AP08096

XC886/XC888

XC888 Starter Kit "Cookery-Book" for a "Hello world" application.

Using DAvE (Code Generator) and

DAvE Bench (Open Platform for Free Tools: IDE, Compiler, Debugger, Utility Tools).

Microcontrollers

Never stop thinking

Edition 2010-02-05

Published by
Infineon Technologies AG
81726 München, Germany

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:	2010-03	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 8.

Introduction:

This “Application Note / Appnote” is an Infineon Hands-On-Training / Cookery-Book / step-by-step-book.
It will help inexperienced users to get an XC888/XC886 Evaluation Board / Starter Kit Board / Easy Kit 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 Starter Kit Board.
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.

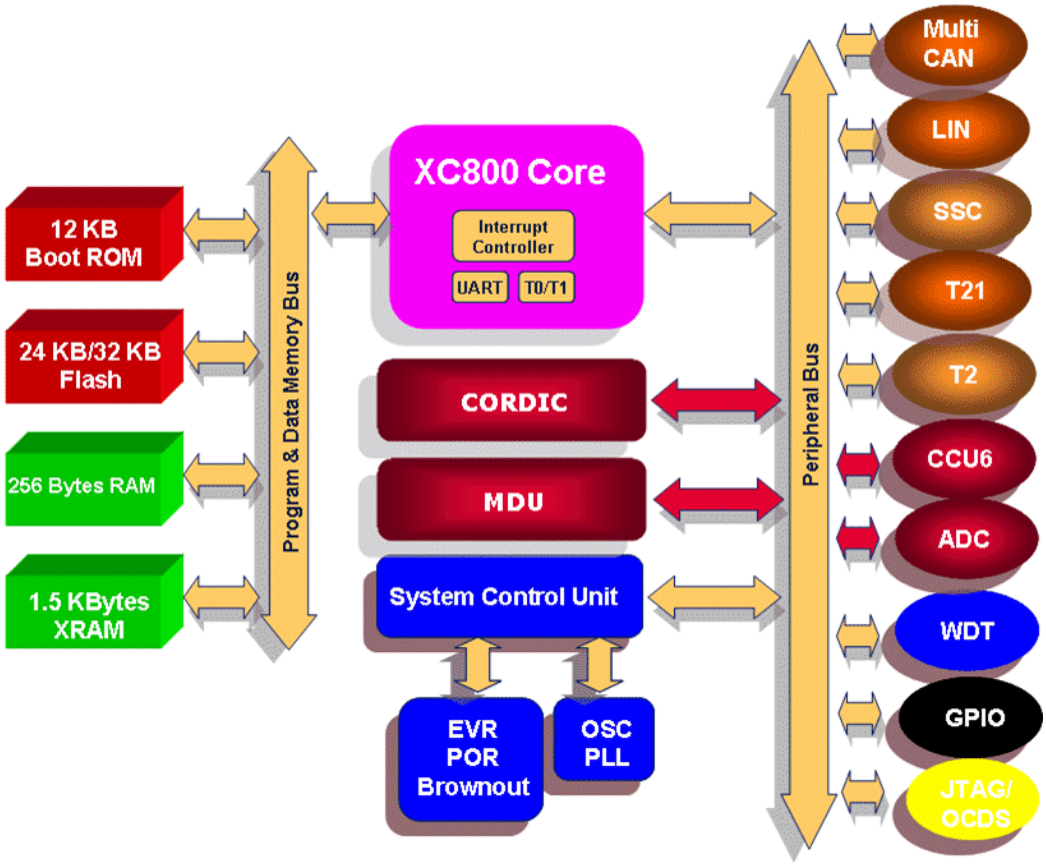
Note:
The style used in this document focuses on working through this material as fast and easily as possible. That means there are full screenshots instead of dialog-window-screenshots; extensive use of colours and page breaks; and listed source-code is not formatted to ease copy & paste.

Have fun and enjoy the XC888/XC886 microcontrollers!

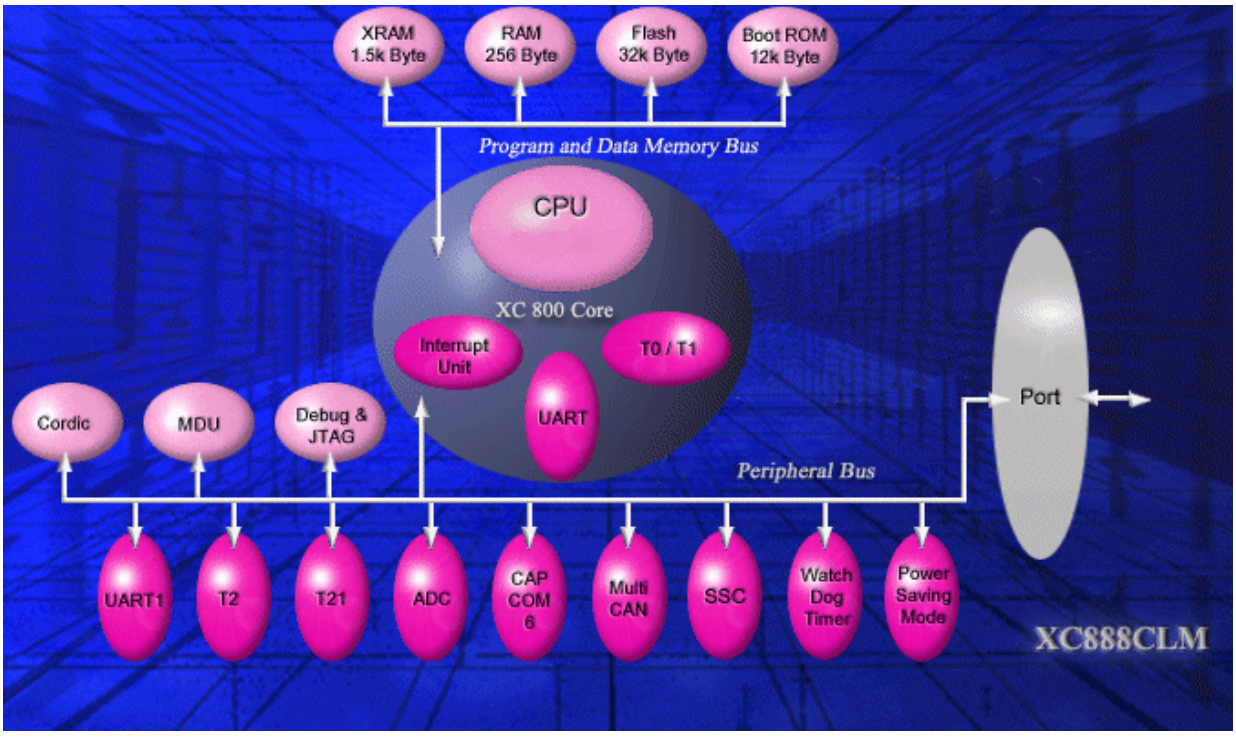
The New
XC800-Family
Designed to Make the Difference

Programming Example
XC888CM-8FFA
Starter Kit

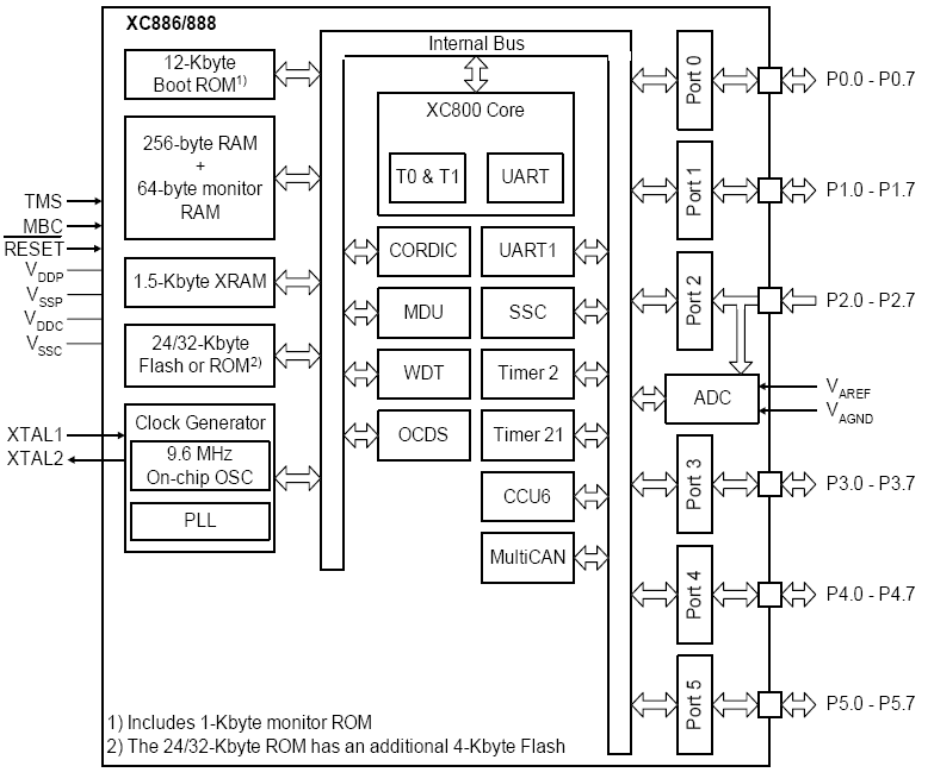
XC888CLM-8FFA Block Diagram (Source: Product Marketing)



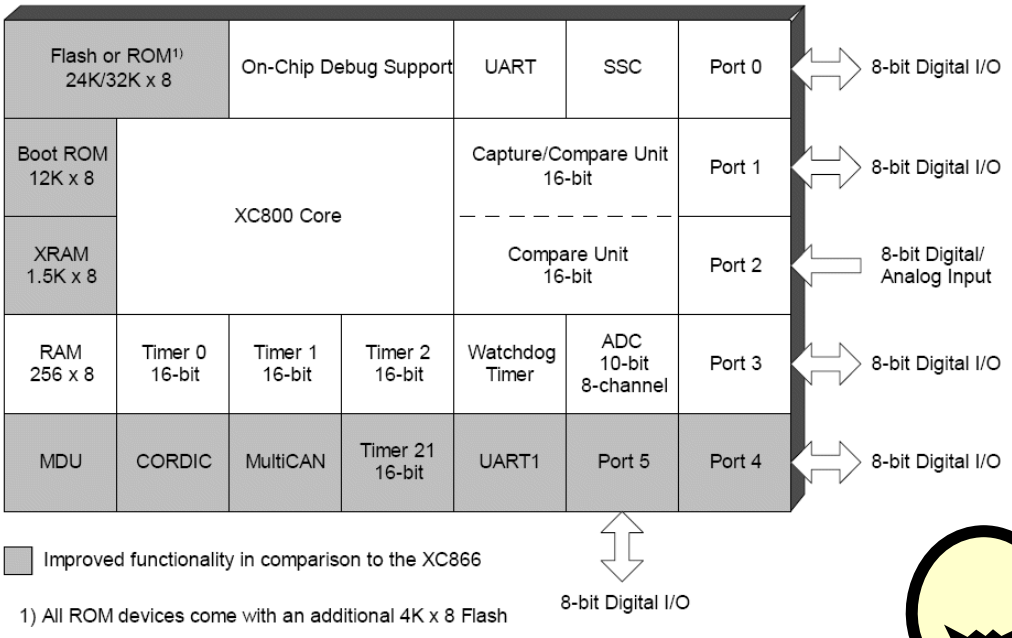
XC888CLM-8FFA Block Diagram (Source: DAvE)



XC888CLM-8FFA Block Diagram (Source: User's Manual)

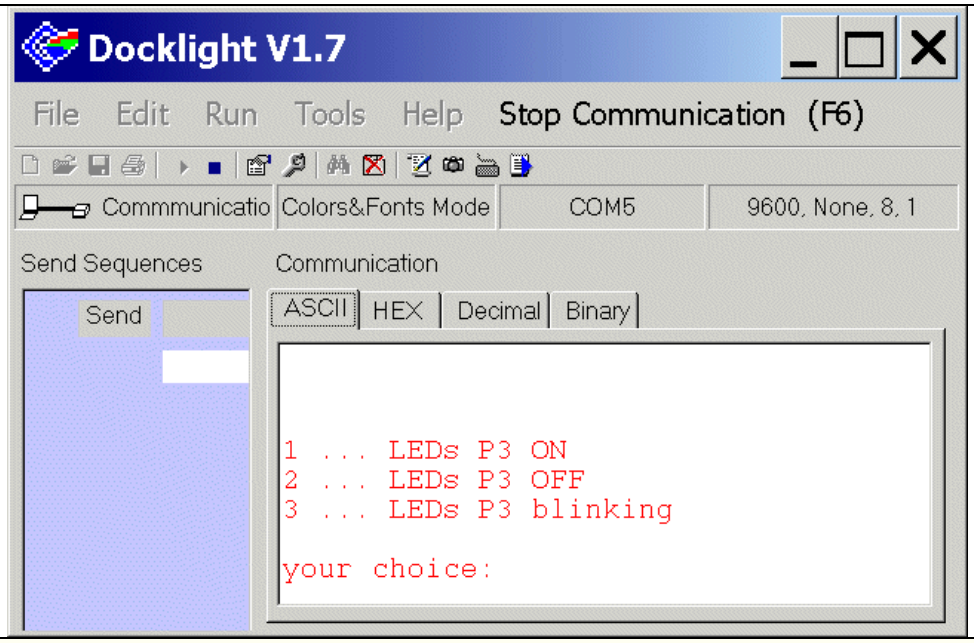


XC888CLM-8FFA functional units (Source: User's Manual)



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

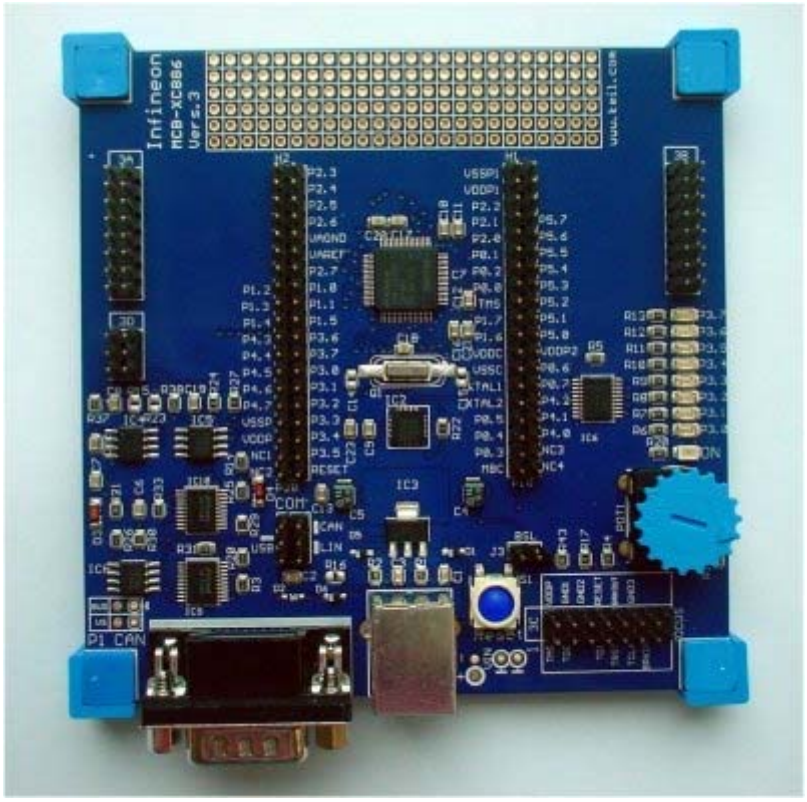
“Cookery-book“
For your first programming example for the XC888 Starter Kit Board:

Your program:	
Chapter/ Step	***** Recipes *****
1.)	**XC888 Starter Kit Board** **Power Supply (via USB), Jumper Setting, Serial Connection (via USB) to the notebook**
2.)	**DAvE (program generator)** **DAvE Installation (mothersystem) + DAvE Update Installation (XC888.DIP) for XC888**
3.)	**Using DAvE** **Microcontroller initialization for your programming example**
4.)	**Using DAvE Bench** **Programming of your application (hello world) with DAvE Bench tool chain**
5.)	**Using the debugger (DAvE Bench)**

Feedback

6.)	**Feedback**

1.) XC888 Starter Kit Board:



Ordering information:

Starter Kits – Type		µC	Order No.
SK-XC886/888LM		SAK-XC888CM	B158-H8743-X-X-7600
SK-XC886/888CLM Easy Kit		SAK-XC888CM	B158-H8744-X-X-7600

Distribution Worldwide:
http://www.infineon.com/cms/en/corporate/company/location/index.html

Screenshot of the XC888 Starter Kit Homepage:

SK-XC886/888CLM Starter Kit - Infineon Technologies - Mozilla Firefox

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

http://www.infineon.com/cms/en/product/channel.html?channel=db3a

Erste Schritte Aktuelle Nachrichten

Never stop thinking

Home | Sitemap | Select Language | Login Search Go About Infineon

Get Product information Select a Category Services for Engineers Select Services

Microcontrollers

Home > Microcontrollers > Development Tools, Software and Training > C500/C800 XC800 Development Tools and Software > Starter Kits, Evaluation Boards and Application Kits > SK-XC886/888CLM Starter Kit

Print Page Send Page

SK-XC886/888CLM Starter Kit

MCU Derivatives: SAK-XC886CM

CPU Clock: 24 MHz

On-Chip Memory:
- 1792 Byte RAM,
- 32 kByte Flash (incl. up to 8kByte data flash)

Interfaces:
- USB Connector for power supply, UART communication, and flash downloading,
- LIN via Header,
- CAN0 via Header and CAN1 via 9 Pin (male) D-Sub,
- JTAG via Header,
- Motor control connector via Header.

Includings:
- Cables and USB-to-JTAG Debugger-Box,
- 4 port USB Hub,
- Technical Documentation:e.g. user manuals (CD),
- Free unlimited assembly debugger,
- Evaluation Versions of development Tools: e.g. Compiler, Debugger, DAvE(CD).

To download the latest version of the CD content, please click here

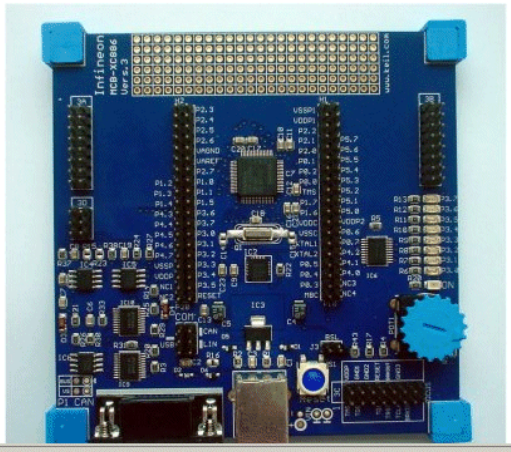
Order Nr.: B158-H8743-X-X-7600

Price: 150,- EUR

How to order?
To order your kit, please click here .

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

Where to buy
Please use our location finder to get in contact with your nearest Infineon distributor or sales office
Find a location



Fertig

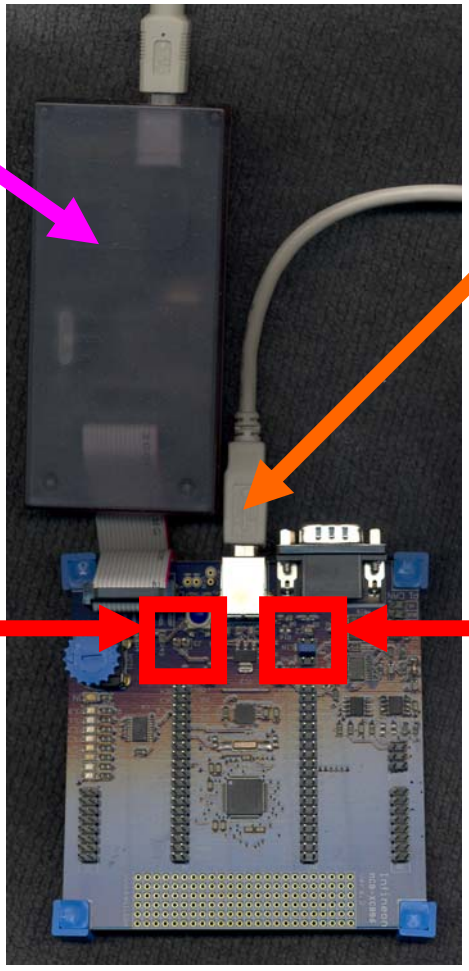
Overview of the XC888 Starter Kit Board connection to the environment:

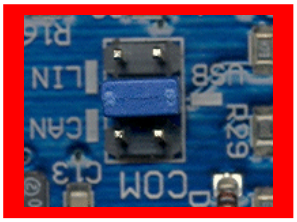
Note:
Do not connect now!
This is just information! We are going to connect the board later!

Reason:
When the TANTINO-XC800-DEBUGGER-BOX is already connected to the Starter Kit Board, the Starter Kit Board must be supplied with power for the TANTINO-XC800-DEBUGGER-BOX to work properly.
For the power supply we are going to use the USB cable – by connecting the USB cable a USB driver is needed.

TANTINO-XC800-DEBUGGER-BOX
used for: On-Chip-Flash-Programming and Debugging (JTAG).

USB Cable
used for: UART communication (the RS232 serial interface is available via USB); the USB connection works also as the power supply.





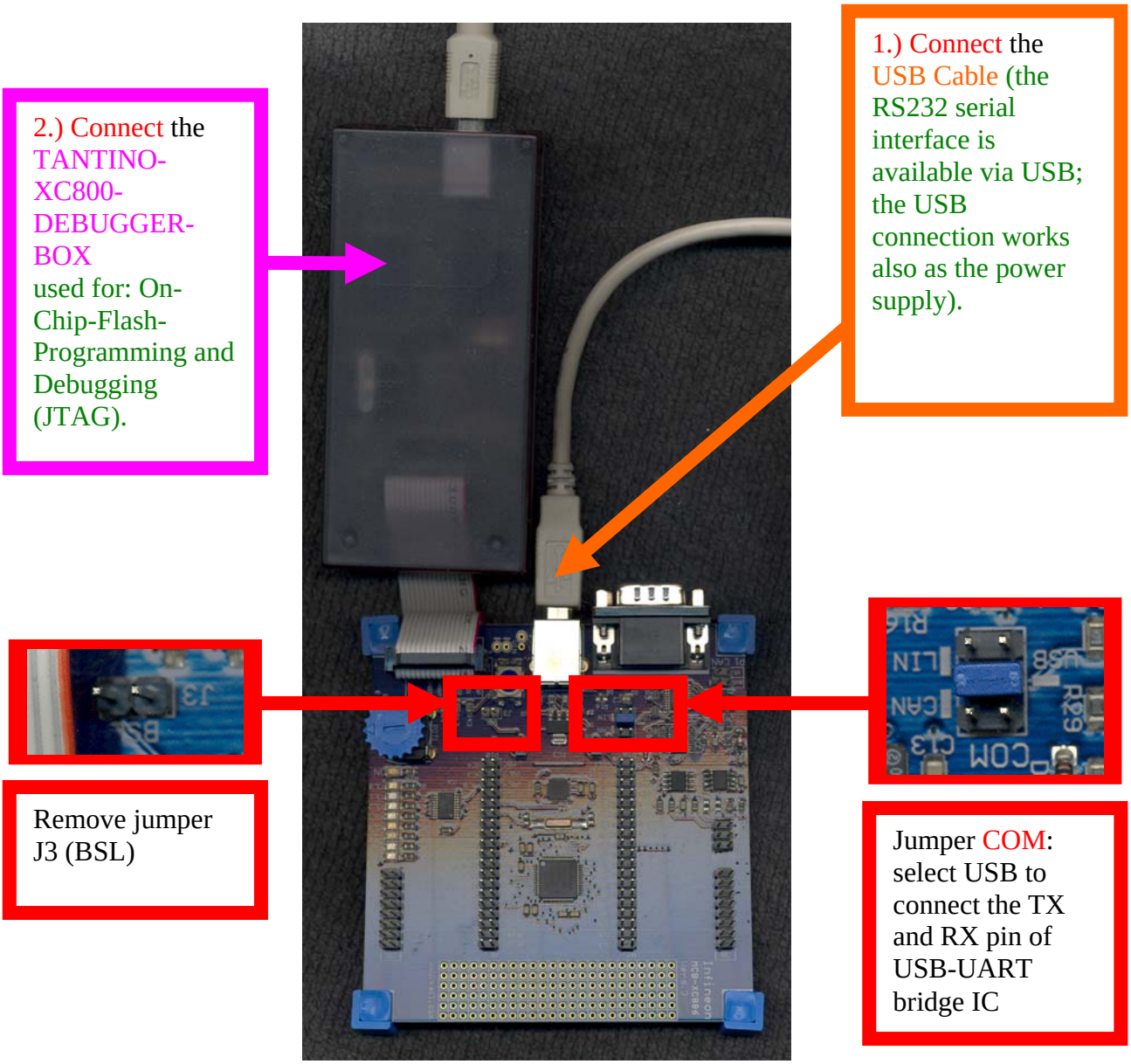
Remove jumper J3 (BSL)
➔ Open:
Bootstrap loader is disabled or User mode enabled

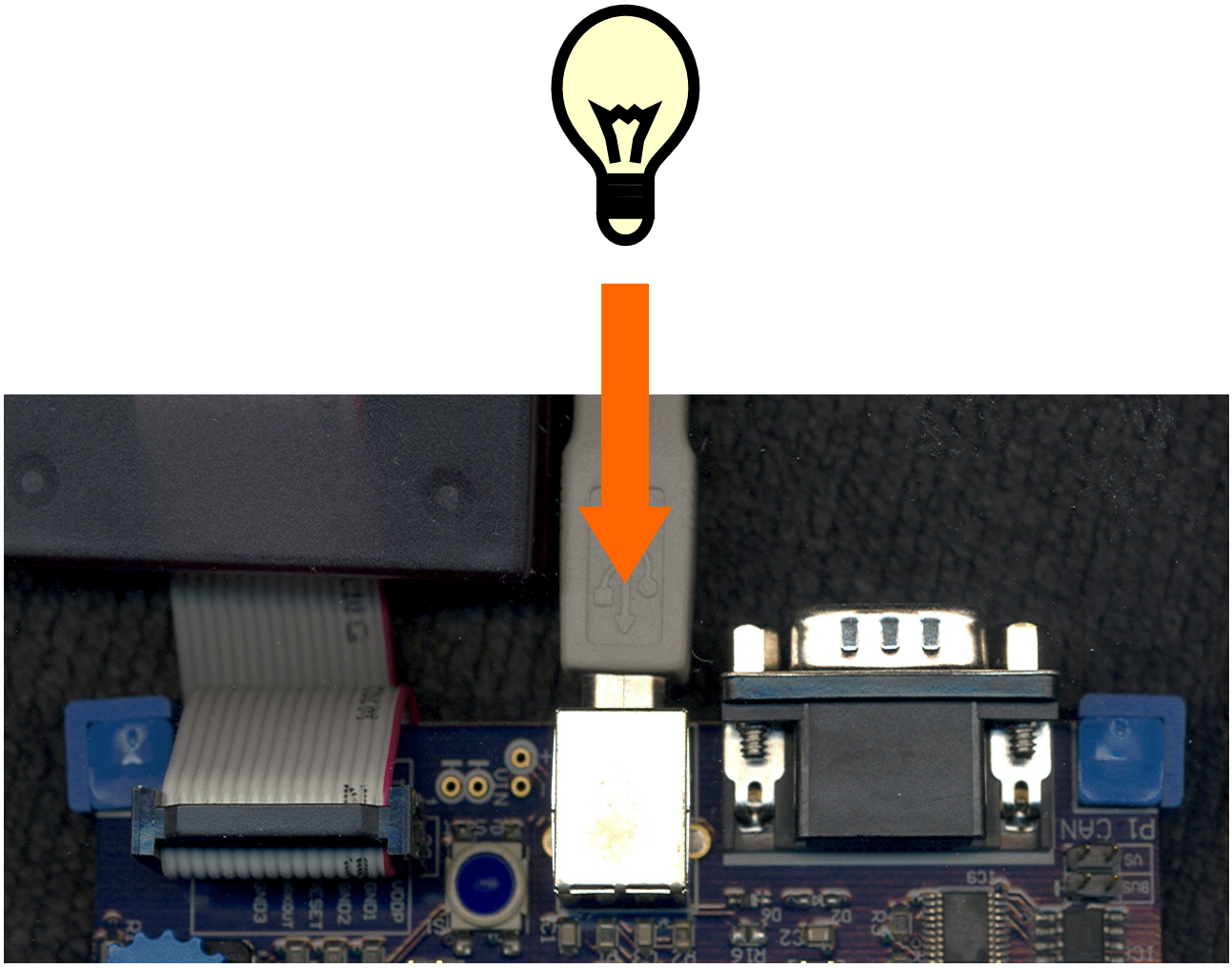
Jumper COM:
select USB to connect the TX and RX pin of USB-UART bridge IC

Note:
For further information, please refer to the XC888 Board Manual, V2.1, Sept 2006 .

Using TANTINO-XC800-DEBUGGER-BOX [used for: OnChipFlash-Programming and Debugging (using the JTAG interface)]:

Connecting the XC888-Board to the Environment:

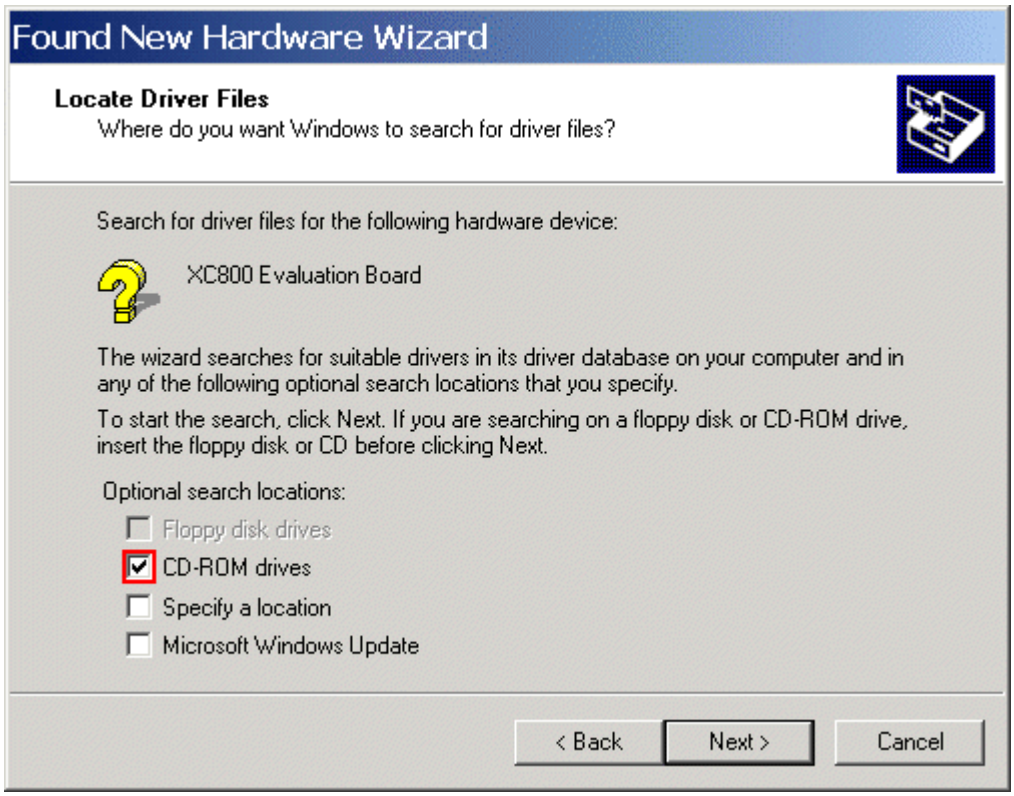




Note:
A USB driver is needed the first time while connecting the Starter Kit Board via the USB cable with your computer.
Therefore a pop-up window might appear to prompt for a driver:

If so: click Next

If so: click Next

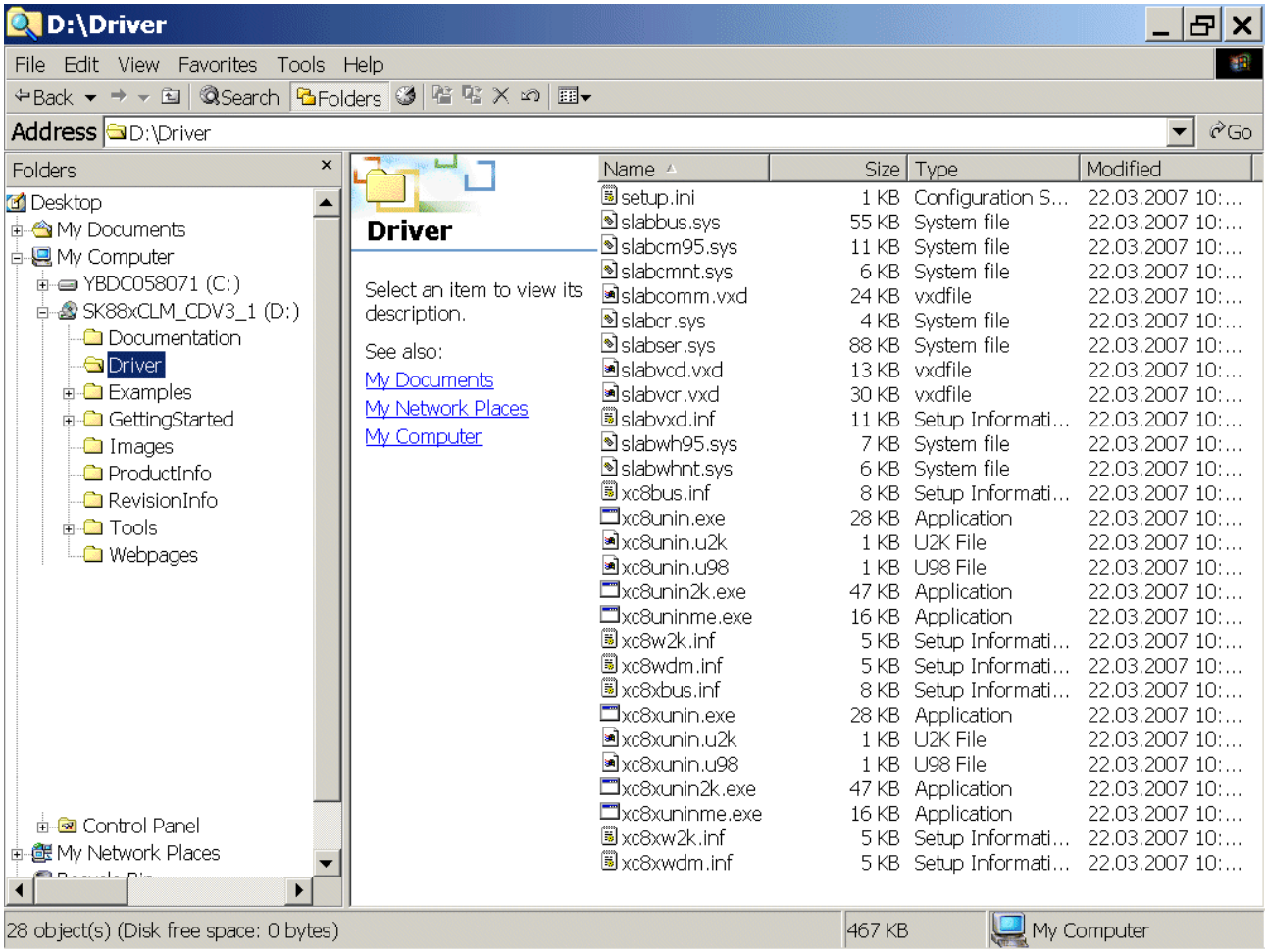


If so, please insert the XC88x Starter Kit CD check ☑ CD-ROM drives and click Next

If so: click Next

If so: click Finish

Or select the USB driver from the directory SK88xCLM_CDV3_1\Driver of your XC88x Starter Kit CD:

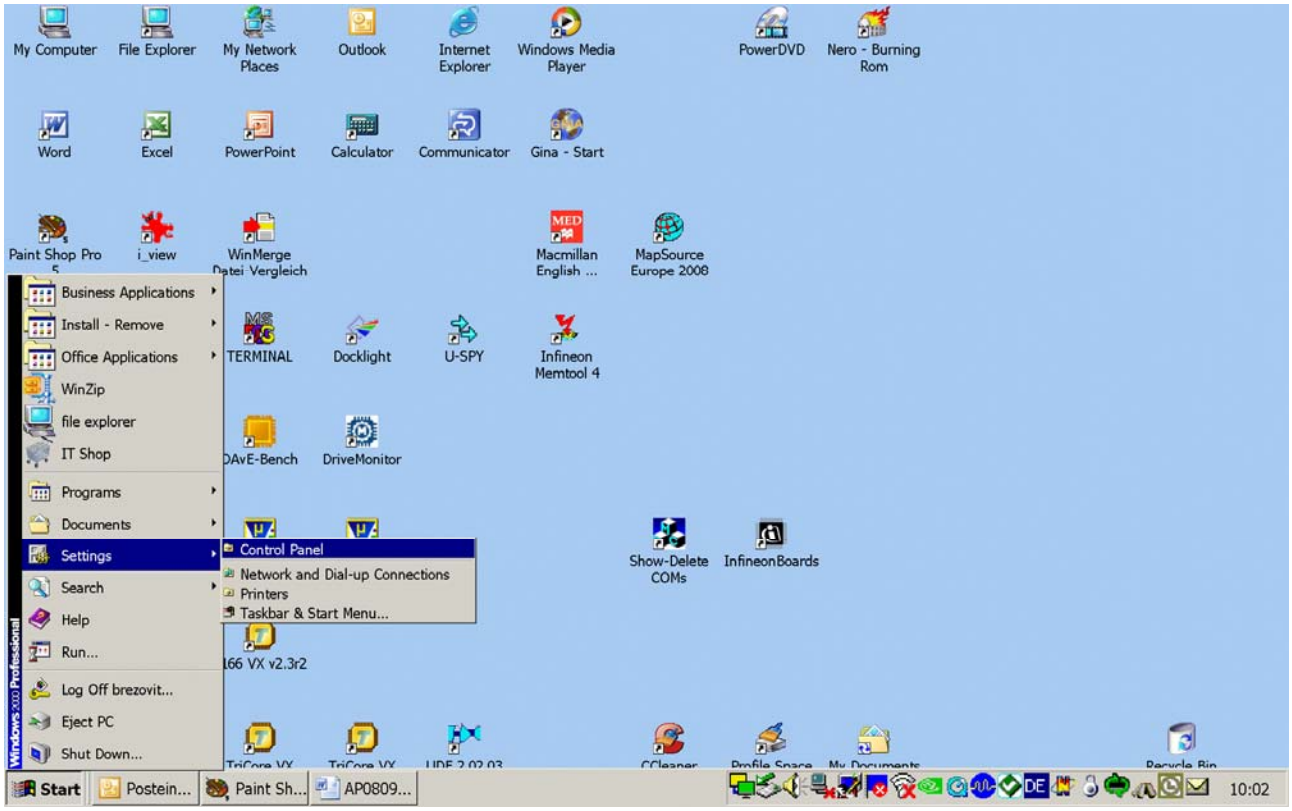


Note:
Skip this step when the USB driver is auto-detected and auto-installed.

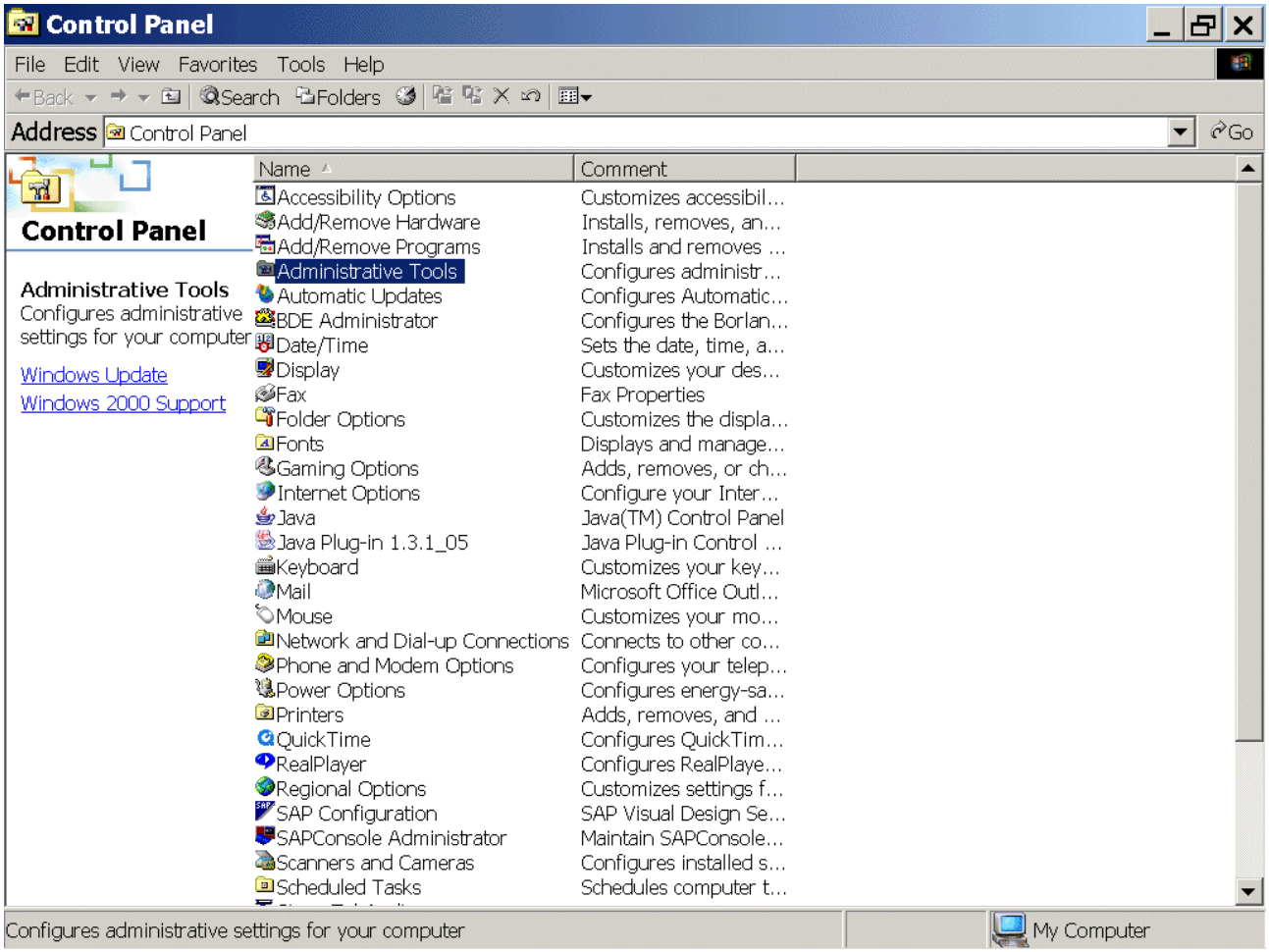
Note:
A default COM Port is generated after the USB driver is installed.

Using a Windows 2000 operating system, we are now going to search for the COM Port which was generated after connecting our XC888 Evaluation Board:

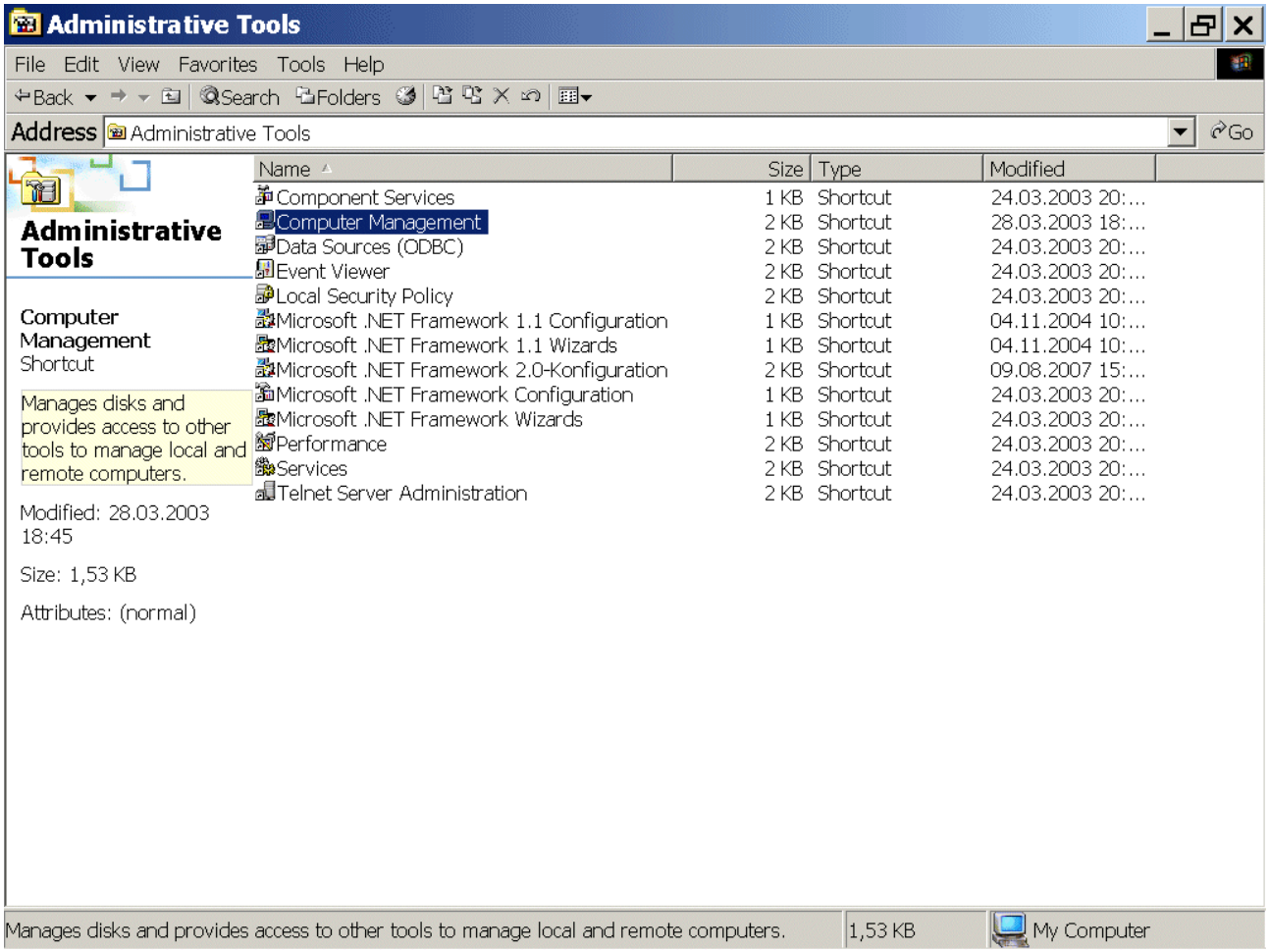
Start – Settings – Control Panel



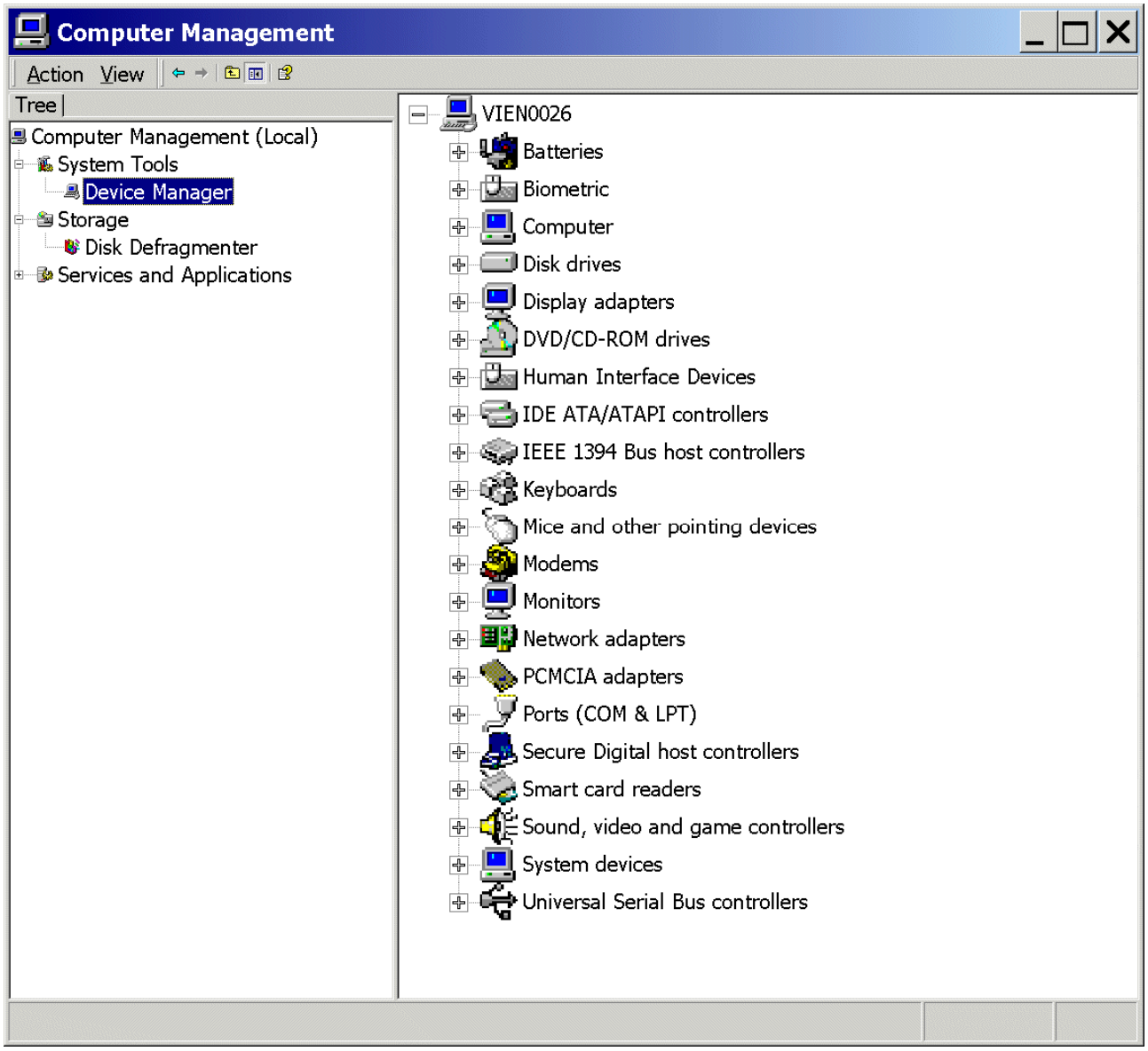
Double click: Administrative Tools



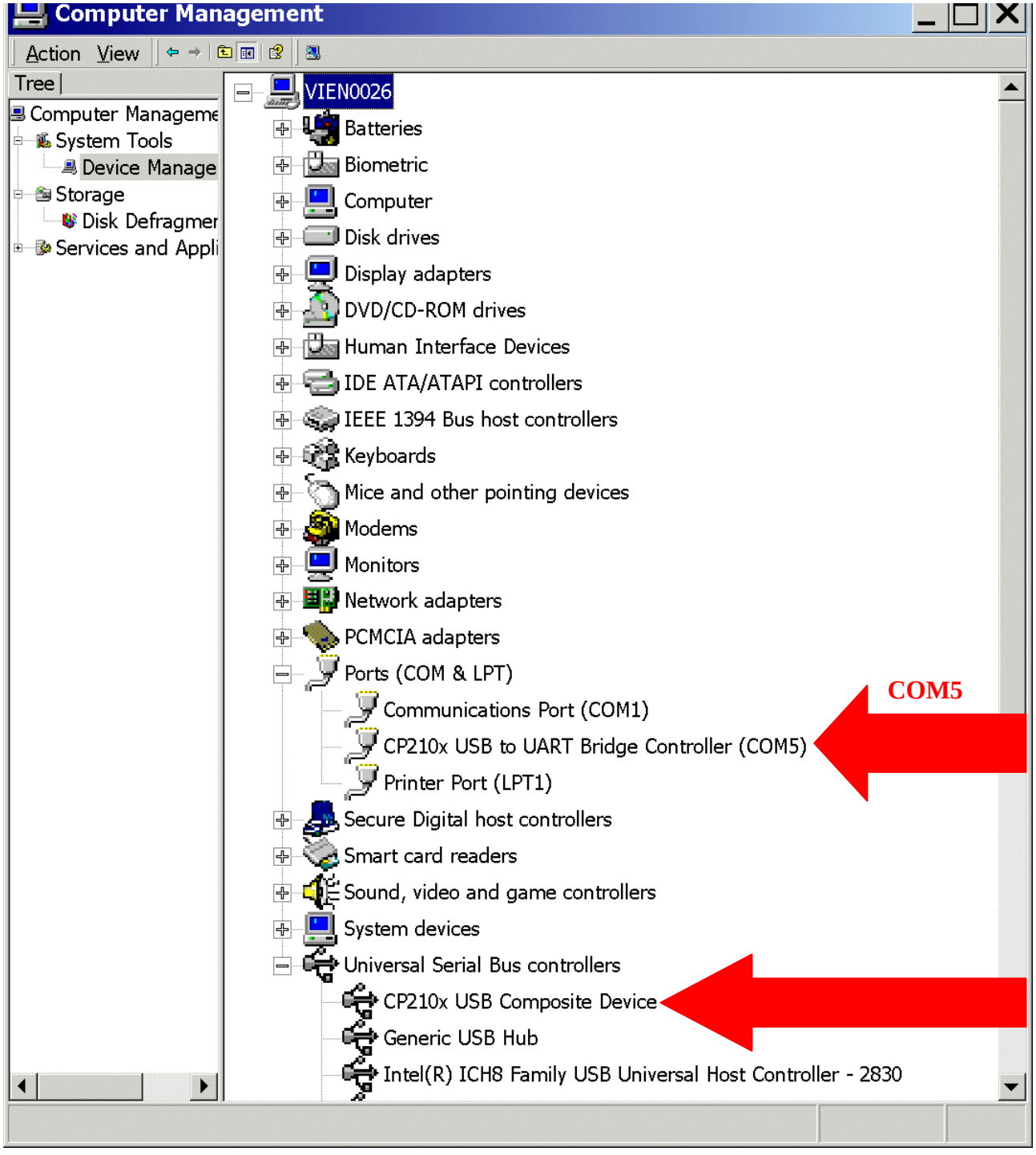
Double click: Computer Management



Click/Select: Device Manager



Expand: Ports (COM & LPT):
Expand: Universal Serial Bus controllers:



Note:
As we can see in the screenshot above:
our COM Port for UART/RS232 communication with the Starter Kit Board via USB is COM5 !

Additional information:
Using a SILICON LABS CP2102 ”Single-Chip USB To UART Bridge”:

Note:
IC2 soldered on the XC888 Starter Kit is a Silicon Labs CP2102 chip (Single-Chip USB To UART Bridge) using Virtual COM Port Device Drivers.
Using Virtual COM Port drivers, the data format and baud rate are set during COM port configuration on the PC.

Supported Data Formats and Baud Rates (Source: CP2102 Data Sheet):

Data Bits	5, 6, 7, and 8
Stop Bits	1, 1.5[1], and 2
Parity Type	None, Even, Odd, Mark, Space
Baud Rates[2]	300, 600, 1200, 1800, 2400, 4000, 4800, 7200, 9600, 14400, 16000, 19200, 28800, 38400, 51200, 56000, 57600, 64000, 76800, 115200, 128000, 153600, 230400, 250000, 256000, 460800, 500000, 576000, 921600[3]
Notes: 3. 5-bit only. 4. Additional baud rates are supported. See “AN205”. 5. 7 or 8 data bits only.	

The CP2102 Virtual COM Port (VCP) device drivers allow a CP2102-based device to appear as a COM port to the PC’s application software.
The application software (e.g. Docklight) running on the PC accesses the CP2102-based device as it would access a standard hardware COM port.

Every CP2102 device is delivered with a unique Serial Number making it possible to use more than one XC888 Starter Kit at the same time.
That means every Starter Kit gets its own Virtual COM Port.

Note:
For further information, please refer to the XC888 Board Manual, V2.1, Sept 2006 .
For further information, please refer to the SILICON LABS CP2102 Datasheet

2.) DAvE – Installation for XC888 microcontrollers:

Install DAvE (mothersystem):

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

Title	Date	Version	Size
Tool Package			
DAvE - Mothersystem (DAvE_Mothersystem_v2_2r1.zip)	14 Dec 2009	V2.2	8.8 MB
DAvE - Mothersystem (setup.exe)	14 Dec 2009	V2.2	8.9 MB

and execute setup.exe to install DAvE .

Install the XC888 microcontroller support/update (XC888CLM DIP file):

1.)
Download DAvE_XC888CLM_v1_6.zip (- or any higher version !!!)
- the DAvE-update-file (.DIP) for the required microcontroller @
http://www.infineon.com/DAvE:

Title	Date	Version	Size
Development Tools			
XC864 DIP file for DAvE (Microcontroller Configuration Tool)-latest version (DAvE_XC864_v1_2.zip)	14 Dec 2009	v1.2	5.6 MB
XC878 DIP file for DAvE (Microcontroller Configuration Tool)-latest version (DAvE_XC878CLM_v2_1.zip)	14 Dec 2009	v2.1	9.4 MB
XC866 DIP file for DAvE (Microcontroller Configuration Tool)-latest version (DAvE_XC866_v2_2.zip)	14 Dec 2009	v2.2	5 MB
XC878CLM DIP file for DAvE (Microcontroller Configuration Tool) (XC878CLM.zip)	08 Jul 2008	v1.1	9.1 MB
XC888CLM DIP file for DAvE (Microcontroller Configuration Tool)-latest version (DAvE_XC888CLM_v1_6.zip)	14 Dec 2009	v1.6	7.6 MB
XC886CLM DIP file for DAvE (Microcontroller Configuration Tool)-latest version (DAvE_XC886CLM_v1_8.zip)	14 Dec 2009	v1.8	7.6 MB
XC866 DIP file for DAvE (Microcontroller Configuration Tool) (XC866_v2.0.zip)	08 Feb 2008	v2.0	4.9 MB
XC888CLM DIP file for DAvE (Microcontroller Configuration Tool), V1.3 (XC888CLM_v1.3.zip)	14 Jan 2008	V1.3	7.6 MB
XC866 DIP file for DAvE (Microcontroller Configuration Tool) (XC866_v1.9.zip)	14 Jan 2008	V1.9	4.9 MB
XC886CLM DIP file for DAvE (Microcontroller Configuration Tool), V1.5 (XC886CLM_v1.5.zip)	14 Jan 2008	V1.5	7.5 MB
XC888CLM DIP file for DAvE (Microcontroller Configuration Tool), V1.1 (DAvE_XC888CLM_v1.1.zip)	01 Mar 2007		7.5 MB
XC886CLM DIP file for DAvE (Microcontroller Configuration Tool), V1.3 (XC886CLM_v1.3.zip)	22 Mar 2007		7.5 MB
DAvE XC888 RELEASE NOTES (DAvE_XC888_RELEASE_NOTES.pdf)	24 May 2007		351 KB
XC886CLM DIP file for DAvE (Microcontroller Configuration Tool), V1.1 (XC886CLM_v1.1.zip)	01 Sep 2006		6.4 MB

Unzip the zip-file "DAvE_XC888CLM_v1.6.zip"
(- or any higher version !!!)

and save "XC888CLM_v1.6.dip "
@ e.g. D:\DAvE\XC888CLM_v1.6.dip.

2.)

Start DAvE - (click 

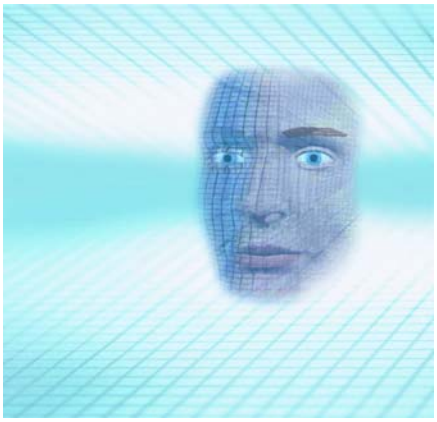
)

3.)

View
Setup Wizard
Default: • Installation
Forward>
Select: • I want to install products from the DAvE's web site
Forward>
Select: D:\DAvE
Forward>
Select: Available Products
click ✔ XC888CLM
Forward>
Install
End

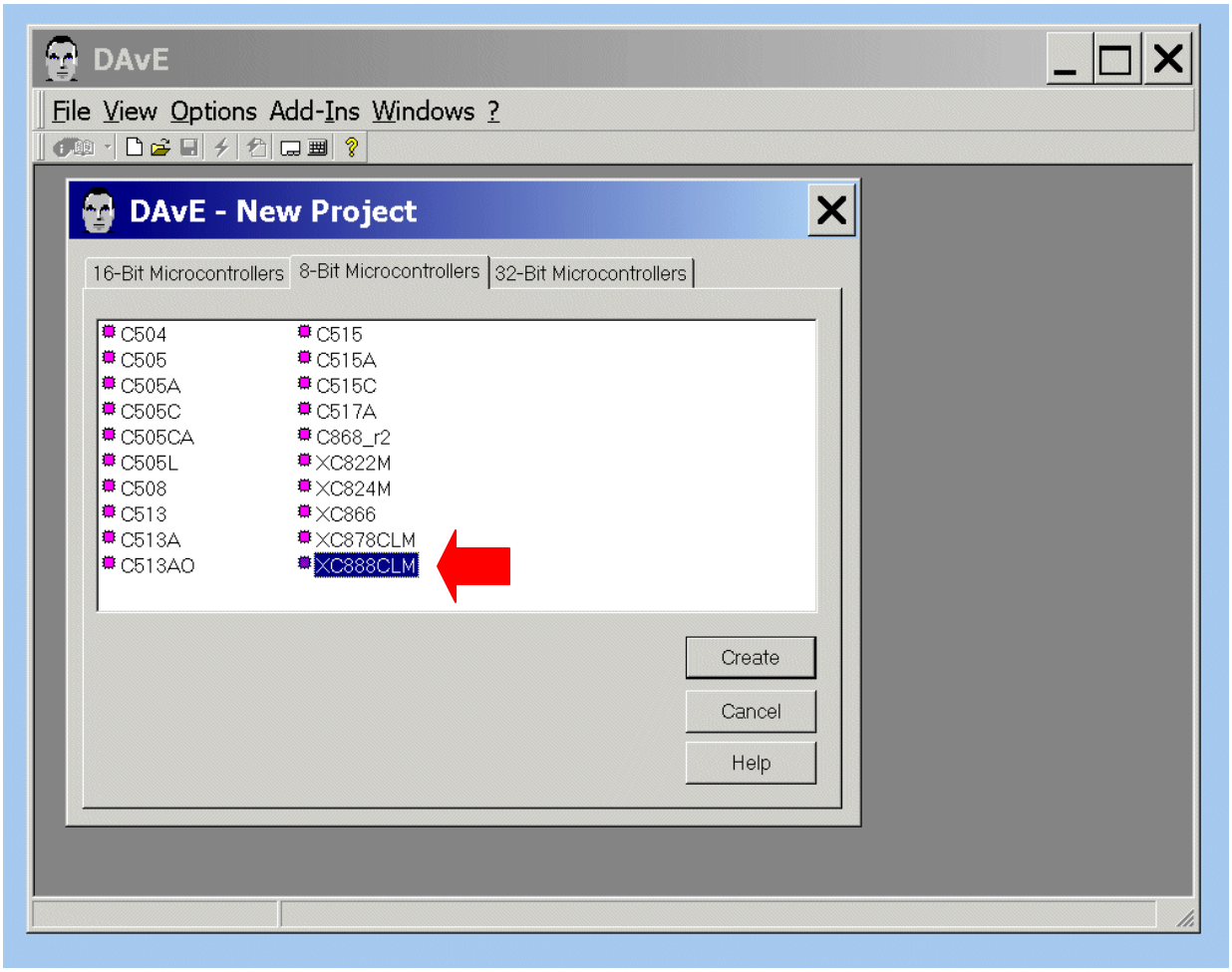
4.) DAvE is now ready to generate code for the XC888CLM microcontroller.

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



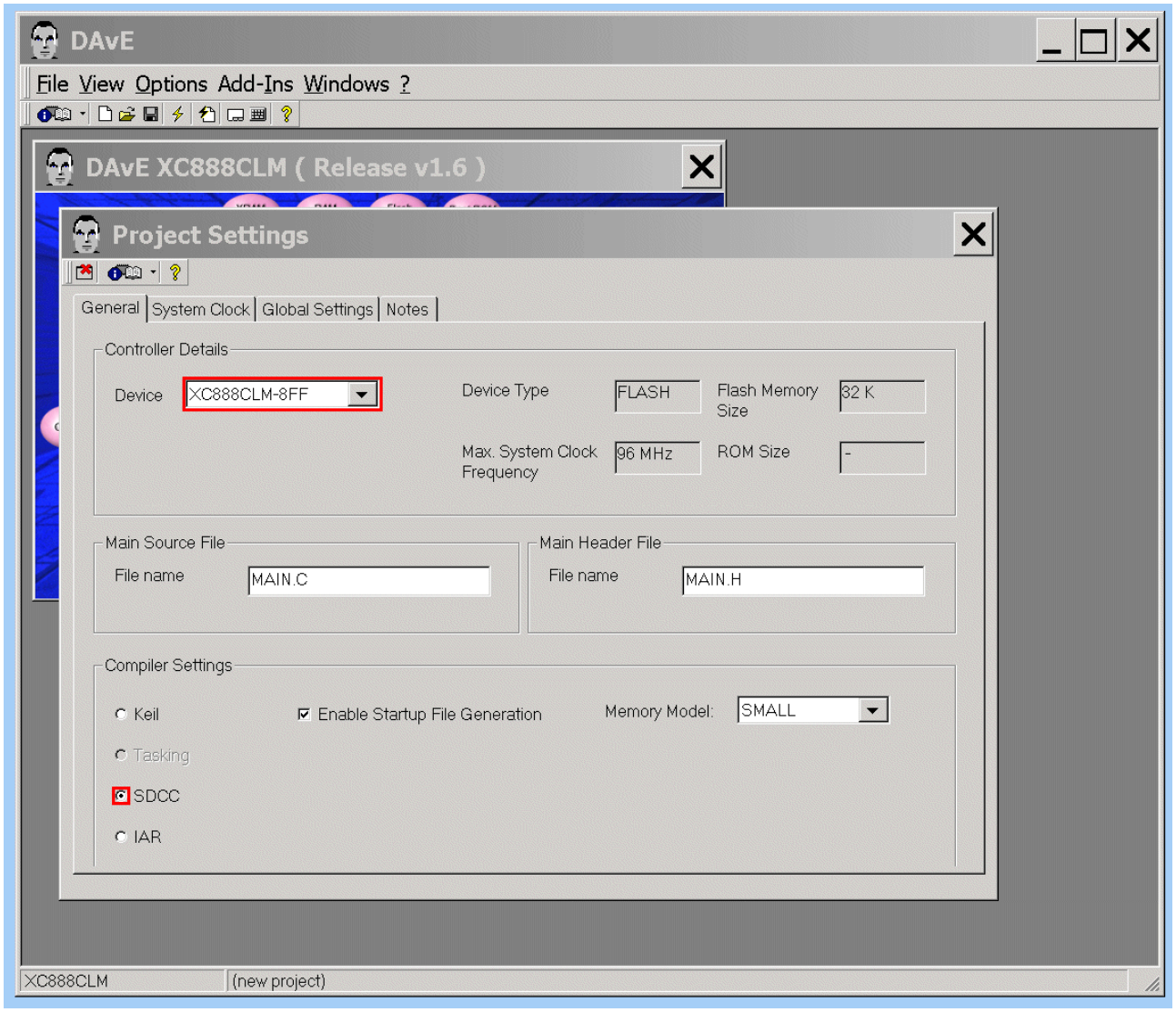
Start the program generator DAvE and select the XC888CLM microcontroller:

File
New
8-Bit Microcontrollers
Select XC888CLM
Create

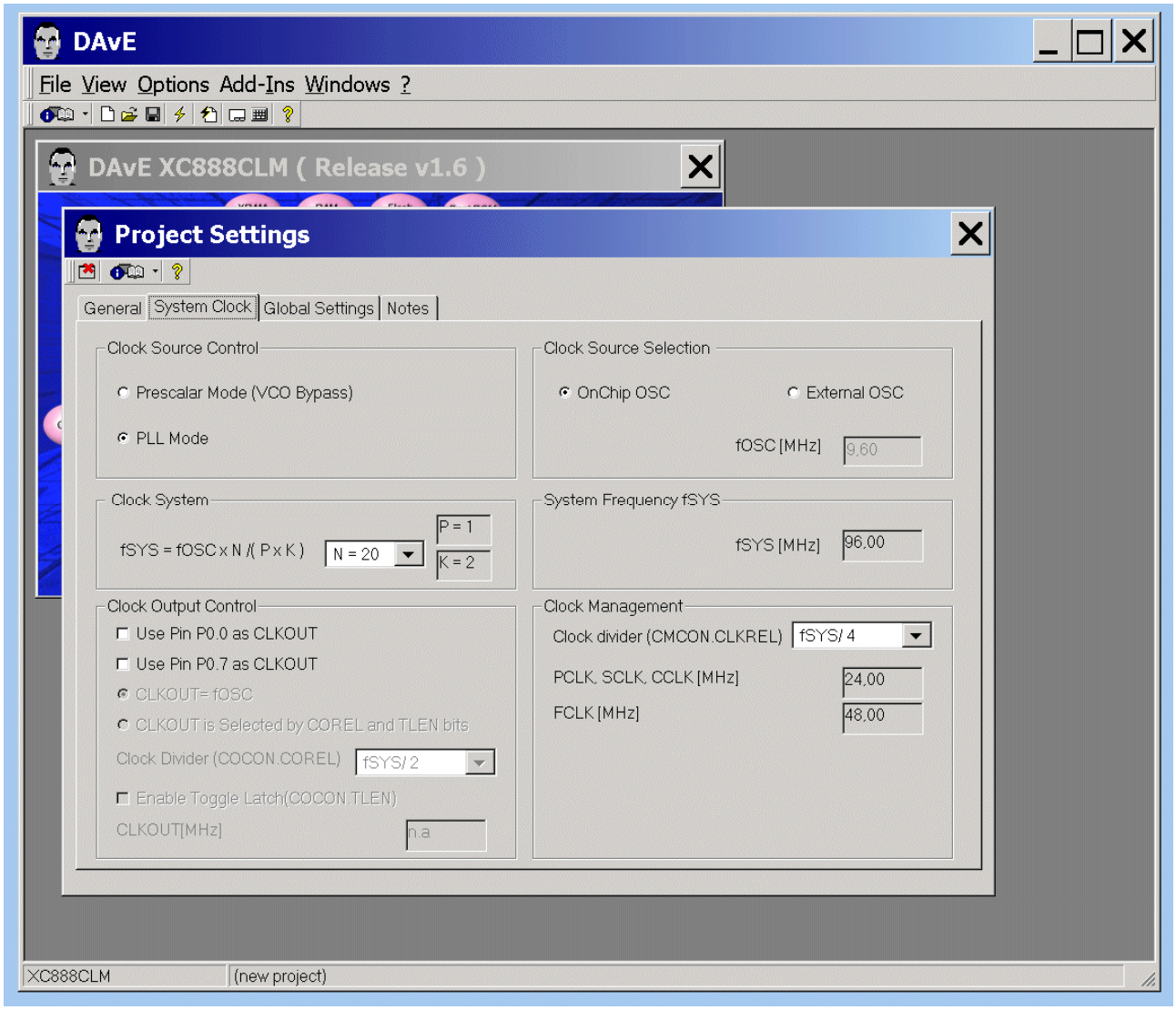


Choose the Project Settings as you can see in the following screenshots:

General: Controller Details: Device: check/select XC888CLM-8FF
General: Compiler Settings: For DAvE-Bench check/choose ⊙ SDCC (DAvE Bench)



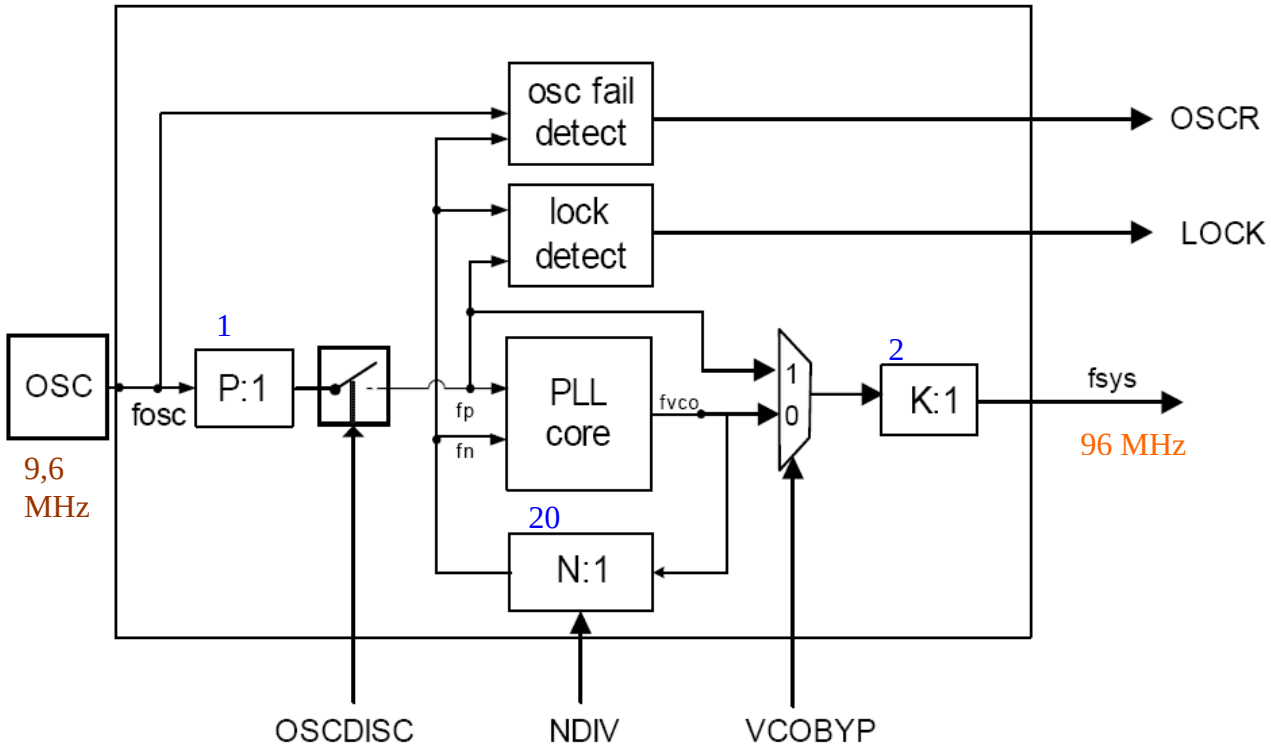
System Clock: (do nothing)



Note:
CPU clock is 24 MHz.

