

# AP08067

## XC886/XC888

XC888 Starter Kit "Cookery-Book" for a "Hello world" application. You can do the "Hello world" example in this document with the evaluation version of the KEIL tool chain.

Microcontrollers



Never stop thinking

**Edition 2008-07-11**

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

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

#### **LEGAL DISCLAIMER**

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

#### **Information**

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

#### **Warnings**

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

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



**Note:** Table of Contents [see page 8](#).

### Introduction:

This “Appnote” is an Infineon Hands-On-Training / Cookery-Book / step-by-step-book. It will help inexperienced users to get an XC88x Evaluation Board / Starter Kit Board 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 XC88x microcontrollers!

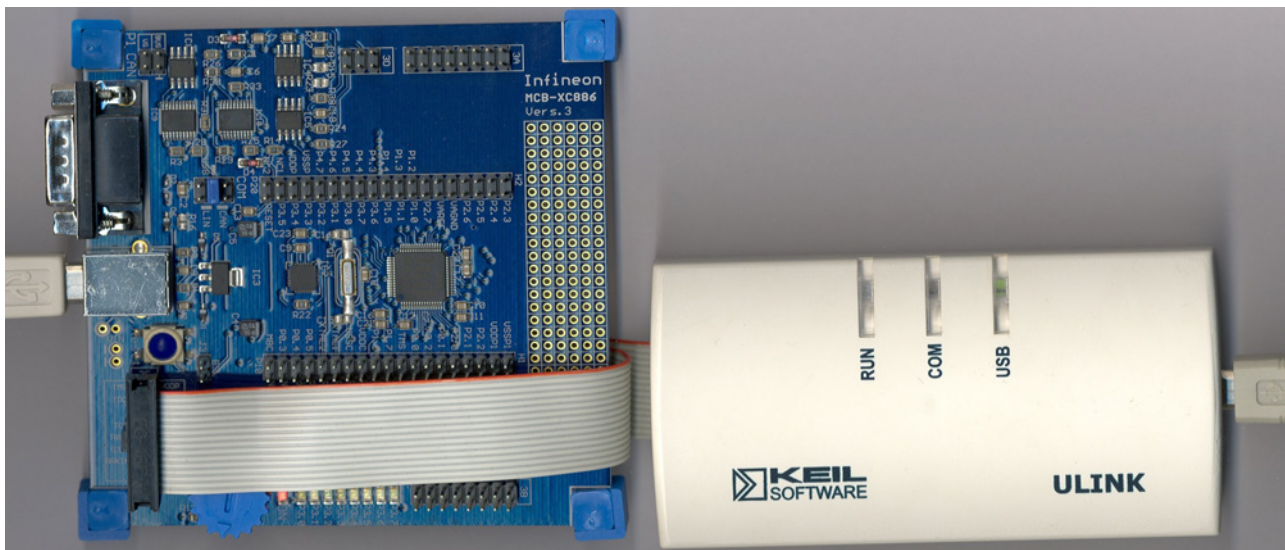


The New  
**XC800-Family**  
Designed to Make the Difference

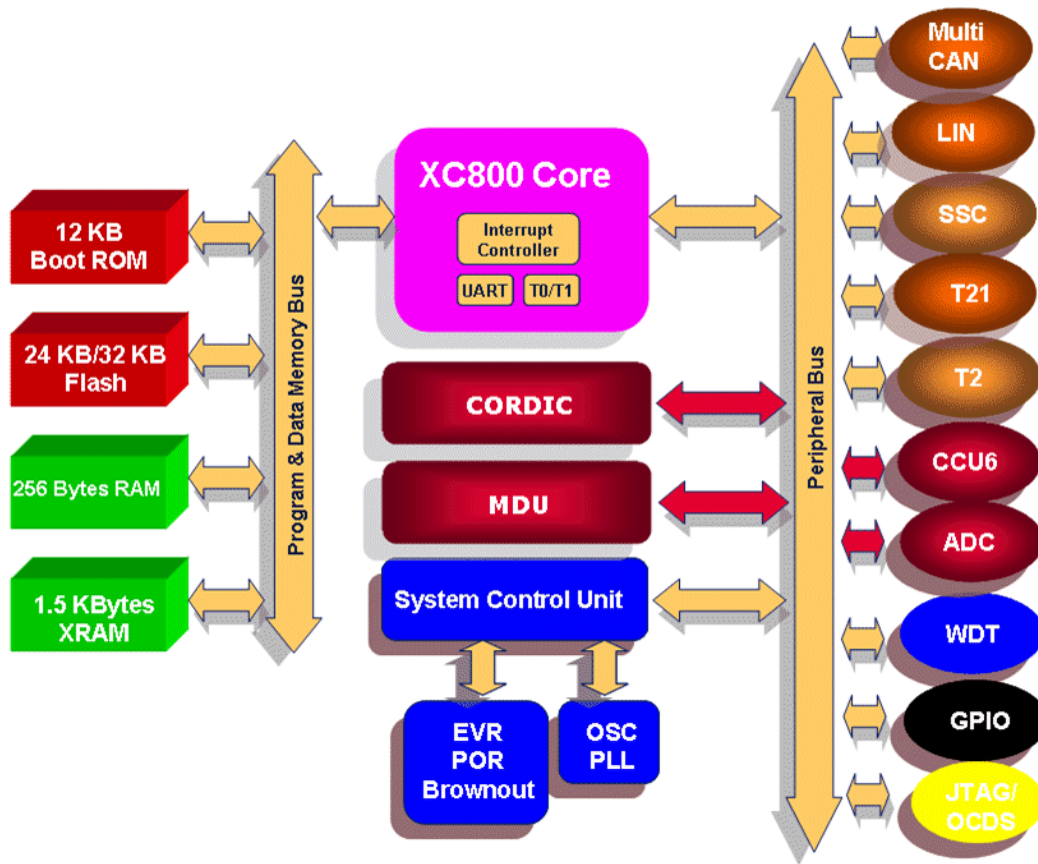
# Programming Example

## XC888CM-8FFA

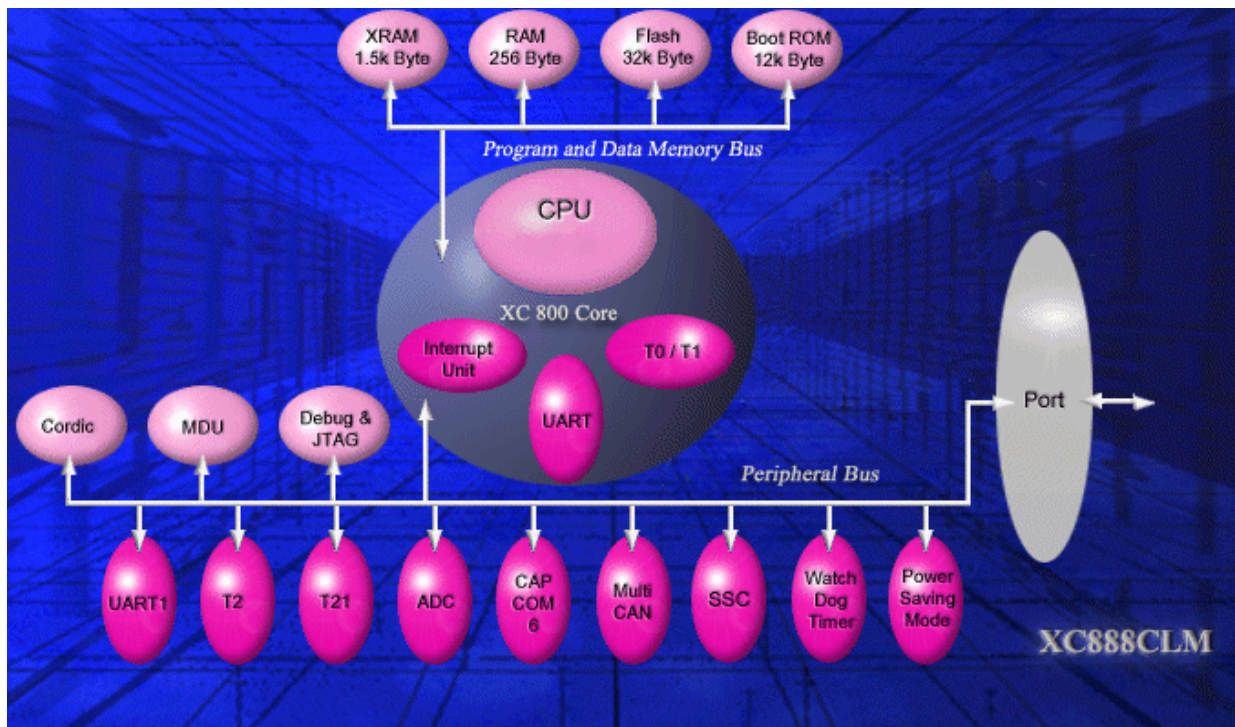
### Starter Kit



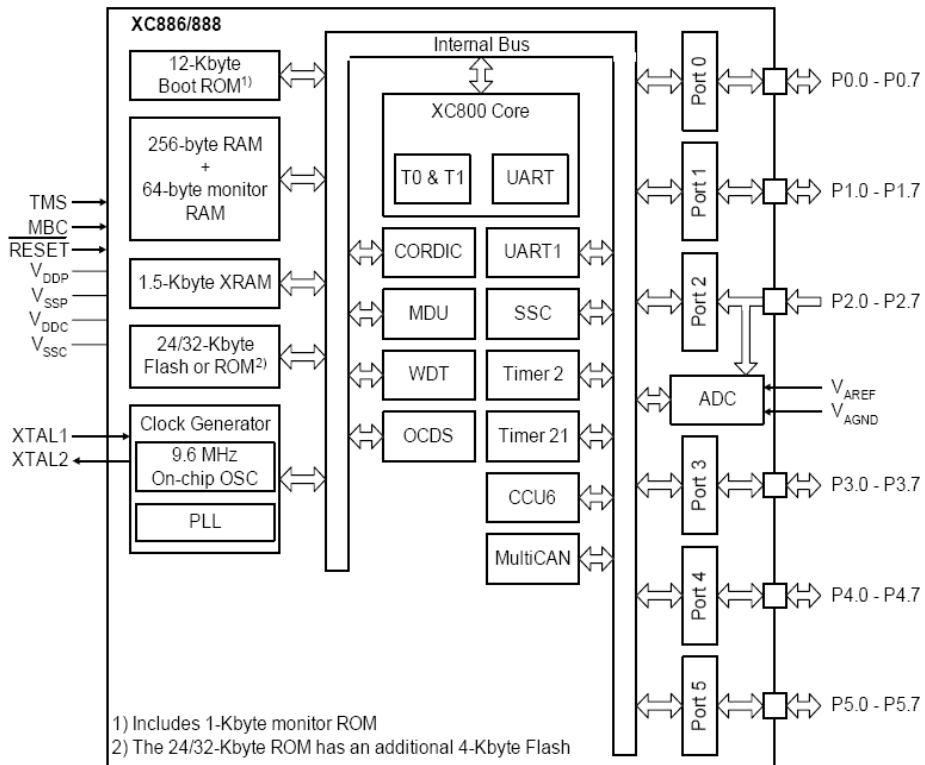
**XC888CLM-8FFA Block Diagram (Source: Product Marketing)**



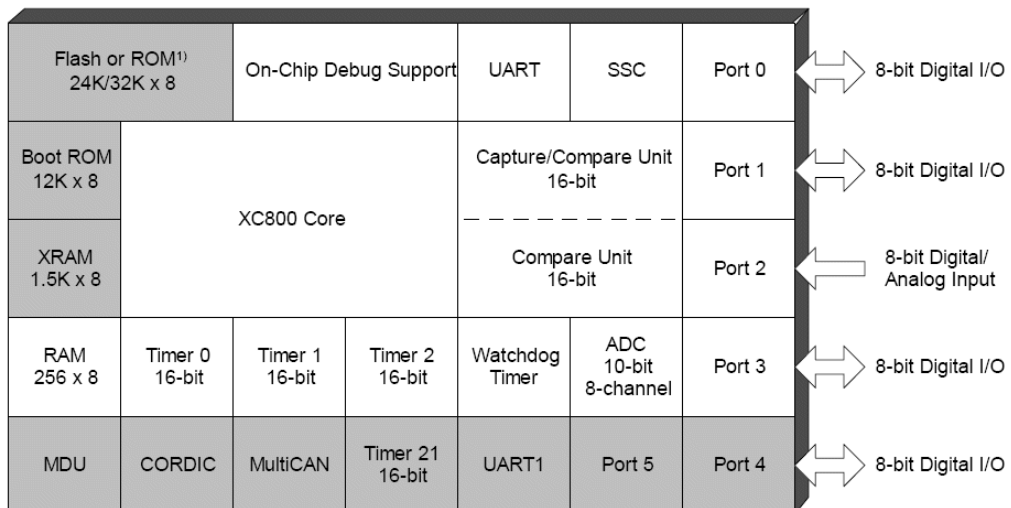
**XC888CLM-8FFA Block Diagram (Source: DAVe)**



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

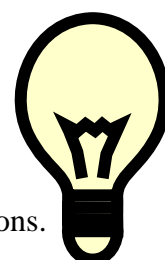
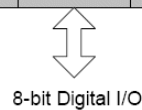


**XC888CLM-8FFA functional units (Source: User's Manual)**



Improved functionality in comparison to the XC866

1) All ROM devices come with an additional 4K x 8 Flash

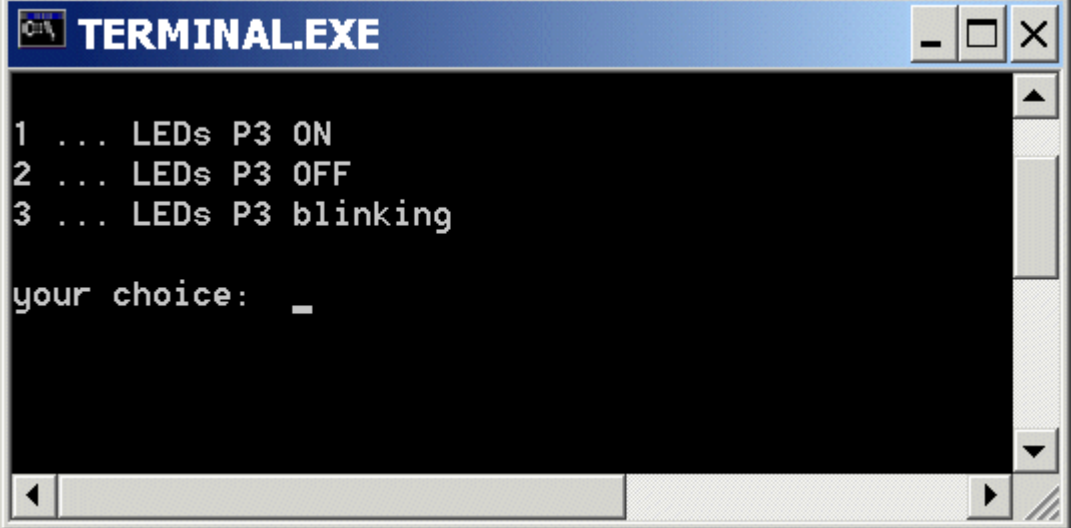


**Note:**

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.

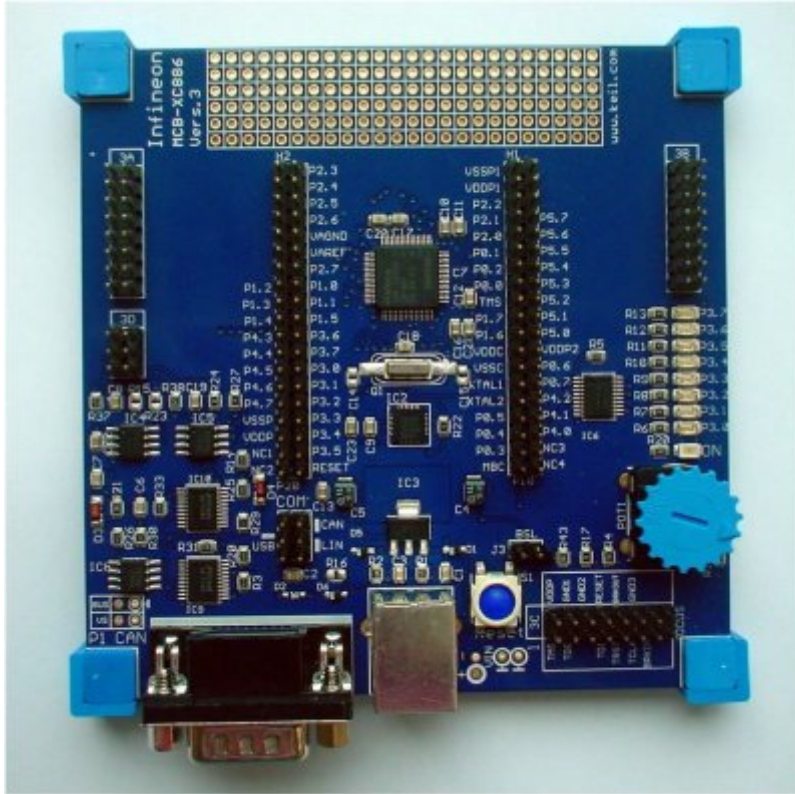
**“Cookery-book“**

For your first programming example for the XC888 Starter Kit Board:

<p>Your program:</p>	
<p>Chapter/ Step</p>	<p style="text-align: center;">*** Recipes ***</p>
<p>1.)</p>	<p><a href="#">XC888 Starter Kit Board</a> <a href="#">Power Supply (via USB), Jumper Setting, Serial Connection (via USB) to the notebook</a></p>
<p>2.)</p>	<p><a href="#">DAvE (program generator)</a> <a href="#">DAvE Installation (mothersystem) + DAvE Update Installation (XC888.DIP) for XC888</a></p>
<p>3.)</p>	<p><a href="#">Using DAvE</a> <a href="#">Microcontroller initialization for your programming example</a></p>
<p>4.)</p>	<p><a href="#">Using the KEIL Development Tools (C-Compiler)</a> <a href="#">Programming of your application (XC888) with KEIL tool chain (µVision3) Compiler V8.09a + first steps with the Simulator</a></p>
<p>5.)</p>	<p><a href="#">Using the simulator</a></p>
<p>6.)</p>	<p><a href="#">Using real hardware (+ OnChipFlash-Programming)</a></p>
<p><b>Feedback</b></p>	<p>7.) <a href="#">Feedback</a></p>



1.) XC888 Starter Kit Board:



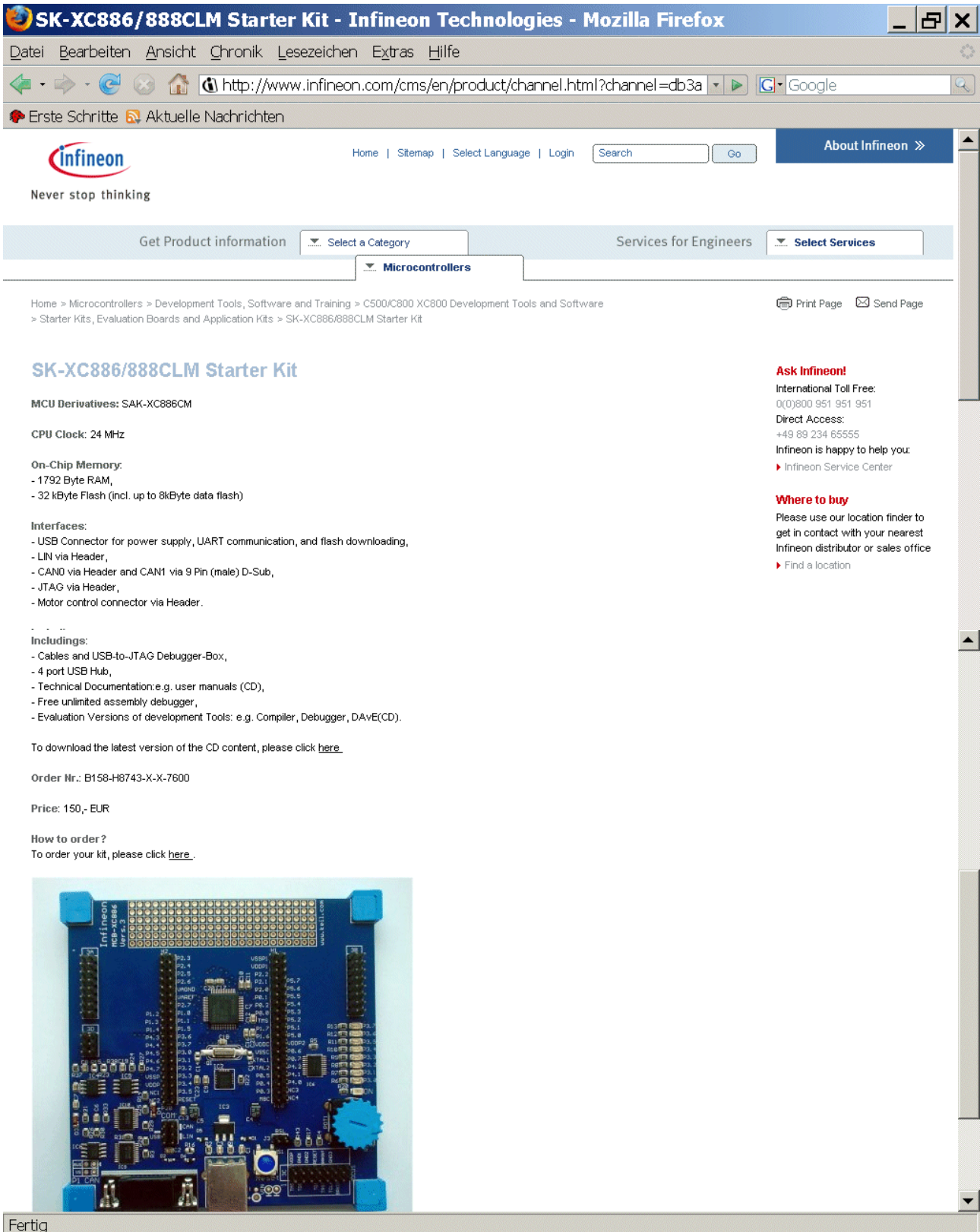
Ordering information:

Starter Kits – Type	$\mu$ C	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:



The screenshot shows a Mozilla Firefox browser window displaying the Infineon website for the SK-XC886/888CLM Starter Kit. The browser's address bar shows the URL: <http://www.infineon.com/cms/en/product/channel.html?channel=db3a>. The page features the Infineon logo and the slogan "Never stop thinking". Navigation links include Home, Sitemap, Select Language, and Login. A search bar and a "Go" button are present. A blue button labeled "About Infineon" is also visible.

The main content area includes a "Get Product information" section with a "Select a Category" dropdown menu set to "Microcontrollers". A "Services for Engineers" section has a "Select Services" dropdown menu. The breadcrumb trail reads: 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. There are links for "Print Page" and "Send Page".

The product title is "SK-XC886/888CLM Starter Kit". Key specifications listed are:

- 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).

Additional information includes: "To download the latest version of the CD content, please click [here](#)." "Order Nr.: B158-H8743-X-X-7600" and "Price: 150,- EUR". A "How to order?" section states: "To order your kit, please click [here](#)." At the bottom, there is a photograph of the blue PCB starter kit with various components and connectors. The word "Fertig" (Ready) is written below the image.

On the right side of the page, there are sections for "Ask Infineon!" with contact information: "International Toll Free: 0(0)800 951 951 951", "Direct Access: +49 89 234 65555", and "Infineon is happy to help you." with a link to "Infineon Service Center". Below this is a "Where to buy" section with the text: "Please use our location finder to get in contact with your nearest Infineon distributor or sales office" and a link "Find a location".

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!**

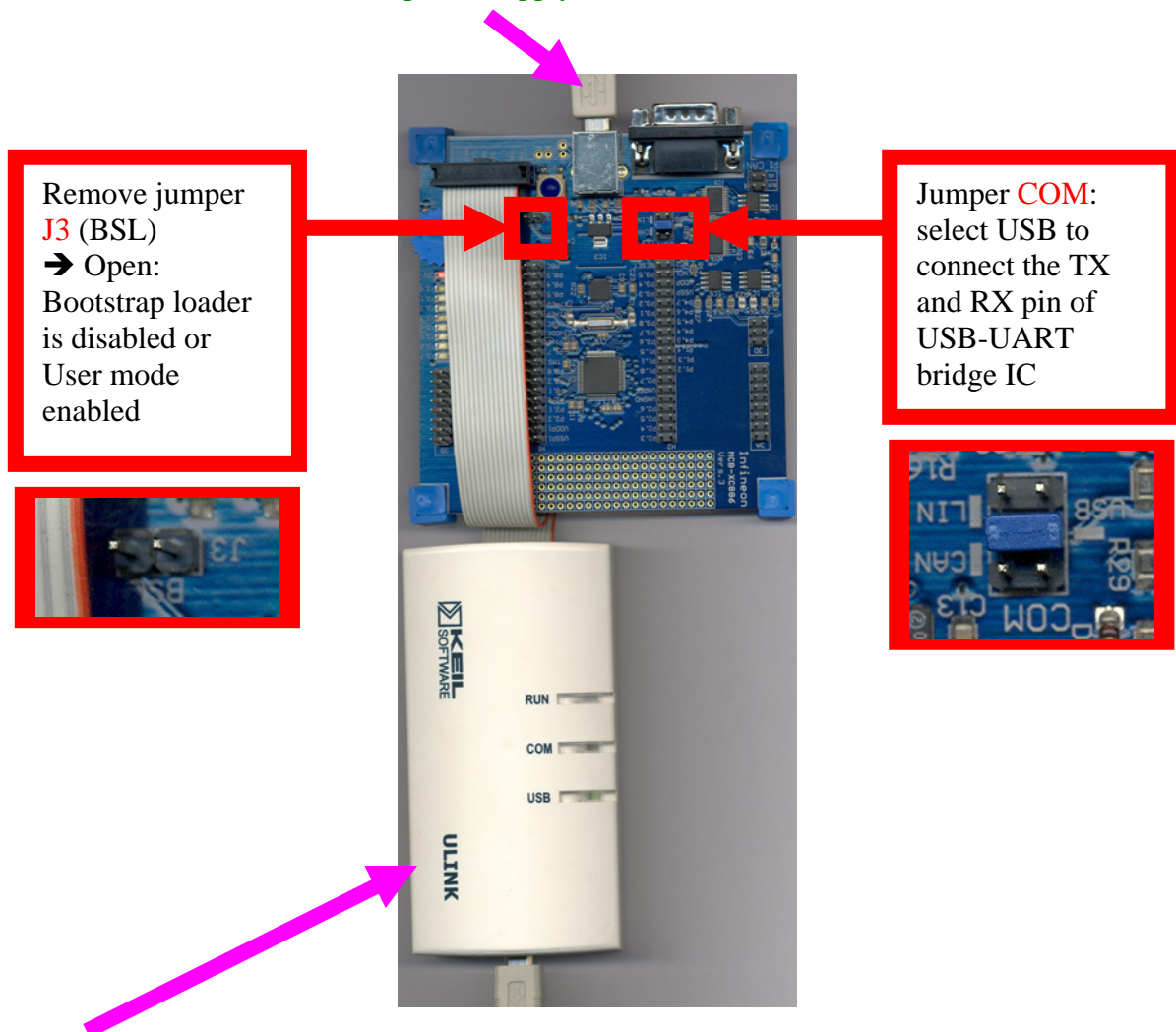


**Reason:**

When the ULINK is already connected to the Starter Kit Board, the Starter Kit Board must be supplied with power for the ULINK 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.

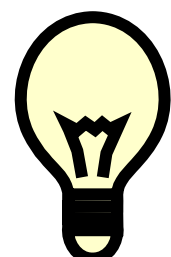
**USB Cable** [used for: UART communication (the RS232 serial interface is available via USB); the USB connection works also as the power supply].



**KEIL-ULINK** (used for: On-Chip-Flash-Programming and Debugging).

**Note:**

For further information, please refer to the [XC888 Board Manual, V2.1, Sept 2006](#) .



## 2.) DAvE – Installation for XC888 microcontrollers:



### Install DAvE (mothersystem):

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

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

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

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

1.)

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


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

Title	Date	Version	Size
<b>Development Tools</b>			
 XC888CLM DIP file for DAVe (Microcontroller Configuration Tool), V1.2	25 Jun 2007	V1.2	7.6 MB

**Unzip** the zip-file “XC888CLM\_v1.2.zip” and save “XC888CLM\_v12.DIP “

@ e.g. D:\DAvE\XC888-2007-08-14\XC888CLM\_v12 .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\XC888-2007-08-14](#)

Forward>

Select: Available Products

click ✓ [XC888CLM](#)

Forward>

Install

End

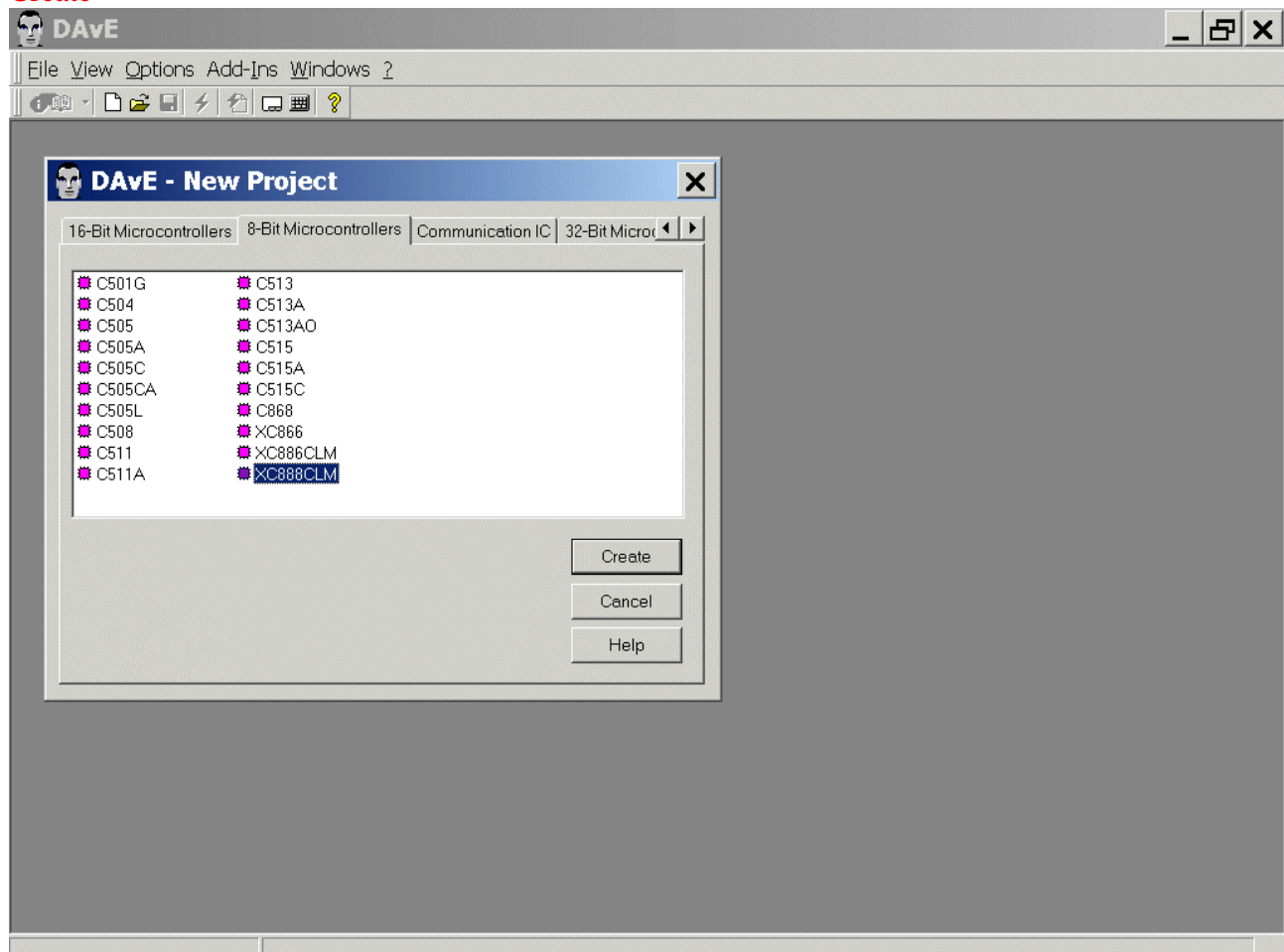
4.) DAVe is now ready to generate code for the XC888CLM microcontroller.

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



Start the program generator DAVe and select the XC888CLM microcontroller:

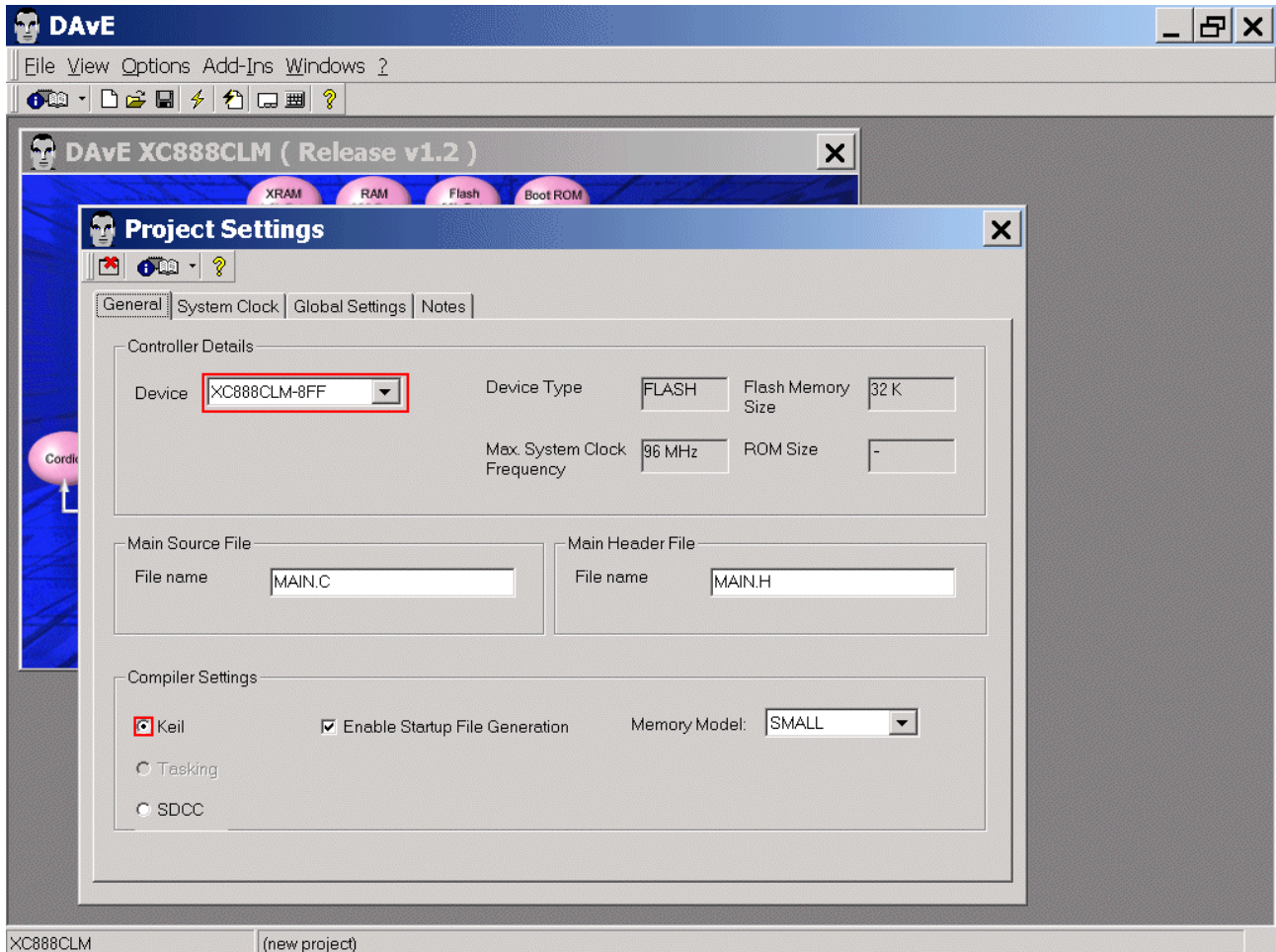
File  
New  
8-Bit Microcontrollers  
select XC888CLM  
Create



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

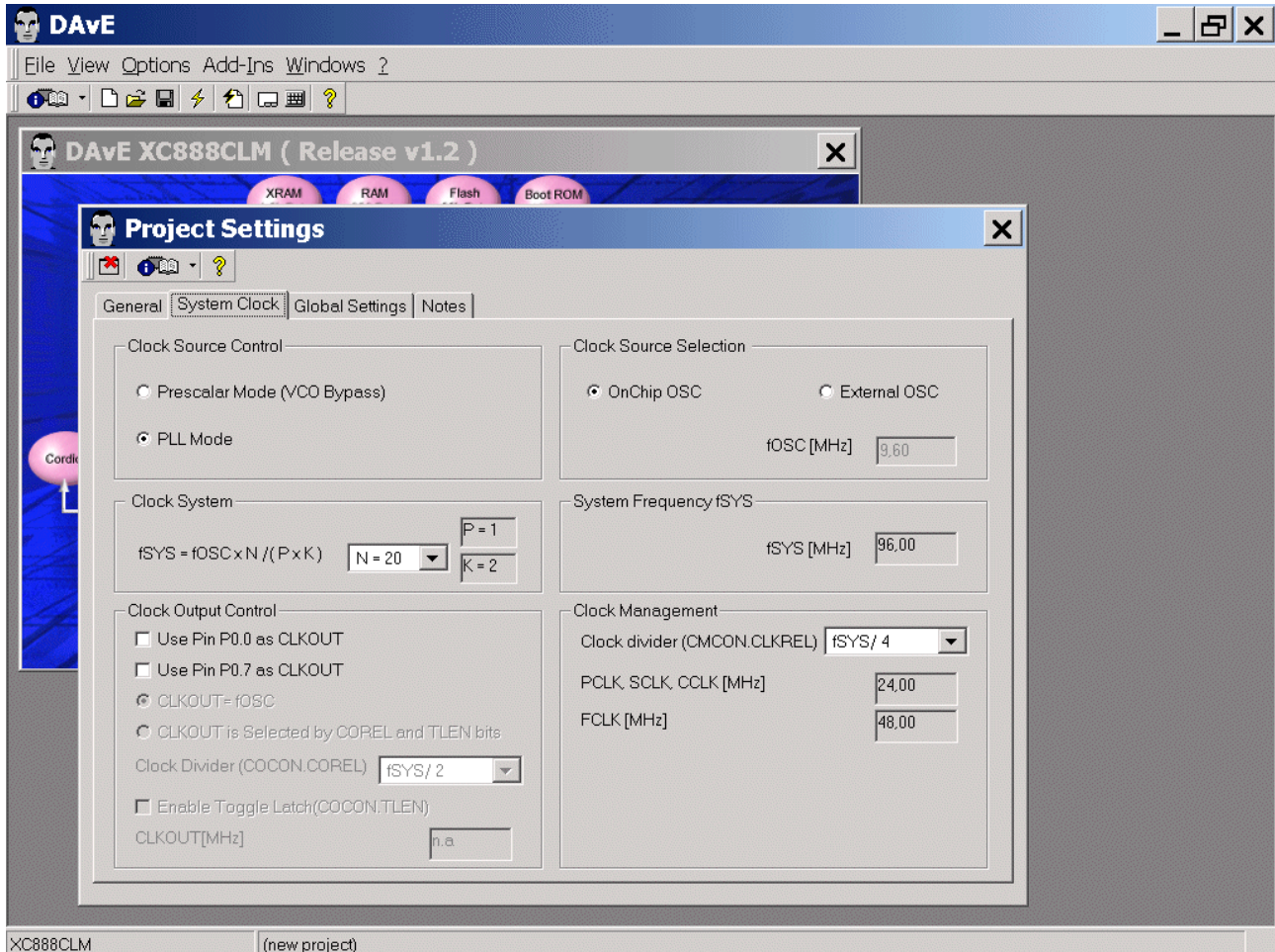
General: Controller Details: Device: **check/select** XC888CLM-8FF

General: For the **KEIL** Compiler **check/choose**  Keil in the Compiler Settings:



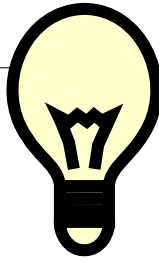


System Clock: (do nothing)



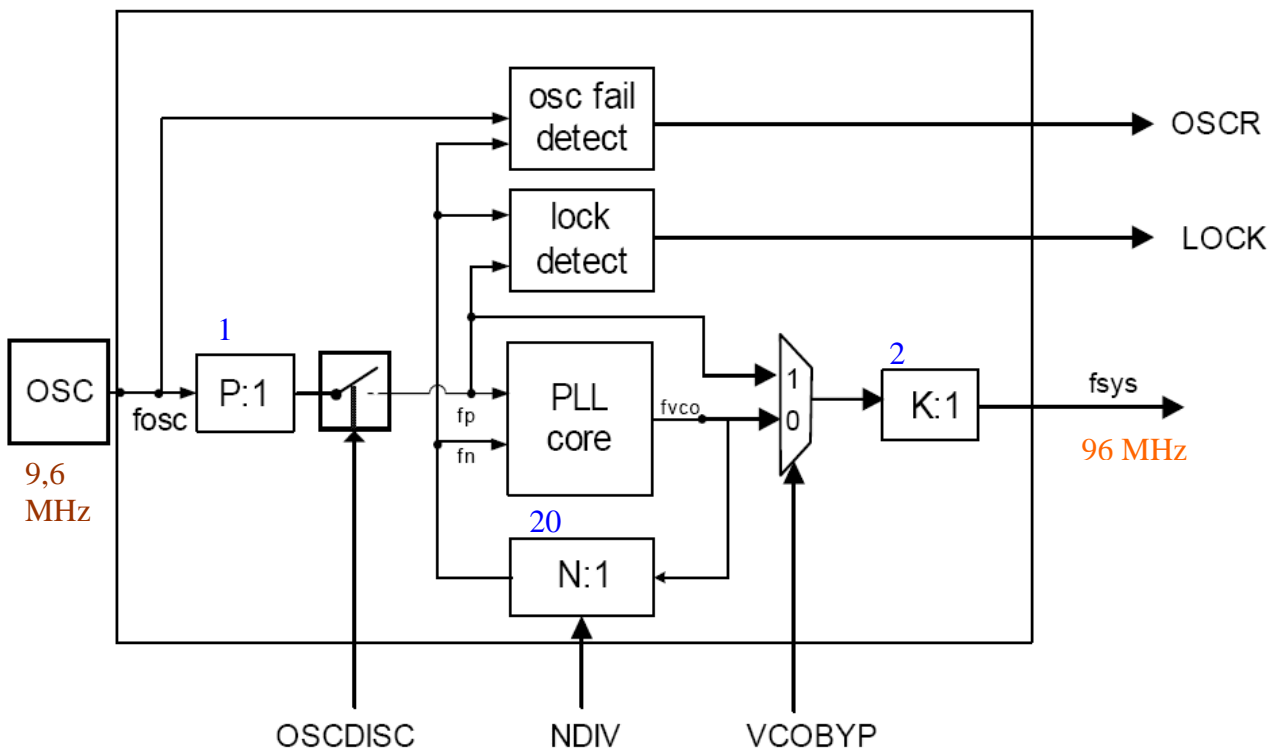
**Note:**  
CPU clock is 24 MHz.





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

Clock Generation Unit (CGU) Block Diagram



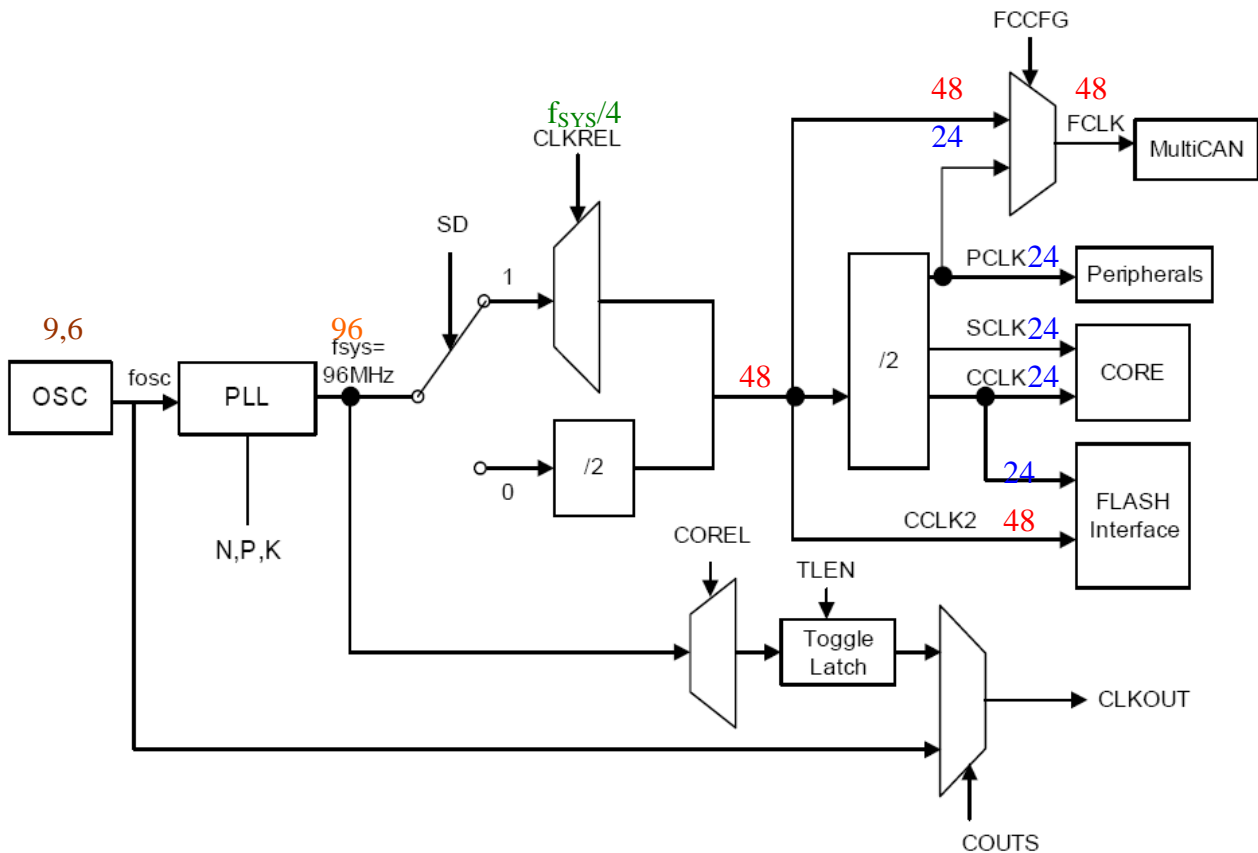
Note:

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



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

Clock Generation from  $f_{sys}$ :



Note:

$f_{sys} = 96$  MHz

CPU clock: CCLK, SCLK = 24 MHz

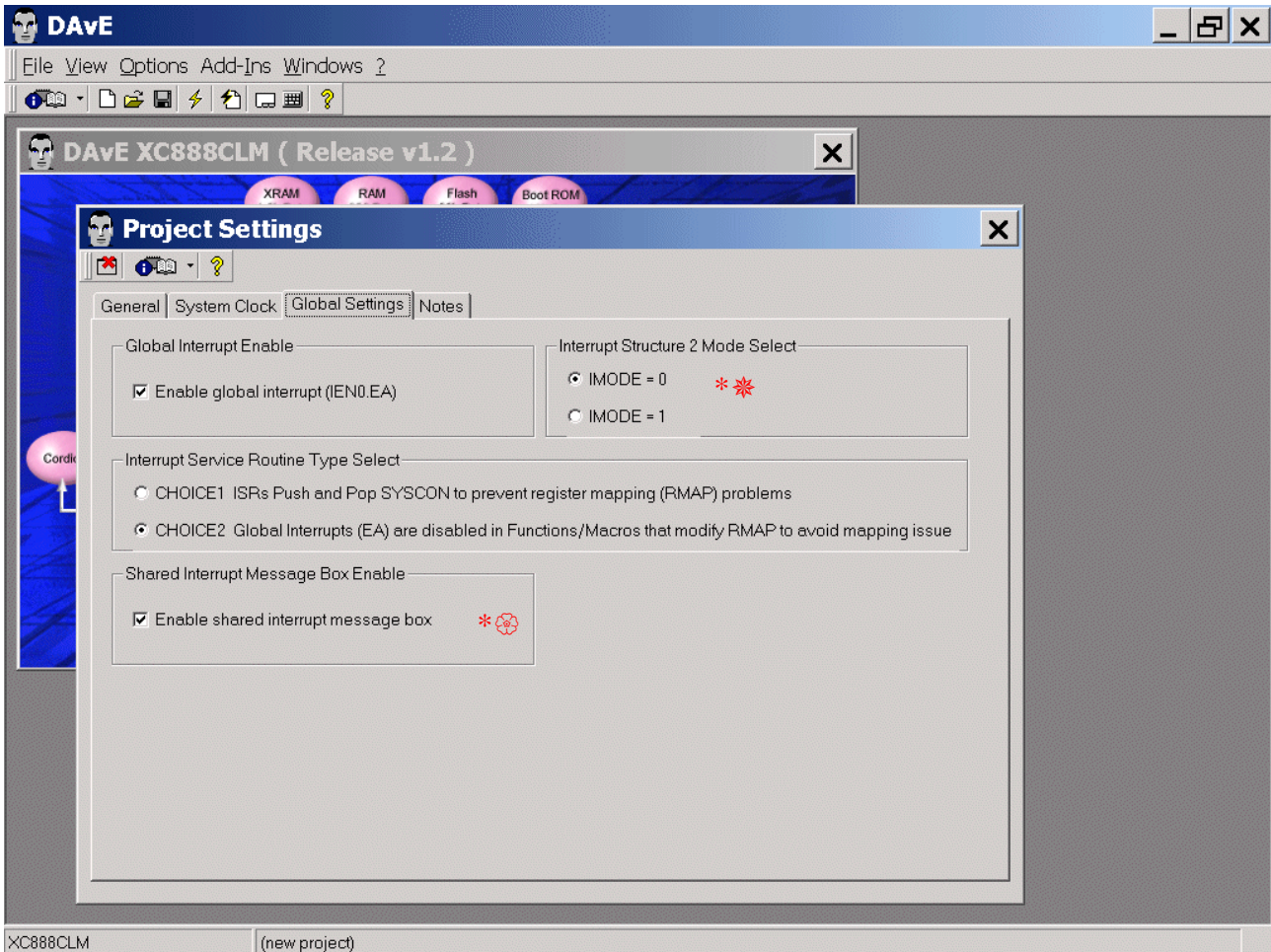
Fast clock: FCLK = 24 or 48 MHz

Peripheral clock: PCLK = 24 MHz

Flash Interface clock: CCLK2 = 48 MHz and CCLK = 24 MHz

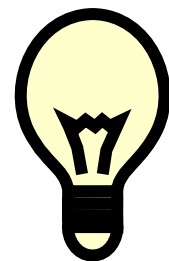
CLKREL: The clock division factor  $f_{sys}/4$  (see DAVe screenshot page 17) is inclusive the fixed divider factor of 2.

