AP08073

XC888

XC800 USCALE start kit: "Cookery Book" for a hello world application using the KEIL tool chain

Microcontrollers

Never stop thinking

Edition 2008-07-16

Published by
Infineon Technologies AG
81726 München, Germany

© Infineon Technologies AG 2008.
All Rights Reserved.

LEGAL DISCLAIMER

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

Information

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

Warnings

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

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

AP08048		
Revision History:	2008-05	V2.0
Previous Version:	none	

Page	Subjects (major changes since last revision)

We Listen to Your Comments

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

mcdocu.comments@infineon.com

Table of Contents **Page**

Note: Table of Contents see page 9.

Introduction:

This “Appnote” is a Hands-On-Training / Cookery Book / step-by-step book.
It will help inexperienced users to get the XC800-USCALE-start-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 XC800-USCALE-start-kit.
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.

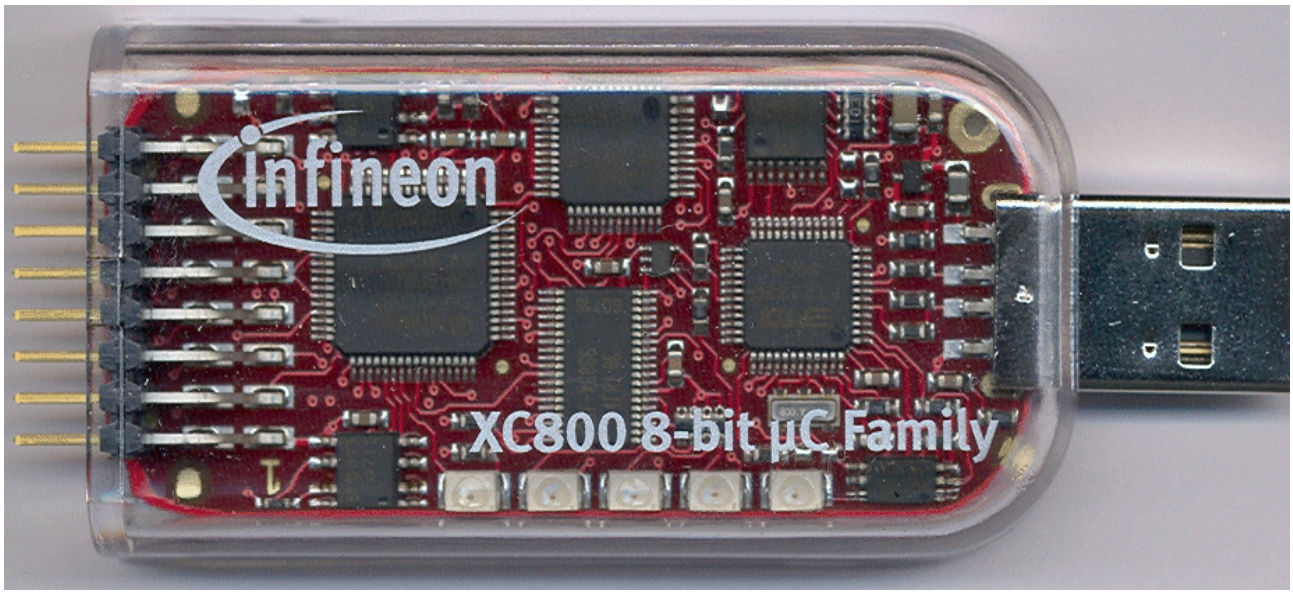
Note:
The style used in this document focuses on working through this material as fast and easily as possible. That means there are full screenshots instead of dialog-window-screenshots; extensive use of colours and page breaks; and listed source-code is not formatted to ease copy & paste.

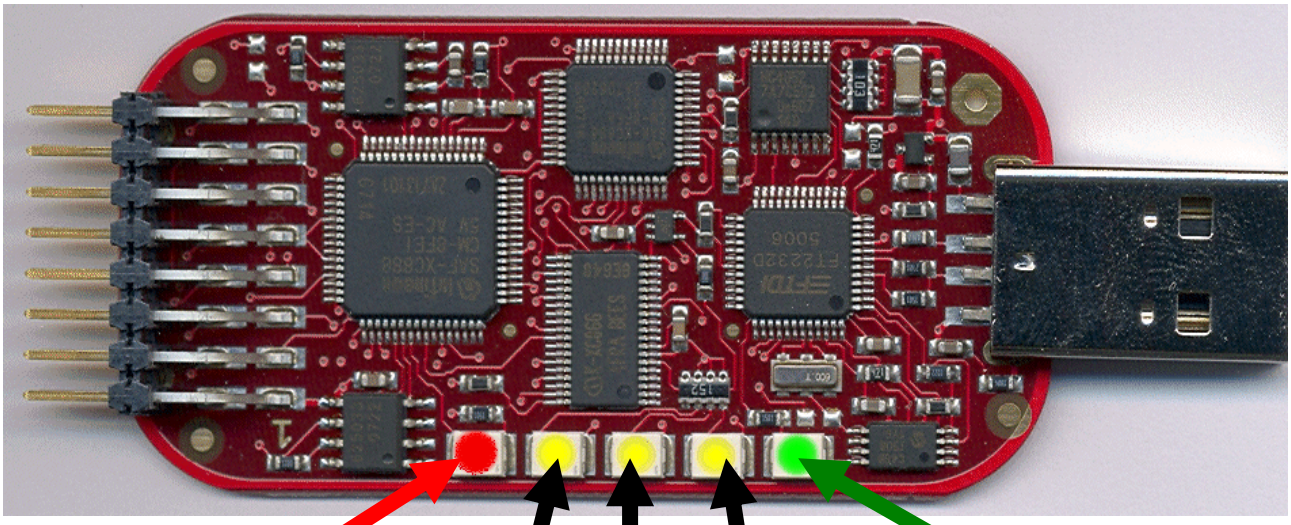
Have fun and enjoy the XC800-USCALE-start-kit!

Programming Example

XC800 USCALE start kit

Note:
The XC800 USCALE START KIT includes 3 XC800 (XC866, XC886, XC888) microcontrollers. Only one microcontroller can be active at any given time. The other 2 will remain in reset. Therefore the debugger must be permanently connected to the XC800 USCALE START KIT during operation to select the active microcontroller.
The active microcontroller will be indicated by one of the 3 yellow LEDs.
In this document we are going to use the XC888 microcontroller.





Port_1_Pin_5 (User LED red)

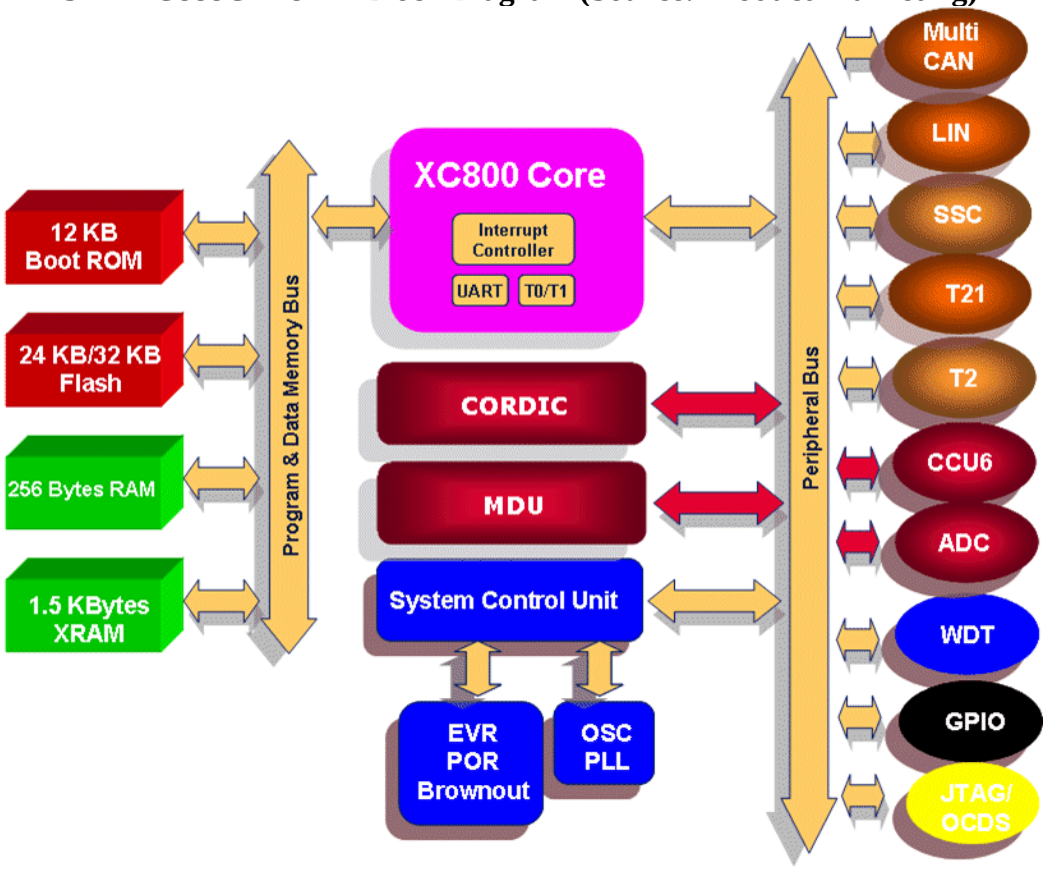
XC866 **XC886**

XC888: when the XC888 is selected by the debugger the LED is lit.

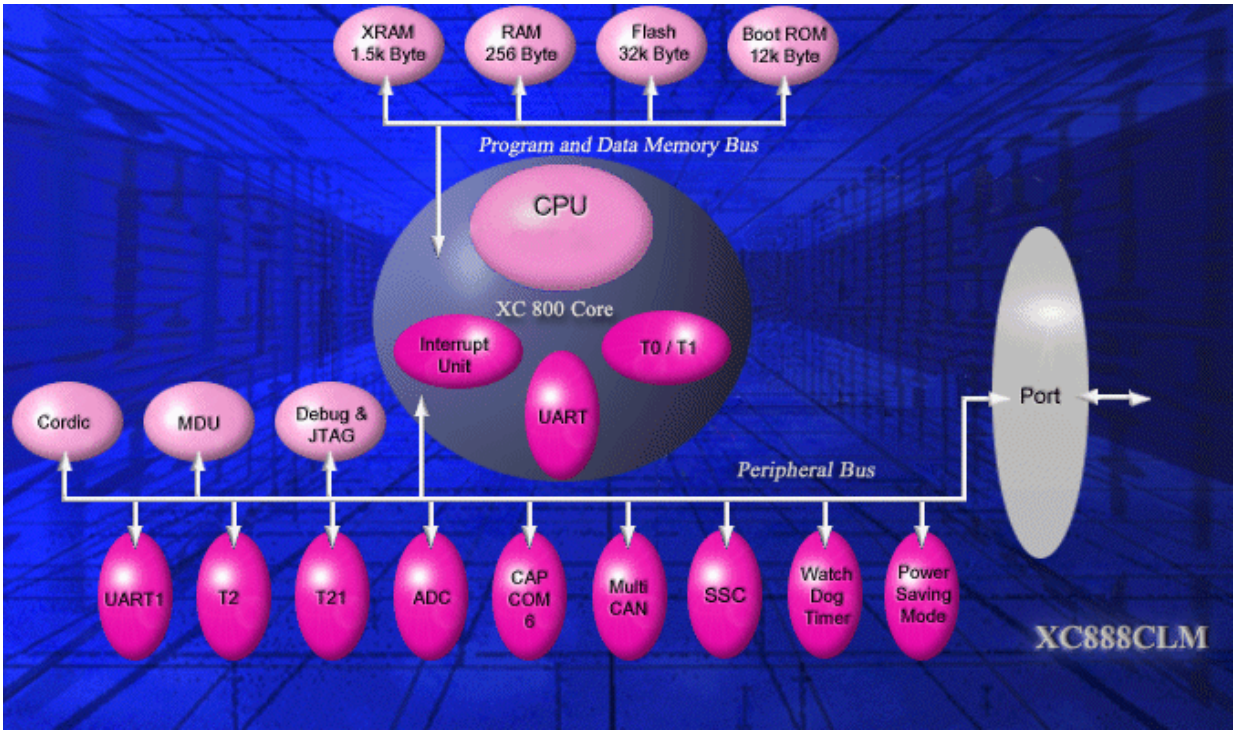
Run (LED green)

Used/selected microcontroller:

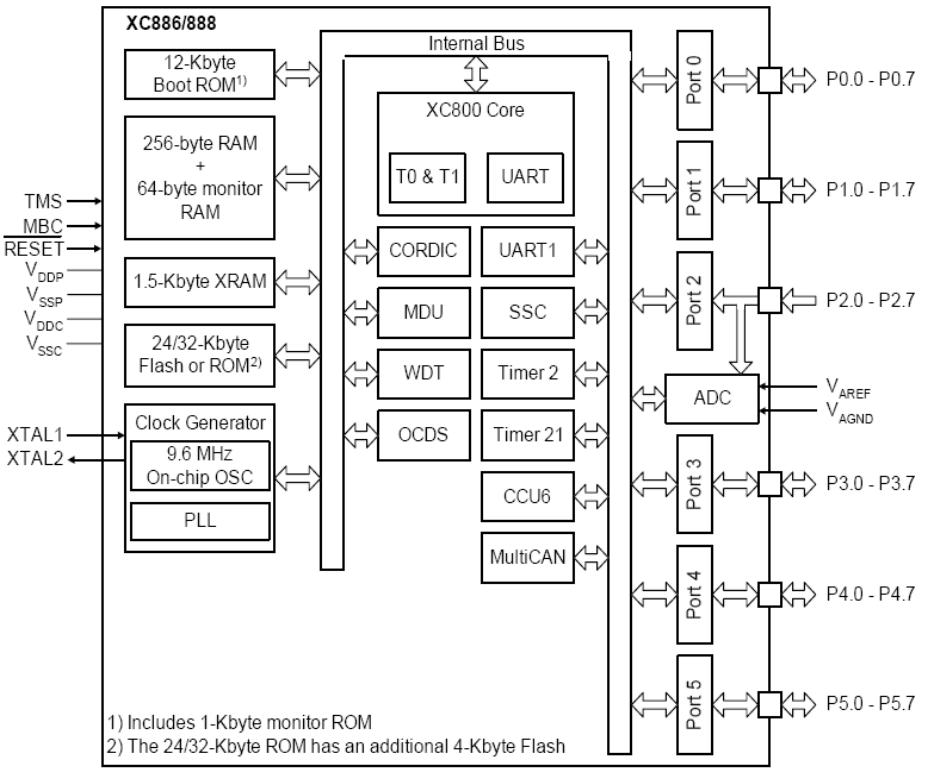
SAF XC888CM-8FFI Block Diagram (Source: Product Marketing)



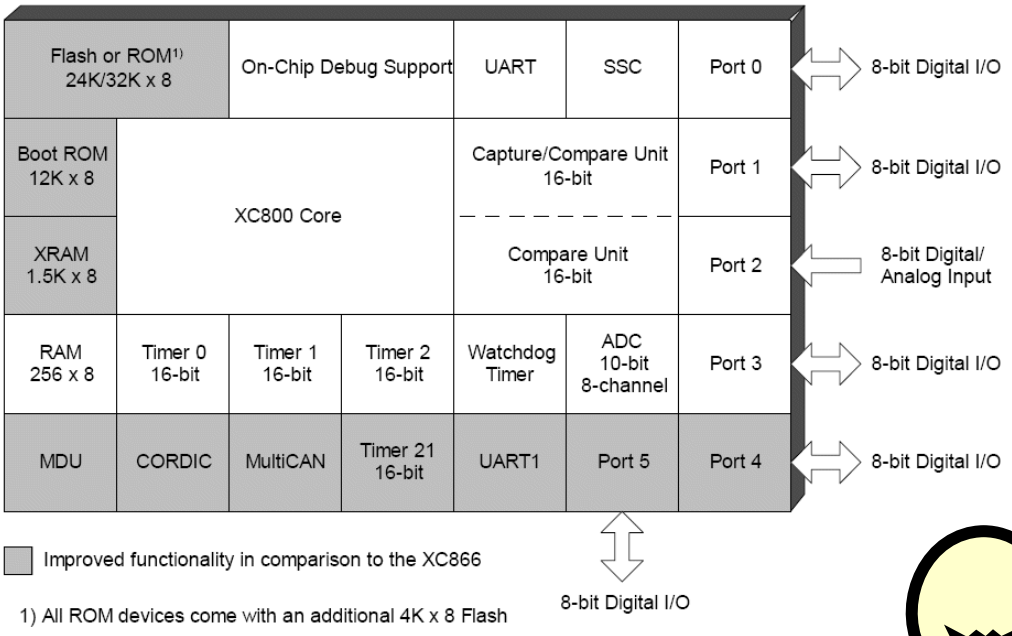
XC888CM Block Diagram (Source: DAvE)



XC888CM Block Diagram (Source: User's Manual)



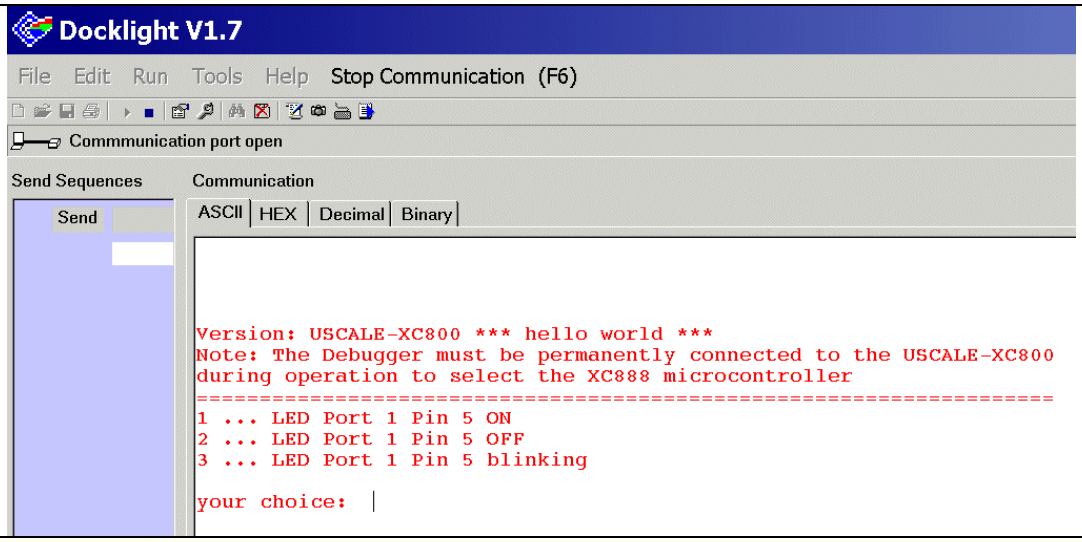
XC888CM functional units (Source: User's Manual)



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 XC800 USCALE start kit:

Your program:	
Chapter/ Step	***** Recipes *****
1.)	**DAS Installation + Connecting the XC800 USCALE start kit**
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 the KEIL Development Tools (C-Compiler)** **Programming of your application (XC888) with the KEIL tool chain (µVision3)** **Compiler V8.12c + first steps with the Simulator**
5.)	**Using the simulator**
6.)	**Using real hardware (+ OnChipFlash-Programming)**

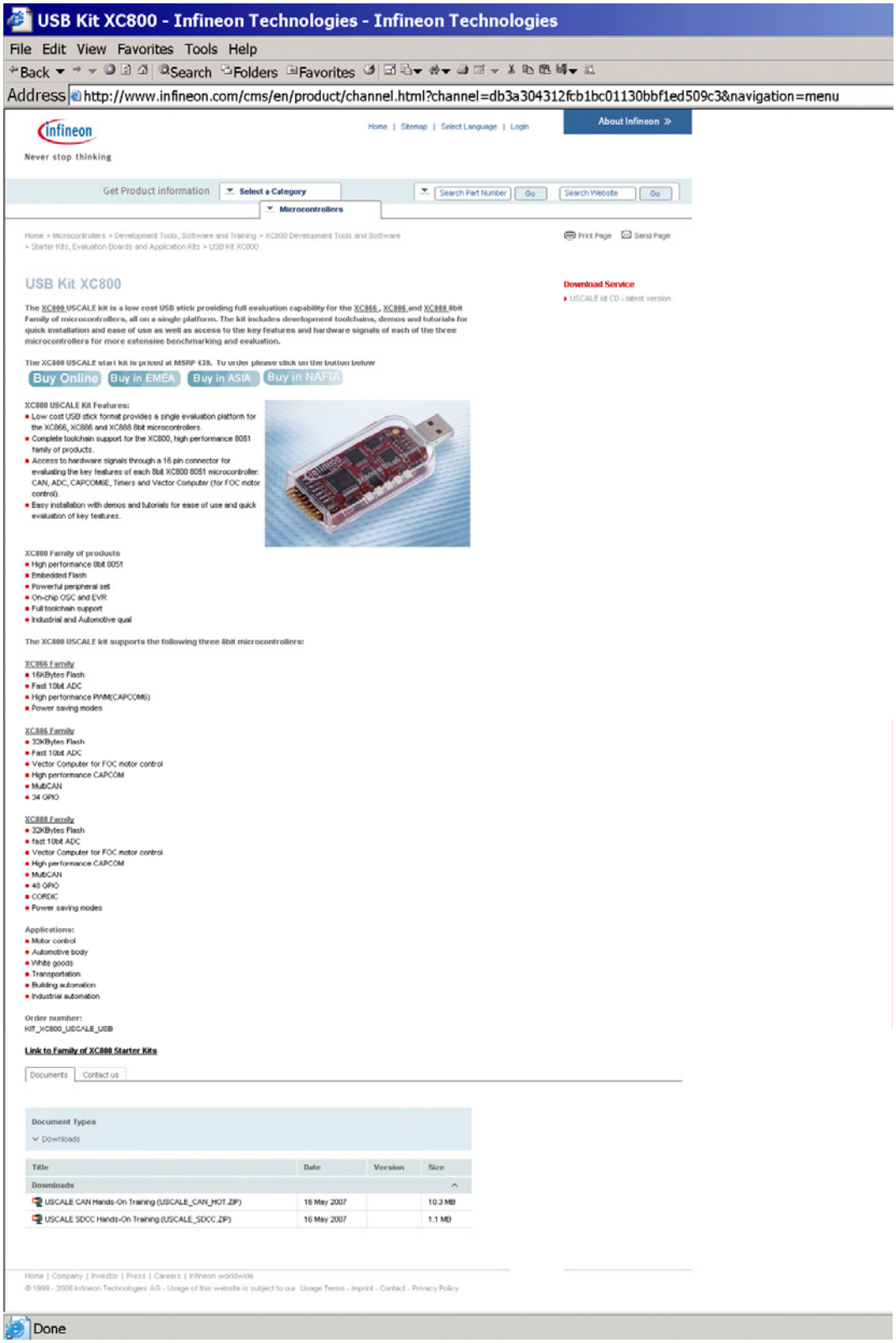
Feedback

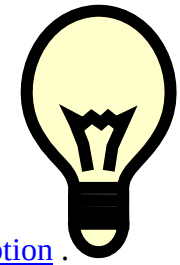
7.)	**Feedback**

1.) DAS Installation + Connecting the XC800 USCALE start kit:

Screenshot of the XC800 USCALE start kit homepage:

http://www.infineon.com/cms/en/product/channel.html?channel=db3a304312fcb1bc01130bbf1ed509c3&navigation=menu





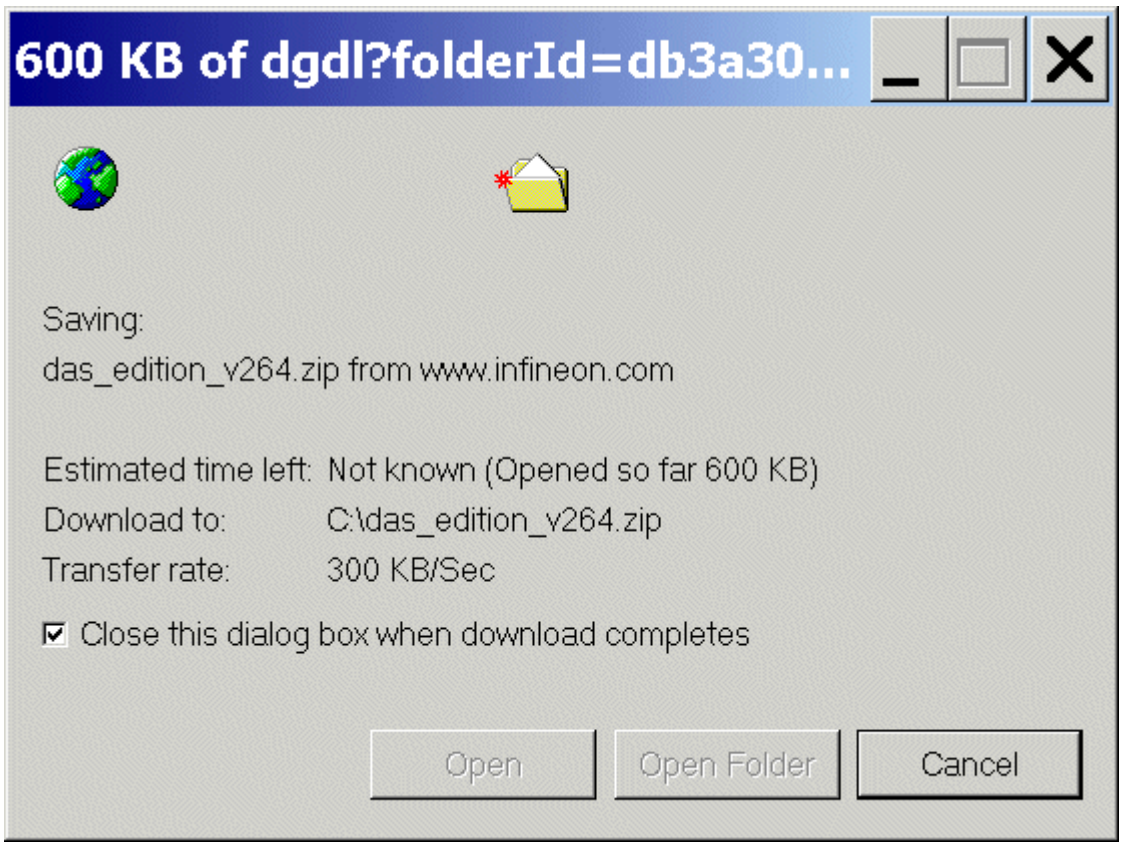
Note:
For further information, please refer to the USCALE-XC800 System Description .

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

Go to www.infineon.com/DAS:

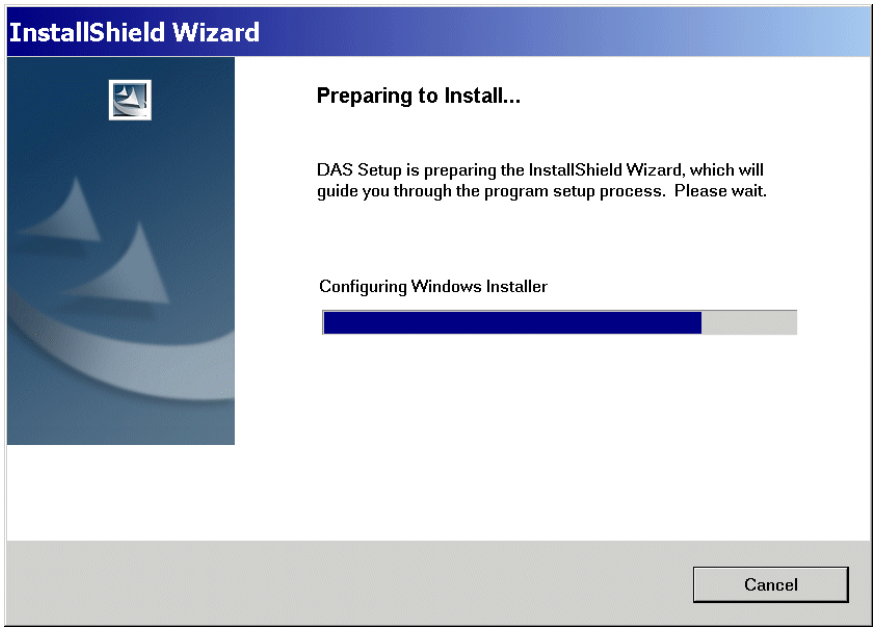
Note:
The DAS Server must be installed on your host computer!
The goal of the DAS software is to provide one single interface for all types of tools.
The USB Device driver communicates with the XC800 USCALE start kit when connected to the host computer.
The USB Device driver for the XC800 USCALE start kit USB interface is included in the DAS software.
A virtual COM port driver is also included.

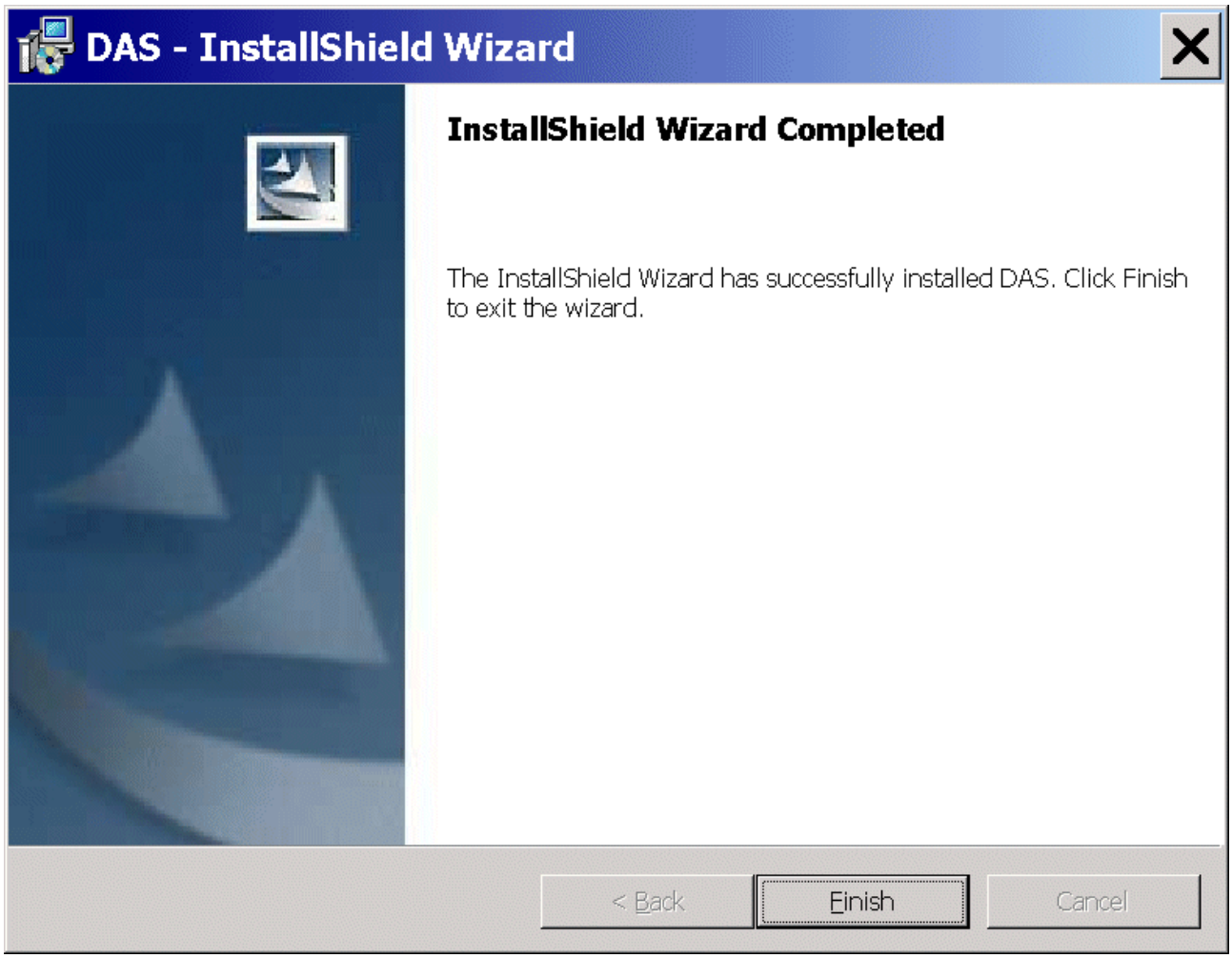
Download “The latest release version for download: DAS Edition v2.6.4”:



Unzip das_edition_v264.zip and

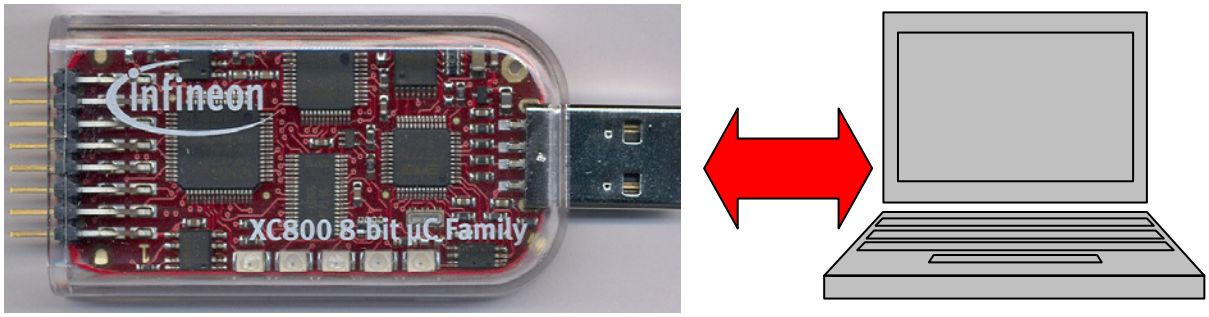
execute “DAS_v264_setup.exe” to install the DAS Server.



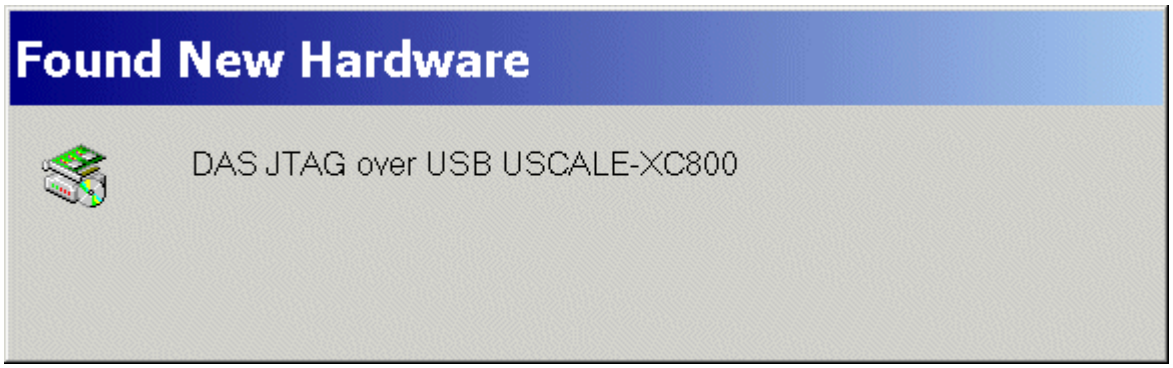


Click Finish

Connect the XC800 USCALE start kit to the host computer:



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

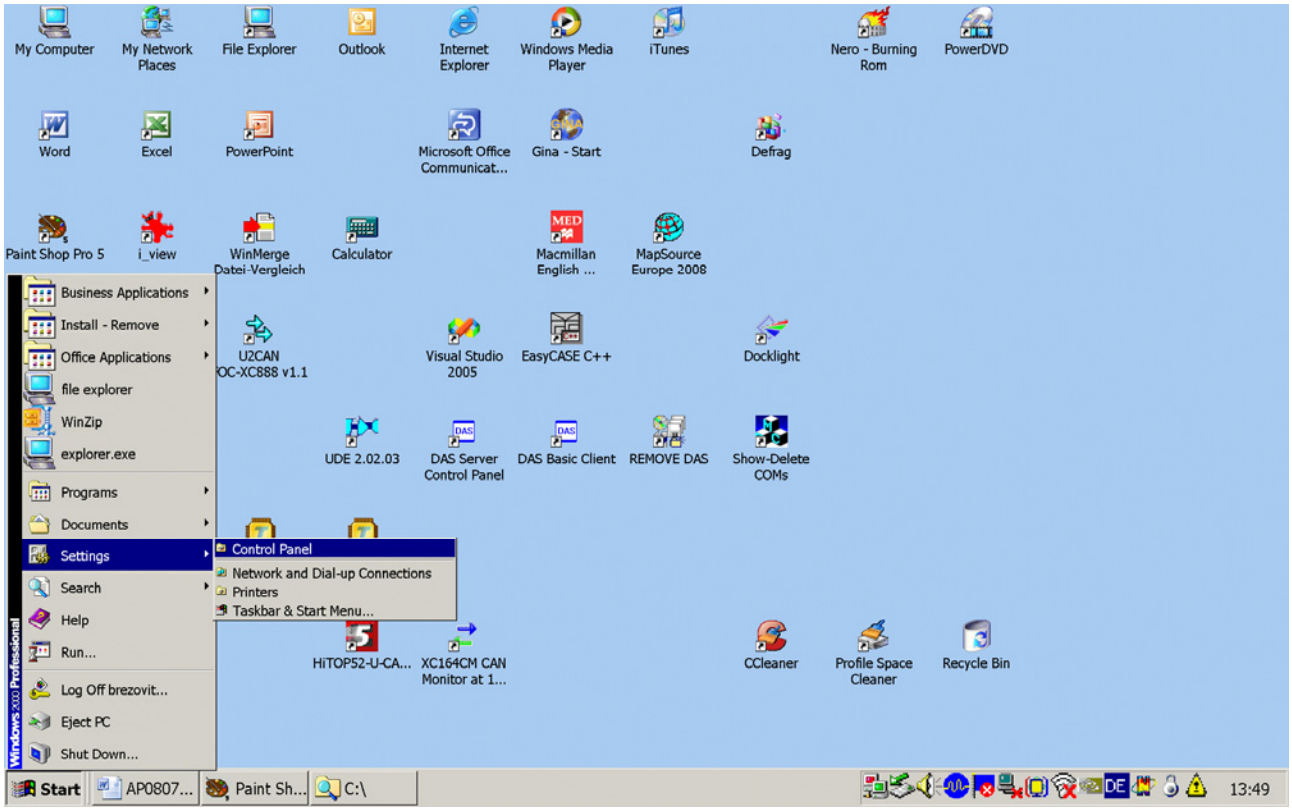


Note:
A USB driver is installed the first time while connecting the XC800 USCALE start kit via USB to your host computer.

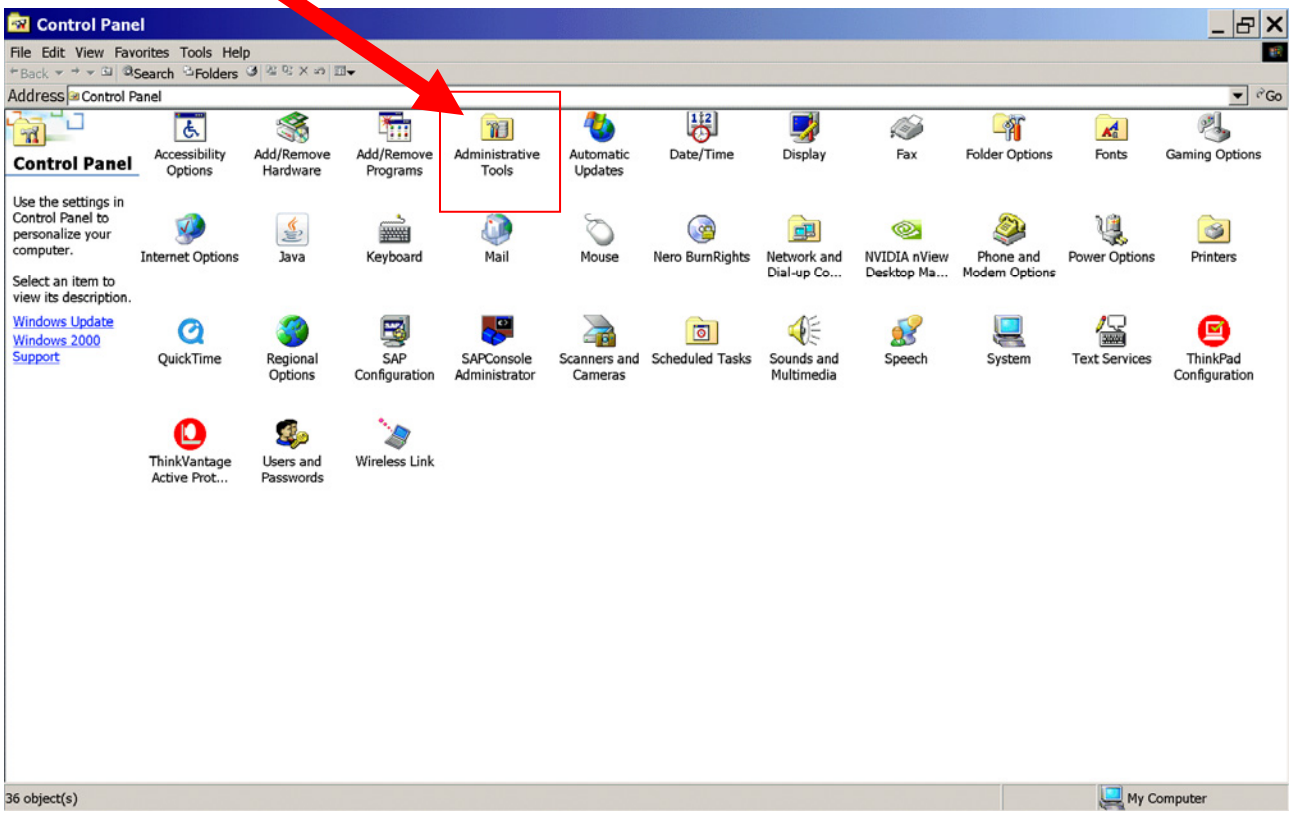
Note:
A default virtual COM Port is generated.

