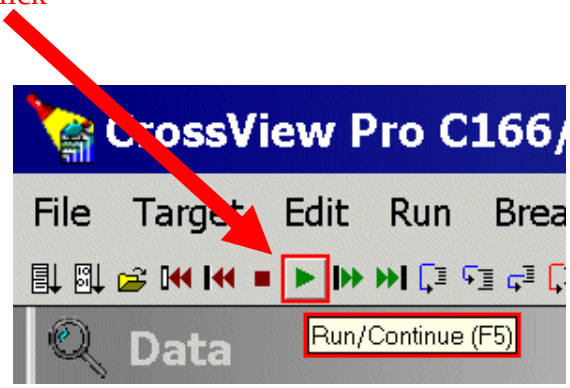
Go to TASKING EDE Classic:

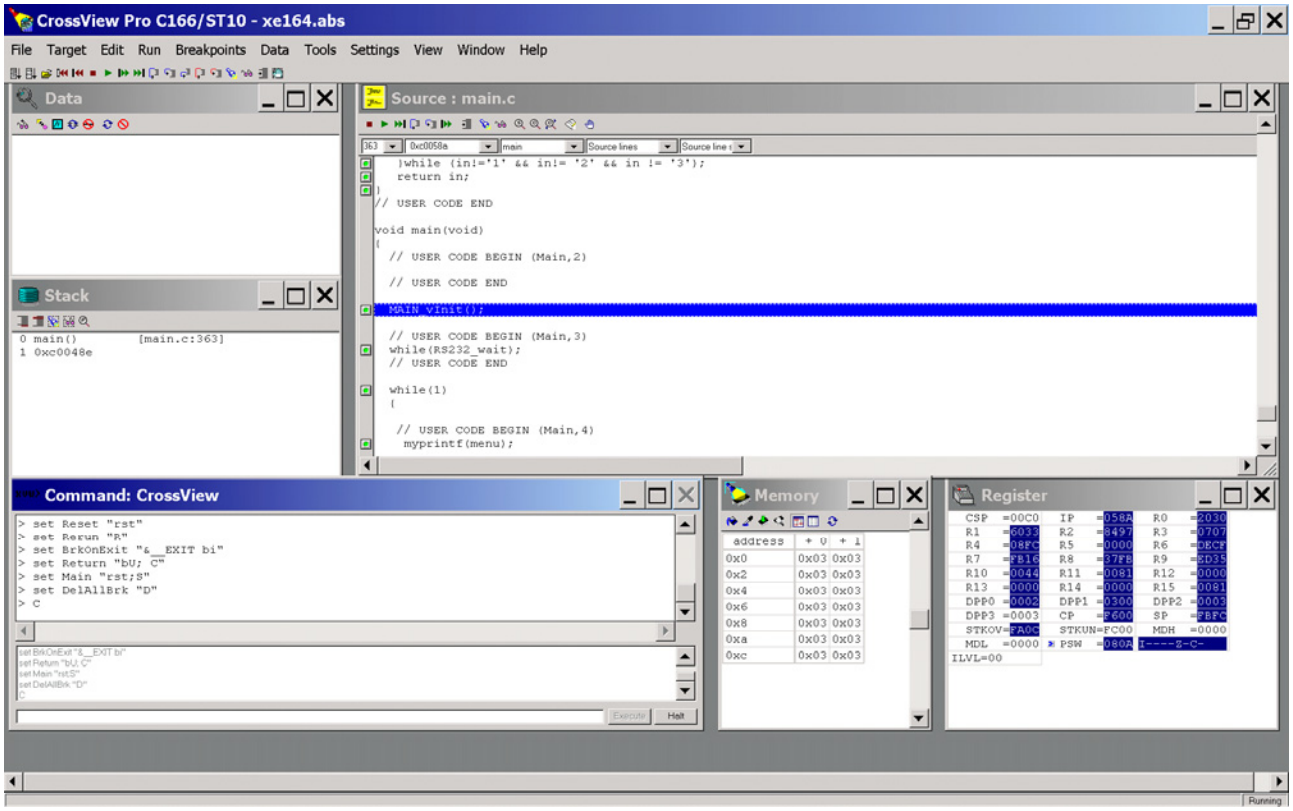
Build – Debug	or	click
		



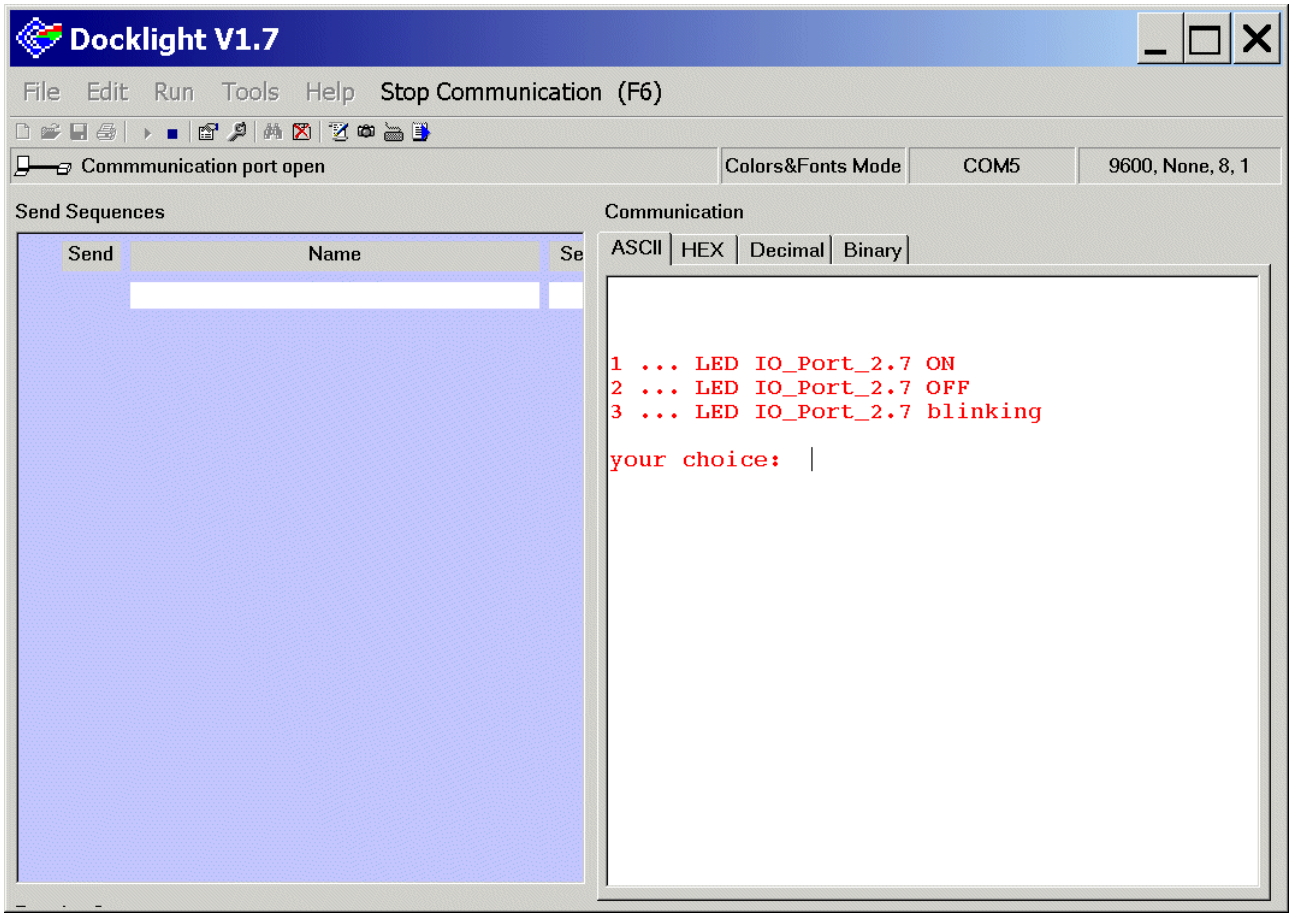
Note:
As you can see in the screenshot above, the on chip program flash is automatically programmed when the debugger is launched.


CrossView Pro C166/ST10 - xe164.abs
File Target Edit Run Breakpoints Data Tools Settings View Window Help
Data
Stack
0 main() [main.c:363]
1 0xc0048e
Source : main.c
}while (in!='1' && in!= '2' && in != '3');
return in;
}
// USER CODE END
void main(void)
{
// USER CODE BEGIN (Main,2)
// USER CODE END
MAIN_vInit();
// USER CODE BEGIN (Main,3)
while(RS232_wait);
// USER CODE END
while(1)
{
// USER CODE BEGIN (Main,4)
myprintf(menu);
Command: CrossView
main#363: MAIN_vInit();
> set Reset "rst"
> set Rerun "R"
> set BrkOnExit "&__EXIT bi"
> set Return "bU; C"
> set Main "rst;S"
> set DelAllBrk "D"
Execute
Halt
Memory
address
0x0 0x03 0x03
0x2 0x03 0x03
0x4 0x03 0x03
0x6 0x03 0x03
0x8 0x03 0x03
0xa 0x03 0x03
0xc 0x03 0x03
Register
CSP =00C0 IP =058A R0 =2030
R1 =5033 R2 =8497 R3 =7C88
R4 =08FC R5 =0000 R6 =DECF
R7 =FB16 R8 =37FB R9 =ED35
R10 =FF5D R11 =C6EF R12 =0000
R13 =0000 R14 =0000 R15 =5F7E
DPP0 =0002 DPP1 =0300 DPP2 =0003
DPP3 =0003 CP =F600 SP =FBFC
STKOV=FA0C STKUN=FC00 MDH =0000
MDL =0000 PSW =080A I----S-C-
ILVL=00
Halted

