AP32133

TC1766

TC1766 Starter Kit: "Cookery Book" for a hello world application using Altium's TASKING TriCore toolset

Microcontrollers

Never stop thinking

Edition 2008-11-14

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.

AP32110		
Revision History:	2008-10	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 10.

Introduction:

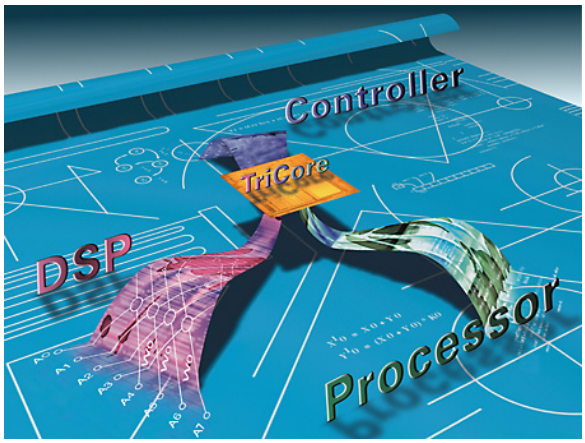
This “Application Note” / “Appnote” is a Hands-On Training / Cookery Book / step-by-step book. It will help inexperienced users to get the TC1766 / TC176x / TC116x Family Starter Kit up and running.

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 TC1766 Starter 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.

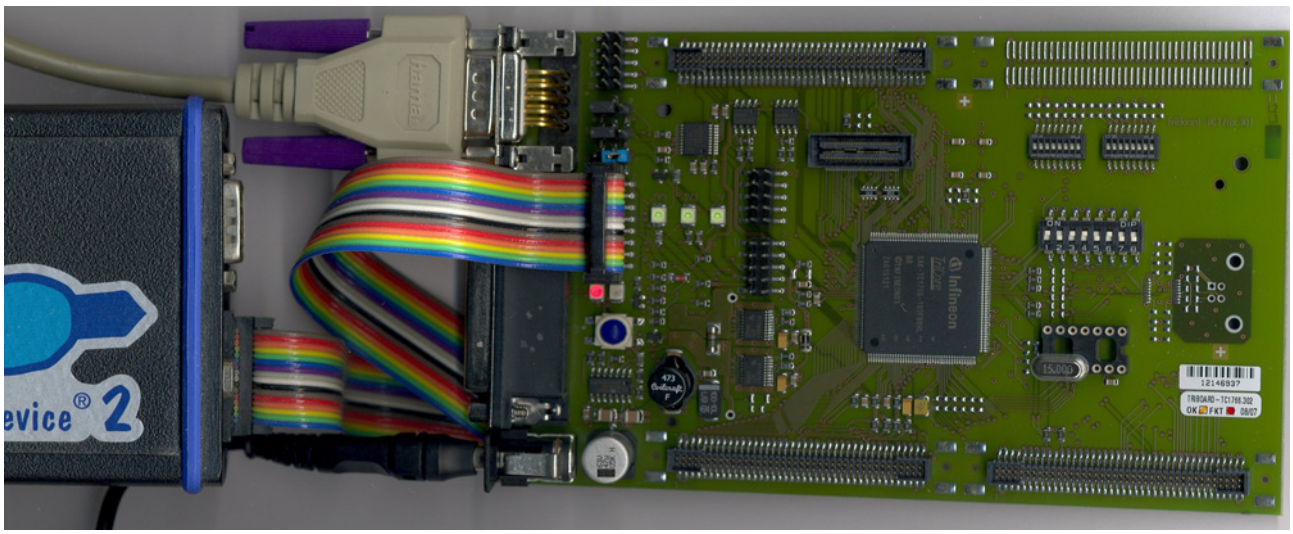
Have fun and enjoy TriCore!



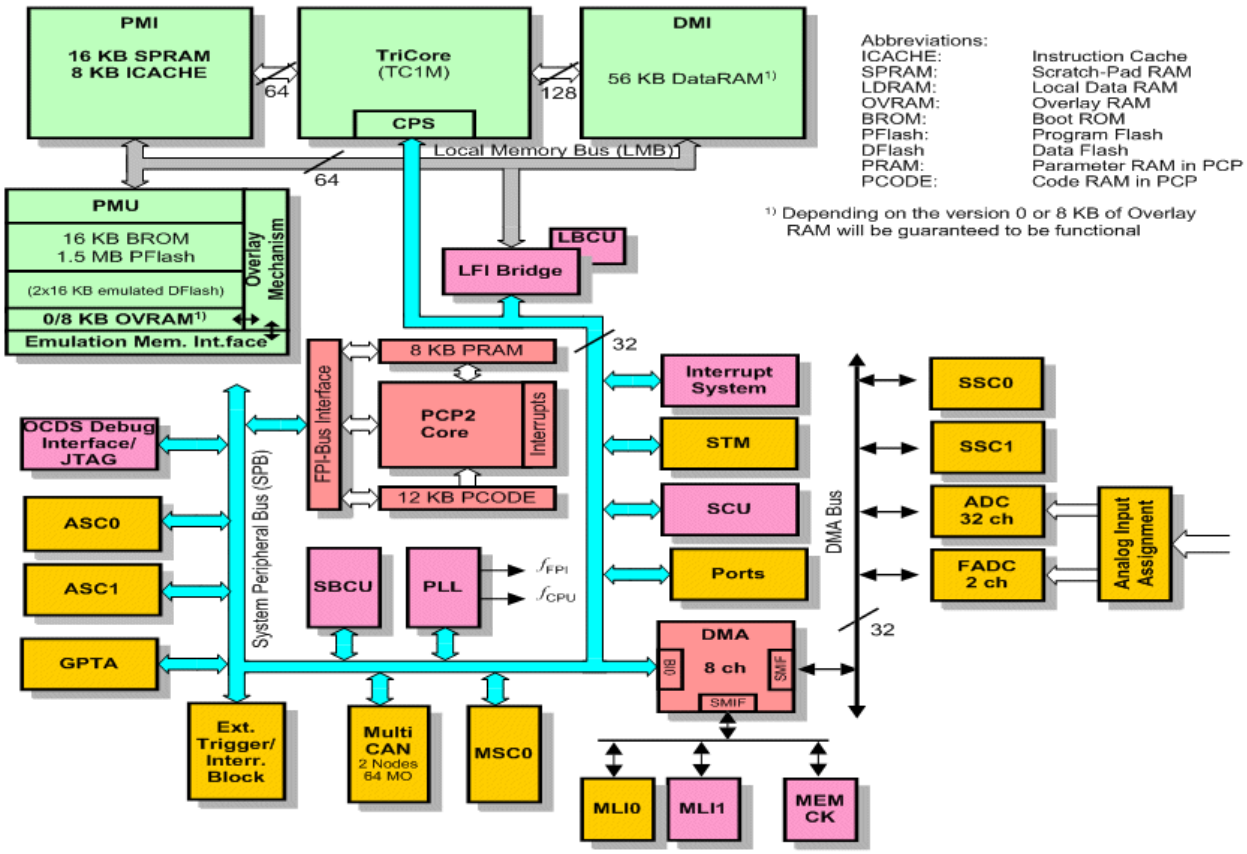
Programming Examples

TC1766

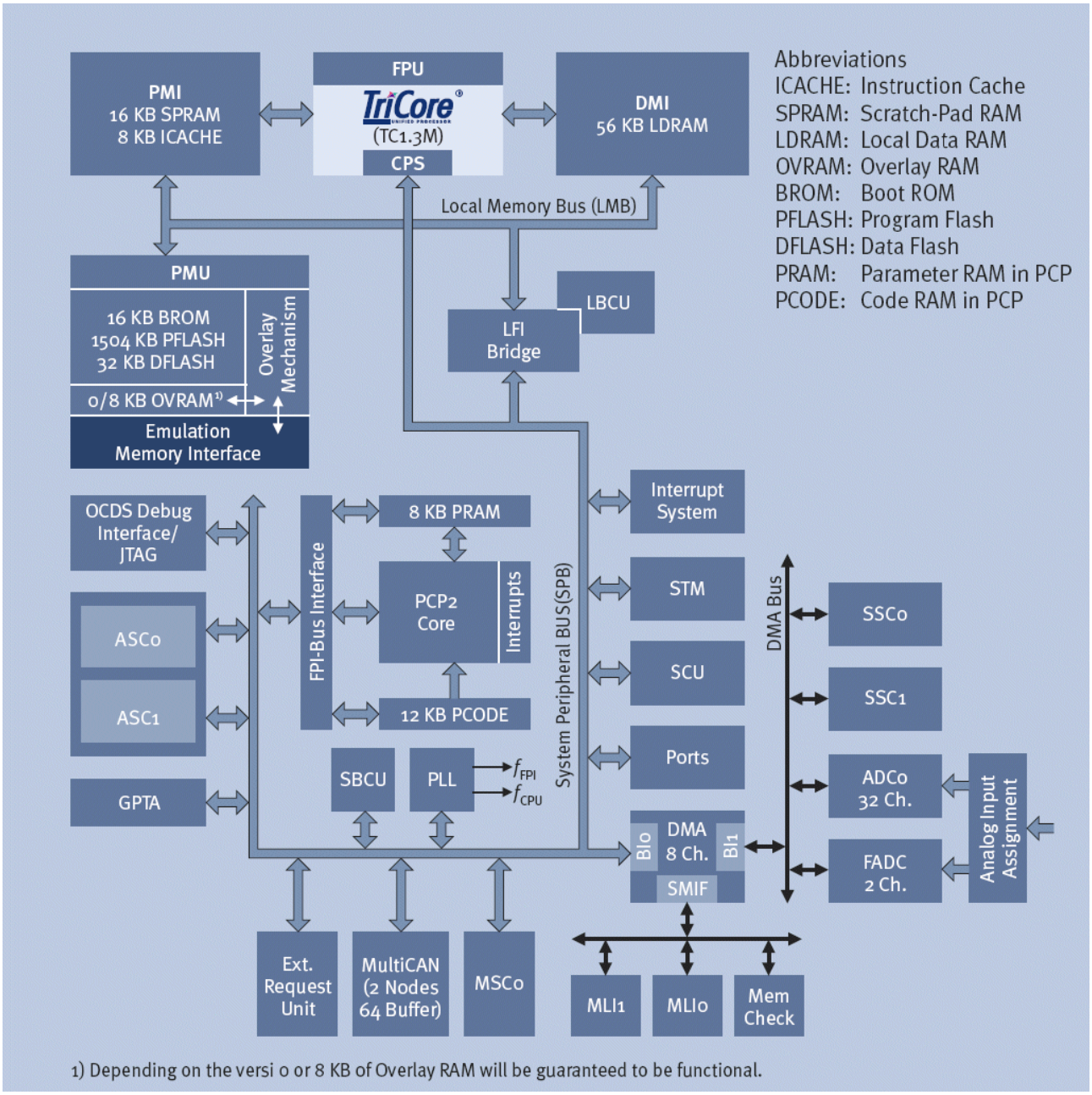
Starter Kit



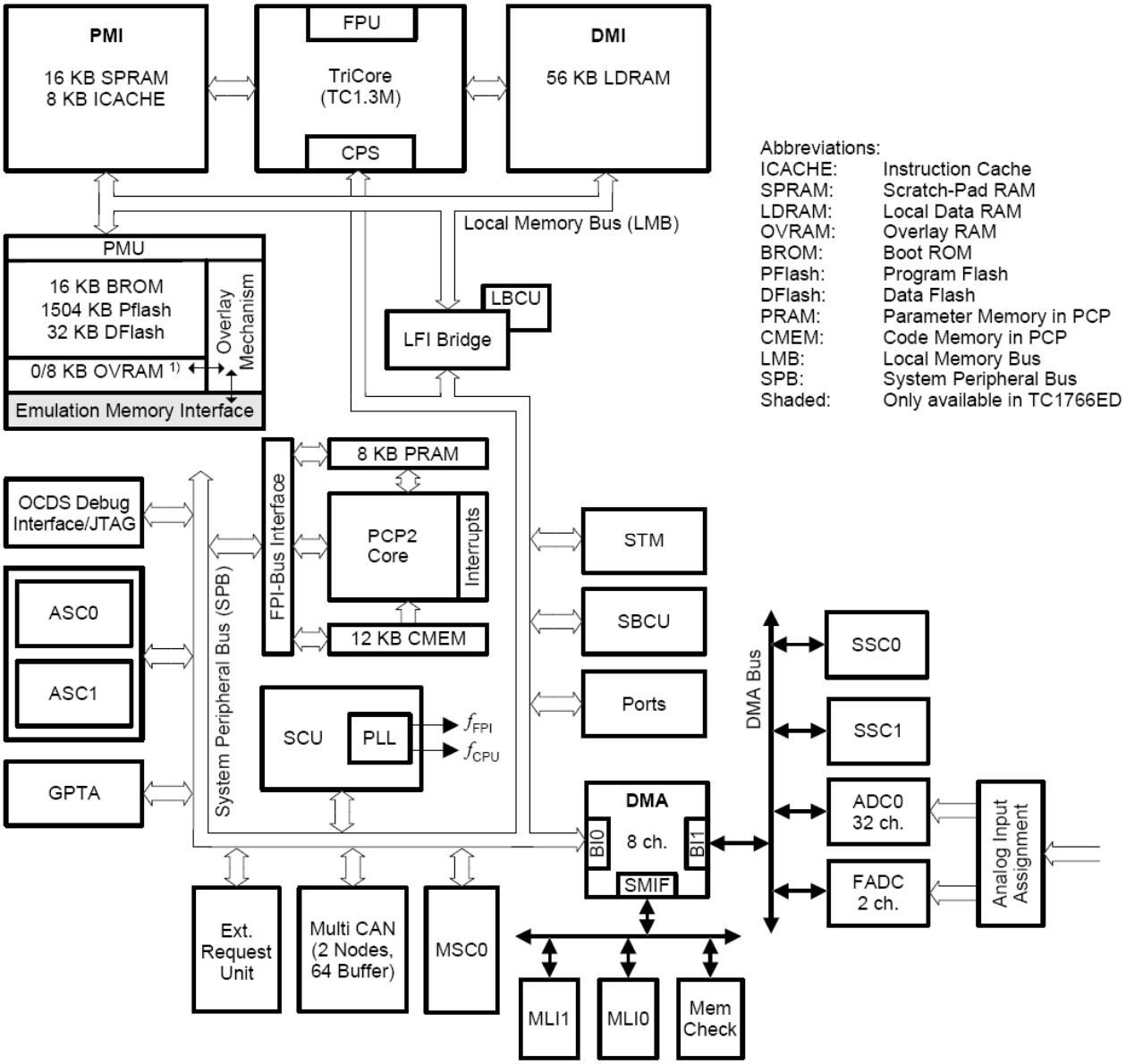
TC1766 Block Diagram (Source: Product Marketing)



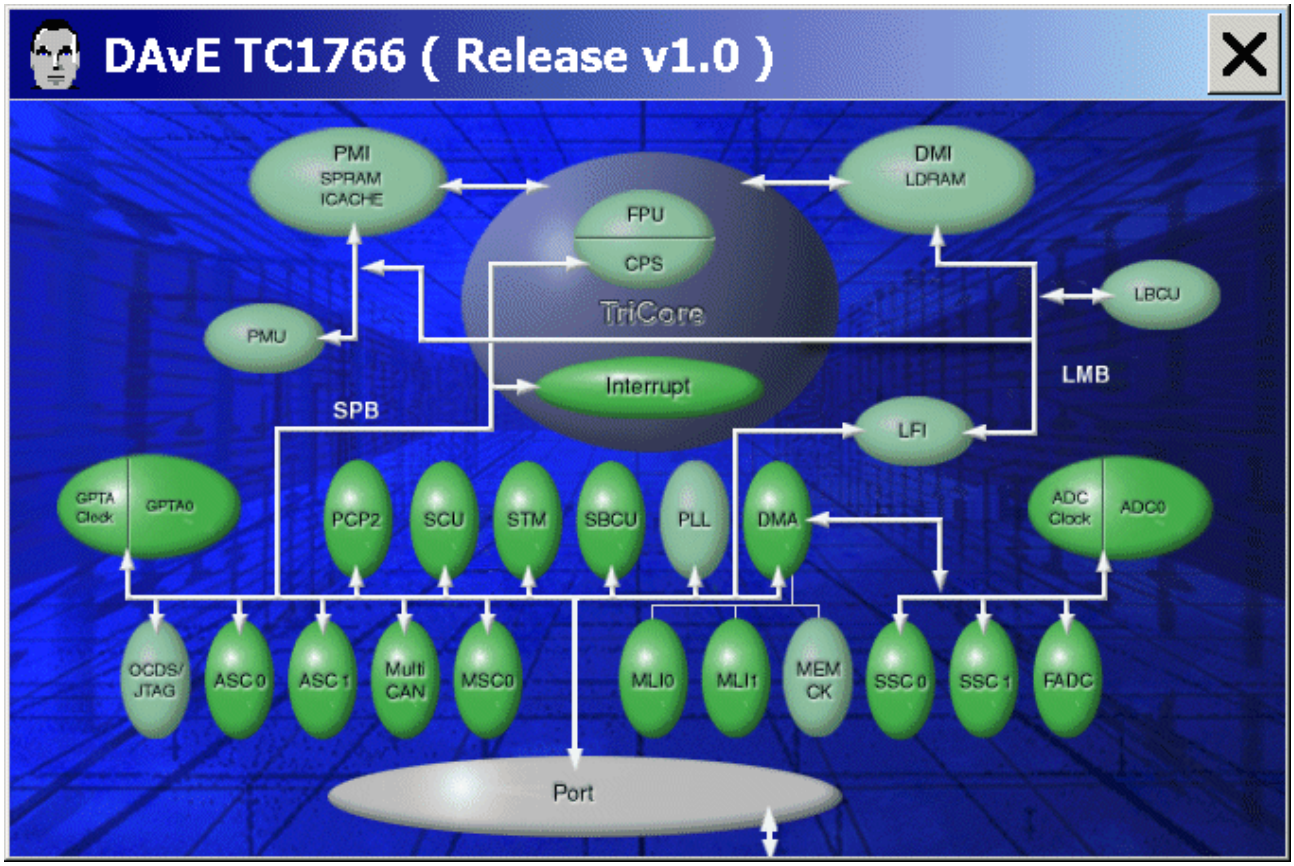
TC1766 Block Diagram (Source: Product Sheet)



TC1766 Block Diagram (Source: User's Manual)



TC1766 Block Diagram (Source: DAvE)



Note:
Just by comparing the different sources of block diagrams, you should be able to get a complete picture of the TC1766 microcontroller and to answer some of your initial questions.

“Cookery Book“

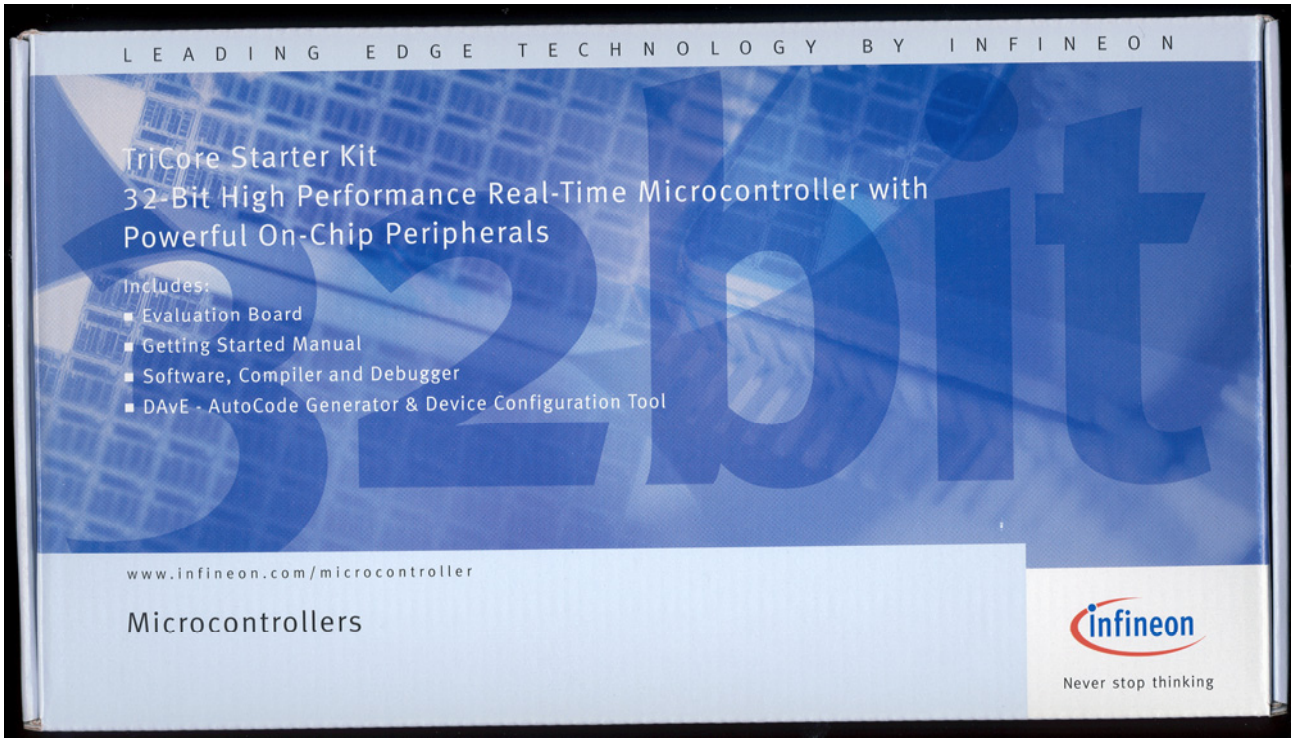
For your first programming example for the TC1766 Starter Kit Board:

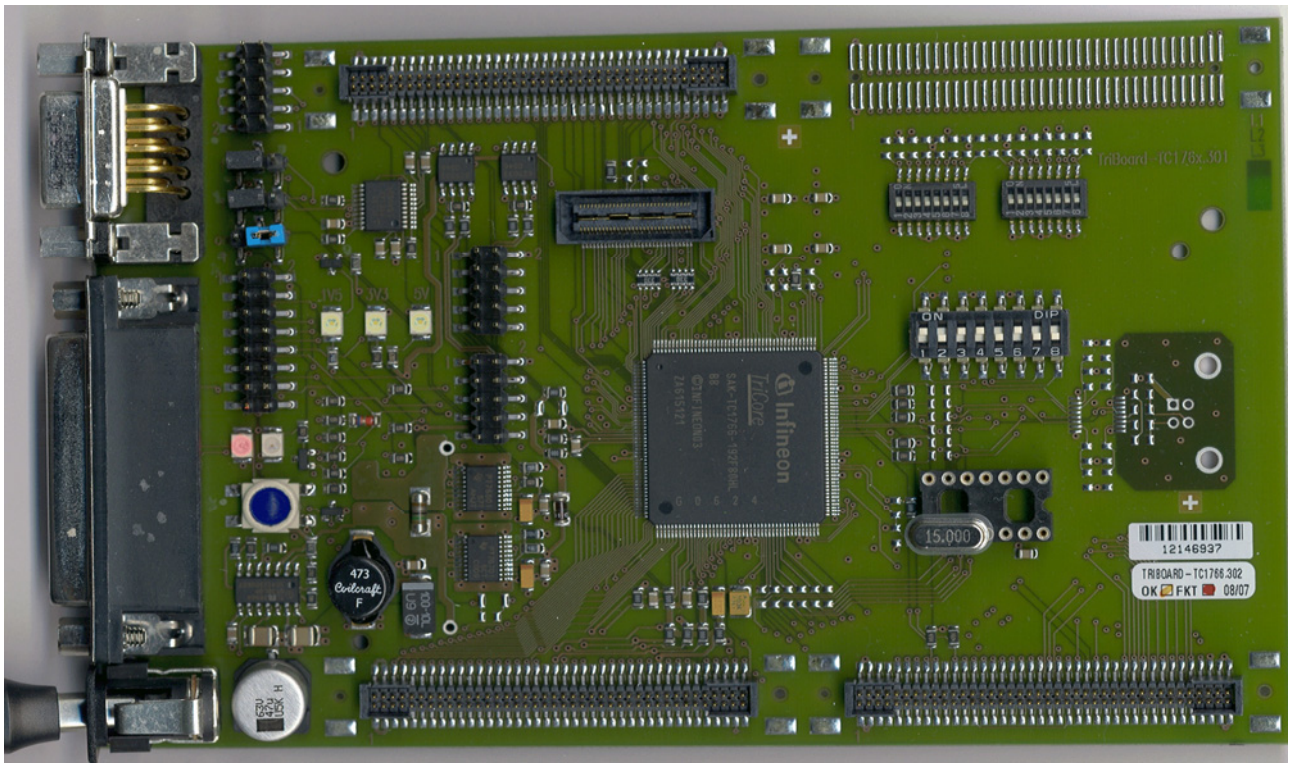
Your program:	
Chapter/ Step:	*** **Recipes** ***
1.)	**TC1766 Board** **Power Supply, Jumper Setting, Serial cable to the notebook, pls-Debugger**
2.)	**DAvE – program generator** **DAvE installation (mothersystem) + DAvE Update-installation for TC1766 (DIP-file)**
3.)	**Using DAvE** Microcontroller initialization for your programming example
4.) 	**Using the TASKING Development Tools (C/C++/EC++ Compiler)** Programming of your application with Altium’s TASKING TriCore tool chain (EDE) - v2.3r1 Locating programs into the 1,5 MByte OnChipProgramFlash (PFLASH), using OnChipSRAM)
5.)	**Using the pls Debugger** Using the pls Debugger to download (program into Flash) and run your program

Feedback

6.)	Feedback

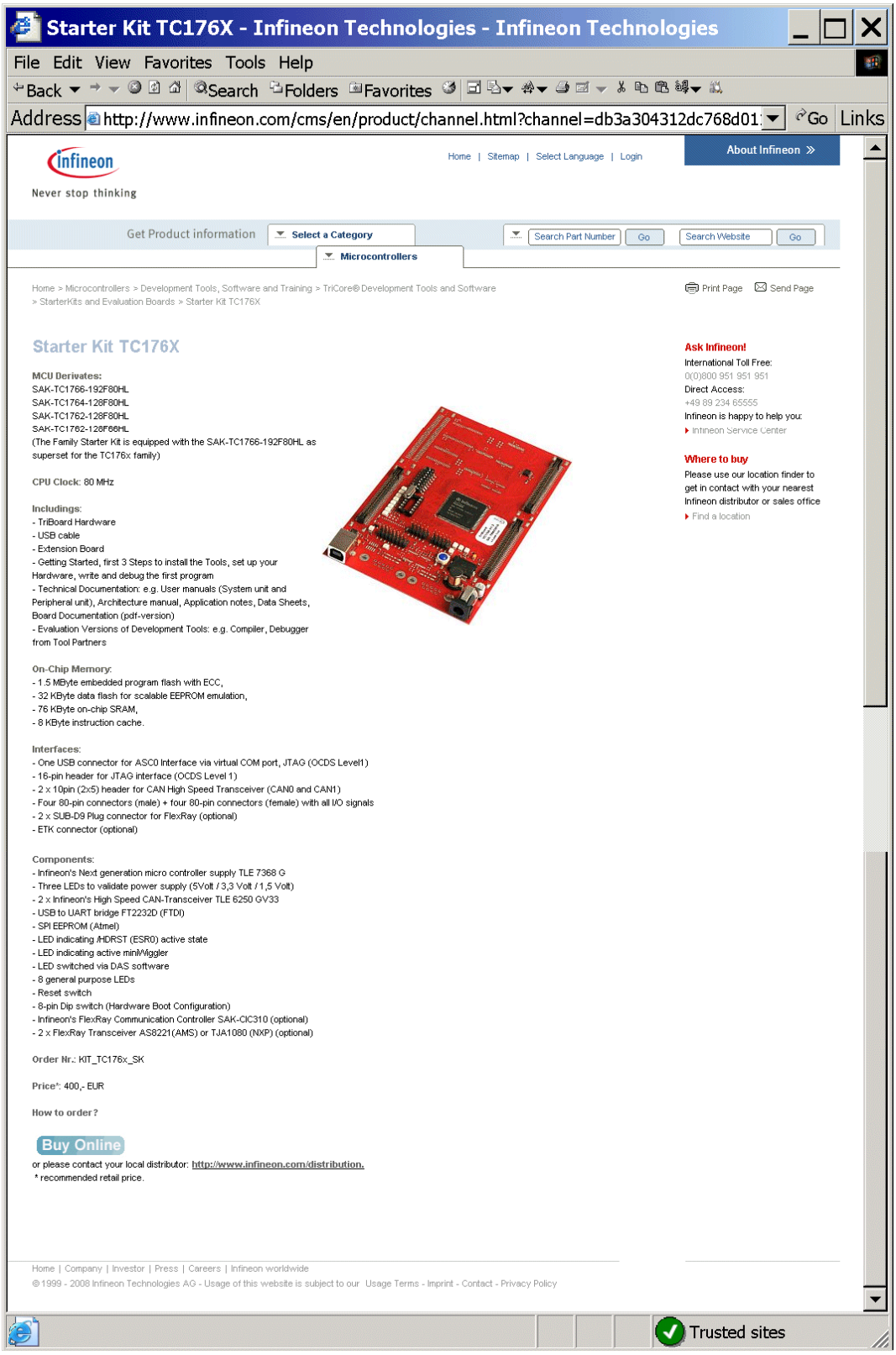
1.) TC1766 Starter Kit Board:





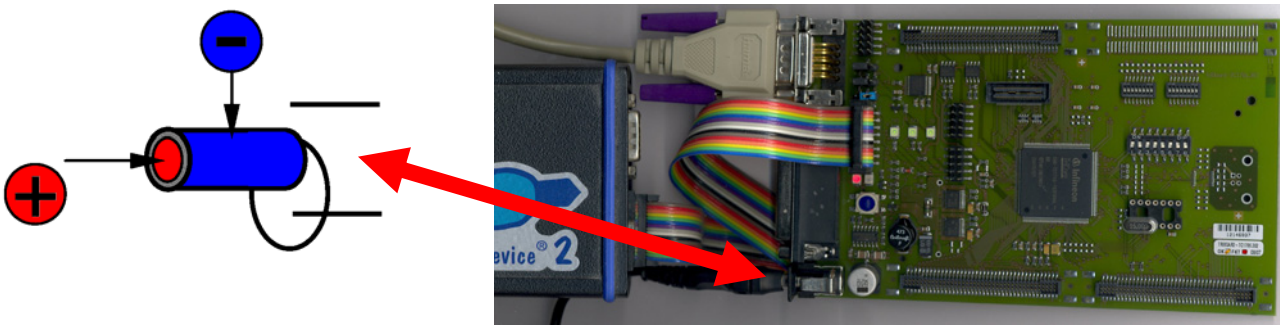
Screenshot of the TC1766 Starter Kit homepage:

http://www.infineon.com/cms/en/product/channel.html?channel=db3a304312dc768d0112e71c62150b30



Connecting the TC1766 Starter Kit:

1. Connect a Power Supply:
The TC1766 Board requires an external power supply.
A (un)regulated DC power supply from 5,5 to 60 Volts can be connected to the power connector.
500 mA are sufficient for the TC1766 Starter Kit.



2. Connect a RS-232 Serial Cable
(1:1; 9-pin Sub-D plug – 9-pin Sub-D connector; the “Hello World” example uses this interface):

3. Connect the pls-Debugger (Flash-Programming und Debugging):

For further information, please refer to the TriBoard TC176X User’s Manual,V1.0, June 2005.

Jumper Settings (Jumper JP501):

Source: TriBoard TC176X User's Manual, V1.0, June 2005

Table 5-4 Jumper for On Board Wiggler

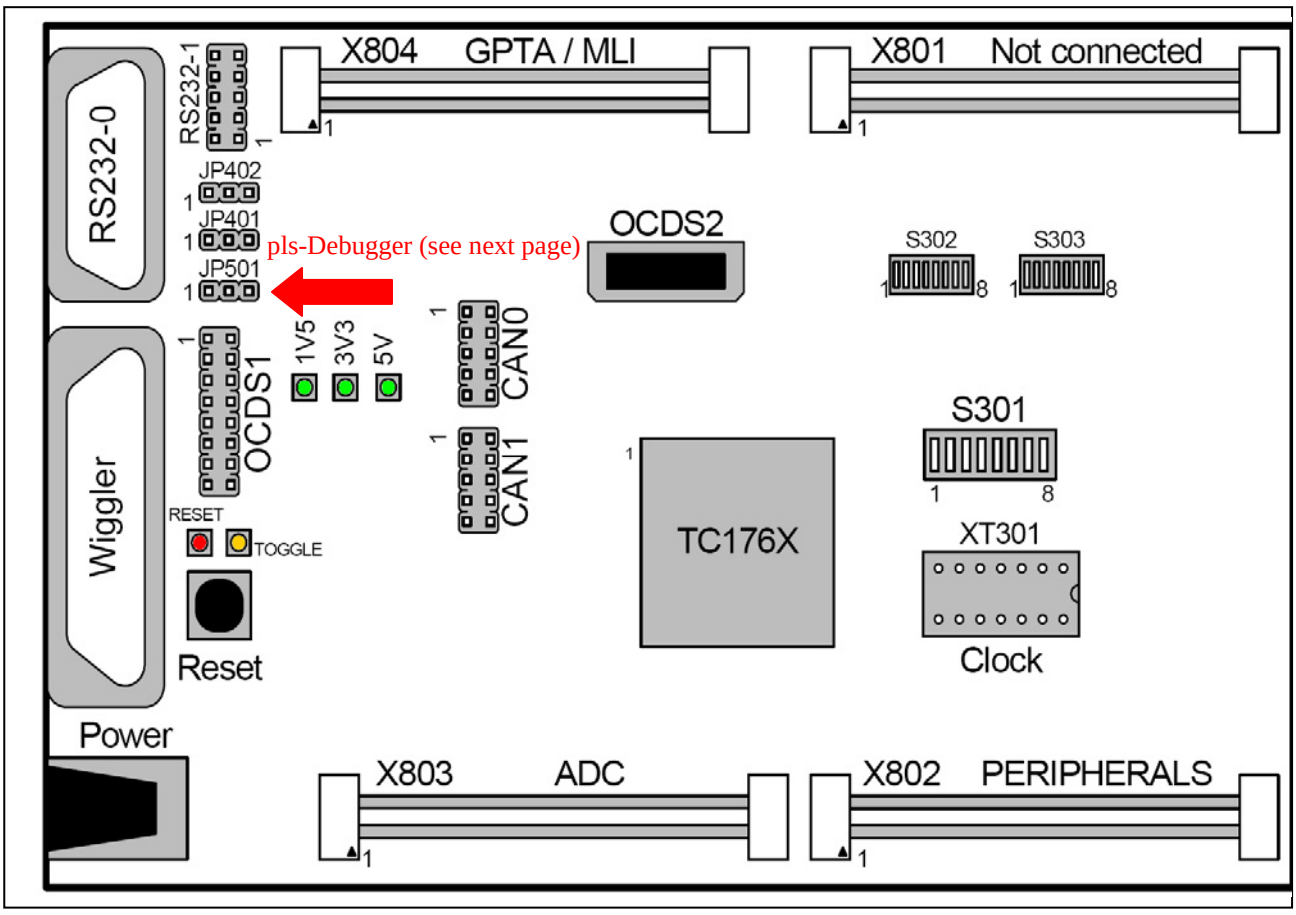
Note: The shadowed line indicates the default setting

Setting	On Board Wiggler
1 - 2	Enable On Board Wiggler
2 - 3	Disable On Board Wiggler

pls-Debugger

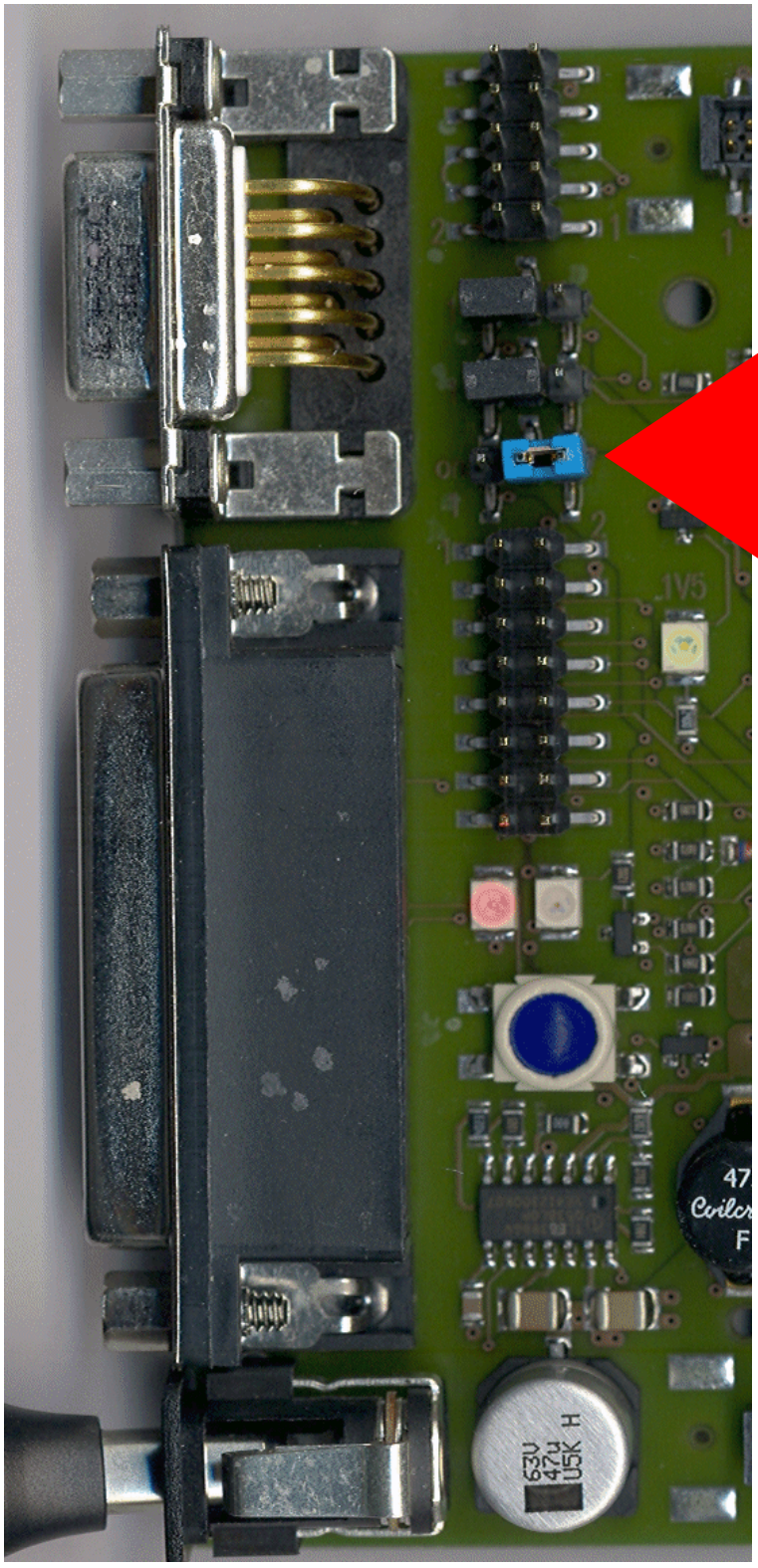
Jumper JP501
1-2 … Enable On-Board Wiggler (use parallel-on-board-interface)
2-3 … Disable On-Board Wiggler (use pls-Debugger)

pls-Debugger



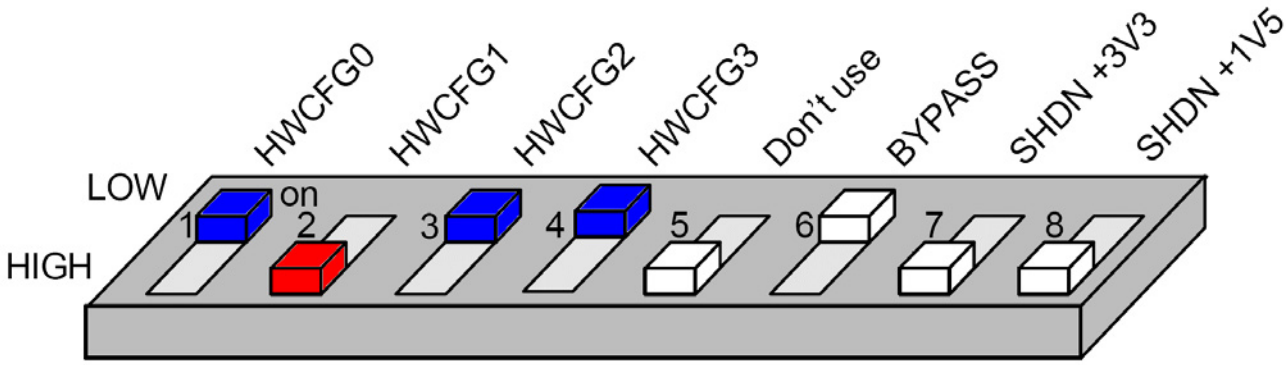
Jumper JP501
1-2 … Enable On-Board Wiggler (use parallel-on-board-interface)
2-3 … Disable On-Board Wiggler (use pls-Debugger)

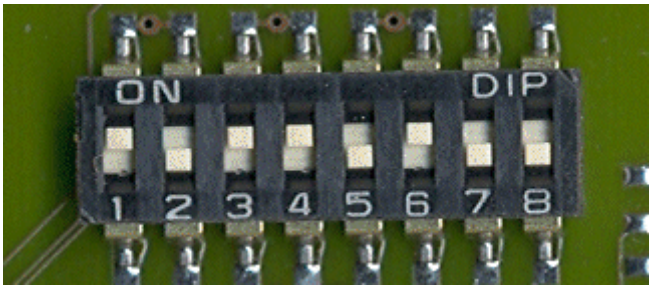
pls-Debugger



TC1766 Execution-Environment = OnChipFlash:

Jumper Settings (HW-Configuration DIP-Switch):

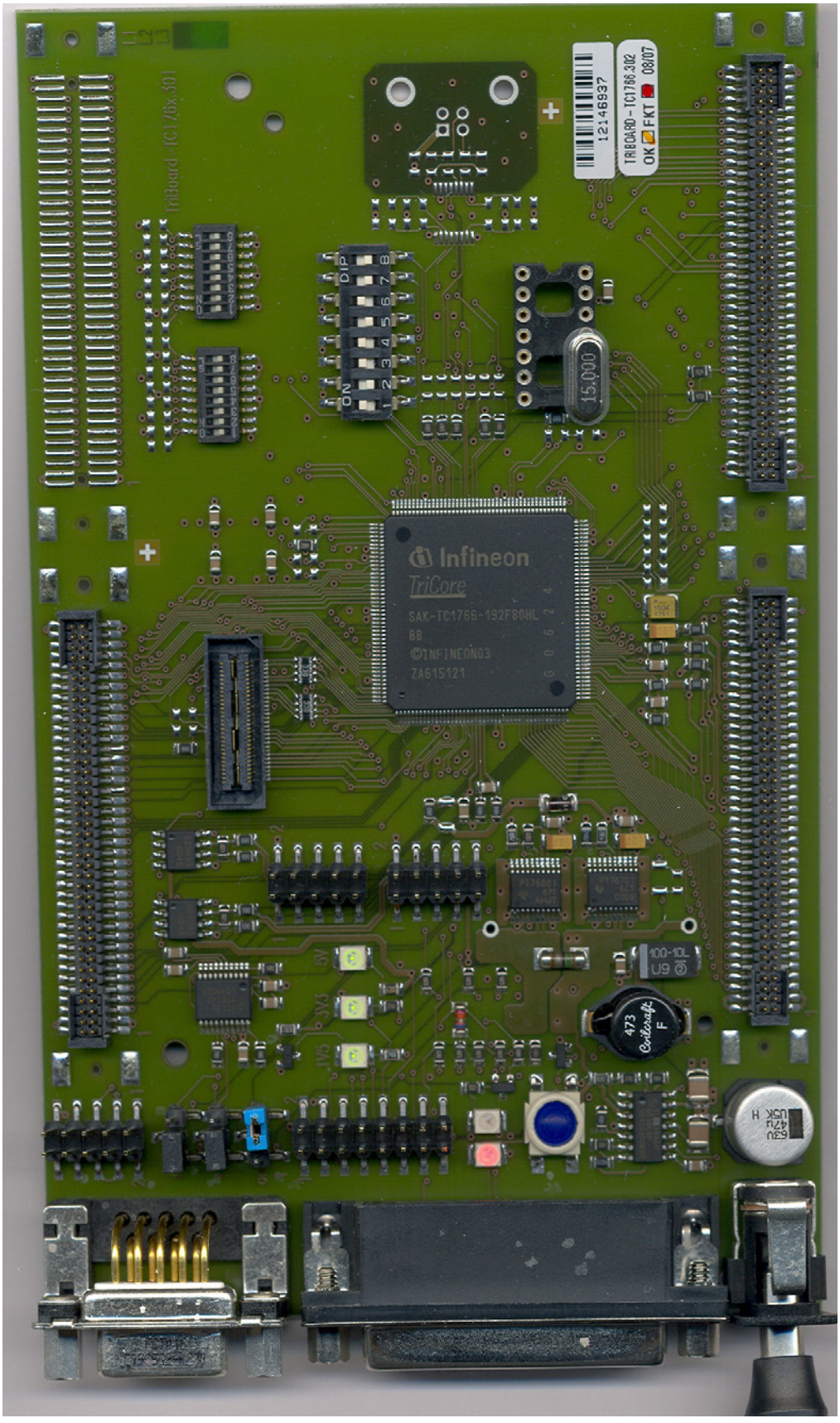




/BRKIN	HWCFG[3...0]	Type of Boot	PC Start value
1	0000	Serial boot from ASC to PMI scratchpad, run loaded program	0xD4000000
1	0001	Serial boot from CAN to PMI scratchpad, run loaded program	0xD4000000
1	0010	Start from internal flash	0xA0000000
1	0011	Alternate Bootmode from internal flash	from Header or 0xD4000000
1	1000	Internal Start in EEC SRAM, if ED	0xAFF20000
1	1111	Serial boot from ASC via CAN pins to PMI scratchpad, run loaded program	from Header or 0xD4000000
1	all others	reserved; don't use this combination	-
0	0000	put chip in tristate (deep sleep)	-
0	all others	reserved; don't use this combination	-

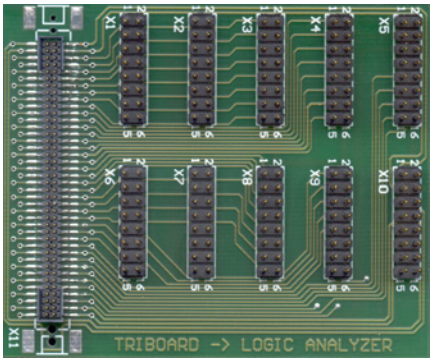
HW Configuration DIP-Switch:
1, 3, 4, 6 : ON
2, 5, 7, 8 : OFF

TC1766 Execution-Environment = OnChipFlash:



Accessories for the TC1766 Starter Kit: Extension Boards

“TriBoard+XC16x-Adapter-Board” to have access to all microcontroller pins.
Stencils are available with the Board



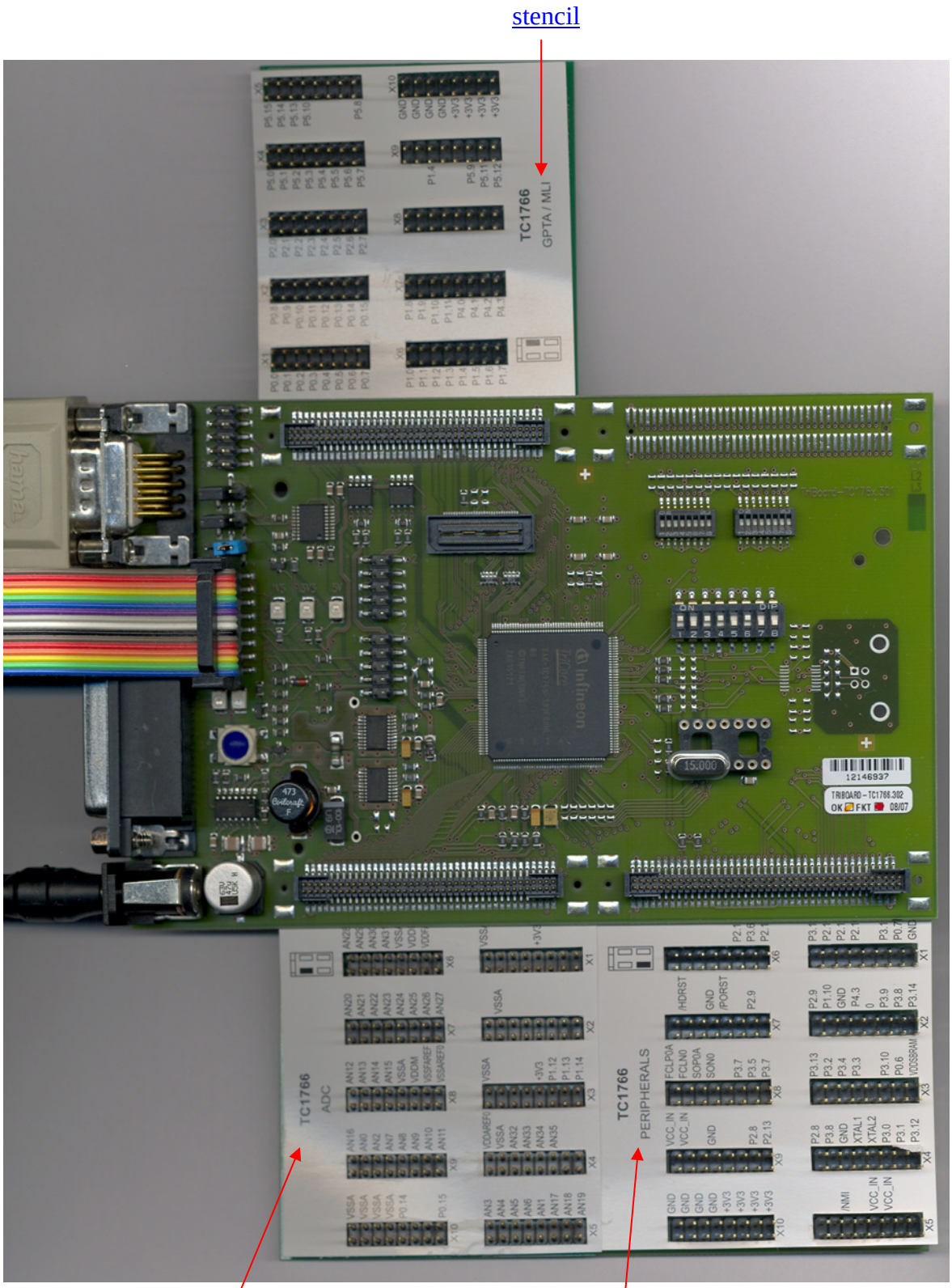
Ordering information:

Name: TriBoard+XC16x-Adapter-Platine.
The price is approximately € 32 per extension board (3 required).

Purpose: extension boards are used for easy measuring of the signals on the extension connectors to have access to all microcontroller pins and/or to connect either a part of – or the entire application to the TC1766 Starter Kit.

You can order them at:

TQ Components GmbH
Schulstraße 29a
D-82234 Weßling
Deutschland
T: +49-8153-9308-161
Mr. Rolf Müller


stencil
TC1766
GPTA / MLI
TC1766
ADC
TC1766
PERIPHERALS
stencil
stencil

2.) DAvE – Installation for TC1766 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 TC1766 microcontroller support/update (TC1766 DIP file):

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

Title	Date	Version	Size
Development Tools			
TC179x Family DIP files for DAvE (Microcontroller Configuration Tool)-latest version (TC179x_Series_v1.1.zip)	07 Jul 2008	v1.1	19 MB
TC176x DIP file for DAvE (Microcontroller Configuration Tool)-latest version (TC176x_Series_v1.0.zip)	07 Jul 2008	v1.0	11.1 MB
TC116x family DIP file for DAvE (Microcontroller Configuration Tool)-latest version (TC116x_Series.zip)	19 Jun 2006	V0.2	8.9 MB

Unzip the zip-file "TC176x_series_v1.0.zip" and save " TC176x_Series.dip "
@ e.g. D:\DAvE\TC1766\ TC176x_Series.dip.

2.)

Start DAvE - (click 

)

3.)

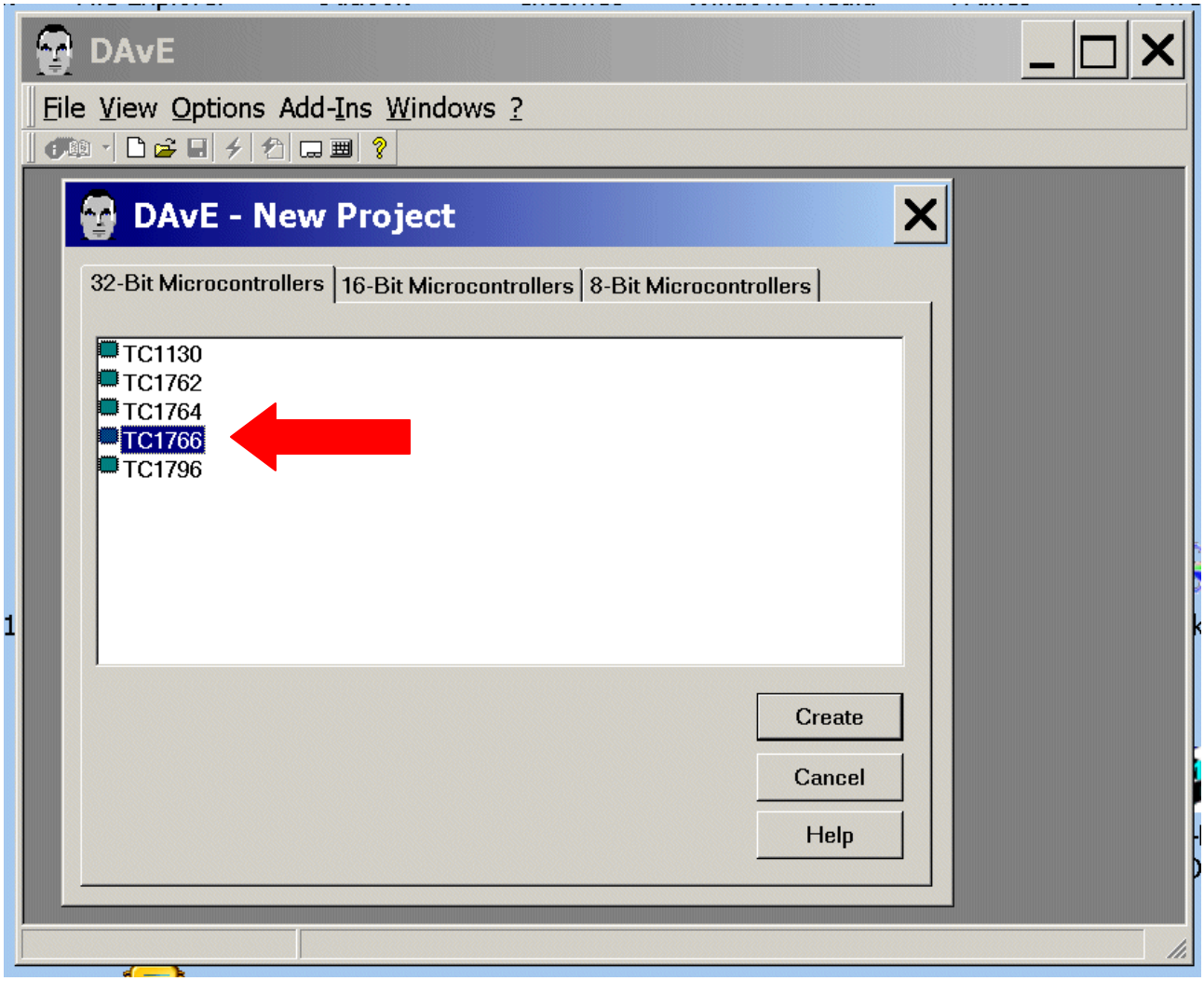
View
Setup Wizard
Default: • Installation
Forward>
Select: • I want to install products from the DAvE's web site
Forward>
Select: D:\DAvE\TC1766
Forward>
Select: Available Products
click ✔ TC176x_Series
Forward>
Install
End

4.) DAvE is now ready to generate code for the TC1766 microcontroller.

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

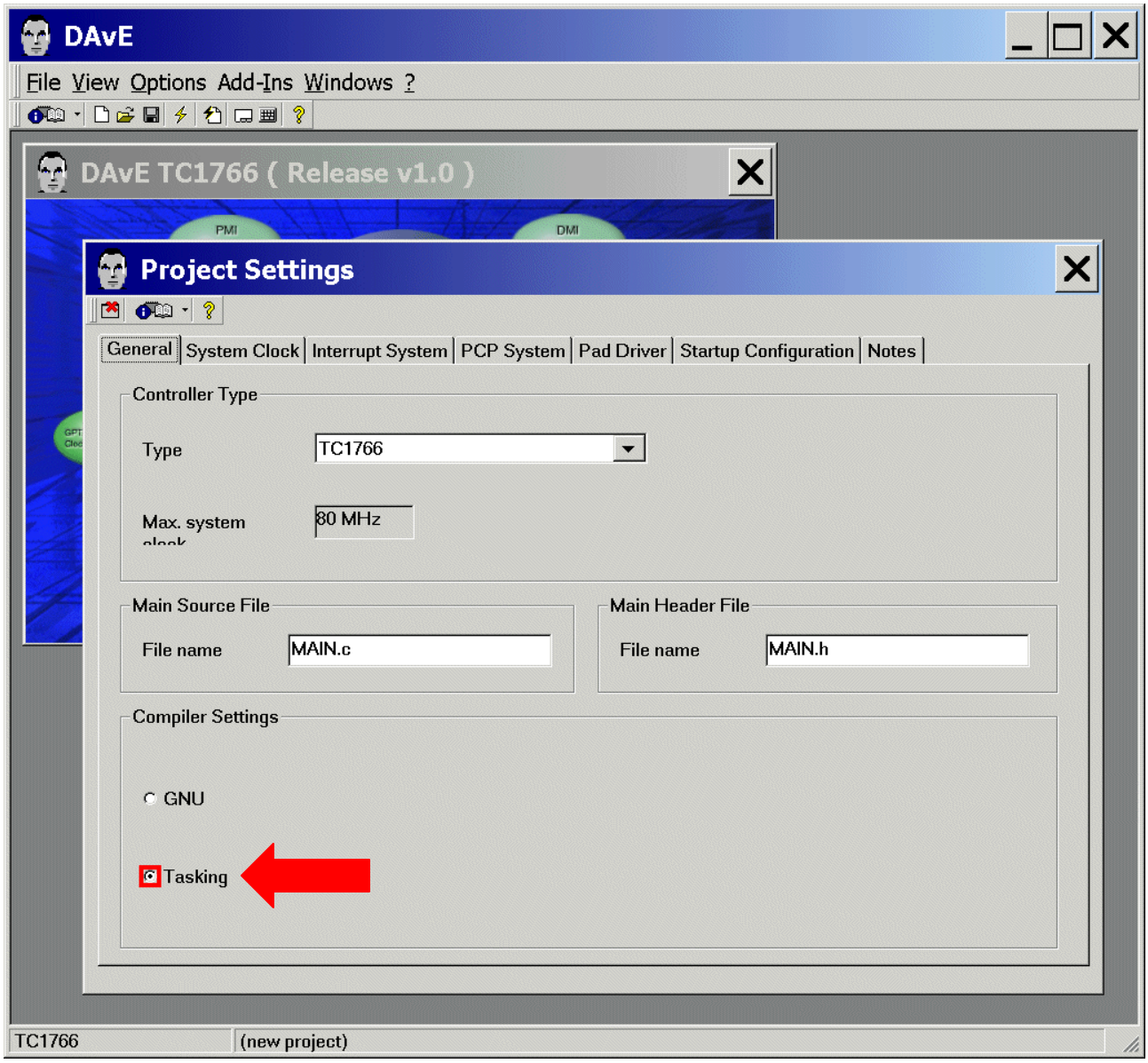
Start the program generator DAvE and select the TC1766 microcontroller:

File
New
32-Bit Microcontrollers
TC1766
Create



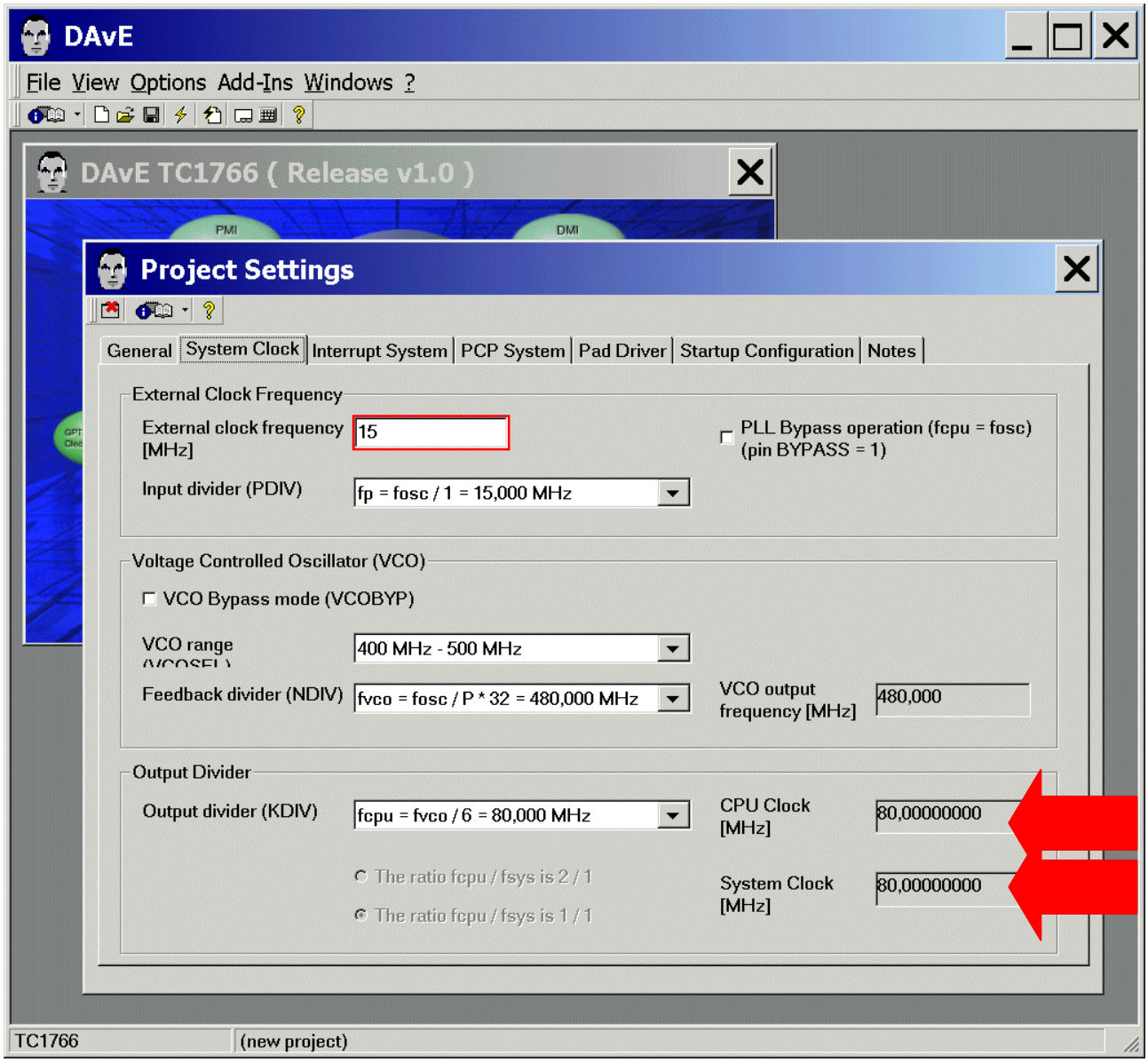
Choose the Project Settings as you can see in the screenshots:

General: Compiler Settings:
For the Tasking Compiler check/choose ⊙ Tasking in the Compiler Settings:



System Clock: CPU Clock will be 80 MHz:

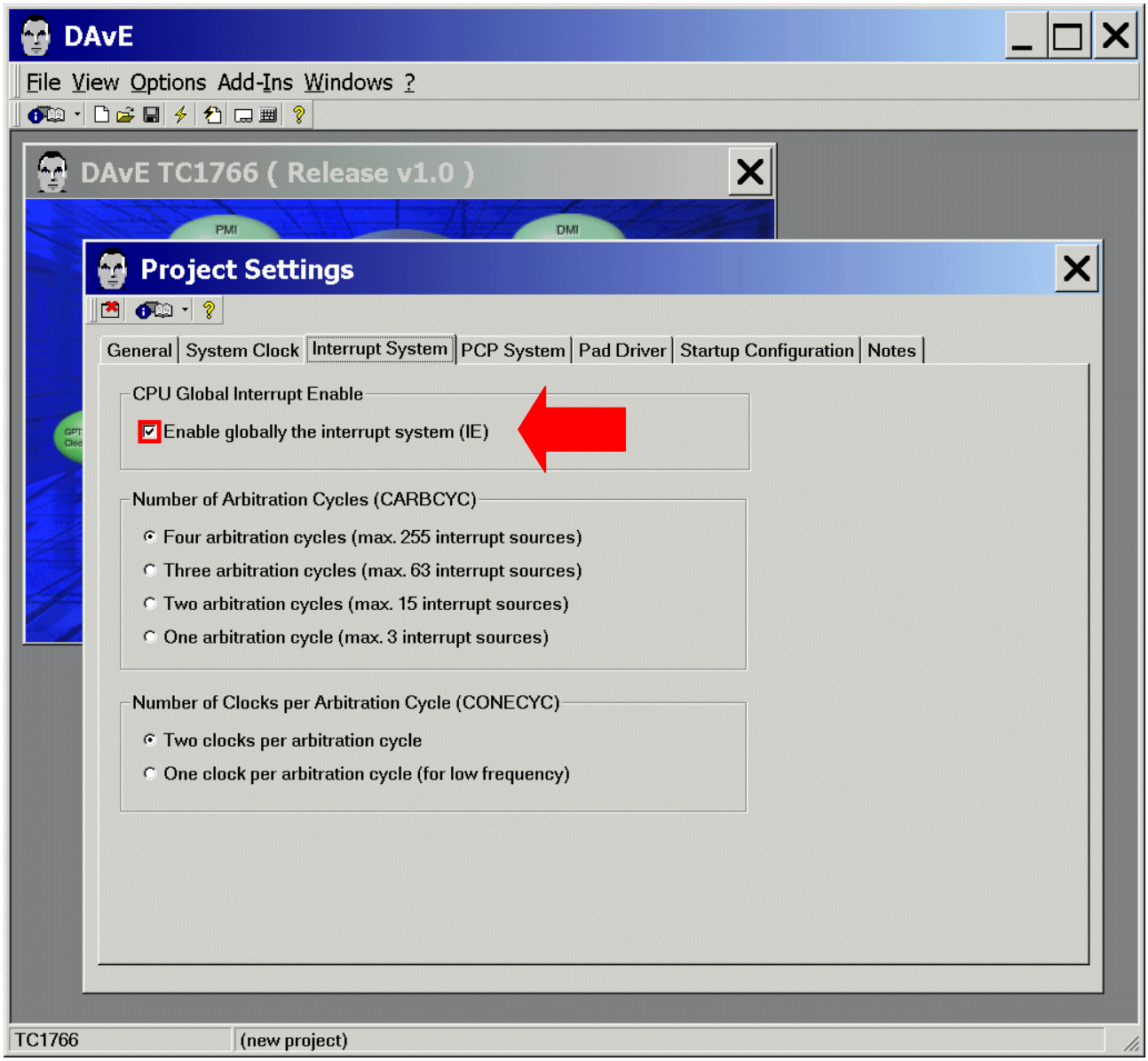
System Clock: External Clock Frequency: External clock frequency check/insert 15 [MHz]



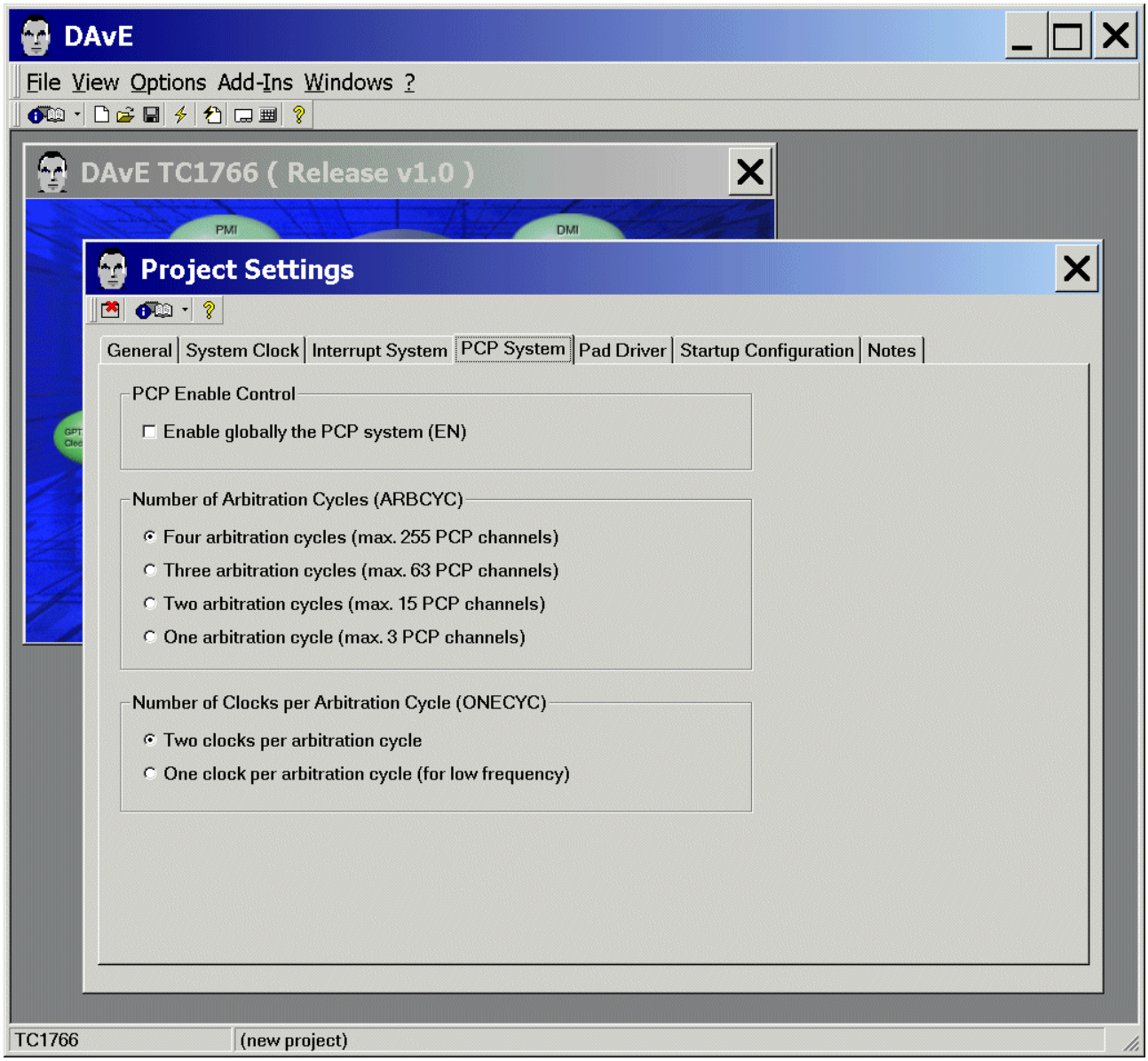
Note:
We strongly suggest that you check first to see if your board is equipped with a 15 MHz Crystal (default).

