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

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

Table of Contents **Page**

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.

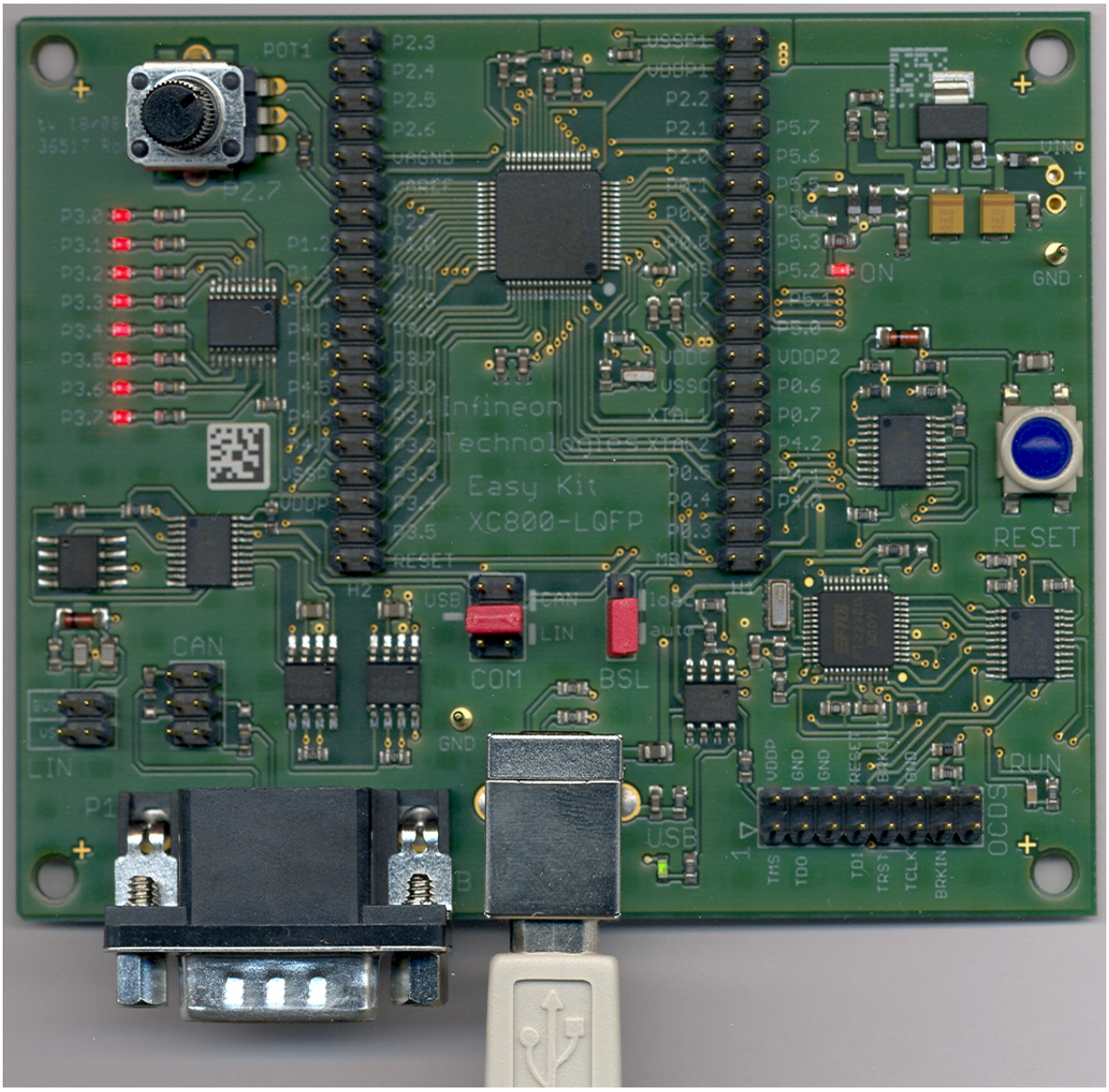
The purpose of this document is to gain know-how of the microcontroller and the tool-chain.
Additionally, the "hello world example" can easily be expanded to suit your needs.
You can connect either a part of - or your entire application to the XC878 Easy Kit.
You are also able to benchmark any of your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

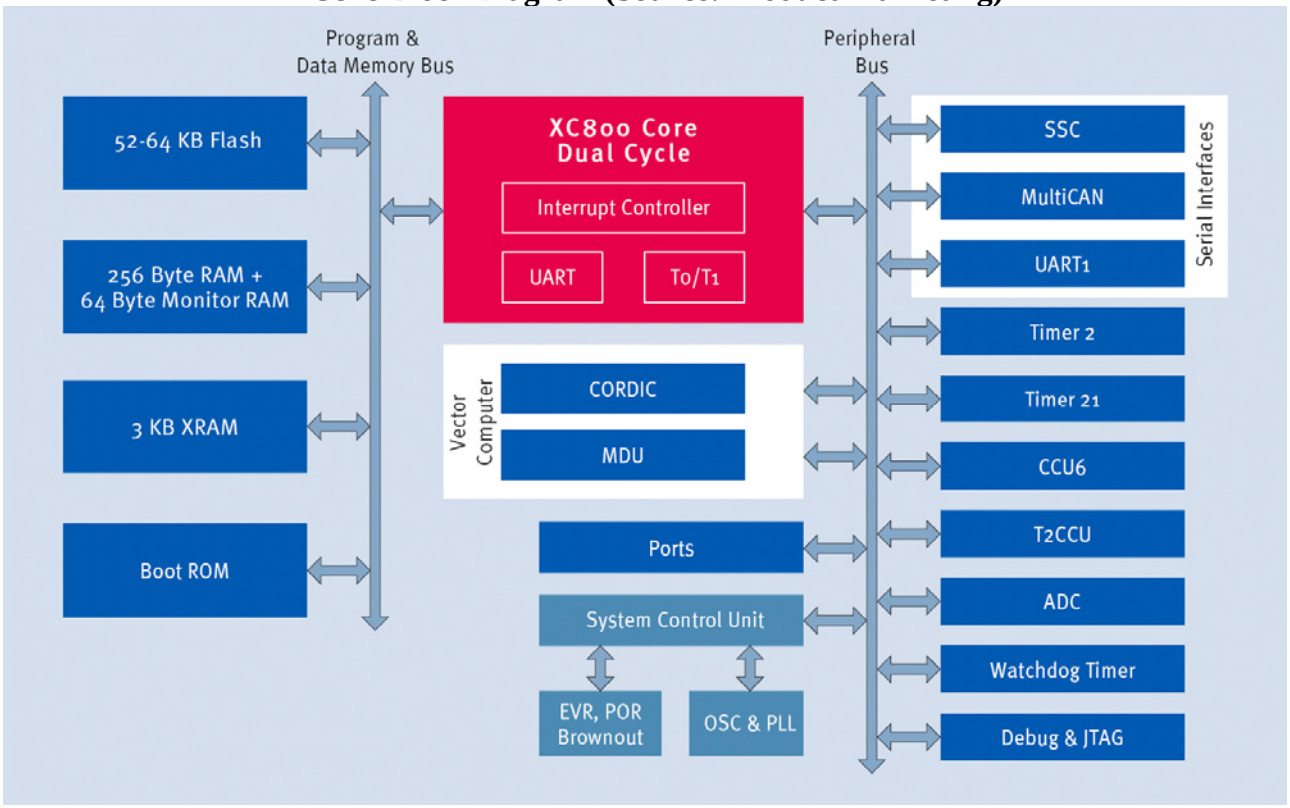
Programming Example

XC878 Easy Kit

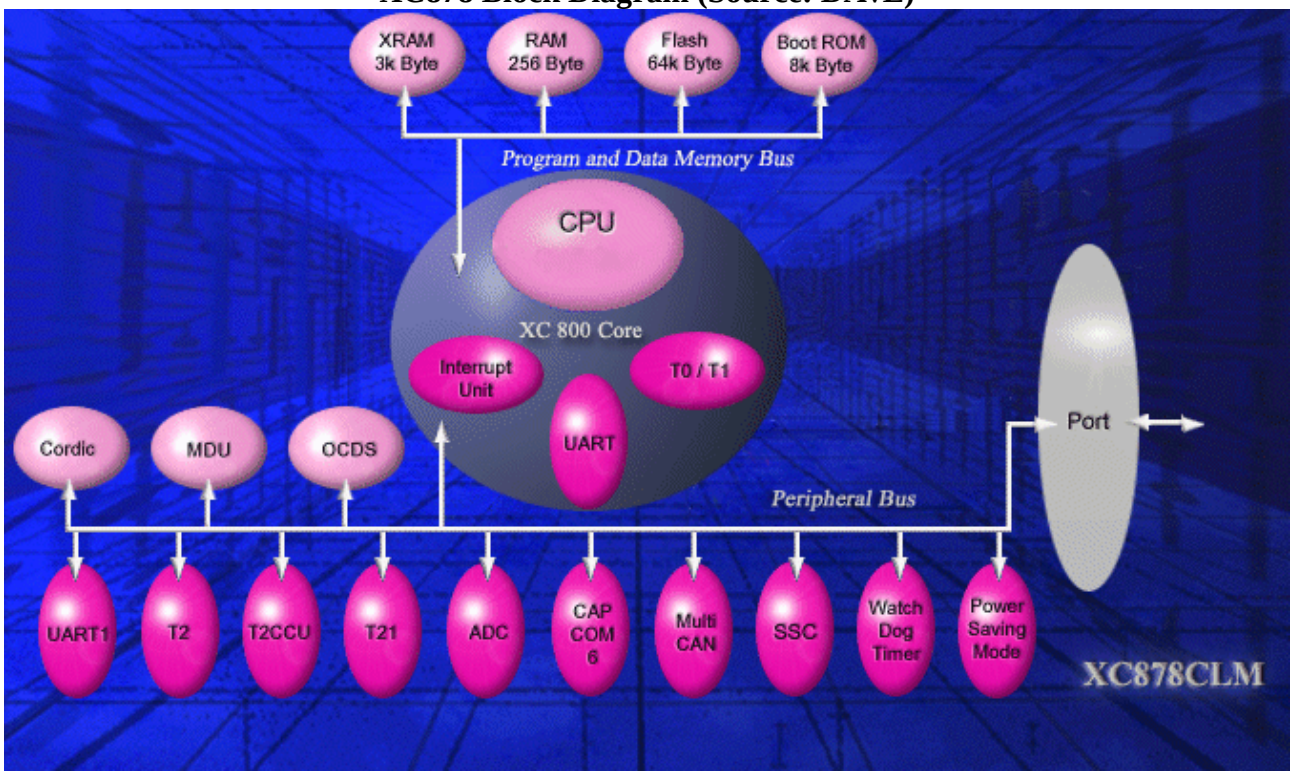


Used/selected microcontroller:

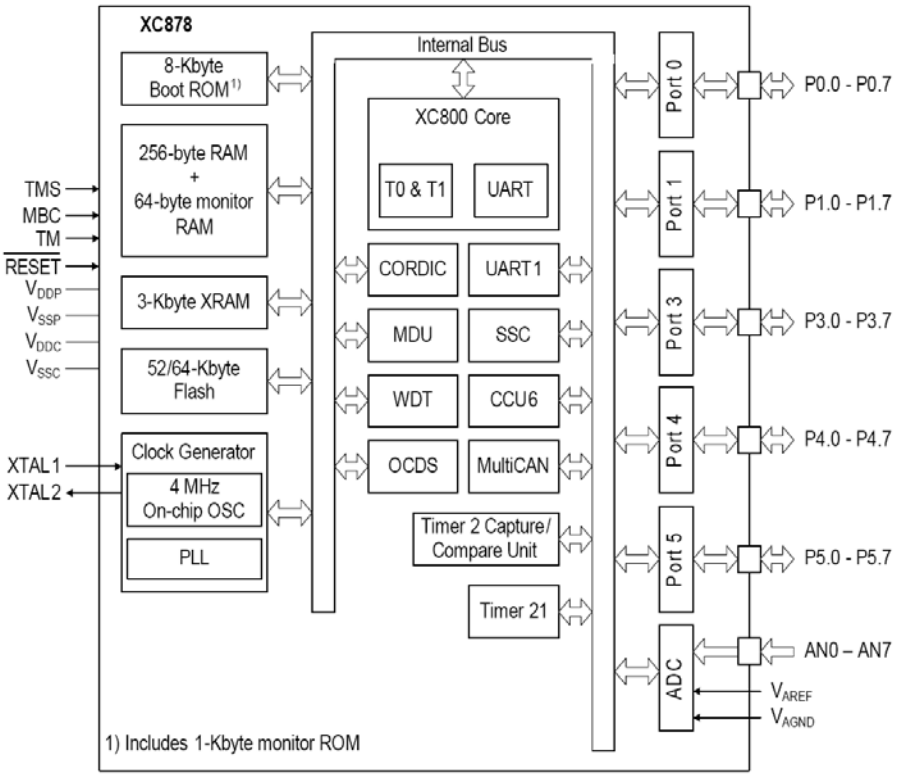
XC878 Block Diagram (Source: Product Marketing)



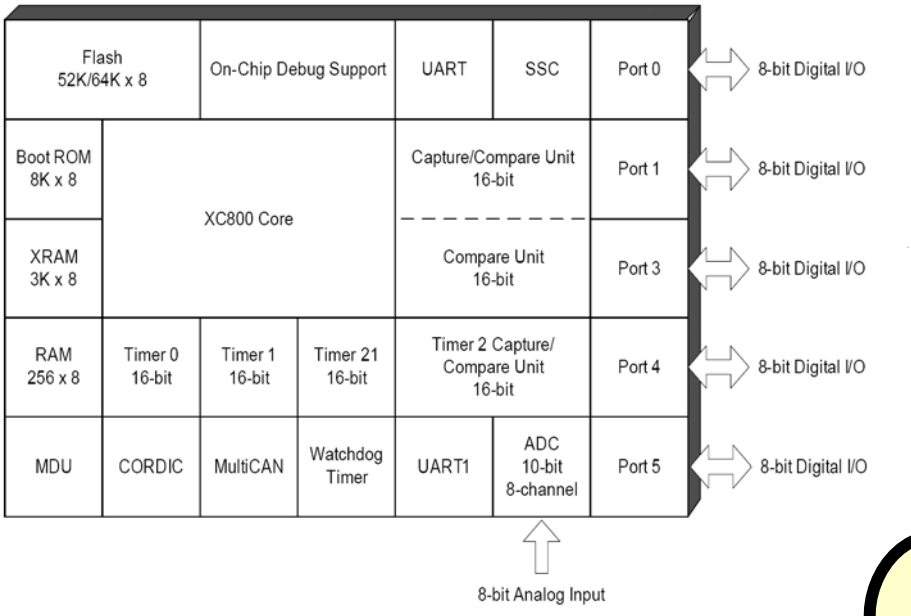
XC878 Block Diagram (Source: DAvE)



XC878 Block Diagram (Source: User's Manual)

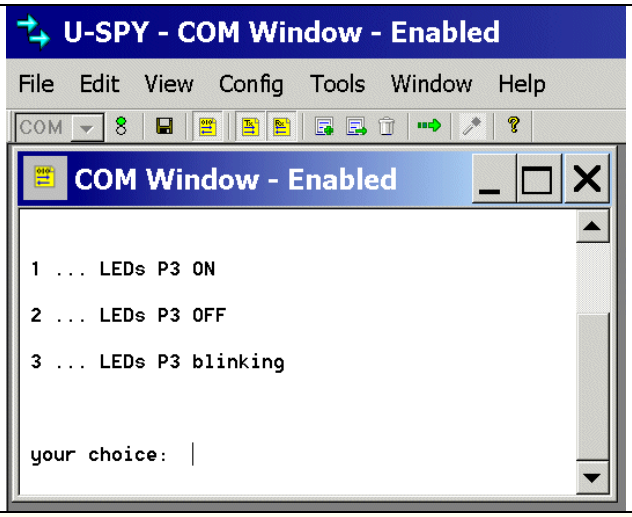


XC878 functional units (Source: User's Manual)



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.

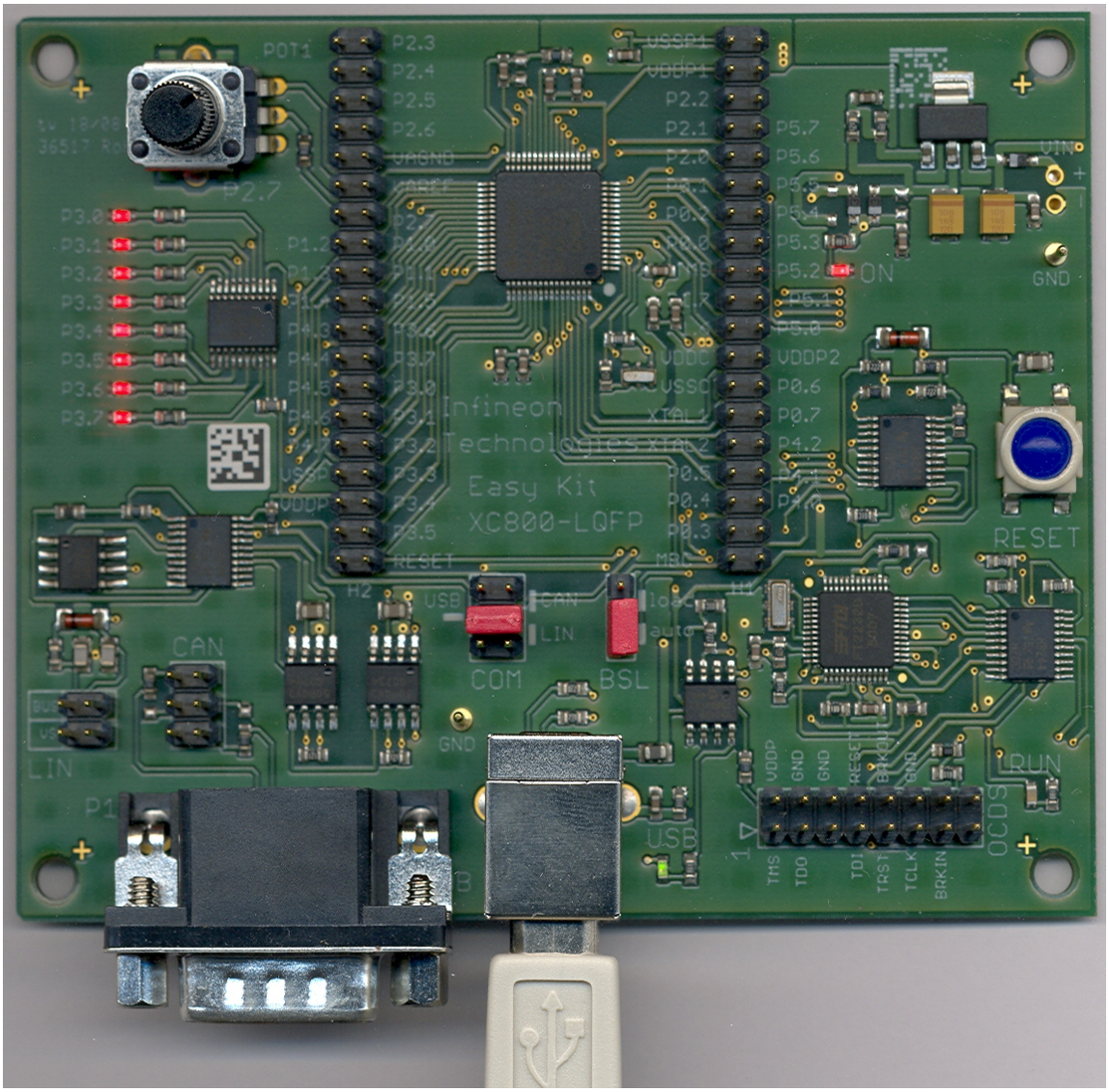
“Cookery book“
For your first programming example for the XC878 Easy Kit:

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

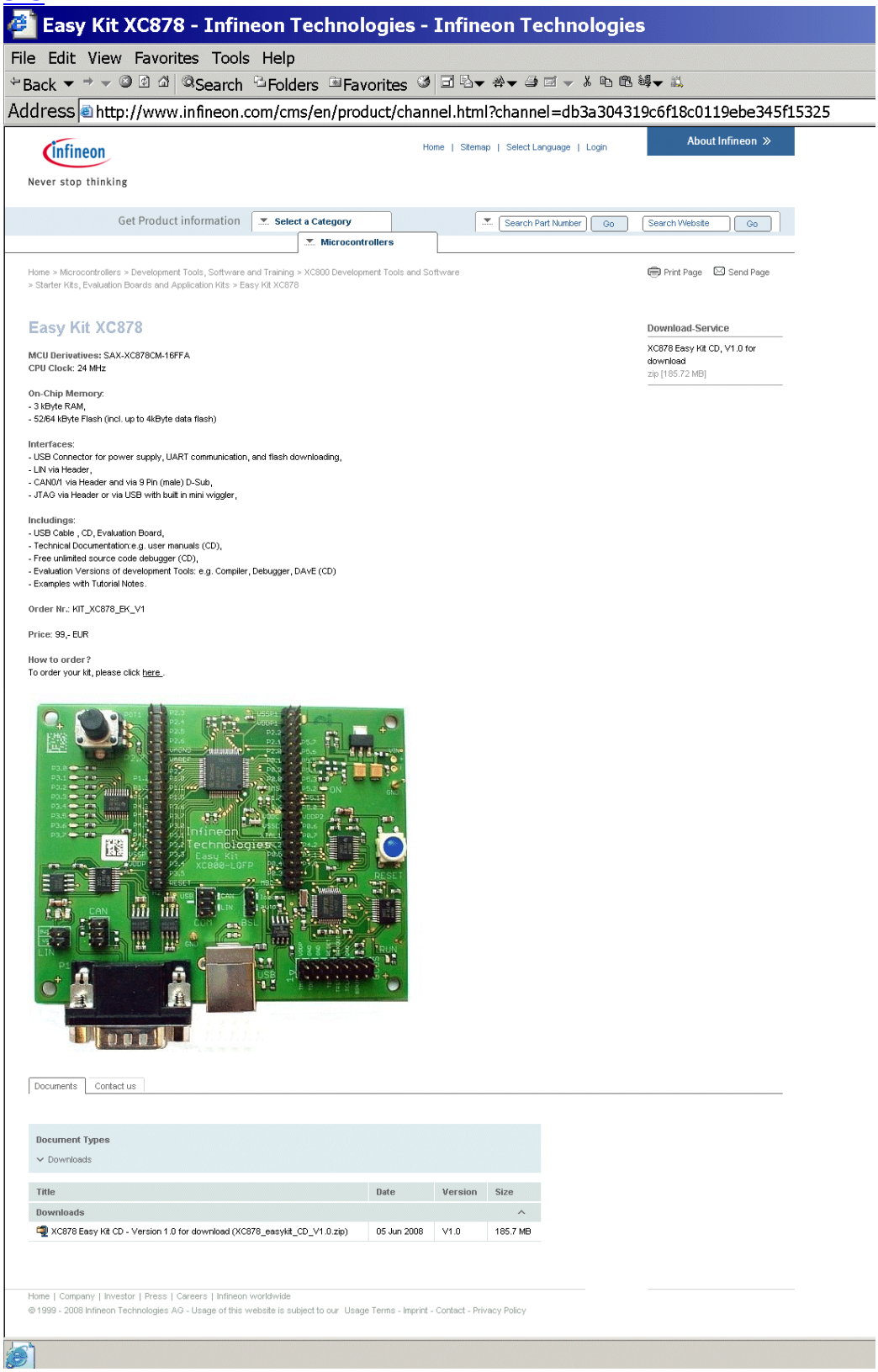
7.)	**Thanks To**
8.)	**Feedback**

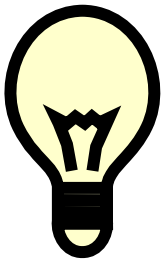
1.) DAS Installation + Connecting the XC878 Easy Kit:



Screenshot of the XC878 Easy Kit homepage:

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





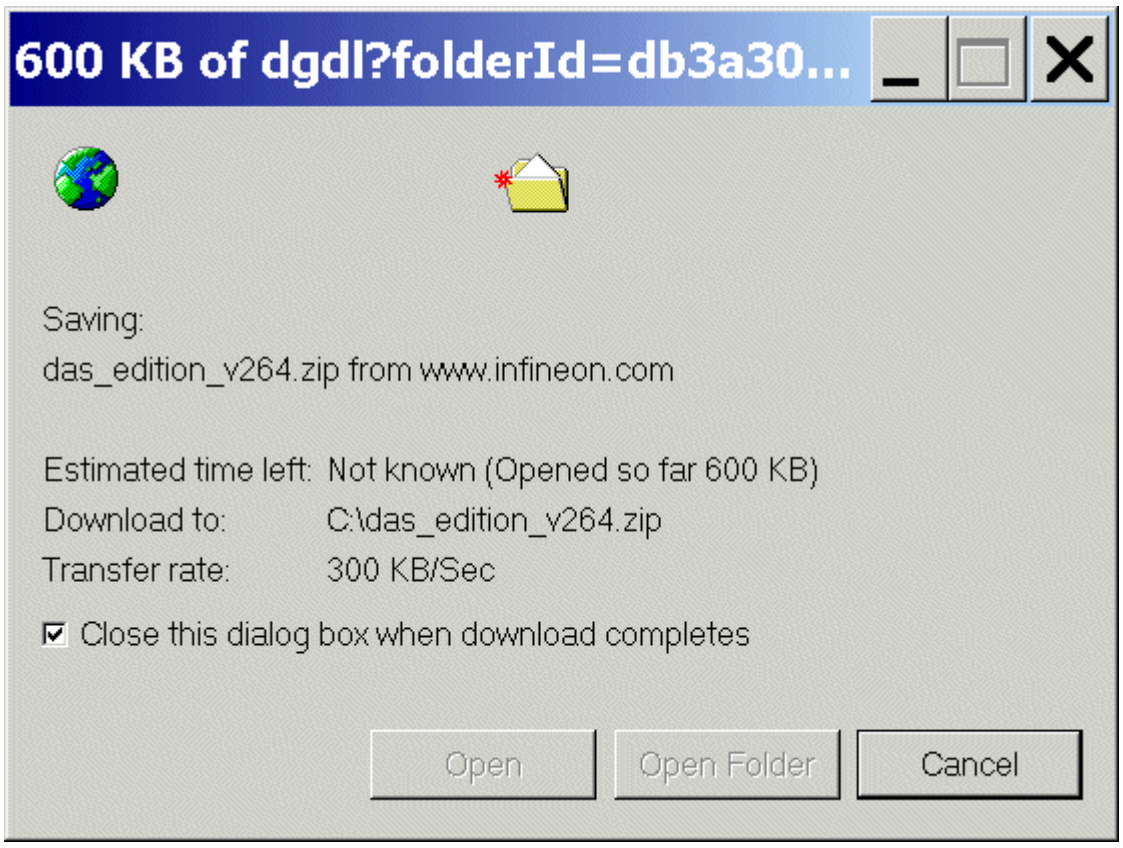
Note:
For further information, please refer to the XC878 Easy Kit Board Manual V1.0, April 2008.

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:

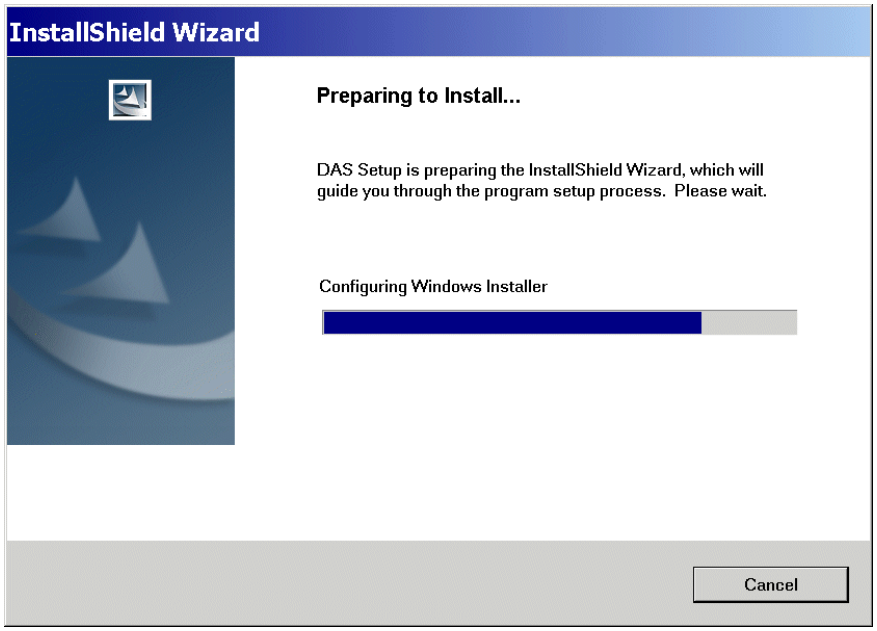
Note:
The DAS Server must be installed on your host computer!
The goal of the DAS software is to provide one single interface for all types of tools.
The USB Device driver communicates with the XC878 Easy Kit when connected to the host computer.
The USB Device driver for the XC878 Easy Kit USB interface is included in the DAS software.
A virtual COM port driver is also included.

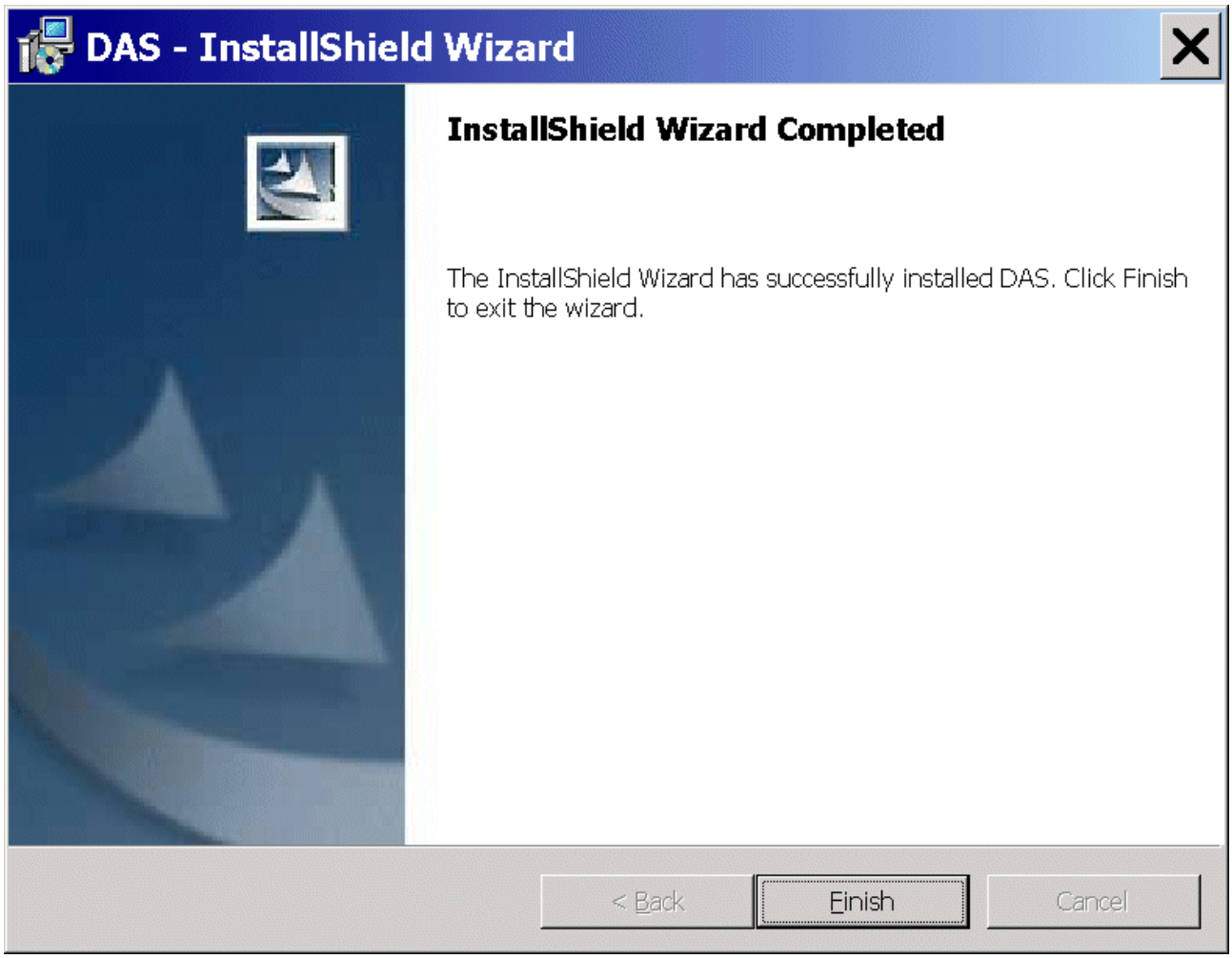
Download “The latest release version for download: DAS Edition v2.6.4”:



Unzip das_edition_v264.zip and

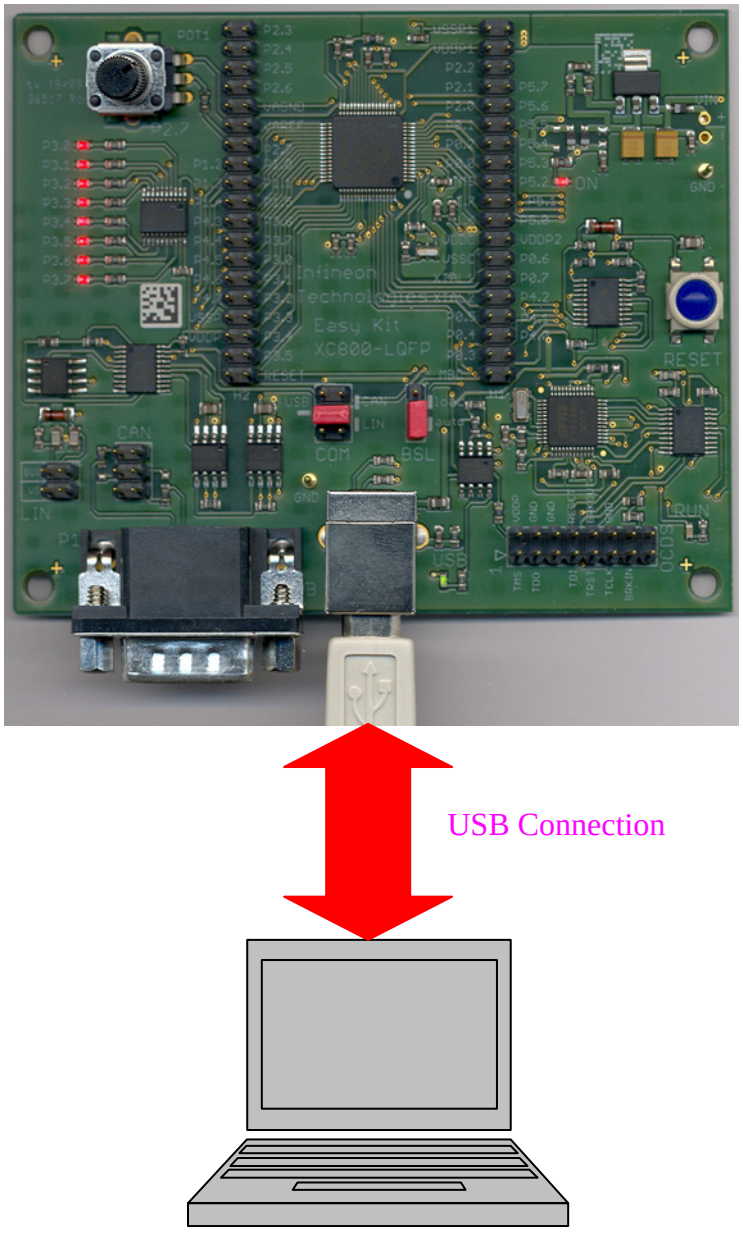
execute “DAS_v264_setup.exe” to install the DAS Server.



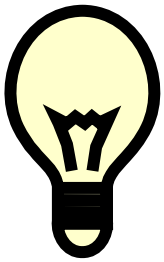


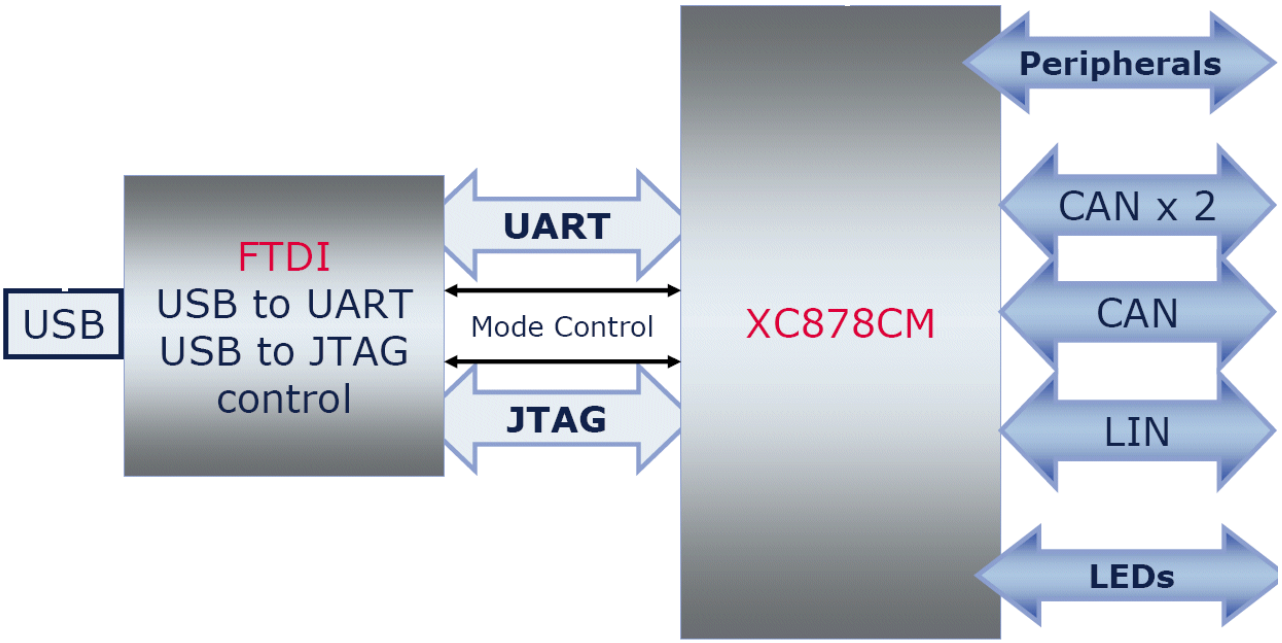
Click Finish

Connect the XC878 Easy Kit to the host computer:



USB Connection:
.) 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.



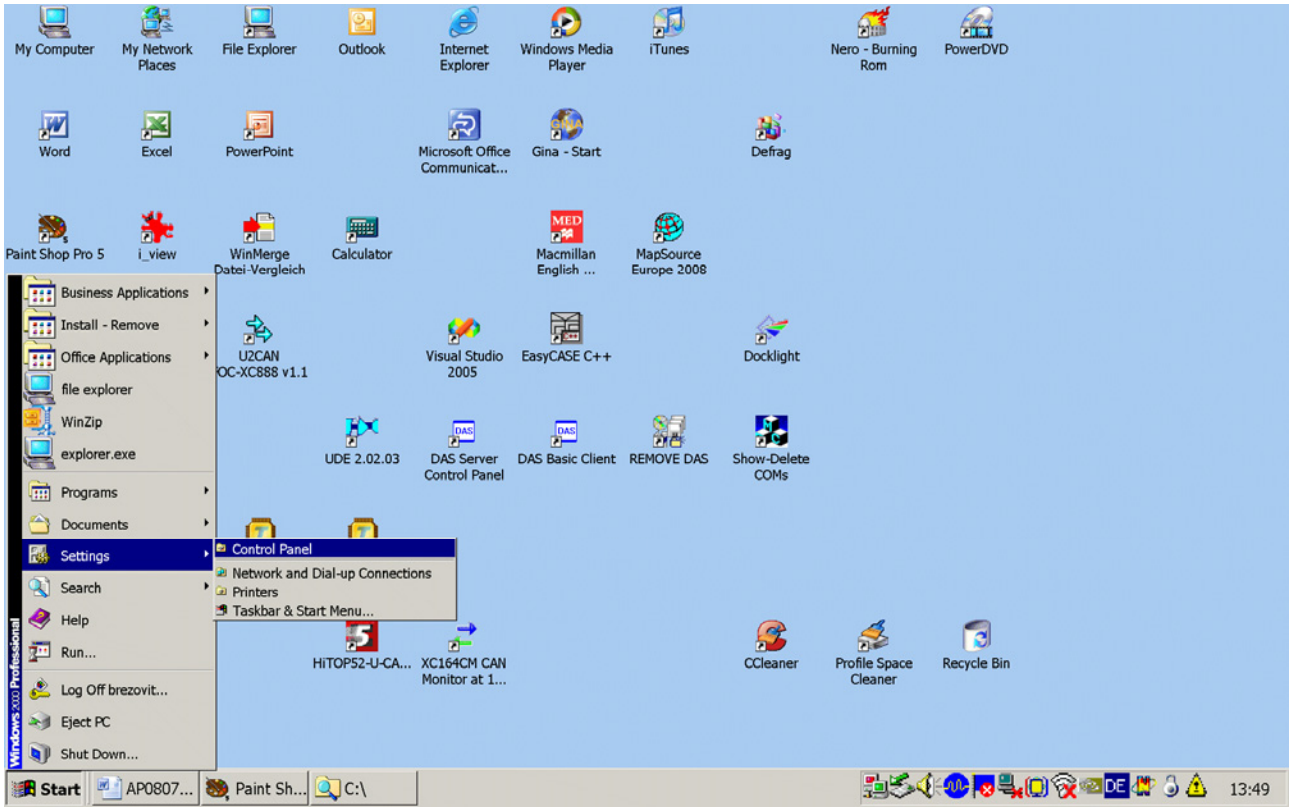

Peripherals
CAN x 2
CAN
LIN
LEDs
UART
Mode Control
JTAG
FTDI
USB to UART
USB to JTAG
control
USB
XC878CM

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

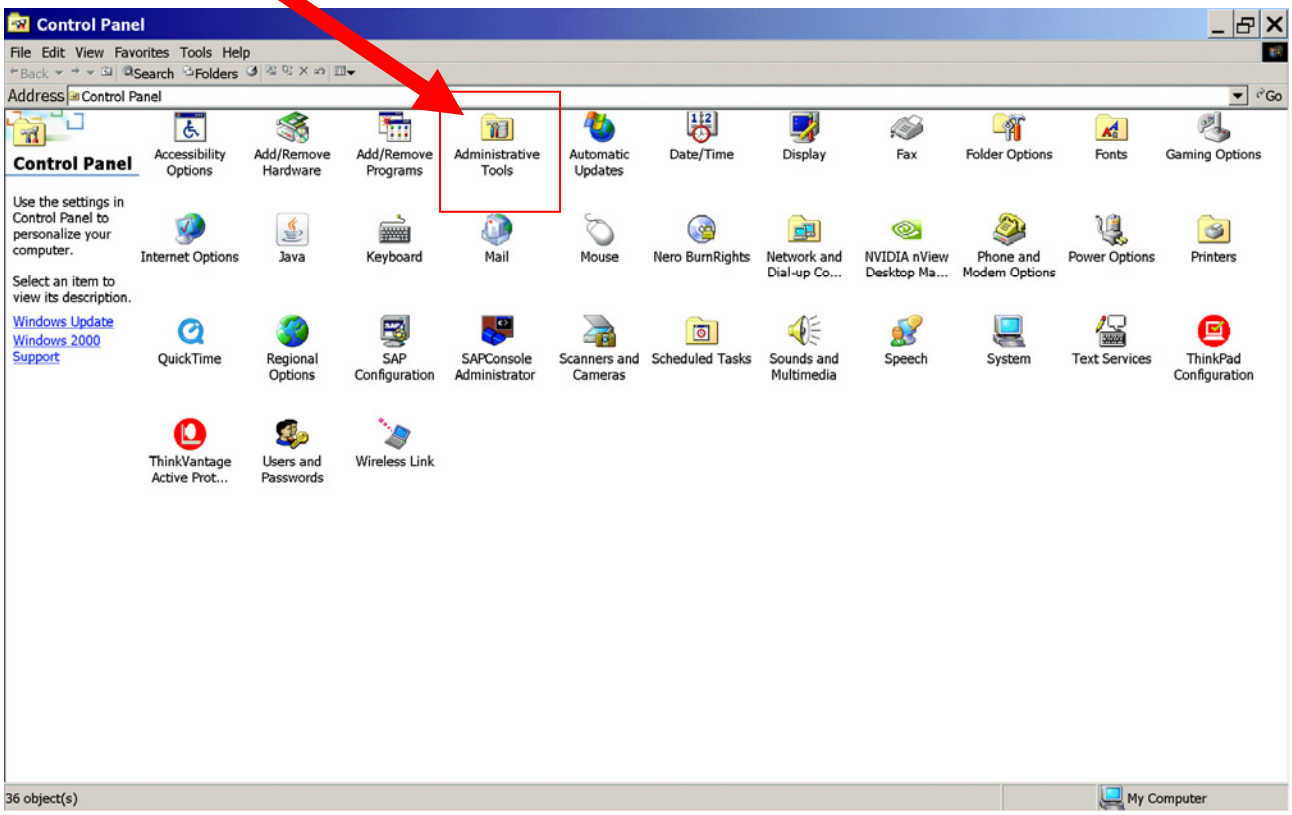
Note:
A default virtual COM Port is generated.