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

Clock Generation Unit (CGU) Block Diagram

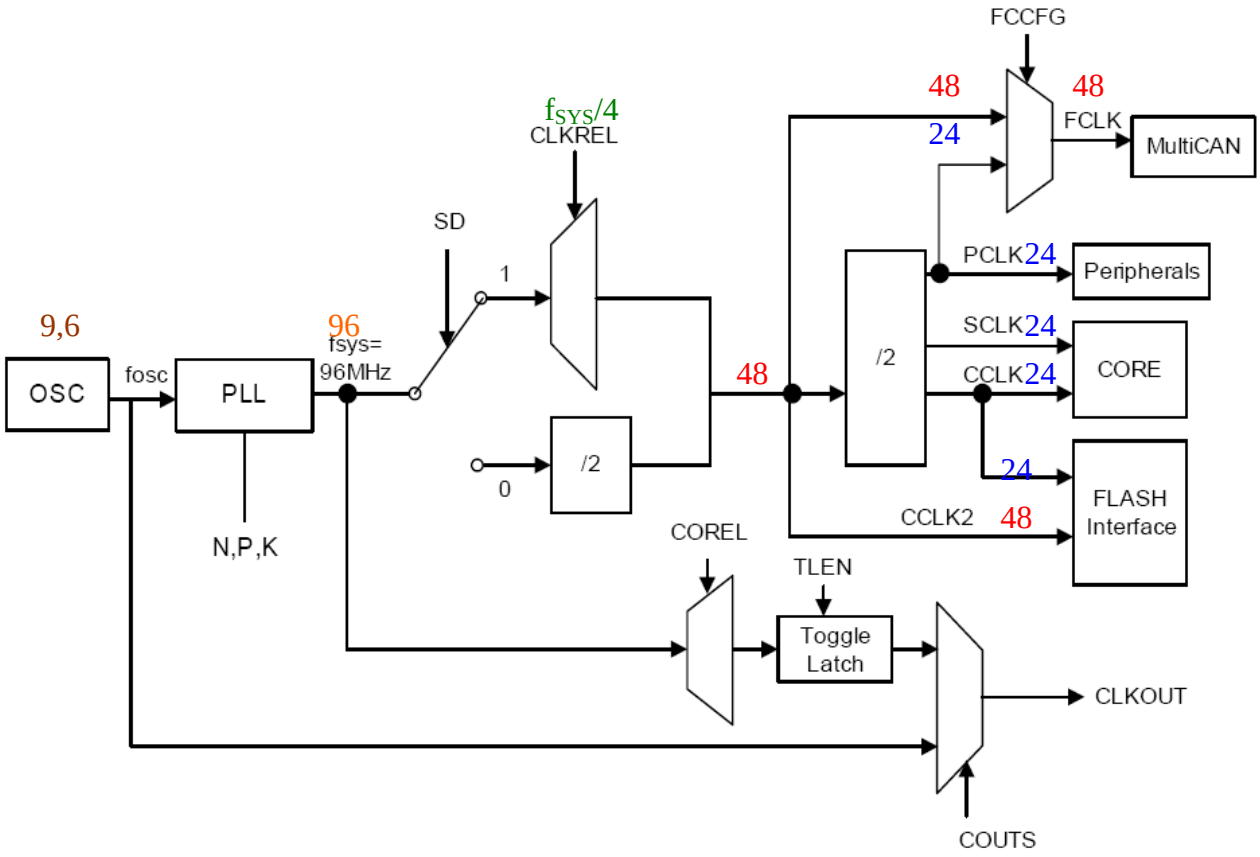


Note:

$f_{SYS} = f_{OSC} * N / (P * K) = 9,6 \text{ MHz} * 20 / (1 * 2) = 96 \text{ MHz}$

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

Clock Generation from fsys:



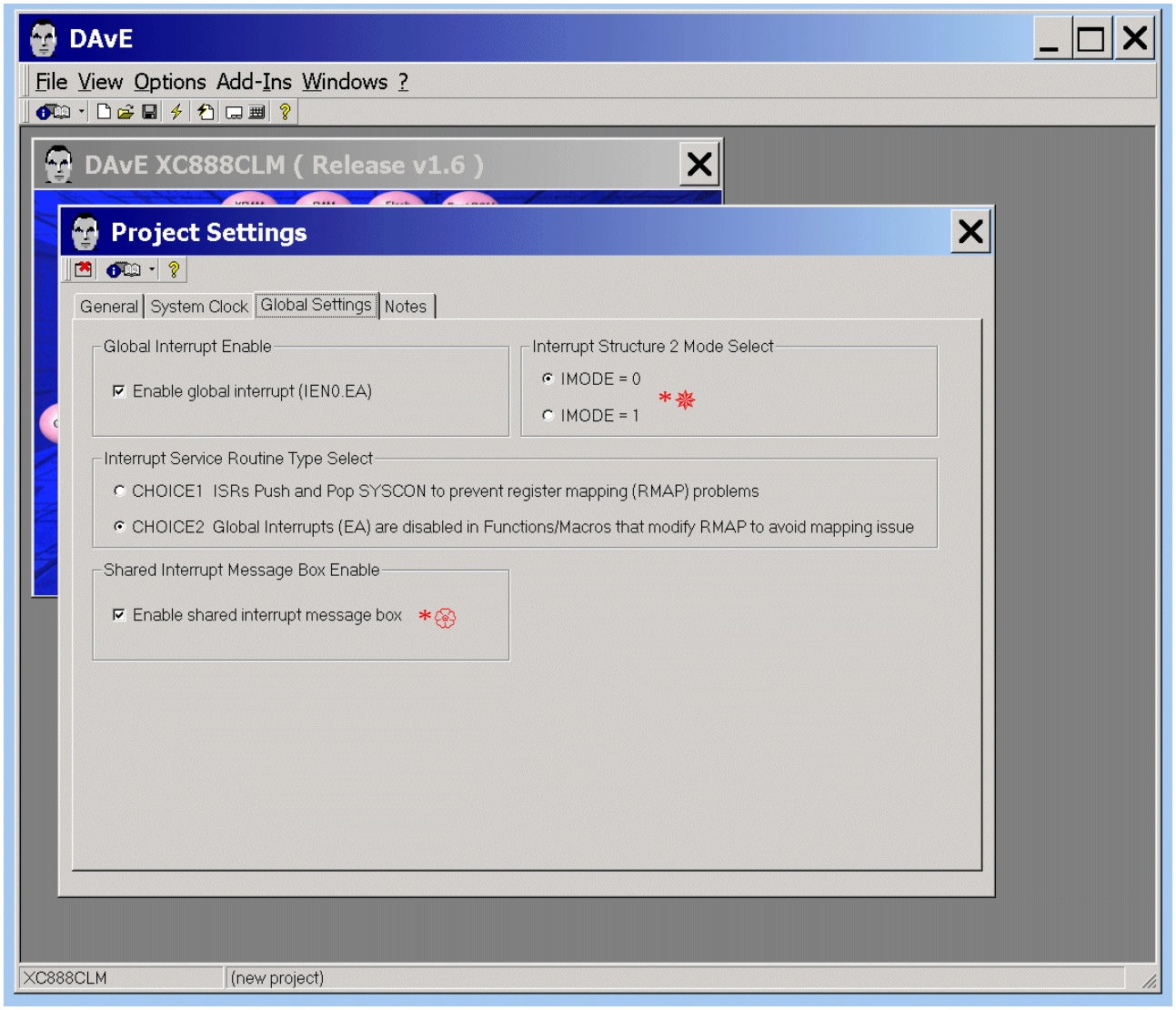
Note:

f_{SYS} = 96 MHz

CPU clock: CCLK, SCLK = 24 MHz
Fast clock: FCLK = 24 or 48 MHz
Peripheral clock: PCLK = 24 MHz
Flash Interface clock: CCLK2 = 48 MHz and CCLK = 24 MHz

CLKREL: The clock division factor $f_{SYS}/4$ (see DAvE screenshot page 17) is inclusive the fixed divider factor of 2.

Global Settings: (do not change configuration)



Note (Source: DAvE):

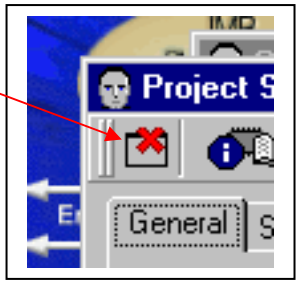
```
//  You have two choices for interrupt type select in Project Settings Page
//  under Global Settings Section.
//  If you select CHOICE 1 then ISR will be generated with push and pop.
//  If you select CHOICE 2 then ISR will be generated without push and pop.
//  Default choice is CHOICE 2.
//  Current selection is CHOICE 2
```

Note:

*❋ = Interrupt Structure 2 applies to Timer 2, Timer 21, UART1, LIN, external interrupts 2 to 6, ADC, SSC, CCU6, Flash, MDU, CORDIC and MultiCAN interrupt sources.
There is a slightly different behavior between MODE=0 and MODE=1 in setting/clearing the pending interrupt request bit.
*❁ = If an interrupt node is shared with another interrupt node, the ISR code will be generated in the SHARED_INT.C file.

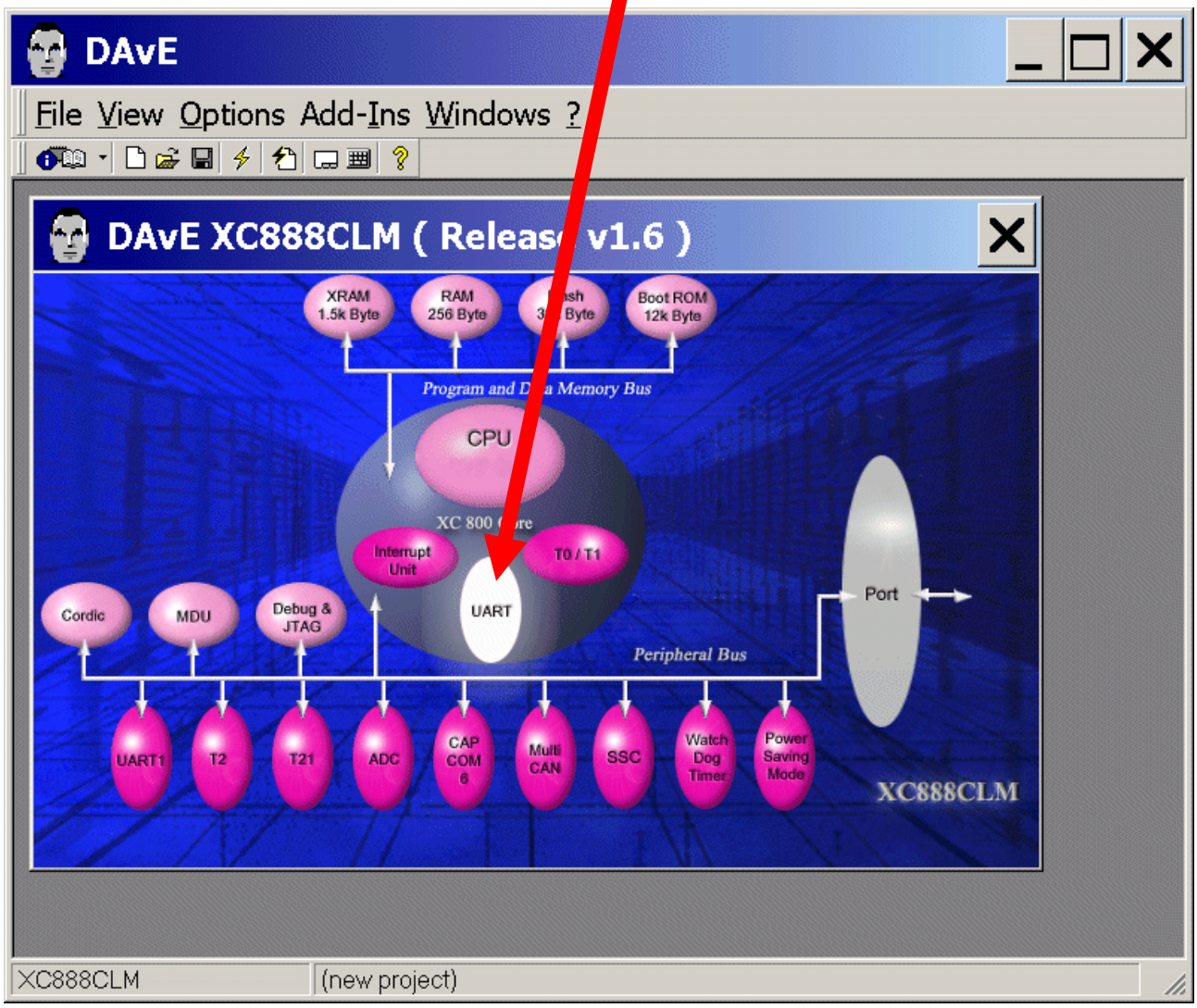
Notes: If you wish, you can insert your comments here.

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

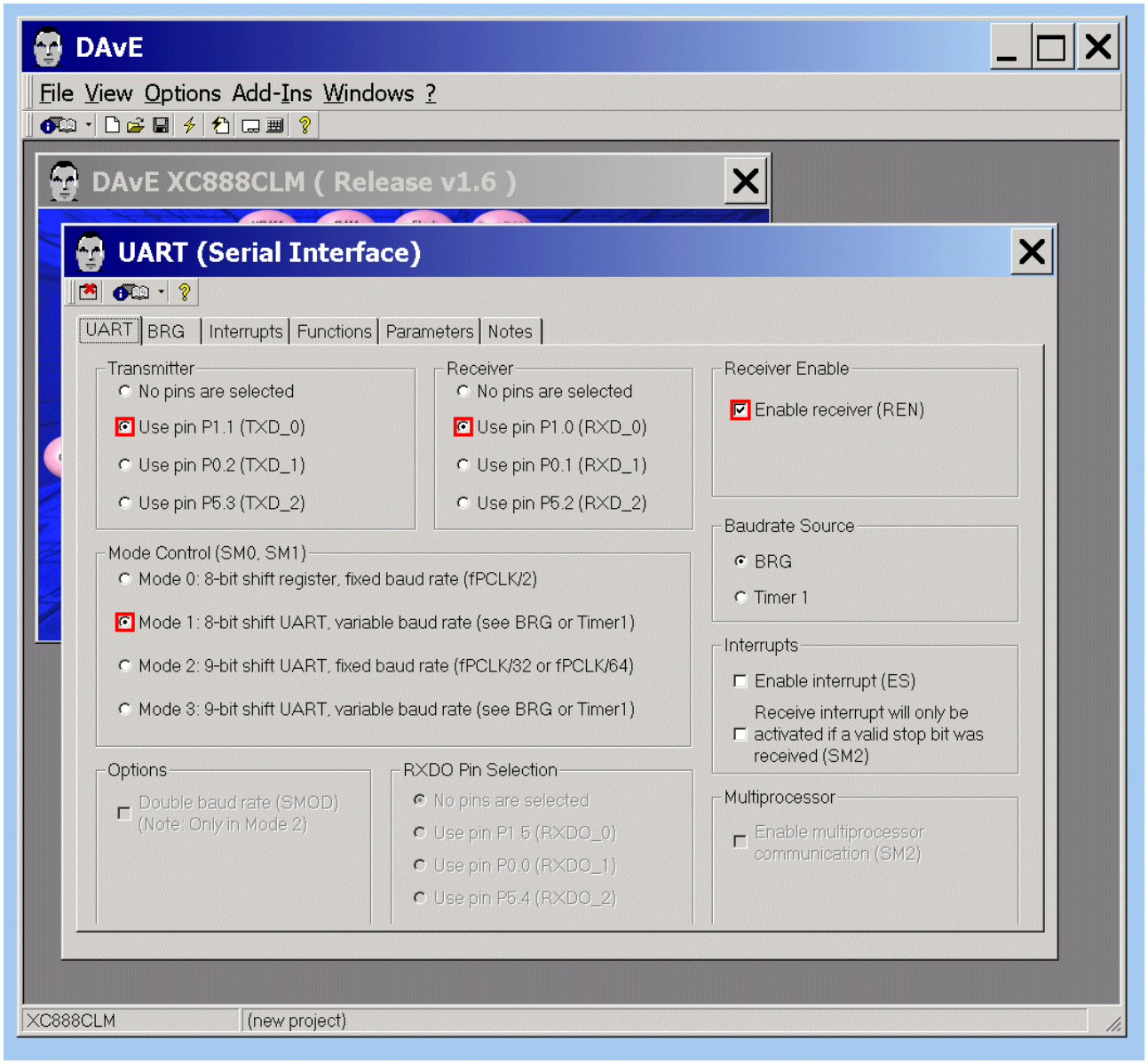


Configuration of the UART:

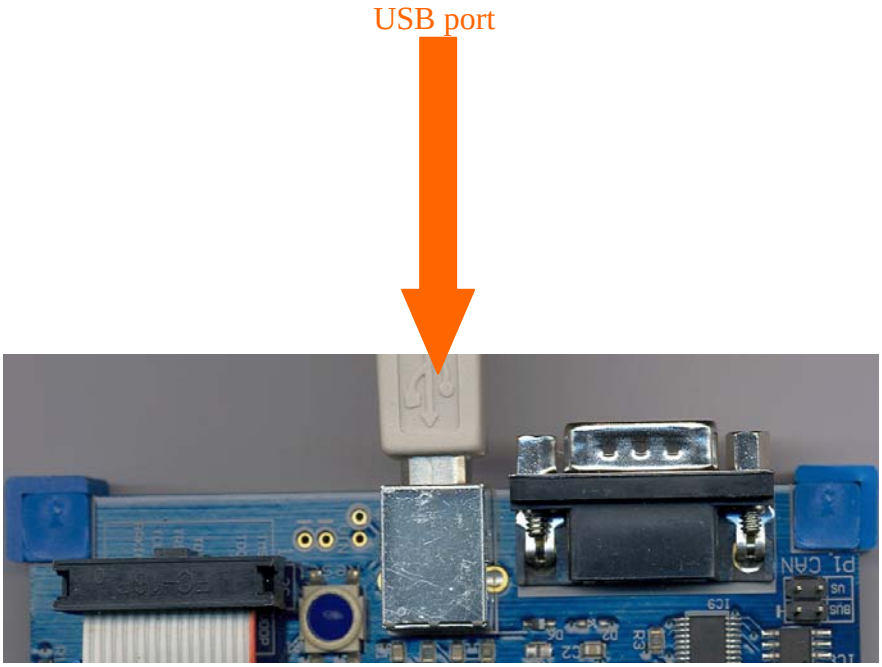
The configuration window/dialog can be opened by clicking the specific block/module (UART).



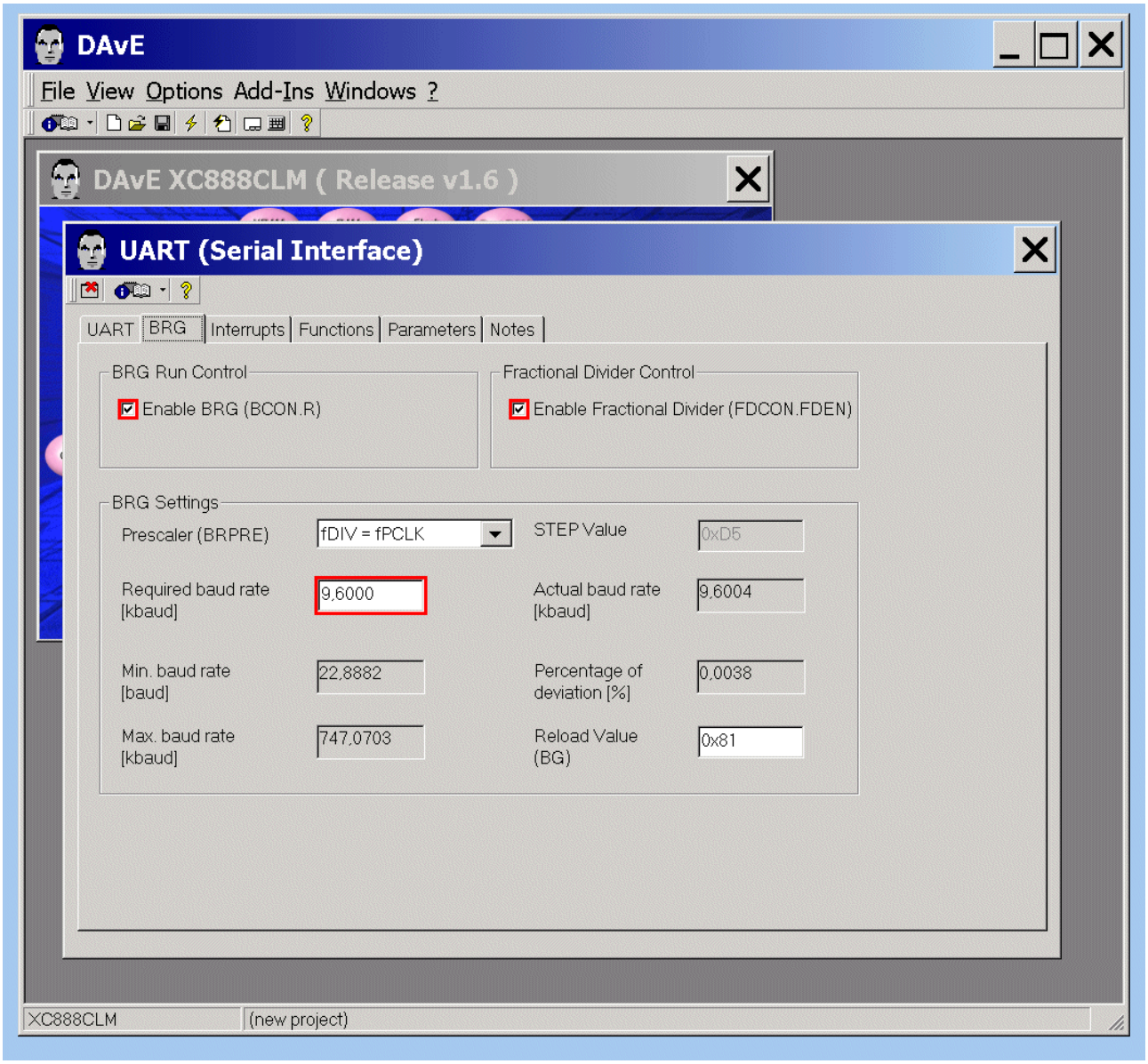
UART: Transmitter: click ⊙ Use pin P1.1 (TXD_0)
UART: Receiver: click ⊙ Use pin P1.0 (RXD_0)
UART: Receiver Enable: tick ✓ Enable receiver (REN)
UART: Mode Control: click ⊙ Mode 1: 8-bit shift UART, variable baud rate (see BRG or Timer1)



Note:
The RS232 serial interface (UART pins P1.0 and P1.1) is available via the USB port which converts the TTL-UART-signals to USB-signals
(using a SILICON LABS CP2102 ”Single-Chip USB To UART Bridge”).

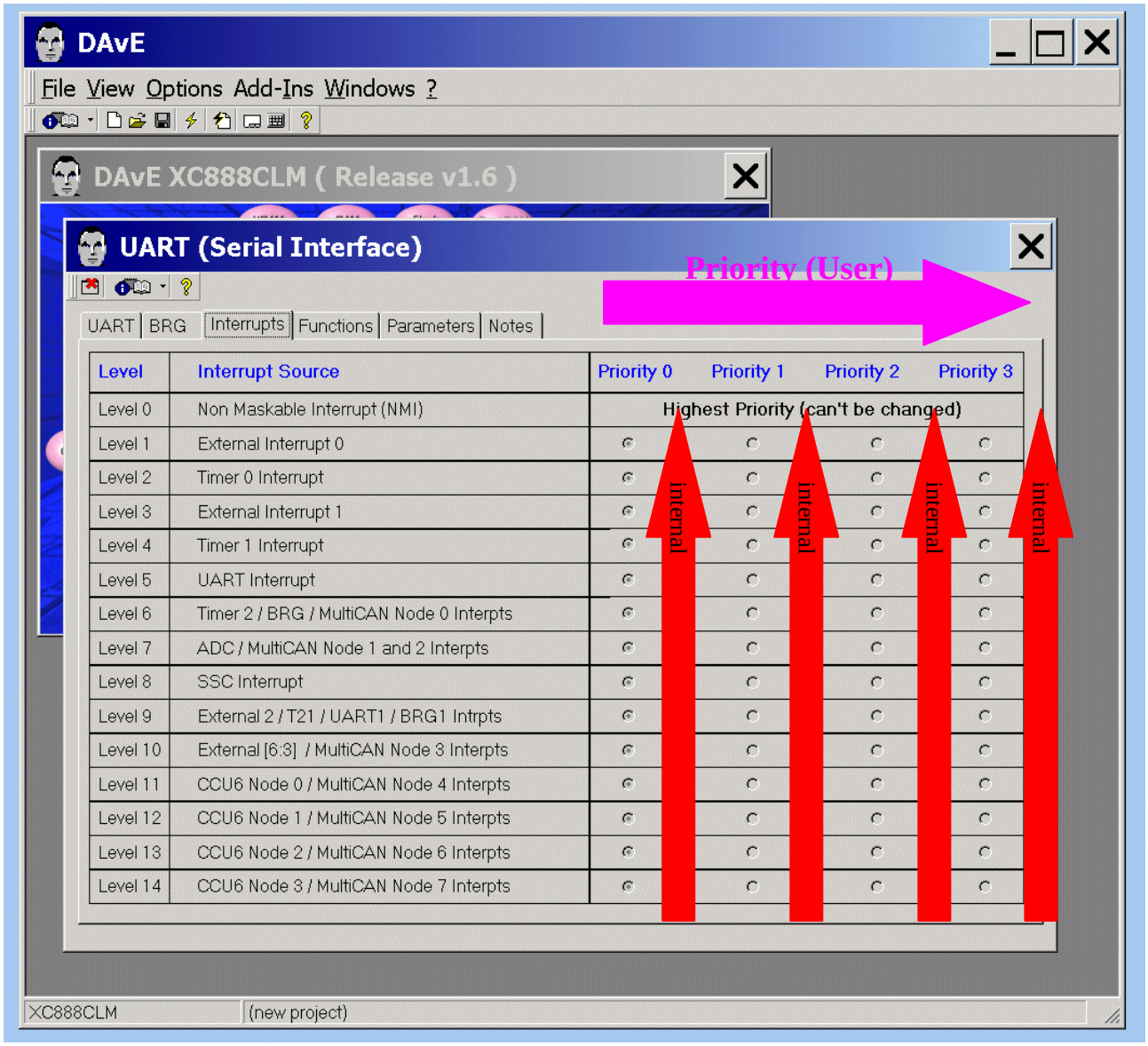


BRG: BRG Run Control: tick/check ✓ Enable BRG
BRG: Fractional Divider Control: tick/check ✓ Enable Fractional Divider
BRG: BRG Settings: Required baud rate [kbaud] insert 9,600 <ENTER>



Note:
Validate each alphanumeric entry by pressing ENTER.

Interrupts: (do nothing)



Note:
For the serial communication with a terminal program running on your Personal-Computer the printf / printf_small() / printf_fast_f() - function is used. The print function uses Software-Polling-Mode therefore we do not need to configure any interrupts.

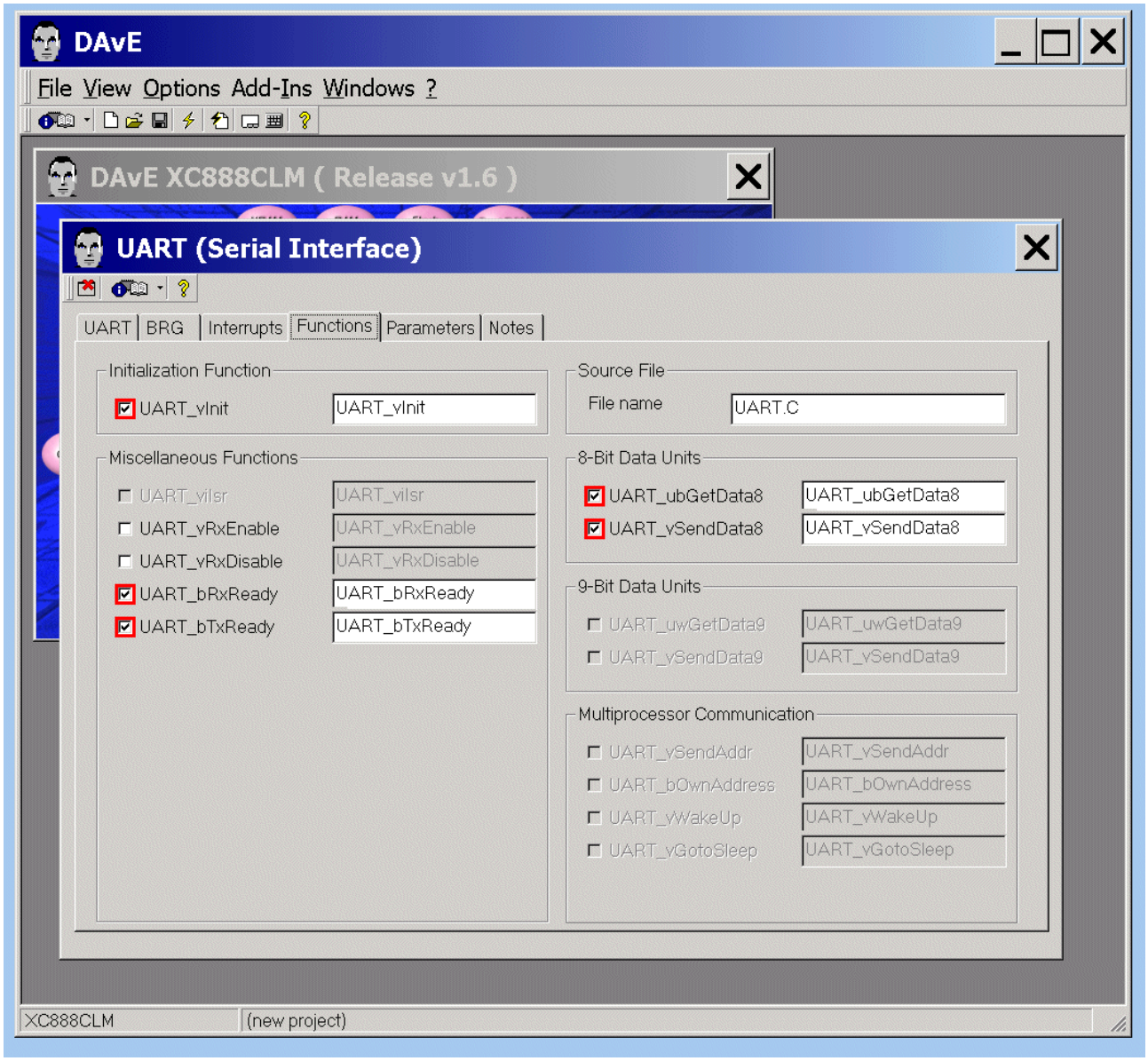
Interrupt Priorities (User):

Note (Source: Application Note AP08053):
There could be six interrupt priorities.
These priorities, with 6 beeing the highest, are as follows:

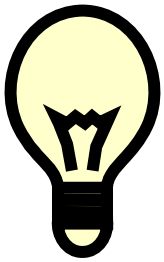
Interrupt Priority:	
6	NMI
5	Interrupt Priority 3
4	Interrupt Priority 2
3	Interrupt Priority 1
2	Interrupt Priority 0
1	Main

Main refers to routines that run prior to any interrupt and can be interrupted by any interrupt.
Each interrupt source can be programmed to any of the four interrupt priorities (0-3).
An interrupt that is currently being serviced can only be interrupted by a higher-priority interrupt, but not by another interrupt of the same or lower priority.
Hence, an interrupt of the highest priority cannot be interrupted by any other interrupt request.
In any case, the NMI always has the highest priority (above level 3) and its priority cannot be programmed.

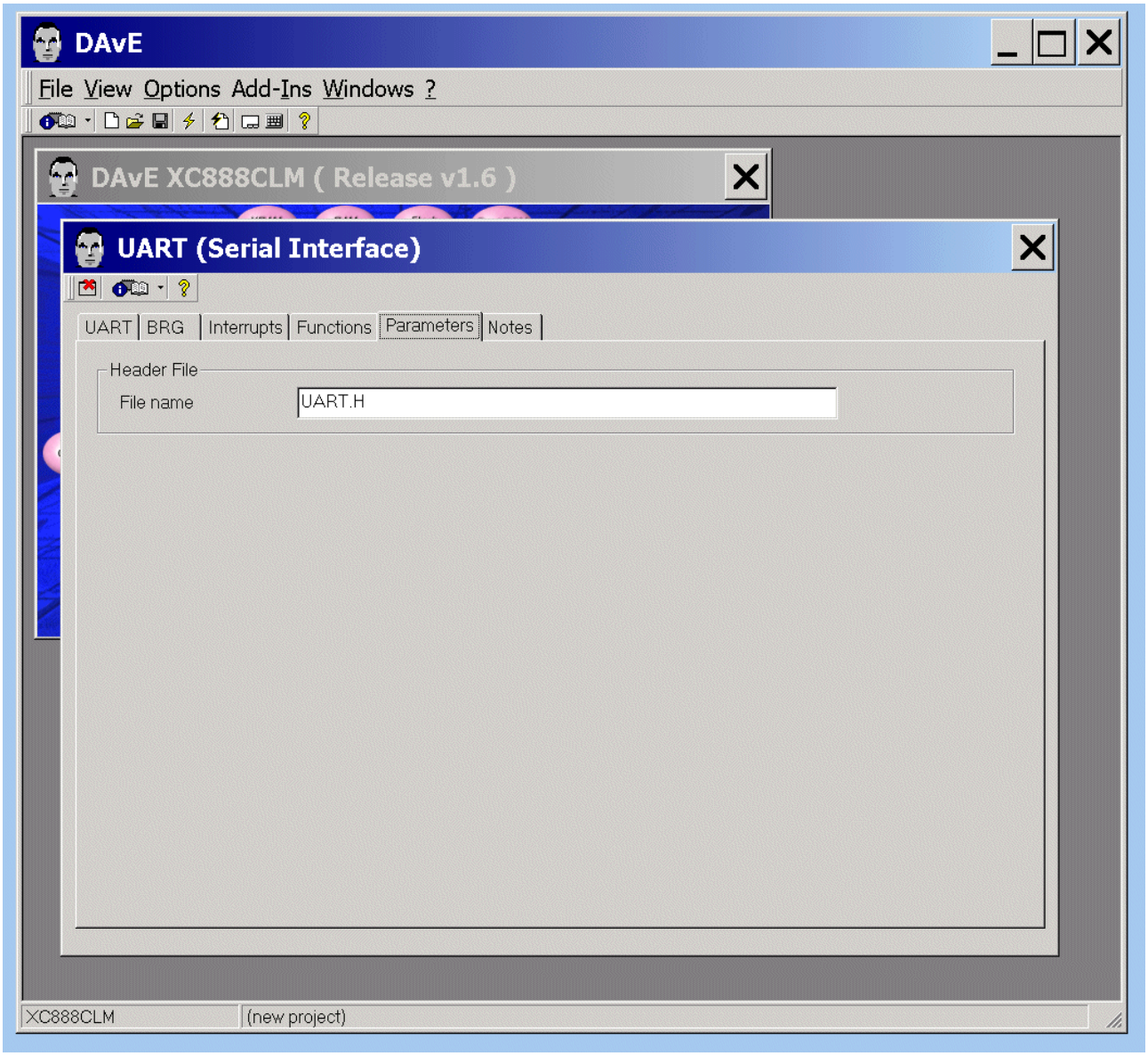
Functions: Initialization Function: tick ✓ UART_vInit
Functions: Miscellaneous Functions: tick ✓ UART_bRxReady
Functions: Miscellaneous Functions: tick ✓ UART_bTxReady
Functions: 8-Bit Data Units: tick ✓ UART_ubGetData8
Functions: 8-Bit Data Units: tick ✓ UART_vSendData8



Note:
You can change function names (e.g. UART_vInit) and file names (e.g. UART.C) anytime.



Parameters: (do nothing)

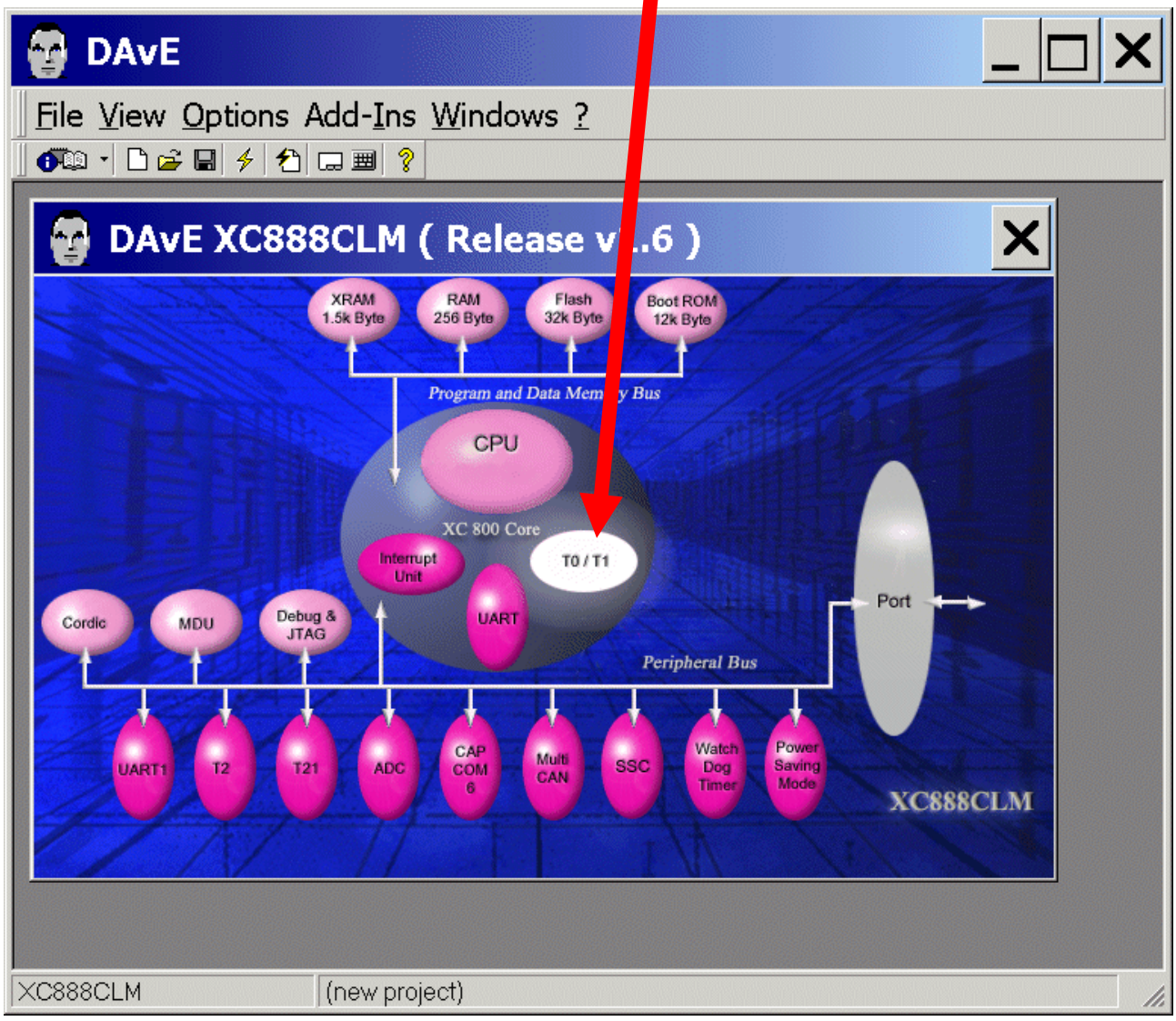


Notes: If you wish, you can insert your comments here.

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

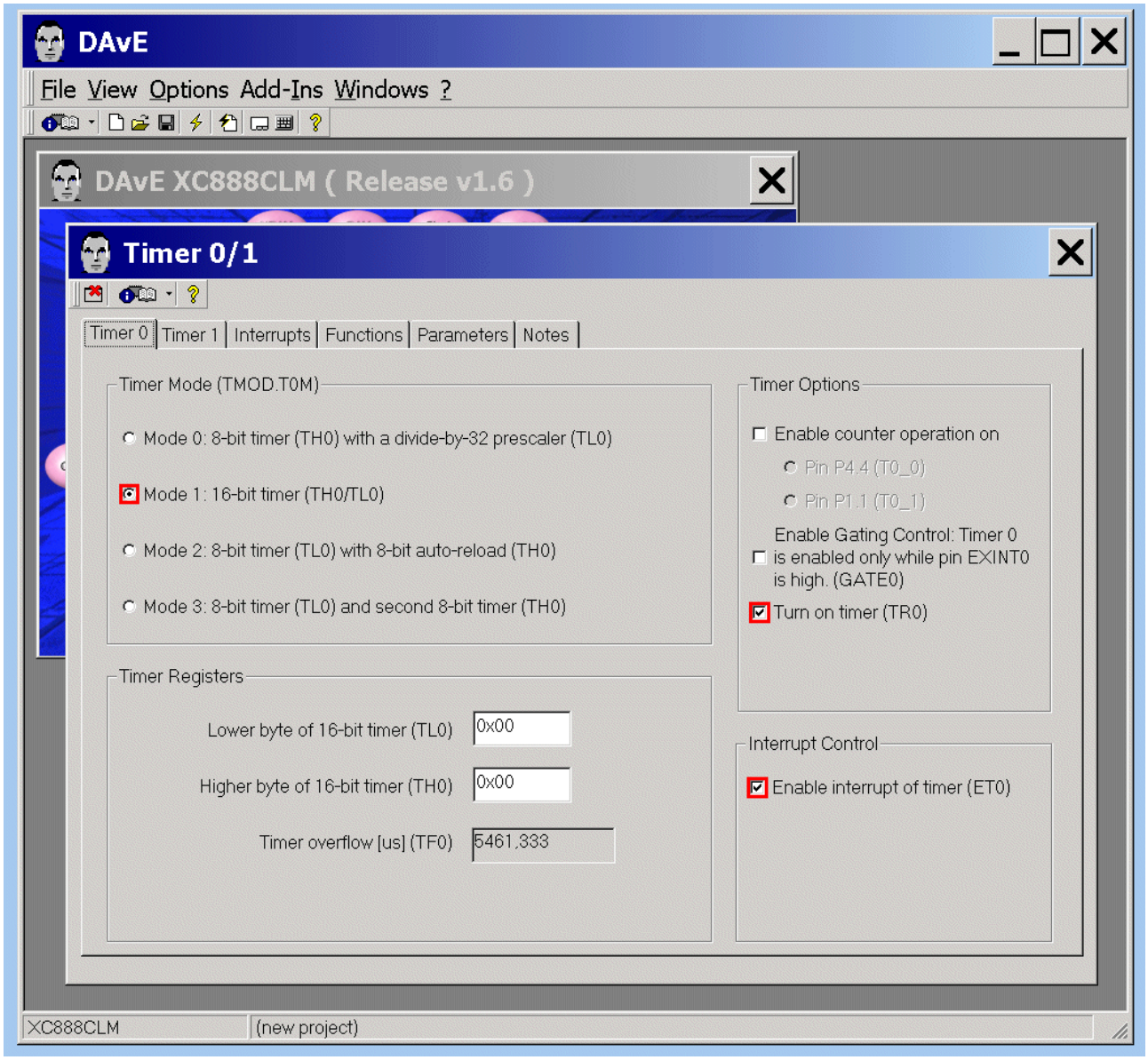
Configure Timer T0:

The configuration window/dialog can be opened by clicking the specific block/module (T0/T1).



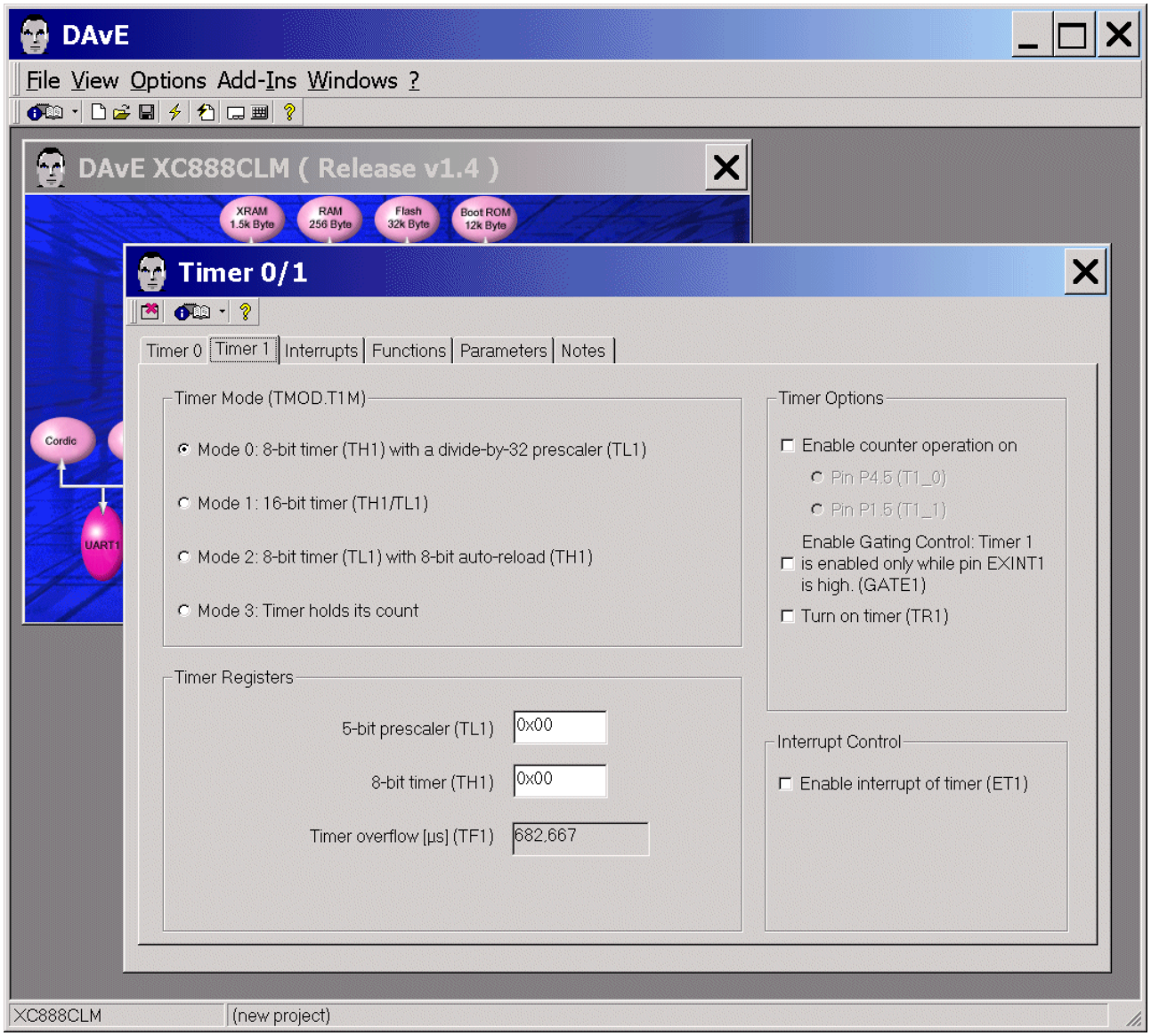
Note:
The LEDs on IO_Port_3 will be blinking (if selected in the main menu)
with a frequency of about 1 second (done in the Timer_0-Interrupt-Service-Routine).
Therefore we have to configure Timer_0.

Timer0: Timer Mode: click ⦿ Mode 1: 16-bit timer
Timer0: Timer Options: tick ✓ Turn on timer (TR0)
Timer0: Interrupt Control: tick ✓ Enable interrupt of timer (ET0)

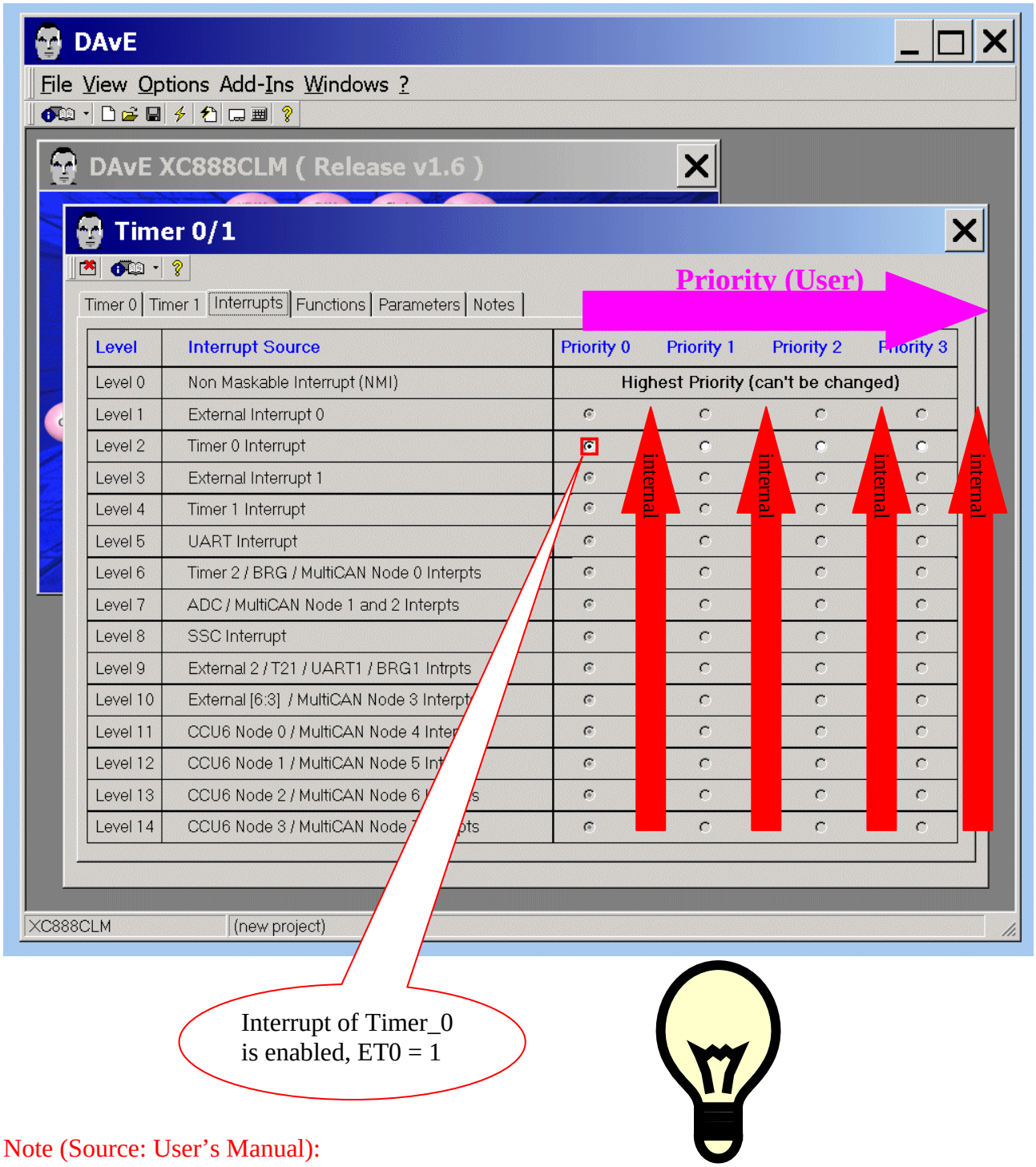


Note:
We need 183 Timer_0 overflows to achieve an approximate 1 second delay.
This will be handled in the Timer_0 interrupt function.
183 * 5461,333 µs = 0,9994 s.

Timer1: do nothing (not used)

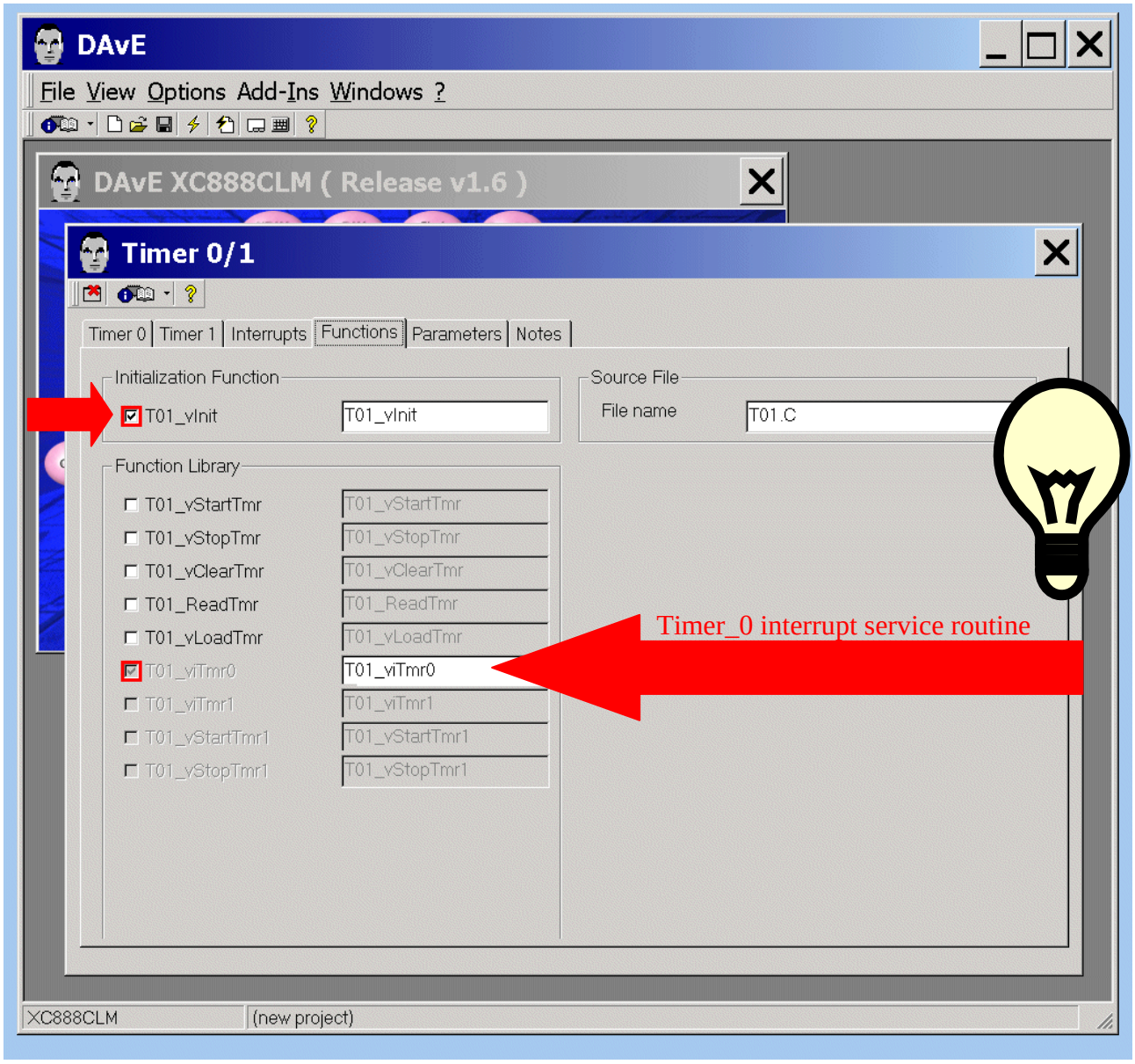


Interrupts: (do nothing)



Note (Source: User's Manual):
An interrupt that is currently being serviced can only be interrupted by a higher-priority interrupt, but not by another interrupt of the same or lower priority.
Hence, an interrupt of the highest priority cannot be interrupted by any other interrupt request.
If two or more requests of different priority levels are received simultaneously, the request with the highest priority is serviced first. If requests of the same priority are received simultaneously, an internal polling sequence determines which request is serviced first. Thus, within each priority level, there is a second priority structure determined by a polling sequence as shown in the User's Manual and above.

Functions: Initialization Function: tick ✓ T01_vInit



Note:
Timer_0 has a dedicated interrupt vector address ($000B_H$), interrupt node and its own interrupt status flag TF0.
The vector is used to service the corresponding interrupt node request – when enabled (ET0=1), which means: the interrupt system will hardware-generate an LCALL to the appropriate service routine at $000B_H$.
TF0 will be automatically cleared by hardware (the core) once its pending interrupt request is serviced.

Additional information: Interrupt Handling (Source: User’s Manual):

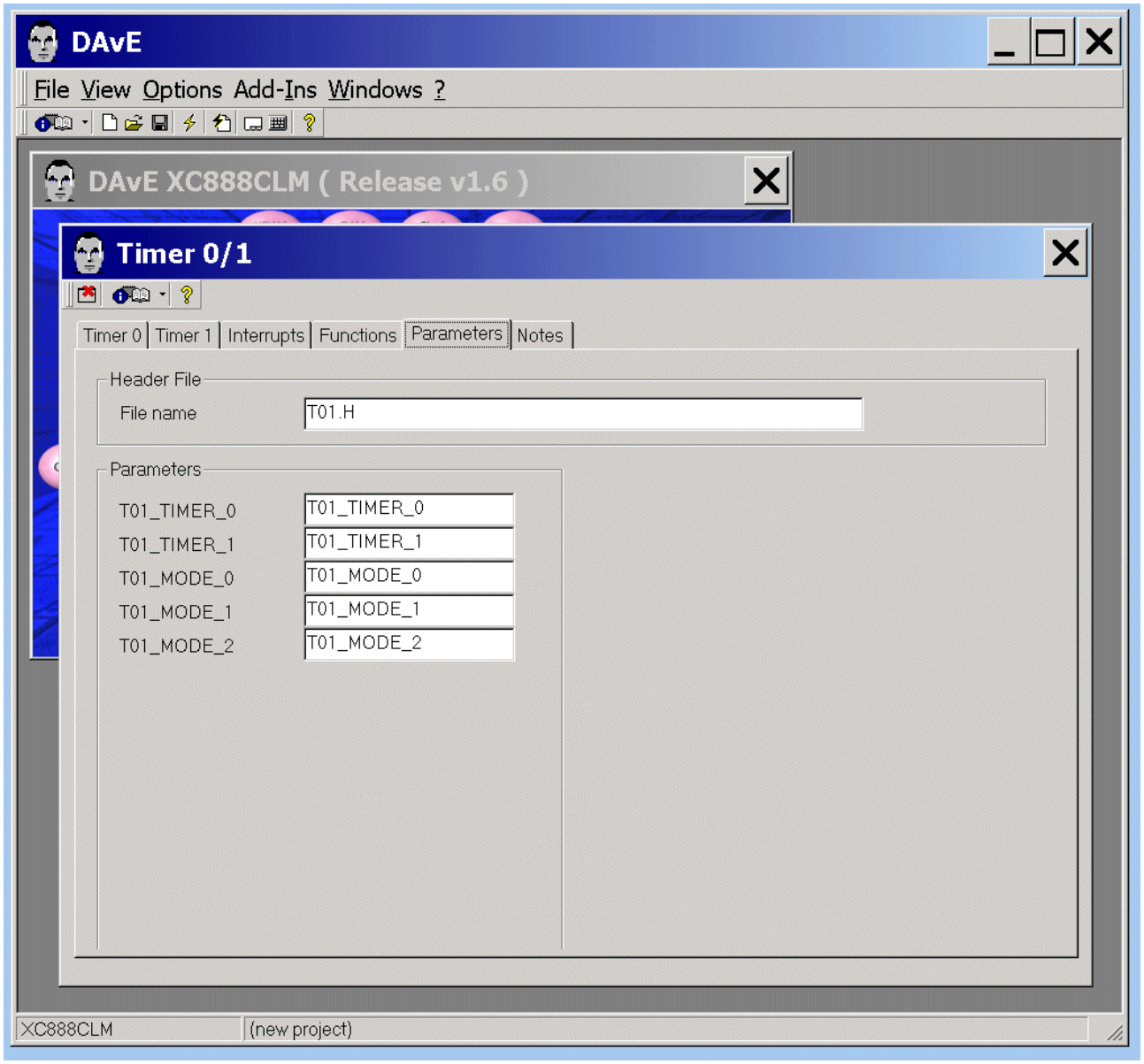
The processor acknowledges an interrupt request by executing a hardware generated LCALL to the appropriate service routine (interrupt vector address).

In some cases, hardware also clears the flag that generated the interrupt, while in other cases, the flag must be cleared by the user’s software (e.g. see DAvE Source Code).

The hardware-generated LCALL pushes the contents of the Program Counter (PC) onto the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to (interrupt vector addresses see User’s Manual).

Program execution returns to the next instruction after calling the interrupt when the RETI instruction is encountered. The RETI instruction informs the processor that the interrupt routine is no longer in progress, then pops the two top bytes from the stack and reloads the PC.
Execution of the interrupted program continues from the point where it was stopped.
Note that the RETI instruction is important because it informs the processor that the program has left the current interrupt priority level.
A simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system on the assumption that an interrupt was still in progress. In this case, no interrupt of the same or lower priority level would be acknowledged.

Parameters: (do nothing)

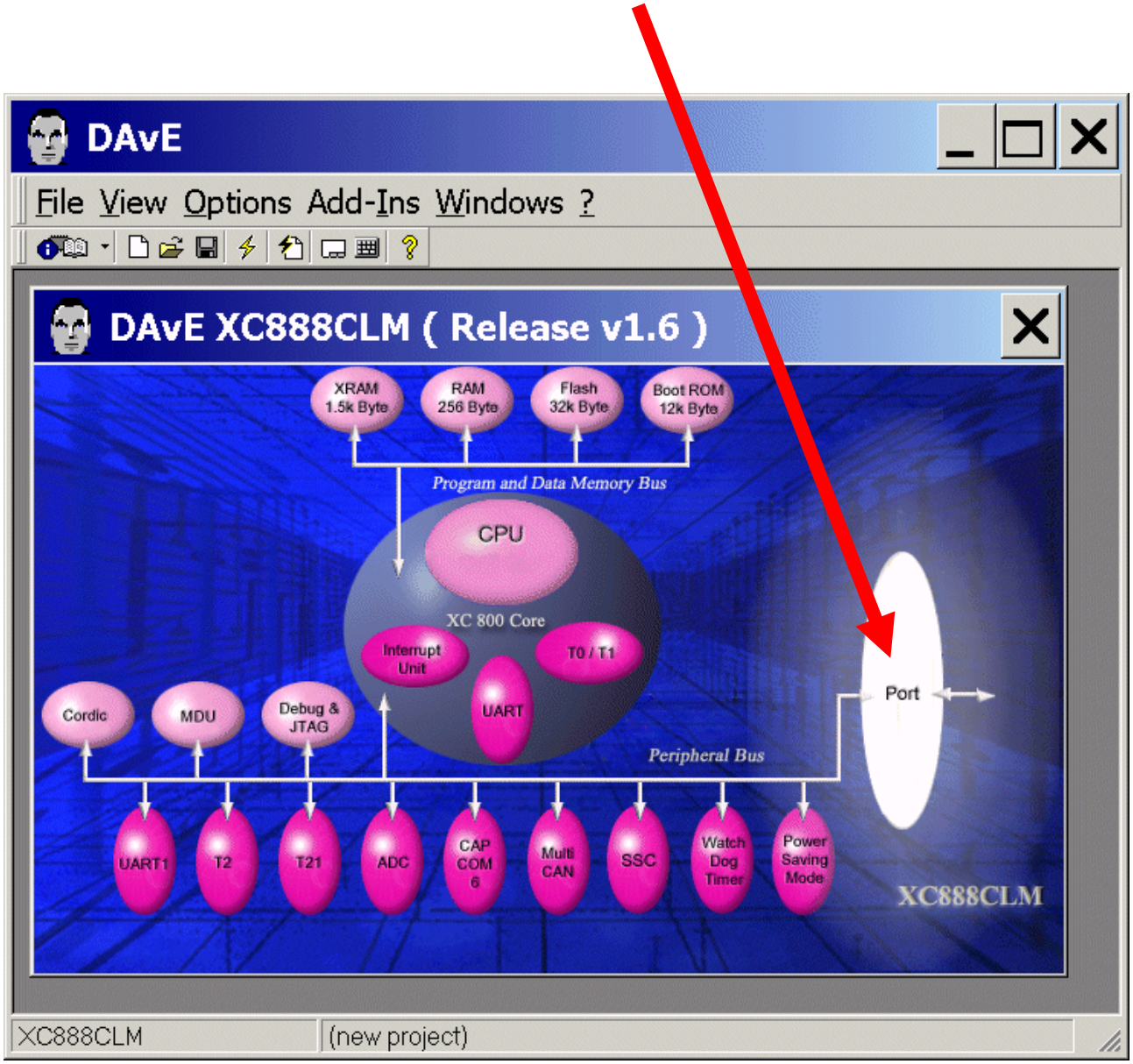


Notes: If you wish, you can insert your comments here.

Exit this dialog now by clicking the close button.

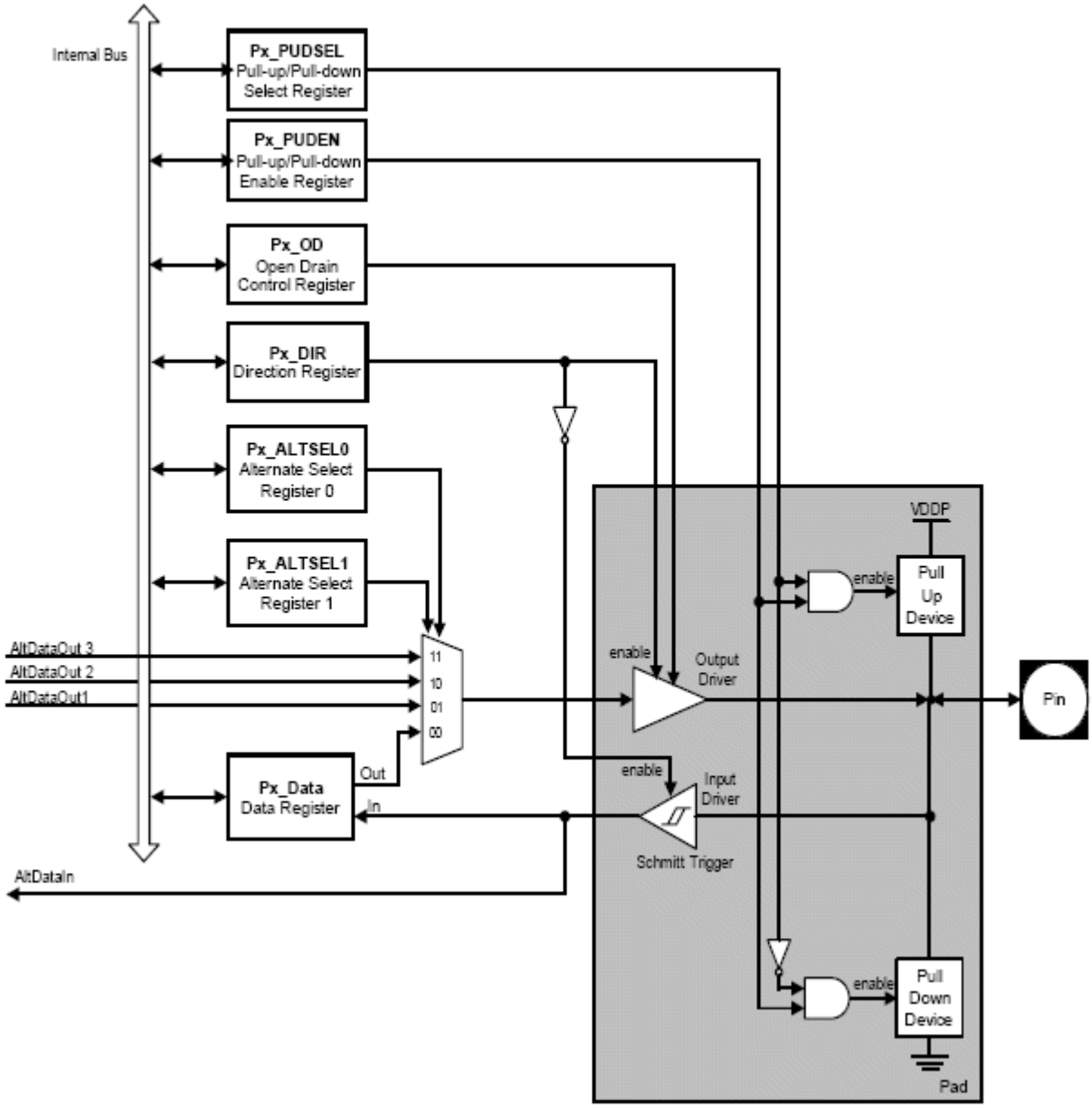
Configure Port 3 to Output:

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

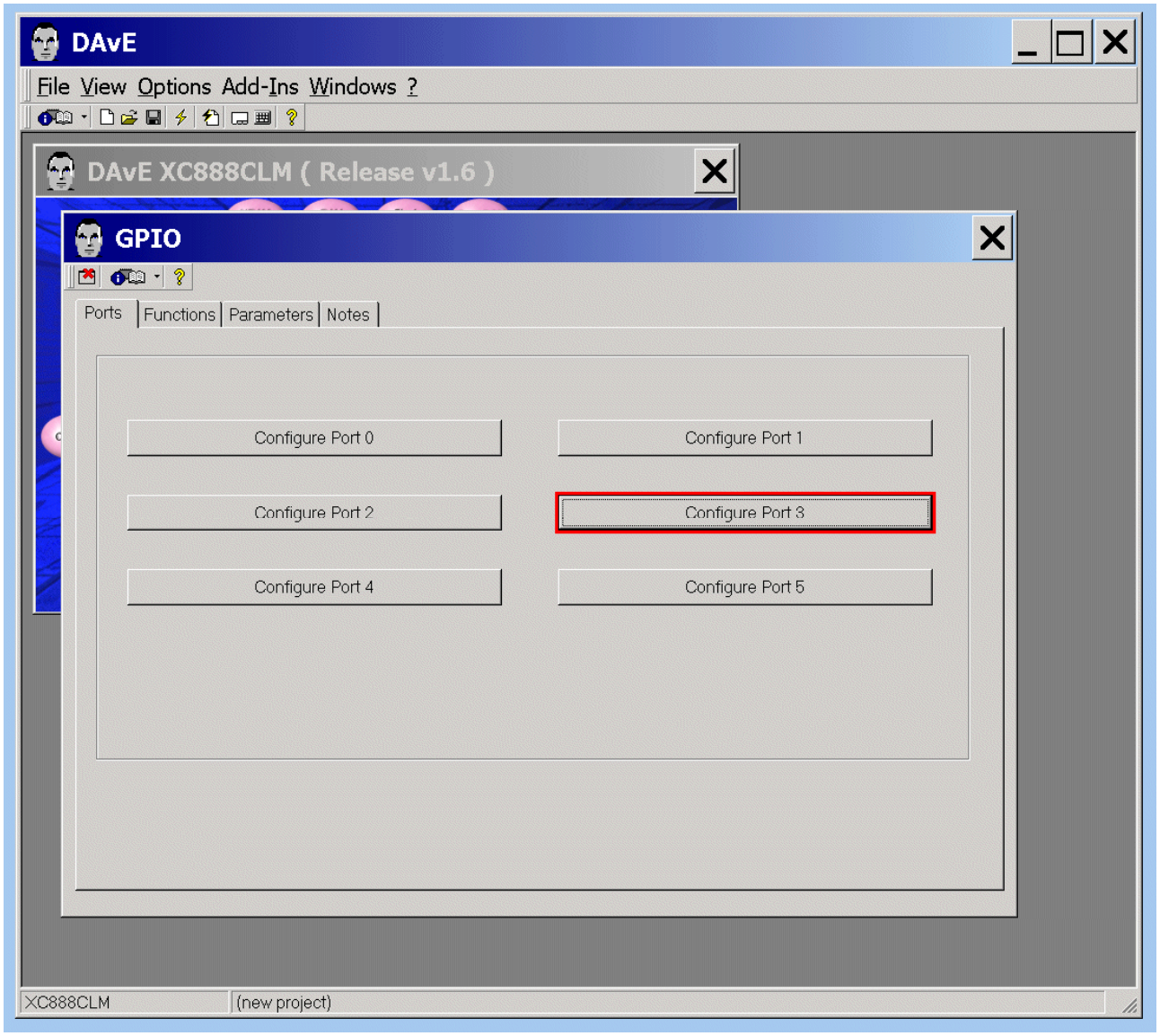


Note:
The LEDs are connected to IO_Port_3.