Using a Windows 2000 operating system, we are now going to search for the virtual COM Port which was generated after connecting our XC800 USCALE start kit:

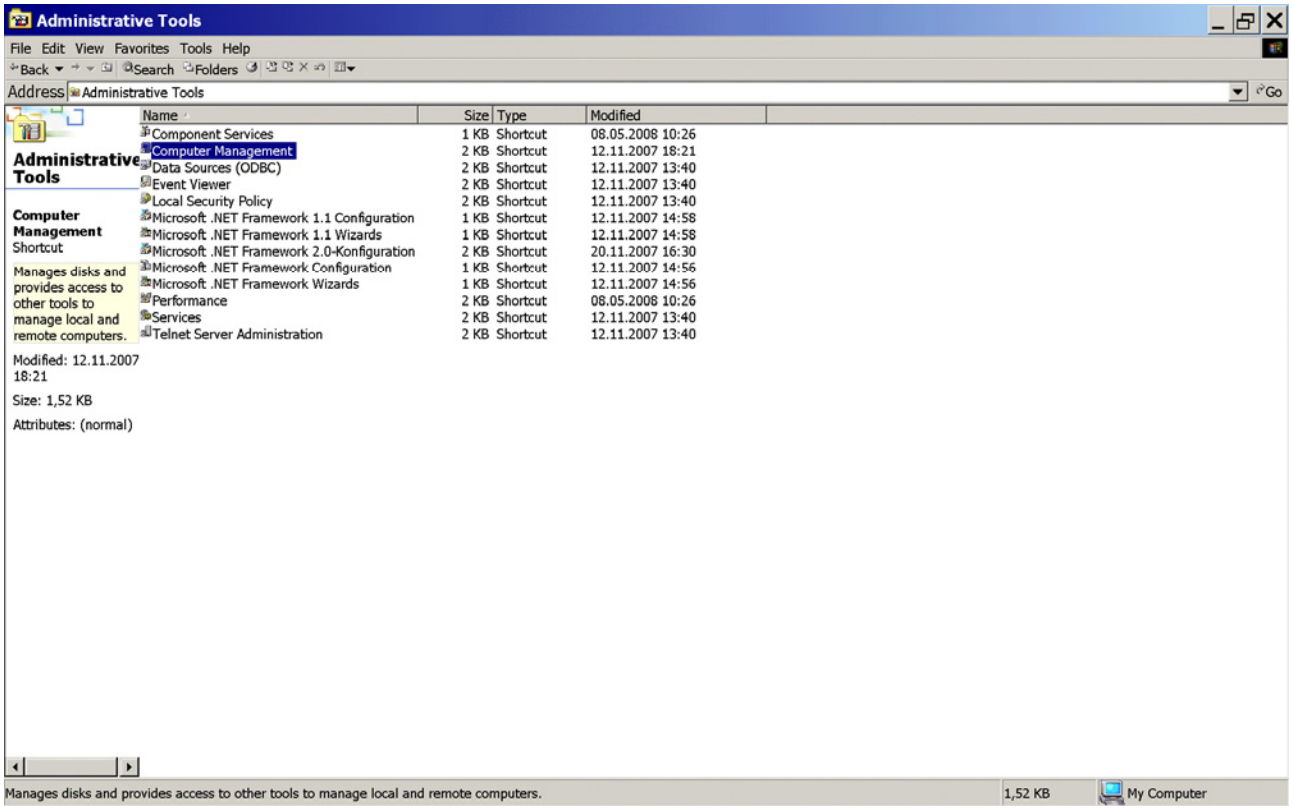
Start – Settings – Control Panel



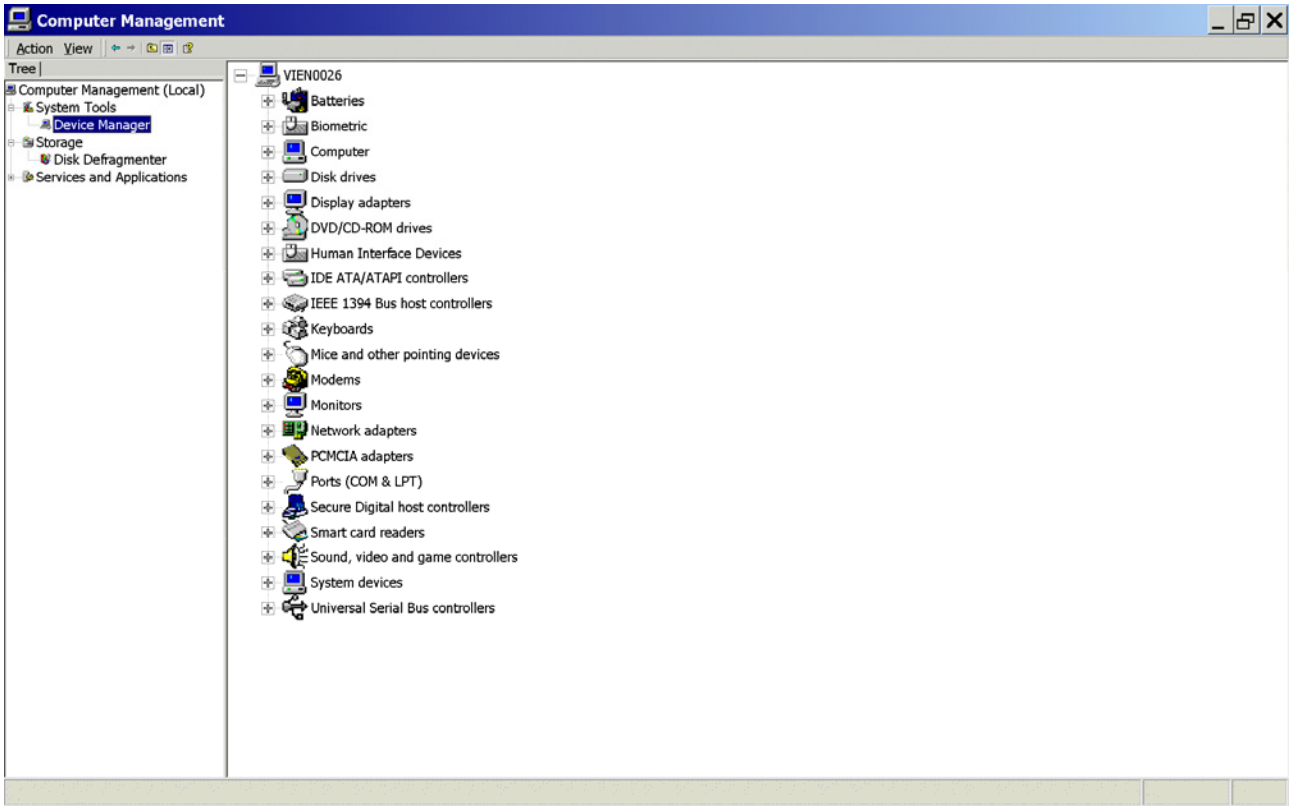
Double click: Administrative Tools



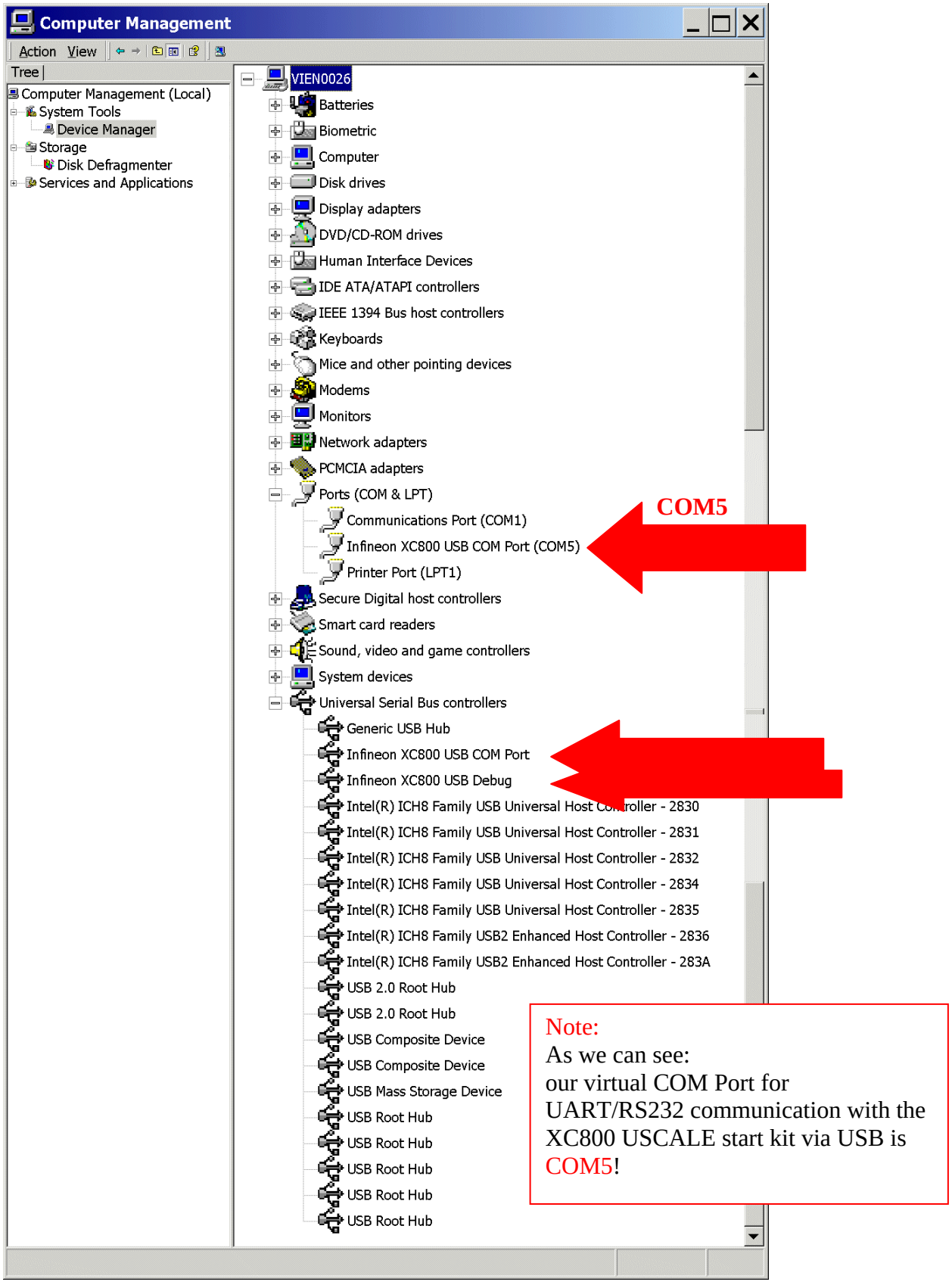
Double click: Computer Management



Click: Device Manager



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



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 - latest version	05 Feb 2007	V2.1 r24	14.8 MB
DAvE - Mothersystem	04 Jul 2006	V2.1 r23	15.1 MB

and execute setup.exe to install DAvE .

Note:
Abort the installation of Acrobat Reader.

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

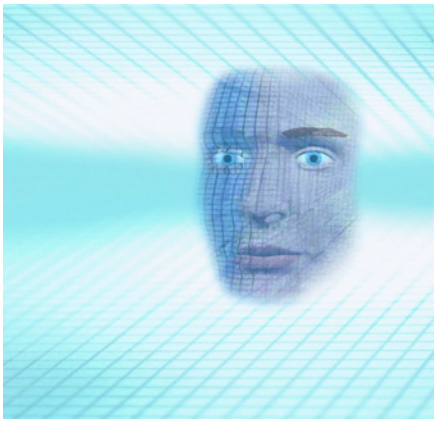
1.)
Download the DAvE-update-file (.DIP) for the required microcontroller
@ http://www.infineon.com/DAvE

Title	Date	Version	Size
Development Tools			
XC888CLM DIP file for DAvE (Microcontroller Configuration Tool)-latest version (XC888CLM_v1.4.zip)	08 Feb 2008	v1.4	7.5 MB

Unzip the zip-file "XC888CLM_v1[1].4.zip" and save " XC888CLM.DIP "
@ e.g. D:\DAvE\XC888-2008-05-26\XC888CLM.dip.

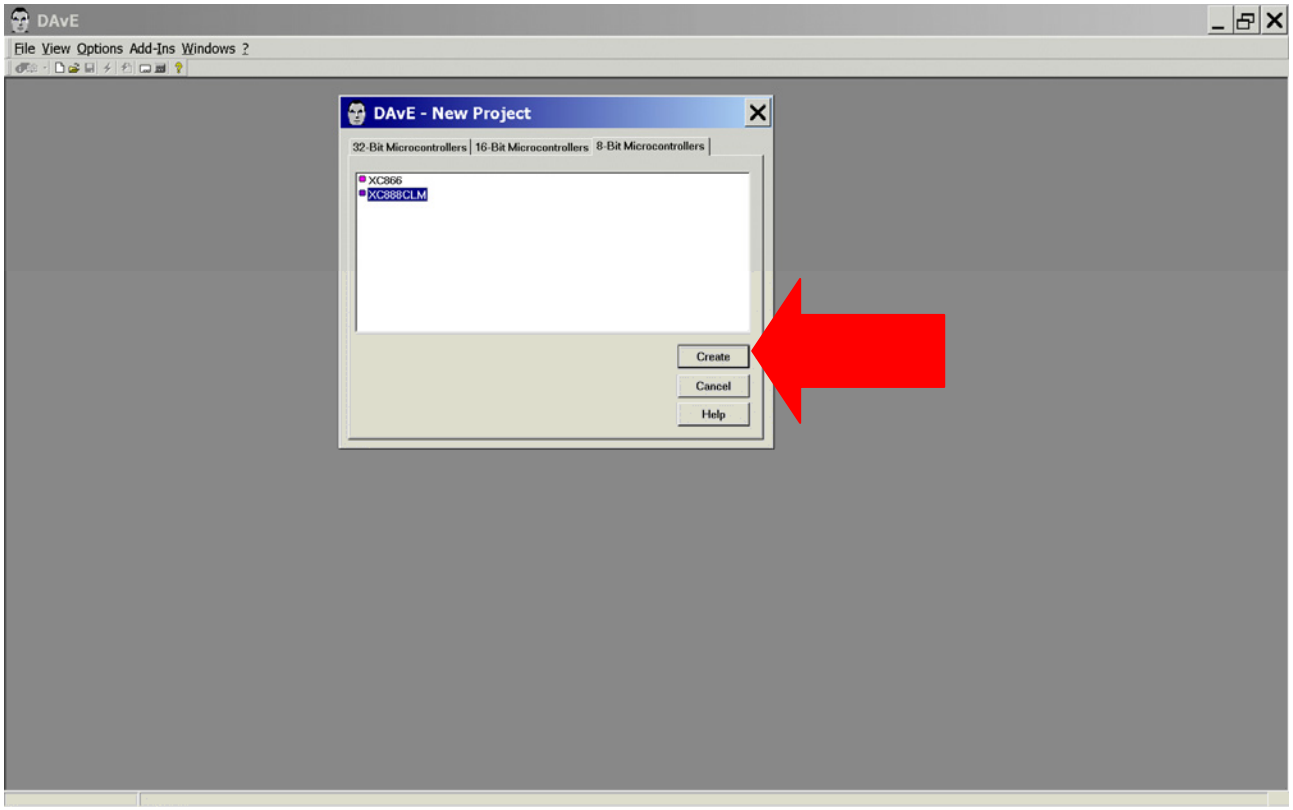
2.) Start DAvE - (click )
3.) View Setup Wizard Default: • Installation Forward> Select: • I want to install products from the DAvE's web site Forward> Select: D:\DAvE\XC888-2008-05-26 Forward> Select: Available Products click ✔ XC888CLM Forward> Install End
4.) DAvE is now ready to generate code for the XC888CLM microcontroller.

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



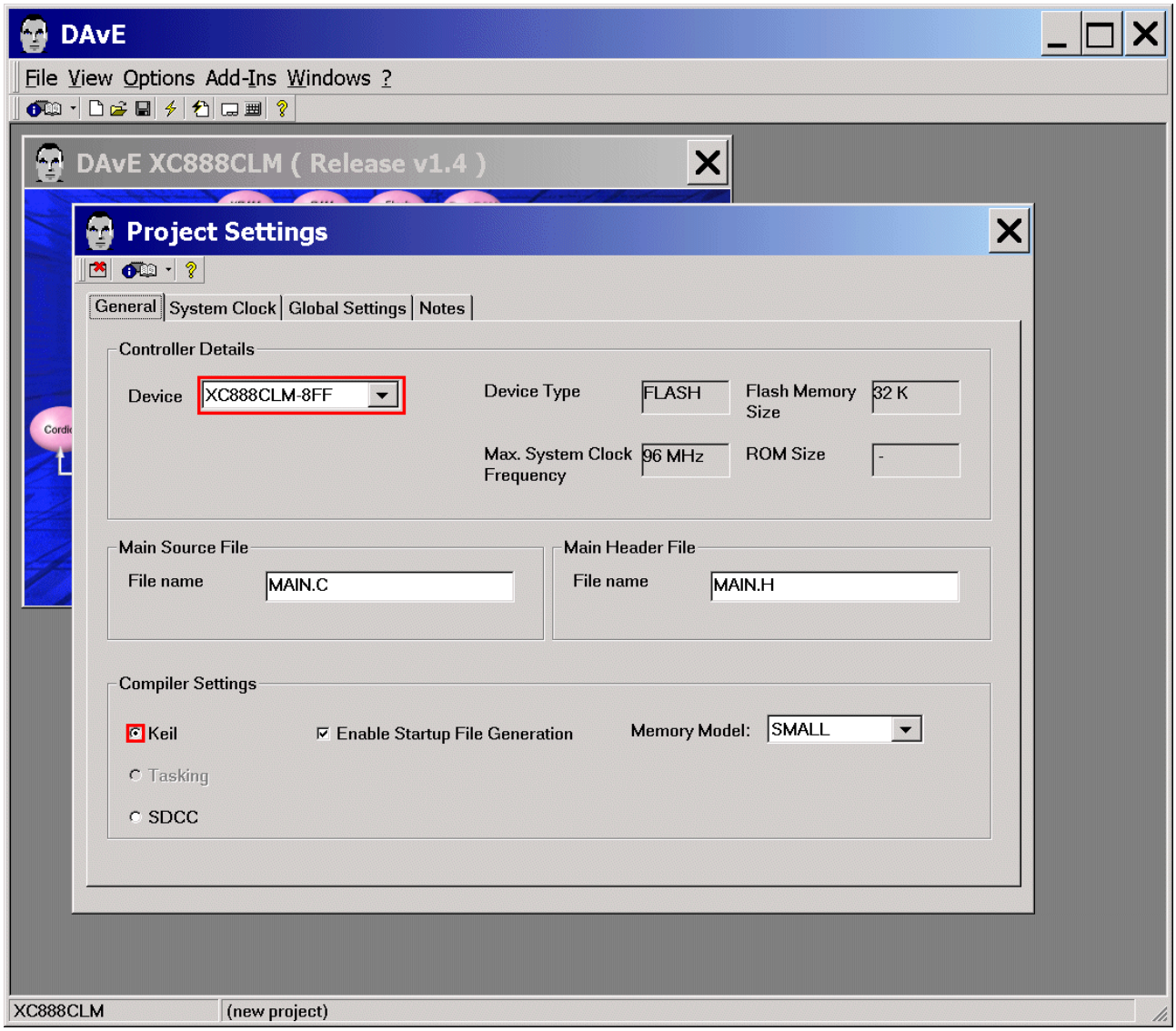
Start the program generator DAvE and select the XC888CLM microcontroller:

File
New
8-Bit Microcontrollers
select XC888CLM
Create

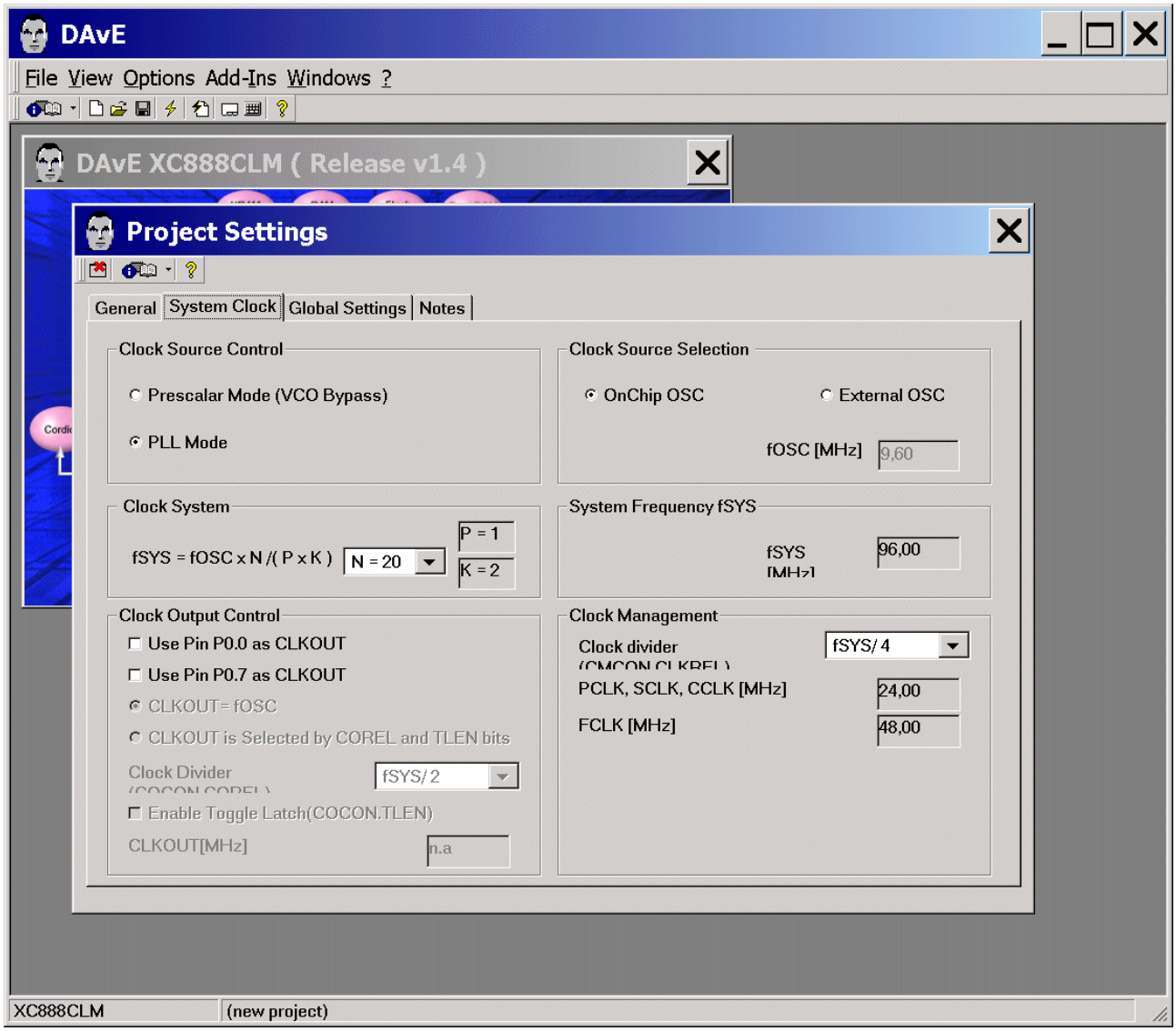


Choose the Project Settings as you can see in the following screenshots:

General: Controller Details: Device: check/select XC888CLM-8FF
General: For the KEIL Compiler check/choose ⊙ Keil in the Compiler Settings:



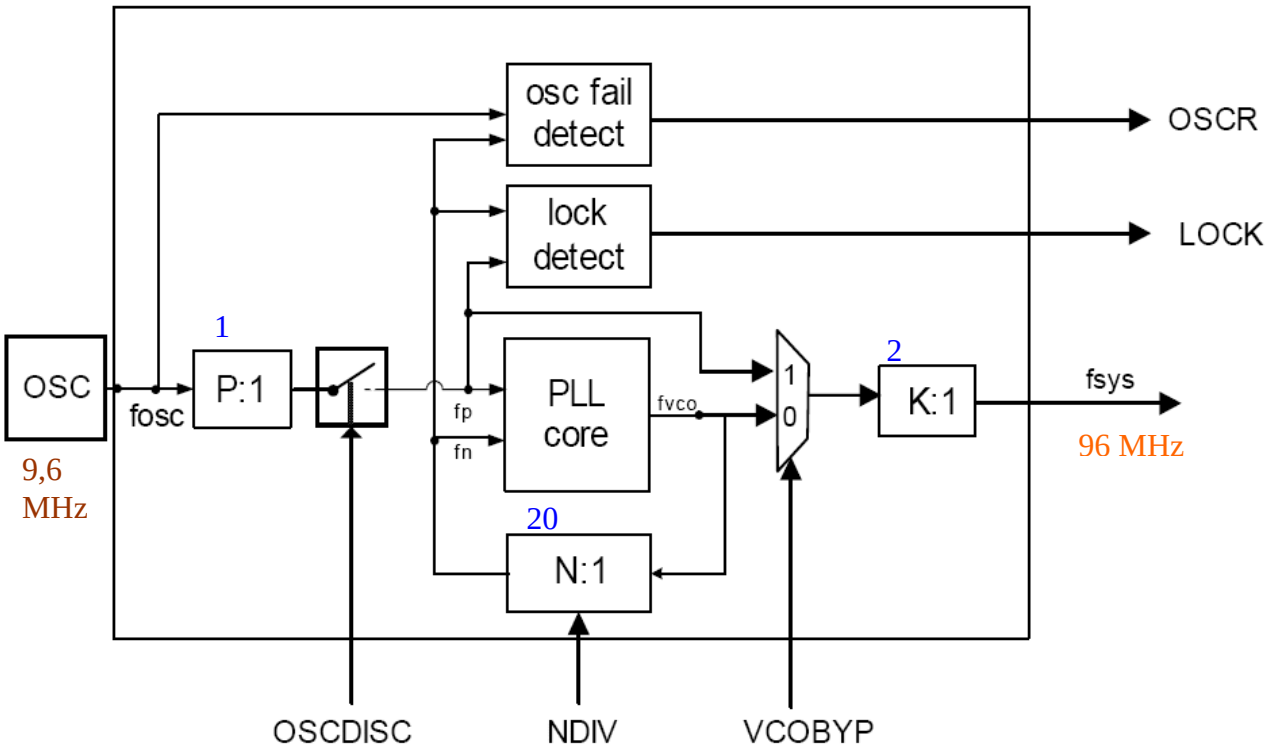
System Clock: (do nothing)



Note:
CPU clock is 24 MHz.

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

Clock Generation Unit (CGU) Block Diagram

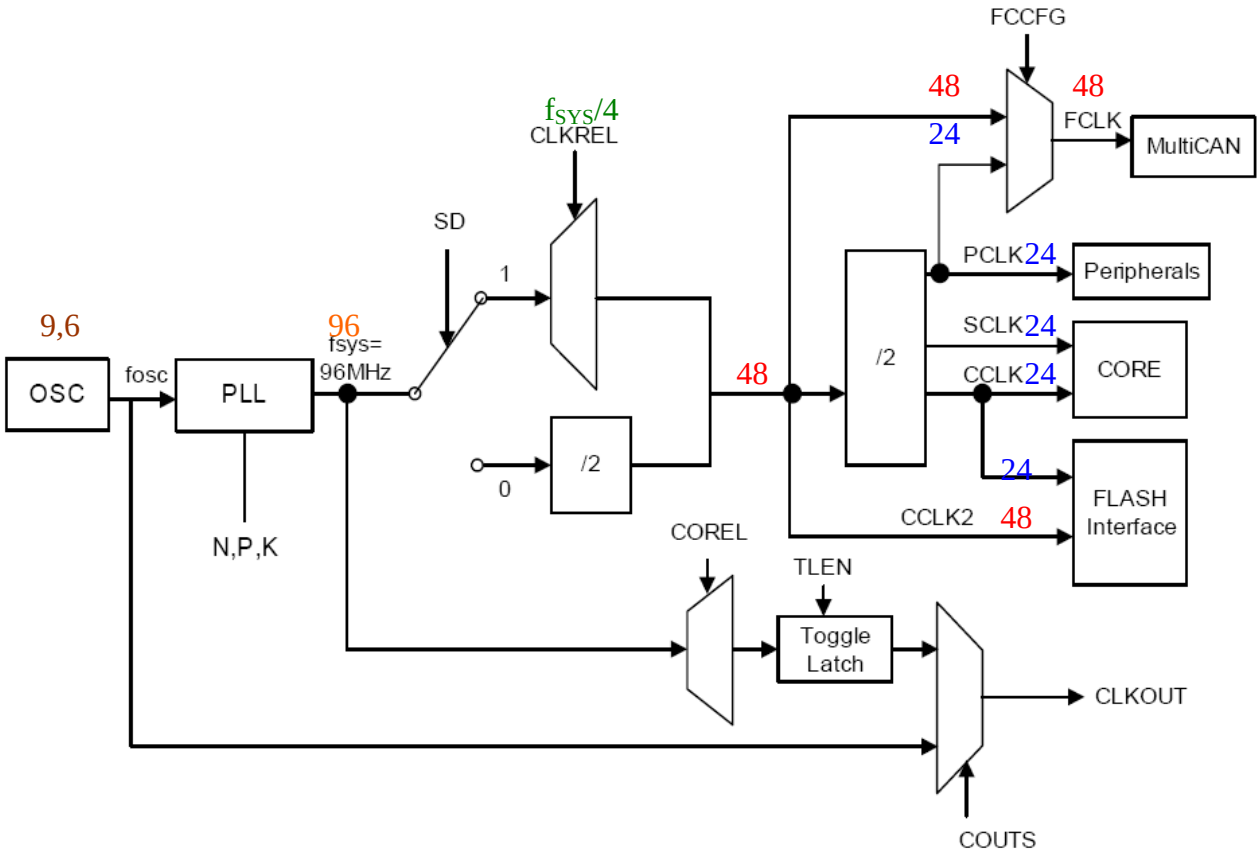


Note:

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

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

Clock Generation from fsys:



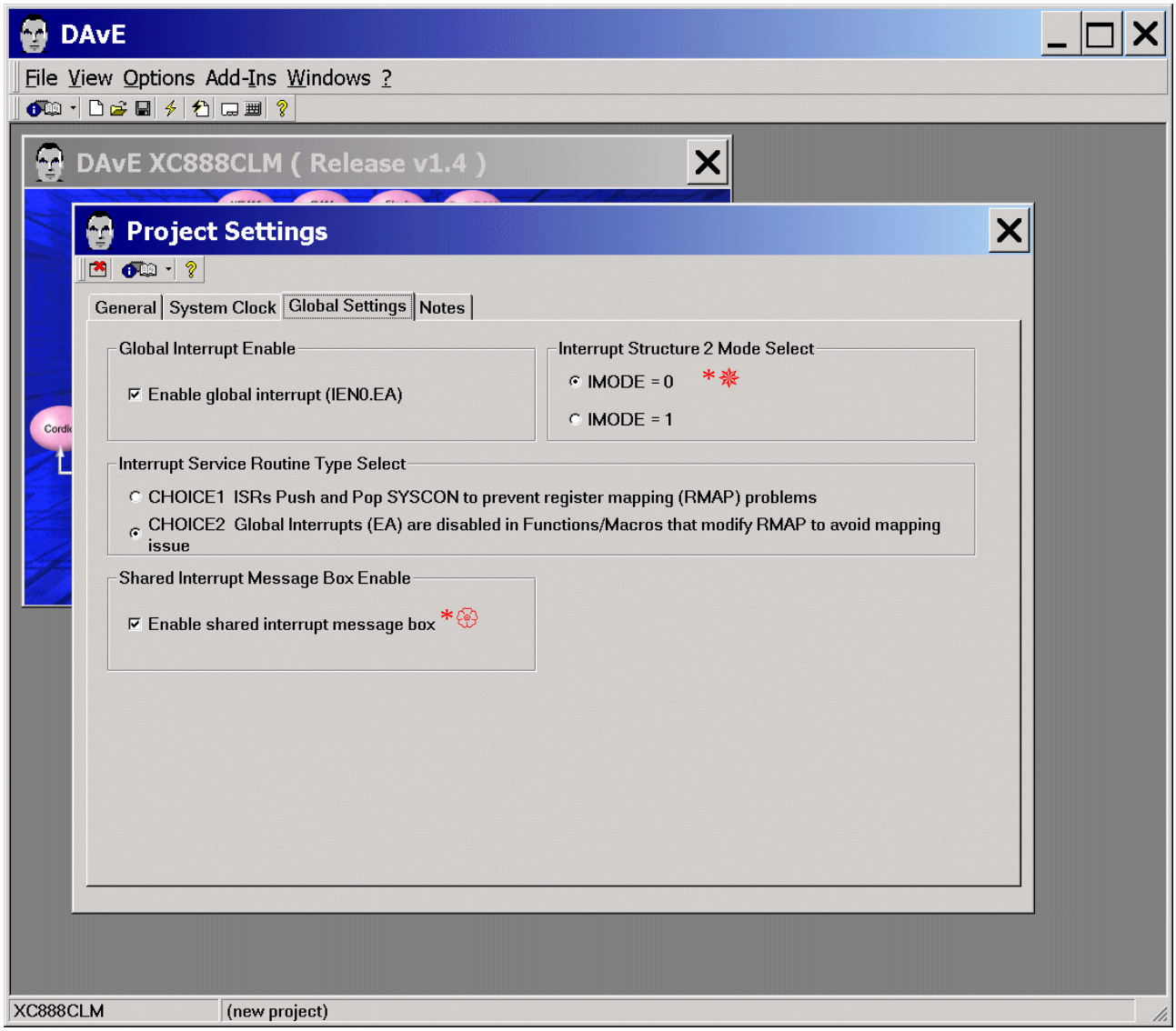
Note:

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 28) is inclusive the fixed divider factor of 2.

Global Settings: (do not change configuration)



Note (Source: DAvE):

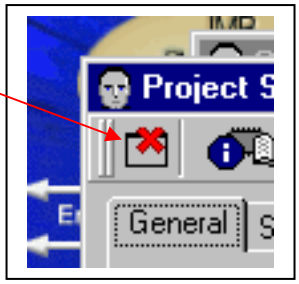
```
//  You have two choices for Interrupt Service Routine Type Select.
//  If you select CHOICE 1 then ISR will be generated with push and pop.
//  If you select CHOICE 2 then ISR will be generated without push and pop.
//  Default choice is CHOICE 2.
//  Current selection is CHOICE 2
```

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.

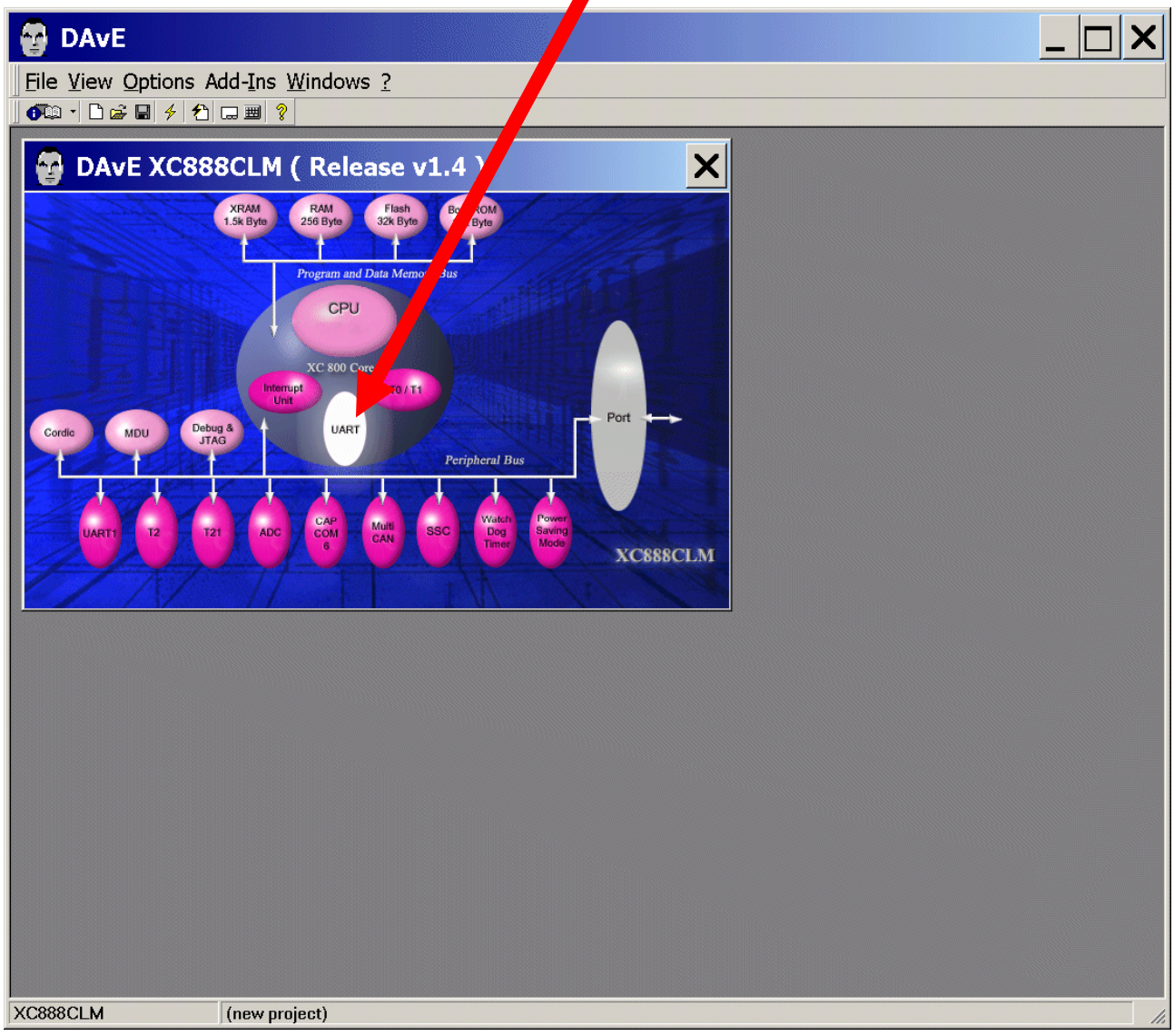
Notes: If you wish, you can insert your comments here.

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

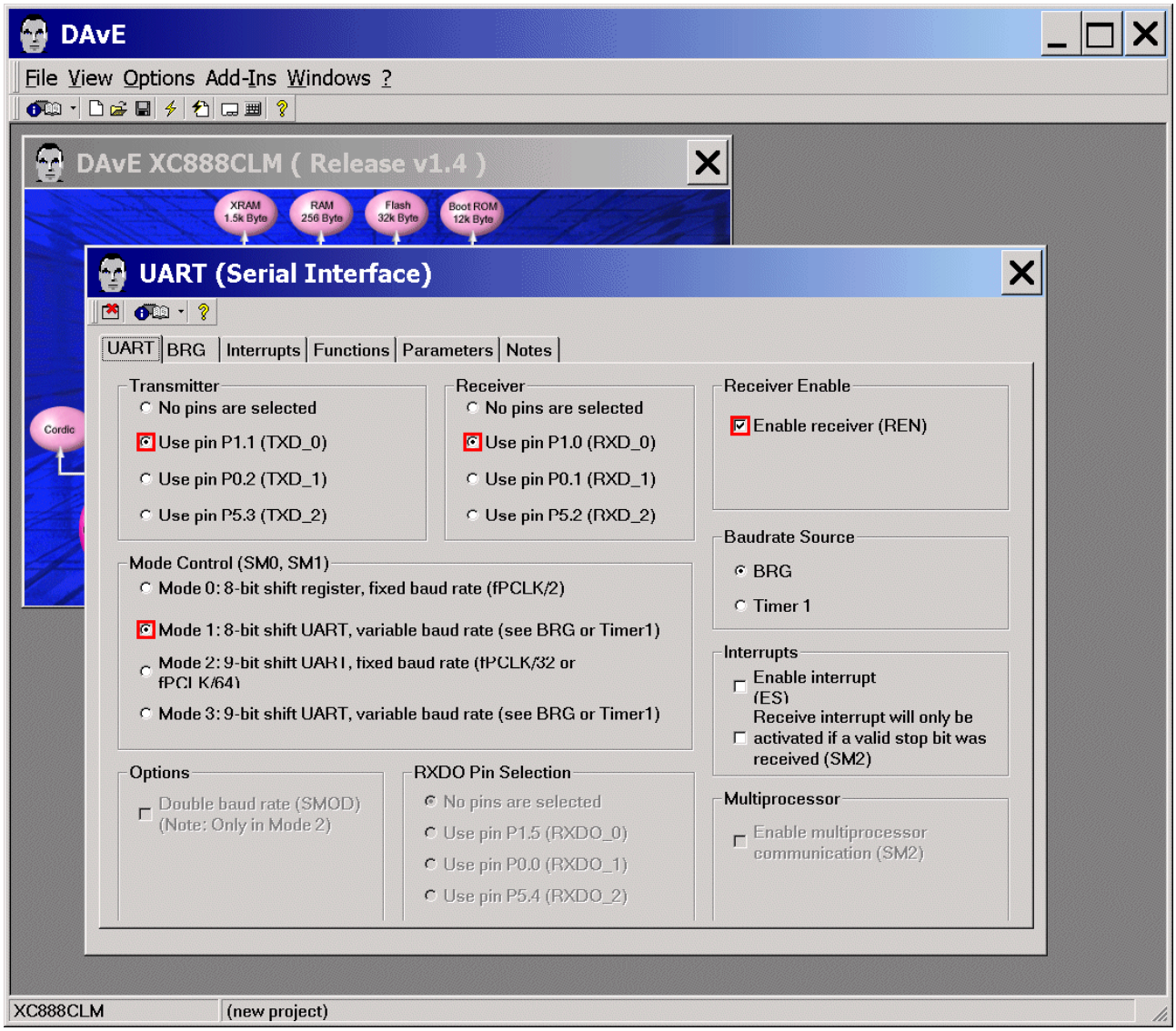


Configuration of the UART:

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

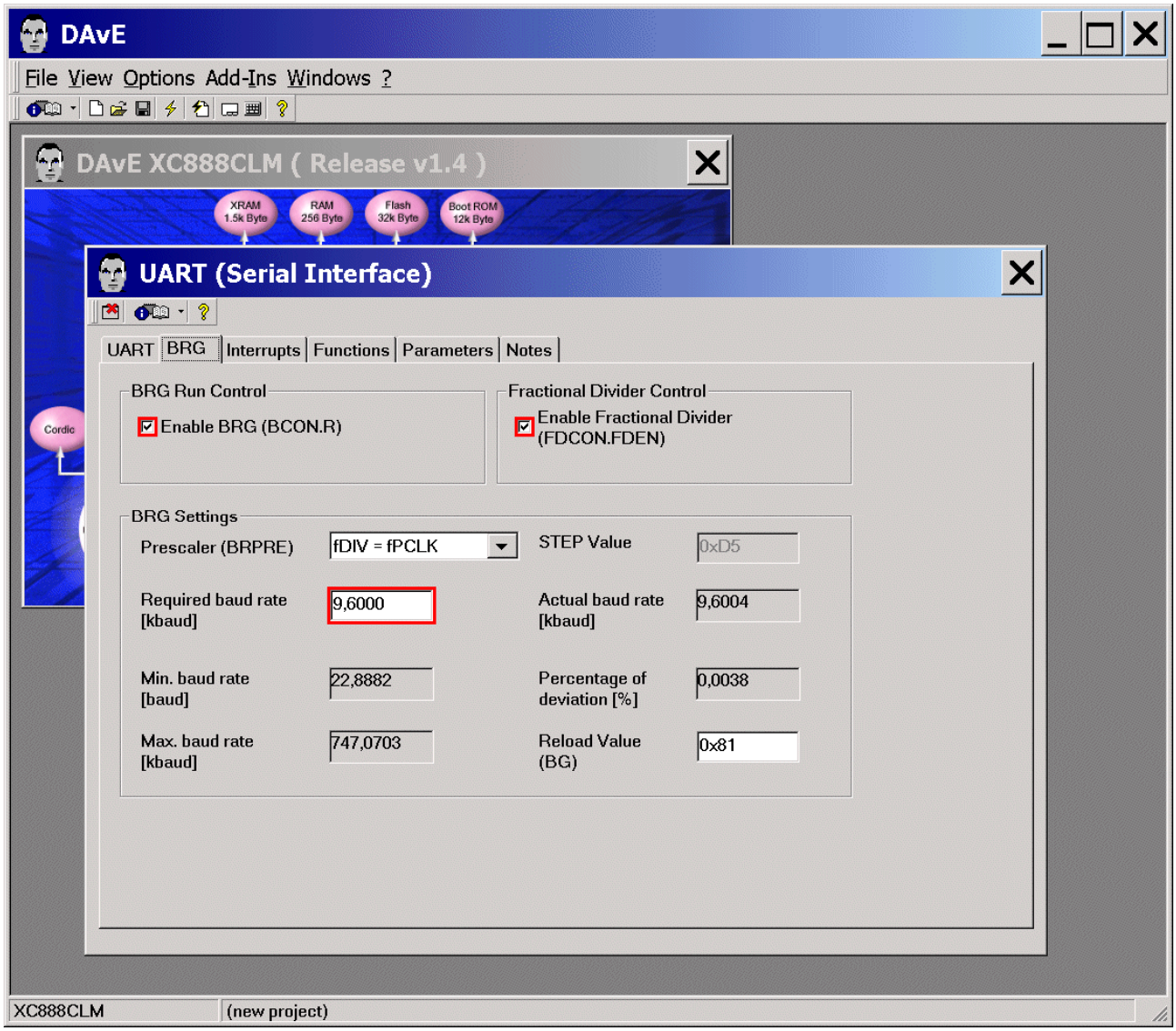


UART: Transmitter: click ⊙ Use pin P1.1 (TXD_0)
UART: Receiver: click ⊙ Use pin P1.0 (RXD_0)
UART: Receiver Enable: click ✓ Enable receiver (REN)
UART: Mode Control: click ⊙ Mode 1: 8-bit shift UART, variable baud rate (see BRG or Timer1)



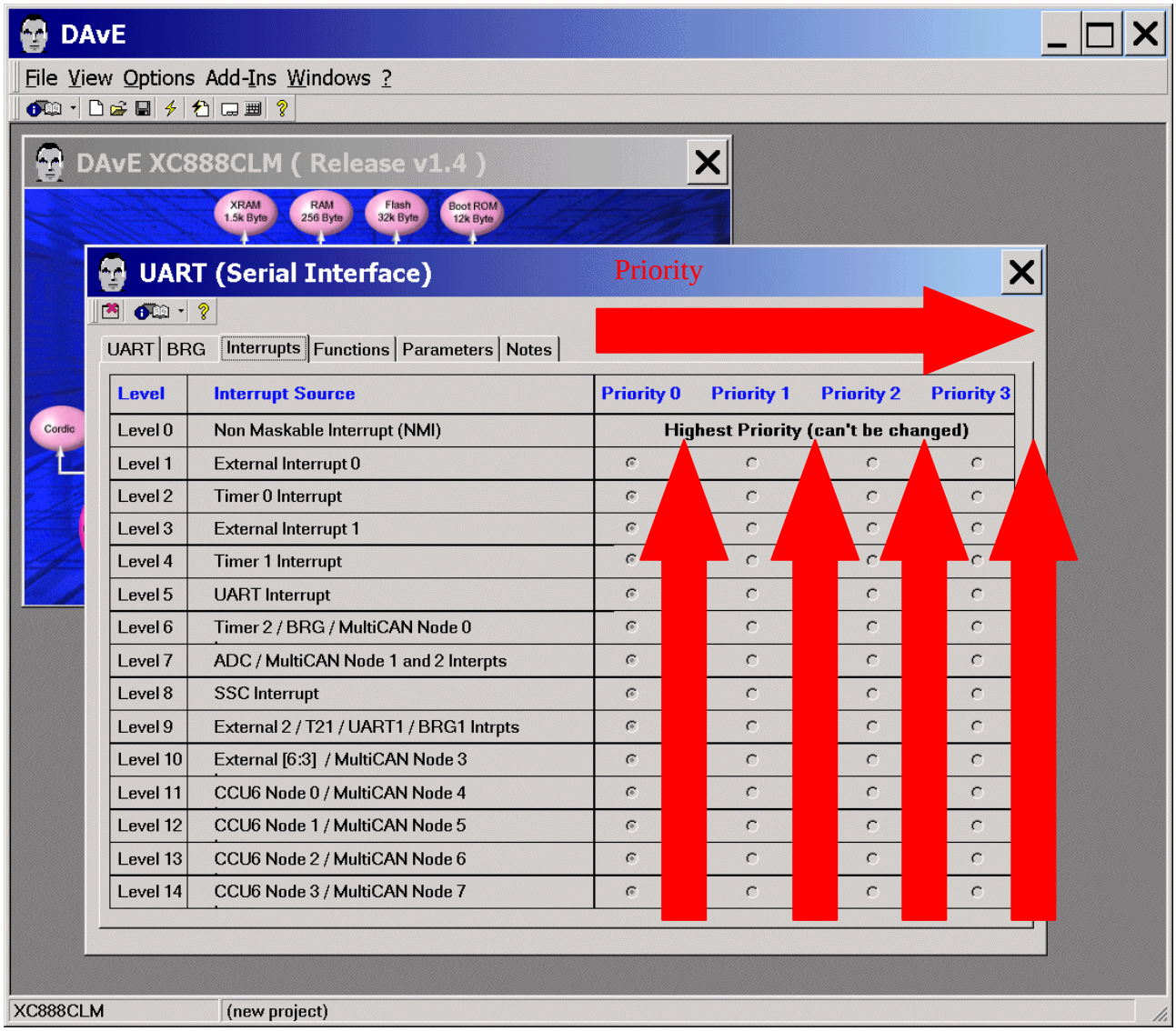
Note:
The RS232 serial interface (UART pins P1.0 and P1.1) is available via the USB port as virtual COM port (e.g. COM5) which converts the TTL-UART-signals to USB-signals.

