

AP08079

XC878CM-16FF

XC878 Easy Kit: "Cookery Book" for a hello world application using the KEIL tool chain

Microcontrollers



Never stop thinking

Edition 2008-10-13

**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: 2008-09 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



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

Introduction:

This “Appnote” is a Hands On Training / Cookery Book / step-by-step book.
It will help inexperienced users to get the XC878 Easy Kit up and running.

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

The purpose of this document is to gain know-how of the microcontroller and the tool-chain.
Additionally, the "hello world example" can easily be expanded to suit your needs.
You can connect either a part of - or your entire application to the XC878 Easy 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:

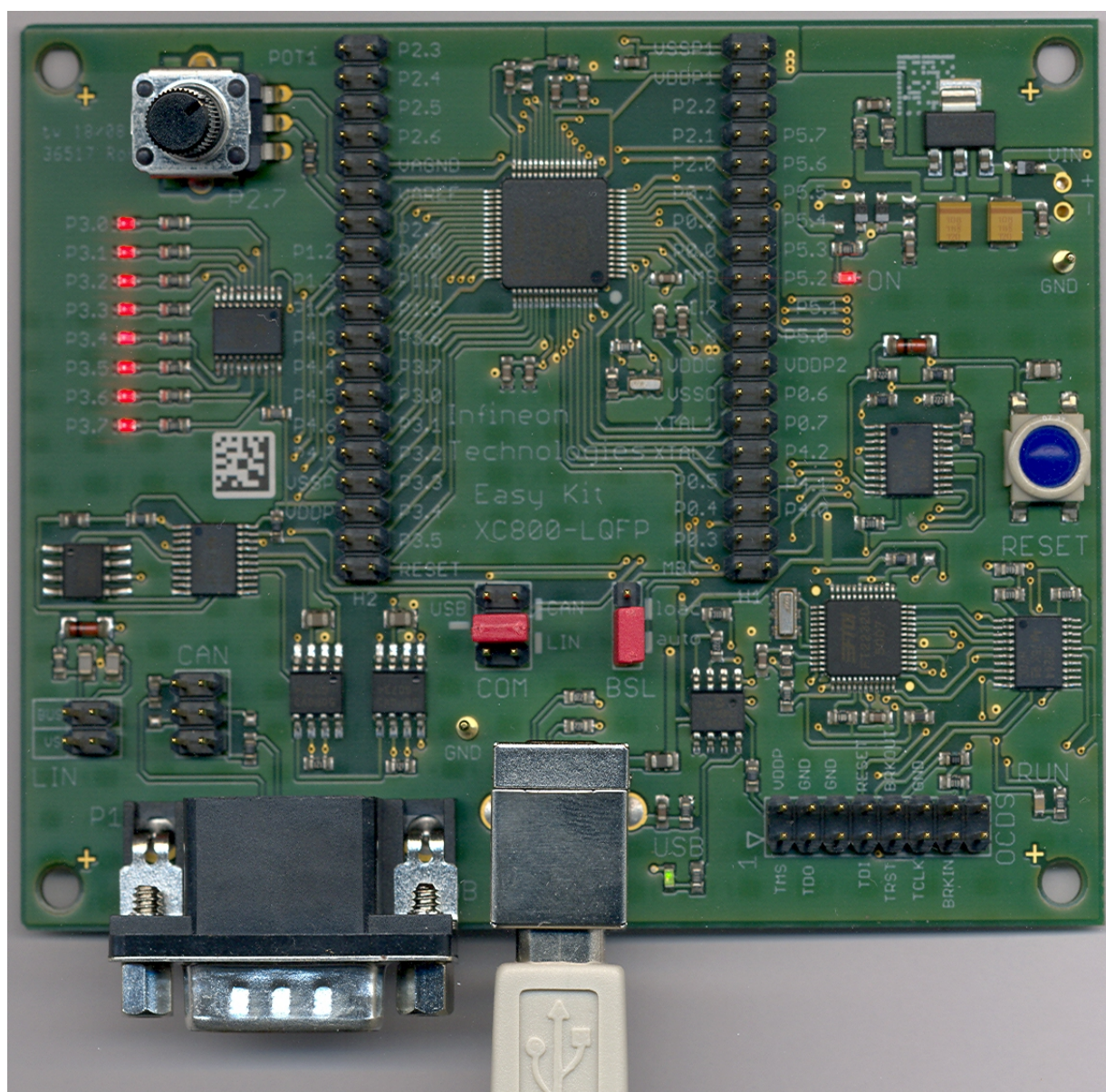
The style used in this document focuses on working through this material as fast and easily as possible. That means there are full screenshots instead of dialog-window-screenshots; extensive use of colours and page breaks; and listed source-code is not formatted to ease copy & paste.

Have fun and enjoy the XC878 Easy Kit!



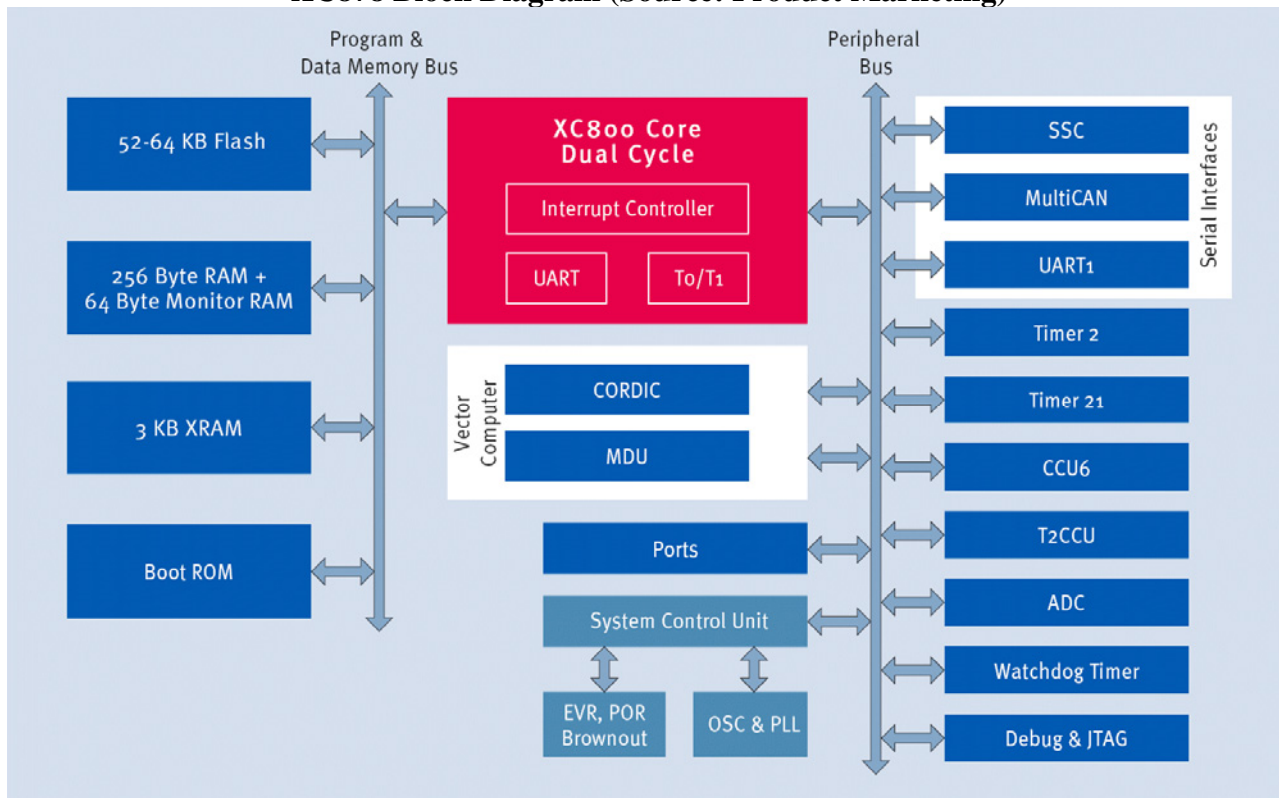
Programming Example

XC878 Easy Kit

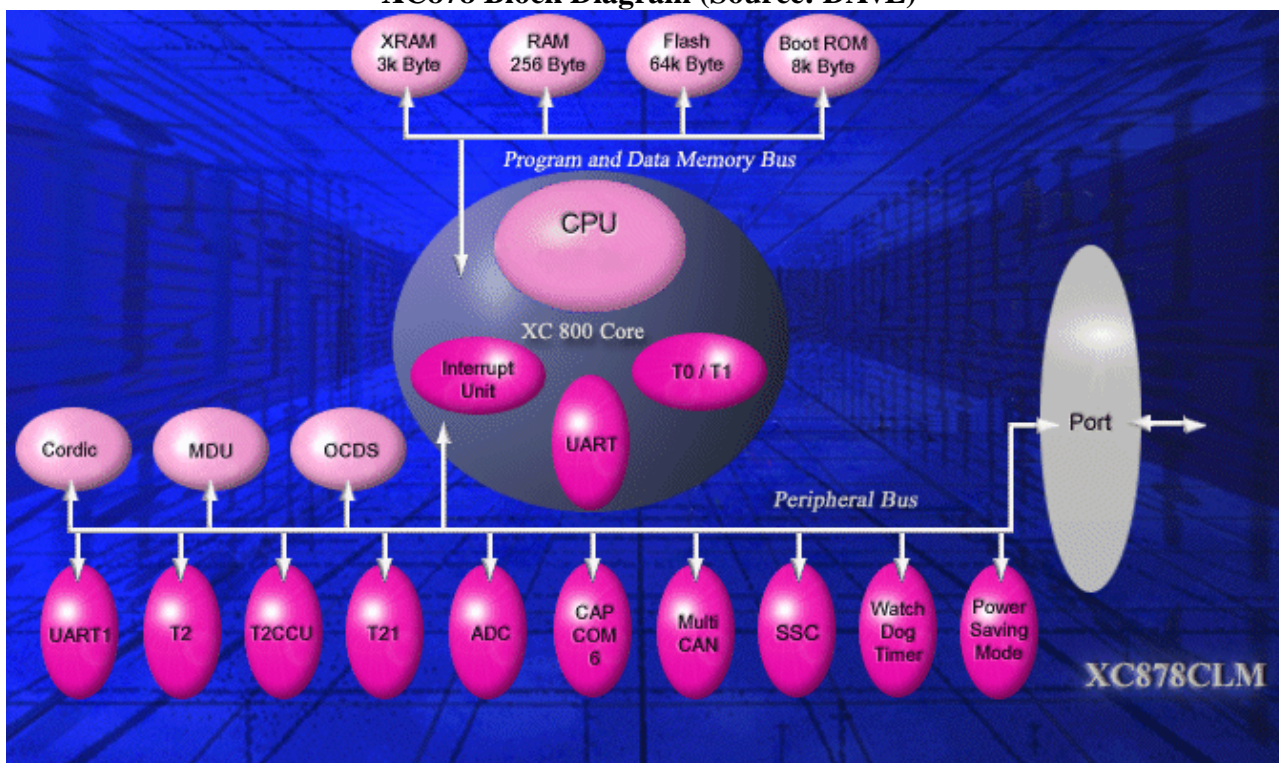


Used/selected microcontroller:

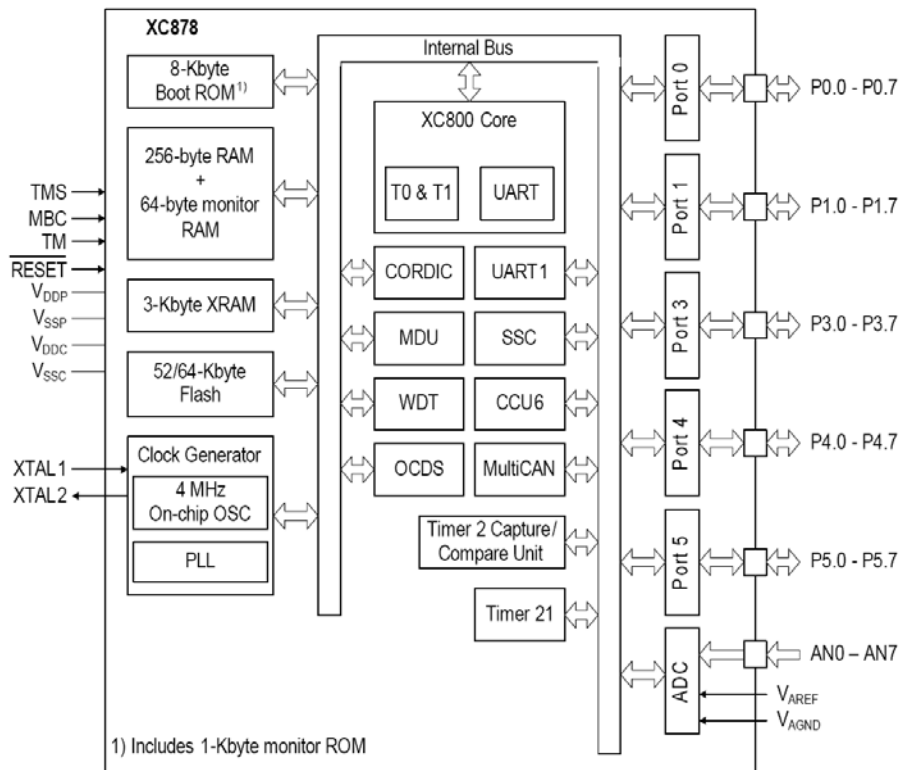
XC878 Block Diagram (Source: Product Marketing)



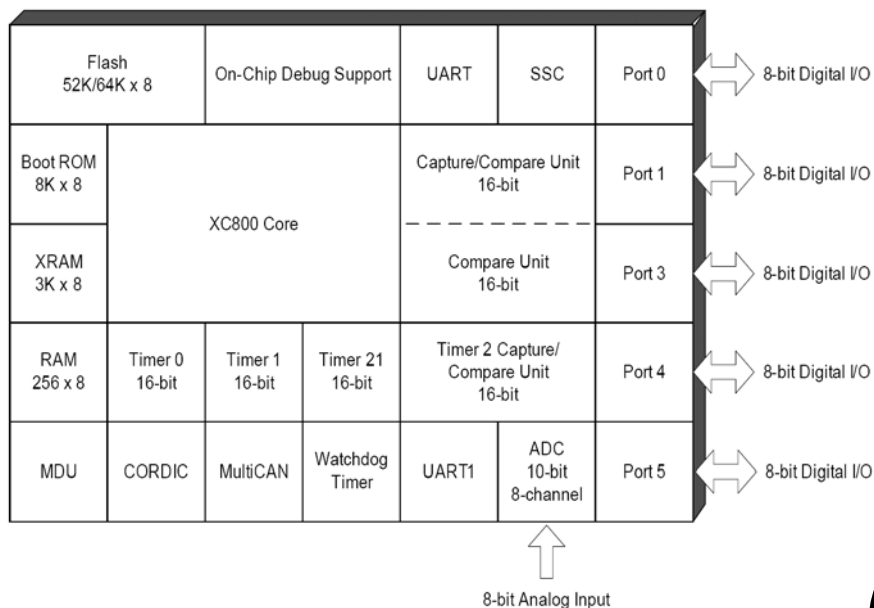
XC878 Block Diagram (Source: DAVe)



XC878 Block Diagram (Source: User's Manual)



XC878 functional units (Source: User's Manual)



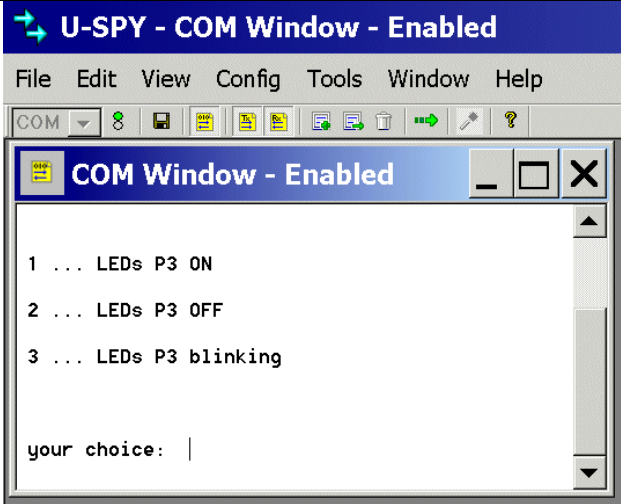
Note:

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“

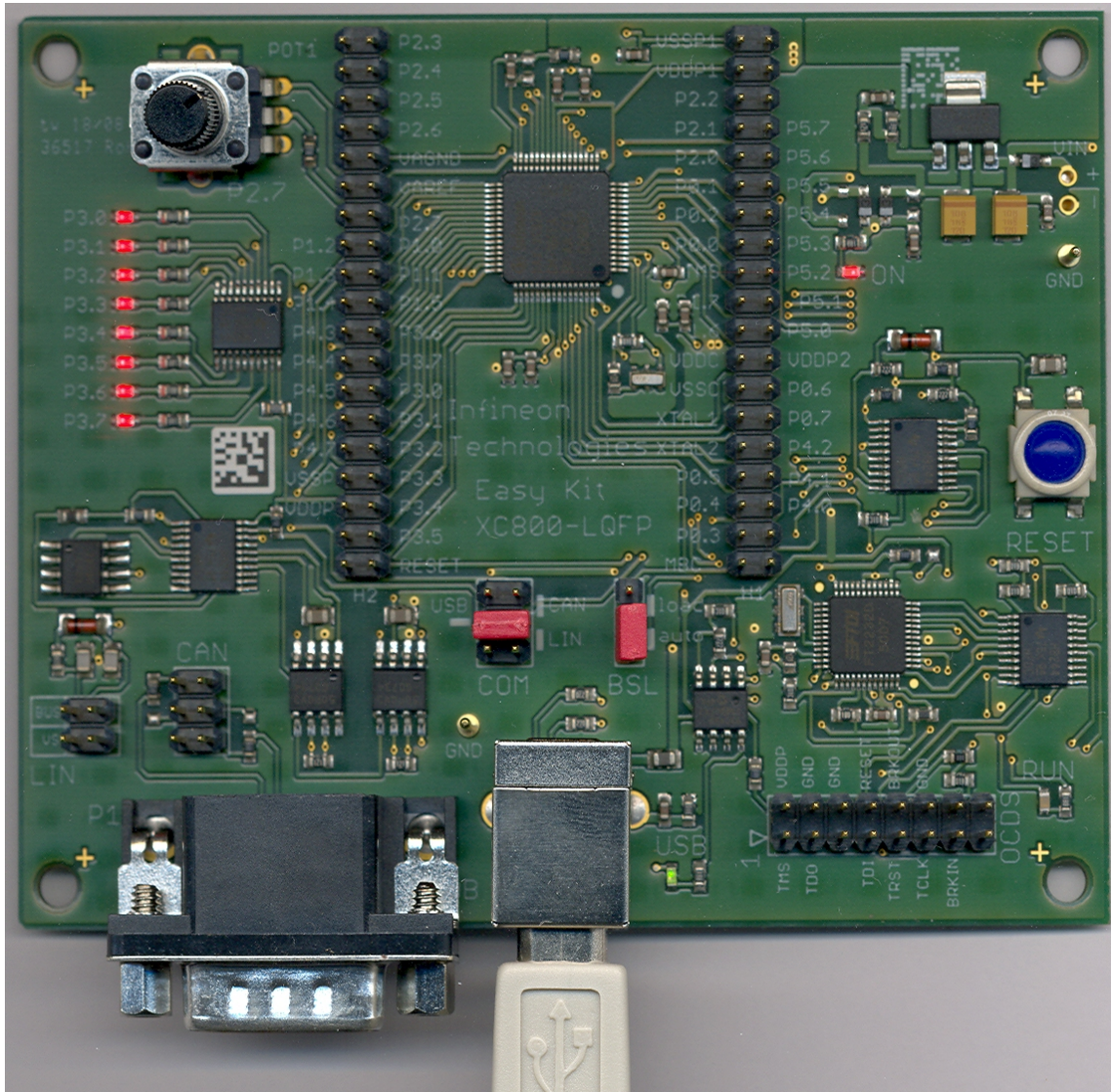
For your first programming example for the XC878 Easy Kit:

Your program:	
Chapter/ Step	*** Recipes ***
1.)	DAS Installation + Connecting the XC878 Easy Kit
2.)	DAvE (program generator) DAvE Installation (mothersystem) + DAvE Update Installation (XC878.DIP) for XC878
3.)	Using DAvE Microcontroller initialization for your programming example
4.)	Using the KEIL Development Tools (C-Compiler) Programming of your application (XC878) with the KEIL tool chain (µVision3) V8.16a
5.)	Using the simulator
6.)	Using real hardware (+ OnChipFlash-Programming)

Thanks To & Feedback


7.)	Thanks To
8.)	Feedback

1.) DAS Installation + Connecting the XC878 Easy Kit:



Screenshot of the XC878 Easy Kit homepage:

<http://www.infineon.com/cms/en/product/channel.html?channel=db3a304319c6f18c0119ebe345f15325>



Never stop thinking

[Home](#) | [Sitemap](#) | [Select Language](#) | [Login](#)

[About Infineon >>](#)

Get Product information

Select a Category

Microcontrollers

Search Part Number
Go
Search Website
Go

Home > Microcontrollers > Development Tools, Software and Training > XC800 Development Tools and Software > Starter Kits, Evaluation Boards and Application Kits > Easy Kit XC878

Print Page
Send Page

Easy Kit XC878

MCU Derivatives: SAX-XC878CM-16FFA
CPU Clock: 24 MHz

On-Chip Memory:

- 3 kByte RAM,
- 52/64 kByte Flash (incl. up to 4kByte data flash)

Interfaces:

- USB Connector for power supply, UART communication, and flash downloading,
- LIN via Header,
- CAN0/1 via Header and via 9 Pin (male) D-Sub,
- JTAG via Header or via USB with built in mini wiggler,

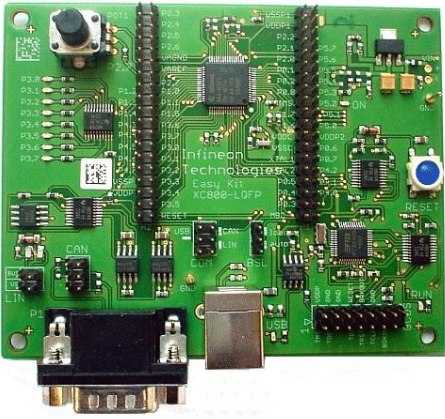
Includings:

- USB Cable, CD, Evaluation Board,
- Technical Documentation e.g. user manuals (CD),
- Free unlimited source code debugger (CD),
- Evaluation Versions of development Tools: e.g. Compiler, Debugger, DAVE (CD)
- Examples with Tutorial Notes.

Order Nr.: KIT_XC878_EK_V1

Price: 99,- EUR

How to order?
To order your kit, please click [here](#).



Documents

Contact us

Document Types

Downloads

Title	Date	Version	Size
XC878 Easy Kit CD - Version 1.0 for download (XC878_easykit_CD_V1.0.zip)	05 Jun 2008	V1.0	185.7 MB

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

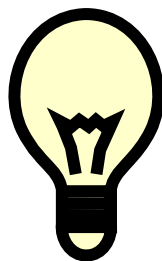
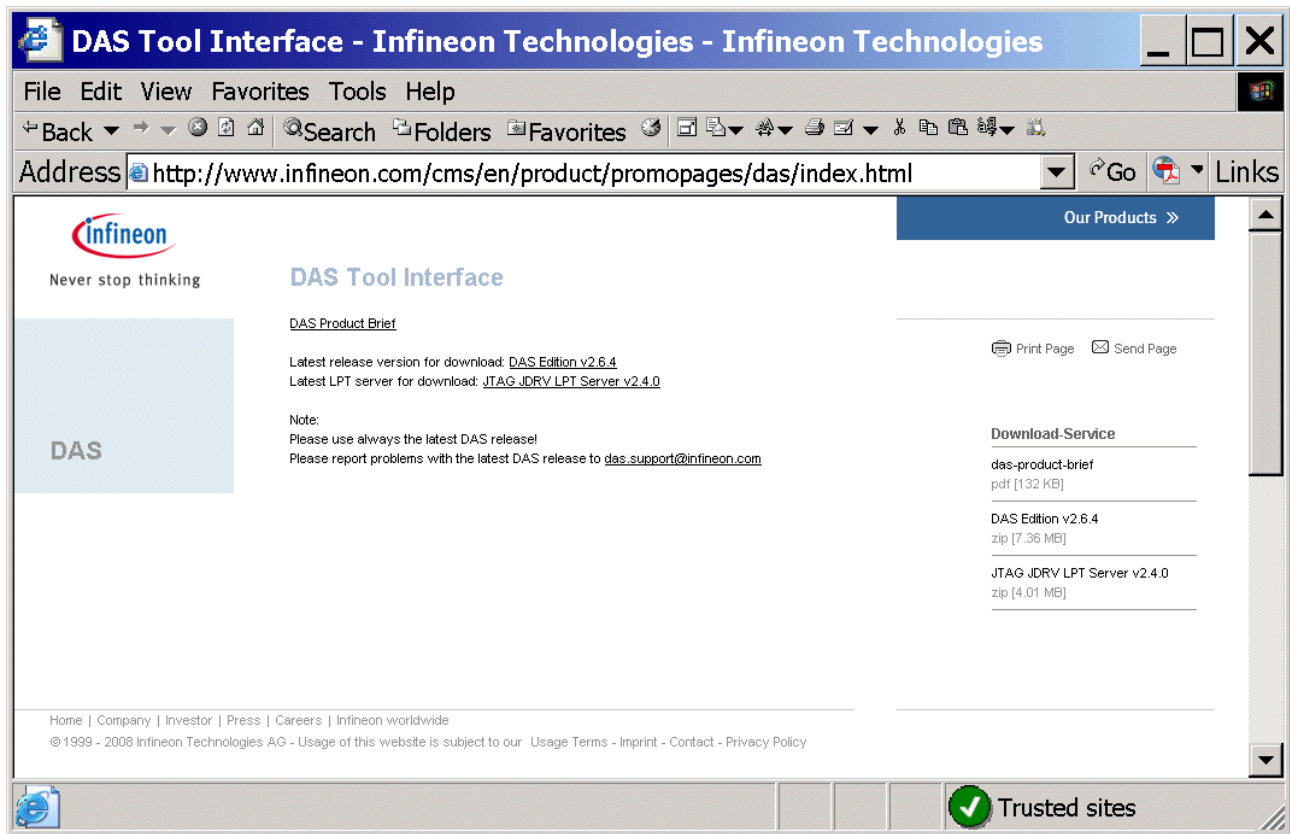


Note:

For further information, please refer to the [XC878 Easy Kit Board Manual V1.0, April 2008](#).

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

Go to www.infineon.com/DAS:



Note:

The DAS Server must be installed on your host computer!

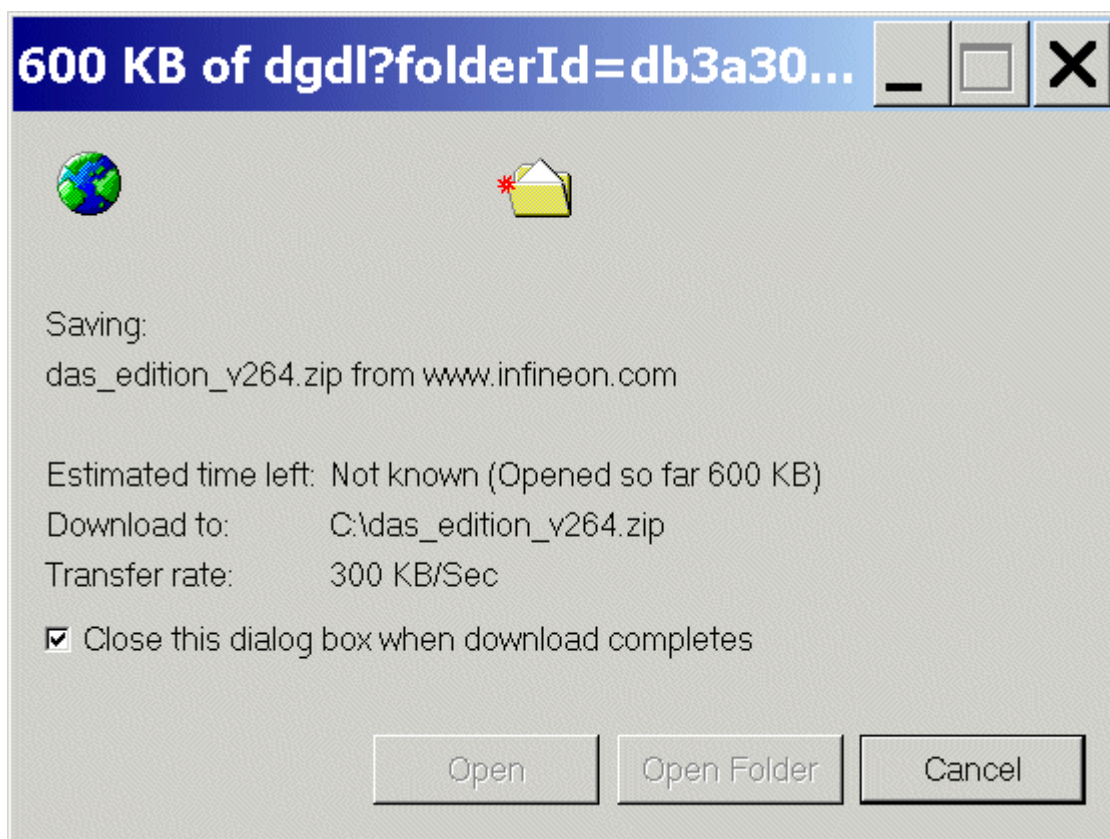
The goal of the DAS software is to provide one single interface for all types of tools.

The USB Device driver communicates with the XC878 Easy Kit when connected to the host computer.

The USB Device driver for the XC878 Easy Kit USB interface is included in the DAS software.

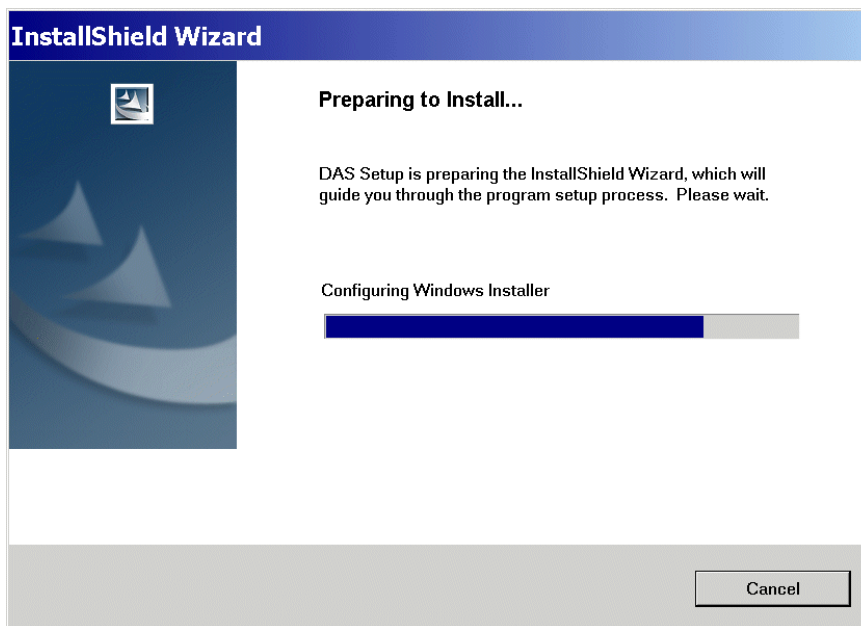
A virtual COM port driver is also included.

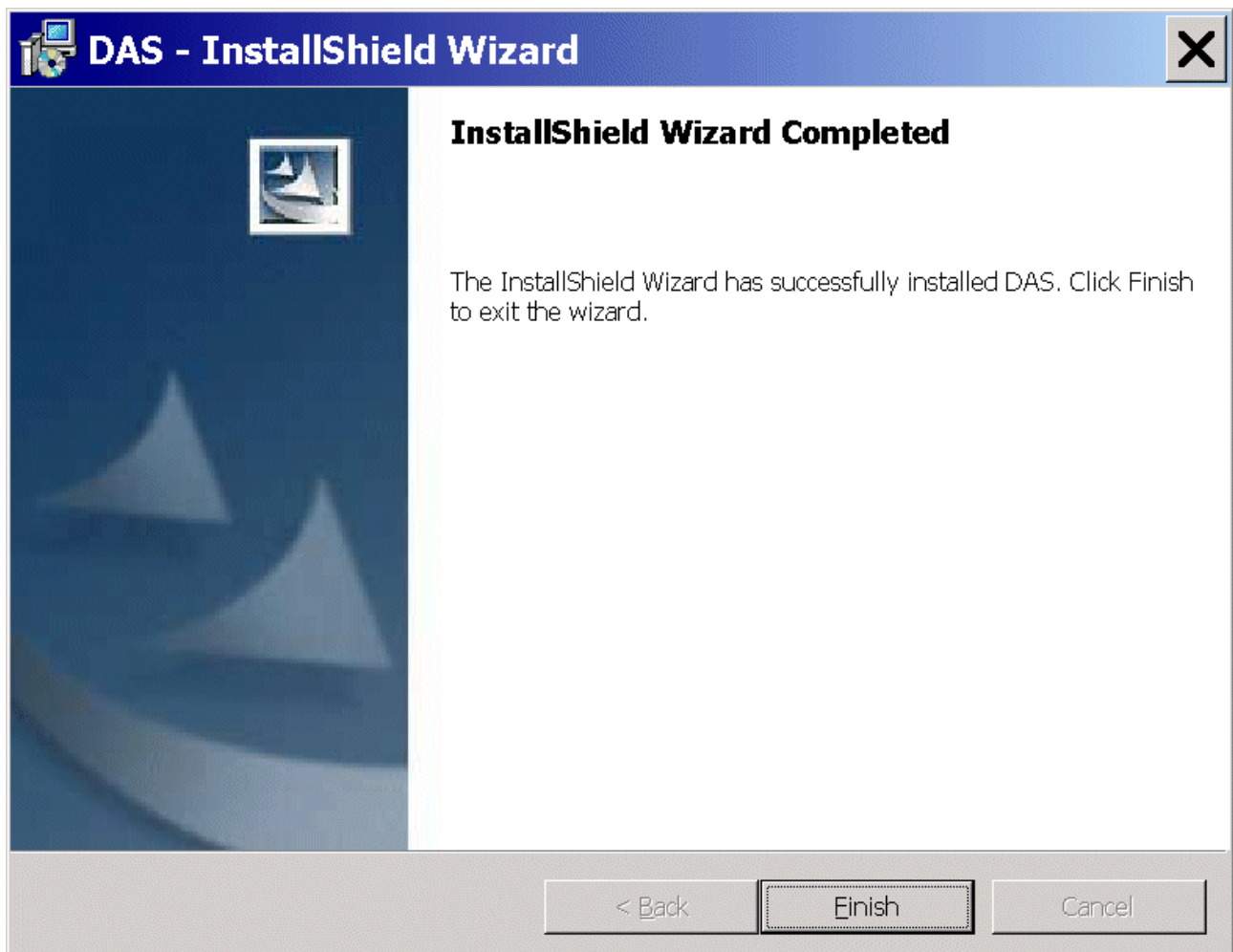
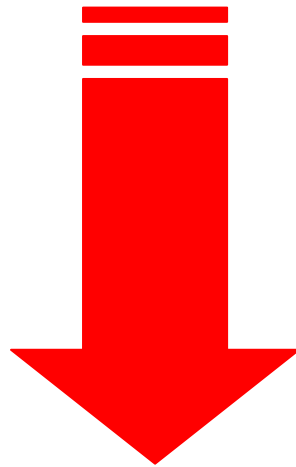
Download "The latest release version for download: DAS Edition v2.6.4":



Unzip [das_edition_v264.zip](#) and

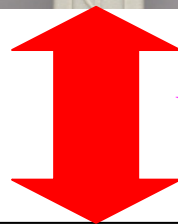
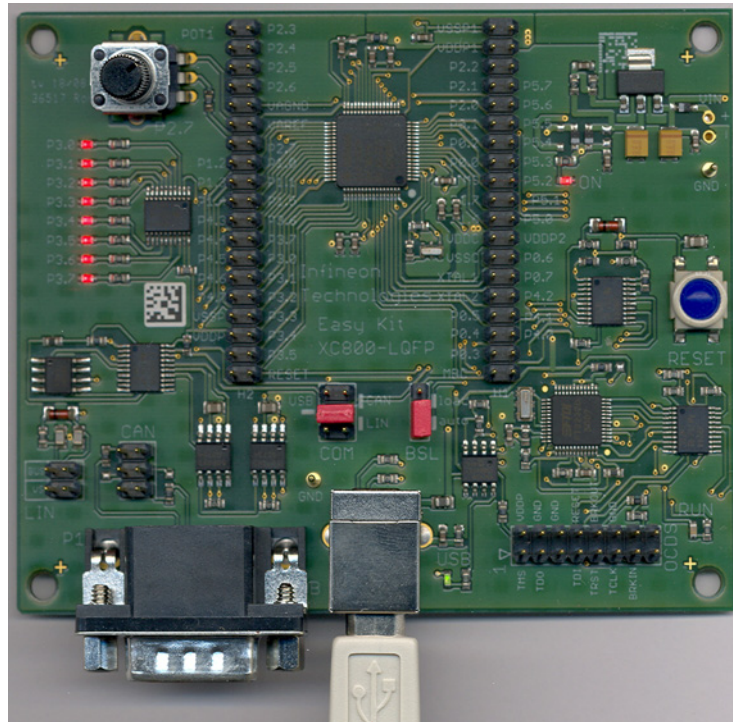
execute "DAS_v264_setup.exe" to install the DAS Server.



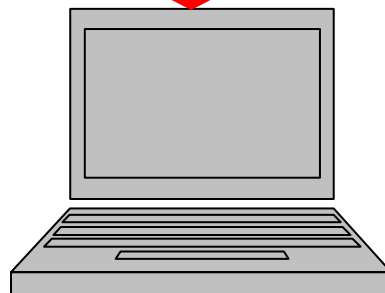


Click Finish

Connect the XC878 Easy Kit to the host computer:



USB Connection

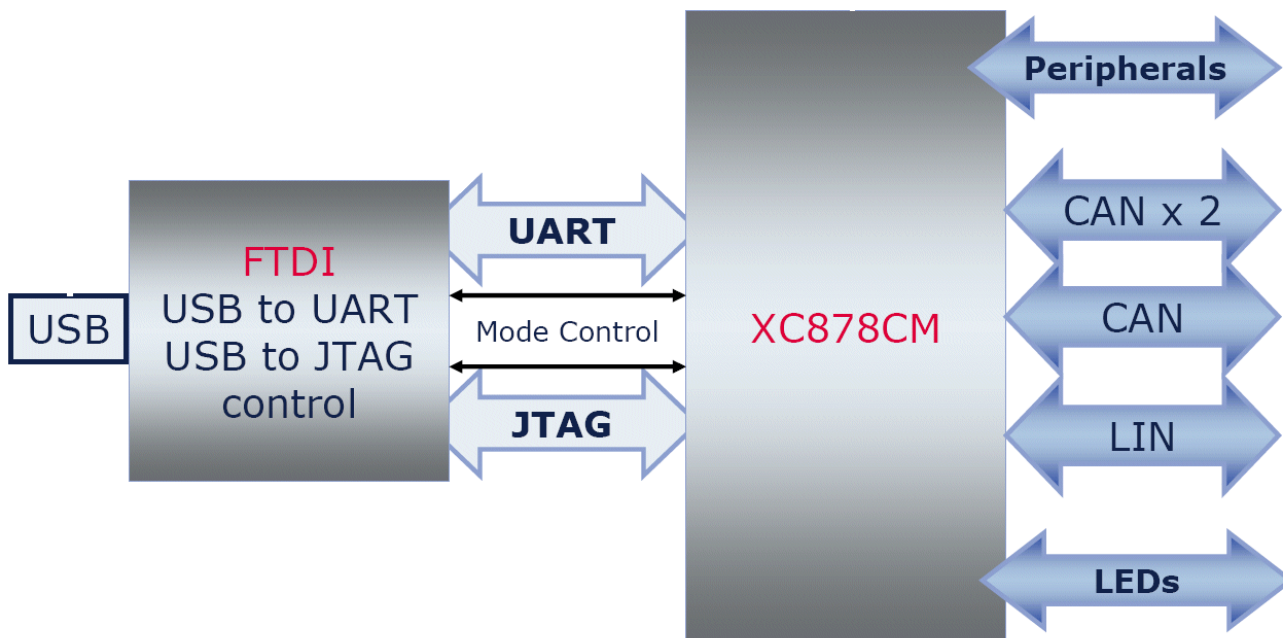


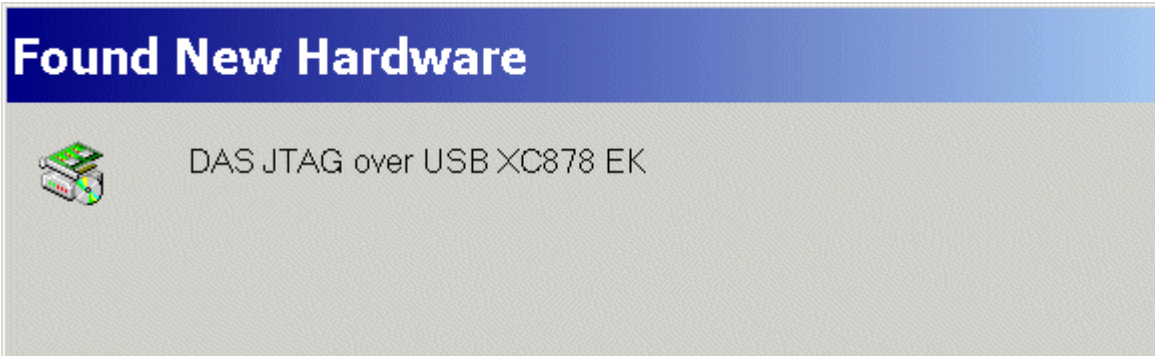
USB Connection:

.) 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:

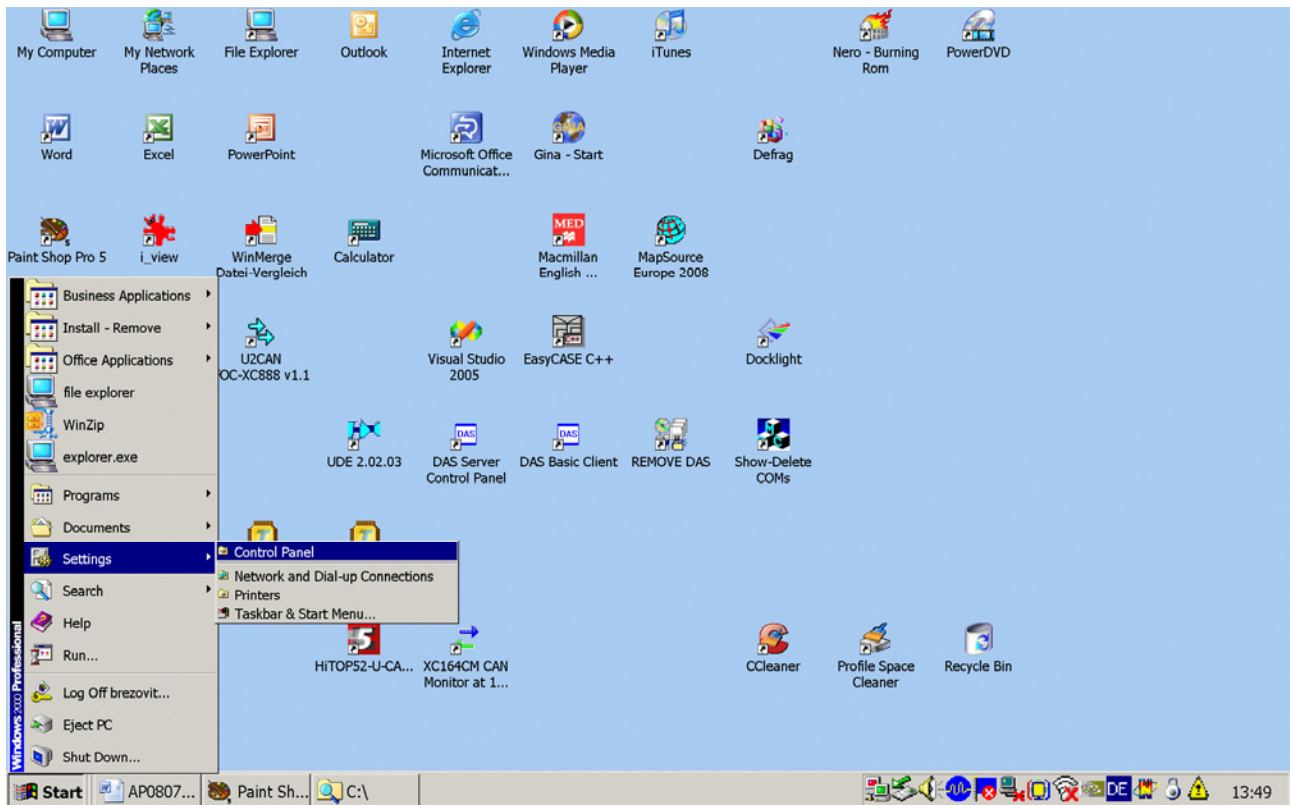
A USB driver is installed the first time while connecting the XC878 Easy Kit via USB to your host computer.

Note:

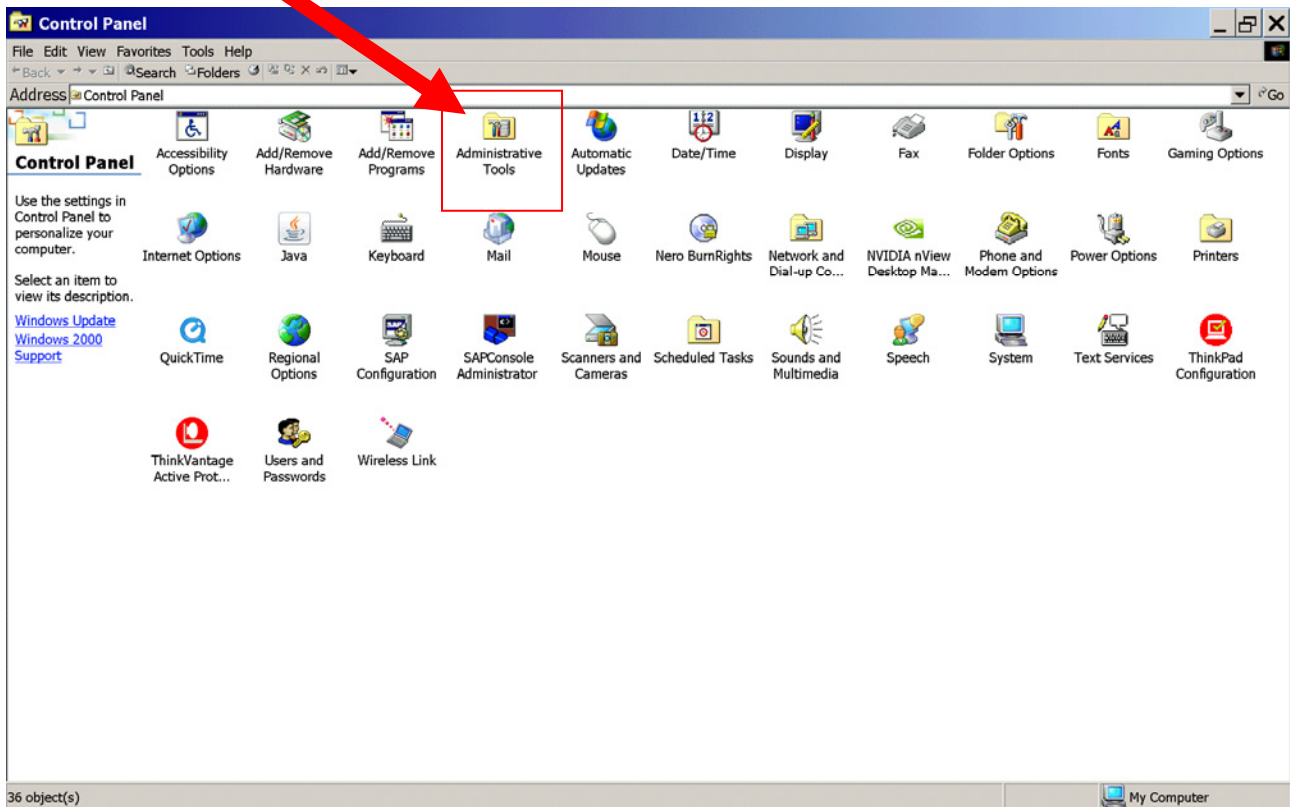
A default virtual COM Port is generated.