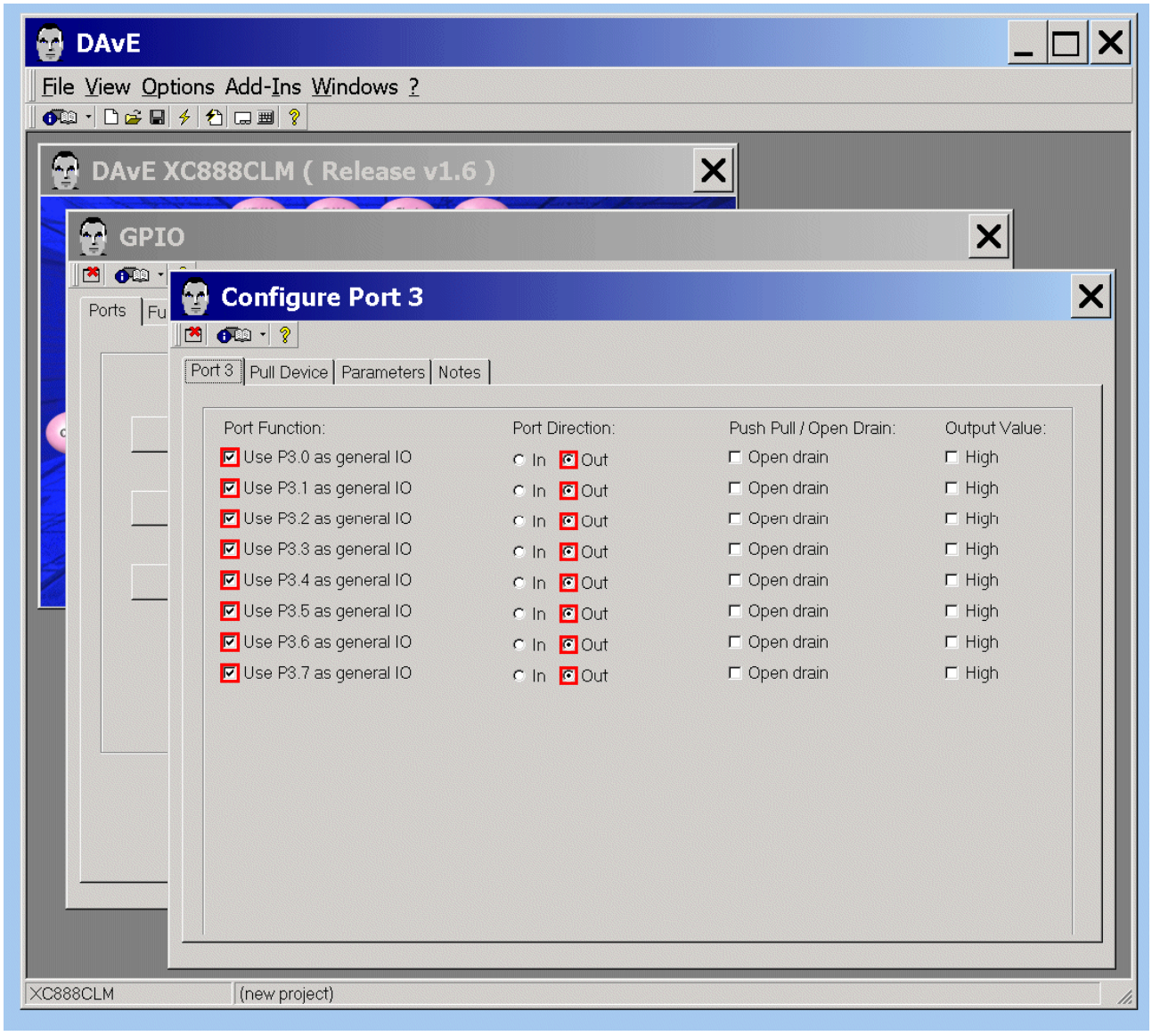
Additional information: Parallel Ports – General Structure (Source: User's Manual):



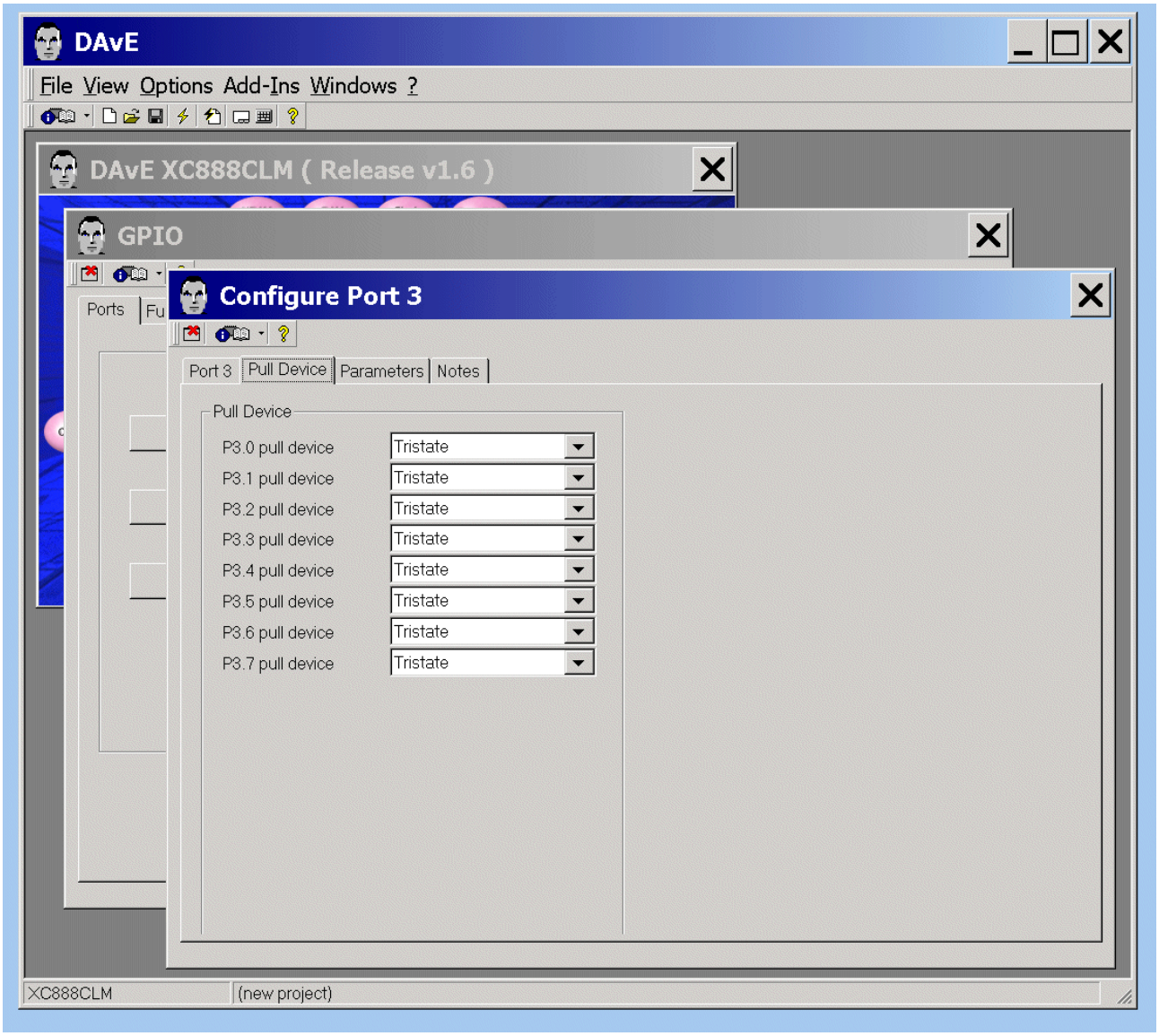
Ports: click “Configure Port 3”



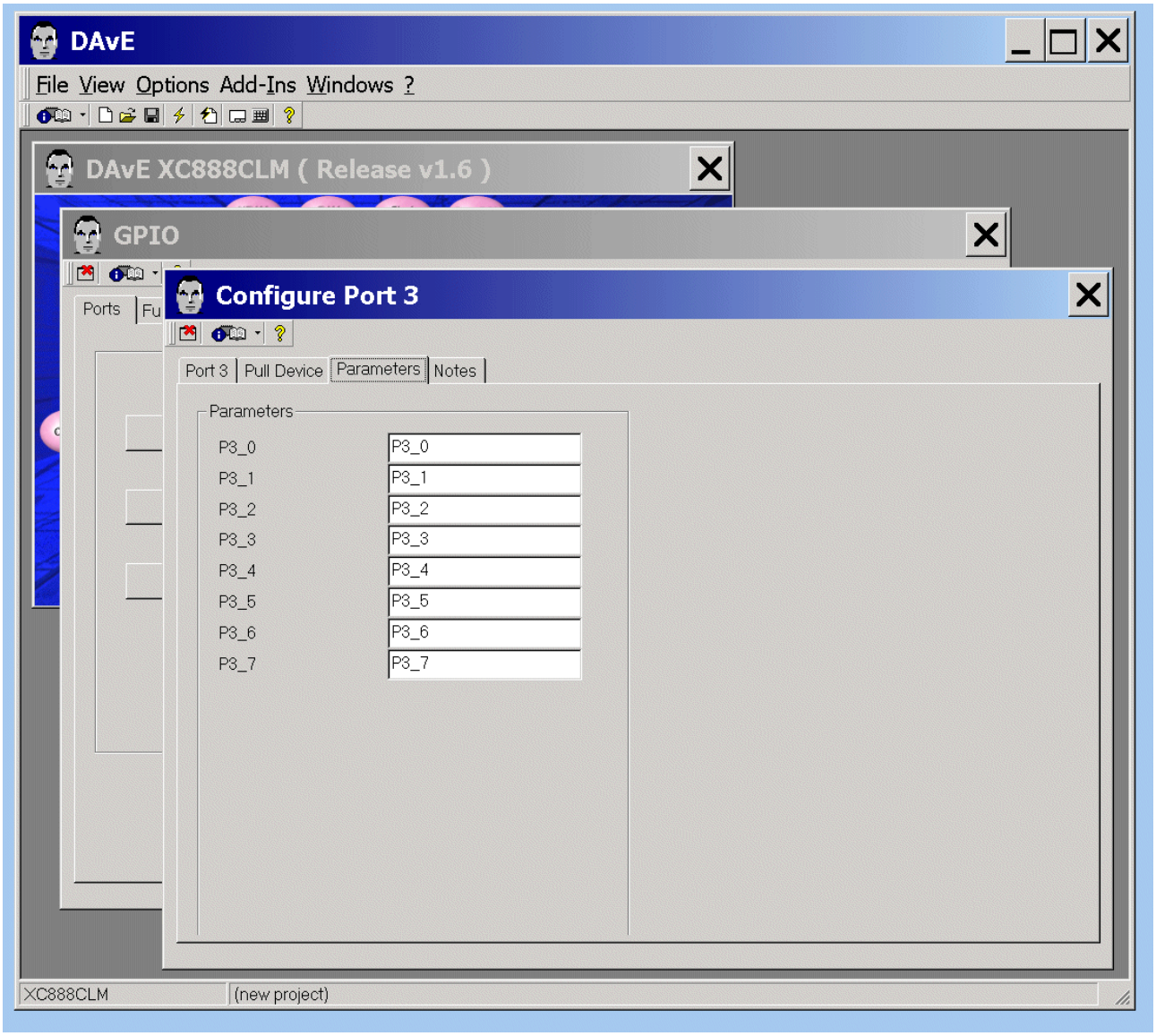
Port 3: Port Function: tick ✓ Use P3.0 as general IO - Port Direction: click ⦿ Out
Port 3: Port Function: tick ✓ Use P3.1 as general IO - Port Direction: click ⦿ Out
Port 3: Port Function: tick ✓ Use P3.2 as general IO - Port Direction: click ⦿ Out
Port 3: Port Function: tick ✓ Use P3.3 as general IO - Port Direction: click ⦿ Out
Port 3: Port Function: tick ✓ Use P3.4 as general IO - Port Direction: click ⦿ Out
Port 3: Port Function: tick ✓ Use P3.5 as general IO - Port Direction: click ⦿ Out
Port 3: Port Function: tick ✓ Use P3.6 as general IO - Port Direction: click ⦿ Out
Port 3: Port Function: tick ✓ Use P3.7 as general IO - Port Direction: click ⦿ Out



Pull Device: (do nothing)



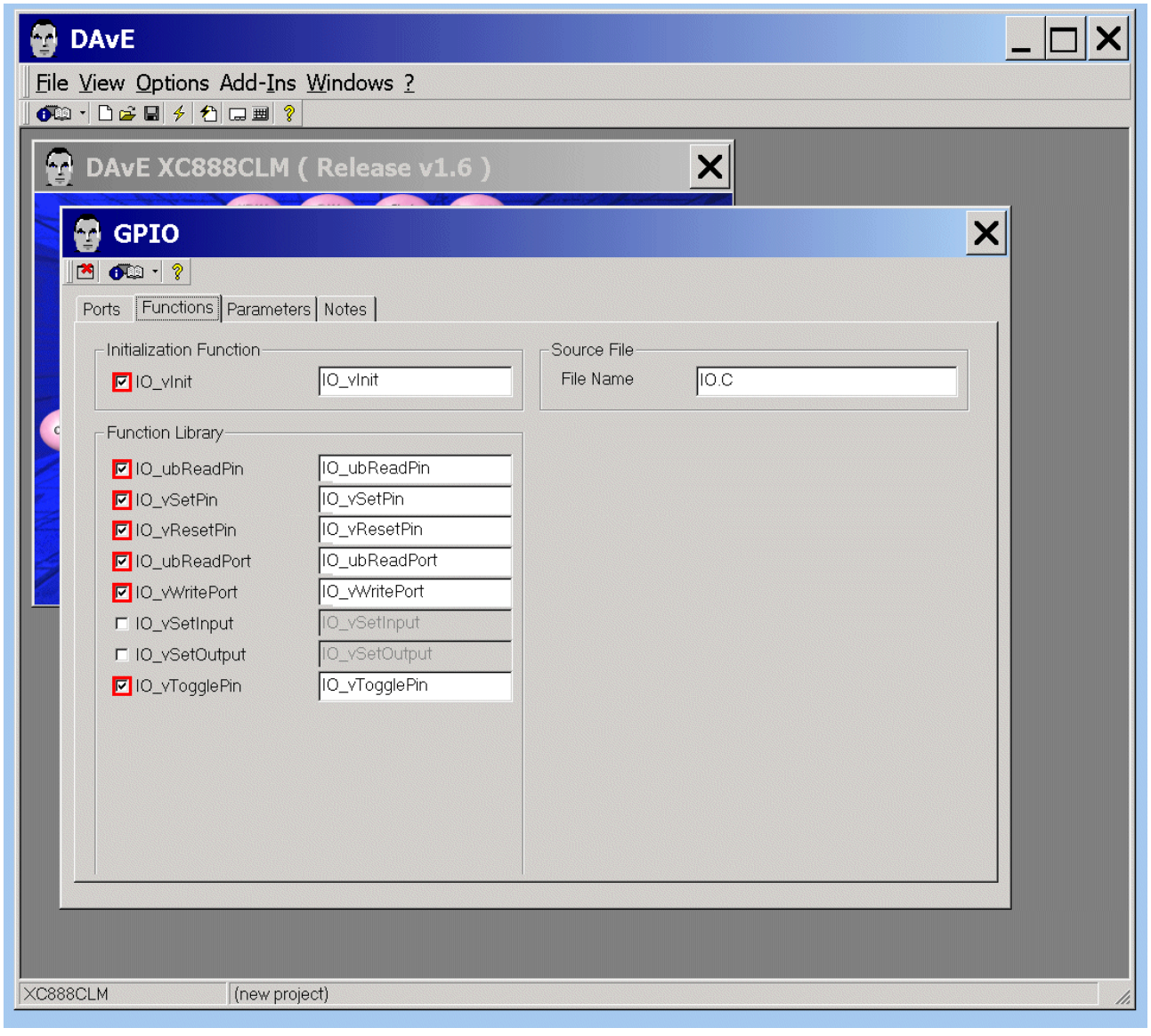
Parameters: (do nothing)



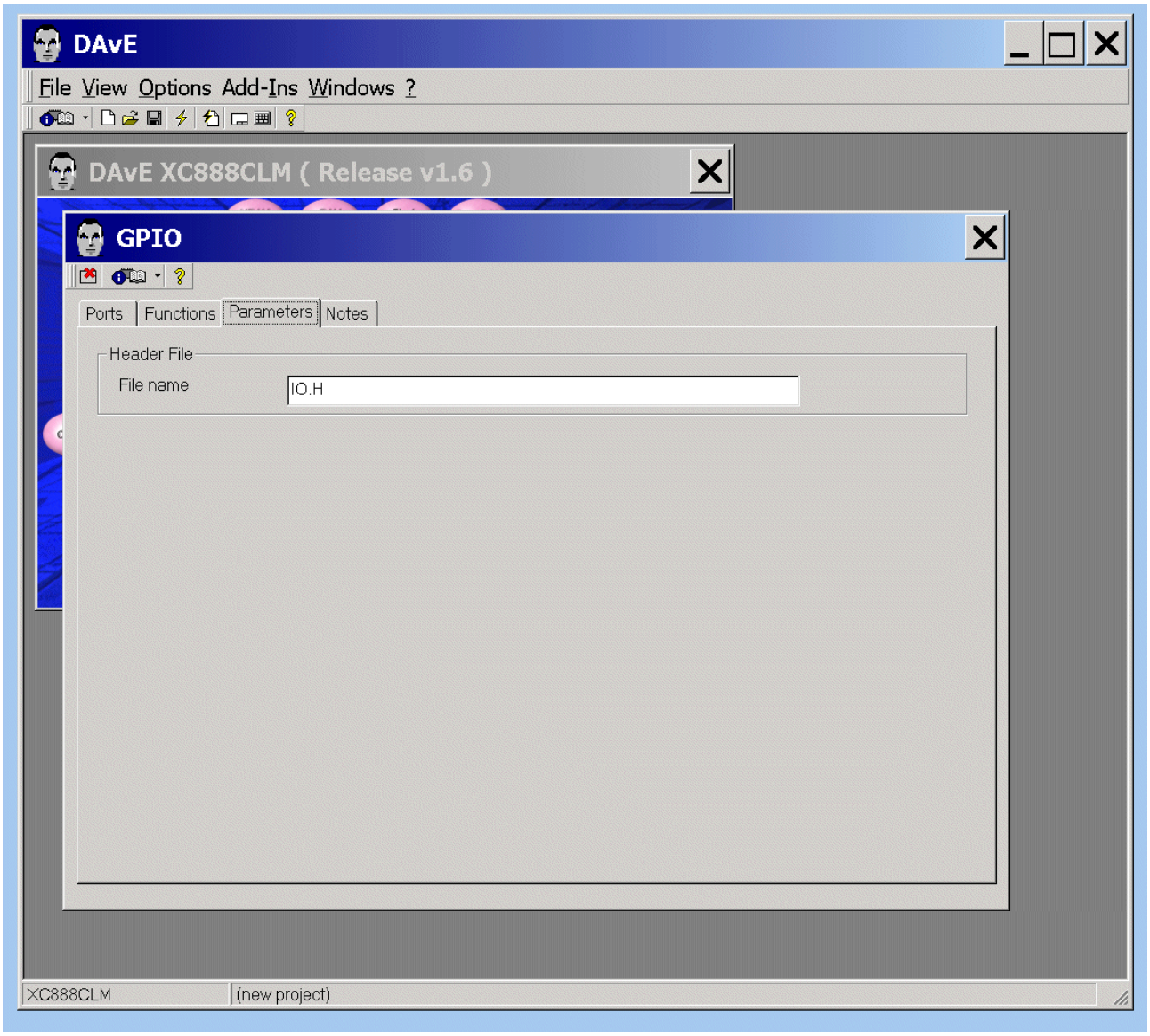
Notes: If you wish, you can insert your comments here.

Exit this dialog now by clicking the close button.

Functions: Initialization Functions: tick ✓ IO_vInit
Functions: Function Library: tick ✓ IO_ubReadPin
Functions: Function Library: tick ✓ IO_vSetPin
Functions: Function Library: tick ✓ IO_vResetPin
Functions: Function Library: tick ✓ IO_ubReadPort
Functions: Function Library: tick ✓ IO_vWritePort
Functions: Function Library: tick ✓ IO_vTogglePin



Parameters: (do nothing)



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

Exit this dialog now by clicking the close button.

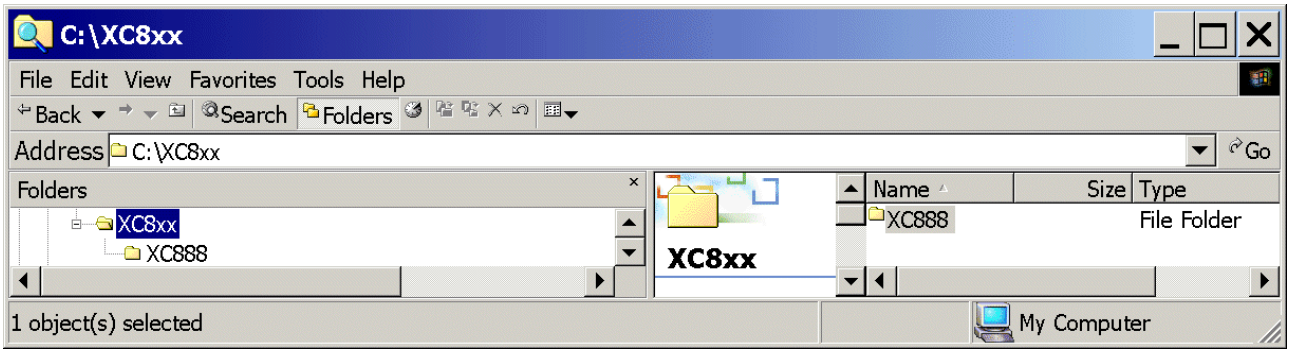
Note:
Before we save the DAvE Project we are going to create a suitable directory structure with Windows File Explorer (see next page!).

Create a suitable directory structure for DAvE-Bench (Eclipse IDE, SDCC compiler):

Start Windows File Explorer

Create directory/folder C:\XC8xx

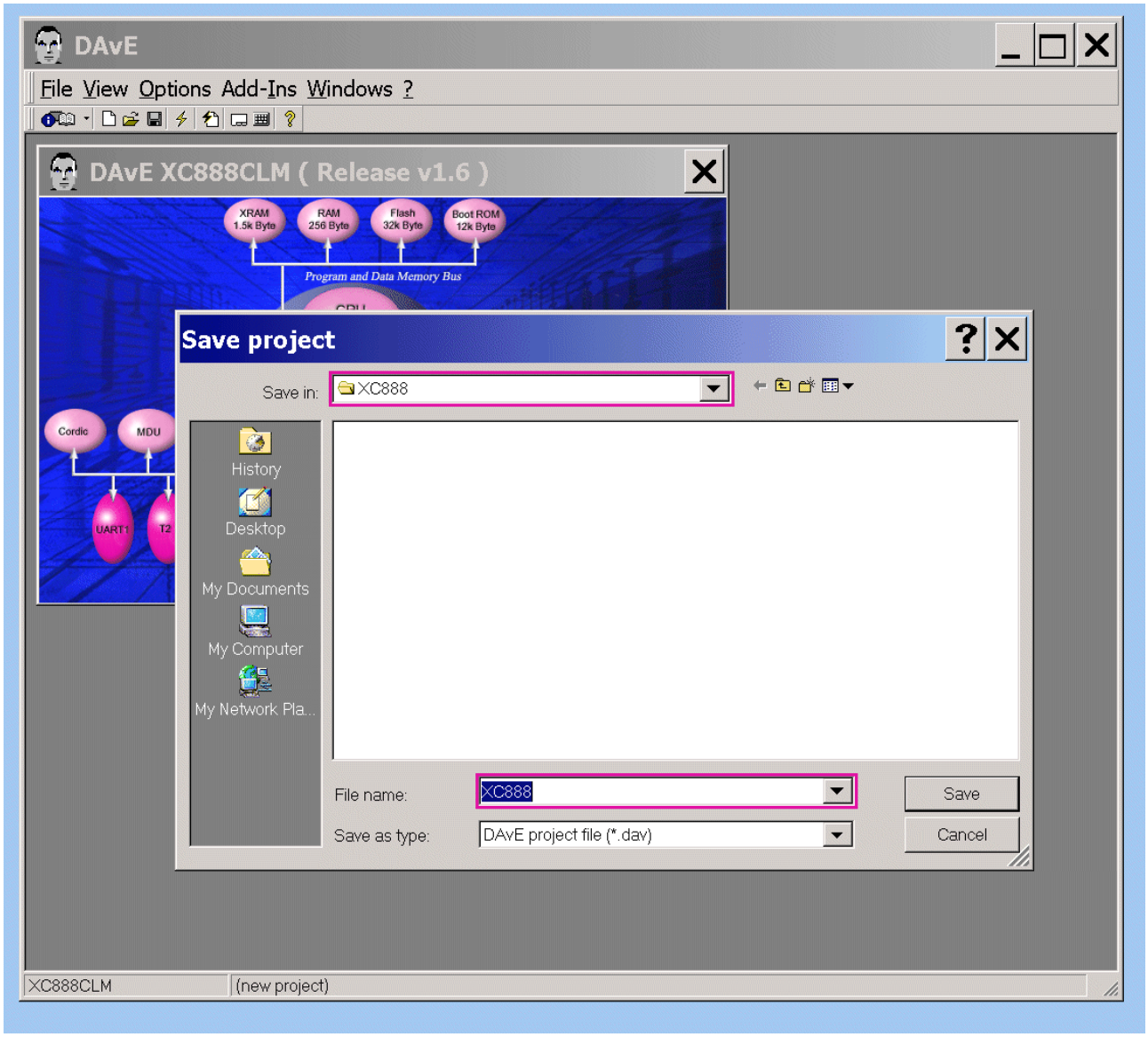
Create directory/folder C:\XC8xx\XC888



Save the project:

File
Save

Save project: Save in select C:\XC8xx\XC888
File name: insert XC888

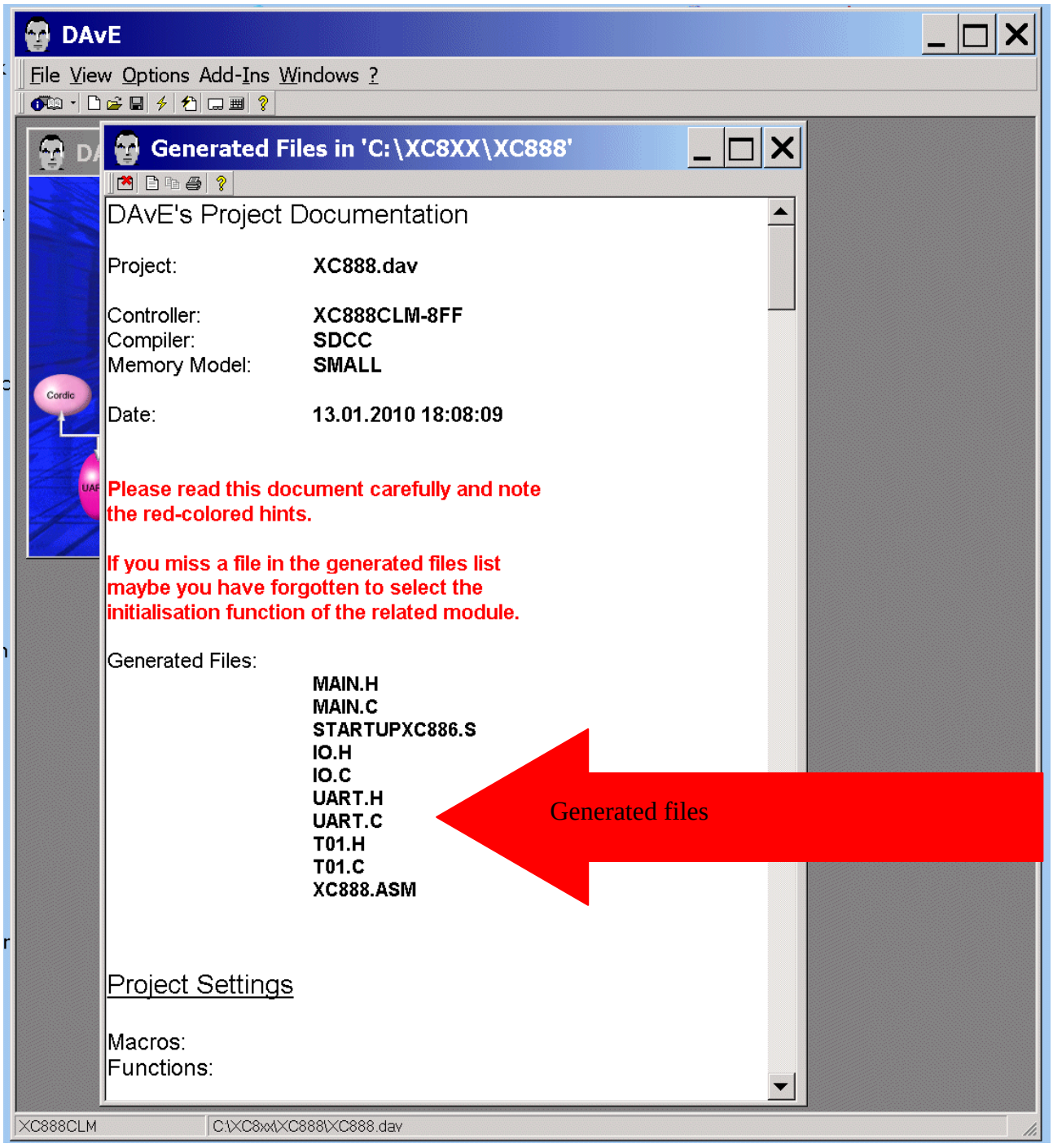


Save

Generate Code:

File Generate Code	or click 

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



File - Exit

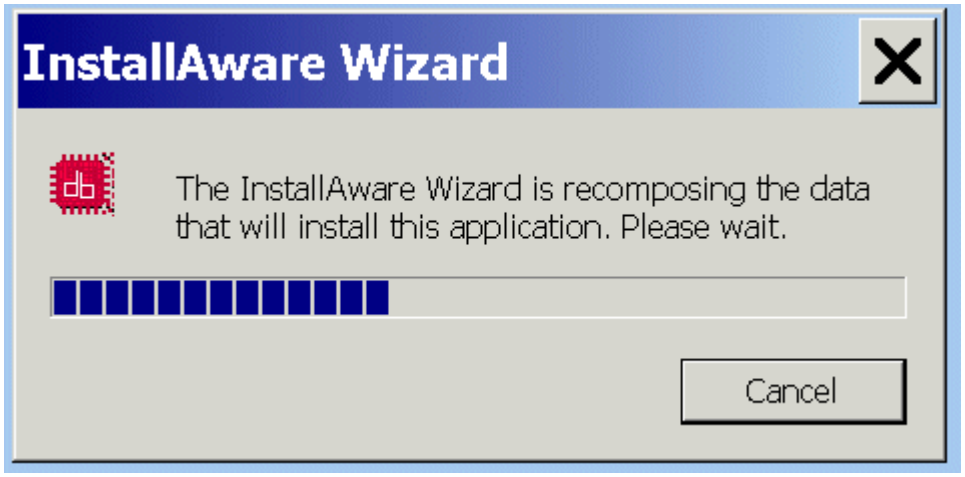
Save changes?

Click: Yes

4.) Using DAvE Bench:

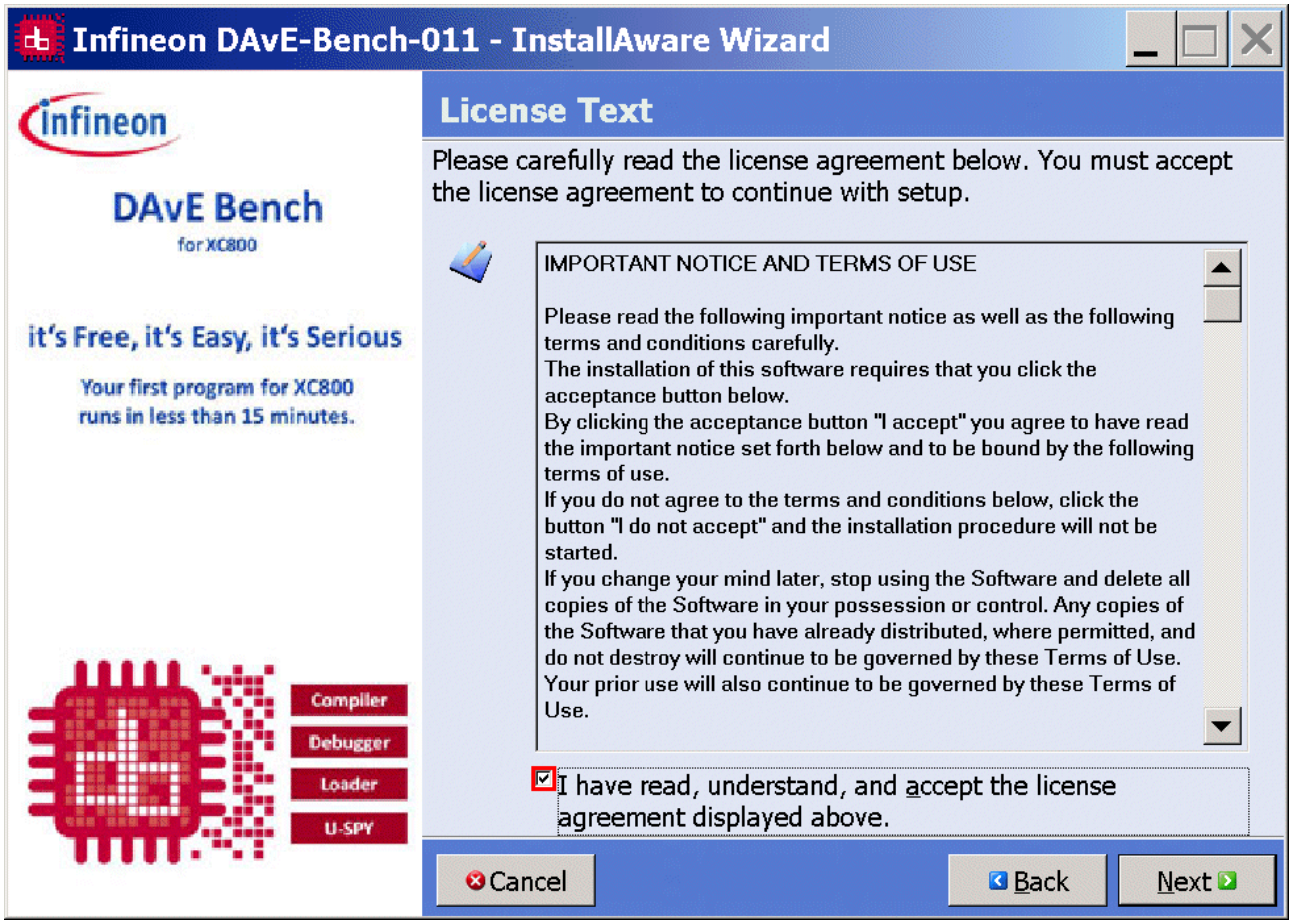
Download and install DAvE Bench (http://www.infineon.com/DAvE-Bench):

Double-click setup.exe:



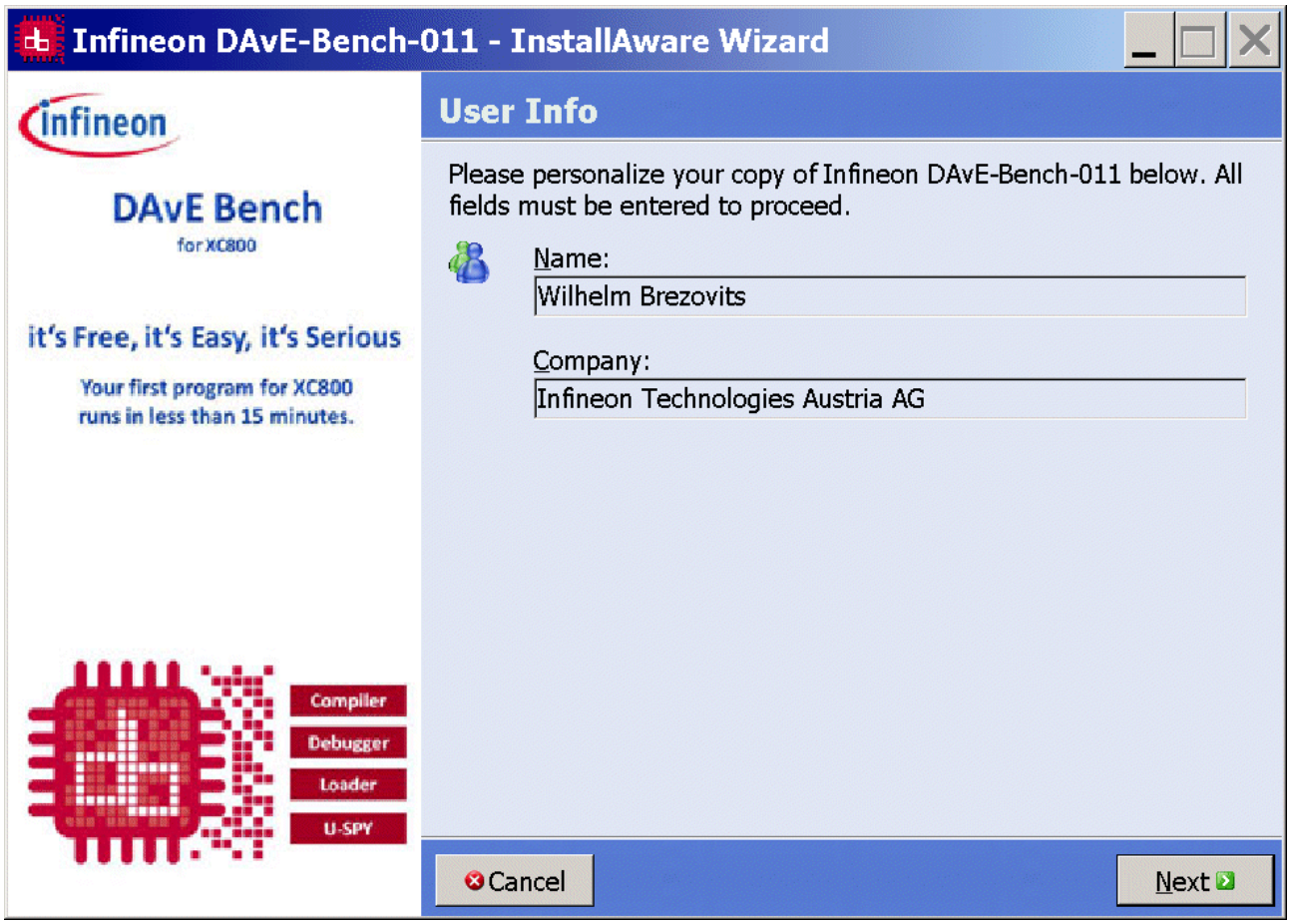

Infineon DAvE-Bench-011 - InstallAware Wizard
Welcome
Welcome to the InstallAware Wizard for Infineon DAvE-Bench-011 Setup. This wizard will install release 00.00.011 on your computer.
You may navigate through the pages in this wizard by clicking the Back and Next buttons below. Click Cancel at anytime to exit the
DAvE Bench
for XC800
it's Free, it's Easy, it's Serious
Your first program for XC800 runs in less than 15 minutes.
Compiler
Debugger
Loader
U-SPY
Cancel
Next

Click 
Next



Tick ☑ I have read, understand…

Click 

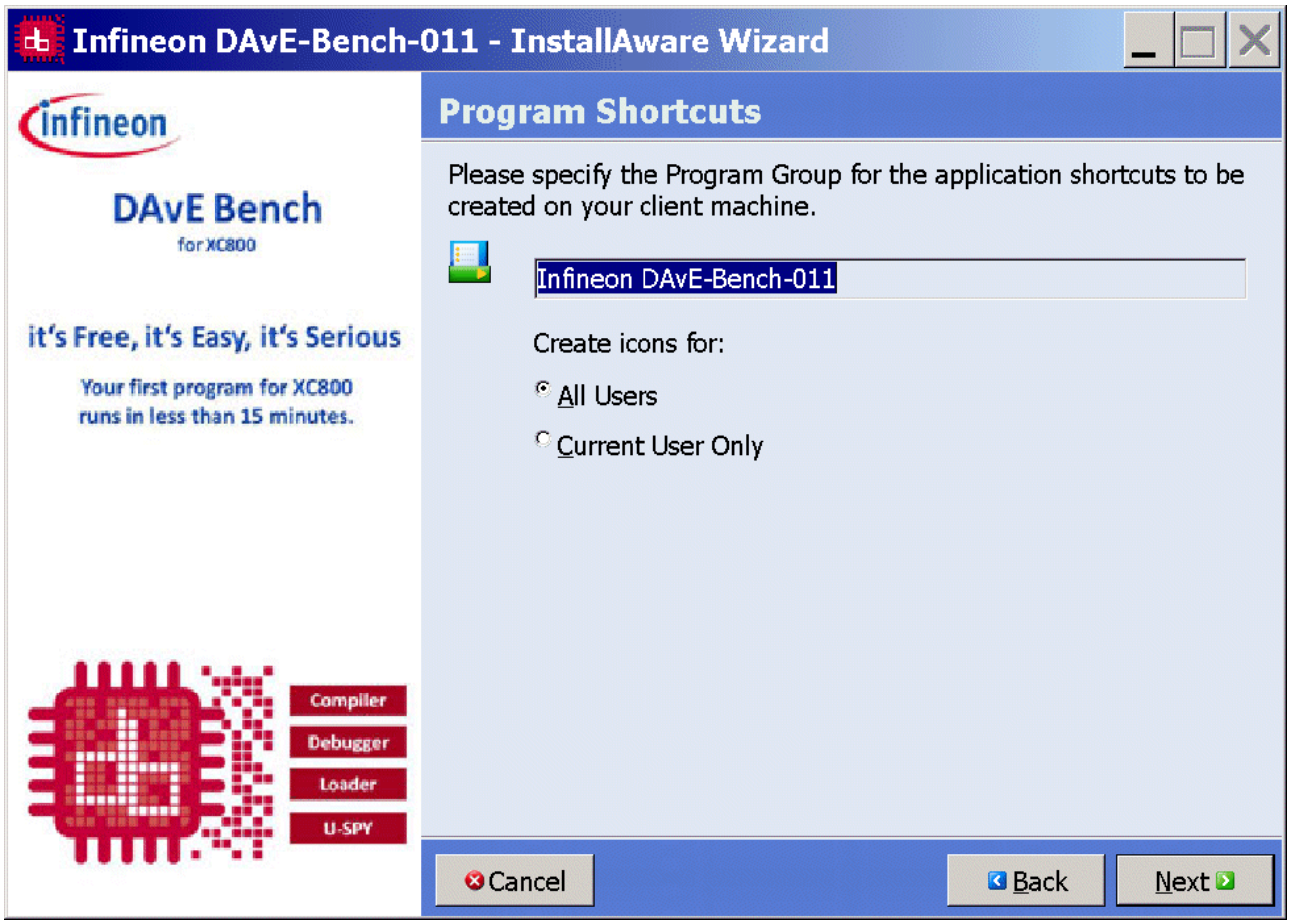


Insert Name: Your name
Insert Company: Your company

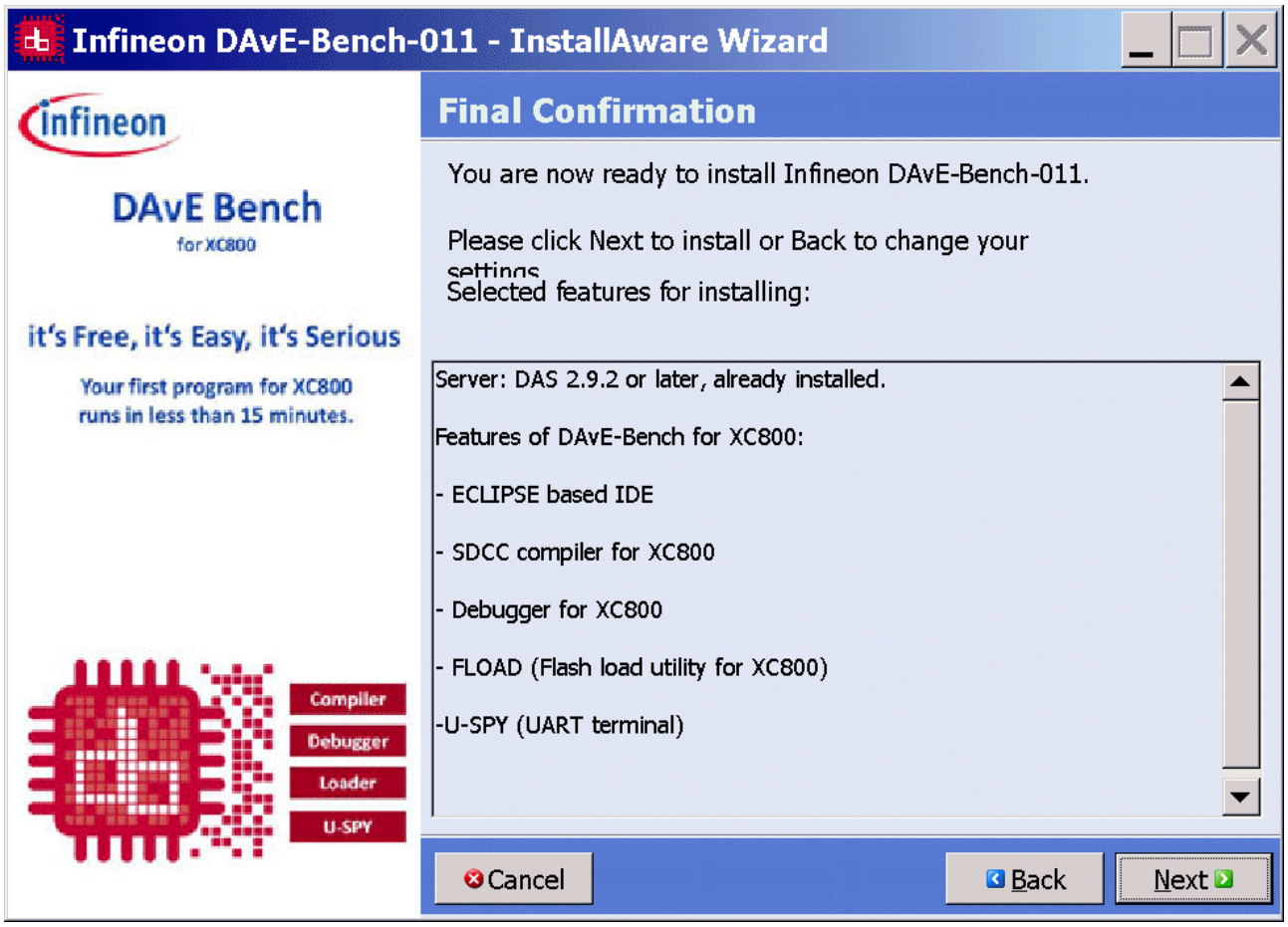
Click 

Select a suitable directory and

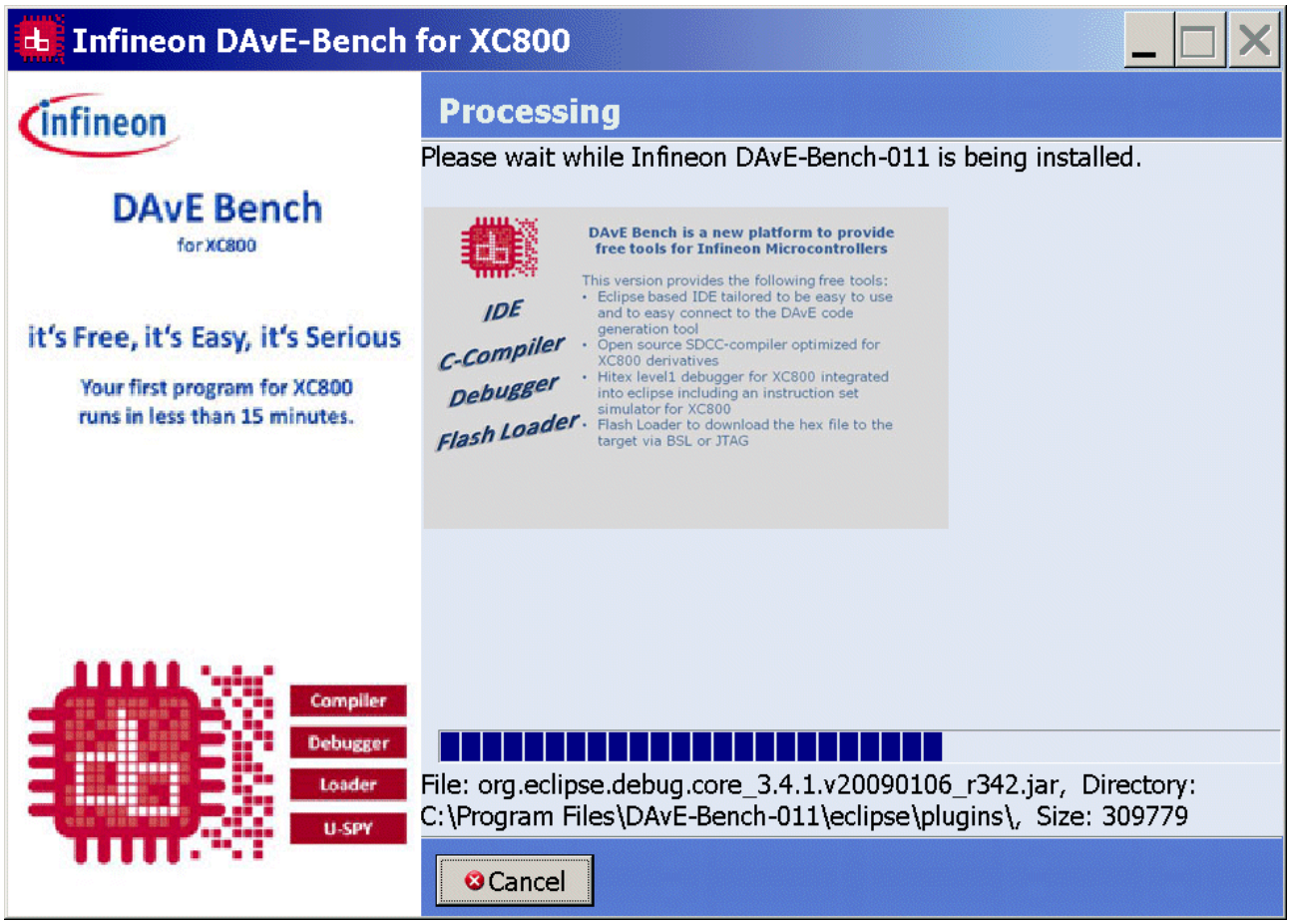
Click 

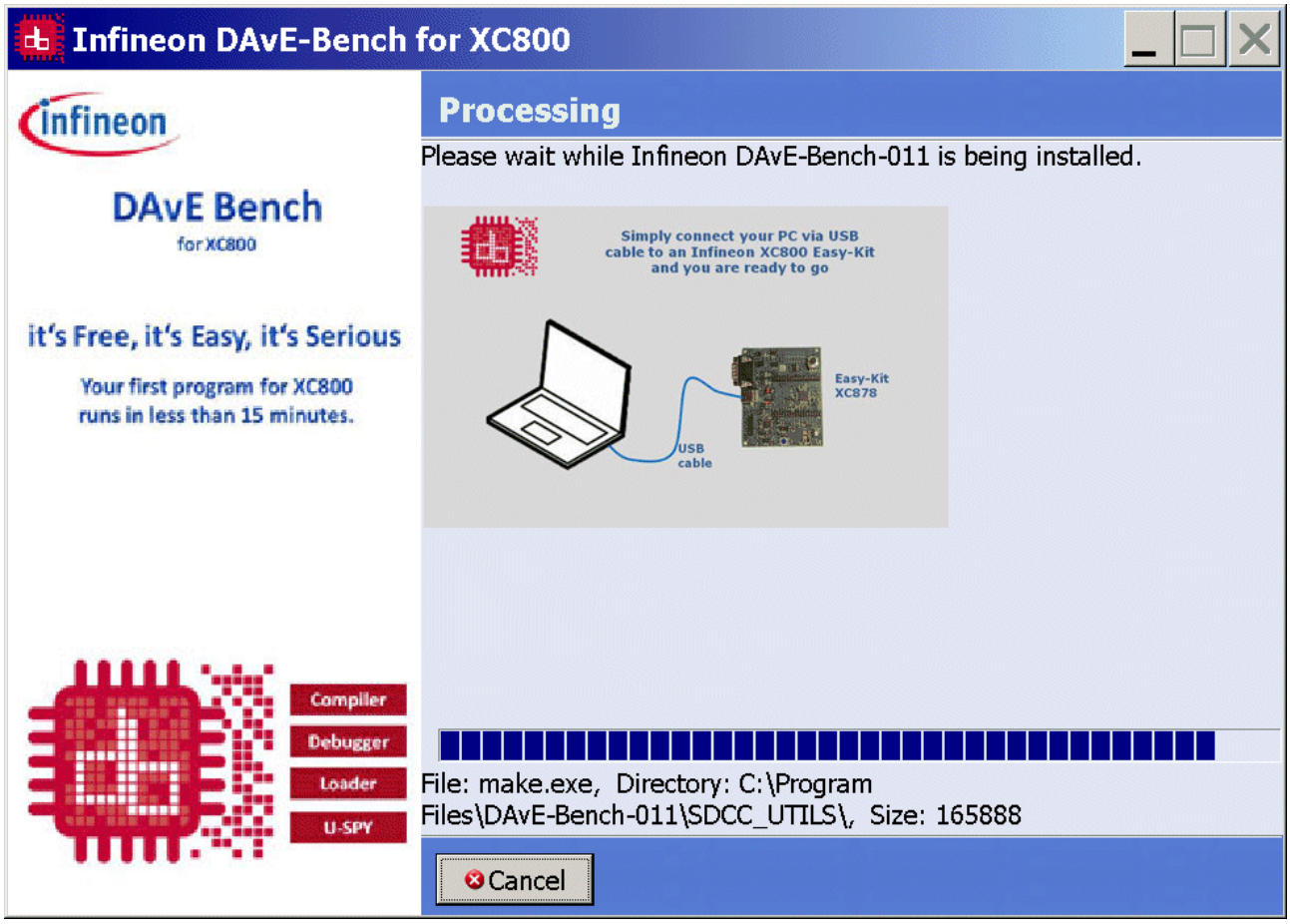

Infineon DAvE-Bench-011 - InstallAware Wizard
DAvE Bench
for XC800
it's Free, it's Easy, it's Serious
Your first program for XC800
runs in less than 15 minutes.
Compiler
Debugger
Loader
U-SPY
Program Shortcuts
Please specify the Program Group for the application shortcuts to be created on your client machine.
Infineon DAvE-Bench-011
Create icons for:
All Users
Current User Only
Cancel
Back
Next

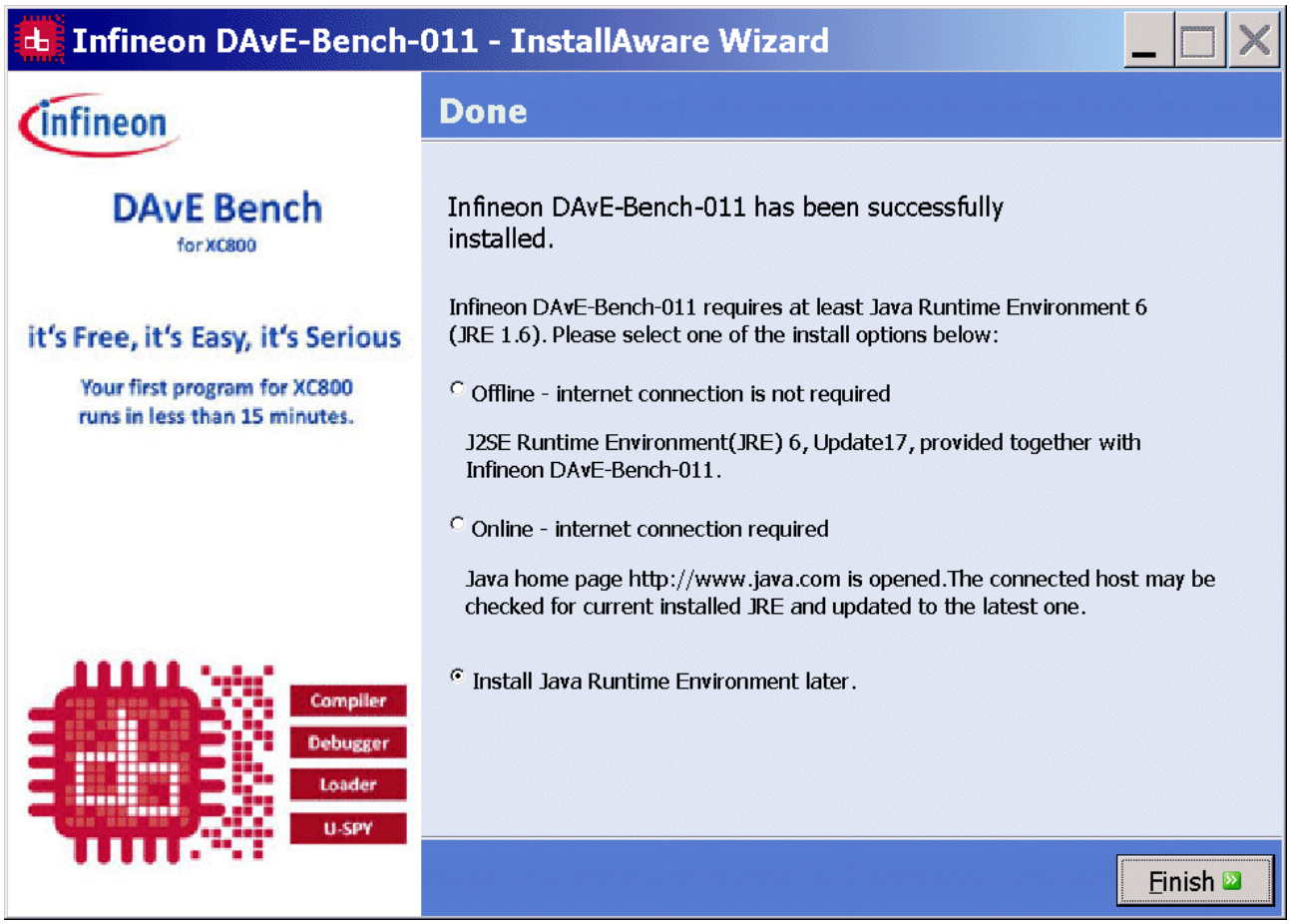
Click 
Next


Infineon DAvE-Bench-011 - InstallAware Wizard
Infineon
DAvE Bench
for XC800
it's Free, it's Easy, it's Serious
Your first program for XC800
runs in less than 15 minutes.
Compiler
Debugger
Loader
U-SPY
Final Confirmation
You are now ready to install Infineon DAvE-Bench-011.
Please click Next to install or Back to change your settings.
Selected features for installing:
Server: DAS 2.9.2 or later, already installed.
Features of DAvE-Bench for XC800:
- ECLIPSE based IDE
- SDCC compiler for XC800
- Debugger for XC800
- FLOAD (Flash load utility for XC800)
-U-SPY (UART terminal)
Cancel
Back
Next

Click 
Next

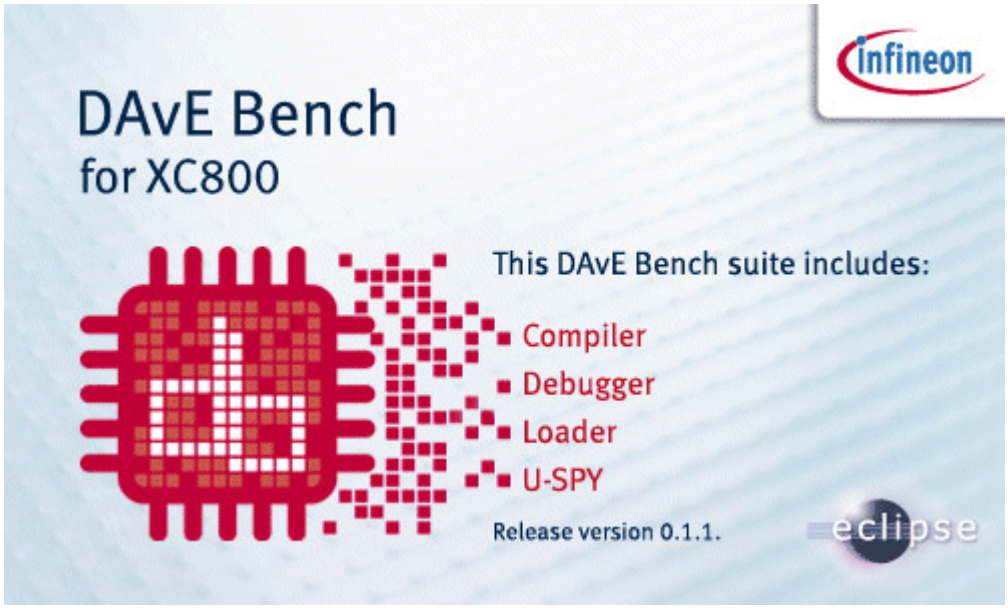

Infineon DAvE-Bench for XC800
Processing
Please wait while Infineon DAvE-Bench-011 is being installed.
DAvE Bench
for XC800
it's Free, it's Easy, it's Serious
Your first program for XC800
runs in less than 15 minutes.
DAvE Bench is a new platform to provide free tools for Infineon Microcontrollers
This version provides the following free tools:
IDE
C-Compiler
Debugger
Flash Loader
Eclipse based IDE tailored to be easy to use and to easy connect to the DAvE code generation tool
Open source SDCC-compiler optimized for XC800 derivatives
Hitex level1 debugger for XC800 integrated into eclipse including an instruction set simulator for XC800
Flash Loader to download the hex file to the target via BSL or JTAG
Compiler
Debugger
Loader
U-SPY
File: org.eclipse.debug.core_3.4.1.v20090106_r342.jar, Directory:
C:\Program Files\DAvE-Bench-011\eclipse\plugins\, Size: 309779
Cancel

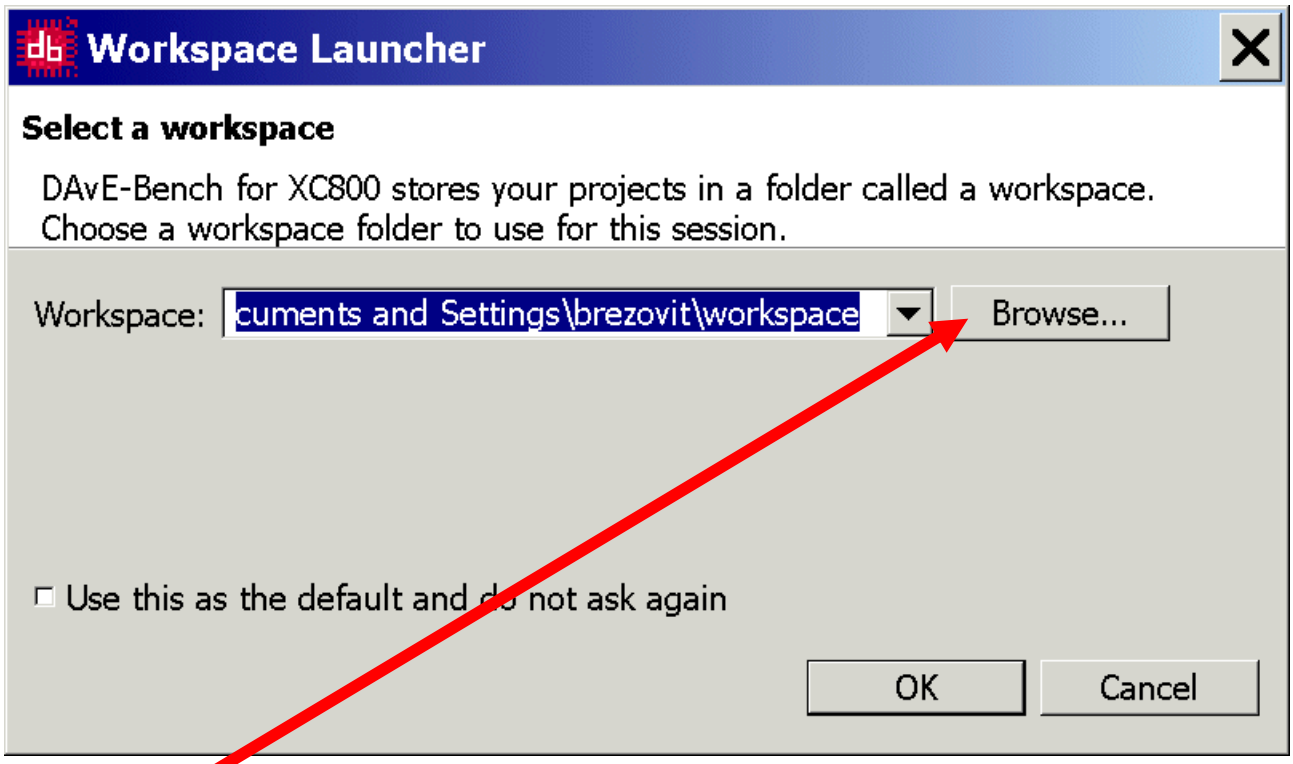

Infineon DAvE-Bench for XC800
Processing
Please wait while Infineon DAvE-Bench-011 is being installed.
DAvE Bench
for XC800
it's Free, it's Easy, it's Serious
Your first program for XC800
runs in less than 15 minutes.
Simply connect your PC via USB
cable to an Infineon XC800 Easy-Kit
and you are ready to go
Easy-Kit
XC878
USB
cable
Compiler
Debugger
Loader
U-SPY
File: make.exe, Directory: C:\Program
Files\DAvE-Bench-011\SDCC_UTILS\, Size: 165888
Cancel


Infineon DAvE-Bench-011 - InstallAware Wizard
Done
DAvE Bench
for XC800
it's Free, it's Easy, it's Serious
Your first program for XC800
runs in less than 15 minutes.
Compiler
Debugger
Loader
U-SPY
Infineon DAvE-Bench-011 has been successfully installed.
Infineon DAvE-Bench-011 requires at least Java Runtime Environment 6 (JRE 1.6). Please select one of the install options below:
Offline - internet connection is not required
J2SE Runtime Environment(JRE) 6, Update17, provided together with Infineon DAvE-Bench-011.
Online - internet connection required
Java home page http://www.java.com is opened.The connected host may be checked for current installed JRE and updated to the latest one.
Install Java Runtime Environment later.
Finish

Click 
Finish

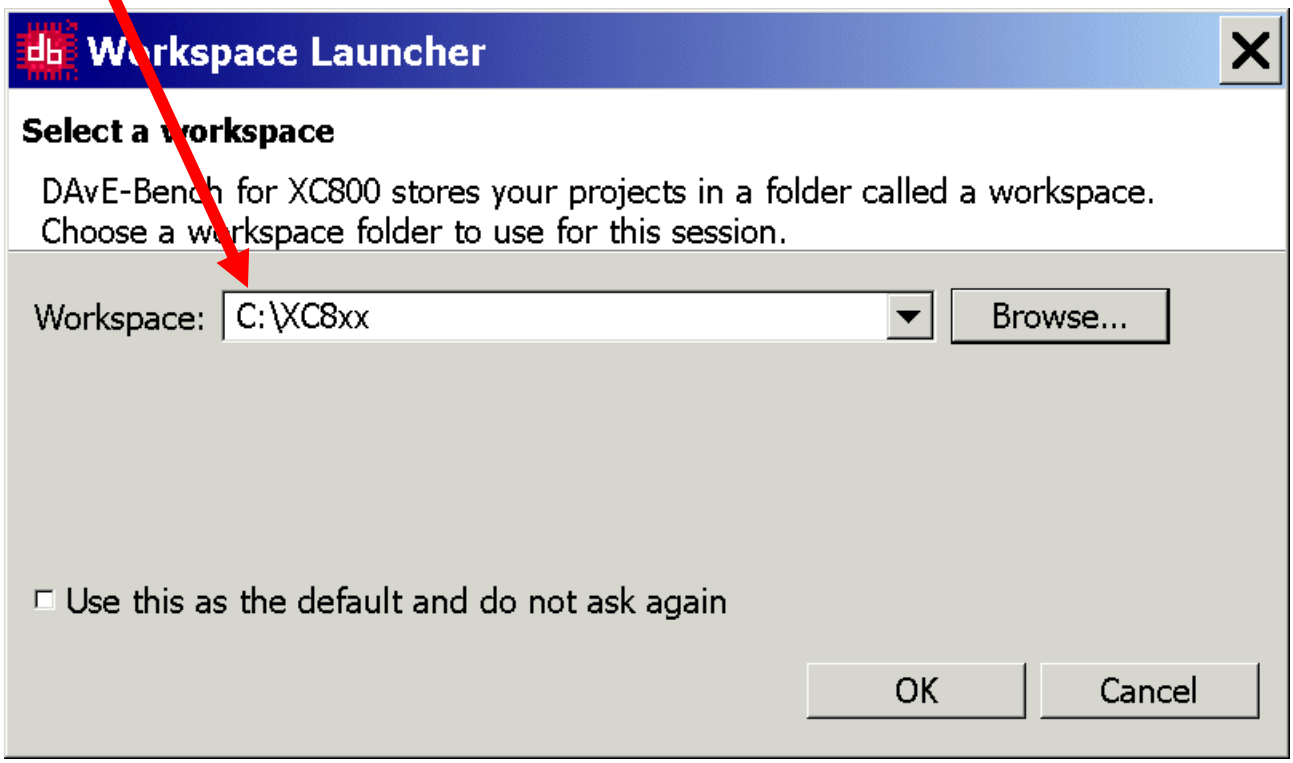
Start/Launch DAvE Bench and open/add the DAvE Project:



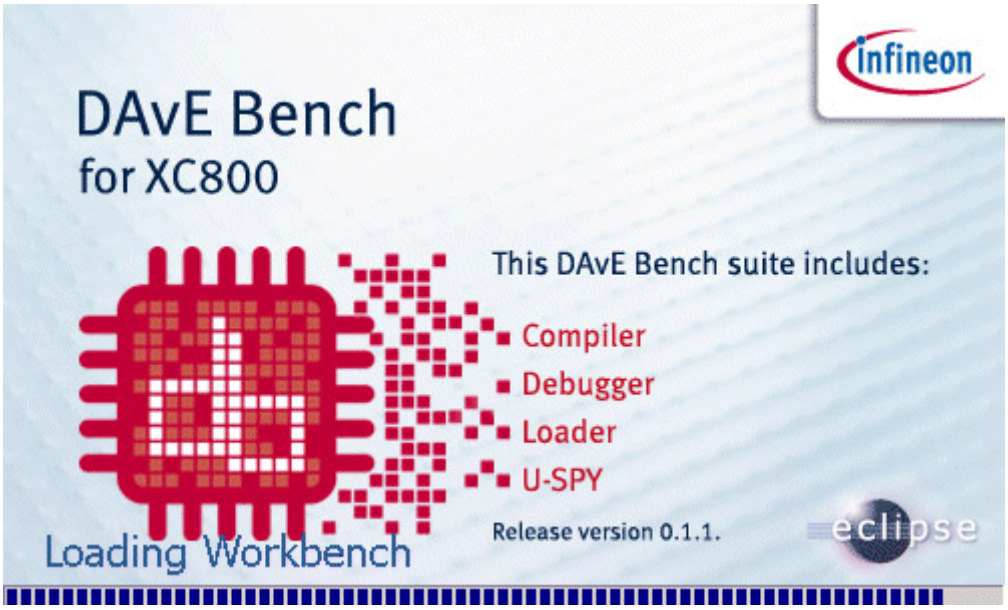


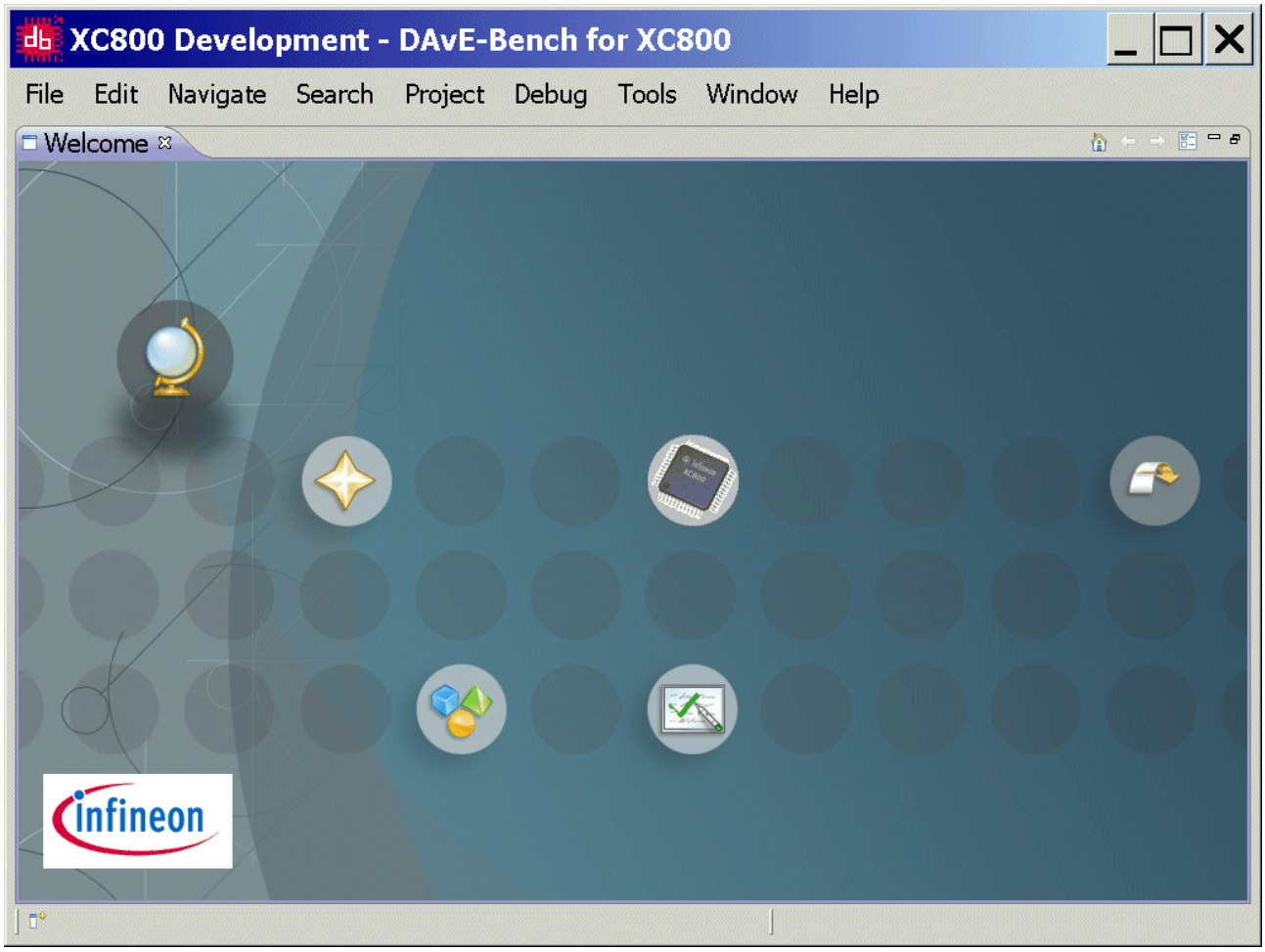
Click Browse…

and select C:\XC8xx:

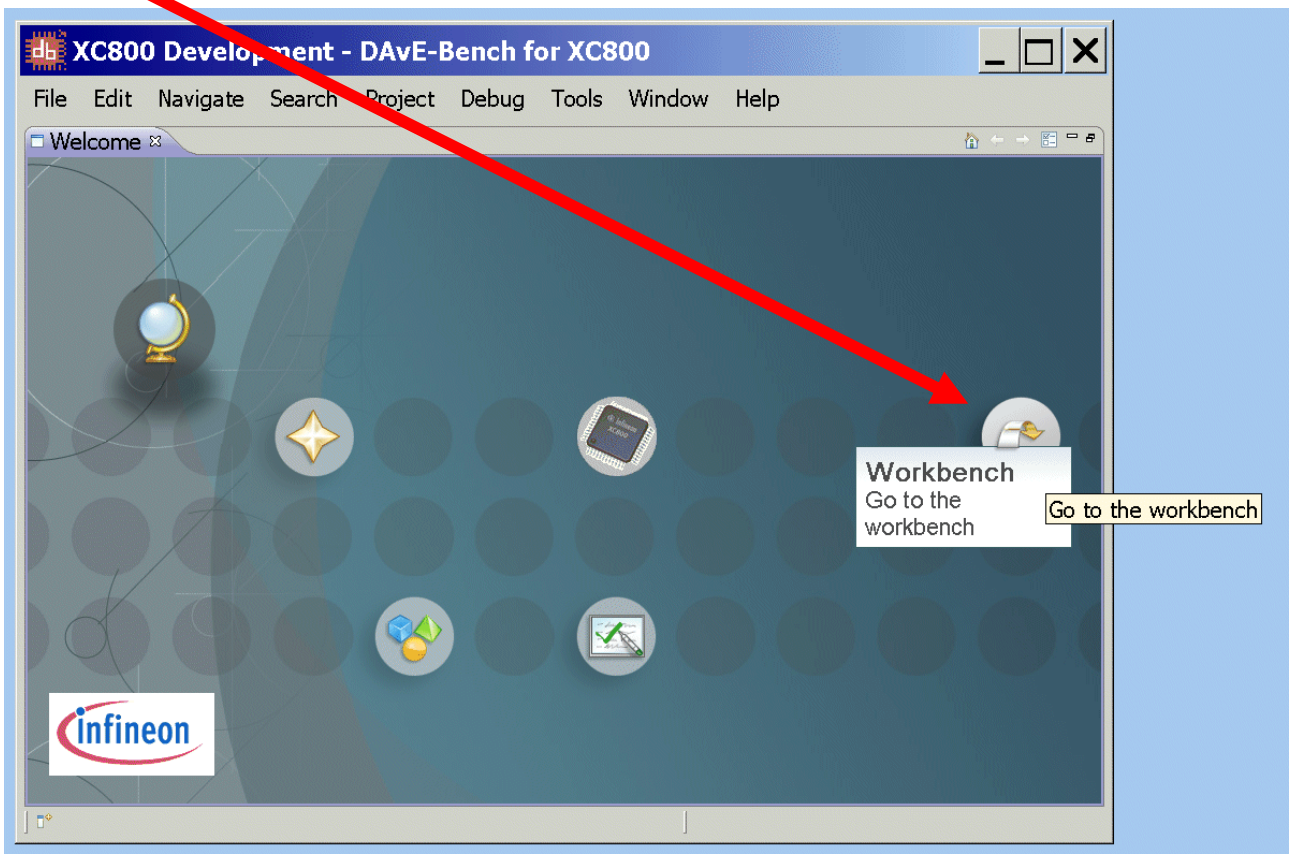


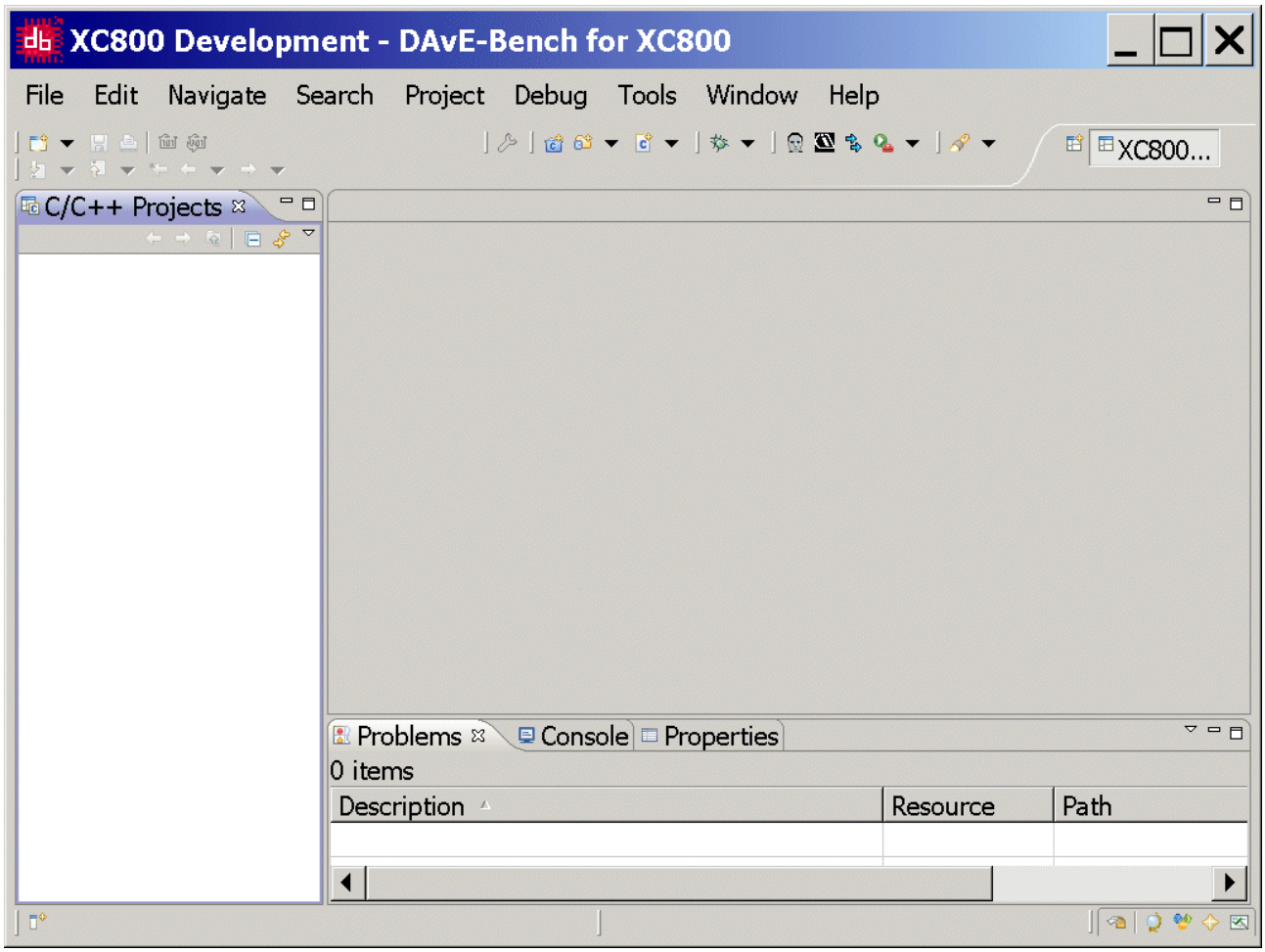
OK - OK


DAvE Bench
for XC800
This DAvE Bench suite includes:
Compiler
Debugger
Loader
U-SPY
Release version 0.1.1.
Loading Workbench
eclipse

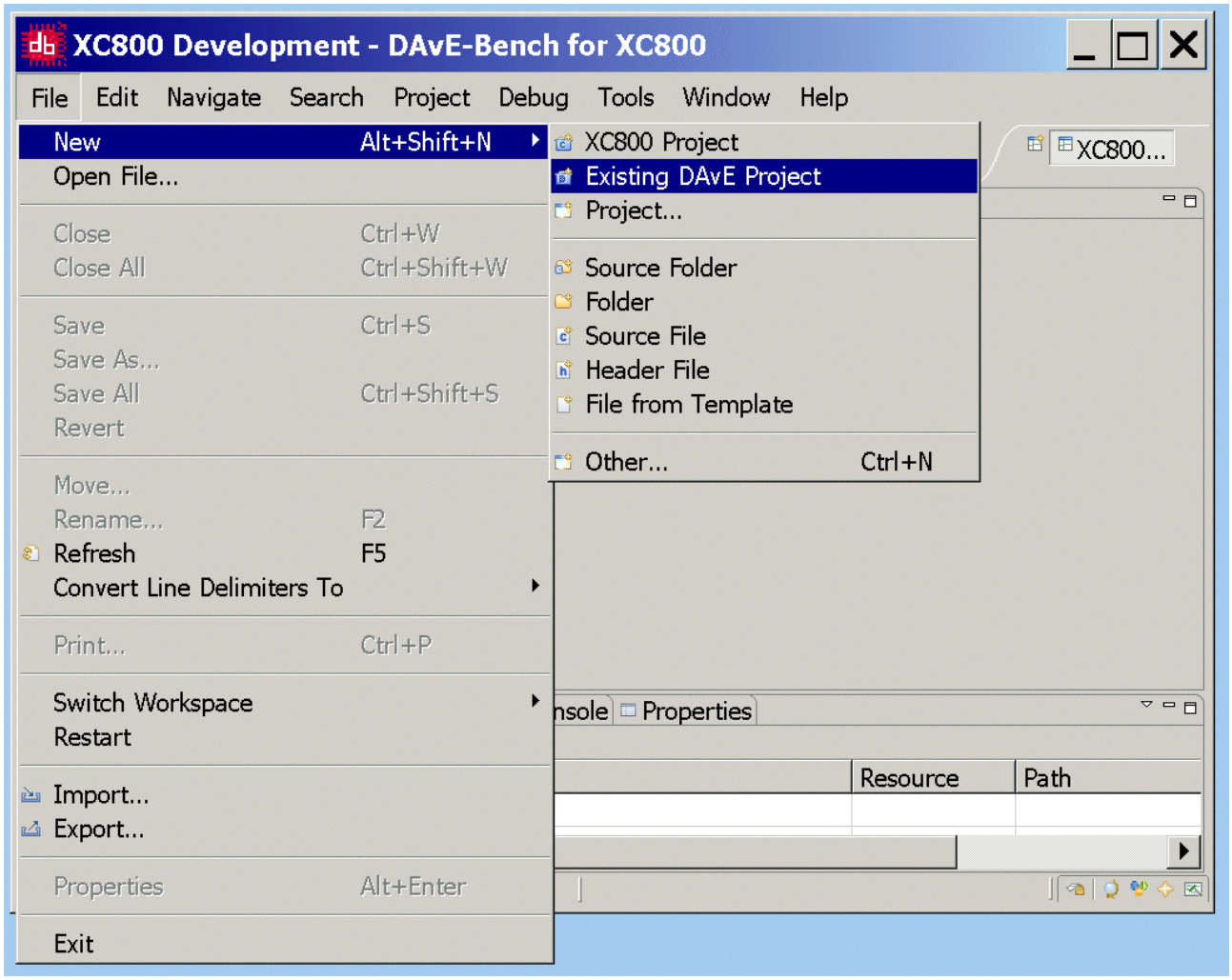

XC800 Development - DAvE-Bench for XC800
File
Edit
Navigate
Search
Project
Debug
Tools
Window
Help
Welcome

Click “Go to the workbench”:



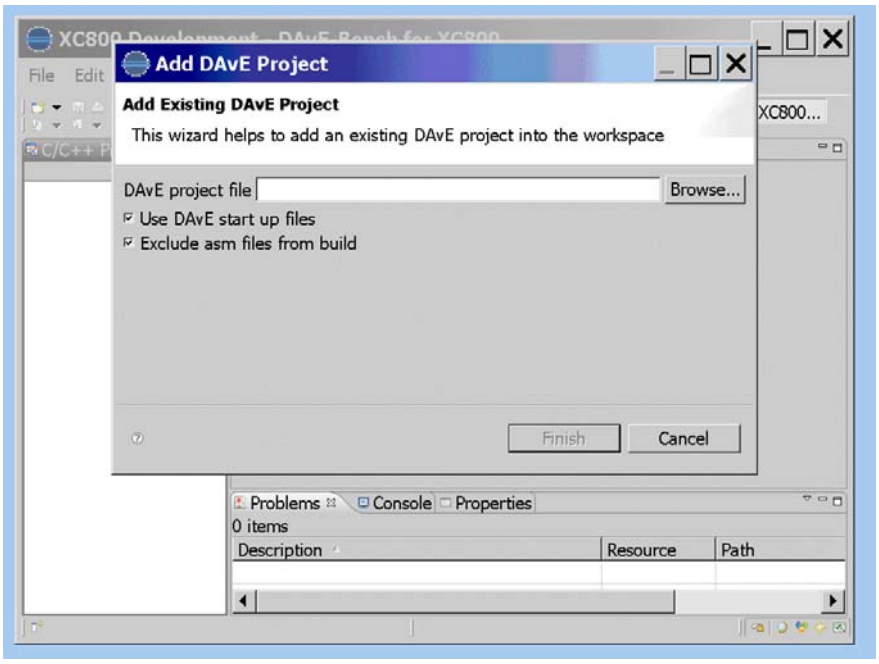

XC800 Development - DAvE-Bench for XC800
File
Edit
Navigate
Search
Project
Debug
Tools
Window
Help
XC800...
C/C++ Projects
Problems
Console
Properties
0 items
Description
Resource
Path

File – New – Existing DAvE Project

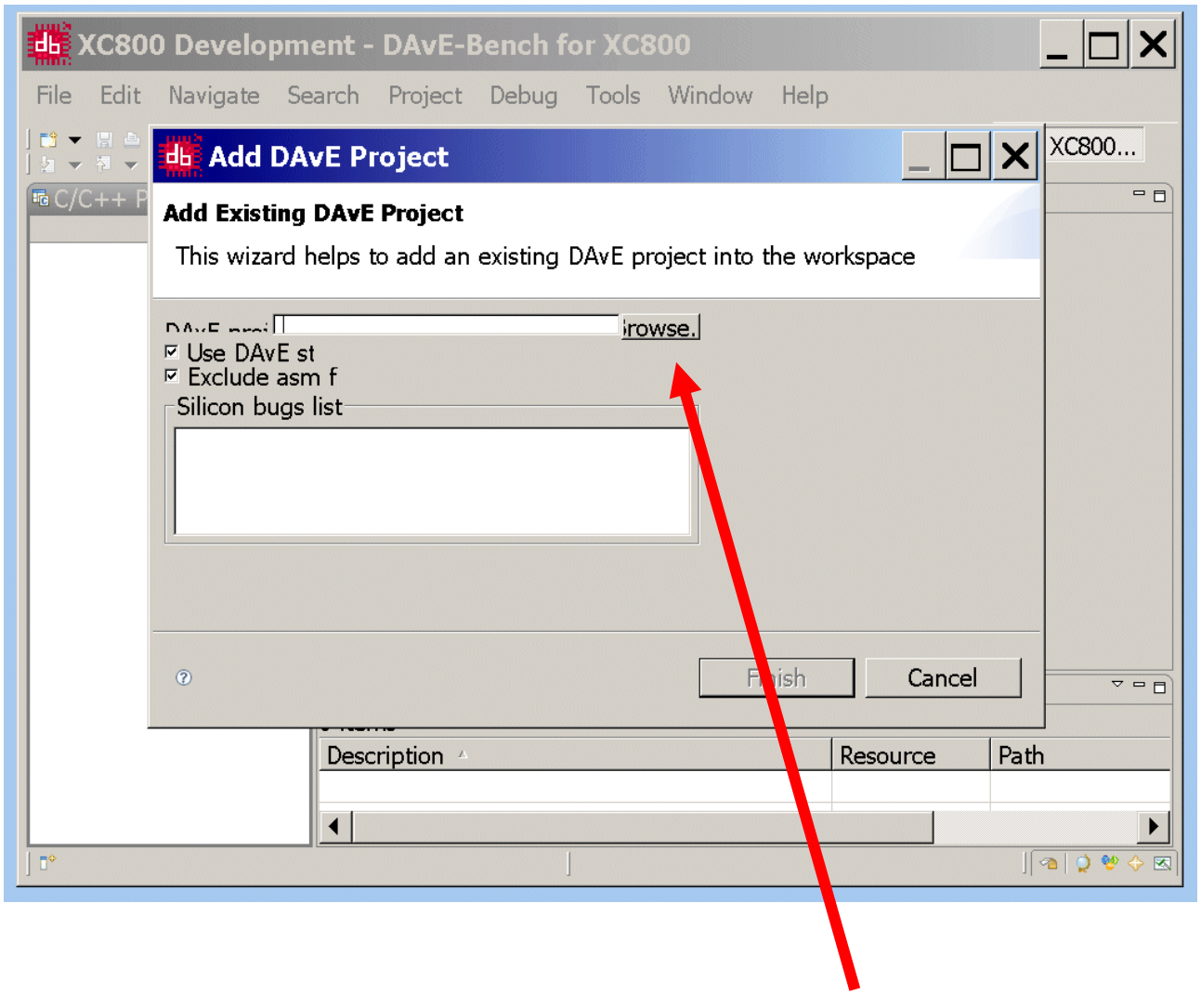


☹

This dialog in DAvE Bench is so illegible. Therefore we show the dialog from the old DAvE Bench version.