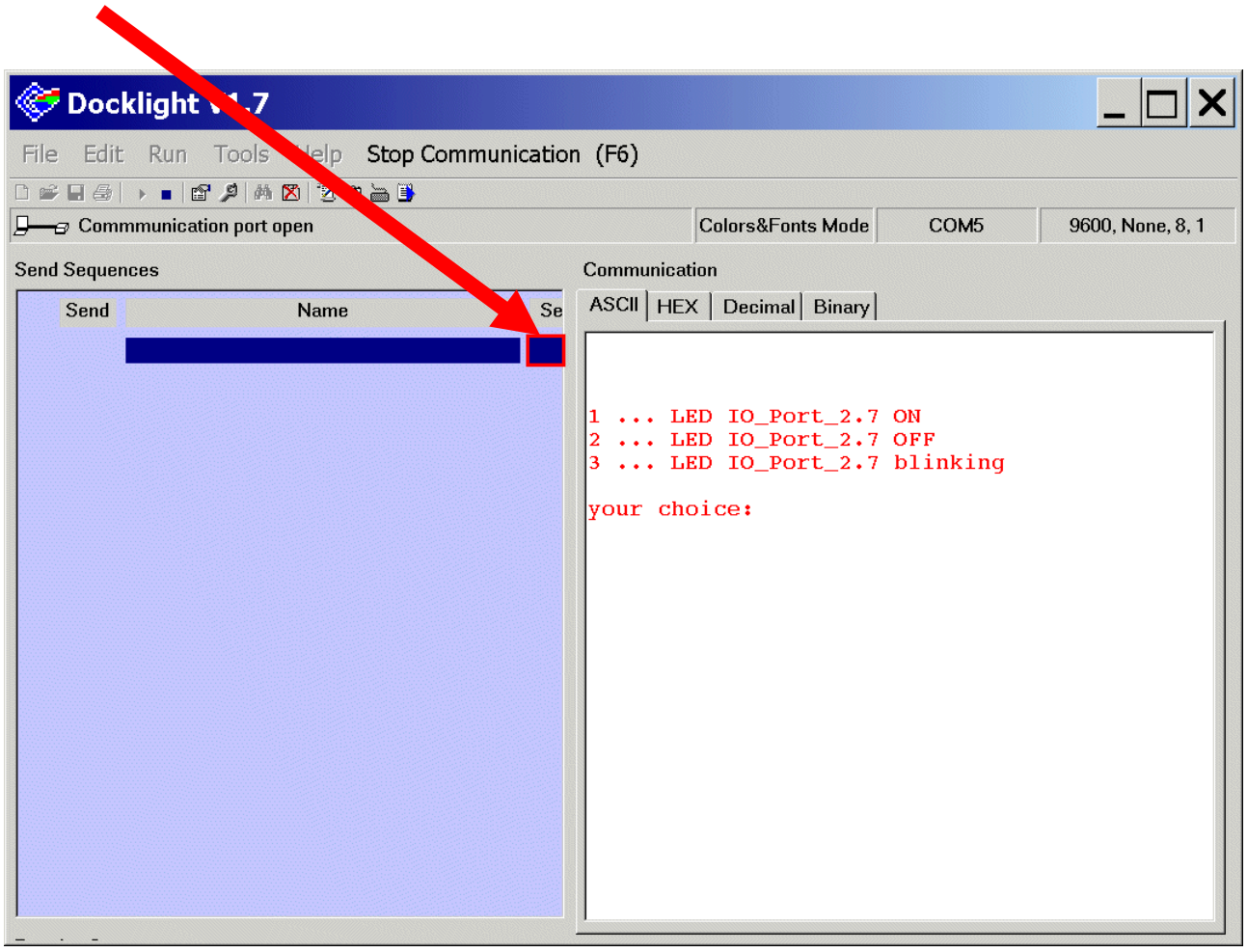
Run – Run  	or	click 



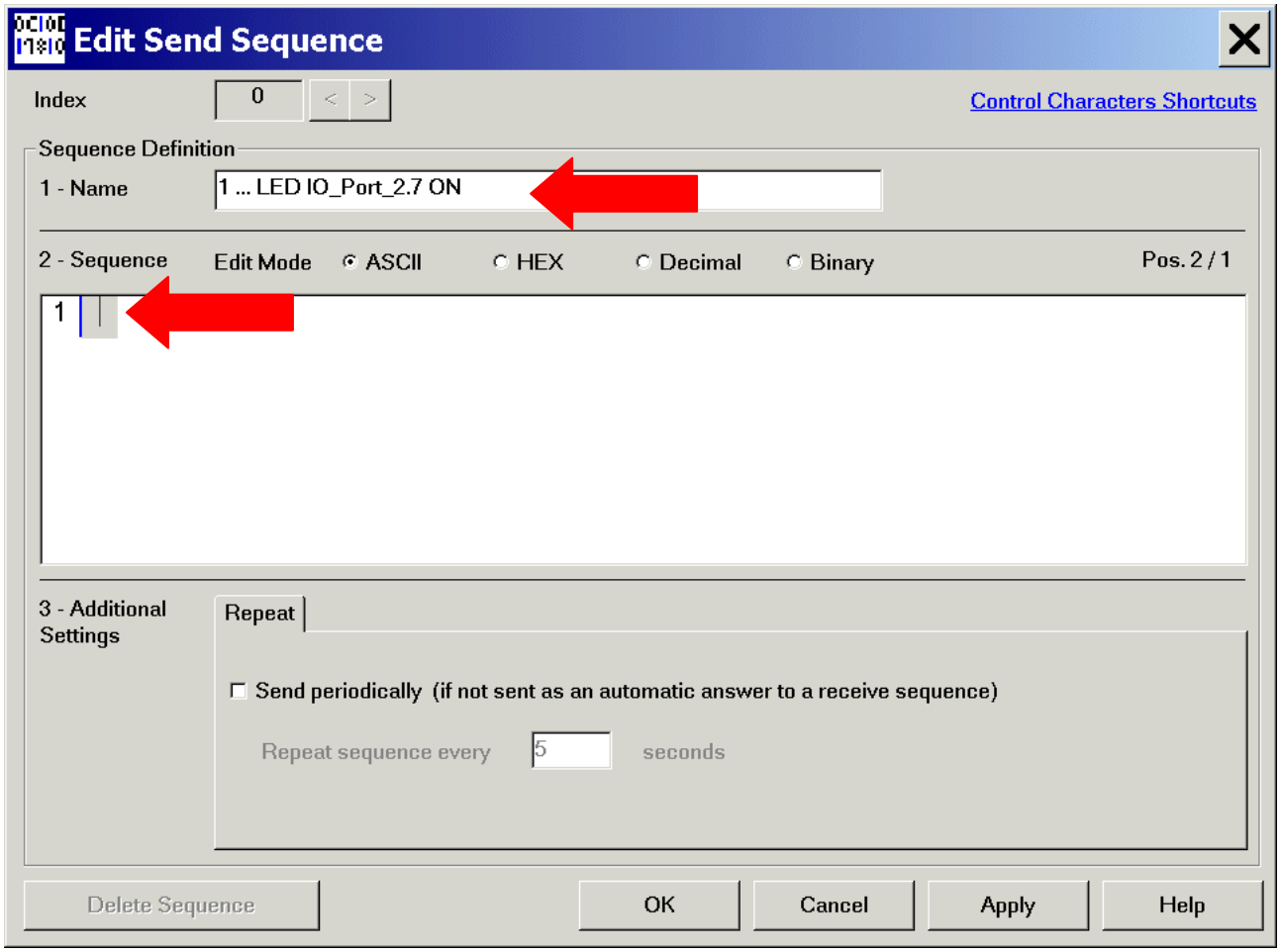
Go to Docklight and see the result:



Double click inside the red box:

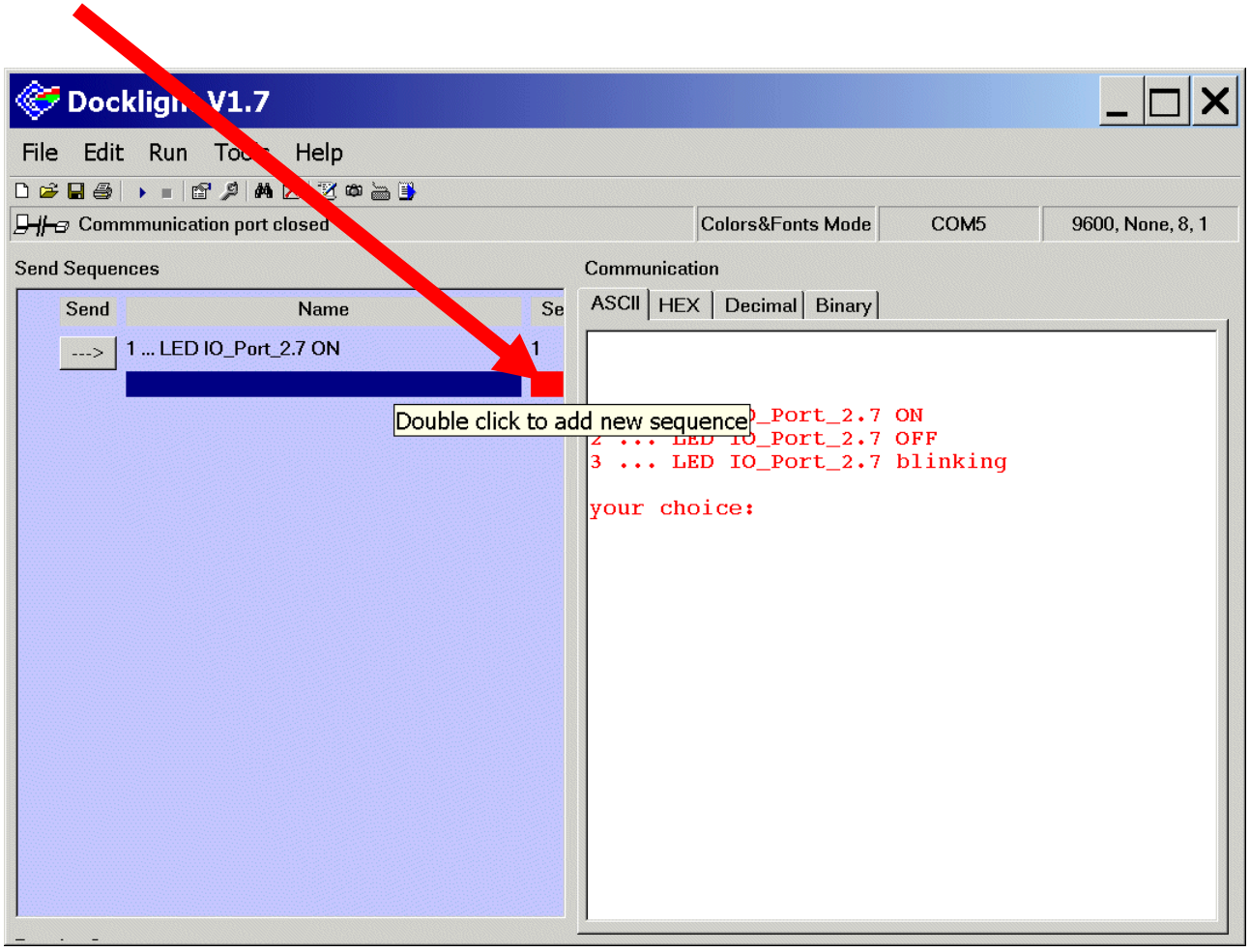


Edit Send Sequence: Sequence Definition: 1- Name: insert: 1 ... LED IO_Port_2.7 ON
Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 1