Using a Windows 2000 operating system, we are now going to search for the virtual COM Port which was generated after connecting our XC878 Easy Kit:

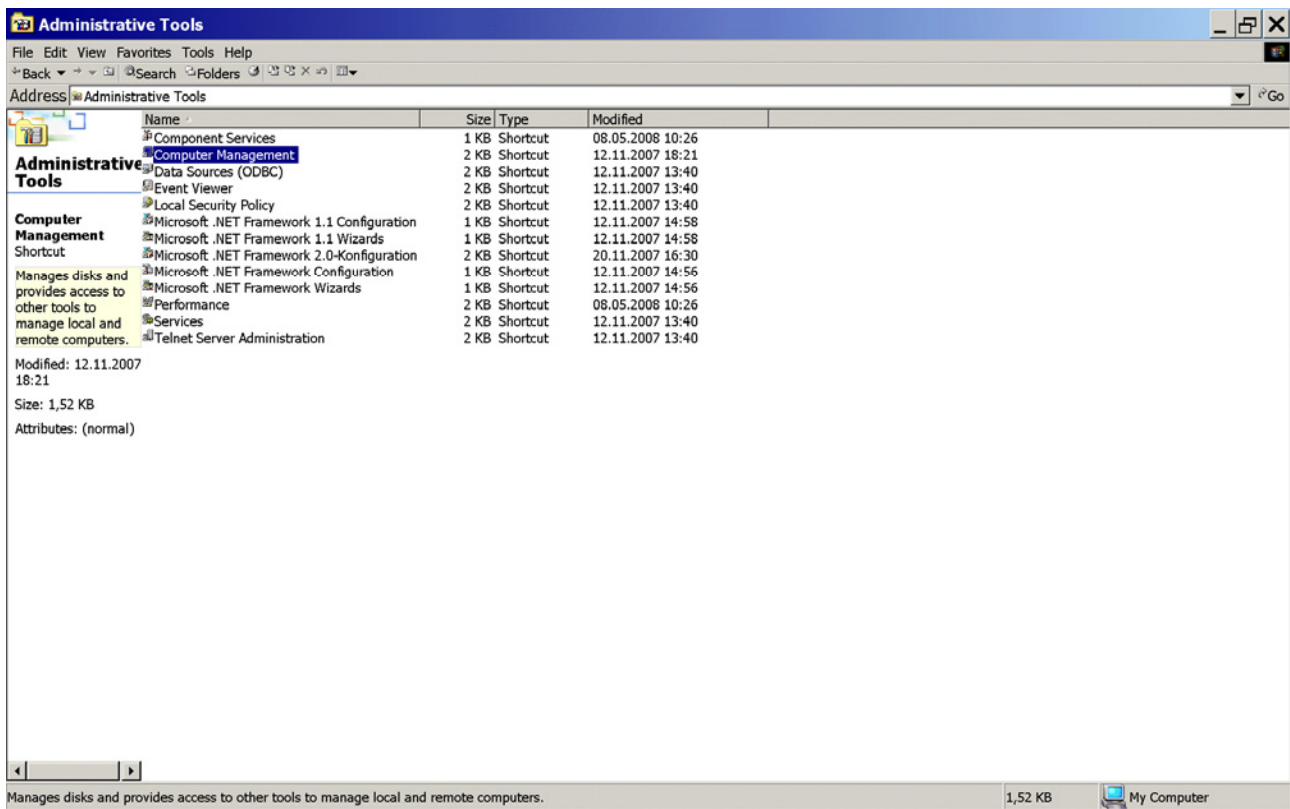
Start – Settings – Control Panel



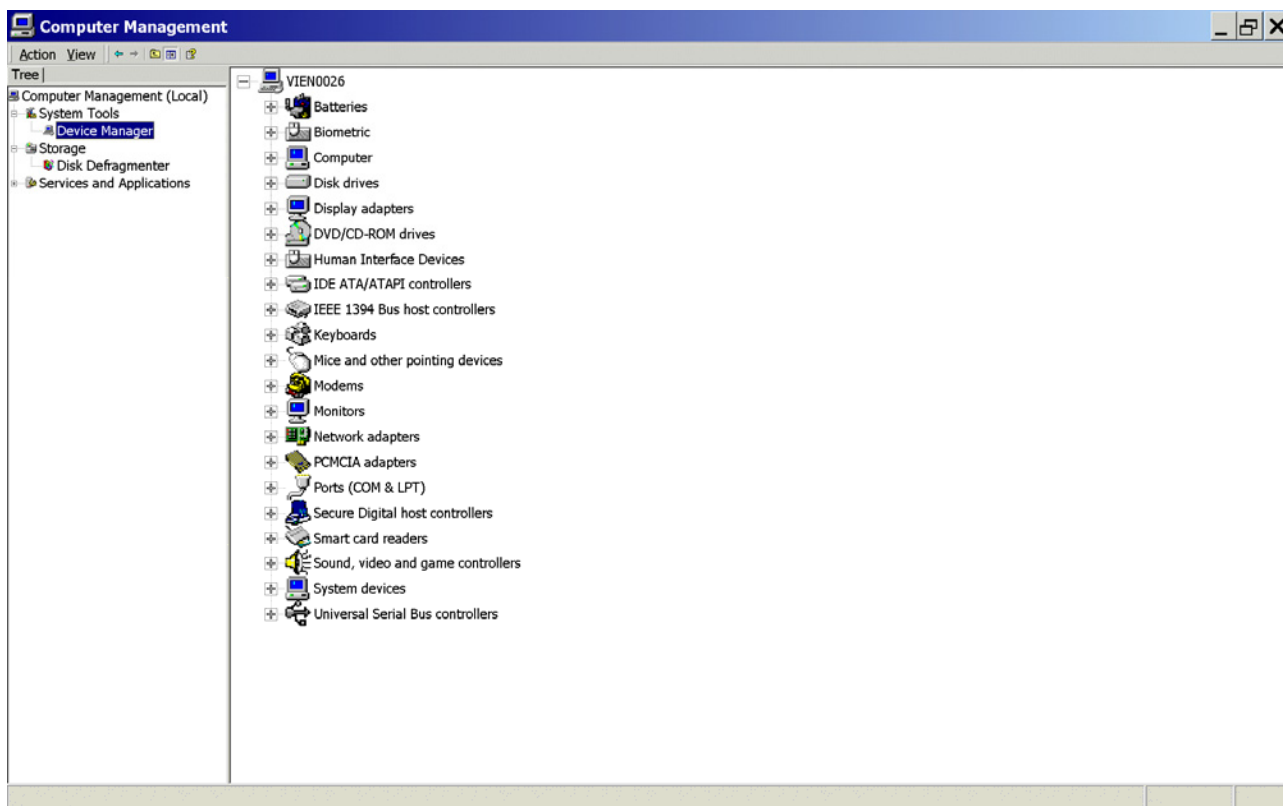
Double click: Administrative Tools



Double click: Computer Management

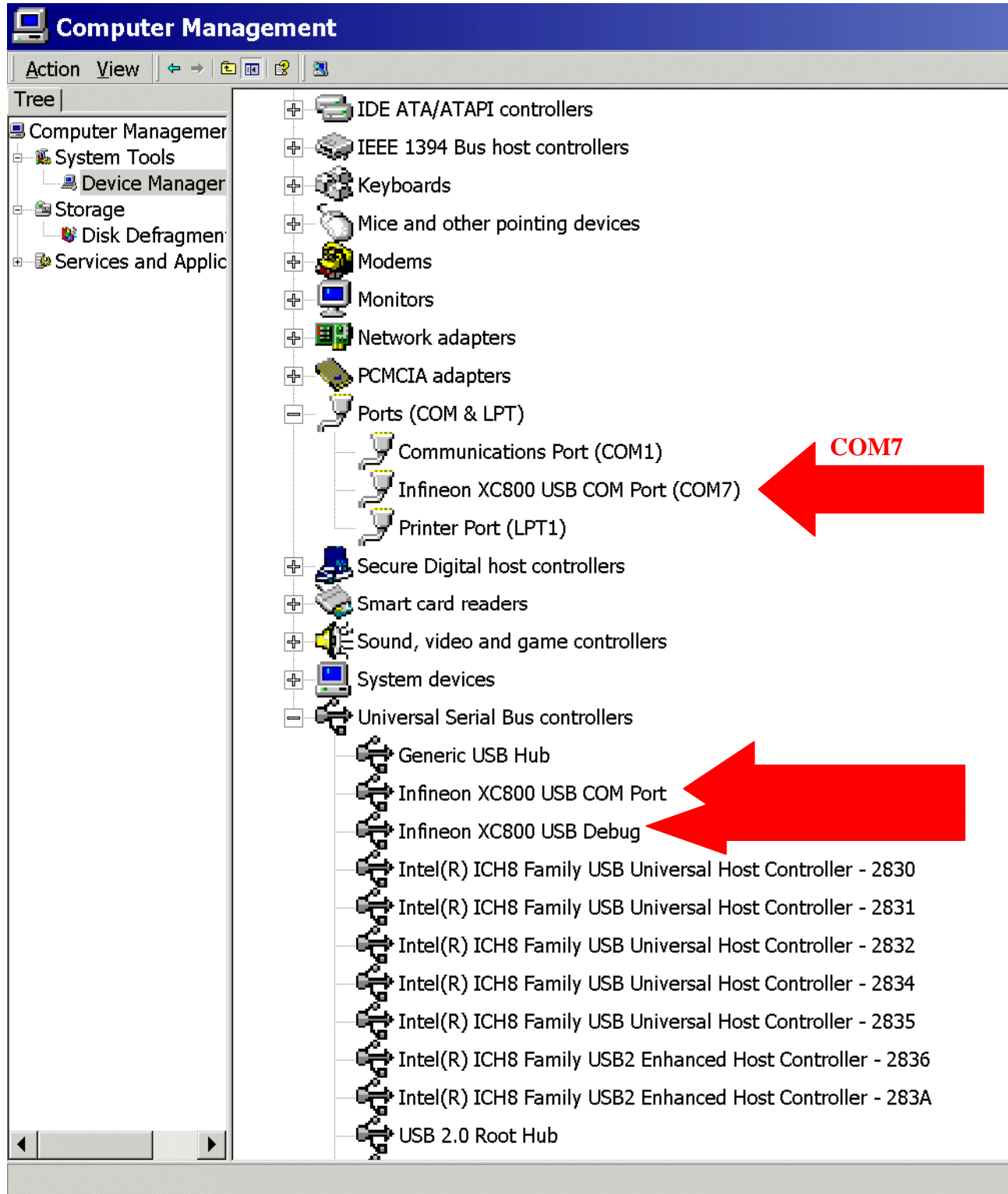


Click: Device Manager



Expand: Ports (COM & LPT):

Expand: Universal Serial Bus controllers:




2.) DAvE – Installation for XC878 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 XC878 microcontroller support/update (XC878 DIP file):

1.)

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


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

Title	Date	Version	Size
Development Tools			^
 XC878CLM DIP file for DAvE (Microcontroller Configuration Tool)-latest version (XC878CLM.zip)	08 Jul 2008	v1.1	9.1 MB

Unzip the zip-file "XC878CLM.zip" and save "XC878CLM.DIP"

@ e.g. D:\DAvE\XC878-2008-09-04\XC878.dip.

2.)

Start DAvE - ( click DAvE)

3.)

View

Setup Wizard

Default: • Installation

Forward>

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

Forward>

Select: D:\DAvE\XC878-2008-09-04

Forward>

Select: Available Products

click ✓ XC878CLM

Forward>

Install

End

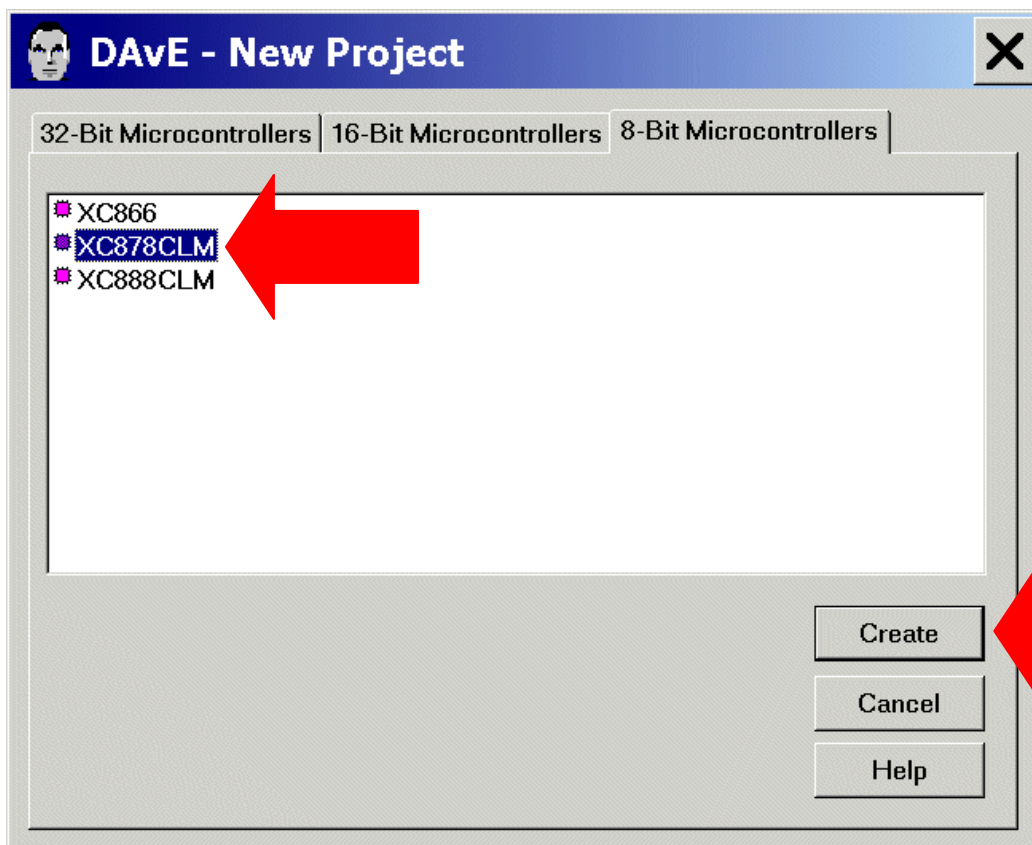
4.) DAvE is now ready to generate code for the XC878 microcontroller.

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



Start the program generator DAVe and select the XC878 microcontroller:

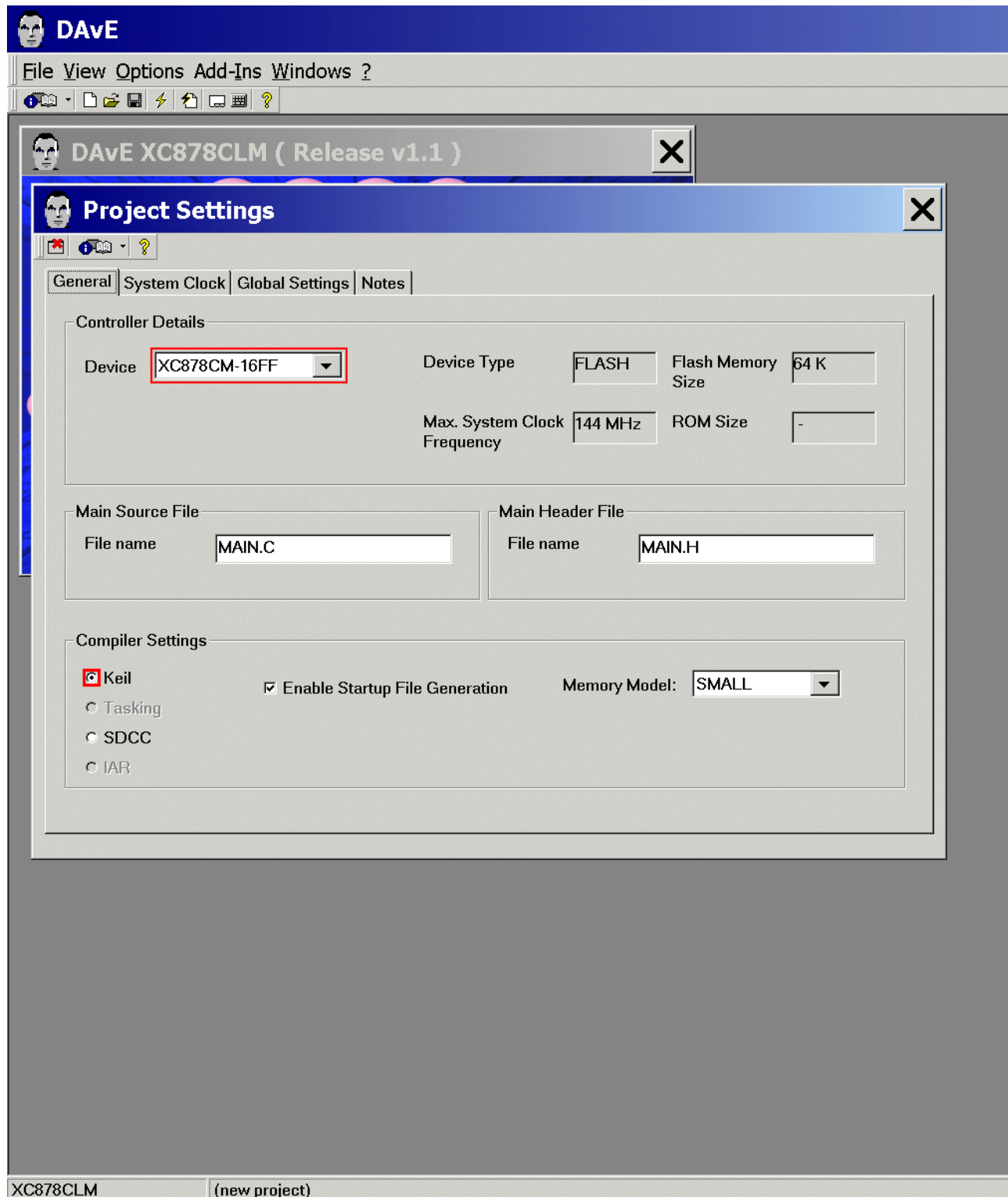
File
New
8-Bit Microcontrollers
select XC878CLM
Create



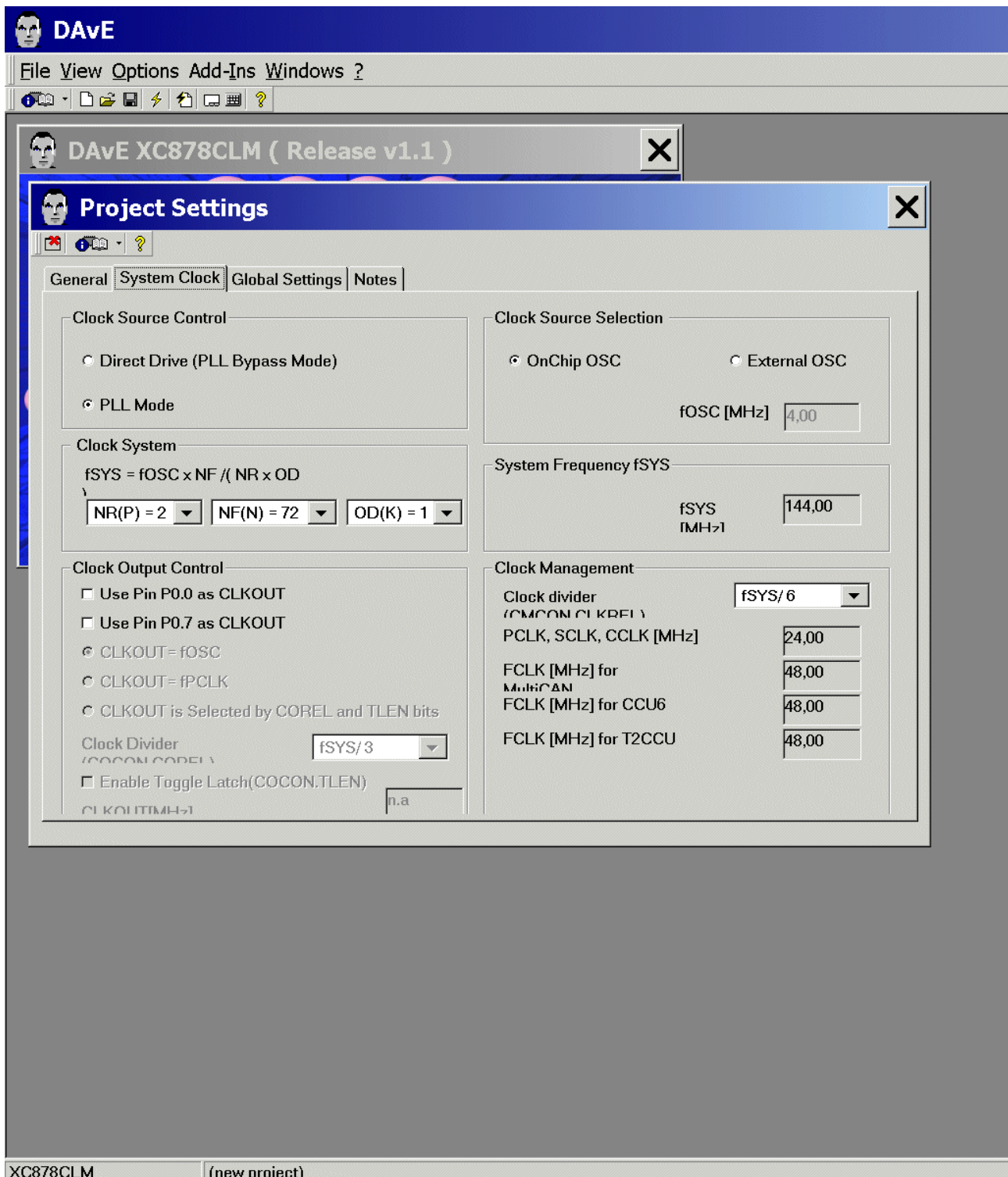
Choose the Project Settings as you can see in the following screenshots:

General: Controller Details: Device: check/select XC878CM-16FF

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



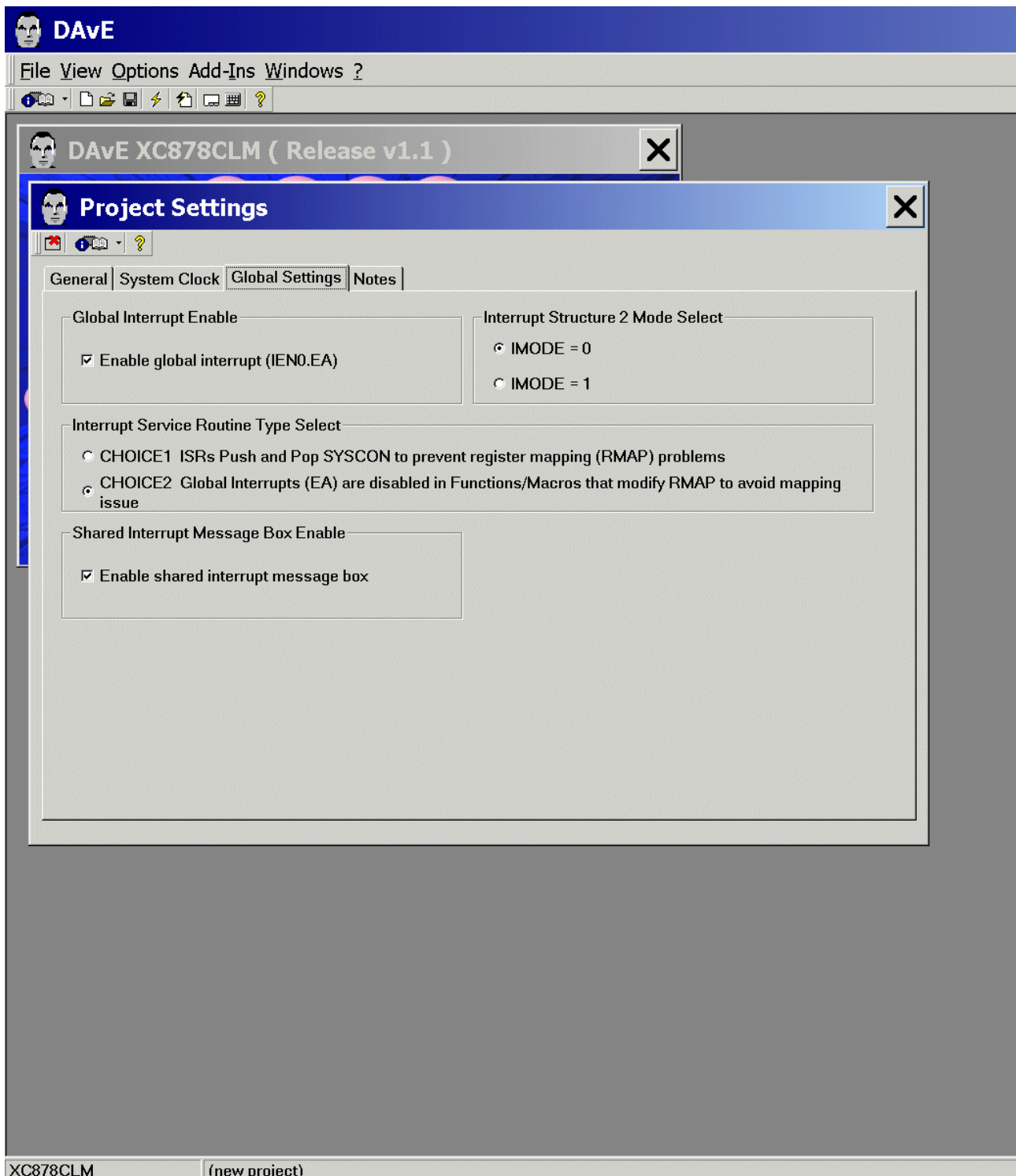
System Clock: (do nothing)




Note:
CPU clock is 24 MHz.



Global Settings: (do not change configuration)



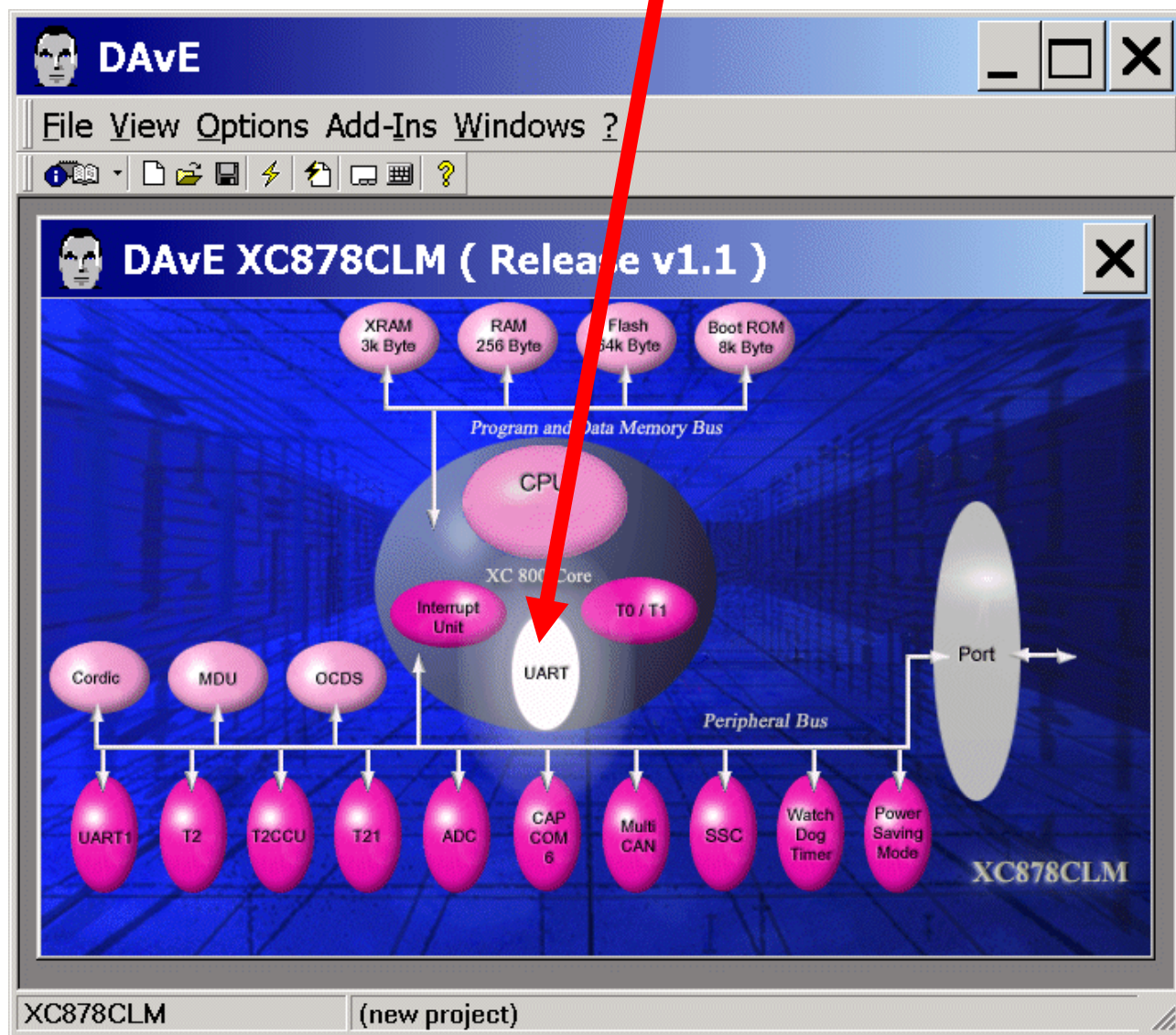
Notes: If you wish, you can insert your comments here.

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.

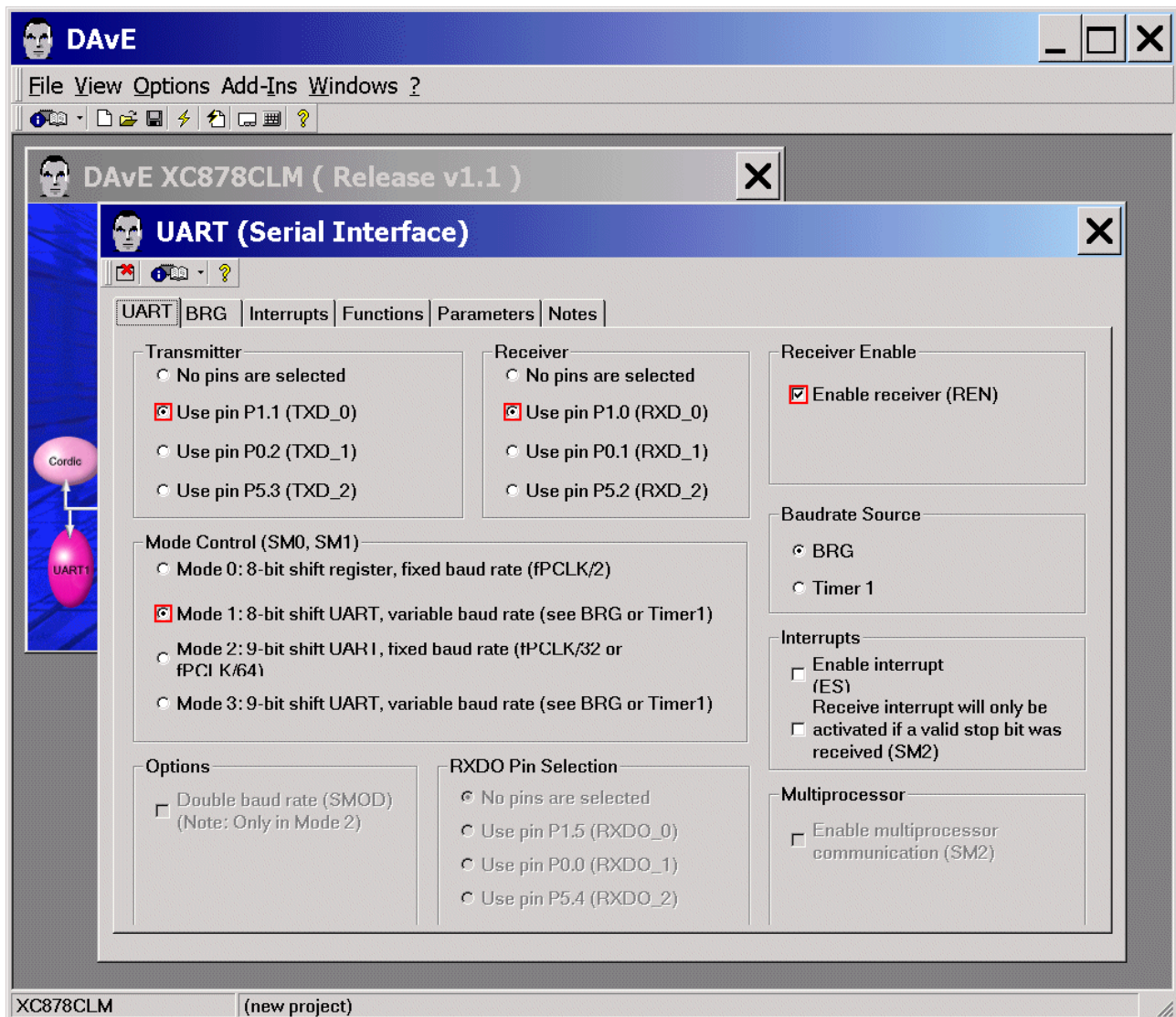


UART: Transmitter: **click** ☐ Use pin P1.1 (TXD_0)

UART: Receiver: **click** ☐ Use pin P1.0 (RXD_0)

UART: Receiver Enable: **tick** ☒ Enable receiver (REN)

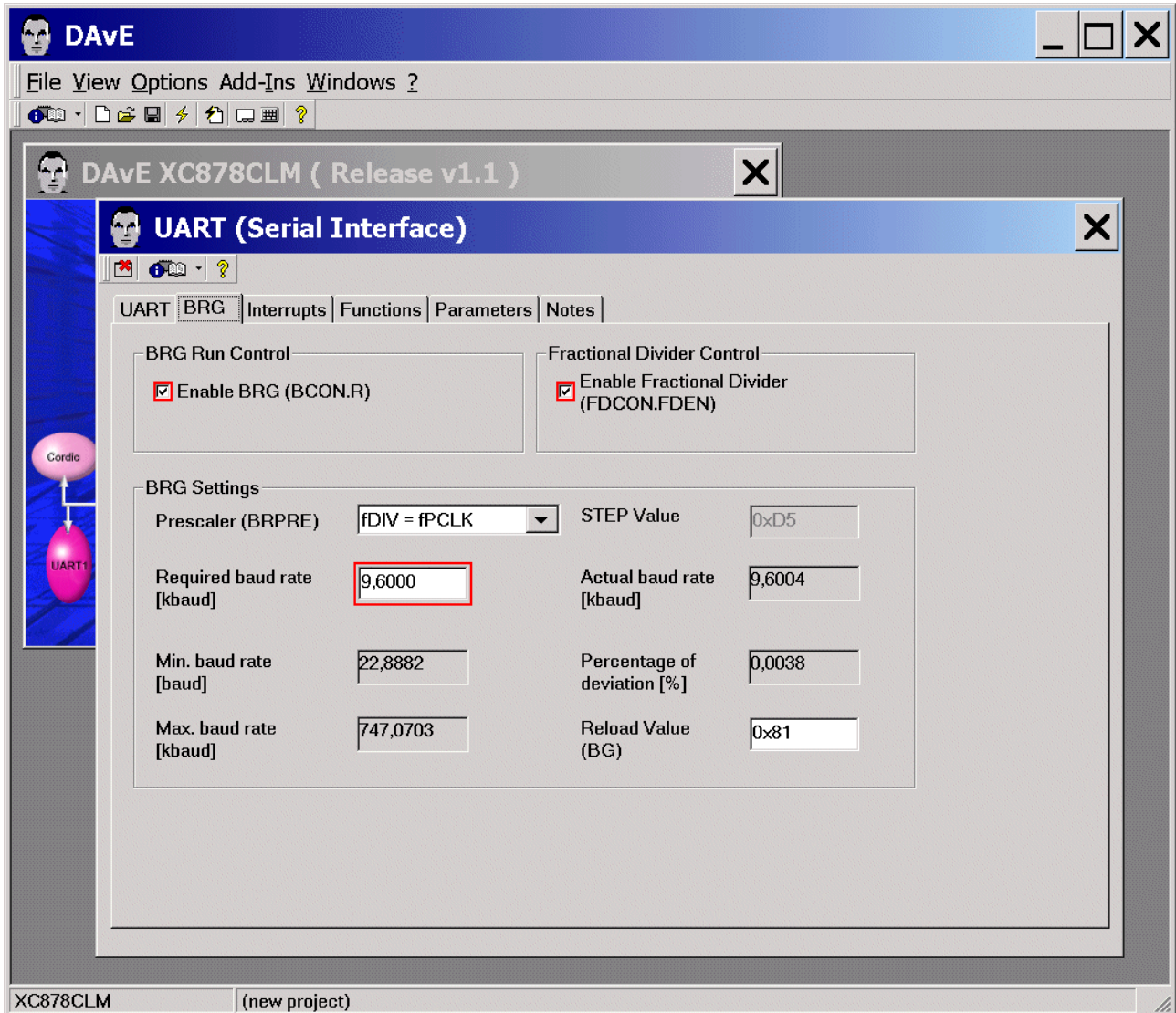
UART: Mode Control: **click** ☐ Mode 1: 8-bit shift UART, variable baud rate (see BRG or Timer1)



Note:

The RS232 serial interface (UART pins P1.0 and P1.1) is available via the **USB port** as virtual COM port (e.g. COM7) which converts the TTL-UART-signals to USB-signals.

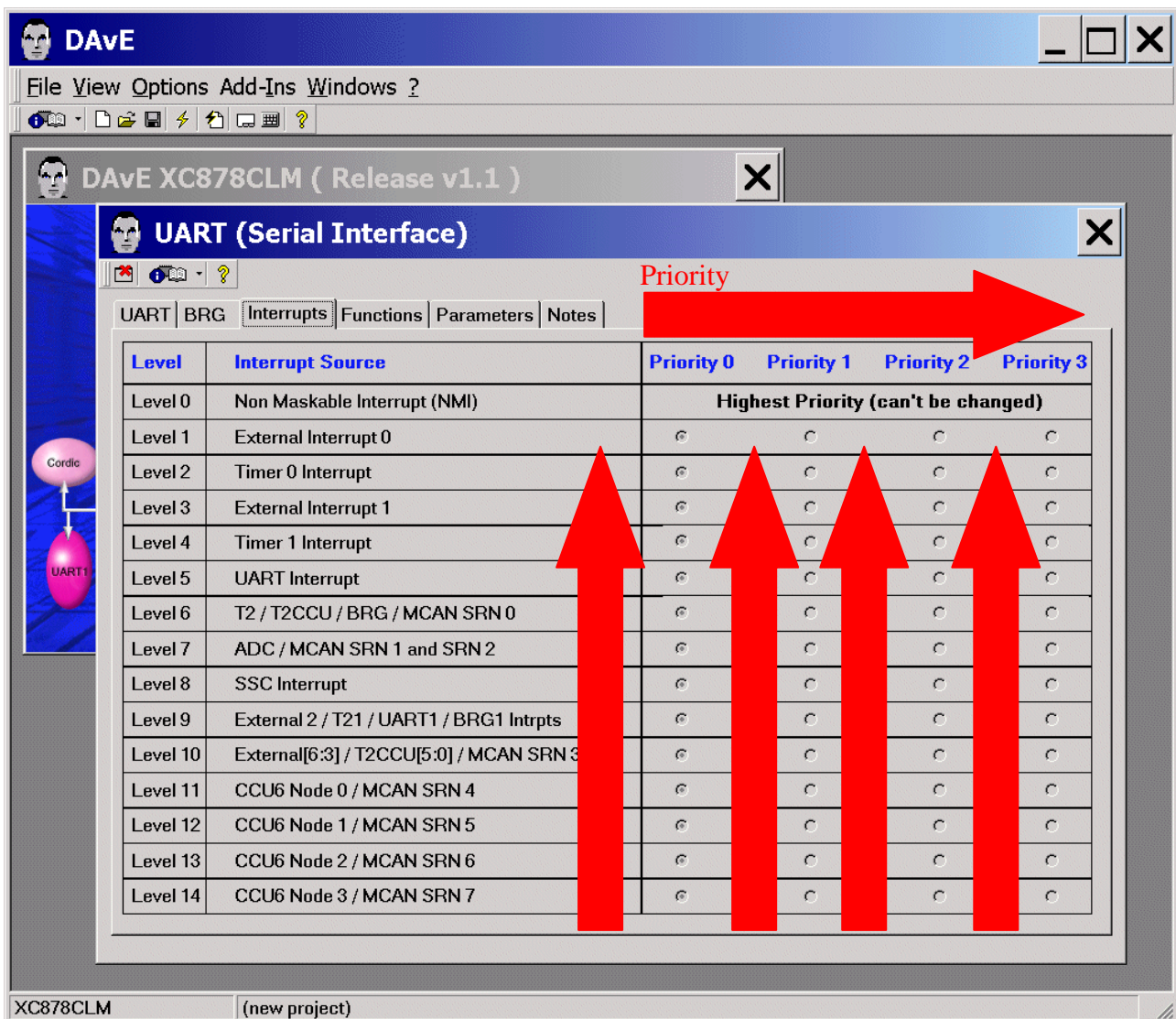
BRG: BRG Run Control: **check/tick** ✓ Enable BRG
 BRG: Fractional Divider Control: **tick** ✓ Enable Fractional Divider
 BRG: BRG Settings: Required baud rate [kbaud] **insert** 9,600 <ENTER>



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	T2 / T2CCU / BRG / MCAN SRN 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 7	ADC / MCAN SRN 1 and SRN 2	<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] / T2CCU[5:0] / MCAN SRN 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 11	CCU6 Node 0 / MCAN SRN 4	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 12	CCU6 Node 1 / MCAN SRN 5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 13	CCU6 Node 2 / MCAN SRN 6	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 14	CCU6 Node 3 / MCAN SRN 7	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



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 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.

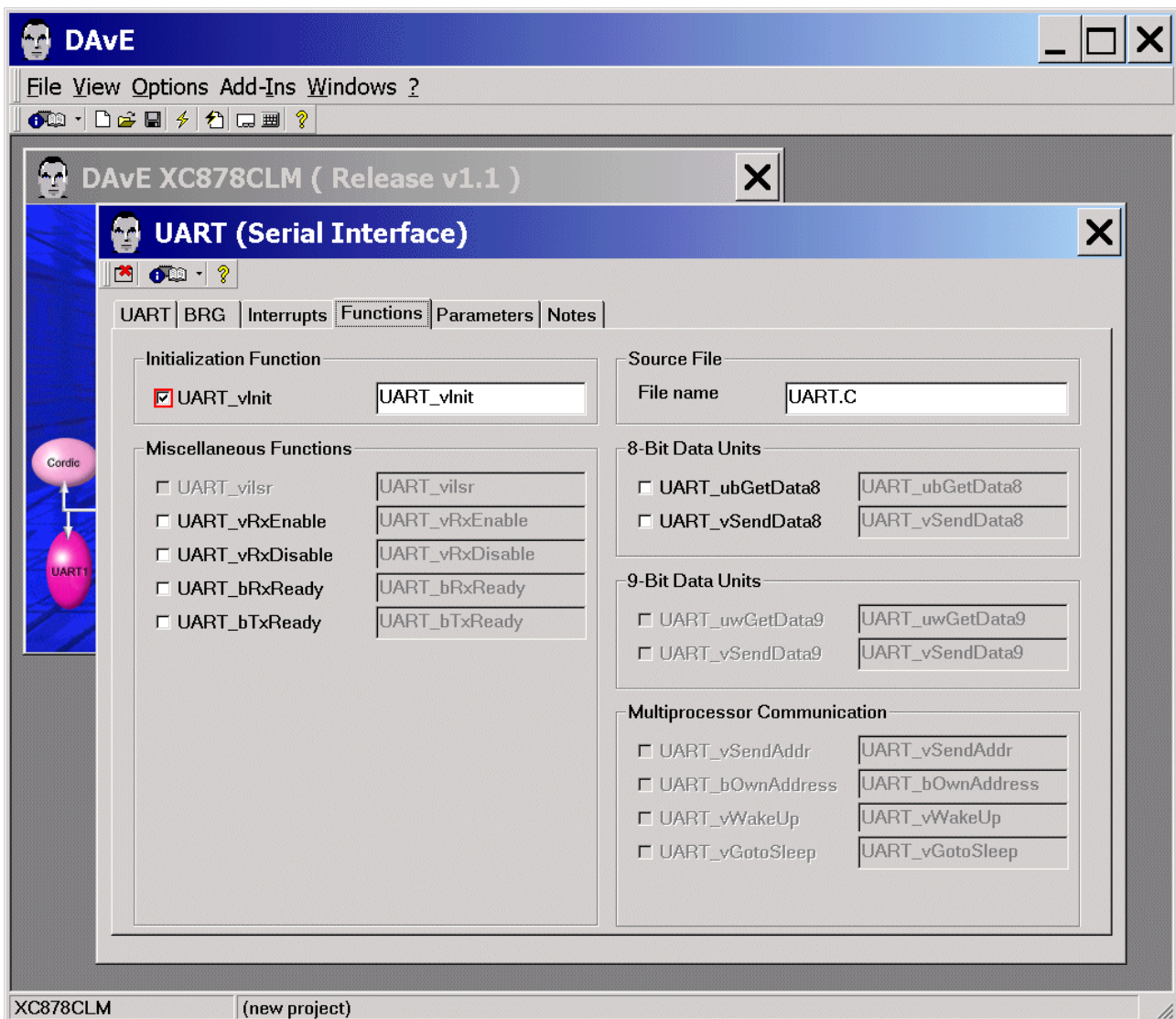
Each interrupt source can be programmed to any of the four interrupt priorities (0-3).

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: **tick** ✓ UART_vInit

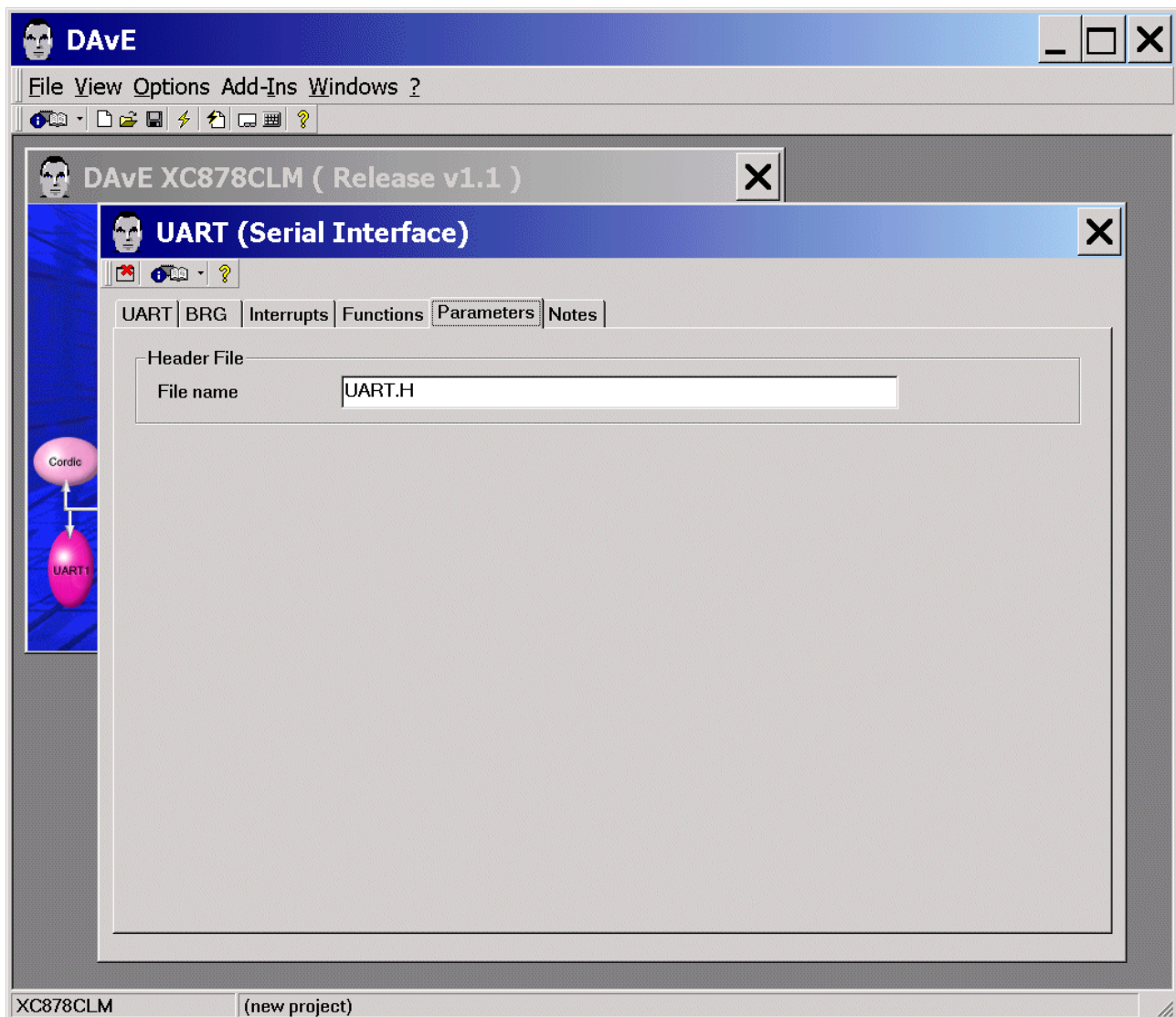


Note:


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



Parameters: (do nothing)

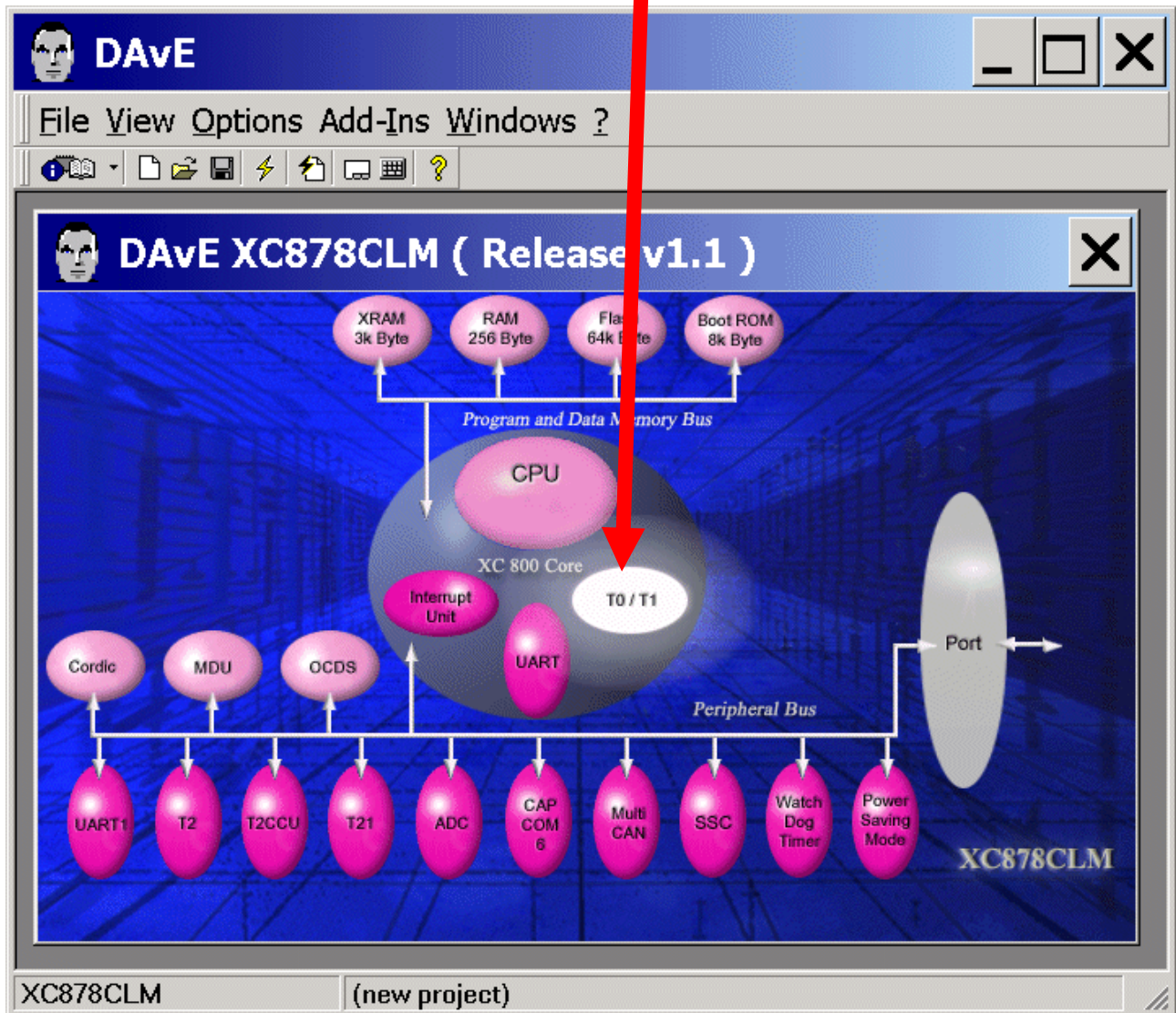


Notes: If you wish, you can insert your comments here.

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

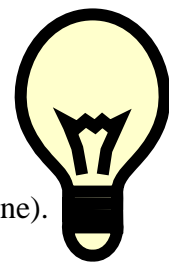
Configure Timer T0:

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



Note:

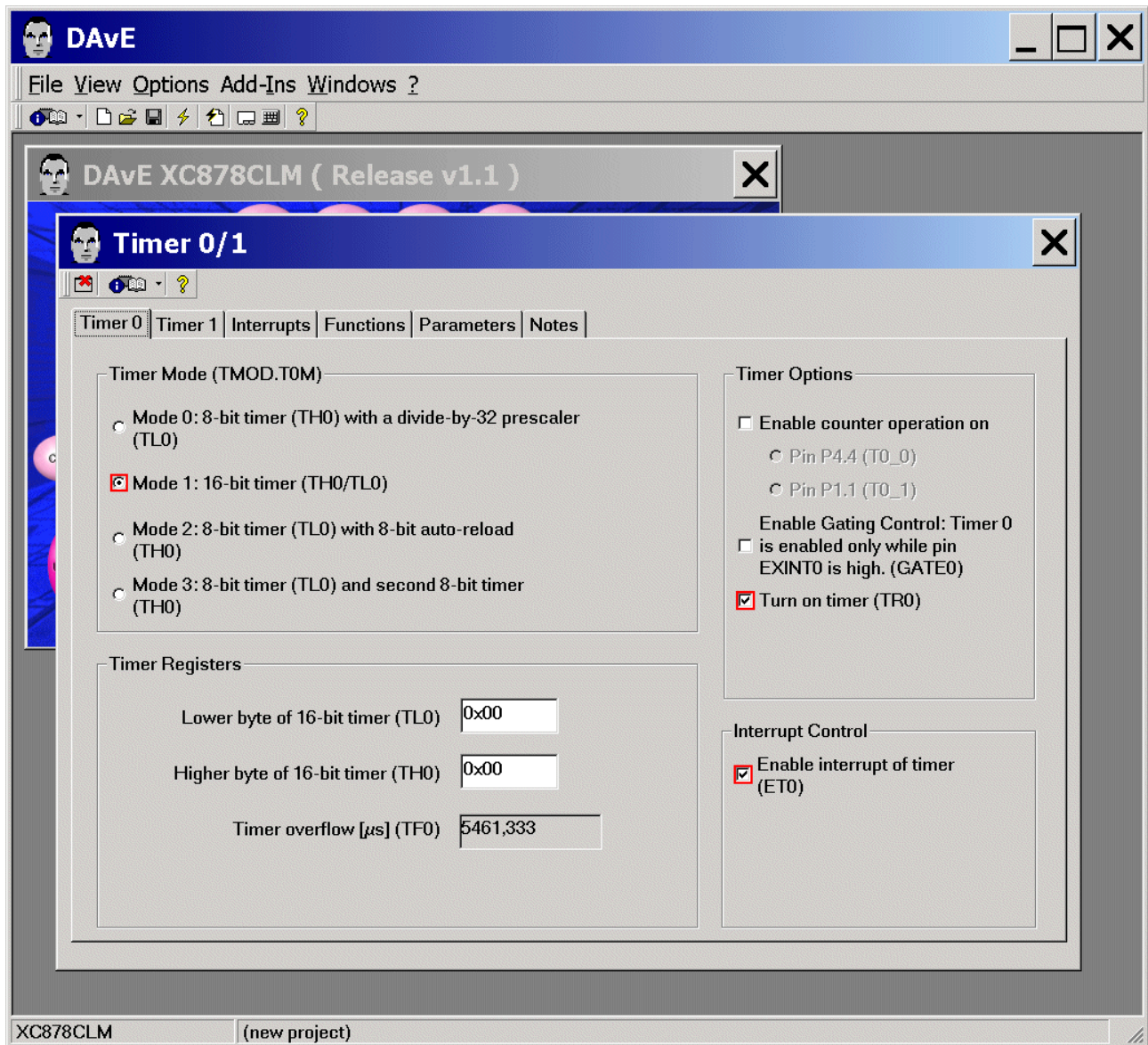
The LEDs on Port_3 will be blinking (if selected in the main menu) with a frequency of about 1 second (done in the Timer_0-Interrupt-Service-Routine). Therefore we have to configure Timer_0.



