

AP16133

XE167

XE166 "Cookery-Book" for a "Hello world" application. You can do the "Hello world" example in this document with the evaluation version of the KEIL tool chain.

Microcontrollers



Never stop thinking

Edition 2008-07-16

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

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

LEGAL DISCLAIMER

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

Information

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

Warnings

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

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

AP08048

Revision History: 2008-01 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 an XE167 Easy Kit Board / Evaluation Board / Starter Kit Board up and running.

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

The purpose of this document is to gain know-how of the microcontroller and the tool-chain.
Additionally, the "hello-world-example" can easily be expanded to suit your needs.
You can connect either a part of - or your entire application to the Easy Kit Board.
You are also able to benchmark any of your algorithms to find out if the selected microcontroller fulfils all the required functions within the time frame needed.

Note:

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

Have fun and enjoy the XE167 microcontrollers!

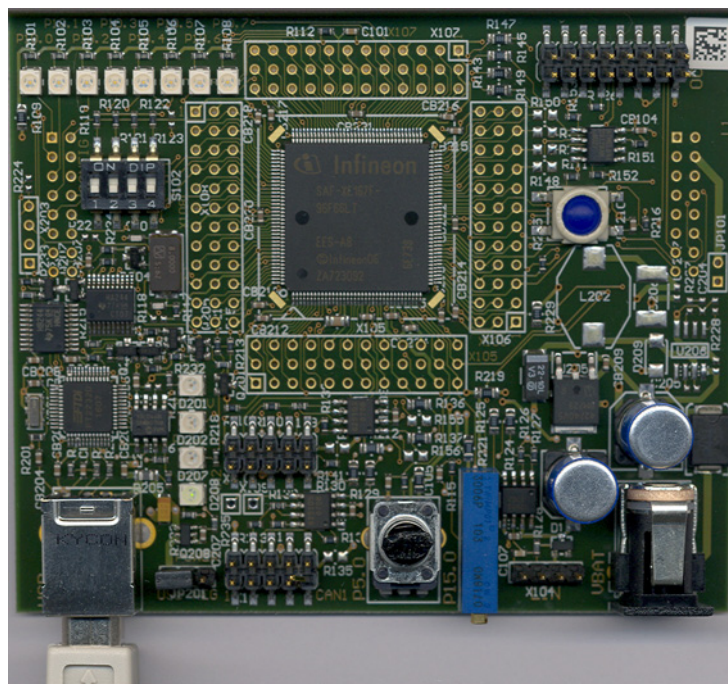


Programming Example

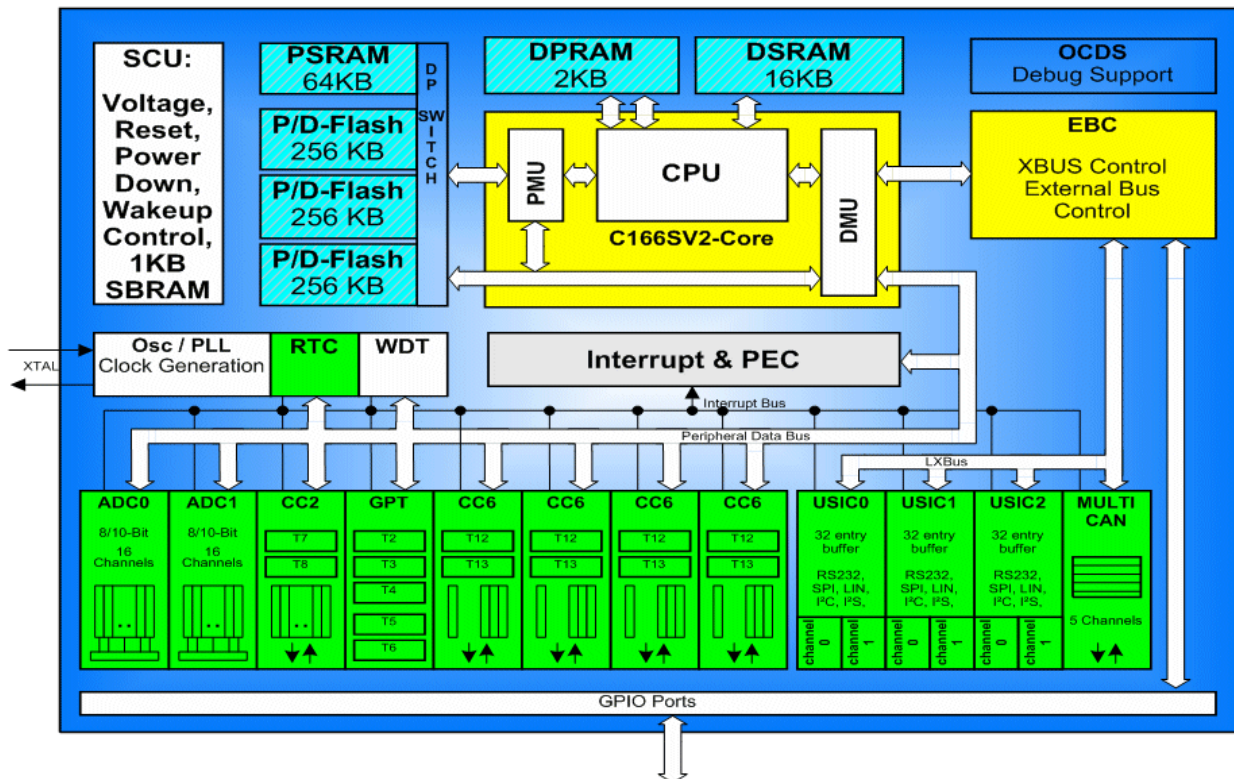
SAF-XE167F-96F80L /

SAK-XC2287-96F80L

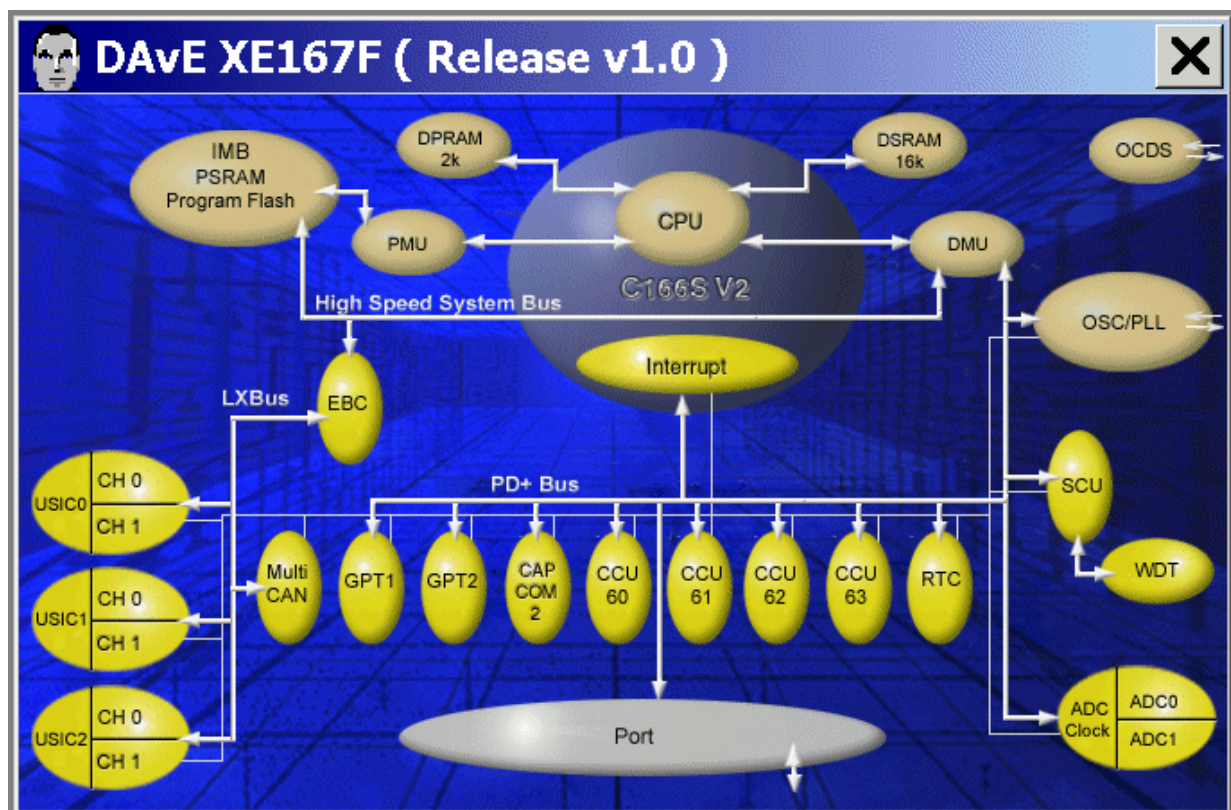
Easy Kit



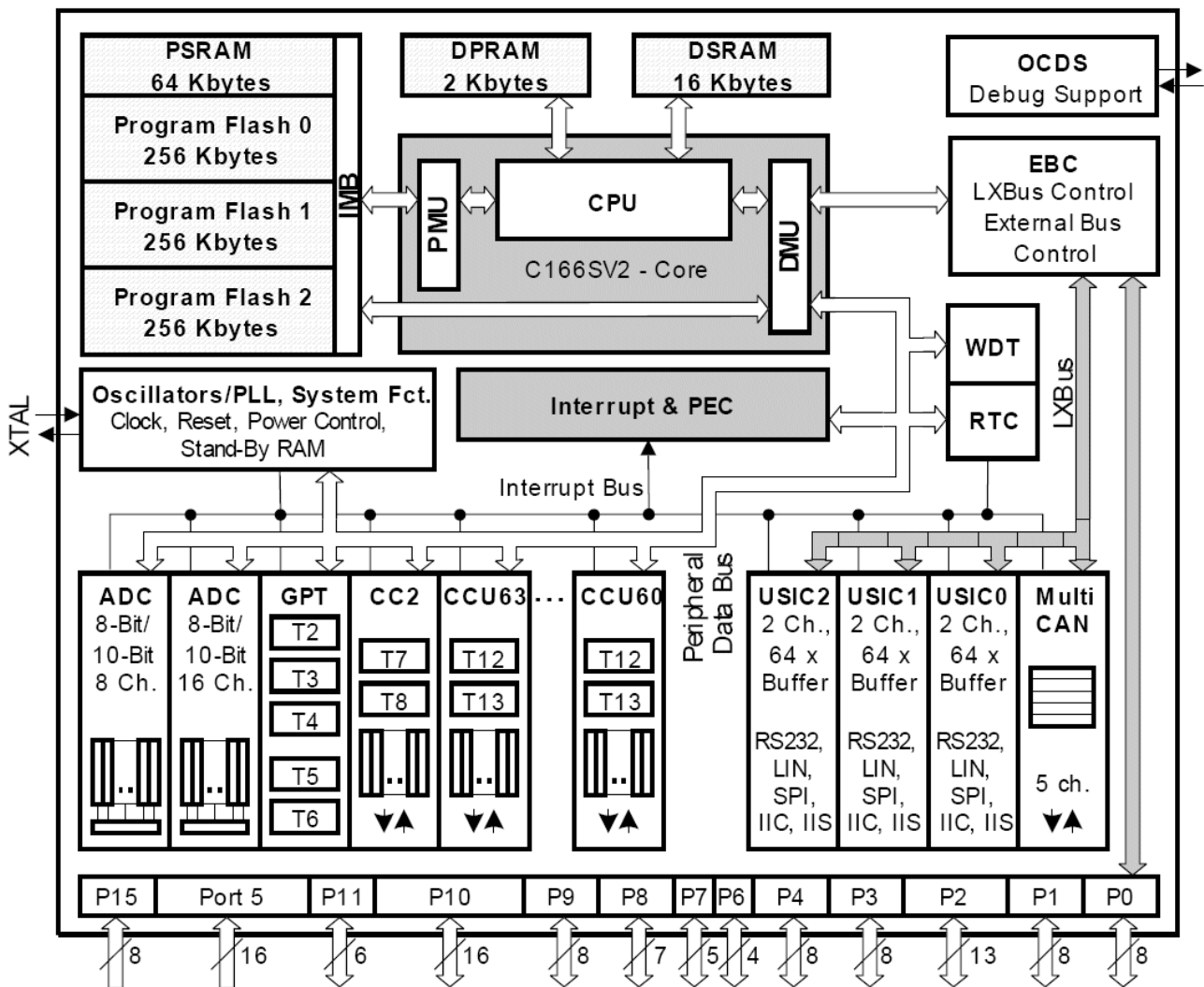
SAF-XE167F-96F66L **Block Diagram** (Source: Product Marketing)



SAF-XE167F-96F66L **Block Diagram** (Source: DAVe)



SAF-XE167F-96F66L **Block Diagram** (Source: User's Manual)



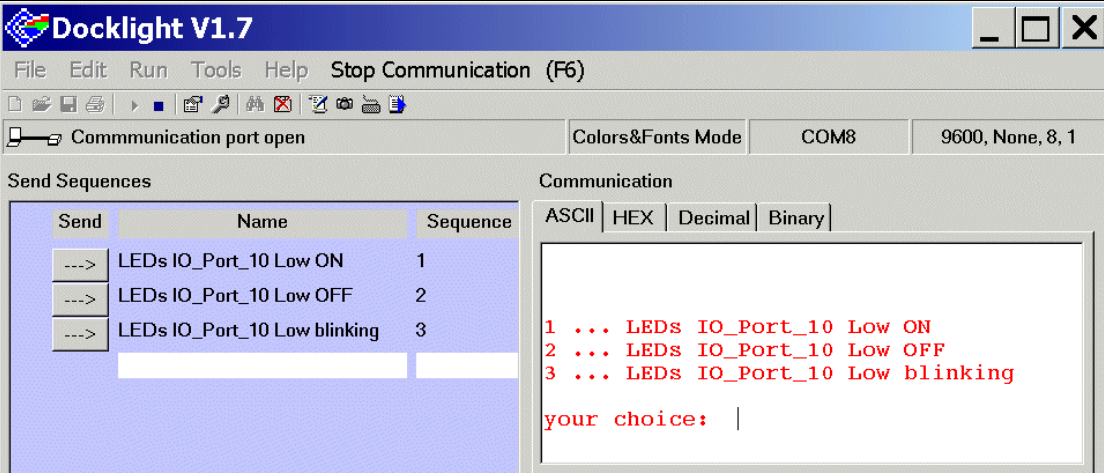
Note:

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 XE167 Easy Kit:

Your program:	
Chapter/ Step	*** Recipes ***
1.)	XE167 Easy Kit Board Power Supply (via USB), Jumper Setting, Serial Connection (via USB) to the host computer
2.)	DAvE (program generator) DAvE Installation (mothersystem) + DAvE Update Installation (XE167.DIP) for XE167
3.)	Using DAvE Microcontroller initialization for your programming example
4.)	Using the KEIL Development Tools (C-Compiler) Programming of your application (XE167) with KEIL tool chain (µVision3) Compiler v6.10a
5.)	Running your first programming example Using real hardware (+ DAS Installation + OnChipFlash-Programming)

Feedback

6.)	Feedback
-----	--------------------------

1.) XE167 Easy Kit Board:



Ordering information:

Order Code:	μC
Easy Kit XE167F	SAK-XE167CM
Easy Kit XE164F	SAK-XE164CM

Distribution Worldwide:

<http://www.infineon.com/cms/en/corporate/company/location/index.html>


Screenshot of the XE167 Easy Kit Homepage:

Easy Kit XE167F - Infineon Technologies - Infineon Technologies

File Edit View Favorites Tools Help

Back
Forward
Home
Search
Folders
Favorites
Print
Print & Preview
Stop
Reload
Address Bar
Go

Address
http://www.infineon.com/cms/en/product/channel.html?channel=db3a3043156fd573011600a601
Go


Never stop thinking

Home | Sitemap | Select Language | Login

Site Search

Go

About Infineon >>

Get Product information

Select a Category

Find Products

Search

Microcontrollers

Home > Microcontrollers > Development Tools, Software and Training > XE166 Development Tools and Software > Easy Kits > Easy Kit XE167F

Print Page
Send Page


Easy Kit XE167F

MCU Derivate: XE167F-96F66F

CPU Clock: 66 MHz

On-Chip Memory:
82 kByte SRAM,
768 kByte FLASH

One USB Cable for
- Power supply
- Virtual COM Port
- Download & Debug Interface



Includings:
- Easy Kit Hardware
- Getting Started, first 3 Steps to set up your Hardware, install the Tools and Debug the first Program
- Technical Documentation: e.g. User manuals (System unit and Peripheral unit), Architecture manual, Application notes, Errata sheets, Data Sheets, Board Documentation (pdf-version)
- Evaluation Versions of Development Tools: e.g. Compiler, Debugger, DAVe Mother System v2.1 and Derivative Implementation Files (DIP), Memtool (Flash Programming Tool)
- Training Documentation, Demo Programs, Hands On Training (HOT).

Interfaces :
- One USB connector for ASC0 Interface via virtual COM port, JTAG (OCDS Level 1) and Power Supply
- 4 pin header for LIN Transceiver (ASC1)
- 16-pin header for JTAG interface (OCDS)
- 2x10pin Header for CAN High Speed Transceiver (CAN1/CAN2)
- Easy access to all pins

Ask Infineon!

International Toll Free:
0(0)800 951 951 951
Direct Access:
+49 89 234 65555
Infineon is happy to help you:
▶ Infineon Service Center

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

Components:

- FTDI2232D for Serial and JTAG connection, plus Power Supply 5V
- Two LEDs to validate the USB enumerate State and Debug RUN state
- Optional selectable Step down Voltage Regulator TLE6365G to supply 5V
- LEDs to validate power supply (5Volt)
- LED indicating JSTIN active state
- 2 x CAN-Transceiver TLE 6250G
- AT25128N SPI EEPROM
- LIN Transceiver TLE 7259G
- 8 general purpose LEDs
- Reset switch

To browse through the latest version of the Easy Kit CD, please click [here](#).

Order Code: Easy Kit XE167F

Price*: 129,- EUR

How to order?

[Buy Online](#)

* recommended retail price. Valid from 2007-Nov-01.

[Documents](#)

[Contact us](#)

Jump to Document Type

[User's Manual](#)

[Show all documents](#)

Title	Date	Version	Size
User's Manual			
 XE166 Family Easy Kit Manual (XE166Board_V10.pdf)	01 Oct 2007	v1.0	2 MB

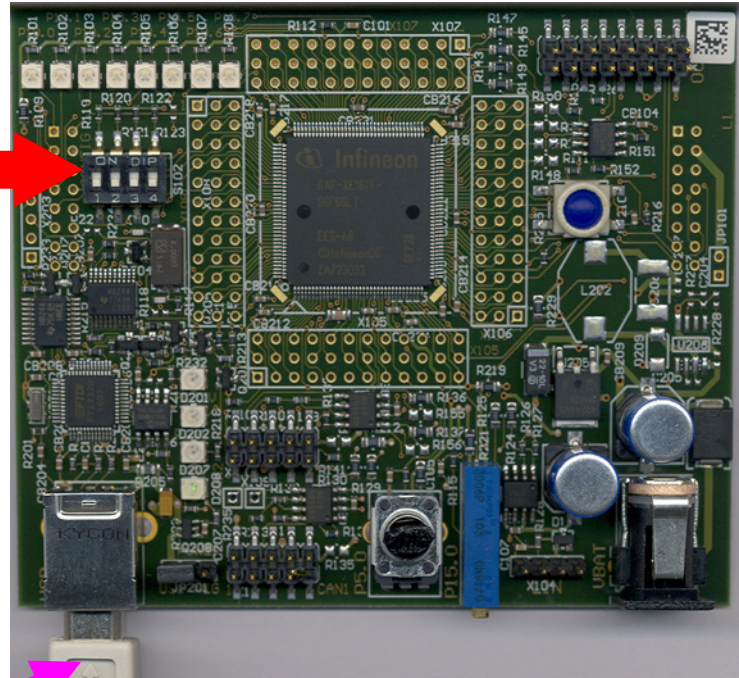
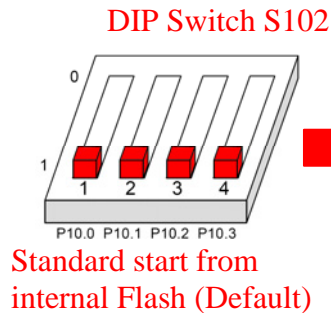
[Home](#) | [Company](#) | [Investor](#) | [Press](#) | [Careers](#) | [Infineon worldwide](#)

© 1999 - 2006 Infineon Technologies AG - Usage of this website is subject to our [Usage Terms](#) - [Imprint](#) - [Contact](#) - [Privacy Policy](#)



Trusted sites

Overview of the XE167 Easy Kit Board connection to the environment:



USB Cable:

.) used for: UART communication (the UART/RS232/ASC0/USIC0_CH0 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:

Do not connect now!

This is just information! We are going to connect the board later!



Reason:

By connecting the Easy Kit Board to the computer a USB driver is needed.

Note:

[For further information, please refer to the XE166 family Easy Kit Manual, V1.0, Oct 2007 .](#)

2.) DAvE – Installation for XE167 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 XE167 microcontroller support/update (XE167 DIP file):

1.)

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


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

Title	Date	Version	Size
Development Tools			
 XE16xx-Series DIP file for DAVe (Microcontroller Configuration Tool) (XE16xx_Series_v1.0.zip)	09 Jan 2008	v1.0	4.2 MB

Unzip the zip-file “XE16xx_Series_v1[1].0.zip” and save “XE16xx_Series.dip“

@ e.g. C:\DAvE\XE167-2008-01-15\XE16xx_Series.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: C:\DAVe\XE167-2008-01-15

Forward>

Select: Available Products

click ✓ XE16xx_Series

Forward>

Install

End

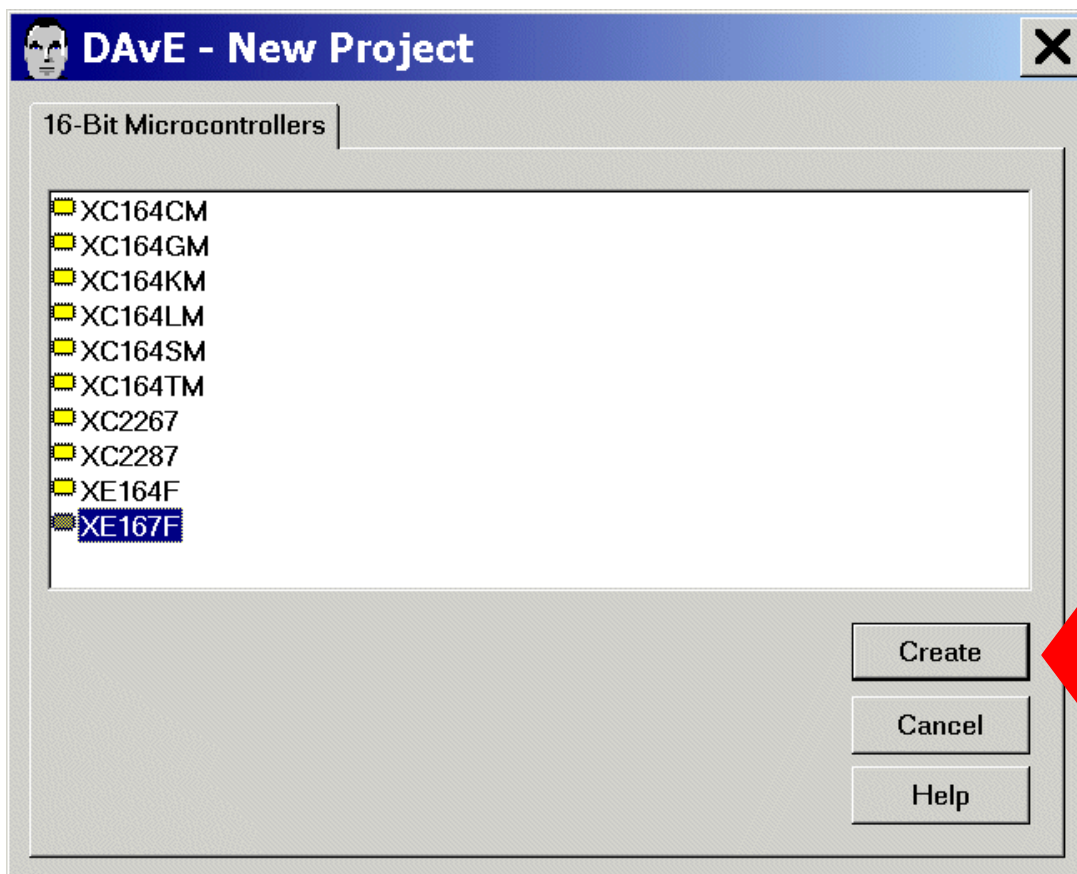
4.) DAVe is now ready to generate code for the XE167 microcontroller.

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



Start the program generator DAVe and select the XE167 microcontroller:

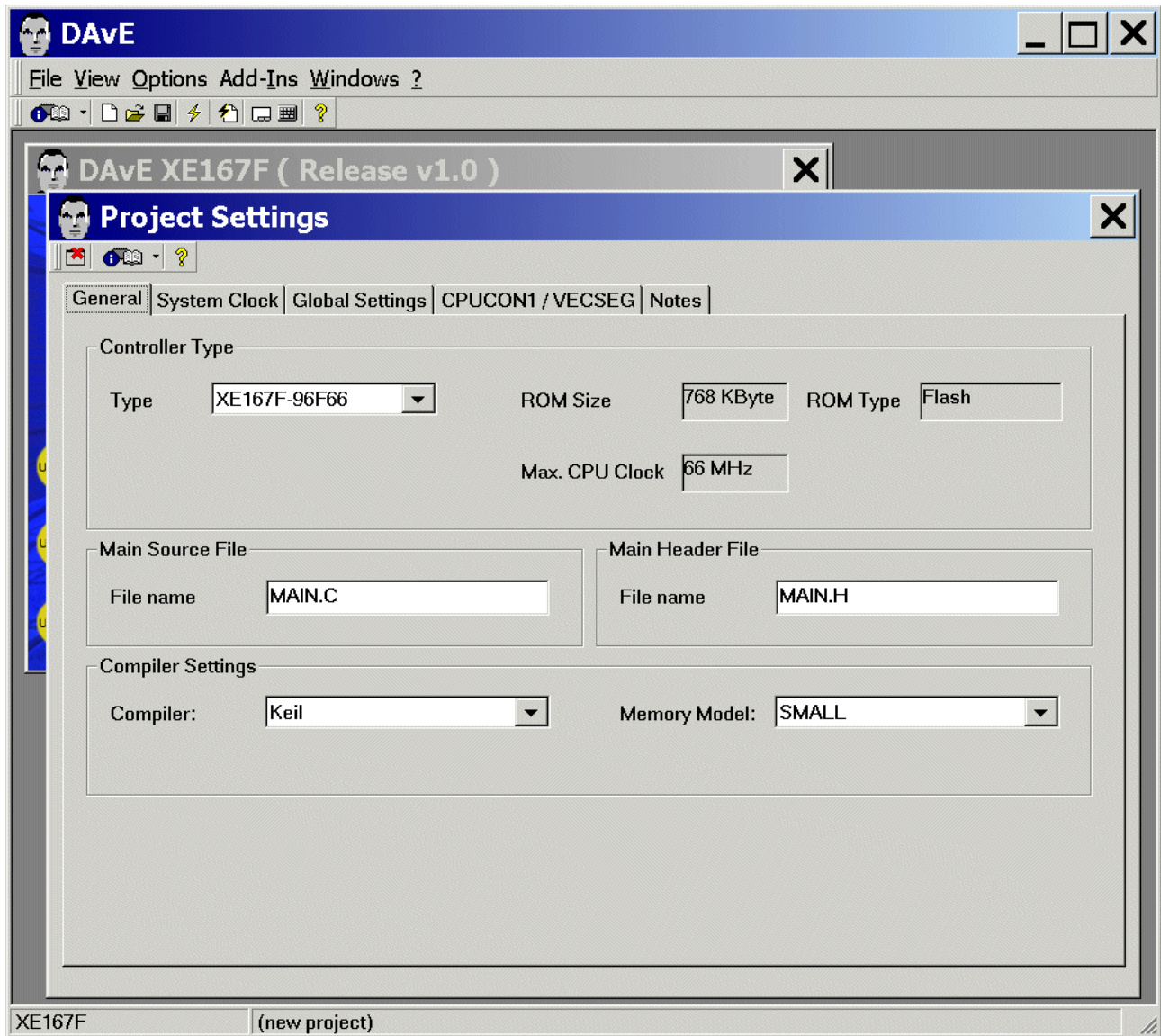
File
New
16-Bit Microcontrollers
select **XE167F**
Create



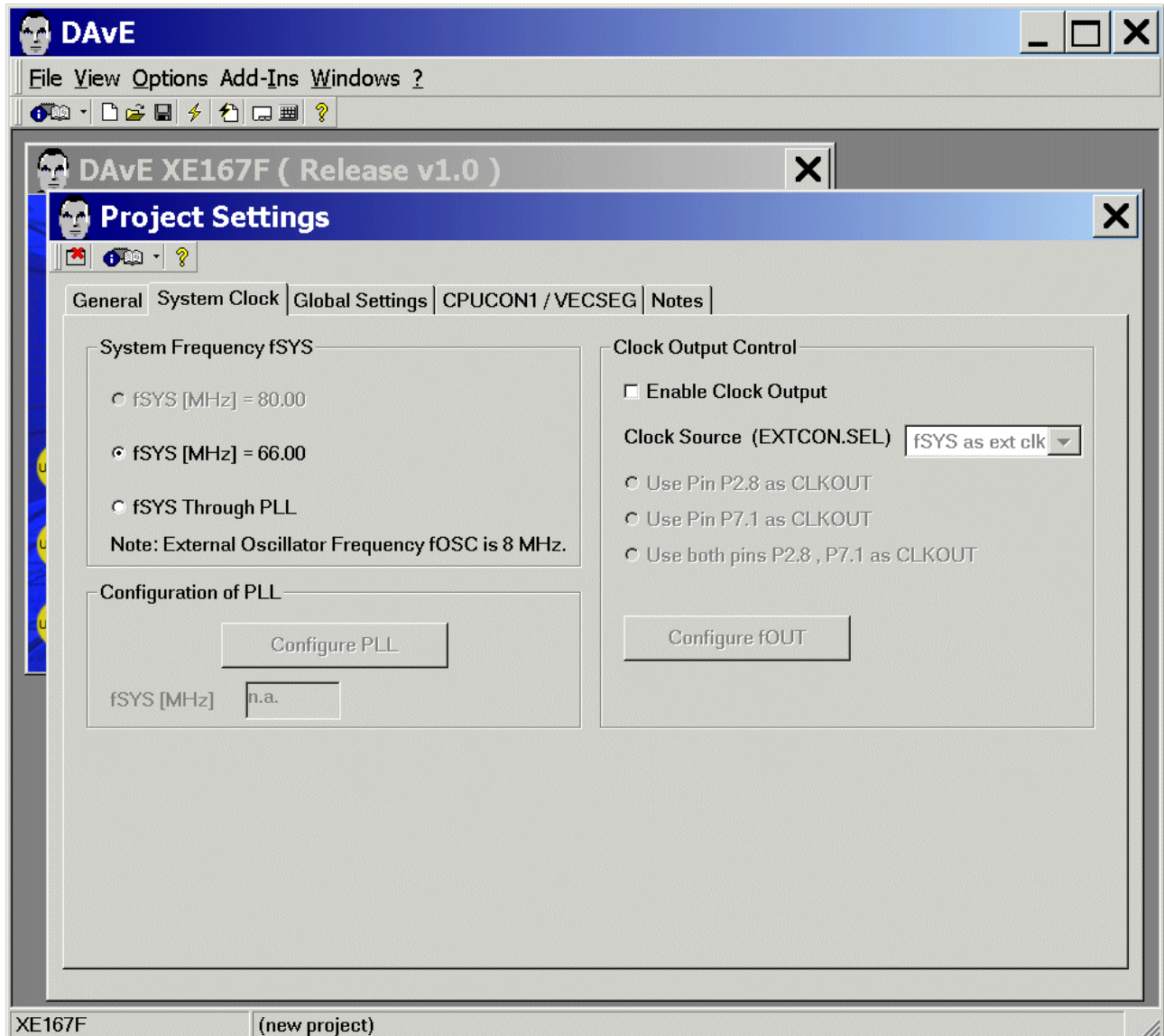
Choose the Project Settings as you can see in the following screenshots:

General: Controller Type: Type: check/select XE167F-96F66

General: Compiler Settings: Compiler: check/choose Keil



System Clock: (do nothing)

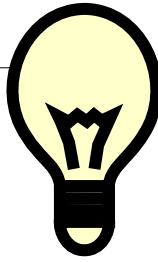


Note (Source: DAVE):

Configuration of the System Clock:

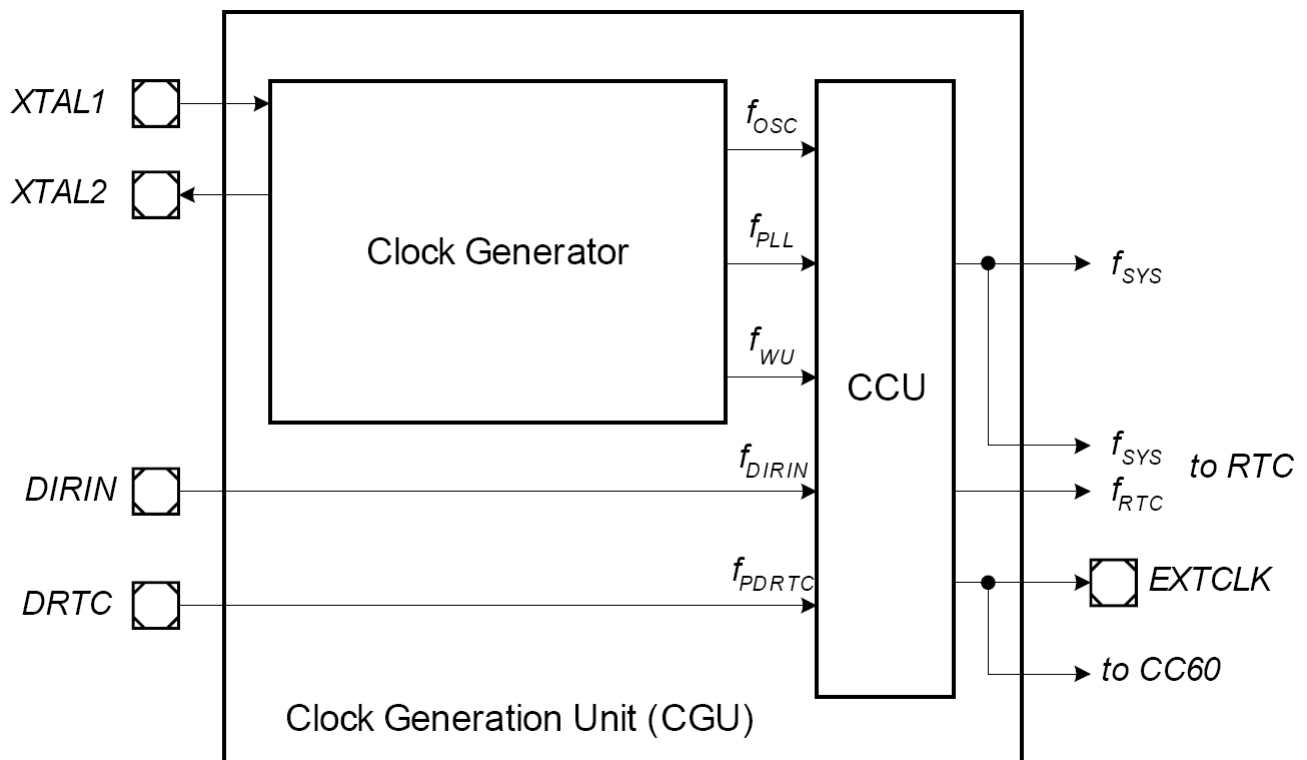
- VCO clock used, input clock is connected
- input frequency is 8,00 MHz (XTAL1)
- configured system frequency is 66,00 MHz
- system clock is 66.00 MHz





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

Clock Generation Unit (CGU) Block Diagram:



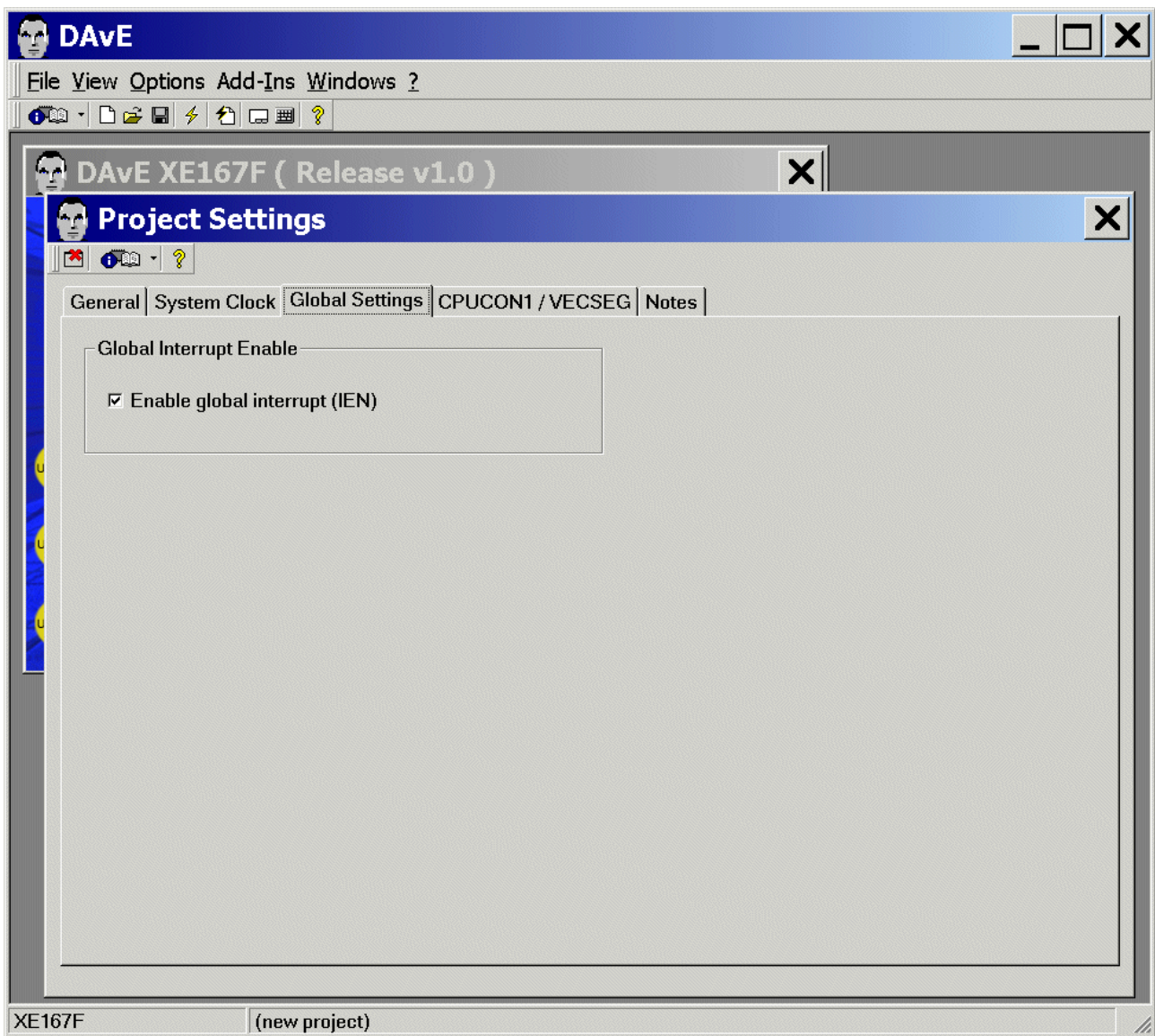
Note:

The CGU can convert a low-frequency external clock to a high-speed internal clock, or can create a high-speed internal clock without external input.

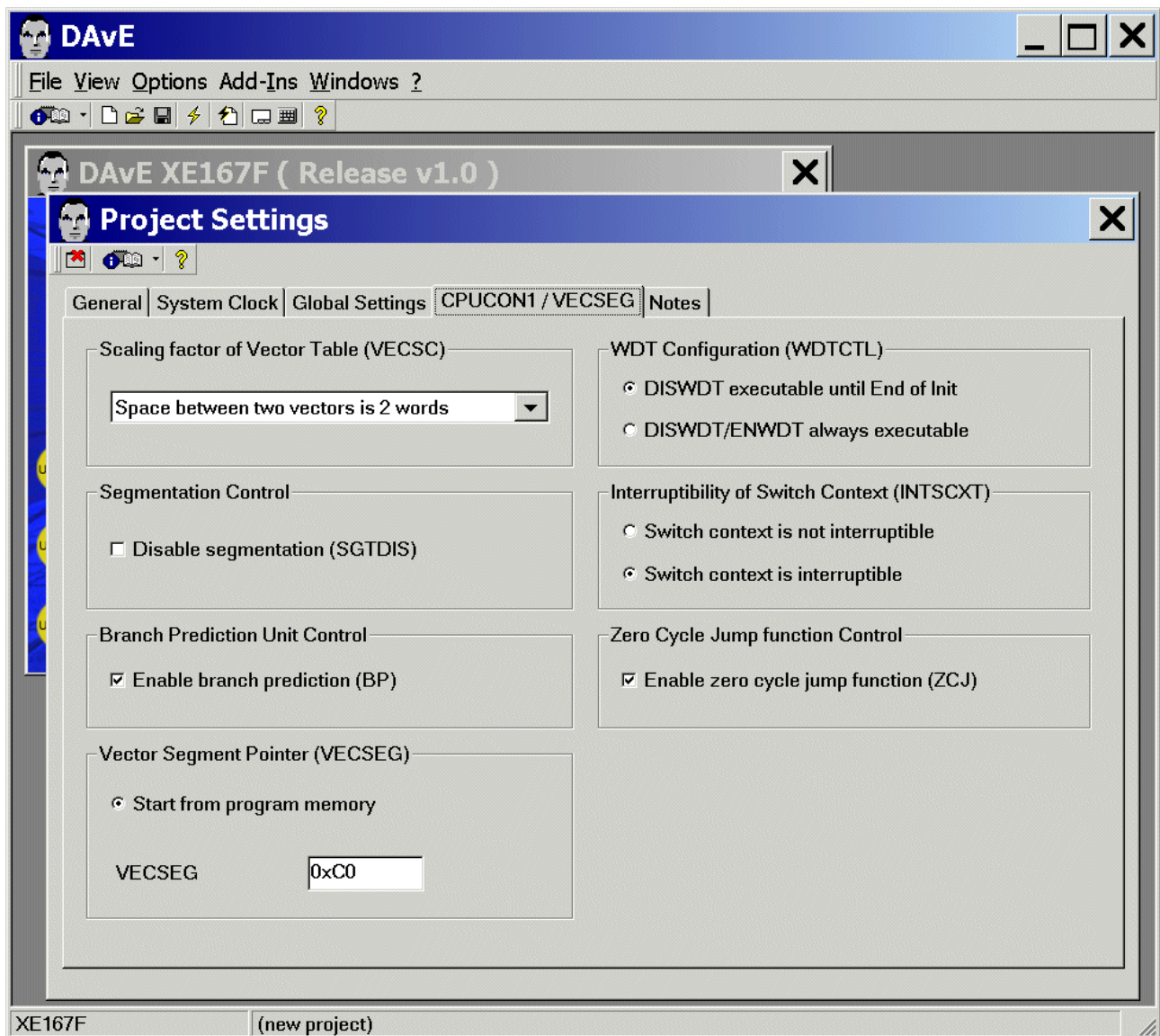
The system clock f_{SYS} is generated out of four selectable clocks:

- PLL clock f_{PLL}
- Wake-Up clock f_{WU}
- The Direct Clock f_{OSC} , from pin XTAL1
- Input DIRIN as Direct Clock Input f_{DIR}

Global Settings: (do nothing, do not change configuration)



CPUCON1/VECSEG: (do nothing)




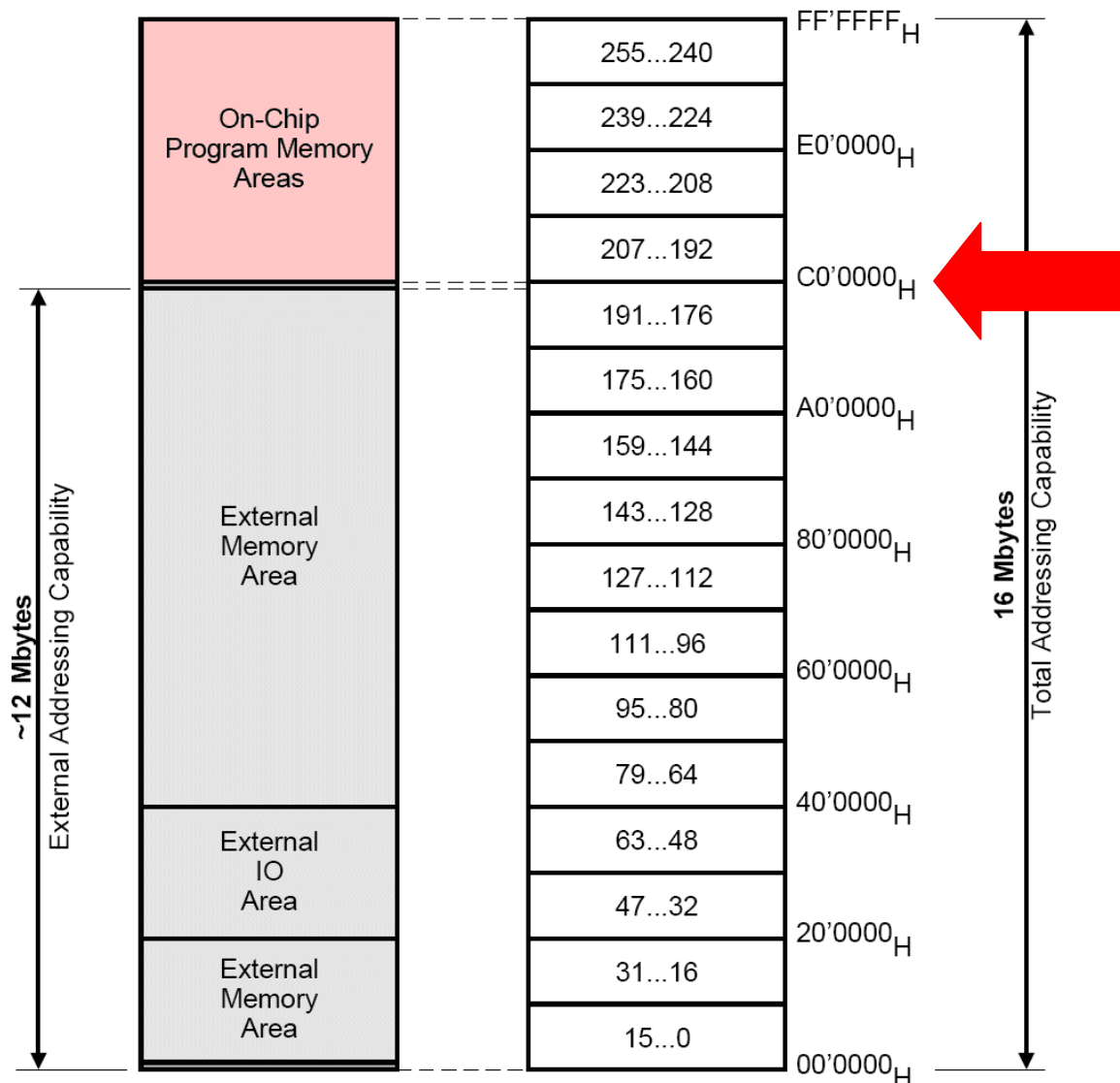


Additional information: Start from program memory (Source: User's Manual):

Vector Segment Pointer (VECSEG)


☒ Start from program memory

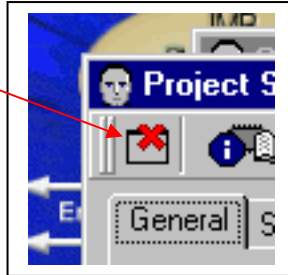
VECSEG 



Total Address Space
16 Mbytes, Segments 255...0

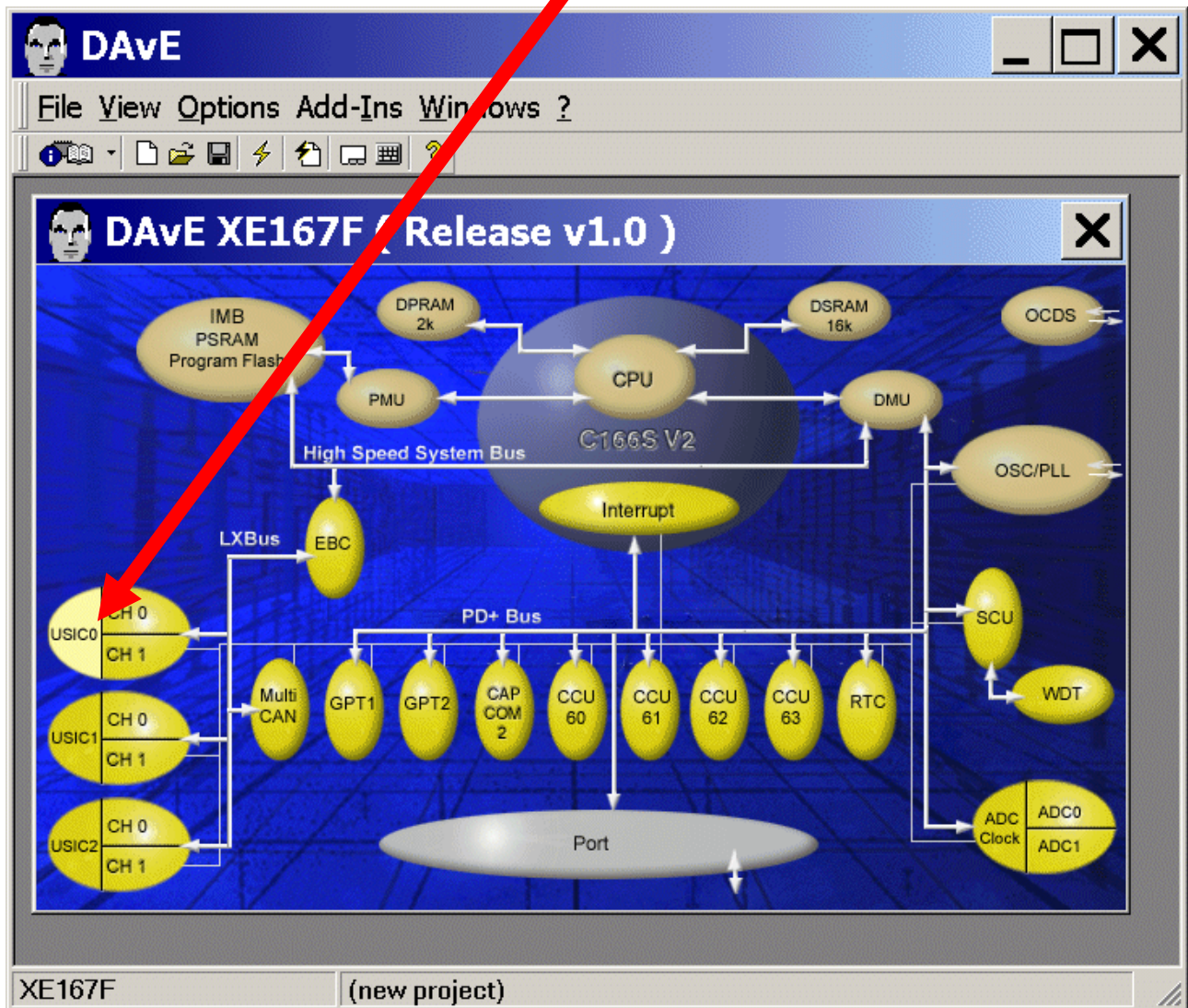
Notes: If you wish, you can insert your comments here.