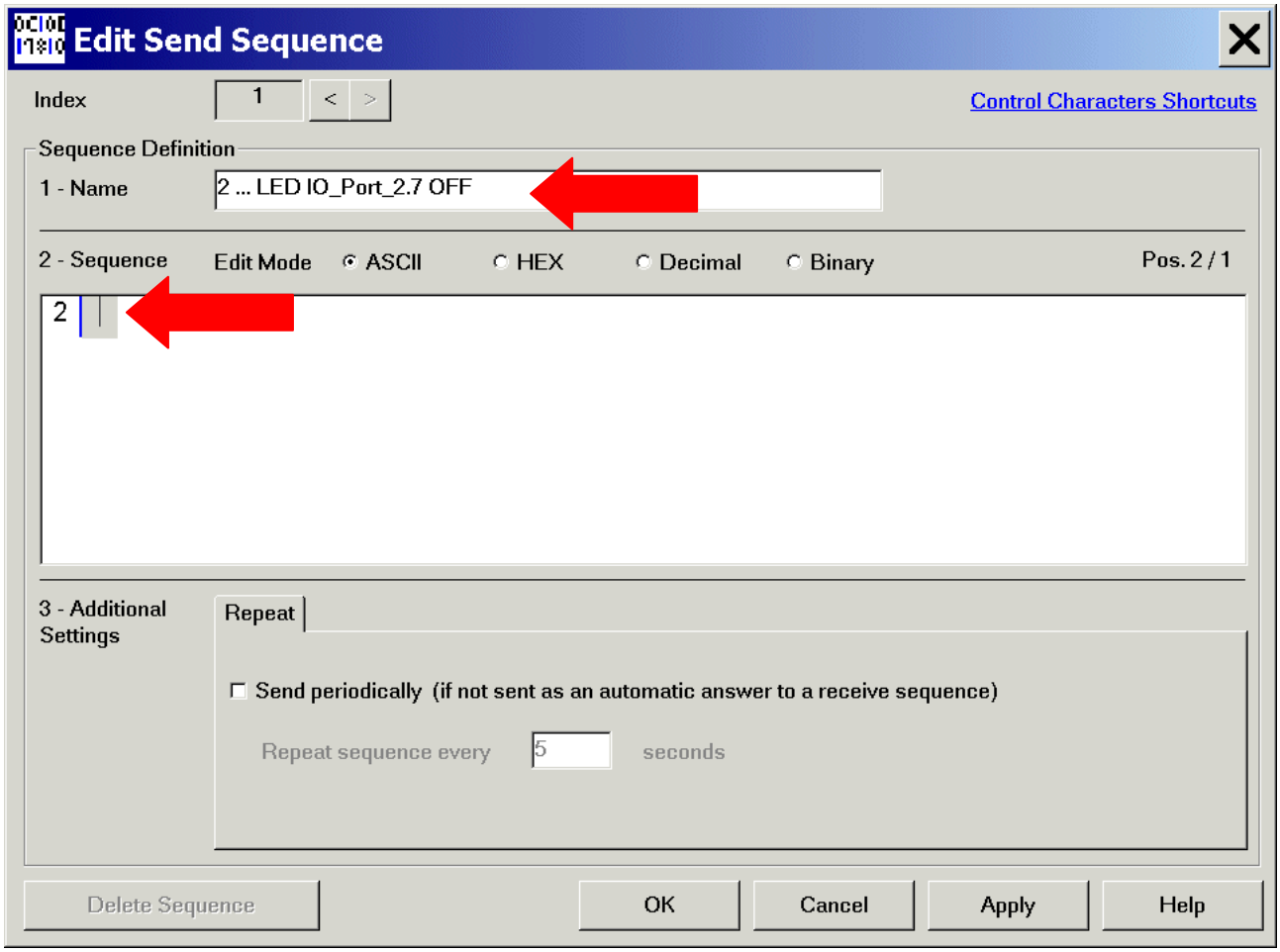


OK

Double click inside the red box:

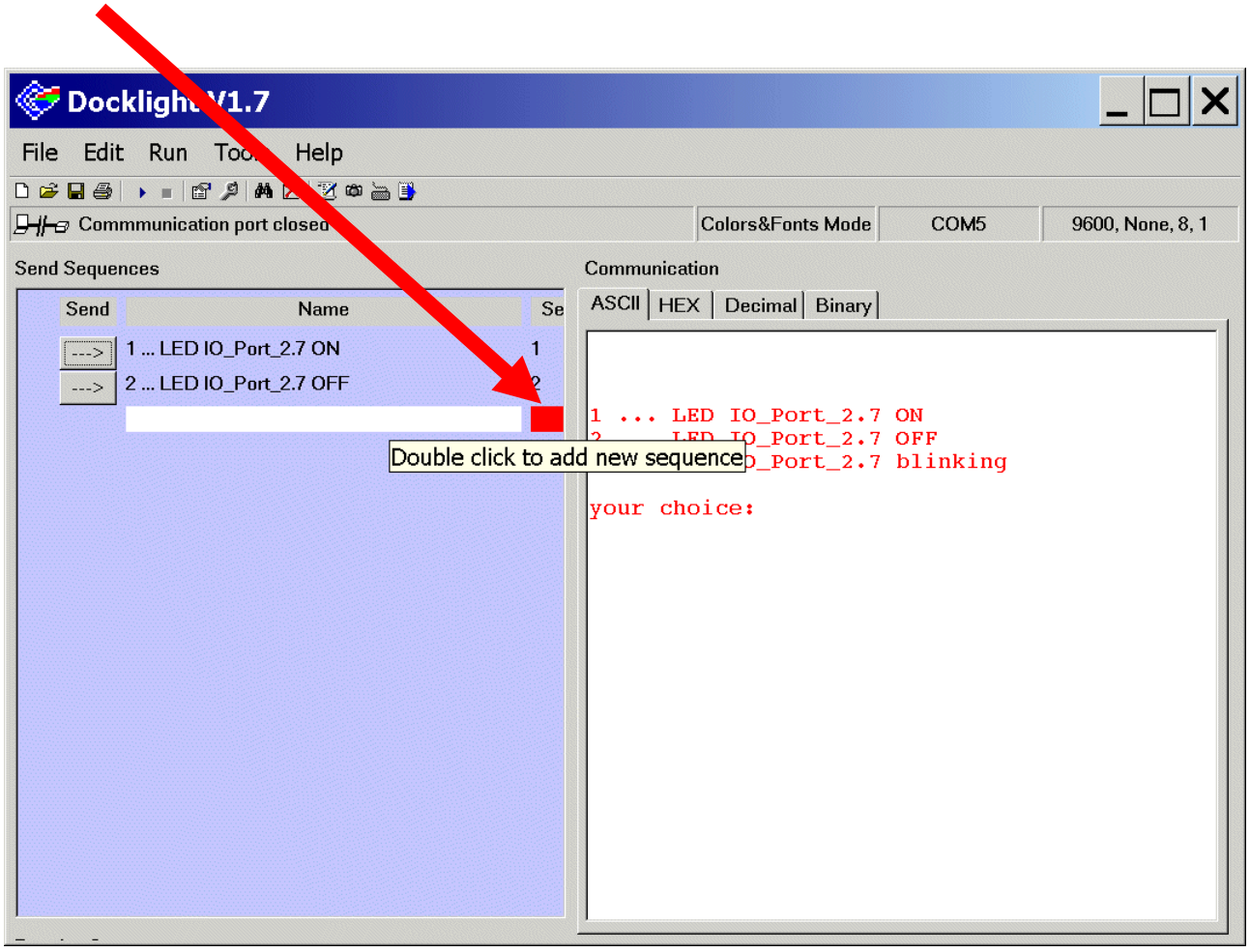


Edit Send Sequence: Sequence Definition: 1- Name: insert: 2 ... LED IO_Port_2.7 OFF
Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 2