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)



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

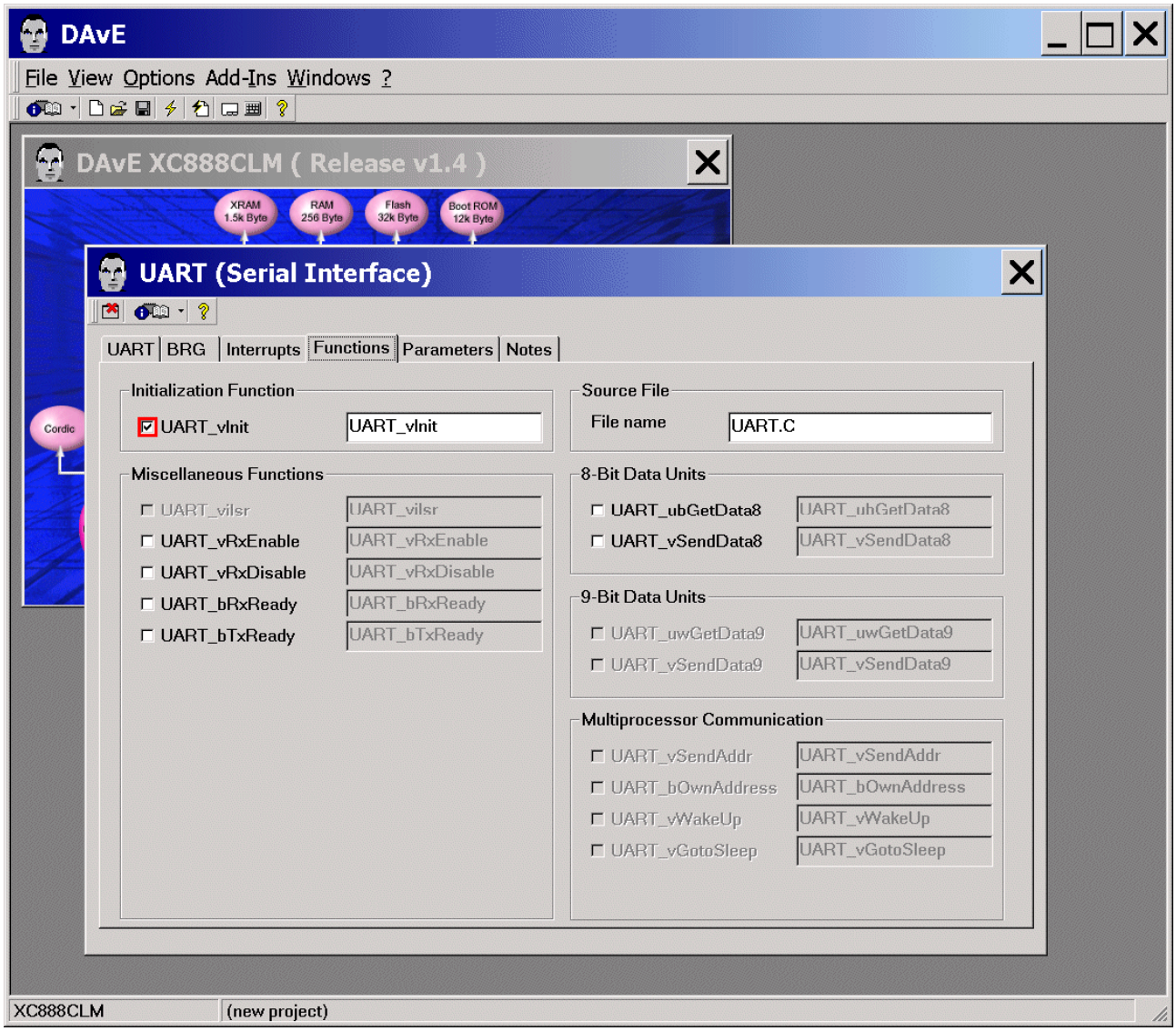
Interrupt Priorities:

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

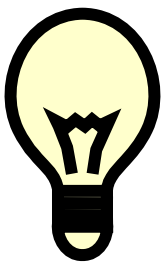
Interrupt Priority:	
6	NMI
5	Interrupt Priority 3
4	Interrupt Priority 2
3	Interrupt Priority 1
2	Interrupt Priority 0
1	Main

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.

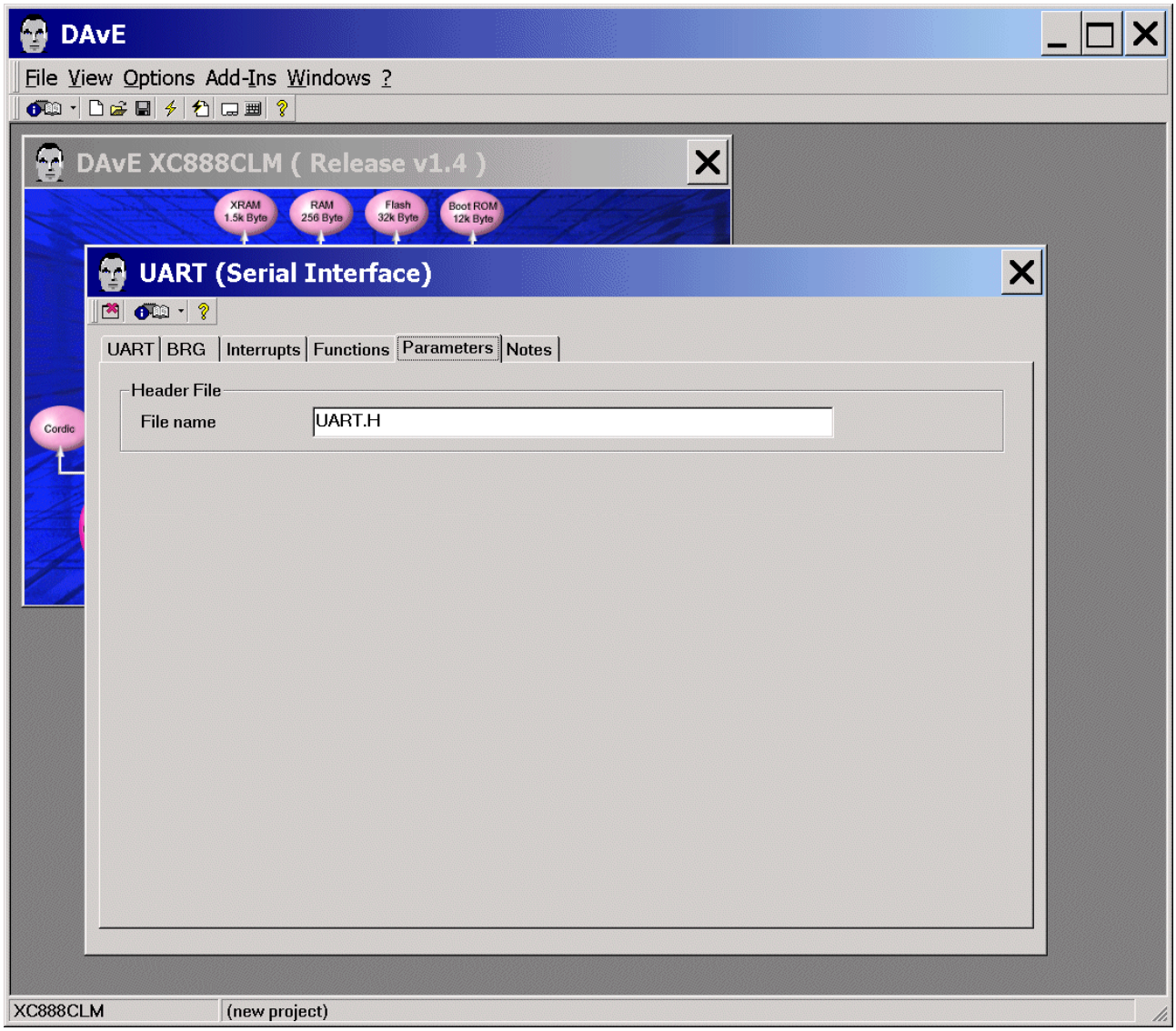
Functions: Initialization Function: click ✓ UART_vInit



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



Parameters: (do nothing)

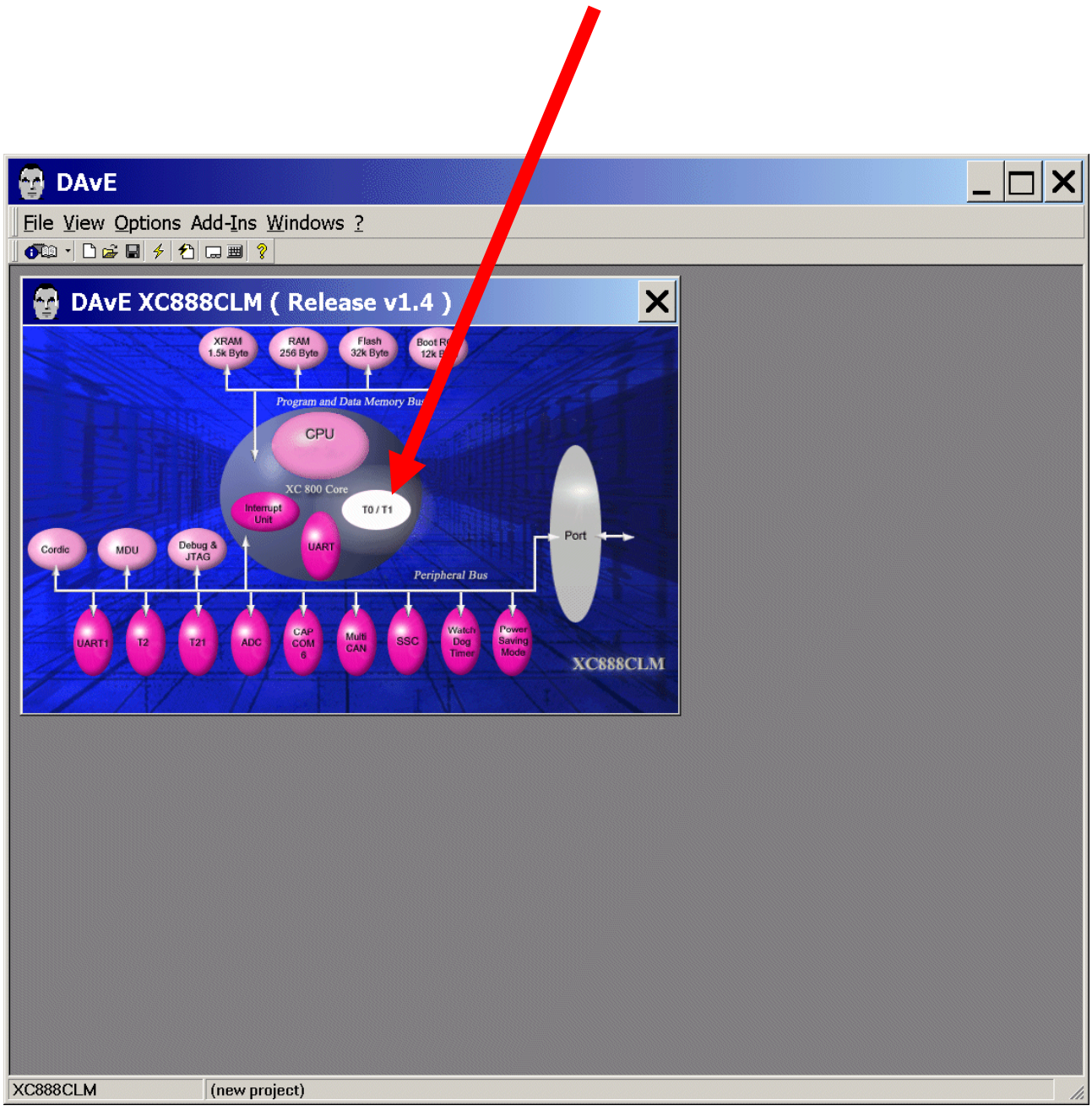


Notes: If you wish, you can insert your comments here.

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

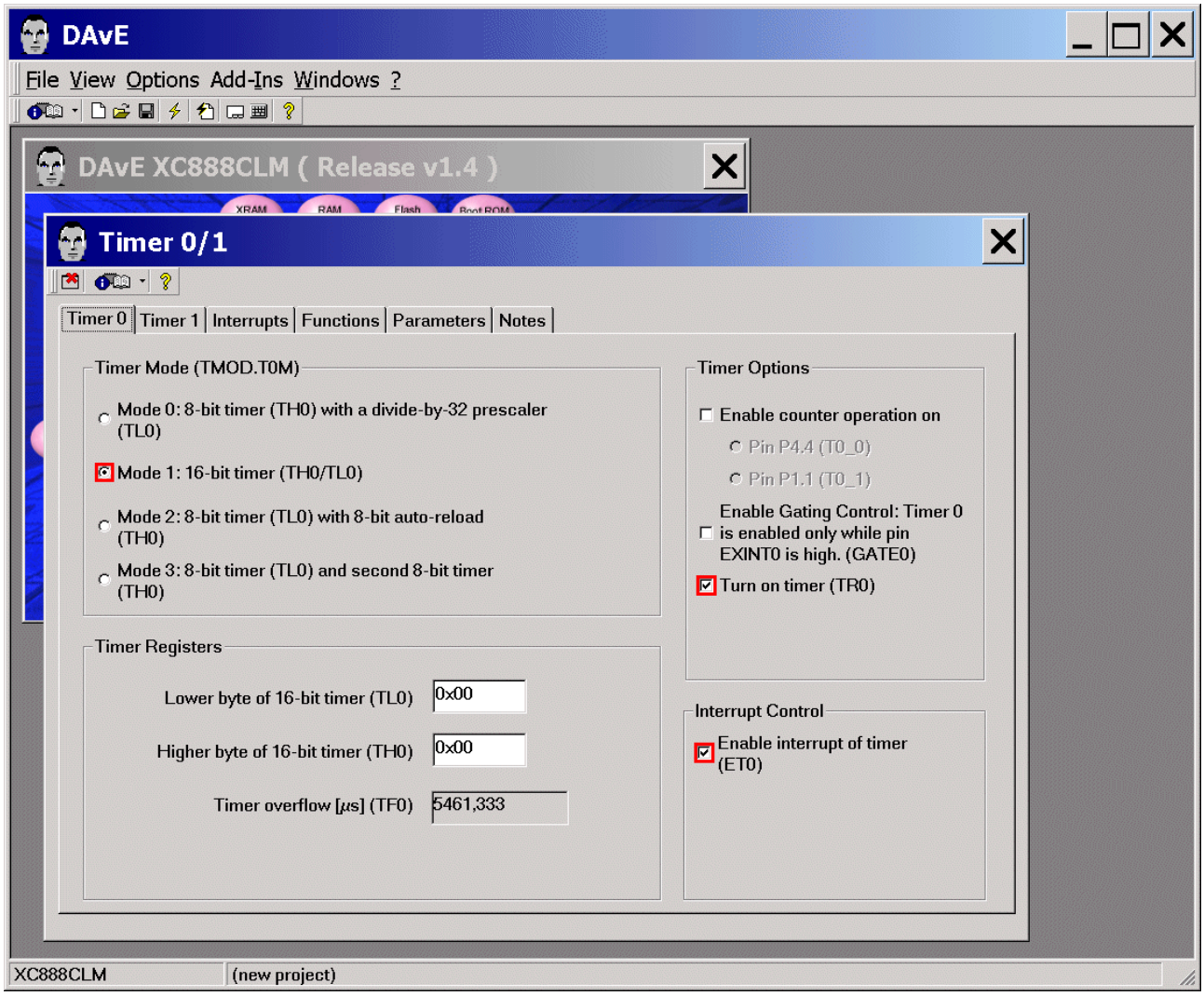
Configure Timer T0:

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



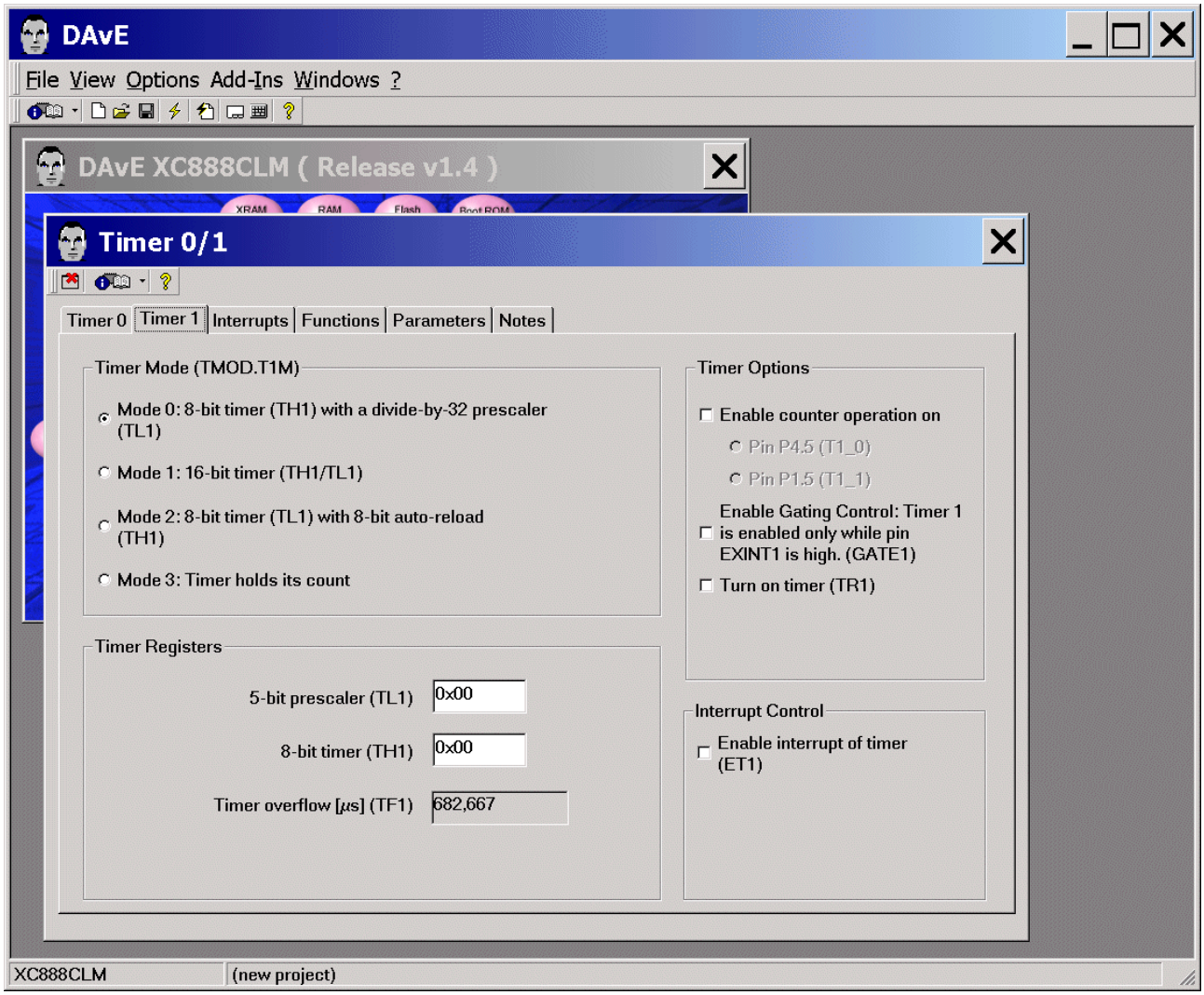
Note:
The LED on Port_1_Pin_5 will be blinking (if selected in the main menu) with a frequency of about 1 second (done in the Timer_0-Interrupt-Service-Routine). Therefore we have to configure Timer_0.

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)

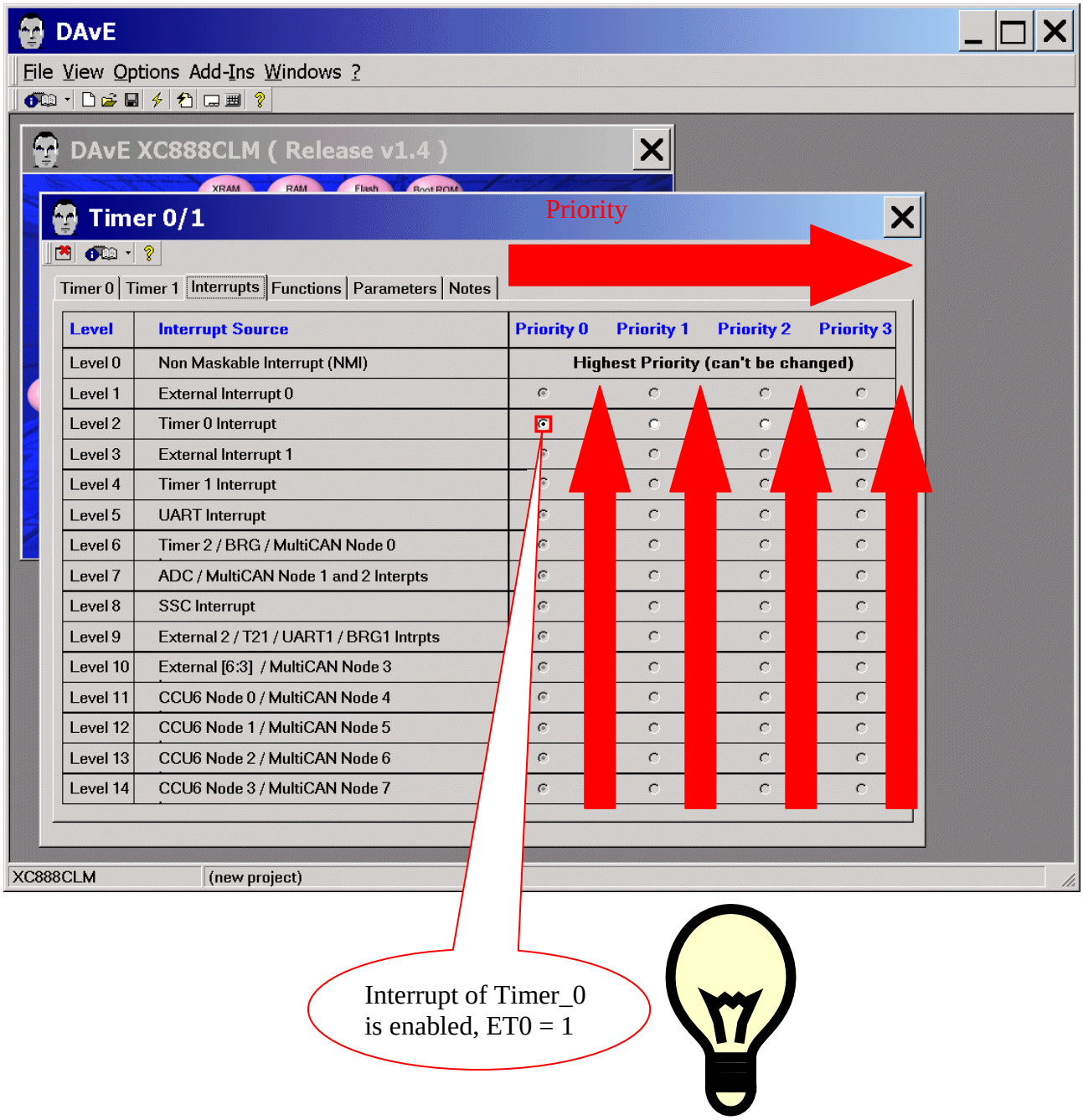


Note:
We need 183 Timer_0 overflows to achieve an approximate 1 second delay.
This will be handled in the Timer_0 interrupt function.
183 * 5461,333 µs = 0,9994 s.

Timer1: do nothing (not used)

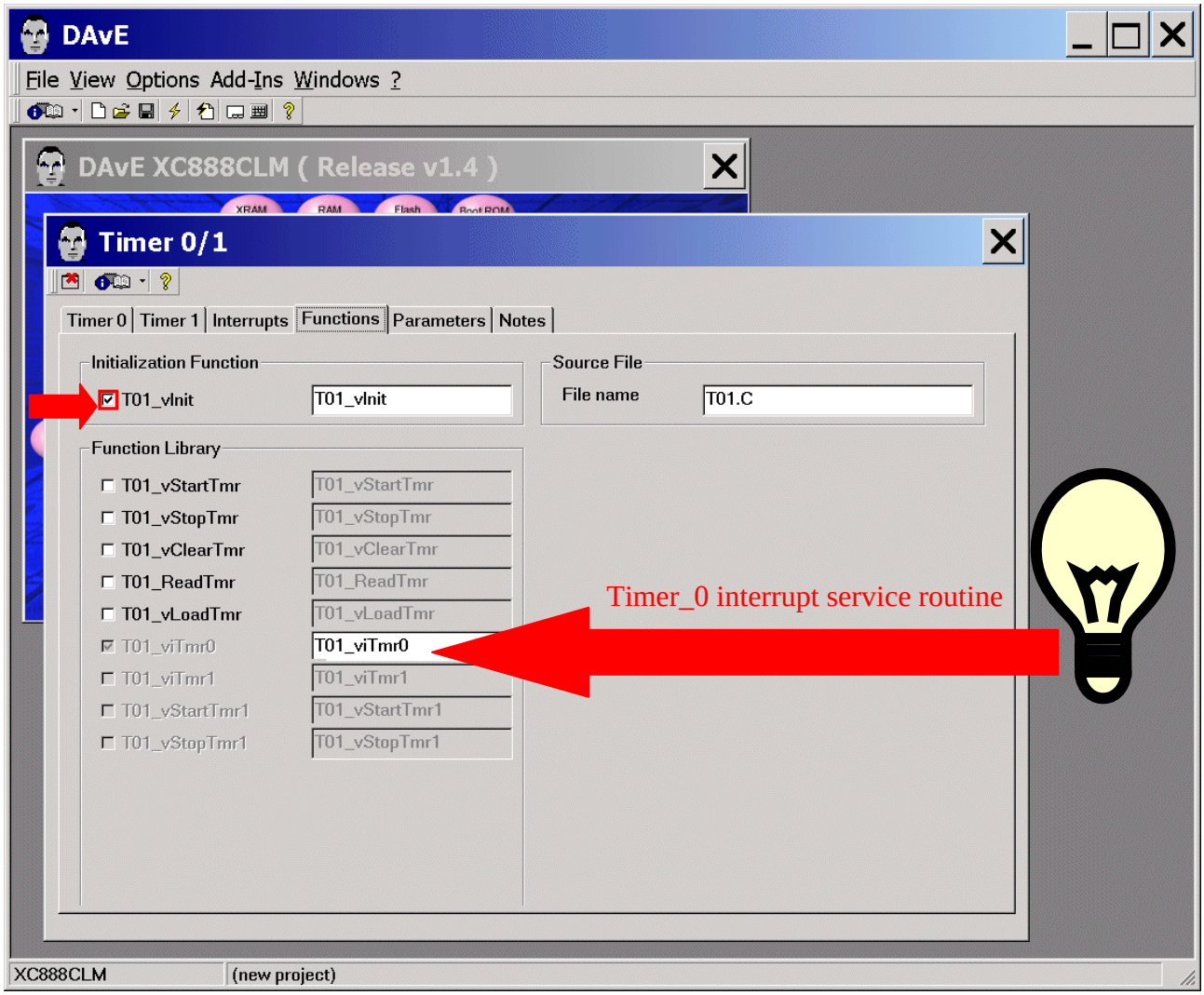


Interrupts: (do nothing)



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: click ✓ 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).

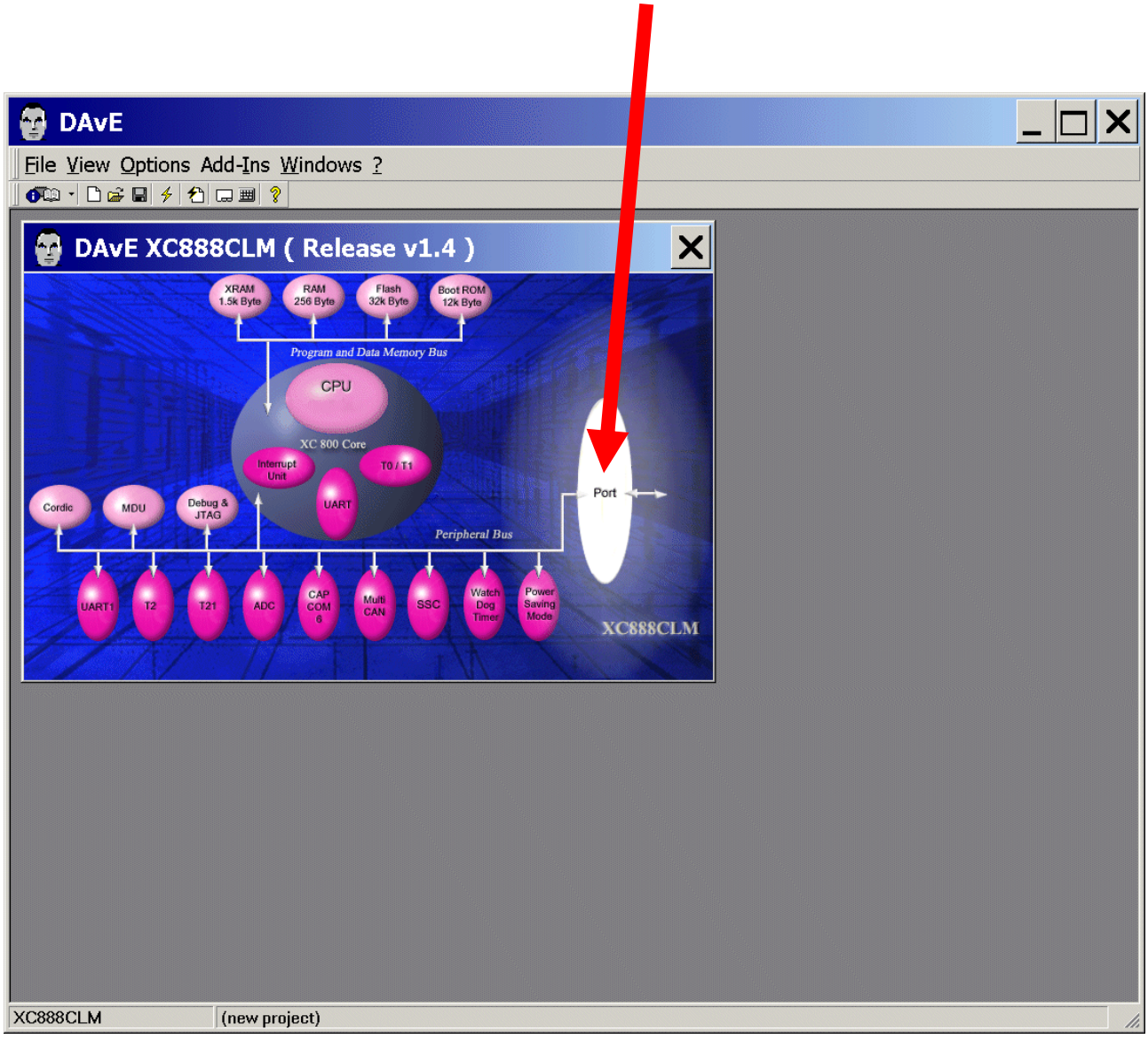
The hardware-generated LCALL pushes the contents of the Program Counter (PC) onto the stack (but it does not save the PSW) and reloads the PC with an address that depends on the source of the interrupt being vectored to (interrupt vector addresses see User’s Manual).

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.

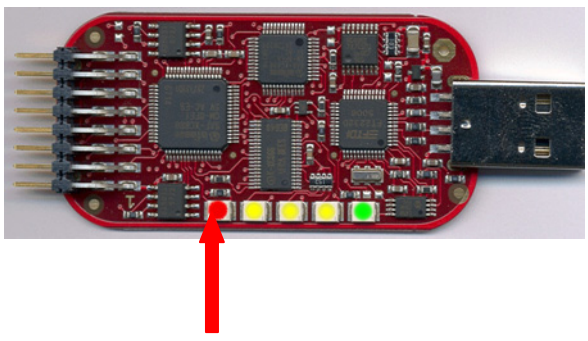
Parameters: (do nothing)
Notes: If you wish, you can insert your comments here.
Exit this dialog now by clicking  the close button.

Configure Port 1 Pin 5 to Output:

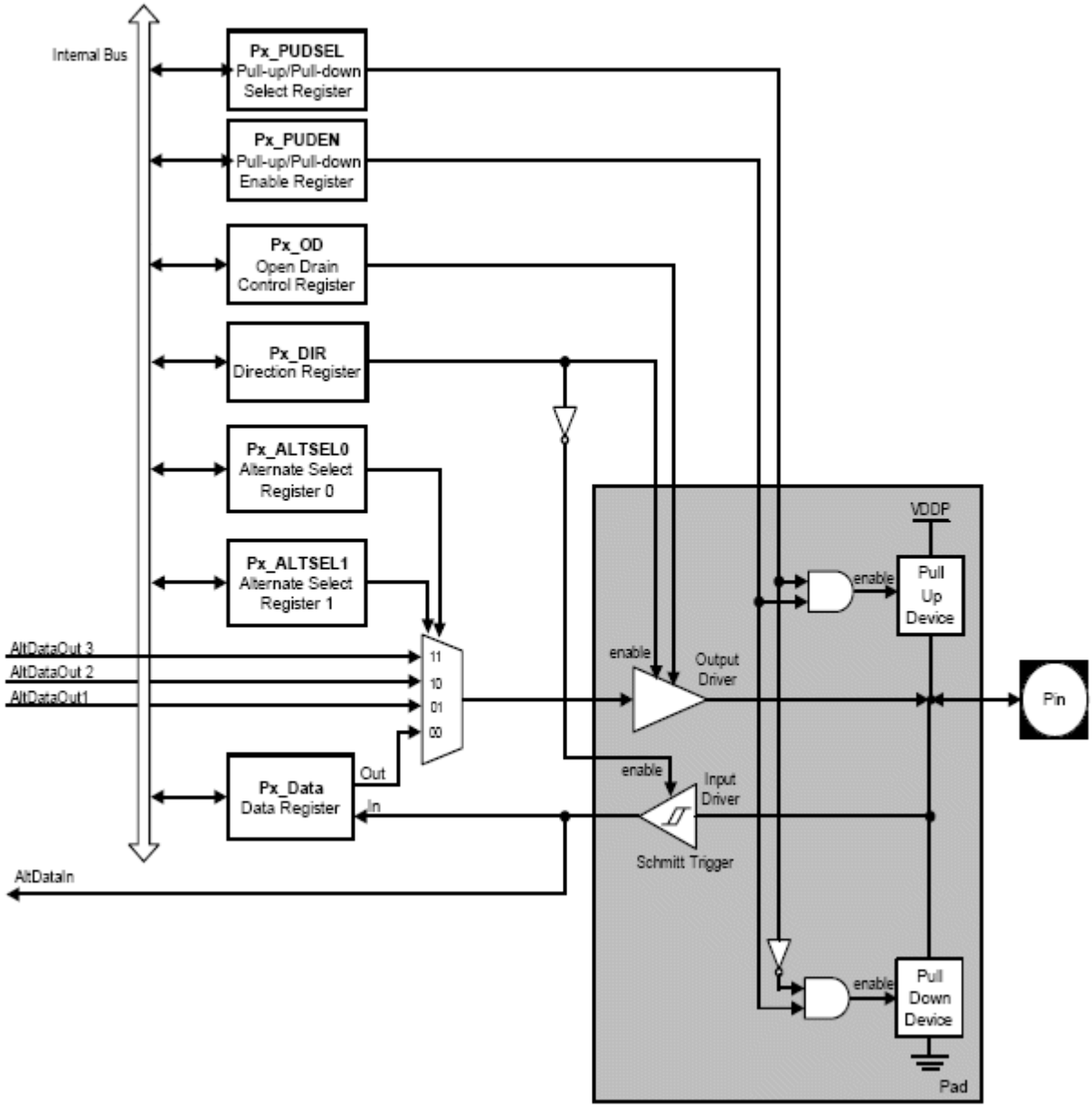
The configuration window/dialog can be opened by clicking the specific block/module.



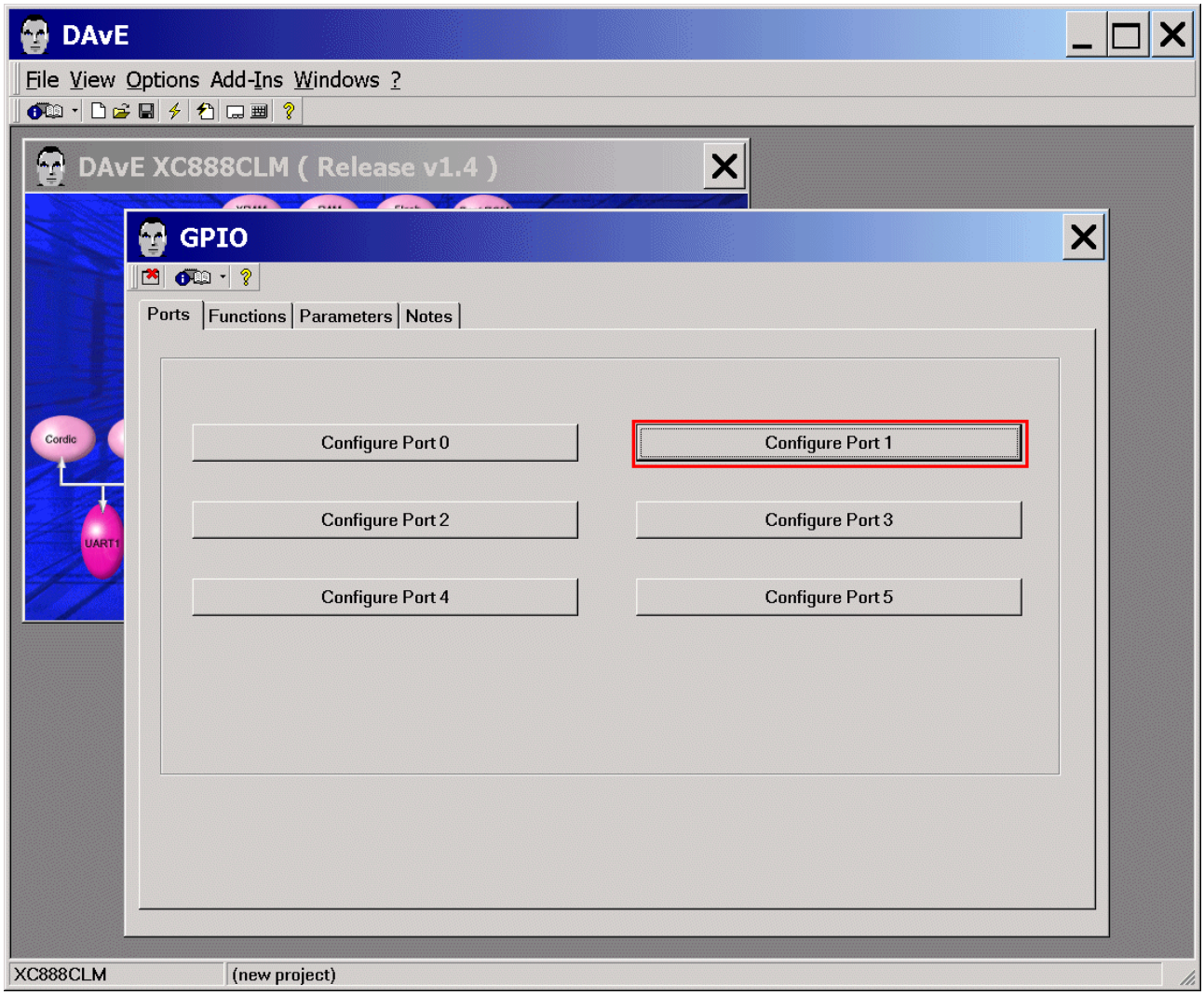
Note:
The User LED (red) is connected to Port_1_Pin_5.