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:

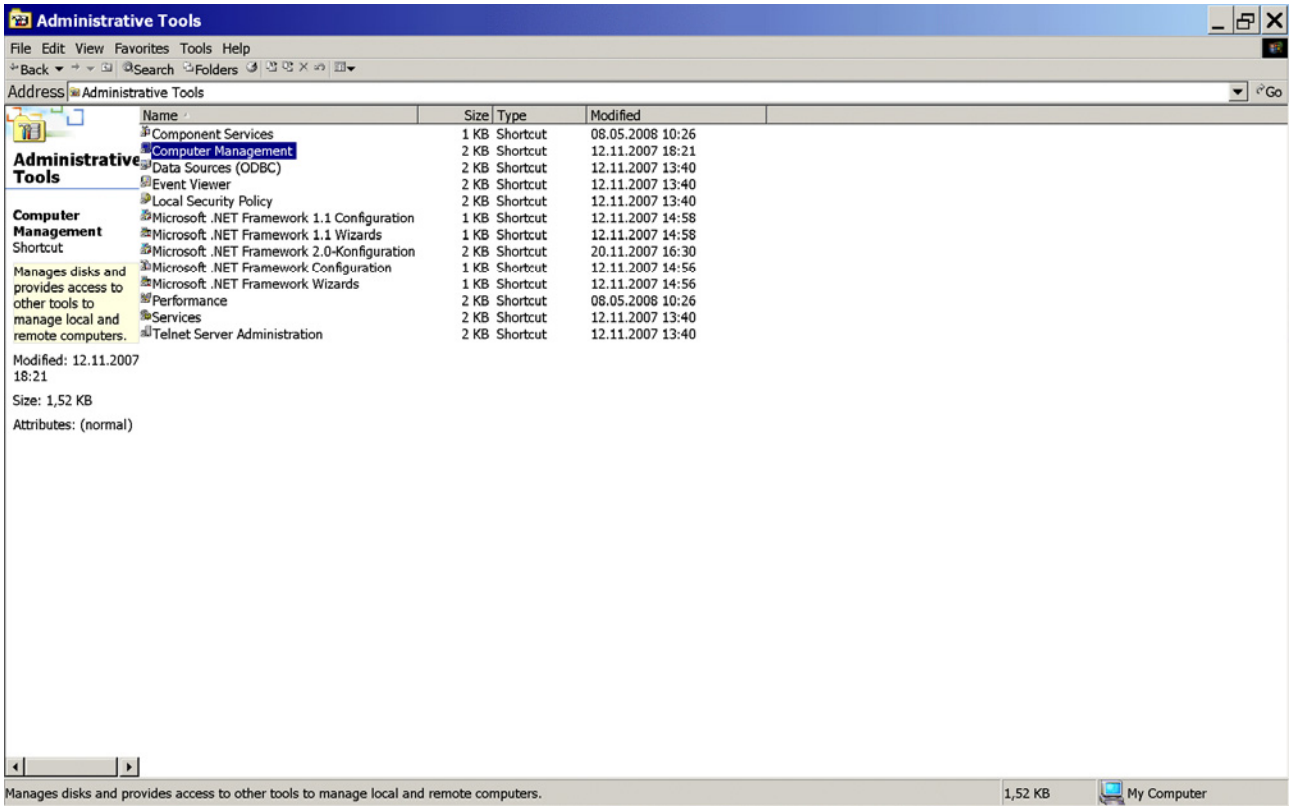
Start – Settings – Control Panel



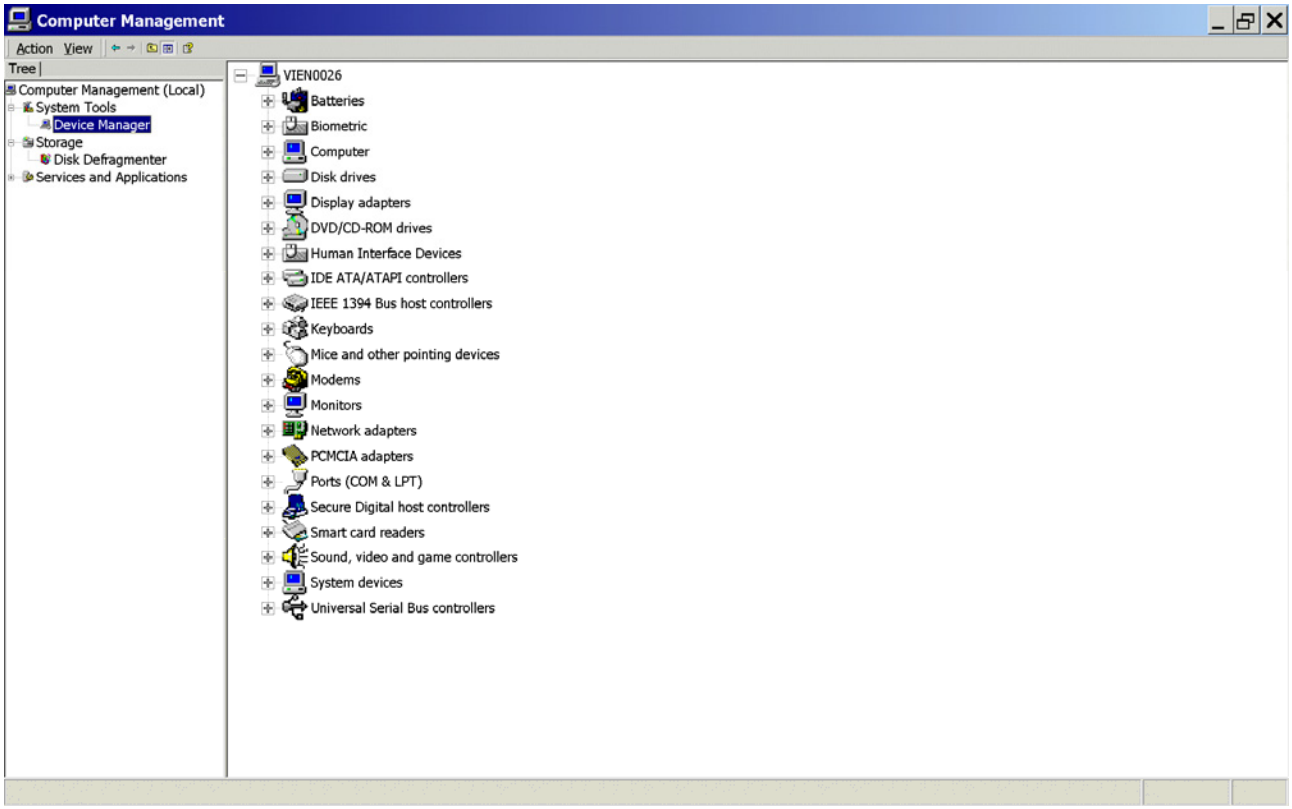
Double click: Administrative Tools



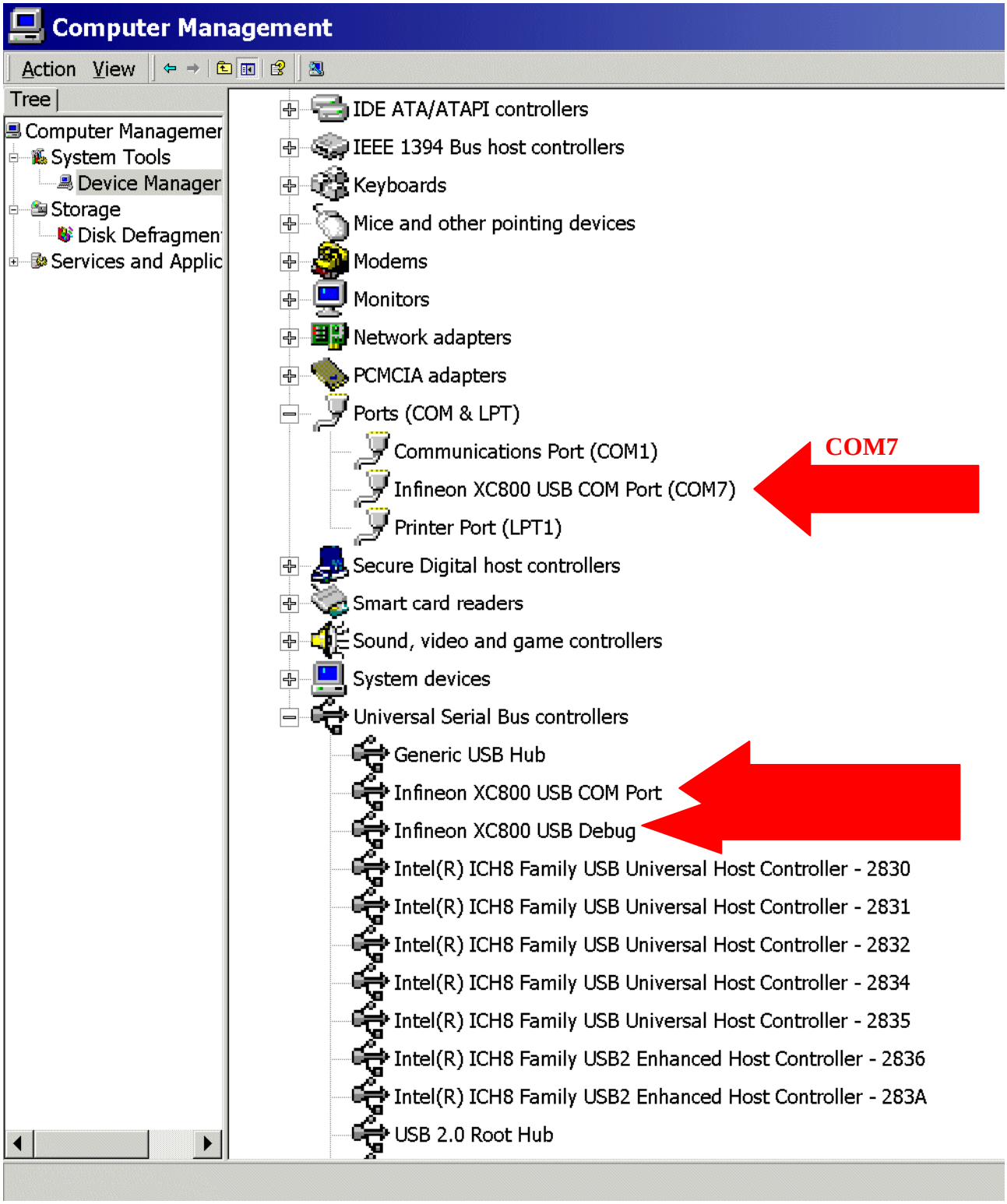
Double click: Computer Management



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 

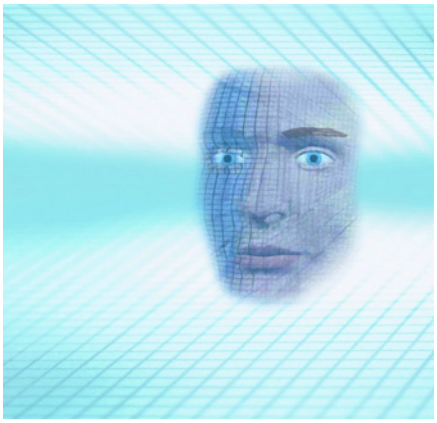
)

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

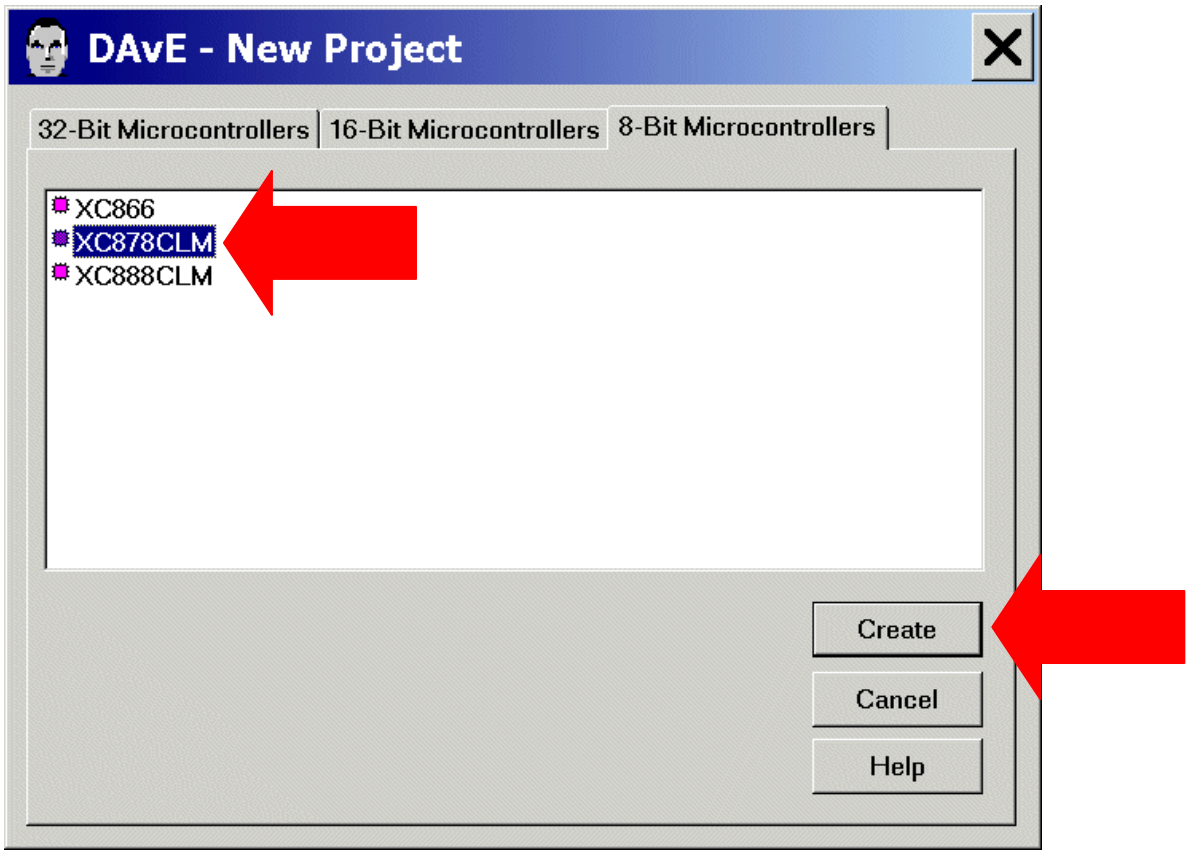
4.) DAvE is now ready to generate code for the XC878 microcontroller.

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



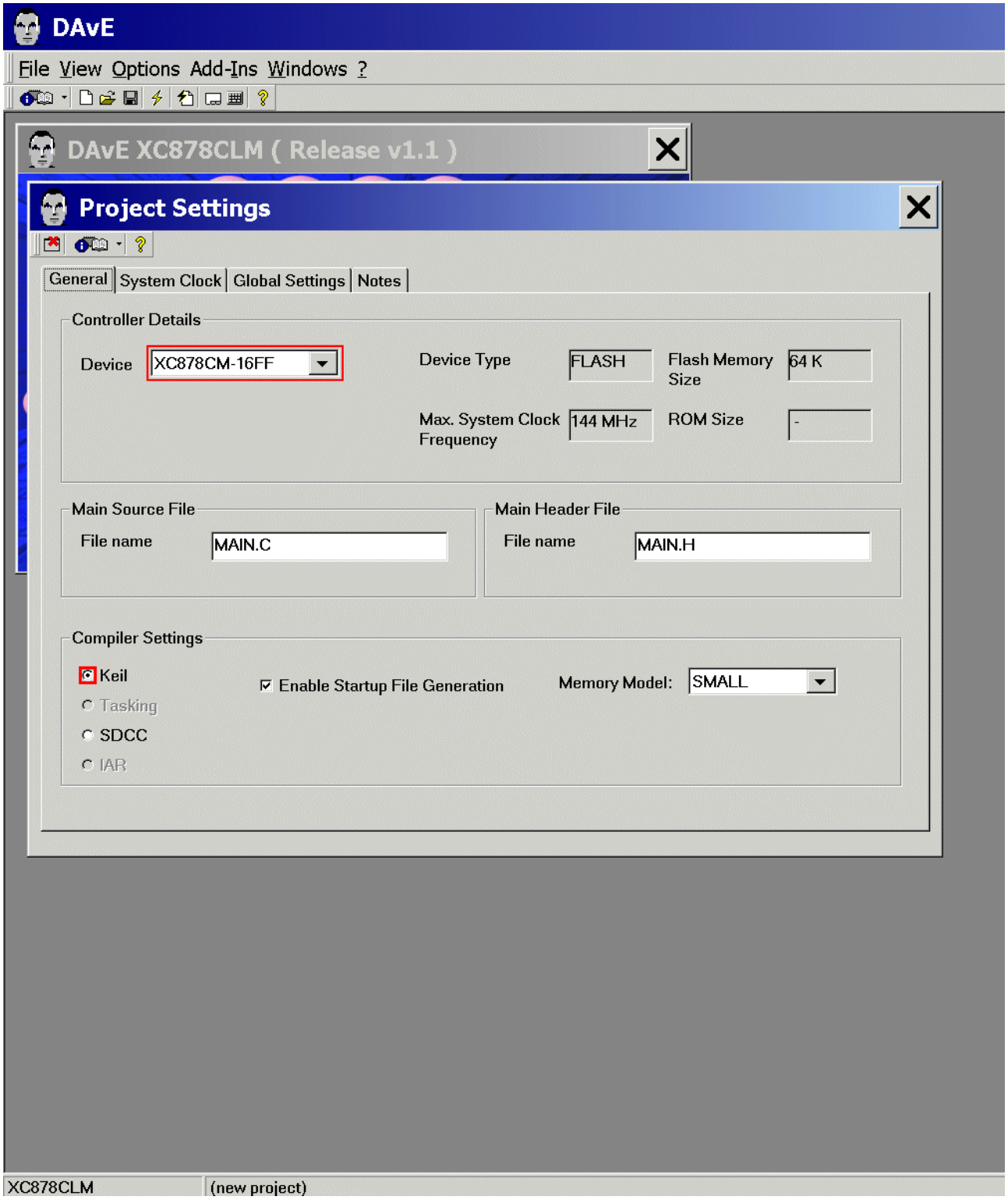
Start the program generator DAvE and select the XC878 microcontroller:

File
New
8-Bit Microcontrollers
select XC878CLM
Create

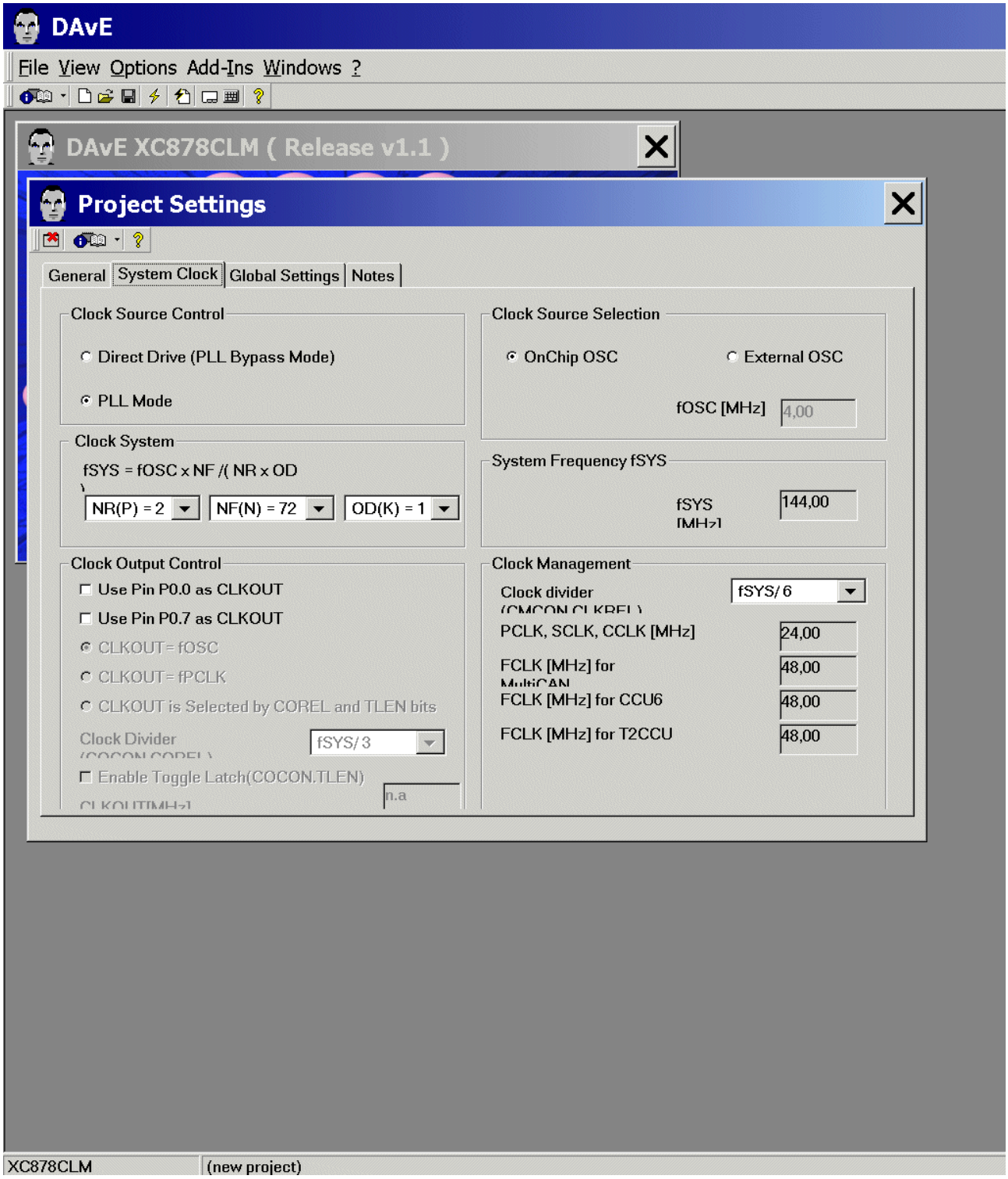


Choose the Project Settings as you can see in the following screenshots:

General: Controller Details: Device: check/select XC878CM-16FF
General: For the KEIL Compiler check/choose ⊙ Keil in the Compiler Settings:

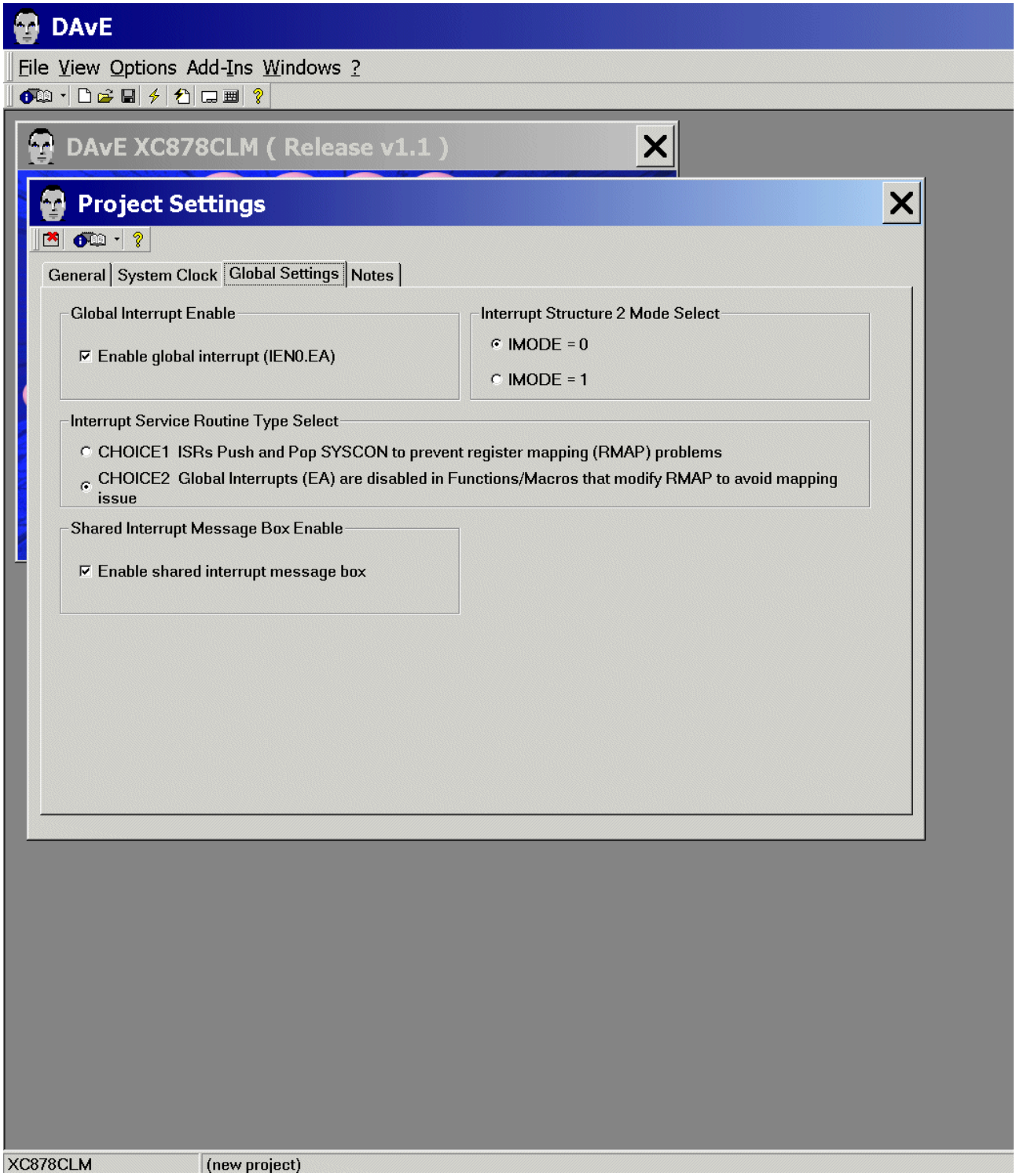


System Clock: (do nothing)



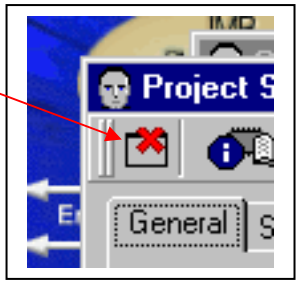
Note:
CPU clock is 24 MHz.

Global Settings: (do not change configuration)



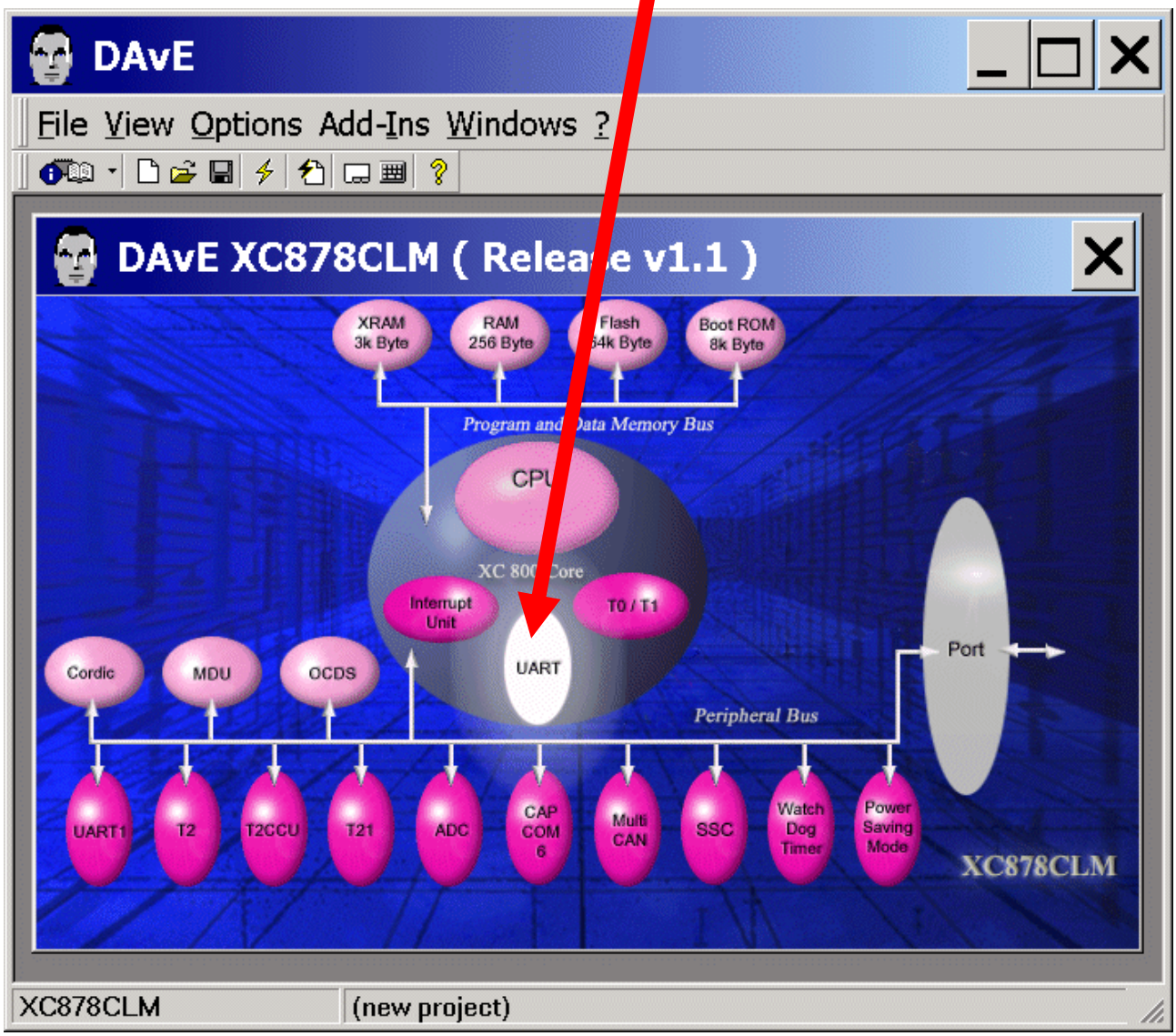
Notes: If you wish, you can insert your comments here.

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

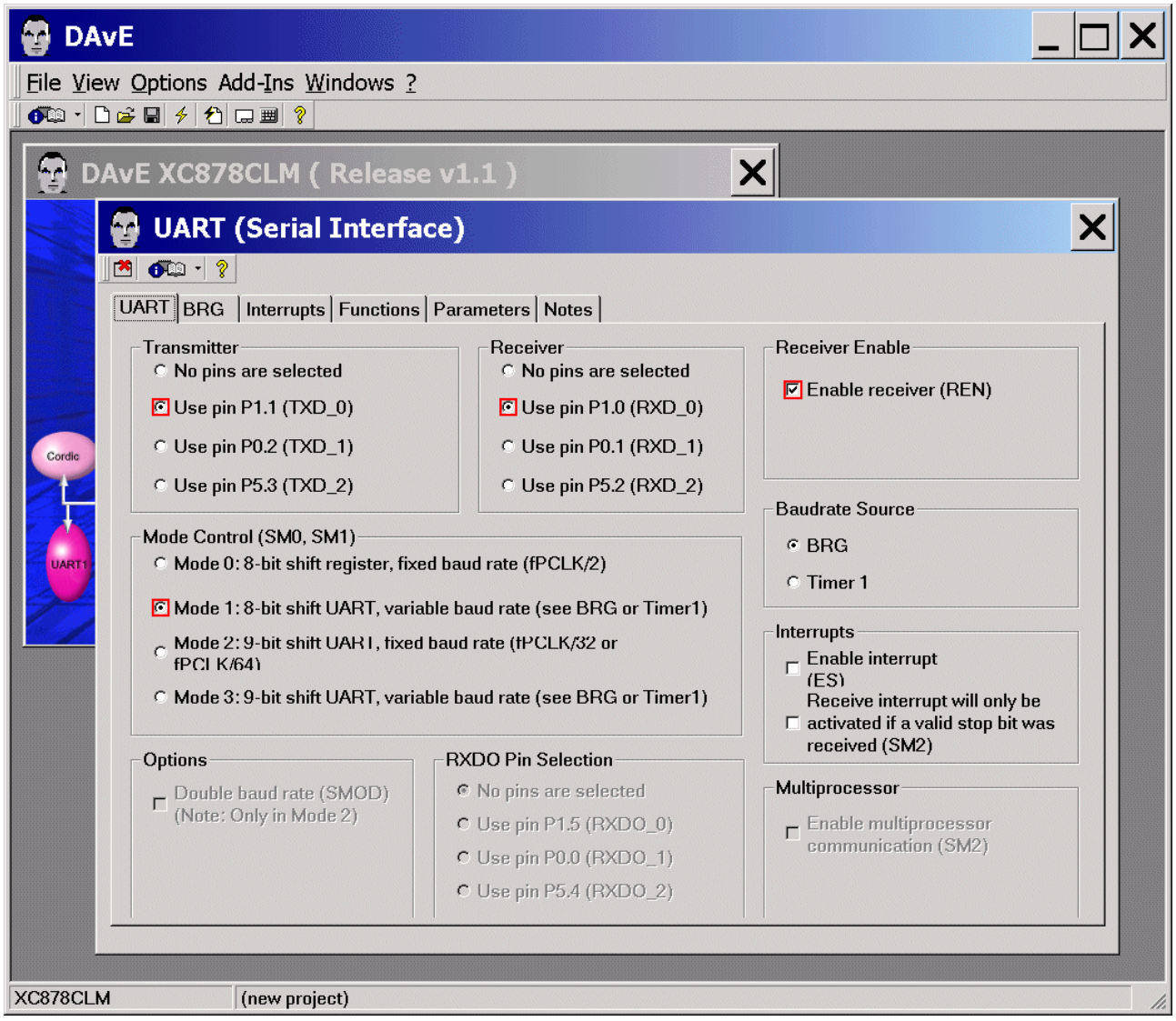


Configuration of the UART:

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

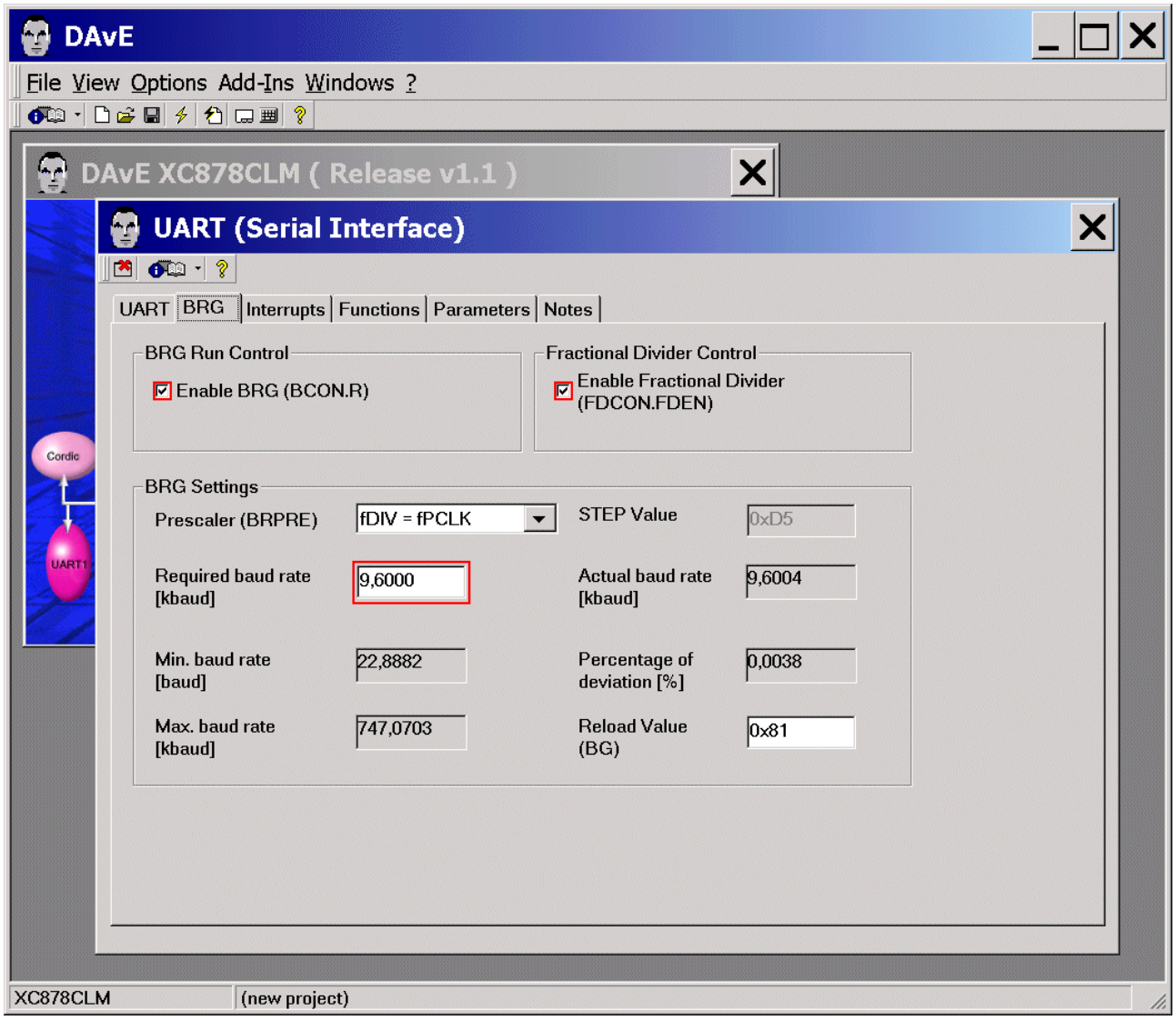


UART: Transmitter: click ⊙ Use pin P1.1 (TXD_0)
UART: Receiver: click ⊙ Use pin P1.0 (RXD_0)
UART: Receiver Enable: tick ✓ Enable receiver (REN)
UART: Mode Control: click ⊙ Mode 1: 8-bit shift UART, variable baud rate (see BRG or Timer1)



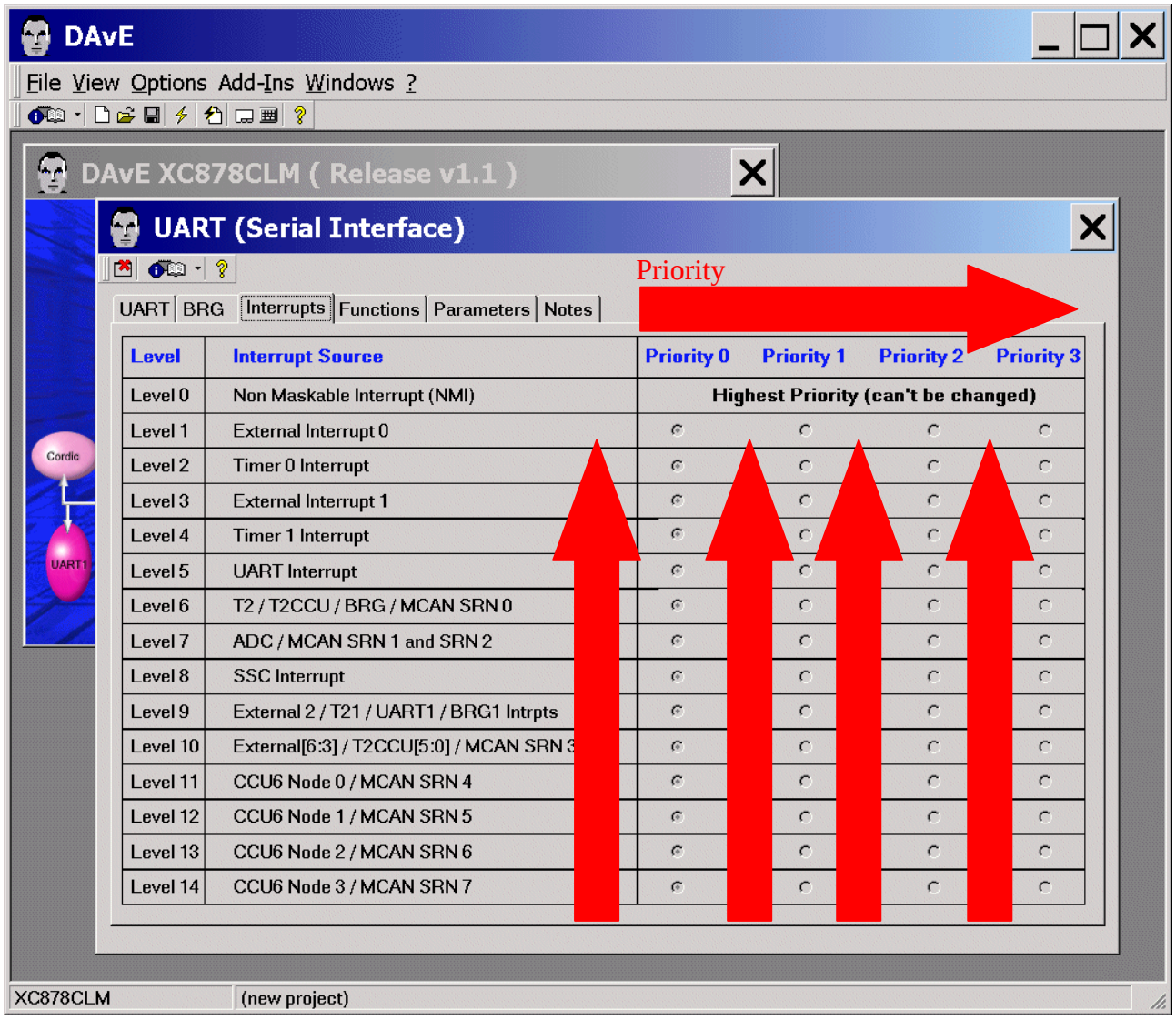
Note:
The RS232 serial interface (UART pins P1.0 and P1.1) is available via the USB port as virtual COM port (e.g. COM7) which converts the TTL-UART-signals to USB-signals.

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)



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.

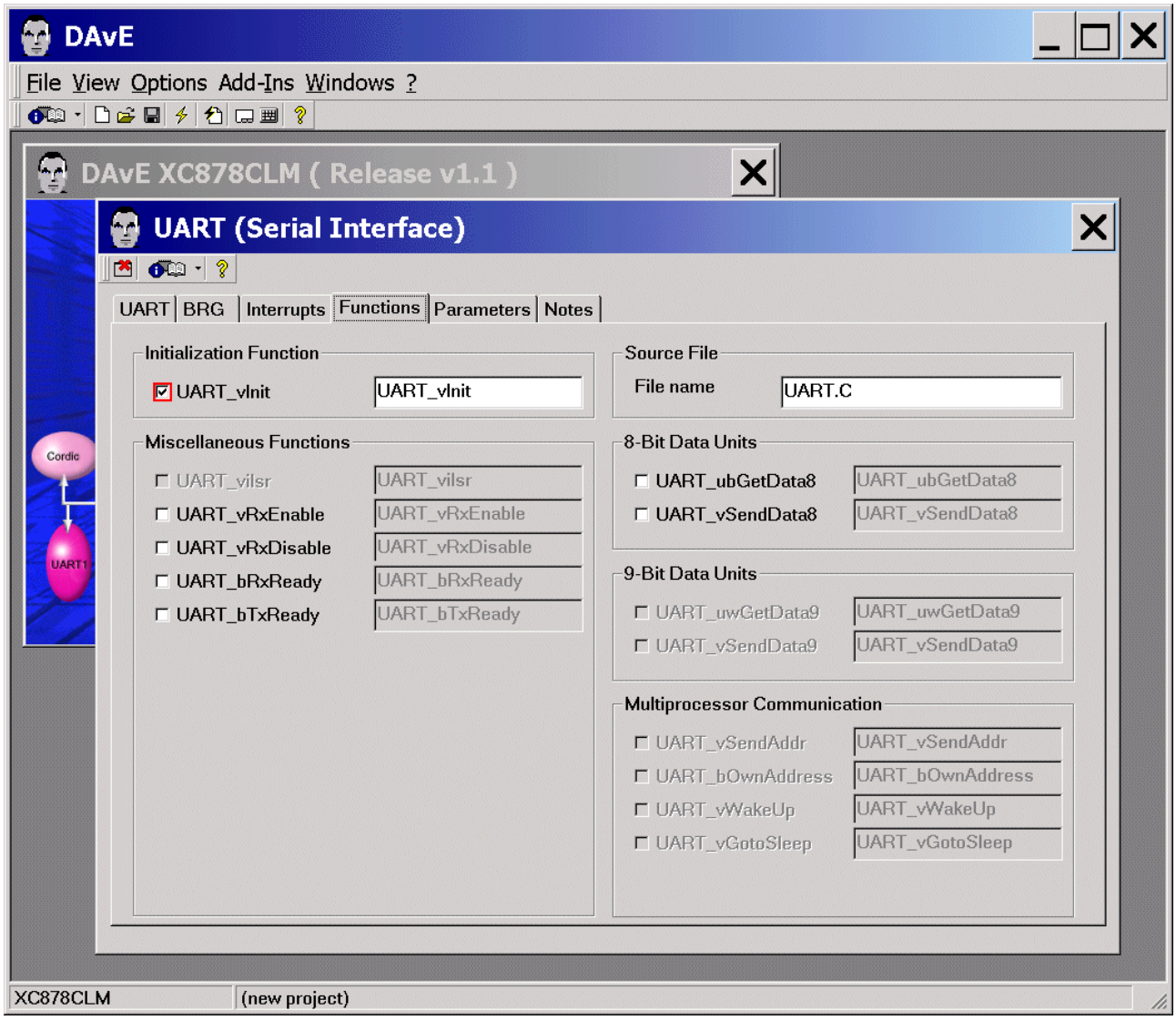
Interrupt Priorities:

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

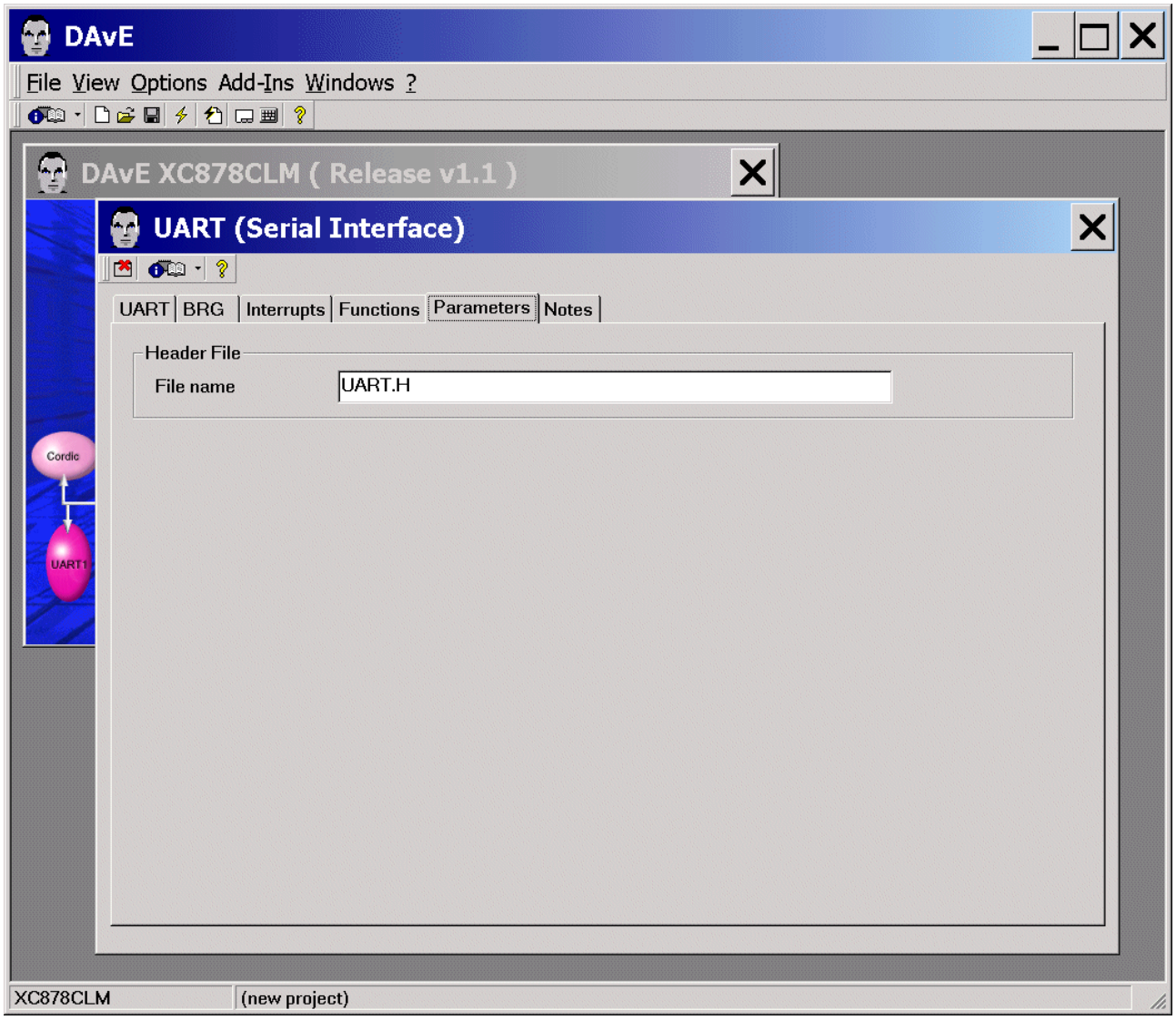
Main refers to routines that run prior to any interrupt and can be interrupted by any interrupt.
Each interrupt source can be programmed to any of the four interrupt priorities (0-3).
An interrupt that is currently being serviced can only be interrupted by a higher-priority interrupt, but not by another 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: tick ✓ UART_vInit



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

Parameters: (do nothing)

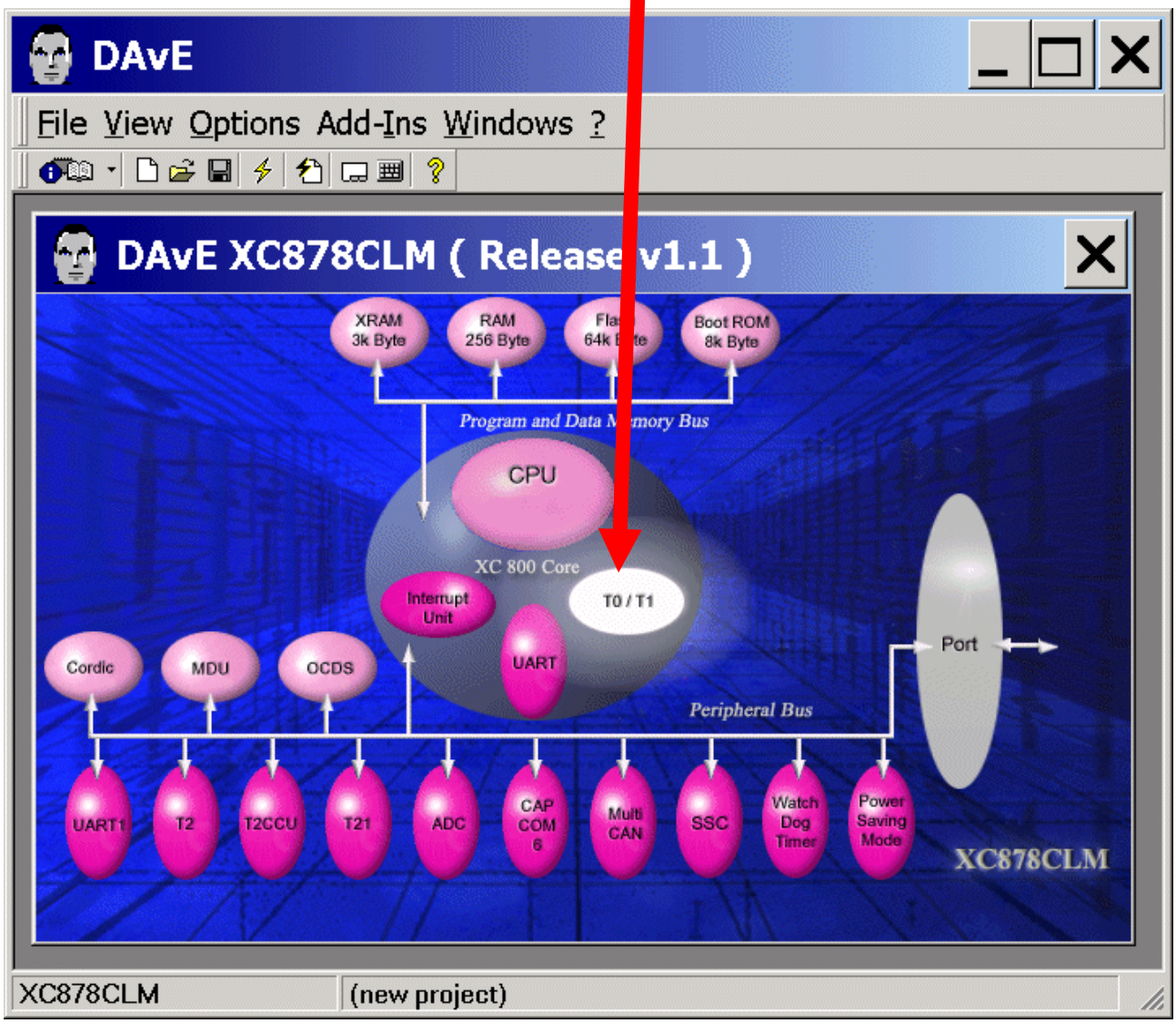


Notes: If you wish, you can insert your comments here.