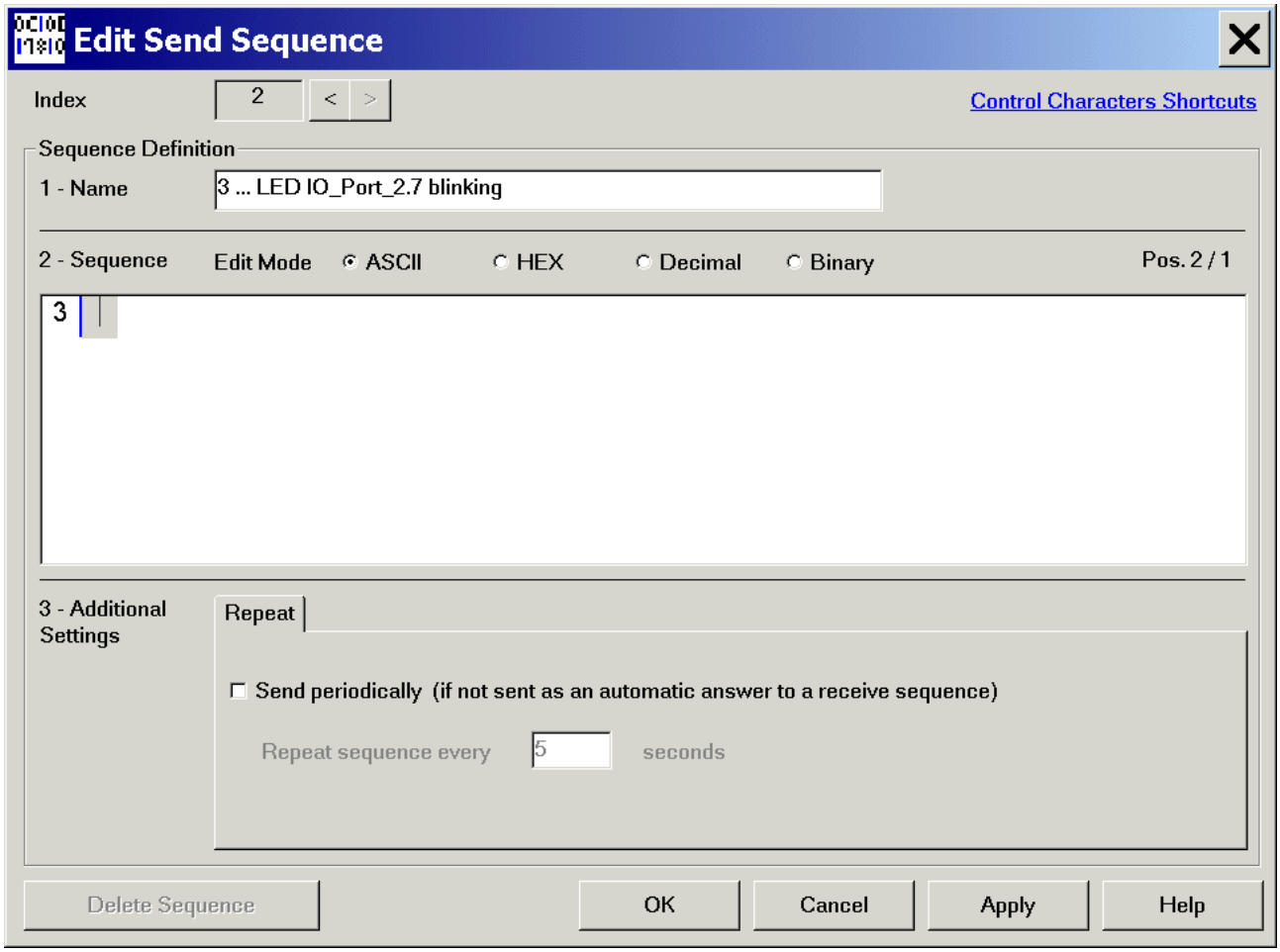


OK

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

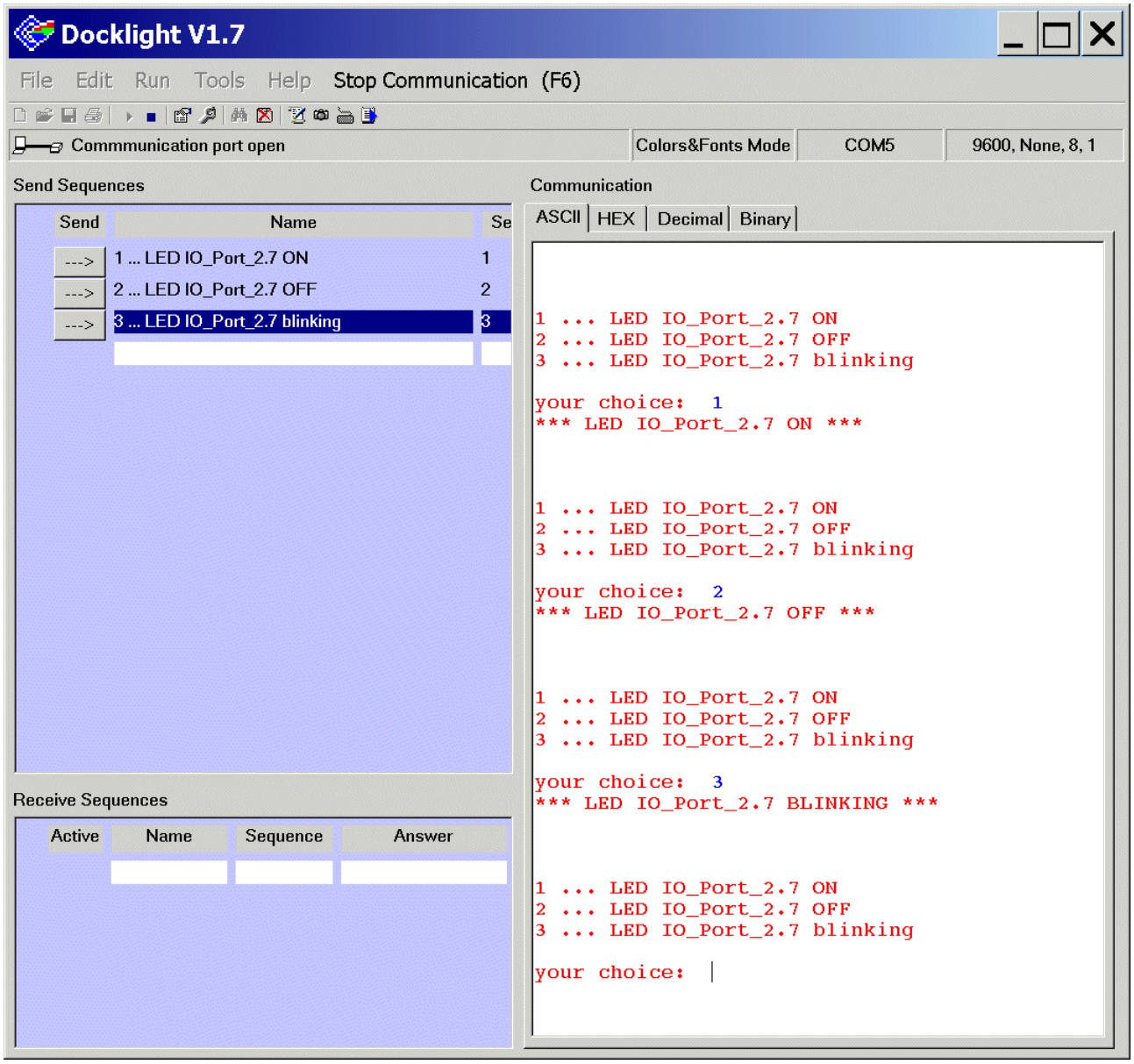


OK

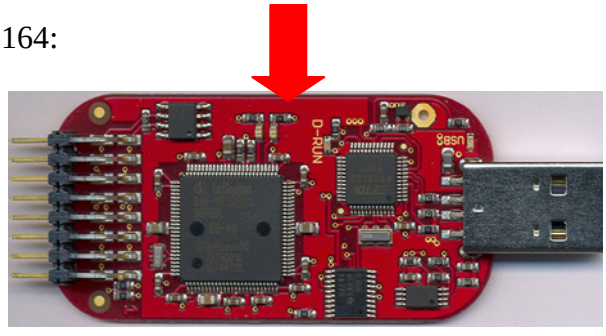
Click ---> 1 ... LED IO_Port_2.7 ON or

Click ---> 2 ... LED IO_Port_2.7 OFF or

Click ---> 3 ... LED IO_Port_2.7 blinking

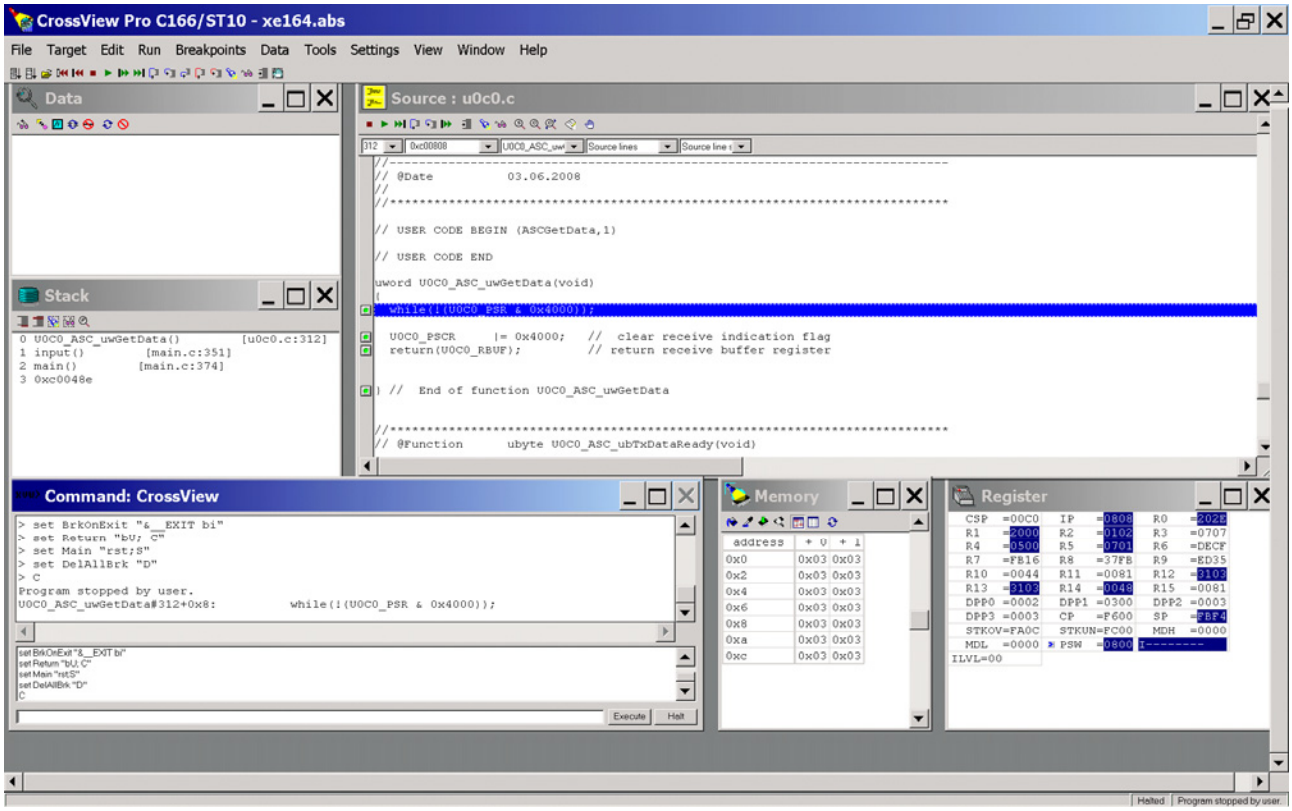


and check the results on your UConnect-CAN XE164:



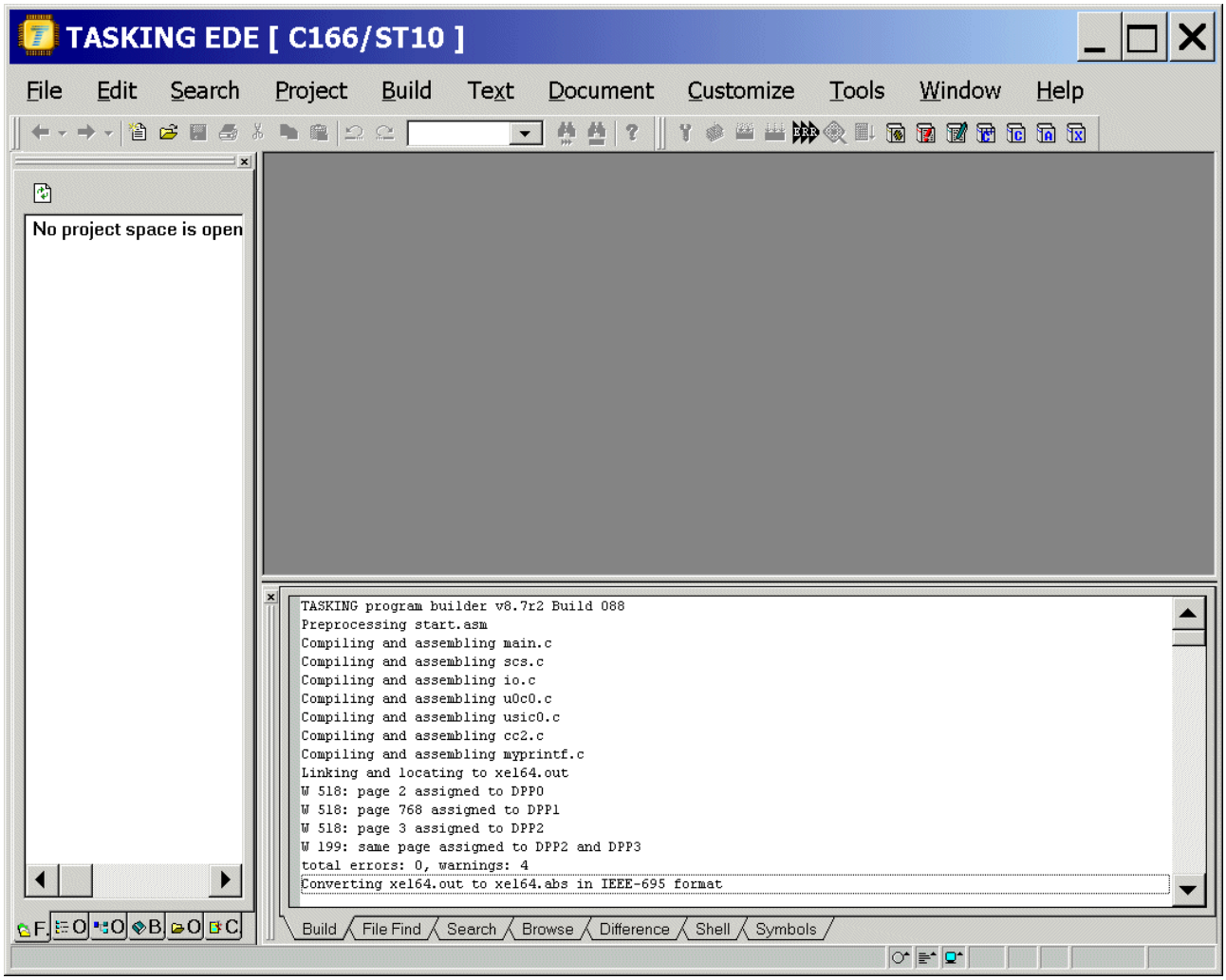
Now we close the Debugger and TASKING EDE Classic:

Run – Halt



File - Exit

File – Close Project Space



File - Exit

Conclusion:

In this step-by-step book you have learned how to use the UConnect-CAN XE164 together with the TASKING EDE Classic tool chain.

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).

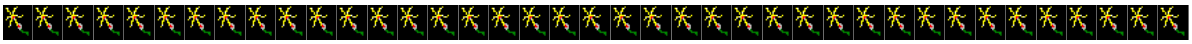
All these step-by-step books use the same microcontroller resources and the same example code.

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.

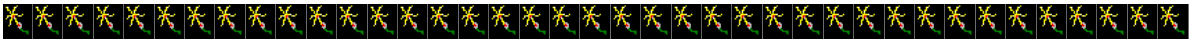
6.) Feedback (UConnect-CAN XE164, TASKING EDE Classic): Your opinion, suggestions and/or criticisms



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
FAX: +43 (0) 4242 3020 5783



Your suggestions:

http://www.infineon.com

Published by Infineon Technologies AG