Note:
The final result should be 80 MHz CPU Clock and 80 MHz System Clock

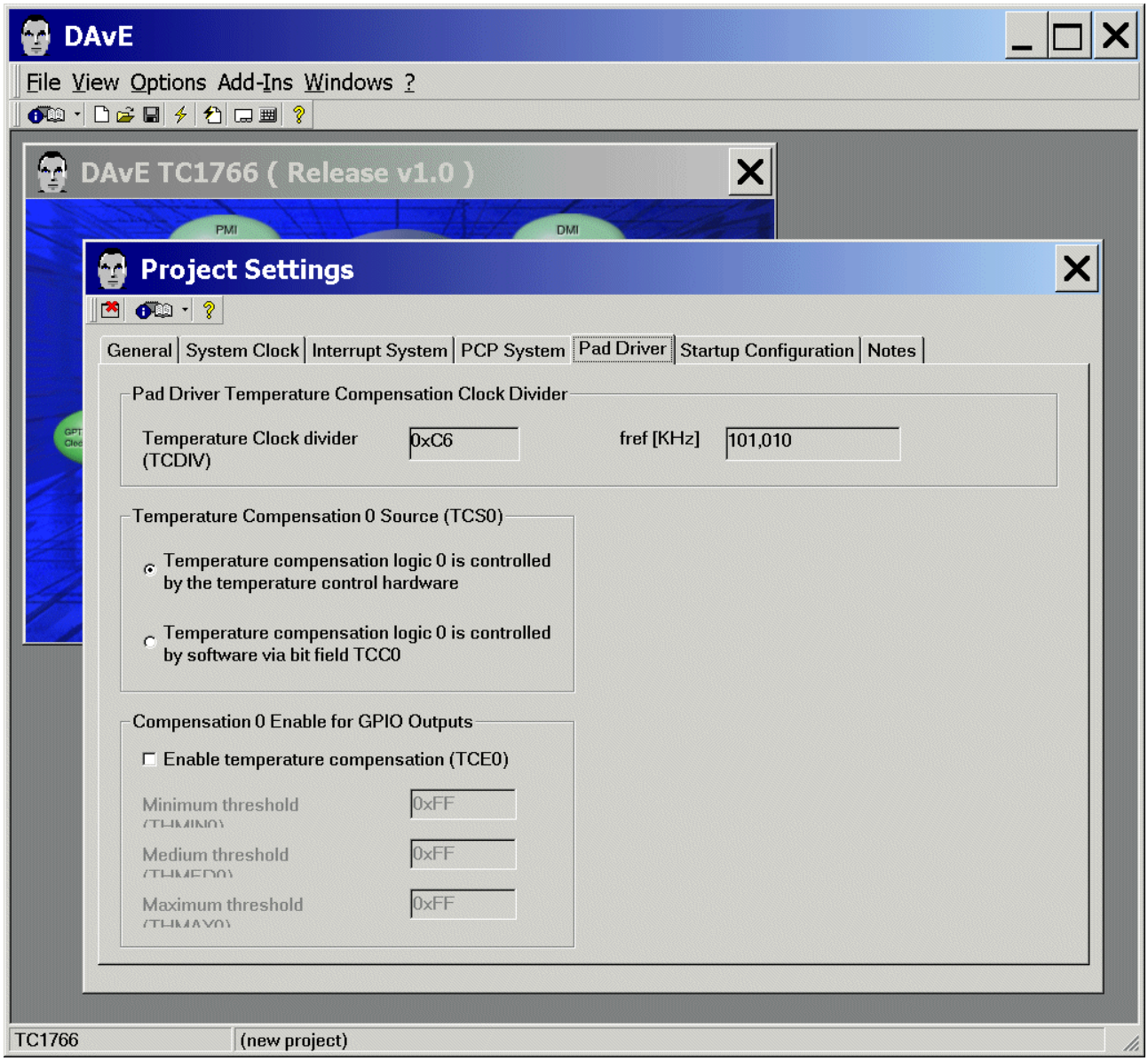
Interrupt System: CPU Global Interrupt Enable: tick ✓ Enable globally the interrupt system (IE)



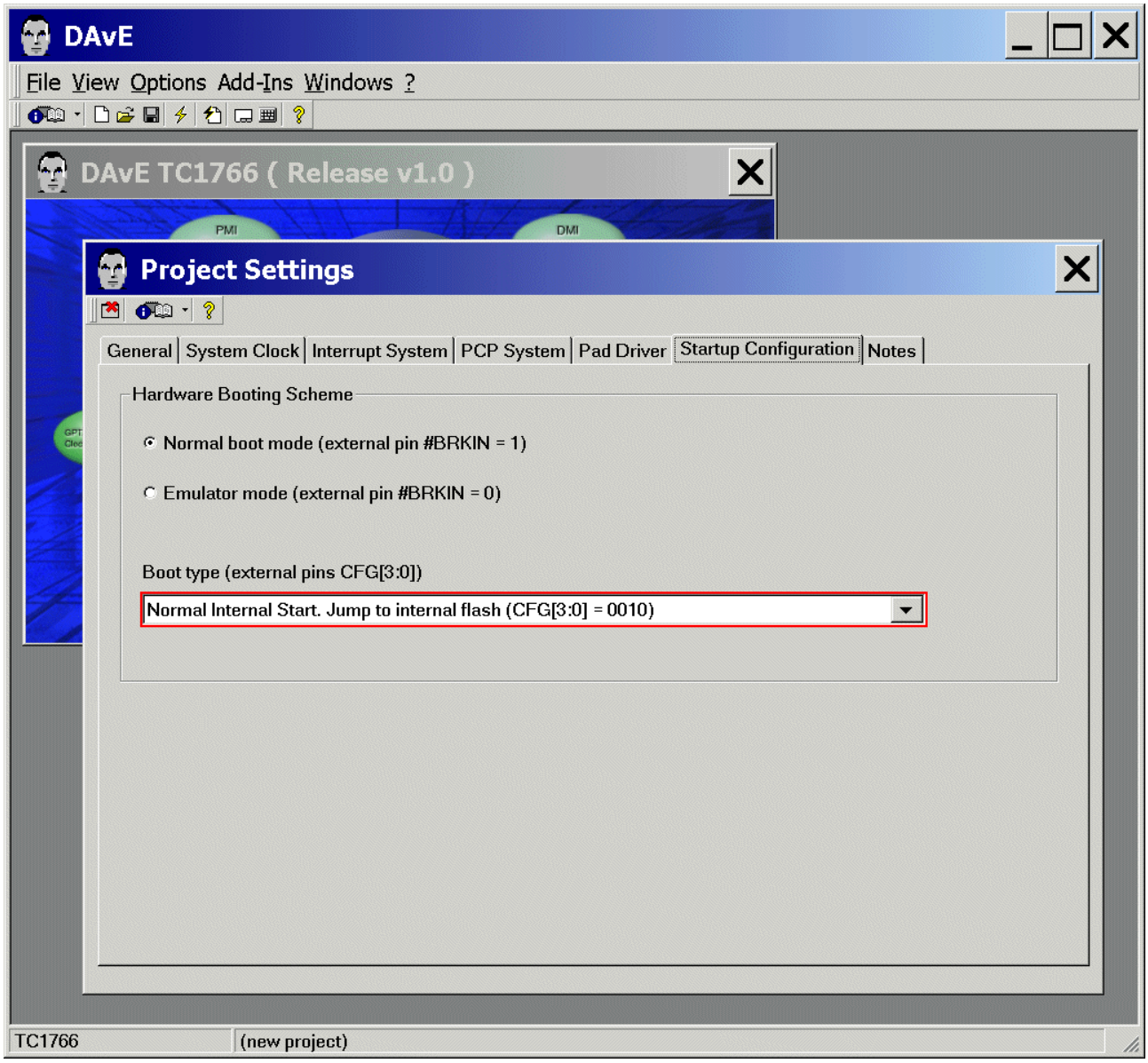
PCP System: (do nothing)



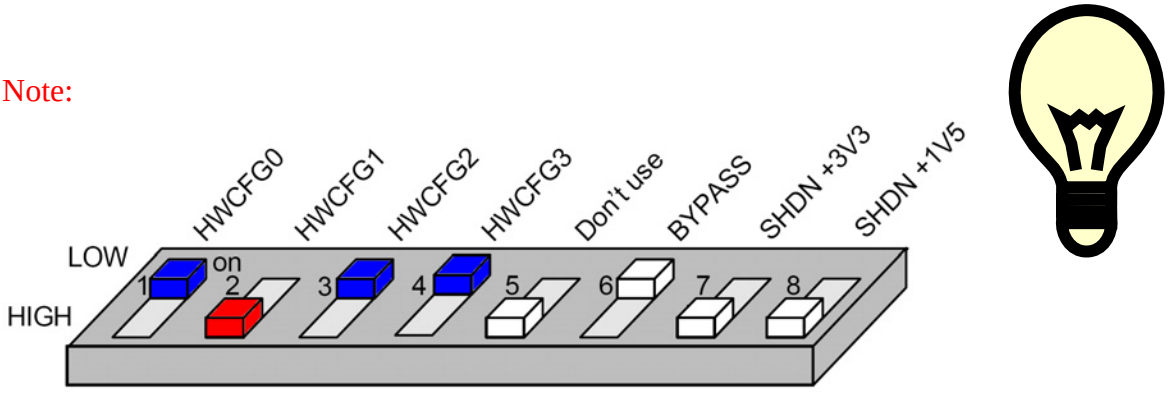
Pad Driver: (do nothing)



Startup Configuration: Hardware Booting Scheme: Boot type (external pins CFG[3:0]) select Normal Internal Start. Jump to internal flash (CFG[3:0] = 0010)

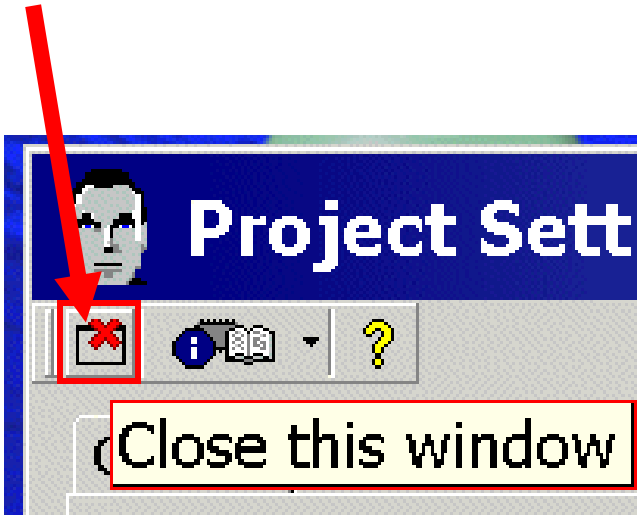


Note:



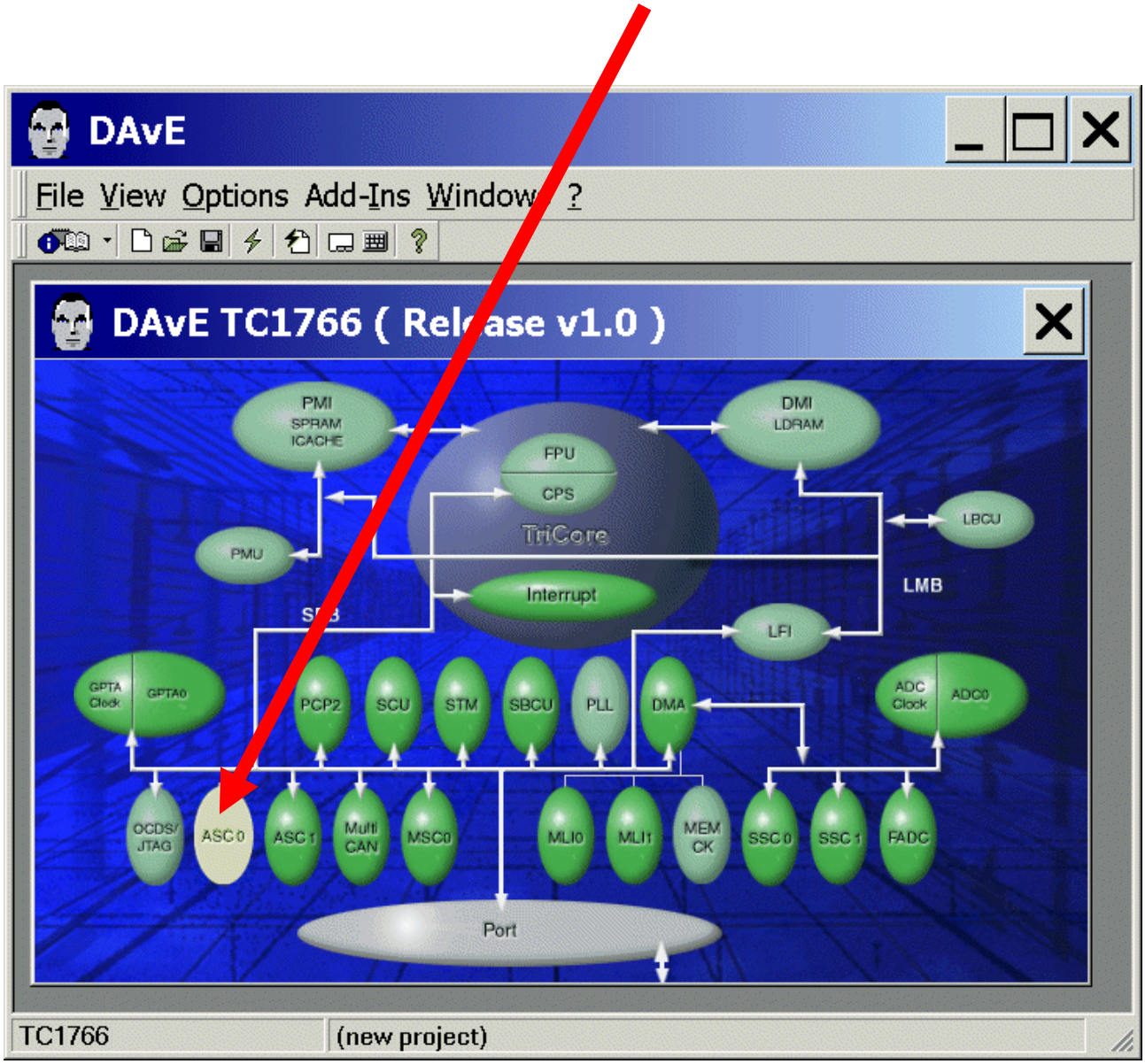
Notes: If you wish, you can insert your comments here.

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



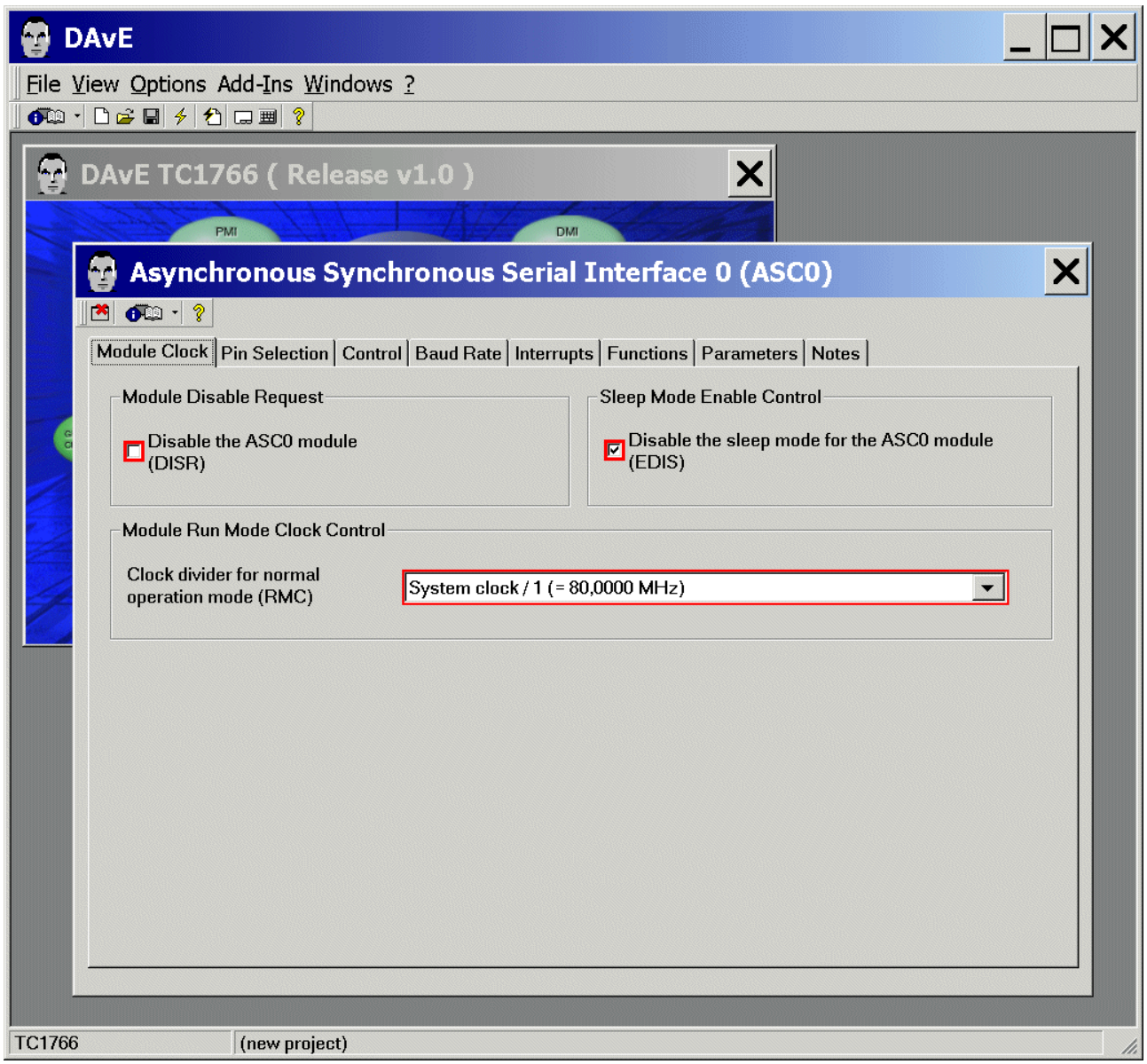
Configuration of the ASC0:

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

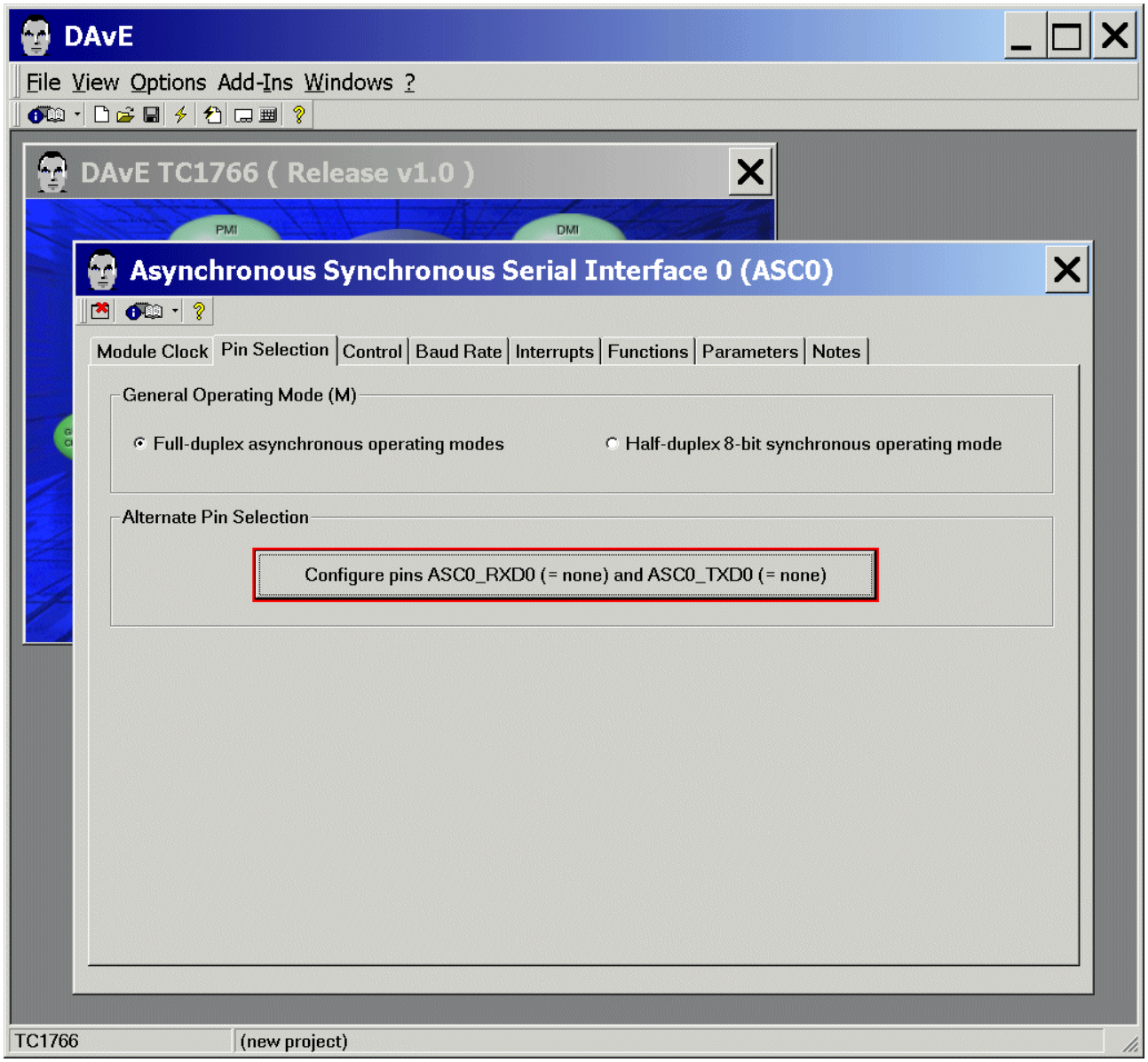


Note:
ASC0 is used for the serial communication with a terminal program running on your host computer.

Module Clock: Module Disable Request: untick ☐ Disable the ASC0 module
Module Clock: Module Run Mode Clock Control: choose System clock/1 (=80,0000 MHz)
Module Clock: Sleep Mode Enable Control: tick ✓ Disable the sleep mode

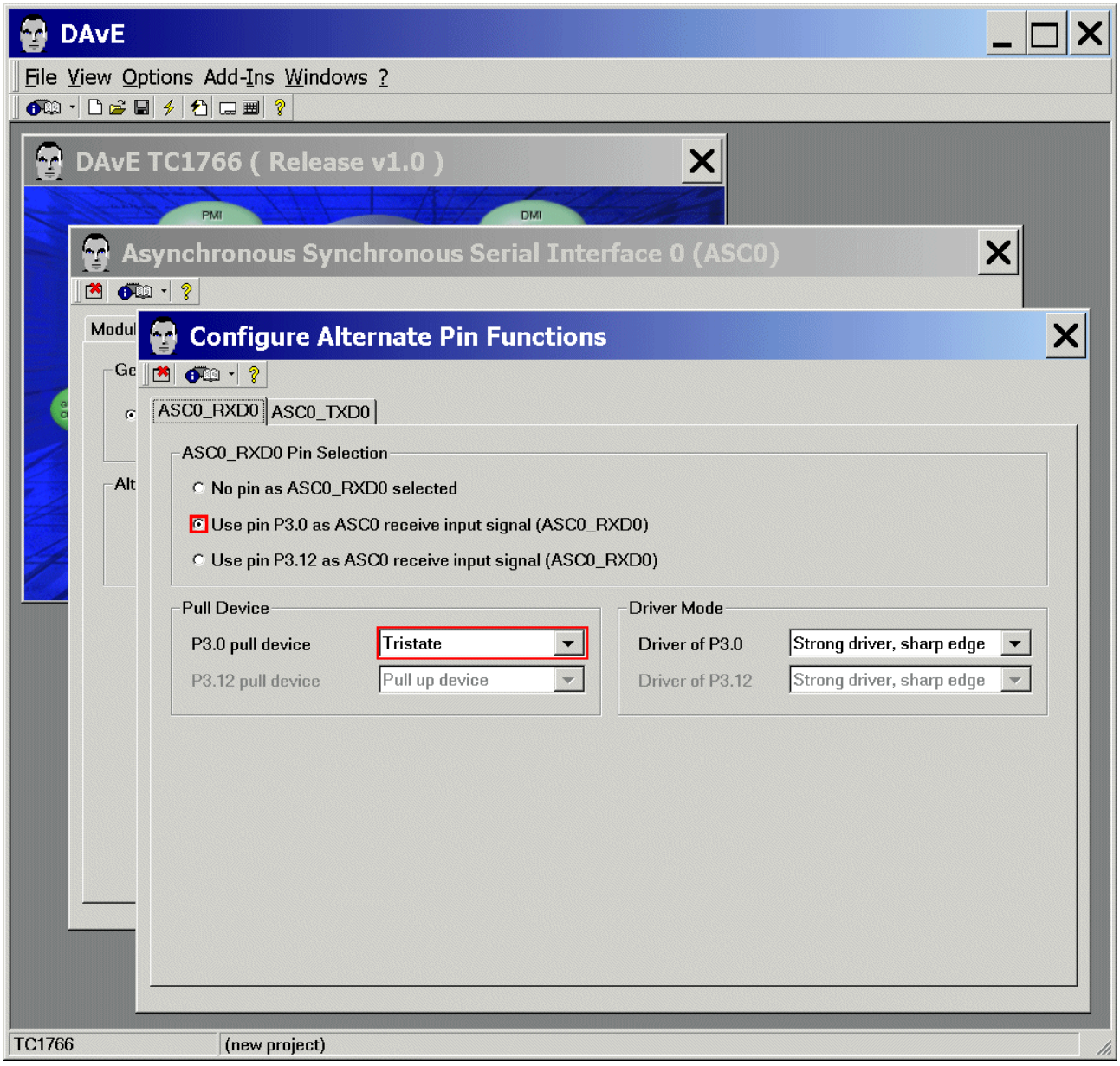


Pin Selection: Alternate Pin Selection: click Configure pins ASC0_RXD0 and ASC0_TXD0

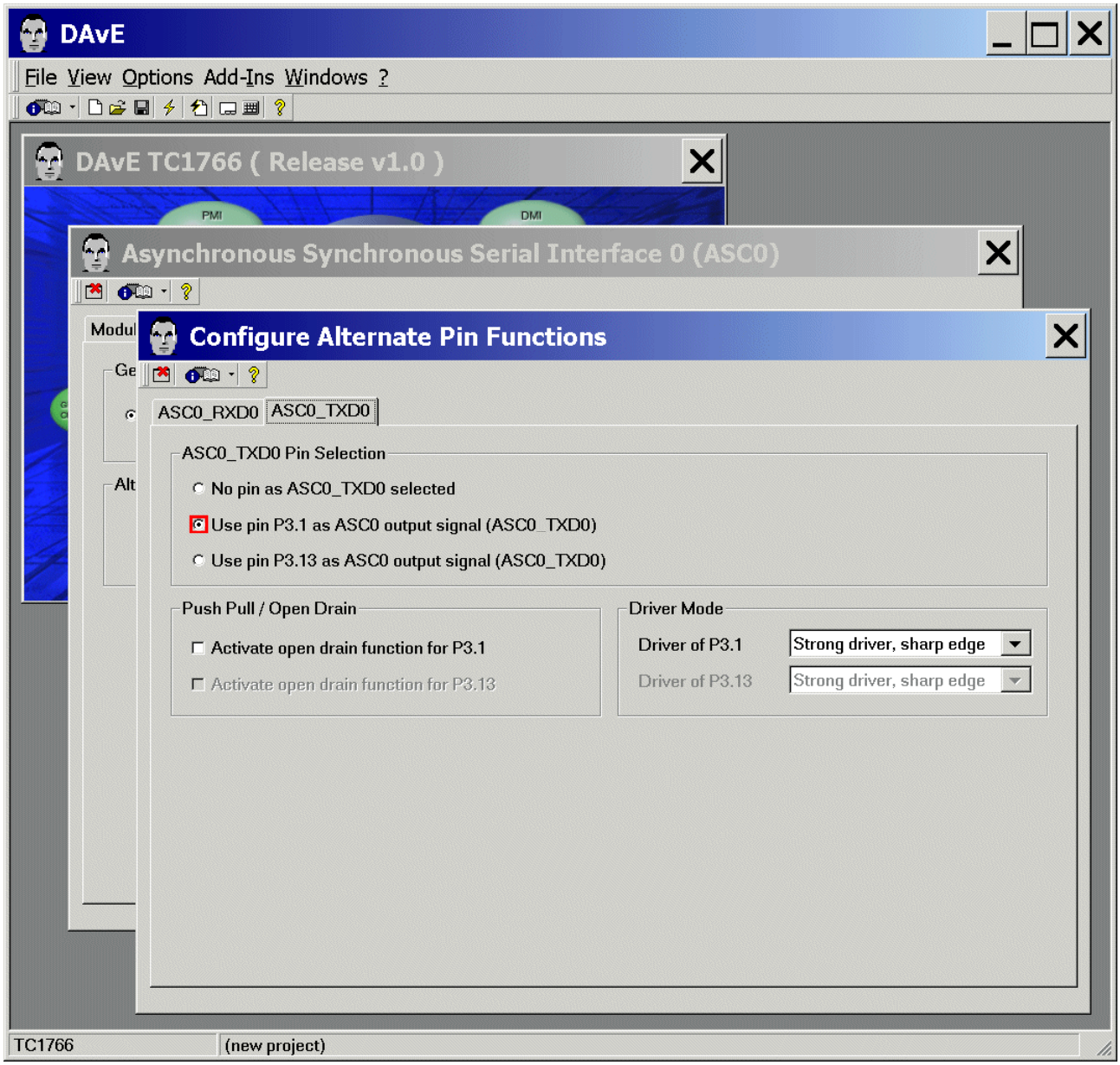


Pin Selection: Alternate Pin Selection: Configure pins ASC0_RXD0 and ASC0_TXD0: ASC0_RXD0: ASC0_RXD0 Pin Selection: click ⊙ Use pin P3.0 as ASC0 receive input signal

Pin Selection: Alternate Pin Selection: Configure pins ASC0_RXD0 and ASC0_TXD0: ASC0_RXD0: Pull Device: P3.0 pull device: select Tristate

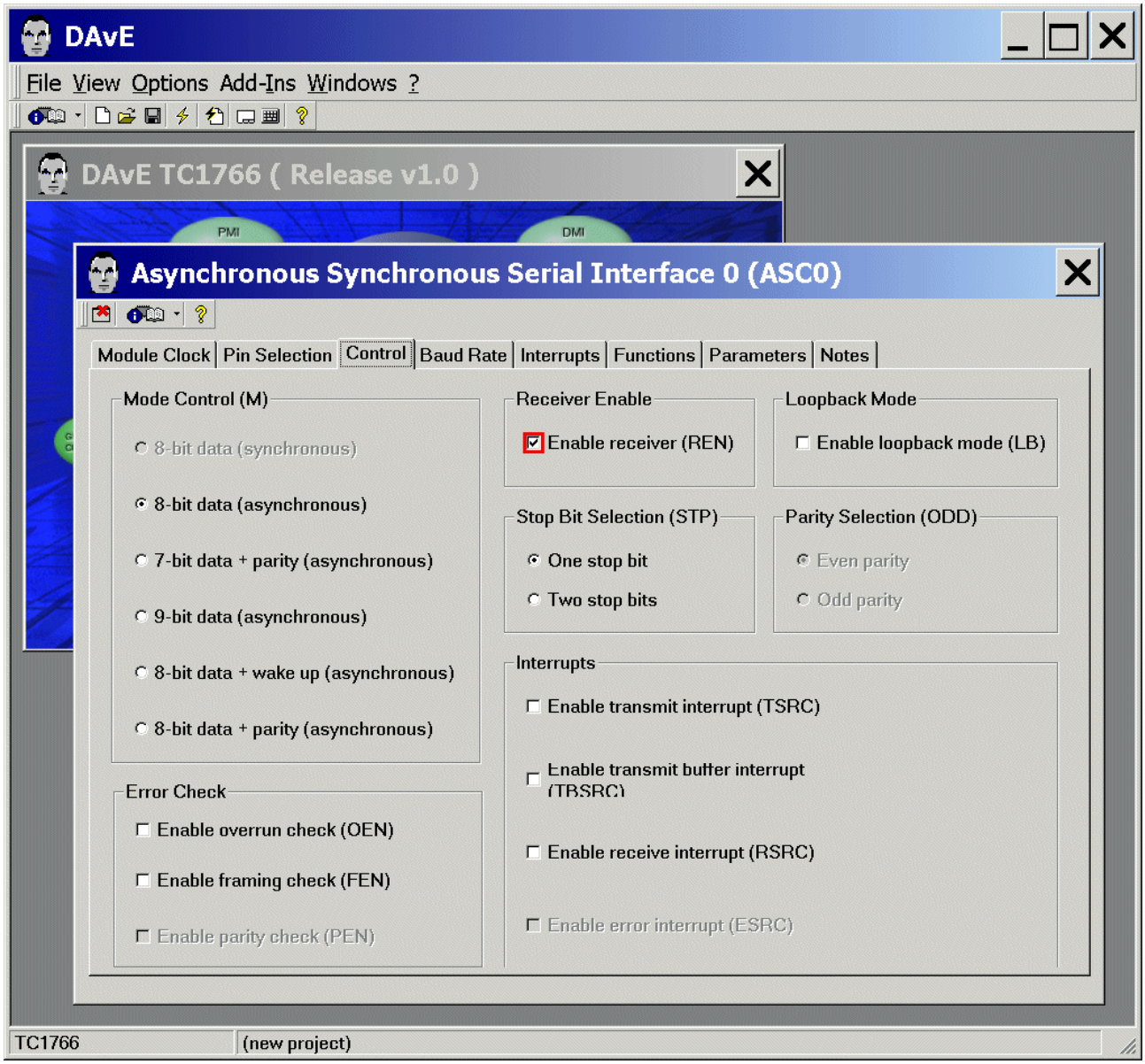


Pin Selection: Alternate Pin Selection: Configure pins ASC0_RXD0 and ASC0_TXD0: ASC0_TXD0: ASC0_TXD0 Pin Selection: click ⊙ Use pin P3.1 as ASC0 output signal

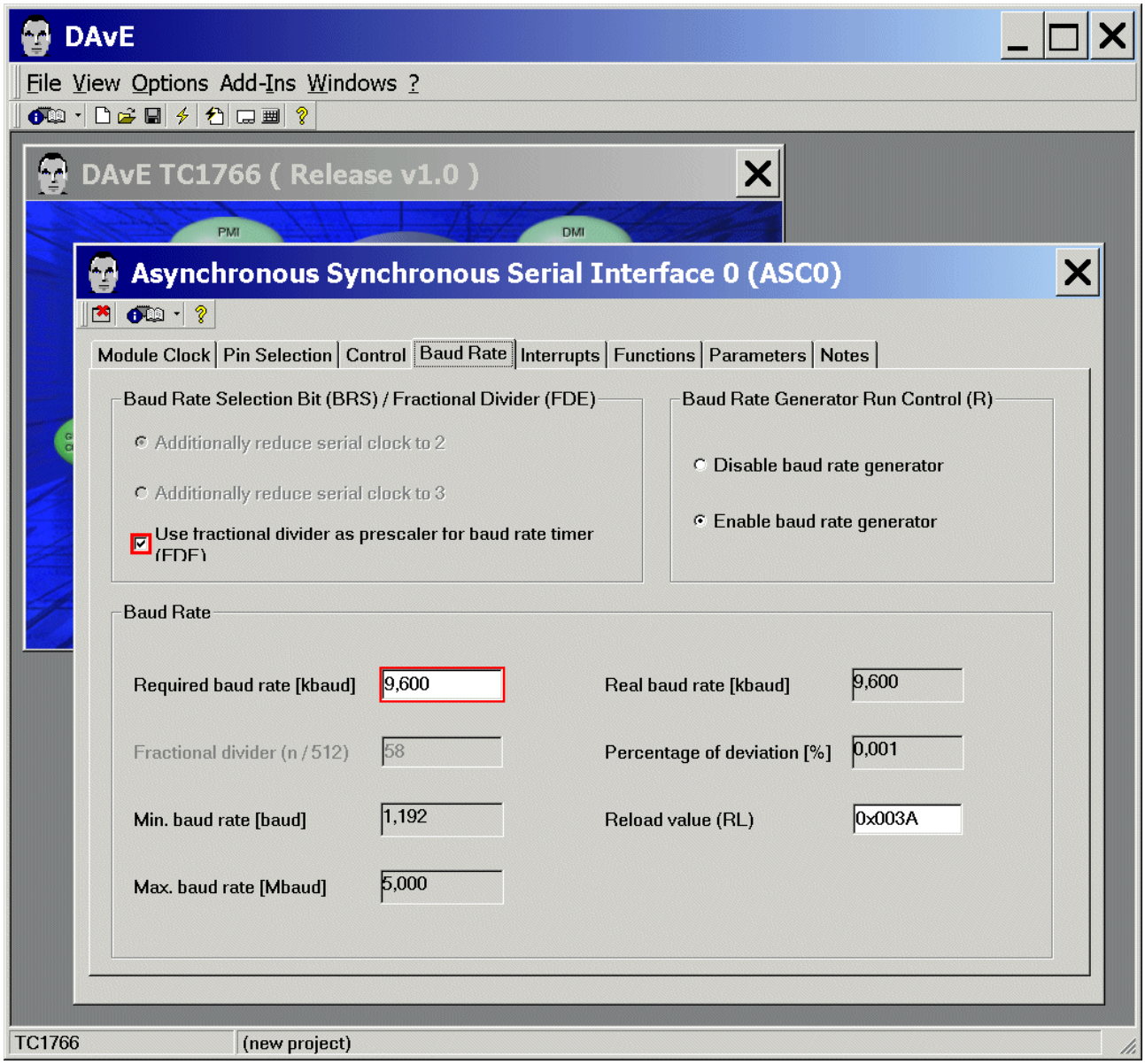


Exit and Save this dialog now by clicking the close button.

Control: Receiver Enable: tick ✓ Enable receiver (REN)

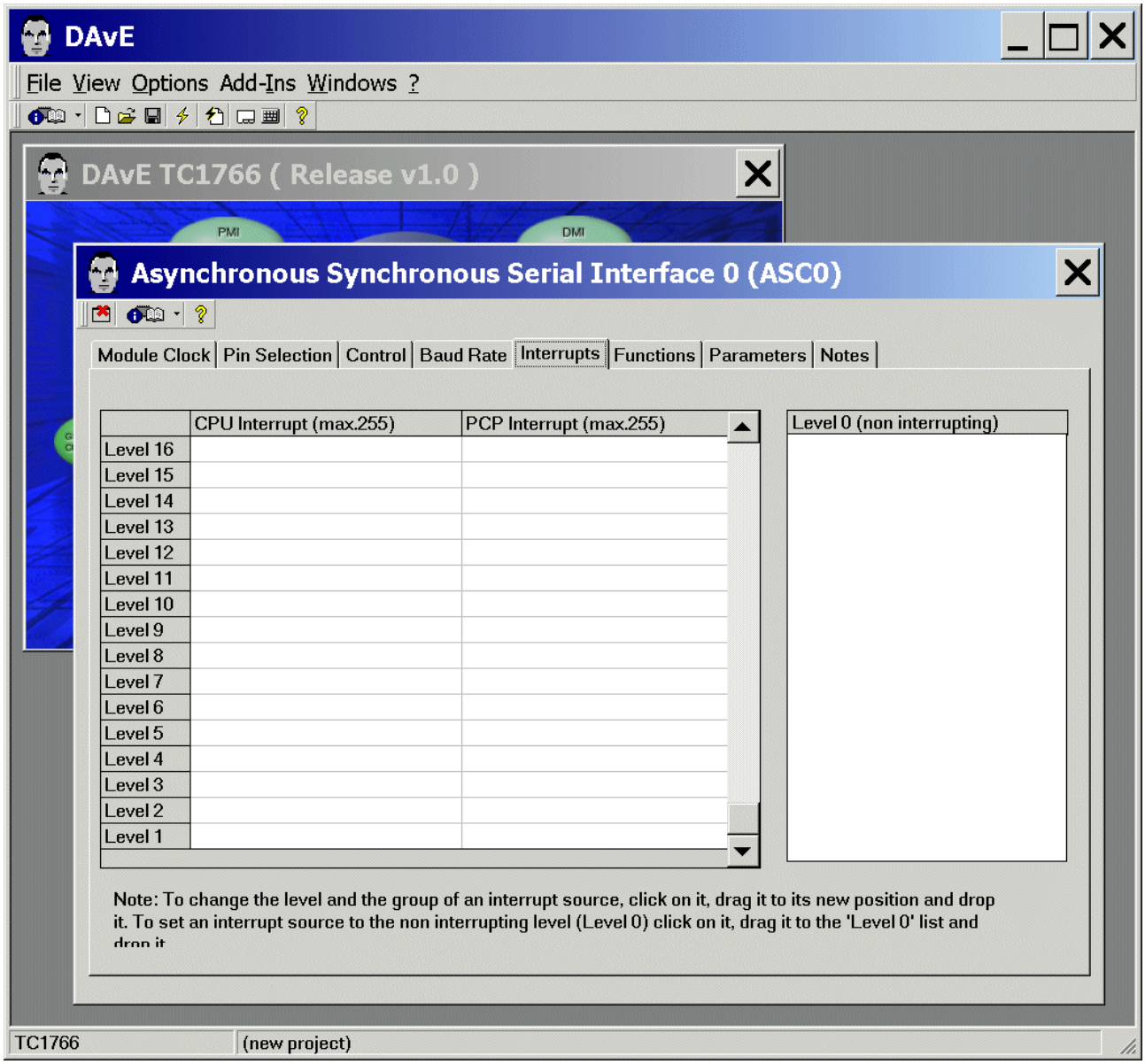


Baud Rate: Baud Rate: Required baud rate [kBaud] insert 9,600 < ENTER >
Baud Rate: Baud Rate Selection Bit / Fractional Divider: tick ✓ Use fractional divider



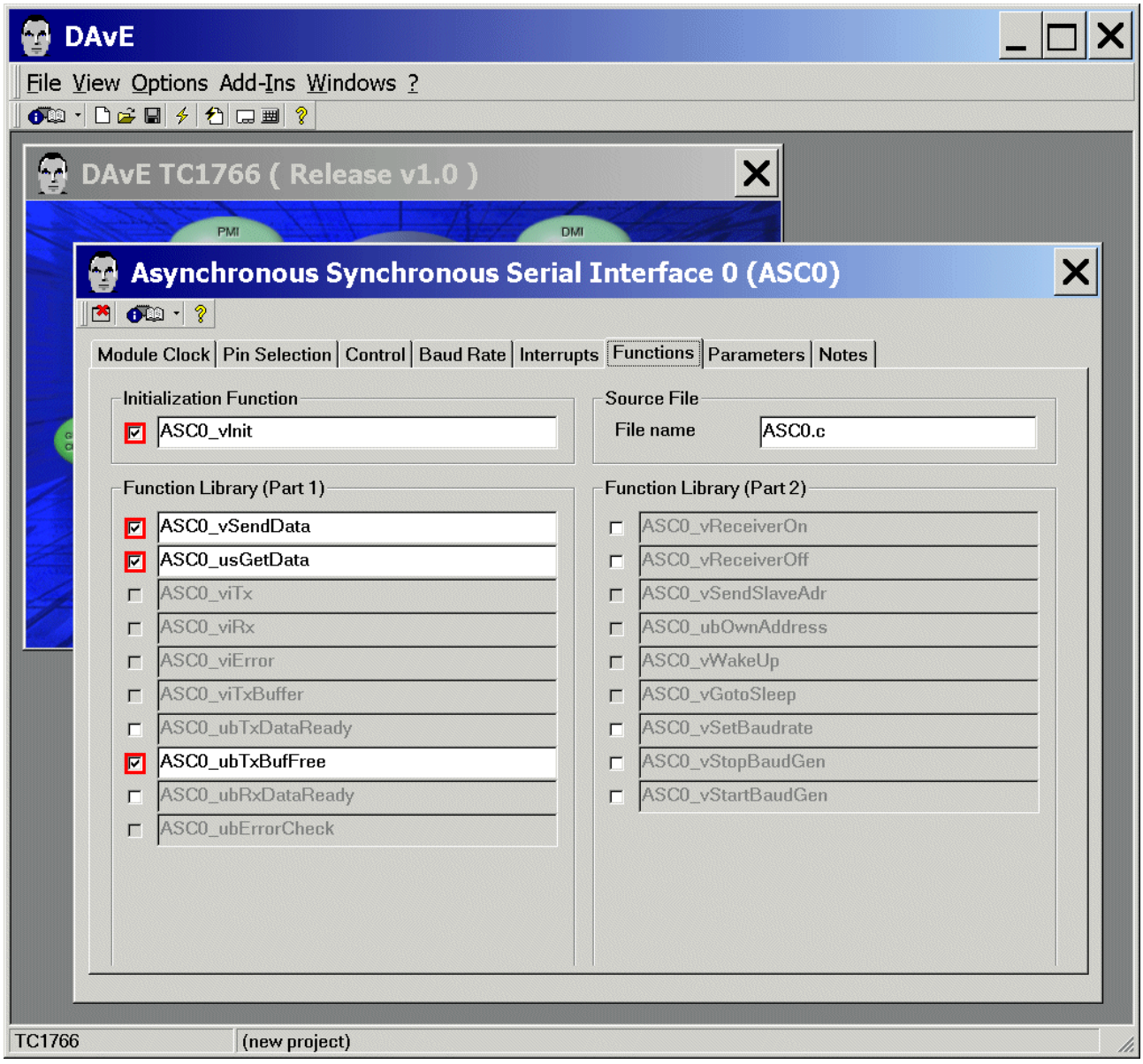
Note:
Validate each alpha numeric entry by pressing ENTER.

Interrupts: (do nothing)



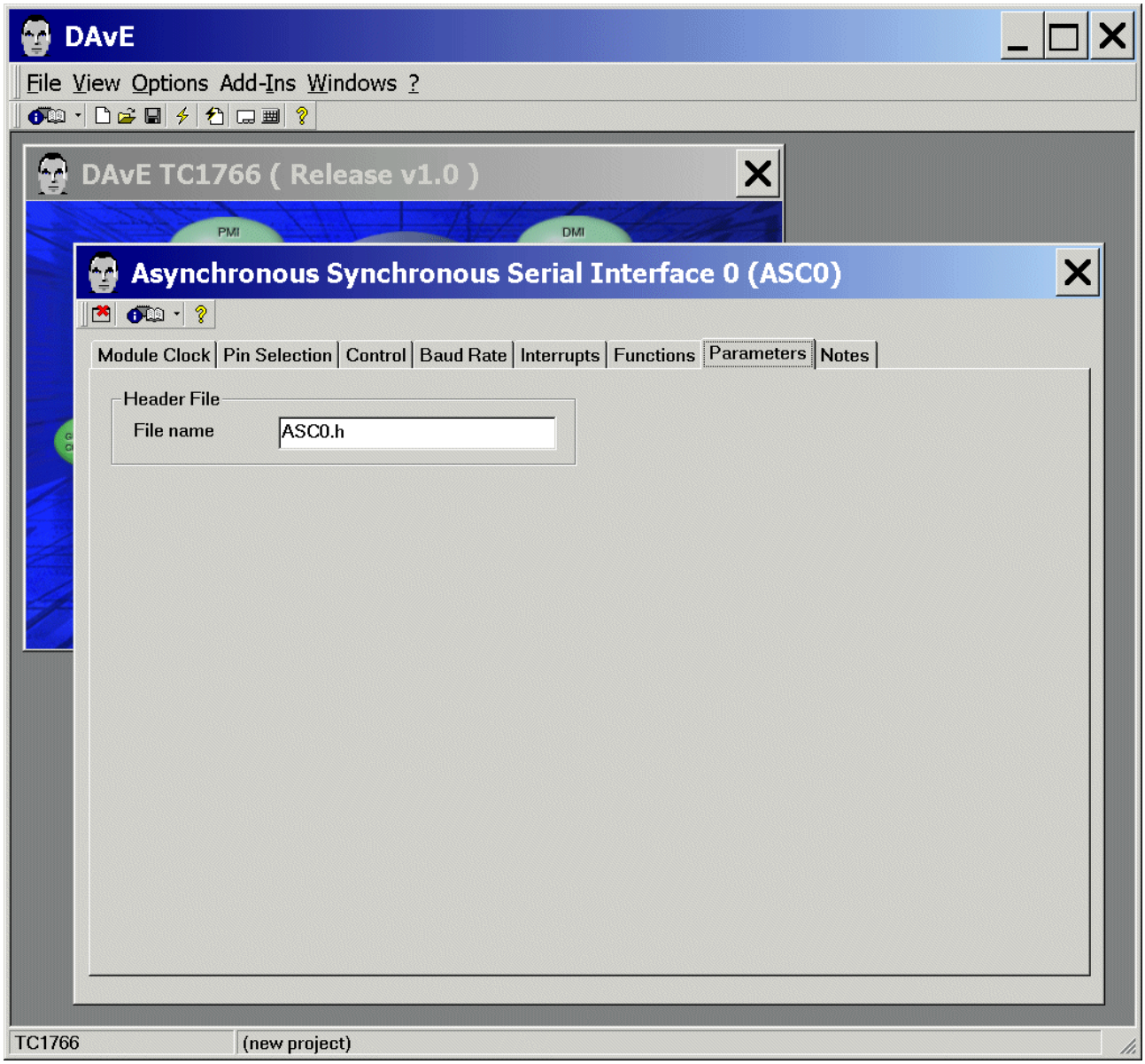
Note:
For the serial communication with a terminal program running on your host computer the myprintf function is used. The myprintf function uses Software-Polling-Mode therefore we do not need to configure any interrupts for this task.

Functions: Initialization Function: tick ✓ ASC0_vInit
Functions: Function Library (Part 1): tick ✓ ASC0_vSendData
Functions: Function Library (Part 1): tick ✓ ASC0_usGetData
Functions: Function Library (Part 1): tick ✓ ASC0_ubTxBufFree



Note:
You can change function names (e.g. ASC0_vInit) and file names (e.g. ASC0.c) anytime.

Parameters: (do nothing)

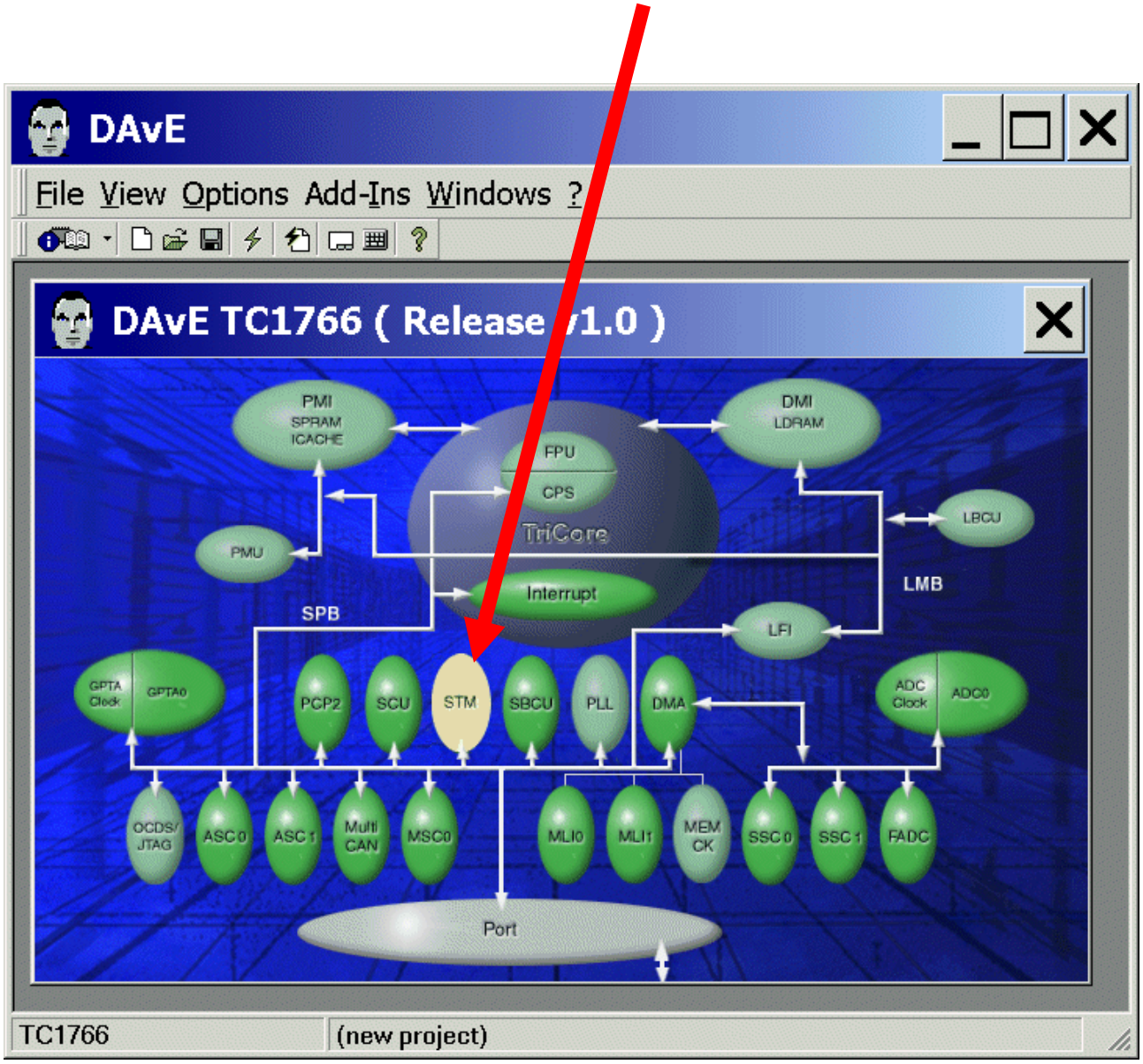


Notes: If you wish, you can insert your comments here.

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

Configuration of the STM:

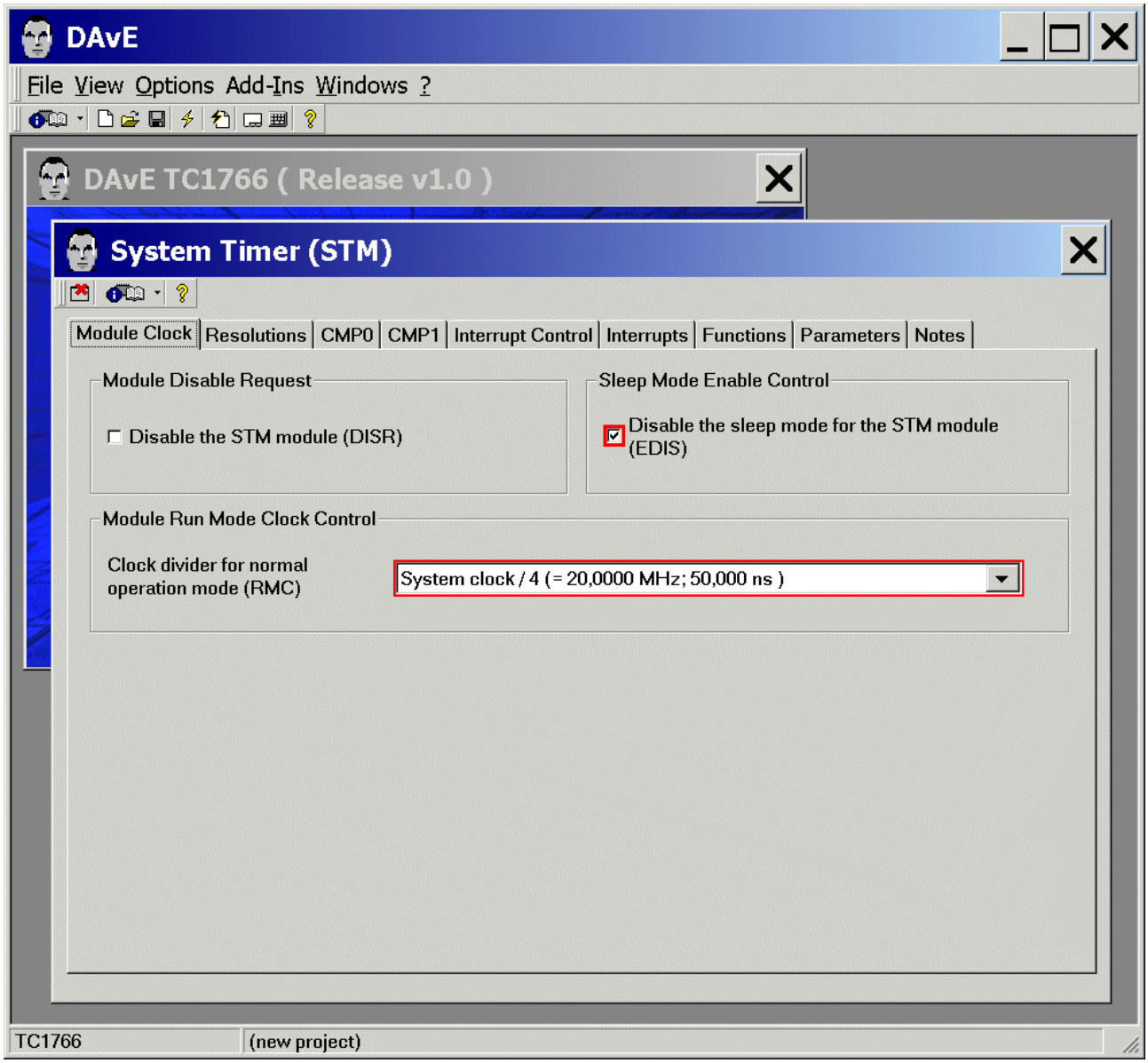
The configuration window/dialog can be opened by clicking the specific block/module.



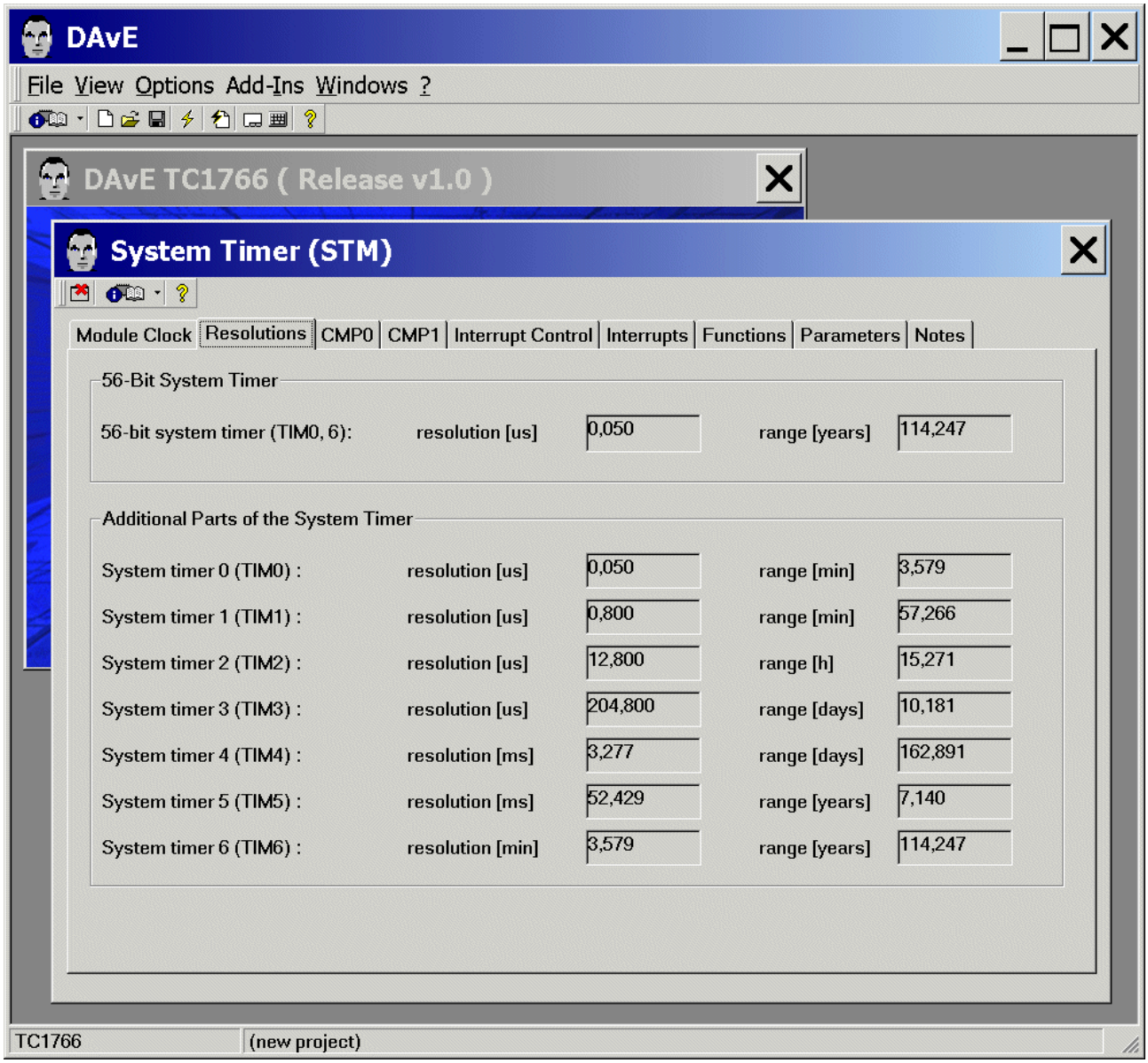
Note:
The LED on Port_1 Pin_0 will blink (after program start and if selected in the main menu) at a frequency of 1 second (done in the STM-Interrupt-Service-Routine).
Therefore we now have to configure the STM.