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

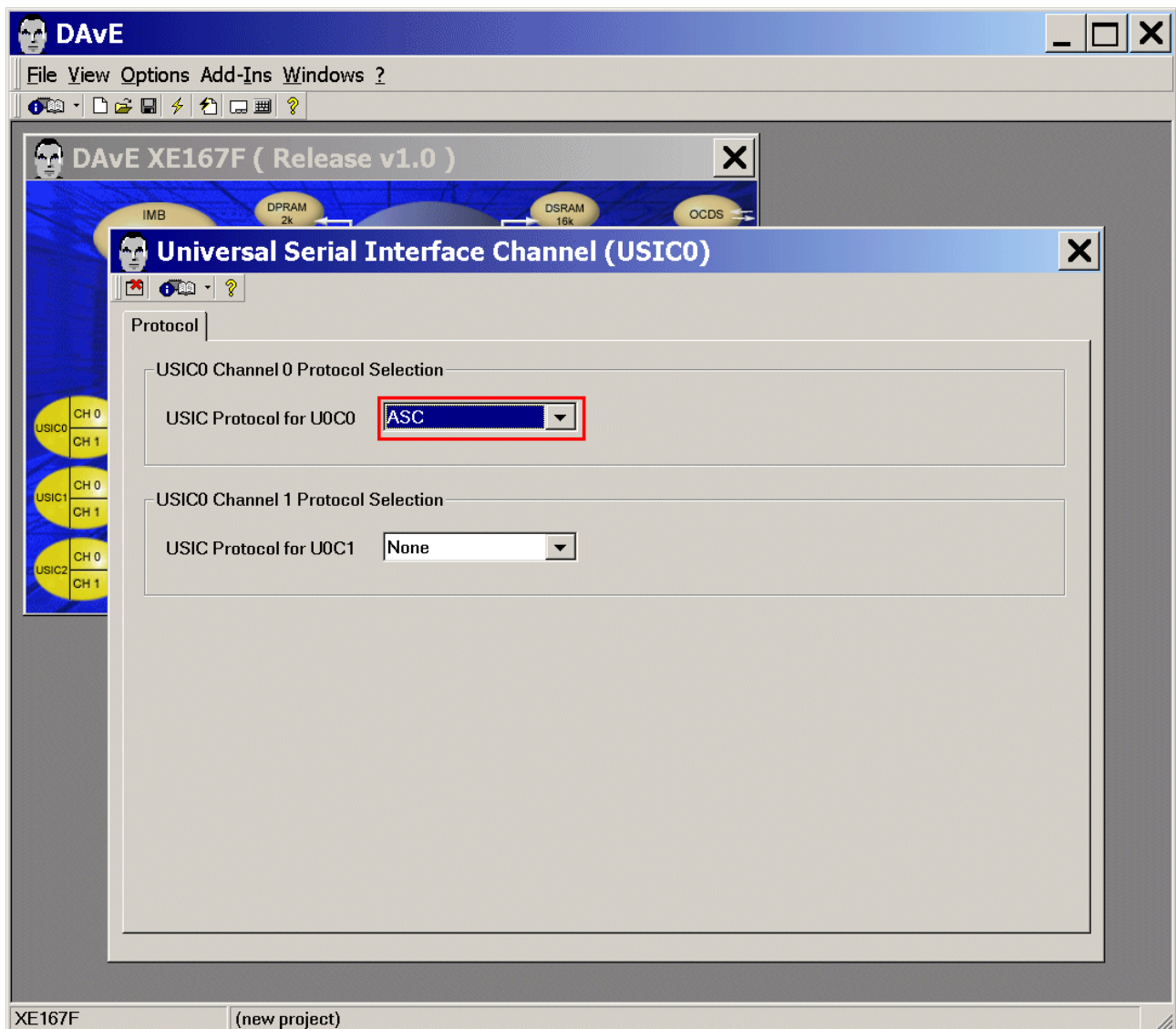



Configuration of the serial interface: "ASC0" / USIC0_CH0 / U0C0:

The configuration window/dialog can be opened by clicking the specific block/module (USIC0).



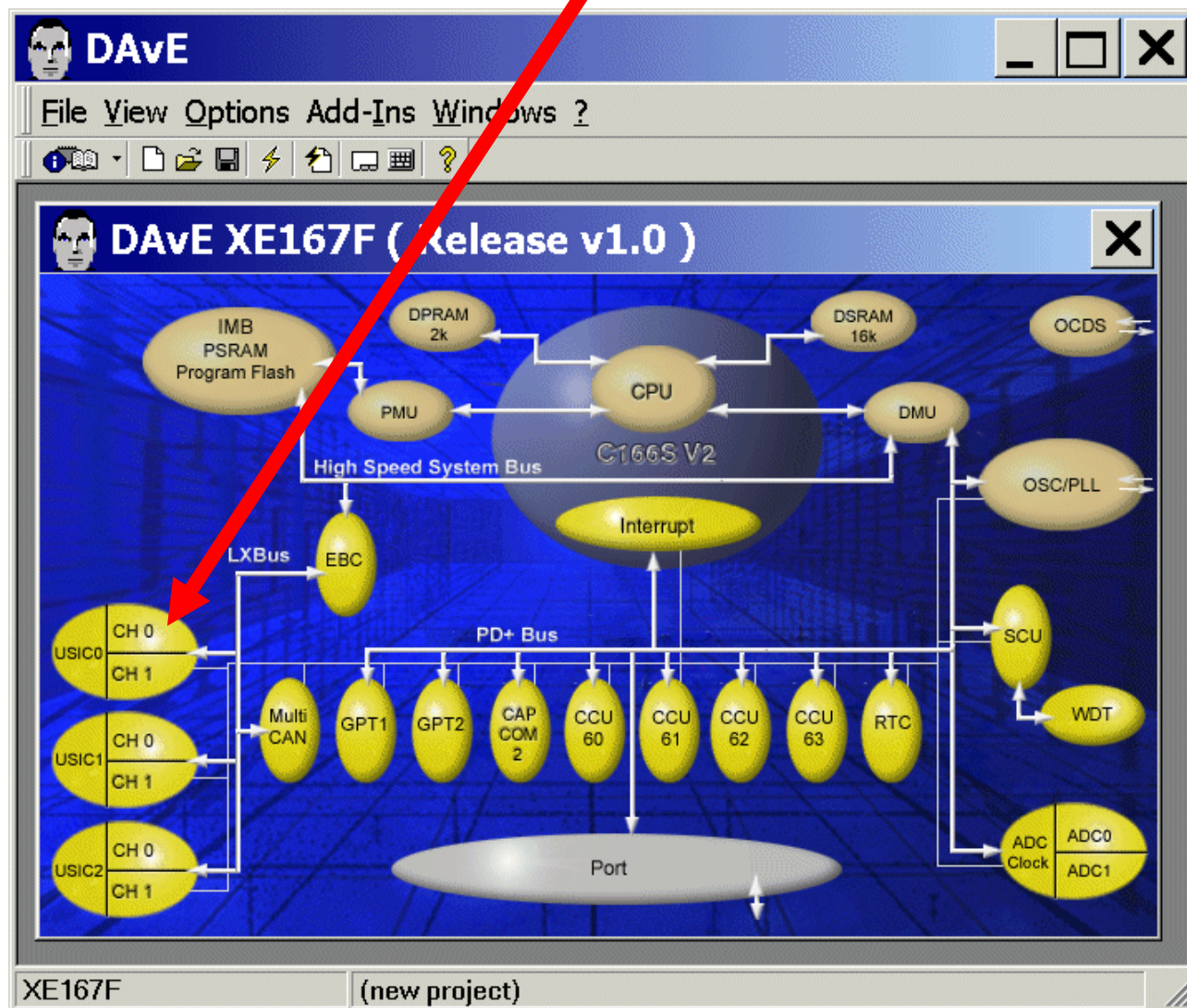
Protocol: USIC0 Channel 0 Protocol Selection: USIC Protocol for U0C0: select ASC



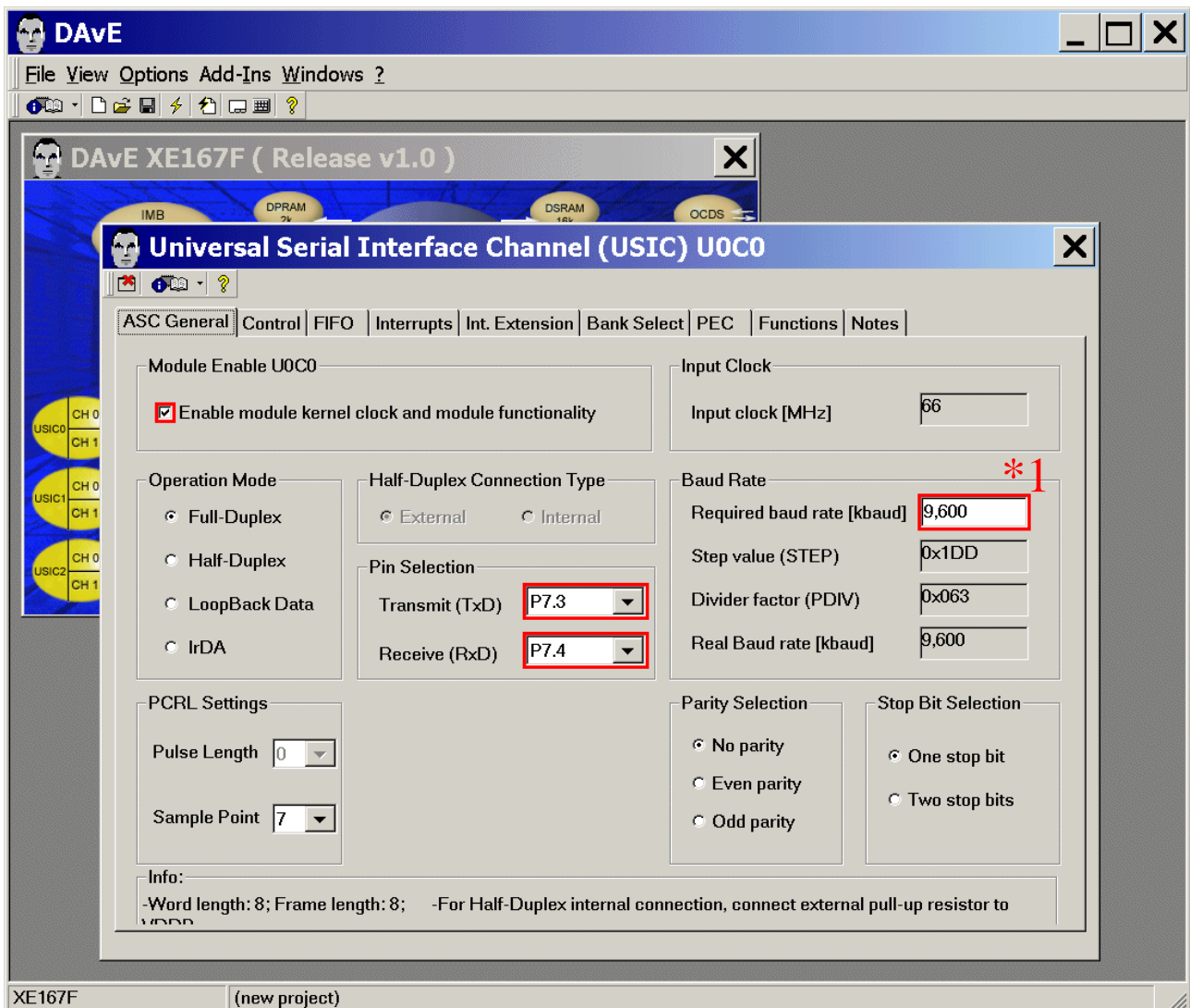
Exit and Save this dialog now by clicking  the close button.

Configuration of the serial interface: USIC0_CH0 / U0C0:

The configuration window/dialog can be opened by clicking the specific block/module (CH 0).



ASC General: Module Enable U0C0: **click** ☒ Enable module kernel clock and module functionality
 ASC General: Pin Selection: Transmit (TxD): **select** P7.3
 ASC General: Pin Selection: Receive (RxD): **select** P7.4
 ASC General: Baud Rate: Required baud rate [kbaud]: **insert** 9,600 <ENTER>



Note (*1):
 Validate each alphanumeric entry by pressing <ENTER>.

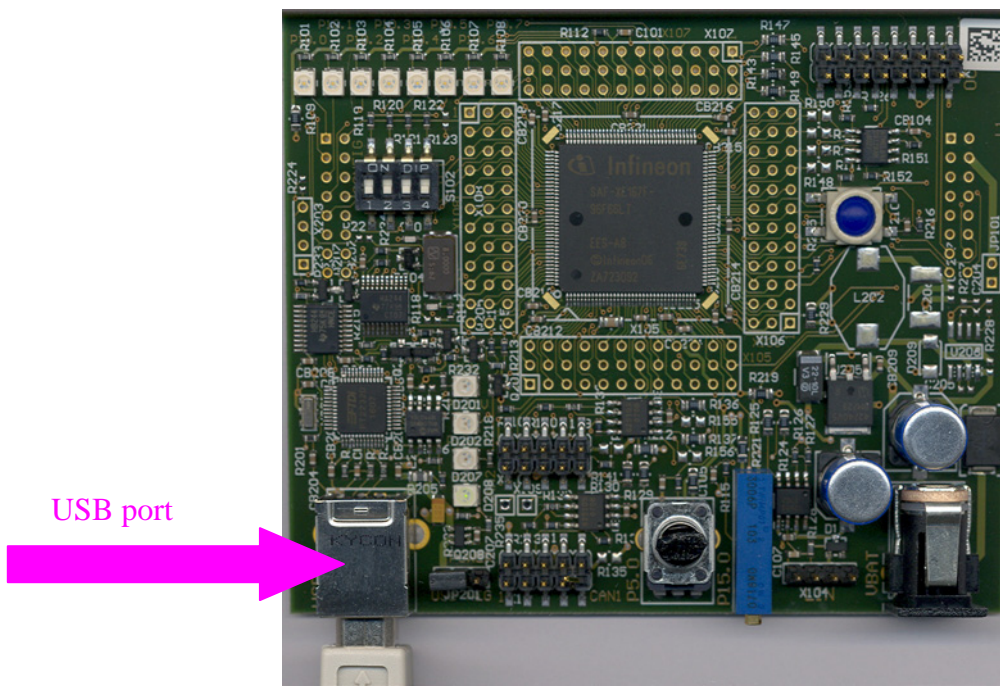




Additional information: RS232 serial interface:

Note:

The RS232 serial interface (USIC_0_Channel_0 pins P7.3 and P7.4) is available via the [USB port](#) which converts the TTL-UART-signals to USB-signals (using a virtual COM port of the second USB channel of the FTDI FT2232 Dual USB to UART/JTAG interface).





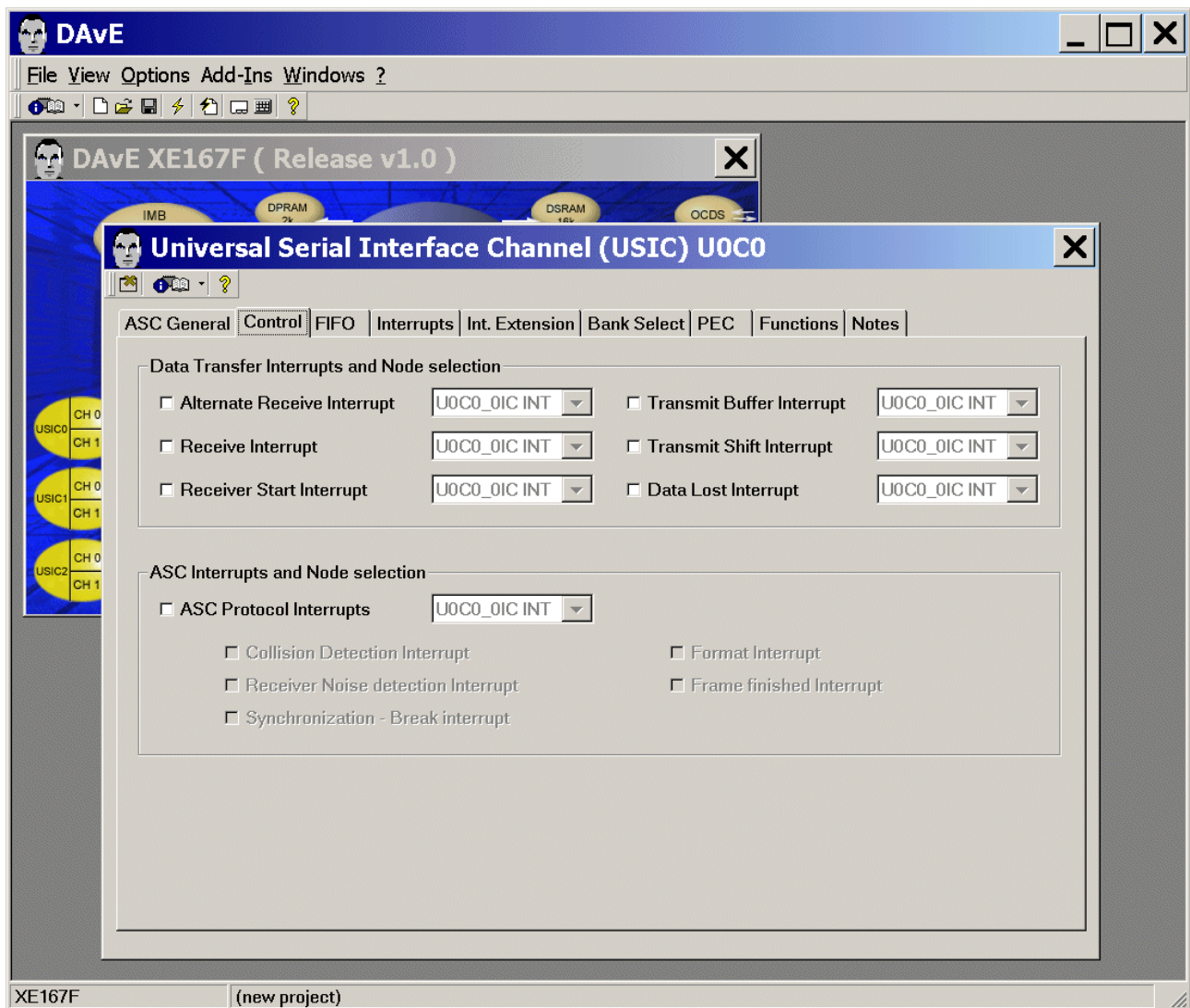
Additional information: Standard UART Pins (Source: User's Manual):

Table 10-10 Configuration Data for Bootstrap Loader Modes

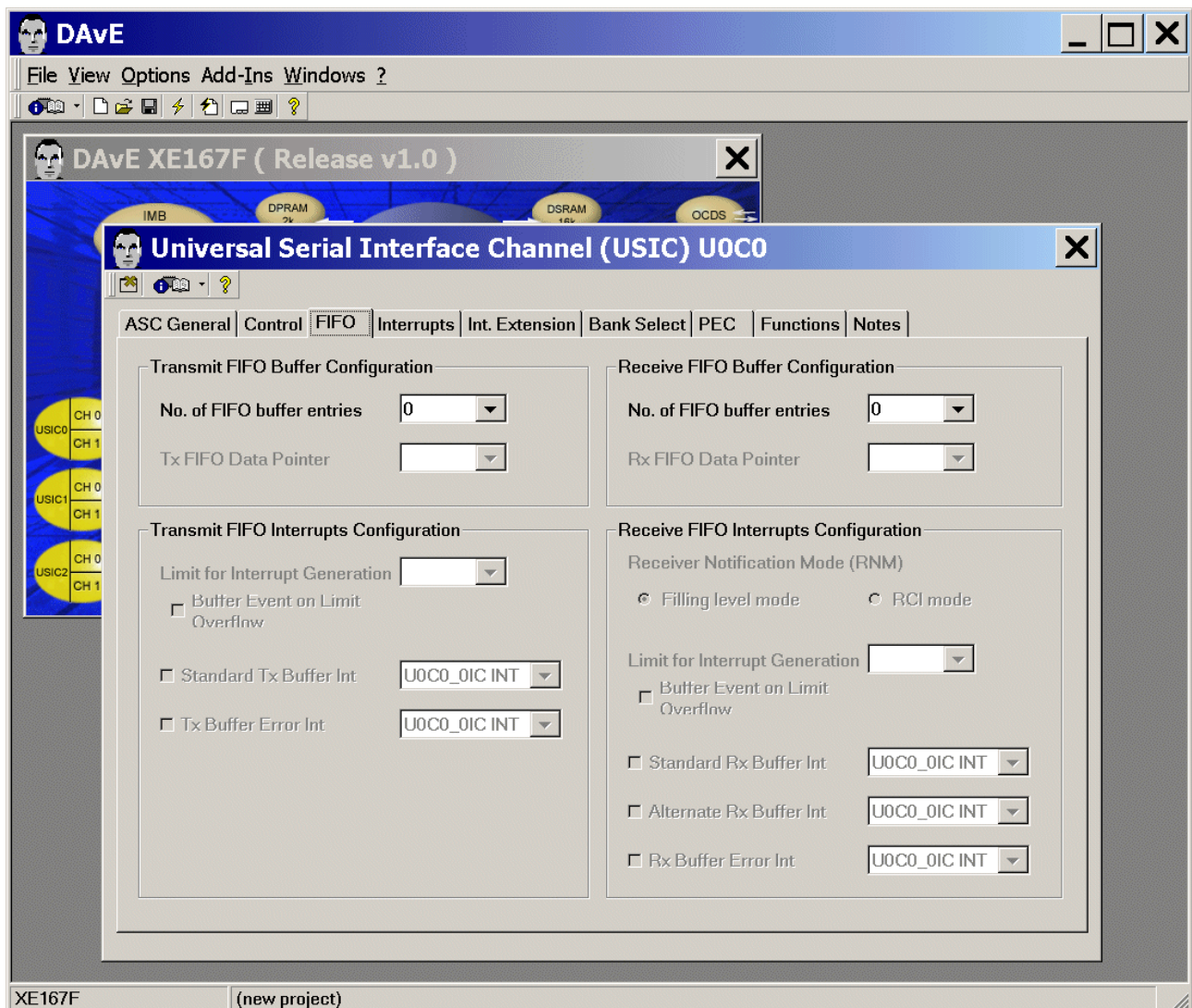
Bootstrap Loader Mode	Configuration on P10.3-0 ¹⁾	Receive Line from Host	Transmit Line to Host	Transferred Data
Standard UART	x110 _B	RxD = P7.4	TxD = P7.3	32 Bytes
Sync. Serial	1001 _B	MRST = P2.4	MTSR = P2.3 SCLK = P2.5 SLS = P2.6	n Bytes; 1 ... 65,280
MultiCAN	x101 _B	RxDC0 = P2.6	TxDC0 = P2.5	8 × n Bytes

1) x means that the level on the corresponding pin is irrelevant.

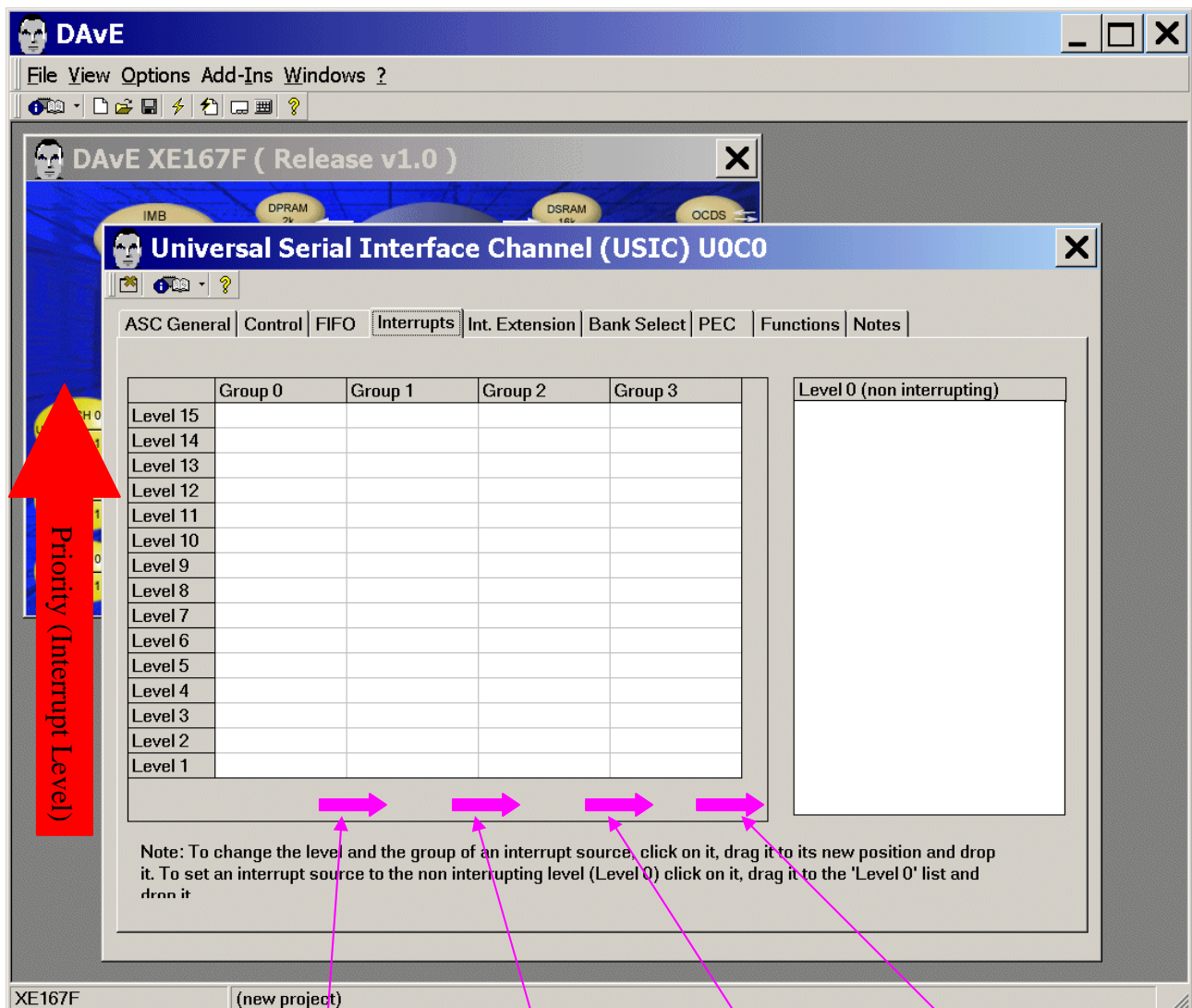
Control: (do nothing)



FIFO: (do nothing)



Interrupts: (do nothing)



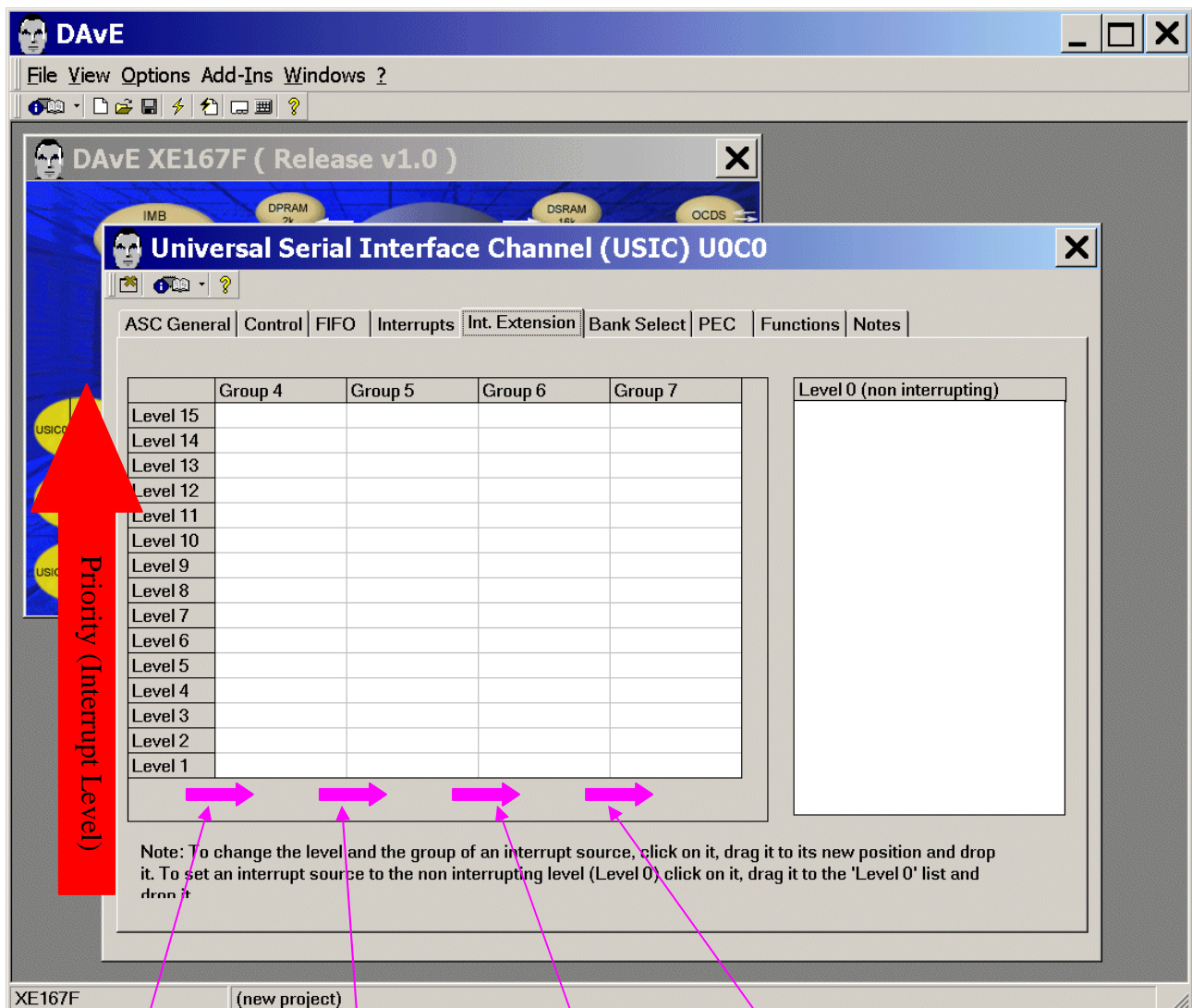
Group Priority 0 → Group Priority 1 → Group Priority 2 → Group Priority 3 →



Note:

For the serial communication with a terminal program (e.g. Docklight, www.docklight.de) running on your host computer the myprintf function is used. The myprintf function uses Software-Polling-Mode therefore we do not need to configure any interrupts.

Int. Extension: (do nothing)



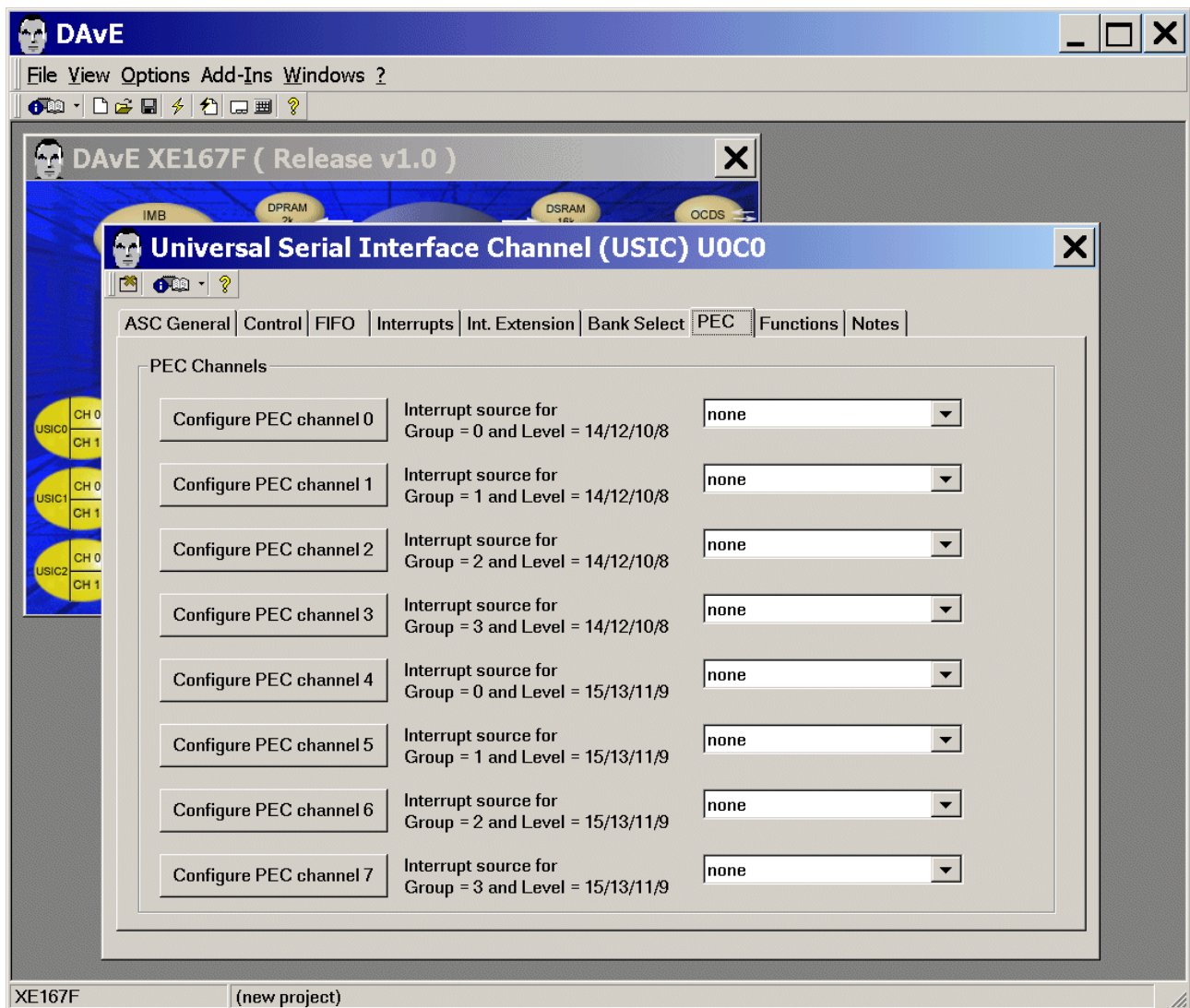
→ Group Priority 4 → Group Priority 5 → Group Priority 6 → GroupPriority 7

The screenshot shows the DAVE XE167F (Release v1.0) application window. The main menu bar includes File, View, Options, Add-Ins, Windows, and ?. Below the menu bar is a toolbar with icons for file operations and help. The central workspace displays the 'Universal Serial Interface Channel (USIC) U0C0' configuration window. This window has a sidebar on the left with a tree view showing components like IMB, DPRAM, DSRAM, OCDS, and USIC channels. The main area of the USIC window contains tabs for ASC General, Control, FIFO, Interrupts, Int. Extension, Bank Select, PEC, Functions, and Notes. The 'Bank Select' tab is active, showing a table with columns for Local Reg Bank 1, Local Reg Bank 2, Fast Interrupts, and Global Register Bank. A scroll bar is visible next to the table. At the bottom of the window, there are two notes:

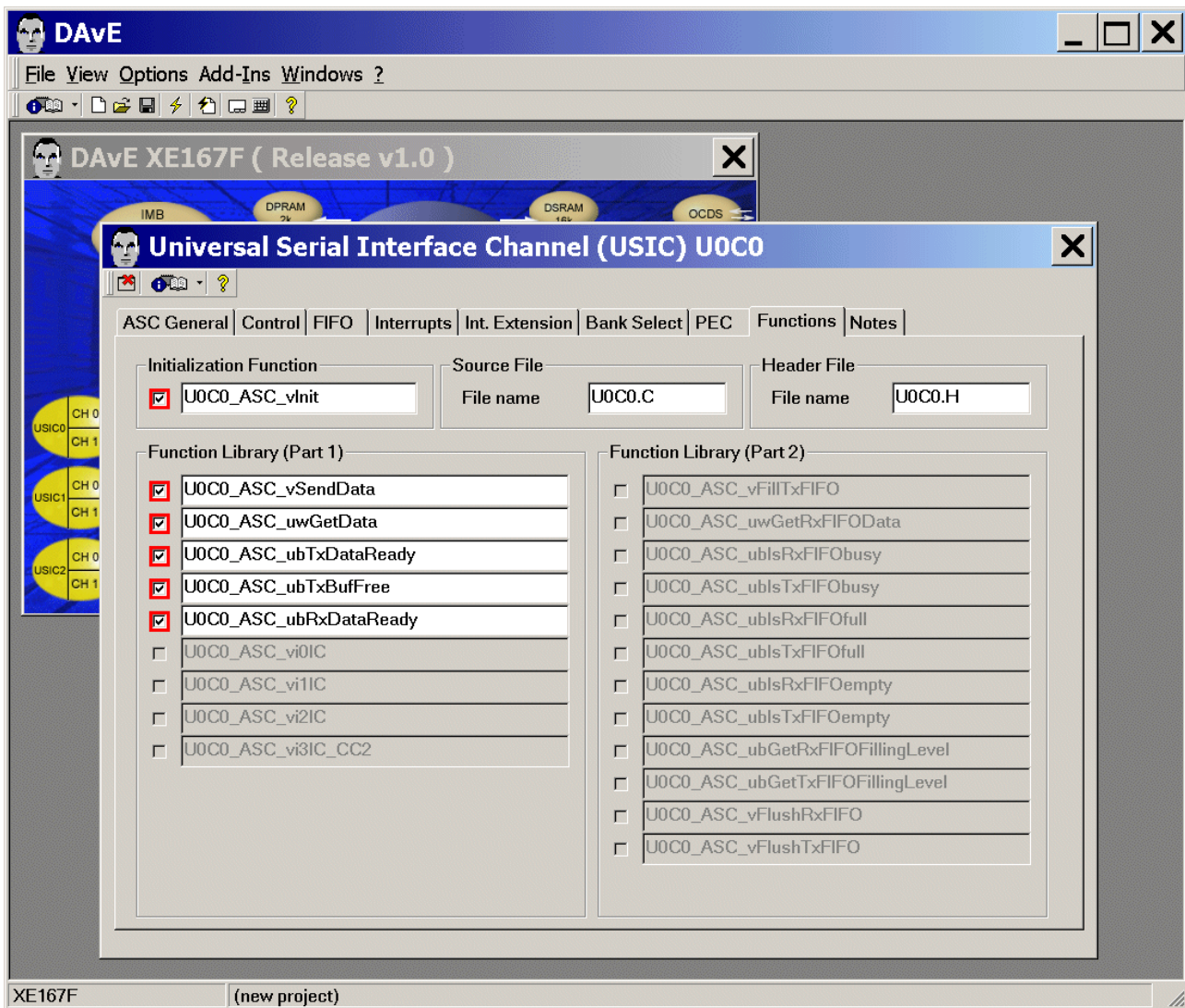
- Note 1: For all int's (with priority > = 12), Local Reg Banks (1 or 2) can be assigned by dragging and dropping to the desired bank.
- Note 2: Only 2 Fast Int's. can be assigned (with priority > = 12) at a time in the bottom 2 cells of Fast Interrupt

The status bar at the bottom of the application shows 'XE167F' and '(new project)'.

PEC: (do nothing)



Functions: Initialization Function: **click** ☒ U0C0_ASC_vInit
 Functions: Function Library (Part 1): **click** ☒ U0C0_ASC_vSendData
 Functions: Function Library (Part 1): **click** ☒ U0C0_ASC_uwGetData
 Functions: Function Library (Part 1): **click** ☒ U0C0_ASC_ubTxDataReady
 Functions: Function Library (Part 1): **click** ☒ U0C0_ASC_ubTxBufFree
 Functions: Function Library (Part 1): **click** ☒ U0C0_ASC_ubRxDataReady

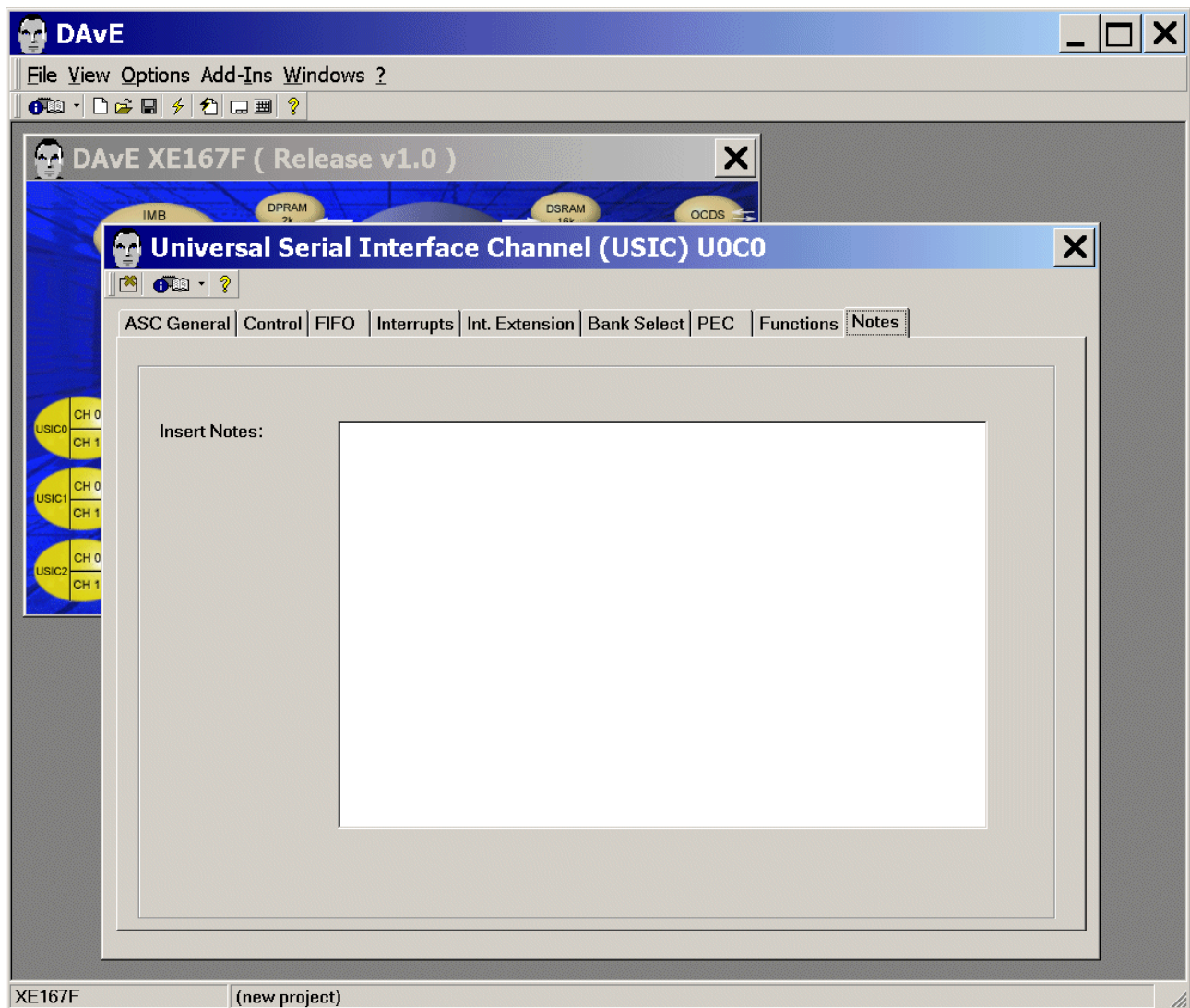


Note:

You can change function names (e.g. U0C0_ASC_vInit) and file names (e.g. U0C0.C, U0C0.H) anytime.




Notes: (do nothing)



Note:

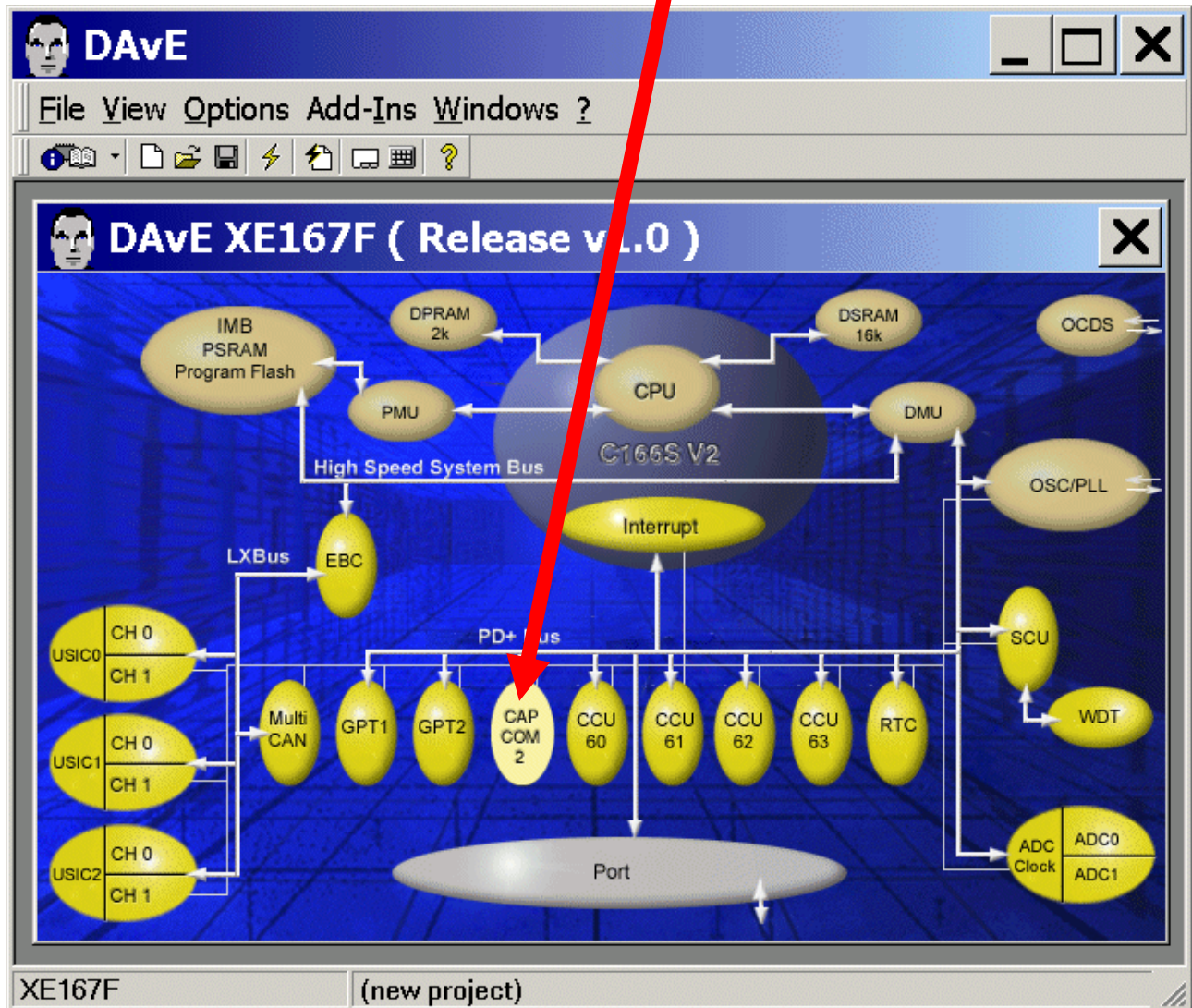
Notes: If you wish, you can insert your comments here.



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

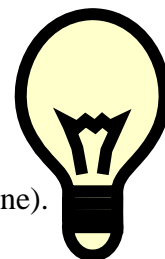
Configure Timer T7 in the CAPCOM 2 module:

The configuration window/dialog can be opened by clicking the specific block/module (CAPCOM2).

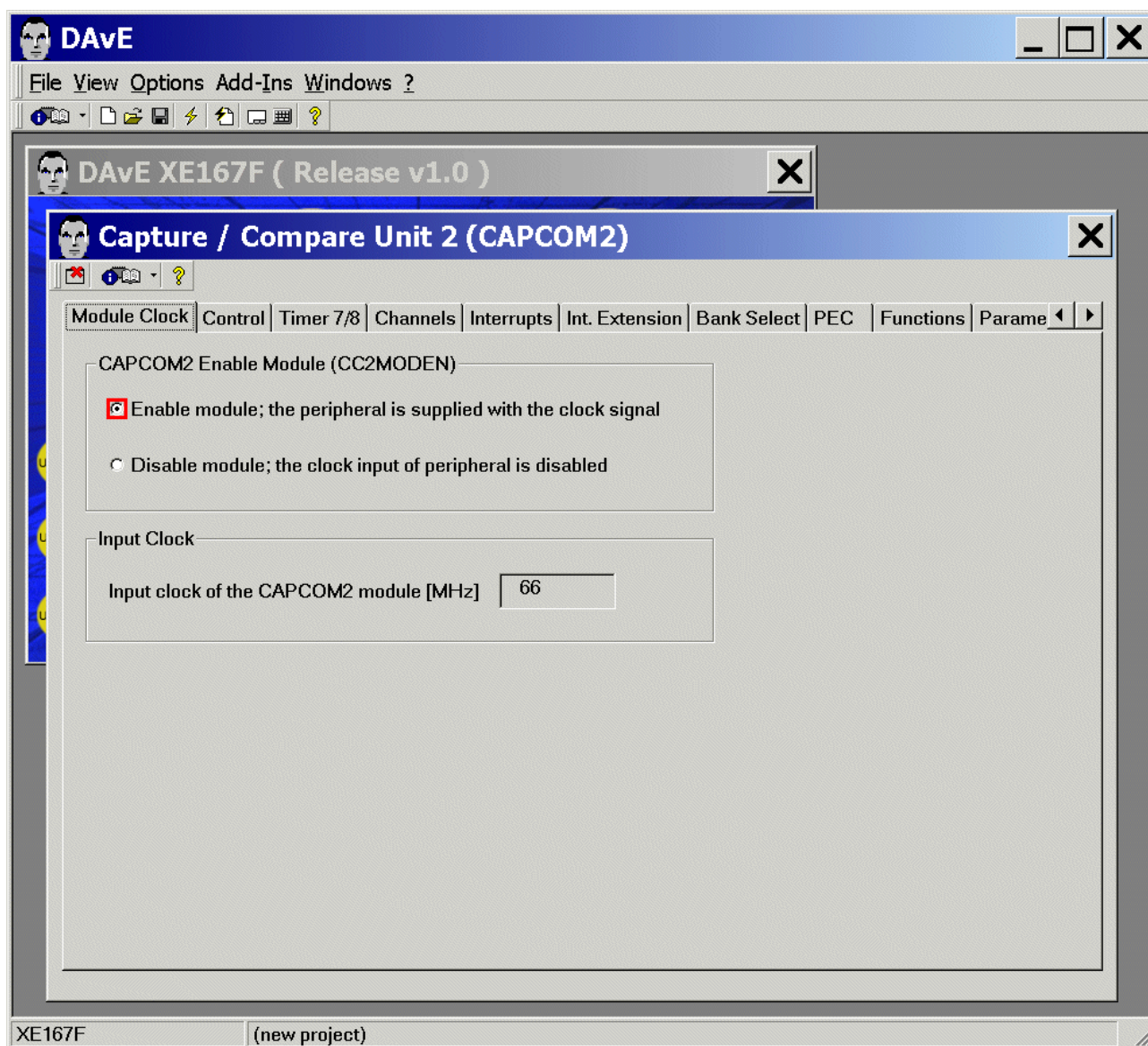


Note:

The LEDs on IO_Port_10_Low will be blinking (if selected in the main menu) with a frequency of about 1 second (done in the Timer_7-Interrupt-Service-Routine). Therefore we have to configure Timer_7.

