AP32110

TC1796

TC1796 "Cookery-Book" for a "Hello world" application

Microcontrollers

Never stop thinking

Edition 2008-07-16

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:	2006-07	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 7.

Introduction:

This “Appnote” is an Infineon Hands-On-Training.
It will help inexperienced users to get an TC1796 Evaluation-Board / Starter-Kit-Board up and running.

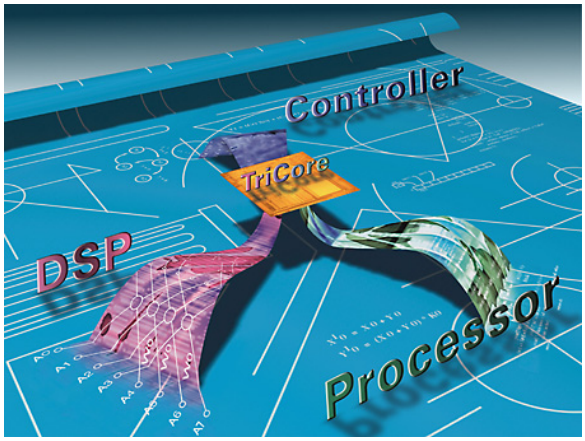
With this Hands-On-Training / Cookery-Book / step-by-step-book you should be able to get your first useful program in less than 2 hours.

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 your needs.
You can connect either a part of - or your entire application to the Starter-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.

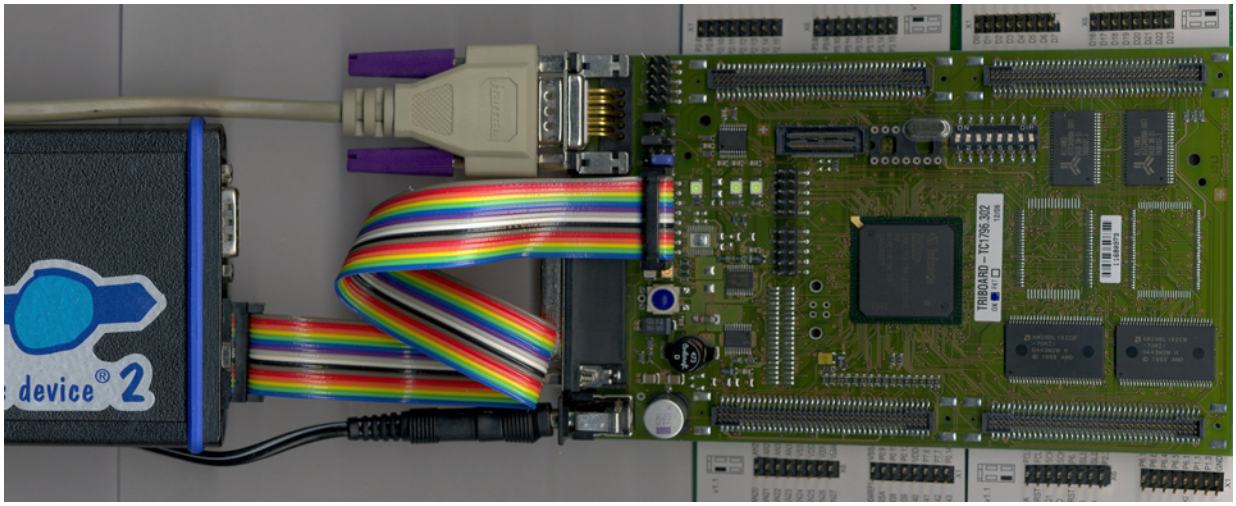
The main chapters are:
Chapter 4: Program_Execution_From_Scratch-Pad-RAM_SPRAM
Chapter 5: Program_Execution_From_OnChipProgramFlash
Chapter 6: Program_Execution_From_OnBoardProgramFlash
Chapter 7: Program_Execution_From_OnBoardProgramFlash_Burst-Mode

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.

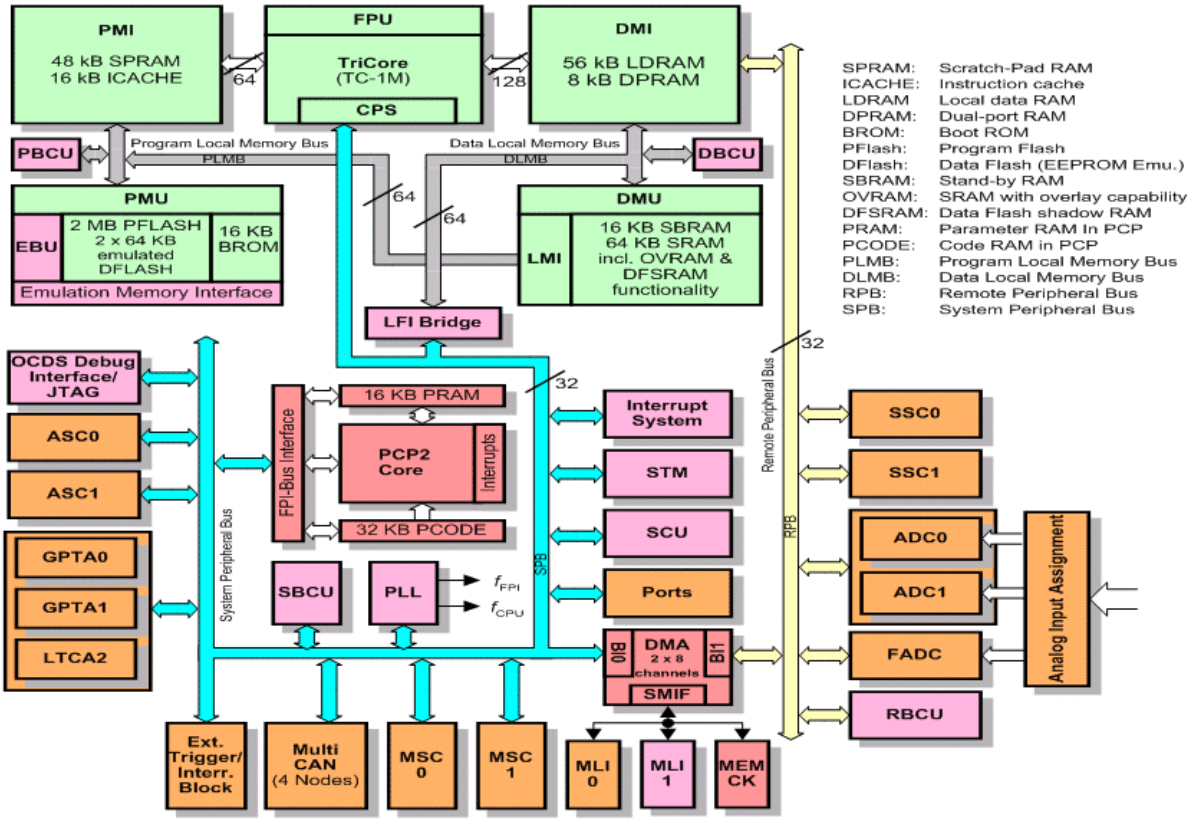
Have fun and enjoy TriCore!

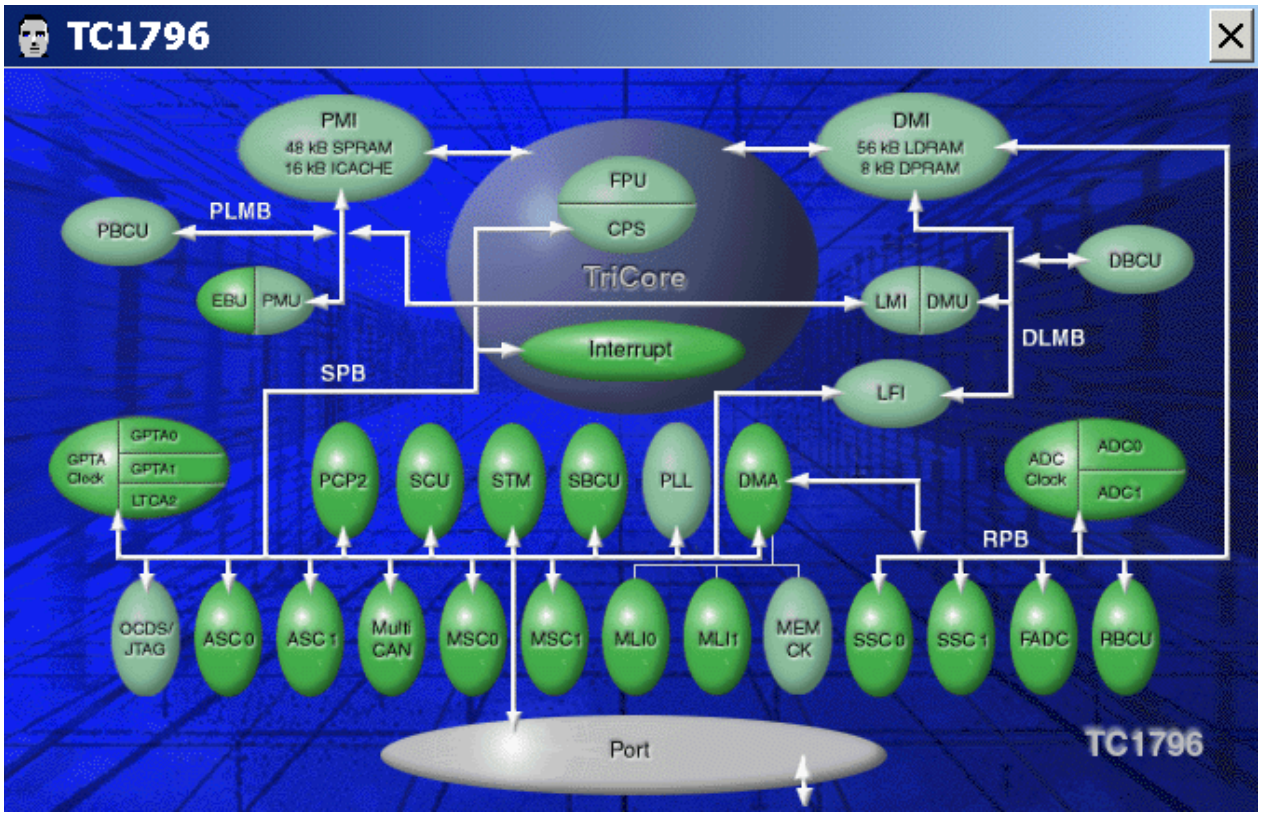


Programming Examples TC1796 Starter Kit



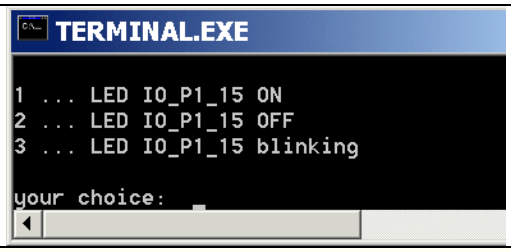
TC1796 Block Diagram





“Cookery-Book“

For your first programming example for the TC1796 Starter-Kit-Board:

	Your program: 
Chapter/ Step:	***** Recipes *****
1.)	**TC1796 Board** Power Supply, Jumper Setting, Serial cable to the notebook, pls-Debugger
2.)	**DAvE – program generator** DAvE installation (mothersystem) + DAvE Update-installation for TC1796 (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 TASKING (Altium) tool chain (EDE) - v2.3r1 Locating programs in the 48 KByte code scratchpad RAM. (SPRAM), using OnChipSRAM)
5.) 	**Using the TASKING Development Tools (C/C++/EC++ Compiler)** Programming of your application with TASKING (Altium) tool chain (EDE) - v2.3r1 Locating programs in the 2 MByte OnChipProgramFlash (PFLASH), using OnChipSRAM)
6.) 	**Using the TASKING Development Tools (C/C++/EC++ Compiler)** Programming of your application with TASKING (Altium) tool chain (EDE) - v2.3r1 Locating programs in the 4 MByte OnBoardFlash, using OnChipSRAM + OnBoardSRAM)
7.) 	**Using the TASKING Development Tools (C/C++/EC++ Compiler)** Programming of your application with TASKING (Altium) tool chain (EDE) - v2.3r1 Locating programs in the 4 MByte OnBoardFlash (Burst!), using OnChipSRAM + OnBoardSRAM

Additional exercises

8.)	Time - Measurement

Feedback

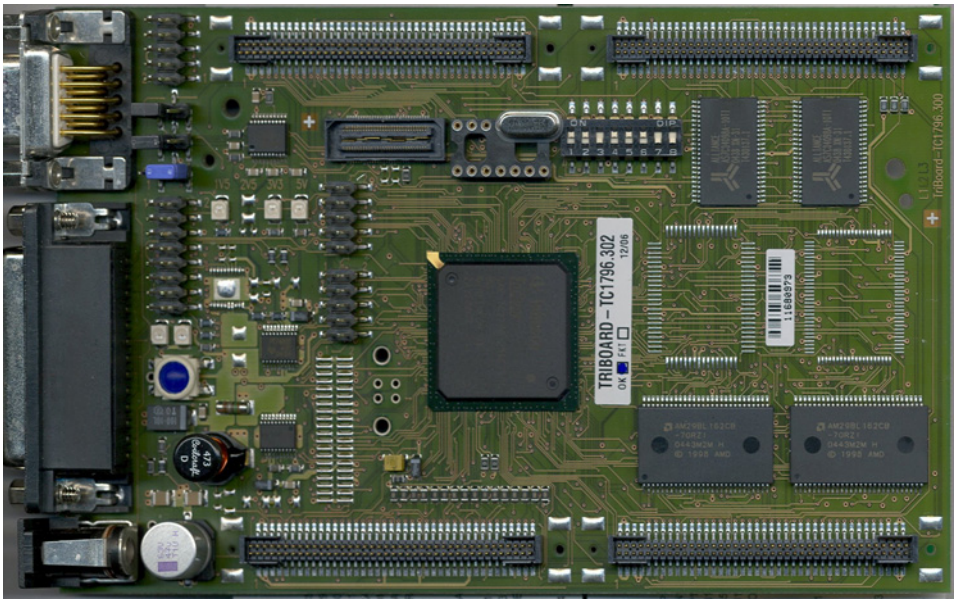
9.)	Feedback

1.) TC1796 Starter Kit Board:

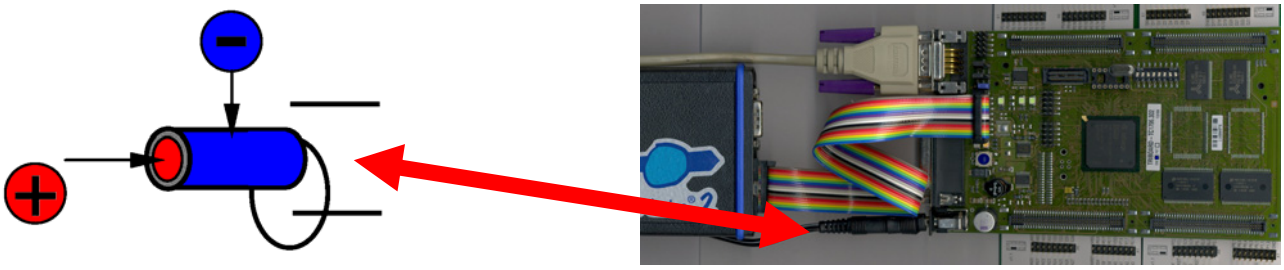
Ordering information:

Starter Kits - Type	Title	Ordering No.	Comment
SK-TC1796	TC1796 Starter Kit	B158-H8537-G1-0-7600	

Distribution Worldwide:
http://www.infineon.com/cgi-bin/ifx/portal/ep/channelView.do?channelId=-66982&pageTypeId=17224



You need a Power Supply:
The TC1796 Board requires an external power supply.
A (un)regulated DC power supply from 6 to 60 Volts can be connected to the power connector.
300mA are sufficient for the TC1796 Starterkit.