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.

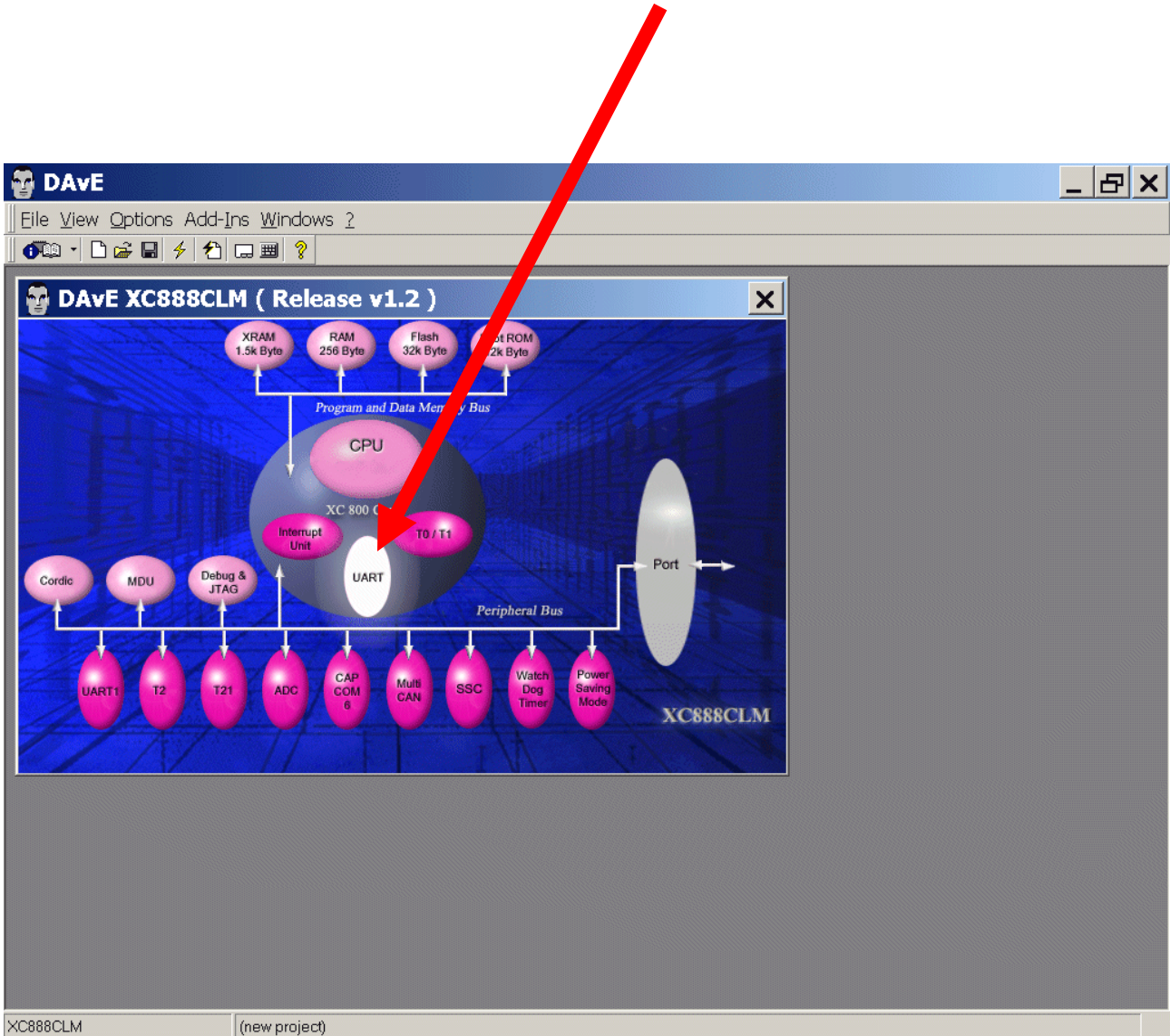
**Notes:** If you wish, you can insert your comments here.

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



Configuration of the ASC0:

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

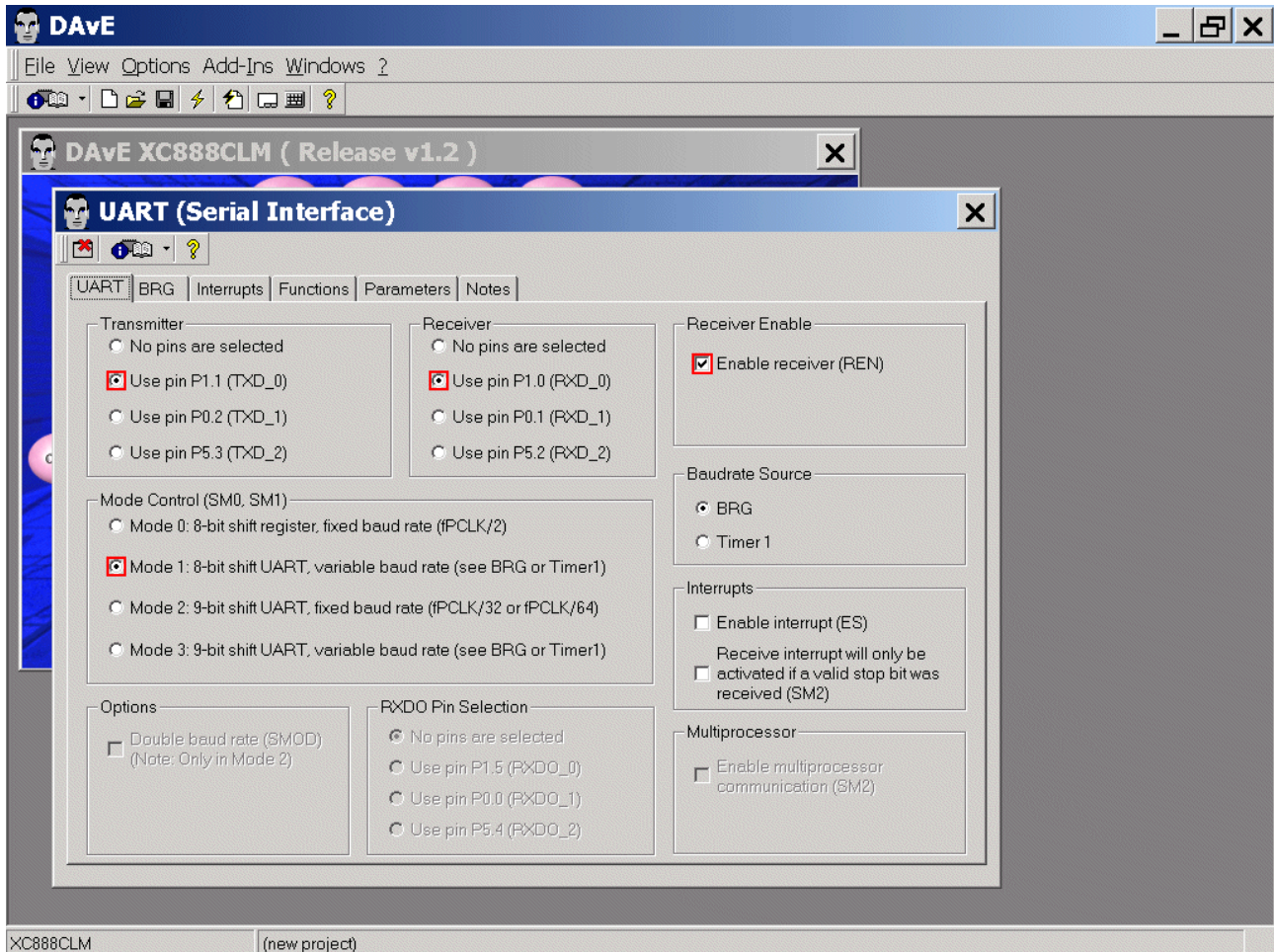


UART: Transmitter: **click**  Use pin P1.1 (TXD\_0)

UART: Receiver: **click**  Use pin P1.0 (RXD\_0)

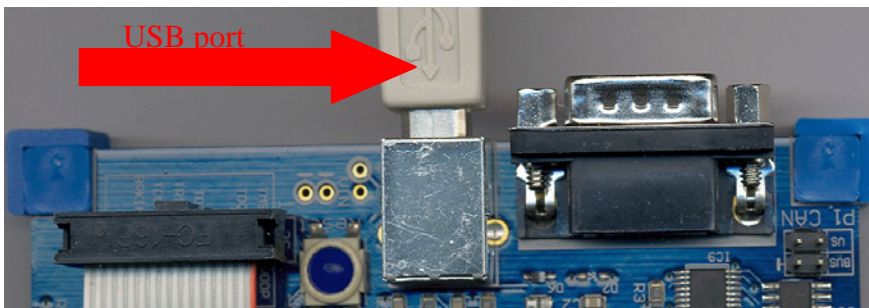
UART: Receiver Enable: **click**  Enable receiver (REN)

UART: Mode Control: **click**  Mode 1: 8-bit shift UART, variable baud rate (see BRG or Timer1)



**Note:**

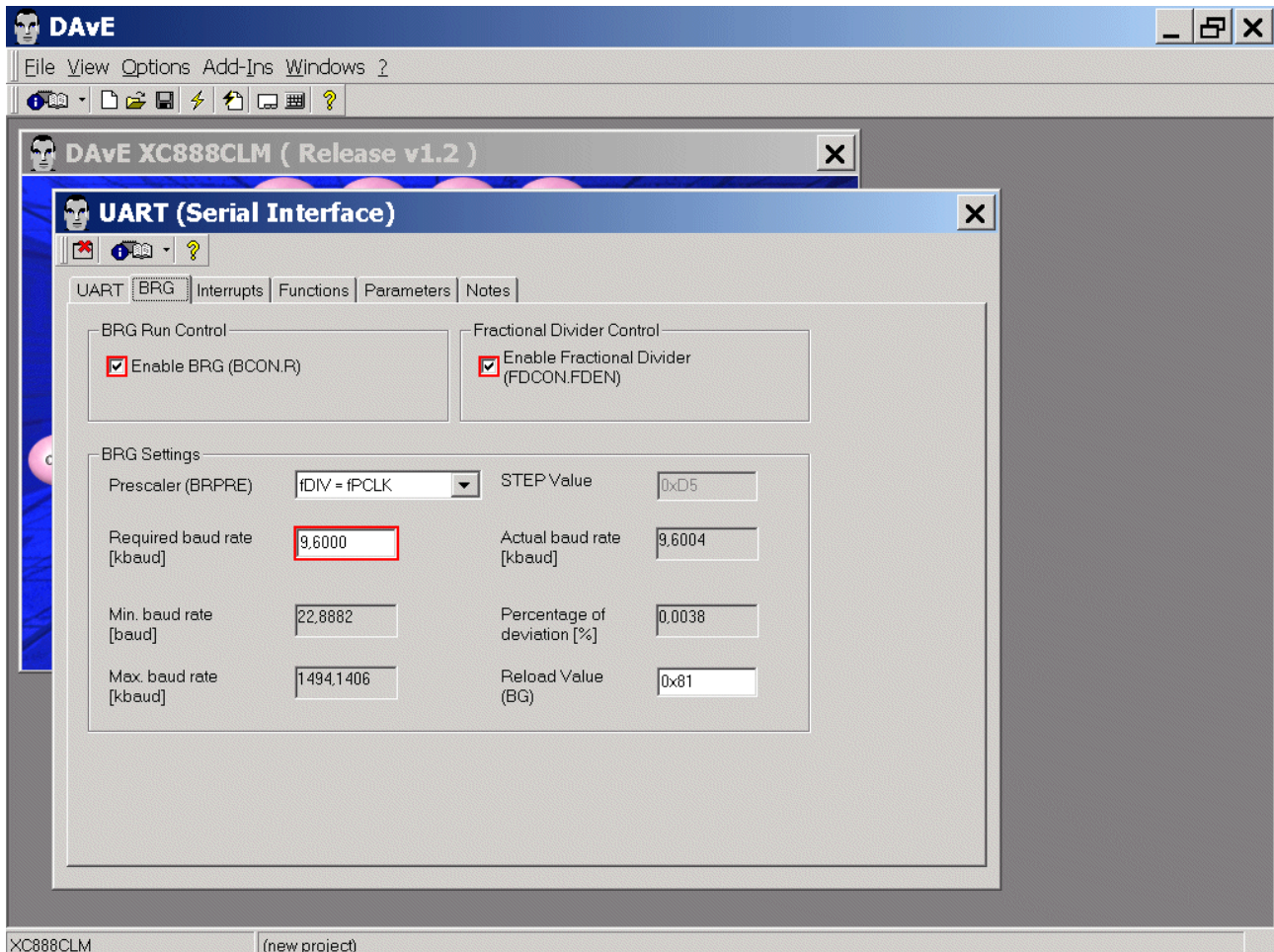
The RS232 serial interface (UART pins P1.0 and P1.1) is available via the [USB port](#) which converts the TTL-UART-signals to USB-signals (using a SILICON LABS [CP2102](#) "Single-Chip USB To UART Bridge").



BRG: BRG Run Control: **click/check** ✓ Enable BRG

BRG: Fractional Divider Control: **click/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)

Level	Interrupt Source	Priority 0	Priority 1	Priority 2	Priority 3
Level 0	Non Maskable Interrupt (NMI)	Highest Priority (can't be changed)			
Level 1	External Interrupt 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 2	Timer 0 Interrupt	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 3	External Interrupt 1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 4	Timer 1 Interrupt	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 5	UART Interrupt	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 6	Timer 2 / BRG / MultiCAN Node 0 Interpts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 7	ADC / MultiCAN Node 1 and 2 Interpts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 8	SSC Interrupt	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 9	External 2 / T21 / UART1 / BRG1 Intrpts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 10	External [6:3] / MultiCAN Node 3 Interpts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 11	CCU6 Node 0 / MultiCAN Node 4 Interpts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 12	CCU6 Node 1 / MultiCAN Node 5 Interpts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 13	CCU6 Node 2 / MultiCAN Node 6 Interpts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 14	CCU6 Node 3 / MultiCAN Node 7 Interpts	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



**Note:**

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



**Interrupt Priorities:**

**Note (Source: Application Note AP08053):**

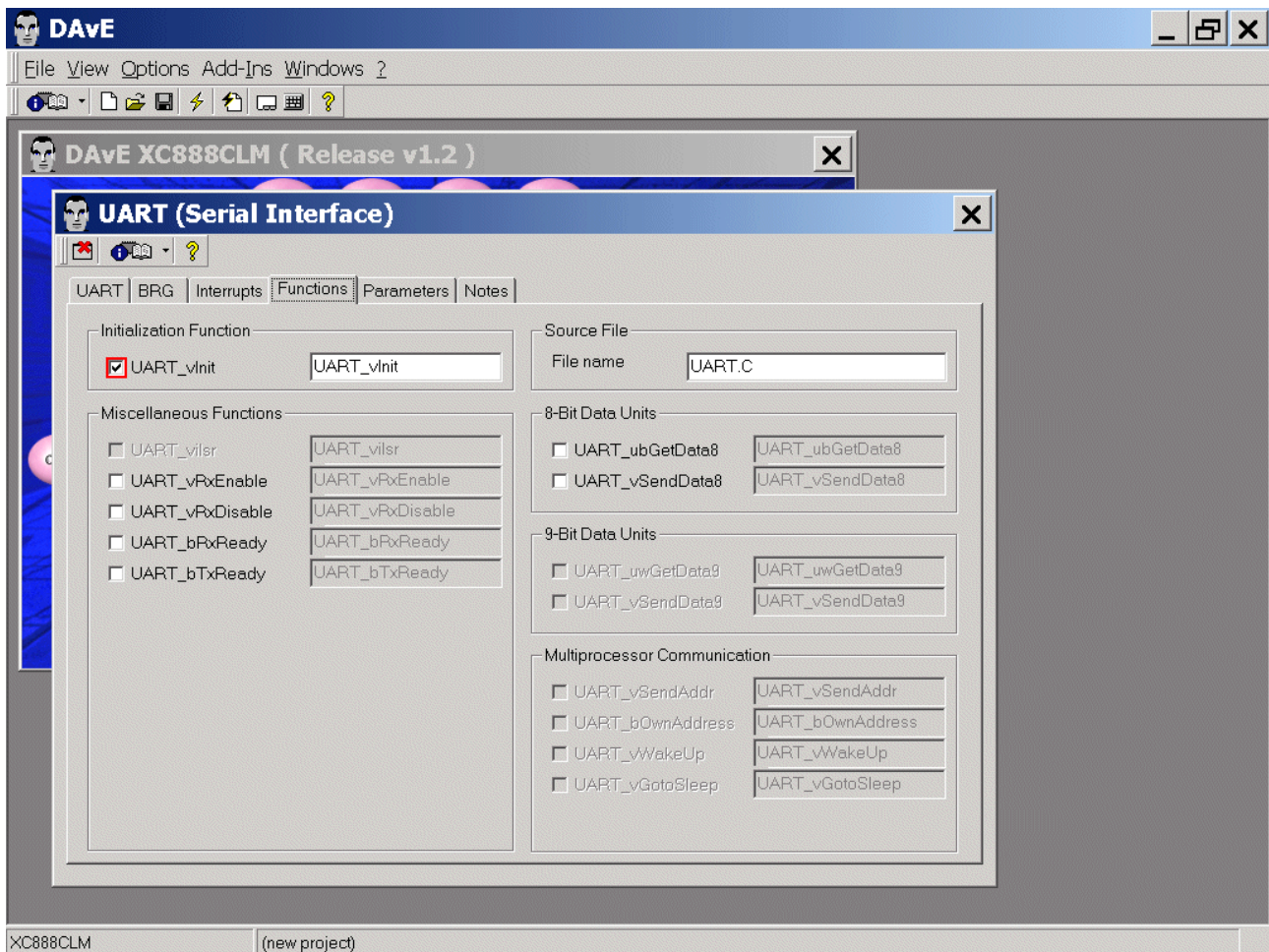
There could be **six** interrupt priorities.

These priorities, with **6** being 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: **click** ✓ UART\_vInit

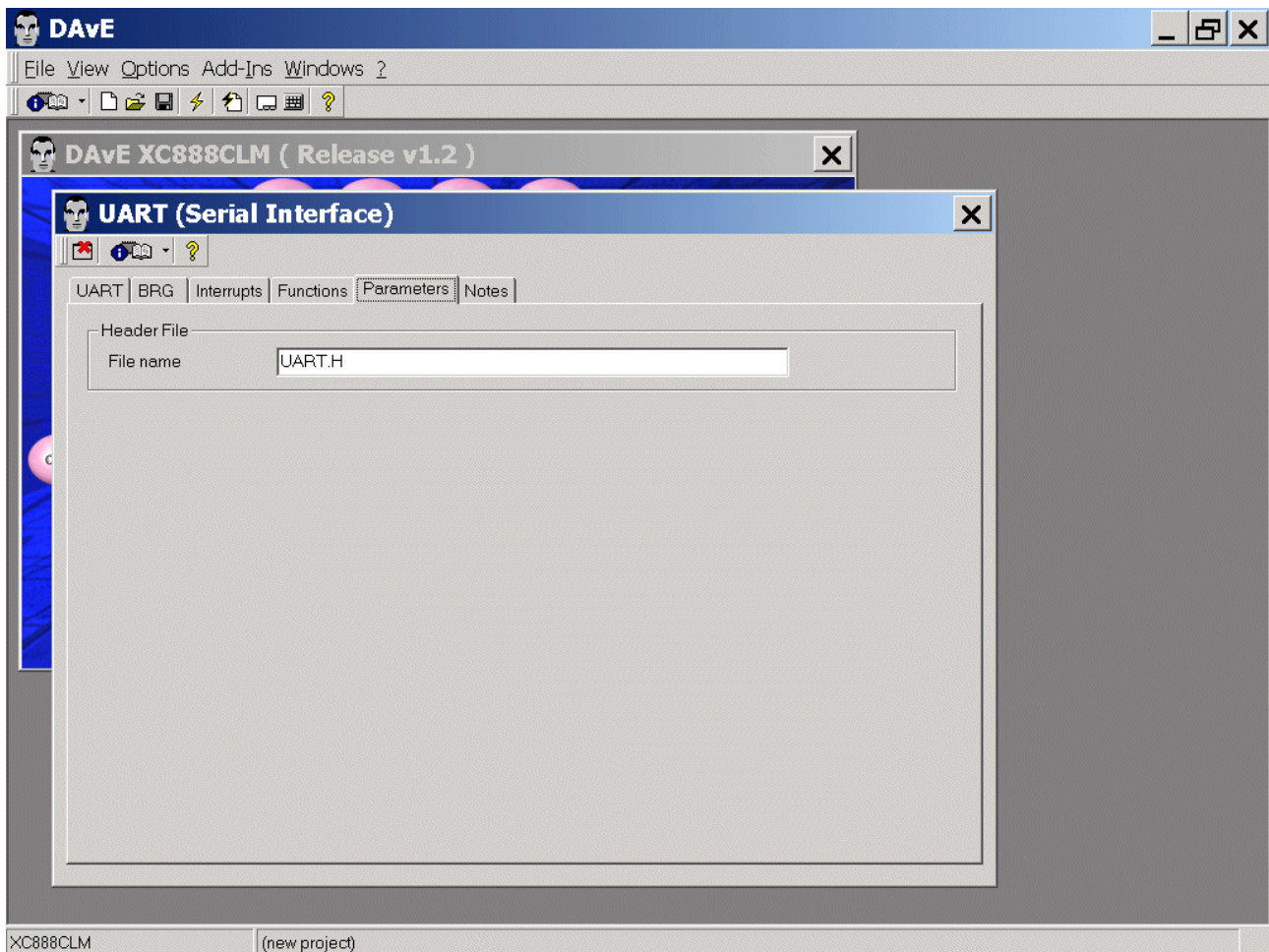


**Note:**


You can change function names (e.g. UART\_vInit) and file names (e.g. UART.C) anytime.



Parameters: (do nothing)

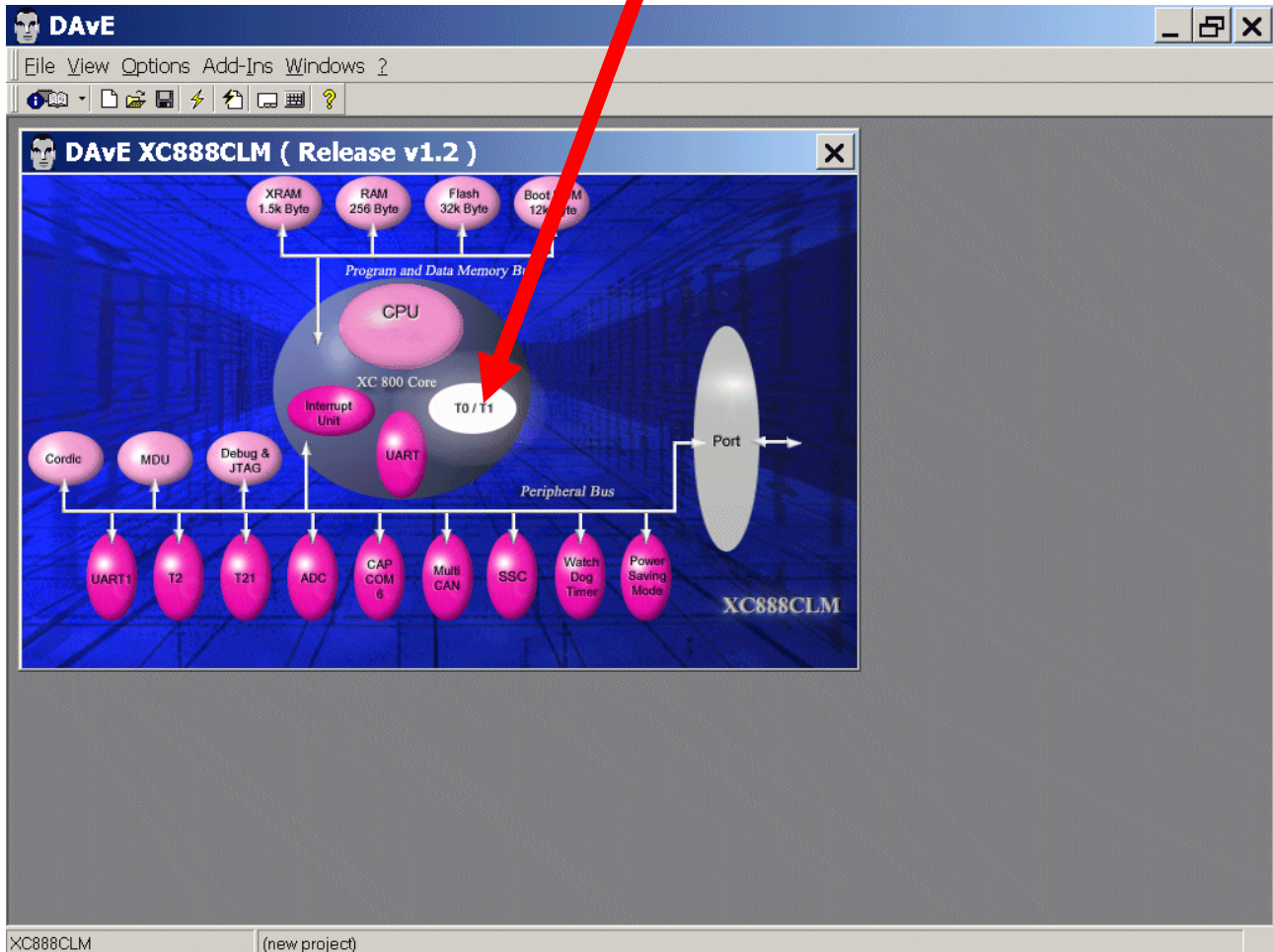


Notes: If you wish, you can insert your comments here.

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.



**Note:**

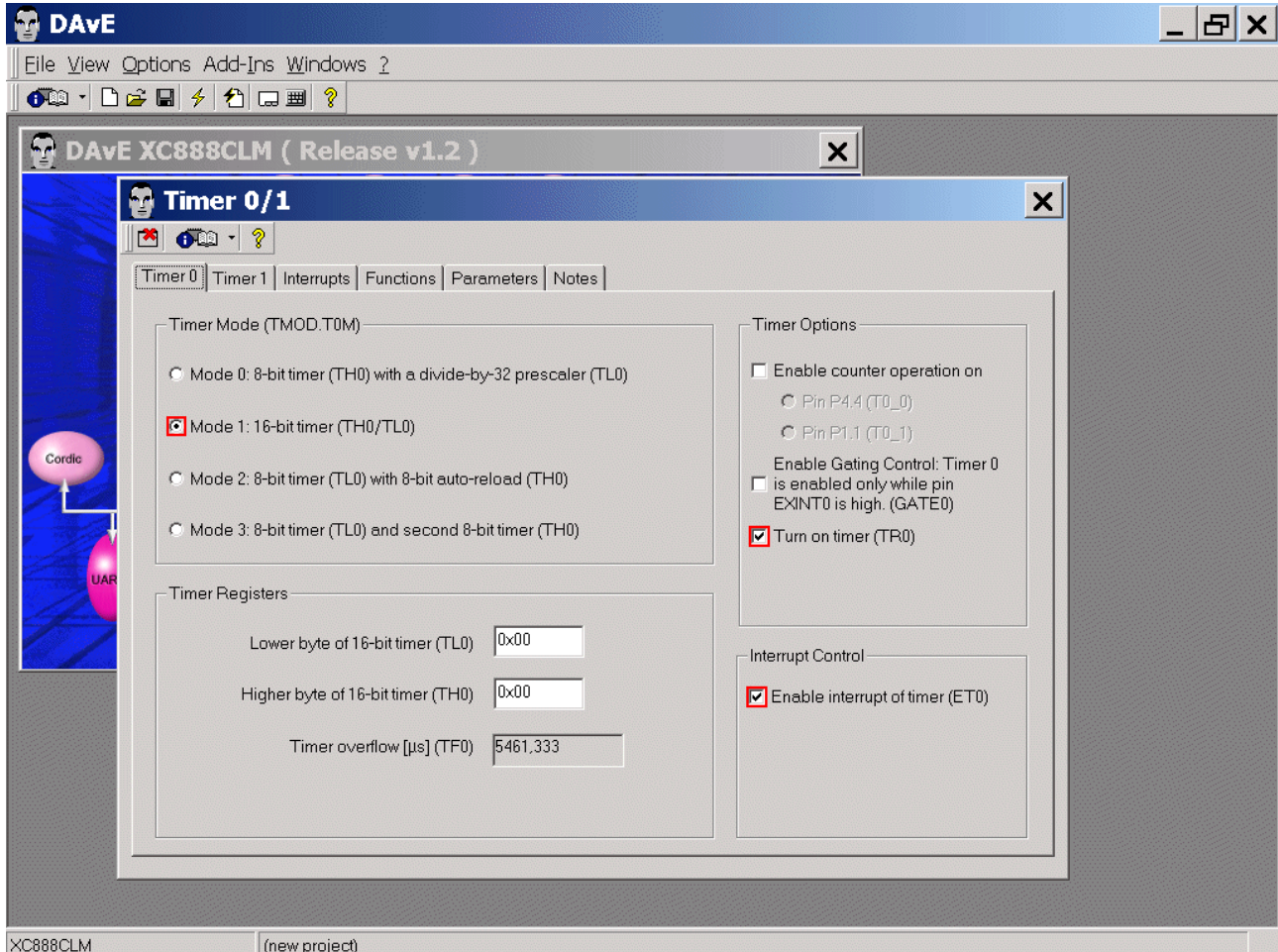
The LEDs on IO\_Port\_3 will be blinking (if selected in the main menu) with a frequency of about 1 second (done in the Timer\_0-Interrupt-Service-Routine). Therefore we have to configure Timer\_0.



Timer0: Timer Mode: **click**  Mode 1: 16-bit timer

Timer0: Timer Options: **click**  Turn on timer (TR0)

Timer0: Interrupt Control: **click**  Enable interrupt of timer (ET0)



**Note:**

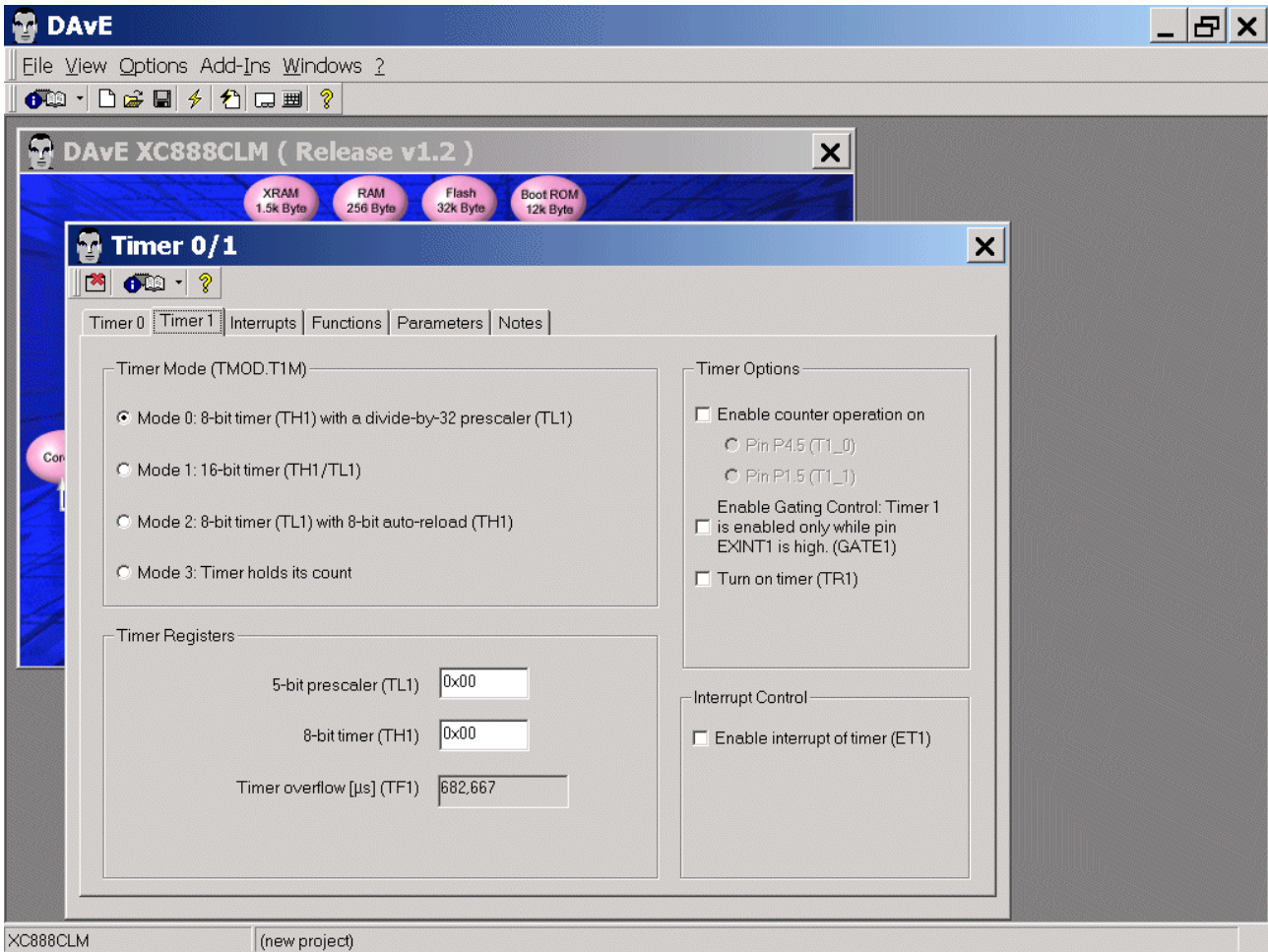
We need 183 Timer\_0 overflows to achieve an approximate 1 second delay.

This will be handled in the Timer\_0 interrupt function.

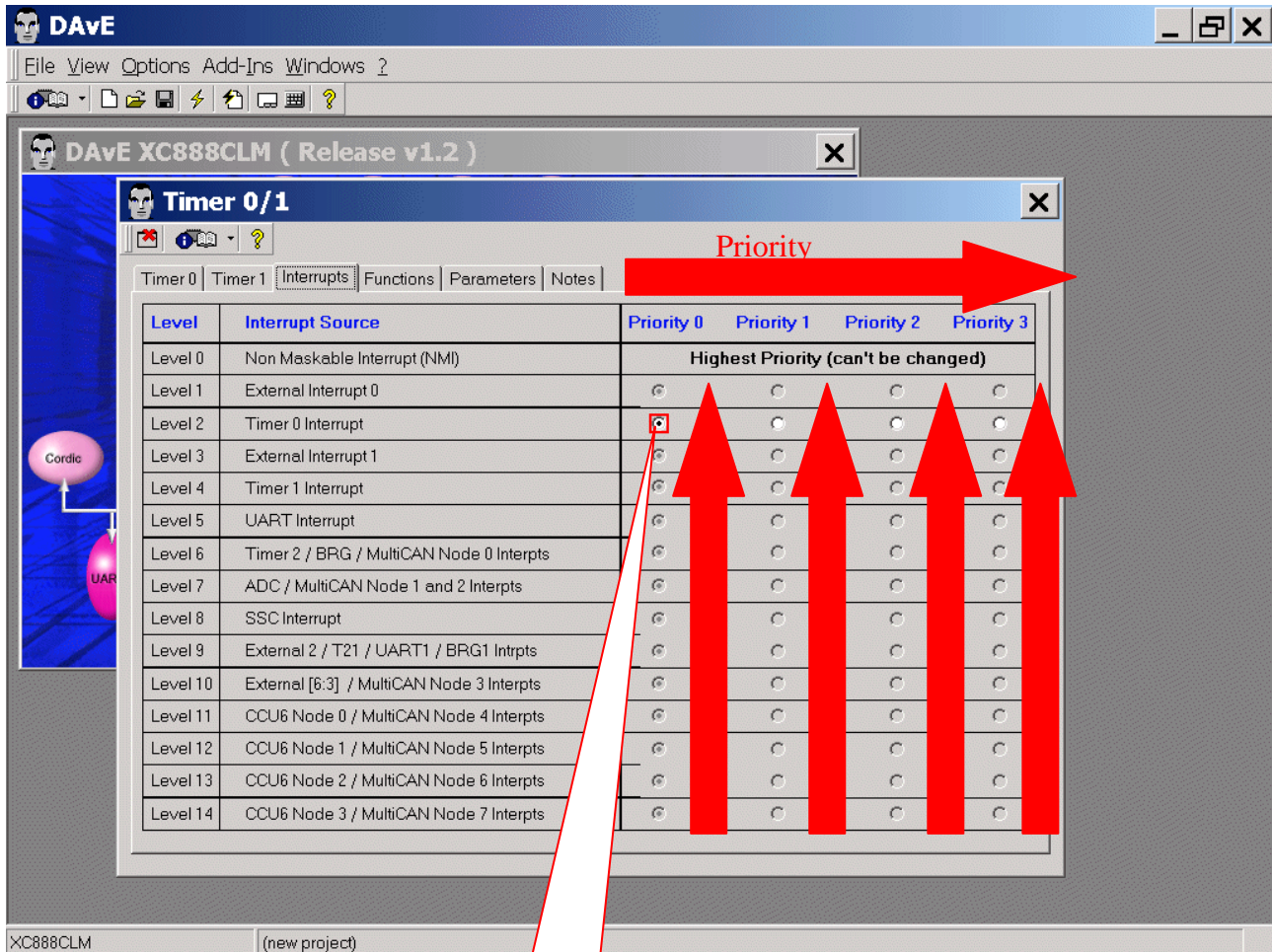
$$183 * 5461,333 \mu\text{s} = 0,9994 \text{ s.}$$



Timer1: do nothing (not used)



Interrupts: (do nothing)



Interrupt of Timer\_0 is enabled, ET0 = 1

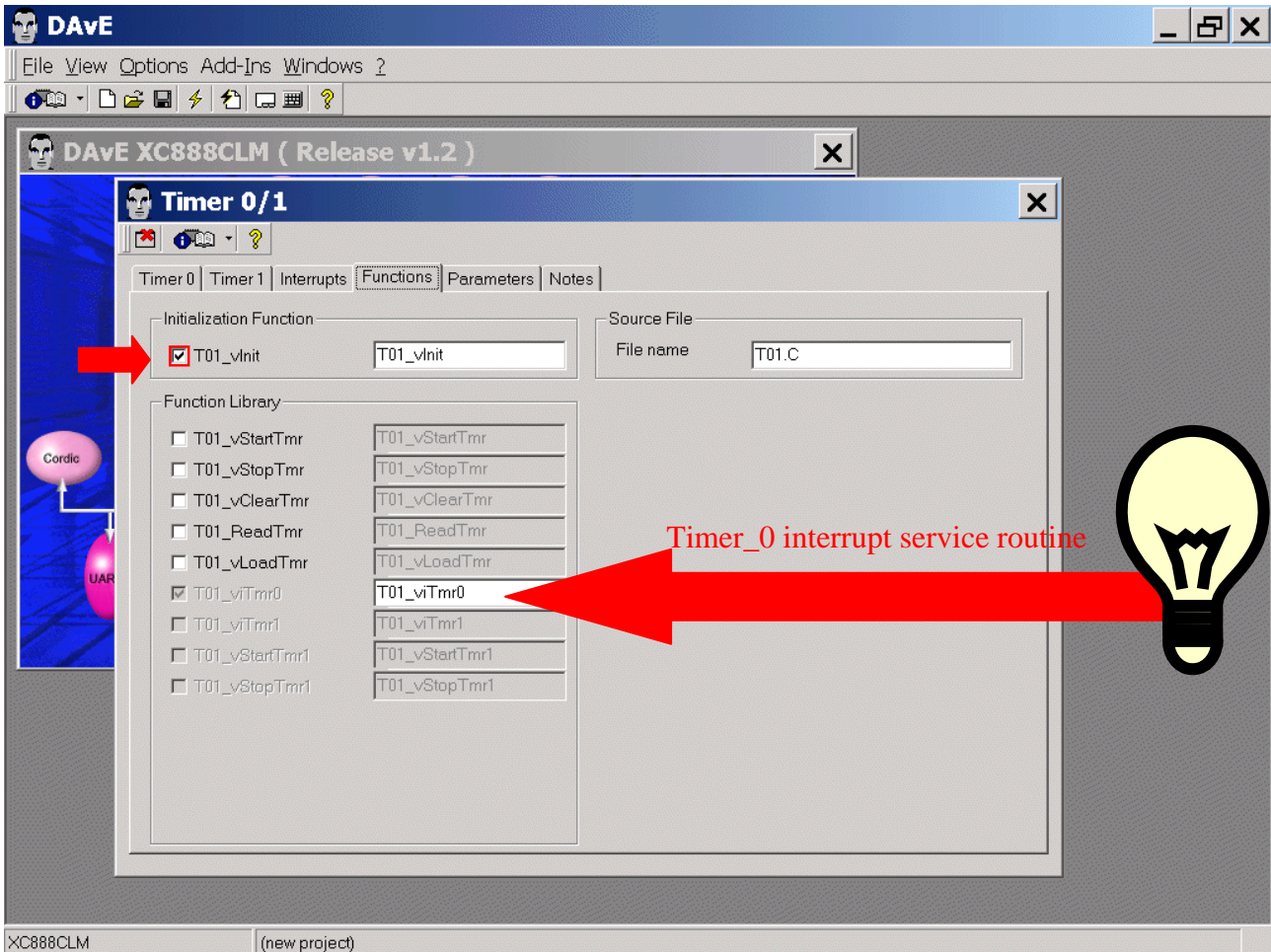


**Note (Source: User's Manual):**

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: **click** ✓ T01\_vInit



**Note:**

Timer\_0 has a dedicated interrupt vector address (000B<sub>H</sub>), 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<sub>H</sub>.

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.