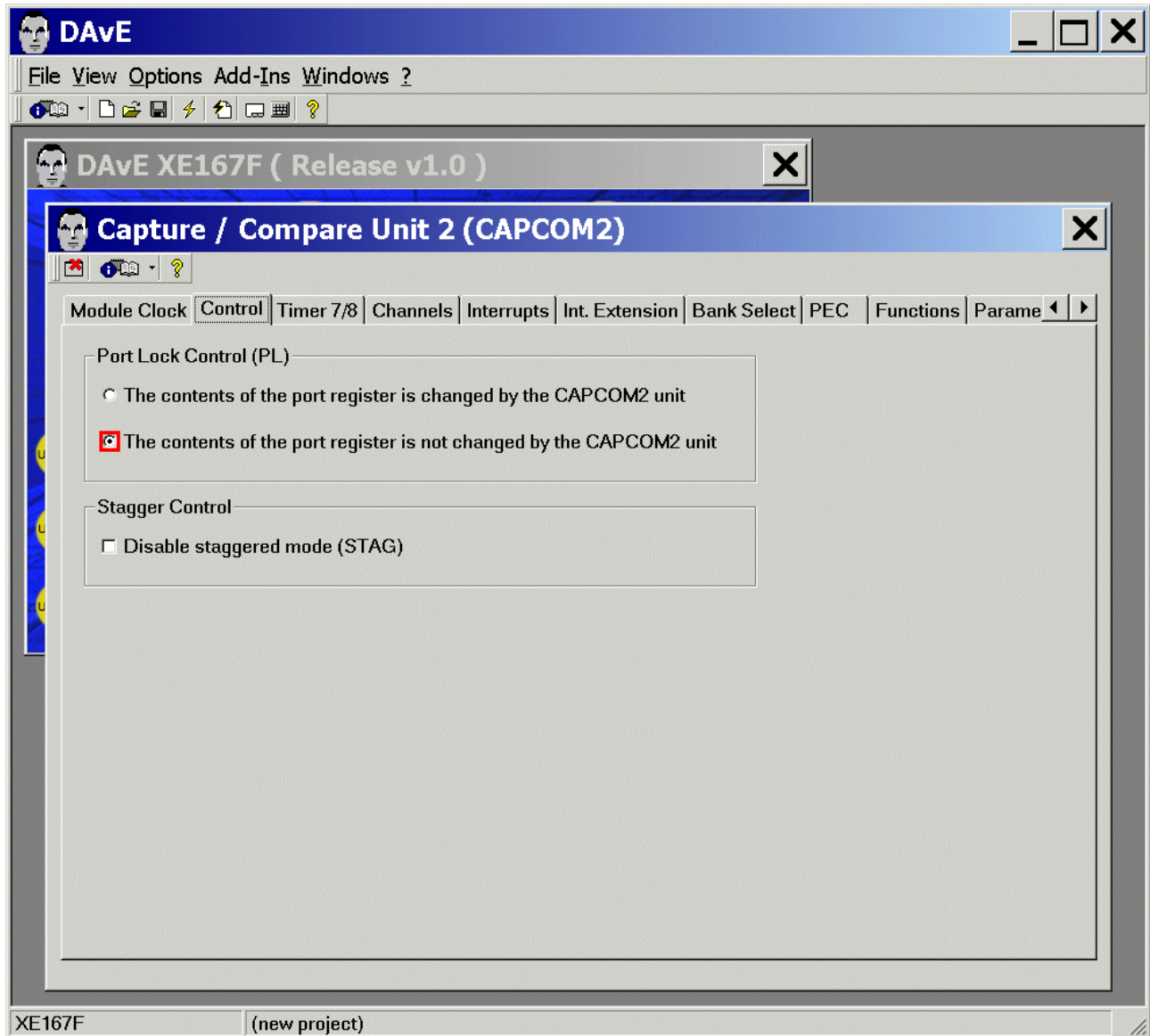


Module Clock: CAPCOM2 Enable Module: **click** ☒ Enable module



Control: Port Lock Control:

click ☒ The contents of the port register is not changed by the CAPCOM2 unit

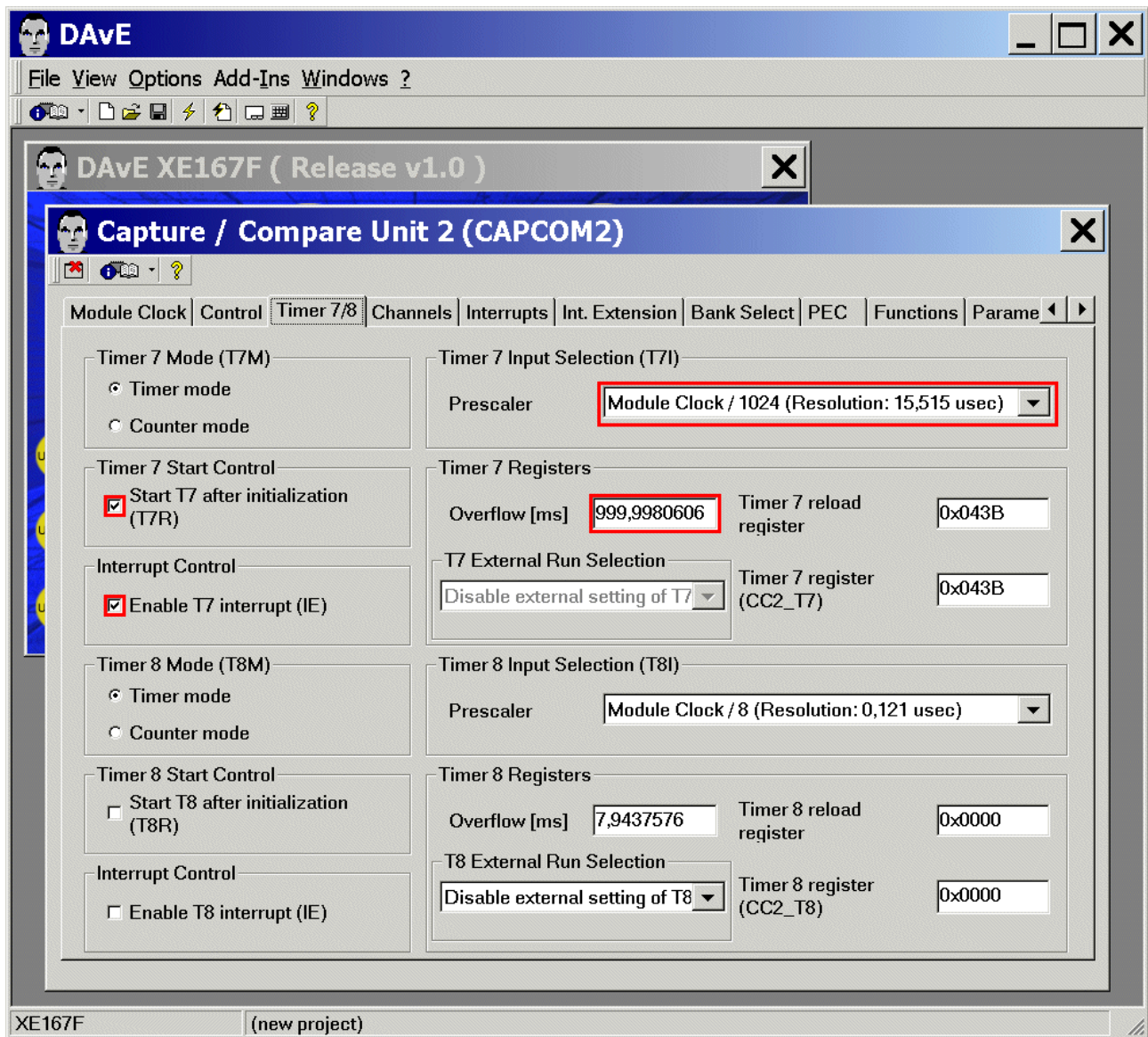


Timer 7/8: Timer 7 Start Control: **click** ✓ Start T7 after initialization (T7R)

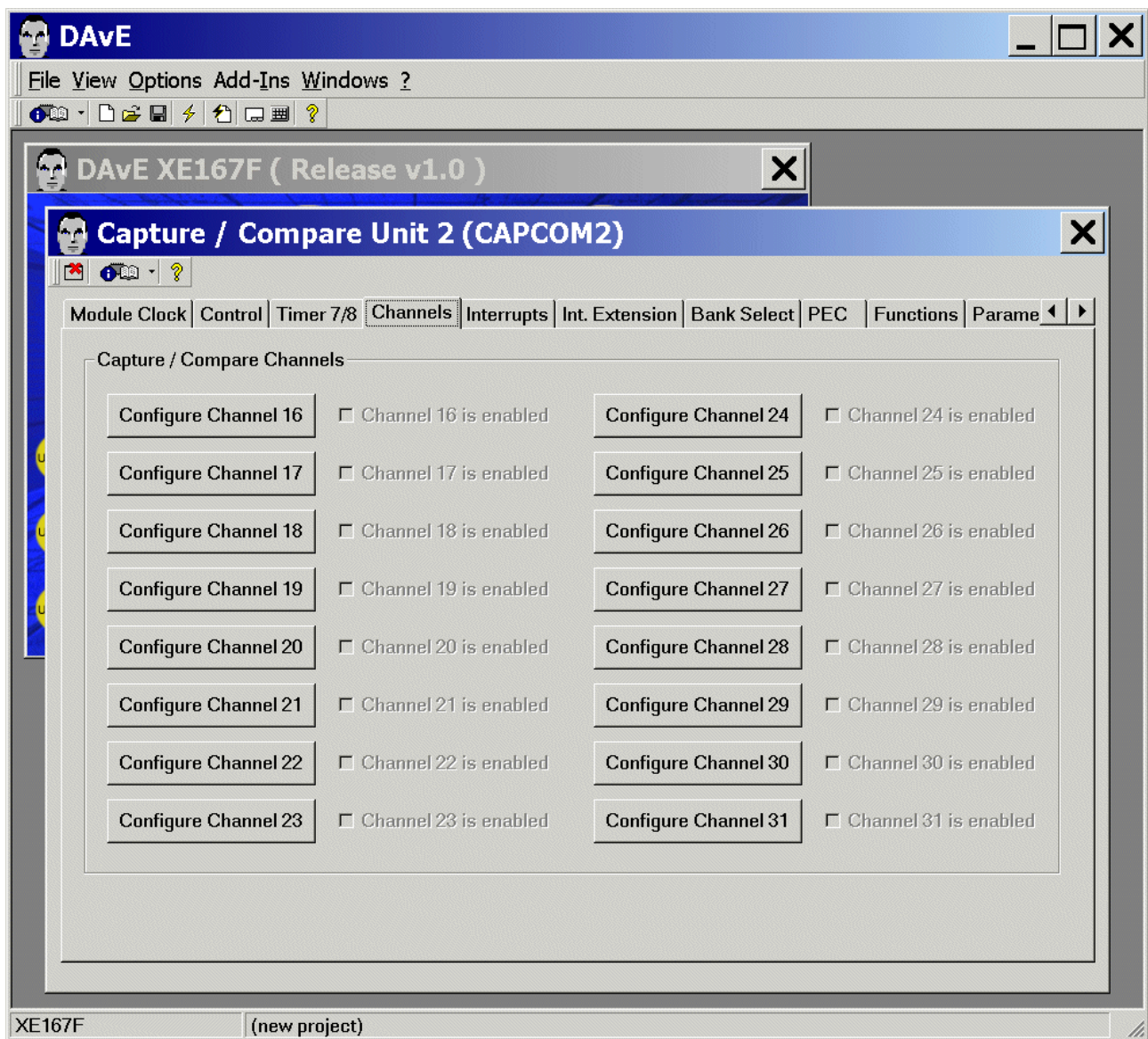
Timer 7/8: Interrupt Control: **click** ✓ Enable T7 interrupt (IE)

Timer 7/8: Timer 7 Input Selection (T7I): Prescaler: **choose** Module Clock/1024

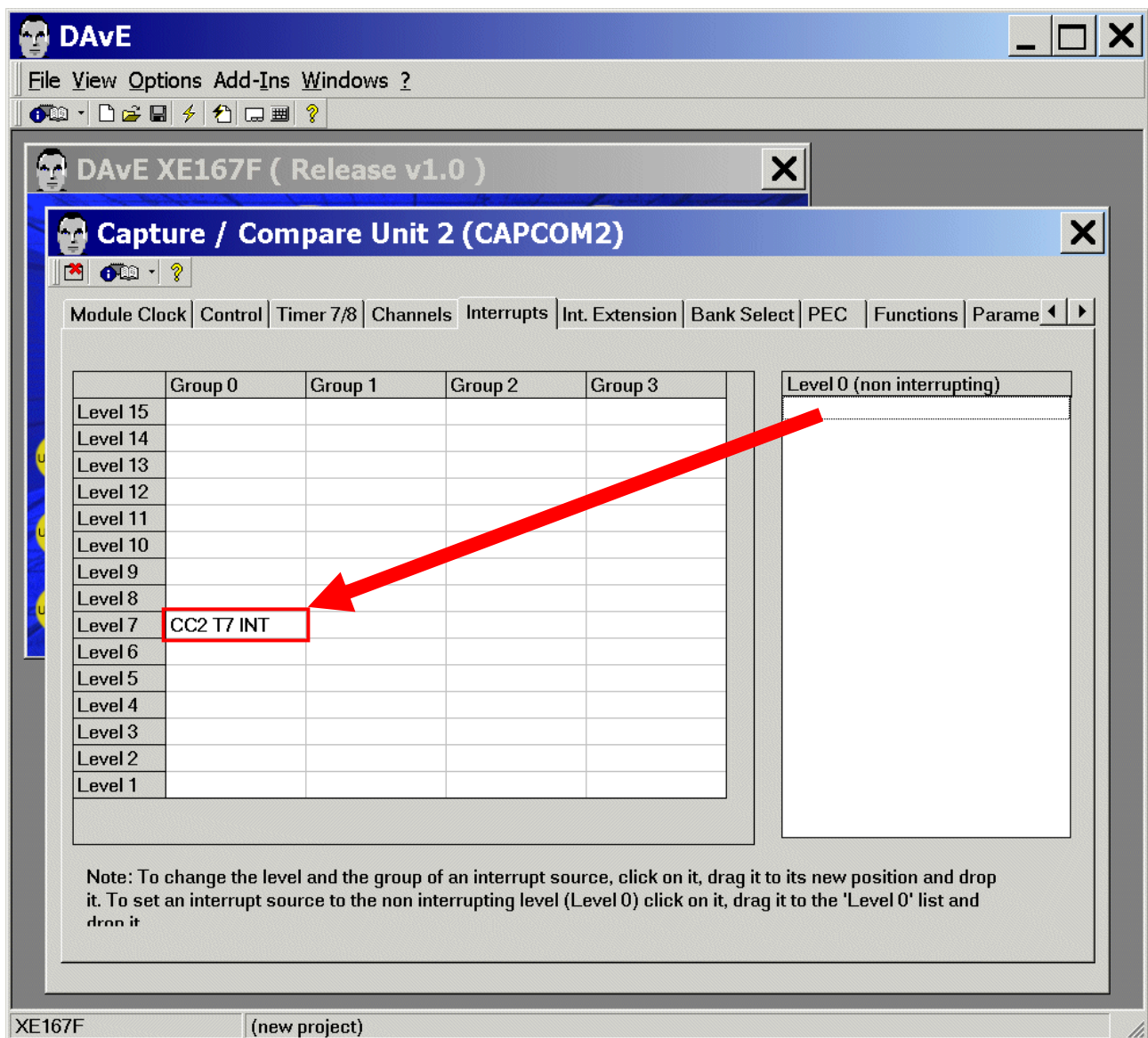
Timer 7/8: Timer 7 Registers: Overflow [s]: **insert** 1 <ENTER>



Channels: (do nothing)



Interrupts: drag and drop the CC2 T7 INT to Interrupt Level 7, Group 0

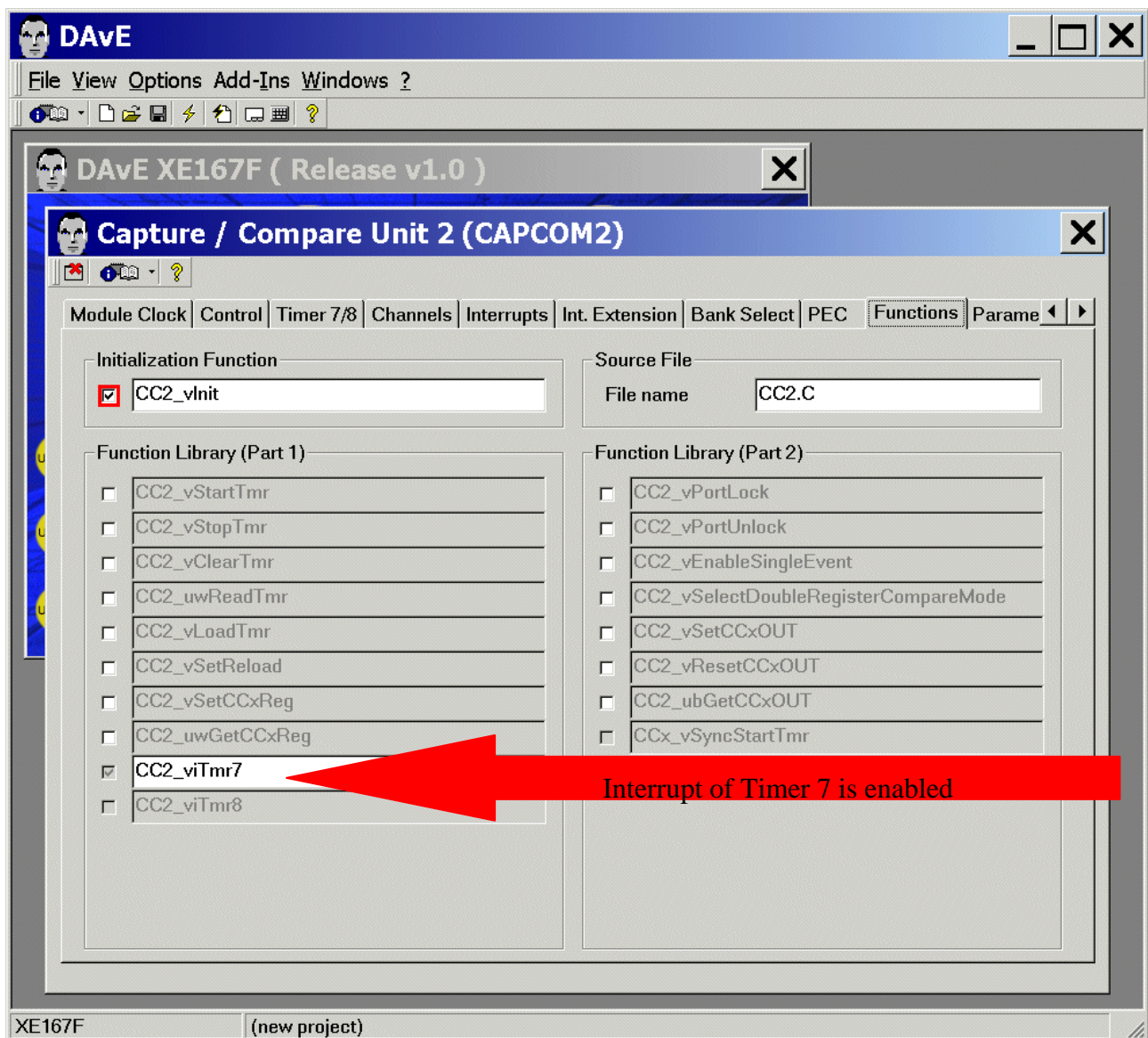


Int. Extension: (do nothing)

Bank Select: (do nothing)


PEC: (do nothing)

Functions: Initialization Function: click ☒ CC2_vInit



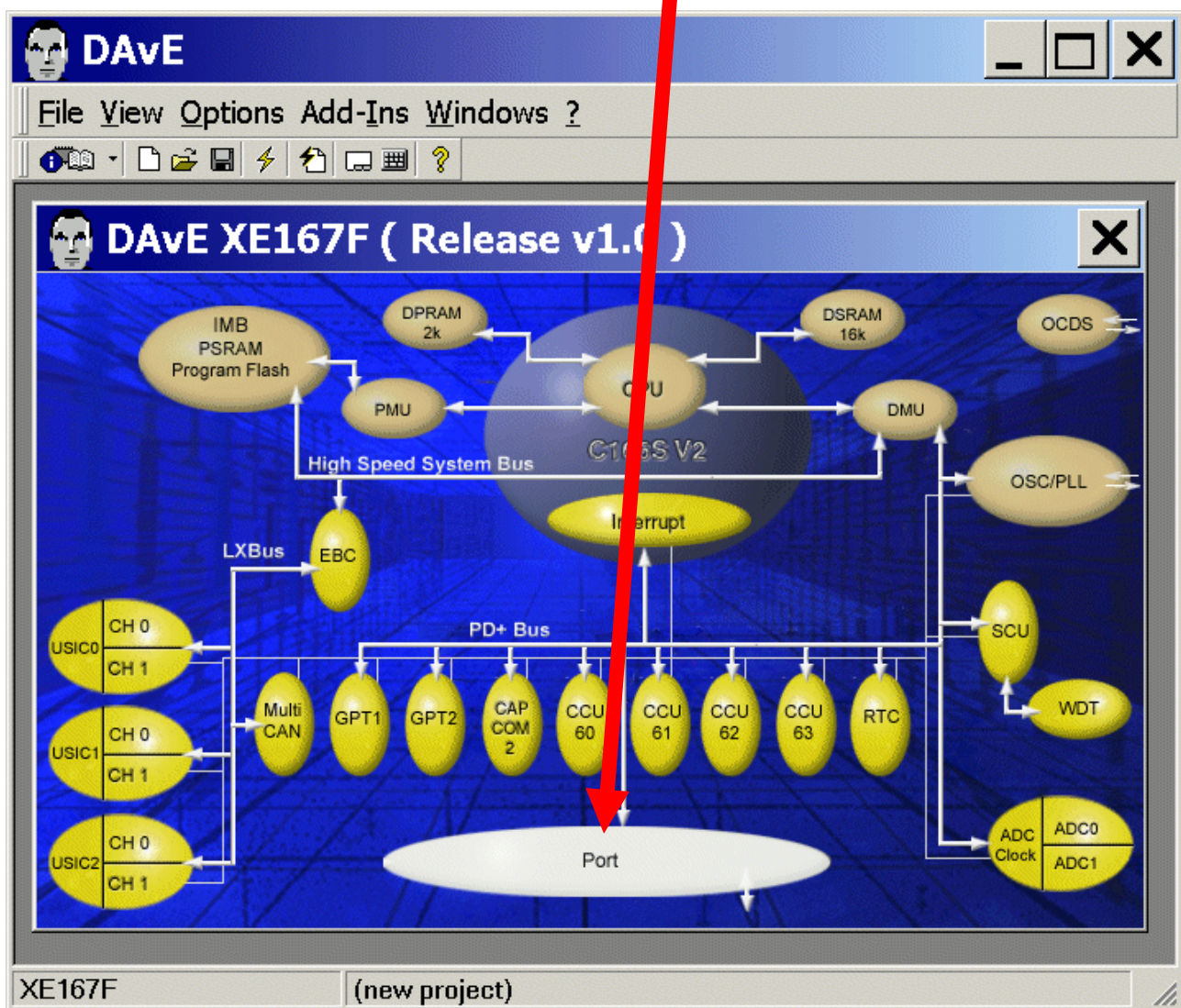
Parameters: (do nothing)

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

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

Configure Port 10 Low (Pin 0, 1, 2, 3, 4, 5, 6 and 7) to Output :

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

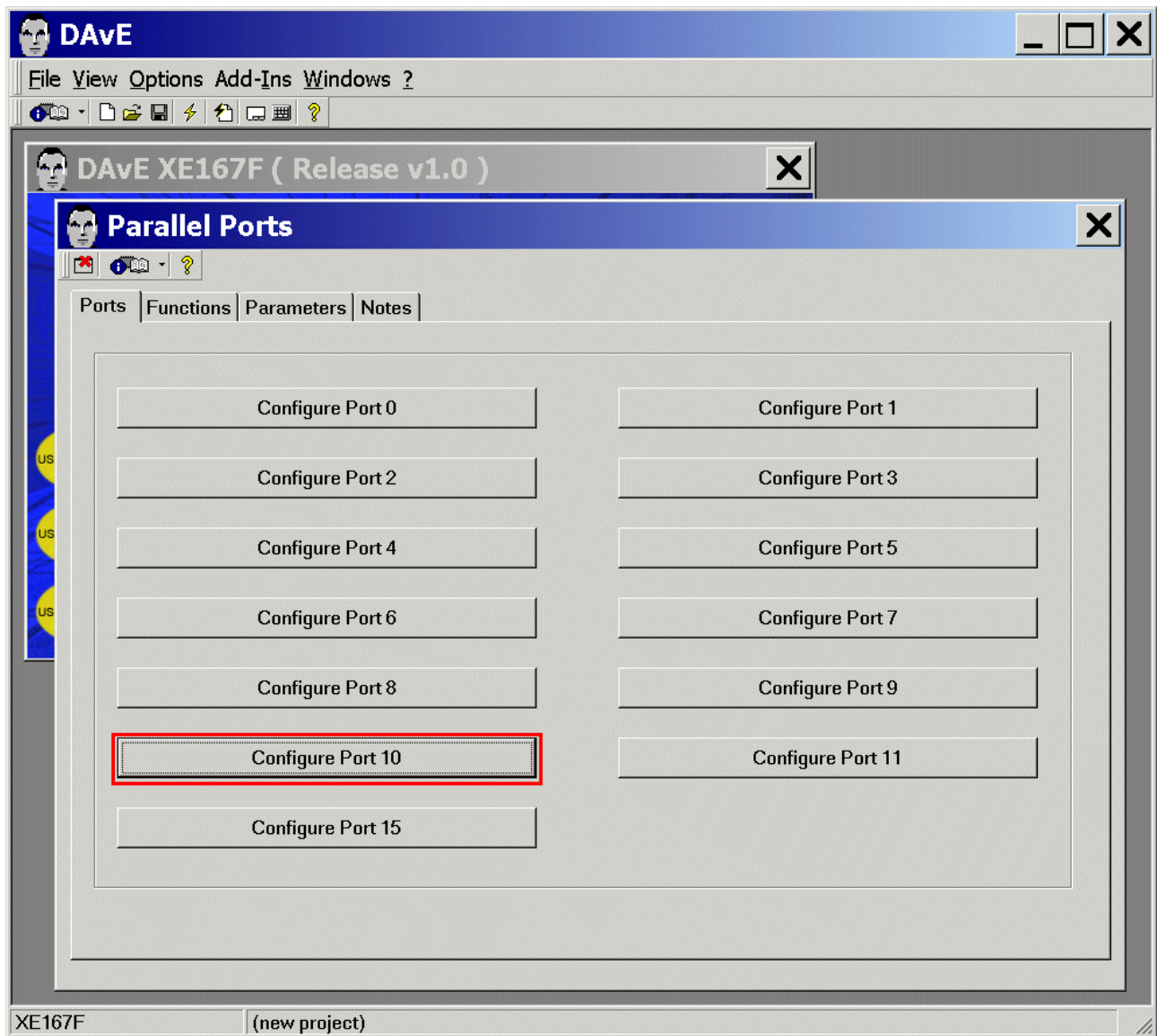


Note:

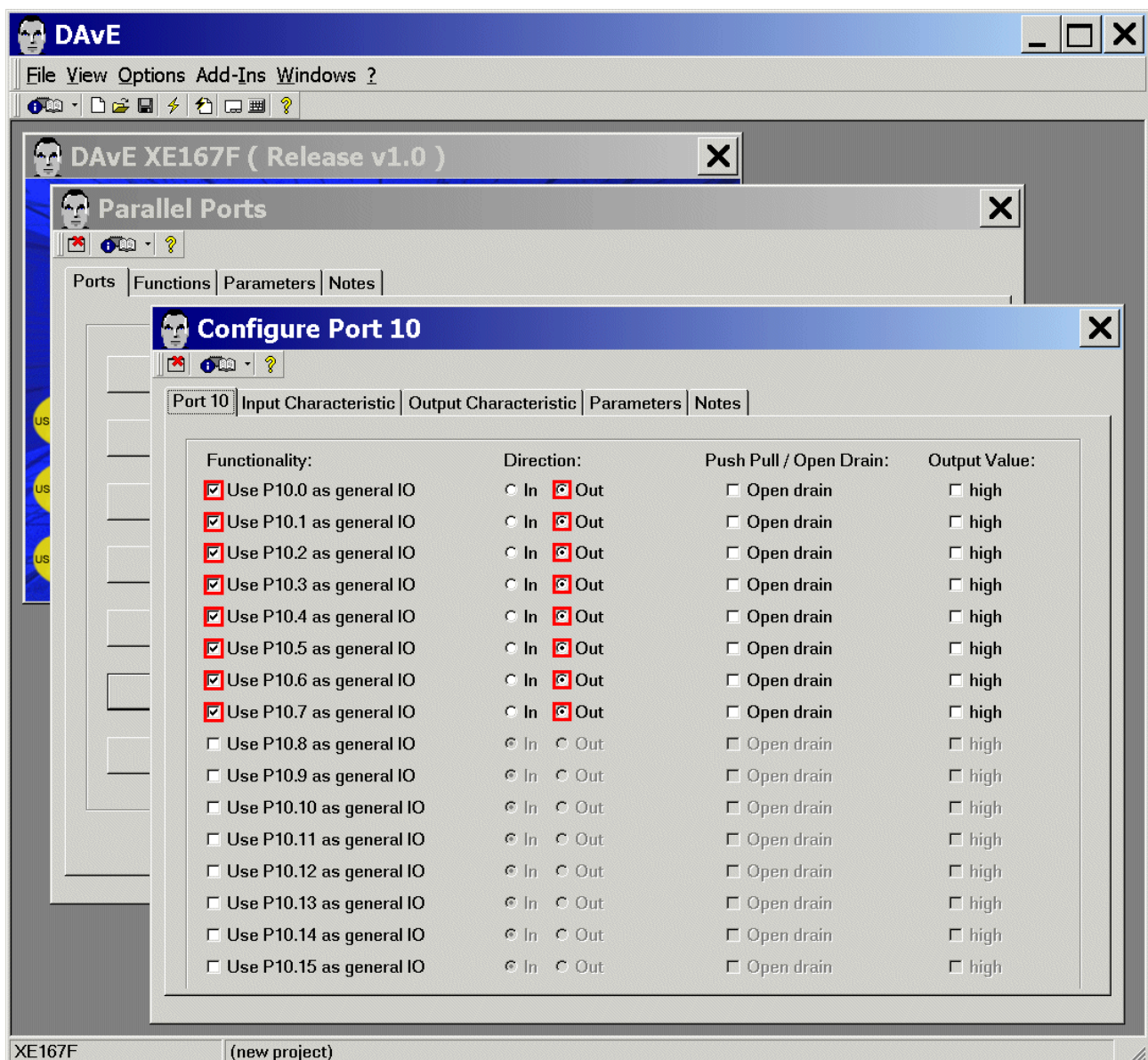
The LEDs are connected to IO_Port_10_Low.



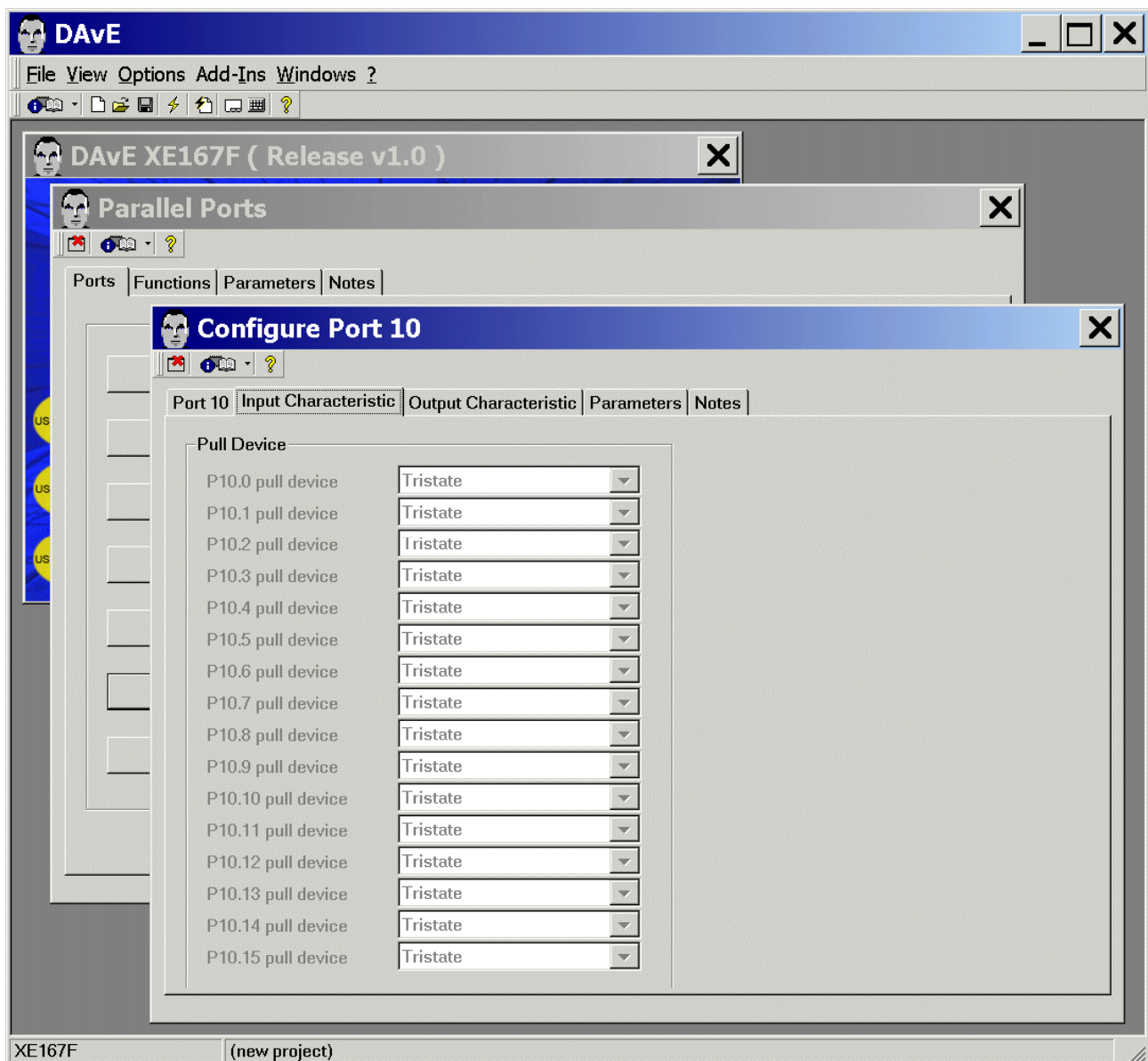
Ports: click "Configure Port 10"



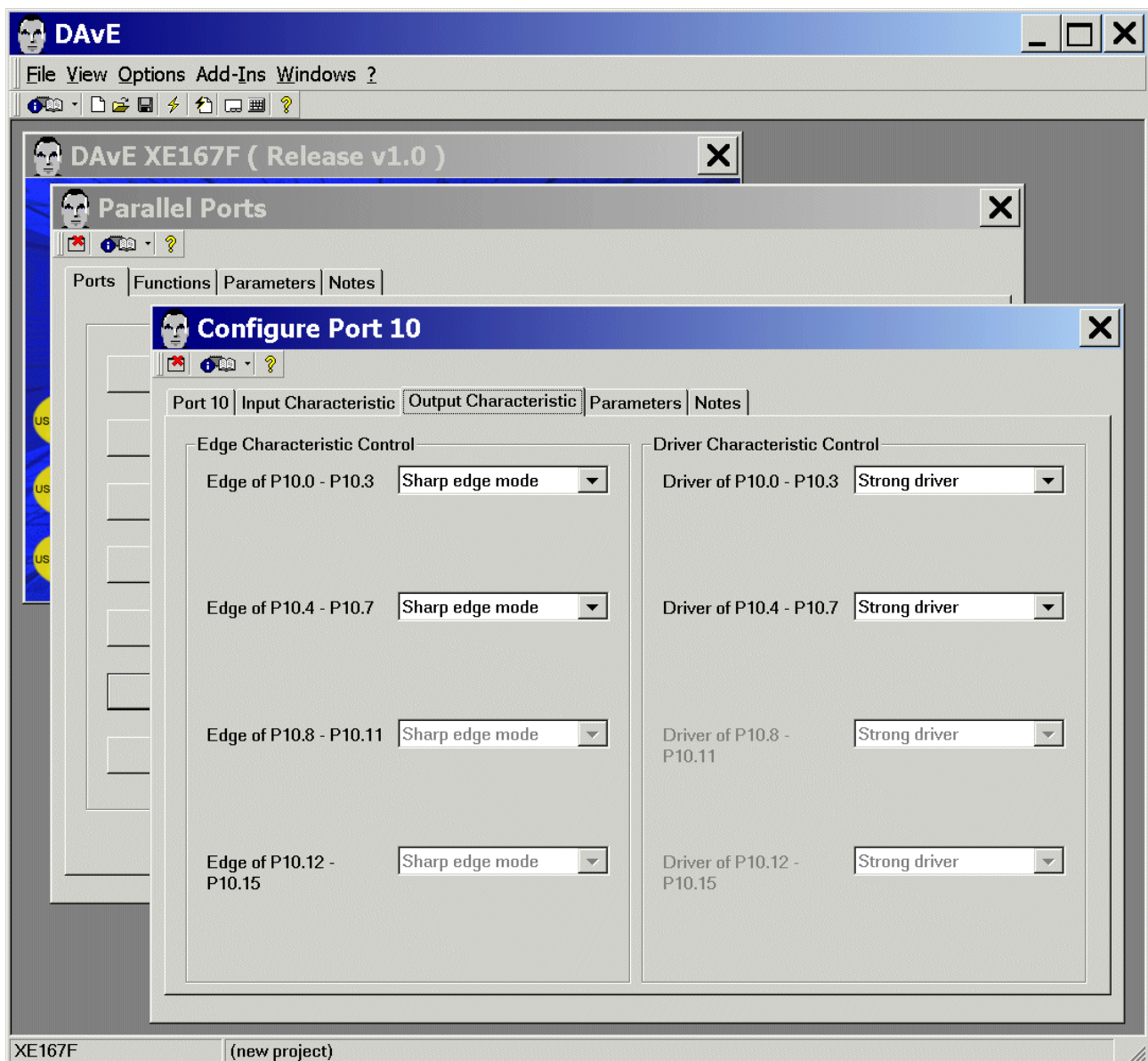
Port 10: Functionality: [click](#) ☒ Use P10.0 as general IO - Direction: [click](#) ☐ In ☒ Out
 Port 10: Functionality: [click](#) ☒ Use P10.1 as general IO - Direction: [click](#) ☐ In ☒ Out
 Port 10: Functionality: [click](#) ☒ Use P10.2 as general IO - Direction: [click](#) ☐ In ☒ Out
 Port 10: Functionality: [click](#) ☒ Use P10.3 as general IO - Direction: [click](#) ☐ In ☒ Out
 Port 10: Functionality: [click](#) ☒ Use P10.4 as general IO - Direction: [click](#) ☐ In ☒ Out
 Port 10: Functionality: [click](#) ☒ Use P10.5 as general IO - Direction: [click](#) ☐ In ☒ Out
 Port 10: Functionality: [click](#) ☒ Use P10.6 as general IO - Direction: [click](#) ☐ In ☒ Out
 Port 10: Functionality: [click](#) ☒ Use P10.7 as general IO - Direction: [click](#) ☐ In ☒ Out



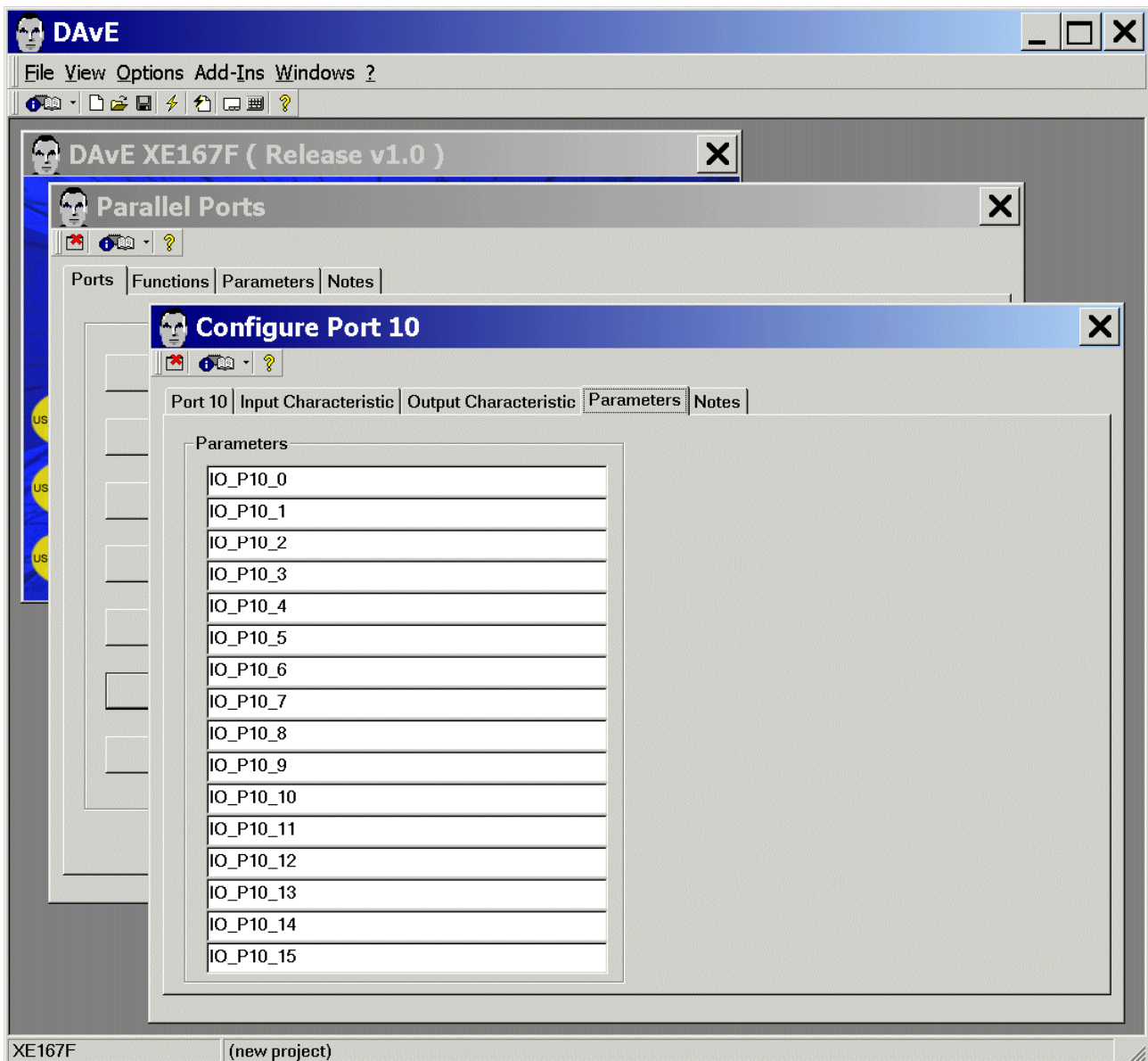
Input Characteristic: (do nothing)




Output Characteristic: (do nothing)



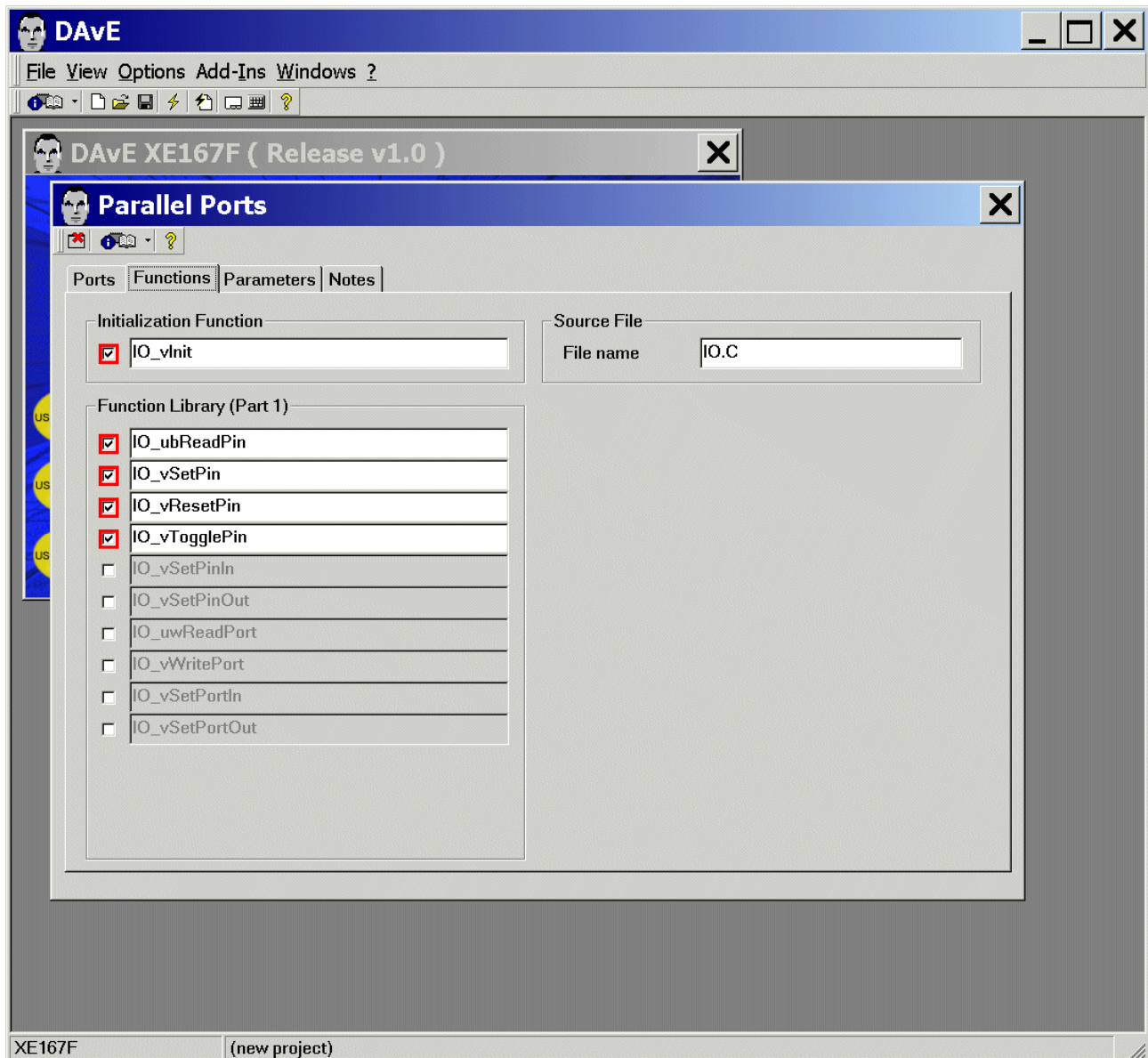
Parameters: (do nothing)



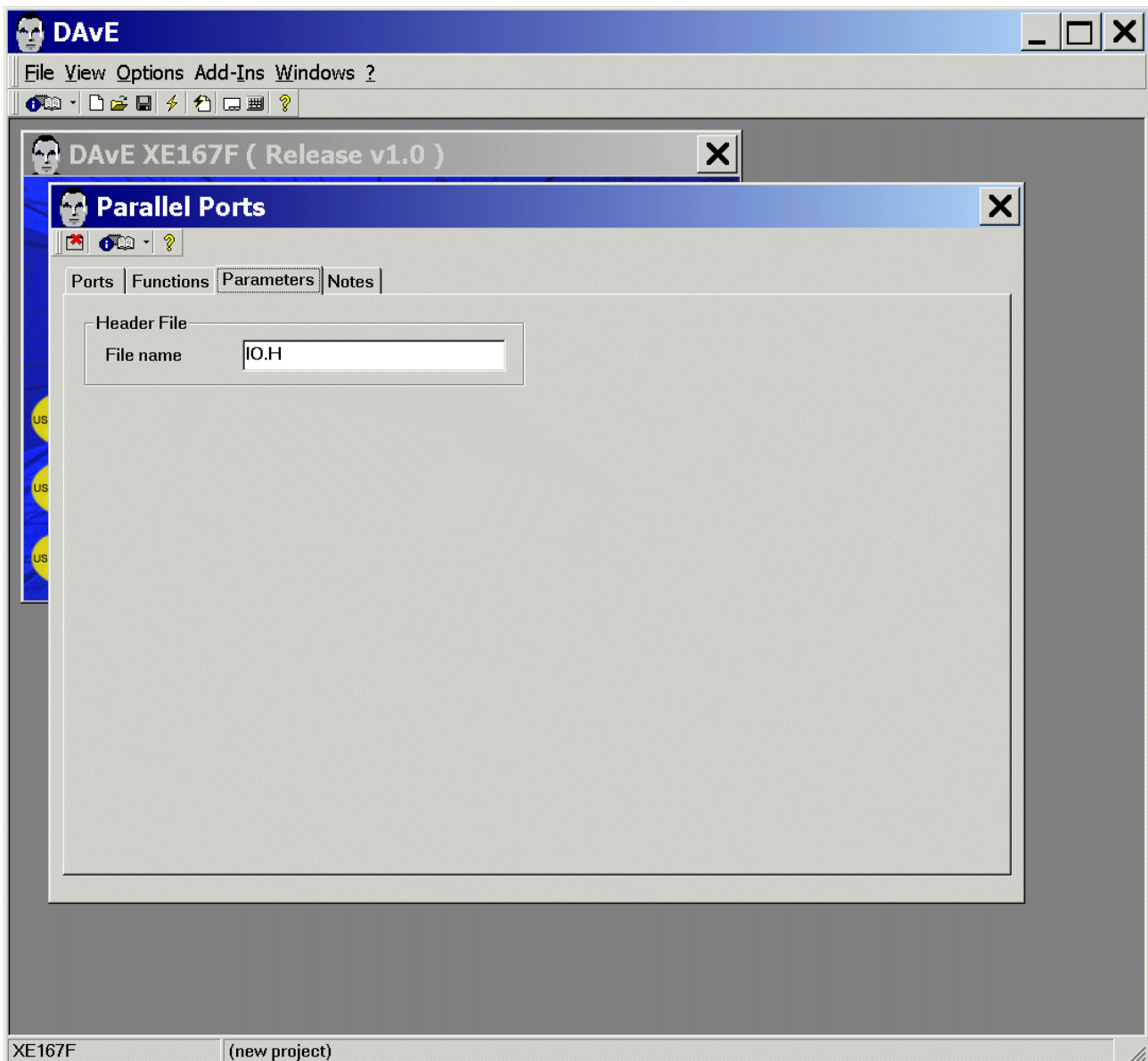
Notes: If you wish, you can insert your comments here.

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


Functions: Initialization Functions: **click** ☒ IO_vInit
 Functions: Function Library (Part 1): **click** ☒ IO_ubReadPin
 Functions: Function Library (Part 1): **click** ☒ IO_vSetPin
 Functions: Function Library (Part 1): **click** ☒ IO_vResetPin
 Functions: Function Library (Part 1): **click** ☒ IO_vTogglePin



Parameters: (do nothing)



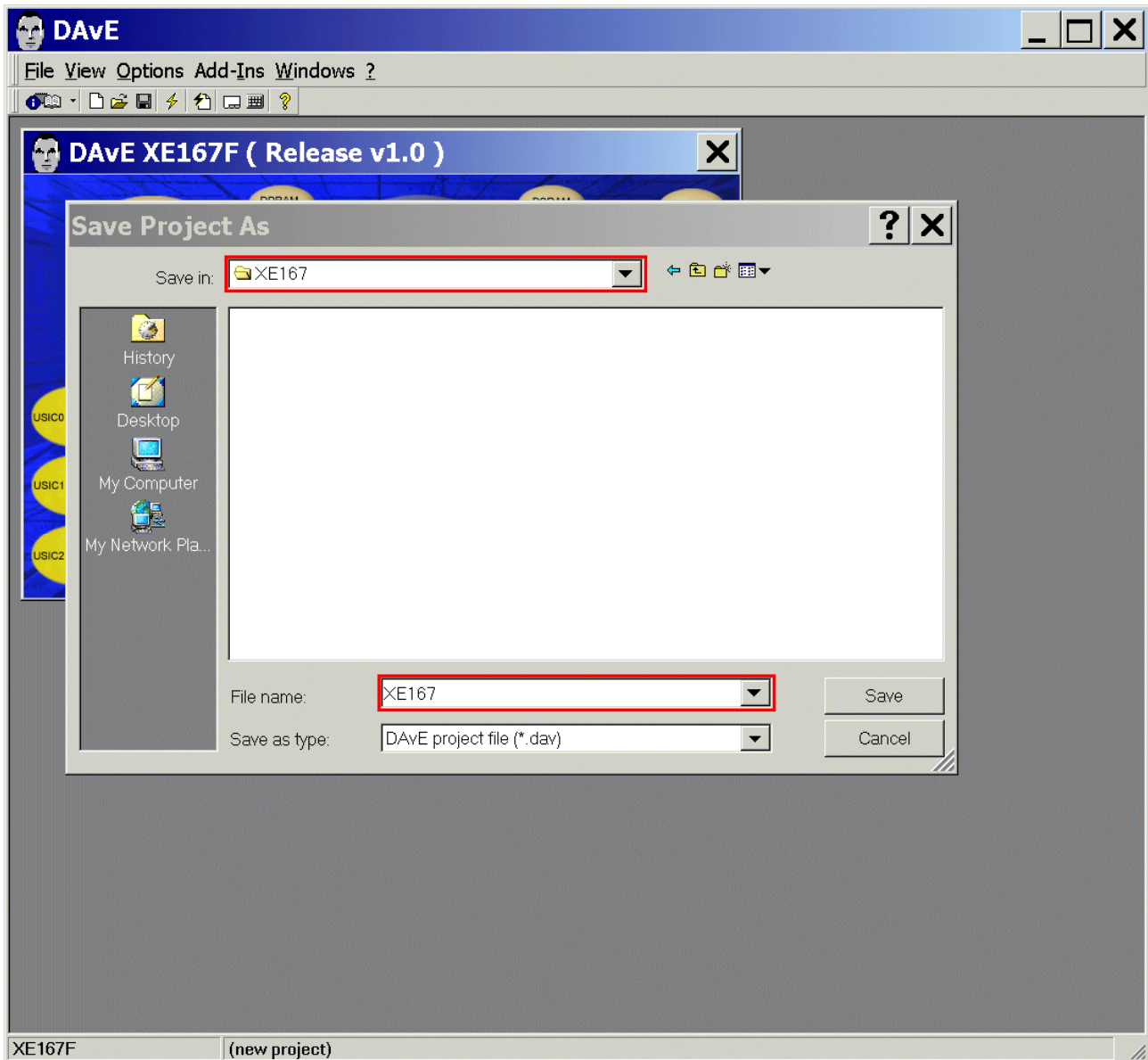
Notes: If you wish, you can insert your comments here.