Module Clock: Sleep Mode Enable Control: tick ✓ Disable the sleep mode for the STM module

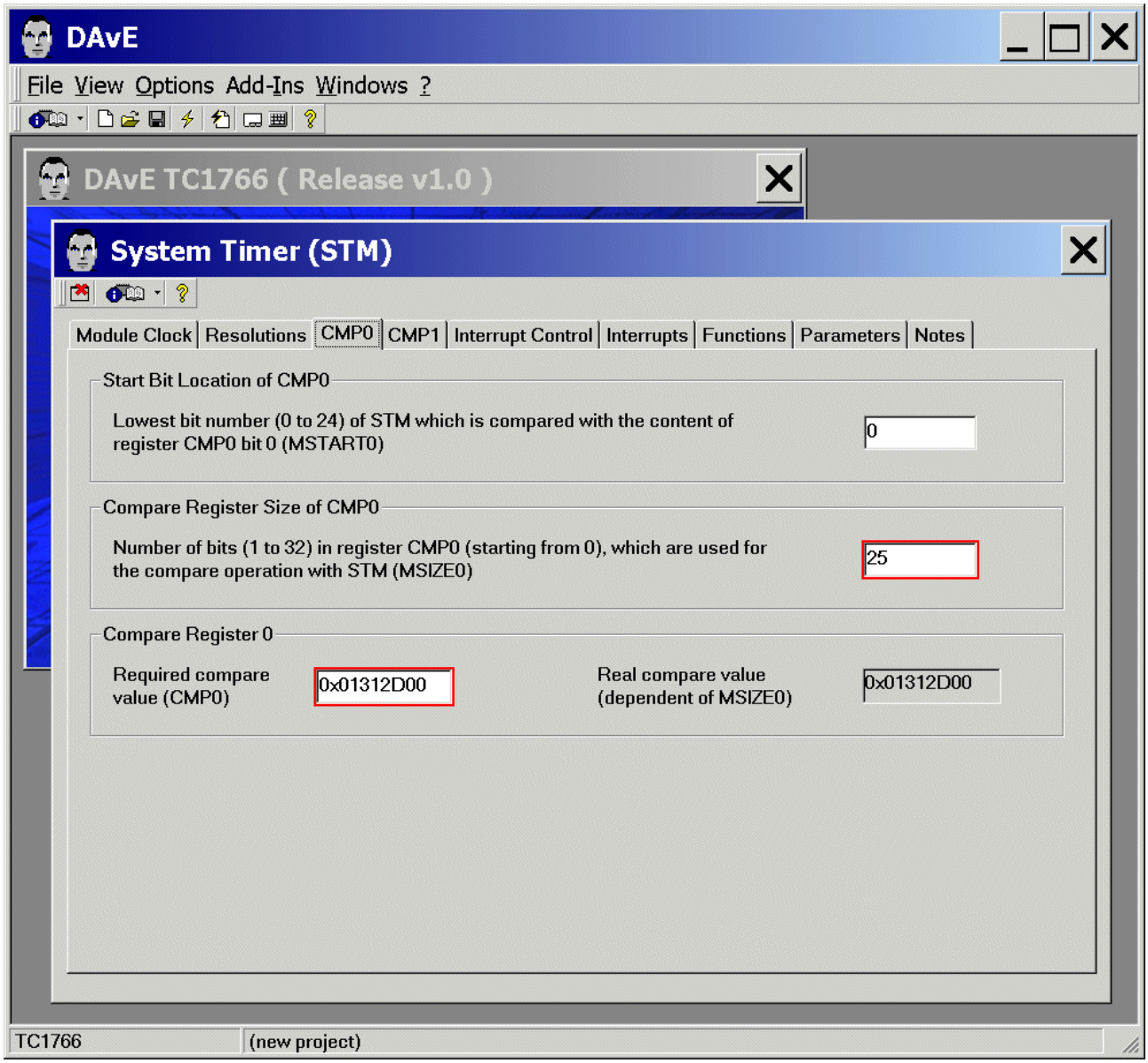
Module Clock: Module Run Mode Clock Control: Clock divider for normal operation mode: select System clock / 4 (= 50 ns)



Resolutions: (do nothing)

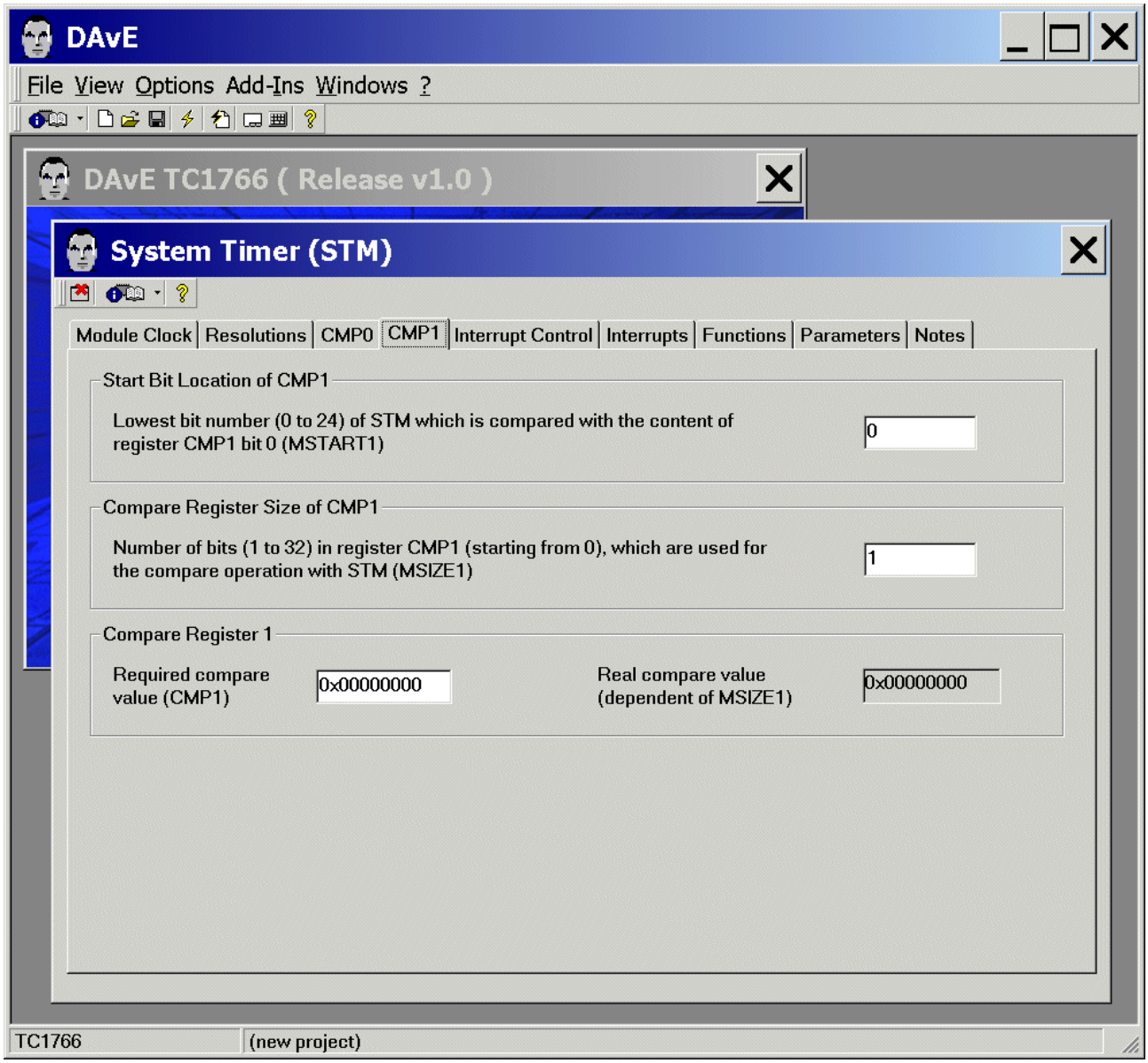


CMP0: Compare Register Size of CMP0: Number of bits for compare: insert 25 <ENTER>
CMP0: Compare Register 0: Required compare value (CMP0): insert 20000000 <ENTER>



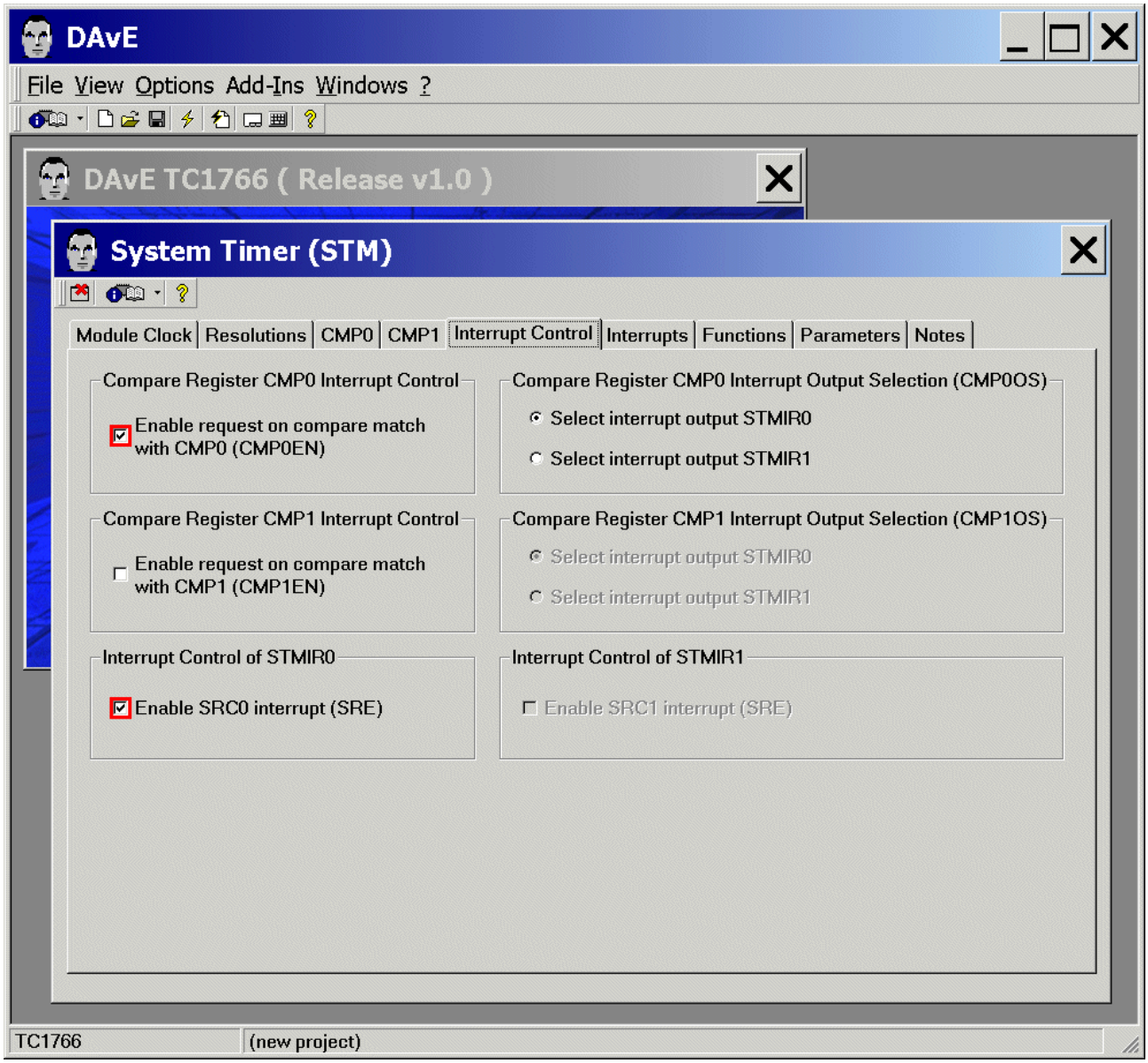
Note:
20.000.000 * 50 ns = 1 s

CMP1: (do nothing)

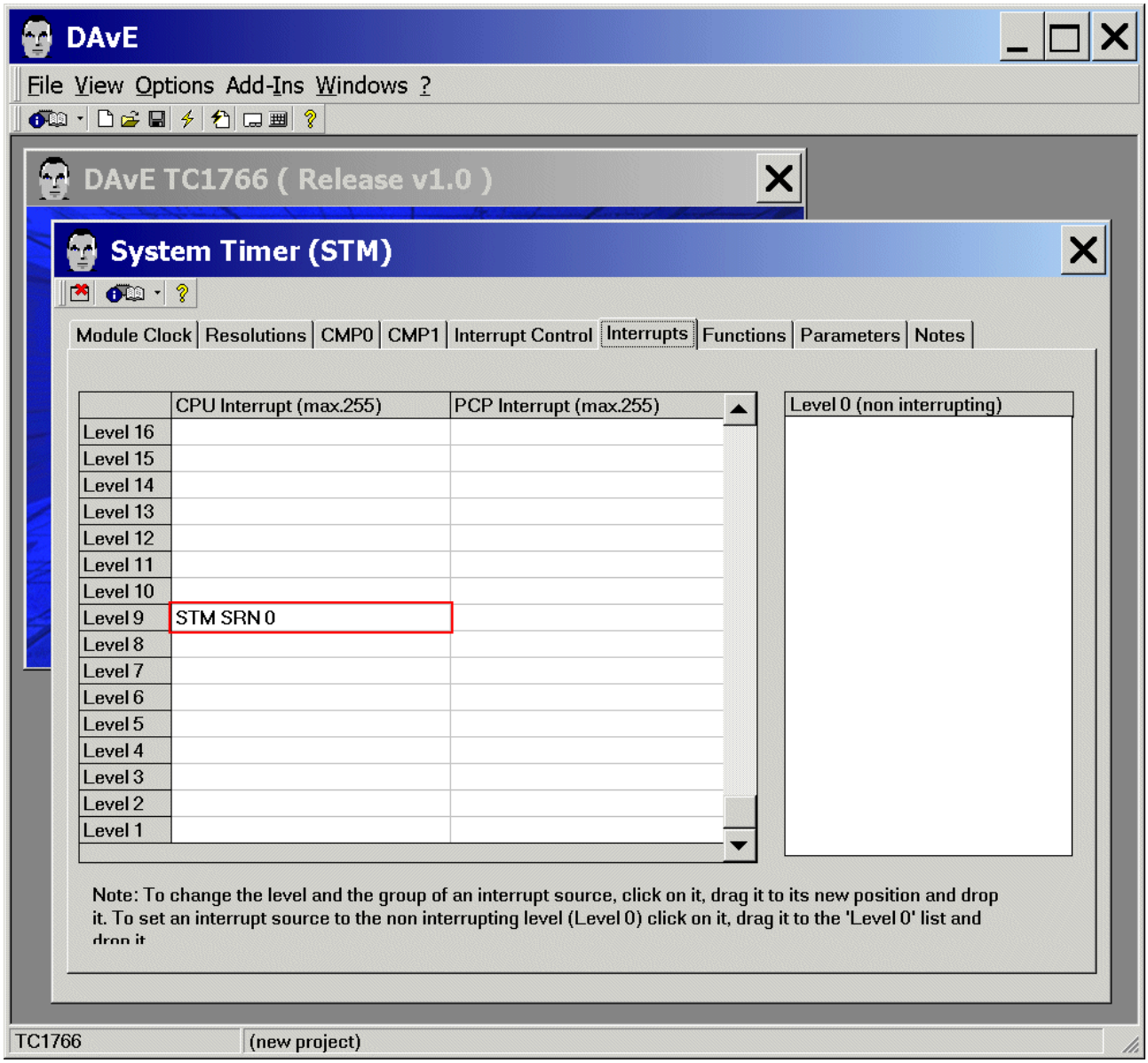


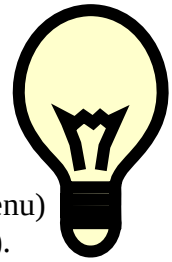
Interrupt Control: Compare Register CMP0 Interrupt Control:
tick ✓ Enable request on compare match with CMP0

Interrupt Control: Interrupt Control of STMIR0: tick ✓ Enable SRC0 interrupt



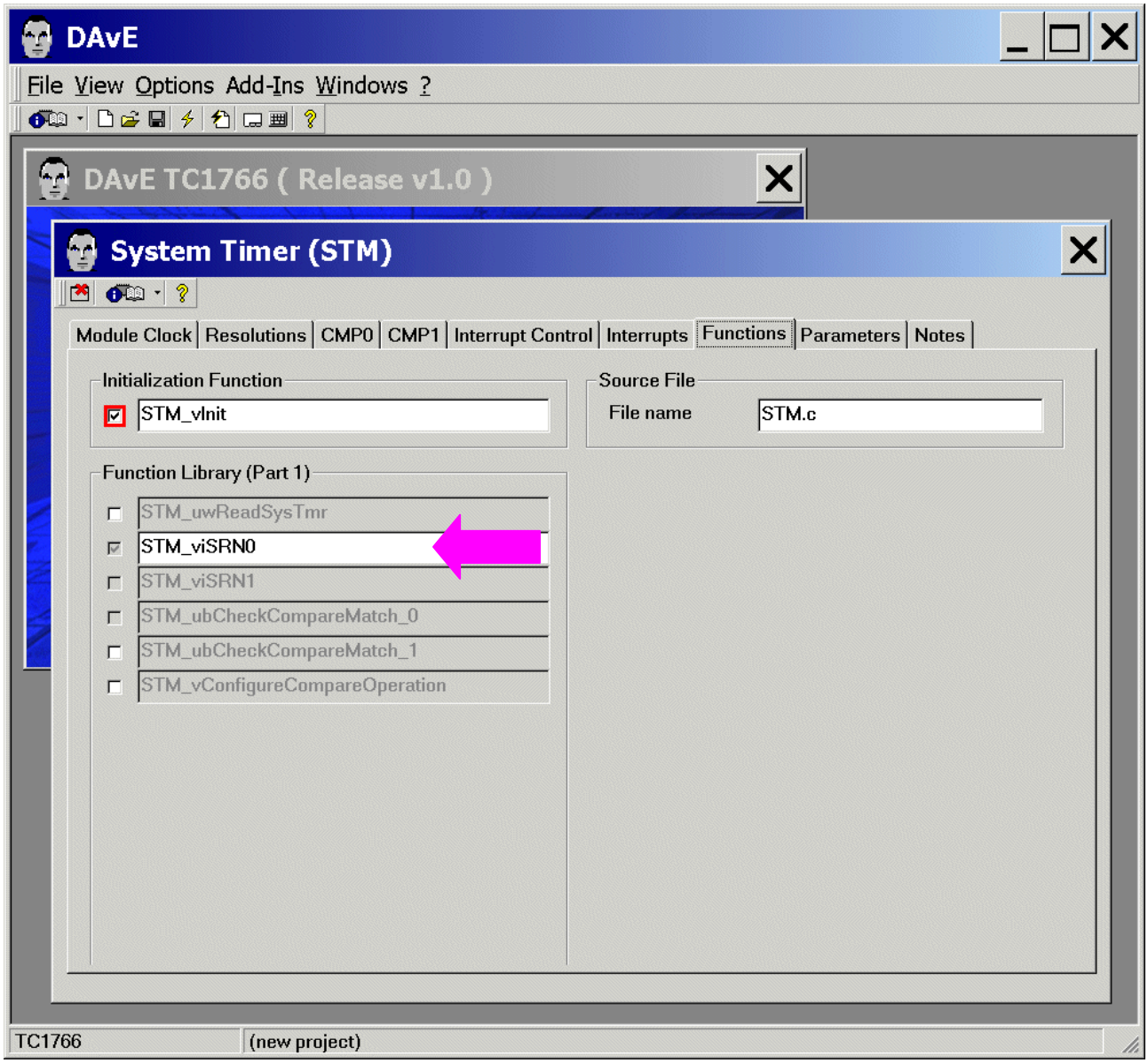
Interrupts: drag and drop STM SRN 0 from Level 0 to CPU Interrupt: Level 9

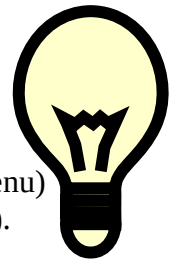




Note:
The LED on Port_1 Pin_0 will blink (after program start and if selected in the main menu) at a frequency of 1 second (done in the STM-Interrupt-Service-Routine STM_viSRN0).

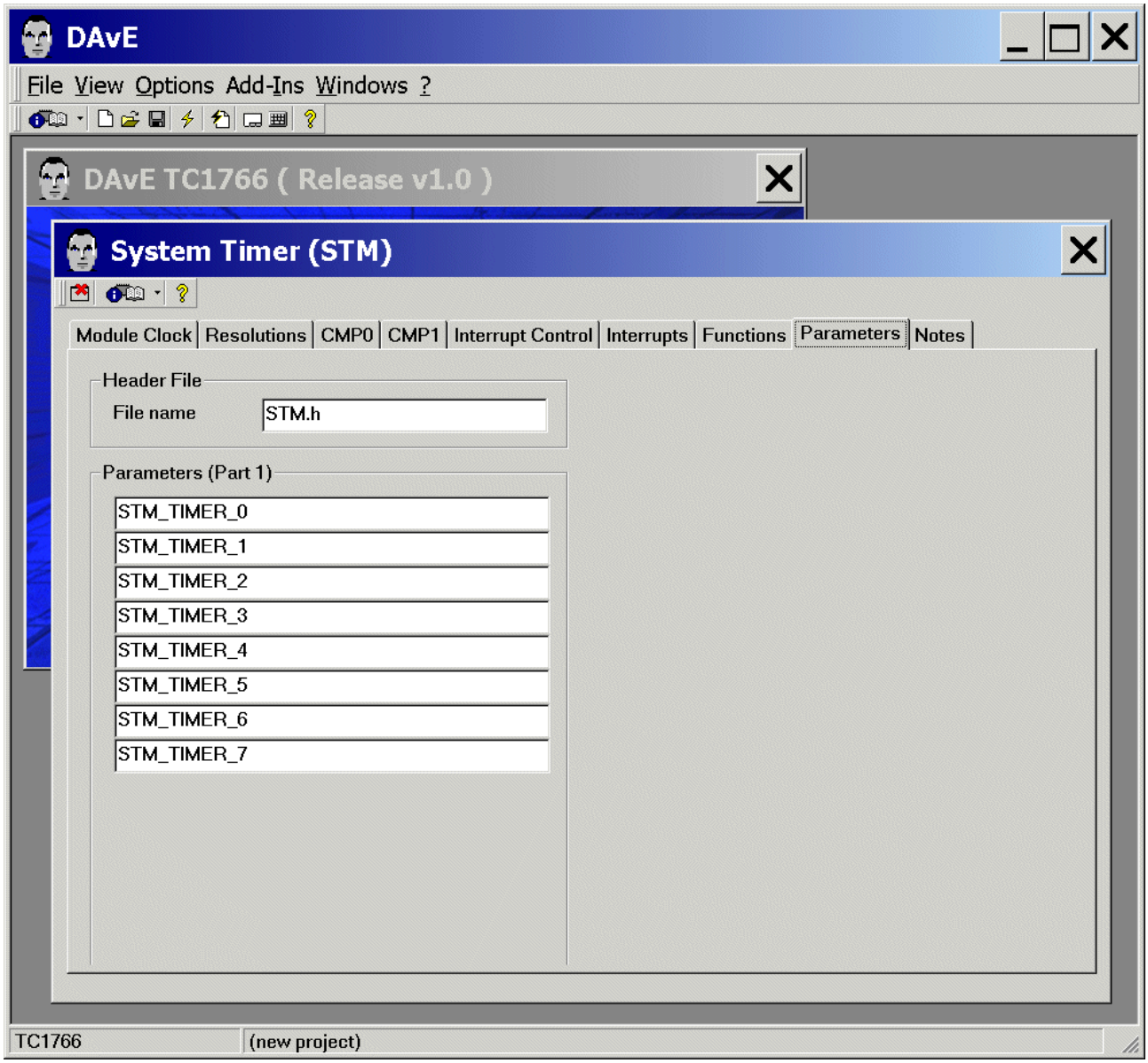
Functions: Initialization Function: tick ✓ STM_vInit





Note:
The LED on Port_1 Pin_0 will blink (after program start and if selected in the main menu) at a frequency of 1 second (done in the STM-Interrupt-Service-Routine STM_viSRN0).

Parameters: (do nothing)

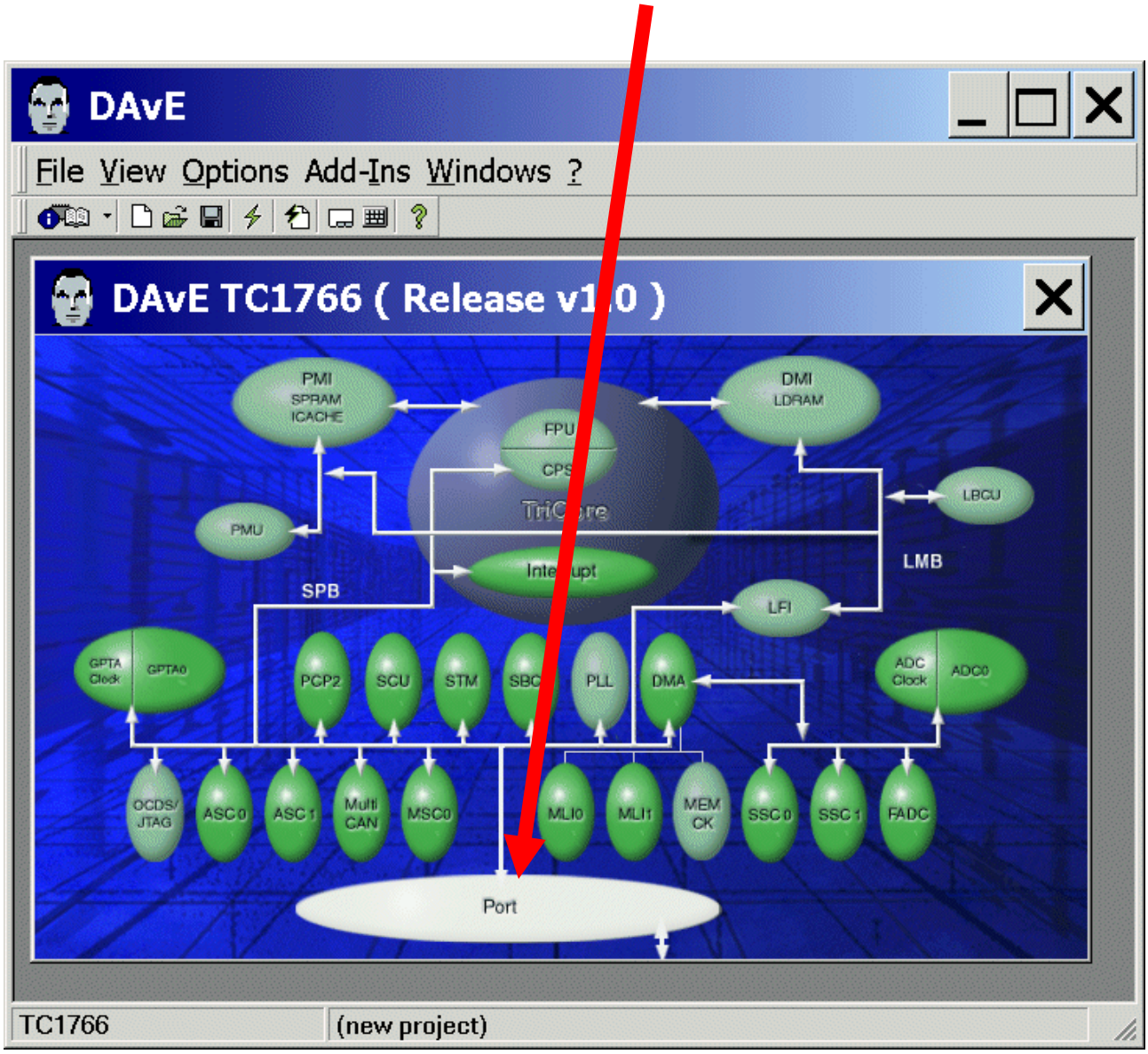


Notes: If you wish, you can insert your comments here.

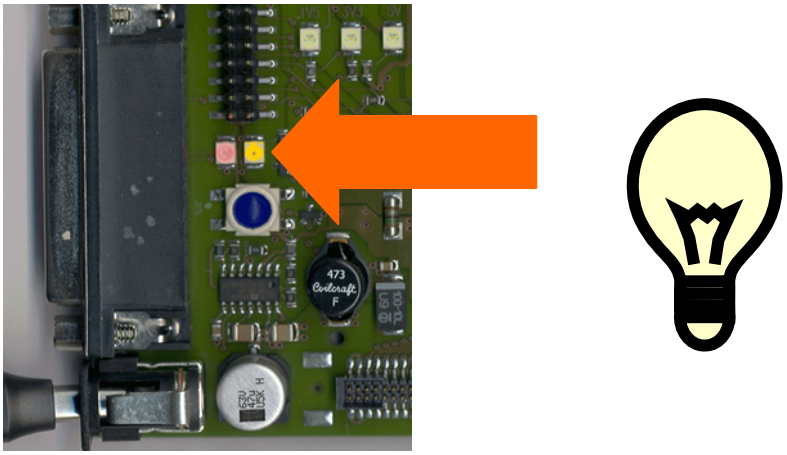
Exit and Save this dialog now by clicking the close button.

Port Configuration:

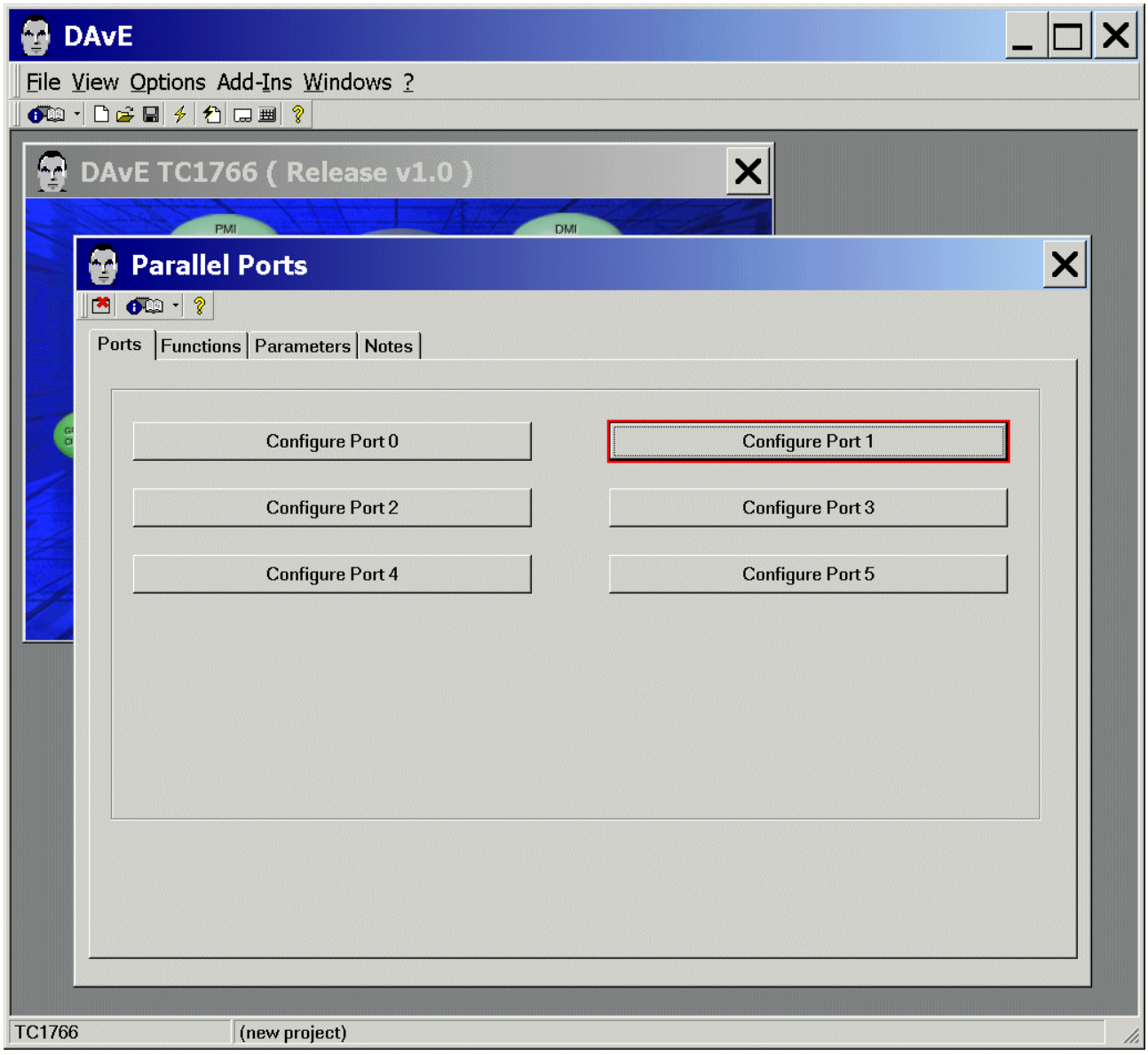
The configuration window/dialog can be opened by clicking the specific block/module.



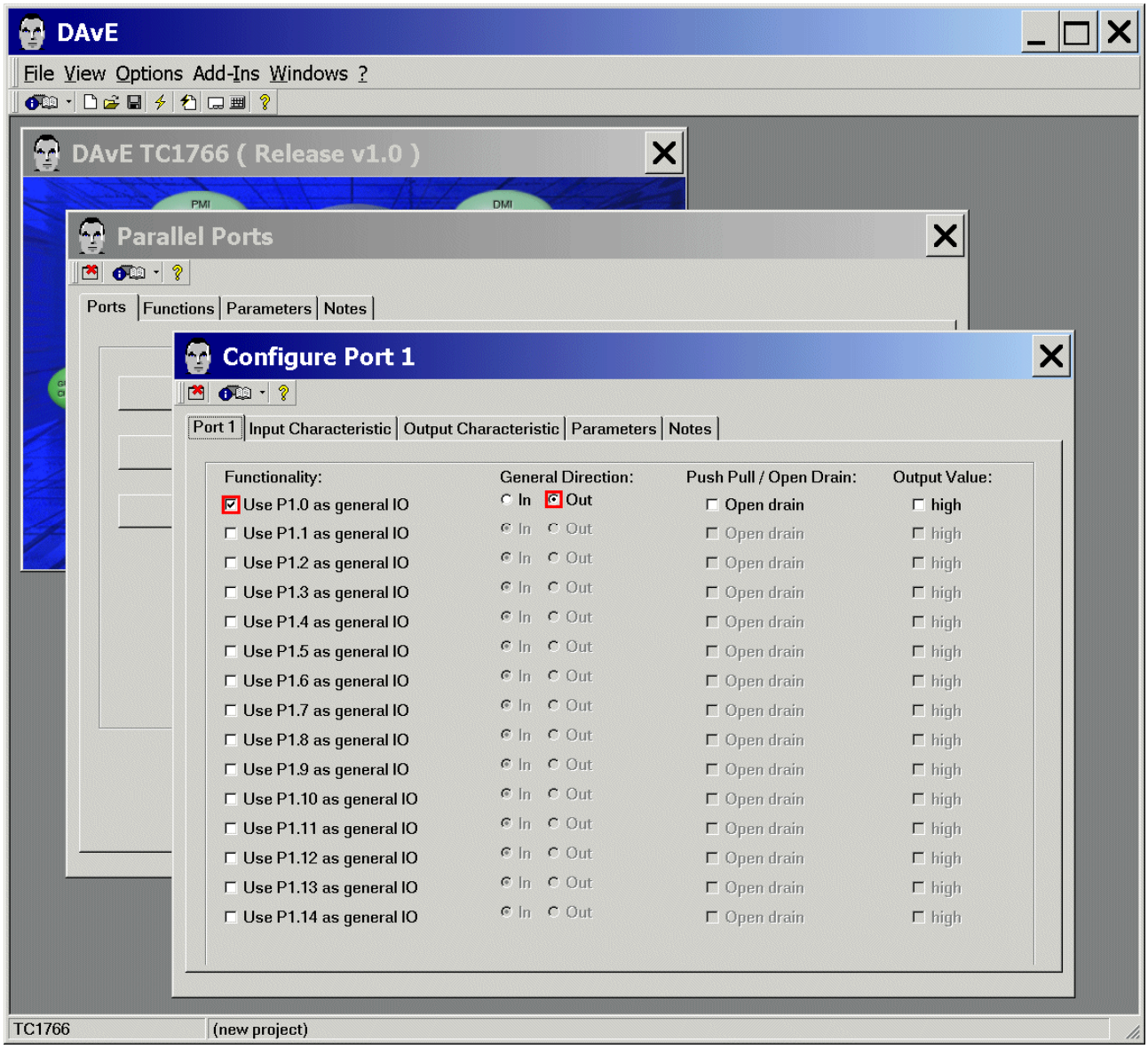
Note:
The User LED (orange) is connected to Port_1 Pin_0.



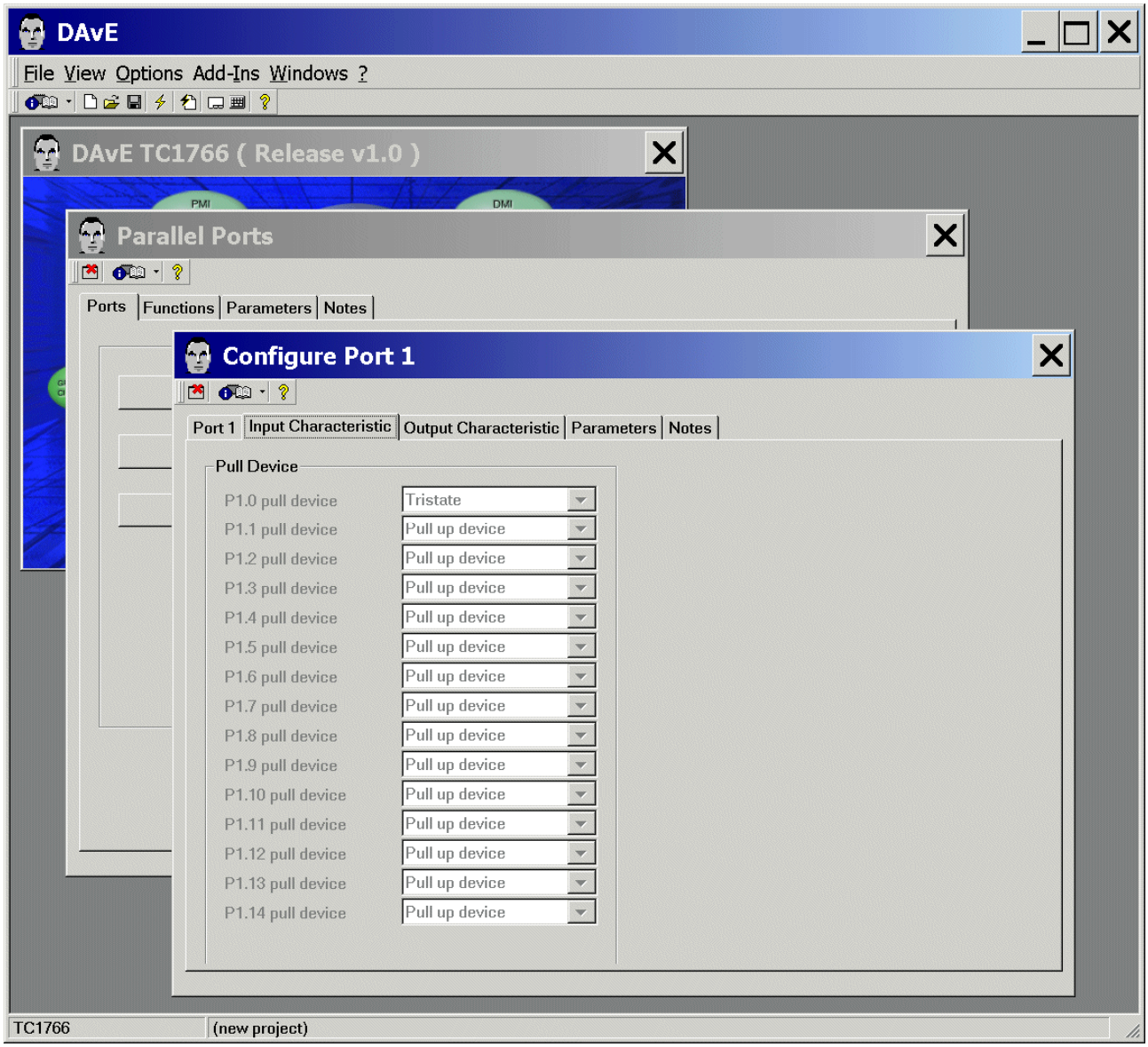
Ports: click Configure Port 1



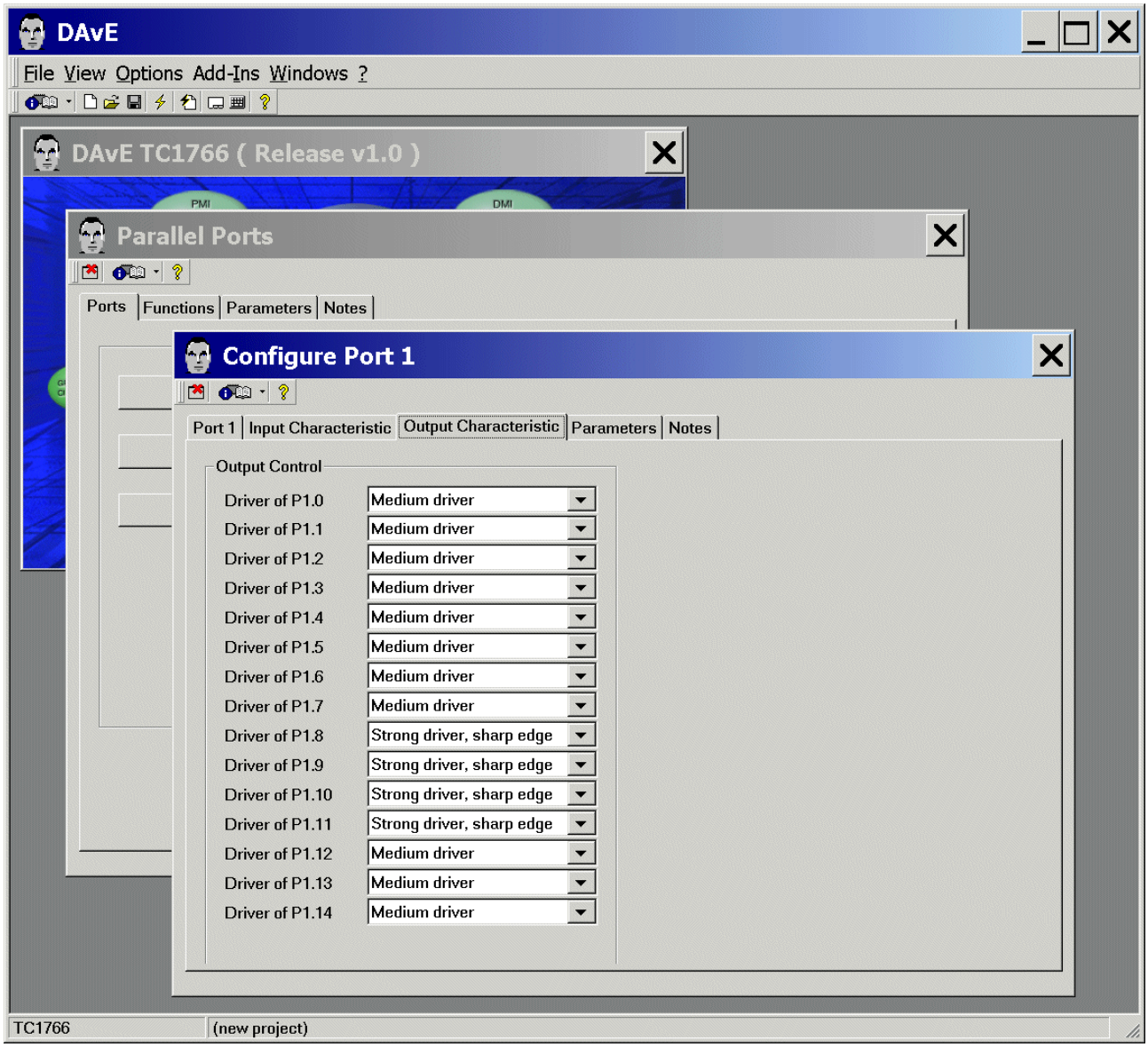
Ports: Configure Port 1:
Port 1: Functionality: tick ✓ Use P1.0 as general IO, General Direction: click ⊙ Out



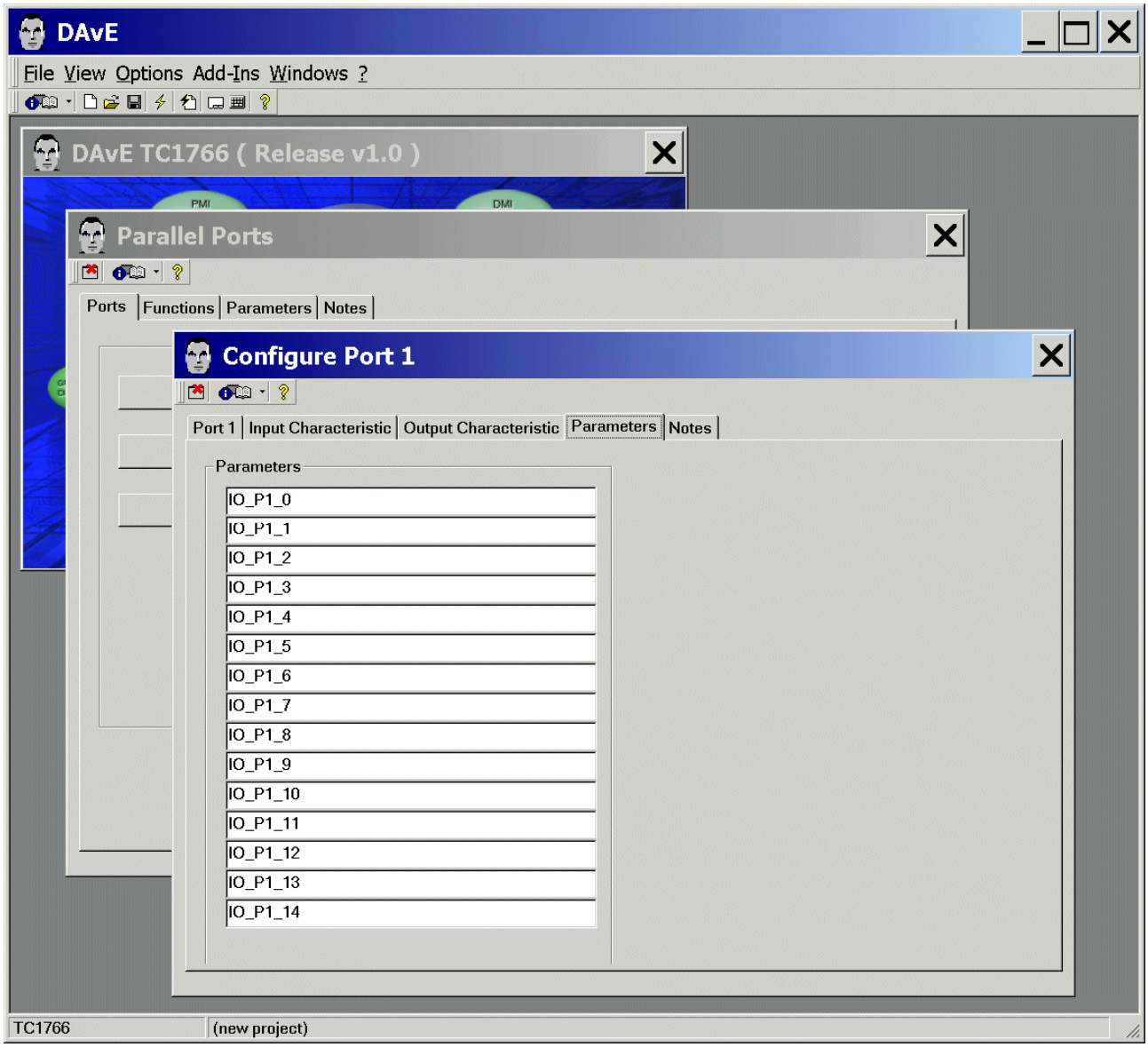
Input Characteristic: (do nothing)



Output Characteristic: (do nothing)



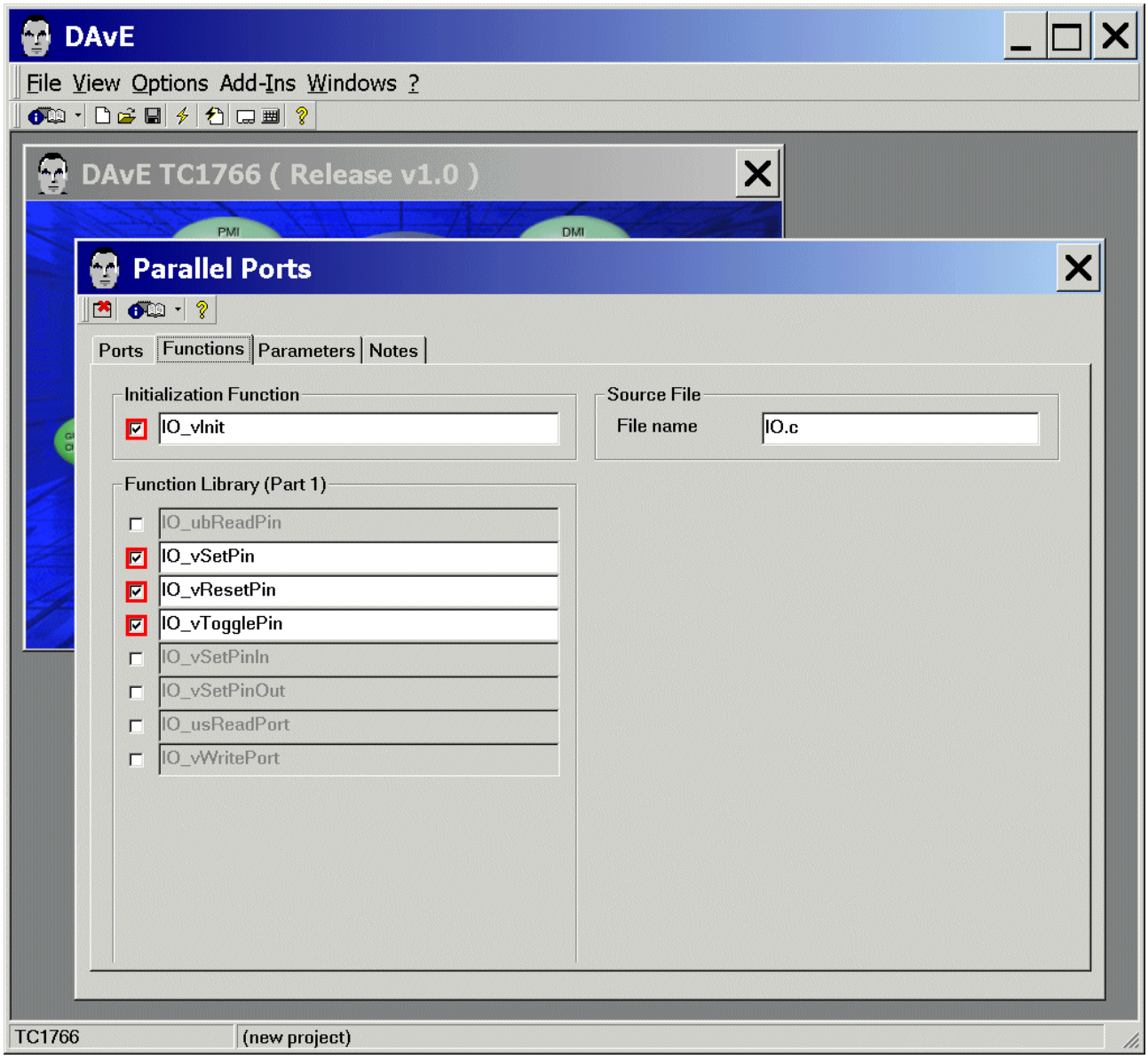
Parameters: (do nothing)



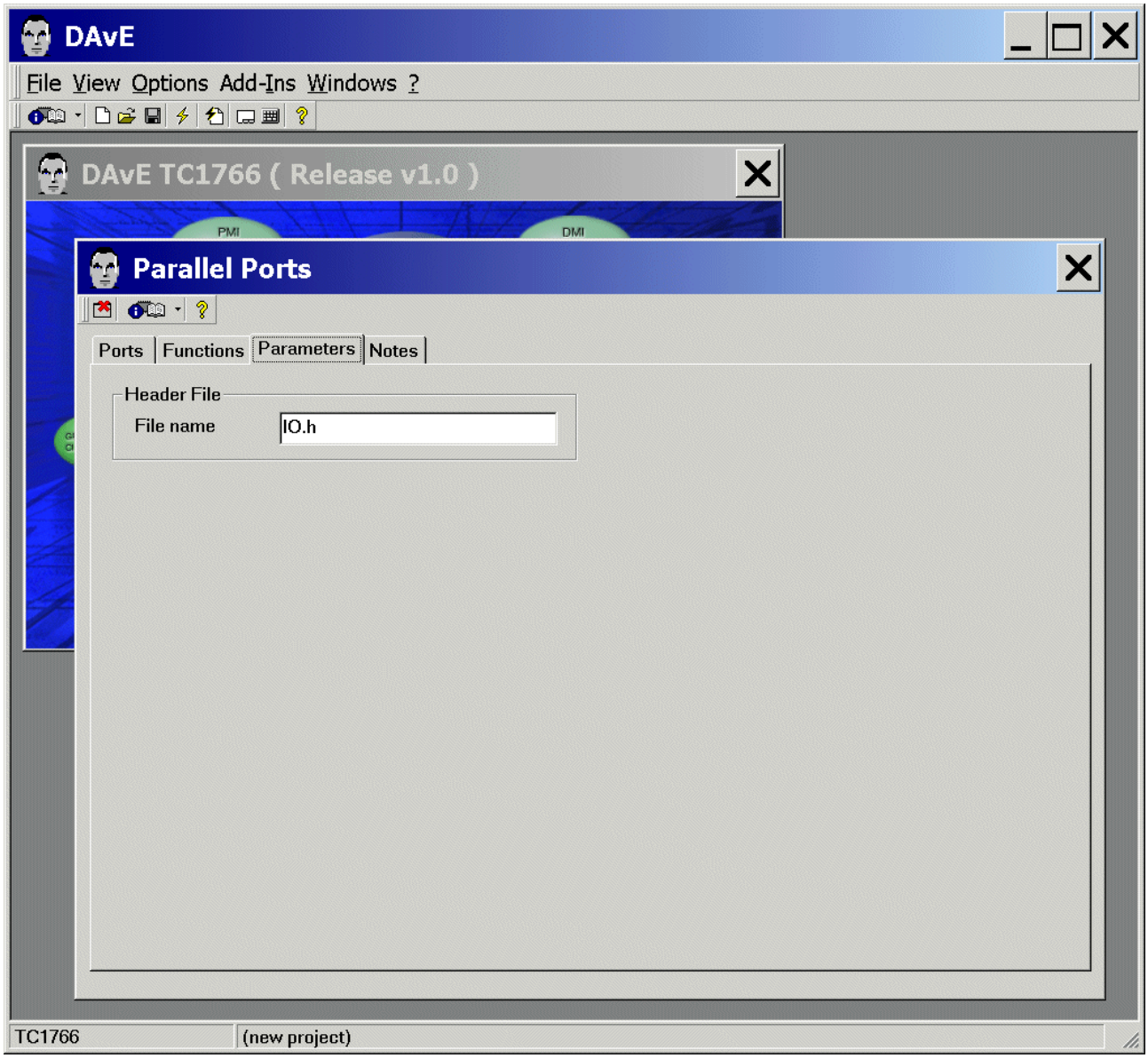
Notes: If you wish, you can insert your comments here.

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

Functions: Initialization Function: tick ✓ IO_vInit
Functions: Function Library (Part 1): tick ✓ IO_vSetPin
Functions: Function Library (Part 1): tick ✓ IO_vResetPin
Functions: Function Library (Part 1): tick ✓ IO_vTogglePin



Parameters : (do nothing)

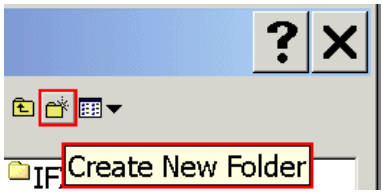


Notes: If you wish, you can insert your comments here.

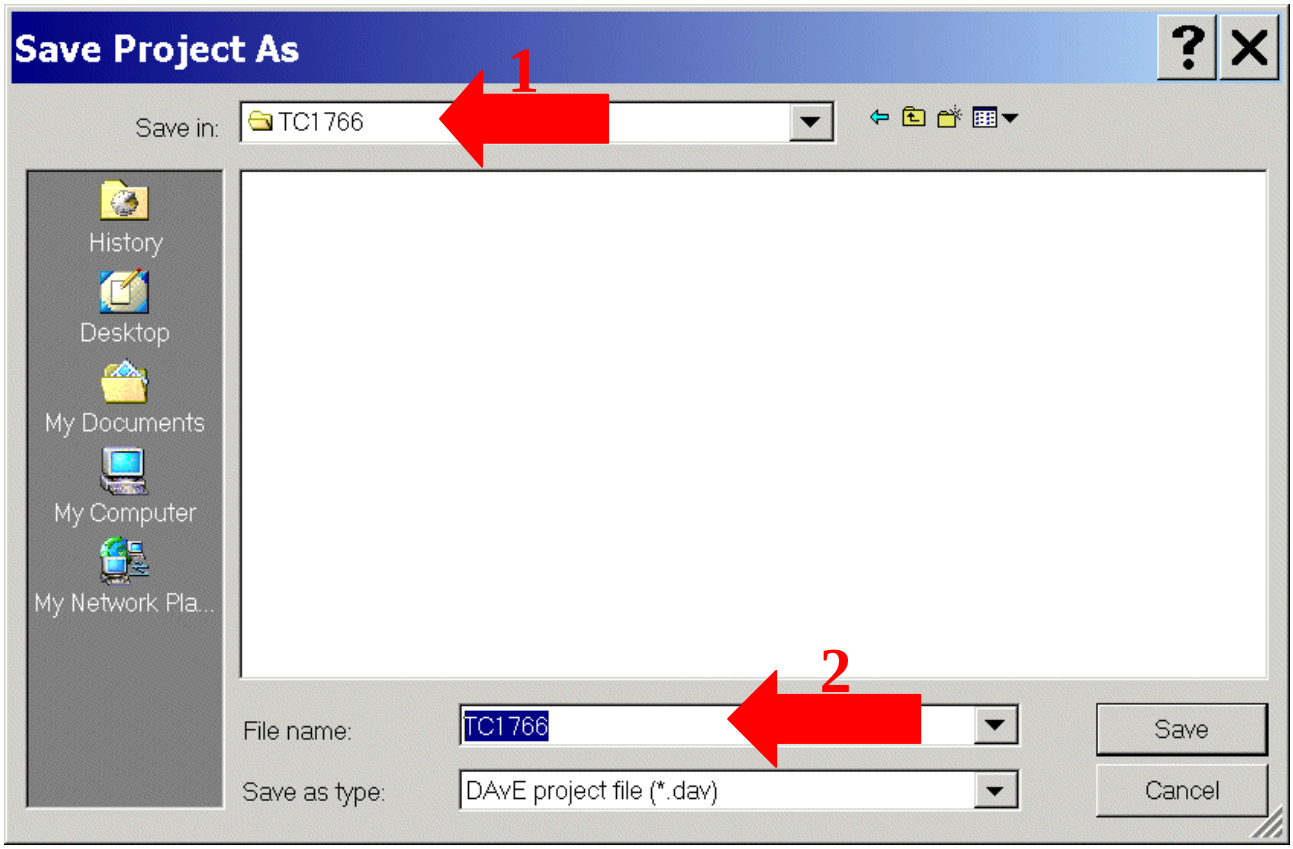
Exit and Save this dialog now by clicking  the close button.

Save the project:

File
Save

Save project: Save in C:\TC1766 [create new directory 

(1)]
File name: TC1766 (2)

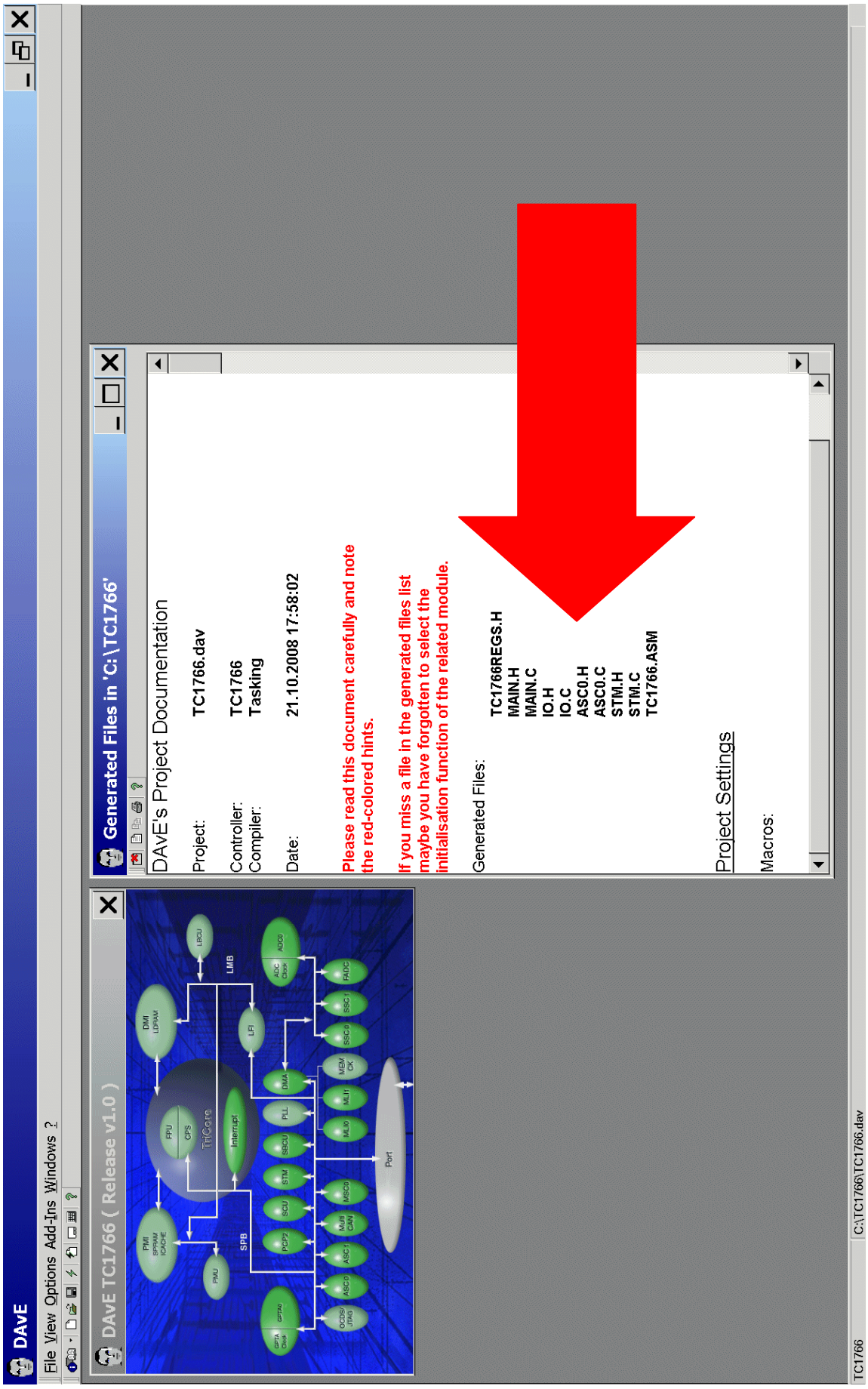


Save

Generate Code:

File Generate Code	or click 

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


DAvE
File View Options Add-Ins Windows ?
DAvE TC1766 (Release v1.0)
Generated Files in 'C:\TC1766'
DAvE's Project Documentation
Project: TC1766.dav
Controller: TC1766
Compiler: Tasking
Date: 21.10.2008 17:58:02
Please read this document carefully and note the red-colored hints.
If you miss a file in the generated files list maybe you have forgotten to select the initialisation function of the related module.
Generated Files:
TC1766REGS.H
MAIN.H
MAIN.C
IO.H
IO.C
ASC0.H
ASC0.C
STM.H
STM.C
TC1766.ASM
Project Settings
Macros:
TC1766
C:\TC1766\TC1766.dav

Close DAvE:

File

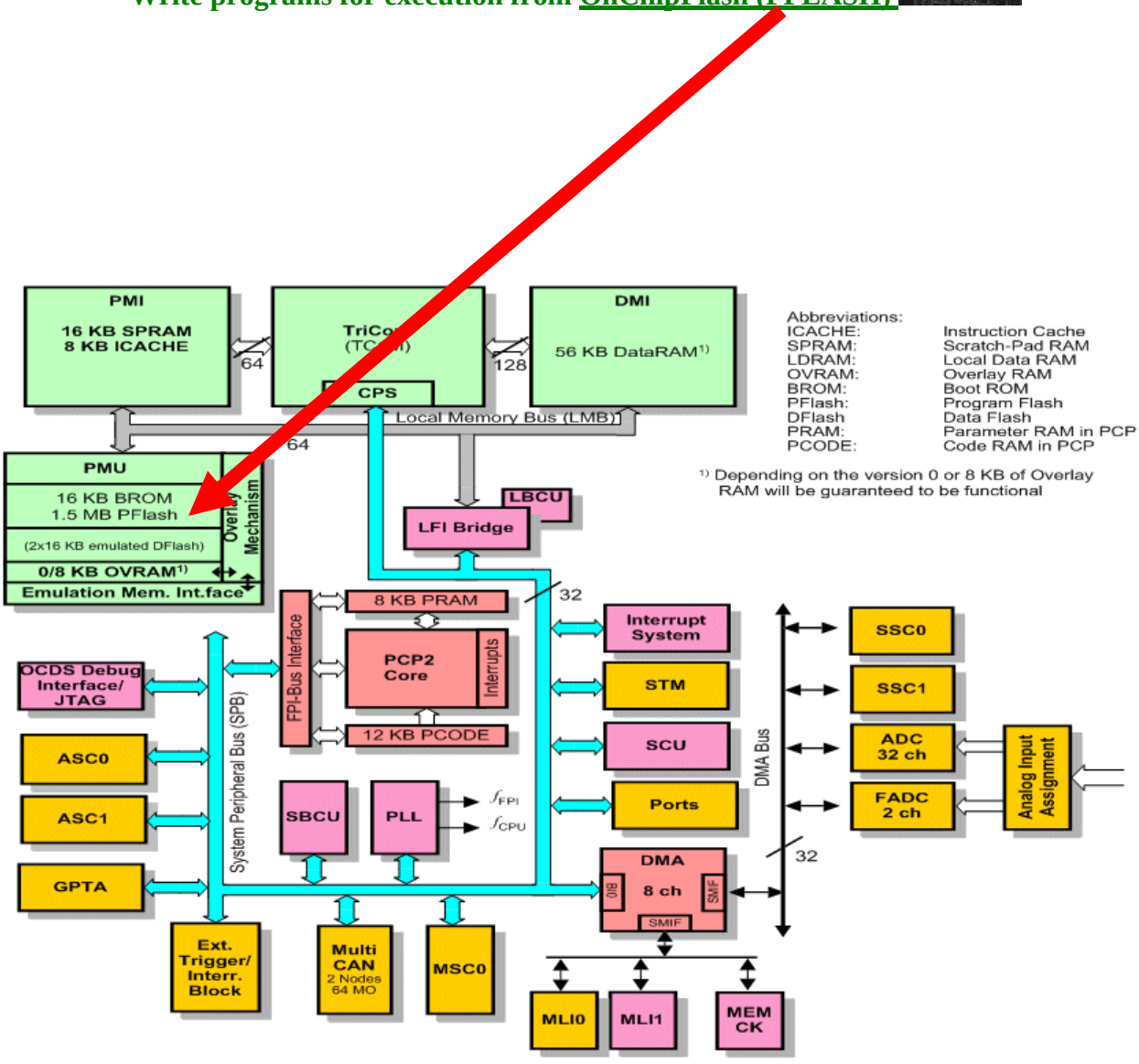
Exit

Save changes?

click Yes

4.) Using the TASKING - EDE Development Tools:

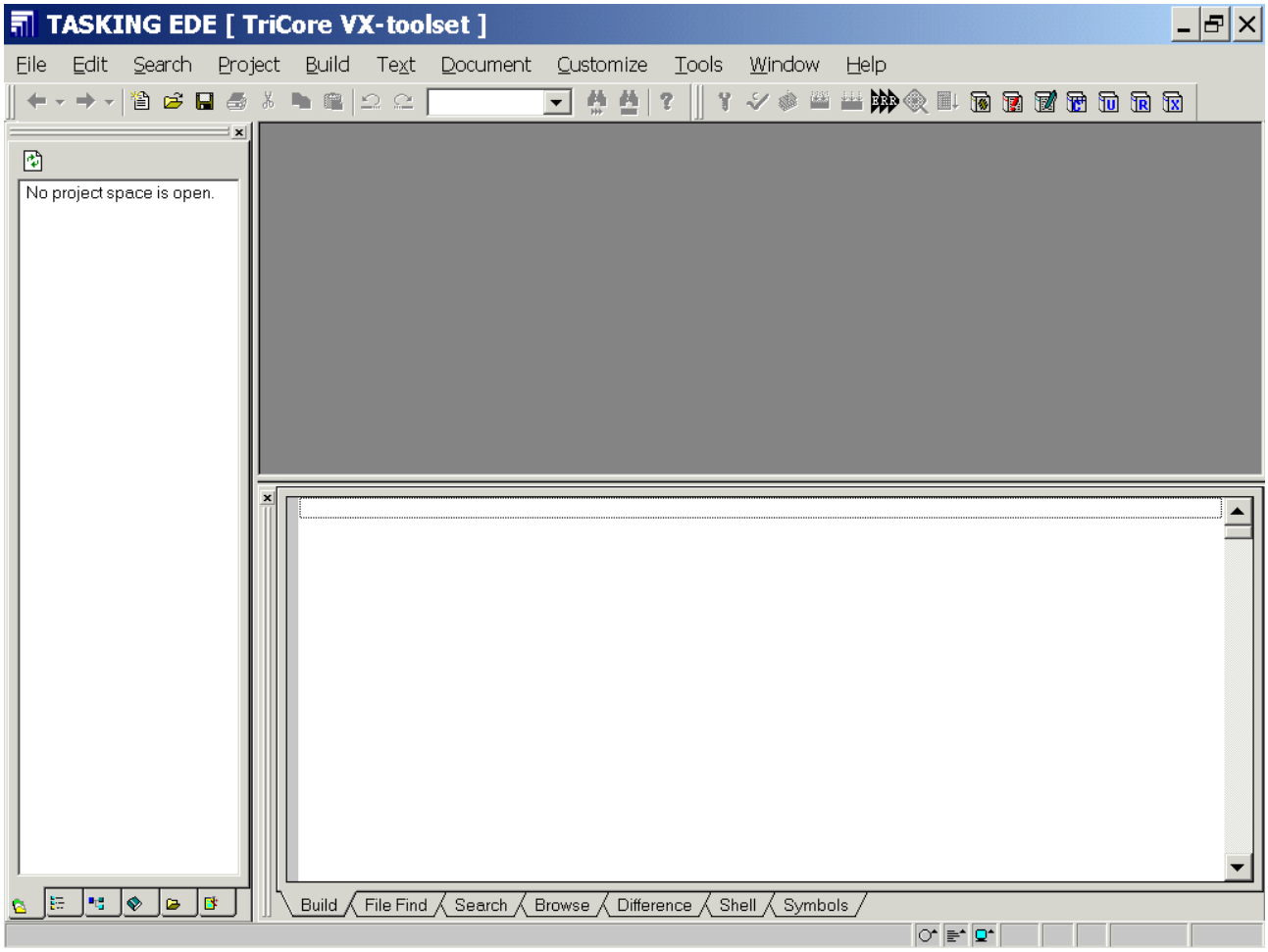
Write programs for execution from OnChipFlash (PFLASH) 