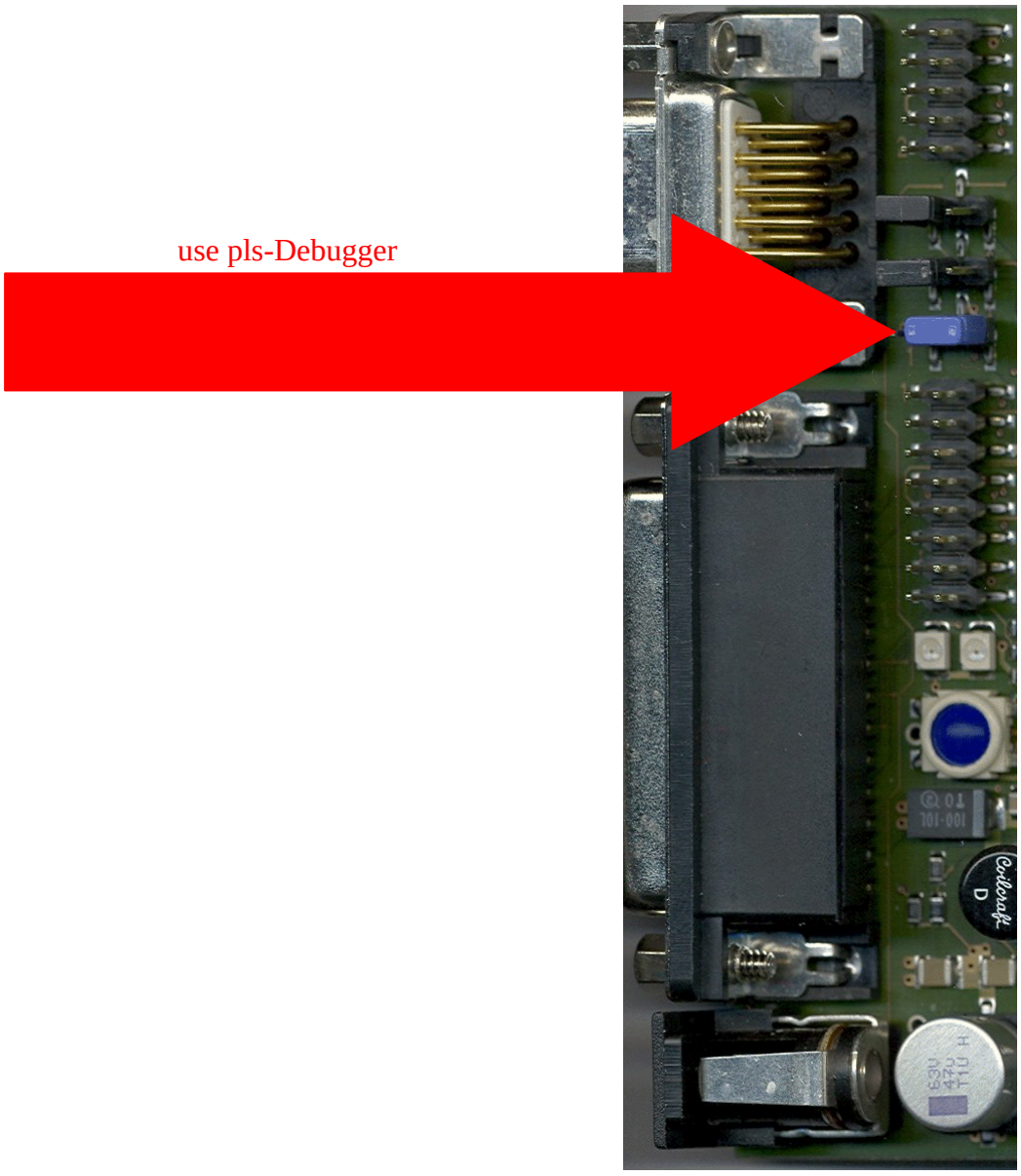
You need a RS-232 Serial Cable
(1:1; 9-pin Sub-D plug – 9-pin Sub-D connector; the “Hello World” example uses this interface):

You need the pls-Debugger (Flash-Programming und Debugging):

For further information, please refer to the TriBoard TC1796 V1.0, March 2003 Hardware Manual .
For further information, please refer to the TriBoard TC1796 V2.0, June 2004 Hardware Manual .
For further information, please refer to the TriBoard TC1796 V3.0, June 2004 Hardware Manual .
For further information, please refer to the TriBoard TC1796 V3.1, Jan. 2005 Hardware Manual .
For further information, please refer to the TriBoardManual-TC179X-V32.pdf (September 2006).

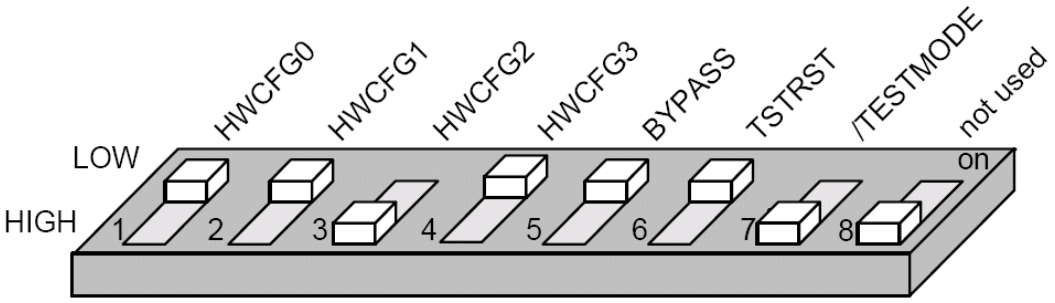
Jumper Settings (Jumper JP501):

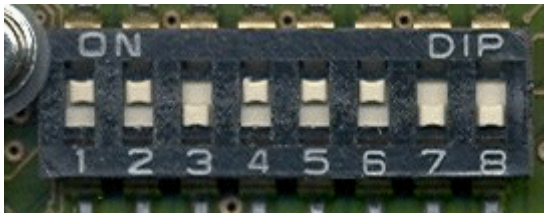
Jumper JP501
1-2 … Enable On Board Wiggler (use parallel-on-board-interface)
2-3 … Disable On Board Wiggler (use pls-Debugger)



TC1796-Execution-Environment-SPRAM:

Jumper Settings (HW-Configuration DIP-Switch):

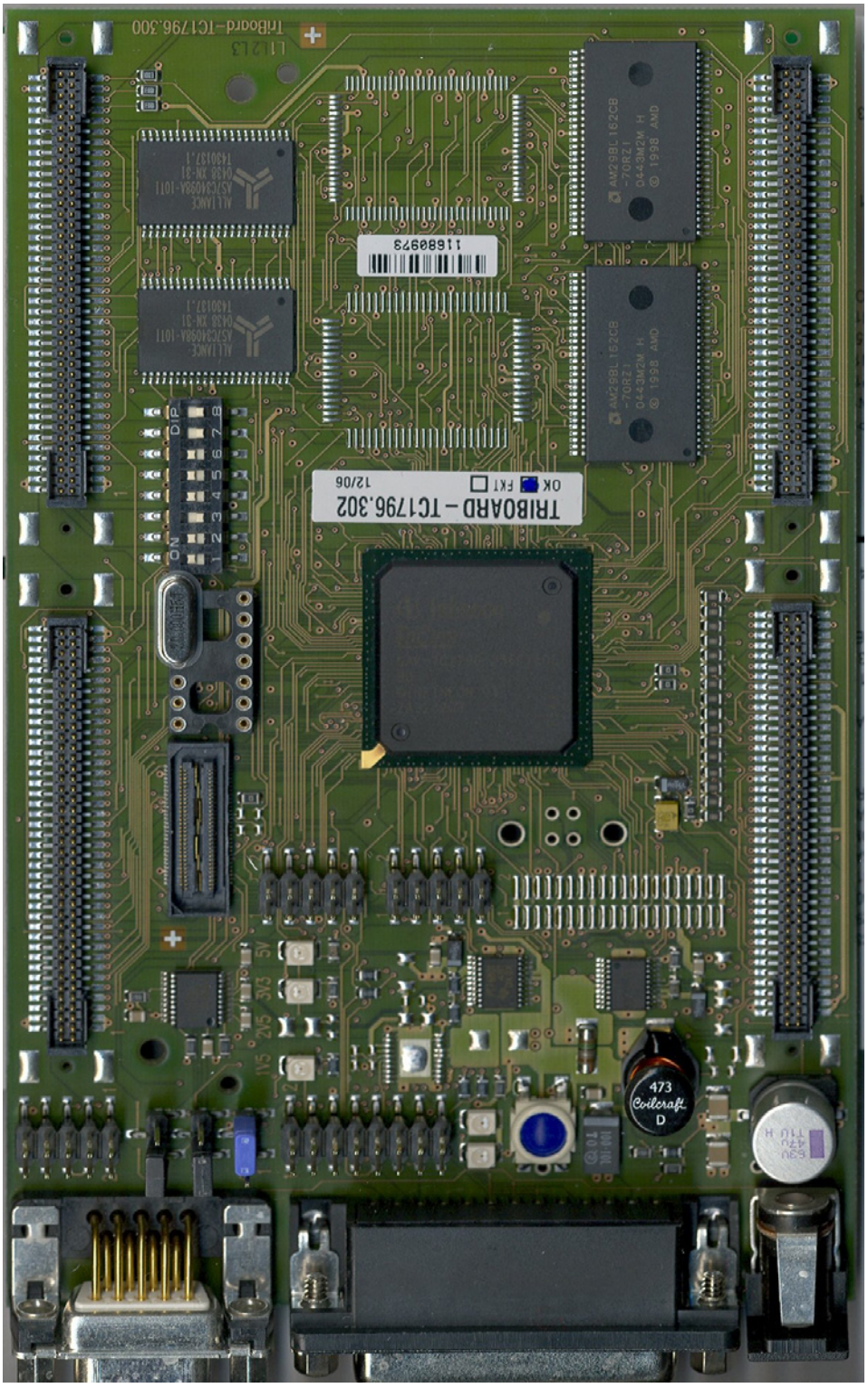




S301:
1, 2, 4, 5, 6 : ON
3, 7, 8 : OFF

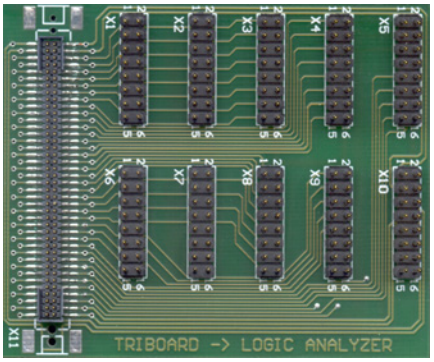
TC1796-Execution-Environment-SPRAM:

Jumper Settings (>= B-Step, HW-Configuration DIP-Switch):



Accessories for the TC1796 Starter Kit – Extension Boards

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

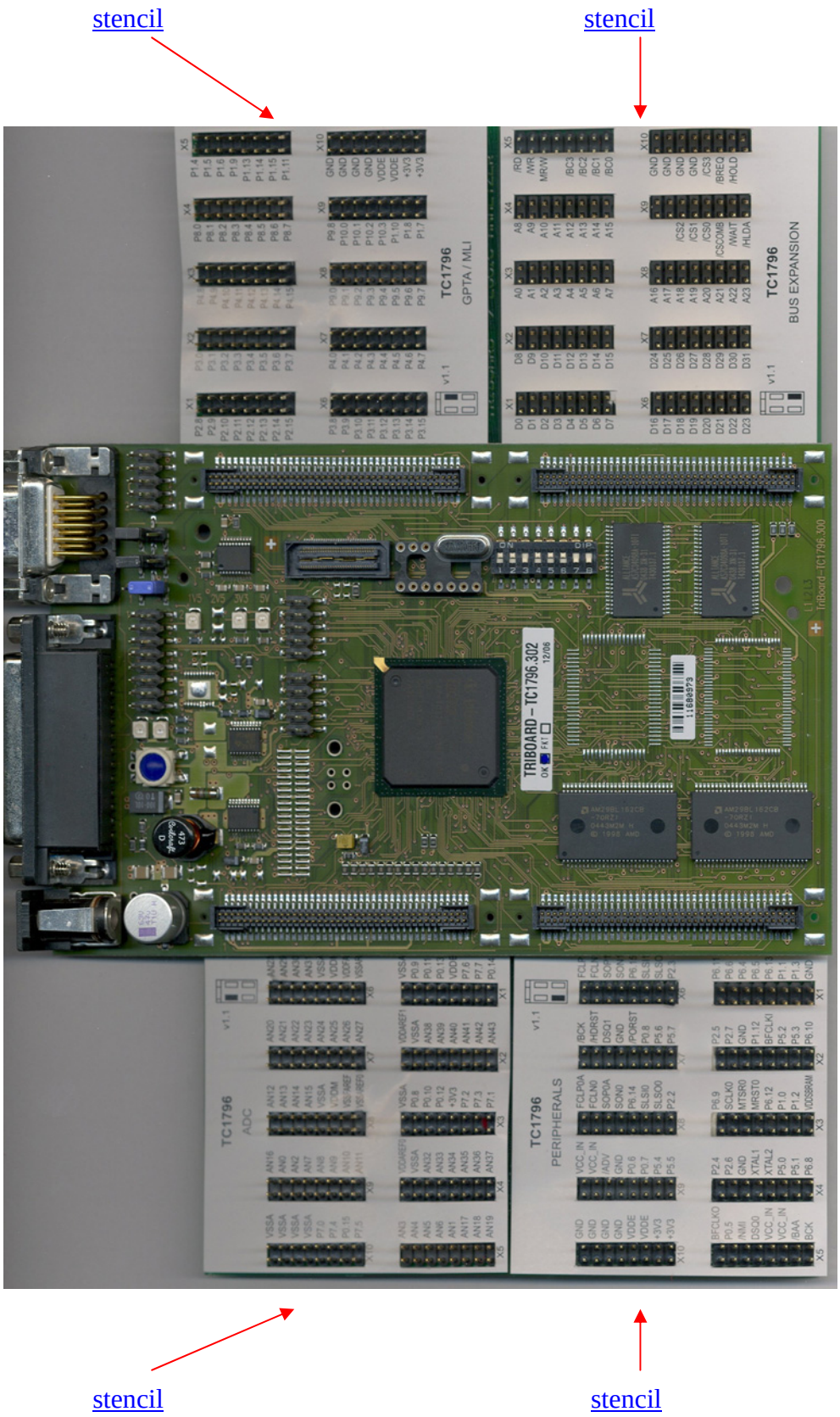


Ordering information:

TriBoard+XC16x-Adapter-Platine
Price: approximately € 32,- (4 required)

Extension Boards for easy measuring of the signals on the extension connectors.
You can order them at:

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


stencil
stencil
TC1796
GPTA / MLI
TC1796
BUS EXPANSION
TRIBOARD – TC1796.302
TC1796
ADC
TC1796
PERIPHERALS
stencil
stencil

2.) DAvE – Installation for TC1796 microcontrollers:

Install DAvE:

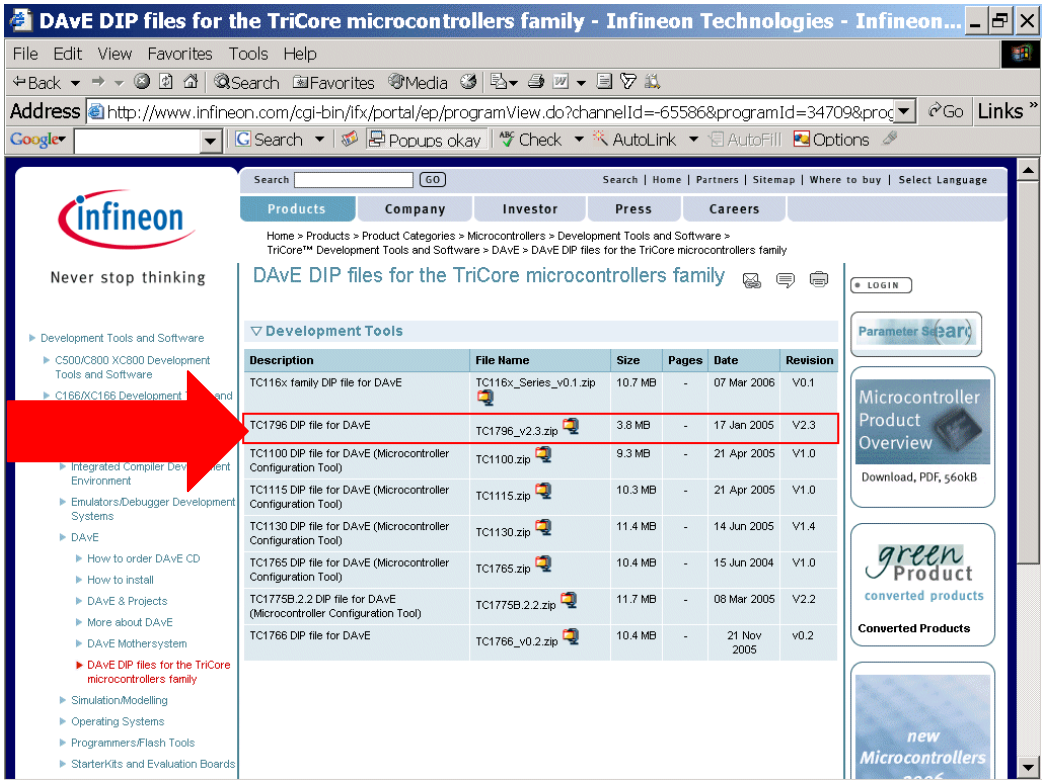
Download @ http://www.infineon.com/DAvE the DAvE-mothersystem setup.exe

Description	File Name	Size	Date	Revision
DAvE - Mothersystem without derivatives	setup.exe	15.2 MB	7 Jul 2005	

and execute setup.exe to install DAvE .