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

Save the project:


File
Save

Save project: Save in C:\XE167 (create new directory)
File name: XE167



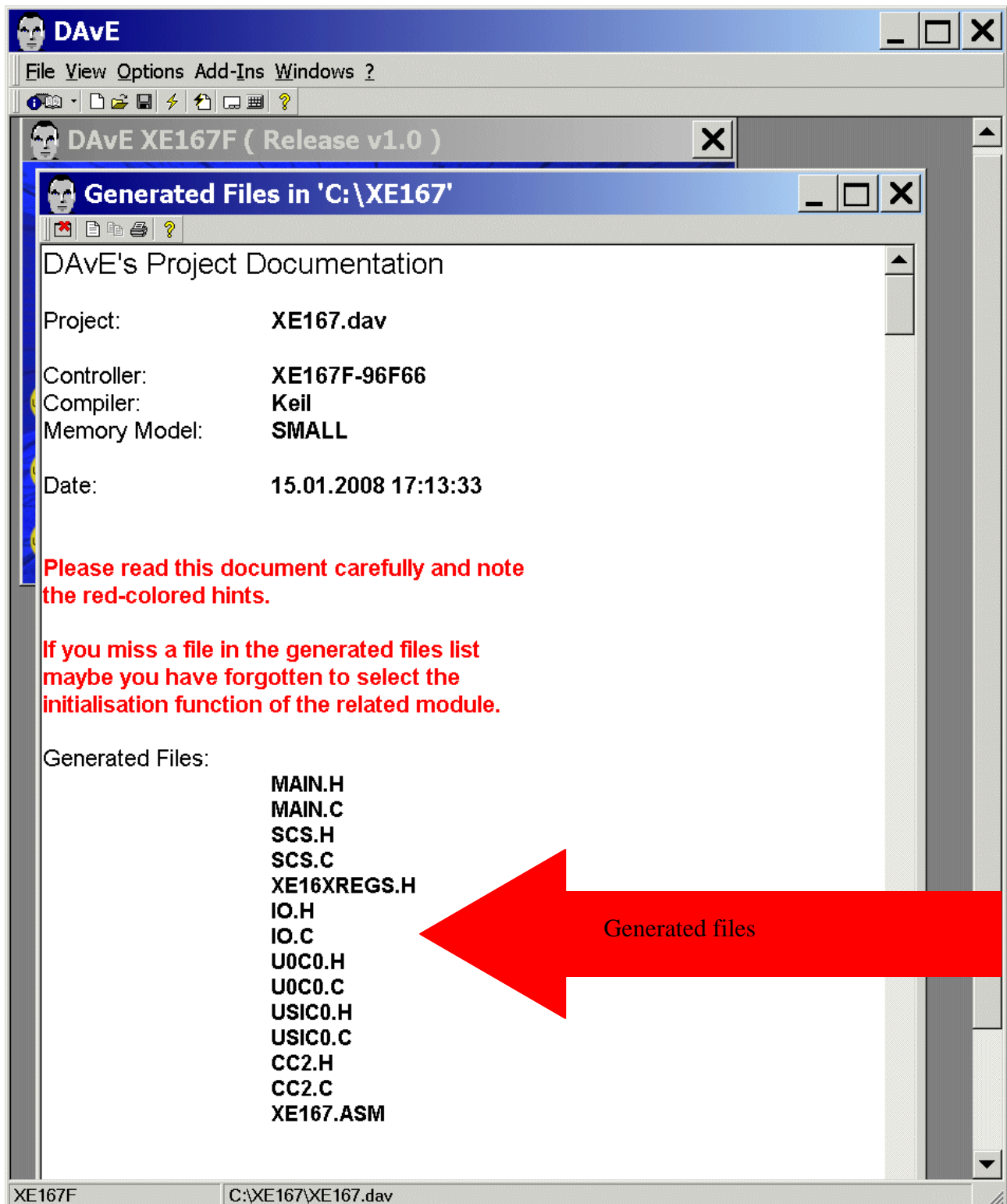
Save

Generate Code:

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



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



File - Exit

Save changes?

Click: **Yes**

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

Install the tool chain: You can download the Keil Development Tools @ <http://www.keil.com> :



Download and **Execute** C166V610A.EXE (- or any higher version) and install the Keil tool chain.



Start Keil μ Vision3 and open the DAVE Project:

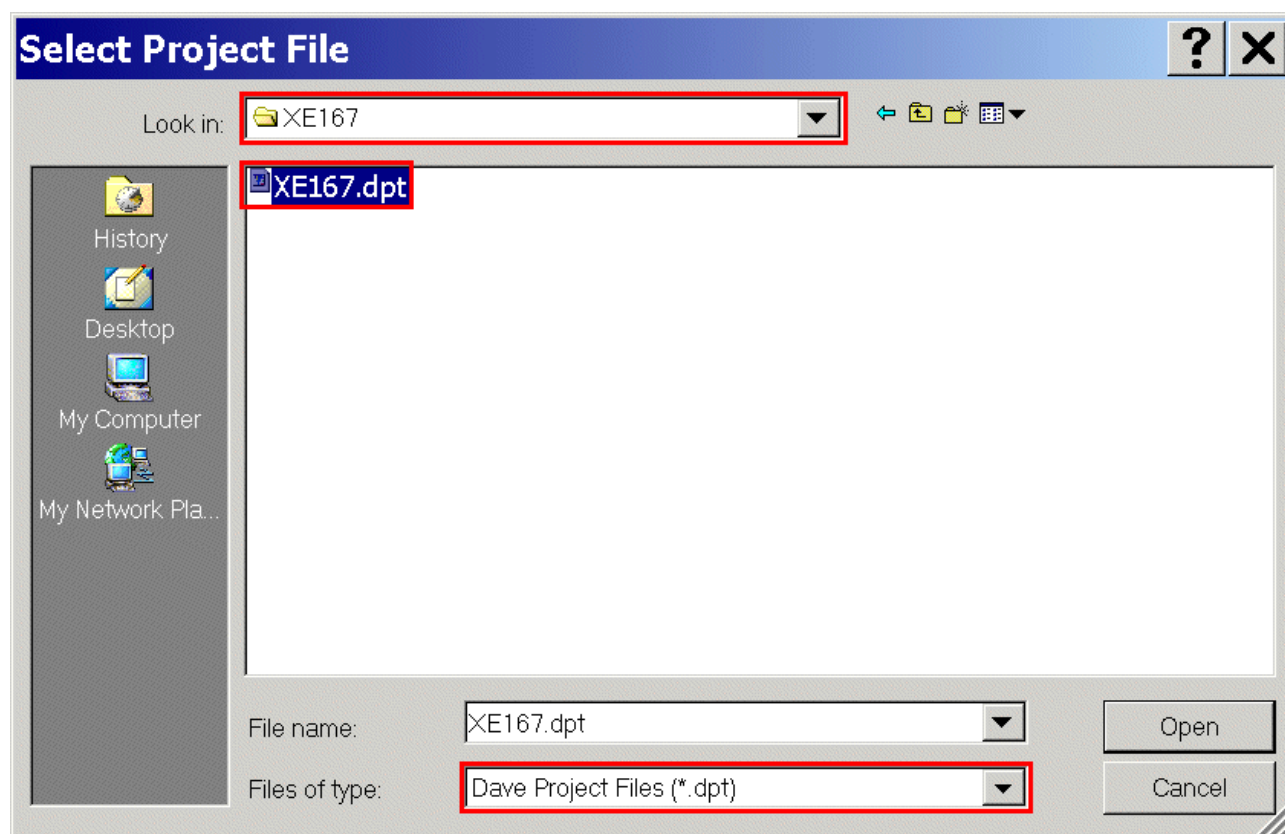
If you see an open project – close it: **Project - Close Project**

Project - Open Project

Select Project File: **Look in:** choose C:\XE167

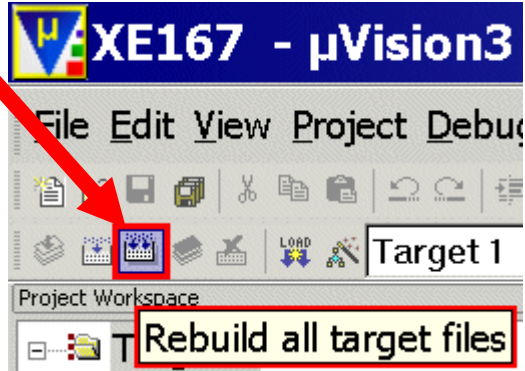
Select Project File: **Files of type:** choose Dave Project Files

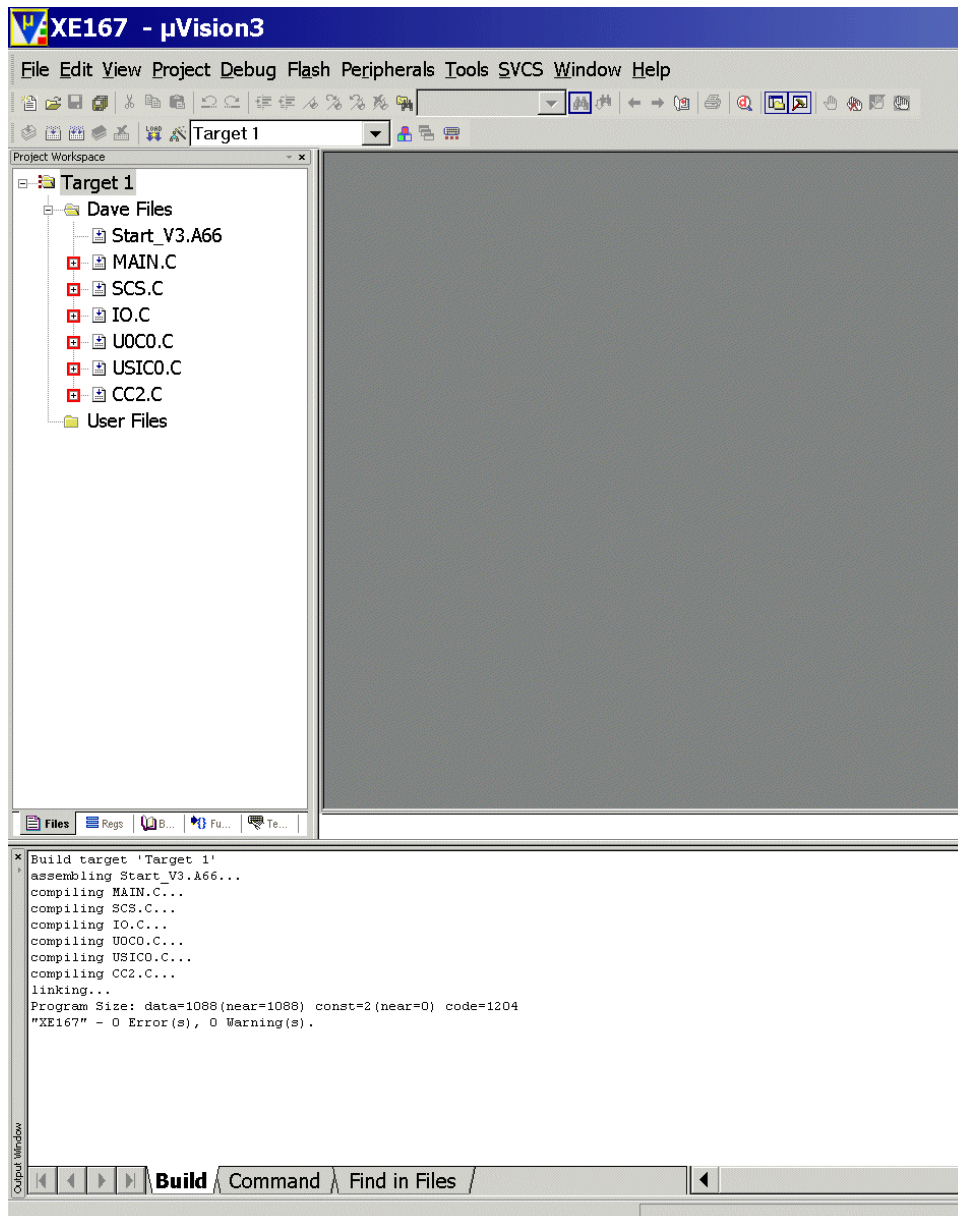
Click XE167.dpt



Open

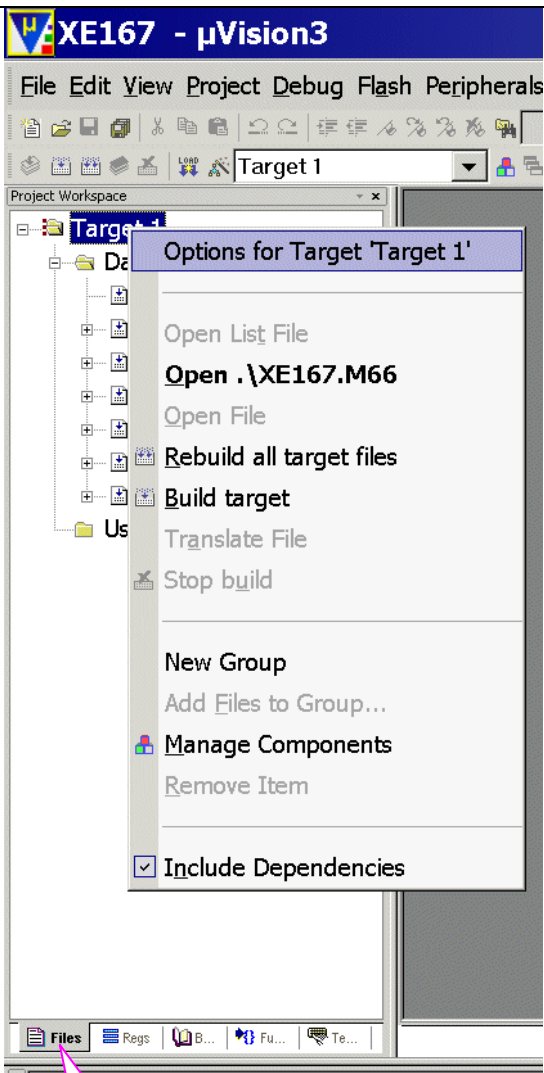
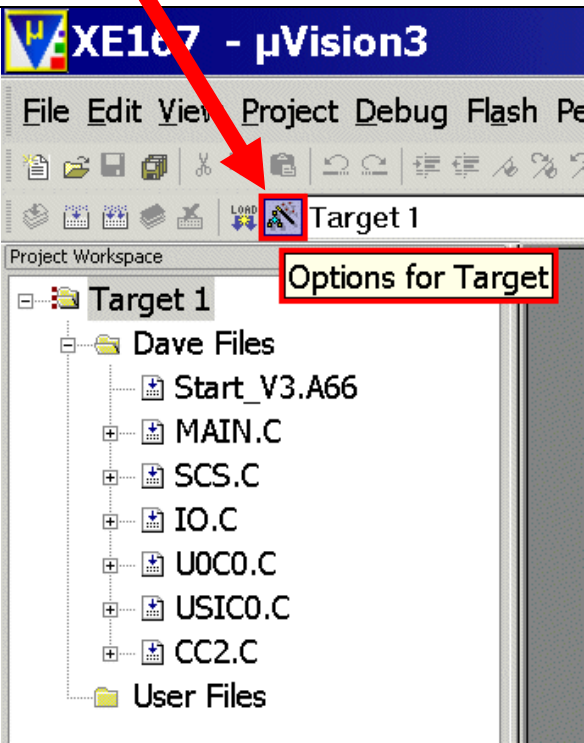
Generate „make“- file:

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



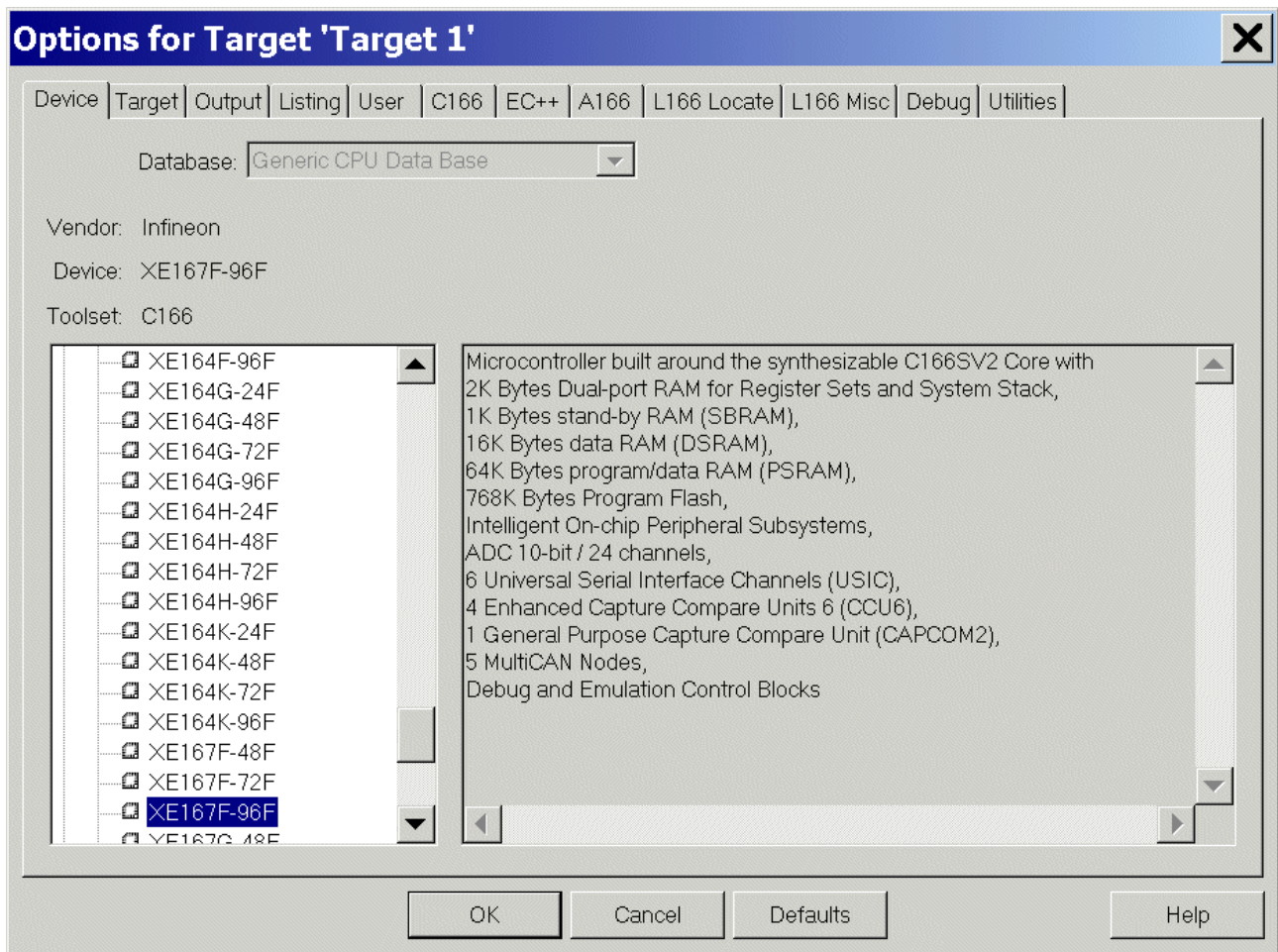
Configure:

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

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

Project Workspace, Files

Device: **check** XE167F-96F



Target: Clock(MHz): **check** 8.0

Target: **check** ☒ Use On-chip ROM

Target: **check** ☒ Use On-chip ROM

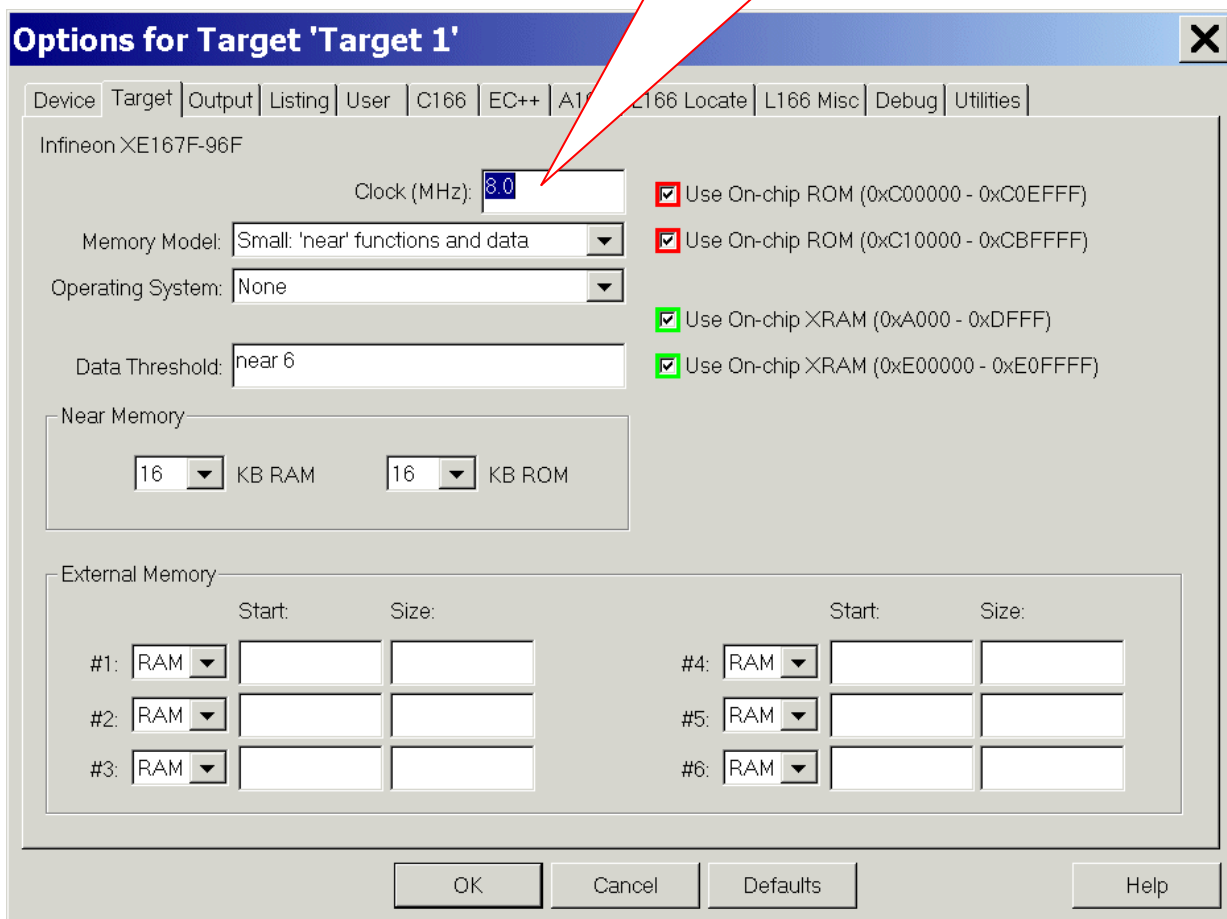
Target: **check** ☒ Use On-chip XRAM

Target: **check** ☒ Use On-chip XRAM

Note (Source: DAVe):

Configuration of the System Clock:

- VCO clock used, input clock is connected
- input frequency is 8,00 MHz
- configured system frequency is 66,00 MHz
- system clock is 66.00 MHz



Options for Target 'Target 1'

Device: Infineon XE167F-96F

Target: Clock (MHz): 8.0

Memory Model: Small: 'near' functions and data

Operating System: None

Data Threshold: near 6

Near Memory

16 KB RAM 16 KB ROM

External Memory

	Start:	Size:		Start:	Size:
#1: RAM			#4: RAM		
#2: RAM			#5: RAM		
#3: RAM			#6: RAM		

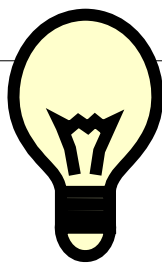
☒ Use On-chip ROM (0xC00000 - 0xC0EFFF)

☒ Use On-chip ROM (0xC10000 - 0xCBFFFF)

☒ Use On-chip XRAM (0xA000 - 0xDFFF)

☒ Use On-chip XRAM (0xE00000 - 0xE0FFFF)

OK Cancel Defaults Help



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

Options for Target 'Target 1'

Device
Target
Output
Listing
User
C166
EC++
A166
L166 Locate
L166 Misc
Debug
Utilities

Infineon XE167F-96F

Clock (MHz):

Memory Model:

Operating System:

Data Threshold:

☒ Use On-chip ROM (0xC00000 - 0xC0EFFF)
☒ Use On-chip ROM (0xC10000 - 0xCBFFFF)
☒ Use On-chip XRAM (0xA000 - 0xDFFF)
☒ Use On-chip XRAM (0xE00000 - 0xE0FFFF)

Table 3-1 XE16x Memory Map ¹⁾

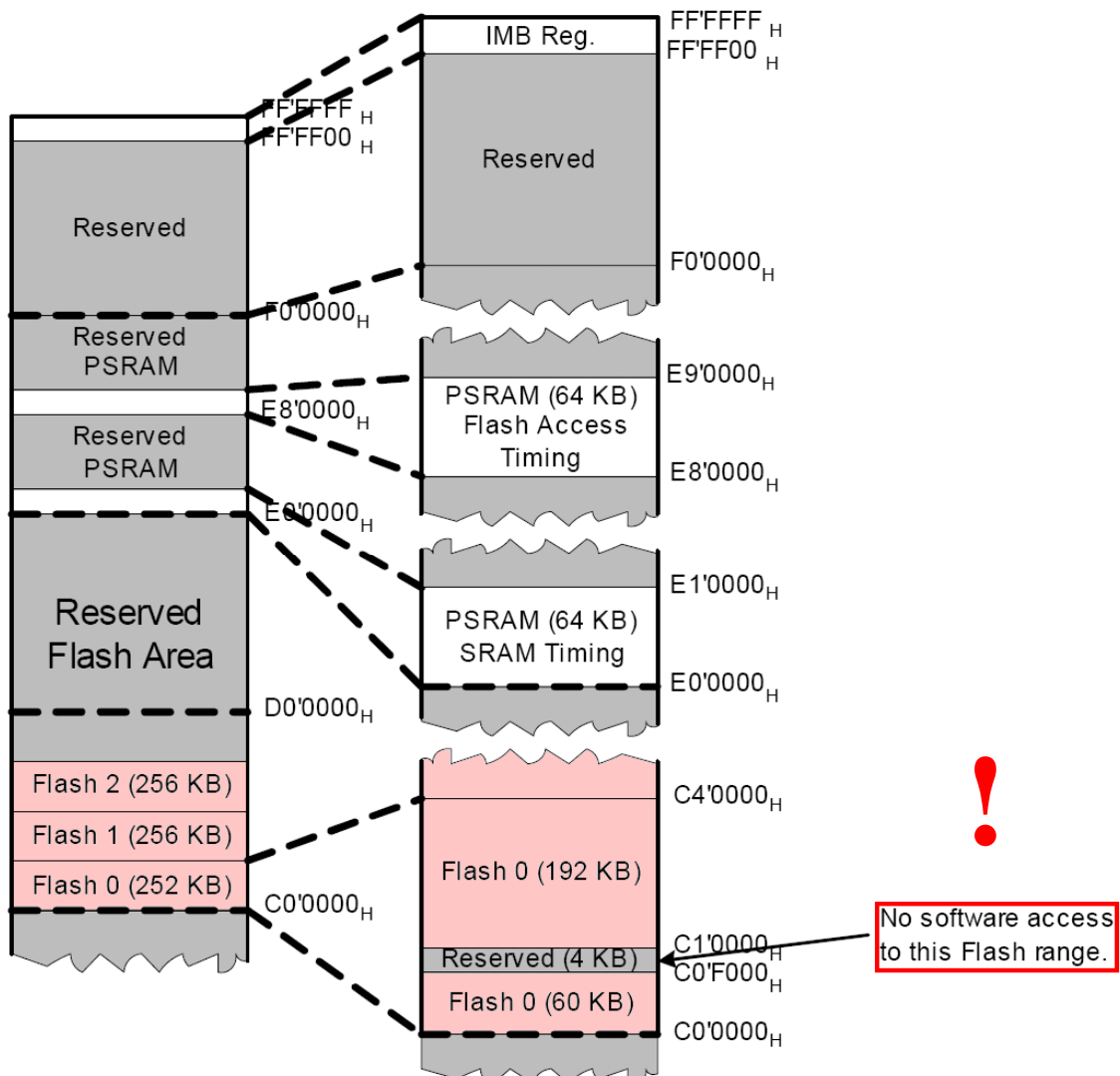
Address Area	Start Loc.	End Loc.	Area Size ²⁾	Notes
IMB register space	FF'FF00 _H	FF'FFFF _H	256 Bytes	
Reserved (access trap)	F0'0000 _H	FF'FEFF _H	< 1 MByte	Minus IMB registers.
Reserved for EPSRAM	E9'0000 _H	EF'FFFF _H	448 KBytes	
EPSRAM	E8'0000 _H	E8'FFFF _H	64 KBytes	PSRAM with Flash timing.
Reserved for PSRAM	E1'0000 _H	E7'FFFF _H	448 KBytes	
PSRAM	E0'0000 _H	E0'FFFF _H	64 KBytes	Program SRAM.
Reserved for Flash	CC'0000 _H	DF'FFFF _H	<1.25 MBytes	
Flash 2	C8'0000 _H	CB'FFFF _H	256 KBytes	
Flash 1	C4'0000 _H	C7'FFFF _H	256 KBytes	
Flash 0	C0'0000 _H	C3'FFFF _H	252 KBytes ³⁾	Minus res. seg.
External memory area	40'0000 _H	BF'FFFF _H	8 MBytes	
External IO area ⁴⁾	20'5800 _H	3F'FFFF _H	< 2 MBytes	Minus CAN/USIC
USIC registers	20'4000 _H	20'57FF _H	6 KBytes	Accessed via EBC
MultiCAN registers	20'0000 _H	20'3FFF _H	16 KBytes	Accessed via EBC
External memory area	01'0000 _H	1F'FFFF _H	< 2 MBytes	Minus segment 0
SFR area	00'FE00 _H	00'FFFF _H	0.5 KBytes	
Dual-port RAM (DPRAM)	00'F600 _H	00'FDFF _H	2 KBytes	
Reserved for DPRAM	00'F200 _H	00'F5FF _H	1 KBytes	
ESFR area	00'F000 _H	00'F1FF _H	0.5 KBytes	
XSFR area	00'E000 _H	00'EFFF _H	4 KBytes	
Data SRAM (DSRAM)	00'A000 _H	00'DFFF _H	16 KBytes	
Reserved for DSRAM	00'8000 _H	00'9FFF _H	8 KBytes	
External memory area	00'0000 _H	00'7FFF _H	32 KBytes	



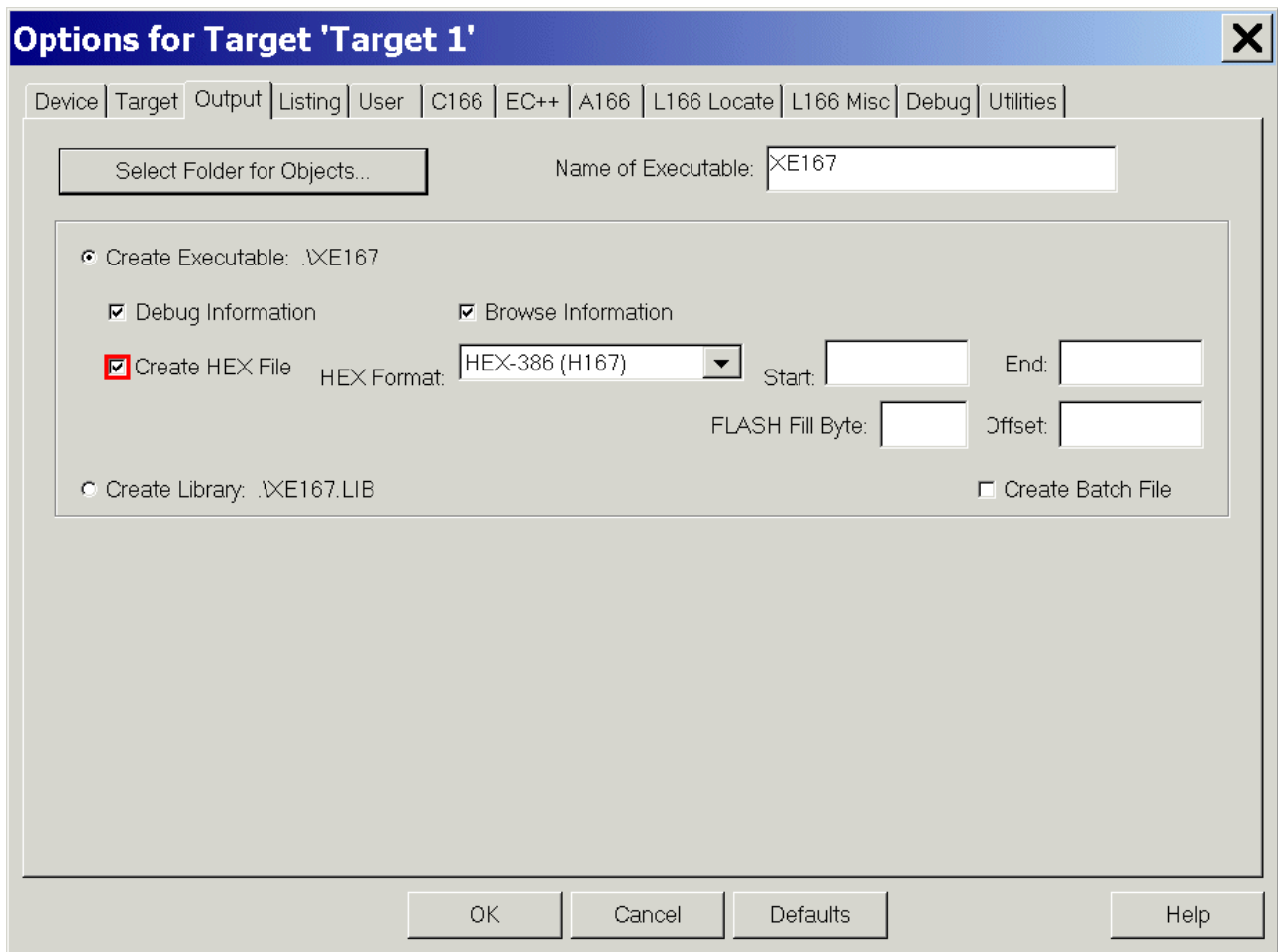
Additional information: Memory Map (Source: User's Manual):

Options for Target 'Target 1'

Device	Target	Output	Listing	User	C166	EC++	A166	L166 Locate	L166 Misc	Debug	Utilities
Infineon XE167F-96F											
Clock (MHz):		8.0		<input checked="" type="checkbox"/> Use On-chip ROM (0xC00000 - 0xC0EFFF)							
Memory Model:		Small: 'near' functions and data		<input checked="" type="checkbox"/> Use On-chip ROM (0xC10000 - 0xCBFFFF)							
Operating System:		None		<input checked="" type="checkbox"/> Use On-chip XRAM (0xA000 - 0xDFFF)							
Data Threshold:		near 6		<input checked="" type="checkbox"/> Use On-chip XRAM (0xE00000 - 0xE0FFFF)							



Output: **click** ☒ Create HEX File



Options for Target 'Target 1'

Device | Target | Output | Listing | User | C166 | EC++ | A166 | L166 Locate | L166 Misc | Debug | Utilities

Select Folder for Objects... Name of Executable: XE167

☒ Create Executable: .\XE167

☒ Debug Information ☒ Browse Information

☒ Create HEX File HEX Format: HEX-386 (H167) Start: End: FLASH Fill Byte: Offset:

☐ Create Library: .\XE167.LIB ☐ Create Batch File

OK Cancel Defaults Help

Note:

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



Listing: (do nothing)

Options for Target 'Target 1' [X]

Device | Target | Output | Listing | User | C166 | EC++ | A166 | L166 Locate | L166 Misc | Debug | Utilities

Select Folder for Listings... Page Width: 120 Page Length: 65

☒ C Compiler Listing: *.lst

☒ Conditional ☐ Symbols ☐ #include Files ☐ Assembly Code

☐ C Preprocessor Listing: *.i

☒ Assembler Listing: *.lst

☒ Conditional ☒ Symbols Macros: Final expansion only ☐ Cross Reference

☒ Linker Listing: .XE167.m66

☒ Memory Map ☒ Public Symbols ☒ Line Numbers ☐ Cross Reference

☒ Local Symbols ☒ Comment Records ☒ Generated Symbols

☒ Library Symbols

OK Cancel Defaults Help

User: (do nothing)

Options for Target 'Target 1' [X]

Device | Target | Output | Listing | **User** | C166 | EC++ | A166 | L166 Locate | L166 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

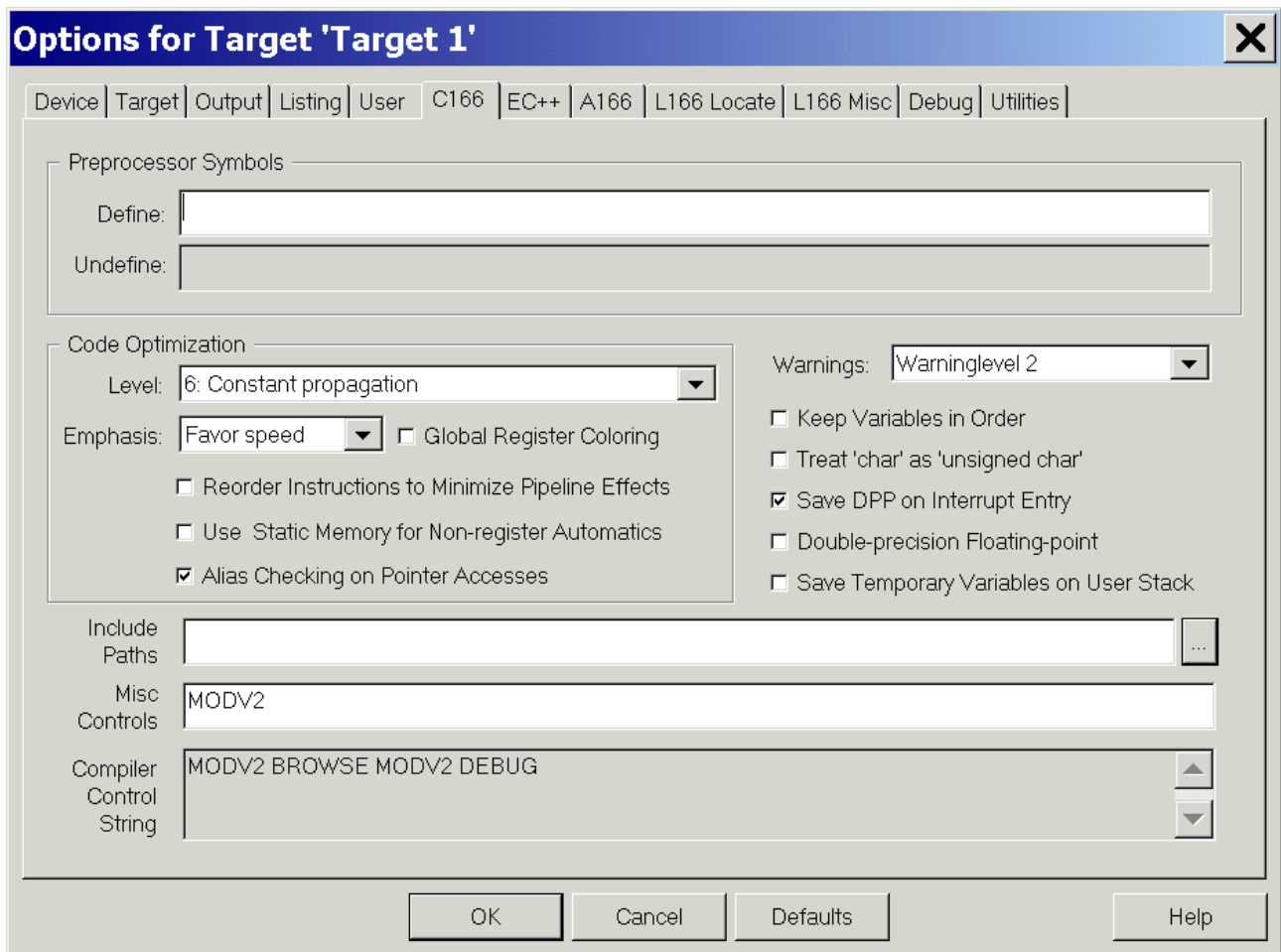
☐ Run #1: [] ... ☐ DOS16

☐ Run #2: [] ... ☐ DOS16

☒ Beep When Complete ☐ Start Debugging

OK Cancel Defaults Help

C166: (do nothing)



Options for Target 'Target 1'

Device | Target | Output | Listing | User | C166 | EC++ | A166 | L166 Locate | L166 Misc | Debug | Utilities

Preprocessor Symbols

Define:

Undefine:

Code Optimization

Level: 6: Constant propagation

Emphasis: Favor speed ☐ Global Register Coloring

☐ Reorder Instructions to Minimize Pipeline Effects

☐ Use Static Memory for Non-register Automatics

☒ Alias Checking on Pointer Accesses

Warnings: Warninglevel 2

☐ Keep Variables in Order

☐ Treat 'char' as 'unsigned char'

☒ Save DPP on Interrupt Entry

☐ Double-precision Floating-point

☐ Save Temporary Variables on User Stack

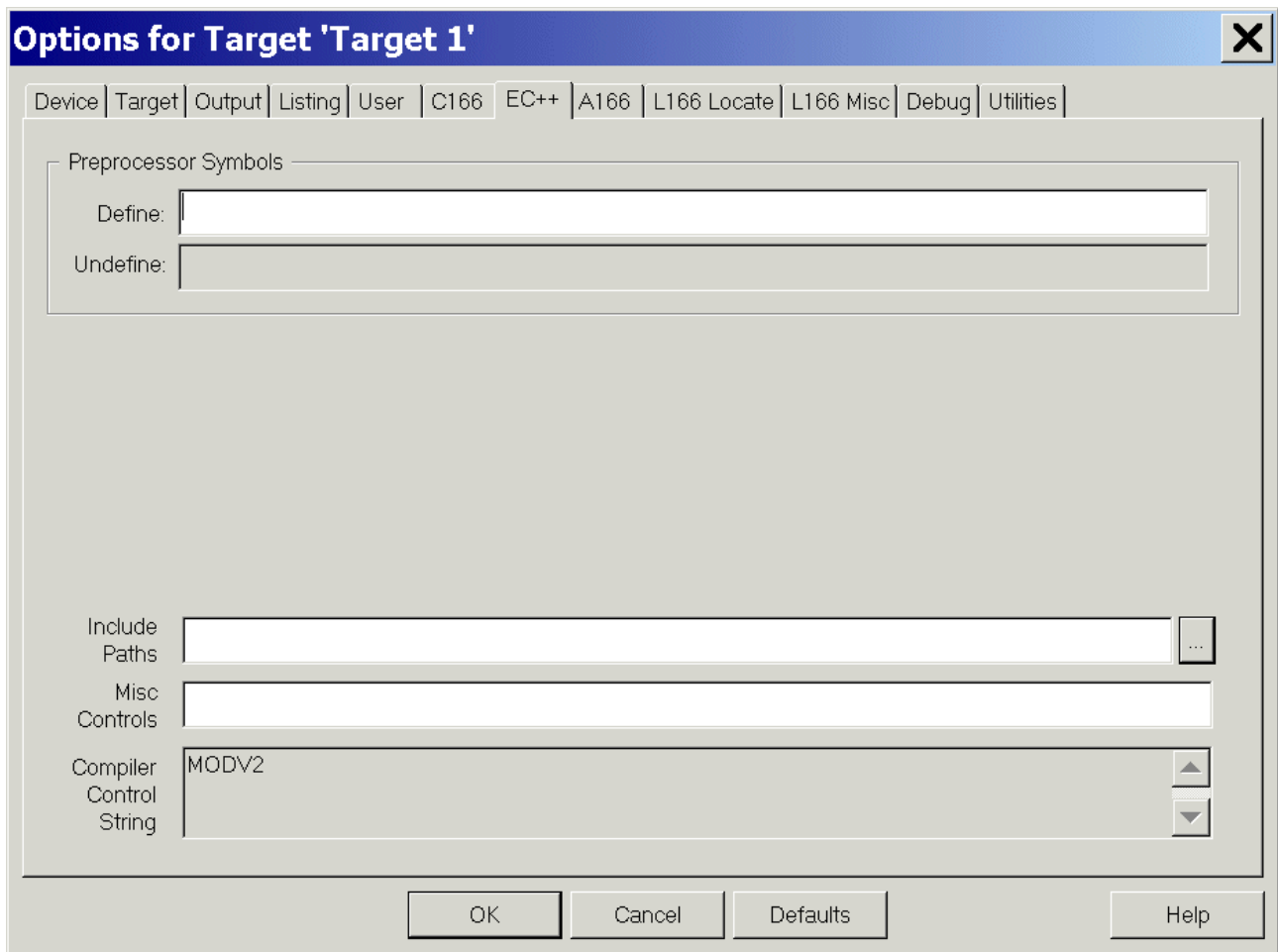
Include Paths: ...

Misc Controls: MODV2

Compiler Control String: MODV2 BROWSE MODV2 DEBUG

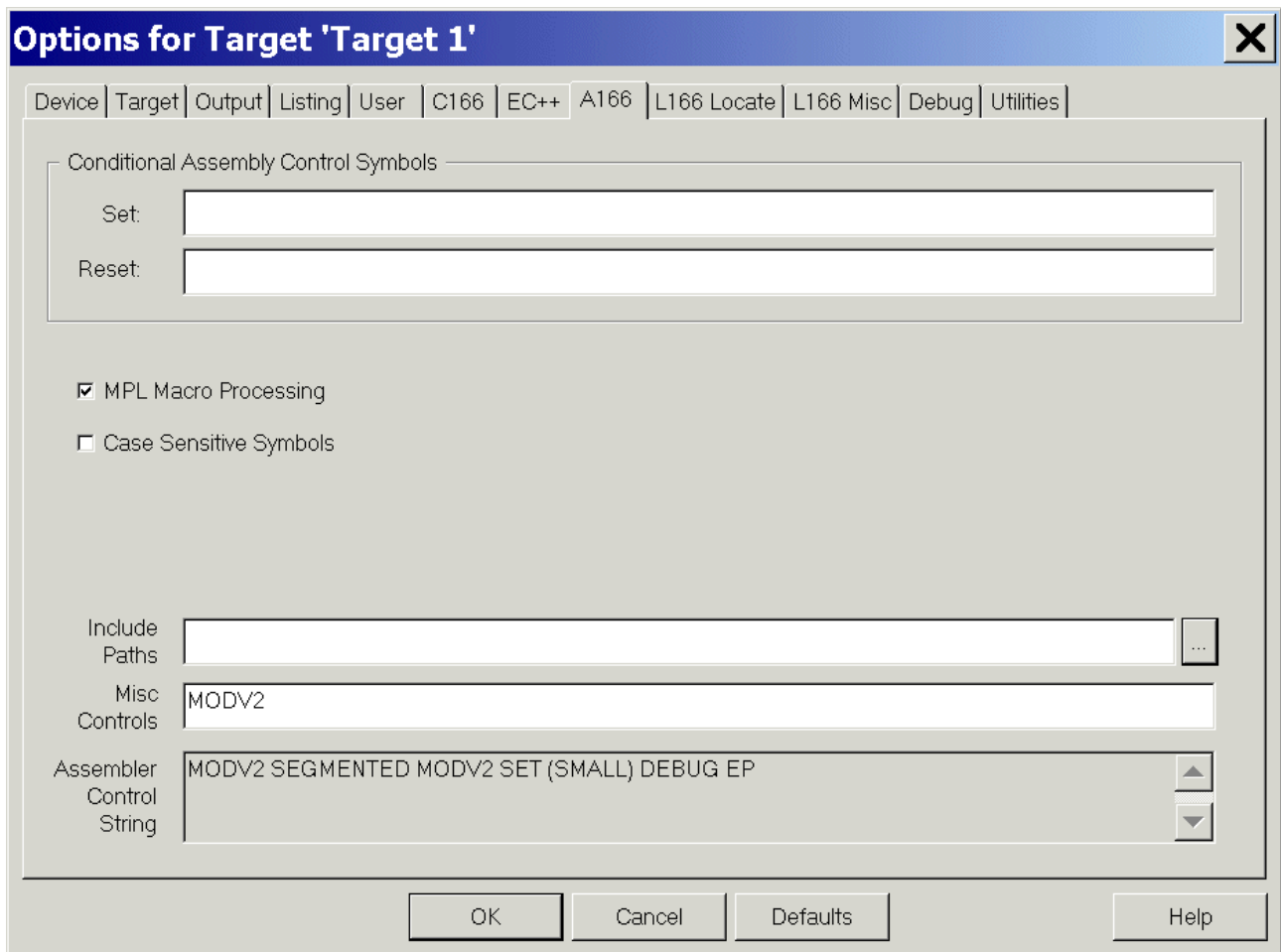
OK Cancel Defaults Help

EC++: (do nothing)



The dialog box 'Options for Target 'Target 1'' features a tabbed interface with the following tabs: Device, Target, Output, Listing, User, C166, EC++, A166, L166 Locate, L166 Misc, Debug, and Utilities. The 'EC++' tab is currently selected. Within this tab, there is a section for 'Preprocessor Symbols' containing 'Define:' and 'Undefine:' text labels followed by empty input fields. Below this, there are three more input fields: 'Include Paths' with a browse button (...), 'Misc Controls', and 'Compiler Control String' which currently displays 'MODV2' and has up/down arrow buttons. At the bottom of the dialog are four buttons: 'OK', 'Cancel', 'Defaults', and 'Help'.

A166: (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, C166, EC++, A166, L166 Locate, L166 Misc, Debug, and Utilities. The "A166" tab is currently selected.

Under the "Conditional Assembly Control Symbols" section, there are two text input fields labeled "Set:" and "Reset:", both of which are empty.

Below this, there are two checkboxes:
- ☒ MPL Macro Processing
- ☐ Case Sensitive Symbols

Further down, there is an "Include Paths" section with a text input field that is empty and a browse button (three dots).

Below that is a "Misc Controls" section with a text input field containing the text "MODV2".

At the bottom of the main area is an "Assembler Control String" section with a text input field containing the text "MODV2 SEGMENTED MODV2 SET (SMALL) DEBUG EP" and up/down arrow buttons.

The dialog box has four buttons at the bottom: "OK", "Cancel", "Defaults", and "Help".