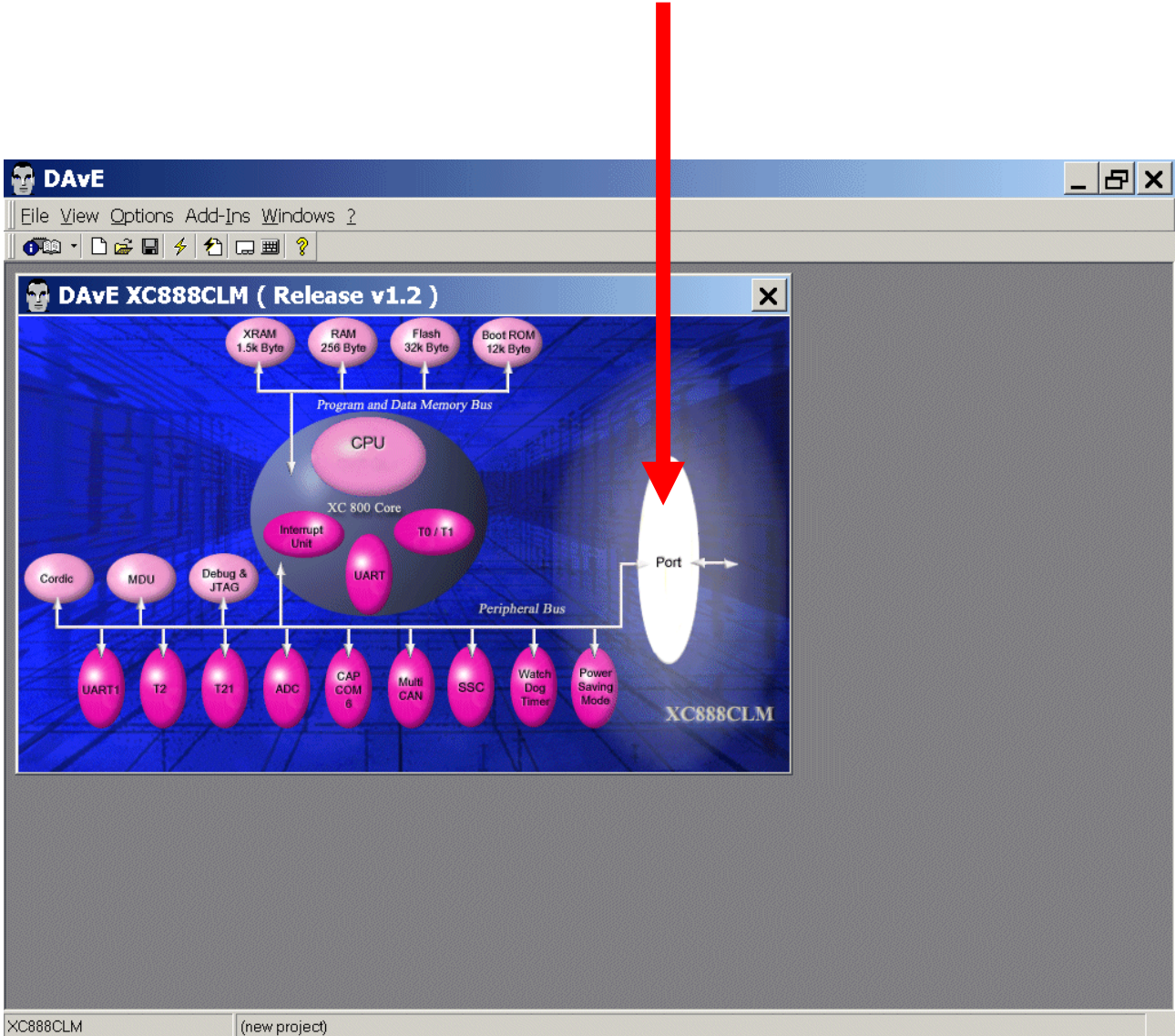
**Parameters:** (do nothing)

**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.

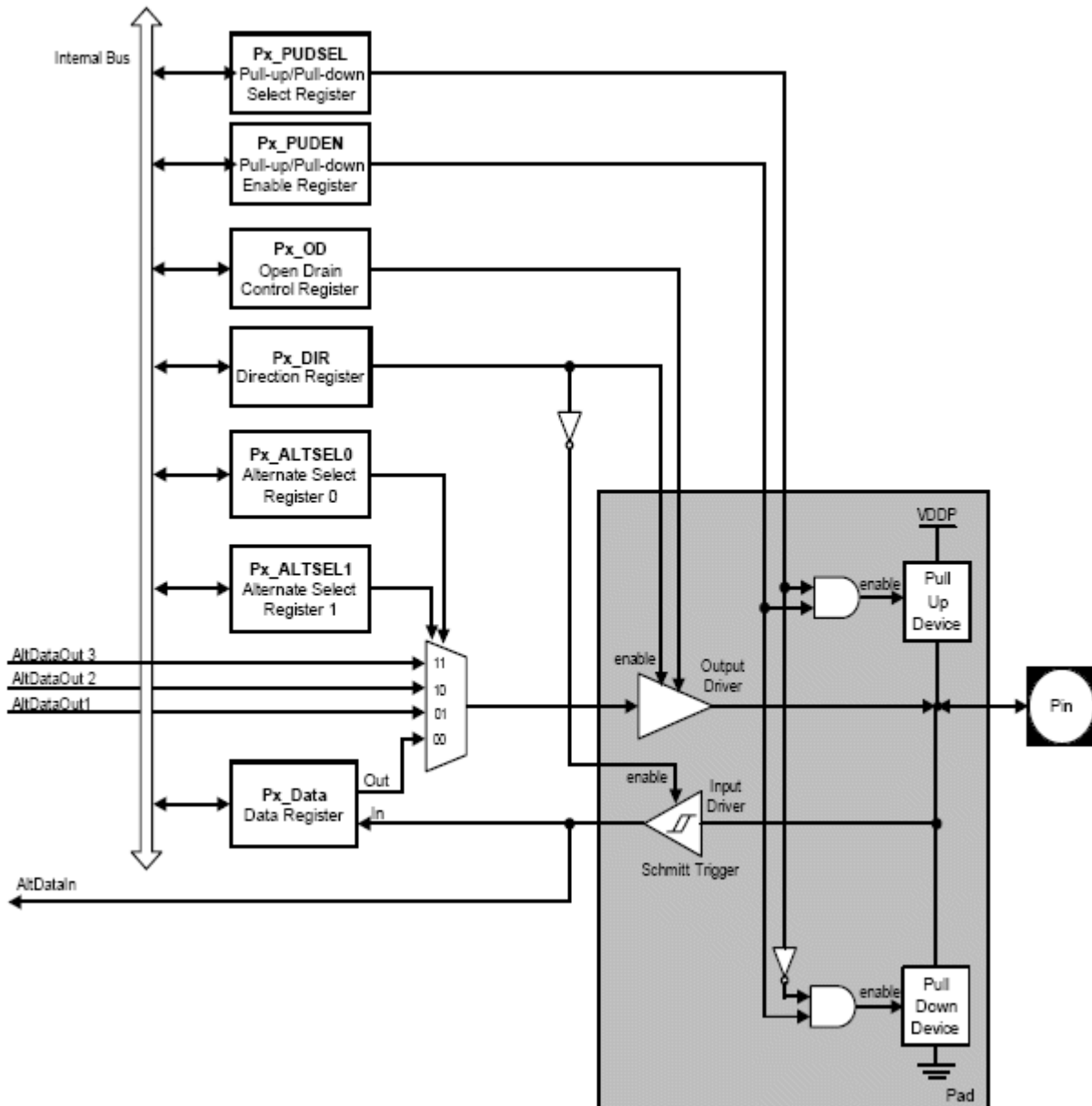


**Note:**  
The LEDs are connected to IO\_Port\_3.

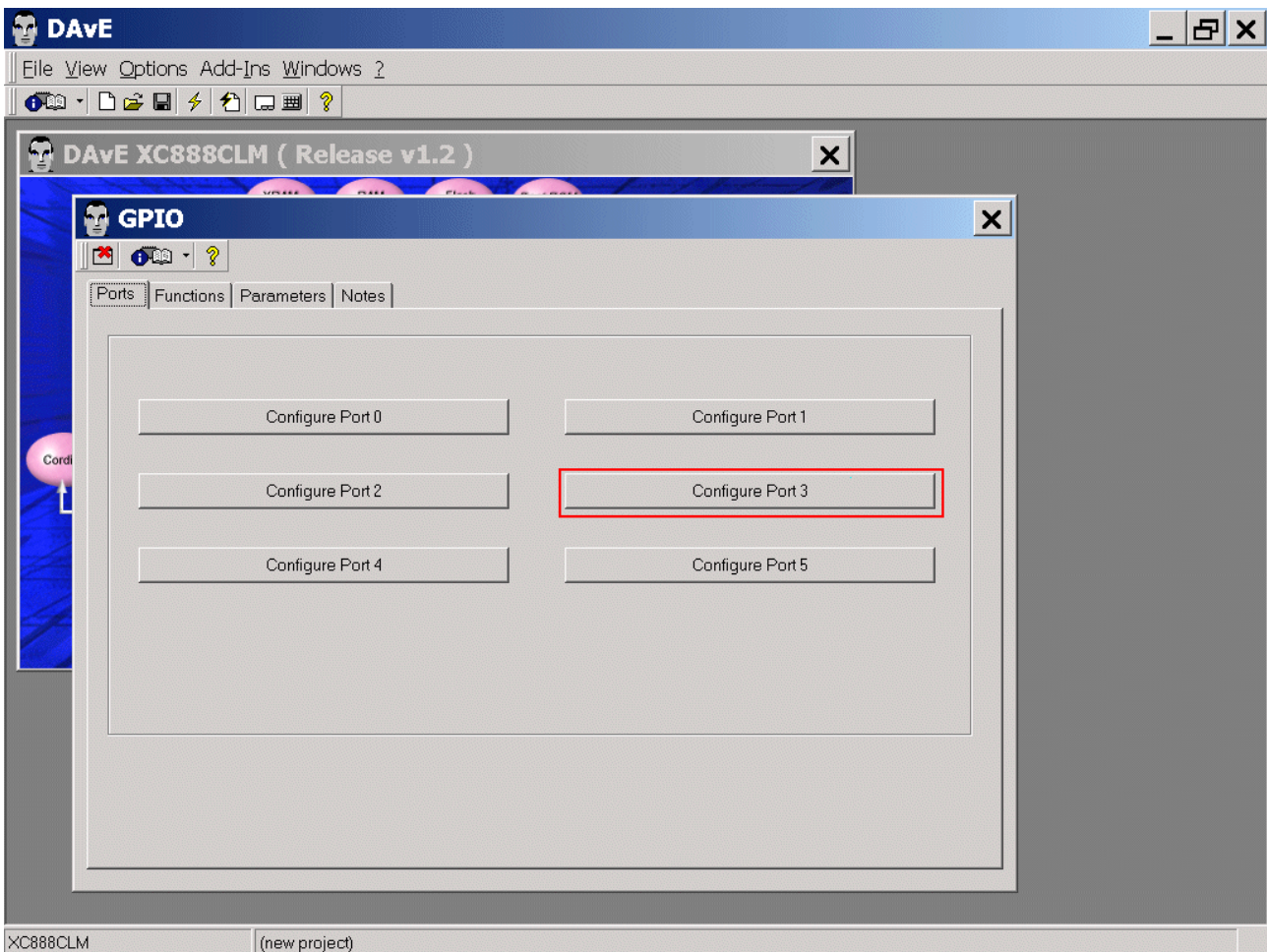




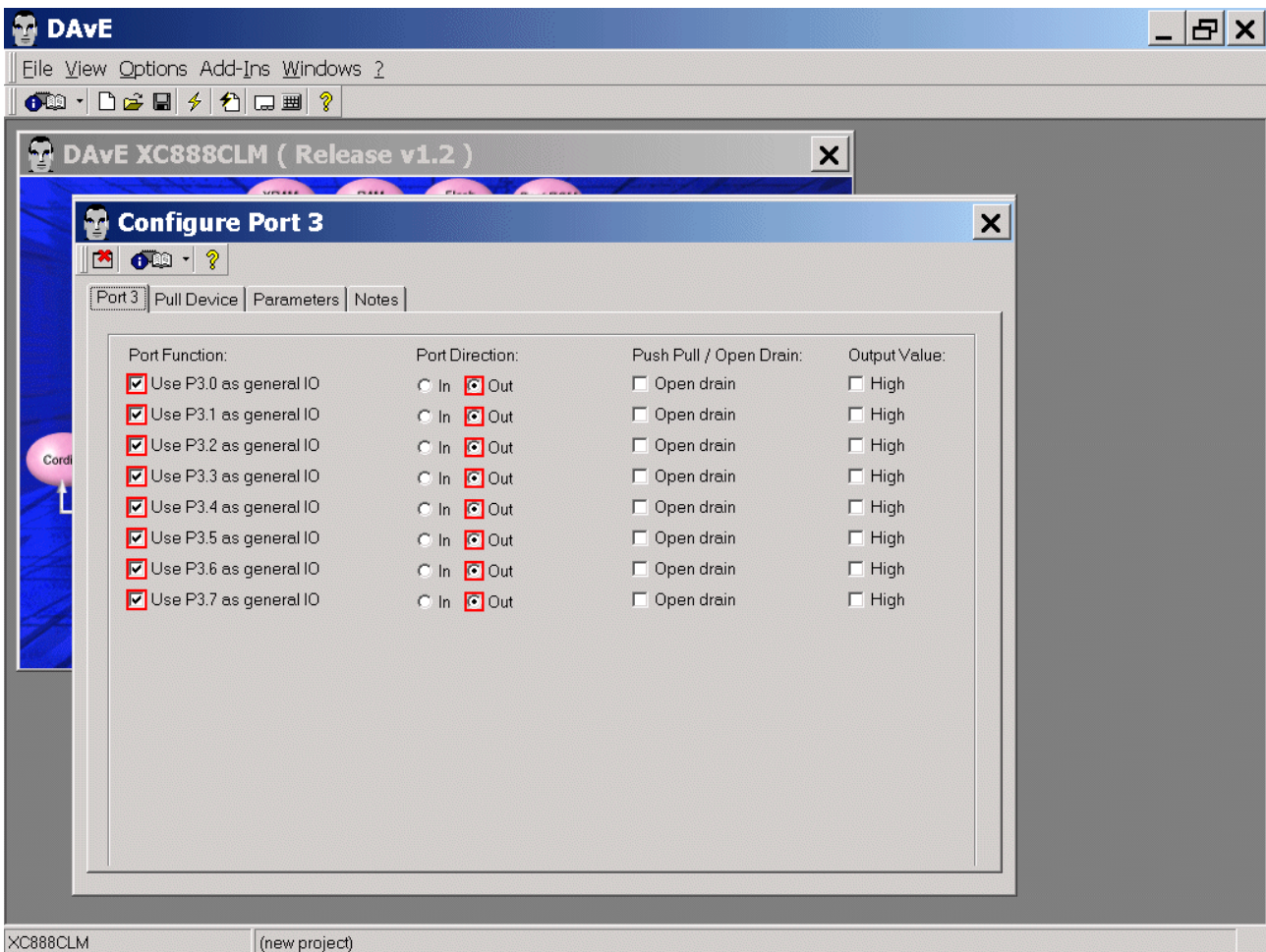
Additional information: Parallel Ports – General Structure (Source: User's Manual):



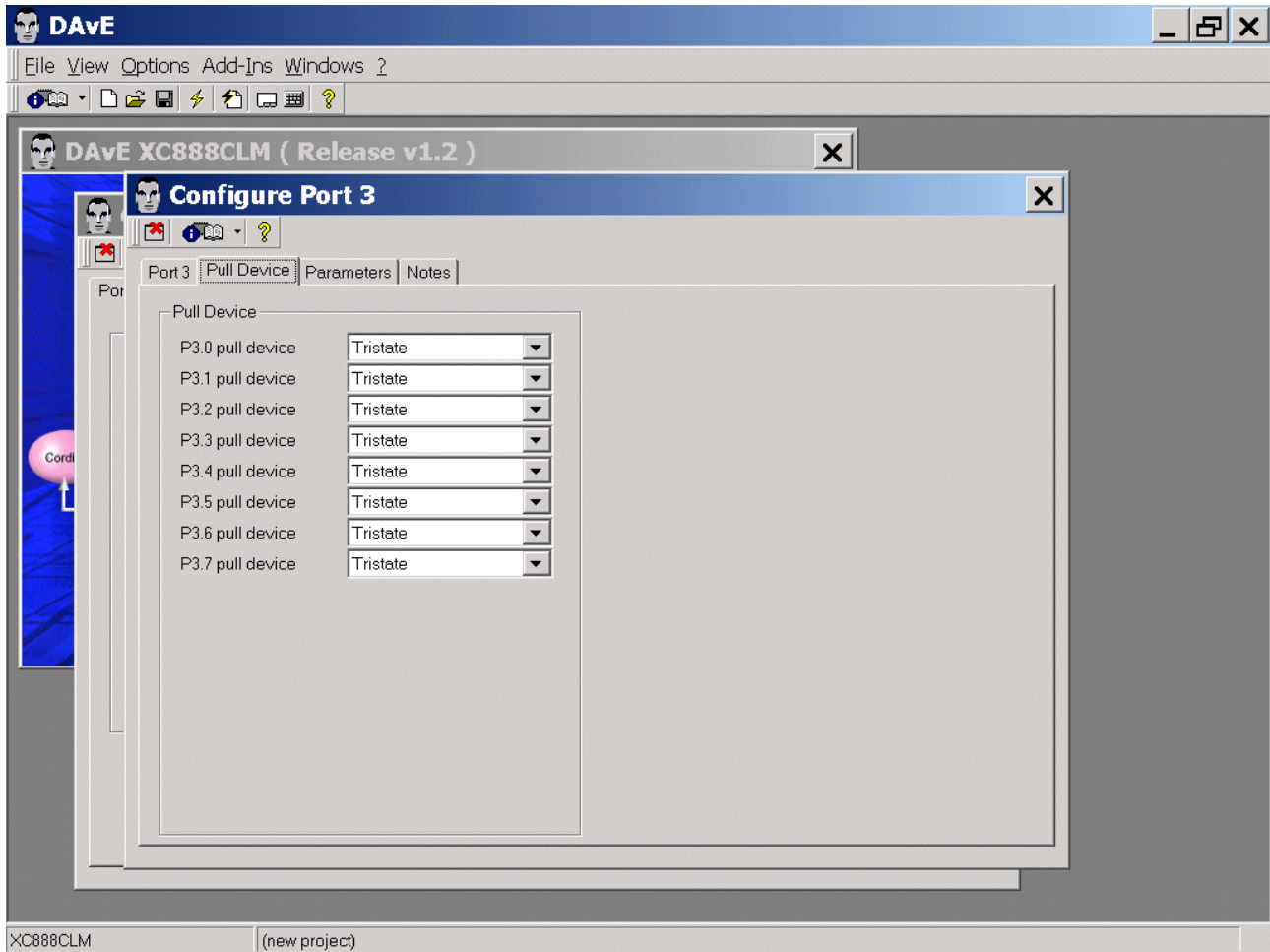
Ports: click "Configure Port 3"



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

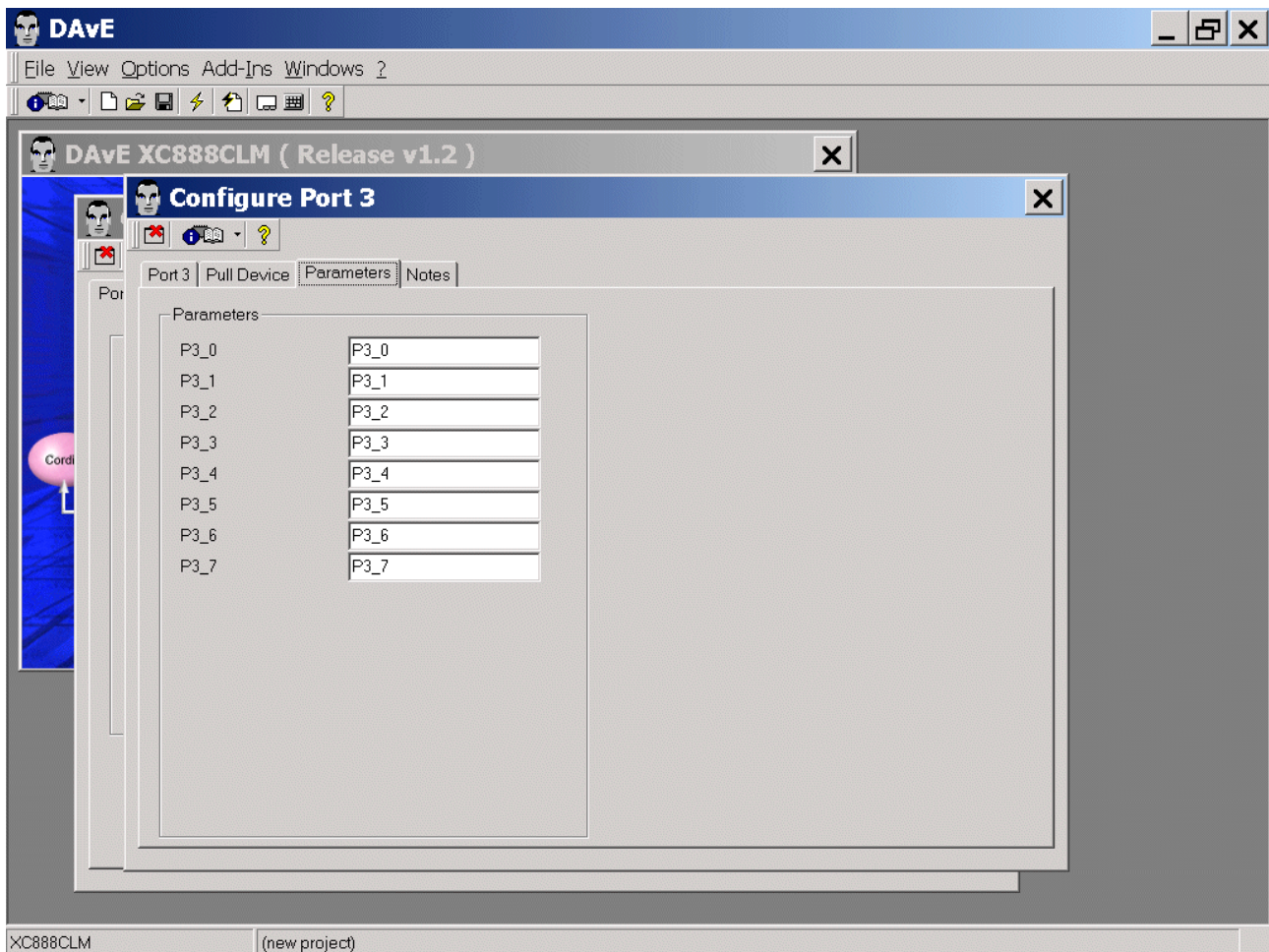


Pull Device: (do nothing)





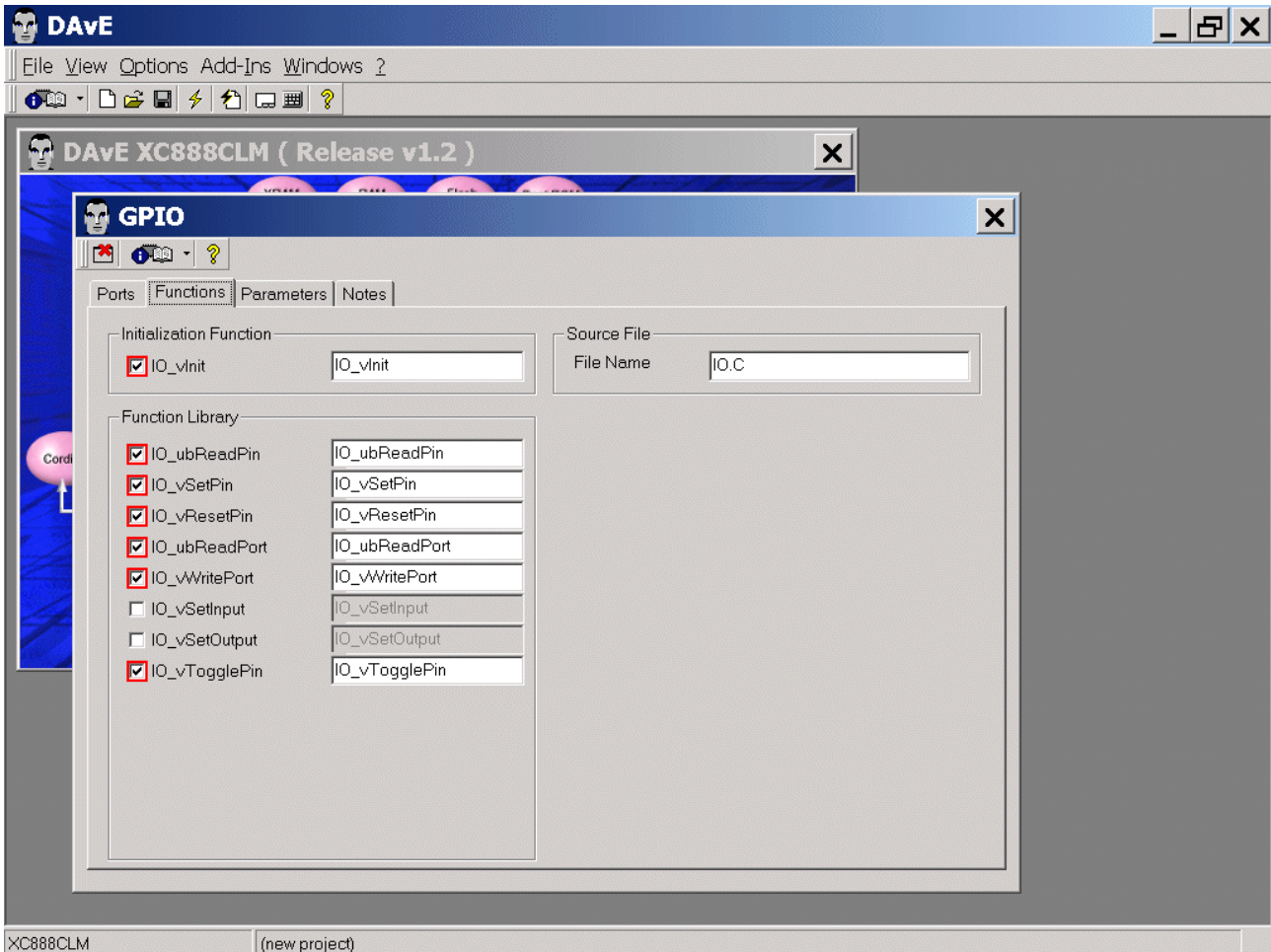
Parameters: (do nothing)




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

**Exit** this dialog now by clicking  the close button.

Functions: Initialization Functions: **click** ✓ IO\_vInit  
 Functions: Function Library: **click** ✓ IO\_ubReadPin  
 Functions: Function Library: **click** ✓ IO\_vSetPin  
 Functions: Function Library: **click** ✓ IO\_vResetPin  
 Functions: Function Library: **click** ✓ IO\_ubReadPort  
 Functions: Function Library: **click** ✓ IO\_vWritePort  
 Functions: Function Library: **click** ✓ IO\_vTogglePin

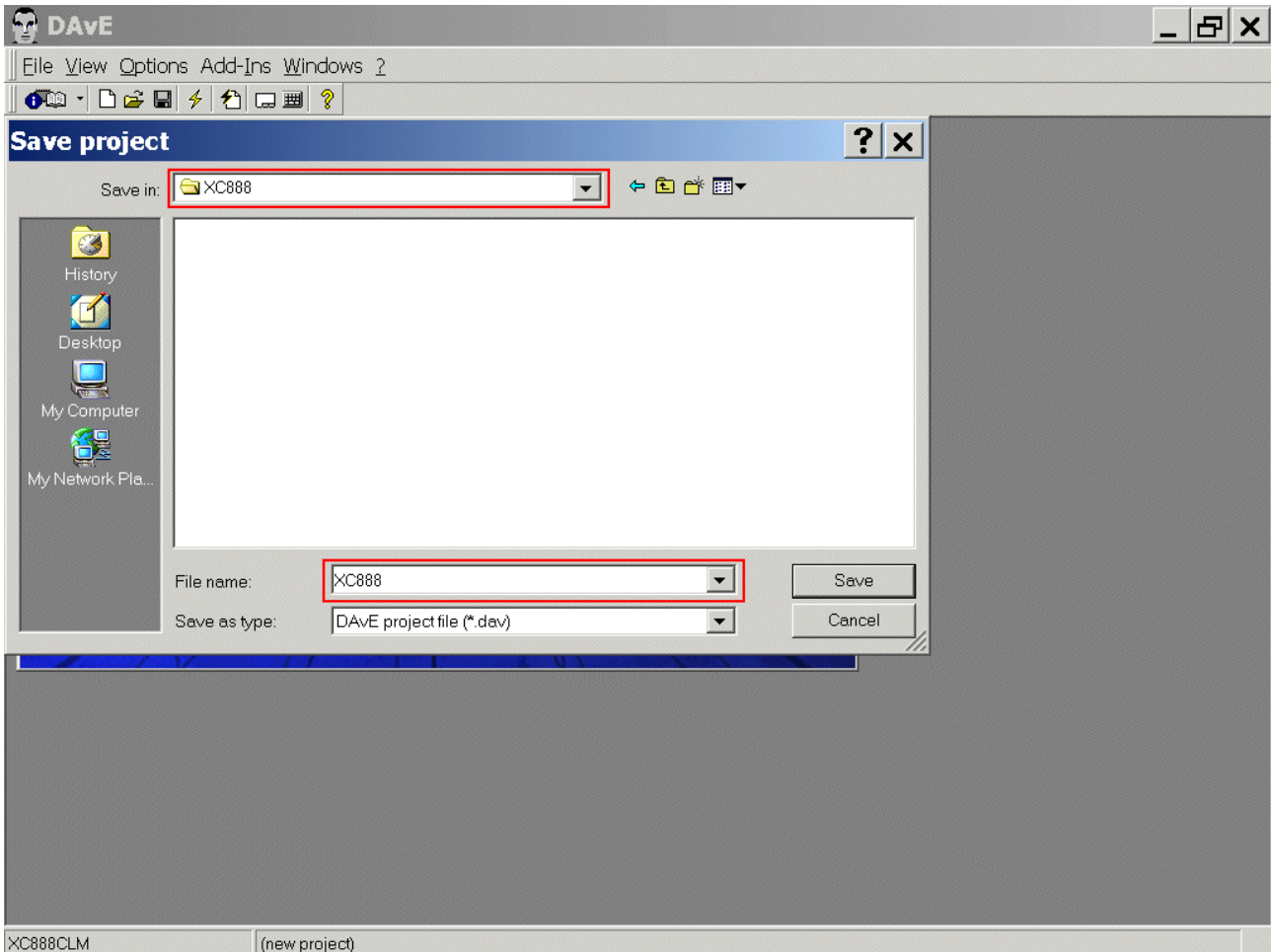


Parameters: (do nothing)  
 Notes: If you wish, you can insert your comments here.  
 Exit this dialog now by clicking  the close button.

Save the project:


File  
Save

Save project: Save in C:\XC888 (create new directory)  
File name: XC888



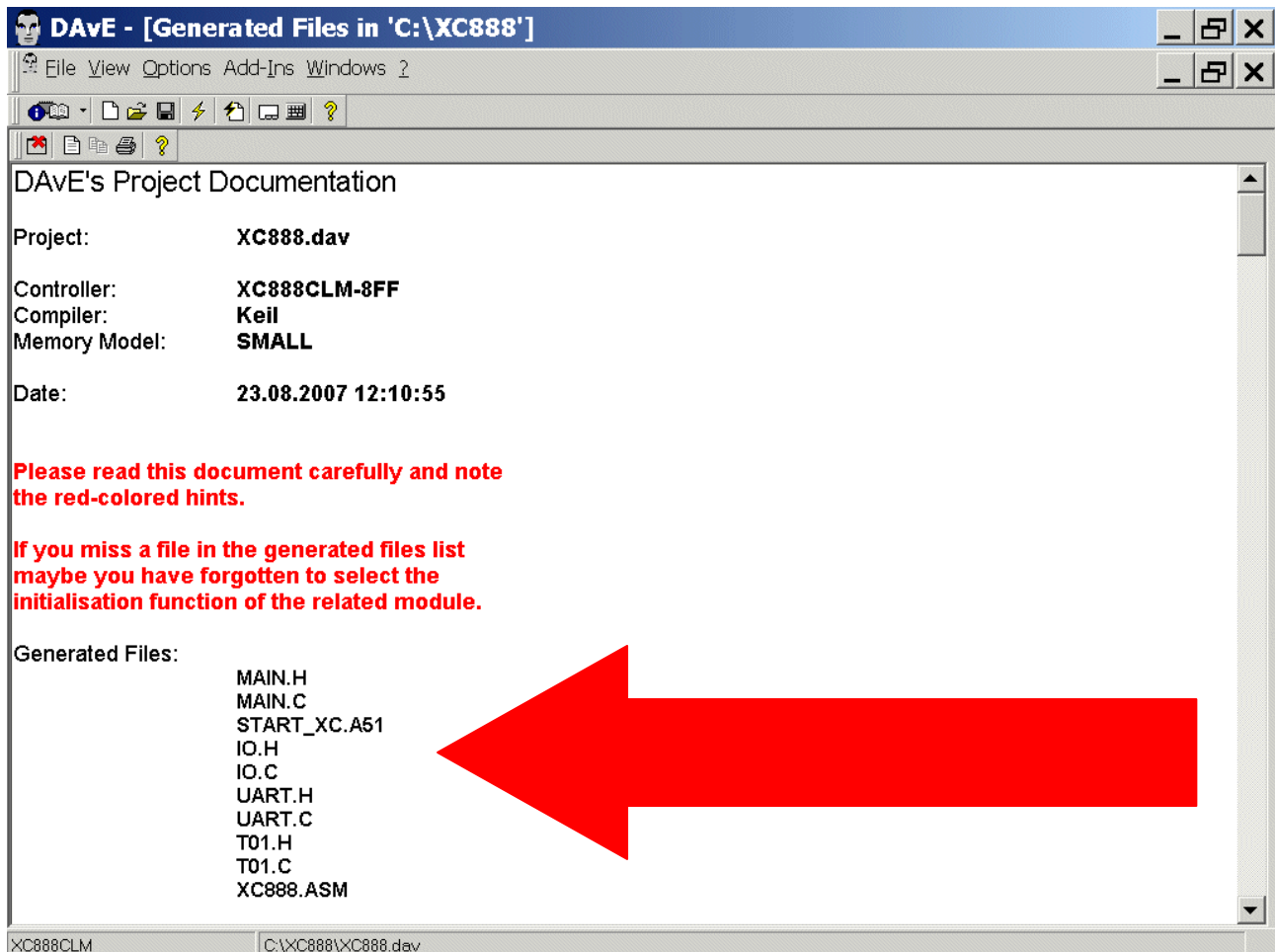
Save

Generate Code:

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



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



**File - Exit**

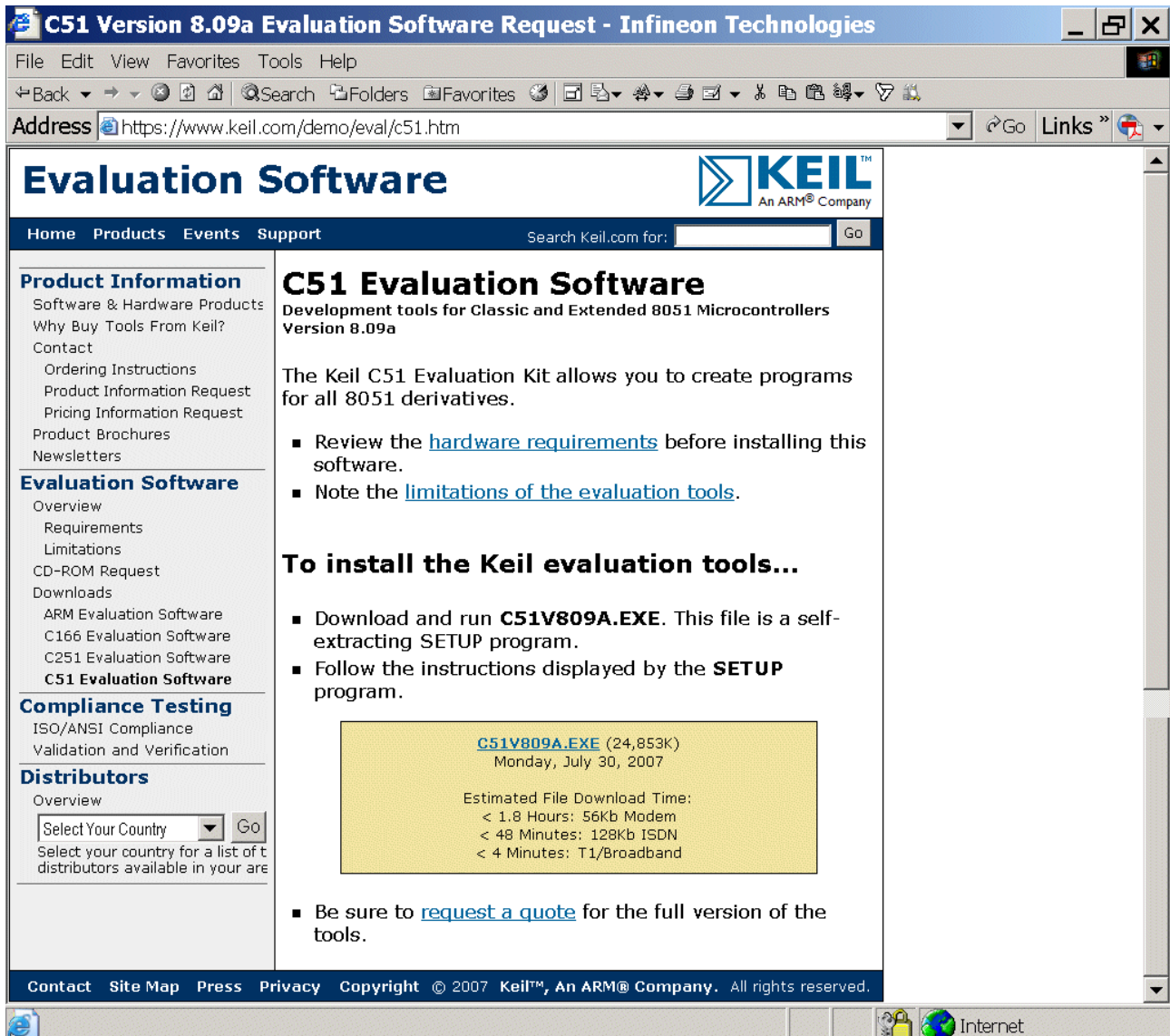
**Save changes?**

**click Yes**

#### 4.) Using the KEIL - $\mu$ Vision 3 Development Tools:

Install the Tool chain:

You can download the Keil Development Tools @ <http://www.keil.com/demo/eval/c51.htm>



**C51 Version 8.09a Evaluation Software Request - Infineon Technologies**

File Edit View Favorites Tools Help

Address <https://www.keil.com/demo/eval/c51.htm>

## Evaluation Software

Home Products Events Support Search Keil.com for:  Go

### Product Information

- Software & Hardware Products
- Why Buy Tools From Keil?
- Contact
  - Ordering Instructions
  - Product Information Request
  - Pricing Information Request
- Product Brochures
- Newsletters

### Evaluation Software

- Overview
- Requirements
- Limitations
- CD-ROM Request
- Downloads
  - ARM Evaluation Software
  - C166 Evaluation Software
  - C251 Evaluation Software
  - C51 Evaluation Software

### Compliance Testing

- ISO/ANSI Compliance
- Validation and Verification

### Distributors

Overview

Select Your Country  Go

Select your country for a list of distributors available in your area

## C51 Evaluation Software

Development tools for Classic and Extended 8051 Microcontrollers  
Version 8.09a

The Keil C51 Evaluation Kit allows you to create programs for all 8051 derivatives.

- Review the [hardware requirements](#) before installing this software.
- Note the [limitations of the evaluation tools](#).

### To install the Keil evaluation tools...

- Download and run **C51V809A.EXE**. This file is a self-extracting SETUP program.
- Follow the instructions displayed by the **SETUP** program.

[C51V809A.EXE](#) (24,853K)  
Monday, July 30, 2007

Estimated File Download Time:

- < 1.8 Hours: 56Kb Modem
- < 48 Minutes: 128Kb ISDN
- < 4 Minutes: T1/Broadband

- Be sure to [request a quote](#) for the full version of the tools.

Contact Site Map Press Privacy Copyright © 2007 Keil™, An ARM® Company. All rights reserved.

Execute **C51V809A.EXE** ( - or any higher version )



Start Keil  $\mu$ Vision3 and open the DAVE Project:

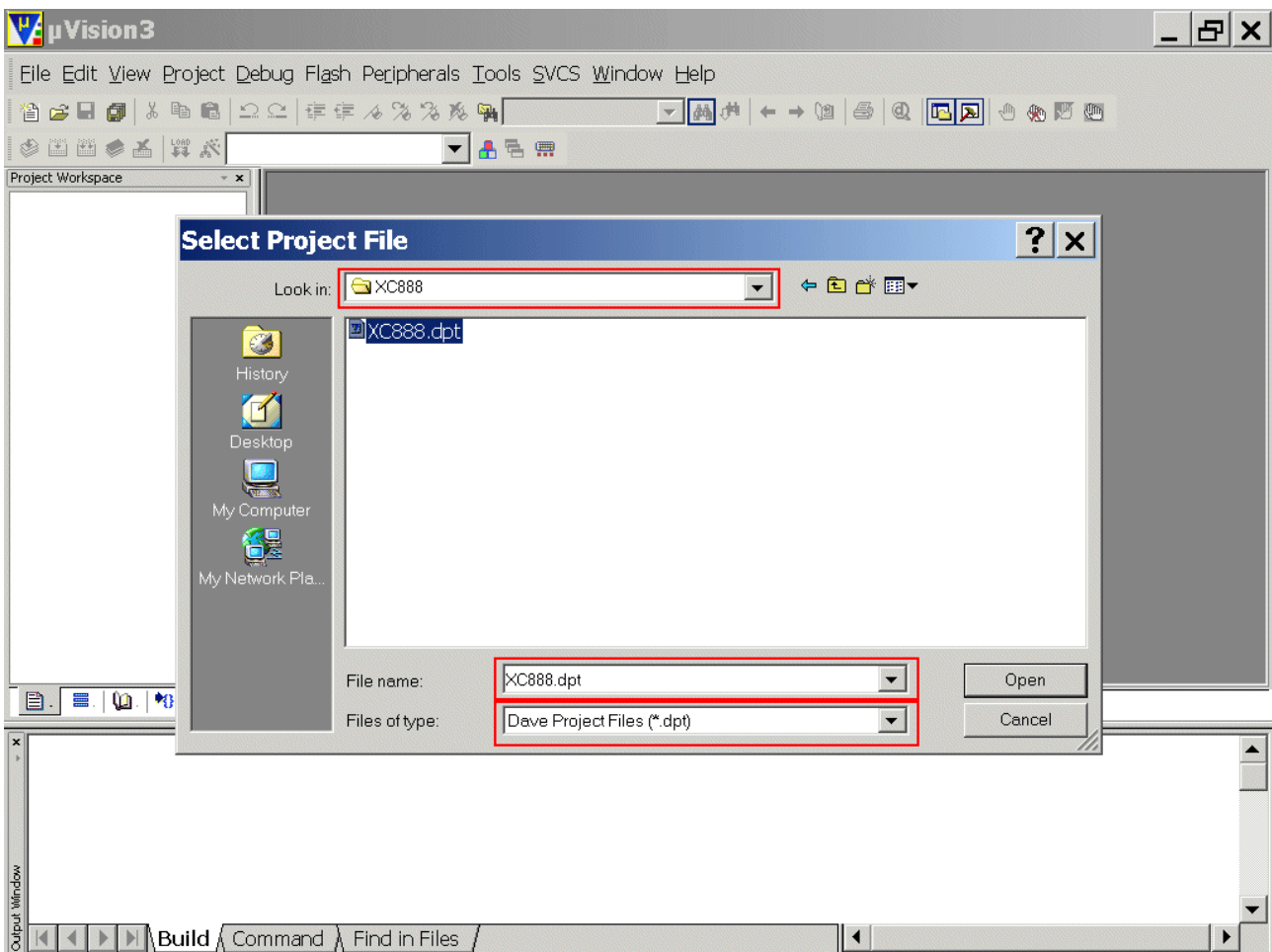
If you see an open project – close it: **Project - Close Project**

**Project - Open Project**

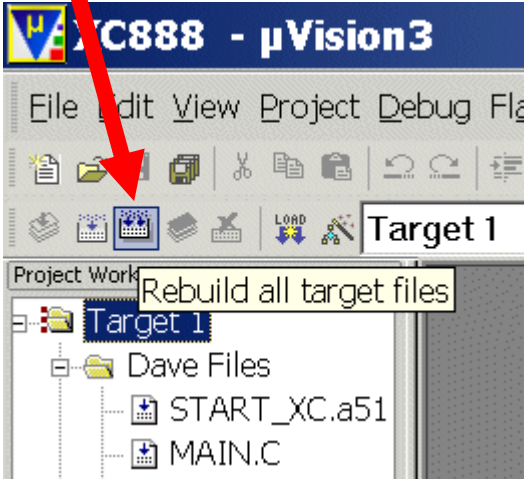
Select Project File: **Look in:** choose C:\XC888

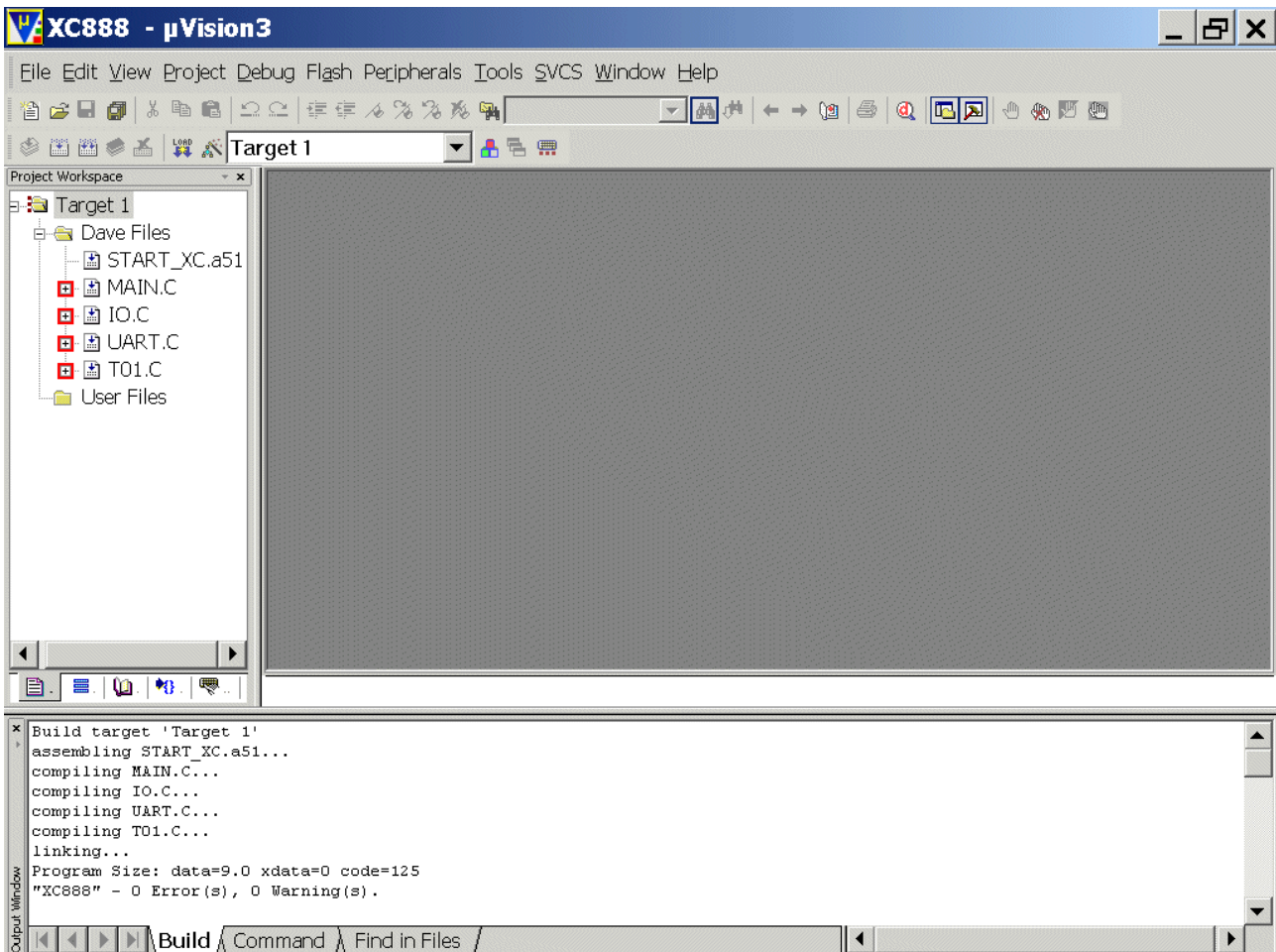
Select Project File: **Files of type:** choose Dave Project Files

Choose XC888.dpt

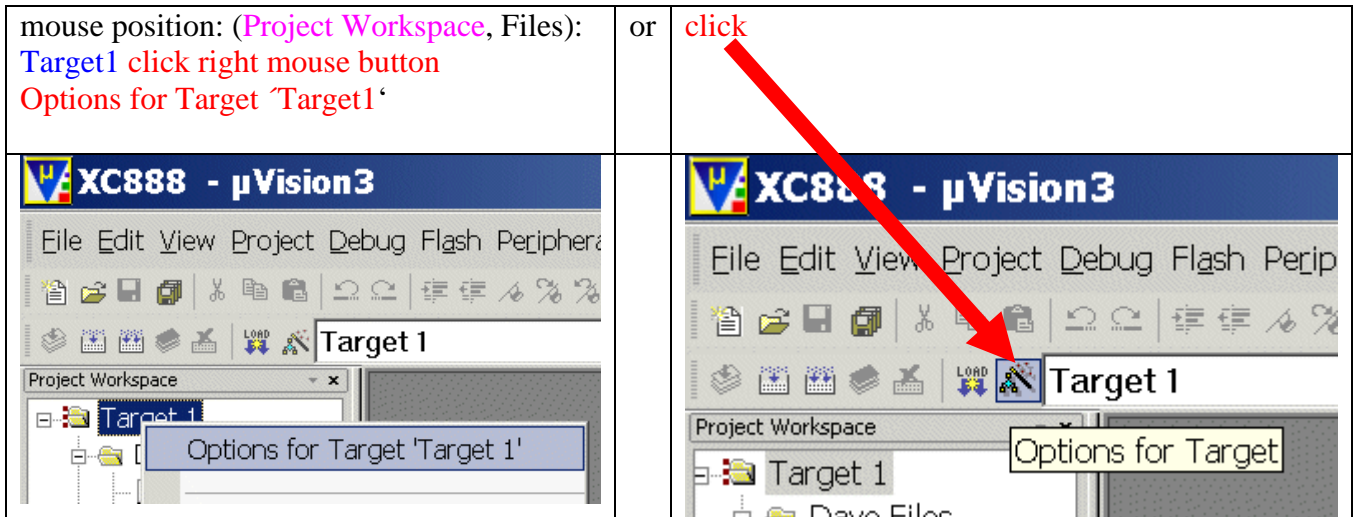


**Open**

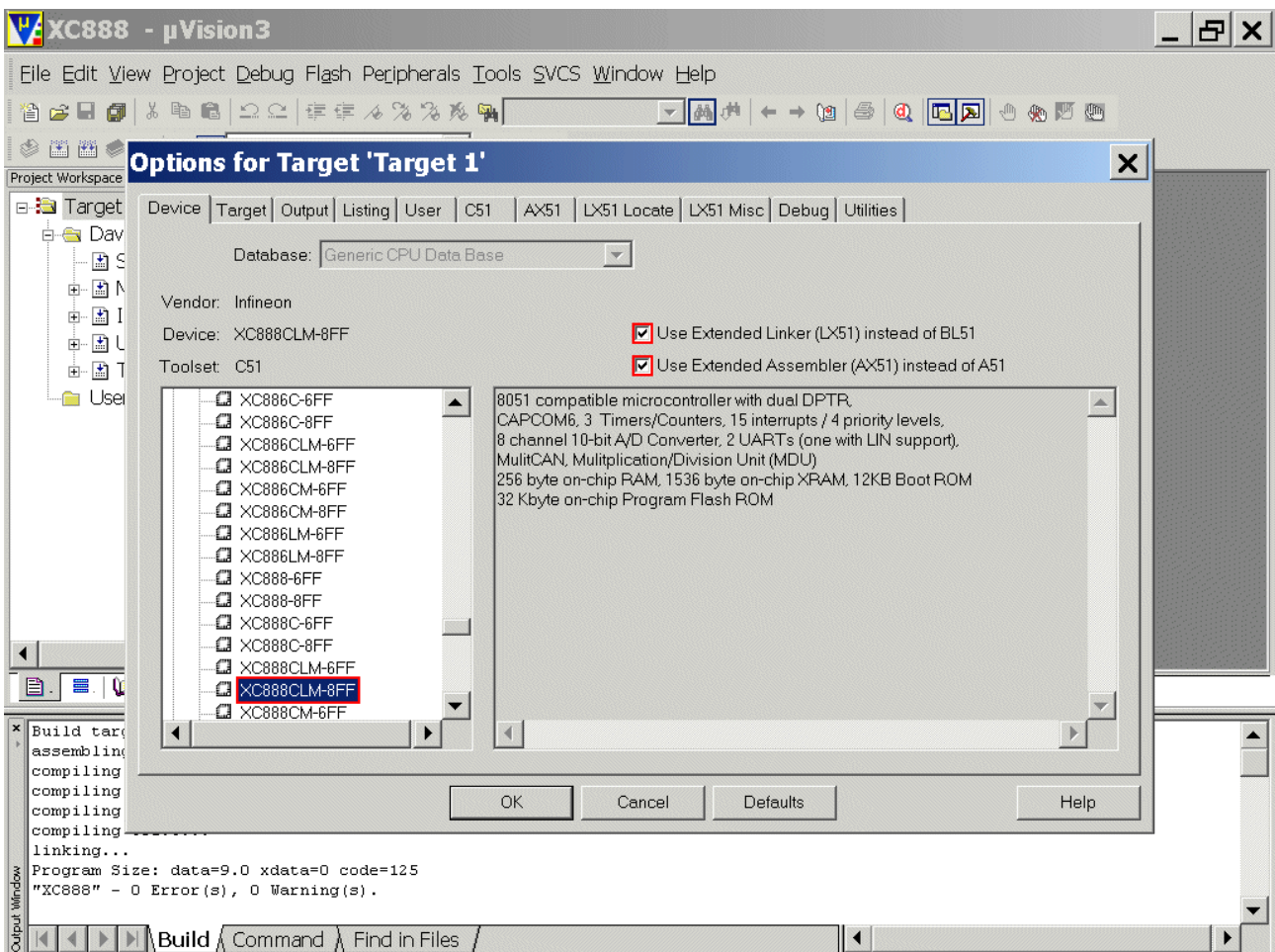
<p>Project – Rebuild all target files</p>	<p>or</p>	<p>click</p> 
---	-----------	---