Install the TC1796 microcontroller Update:

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



Unzip the zip-file and save " TC1796.DIP "
@ e.g. D:\DAvE\TC1796-2006-03-16\TC1796_v2.3 .

2.)

Start DAvE - (click 

)

3.)

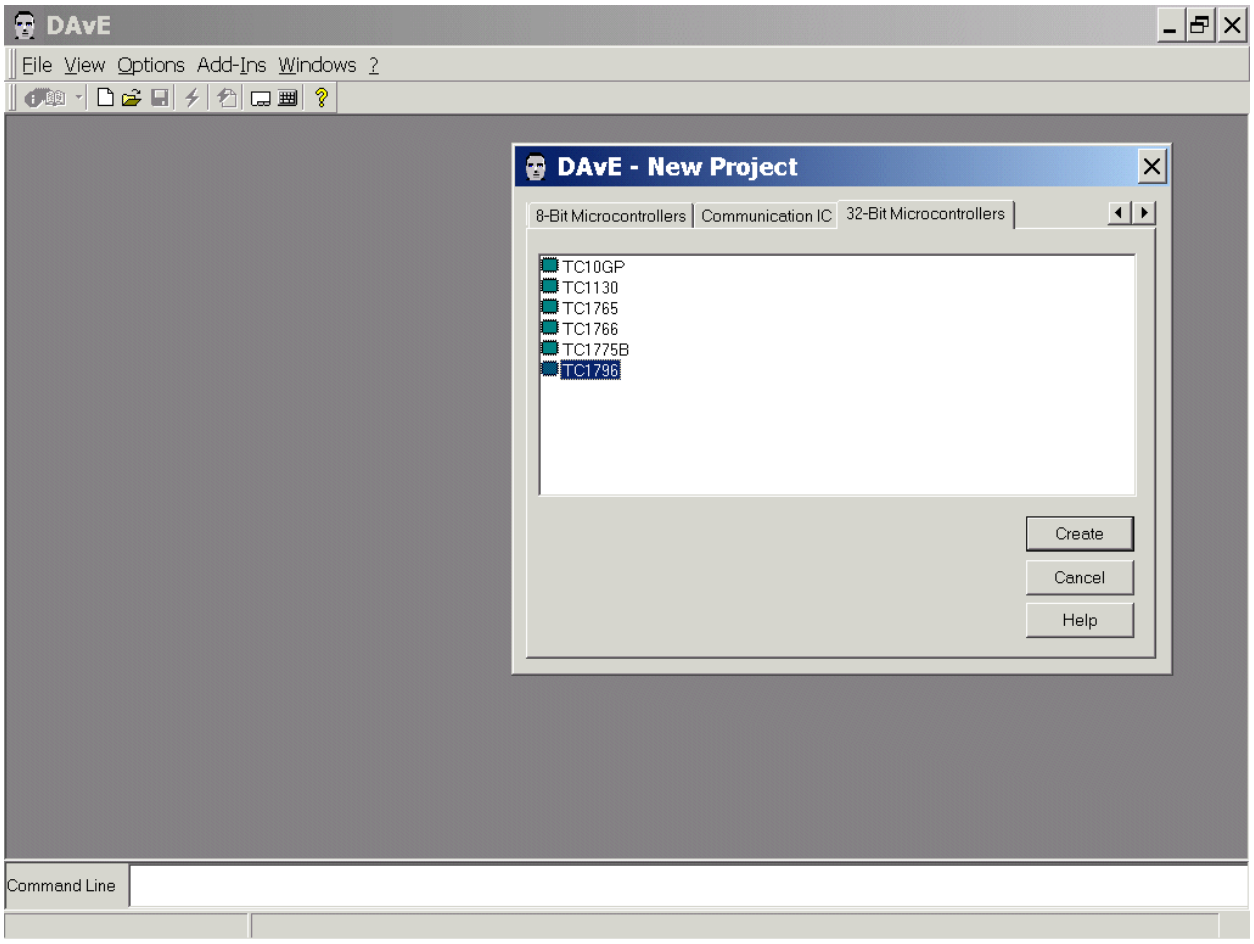
View
Setup Wizard
Default: • Installation
Forward>
Select: • I want to install products from the DAvE's web site
Forward>
Select: D:\DAvE\TC1796-2006-03-16\TC1796_v2.3
Forward>
Select: Available Products
click ✔ TC1796
Forward>
Install
End

4.) DAvE is now ready to generate code for the TC1796 microcontroller.

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

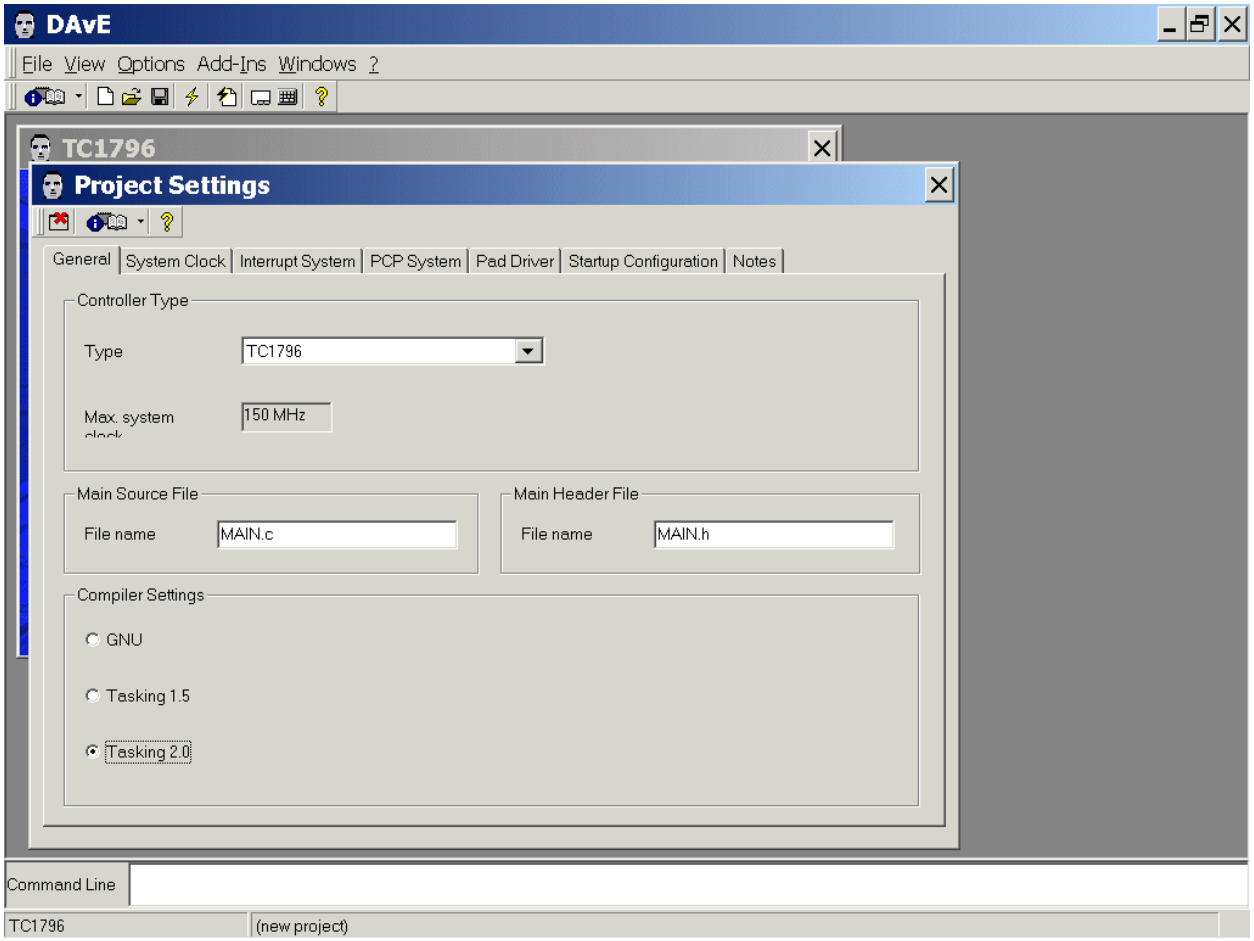
Start the program generator DAvE and select the TC1796 microcontroller:

File
New
32-Bit Microcontrollers
TC1796
Create



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

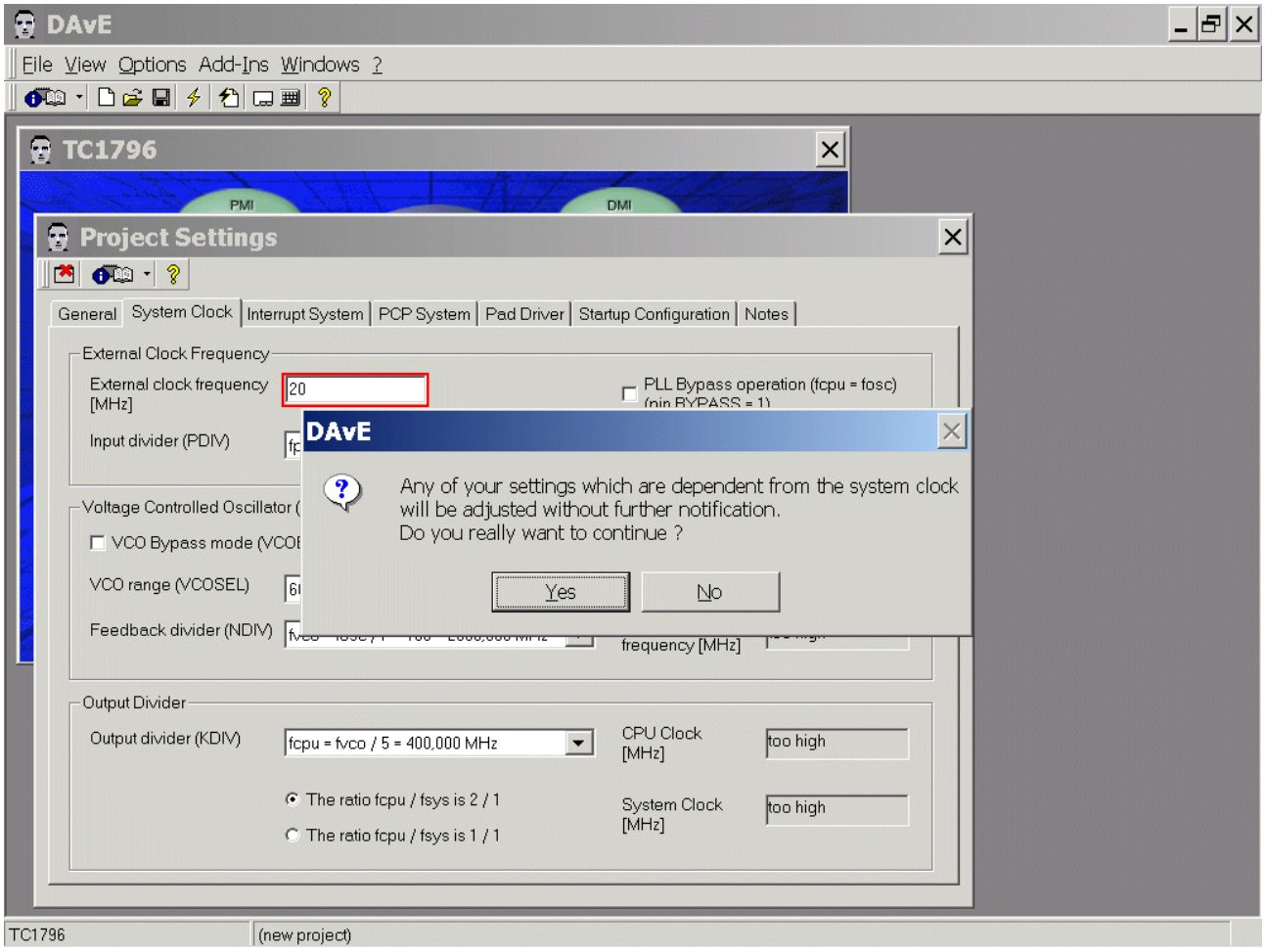
General: Compiler Settings:
For the Tasking Compiler choose Tasking 2.0 in the Compiler Settings:



System Clock: CPU Clock will be 150 MHz:

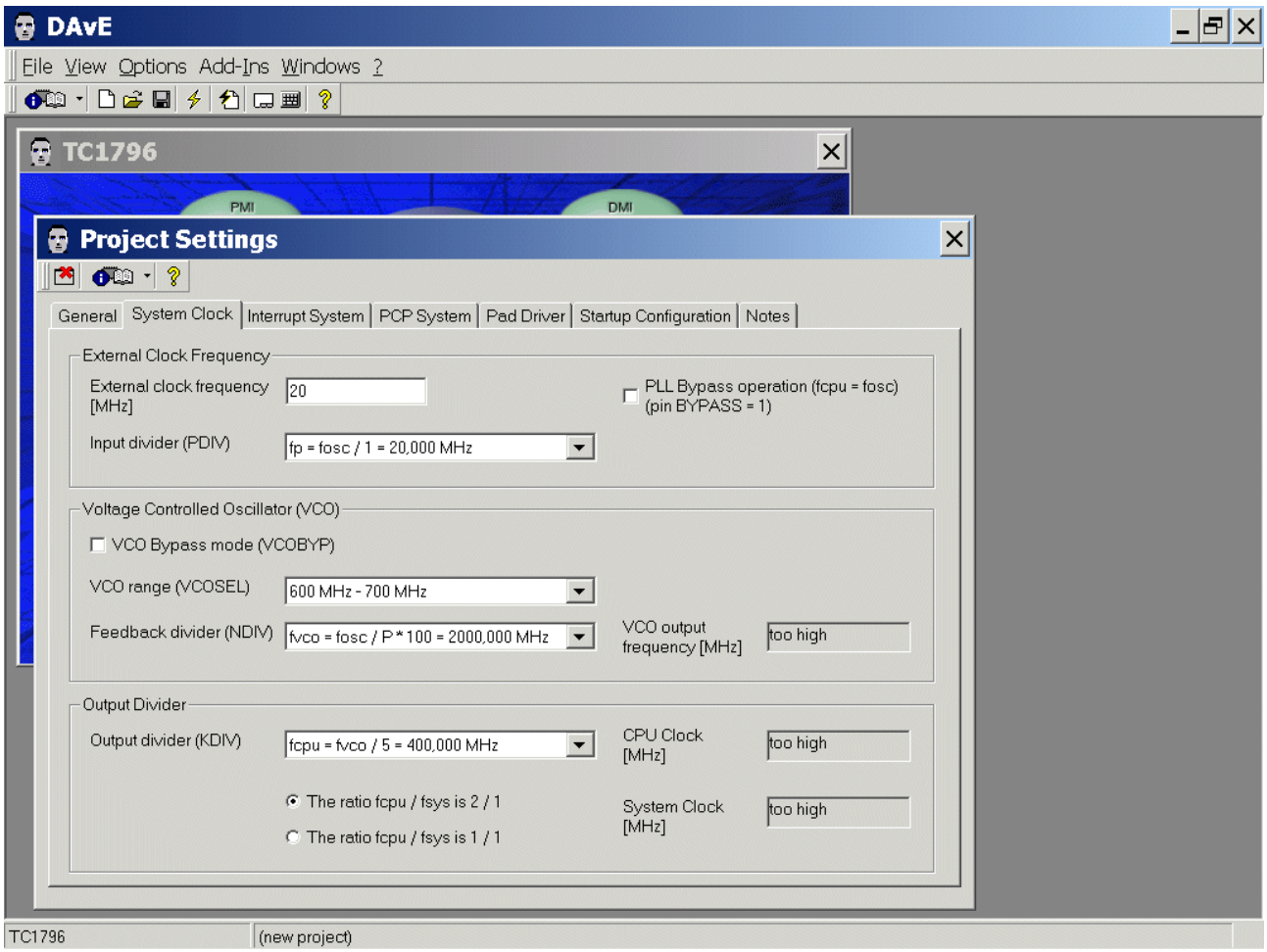
System Clock: External Clock Frequency: External clock frequency insert 20 [MHz]

Note:
We would be very grateful if you would check if your board is equipped with a 20 MHz Crystal (default).