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

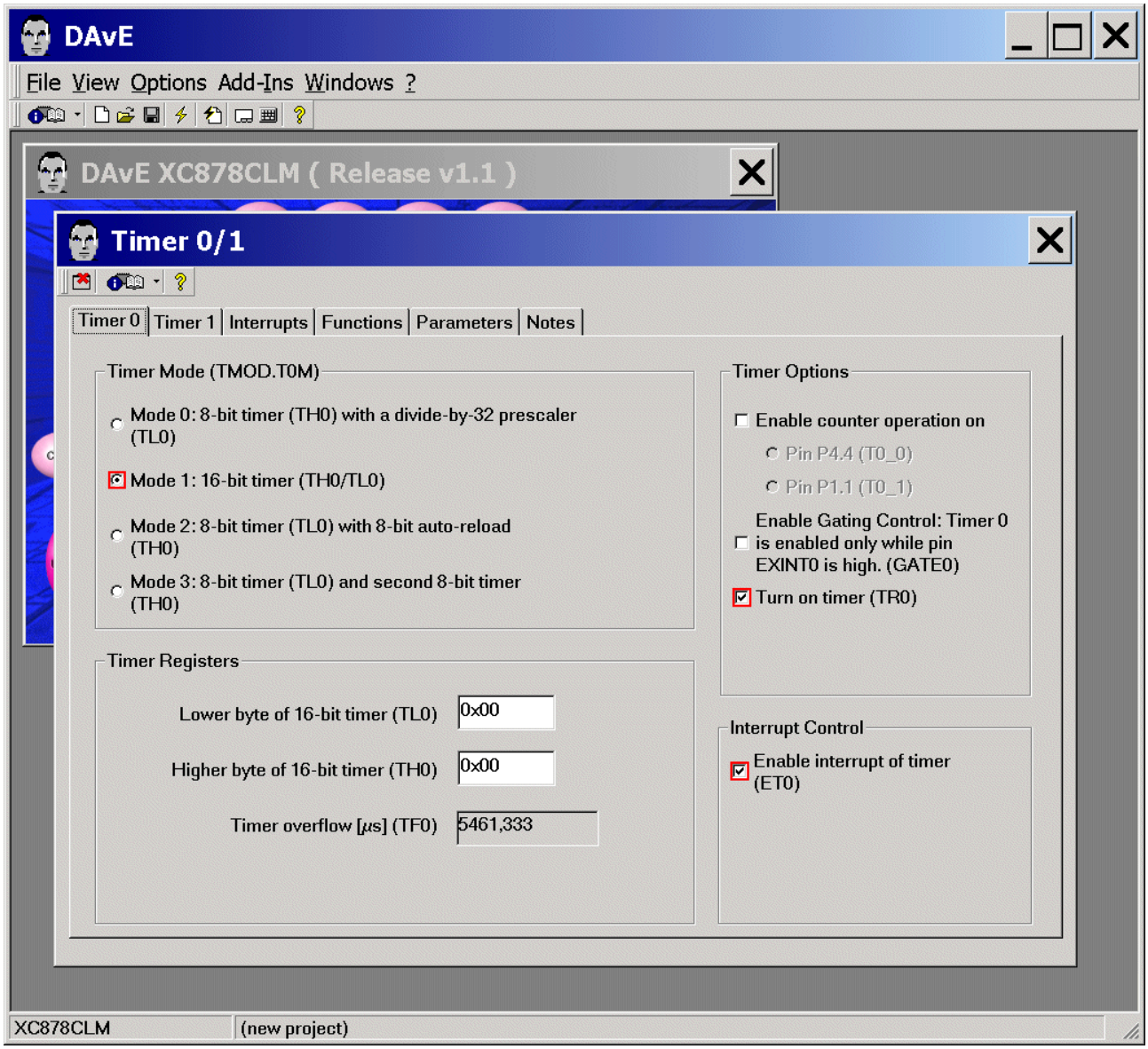
Configure Timer T0:

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



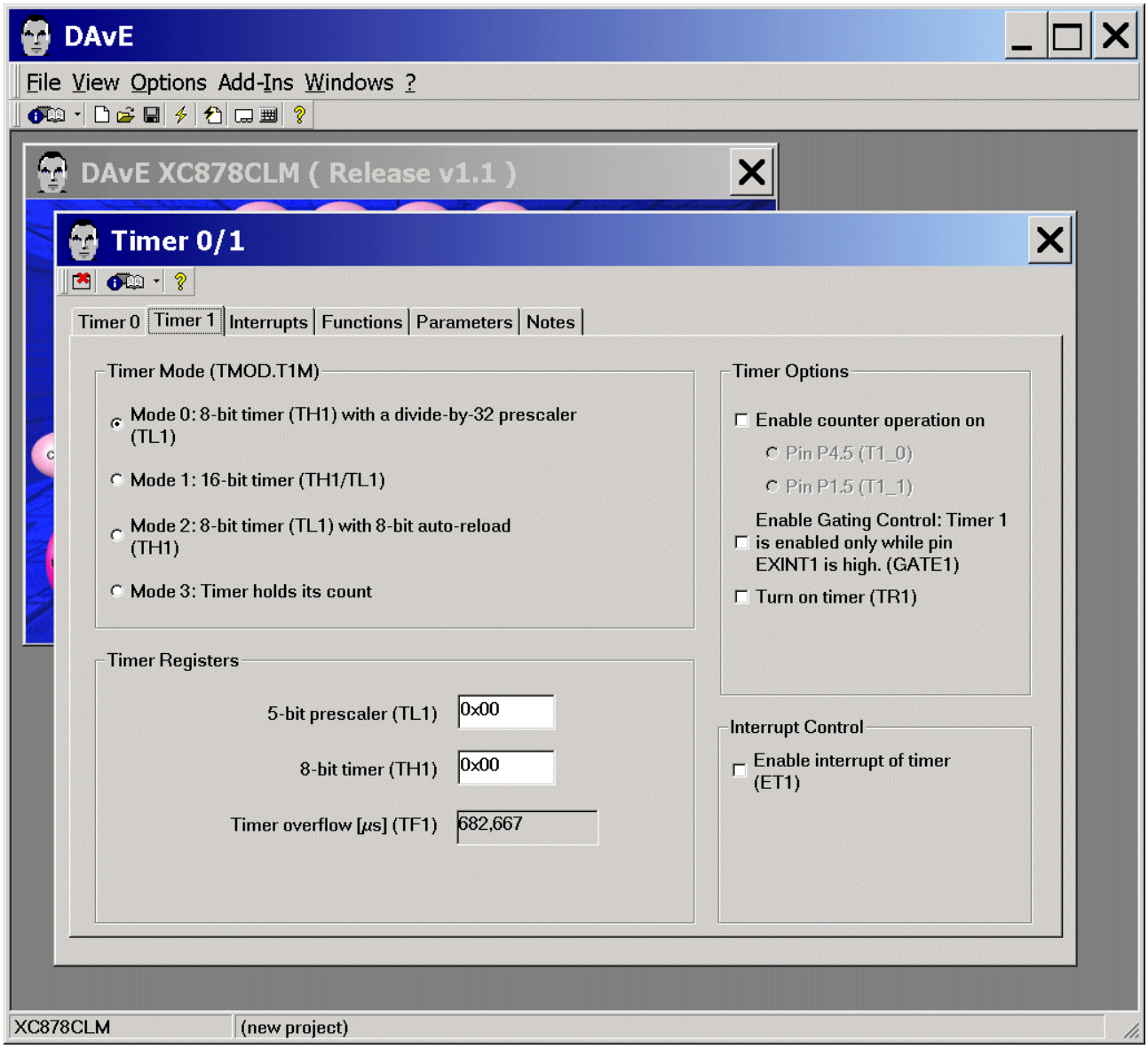
Note:
The LEDs on Port_3 will be blinking (if selected in the main menu)
with a frequency of about 1 second (done in the Timer_0-Interrupt-Service-Routine).
Therefore we have to configure Timer_0.

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)

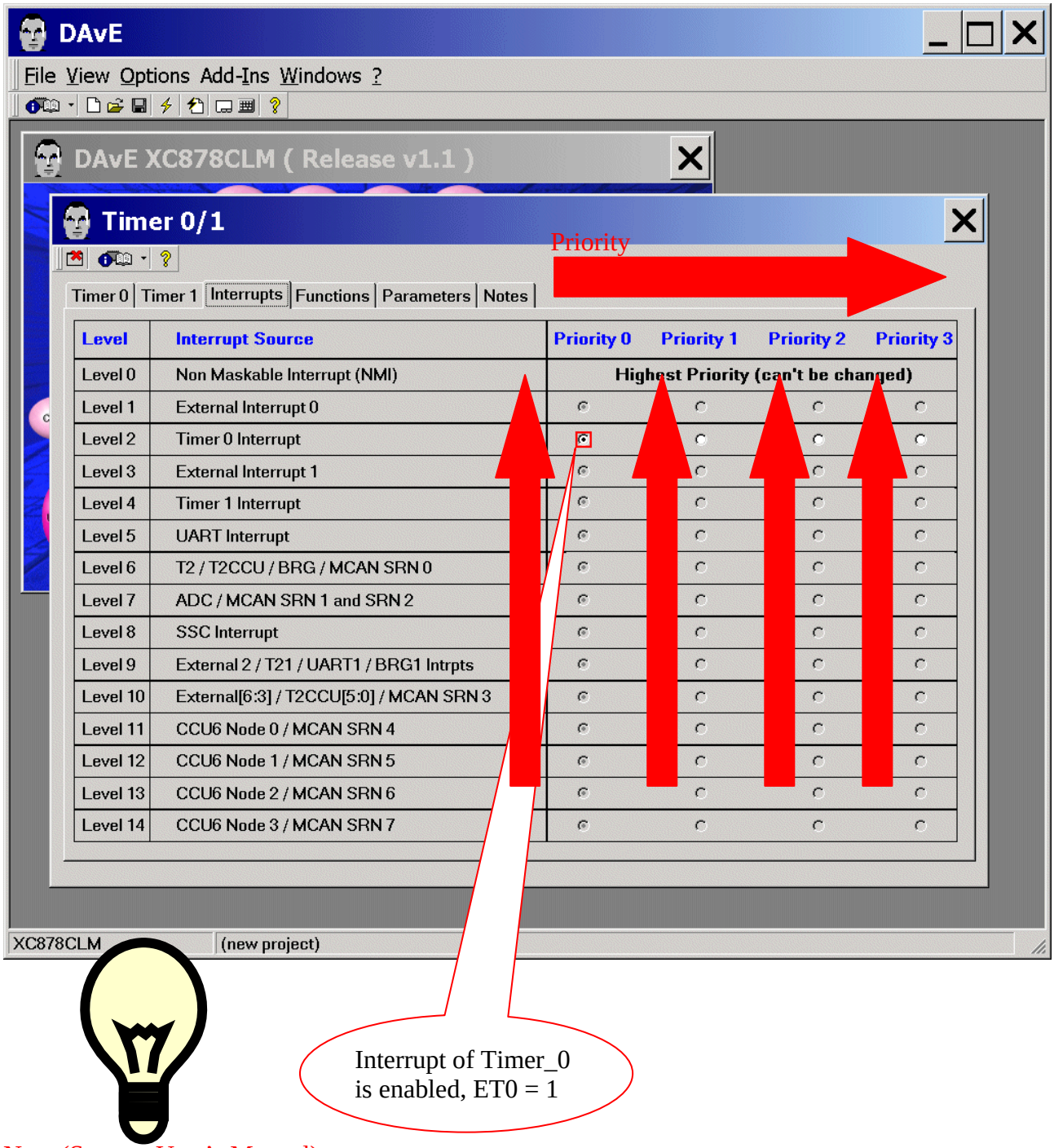


Note:
We need 183 Timer_0 overflows to achieve an approximate 1 second delay.
This will be handled in the Timer_0 interrupt function.
183 * 5461,333 µs = 0,9994 s.

Timer1: do nothing (not used)

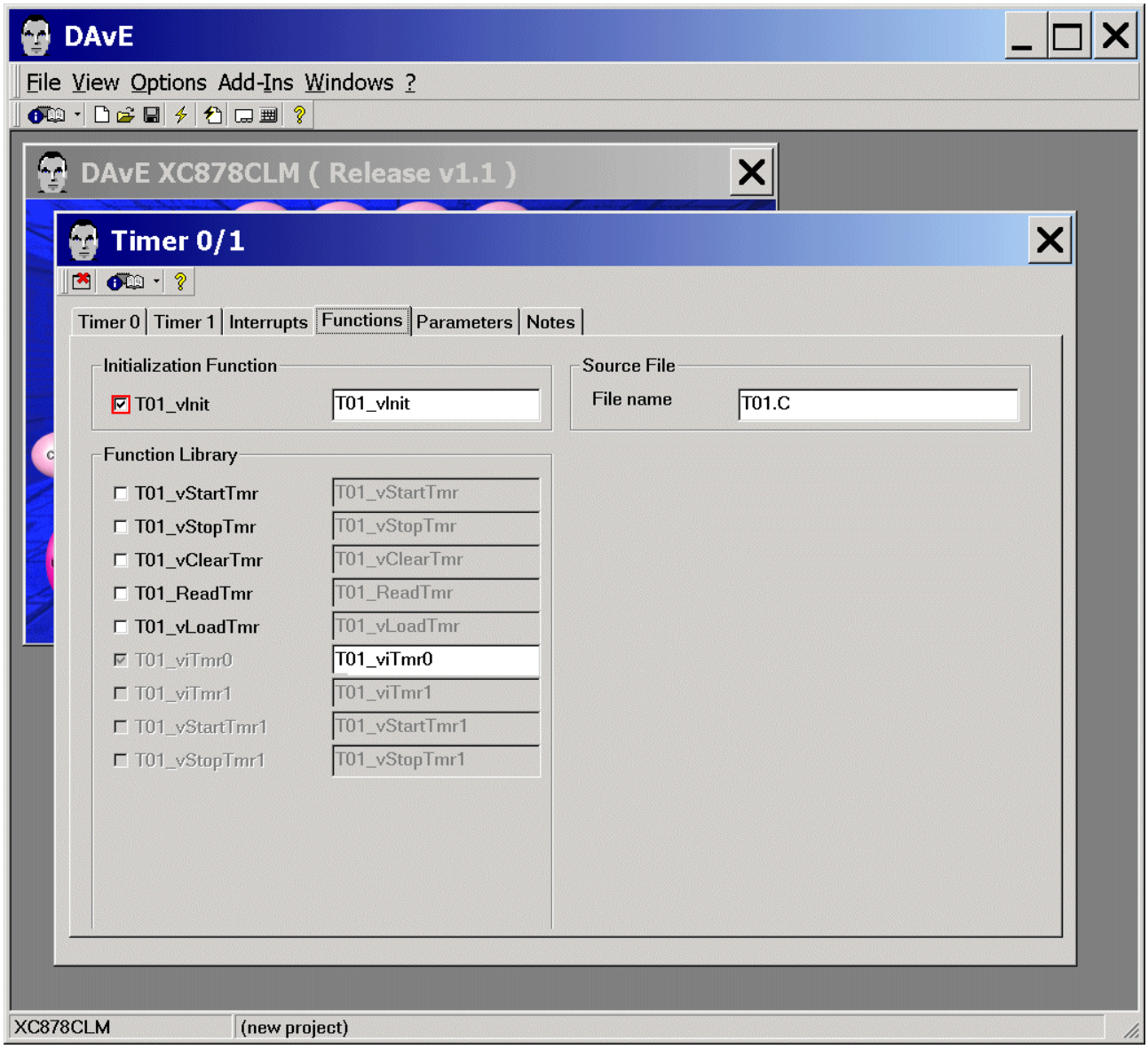


Interrupts: (do nothing)



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

Functions: Initialization Function: tick ✓ T01_vInit

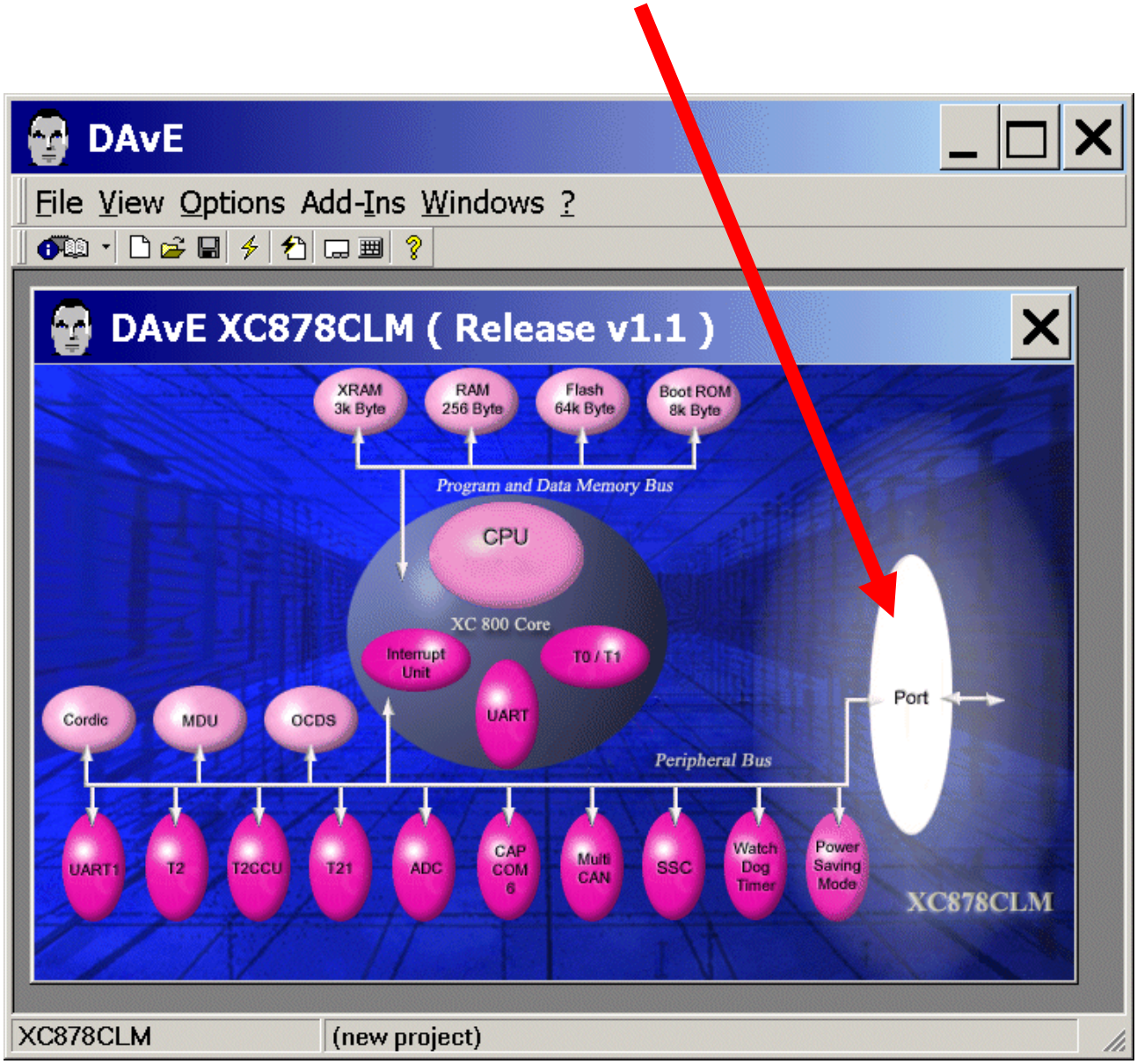


Parameters: (do nothing)
Notes: If you wish, you can insert your comments here.

Exit this dialog now by clicking the close button.

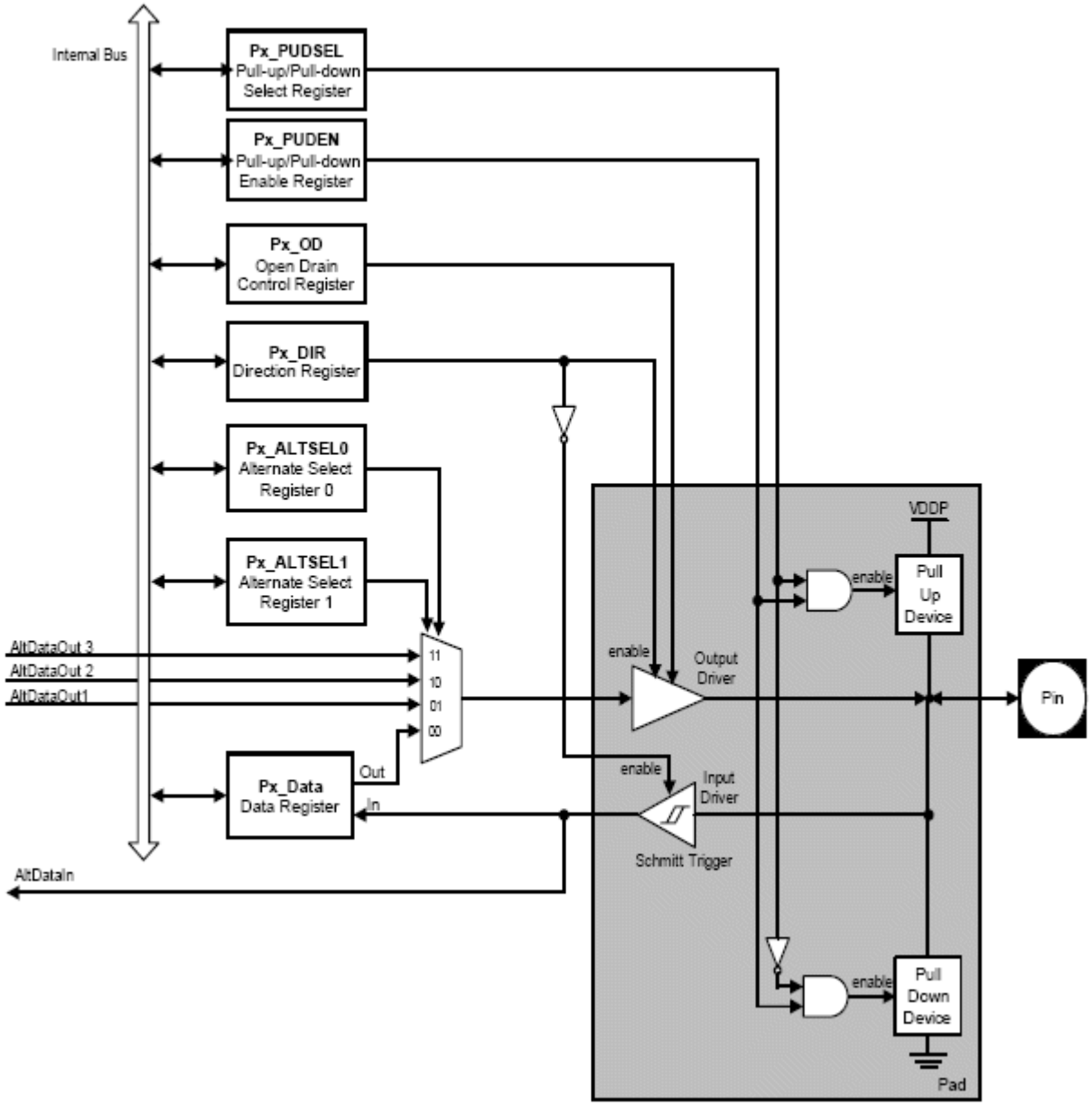
Configure Port 3 to Output:

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

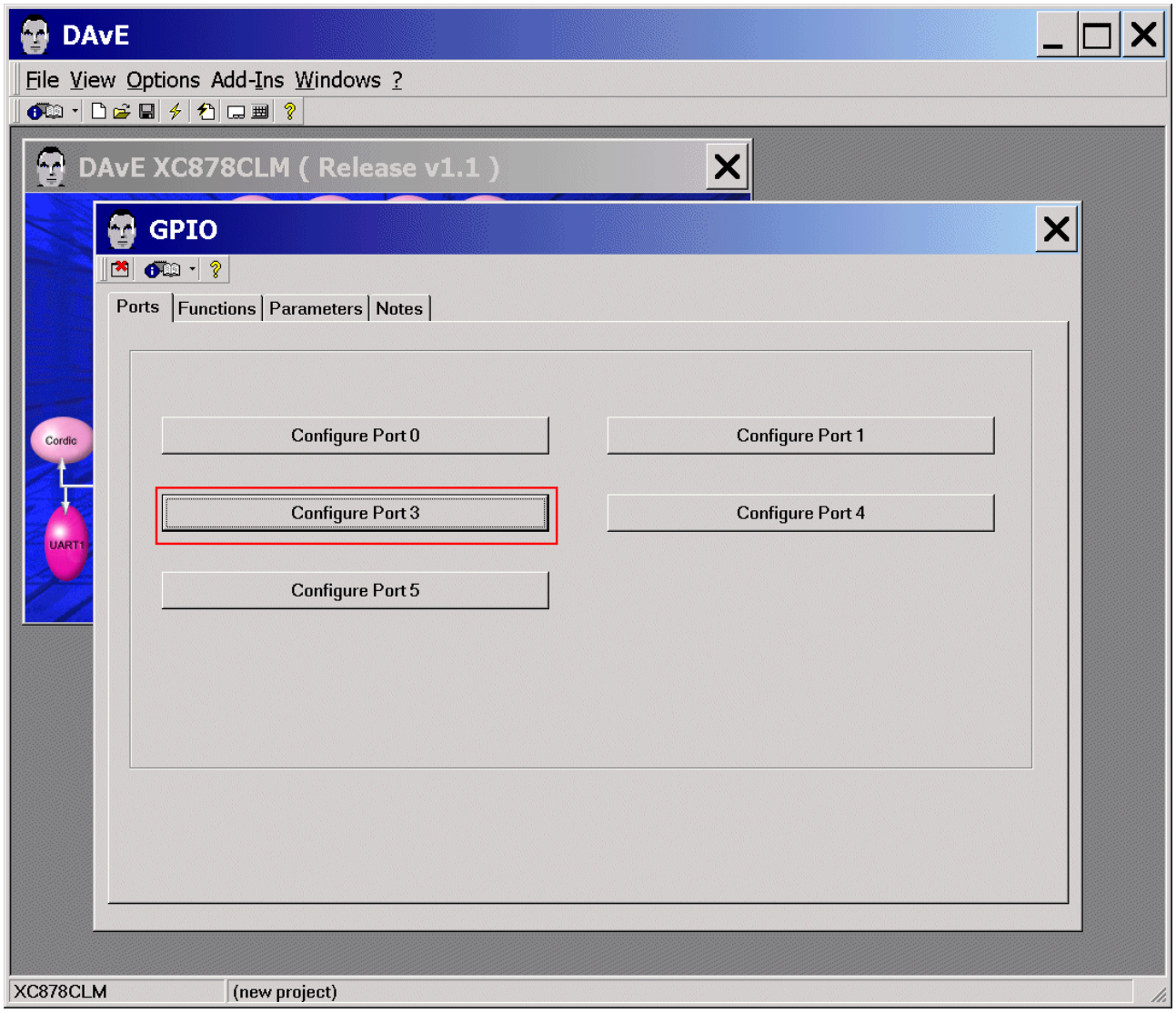


Note:
The User LEDs (red) are connected to Port_3.

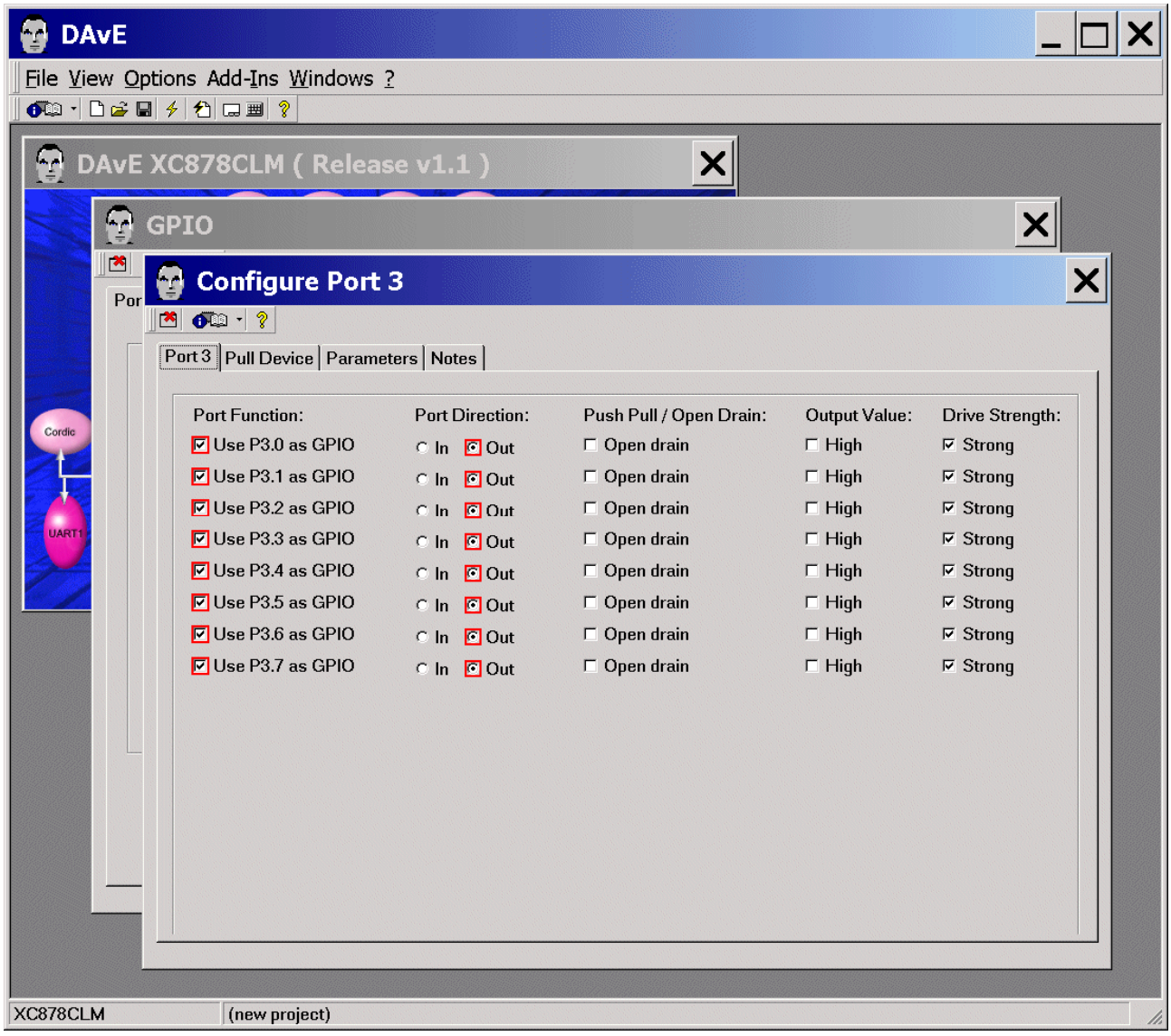
Additional information: Parallel Ports – General Structure (Source: User's Manual):



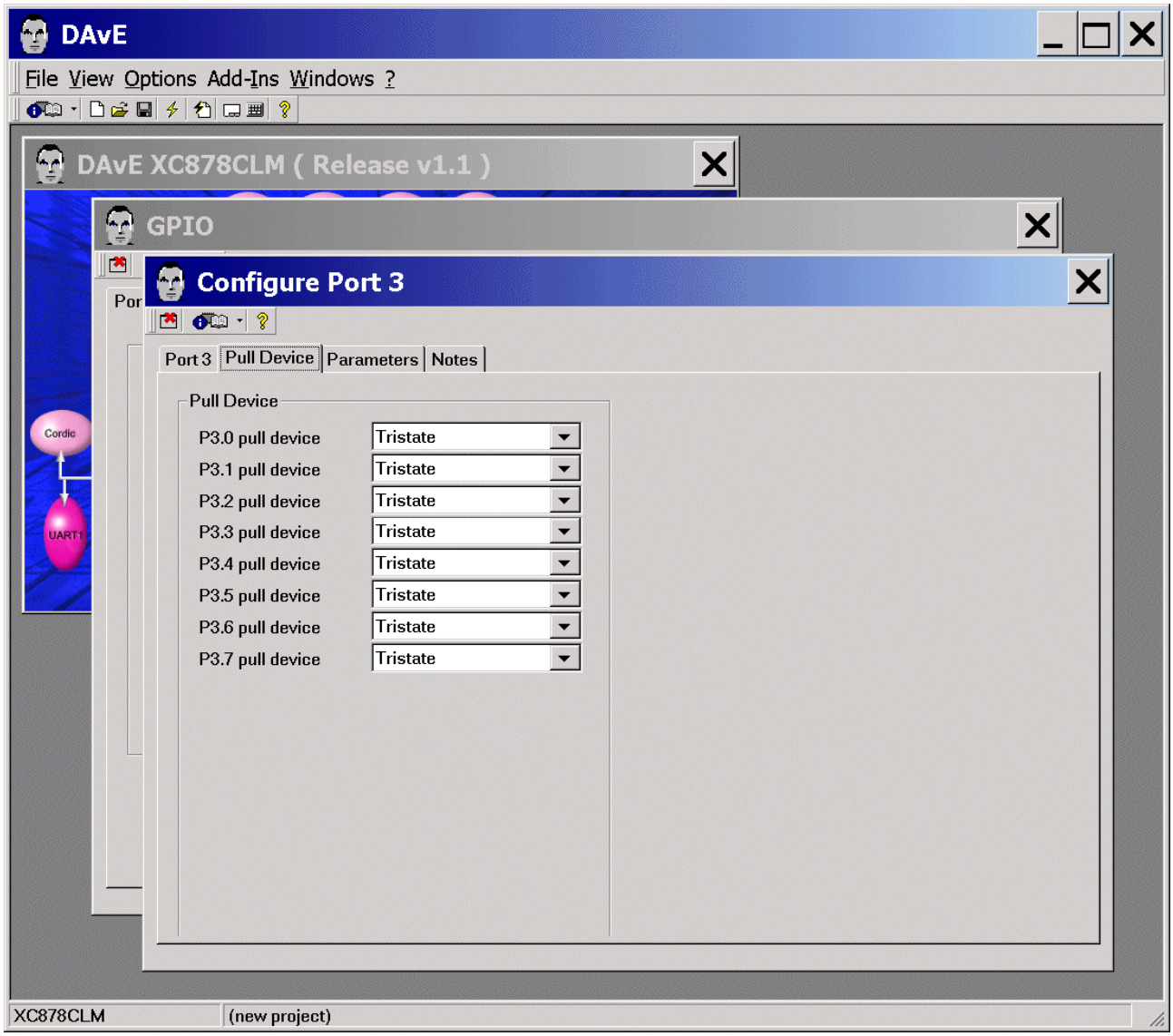
Ports: click “Configure Port 3”



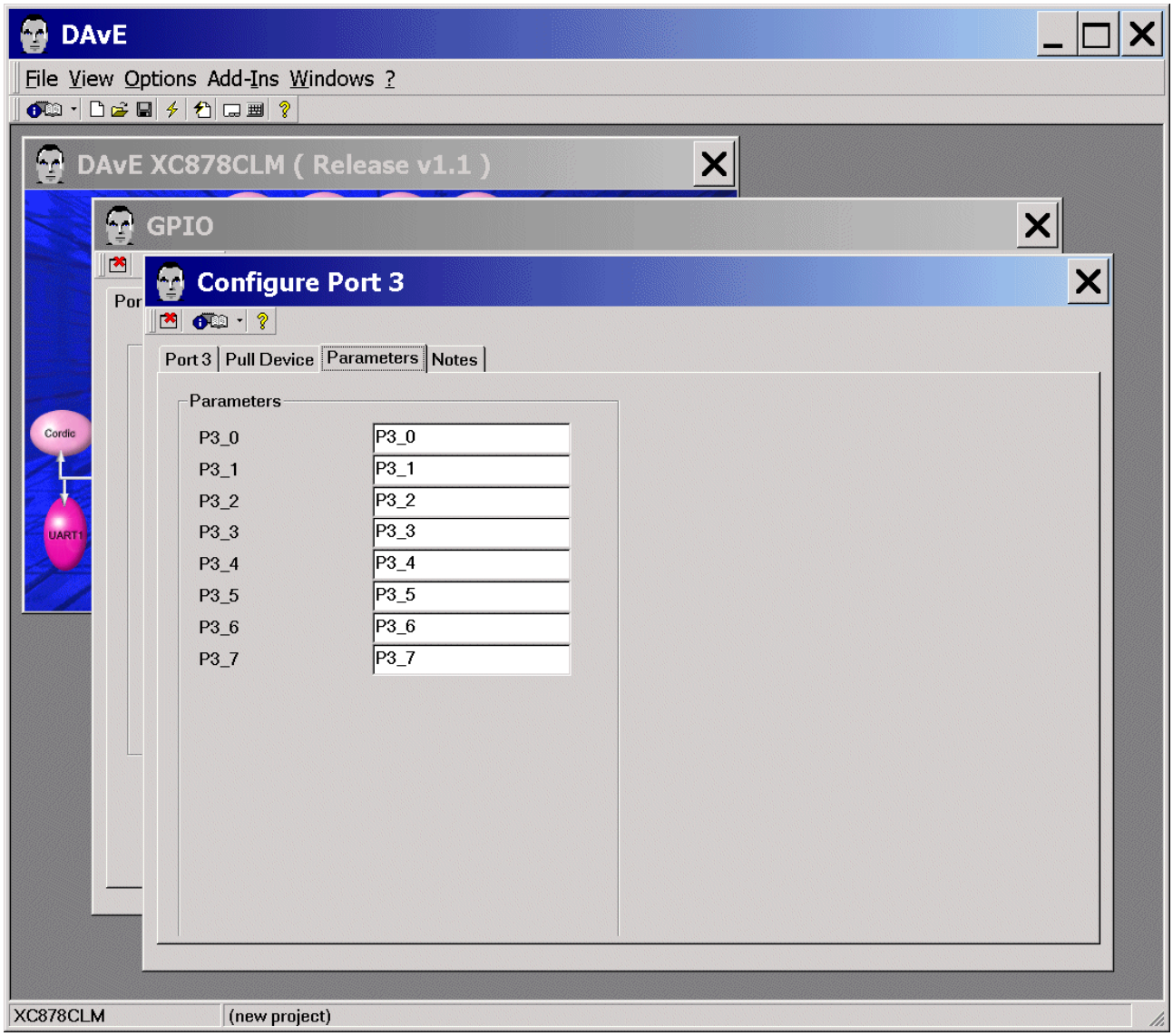
Port 3: Port Function: tick ✓ Use P3.0 as GPIO - Port Direction: click ⊙ Out
Port 3: Port Function: tick ✓ Use P3.1 as GPIO - Port Direction: click ⊙ Out
Port 3: Port Function: tick ✓ Use P3.2 as GPIO - Port Direction: click ⊙ Out
Port 3: Port Function: tick ✓ Use P3.3 as GPIO - Port Direction: click ⊙ 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)



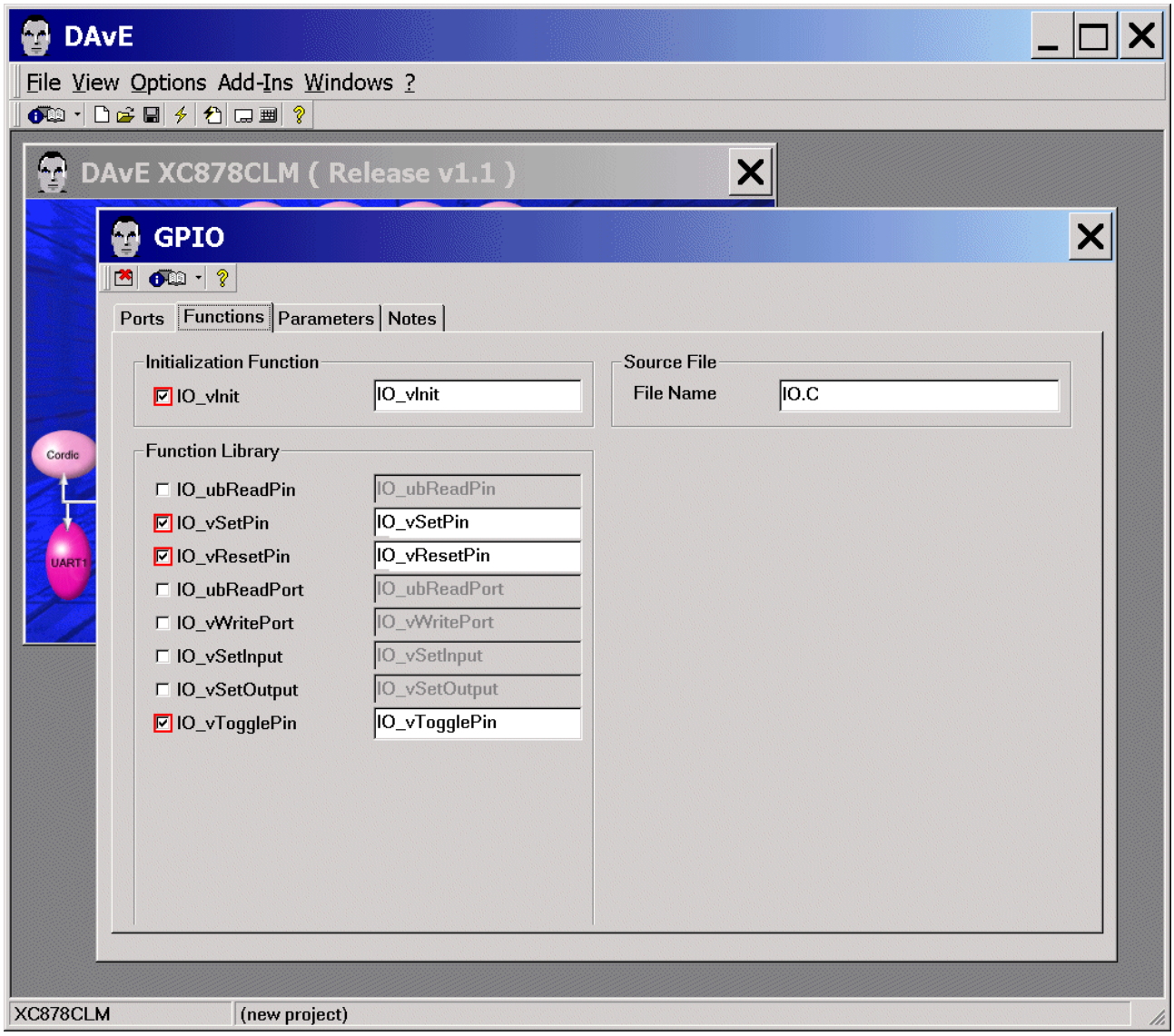
Parameters: (do nothing)



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



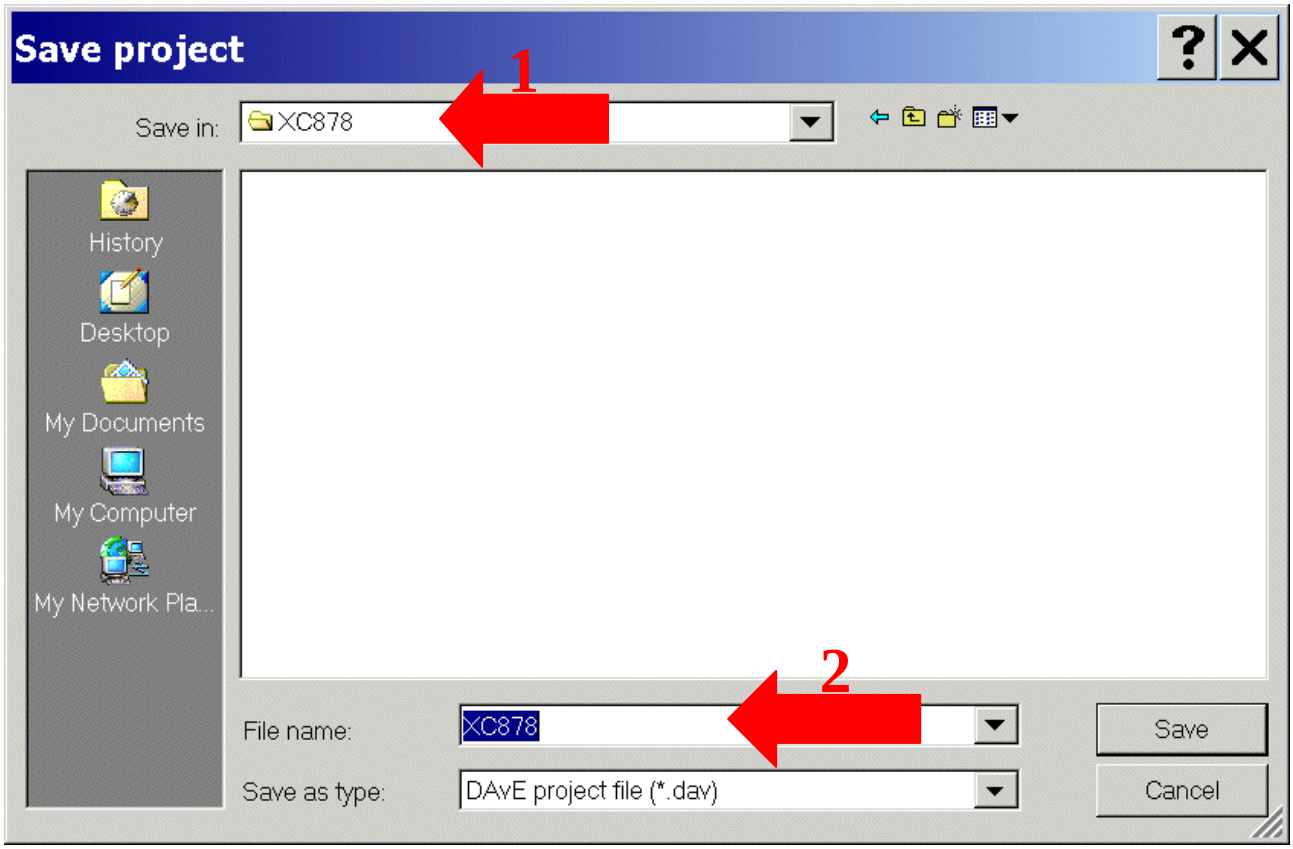
Parameters: (do nothing)
Notes: If you wish, you can insert your comments here.
Exit this dialog now by clicking the close button.

Save the project:

File
Save

Save project: Save in C:\XC878 [create new directory 

(1)]
File name: XC878 (2)

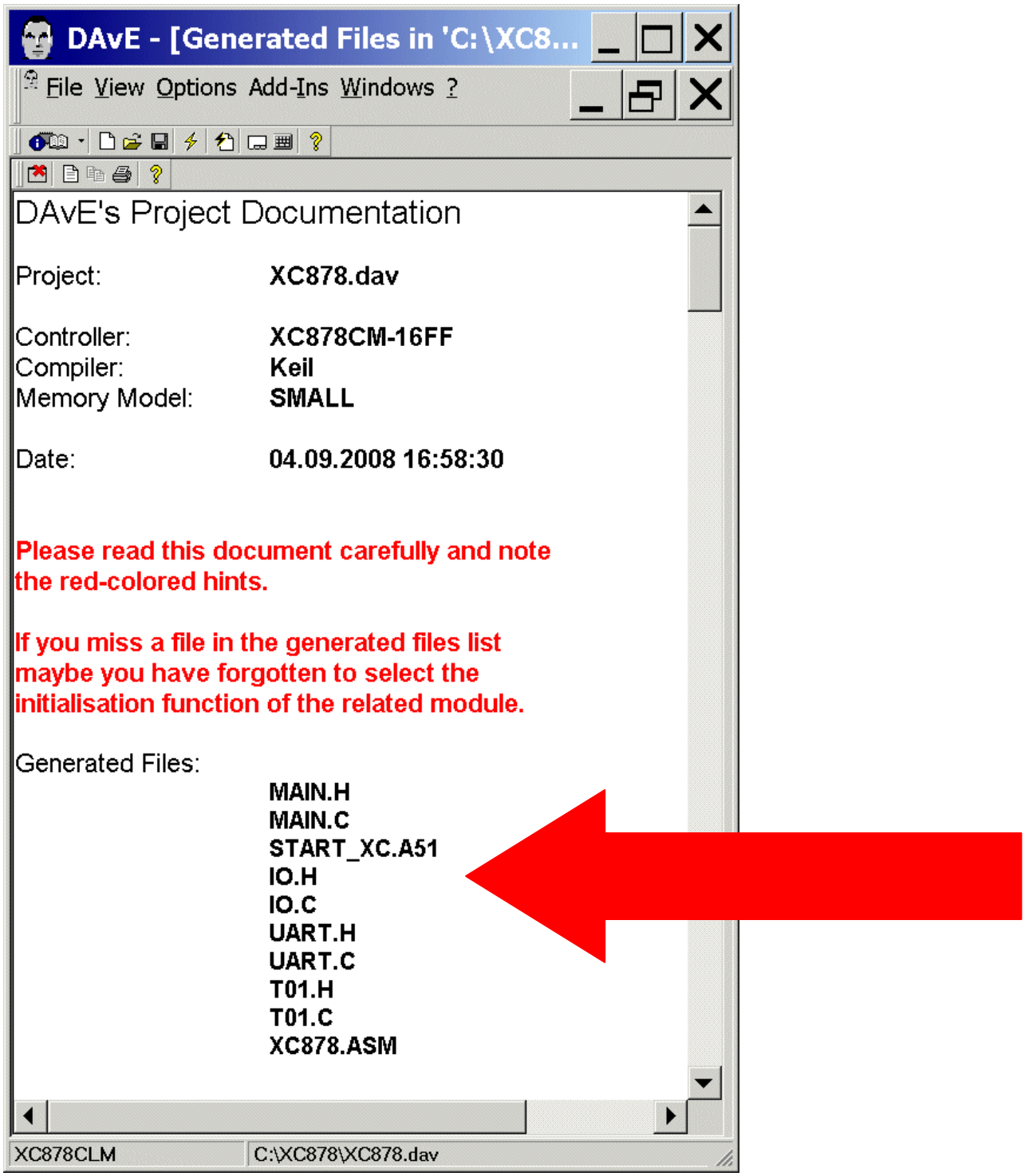


Save

Generate Code:

File Generate Code	or click 

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



File - Exit

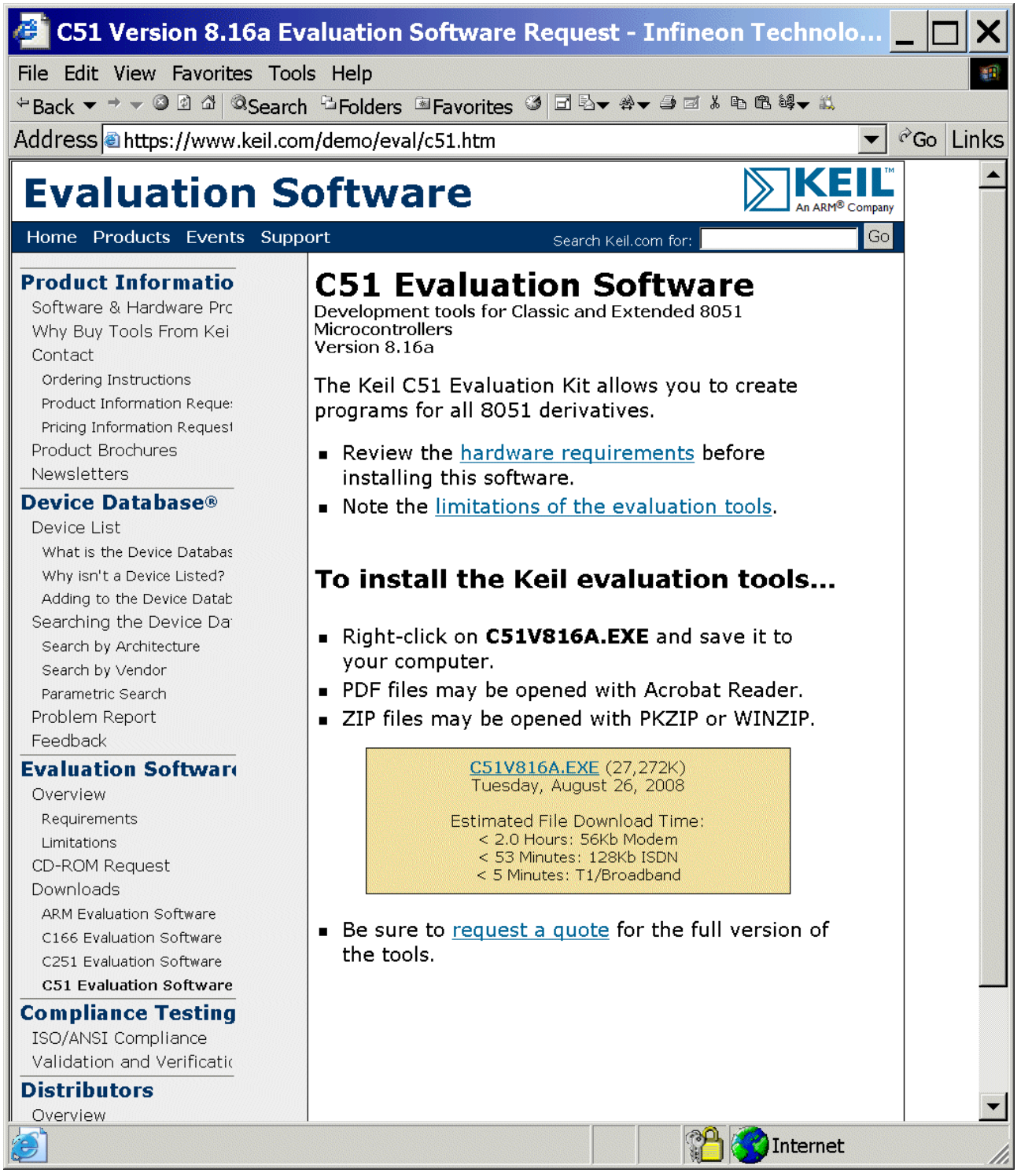
Save changes?