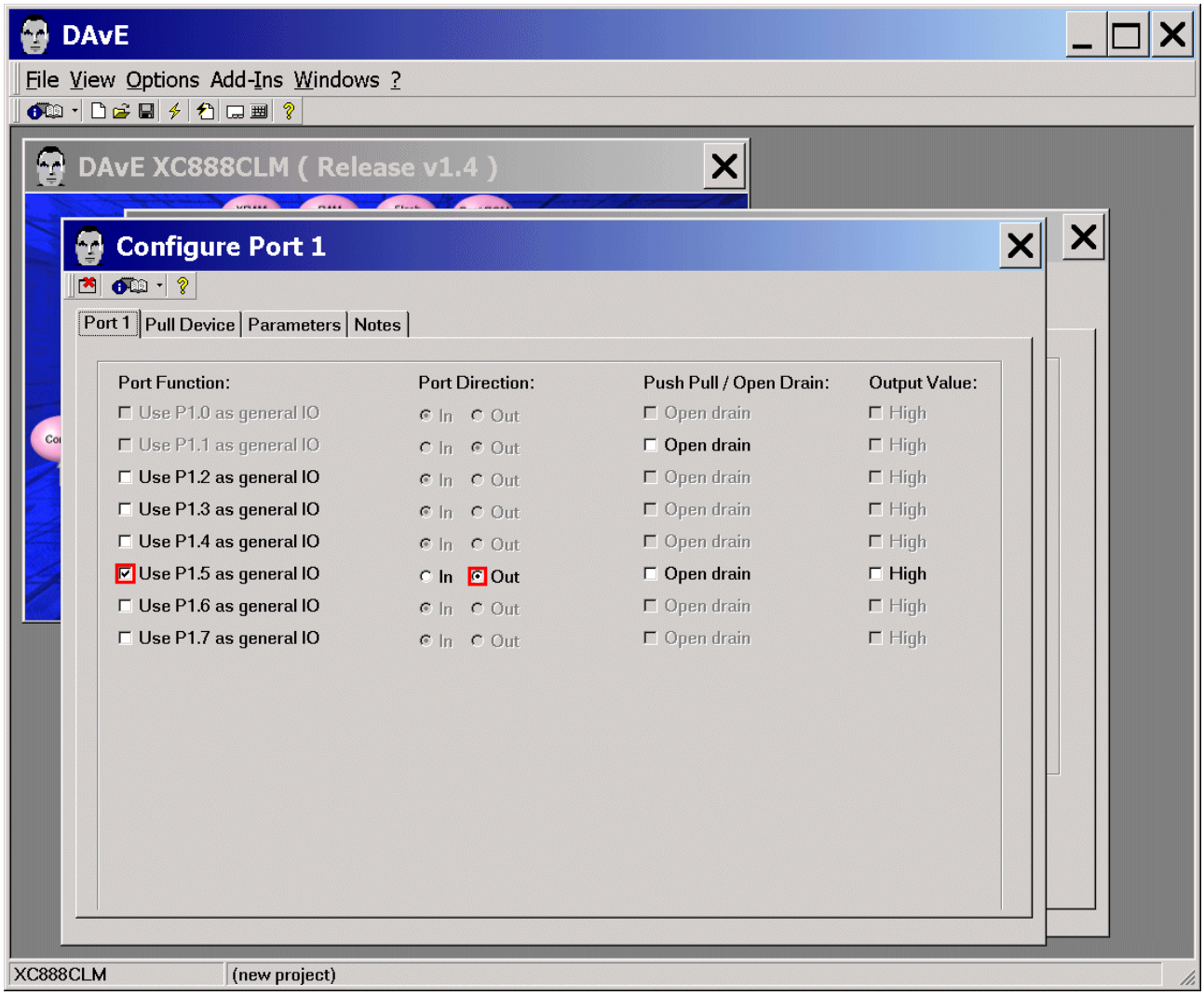
Additional information: Parallel Ports – General Structure (Source: User's Manual):



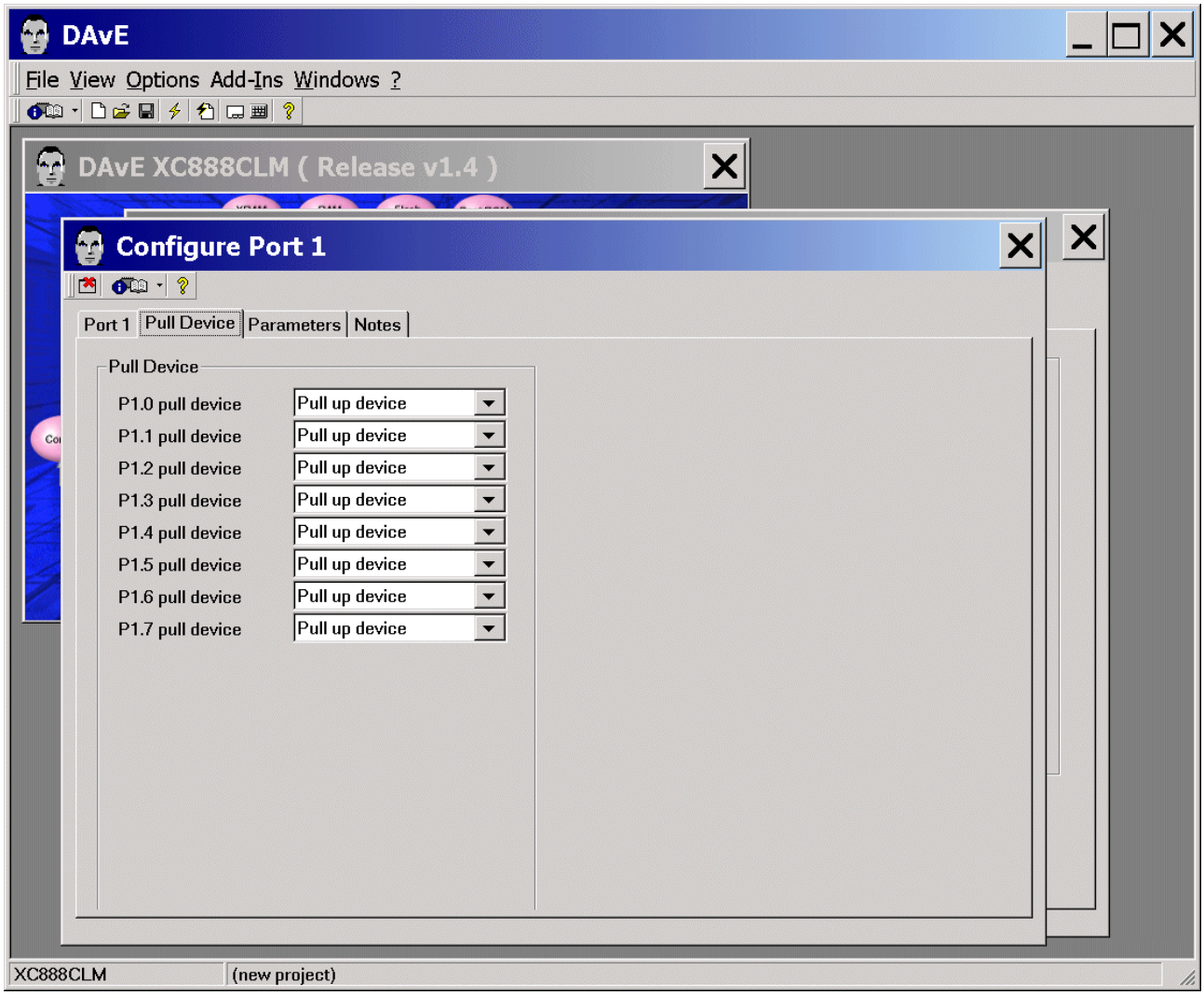
Ports: click “Configure Port 1”



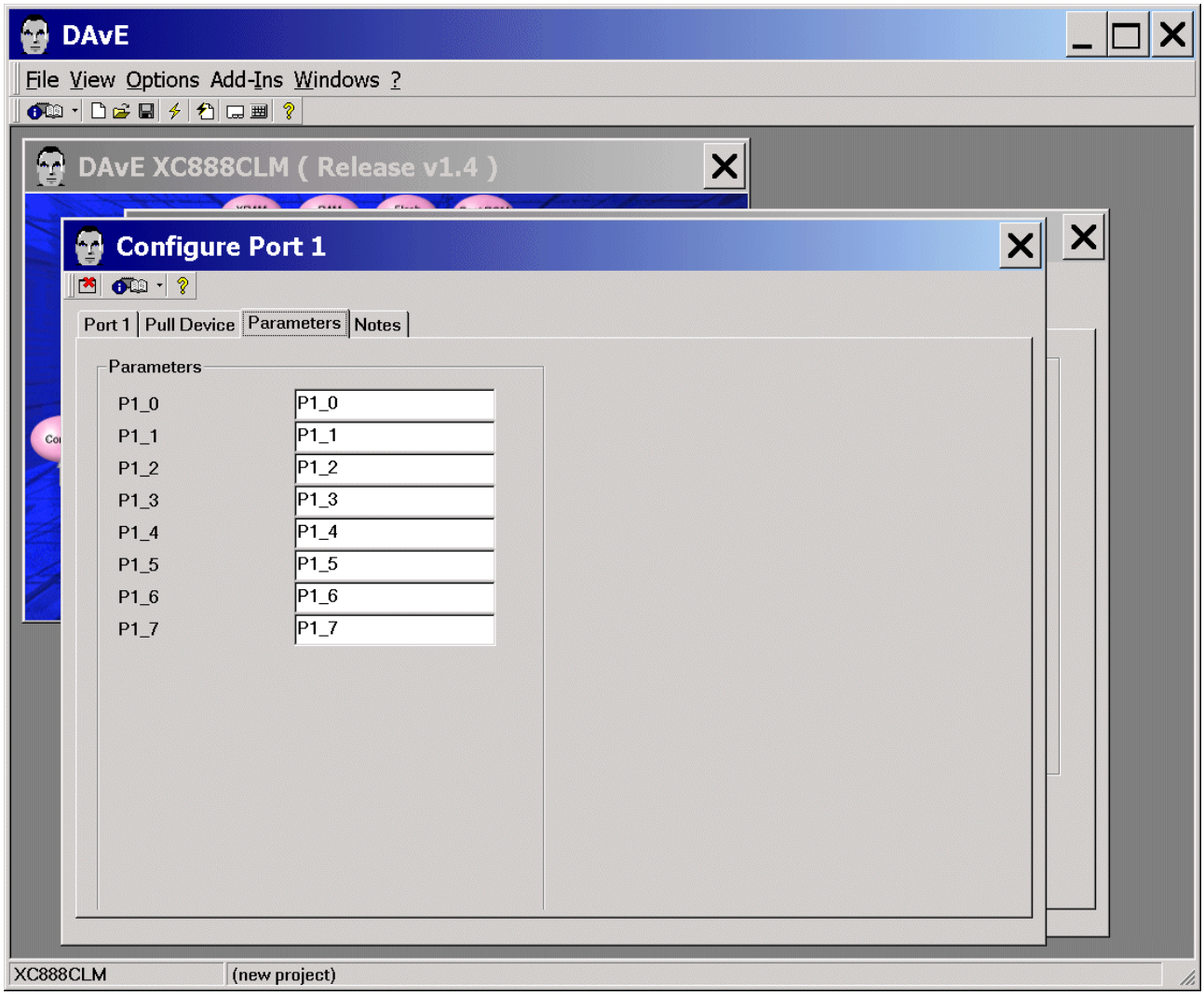
Port 1: Port Function: click ✓ Use P1.5 as general IO - Port Direction: click ⊙ Out



Pull Device: (do nothing)



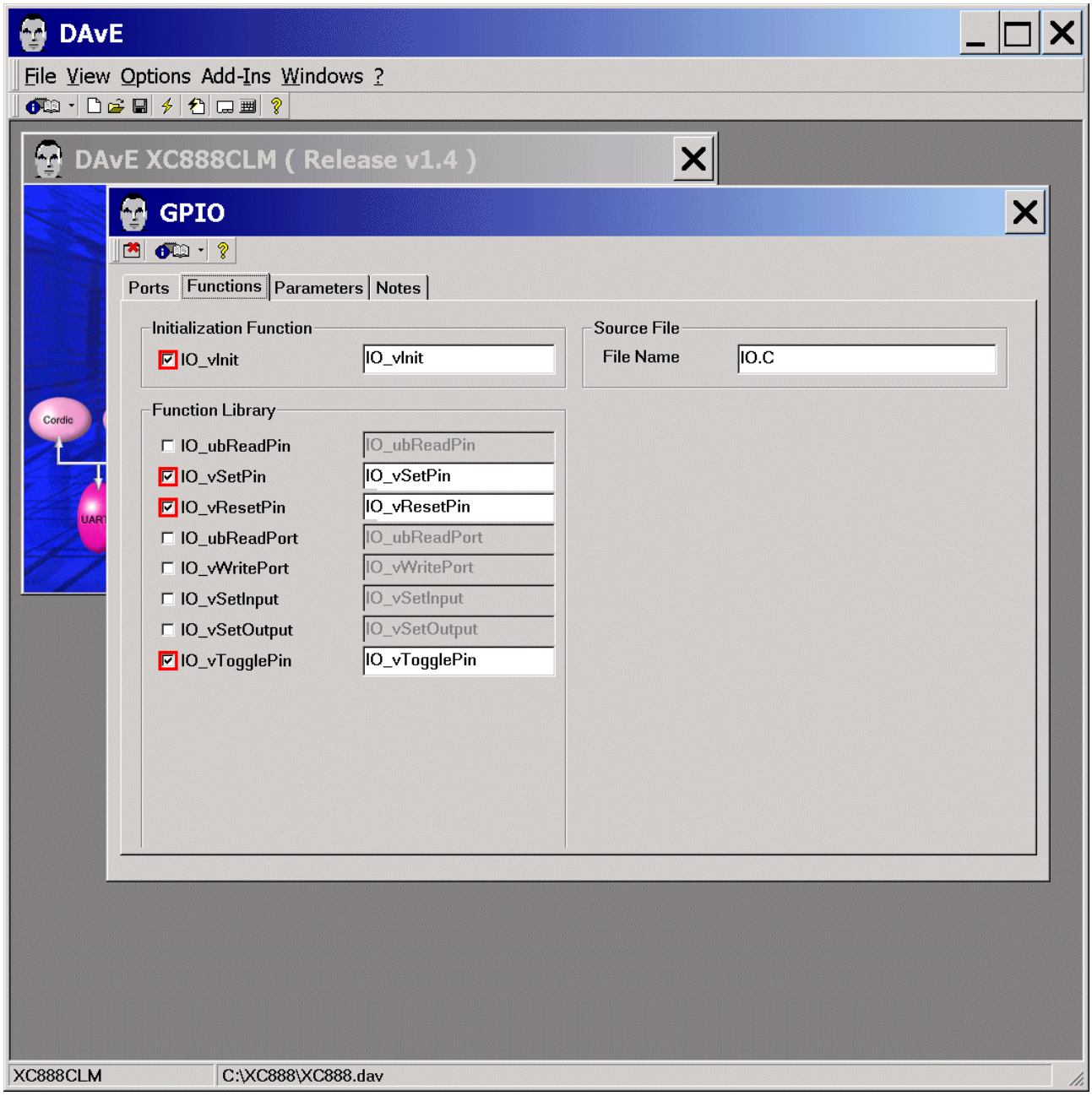
Parameters: (do nothing)



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

Exit this dialog now by clicking the close button.

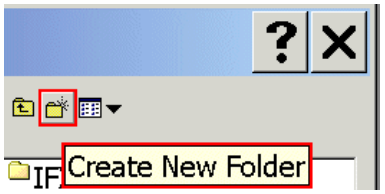
Functions: Initialization Functions: click ✓ IO_vInit
Functions: Function Library: click ✓ IO_vSetPin
Functions: Function Library: click ✓ IO_vResetPin
Functions: Function Library: click ✓ IO_vTogglePin



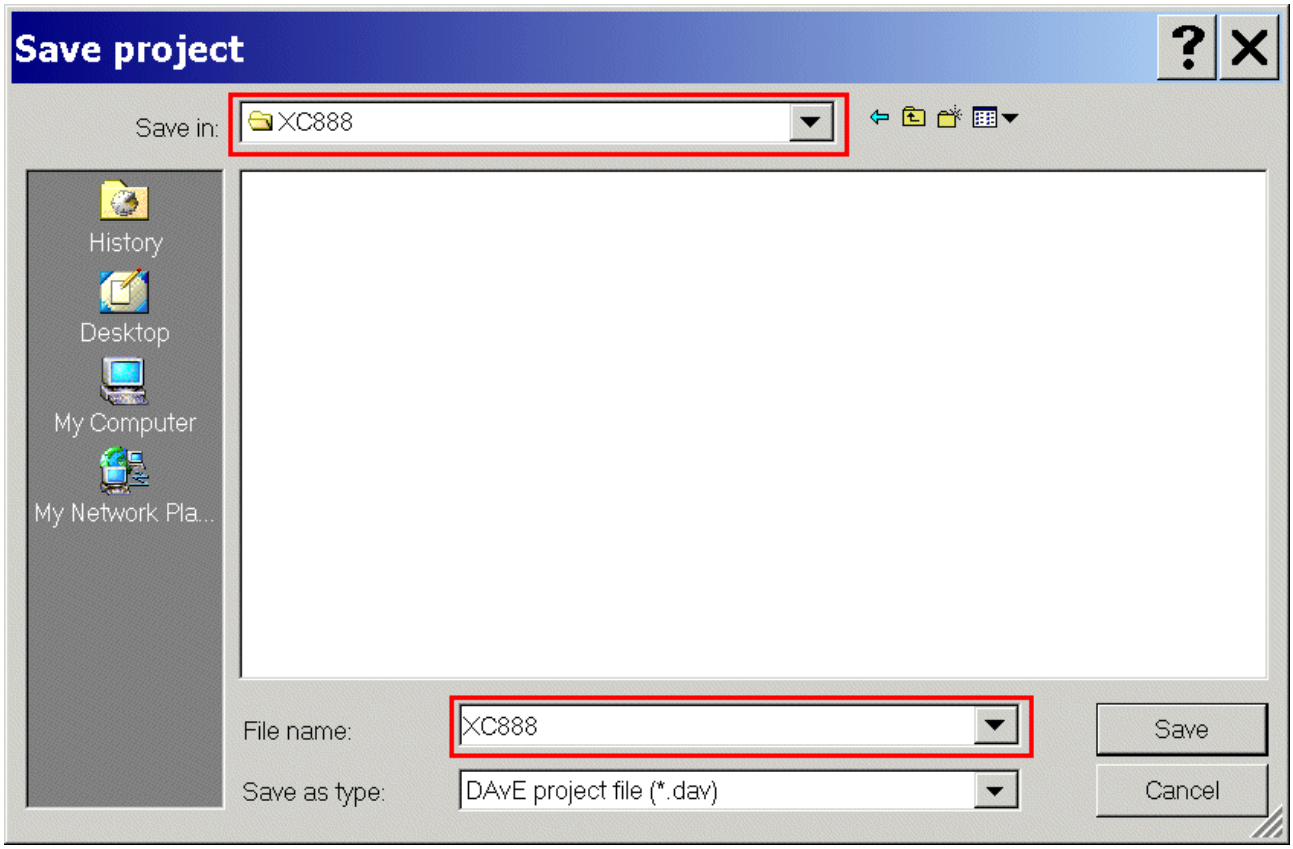
Parameters: (do nothing)
Notes: If you wish, you can insert your comments here.
Exit this dialog now by clicking the close button.

Save the project:

File
Save

Save project: Save in C:\XC888 (create new directory 

)
File name: XC888

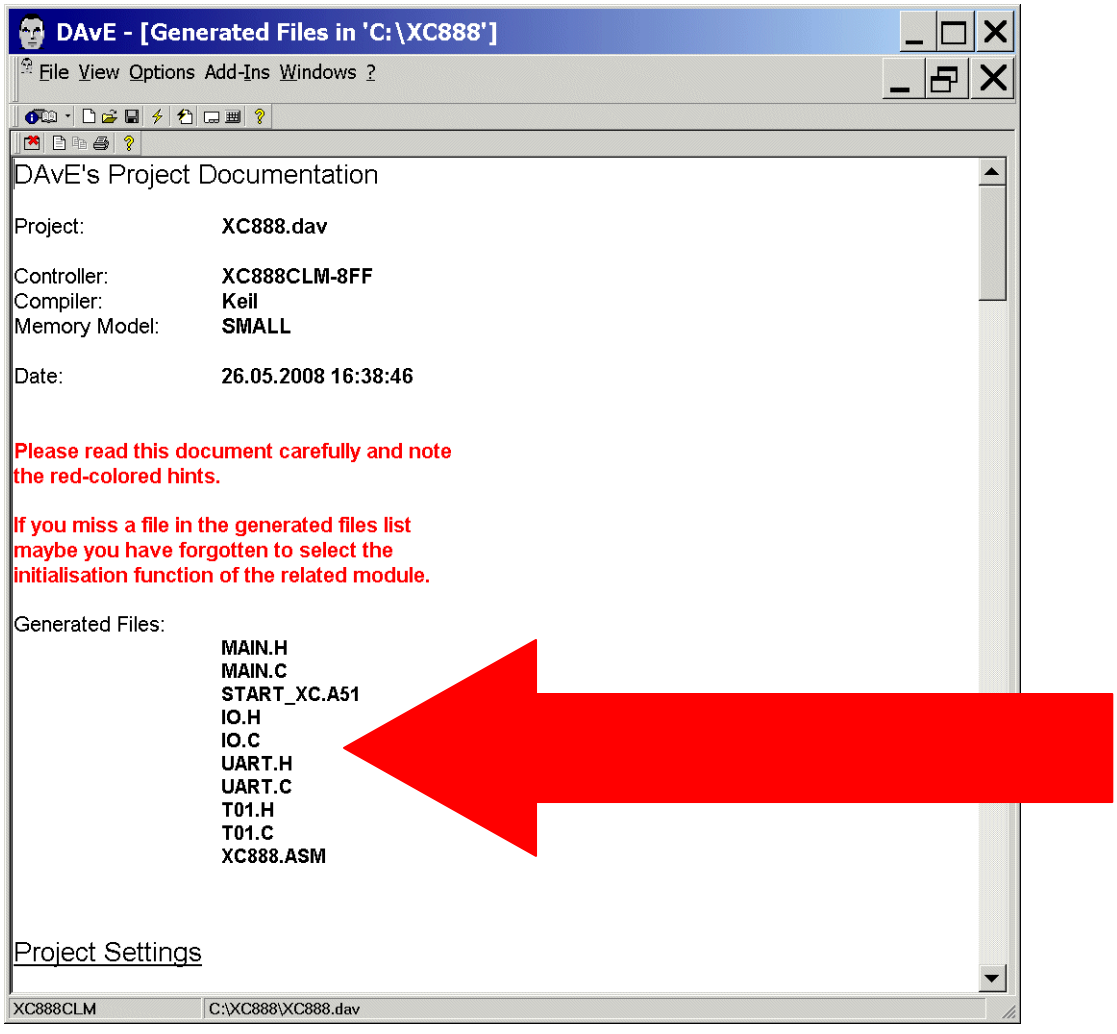


Save

Generate Code:

File Generate Code	or click 

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



File - Exit

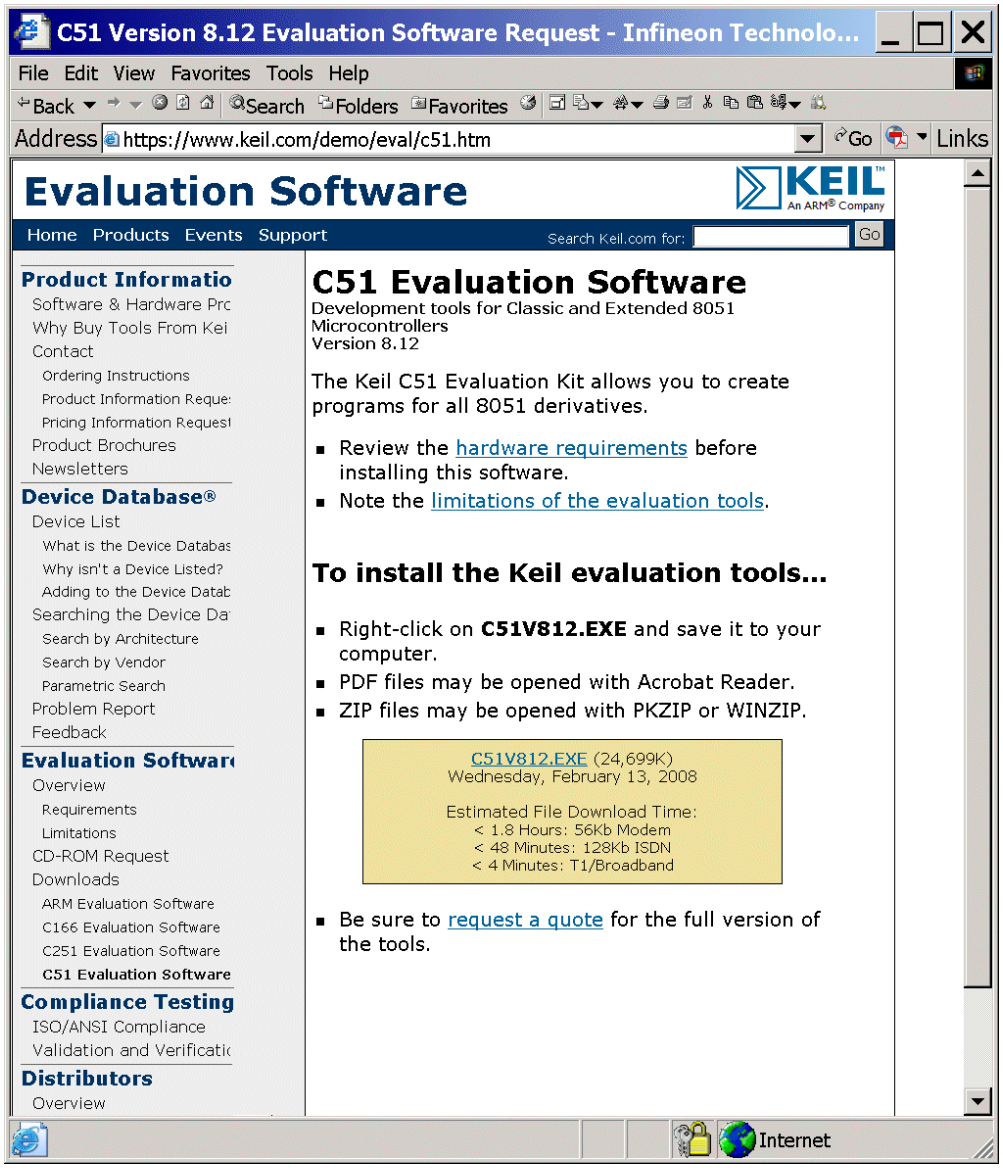
Save changes?

click Yes

4.) Using the KEIL - µVision 3 Development Tools:

Install the Tool chain:

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



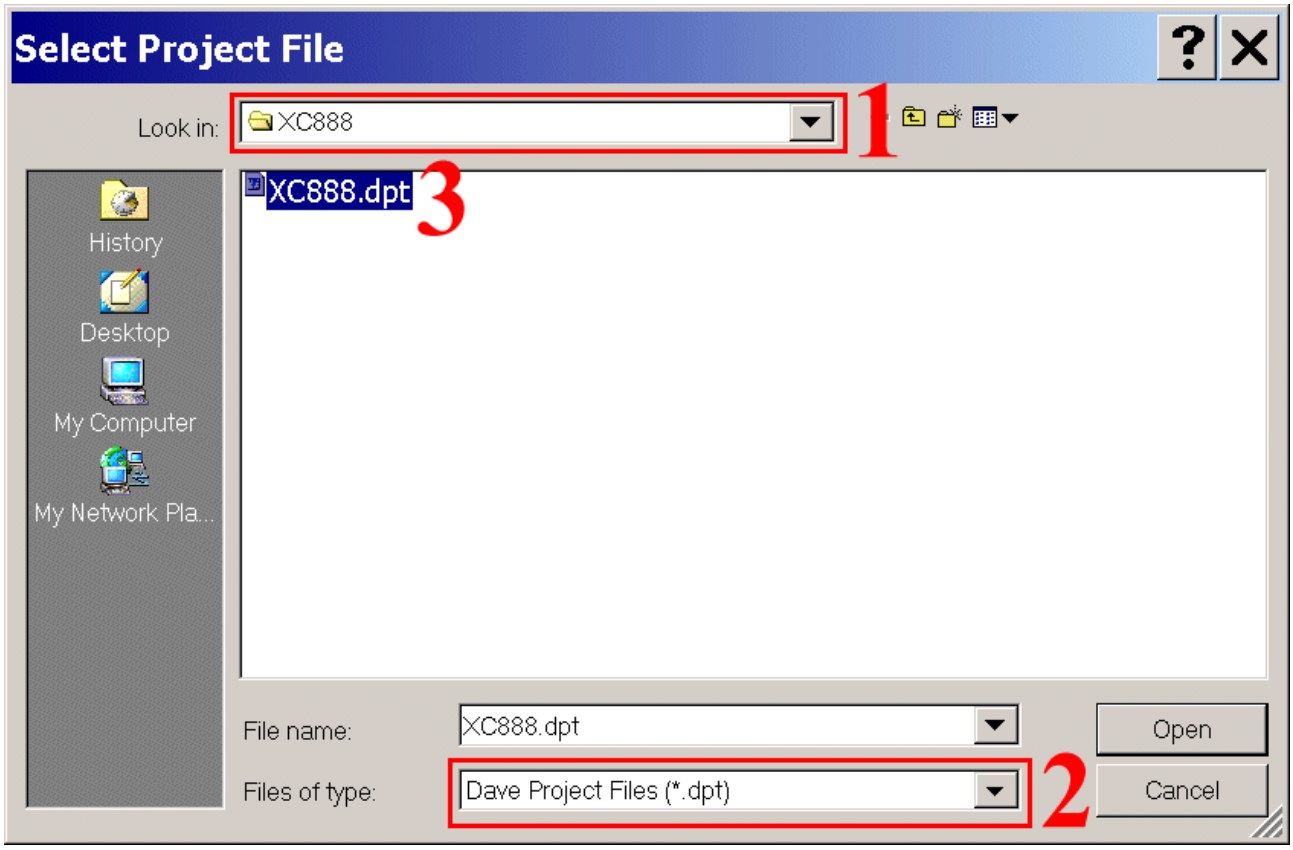
Execute C51V812c.EXE (- or any higher version)

Note:
Version 8.12c or higher is needed for the XC800 USCALE start kit!

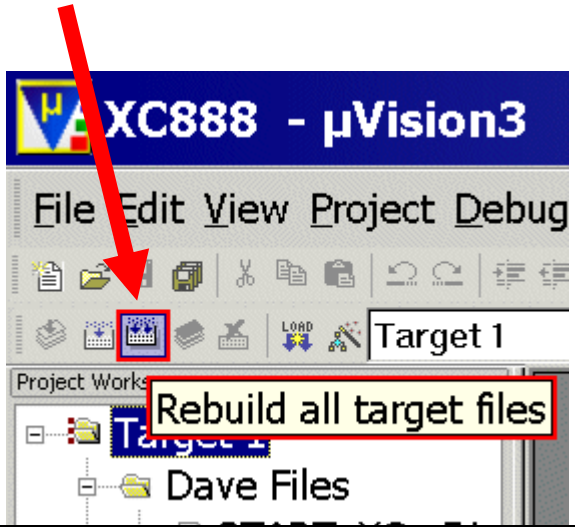
Start Keil µVision3 and open the DAvE Project:

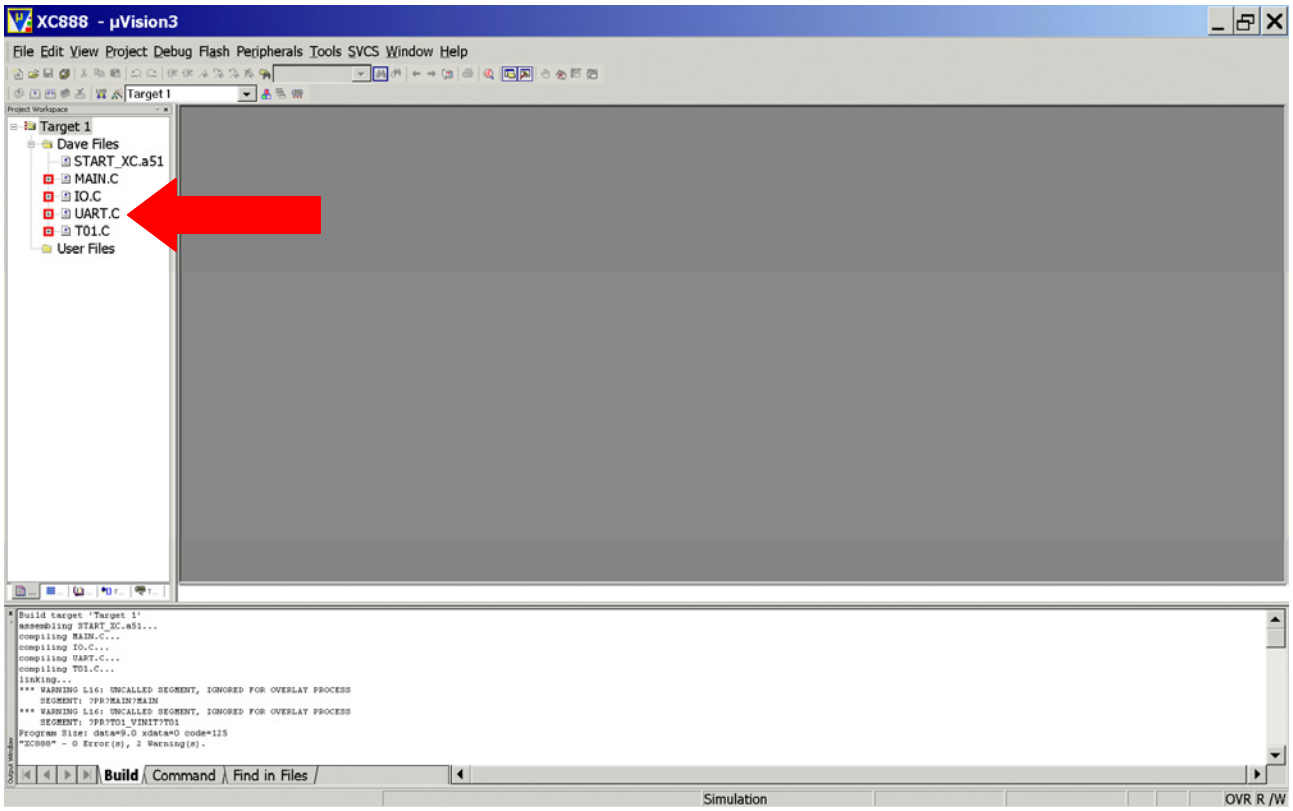
If you see an open project – close it: Project - Close Project

Project - Open Project
Select Project File: Look in: choose C:\XC888 (1)
Select Project File: Files of type: select Dave Project Files (2)
Choose/click XC888.dpt (3)



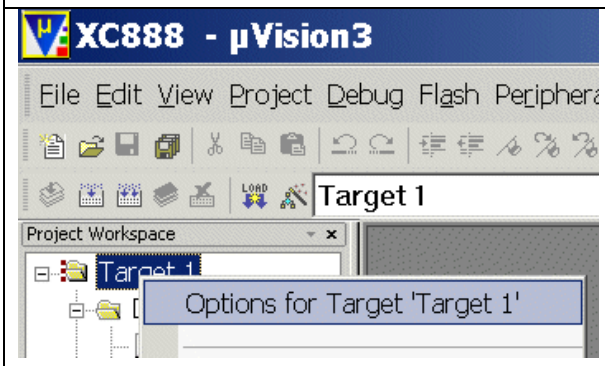
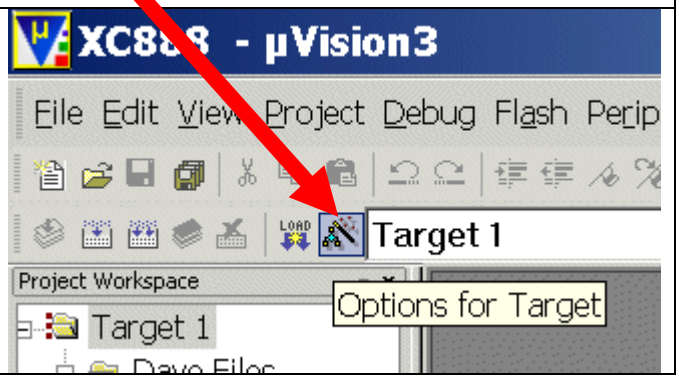
Click Open

Project – Rebuild all target files	**or**	**click** 

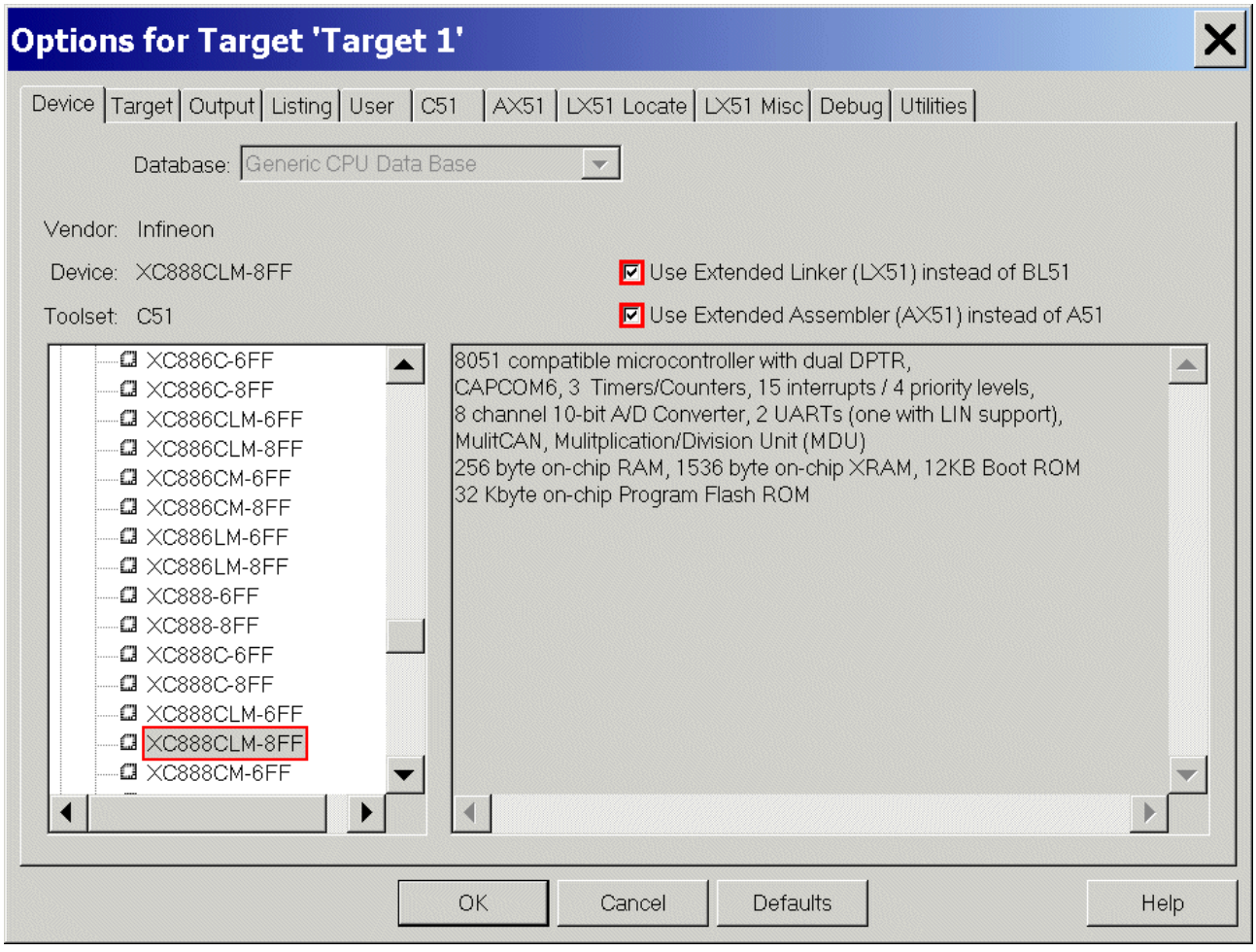


Note:
This step generates a makefile and shows the include files.

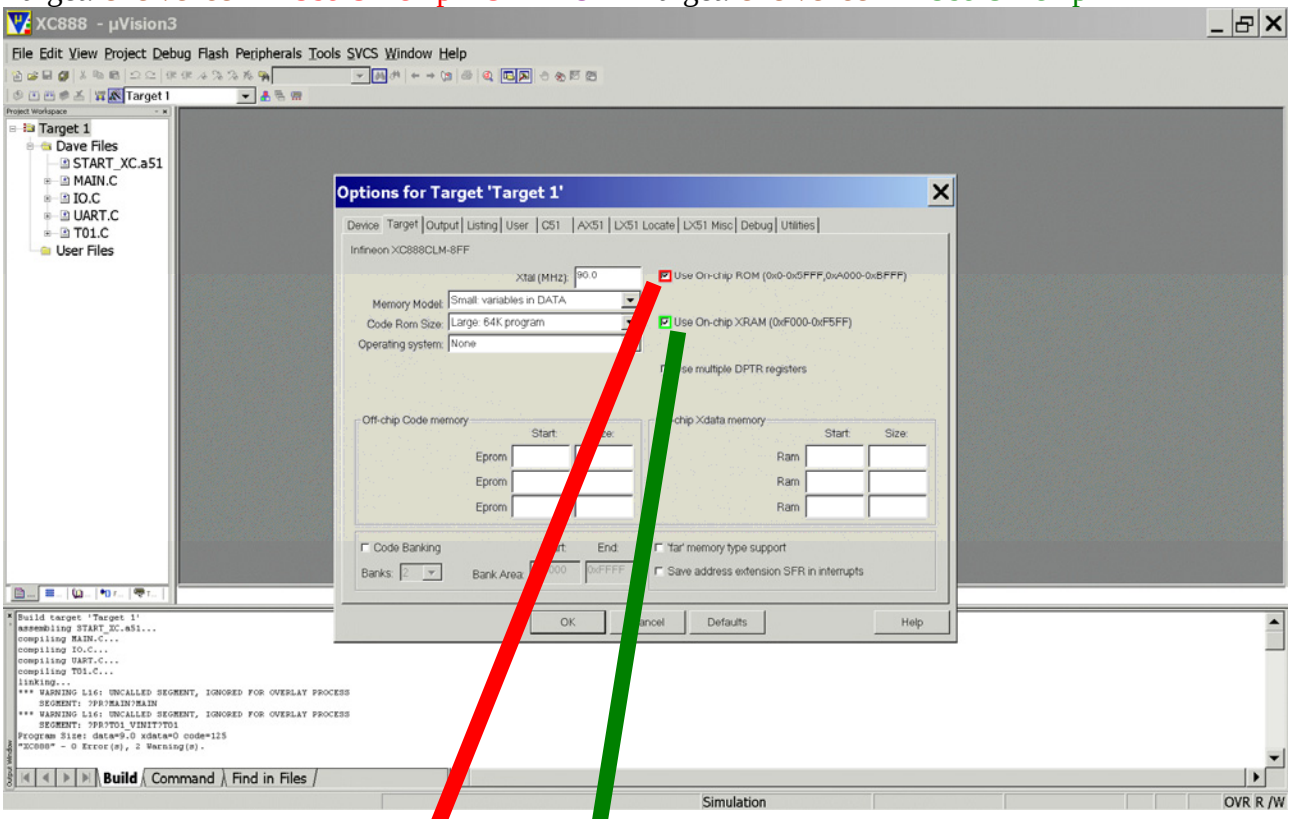
Configure Compiler, Assembler, Linker, Locater, Hex-Converter, Build Control, Simulator, Debugger and Utilities:

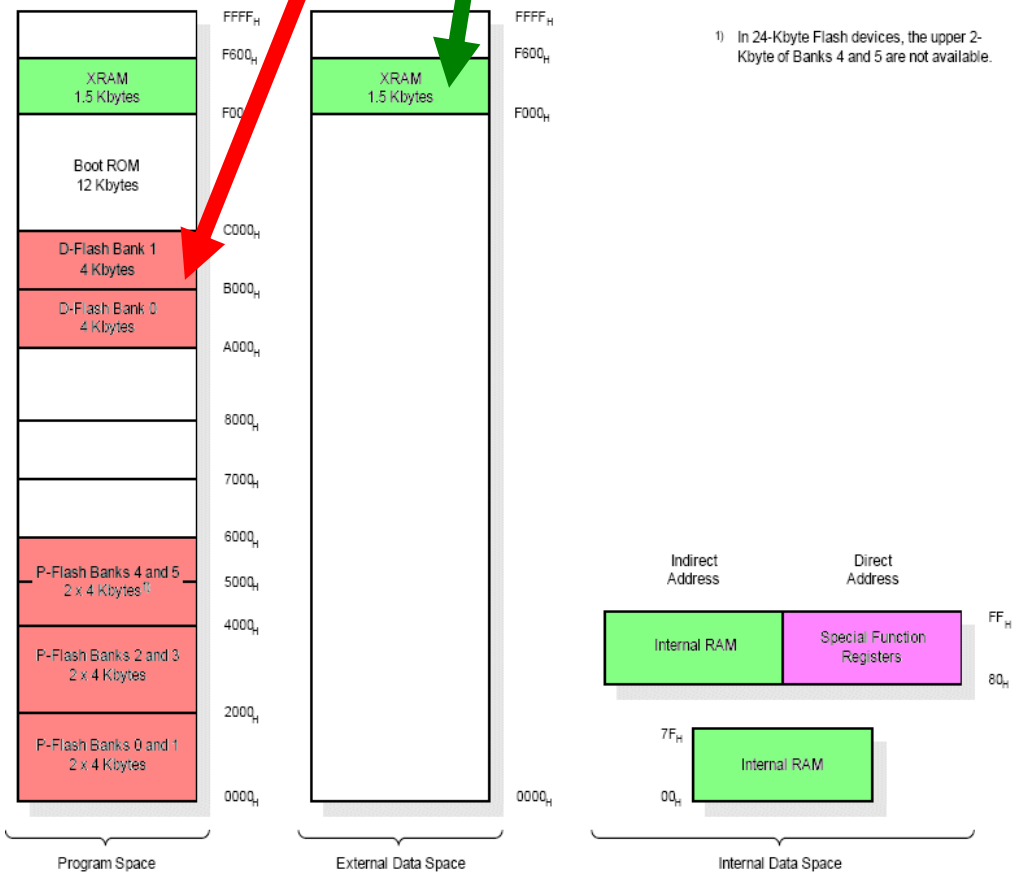
mouse position: (Project Workspace, Files): Target1 click right mouse button Options for Target ´Target1‘	or	click
		

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)

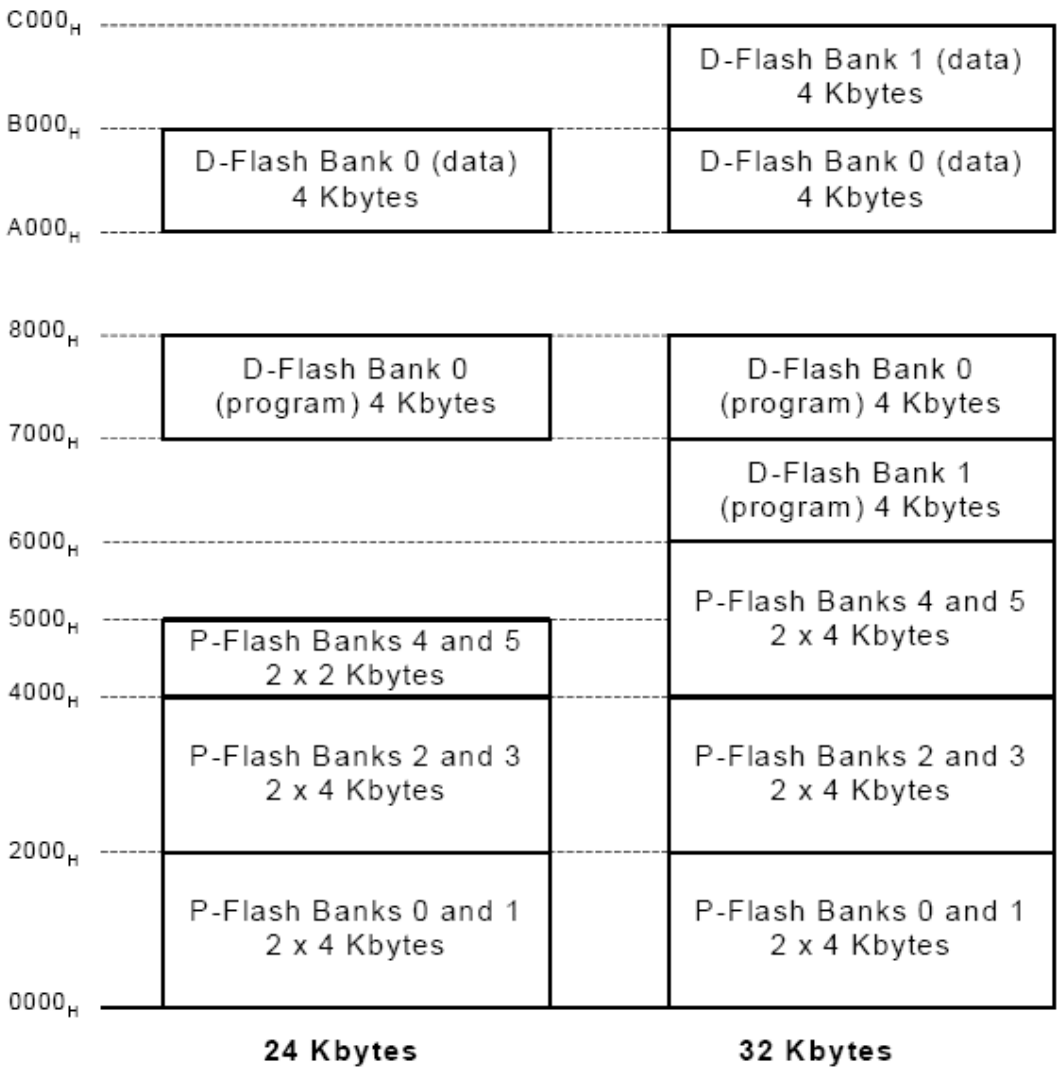


Target: click/check ✓ Use On-chip ROM & Target: click/check ✓ Use On-chip XRAM





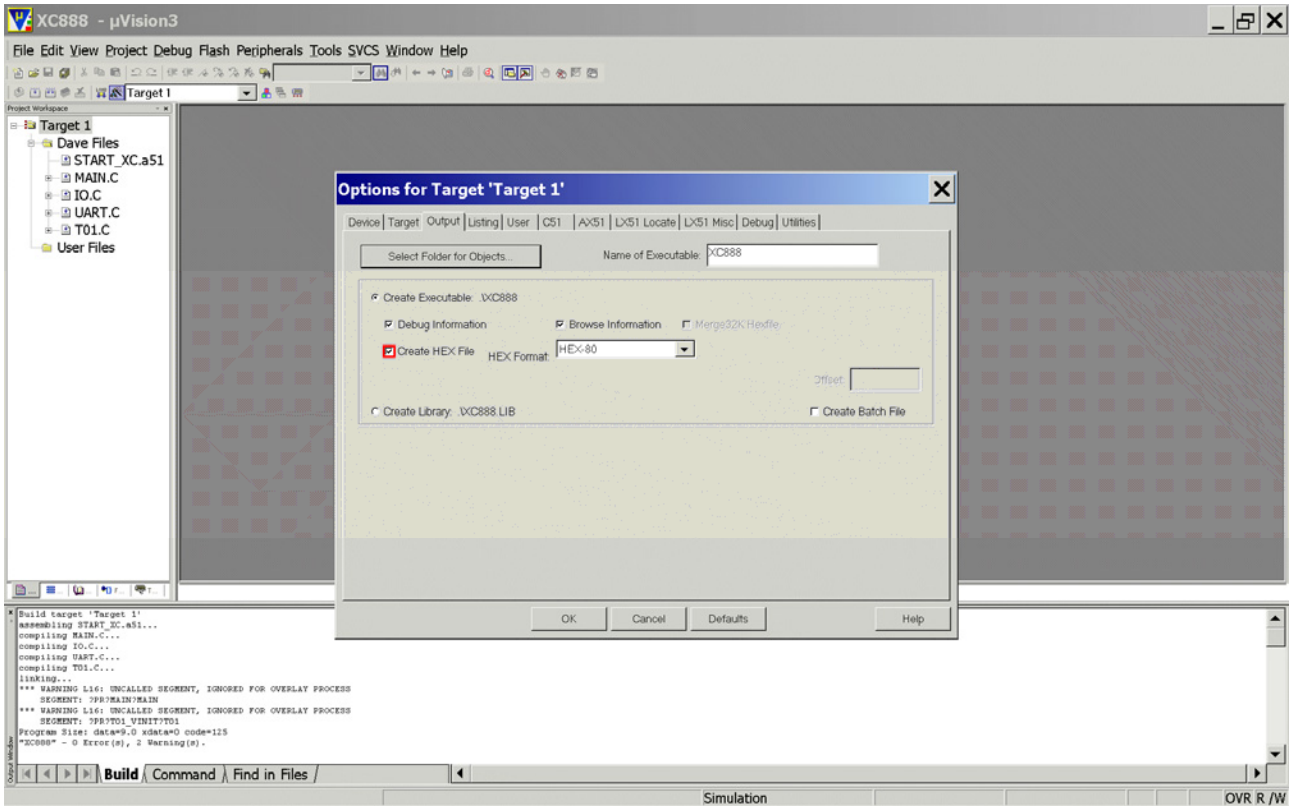
Additional information: Flash Memory Map (Source: User's Manual):



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_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 $B000_H$ – $BFFF_H$.