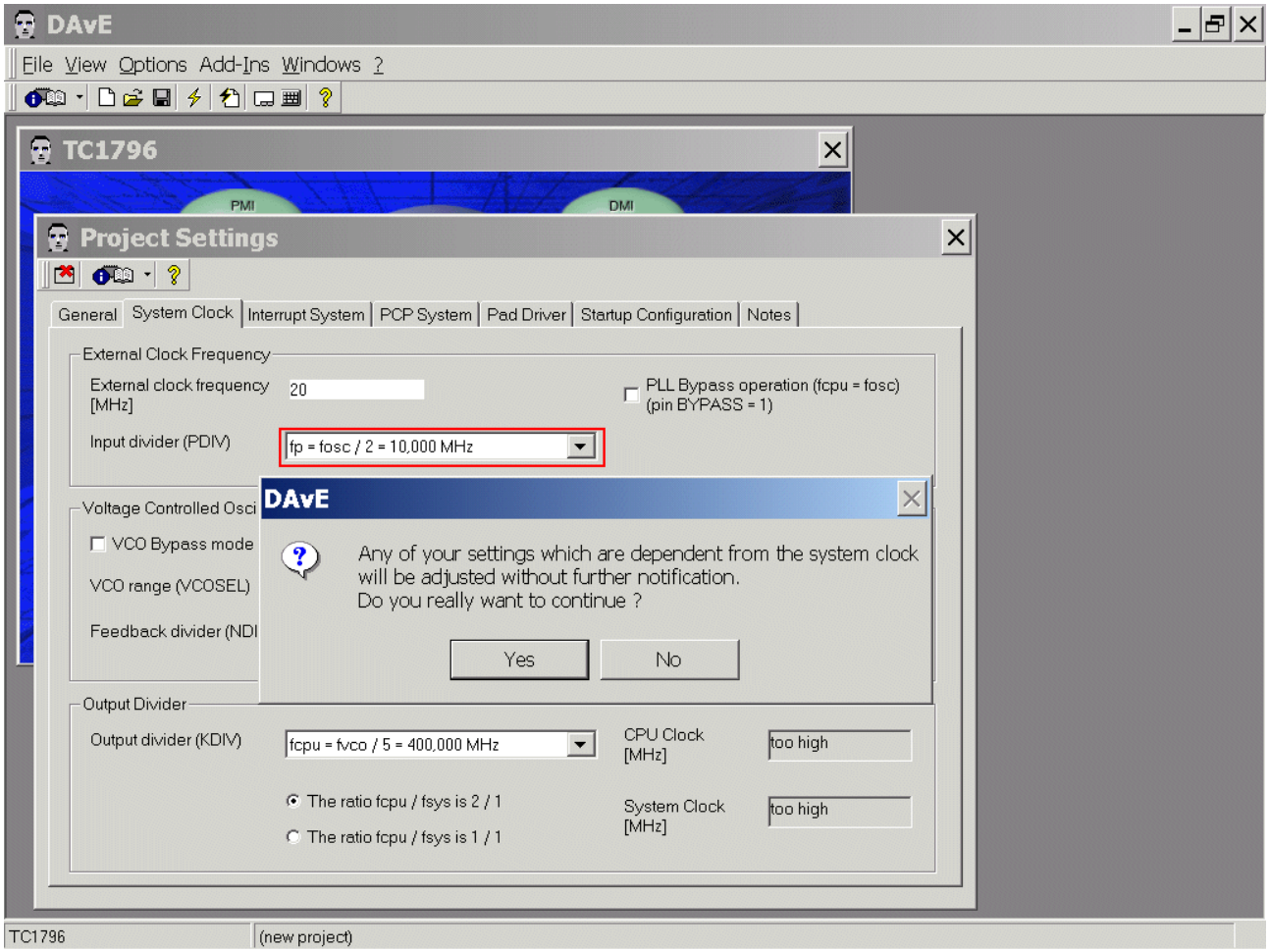


Yes

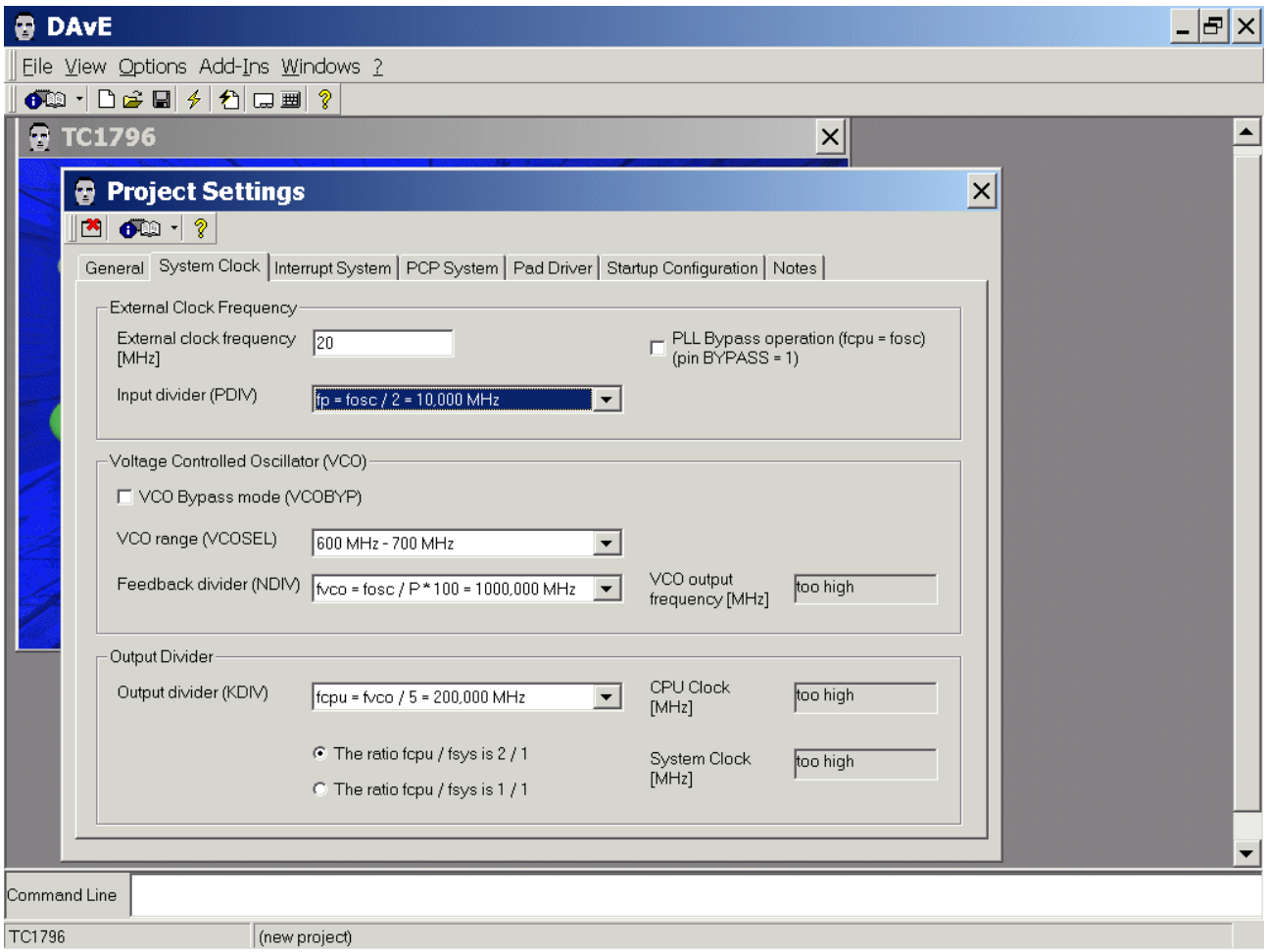
Note:
Validate each alpha numeric entry by pressing ENTER.


DAvE
File View Options Add-Ins Windows ?
TC1796
PMI
DMI
Project Settings
General
System Clock
Interrupt System
PCP System
Pad Driver
Startup Configuration
Notes
External Clock Frequency
External clock frequency [MHz]
20
PLL Bypass operation (fcpu = fosc) (pin BYPASS = 1)
Input divider (PDIV)
fp = fosc / 1 = 20,000 MHz
Voltage Controlled Oscillator (VCO)
VCO Bypass mode (VCOBYP)
VCO range (VCOSEL)
600 MHz - 700 MHz
Feedback divider (NDIV)
fvco = fosc / P * 100 = 2000,000 MHz
VCO output frequency [MHz]
too high
Output Divider
Output divider (KDIV)
fcpu = fvco / 5 = 400,000 MHz
CPU Clock [MHz]
too high
The ratio fcpu / fsys is 2 / 1
The ratio fcpu / fsys is 1 / 1
System Clock [MHz]
too high
TC1796
(new project)

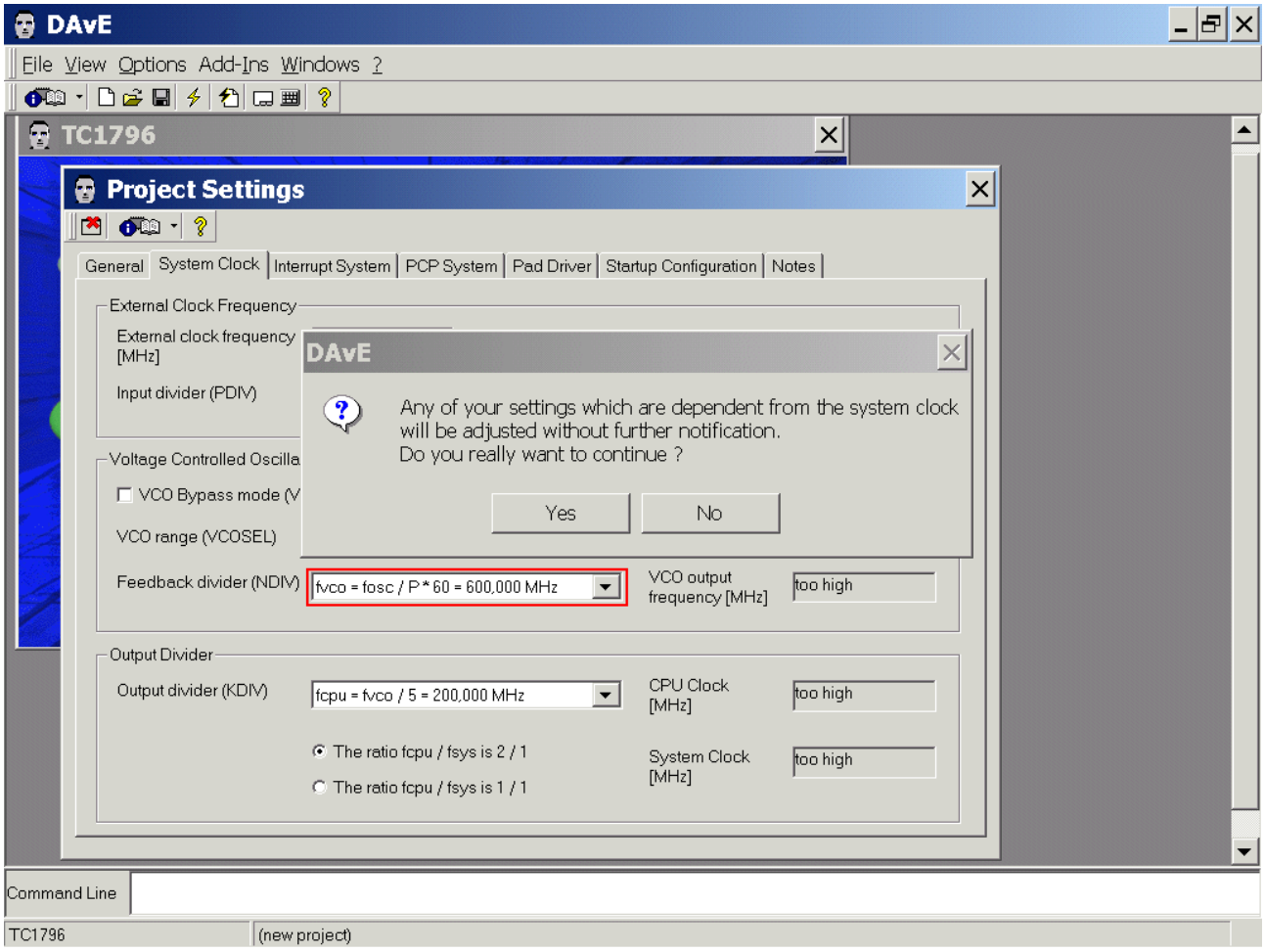
System Clock: External Clock Frequency: Input divider (PDIV) select fp=fosc/2=10,000 MHz