L166 Locate: (do nothing)

Options for Target 'Target 1'

Device

Target

Output

Listing

User

C166

EC++

A166

L166 Locate

L166 Misc

Debug

Utilities

☒ Use Memory Layout from Target Dialog
 C166 Variable Initialization Tables

0xC10000 - 0xCBFFFF

DPP Usage

☐ DPPUSE
 ndata

dpp2

 nconst

dpp1

Target Classes

ICODE (0xC00000-0xC0EFFF), NCODE (0xC10000-0xC1FFFF),
 FCONST (0xC00000-0xC0EFFF, 0xC10000-0xCBFFFF), HCONST (0xC00000-0xC0EFFF, 0xC10000-0xC1FFFF),
 XCONST (0xC00000-0xC0EFFF, 0xC10000-0xCBFFFF), NCONST (0xC04000-0xC07FFF),

User Classes

User Sections

Linker Control String

TO "XE167"
 CLASSES (ICODE (0xC00000-0xC0EFFF), NCODE (0xC10000-0xC1FFFF),
 FCONST (0xC00000-0xC0EFFF, 0xC10000-0xCBFFFF), HCONST (0xC00000-0xC0EFFF, 0xC10000-0xC1FFFF),
 XCONST (0xC00000-0xC0EFFF, 0xC10000-0xCBFFFF), NCONST (0xC04000-0xC07FFF))

OK

Cancel

Defaults

Help

L166 Misc: Interrupt Vector Table Address: insert 0x0C00000

Options for Target 'Target 1' [X]

Device | Target | Output | Listing | User | C166 | EC++ | A166 | L166 Locate | **L166 Misc** | Debug | Utilities

Warnings
Level 2 [v] Disable Warning Numbers: []

☐ use linker control file:
[Create...] [Browse...] [Edit...]

☐ Create Relocatable Output File (LINKONLY) Interrupt Vector Table Address: 0x0C00000

Assign []

RegBank []

Reserve []

Misc Controls [] [v]

Linker Control String TO "XE167"
VECTAB (0x0C00000)
CLASSES (ICODE (0xC00000-0xC0EFFF), NCODE (0xC10000-0xC1FFFF),

[OK] [Cancel] [Defaults] [Help]

Debug: (do nothing)

Options for Target 'Target 1' [X]

Device | Target | Output | Listing | User | C166 | EC++ | A166 | L166 Locate | L166 Misc | **Debug** | Utilities

☒ Use Simulator Settings ☐ Use: ULINK Driver for XC16x Settings

☐ Limit Speed to Real-Time

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

☒ Breakpoints ☒ Toolbox
☒ Watchpoints & PA
☒ Memory Display

CPU DLL: S166.DLL Parameter: -cMODV2

Dialog DLL: D167.DLL Parameter: -pXE167F

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

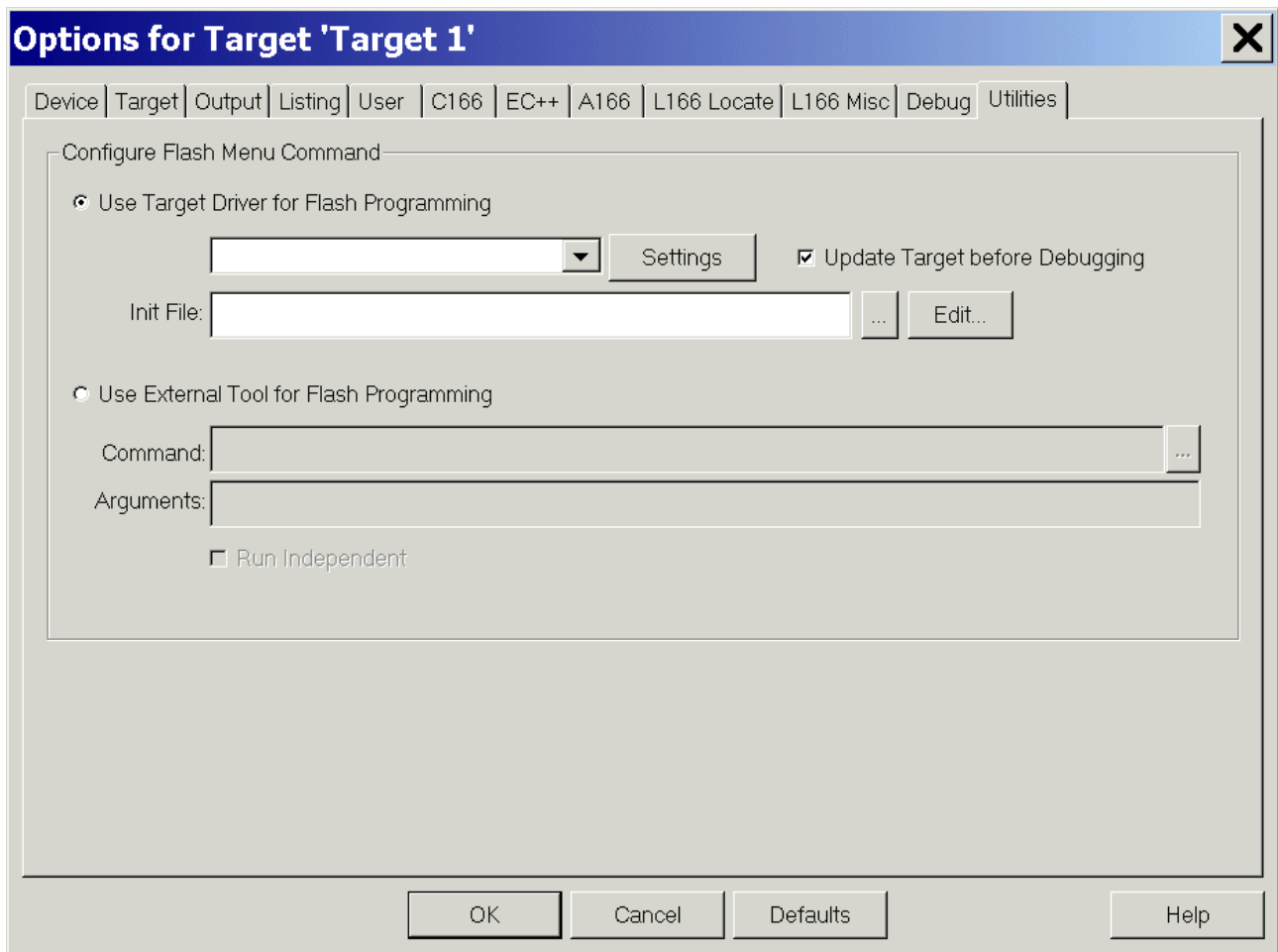
☒ Breakpoints ☒ Toolbox
☒ Watchpoints
☒ Memory Display

Driver DLL: S166.DLL Parameter: -cMODV2

Dialog DLL: T167.DLL Parameter: -pXE167F

OK Cancel Defaults Help

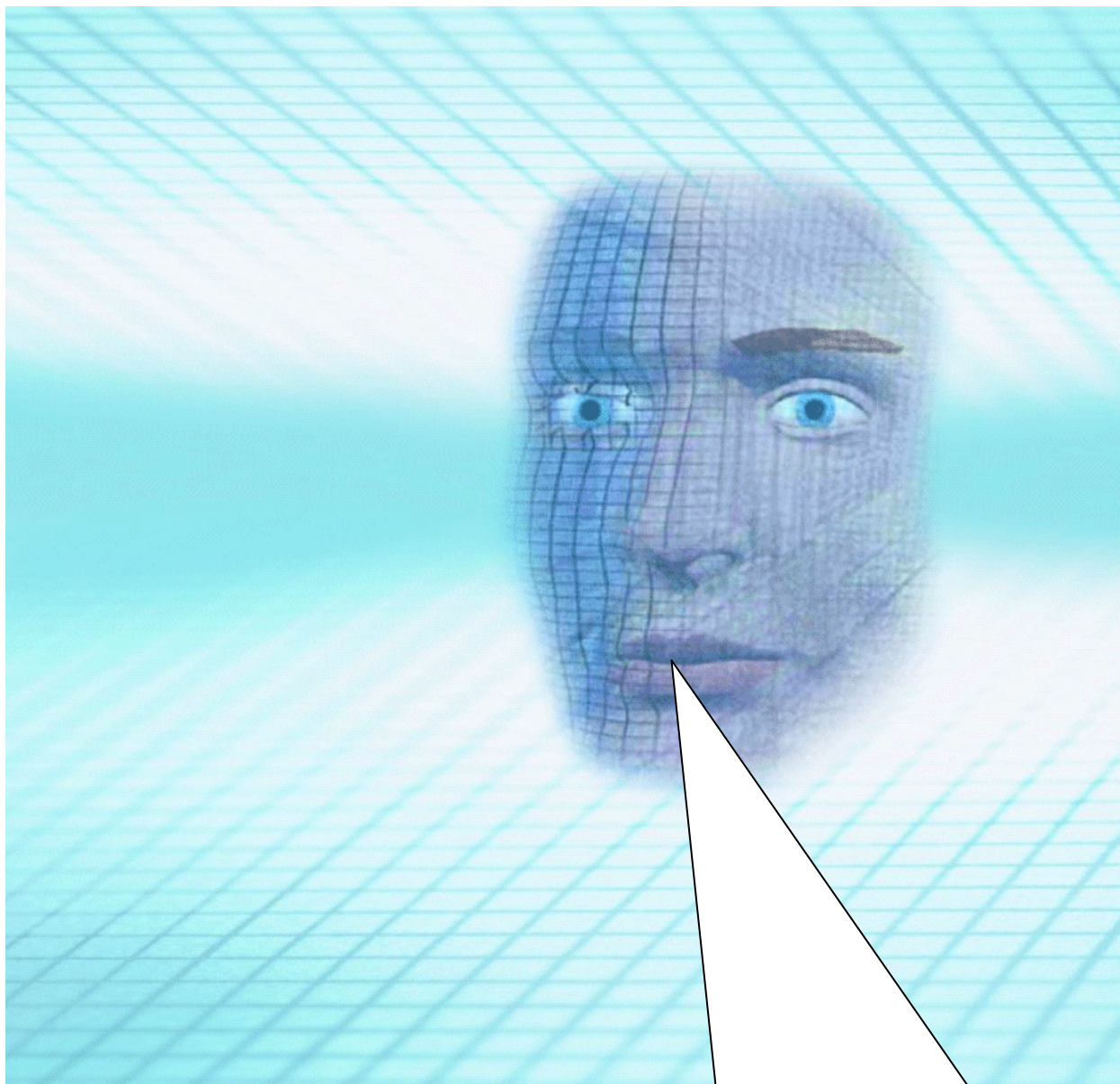
Utilities: (do nothing)



The image shows a Windows-style dialog box titled "Options for Target 'Target 1'". It has a tabbed interface with tabs for Device, Target, Output, Listing, User, C166, EC++, A166, L166 Locate, L166 Misc, Debug, and Utilities. The Utilities tab is currently selected. Inside the dialog, there is a section titled "Configure Flash Menu Command". It contains two radio buttons: "Use Target Driver for Flash Programming" (which is selected) and "Use External Tool for Flash Programming". Under the selected radio button, there is a dropdown menu, a "Settings" button, and a checked checkbox labeled "Update Target before Debugging". Below this is an "Init File:" label followed by a text input field, an ellipsis button "...", and an "Edit..." button. Under the unselected radio button, there are "Command:" and "Arguments:" labels, each followed by a text input field and an ellipsis button "...". At the bottom of this section is a checkbox labeled "Run Independent". At the very bottom of the dialog are four buttons: "OK", "Cancel", "Defaults", and "Help".

OK

Insert your application specific program:



Note:

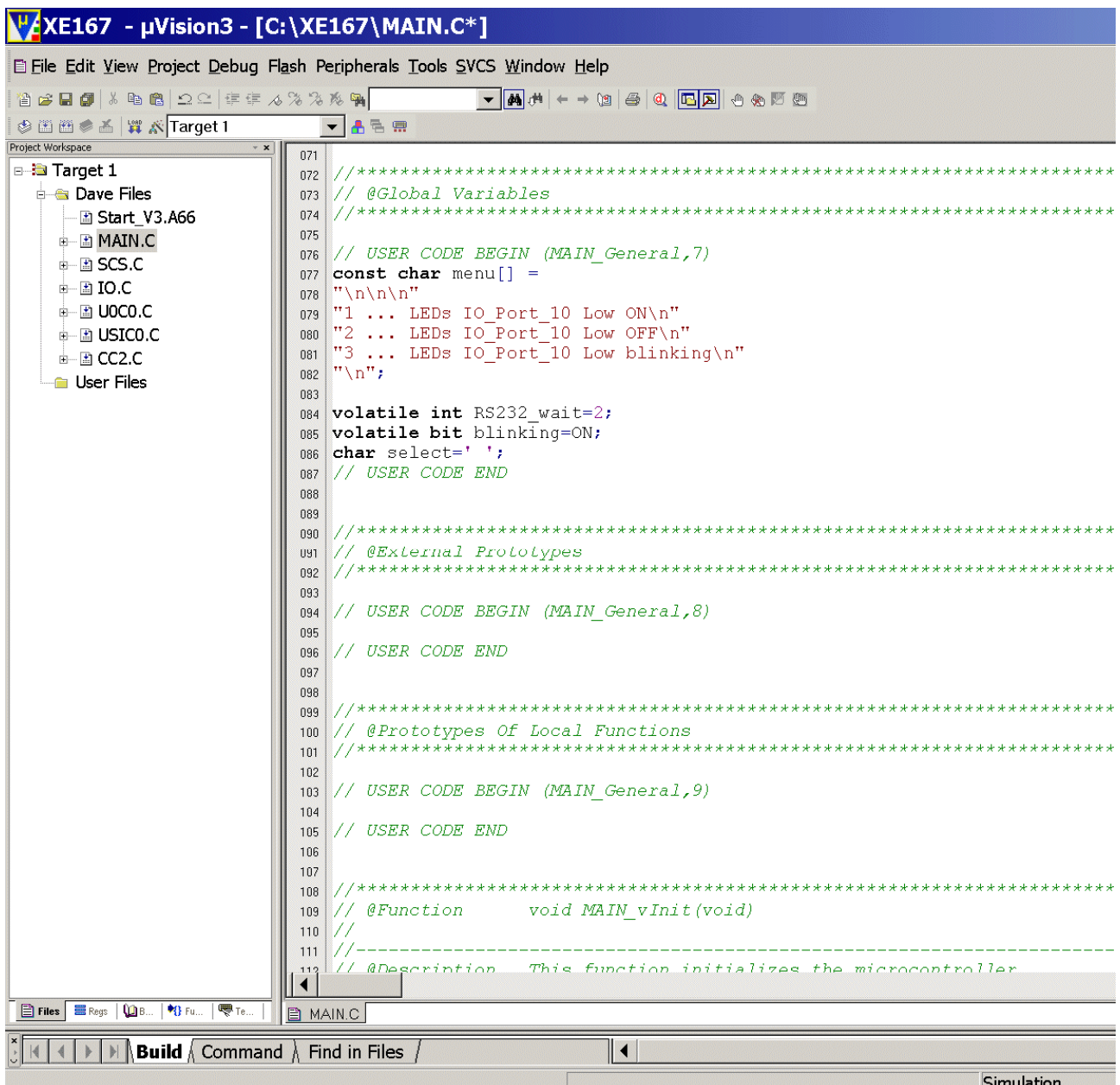
DAvE doesn't change code which is inserted between '`// USER CODE BEGIN`' and '`// USER CODE END`'. Therefore, whenever adding code to DAvE's generated code, write it between '`// USER CODE BEGIN`' and '`// USER CODE END`'.

If you wish to change DAvE's generated code or add code outside these 'USER CODE' sections you will have to insert/modify your changes each time after letting DAvE regenerate code!

Double click **MAIN.C** and insert Global Variables:

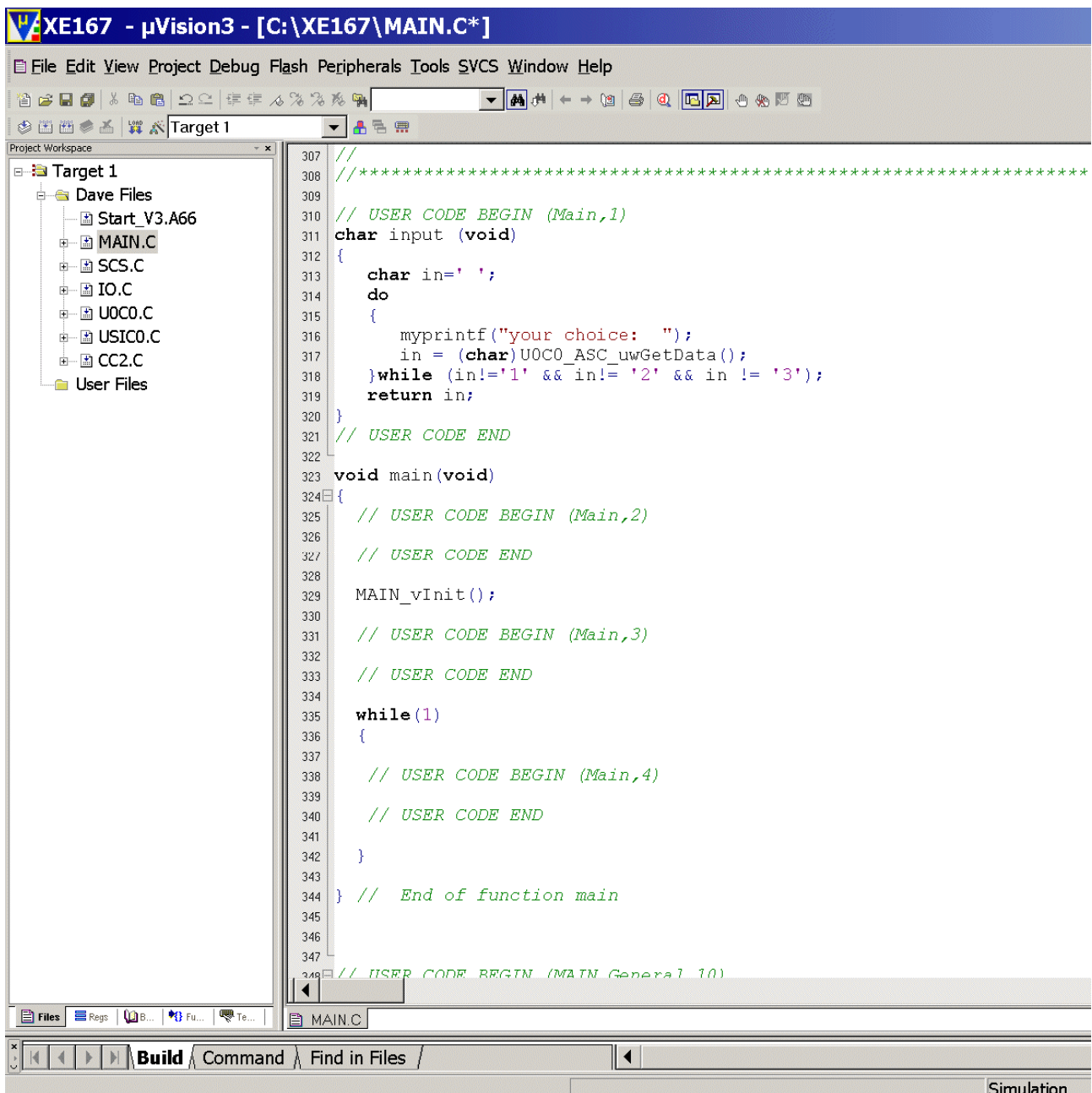
```
const char menu[] =
"\n\n"
"1 ... LEDs IO_Port_10 Low ON\n"
"2 ... LEDs IO_Port_10 Low OFF\n"
"3 ... LEDs IO_Port_10 Low blinking\n"
"\n";

volatile int RS232_wait=2;
volatile bit blinking=ON;
char select=' ';
```



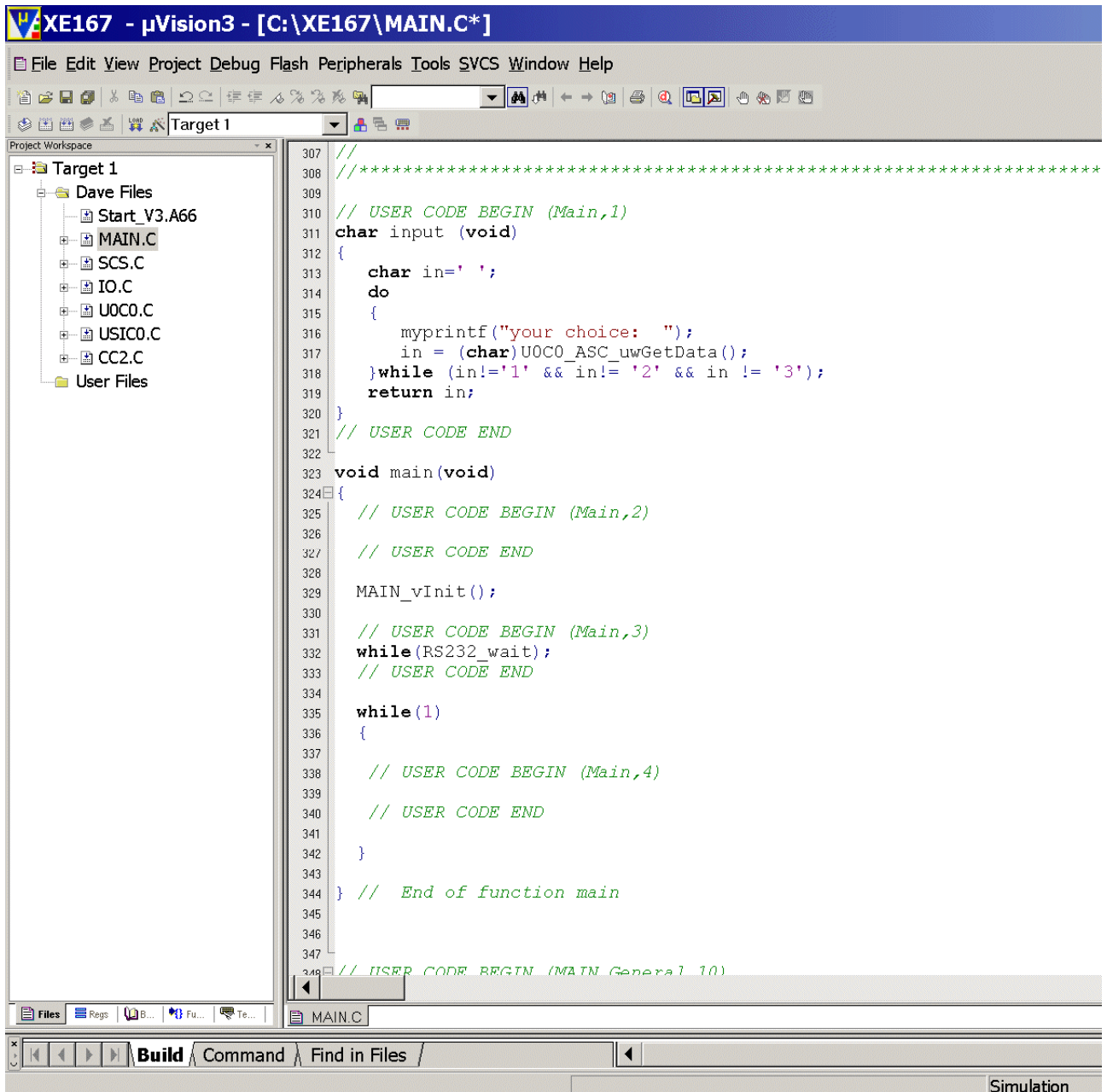
Double click MAIN.C and insert the function input():

```
char input (void)
{
    char in=' ';
    do
    {
        myprintf("your choice: ");
        in = (char)U0C0_ASC_uwGetData();
    }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);
```



```

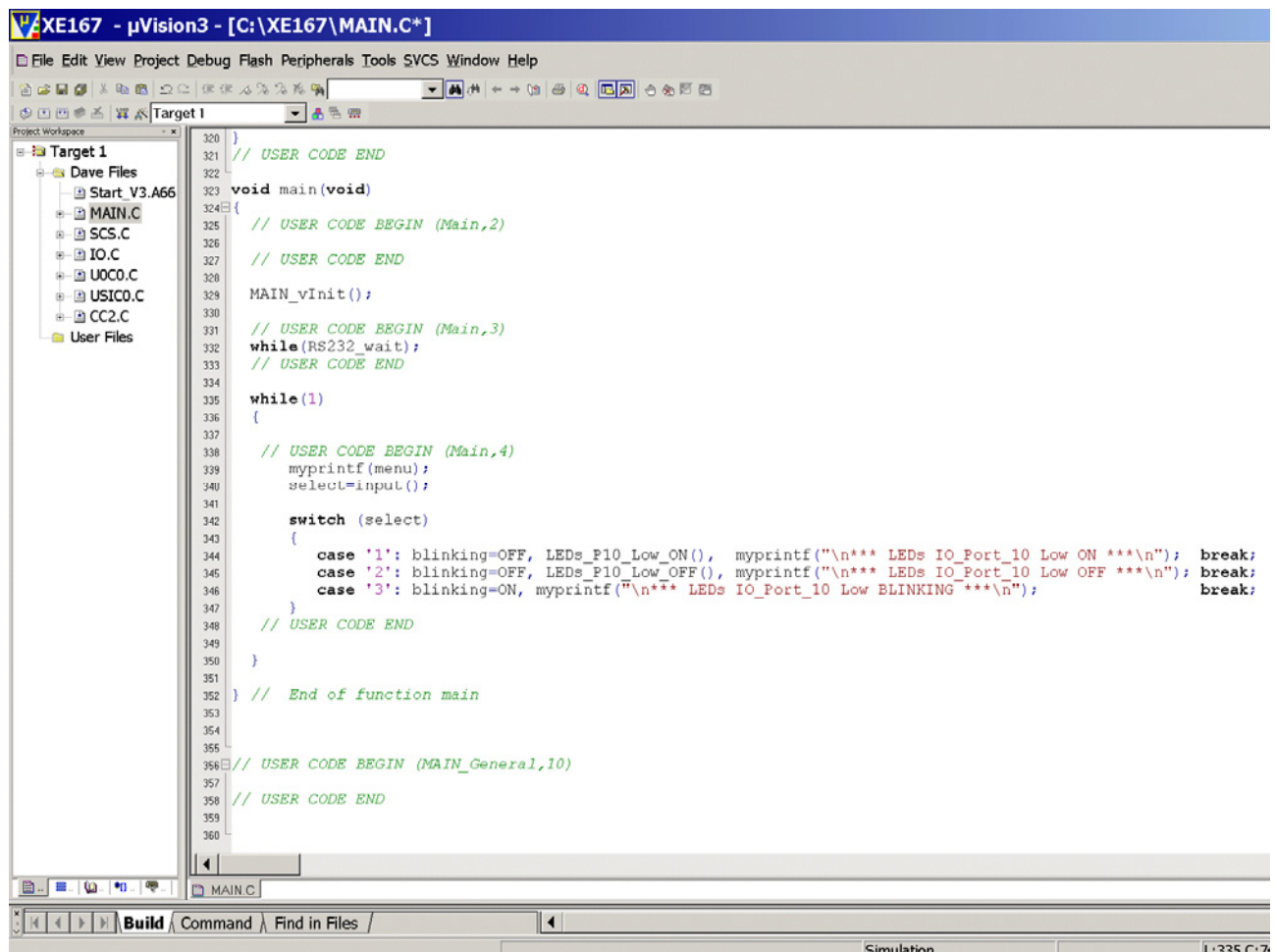
307 //
308 //*****
309 // USER CODE BEGIN (Main,1)
310 char input (void)
311 {
312     char in=' ';
313     do
314     {
315         myprintf("your choice: ");
316         in = (char)U0C0_ASC_uwGetData();
317         while (in!='1' && in!= '2' && in != '3');
318         return in;
319     }
320 }
321 // USER CODE END
322
323 void main(void)
324 {
325     // USER CODE BEGIN (Main,2)
326     // USER CODE END
327
328     MAIN_vInit();
329
330     // USER CODE BEGIN (Main,3)
331     while(RS232_wait);
332     // USER CODE END
333
334     while(1)
335     {
336
337         // USER CODE BEGIN (Main,4)
338         // USER CODE END
339
340     }
341 } // End of function main
342
343
344
345
346
347
348 // USER CODE BEGIN (MAIN General 10)

```

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

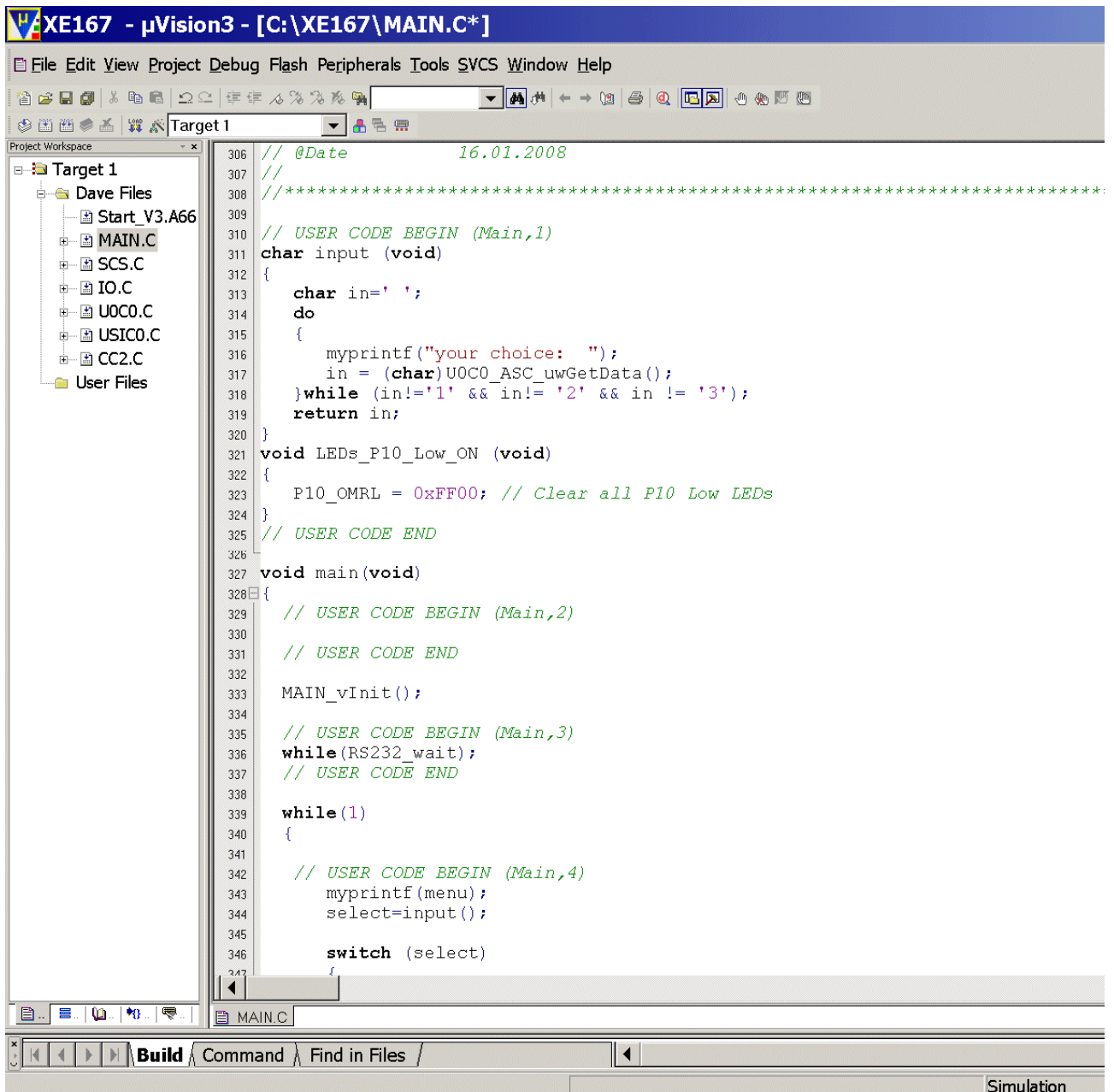
```
myprintf(menu);
select=input();

switch (select)
{
    case '1': blinking=OFF, LEDs_P10_Low_ON(), myprintf("\n*** LEDs IO_Port_10 Low ON ***\n"); break;
    case '2': blinking=OFF, LEDs_P10_Low_OFF(), myprintf("\n*** LEDs IO_Port_10 Low OFF ***\n"); break;
    case '3': blinking=ON, myprintf("\n*** LEDs IO_Port_10 Low BLINKING ***\n"); break;
}
```



Double click **MAIN.C** and insert the function **LEDs_P10_Low_ON()**:

```
void LEDs_P10_Low_ON (void)
{
    P10_OMRL = 0xFF00; // Clear all P10 Low LEDs
}
```





Additional information: Port Output Modification Register (Source: User's Manual):

Pn_OMRL (n=6-11)

Port n Output Modification Register LowXSFR (E9C0_H+4*n) Reset Value: XXXX_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC	PC	PC	PC	PC	PC	PC	PC	PS	PS	PS	PS	PS	PS	PS	PS
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Field	Bits	Type	Description
PSx (x = 0-7)	x	W	Port Set Bit x Setting this bit sets or toggles the corresponding bit in the port output register Pn_OUT (see Table 7-4). On a read access, this bit returns 0.
PCx (x = 0-7)	x + 8	W	Port Clear Bit x Setting this bit clears or toggles the corresponding bit in the port output register Pn_OUT. (see Table 7-4). On a read access, this bit returns 0.

Function of the PCx and PSx bit fields

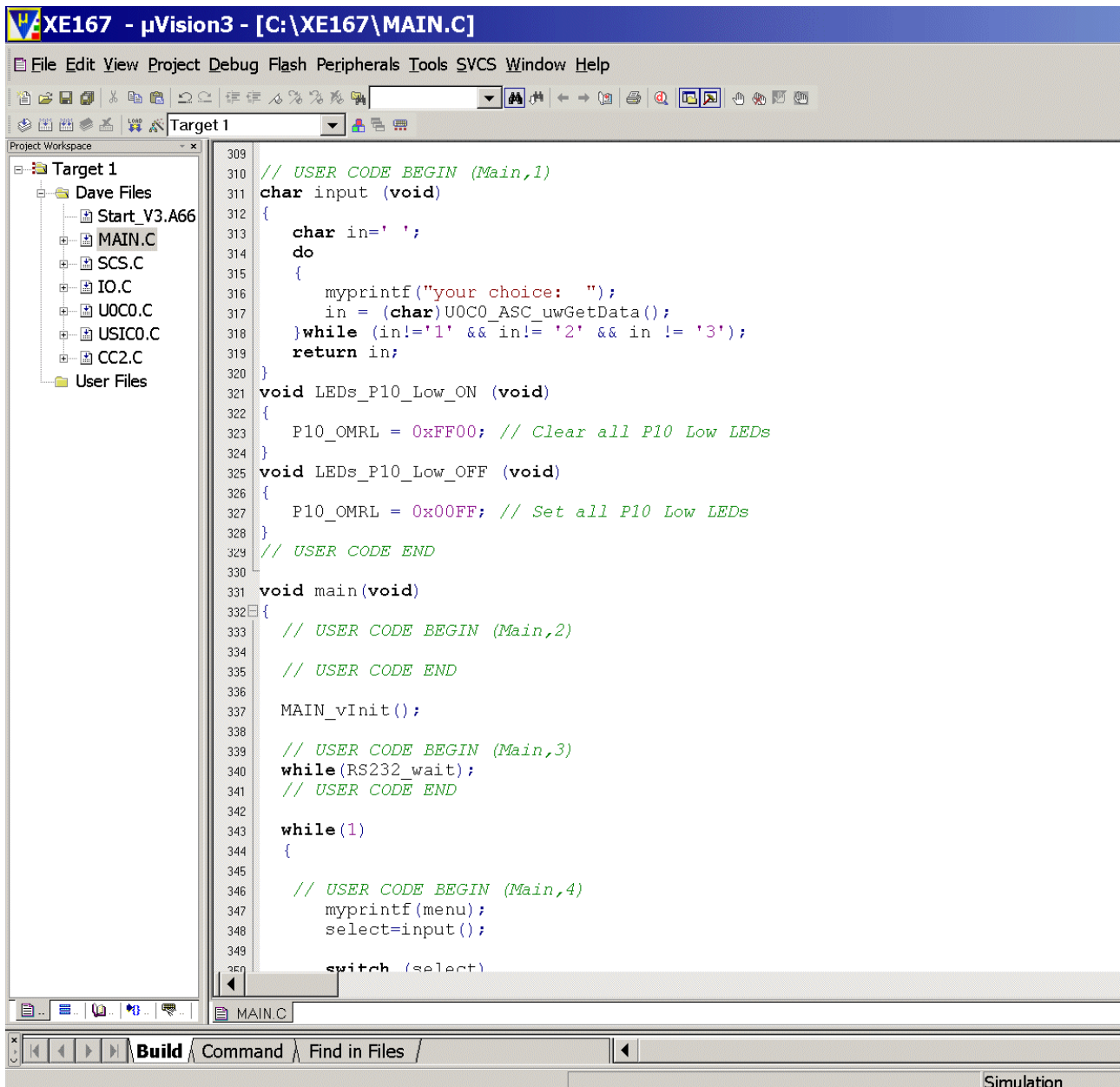
Table 7-4 Function of the Bits PCx and PSx

PCx	PSx	Function
0 or no write access	0 or no write access	Bit Pn_OUT.Px is not changed.
0 or no write access	1	Bit Pn_OUT.Px is set.
1	0 or no write access	Bit Pn_OUT.Px is cleared.
1	1	Bit Pn_OUT.Px is toggled.

Note: If a bit position is not written (one out of two bytes not targeted by a byte write), the corresponding value is considered as 0. Toggling a bit requires one 16-bit write.

Double click **MAIN.C** and insert the function **LEDs_P10_Low_OFF()**:

```
void LEDs_P10_Low_OFF (void)
{
    P10_OMRL = 0x00FF; // Set all P10 Low LEDs
}
```

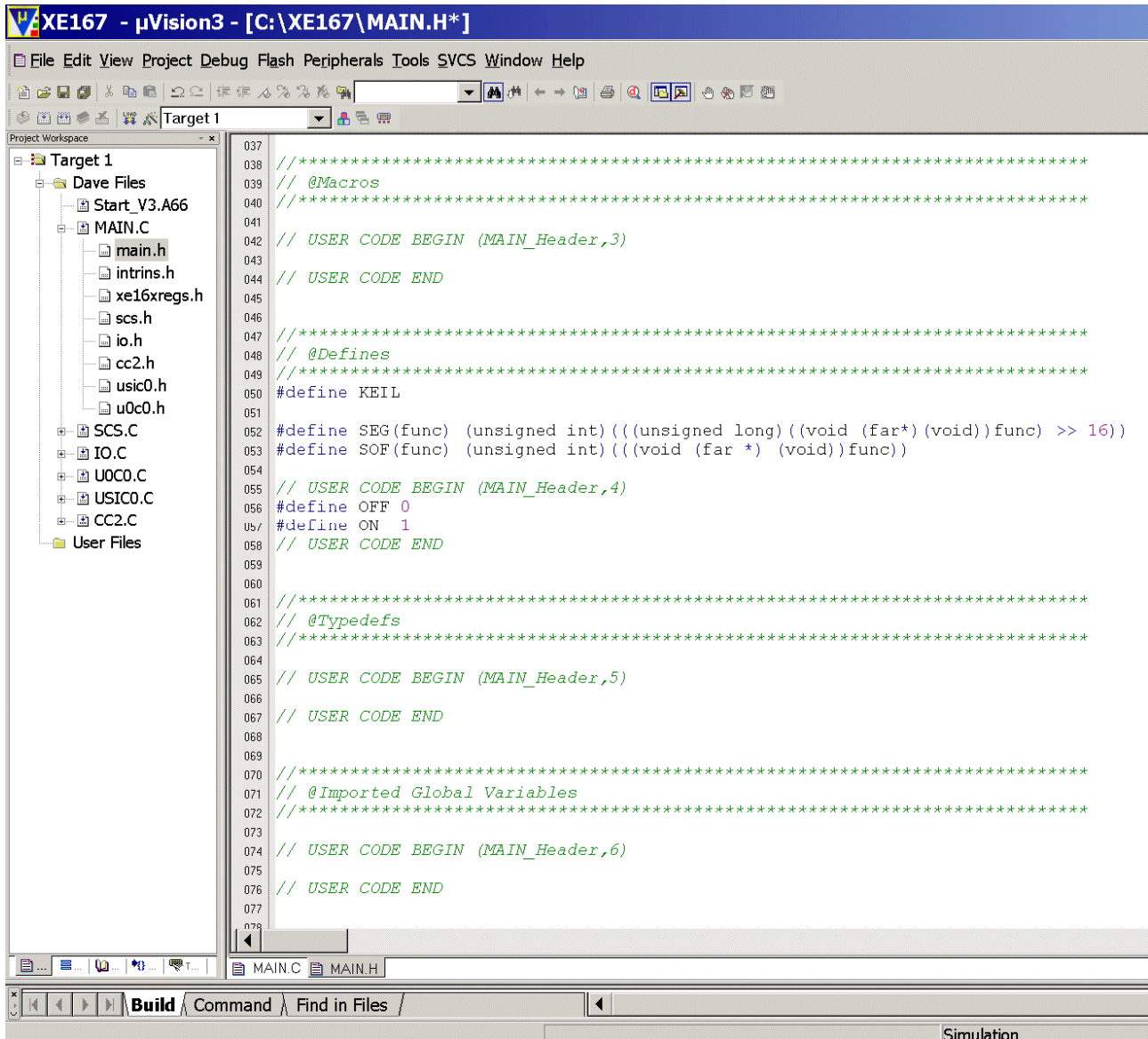


The screenshot shows the Infineon µVision3 IDE interface. The title bar reads "XE167 - µVision3 - [C:\XE167\MAIN.C]". The menu bar includes File, Edit, View, Project, Debug, Flash, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations, editing, and debugging. The Project Workspace on the left shows a tree view with "Target 1" expanded, containing "Dave Files" (Start_V3.A66, MAIN.C, SCS.C, IO.C, U0C0.C, USIC0.C, CC2.C) and "User Files". The main editor window displays the C code for MAIN.C, with line numbers 309 to 350 visible. The code includes comments for user code sections and the implementation of the LEDs_P10_Low_OFF function. The status bar at the bottom shows "Build", "Command", "Find in Files", and "Simulation".

```
309 // USER CODE BEGIN (Main,1)
310 char input (void)
311 {
312     char in=' ';
313     do
314     {
315         myprintf("your choice: ");
316         in = (char)U0C0_ASC_uwGetData();
317     }while (in!='1' && in!= '2' && in != '3');
318     return in;
319 }
320 void LEDs_P10_Low_ON (void)
321 {
322     P10_OMRL = 0xFF00; // Clear all P10 Low LEDs
323 }
324 void LEDs_P10_Low_OFF (void)
325 {
326     P10_OMRL = 0x00FF; // Set all P10 Low LEDs
327 }
328 // USER CODE END
329
330 void main(void)
331 {
332     // USER CODE BEGIN (Main,2)
333     // USER CODE END
334     MAIN_vInit();
335     // USER CODE BEGIN (Main,3)
336     while(RS232_wait);
337     // USER CODE END
338     while(1)
339     {
340         // USER CODE BEGIN (Main,4)
341         myprintf(menu);
342         select=input();
343         switch (select)
```

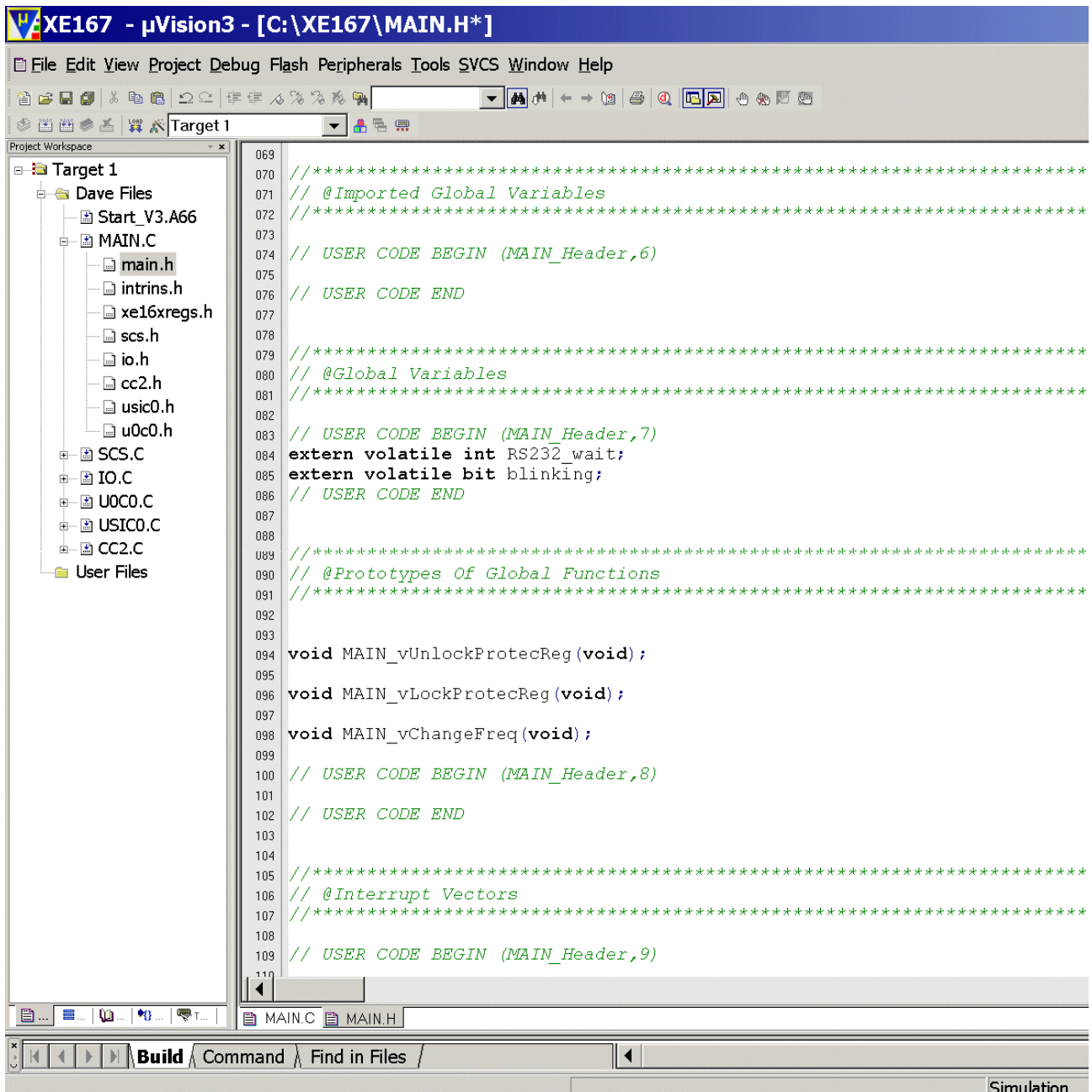
Double click **Main.h** and **insert** the following Defines:

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



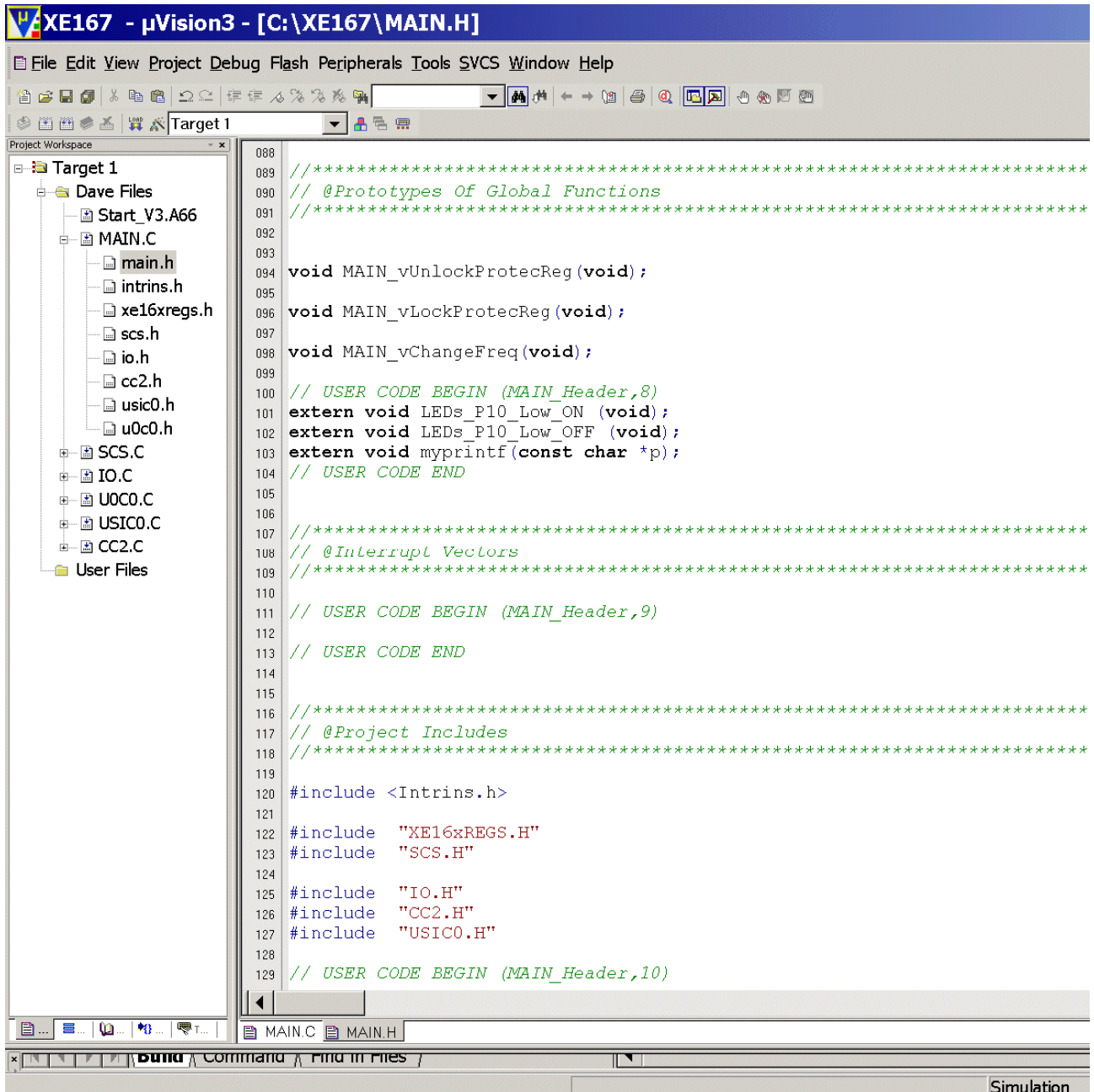
Double click **Main.h** and insert extern declarations "Global Variables":

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



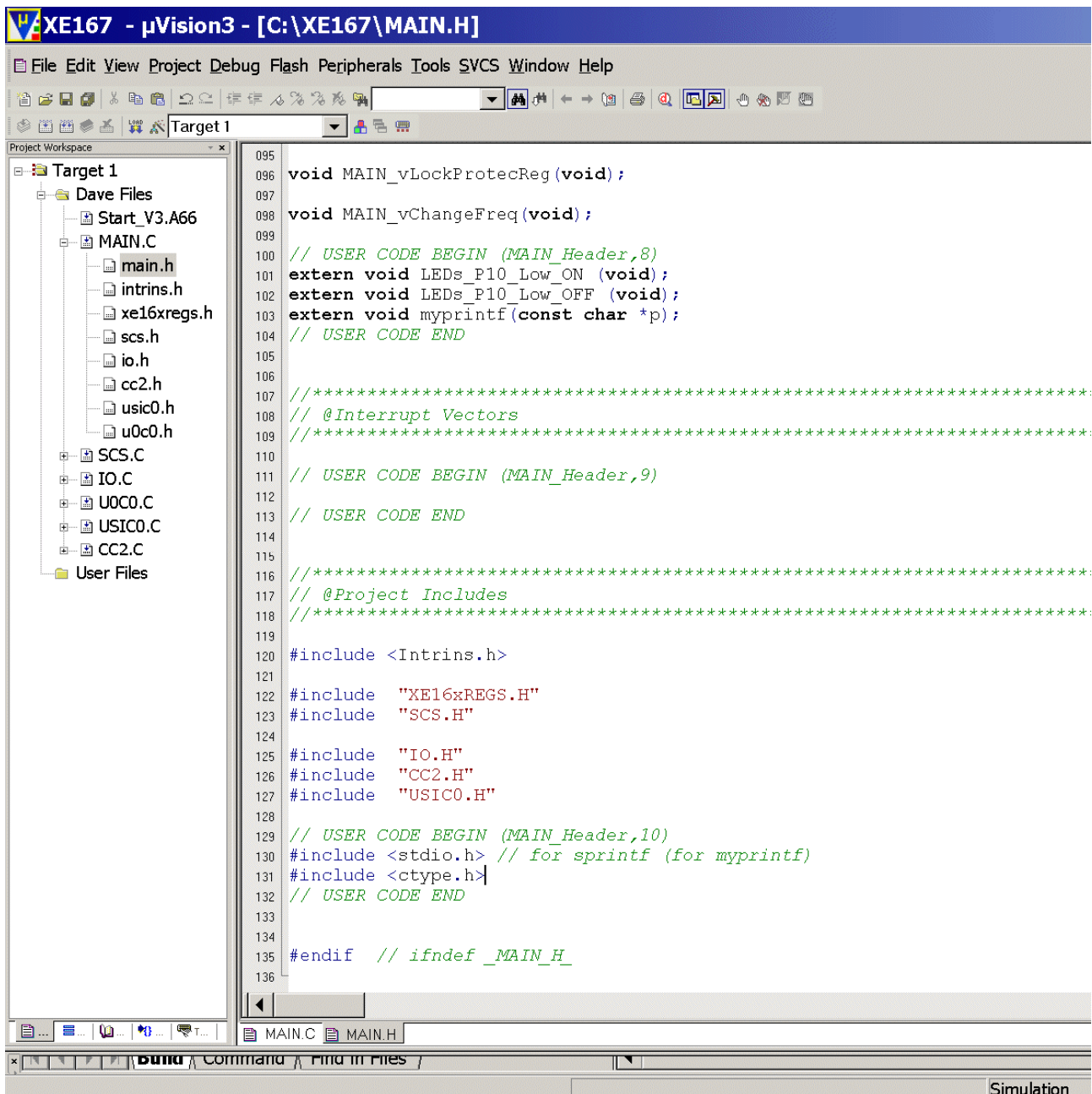
Double click **Main.h** and insert extern declarations "Global Functions":

```
extern void LEDs_P10_Low_ON (void);
extern void LEDs_P10_Low_OFF (void);
extern void myprintf(const char *p);
```

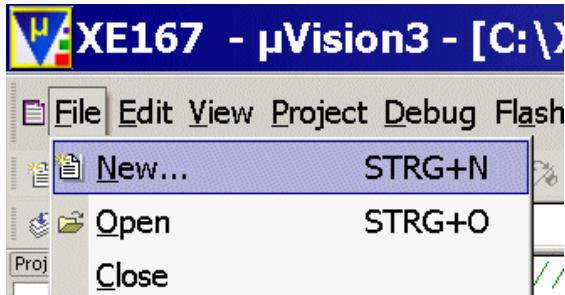


Double click **Main.h** and **insert** include files:

```
#include <stdio.h> // for sprintf (for myprintf)
#include <ctype.h>
```



File – New



Insert:

```
#include "main.h"

void myprintf(const char *p)
{
    while(*p)
    {
        U0C0_ASC_vSendData(*p++);
    }
}

/*

// Example 1 (use of myprintf):
// =====

void main(void)
{
    myprintf("Hello World!\r\n");
}

// Example 2 (use of myprintf):
// =====

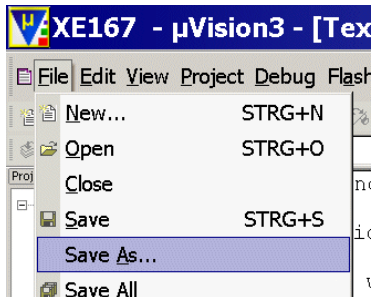
char mb[200]; // message buffer for sprintf()

void main(void)
{
    int dummy;

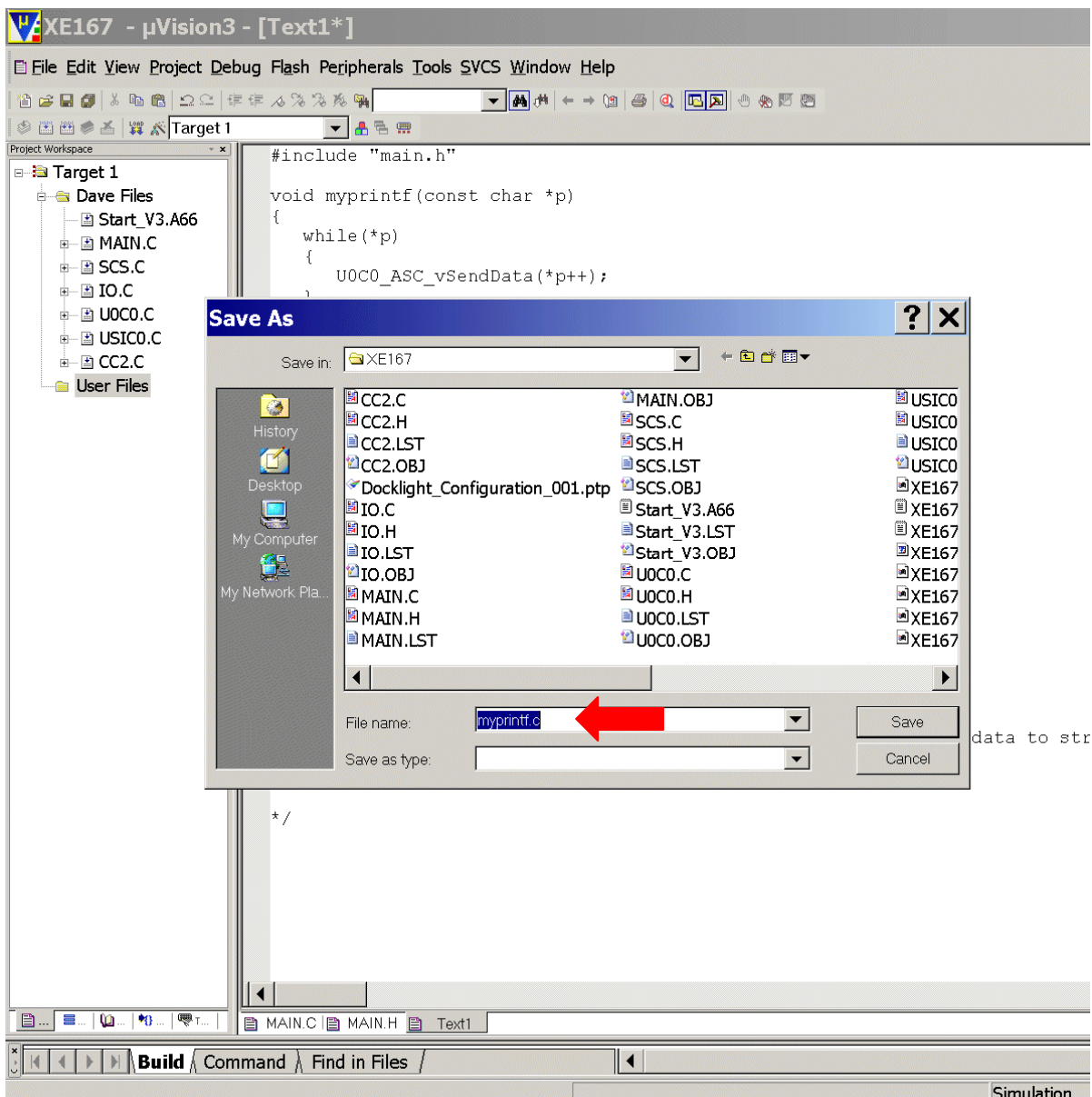
    sprintf(mb,"Variable dummy = %d",dummy); // Write formatted data to string mb
    myprintf(mb);
}

*/
```

File – Save As...