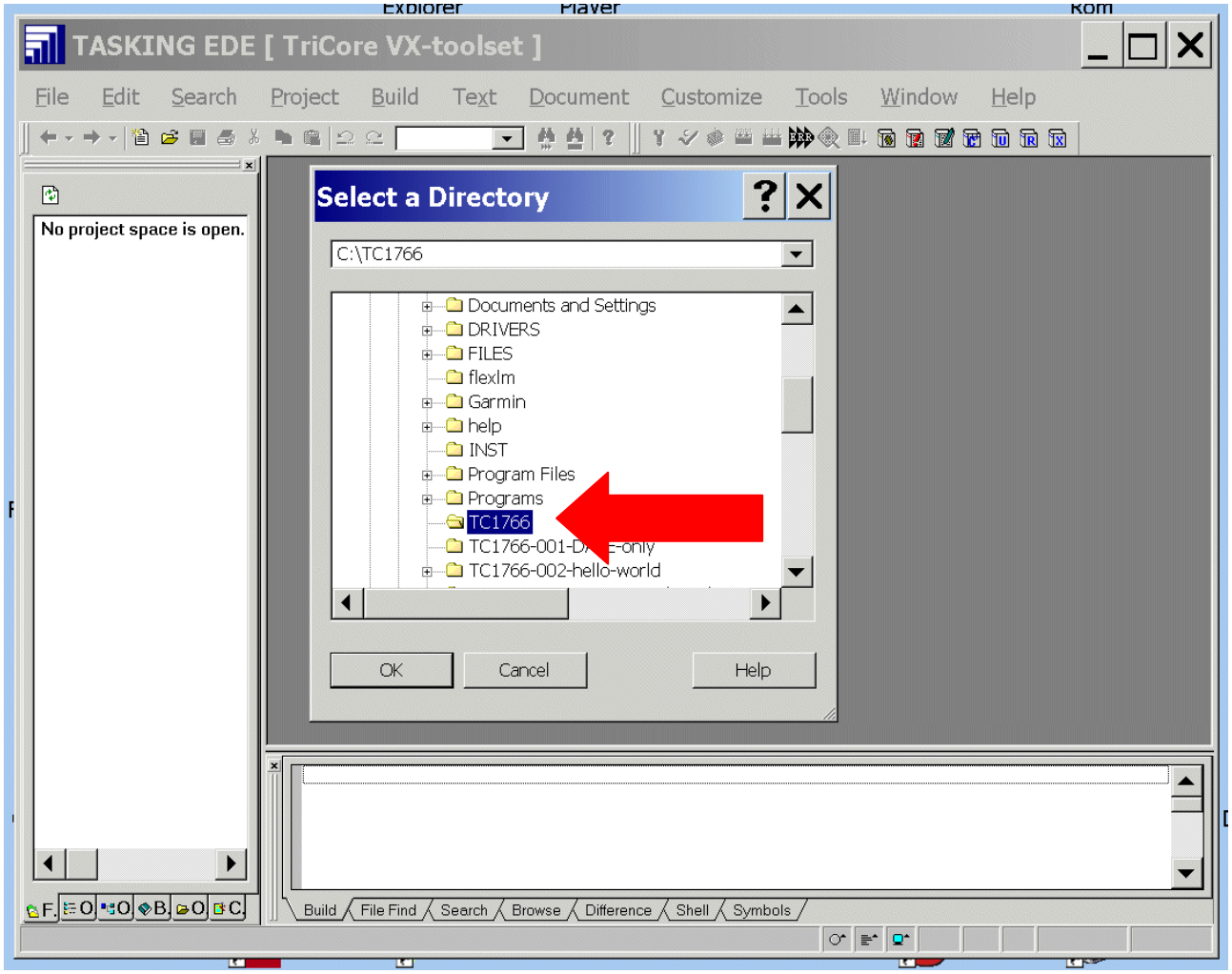
Install the Tasking Development Tools TriCore v2.3r1

Start Tasking EDE, select directory and include the DAvE Files:

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

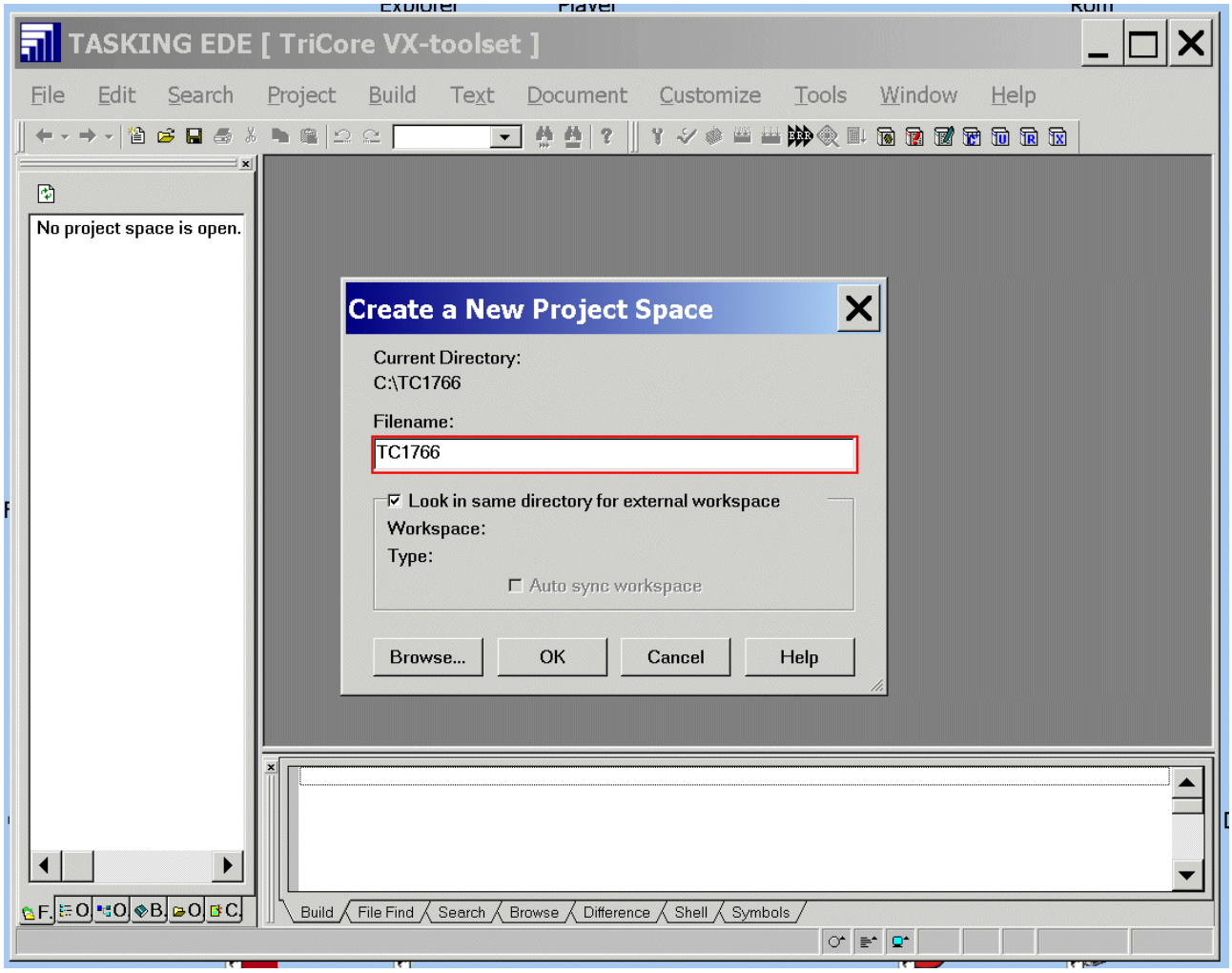


File - Change Directory…
Select a Directory: choose C:\TC1766



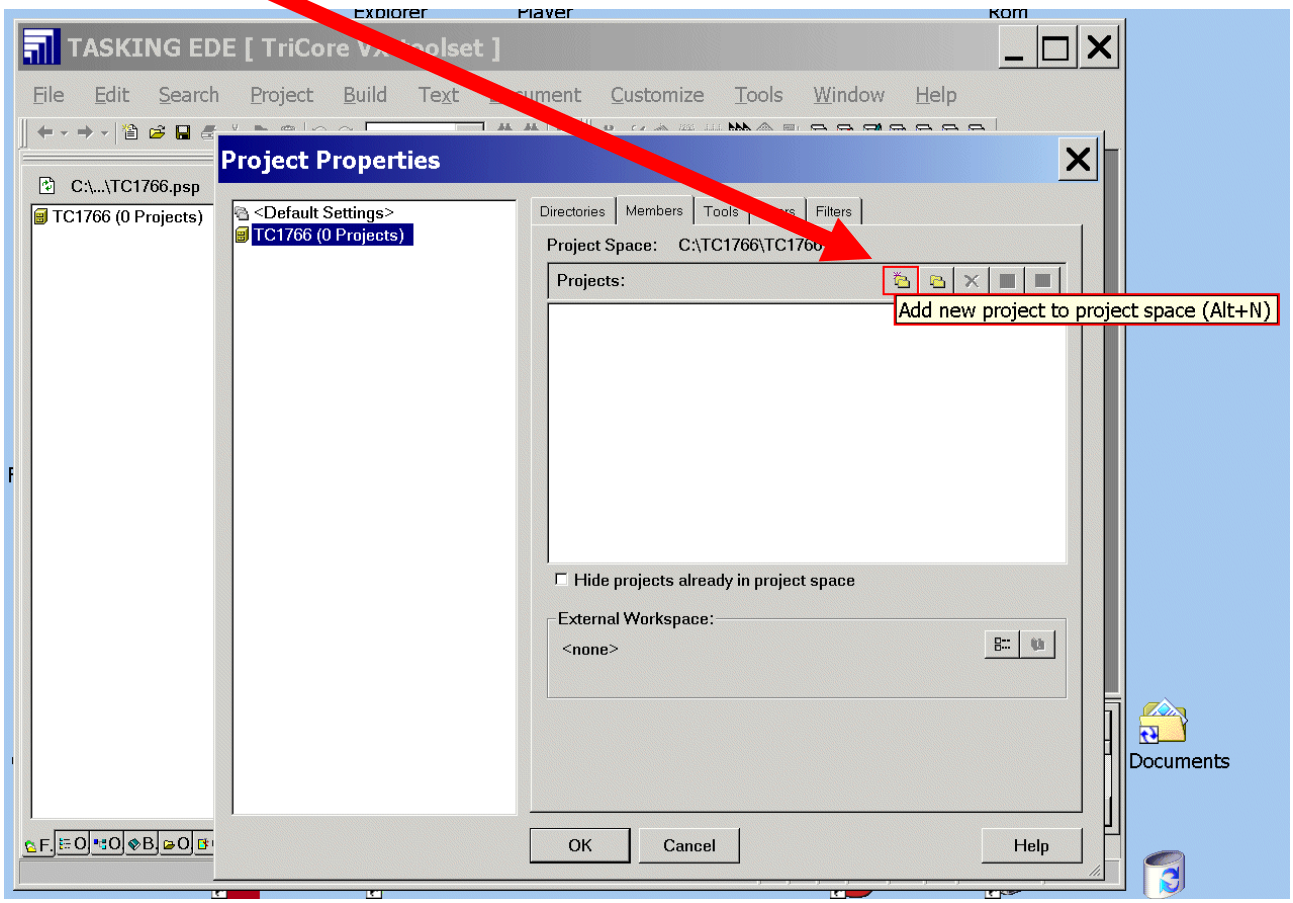
OK

File - New Project Space…
Create a New Project Space: Filename: insert TC1766

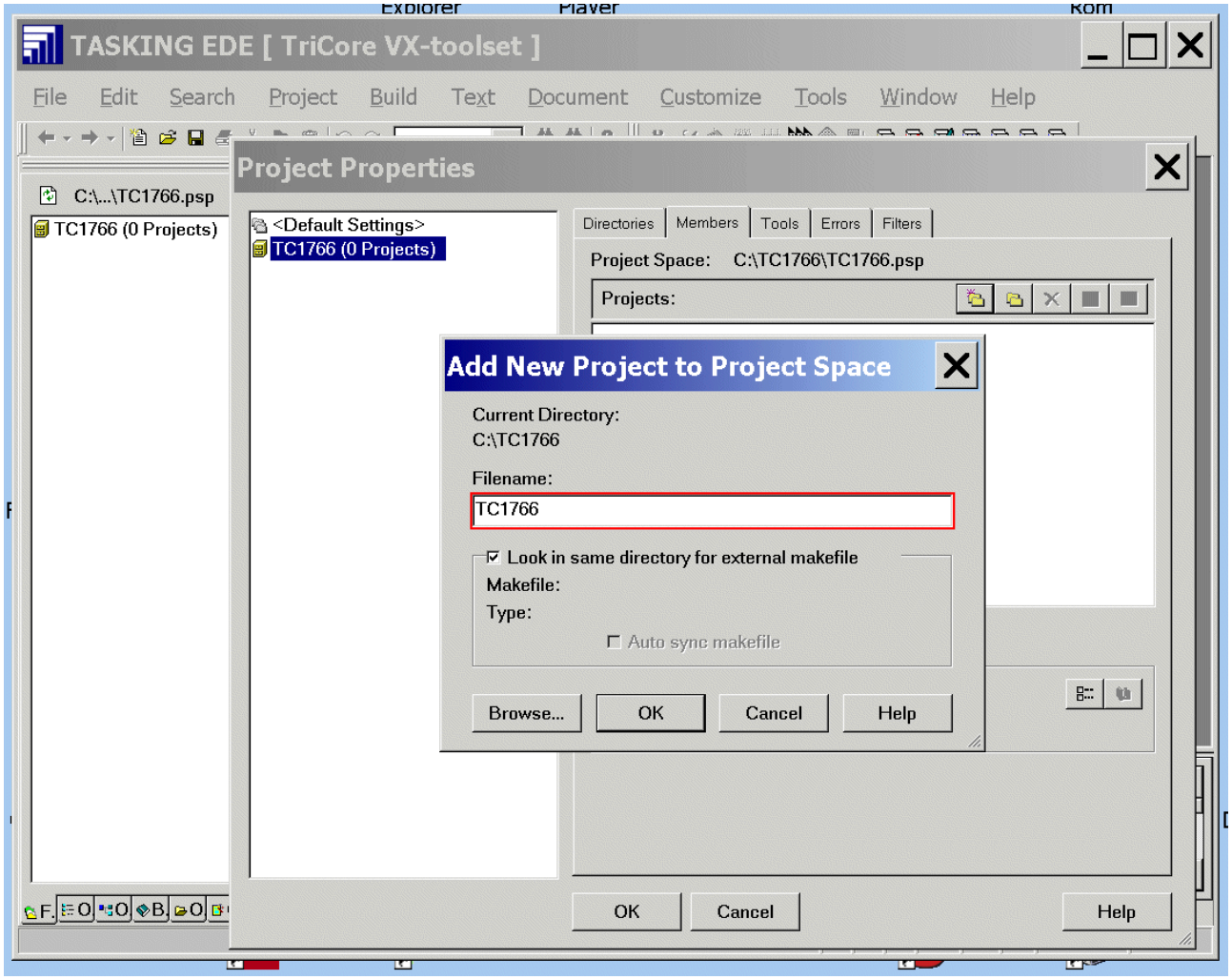


OK

Click:  “Add new project to project space”

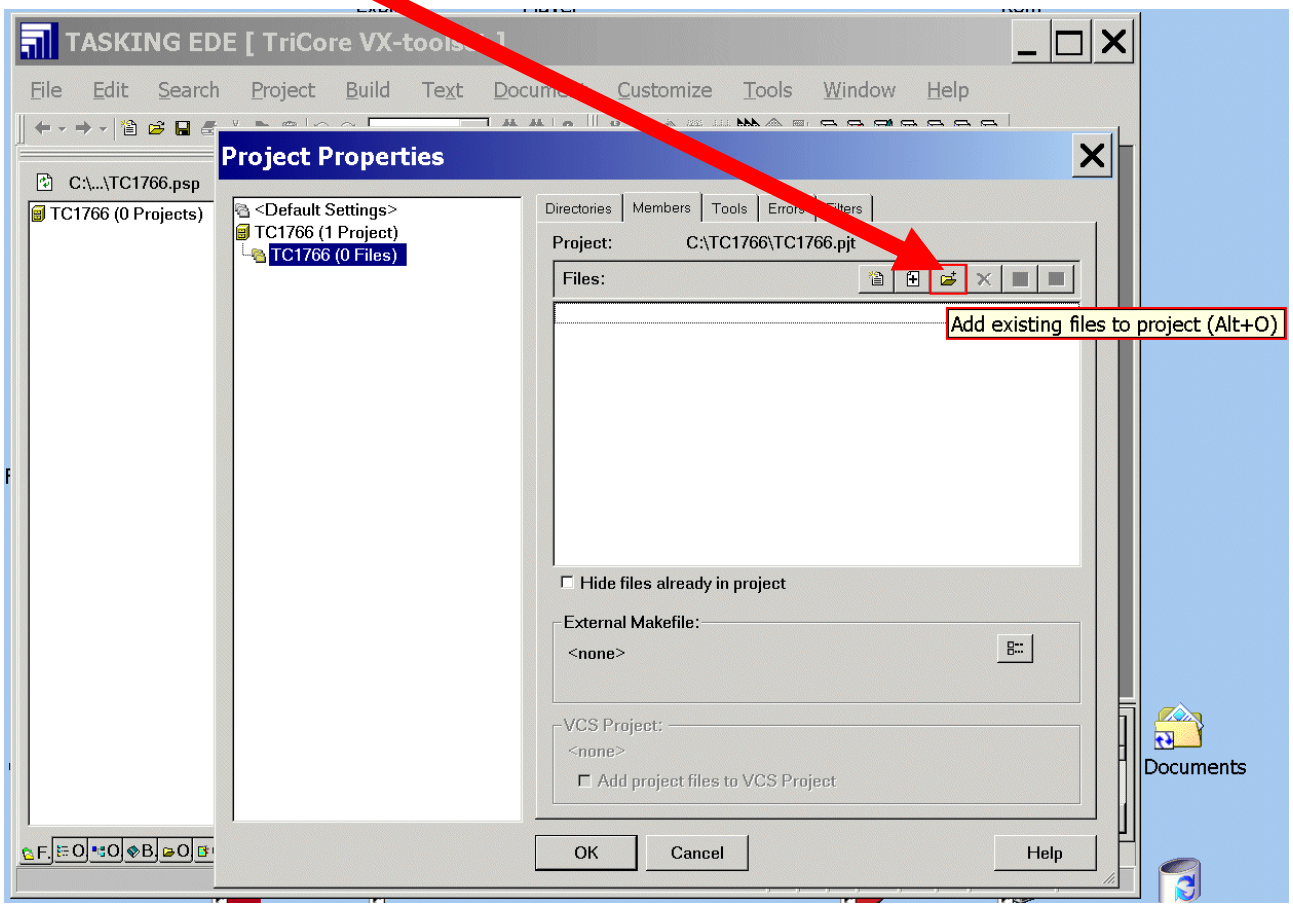


Add New Project to Project Space: Filename: insert TC1766

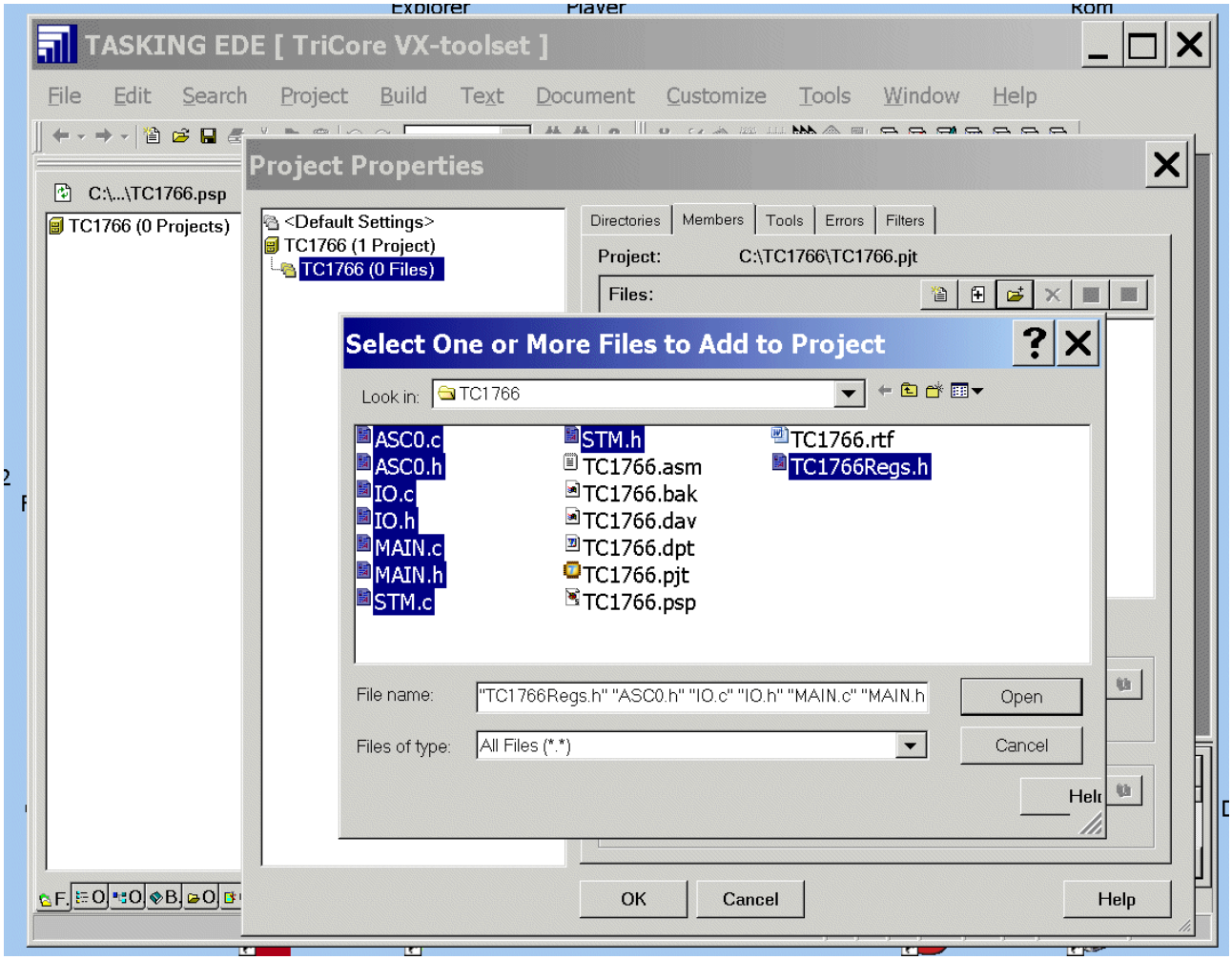


OK

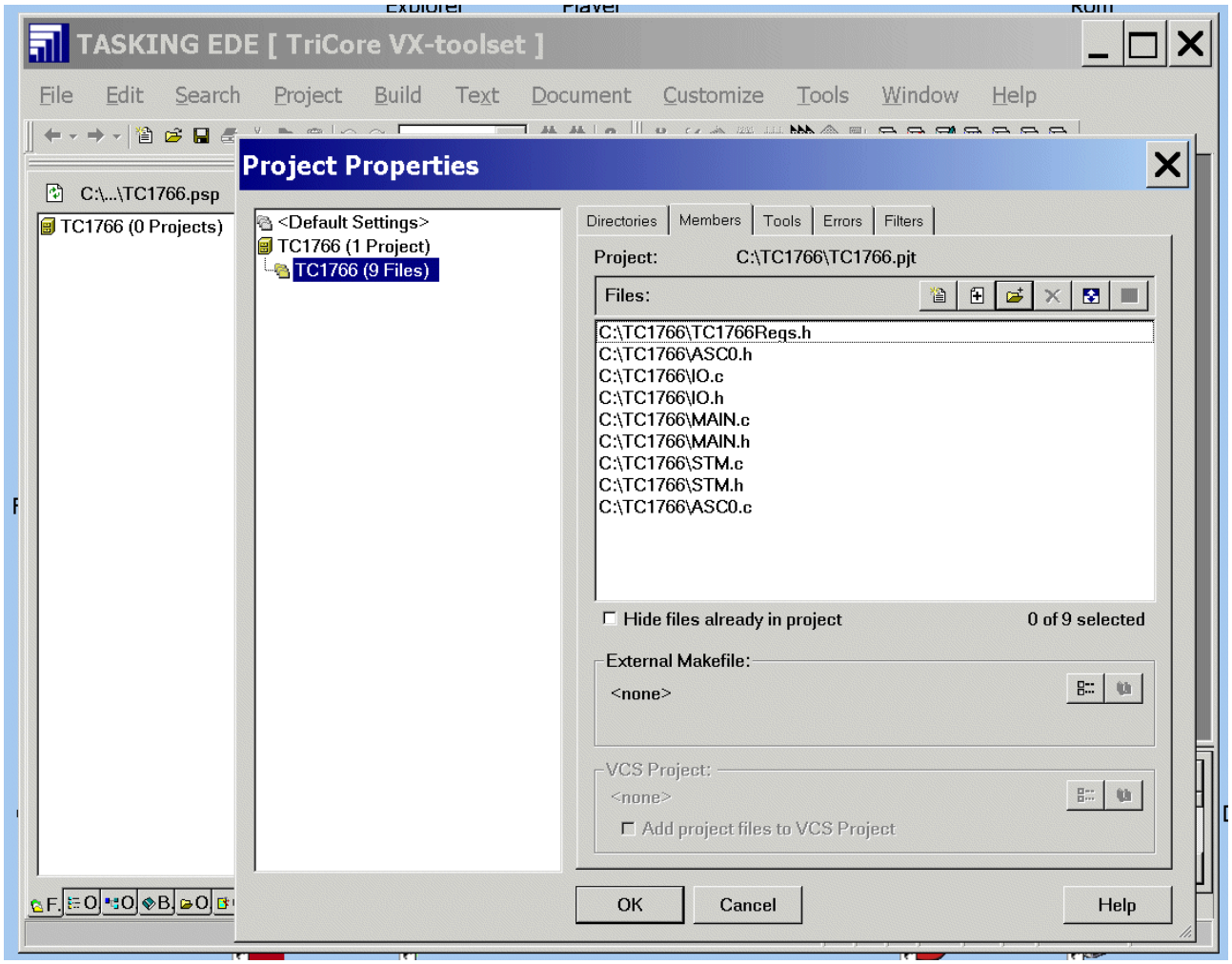
Click:  “Add existing files to project”



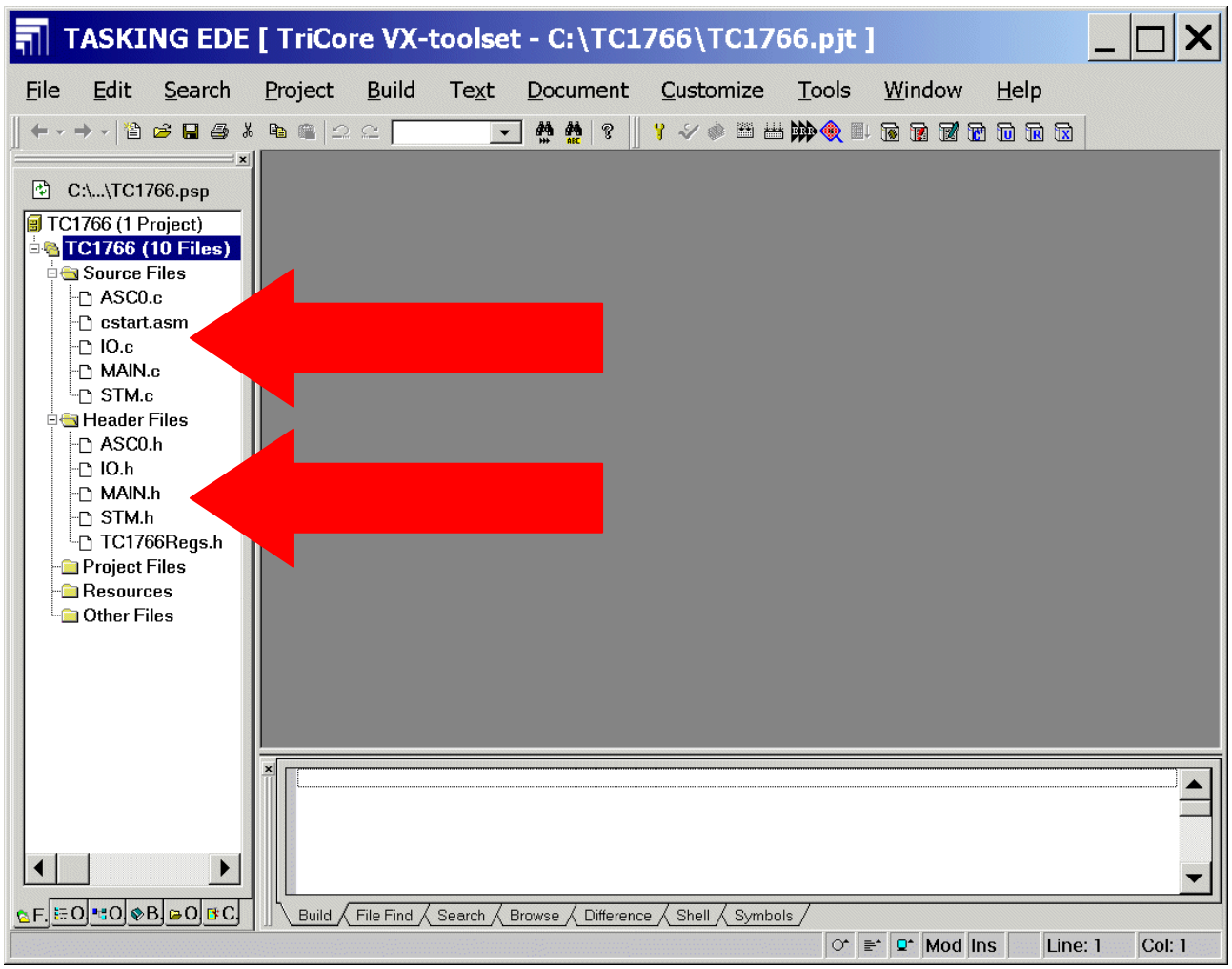
Select One or More Files to Add to Project: select ASC0.c
Select One or More Files to Add to Project: select ASC0.h
Select One or More Files to Add to Project: select IO.c
Select One or More Files to Add to Project: select IO.h
Select One or More Files to Add to Project: select MAIN.c
Select One or More Files to Add to Project: select MAIN.h
Select One or More Files to Add to Project: select STM.c
Select One or More Files to Add to Project: select STM.h
Select One or More Files to Add to Project: select TC1766Regs.h



Open


TASKING EDE [TriCore VX-toolset]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1766.psp
TC1766 (0 Projects)
Project Properties
<Default Settings>
TC1766 (1 Project)
TC1766 (9 Files)
Directories Members Tools Errors Filters
Project: C:\TC1766\TC1766.pjt
Files:
C:\TC1766\TC1766Regs.h
C:\TC1766\ASC0.h
C:\TC1766\IO.c
C:\TC1766\IO.h
C:\TC1766\MAIN.c
C:\TC1766\MAIN.h
C:\TC1766\STM.c
C:\TC1766\STM.h
C:\TC1766\ASC0.c
Hide files already in project
0 of 9 selected
External Makefile:
<none>
VCS Project:
<none>
Add project files to VCS Project
OK
Cancel
Help

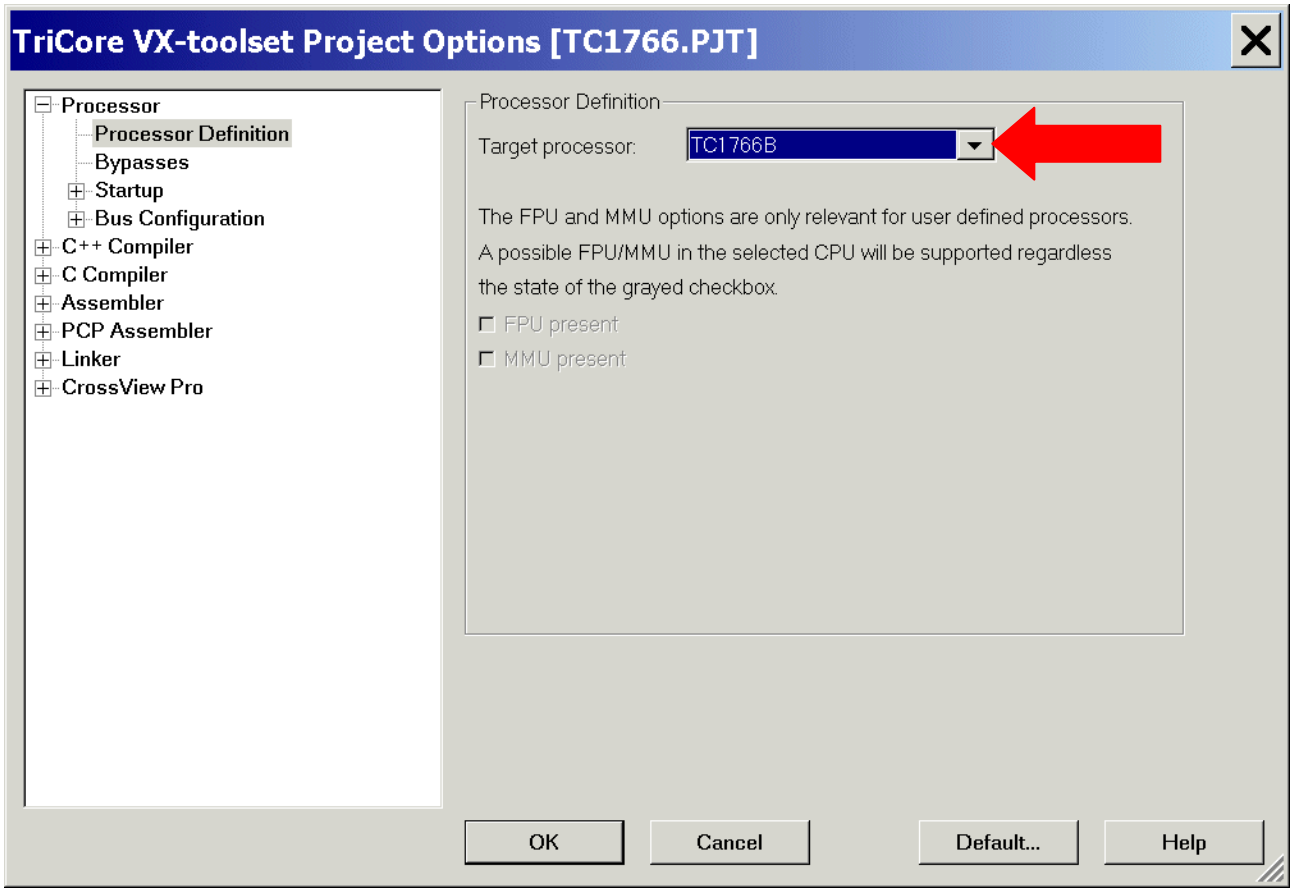
OK


TASKING EDE [TriCore VX-toolset - C:\TC1766\TC1766.pjt]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1766.psp
TC1766 (1 Project)
TC1766 (10 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
TC1766Regs.h
Project Files
Resources
Other Files
Build File Find Search Browse Difference Shell Symbols
Mod Ins Line: 1 Col: 1

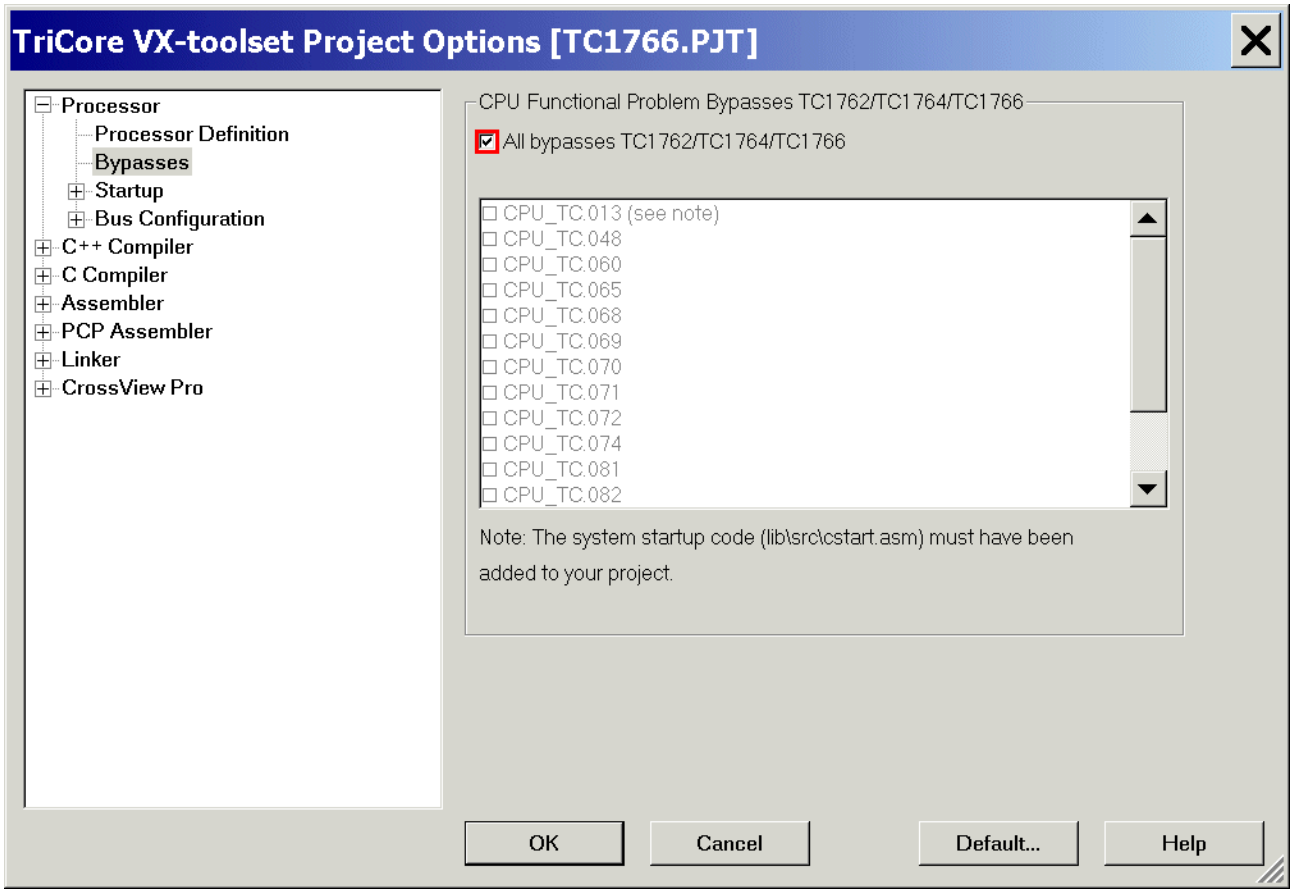
Configure Compiler, Assembler, Linker, Locater and Build – Control:

Project – Project Options

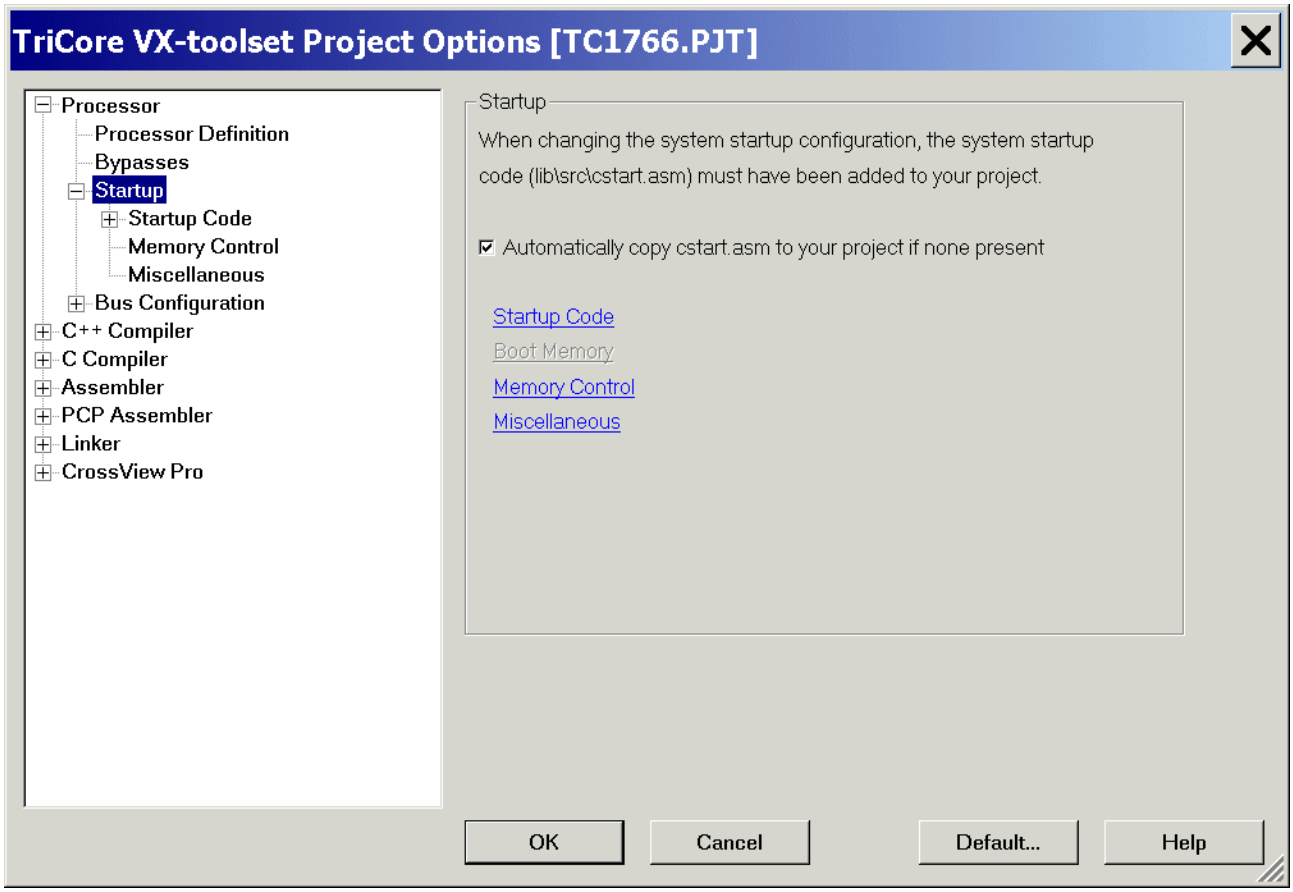
Processor: Processor Definition: Target processor: select TC1766B



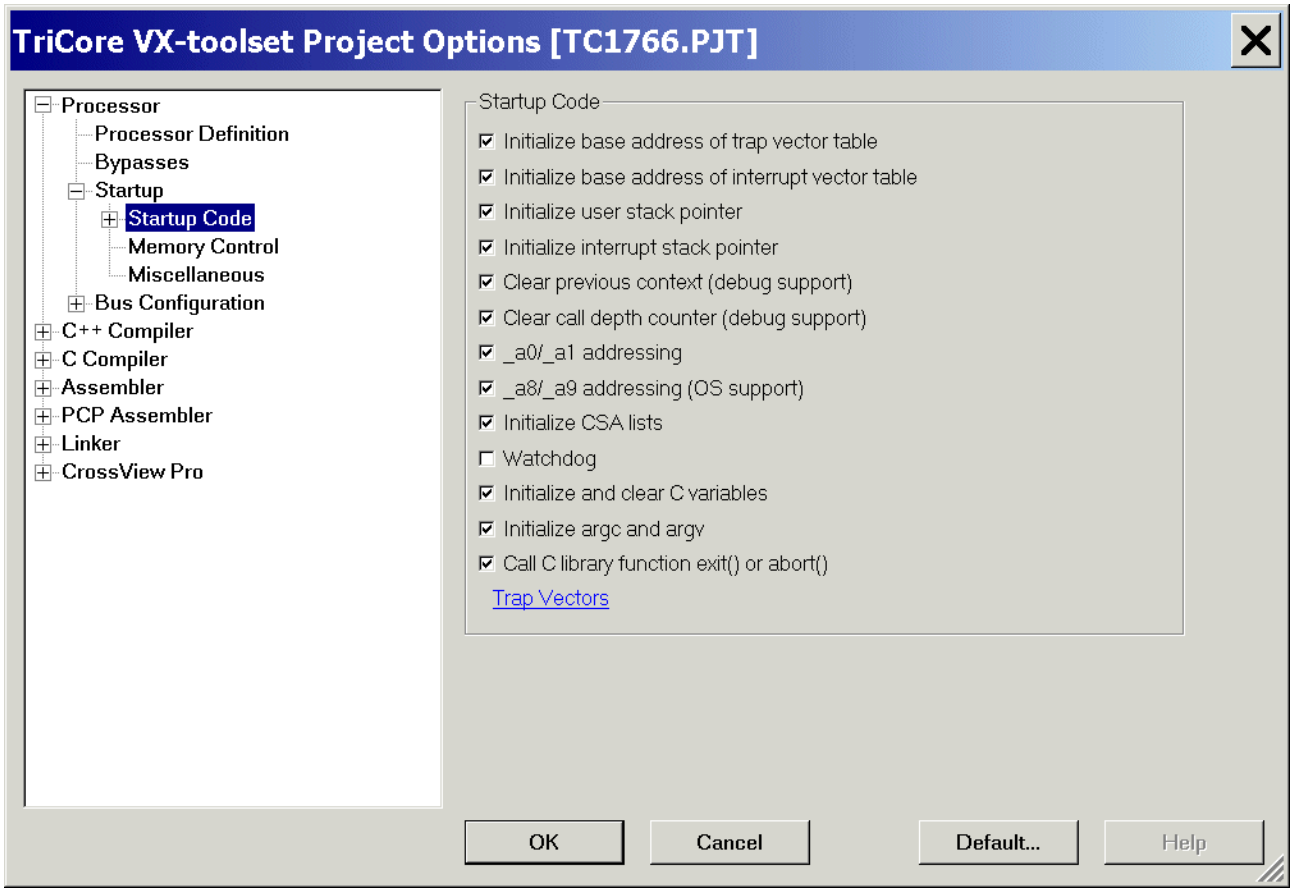
Processor: Bypasses: CPU Functional Problem Bypasses:
tick ✓ All bypasses TC1762/TC1764/TC1766



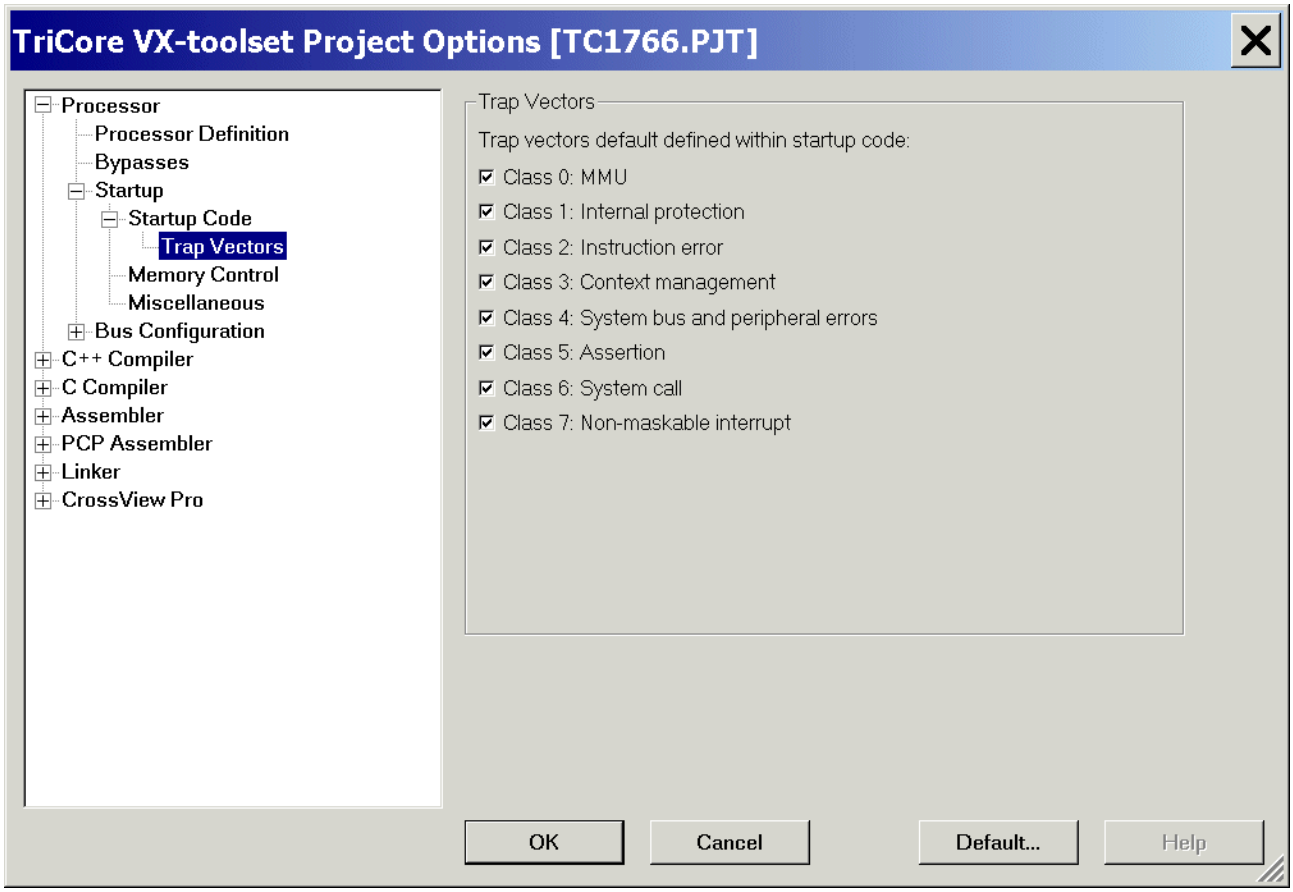
Processor: Startup: (do nothing)



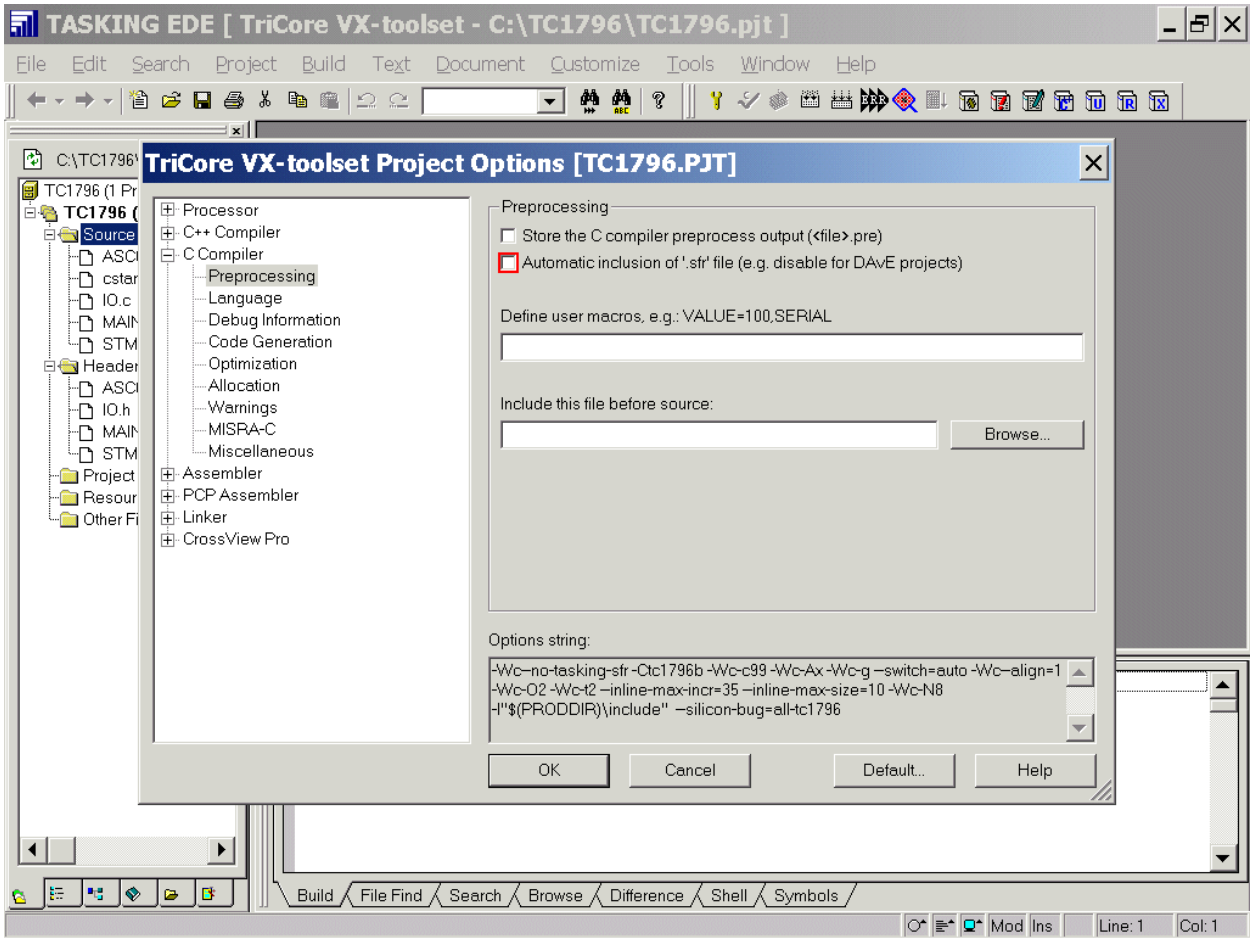
Processor: Startup: Startup Code: (do nothing)



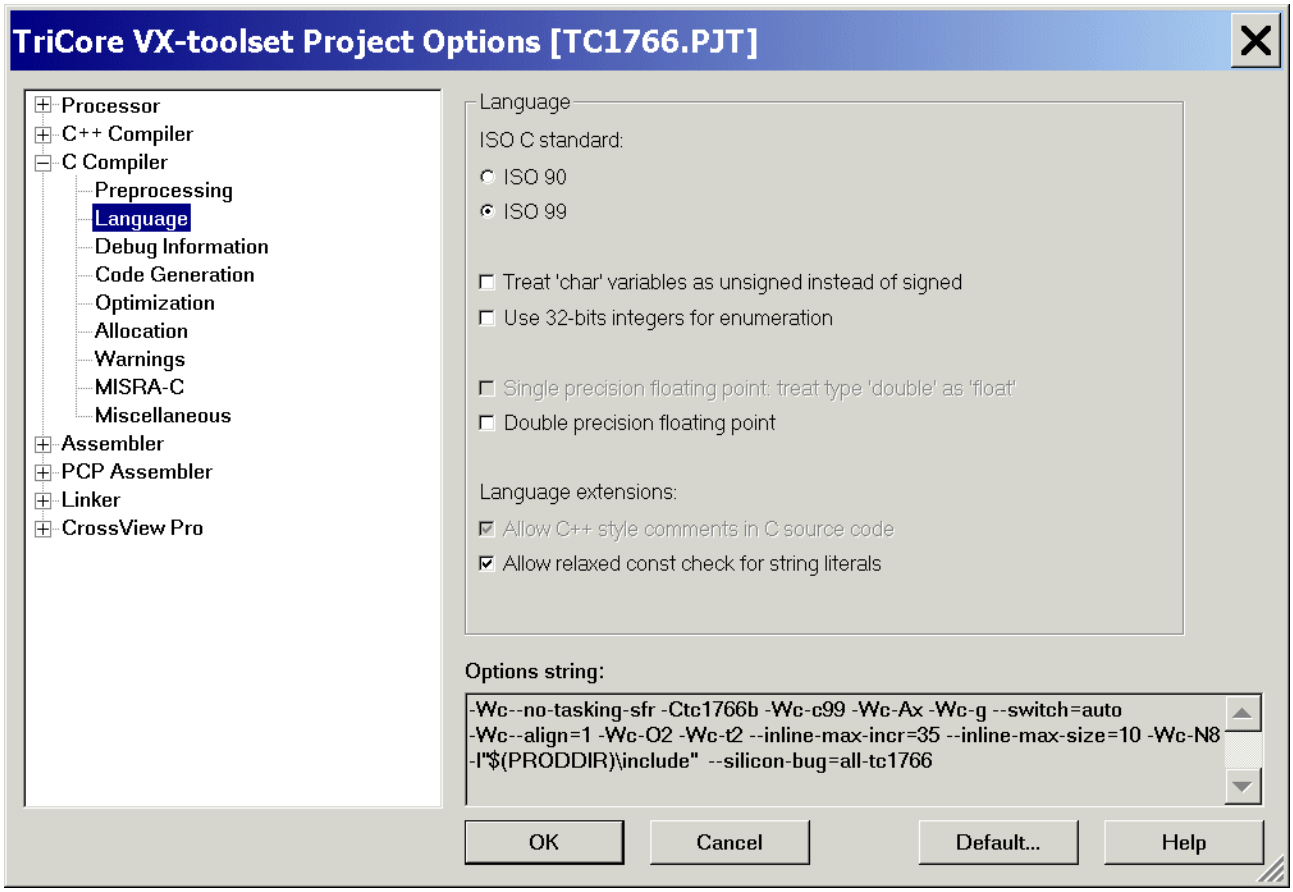
Processor: Startup: Startup Code: Trap Vectors: (do nothing)



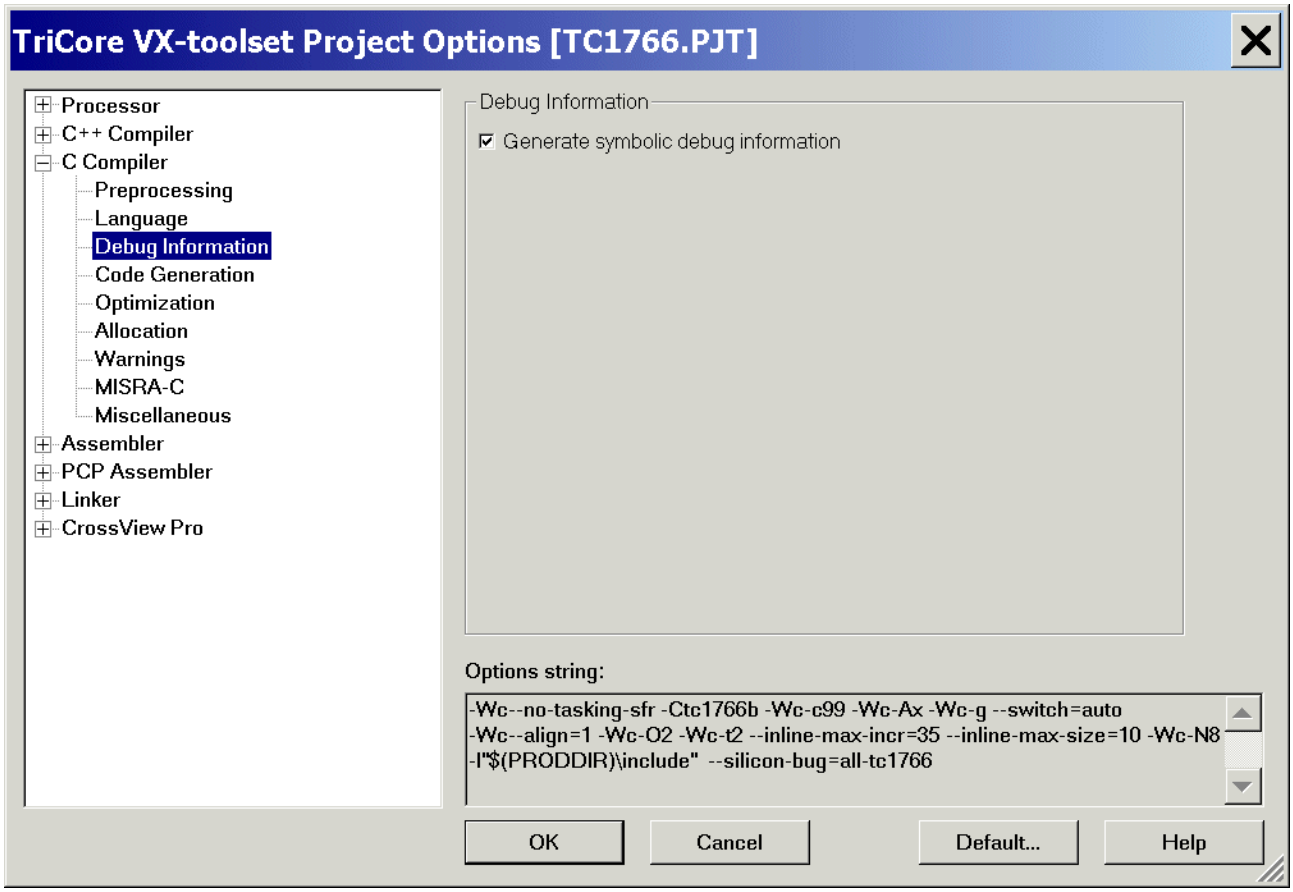
C Compiler: Preprocessing: deactivate (click to untick) ☐ Automatic inclusion of .sfr file



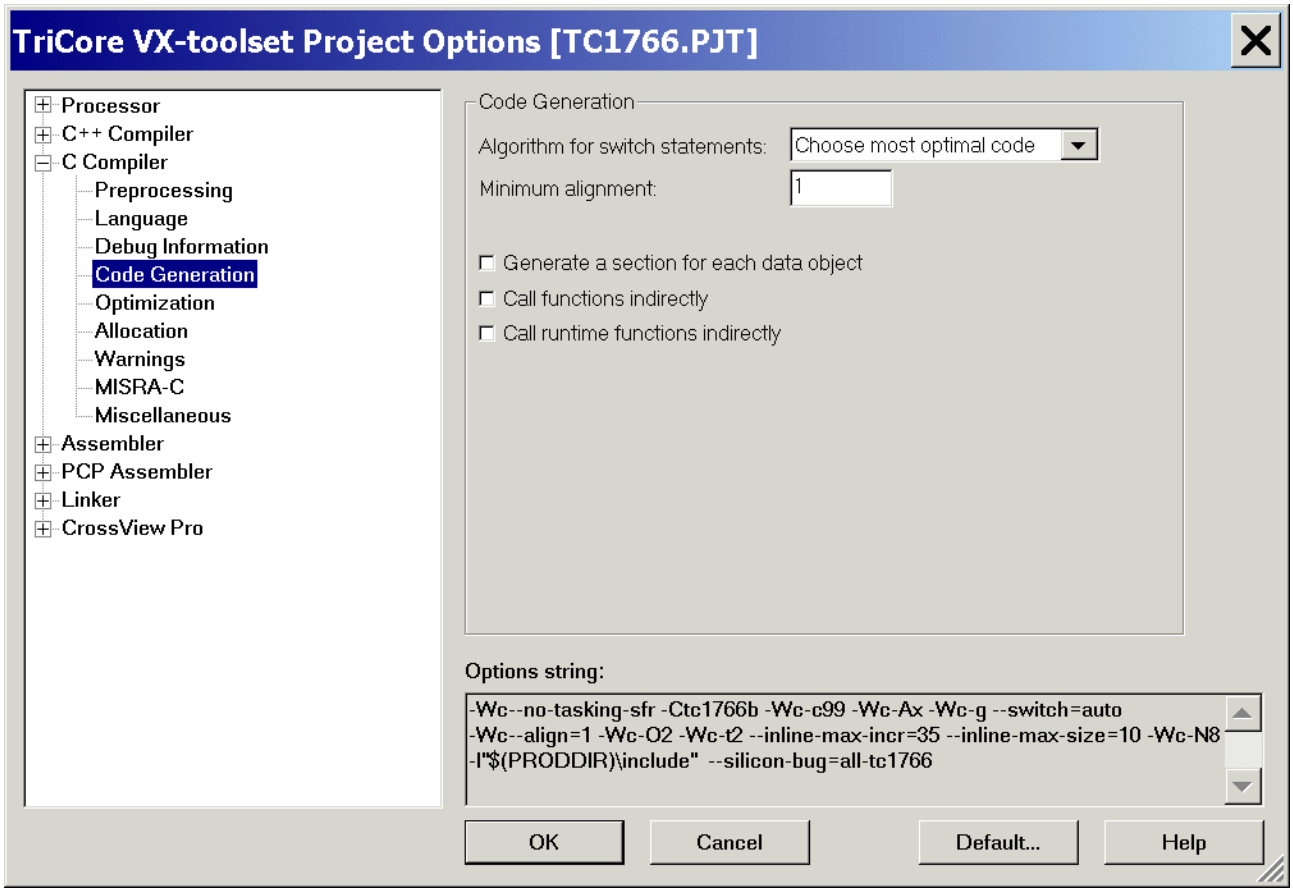
C Compiler: Language: (do nothing)



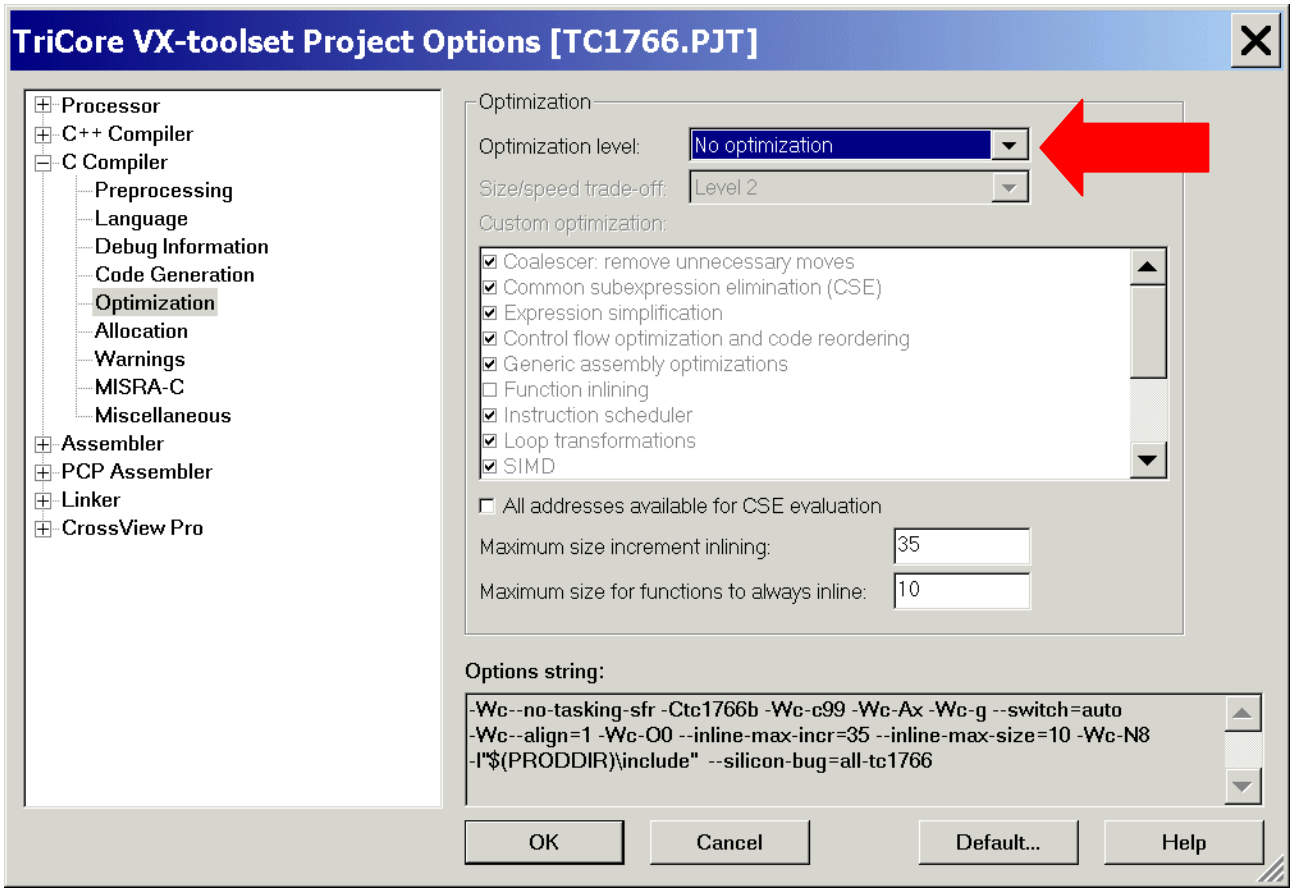
C Compiler: Debug Information: (do nothing)