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

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

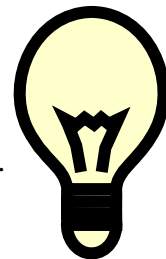
Timer0: Interrupt Control: **tick**  Enable interrupt of timer (ET0)



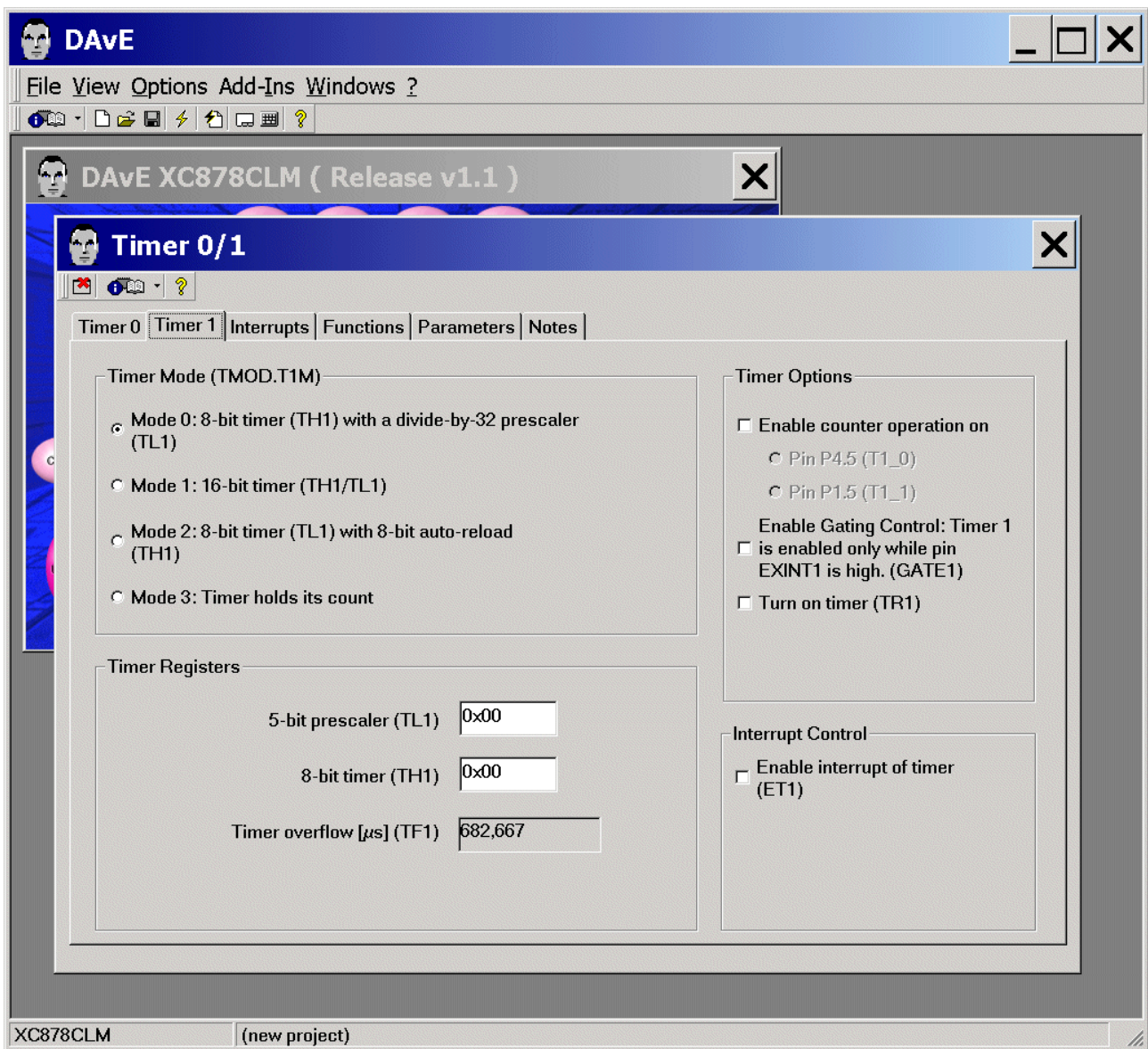
Note:

We need 183 Timer_0 overflows to achieve an approximate 1 second delay. This will be handled in the Timer_0 interrupt function.

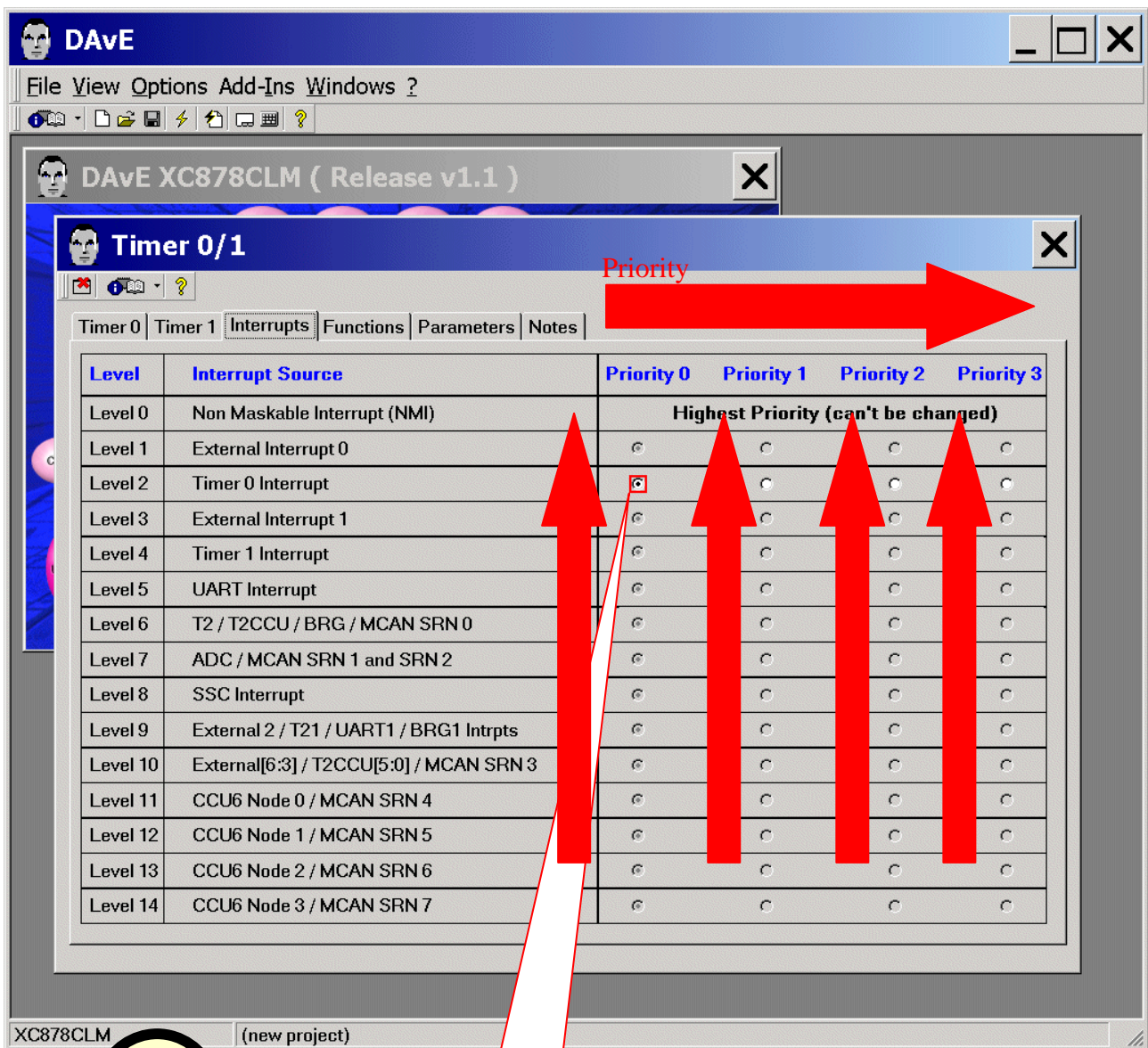
$$183 * 5461,333 \mu s = 0,9994 s.$$



Timer1: do nothing (not used)



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 checked="" 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	T2 / T2CCU / BRG / MCAN SRN 0	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 7	ADC / MCAN SRN 1 and SRN 2	<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] / T2CCU[5:0] / MCAN SRN 3	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 11	CCU6 Node 0 / MCAN SRN 4	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 12	CCU6 Node 1 / MCAN SRN 5	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 13	CCU6 Node 2 / MCAN SRN 6	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Level 14	CCU6 Node 3 / MCAN SRN 7	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>



Interrupt of Timer_0
is enabled, ET0 = 1

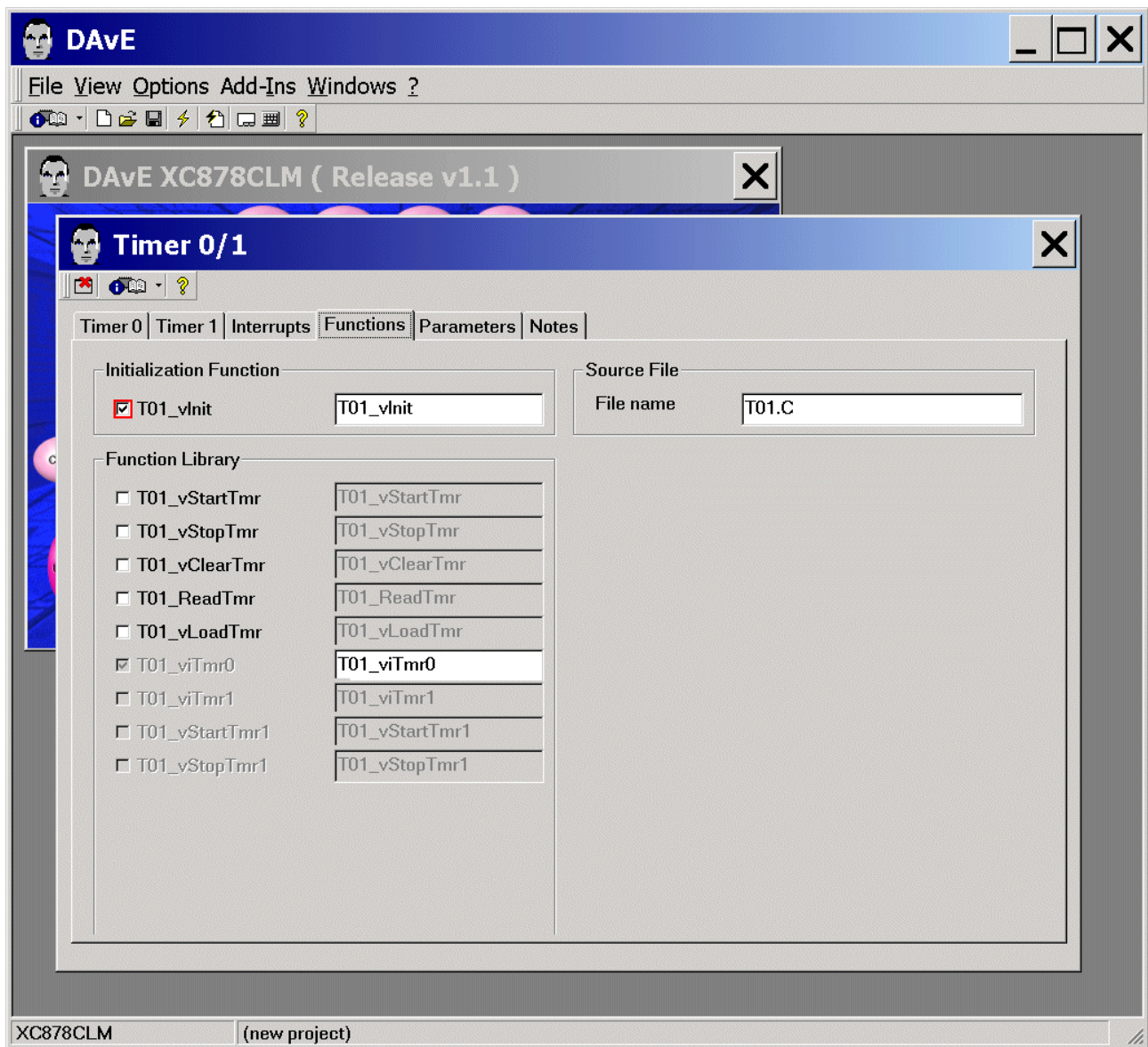
Note (Source: User's Manual):

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: **tick** ✓ T01_vInit



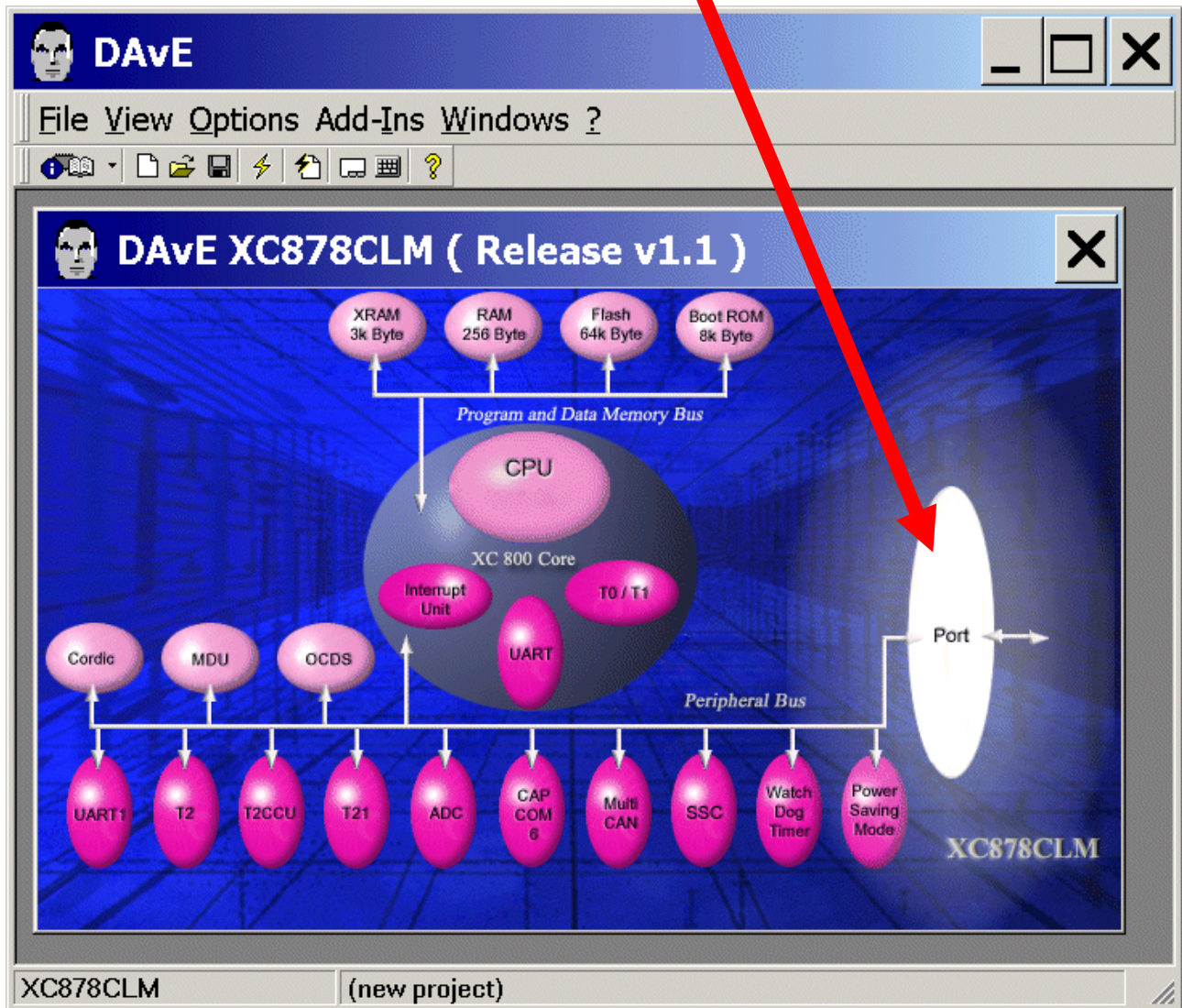
Parameters: (do nothing)

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.



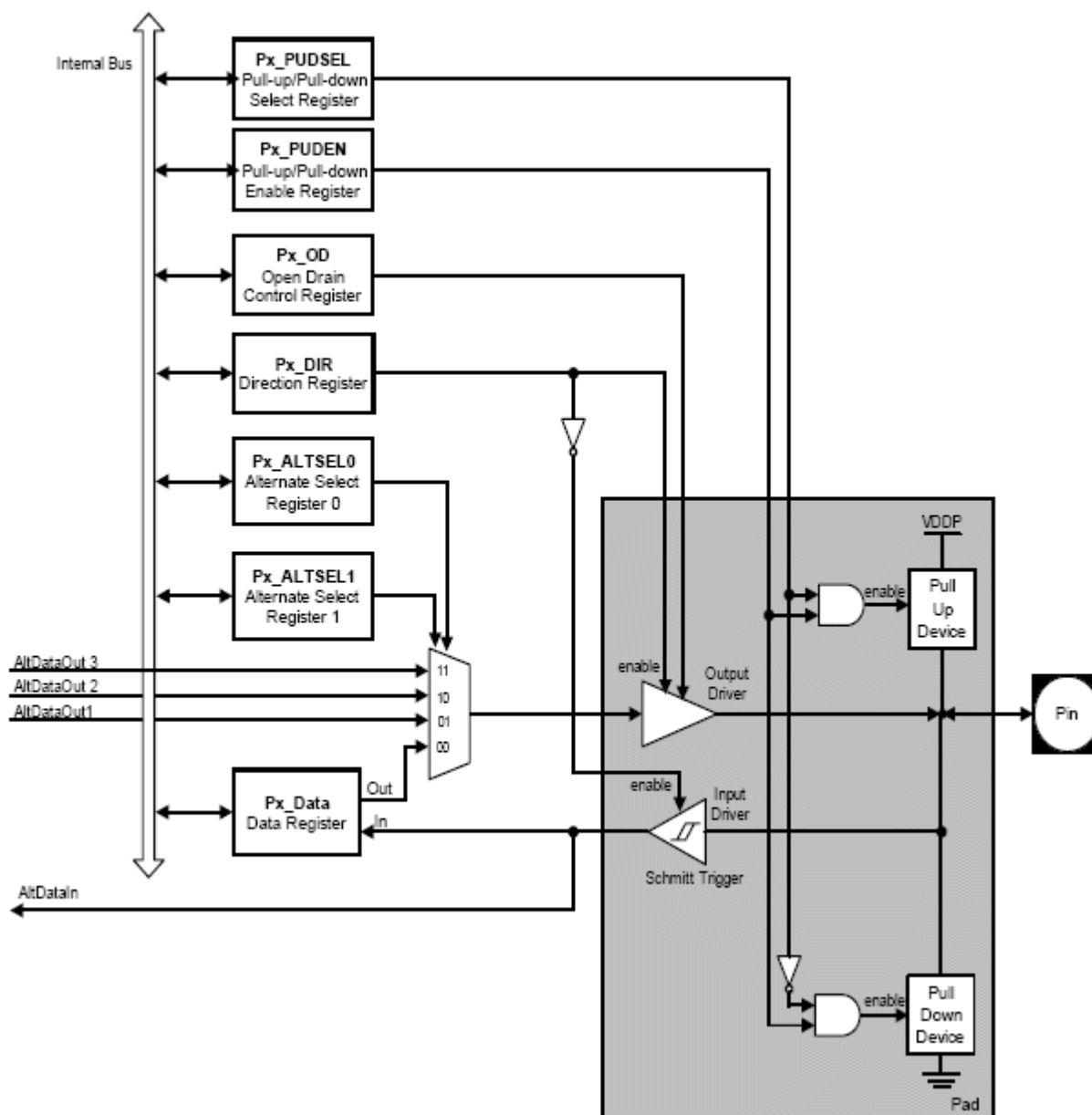
Note:

The User LEDs (red) are connected to Port_3.

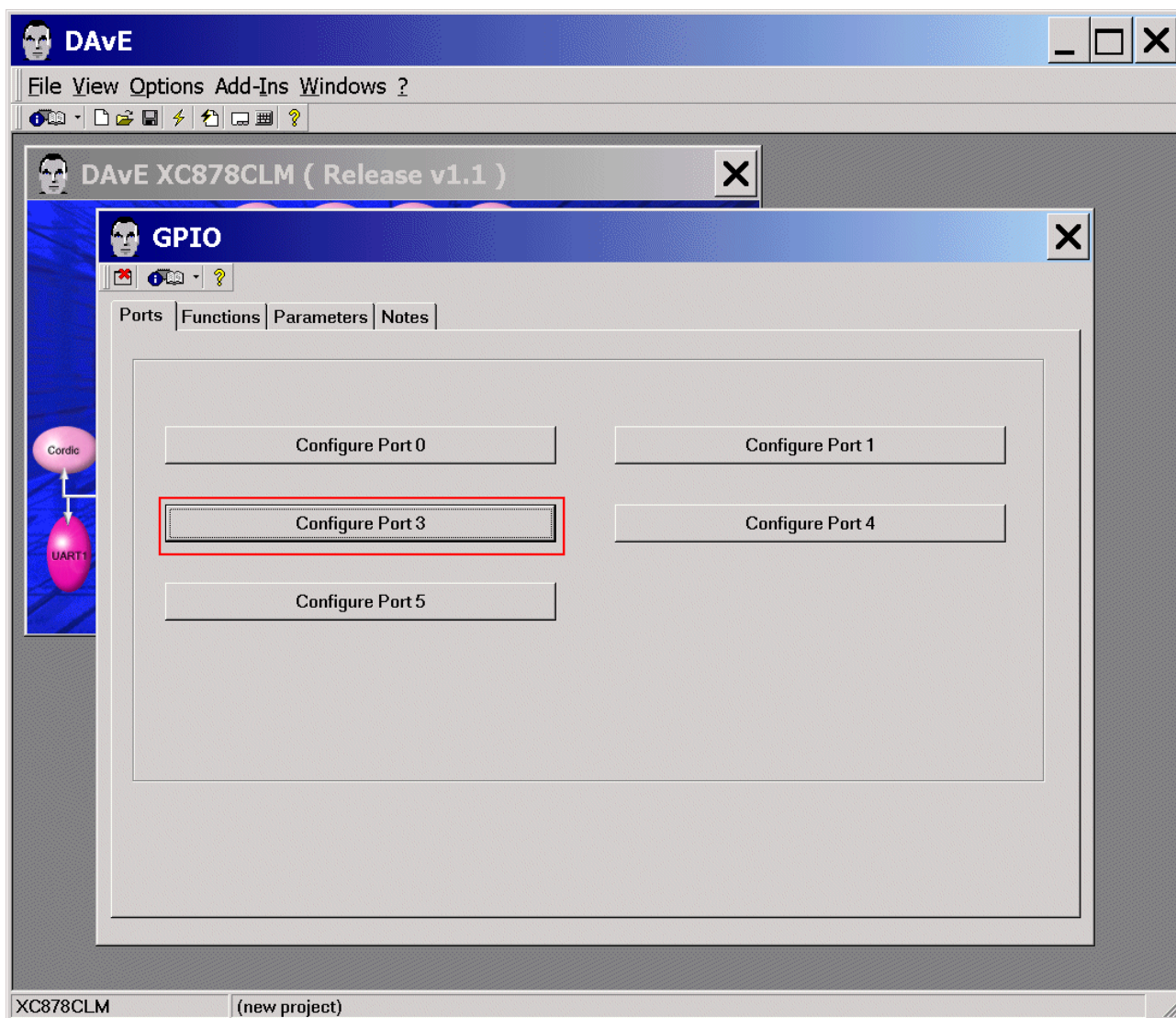




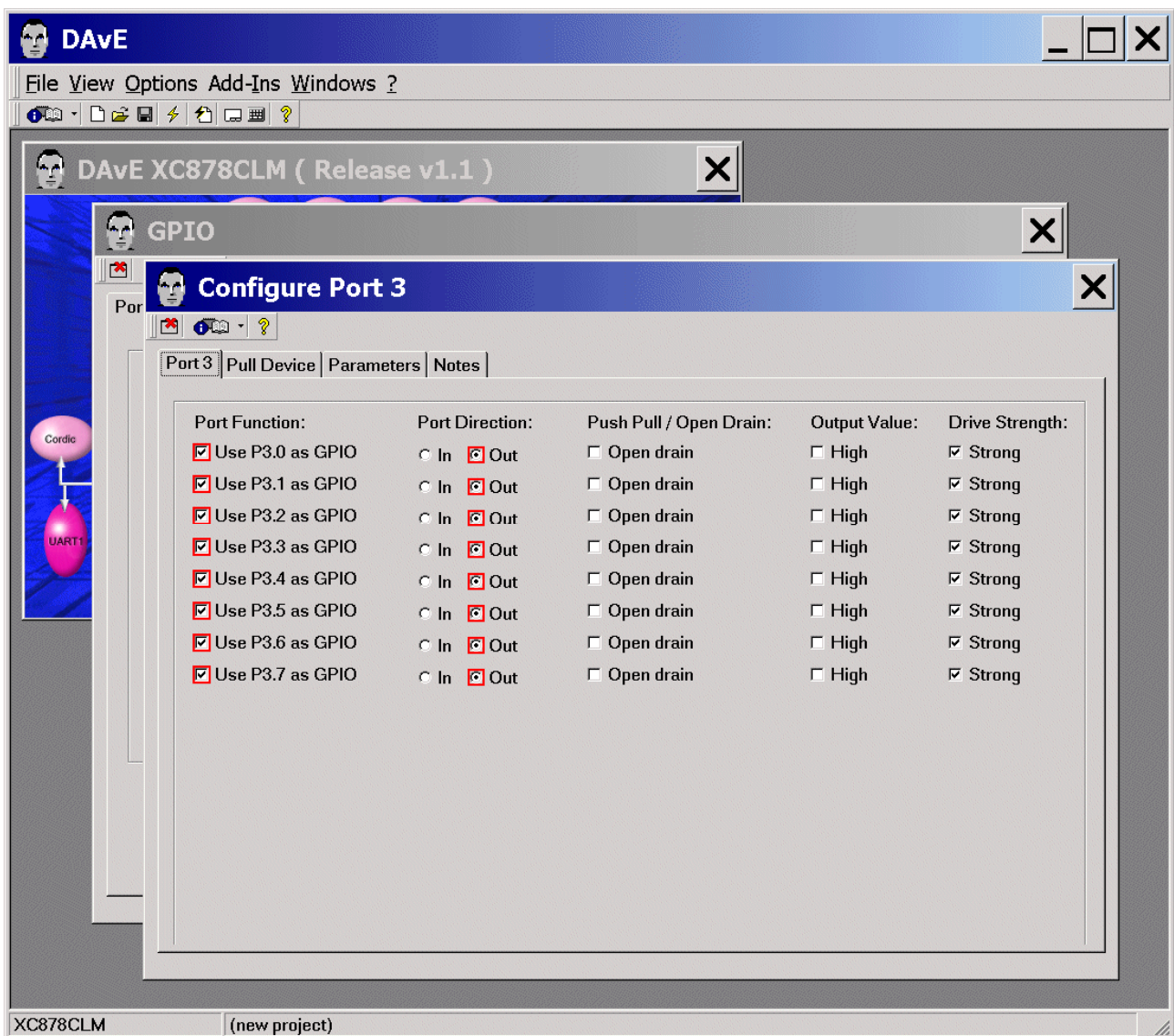
Additional information: Parallel Ports – General Structure (Source: User's Manual):



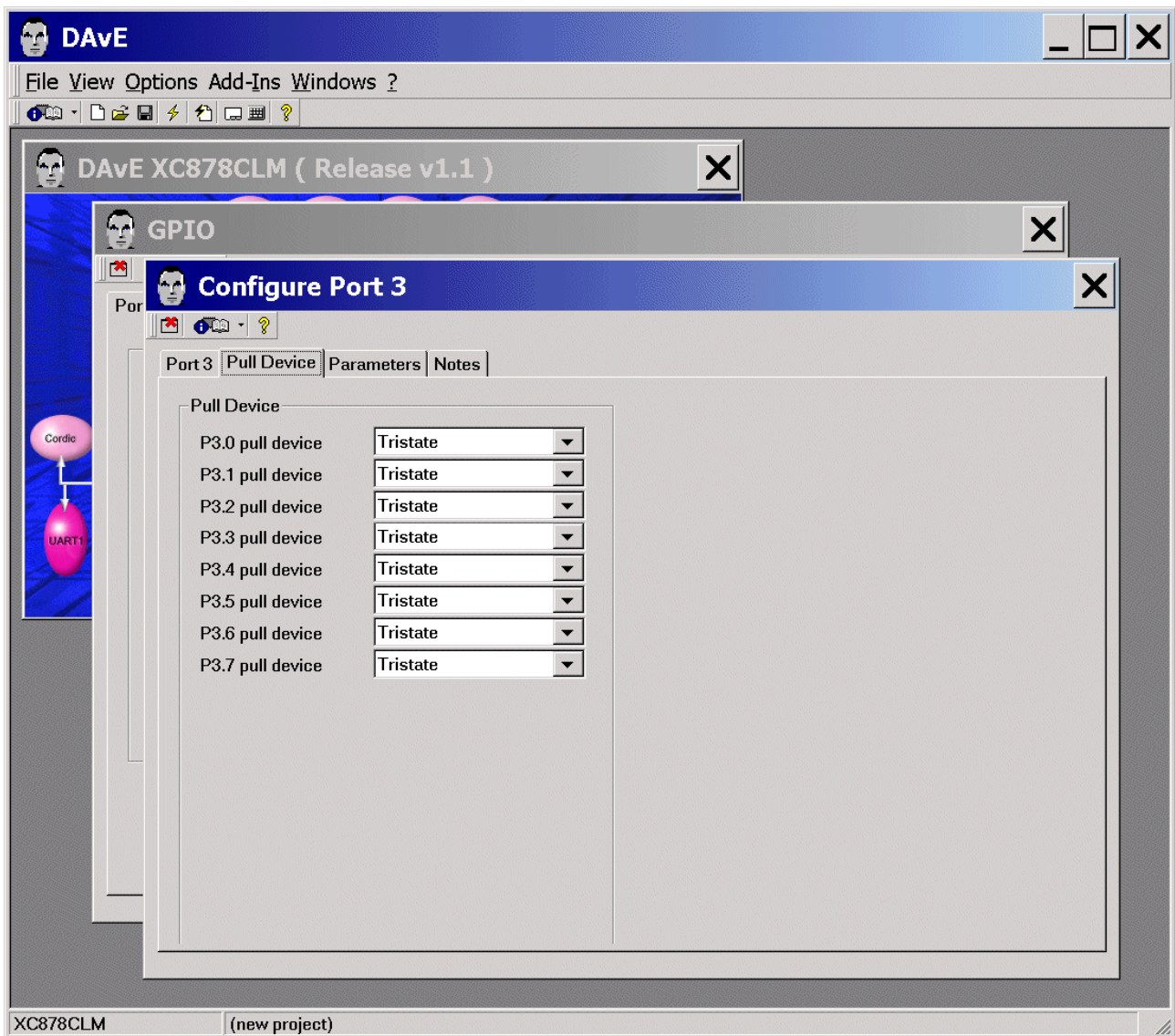
Ports: click "Configure Port 3"



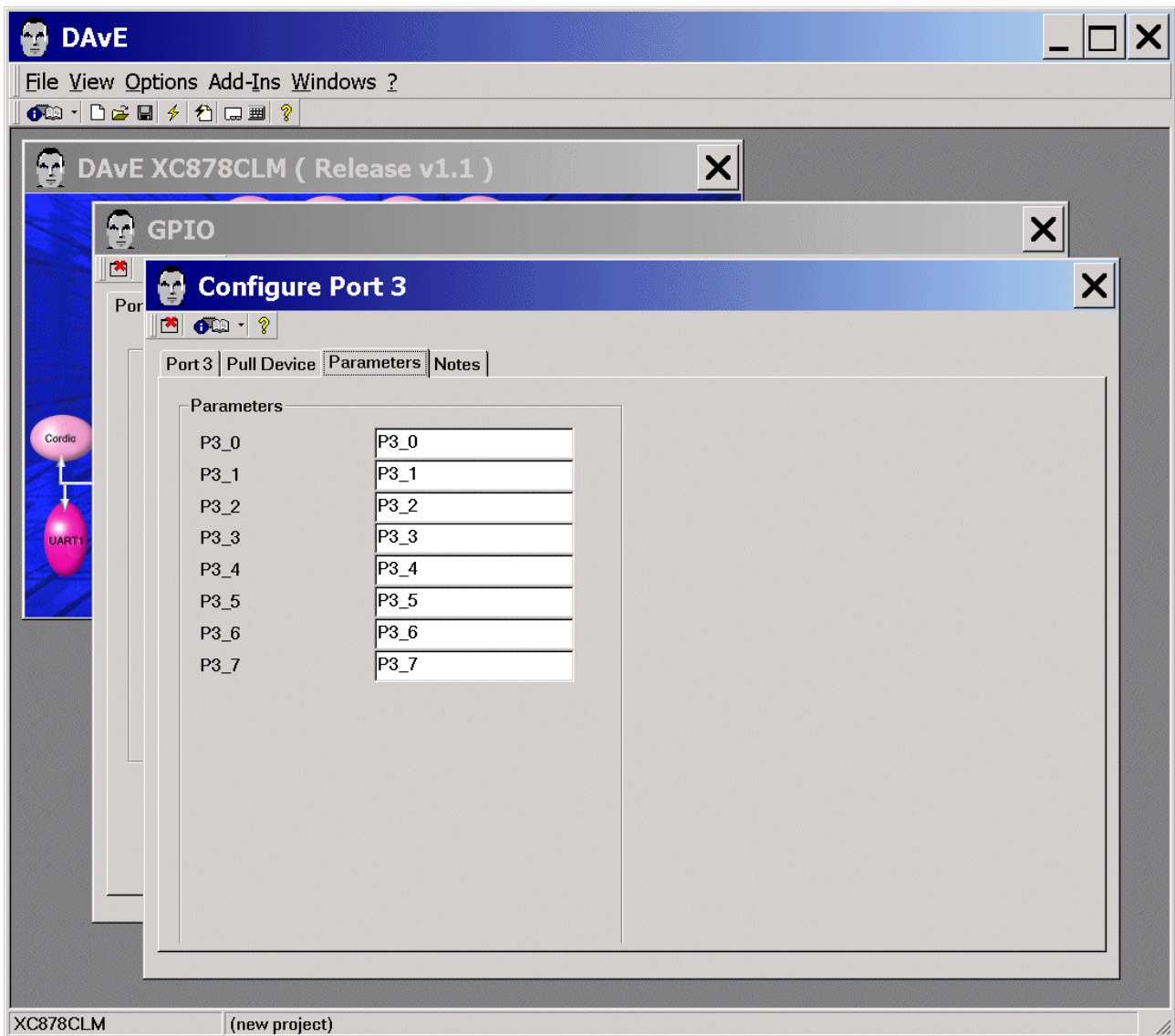
Port 3: Port Function: **tick** ✓ Use P3.0 as GPIO - Port Direction: **click** Ⓐ Out
 Port 3: Port Function: **tick** ✓ Use P3.1 as GPIO - Port Direction: **click** Ⓐ Out
 Port 3: Port Function: **tick** ✓ Use P3.2 as GPIO - Port Direction: **click** Ⓐ Out
 Port 3: Port Function: **tick** ✓ Use P3.3 as GPIO - Port Direction: **click** Ⓐ Out
 Port 3: Port Function: **tick** ✓ Use P3.4 as GPIO - Port Direction: **click** Ⓐ Out
 Port 3: Port Function: **tick** ✓ Use P3.5 as GPIO - Port Direction: **click** Ⓐ Out
 Port 3: Port Function: **tick** ✓ Use P3.6 as GPIO - Port Direction: **click** Ⓐ Out
 Port 3: Port Function: **tick** ✓ Use P3.7 as GPIO - Port Direction: **click** Ⓐ Out



Pull Device: (do nothing)



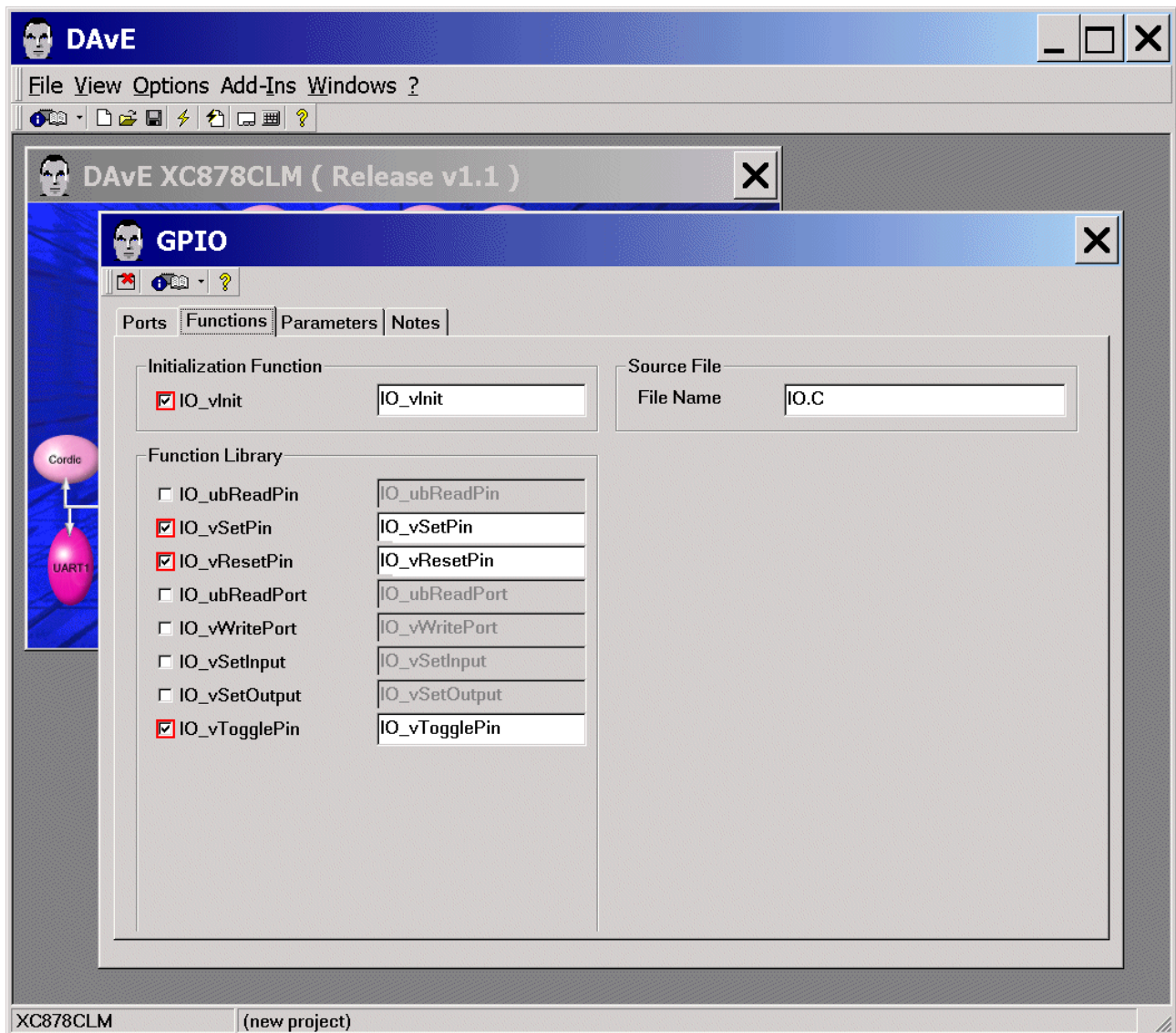
Parameters: (do nothing)




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

Exit this dialog now by clicking  the close button.

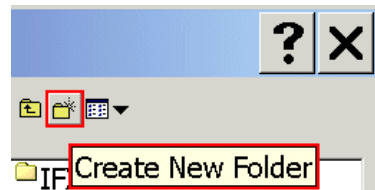
Functions: Initialization Functions: **tick** ✓ IO_vInit
 Functions: Function Library: **tick** ✓ IO_vSetPin
 Functions: Function Library: **tick** ✓ IO_vResetPin
 Functions: Function Library: **tick** ✓ IO_vTogglePin



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

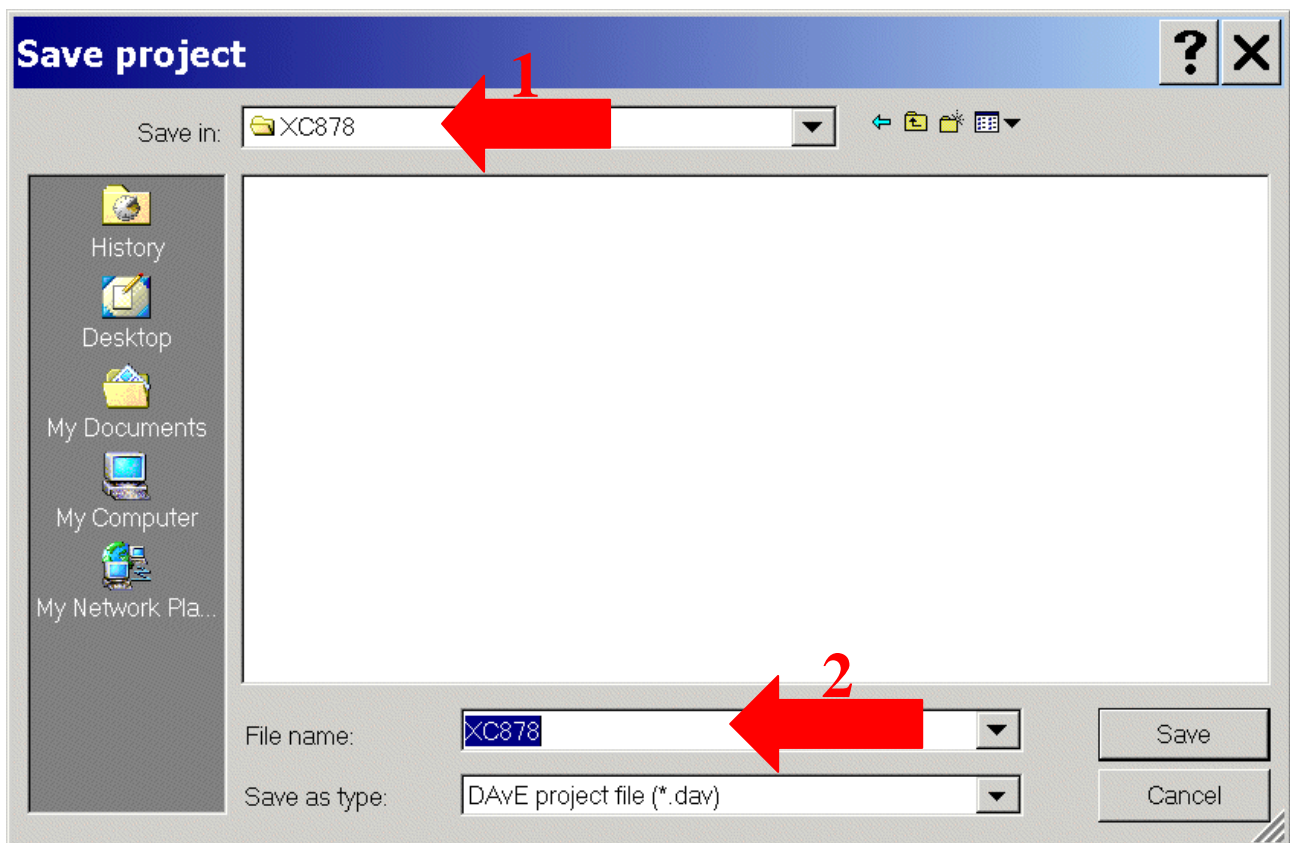
Save the project:

File
Save




Save project: Save in C:\XC878 [create new directory
File name: XC878 (2)

(1)]



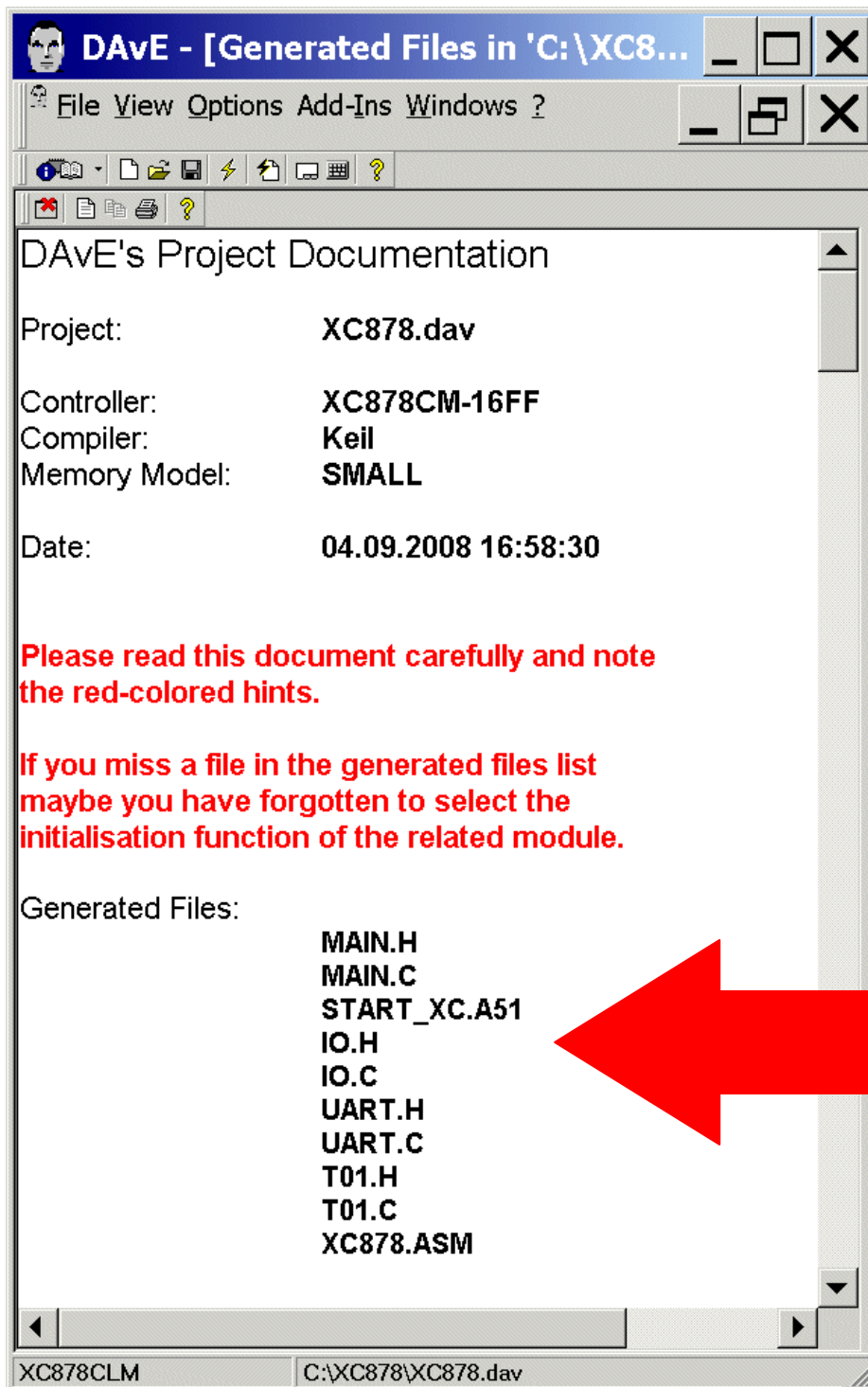
Save

Generate Code:

File Generate Code	or click 
-----------------------	---



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



File - Exit

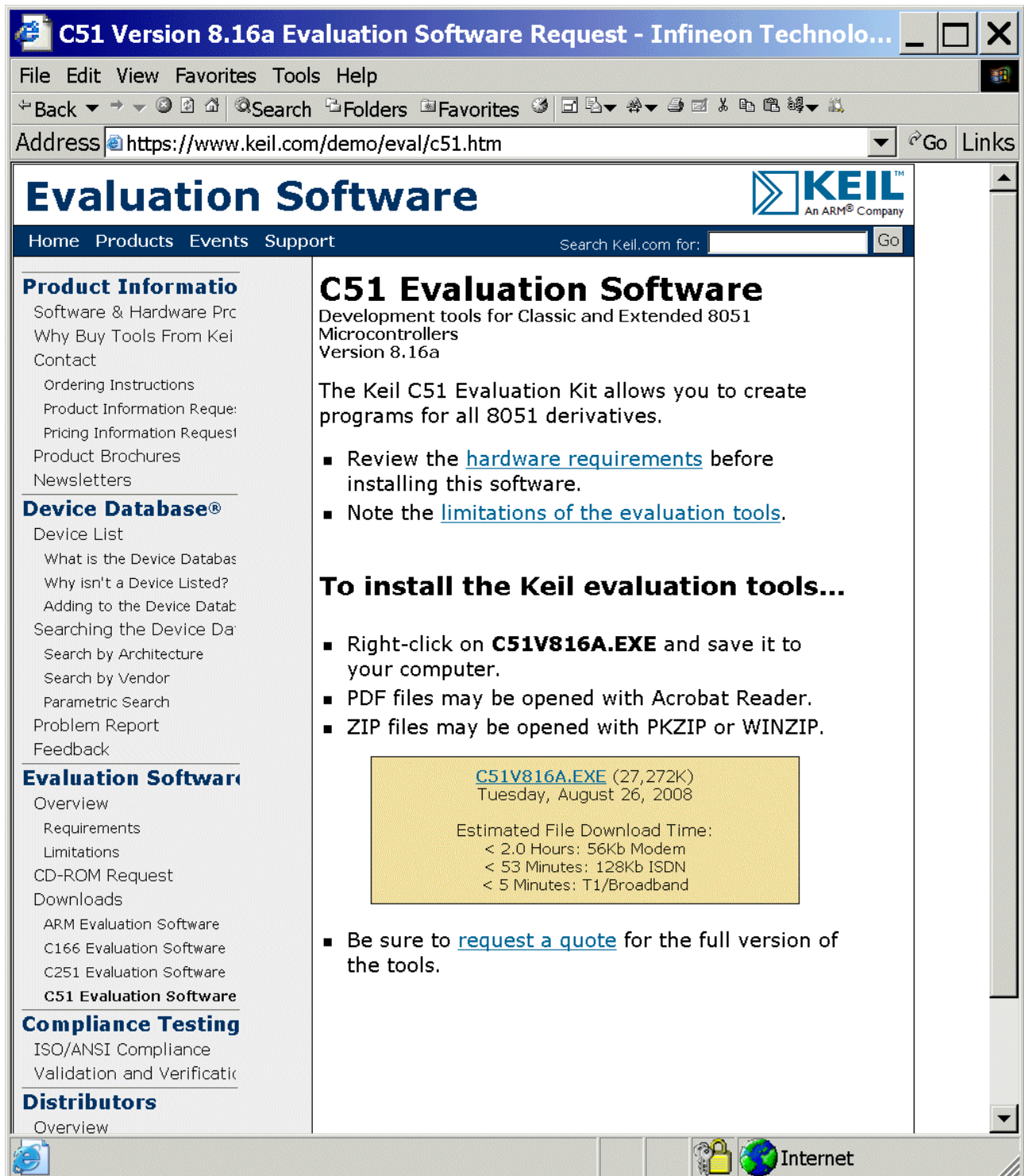
Save changes?

click **Yes**

4.) Using the KEIL - μ Vision 3 Development Tools:

Install the Tool chain:

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



The screenshot shows a web browser window titled "C51 Version 8.16a Evaluation Software Request - Infineon Technolo...". The address bar shows the URL <http://www.keil.com/demo/eval/c51.htm>. The page content includes a navigation menu with links like Home, Products, Events, and Support. The main content area is titled "Evaluation Software" and features a sidebar with links to Product Information, Device Database, Evaluation Software, Compliance Testing, and Distributors. The main content area displays the "C51 Evaluation Software" section, which includes a description of the software, a list of instructions for installation, and a download link for "C51V816A.EXE".