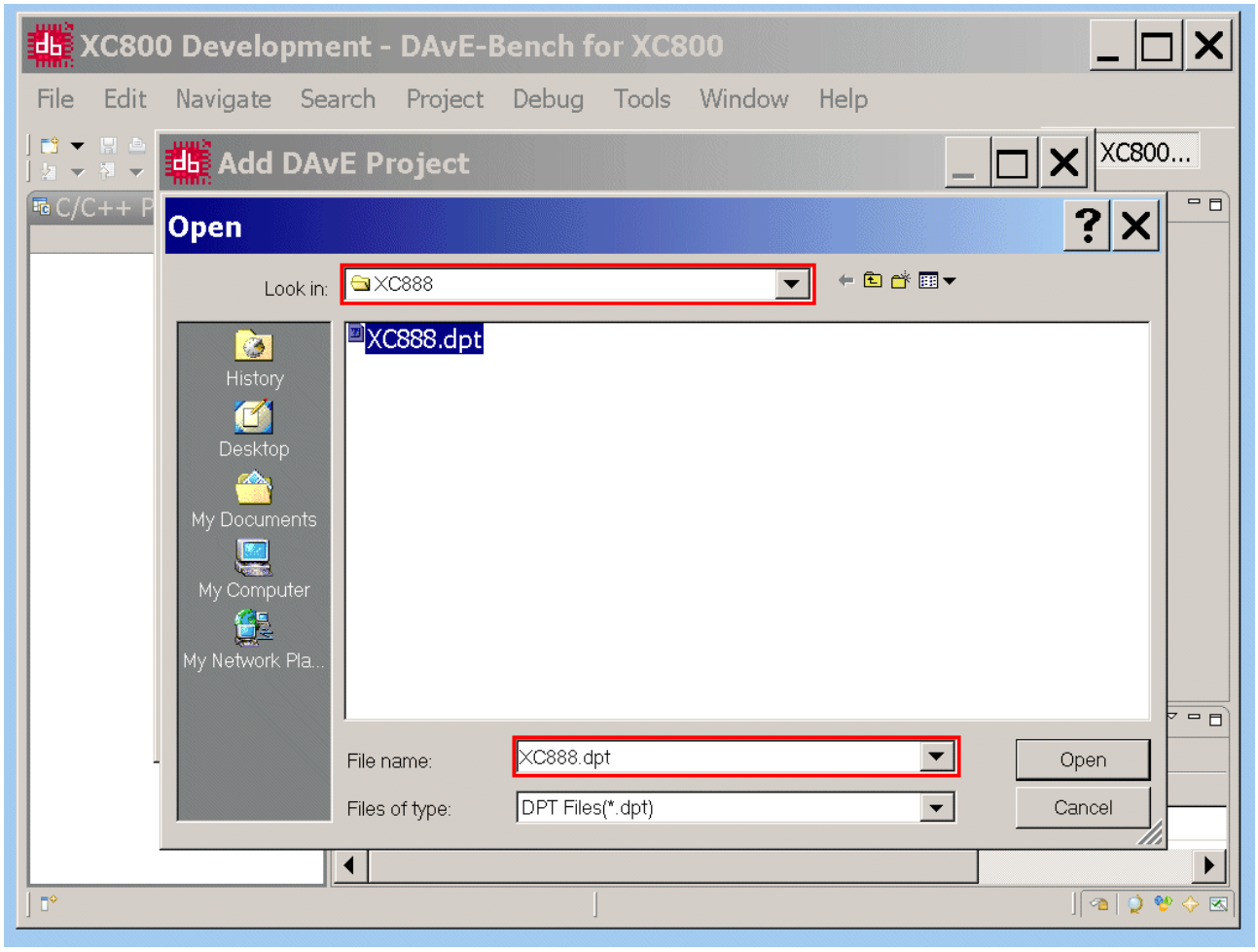
(Old DAvE Bench Version: 

)



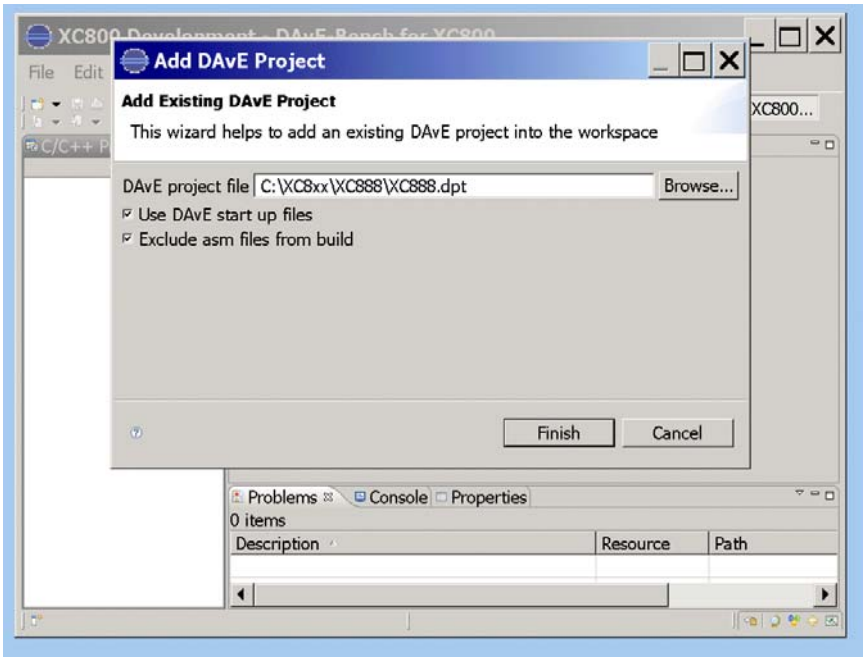
Add DAvE Project: Add Existing DAvE Project: DAvE project file: click Browse…

Open: Look in: select C:\XC8xx\XC888
Open: File name: select XC888.dpt

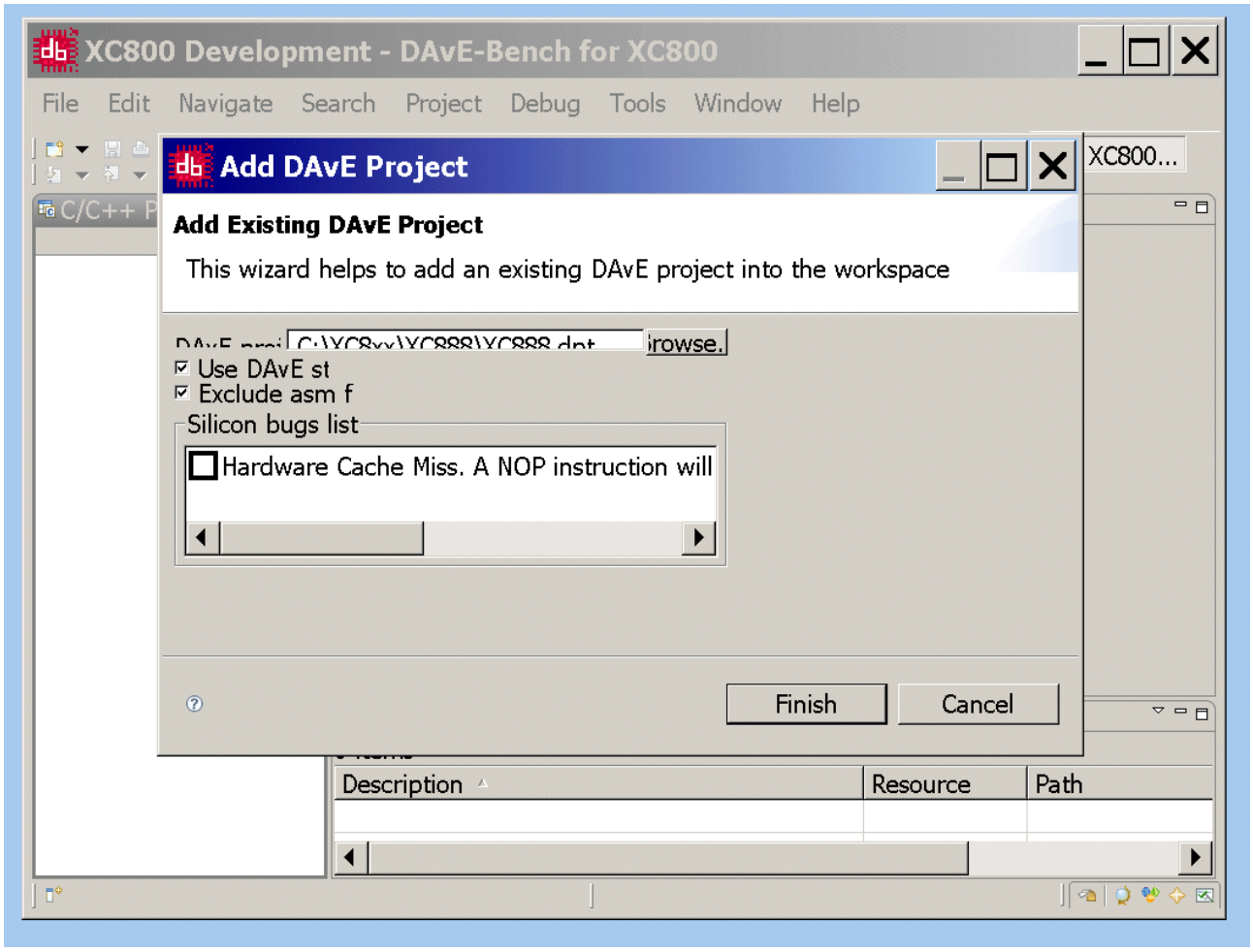


Open

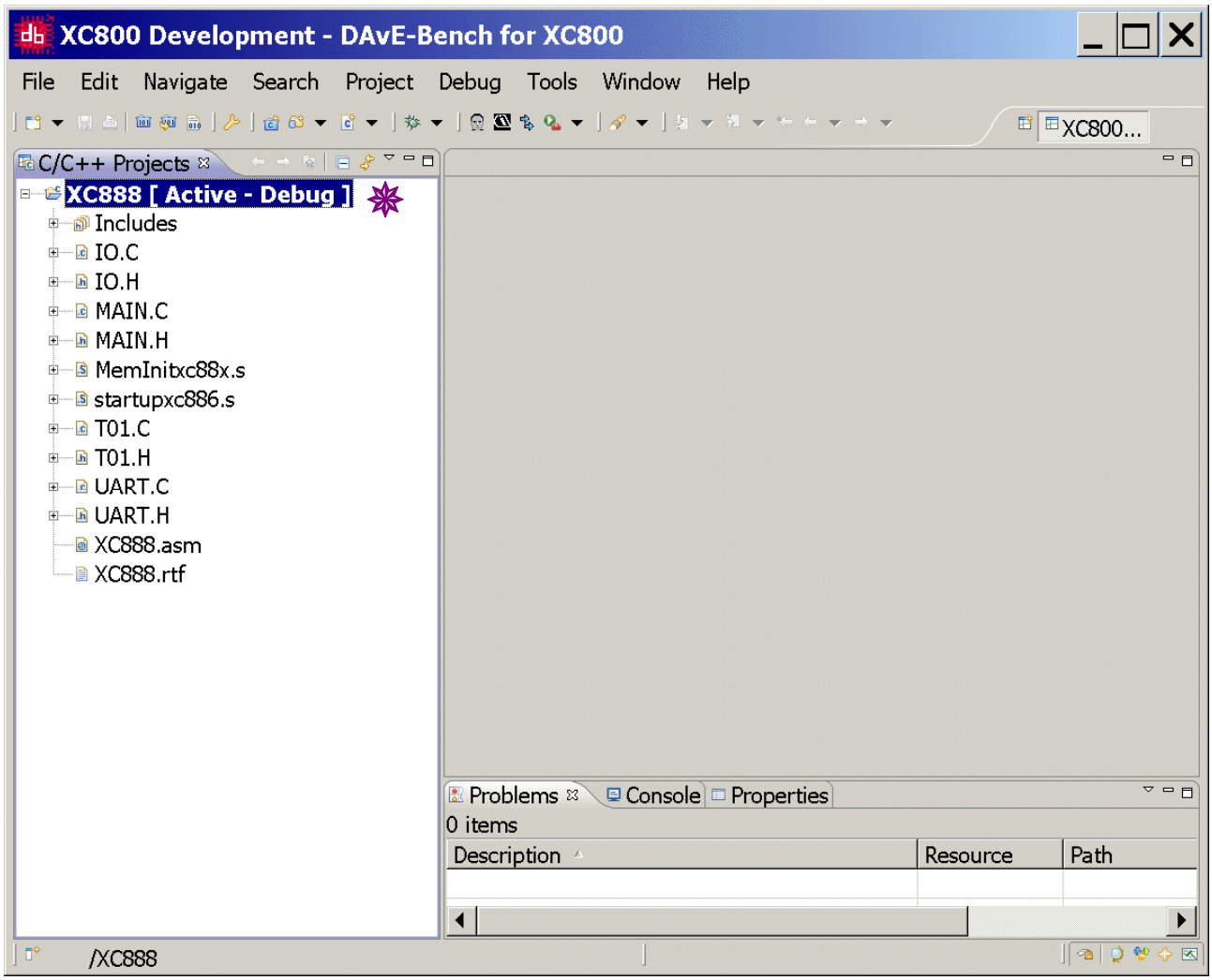
This dialog in DAvE Bench is so illegible. Therefore we show the dialog from the old DAvE Bench version.

(Old DAvE Bench Version: 

)

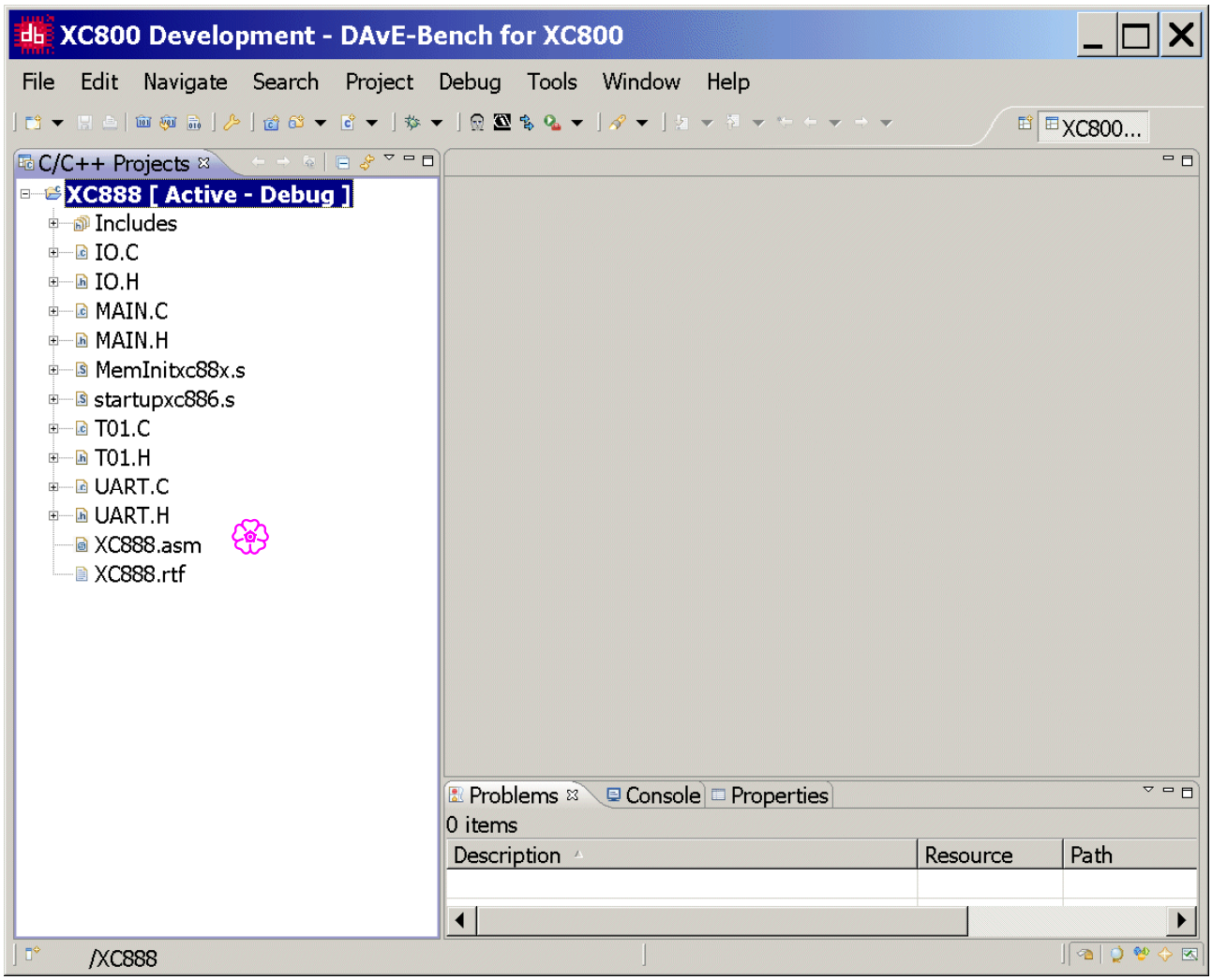


Finish



Note:

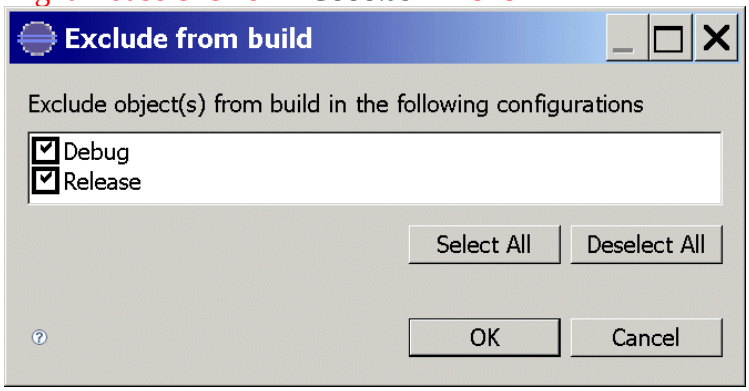
Check that the active project is **XC888 [Active - Debug]** and NOT **XC888 [Active - Release]**.



Note:

Check that XC888.asm is NOT part of the build process:

Right mouse click on XC888.asm – click Exclude from build...



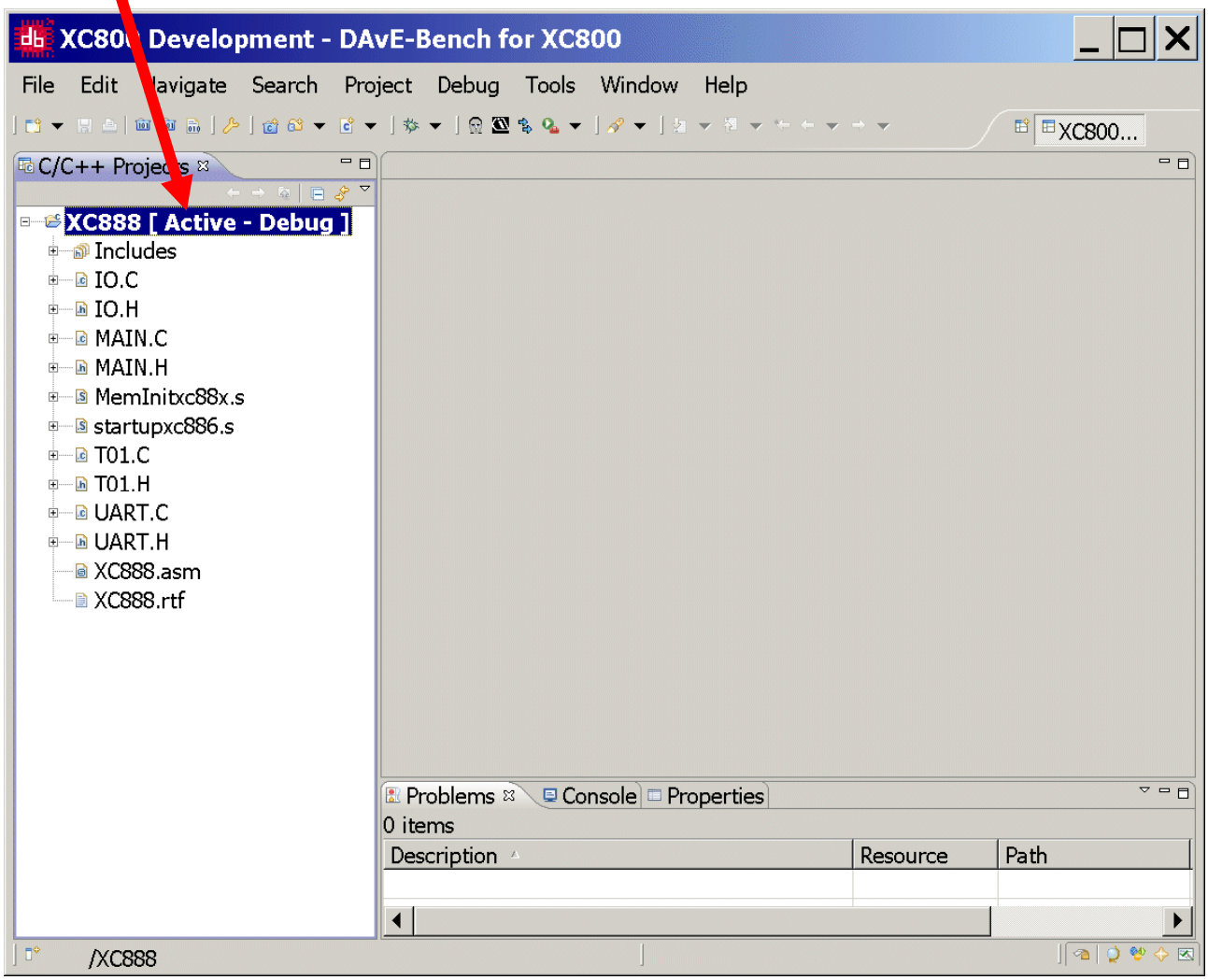
OK

Note:
The DAvE created files can be directly accessed in the IDE (DAvE Bench) or in DAvE.
The user has to make sure that no conflicts or data losses happen.
This can be avoided by saving the files before switching from the IDE to DAvE or vice versa.

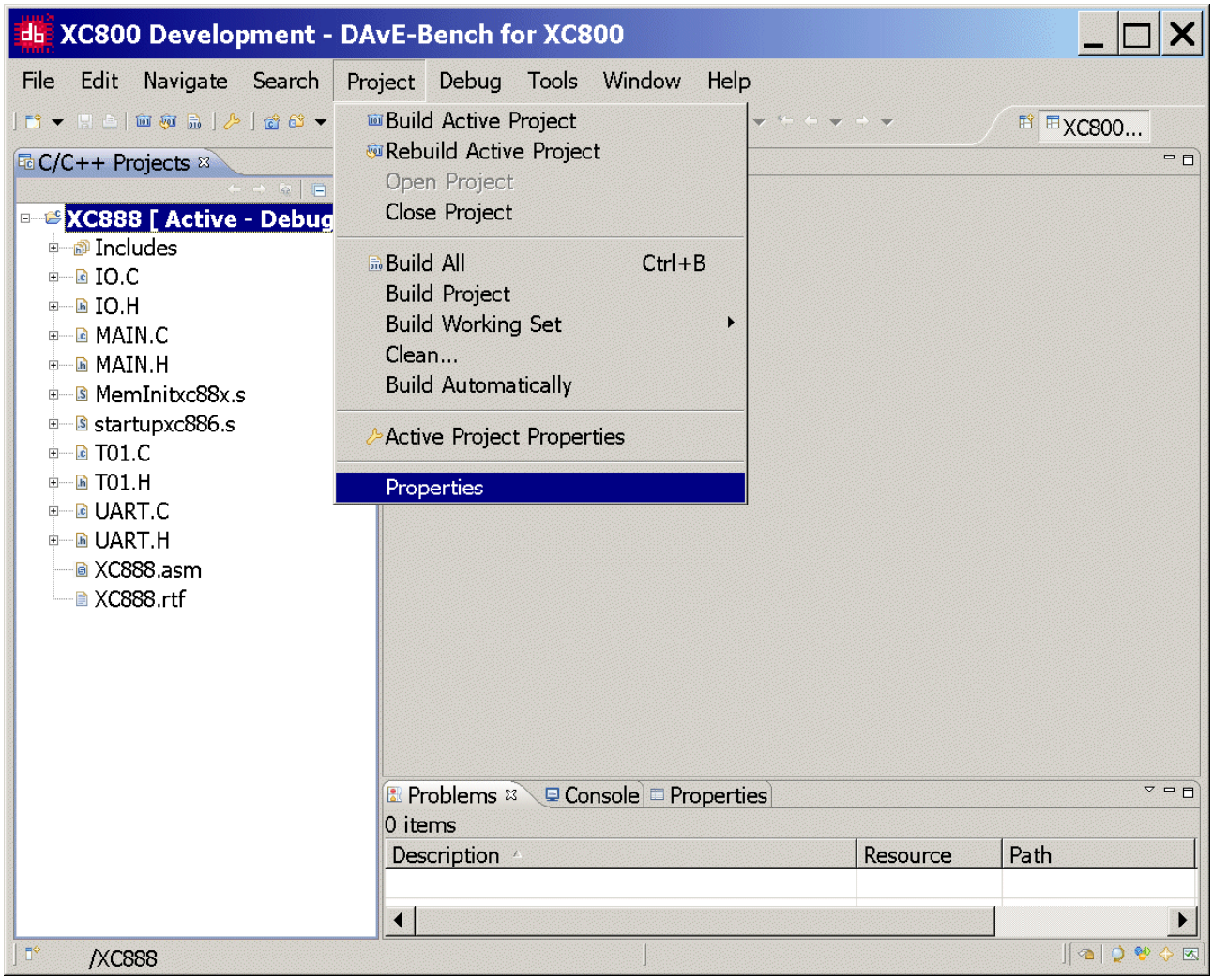
Note:
The following pages/screenshots exist for documentation purposes only.
There is nothing to do!
If you are in a hurry, we suggest you jump to page 132 (Insert your application specific program) and continue working there.

Configure Compiler, Assembler, Linker, Locater, Hex-Converter, Build – Control, Debugger and Utilities:

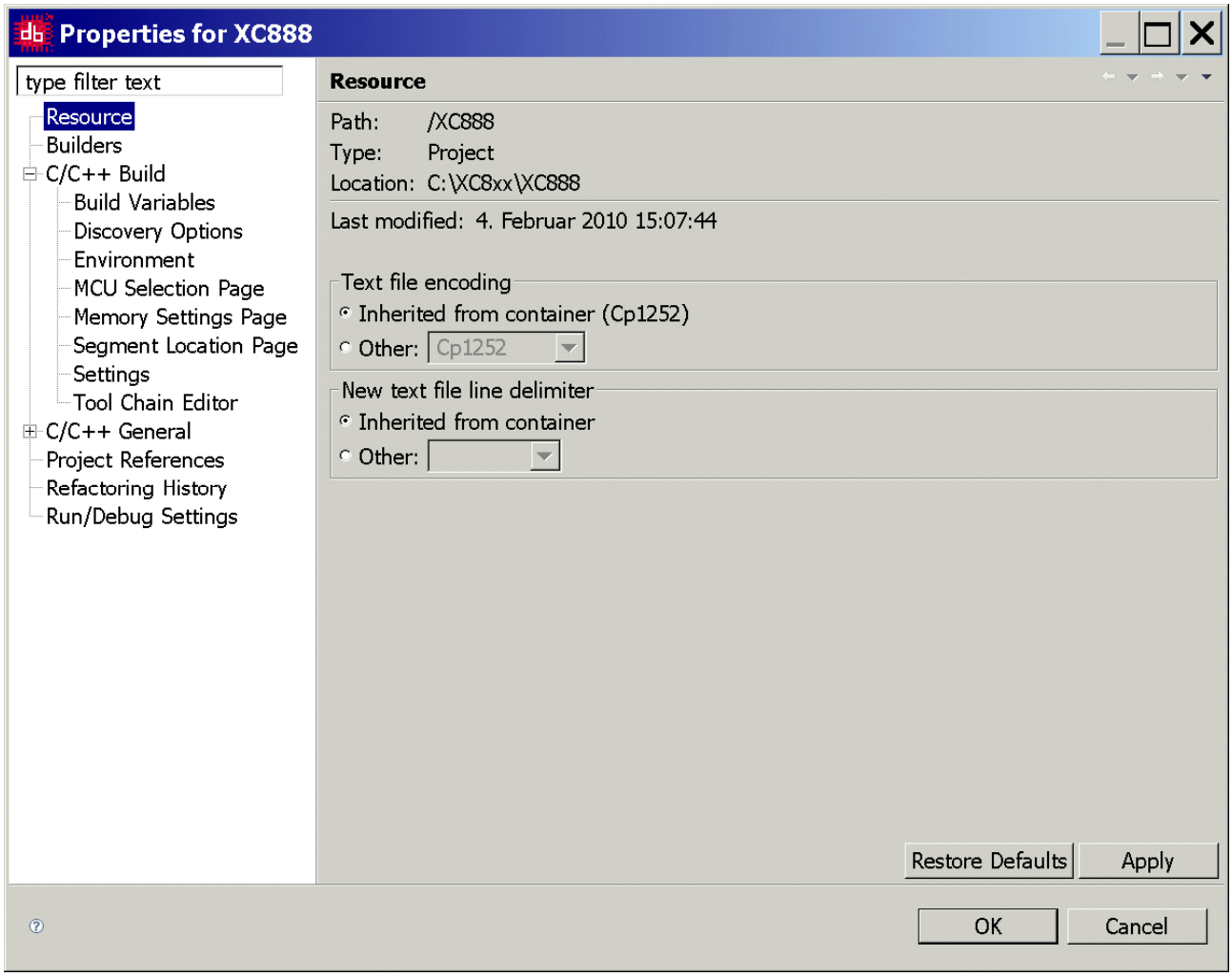
Click/Mark (left mouse click on) **XC888 [Active - Debug]**



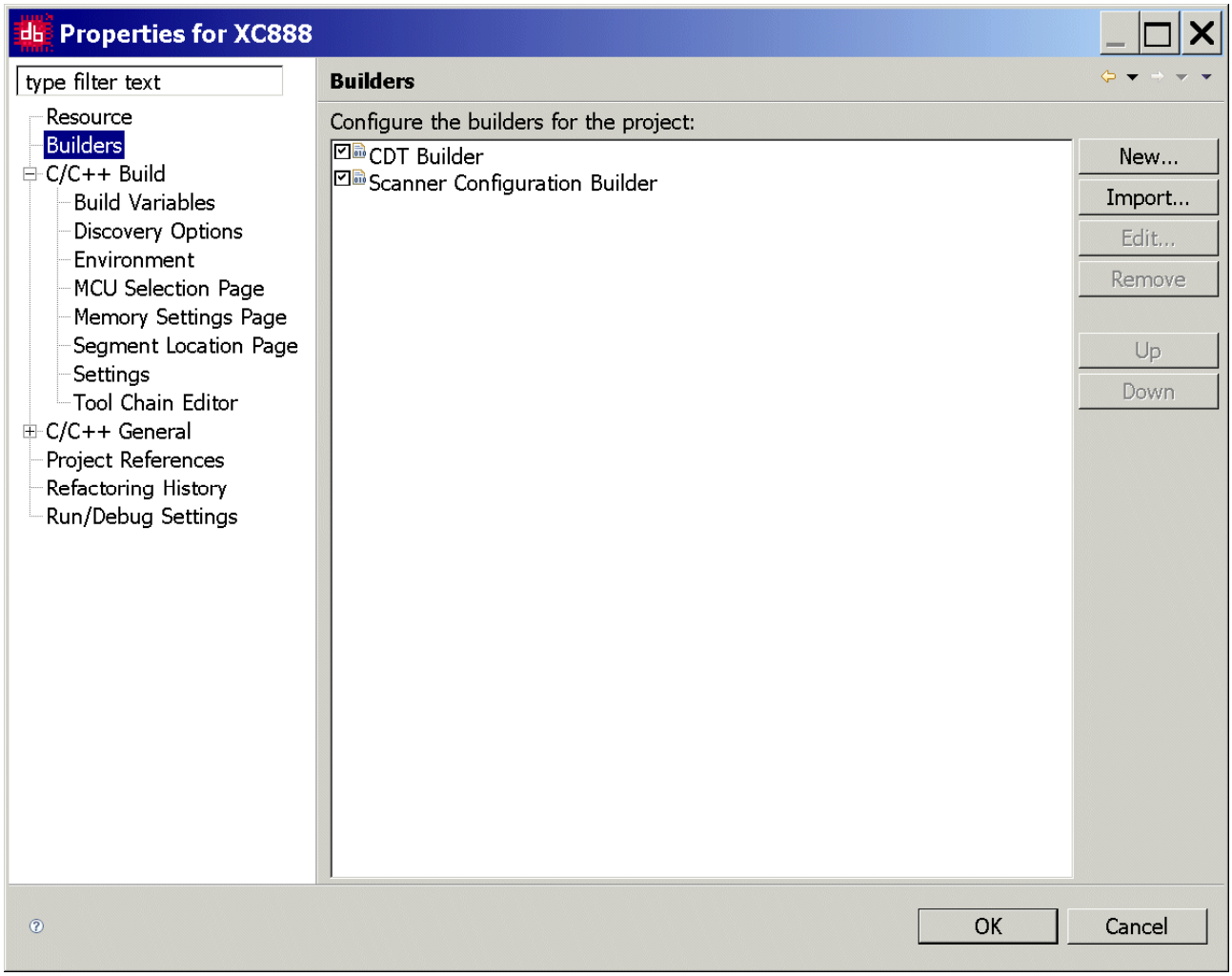
Project - Properties



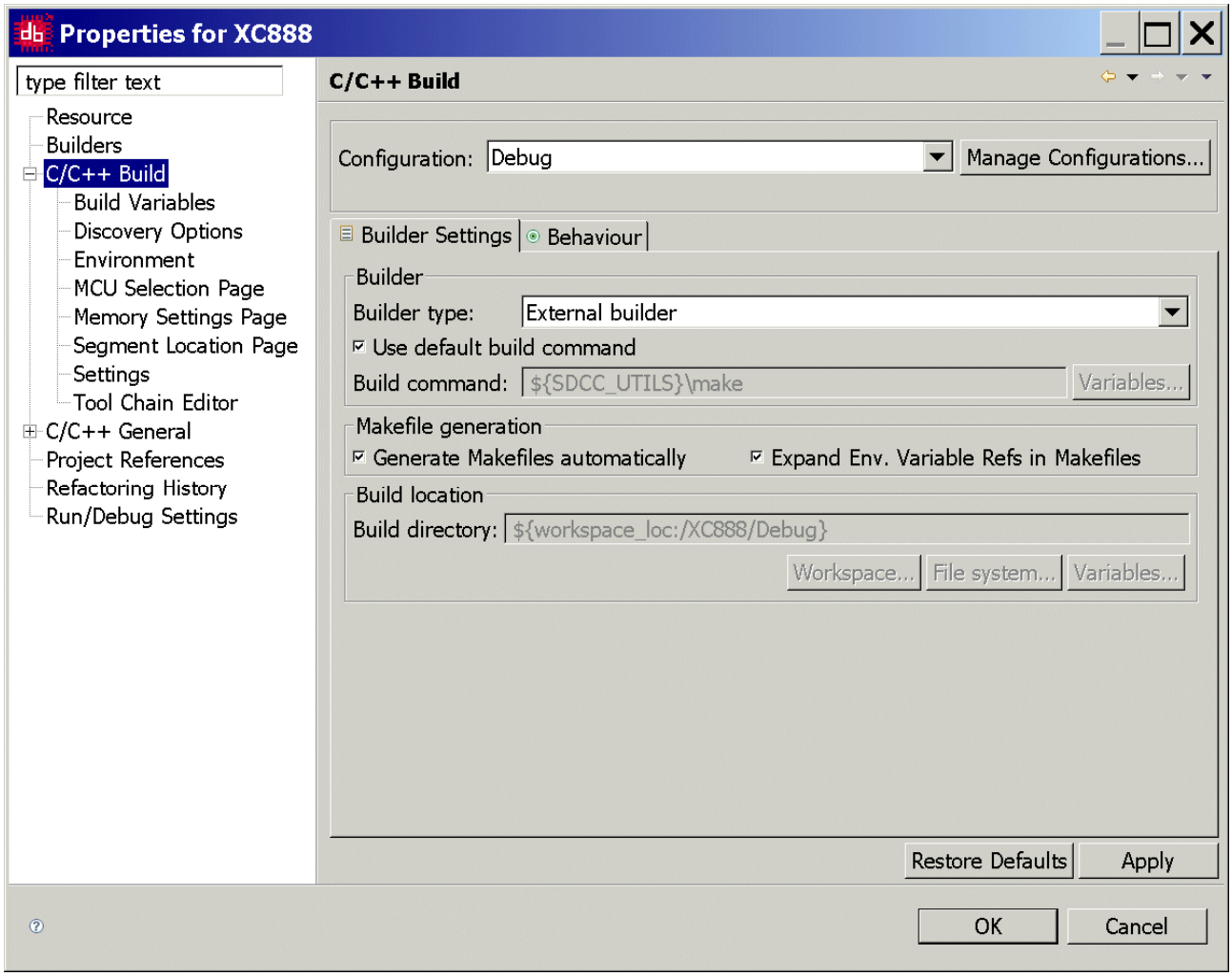
Project – Properties: Resource:



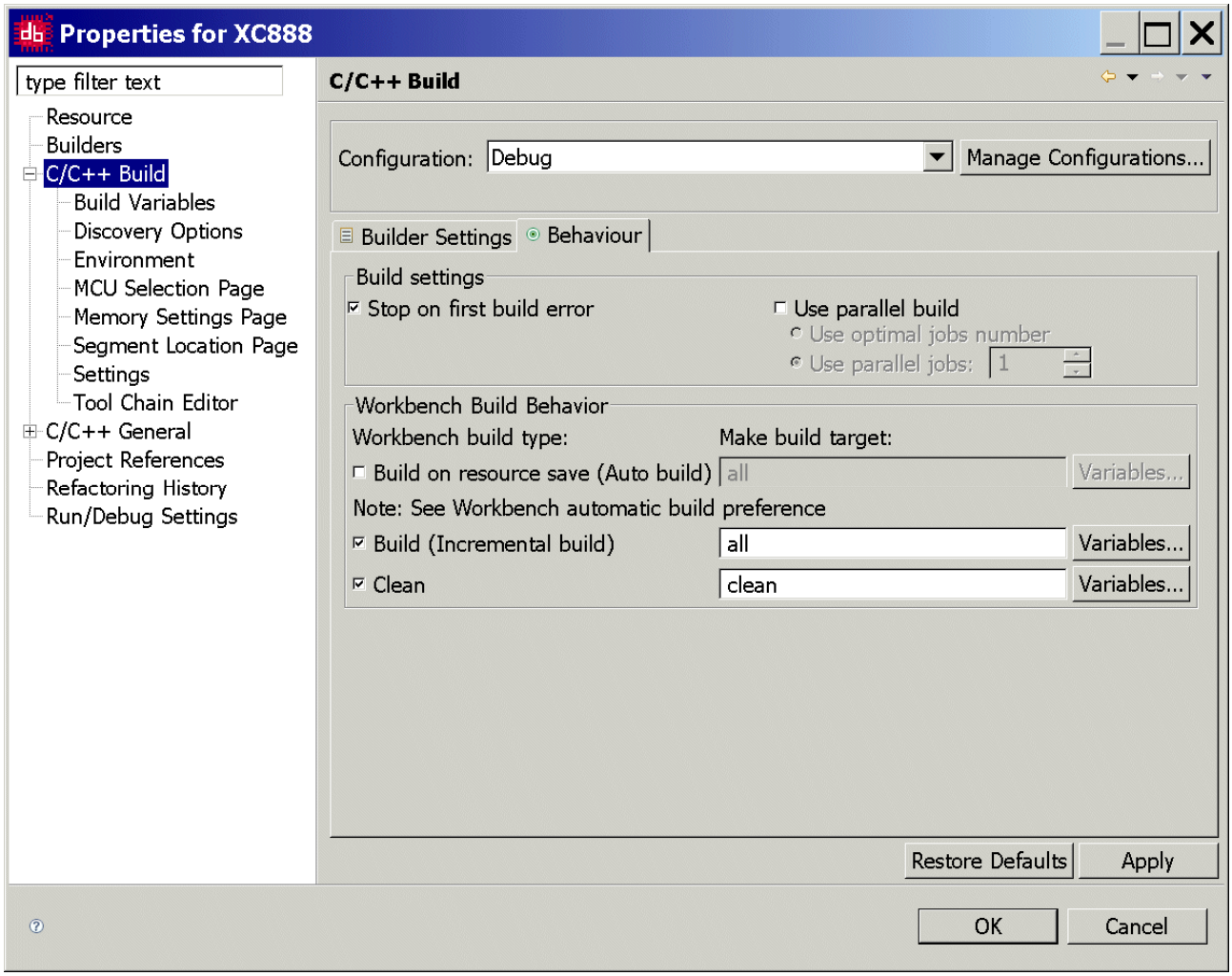
Project – Properties: Builders:



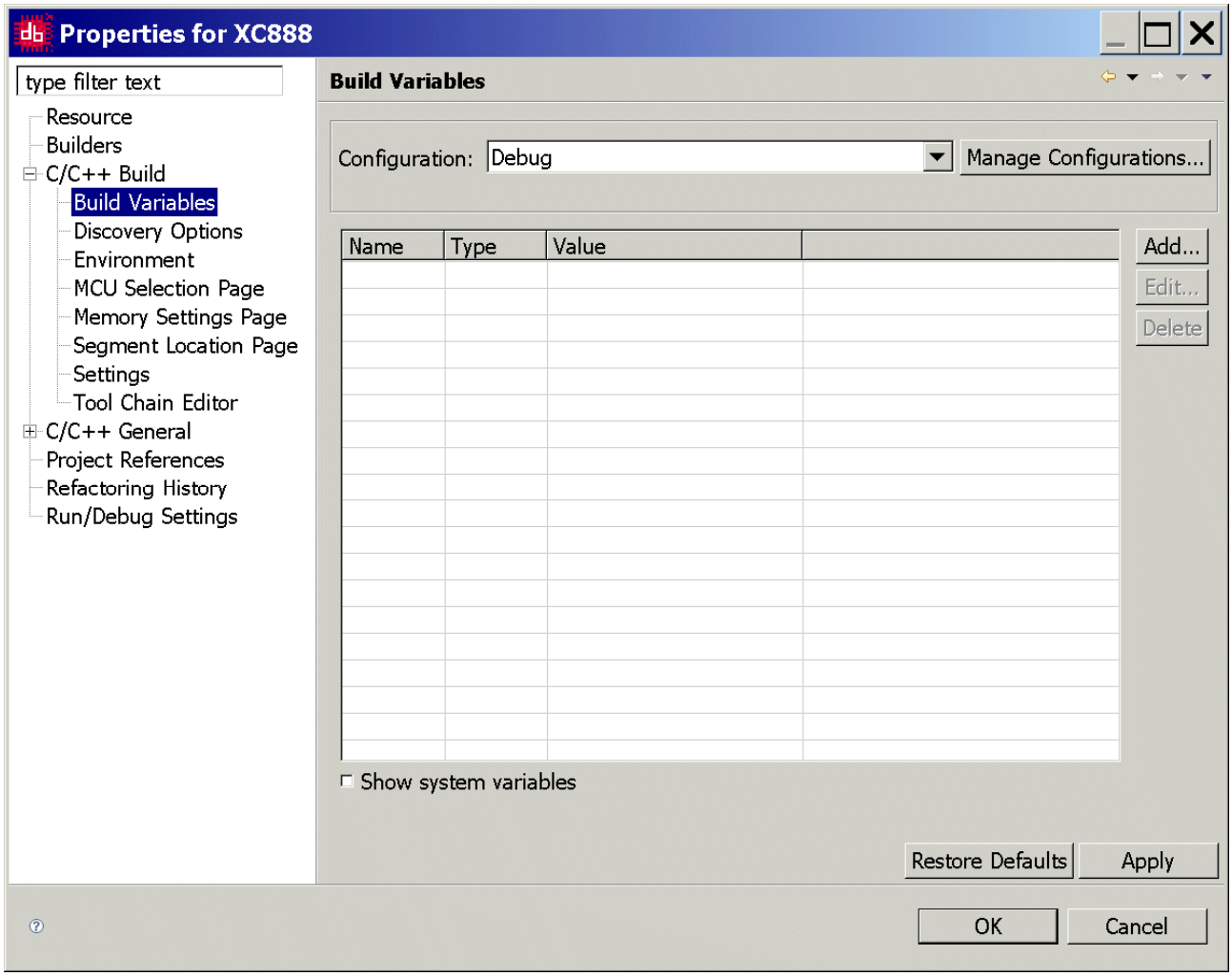
Project – Properties: C/C++ Build: Builder Settings:



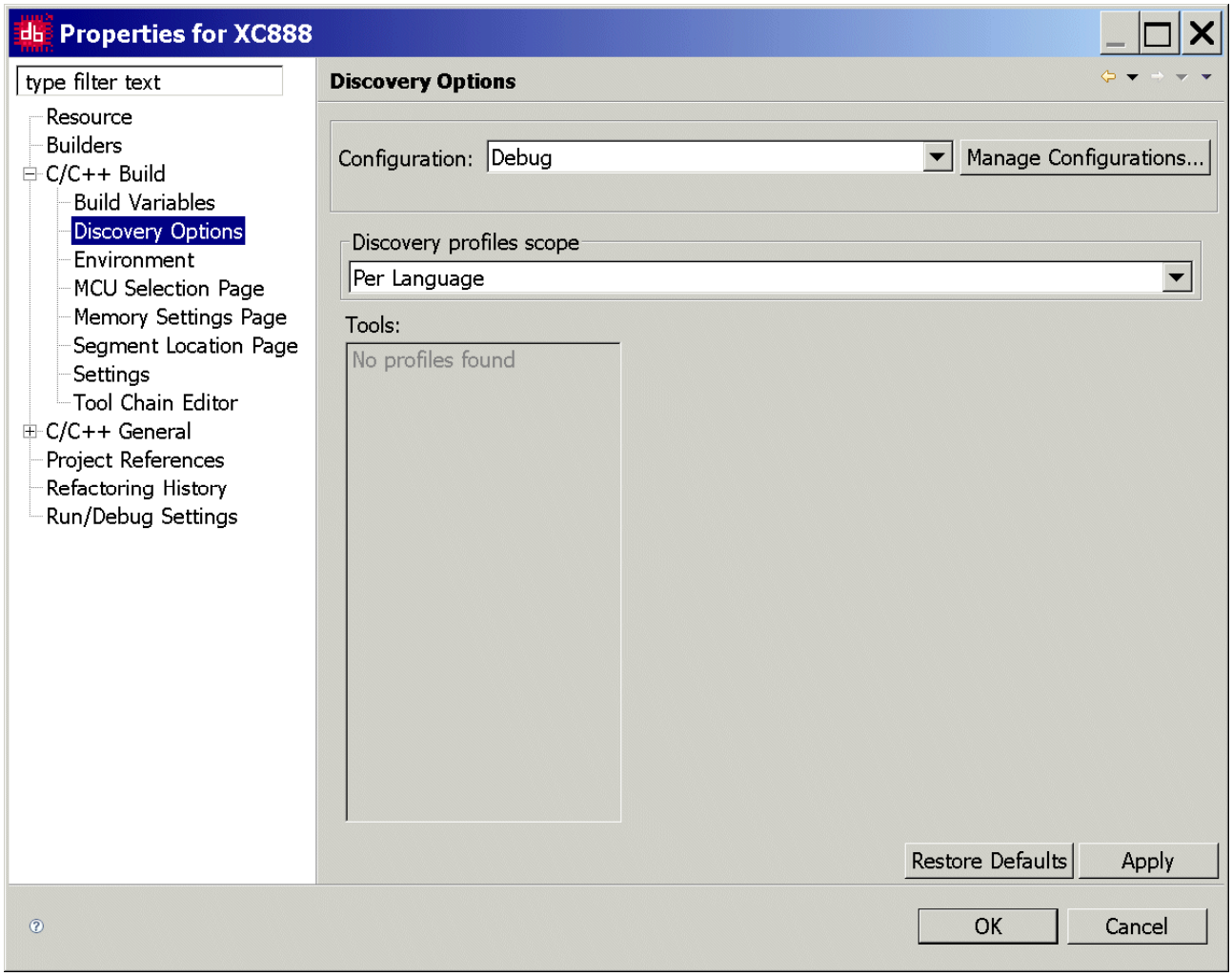
Project – Properties: C/C++ Build: Behaviour:



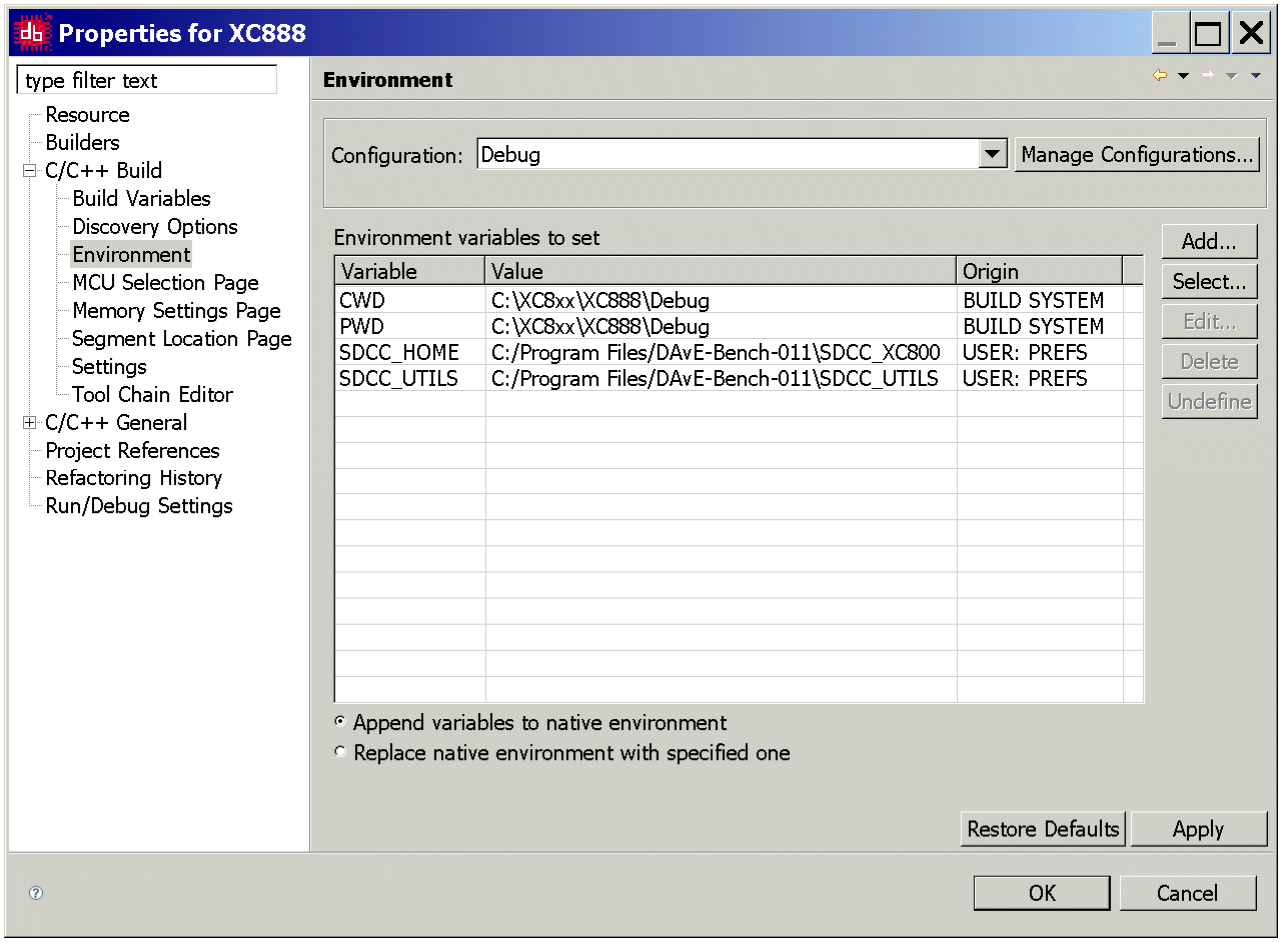
Project – Properties: C/C++ Build: Build Variables:



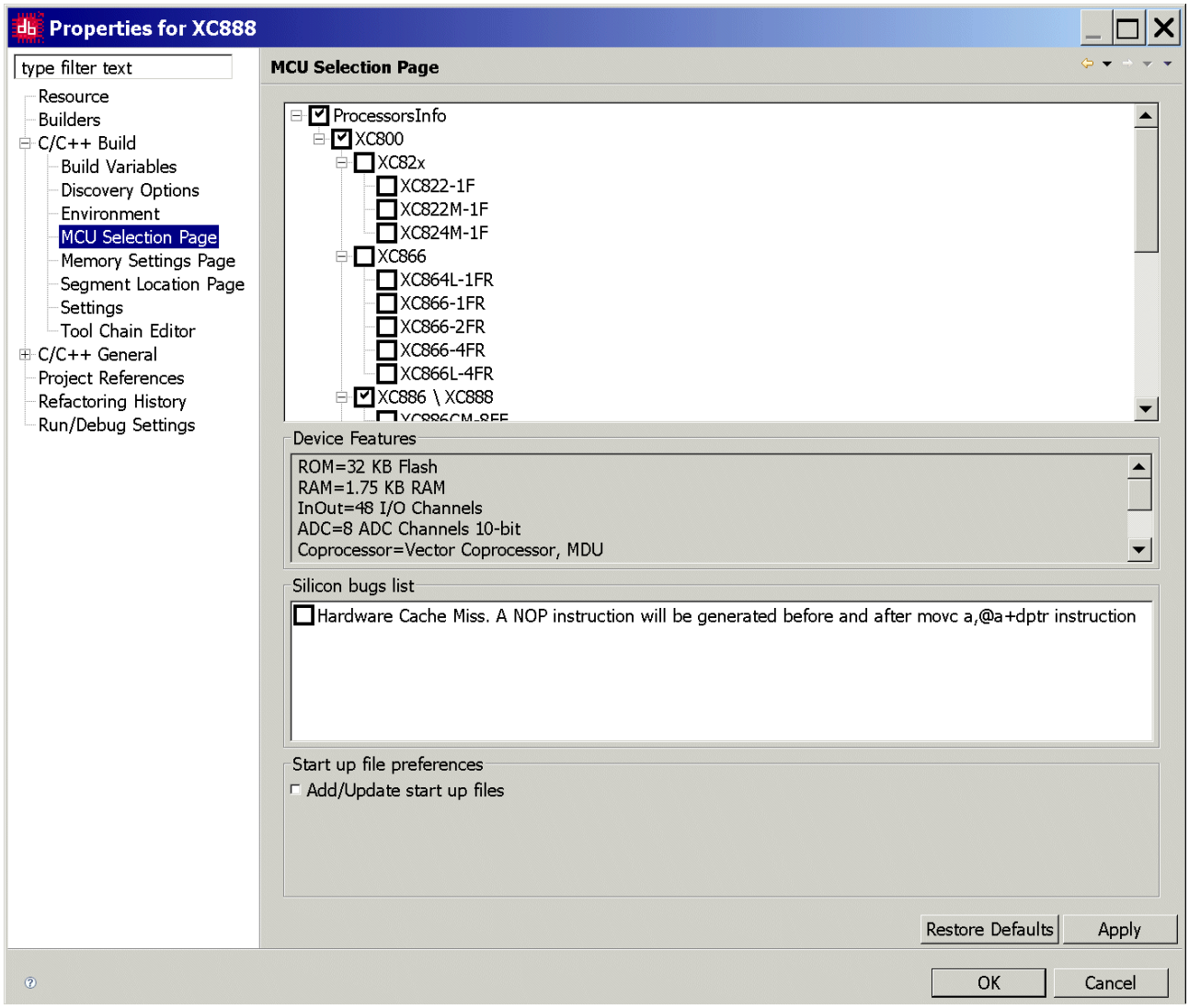
Project – Properties: C/C++ Build: Discovery Options:



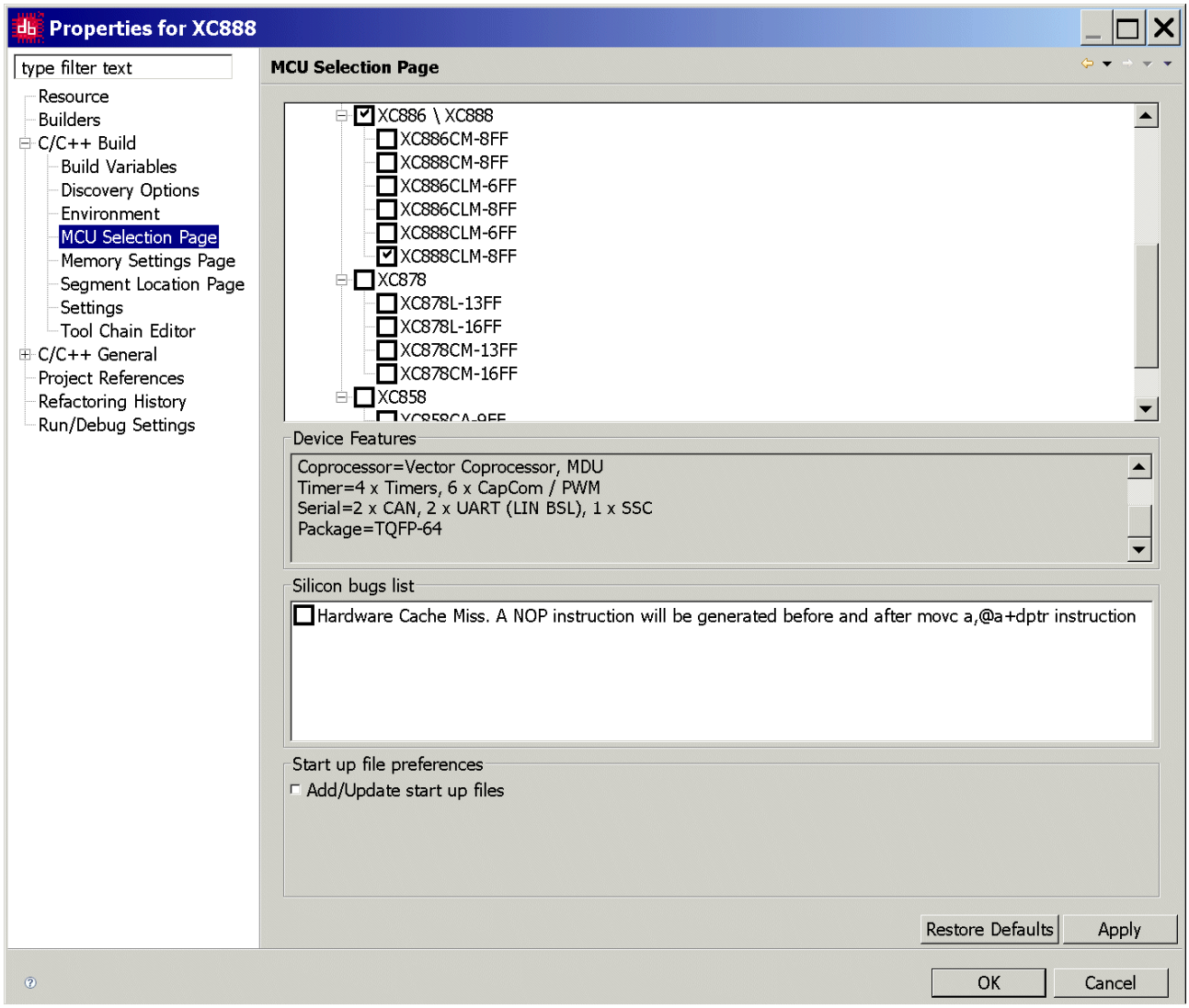
Project – Properties: C/C++ Build: Environment:



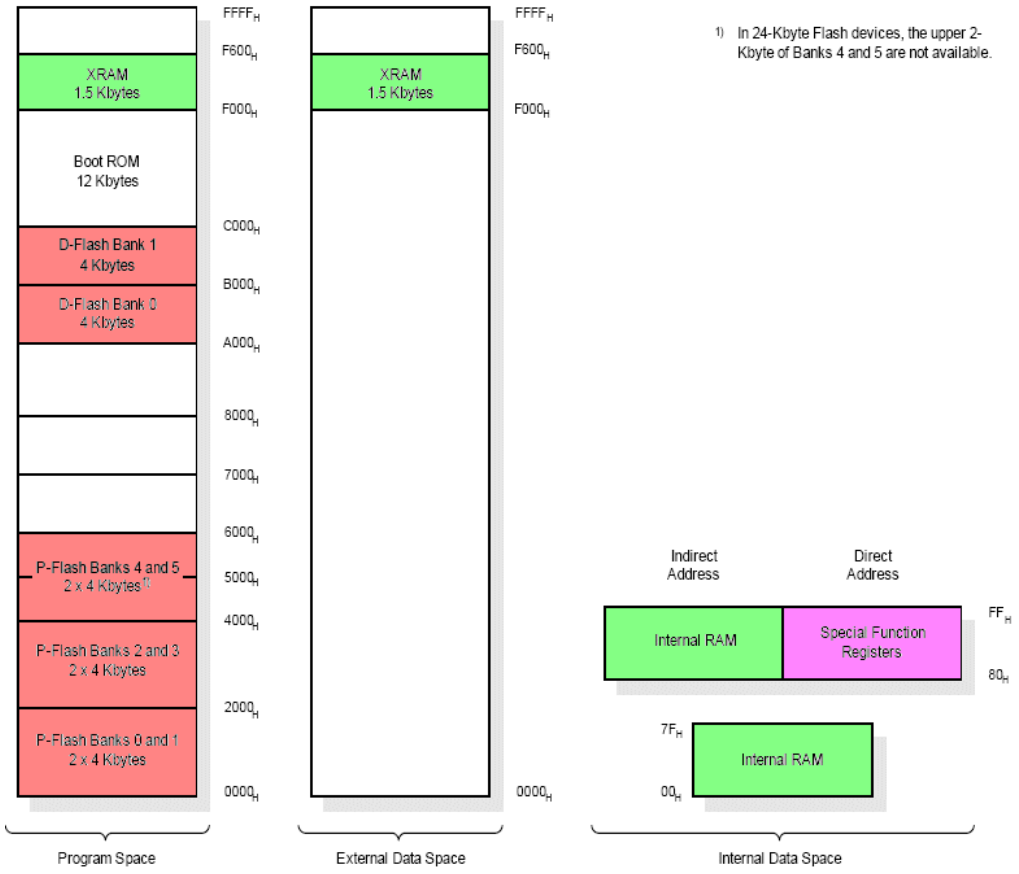
Project – Properties: C/C++ Build: MCU Selection Page (part 1 of 2):



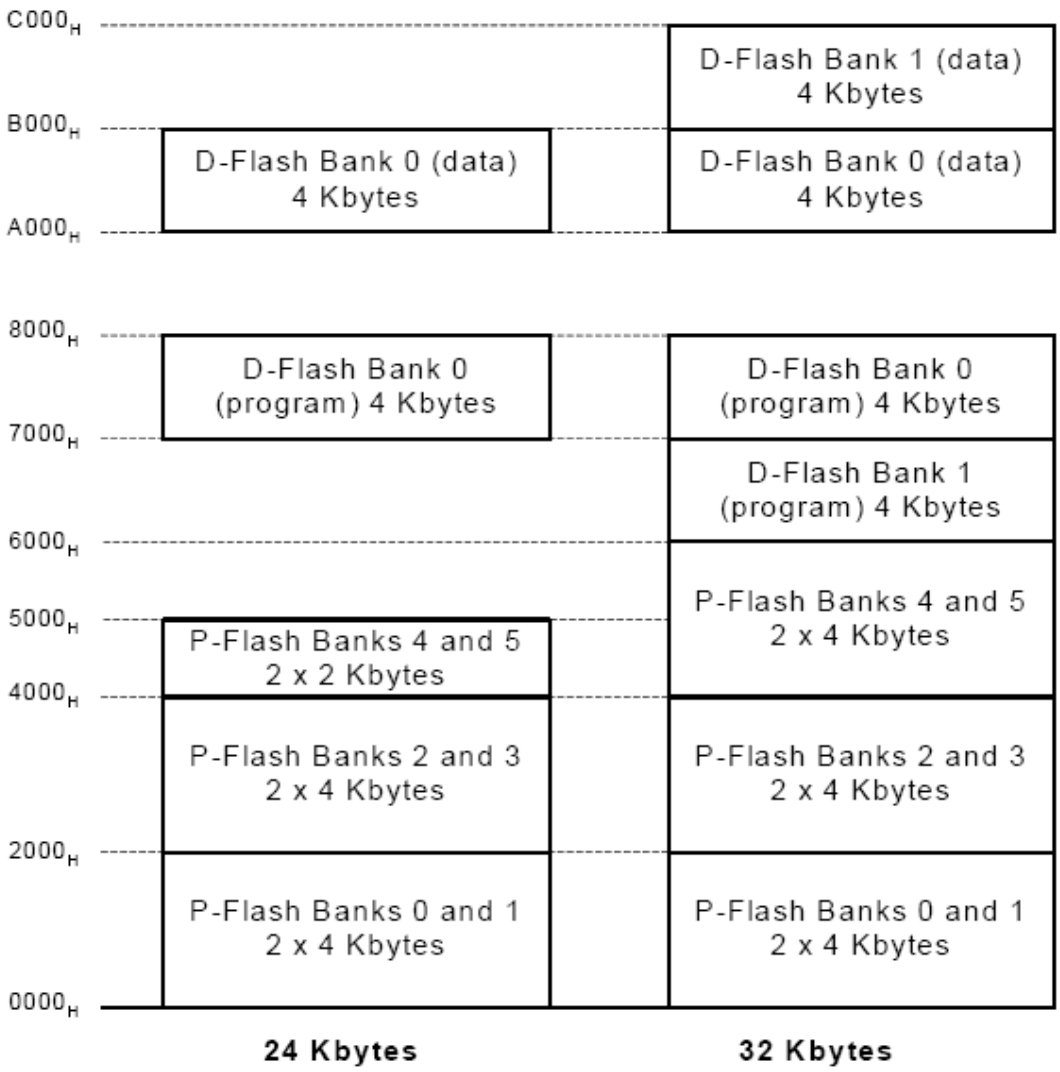
Project – Properties: C/C++ Build: MCU Selection Page (part 2 of 2):



Note: On-chip Memories:



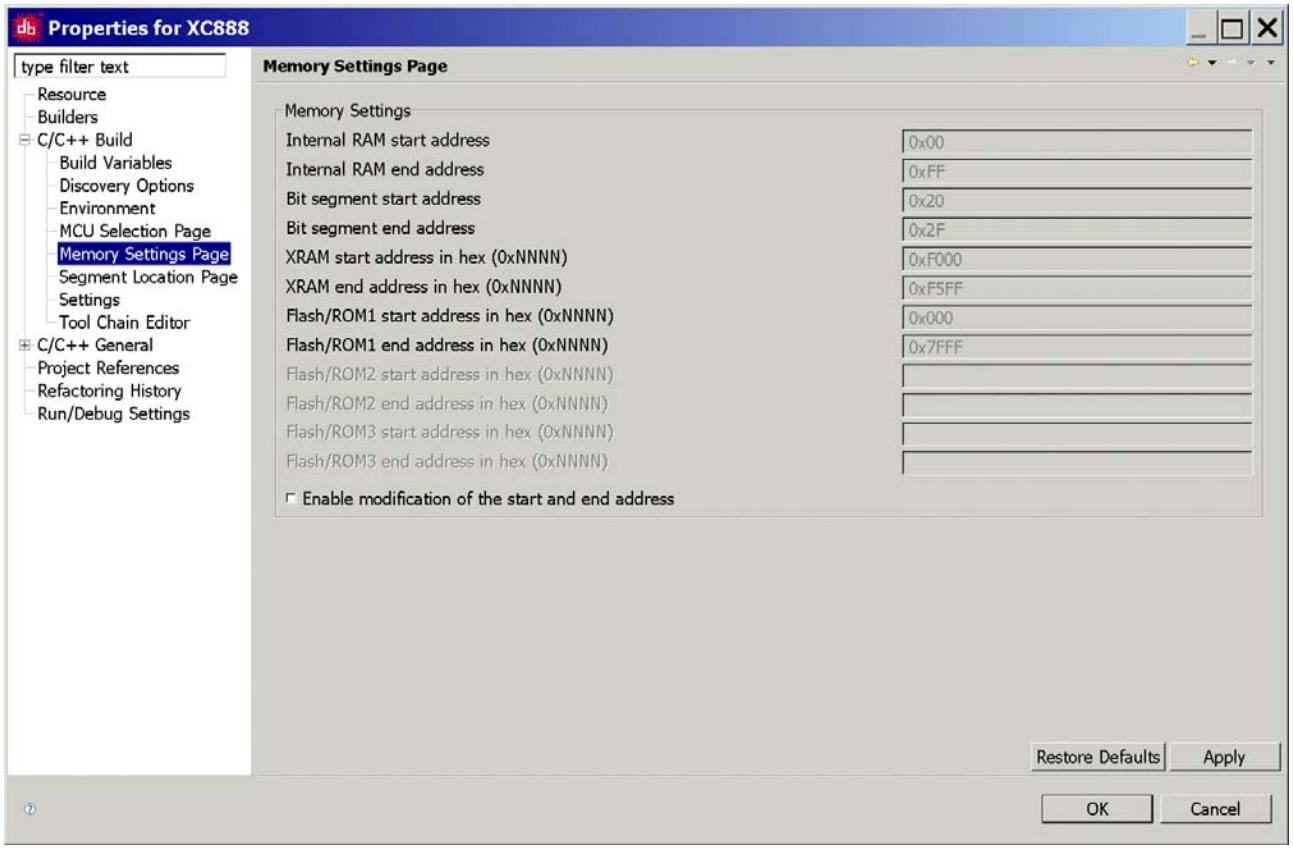
Additional information: Flash Memory Map (Source: User's Manual):



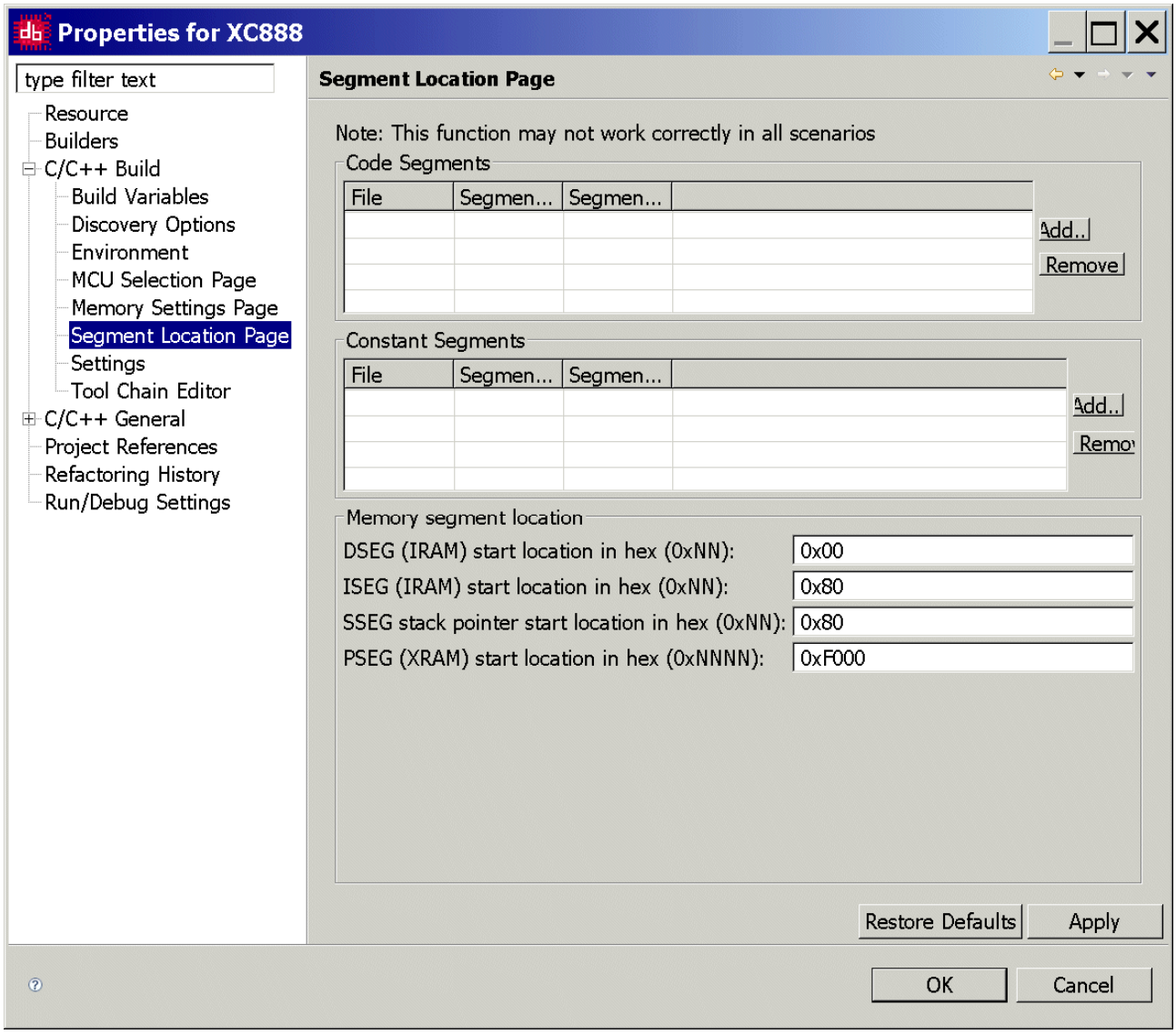
Note (Source: User's Manual):
The D-Flash bank(s) in the XC886/888 Flash devices are mapped to two program memory address spaces:
D-Flash Bank 0 is mapped to 7000_H – $7FFF_H$ and $A000_H$ – $AFFF_H$.
D-Flash Bank 1, which is only available in the 32-Kbyte Flash device, is mapped to 6000_H – $6FFF_H$ and $B000_H$ – $BFFF_H$.