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

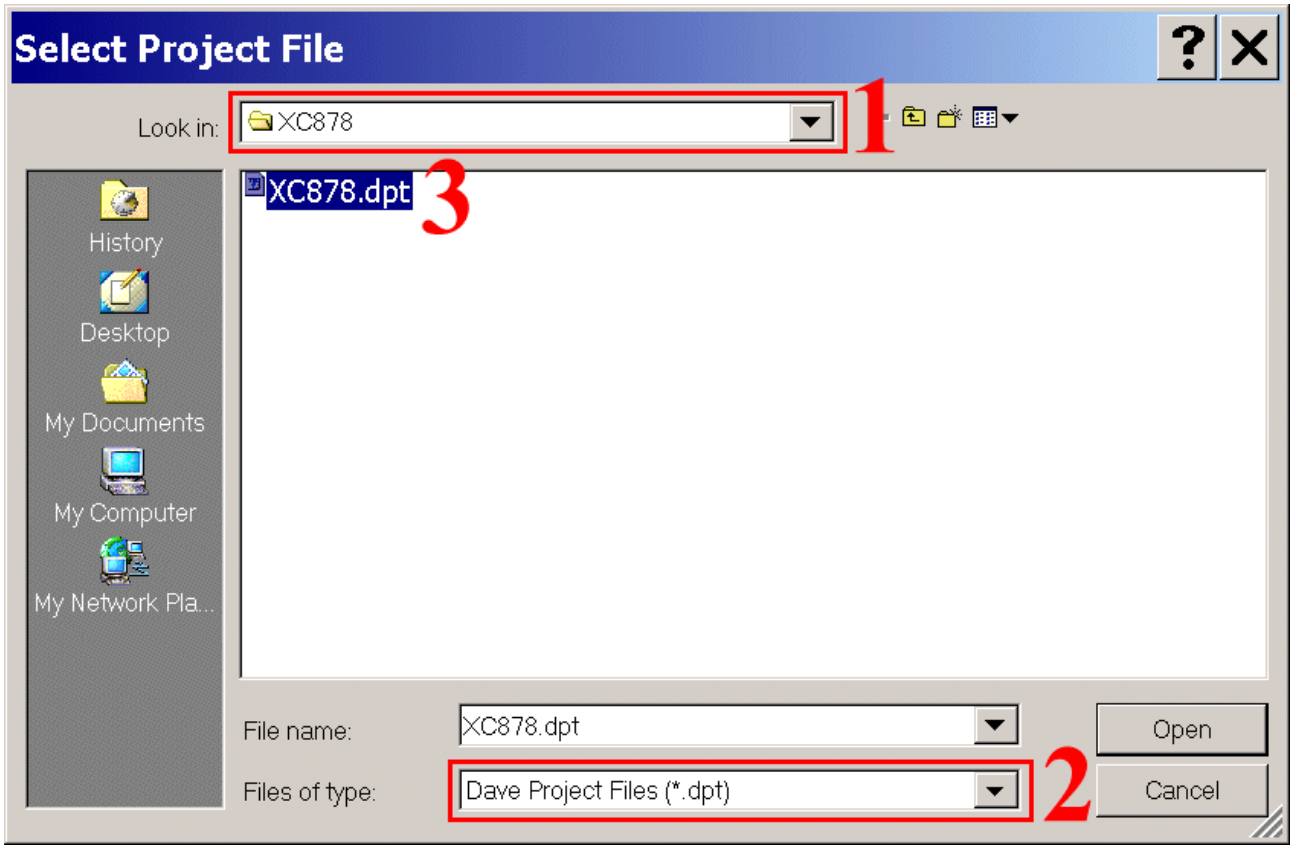


Execute C51V816A.EXE (- or any higher version)

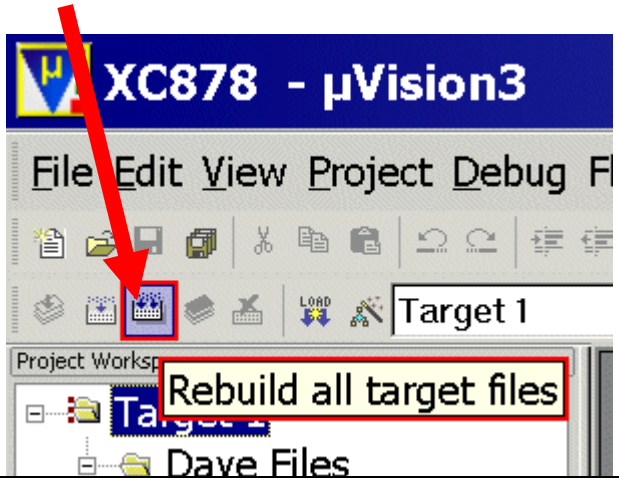
Start Keil µVision3 and open the DAvE Project:

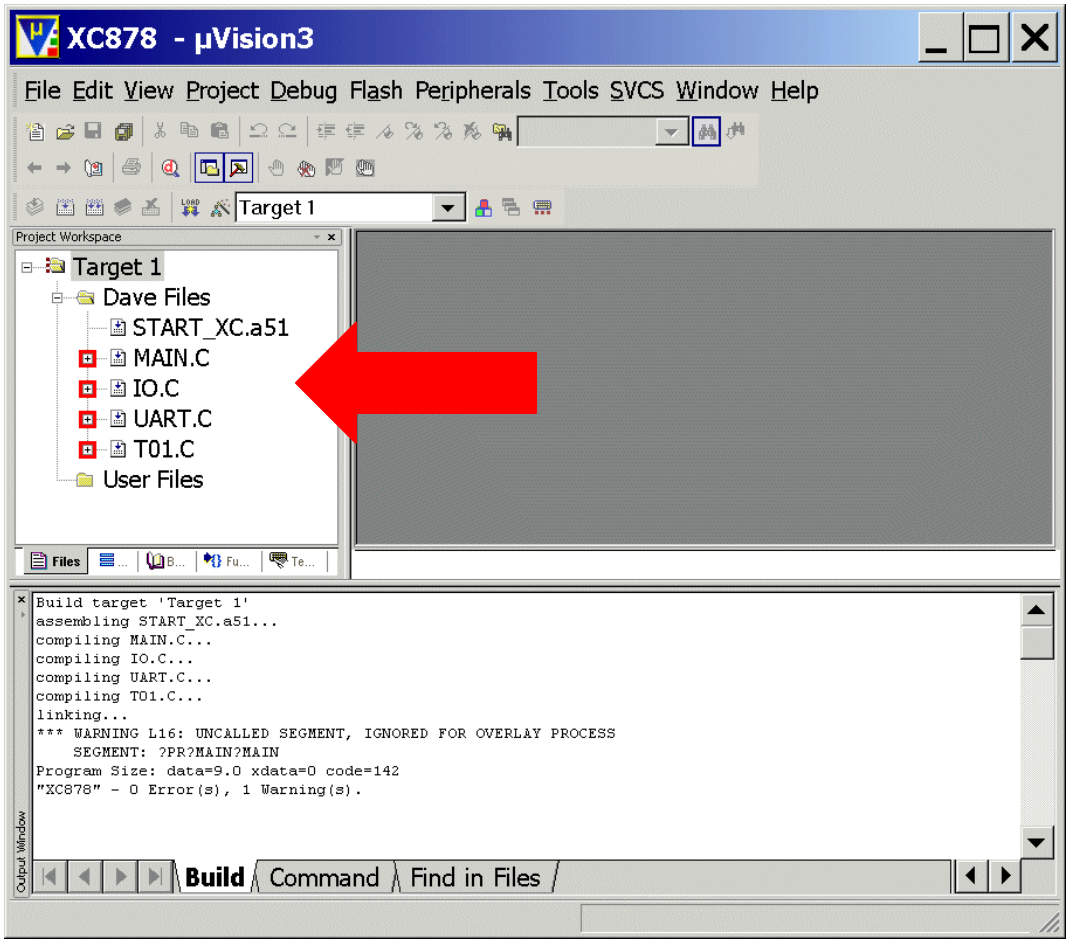
If you see an open project – close it: Project - Close Project

Project - Open Project
Select Project File: Look in: choose C:\XC878 (1)
Select Project File: Files of type: select Dave Project Files (2)
Choose/click XC878.dpt (3)



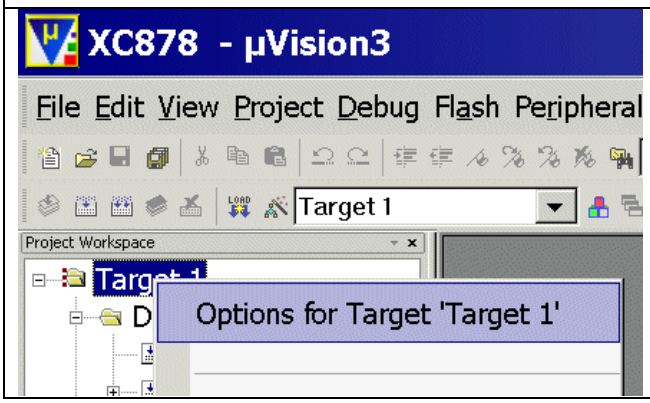
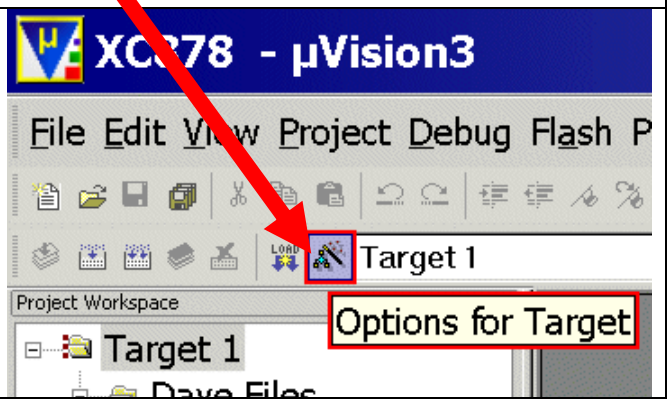
Click Open

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

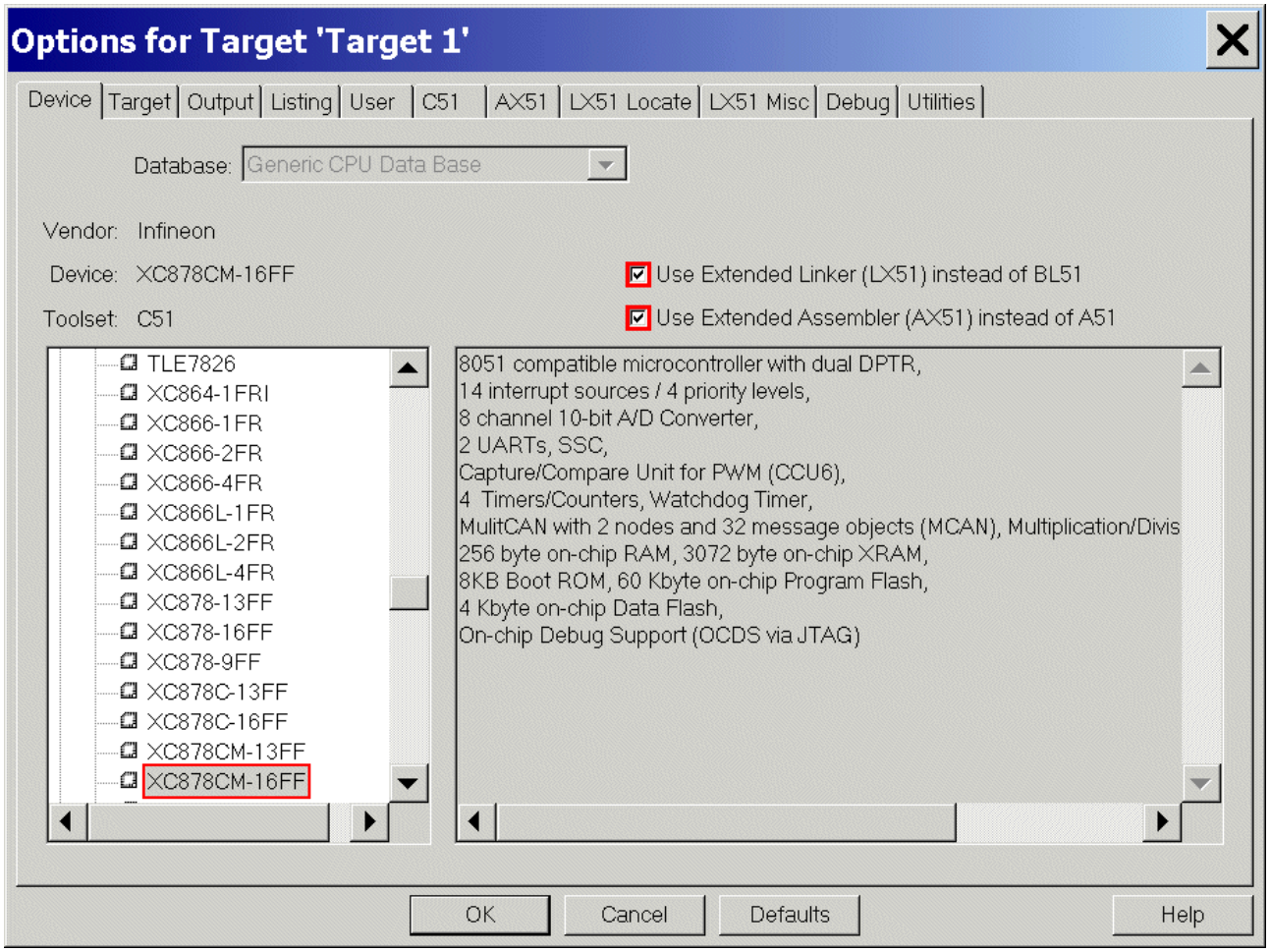


Note:
This step generates a makefile and shows the include files.

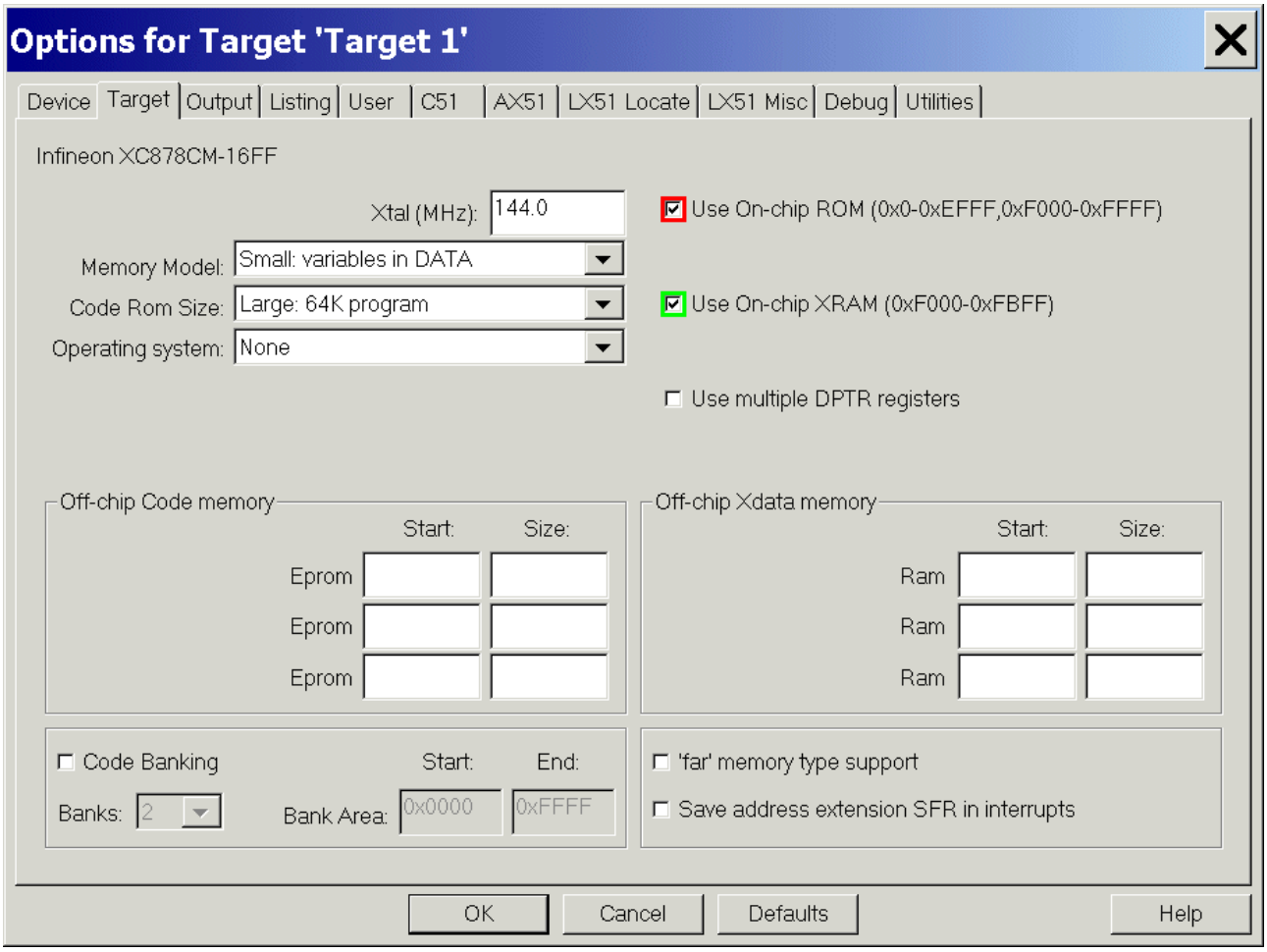
Configure Compiler, Assembler, Linker, Locater, Hex-Converter, Build Control, Simulator, Debugger and Utilities:

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

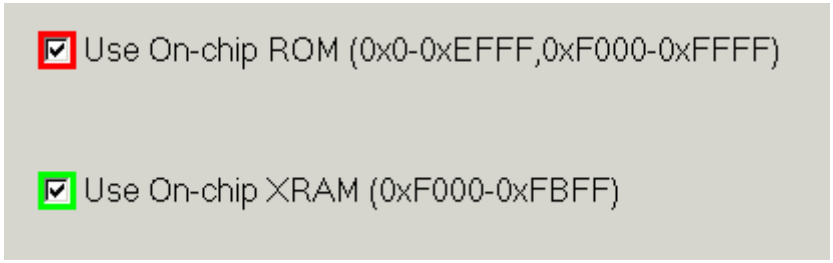
Options for Target ‘Target 1’: Device: check XC878-16FF
Options for Target ‘Target 1’: Device: tick ✓ Use Extended Linker (LX51)
Options for Target ‘Target 1’: Device: tick ✓ Use Extended Assembler (AX51)

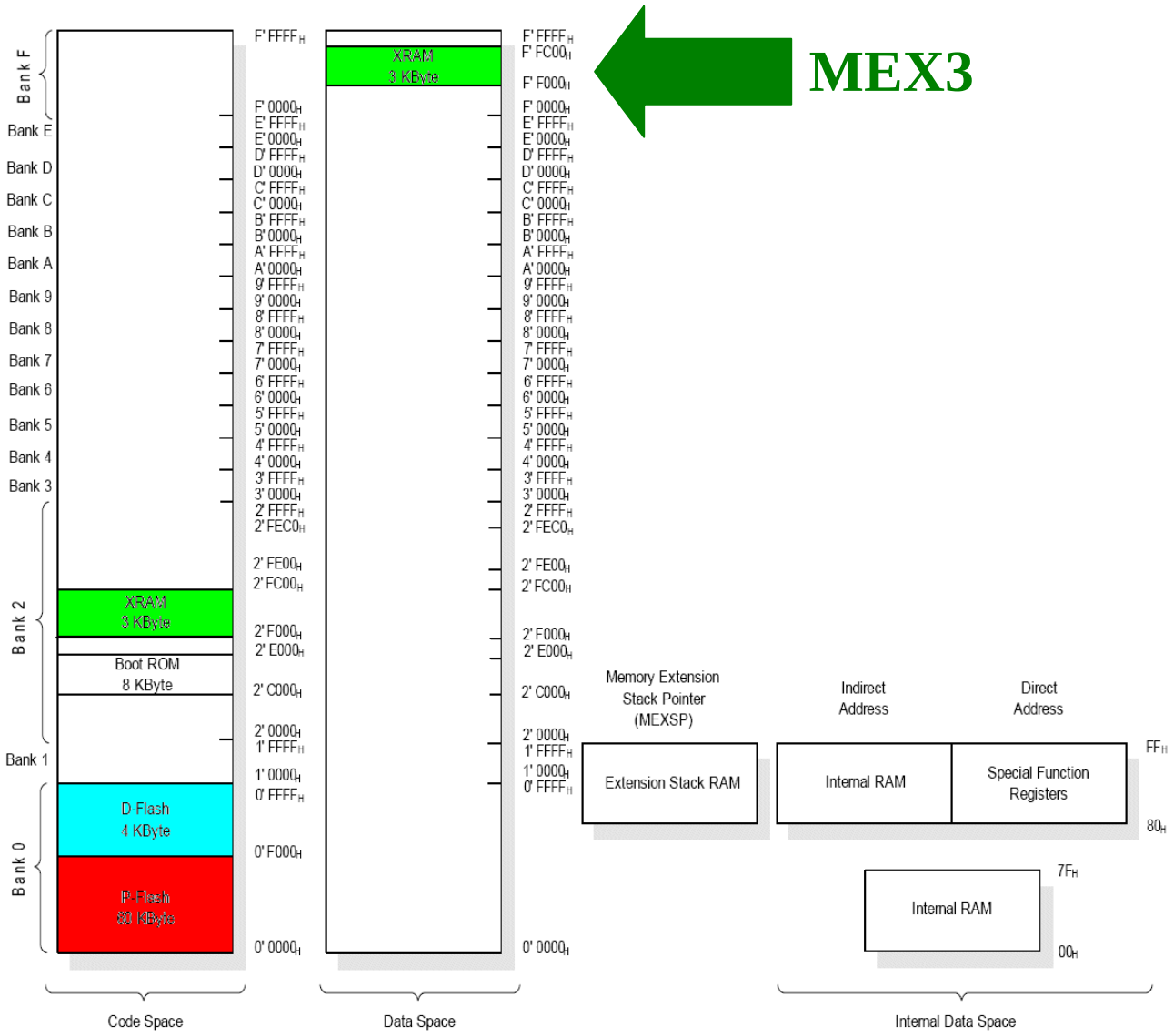


Target: tick ✓ Use On-chip ROM & Target: tick ✓ Use On-chip XRAM



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





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):

C:\XC878\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 ;-------------------------------------------------------------------------------
```

Text Editor | Configuration Wizard

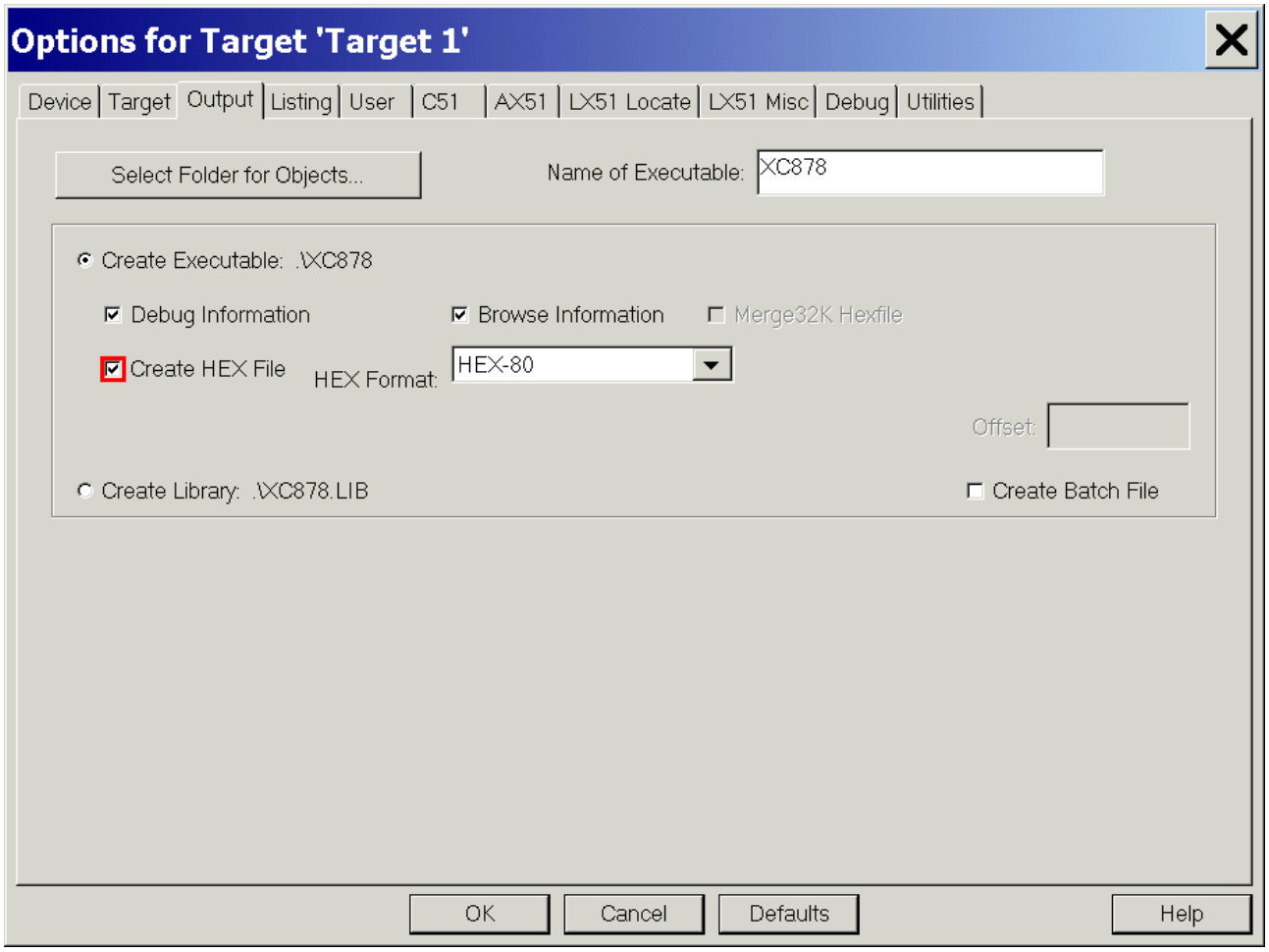
C:\XC878\START_XC.a51

```
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
```

MEX3

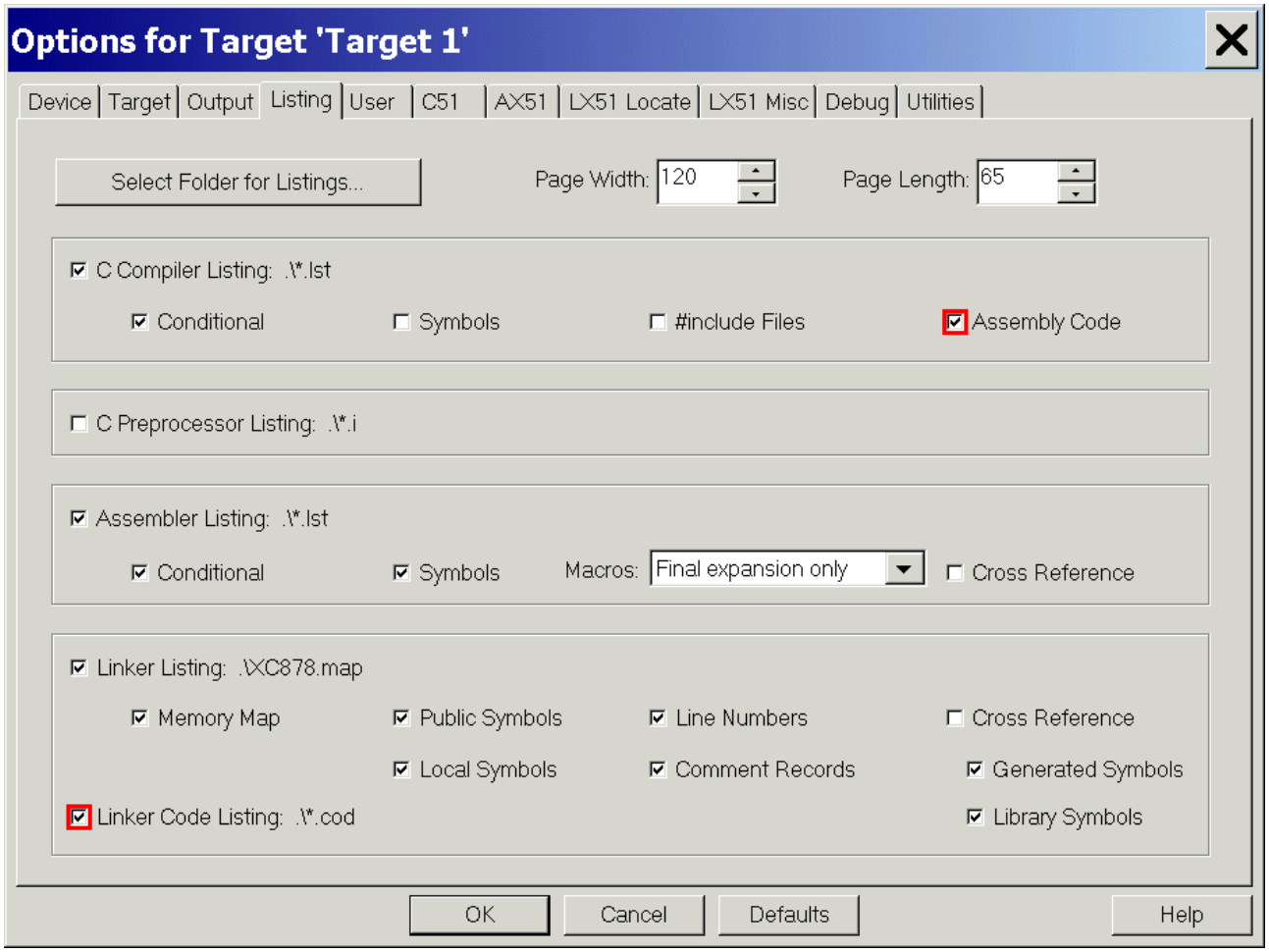
Text Editor | Configuration Wizard

Output: tick ✓ Create HEX File



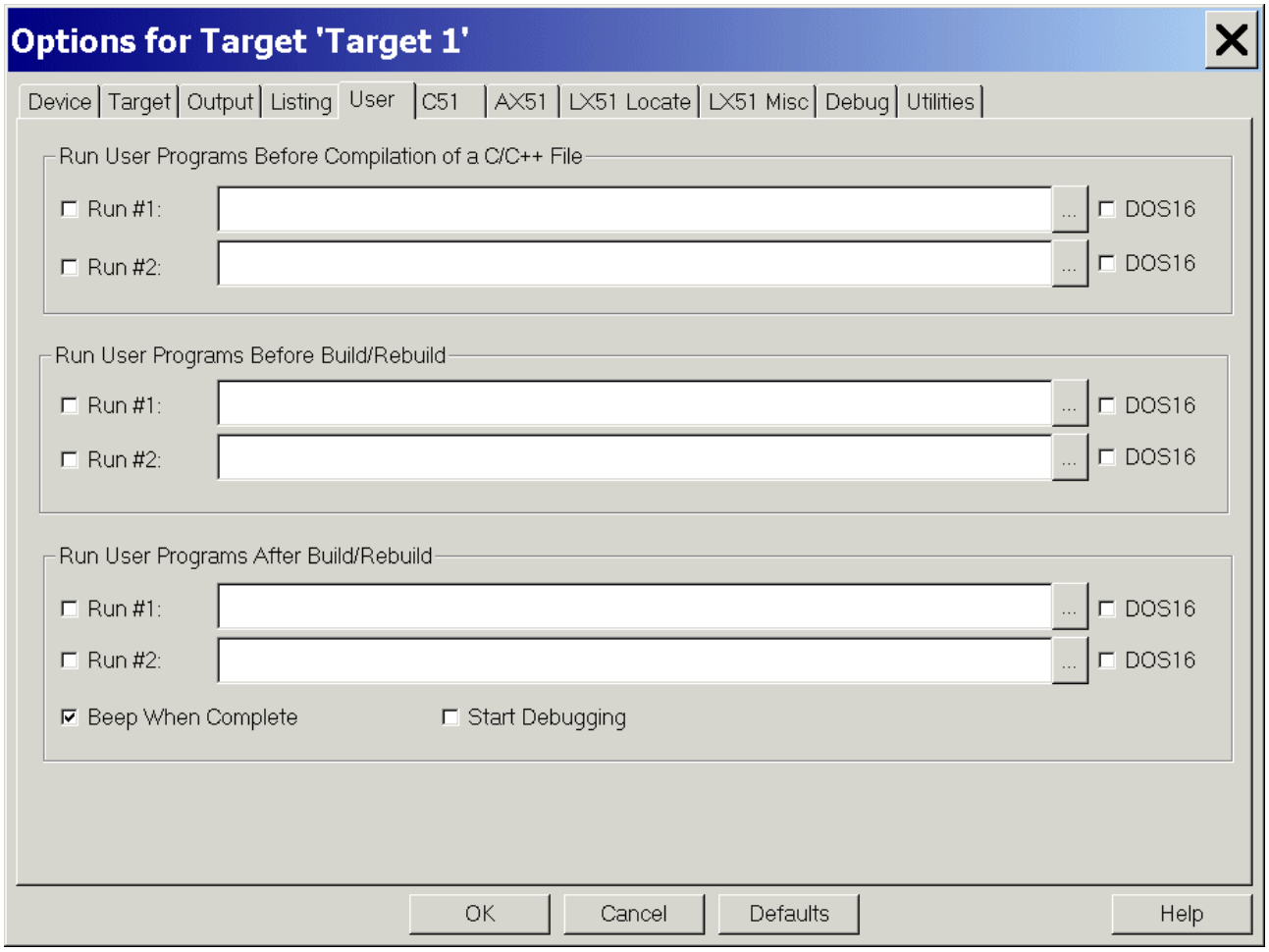
Note:
The HEX-File could be used while working with the program XC800_FLOAD for OnChip-Flash-Programming via RS232-interface [Bootstrap Loader (BSL) Mode via UART].

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

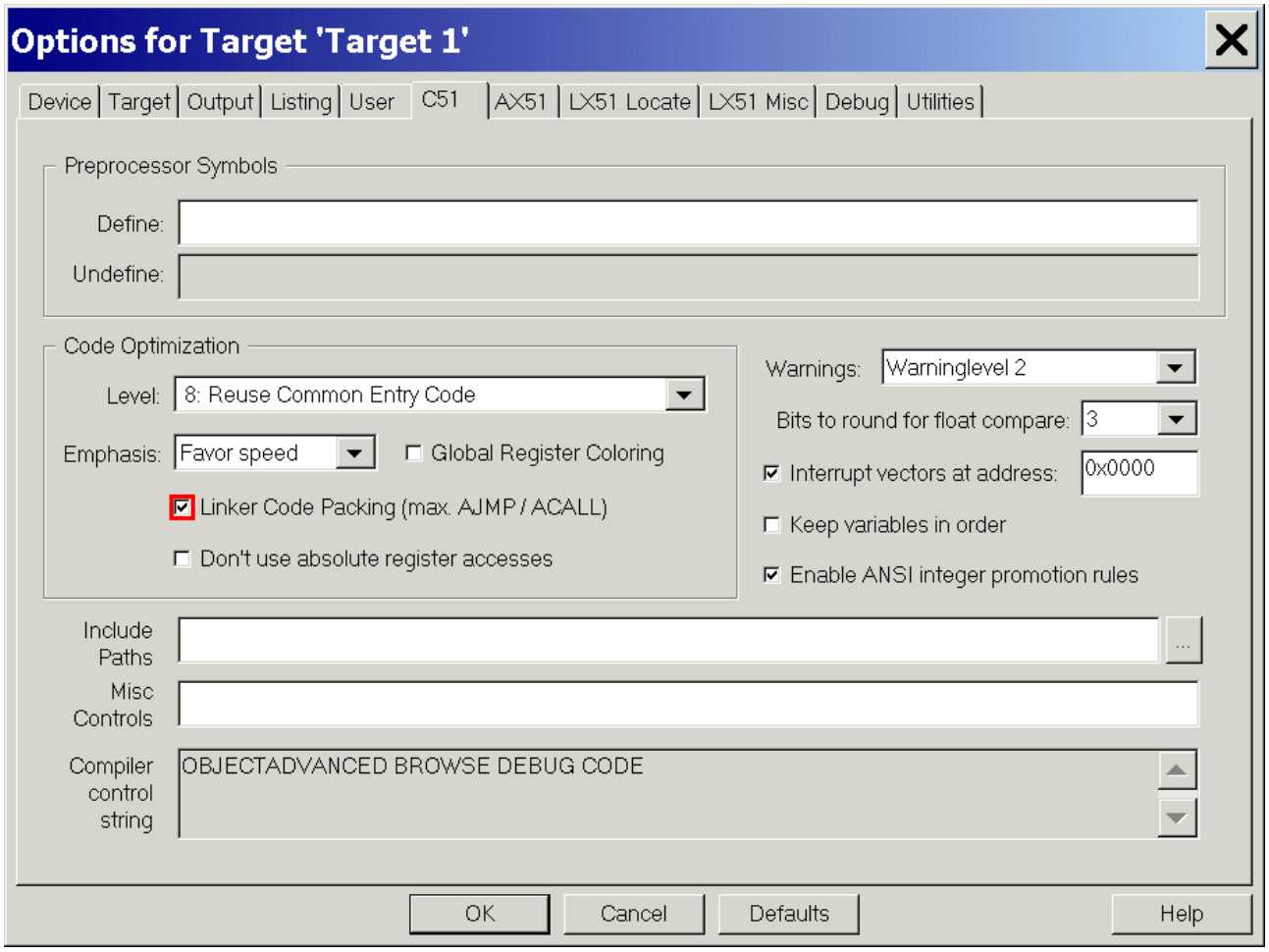


Note:
With the cod-file you can do the following:
1.) position the mouse on the source code you are interested in
2.) click right mouse button and select Open Linker COD File
3.) see the result: Assembler-Code of your C-Source-Code

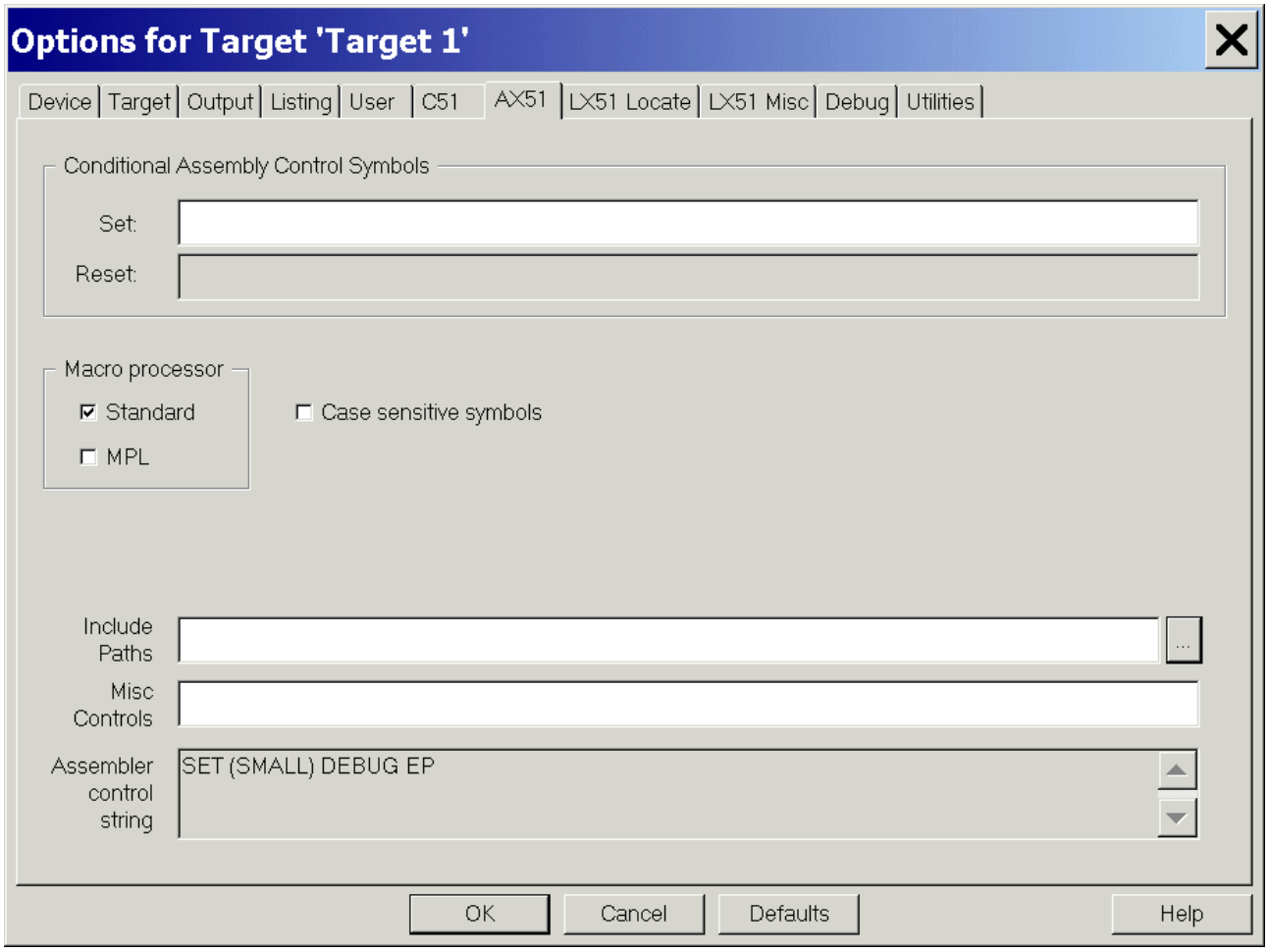
User: (do nothing)



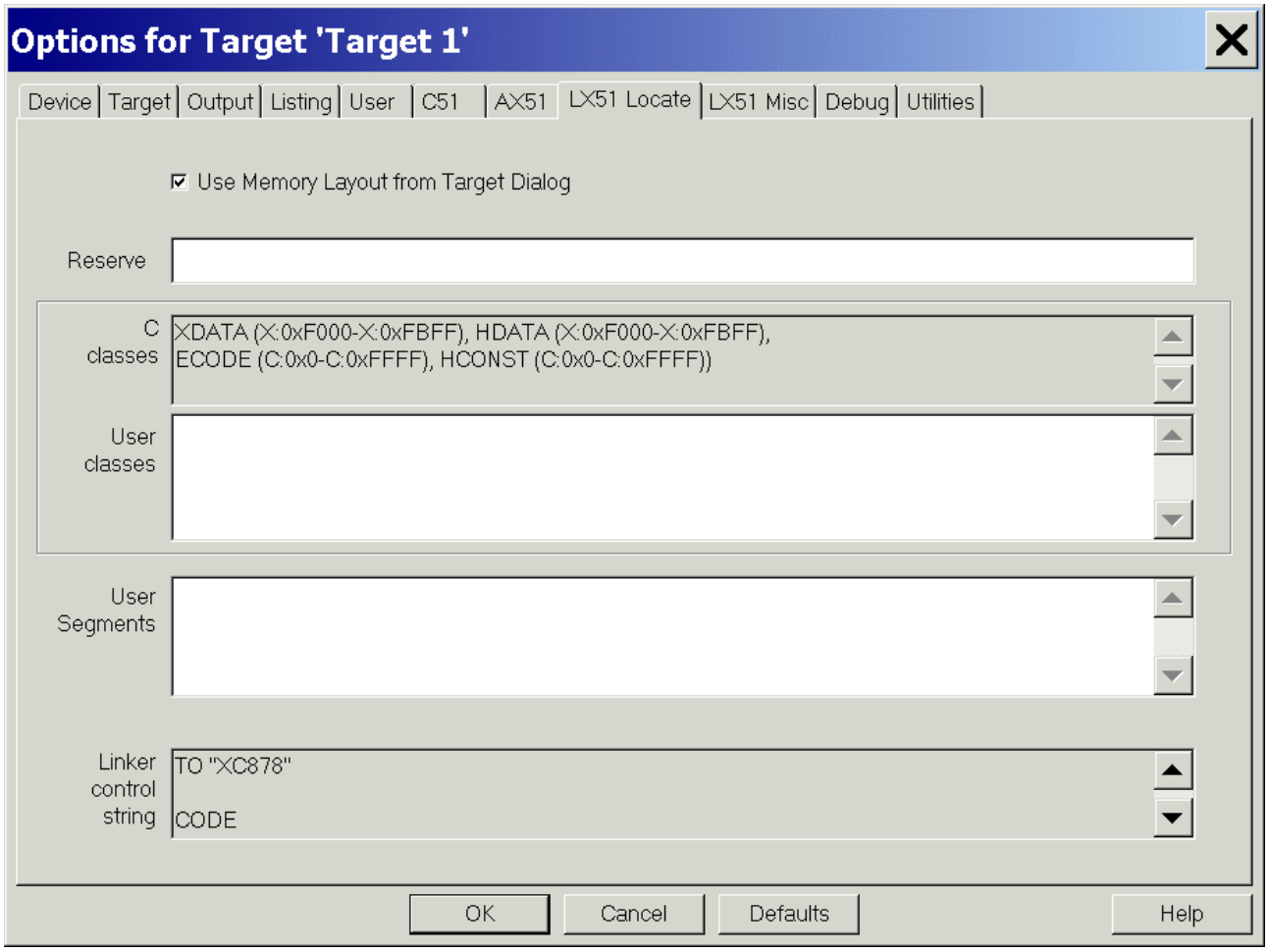
C51: Code Optimization: click ✓ Linker Code Packing (max. AJMP/ACALL)



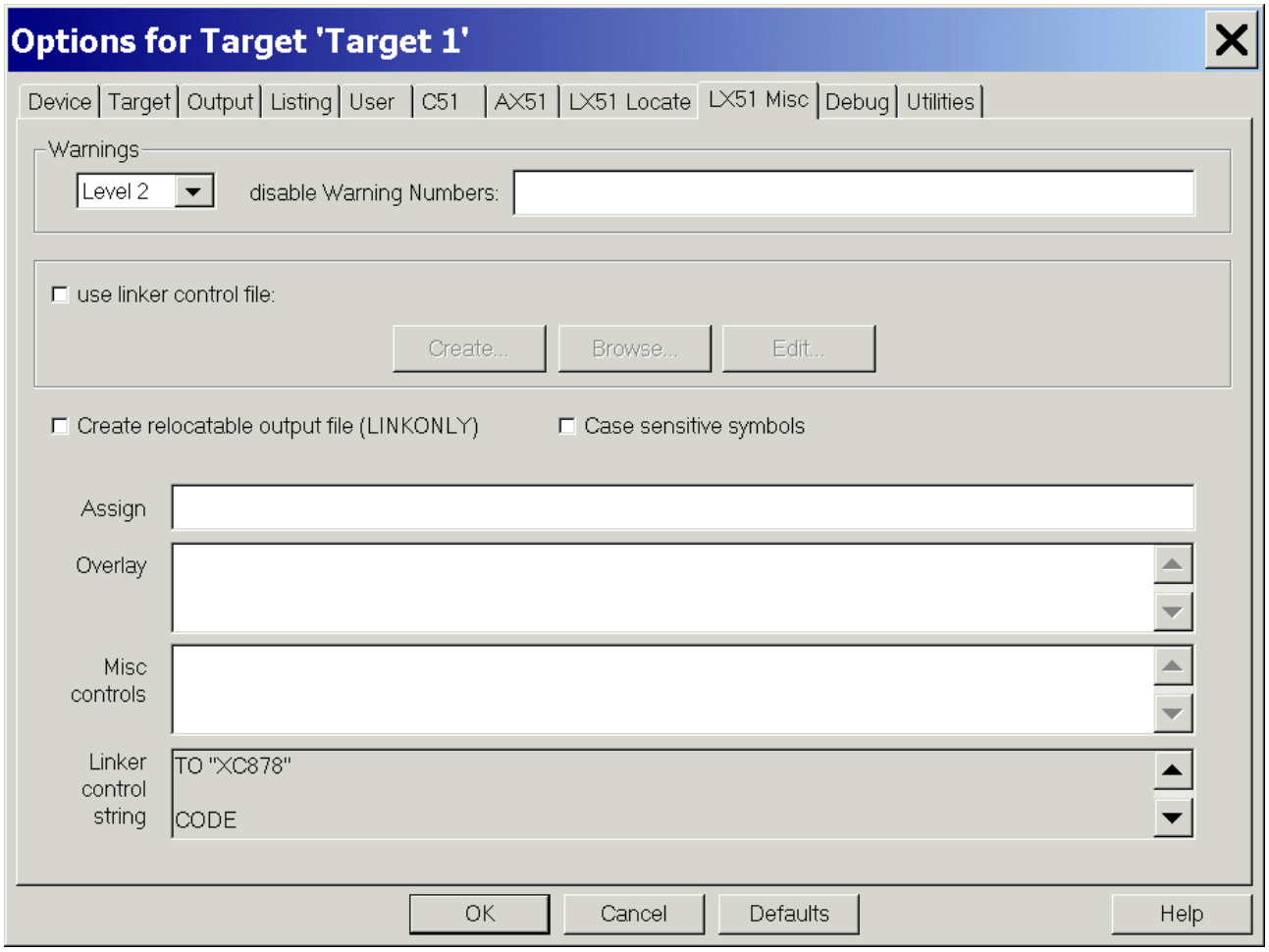
AX51: (do nothing)



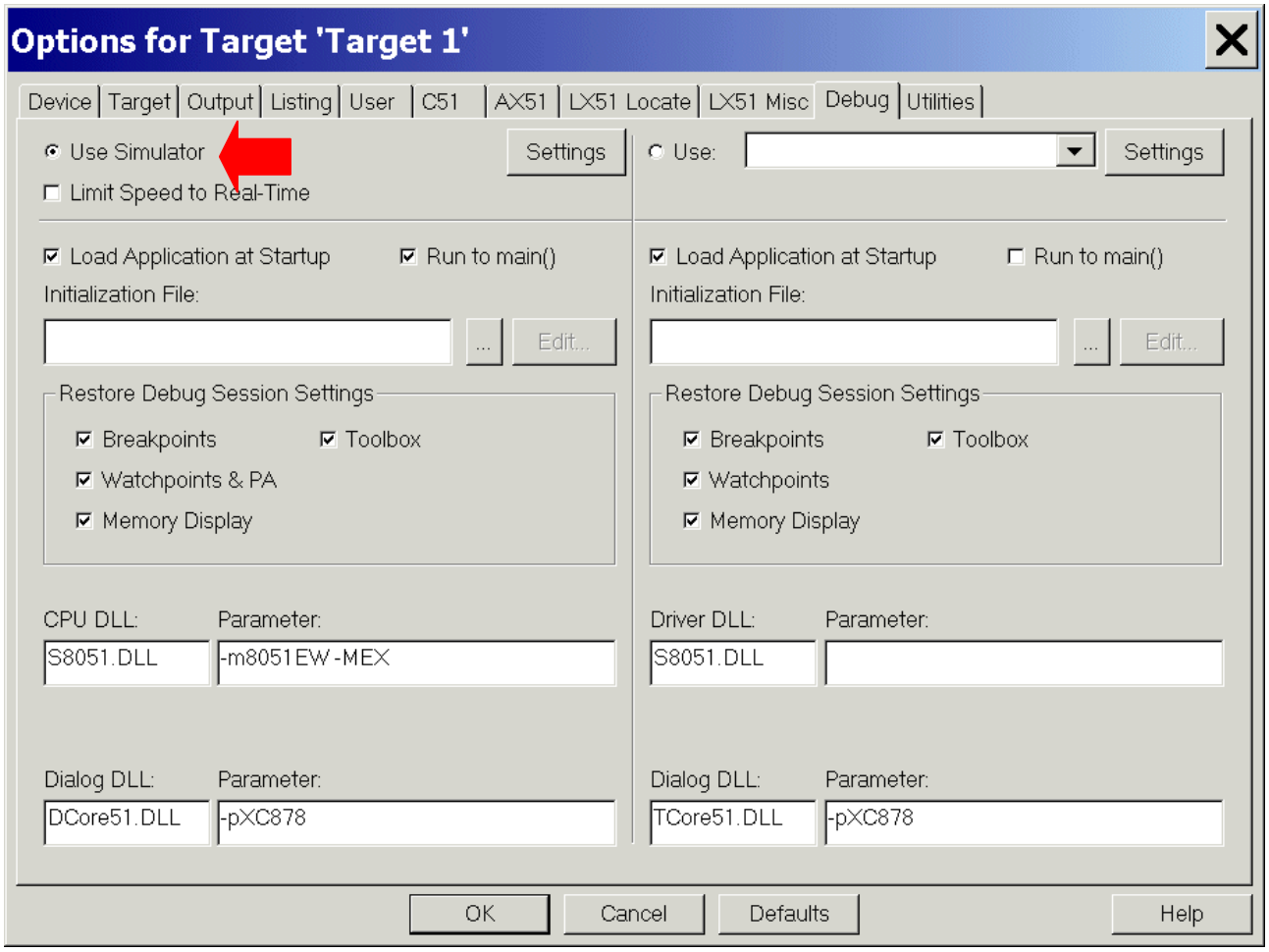
LX51 Locate: (do nothing)

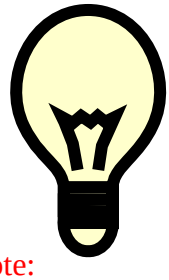


LX51 Misc: (do nothing)



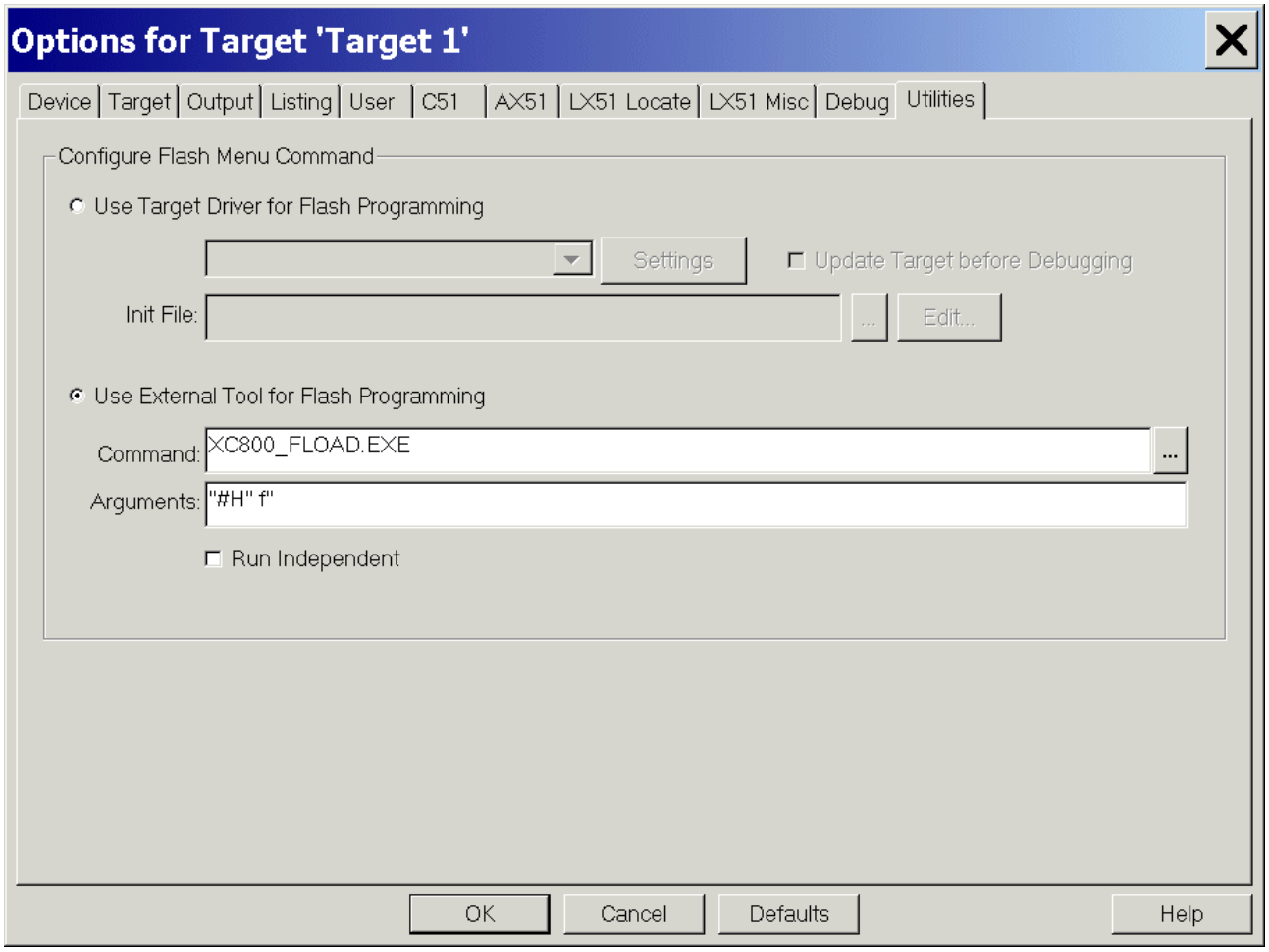
Debug: (do nothing)





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

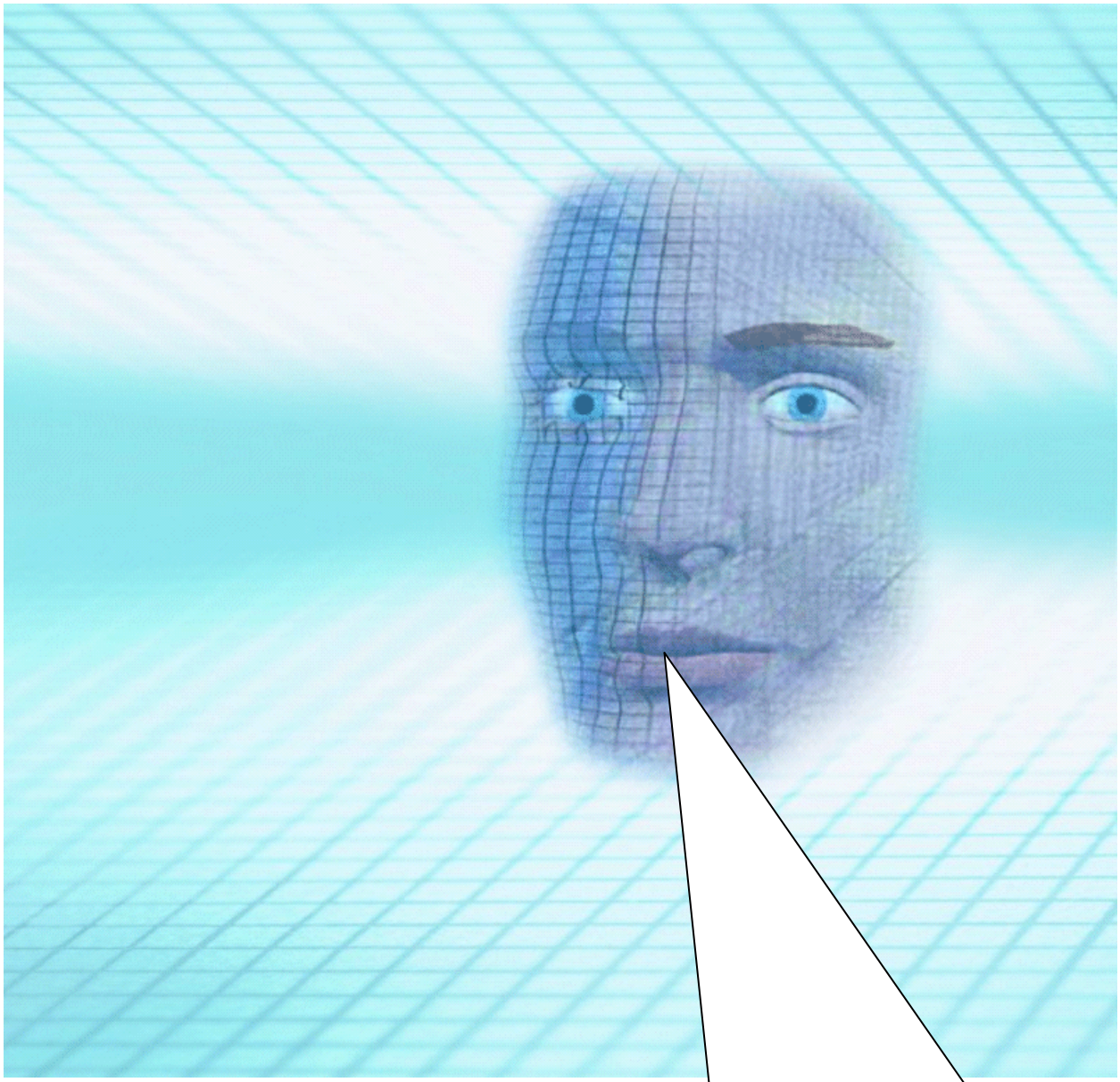
Utilities: (do nothing)



Click OK

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

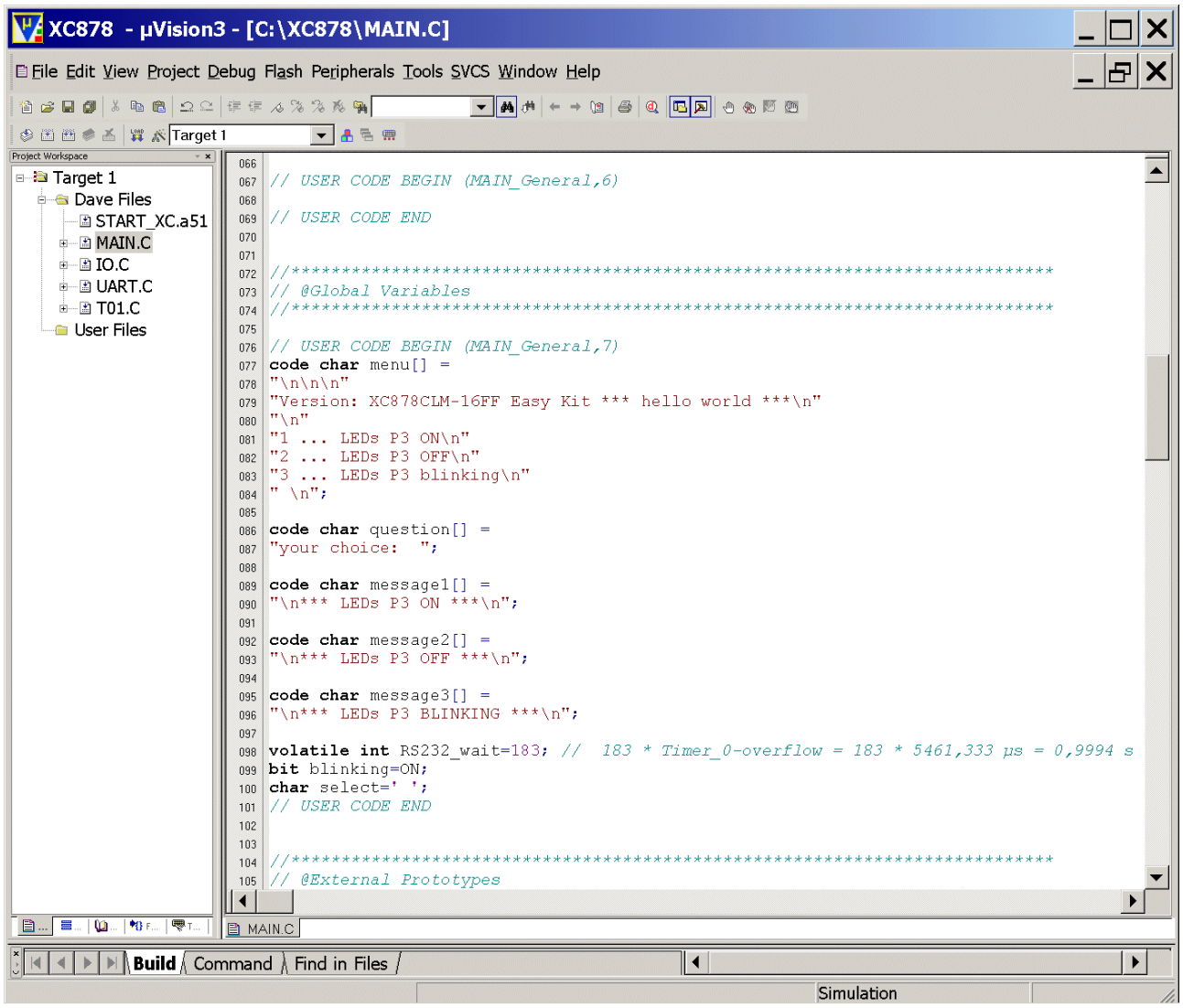
Insert your application specific program:



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!

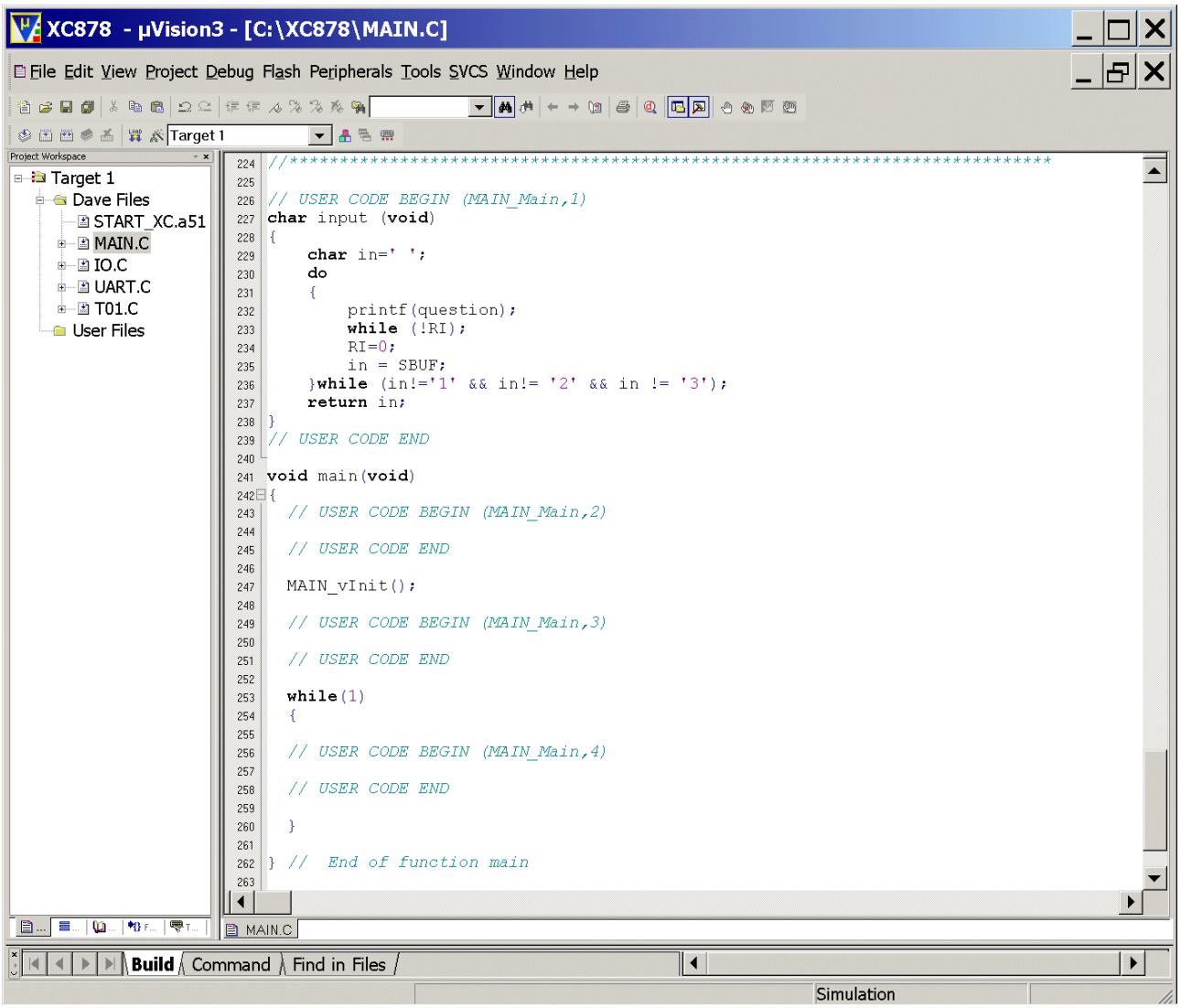
Double click MAIN.C and insert Global Variables:

```
code char menu[] =
"\n\n\n"
"Version: 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=' ';
```


XC878 - µVision3 - [C:\XC878\MAIN.C]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
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: 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=' ';
// USER CODE END
//***
// @External Prototypes
MAIN.C
Build
Command
Find in Files
Simulation

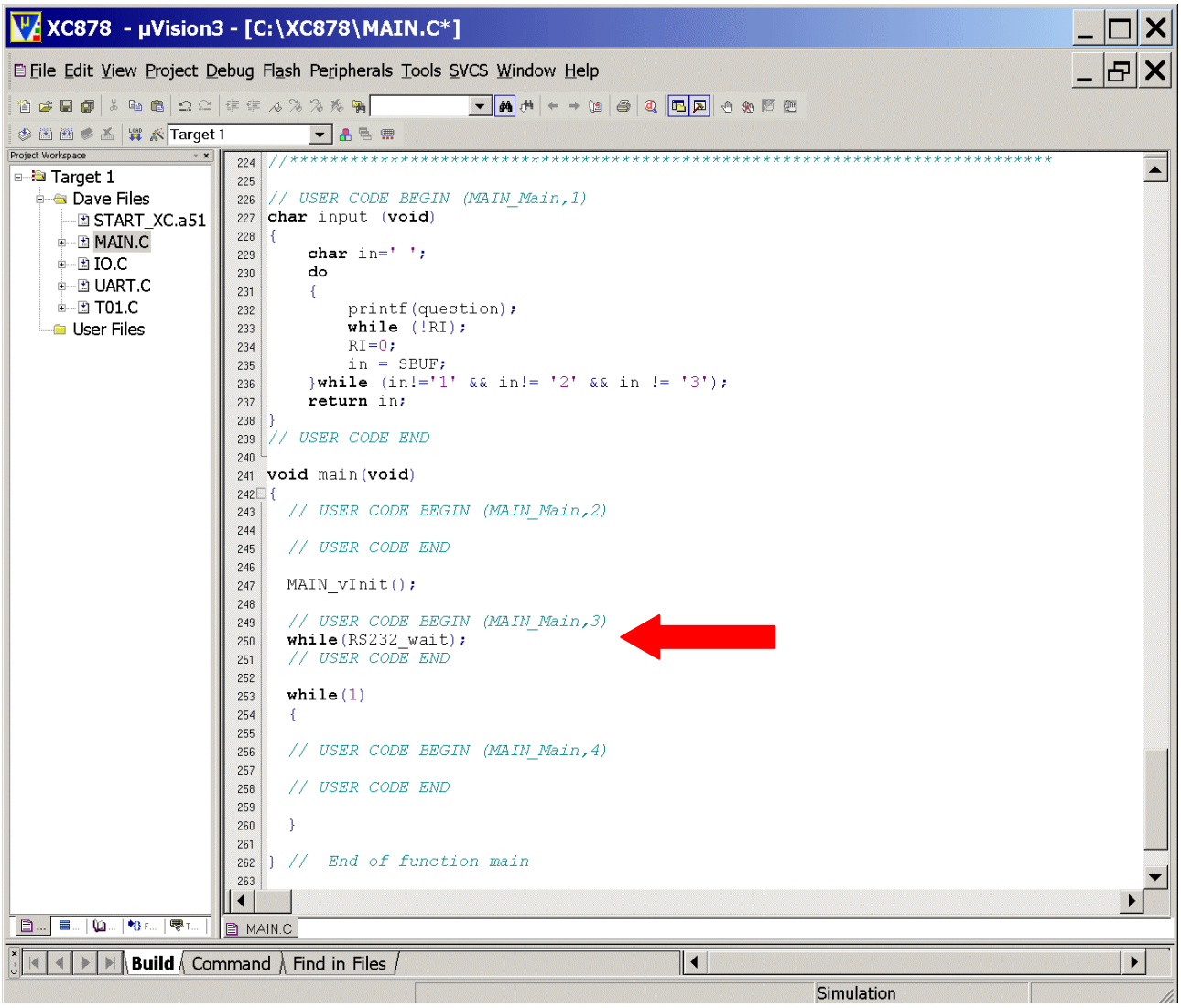
Double click MAIN.C and insert the function input():

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



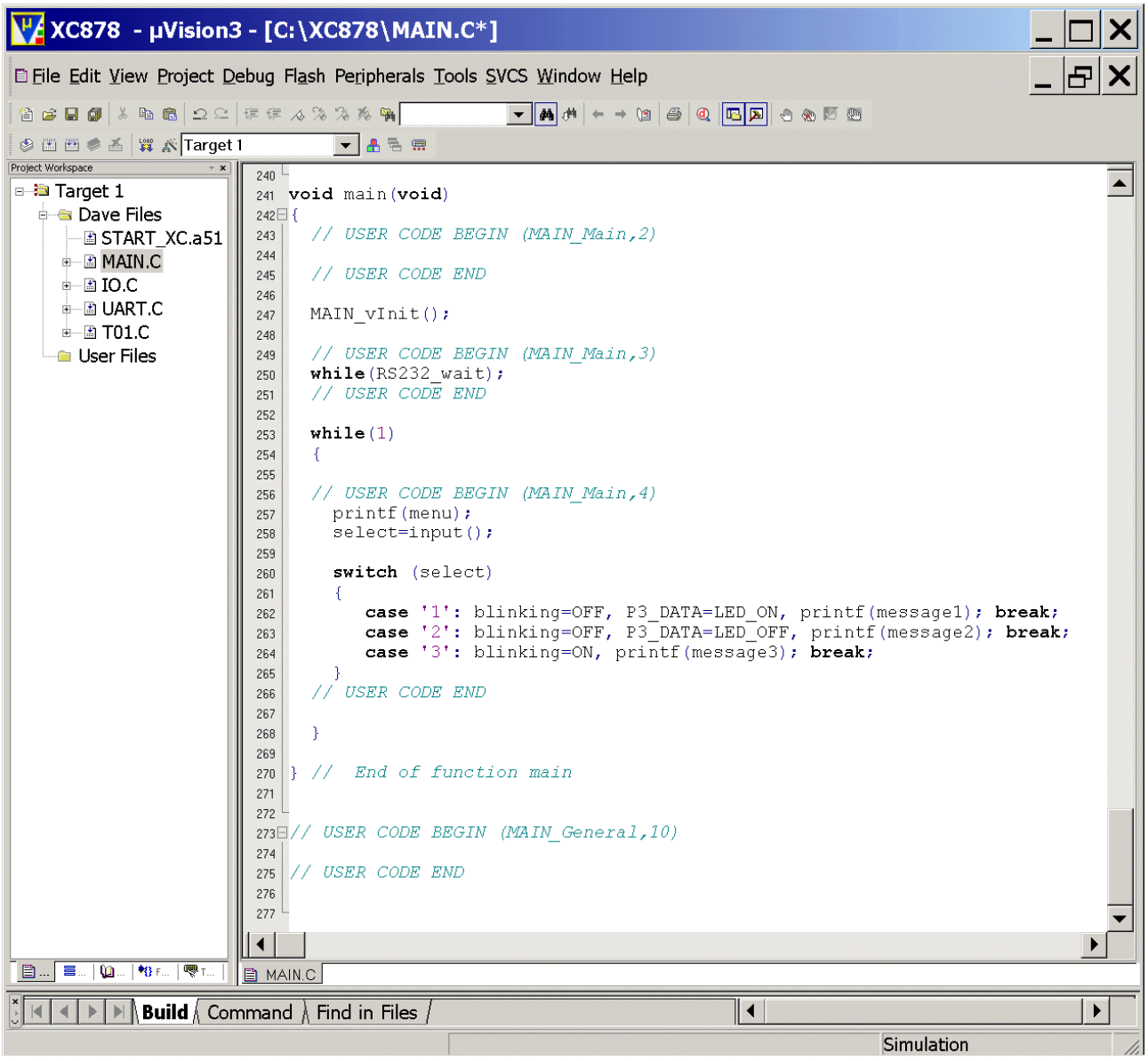
Double click MAIN.C and insert the following code in the main function:

```
while(RS232_wait);
```



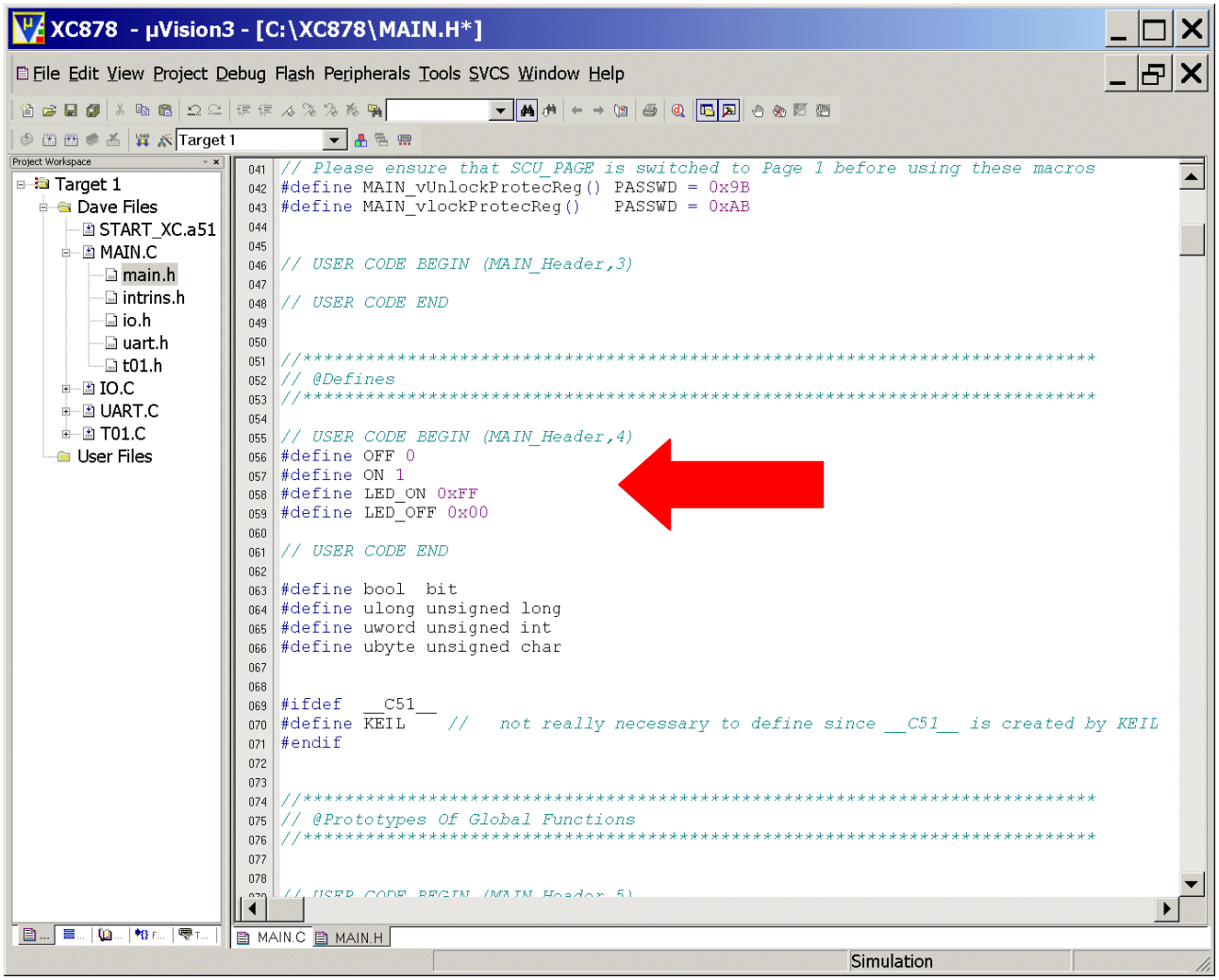
Double click MAIN.C and insert the following code in the main function into the while(1) loop:

```
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;
}
```



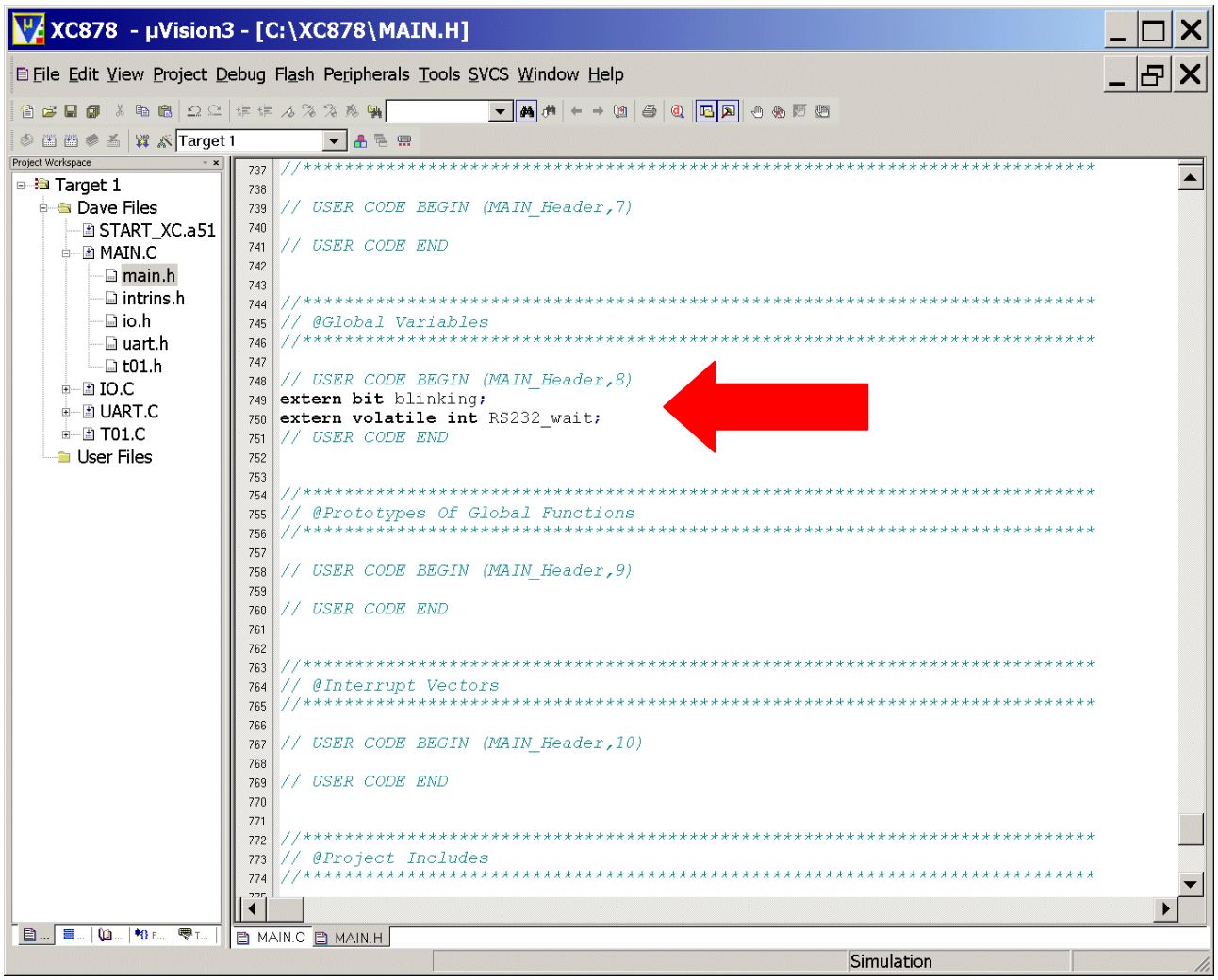
Double click Main.h and insert the following Defines:

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



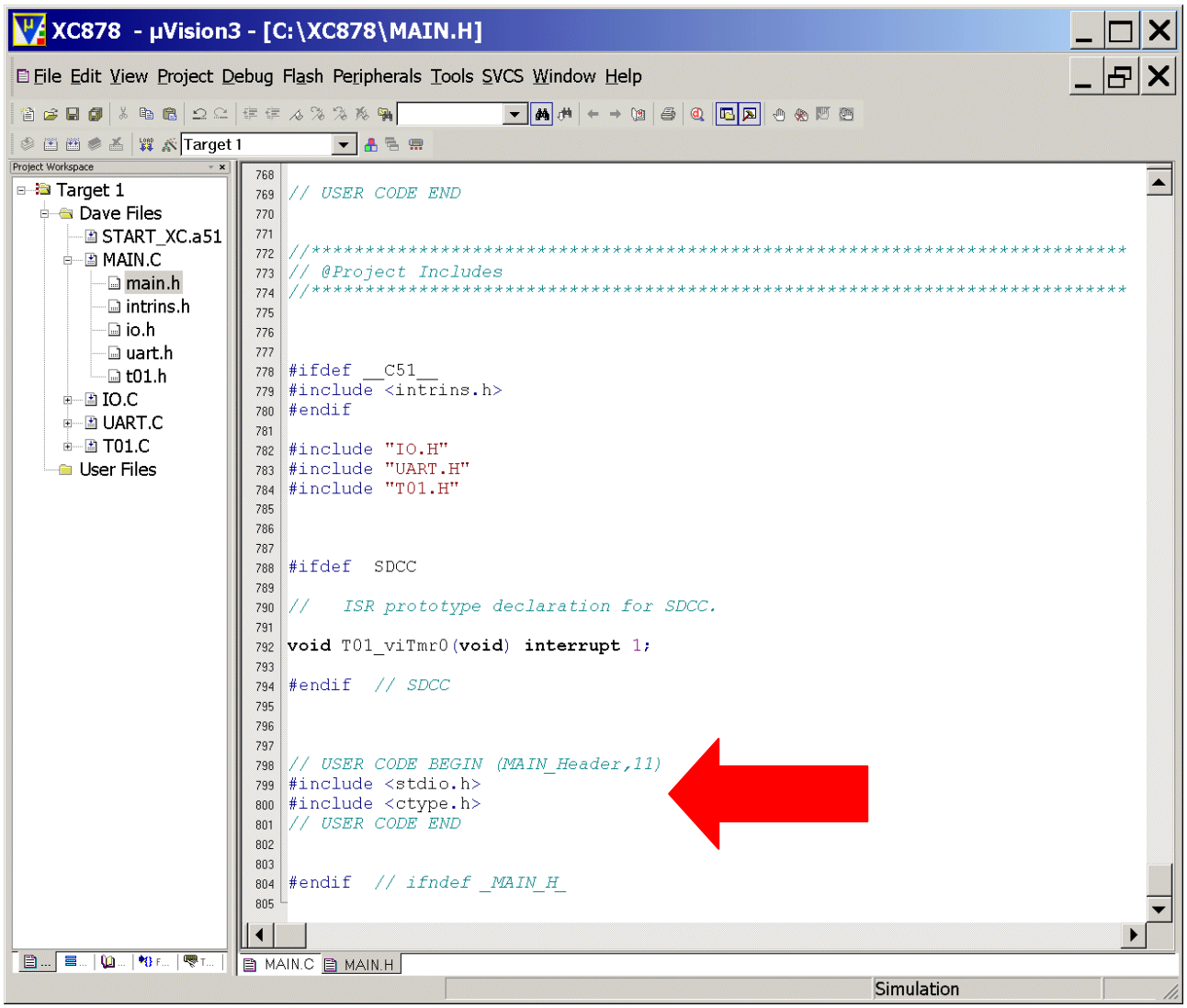
Double click Main.h and insert extern-declarations “Global Variables”:

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



Double click Main.h and insert include files:

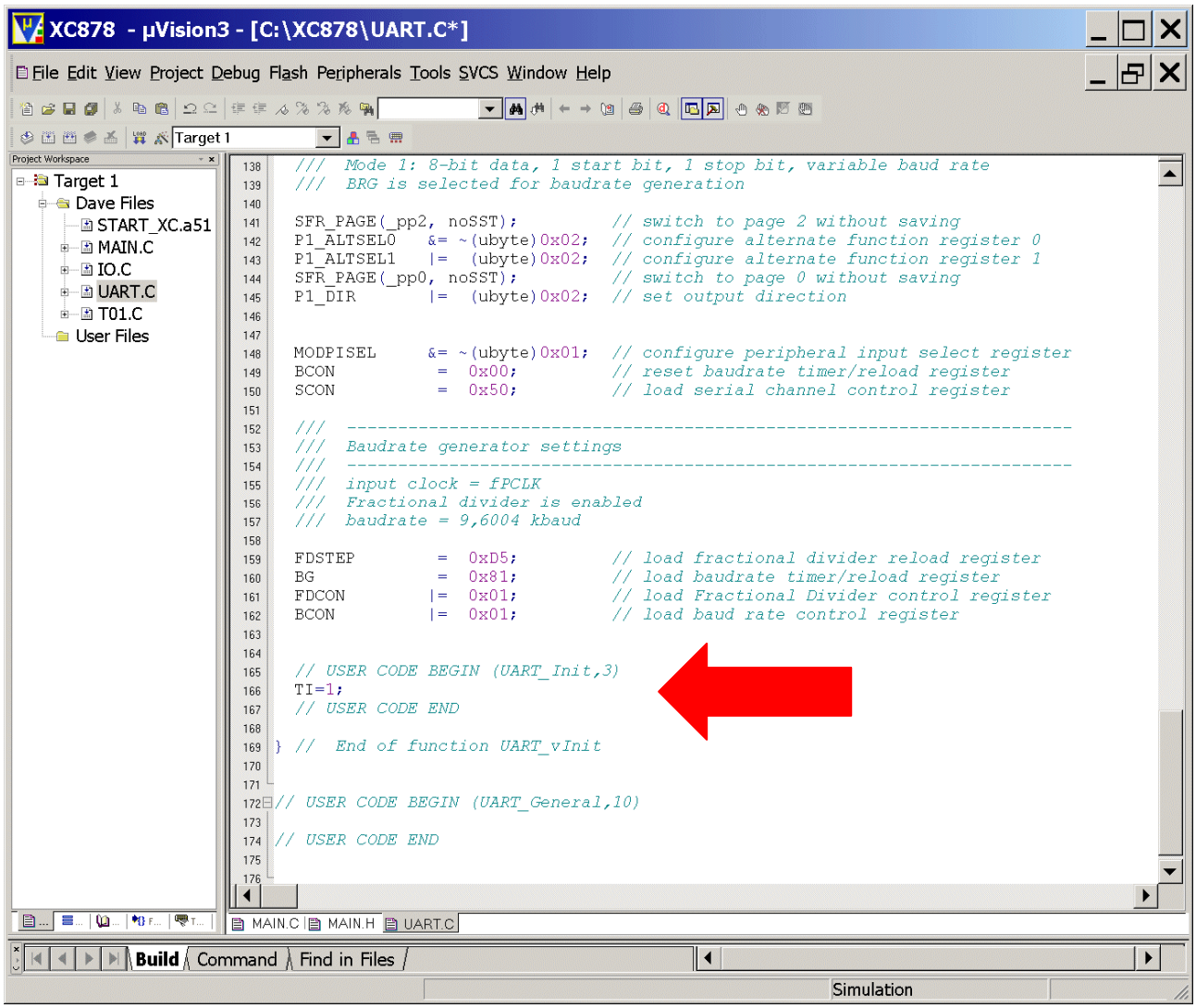
```
#include <stdio.h>
#include <ctype.h>
```



Double click UART.C

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

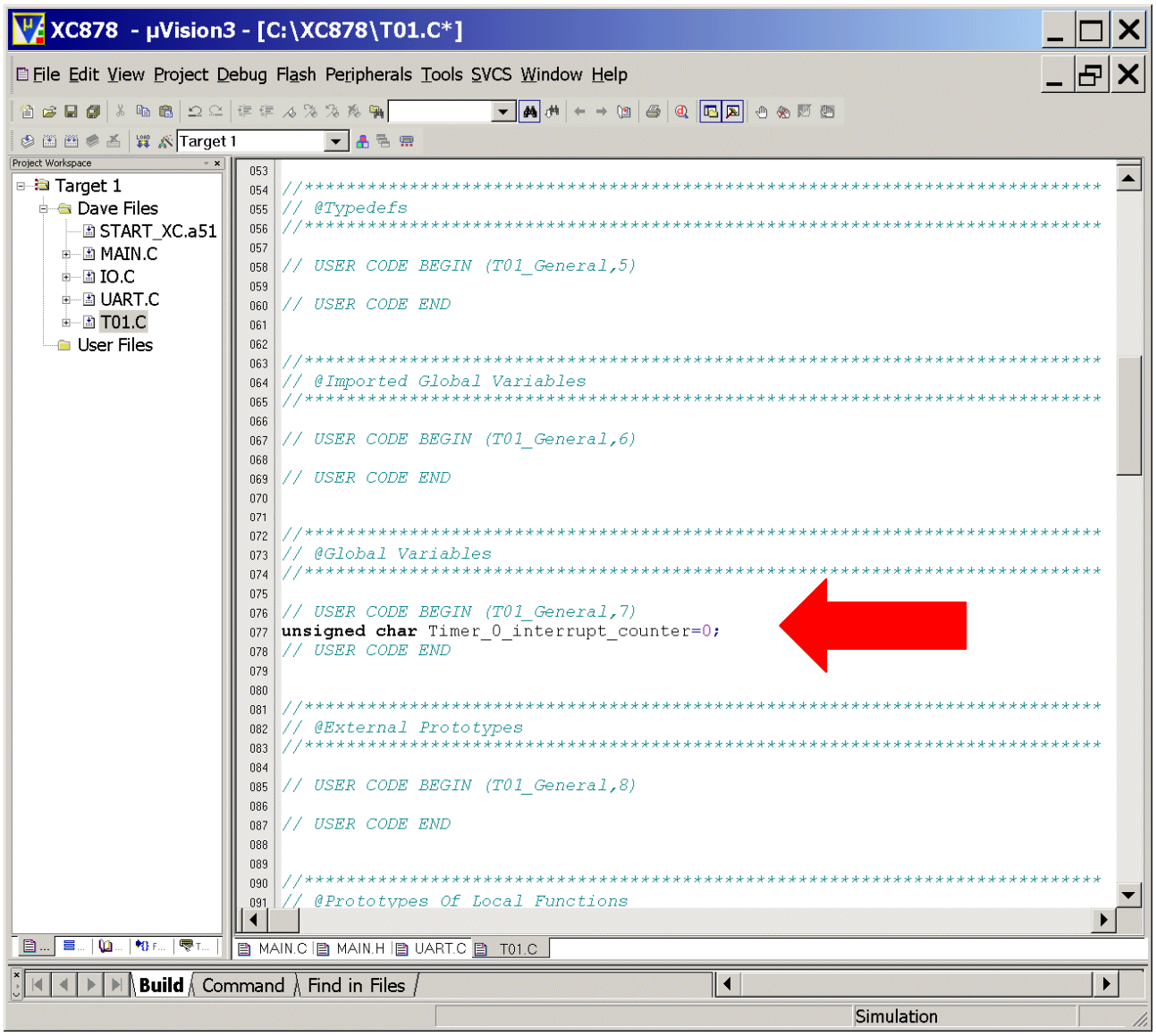
```
TI=1;
```



Double click T01.C

Insert the following global variable:

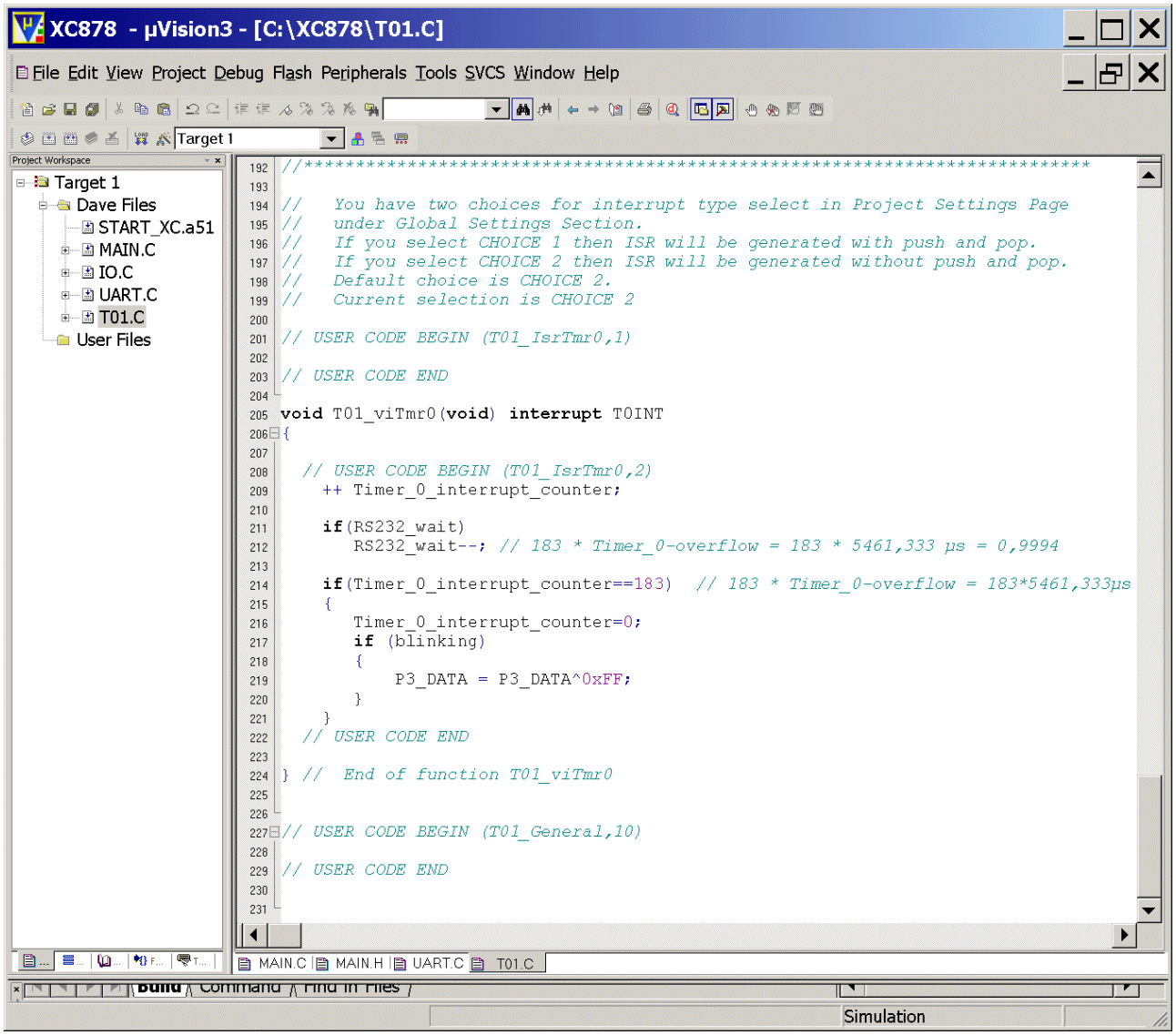
```
unsigned char Timer_0_interrupt_counter=0;
```



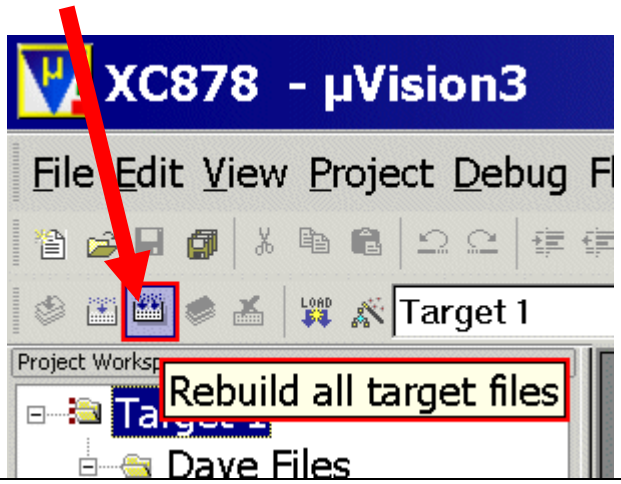
Double click T01.C

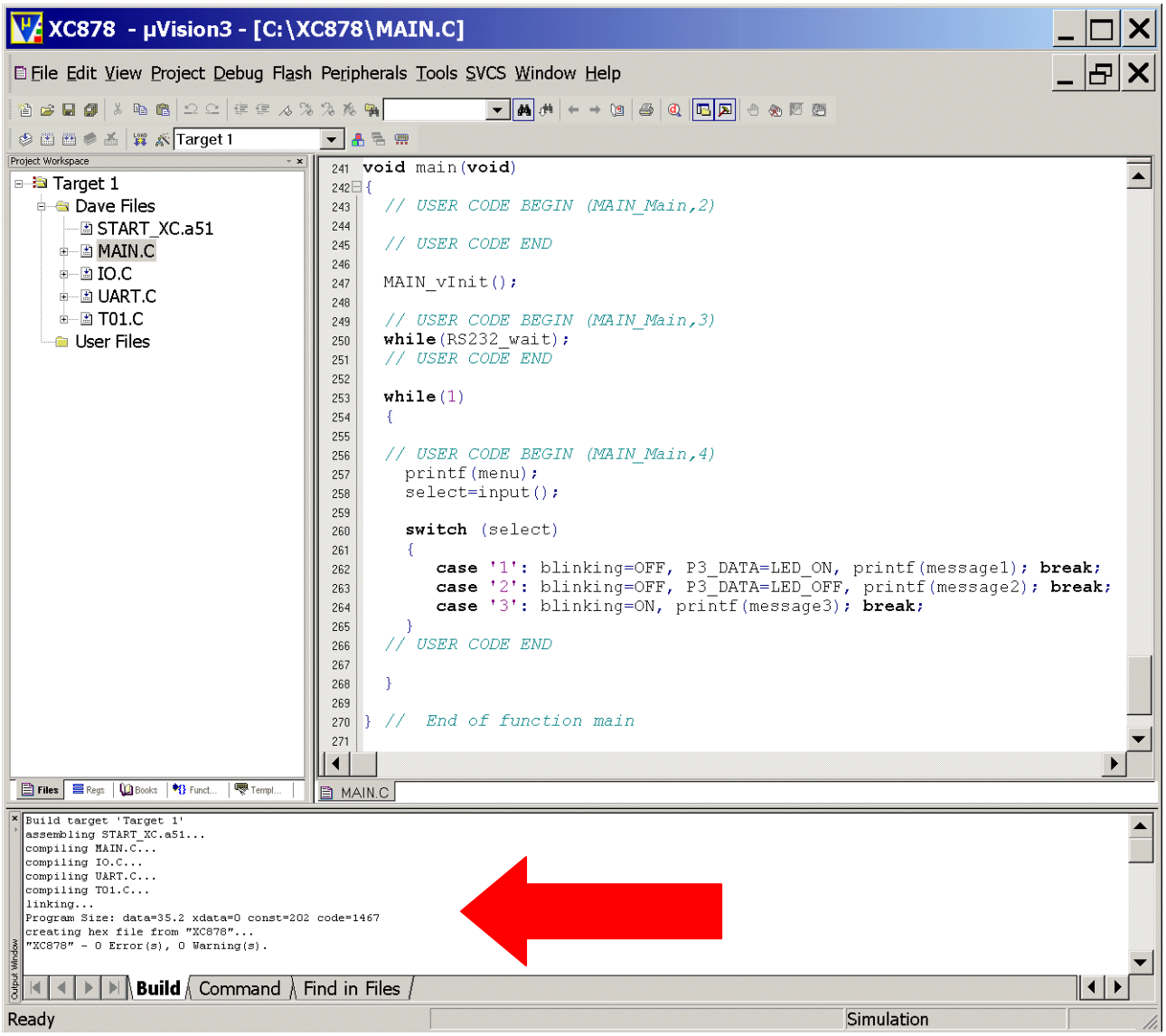
Insert code for T0 interrupt service routine:

```
++ 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:

Project – Rebuild all target files	or	click 



Now we close our project and µVision 3:

Project
Close Project

File

Exit

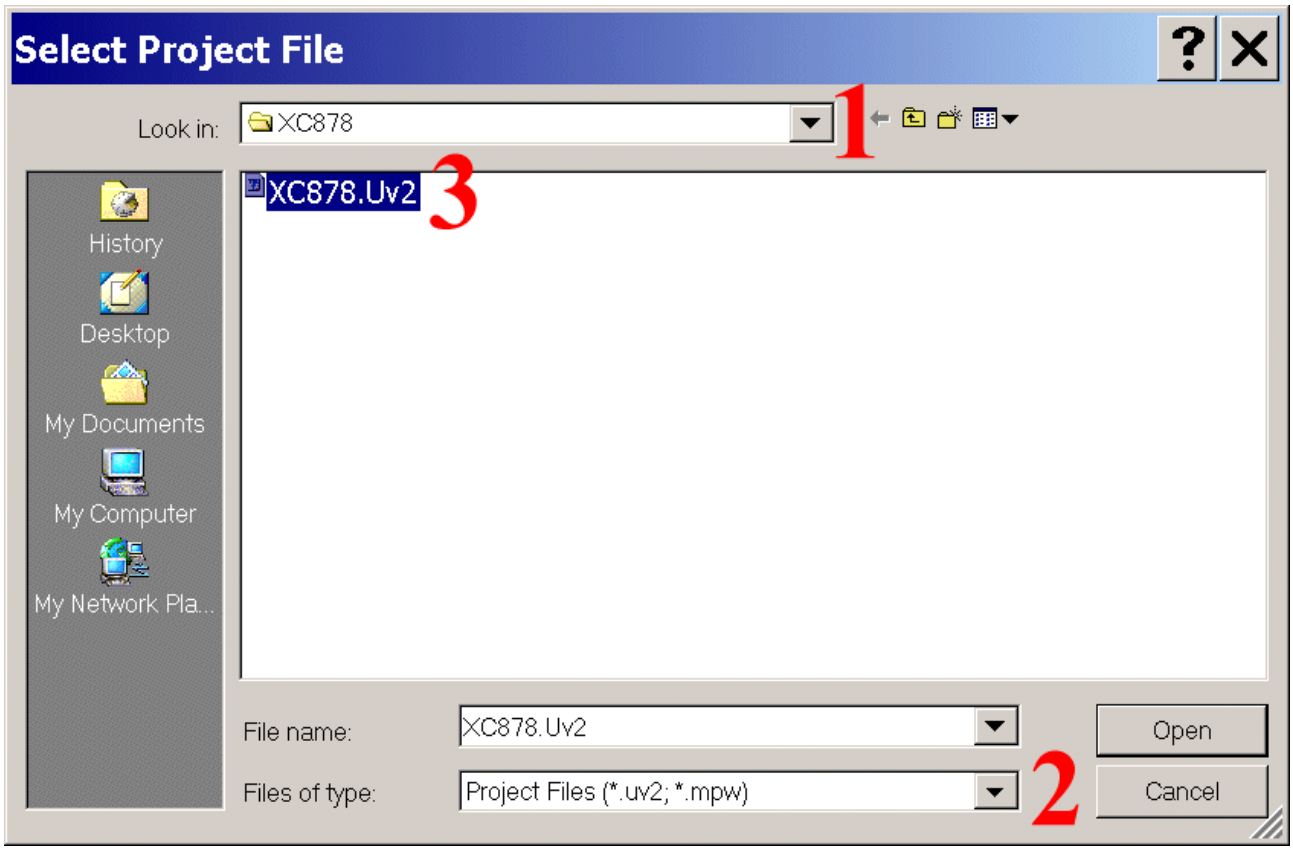
5.) Using the Simulator (first we will test our program with the Simulator):

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:\XC878 (1)
Select Project File: Files of type: select Project Files (*.uv2) (2)
Click XC878.Uv2 (3)

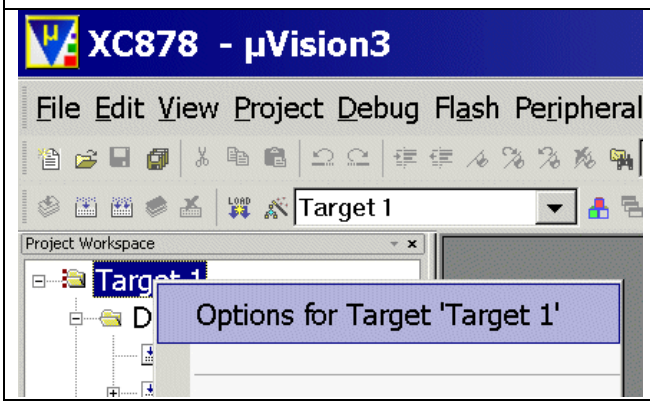
Click Open

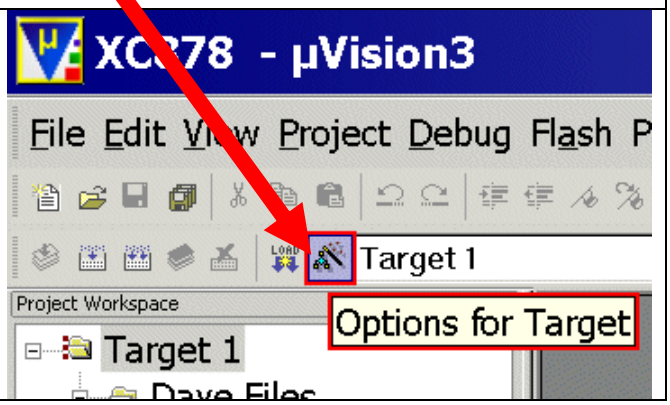


Note:
From now on just open your µVision project (not the DAvE project). µVision will automatically recognise if there has been a code regeneration done by DAvE!

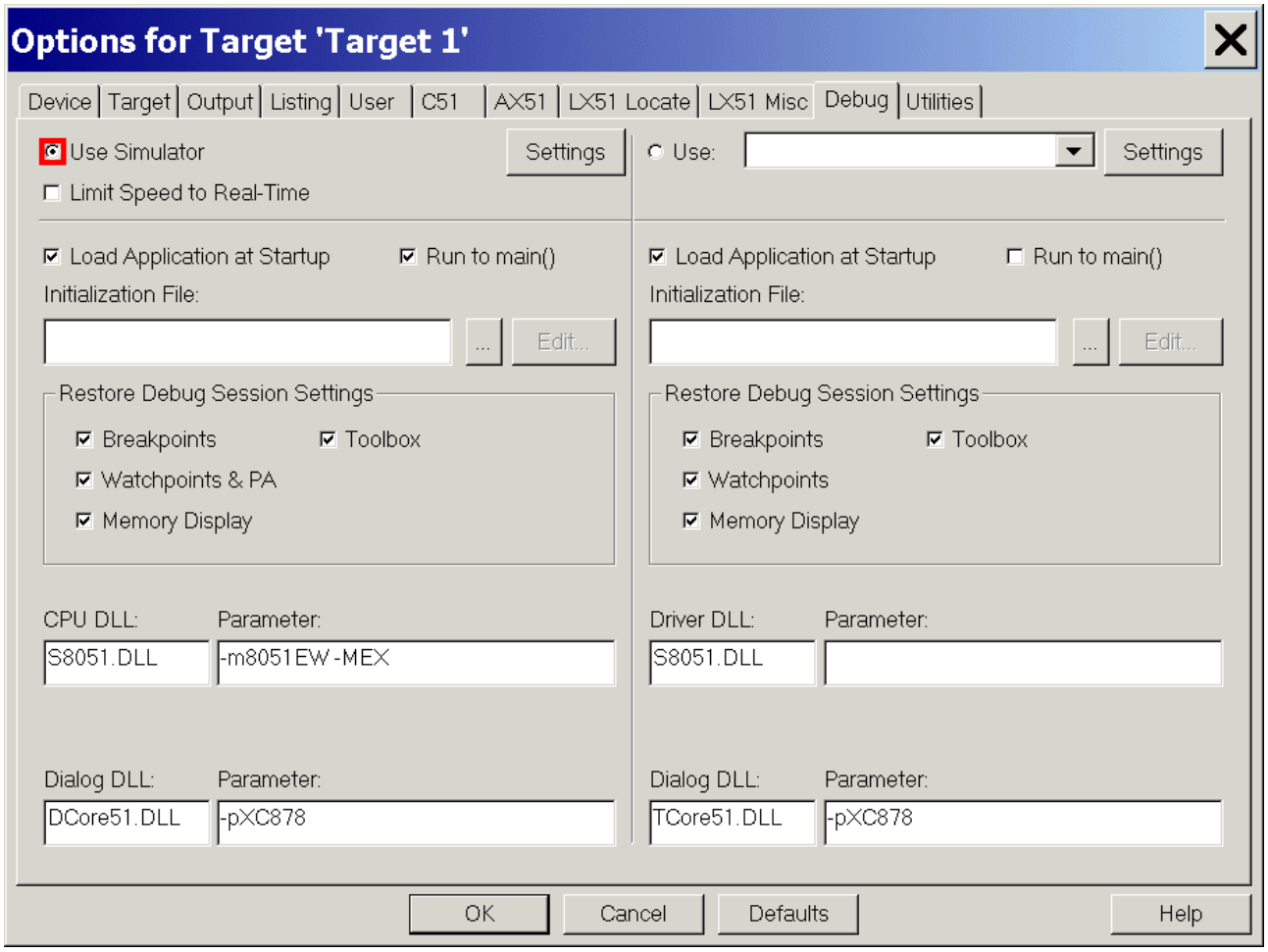
Check the configuration of the µVision simulator:

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



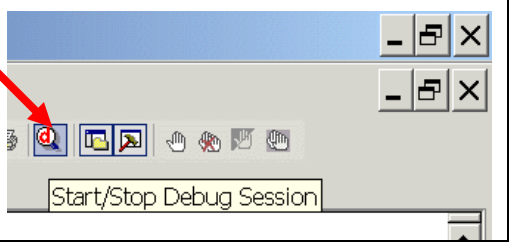


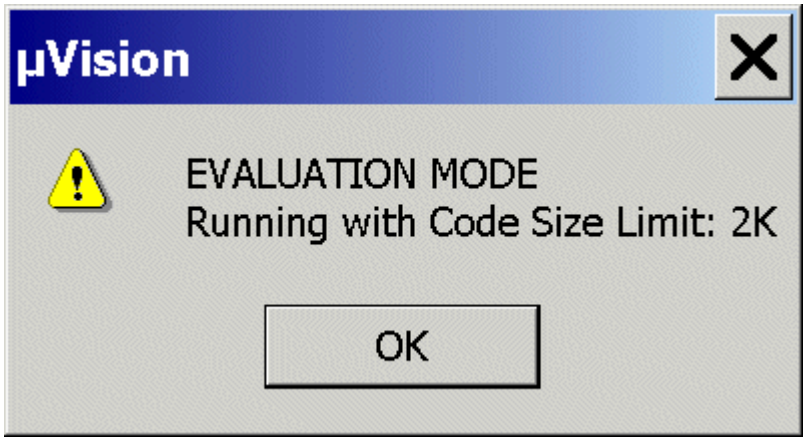
Options for Target ´Target1‘: Debug: check ⊙ Use Simulator



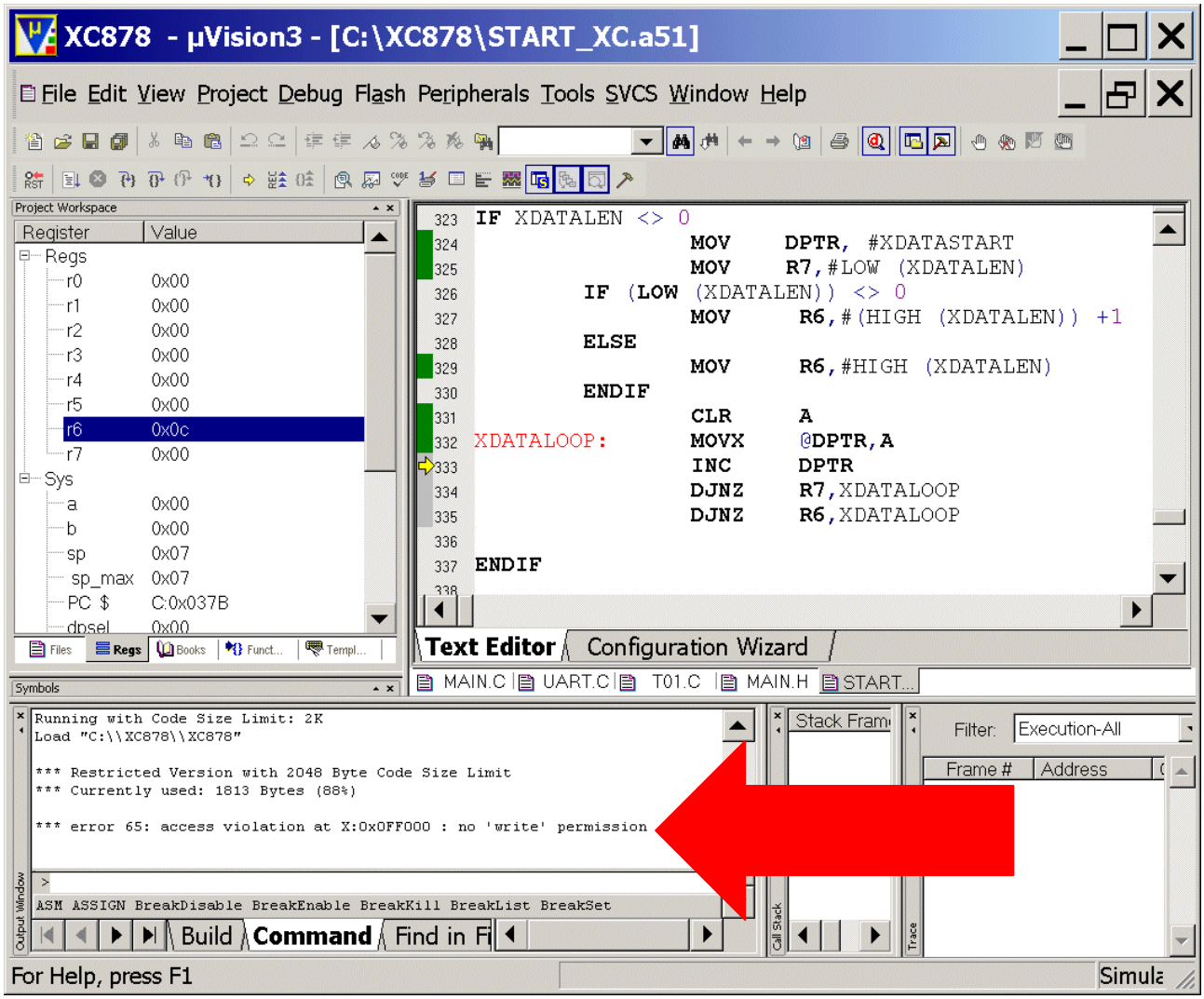
OK

Start the µVision Simulator:

Debug - Start/Stop Debug Session	or	click 

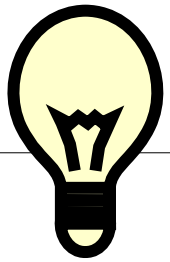


OK



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 µVision3 access and use memory. µVision3 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 µVision3.

When you run your target program, µVision3 checks each memory access to determine if it is outside the memory map. If an invalid access is made, µVision3 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.

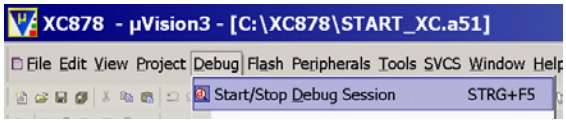
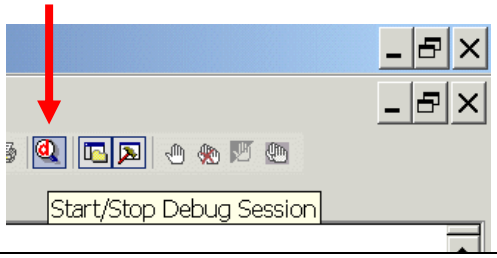
The VNM option identifies the specified memory range as von Neumann memory. When VMN is specified with an address range, µVision3 overlaps external data memory (XDATA) and code memory. Write accesses to external data memory also change code memory. Memory ranges specified with VNM may not be a range 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.

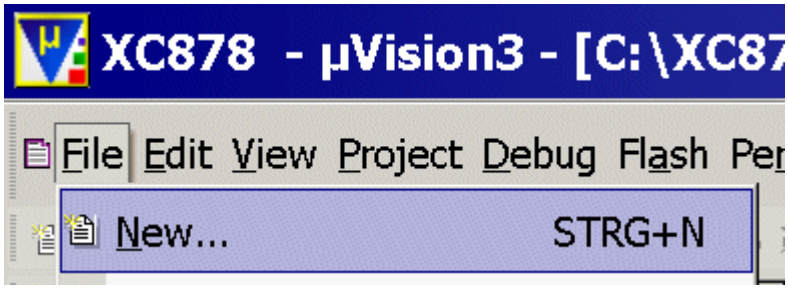
When µVision3 loads, the following memory maps are defined.
CPU Address Range Access
8051 Family 0x000000-0x00FFFF (DATA) READ WRITE
0x010000-0x01FFFF (XDATA) READ WRITE
0xFF0000-0xFFFFFF (CODE) EXEC READ

µVision3 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 µVision 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 */

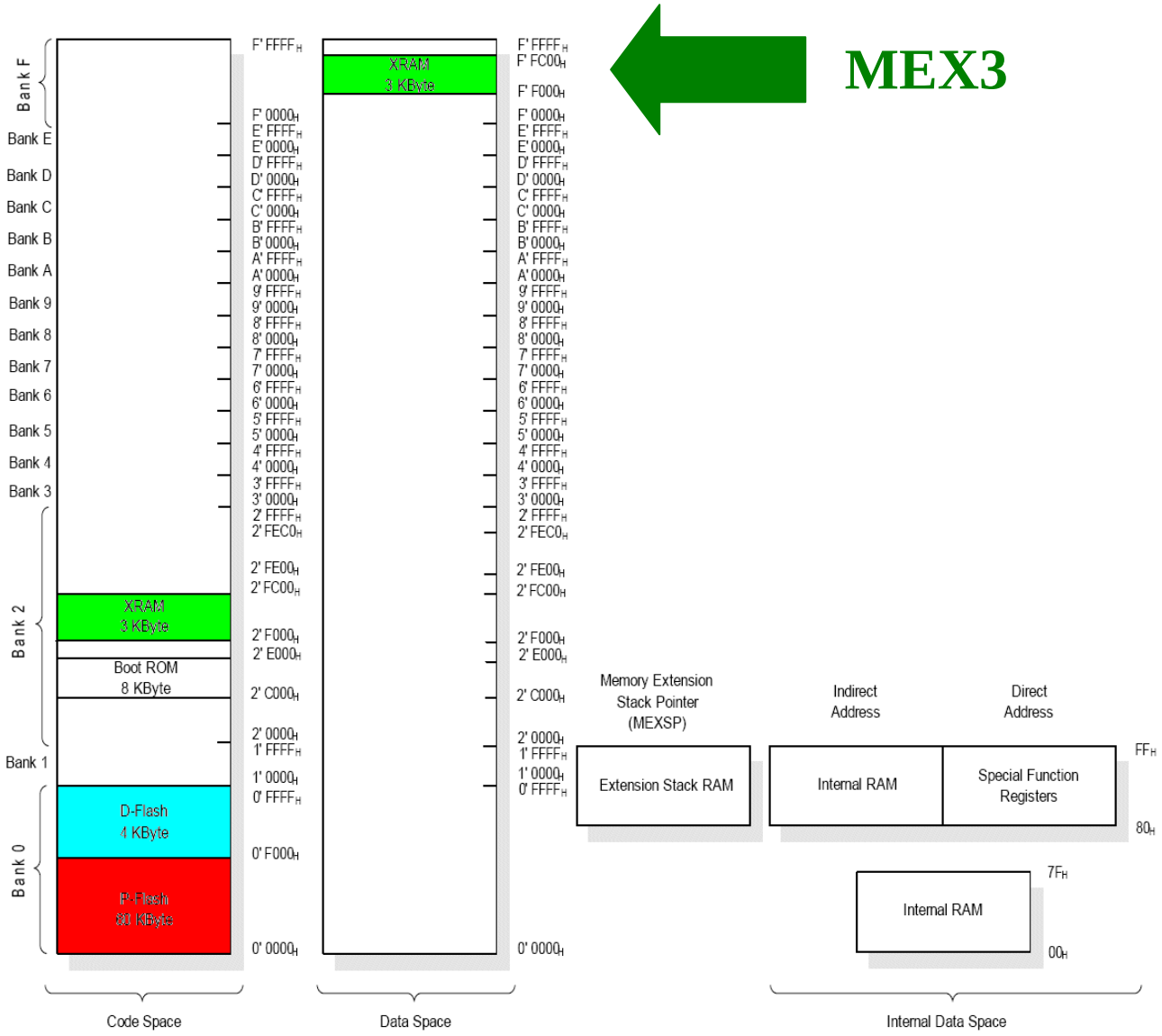
Debug - Start/Stop Debug Session 	or	click 

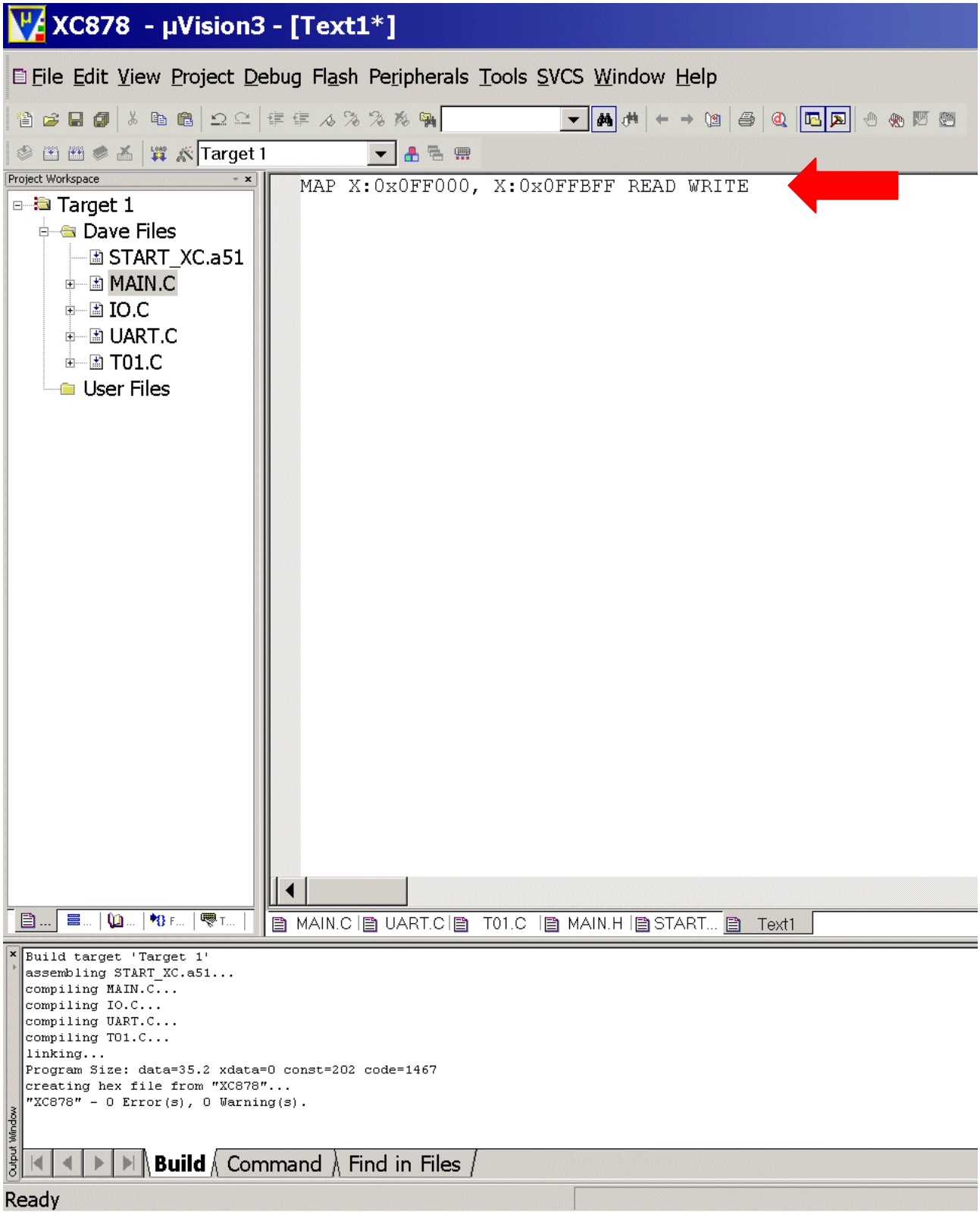
File – New…



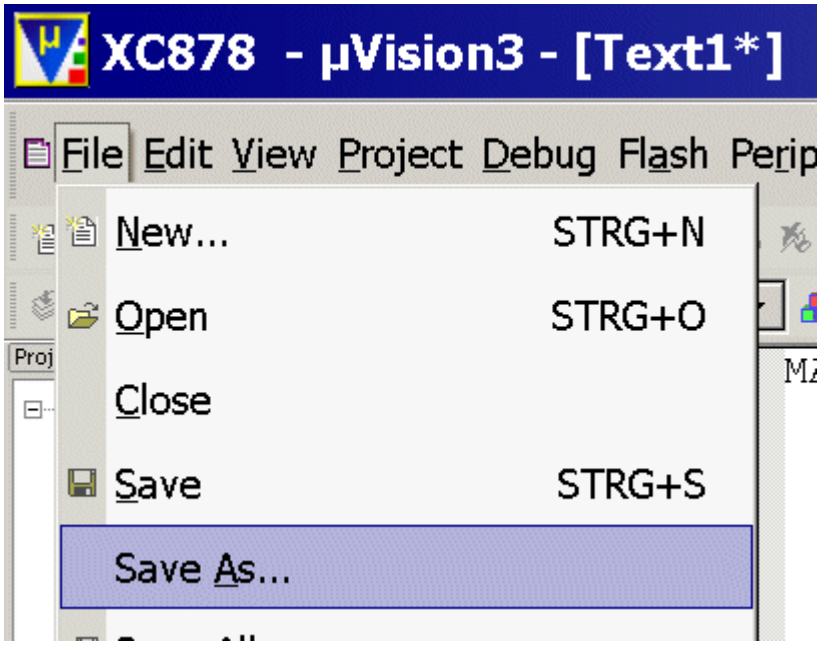
Insert:

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

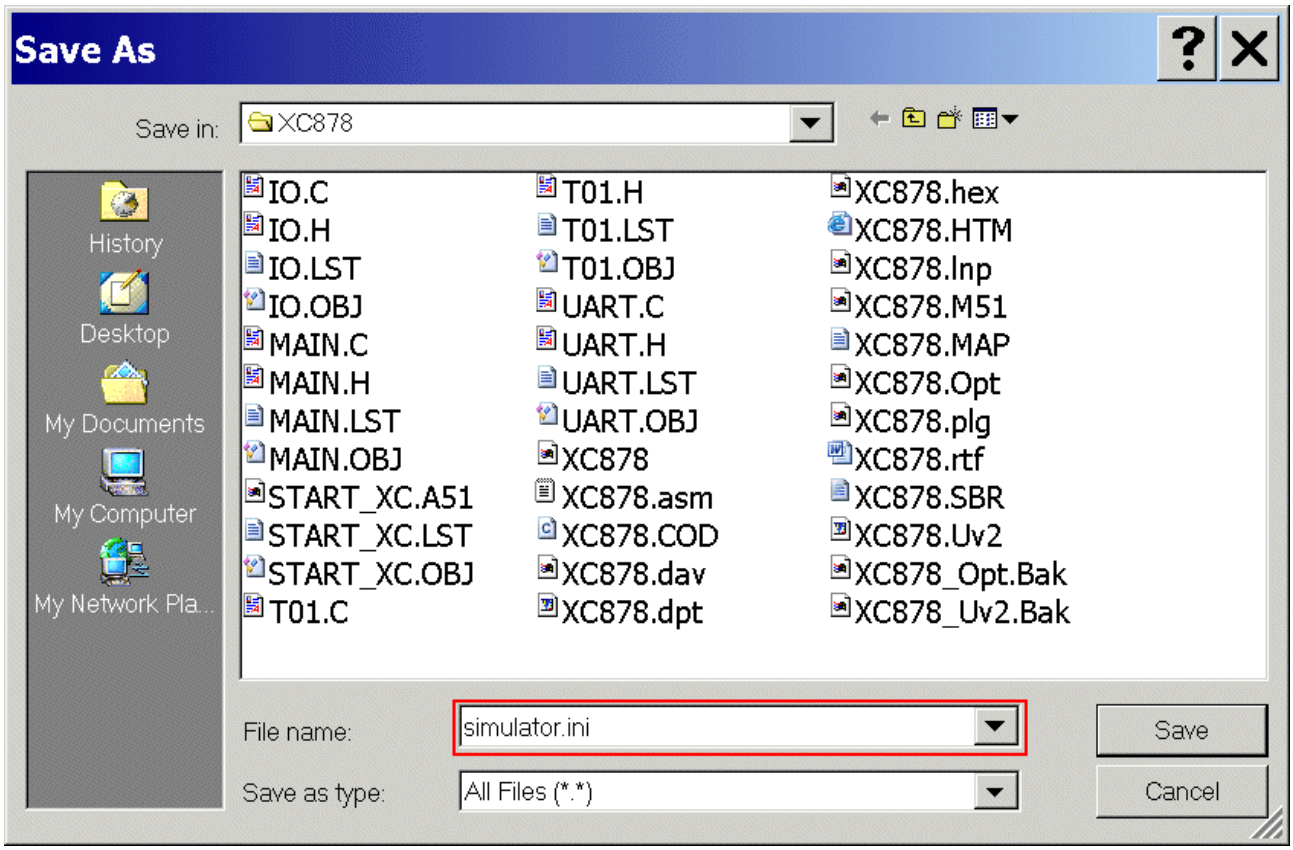



XC878 - µVision3 - [Text1*]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Dave Files
START_XC.a51
MAIN.C
IO.C
UART.C
T01.C
User Files
MAP X:0x0FF000, X:0x0FFBFF READ WRITE
MAIN.C
UART.C
T01.C
MAIN.H
START...
Text1
Build target 'Target 1'
assembling 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).
Output Window
Build
Command
Find in Files
Ready

File – Save As…

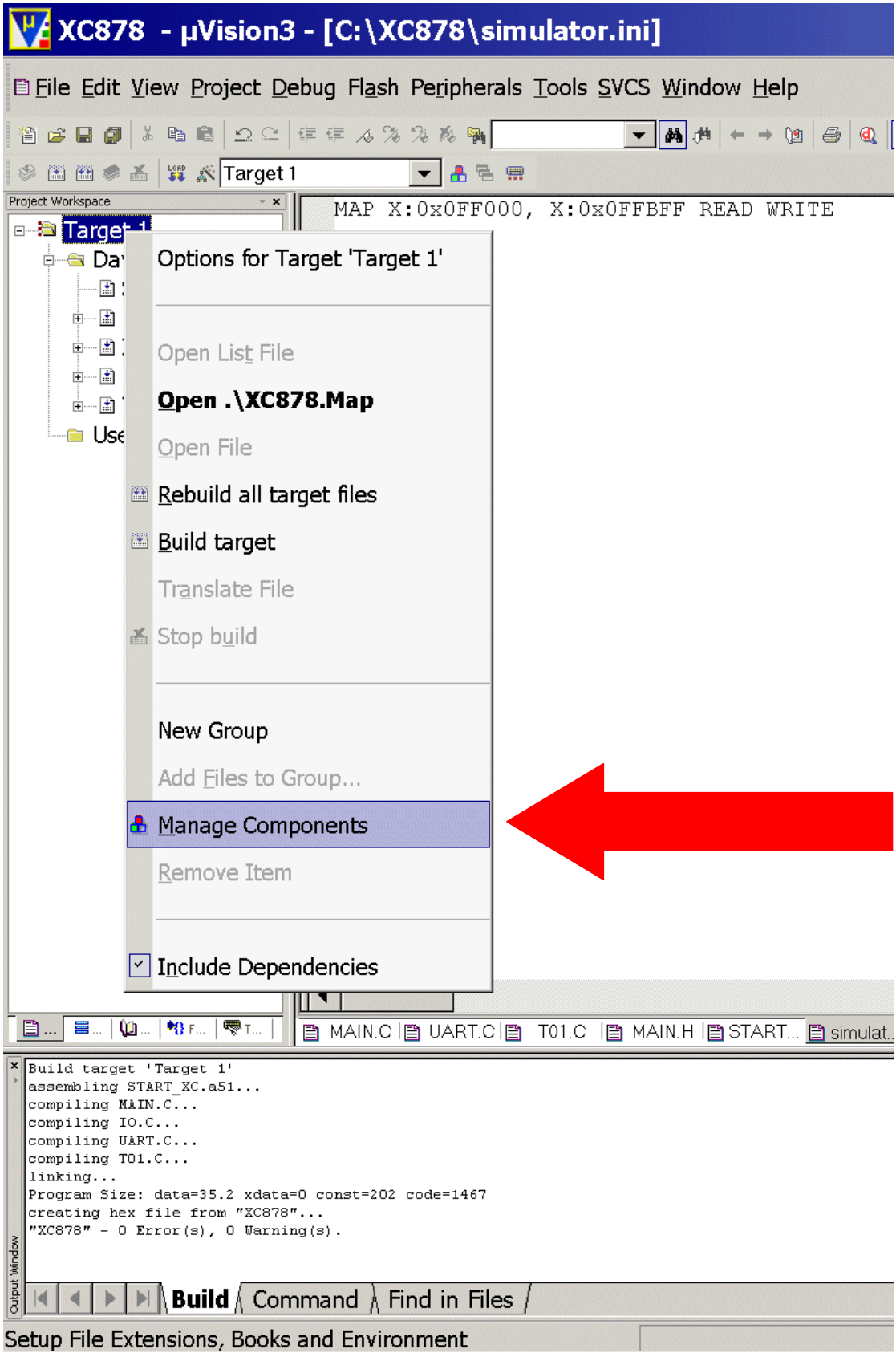


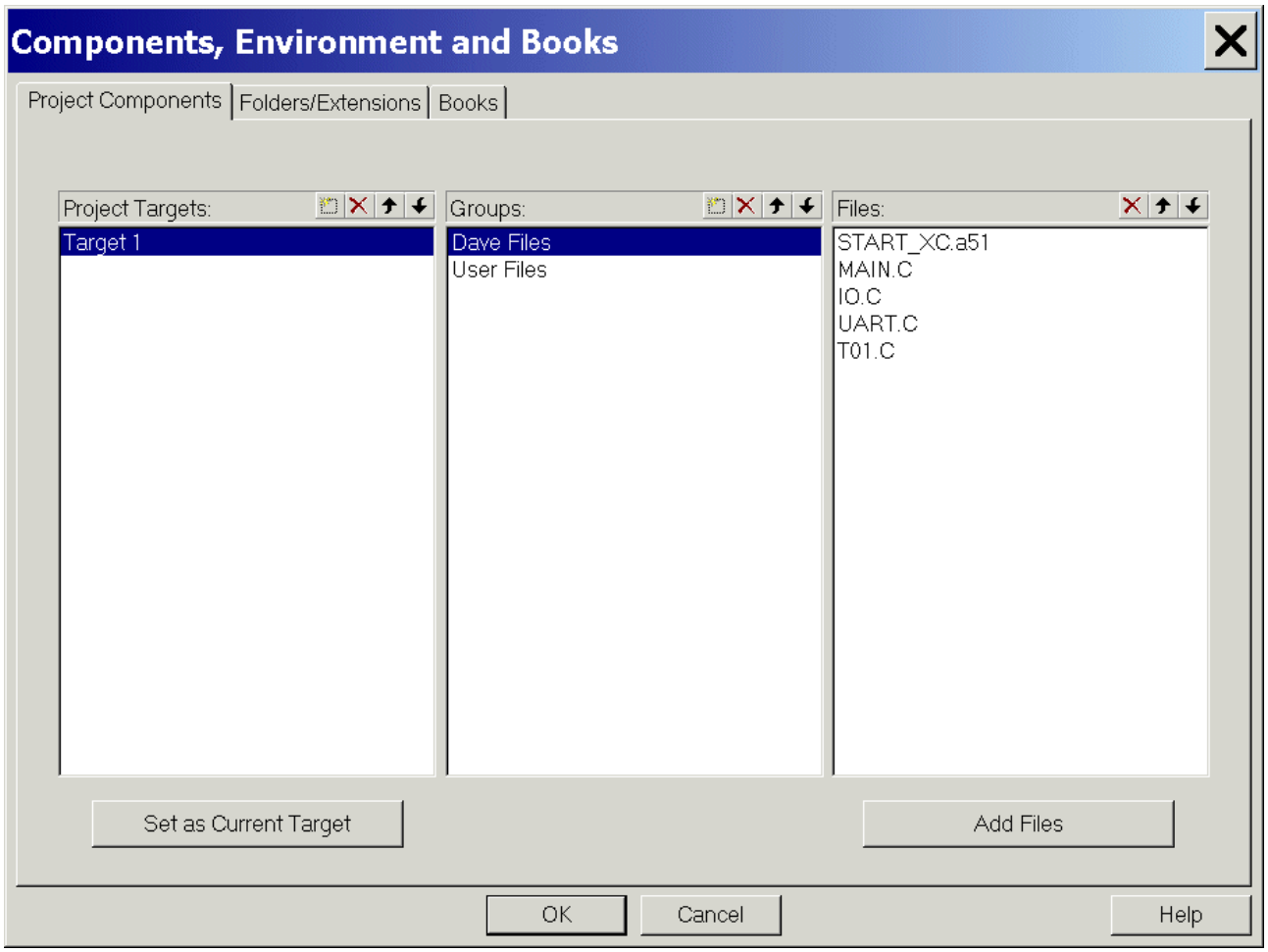
Save As: File name: insert: simulator.ini



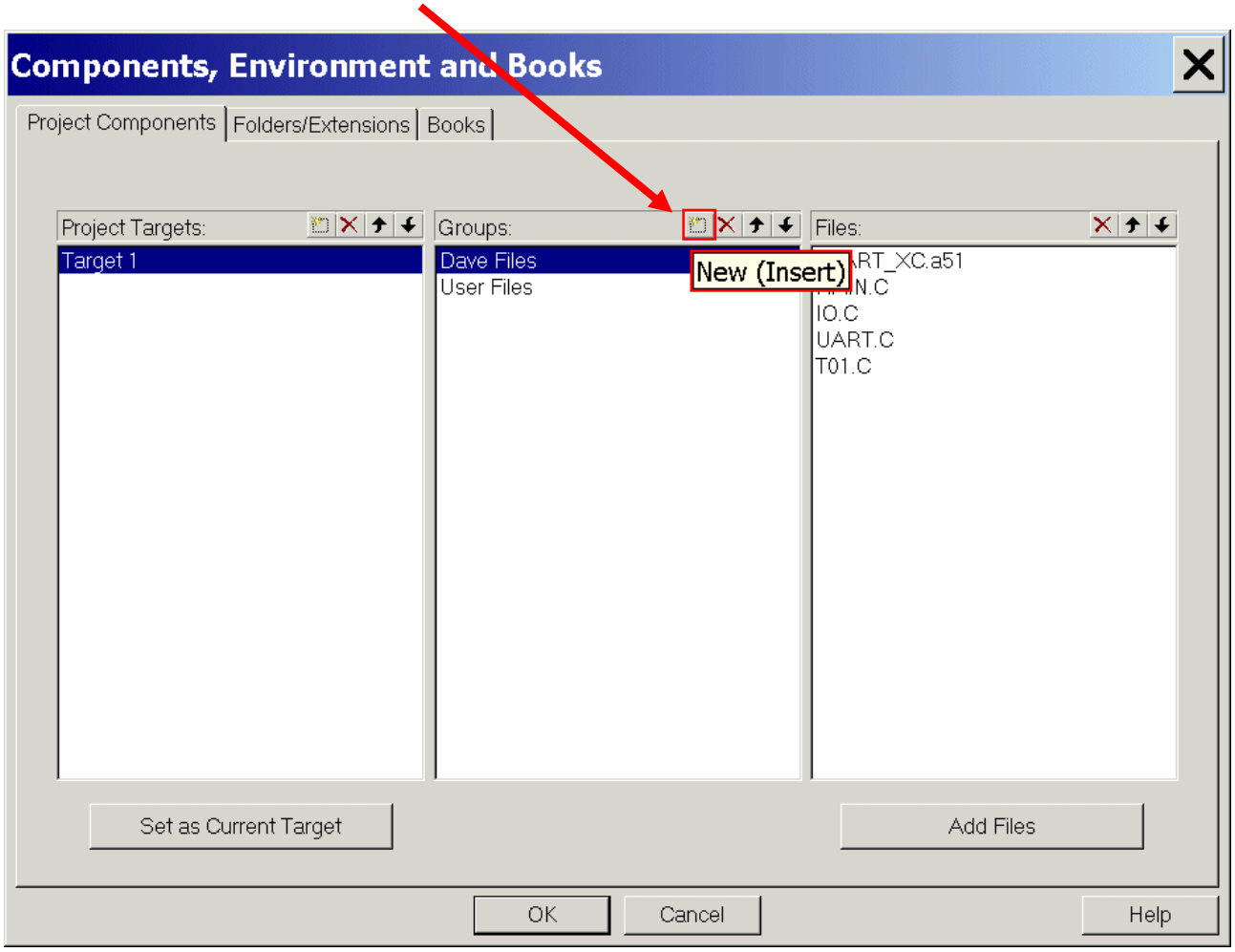
Click Save

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

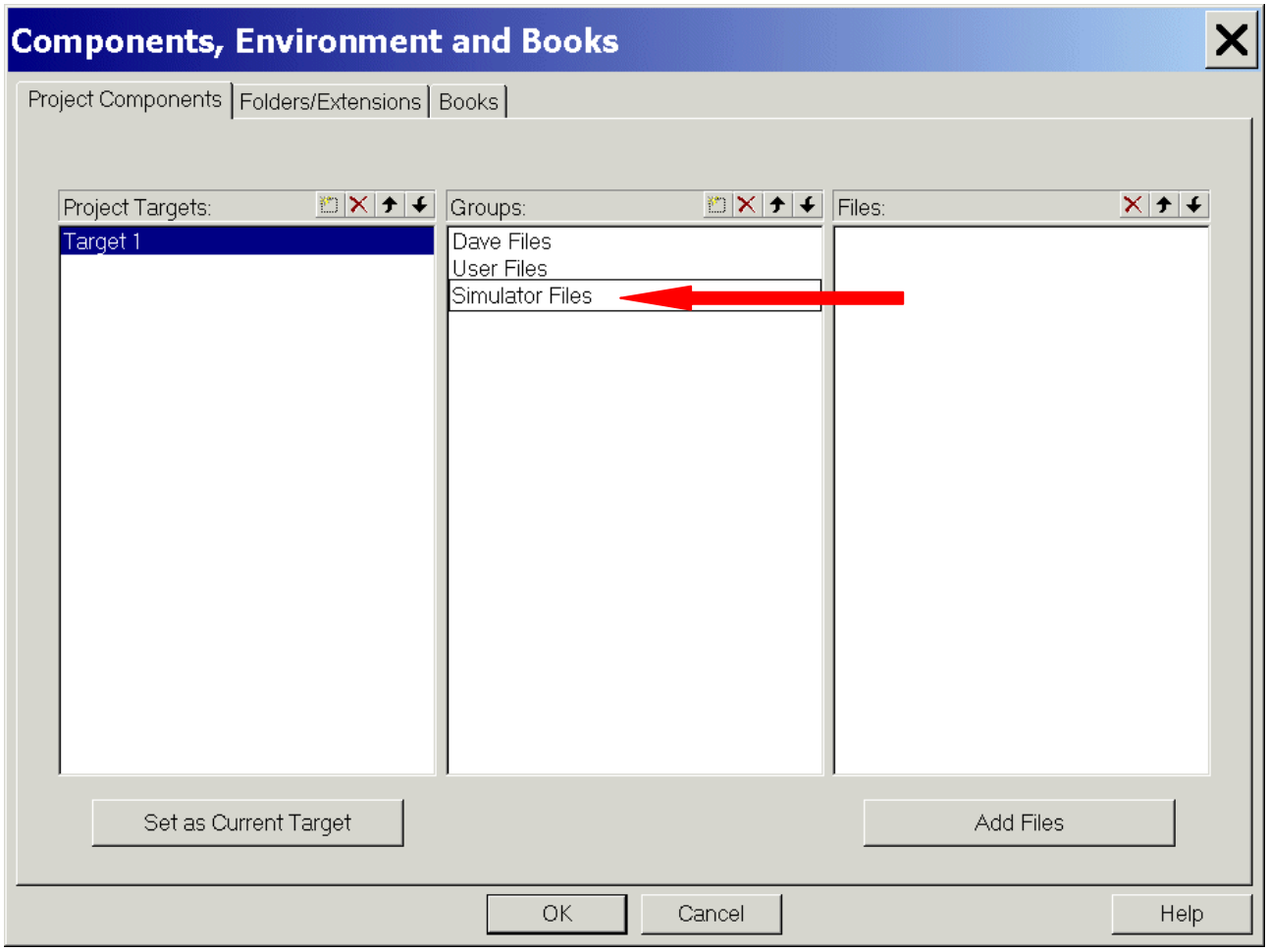



Components, Environment and Books
Project Components
Folders/Extensions
Books
Project Targets:
Target 1
Groups:
Dave Files
User Files
Files:
START_XC.a51
MAIN.C
IO.C
UART.C
T01.C
Set as Current Target
Add Files
OK
Cancel
Help

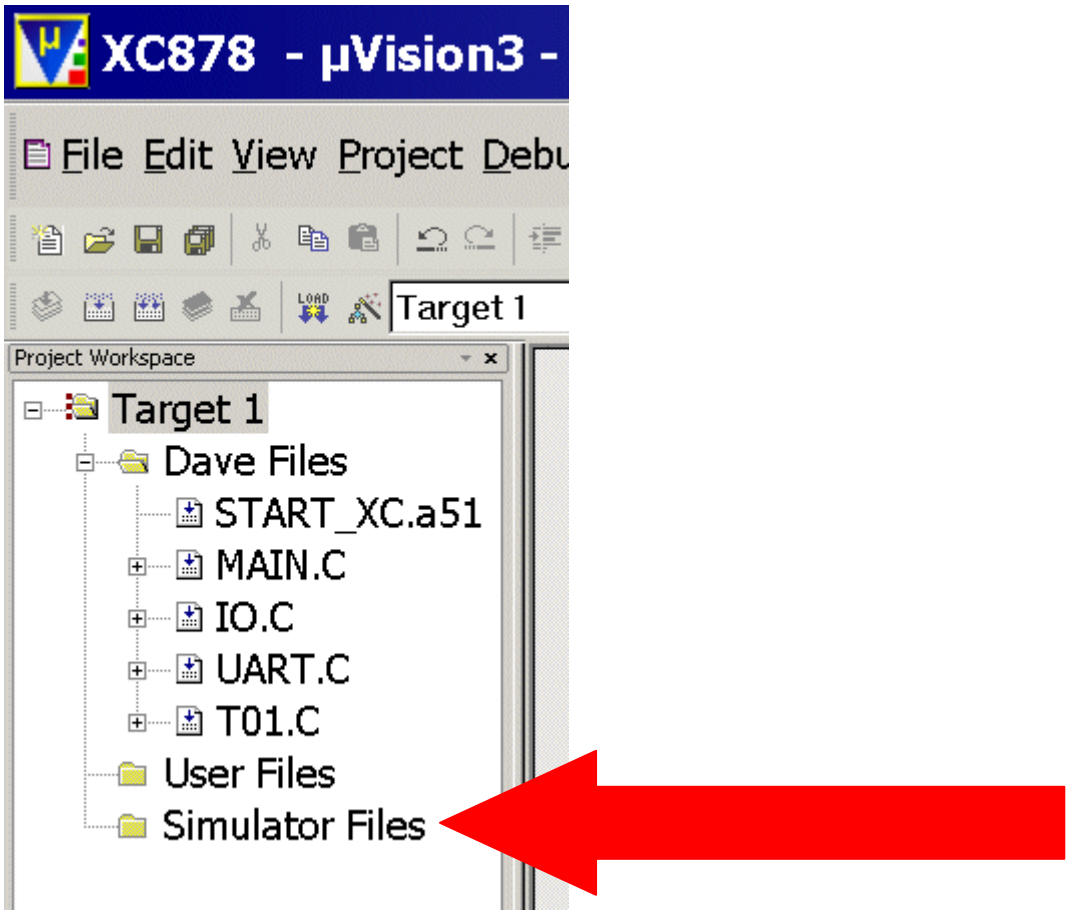
Project Components: Groups: click New (Insert)