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

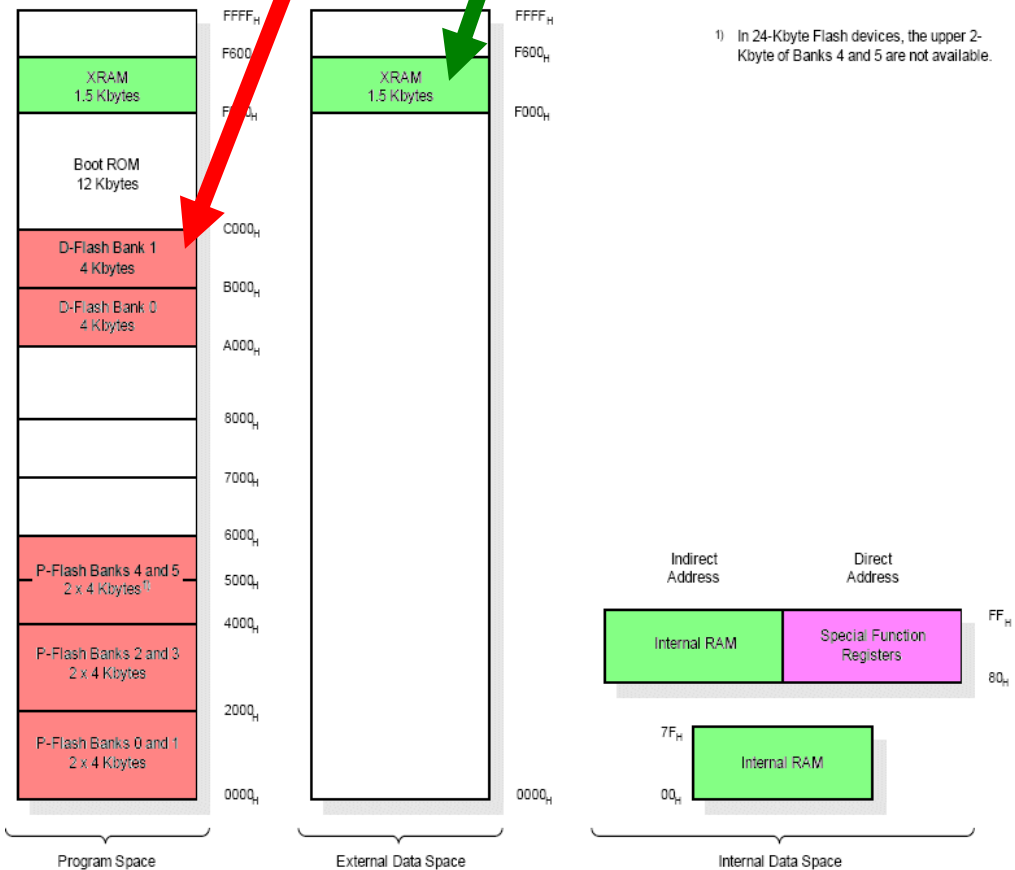
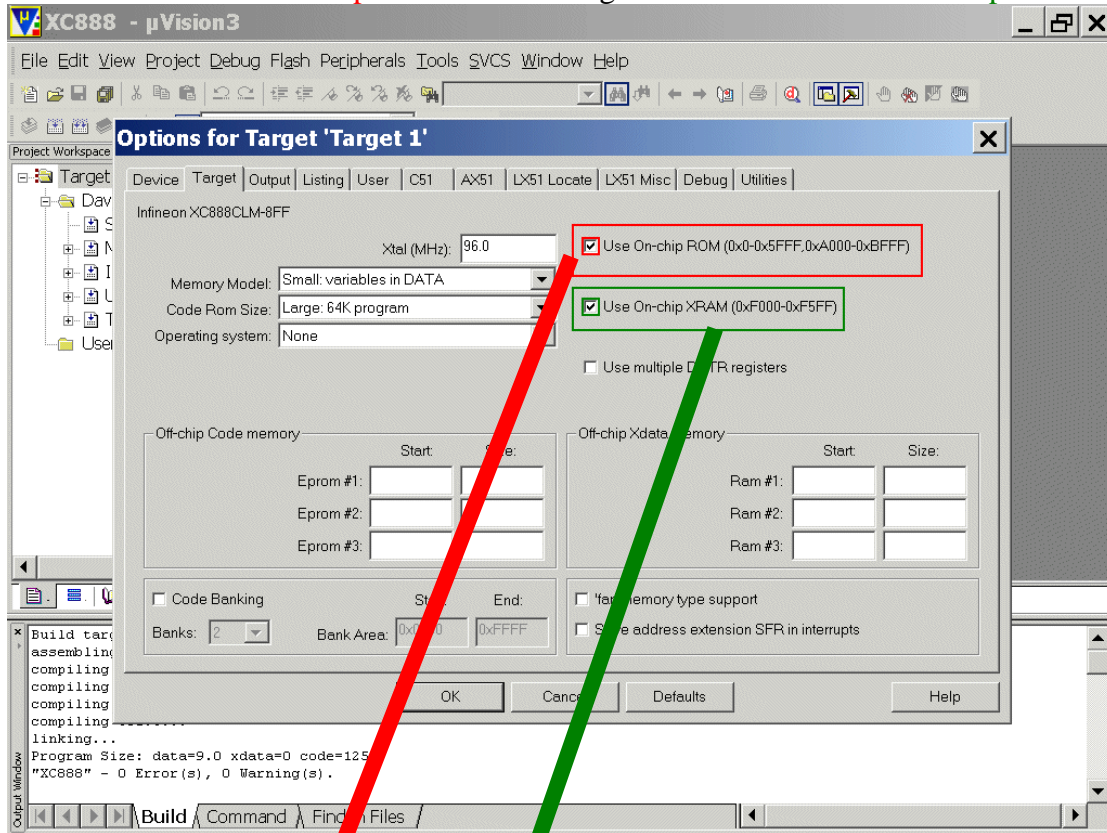


- Options for Target 'Target 1': Device: **check** XC888CLM-8FF
- Options for Target 'Target 1': Device: **click** ✓ Use Extended Linker (LX51)
- Options for Target 'Target 1': Device: **click** ✓ Use Extended Assembler (AX51)



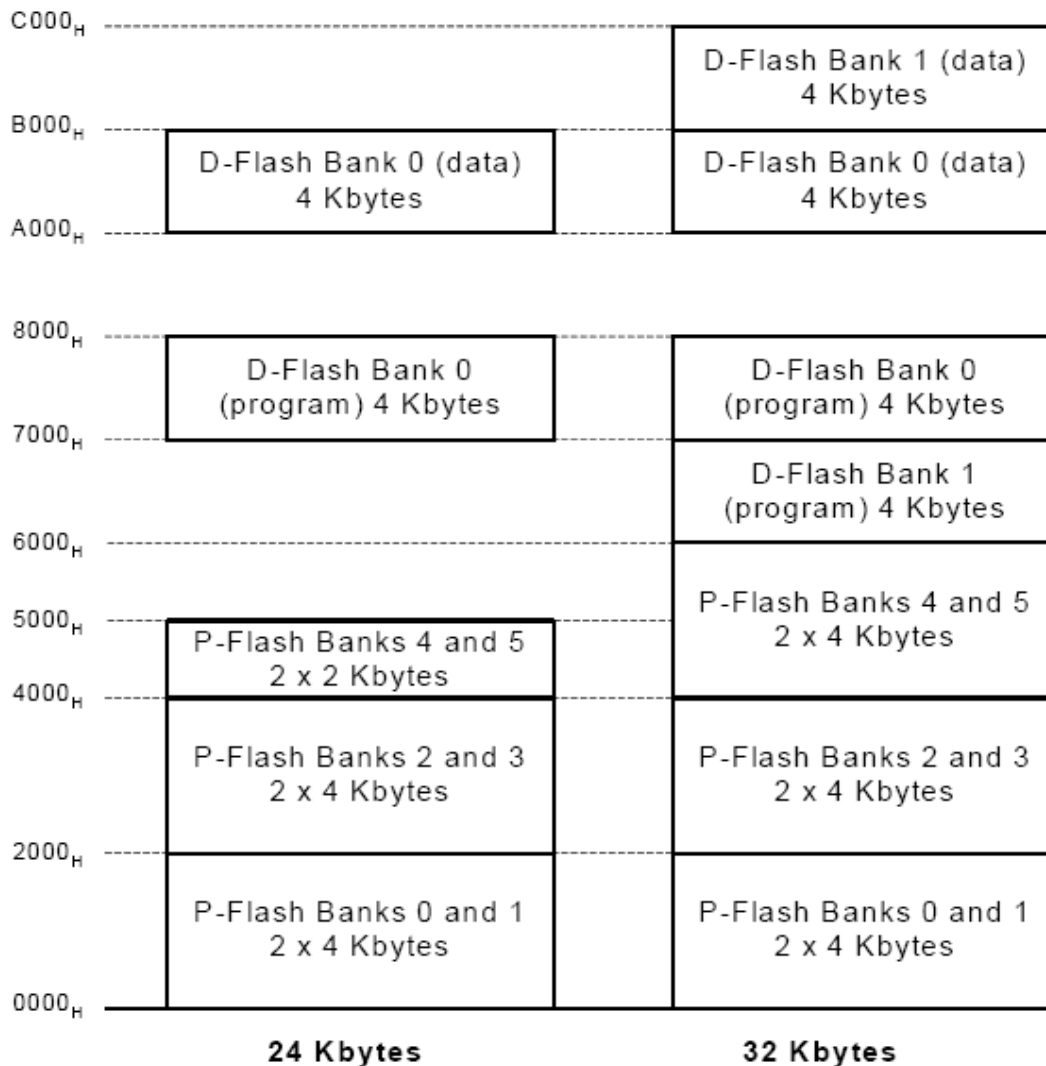


Target: click/check ✓ Use On-chip ROM & Target: click/check ✓ Use On-chip XRAM





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:

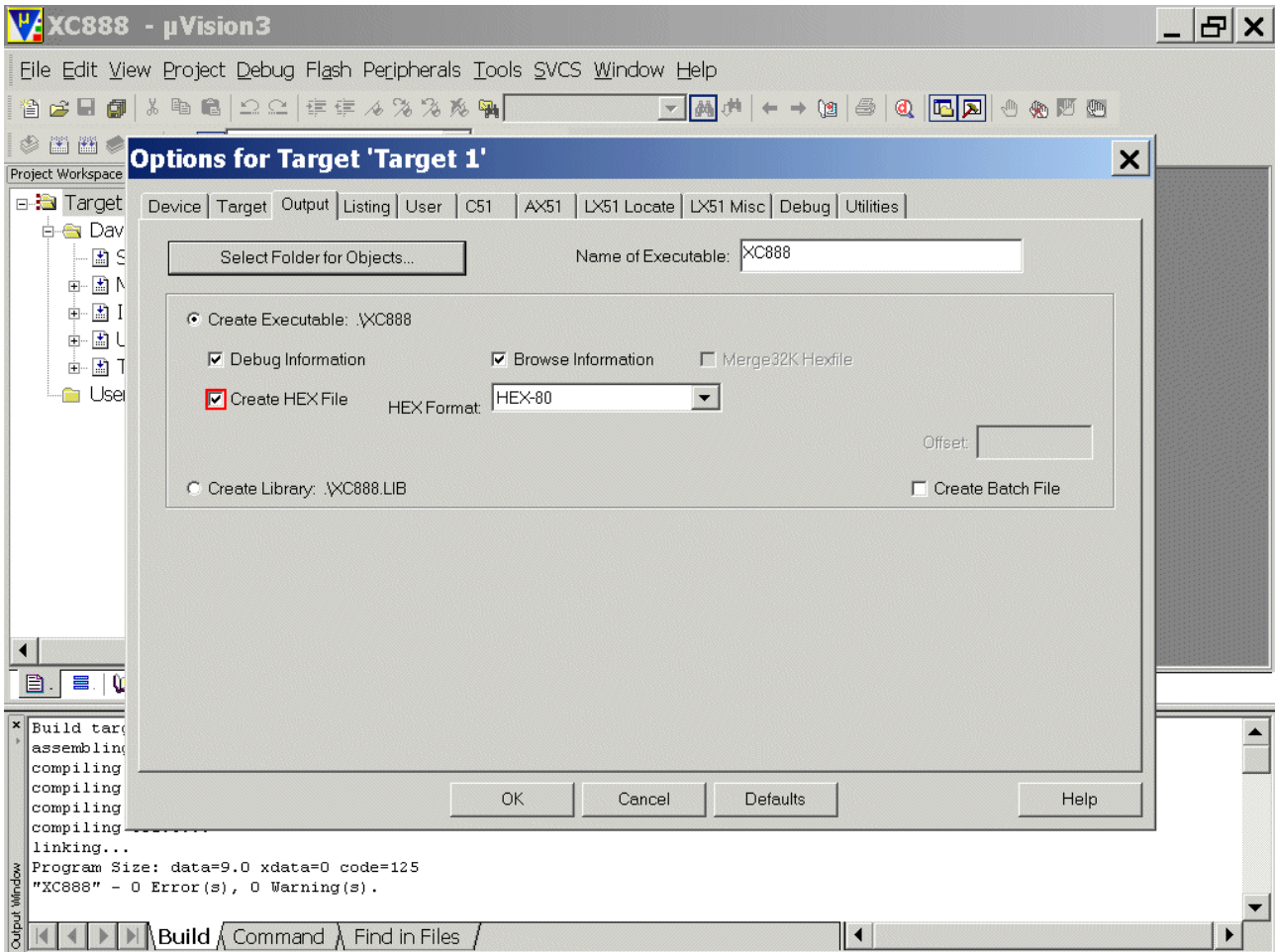
D-Flash Bank 0 is mapped to 7000<sub>H</sub> – 7FFF<sub>H</sub> and A000<sub>H</sub> – AFFF<sub>H</sub>.

D-Flash Bank 1, which is only available in the 32-Kbyte Flash device, is mapped to 6000<sub>H</sub> – 6FFF<sub>H</sub> and B000<sub>H</sub> – BFFF<sub>H</sub>.

In general, the lower address spaces (6000<sub>H</sub> – 6FFF<sub>H</sub> and 7000<sub>H</sub> – 7FFF<sub>H</sub>) should be used for D-Flash bank(s) contents that are intended to be used as program code.

Alternatively, the higher address spaces (A000<sub>H</sub> – AFFF<sub>H</sub> and B000<sub>H</sub> – BFFF<sub>H</sub>) should be used for D-Flash bank(s) contents that are intended to be used as data.

Output: **click** ✓ Create HEX File



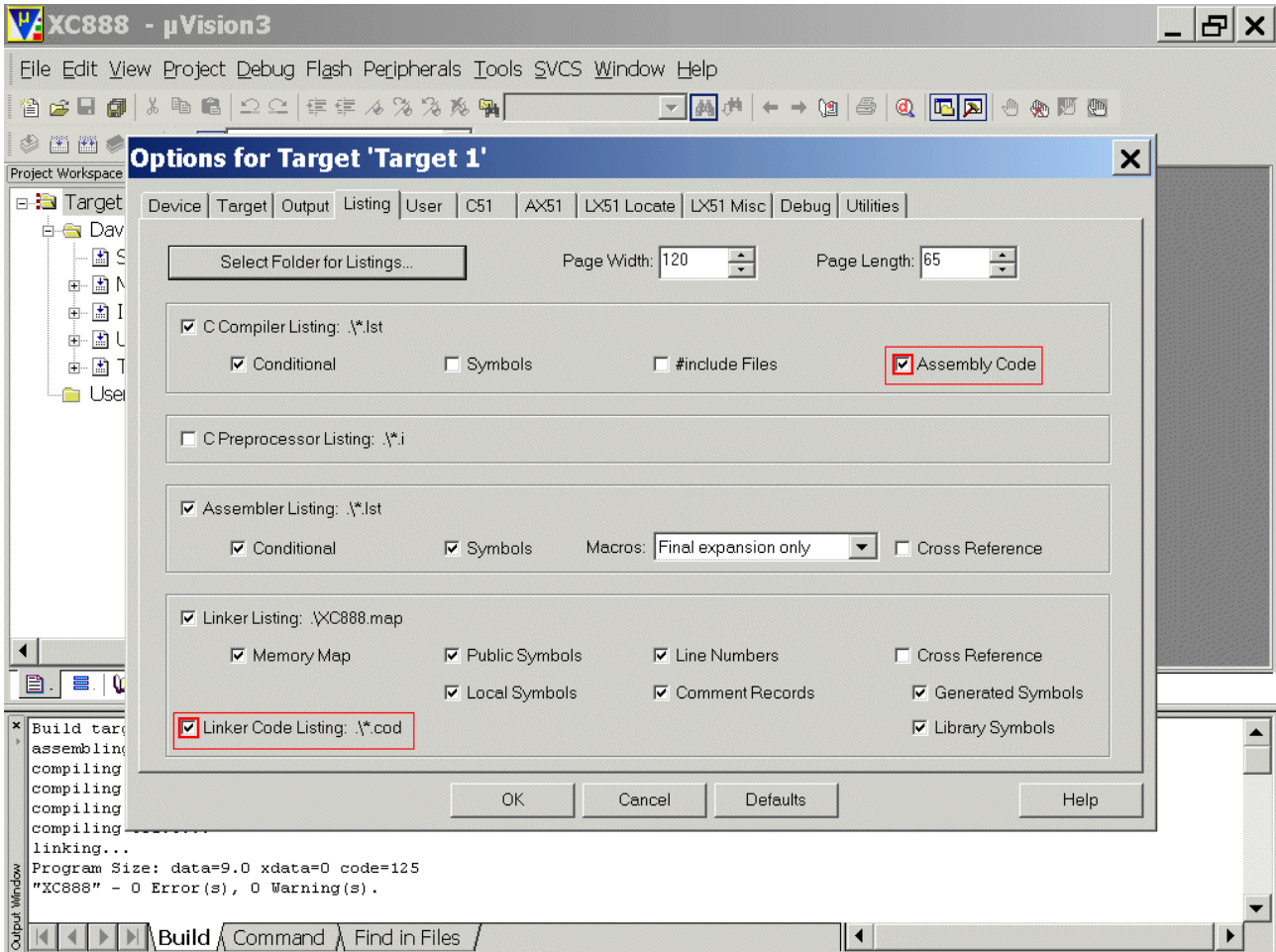
**Note:**

The HEX-File could be used while working with the program XC800\_FLOAD for OnChip-Flash-Programming via RS232-interface [Bootstrap Loader (BSL) Mode via UART].



Listing: [C Compiler Listing](#): [click](#) ✓ Assembly Code

Listing: [Linker Listing](#): [click](#) ✓ Linker Code Listing: `./*.cod`



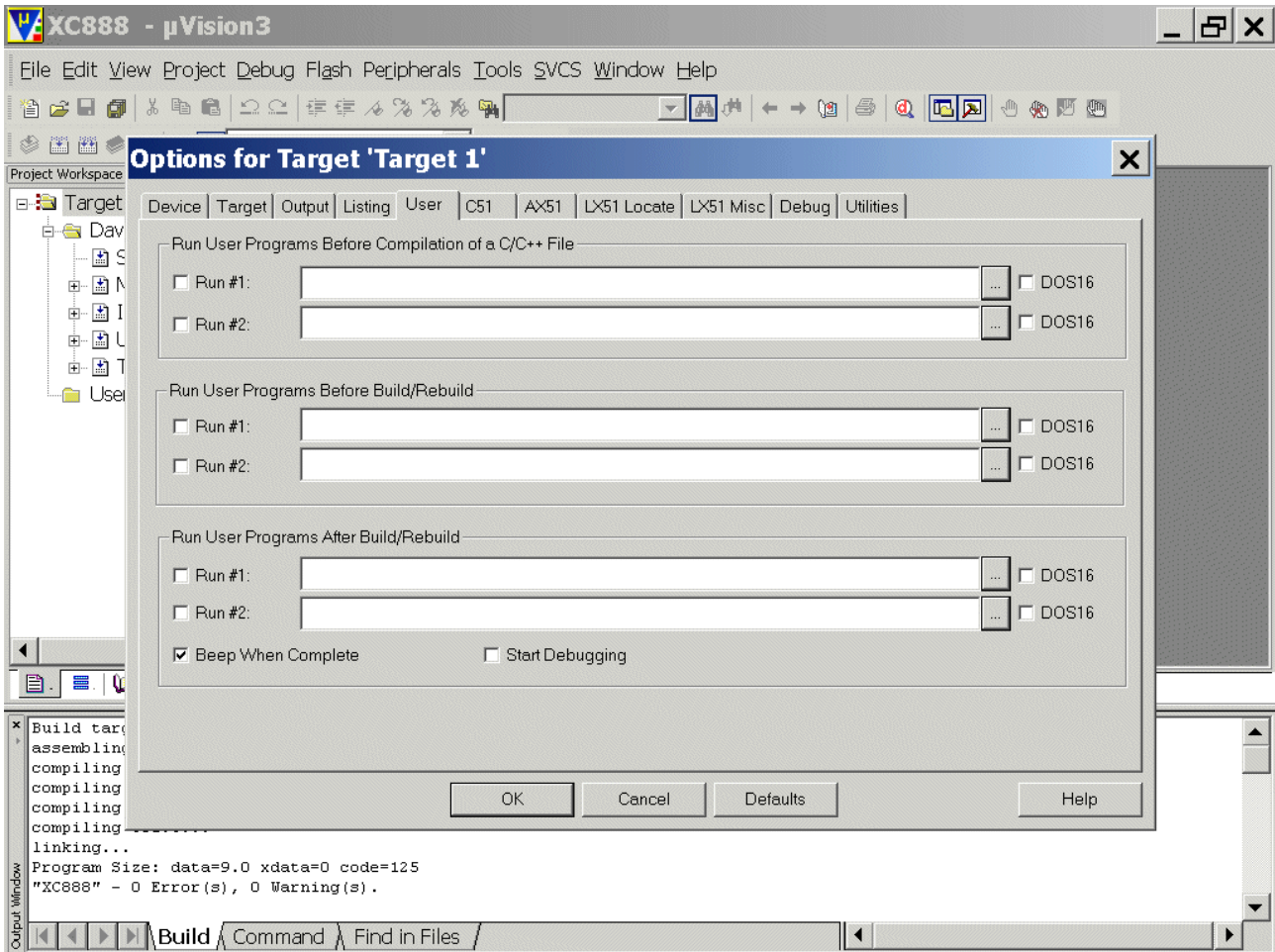
**Note:**

With the `cod`-file you can do the following:

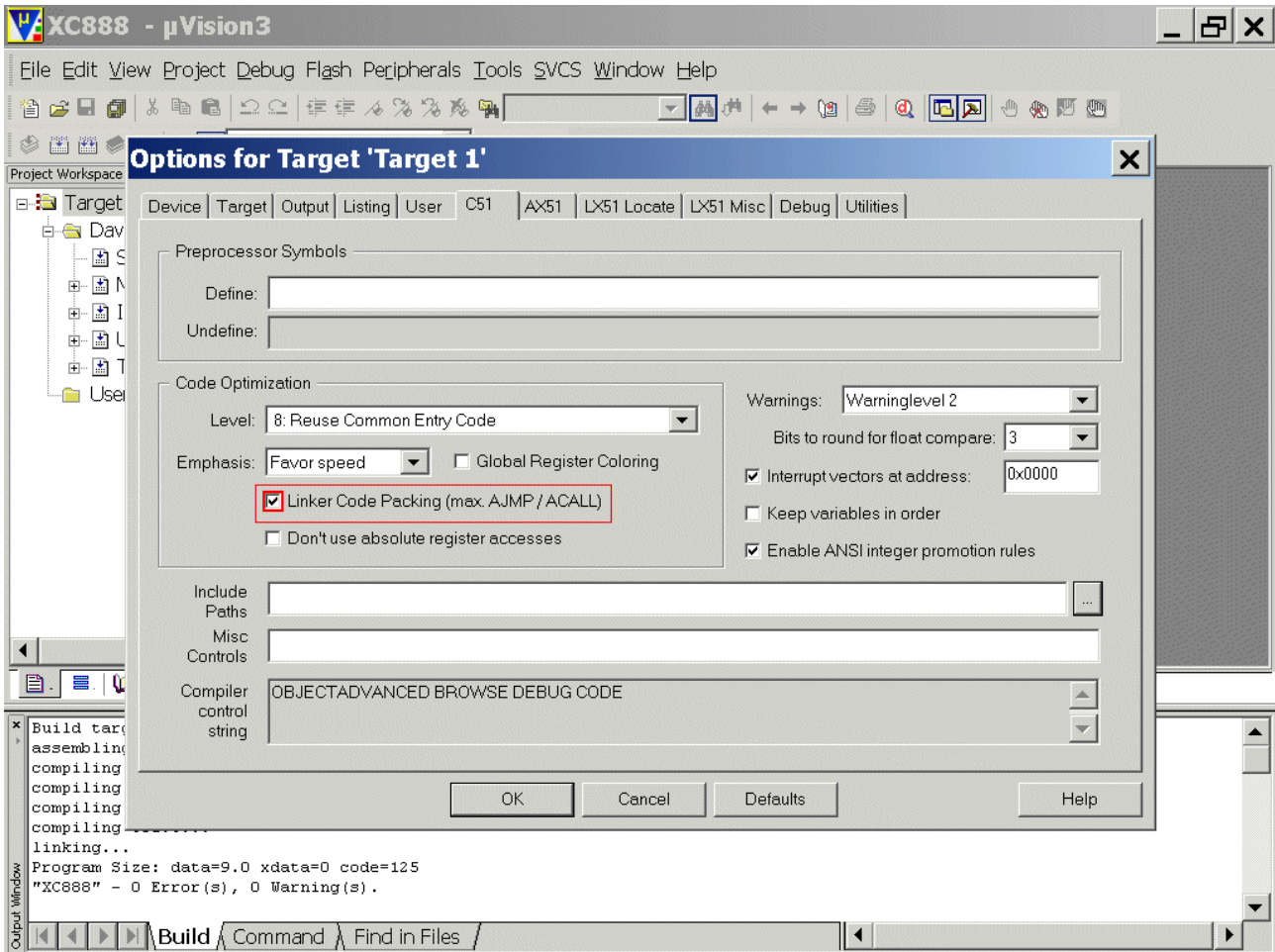
- 1.) position the mouse on the source code you are interested in
- 2.) [click](#) right mouse button and [select](#) Open Linker COD File
- 3.) see the result: Assembler-Code of your C-Source-Code



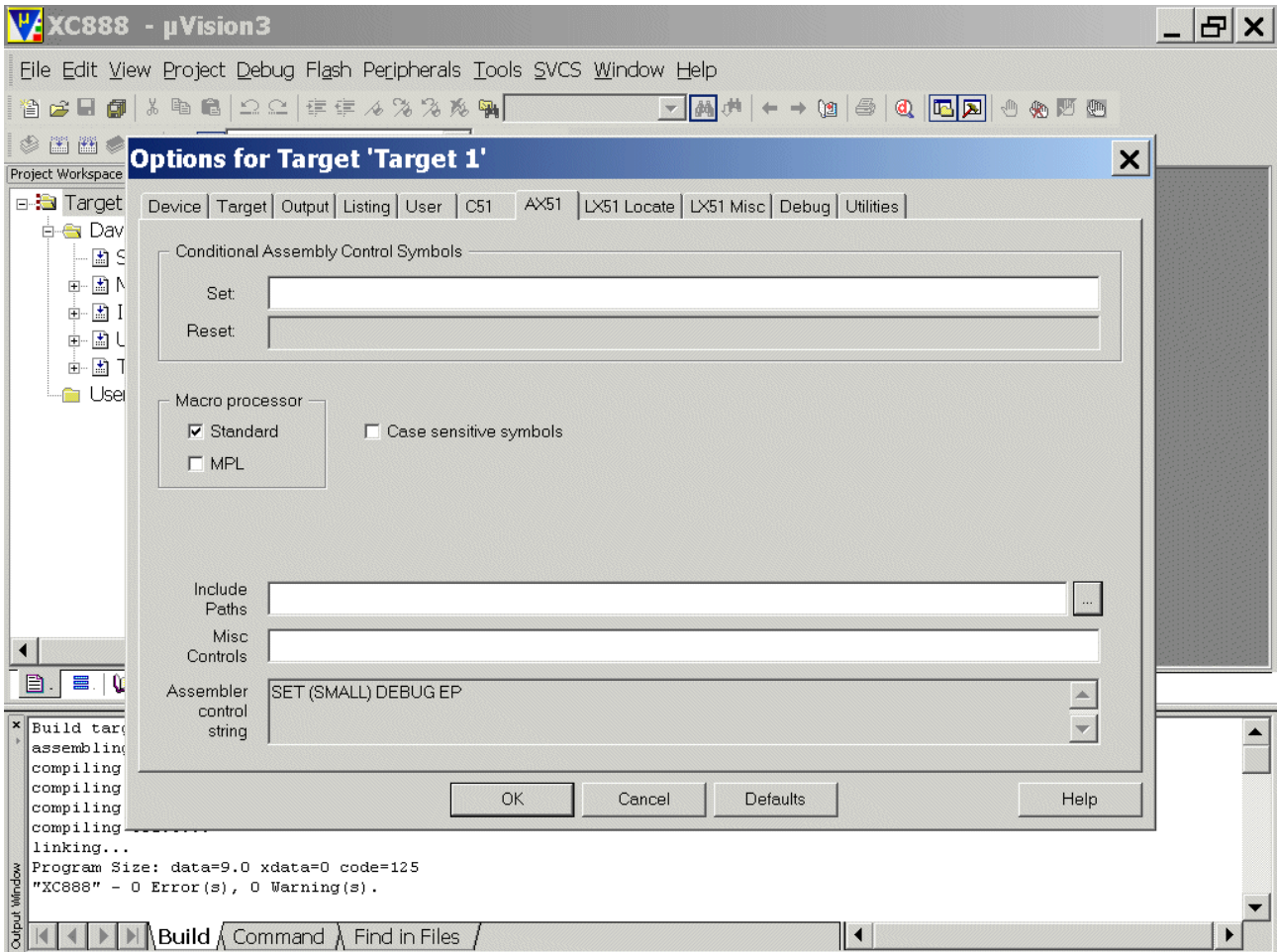
User: (do nothing)



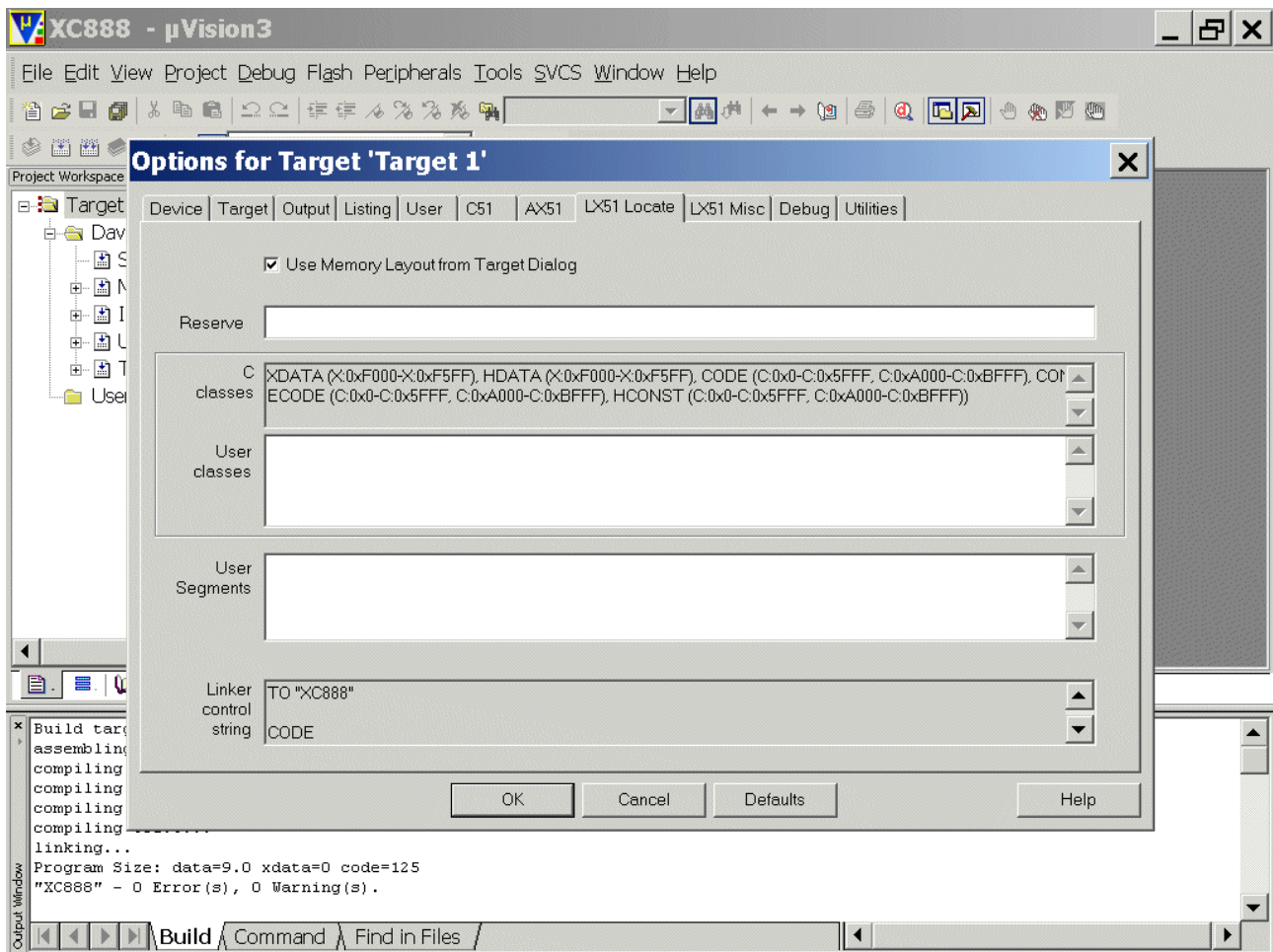
C51: Code Optimization: **click** ✓ Linker Code Packing (max. AJMP/ACALL)



AX51: (do nothing)

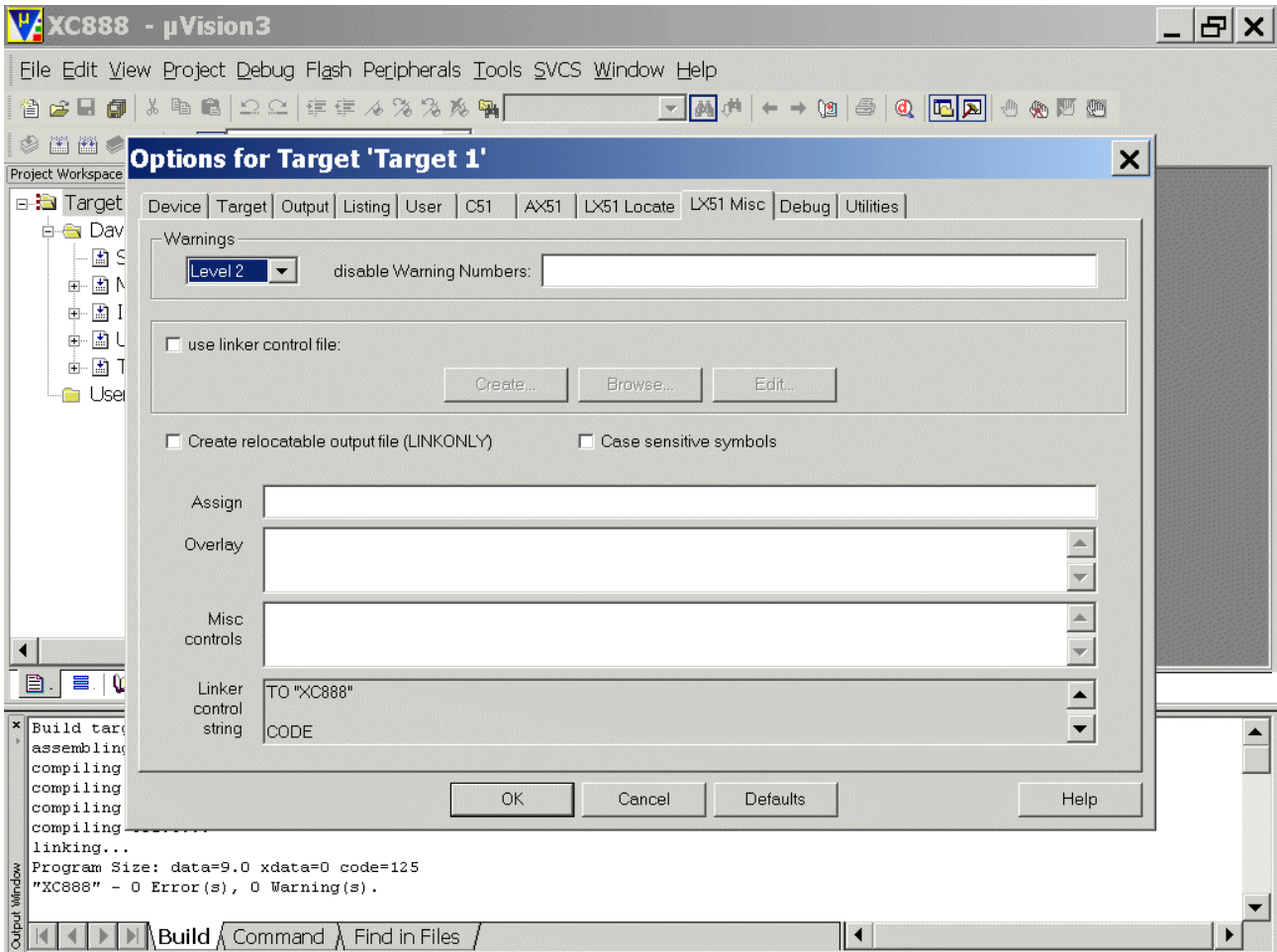


LX51 Locate: (do nothing)

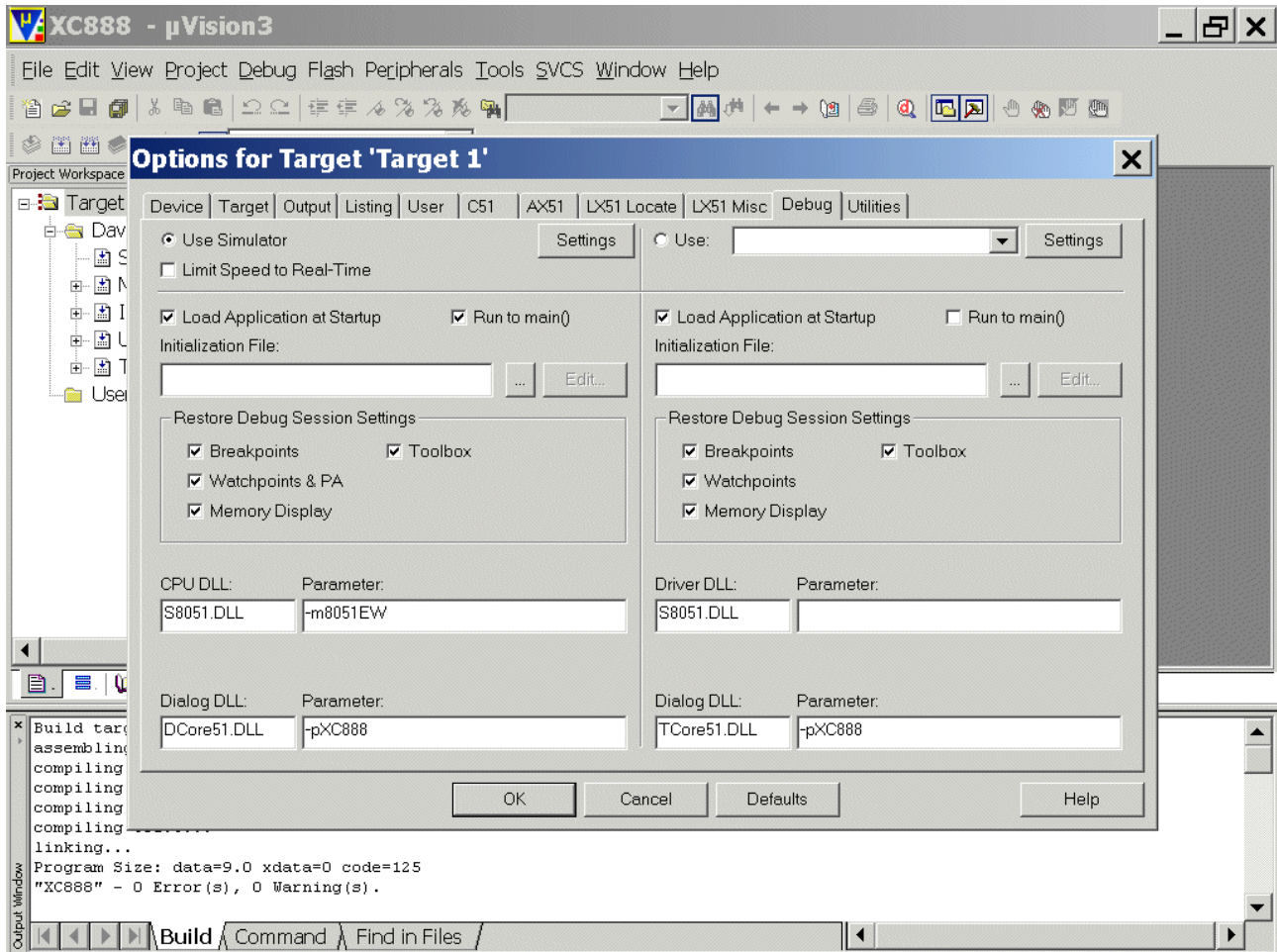




LX51 Misc: (do nothing)



Debug: (do nothing)

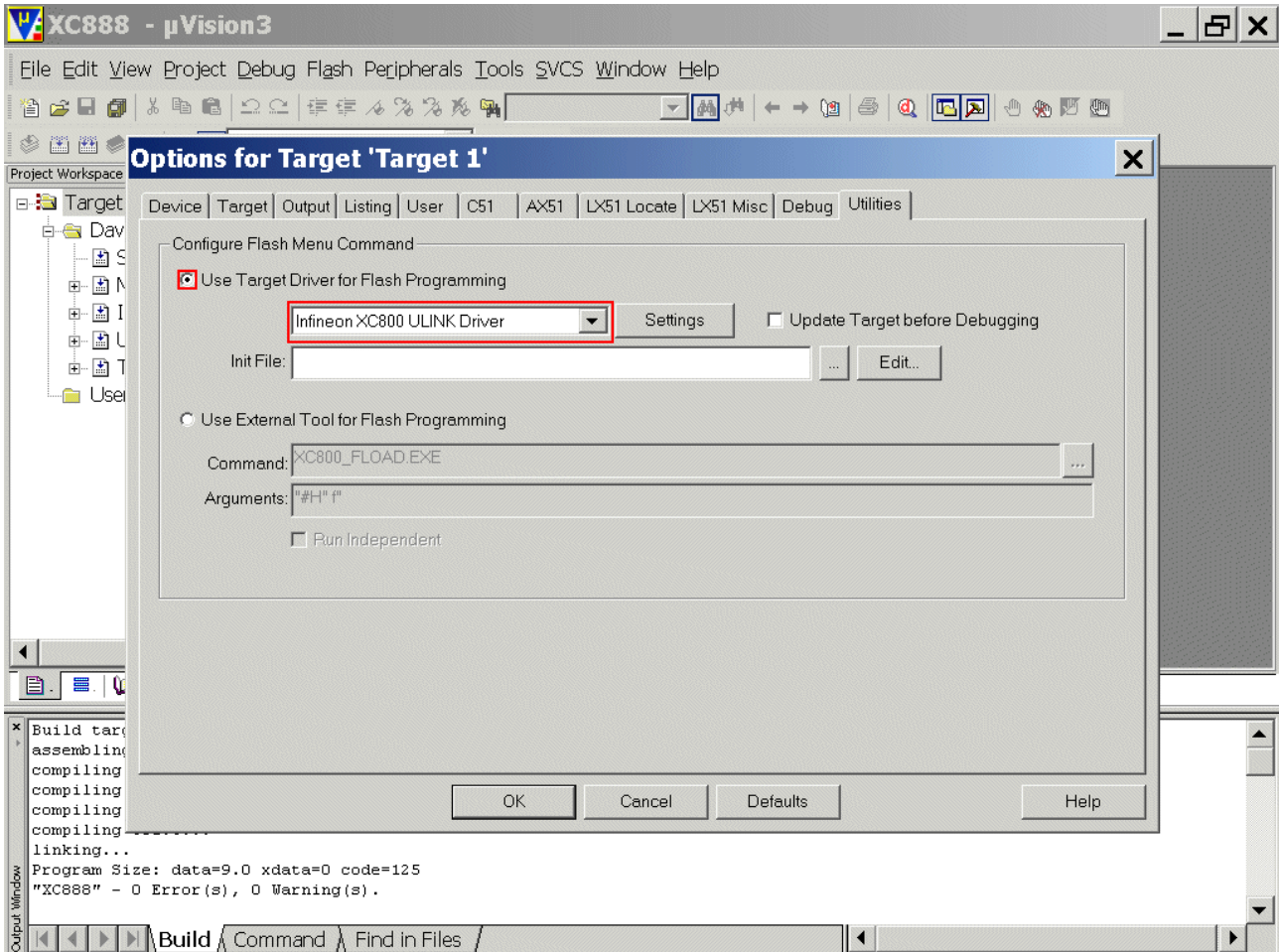


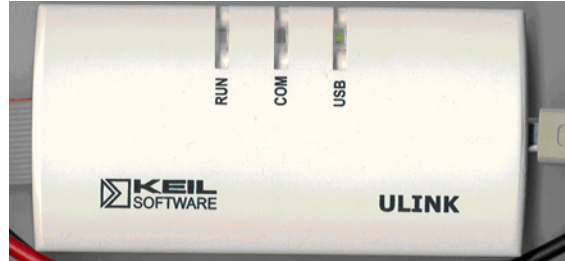
**Note:**

First we are going to use the simulator.