In general, the lower address spaces (6000_H – $6FFF_H$ and 7000_H – $7FFF_H$) should be used for D-Flash bank(s) contents that are intended to be used as program code. Alternatively, the higher address spaces ($A000_H$ – $AFFF_H$ and $B000_H$ – $BFFF_H$) should be used for D-Flash bank(s) contents that are intended to be used as data.

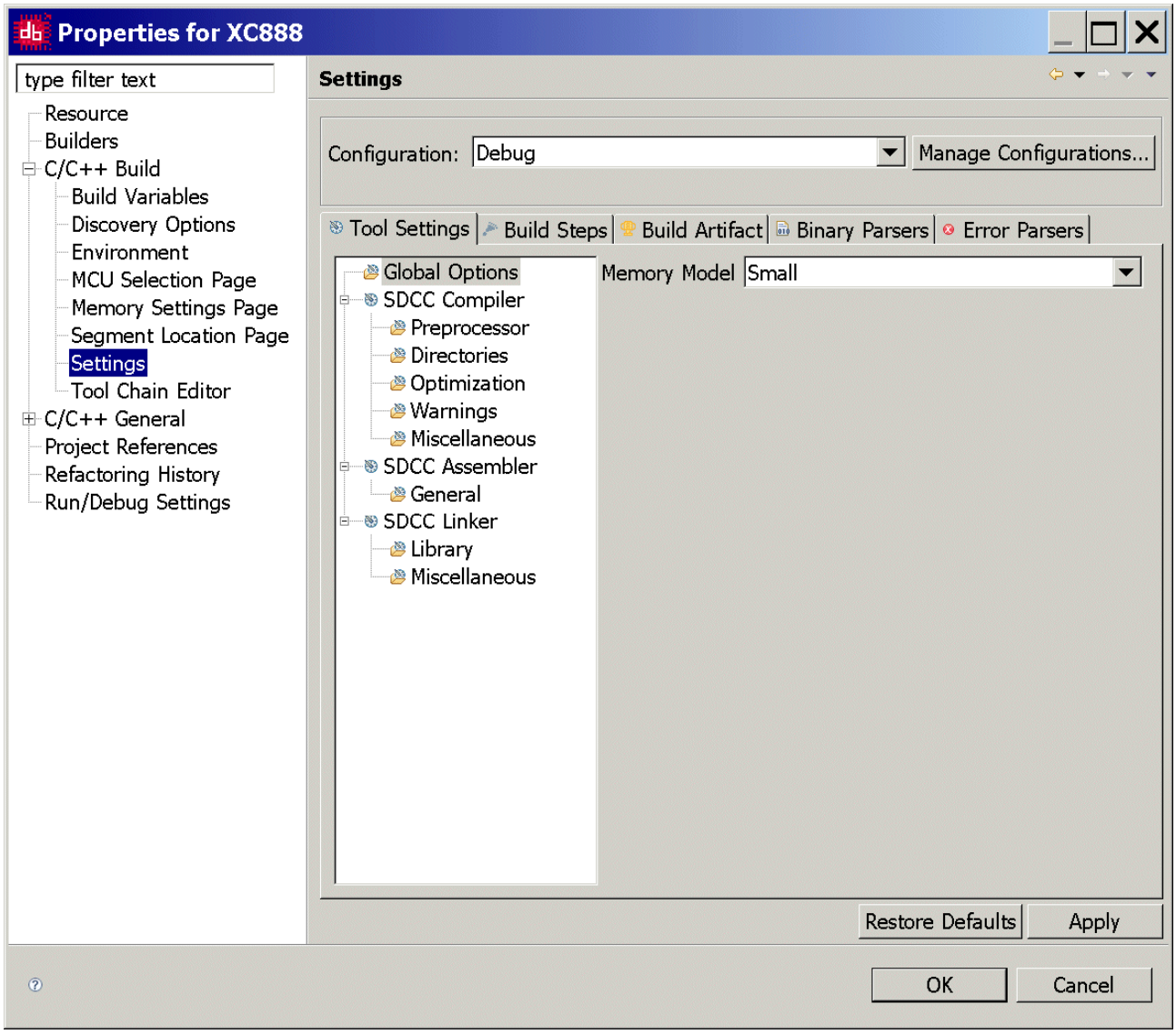
Project – Properties: C/C++ Build: Memory Setting Page:



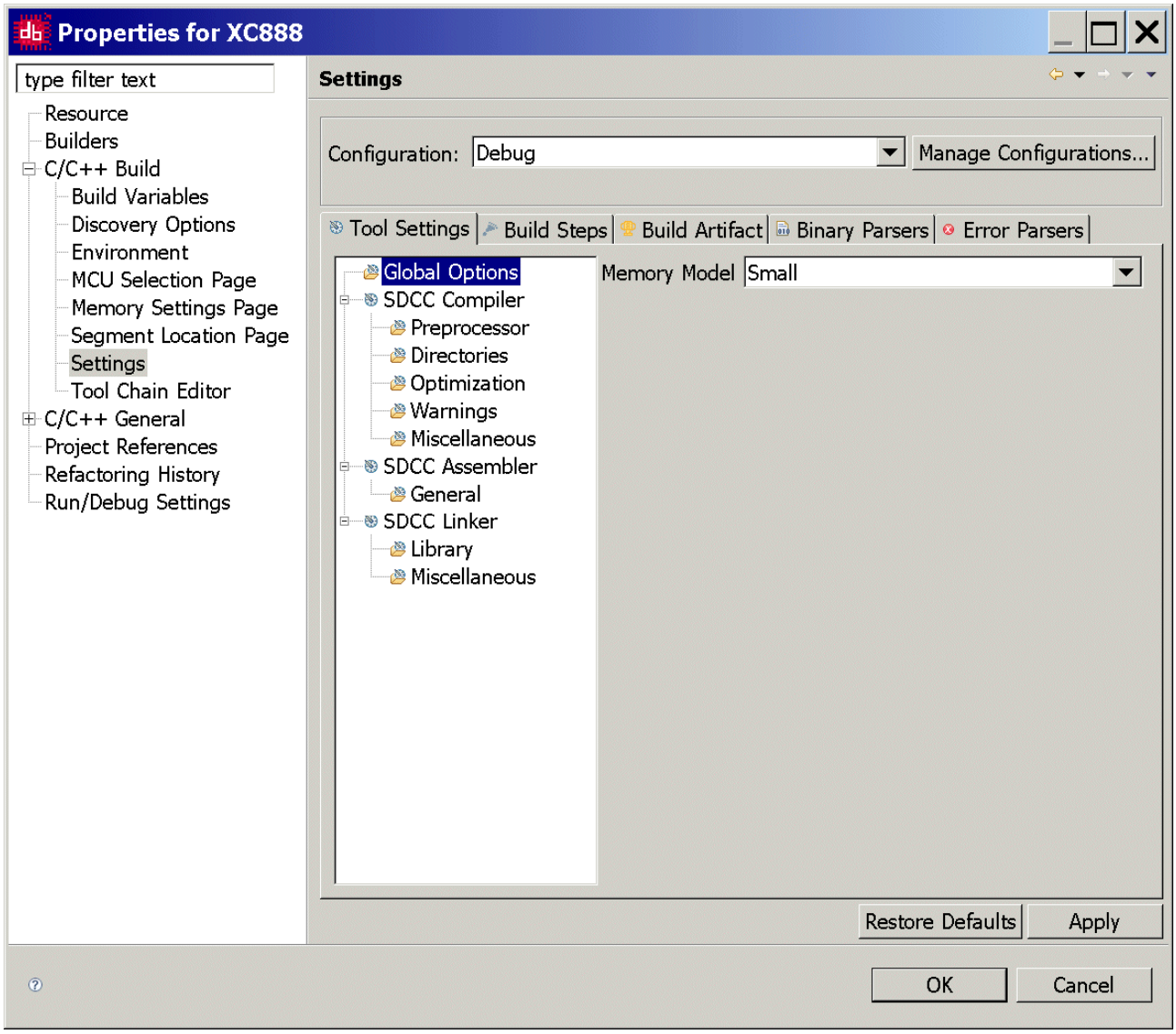
Project – Properties: C/C++ Build: Segment Location Page:



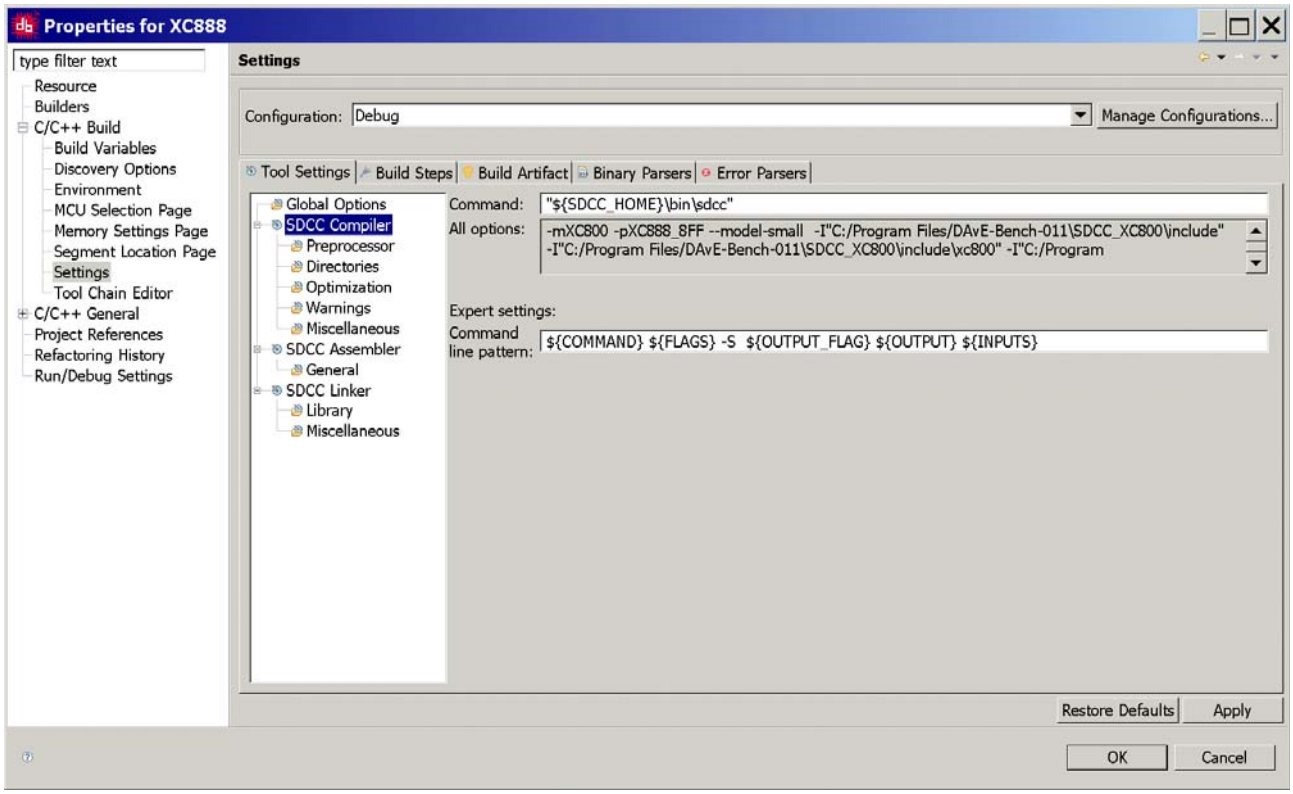
Project – Properties: C/C++ Build: Settings:



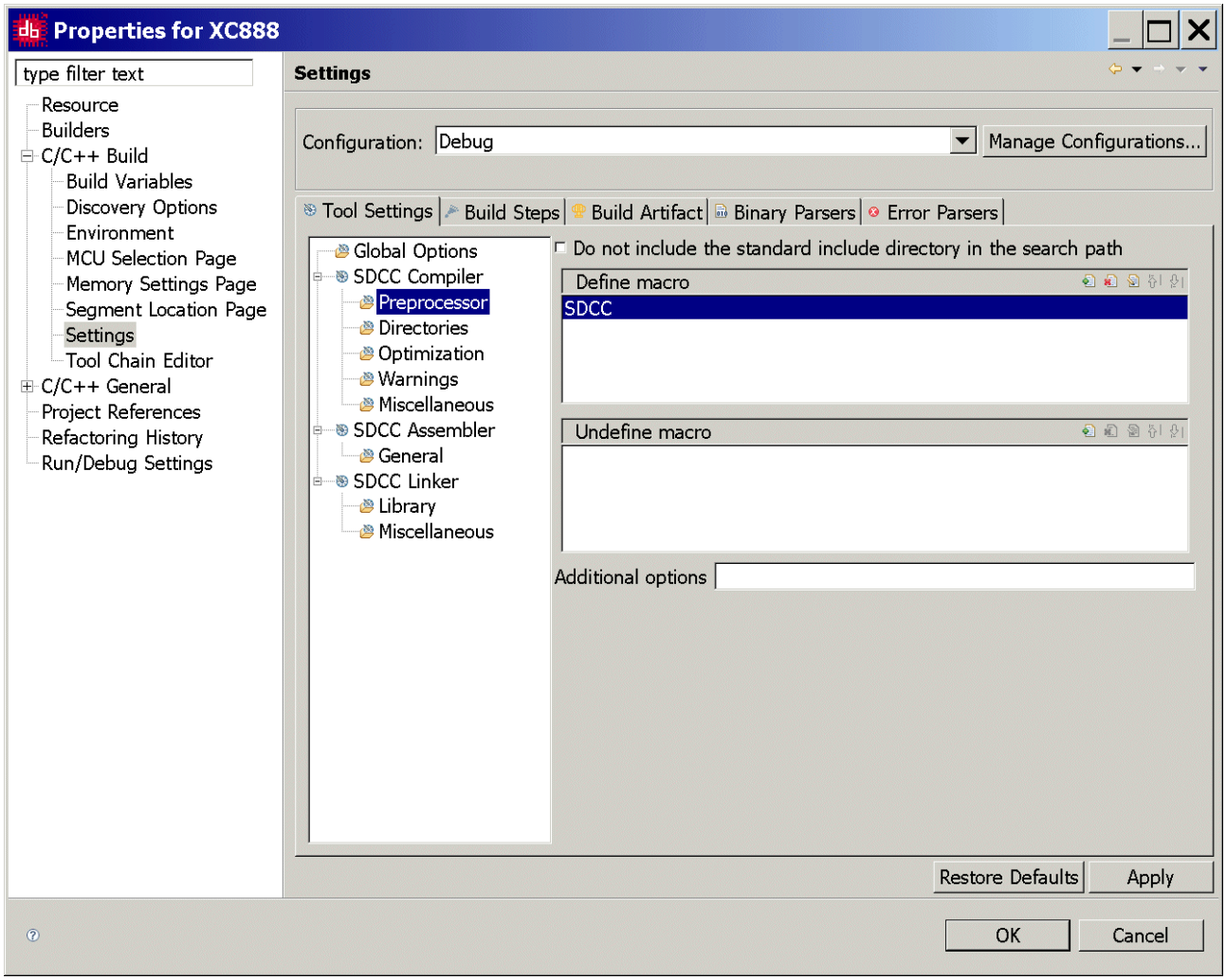
Project – Properties: C/C++ Build: Settings: Tool Settings: Global Options:



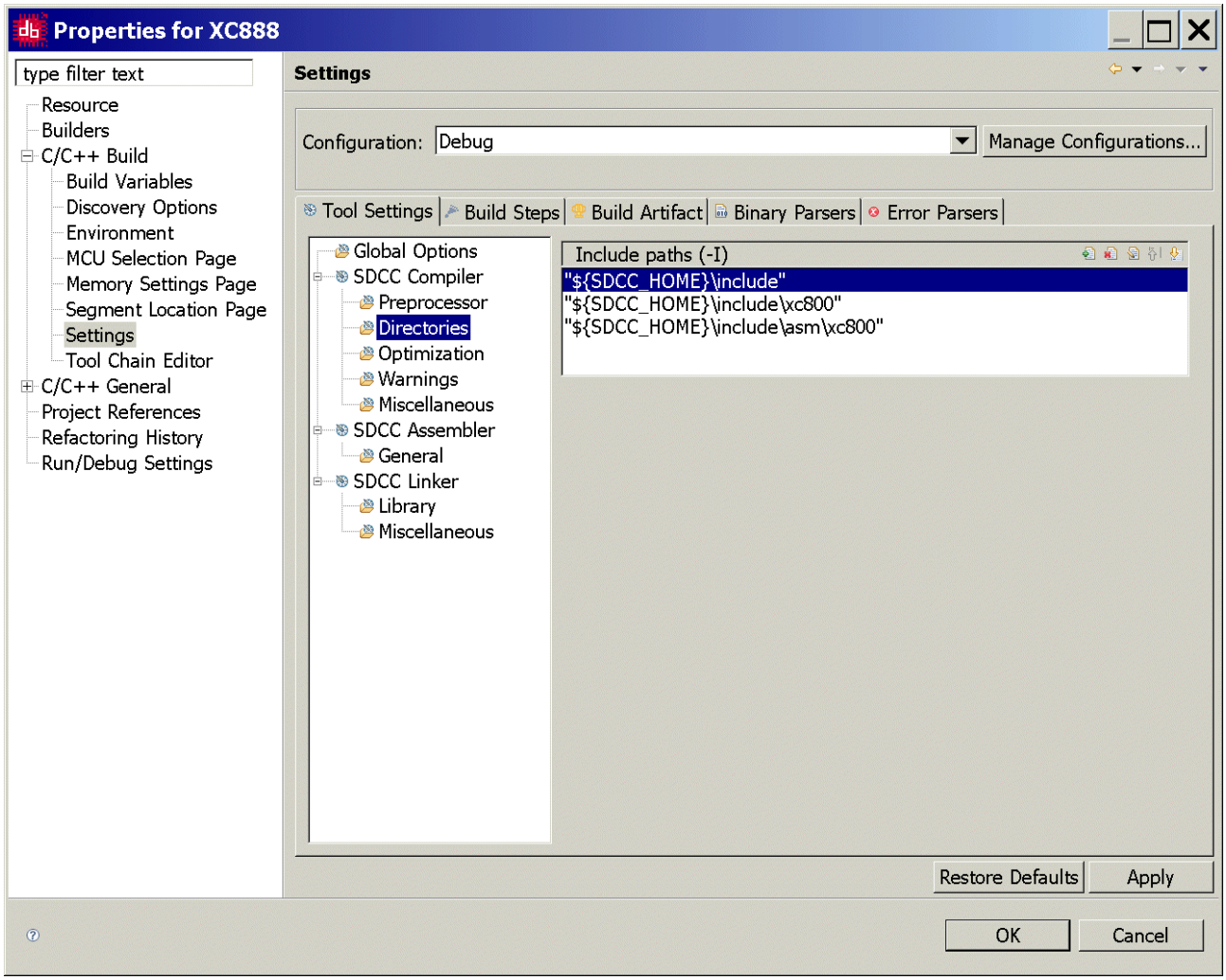
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Compiler:



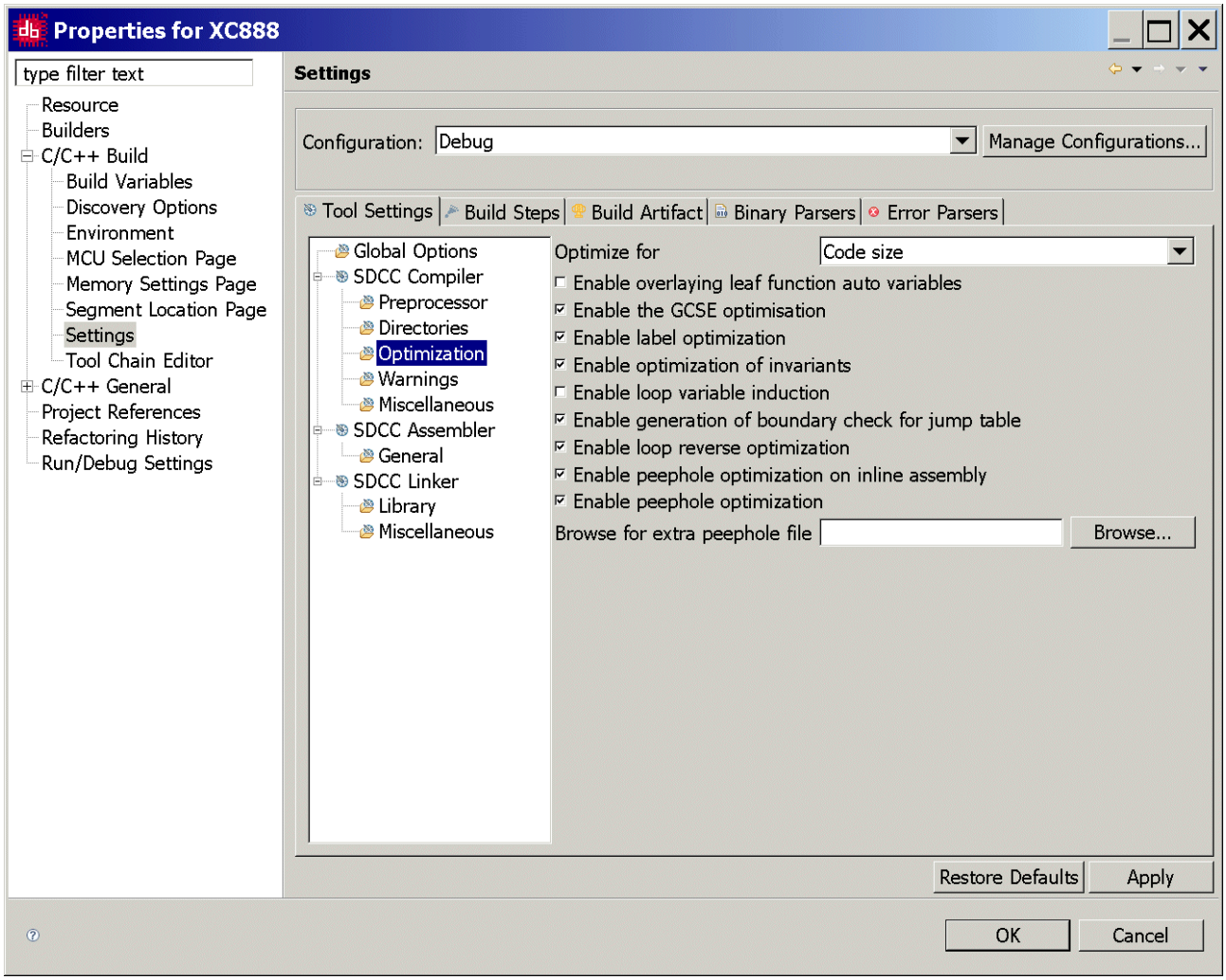
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Compiler: Preprocessor:



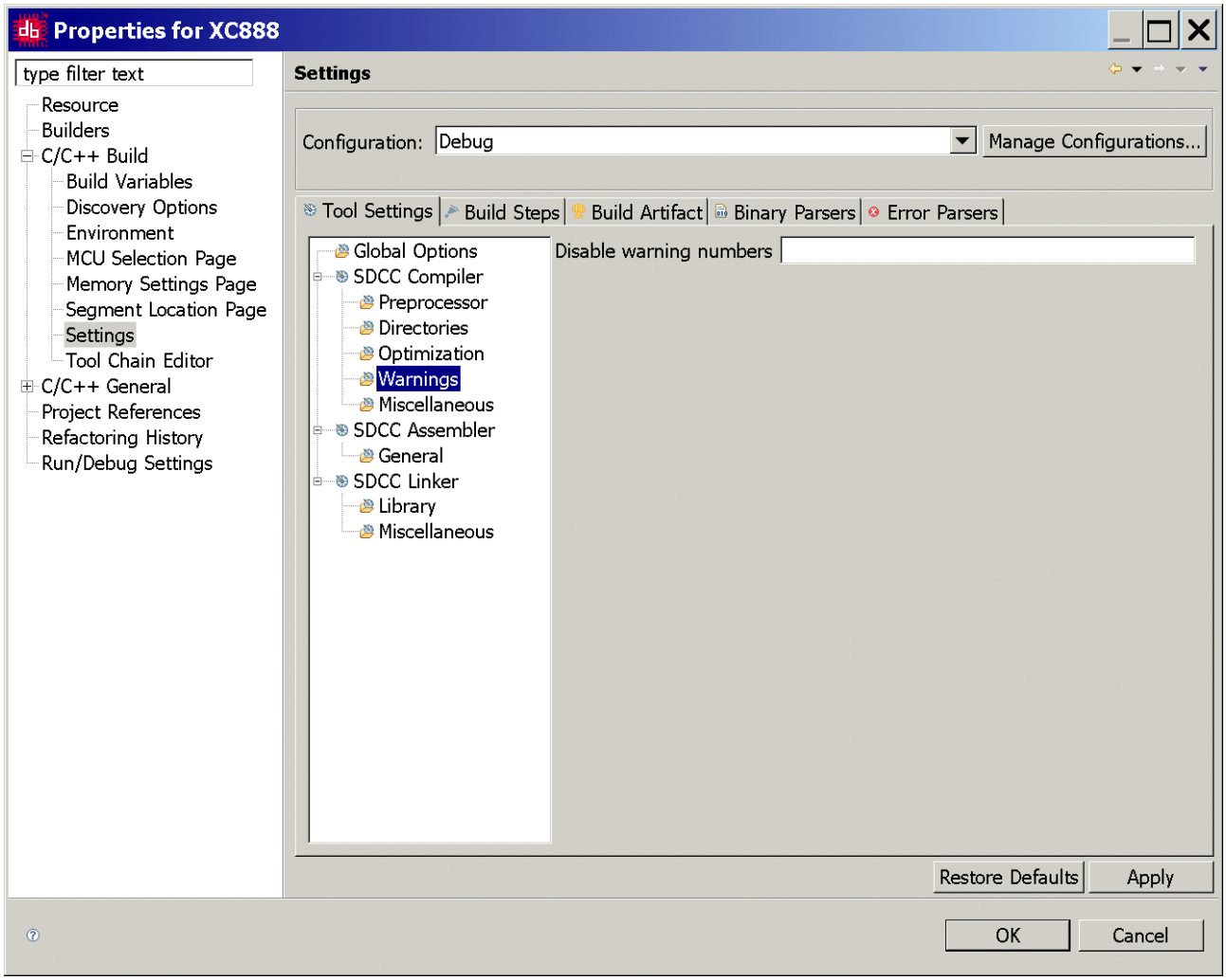
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Compiler: Directories



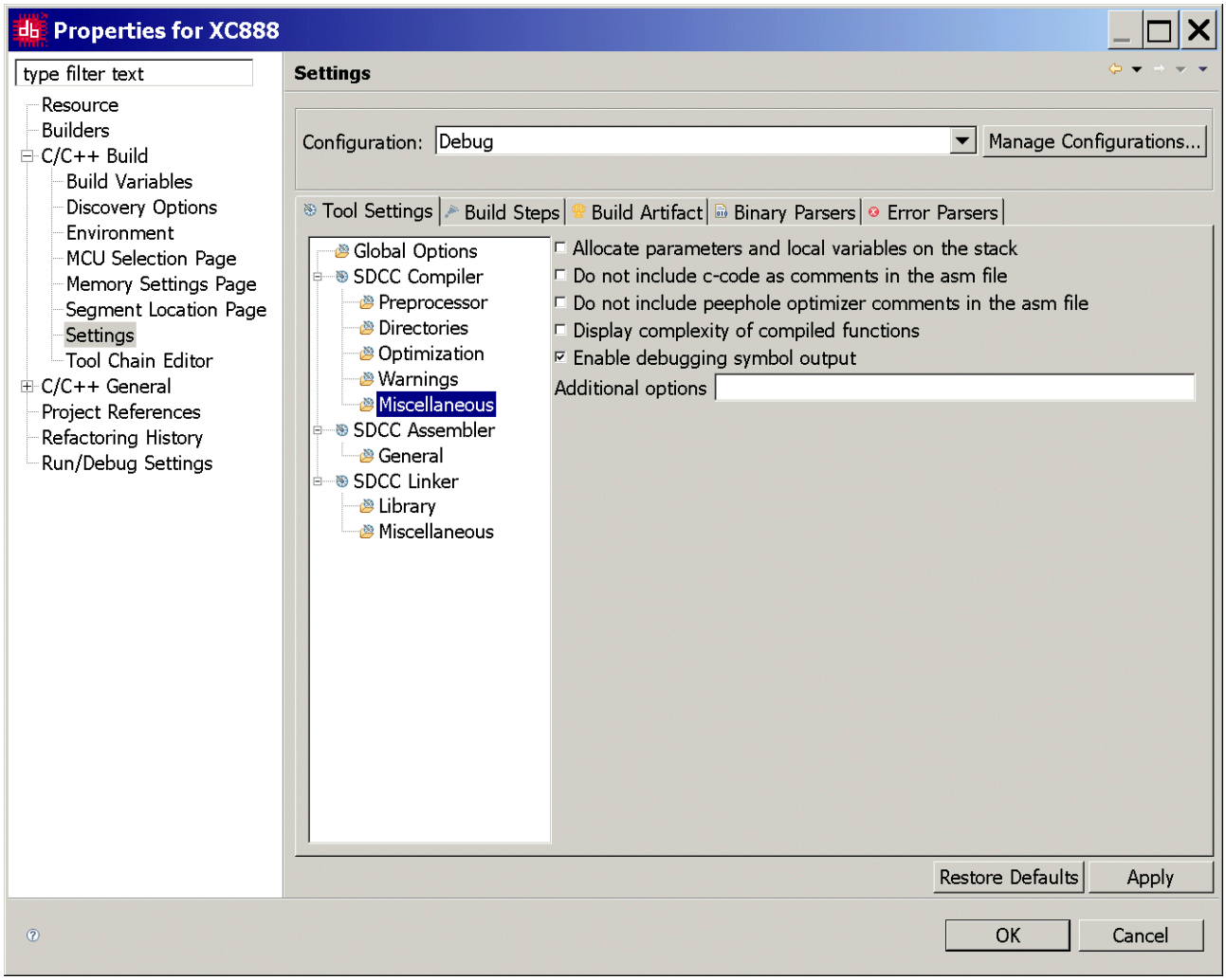
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Compiler: Optimization:



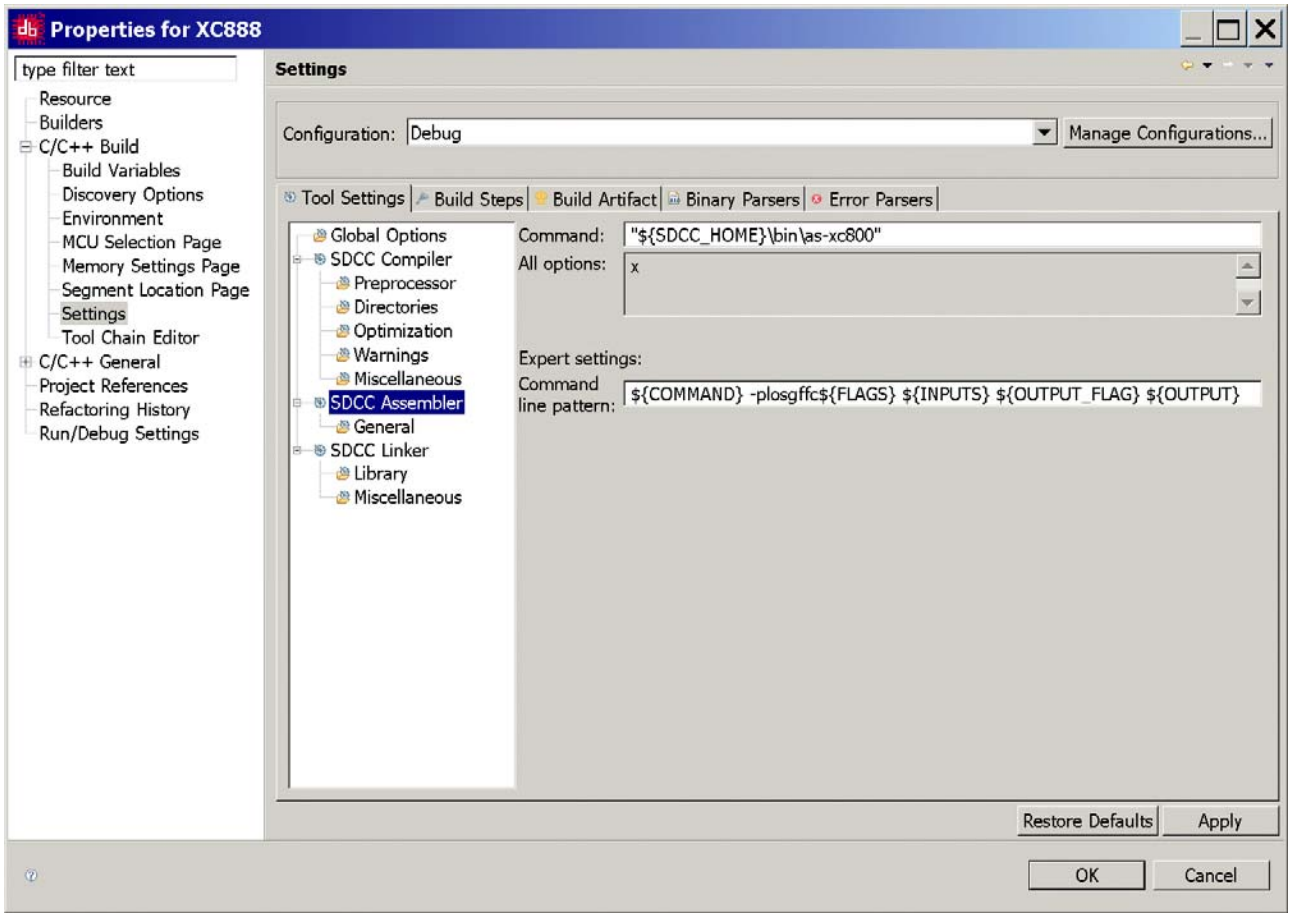
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Compiler: Warnings:



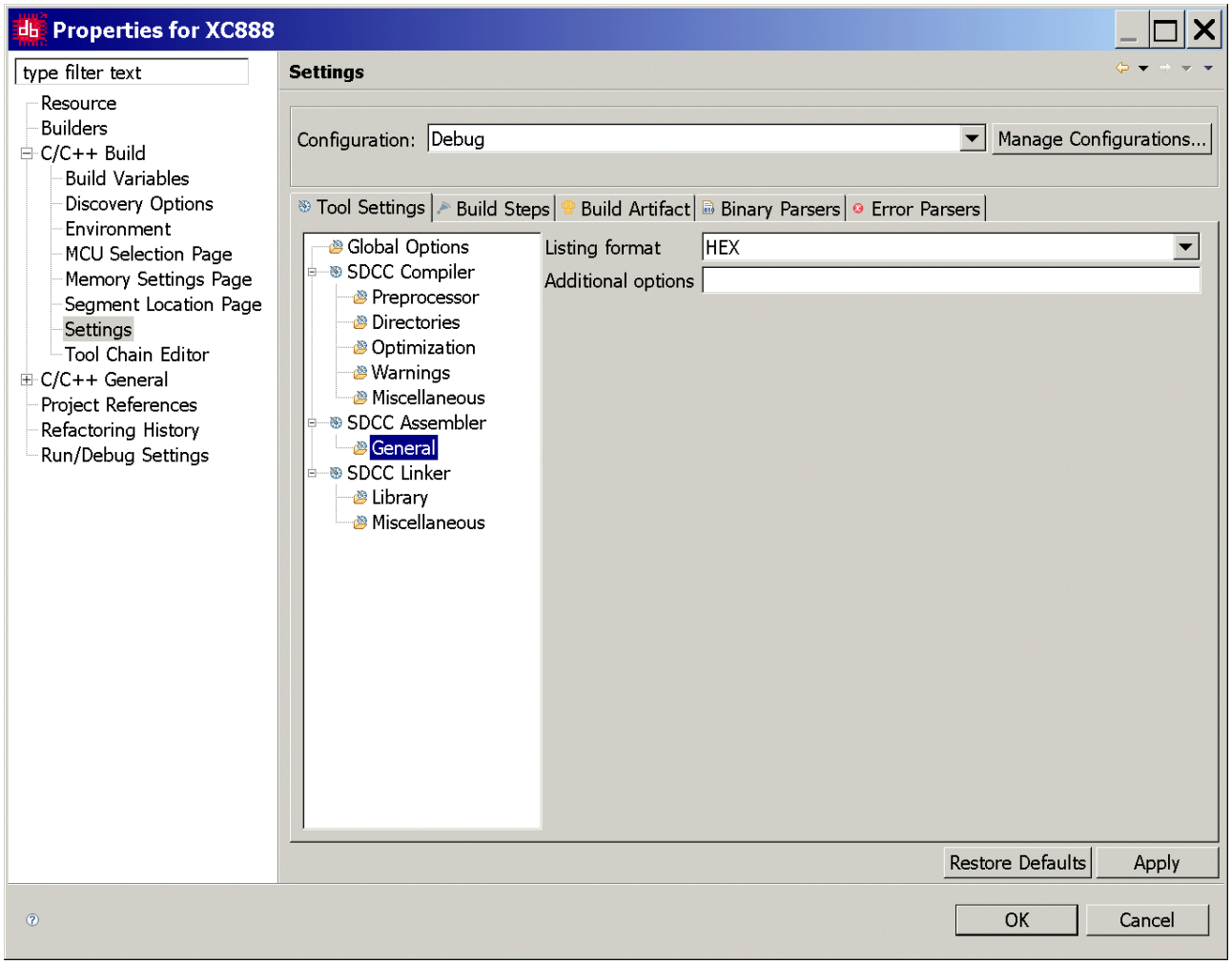
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Compiler: Miscellaneous:



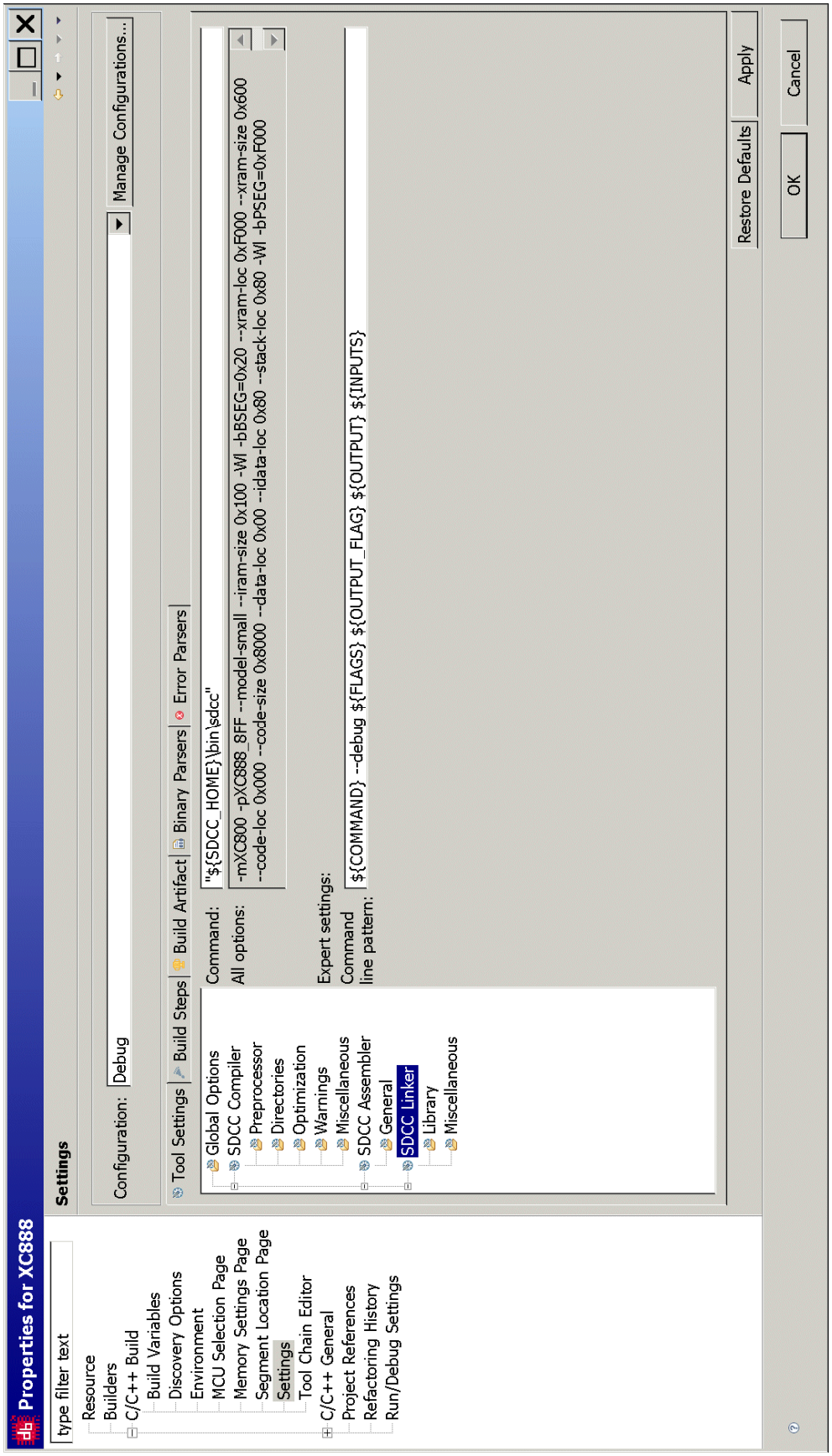
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Assembler:



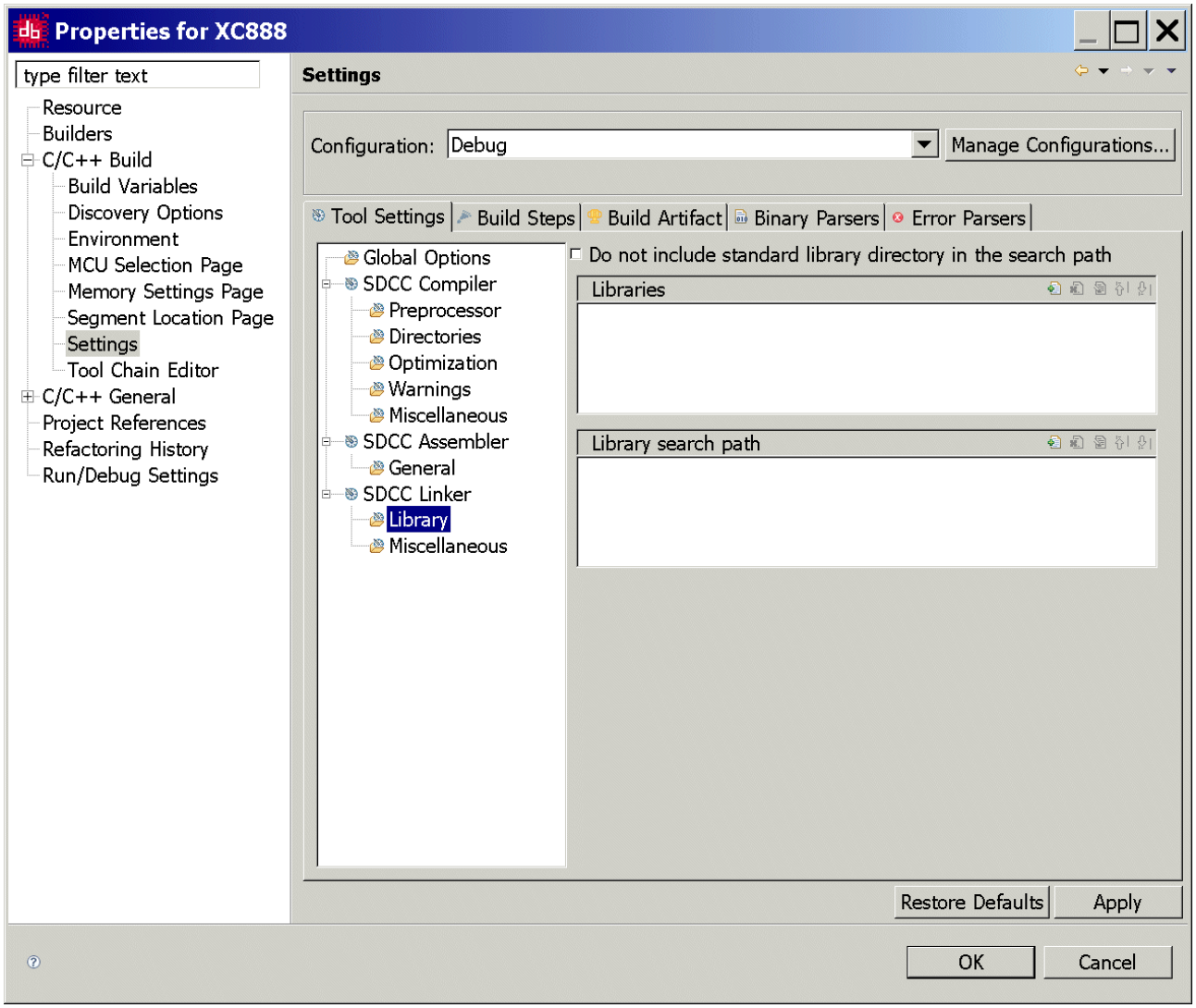
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Assembler: General:



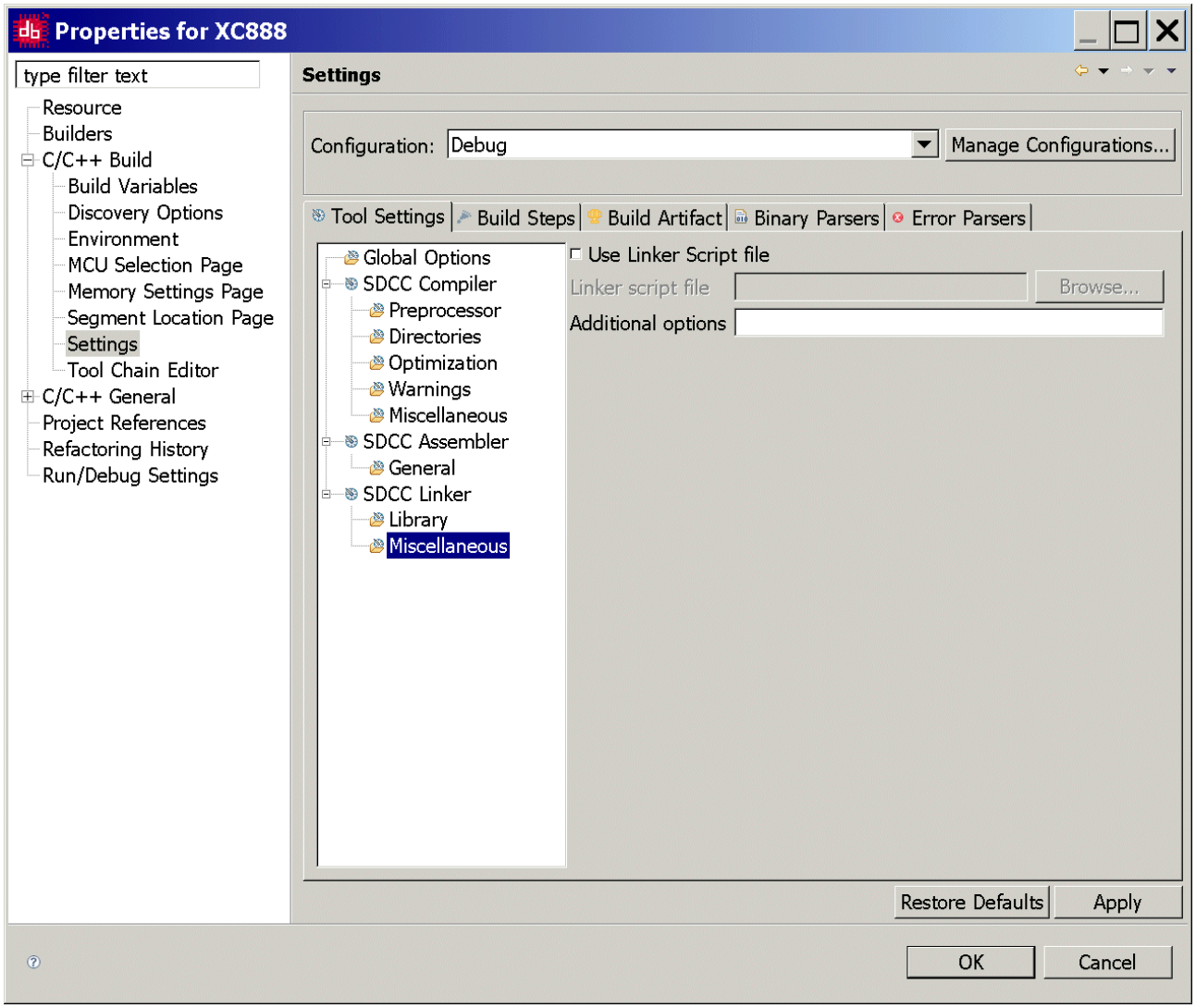
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Linker:



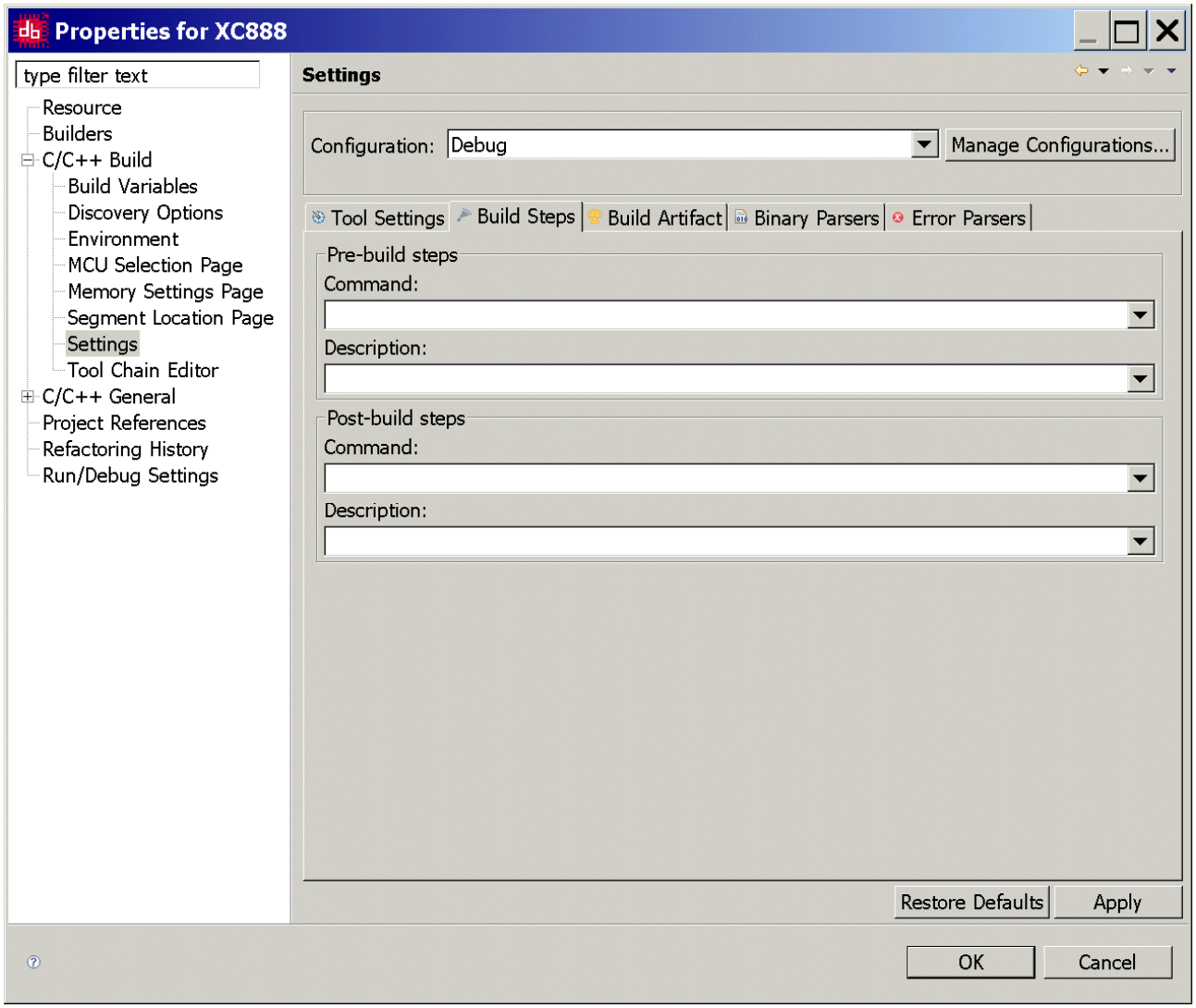
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Linker: Library:



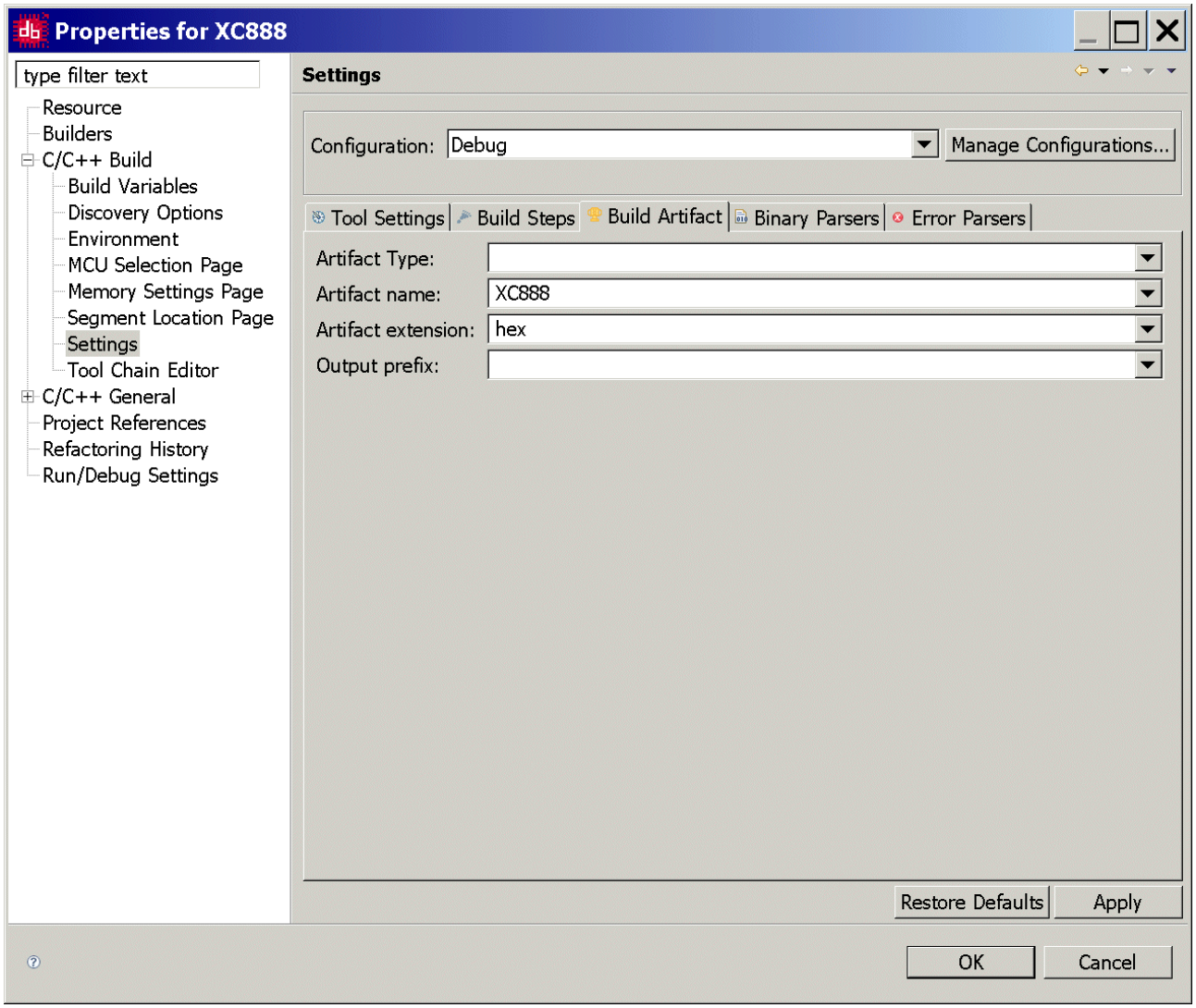
Project – Properties: C/C++ Build: Settings: Tool Settings: SDCC Linker: Miscellaneous:



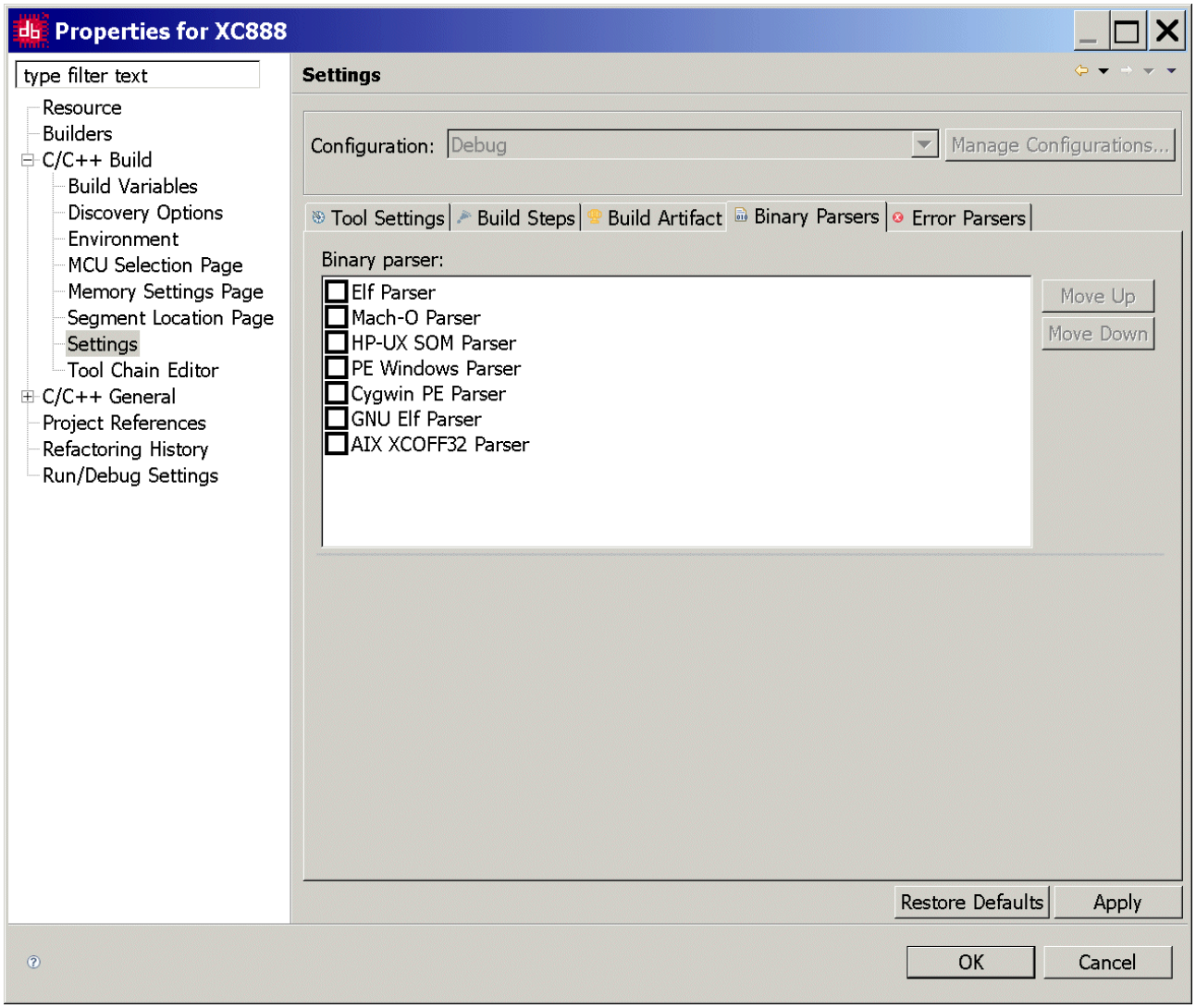
Project – Properties: C/C++ Build: Settings: Build Steps:



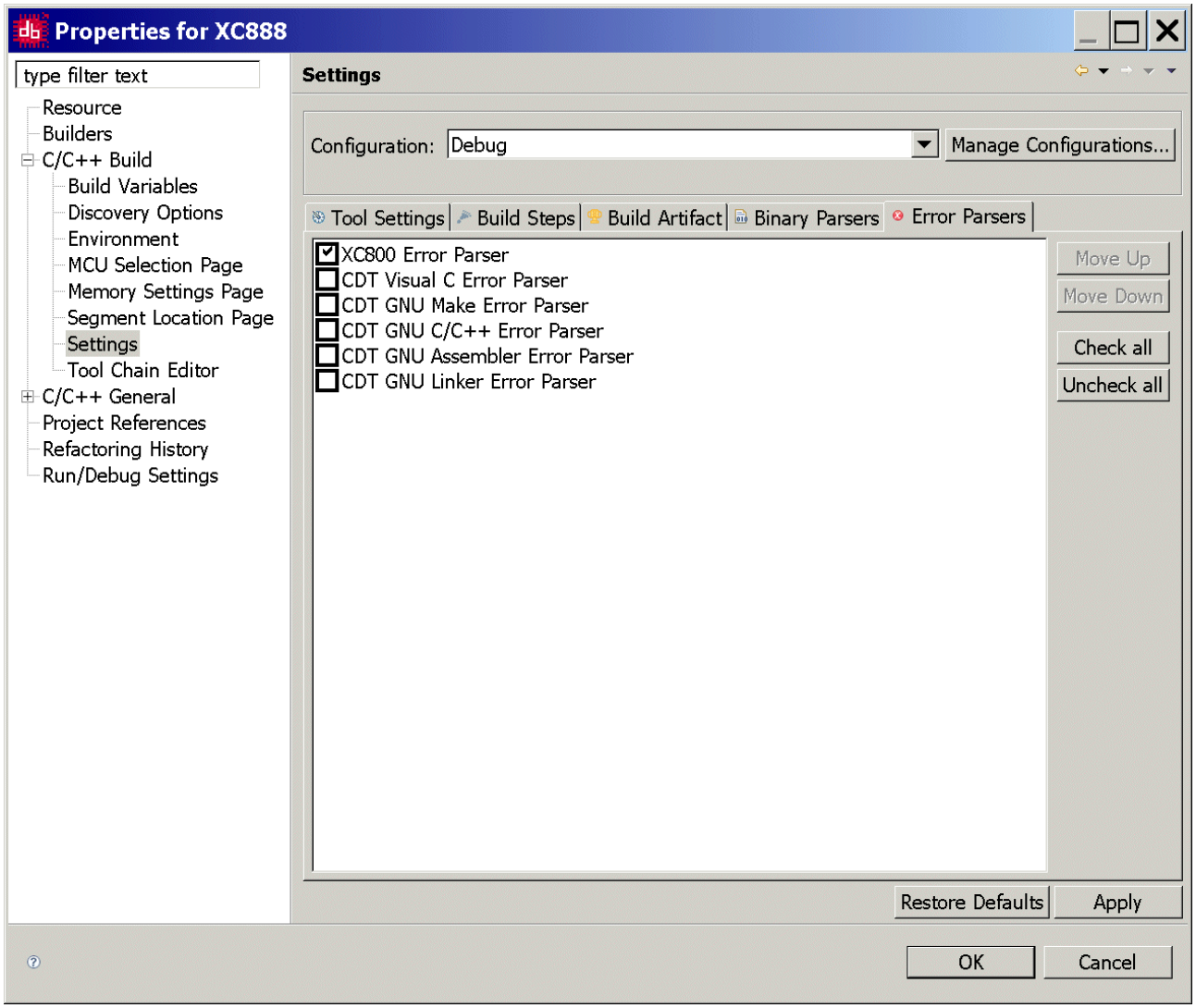
Project – Properties: C/C++ Build: Settings: Build Artifact:



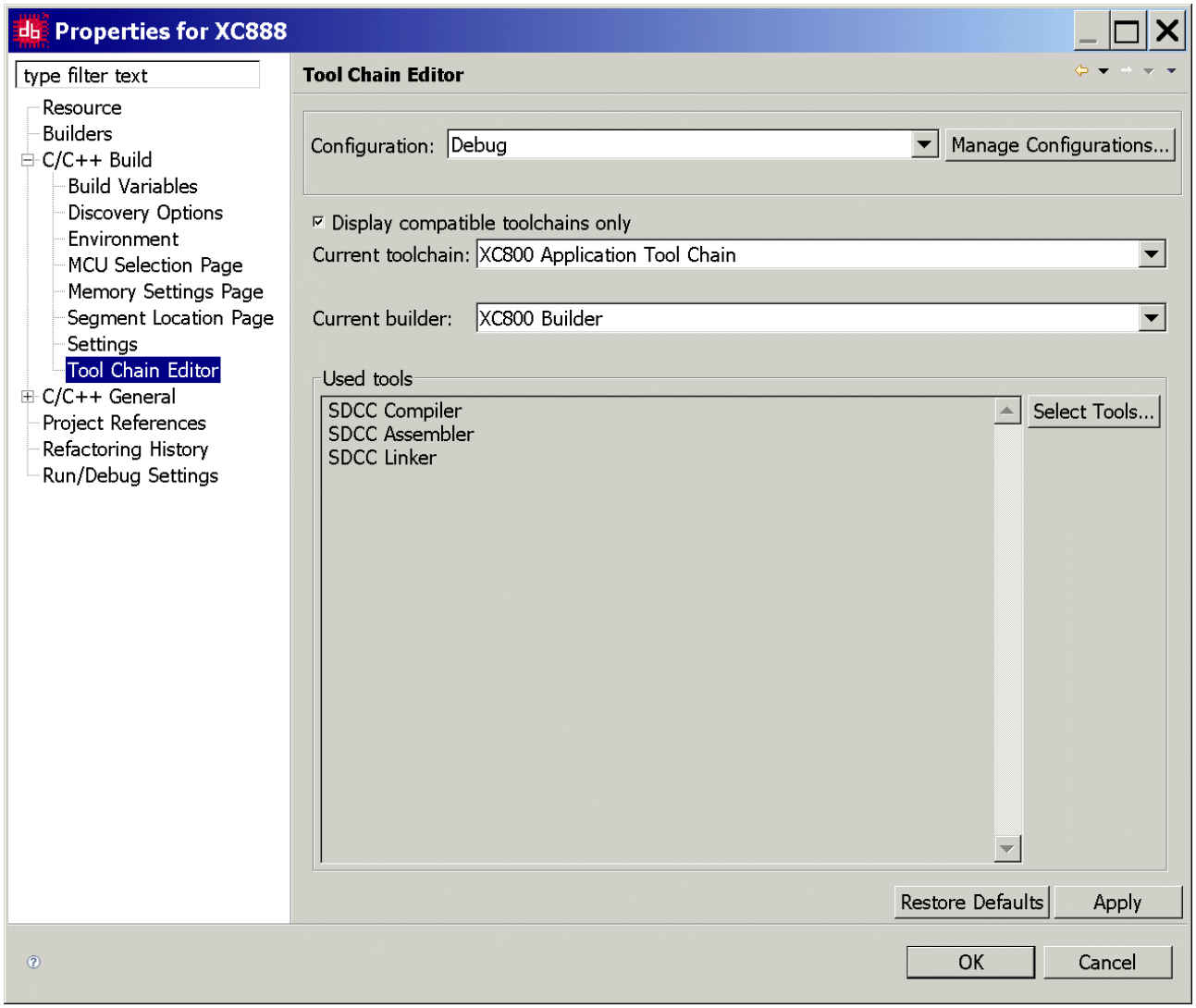
Project – Properties: C/C++ Build: Settings: Binary Parsers:



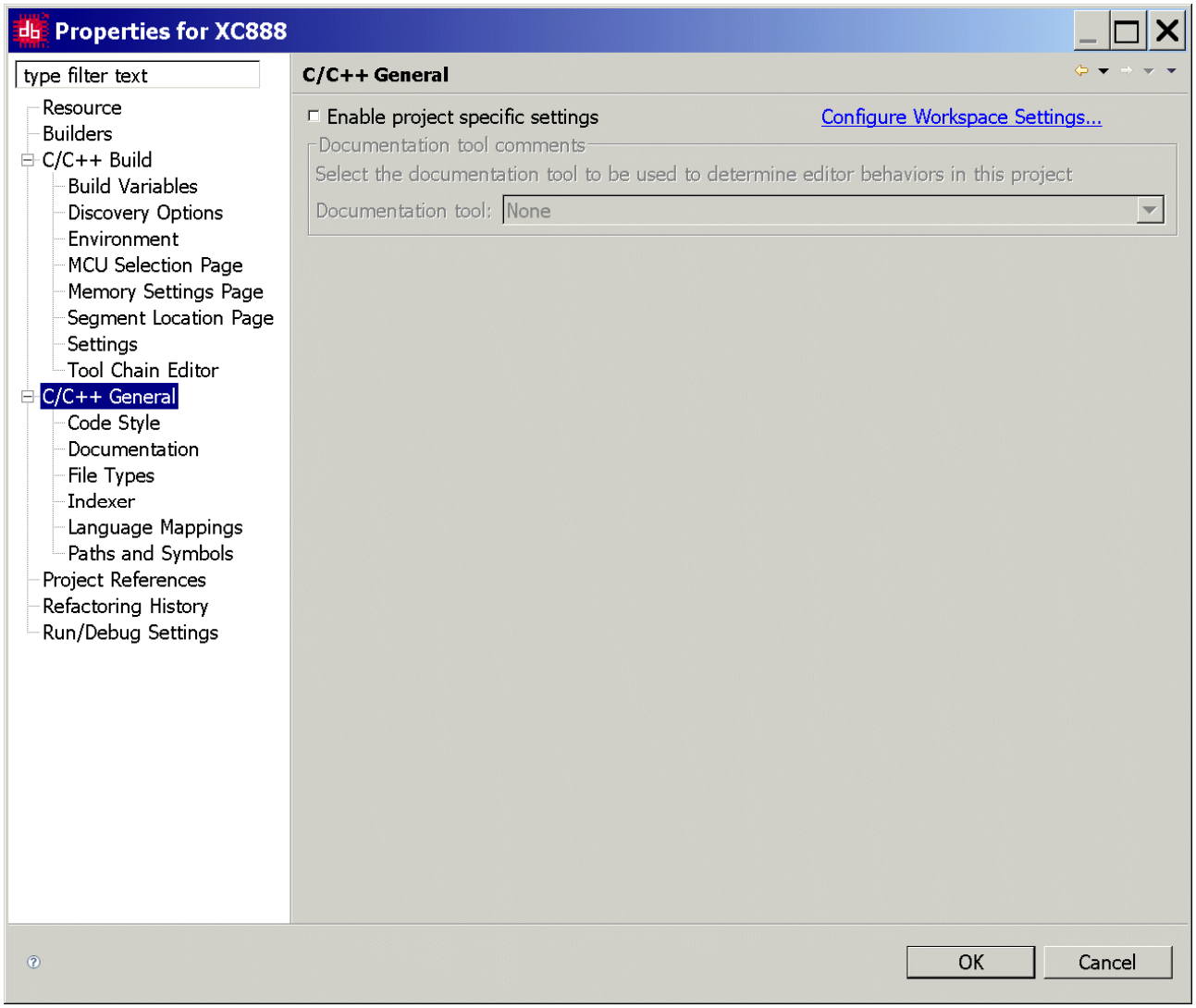
Project – Properties: C/C++ Build: Settings: Error Parsers:



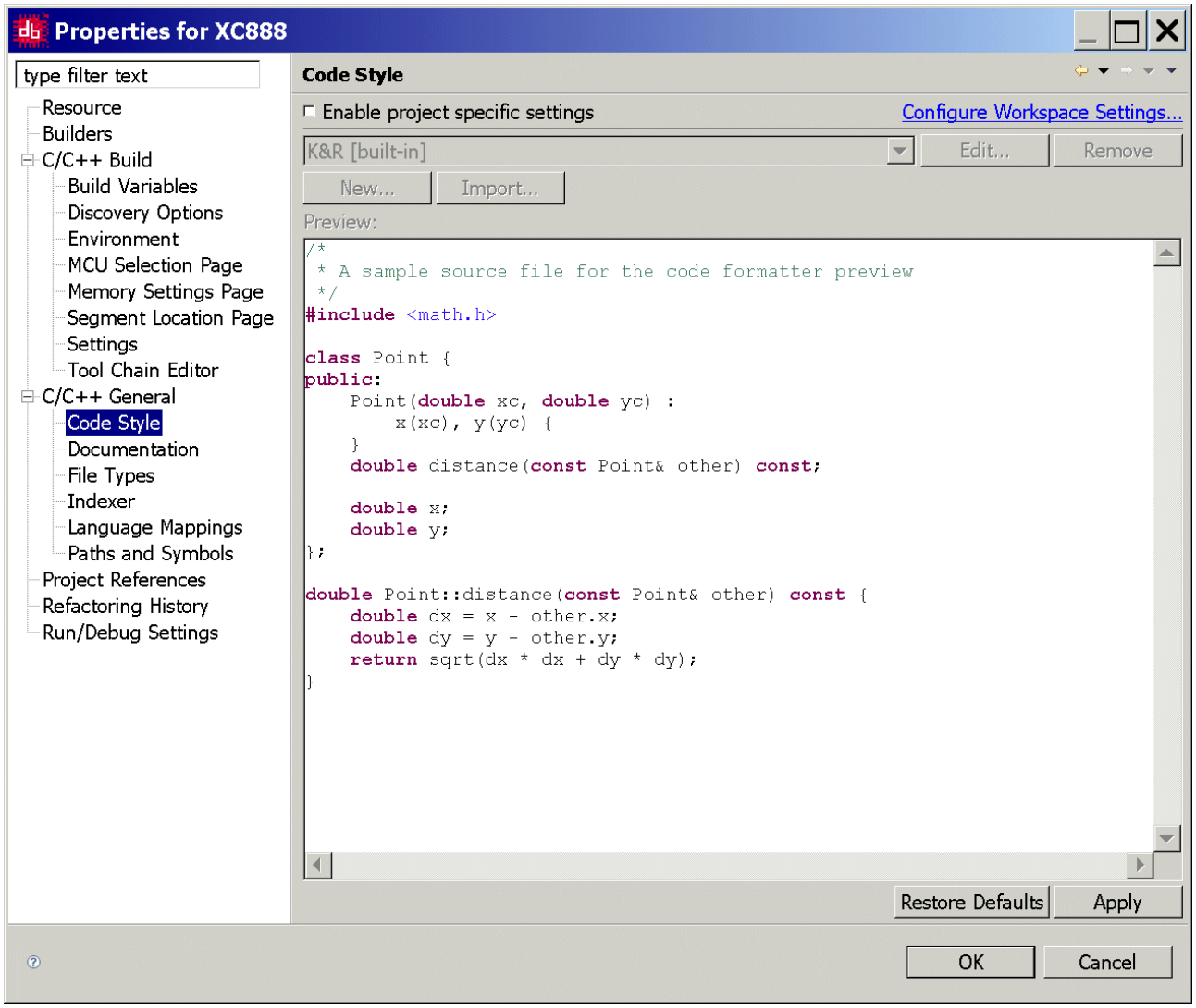
Project – Properties: C/C++ Build: Tool Chain Editor:



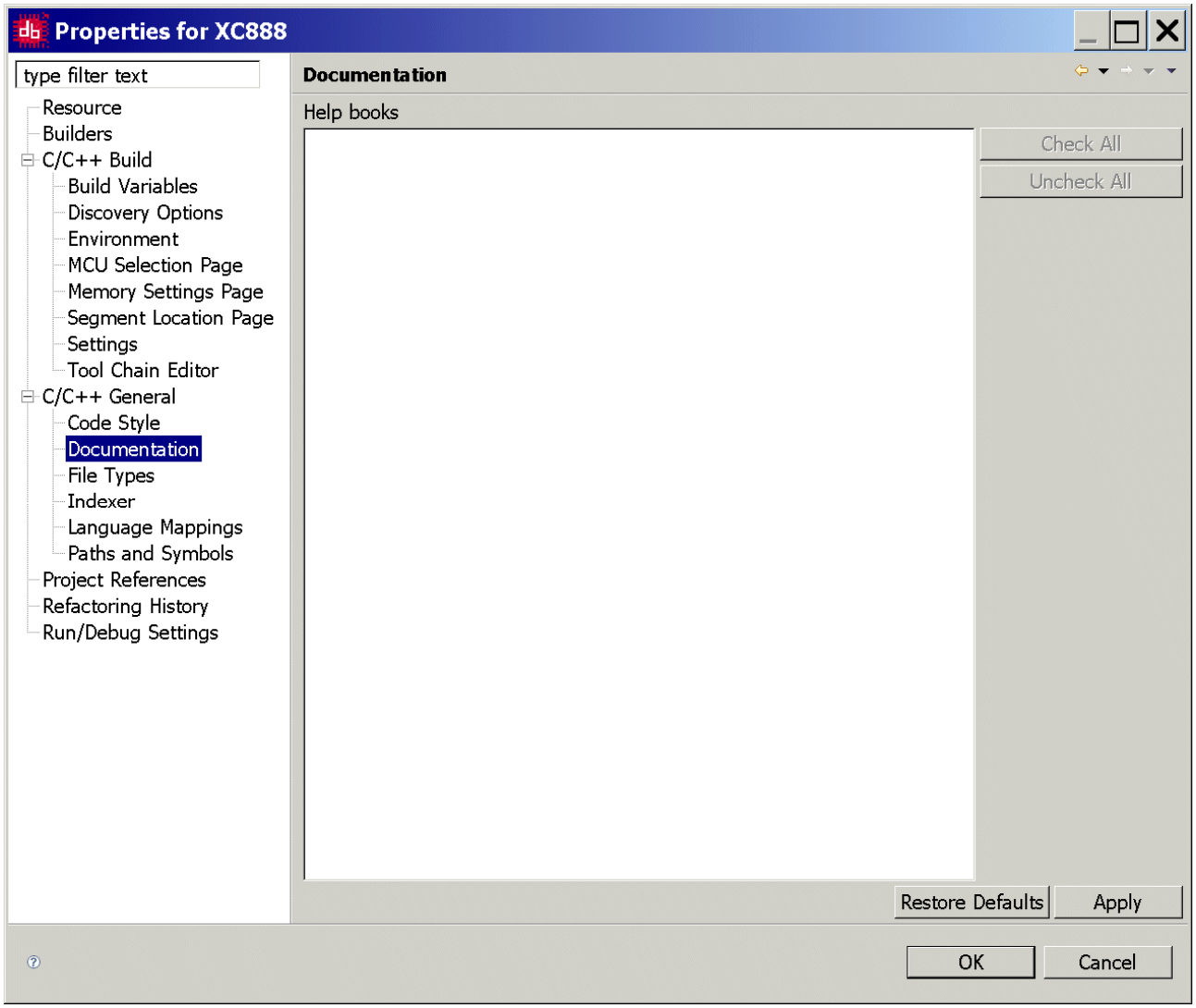
Project – Properties: C/C++ General:



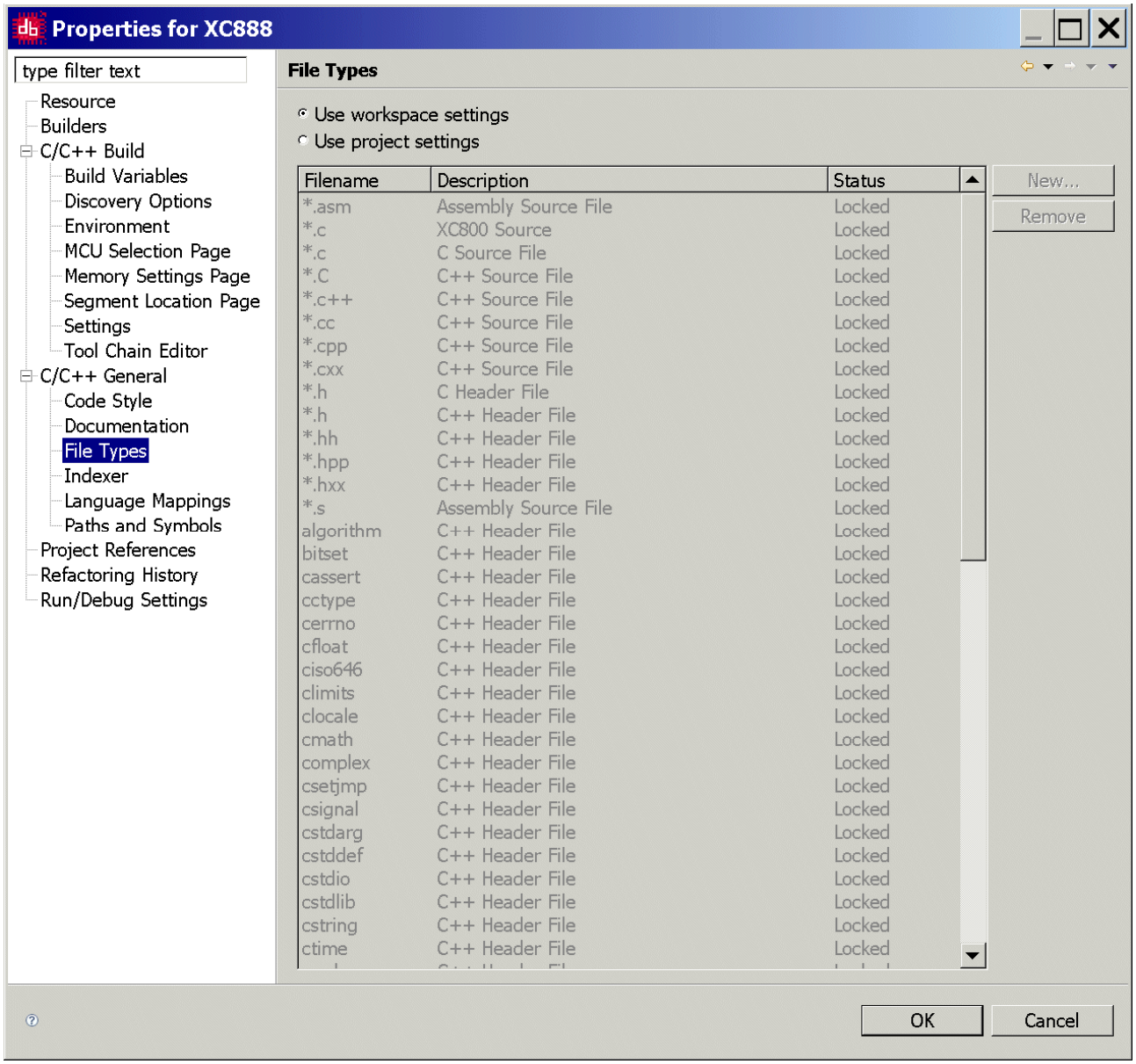
Project – Properties: C/C++ General: Code Style:



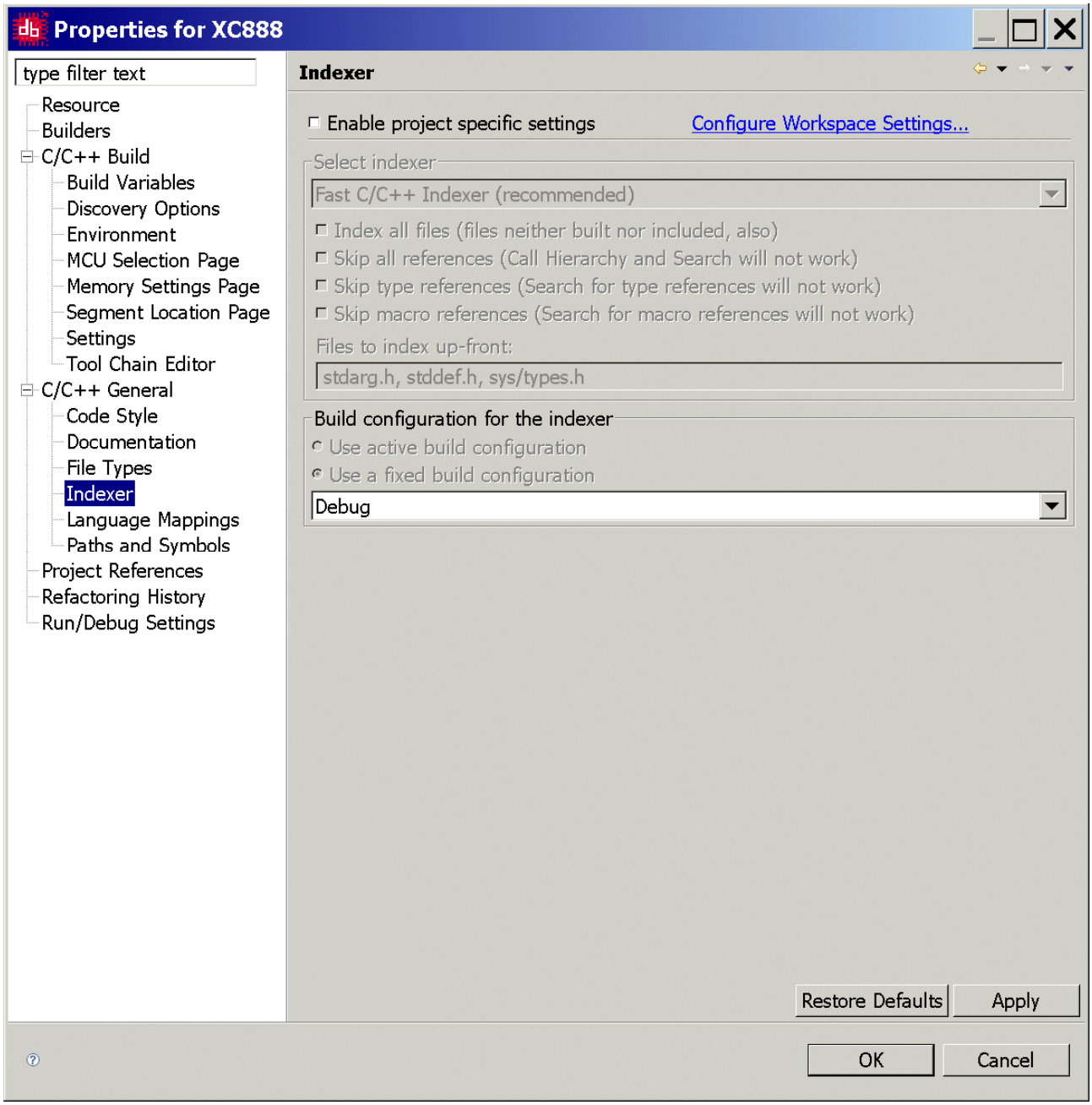
Project – Properties: C/C++ General: Documentation:



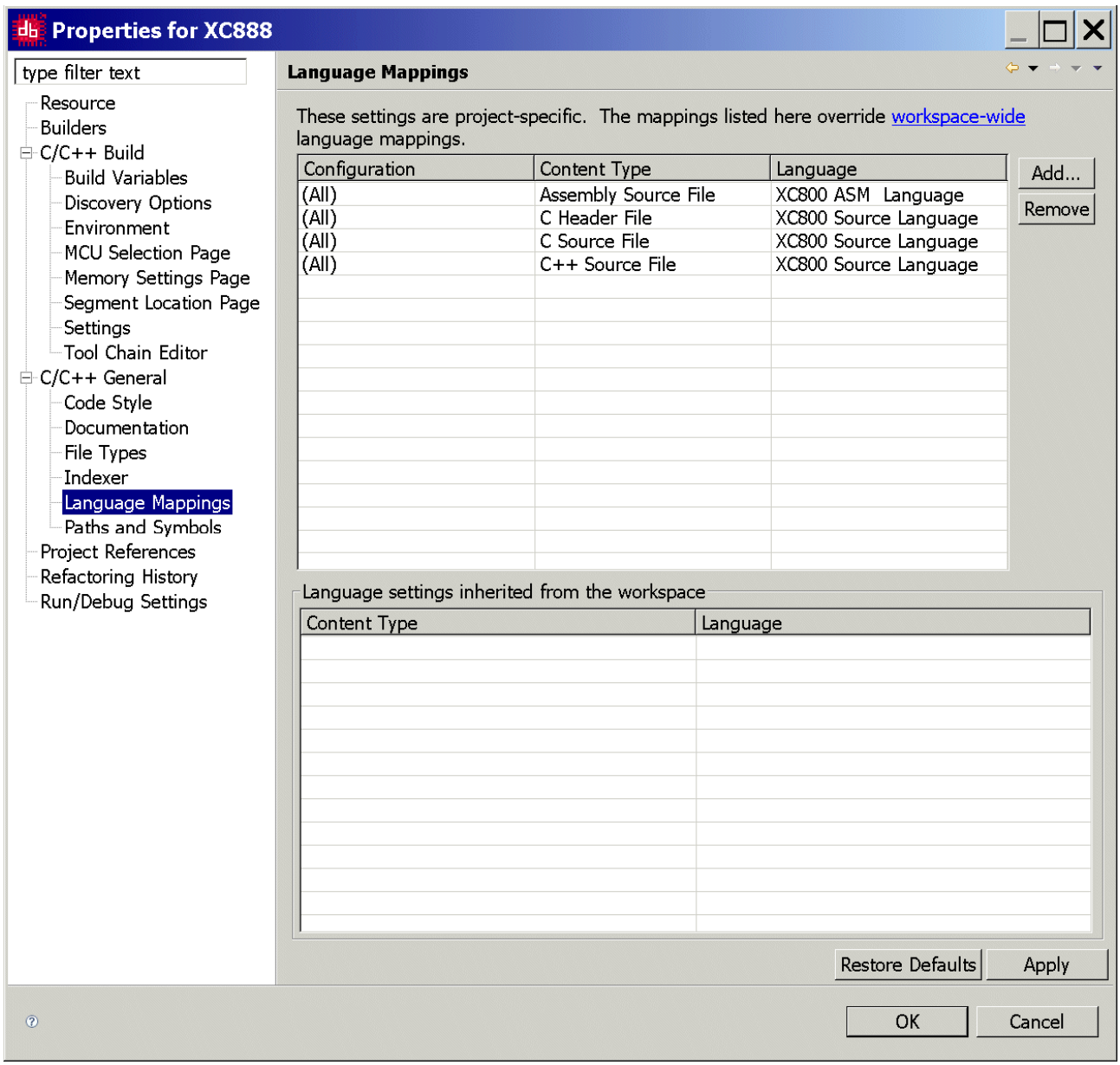
Project – Properties: C/C++ General: File Types:



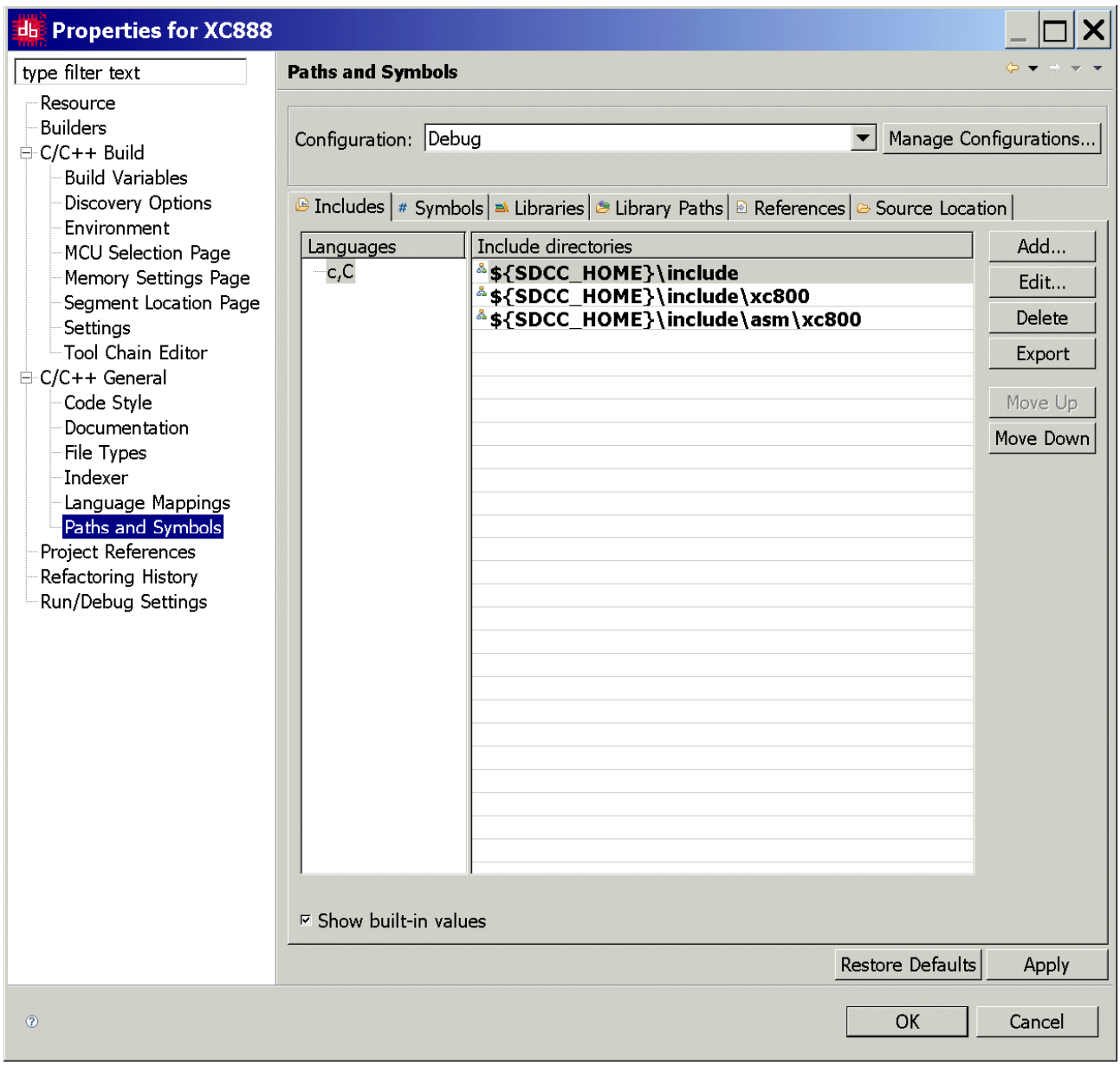
Project – Properties: C/C++ General: Indexer:



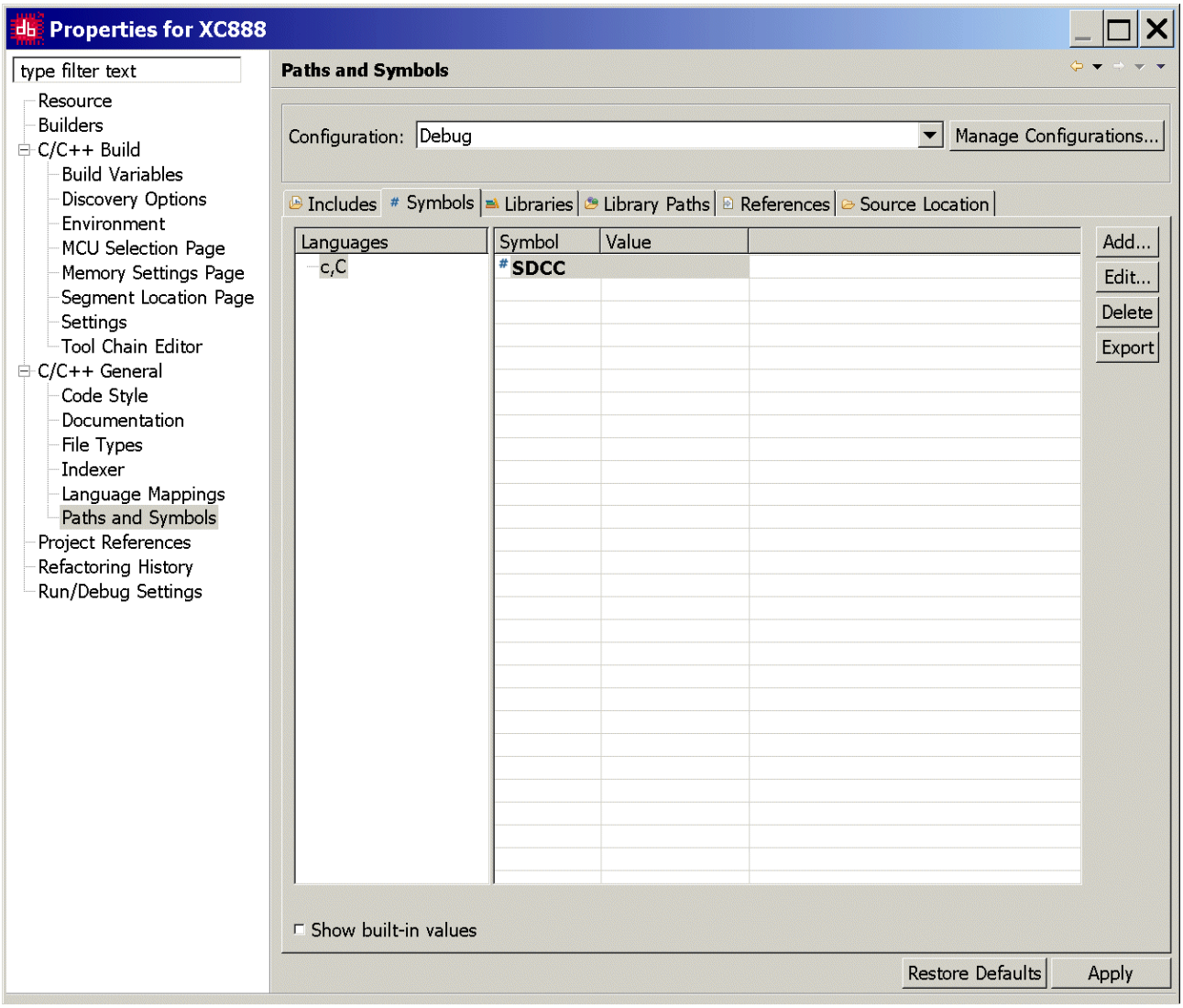
Project – Properties: C/C++ General: Language Mappings:



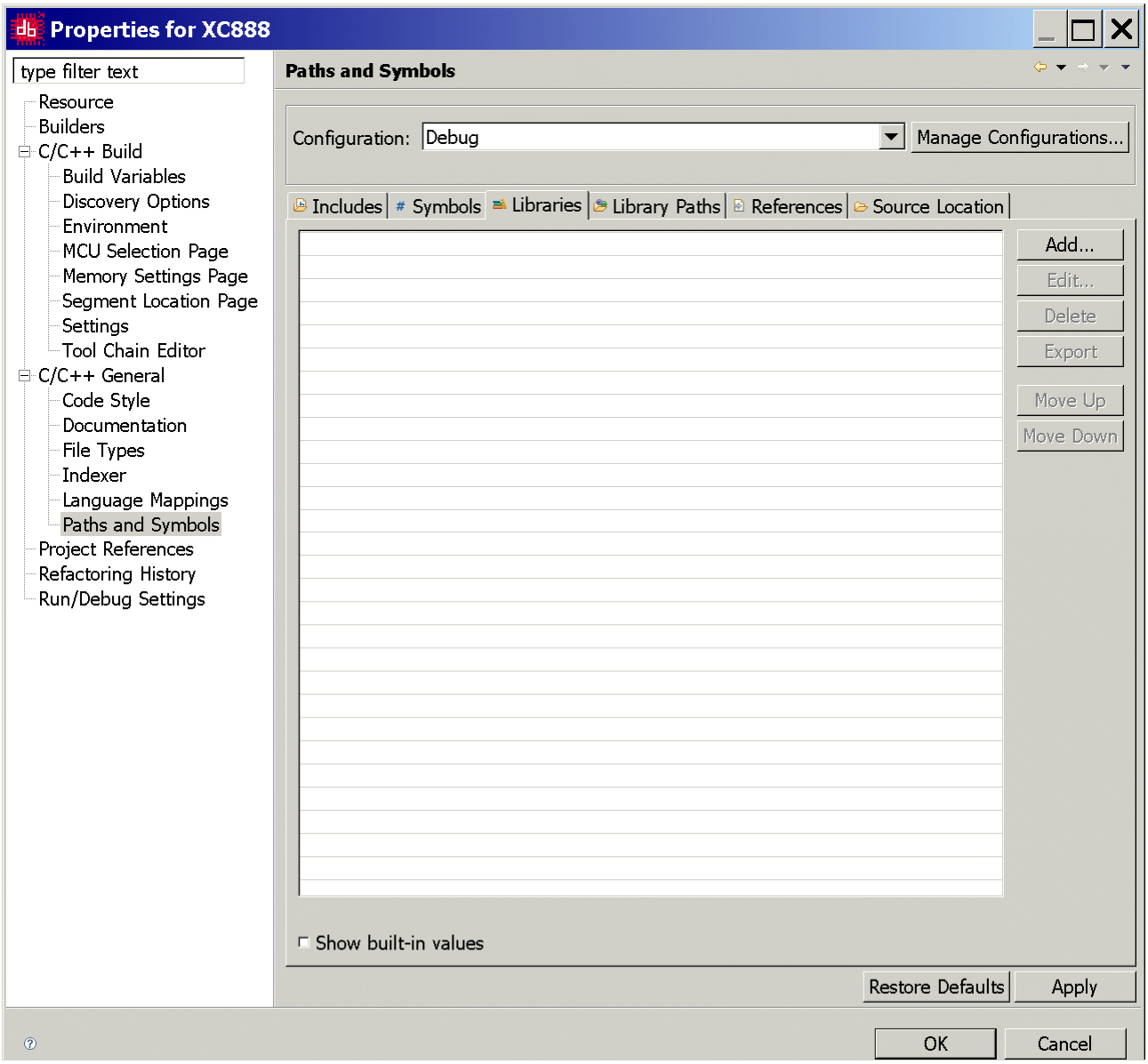
Project – Properties: C/C++ General: Paths and Symbols: Includes:



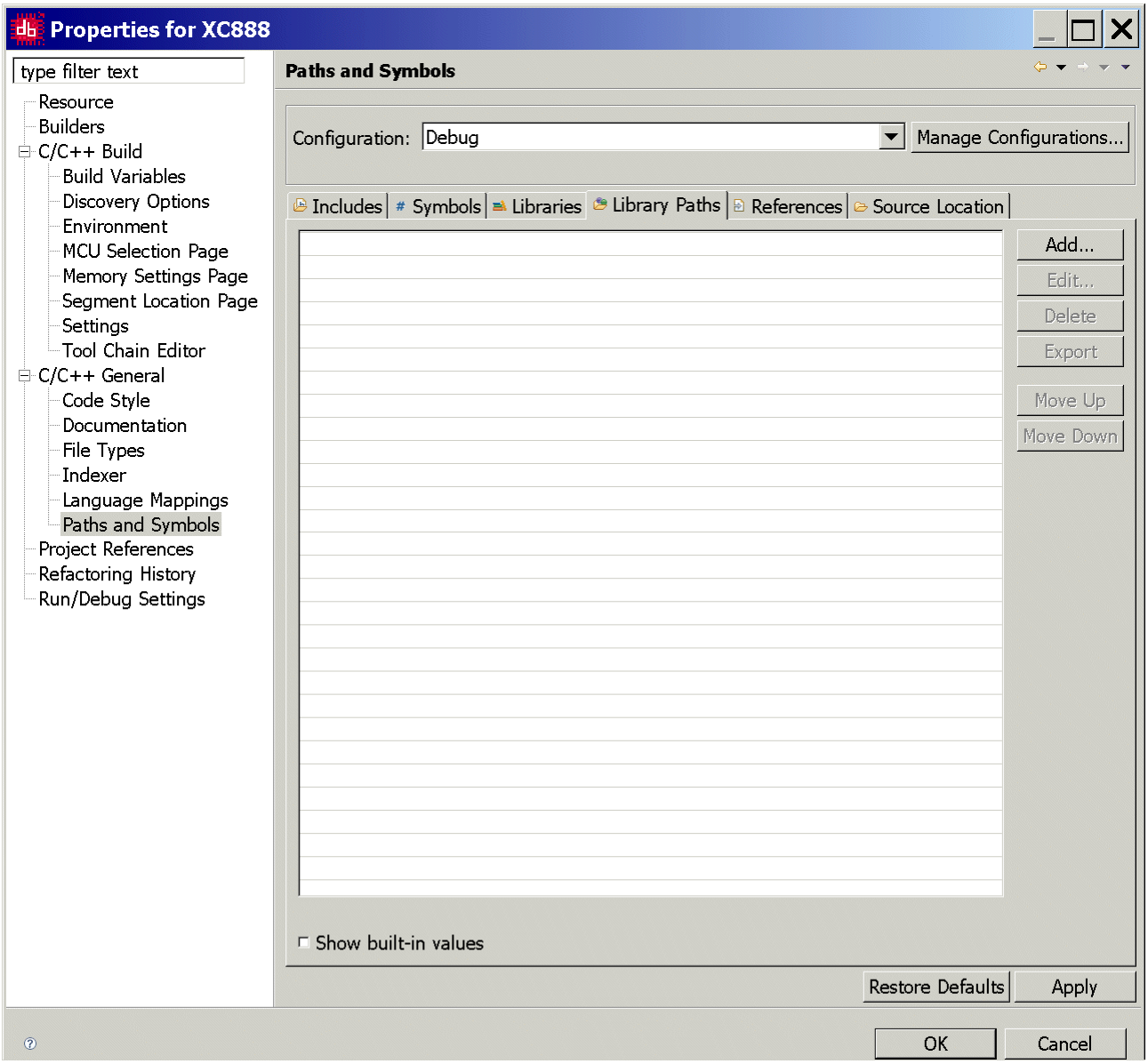
Project – Properties: C/C++ General: Paths and Symbols: Symbols:



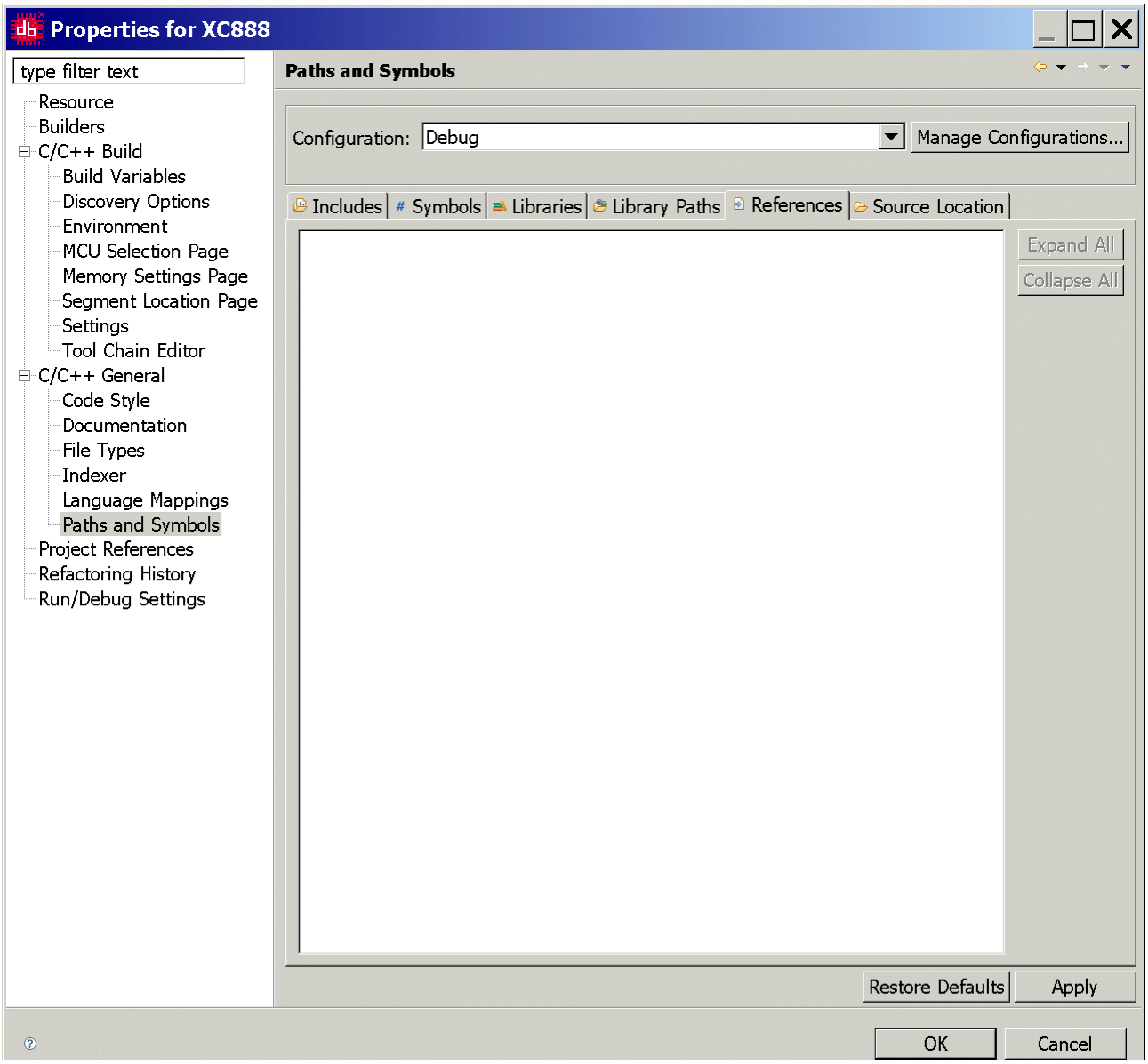
Project – Properties: C/C++ General: Paths and Symbols: Libraries:



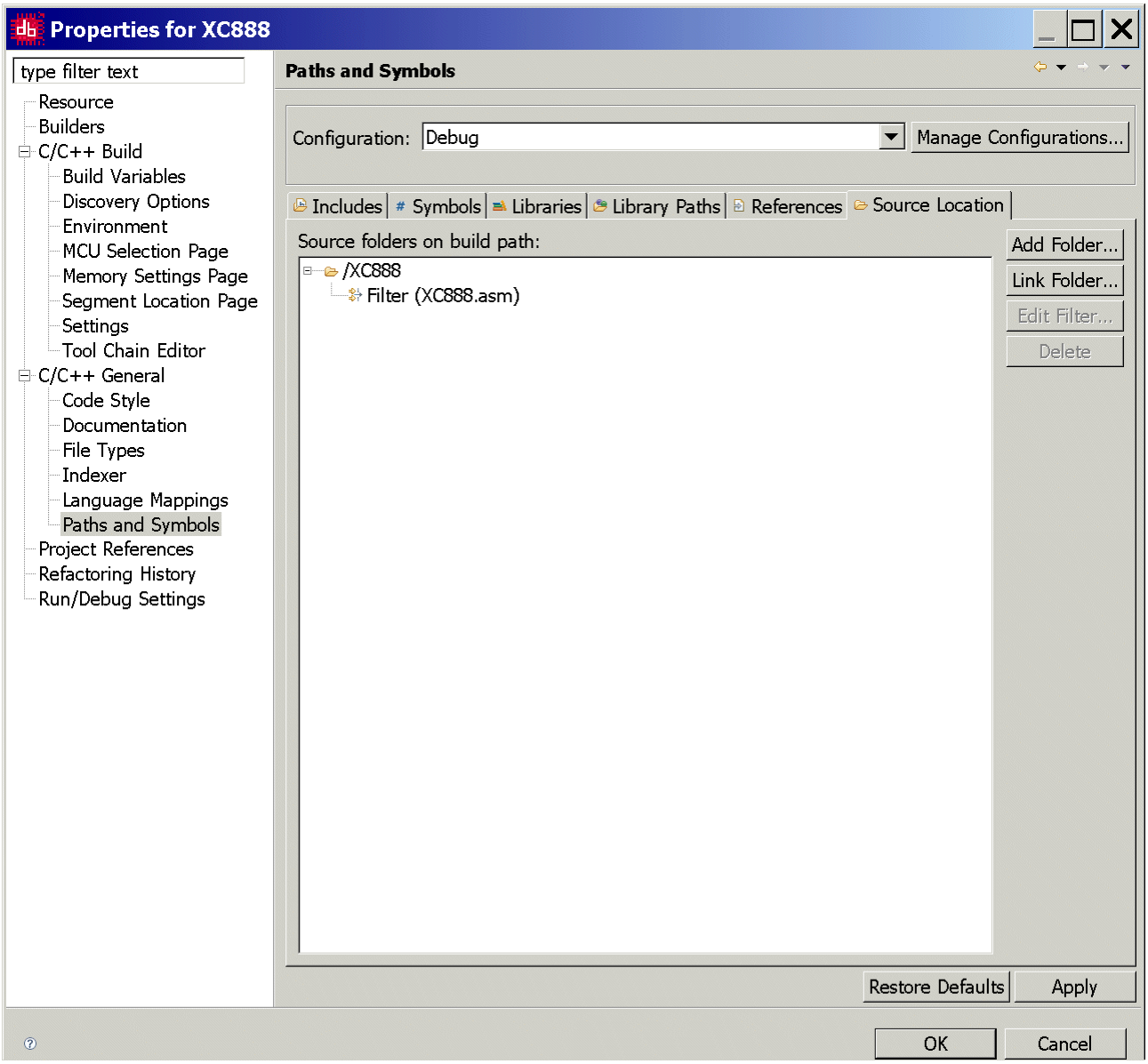
Project – Properties: C/C++ General: Paths and Symbols: Library Paths:



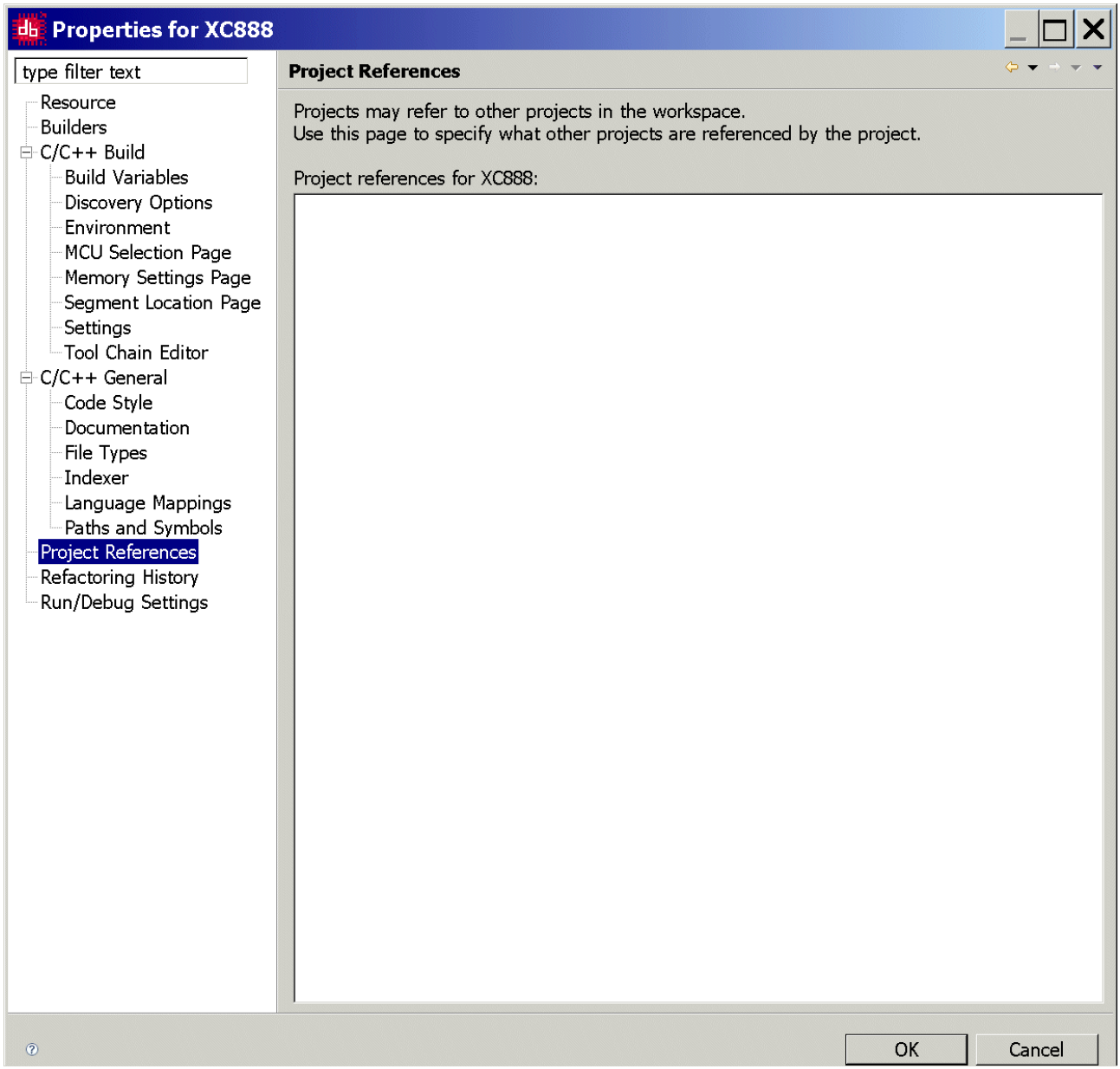
Project – Properties: C/C++ General: Paths and Symbols: References:



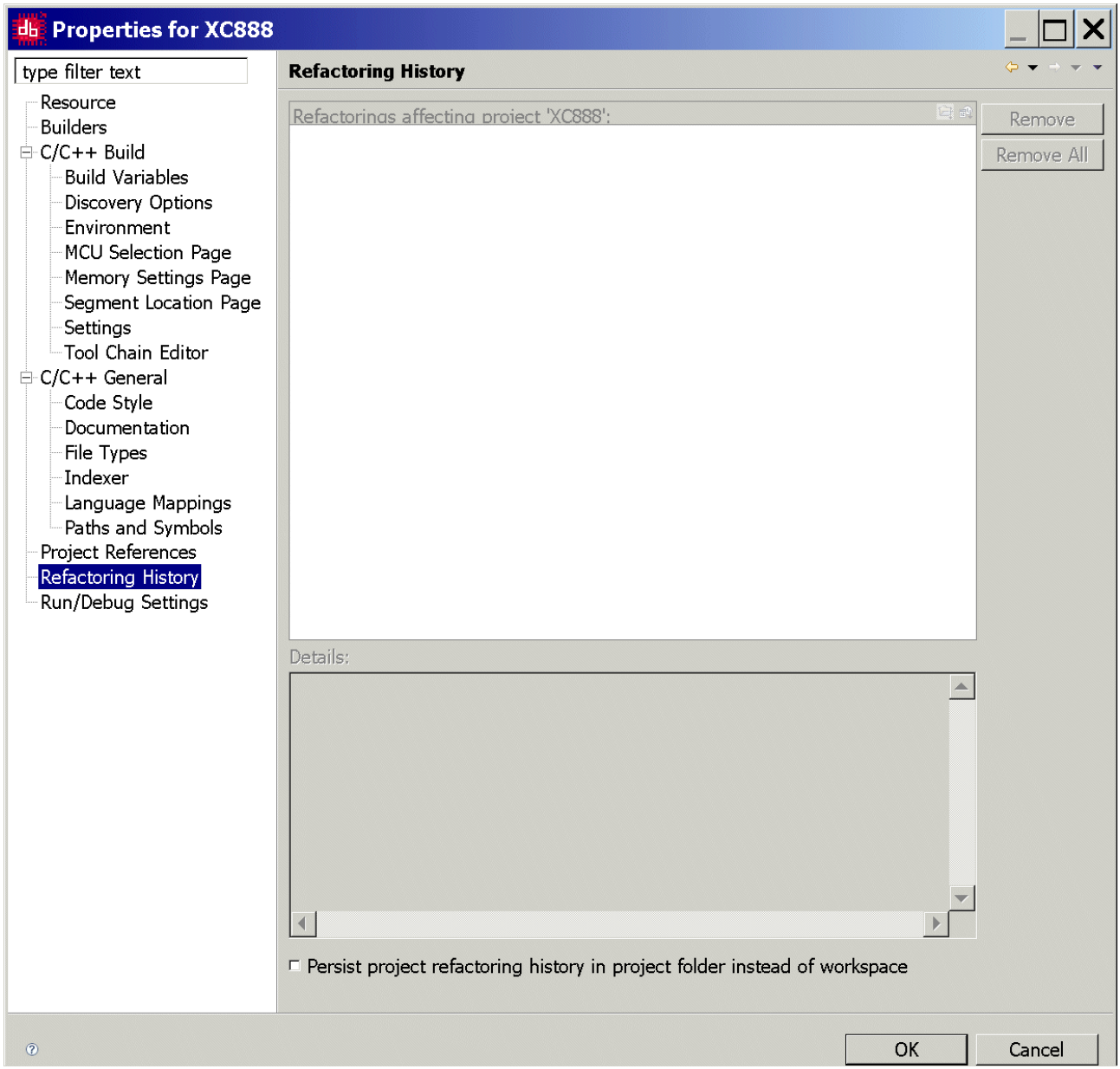
Project – Properties: C/C++ General: Paths and Symbols: Source Location:



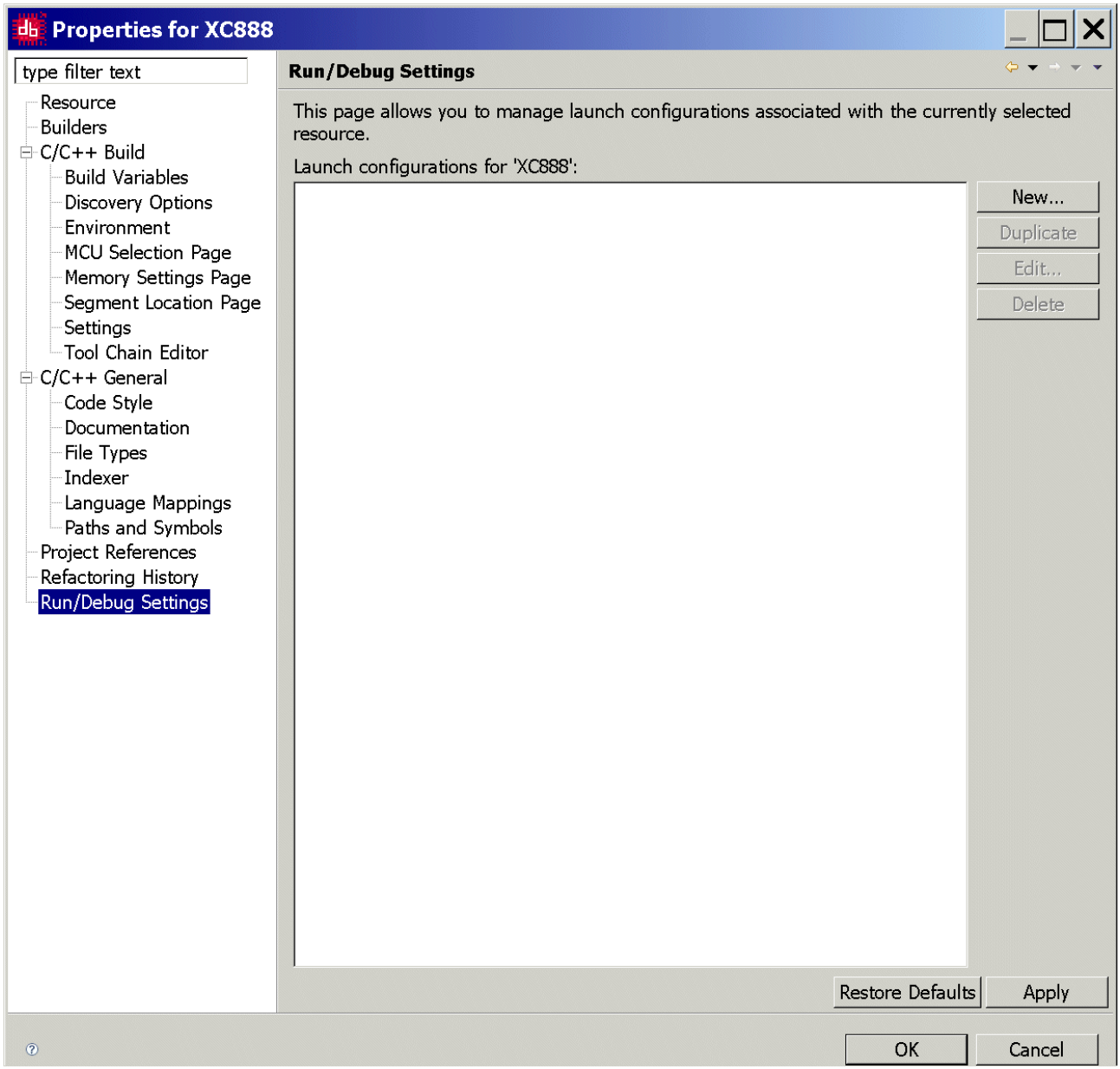
Project – Properties: Project References:



Project – Properties: Refactoring History:

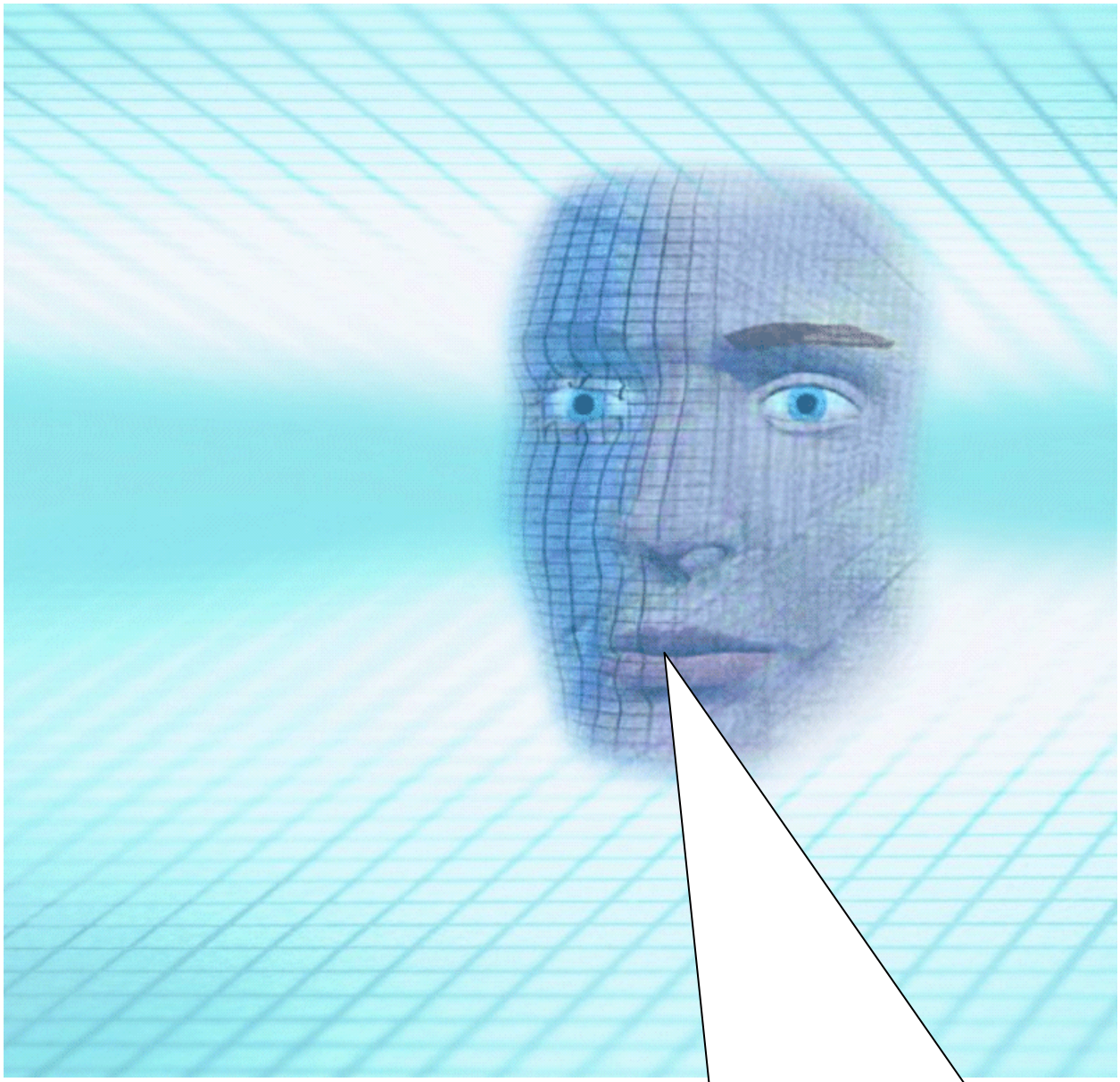


Project – Properties: Run/Debug Settings:



OK

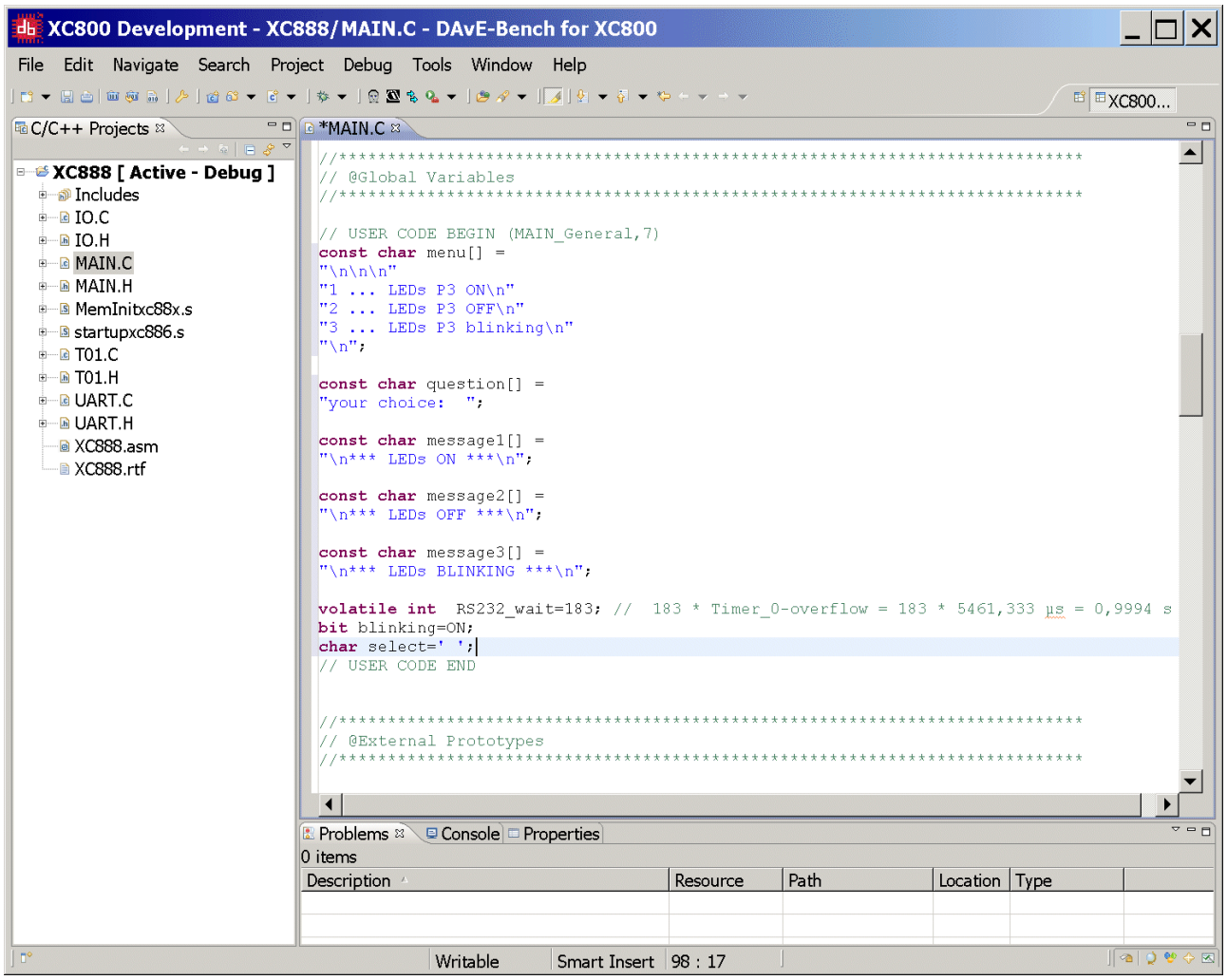
Insert your application specific program:



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!

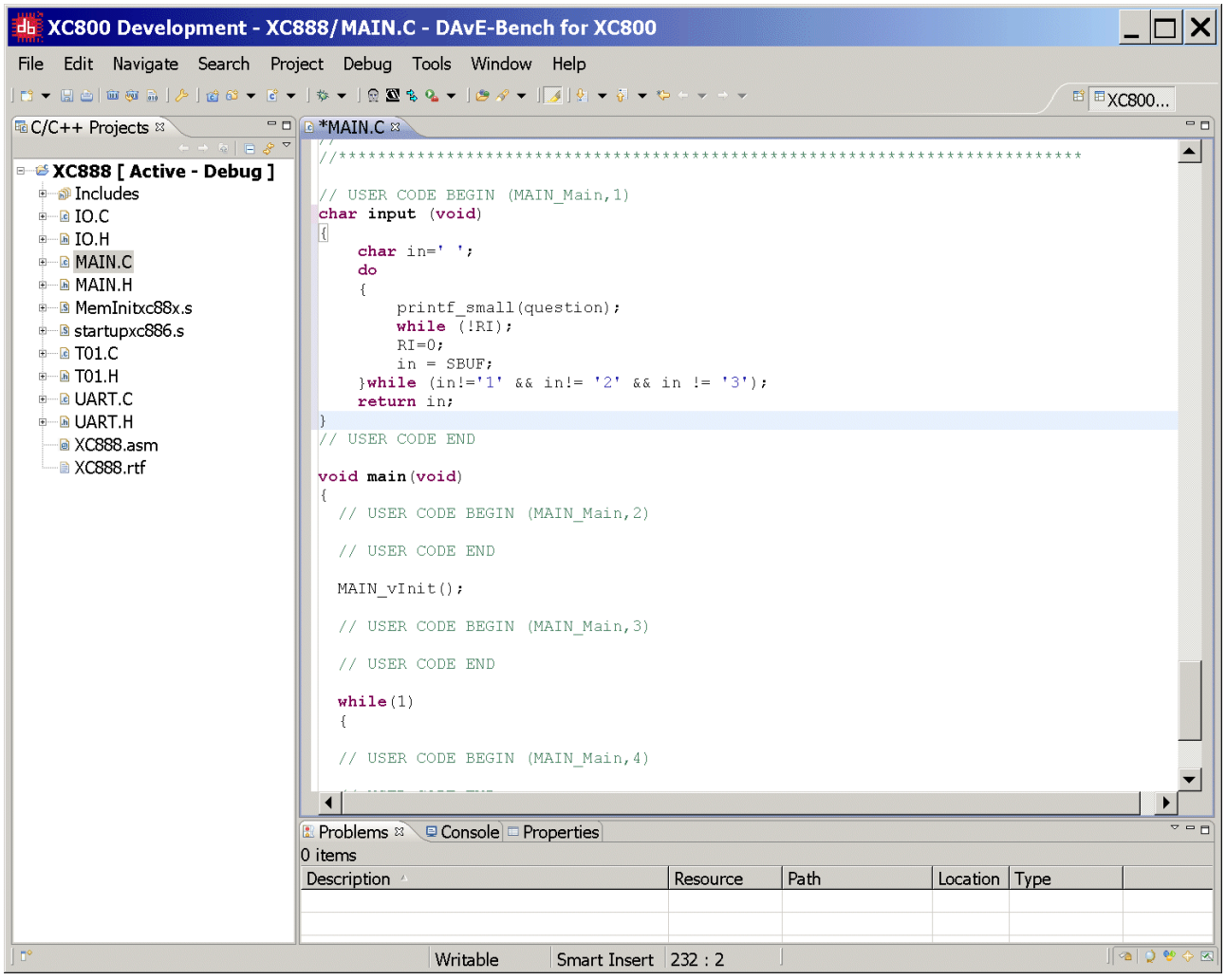
Double click MAIN.C and insert Global Variables:

```
const char menu[] =
"\n\n\n"
"1 ... LEDs P3 ON\n"
"2 ... LEDs P3 OFF\n"
"3 ... LEDs P3 blinking\n"
"\n";

const char question[] =
"your choice:  ";

const char message1[] =
"\n*** LEDs ON ***\n";

const char message2[] =
"\n*** LEDs OFF ***\n";

const char message3[] =
"\n*** LEDs BLINKING ***\n";

volatile int  RS232_wait=183; //  183 * Timer_0-overflow = 183 * 5461,333 µs = 0,9994 s
bit blinking=ON;
char select=' ';
```


XC800 Development - XC888/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC888 [Active - Debug]
Includes
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc88x.s
startupxc886.s
T01.C
T01.H
UART.C
UART.H
XC888.asm
XC888.rtf
*MAIN.C
//***
// @Global Variables
//***
// USER CODE BEGIN (MAIN_General,7)
const char menu[] =
"\n\n\n"
"1 ... LEDs P3 ON\n"
"2 ... LEDs P3 OFF\n"
"3 ... LEDs P3 blinking\n"
"\n";
const char question[] =
"your choice: ";
const char message1[] =
"\n*** LEDs ON ***\n";
const char message2[] =
"\n*** LEDs OFF ***\n";
const char message3[] =
"\n*** LEDs BLINKING ***\n";
volatile int RS232_wait=183; // 183 * Timer_0-overflow = 183 * 5461,333 µs = 0,9994 s
bit blinking=ON;
char select=' ';
// USER CODE END
//***
// @External Prototypes
//***
Problems
Console
Properties
0 items
Description
Resource
Path
Location
Type
Writable
Smart Insert
98 : 17

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

```
char input (void)
{
        char in=' ';
        do
        {
                printf_small(question);
                while (!RI);
                RI=0;
                in = SBUF;
        }while (in!='1' && in!= '2' && in != '3');
        return in;
}
```



Additional information: printf(), printf_small(), printf_fast_f():

SDCC Compiler User Guide

SDCC 2.9.0
$Date:: 2009-03-13 #$
$Revision: 5413 $

3.17.2 Stdclib functions (puts, printf, strcat etc.)

3.17.2.1 <stdio.h>

getchar(), putchar() As usual on embedded systems you have to provide your own `getchar()` and `putchar()` routines. SDCC does not know whether the system connects to a serial line with or without handshake, LCD, keyboard or other device. And whether a `lf` to `crlf` conversion within `putchar()` is intended. You'll find examples for serial routines f.e. in sdcc/device/lib. For the mcs51 this minimalistic polling `putchar()` routine might be a start:

```
void putchar (char c) {
    while (!TI)     /* assumes UART is initialized */
        ;
    TI = 0;
    SBUF = c;
}
```

printf(), printf_small(), printf_fast_f() (continuation):

printf() The default `printf()` implementation in `printf_large.c` does not support float (except on ds390), only <NO FLOAT> will be printed instead of the value. To enable floating point output, recompile it with the option *-DUSE_FLOATS=1* on the command line. Use *--model-large* for the mcs51 port, since this uses a lot of memory. To enable float support for the pic16 targets, see 4.6.8.

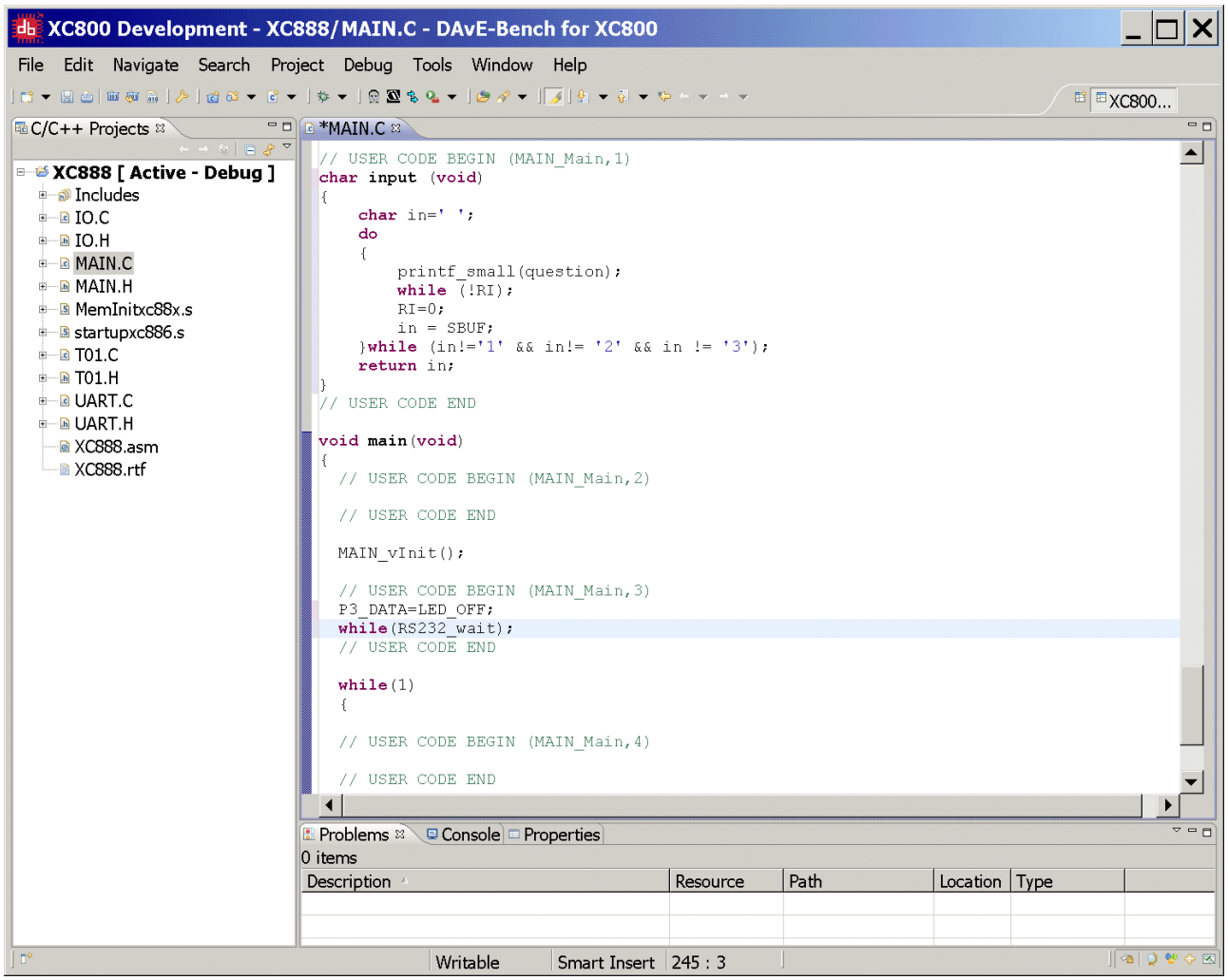
If you're short on code memory you might want to use `printf_small()` *instead* of `printf()`. For the mcs51 there additionally are assembly versions `printf_tiny()` (subset of printf using less than 270 bytes) and `printf_fast()` and `printf_fast_f()` (floating-point aware version of printf_fast) which should fit the requirements of many embedded systems (printf_fast() can be customized by unsetting #defines to *not* support long variables and field widths). Be sure to use only one of these printf options within a project.