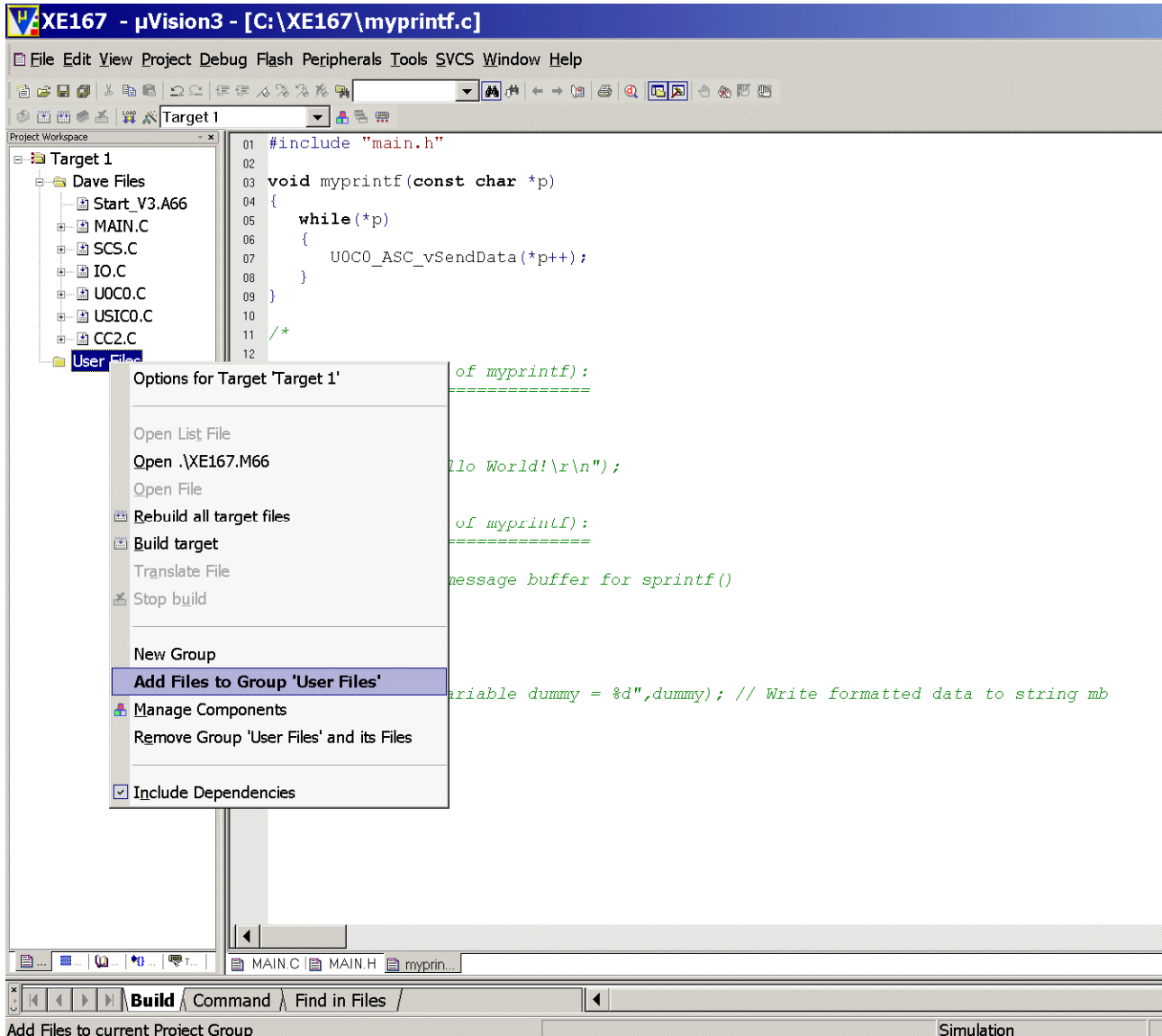


Insert: myprintf.c

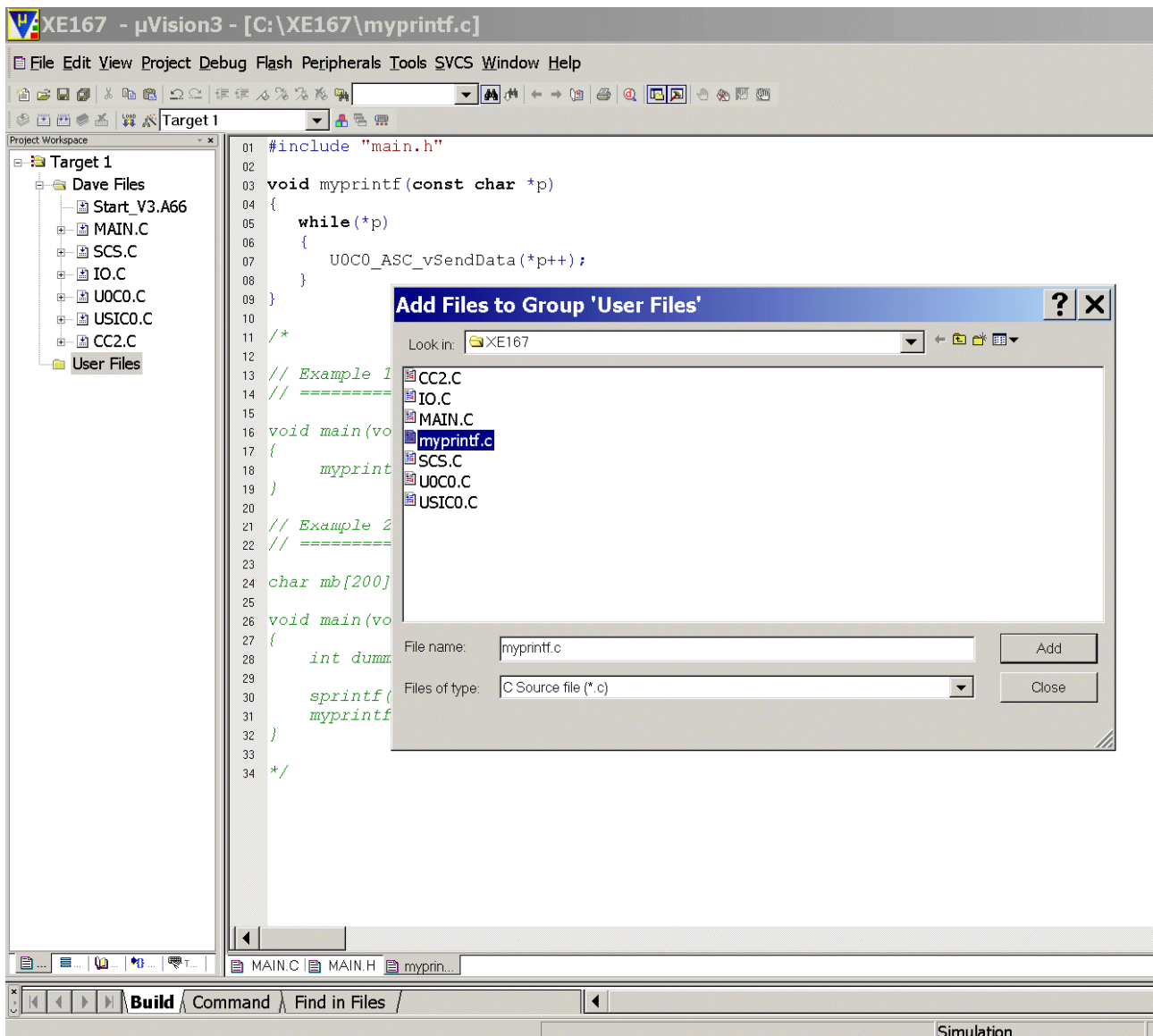


Save

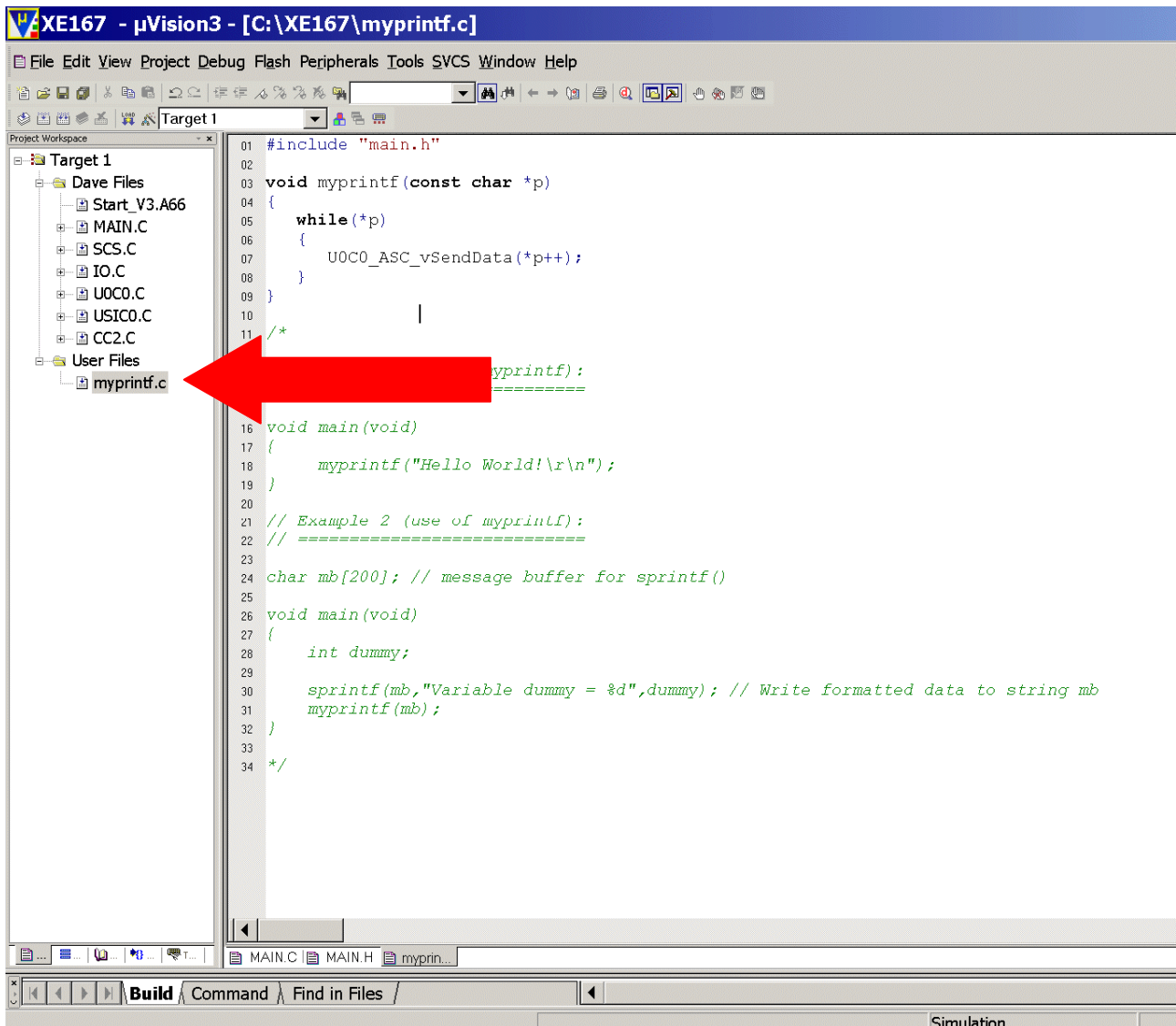
Mouse position: **Project Window**, User Files: **click right mouse button**
click Add Files to Group 'User Files'



Click myprintf.c



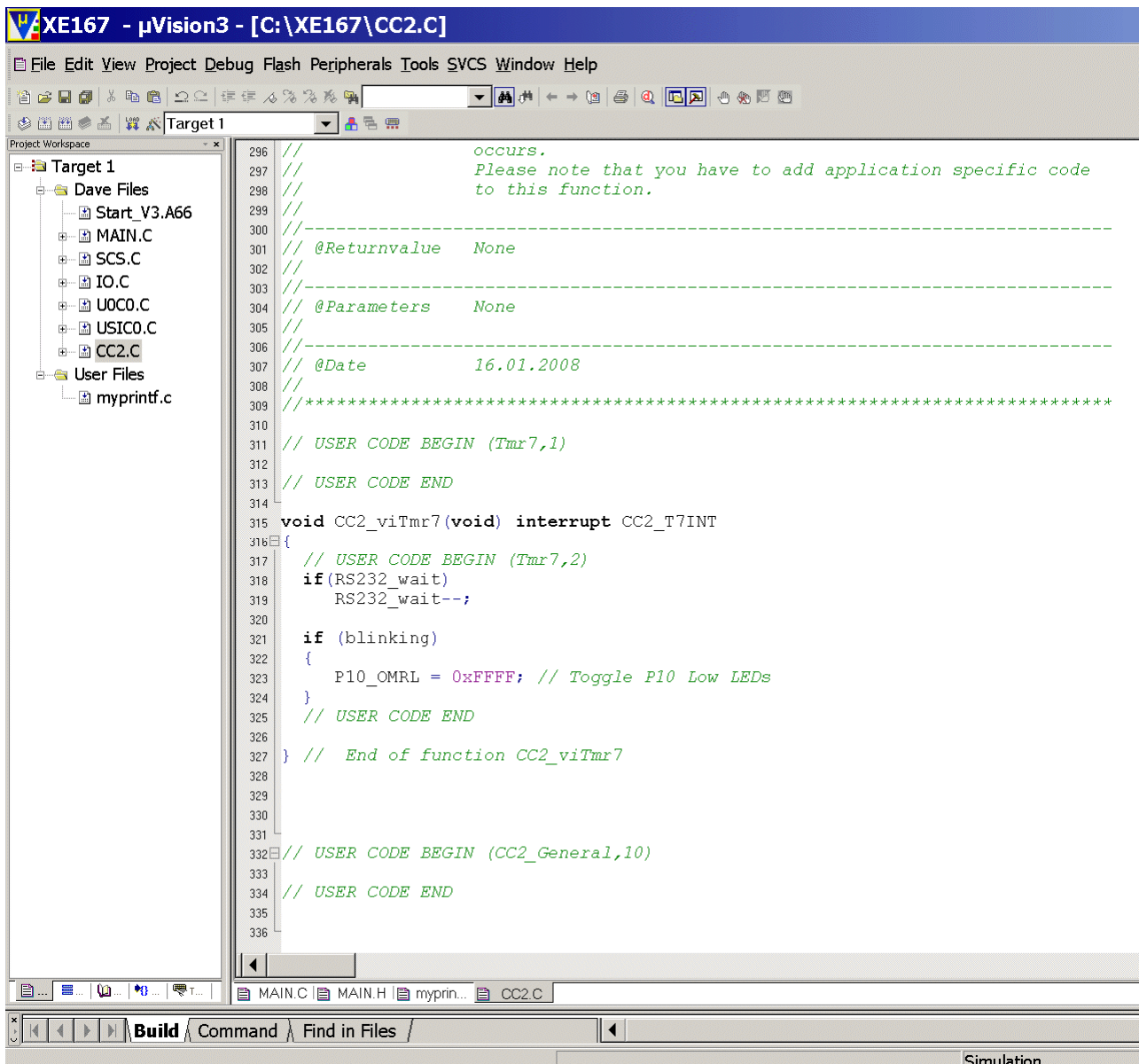
Add
Close



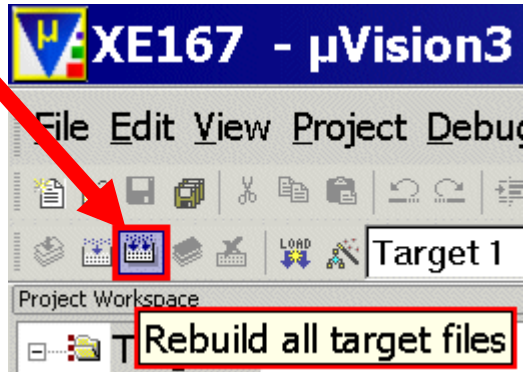
Double click **CC2.C** insert Code (CAPCOM 2 Timer 7 Interrupt Service Routine):

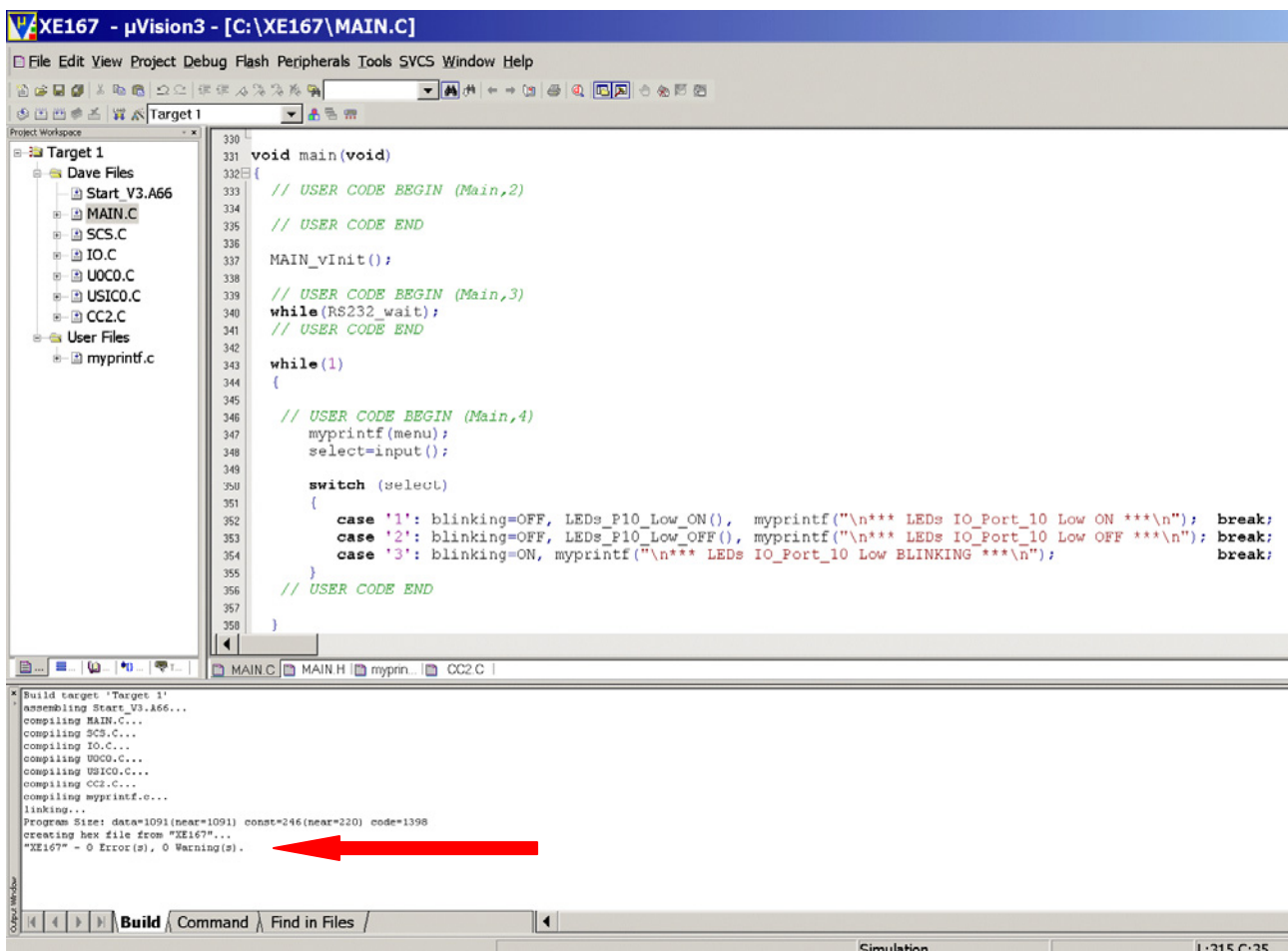
```
if(RS232_wait)
    RS232_wait--;

if (blinking)
{
    P10_OMRL = 0xFFFF; // Toggle P10 Low LEDs
}
```



Generate your application program:

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



Now we close our project and μ Vision 3:

Project - Close Project

File
Exit



Note:

Programming is now complete.

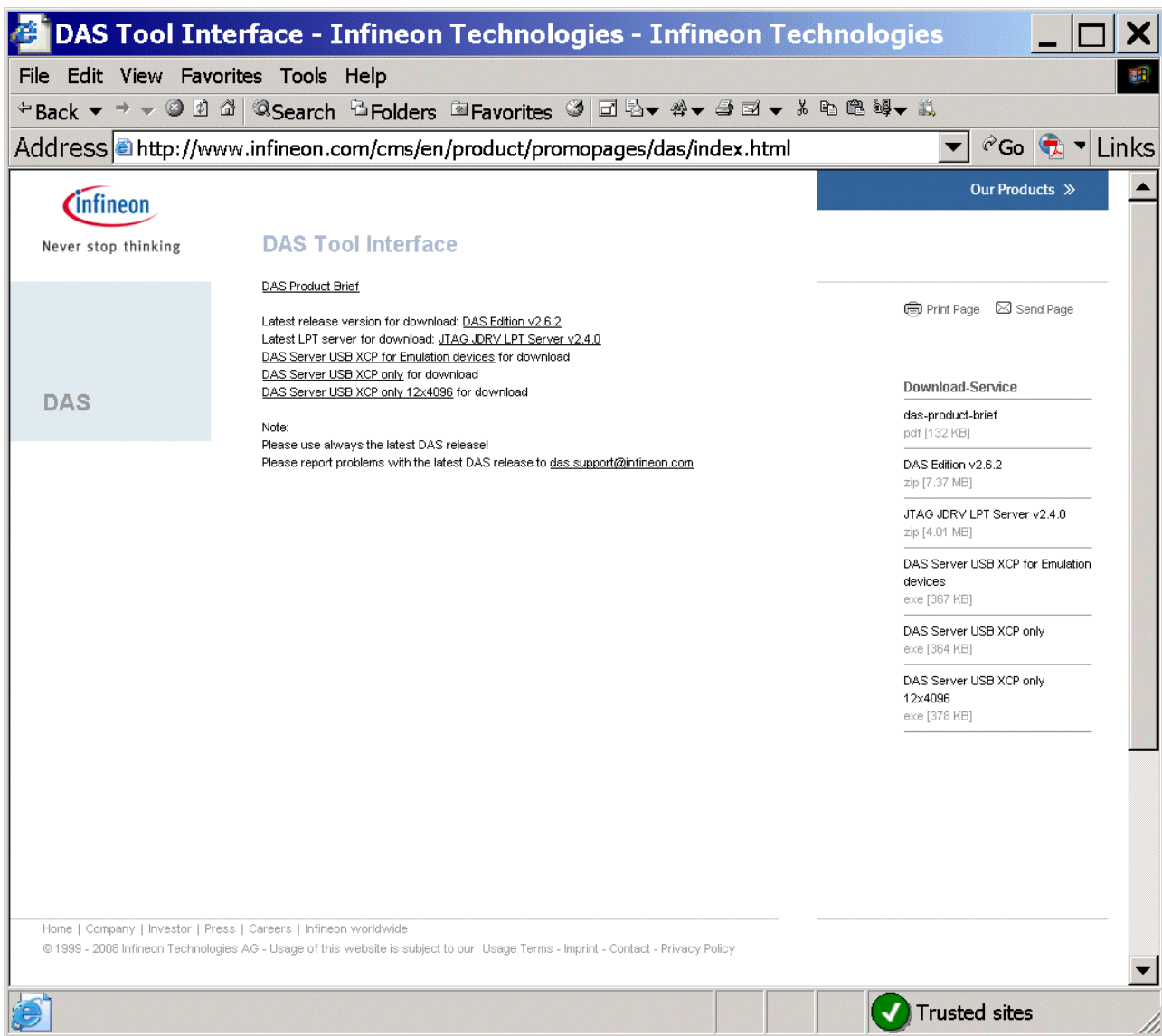
Unfortunately it is not possible to test your program with the Keil Simulator because this feature is currently not supported.

Therefore we are going to **load** (On Chip Flash Programming) and **run** your program on the Easy Kit Board now.

5.) Running your first programming example Using real hardware (+ OnChipFlash-Programming):

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!

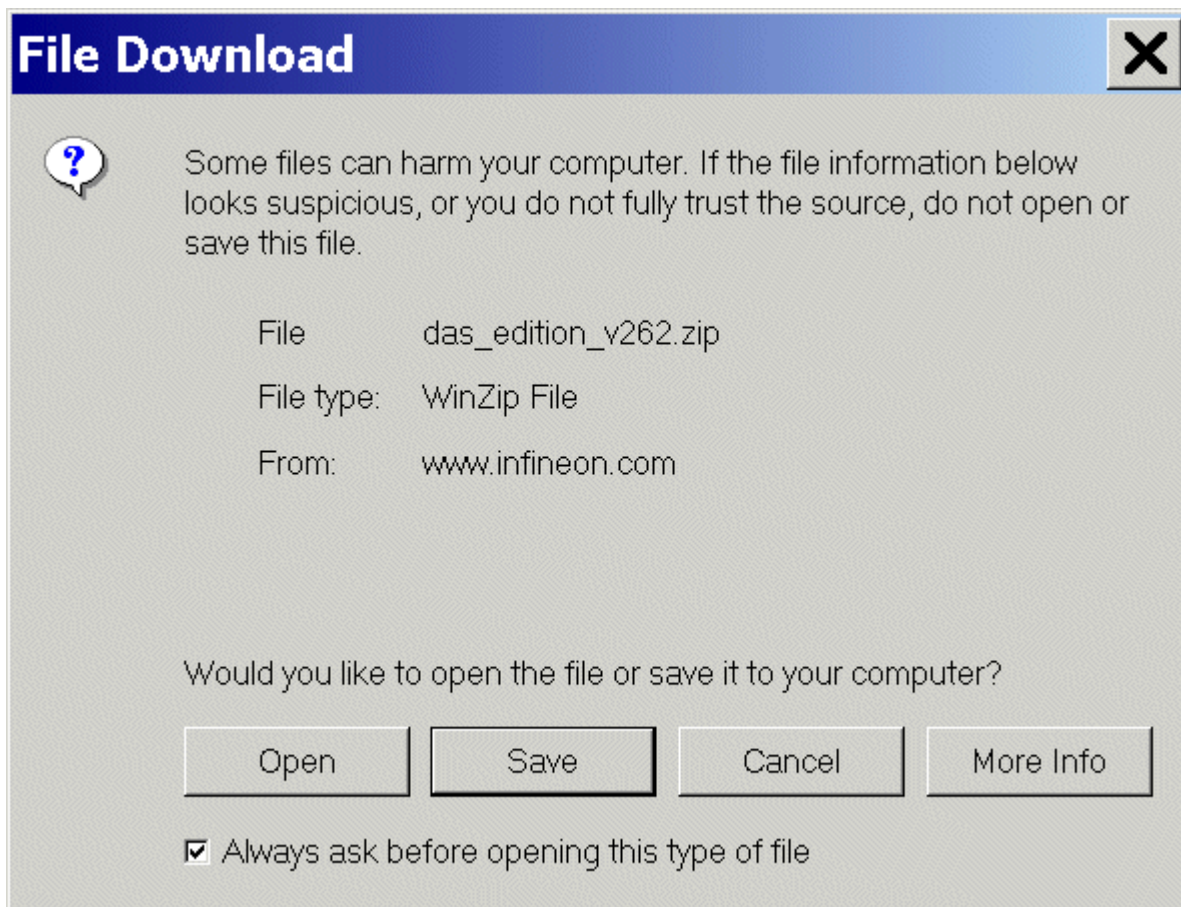
The goal of the DAS software is to provide one single interface for all types of tools.

The USB Device driver communicates with the XE167 Easy Kit Board when connected to the host computer.

The USB Device driver for the XE167 Easy Kit USB interface is included in the DAS software.

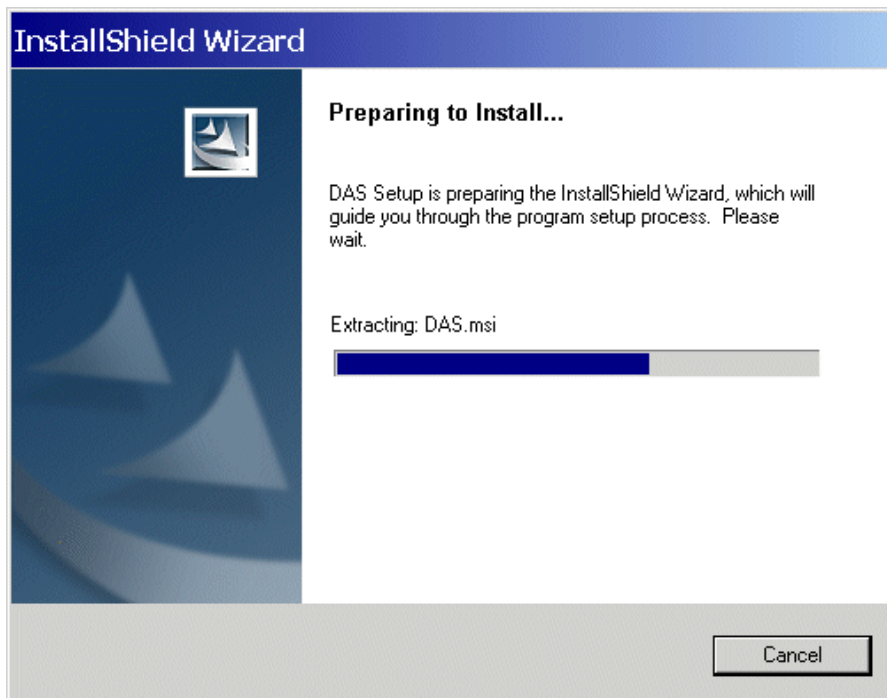
A virtual COM port driver is also included.

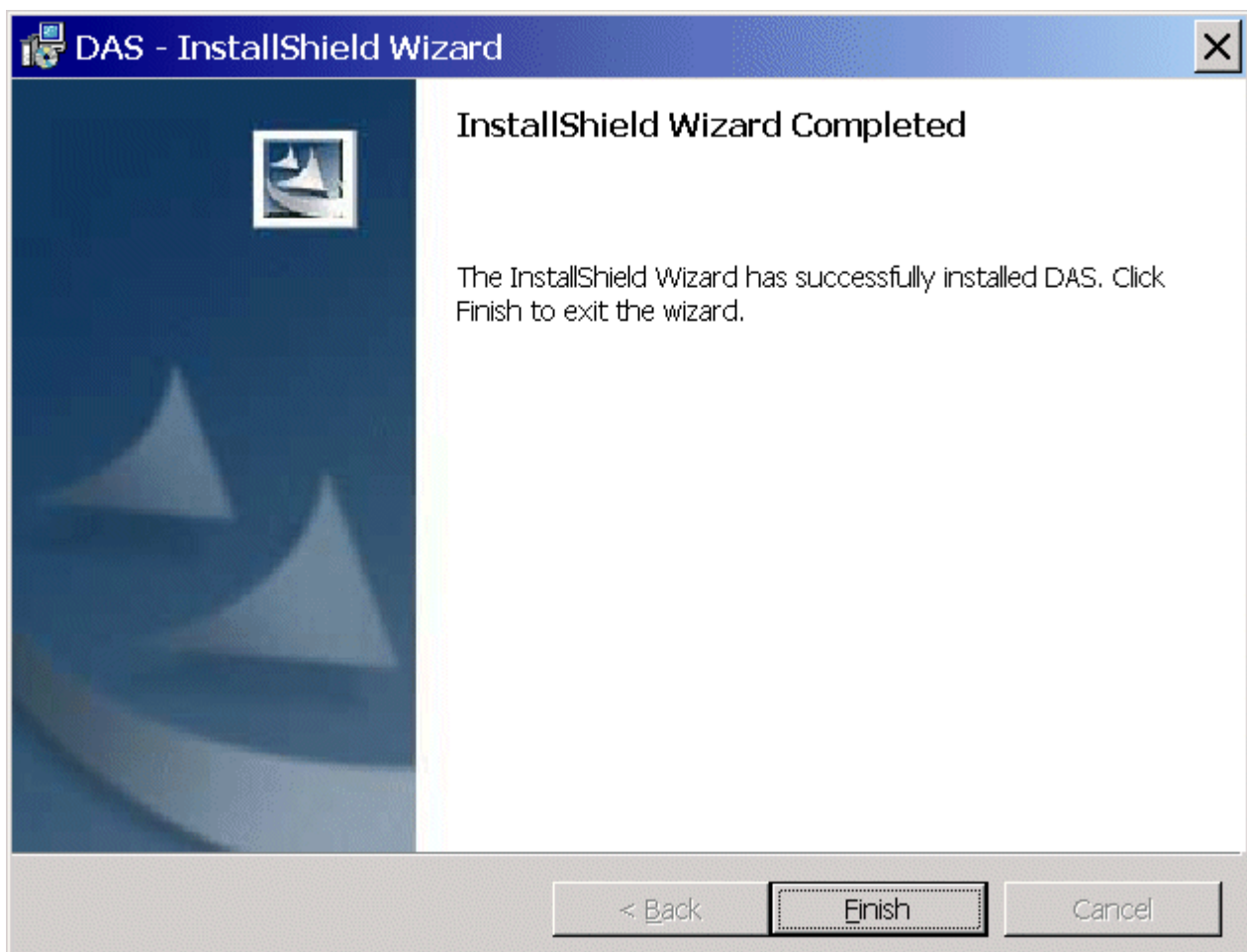
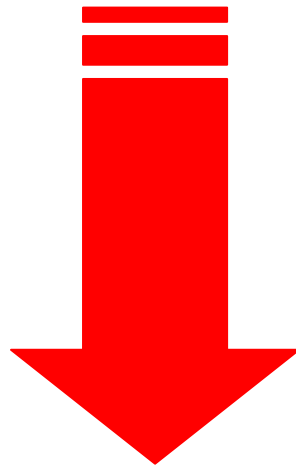
Download "The latest release version for download: DAS Edition v2.6.2":



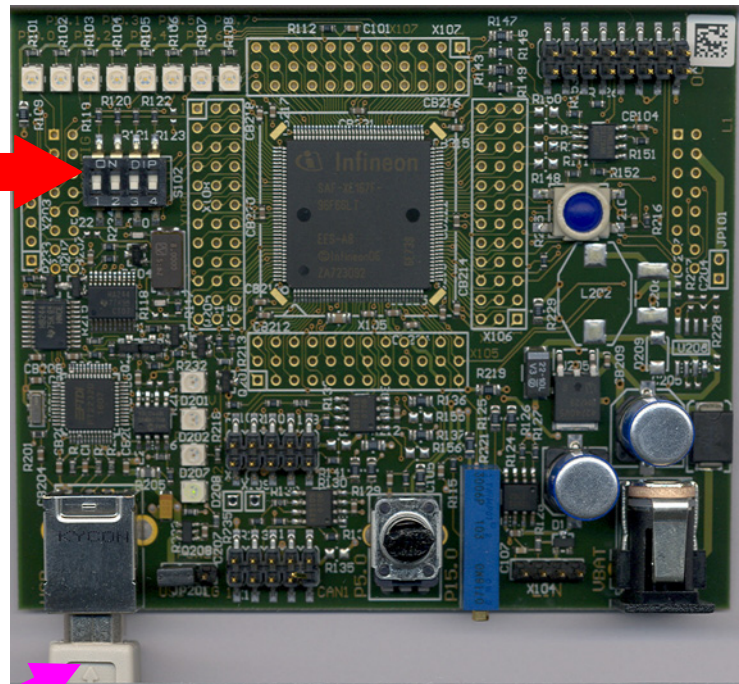
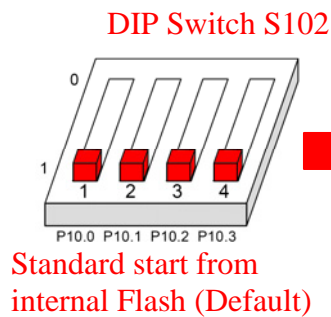
Unzip [das_edition_v262.zip](#) and

execute "DAS_v262_setup.exe" to install the DAS Server.





Connect the XE167 Easy Kit Board to the host computer:



USB Cable:

.) used for: UART communication (the UART/RS232/ASC0/USIC0_CH0 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.

Found New Hardware



DAS JTAG over USB EK XE167



Note:

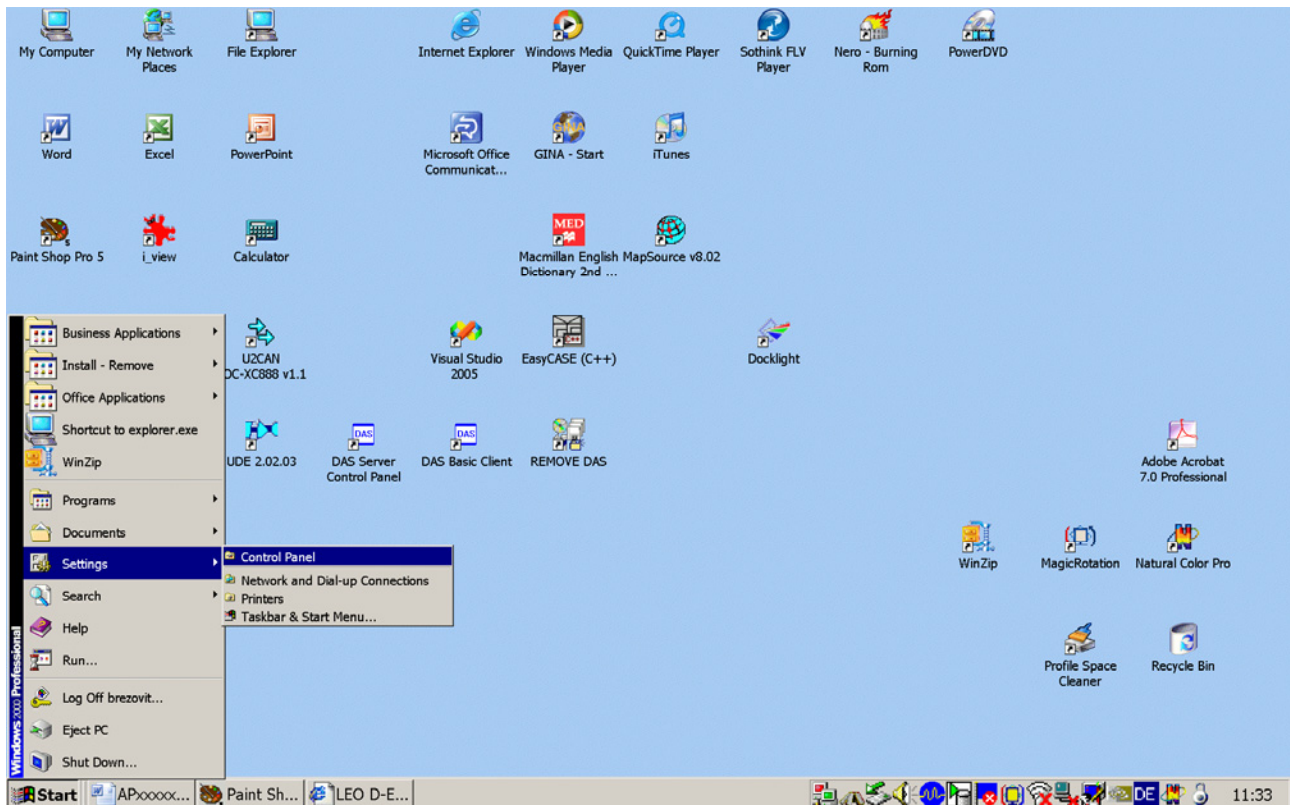
A USB driver is installed the first time while connecting the XC167 Easy Kit Board via the USB cable to your host computer.

Note:

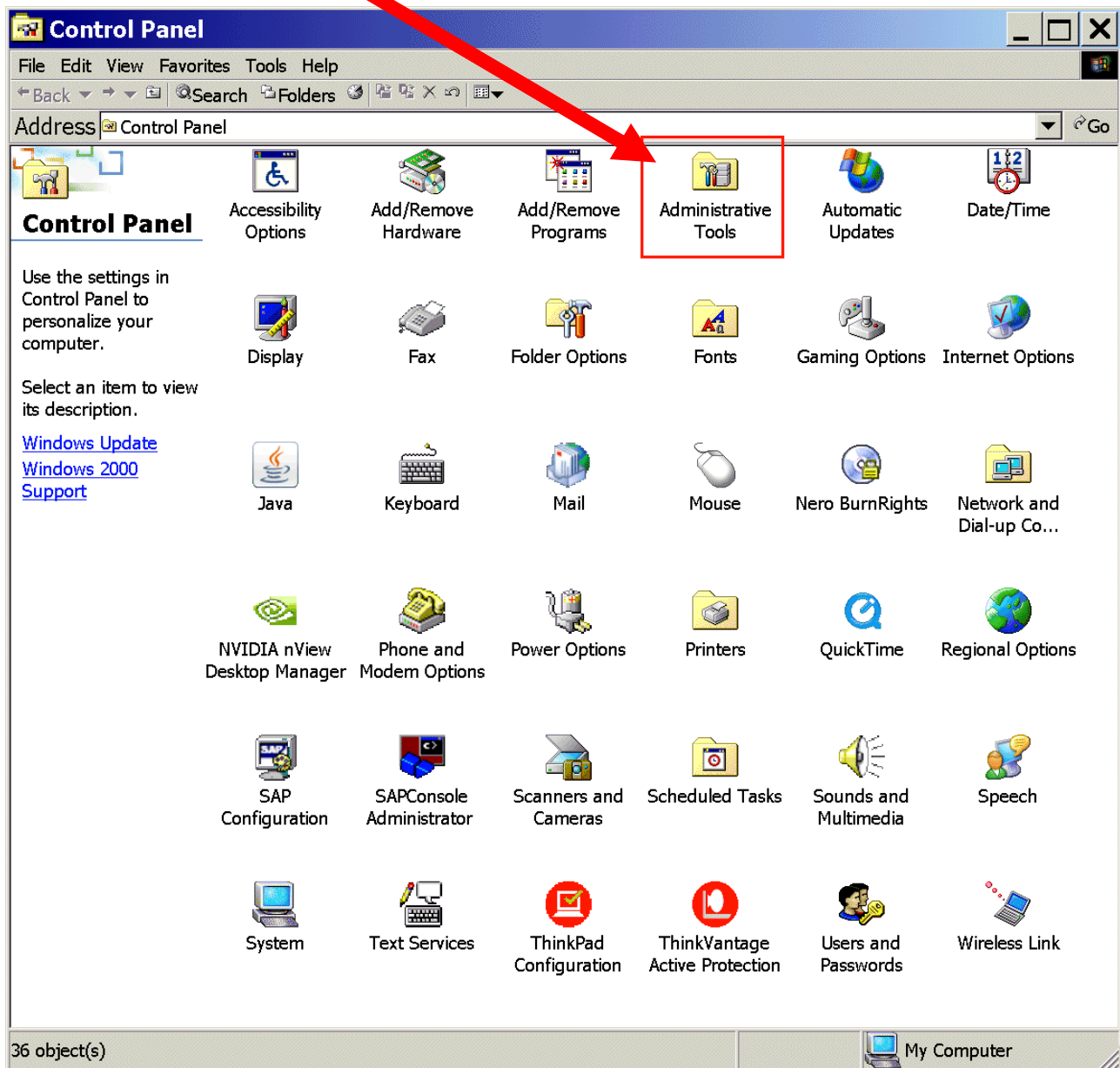
A default virtual COM Port is generated after the USB driver is installed.