Utilities: **Configure Flash Menu Command**: click  Use Target Driver for Flash Programming,  
**Select**: Infineon XC800 ULINK Driver

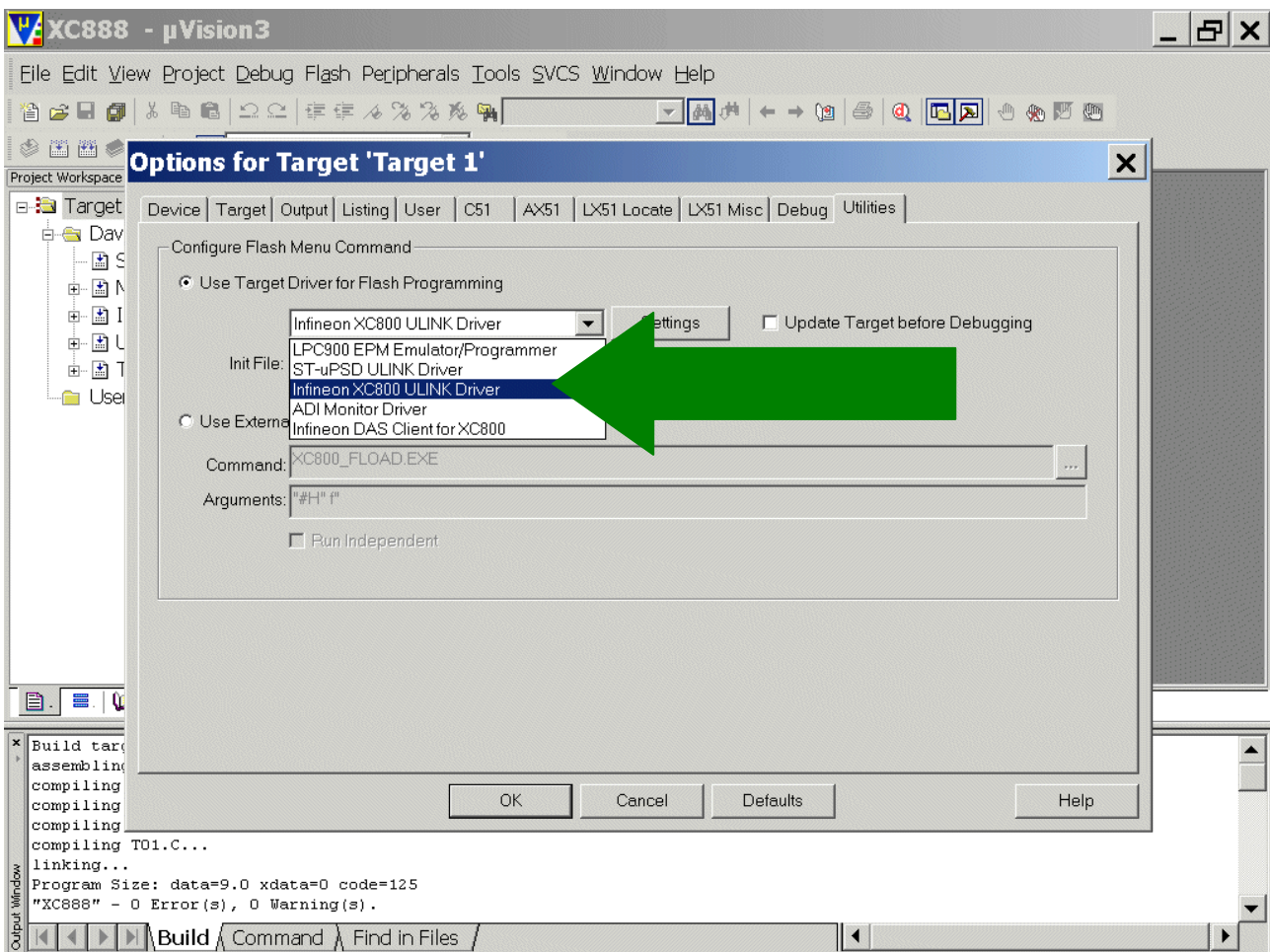


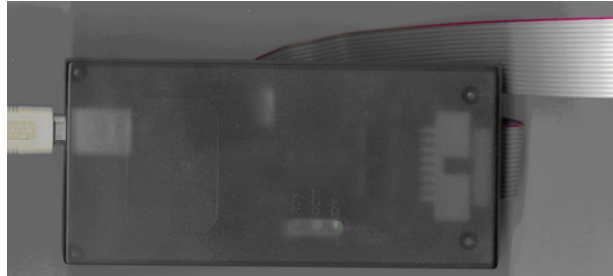


**Note:**

For OnChipFlashProgramming and OCDS-Debugging we are going to use the **ULINK** (“Keil-ULINK-JTAG-Interface”).

Therefore we selected the **Infineon XC800 ULINK Driver**:

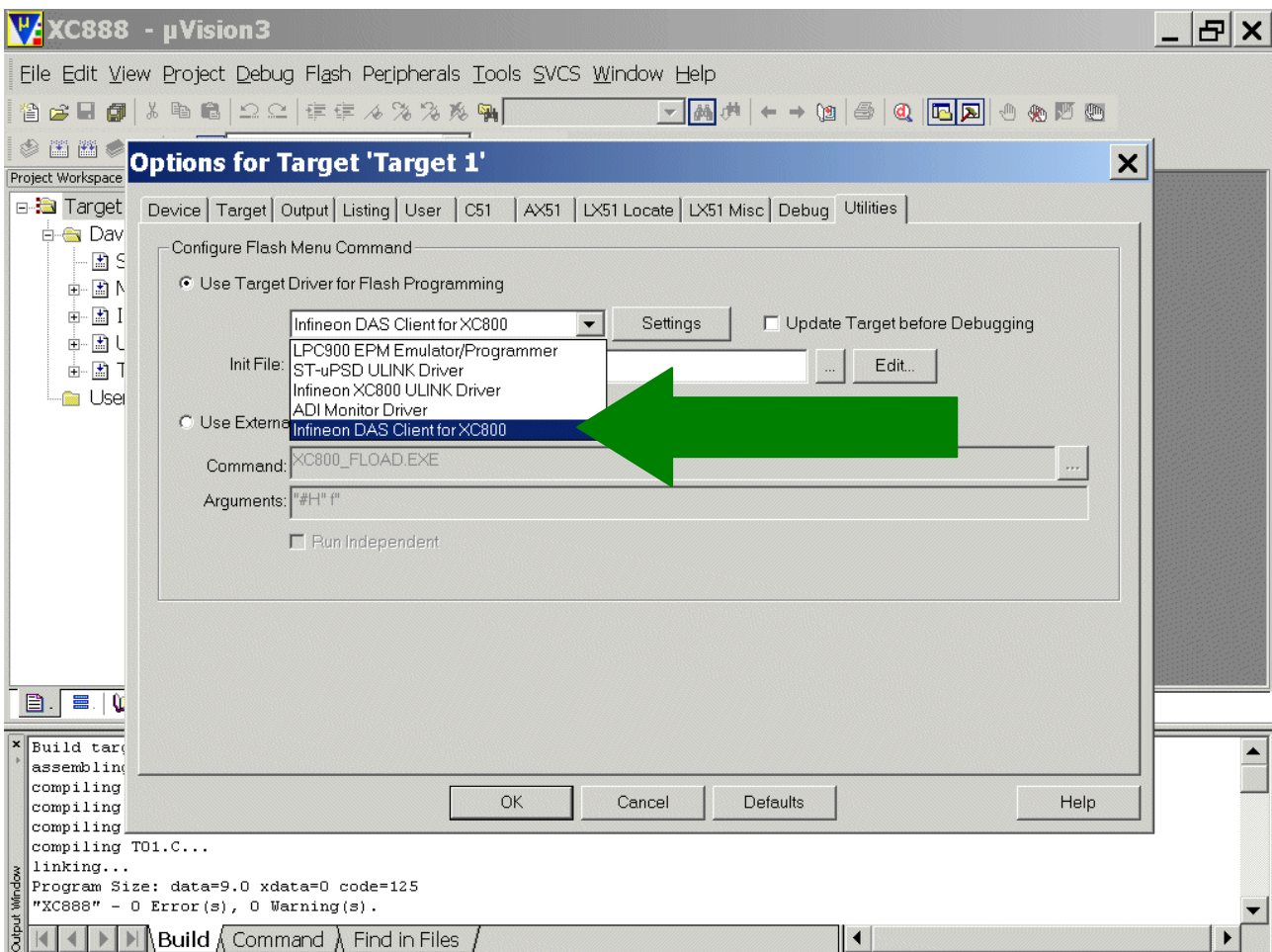




**Note:**

For OnChipFlashProgramming and OCDS-Debugging the Infineon-USB-JTAG-Wiggler-Box could be used instead of the Keil-ULINK-JTAG-Interface.

For the Infineon USB-JTAG-Wiggler-Box the Infineon DAS Client for XC800 must be selected:



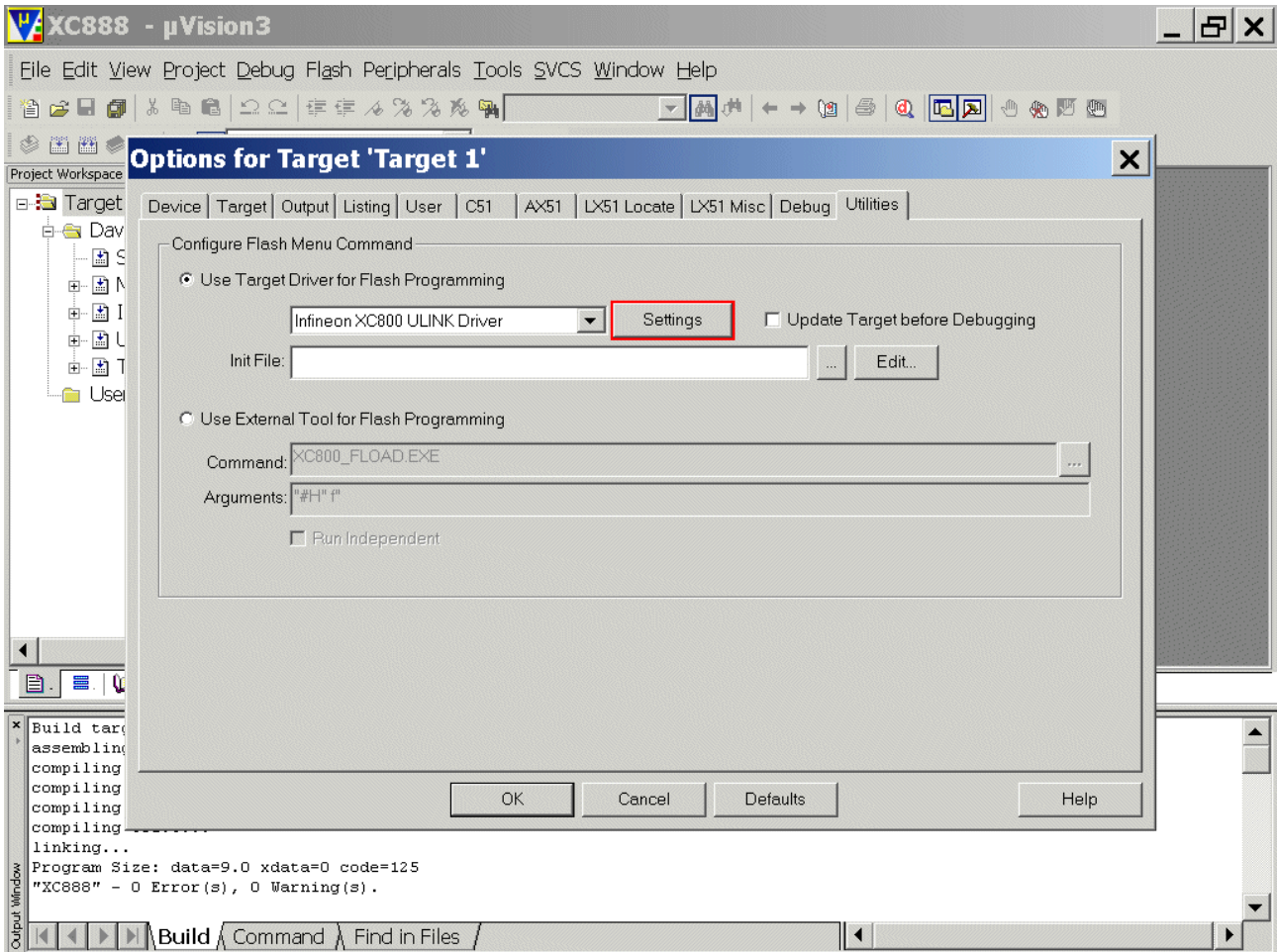
**Additionally, the DAS Server must be installed on your computer!**

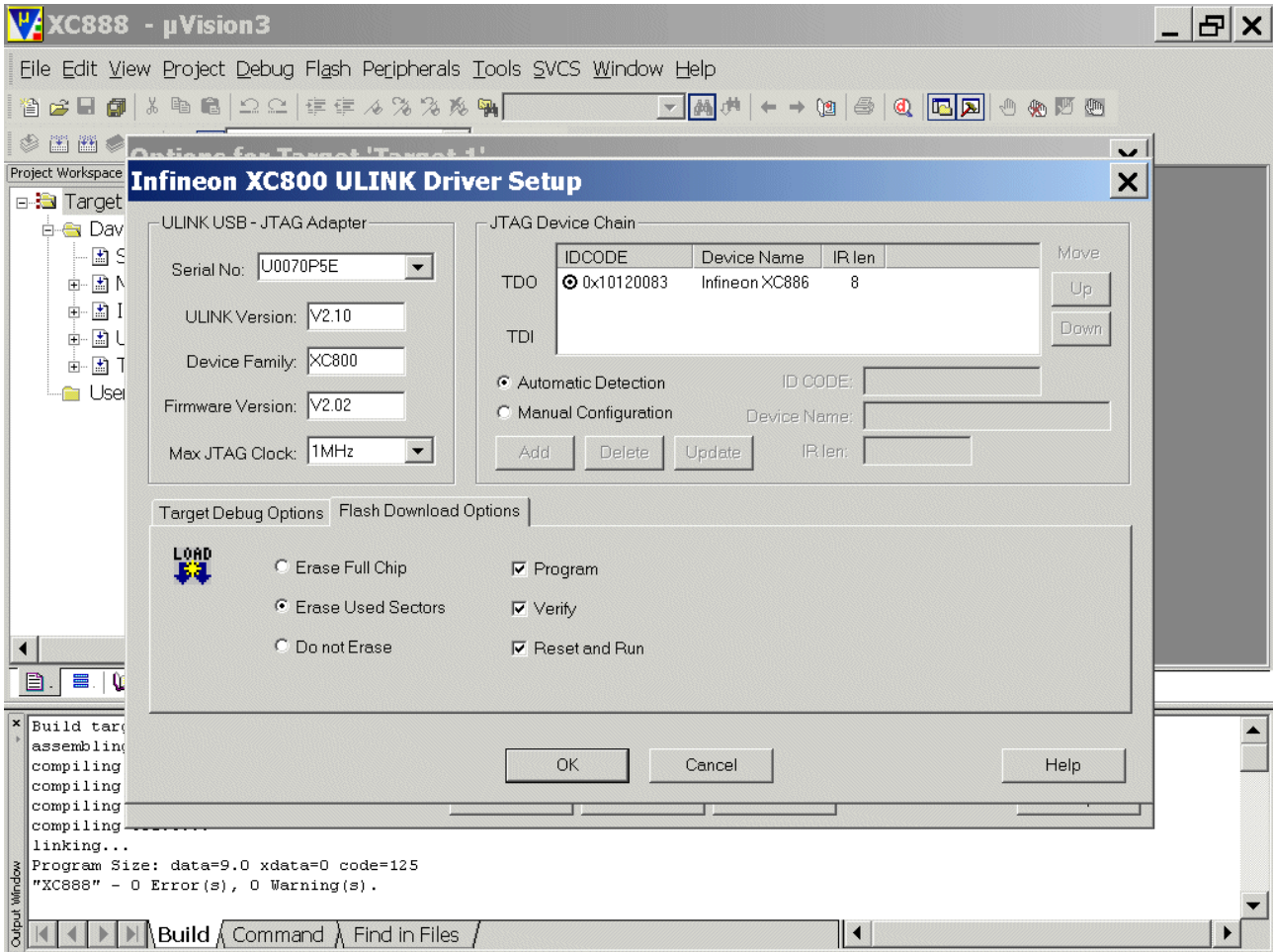
For further information, please see your Starter Kit CD and/or [www.infineon.com/DAS](http://www.infineon.com/DAS).



For OnChipFlashProgramming via UART the program [XC800\\_FLOAD.EXE](#) could be used instead of the [Keil-ULINK-JTAG-Interface](#) or the [Infineon-USB-JTAG-Wiggler-Box](#).  
For more information, please see your Starter Kit CD.

Utilities: [Configure Flash Menu Command](#): [click Settings](#)

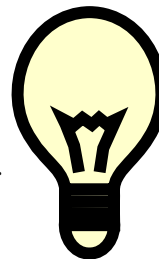




OK  
OK

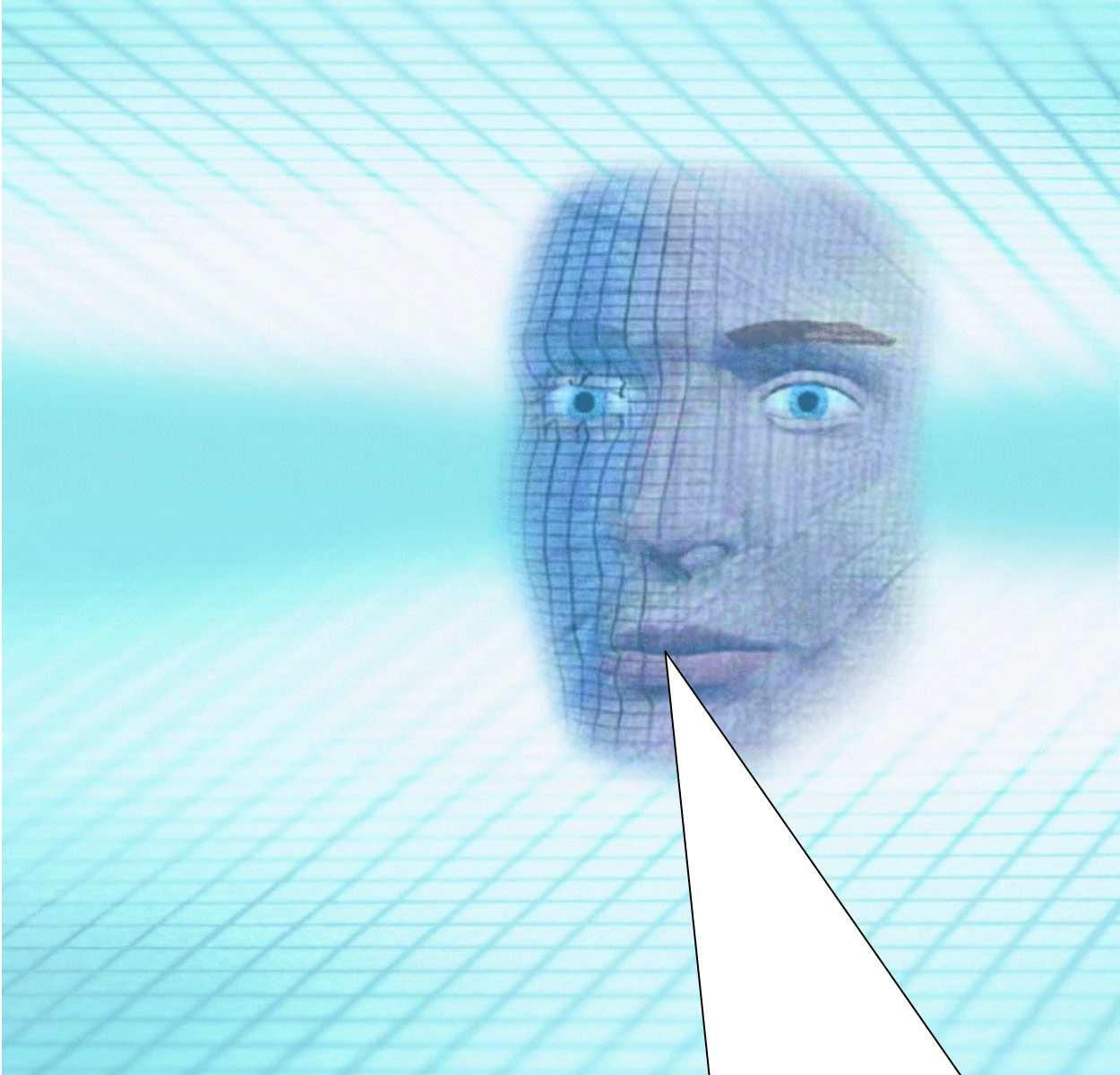
**Note:**

To make the ULINK work properly, the XC888 Evaluation Board must be supplied with power. We are going to do this in chapter 6.





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:

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

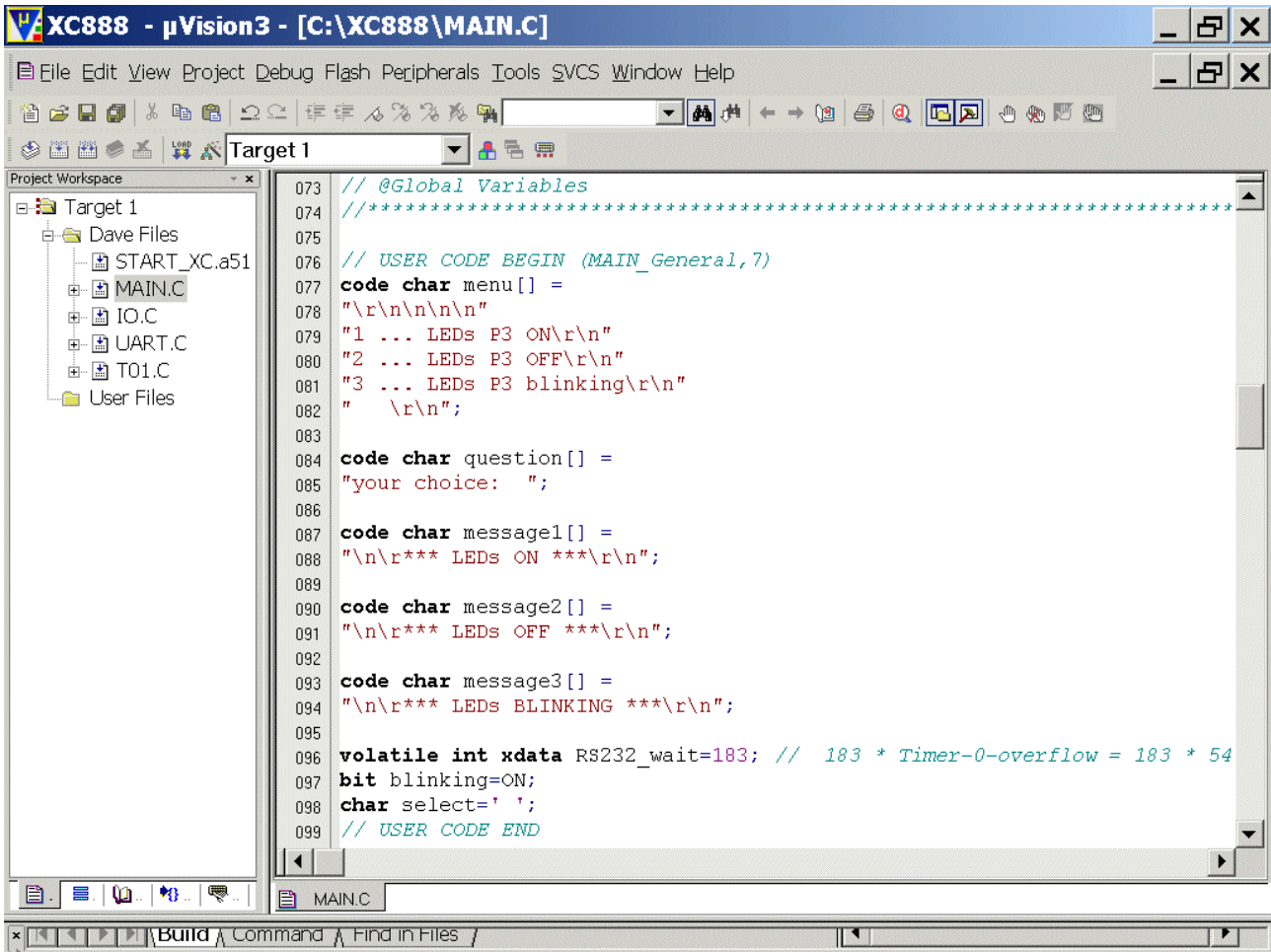
code char question[] =
"your choice: ";

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

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

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

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



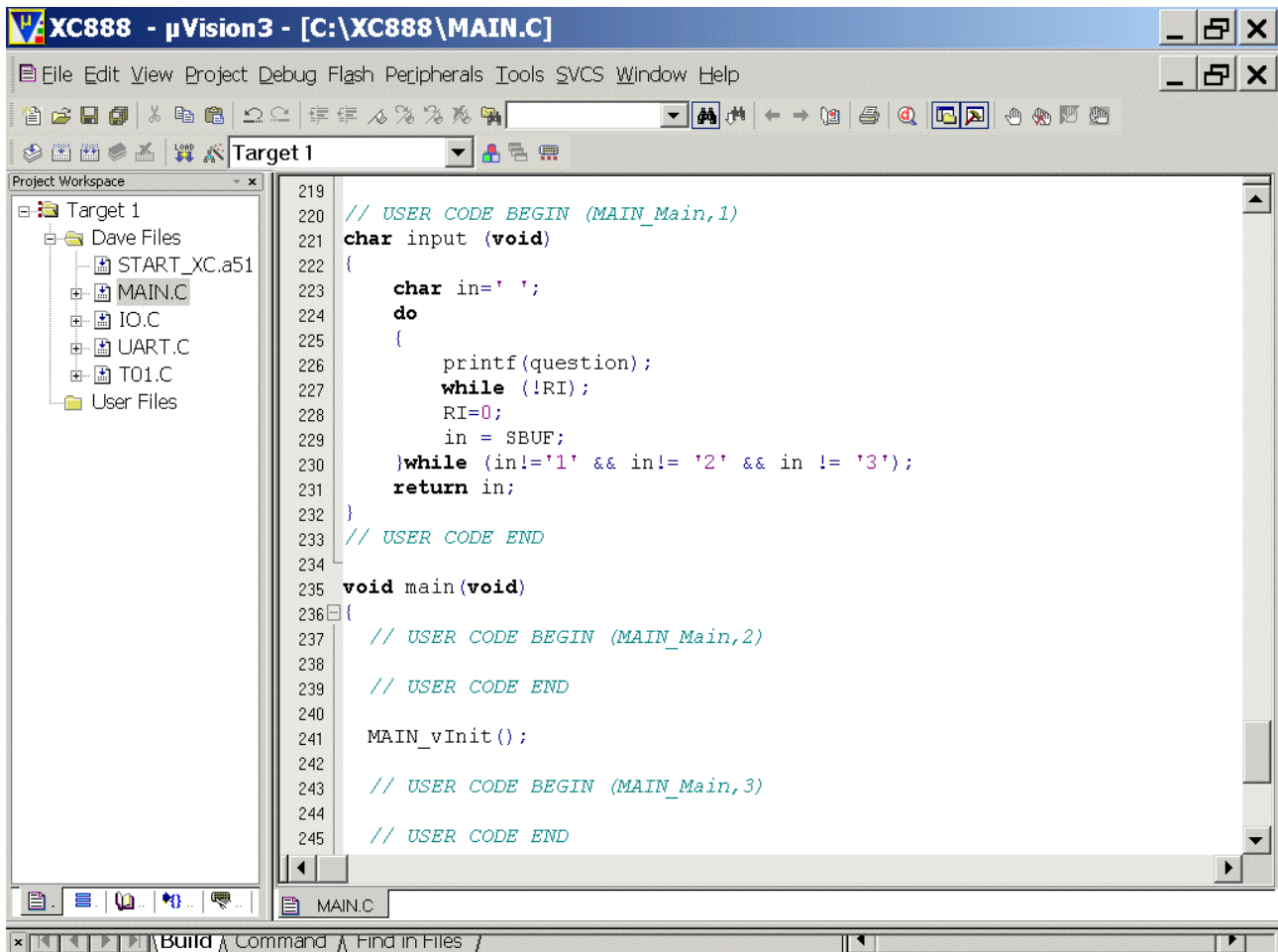
```

073 // @Global Variables
074 //*****
075
076 // USER CODE BEGIN (MAIN_General,7)
077 code char menu[] =
078 "\r\n\r\n\r\n"
079 "1 ... LEDs P3 ON\r\n"
080 "2 ... LEDs P3 OFF\r\n"
081 "3 ... LEDs P3 blinking\r\n"
082 " \r\n";
083
084 code char question[] =
085 "your choice: ";
086
087 code char message1[] =
088 "\n\r*** LEDs ON ***\r\n";
089
090 code char message2[] =
091 "\n\r*** LEDs OFF ***\r\n";
092
093 code char message3[] =
094 "\n\r*** LEDs BLINKING ***\r\n";
095
096 volatile int xdata RS232_wait=183; // 183 * Timer-0-overflow = 183 * 54
097 bit blinking=ON;
098 char select=' ';
099 // USER CODE END

```

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

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



The screenshot shows the XC88x IDE interface. The title bar reads "XC888 - µVision3 - [C:\XC888\MAIN.C]". The menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations and development tools. The Project Workspace on the left shows a tree view with "Target 1" expanded, containing "Dave Files" (START\_XC.a51, MAIN.C, IO.C, UART.C, T01.C) and "User Files". The main editor window displays the following code:

```
219 // USER CODE BEGIN (MAIN_Main,1)
220 char input (void)
221 {
222     char in=' ';
223     do
224     {
225         printf(question);
226         while (!RI);
227         RI=0;
228         in = SBUF;
229     }while (in!='1' && in!= '2' && in != '3');
230     return in;
231 }
232 // USER CODE END
233
234 void main(void)
235 {
236     // USER CODE BEGIN (MAIN_Main,2)
237     // USER CODE END
238
239     MAIN_vInit();
240
241     // USER CODE BEGIN (MAIN_Main,3)
242     // USER CODE END
243
244
245
```

The status bar at the bottom shows "Build" and "Command" tabs.

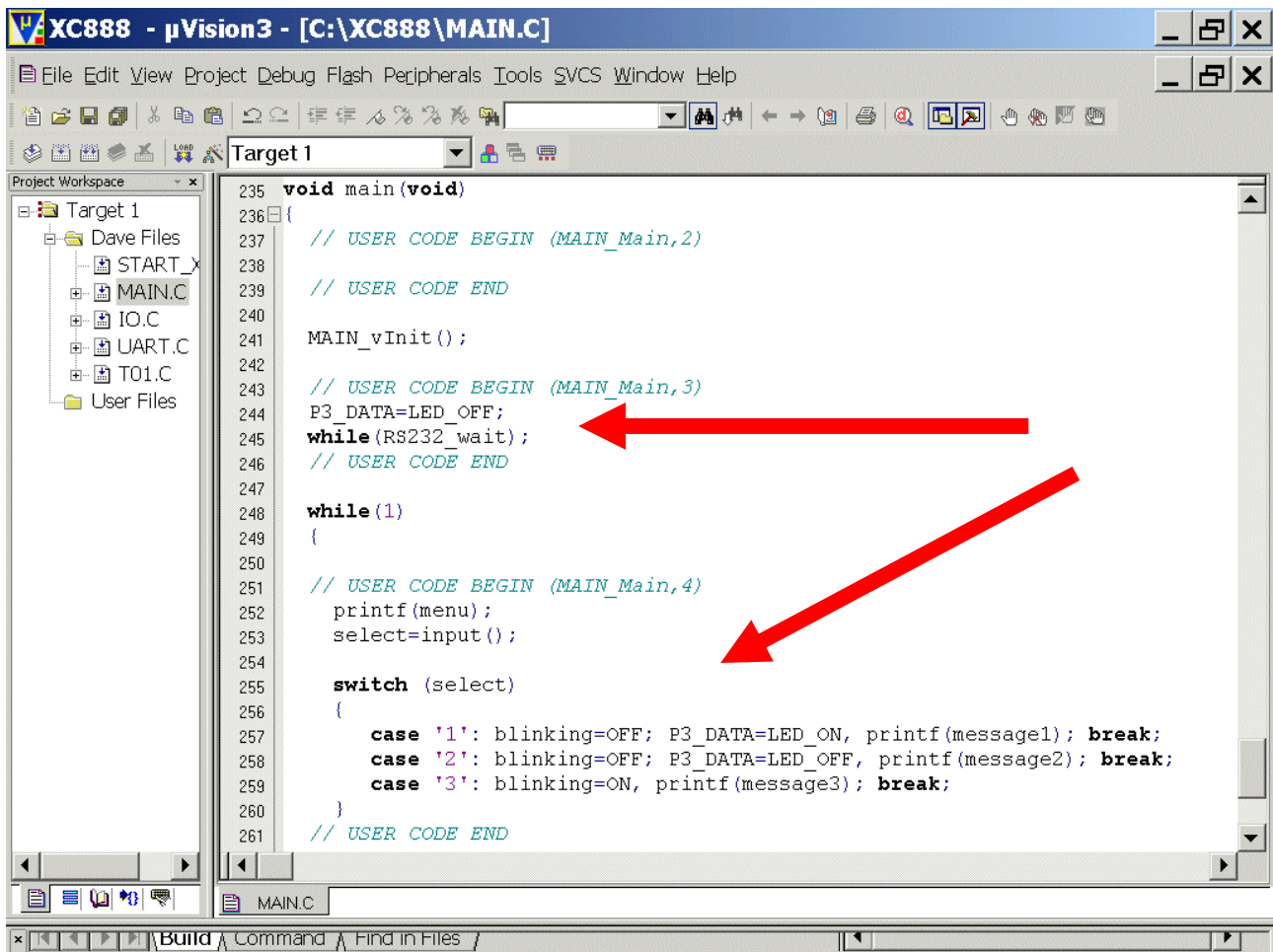
Double click **MAIN.C** and **insert** the following code in the **main** function:

```
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:

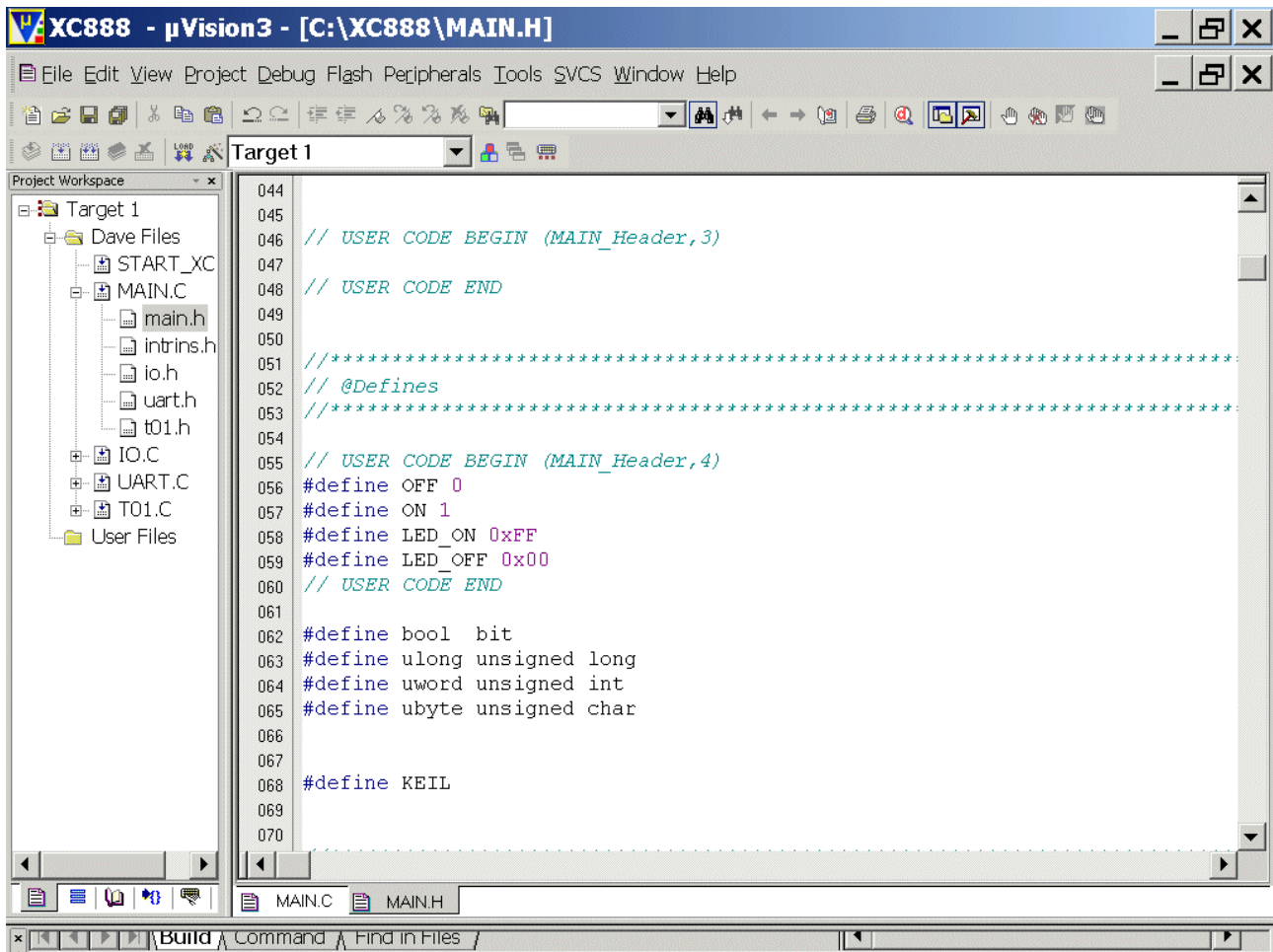
```
printf(menu);
select=input();

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

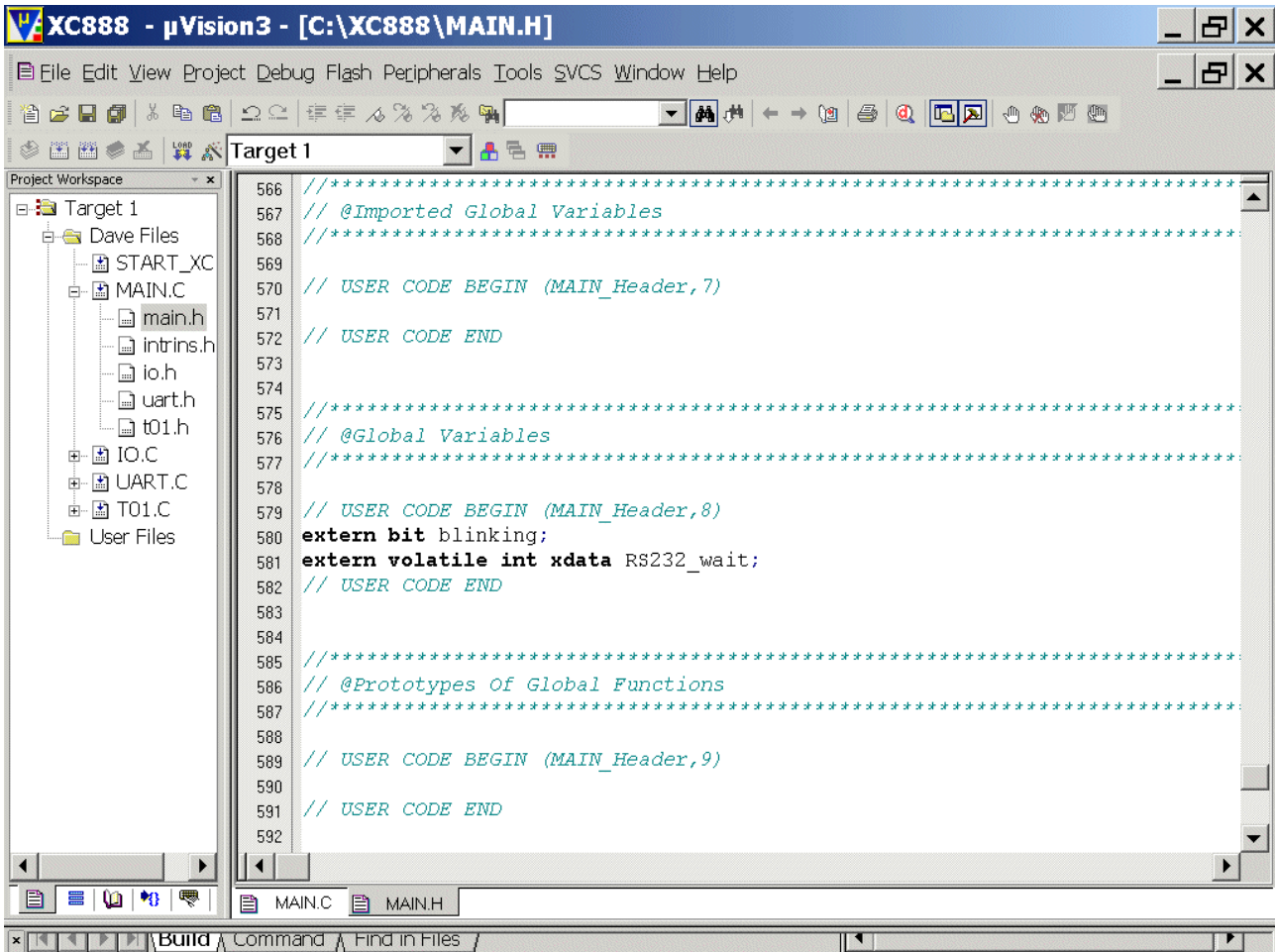


The screenshot shows the Keil uVision3 IDE interface. The title bar reads "XC888 - uVision3 - [C:\XC888\MAIN.H]". The menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The Project Workspace on the left shows a tree view with Target 1, Dave Files, START\_XC, MAIN.C, main.h, intrins.h, io.h, uart.h, t01.h, IO.C, UART.C, T01.C, and User Files. The main editor window displays the content of MAIN.H, showing the defined constants and other code. The status bar at the bottom indicates the current file is MAIN.H.

```
044
045 // USER CODE BEGIN (MAIN_Header,3)
046
047 // USER CODE END
048
049
050
051 //*****
052 // @Defines
053 //*****
054
055 // USER CODE BEGIN (MAIN_Header,4)
056 #define OFF 0
057 #define ON 1
058 #define LED_ON 0xFF
059 #define LED_OFF 0x00
060 // USER CODE END
061
062 #define bool bit
063 #define ulong unsigned long
064 #define uword unsigned int
065 #define ubyte unsigned char
066
067
068 #define KEIL
069
070
```

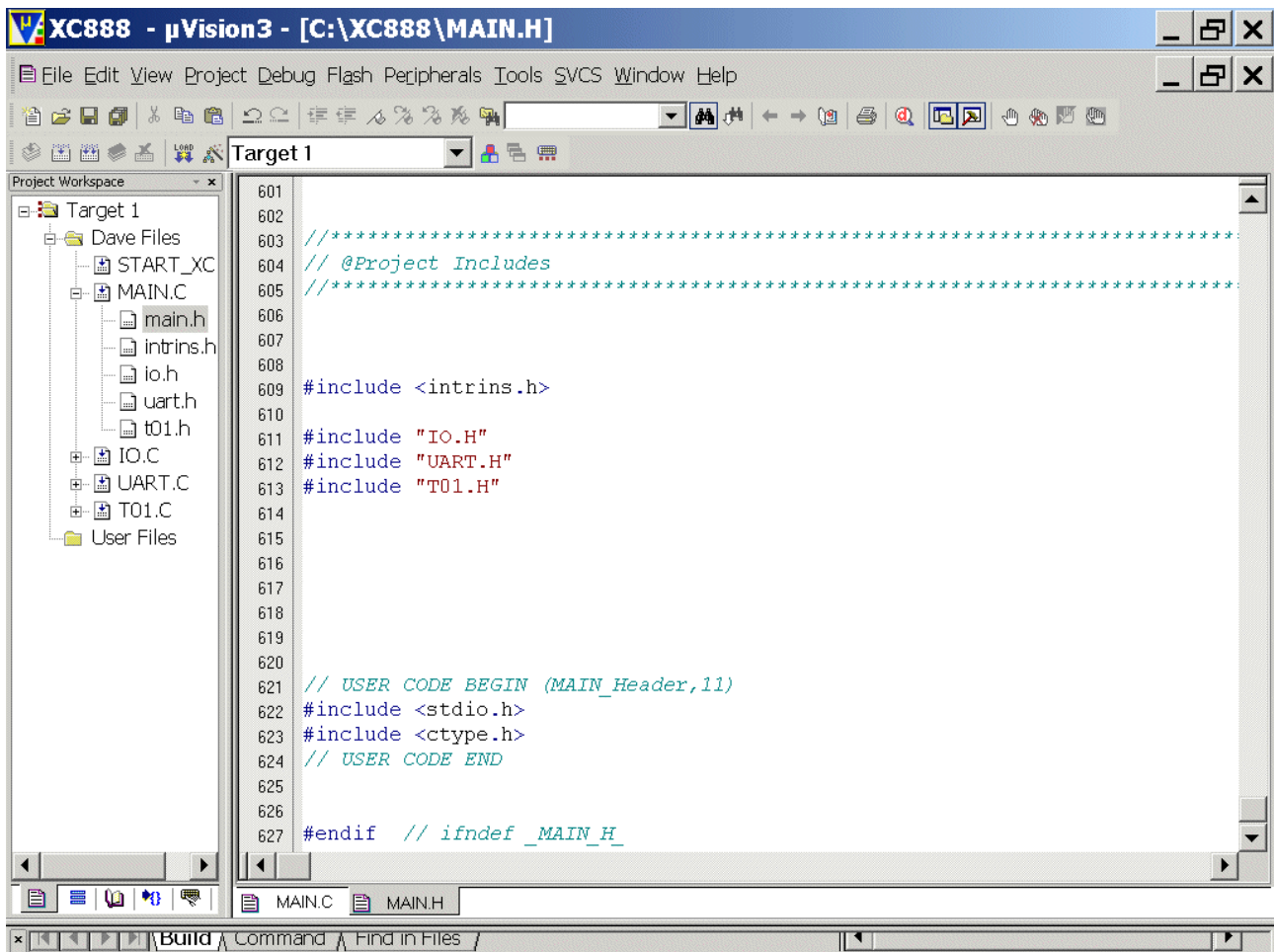
Double click **Main.h** and insert extern-declarations "Global Variables":

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



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

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

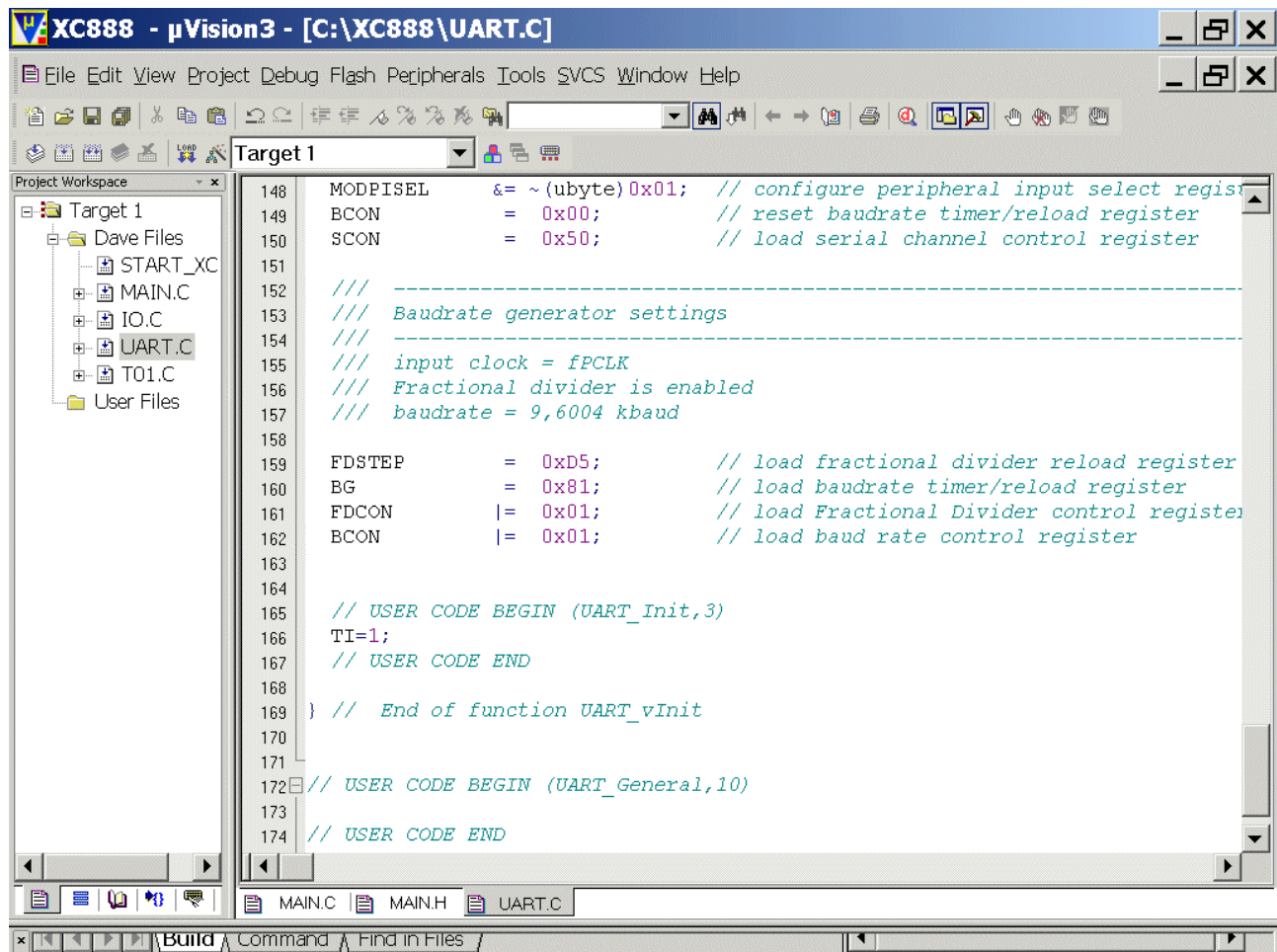




Double click UART.C

Insert code into the UART\_vInit function: (to start printf()):

```
TI=1;
```



The screenshot shows the XC88x IDE interface. The main window displays the source code for UART.C. The code includes initialization for the UART peripheral, setting baudrate generator settings, and configuring the fractional divider. The user code section contains the line `TI=1;` inserted into the `UART_vInit` function.

```

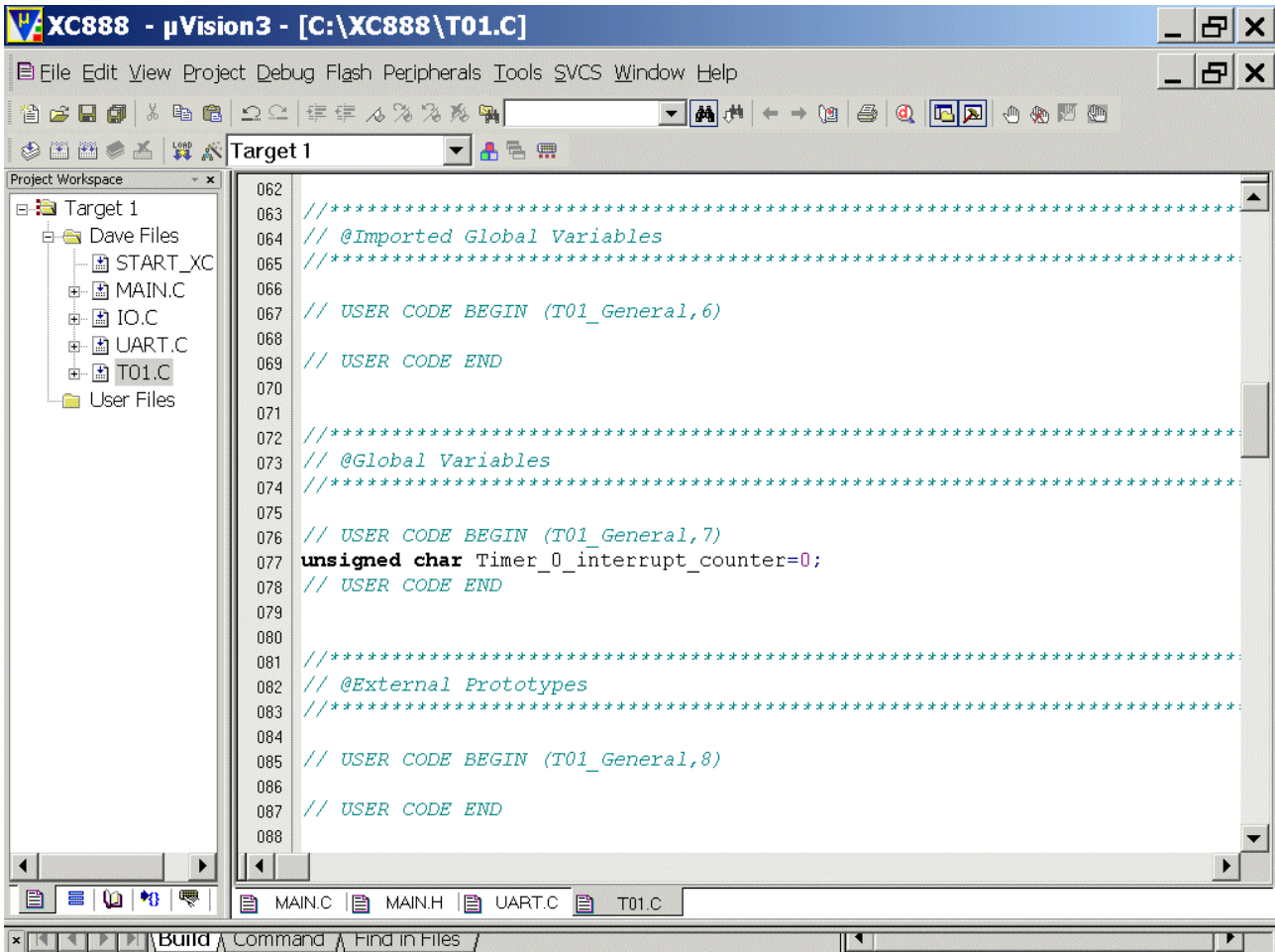
148  MODPSEL    &= ~(ubyte)0x01; // configure peripheral input select register
149  BCON       = 0x00;          // reset baudrate timer/reload register
150  SCON       = 0x50;          // load serial channel control register
151
152  //-----
153  // Baudrate generator settings
154  //-----
155  // input clock = fPCLK
156  // Fractional divider is enabled
157  // baudrate = 9,6004 kbaud
158
159  FDSTEP     = 0xD5;          // load fractional divider reload register
160  BG         = 0x81;          // load baudrate timer/reload register
161  FDCON     |= 0x01;          // load Fractional Divider control register
162  BCON       |= 0x01;          // load baud rate control register
163
164
165  // USER CODE BEGIN (UART_Init,3)
166  TI=1;
167  // USER CODE END
168
169  } // End of function UART_vInit
170
171
172  // USER CODE BEGIN (UART_General,10)
173
174  // USER CODE END