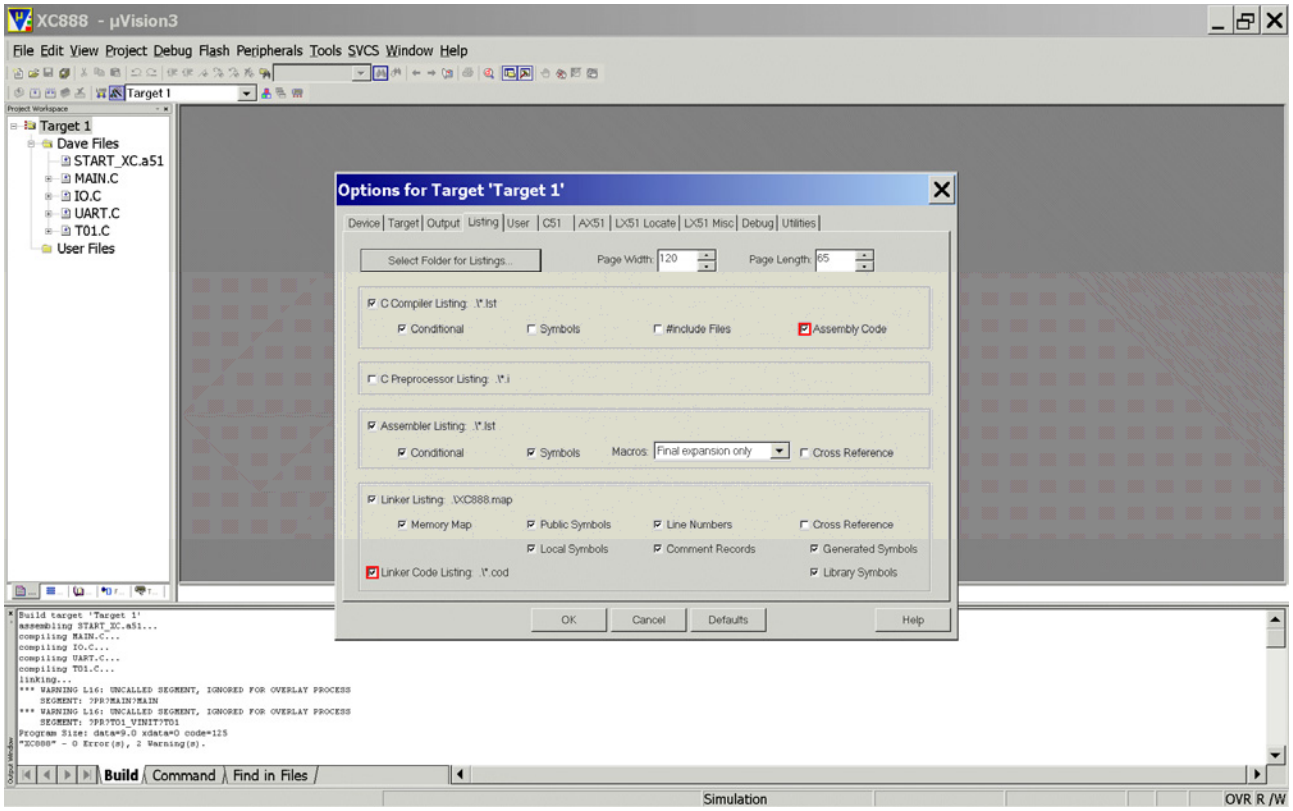
In general, the lower address spaces (6000_H – $6FFF_H$ and 7000_H – $7FFF_H$) should be used for D-Flash bank(s) contents that are intended to be used as program code. Alternatively, the higher address spaces ($A000_H$ – $AFFF_H$ and $B000_H$ – $BFFF_H$) should be used for D-Flash bank(s) contents that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

Output: click ✓ Create HEX File



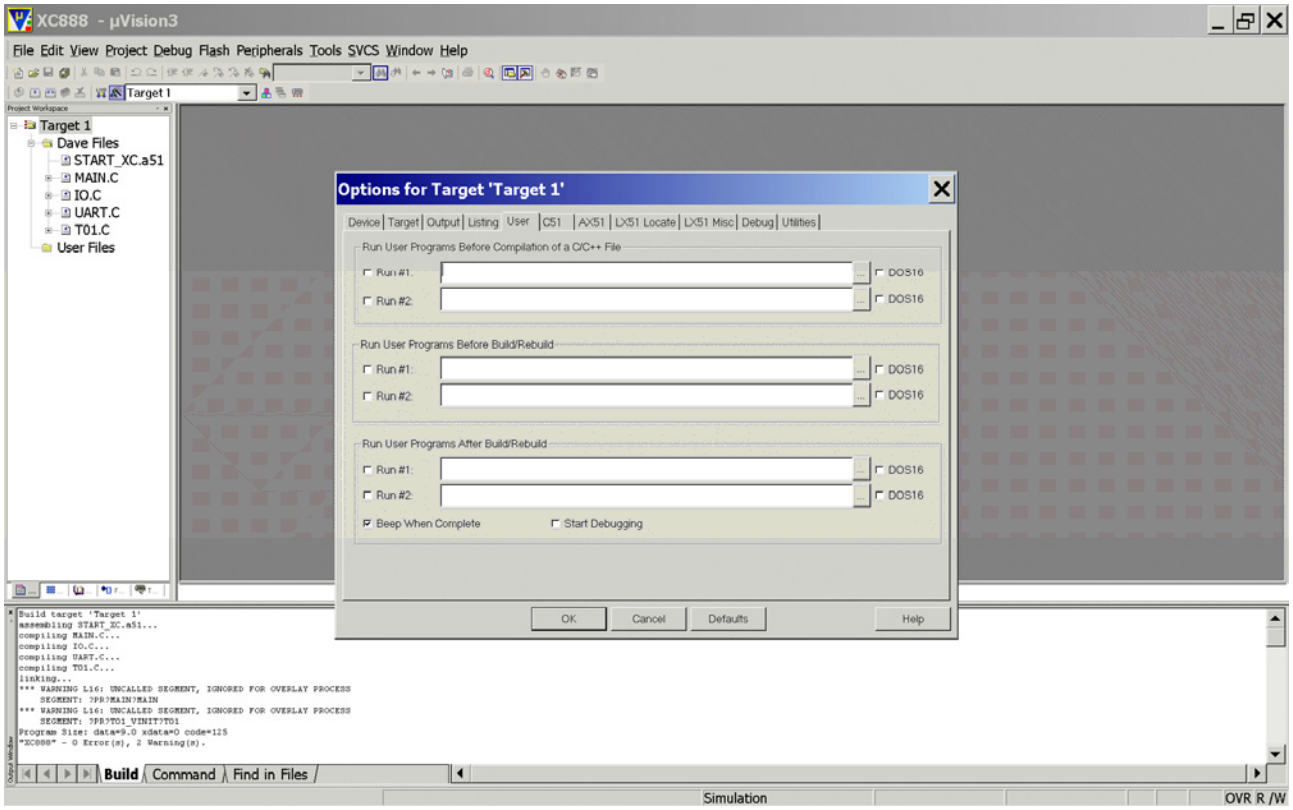
Note:
The HEX-File could be used while working with the program XC800_FLOAD for OnChip-Flash-Programming via RS232-interface [Bootstrap Loader (BSL) Mode via UART].

Listing: C Compiler Listing: click ✓ Assembly Code
Listing: Linker Listing: click ✓ Linker Code Listing: ./*.cod

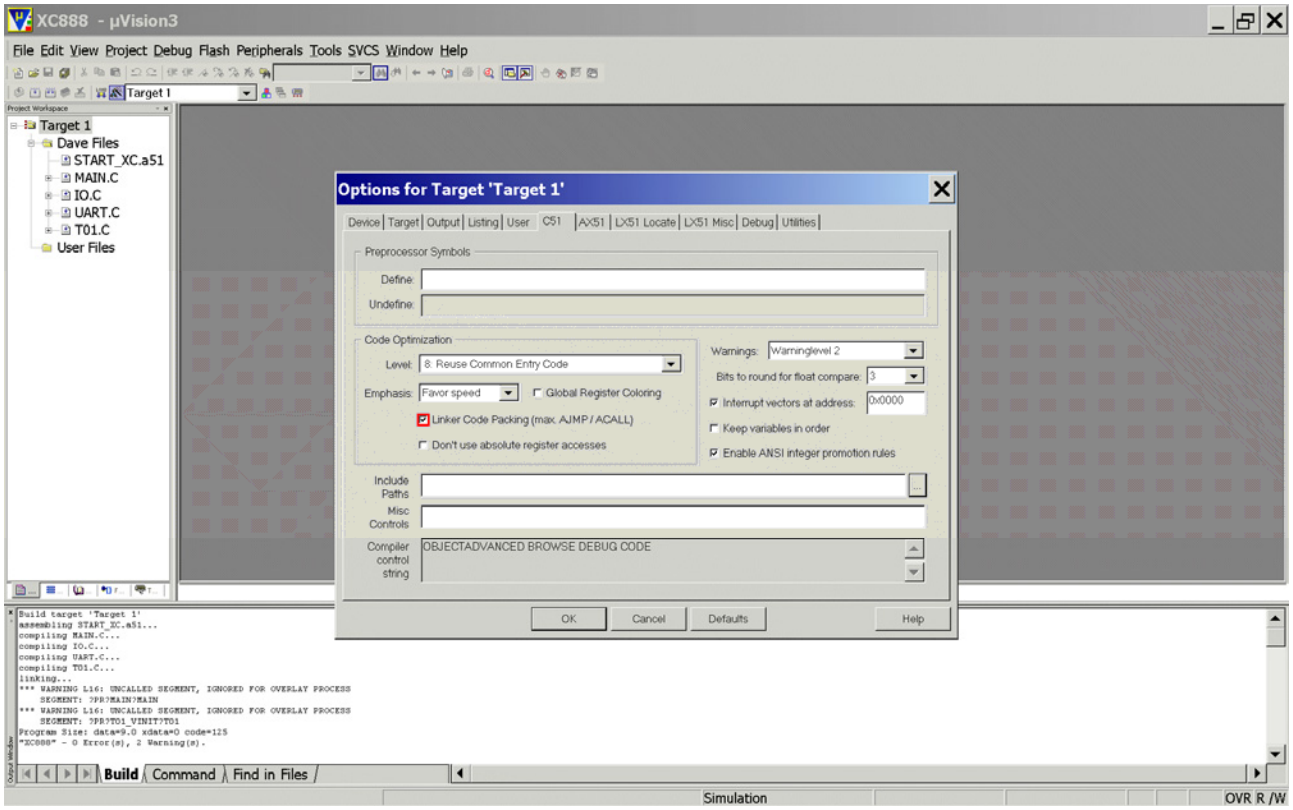


Note:
With the cod-file you can do the following:
1.) position the mouse on the source code you are interested in
2.) click right mouse button and select Open Linker COD File
3.) see the result: Assembler-Code of your C-Source-Code

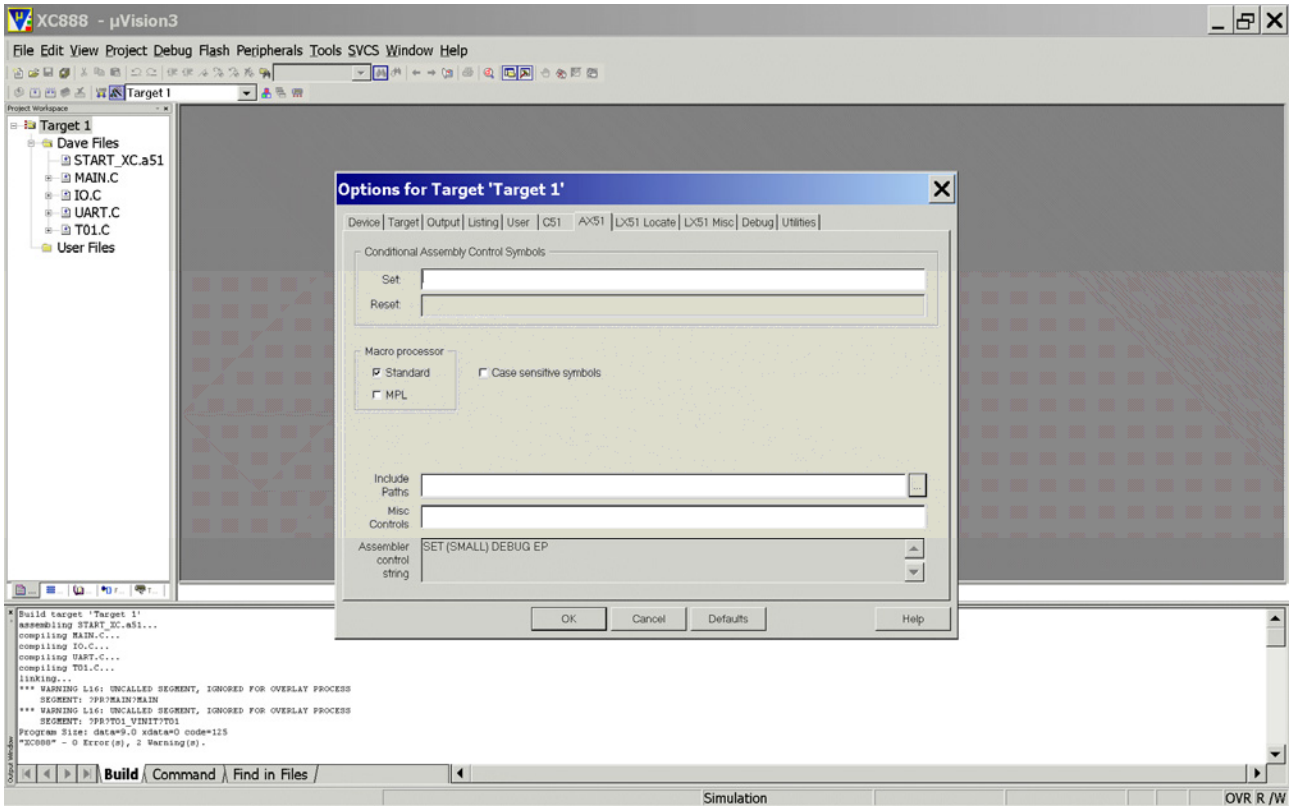
User: (do nothing)



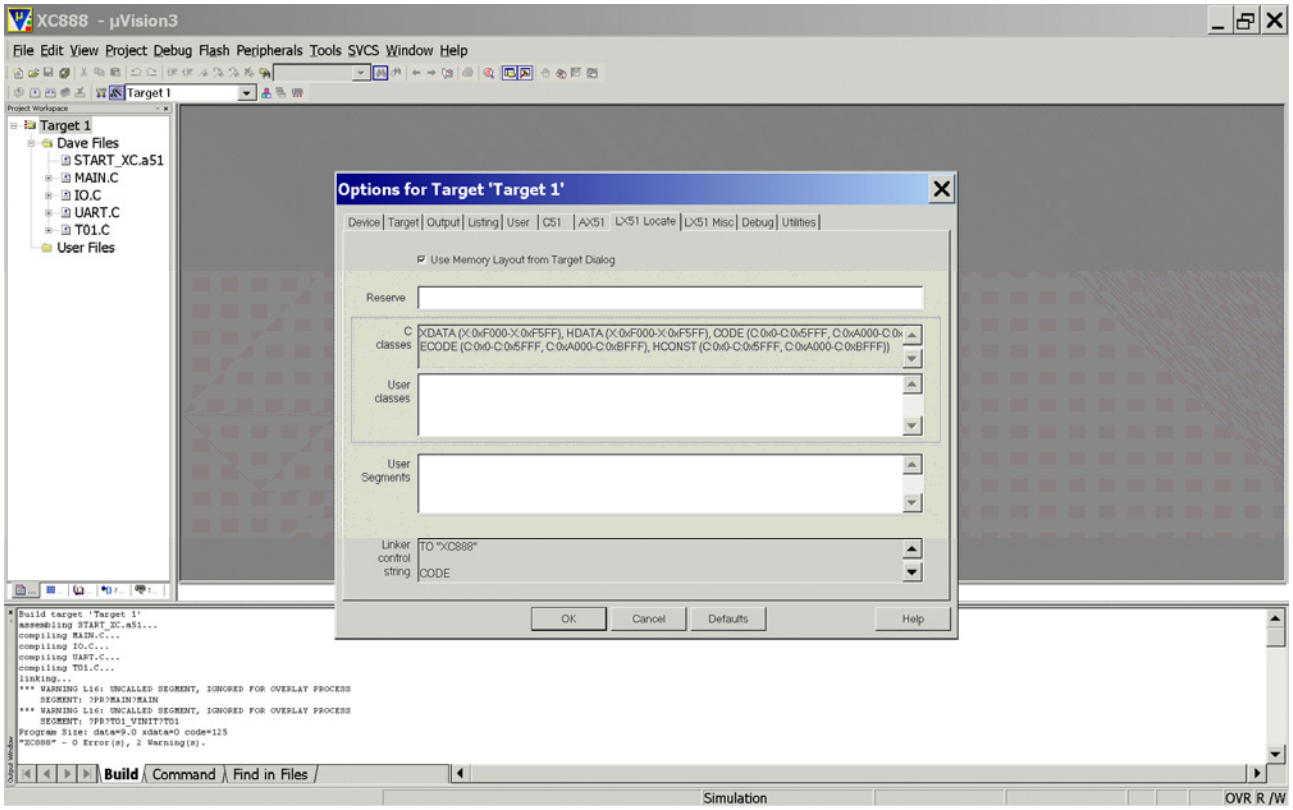
C51: Code Optimization: click ✓ Linker Code Packing (max. AJMP/ACALL)



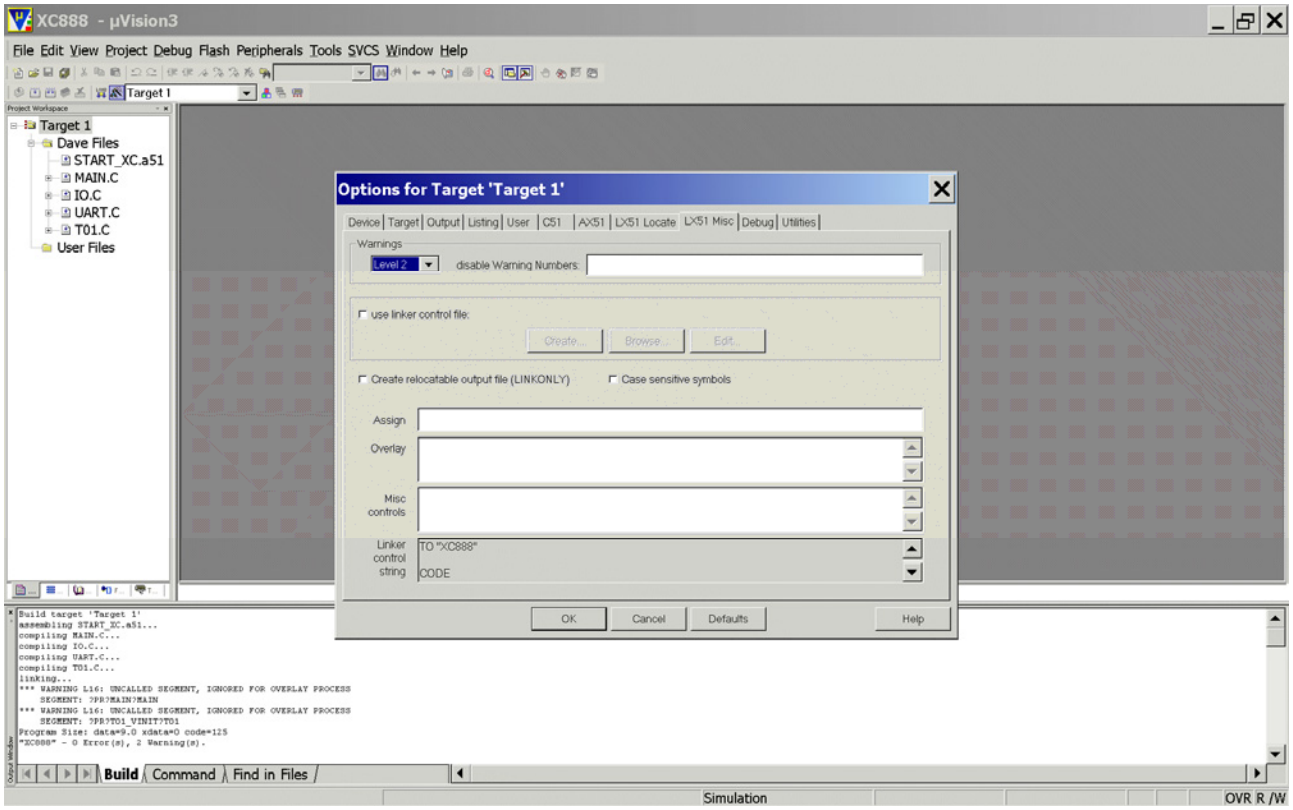
AX51: (do nothing)



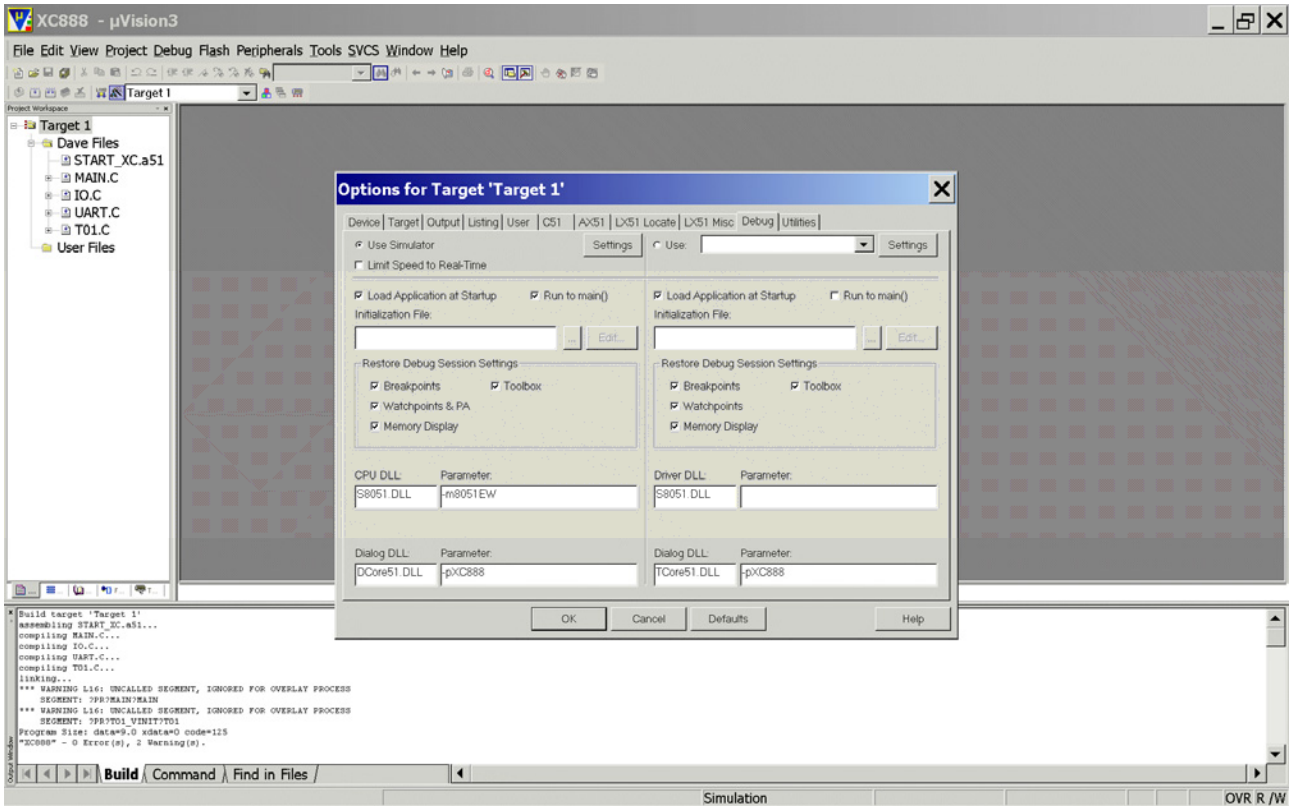
LX51 Locate: (do nothing)

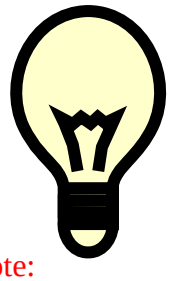


LX51 Misc: (do nothing)



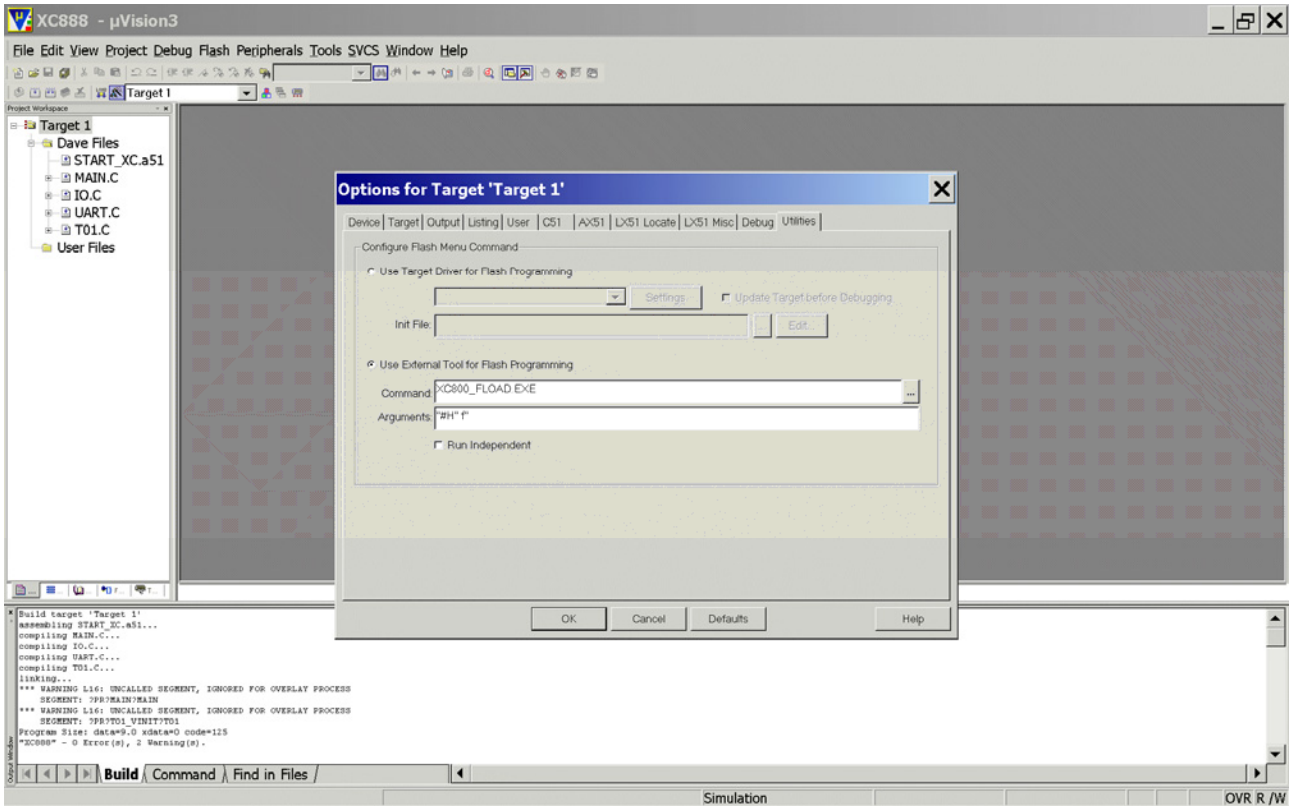
Debug: (do nothing)





Note:
First we are going to use the simulator (we will use the debugger later).

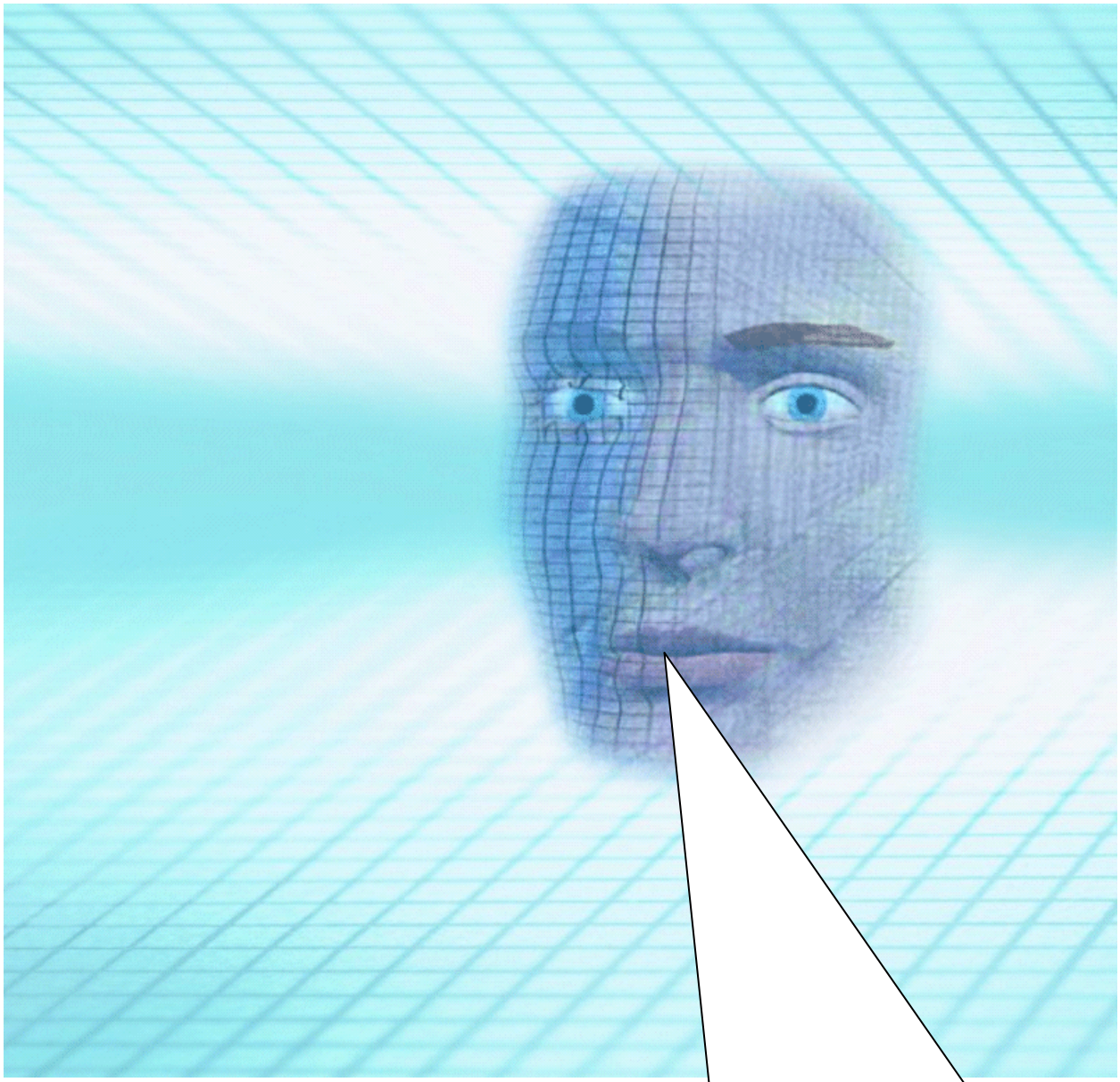
Utilities: (do nothing)



Click OK

Note:
First we are going to use the simulator (we will do the flash programming later).

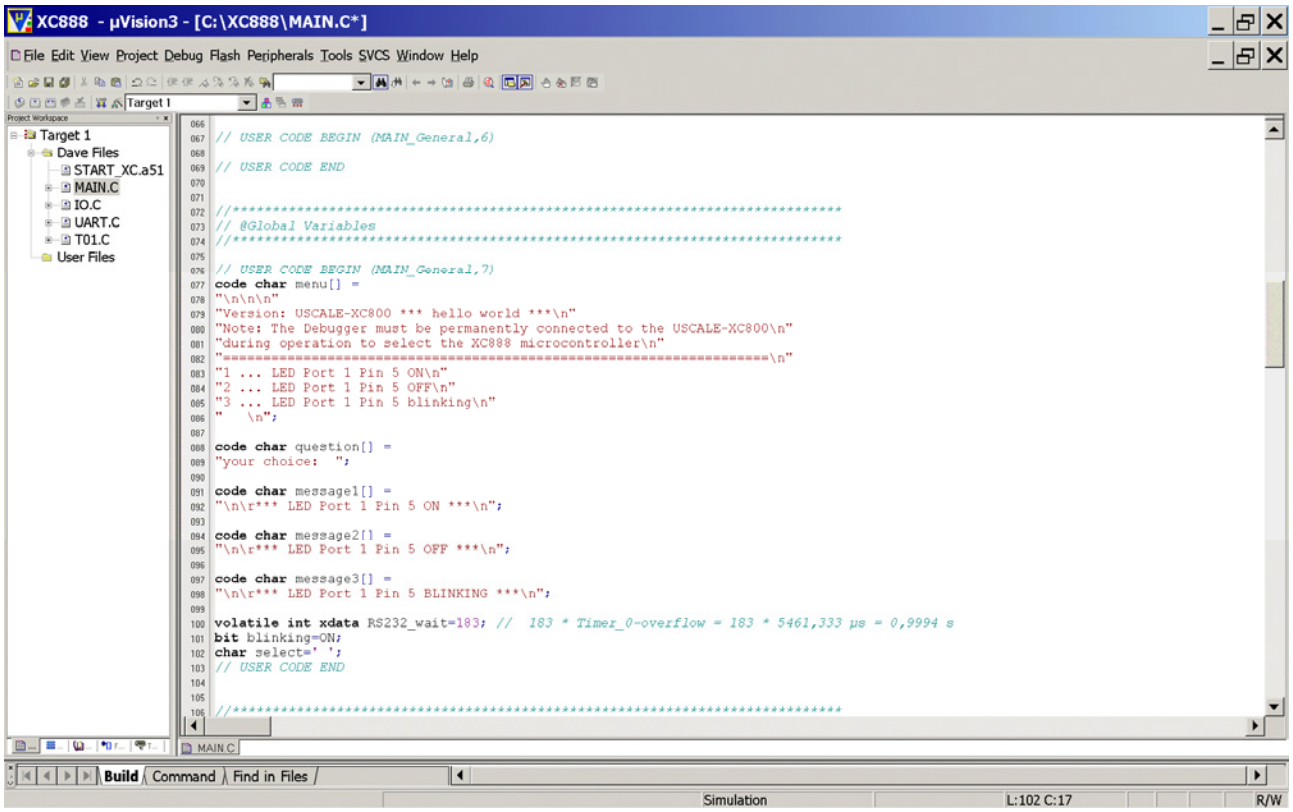
Insert your application specific program:



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!