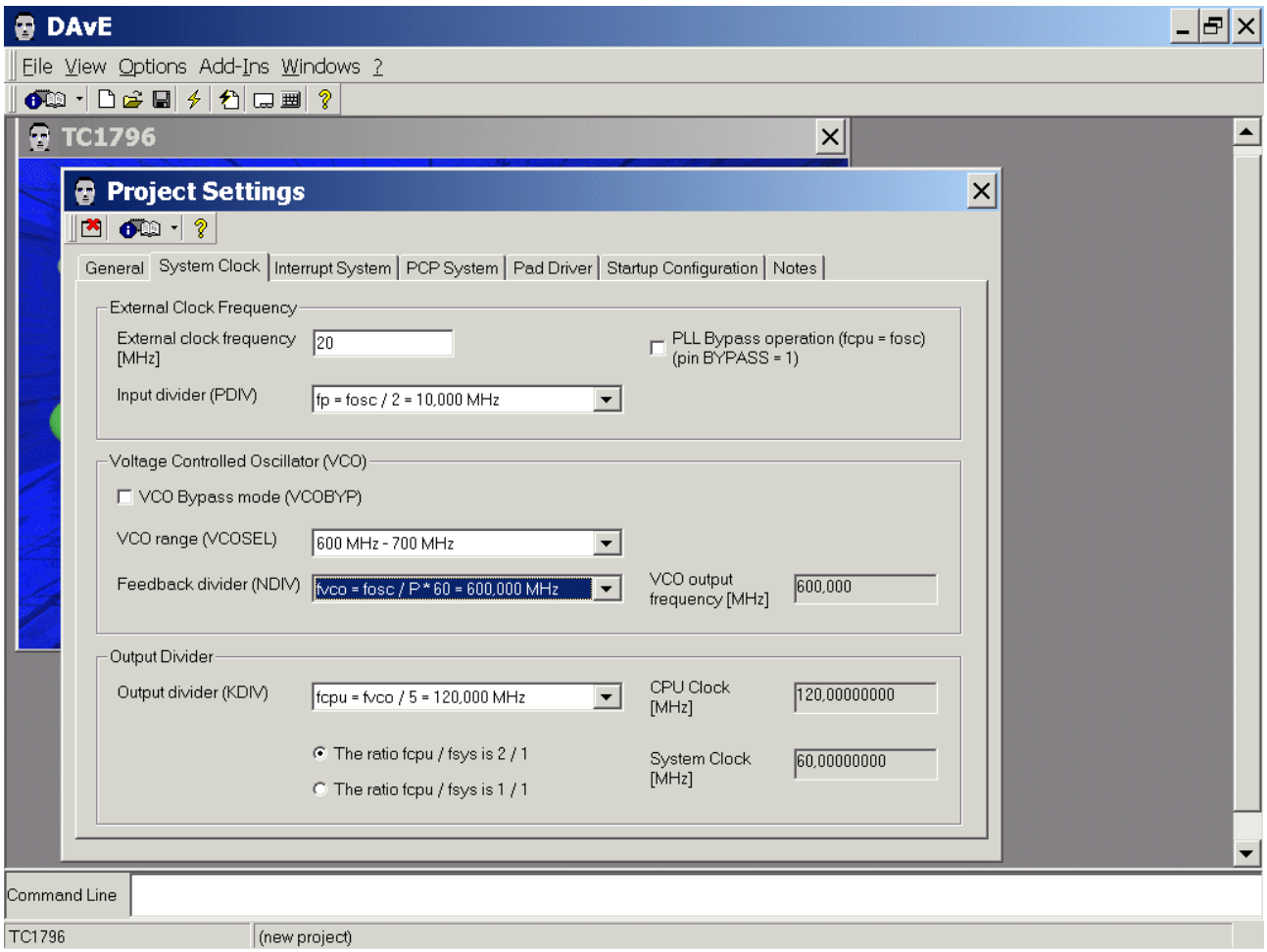
Yes


DAvE
File View Options Add-Ins Windows ?
TC1796
Project Settings
General
System Clock
Interrupt System
PCP System
Pad Driver
Startup Configuration
Notes
External Clock Frequency
External clock frequency [MHz]
20
PLL Bypass operation (fcpu = fosc) (pin BYPASS = 1)
Input divider (PDIV)
fp = fosc / 2 = 10,000 MHz
Voltage Controlled Oscillator (VCO)
VCO Bypass mode (VCOBYP)
VCO range (VCOSEL)
600 MHz - 700 MHz
Feedback divider (NDIV)
fvco = fosc / P * 100 = 1000,000 MHz
VCO output frequency [MHz]
too high
Output Divider
Output divider (KDIV)
fcpu = fvco / 5 = 200,000 MHz
CPU Clock [MHz]
too high
The ratio fcpu / fsys is 2 / 1
The ratio fcpu / fsys is 1 / 1
System Clock [MHz]
too high
Command Line
TC1796
(new project)

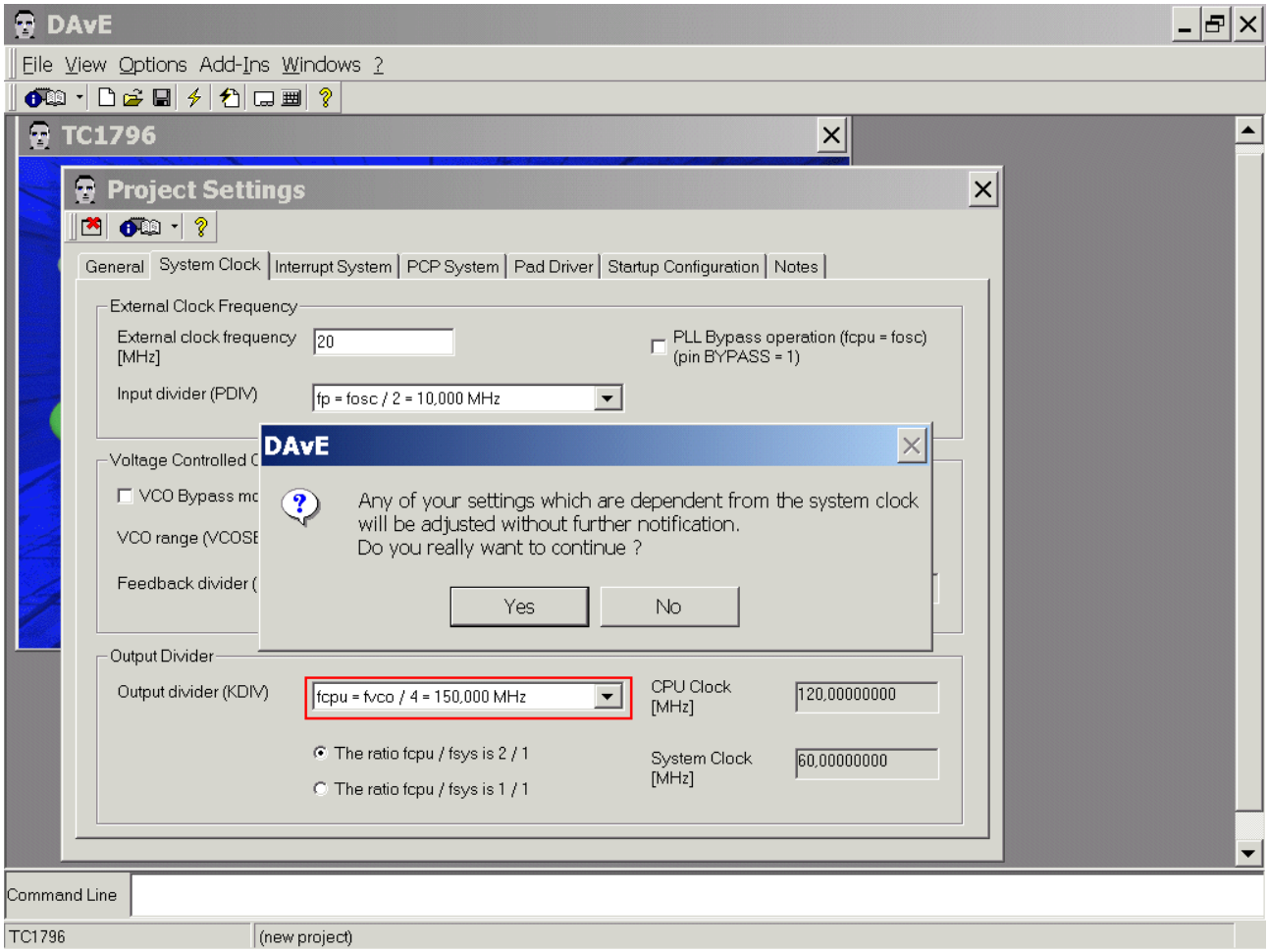
System Clock: Voltage Controlled Oscillator:
Feedback divider (NDIV) select fvco=fosc/P*60=600,000 MHz



Yes

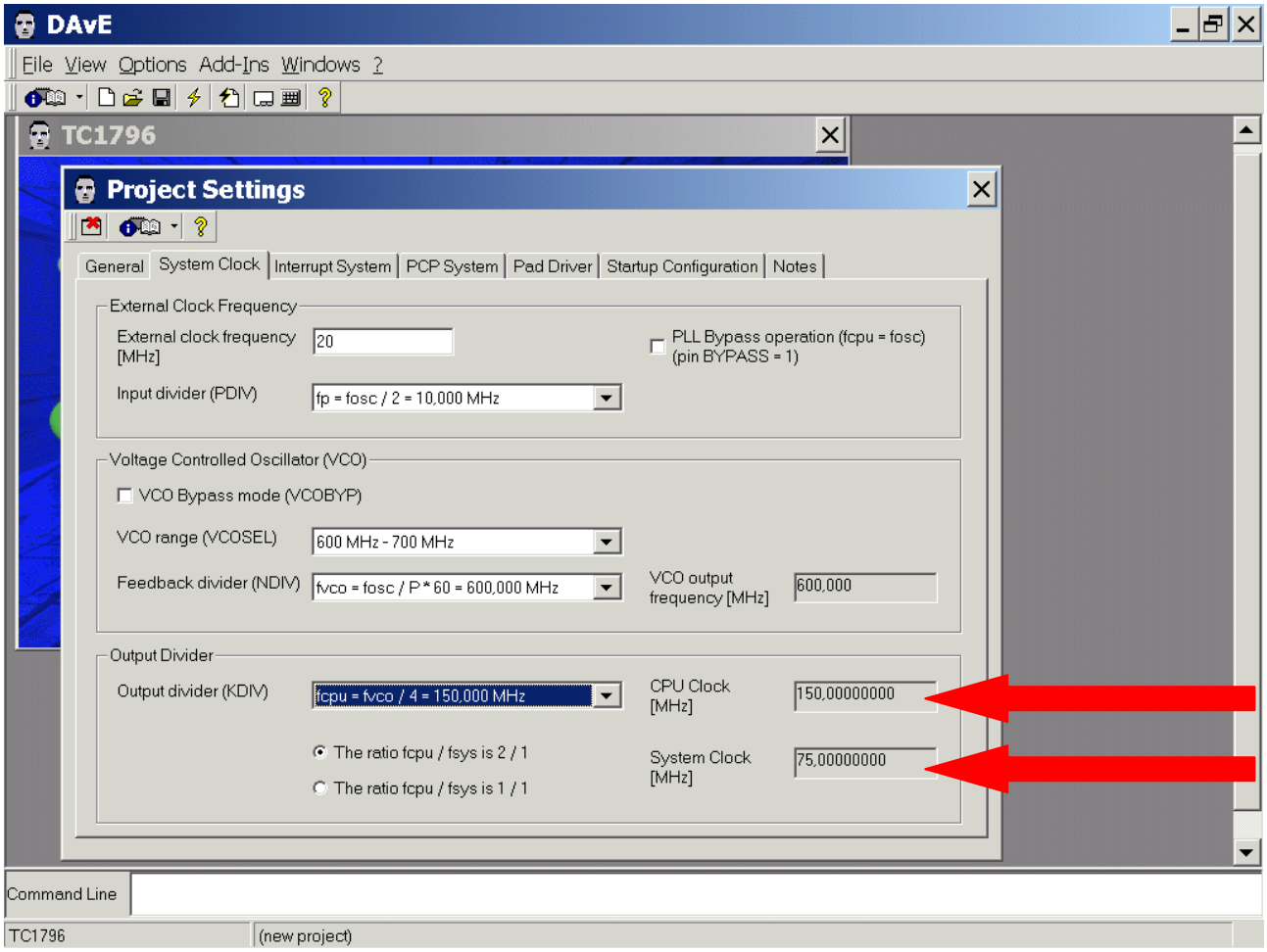

DAvE
File View Options Add-Ins Windows ?
TC1796
Project Settings
General
System Clock
Interrupt System
PCP System
Pad Driver
Startup Configuration
Notes
External Clock Frequency
External clock frequency [MHz]
20
PLL Bypass operation (fcpu = fosc) (pin BYPASS = 1)
Input divider (PDIV)
fp = fosc / 2 = 10,000 MHz
Voltage Controlled Oscillator (VCO)
VCO Bypass mode (VCOBYP)
VCO range (VCOSEL)
600 MHz - 700 MHz
Feedback divider (NDIV)
fvco = fosc / P * 60 = 600,000 MHz
VCO output frequency [MHz]
600,000
Output Divider
Output divider (KDIV)
fcpu = fvco / 5 = 120,000 MHz
CPU Clock [MHz]
120,00000000
The ratio fcpu / fsys is 2 / 1
The ratio fcpu / fsys is 1 / 1
System Clock [MHz]
60,00000000
Command Line
TC1796
(new project)

System Clock: Output Divider: Output divider (KDIV) select fcpu=fvco/4=150,000 MHz