Using a Windows 2000 operating system, we are now going to search for the virtual COM Port which was generated after connecting our XE167 Easy Kit Board:

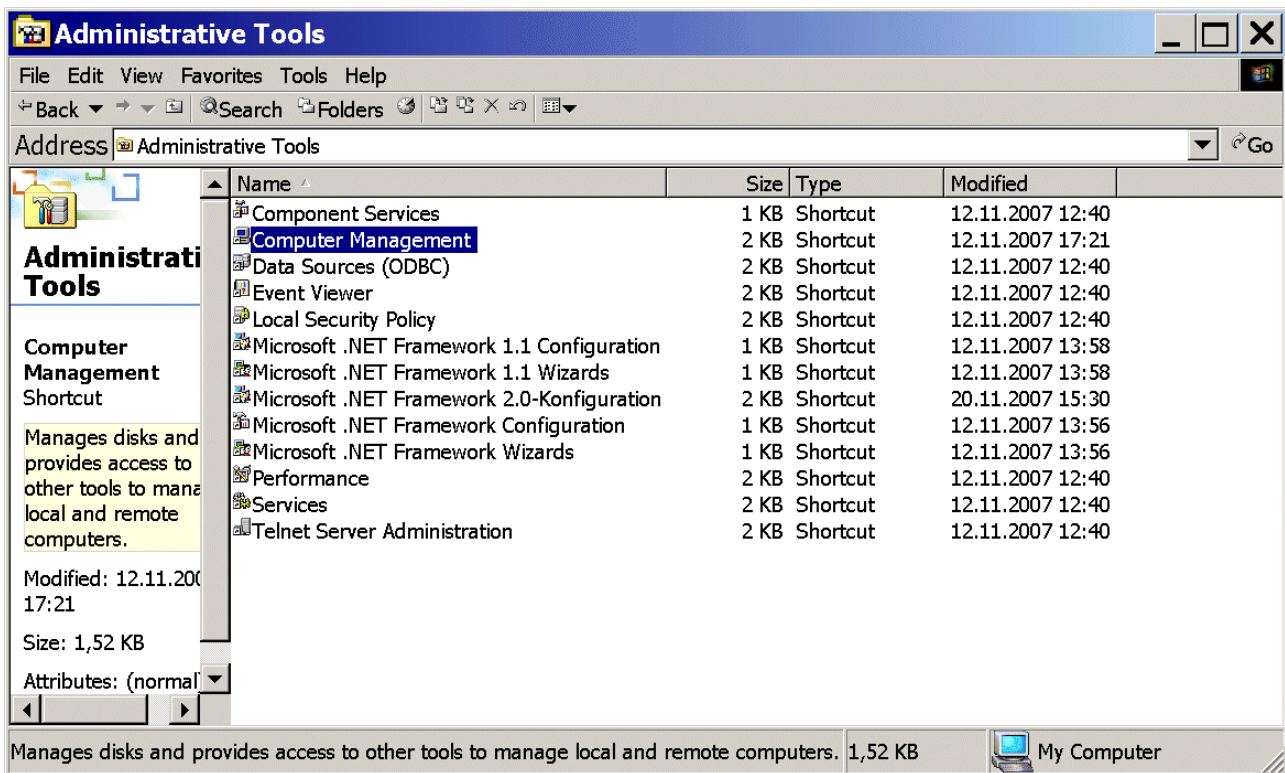
Start – Settings – Control Panel



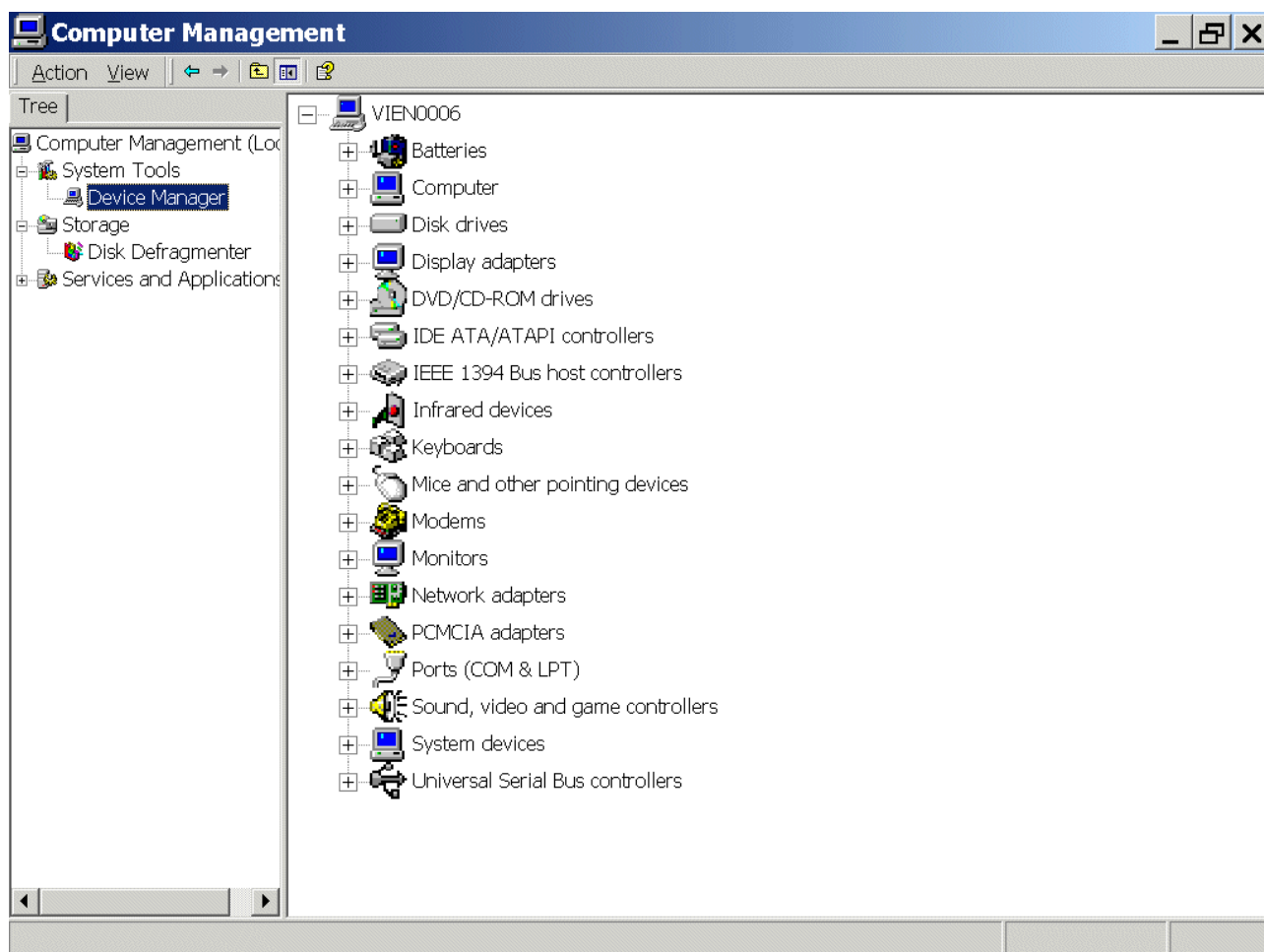
Double click: Administrative Tools



Double click: Computer Management

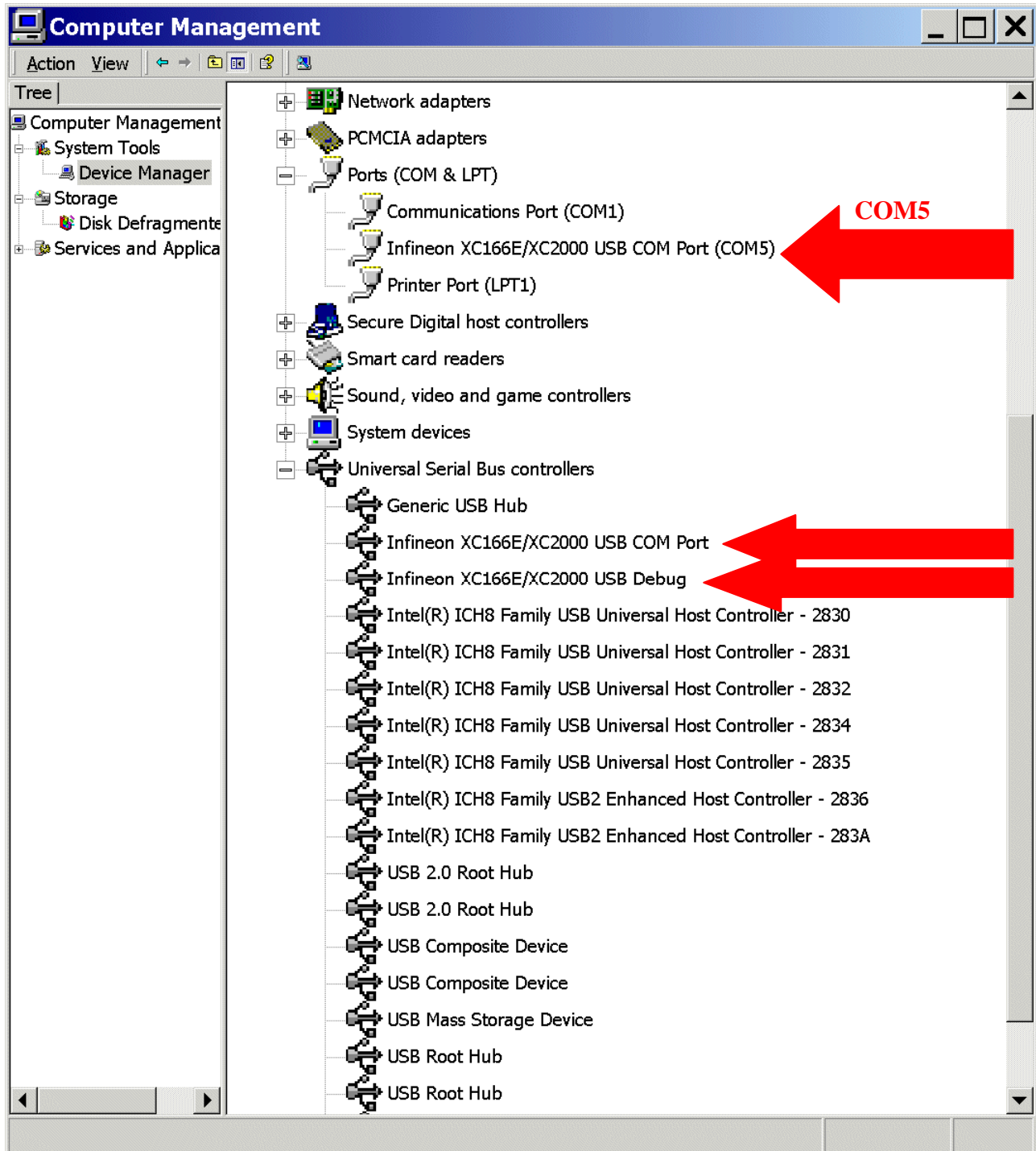


Click: Device Manager



Expand: Ports (COM & LPT):

Expand: Universal Serial Bus controllers:



Note:

As we can see in the screenshot above:
our virtual COM Port for UART/RS232 communication with the XE167 Easy Kit Board via USB is
COM5!



Start Keil μ Vision and open our Keil Project

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

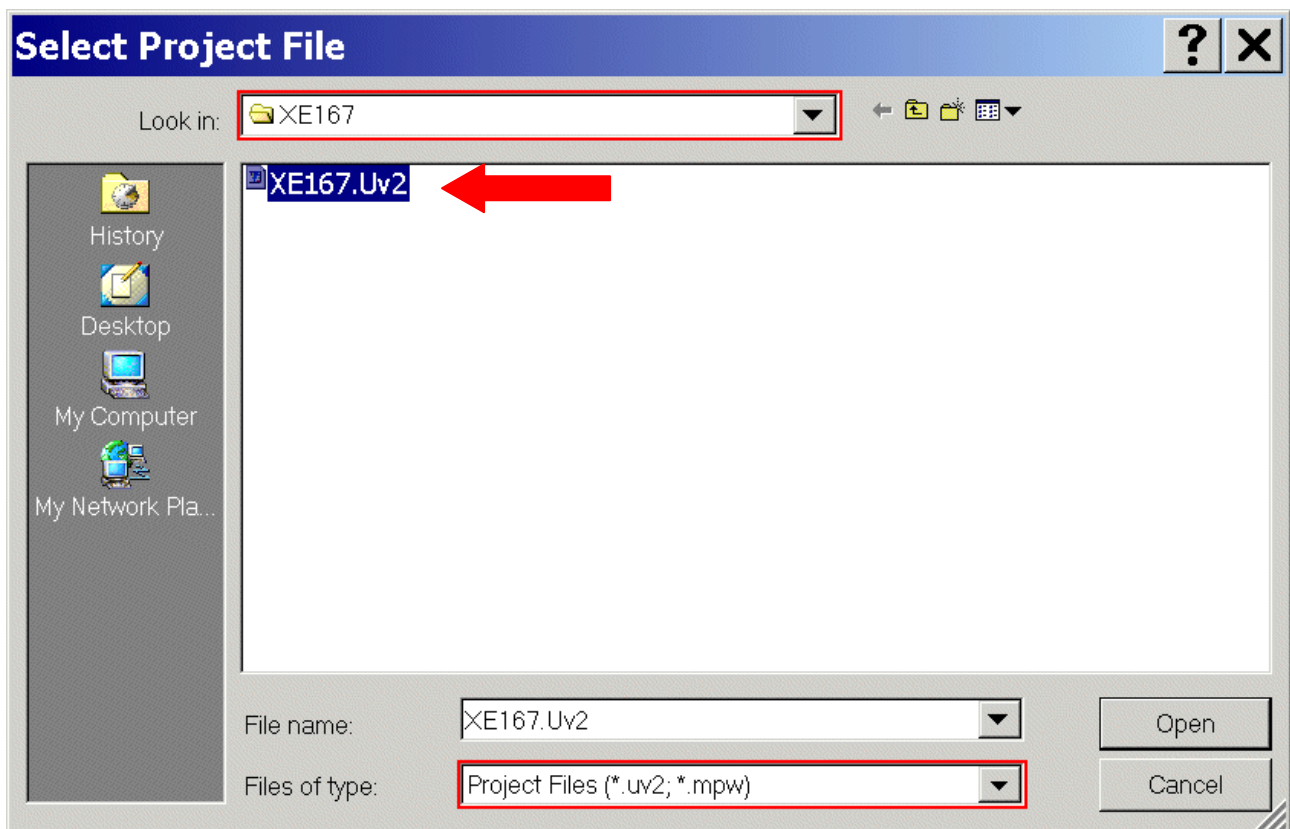
Project - Open Project

Select Project File: **Look in:** choose C:\XE167

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

Click XE167.Uv2

Open

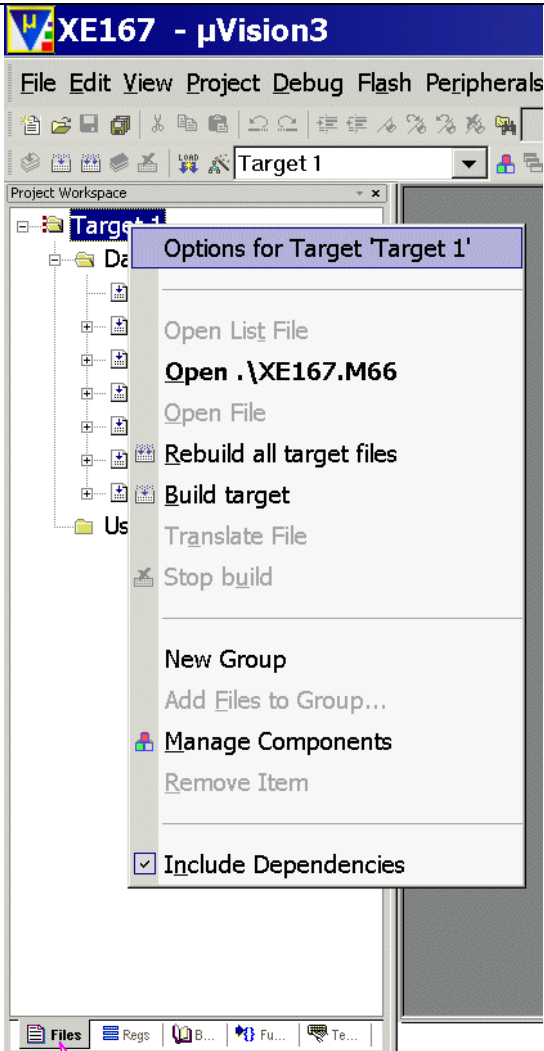
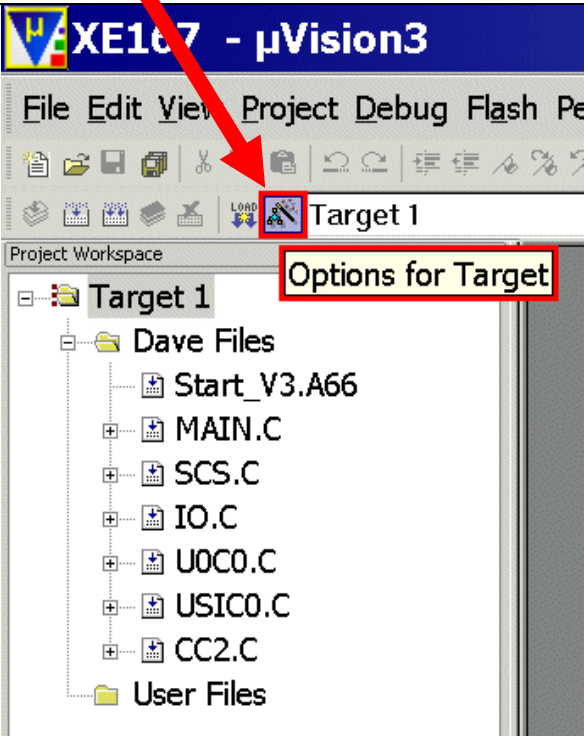


Note:

From now on just open your μ Vision project (not the DAVE project).

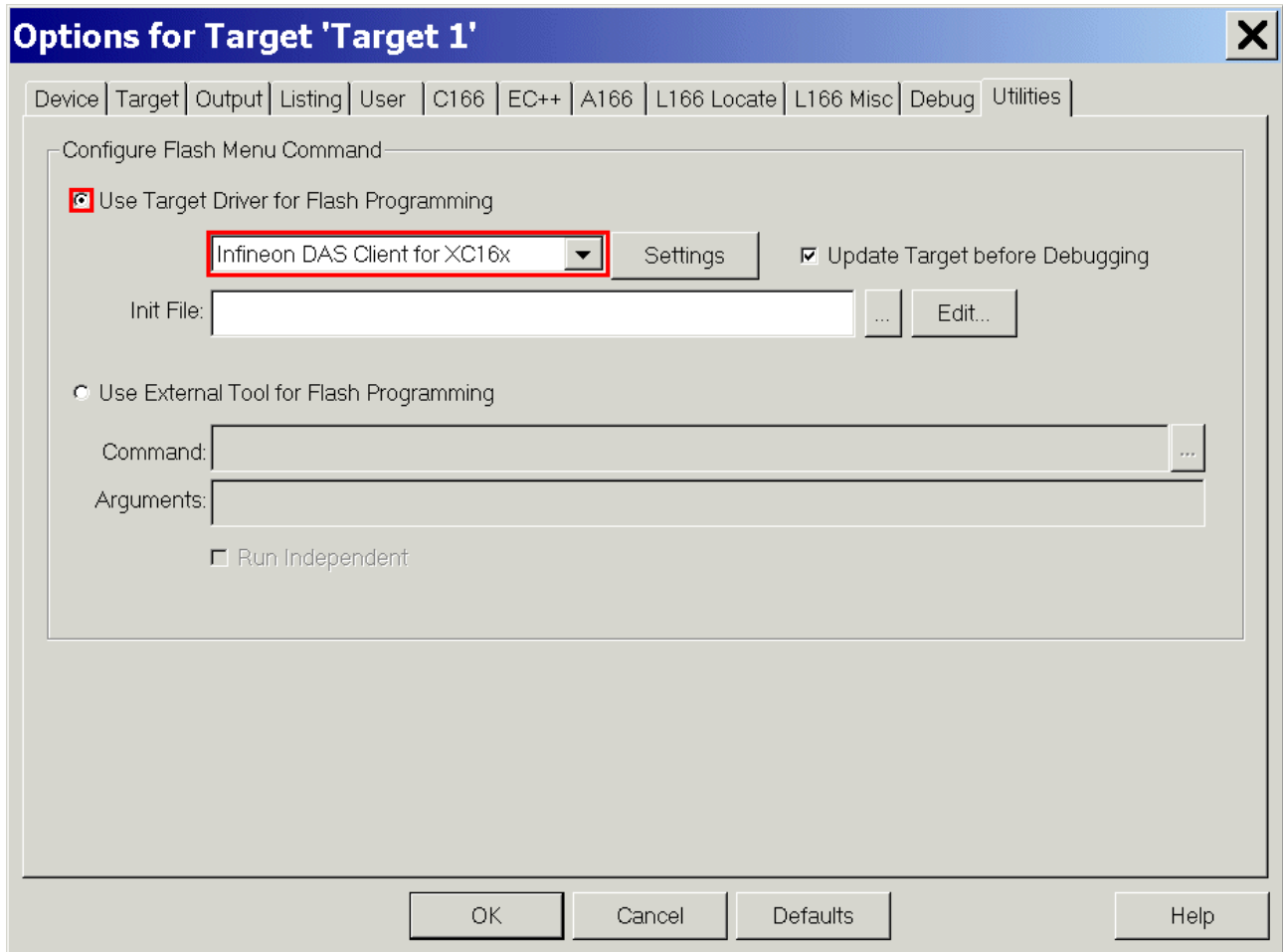
μ Vision will automatically recognise if there has been a code regeneration done by DAVE!

Configuration of the Flash Programming Utility:

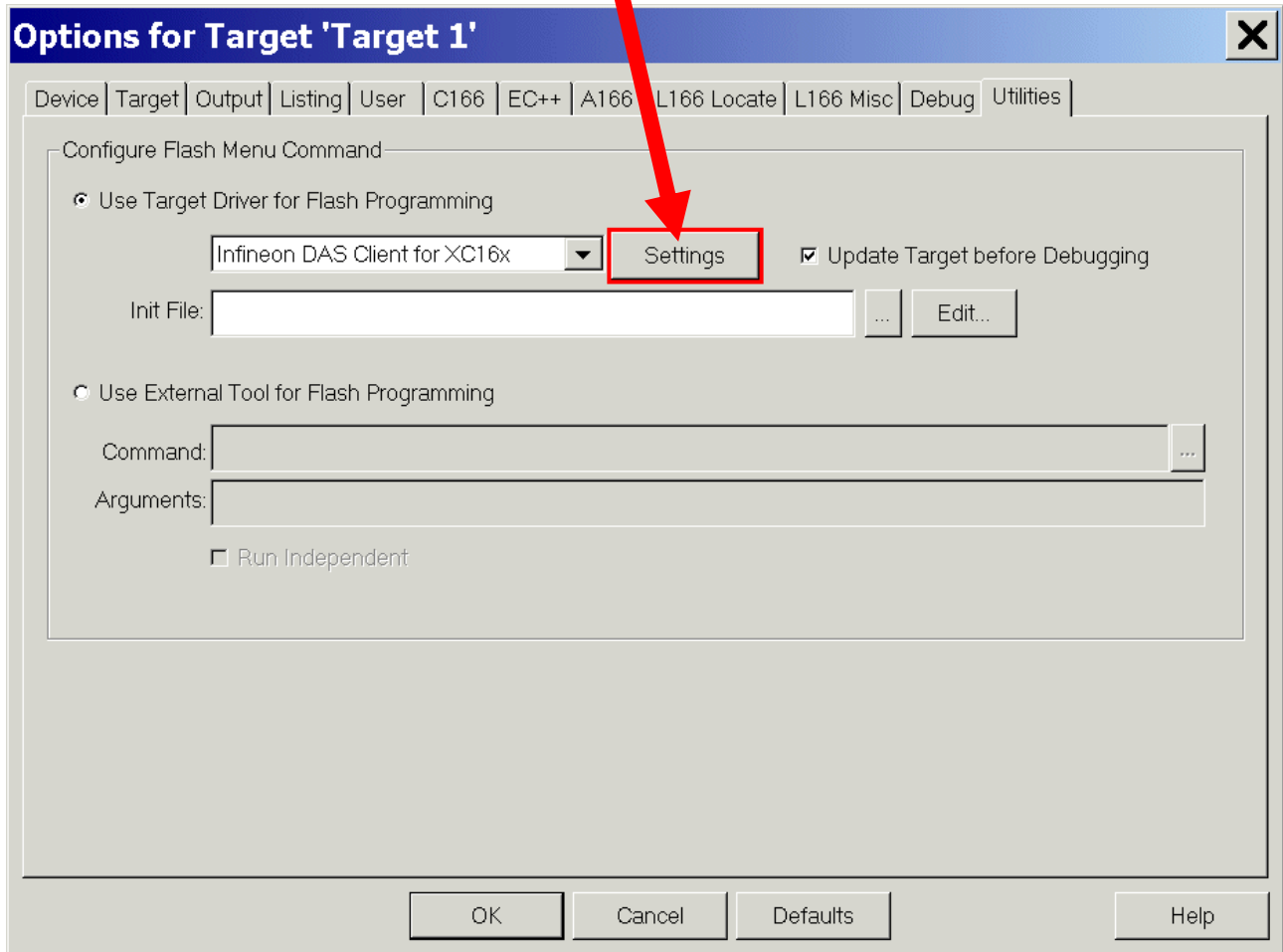
<p>mouse position: (Project Workspace, Files): Target1 click right mouse button click Options for Target 'Target1'</p>	<p>or</p>	<p>click</p>
		

Project Workspace, Files

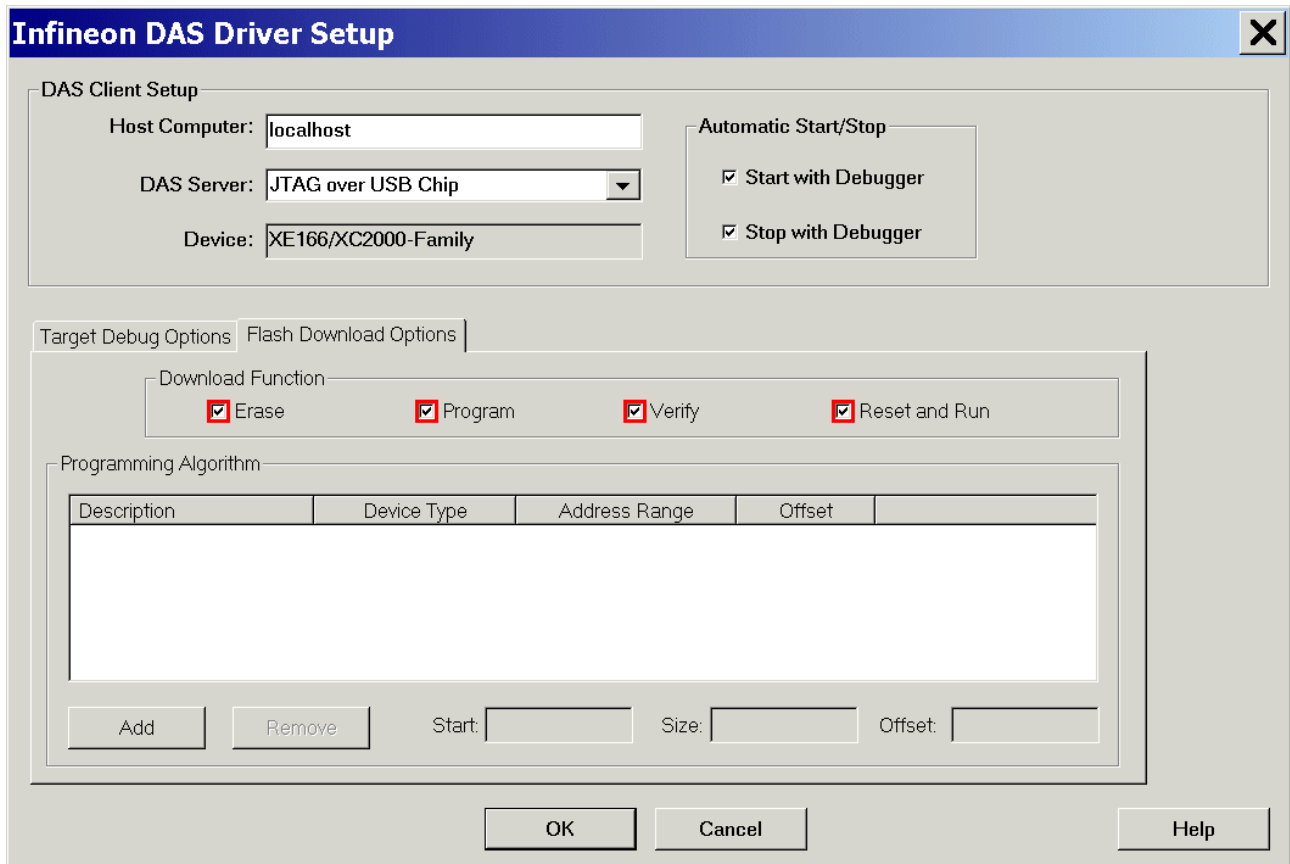
Utilities: Configure Flash Menu Command: **check** ☒ Use Target Driver for Flash Programming
Utilities: Configure Flash Menu Command: **select** Infineon DAS Client for XC16x



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



Flash Download Options: Download Function: check: ☒ Erase
Flash Download Options: Download Function: check: ☒ Program
Flash Download Options: Download Function: check: ☒ Verify
Flash Download Options: Download Function: check: ☒ Reset and Run



Infineon DAS Driver Setup

DAS Client Setup

Host Computer:

DAS Server:

Device:

Automatic Start/Stop

☒ Start with Debugger

☒ Stop with Debugger

Target Debug Options | Flash Download Options

Download Function

☒ Erase ☒ Program ☒ Verify ☒ Reset and Run

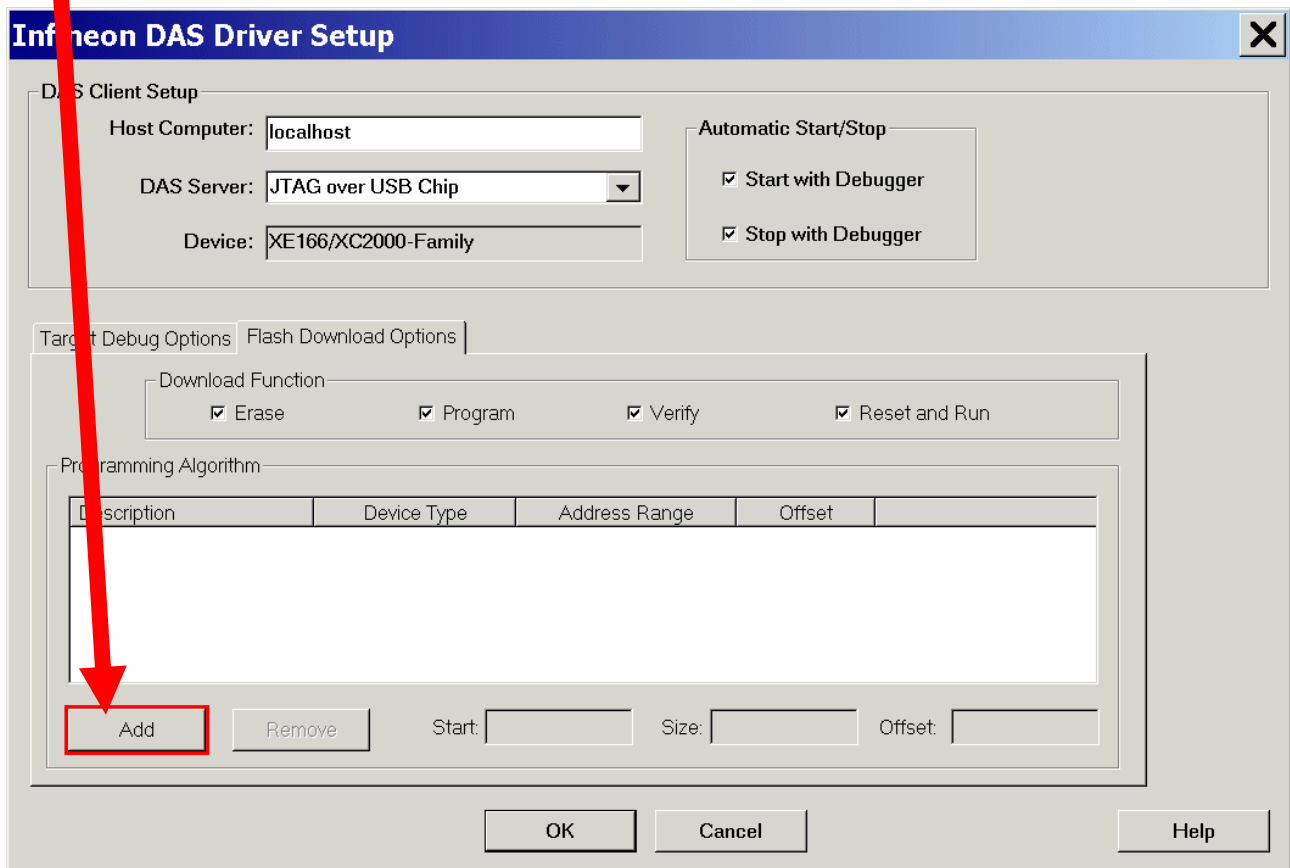
Programming Algorithm

Description	Device Type	Address Range	Offset

Add Remove Start: Size: Offset:

OK Cancel Help

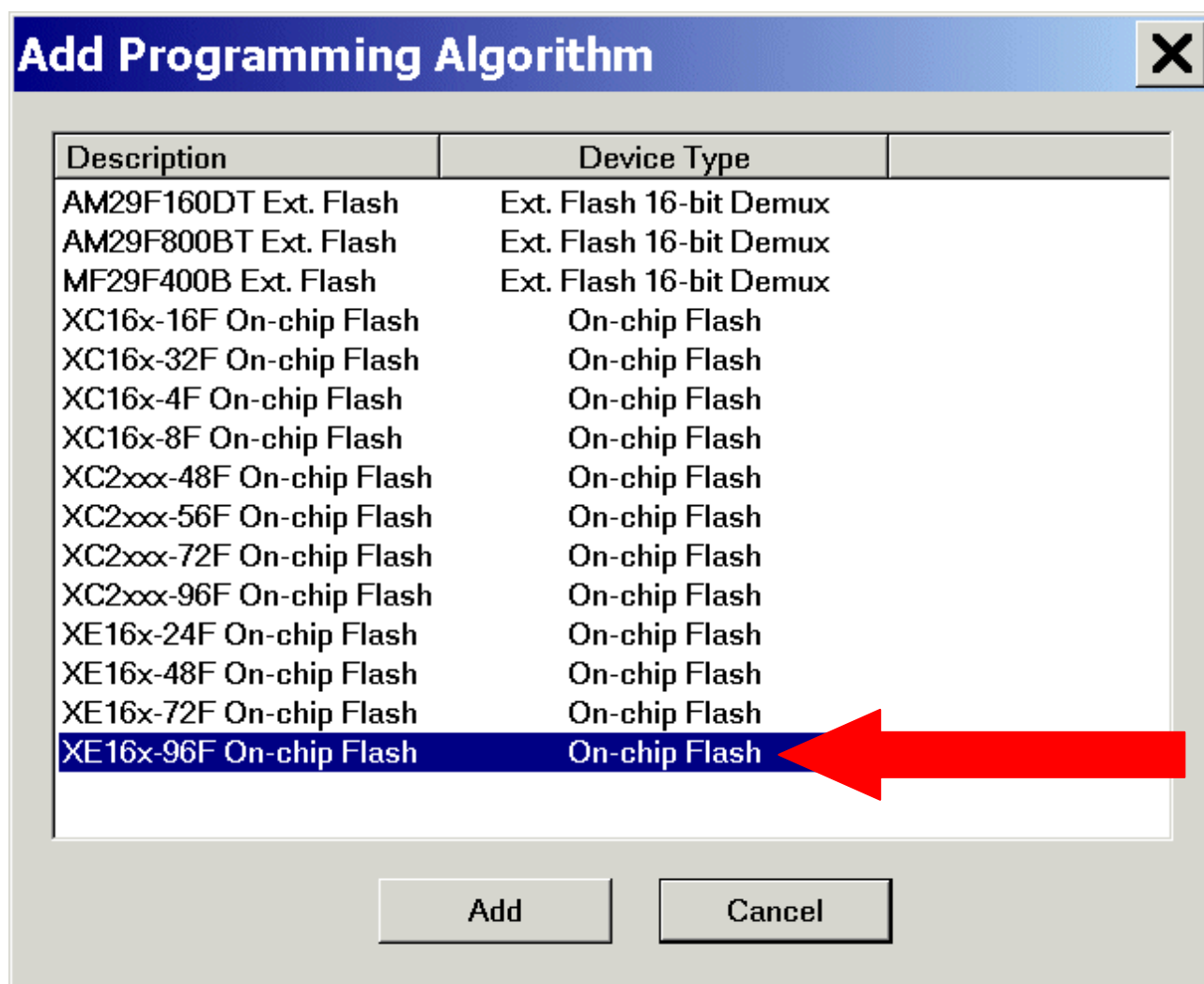
Click Add



The image shows the 'Infineon DAS Driver Setup' dialog box. A red arrow points to the 'Add' button in the 'Programming Algorithm' section. The dialog box has a title bar with 'Infineon DAS Driver Setup' and a close button. It contains several sections: 'DAS Client Setup' with fields for 'Host Computer' (localhost), 'DAS Server' (JTAG over USB Chip), and 'Device' (XE166/XC2000-Family); 'Automatic Start/Stop' with checkboxes for 'Start with Debugger' and 'Stop with Debugger'; 'Target Debug Options' and 'Flash Download Options' tabs; 'Download Function' with checkboxes for 'Erase', 'Program', 'Verify', and 'Reset and Run'; and 'Programming Algorithm' which includes a table with columns 'Description', 'Device Type', 'Address Range', and 'Offset'. Below the table are 'Add', 'Remove', 'Start', 'Size', and 'Offset' buttons. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

Description	Device Type	Address Range	Offset

Select: XE16x-96F On-chip Flash



Click Add

Infineon DAS Driver Setup
✕

DAS Client Setup

Host Computer:
DAS Server:
Device:

Automatic Start/Stop
☒ Start with Debugger
☒ Stop with Debugger

Target Debug Options

Flash Download Options

Download Function

☒ Erase
☒ Program
☒ Verify
☒ Reset and Run

Programming Algorithm

Description	Device Type	Address Range	Offset
XE16x-96F On-chip Flash	On-chip Flash	C00000H - CBFFFFH	000000H

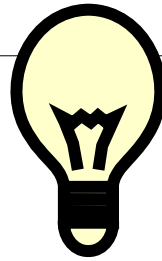
Start:
Size:
Offset:

OK

Cancel

Help

OK
OK



Note:

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



The screenshot shows a web browser window titled "Docklight - Download - Infineon Technologies". The address bar shows the URL http://www.docklight.de/download_en.htm. The main content area is titled "Software Archive" and contains two sections:

Latest Releases

Docklight V1.7	Download Docklight V1.7.37 for Windows Vista, Windows XP, Windows 2000, Windows NT, Windows ME, Windows 98 (3143 KB)
Docklight Scripting V1.7	Download Docklight Scripting V1.7.37 for Windows Vista, Windows XP, Windows 2000, Windows NT, Windows ME, Windows 98 (4314 KB)

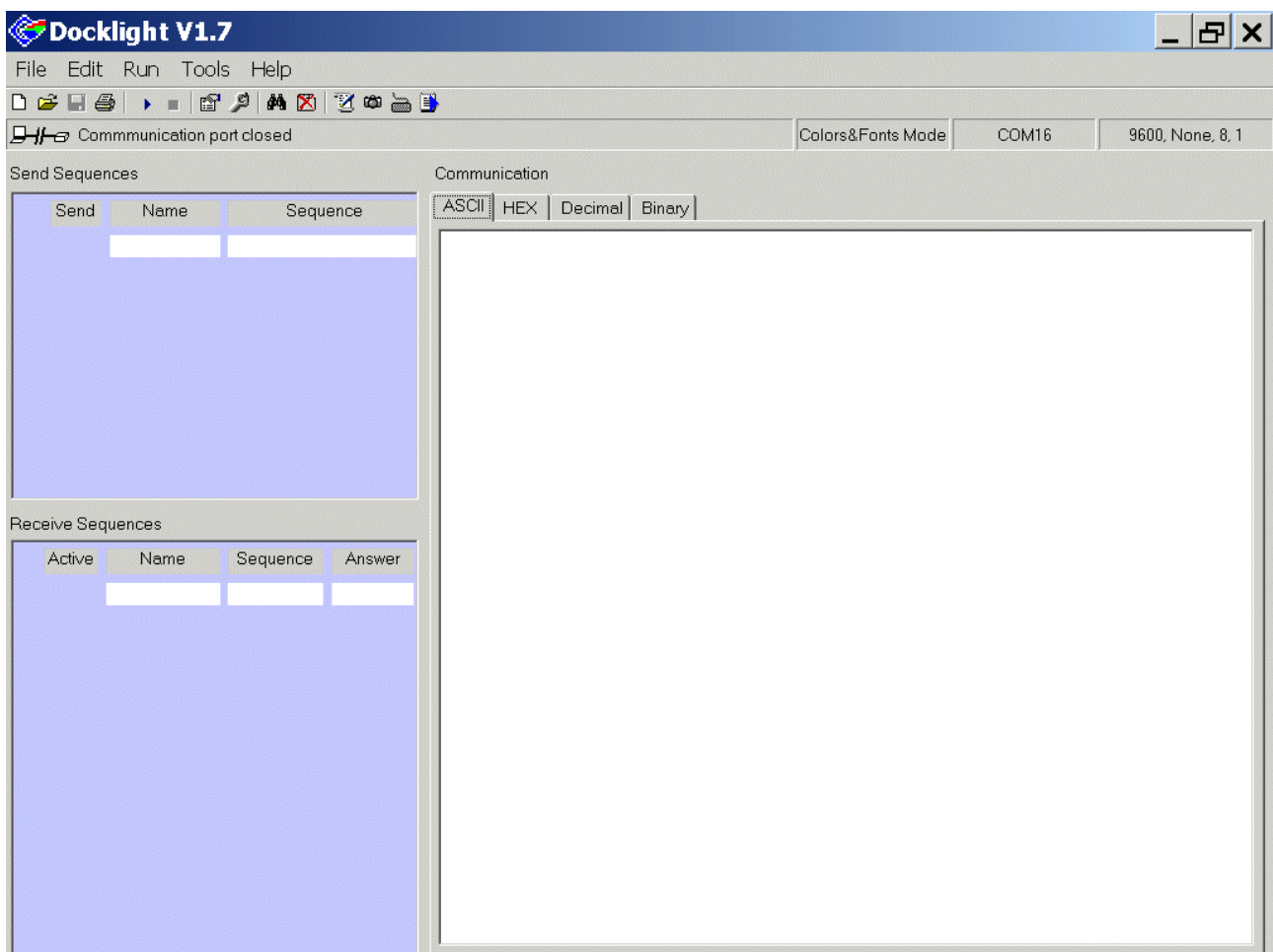
Access to previous and obsolete versions of Docklight

Docklight Scripting V1.6	Download Docklight Scripting V1.6 (3960 KB, Release 01/2007)
Docklight V1.6	Download Docklight V1.6 (3027 KB, Release 01/2007)
Docklight Scripting V1.5	Download Docklight Scripting V1.5 (3909 KB, Release 09/2004)
Docklight Scripting V1.4	Download Docklight Scripting V1.4 (3868 KB, Release 05/2004)
Docklight V1.4	Download Docklight V1.4 (3028 KB, Release 05/2004)
Docklight V1.3	Download Docklight V1.3 (3048 KB, Release 01/2004)
Docklight V1.2	Download Docklight V1.2 (2866 KB, Release 02/2003)
Docklight V1.1	Download Docklight V1.1 (2850 KB, Release 09/2002)
Docklight V1.0	Download Docklight V1.0 (2834 KB, Release 04/2002)

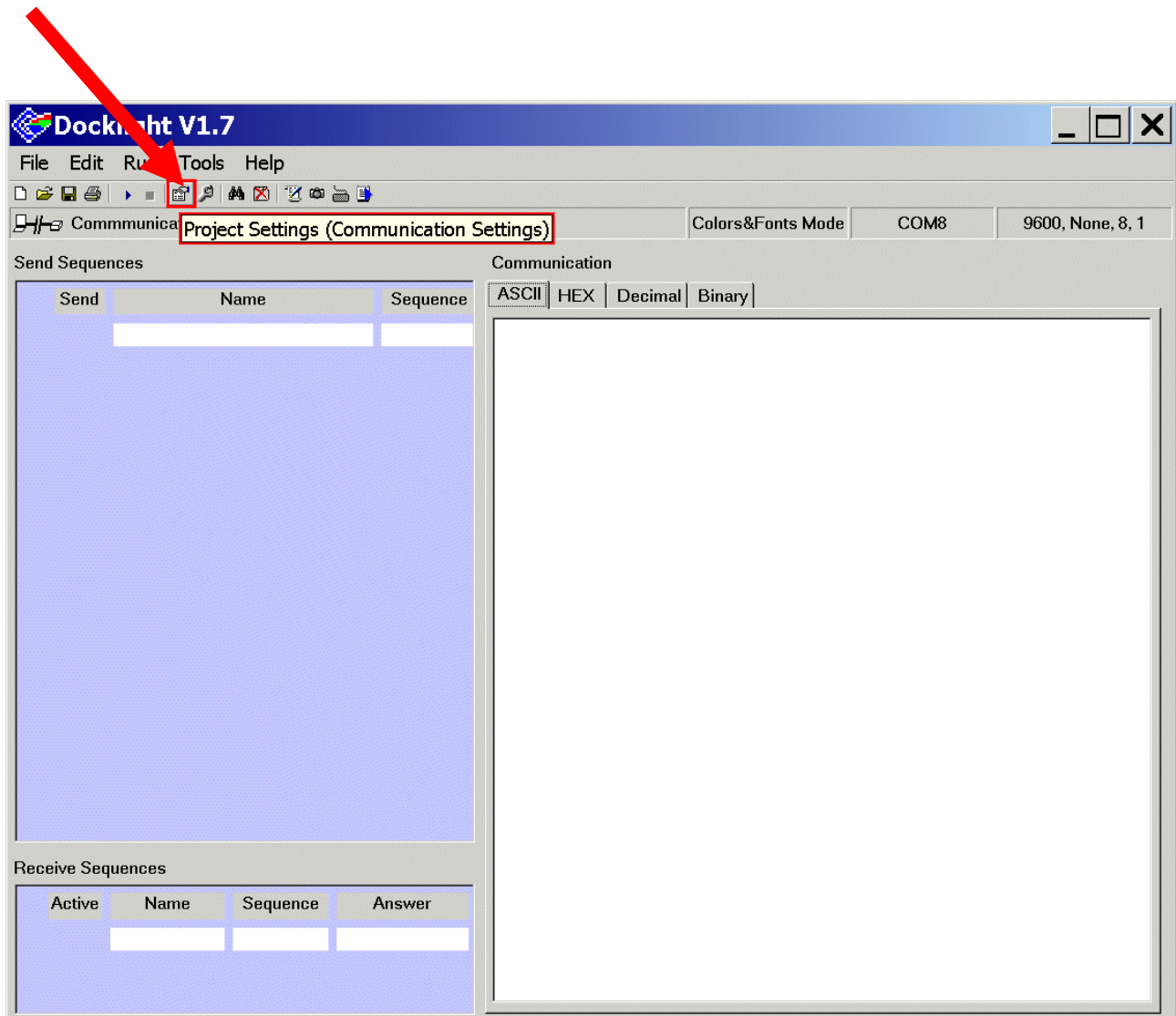
At the bottom of the page, there is a link: [back to the Docklight main page](#).



Now, **start** Docklight:



Click: Project Settings



Project Settings:

Communication: Communication Mode: **click** ☒ Send/Receive

Project Settings:

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

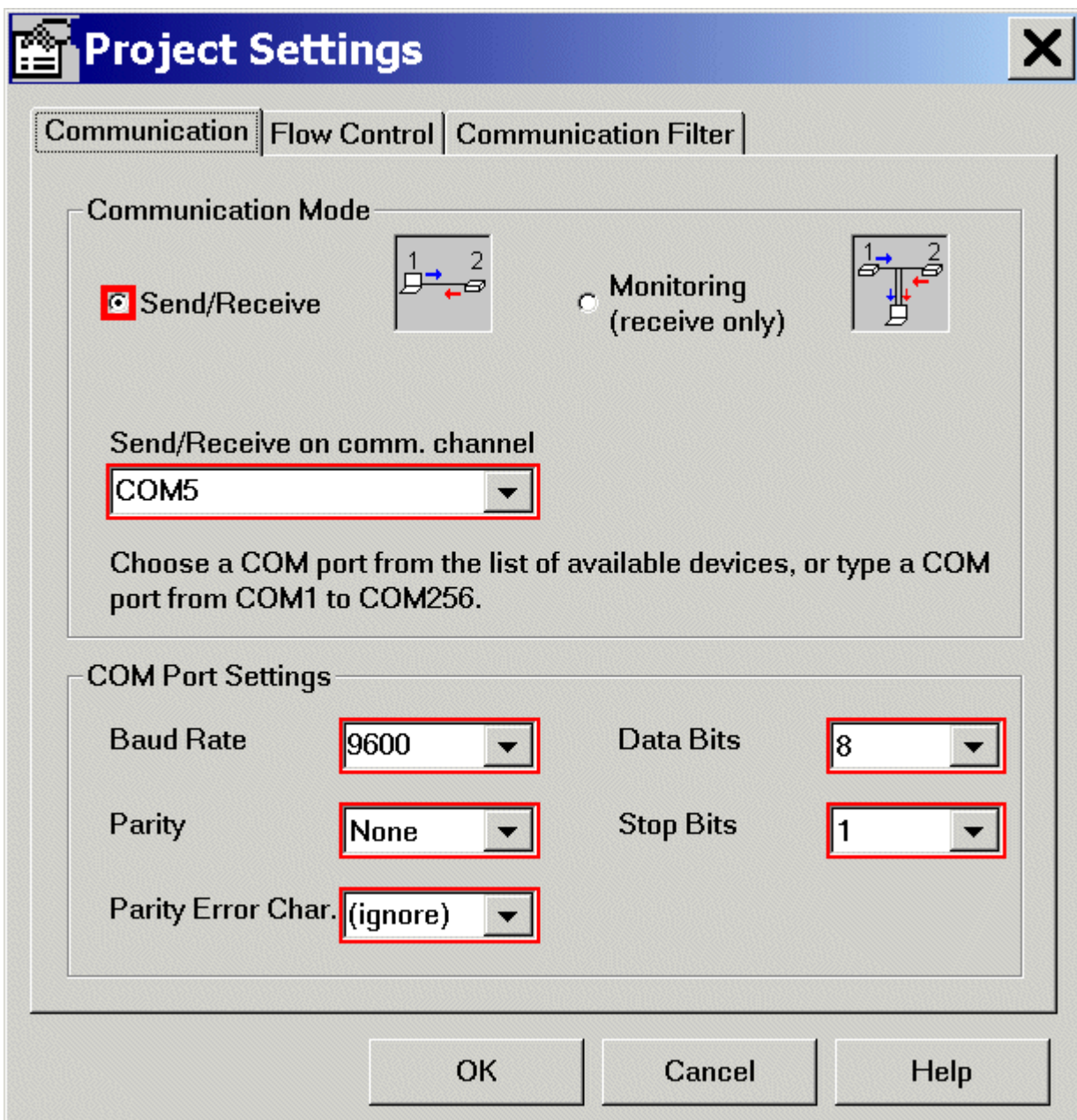
Project Settings: Communication: COM Port Settings: Baud Rate: **select** 9600

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

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

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

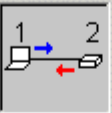
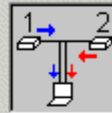
Project Settings: Communication: COM Port Settings: Stop Bits: **select** 1



Project Settings

Communication | Flow Control | Communication Filter

Communication Mode

☒ Send/Receive  ☐ Monitoring (receive only) 

Send/Receive on comm. channel

COM5

Choose a COM port from the list of available devices, or type a COM port from COM1 to COM256.