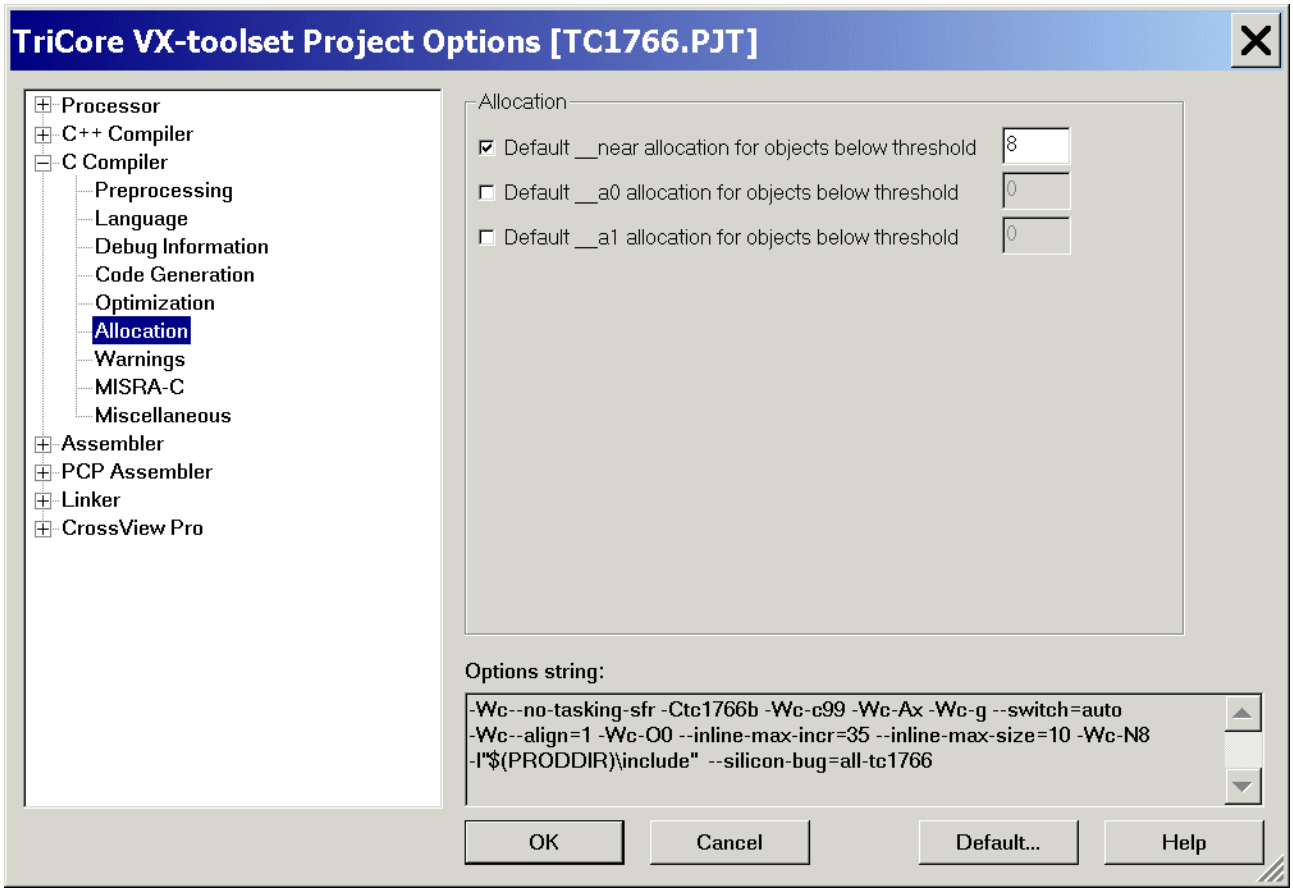
C Compiler: Code Generation: (do nothing)



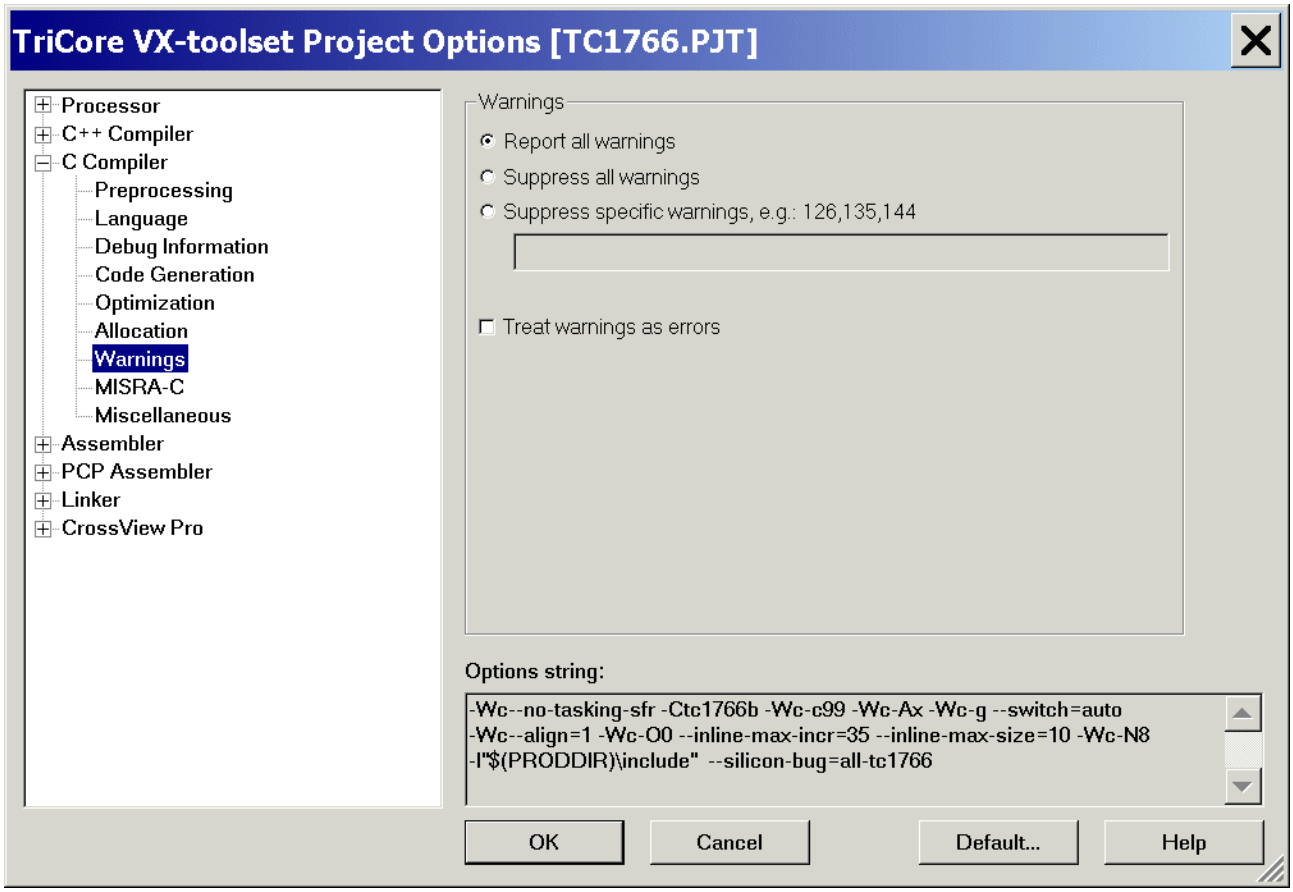
C Compiler: Optimization: Optimization level: select No optimization



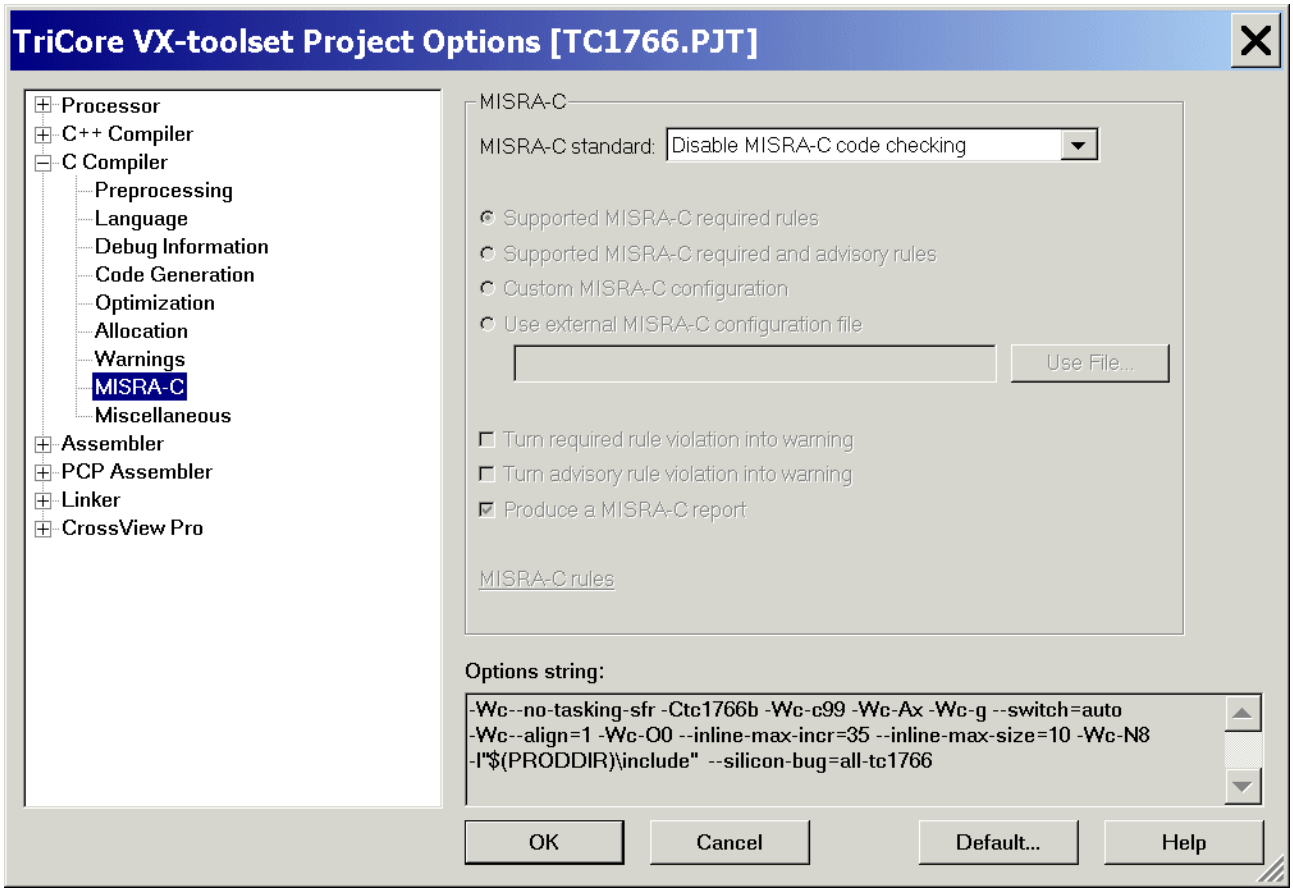
C Compiler: Allocation: (do nothing)



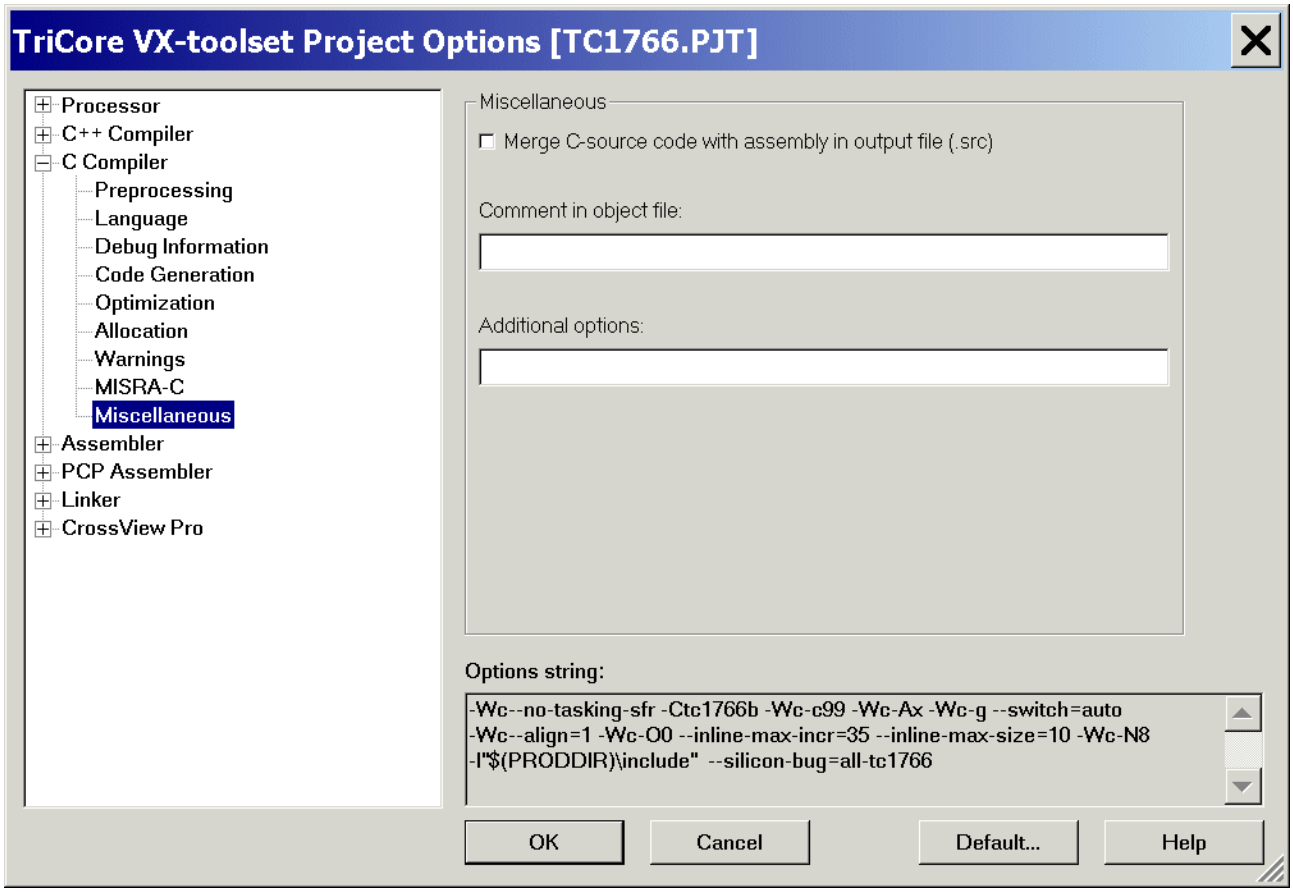
C Compiler: Warnings: (do nothing)



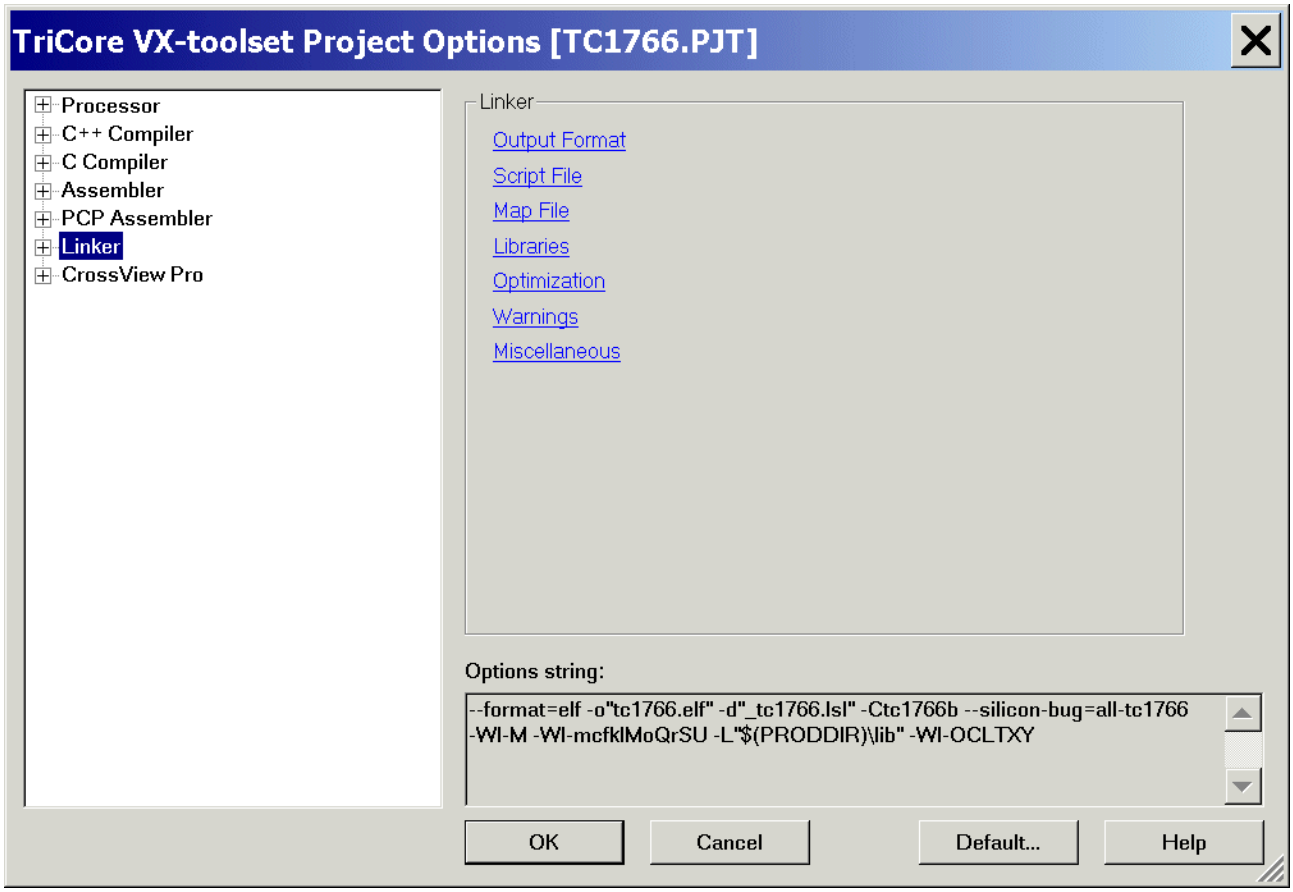
C Compiler: MISRA-C: (do nothing)



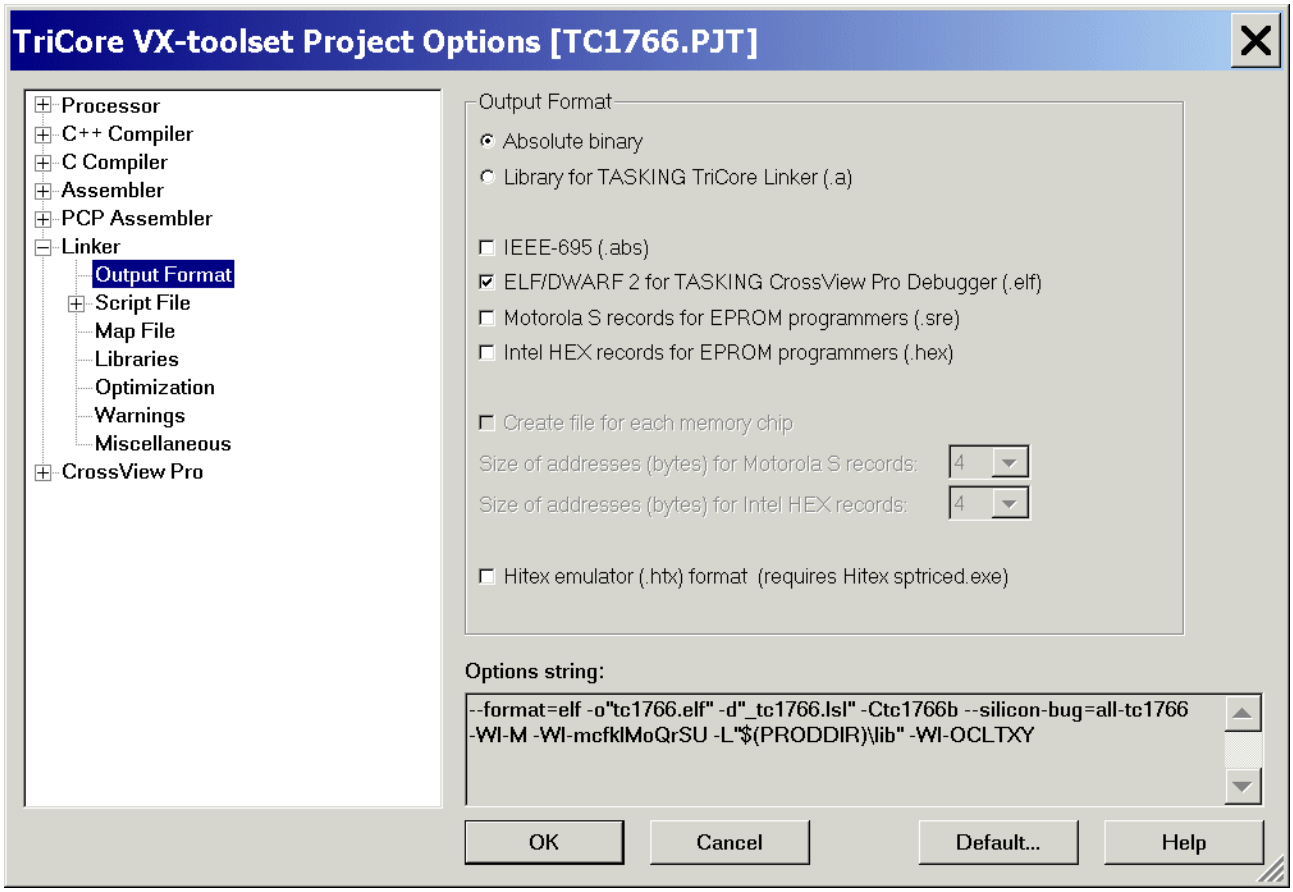
C Compiler: Miscellaneous: (do nothing)



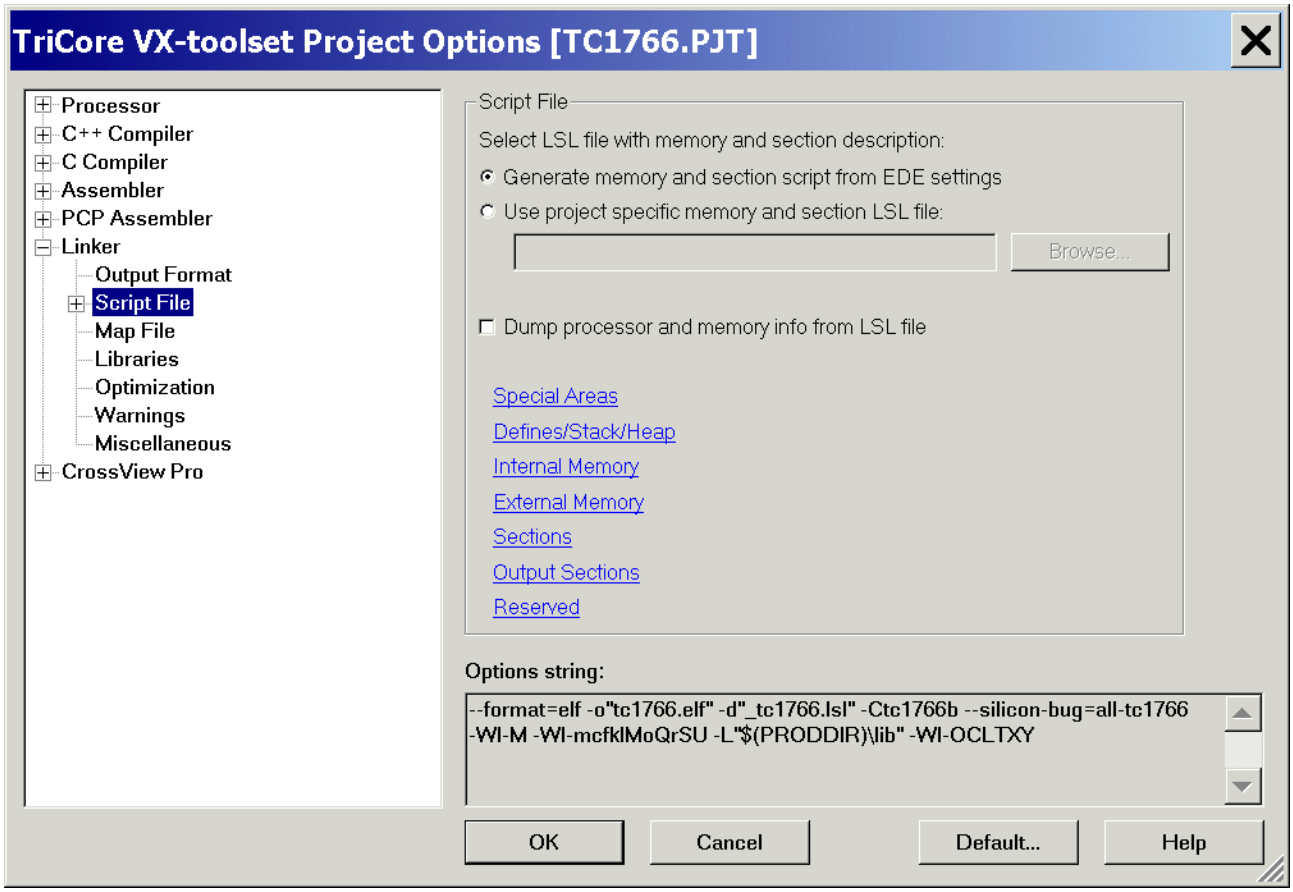
Linker: (do nothing)



Linker: Output Format: (do nothing)

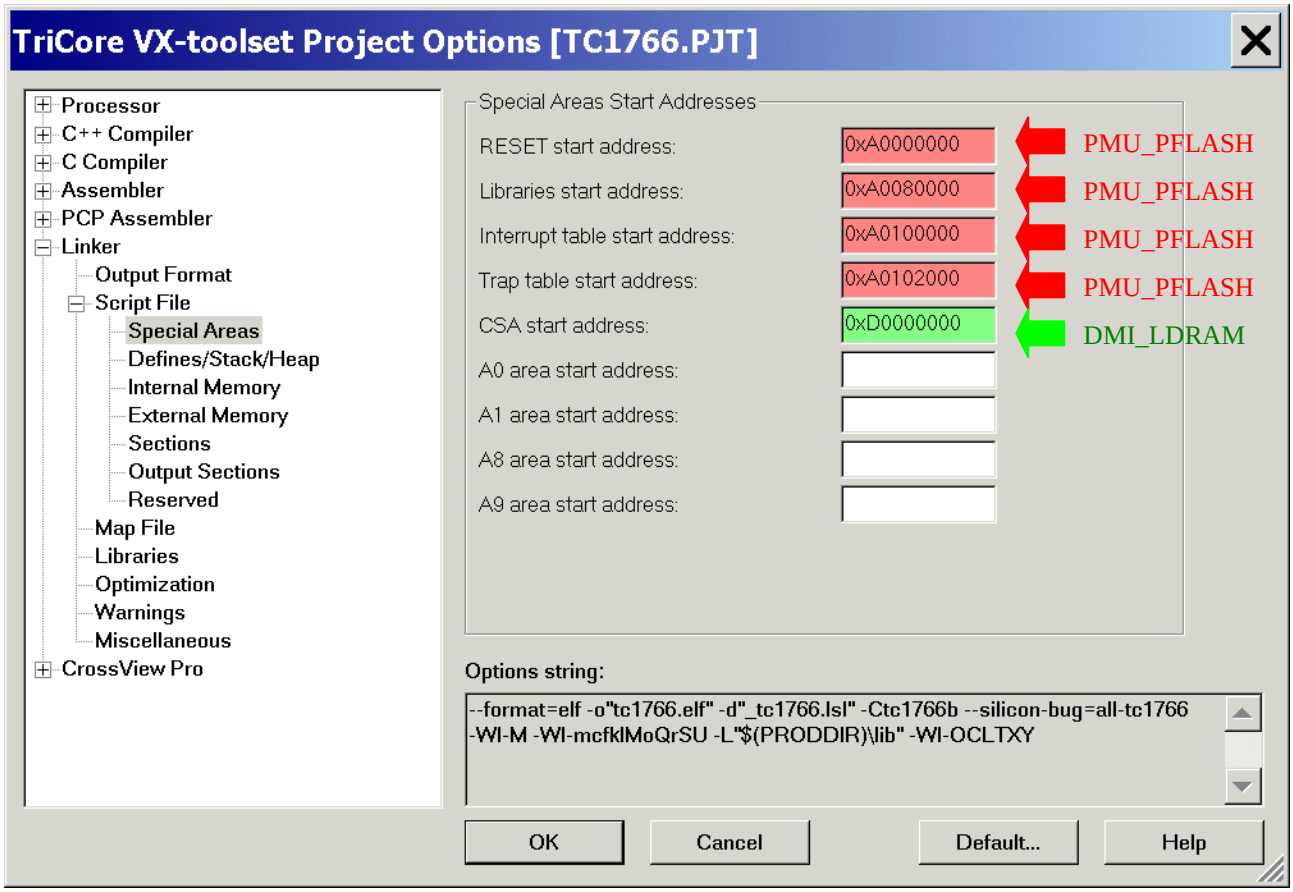


Linker: Script File: (do nothing)



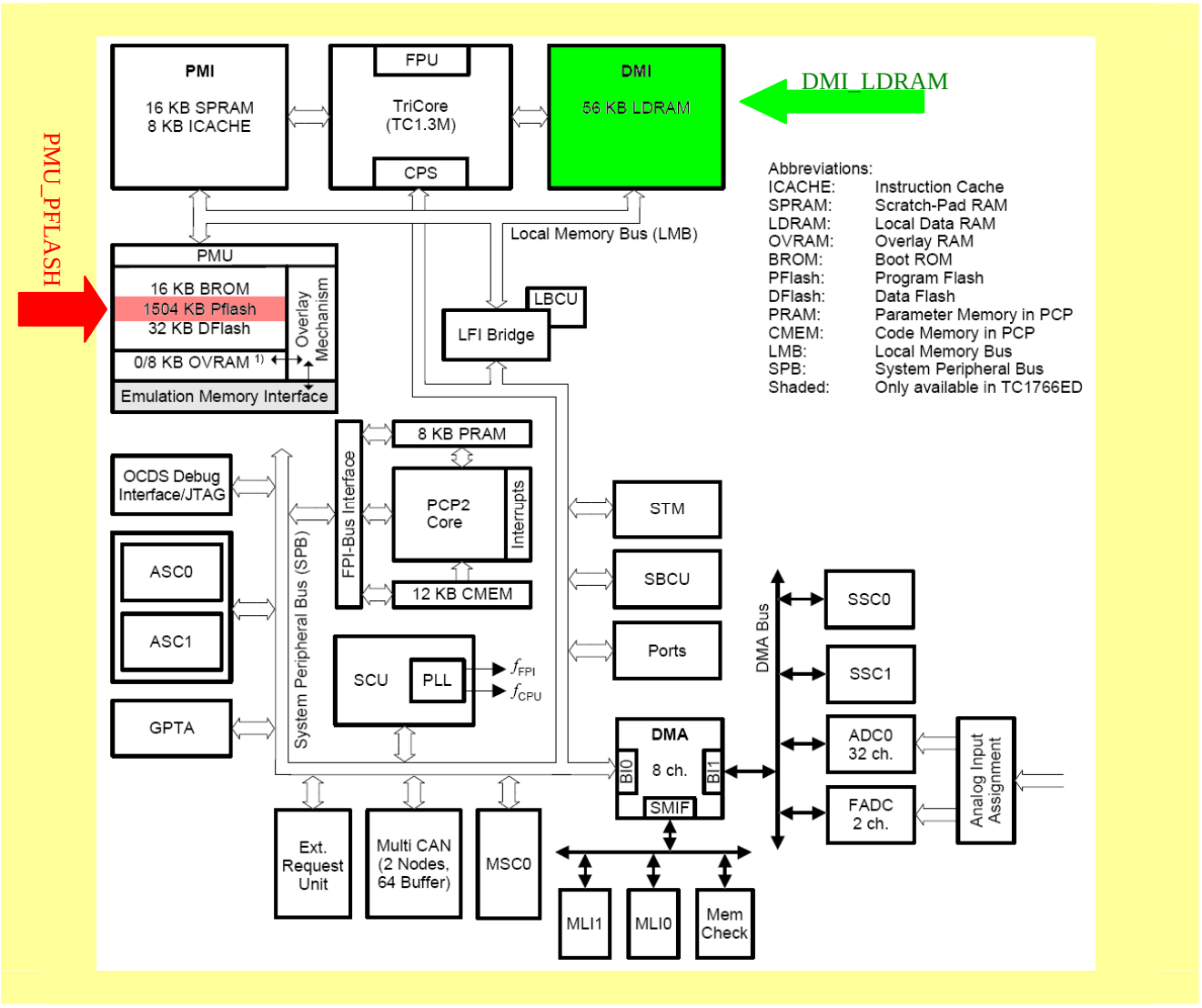
Linker: Script File: Special Areas: RESET start address: insert 0xA0000000 (PFLASH)
Linker: Script File: Special Areas: Libraries start address: insert 0xA0080000 (PFLASH)
Linker: Script File: Special Areas: Interrupt table start address: insert 0xA0100000 (PFLASH)
Linker: Script File: Special Areas: Trap table start address: insert 0xA0102000 (PFLASH)

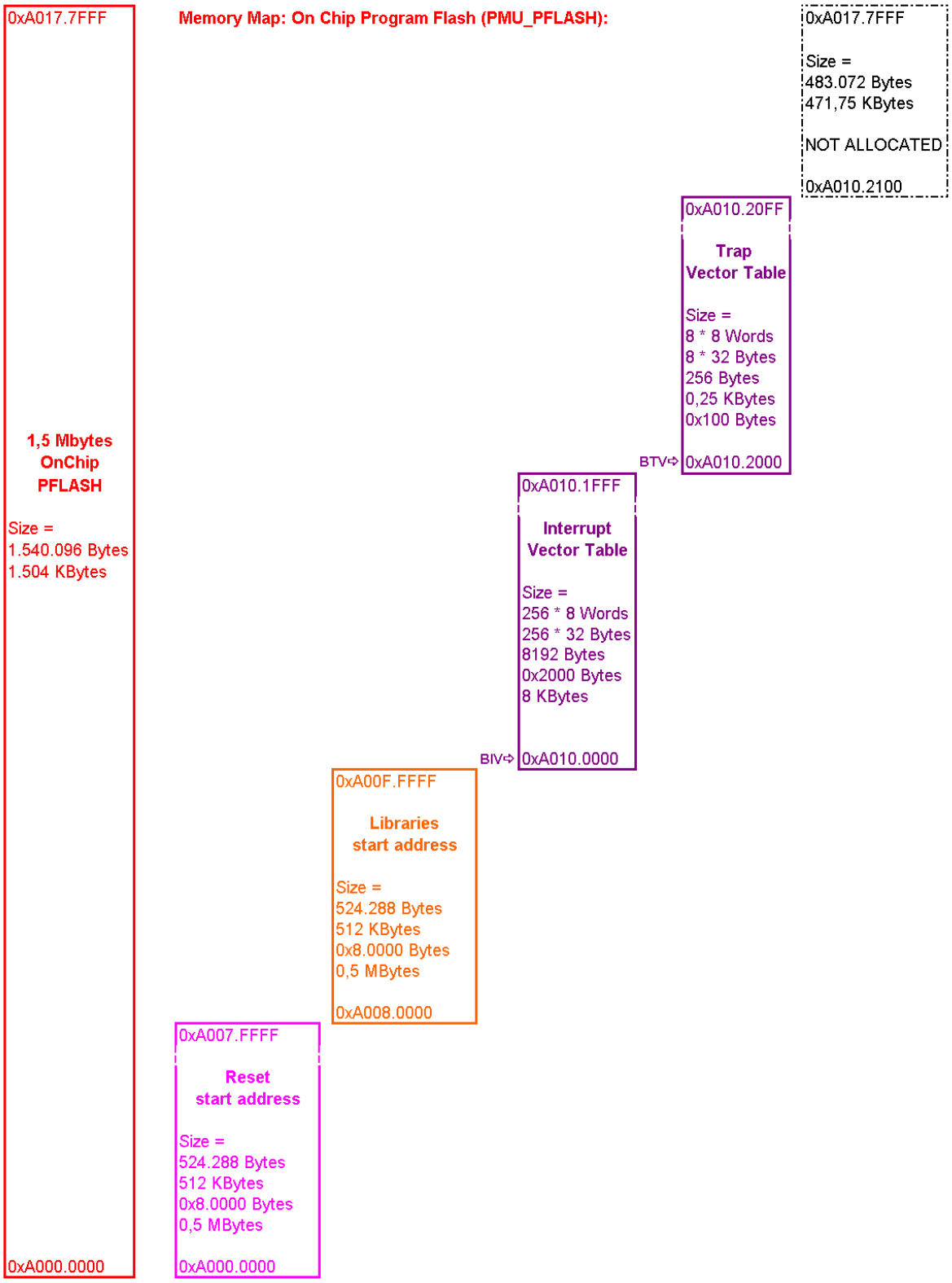
Linker: Script File: Special Areas: CSA start address: insert/check 0xD0000000 (LDRAM)



Additional information: Program Memory / Data Memory:

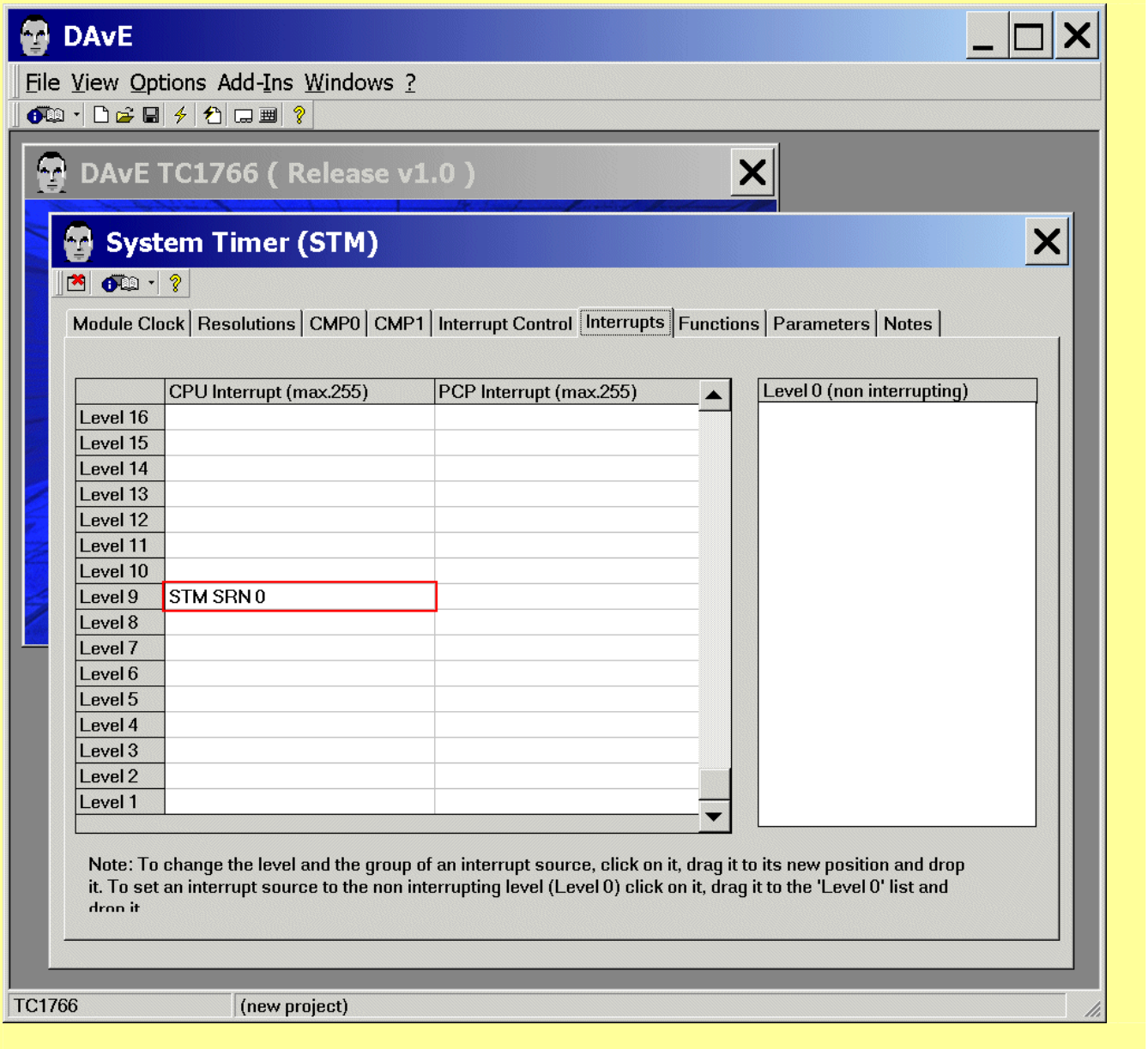
The On Chip PMU_PFLASH memory has a capacity of 1.504 KBytes:
The On Chip DMI_LDRAM memory has a capacity of 56 KBytes.




Memory Map: On Chip Program Flash (PMU_PFLASH):
0xA017.7FFF
1,5 Mbytes
OnChip
PFLASH
Size =
1.540.096 Bytes
1.504 KBytes
0xA000.0000
0xA017.7FFF
Size =
483.072 Bytes
471,75 KBytes
NOT ALLOCATED
0xA010.2100
0xA010.20FF
Trap
Vector Table
Size =
8 * 8 Words
8 * 32 Bytes
256 Bytes
0,25 KBytes
0x100 Bytes
BTV
0xA010.2000
0xA010.1FFF
Interrupt
Vector Table
Size =
256 * 8 Words
256 * 32 Bytes
8192 Bytes
0x2000 Bytes
8 KBytes
BIV
0xA010.0000
0xA00F.FFFF
Libraries
start address
Size =
524.288 Bytes
512 KBytes
0x8.0000 Bytes
0,5 MBytes
0xA008.0000
0xA007.FFFF
Reset
start address
Size =
524.288 Bytes
512 KBytes
0x8.0000 Bytes
0,5 MBytes
0xA000.0000

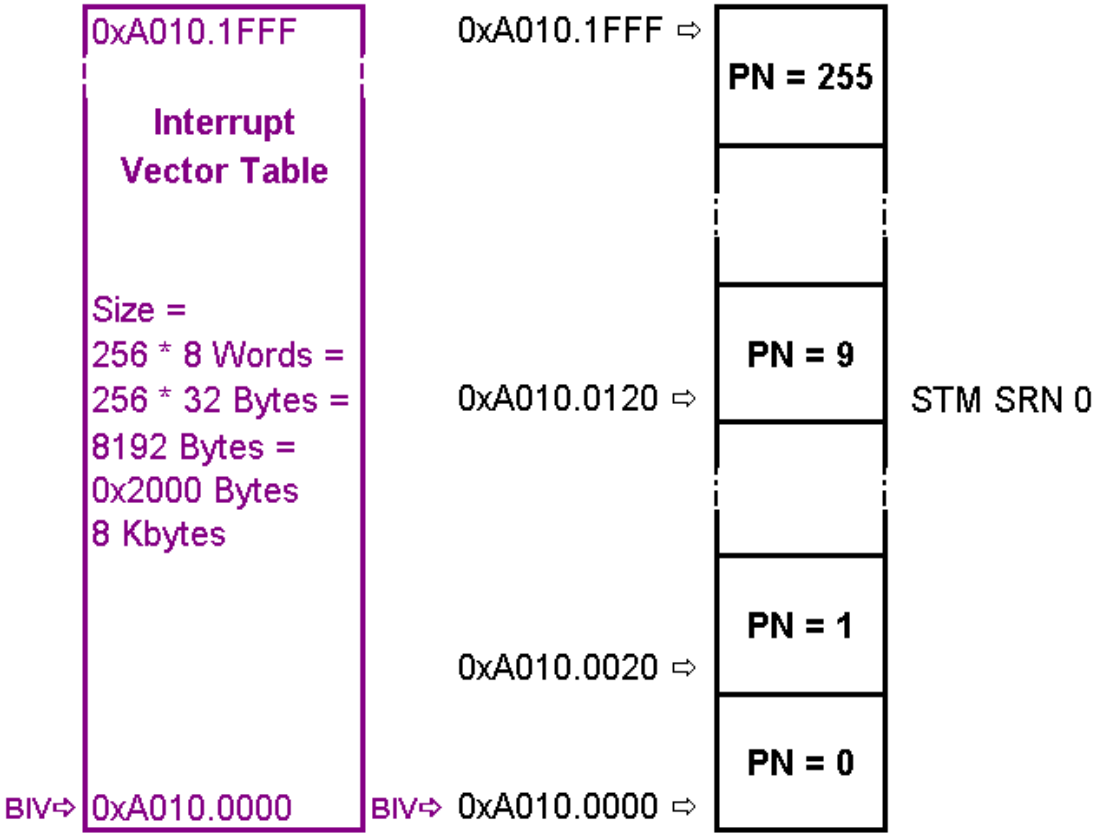
Additional information: Interrupt Vector Table:

Remember:



Additional information: Interrupt Vector Table:

Interrupt Vector Table:

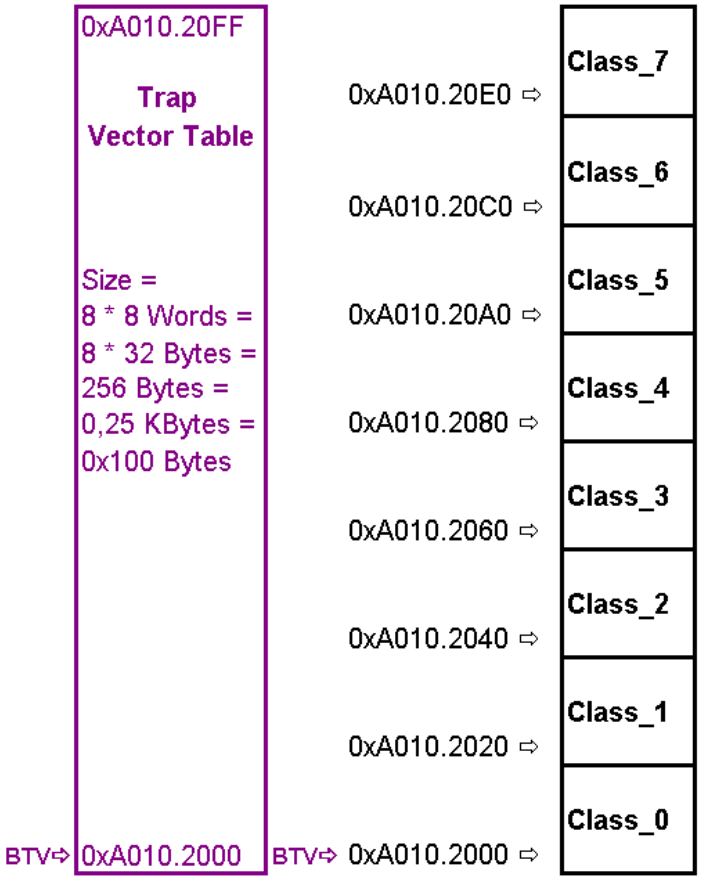


Note:
PN … Priority Number (CPU Interrupt Level)

Note:
Click here to see the Map File

Additional information: TRAP Vector Table:

TRAP Vector Table:

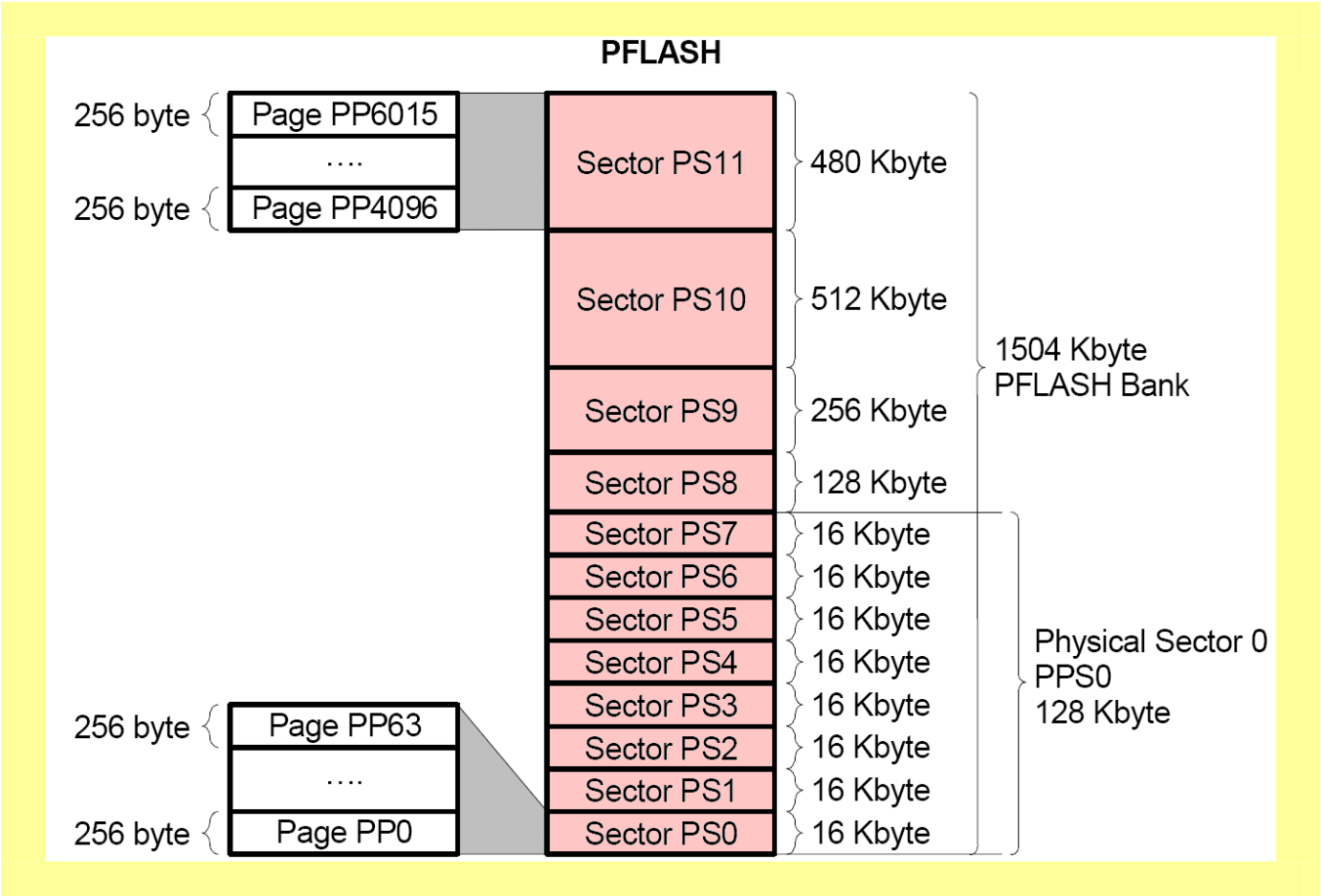


Note:
1 Word = 32 Bits
1 Word = 4 Bytes
8 Words = 32 Bytes

Note:
Click here to see the Map File

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

The on-chip PMU_PFLASH memory has a capacity of 1.504 KBytes:



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

The on-chip PMU_PFLASH memory has a capacity of 1.504 KBytes:

Numbering		Size	Cached Address Range	Non-Cached Address Range
PFLASH Bank				
PB		1504 Kbyte	$8000\ 0000_H$ - $8017\ 7FFF_H$	$A000\ 0000_H$ - $A017\ 7FFF_H$
PFLASH Sectors				
PS0	PPS0[1)]	16 Kbyte	$8000\ 0000_H$ - $8000\ 3FFF_H$	$A000\ 0000_H$ - $A000\ 3FFF_H$
PS1		16 Kbyte	$8000\ 4000_H$ - $8000\ 7FFF_H$	$A000\ 4000_H$ - $A000\ 7FFF_H$
PS2		16 Kbyte	$8000\ 8000_H$ - $8000\ BFFF_H$	$A000\ 8000_H$ - $A000\ BFFF_H$
PS3		16 Kbyte	$8000\ C000_H$ - $8000\ FFFF_H$	$A000\ C000_H$ - $A000\ FFFF_H$
PS4		16 Kbyte	$8001\ 0000_H$ - $8001\ 3FFF_H$	$A001\ 0000_H$ - $A001\ 3FFF_H$
PS5		16 Kbyte	$8001\ 4000_H$ - $8001\ 7FFF_H$	$A001\ 4000_H$ - $A001\ 7FFF_H$
PS6		16 Kbyte	$8001\ 8000_H$ - $8001\ BFFF_H$	$A001\ 8000_H$ - $A001\ BFFF_H$
PS7		16 Kbyte	$8001\ C000_H$ - $8001\ FFFF_H$	$A001\ C000_H$ - $A001\ FFFF_H$
PS8		128 Kbyte	$8002\ 0000_H$ - $8003\ FFFF_H$	$A002\ 0000_H$ - $A003\ FFFF_H$
PS9		256 Kbyte	$8004\ 0000_H$ - $8007\ FFFF_H$	$A004\ 0000_H$ - $A007\ FFFF_H$
PS10		512 Kbyte	$8008\ 0000_H$ - $800F\ FFFF_H$	$A008\ 0000_H$ - $A00F\ FFFF_H$
PS11		480 Kbyte	$8010\ 0000_H$ - $8017\ 7FFF_H$	$A010\ 0000_H$ - $A017\ 7FFF_H$

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

The on-chip PMU_PFLASH memory has a capacity of 1.504 KBytes:

Adobe Reader - [TC1766_um_v2[1].0_2007_07.pdf]

File Edit View Document Tools Window Help

8.3.1 Segments 0 to 14

Table 8-2 shows the address map of segments 0 to 14 as it is seen from the SPB bus masters PCP, DMA and OCDS.

Table 8-2 SPB Address Map of Segment 0 to 14

Segment	Address Range	Size	Description	Access Type	
				Read	Write
0-7	$0000\ 0000_H$ - $0000\ 0007_H$	8 byte	Reserved (virtual address space)	MPN trap	MPN trap
	$0000\ 0008_H$ - $7FFF\ FFFF_H$	8 × 256 Mbyte		SPBBE	SPBBE
8	$8000\ 0000_H$ - $8017\ 7FFF_H$	1.5 Mbyte	Program Flash (PFLASH)	access	access[1)]
	$8017\ 8000_H$ - $807F\ FFFF_H$	6.5 Mbyte	Reserved	LMBBE & SPBBE	LMBBE
	$8080\ 0000_H$ - $8FDF\ FFFF_H$	246 Mbyte	Reserved	LMBBE & SPBBE	LMBBE
	$8FE0\ 0000_H$ - $8FE0\ 3FFF_H$	16 Kbyte	Data Flash (DFLASH) Bank 0	access	access[1)]

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

The on-chip PMU_PFLASH memory has a capacity of 1.504 KBytes:

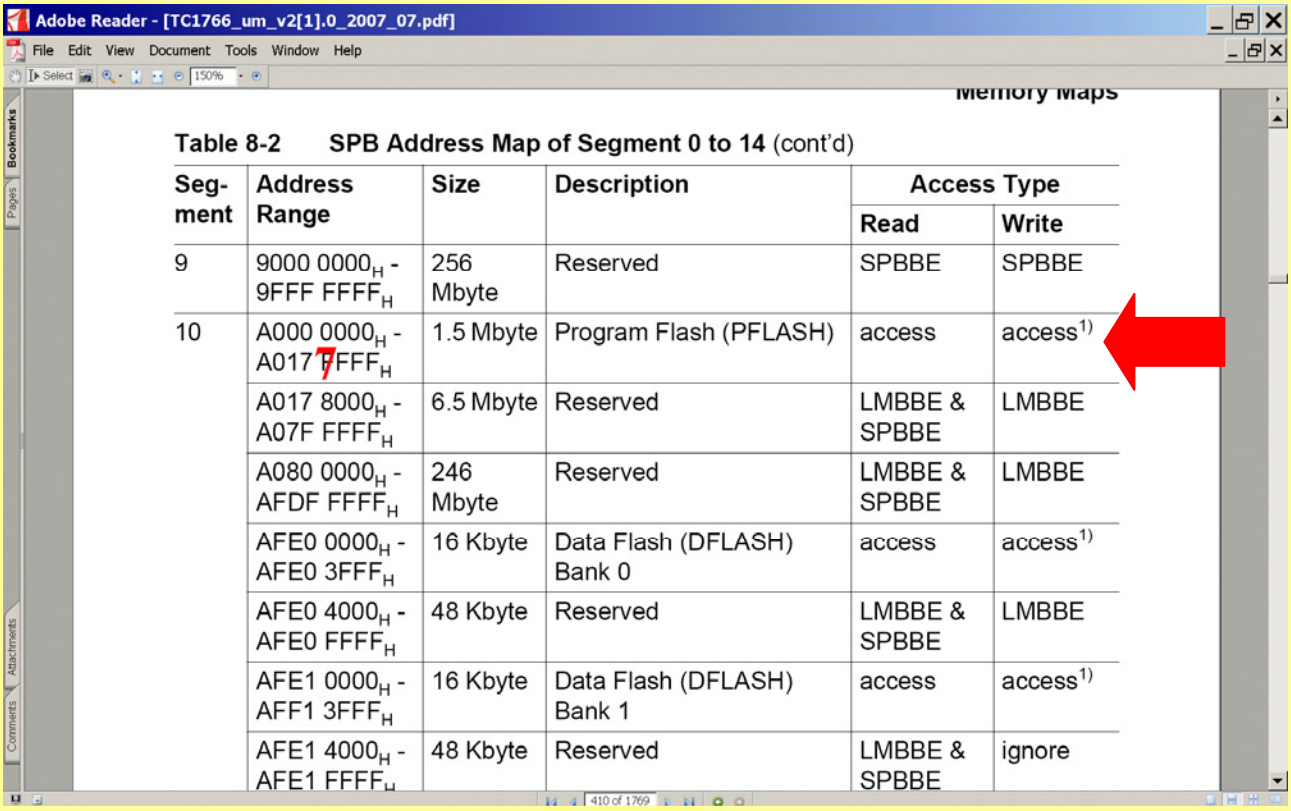
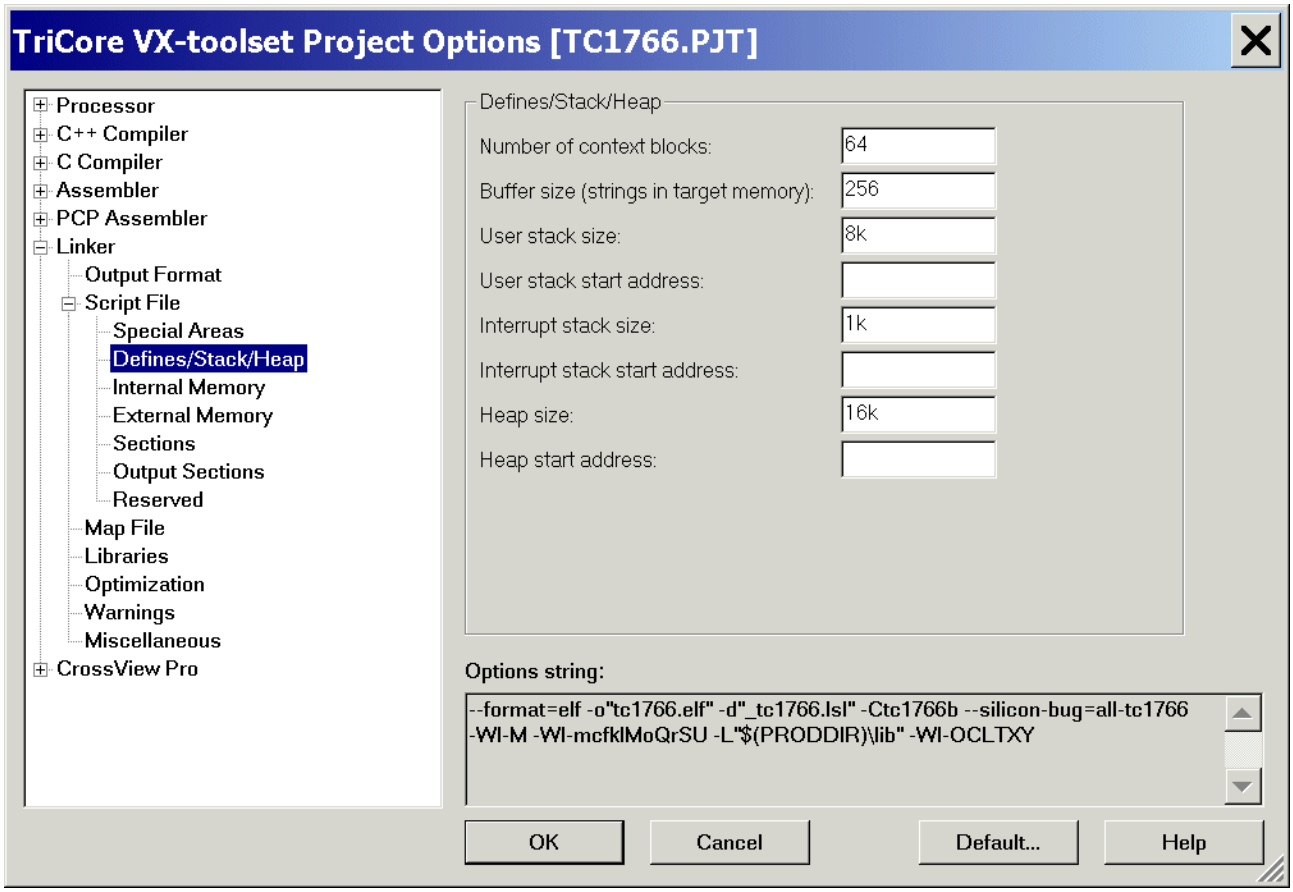


Table 8-2 SPB Address Map of Segment 0 to 14 (cont'd)

Segment	Address Range	Size	Description	Access Type	
				Read	Write
9	9000 0000$_H$ - 9FFF FFFF$_H$	256 Mbyte	Reserved	SPBBE	SPBBE
10	A000 0000$_H$ - A017 7FFF$_H$	1.5 Mbyte	Program Flash (PFLASH)	access	access[1)]
	A017 8000$_H$ - A07F FFFF$_H$	6.5 Mbyte	Reserved	LMBBE & SPBBE	LMBBE
	A080 0000$_H$ - AFDF FFFF$_H$	246 Mbyte	Reserved	LMBBE & SPBBE	LMBBE
	AFE0 0000$_H$ - AFE0 3FFF$_H$	16 Kbyte	Data Flash (DFLASH) Bank 0	access	access[1)]
	AFE0 4000$_H$ - AFE0 FFFF$_H$	48 Kbyte	Reserved	LMBBE & SPBBE	LMBBE
	AFE1 0000$_H$ - AFF1 3FFF$_H$	16 Kbyte	Data Flash (DFLASH) Bank 1	access	access[1)]
	AFE1 4000$_H$ - AFE1 FFFF$_H$	48 Kbyte	Reserved	LMBBE & SPBBE	ignore

Note:
There is a typing error in Table 8-2, page 8-6, TC1766 User's Manual, System Units (Vol. 1 of 2).
The correct address range for the 1.504 Kbyte PMU_PFLASH is A000 0000$_H$ - A017 7FFF$_H$.