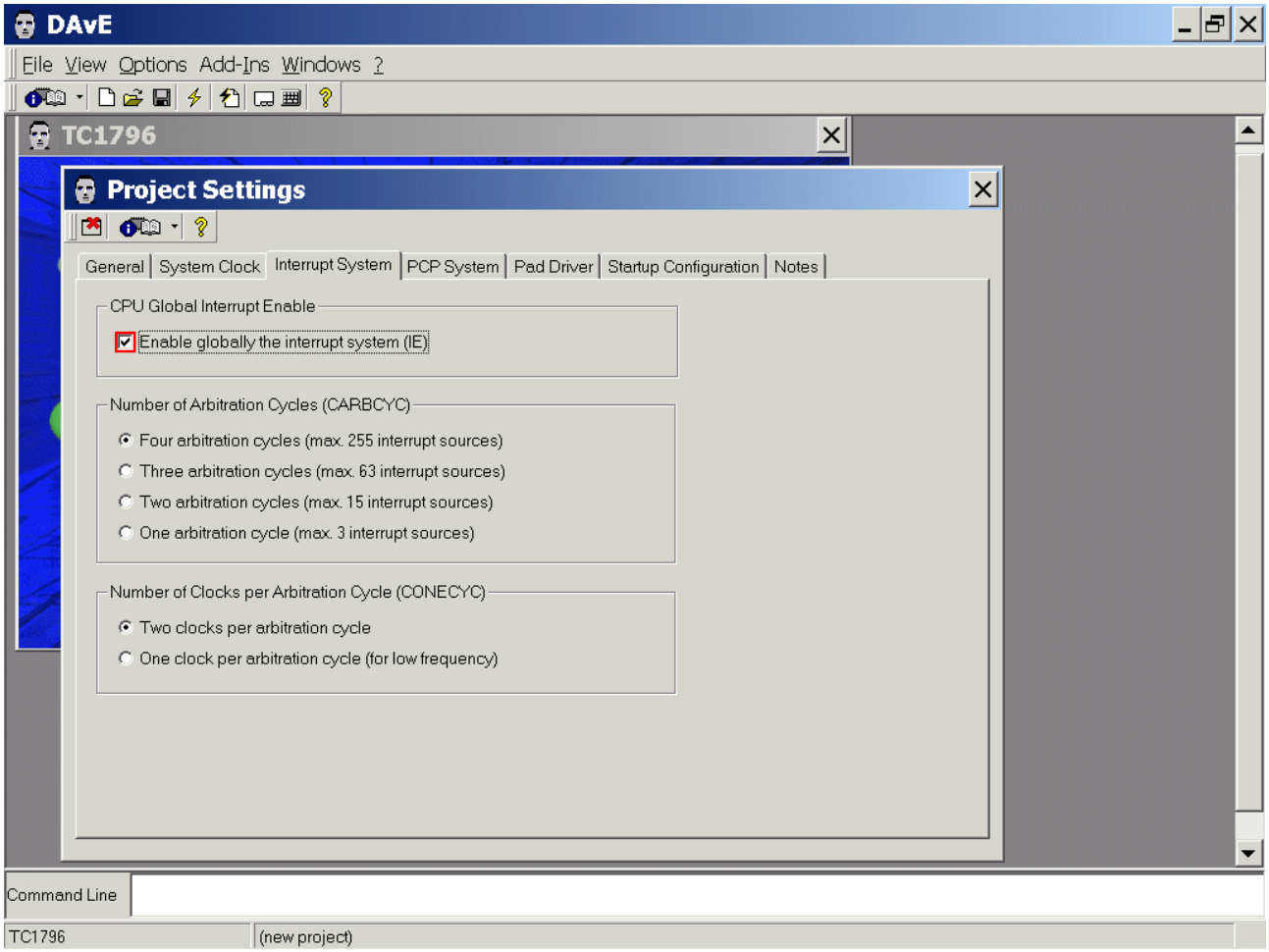


Yes

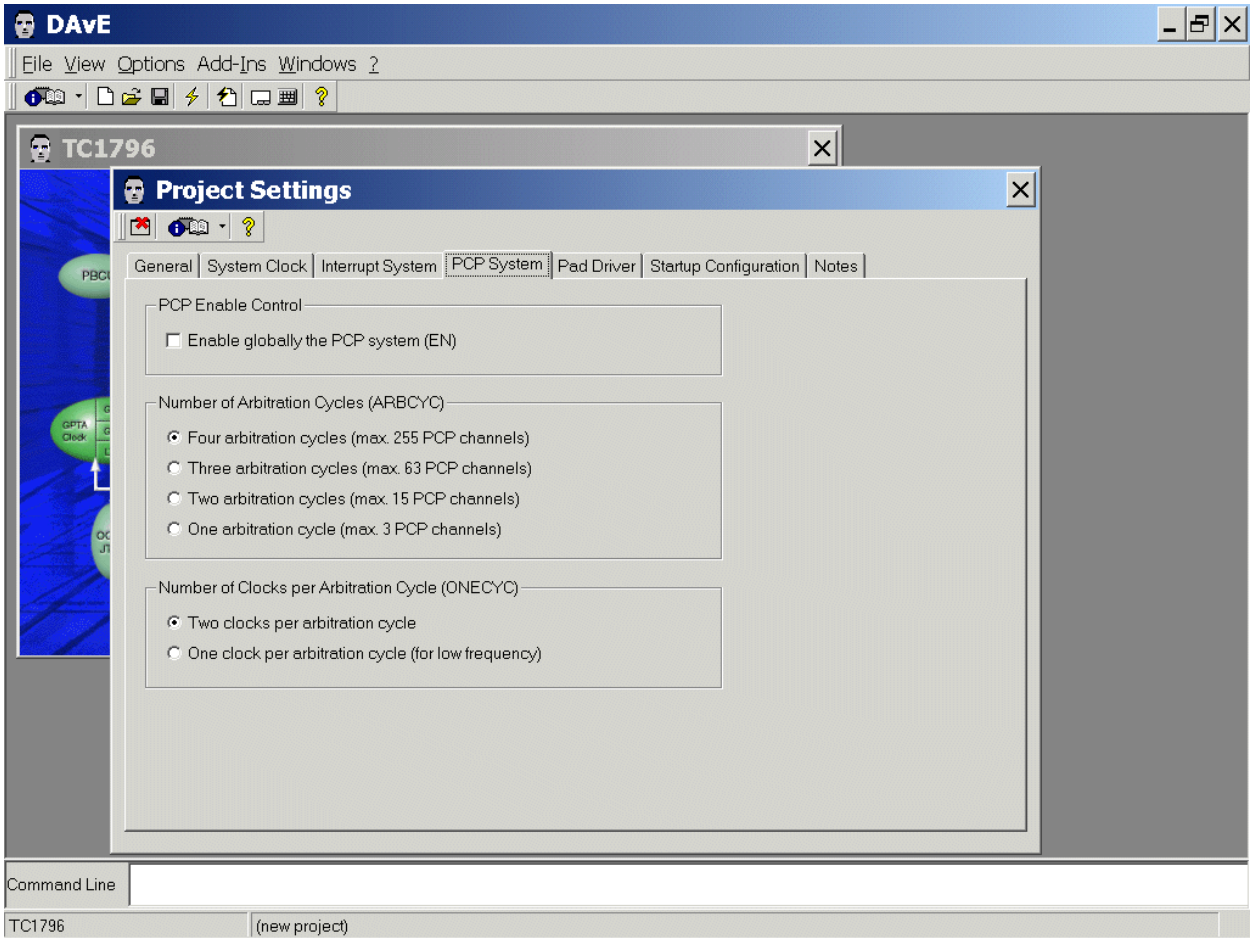
Note:
The final result should be 150 MHz CPU frequency and 75 MHz system frequency.



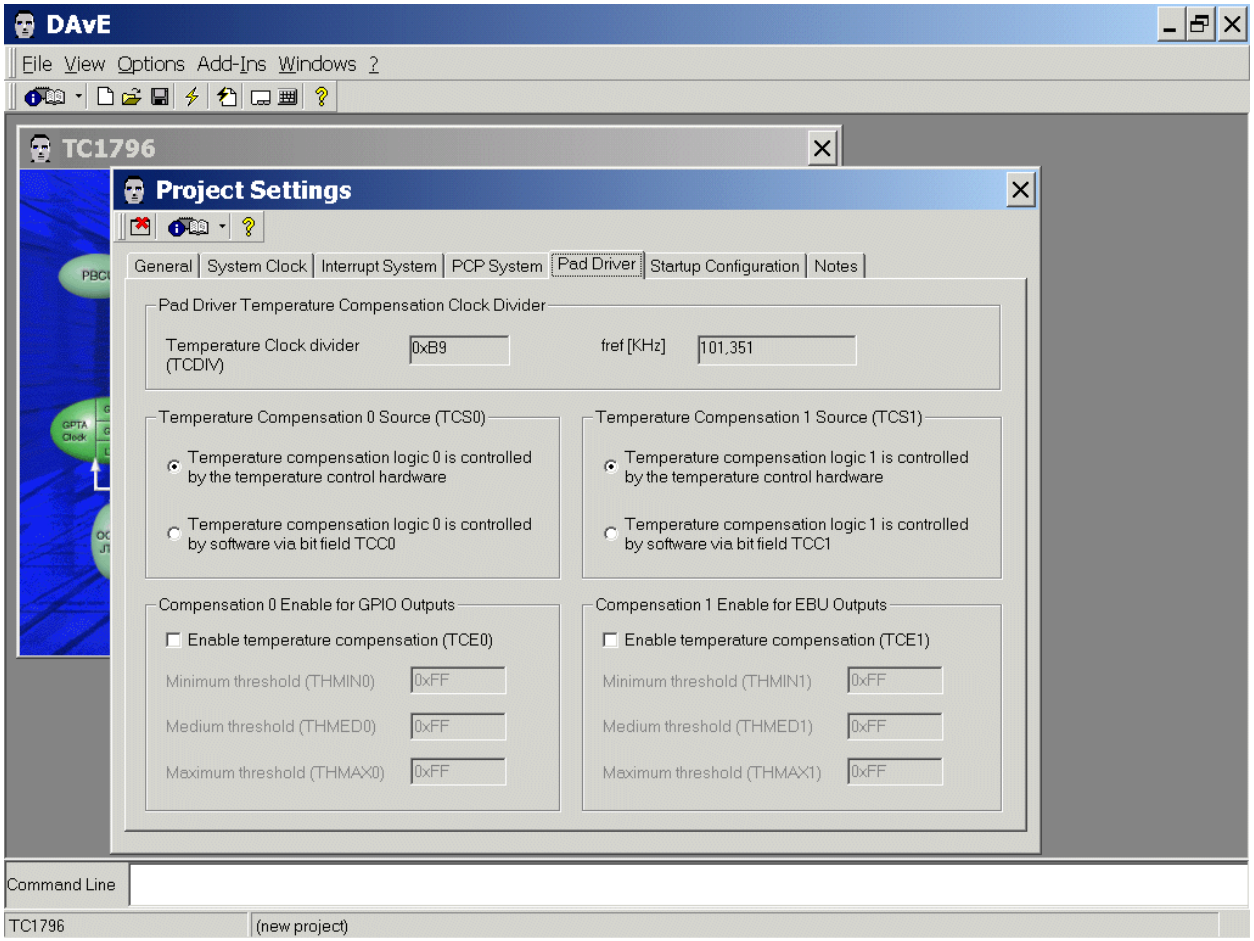
Interrupt System: CPU Global Interrupt Enable: click ✓Enable globally the interrupt system (IE)



PCP System: (do nothing)



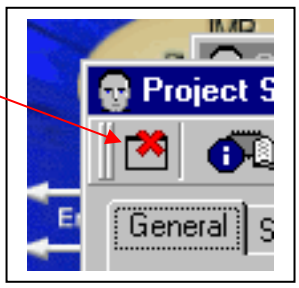
Pad Driver: (do nothing)



Startup Configuration: Hardware Booting Scheme: (do not care !!!)

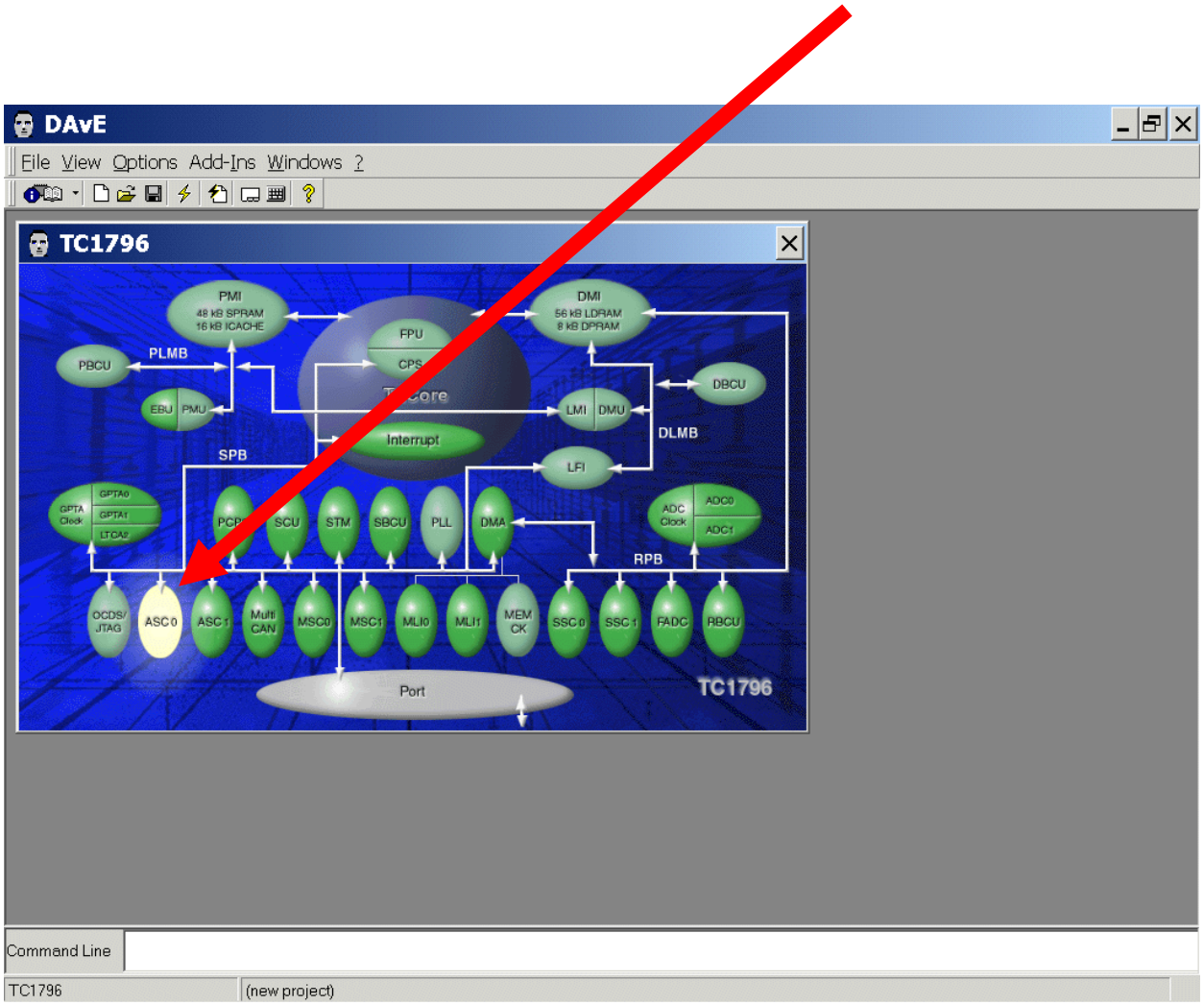
Notes: If you wish, you can insert your comments here.

Exit this dialog now by clicking  the close button:

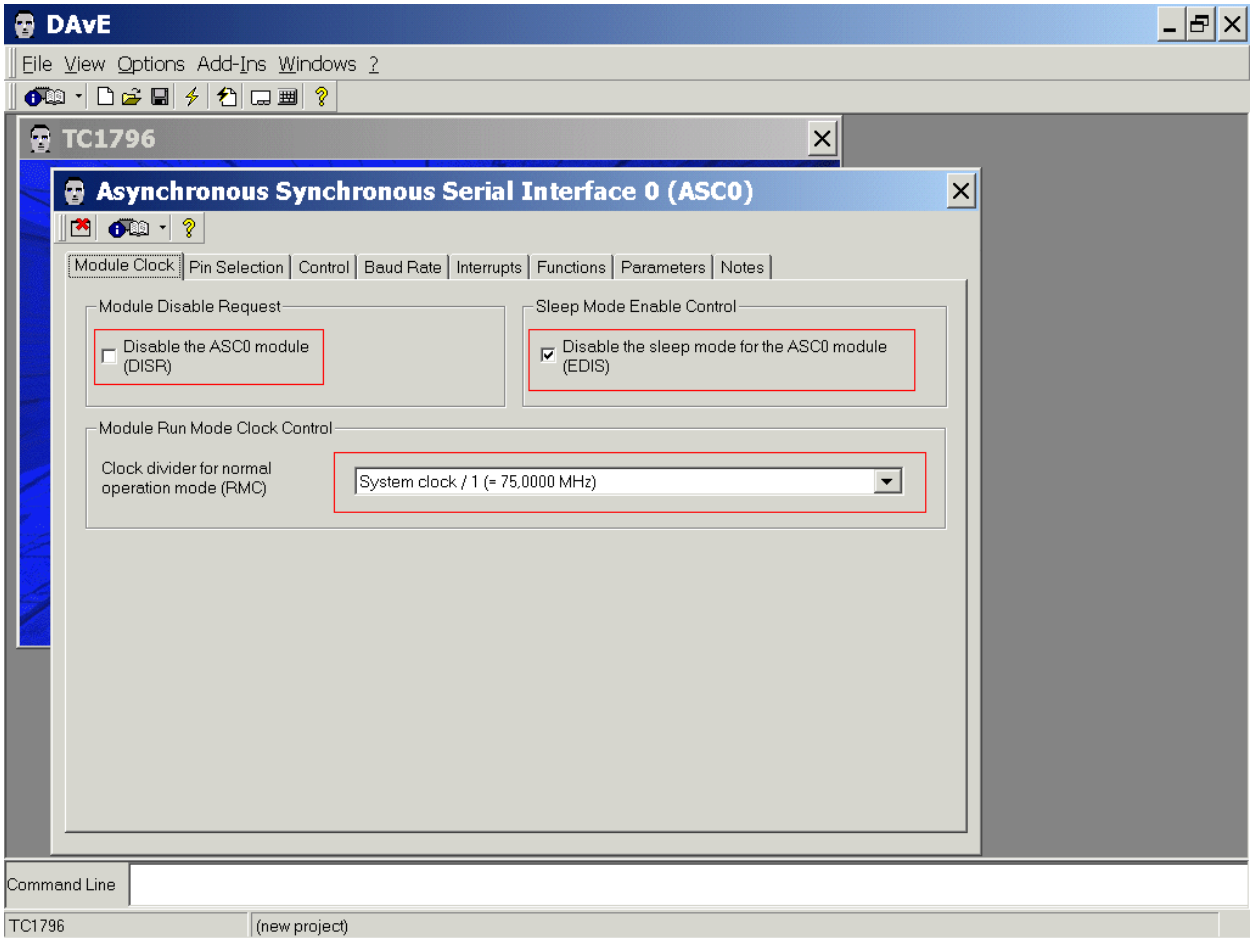


Configuration of the ASC0:

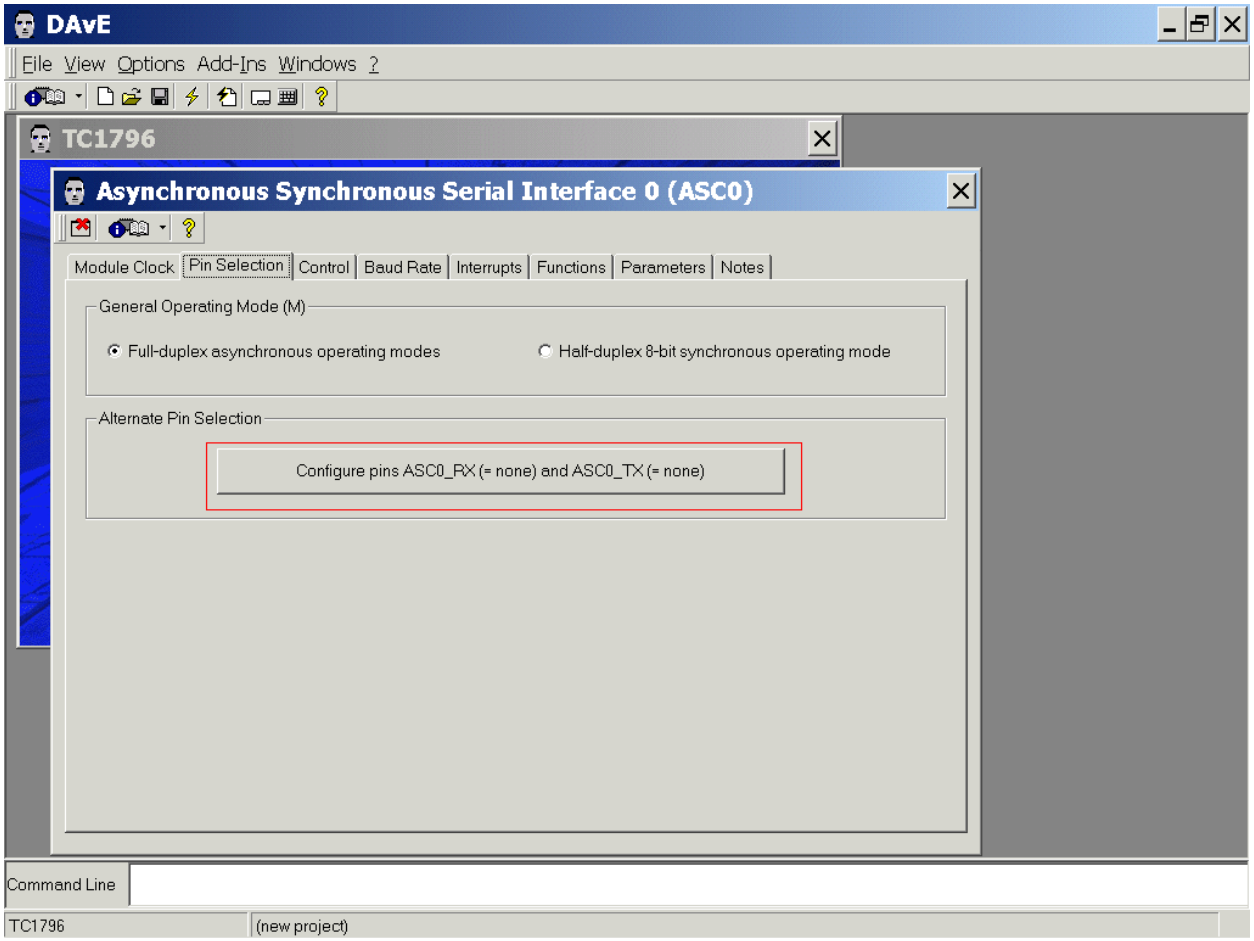
The configuration window can be opened by clicking the specific block/module.



Module Clock: Module Disable Request: select/check ☐ Disable the ASC0 module
Module Clock: Module Run Mode Clock Control: choose System clock/1 (=75,0000 MHz)
Module Clock: Sleep Mode Enable Control: click ✓ Disable the sleep mode

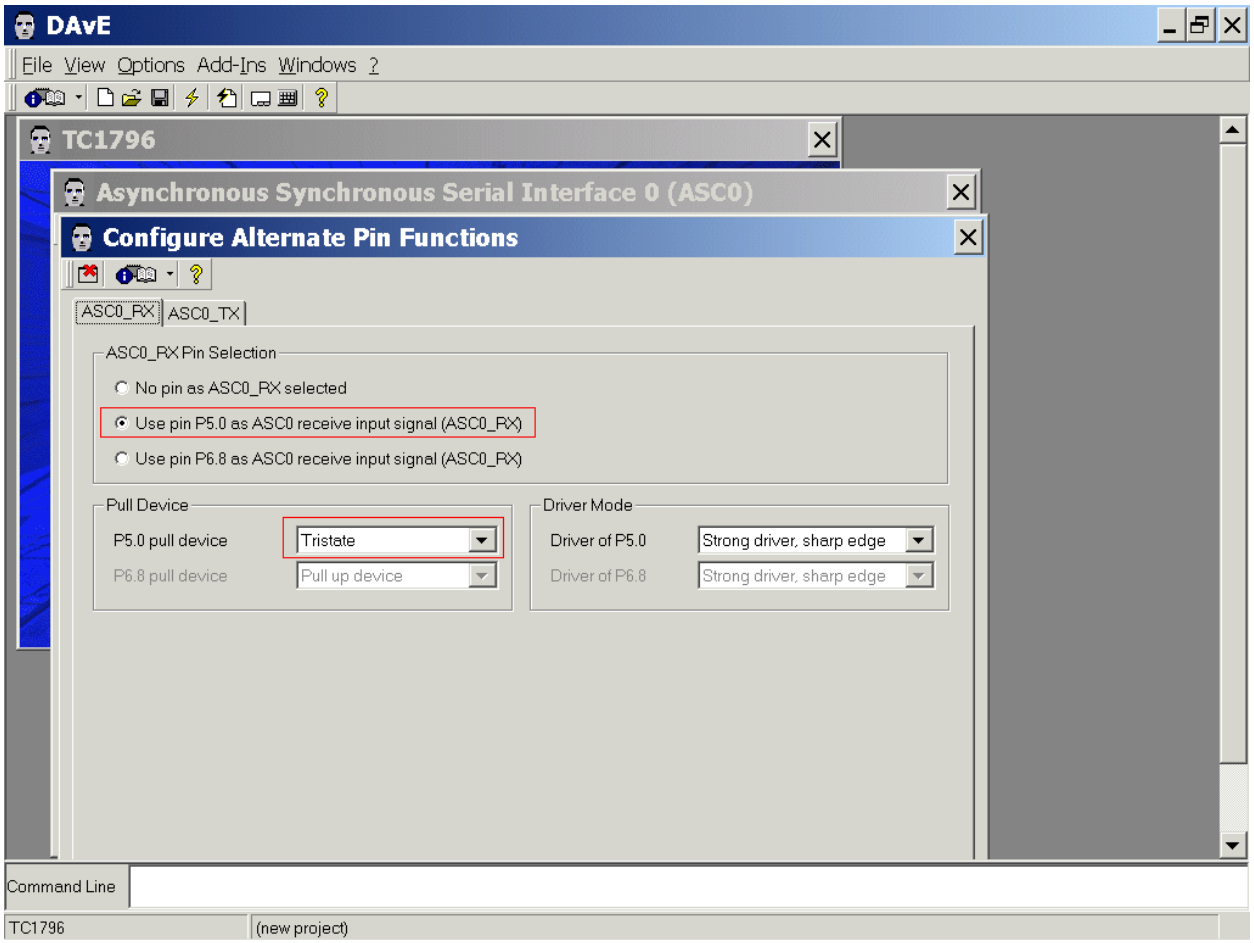


Pin Selection: Alternate Pin Selection: click Configure pins RX and TX

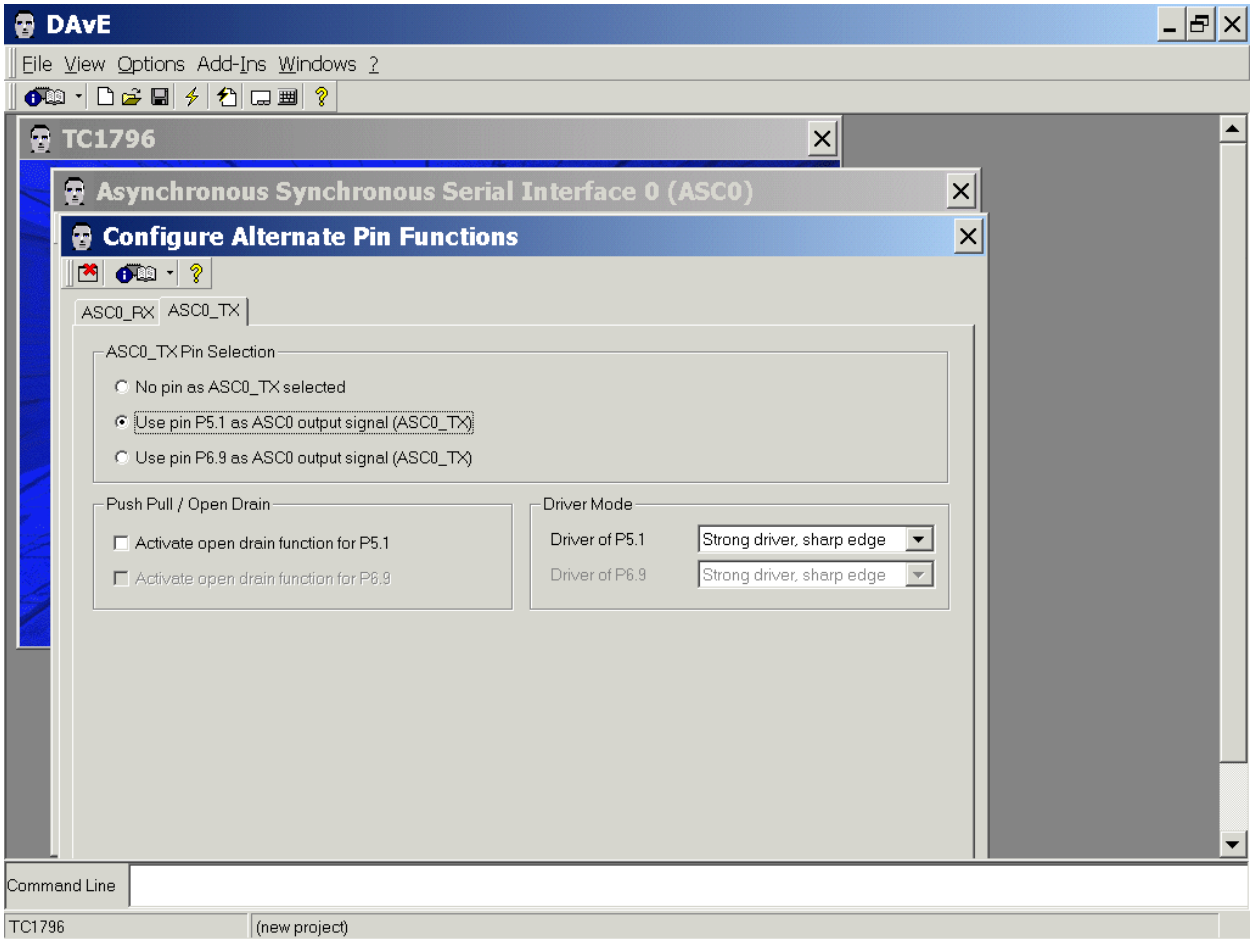


Pin Selection: Alternate Pin Selection: Configure pins RX and TX
ASC0_RX: ASC0_RX Pin Selection: select Use pin P5.0

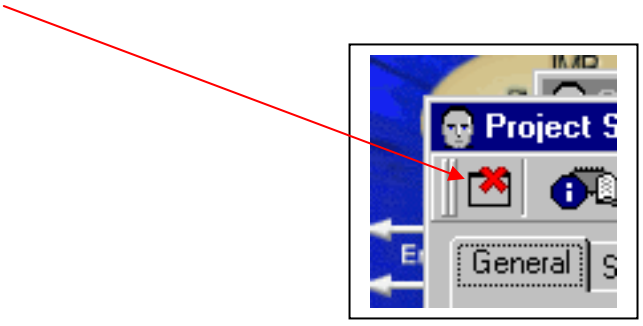
Pin Selection: Alternate Pin Selection: Configure pins RX and TX
ASC0_RX: Pull Device: select Tristate



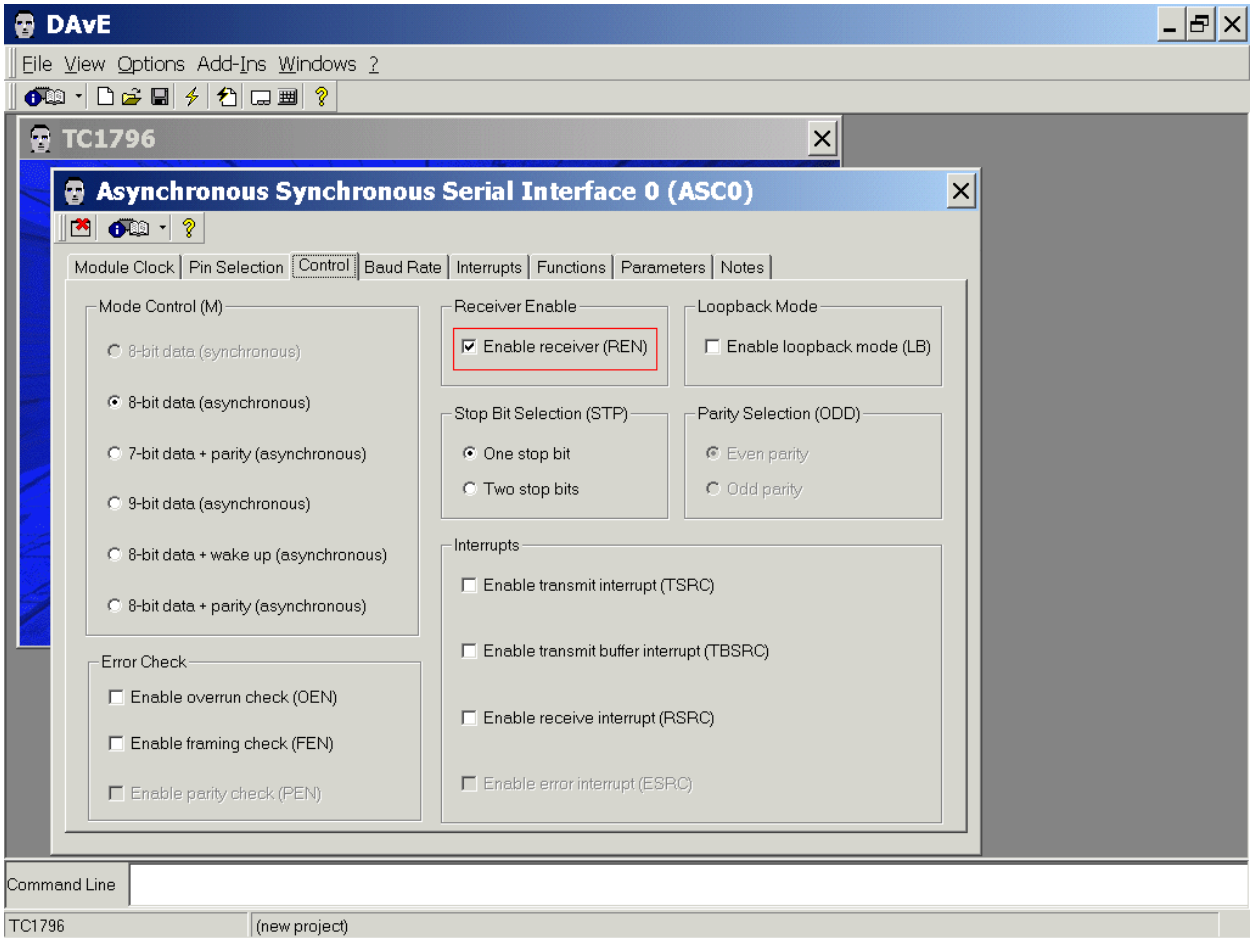
Pin Selection: Alternate Pin Selection: Configure pins RX and TX
ASC0_TX: ASC0_TX Pin Selection: select Use pin P5.1