```

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:

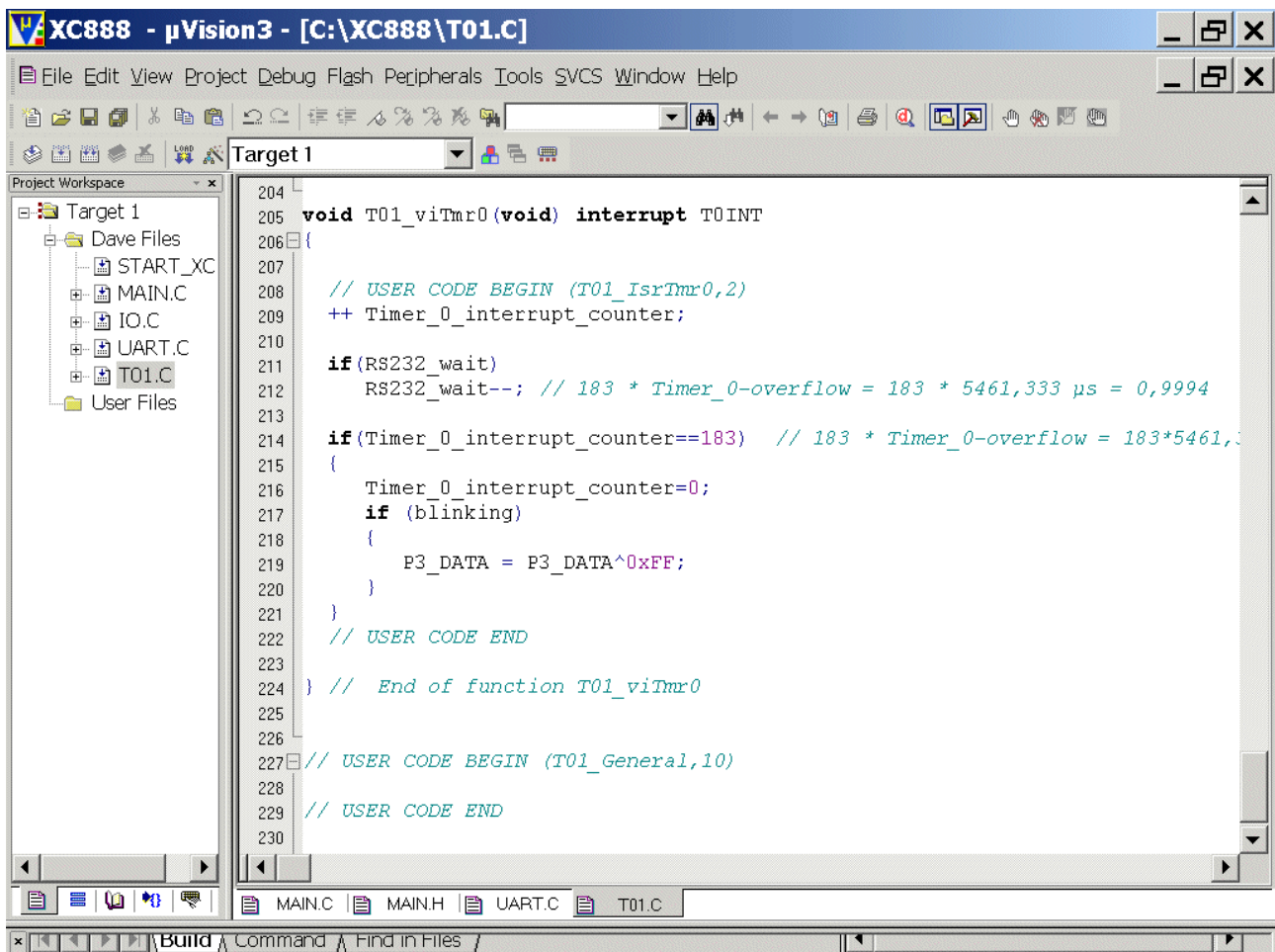
```

++ 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;
    }
}

```

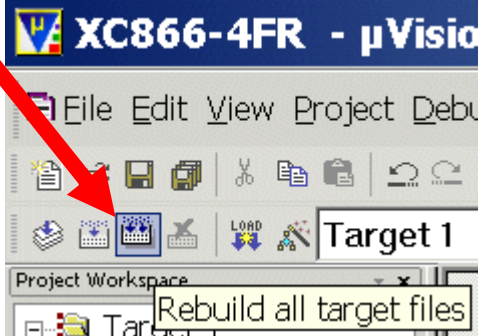


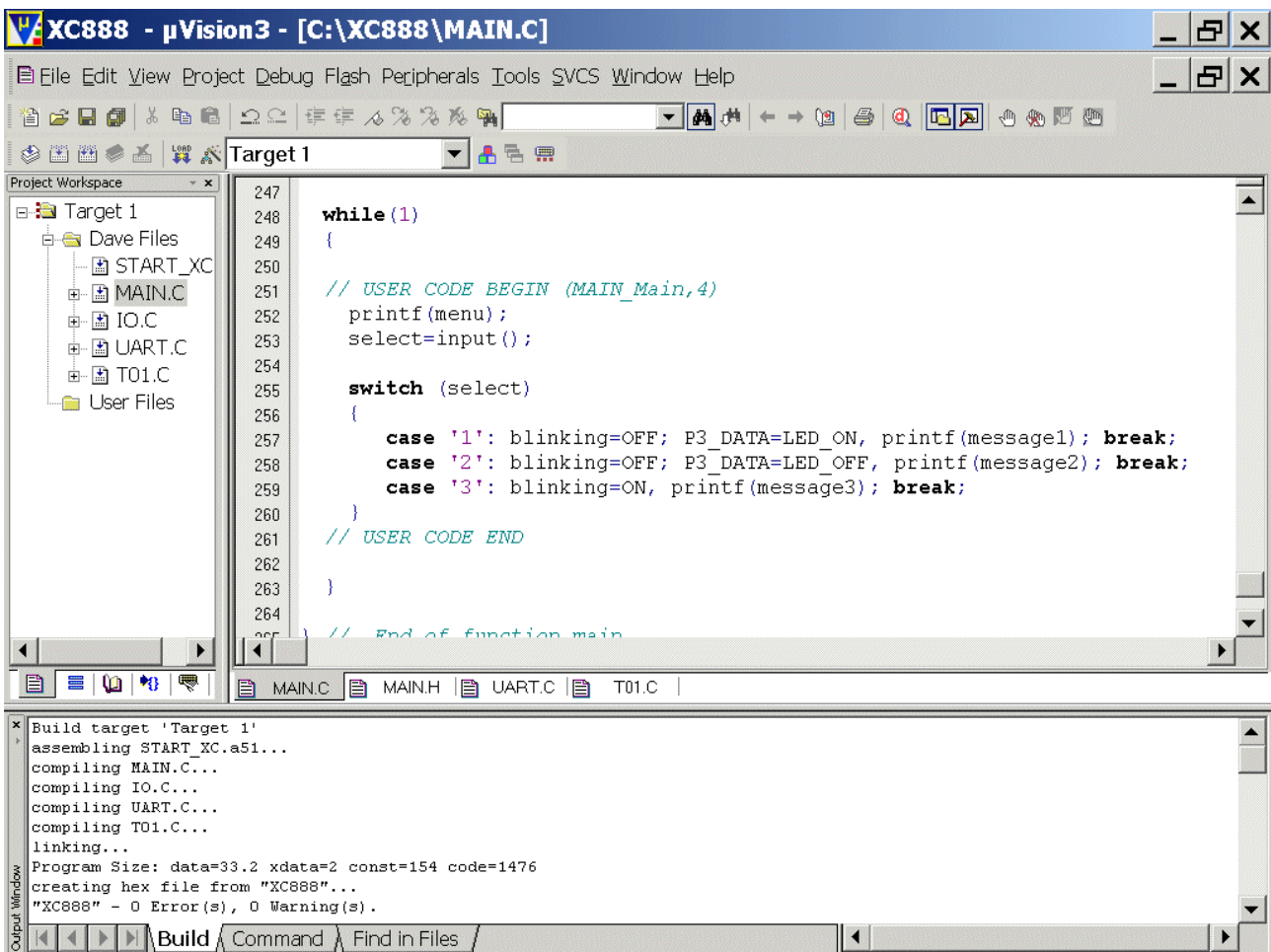
```

XC888 - μVision3 - [C:\XC888\T01.C]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
  Dave Files
  START_XC
  MAIN.C
  IO.C
  UART.C
  T01.C
  User Files
204
205 void T01_viTmr0(void) interrupt T0INT
206 {
207     // USER CODE BEGIN (T01_IsrTmr0,2)
208     ++ Timer_0_interrupt_counter;
209
210     if(RS232_wait)
211         RS232_wait--; // 183 * Timer_0-overflow = 183 * 5461,333 μs = 0,9994
212
213     if(Timer_0_interrupt_counter==183) // 183 * Timer_0-overflow = 183*5461,333μs = 0,9994s
214     {
215         Timer_0_interrupt_counter=0;
216         if (blinking)
217         {
218             P3_DATA = P3_DATA^0xFF;
219         }
220     }
221     // USER CODE END
222 } // End of function T01_viTmr0
223
224 // USER CODE BEGIN (T01_General,10)
225
226 // USER CODE END
227
228
229
230
MAIN.C MAIN.H UART.C T01.C
Build Command Find in Files

```

Generate your application program:

<p>Project – Rebuild all target files</p>	<p>or</p>	<p>click</p> 
---	-----------	---



The screenshot shows the XC888 - μVision3 IDE interface. The title bar indicates the project is located at [C:\XC888\MAIN.C]. The menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations and development tasks. The Project Workspace pane on the left shows a tree view with Target 1, Dave Files, START\_XC, MAIN.C, IO.C, UART.C, T01.C, and User Files. The main editor window displays the following C code:

```

247
248 while (1)
249 {
250
251 // USER CODE BEGIN (MAIN_Main,4)
252 printf(menu);
253 select=input();
254
255 switch (select)
256 {
257 case '1': blinking=OFF; P3_DATA=LED_ON, printf(message1); break;
258 case '2': blinking=OFF; P3_DATA=LED_OFF, printf(message2); break;
259 case '3': blinking=ON, printf(message3); break;
260 }
261 // USER CODE END
262
263 }
264
265 // End of function main

```

The Output Window at the bottom shows the build process for Target 1:

```

Build target 'Target 1'
assembling START_XC.a51...
compiling MAIN.C...
compiling IO.C...
compiling UART.C...
compiling T01.C...
linking...
Program Size: data=33.2 xdata=2 const=154 code=1476
creating hex file from "XC888"...
"XC888" - 0 Error(s), 0 Warning(s).

```

The status bar at the bottom indicates the 'Build' command is active.

Now we close our project and  $\mu$ Vision 3:

Project  
Close Project

File

Exit

5.) Using the Simulator (first we will test our program with the Keil Simulator):



Start Keil  $\mu$ Vision and open our Keil Project

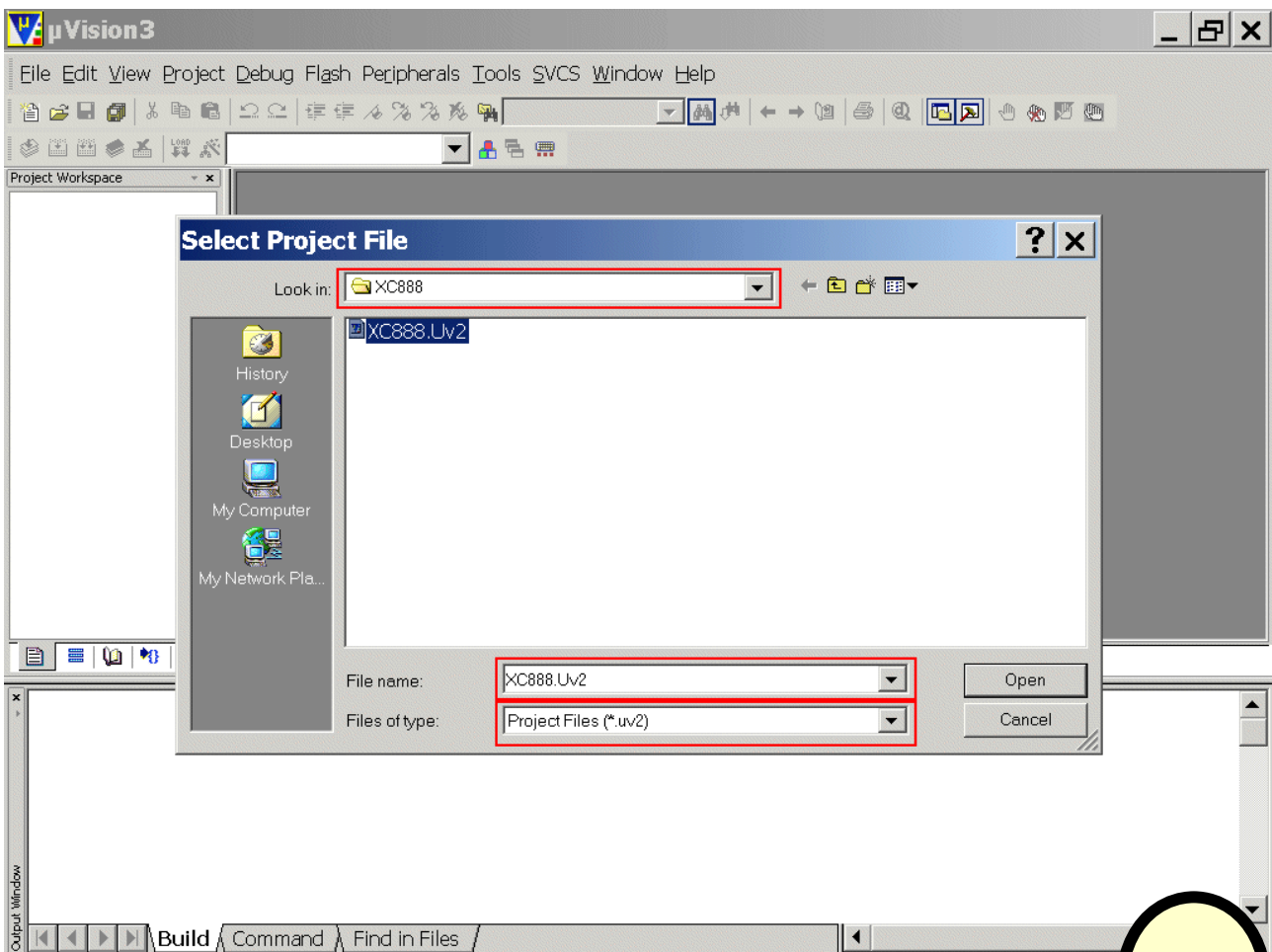
If you see an open project – close it: **Project - Close Project**

**Project - Open Project**

Select Project File: **Look in:** choose C:\XC888

Select Project File: **Files of type:** choose Project Files (\*.uv2)  
choose XC888.Uv2

**Open**



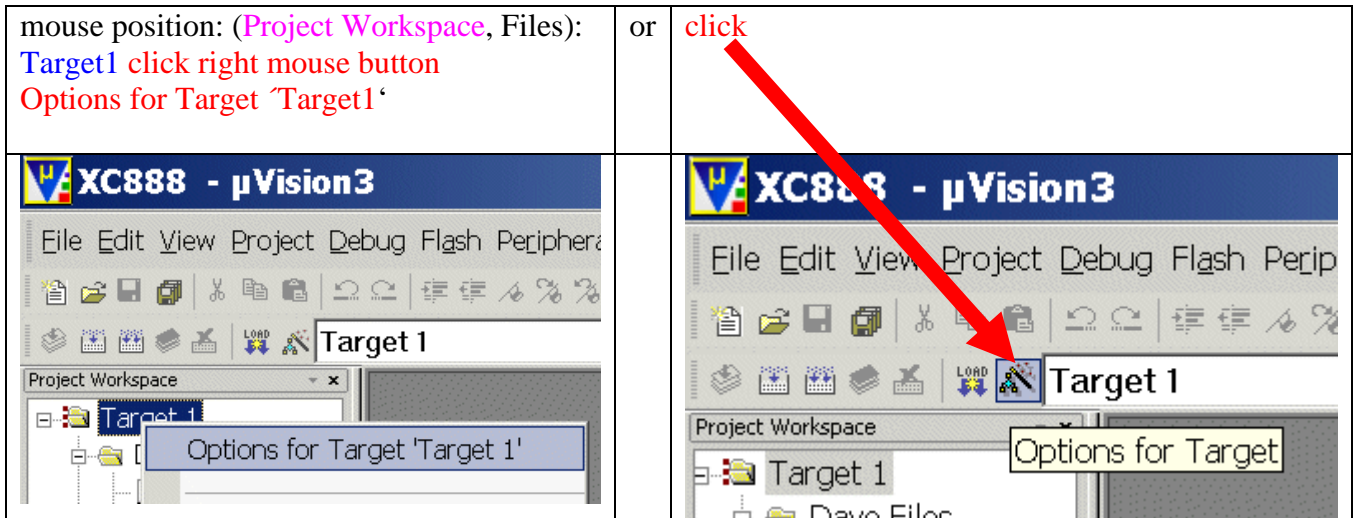
**Note:**

From now on just open your  $\mu$ Vision project (not the DAVE project).

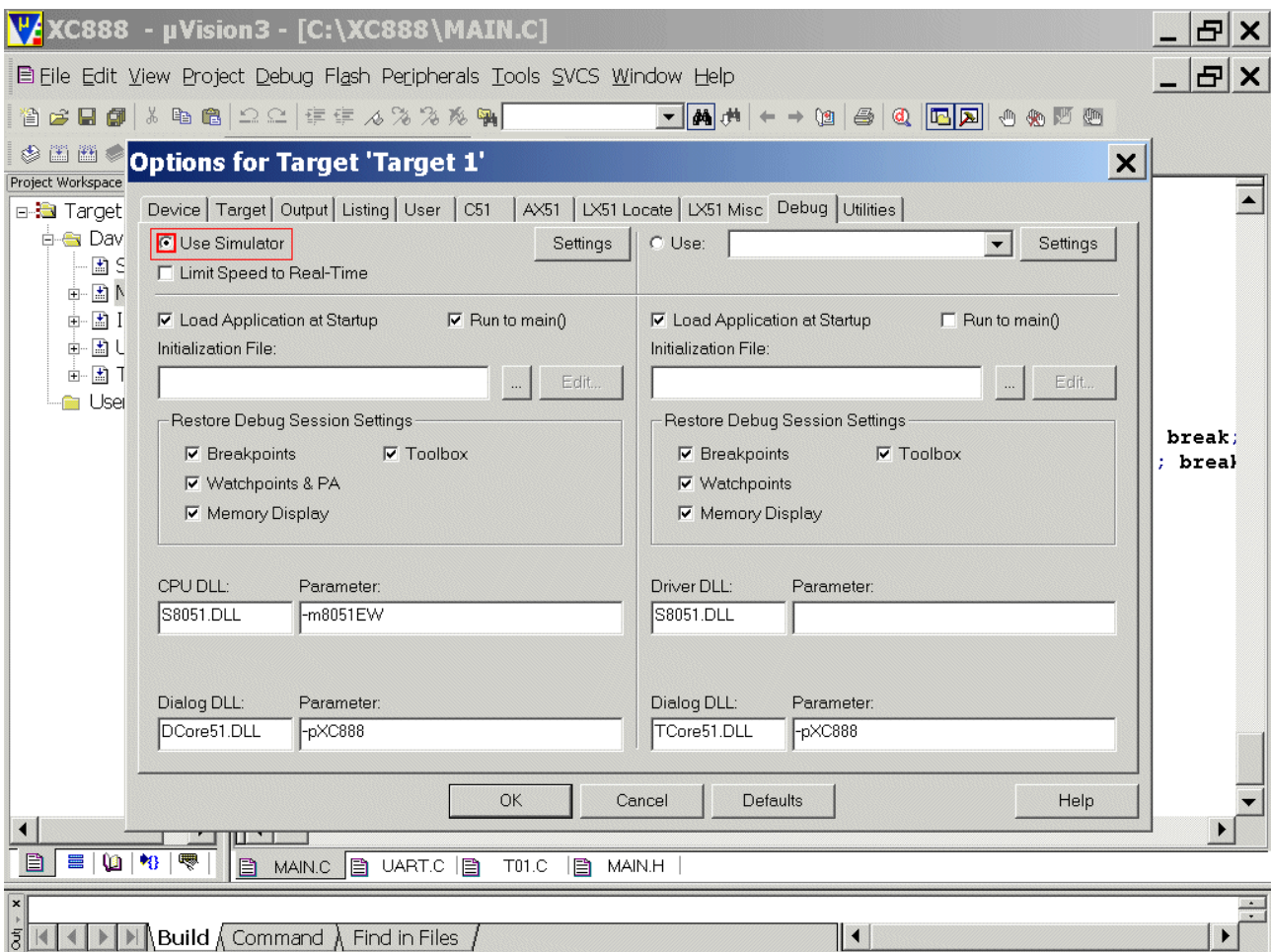
$\mu$ Vision will automatically recognise if there has been a code regeneration done by DAVE!



Check the configuration of the  $\mu$ Vision simulator:




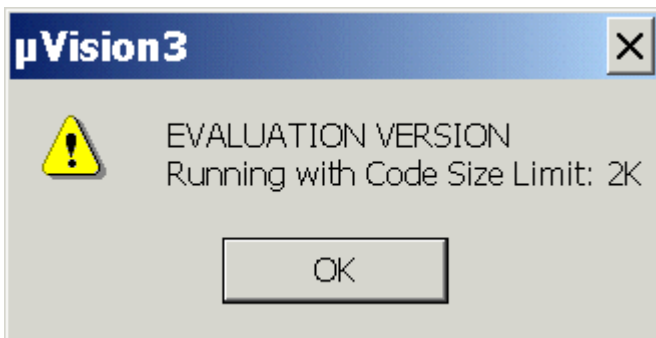
Options for Target 'Target1': Debug: check  Use Simulator



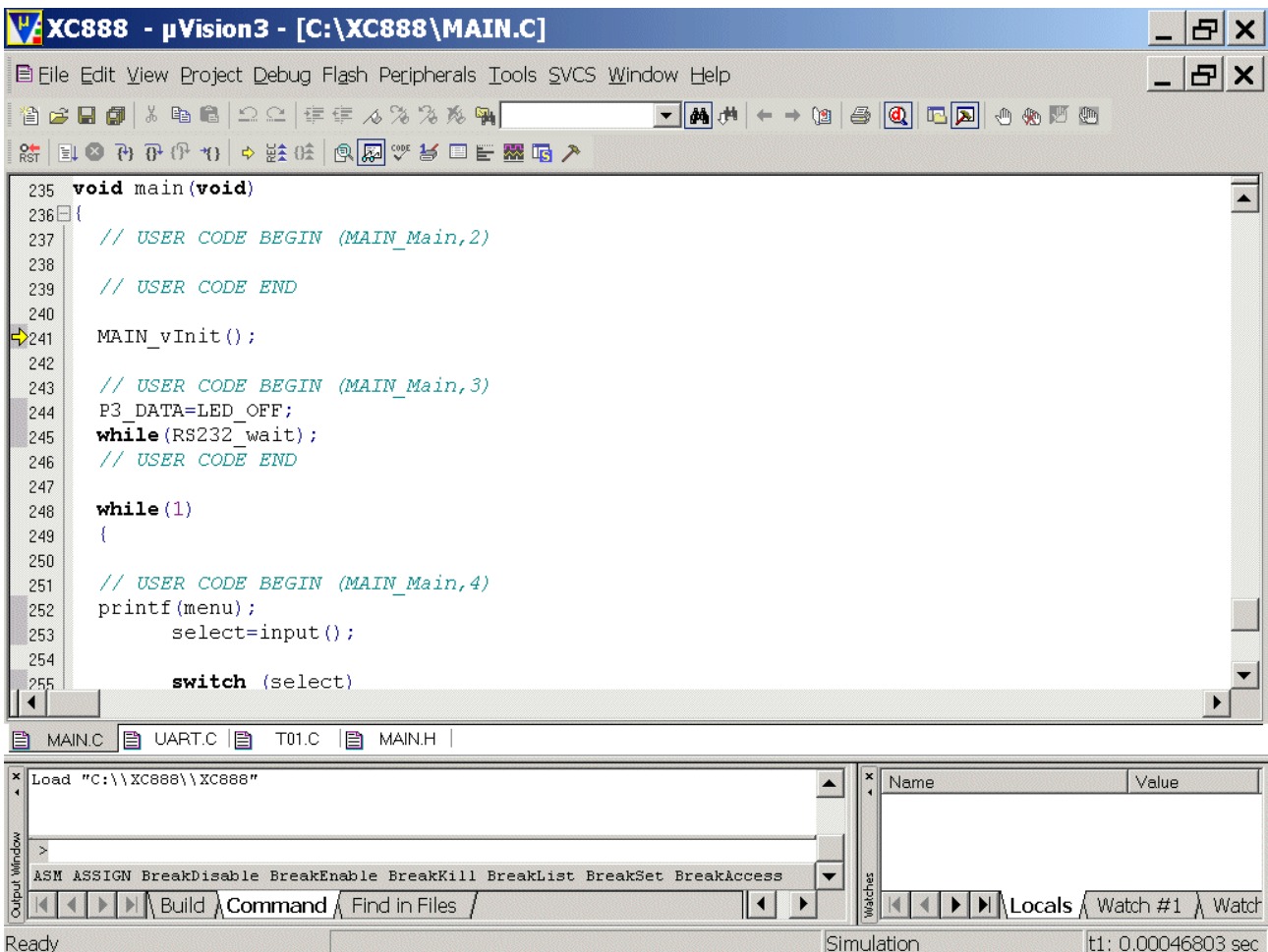
OK

Start the  $\mu$ Vision Simulator:

<p>Debug - Start/Stop Debug Session</p>	<p>or</p>	<p>click</p> 
---	-----------	---



OK



The screenshot shows the µVision3 IDE with the following components:

- Editor:** Displays a C program snippet:
 

```

235 void main(void)
236 {
237     // USER CODE BEGIN (MAIN_Main,2)
238
239     // USER CODE END
240
241     MAIN_vInit();
242
243     // USER CODE BEGIN (MAIN_Main,3)
244     P3_DATA=LED_OFF;
245     while(RS232_wait);
246     // USER CODE END
247
248     while(1)
249     {
250
251     // USER CODE BEGIN (MAIN_Main,4)
252     printf(menu);
253     select=input();
254
255     switch (select)
      
```
- Output Window:** Shows the command: "Load "C:\XC888\XC888"" and a list of assembly instructions: "ASM ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess".
- Watch Window:** Shows a table with columns "Name" and "Value".
- Status Bar:** Displays "Ready" and "Simulation" with a timer at "t1: 0.00046803 sec".



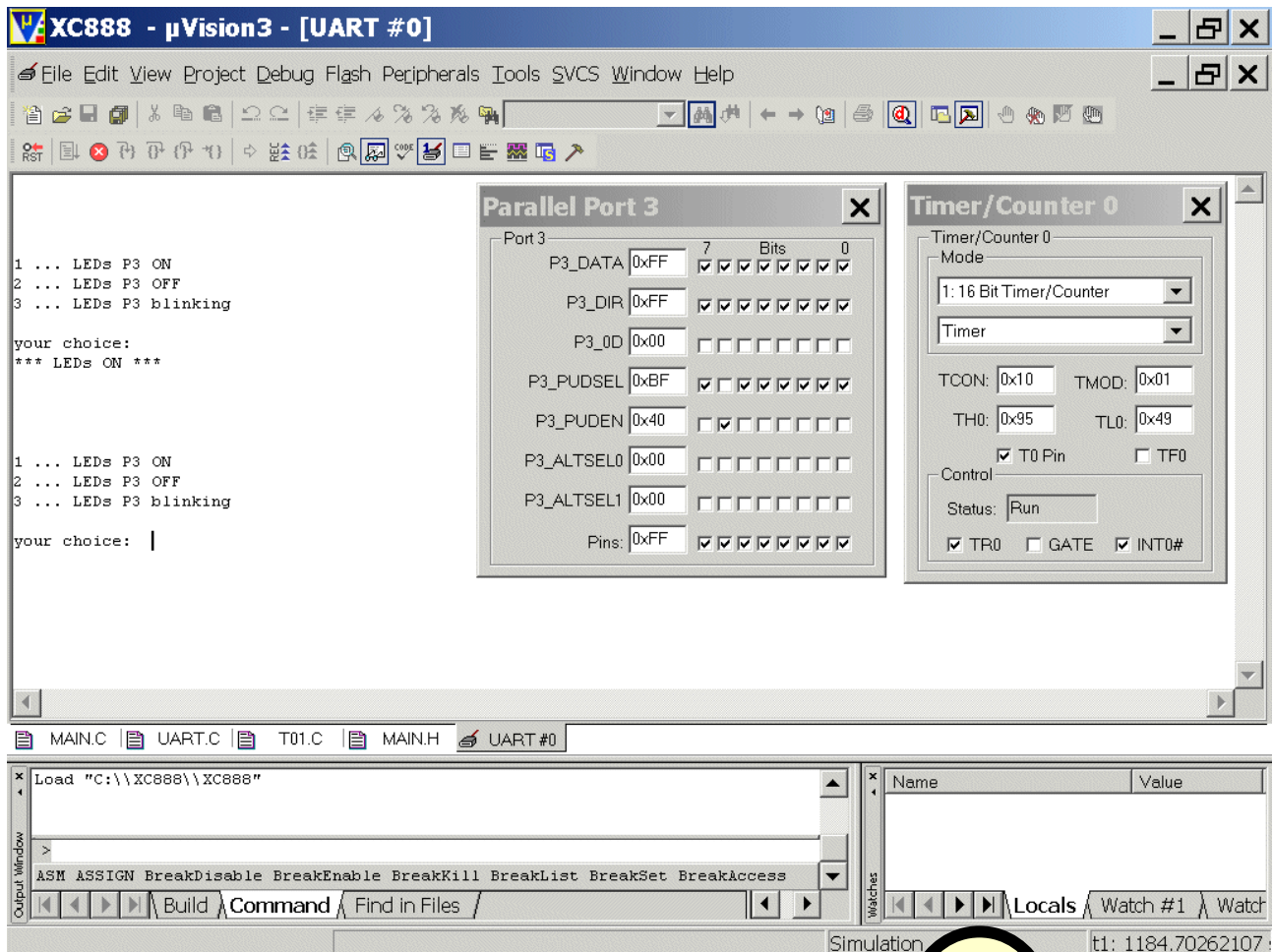
Start program-execution:

Debug – Run

View - Serial Window – UART #0

Peripherals - I/O-Ports – Port3

Peripherals – Timer – Timer0



**Note:**

By activating (clicking) the **UART #0**-window you can then type 1, 2 or 3 and see the result in the Parallel-Port-3-window.



Now we close our simulator session:

Debug - Stop Running

Debug - Start/Stop Debug Session

Now we close our project and µVision 3:

Project - Close Project

File – Exit

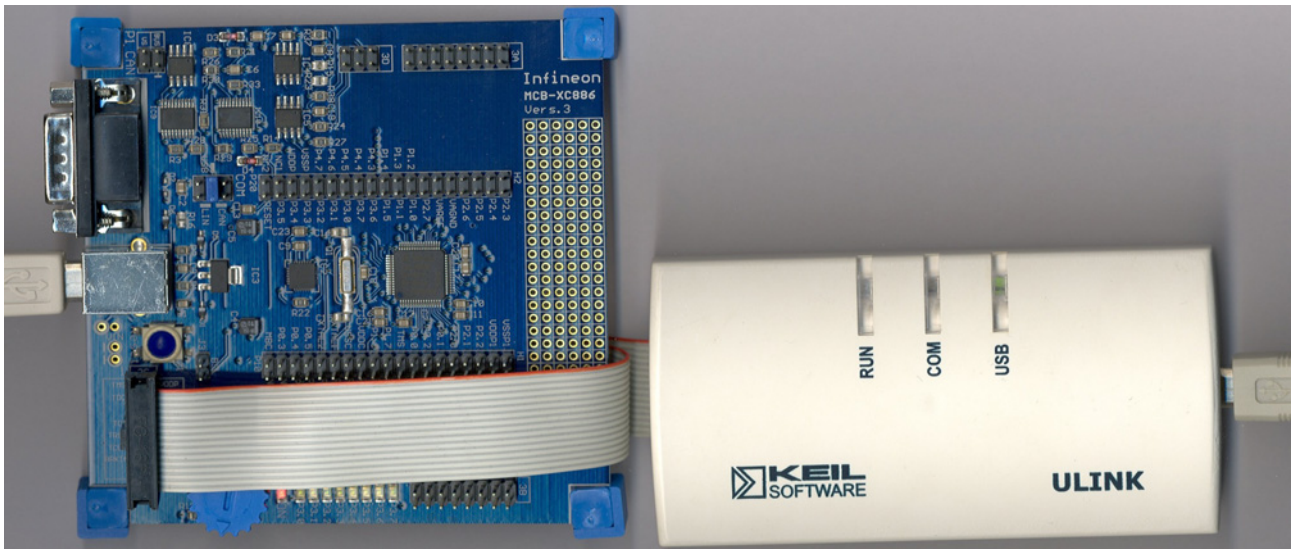


**Note:**

Since our program runs as expected in the simulator we can now use real hardware.

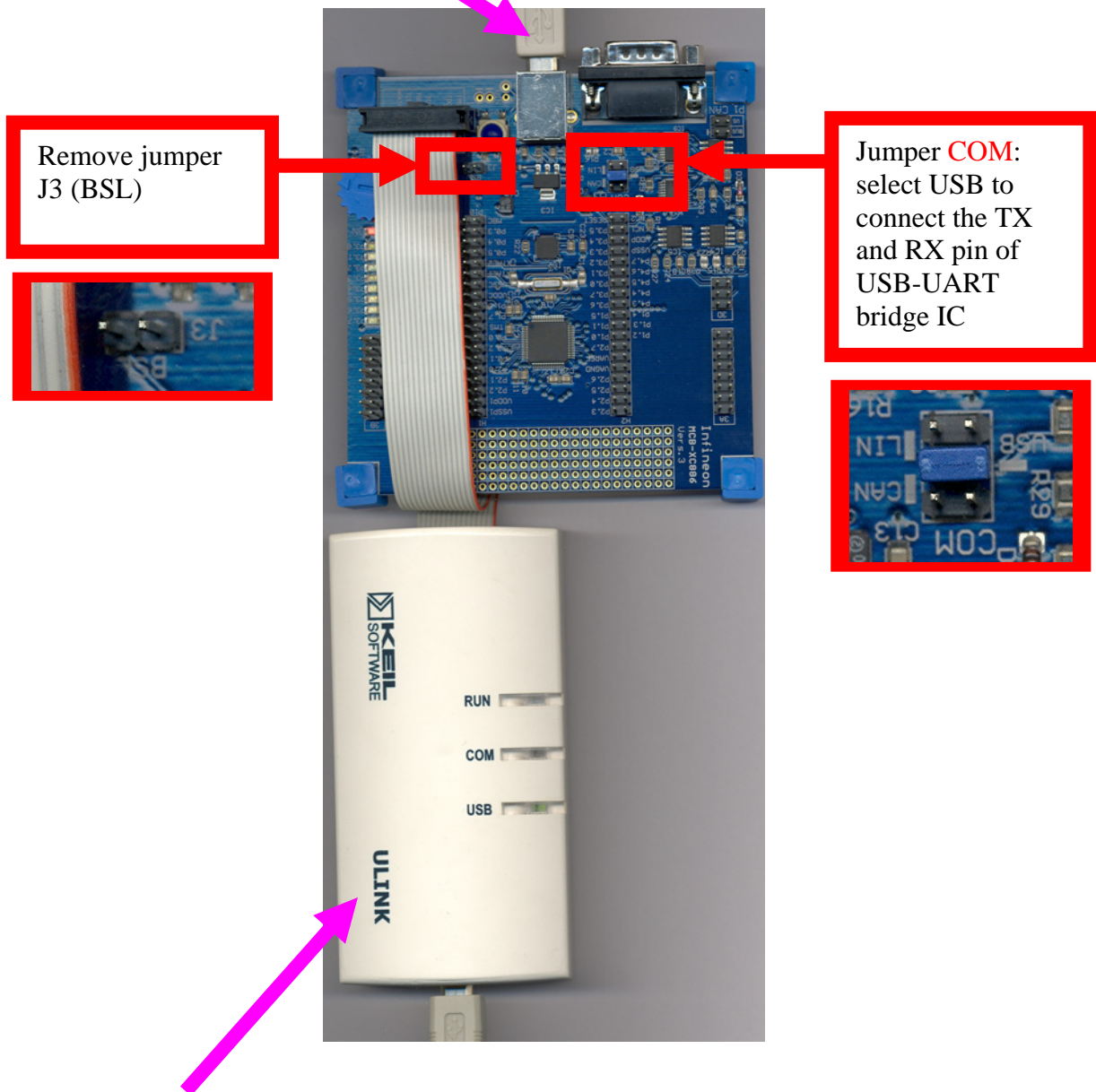
6.) Using real hardware:

Using ULINK [used for: OnChipFlash-Programming and Debugging (using the JTAG interface)]:

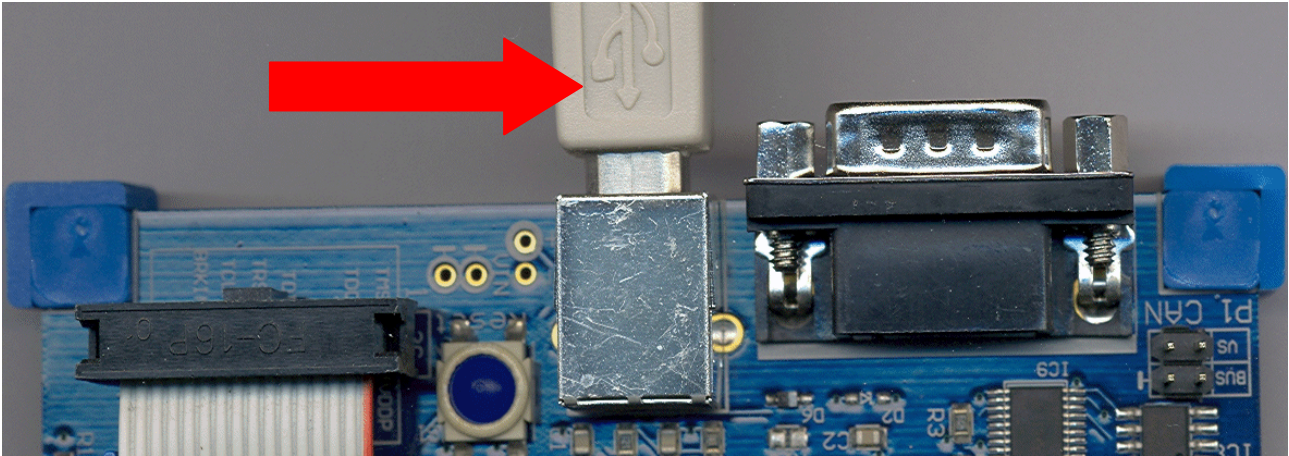


Connecting the XC888-Board to the Environment:

Connect the USB Cable (the RS232 serial interface is available via USB; the USB connection works also as the power supply):



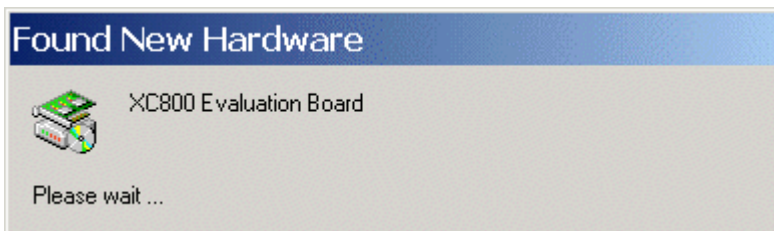
Connect the KEIL-ULINK (used for: On-Chip-Flash-Programming and Debugging).



**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:

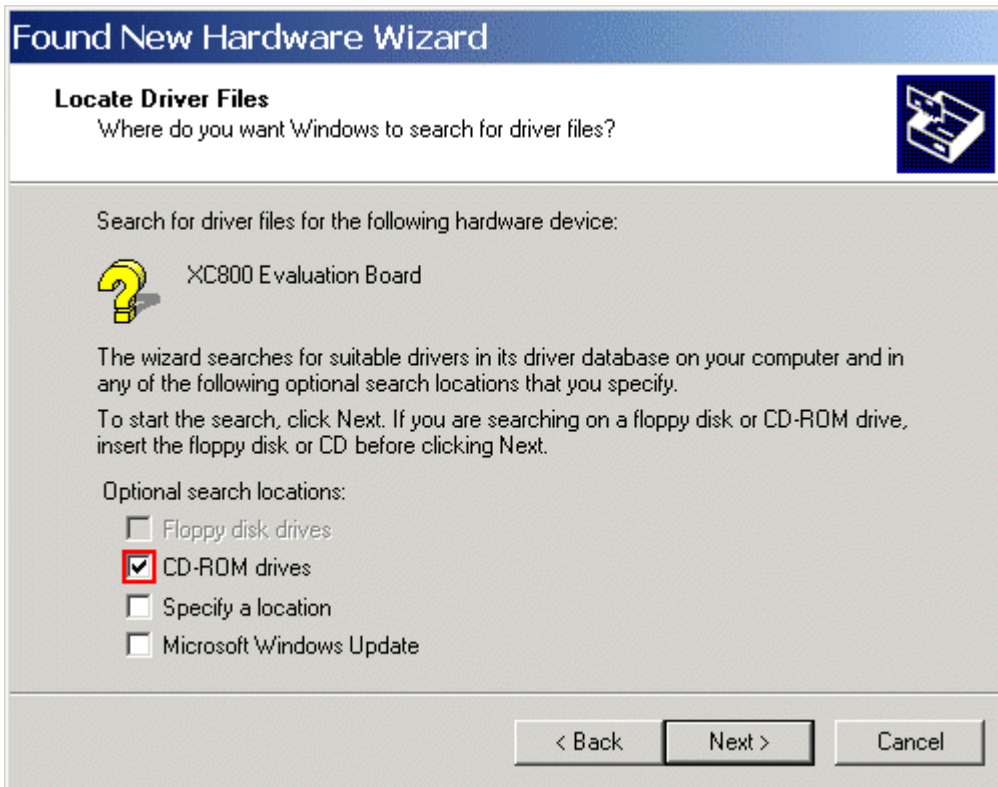




If so: **click** Next



If so: **click** Next



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



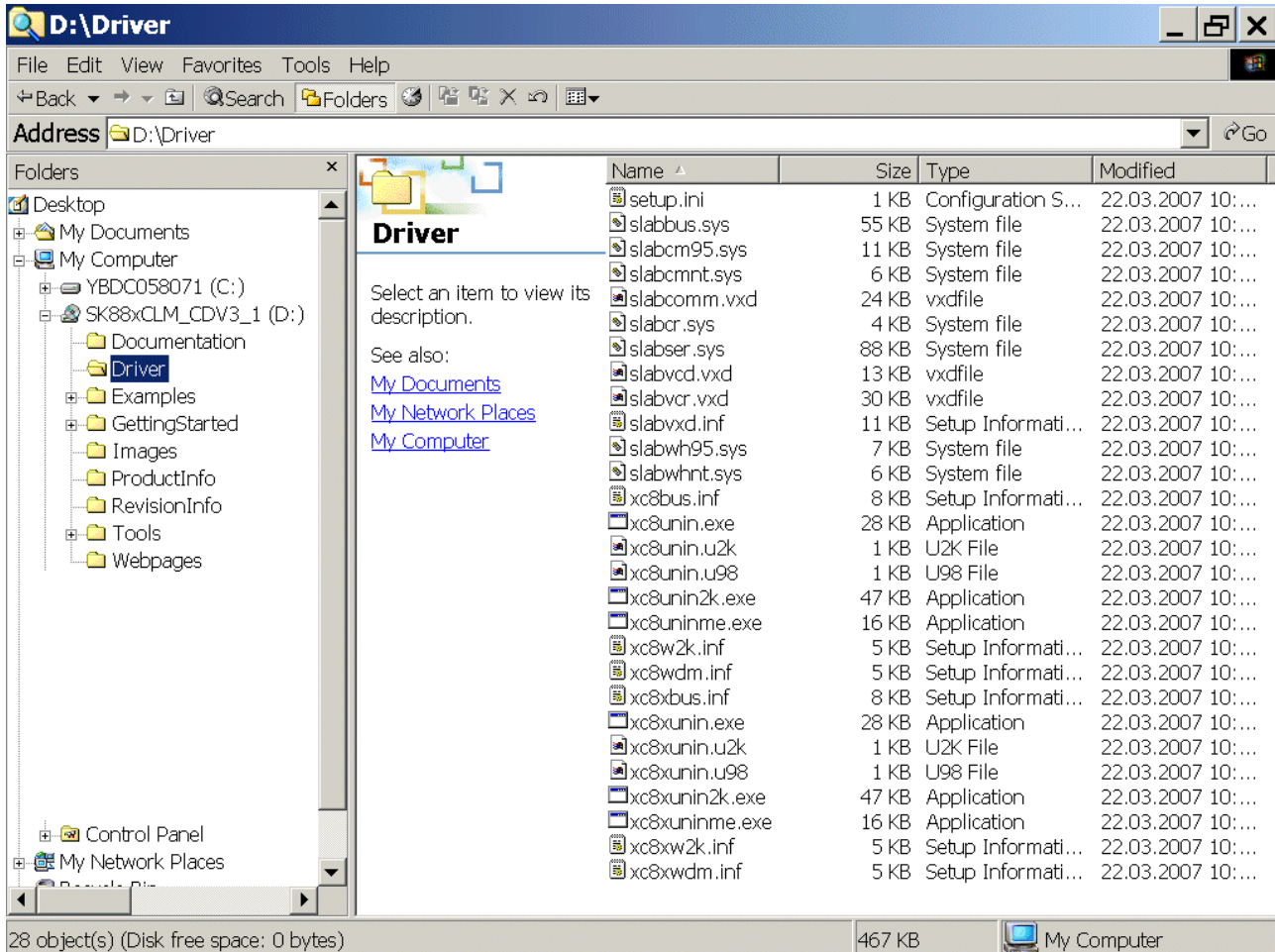
If so: **click** Next



If so: **click** Finish



Or select the USB driver from the directory [SK88xCLM\\_CDV3\\_1\Driver](#) of your XC88x Starter Kit CD:





**Note:**

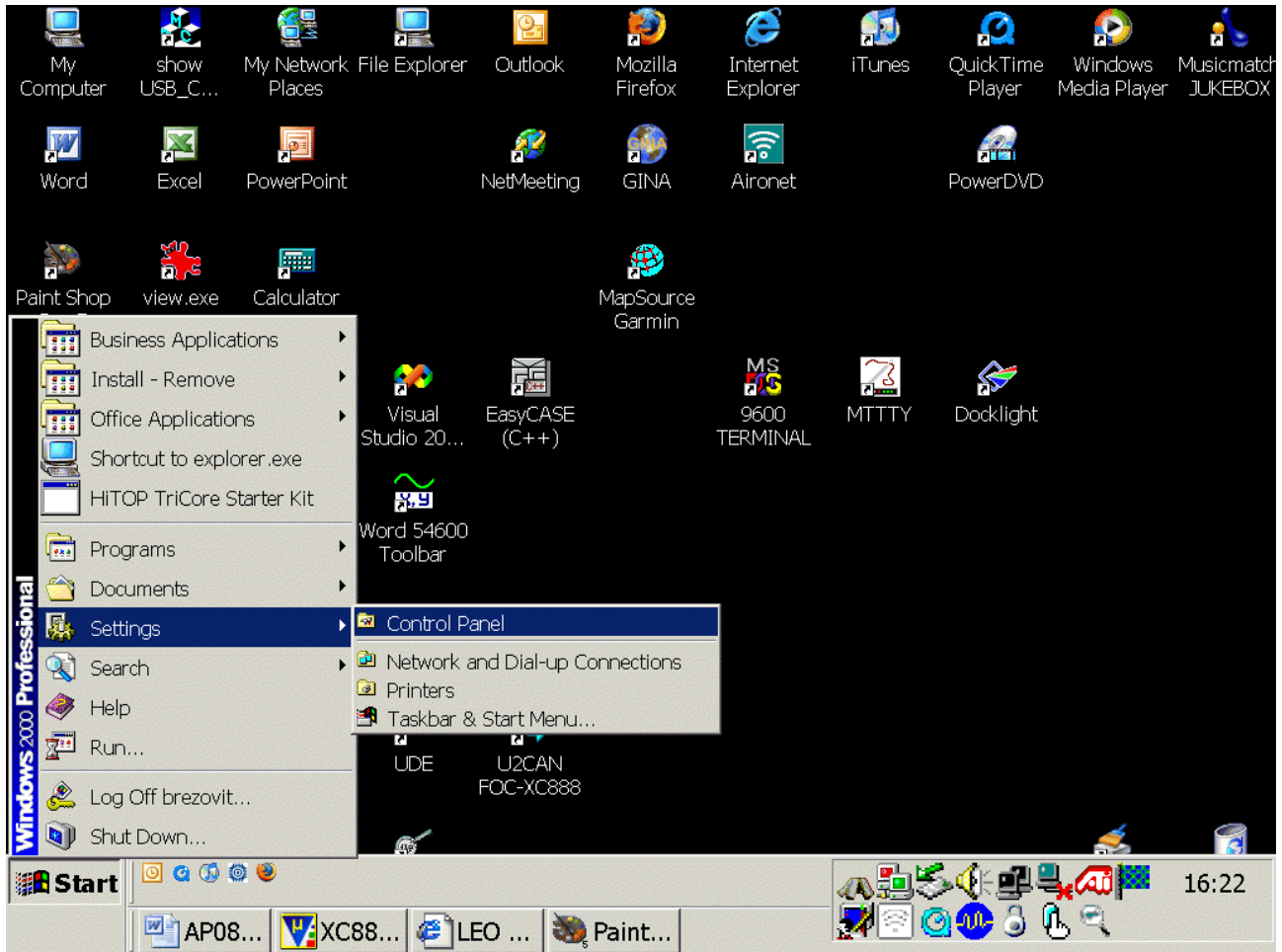
Skip this step when the USB driver is auto-detected and auto-installed.