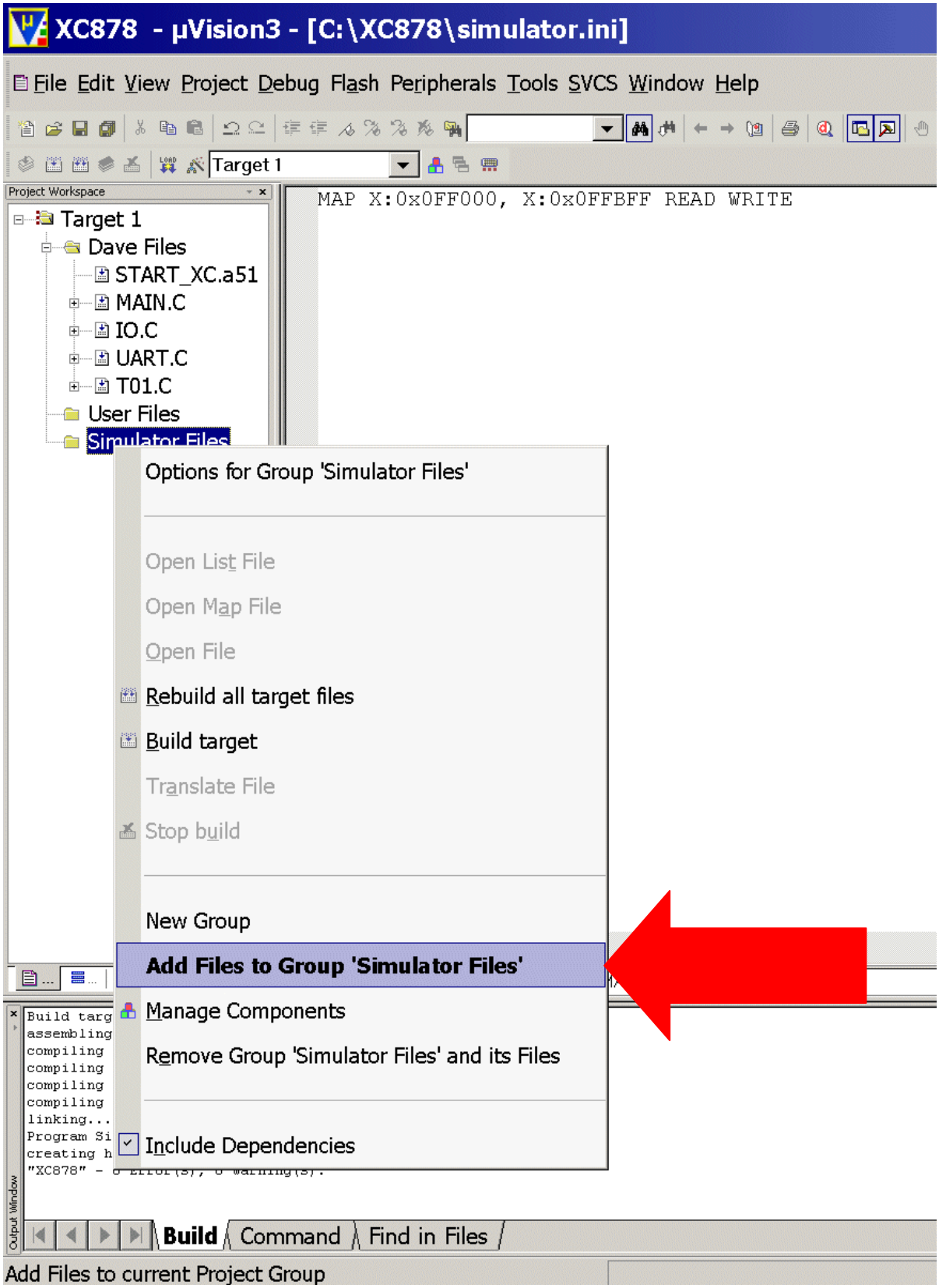
Insert Simulator Files



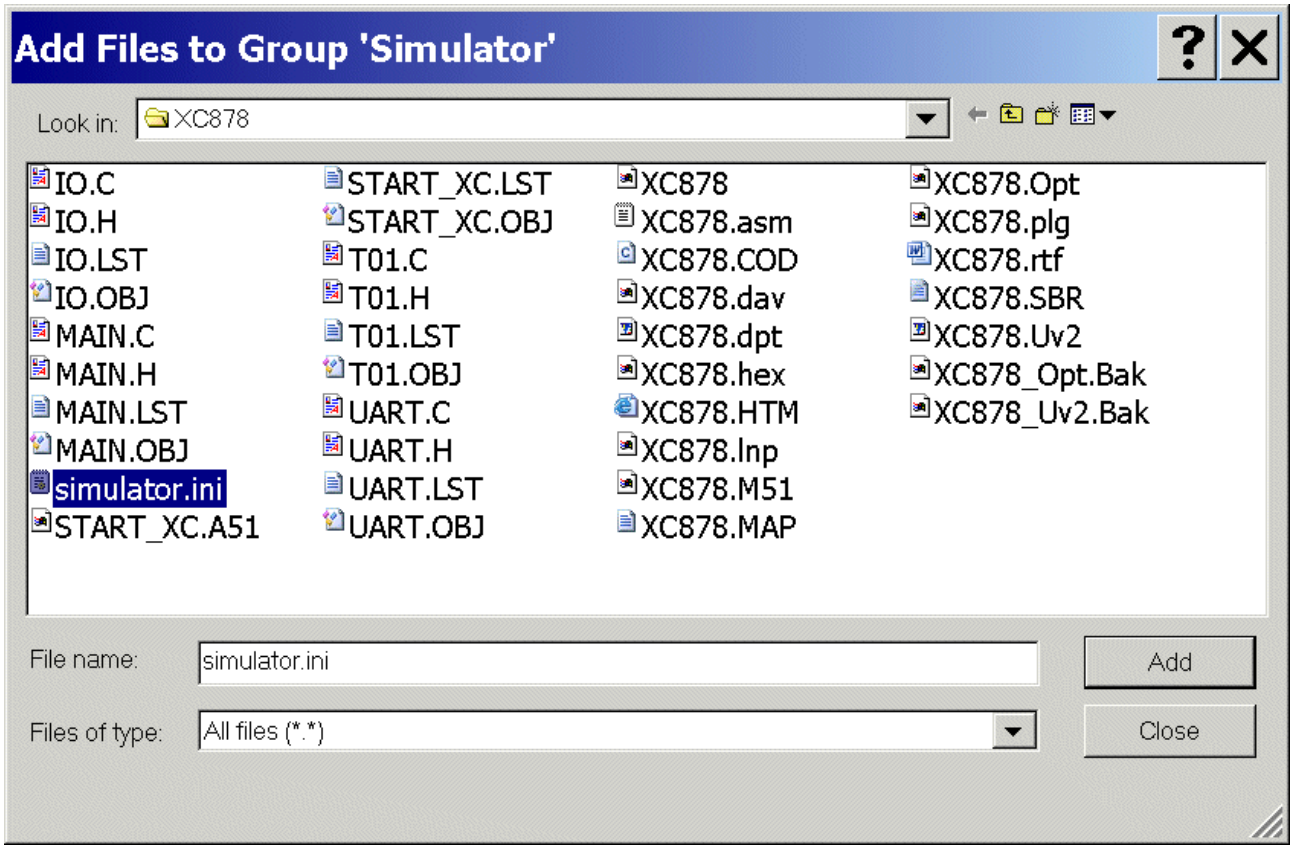
Click OK


XC878 - µVision3 -
File Edit View Project Debu
Target 1
Project Workspace
Target 1
Dave Files
START_XC.a51
MAIN.C
IO.C
UART.C
T01.C
User Files
Simulator Files

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

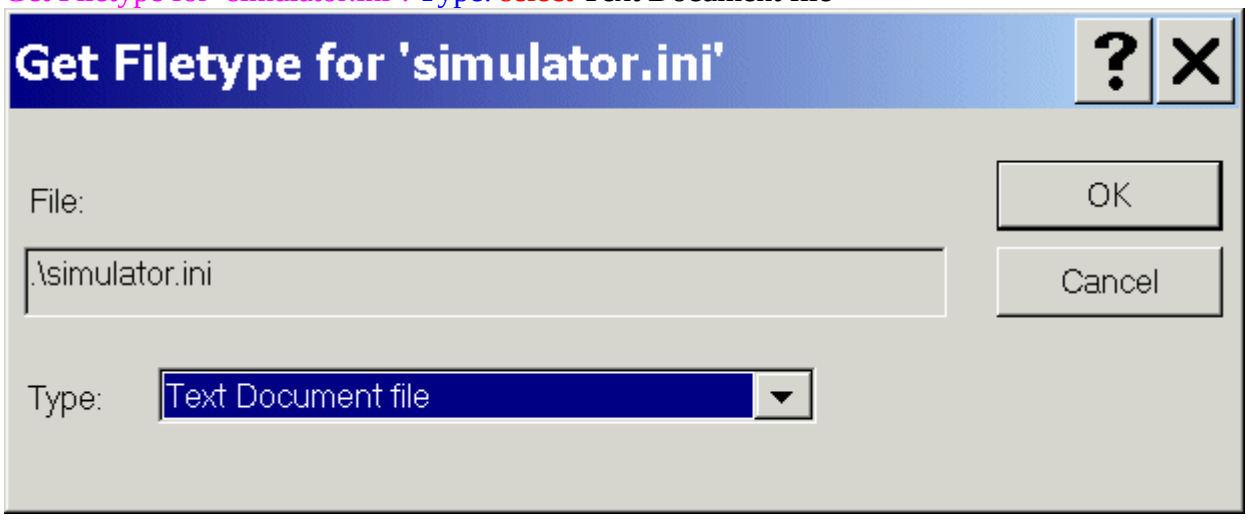


Files of type: select All files (*.*)
Click simulator.ini

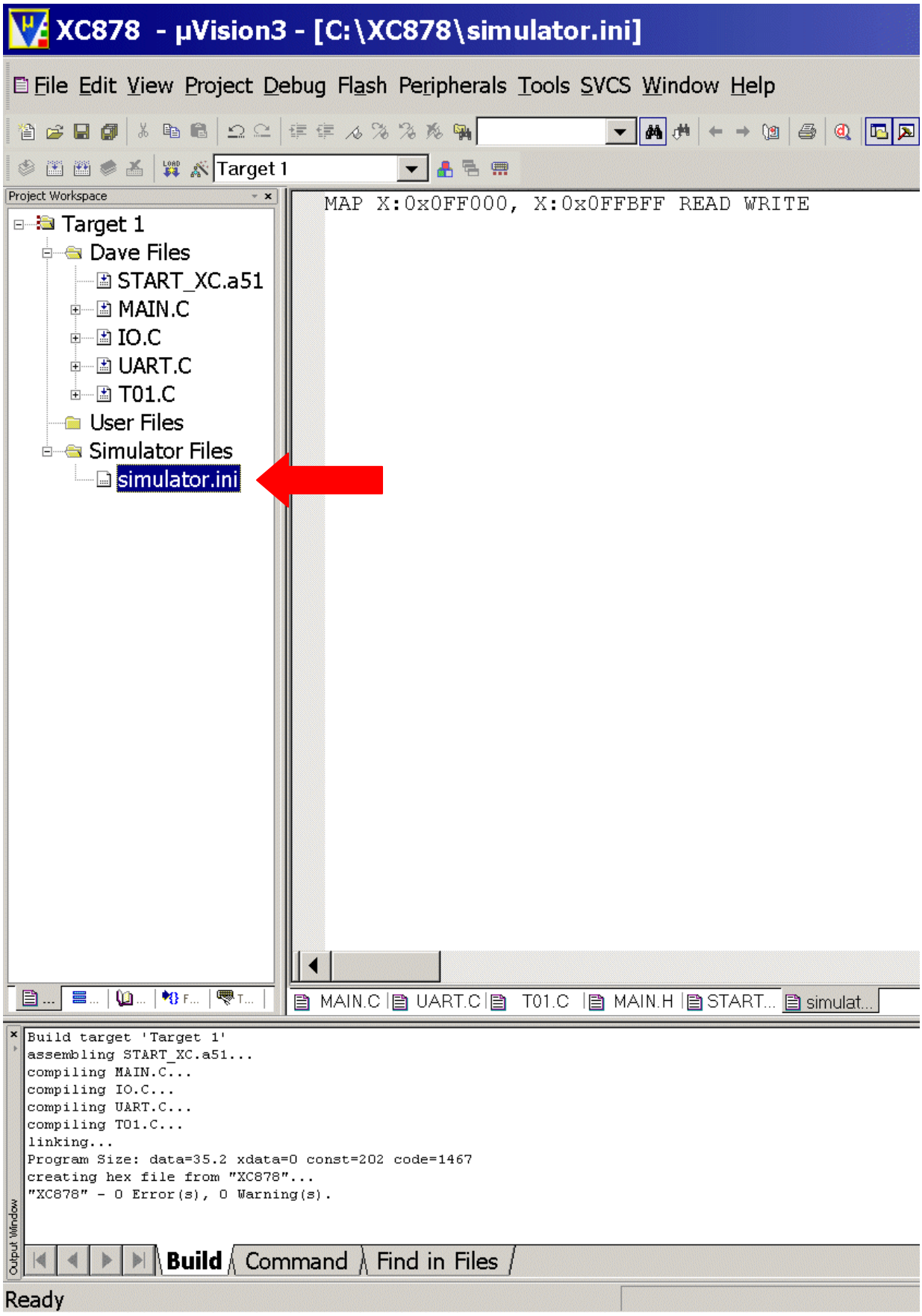


Click Add

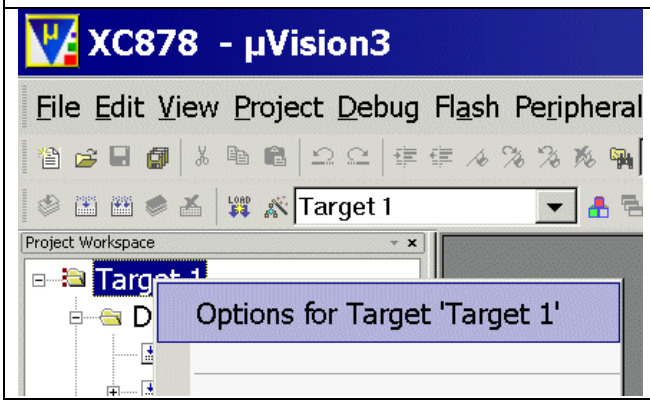
Get Filetype for 'simulator.ini': Type: select Text Document file

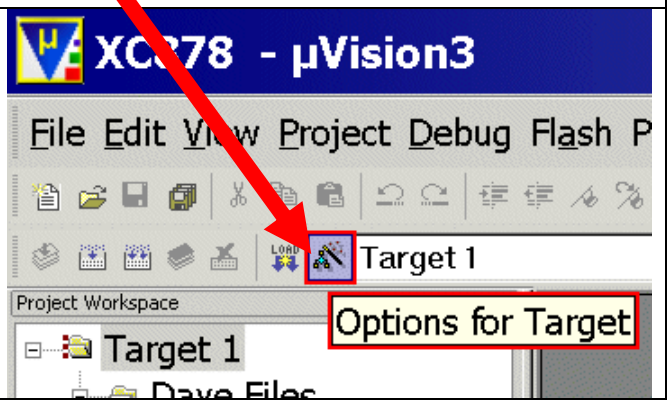


Click OK
Click Close

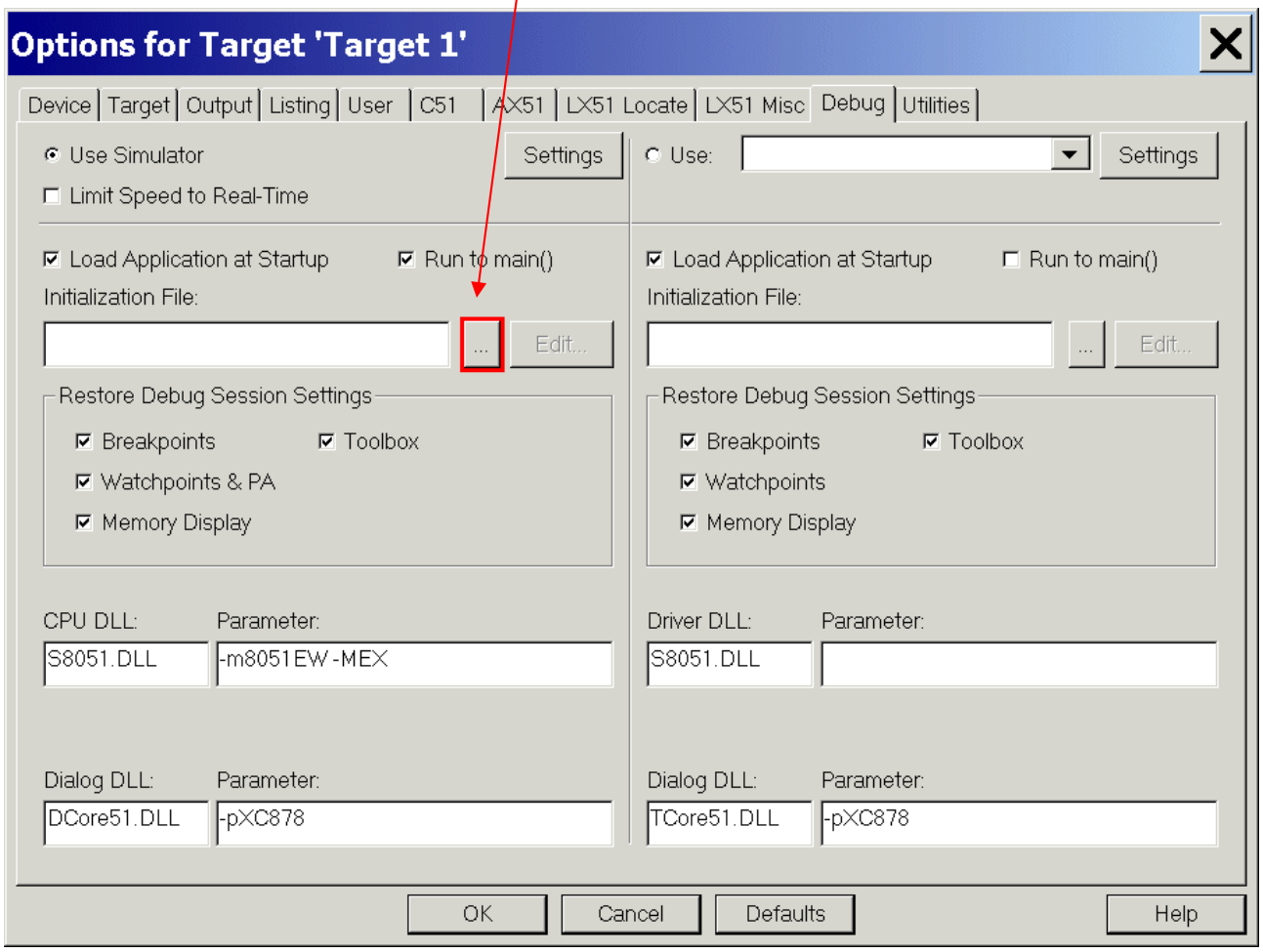

XC878 - µVision3 - [C:\XC878\simulator.ini]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Dave Files
START_XC.a51
MAIN.C
IO.C
UART.C
T01.C
User Files
Simulator Files
simulator.ini
MAP X:0x0FF000, X:0x0FFBFF READ WRITE
MAIN.C
UART.C
T01.C
MAIN.H
START...
simulat...
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).
Output Window
Build
Command
Find in Files
Ready

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

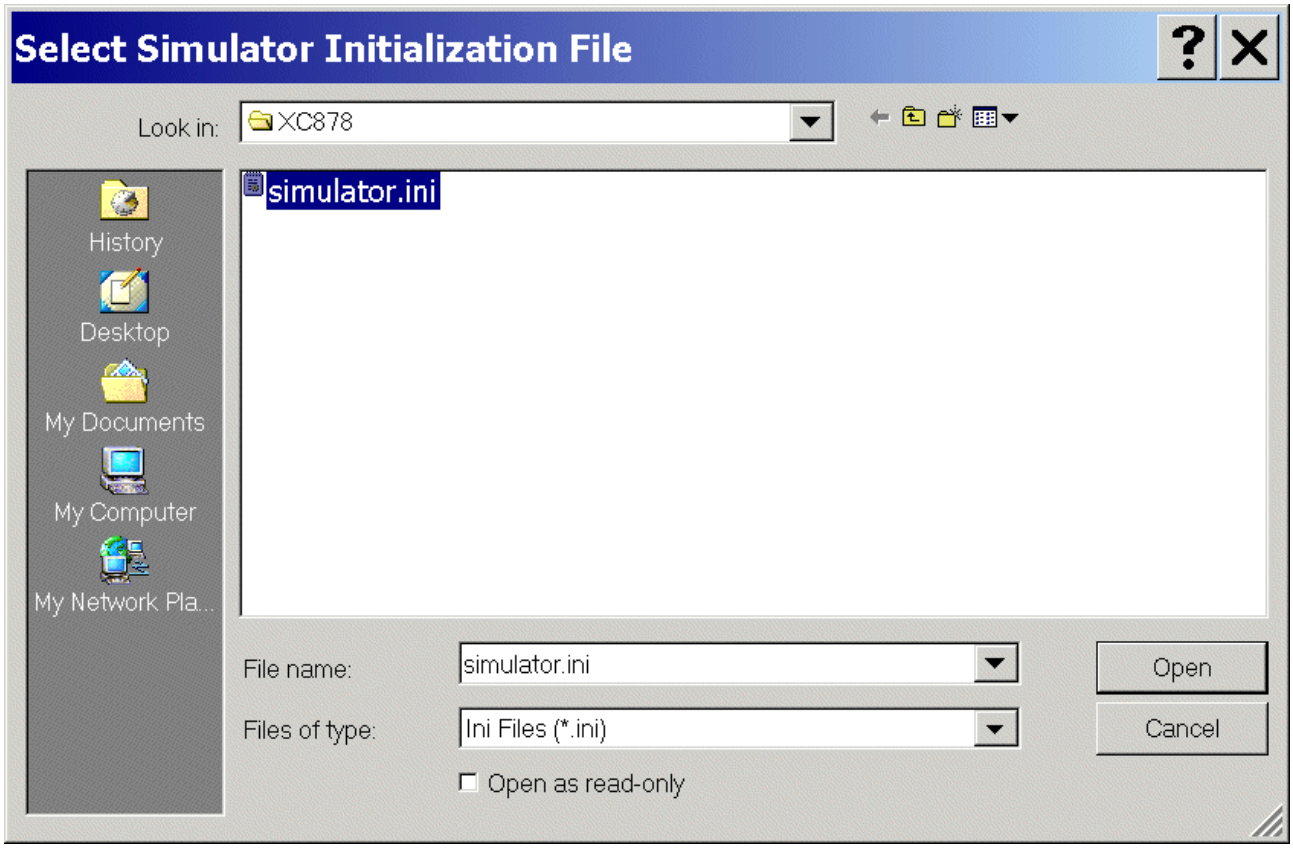




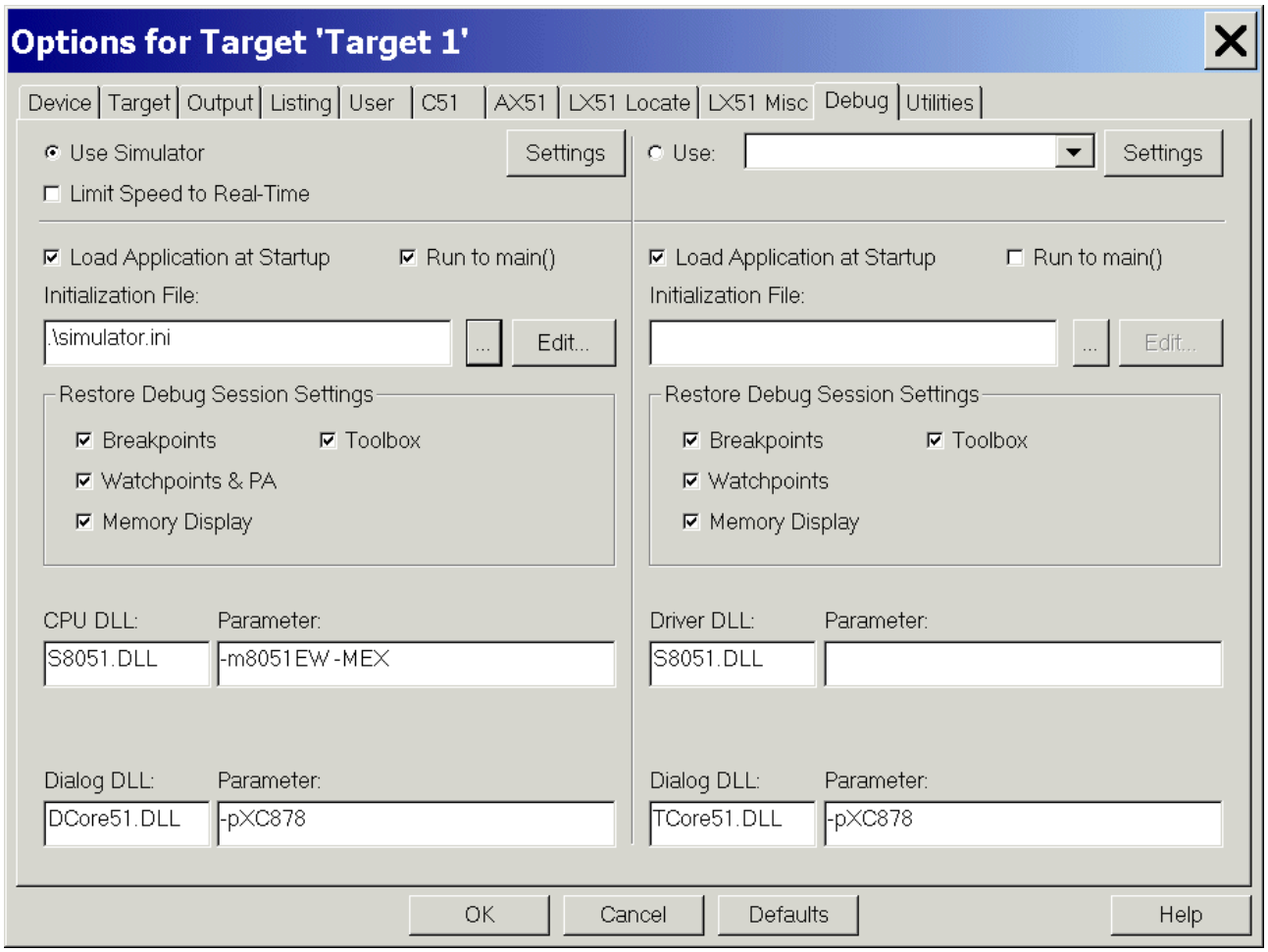
Options for Target ´Target1‘: Debug: click …



Click/select simulator.ini

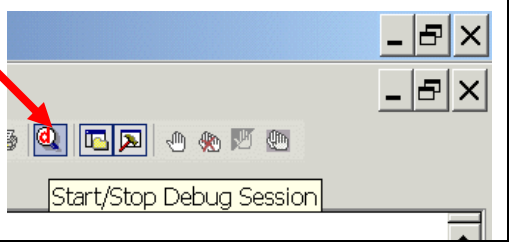


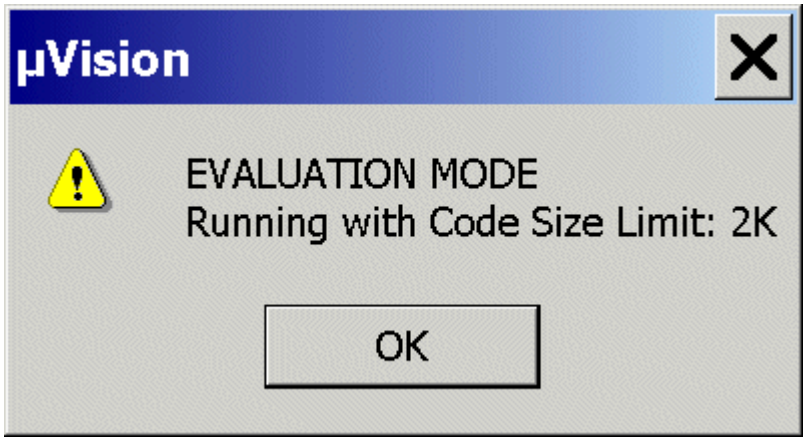
Click Open


Options for Target 'Target 1'
Device
Target
Output
Listing
User
C51
AX51
LX51 Locate
LX51 Misc
Debug
Utilities
Use Simulator
Settings
Limit Speed to Real-Time
Use:
Settings
Load Application at Startup
Run to main()
Initialization File:
.\simulator.ini
Edit...
Restore Debug Session Settings
Breakpoints
Toolbox
Watchpoints & PA
Memory Display
Load Application at Startup
Run to main()
Initialization File:
Edit...
Restore Debug Session Settings
Breakpoints
Toolbox
Watchpoints
Memory Display
CPU DLL:
Parameter:
S8051.DLL
-m8051EW -MEX
Driver DLL:
Parameter:
S8051.DLL
Dialog DLL:
Parameter:
DCore51.DLL
-pXC878
Dialog DLL:
Parameter:
TCore51.DLL
-pXC878
OK
Cancel
Defaults
Help

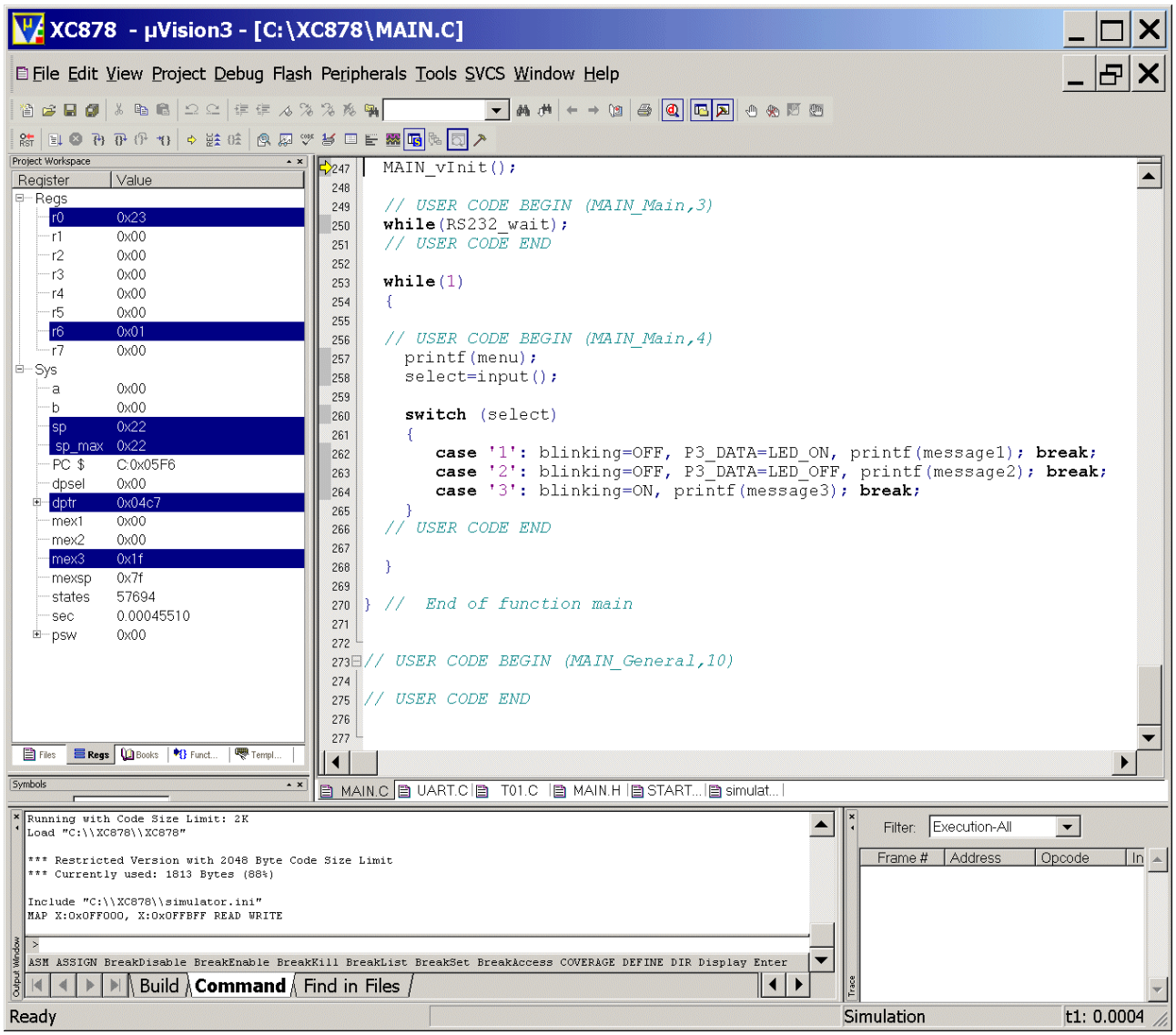
Click OK

Start the µVision Simulator again:

Debug - Start/Stop Debug Session	or	click 



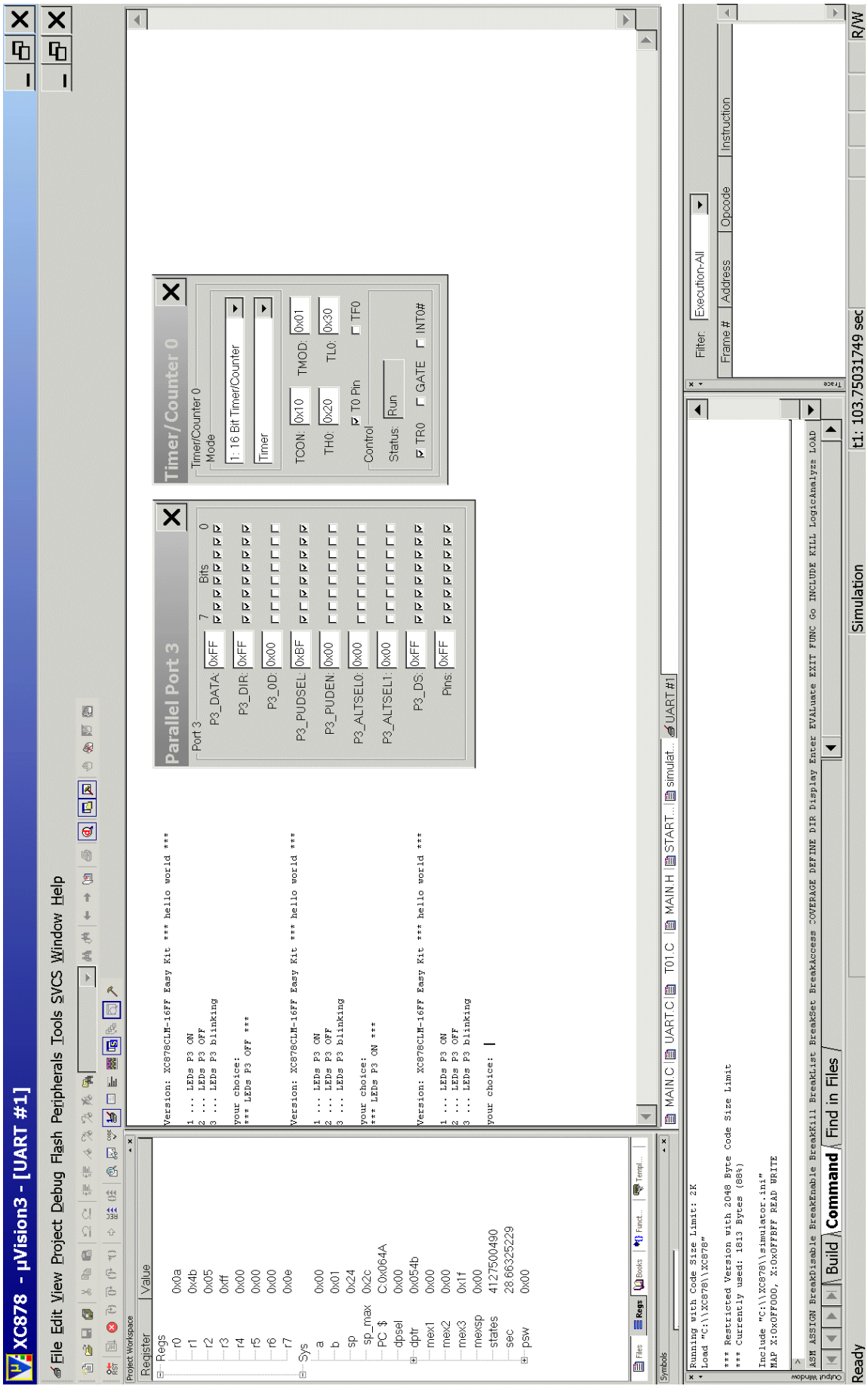
OK



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
Parallel Port 3
Timer/Counter 0
Version: XC878CLM-16FF Easy Kit *** hello world ***
1 ... LEDs P3 ON
2 ... LEDs P3 OFF
3 ... LEDs P3 blinking
your choice:
*** LEDs P3 OFF ***
your choice:
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:0x0FF000, X:0xDFFBFF READ WRITE
Ready
Simulation
t1: 103.75031749 sec

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 µVision 3:

Project - Close Project

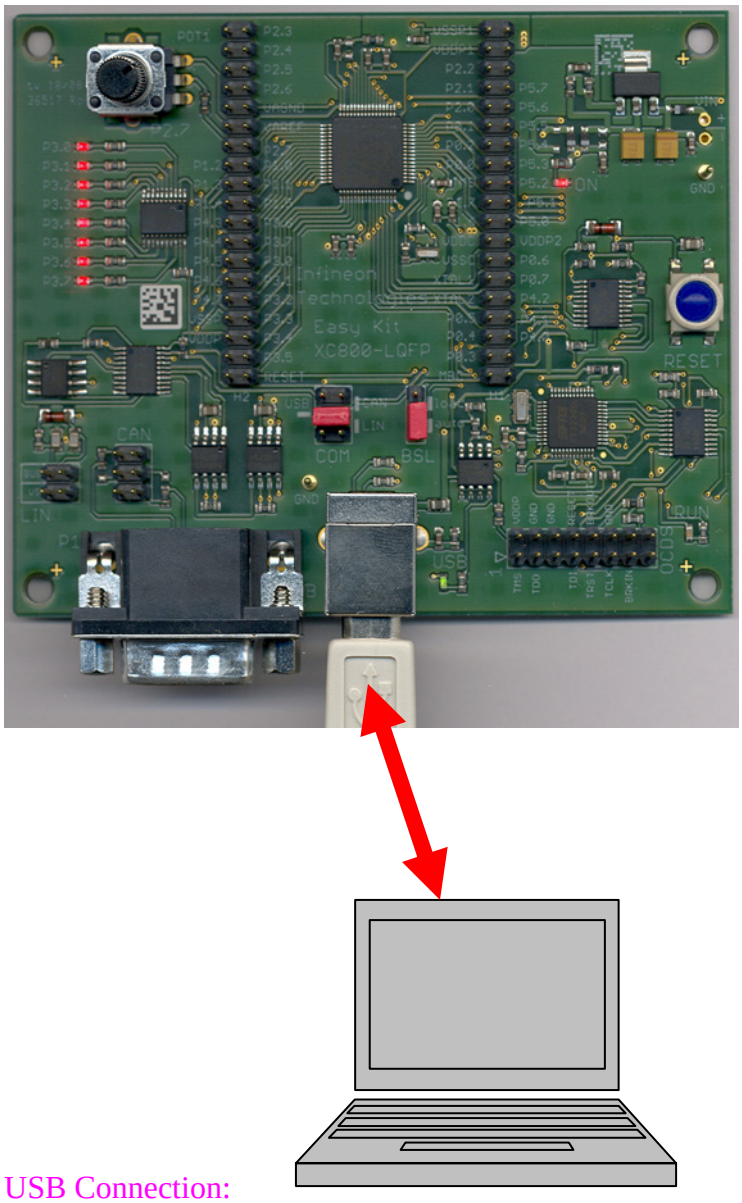
File – Exit

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 XC878 Easy Kit is still connected to the host computer:



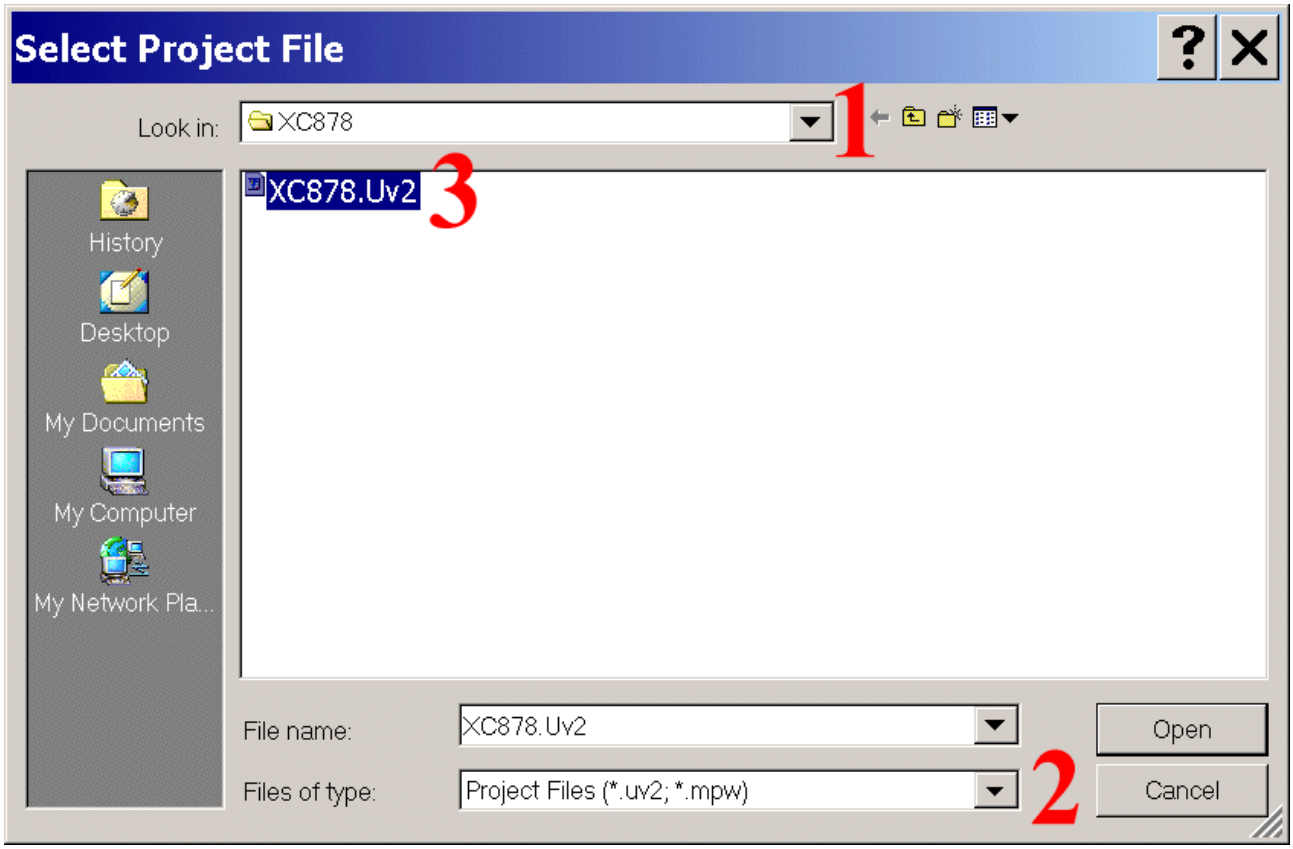
USB Connection:
.) used for: UART communication (the UART/RS232/serial interface is available via USB as a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).
.) used for: On-Chip-Flash-Programming and Debugging (first USB channel of the FTDI FT2232 Dual USB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

Start Keil µVision and open our Keil Project

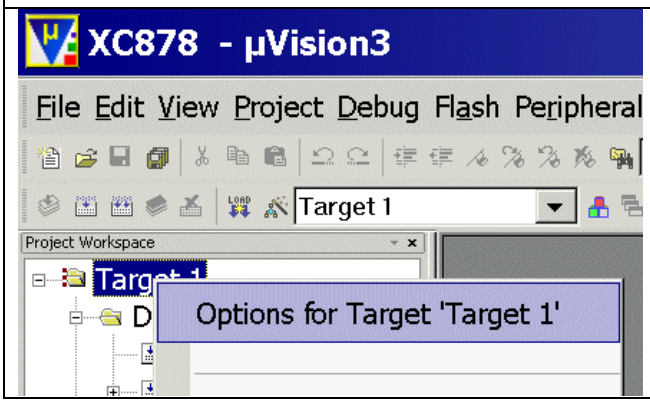
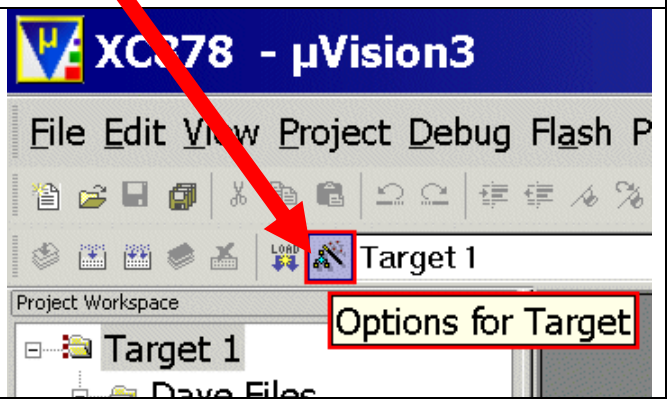
If you see an open project – close it: Project - Close Project

Project - Open Project
Select Project File: Look in: choose C:\XC878 (1)
Select Project File: Files of type: select Project Files (*.uv2) (2)
Click XC878.Uv2 (3)

Click Open

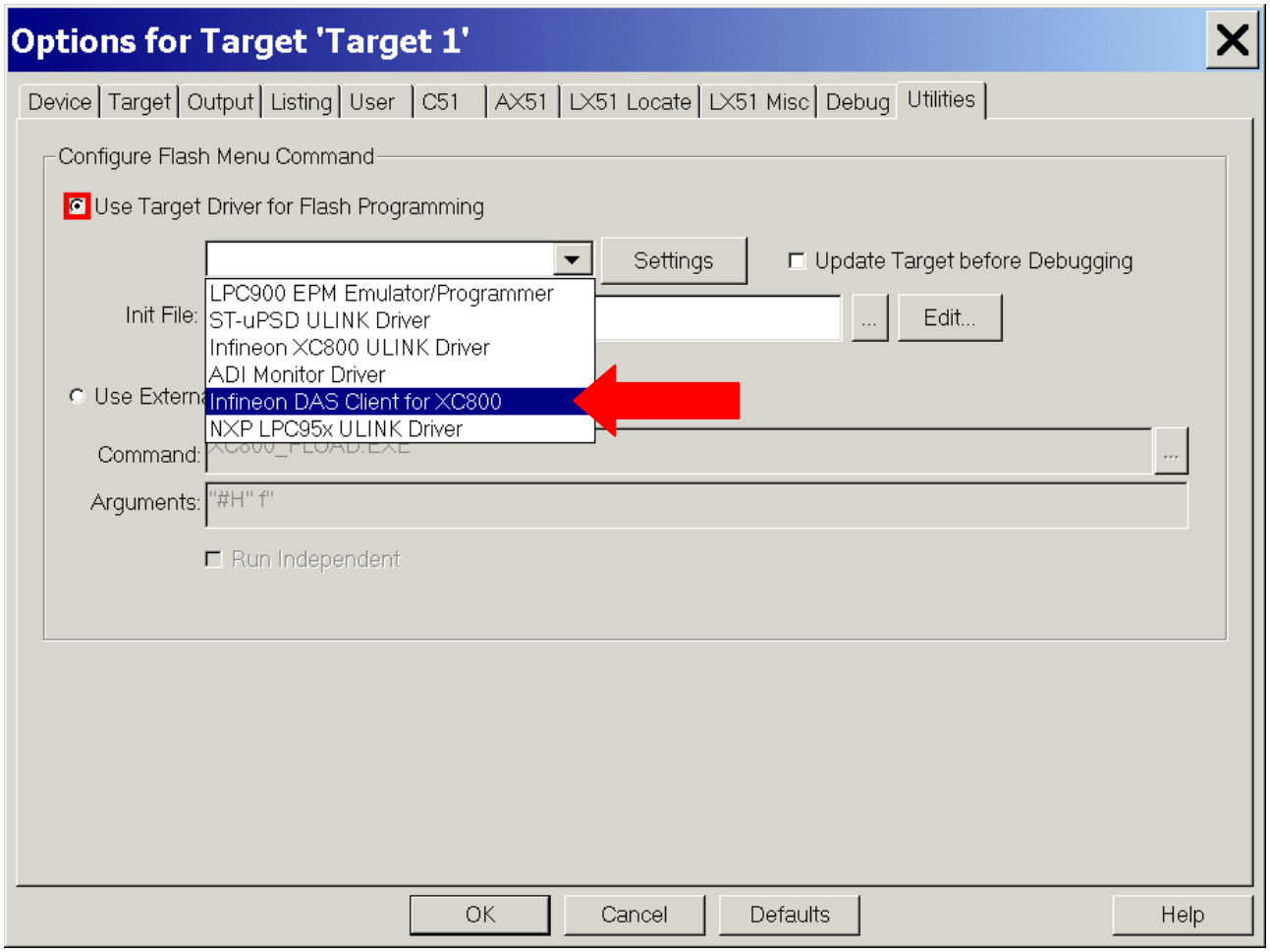


Check/configure the configuration of the Flash Programming Utility :