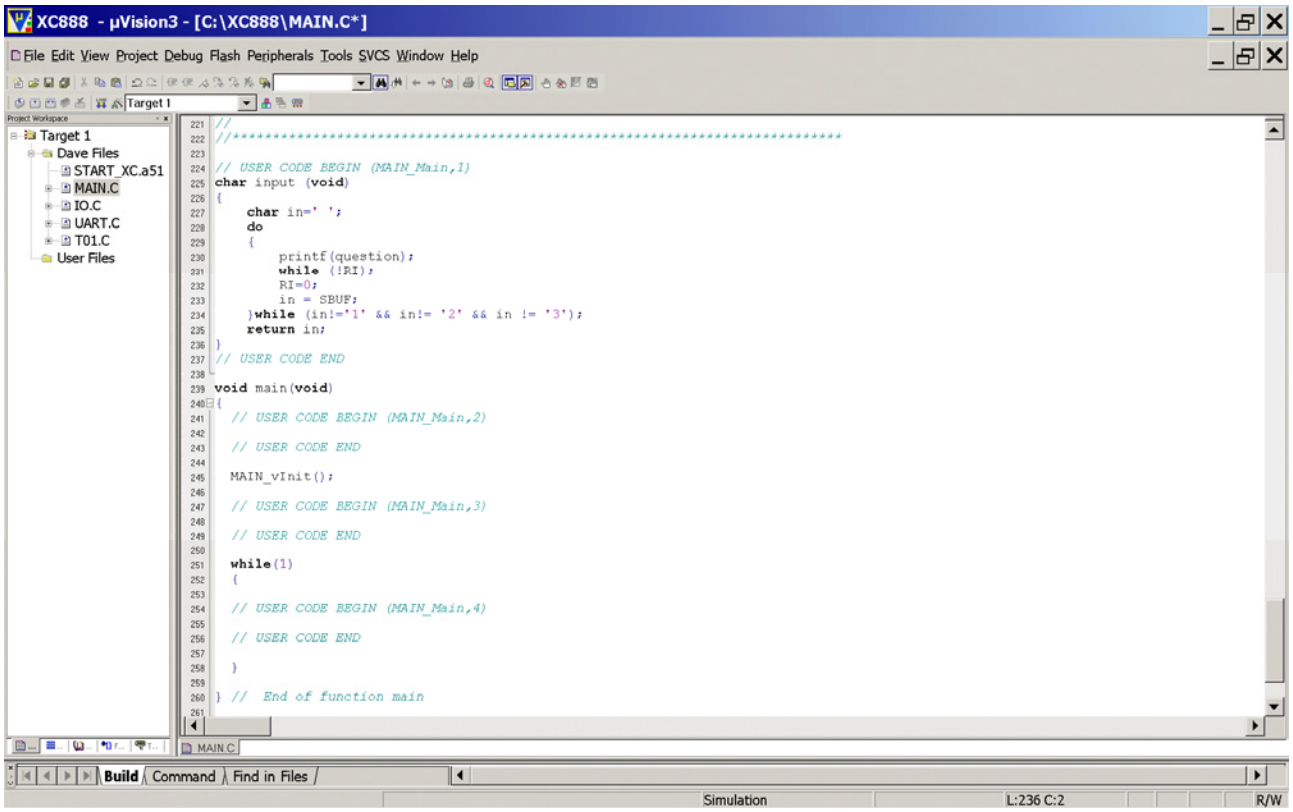
Double click MAIN.C and insert Global Variables:

```
code char menu[] =
"\n\n\n"
"Version: USCALE-XC800 *** hello world ***\n"
"Note: The Debugger must be permanently connected to the USCALE-XC800\n"
"during operation to select the XC888 microcontroller\n"
"================================================================
\n"
"1 ... LED Port 1 Pin 5 ON\n"
"2 ... LED Port 1 Pin 5 OFF\n"
"3 ... LED Port 1 Pin 5 blinking\n"
"   \n";

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

code char message1[] =
"\n\r*** LED Port 1 Pin 5 ON ***\n";

code char message2[] =
"\n\r*** LED Port 1 Pin 5 OFF ***\n";

code char message3[] =
"\n\r*** LED Port 1 Pin 5 BLINKING ***\n";

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


XC888 - µVision3 - [C:\XC888\MAIN.C*]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Dave Files
START_XC.a51
MAIN.C
IO.C
UART.C
T01.C
User Files
// USER CODE BEGIN (MAIN_General,6)
// USER CODE END
//***
// @Global Variables
//***
// USER CODE BEGIN (MAIN_General,7)
code char menu[] =
"\n\n\n"
"Version: USCALE-XC800 *** hello world ***\n"
"Note: The Debugger must be permanently connected to the USCALE-XC800\n"
"during operation to select the XC888 microcontroller\n"
"==\n"
"1 ... LED Port 1 Pin 5 ON\n"
"2 ... LED Port 1 Pin 5 OFF\n"
"3 ... LED Port 1 Pin 5 blinking\n"
"   \n";
code char question[] =
"your choice: ";
code char message1[] =
"\n\r*** LED Port 1 Pin 5 ON ***\n";
code char message2[] =
"\n\r*** LED Port 1 Pin 5 OFF ***\n";
code char message3[] =
"\n\r*** LED Port 1 Pin 5 BLINKING ***\n";
volatile int xdata RS232_wait=183; // 183 * Timer_0-overflow = 183 * 5461,333 µs = 0,9994 s
bit blinking=ON;
char select=' ';
// USER CODE END
//***
MAIN.C
Build
Command
Find in Files
Simulation
L:102 C:17
R/W

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;
}
```

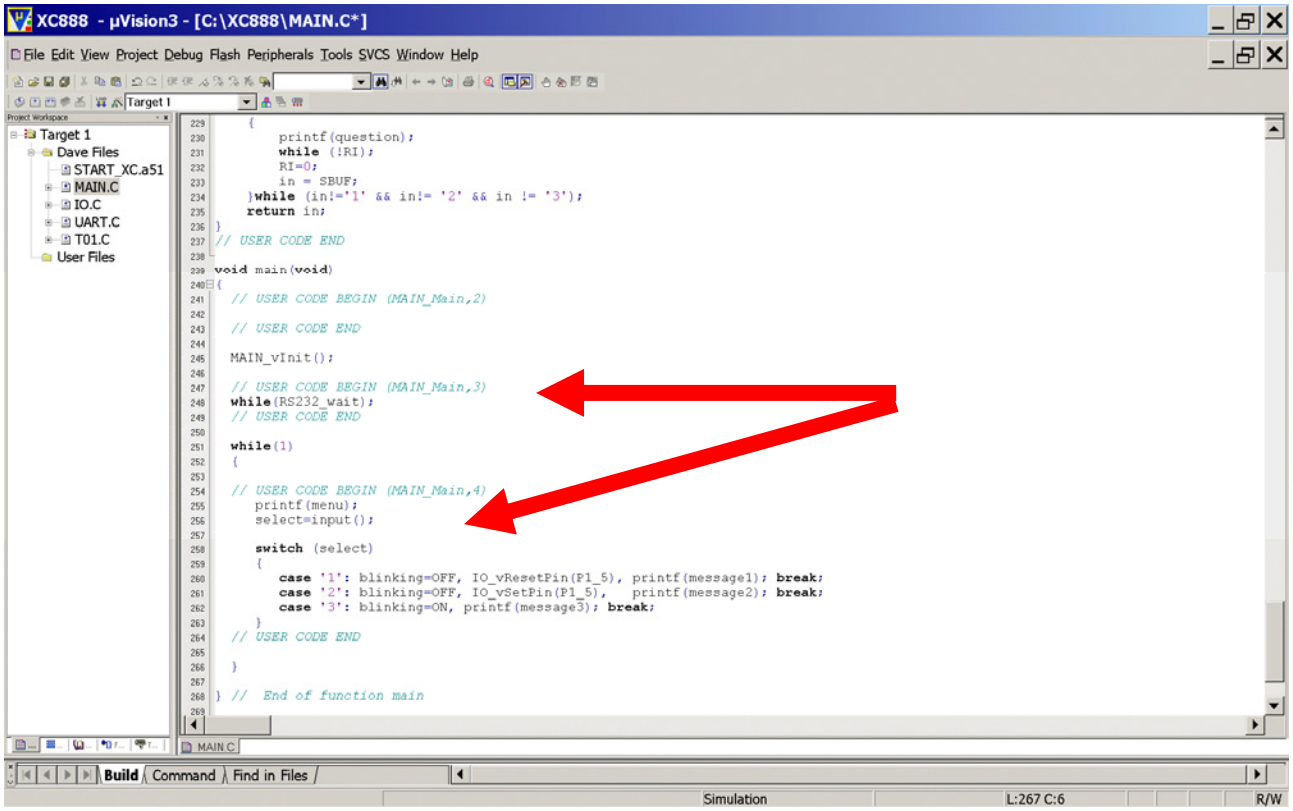


Double click MAIN.C and insert the following code in the main function:

```
        while(RS232_wait);
```

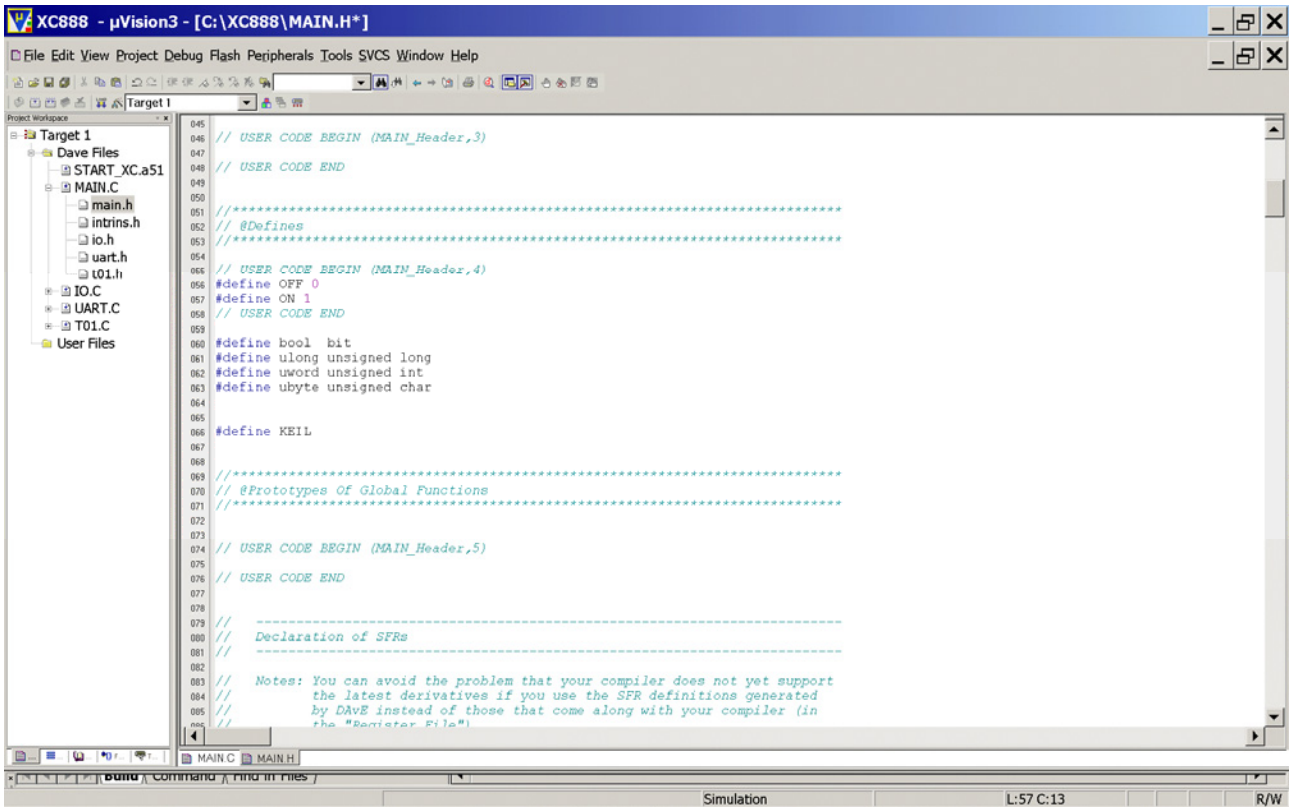
Double click MAIN.C and insert the following code in the main function into the while(1) loop:

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

switch (select)
{
  case '1': blinking=OFF, IO_vResetPin(P1_5), printf(message1); break;
  case '2': blinking=OFF, IO_vSetPin(P1_5),   printf(message2); break;
  case '3': blinking=ON, printf(message3); break;
}
```



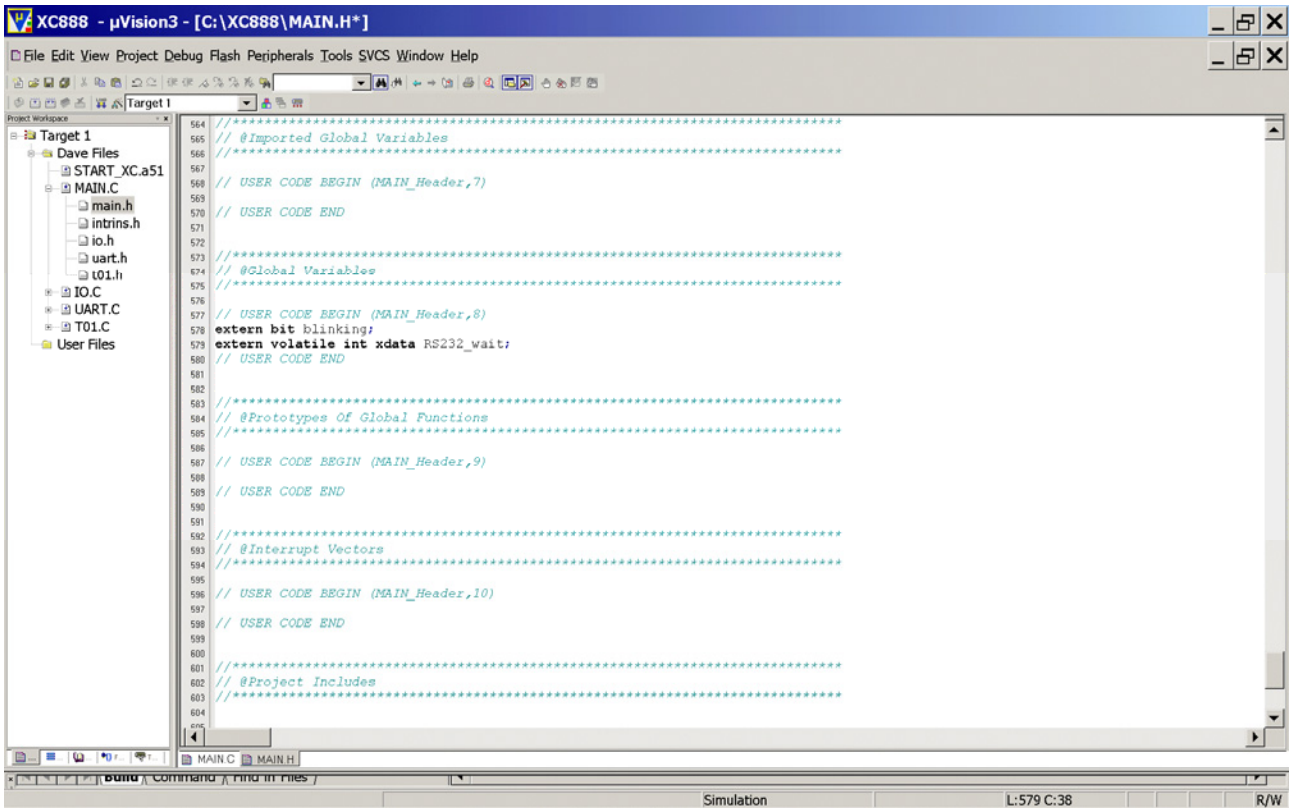
Double click Main.h and insert the following Defines:

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



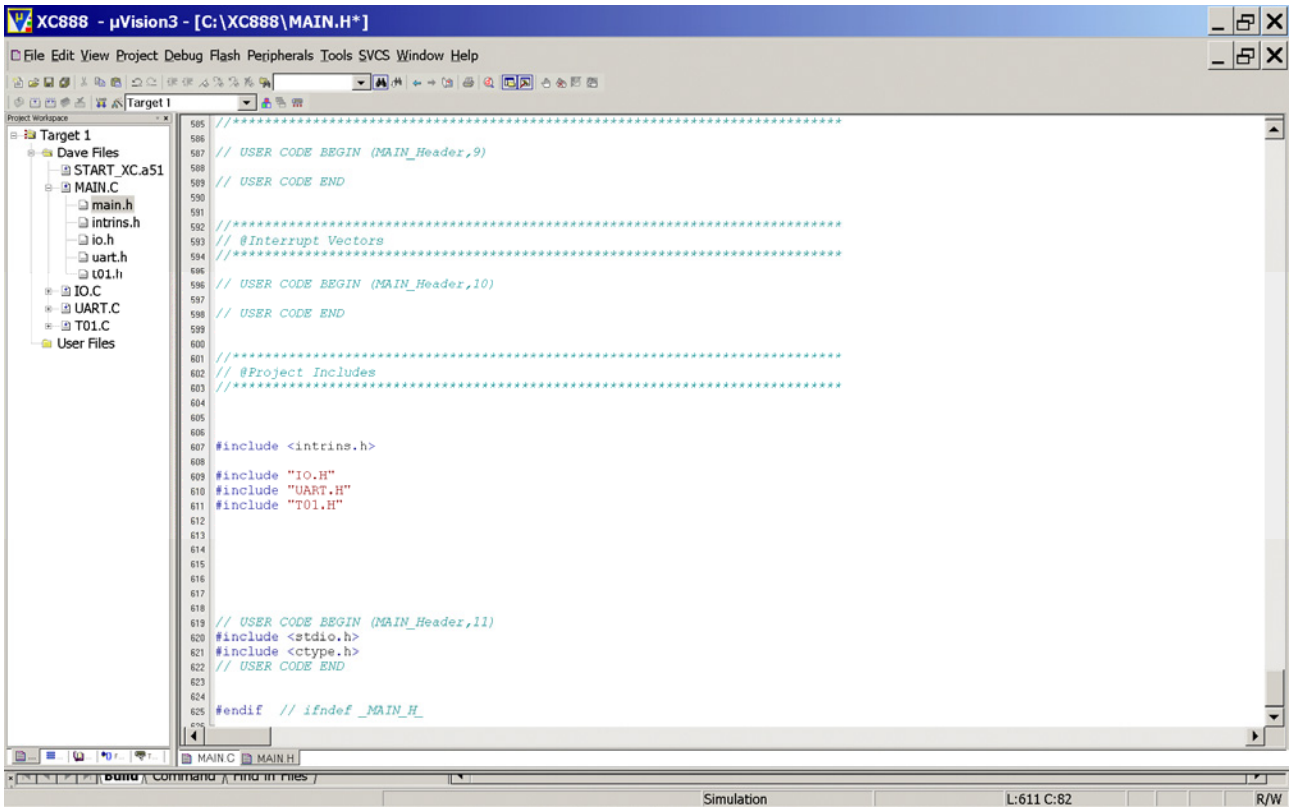
Double click Main.h and insert extern-declarations “Global Variables”:

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



Double click Main.h and insert include files:

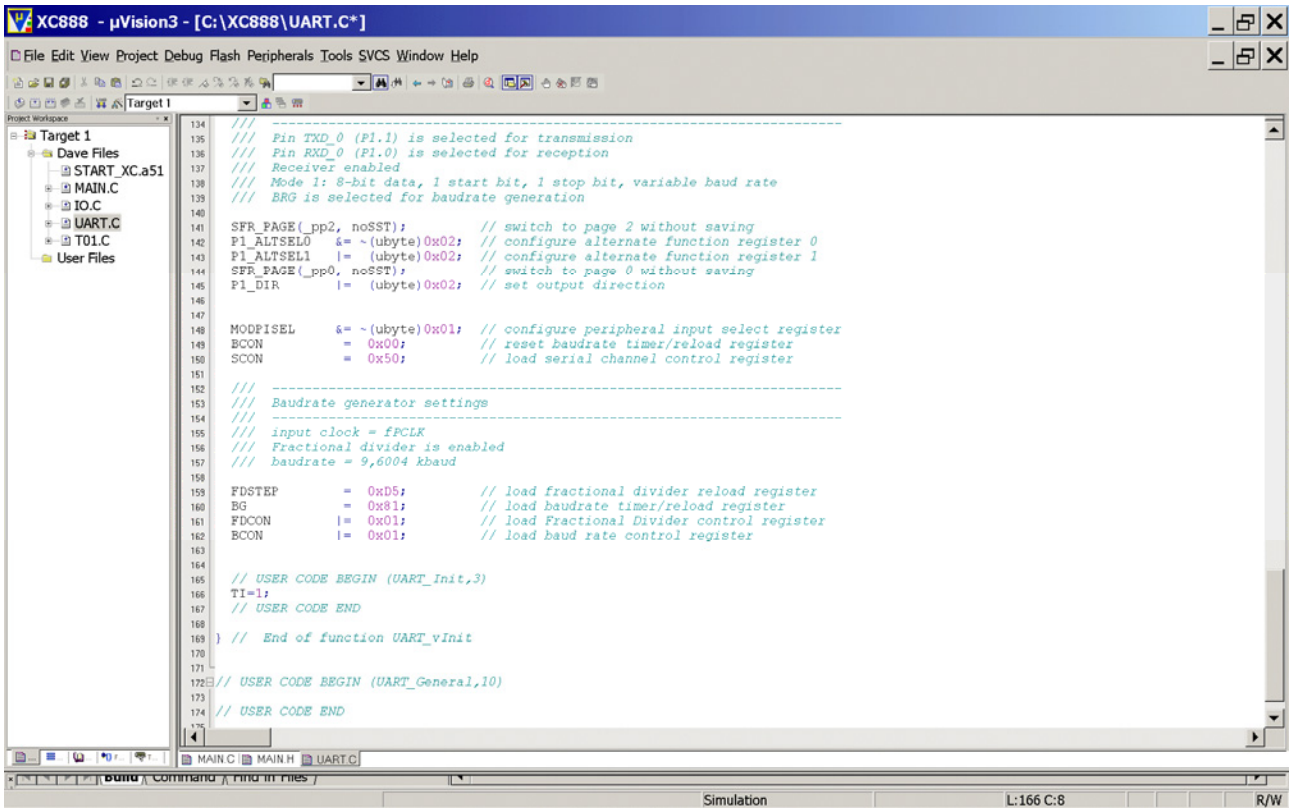
```
#include <stdio.h>
#include <ctype.h>
```



Double click UART.C

Insert code into the UART_vInit function: [to start printf()]:

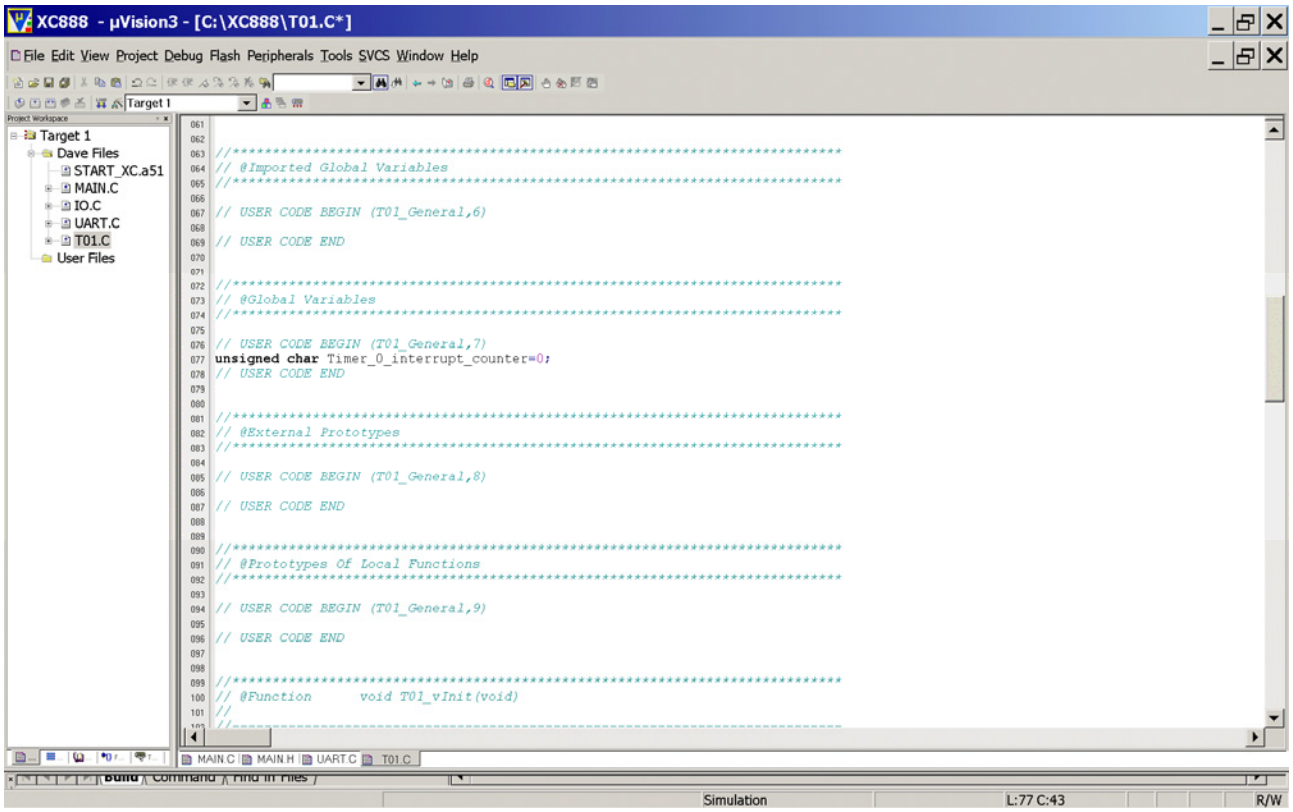
```
TI=1;
```



Double click T01.C

Insert the following global variable:

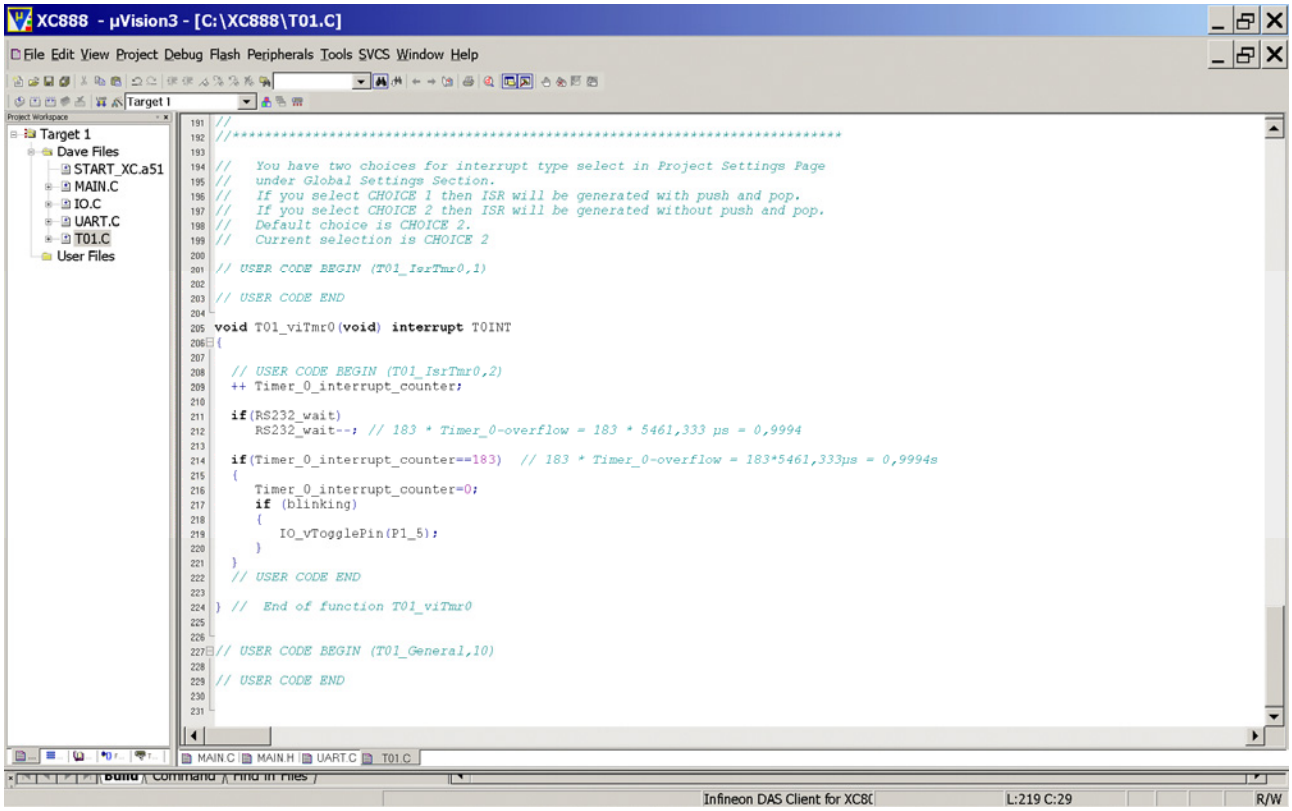
```
unsigned char Timer_0_interrupt_counter=0;
```



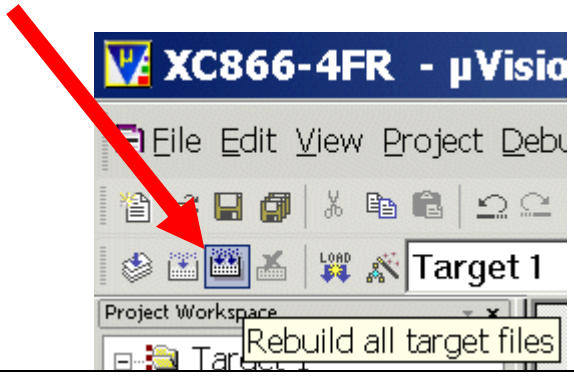
Double click T01.C

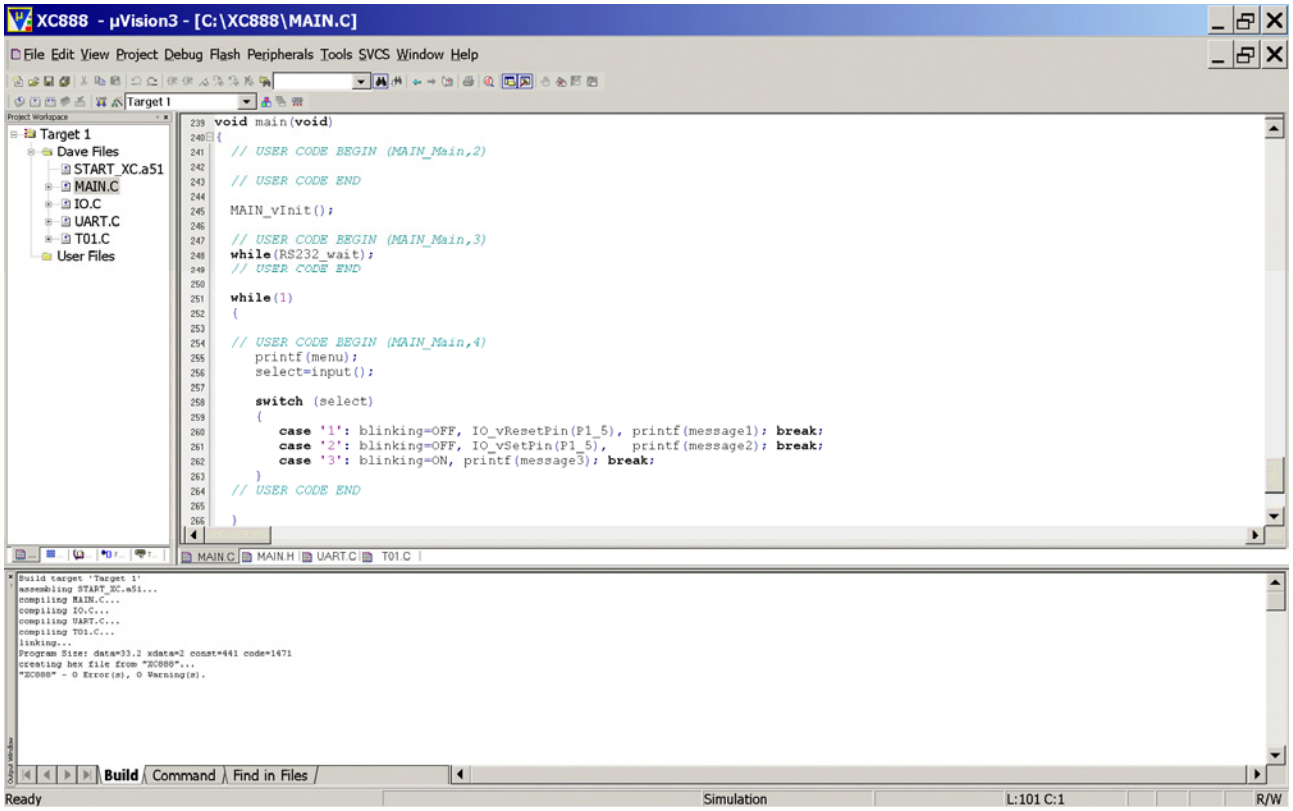
Insert code for T0 interrupt service routine:

```
++ 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)
      {
          IO_vTogglePin(P1_5);
       }
 }
```



Generate your application program:

Project – Rebuild all target files	or	click 



Now we close our project and µVision 3:

Project
Close Project

File

Exit

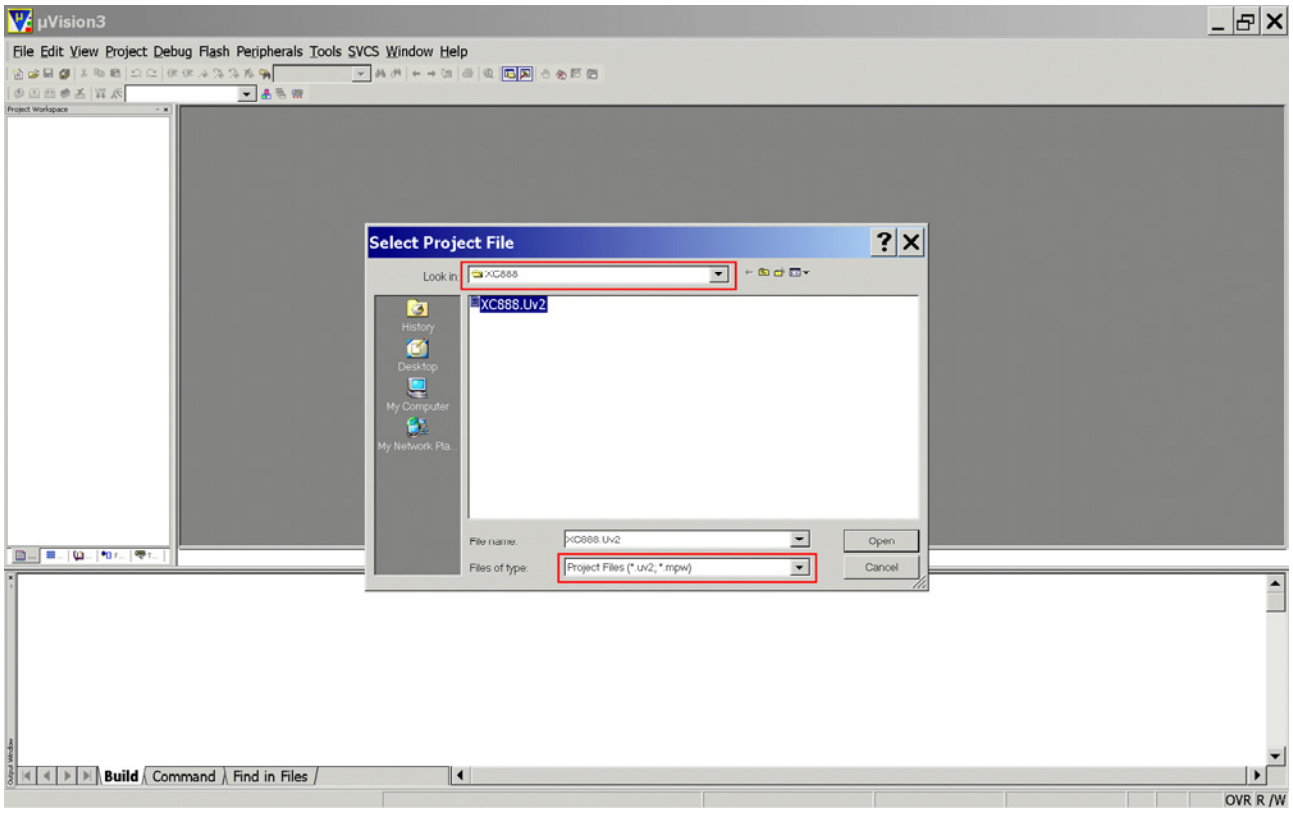
5.) Using the Simulator (first we will test our program with the Keil Simulator):

Start Keil µVision 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: select Project Files (*.uv2)
click XC888.Uv2

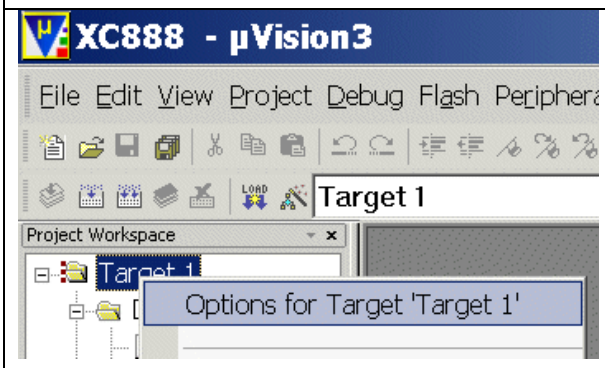
click Open

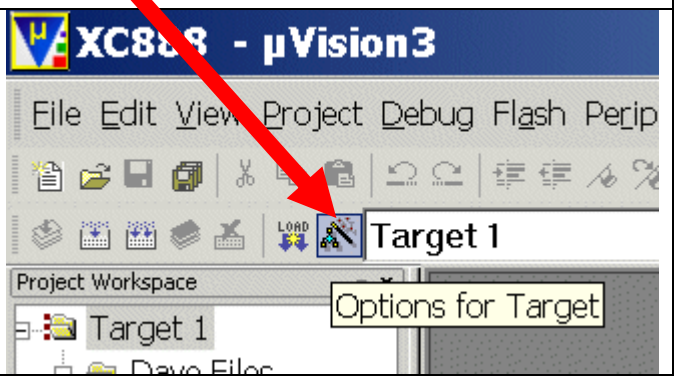


Note:
From now on just open your µVision project (not the DAvE project). µVision will automatically recognise if there has been a code regeneration done by DAvE!

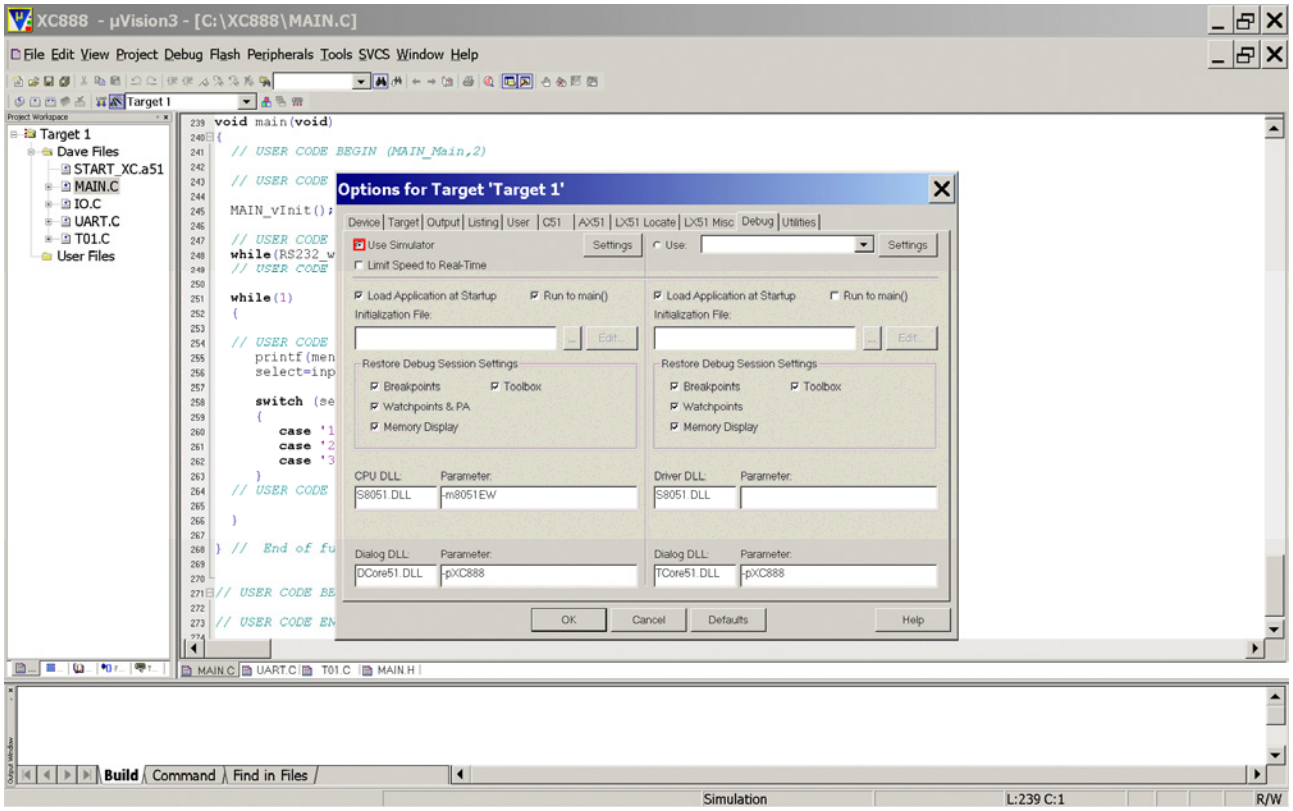
Check the configuration of the µVision simulator:

mouse position: (Project Workspace, Files): Target1 click right mouse button Options for Target ´Target1‘	or	click



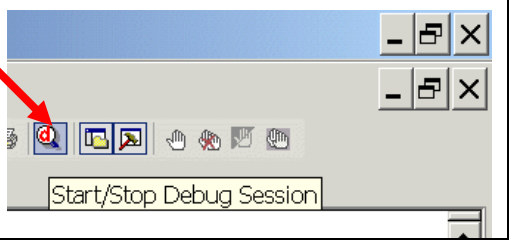


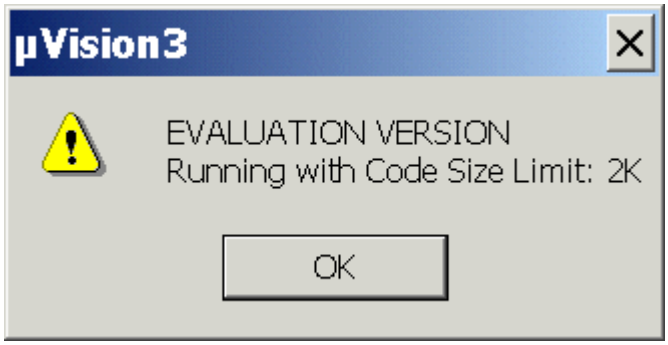
Options for Target ´Target1‘: Debug: check ⊙ Use Simulator



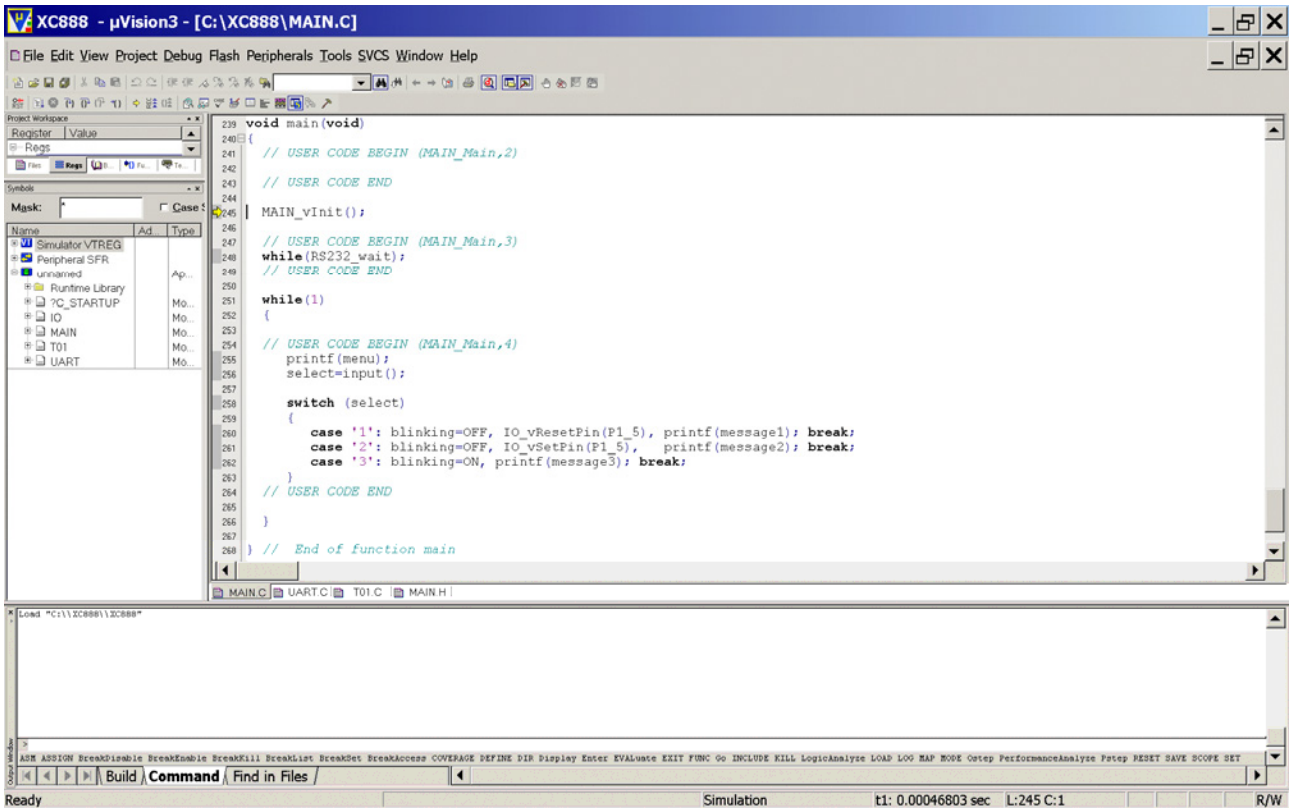
OK

Start the µVision Simulator:

Debug - Start/Stop Debug Session	or	click 

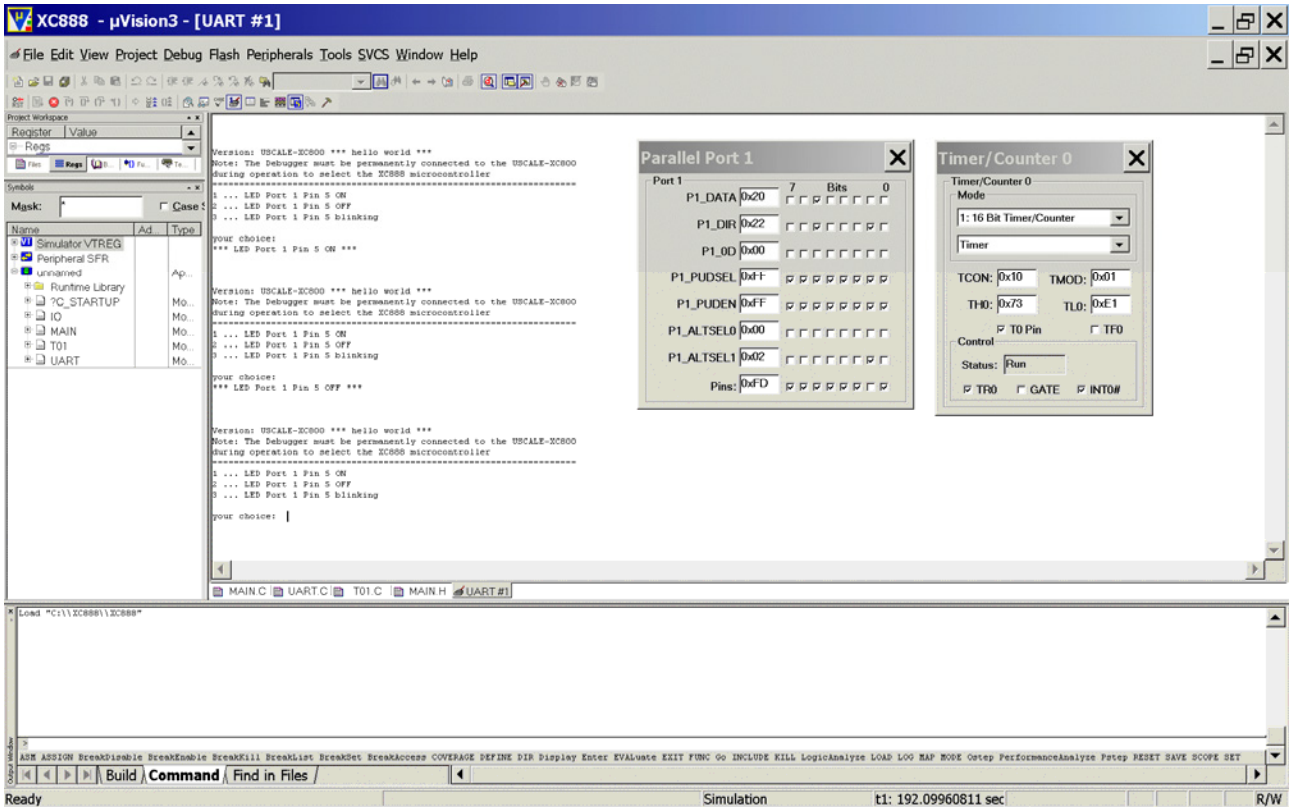


OK



Start program-execution:

Debug – Run
View - Serial Window – UART #1
Peripherals - I/O-Ports – Port1
Peripherals – Timer – Timer0



Note:
By activating (clicking) the **UART #1**-window you can then type 1, 2 or 3 and see the result in the Parallel-Port-1-window (Pin 5).

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

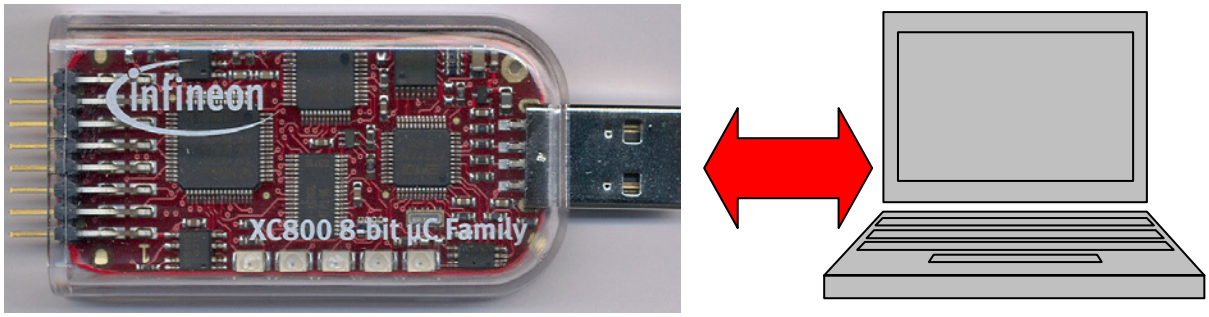
File – Exit

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

6.) Using real hardware:

(+ OnChipFlash-Programming)

Make sure that the XC800 USCALE start kit is still connected to the host computer:



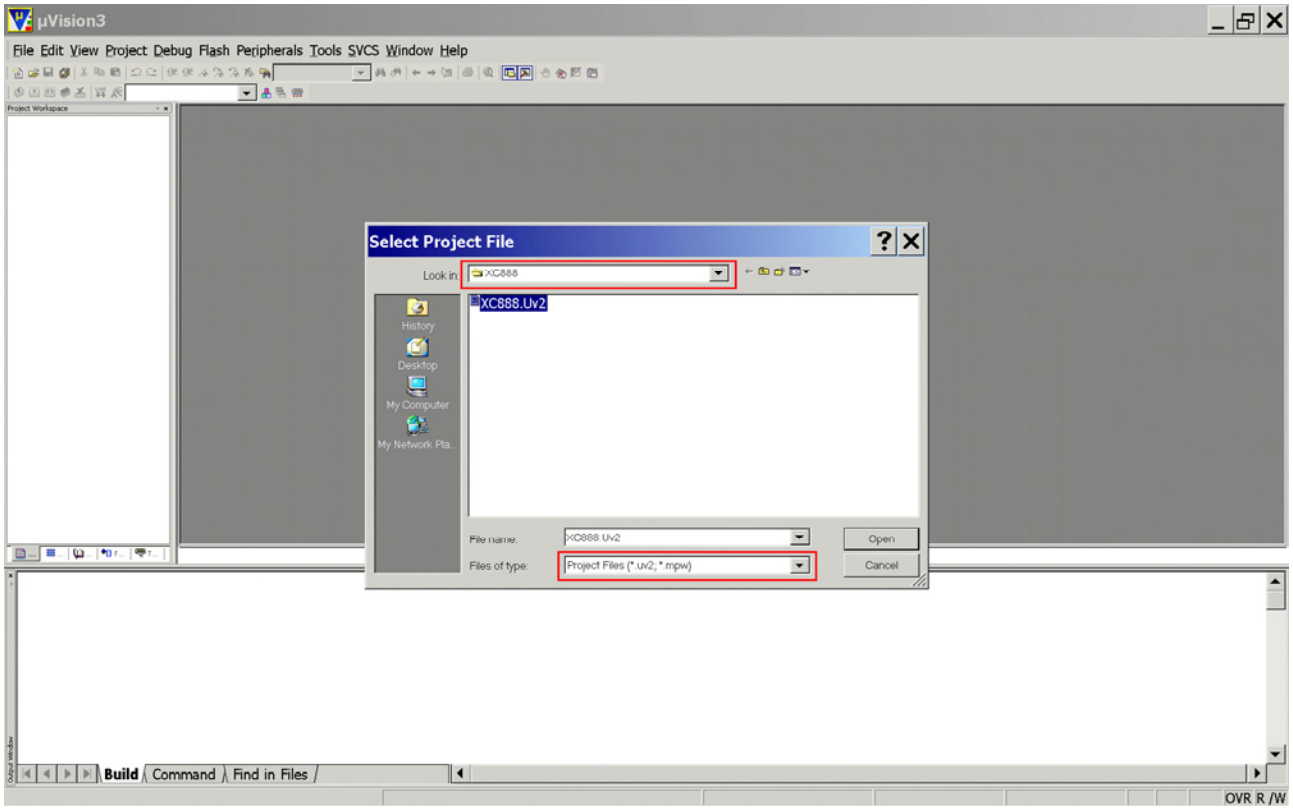
USB Connection:
.) used for: UART communication (the UART/RS232/serial interface is available via USB as a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
.) used for: On-Chip-Flash-Programming and Debugging (first USB channel of the FTDI FT2232 Dual USB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

Start Keil µVision and open our Keil Project

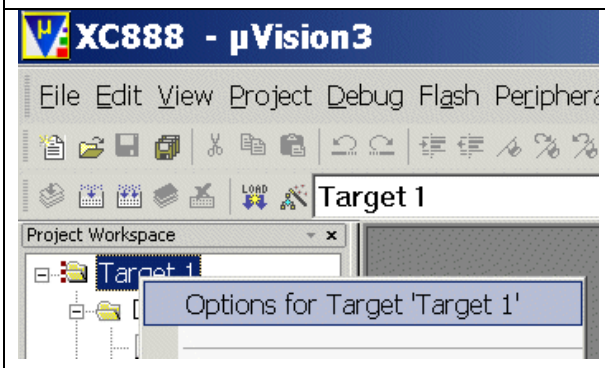
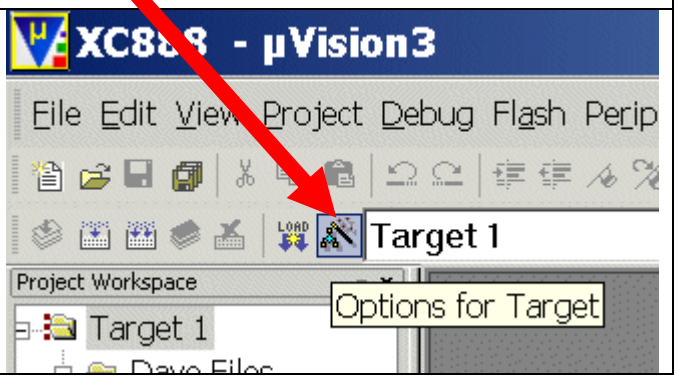
If you see an open project – close it: Project - Close Project

Project - Open Project
Select Project File: Look in: choose C:\XC888
Select Project File: Files of type: select Project Files (*.uv2)
click XC888.Uv2

click Open

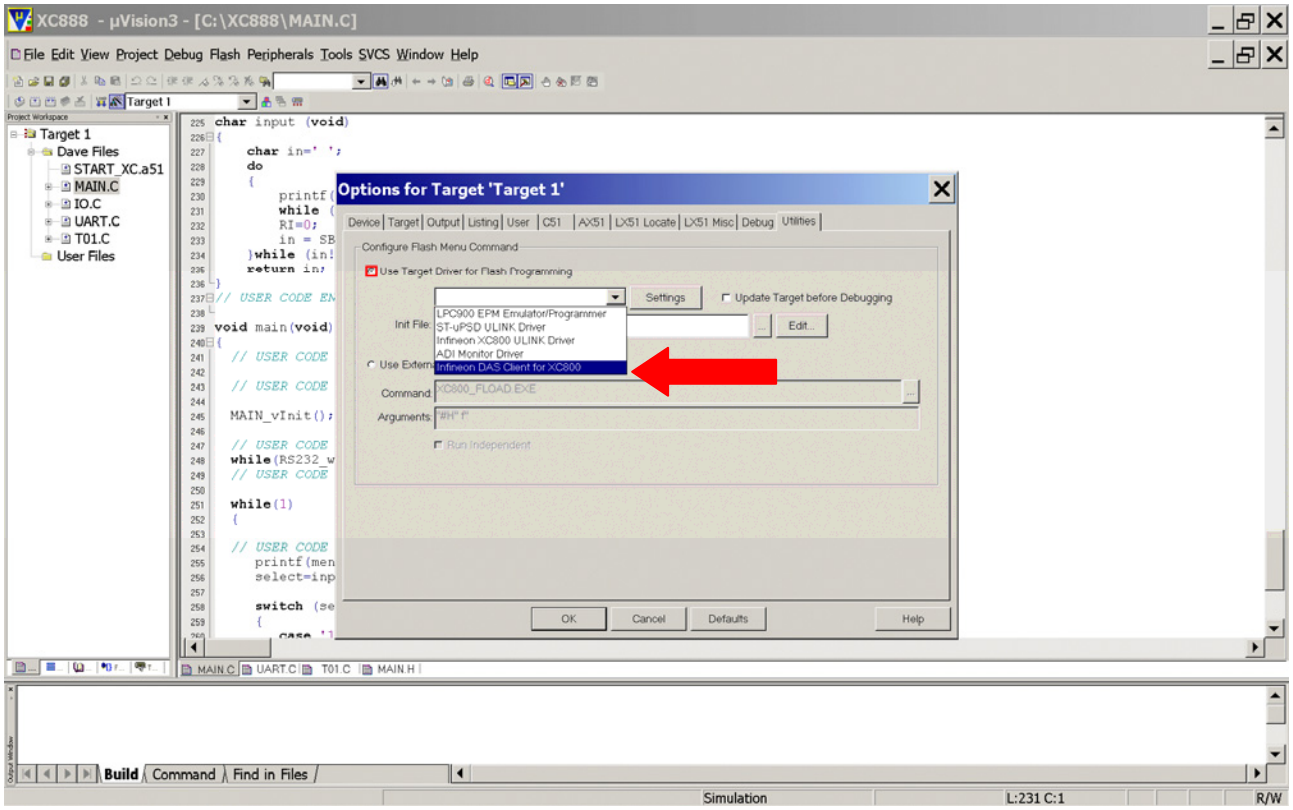


Check/configure the configuration of the Flash-Programming-Utility and select the correct microcontroller (XC888) on the XC800 USCALE start kit:

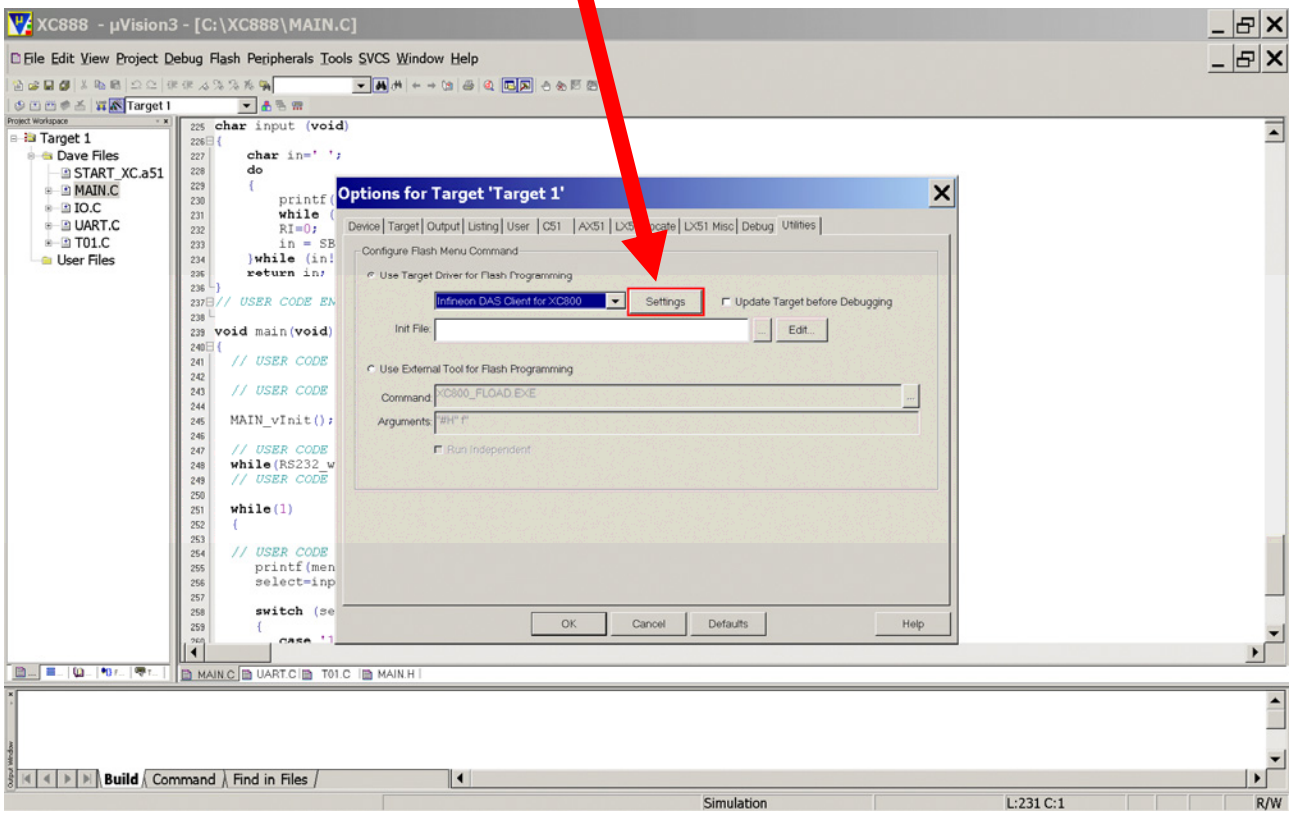
mouse position: (Project Workspace, Files): Target1 click right mouse button Options for Target ´Target1‘	or	click
		

Options for Target ´Target1‘:Utilities:
Configure Flash Menu Command: check/click ⊙ Use Target Driver for Flash Programming

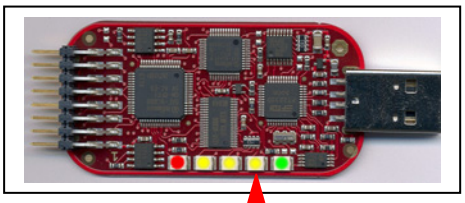
Options for Target ´Target1‘: Utilities:
Configure Flash Menu Command: check/select Infineon DAS Client for XC800



Options for Target ´Target1‘: Utilities:
Configure Flash Menu Command: click Settings

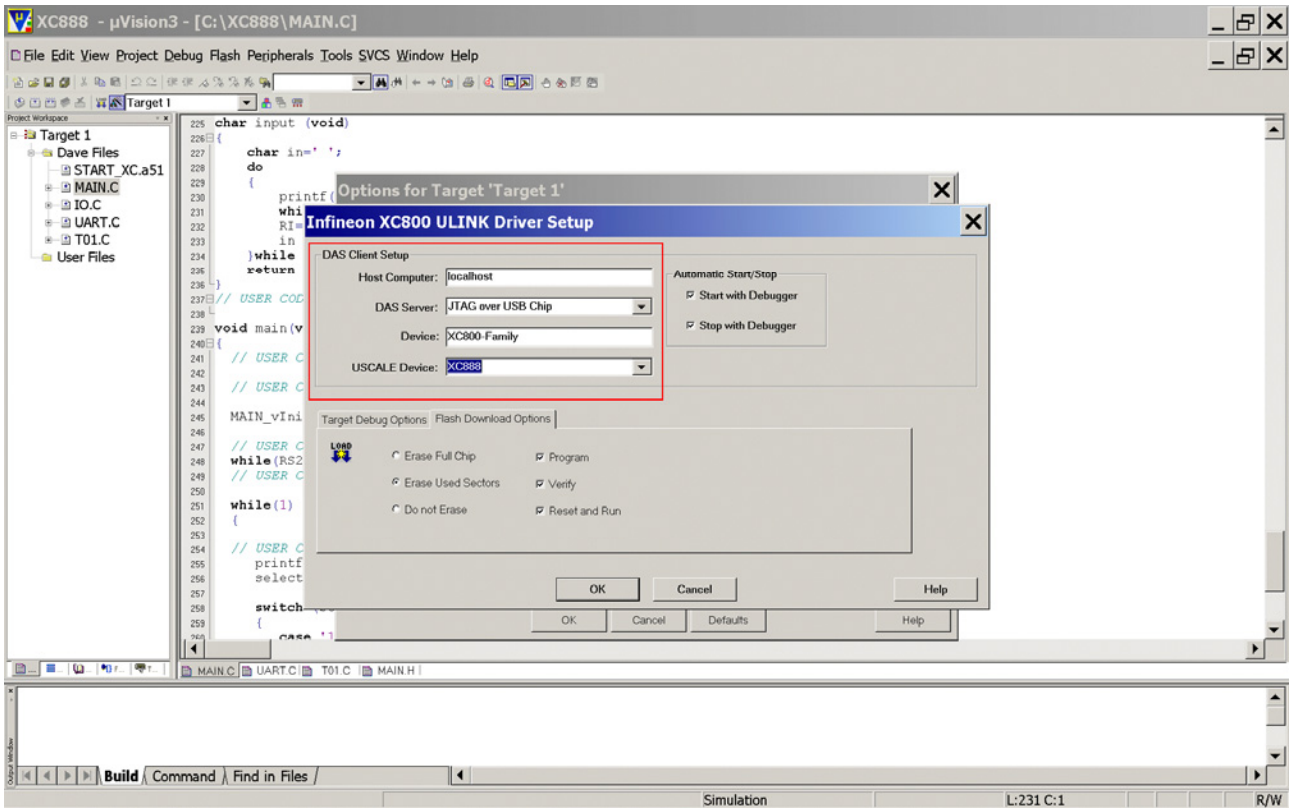


Infineon XC800 ULINK Driver Setup:
DAS Client Setup:
DAS Server: select JTAG over USB Chip
USCALE Device: select XC888



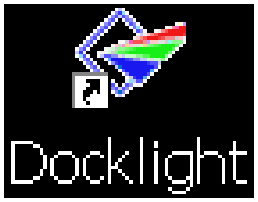
XC888

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

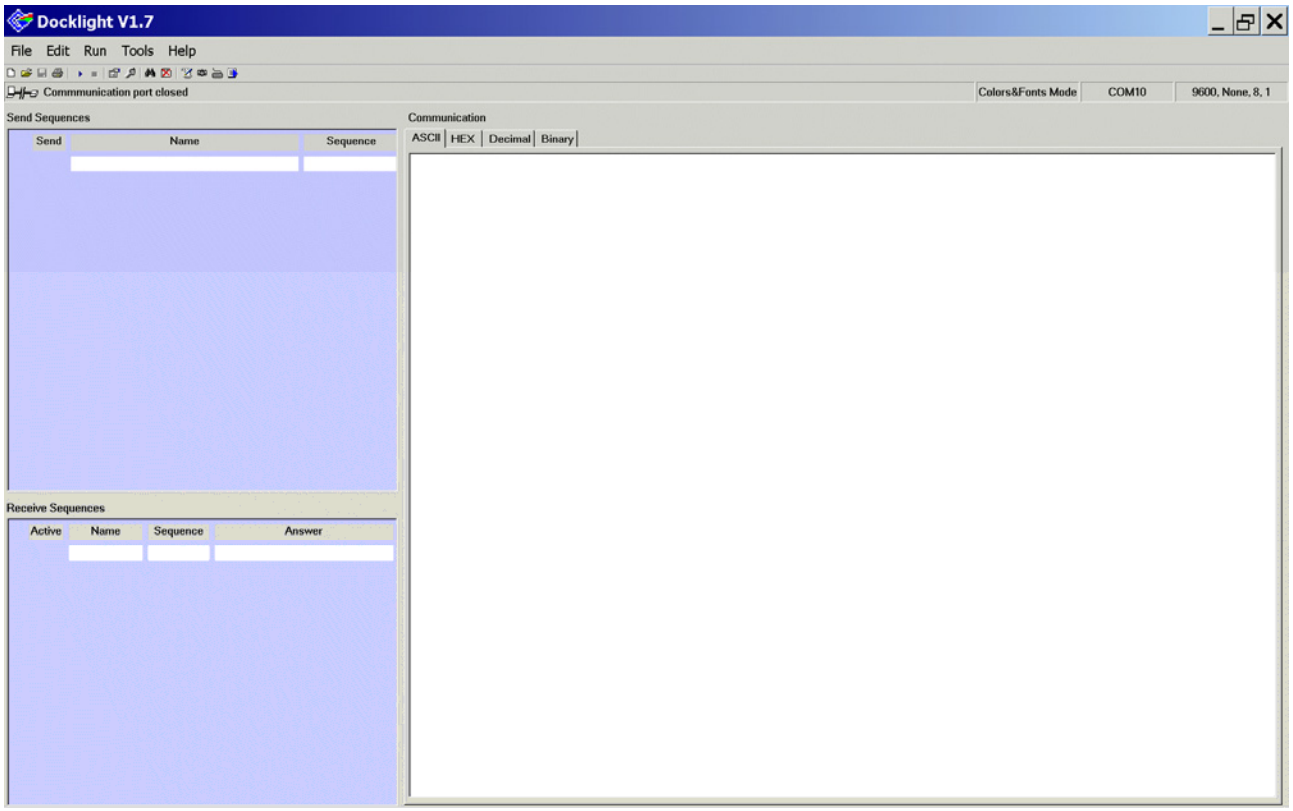


Click OK
Click OK

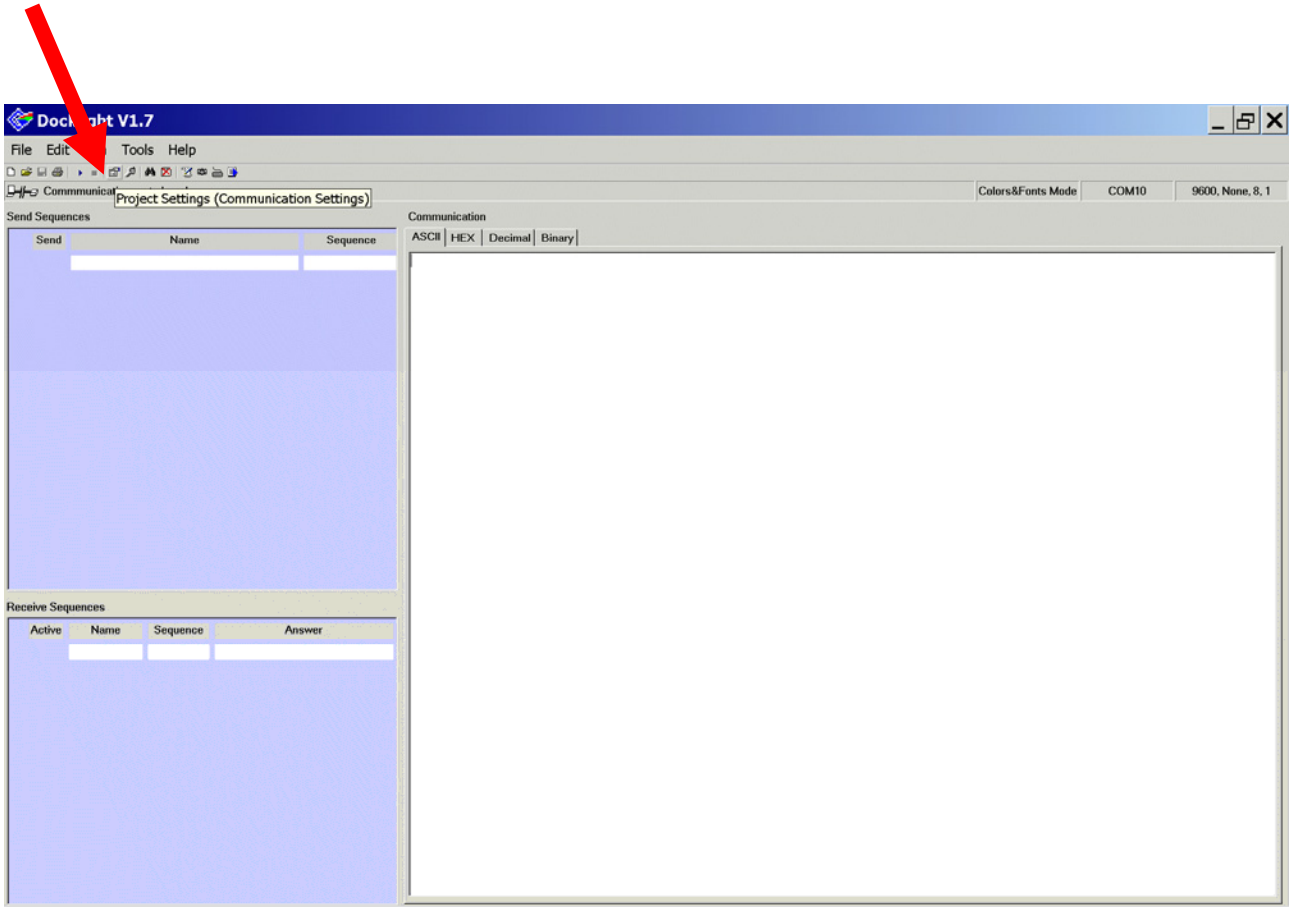
Note:
Now we need a terminal program which is able to handle our virtual COM port (COM5)!
As an example of “any terminal program” we are going to use Docklight.
Docklight can be downloaded @ http://www.docklight.de .



Now, start Docklight:



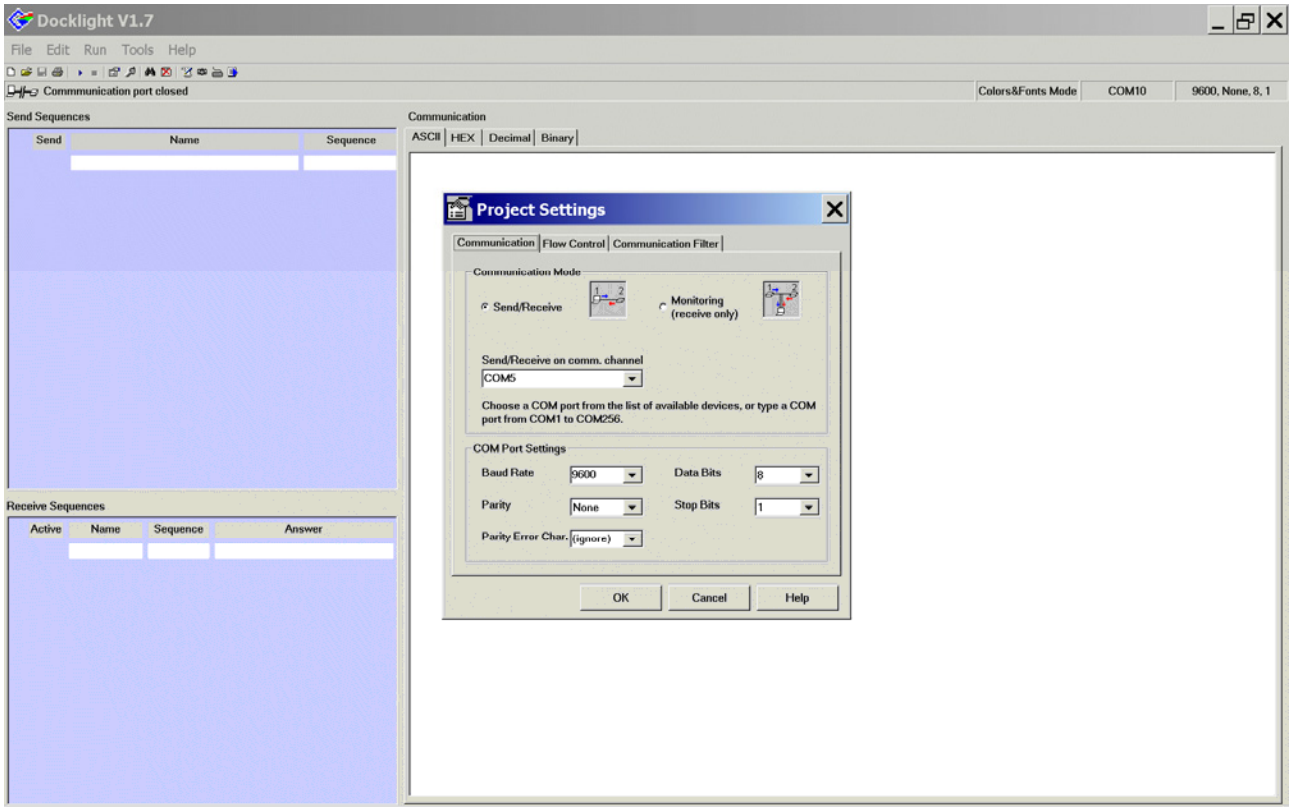
Click: Project Settings



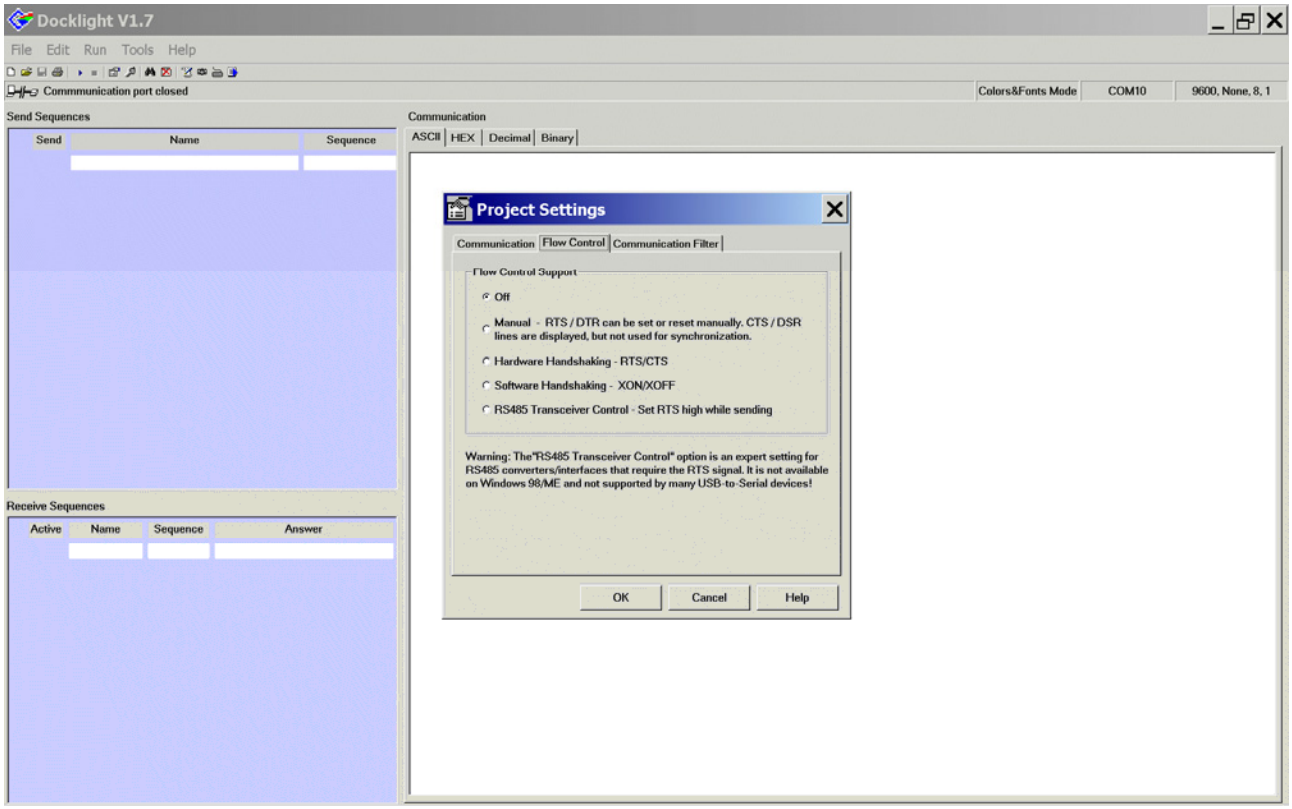
Project Settings:
Communication: Communication Mode: click ⊙ Send/Receive

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

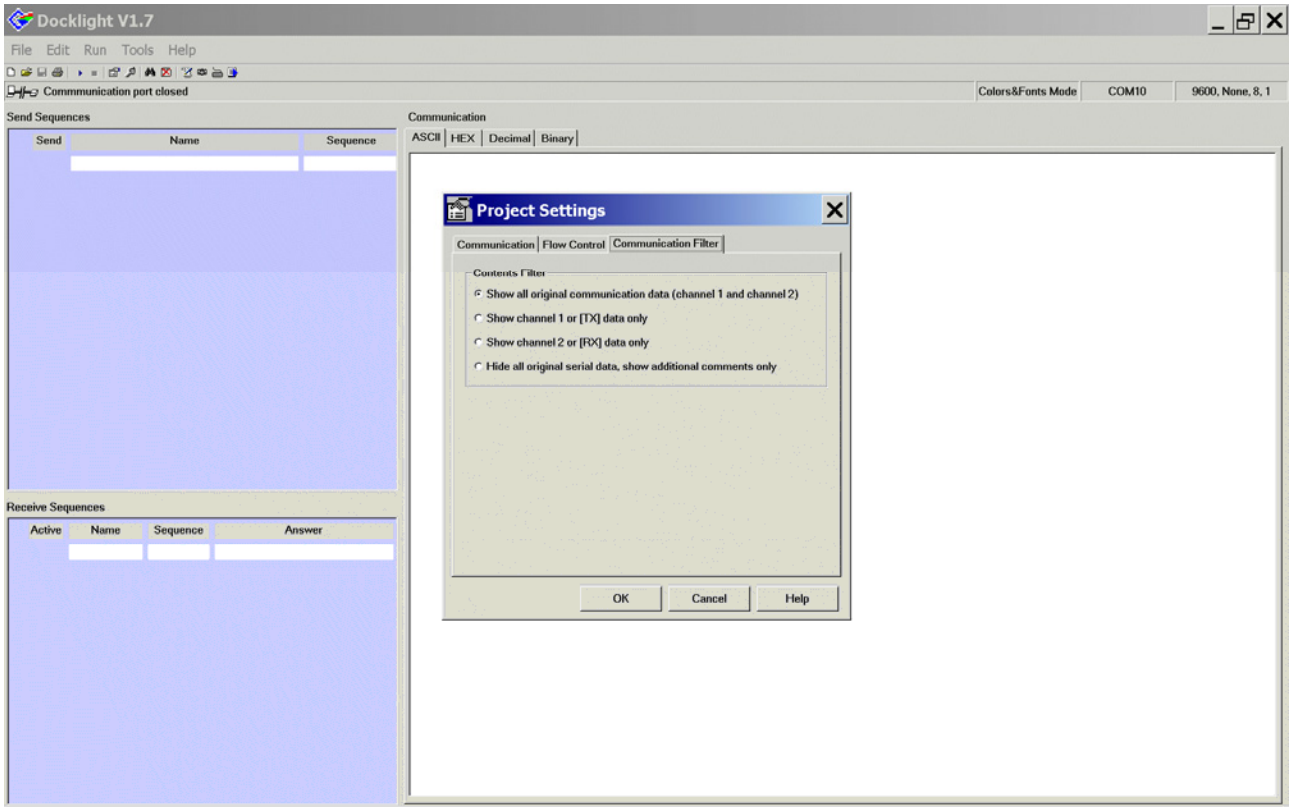
Project Settings: Communication: COM Port Settings: Baud Rate: select 9600