**Note:**

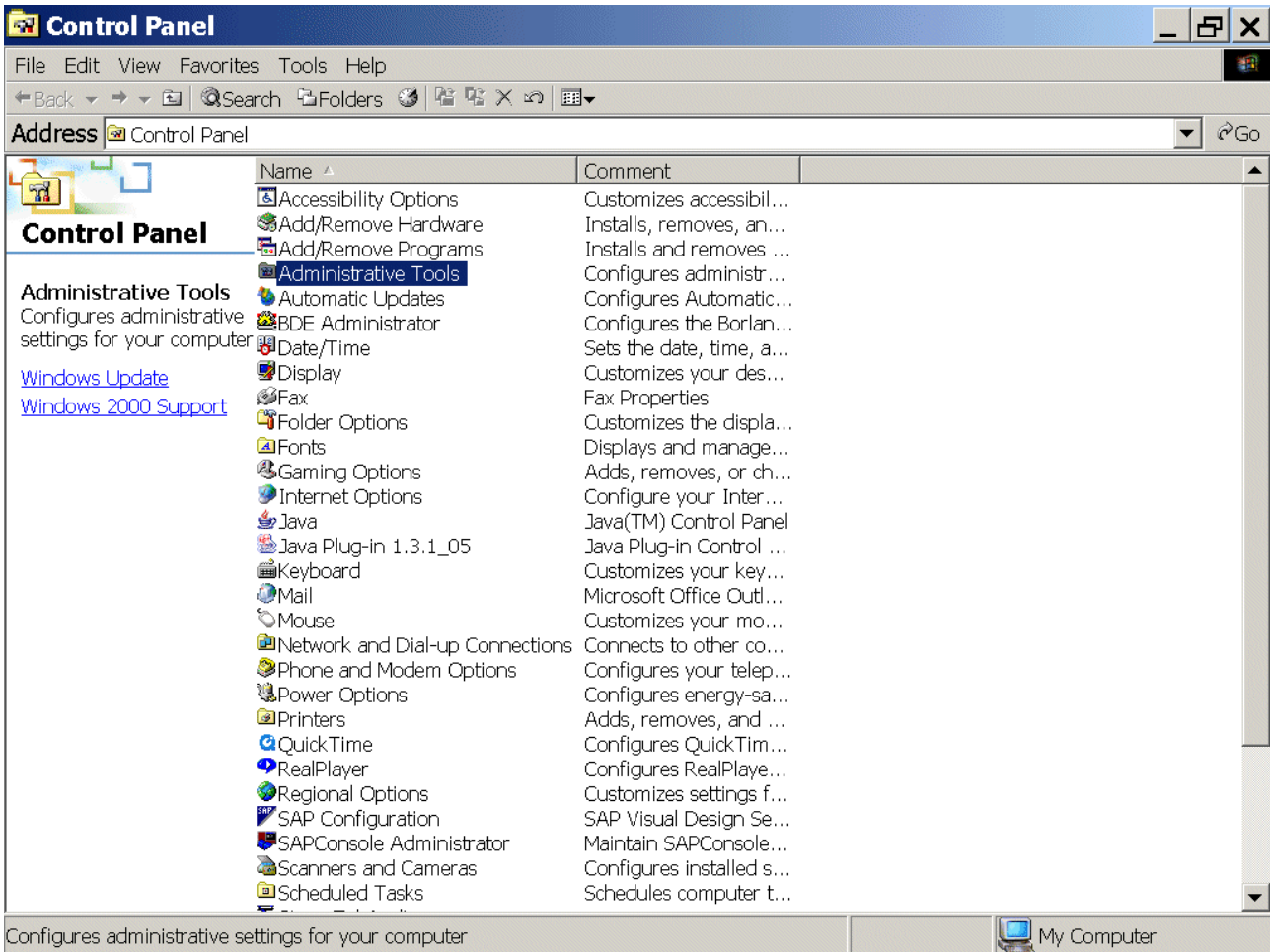
A default COM Port is generated after the USB driver is installed.

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:

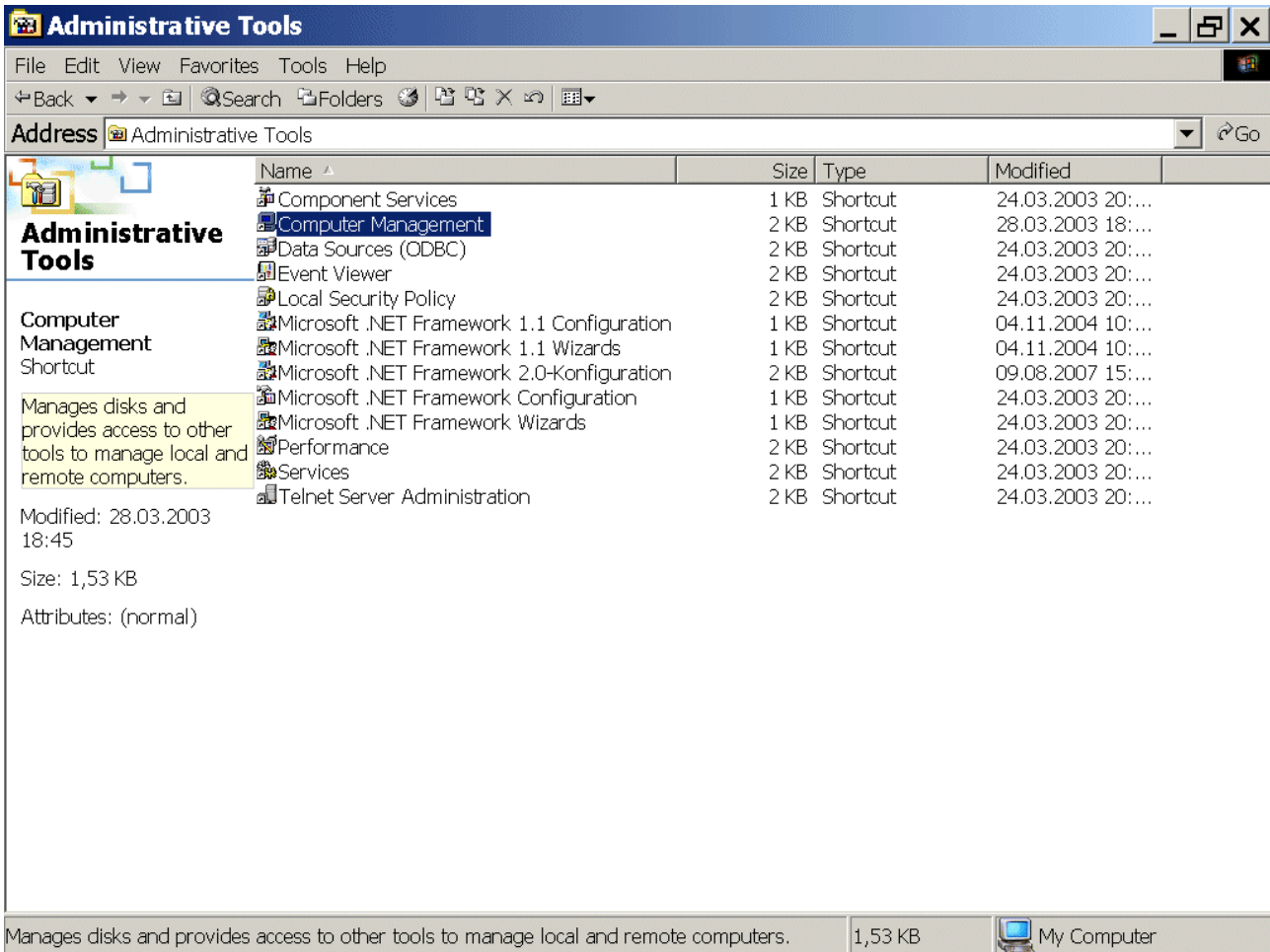
### Start – Settings – Control Panel



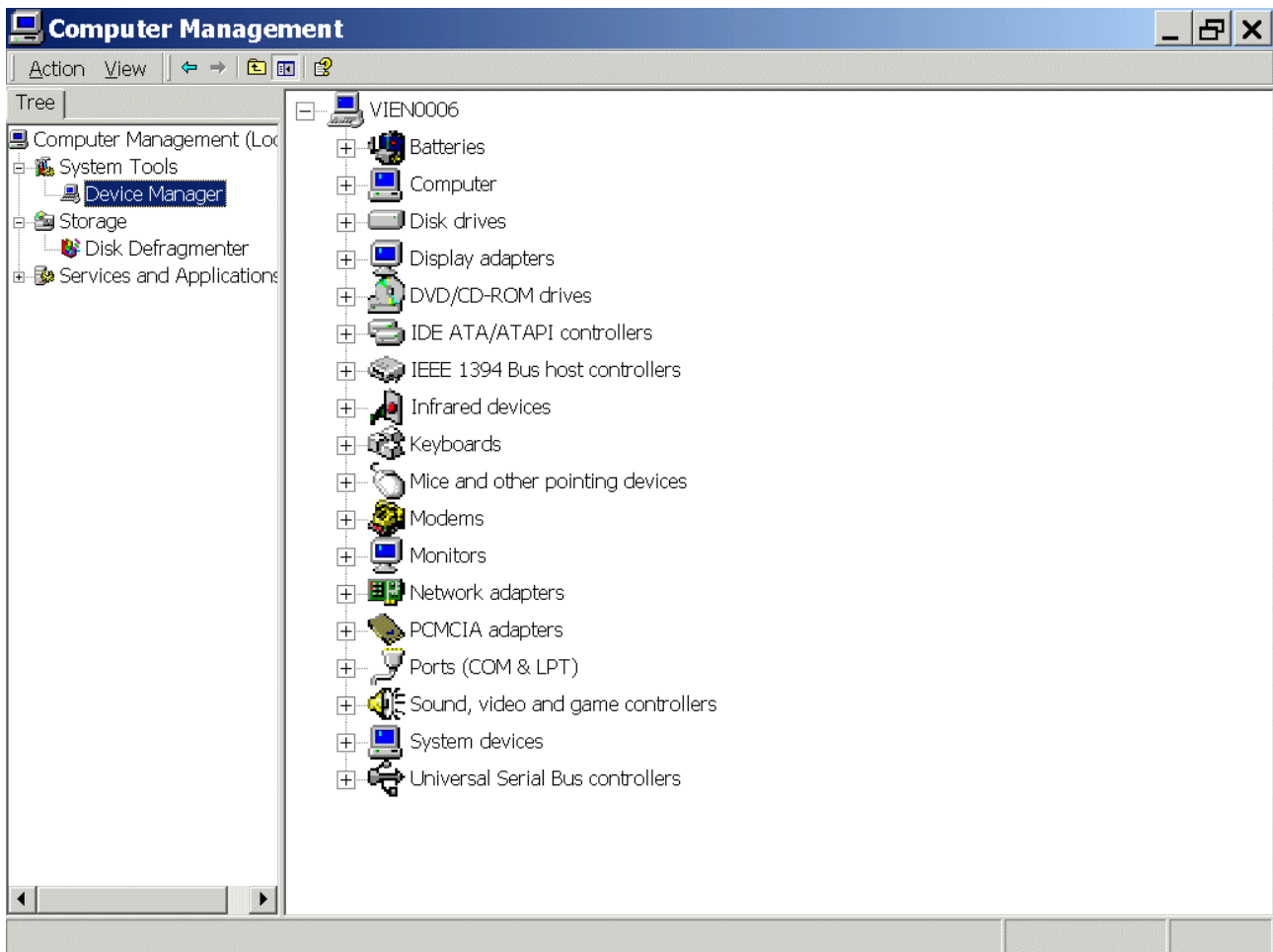
Double click: Administrative Tools



Double click: Computer Management

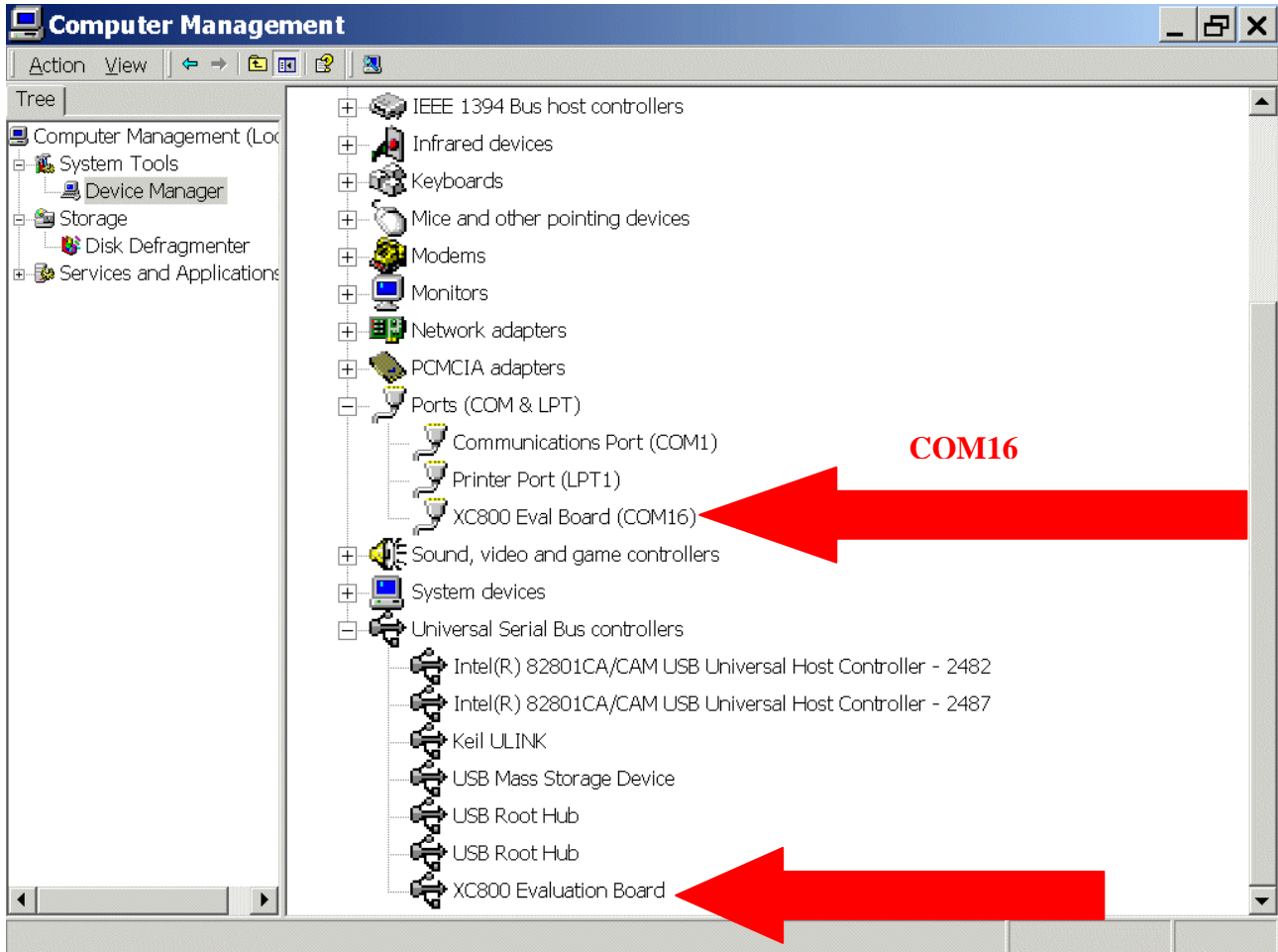


Device Manager:



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 **COM16** !



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):

<b>Data Bits</b>	5, 6, 7, and 8
<b>Stop Bits</b>	1, 1.5 <sup>1</sup> , and 2
<b>Parity Type</b>	None, Even, Odd, Mark, Space
<b>Baud Rates</b> <sup>2</sup>	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 <sup>3</sup>
<b>Notes:</b>	
<ul style="list-style-type: none"> <li>3. 5-bit only.</li> <li>4. Additional baud rates are supported. See "AN205".</li> <li>5. 7 or 8 data bits only.</li> </ul>	

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](#) .





Start Keil  $\mu$ Vision and open our Keil Project

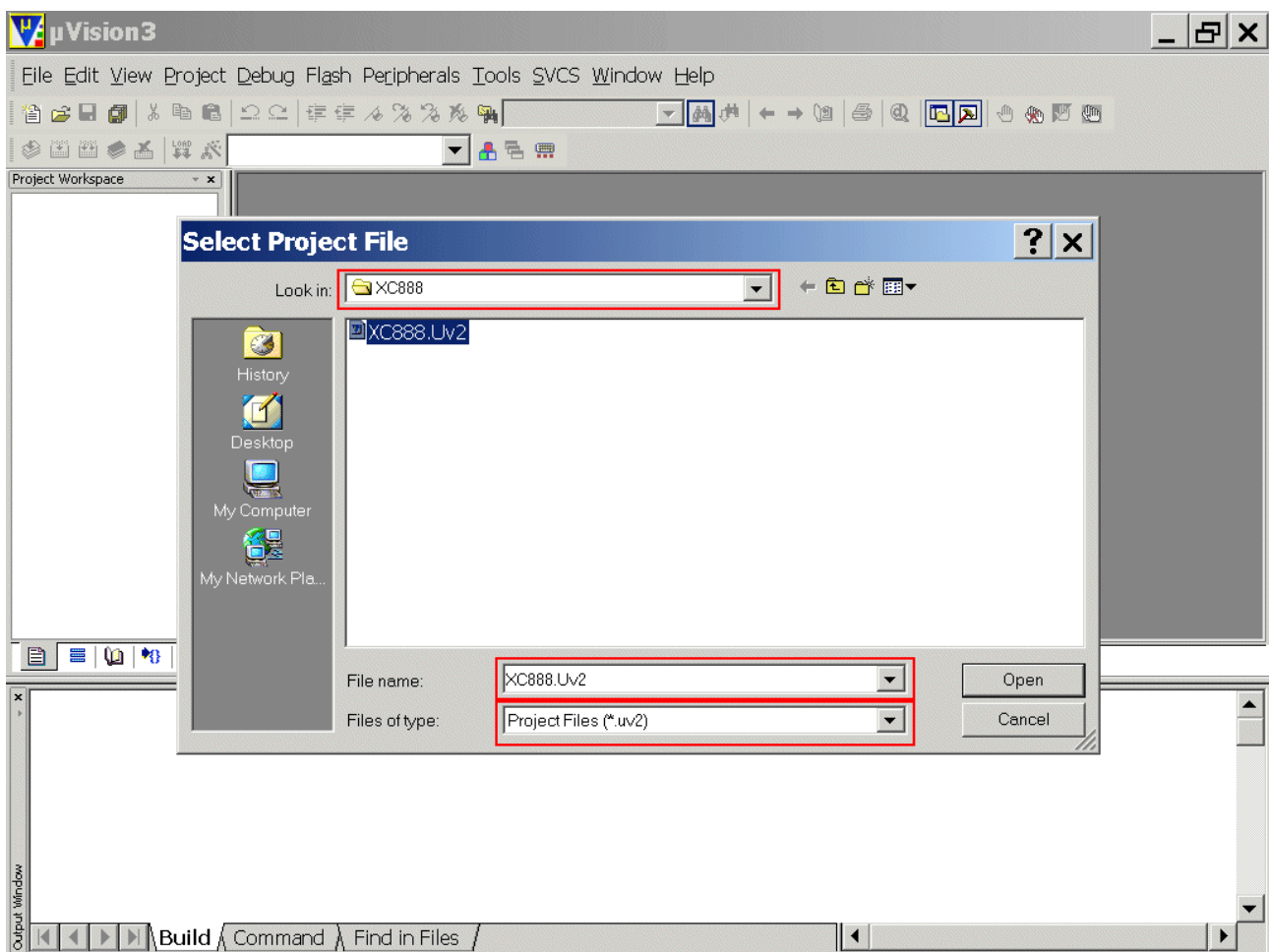
If you see an open project – close it: **Project - Close Project**

**Project - Open Project**

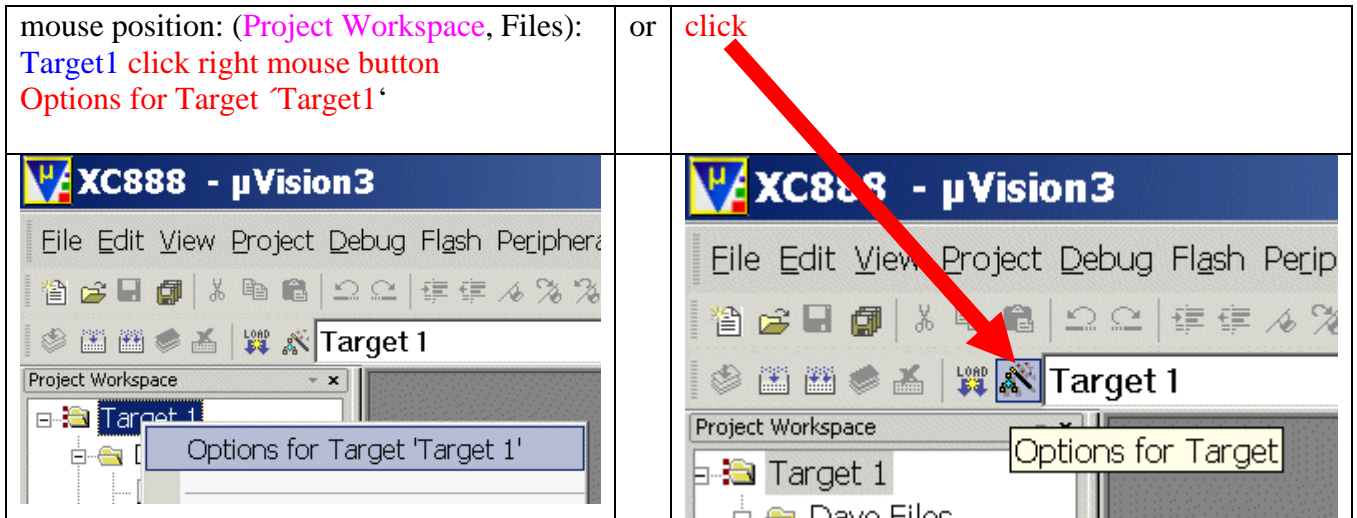
Select Project File: **Look in:** choose C:\XC888

Select Project File: **Files of type:** choose Project Files (\*.uv2)  
choose XC888.Uv2

**Open**



Check the configuration of the Flash-Programming-Utility:

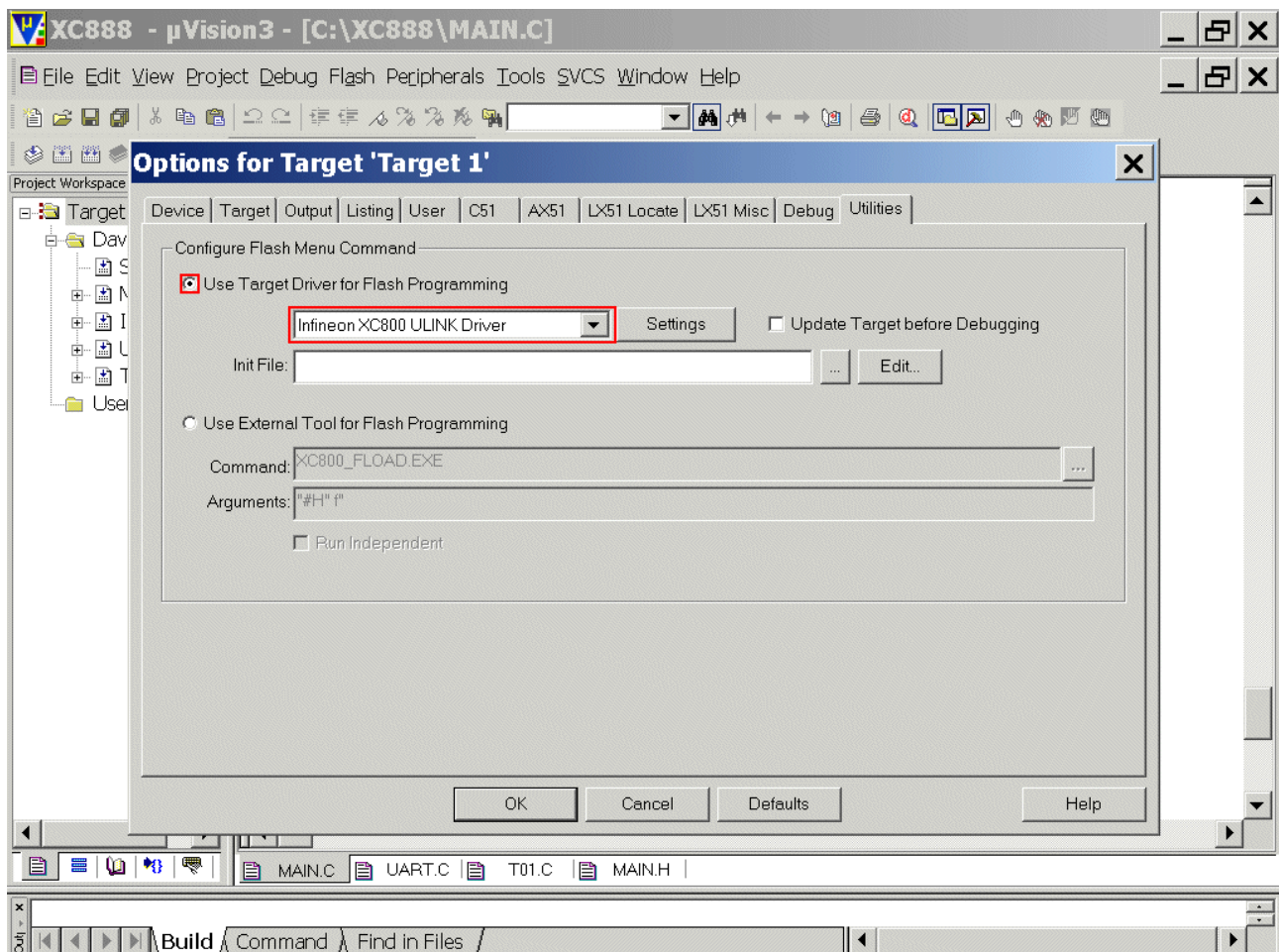


Options for Target 'Target1':

Utilities: Configure Flash Menu Command: check  Use Target Driver for Flash Programming

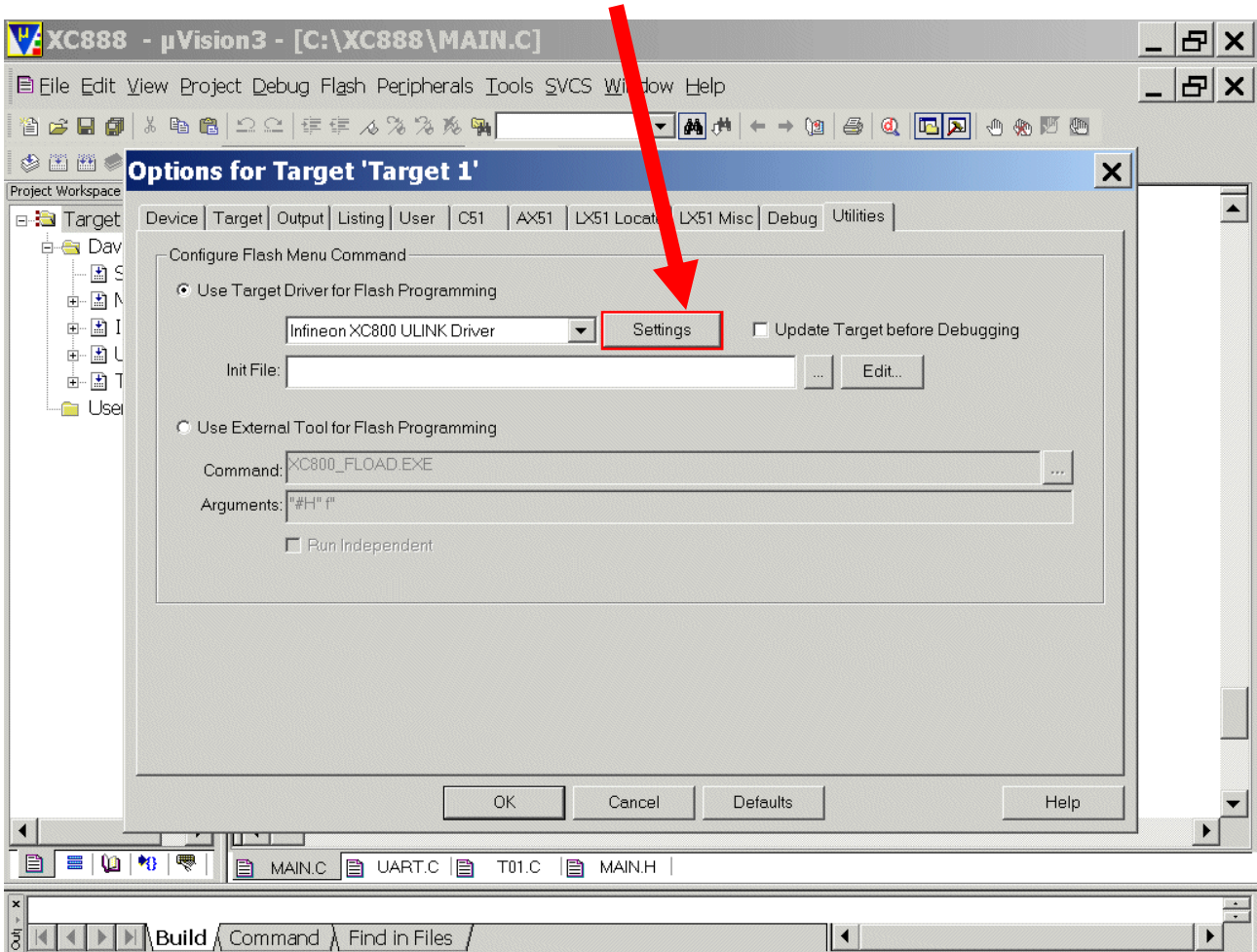
Options for Target 'Target1':

Utilities: Configure Flash Menu Command: check Infineon XC800 ULINK Driver

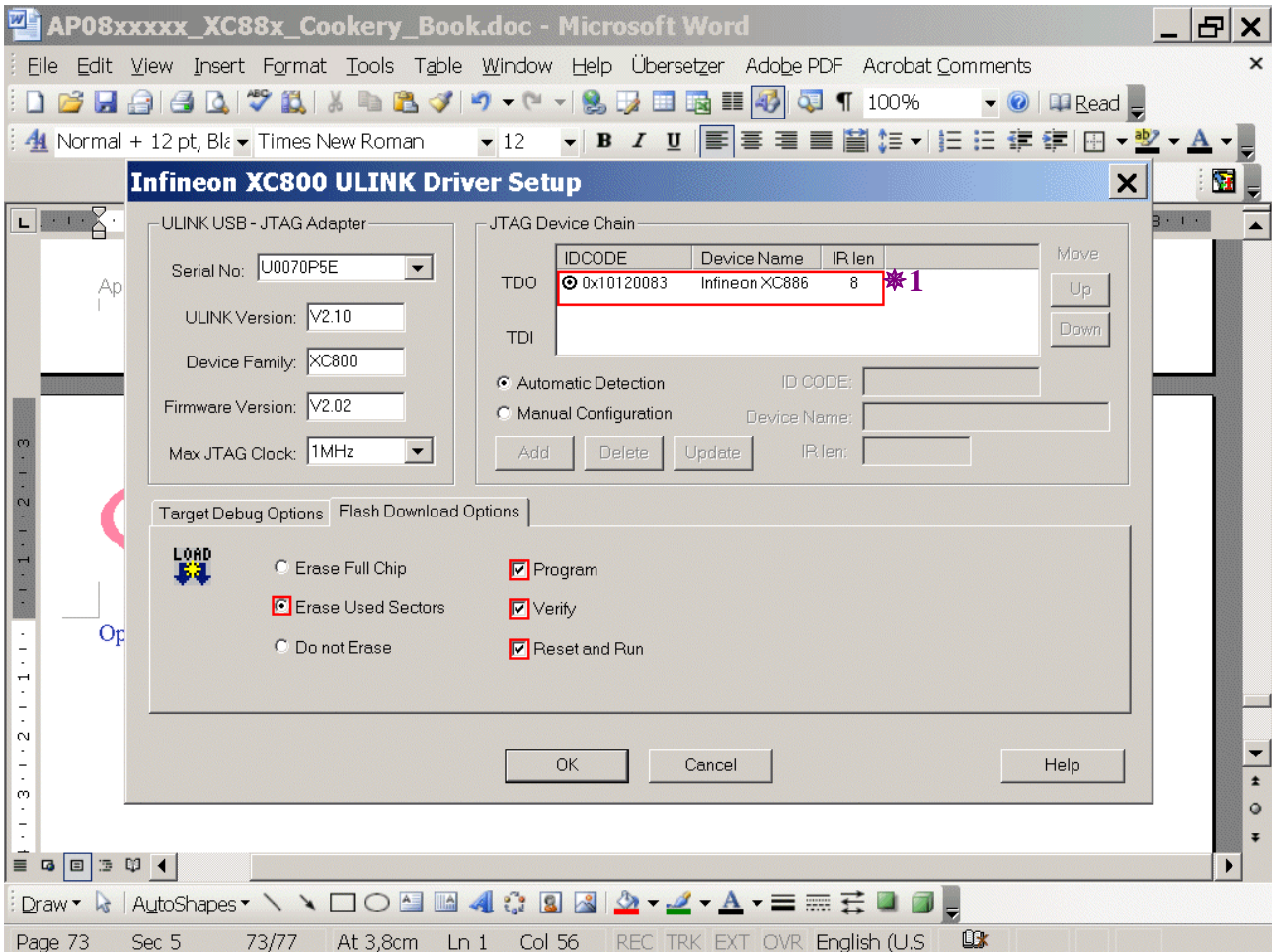


Options for Target 'Target1':

Utilities: Configure Flash Menu Command: **click** Settings



- Flash Download Options: **check:**  Erase Used Sectors  
 Flash Download Options: **check:**  Program  
 Flash Download Options: **check:**  Verify  
 Flash Download Options: **check:**  Reset and Run



OK  
OK

**Note:**

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

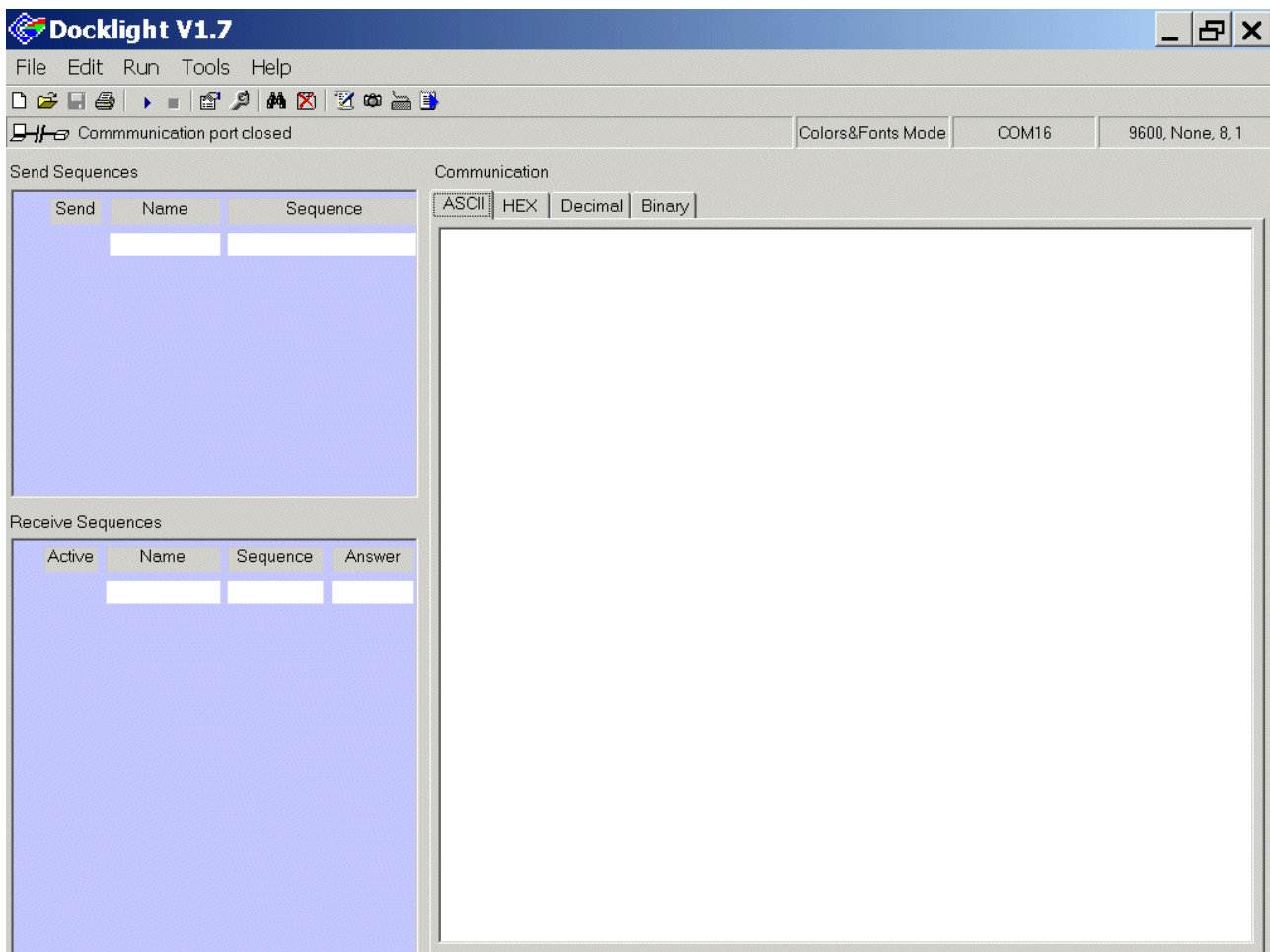


**Note:**

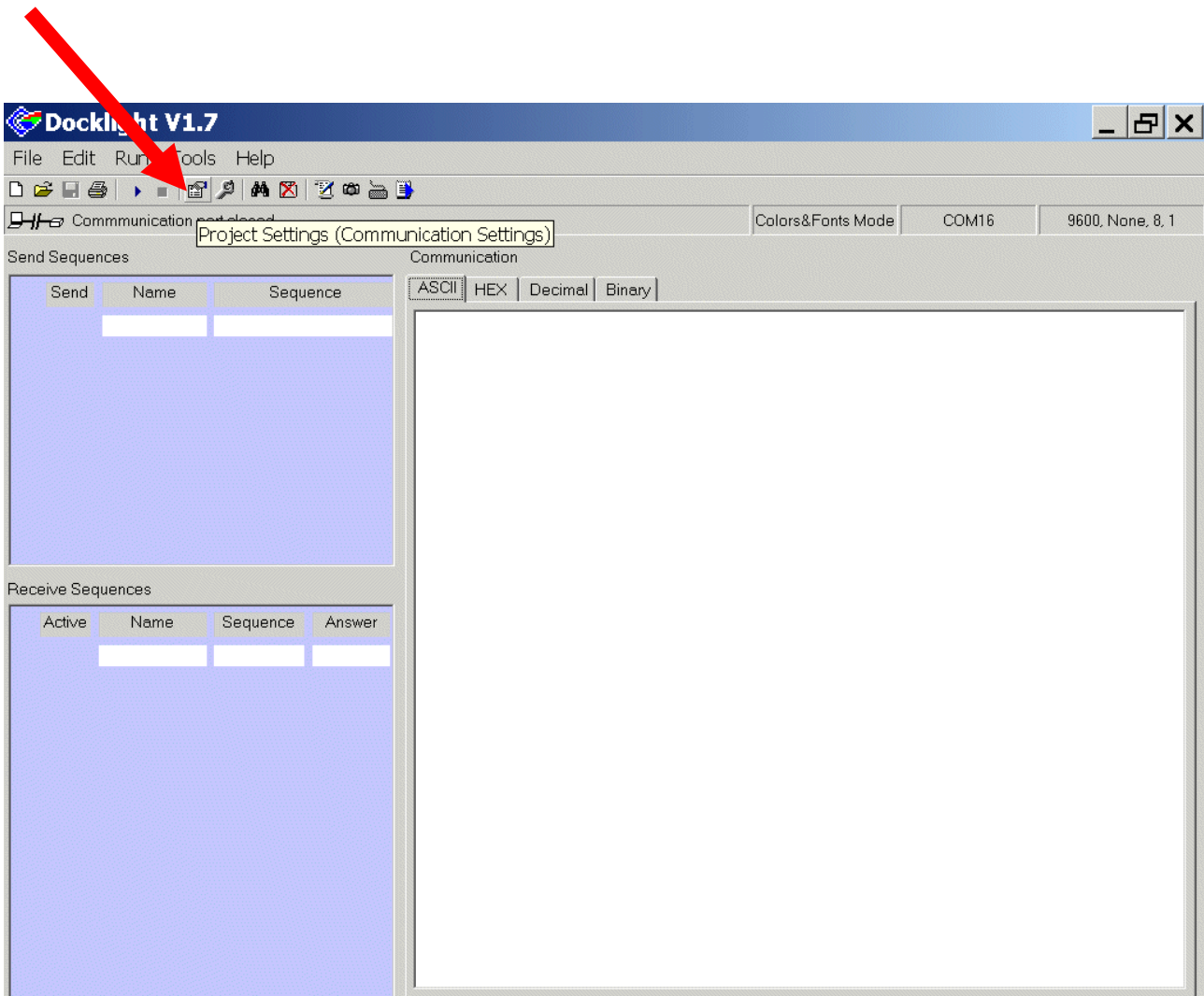
Now we need a terminal program which is able to handle COM16!  
As an example of "any terminal program" we are going to use Docklight.  
Docklight can be downloaded @ <http://www.docklight.de> .