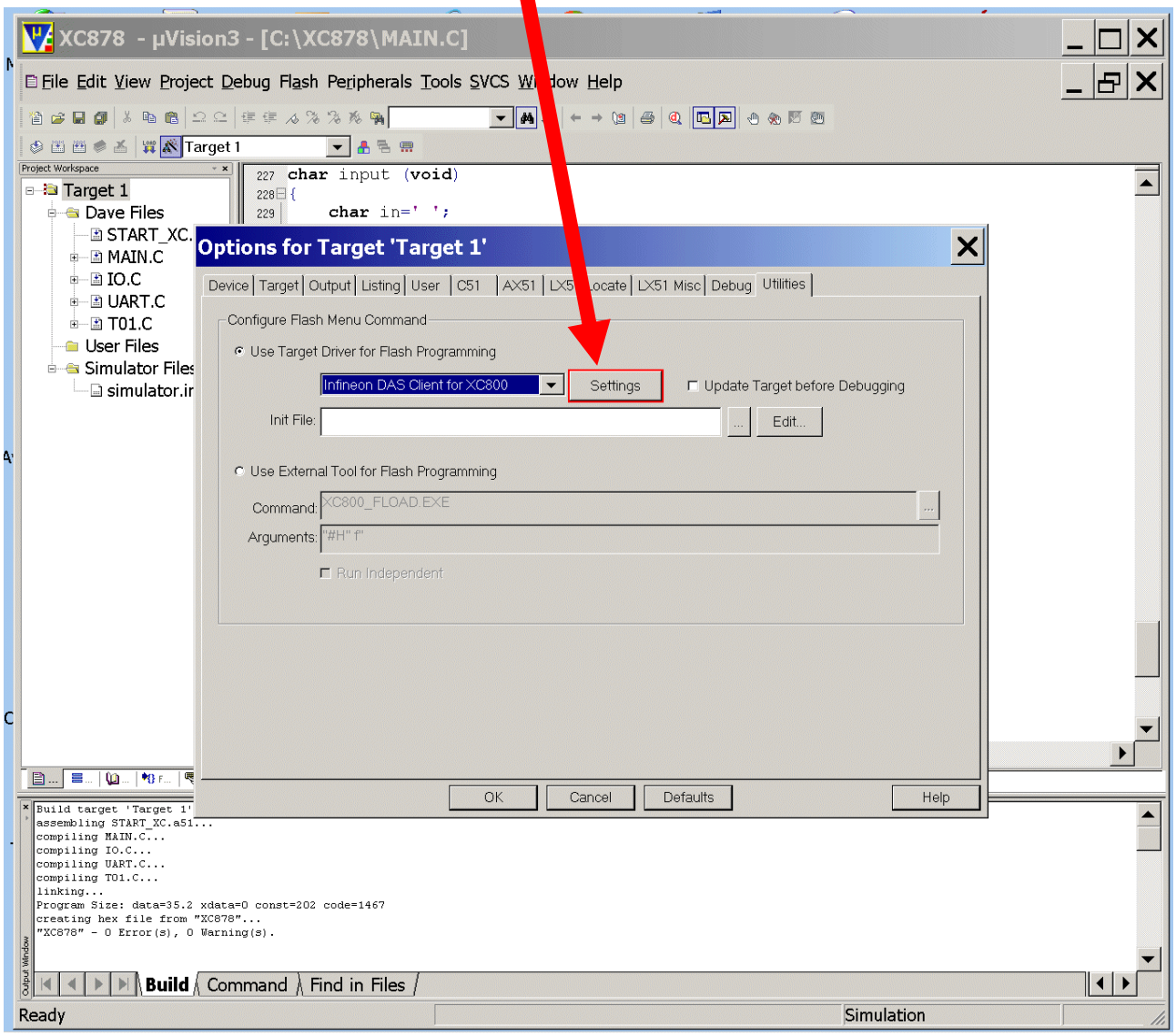
mouse position: (Project Workspace, Files): Target1 click right mouse button Options for Target ´Target1‘	or	click
		

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

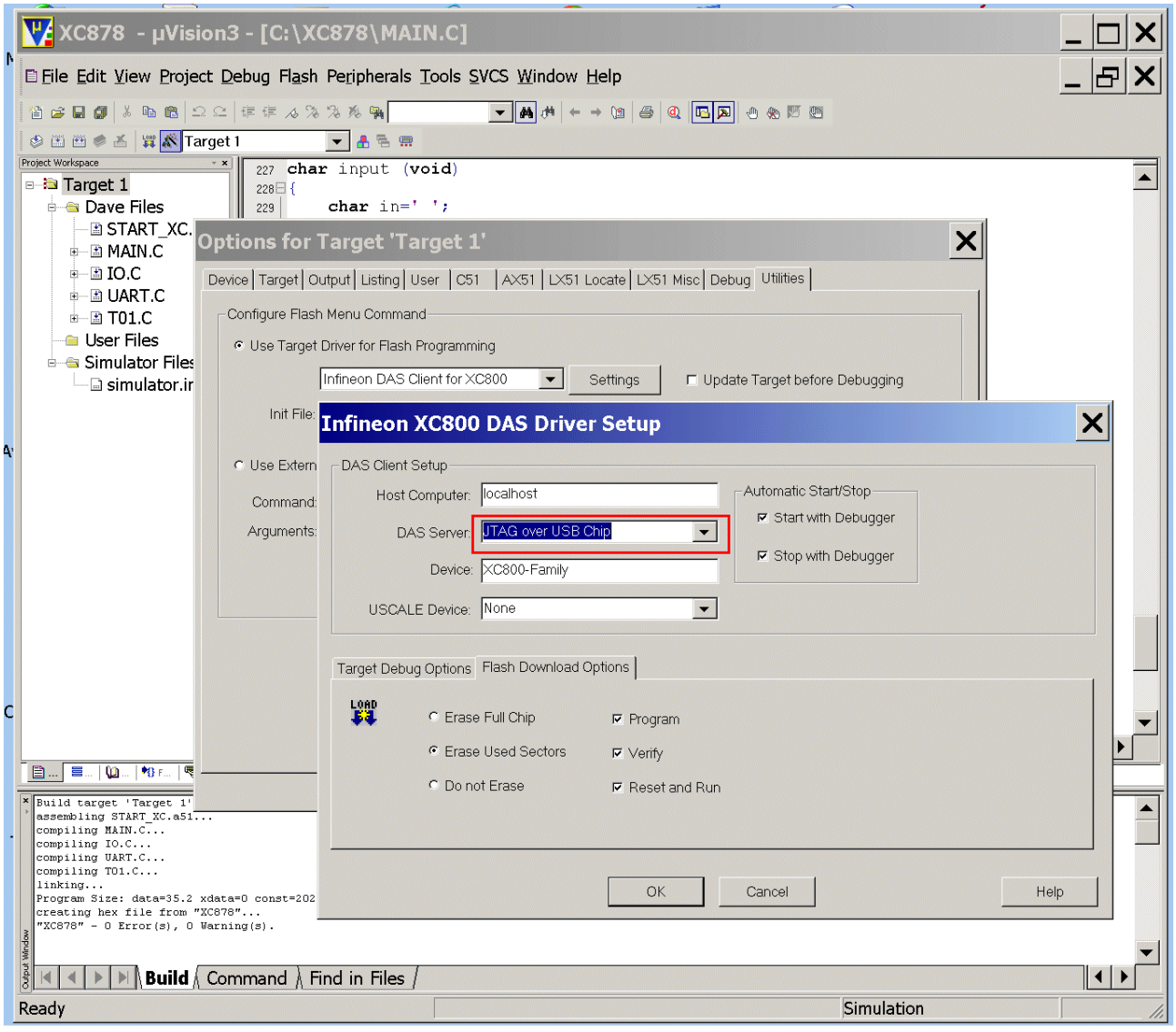


Options for Target ´Target1‘: Utilities:
Configure Flash Menu Command: click Settings



Infineon XC800 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

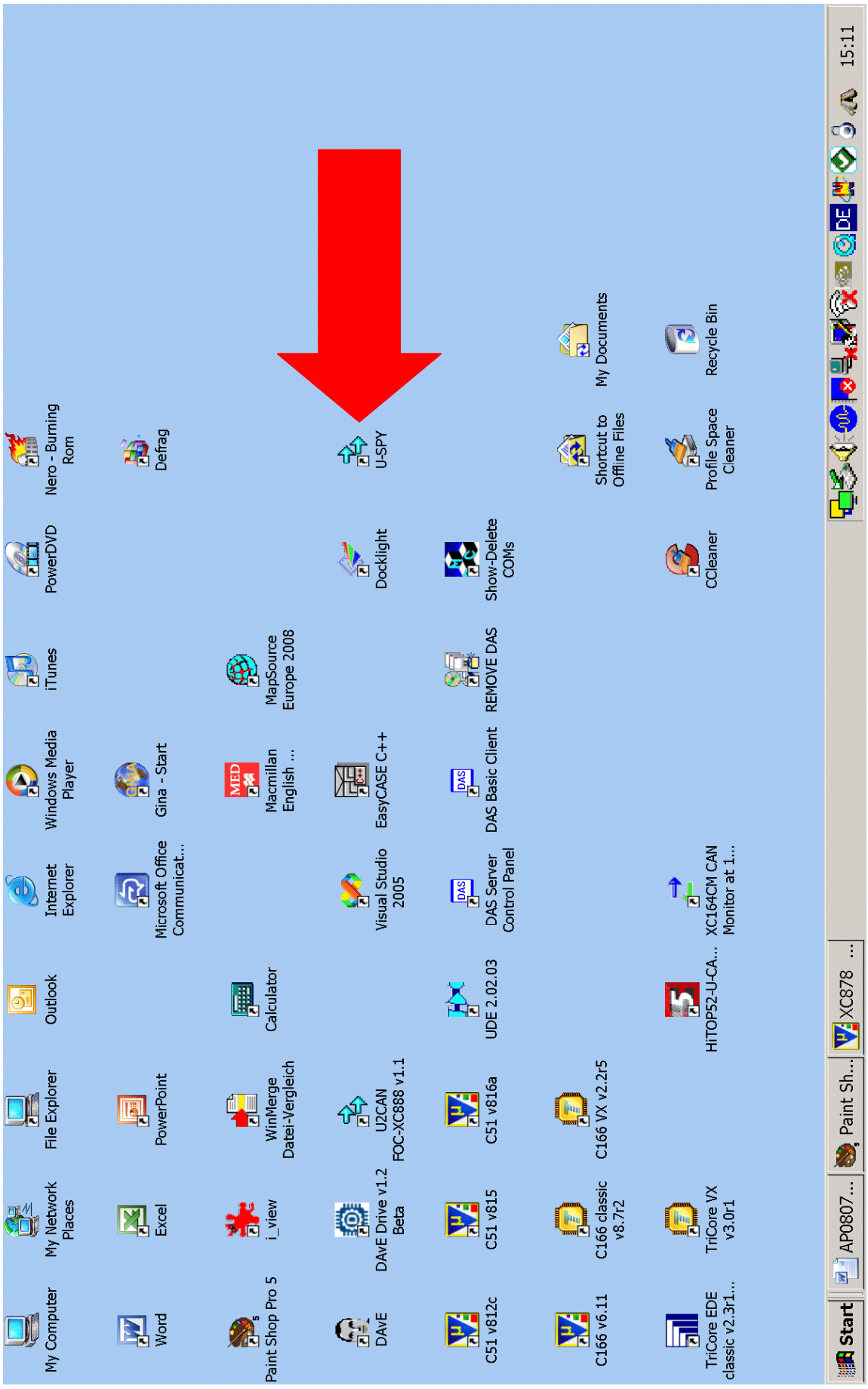
Install U-SPY:

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.

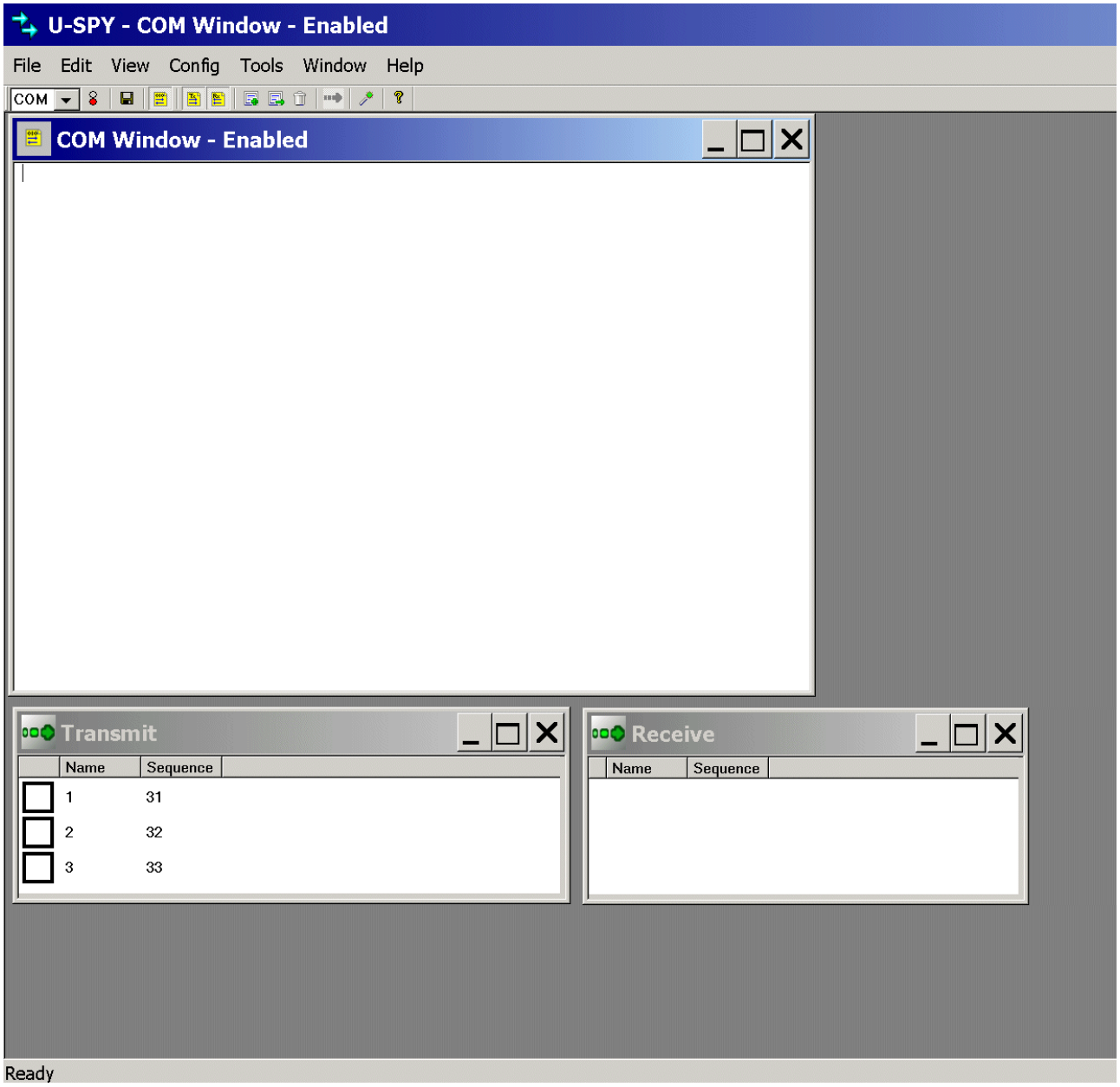
U-SPY can be found either on the XC878 Easy Kit CD (USPY_install.exe)

or on your microcontroller DVD.

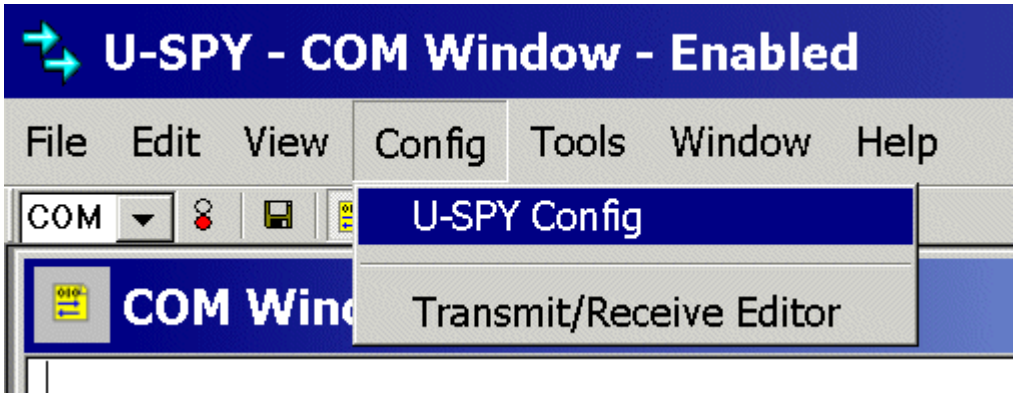
To install U-SPY execute USPY_install.exe


My Computer
My Network Places
File Explorer
Outlook
Internet Explorer
Windows Media Player
iTunes
PowerDVD
Nero - Burning Rom
Word
Excel
PowerPoint
Microsoft Office Communicat...
Gina - Start
Defrag
Paint Shop Pro 5
i_view
WinMerge Datei-Vergleich
Calculator
Macmillan English ...
MapSource Europe 2008
DAvE
DAvE Drive v1.2 Beta
U2CAN FOC-XC888 v1.1
Visual Studio 2005
EasyCASE C++
Docklight
U-SPY
C51 v812c
C51 v815
C51 v816a
DAS Server Control Panel
DAS Basic Client
REMOVE DAS
Show-Delete COMs
C166 v6.11
C166 classic v8.7r2
C166 VX v2.2r5
UDE 2.02.03
CCleaner
Shortcut to Offline Files
My Documents
TriCore EDE classic v2.3r1...
TriCore VX v3.0r1
HiTOP52-U-CA...
XC164CM CAN Monitor at 1...
Profile Space Cleaner
Recycle Bin
Start
AP0807...
Paint Sh...
XC878 ...
15:11

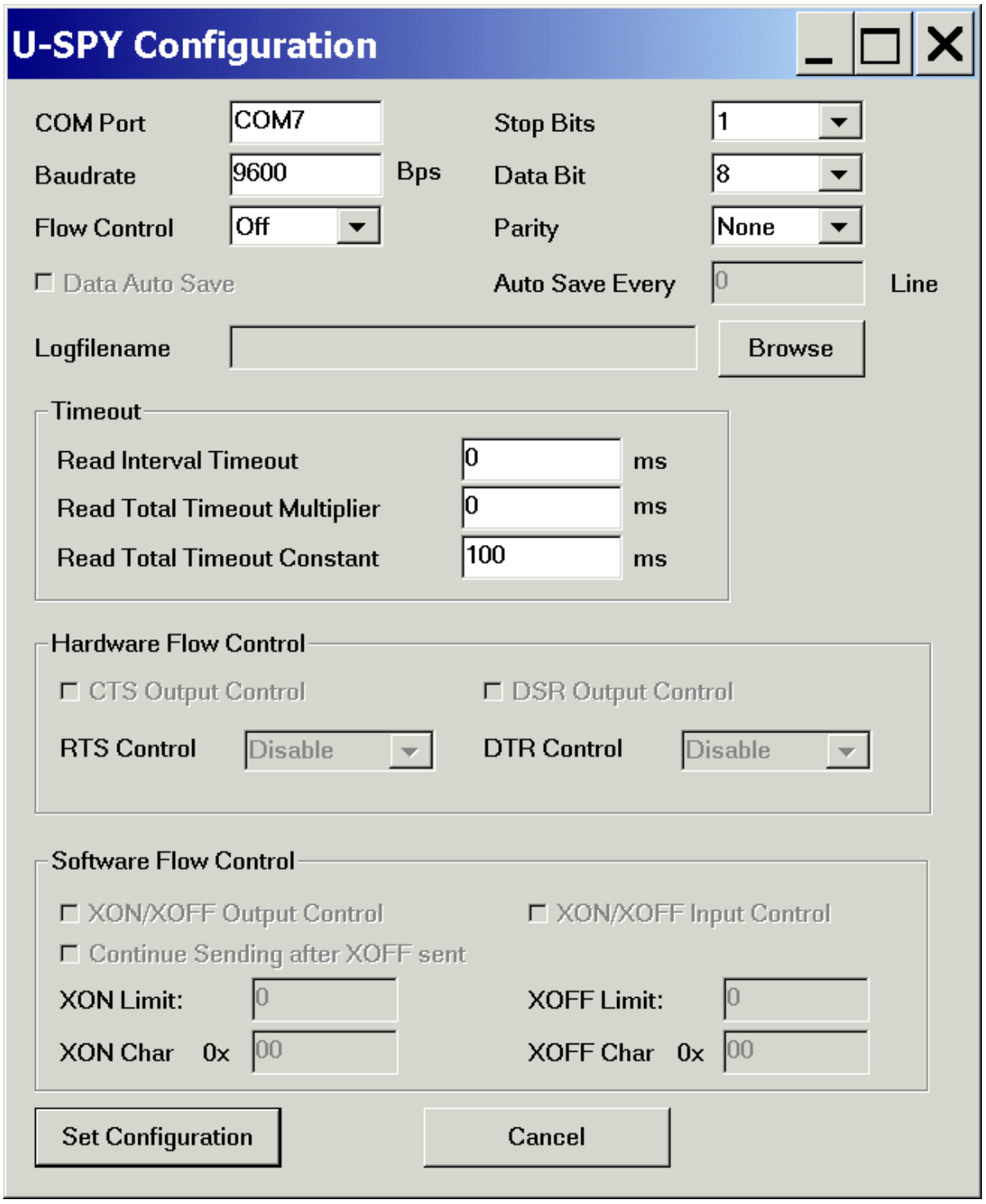
Now, start U-SPY:



Config – U-SPY Config

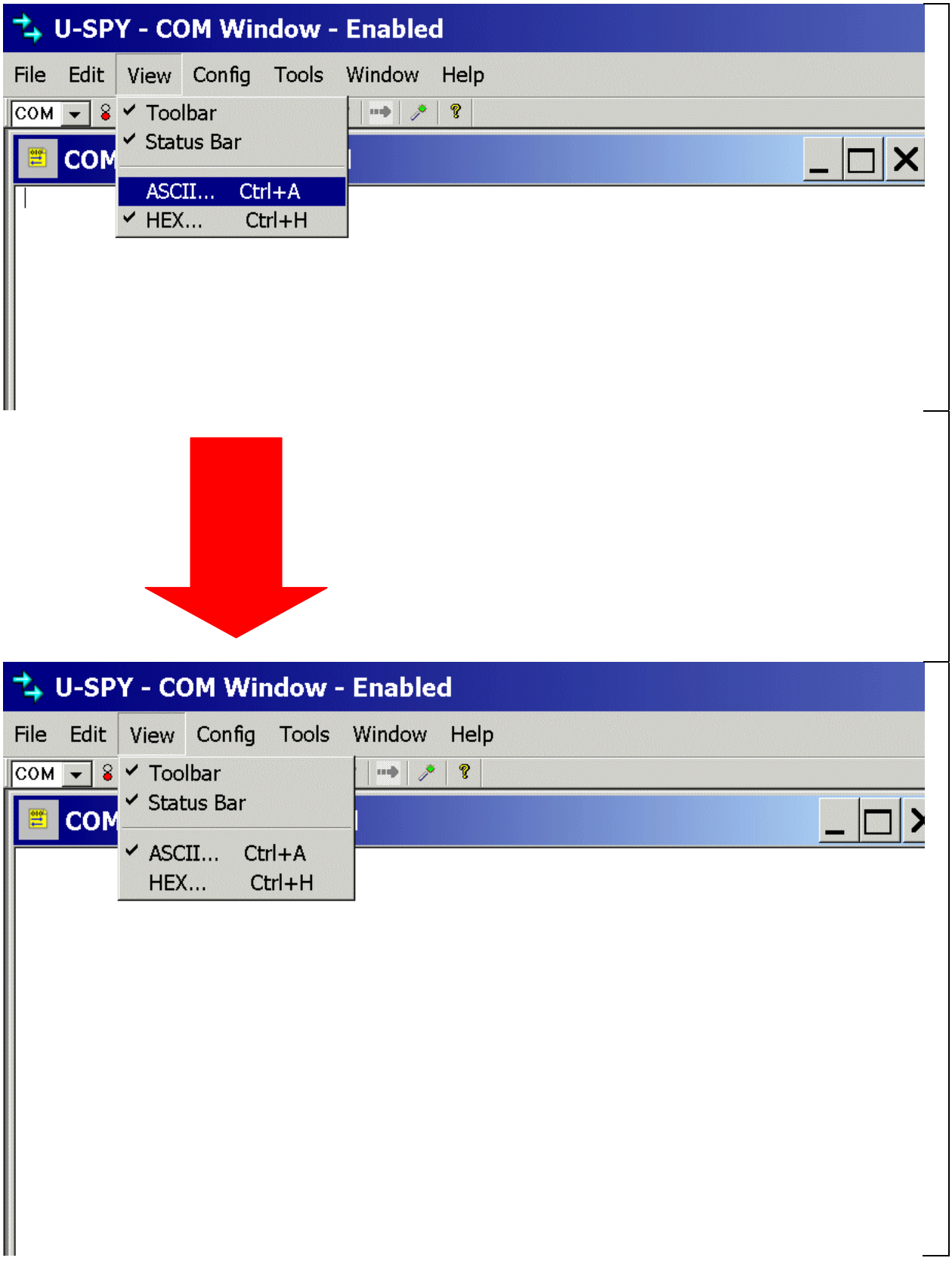


COM Port check/insert COM7
Baudrate check/insert 9600
Flow Control check/insert Off
Stop Bits check/insert 1
Data Bit check/insert 8
Parity check/insert None

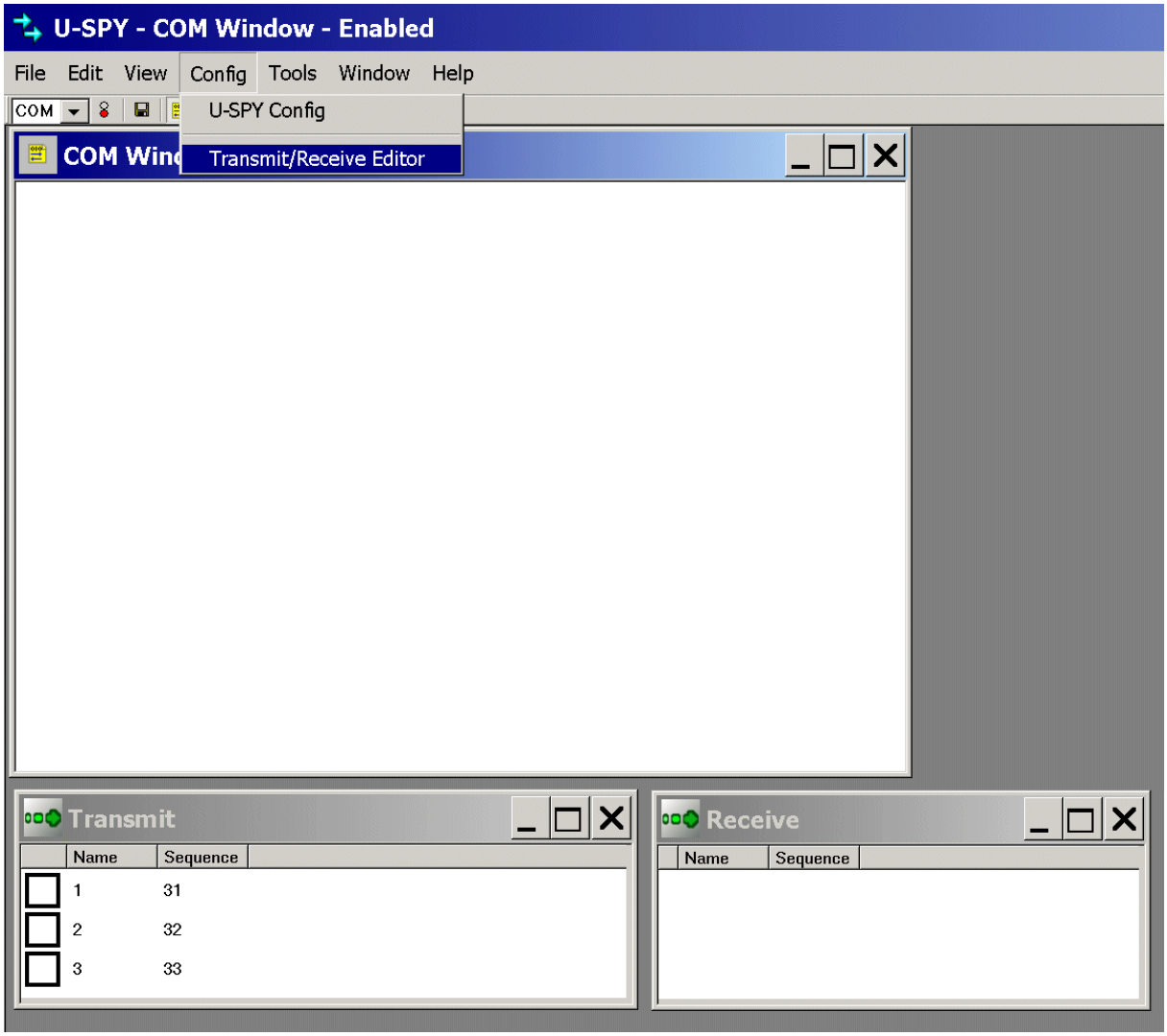


Click Set Configuration

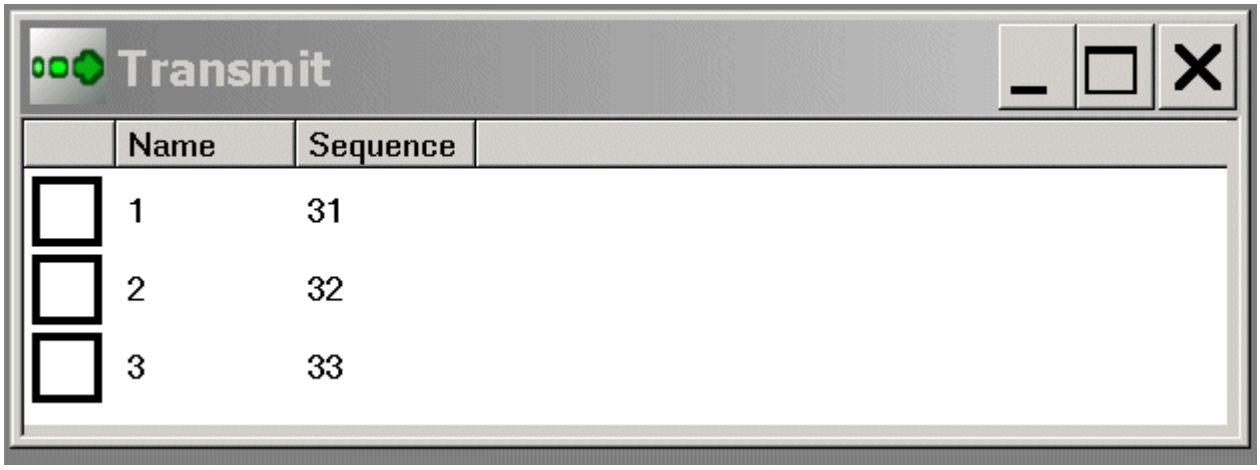
View – select ASCII…

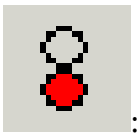


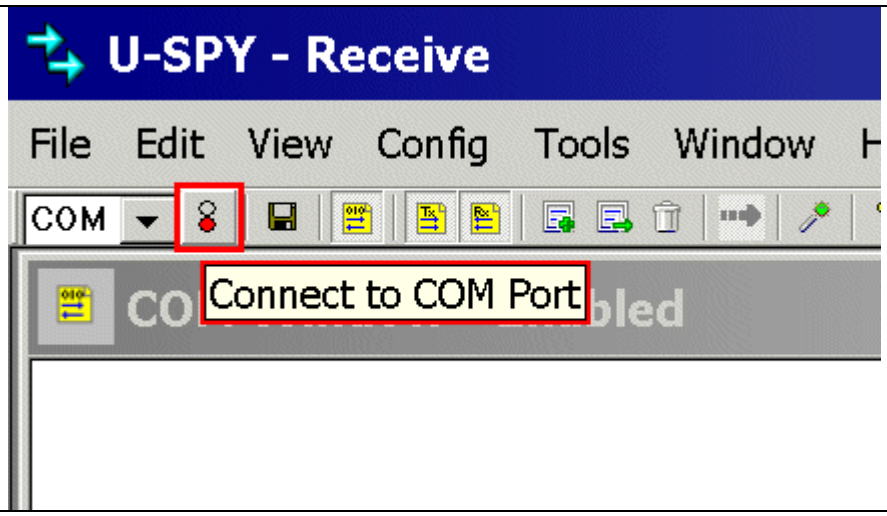
Config – Transmit/Receive Editor

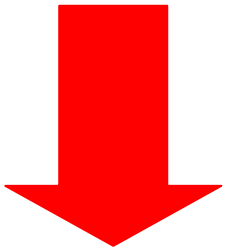


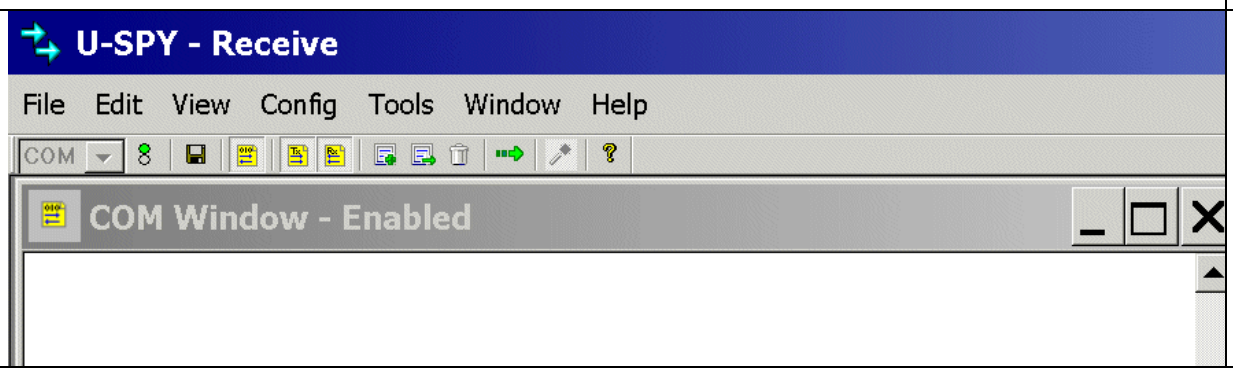
With the Transmit/Receive Editor configure the Transmit Window so that it appears like the Transmit window below:



Click  :



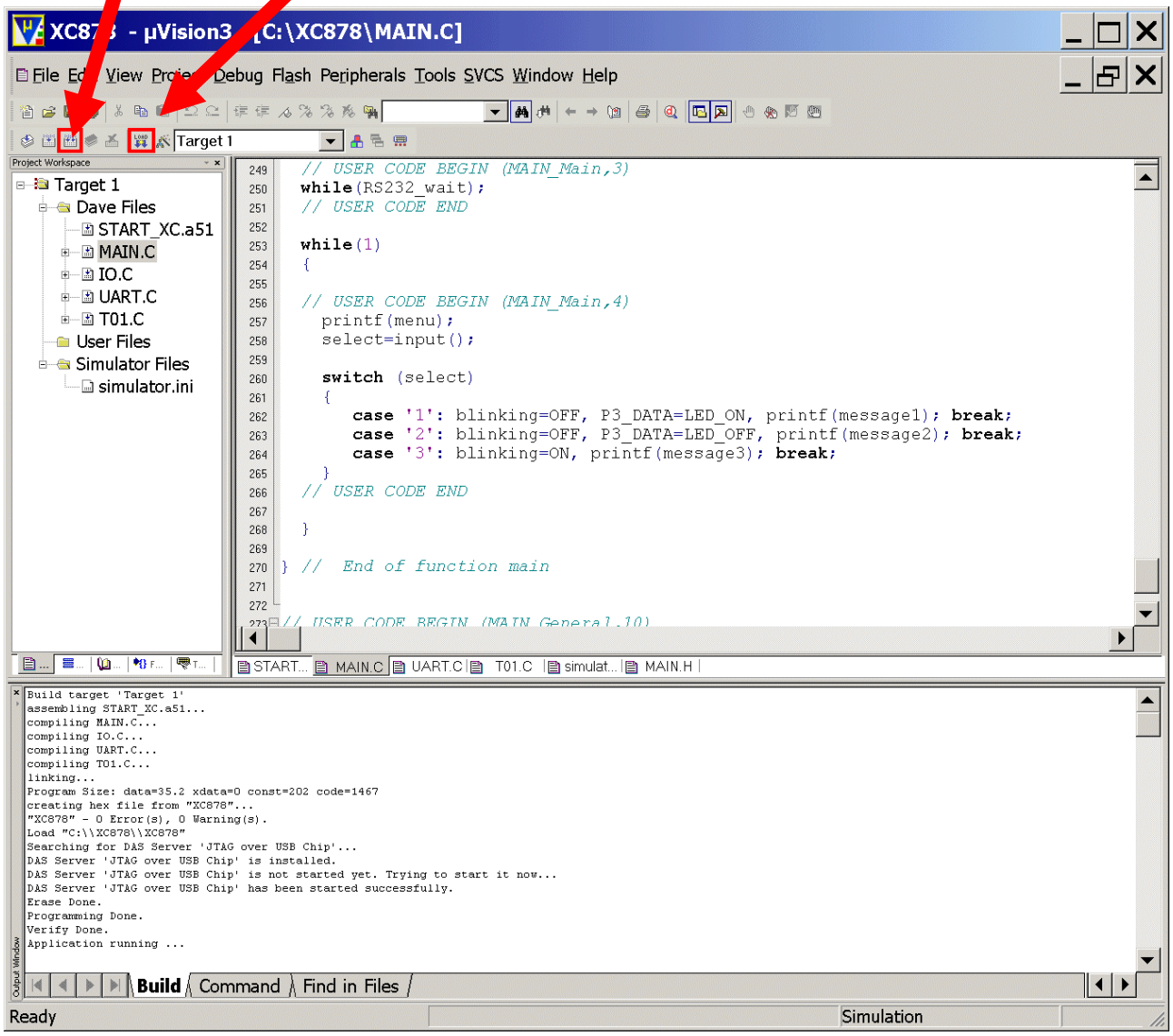




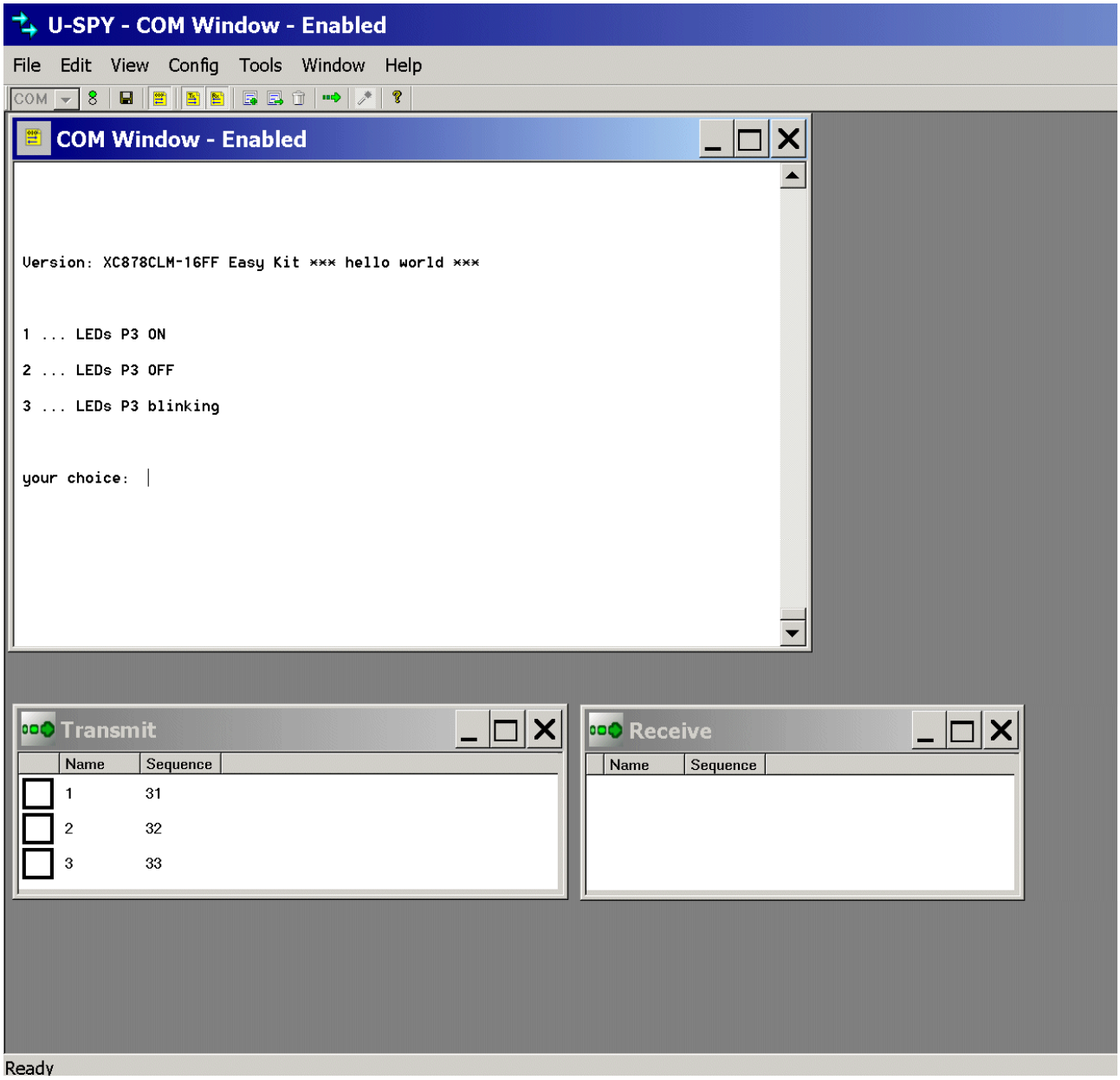
Note:
U-SPY is now ready for serial communication!

Go back to µVision:

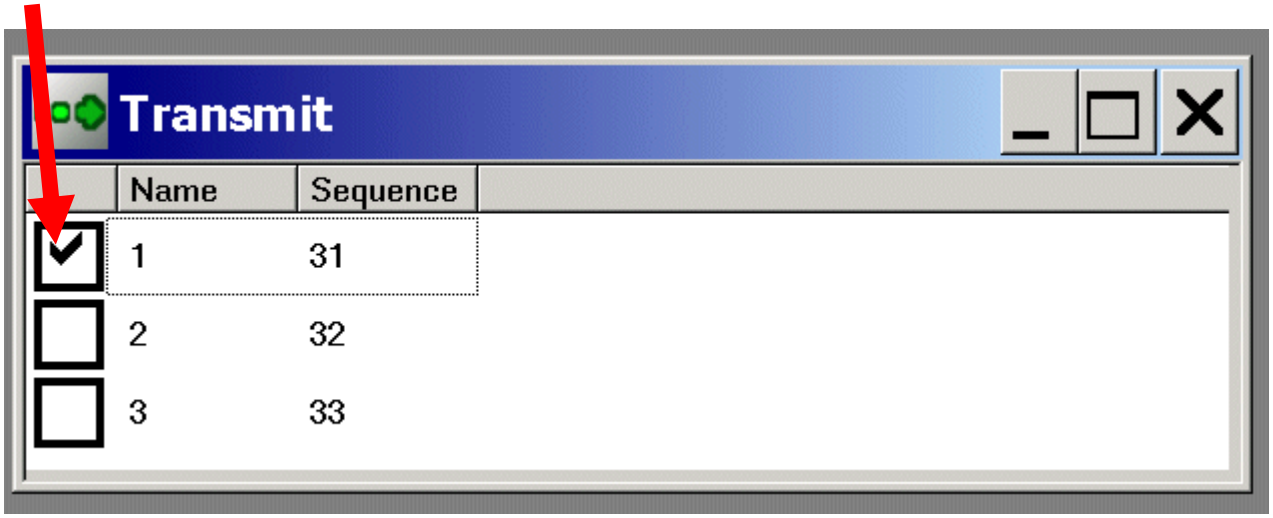
1.) click:  2.) click: 



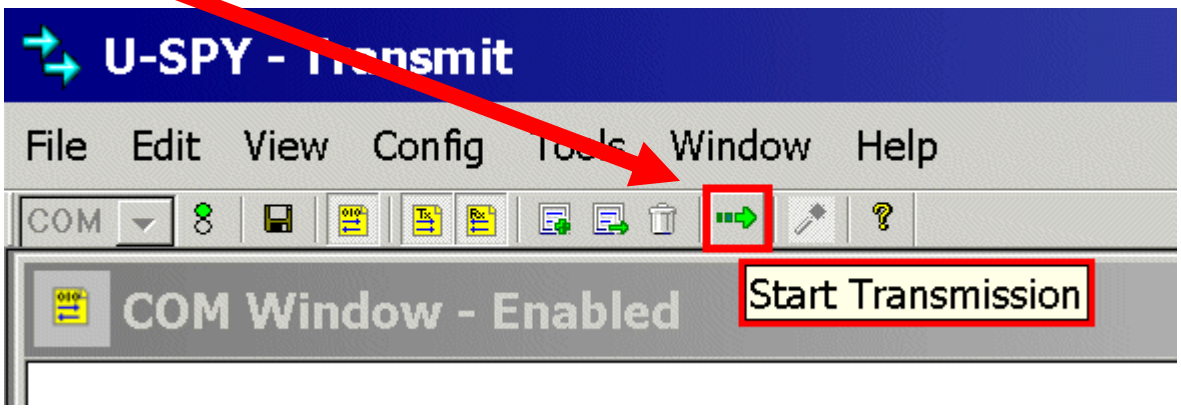
Go back to U-SPY and see the result:



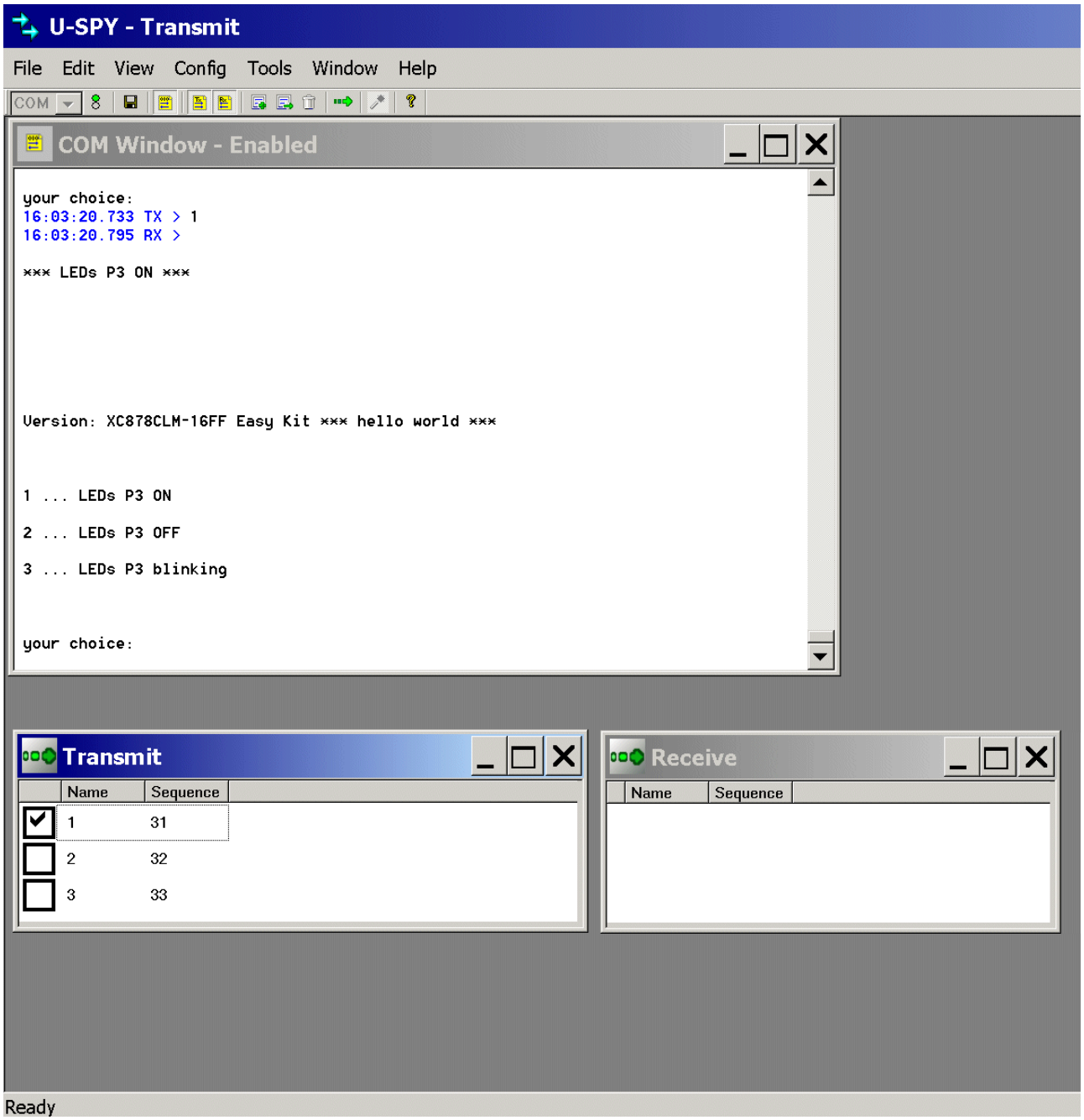
Tick ☑ 1:



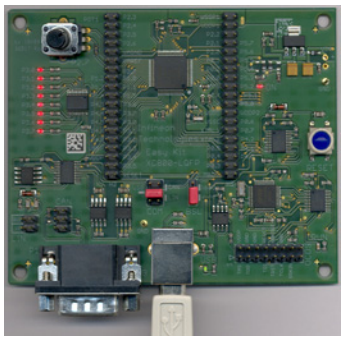
Click 



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.

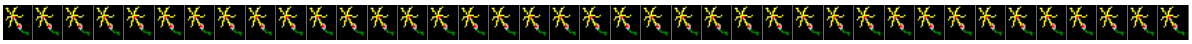
This should give you a good opportunity to get in touch with another Infineon microcontroller family or tool chain!

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.

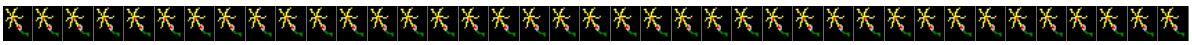
8.) Feedback (XC878 Easy Kit, Keil tools): Your opinion, suggestions and/or criticisms



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

Published by Infineon Technologies AG