Feature matrix of different *printf* options on mcs51.

mcs51	printf	printf USE_FLOATS=1	printf_small	printf_fast	printf_fast_f	printf_tiny
filename	printf_large.c	printf_large.c	printfl.c	printf_fast.c	printf_fast_f.c	printf_tiny.c
"Hello World" size small / large	1.7k / 2.4k	4.3k / 5.6k	1.2k / 1.8k	1.3k / 1.3k	1.9k / 1.9k	0.44k / 0.44k
code size small / large	1.4k / 2.0k	2.8k / 3.7k	0.45k / 0.47k (+ _ltoa)	1.2k / 1.2k	1.6k / 1.6k	0.26k / 0.26k
formats	cdi*o*psux	cdfi*o*psux	cd*o*sx	cdsux	cdfsux	cdsux
long (32 bit) support	x	x	x	x	x	-
byte arguments on stack	b	b	-	-	-	-
float format	-	%f	-	-	%f[9]	-
float formats %e %g	-	-	-	-	-	-
field width	x	x	-	x	x	-
string speed[10], small / large	1.52 / 2.59 ms	1.53 / 2.62 ms	0.92 / 0.93 ms	0.45 / 0.45 ms	0.46 / 0.46 ms	0.45 / 0.45 ms
int speed[11], small / large	3.01 / 3.61 ms	3.01 / 3.61 ms	3.51 / 18.13 ms	0.22 / 0.22 ms	0.23 / 0.23 ms	0.25 / 0.25 ms[12]
long speed[13], small / large	5.37 / 6.31 ms	5.37 / 6.31 ms	8.71 / 40.65 ms	0.40 / 0.40 ms	0.40 / 0.40 ms	-
float speed[14], small / large	-	7.49 / 22.47 ms	-	-	1.04 / 1.04 ms	-

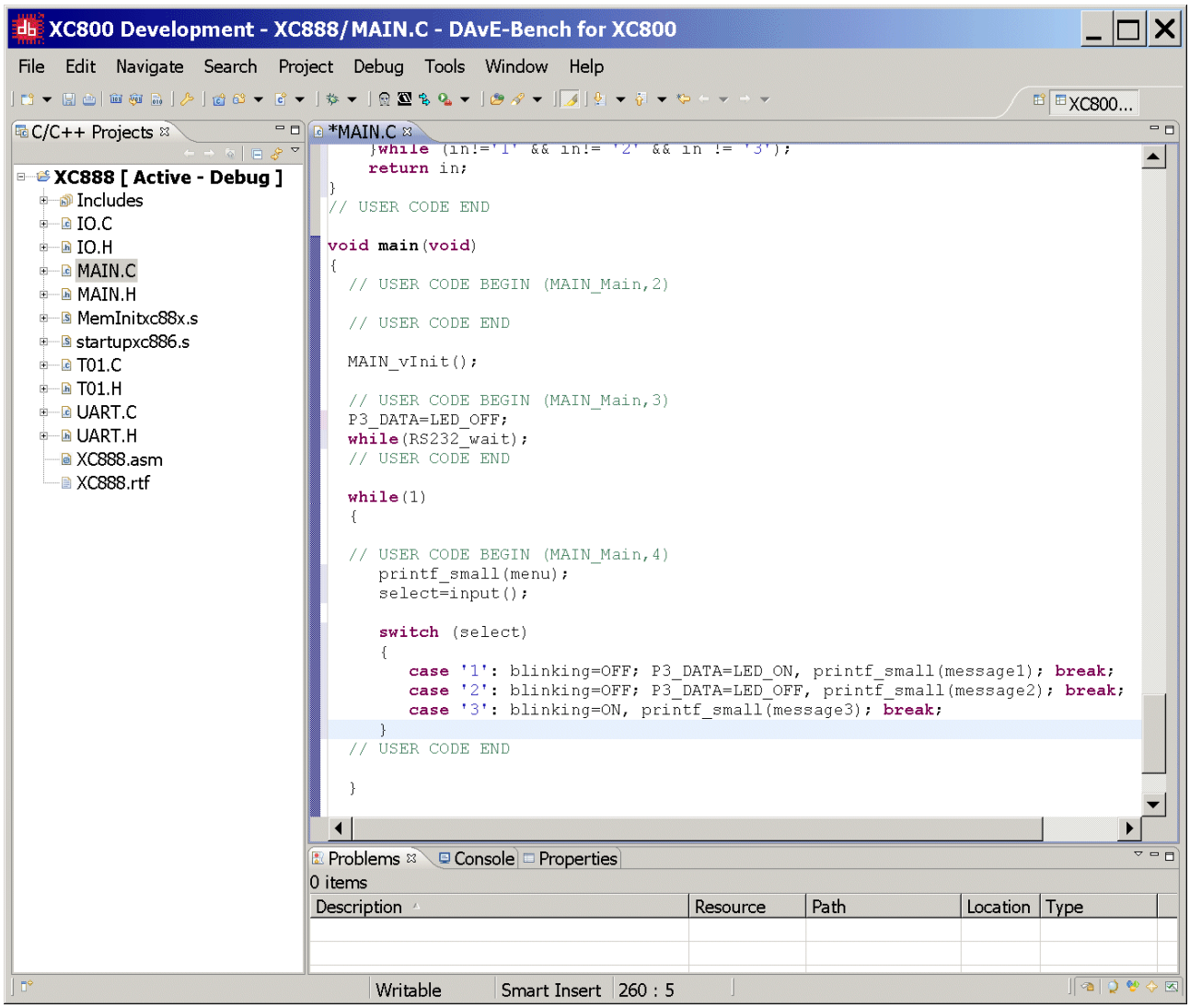
Double click MAIN.C and insert the following code in the main function:

```
P3_DATA=LED_OFF;
while(RS232_wait);
```



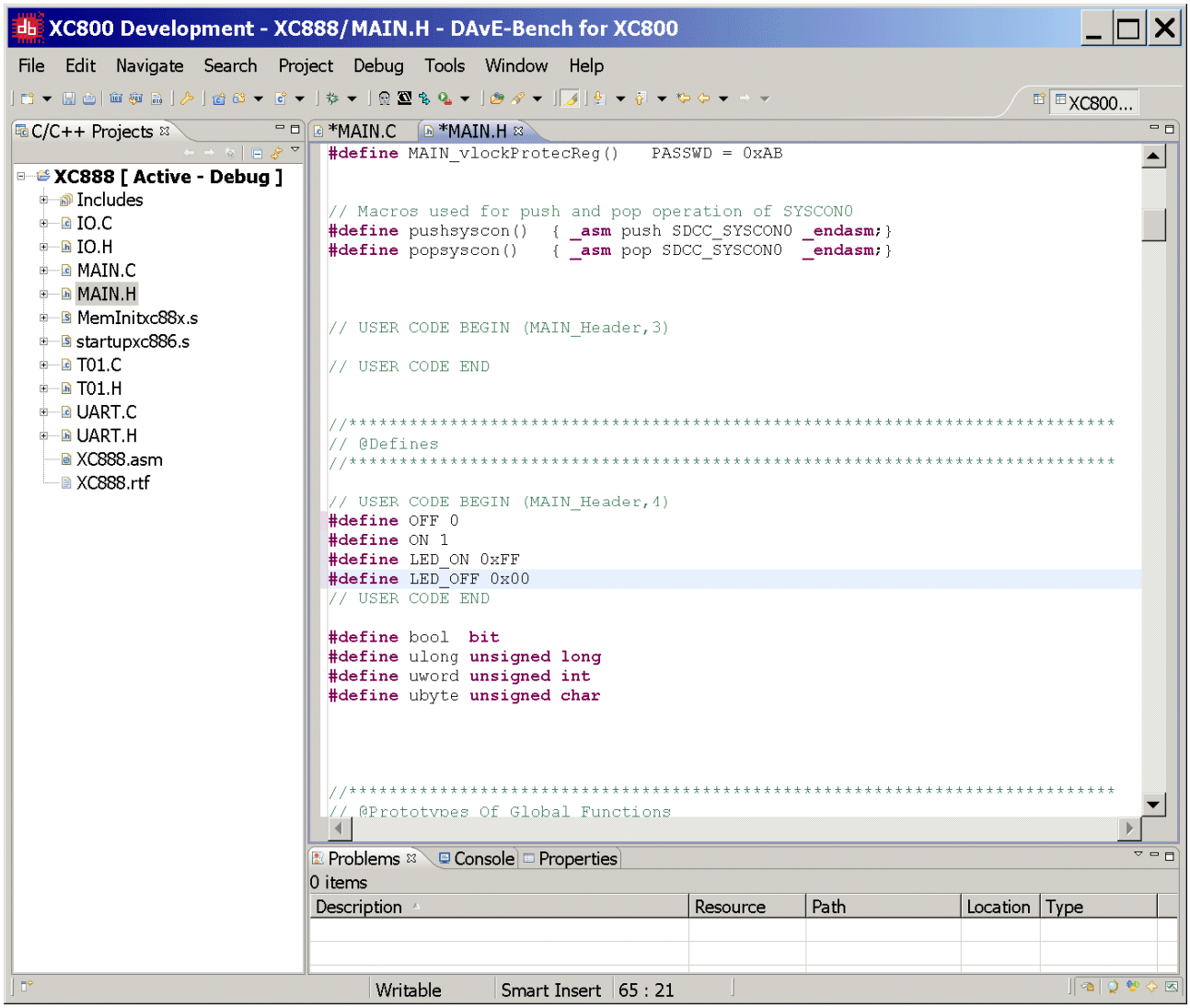
Double click MAIN.C and insert the following code in the main function into the while(1) loop:

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

switch (select)
{
    case '1': blinking=OFF; P3_DATA=LED_ON, printf_small(message1); break;
    case '2': blinking=OFF; P3_DATA=LED_OFF, printf_small(message2); break;
    case '3': blinking=ON, printf_small(message3); break;
}
```



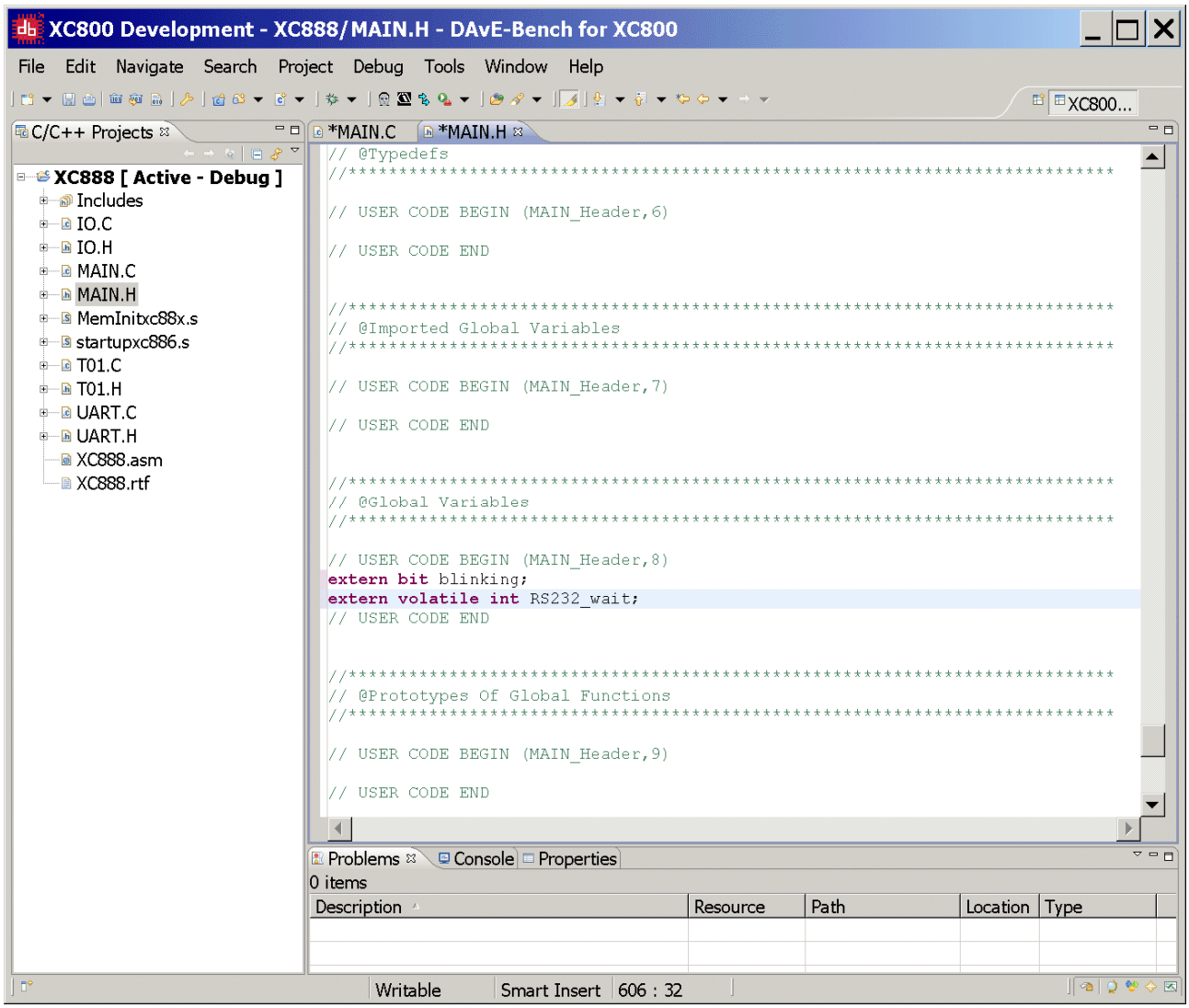
Double click Main.h and insert the following Defines:

```
#define OFF 0
#define ON 1
#define LED_ON 0xFF
#define LED_OFF 0x00
```



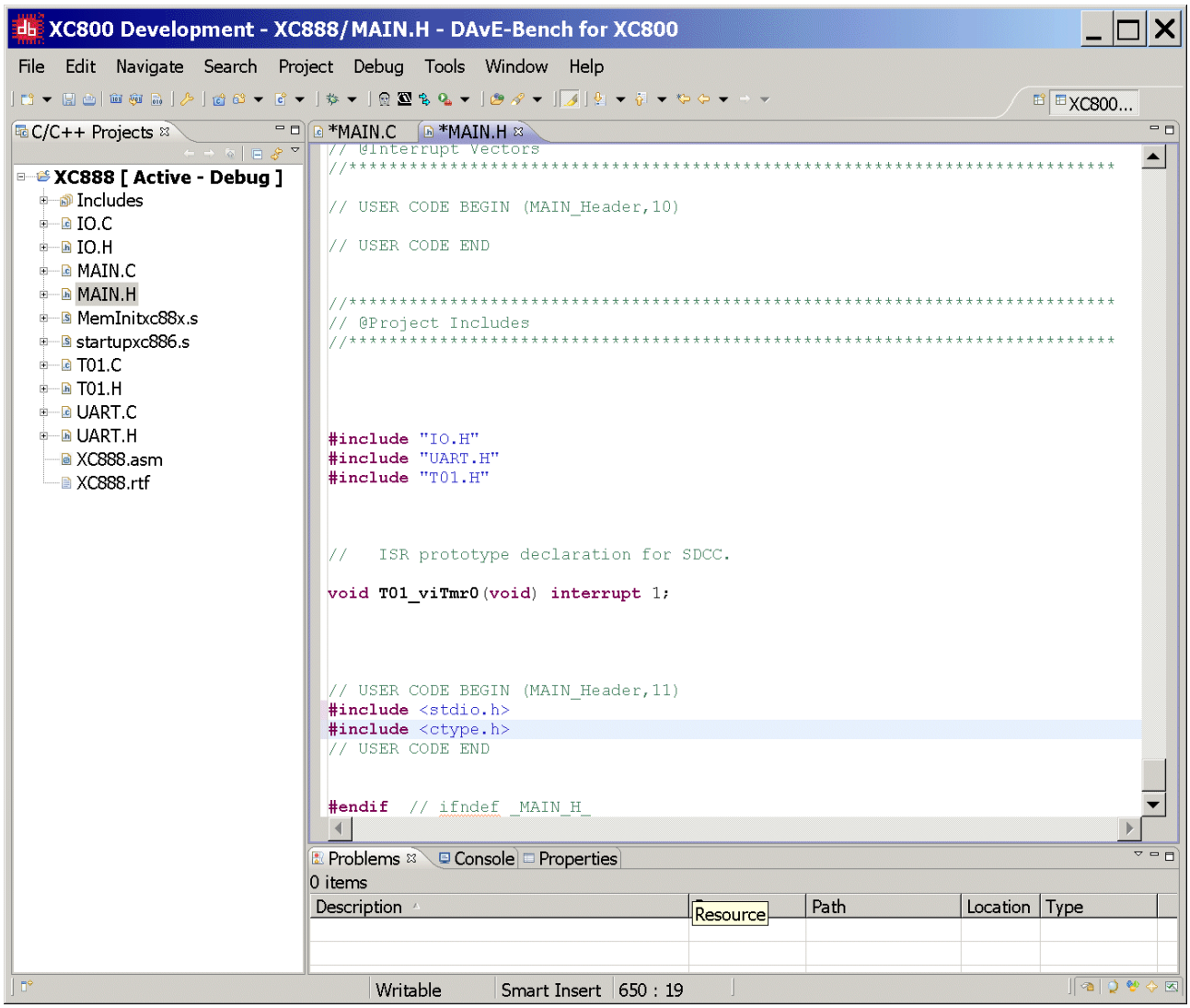
Double click Main.h and insert extern-declarations “Global Variables”:

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



Double click Main.h and insert include files:

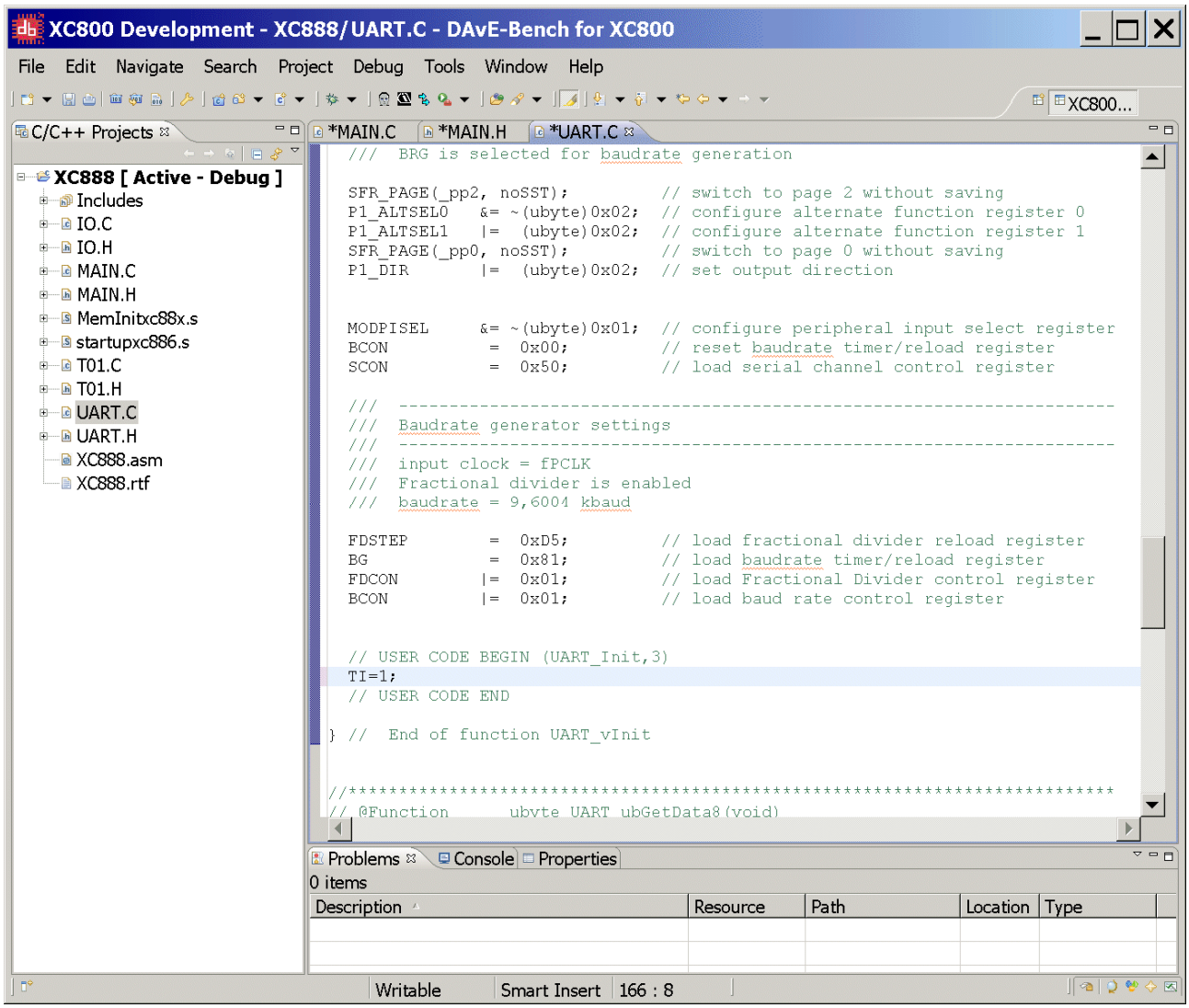
```
#include <stdio.h>
#include <ctype.h>
```



Double click UART.C

Insert code into the UART_vInit function: (to start printf_small()):

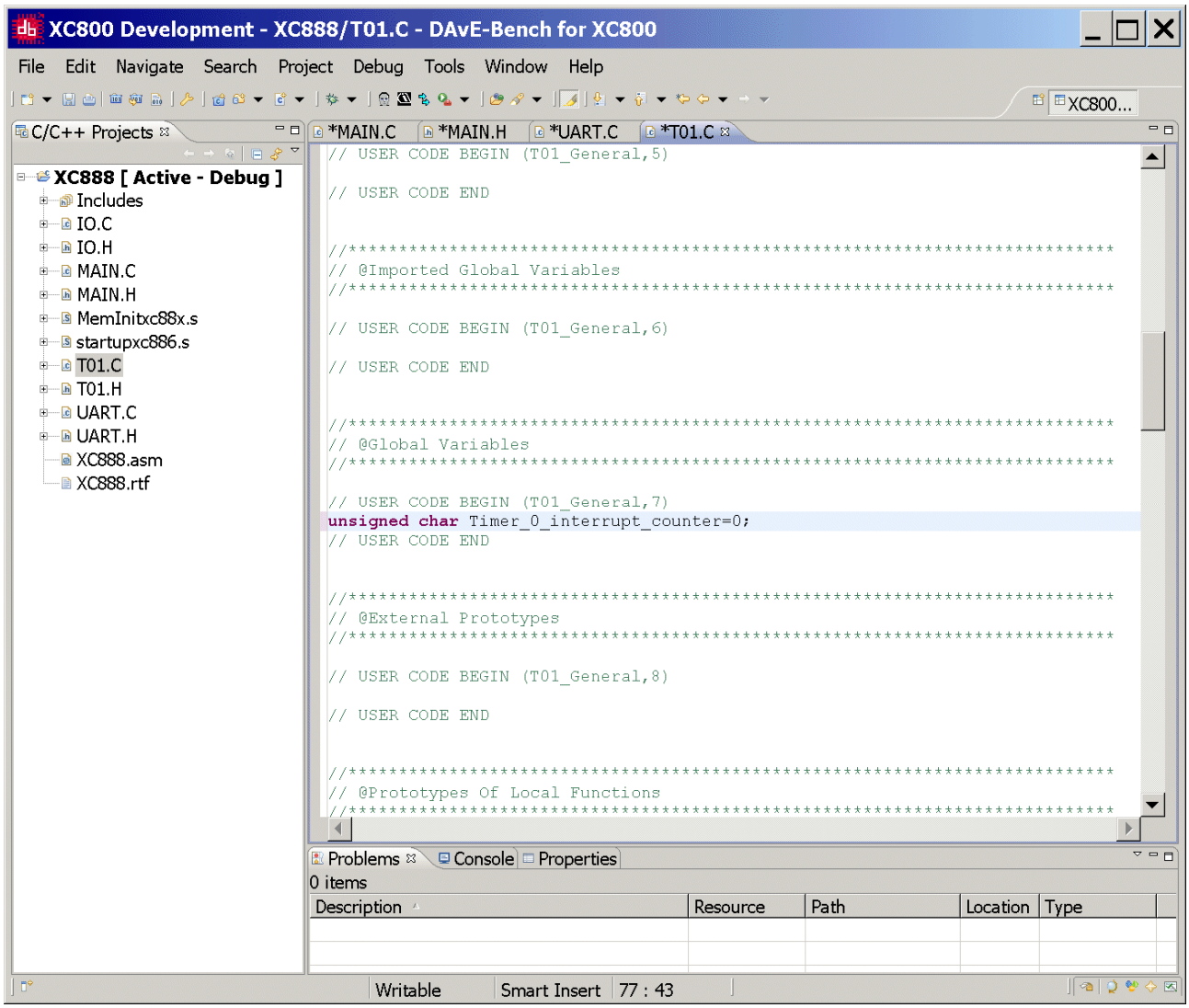
```
TI=1;
```



Double click T01.C

Insert the following global variable:

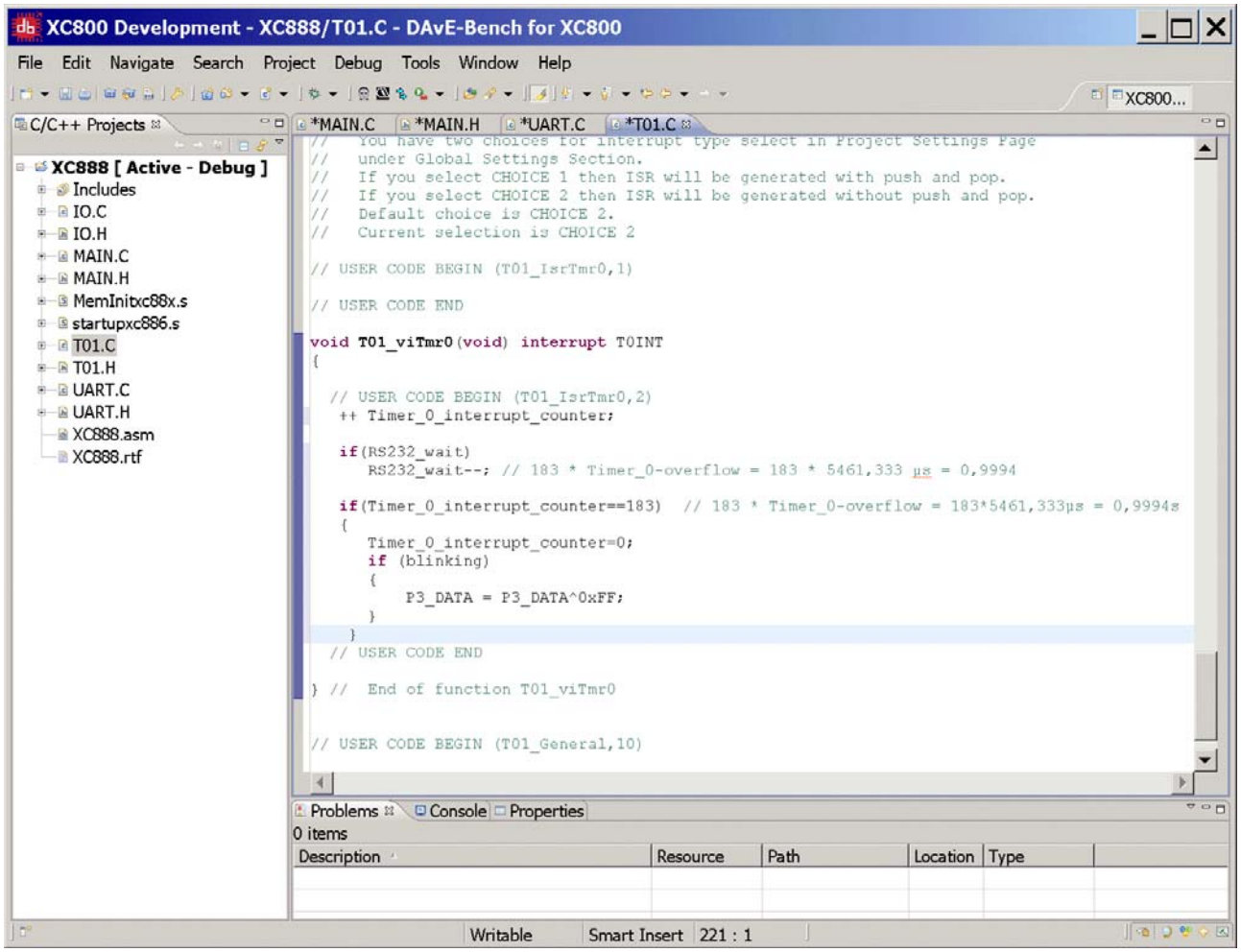
```
unsigned char Timer_0_interrupt_counter=0;
```



Double click T01.C

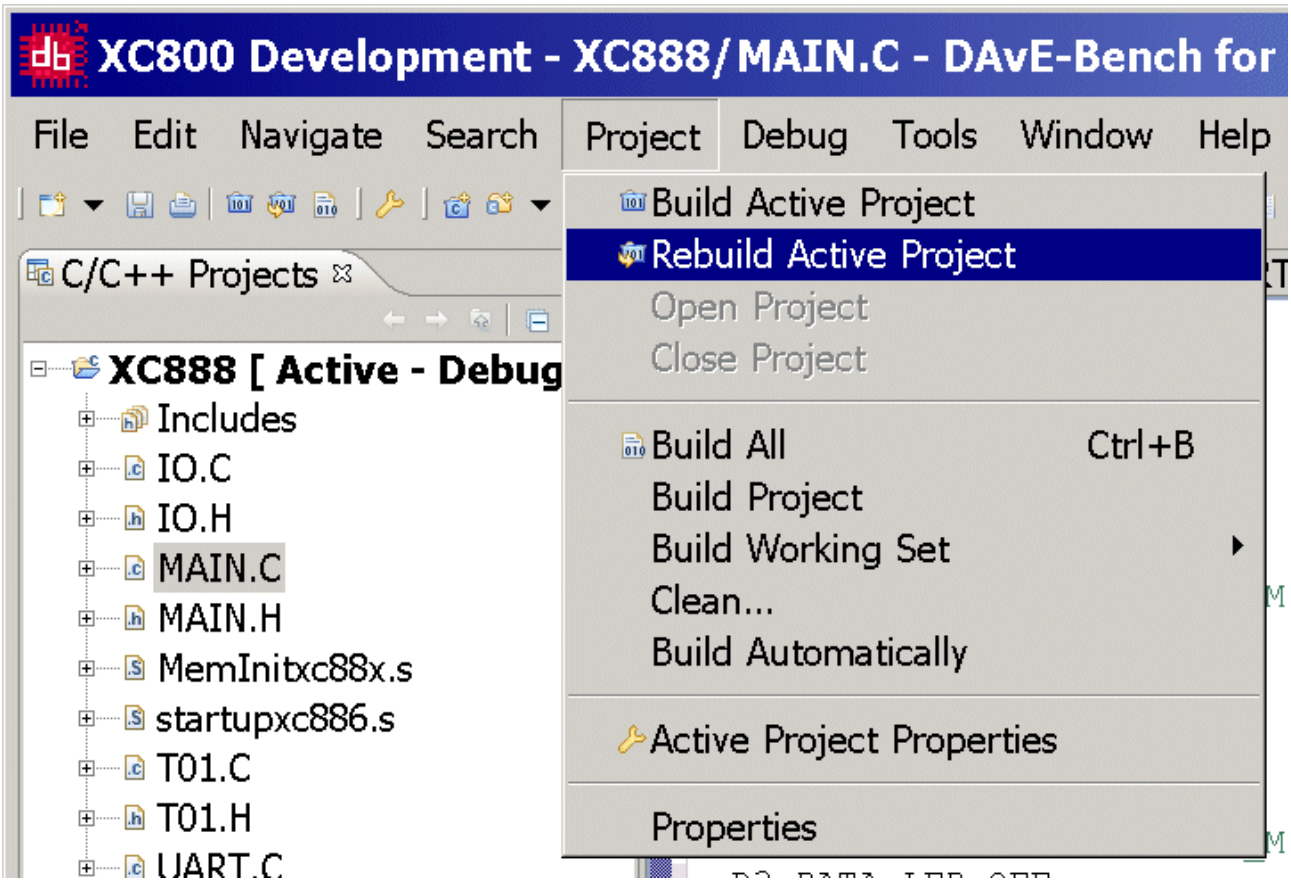
Insert code for T0 interrupt service routine:

```
++ Timer_0_interrupt_counter;

if(RS232_wait)
   RS232_wait--; // 183 * Timer_0-overflow = 183 * 5461,333 µs = 0,9994

if(Timer_0_interrupt_counter==183)  // 183 * Timer_0-overflow = 183*5461,333µs = 0,9994s
{
  Timer_0_interrupt_counter=0;
  if (blinking)
   {
      P3_DATA = P3_DATA^0xFF;
   }
}
```

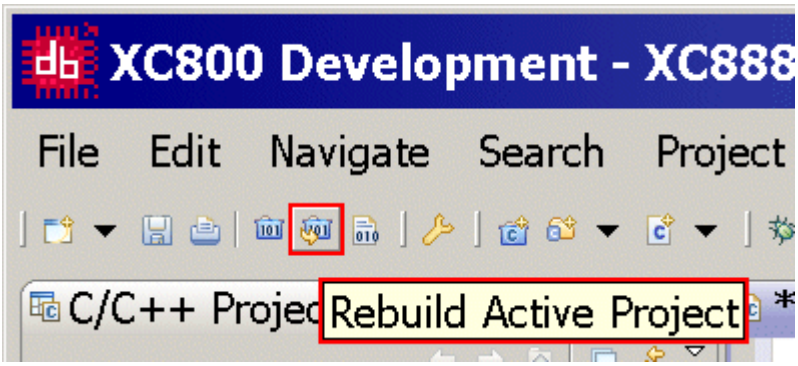


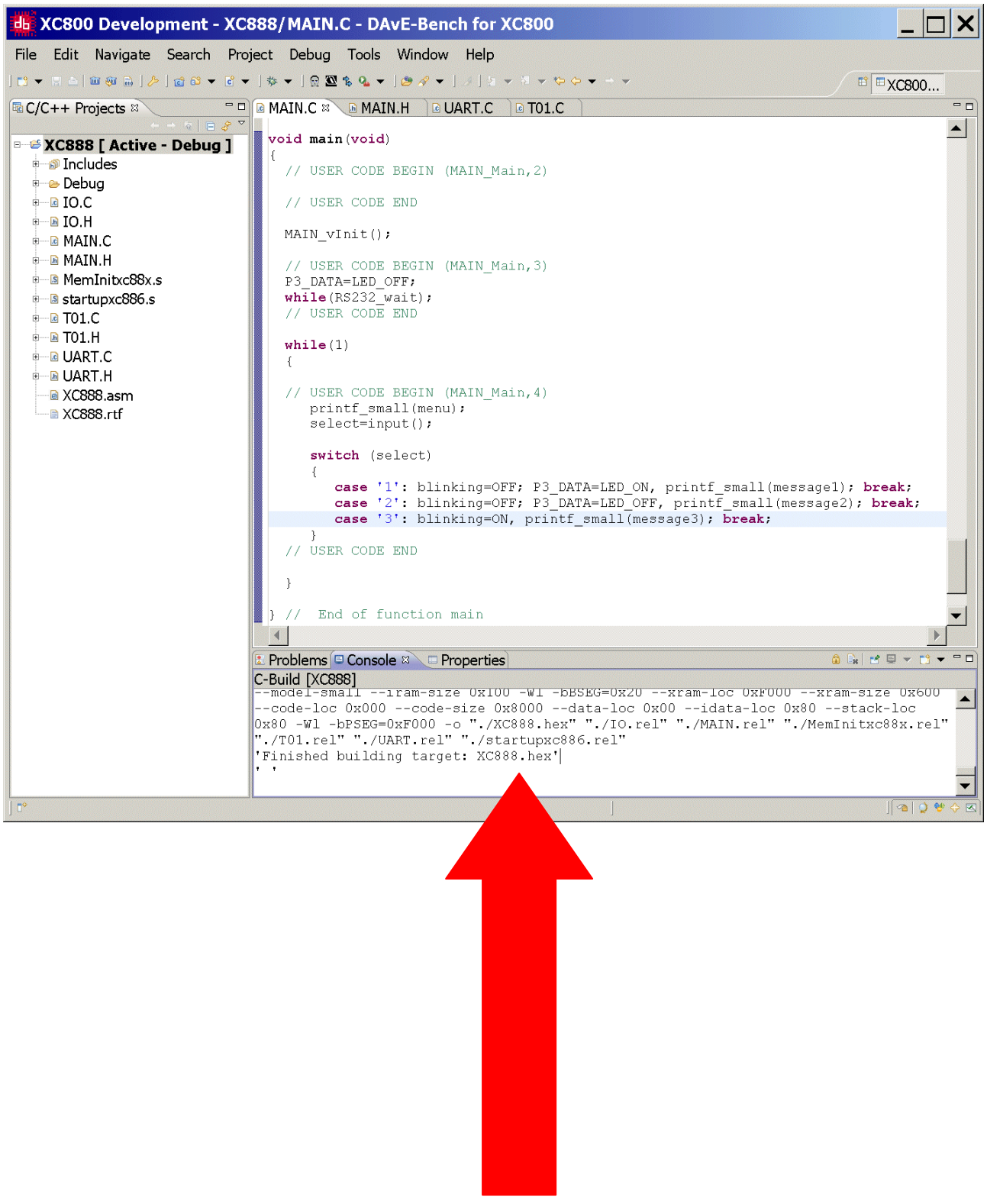
Generate/Build your application program:

Project – Rebuild Active Project



or: click 




XC800 Development - XC888/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
C/C++ Projects
XC888 [Active - Debug]
Includes
Debug
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc88x.s
startupxc886.s
T01.C
T01.H
UART.C
UART.H
XC888.asm
XC888.rtf
MAIN.C
MAIN.H
UART.C
T01.C
void main(void)
{
// USER CODE BEGIN (MAIN_Main,2)
// USER CODE END
MAIN_vInit();
// USER CODE BEGIN (MAIN_Main,3)
P3_DATA=LED_OFF;
while(RS232_wait);
// USER CODE END
while(1)
{
// USER CODE BEGIN (MAIN_Main,4)
printf_small(menu);
select=input();
switch (select)
{
case '1': blinking=OFF; P3_DATA=LED_ON, printf_small(message1); break;
case '2': blinking=OFF; P3_DATA=LED_OFF, printf_small(message2); break;
case '3': blinking=ON, printf_small(message3); break;
}
// USER CODE END
}
} // End of function main
Problems
Console
Properties
C-Build [XC888]
--code-loc 0x000 --code-size 0x8000 --data-loc 0x00 --idata-loc 0x80 --stack-loc
0x80 -Wl -bPSEG=0xF000 -o "./XC888.hex" "./IO.rel" "./MAIN.rel" "./MemInitxc88x.rel"
"./T01.rel" "./UART.rel" "./startupxc886.rel"
'Finished building target: XC888.hex'