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** COM16

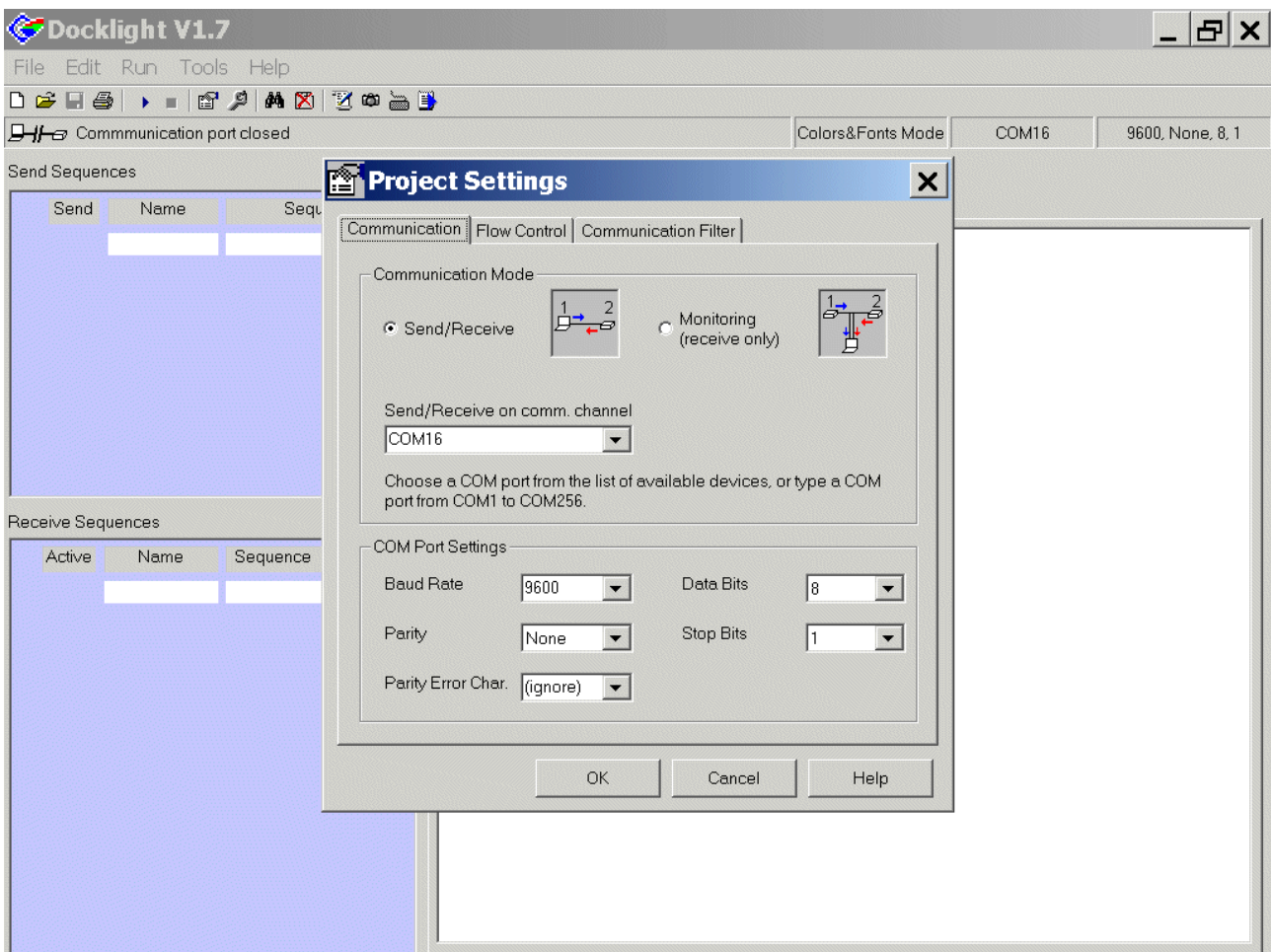
Project Settings: Communication: COM Port Settings: **Baud Rate:** **select** 9600

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

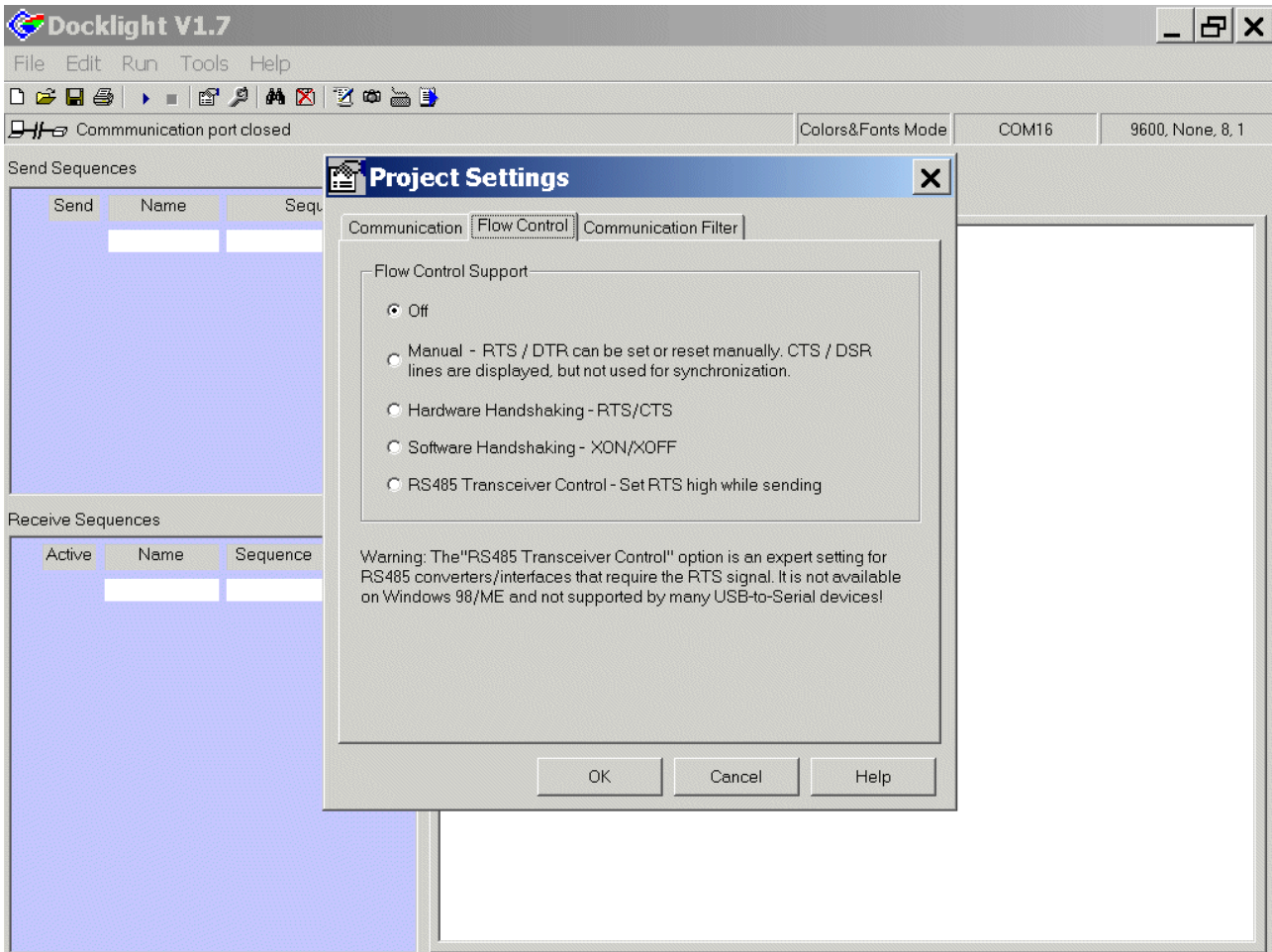
Project Settings: Communication: COM Port Settings: **Parity Error Char.:** **select** (ignore)

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

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



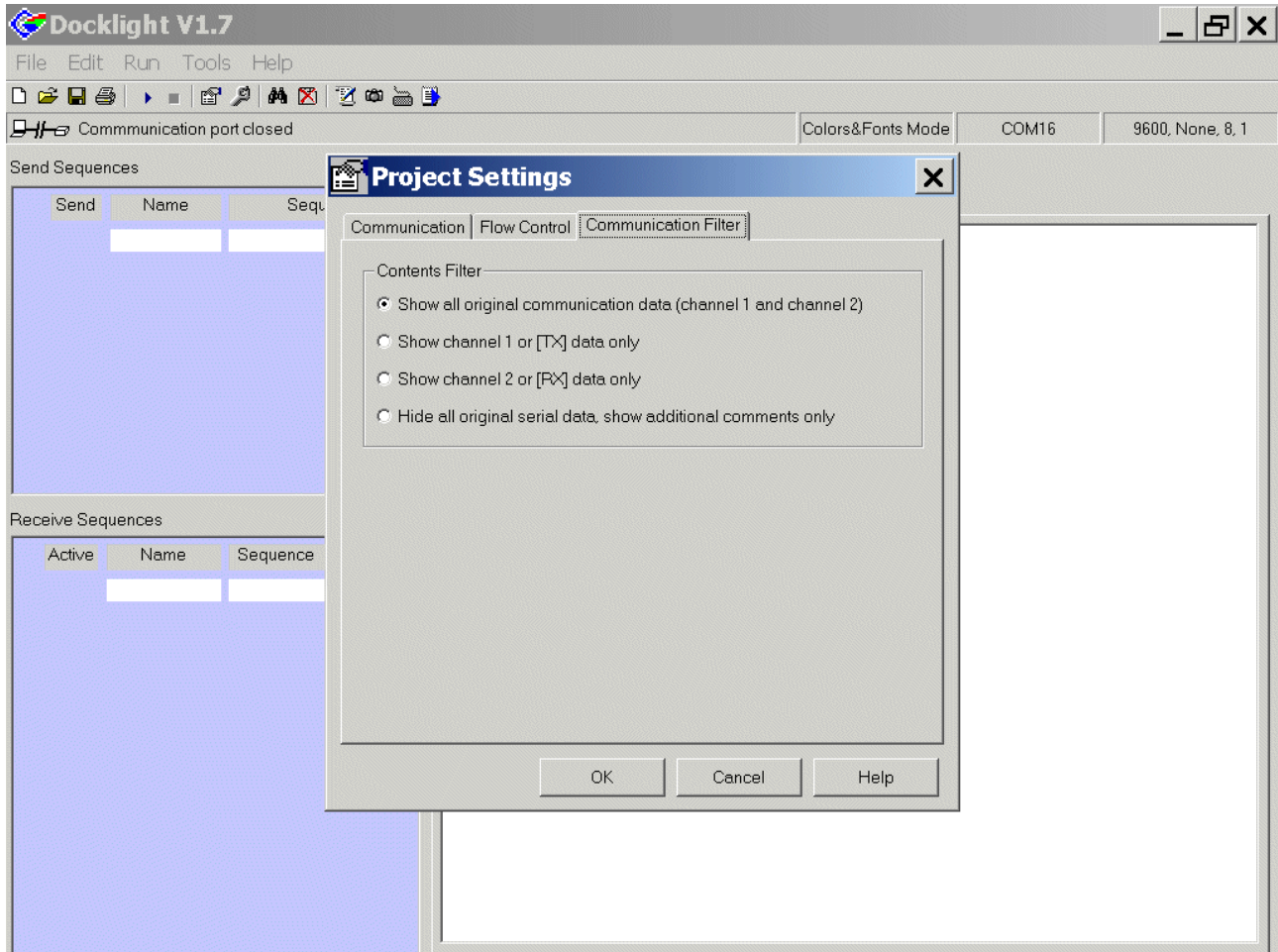
Project Settings: Flow Control: Flow Control Support: **click**  Off





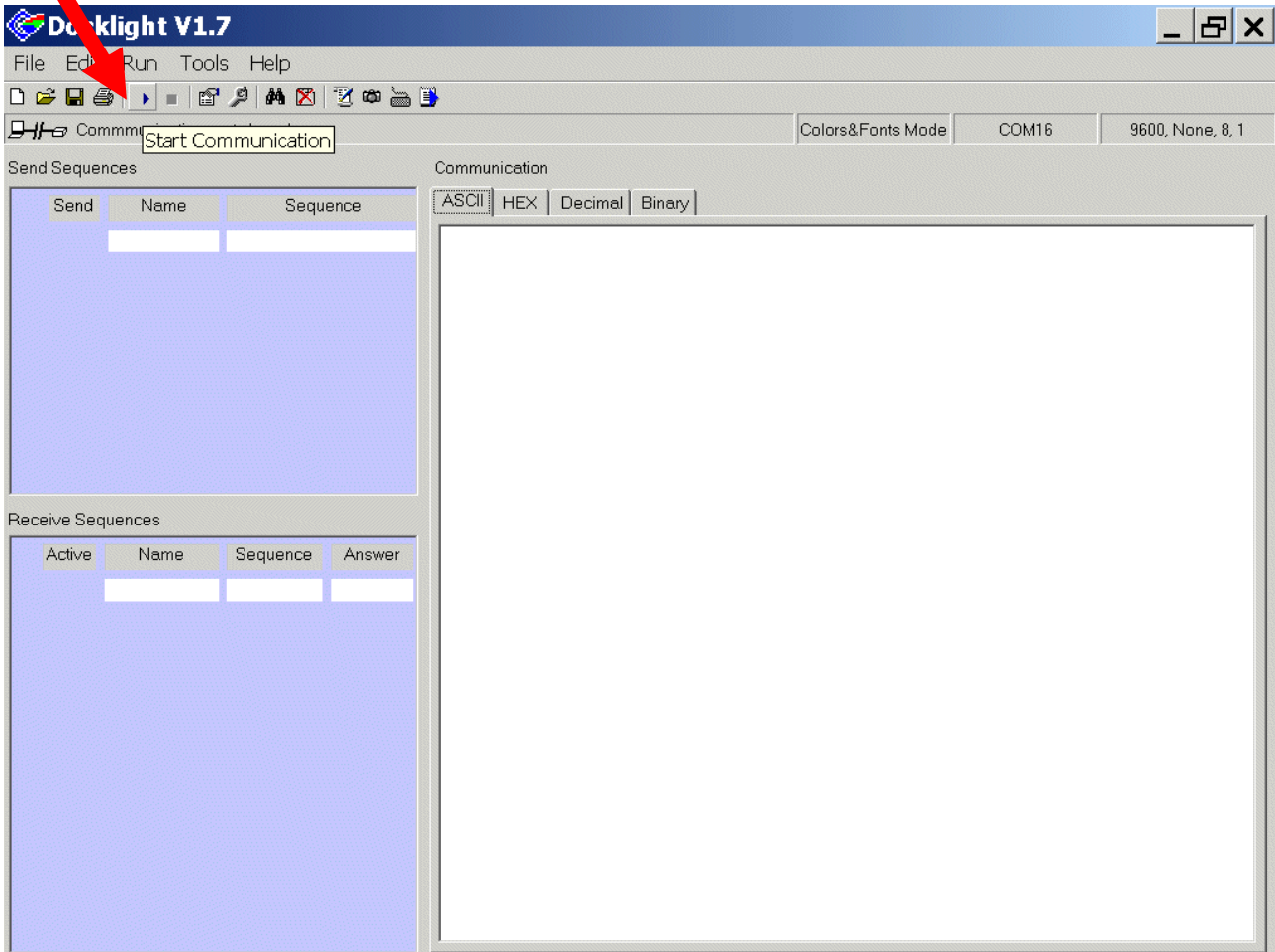
Project Settings:

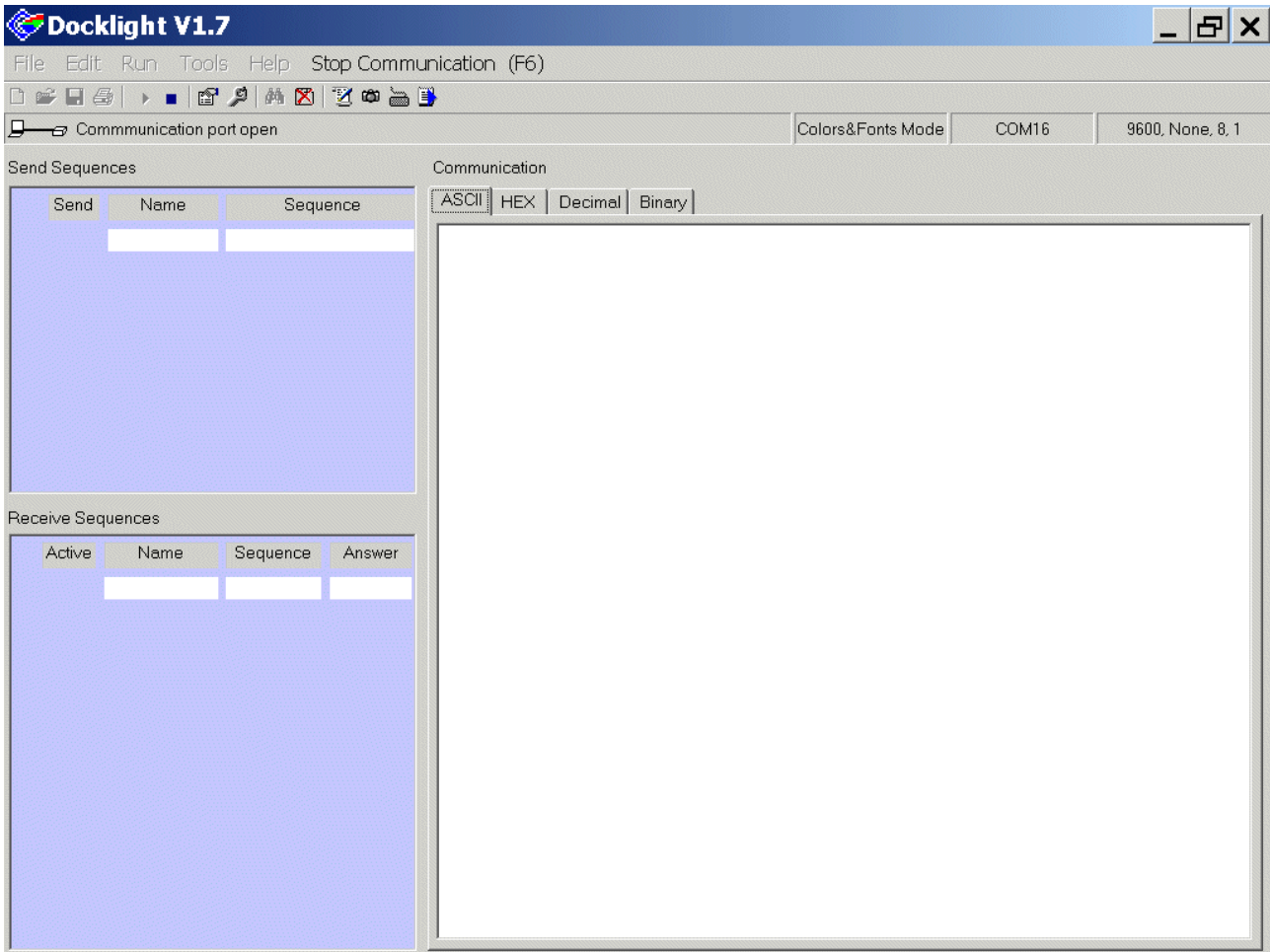
Communication Filter: Contents Filter: **click**  Show all original communication data



OK

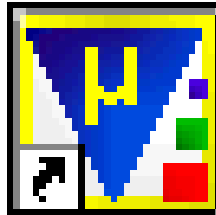
Click: 





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





Go back to  $\mu$ Vision:



The screenshot shows the µVision3 IDE interface for the XC888 project. The Project Workspace on the left lists files: Target 1, Dave Files, START\_XC, MAIN.C, IO.C, UART.C, T01.C, and User Files. The main editor displays the following C code:

```

248 while (1)
249 {
250
251 // USER CODE BEGIN (MAIN_Main,4)
252 printf(menu);
253 select=input();
254
255 switch (select)
256 {
257 case '1': blinking=OFF; P3_DATA=LED_ON, printf(message1); break;
258 case '2': blinking=OFF; P3_DATA=LED_OFF, printf(message2); break;
259 case '3': blinking=ON, printf(message3); break;
260 }

```

The Output Window at the bottom shows the following build and programming log:

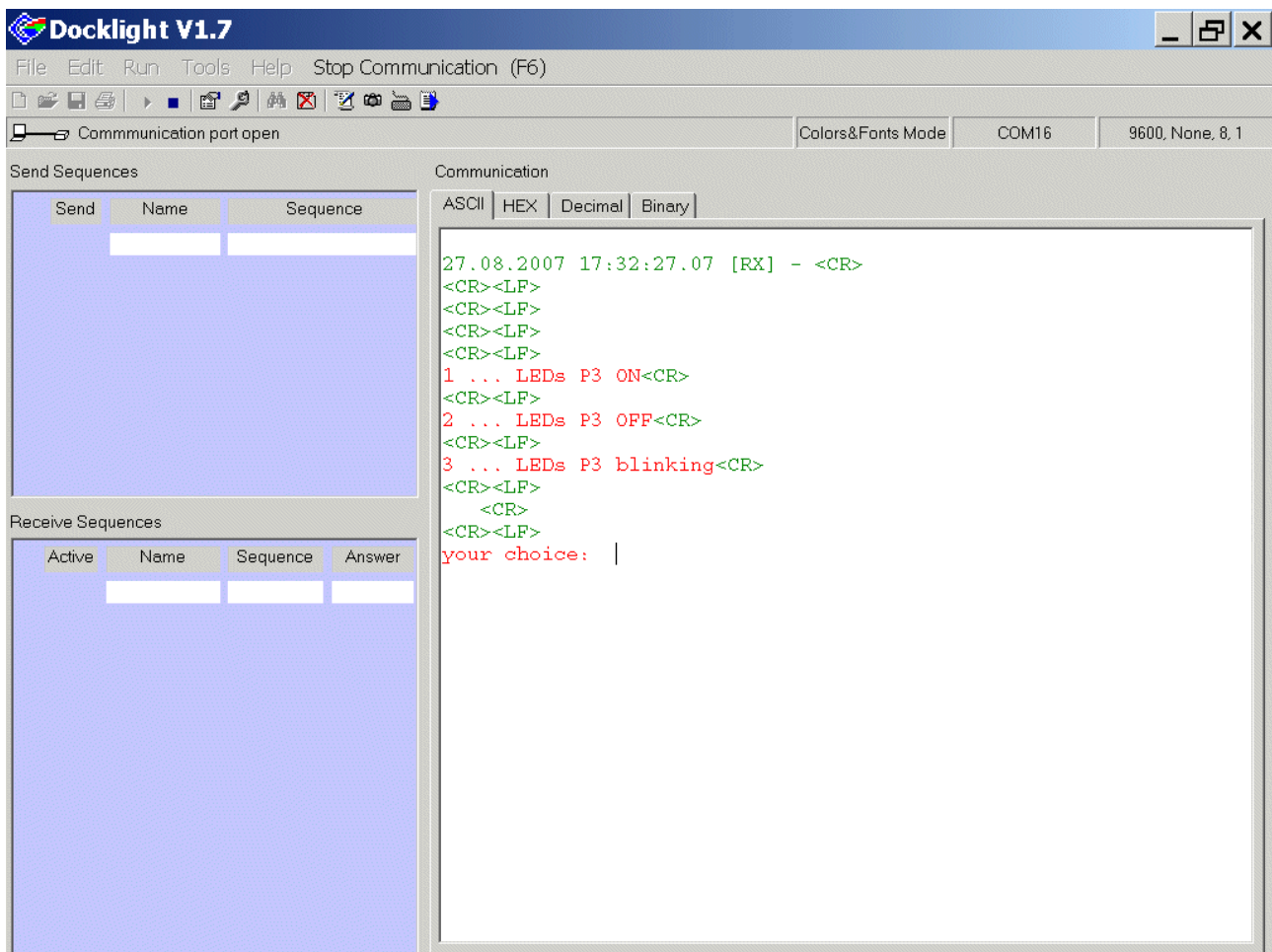
```

Build target 'Target 1'
assembling START_XC.a51...
compiling MAIN.C...
compiling IO.C...
compiling UART.C...
compiling T01.C...
linking...
Program Size: data=33.2 xdata=2 const=154 code=1476
creating hex file from "XC888"...
"XC888" - 0 Error(s), 0 Warning(s).
Load "C:\XC888\XC888"
Erasing Bank 0 Sectors: 0
Erase Done.
Programming Done.
Verify Done.
Application running ...

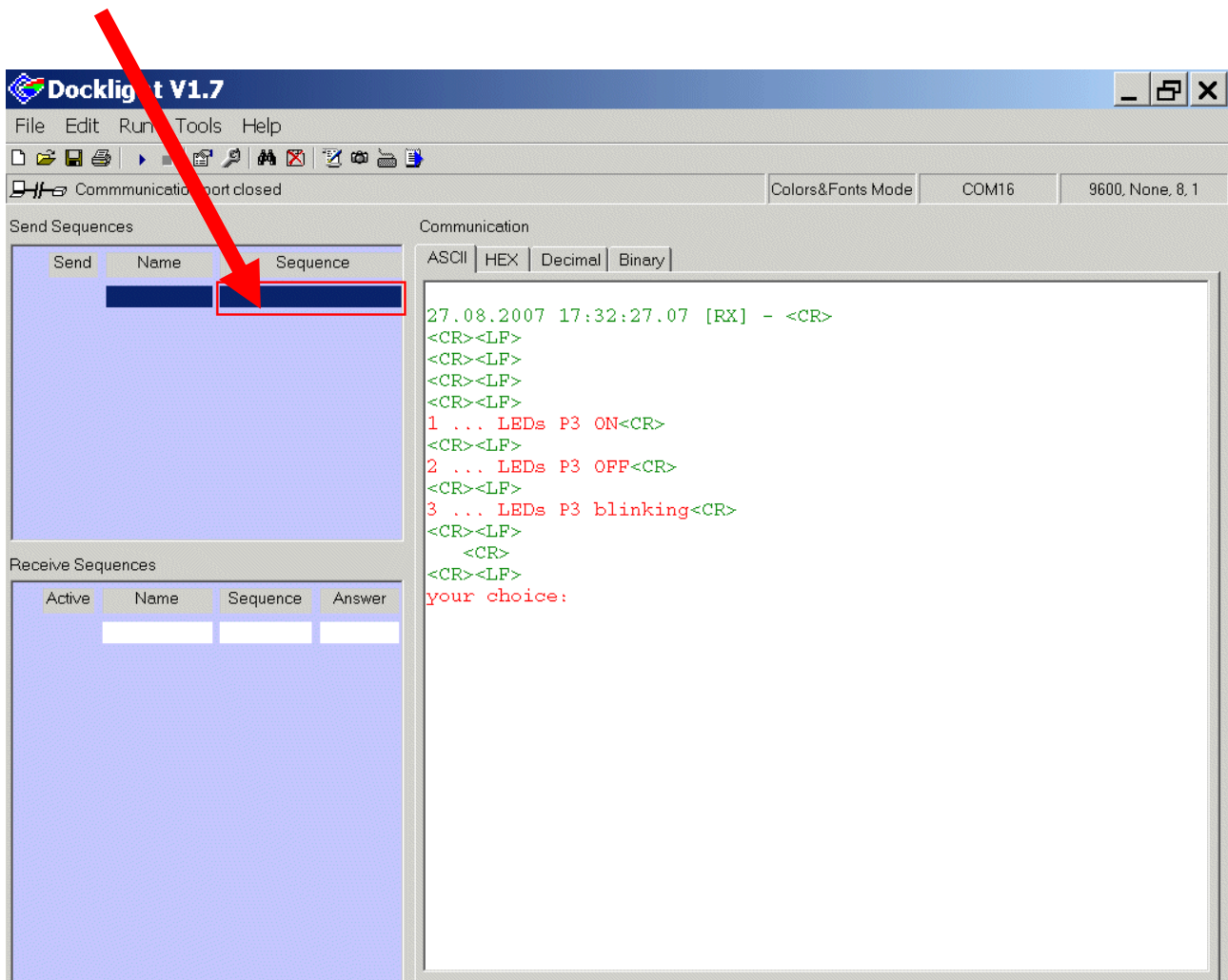
```



Go back to Docklight and see the result:

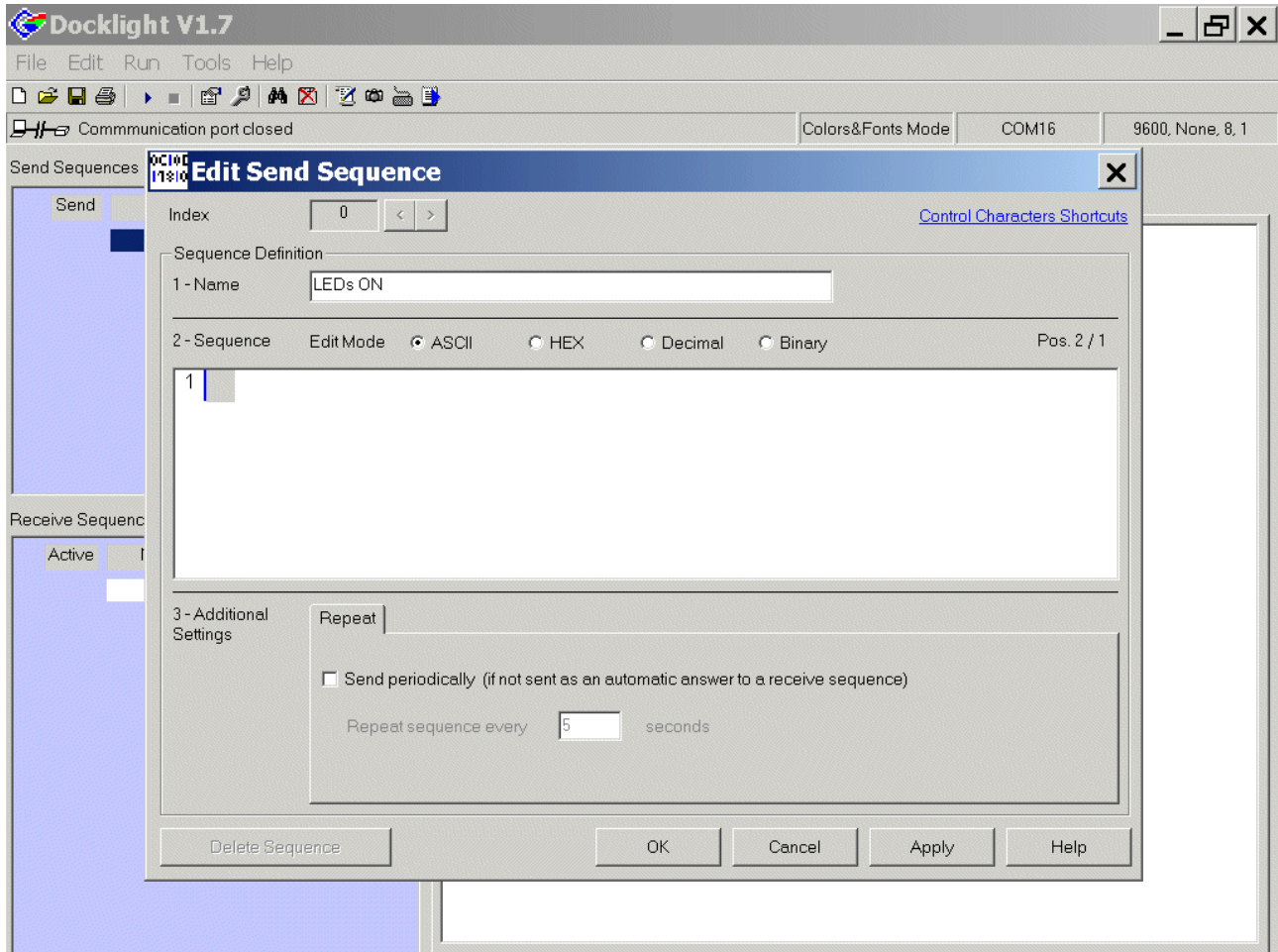


Double click inside the red box:



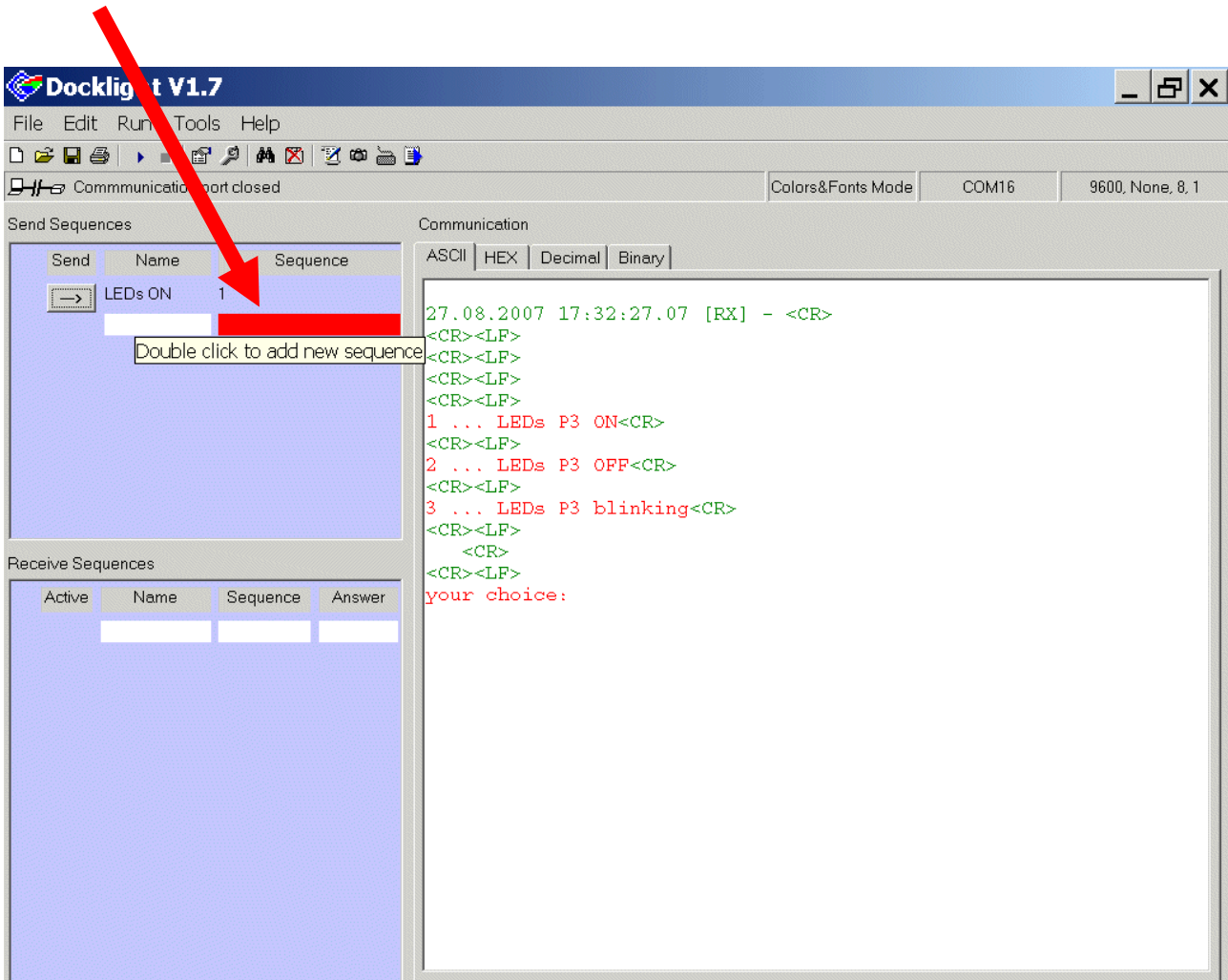
Edit Send Sequence: Sequence Definition: 1- Name: insert: LEDs ON

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



OK

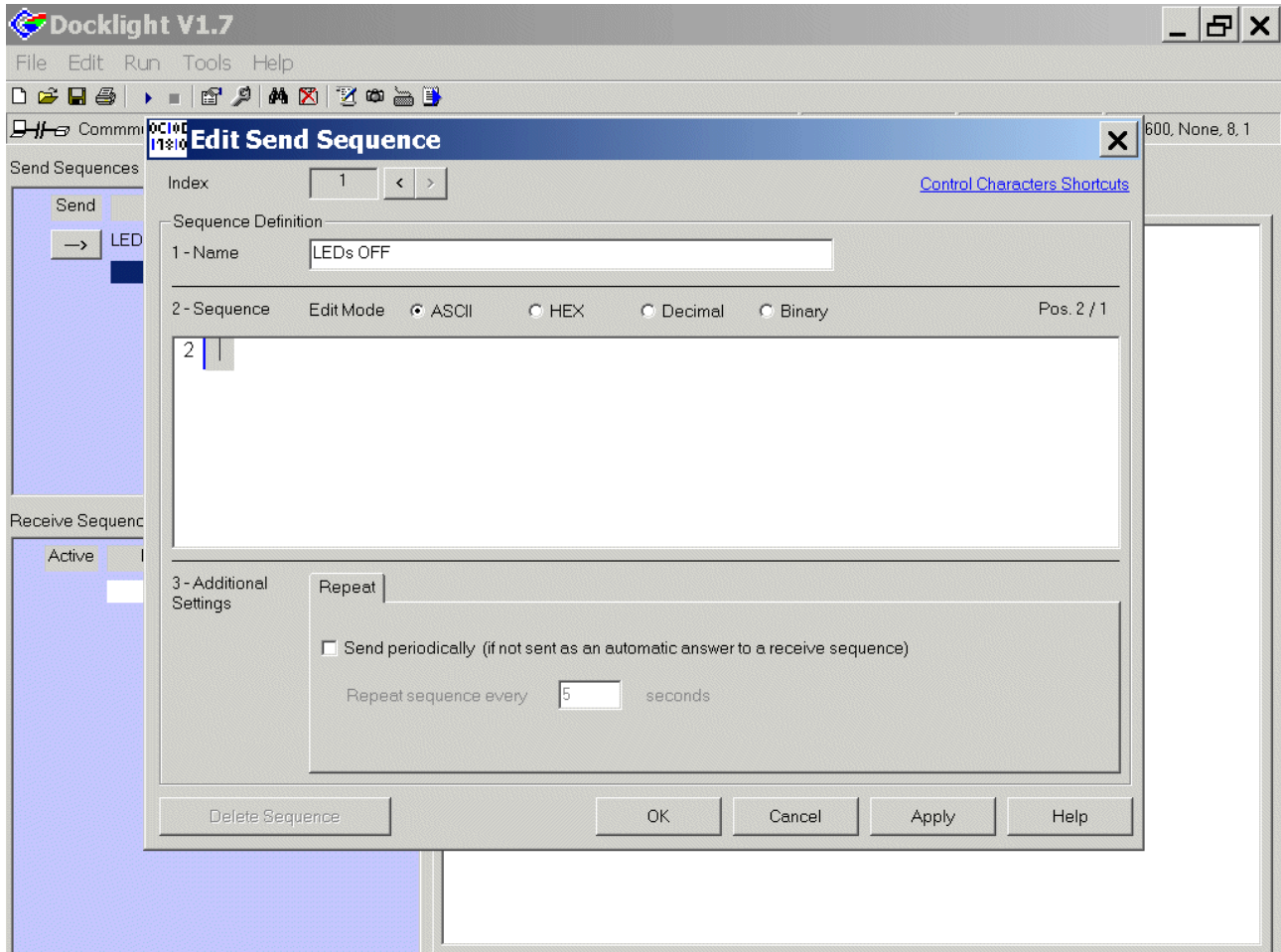
Double click inside the red box:





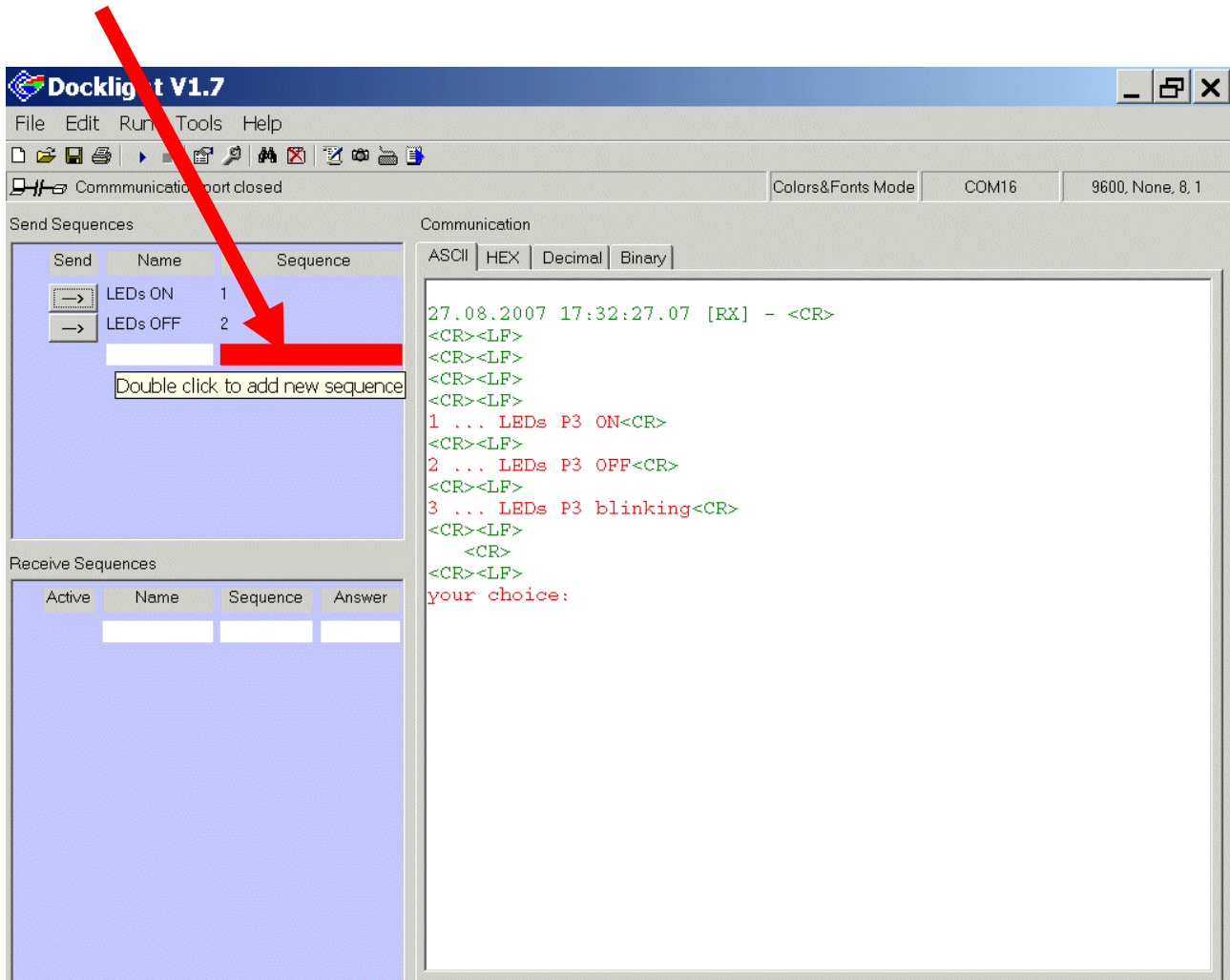
Edit Send Sequence: Sequence Definition: 1- Name: insert: LEDs OFF

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



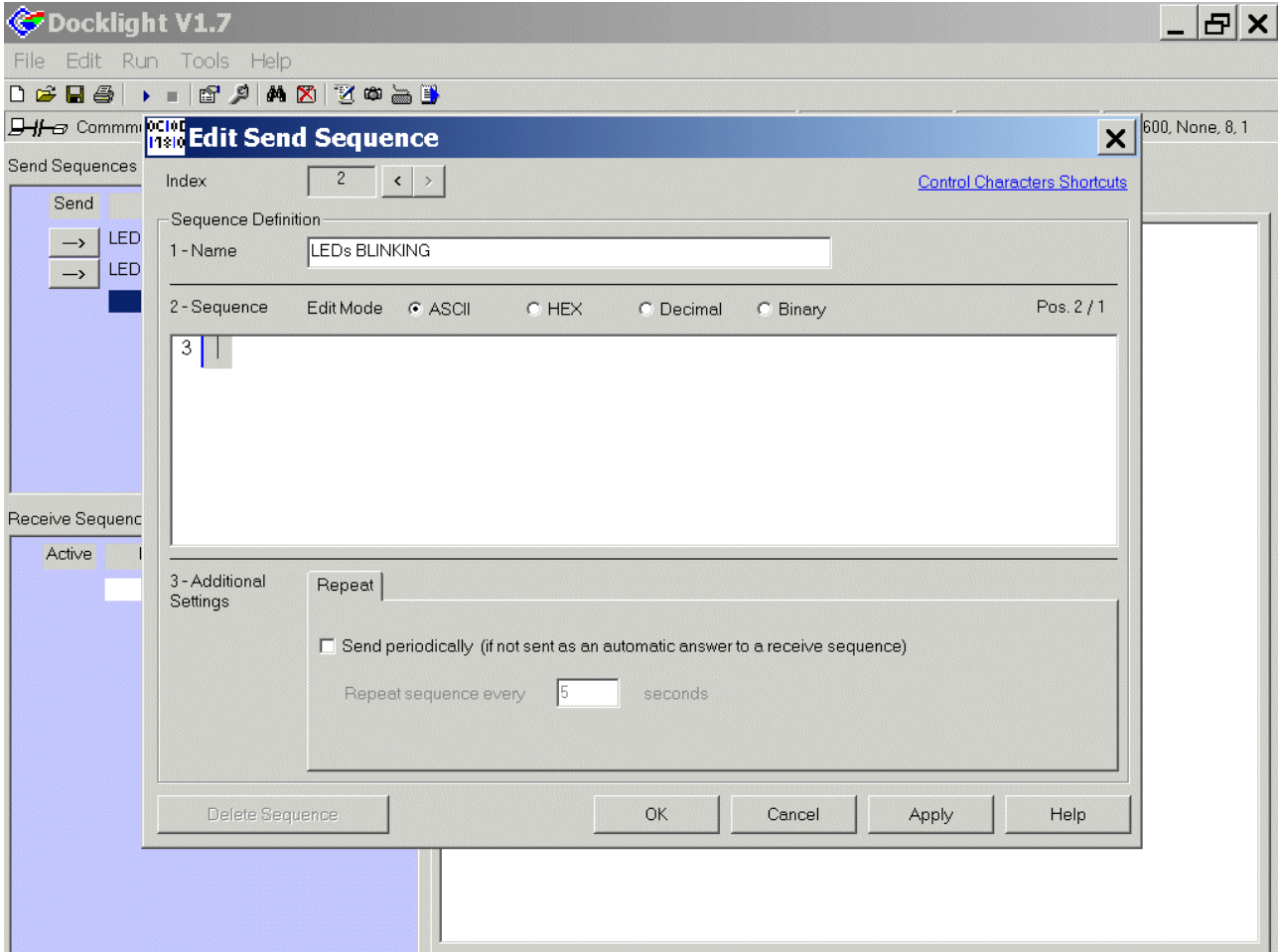
OK

Double click inside the red box:






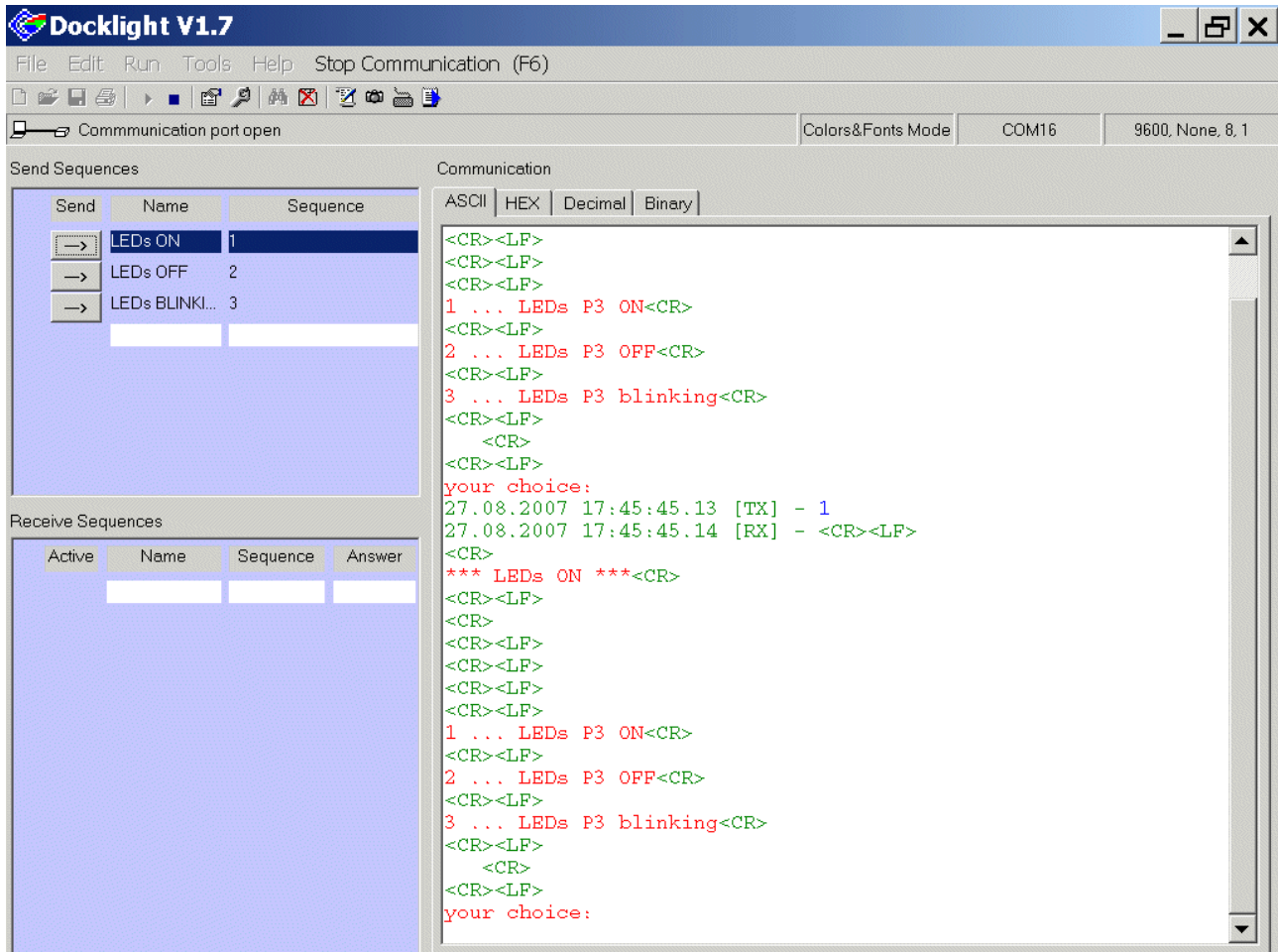
Edit Send Sequence: Sequence Definition: 1- Name: insert: LEDs BLINKING

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 the Keil tool chain.

Now you can easily expand our "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 XC88x microcontrollers!

**Note:**

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

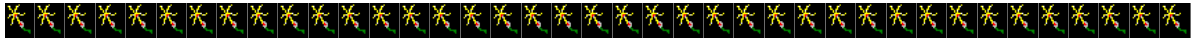
All these step-by-step books use the same microcontroller resources and the same example code.

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.

**7.) Feedback (XC888): Your opinion, suggestions and/or criticisms**



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

---

---

---

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

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

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



**Your suggestions:**

---

---

---

---

---

---

---

---

---

---

<http://www.infineon.com>

Published by Infineon Technologies AG