C51 Evaluation Software
Development tools for Classic and Extended 8051 Microcontrollers
Version 8.16a

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...

- Right-click on **C51V816A.EXE** and save it to your computer.
- PDF files may be opened with Acrobat Reader.
- ZIP files may be opened with PKZIP or WINZIP.

C51V816A.EXE (27,272K)
Tuesday, August 26, 2008

Estimated File Download Time:
 < 2.0 Hours: 56Kb Modem
 < 53 Minutes: 128Kb ISDN
 < 5 Minutes: T1/Broadband

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

Execute **C51V816A.EXE** (- or any higher version)



Start Keil μ Vision3 and open the DAVe Project:

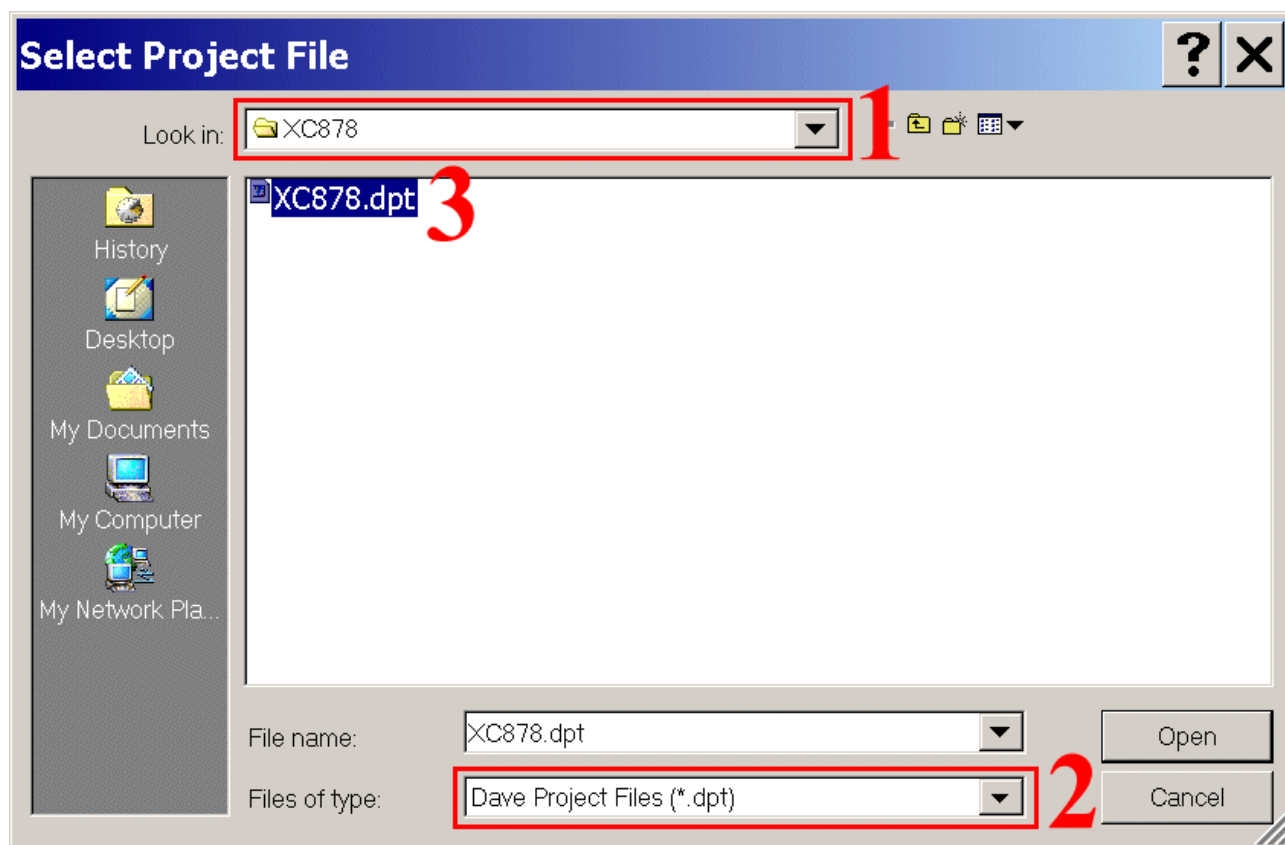
If you see an open project – close it: **Project - Close Project**

Project - Open Project

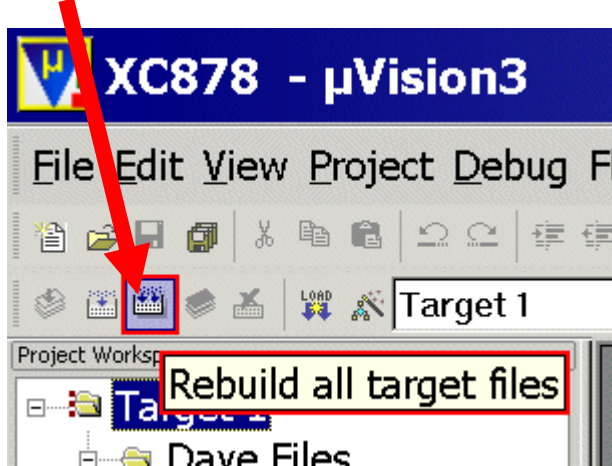
Select Project File: **Look in:** choose C:\XC878 (1)

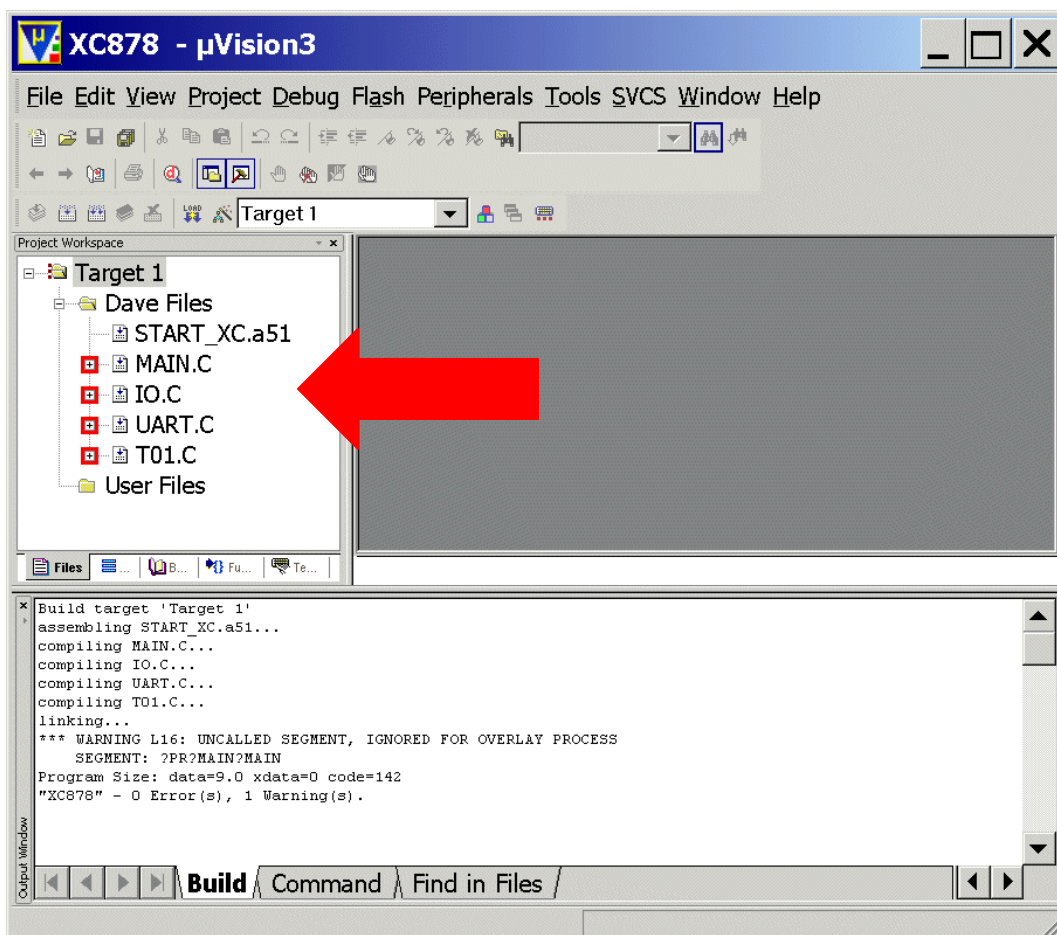
Select Project File: **Files of type:** select Dave Project Files (2)

Choose/click XC878.dpt (3)



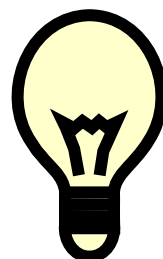
Click Open

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

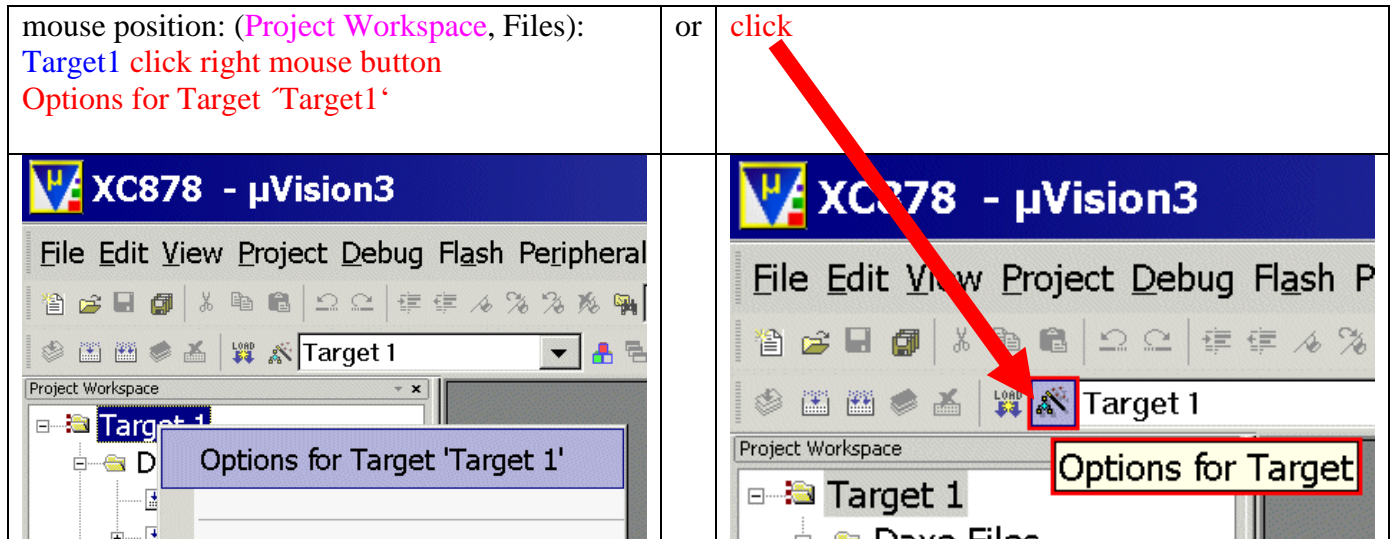


Note:

This step generates a makefile and shows the include files.



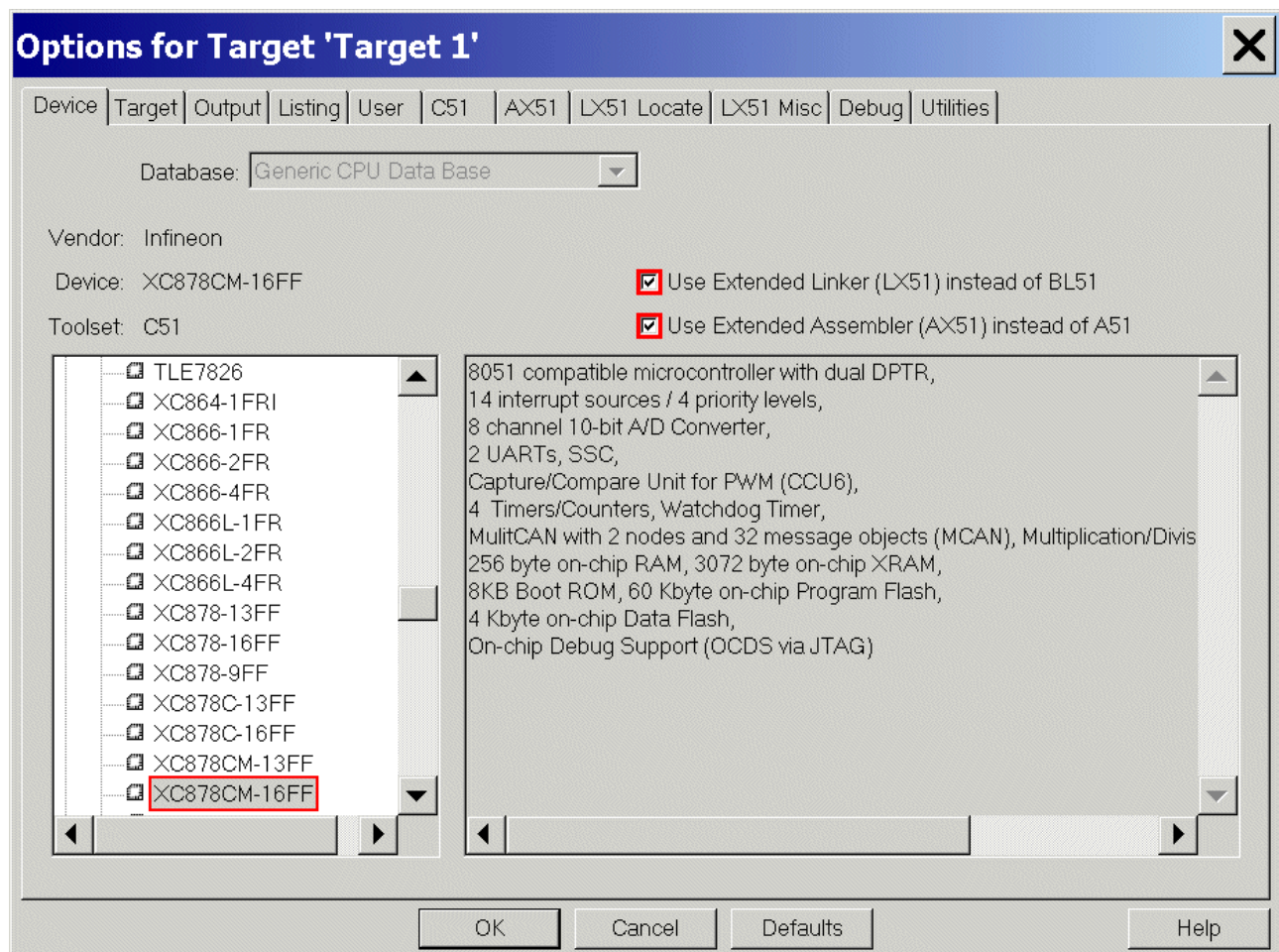
Configure Compiler, Assembler, Linker, Locator, Hex-Converter, Build Control, Simulator, Debugger and Utilities:



Options for Target 'Target 1': Device: **check** XC878-16FF

Options for Target 'Target 1': Device: **tick** ✓ Use Extended Linker (LX51)

Options for Target 'Target 1': Device: **tick** ✓ Use Extended Assembler (AX51)



Target: tick ✓ Use On-chip ROM & Target: tick ✓ Use On-chip XRAM

Options for Target 'Target 1' [X]

Device Target Output Listing User C51 AX51 LX51 Locate LX51 Misc Debug Utilities

Infineon XC878CM-16FF

Xtal (MHz): 144.0

Memory Model: Small: variables in DATA

Code Rom Size: Large: 64K program

Operating system: None

☒ Use On-chip ROM (0x0-0xEFFF,0xF000-0xFFFF)

☒ Use On-chip XRAM (0xF000-0xFBFF)

☐ Use multiple DPTR registers

Off-chip Code memory

	Start:	Size:
Eprom		
Eprom		
Eprom		

Off-chip Xdata memory

	Start:	Size:
Ram		
Ram		
Ram		

☐ Code Banking

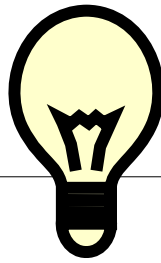
Banks: 2

Bank Area: Start: 0x0000 End: 0xFFFF

☐ 'far' memory type support

☐ Save address extension SFR in interrupts

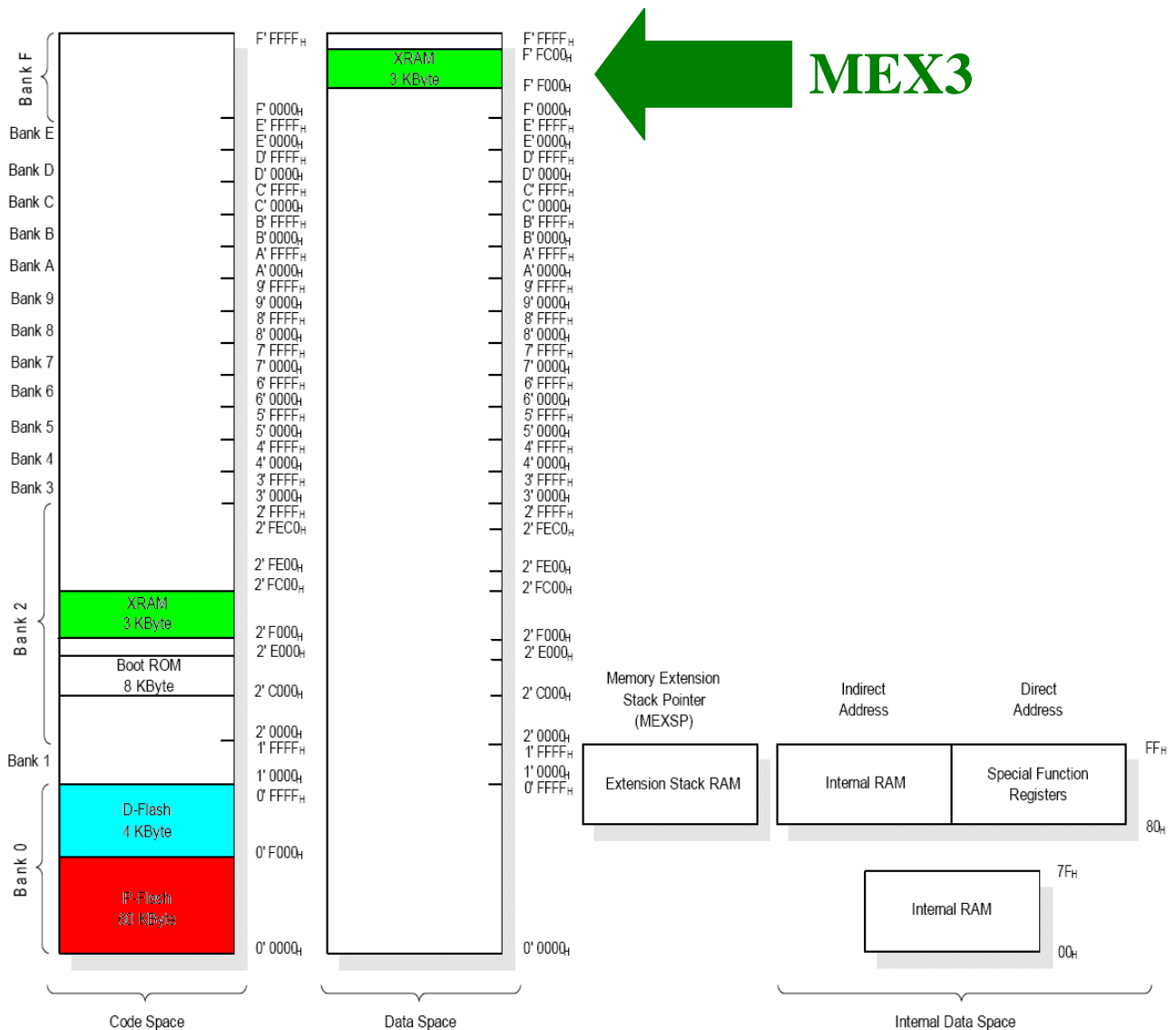
OK Cancel Defaults Help



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

☒ Use On-chip ROM (0x0-0xEFFF,0xF000-0xFFFF)

☒ Use On-chip XRAM (0xF000-0xFBFF)





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

Note (Source: User's Manual):

The standard amount of addressable program or external data memory (or a Bank) in an 8051 system is 64 Kbytes. The XC800 core supports memory expansion of up to 1 Mbyte and this is enabled by the availability of a Memory Management Unit (MMU) and a Memory Extension Stack. The MMU adds a set of Memory Extension registers (MEX1, MEX2, and MEX3) to control access to the extended memory space by different addressing modes.

External Data Memory:

The 3-Kbyte XRAM is mapped to both the external data memory area and the program memory area. It can be accessed using both 'MOVX' and 'MOVC' instructions.

The bank where the memories resides must also be selected with the 4-bit XRAM Bank pointer in MEX3.MX (XRAM bank) or the 4-bit Current Bank pointer in MEX1.CB (current bank), depending on bit MXM.

MEX3

Memory Extension Register 3

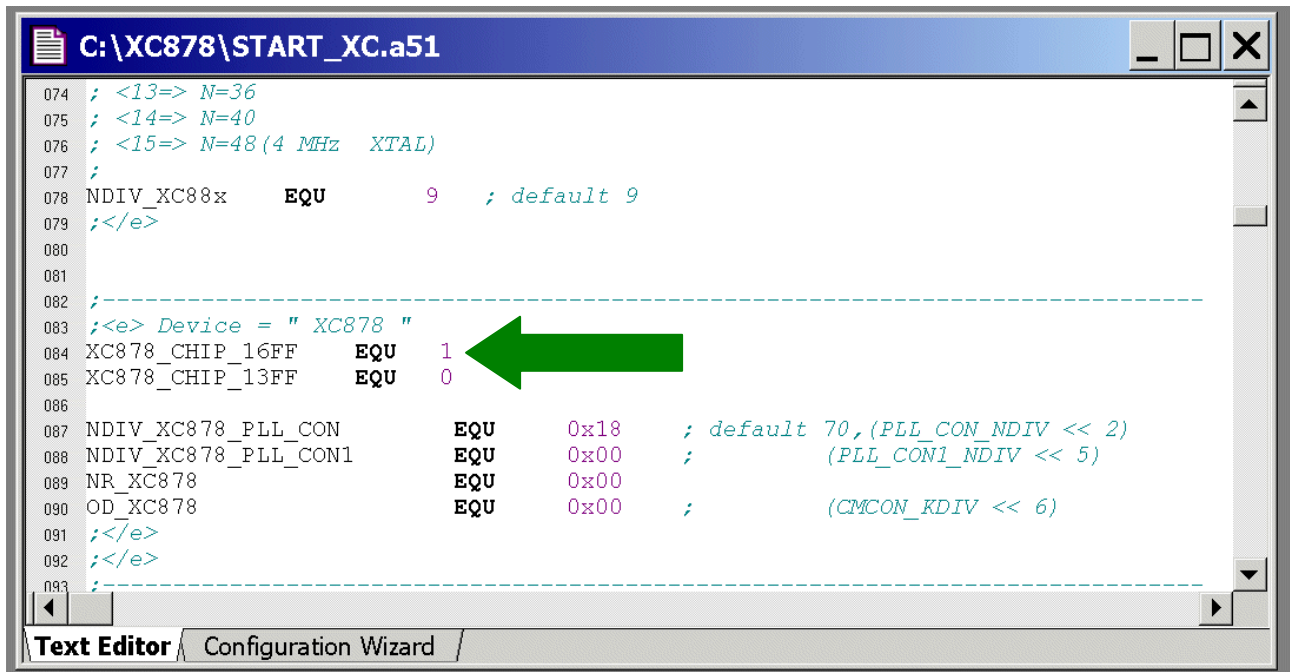
Reset Value: 00_H

7	6	5	4	3	2	1	0
MCB19	0	MXB19	MXM	MXB[18:16]			
rw	rw	rw	rw	rw			

Field	Bits	Type	Description
MXB[19:16]	4, [2:0]	rw	XRAM Bank Number
MXM	3	rw	XRAM Bank Selector 0 MOVX access data in the current bank 1 MOVX access data in the Memory XRAM bank
MCB19	7	rw	Memory Constant Bank Number MSB
0	[6:5]	rw	Reserved Returns 0 if read; should be written with 0.

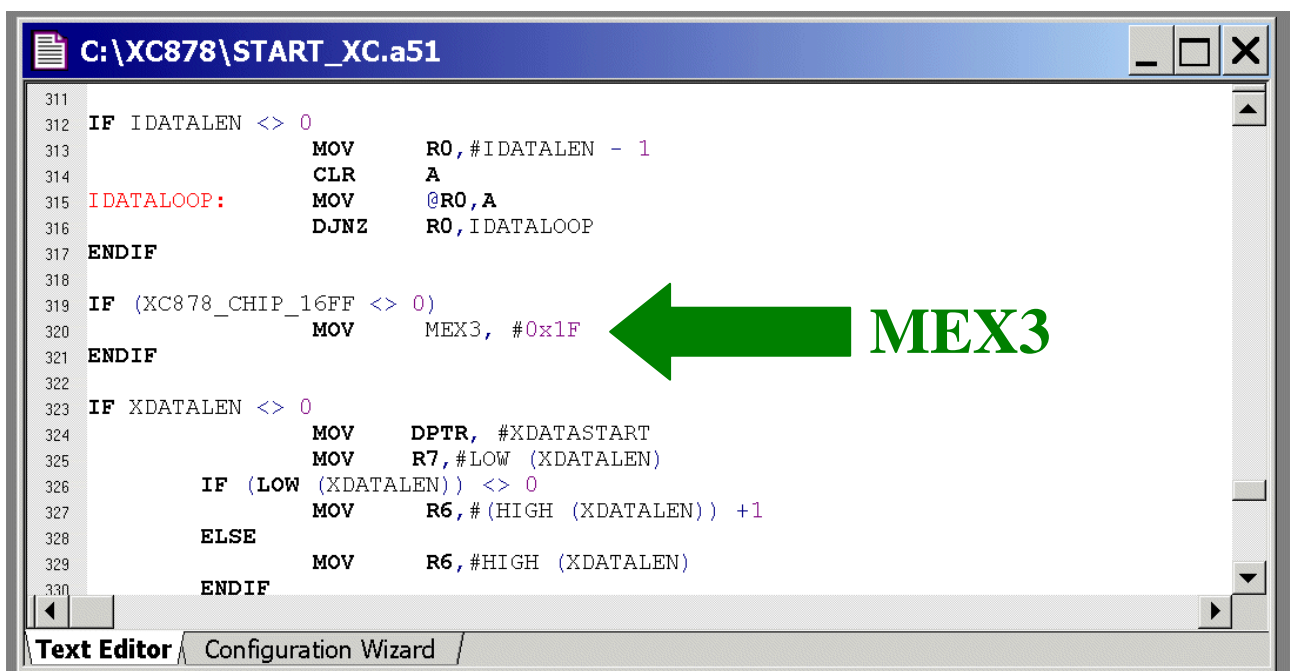


Additional information: Memory Map (Source: START_XC.a51):



```

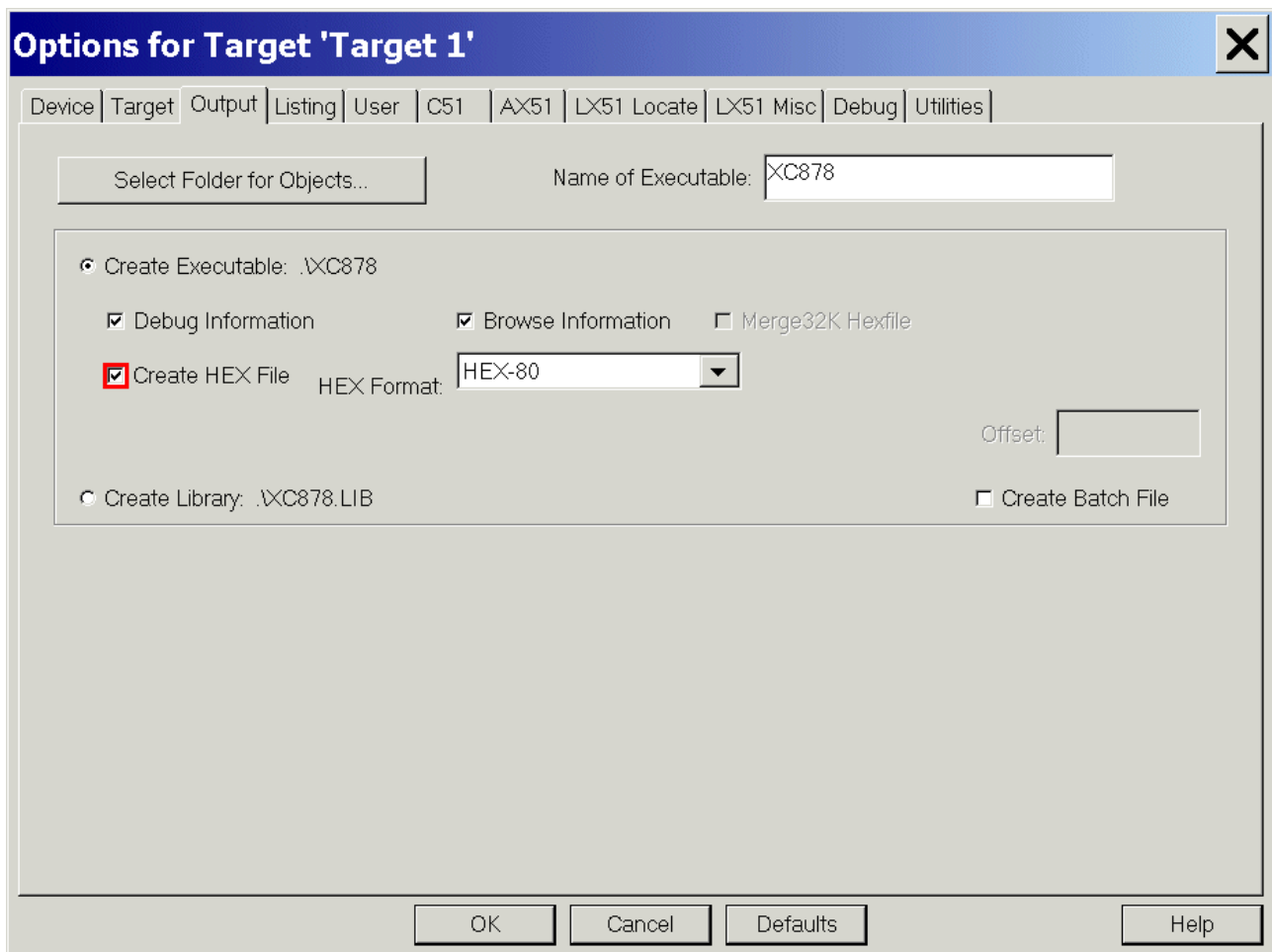
074 ; <13=> N=36
075 ; <14=> N=40
076 ; <15=> N=48 (4 MHz XTAL)
077 ;
078 NDIV_XC88x EQU 9 ; default 9
079 ;</e>
080
081
082 ;-----
083 ;<e> Device = " XC878 "
084 XC878_CHIP_16FF EQU 1
085 XC878_CHIP_13FF EQU 0
086
087 NDIV_XC878_PLL_CON EQU 0x18 ; default 70, (PLL_CON_NDIV << 2)
088 NDIV_XC878_PLL_CON1 EQU 0x00 ; (PLL_CON1_NDIV << 5)
089 NR_XC878 EQU 0x00
090 OD_XC878 EQU 0x00 ; (CMCON_KDIV << 6)
091 ;</e>
092 ;</e>
093 ;-----
  
```



```

311
312 IF IDATALEN <> 0
313     MOV R0, #IDATALEN - 1
314     CLR A
315 IDATALOOP: MOV @R0, A
316     DJNZ R0, IDATALOOP
317 ENDIF
318
319 IF (XC878_CHIP_16FF <> 0)
320     MOV MEX3, #0x1F
321 ENDIF
322
323 IF XDATALEN <> 0
324     MOV DPTR, #XDATASTART
325     MOV R7, #LOW (XDATALEN)
326     IF (LOW (XDATALEN)) <> 0
327         MOV R6, # (HIGH (XDATALEN)) + 1
328     ELSE
329         MOV R6, #HIGH (XDATALEN)
330     ENDIF
  
```

Output: **tick** ✓ Create HEX File



Options for Target 'Target 1'

Device | Target | **Output** | Listing | User | C51 | AX51 | LX51 Locate | LX51 Misc | Debug | Utilities

Select Folder for Objects... Name of Executable: XC878

☒ Create Executable: .\XC878

☒ Debug Information ☒ Browse Information ☐ Merge32K Hexfile

☒ Create HEX File HEX Format: HEX-80

Offset:

☐ Create Library: .\XC878.LIB ☐ Create Batch File

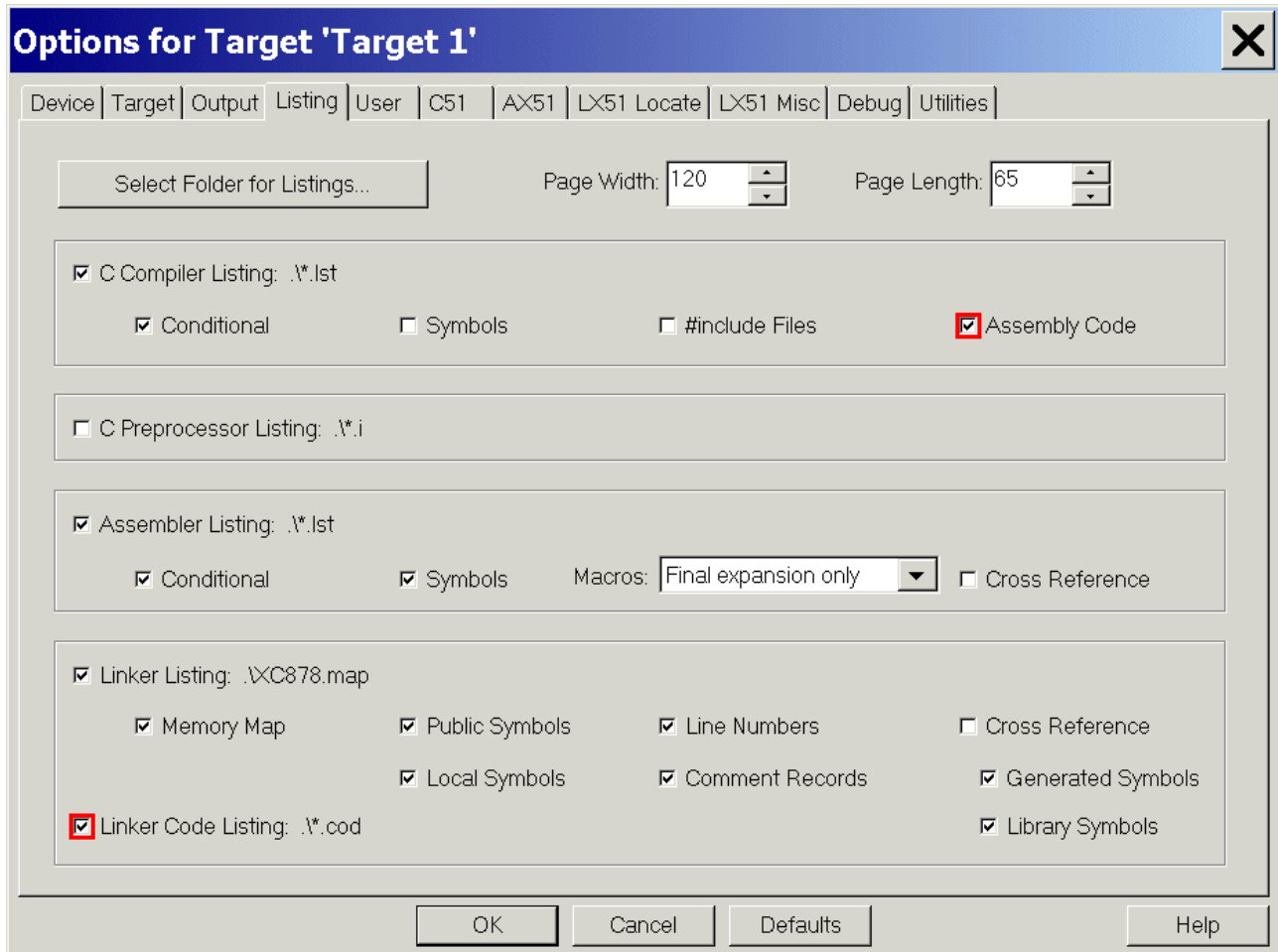
OK Cancel Defaults Help

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: **tick** ✓ Assembly Code
Listing: Linker Listing: **tick** ✓ 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)

Options for Target 'Target 1' [X]

Device | Target | Output | Listing | User | C51 | AX51 | LX51 Locate | LX51 Misc | Debug | Utilities

Run User Programs Before Compilation of a C/C++ File

☐ Run #1: [] ... ☐ DOS16

☐ Run #2: [] ... ☐ DOS16

Run User Programs Before Build/Rebuild

☐ Run #1: [] ... ☐ DOS16

☐ Run #2: [] ... ☐ DOS16

Run User Programs After Build/Rebuild

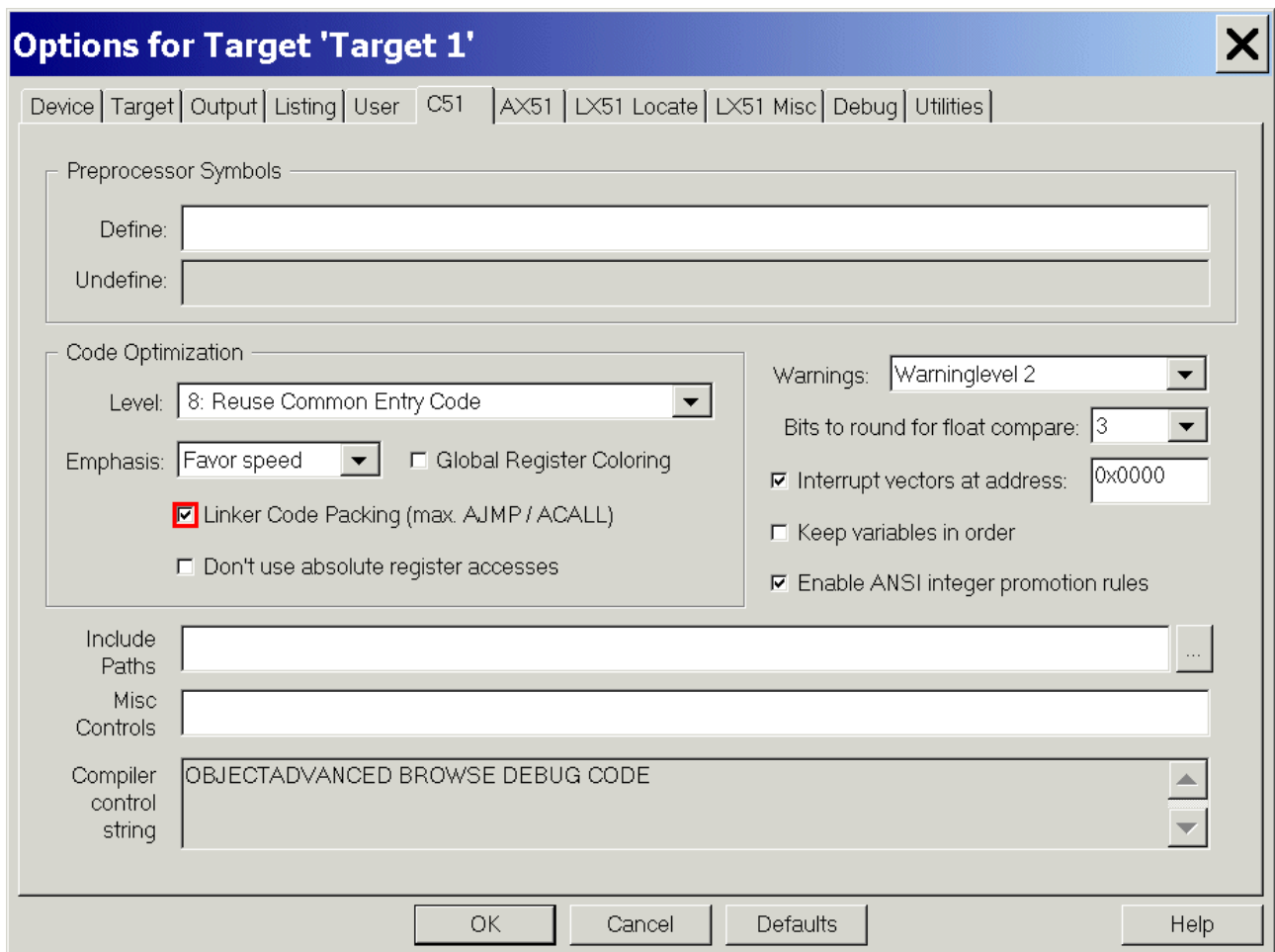
☐ Run #1: [] ... ☐ DOS16

☐ Run #2: [] ... ☐ DOS16

☒ Beep When Complete ☐ Start Debugging

OK Cancel Defaults Help

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



Options for Target 'Target 1'

Device | Target | Output | Listing | User | **C51** | AX51 | LX51 Locate | LX51 Misc | Debug | Utilities

Preprocessor Symbols

Define:

Undefine:

Code Optimization

Level: 8: Reuse Common Entry Code

Emphasis: Favor speed ☐ Global Register Coloring

☒ Linker Code Packing (max. AJMP / ACALL)

☐ Don't use absolute register accesses

Warnings: Warninglevel 2

Bits to round for float compare: 3

☒ Interrupt vectors at address: 0x0000

☐ Keep variables in order

☒ Enable ANSI integer promotion rules

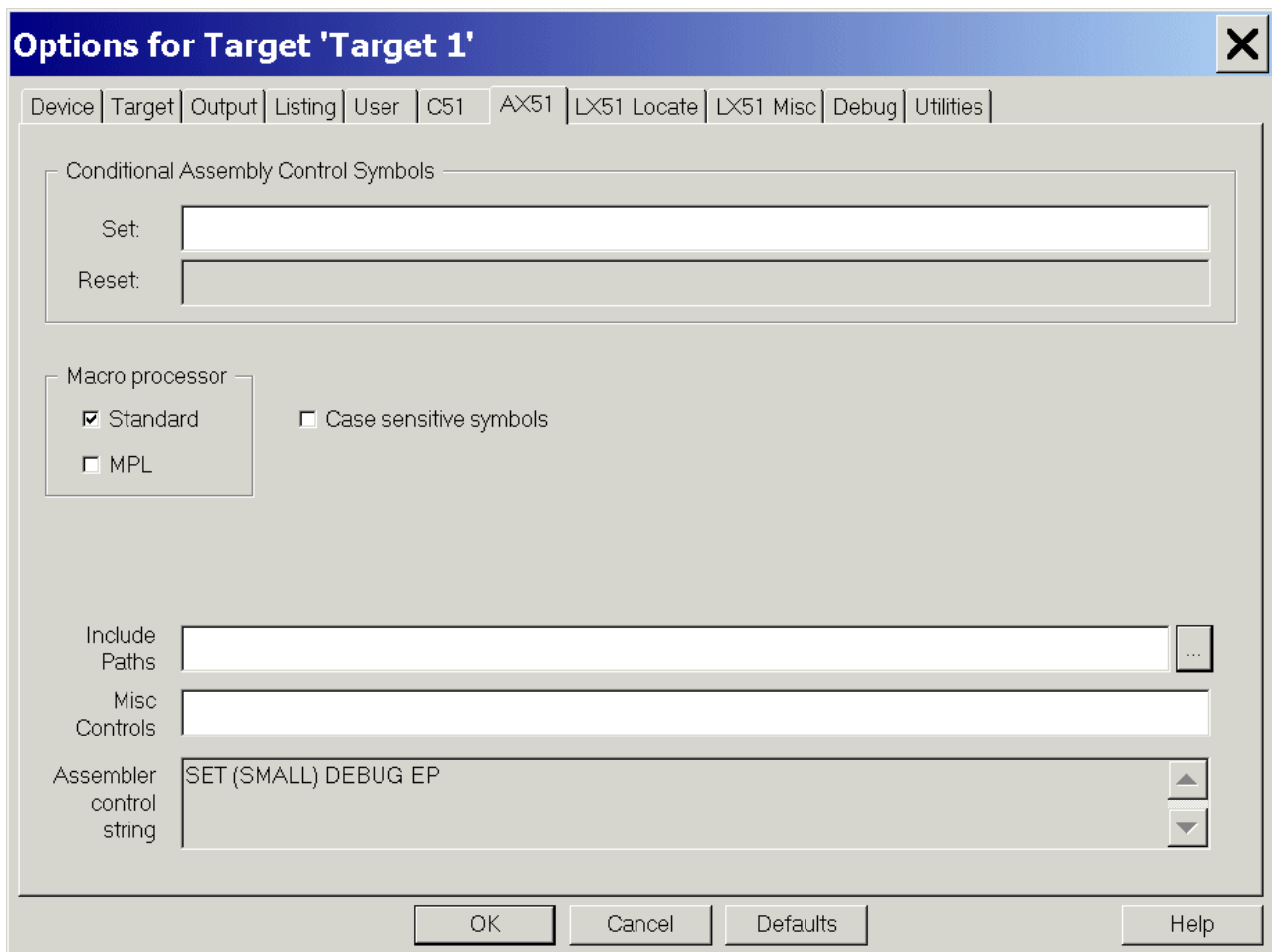
Include Paths: ...

Misc Controls:

Compiler control string: OBJECTADVANCED BROWSE DEBUG CODE

OK Cancel Defaults Help

AX51: (do nothing)

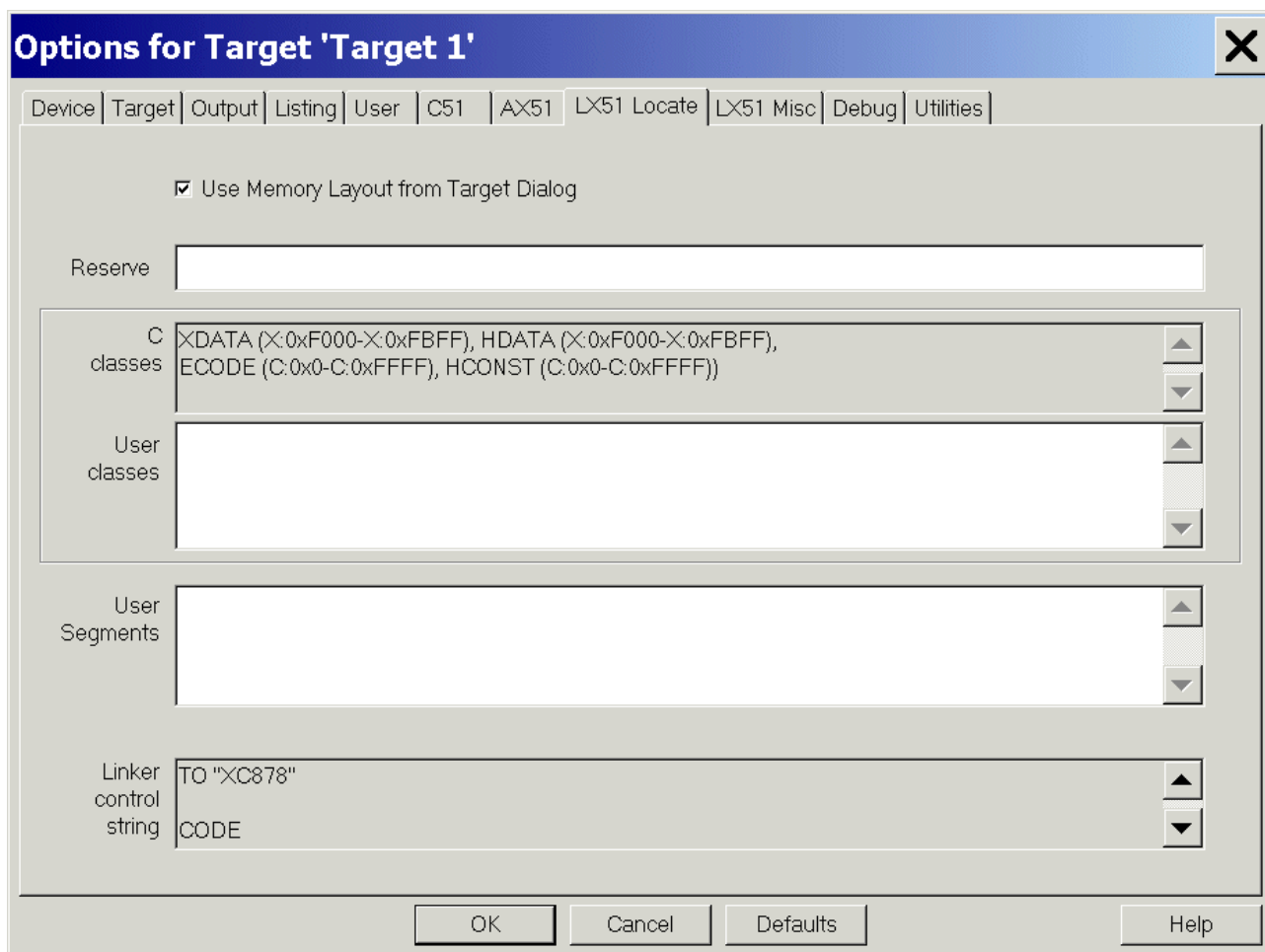


The dialog box is titled "Options for Target 'Target 1'". It features a tabbed interface with the following tabs: Device, Target, Output, Listing, User, C51, AX51 (selected), LX51 Locate, LX51 Misc, Debug, and Utilities. The AX51 tab is active, showing the following options:

- Conditional Assembly Control Symbols**: A section with two text input fields labeled "Set:" and "Reset:", both currently empty.
- Macro processor**: A section containing two checkboxes: ☒ Standard and ☐ Case sensitive symbols. Below these is another checkbox: ☐ MPL.
- Include Paths**: A text input field with a browse button (three dots) to its right.
- Misc Controls**: A text input field.
- Assembler control string**: A text input field containing the text "SET (SMALL) DEBUG EP", with up and down arrow buttons to its right.

At the bottom of the dialog are four buttons: OK, Cancel, Defaults, and Help.