COM Port Settings

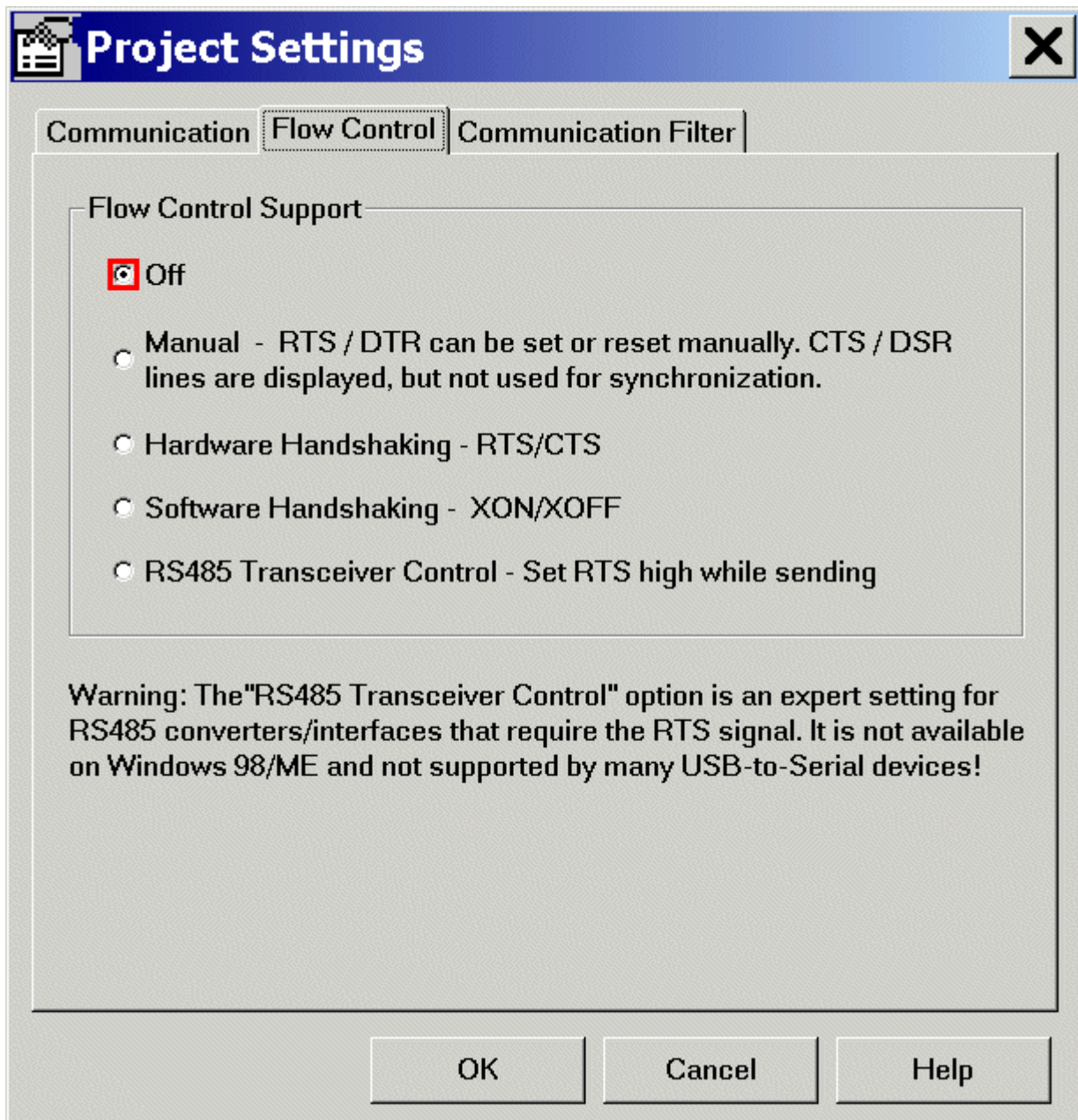
Baud Rate: 9600 Data Bits: 8

Parity: None Stop Bits: 1

Parity Error Char.: (ignore)

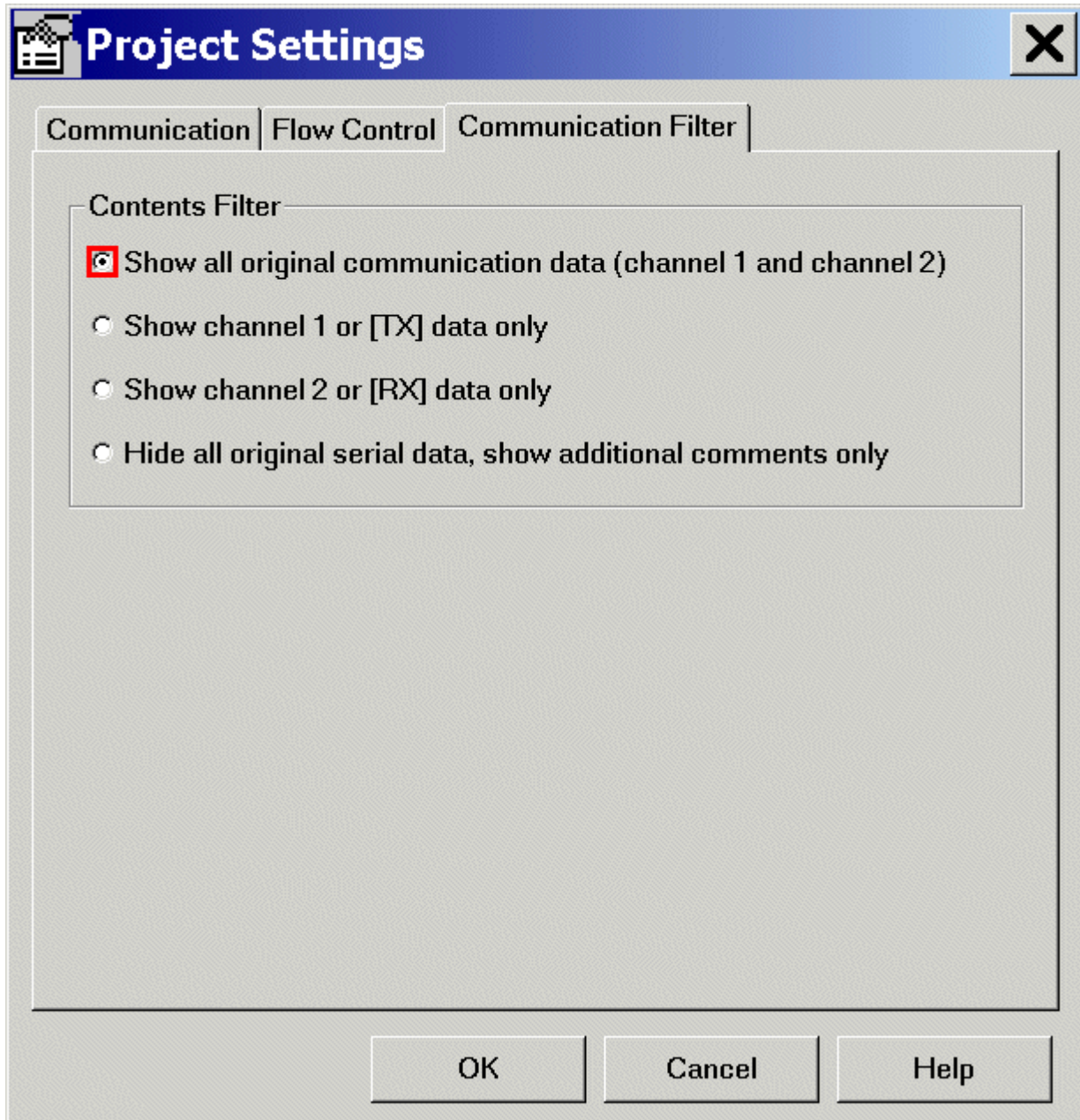
OK Cancel Help

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



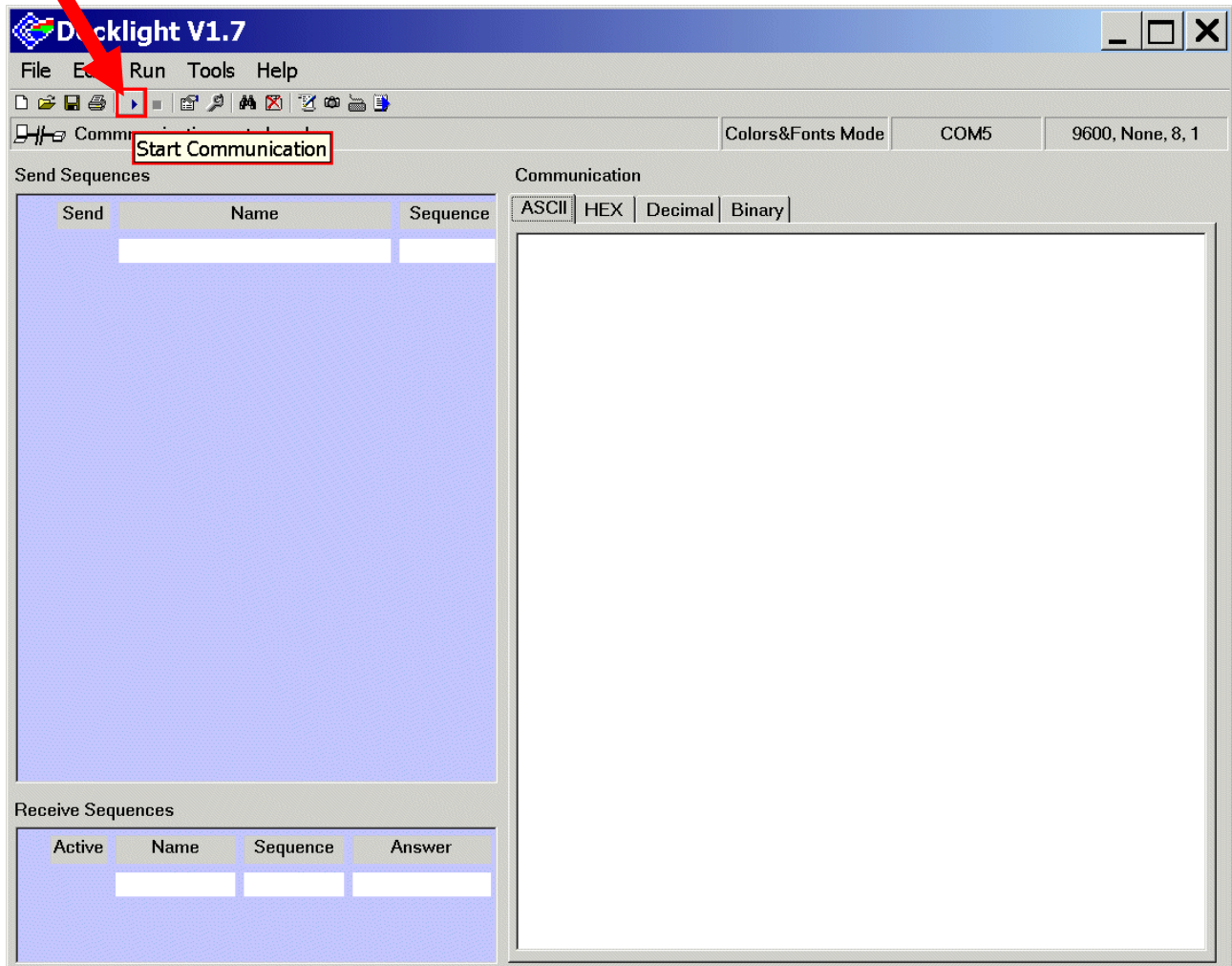
Project Settings:

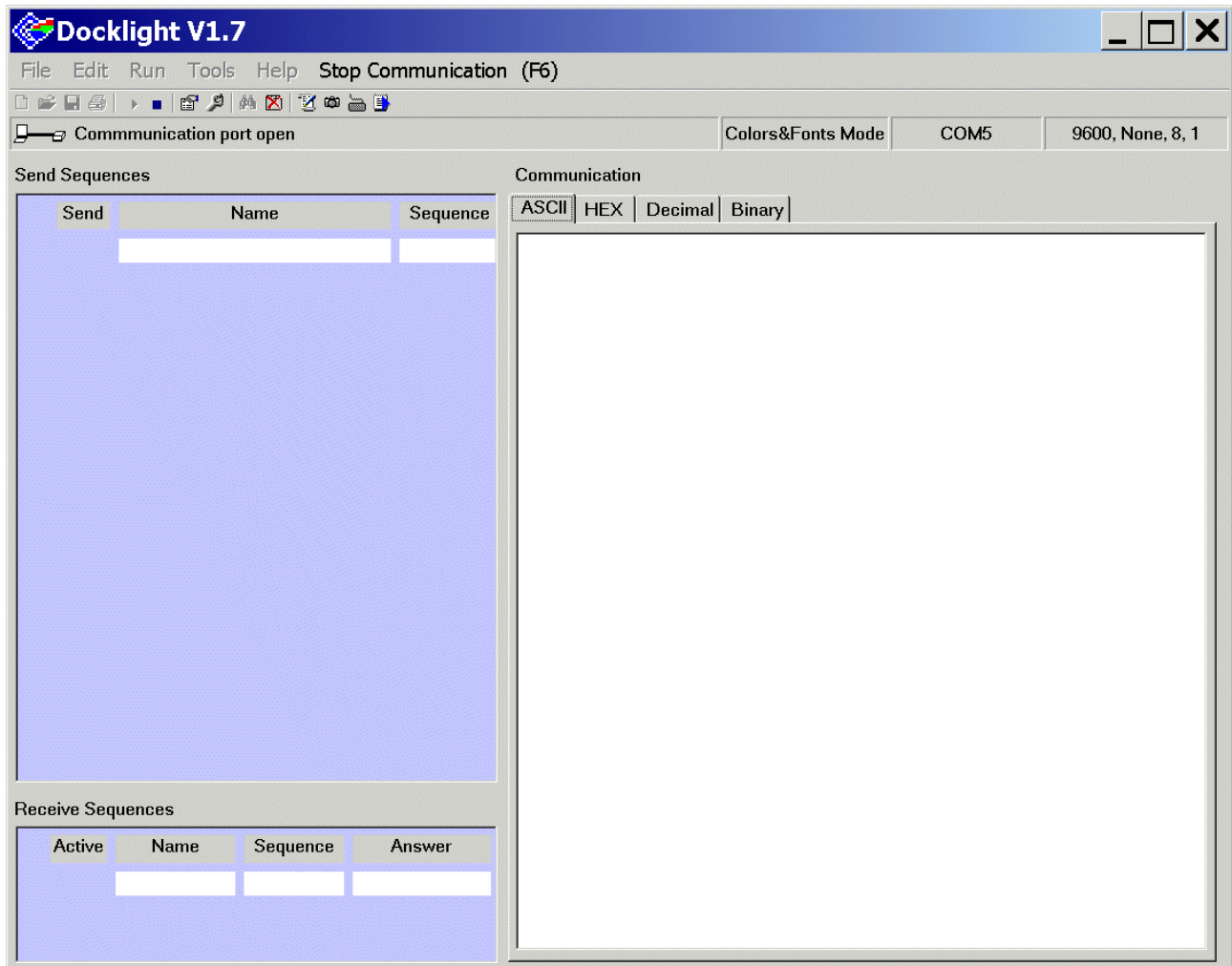
Communication Filter: Contents Filter: **click** ☒ Show all original communication data



OK

Click: 







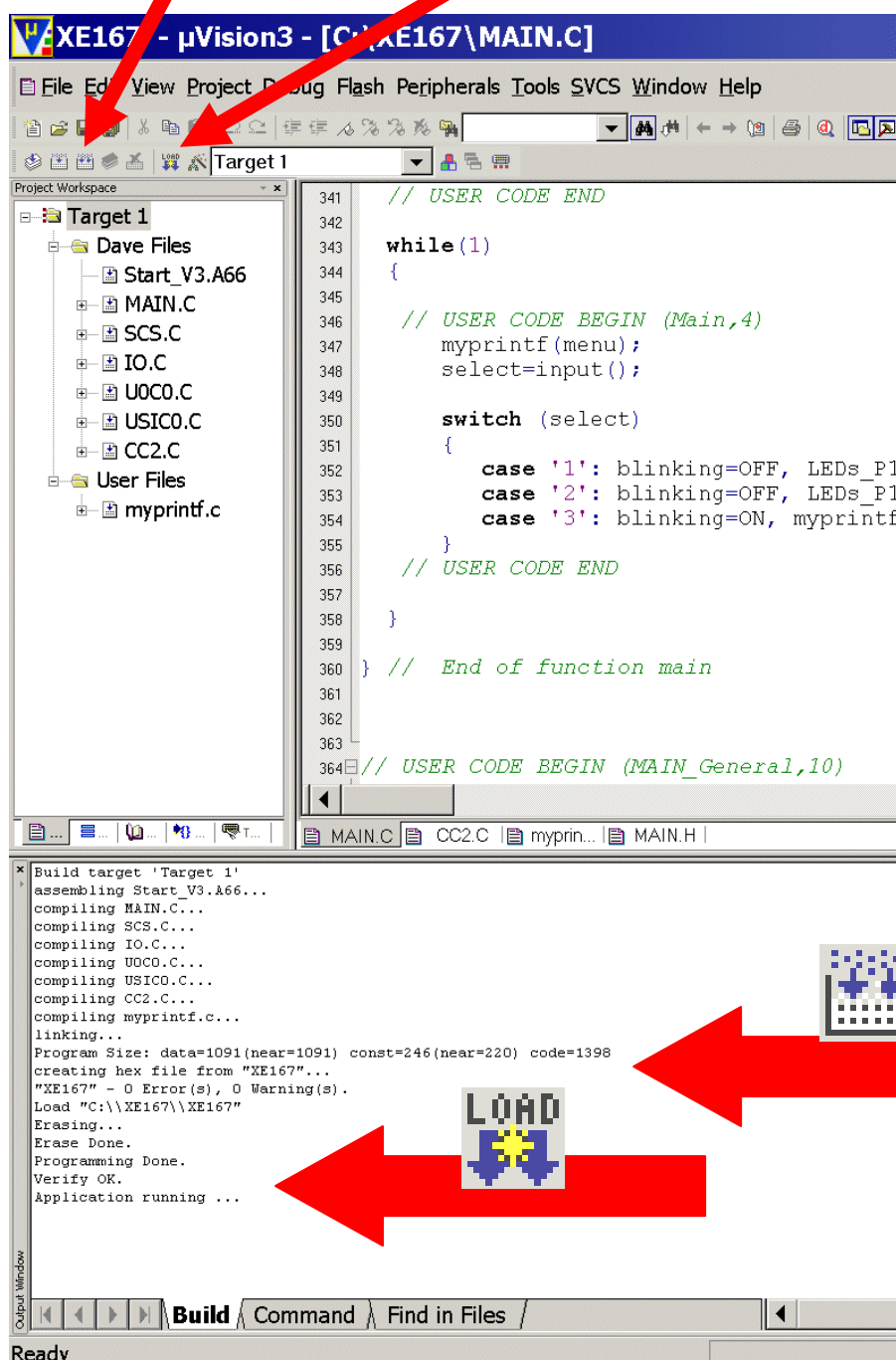
Note:
Docklight is now ready for serial communication!





Go to μ Vision:

1.) click:  2.) click: 



```

341 // USER CODE END
342
343 while(1)
344 {
345
346 // USER CODE BEGIN (Main,4)
347 myprintf(menu);
348 select=input();
349
350 switch (select)
351 {
352 case '1': blinking=OFF, LEDs_P1
353 case '2': blinking=OFF, LEDs_P1
354 case '3': blinking=ON, myprintf
355 }
356 // USER CODE END
357
358 }
359
360 } // End of function main
361
362
363
364 // USER CODE BEGIN (MAIN_General,10)

```

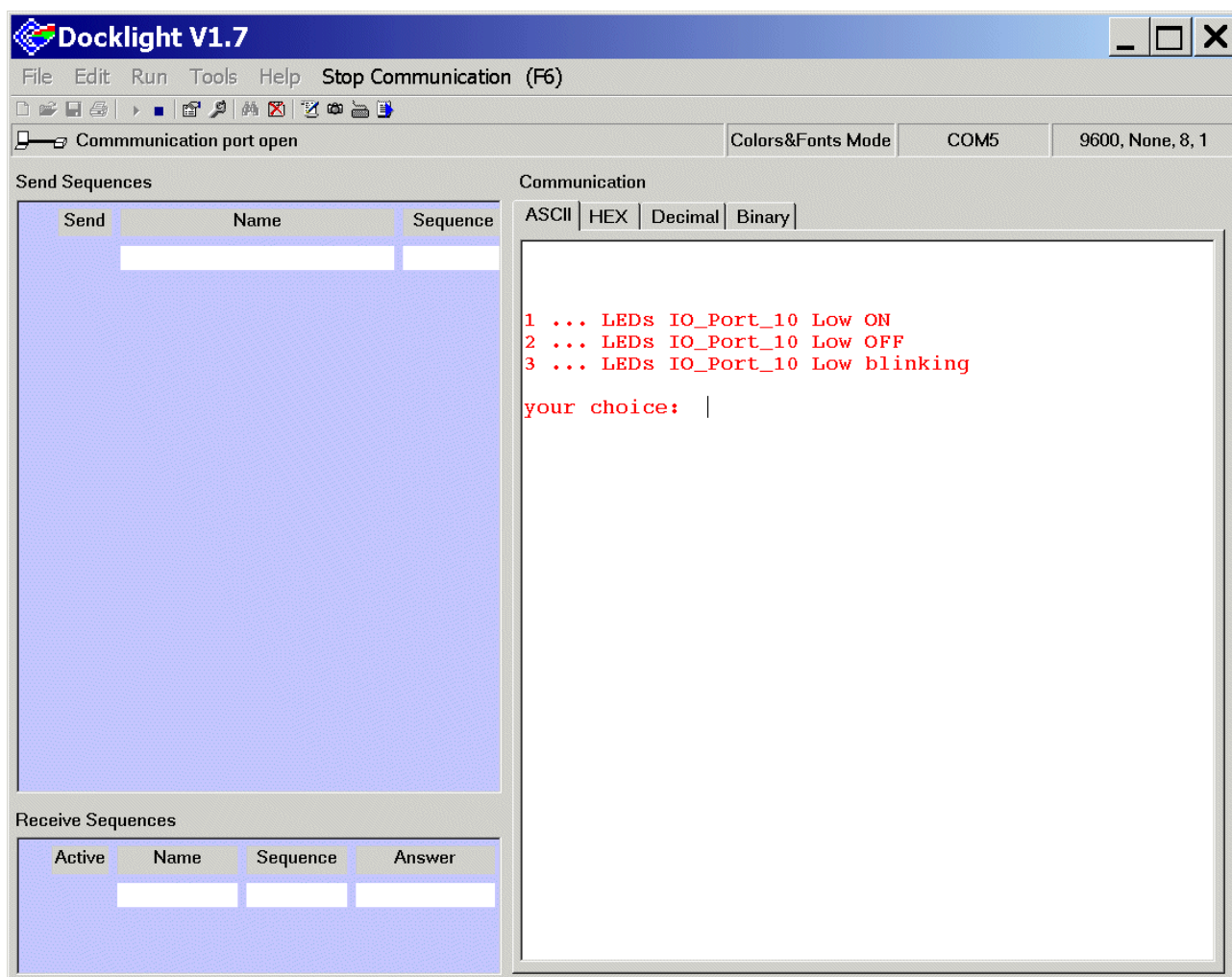
Build target 'Target 1'
assembling Start_V3.A66...
compiling MAIN.C...
compiling SCS.C...
compiling IO.C...
compiling UOC0.C...
compiling USIC0.C...
compiling CC2.C...
compiling myprintf.c...
linking...
Program Size: data=1091(near=1091) const=246(near=220) code=1398
creating hex file from "XE167"...
"XE167" - 0 Error(s), 0 Warning(s).
Load "C:\XE167\XE167"
Erasing...
Erase Done.
Programming Done.
Verify OK.
Application running ...

Build Command Find in Files

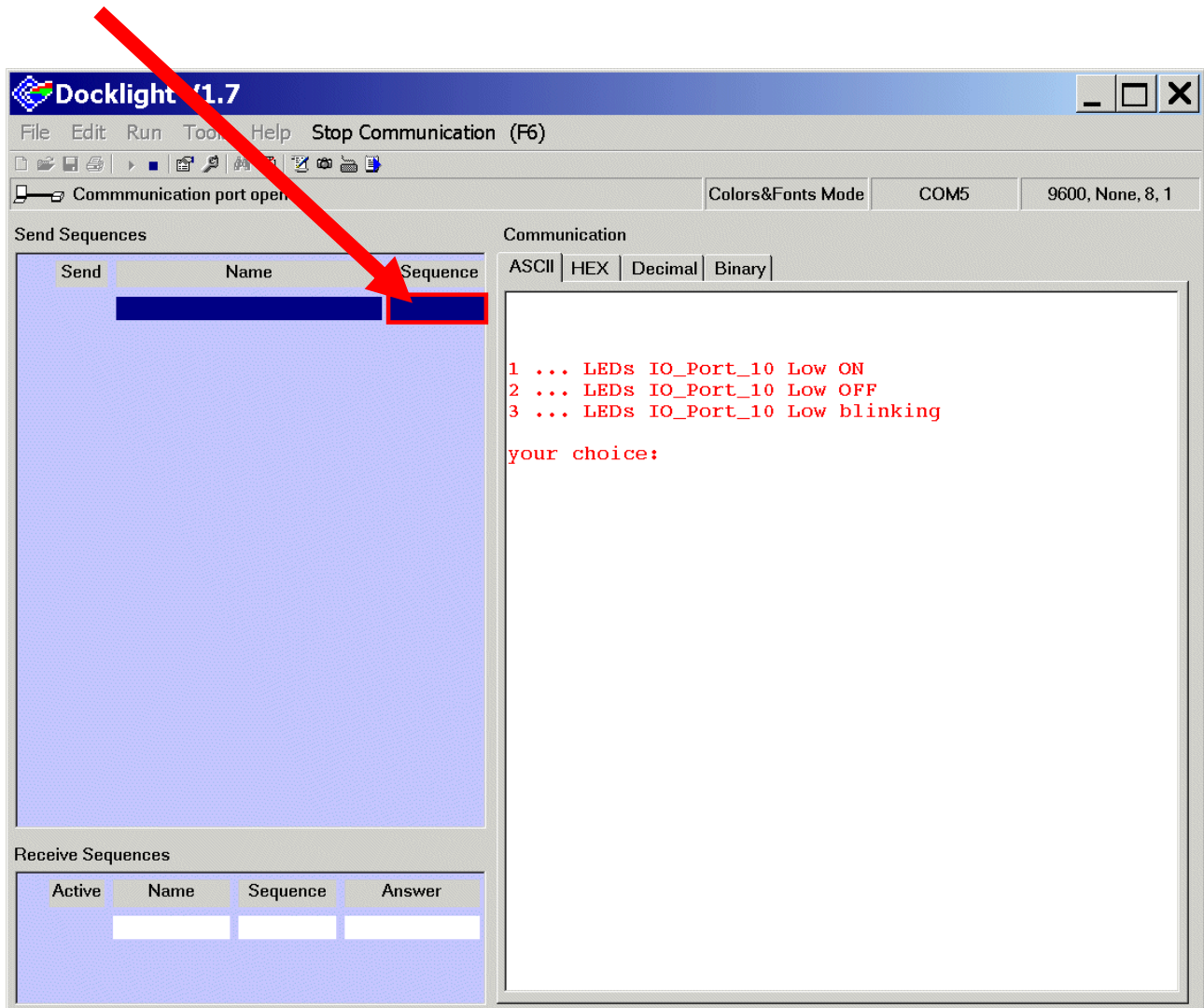
Ready



Go to Docklight and see the result:

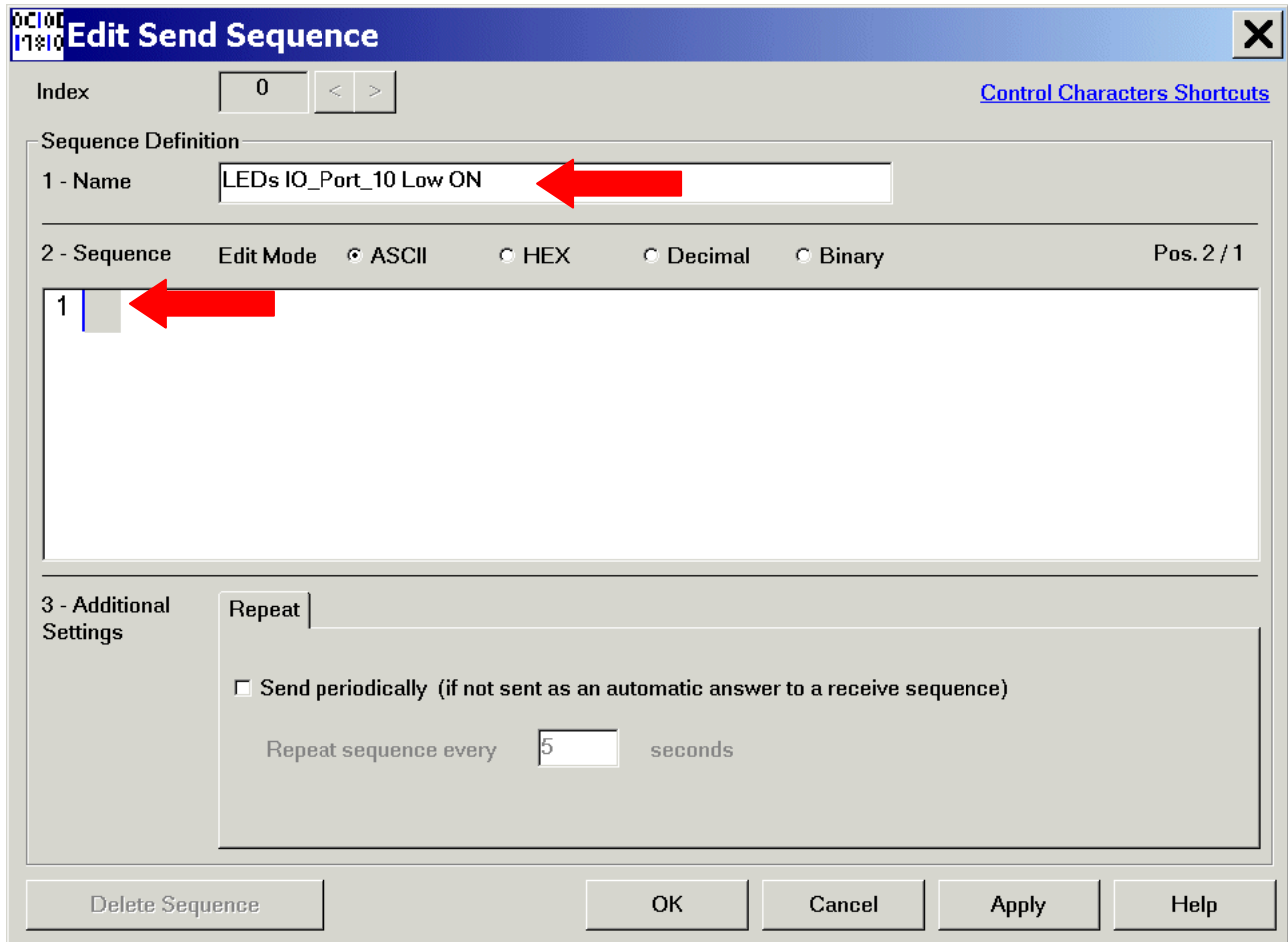


Double click inside the red box:



Edit Send Sequence: Sequence Definition: 1- Name: insert: LEDs IO_Port_10 Low ON

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



Edit Send Sequence

Index: 0

[Control Characters Shortcuts](#)

Sequence Definition

1 - Name: LEDs IO_Port_10 Low ON

2 - Sequence: Edit Mode: ☒ ASCII ☐ HEX ☐ Decimal ☐ Binary Pos. 2 / 1

1

3 - Additional Settings

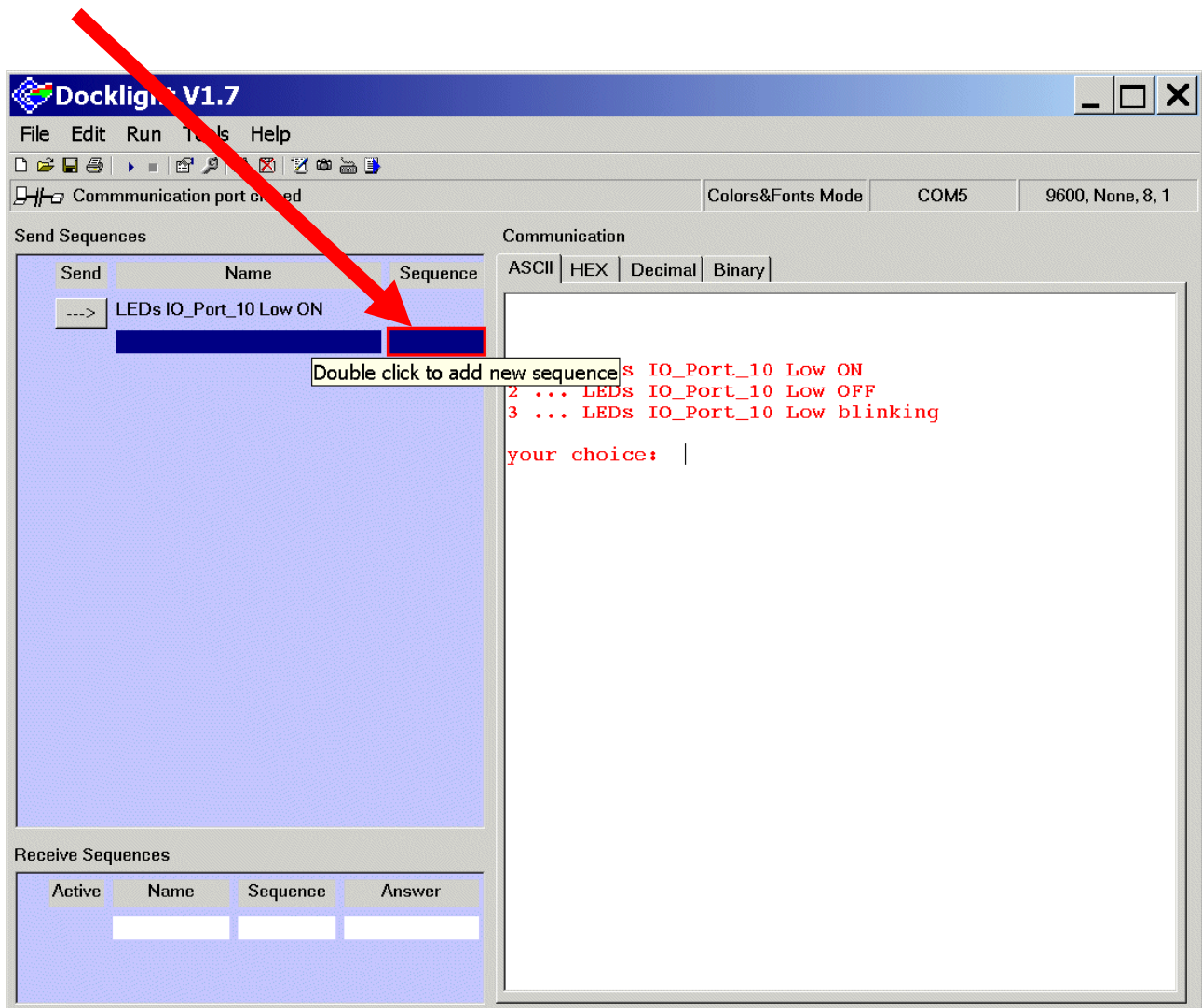
Repeat ☒ Send periodically (if not sent as an automatic answer to a receive sequence)

Repeat sequence every 5 seconds

Delete Sequence OK Cancel Apply Help

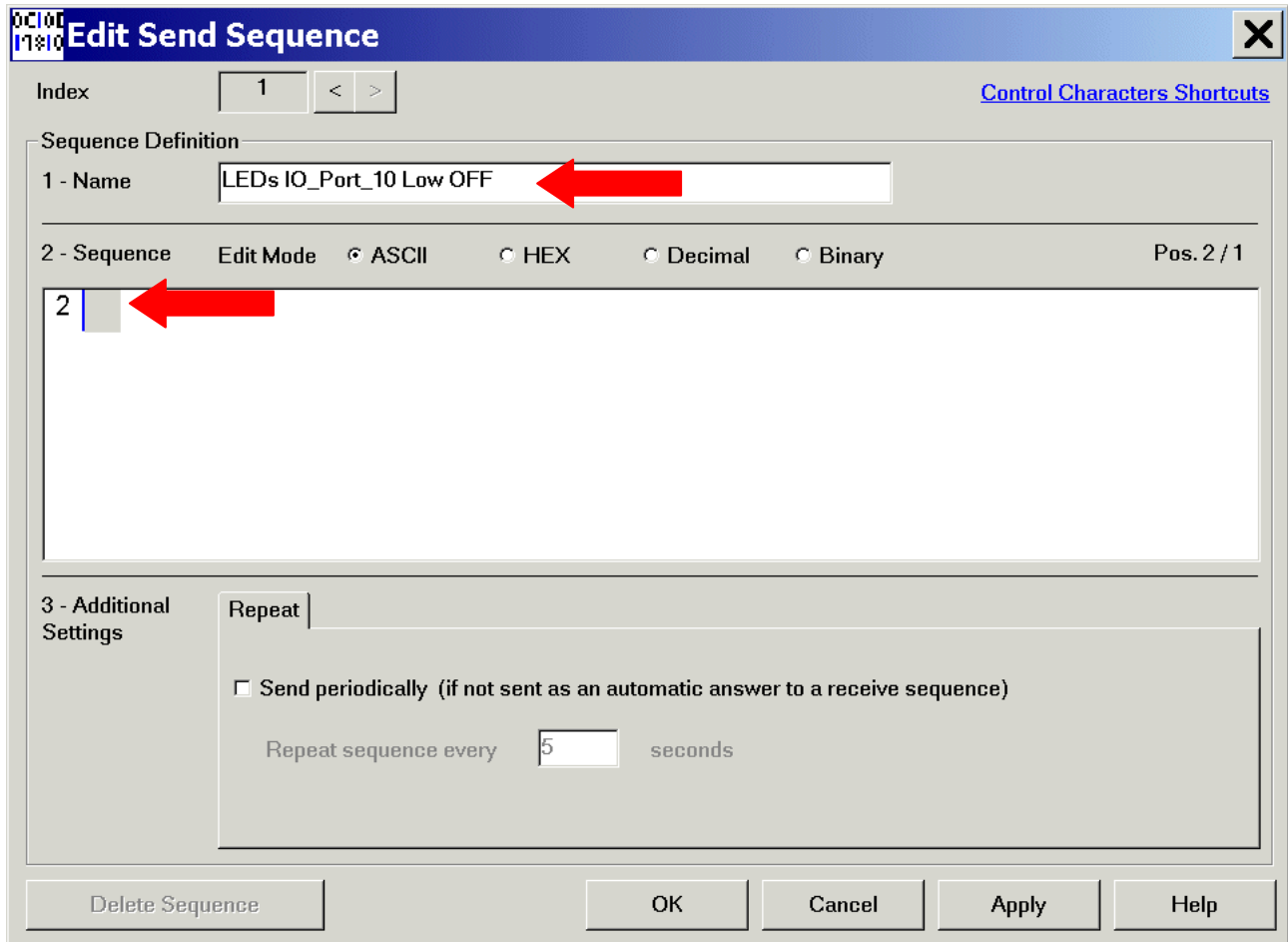
OK

Double click inside the red box:



Edit Send Sequence: Sequence Definition: 1- Name: insert: LEDs IO_Port_10 Low OFF

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



Edit Send Sequence

Index: 1

[Control Characters Shortcuts](#)

Sequence Definition

1 - Name: LEDs IO_Port_10 Low OFF

2 - Sequence: Edit Mode: ☒ ASCII ☐ HEX ☐ Decimal ☐ Binary Pos. 2 / 1

2

3 - Additional Settings

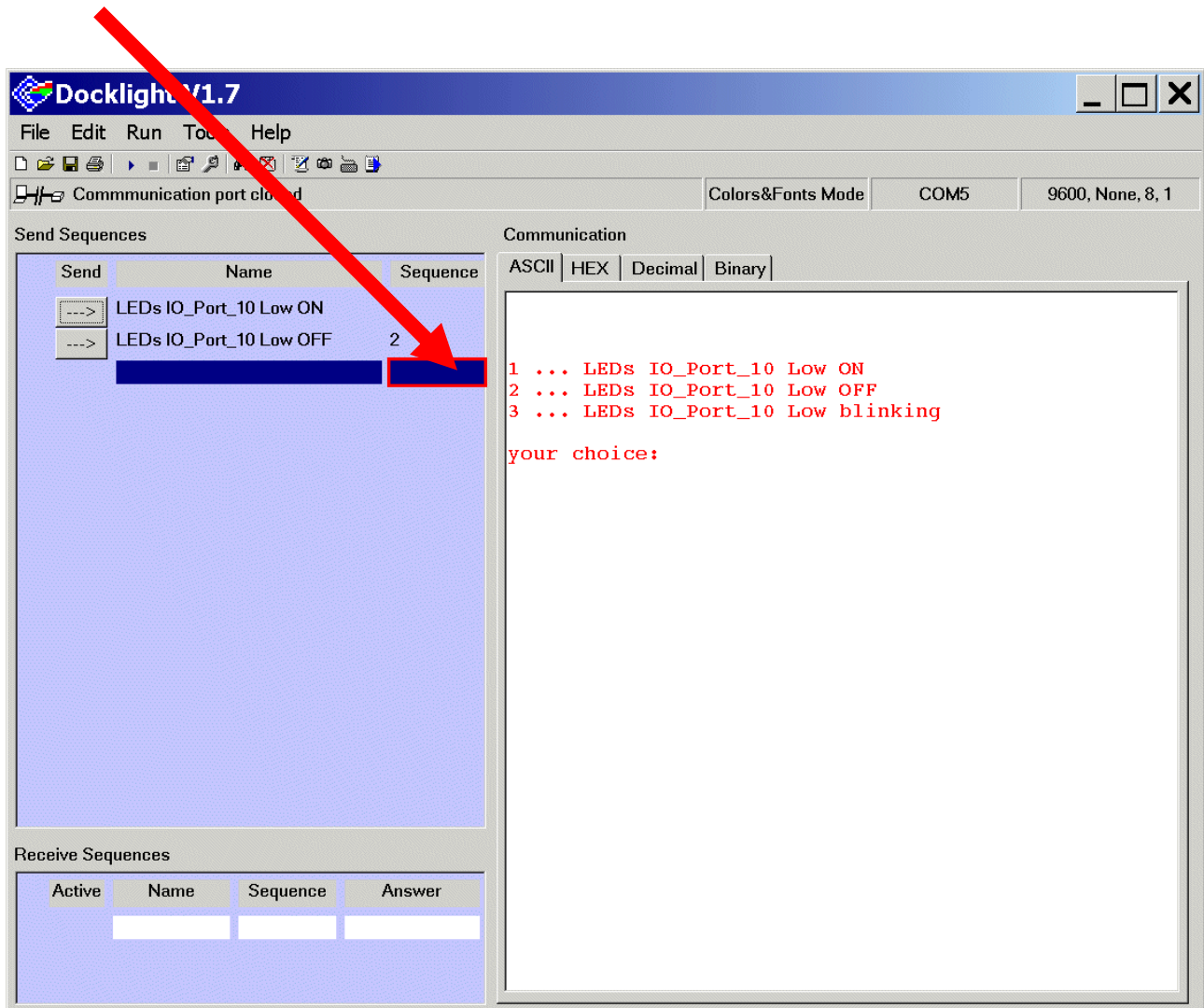
Repeat ☒ Send periodically (if not sent as an automatic answer to a receive sequence)

Repeat sequence every 5 seconds

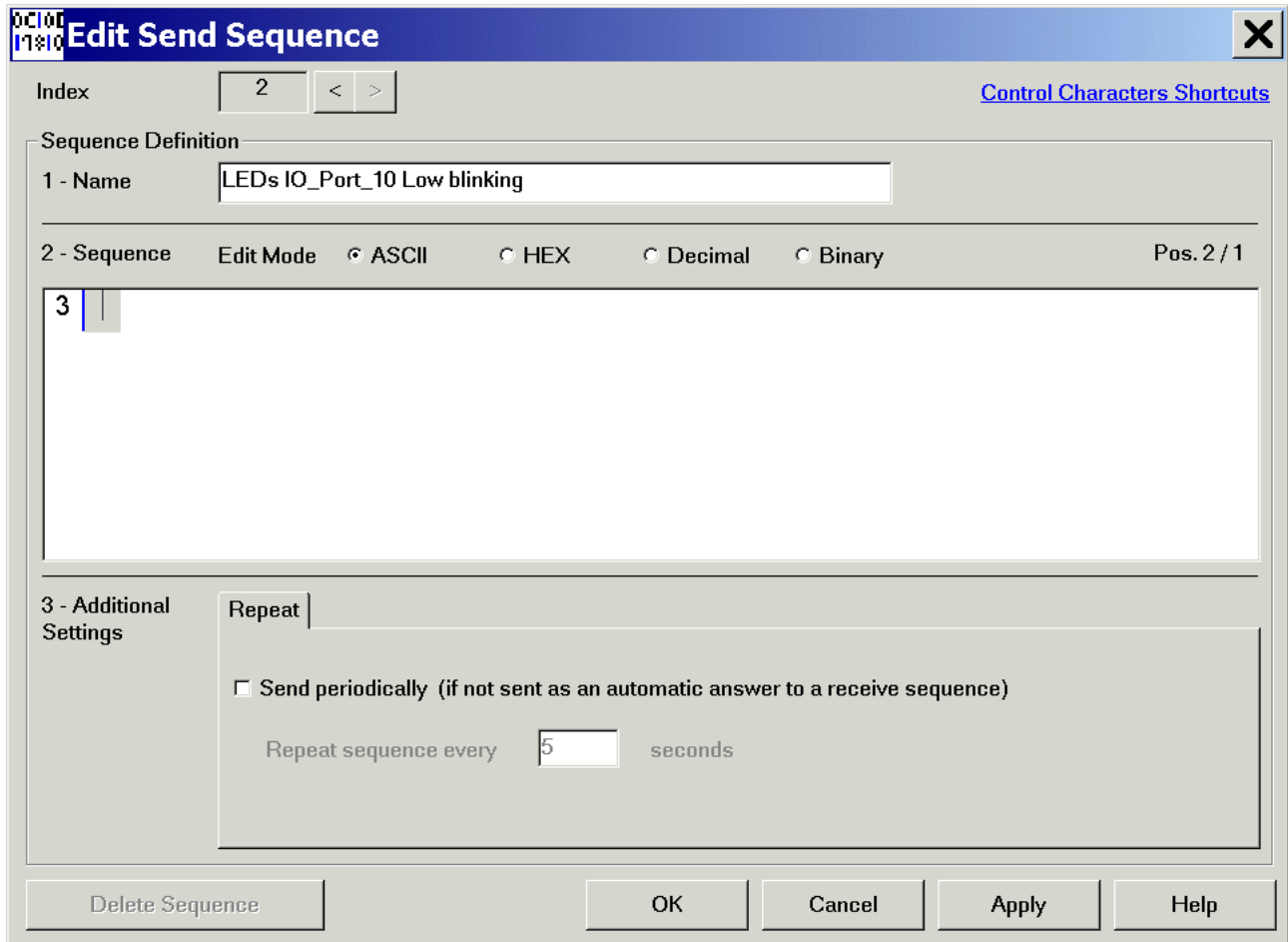
Delete Sequence OK Cancel Apply Help

OK

Double click inside the red box:



Edit Send Sequence: Sequence Definition: 1- Name: insert: LEDs IO_Port_10 Low blinking
Edit Send Sequence: Sequence Definition: 2- Sequence: insert: 3



Edit Send Sequence

Index: 2 < > [Control Characters Shortcuts](#)

Sequence Definition

1 - Name: LEDs IO_Port_10 Low blinking

2 - Sequence: Edit Mode: ☒ ASCII ☐ HEX ☐ Decimal ☐ Binary Pos. 2 / 1

3

3 - Additional Settings

Repeat ☒ Send periodically (if not sent as an automatic answer to a receive sequence)


Repeat sequence every 5 seconds

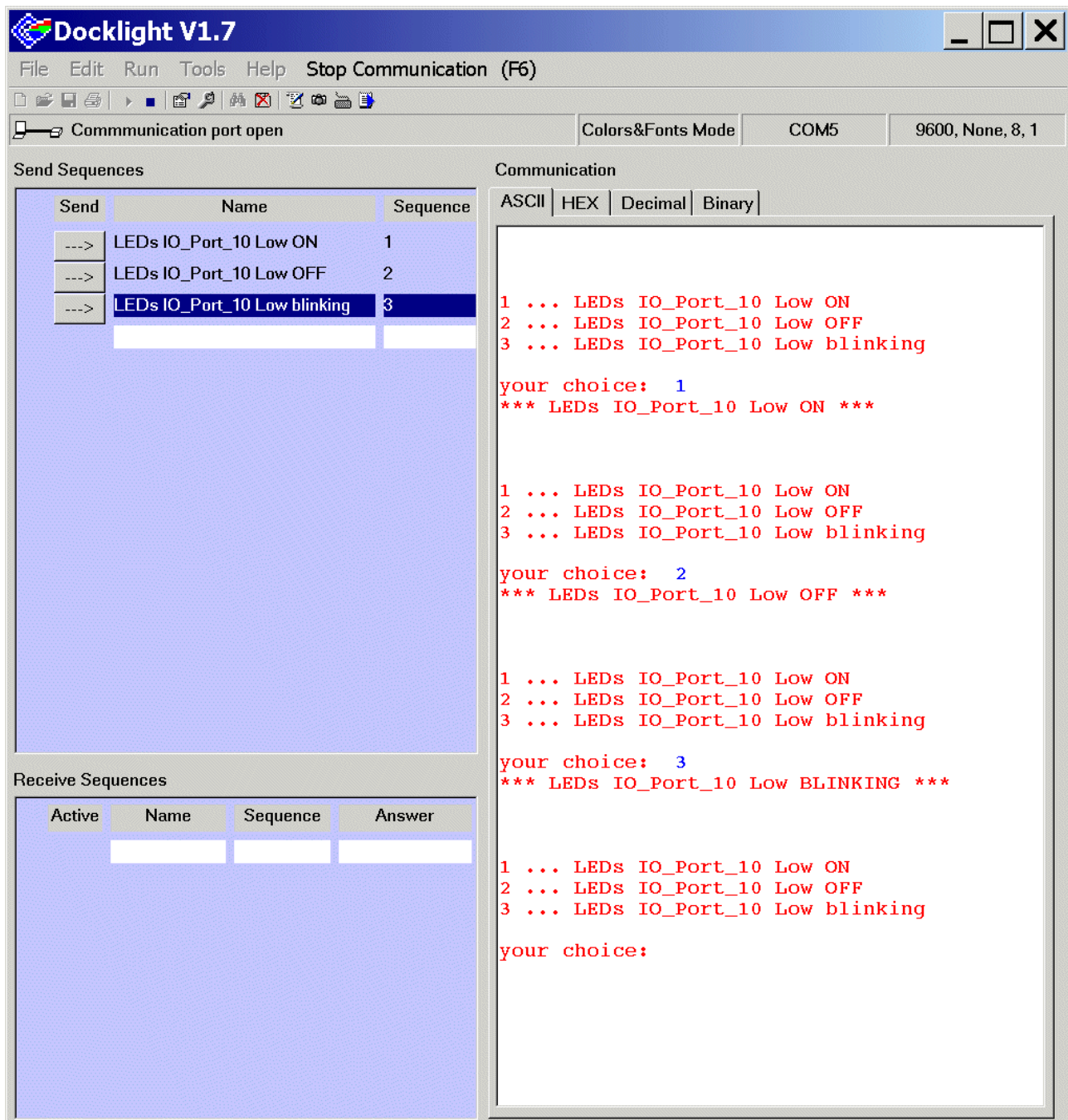
Delete Sequence OK Cancel Apply Help

OK

Click  LEDs IO_Port_10 Low ON or

Click  LEDs IO_Port_10 Low OFF or

Click  LEDs IO_Port_10 Low blinking and **check** the result on your Evaluation Board:





Conclusion:

In this step-by-step book you have learned how to use the XE167 board together with the Keil tool chain.

Now you can easily expand your "hello world" program to suit your needs!

You can connect either a part of - or your entire application to the Easy Kit Board.

You are also able to benchmark any of your algorithms to find out if the selected microcontroller fulfils all the required functions within the time frame needed.

Have fun and enjoy working with XE167 microcontrollers!

Note:

There are step-by-step books for 8 bit microcontrollers (e.g. XC866 and XC888), 16 bit microcontrollers (e.g. C16x, XC16x and XE16x/XC2xxx) and 32 bit microcontrollers (e.g. TC1796 and TC1130).

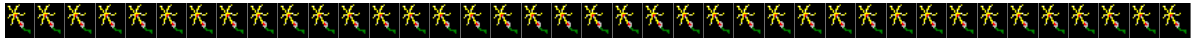
All these step-by-step books use the same microcontroller resources and the same example code.

This means: configuration steps, function names and variable names are identical.

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

There are even more programming examples using the same style available [e.g. ADC-examples, CAPCOM6-examples (e.g. BLDC-Motor, playing music), Simulator-examples, C++ examples] based on these step-by-step books.

6.) Feedback (XE167): 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>