Project Settings: Communication: COM Port Settings: Parity: select None
Project Settings: Communication: COM Port Settings: Parity Error Char.: select (ignore)
Project Settings: Communication: COM Port Settings: Data Bits: select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



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

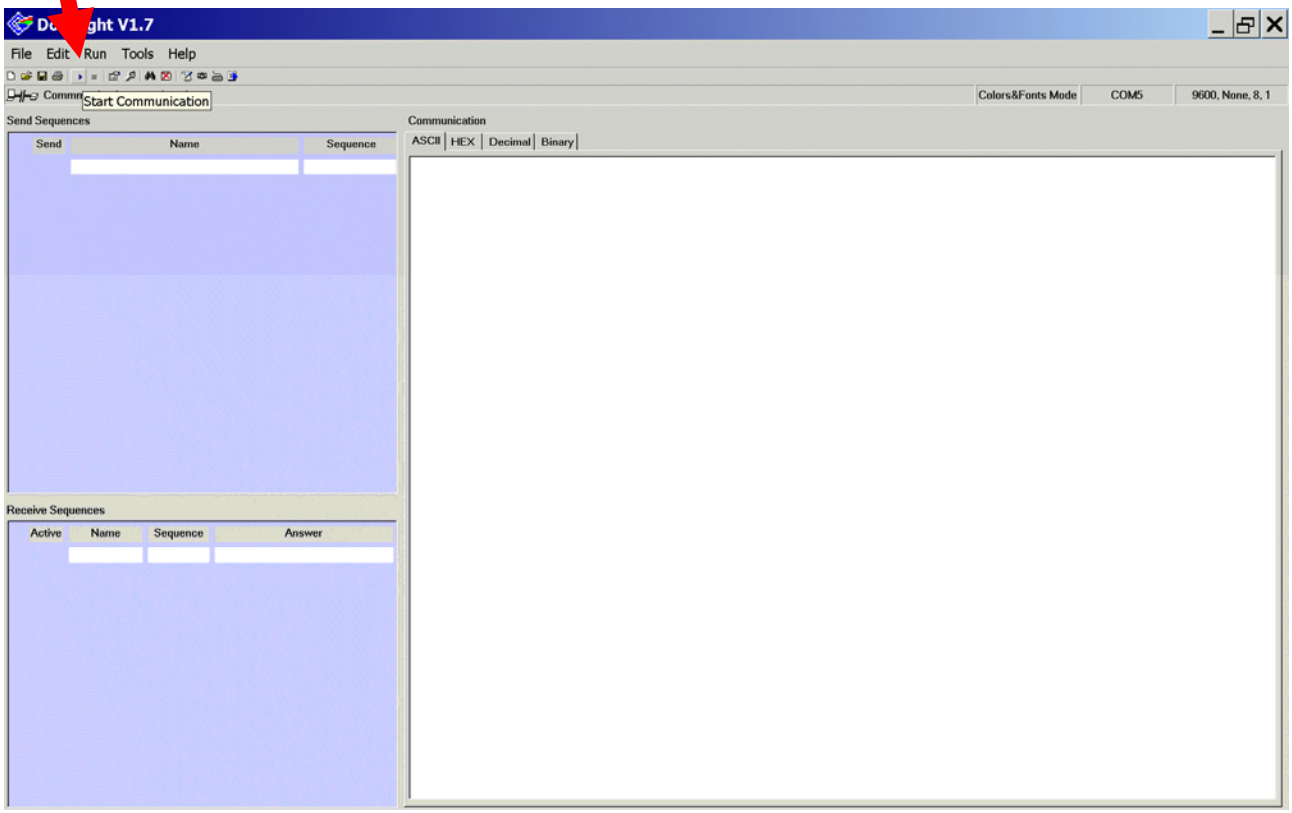


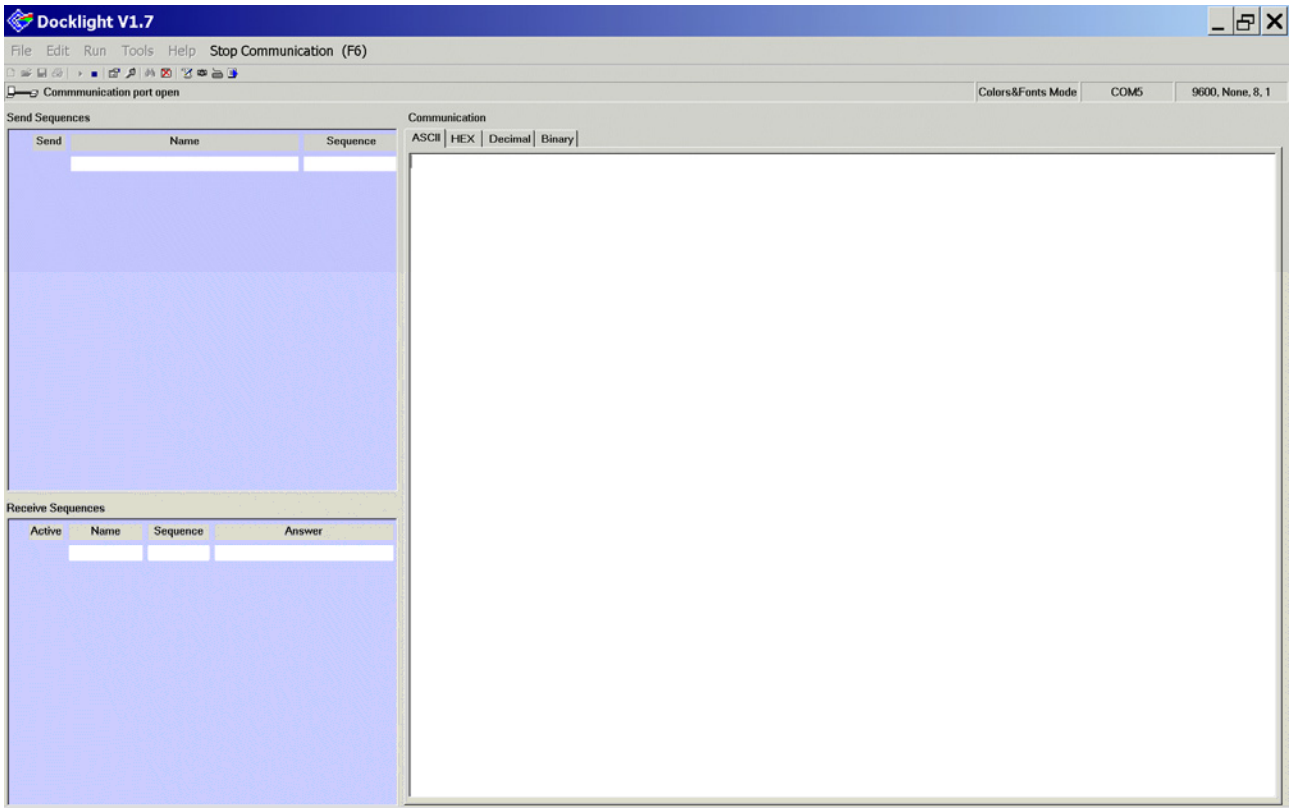
Project Settings:
Communication Filter: Contents Filter: click ⊙ Show all original communication data



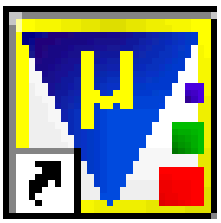
OK

Click: 


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

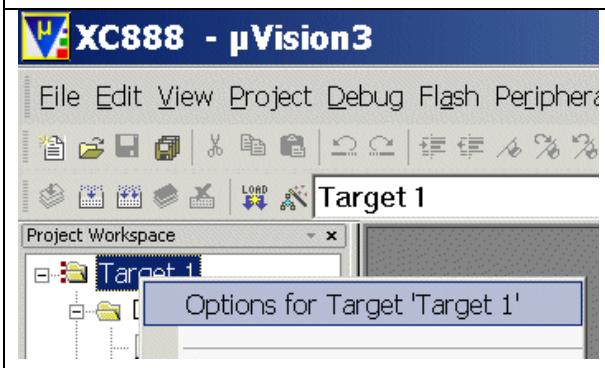
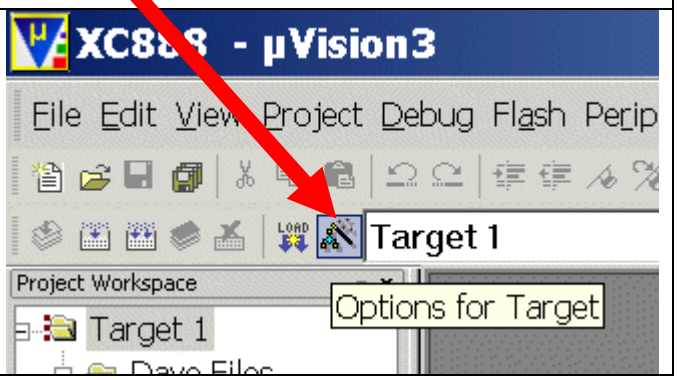


Note:
Docklight is now ready for serial communication!

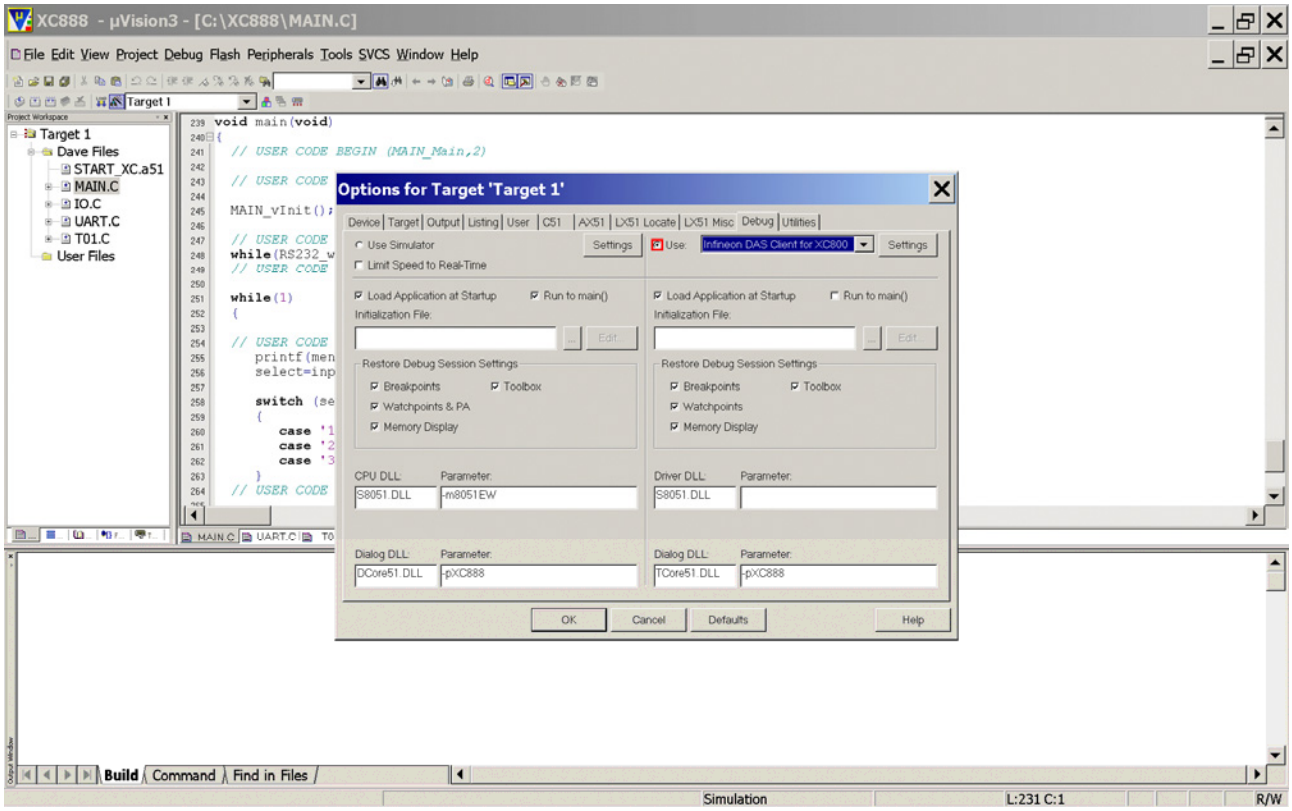


Go back to µVision:

Configure/select the debugger:

mouse position: (Project Workspace, Files): Target1 click right mouse button Options for Target ´Target1‘	or	click
		

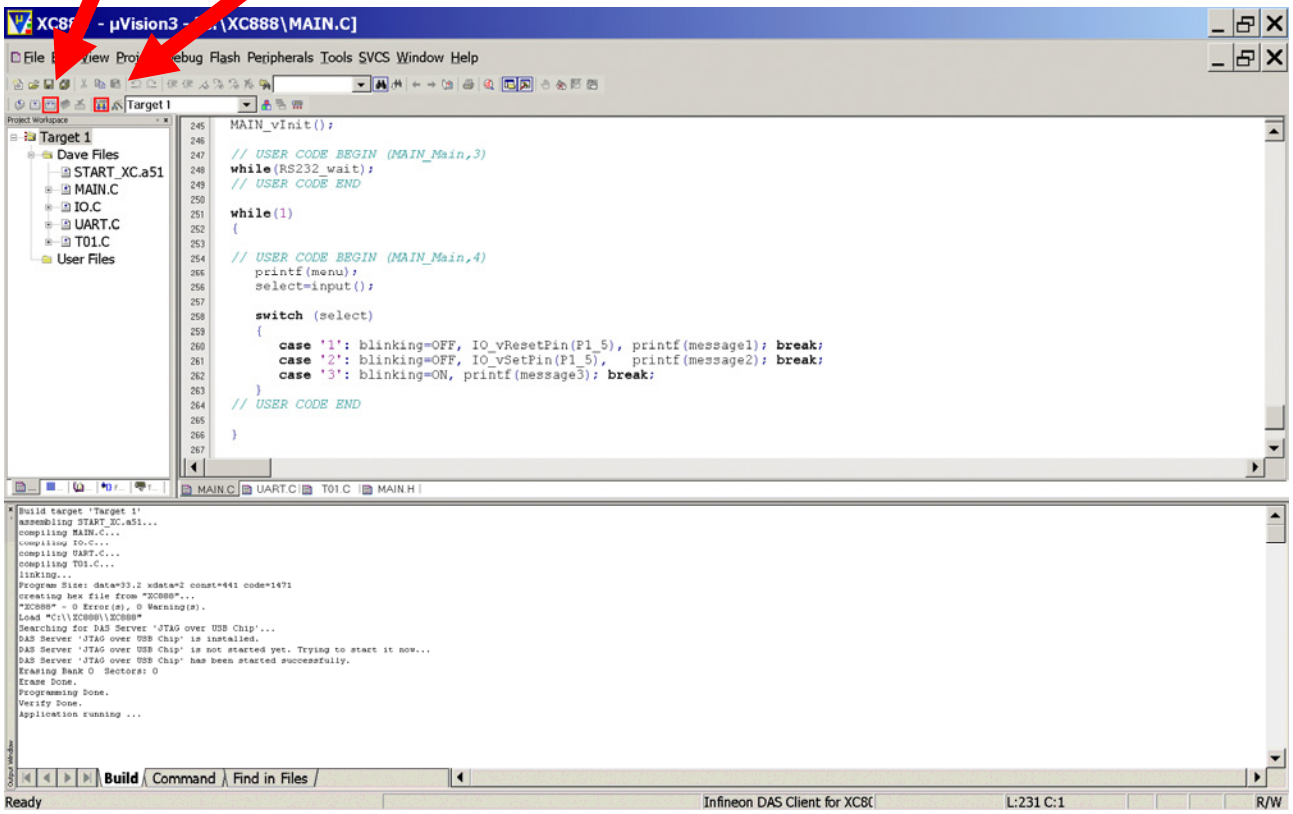
Options for Target ´Target1‘:Debug:
click ⊙ Use: select Infineon DAS Client for XC800

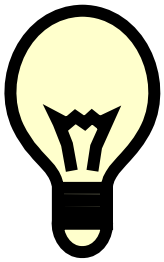


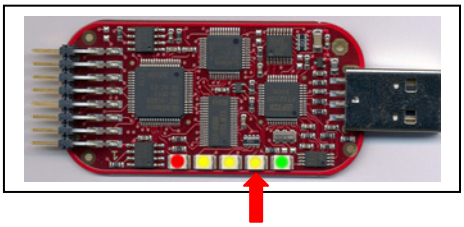
Click OK

1.) click: 

2.) click: 

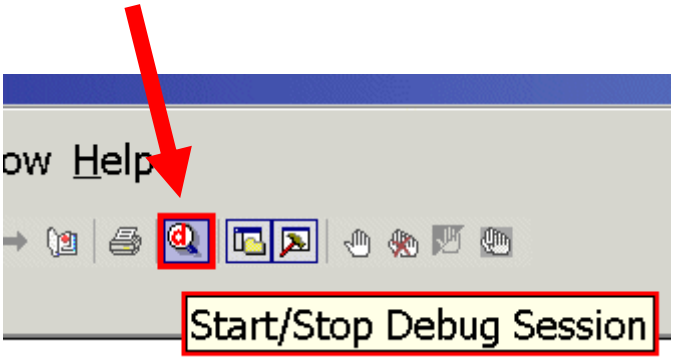


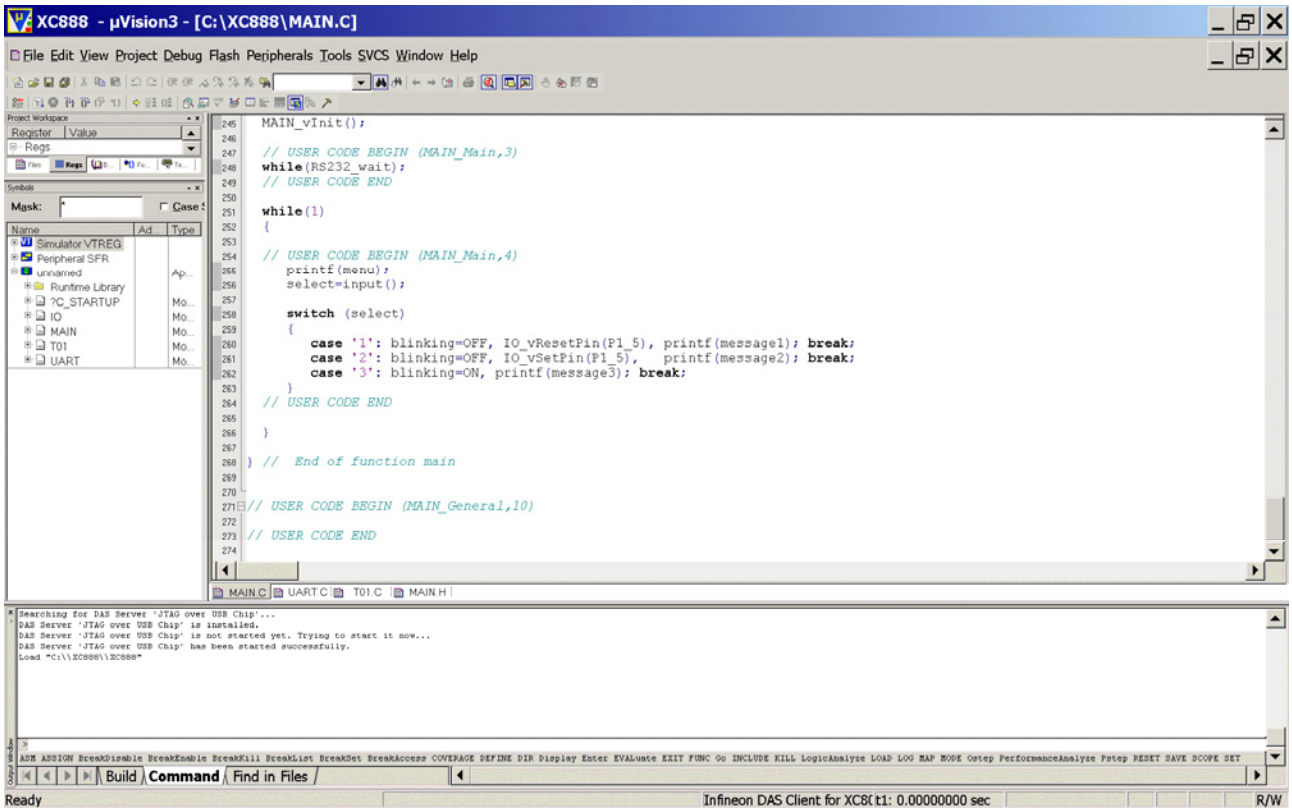


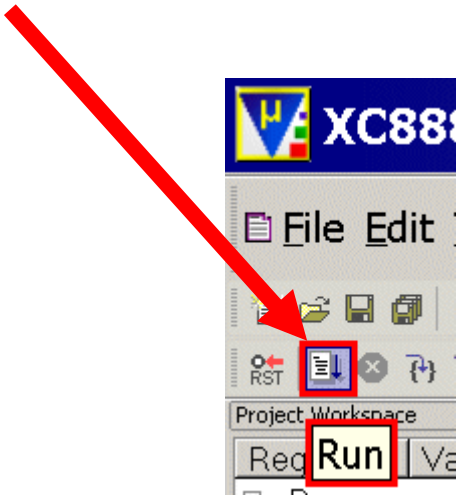


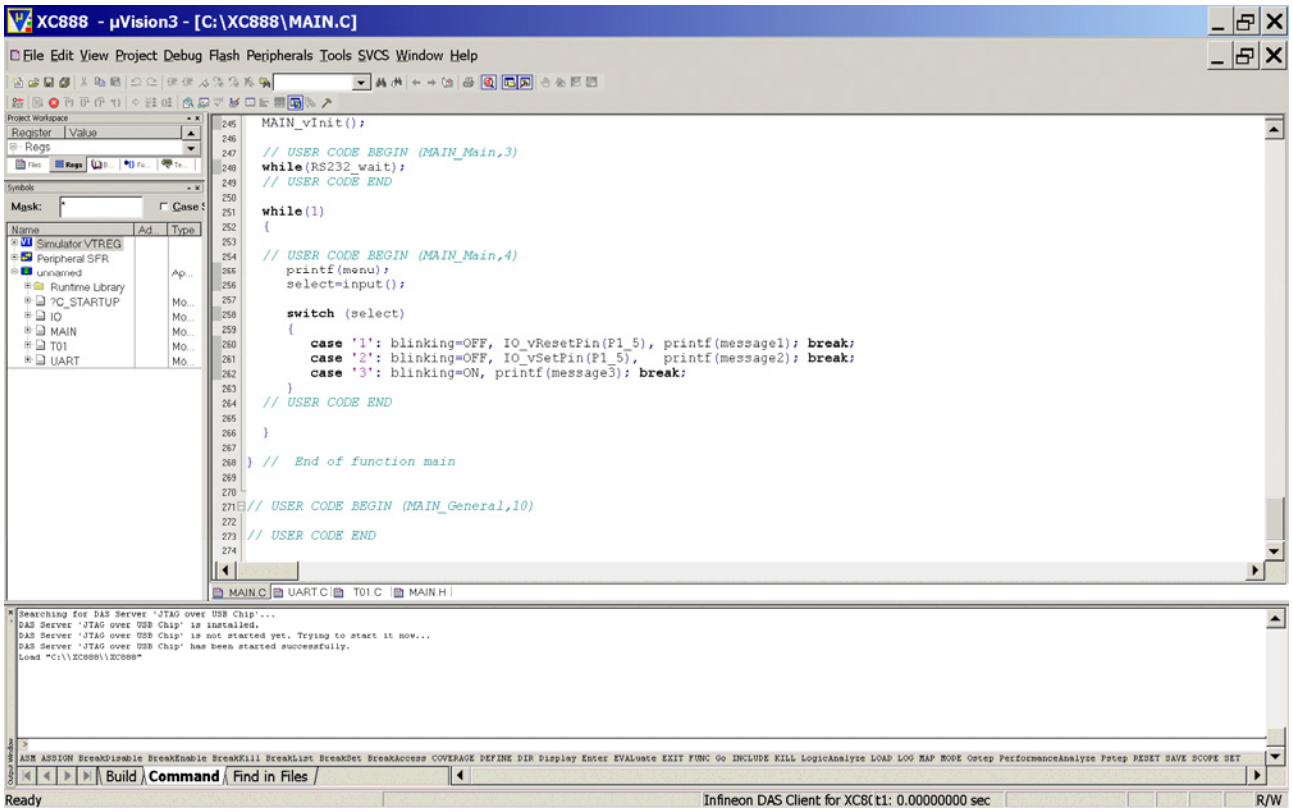
XC888

Note:
The debugger selects the XC888 microcontroller in the XC800 USCALE start kit!!!
Therefore we are going to run our program with the debugger!!!

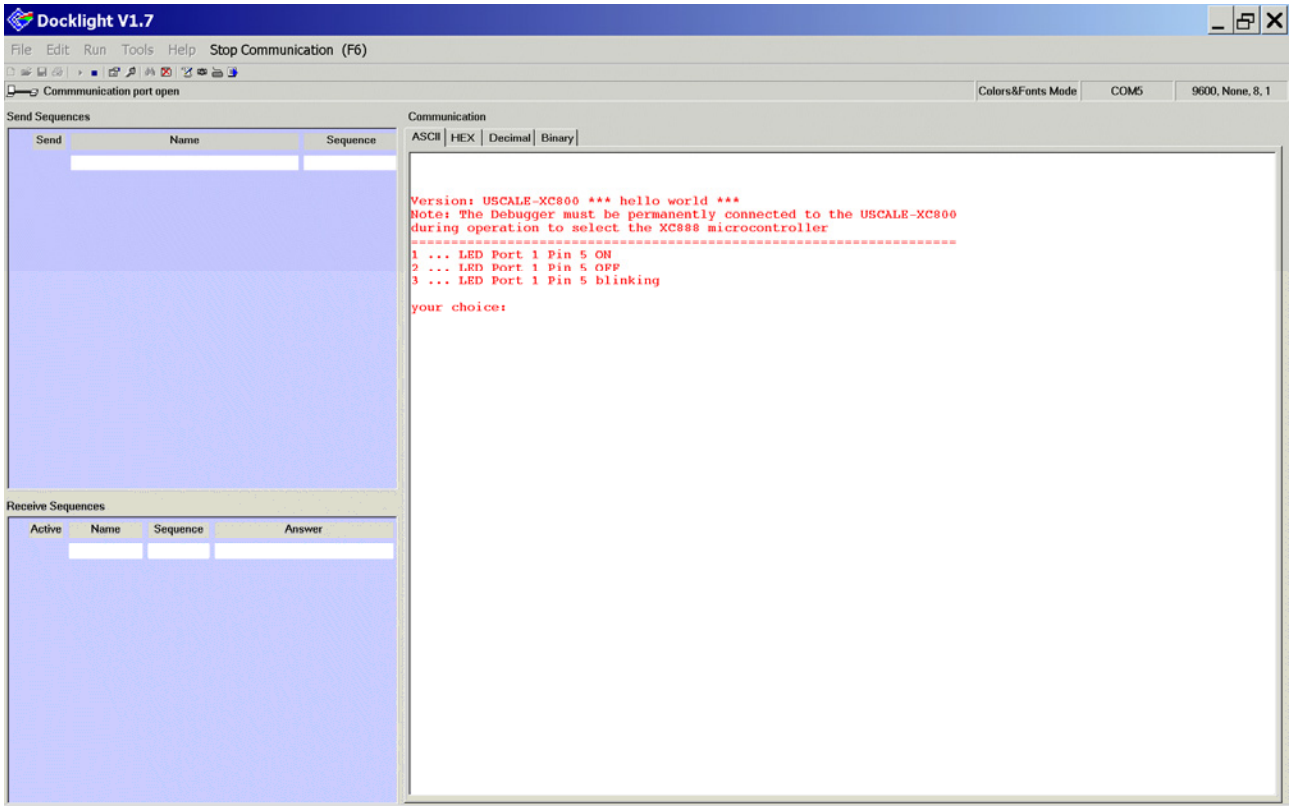
Debug – Start/Stop Debug Session	or click 



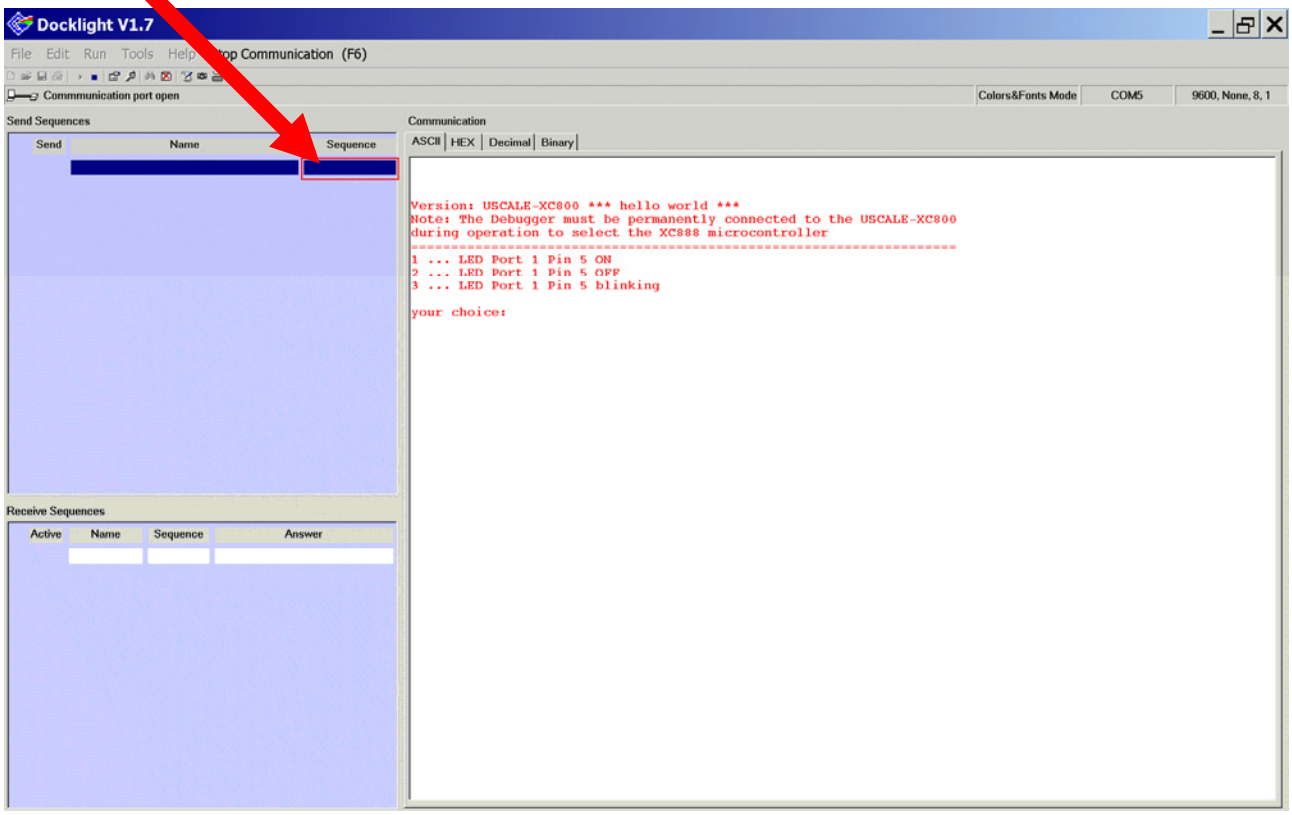
Debug – Run	or click 


XC888 - µVision3 - [C:\XC888\MAIN.C]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
MAIN_vInit();
// USER CODE BEGIN (MAIN_Main,3)
while(RS232_wait);
// USER CODE END
while(1)
{
// USER CODE BEGIN (MAIN_Main,4)
printf(menu);
select=input();
switch (select)
{
case '1': blinking=OFF, IO_vResetPin(P1_5), printf(message1); break;
case '2': blinking=OFF, IO_vSetPin(P1_5), printf(message2); break;
case '3': blinking=ON, printf(message3); break;
}
// USER CODE END
}
} // End of function main
// USER CODE BEGIN (MAIN_General,10)
// USER CODE END
Searching for DAS Server 'JTAG over USB Chip'...
DAS Server 'JTAG over USB Chip' is installed.
DAS Server 'JTAG over USB Chip' is not started yet. Trying to start it now...
DAS Server 'JTAG over USB Chip' has been started successfully.
Load "C:\\XC888\\XC888"
Build
Command
Find in Files
Ready
R/W

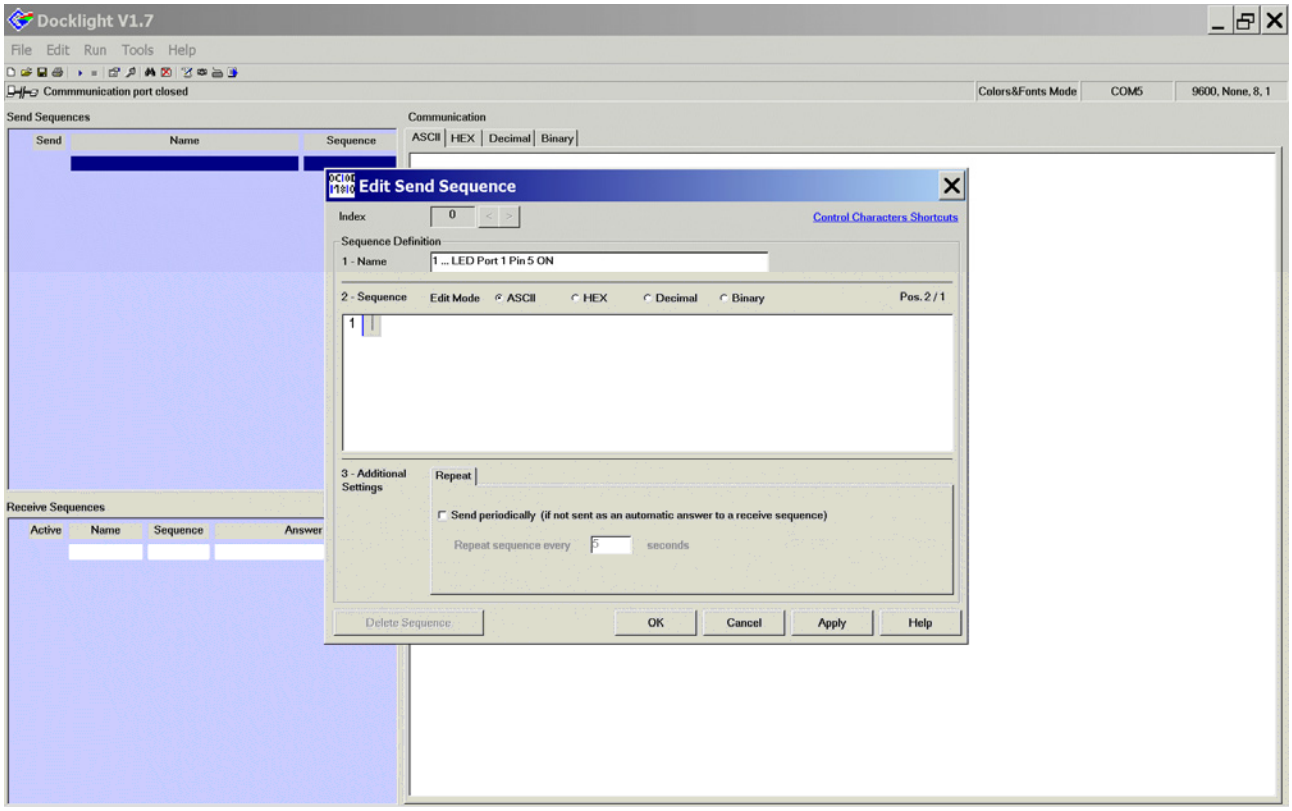
Go back to Docklight and see the result:



Double click inside the red box:

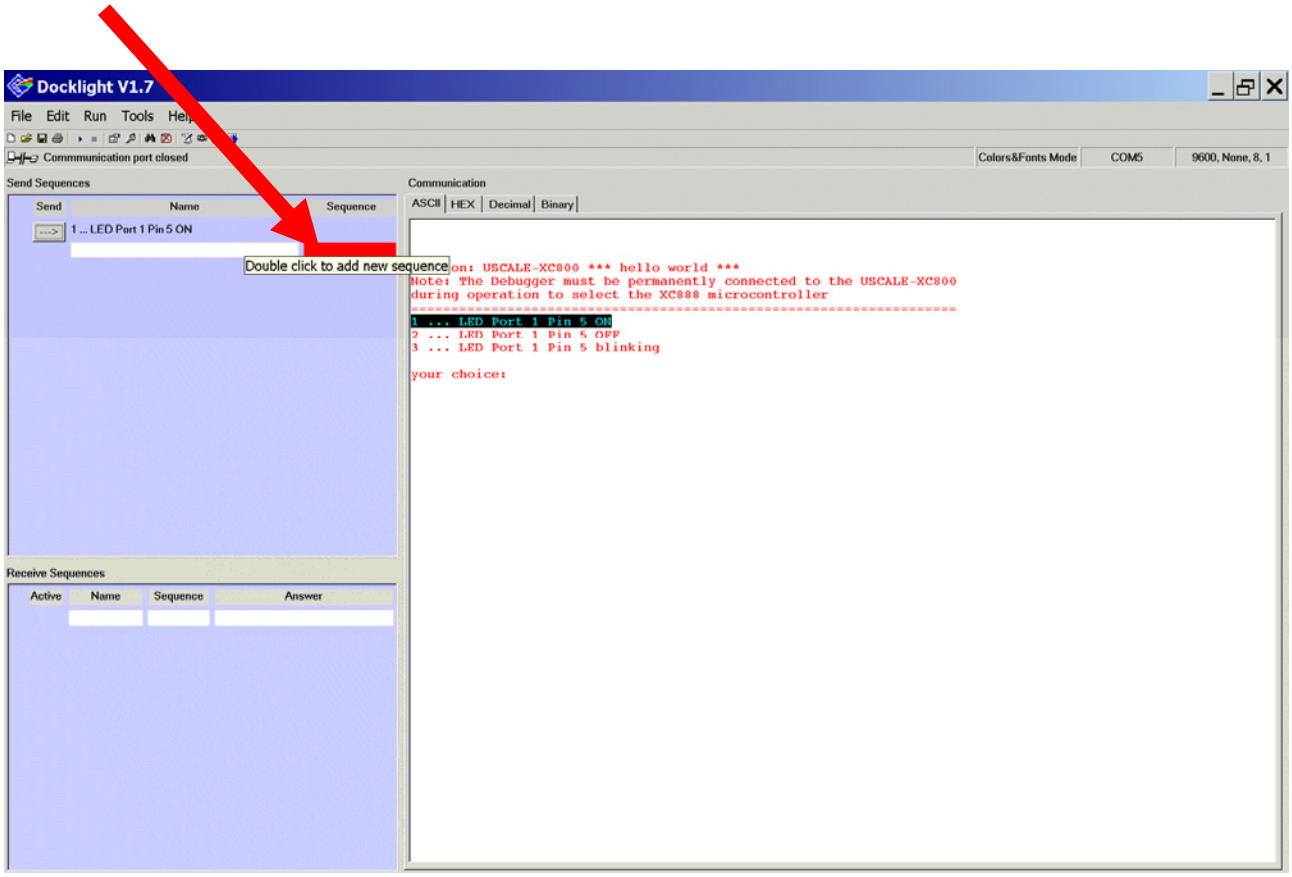


Edit Send Sequence: Sequence Definition: 1- Name: insert: 1 ... LED Port 1 Pin 5 ON
Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 1

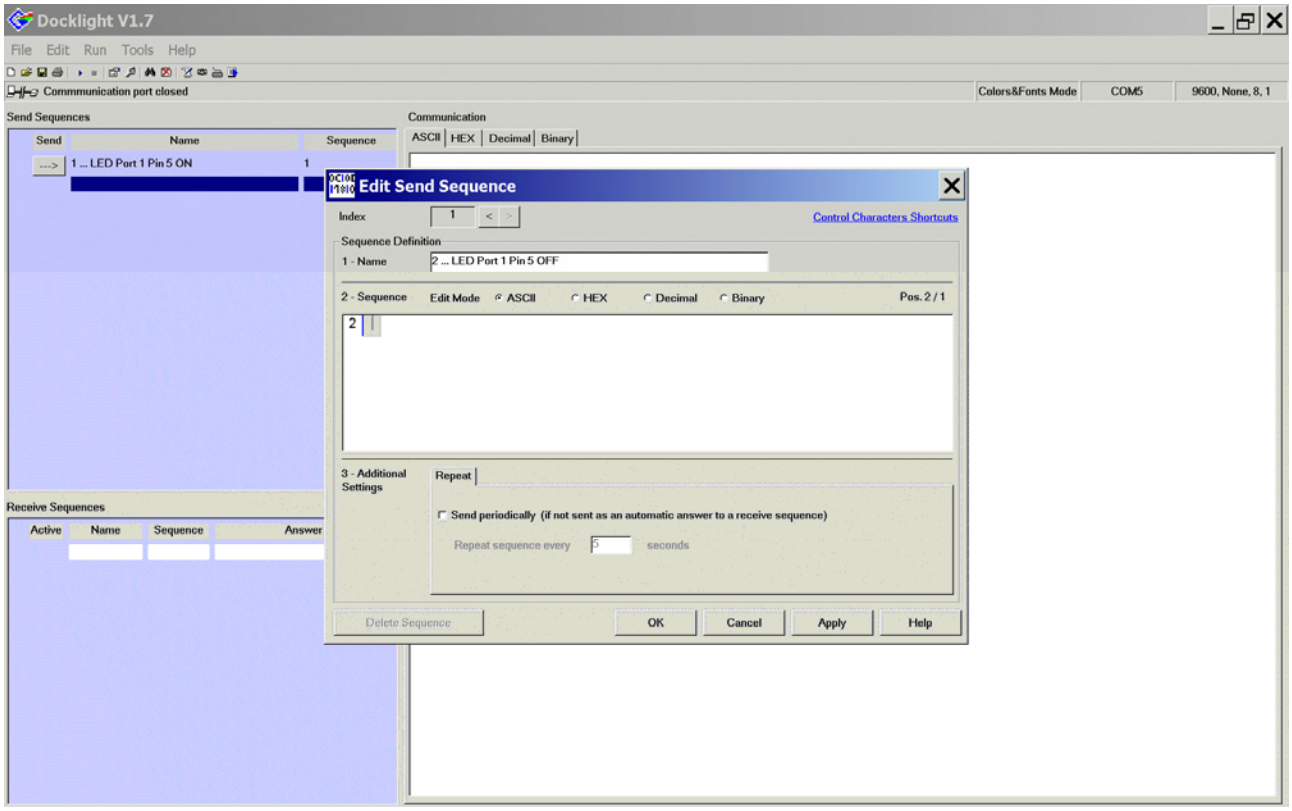


OK

Double click inside the red box:

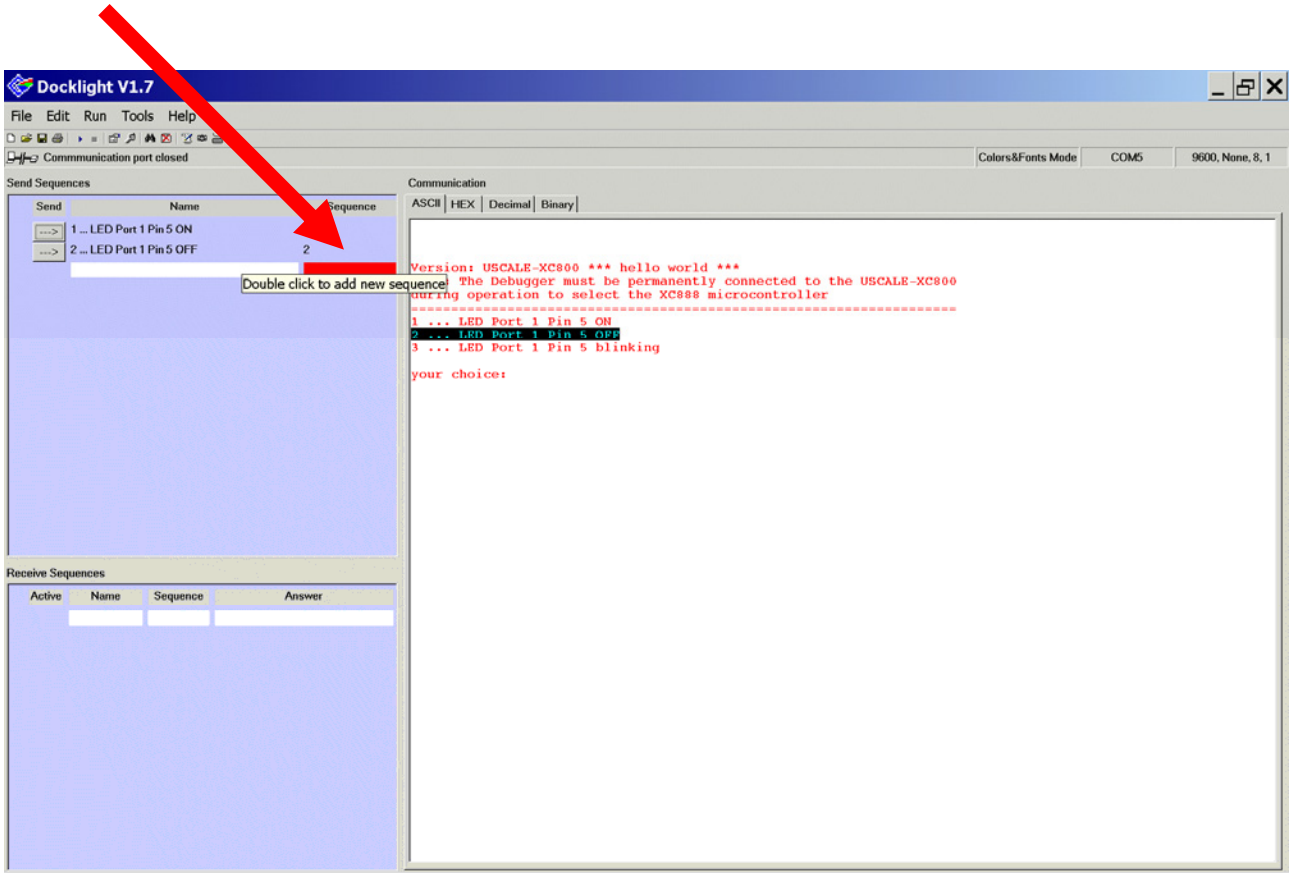


Edit Send Sequence: Sequence Definition: 1- Name: insert: 2 ... LED Port 1 Pin 5 OFF
Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 2

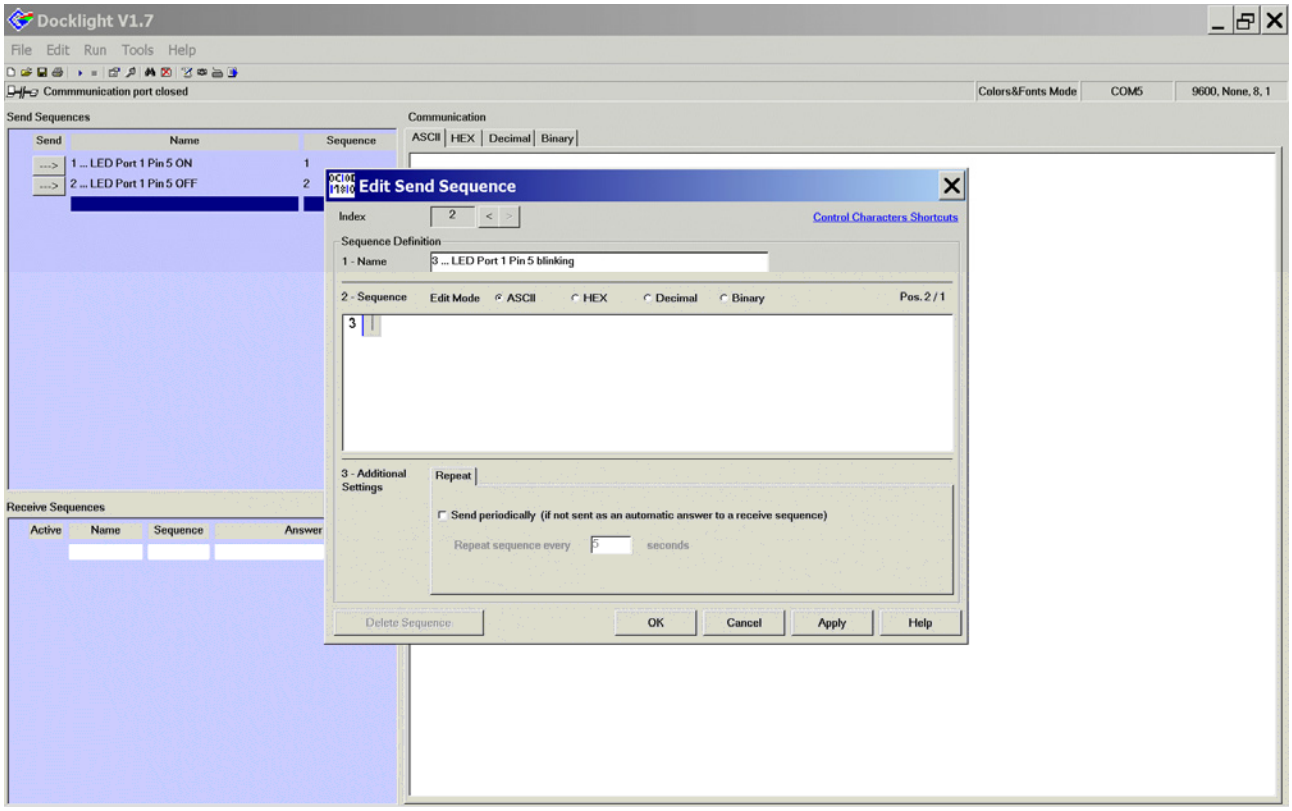


OK

Double click inside the red box:



Edit Send Sequence: Sequence Definition: 1- Name: insert: 3 ... LED Port 1 Pin 5 blinking
Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 3

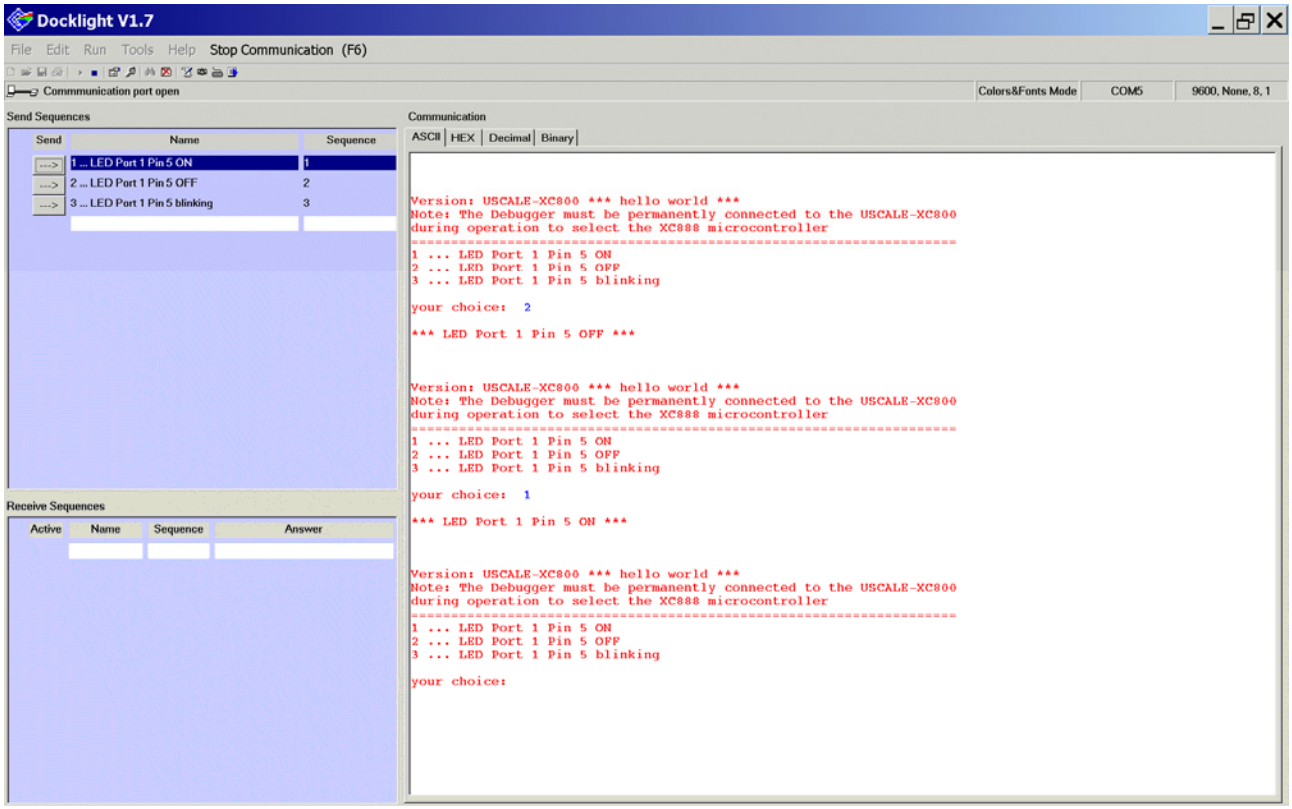


OK

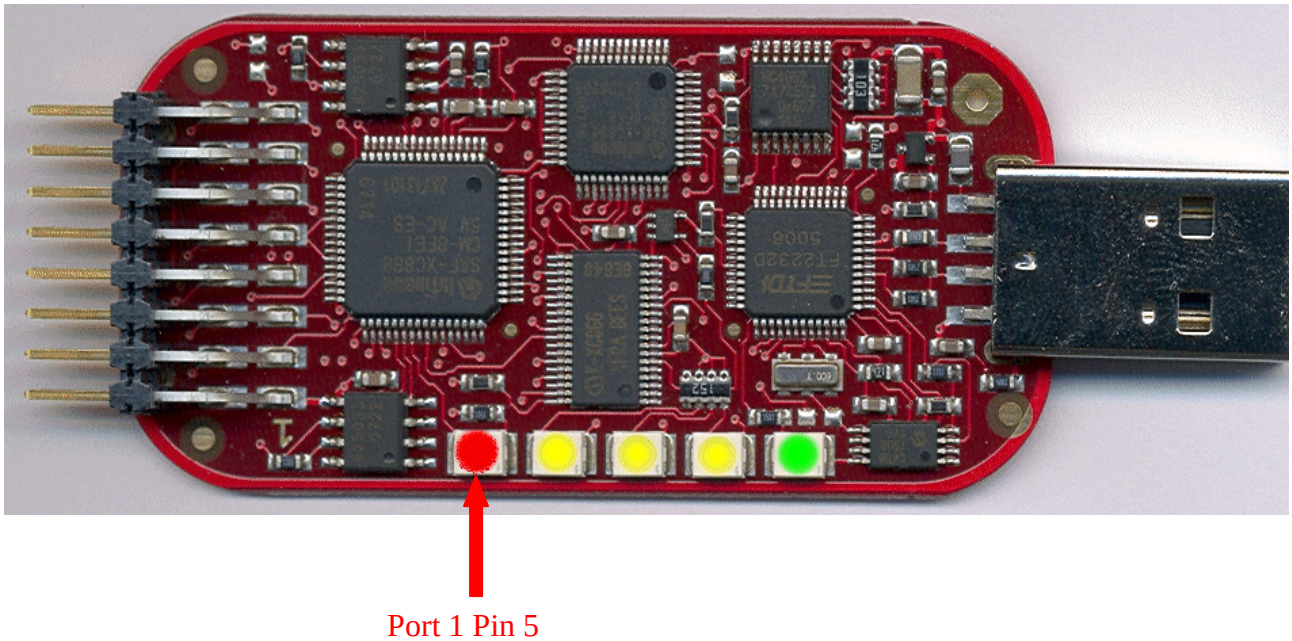
Click  1 ... LED Port 1 Pin 5 ON

or click  2 ... LED Port 1 Pin 5 OFF

or click  3 ... LED Port 1 Pin 5 blinking



and check the result on your XC800 USCALE start kit:



Conclusion:

In this step-by-step book you have learned how to use the XC800 USCALE start kit 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 XC800 USCALE start kit.

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 the XC800 USCALE start kit!

Note:

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

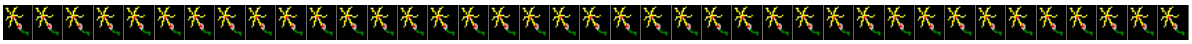
All these step-by-step books use the same microcontroller resources and the same example code.

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.

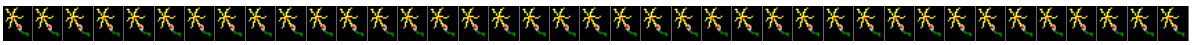
7.) Feedback (XC800 USCALE start kit, Keil tools): Your opinion, suggestions and/or criticisms



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