LX51 Locate: (do nothing)

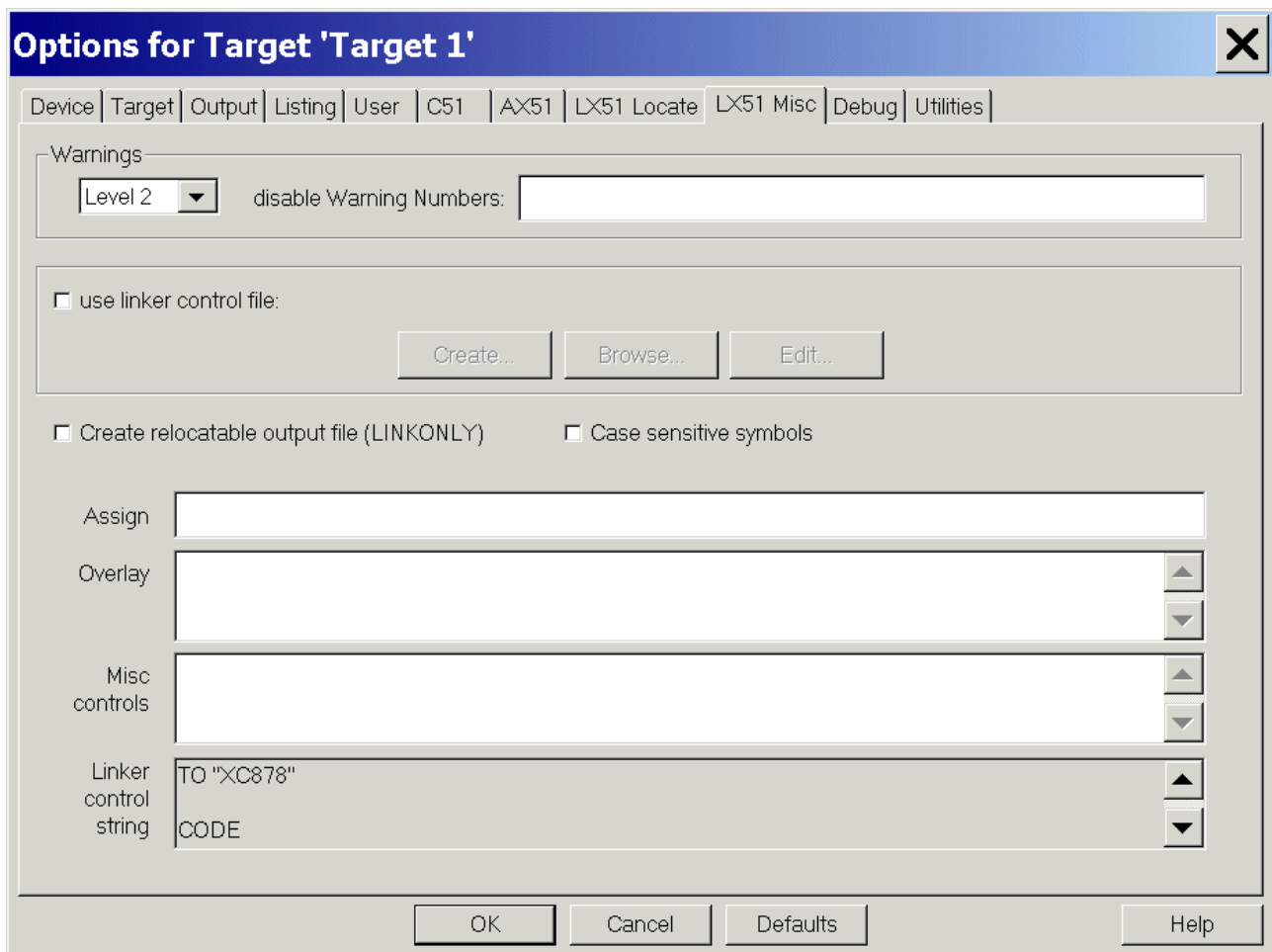


The dialog box 'Options for Target 'Target 1'' has a tabbed interface with the following tabs: Device, Target, Output, Listing, User, C51, AX51, LX51 Locate (selected), LX51 Misc, Debug, and Utilities. The 'LX51 Locate' tab contains the following options:

- ☒ Use Memory Layout from Target Dialog
- Reserve: [Empty text box]
- C classes: XDATA (X:0xF000-X:0xFBFF), HDATA (X:0xF000-X:0xFBFF), ECODE (C:0x0-C:0xFFFF), HCONST (C:0x0-C:0xFFFF)
- User classes: [Empty text box]
- User Segments: [Empty text box]
- Linker control string: TO "XC878"
CODE

Buttons at the bottom: OK, Cancel, Defaults, Help.

LX51 Misc: (do nothing)

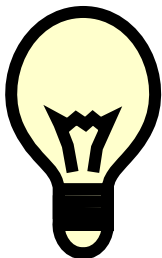
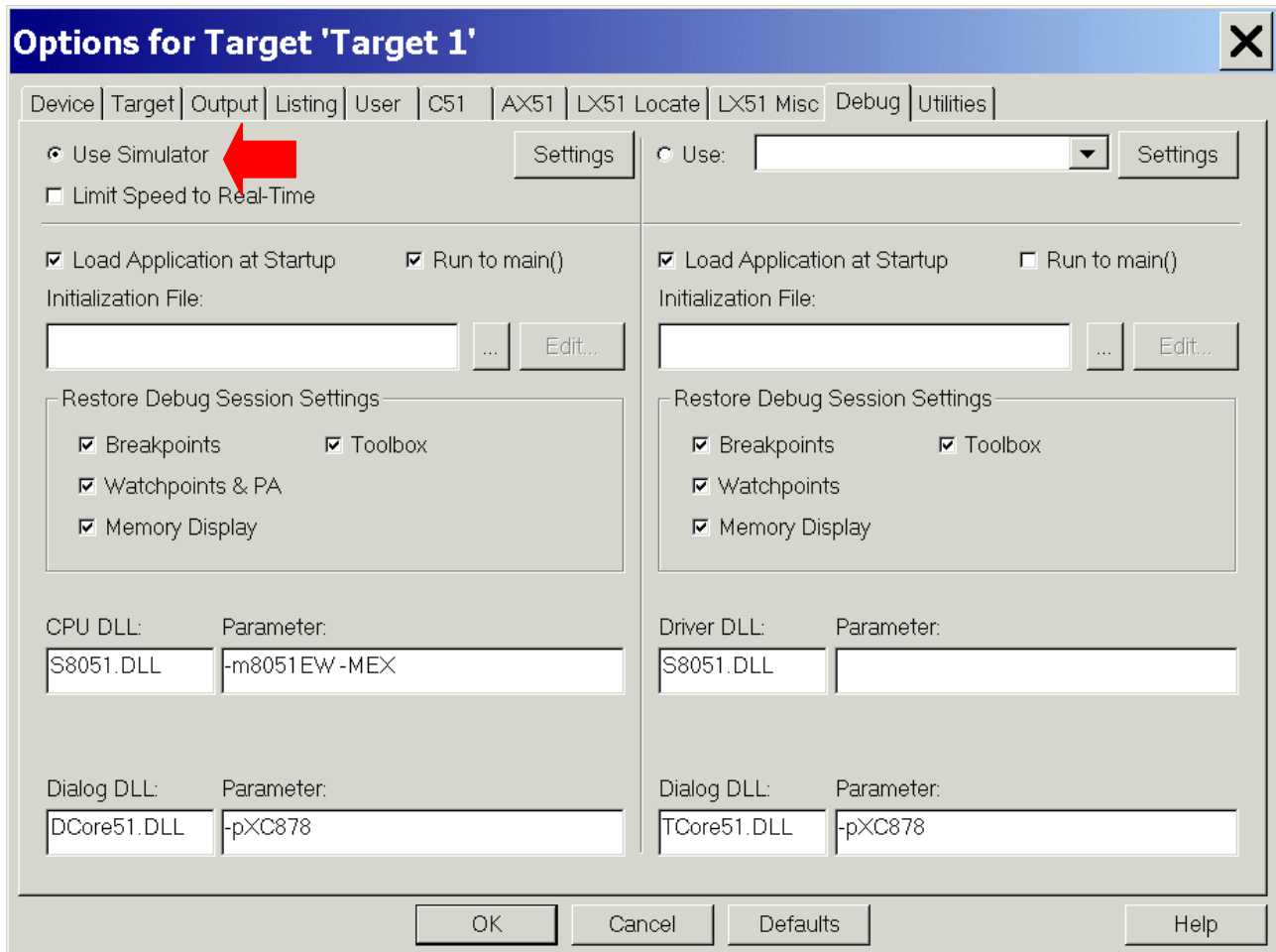


The dialog box 'Options for Target 'Target 1'' features a tabbed interface with the following tabs: Device, Target, Output, Listing, User, C51, AX51, LX51 Locate, LX51 Misc (selected), Debug, and Utilities. The 'LX51 Misc' tab contains the following settings:

- Warnings:** A dropdown menu is set to 'Level 2', and a text field for 'disable Warning Numbers:' is empty.
- use linker control file:** An unchecked checkbox with three buttons: 'Create...', 'Browse...', and 'Edit...'.
- Create relocatable output file (LINKONLY):** An unchecked checkbox.
- Case sensitive symbols:** An unchecked checkbox.
- Assign:** An empty text field.
- Overlay:** An empty text field with up and down arrow buttons on the right.
- Misc controls:** An empty text field with up and down arrow buttons on the right.
- Linker control string:** A text field containing 'TO "XC878"' with an up arrow button on the right, and another text field containing 'CODE' with a down arrow button on the right.

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

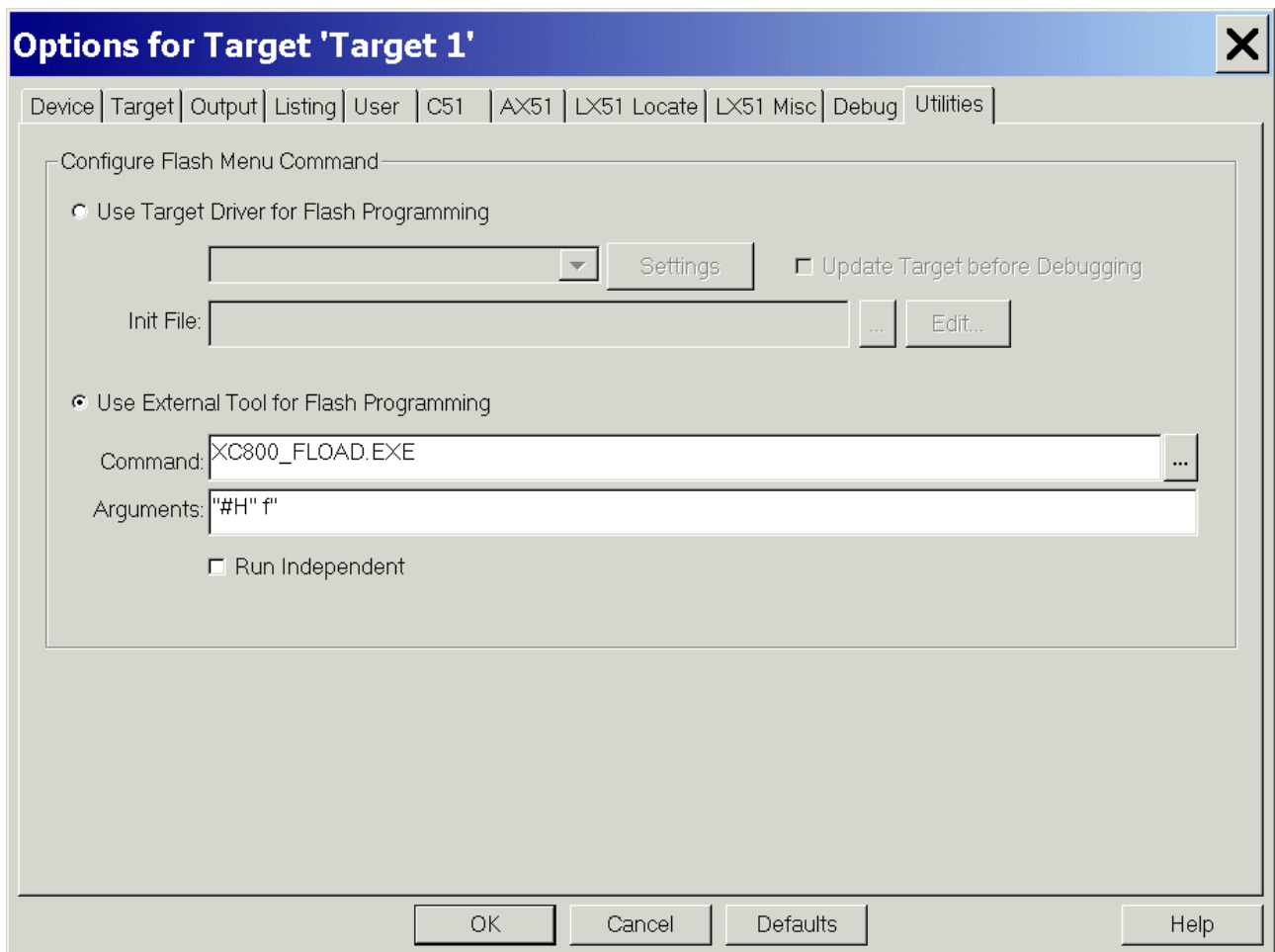
Debug: (do nothing)



Note:

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

Utilities: (do nothing)



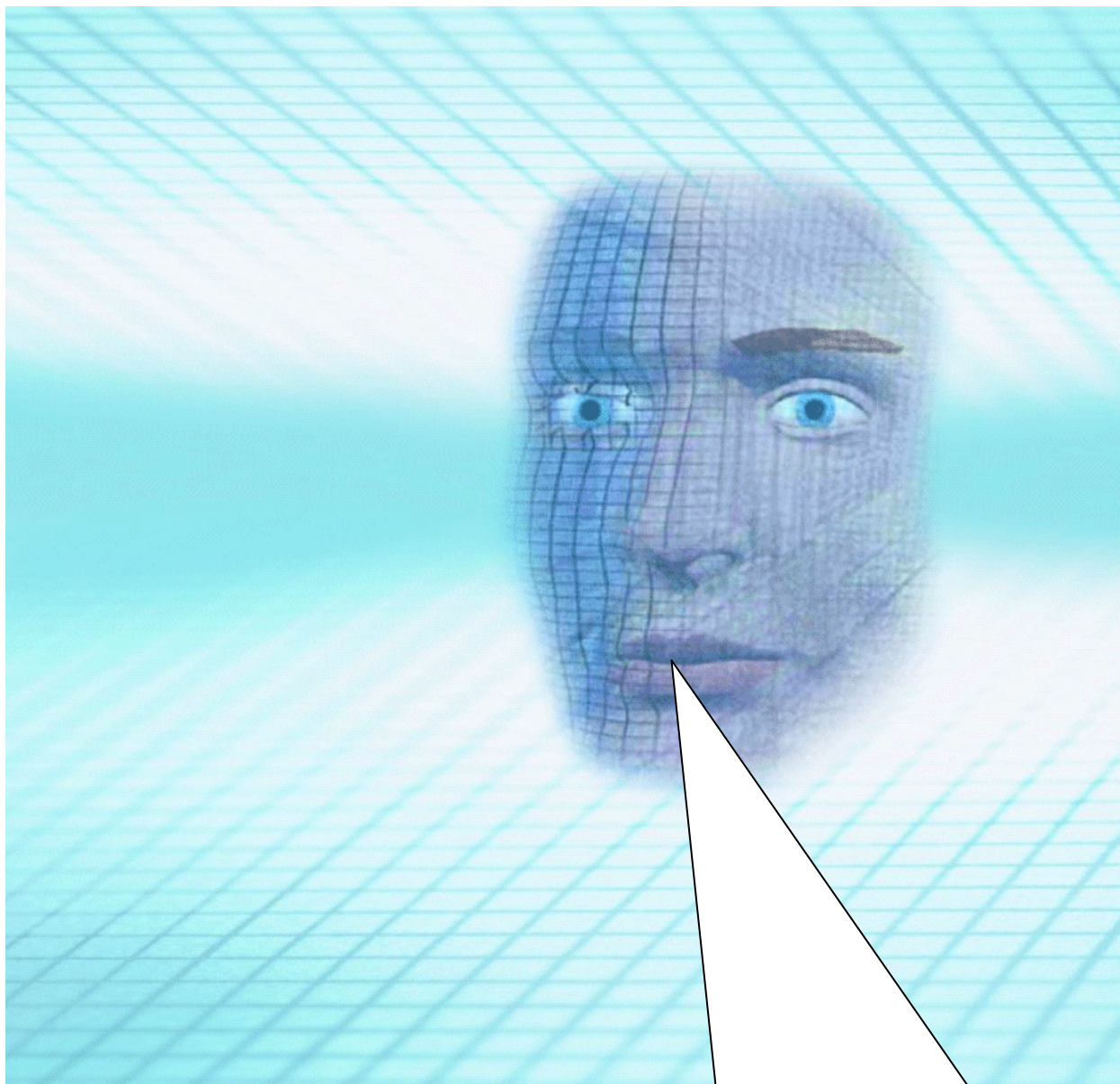
Click OK



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: XC878CLM-16FF Easy Kit *** hello world ***\n"
"\n"
"1 ... LEDs P3 ON\n"
"2 ... LEDs P3 OFF\n"
"3 ... LEDs P3 blinking\n"
" \n";

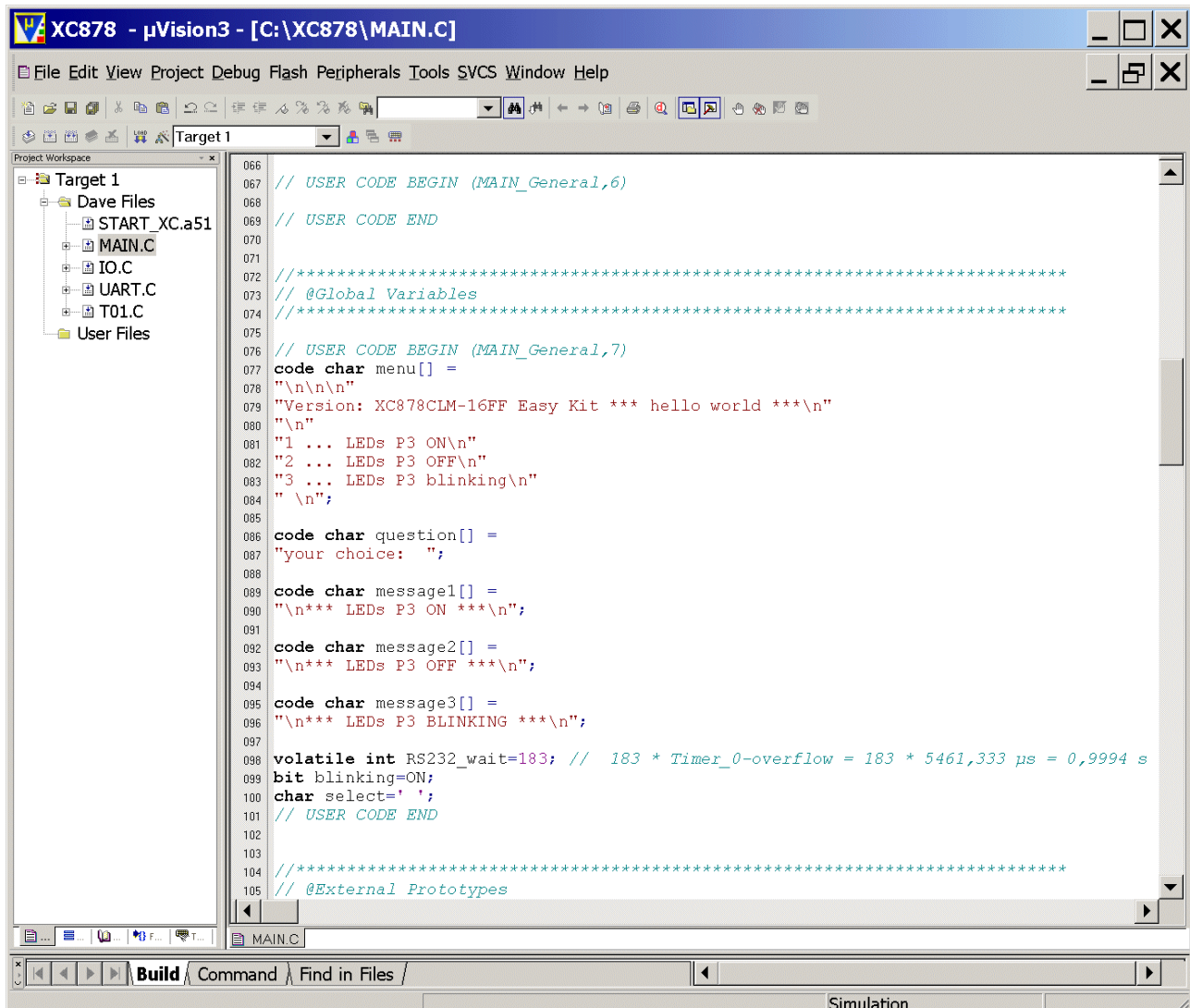
code char question[] =
"your choice: ";

code char message1[] =
"\n*** LEDs P3 ON ***\n";

code char message2[] =
"\n*** LEDs P3 OFF ***\n";

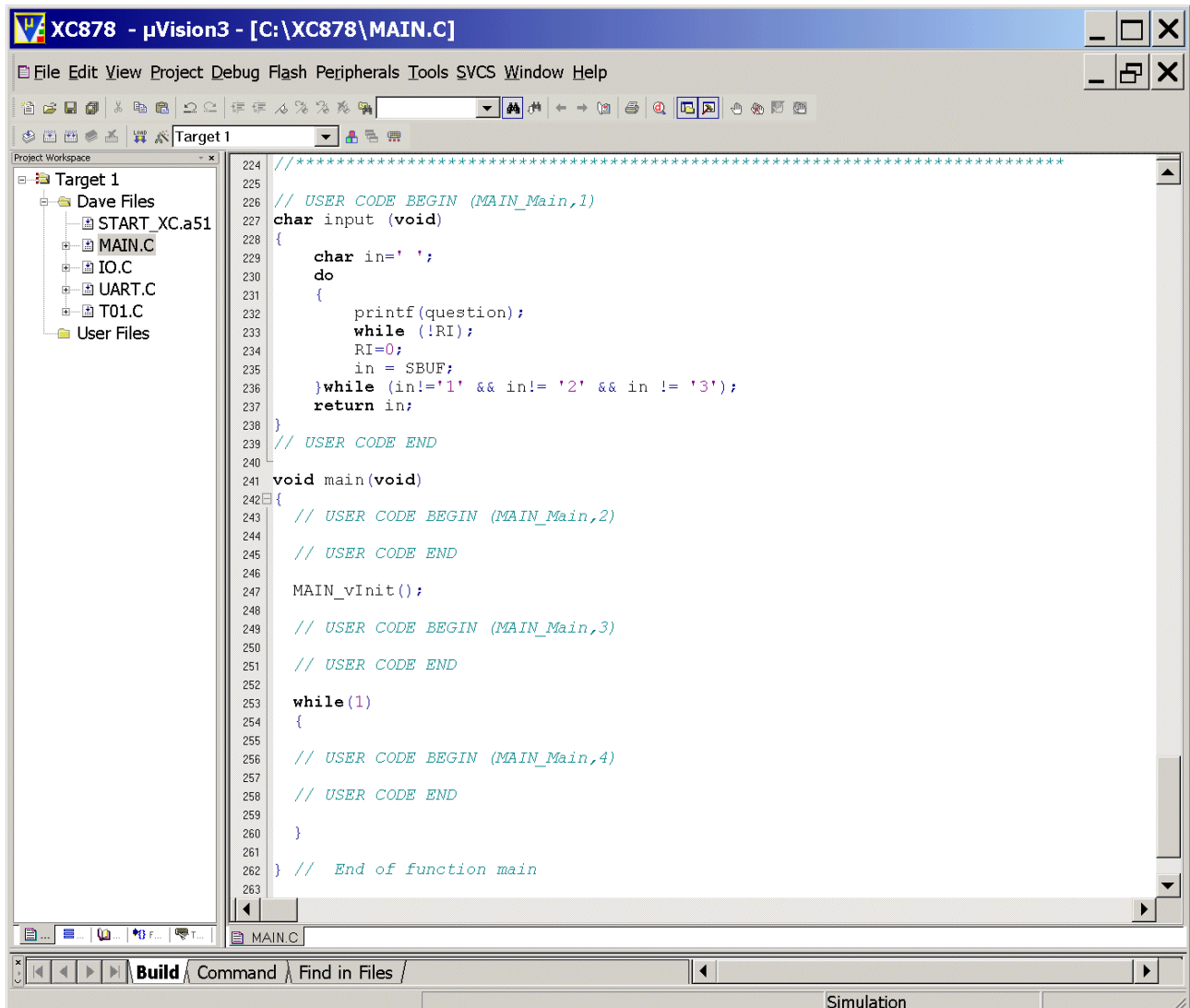
code char message3[] =
"\n*** LEDs P3 BLINKING ***\n";

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

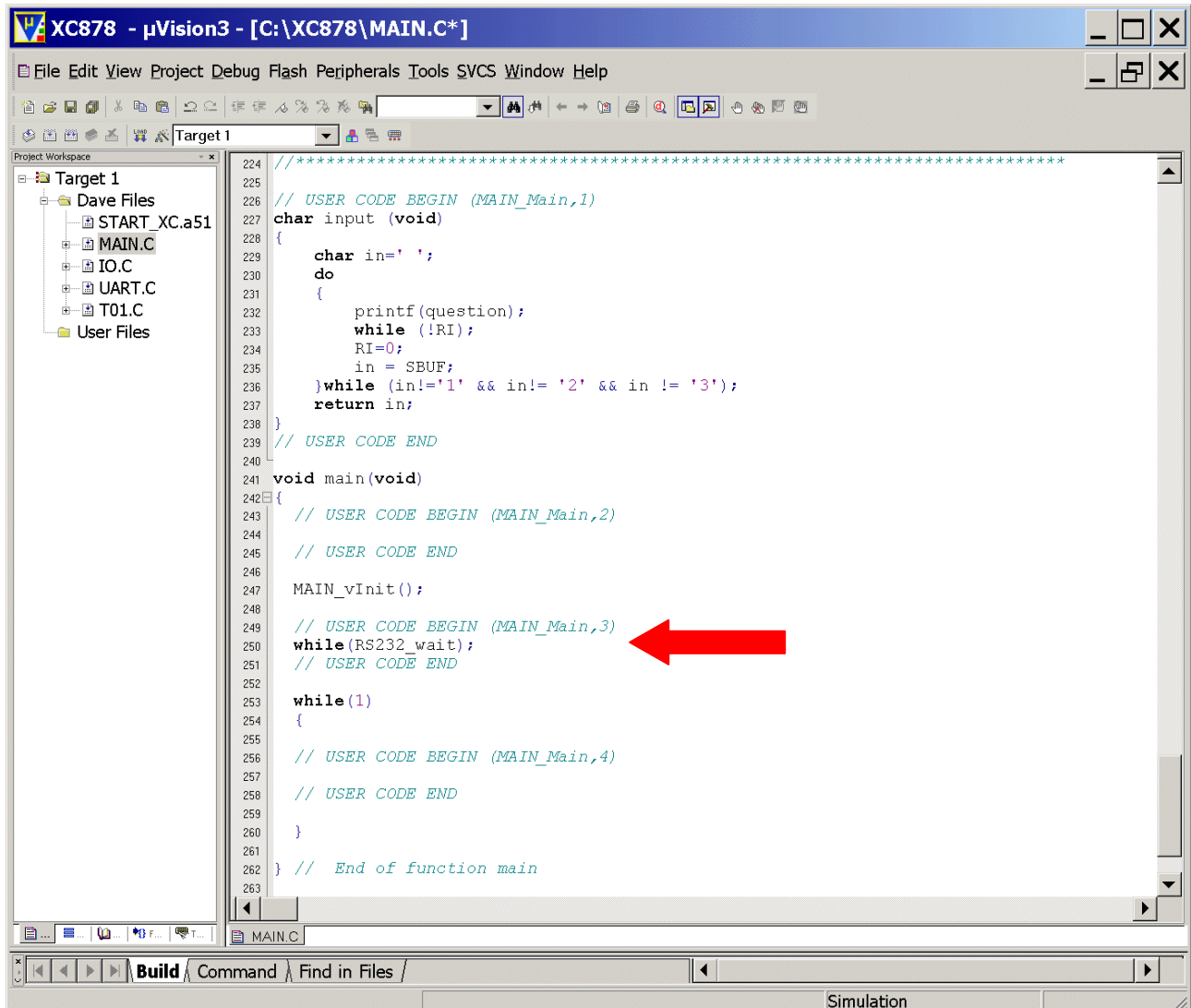
Double click **MAIN.C** and insert the function **input()**:

```
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:

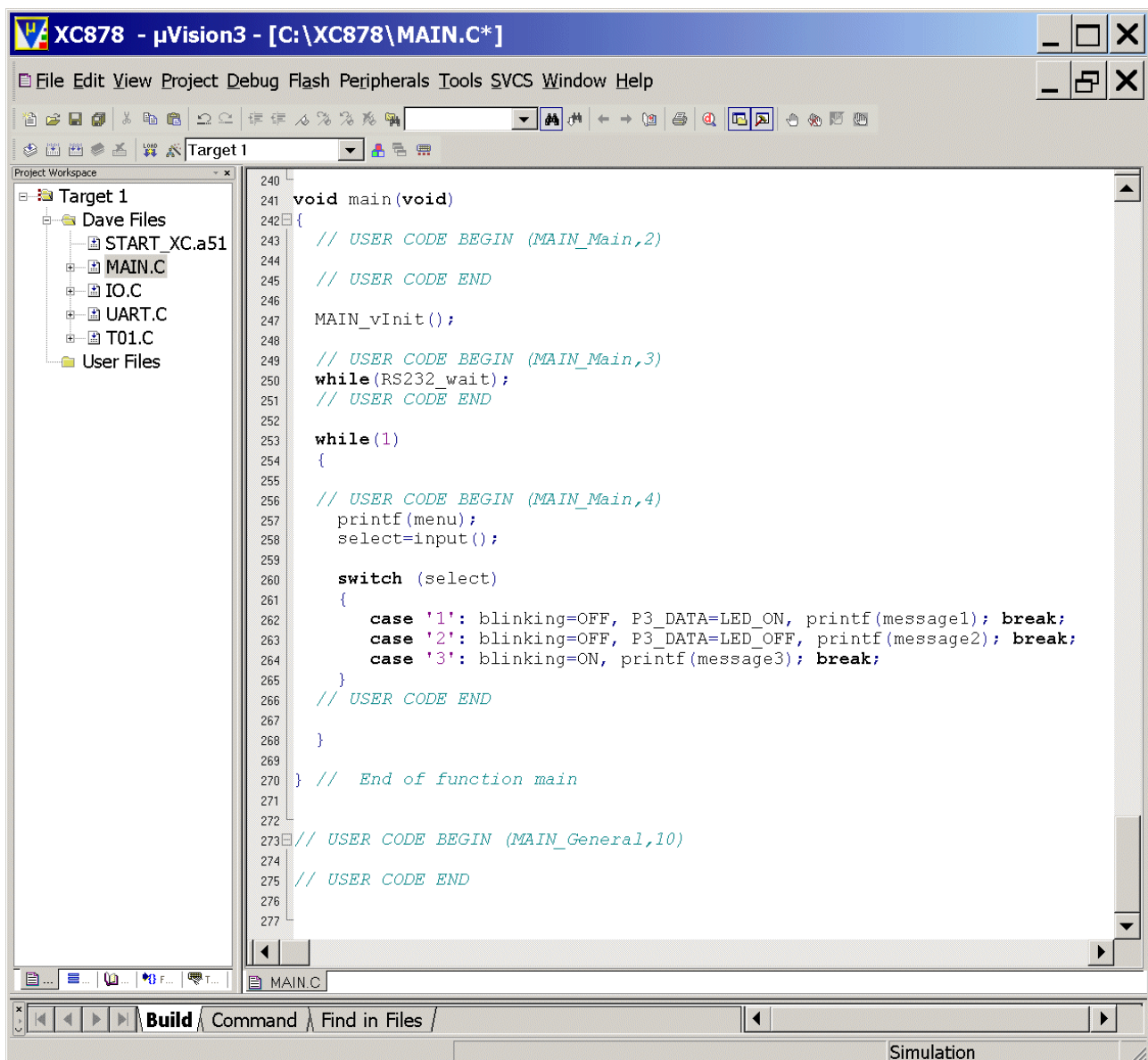
`while(RS232_wait);`



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

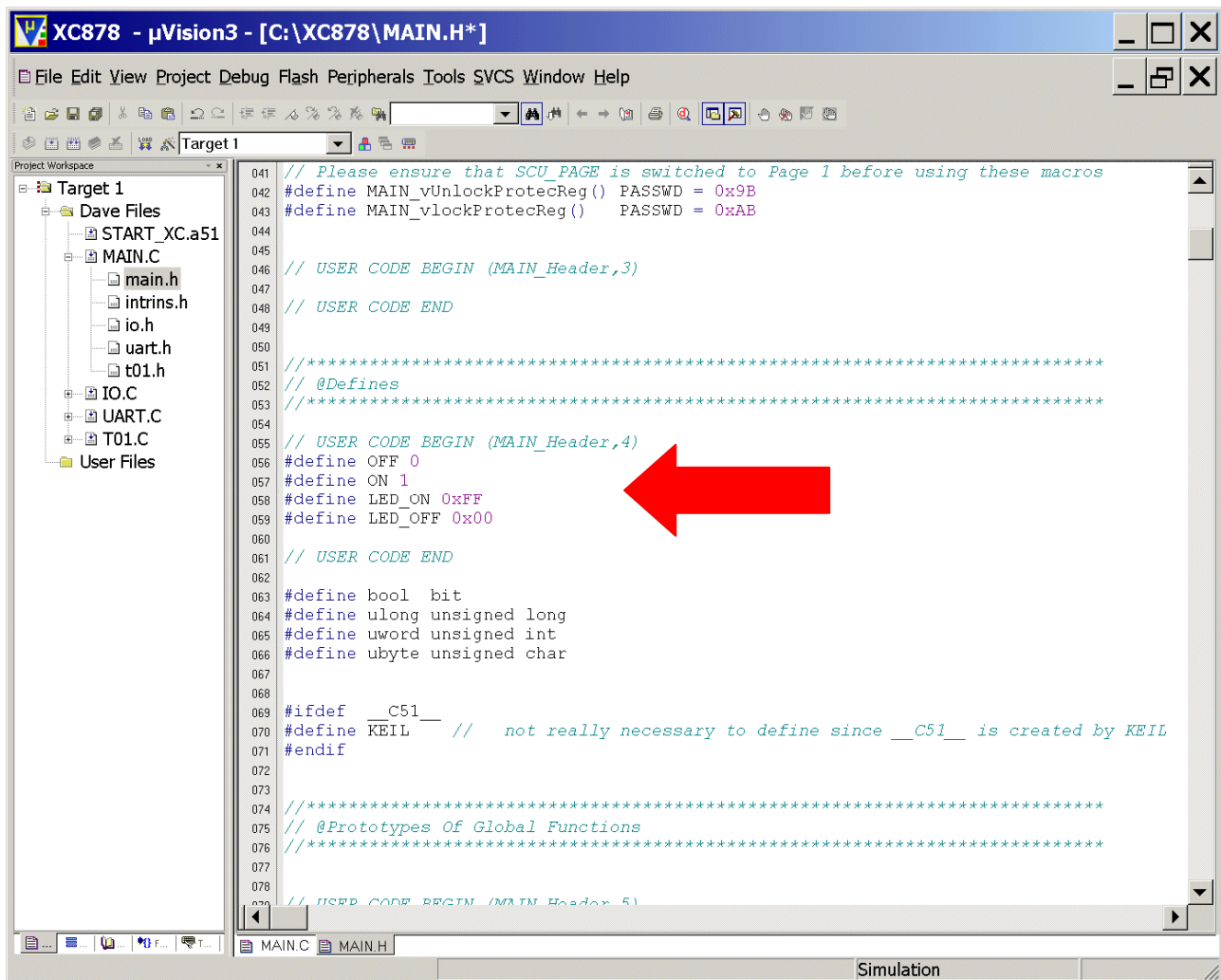
```
printf(menu);
select=input();

switch (select)
{
    case '1': blinking=OFF, 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;
}
```



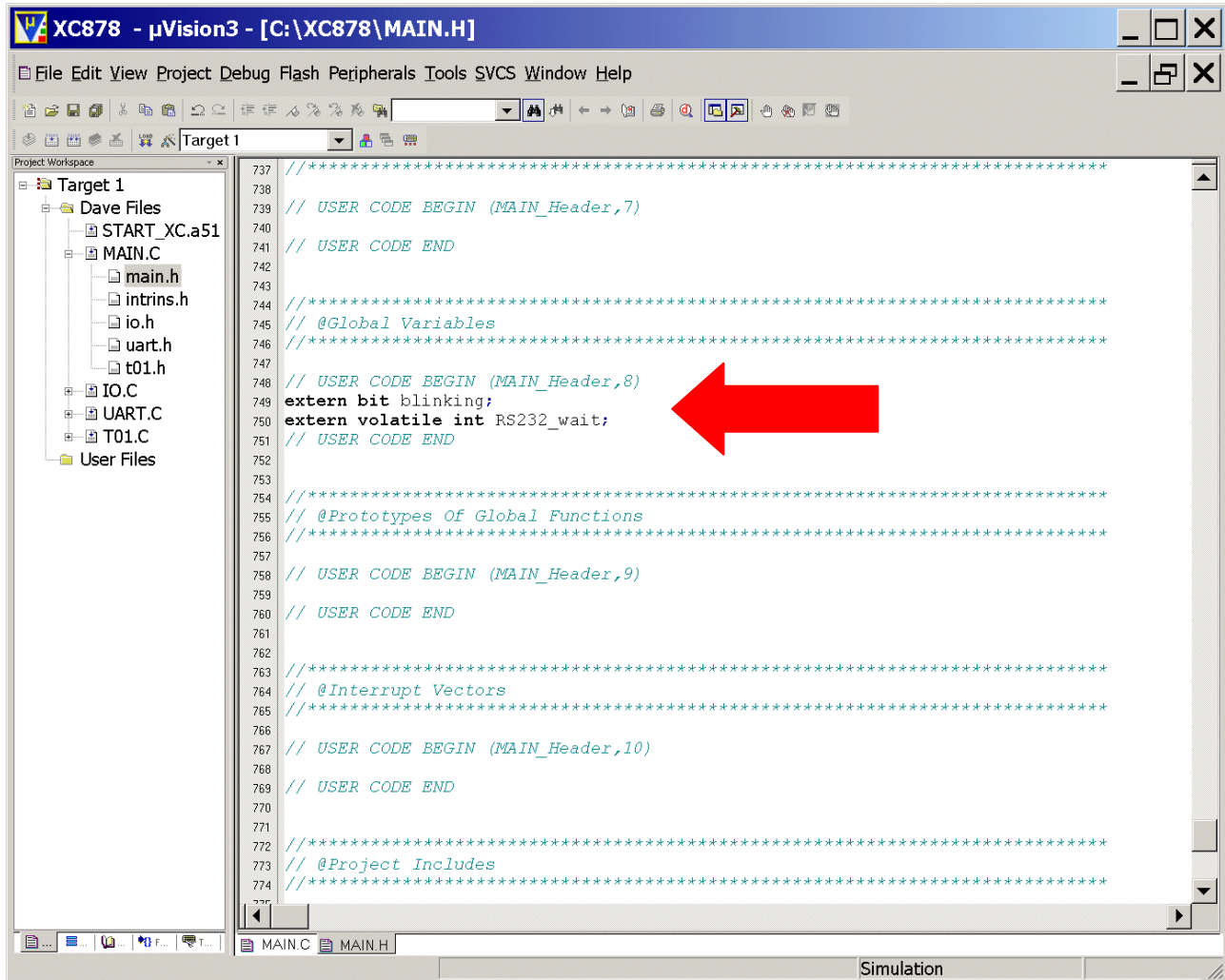
Double click **Main.h** and insert the following Defines:

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



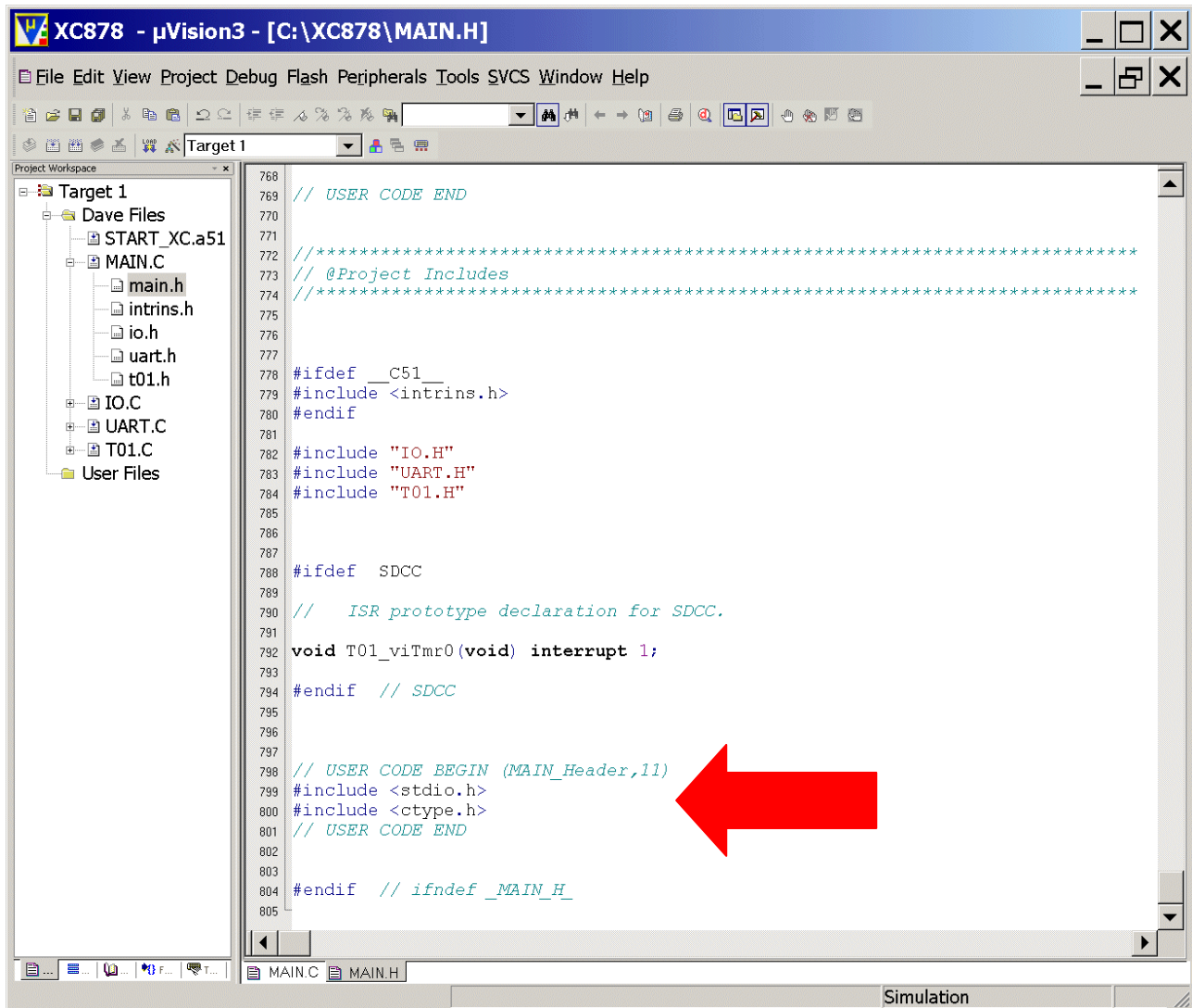
Double click **Main.h** and insert extern-declarations "Global Variables":

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



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

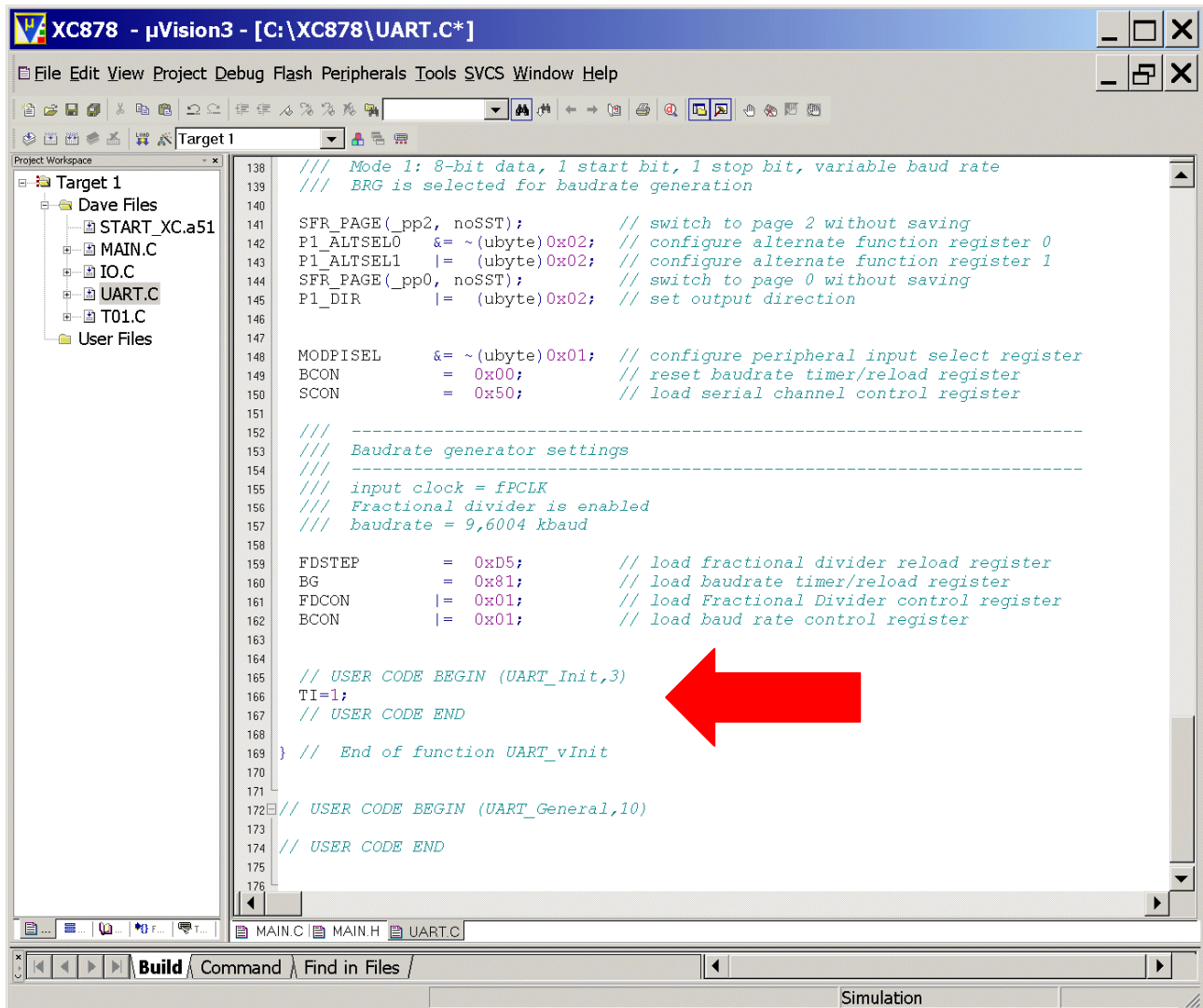
```
#include <stdio.h>
#include <ctype.h>
```



Double click **UART.C**

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

TI=1;



The screenshot shows the XC878 - µVision3 IDE with the UART.C file open. The code is as follows:

```

138  /// Mode 1: 8-bit data, 1 start bit, 1 stop bit, variable baud rate
139  /// BRG is selected for baudrate generation
140
141  SFR_PAGE(_pp2, noSST); // switch to page 2 without saving
142  P1_ALTSEL0  &= ~(ubyte)0x02; // configure alternate function register 0
143  P1_ALTSEL1  |= (ubyte)0x02; // configure alternate function register 1
144  SFR_PAGE(_pp0, noSST); // switch to page 0 without saving
145  P1_DIR      |= (ubyte)0x02; // set output direction
146
147
148  MODPISEL    &= ~(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
175
176

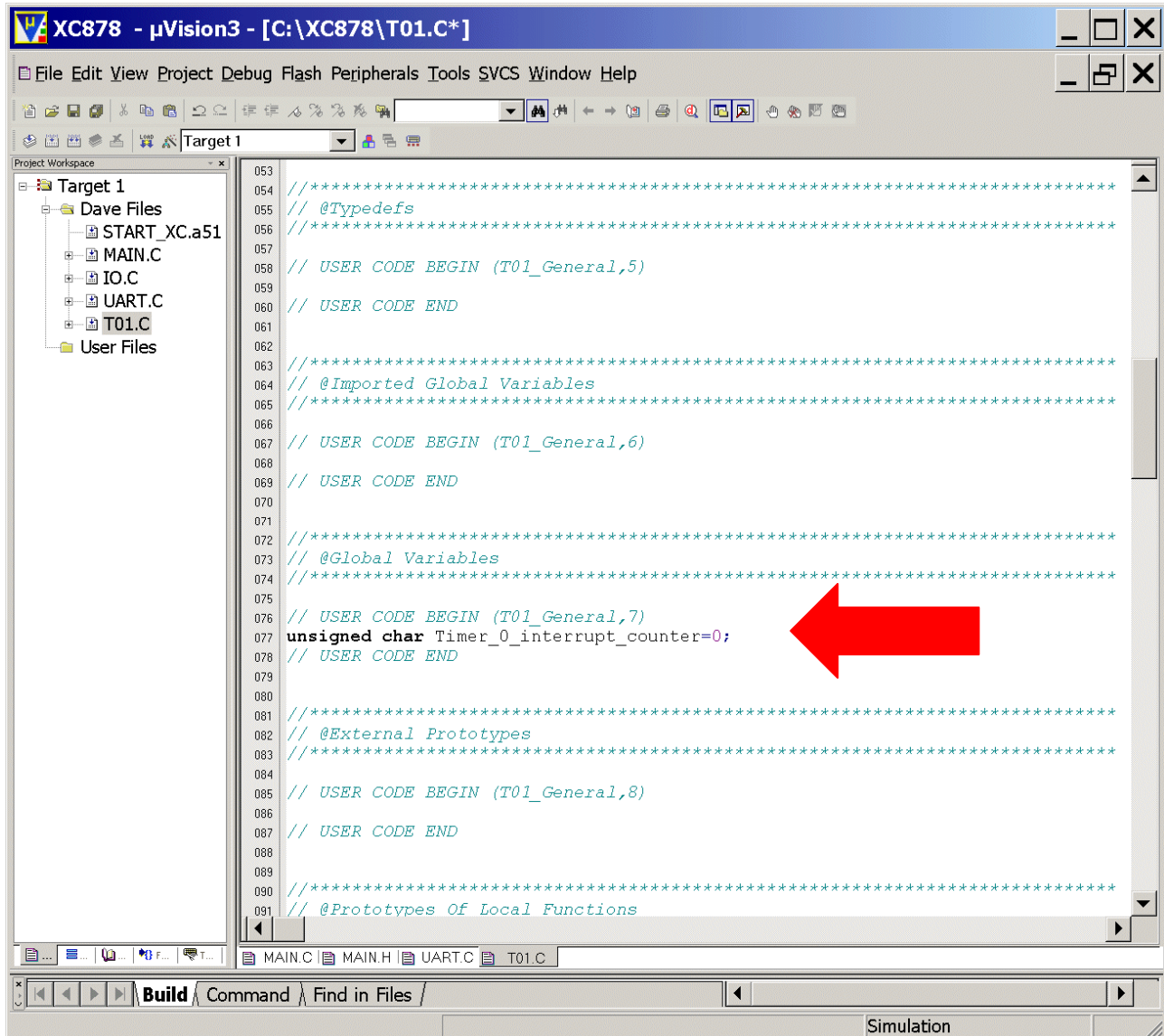
```

A red arrow points to the line `TI=1;` in the `UART_vInit` function.

Double click **T01.C**

Insert the following global variable:

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



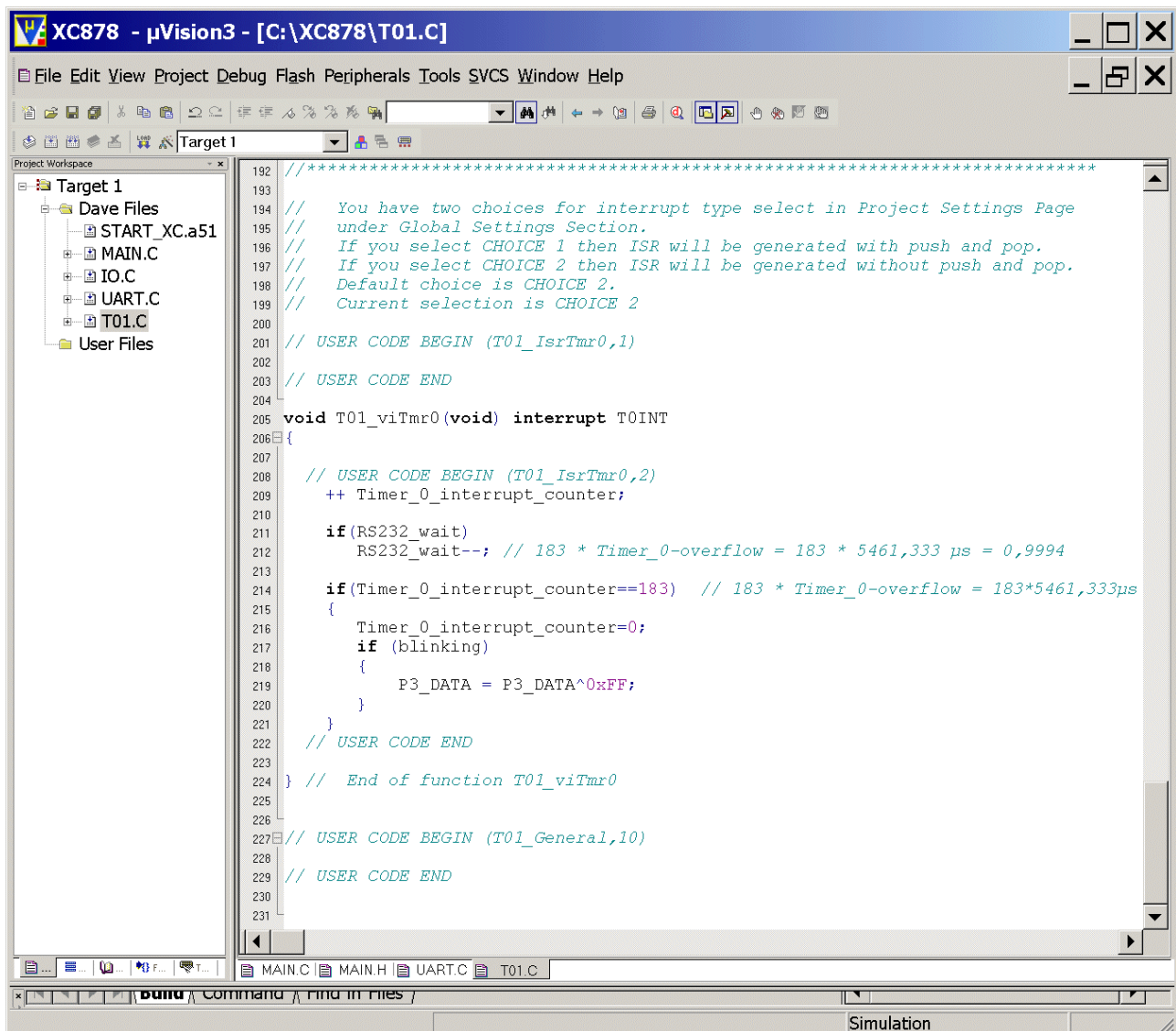
Double click T01.C

Insert code for T0 interrupt service routine:

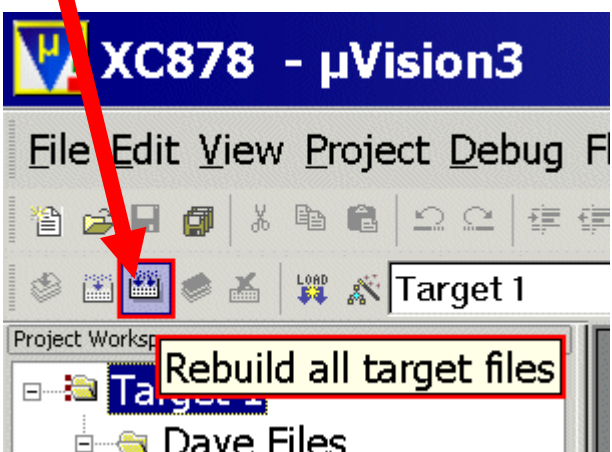
```
++ Timer_0_interrupt_counter;

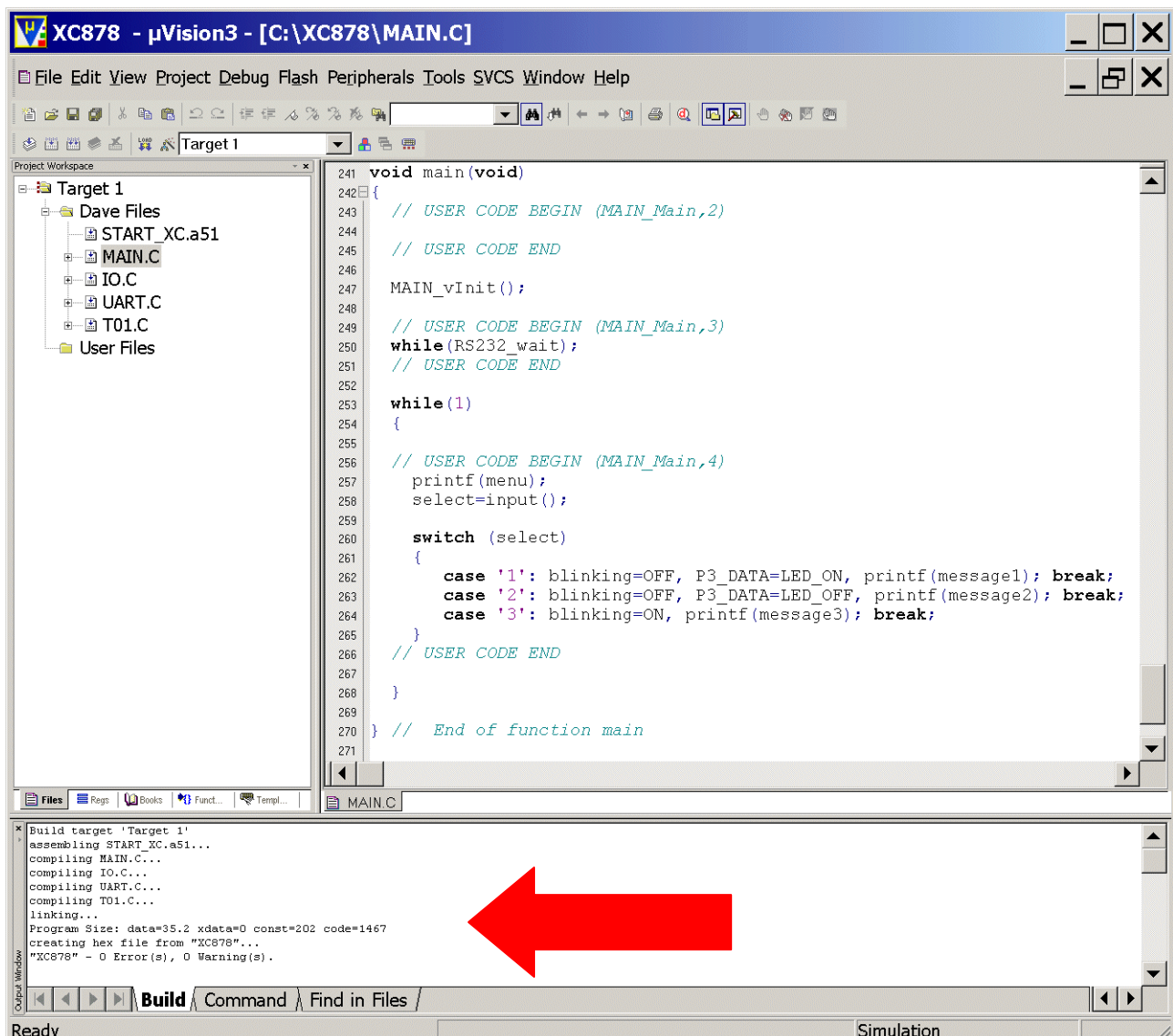
if(RS232_wait)
    RS232_wait--; // 183 * Timer_0-overflow = 183 * 5461,333 μs = 0,9994

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



Generate your application program:

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



```

241 void main(void)
242 {
243     // USER CODE BEGIN (MAIN_Main,2)
244
245     // USER CODE END
246
247     MAIN_vInit();
248
249     // USER CODE BEGIN (MAIN_Main,3)
250     while(RS232_wait);
251     // USER CODE END
252
253     while(1)
254     {
255
256         // USER CODE BEGIN (MAIN_Main,4)
257         printf(menu);
258         select=input();
259
260         switch (select)
261         {
262             case '1': blinking=OFF, P3_DATA=LED_ON, printf(message1); break;
263             case '2': blinking=OFF, P3_DATA=LED_OFF, printf(message2); break;
264             case '3': blinking=ON, printf(message3); break;
265         }
266         // USER CODE END
267     }
268
269 } // End of function main
270
271

```

Build target 'Target 1'
 assembling START_XC.a51...
 compiling MAIN.C...
 compiling IO.C...
 compiling UART.C...
 compiling T01.C...
 linking...
 Program Size: data=35.2 xdata=0 const=202 code=1467
 creating hex file from "XC878"...
 "XC878" - 0 Error(s), 0 Warning(s).