Linker: Script File: Defines/Stack/Heap: (do nothing)



Linker: Script File: Internal Memory: change from brom to PMU_BROM

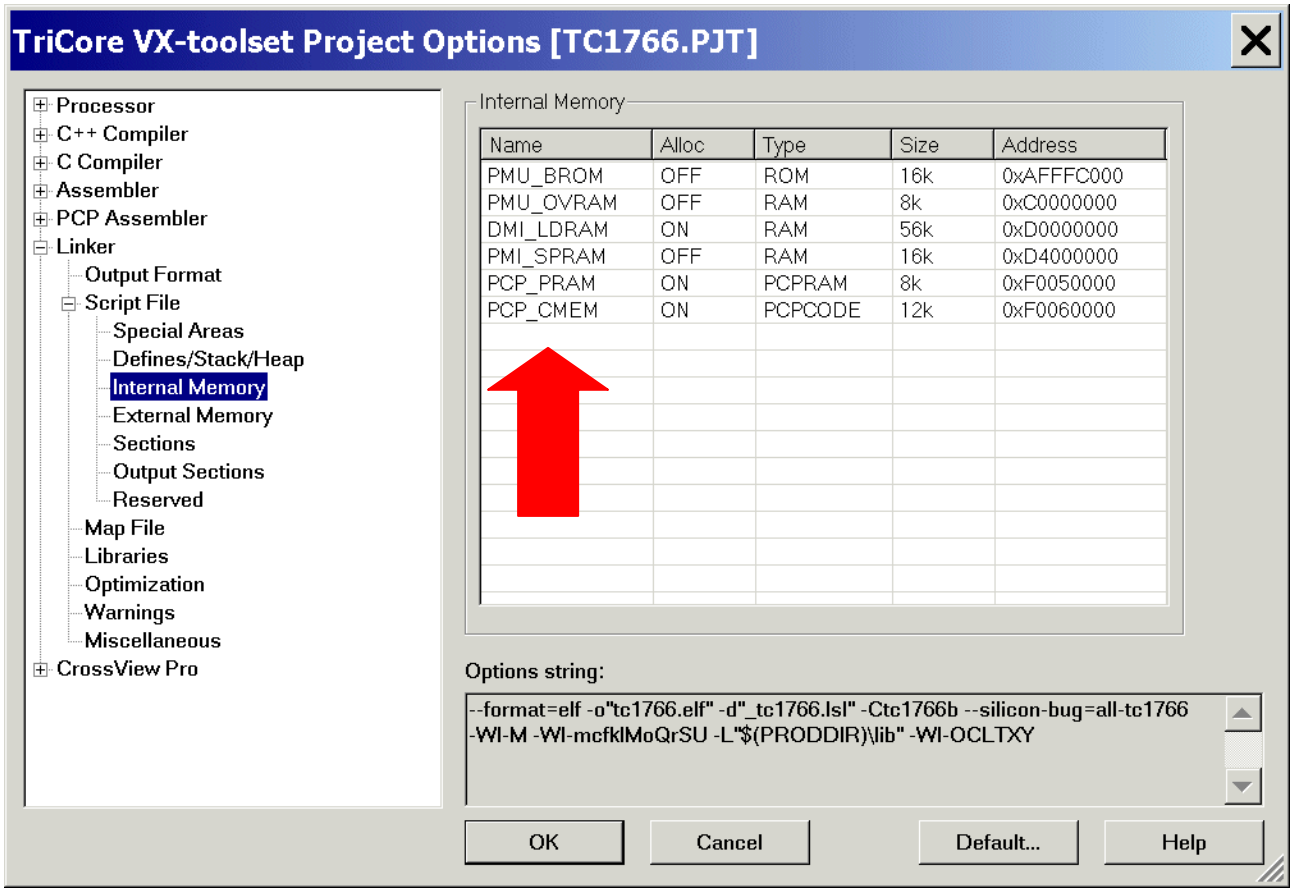
Linker: Script File: Internal Memory: change from ovram to PMU_OVRAM

Linker: Script File: Internal Memory: change from ldram to DMI_LDRAM

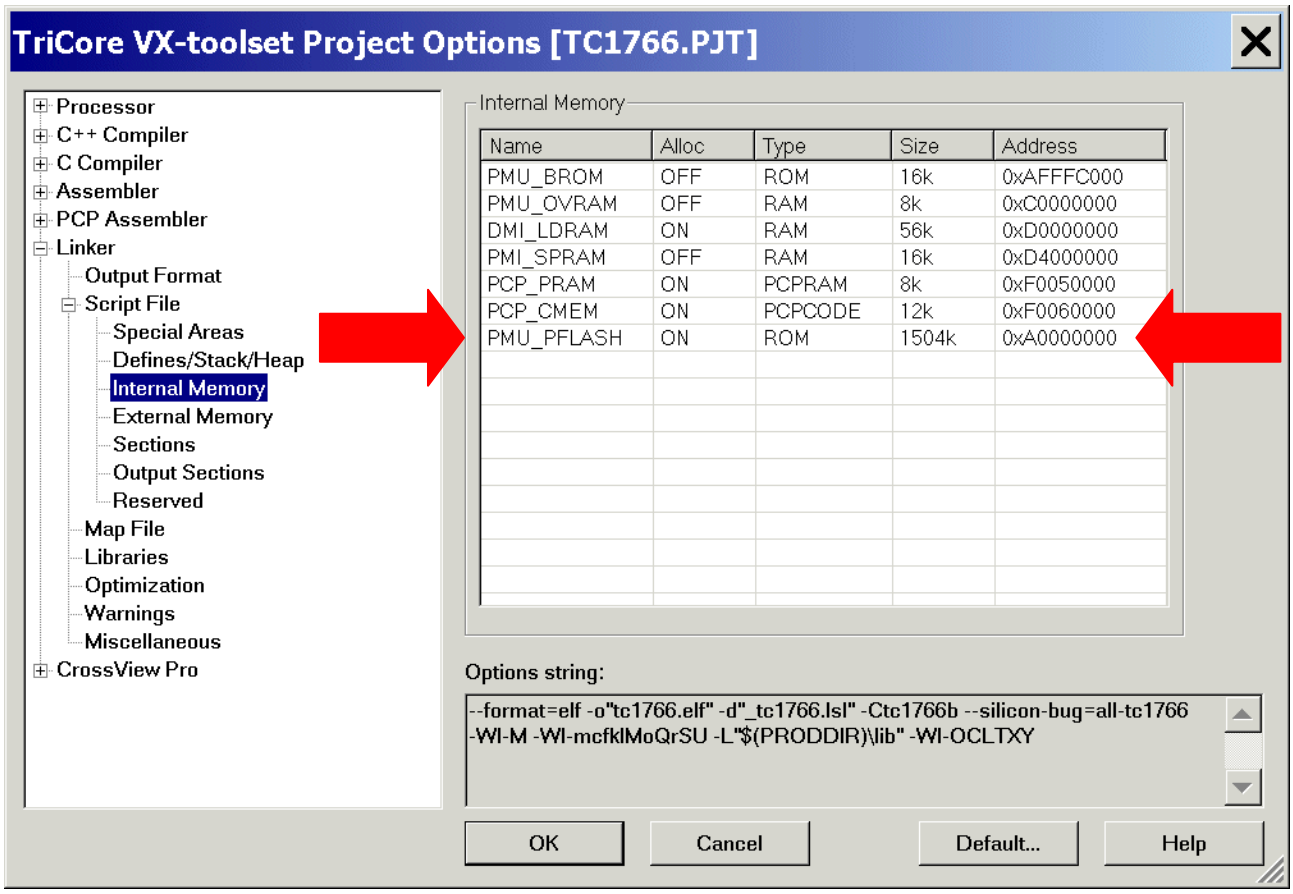
Linker: Script File: Internal Memory: change from spram to PMI_SPRAM

Linker: Script File: Internal Memory: change from pram to PCP_PRAM

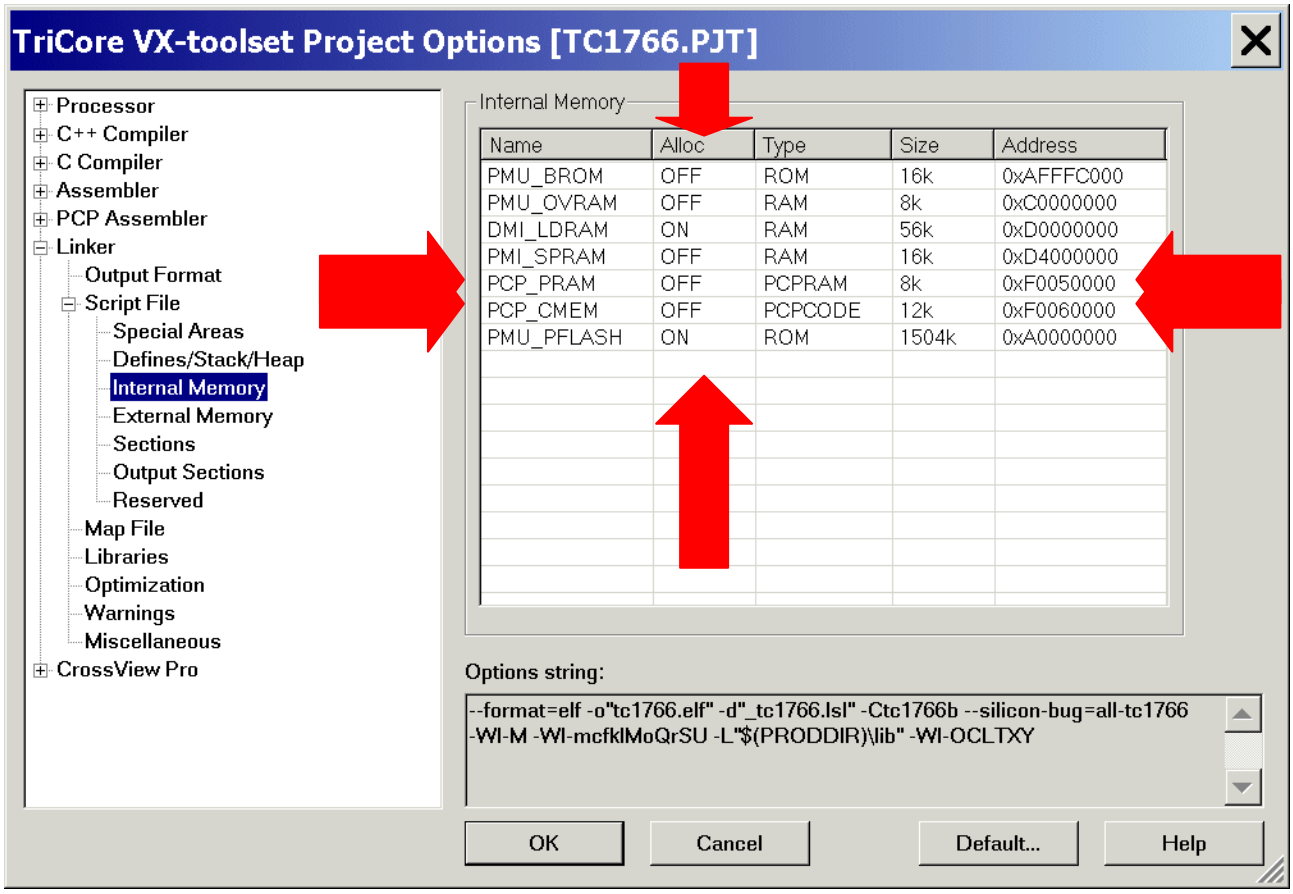
Linker: Script File: Internal Memory: change from pcode to PCP_CMEM



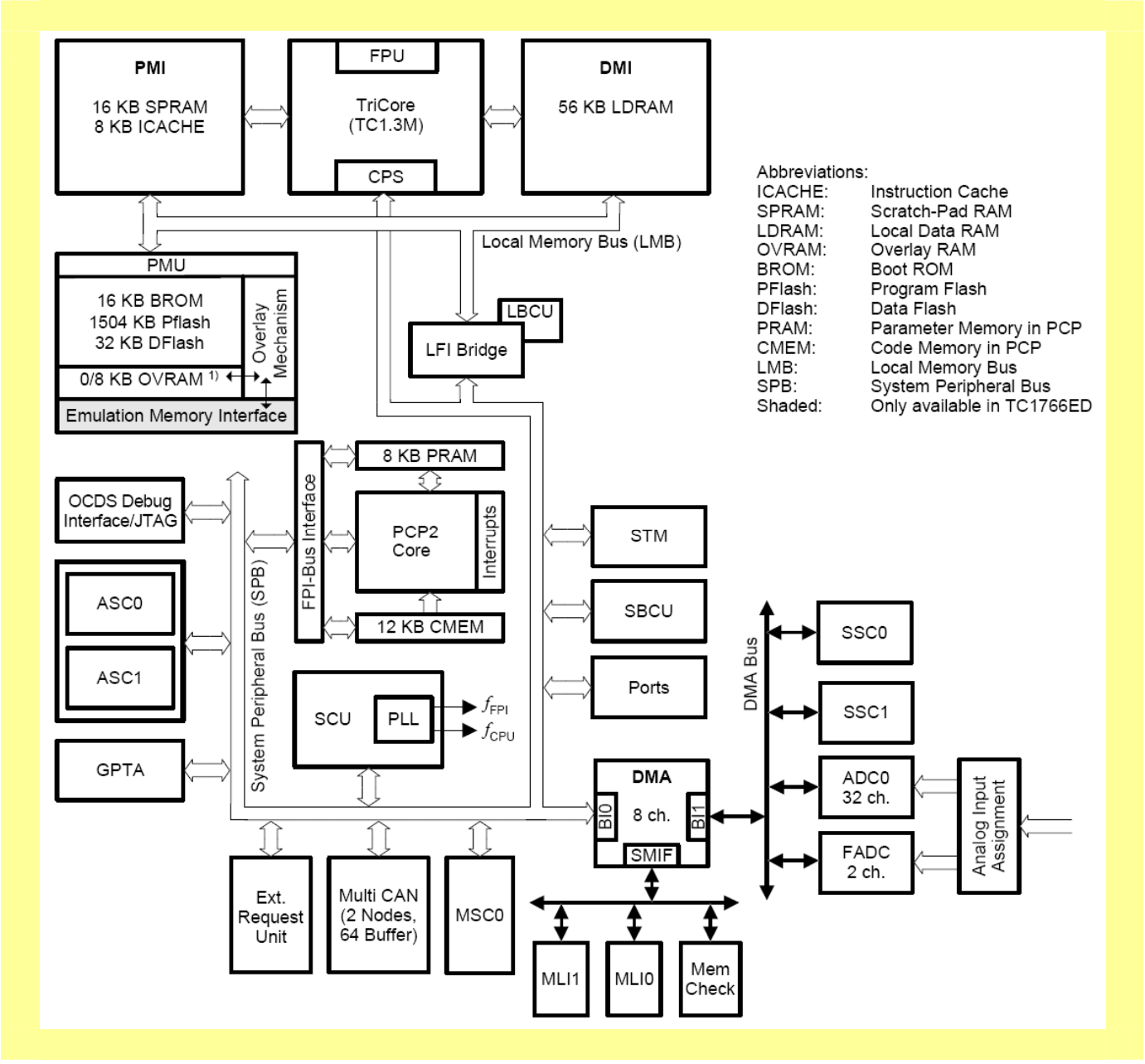
Linker: Script File: Internal Memory: Name: insert PMU_PFLASH
Linker: Script File: Internal Memory: Alloc: select ON
Linker: Script File: Internal Memory: Type: select ROM
Linker: Script File: Internal Memory: Size: insert 1504k
Linker: Script File: Internal Memory: Address insert 0xA0000000



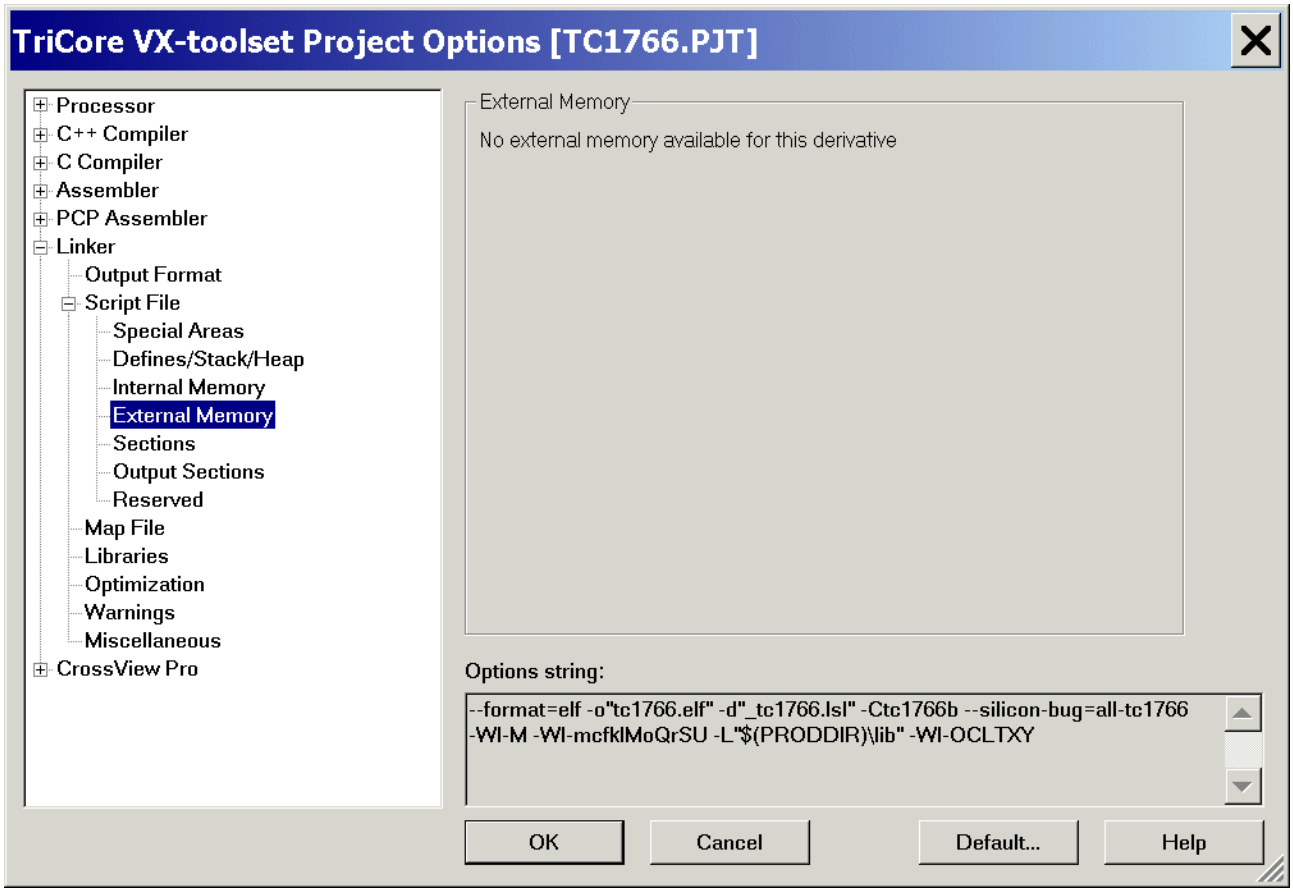
Linker: Script File: Internal Memory: Name=PCP_PRAM: Alloc: select OFF
Linker: Script File: Internal Memory: Name=PCP_CMEM: Alloc: select OFF



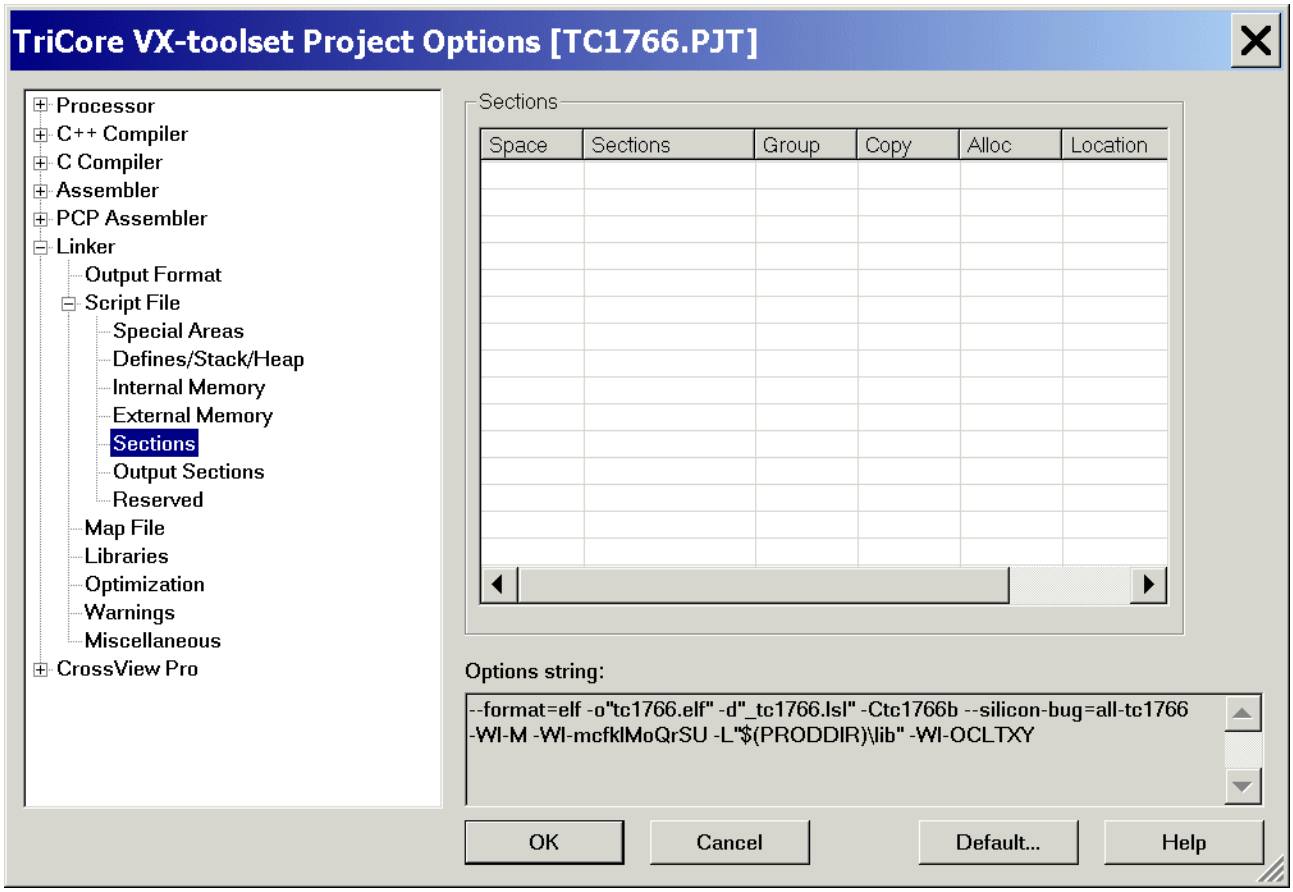
Additional information: Memory (Source: User's Manual):



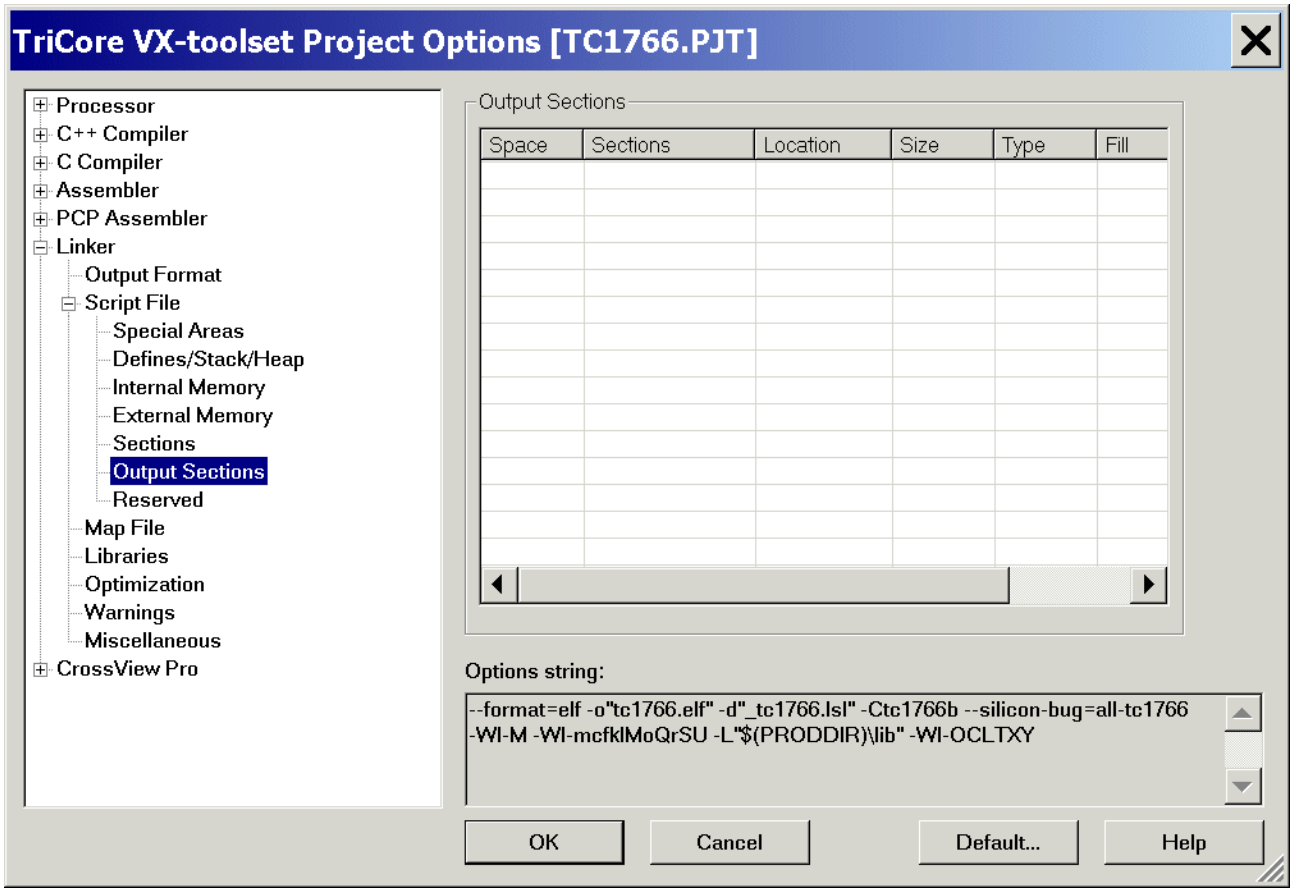
Linker: Script File: External Memory: (do nothing)



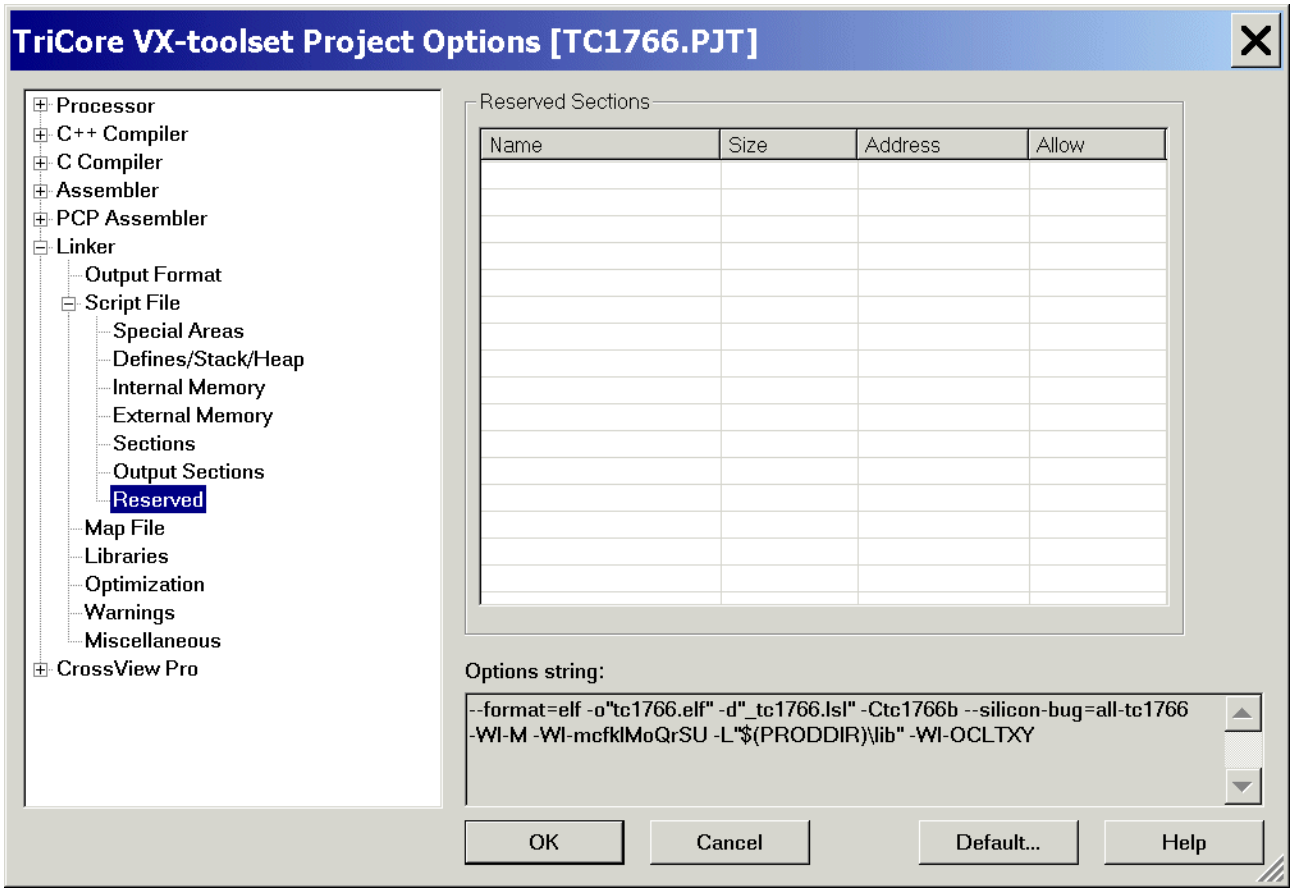
Linker: Script File: Sections: (do nothing)



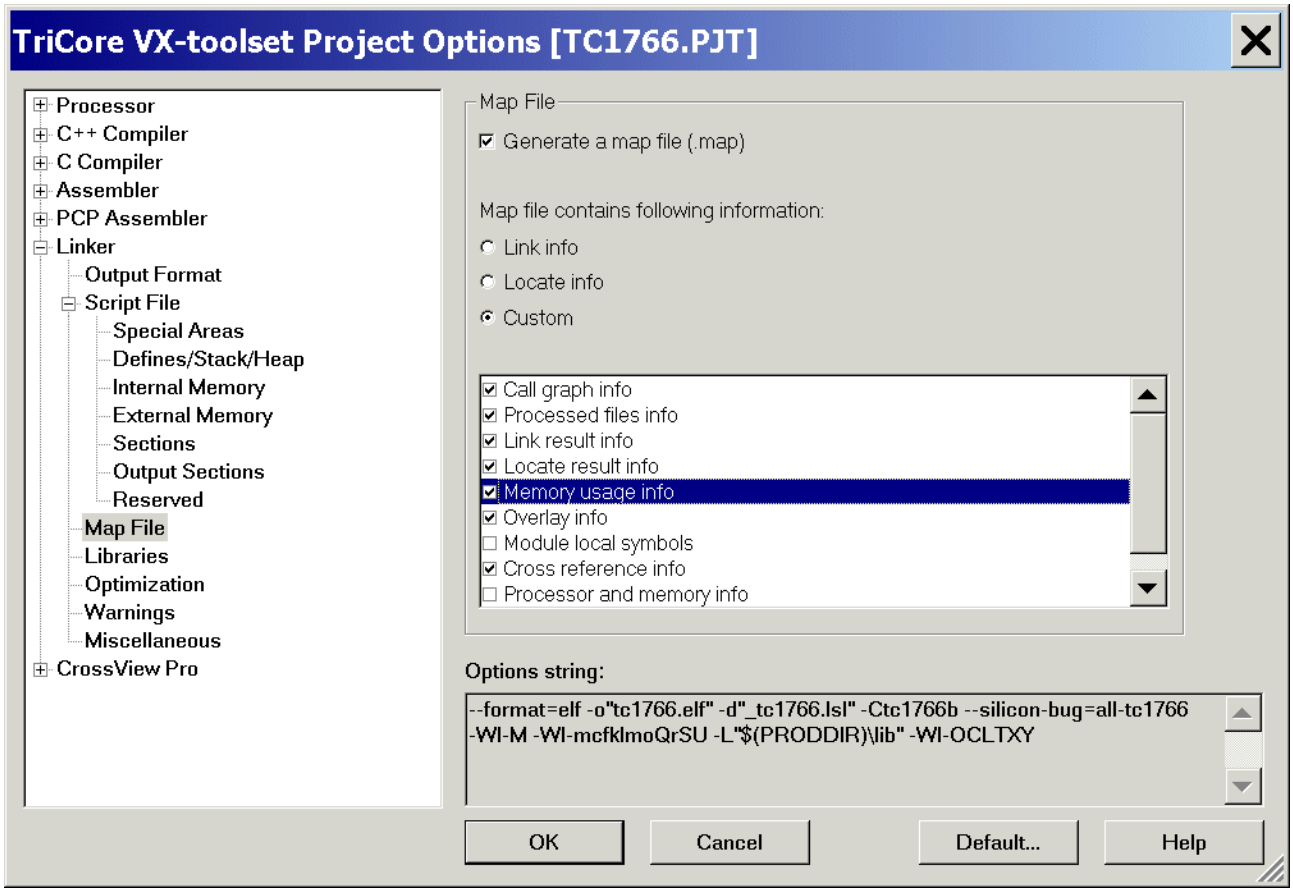
Linker: Script File: Output Sections: (do nothing)



Linker: Script File: Reserved: (do nothing)

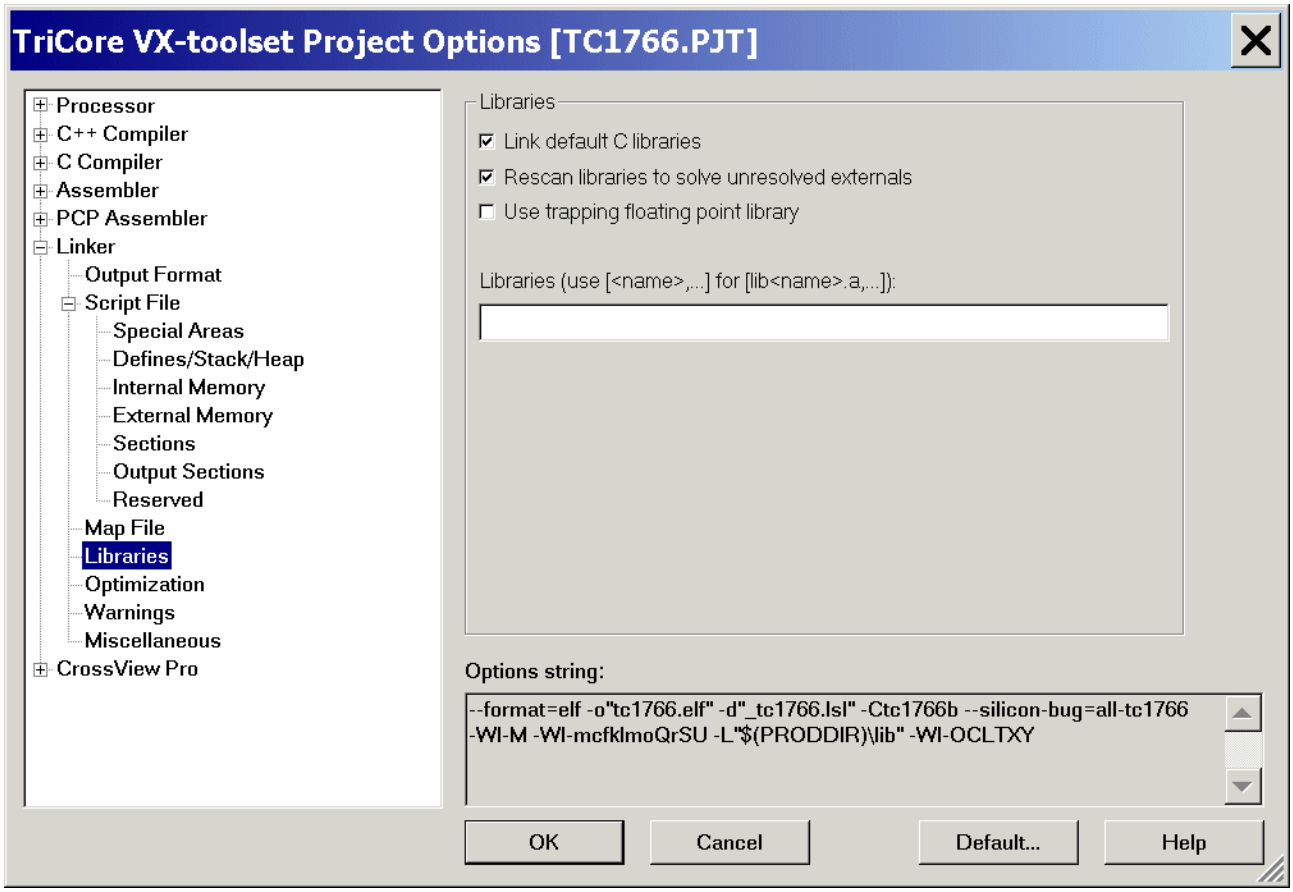


Linker: Map File: tick ✓ Memory usage info

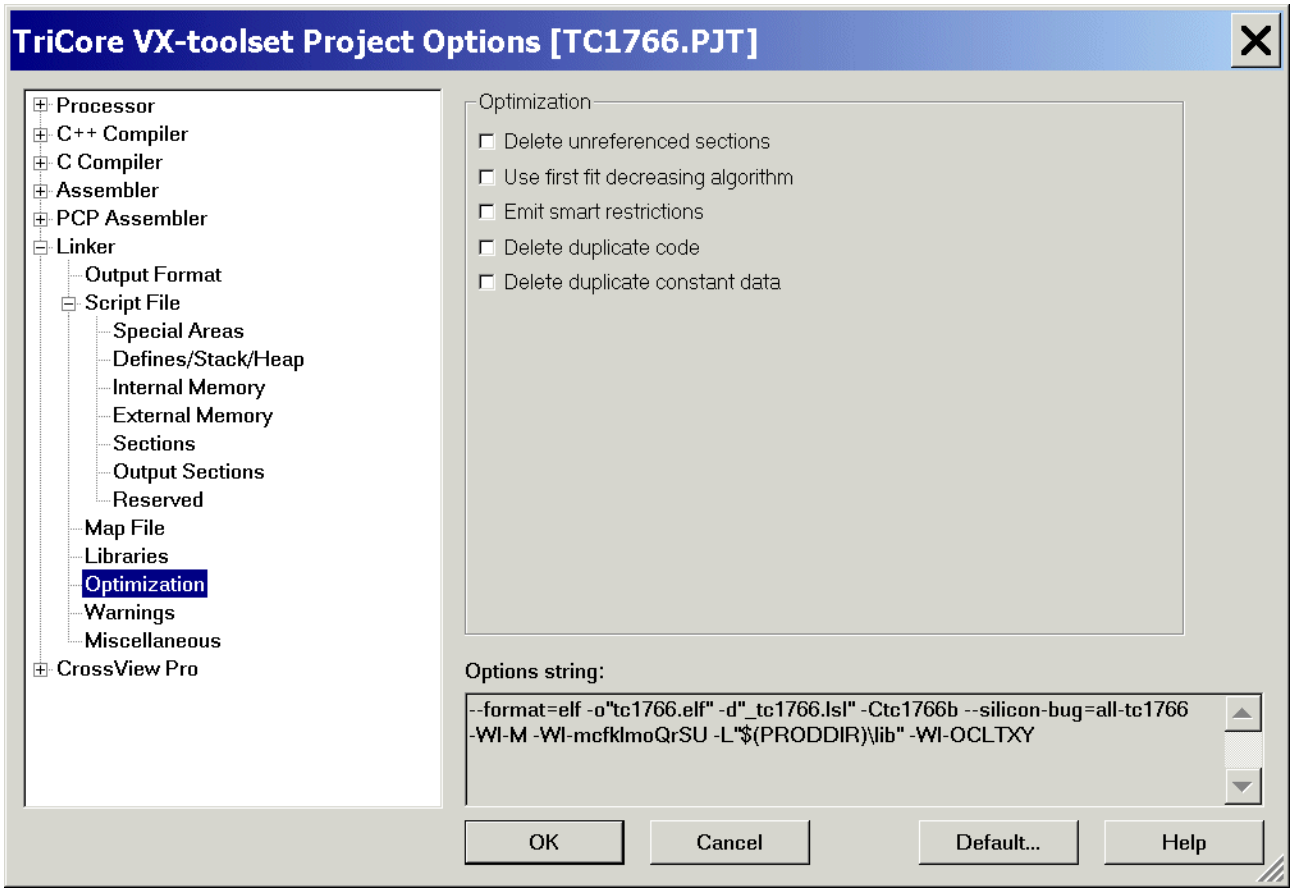


Note:
Click here to see Memory usage info

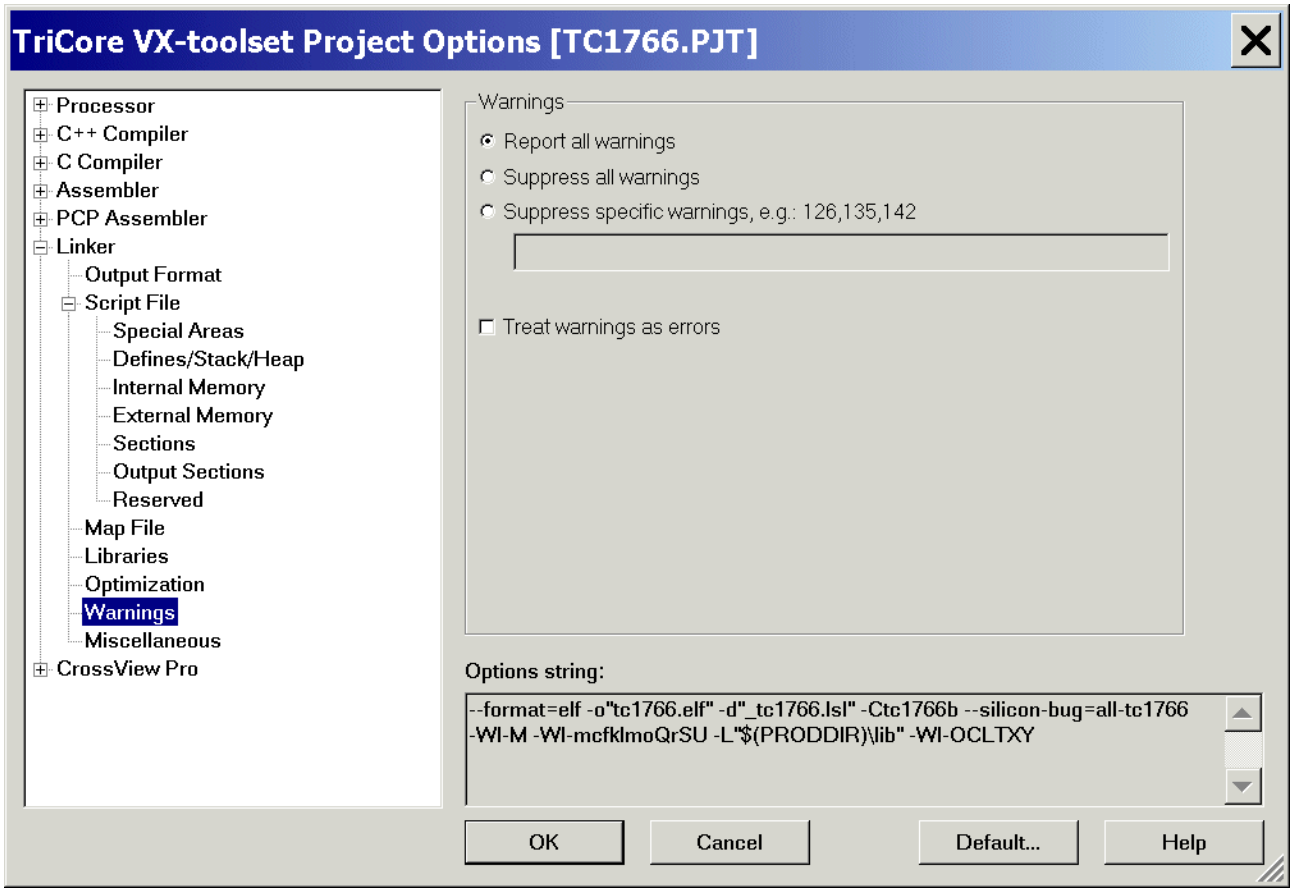
Linker: Script File: Libraries: (do nothing)



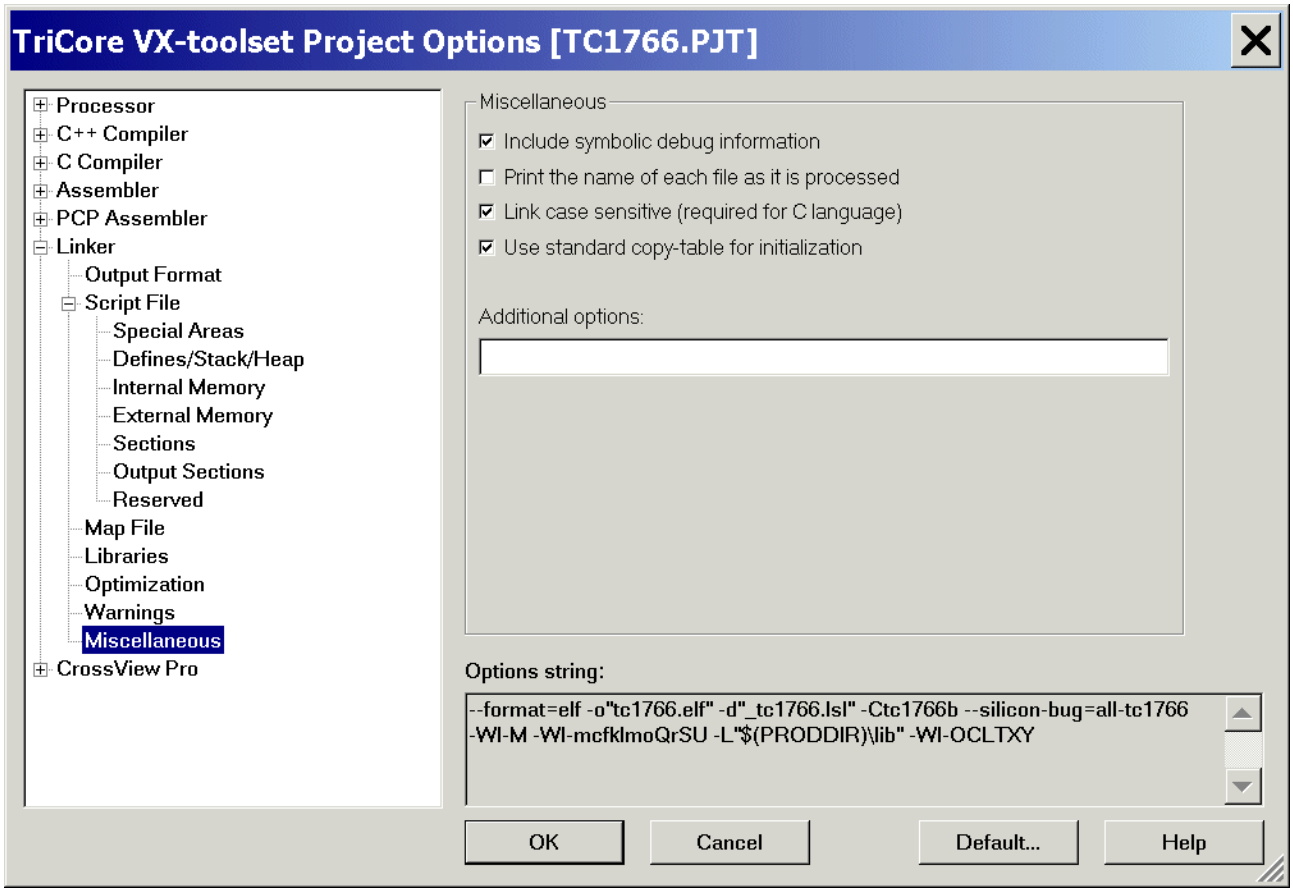
Linker: Script File: Optimization: (do nothing)



Linker: Script File: Warnings: (do nothing)

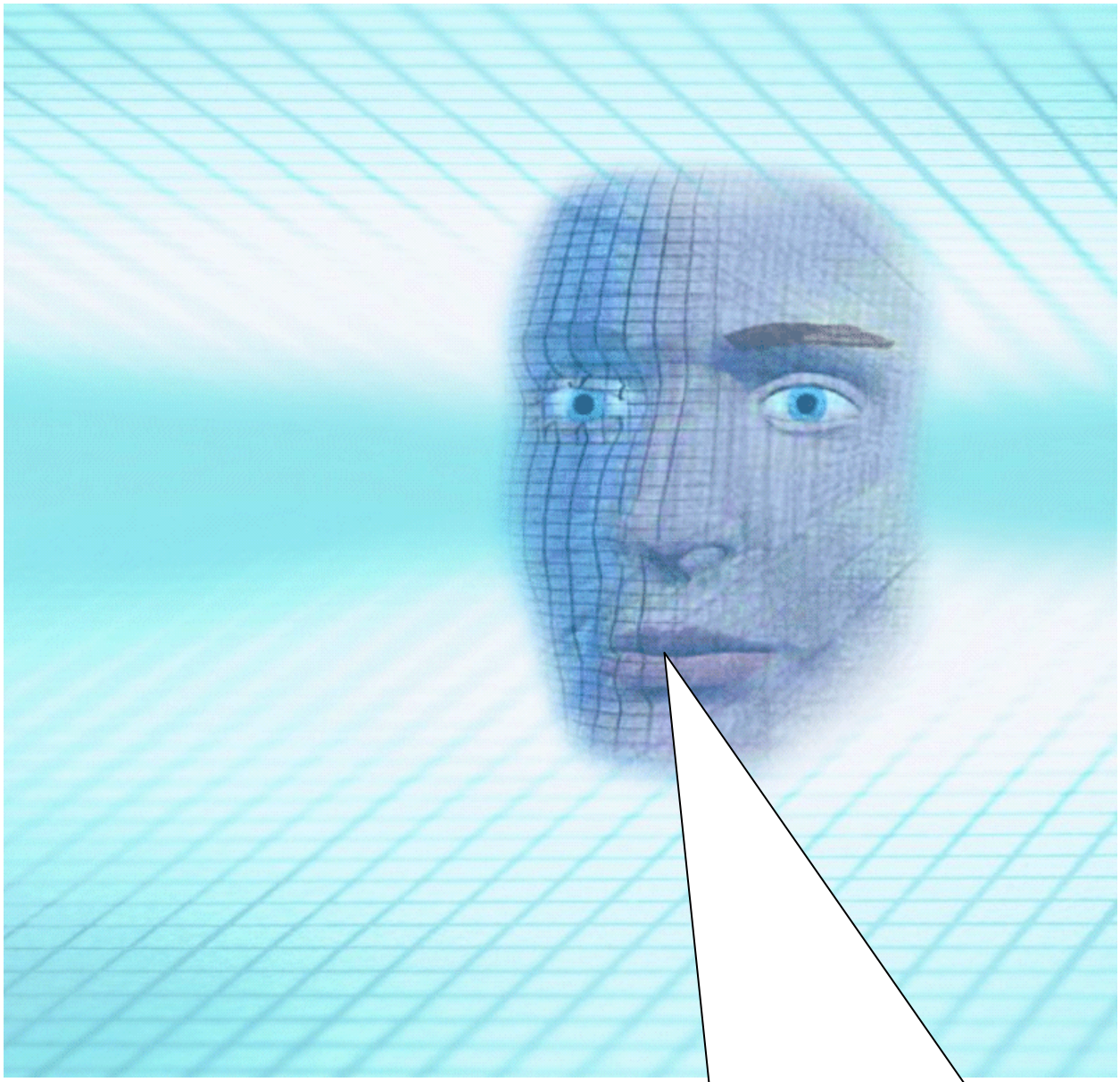


Linker: Script File: Miscellaneous: (do nothing)



OK

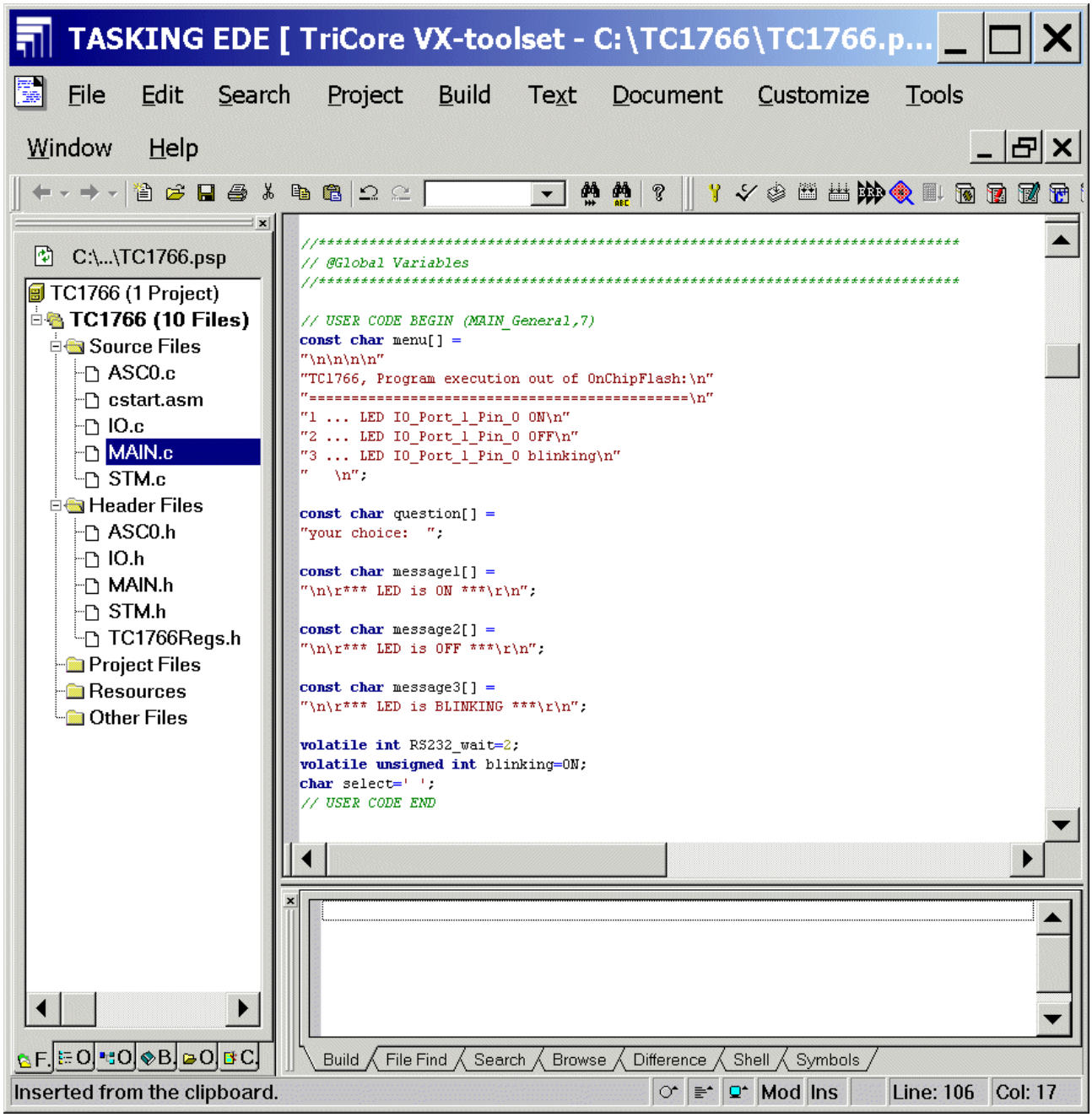
Insert your application specific program:



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!

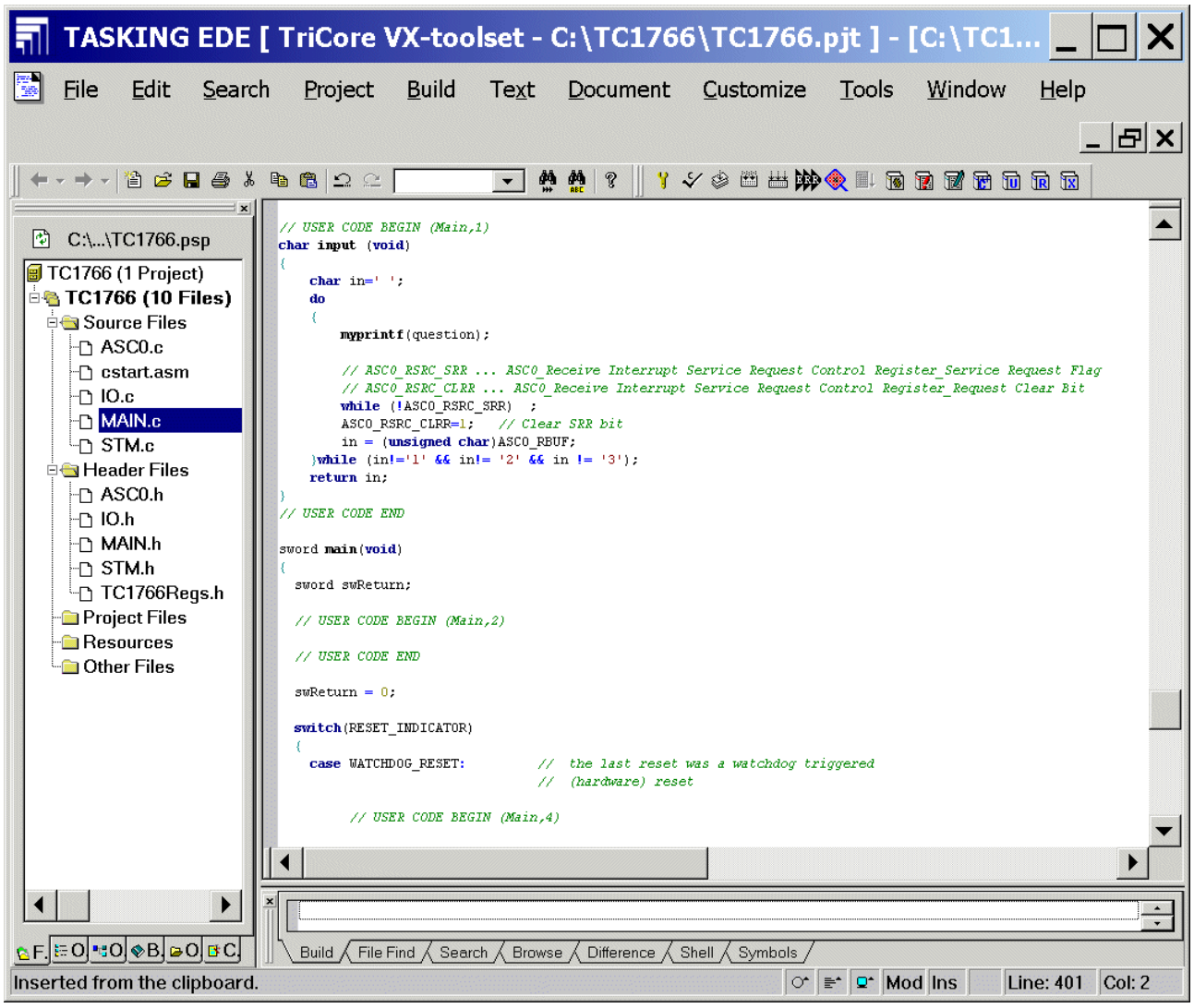
Double click: Main.c insert User Code (Global Variables):

```
const char menu[] =
"\n\n\n\n"
"TC1766, Program execution out of OnChipFlash:\n"
"=============================================\n"
"1 ... LED IO_Port_1_Pin_0 ON\n"
"2 ... LED IO_Port_1_Pin_0 OFF\n"
"3 ... LED IO_Port_1_Pin_0 blinking\n"
"   \n";

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

const char message1[] =
"\n\r*** LED is ON ***\r\n";

const char message2[] =
"\n\r*** LED is OFF ***\r\n";

const char message3[] =
"\n\r*** LED is BLINKING ***\r\n";

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


TASKING EDE [TriCore VX-toolset - C:\TC1766\TC1766.p...
File Edit Search Project Build Text Document Customize Tools
Window Help
C:\...\TC1766.psp
TC1766 (1 Project)
TC1766 (10 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
TC1766Regs.h
Project Files
Resources
Other Files
//***
// @Global Variables
//***
// USER CODE BEGIN (MAIN_General,7)
const char menu[] =
"\n\n\n\n"
"TC1766, Program execution out of OnChipFlash:\n"
"===\n"
"1 ... LED IO_Port_1_Pin_0 ON\n"
"2 ... LED IO_Port_1_Pin_0 OFF\n"
"3 ... LED IO_Port_1_Pin_0 blinking\n"
" \n";
const char question[] =
"your choice: ";
const char message1[] =
"\n\r*** LED is ON ***\r\n";
const char message2[] =
"\n\r*** LED is OFF ***\r\n";
const char message3[] =
"\n\r*** LED is BLINKING ***\r\n";
volatile int RS232_wait=2;
volatile unsigned int blinking=ON;
char select=' ';
// USER CODE END
Build File Find Search Browse Difference Shell Symbols
Inserted from the clipboard.
Mod Ins
Line: 106
Col: 17

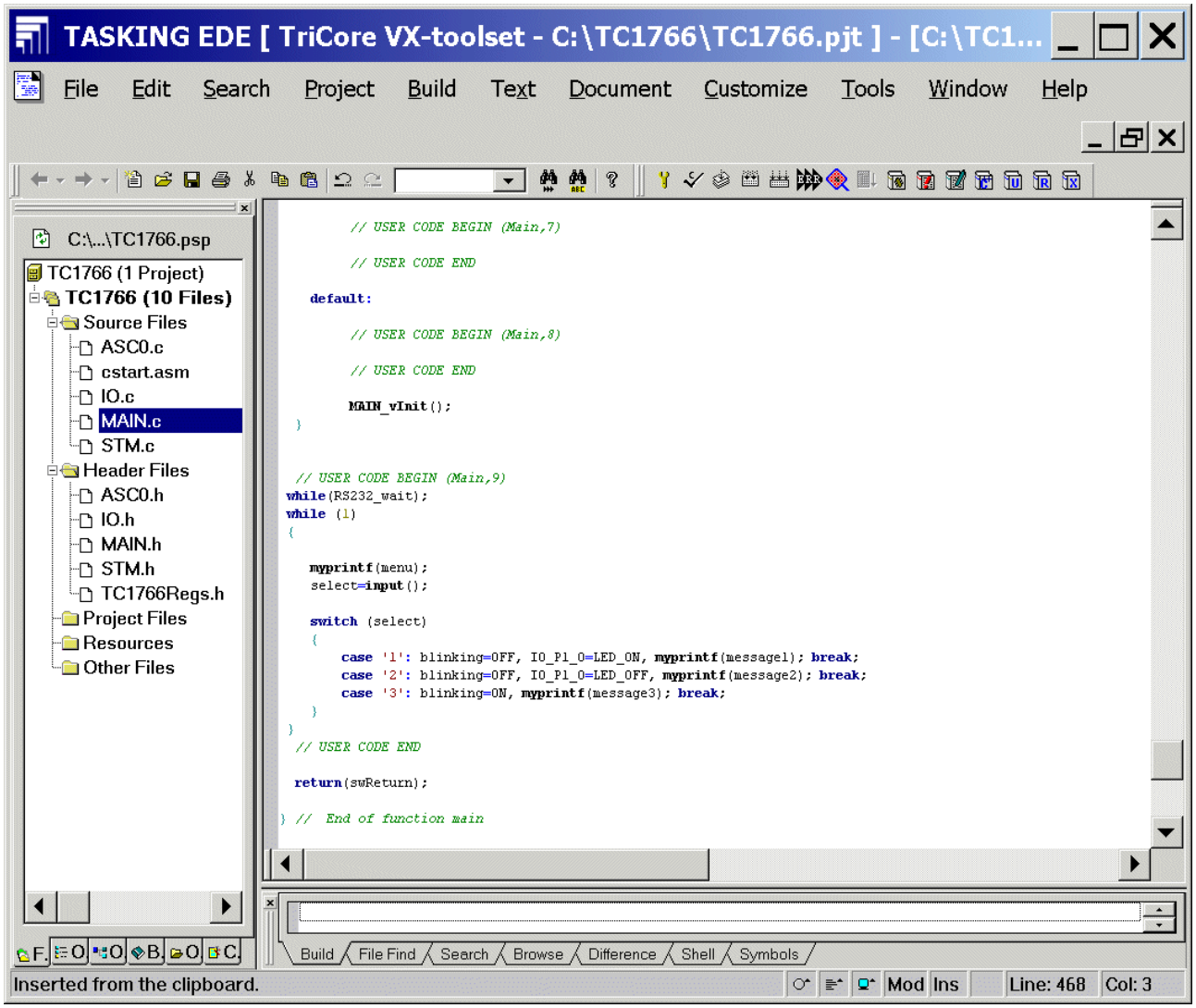
Double click: Main.c insert User Code (function: input()):

```
char input (void)
{
        char in=' ';
        do
        {
                myprintf(question);

                // ASC0_RSRC_SRR ... ASC0_Receive Interrupt Service Request Control Register_Service Request Flag
                // ASC0_RSRC_CLRR ... ASC0_Receive Interrupt Service Request Control Register_Request Clear Bit
                while (!ASC0_RSRC_SRR)  ;
                ASC0_RSRC_CLRR=1;       // Clear SRR bit
                in = (unsigned char)ASC0_RBUF;
        }while (in!='1' && in!= '2' && in != '3');
        return in;
}
```



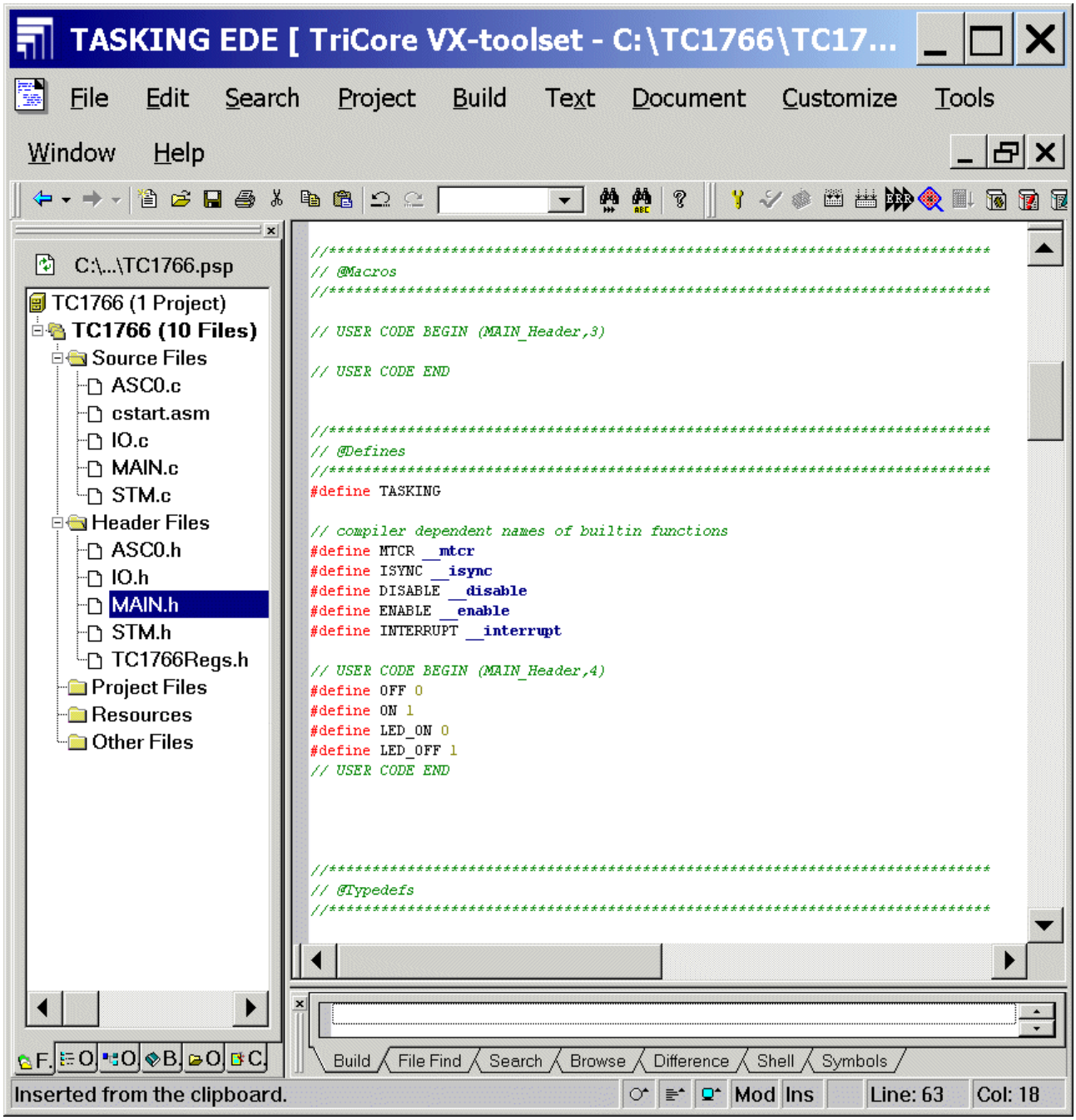
Double click: Main.c insert User Code:

```
while(RS232_wait);
while (1)
{

  myprintf(menu);
  select=input();

  switch (select)
  {
     case '1': blinking=OFF, IO_P1_0=LED_ON, myprintf(message1); break;
     case '2': blinking=OFF, IO_P1_0=LED_OFF, myprintf(message2); break;
     case '3': blinking=ON, myprintf(message3); break;
  }
}
```



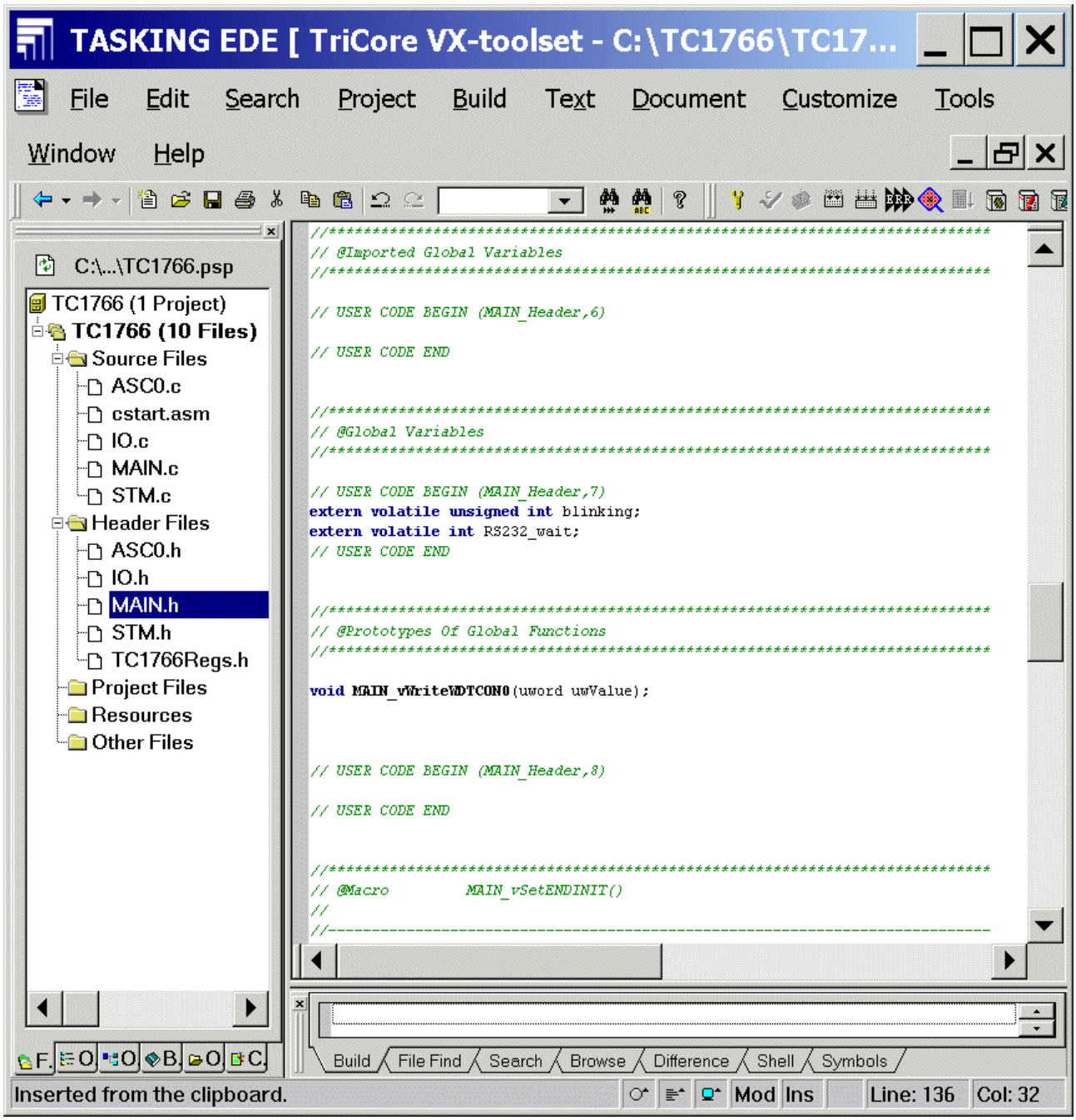
Double click: Main.h and insert the following Defines:

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



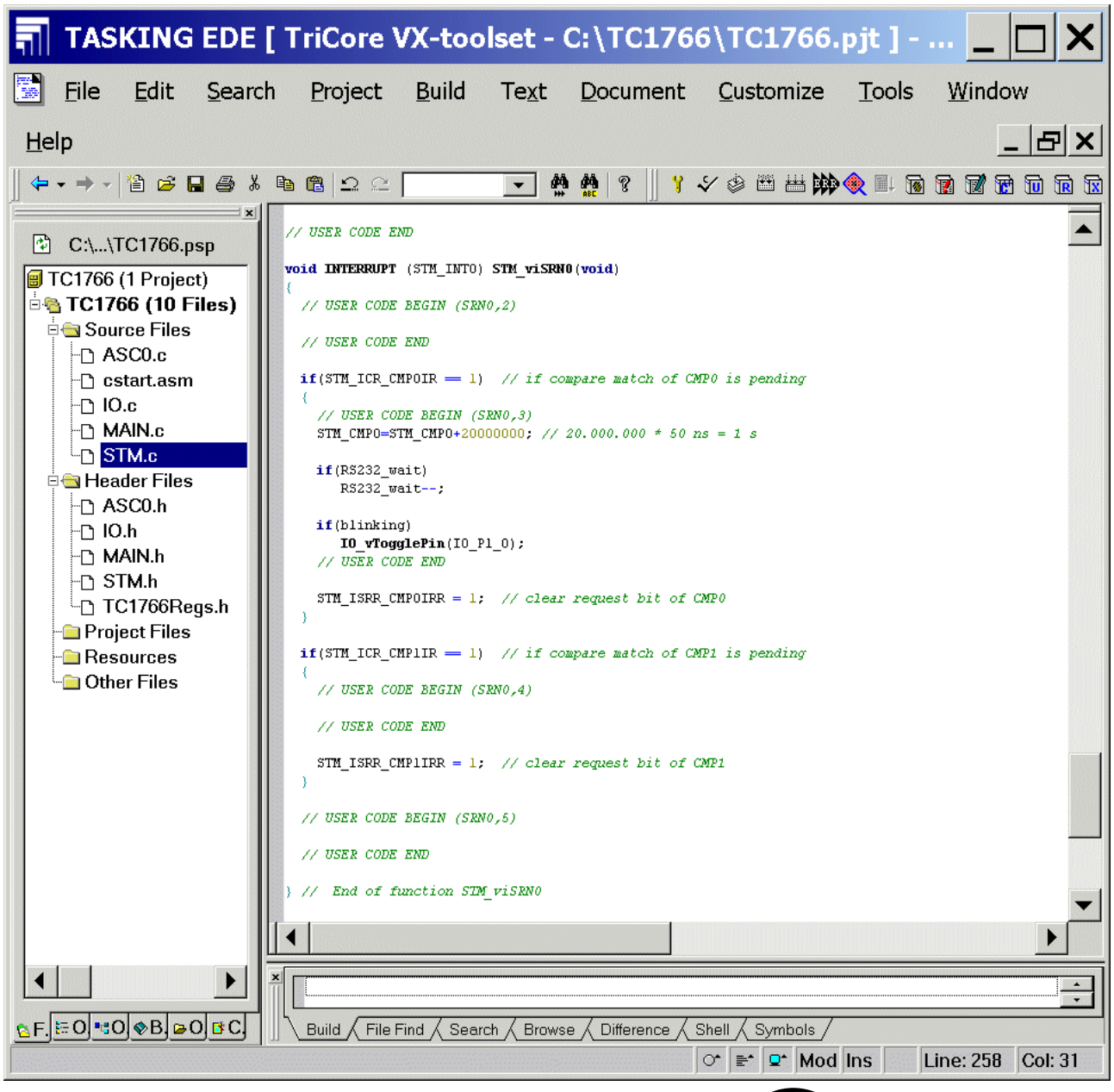
Double click: Main.h and insert Global Variables:

```
extern volatile unsigned int blinking;
extern volatile int RS232_wait;
```



Double click: STM.c insert User Code for interrupt service routine:

```
STM_CMP0=STM_CMP0+20000000; // 20.000.000 * 50 ns = 1 s

if(RS232_wait)
  RS232_wait--;

if(blinking)
  IO_vTogglePin(IO_P1_0);
```



Note:
20.000.000 * 50 ns = 1 s
To get an STM interrupt every 1 second you must change the Compare Value to "STM_CMP0+=20000000;"!

August 2003

Reason for “myprintf.c“

Unfortunately, a low-level I/O implementation similar to example project “IO” (which consists of “serio.c” and “serio.h” files for generating an output stream for “printf” using ASC0) using tool chain C166/ST10 is currently not available for Tasking TriCore tools.
For the moment, Tasking has only got the following “Change Request”:

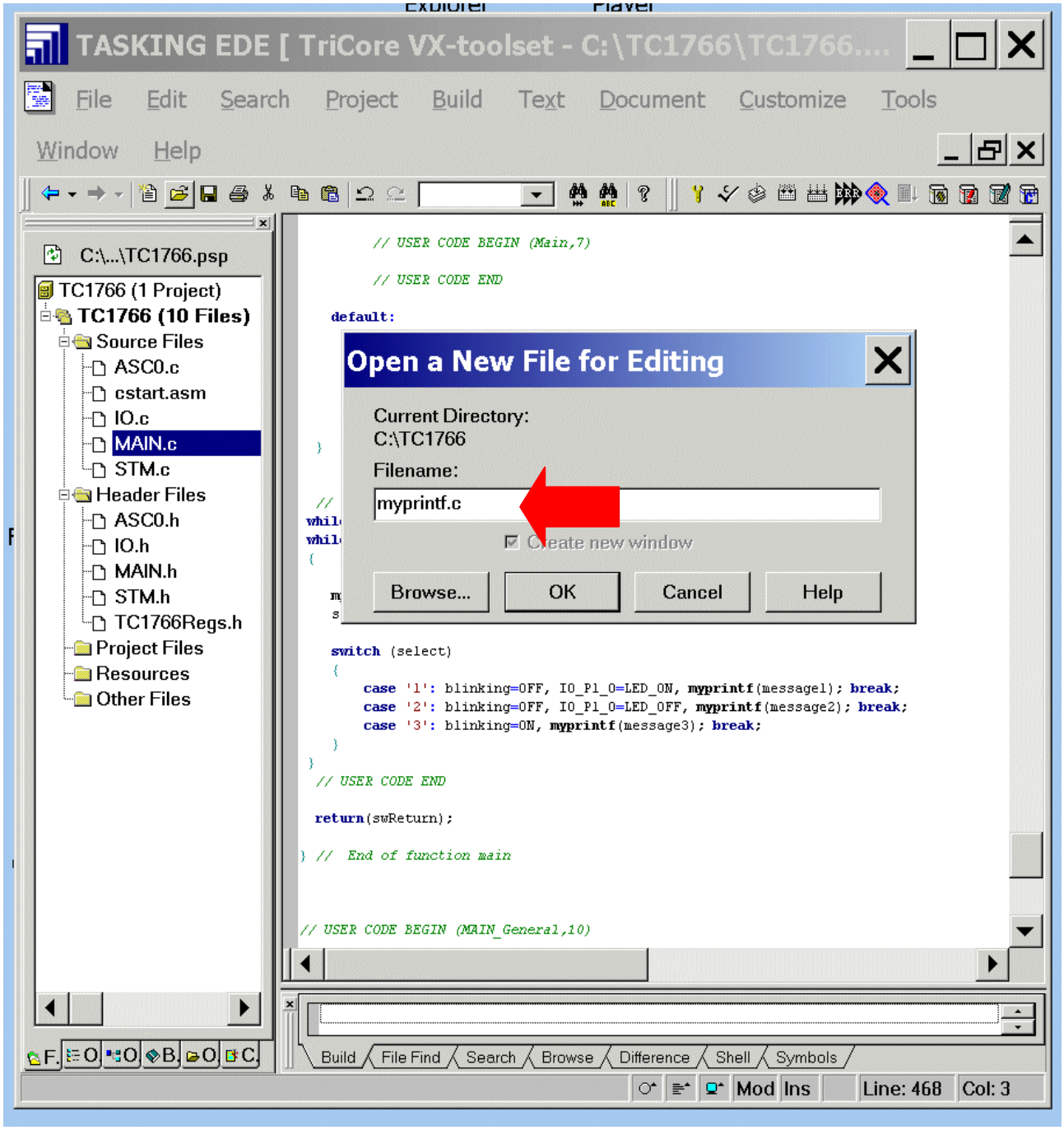
CR32186 CR: Example for _write function implementation using serial interface.

DESCRIPTION
Change request for a low-level I/O (_write function implementation) example which does not use simulated I/O but uses the real serial interface of the controller.

EXAMPLE

WORKAROUND

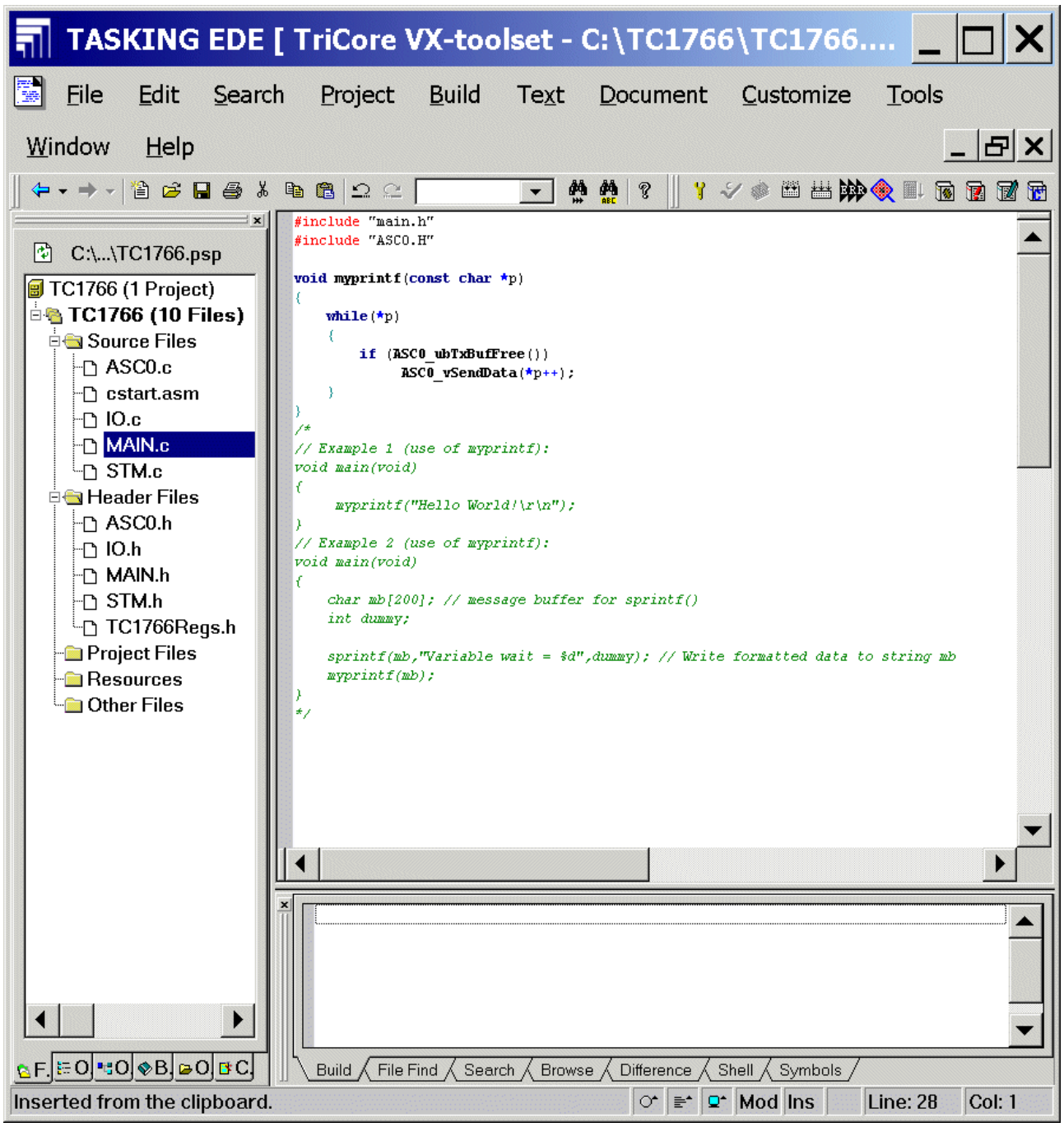
File – New
Open a New File for Editing: Filename: insert myprintf.c



OK

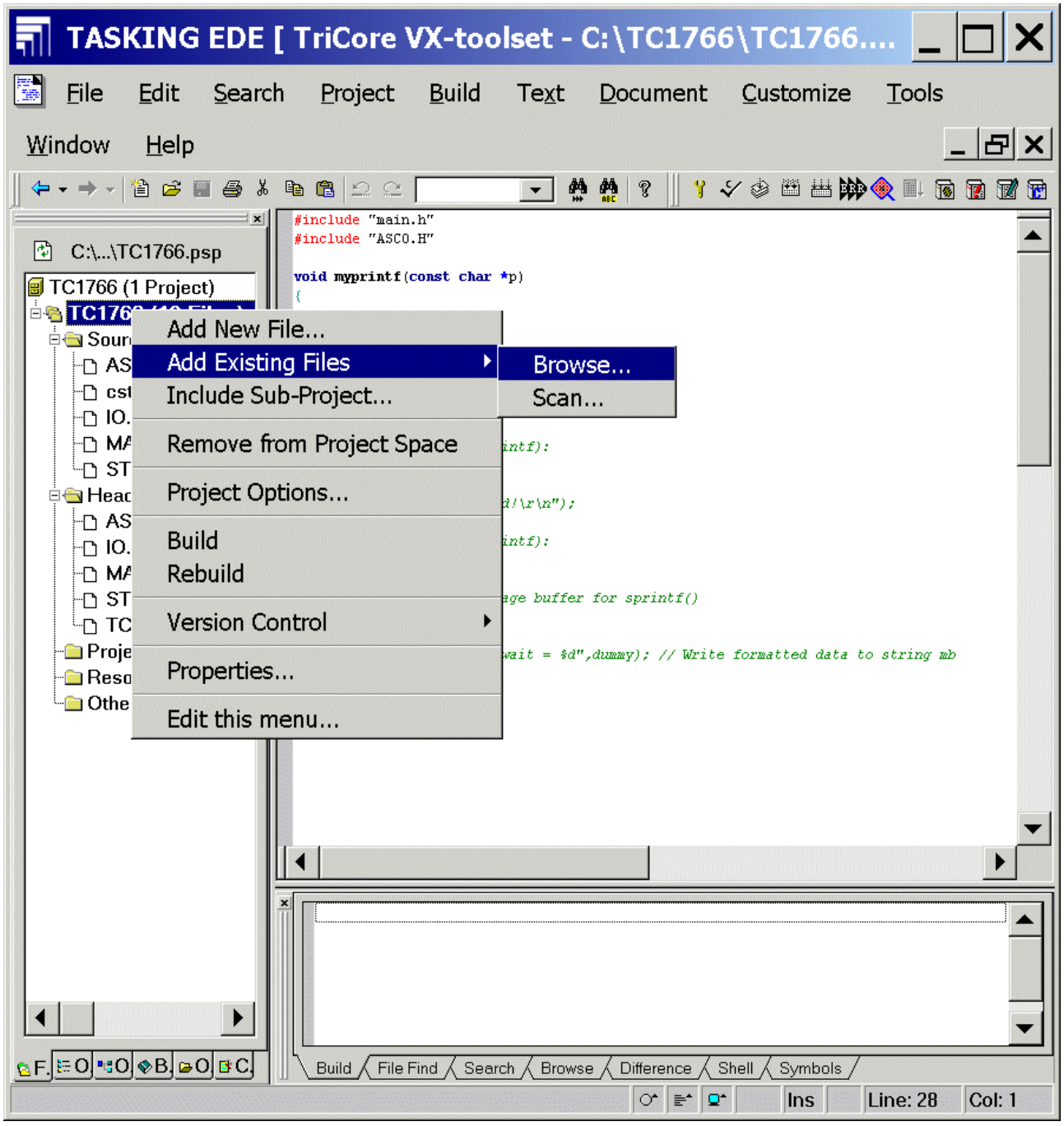
Insert User Code for myprintf():

```
#include "main.h"
#include "ASC0.H"

void myprintf(const char *p)
{
   while(*p)
   {
      if (ASC0_ubTxBufFree())
          ASC0_vSendData(*p++);
   }
}
/*
// Example 1 (use of myprintf):
void main(void)
{
    myprintf("Hello World!\r\n");
}
// Example 2 (use of myprintf):
void main(void)
{
   char mb[200]; // message buffer for sprintf()
   int dummy;

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

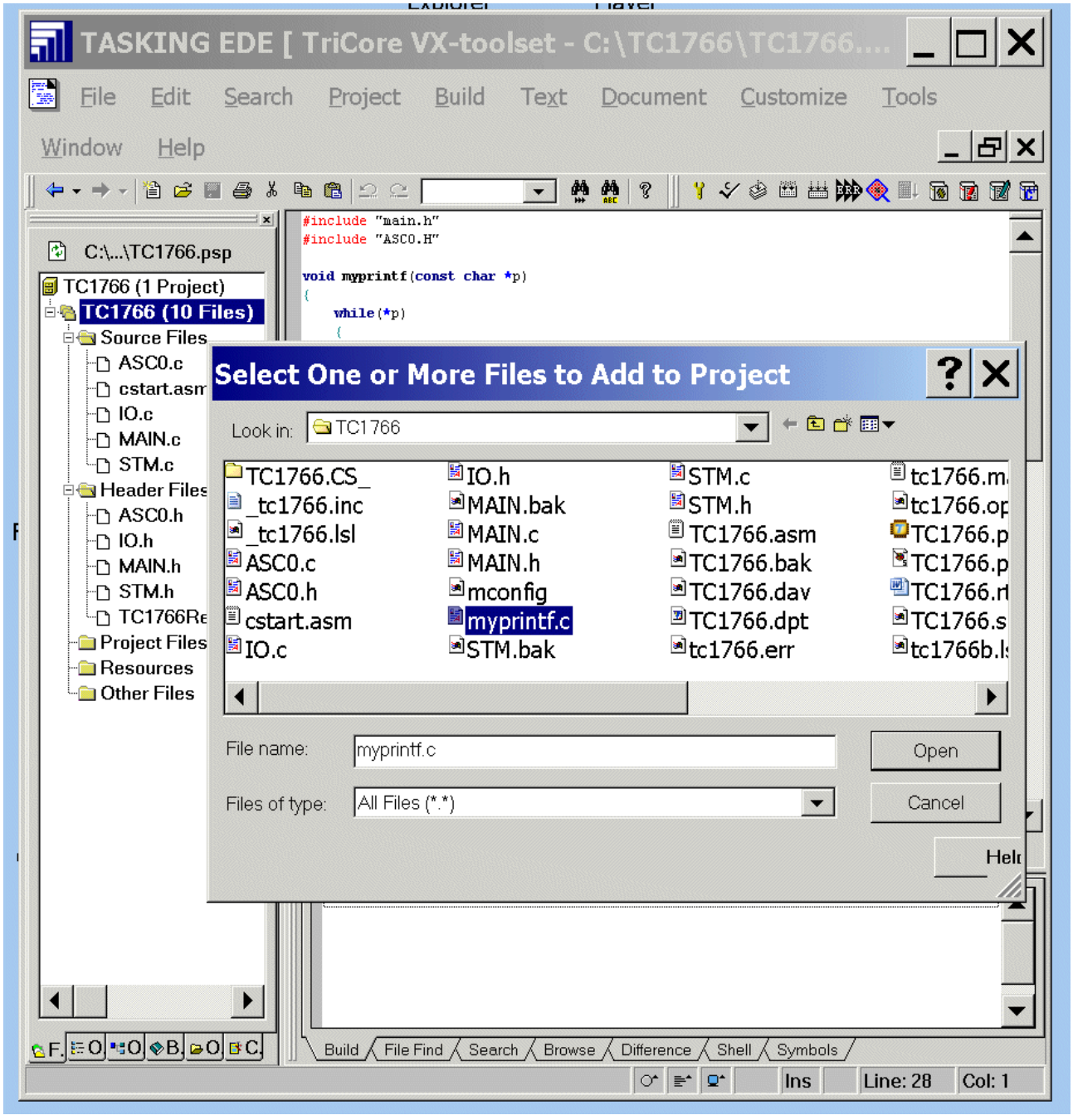

TASKING EDE [TriCore VX-toolset - C:\TC1766\TC1766....
File Edit Search Project Build Text Document Customize Tools
Window Help
C:\...\TC1766.psp
TC1766 (1 Project)
TC1766 (10 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
TC1766Regs.h
Project Files
Resources
Other Files
#include "main.h"
#include "ASC0.H"
void myprintf(const char *p)
{
while(*p)
{
if (ASC0_ubTxBufFree())
ASC0_vSendData(*p++);
}
}
/*
// Example 1 (use of myprintf):
void main(void)
{
myprintf("Hello World!\r\n");
}
// Example 2 (use of myprintf):
void main(void)
{
char mb[200]; // message buffer for sprintf()
int dummy;
sprintf(mb,"Variable wait = %d",dummy); // Write formatted data to string mb
myprintf(mb);
}
*/
Build File Find Search Browse Difference Shell Symbols
Inserted from the clipboard.
Mod Ins
Line: 28
Col: 1

File - Save all

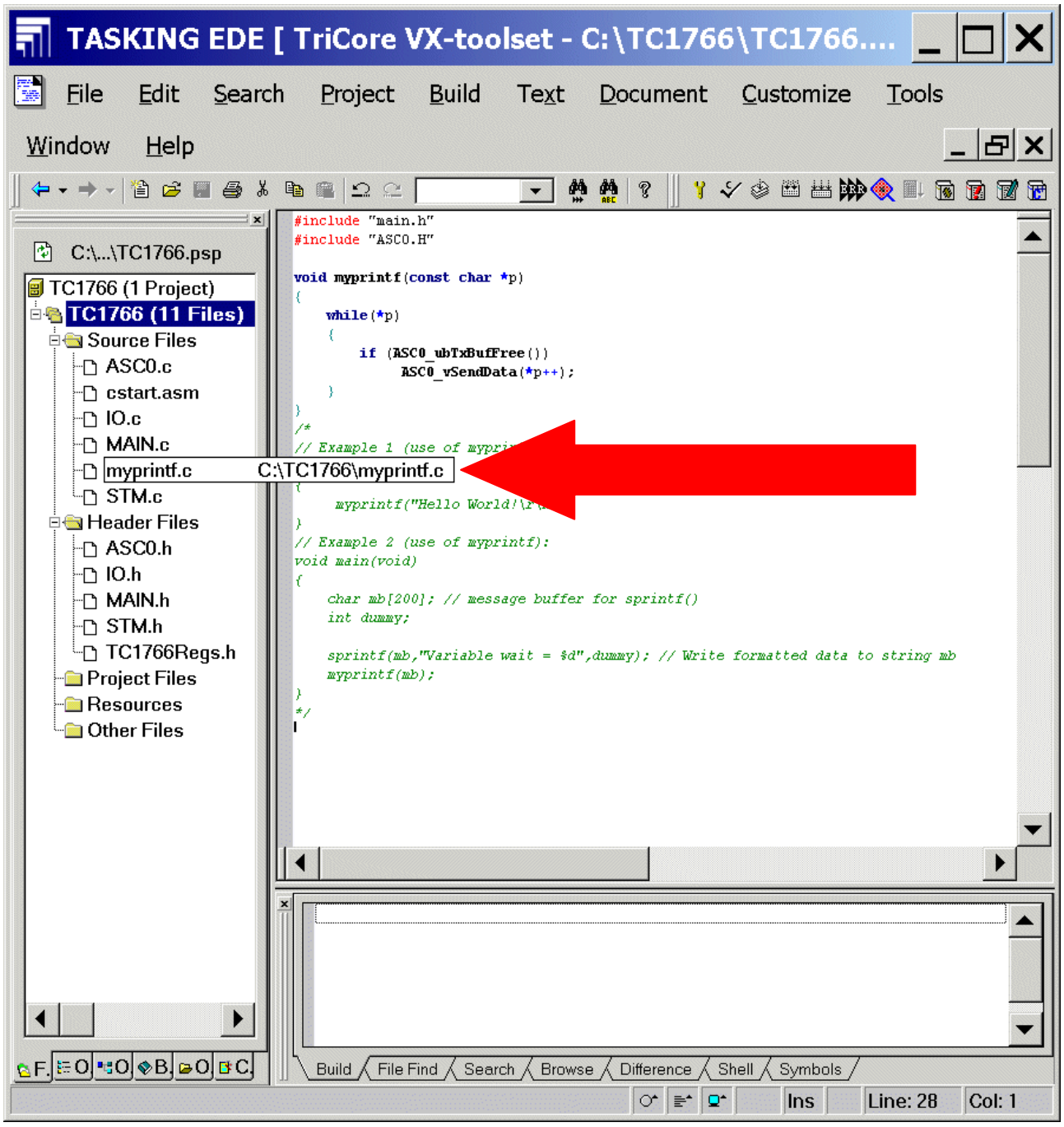
(Project Window File View) – TC1766 (Files) – right mouse button click – Add Existing Files – Browse



Select myprintf.c

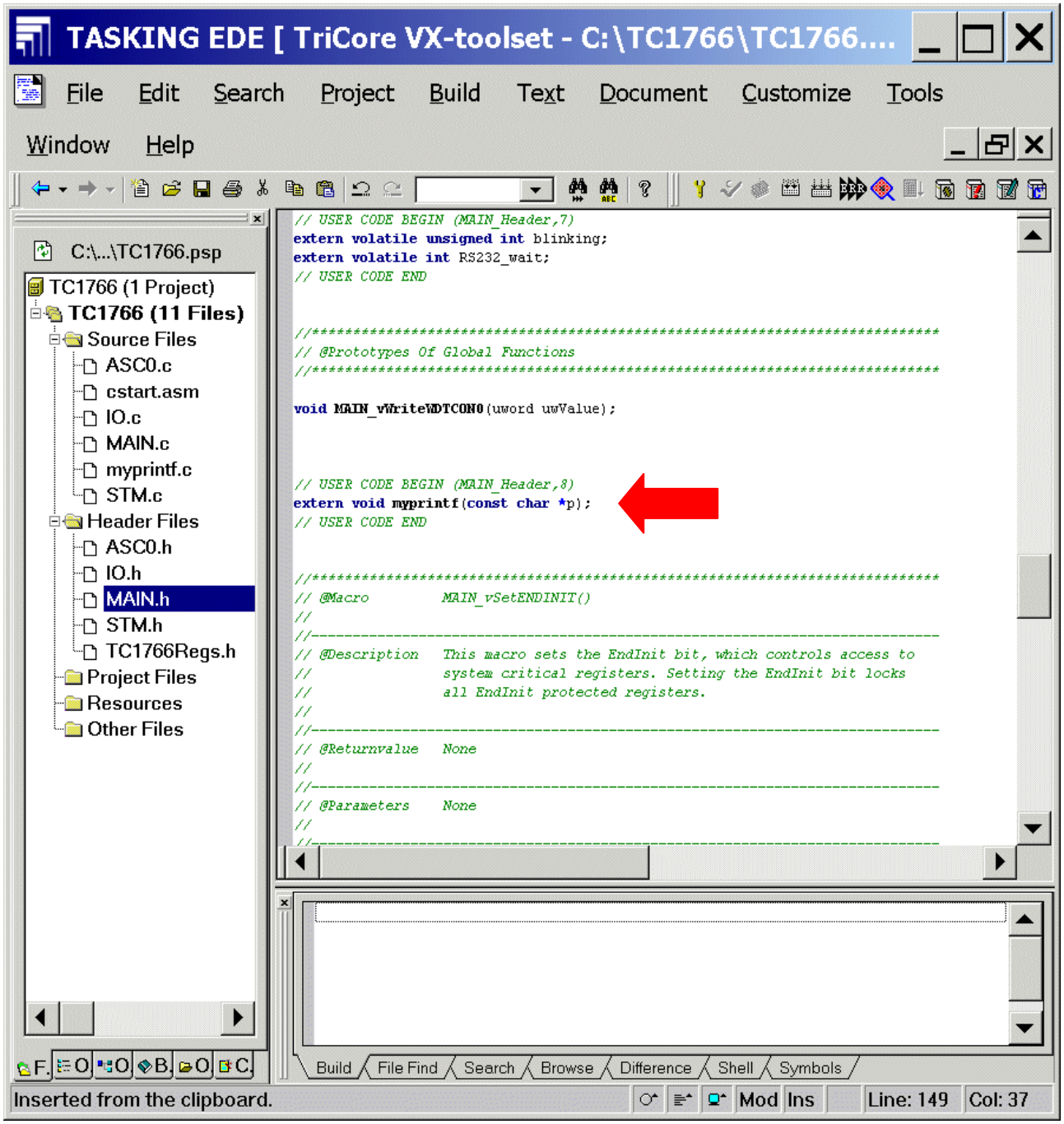


Open - OK


TASKING EDE [TriCore VX-toolset - C:\TC1766\TC1766....
File Edit Search Project Build Text Document Customize Tools
Window Help
C:\...\TC1766.psp
TC1766 (1 Project)
TC1766 (11 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
C:\TC1766\myprintf.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
TC1766Regs.h
Project Files
Resources
Other Files
#include "main.h"
#include "ASC0.H"
void myprintf(const char *p)
{
while(*p)
{
if (ASC0_ubTxBufFree())
ASC0_vSendData(*p++);
}
}
/*
// Example 1 (use of myprintf):
{
myprintf("Hello World!\r\n");
}
// Example 2 (use of myprintf):
void main(void)
{
char mb[200]; // message buffer for sprintf()
int dummy;
sprintf(mb,"Variable wait = %d",dummy); // Write formatted data to string mb
myprintf(mb);
}
*/
Build File Find Search Browse Difference Shell Symbols
Ins
Line: 28
Col: 1

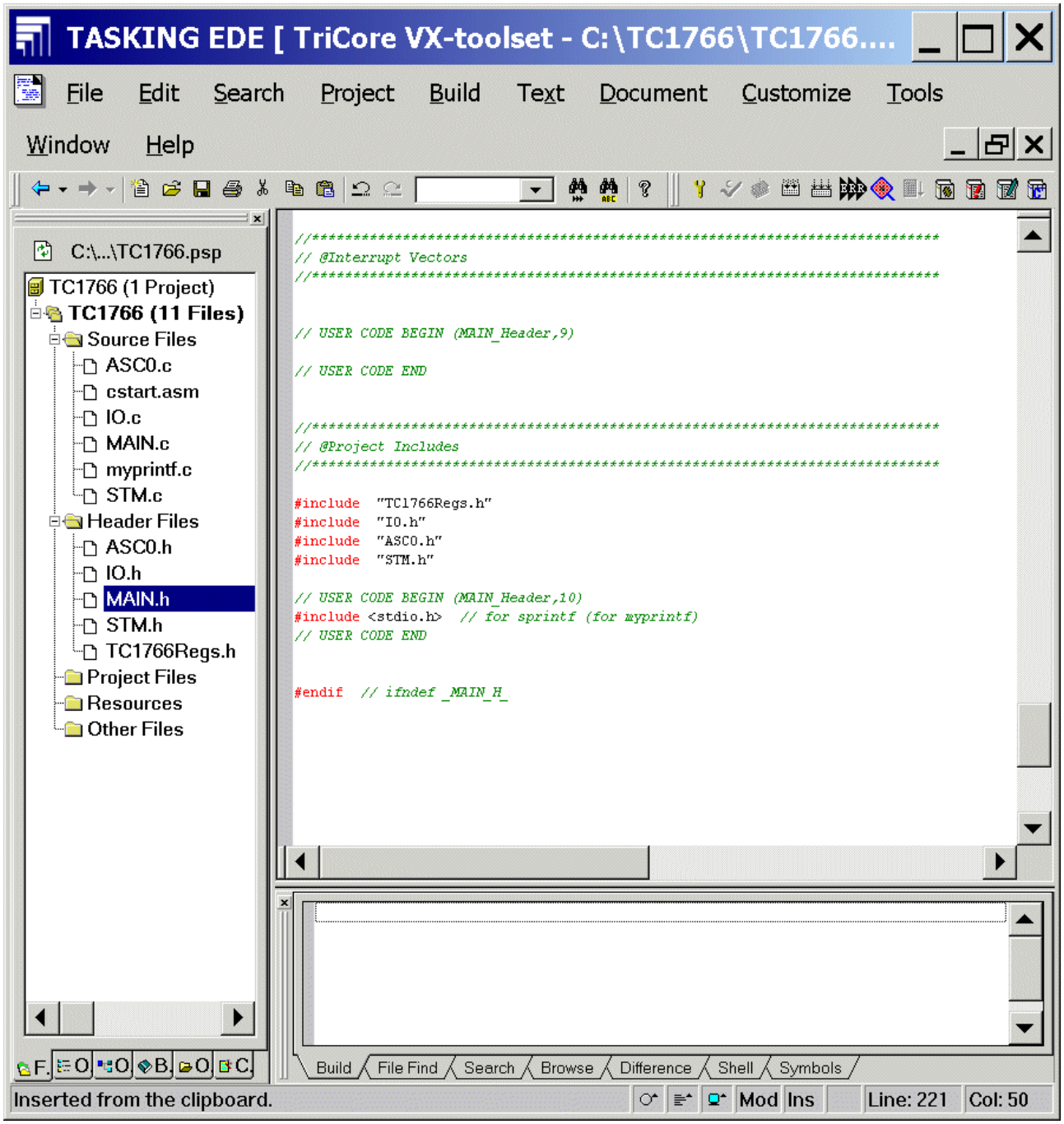
Double click: Main.h and insert Prototypes of Global Functions:

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



Double click: Main.h and insert required Header for sprintf:

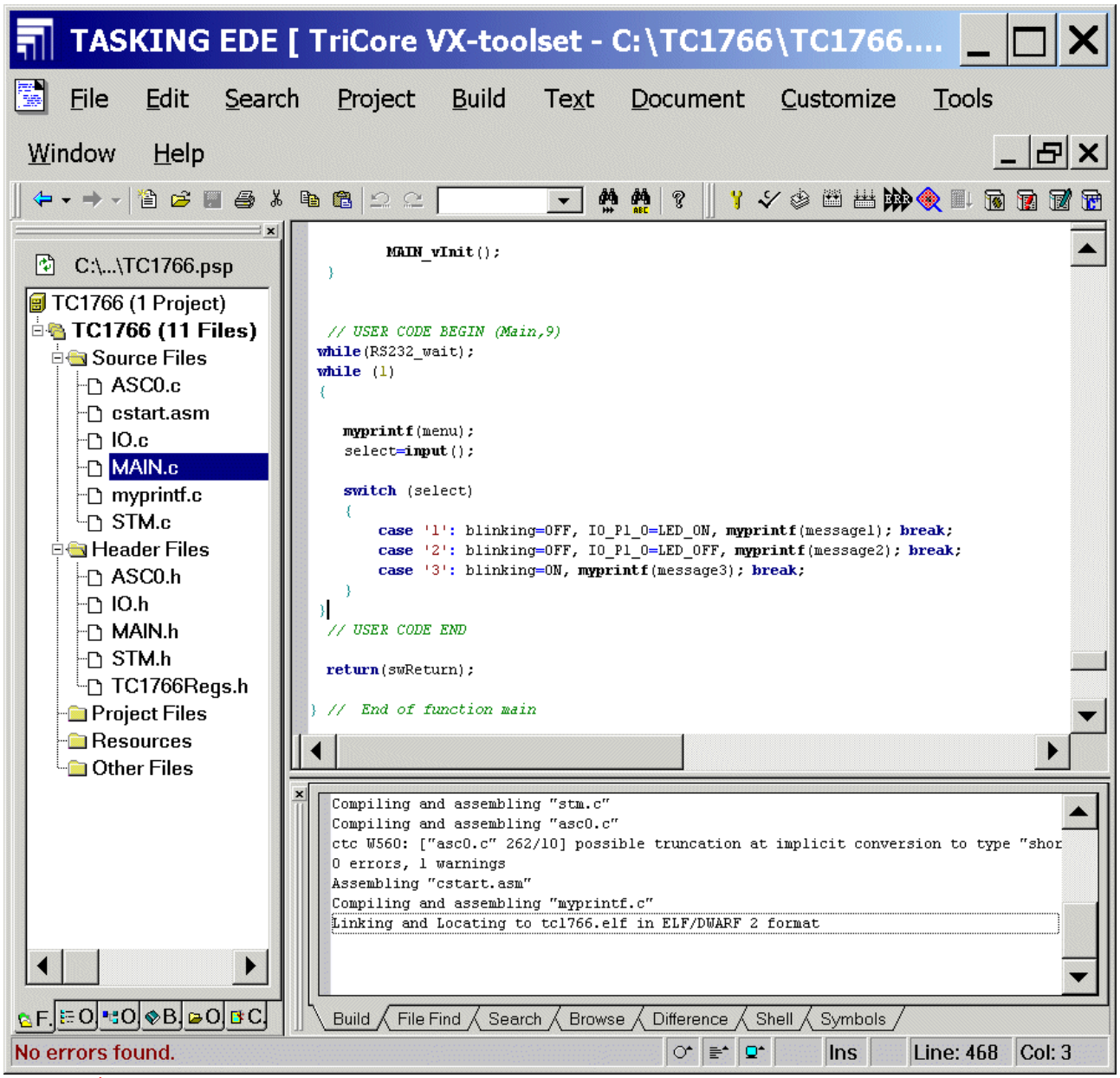
```
#include <stdio.h>     // for sprintf (for myprintf)
```



Generate your application program:

Build - Rebuild

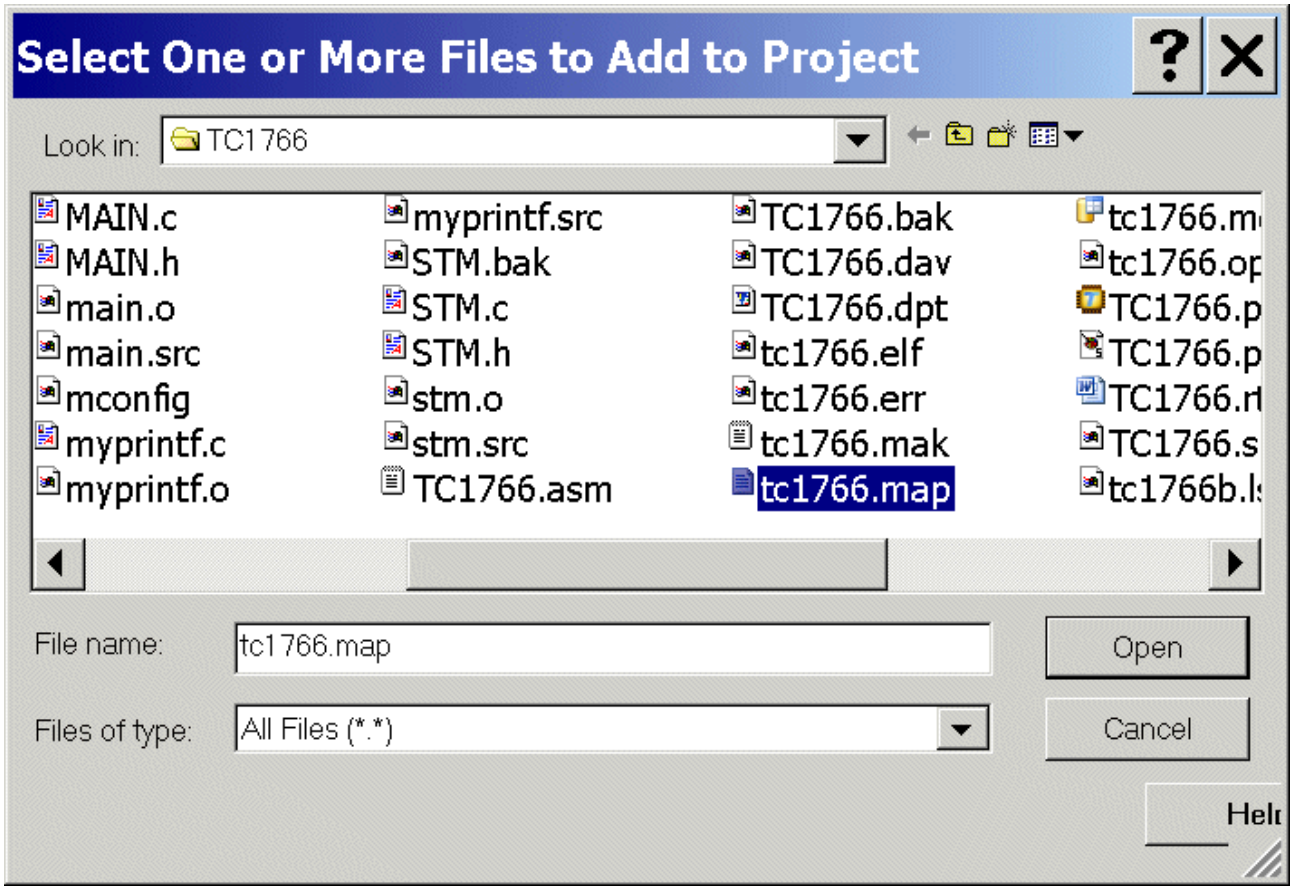
or



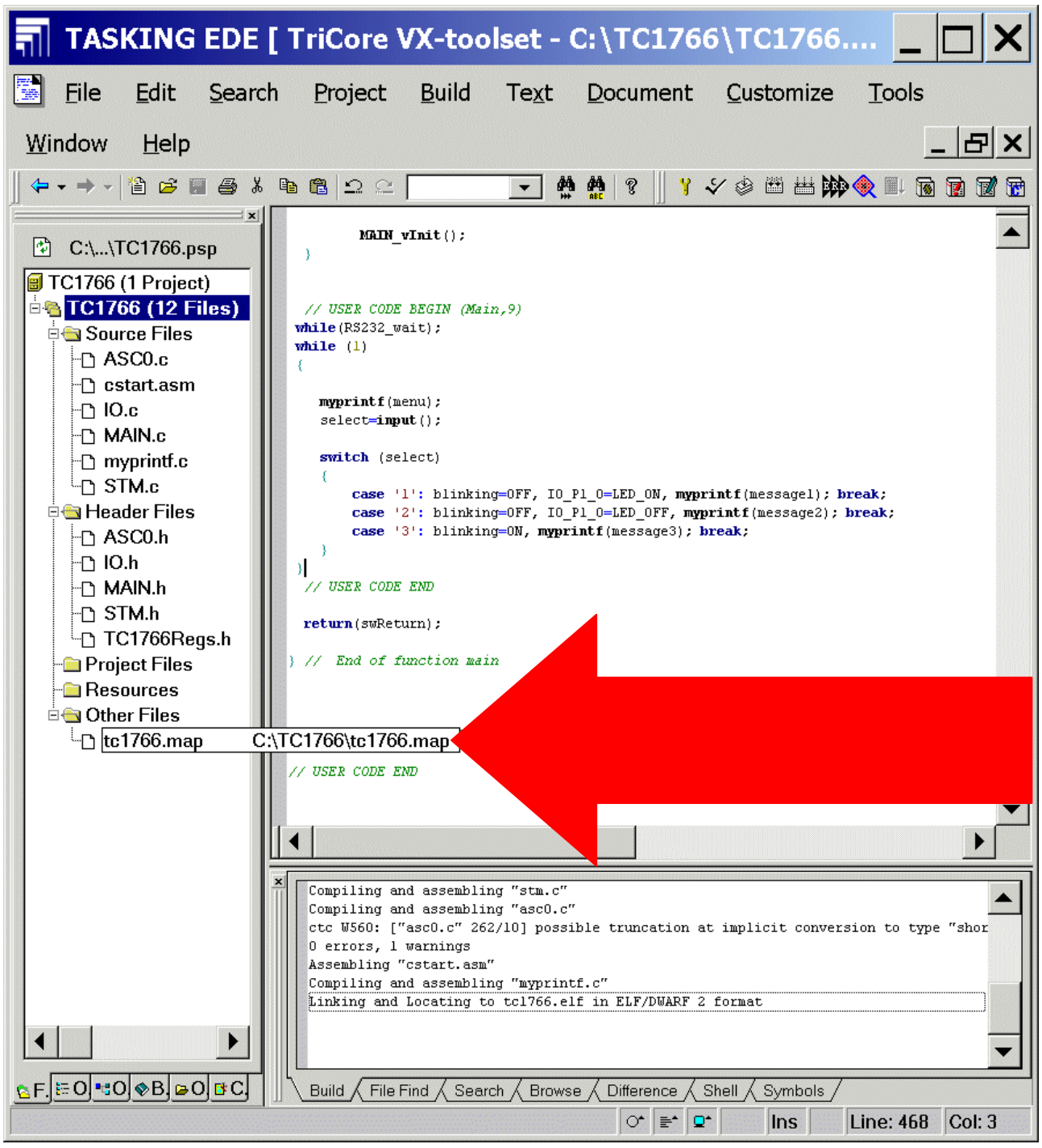
Insert Map File:

(Project Window File View) – TC1766 (Files) – right mouse button click – Add Existing Files – Browse

Select TC1766.map

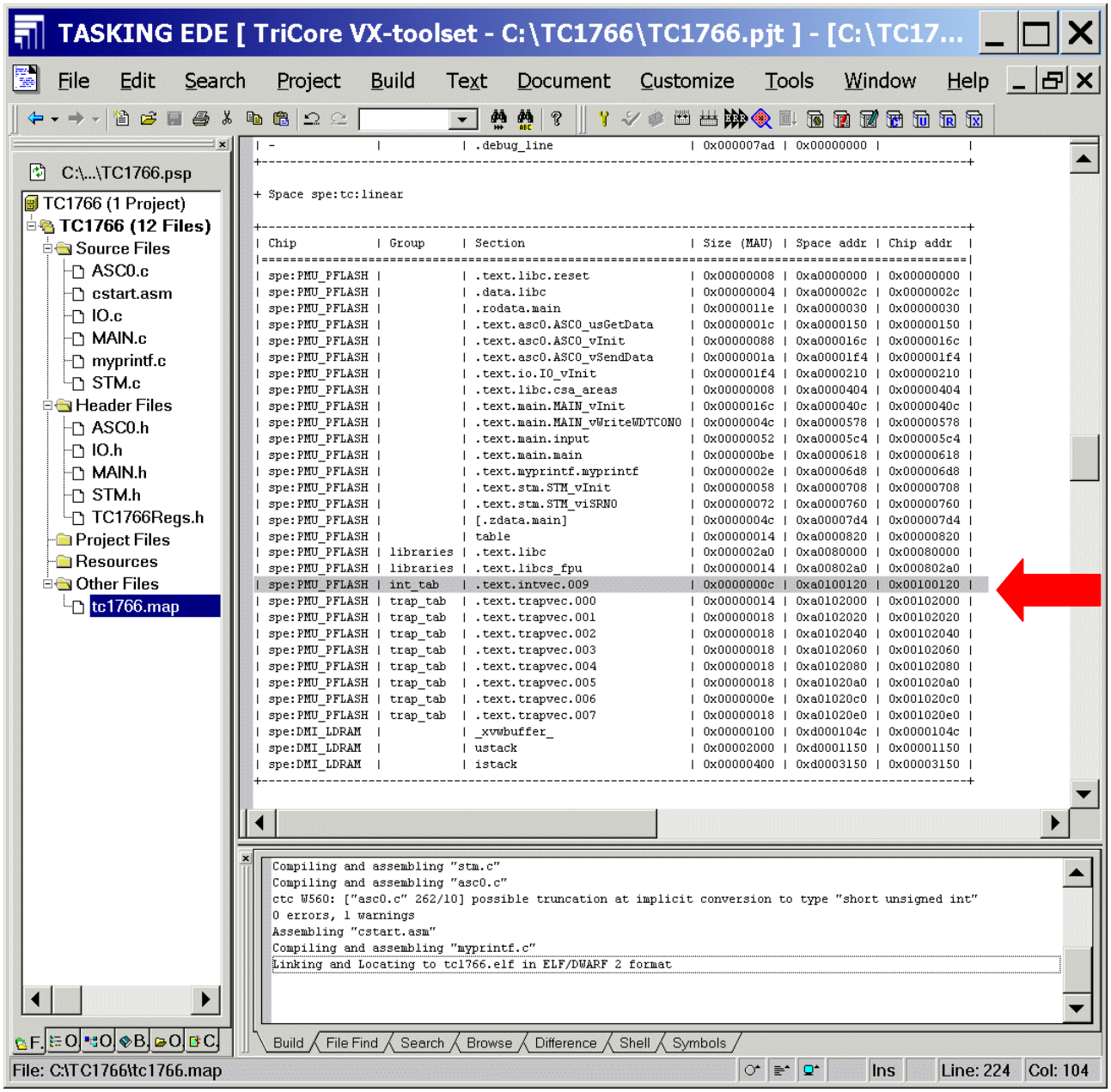


Open - OK


TASKING EDE [TriCore VX-toolset - C:\TC1766\TC1766....
File Edit Search Project Build Text Document Customize Tools
Window Help
C:\...\TC1766.psp
TC1766 (1 Project)
TC1766 (12 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
TC1766Regs.h
Project Files
Resources
Other Files
tc1766.map C:\TC1766\tc1766.map
MAIN_vInit();
// USER CODE BEGIN (Main,9)
while(RS232_wait);
while (1)
{
myprintf(menu);
select=input();
switch (select)
{
case '1': blinking=OFF, IO_P1_0=LED_ON, myprintf(message1); break;
case '2': blinking=OFF, IO_P1_0=LED_OFF, myprintf(message2); break;
case '3': blinking=ON, myprintf(message3); break;
}
}
// USER CODE END
return(swReturn);
} // End of function main
// USER CODE END
Compiling and assembling "stm.c"
Compiling and assembling "asc0.c"
ctc W560: ["asc0.c" 262/10] possible truncation at implicit conversion to type "shor
0 errors, 1 warnings
Assembling "cstart.asm"
Compiling and assembling "myprintf.c"
Linking and Locating to tc1766.elf in ELF/DWARF 2 format
Build File Find Search Browse Difference Shell Symbols
Ins Line: 468 Col: 3

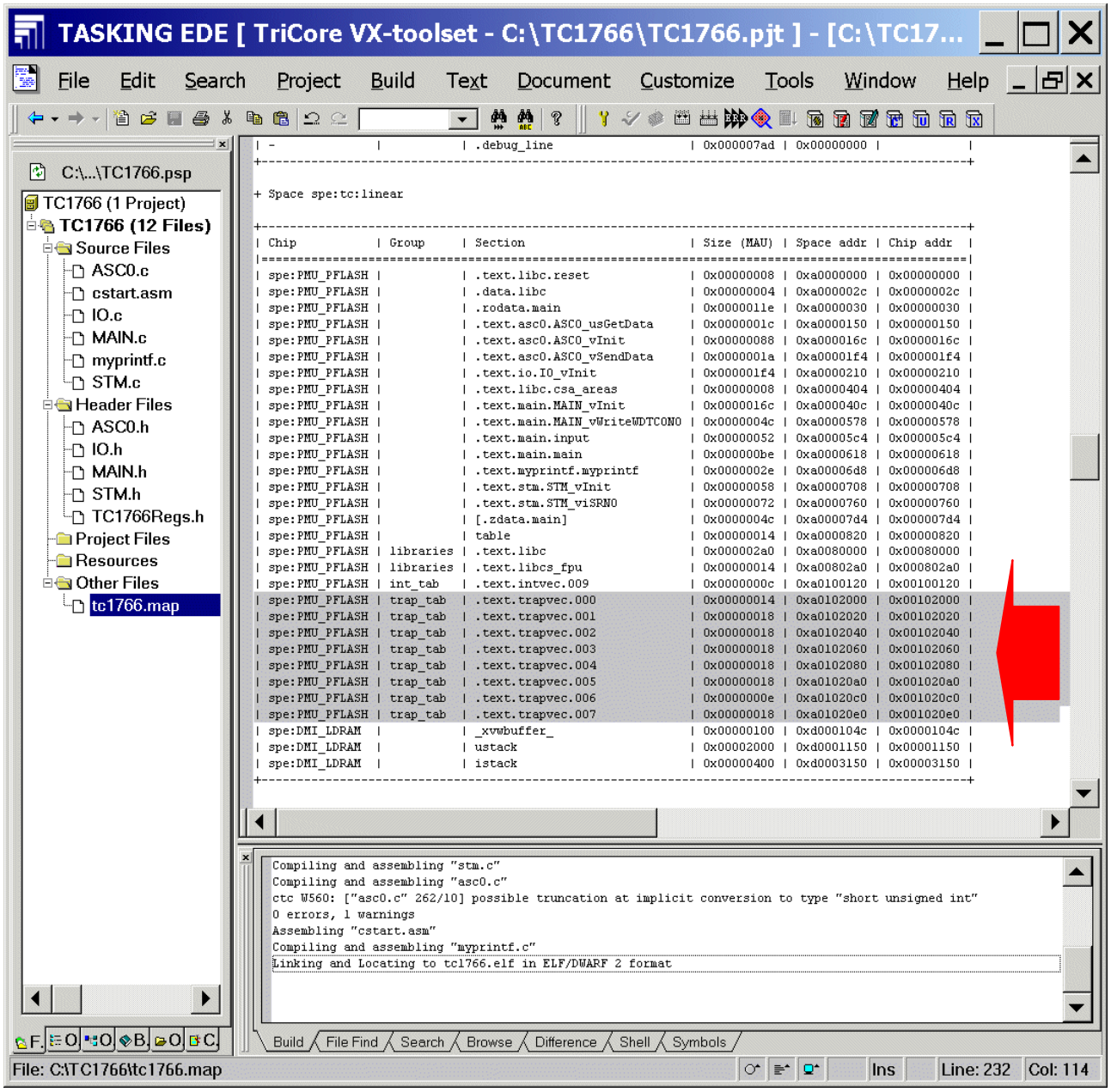
See Map File:

Interrupt Vector Table:



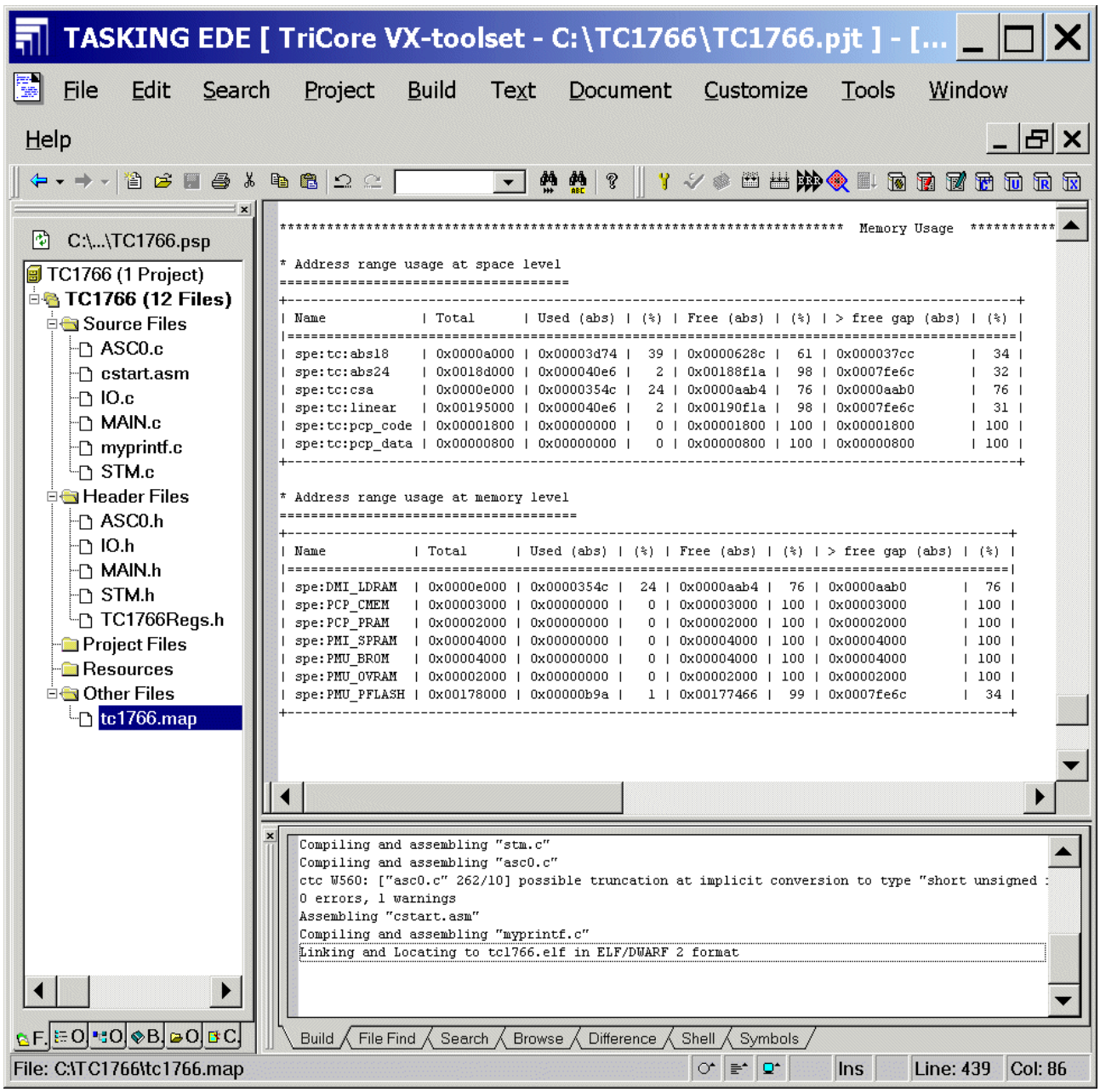
Note:
Click here to see Memory Map

Trap Vector Table:



Note:
Click here to see Memory Map

Memory Usage:

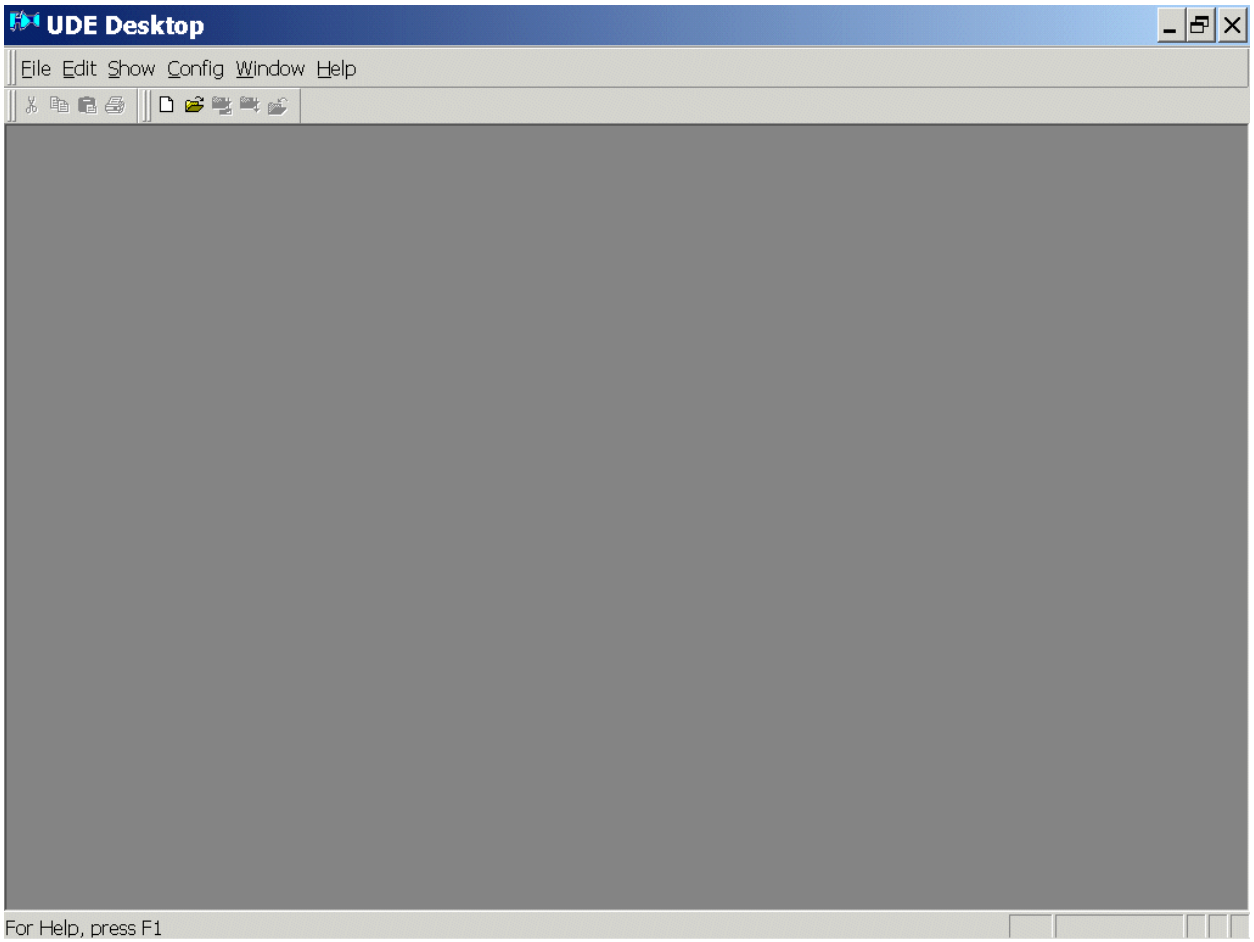


Now you can close your project and Tasking EDE:

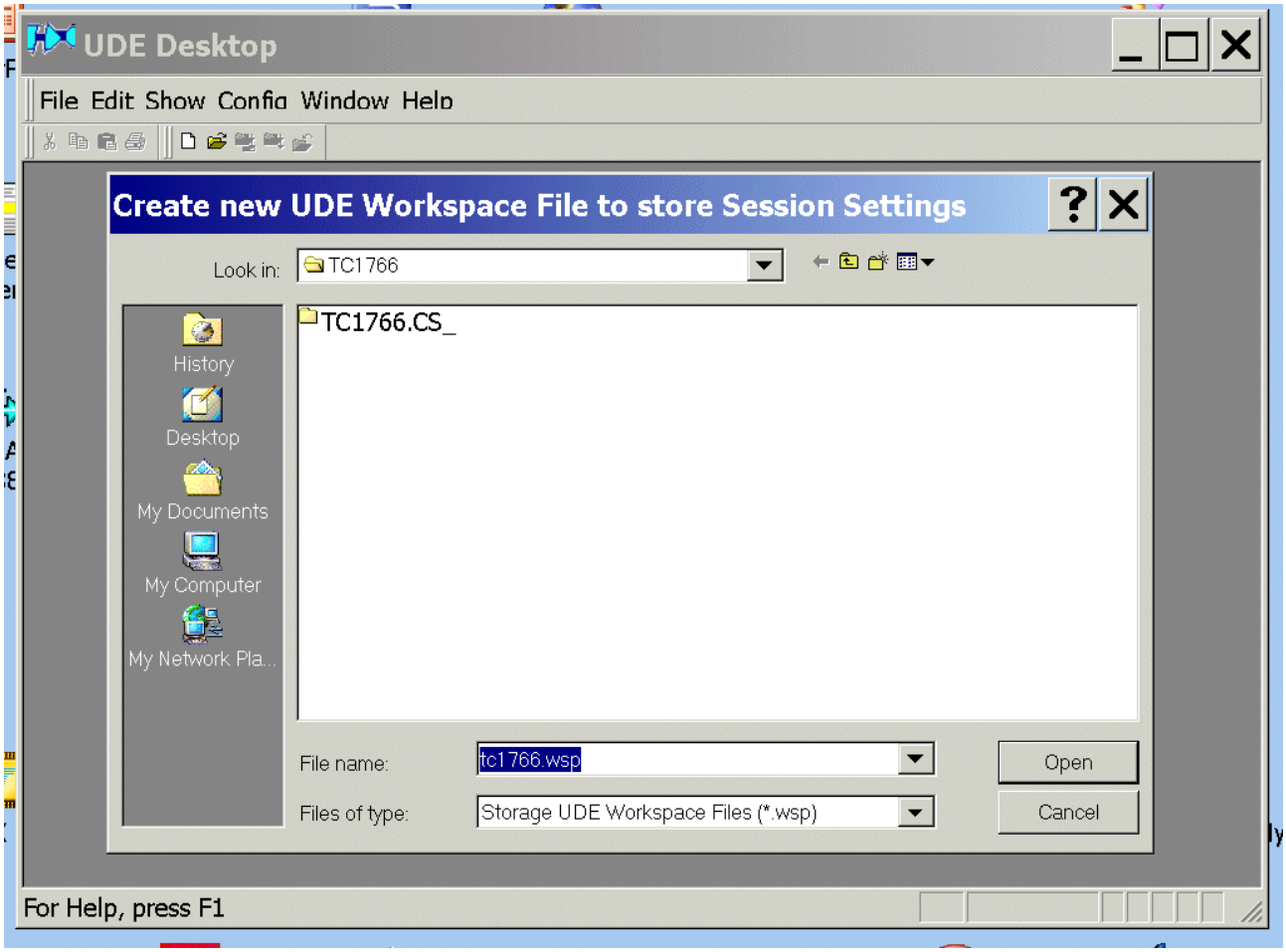
File - Close Project Space
File - Exit

5.) Programming is now complete. You can now load and run your program:

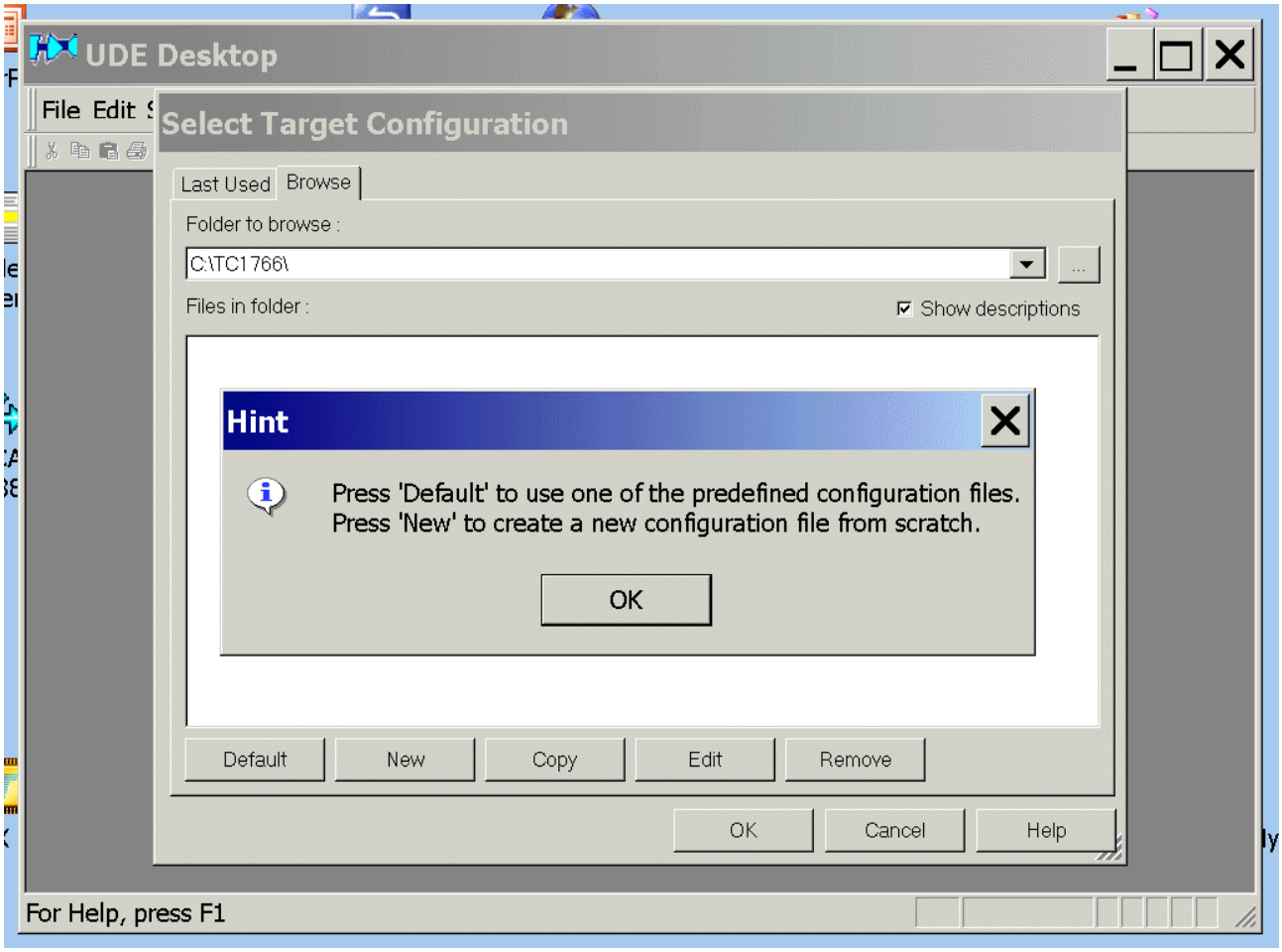
Start pls-Debugger



File – New Workspace



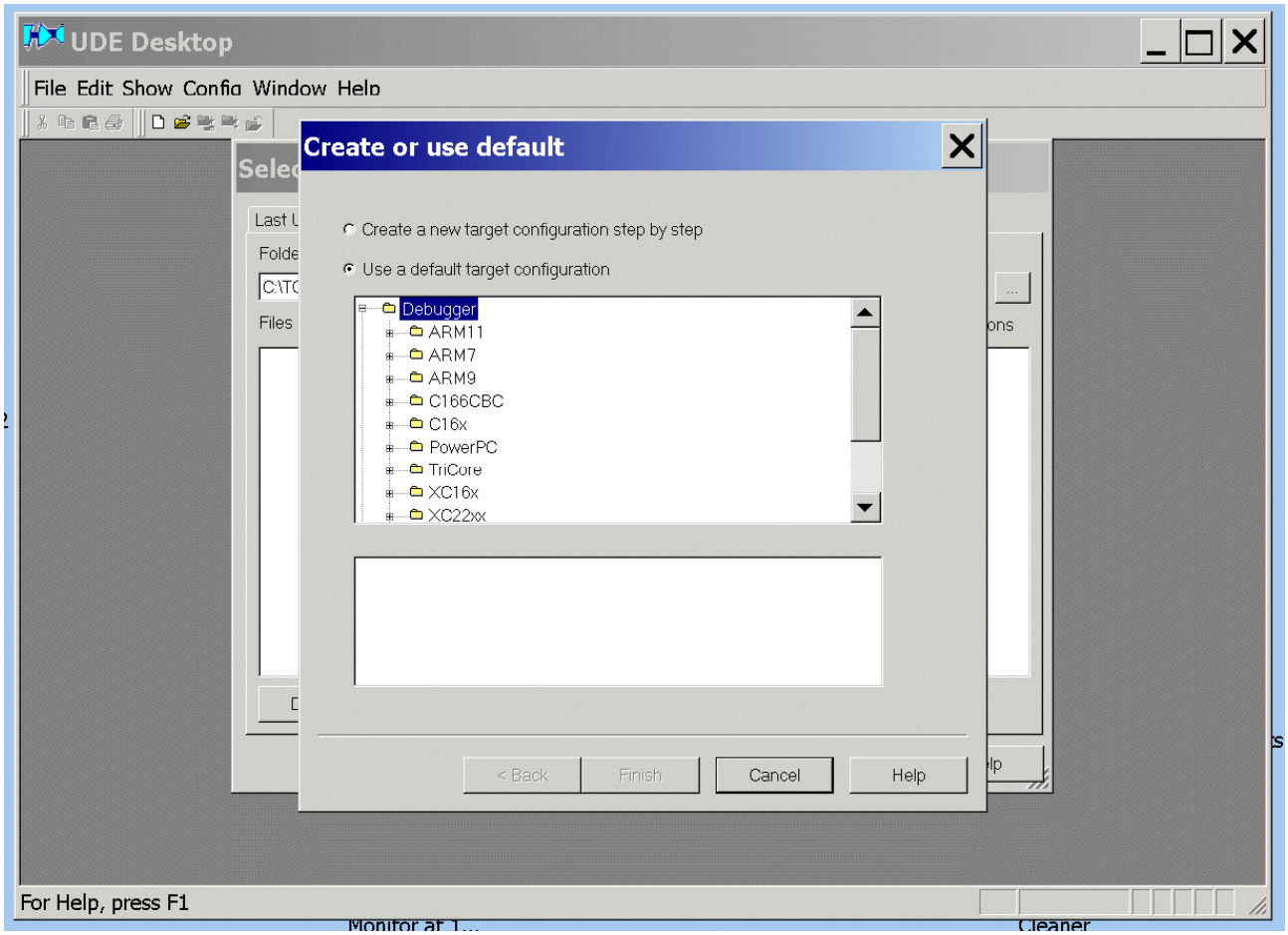
Open



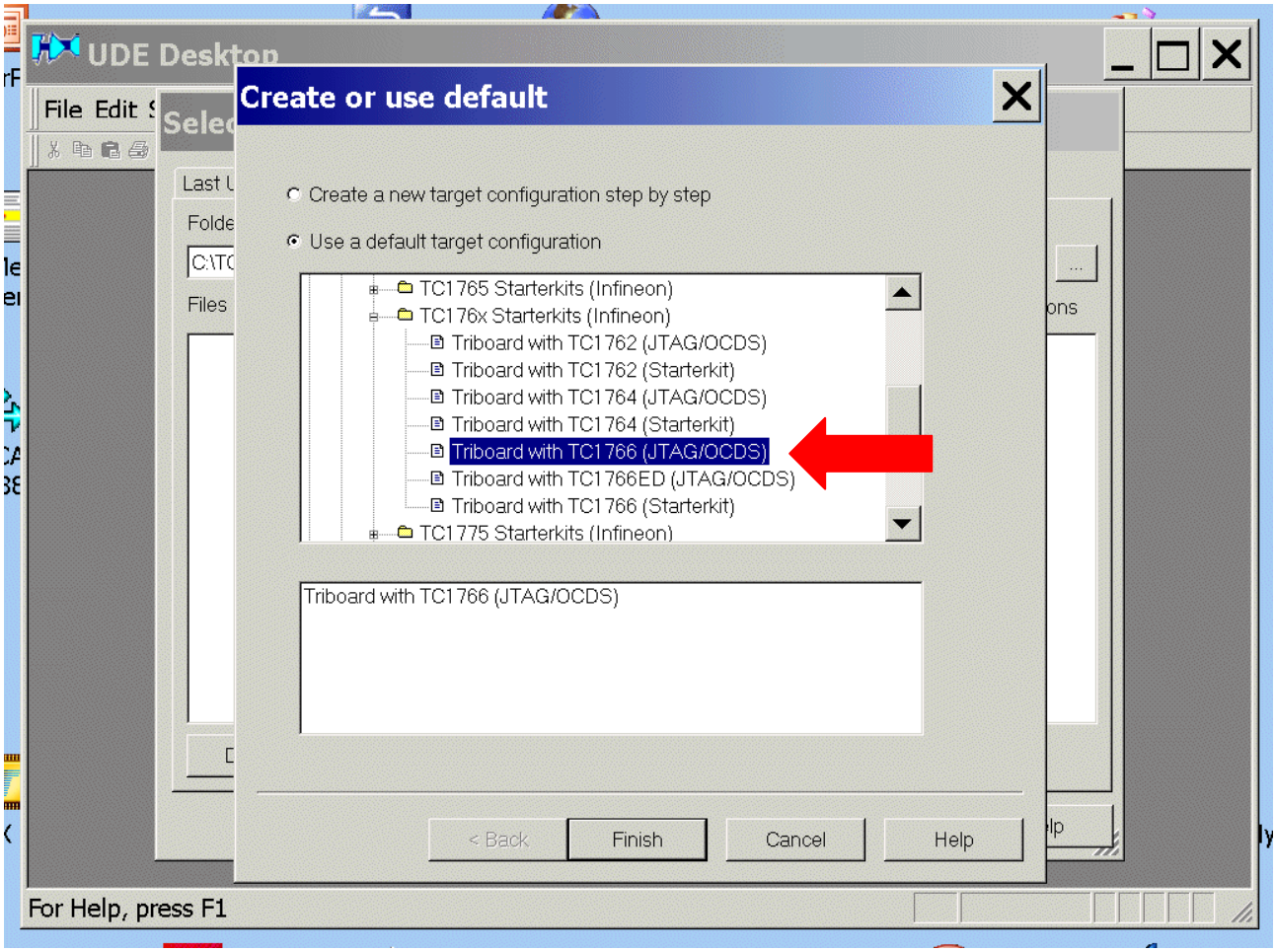
Click OK

Press Default

Create or use default: ⊙ Use a default target configuration: expand Debugger

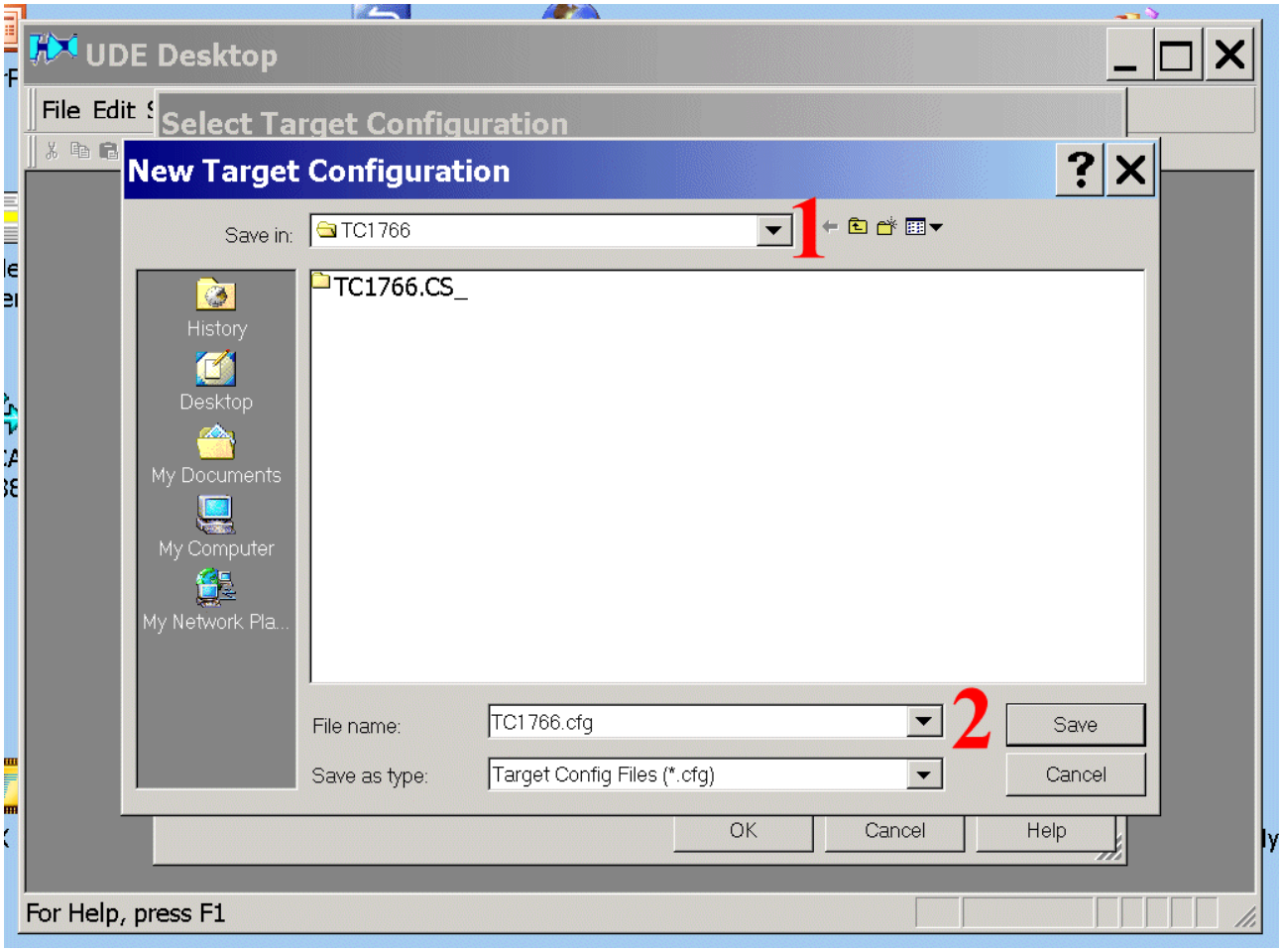


Create or use default: ⊙ Use a default target configuration:
select Triboard with TC1766 (JTAG/OCDS)

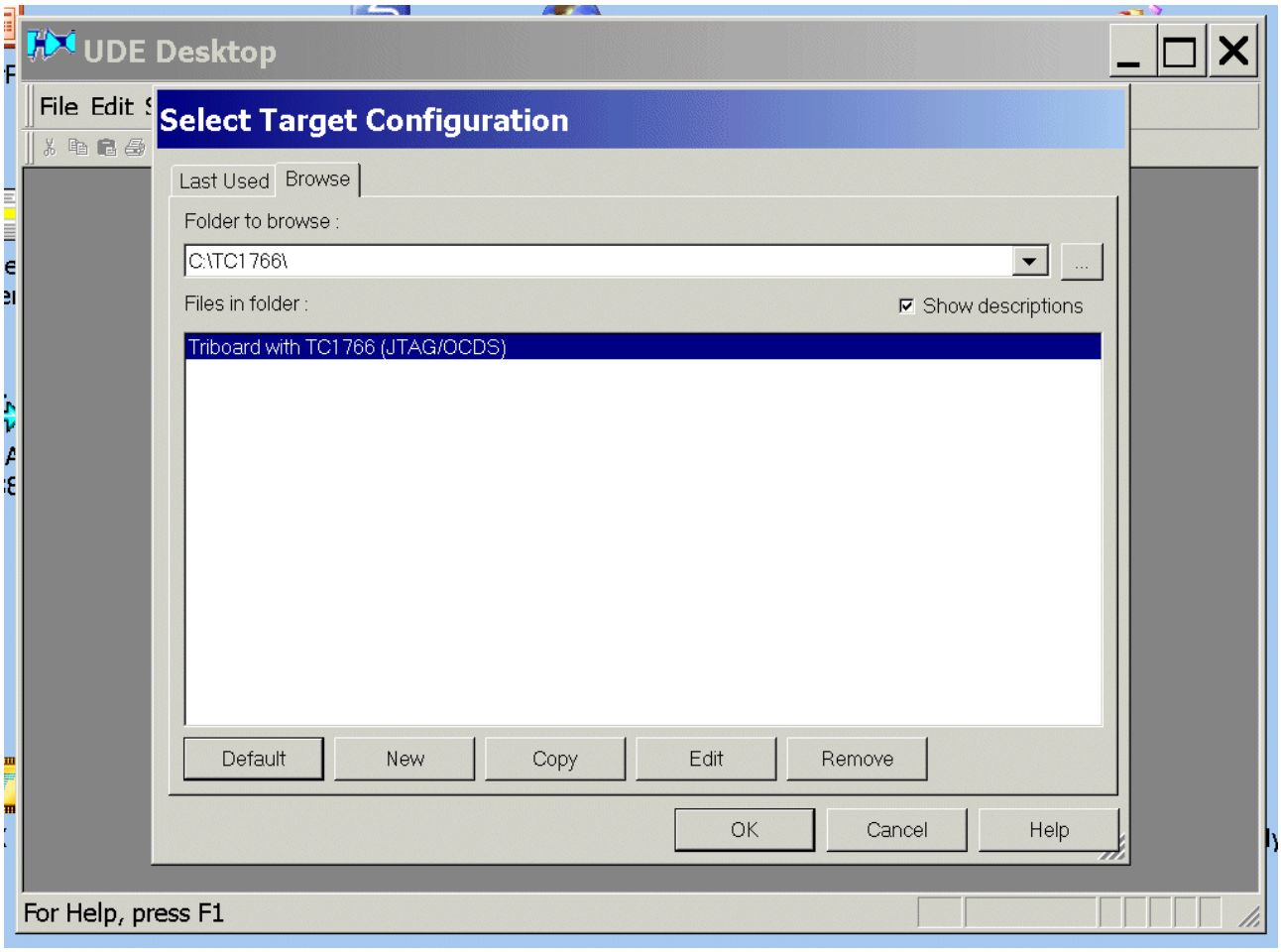


Click Finish

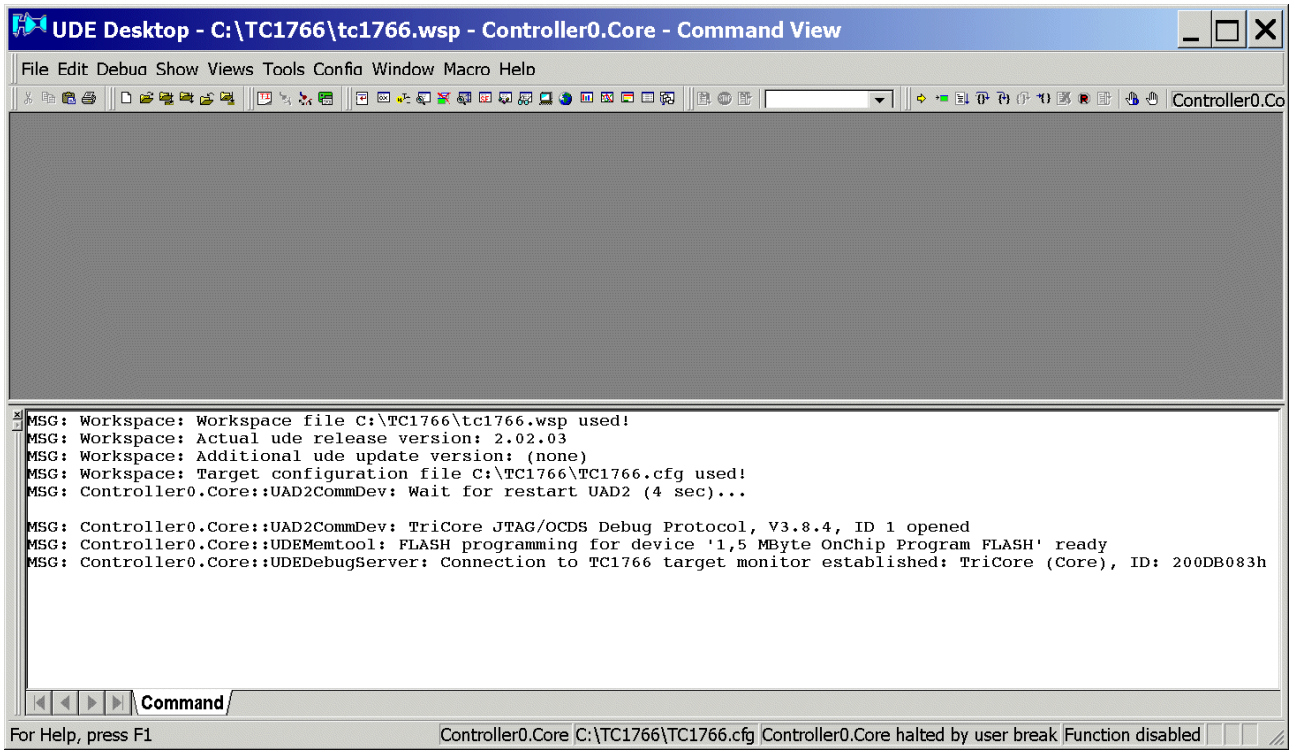
New Target Configuration: Save in: select C:\TC1766 (1)
New Target Configuration: File name: change/insert TC1766 (2)



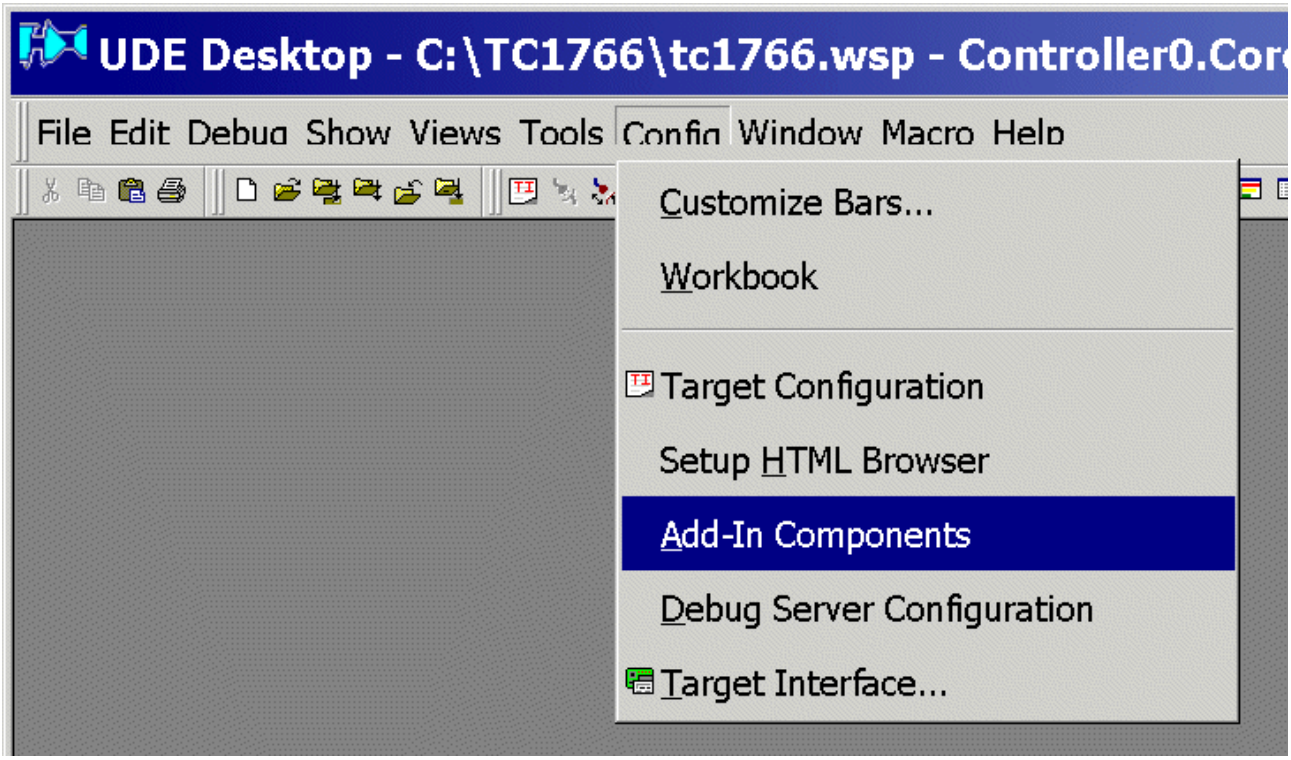
Save


UDE Desktop
File Edit
Select Target Configuration
Last Used
Browse
Folder to browse :
C:\TC1766\
Files in folder :
Show descriptions
Triboard with TC1766 (JTAG/OCDS)
Default
New
Copy
Edit
Remove
OK
Cancel
Help
For Help, press F1

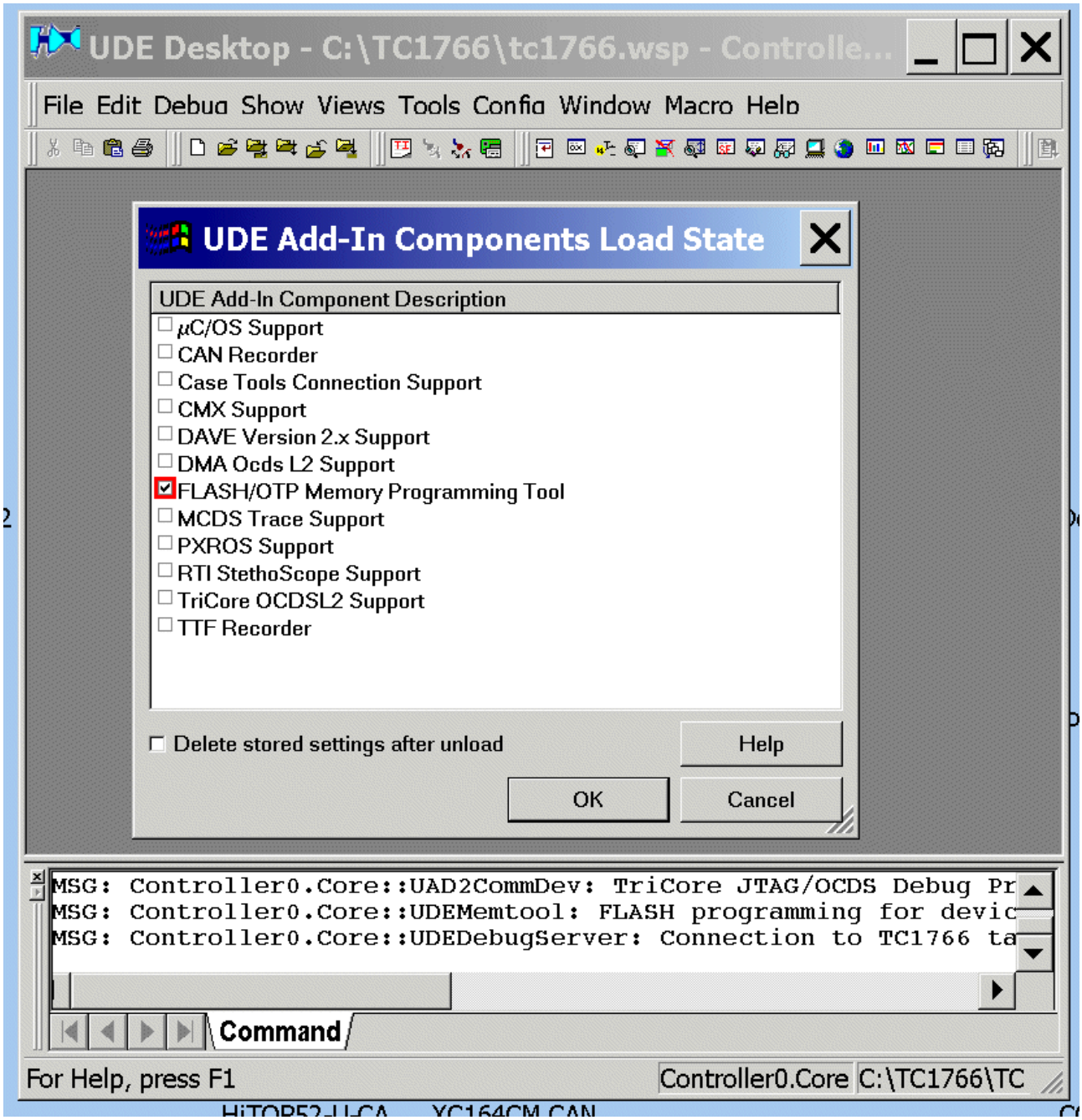
OK


UDE Desktop - C:\TC1766\tc1766.wsp - Controller0.Core - Command View
File Edit Debug Show Views Tools Config Window Macro Help
Controller0.Co
MSG: Workspace: Workspace file C:\TC1766\tc1766.wsp used!
MSG: Workspace: Actual ude release version: 2.02.03
MSG: Workspace: Additional ude update version: (none)
MSG: Workspace: Target configuration file C:\TC1766\TC1766.cfg used!
MSG: Controller0.Core::UAD2CommDev: Wait for restart UAD2 (4 sec)...
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.8.4, ID 1 opened
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '1,5 MByte OnChip Program FLASH' ready
MSG: Controller0.Core::UDEDebugServer: Connection to TC1766 target monitor established: TriCore (Core), ID: 200DB083h
Command
For Help, press F1
Controller0.Core
C:\TC1766\TC1766.cfg
Controller0.Core halted by user break
Function disabled

Config – Add-In Components

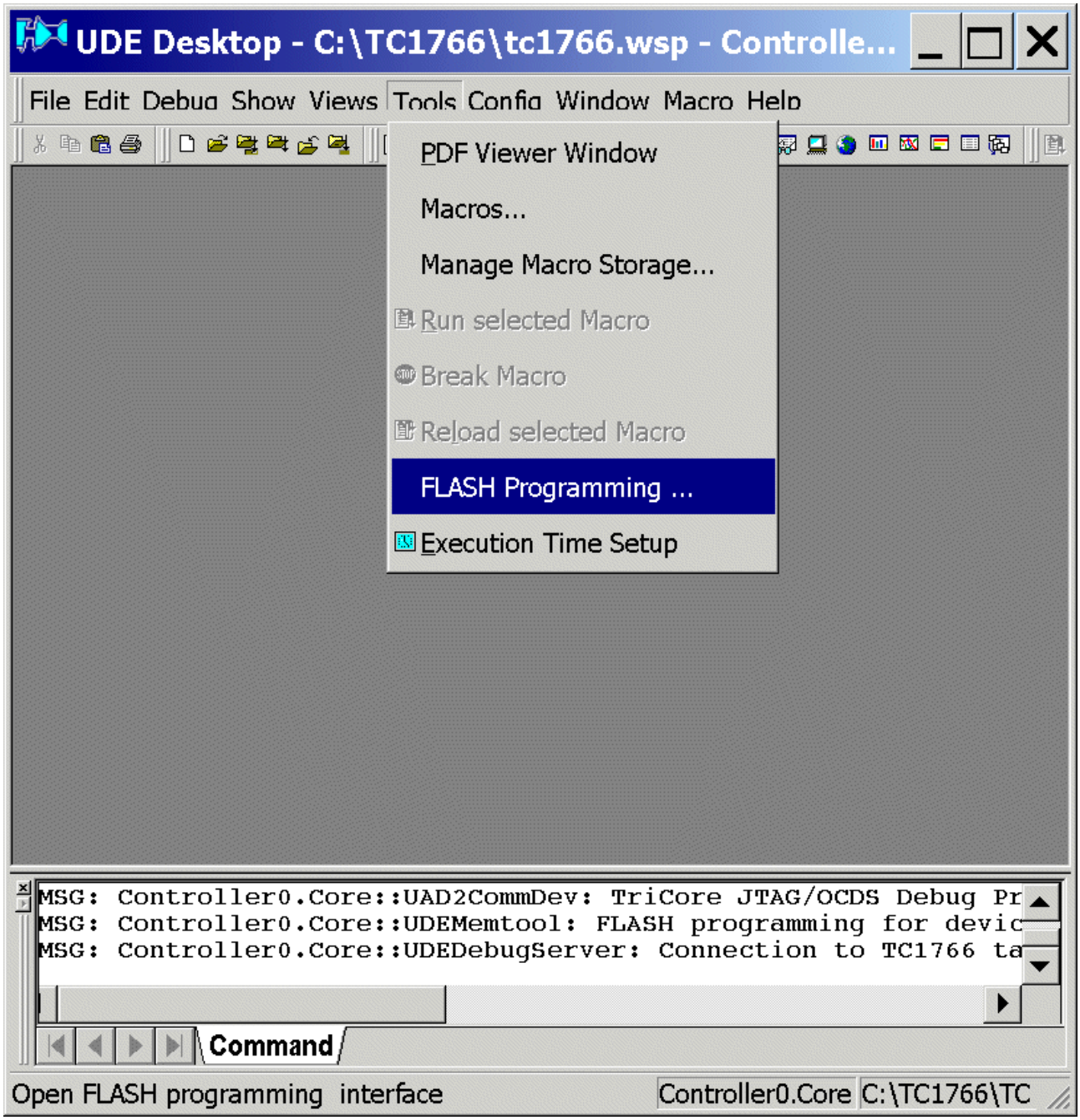


UDE Add-In Components Load State:
UDE Add-In Component Description check/tick ✓ FLASH/OTP Memory Programming Tool

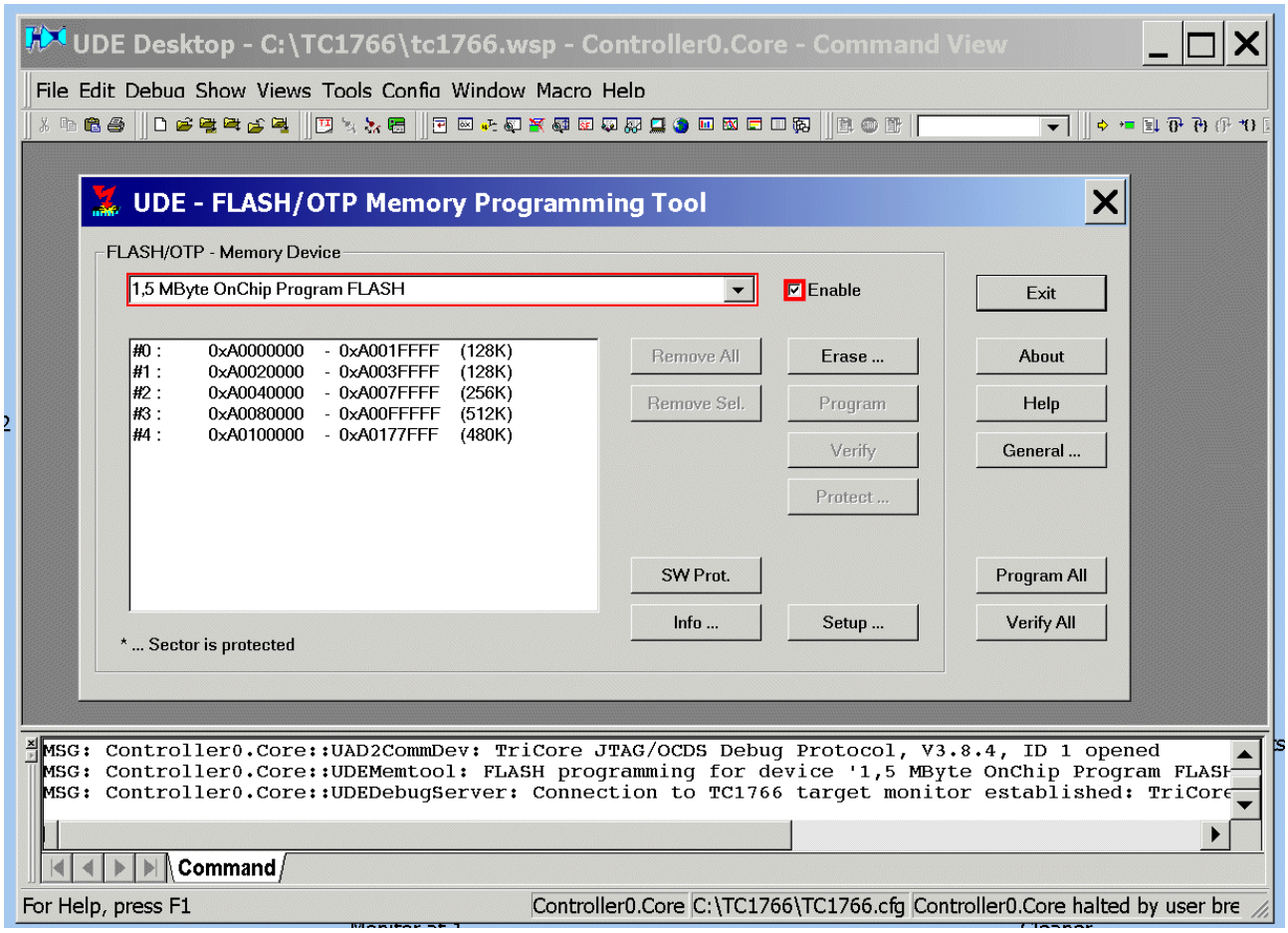


OK

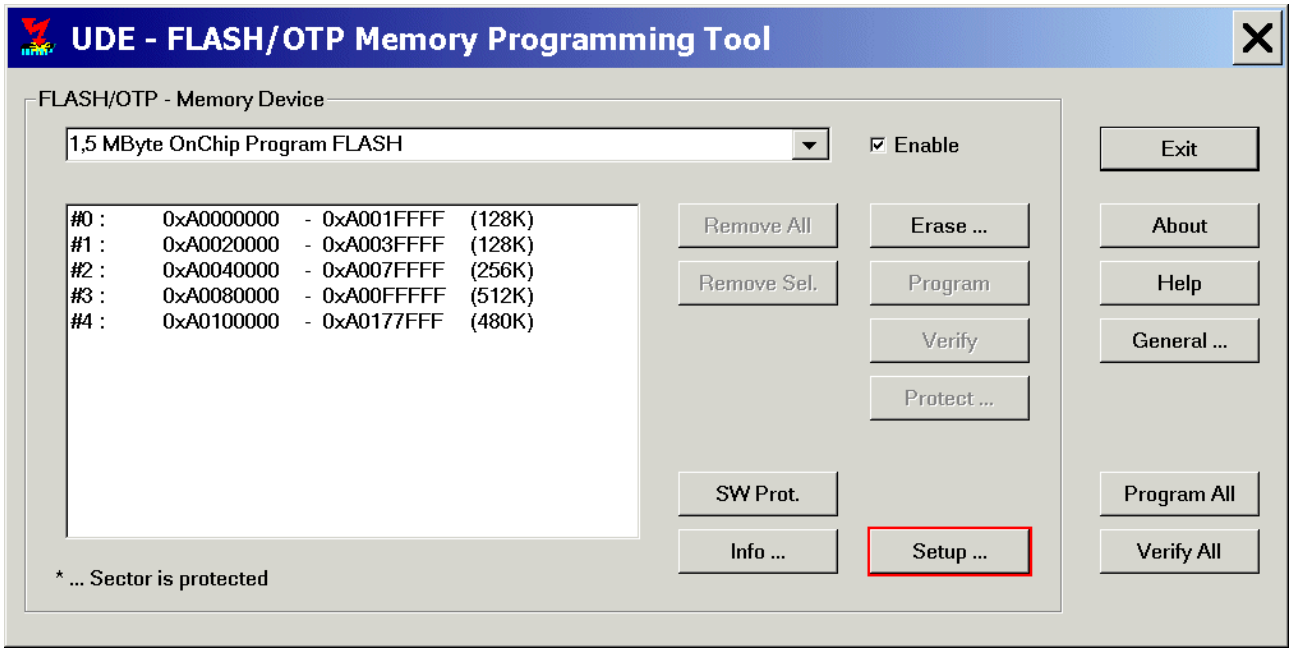
Tools – FLASH Programming …



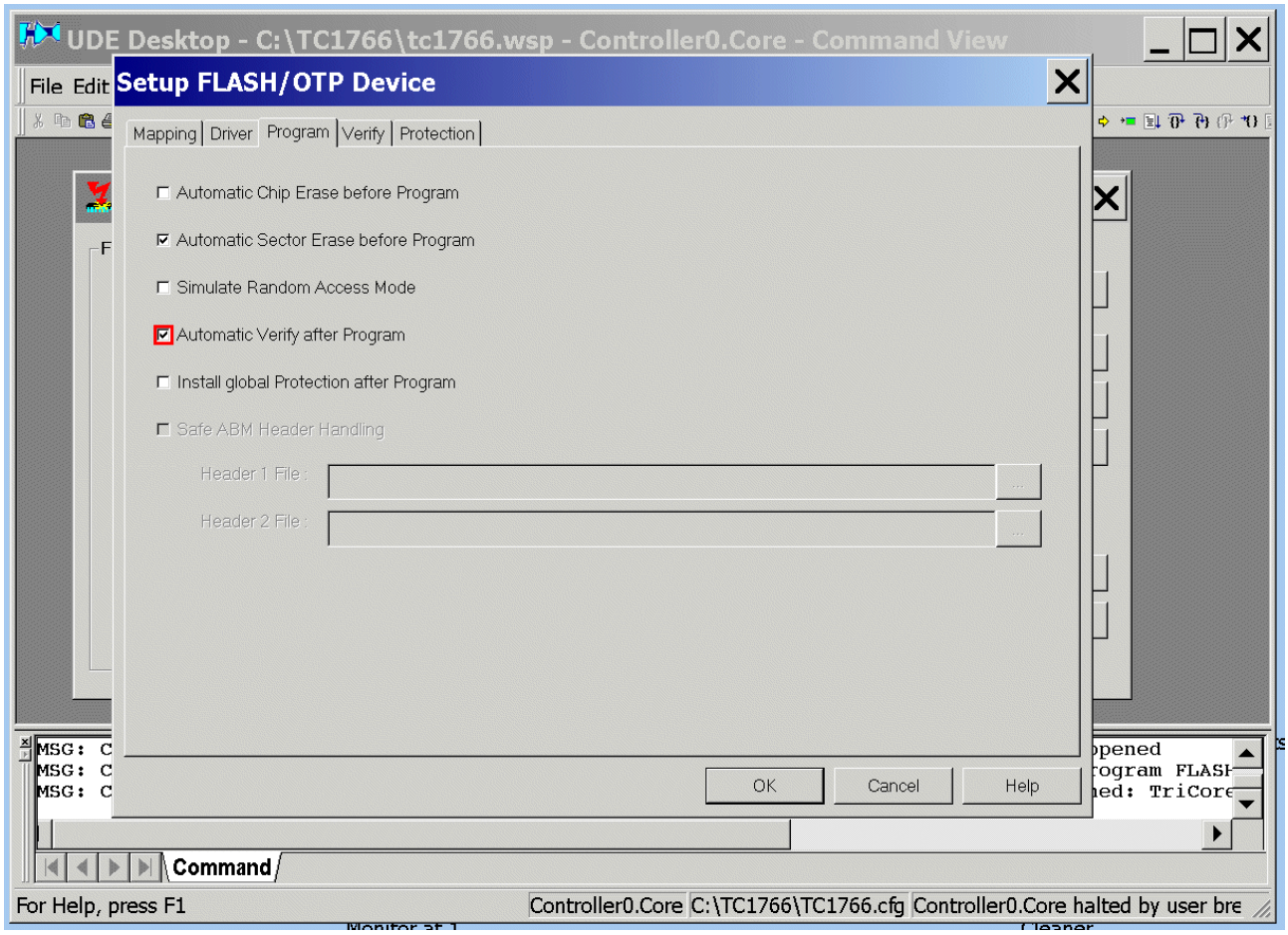
FLASH/OTP – Memory Device: check/select 1,5 MByte OnChip Program FLASH
FLASH/OTP – Memory Device: check/tick ✓ Enable



Click Setup …

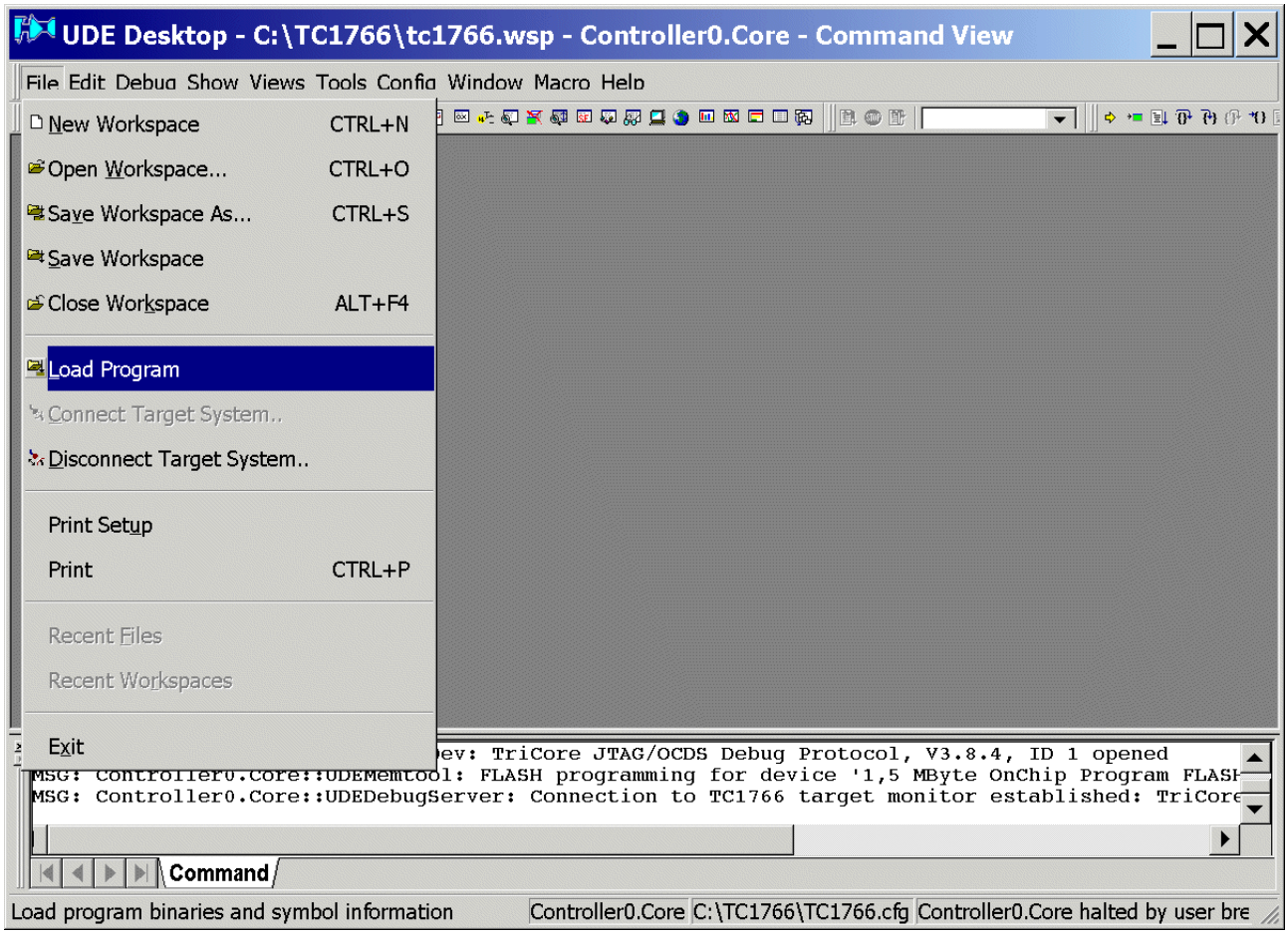

UDE - FLASH/OTP Memory Programming Tool
FLASH/OTP - Memory Device
1,5 MByte OnChip Program FLASH
Enable
Exit
#0 : 0xA0000000 - 0xA001FFFF (128K)
#1 : 0xA0020000 - 0xA003FFFF (128K)
#2 : 0xA0040000 - 0xA007FFFF (256K)
#3 : 0xA0080000 - 0xA00FFFFF (512K)
#4 : 0xA0100000 - 0xA0177FFF (480K)
Remove All
Erase ...
About
Remove Sel.
Program
Help
Verify
General ...
Protect ...
SW Prot.
Program All
Info ...
Setup ...
Verify All
* ... Sector is protected

Program: tick ✓ Automatic Verify after Program

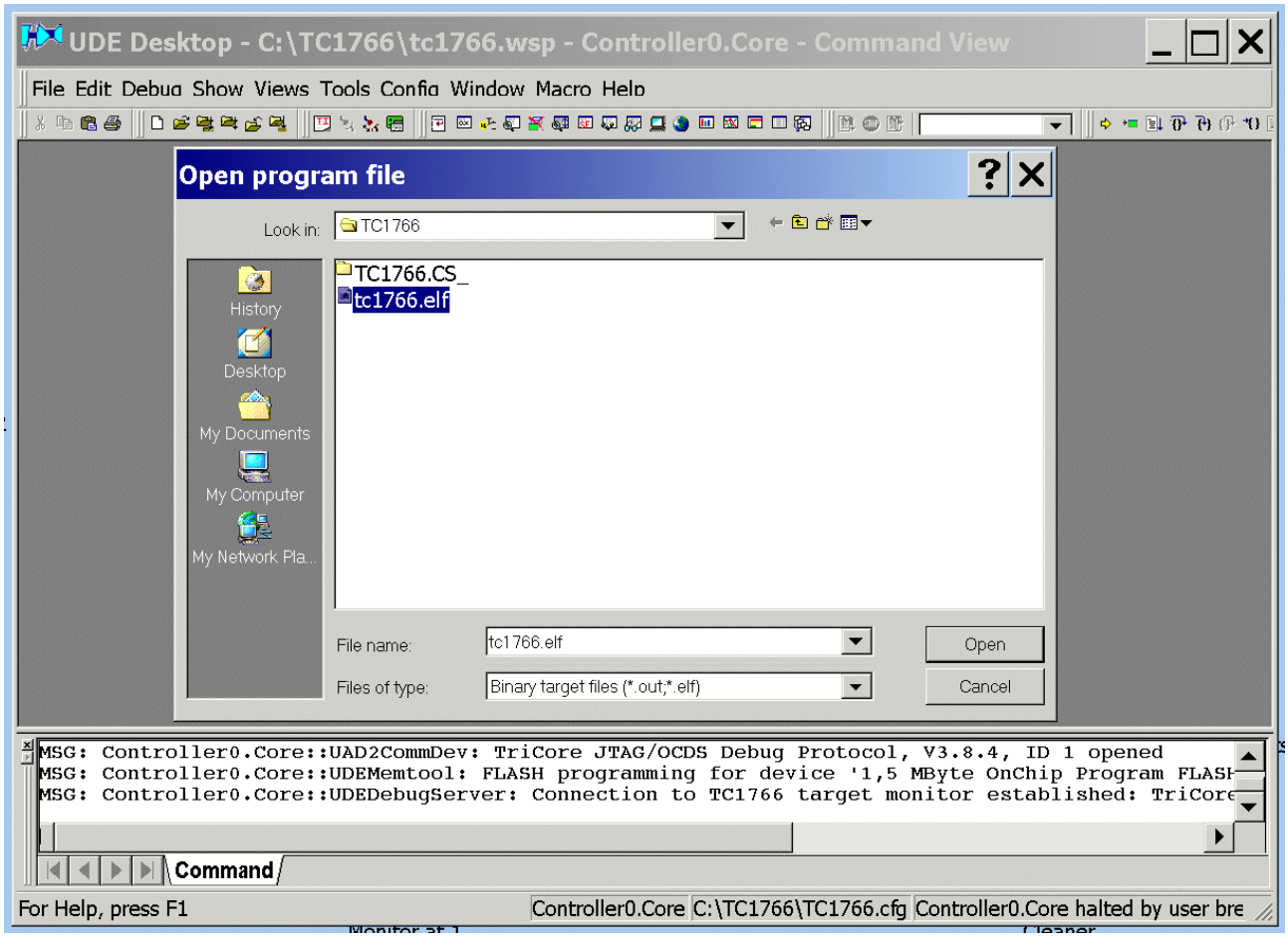


OK
Exit

File – Load Program

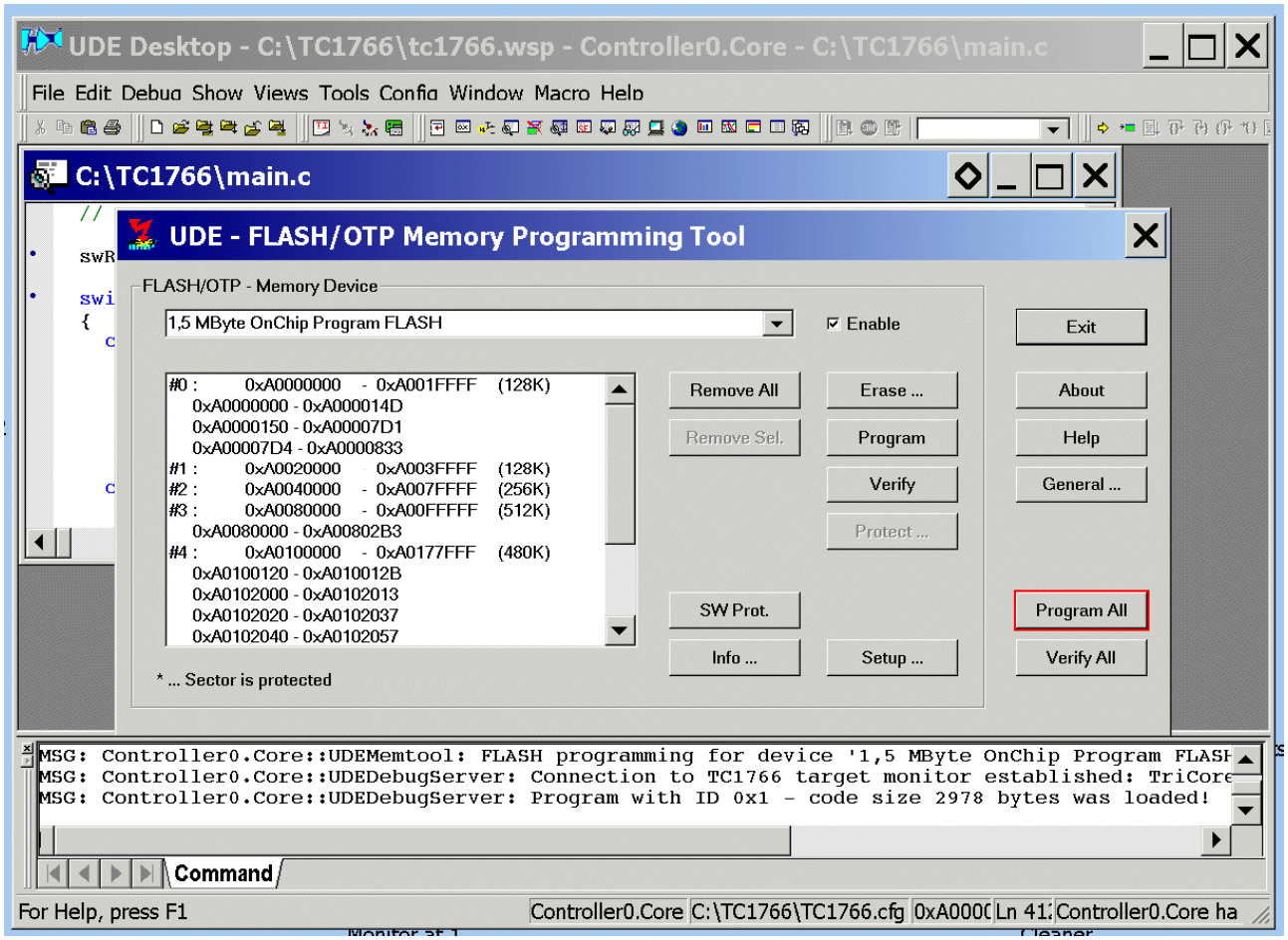


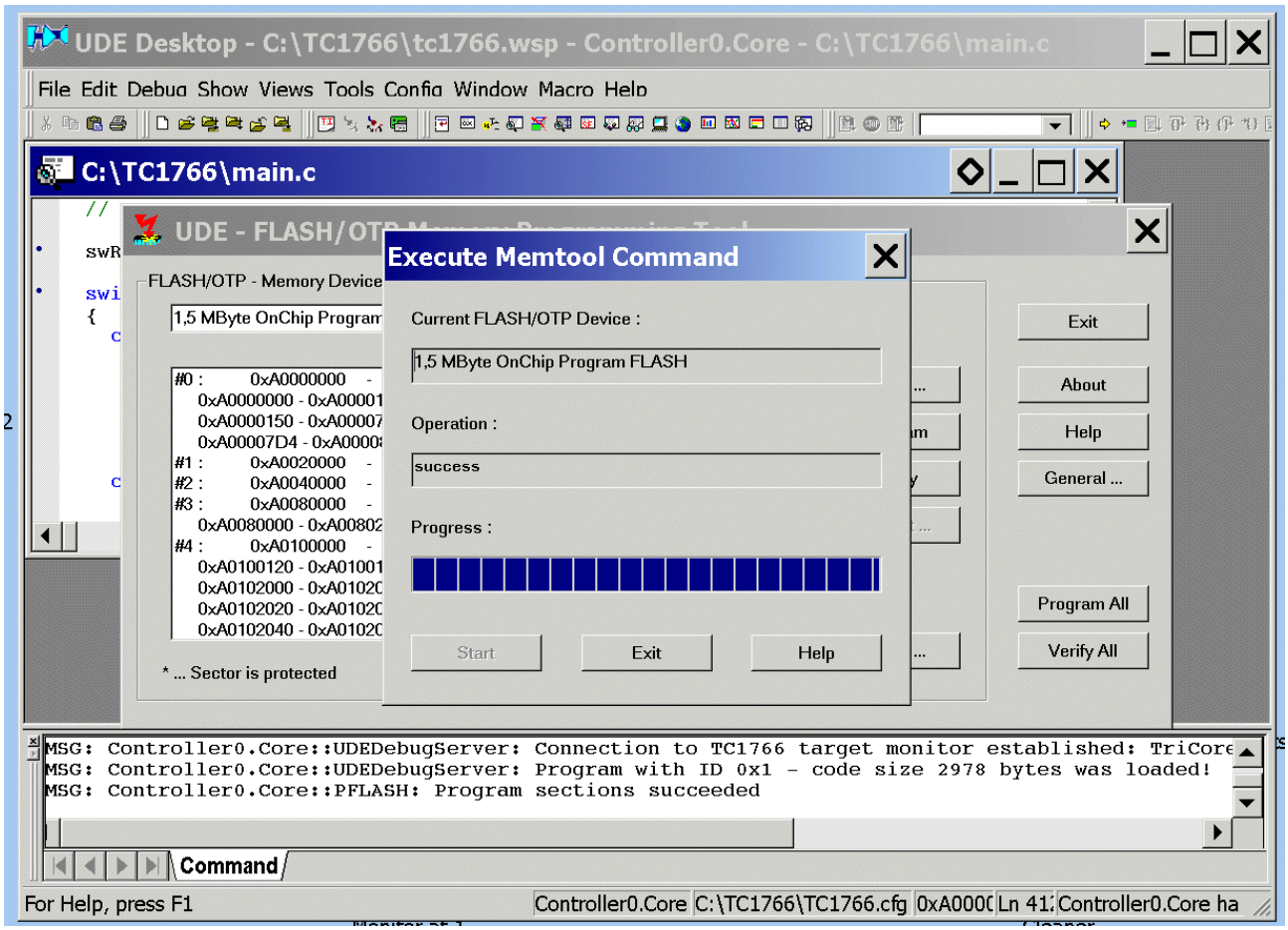
Open program file: Look in: select TC1766
Open program file: File name: select tc1766.elf



Open

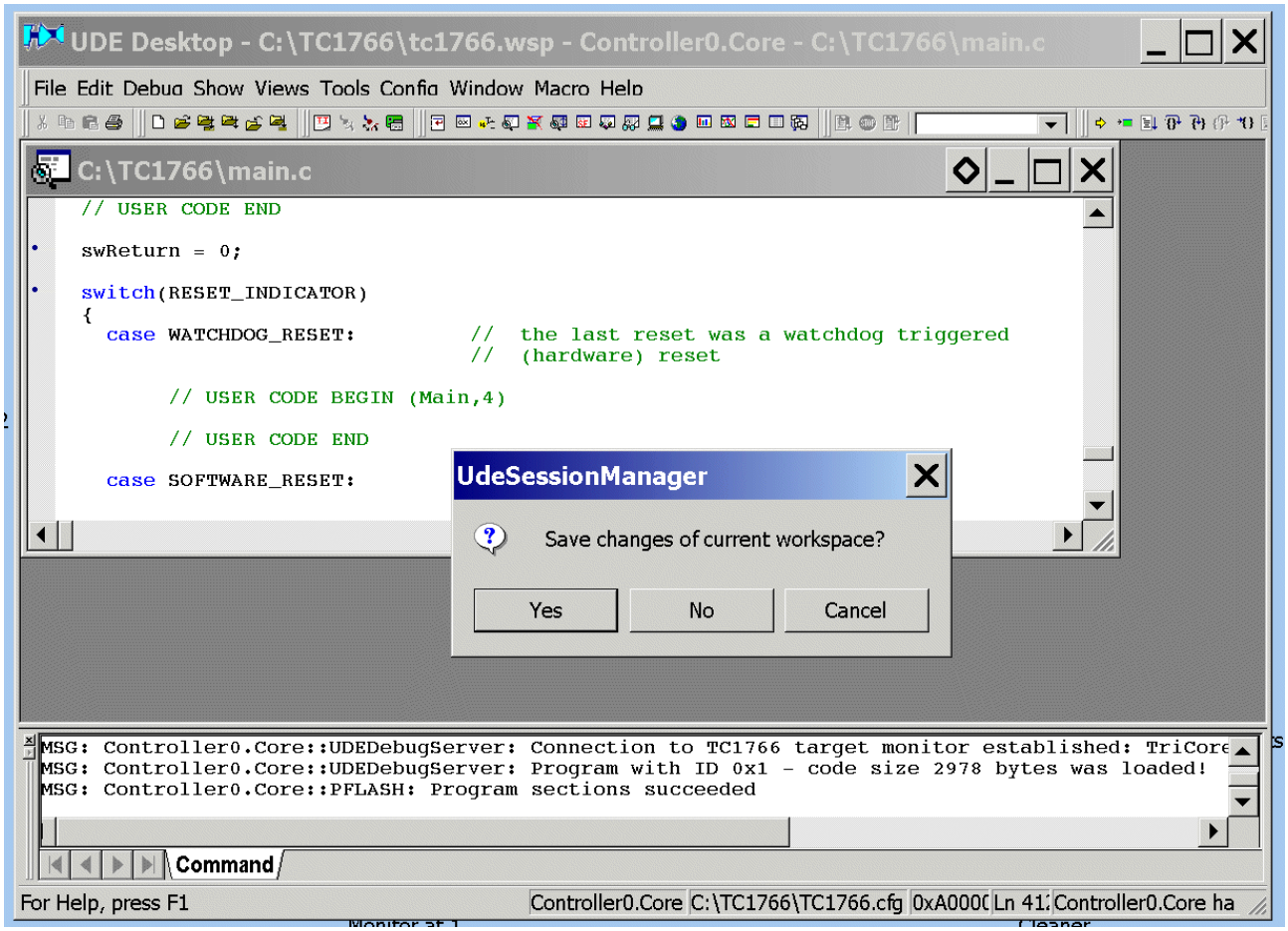
Click Program All




UDE Desktop - C:\TC1766\tc1766.wsp - Controller0.Core - C:\TC1766\main.c
File Edit Debug Show Views Tools Config Window Macro Help
C:\TC1766\main.c
UDE - FLASH/OTP
Execute Memtool Command
FLASH/OTP - Memory Device
1,5 MByte OnChip Program
Current FLASH/OTP Device :
1,5 MByte OnChip Program FLASH
Operation :
success
Progress :
#0 : 0xA0000000 -
0xA0000000 - 0xA00001
0xA0000150 - 0xA00007
0xA00007D4 - 0xA0000
#1 : 0xA0020000 -
#2 : 0xA0040000 -
#3 : 0xA0080000 -
0xA0080000 - 0xA00802
#4 : 0xA0100000 -
0xA0100120 - 0xA01001
0xA0102000 - 0xA0102C
0xA0102020 - 0xA0102C
0xA0102040 - 0xA0102C
* ... Sector is protected
Start
Exit
Help
Exit
About
Help
General ...
Program All
Verify All
MSG: Controller0.Core::UDEDebugServer: Connection to TC1766 target monitor established: TriCore
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 2978 bytes was loaded!
MSG: Controller0.Core::PFLASH: Program sections succeeded
Command
For Help, press F1
Controller0.Core
C:\TC1766\TC1766.cfg
Ln 41
Controller0.Core ha

Exit
Exit

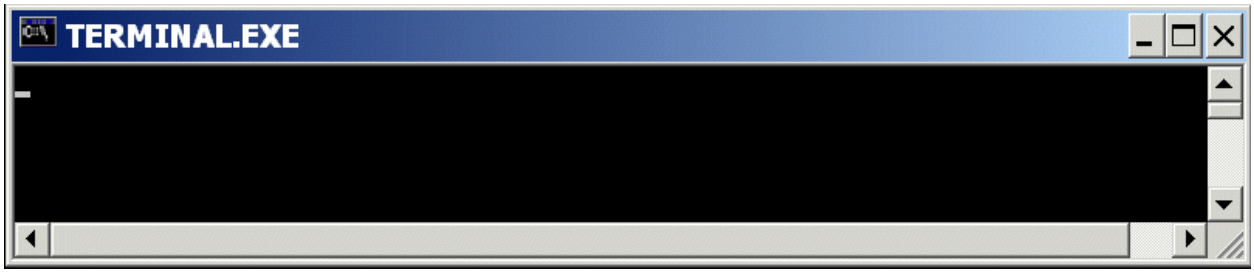
File – Close Workspace



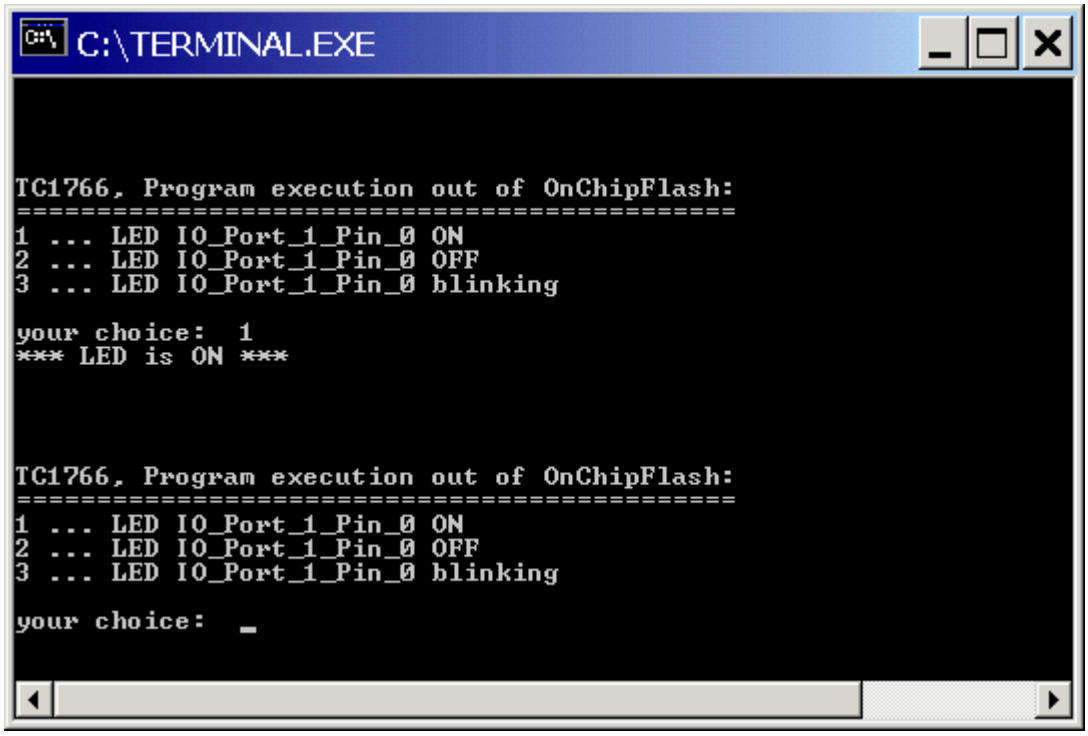
Yes

File – Exit

Execute any terminal program
(9600 Baud, 8 bit Data, no Parity-Bit, 1 Stop-Bit, Xon/Xoff Protocol):



Power-On the Board and see the result:



Conclusion:

In this step-by-step book you have learned how to use the TC1766 Starter Kit together with the Tasking 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 TC1766 Starter 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 TC1766 Starter Kit!

Note:

There are step-by-step books for 8 bit microcontrollers (e.g. XC866, XC88x, and XC878), 16 bit microcontrollers (e.g. C16x, XC16x, and XE16x) and 32 bit microcontrollers (e.g. TC1796 and TC1130).

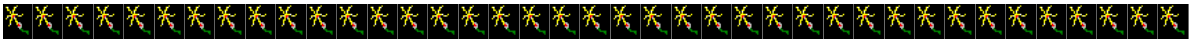
All these step-by-step books use the same microcontroller resources and the same example code.

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.

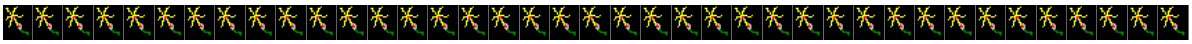
6.) Feedback (TC1766): Your opinions, suggestions and/or criticisms



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