```
**** Build of configuration Debug for project XC888 ****

C:\Program Files\DAvE-Bench-011\SDCC_UTILS\make all
'Building file: ../IO.C'
'Invoking: SDCC Compiler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\sdcc" -mXC800 -pXC888_8FF --
model-small  -I"C:/Program Files/DAvE-Bench-011\SDCC_XC800\include" -
I"C:/Program Files/DAvE-Bench-011\SDCC_XC800\include\xc800" -I"C:/Program
Files/DAvE-Bench-011\SDCC_XC800\include\asm\xc800" --opt-code-size --
nooverlay --noinduction --debug -S  -o "IO.s" "../IO.C"
'Finished building: ../IO.C'
' '
'Building file: IO.s'
'Invoking: SDCC Assembler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\as-xc800" -plosgffcx "IO.s"
-O "IO.rel"
'Finished building: IO.s'
' '
'Building file: ../MAIN.C'
'Invoking: SDCC Compiler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\sdcc" -mXC800 -pXC888_8FF --
model-small  -I"C:/Program Files/DAvE-Bench-011\SDCC_XC800\include" -
I"C:/Program Files/DAvE-Bench-011\SDCC_XC800\include\xc800" -I"C:/Program
Files/DAvE-Bench-011\SDCC_XC800\include\asm\xc800" --opt-code-size --
nooverlay --noinduction --debug -S  -o "MAIN.s" "../MAIN.C"
'Finished building: ../MAIN.C'
' '
'Building file: MAIN.s'
'Invoking: SDCC Assembler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\as-xc800" -plosgffcx
"MAIN.s" -O "MAIN.rel"
'Finished building: MAIN.s'
' '
'Building file: ../MemInitxc88x.s'
'Invoking: SDCC Assembler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\as-xc800" -plosgffcx
"../MemInitxc88x.s" -O "MemInitxc88x.rel"
'Finished building: ../MemInitxc88x.s'
' '
'Building file: ../T01.C'
'Invoking: SDCC Compiler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\sdcc" -mXC800 -pXC888_8FF --
model-small  -I"C:/Program Files/DAvE-Bench-011\SDCC_XC800\include" -
I"C:/Program Files/DAvE-Bench-011\SDCC_XC800\include\xc800" -I"C:/Program
Files/DAvE-Bench-011\SDCC_XC800\include\asm\xc800" --opt-code-size --
nooverlay --noinduction --debug -S  -o "T01.s" "../T01.C"
'Finished building: ../T01.C'
' '
'Building file: T01.s'
'Invoking: SDCC Assembler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\as-xc800" -plosgffcx "T01.s"
-O "T01.rel"
'Finished building: T01.s'
```

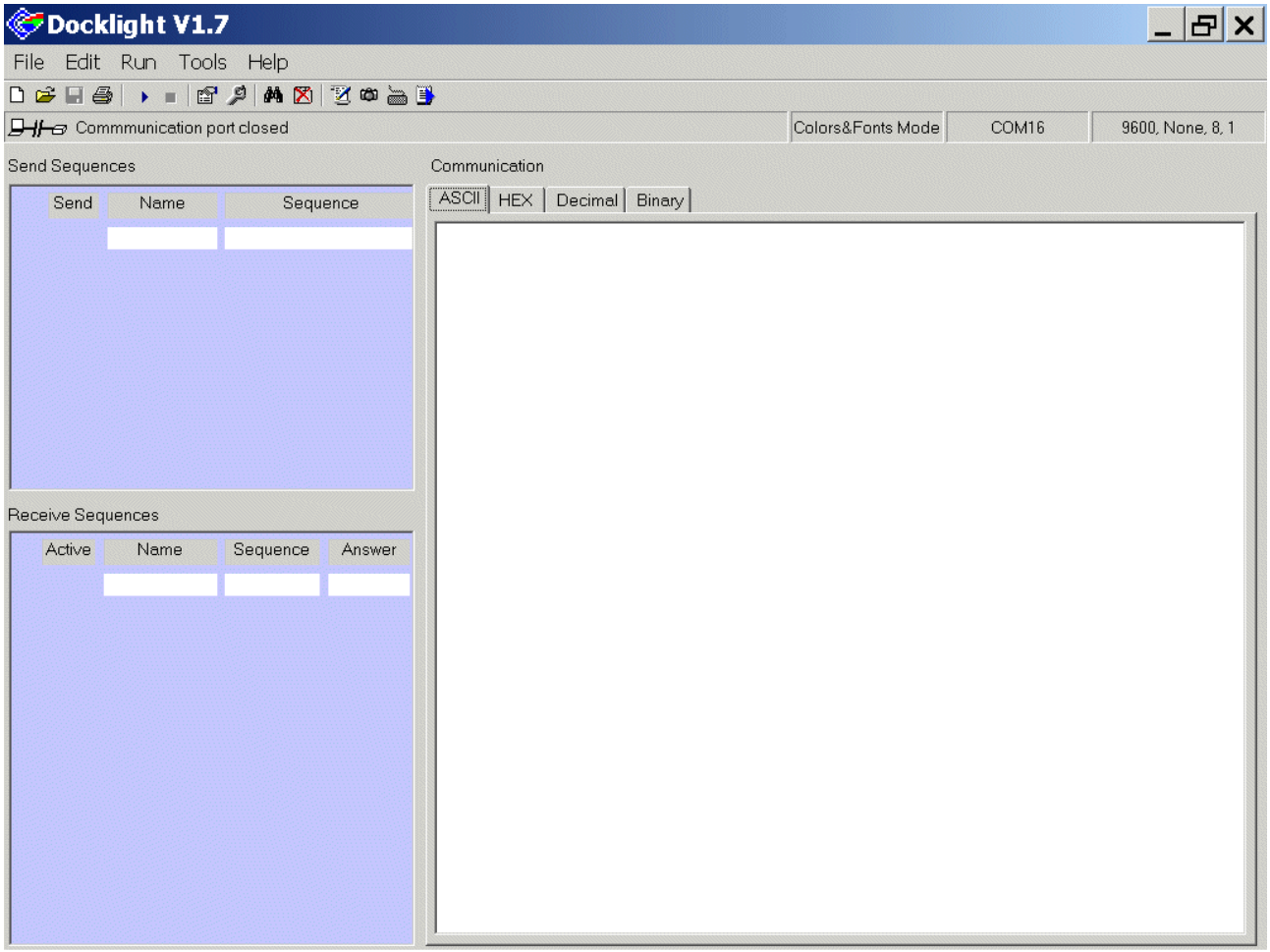
```
' '
'Building file: ../UART.C'
'Invoking: SDCC Compiler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\sdcc" -mXC800 -pXC888_8FF --
model-small  -I"C:/Program Files/DAvE-Bench-011\SDCC_XC800\include" -
I"C:/Program Files/DAvE-Bench-011\SDCC_XC800\include\xc800" -I"C:/Program
Files/DAvE-Bench-011\SDCC_XC800\include\asm\xc800" --opt-code-size --
nooverlay --noinduction --debug -S  -o "UART.s" "../UART.C"
'Finished building: ../UART.C'
' '
'Building file: UART.s'
'Invoking: SDCC Assembler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\as-xc800" -plosgffcx
"UART.s" -O "UART.rel"
MOV dir(0x82),dir(0x99) found at 1431 of UART.s
'Finished building: UART.s'
' '
'Building file: ../startupxc886.s'
'Invoking: SDCC Assembler'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\as-xc800" -plosgffcx
"../startupxc886.s" -O "startupxc886.rel"
'Finished building: ../startupxc886.s'
' '
'Building target: XC888.hex'
'Invoking: SDCC Linker'
"C:/Program Files/DAvE-Bench-011\SDCC_XC800\bin\sdcc" --debug -mXC800 -
pXC888_8FF --model-small --iram-size 0x100 -Wl -bBSEG=0x20 --xram-loc 0xF000
--xram-size 0x600 --code-loc 0x000 --code-size 0x8000 --data-loc 0x00 --
idata-loc 0x80 --stack-loc 0x80 -Wl -bPSEG=0xF000 -o "./XC888.hex"
"./IO.rel" "./MAIN.rel" "./MemInitxc88x.rel" "./T01.rel" "./UART.rel"
"./startupxc886.rel"
'Finished building target: XC888.hex'
                                    ' '
```

5.) Using the debugger (DAvE Bench):

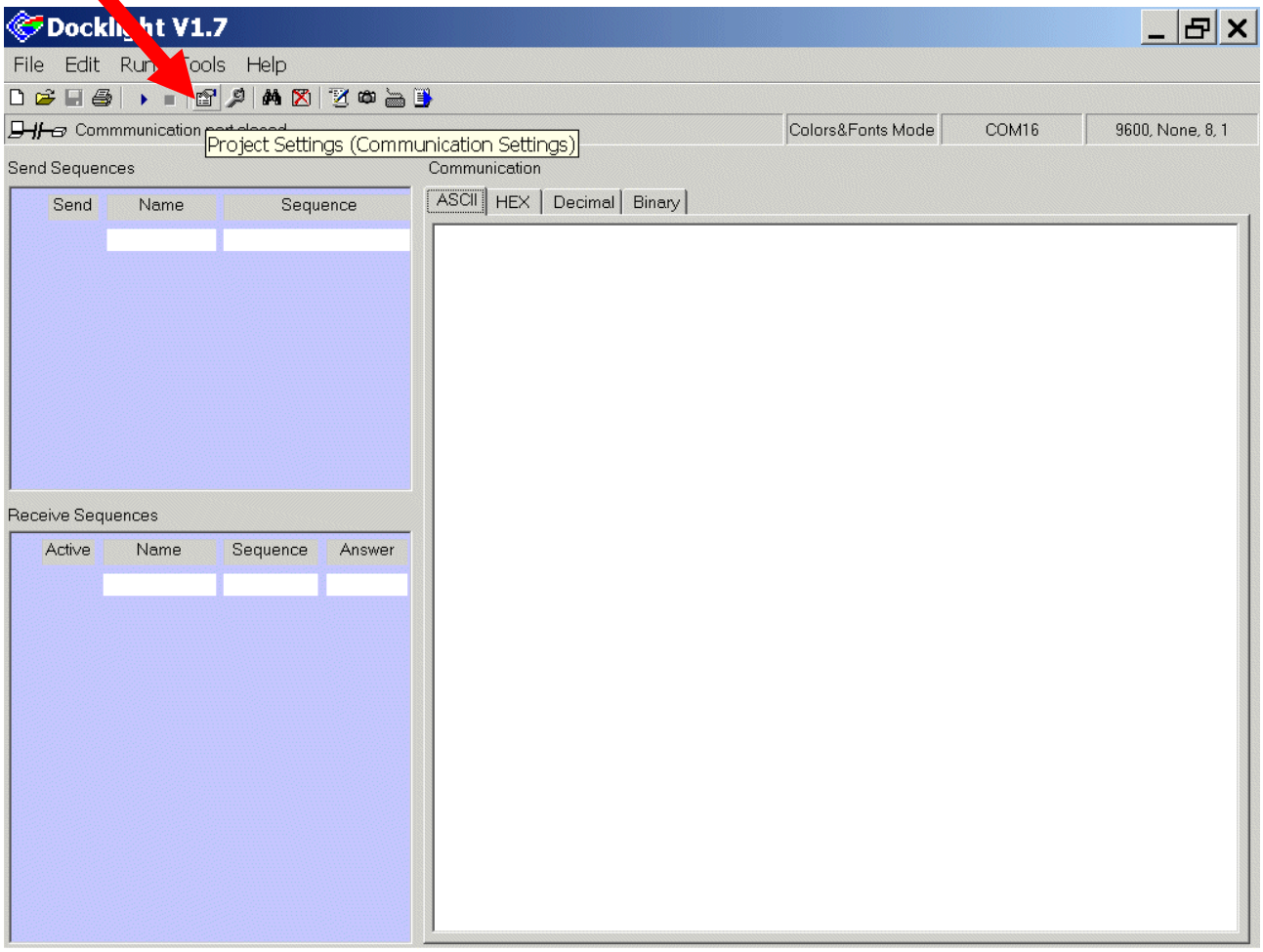
Note:
✵**1:** When the TANTINO-XC800-DEBUGGER-BOX is already connected to the XC888 Starter Kit Board, the Starter Kit Board must be supplied with power for the TANTINO-XC800-DEBUGGER-BOX to work properly.
If the power supply is not connected to the Board, you will see no information in the JTAG Device Chain window.

Note:
Now we need a terminal program which is able to handle 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:



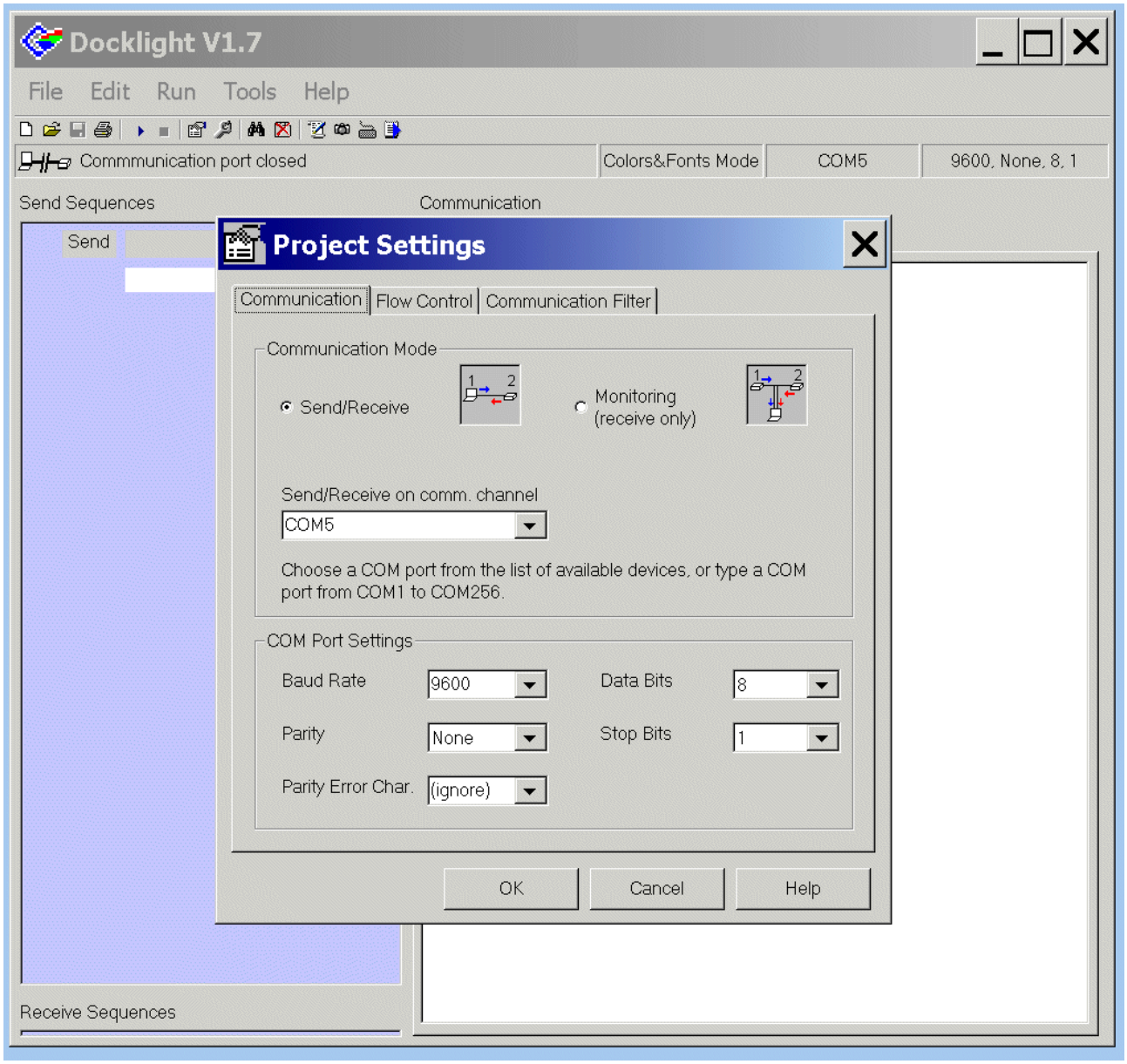
Click: Project Settings



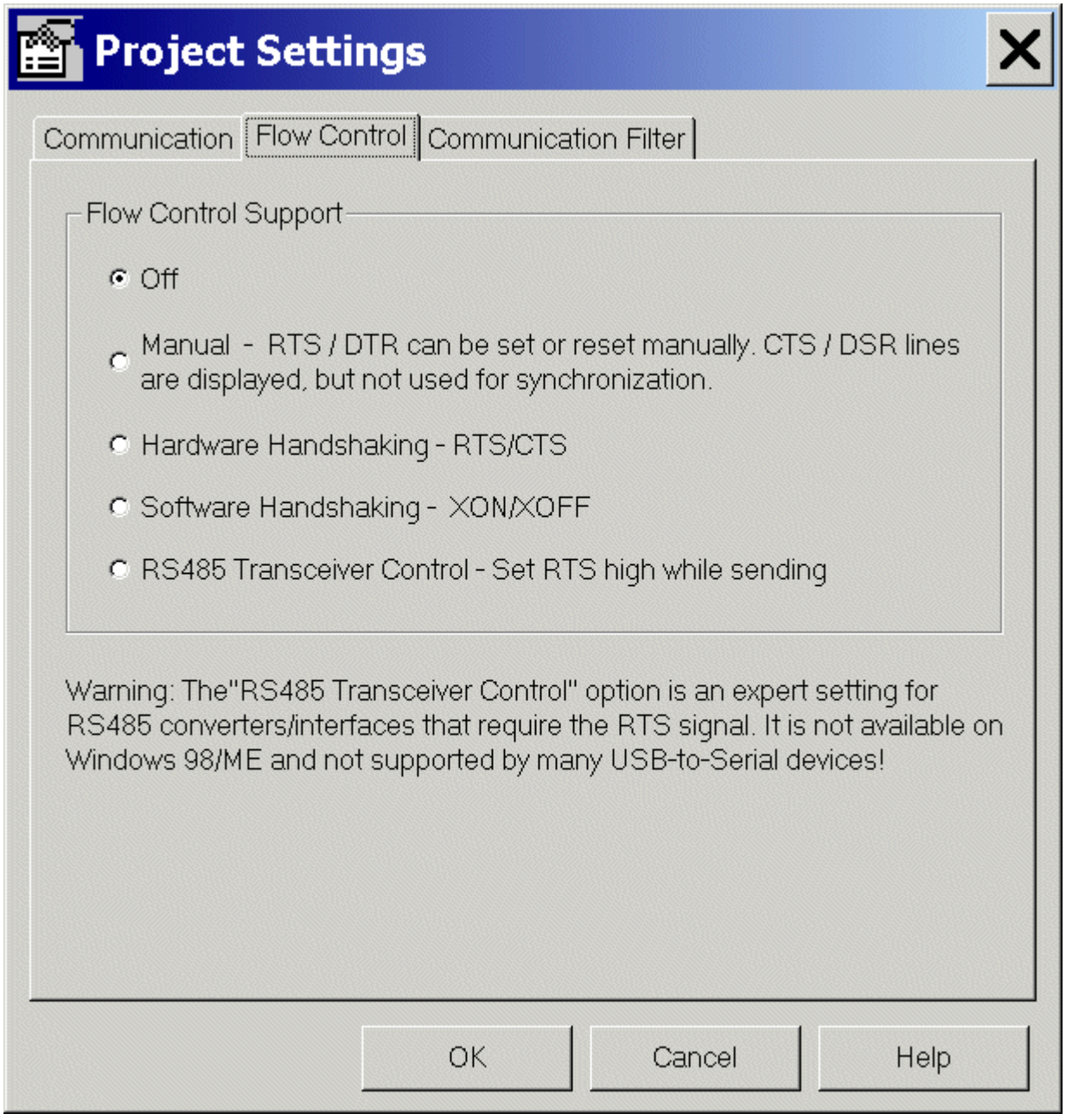
Project Settings:
Communication: Communication Mode: click ⊙ Send/Receive

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

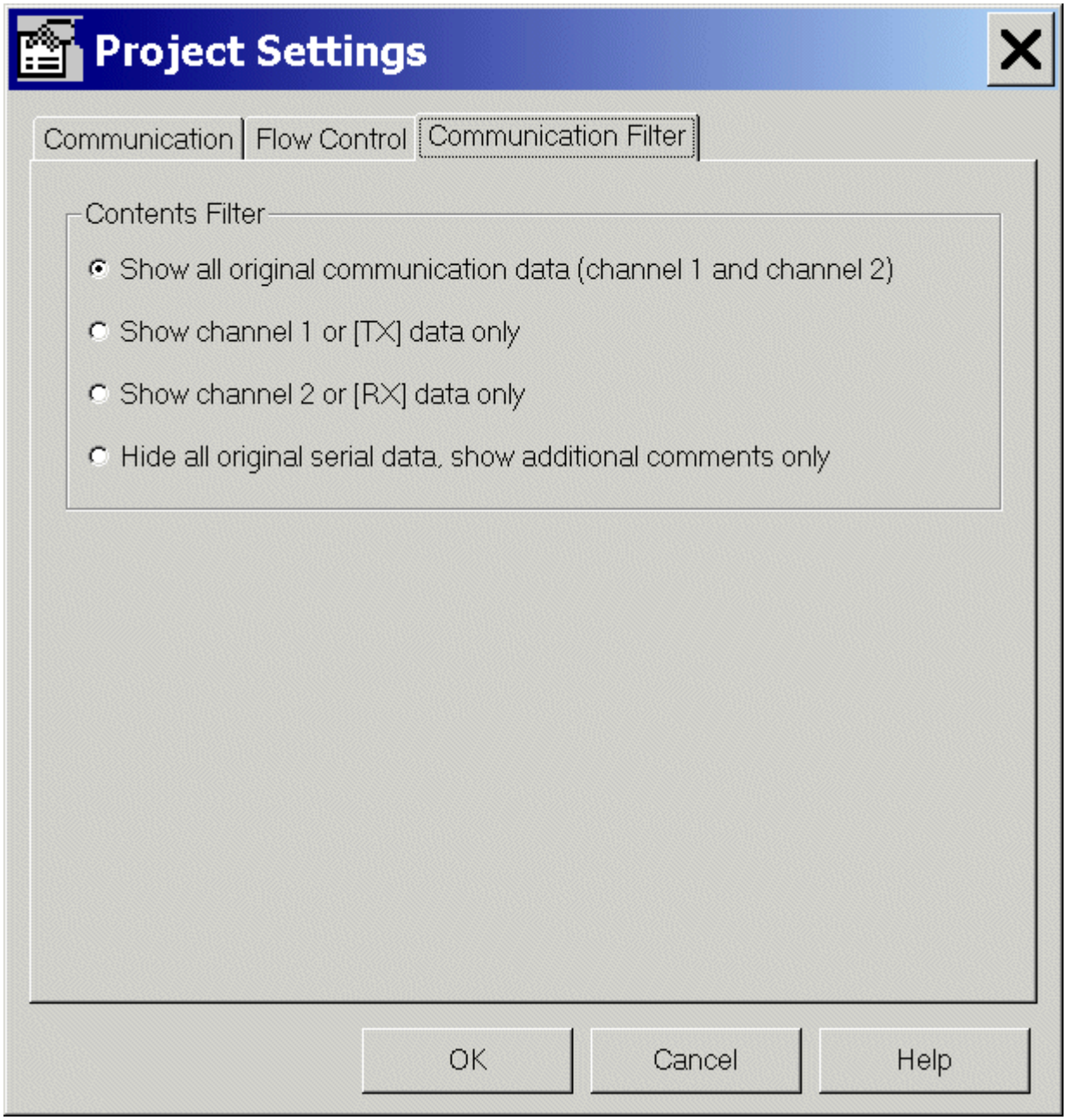
Project Settings: Communication: COM Port Settings: Baud Rate: select 9600
Project Settings: Communication: COM Port Settings: Parity: select None
Project Settings: Communication: COM Port Settings: Parity Error Char.: select (ignore)
Project Settings: Communication: COM Port Settings: Data Bits: select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



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

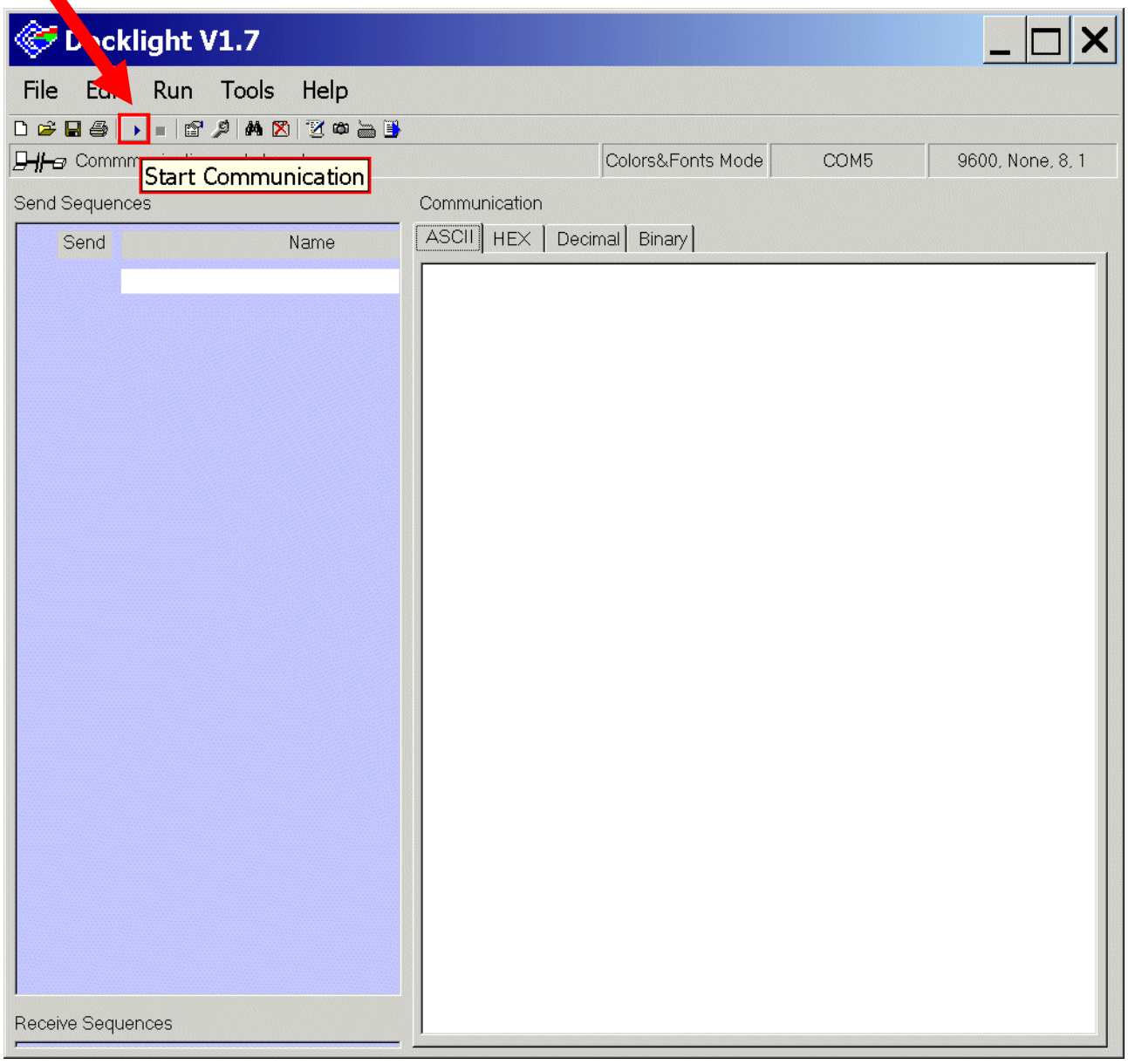


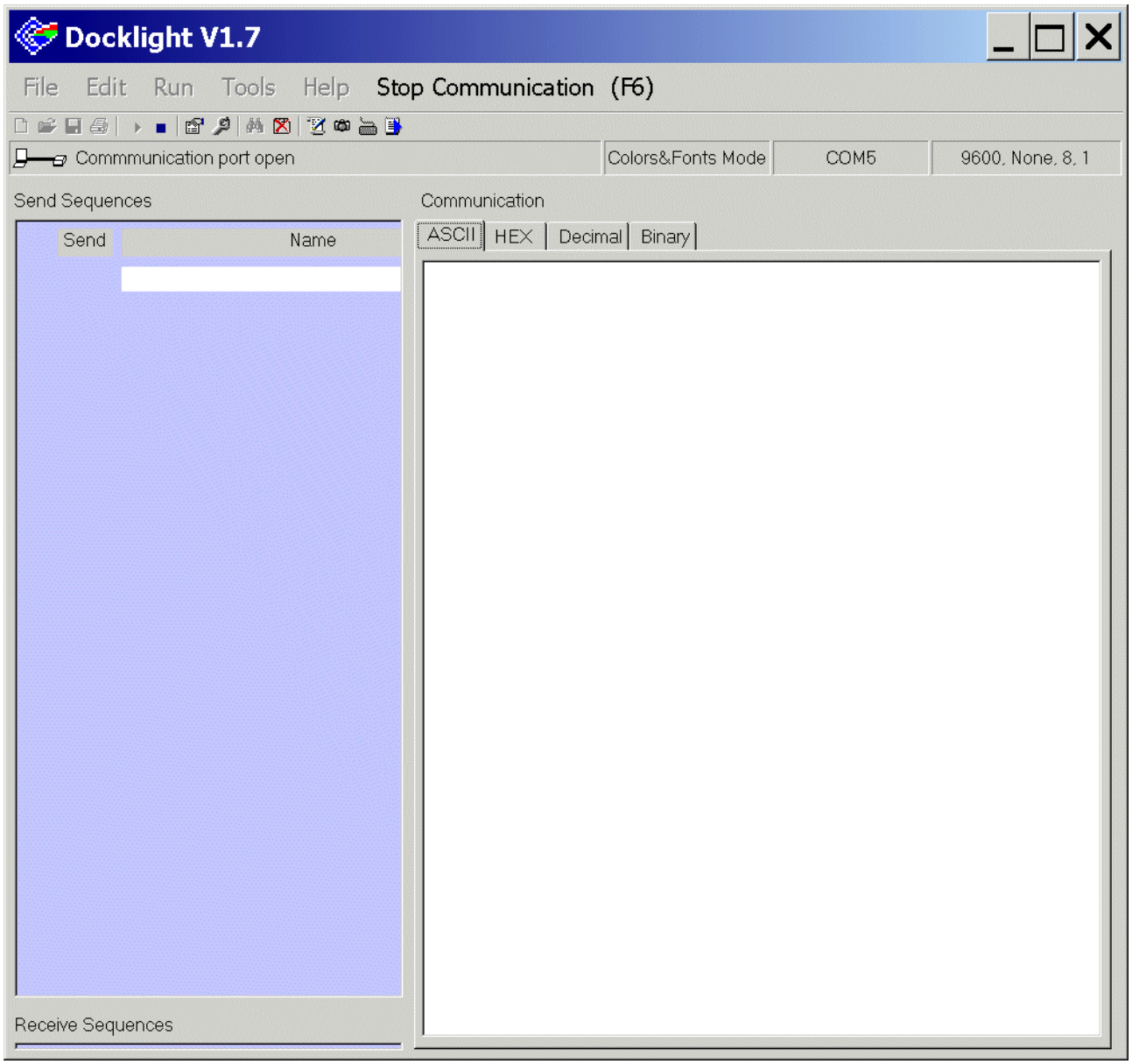
Project Settings:
Communication Filter: Contents Filter: click ⊙ Show all original communication data



OK

Click: 

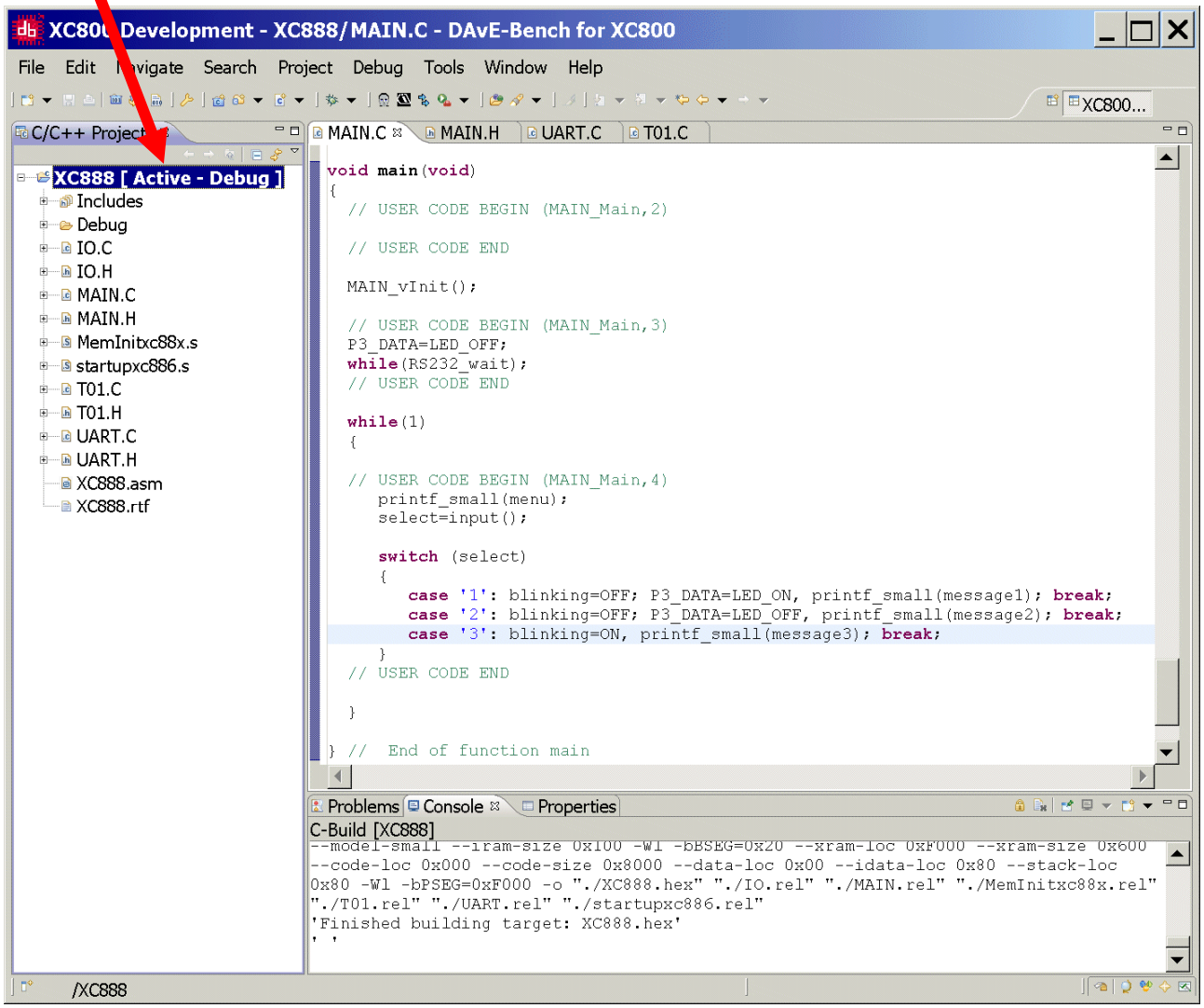

Run
Tools
Help
Start Communication
Colors&Fonts Mode
COM5
9600, None, 8, 1
Send Sequences
Communication
Send
Name
ASCII
HEX
Decimal
Binary
Receive Sequences



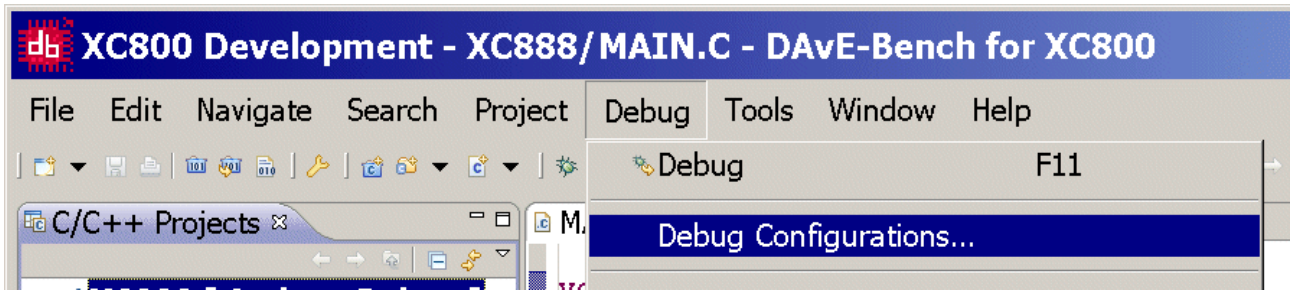
Note:
Docklight is now ready for serial communication!

Go back to DAvE Bench and configure the debugger:

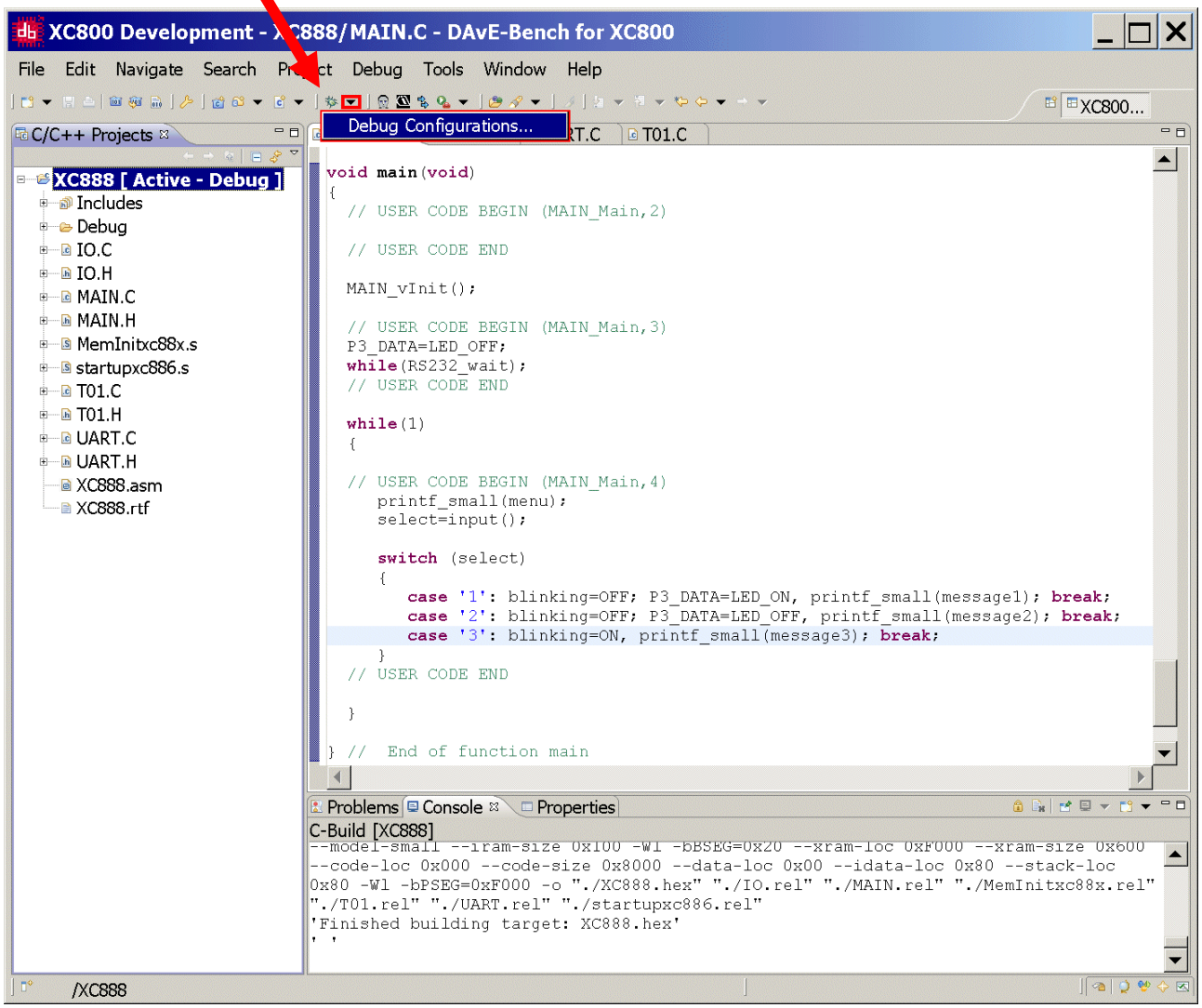
Click on **XC888 [Active - Debug]**:



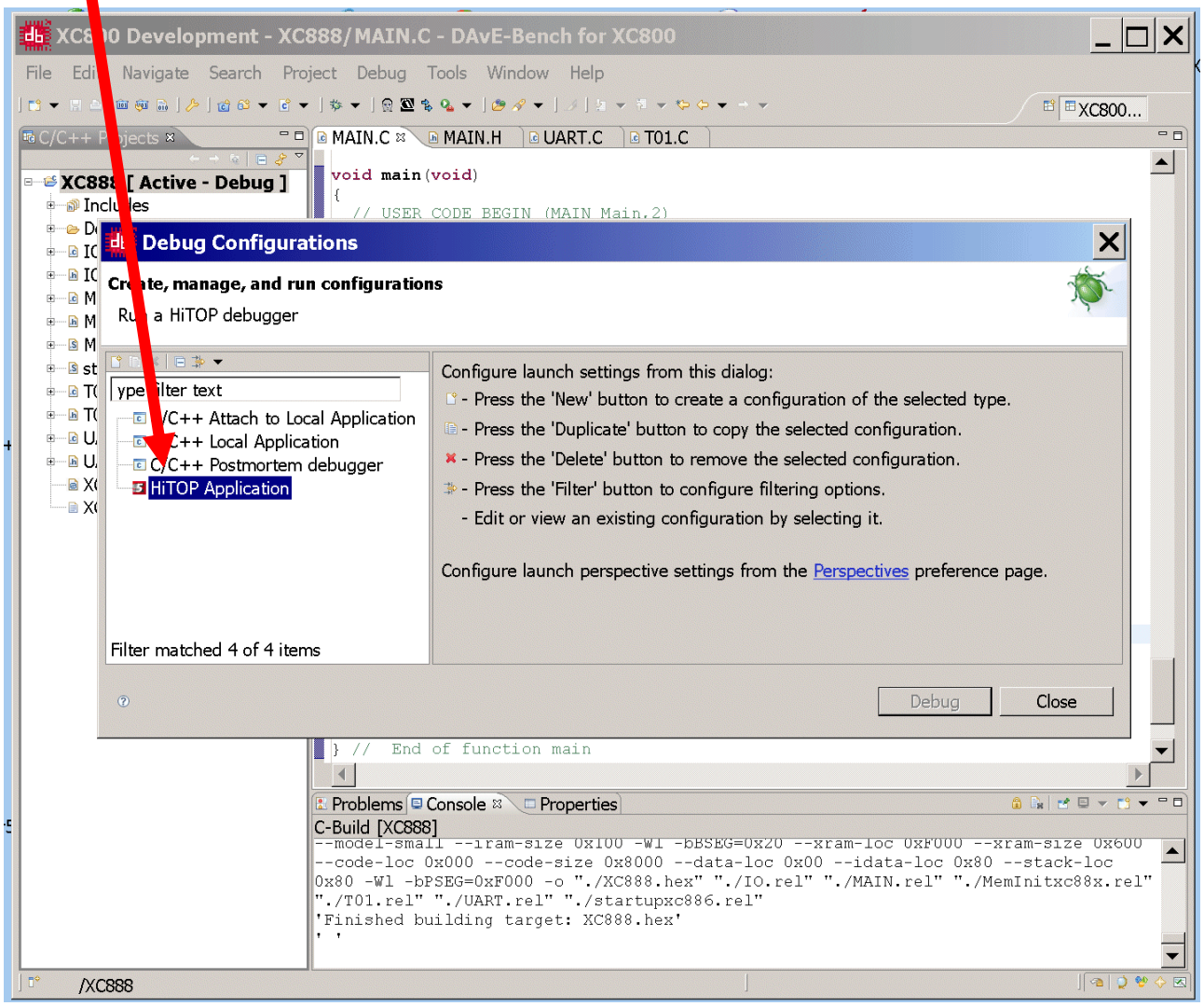
Debug – Debug Configurations…



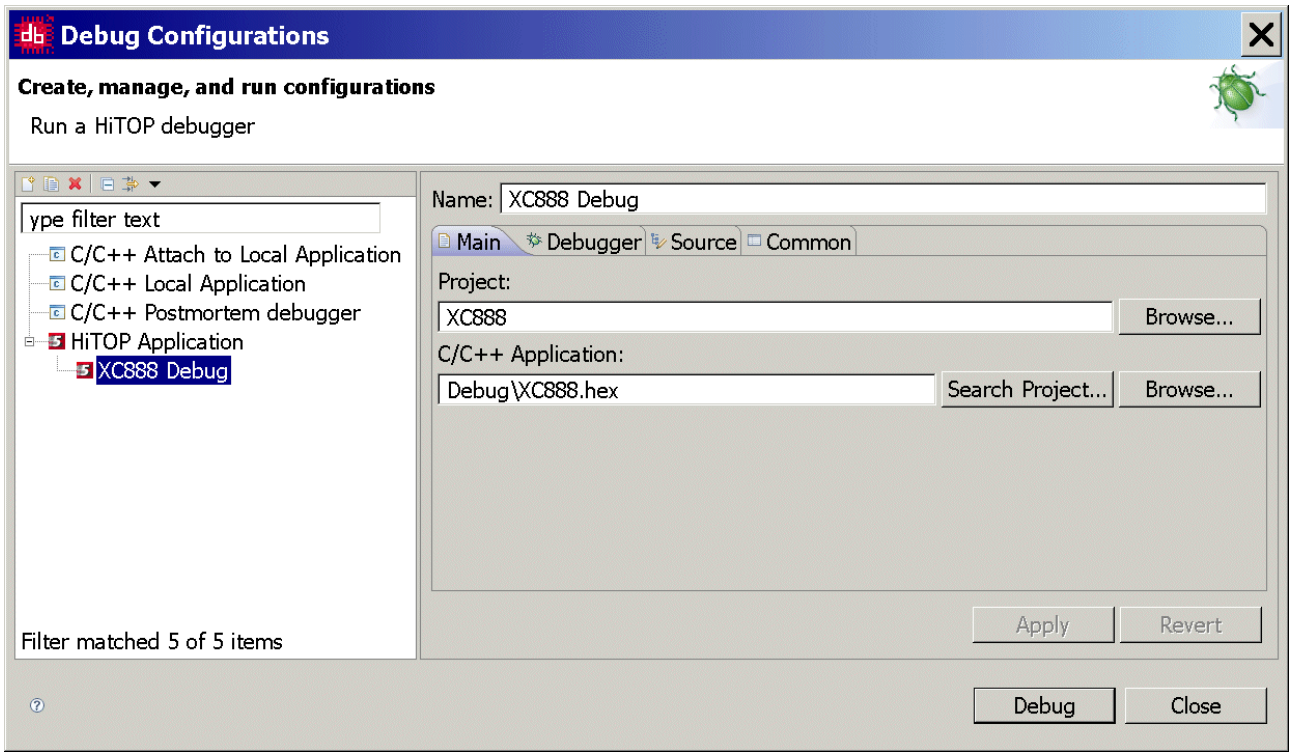
or click  Debug Configurations…

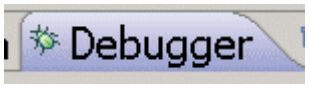


Double click: HiTOP Application HiTOP Application:

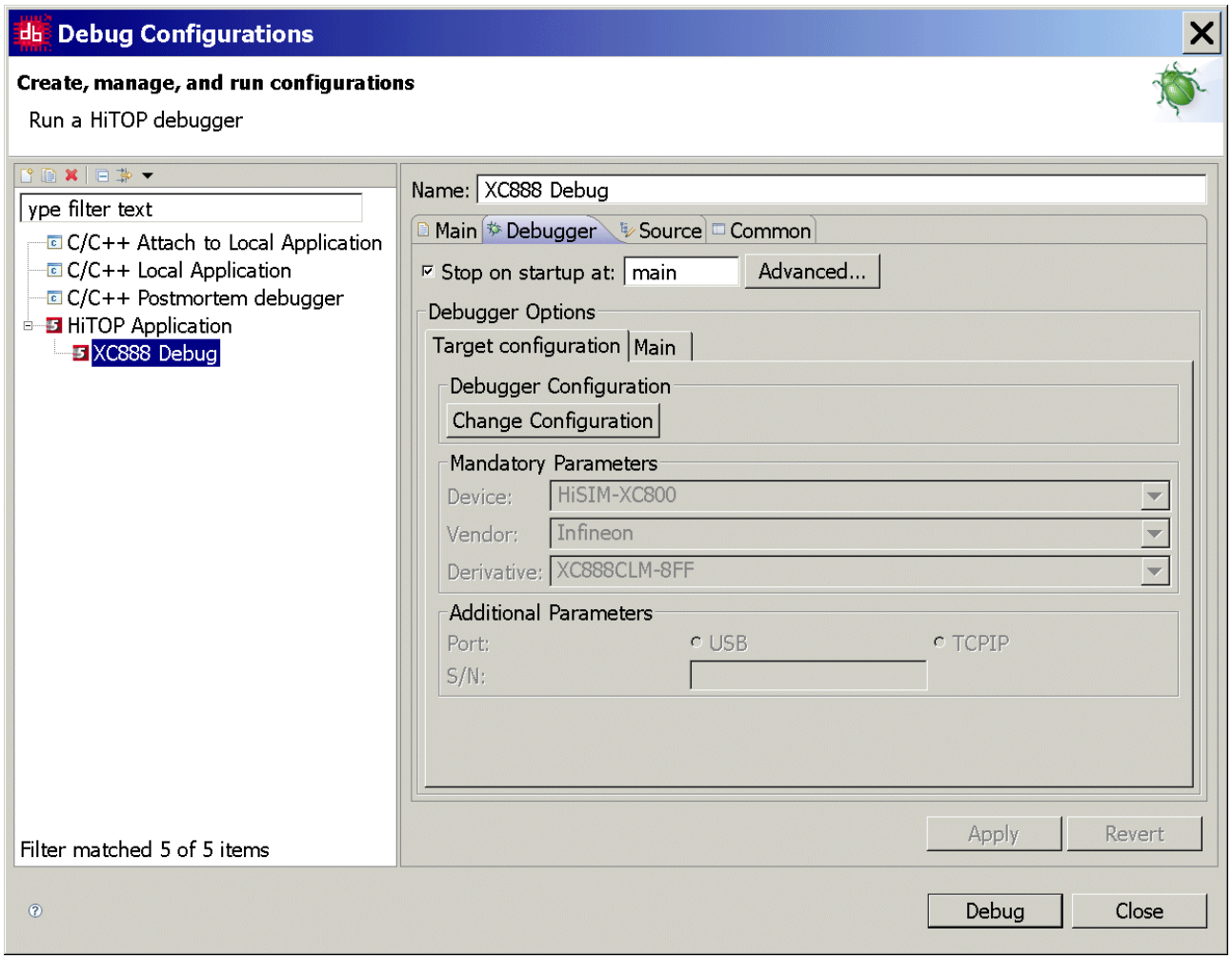



Main
Main:


Debug Configurations
Create, manage, and run configurations
Run a HiTOP debugger
ype filter text
C/C++ Attach to Local Application
C/C++ Local Application
C/C++ Postmortem debugger
HiTOP Application
XC888 Debug
Filter matched 5 of 5 items
Name: XC888 Debug
Main
Debugger
Source
Common
Project:
XC888
Browse...
C/C++ Application:
Debug\XC888.hex
Search Project...
Browse...
Apply
Revert
Debug
Close

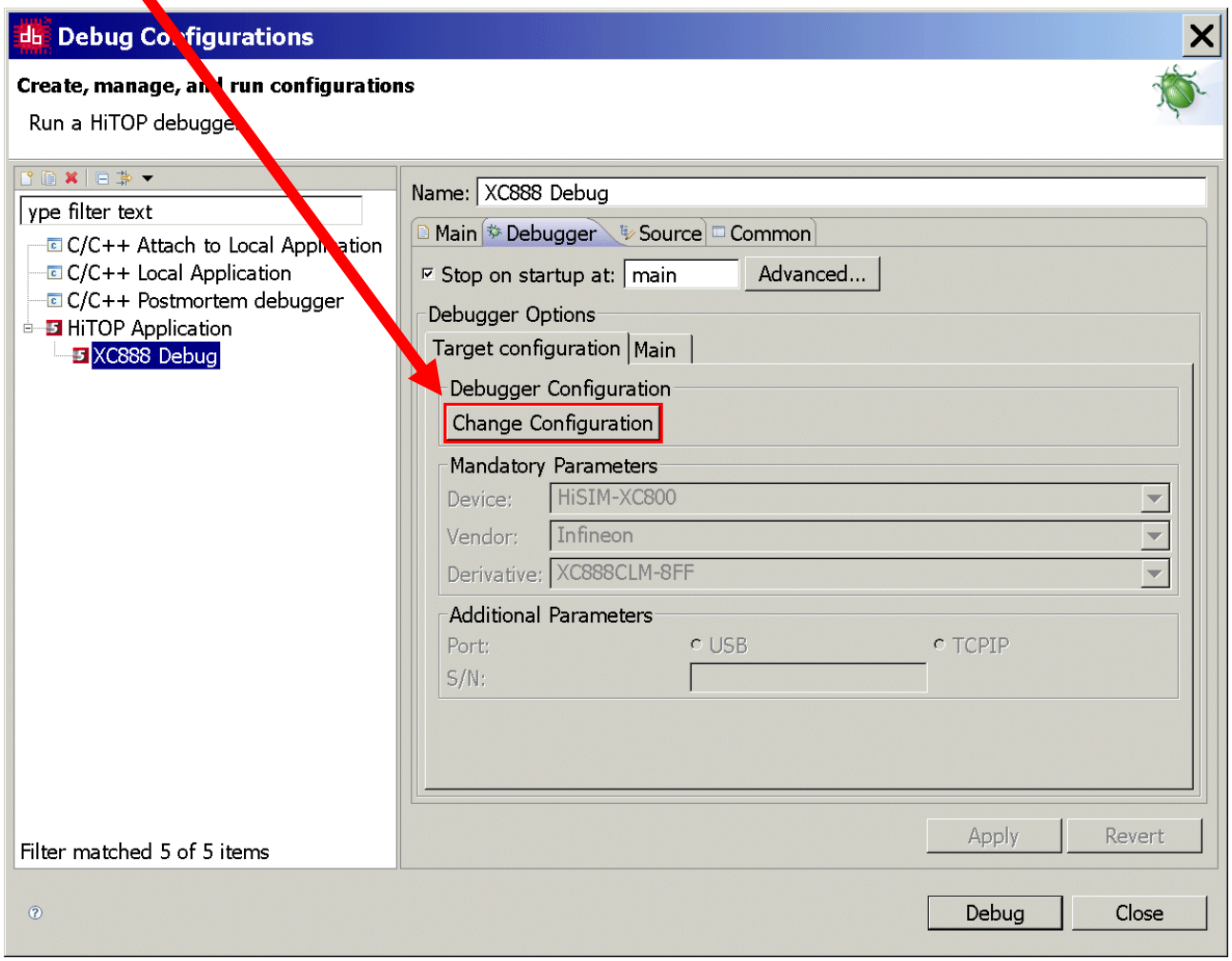


Debugger, Target configuration:

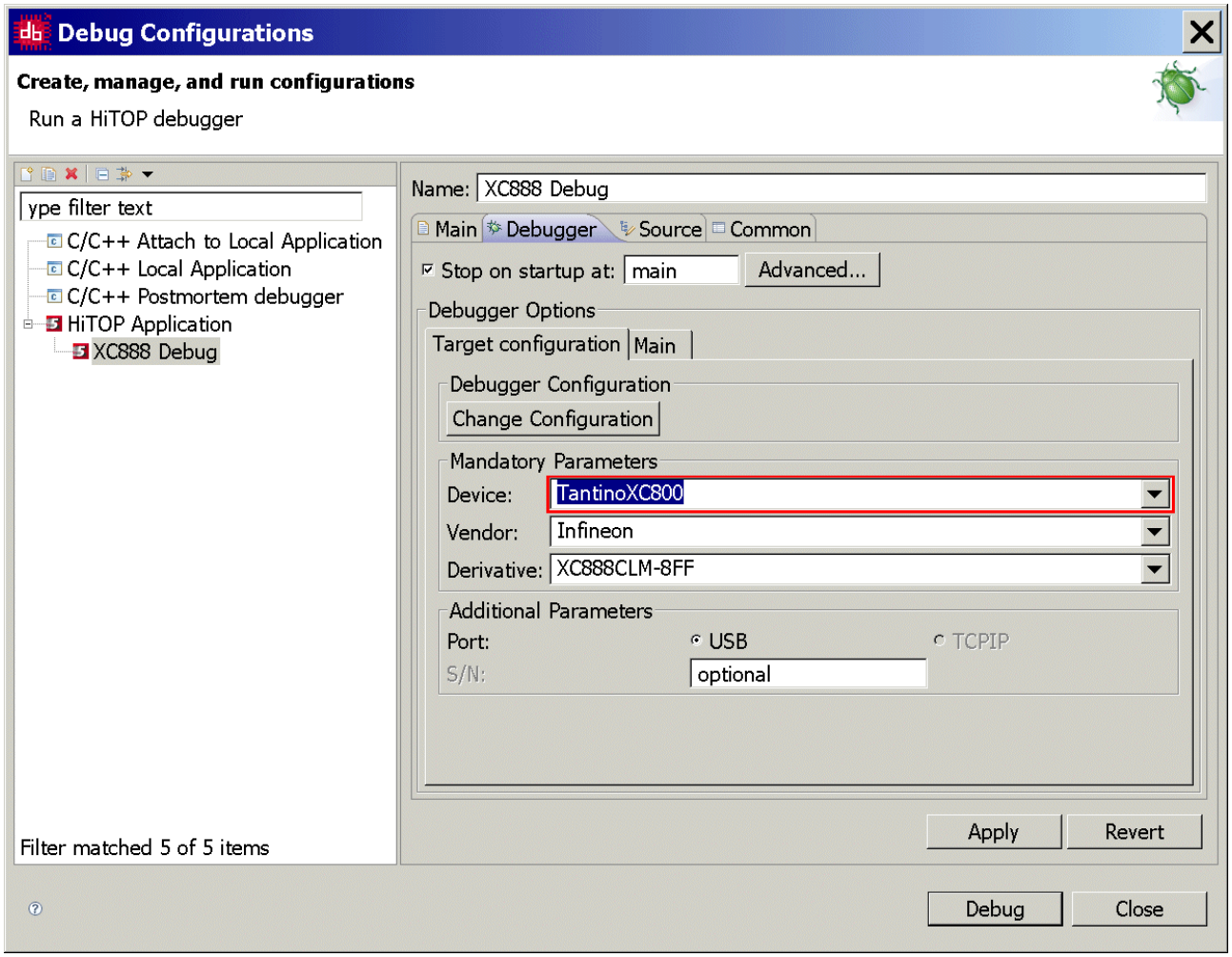


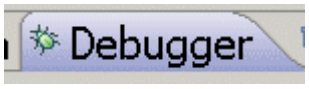
Click: 

Change Configuration:

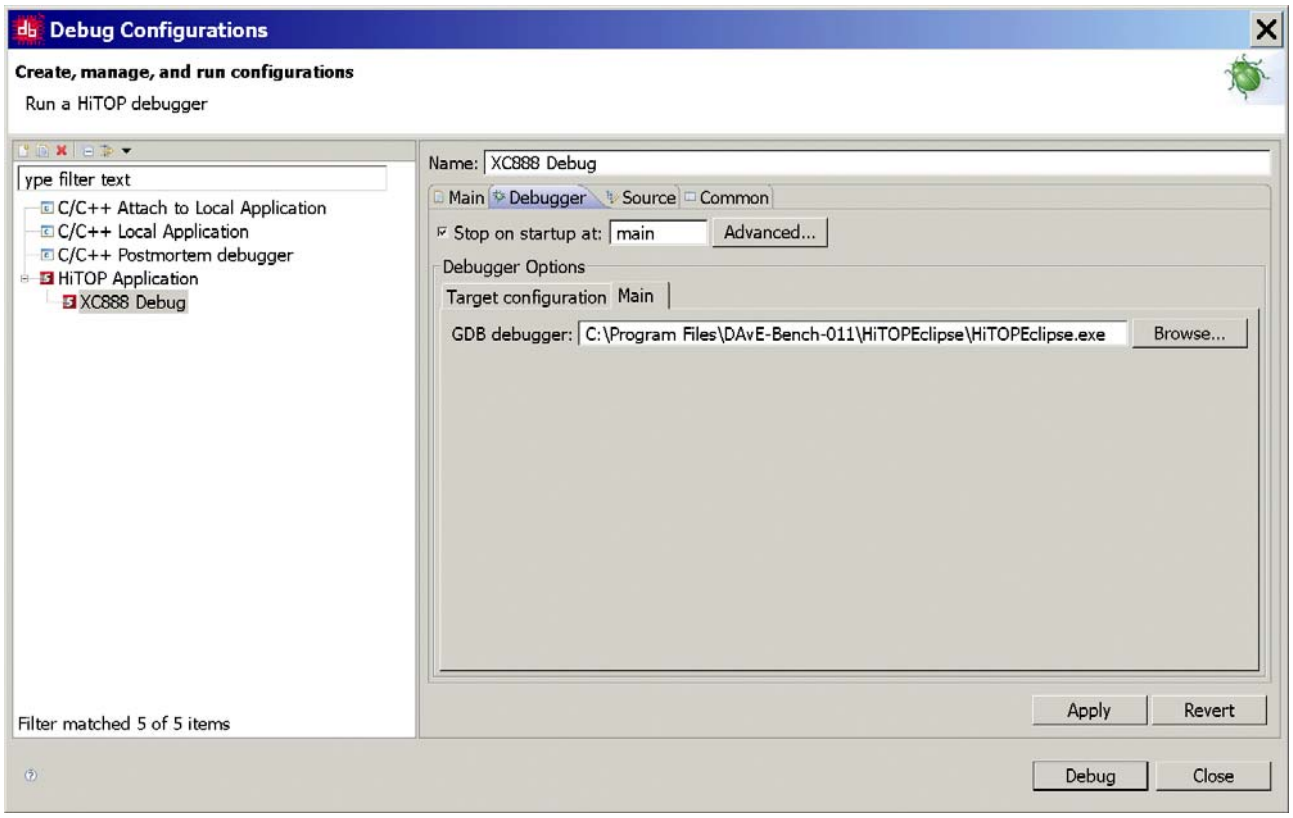


Mandatory Parameters: Device: select TantinoXC800

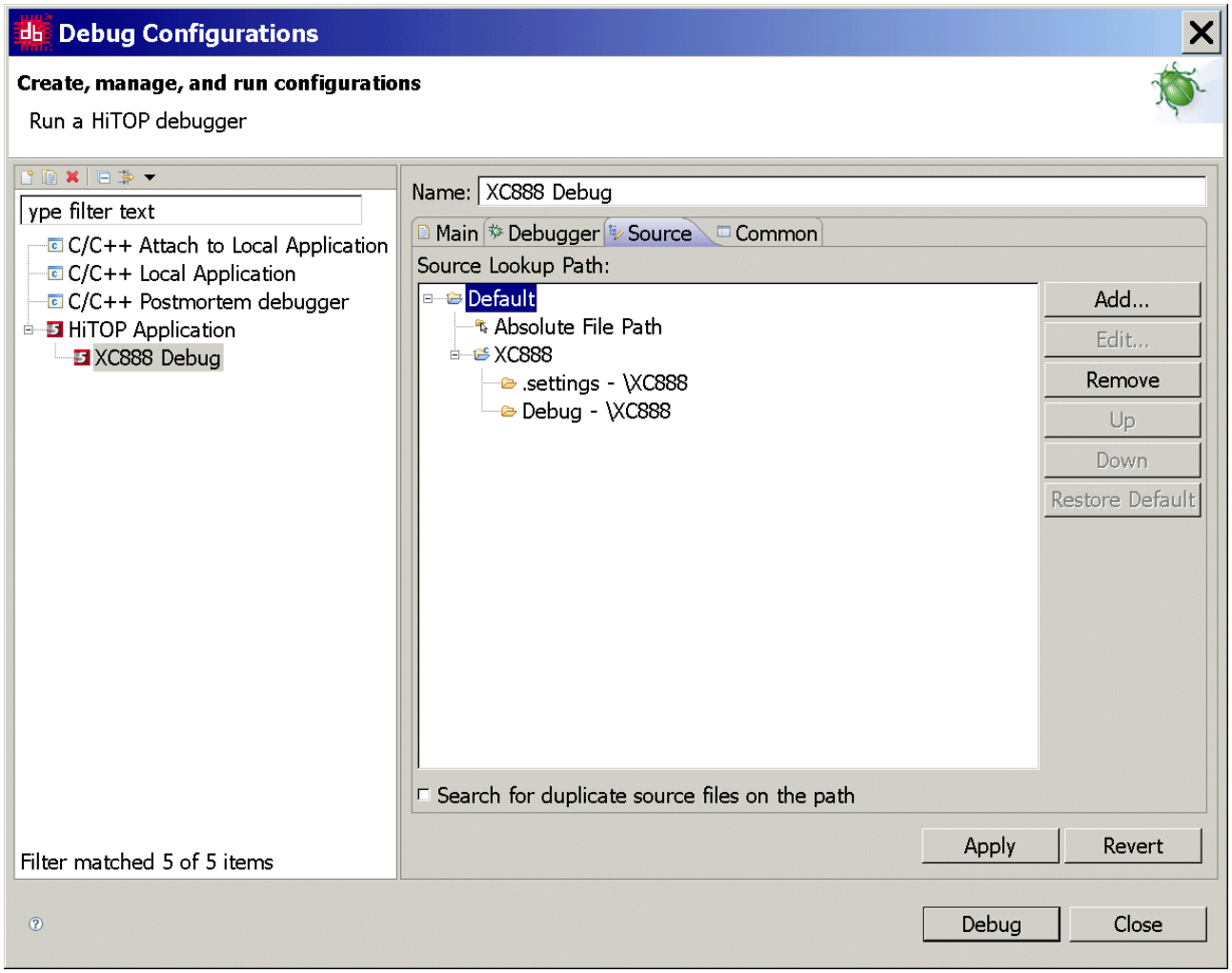




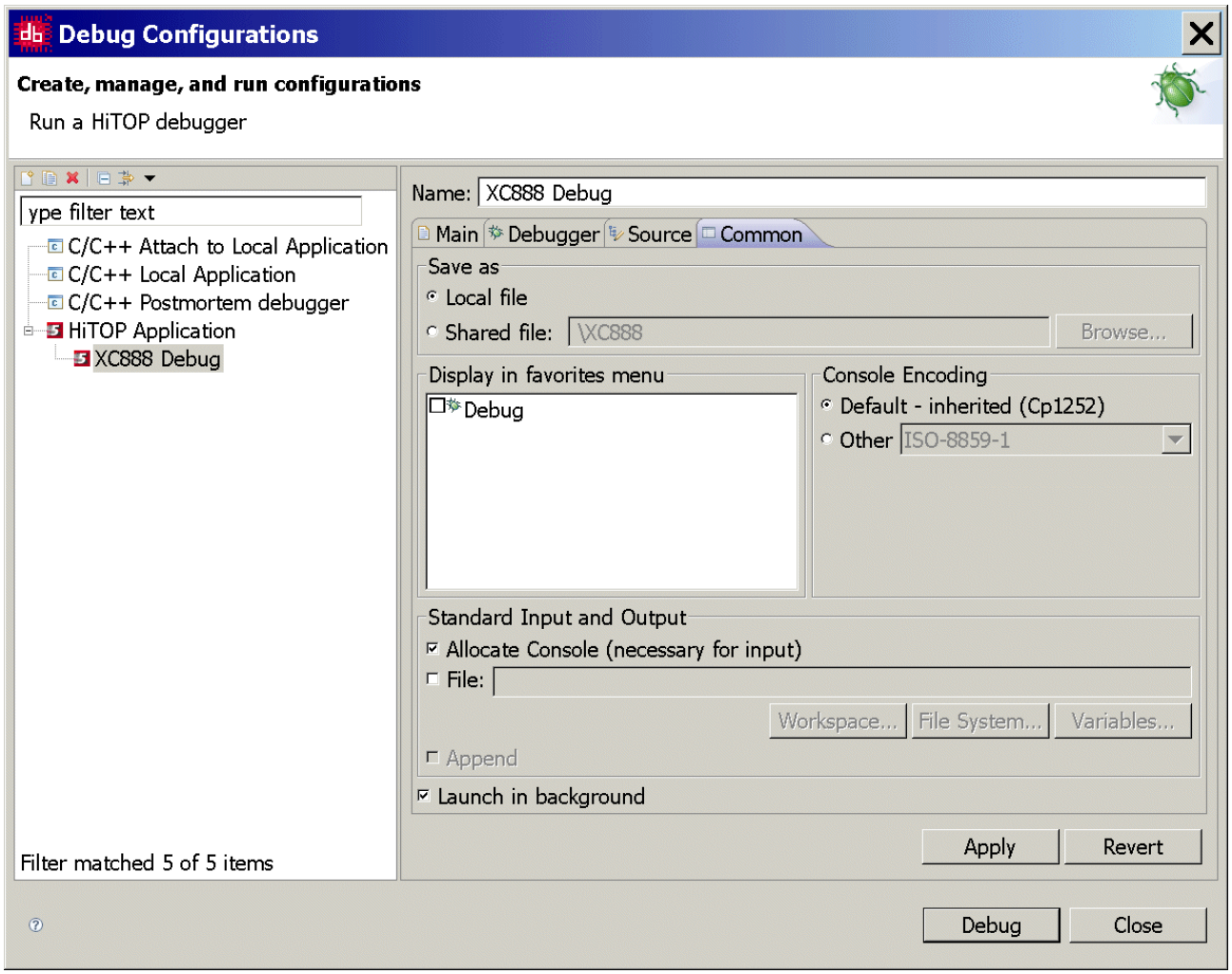
Debugger, Main: (do nothing)



Source: (do nothing)



Common: (do nothing)

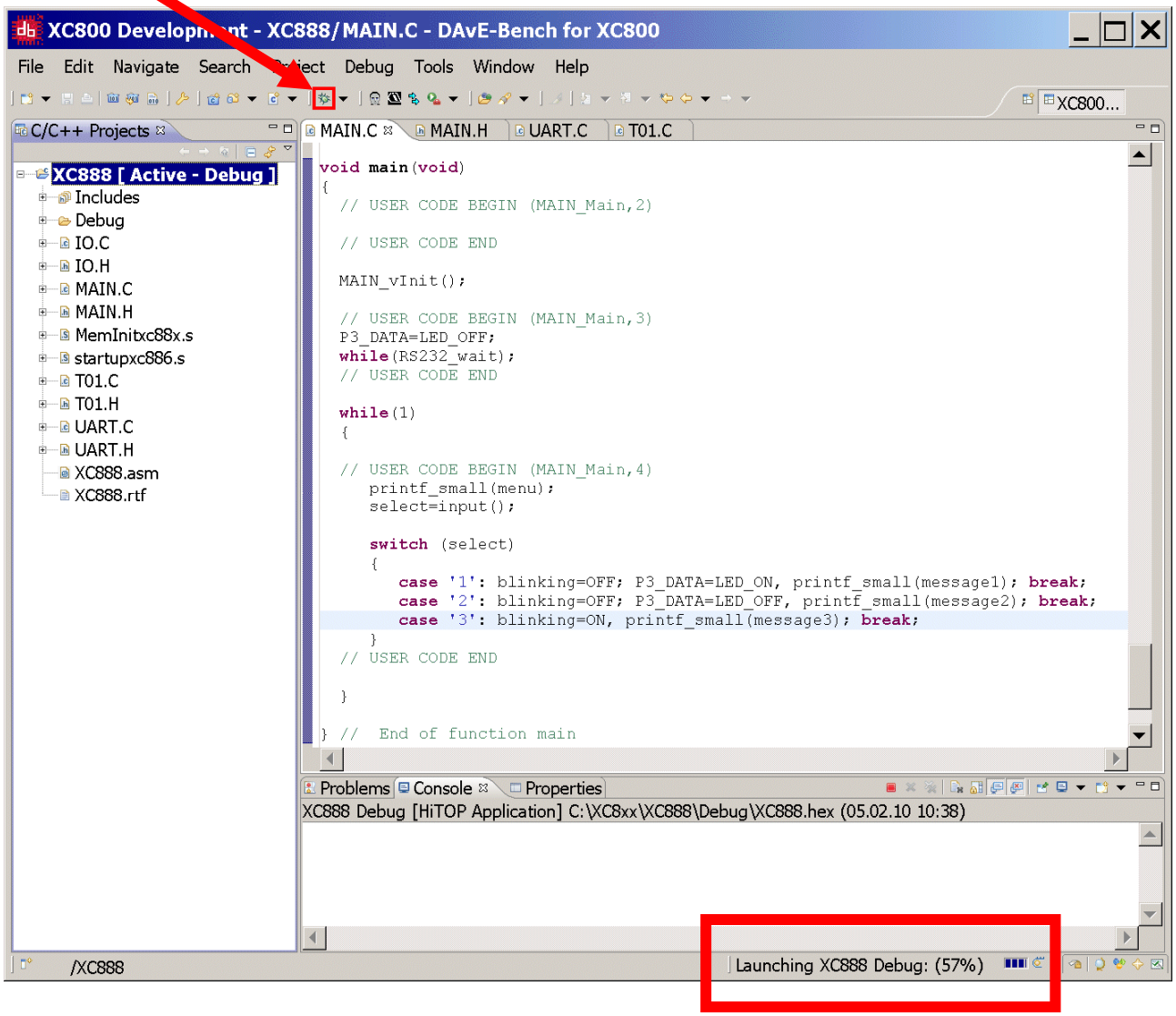


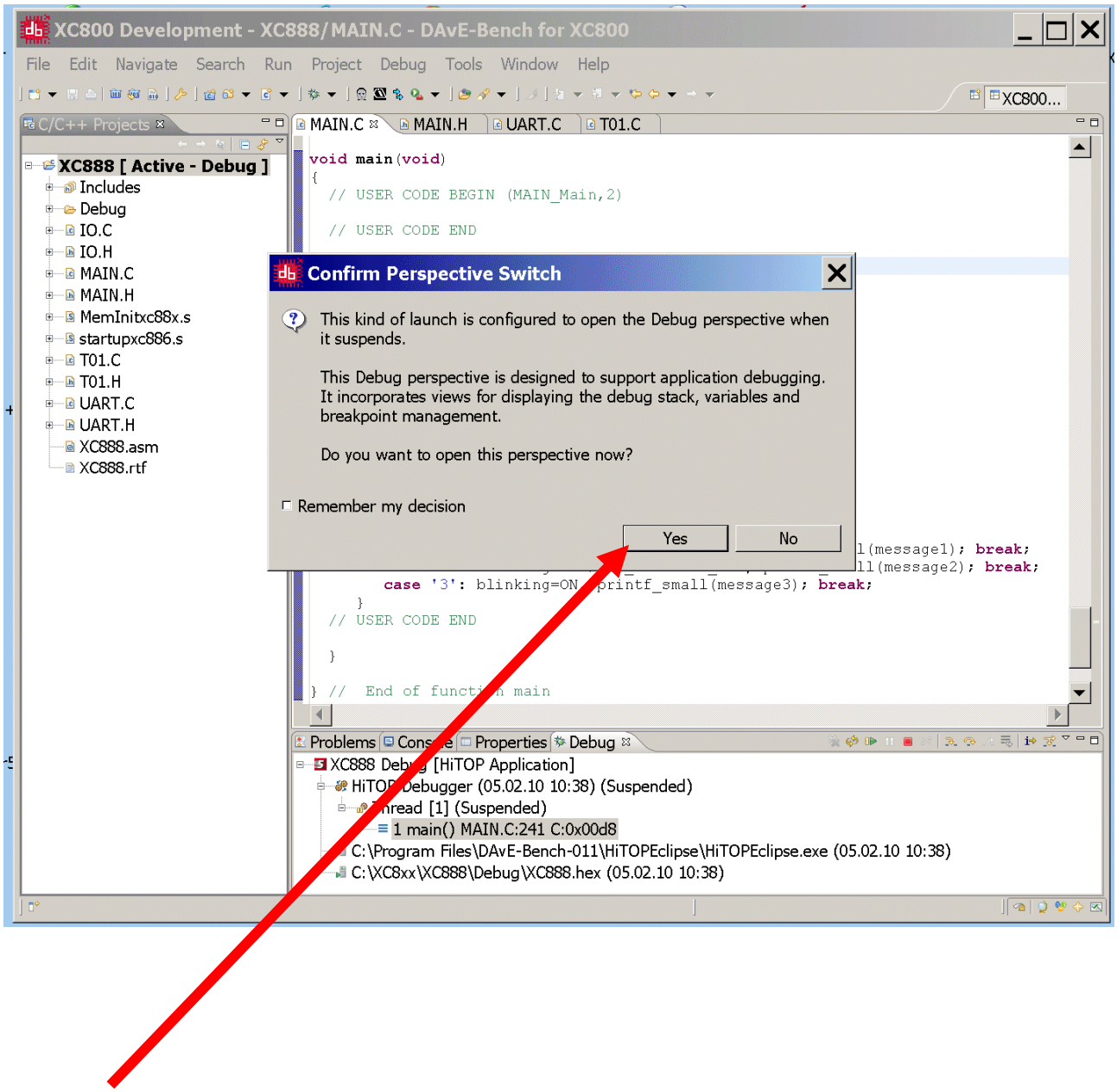
Apply

Close

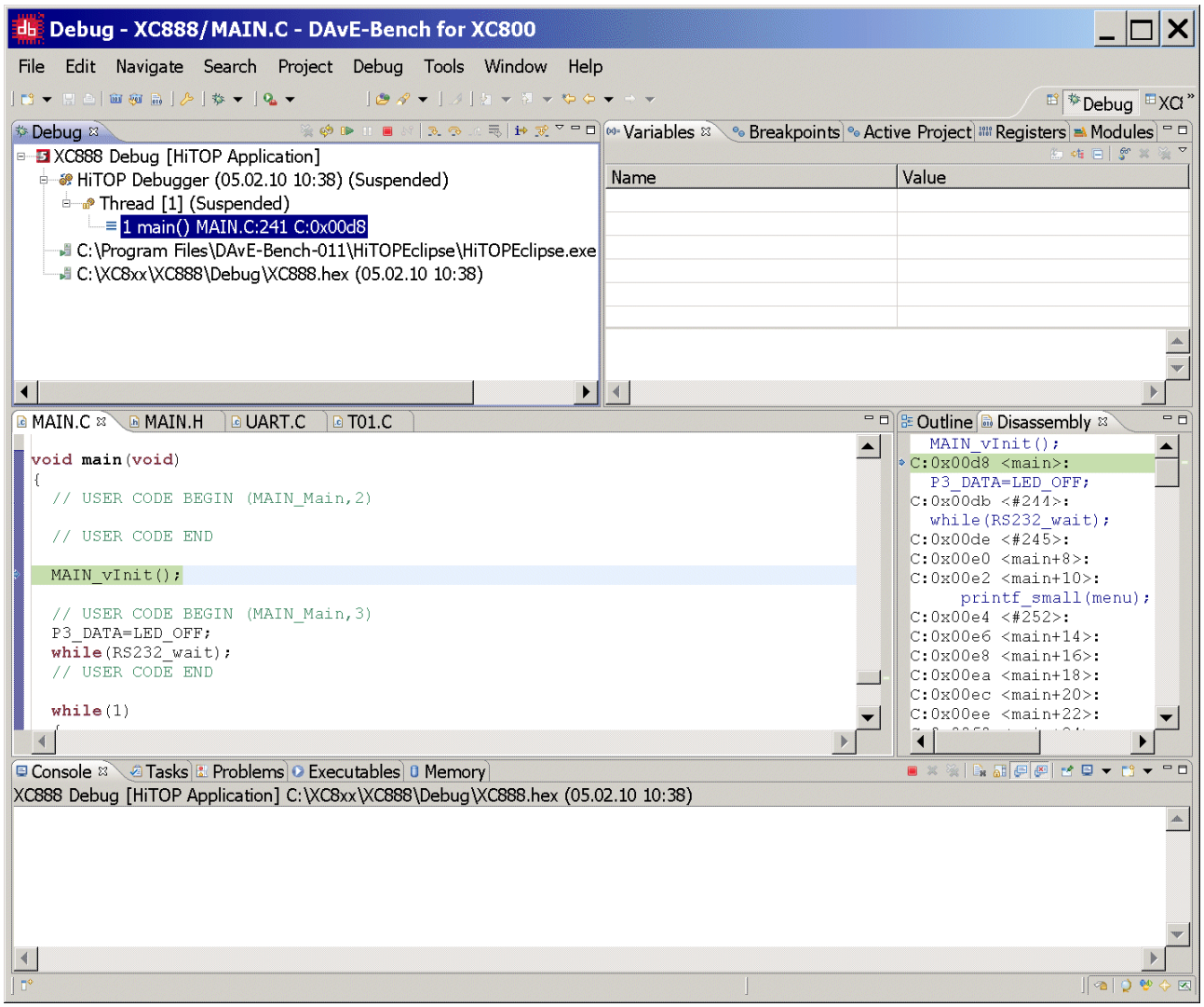
Start/Launch the debugger:

Click  :

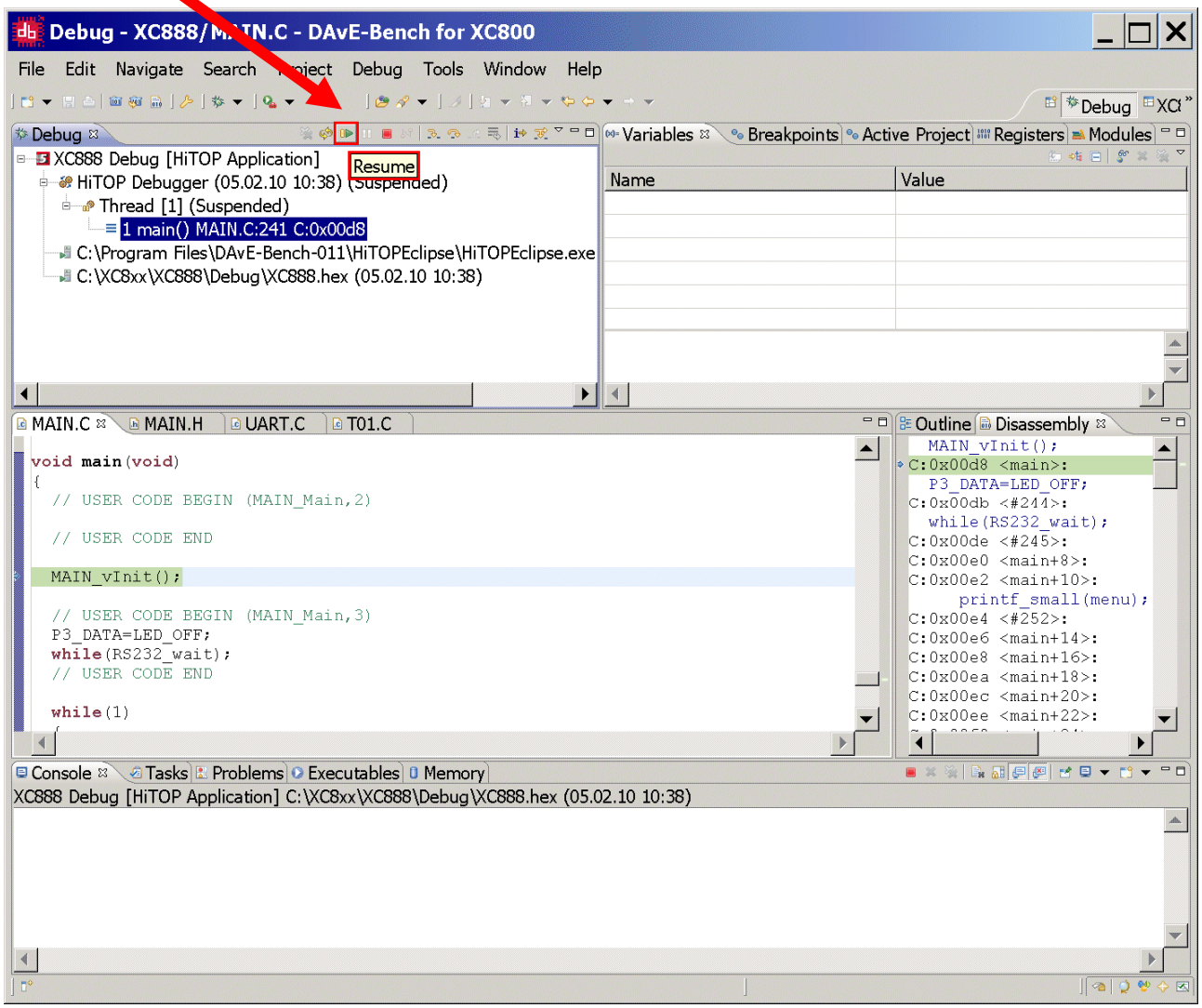



XC800 Development - XC888/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Run Project Debug Tools Window Help
C/C++ Projects
XC888 [Active - Debug]
Includes
Debug
IO.C
IO.H
MAIN.C
MAIN.H
MemInitxc88x.s
startupxc886.s
T01.C
T01.H
UART.C
UART.H
XC888.asm
XC888.rtf
MAIN.C
MAIN.H
UART.C
T01.C
void main(void)
{
// USER CODE BEGIN (MAIN_Main,2)
// USER CODE END
Confirm Perspective Switch
This kind of launch is configured to open the Debug perspective when it suspends.
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.
Do you want to open this perspective now?
Remember my decision
Yes
No
case '3': blinking=ON; printf_small(message3); break;
// USER CODE END
Problems
Properties
Debug
HiTOP Debugger (05.02.10 10:38) (Suspended)
Thread [1] (Suspended)
1 main() MAIN.C:241 C:0x00d8
C:\Program Files\DAvE-Bench-011\HiTOPEclipse\HiTOPEclipse.exe (05.02.10 10:38)
C:\XC8xx\XC888\Debug\XC888.hex (05.02.10 10:38)

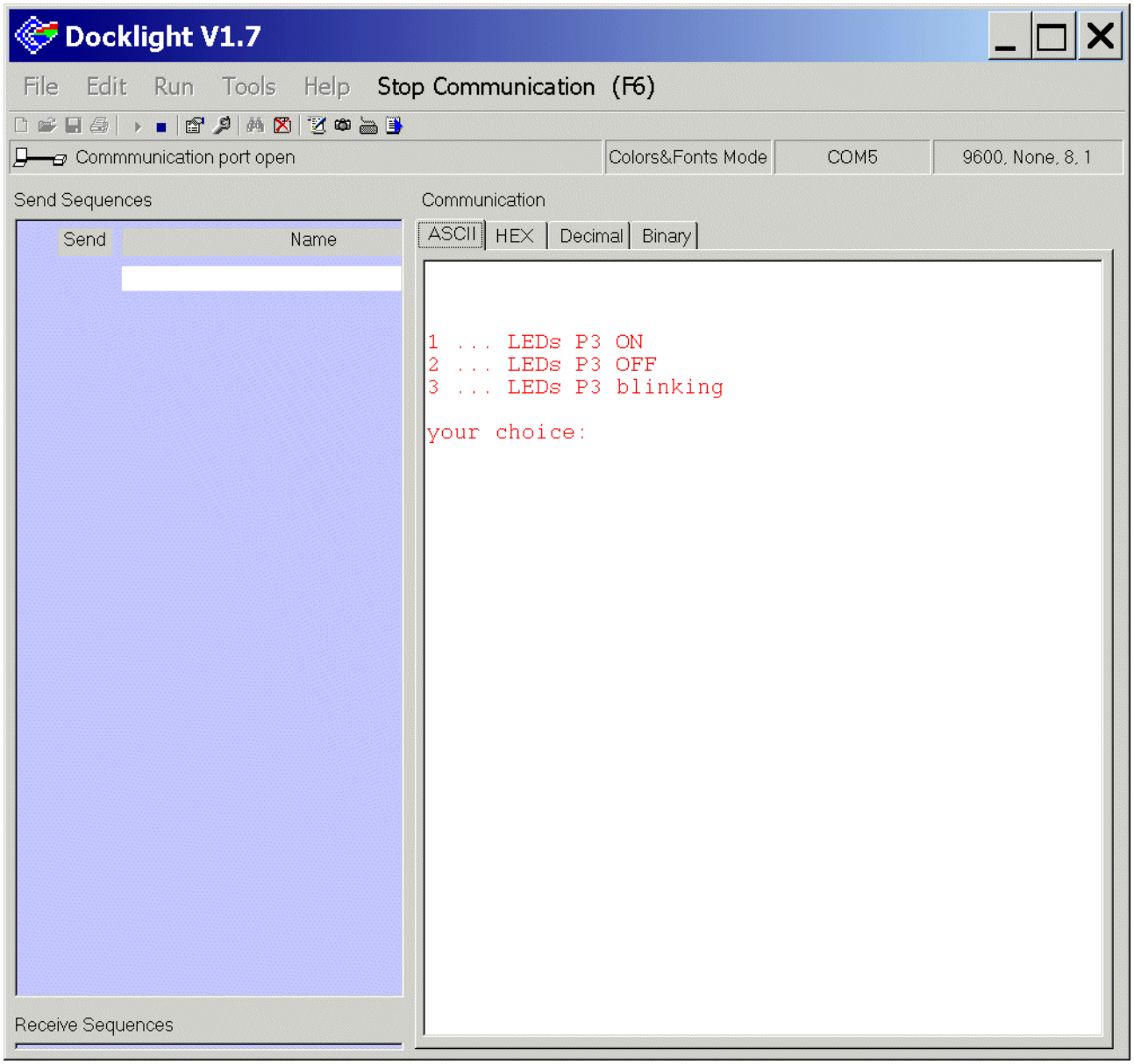
Click Yes


Debug - XC888/MAIN.C - DAvE-Bench for XC800
File Edit Navigate Search Project Debug Tools Window Help
Debug
XC888 Debug [HiTOP Application]
HiTOP Debugger (05.02.10 10:38) (Suspended)
Thread [1] (Suspended)
1 main() MAIN.C:241 C:0x00d8
C:\Program Files\DAvE-Bench-011\HiTOPEclipse\HiTOPEclipse.exe
C:\XC8xx\XC888\Debug\XC888.hex (05.02.10 10:38)
Variables Breakpoints Active Project Registers Modules
Name Value
MAIN.C MAIN.H UART.C T01.C
void main(void)
{
// USER CODE BEGIN (MAIN_Main,2)
// USER CODE END
MAIN_vInit();
// USER CODE BEGIN (MAIN_Main,3)
P3_DATA=LED_OFF;
while(RS232_wait);
// USER CODE END
while(1)
Outline Disassembly
MAIN_vInit();
C:0x00d8 <main>:
P3_DATA=LED_OFF;
C:0x00db <#244>:
while(RS232_wait);
C:0x00de <#245>:
C:0x00e0 <main+8>:
C:0x00e2 <main+10>:
printf_small(menu);
C:0x00e4 <#252>:
C:0x00e6 <main+14>:
C:0x00e8 <main+16>:
C:0x00ea <main+18>:
C:0x00ec <main+20>:
C:0x00ee <main+22>:
Console Tasks Problems Executables Memory
XC888 Debug [HiTOP Application] C:\XC8xx\XC888\Debug\XC888.hex (05.02.10 10:38)

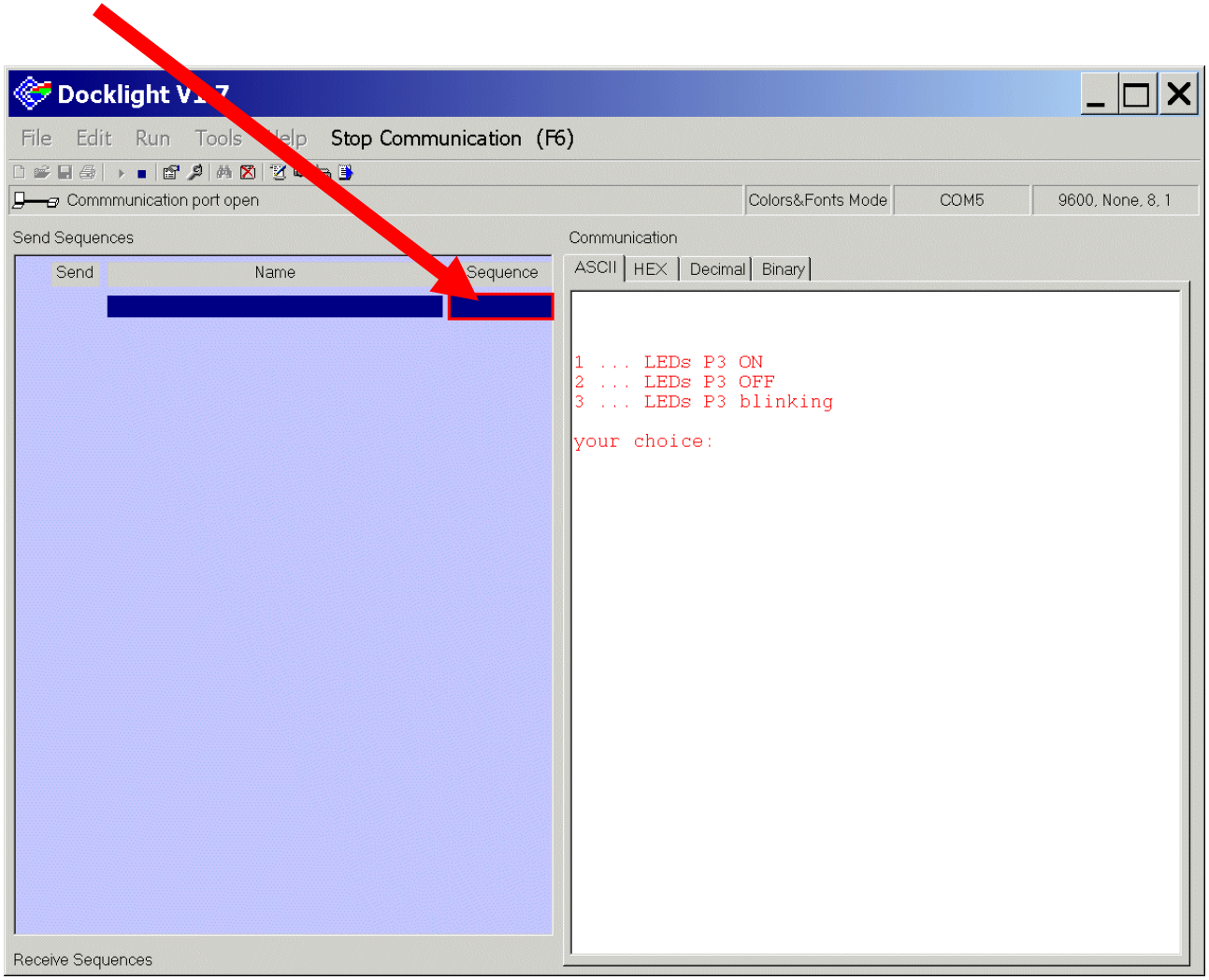
Click:  Resume



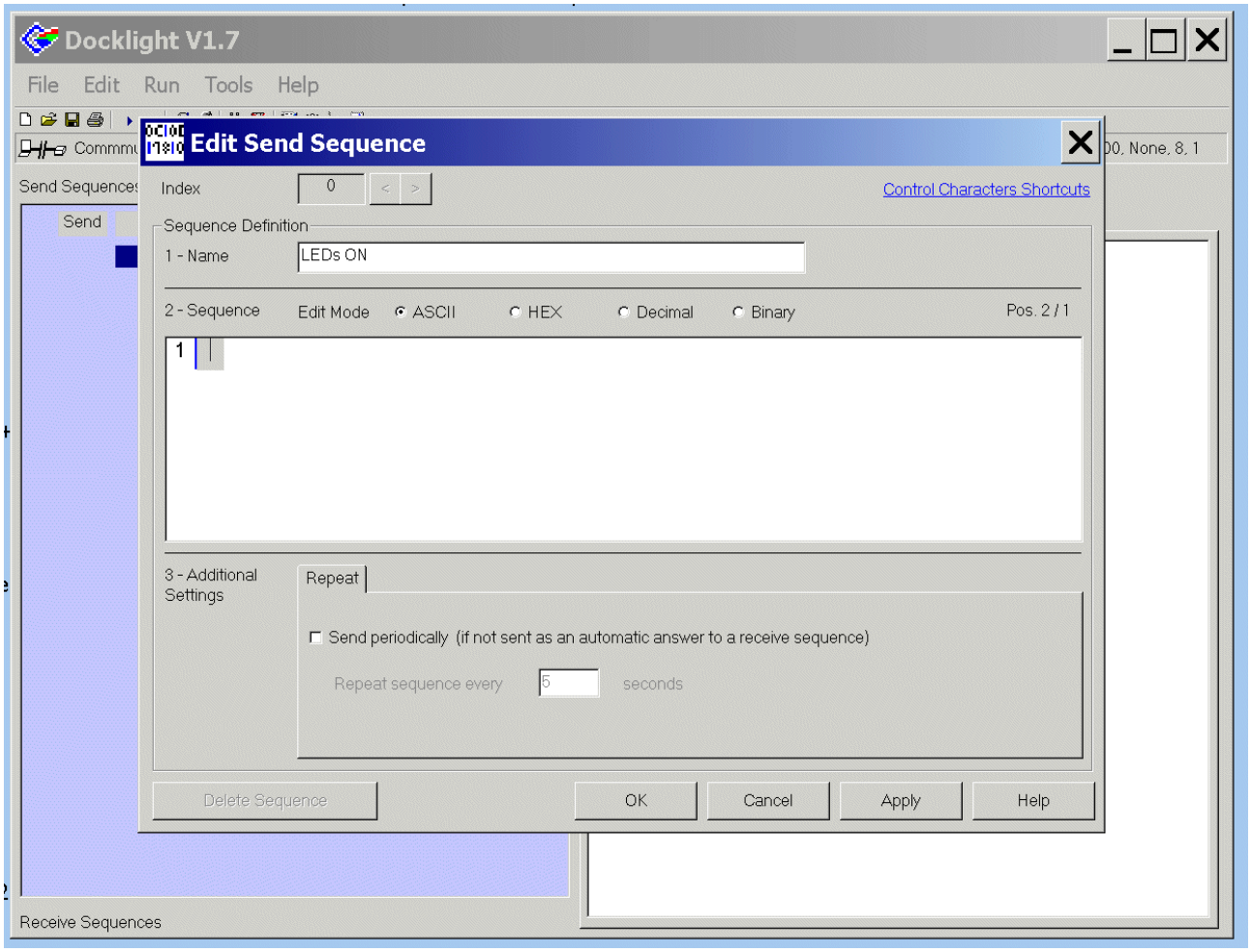
Go back to Docklight and see the result:



Double click inside the red box:

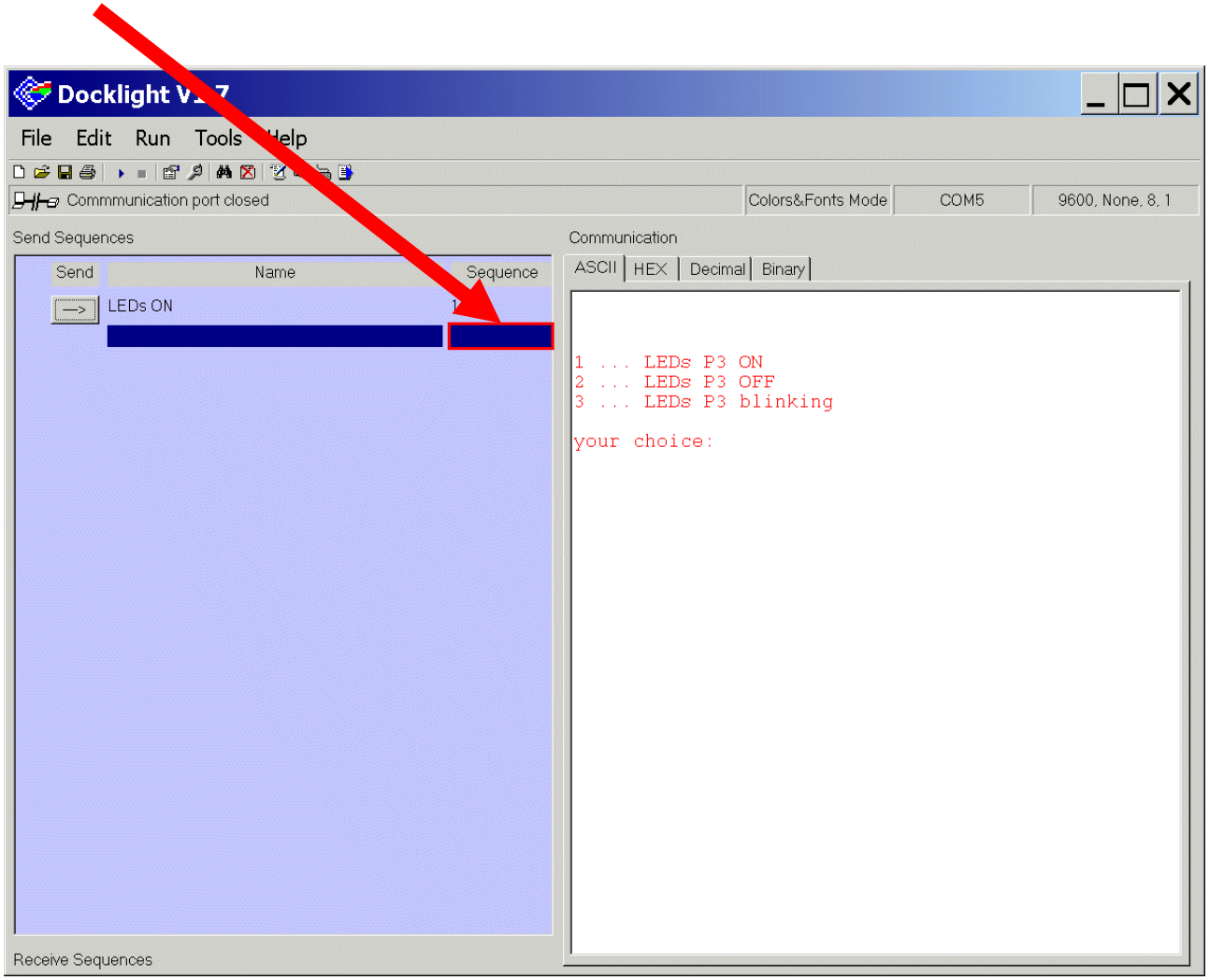


Edit Send Sequence: Sequence Definition: 1- Name: insert: LEDs ON
Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 1

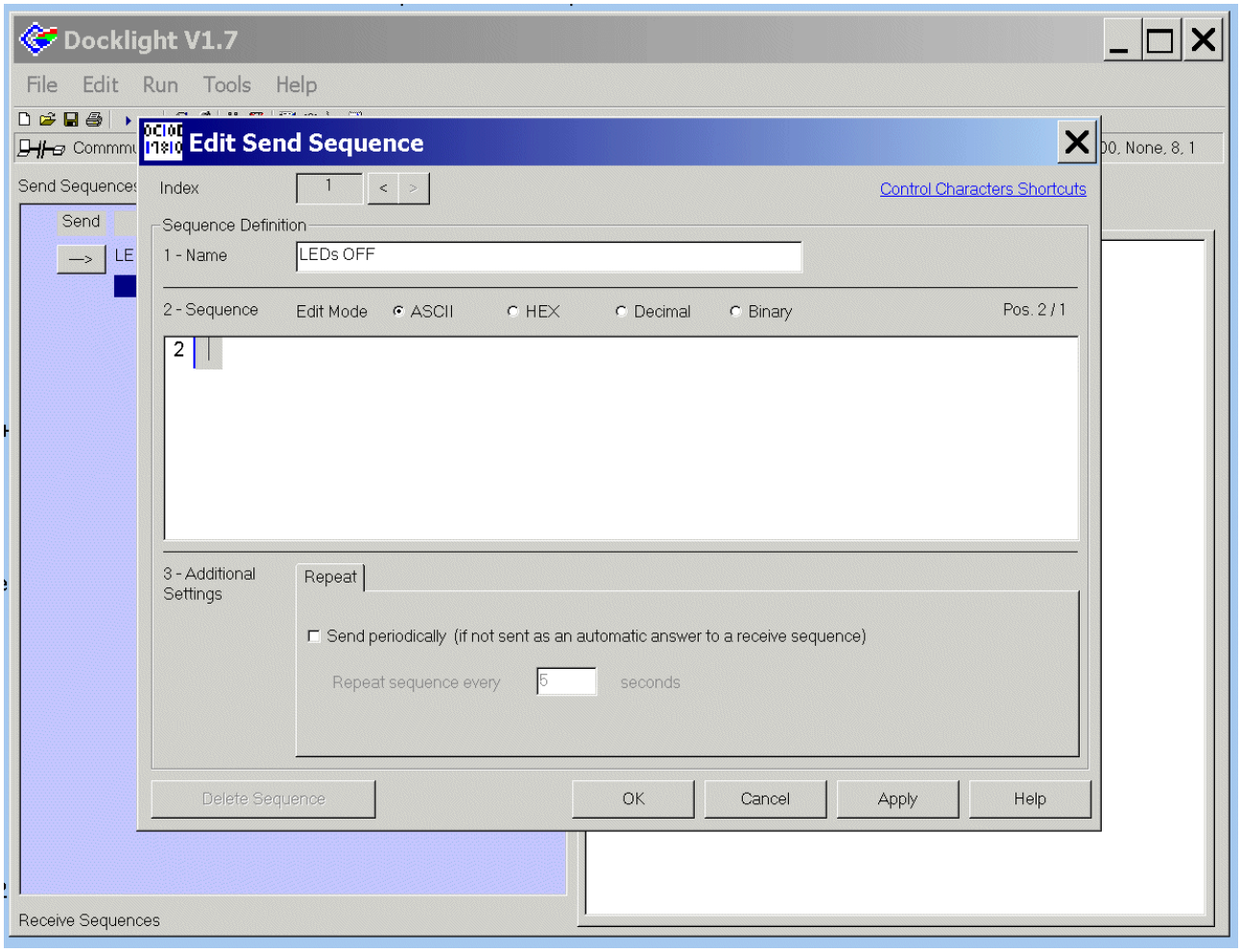


OK

Double click inside the red box:

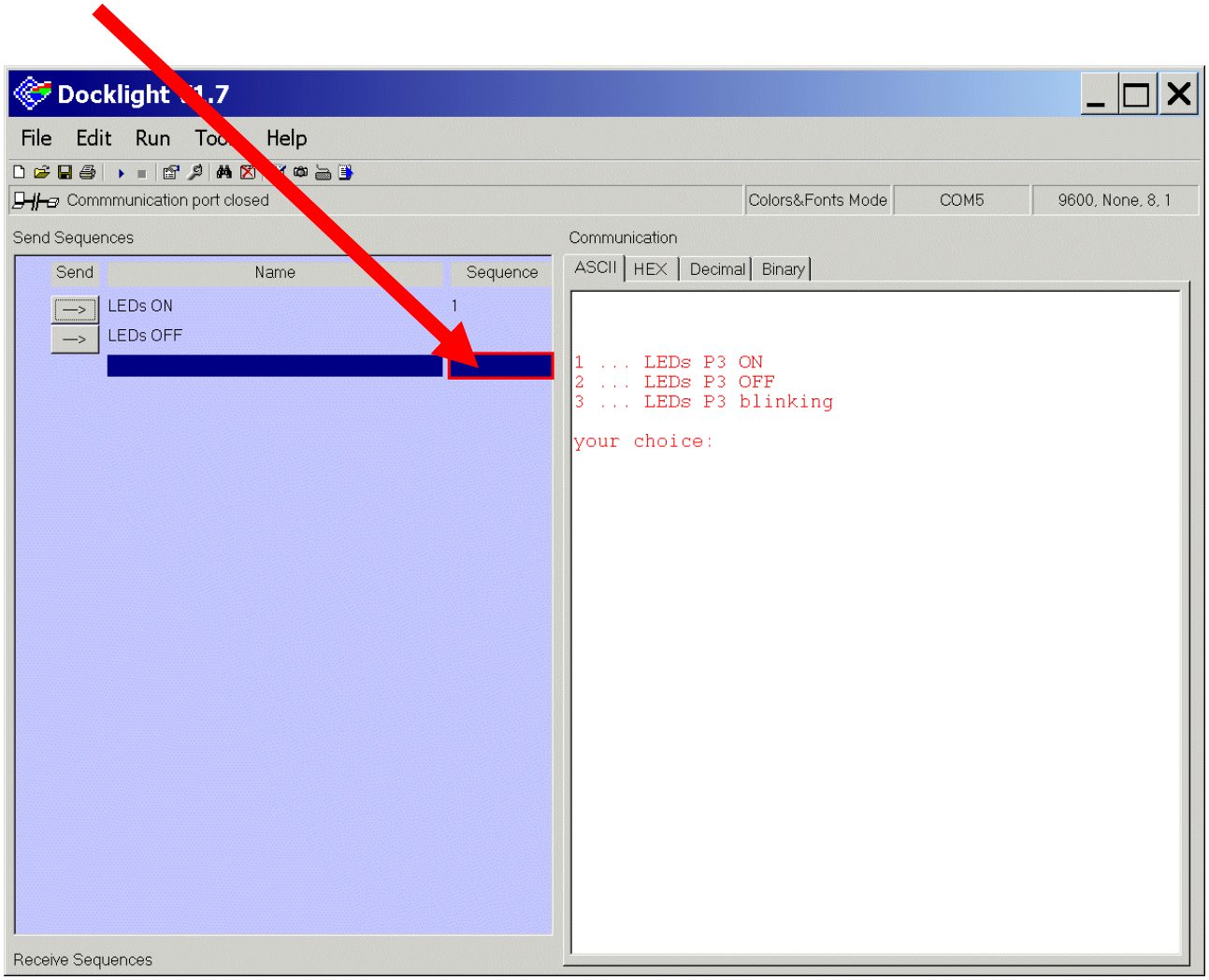


Edit Send Sequence: Sequence Definition: 1- Name: insert: LEDs OFF
Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 2

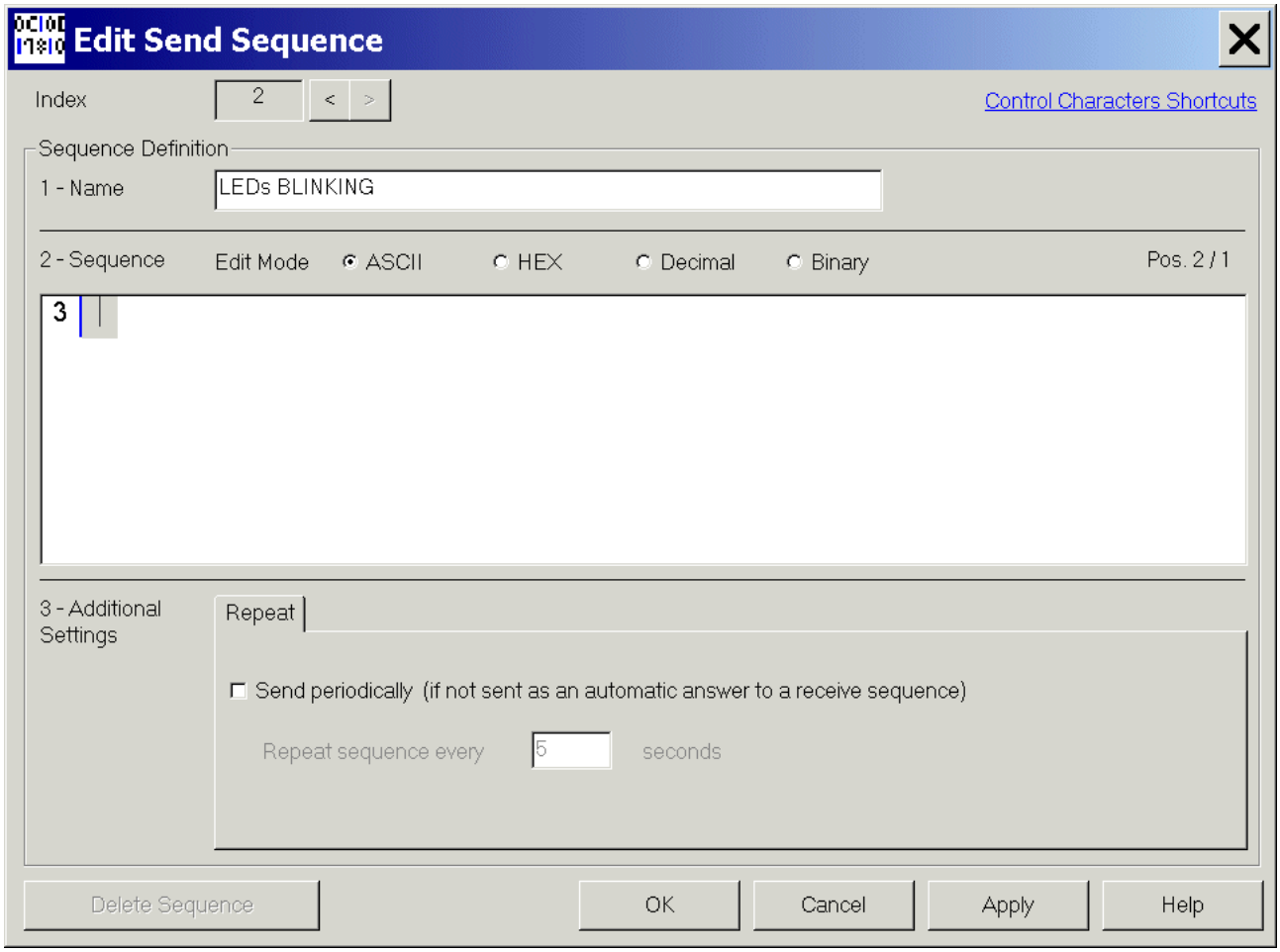


OK

Double click inside the red box:

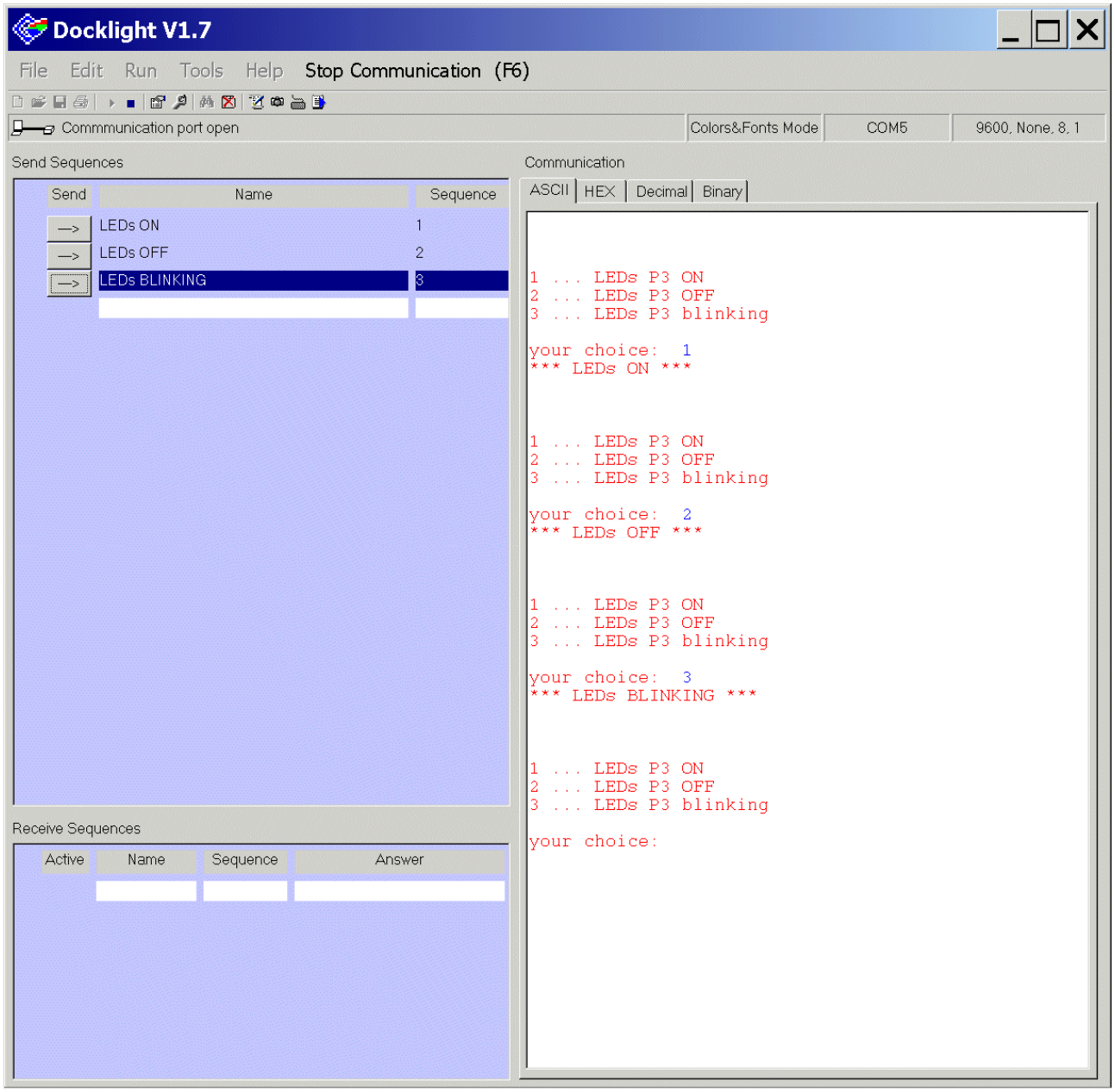


Edit Send Sequence: Sequence Definition: 1- Name: insert: LEDs BLINKING
Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 3



OK

Click —> LEDs ON or click —> LEDs OFF or click —> LEDs BLINKING and check the result on your Evaluation Board:



Conclusion:

In this step-by-step book you have learned how to use the XC888 board together with DAvE Bench.

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 Starter Kit Board.

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 XC888/XC886 microcontrollers!

Note:

There are step-by-step books for 8 bit microcontrollers (e.g. XC866 and XC88x), 16 bit microcontrollers (e.g. C16x, XC16x and XE16x) and 32 bit microcontrollers (e.g. TC1796 and TC1130).

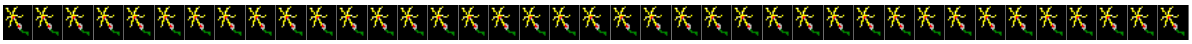
All these step-by-step books use the same microcontroller resources and the same example code.

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.

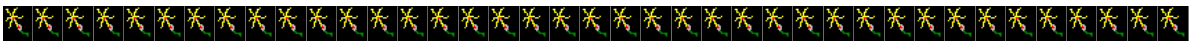
6.) Feedback (XC888, DAvE Bench): Your opinion, suggestions and/or criticisms



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