Now we close our project and μ Vision 3:

Project
Close Project

File

Exit

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



Start Keil μ Vision and open our Keil Project

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

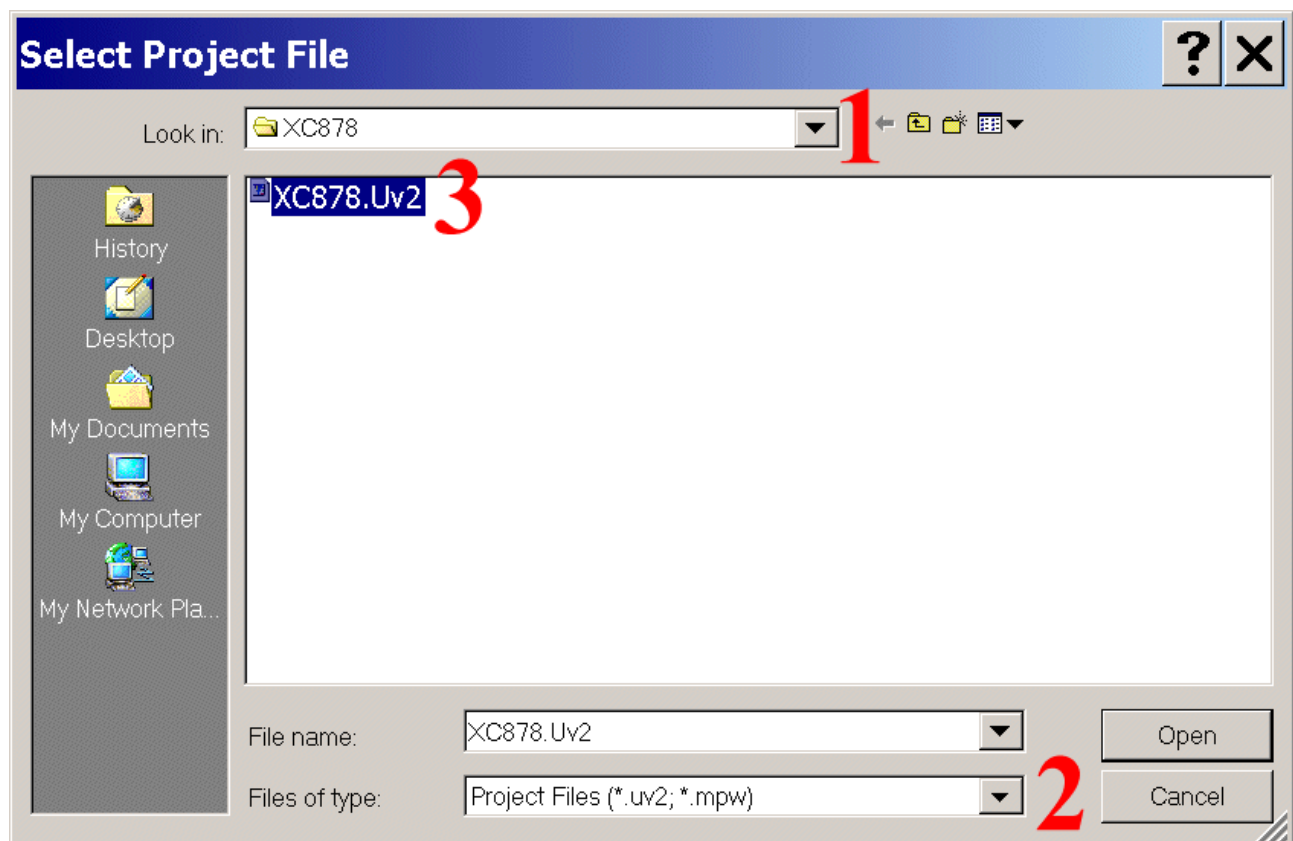
Project - Open Project

Select Project File: **Look in:** choose **C:\XC878** (1)

Select Project File: **Files of type:** select **Project Files (*.uv2)** (2)

Click XC878.Uv2 (3)

Click Open

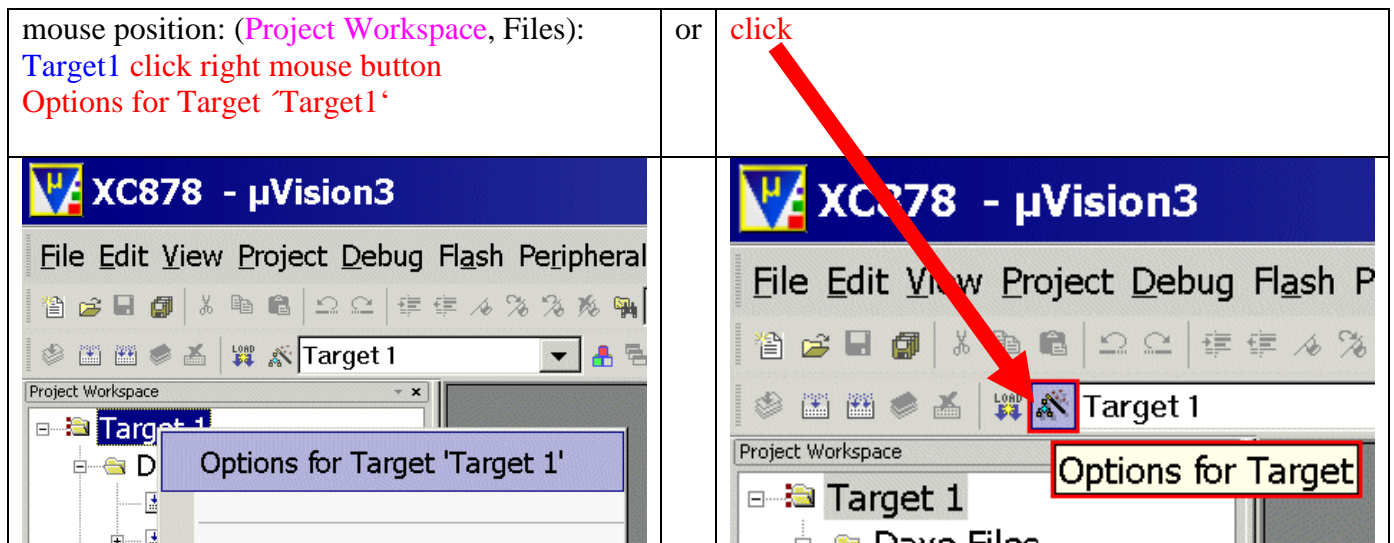


Note:

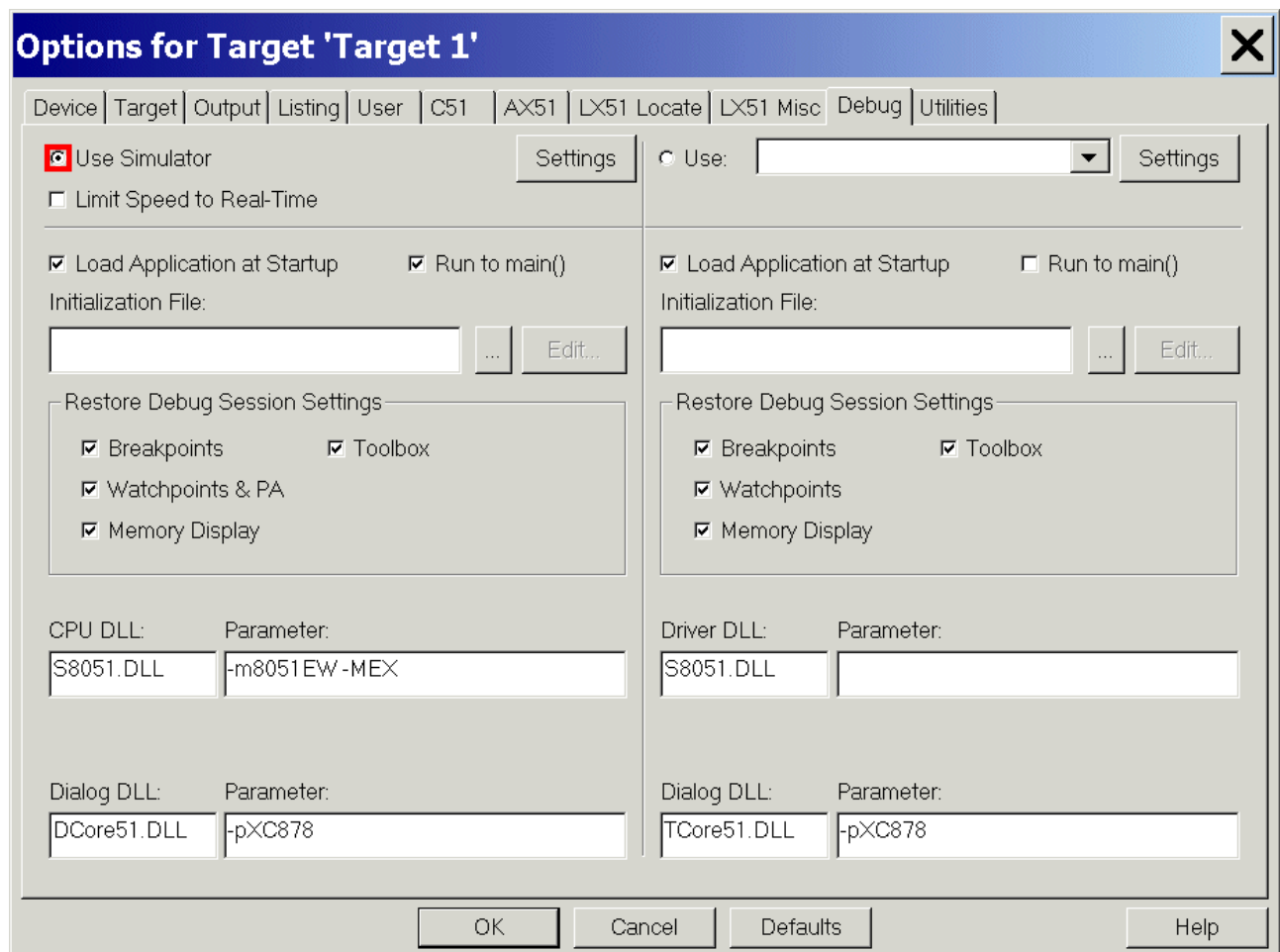
From now on just open your μ Vision project (not the DAvE project).
 μ Vision will automatically recognise if there has been a code regeneration done by DAvE!



Check the configuration of the μ Vision simulator:

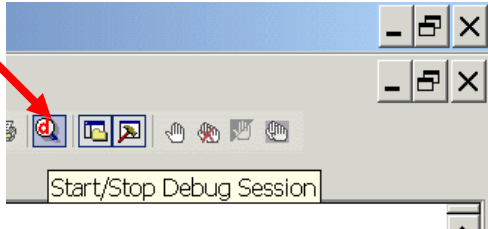


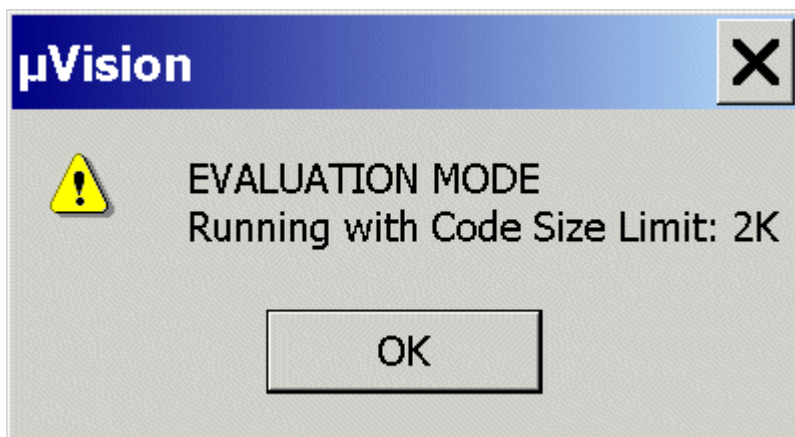
Options for Target 'Target1': Debug: check ☒ Use Simulator



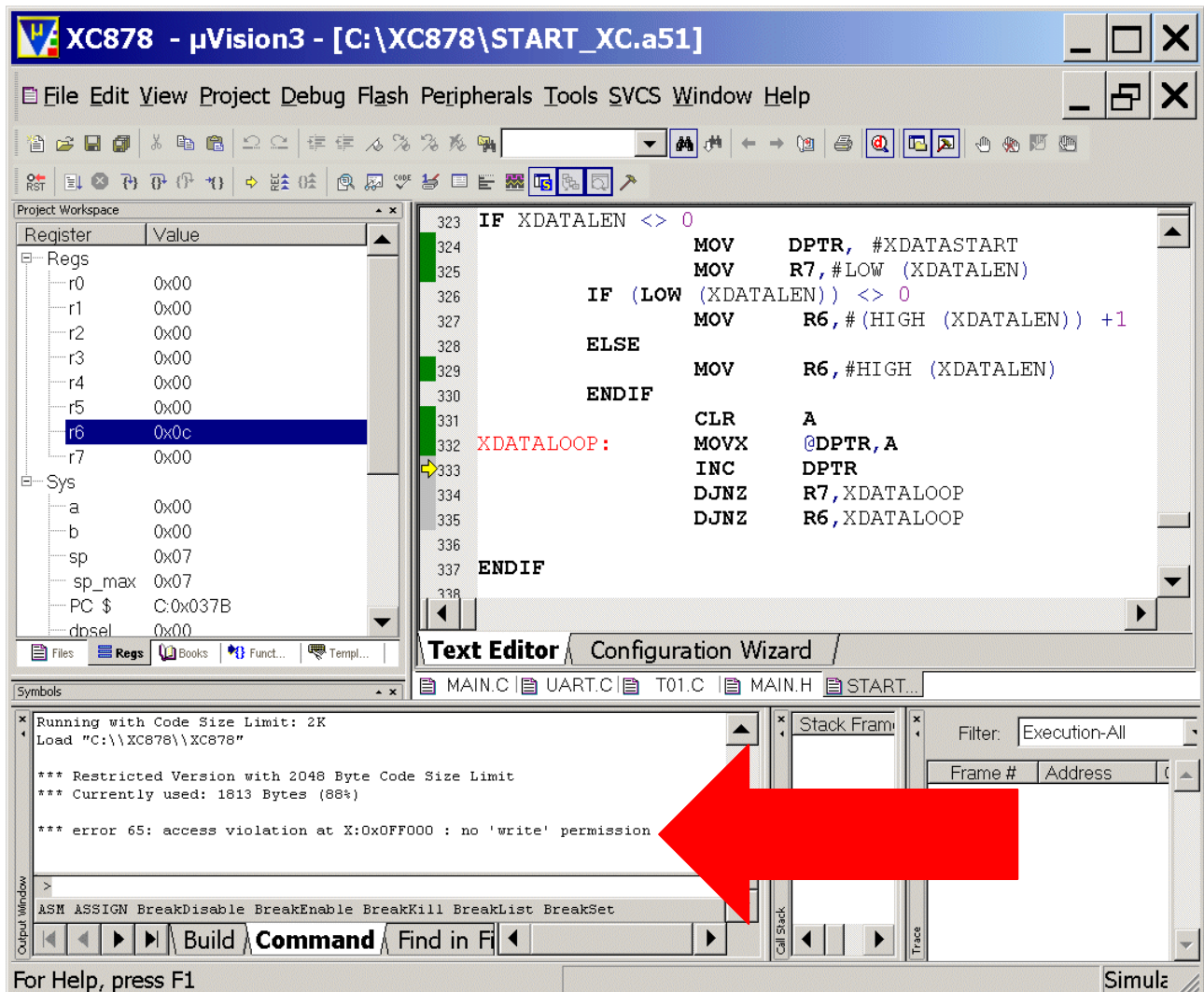
OK

Start the μ Vision Simulator:

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



OK



Note:

*** error 65: access violation at X:0x0FF000 : no 'write' permission

To get rid of error 65 we should look at either <http://www.keil.com/support/docs/814.htm> or the next two pages to better understand the problem.





µVISION DEBUGGER: ERROR 65 (ACCESS VIOLATION)

Information in this article applies to:

- µVision All Versions
- Cx51 All Versions

SYMPTOMS

My project includes a variable that I access using an absolute memory address. The µVision Debugger generates the following error whenever I try to write to the variable:

```
*** Error 65: Access violation at 0x00000004 : No 'write' permission
```

I have specified this memory area to the linker but I still receive this error.

CAUSE

When the µVision Debugger loads an executable program, it creates a memory map using the program and data segments from the program. Code segments are marked as executable while data segments are marked as read/write. All other memory is unmapped and is, therefore, not marked for any type of access.

The µVision Debugger checks that all memory accesses are made to objects that are defined in your C program. For any access that is outside of a defined C object, the µVision debugger generates an **error 65: access violation** message.

By default, the debugger allows only memory accesses to valid data objects. This is useful for finding uninitialized or incorrectly initialized pointers, for example. Usually, there is a programming error when you try to access unexpected locations.

RESOLUTION

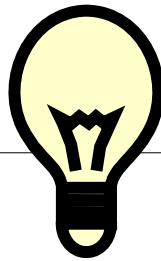
Enable memory regions for absolute memory accesses using the **MAP** debugger command or the **Debug - Memory Map** dialog. Define all the memory ranges that your program is permitted to access. Ensure that you set the permissions (read, write, execute) for each range to match your actual hardware.

You may put the required MAP commands into a debugger INI file that can be entered under **Project -> Options for Target -> Debug -> Initialization file**. For example, the content of such a file may be:

```
MAP 0xF800, 0xF8FF READ WRITE // allow R/W access to IO space
```

MORE INFORMATION

- Refer to [MAP](#) in the µVision® User's Guide.



MAP - Syntax Description:

MAP Displays the current memory map.

MAP start, end READ WRITE EXEC VNM Maps the specified memory range (start-end) accesses as specified.

MAP start, end CLEAR Clears a mapped memory range.

Target programs you debug with μ Vision3 access and use memory. μ Vision3 uses the symbol information in your target program to automatically setup the memory map for most applications. The MAP command lets you specify the memory areas your program uses that are not automatically detected by μ Vision3.

When you run your target program, μ Vision3 checks each memory access to determine if it is outside the memory map. If an invalid access is made, μ Vision3 reports an access violation error. This helps you locate and correct memory problems in your program.

If your program uses memory-mapped I/O devices or dynamically accesses memory through pointers, you may need to make changes to the memory map.

You specify an address range with the MAP command along with the accesses allowed for that range. Read (READ), write (WRITE), and execution (EXEC) accesses (or any combination) may be specified. The memory map supports 1-byte granularity.

The VNM option identifies the specified memory range as von Neumann memory. When VMN is specified with an address range, μ Vision3 overlaps external data memory (XDATA) and code memory. Write accesses to external data memory also change code memory. Memory ranges specified with VNM may not be a range from the code area and may not cross a 64K boundary. The address range specified must be from the external data area.

The MAP command, when entered with no other parameters, displays the current memory map for your target program. This lets you check your memory map settings.

The CLEAR option lets you remove an address range previously specified with the MAP command.

When μ Vision3 loads, the following memory maps are defined.

CPU Address Range Access

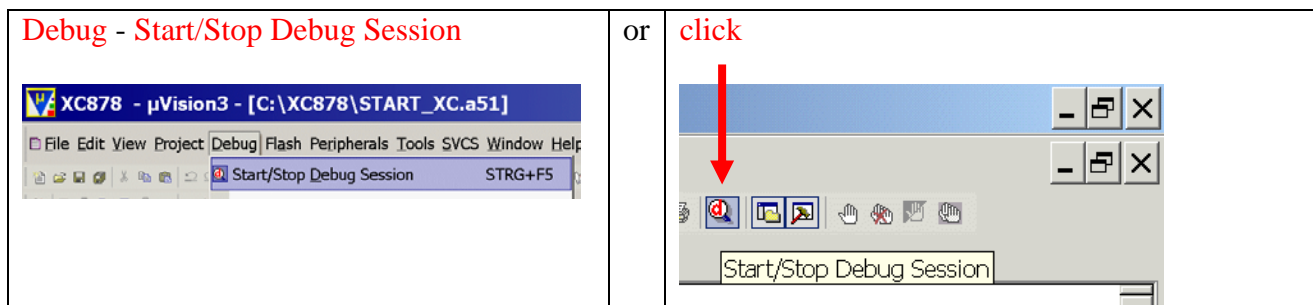
```
8051 Family 0x000000-0x00FFFF (DATA) READ WRITE
             0x010000-0x01FFFF (XDATA) READ WRITE
             0xFF0000-0xFFFFFFF (CODE) EXEC READ
```

μ Vision3 supports up to 16MB of memory.

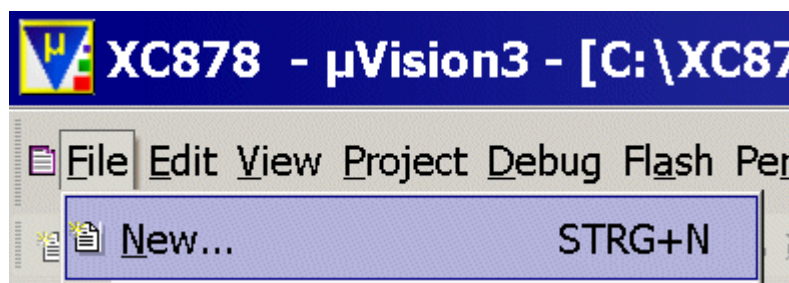
Simulate external I/O Devices:

External I/O devices are typical memory mapped. You may simulate such I/O devices with the Memory Window provided in the μ Vision debugger. Since the C user program does not contain any variable declarations for such memory regions it is required that you map this memory with the MAP command:

```
MAP 0x100000, 0x100FFF READ WRITE          /* MAP memory for I/O area */
```

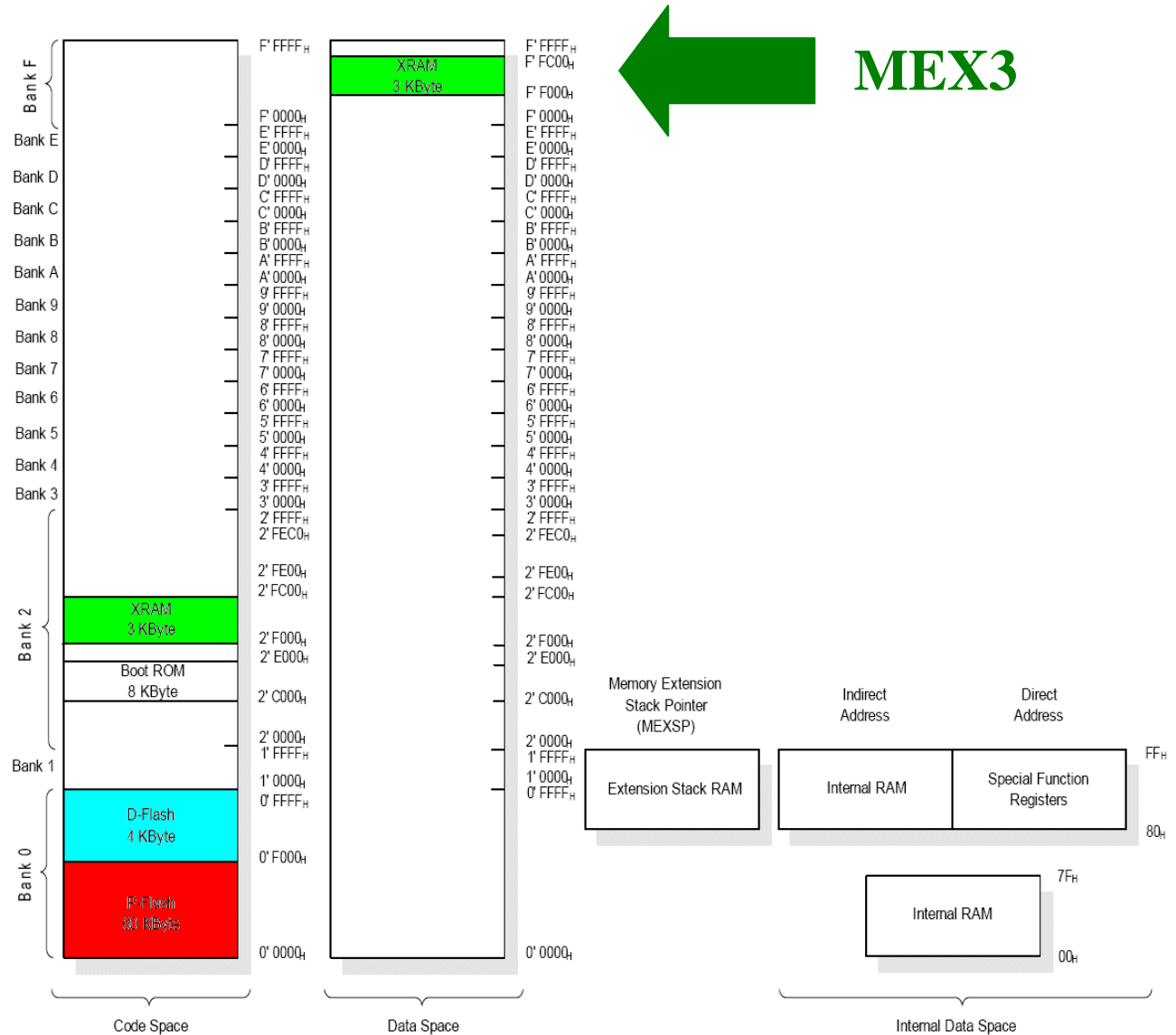


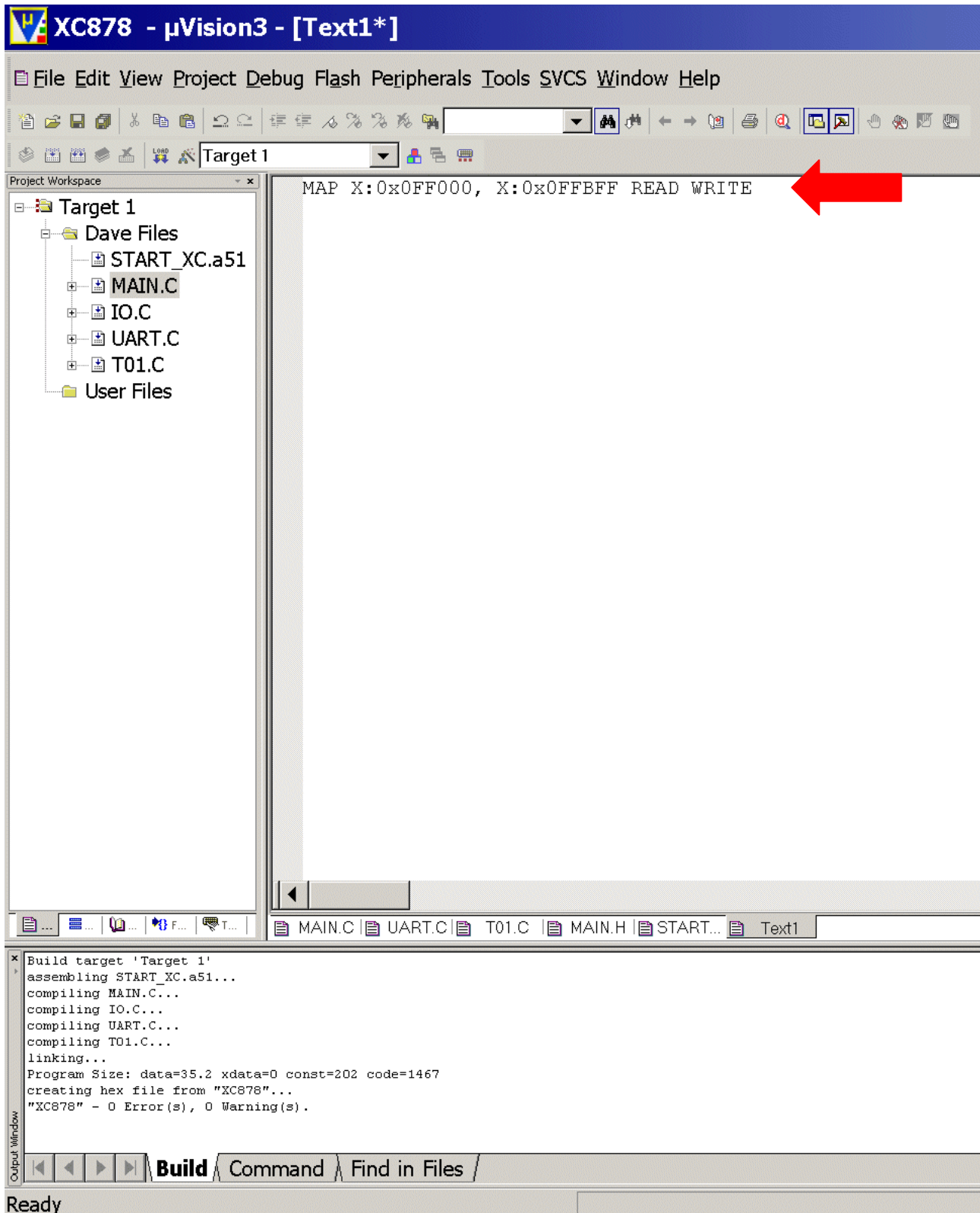
File – New...



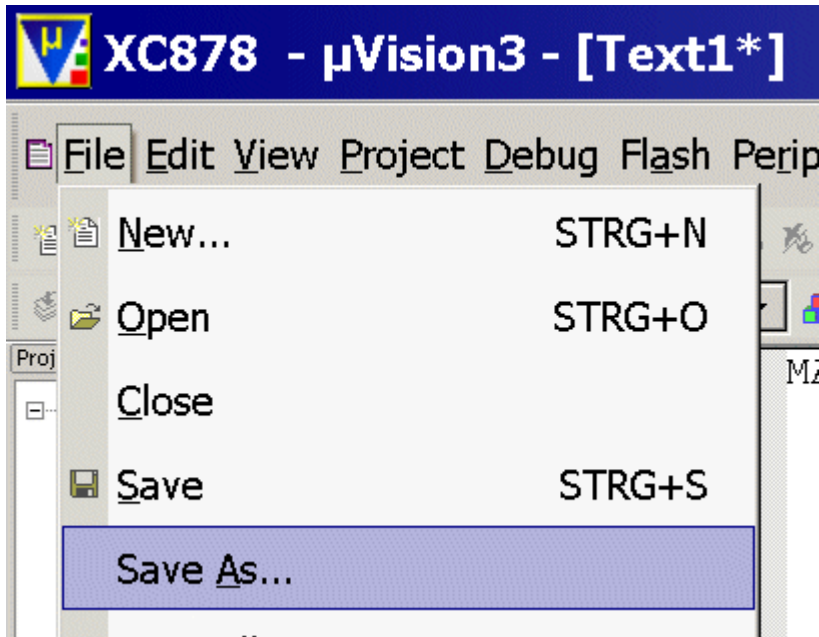
Insert:

MAP X:0x0FF000, X:0x0FFBFF READ WRITE

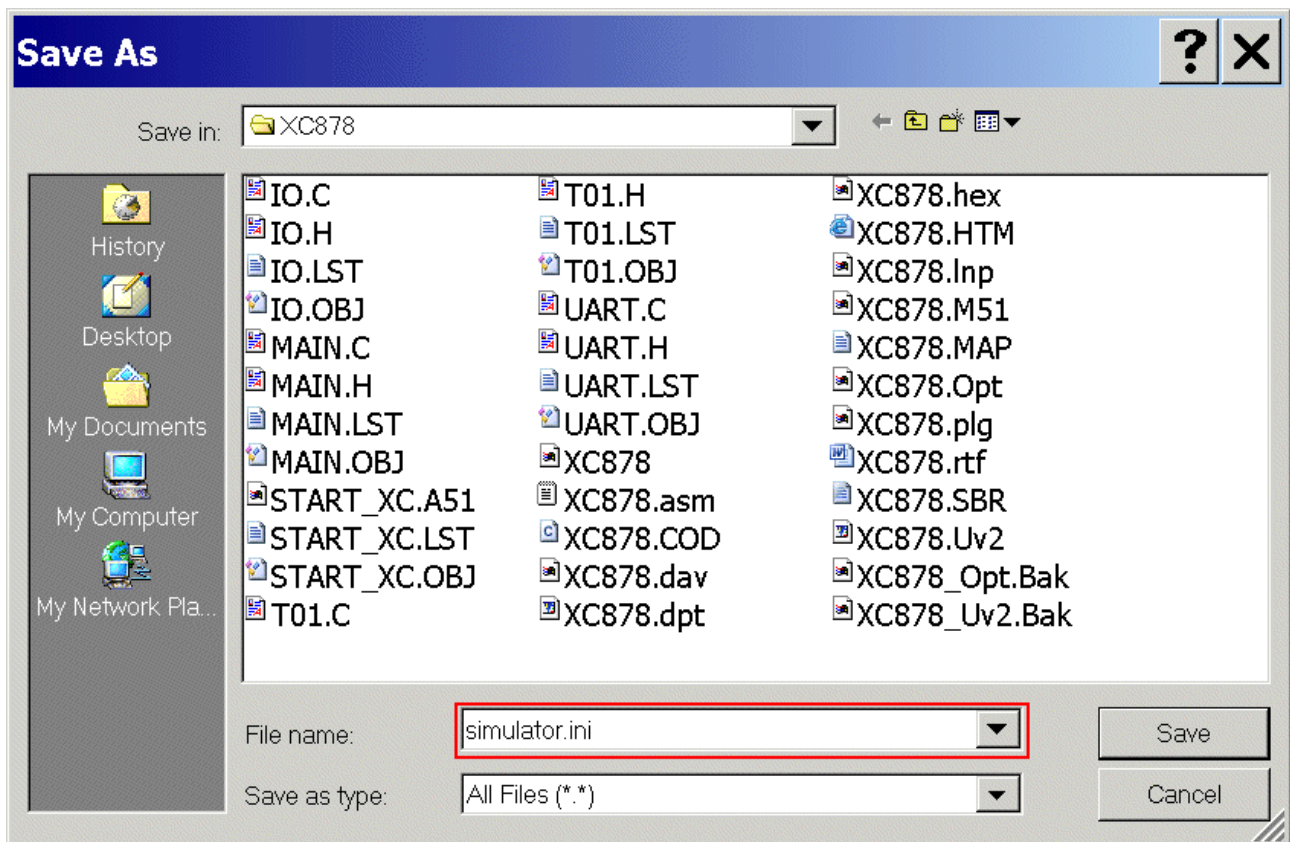




File – Save As...

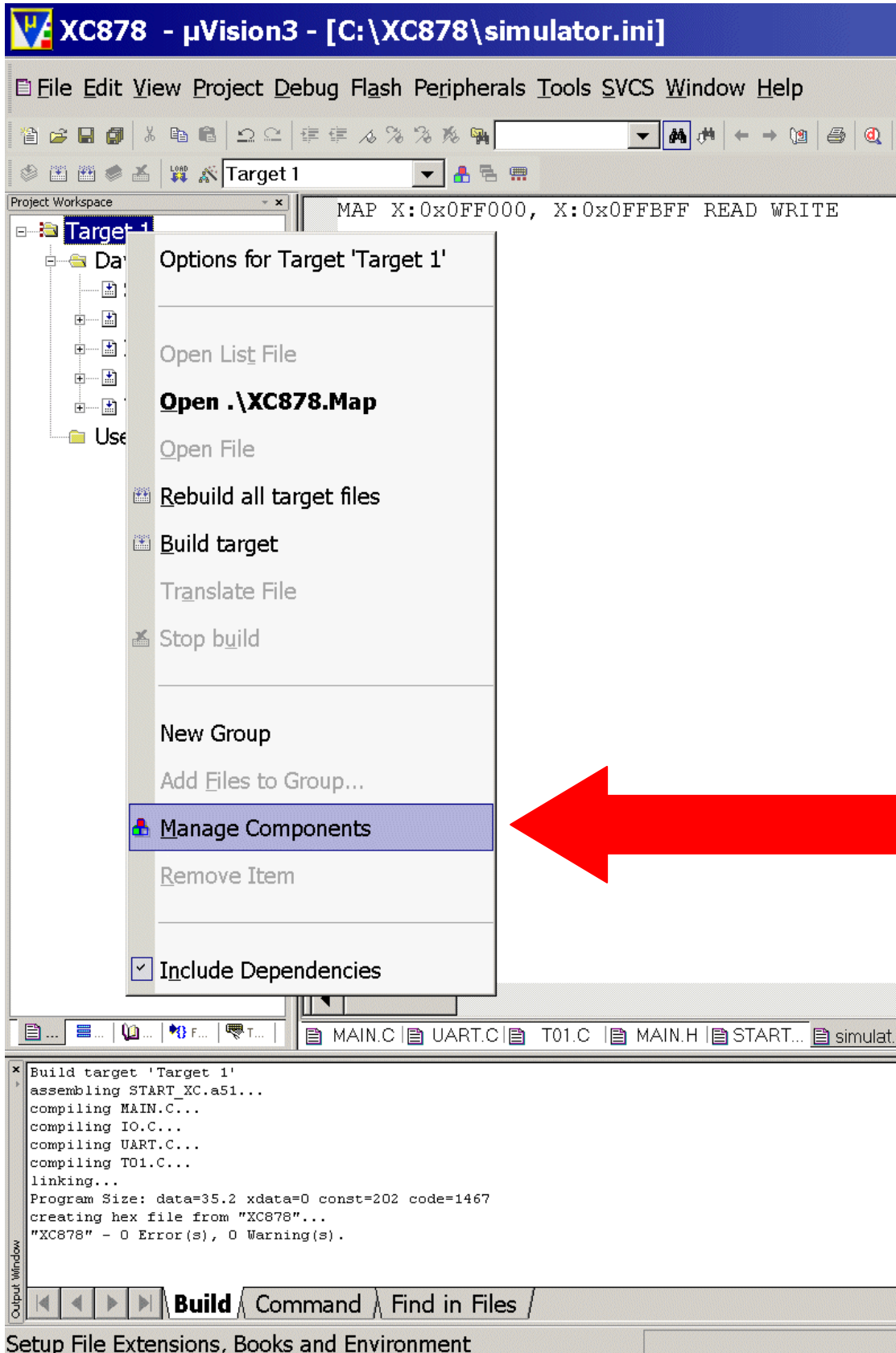


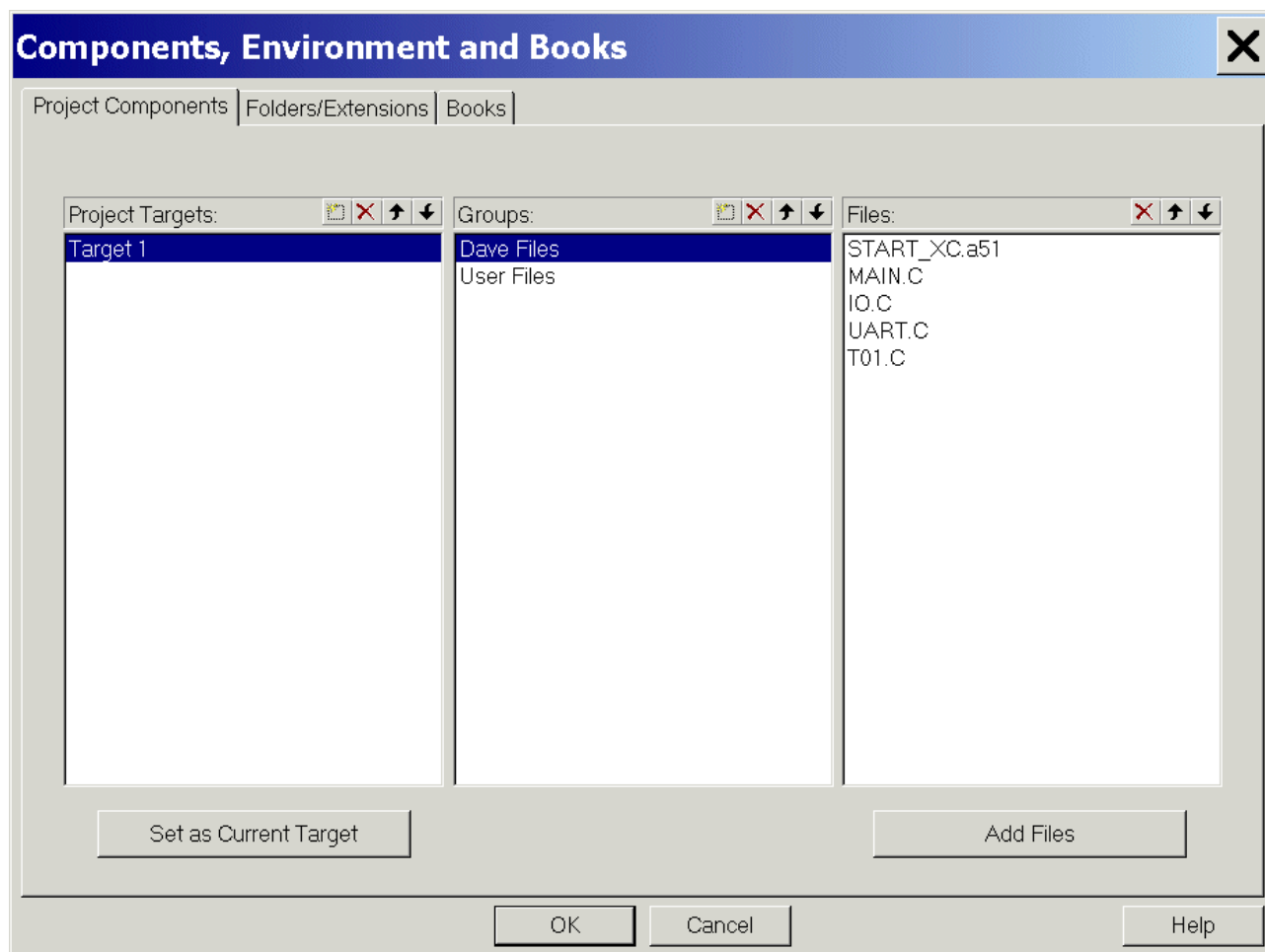
Save As: File name: insert: simulator.ini



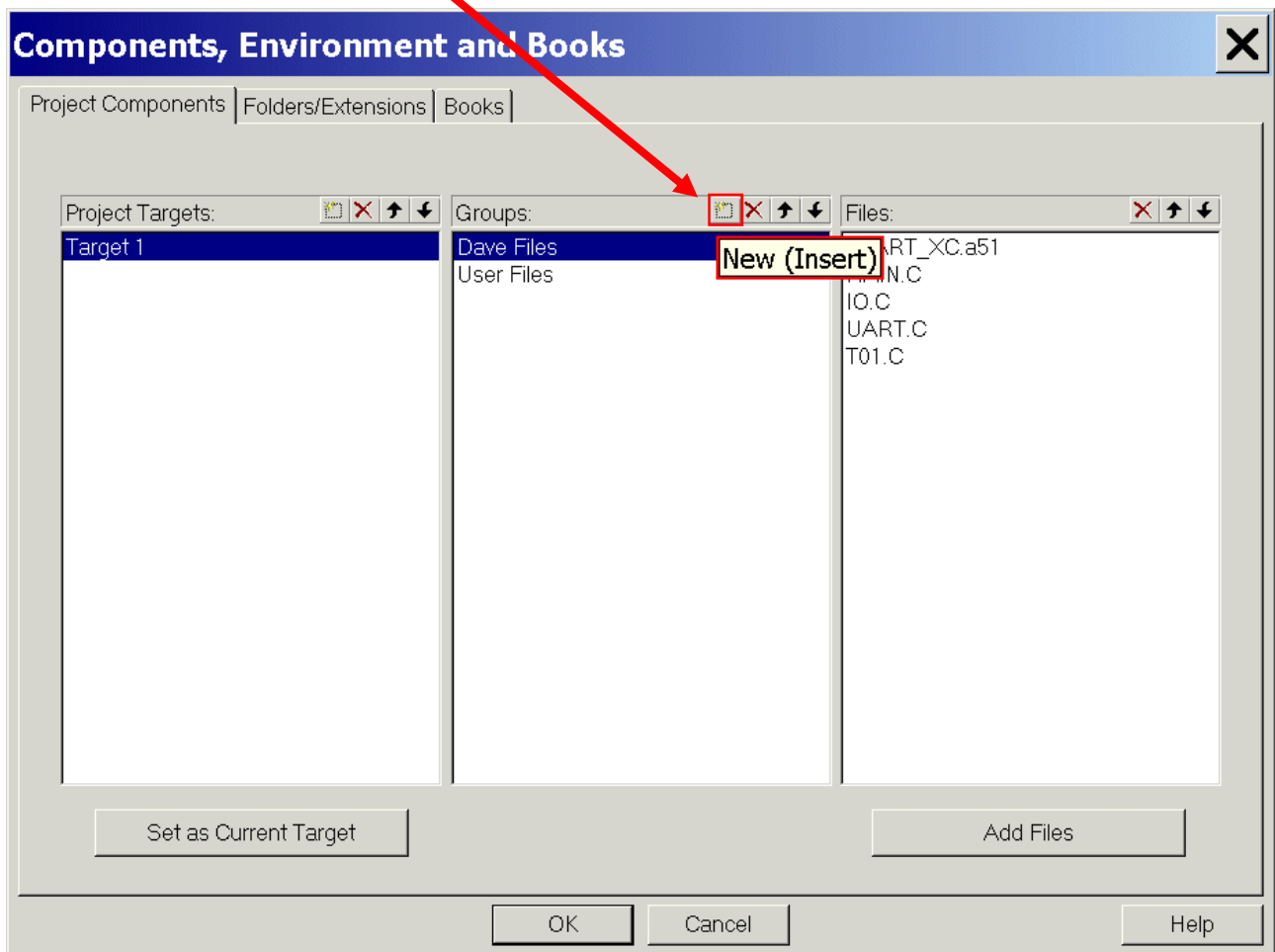
Click Save

Mouse position: **Project Window**, Target1: **click right mouse button**
click Manage Components

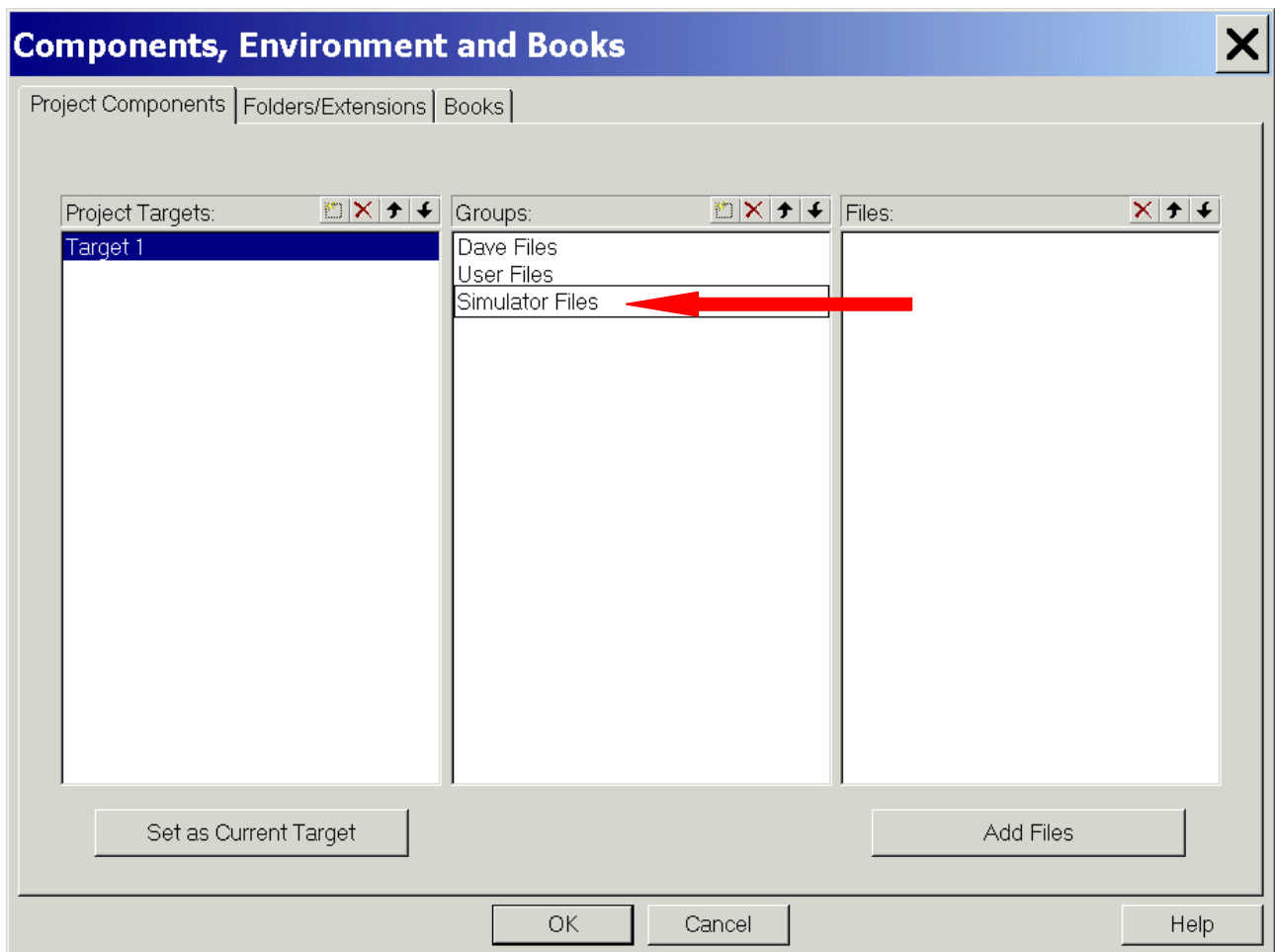




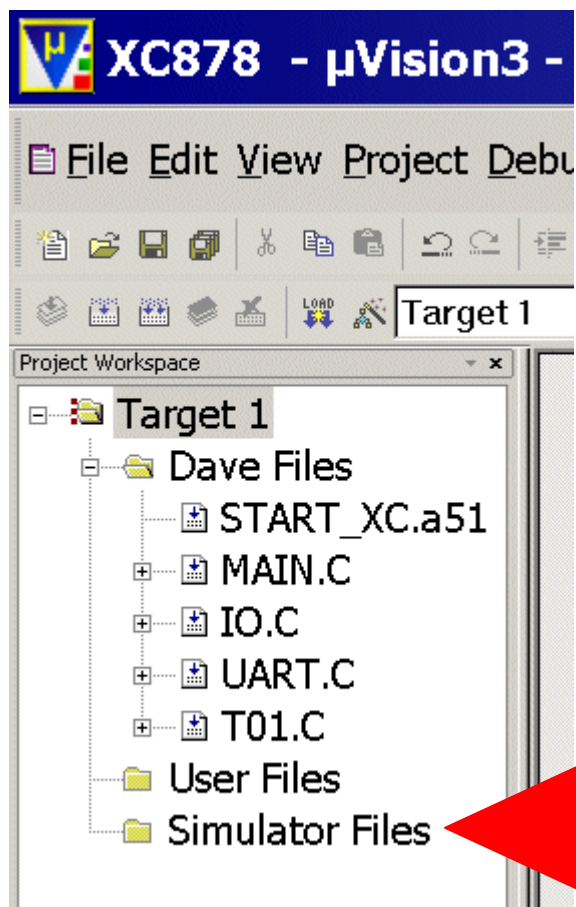
Project Components: Groups: **click** New (Insert)



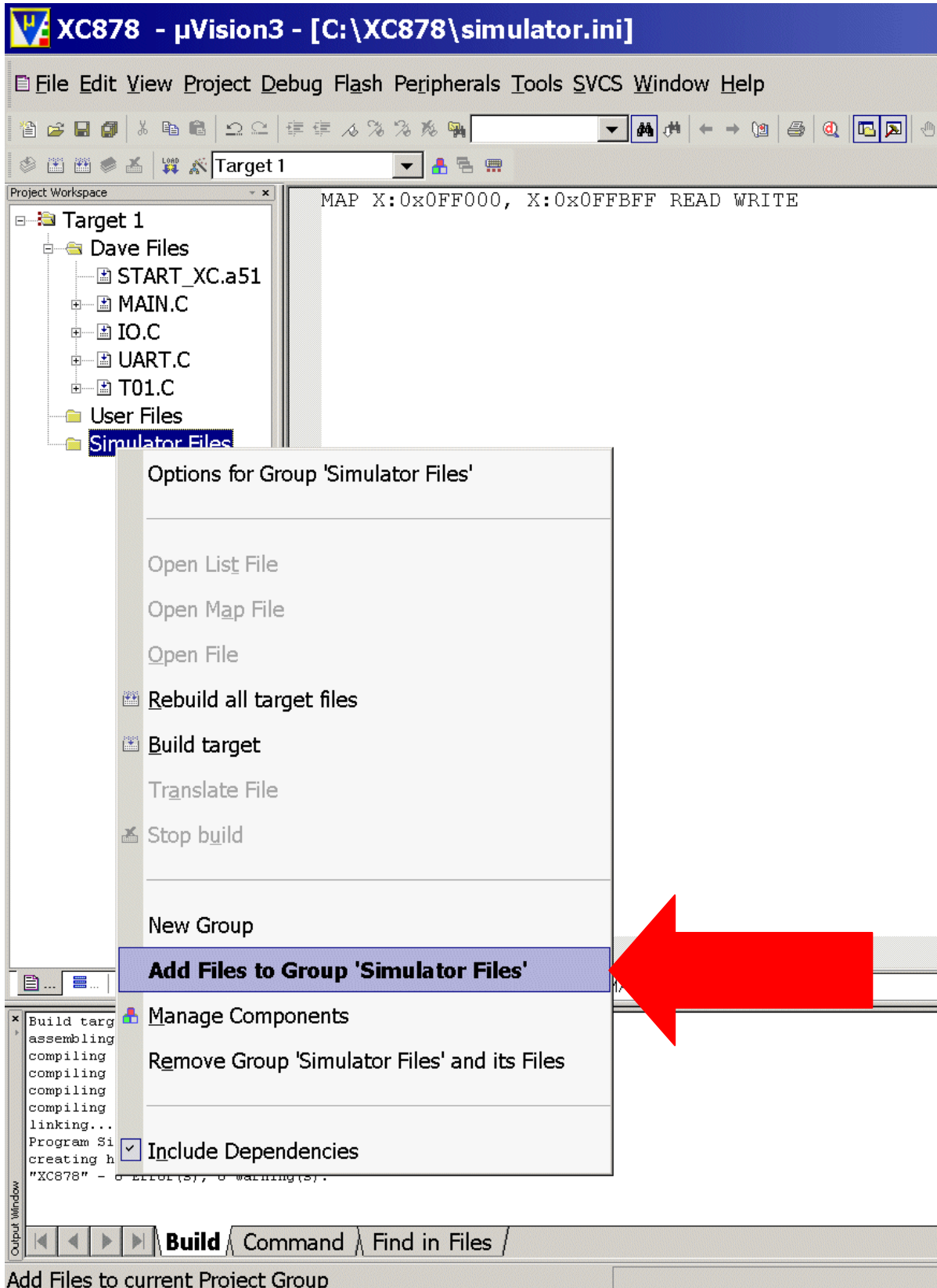
Insert Simulator Files



Click OK

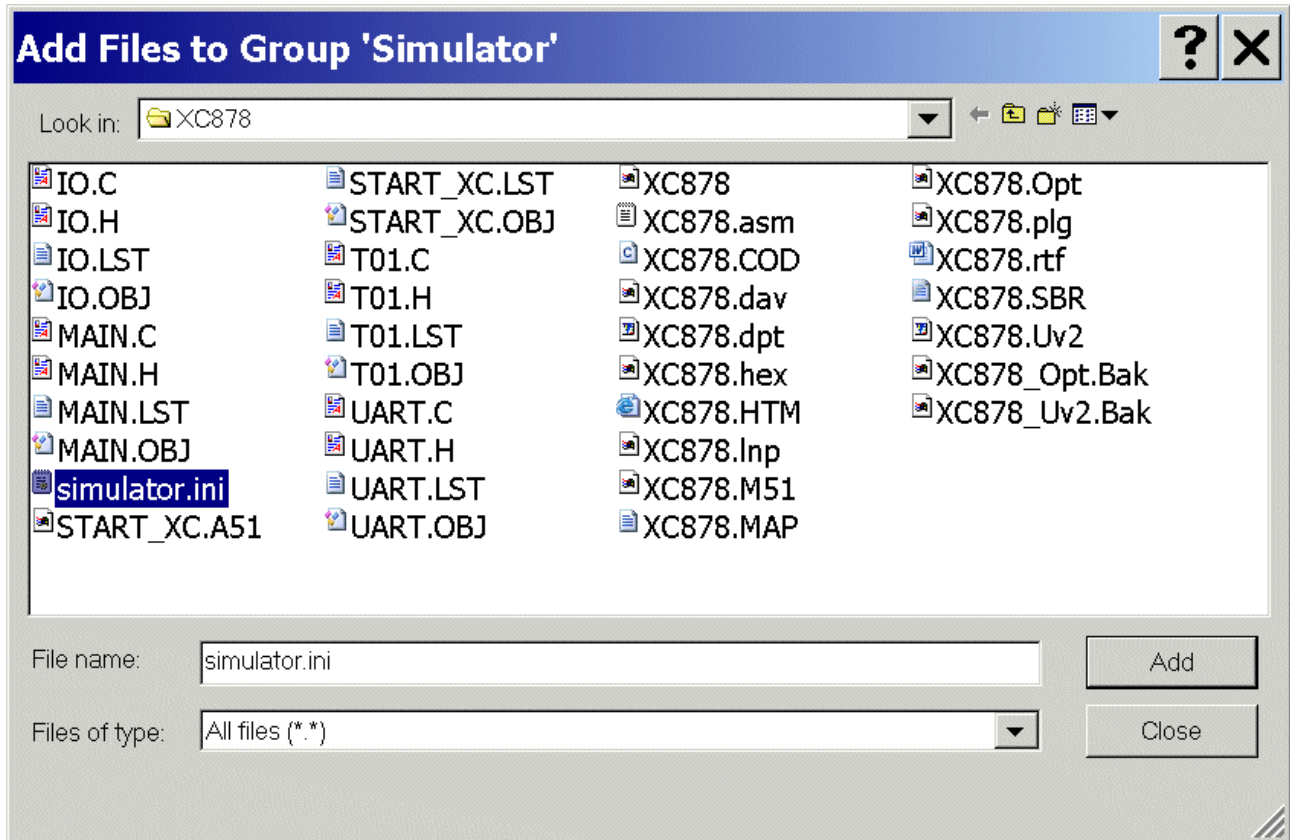


Mouse position: **Project Window**, Simulator Files: **click right mouse button**
Click Add Files to Group 'Simulator Files'



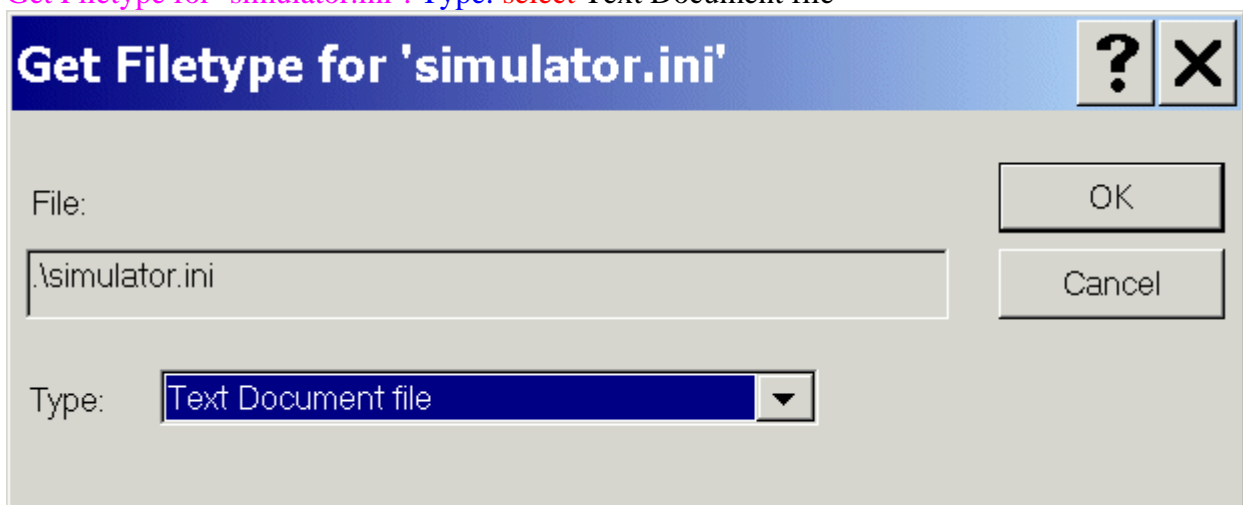
Files of type: select All files (*.*)

Click simulator.ini



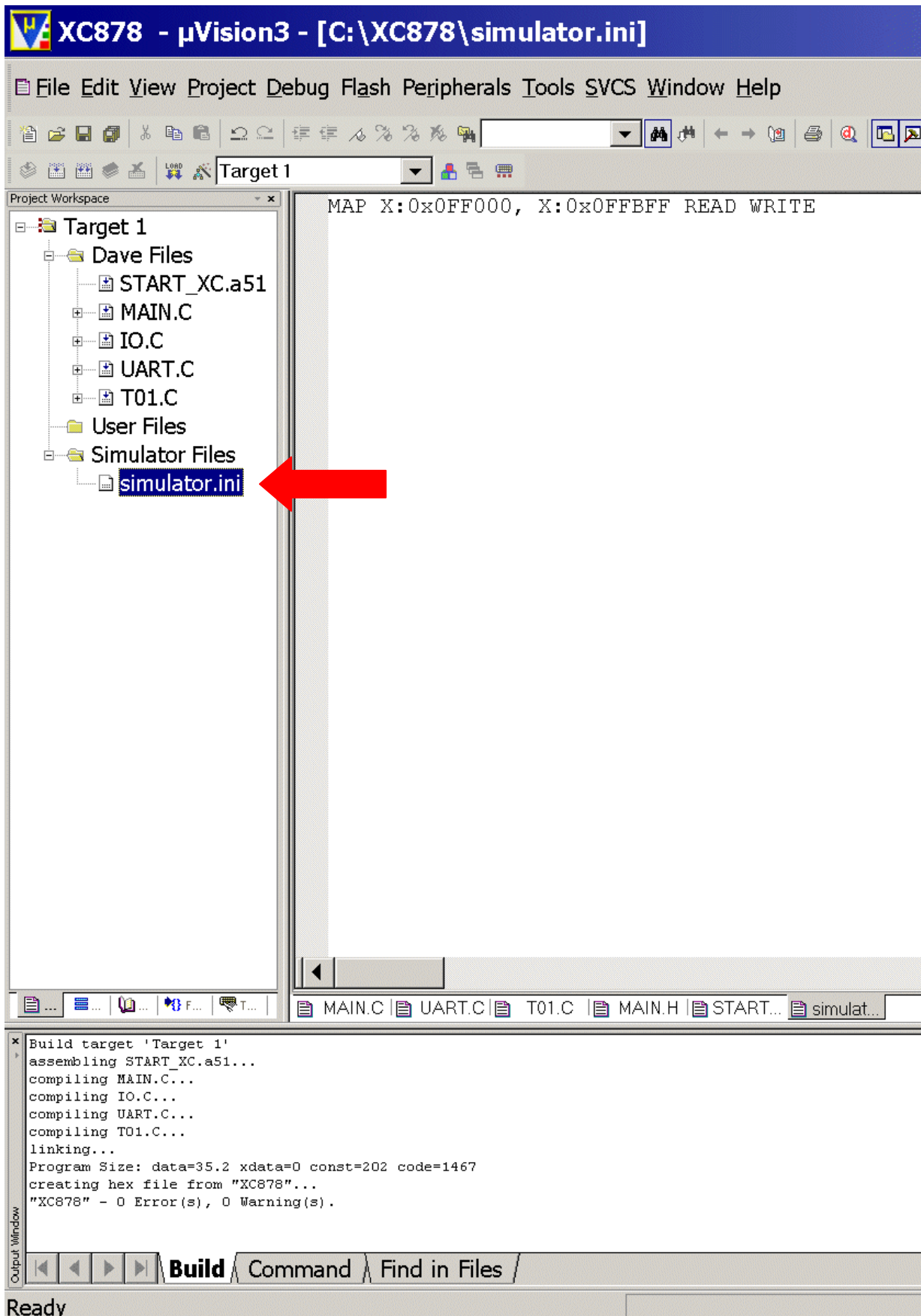
Click Add

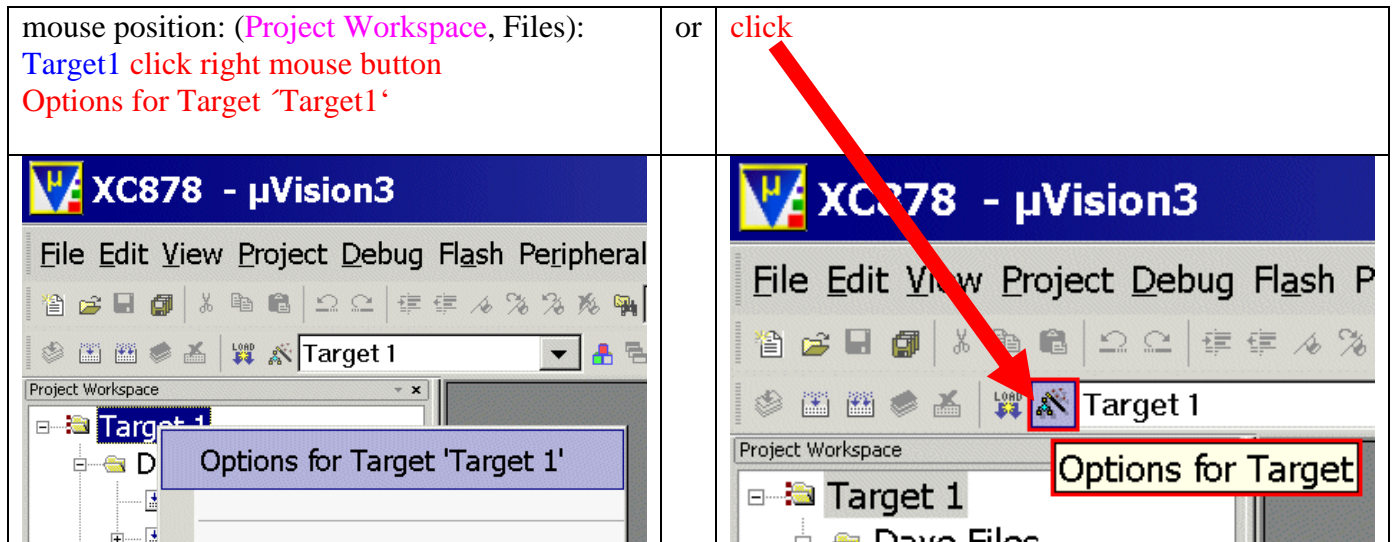
Get Filetype for 'simulator.ini': Type: select Text Document file



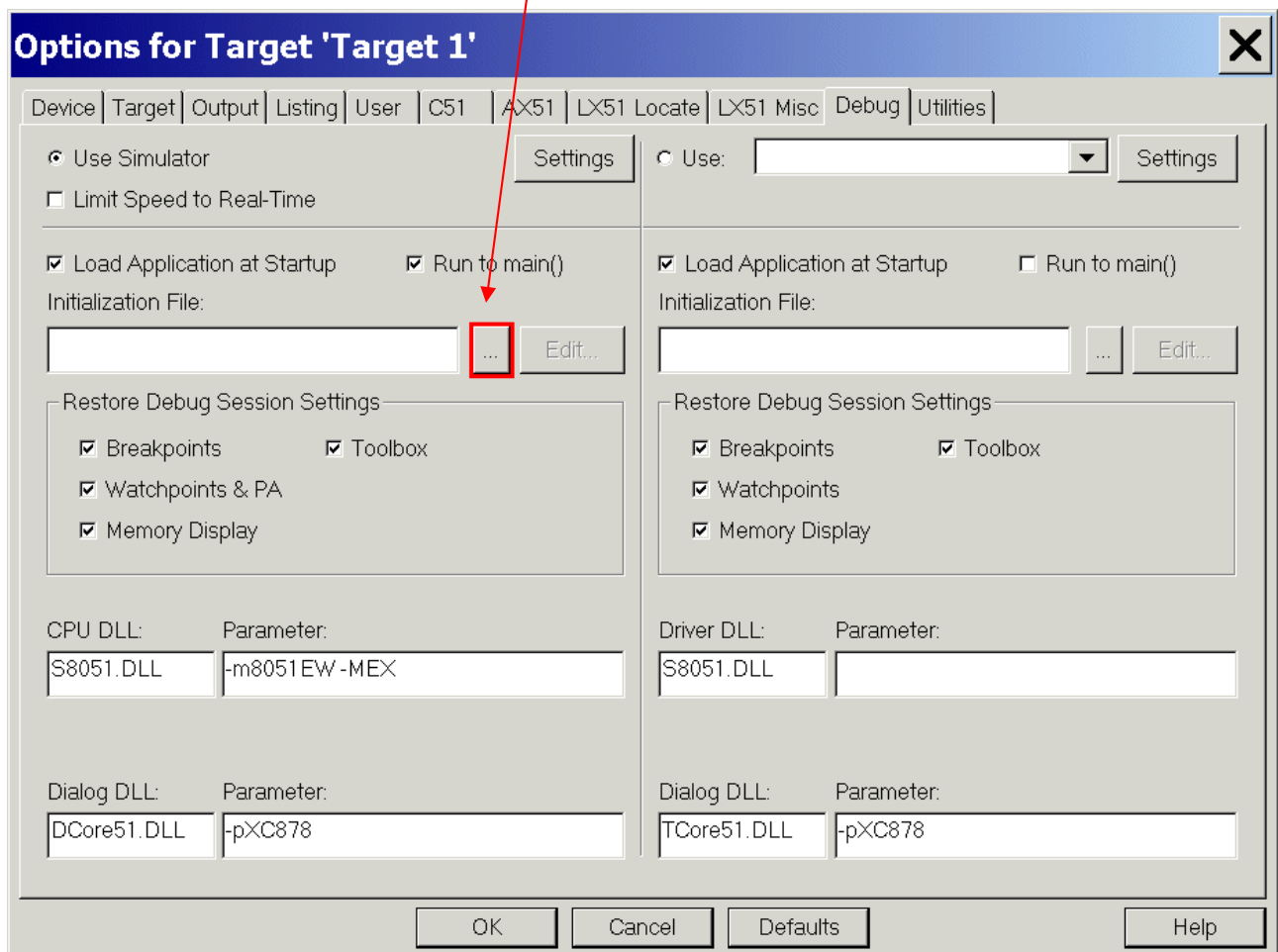
Click OK

Click Close

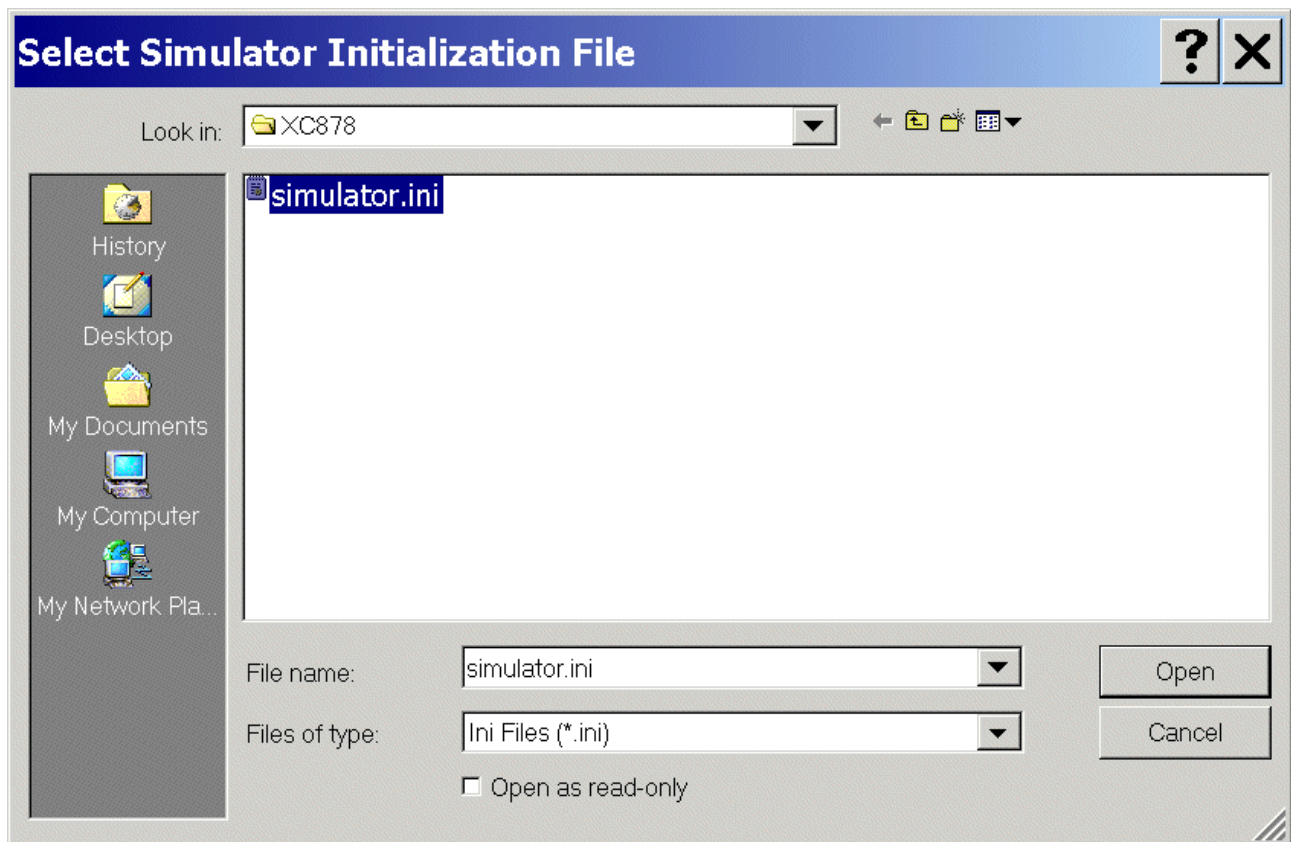




Options for Target 'Target1': Debug: click ...



Click/select simulator.ini



Click Open

Options for Target 'Target 1' [X]

Device | Target | Output | Listing | User | C51 | AX51 | LX51 Locate | LX51 Misc | Debug | Utilities

☒ Use Simulator Settings ☐ Use: Settings

☐ Limit Speed to Real-Time

☒ Load Application at Startup ☒ Run to main()
Initialization File: ... Edit...

Restore Debug Session Settings

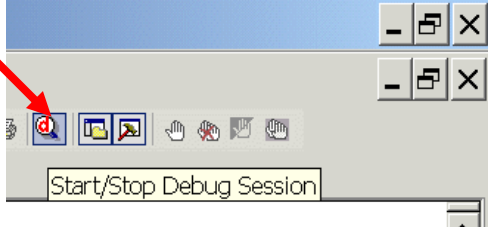
☒ Breakpoints ☒ Toolbox
☒ Watchpoints & PA
☒ Memory Display

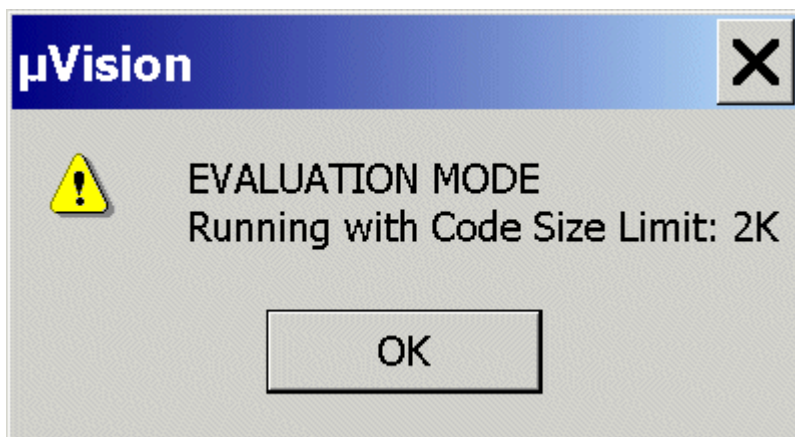
CPU DLL: Parameter:

Dialog DLL: Parameter:

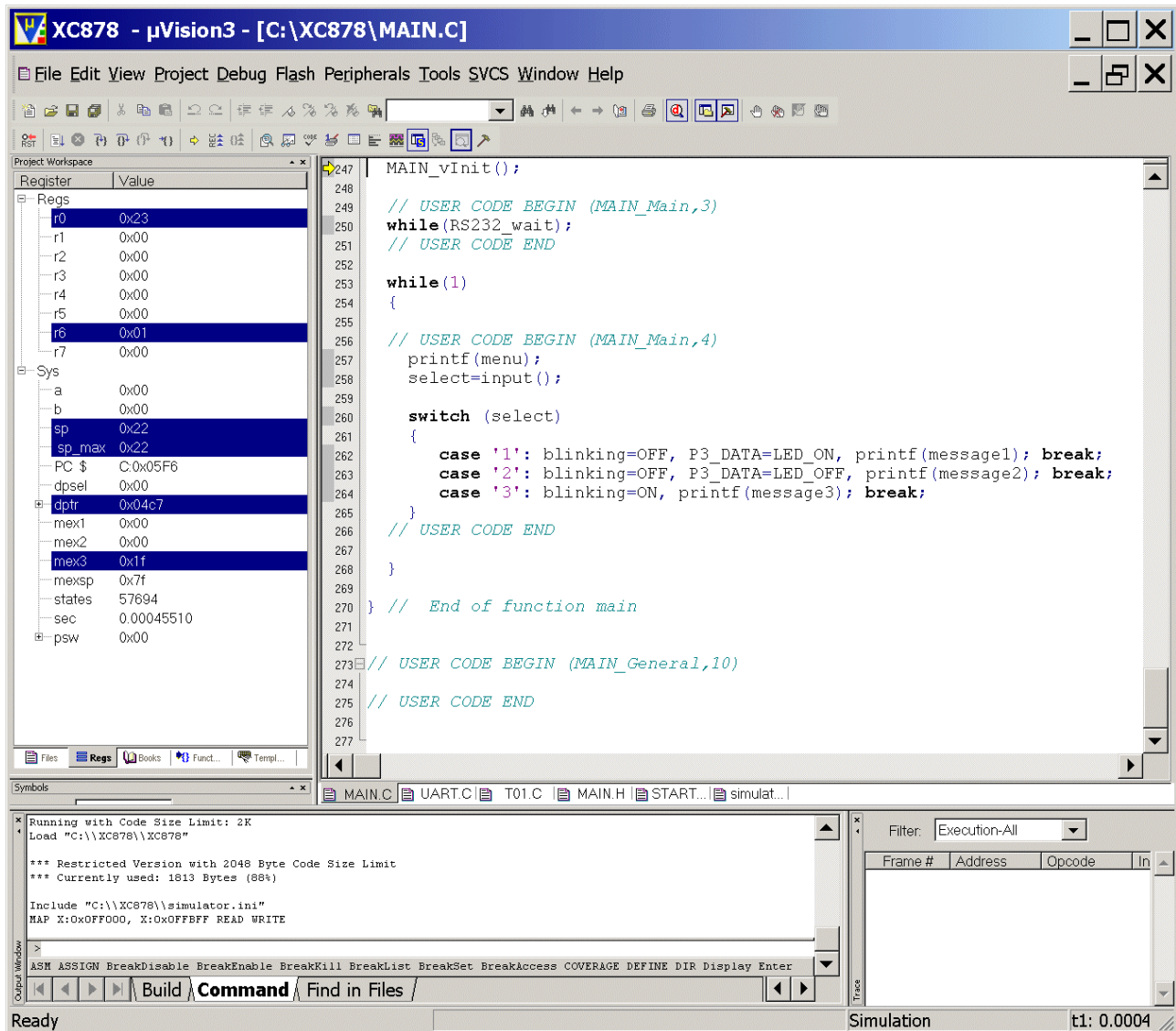
Click OK

Start the μ Vision Simulator again:

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



OK



Note:
No more ERROR 65.

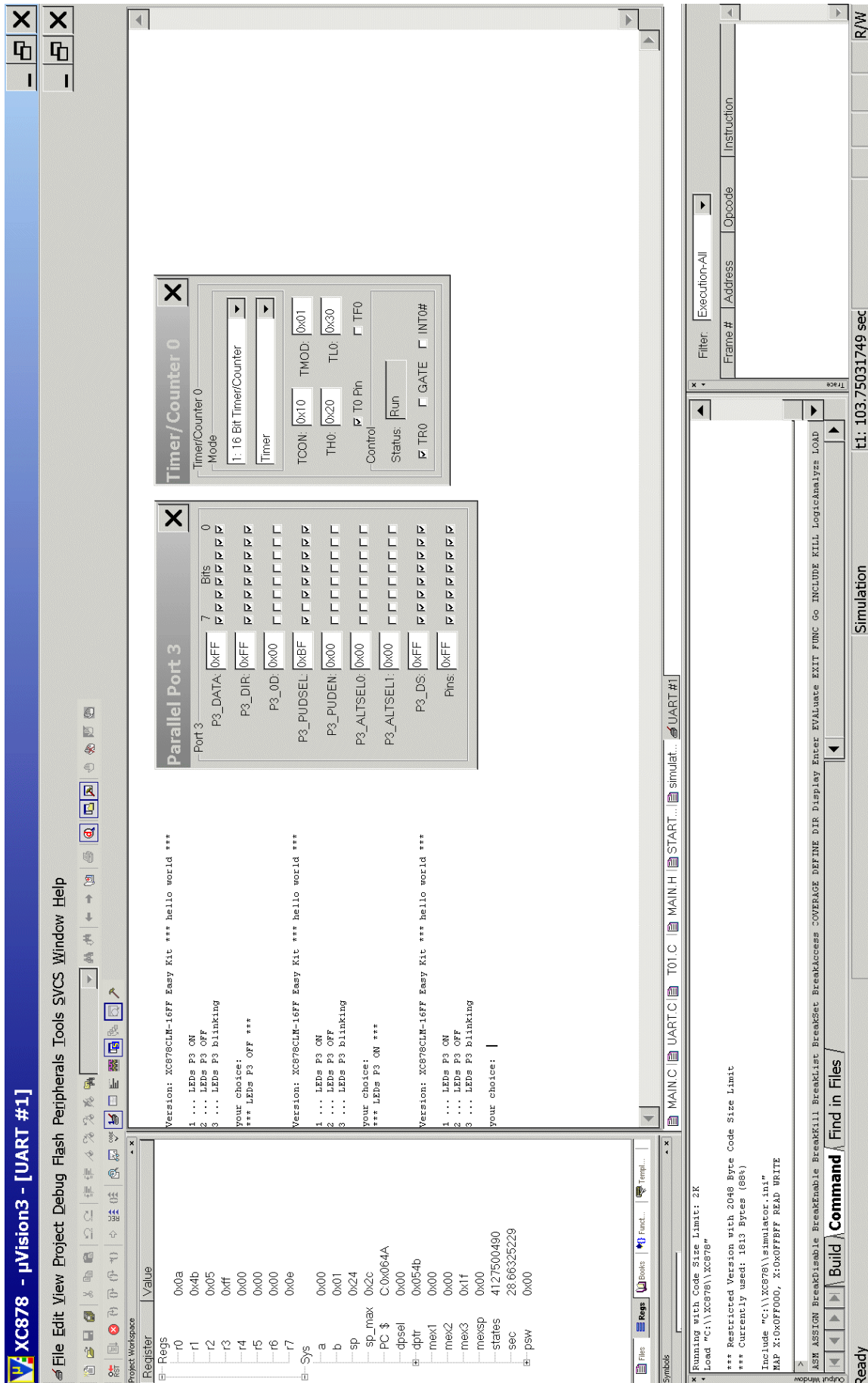
Start program execution:

Debug – Run

View - Serial Window – UART #1

Peripherals - I/O-Ports – Port3

Peripherals – Timer – Timer0



XC878 - µVision3 - [UART #1]

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help

Project Workspace

```

Version: XC878CLM-16FF Easy Kit *** hello world ***
1 ... LEDs P3 ON
2 ... LEDs P3 OFF
3 ... LEDs P3 Blinking
your choice:
*** LEDs P3 OFF ***

Version: XC878CLM-16FF Easy Kit *** hello world ***
1 ... LEDs P3 ON
2 ... LEDs P3 OFF
3 ... LEDs P3 Blinking
your choice:
*** LEDs P3 ON ***

Version: XC878CLM-16FF Easy Kit *** hello world ***
1 ... LEDs P3 ON
2 ... LEDs P3 OFF
3 ... LEDs P3 Blinking
your choice:

```

Register

Register	Value
r0	0x0a
r1	0x4b
r2	0x05
r3	0xff
r4	0x00
r5	0x00
r6	0x00
r7	0x0e
Sys	
a	0x00
b	0x01
sp	0x24
sp_max	0x2c
PC \$	C:0x064A
dpsel	0x00
dpr	0x054b
mex1	0x00
mex2	0x00
mex3	0x1f
mexsp	0x00
states	4127500490
sec	286635229
psw	0x00

Parallel Port 3

Port 3

7 Bits	0
P3_DATA	0xFF
P3_DIR	0xFF
P3_OD	0x00
P3_PUDSEL	0xBF
P3_PUDEN	0x00
P3_ALTSSEL0	0x00
P3_ALTSSEL1	0x00
P3_DS	0xFF
Pins	0xFF

Timer/Counter 0

TimerCounter 0

Mode: 1. 16 Bit Timer/Counter

TOON: 0x10 TMOD: 0x01

TH0: 0x20 TL0: 0x30

☒ T0 Pin ☐ TFO

Control: Run

Status: ☒ TR0 ☐ GATE ☐ INT0#

UART #1

MAIN.C UART.C T01.C MAIN.H START... simulat...

Running with Code Size Limit: 2K

Load "C:\XC878\XC878"

*** Restricted Version with 2048 Byte Code Size Limit

*** Currently used: 1813 Bytes (88%)

Include "C:\XC878\simulator.ini"

MAP X:0xFF000, X:0xFF000 READ WRITE

ASW ASSTON BreakDisable BreakEnable BreakList BreakSet BreakAccess COVERAGE DEFINE DIR Display Enter Evaluate EXIT FUNC Go INCLUDE KILL LogicalAnalyze LOAD

Ready

Simulation

tl: 103.75031749 sec

R/W

Note:

By activating (clicking) the **UART #1**-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 μ Vision 3:

Project - Close Project

File – Exit





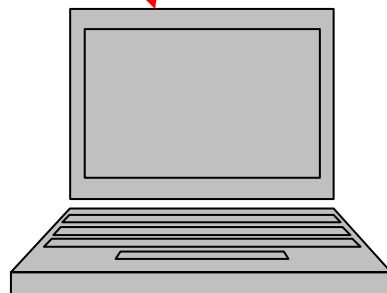
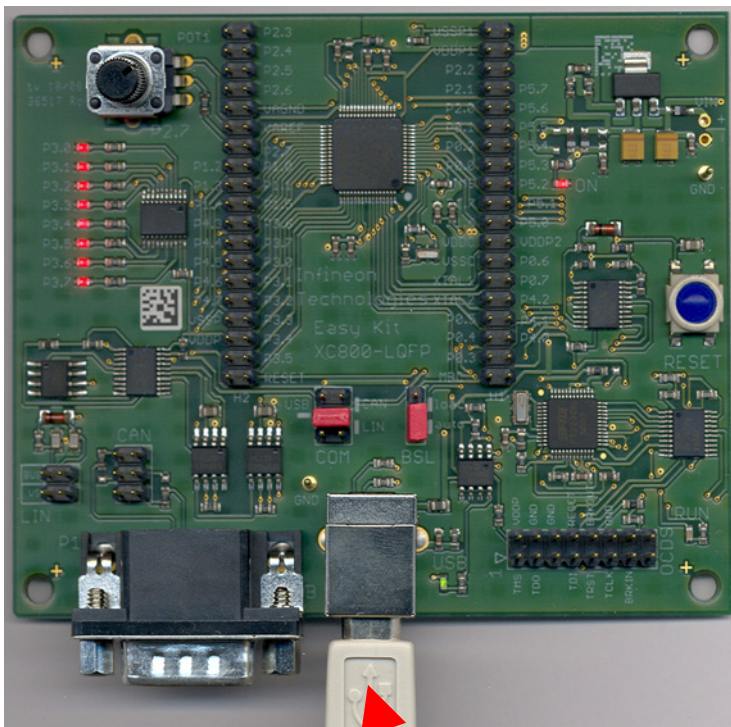
Note:

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 XC878 Easy 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 μ Vision and open our Keil Project

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

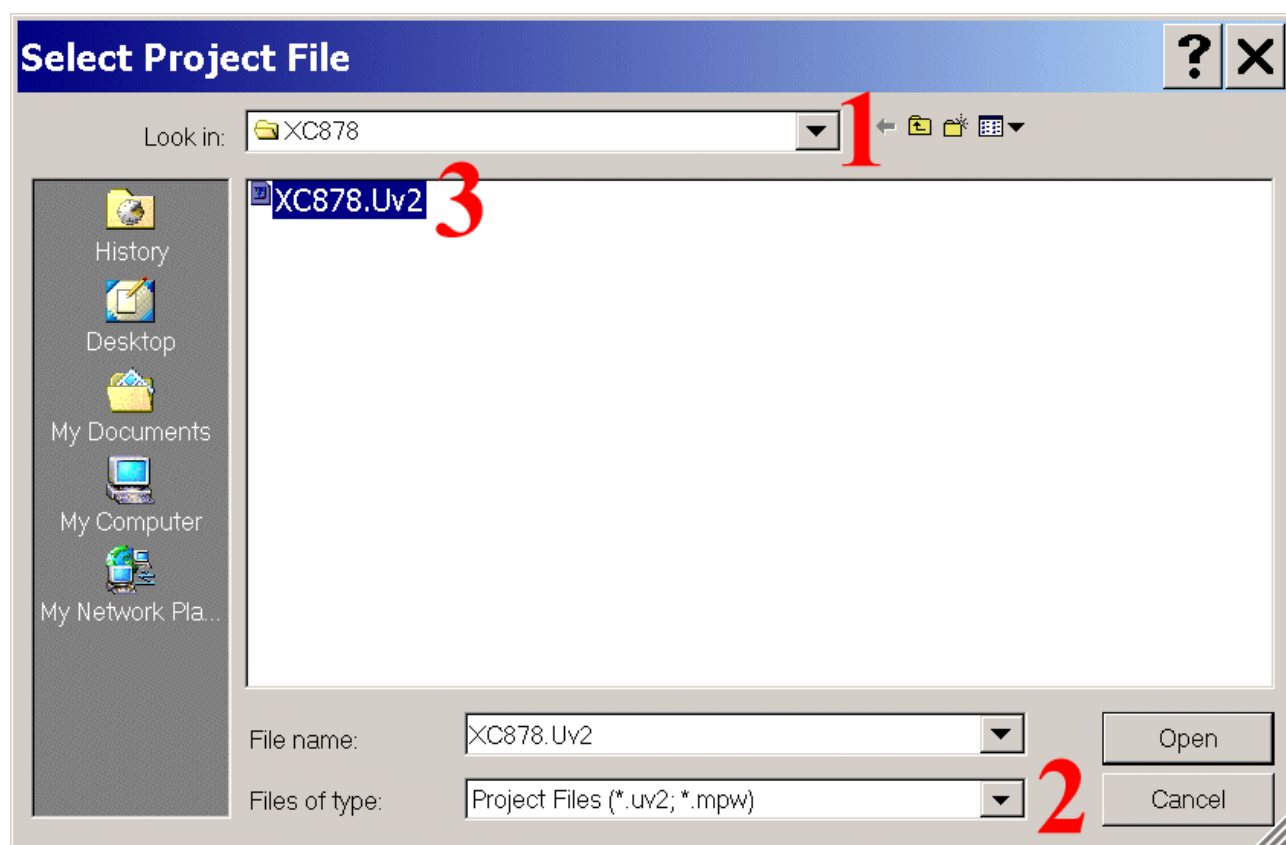
Project - Open Project

Select Project File: **Look in:** choose C:\XC878 (1)

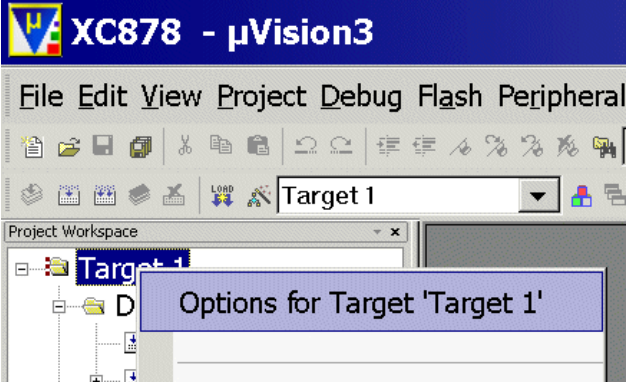
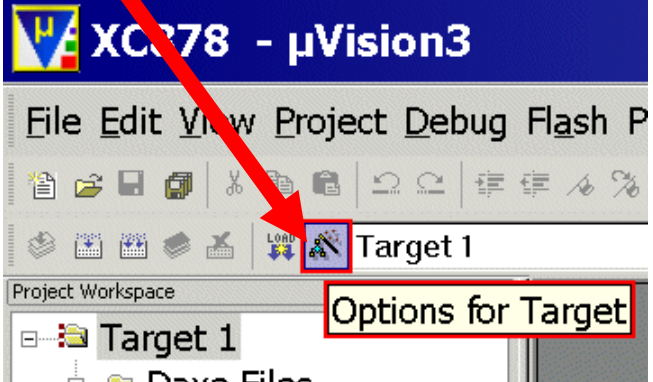
Select Project File: **Files of type:** select Project Files (*.uv2) (2)

Click XC878.Uv2 (3)

Click Open



Check/configure the configuration of the Flash Programming Utility :

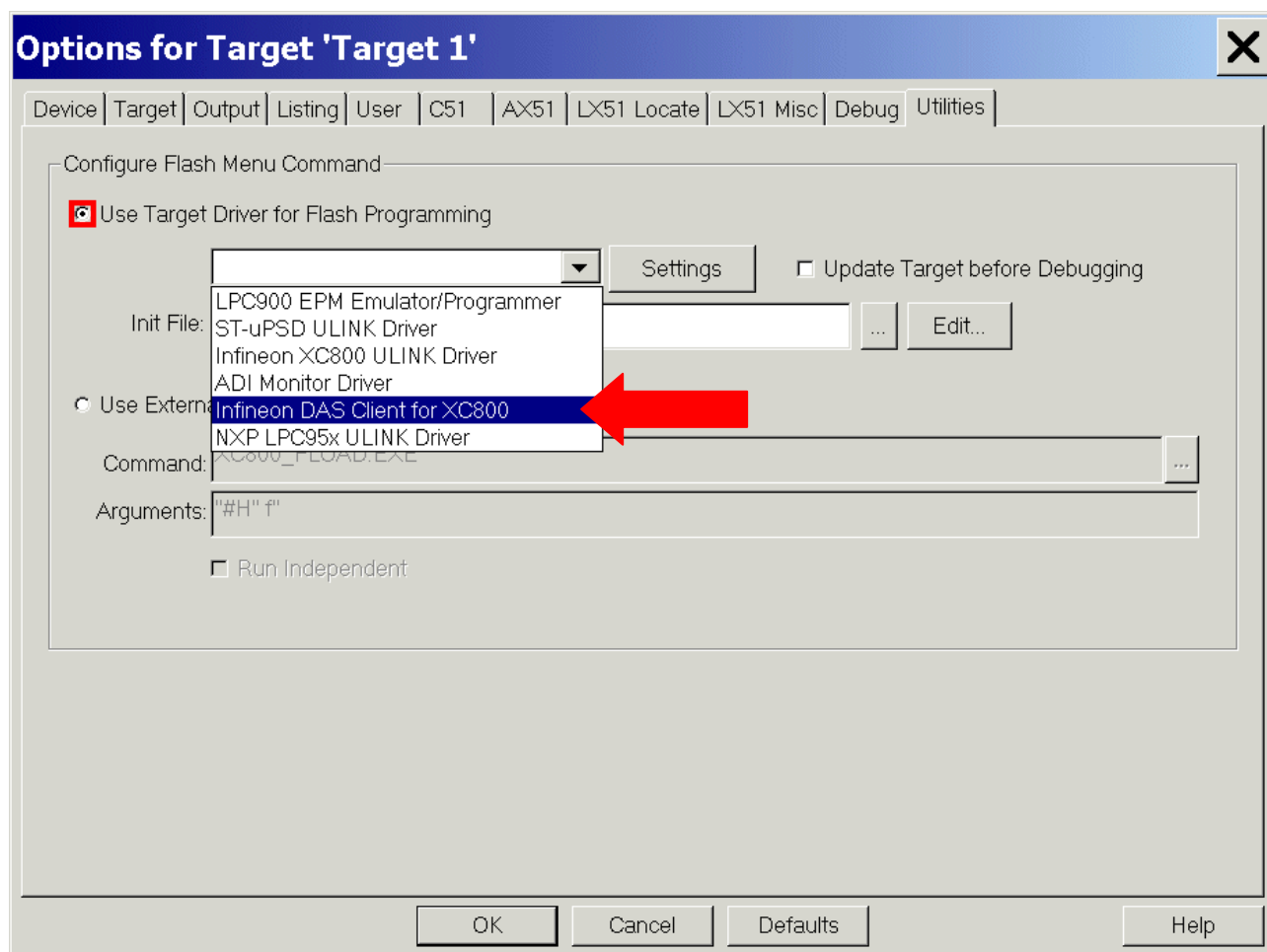
<p>mouse position: (Project Workspace, Files): Target1 click right mouse button Options for Target 'Target1'</p>	or	<p>click</p>
		

Options for Target 'Target1': Utilities:

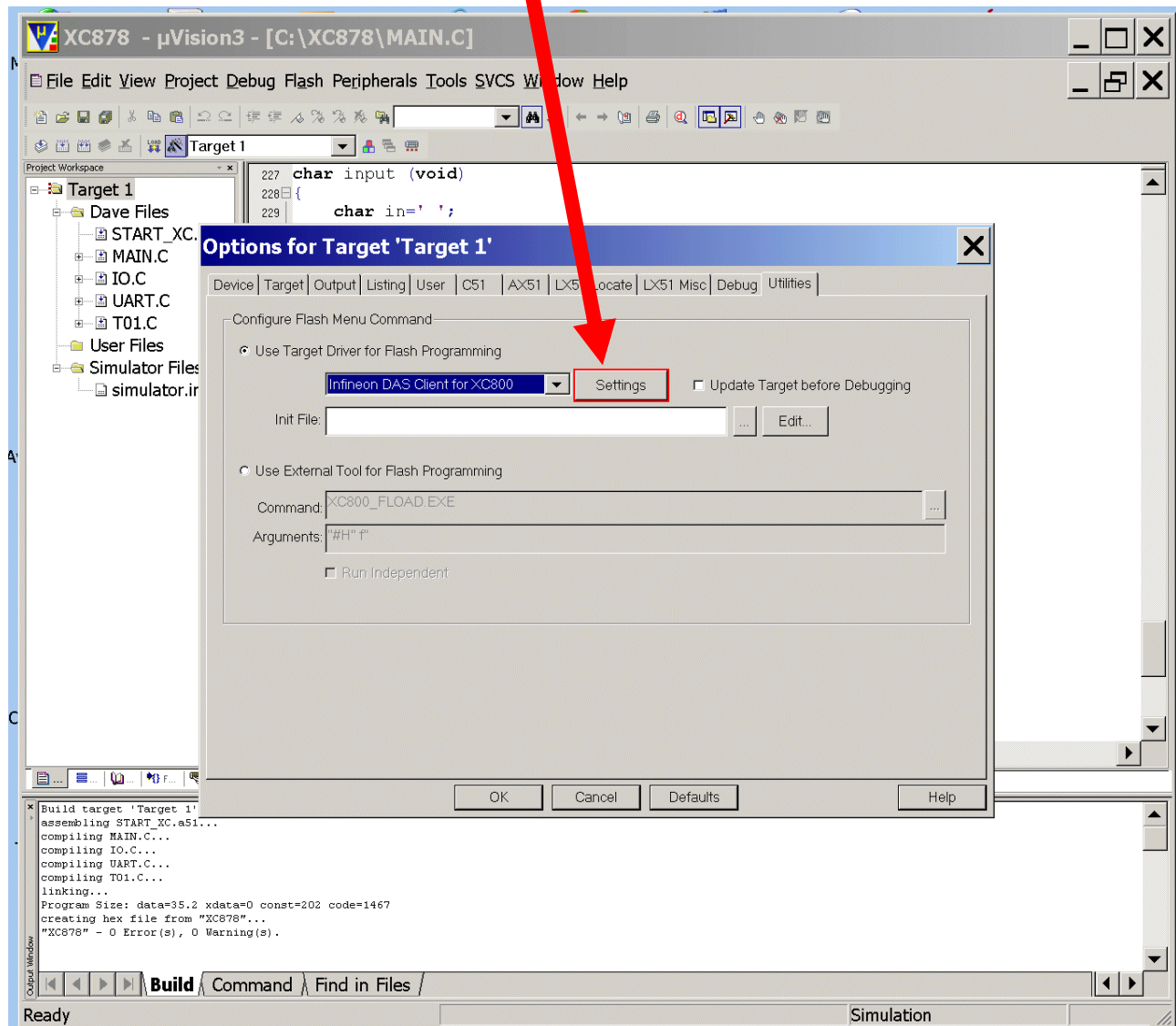
Configure Flash Menu Command: **check/click** ☒ Use Target Driver for Flash Programming

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 DAS Driver Setup:

DAS Client Setup: **DAS Server:** select JTAG over USB Chip

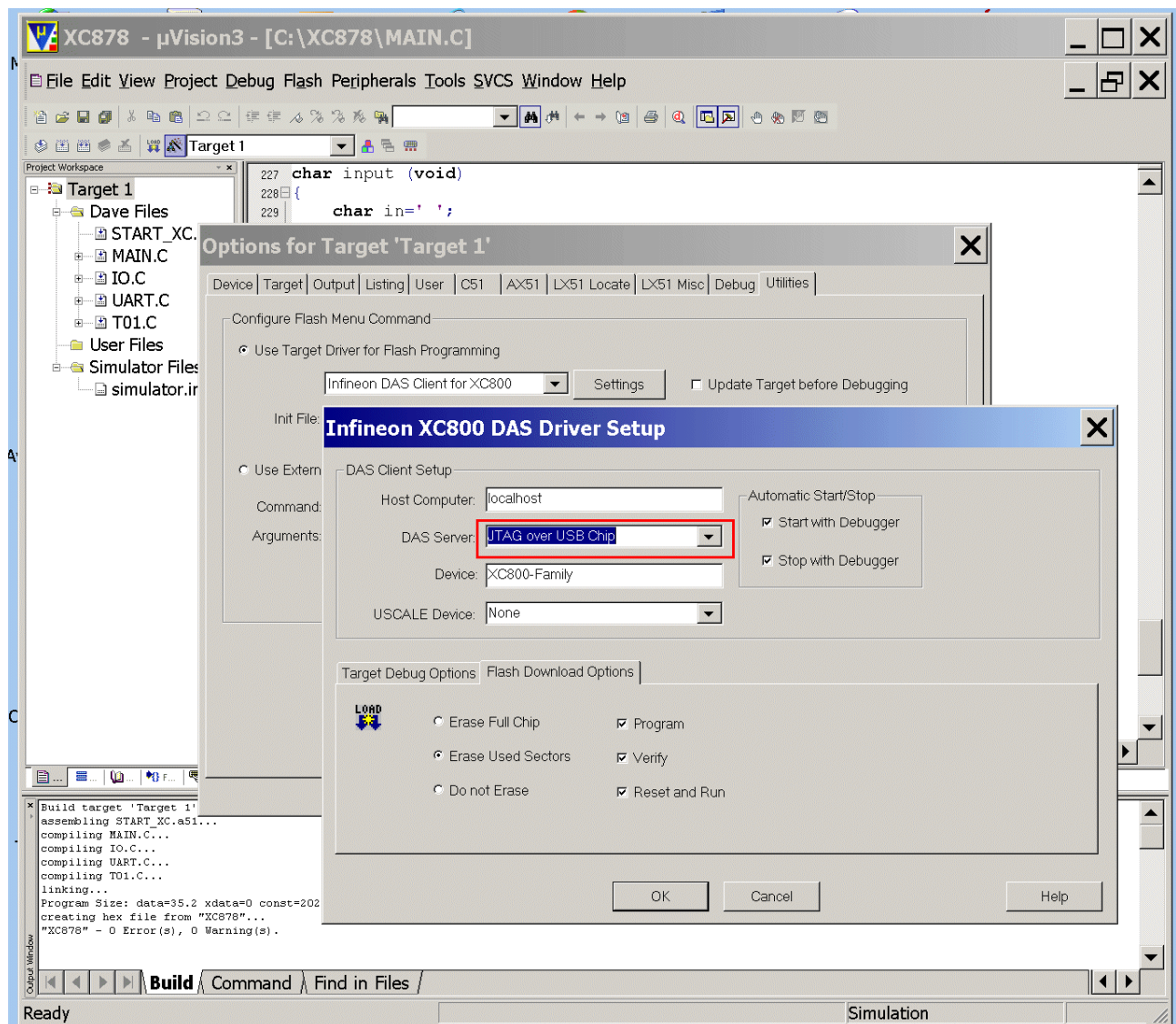
DAS Client Setup: **USCALE Device:** select/check XC800-Family

Flash Download Options: **check:** ☐ Erase Used Sectors

Flash Download Options: **check/tick:** ☒ Program

Flash Download Options: **check/tick:** ☒ Verify

Flash Download Options: **check/tick:** ☒ Reset and Run



Click OK

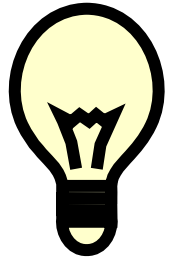
Click OK

Install U-SPY:

Note:

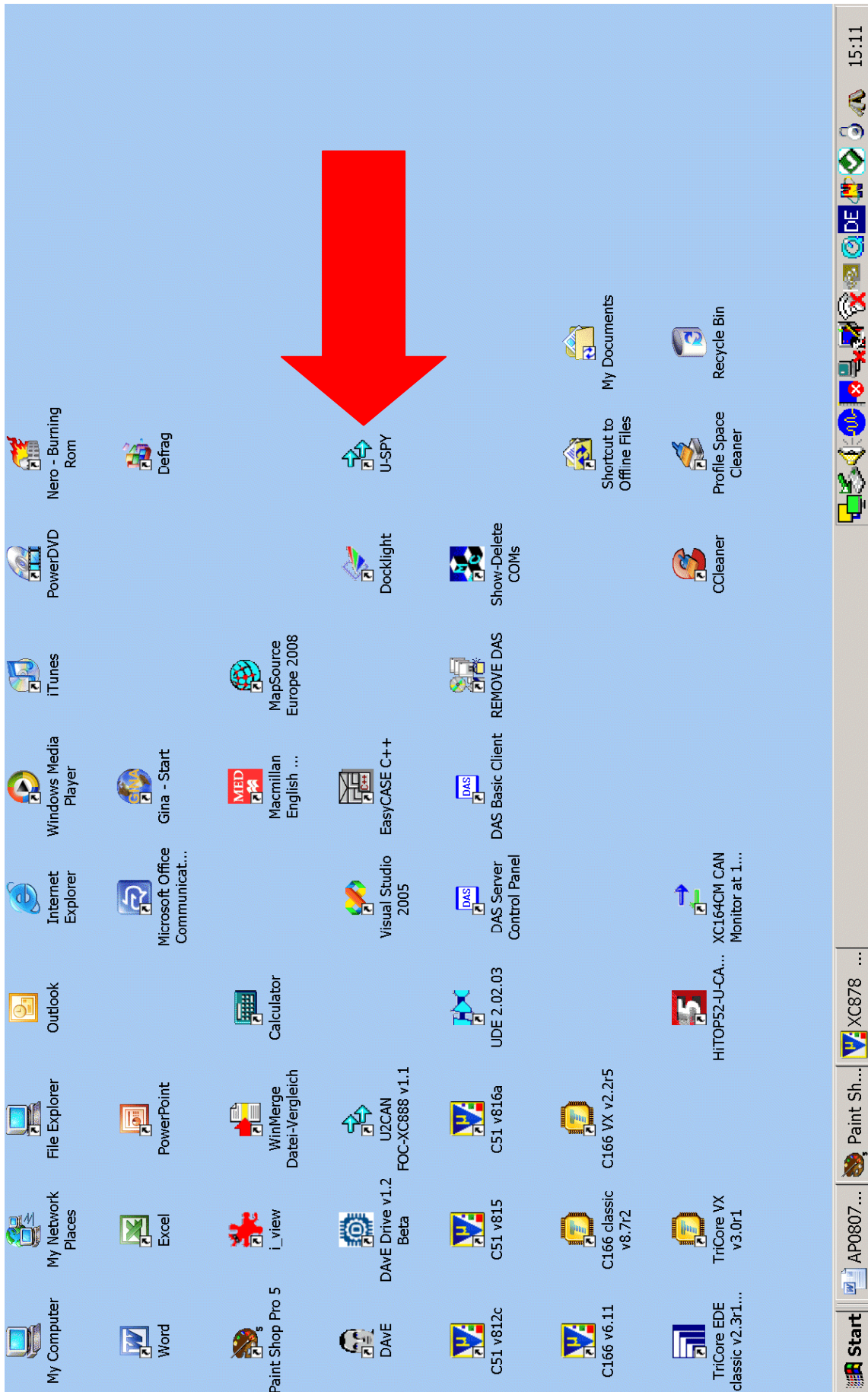
Now we need a terminal program which is able to handle our **virtual COM port** (COM7)!
As an example of "any terminal program" we are going to use U-SPY.

U-SPY can be found either on the XC878 Easy Kit CD (USPY_install.exe)



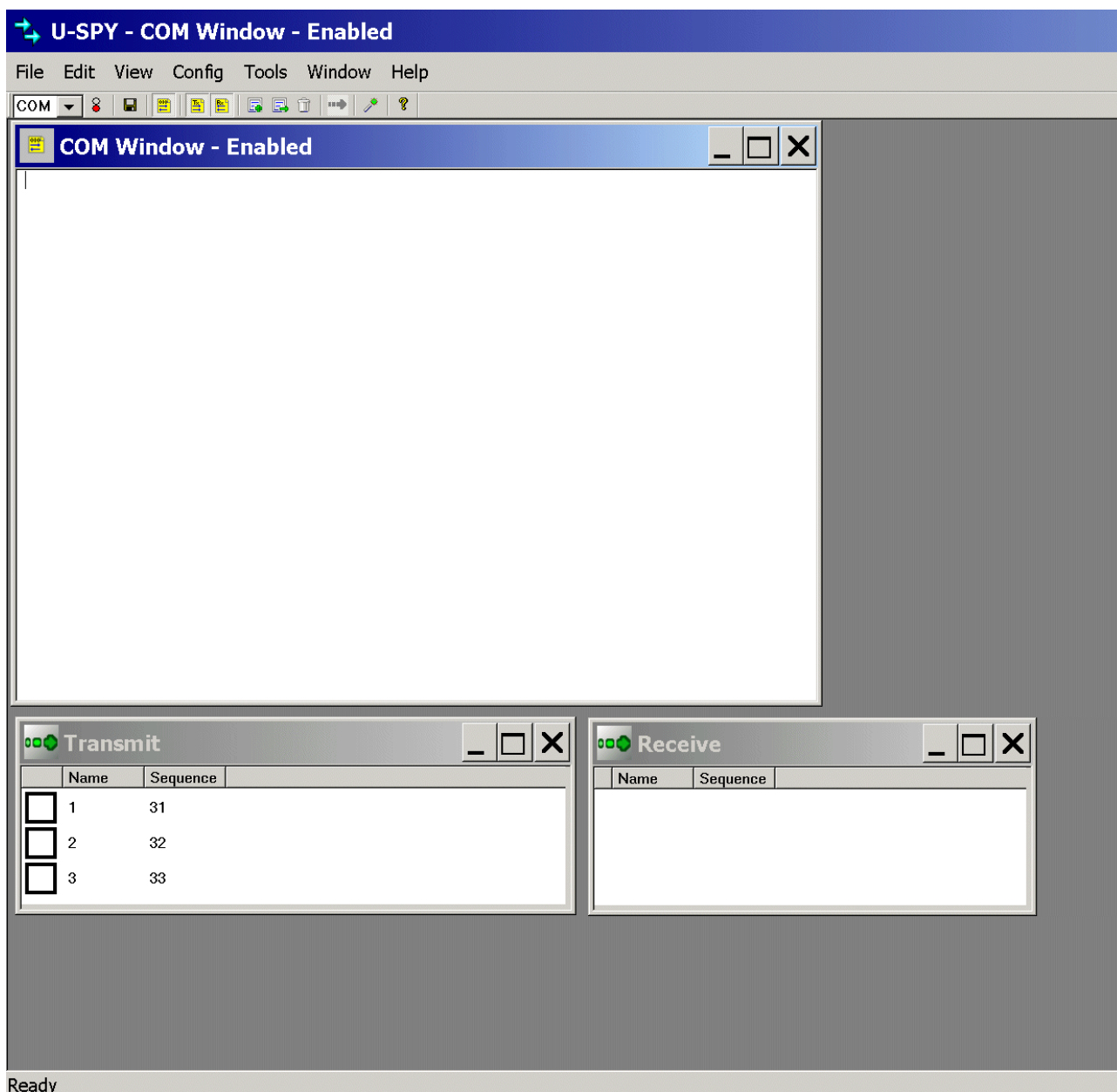
or on your microcontroller DVD.

To install U-SPY **execute** USPY_install.exe

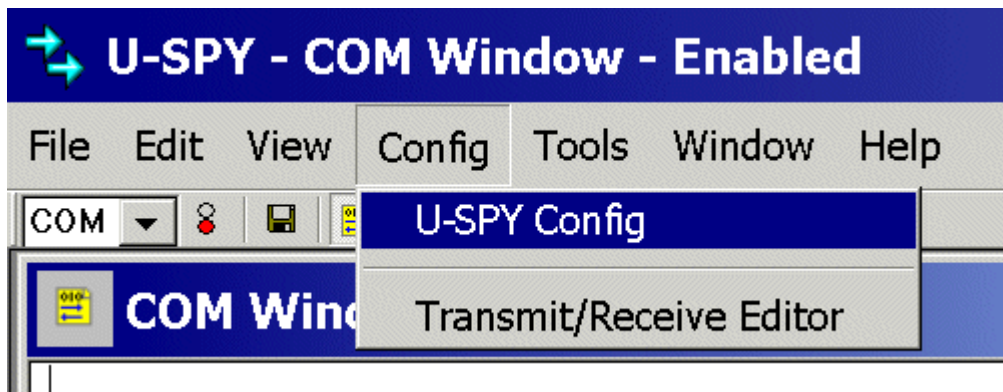




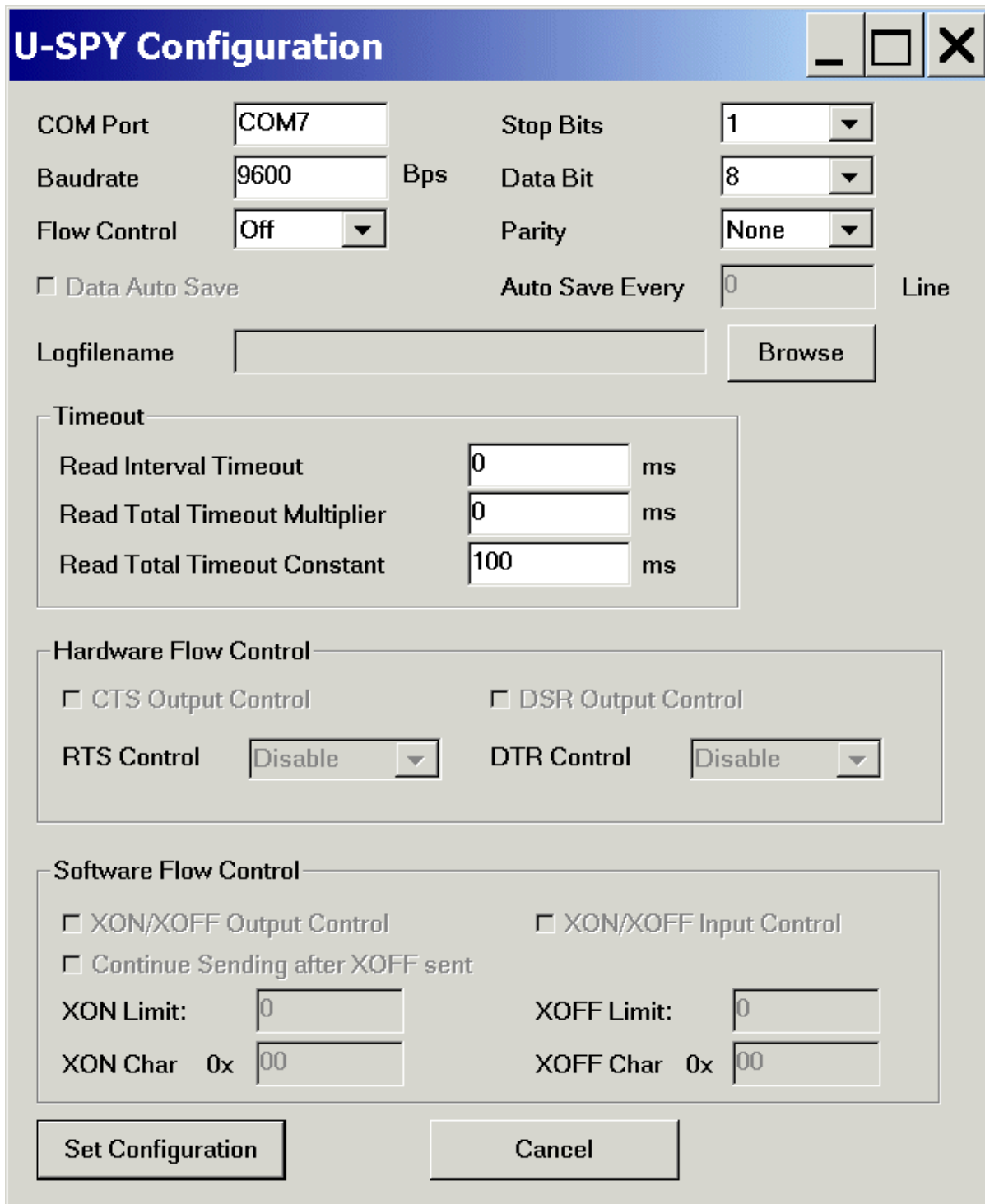
Now, **start** U-SPY:



Config – U-SPY Config



COM Port **check/insert** COM7
Baudrate **check/insert** 9600
Flow Control **check/insert** Off
Stop Bits **check/insert** 1
Data Bit **check/insert** 8
Parity **check/insert** None



U-SPY Configuration

COM Port: Stop Bits:

Baudrate: Bps Data Bit:

Flow Control: Parity:

☐ Data Auto Save Auto Save Every: Line

Logfilename:

Timeout

Read Interval Timeout: ms

Read Total Timeout Multiplier: ms

Read Total Timeout Constant: ms

Hardware Flow Control

☐ CTS Output Control ☐ DSR Output Control

RTS Control: DTR Control:

Software Flow Control

☐ XON/XOFF Output Control ☐ XON/XOFF Input Control

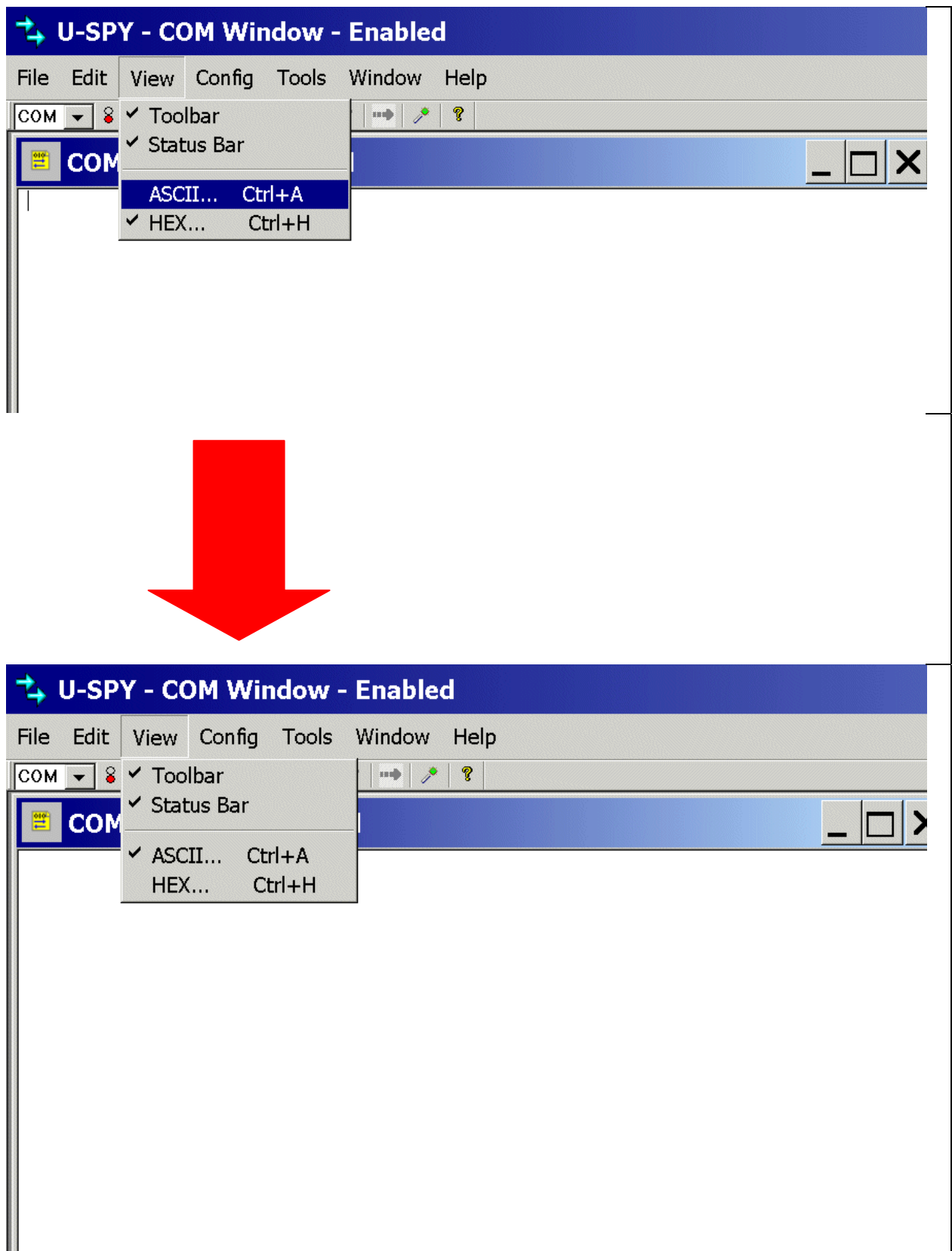
☐ Continue Sending after XOFF sent

XON Limit: XOFF Limit:

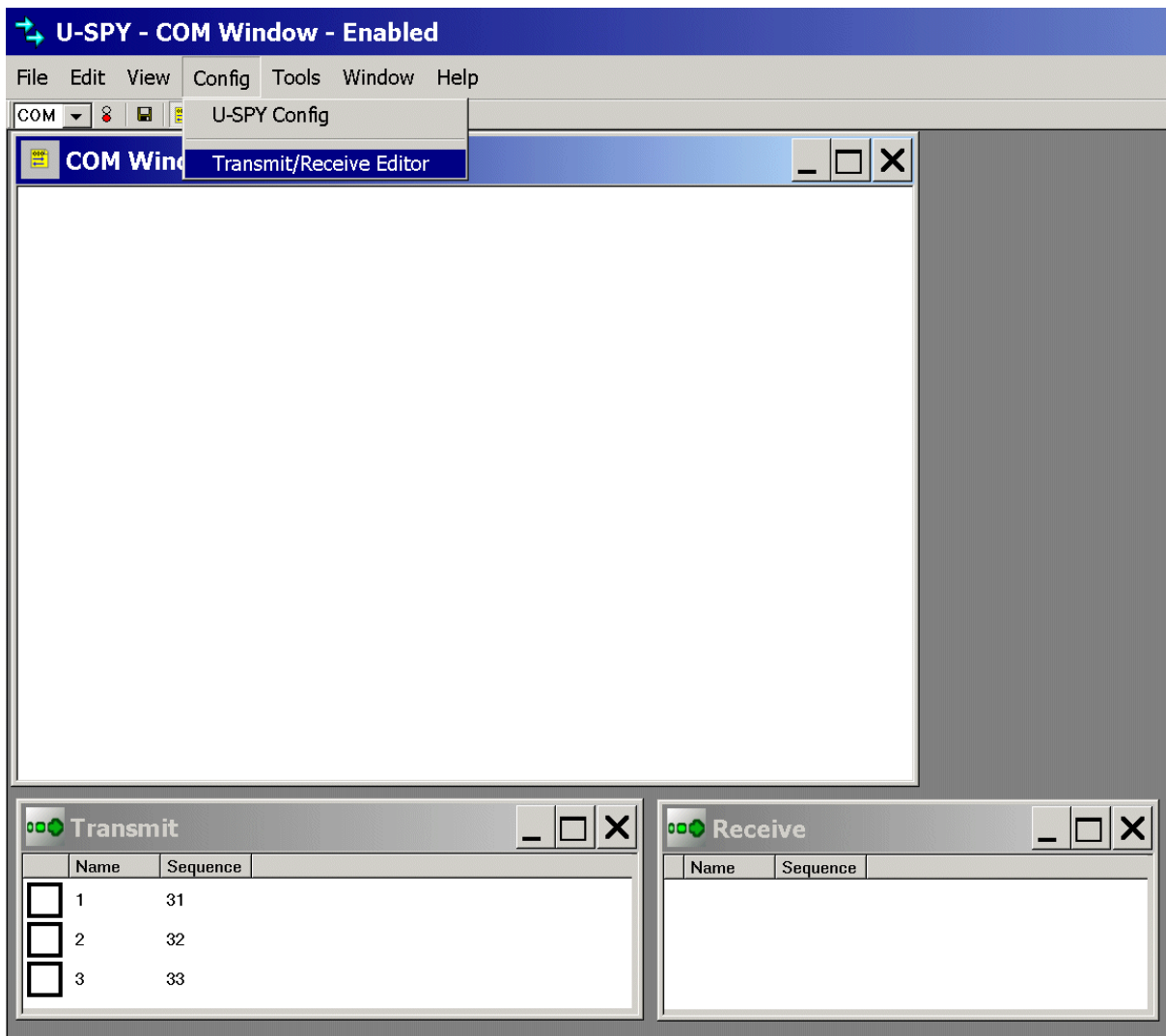
XON Char 0x: XOFF Char 0x:

Click Set Configuration

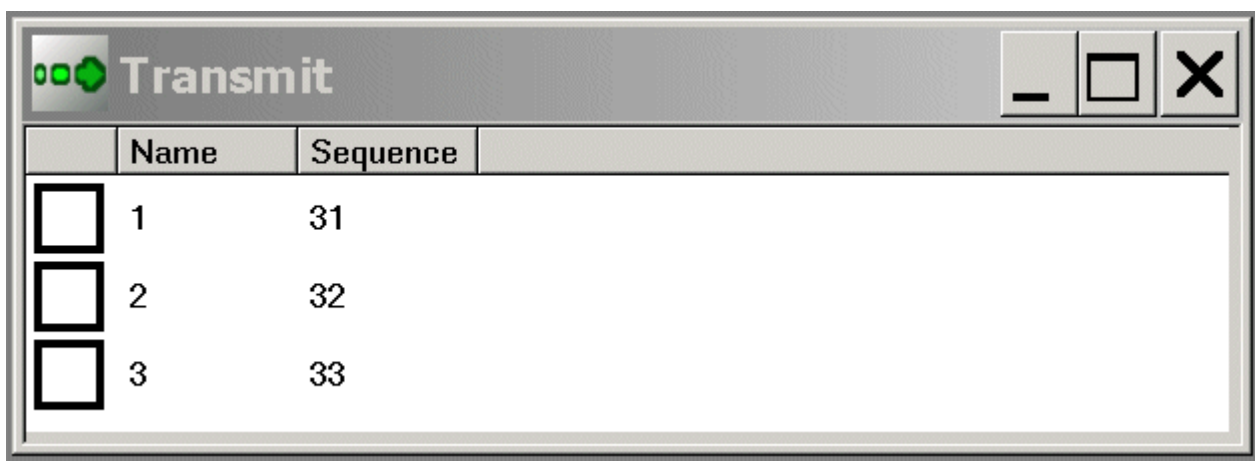
View – select ASCII...

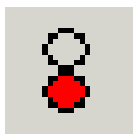


Config – Transmit/Receive Editor

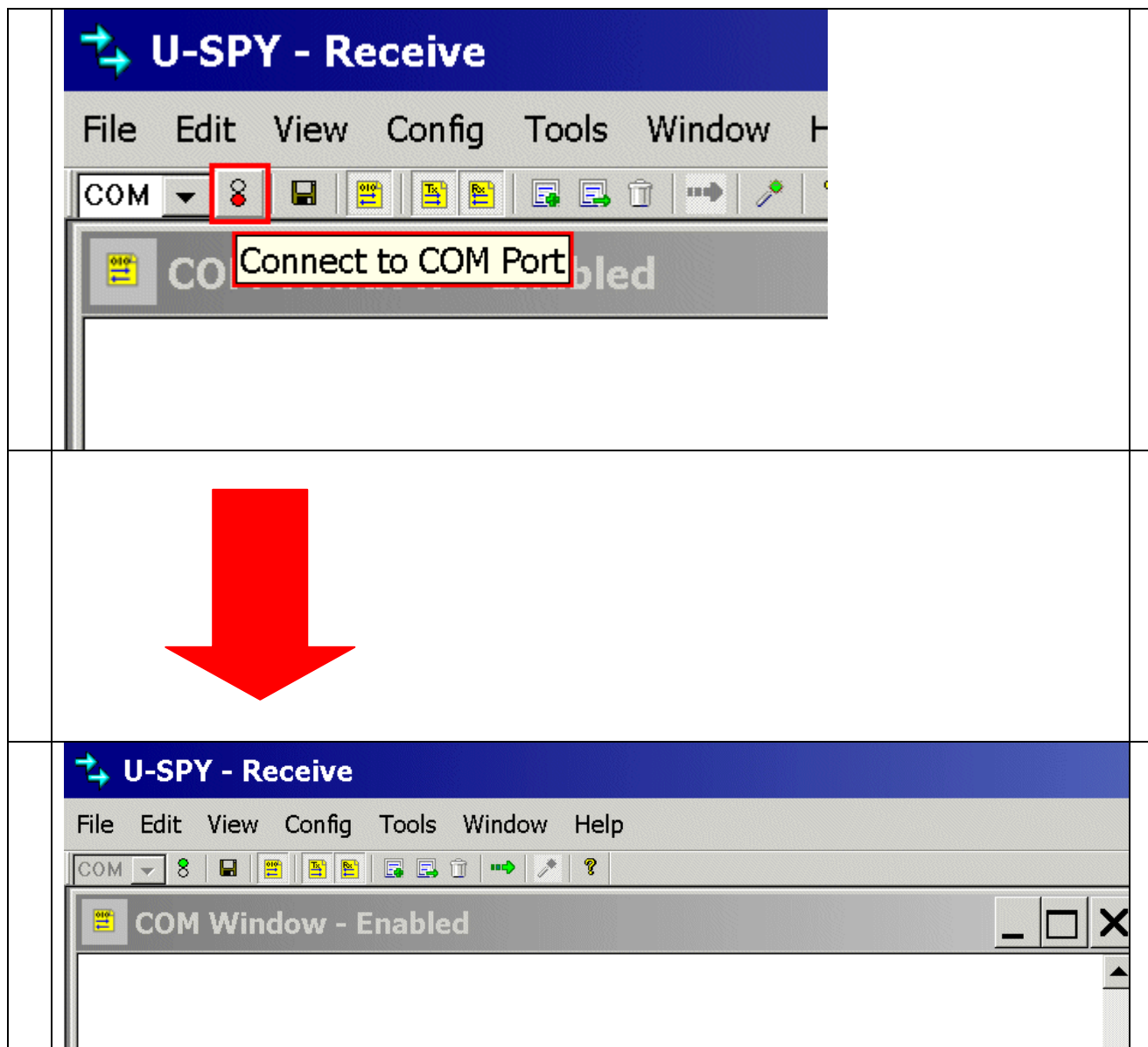


With the [Transmit/Receive Editor](#) **configure** the Transmit Window so that it appears like the Transmit window below:





Click





Note:

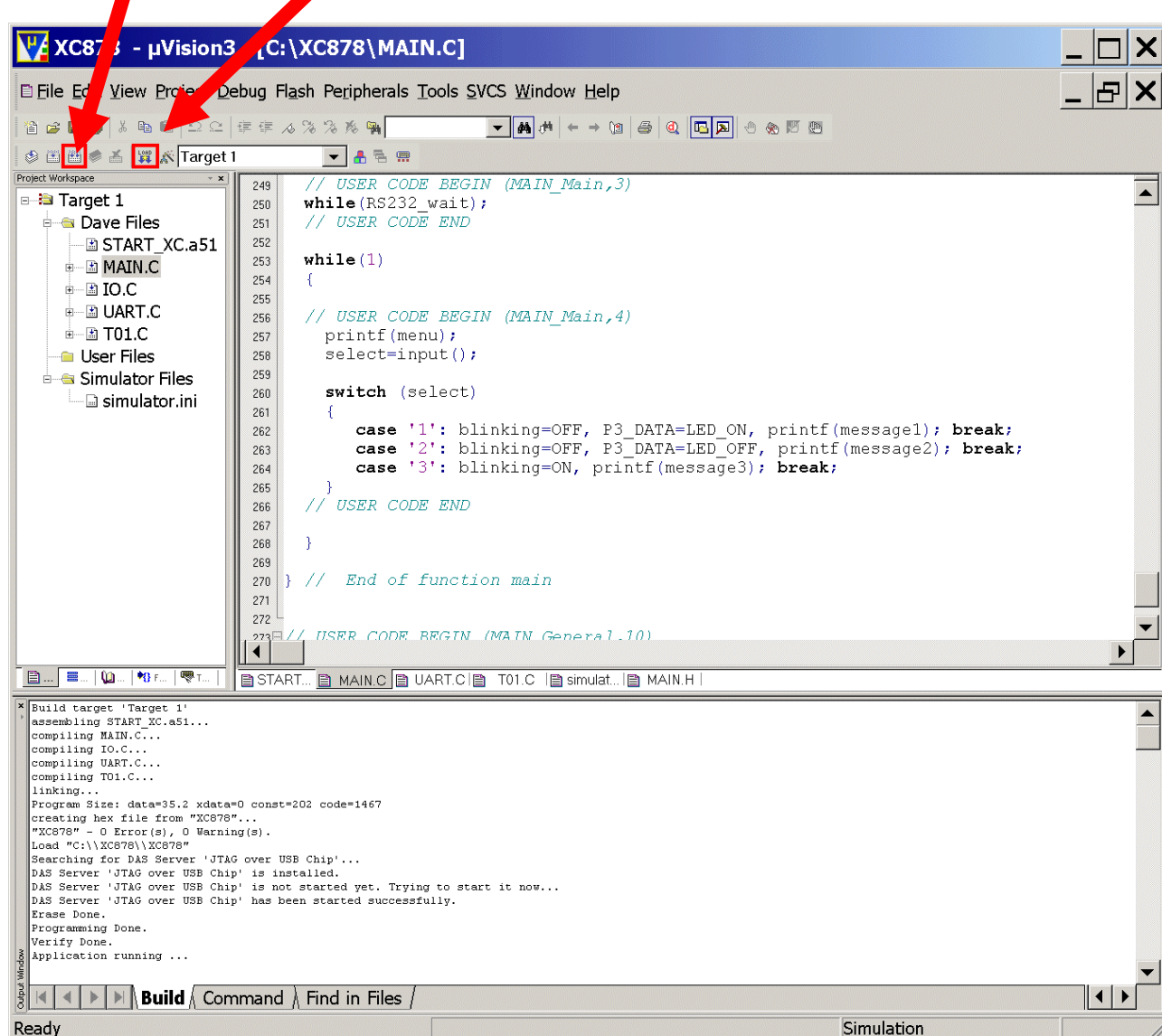
U-SPY is now ready for serial communication!





Go back to µVision:

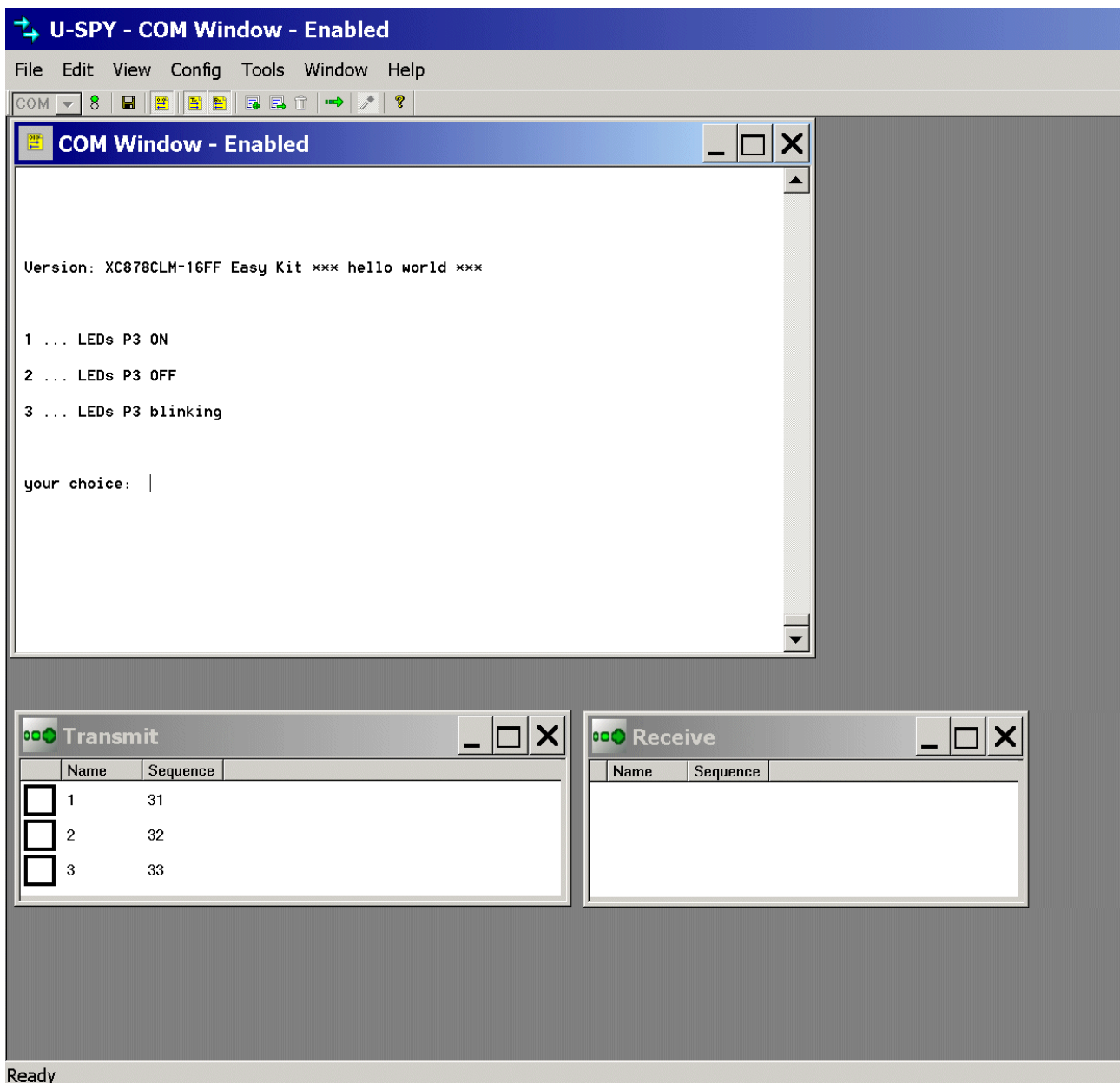
1.) click:  2.) click: 



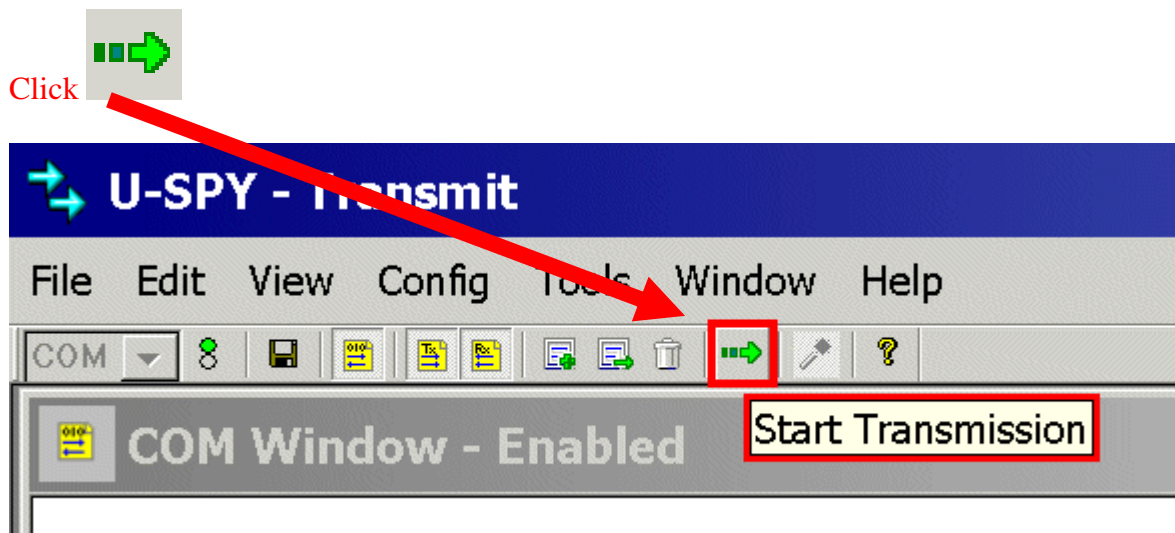
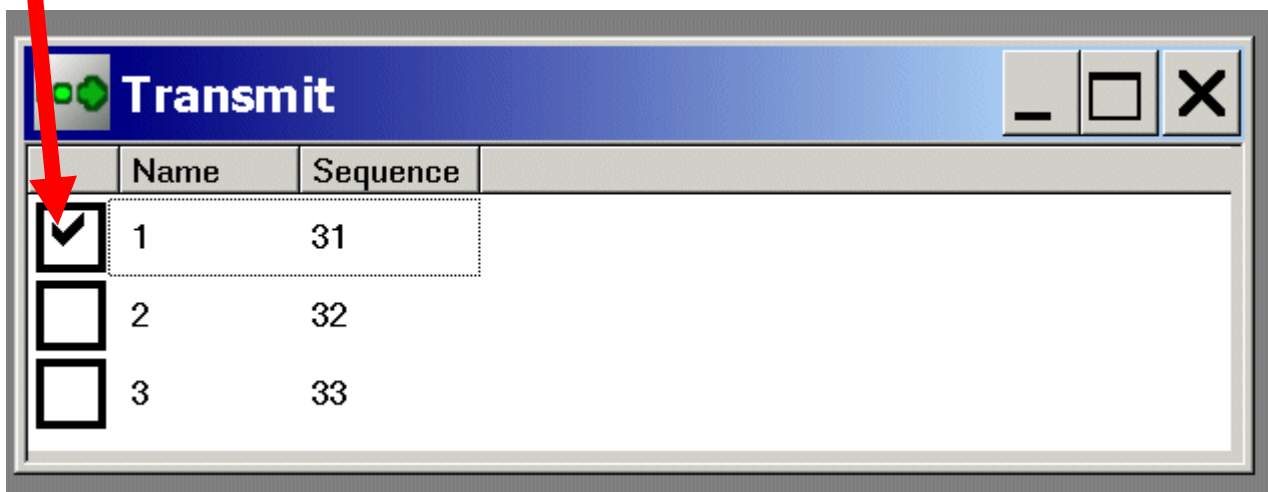
The screenshot shows the µVision3 IDE interface. The top menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, editing, and debugging. The Project Workspace on the left shows a tree structure with 'Target 1' expanded, containing 'Dave Files' (START_XC.a51, MAIN.C, IO.C, UART.C, T01.C) and 'User Files' (simulator.ini). The Source Code Editor displays the contents of MAIN.C, which includes user code for RS232 communication and a menu-driven program. The Output Window at the bottom shows the build process for 'Target 1', including assembly, compilation, linking, and loading of the program. The status bar at the bottom indicates 'Ready' and 'Simulation'.



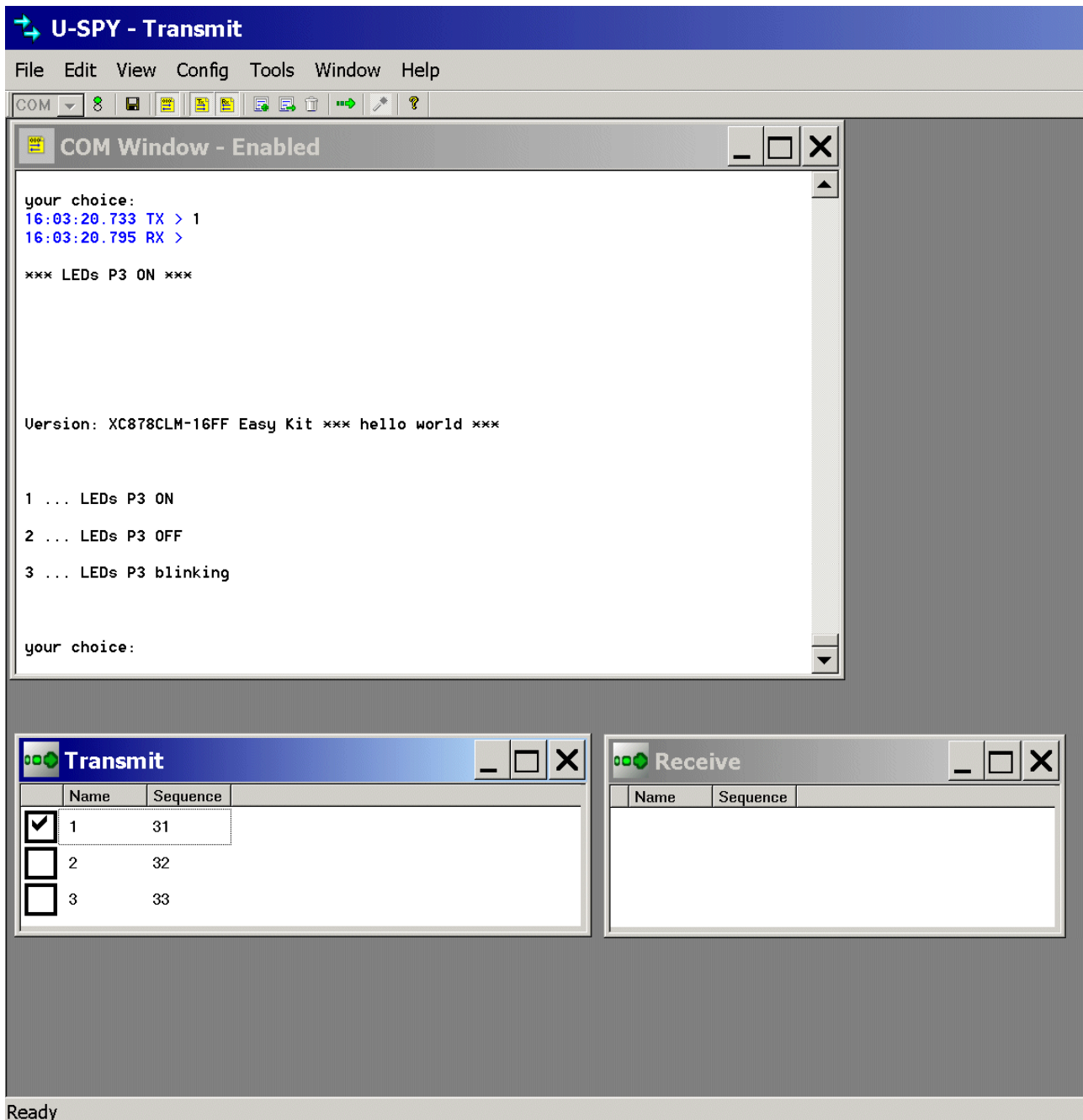
Go back to U-SPY and see the result:



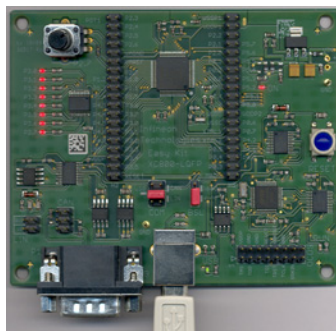
Tick ☒ 1:



See the result:



And also **check** the result on your XC878 Easy Kit:



**Conclusion:**

In this step-by-step book you have learned how to use the XC878 Easy 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 XC878 Easy 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 XC878 Easy 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.

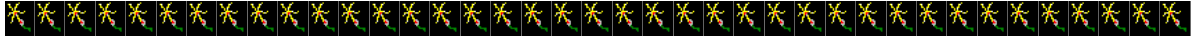
7.) Thanks To



Reinhard and Maureen for their support.



8.) Feedback (XC878 Easy 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>