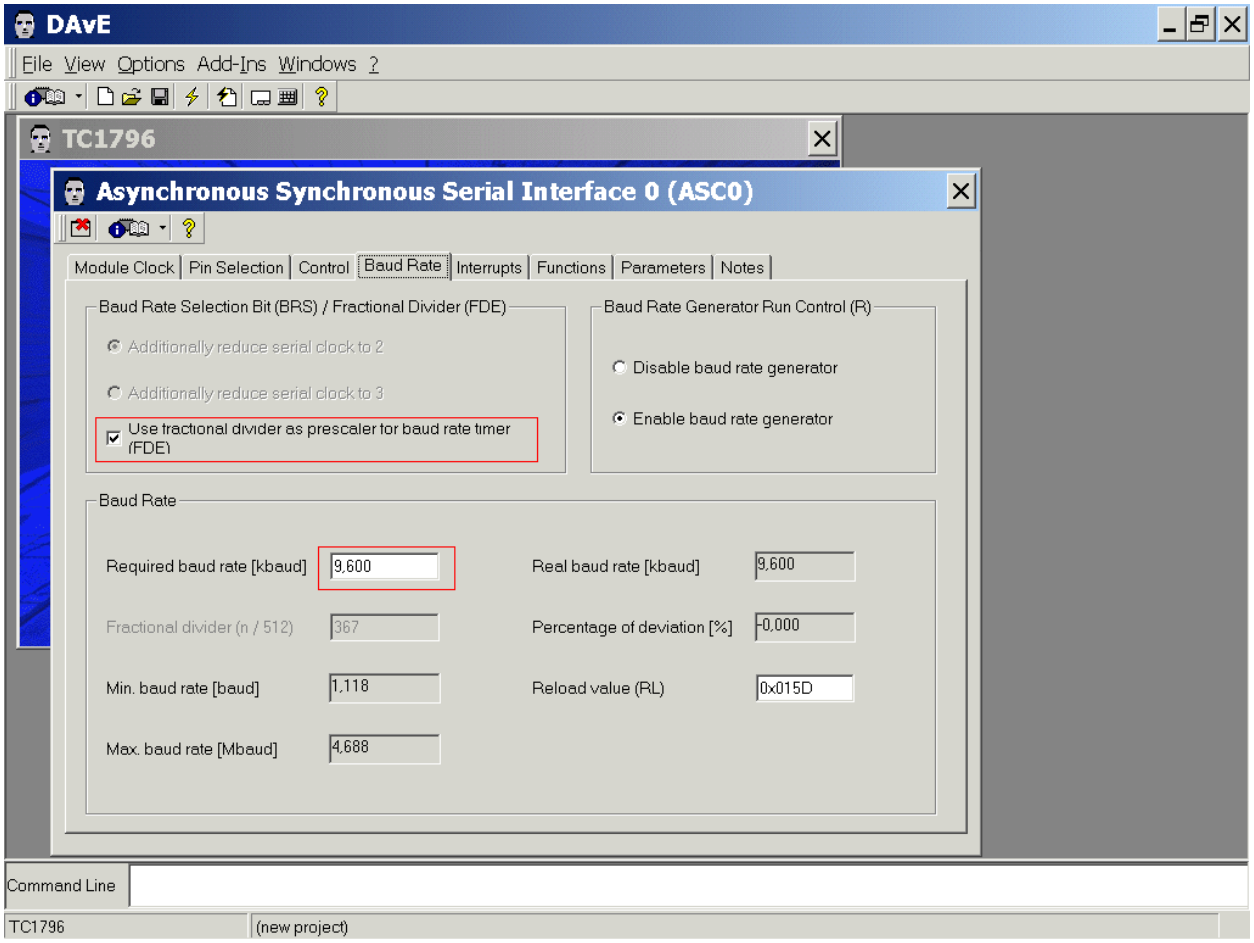
Exit this dialog now by clicking X in the close button:



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

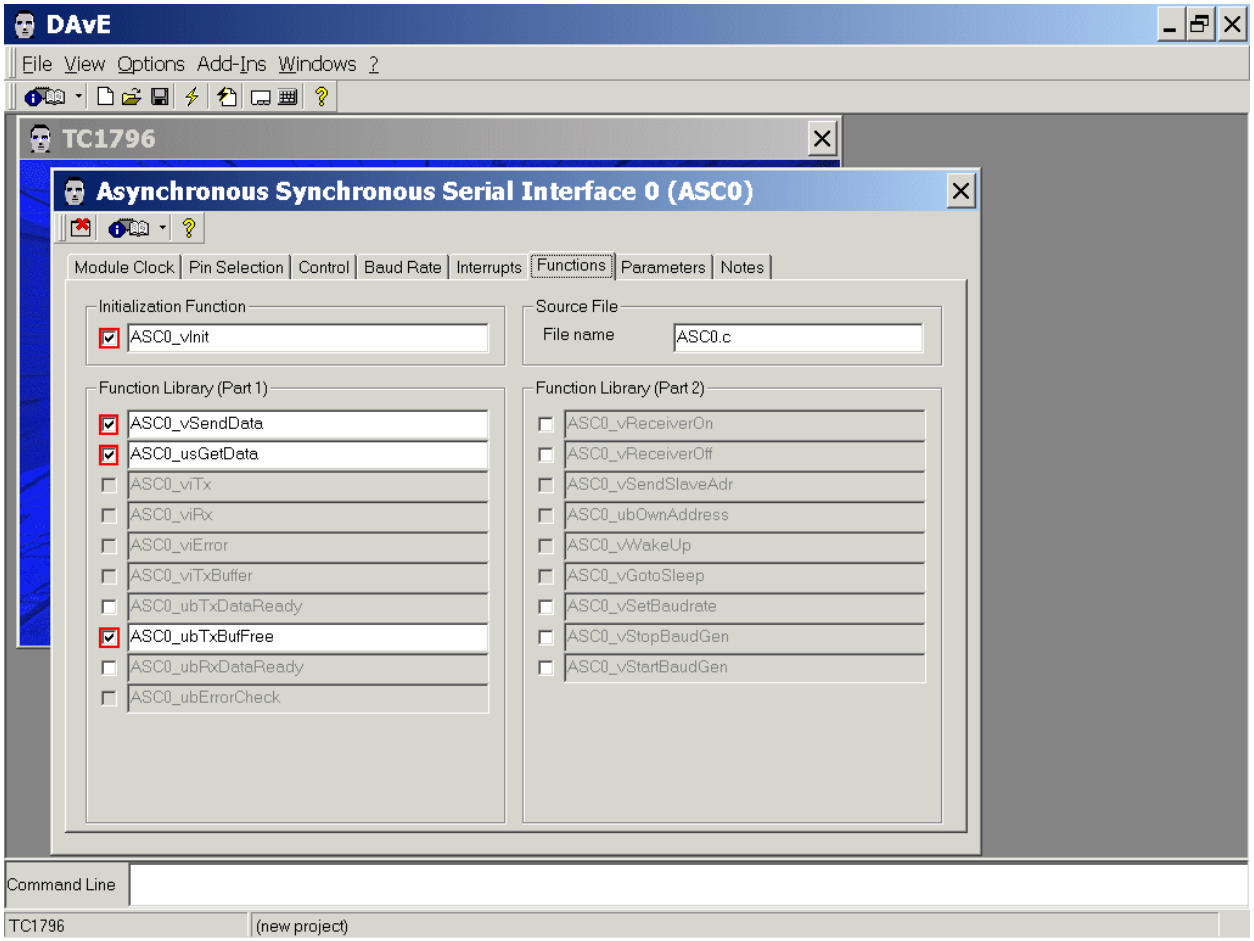


Baud Rate: Baud Rate: Required baud rate [kBaud] insert 9,600
Baud Rate: Baud Rate Selection Bit / Fractional Divider: click ✓ Use fractional divider

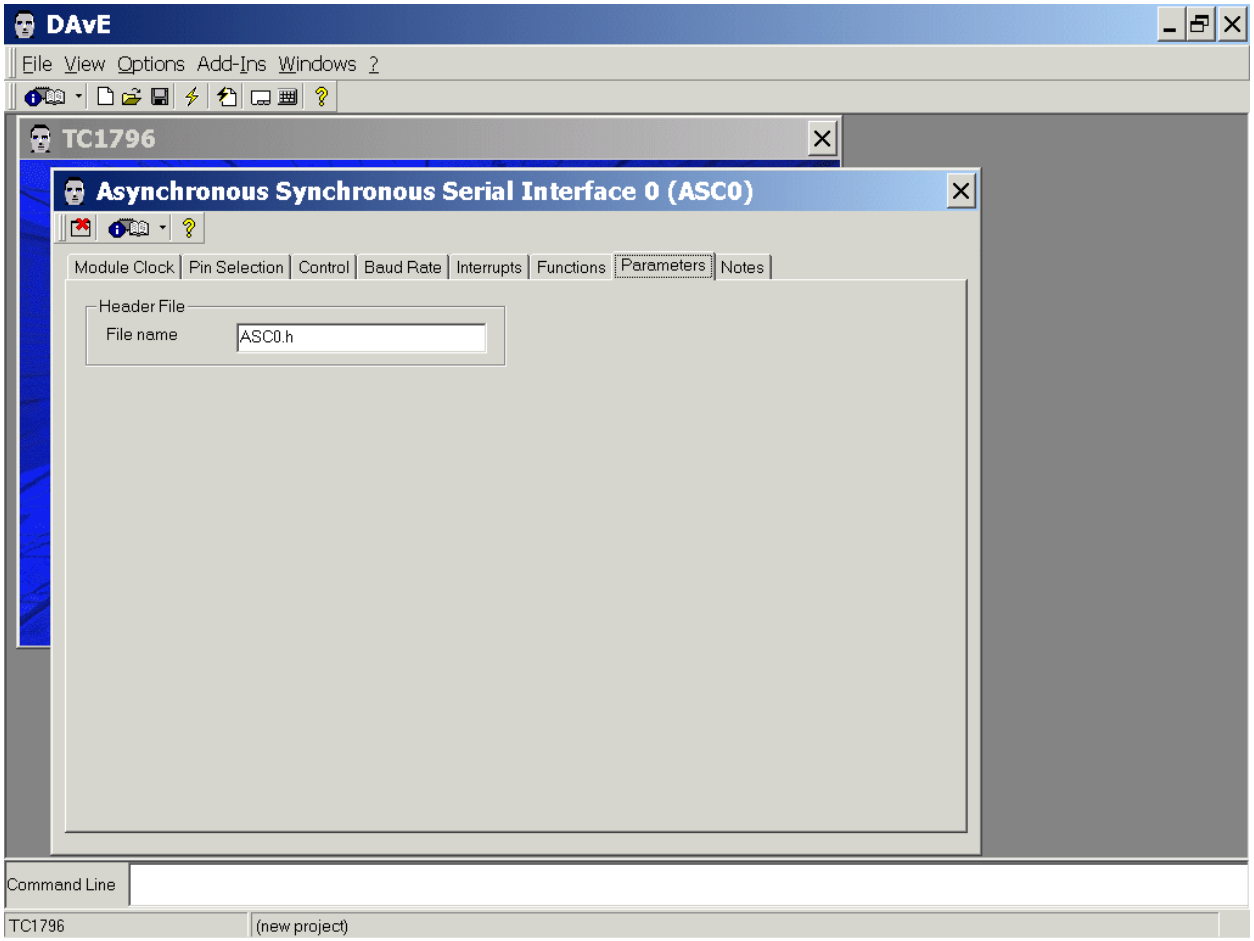


Interrupts: (do nothing)

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

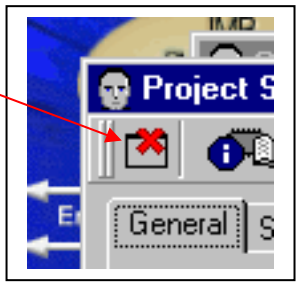


Parameters: (do nothing)



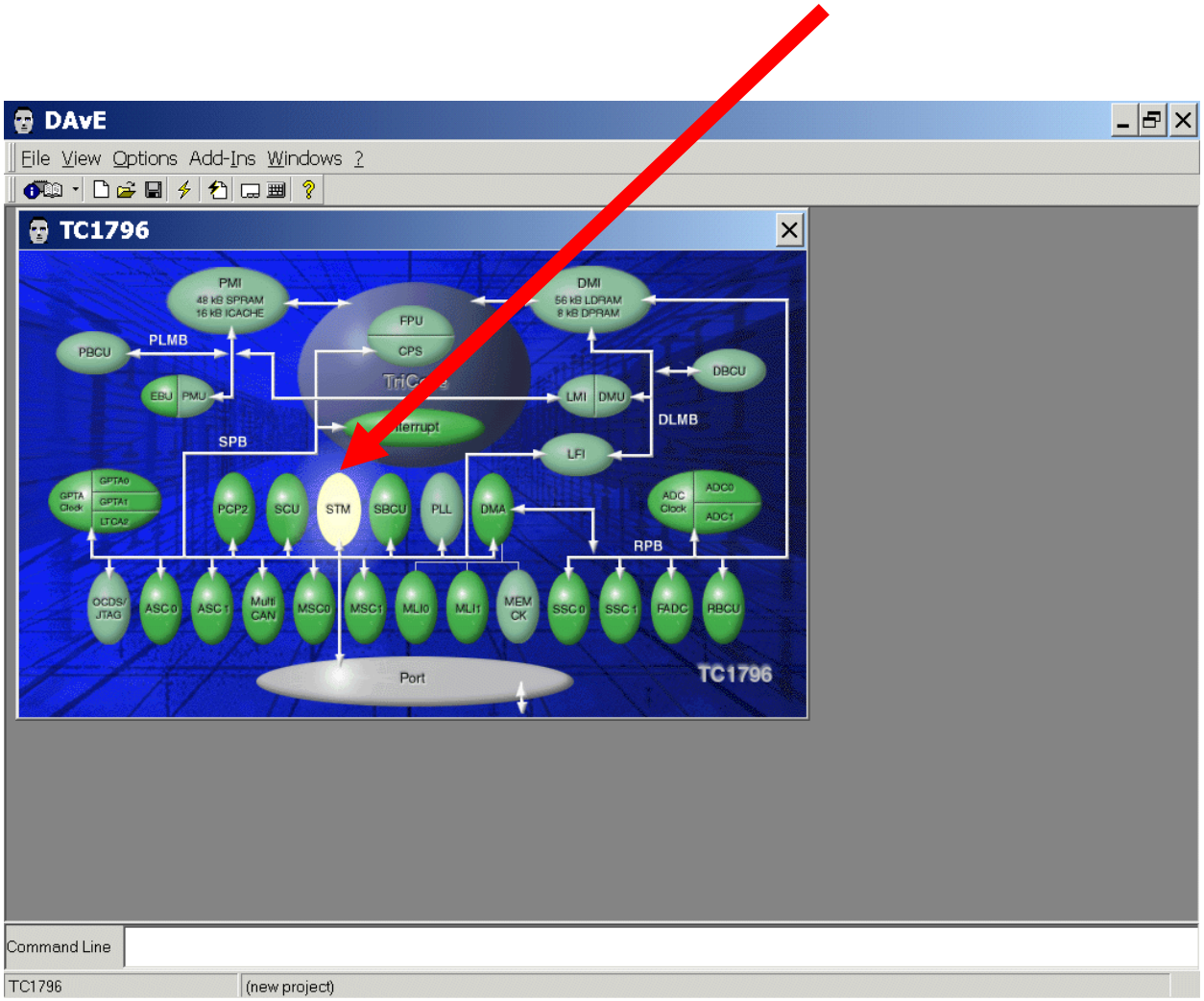
Notes: If you wish, you can insert your comments here.

Exit this dialog now by clicking  the close button:

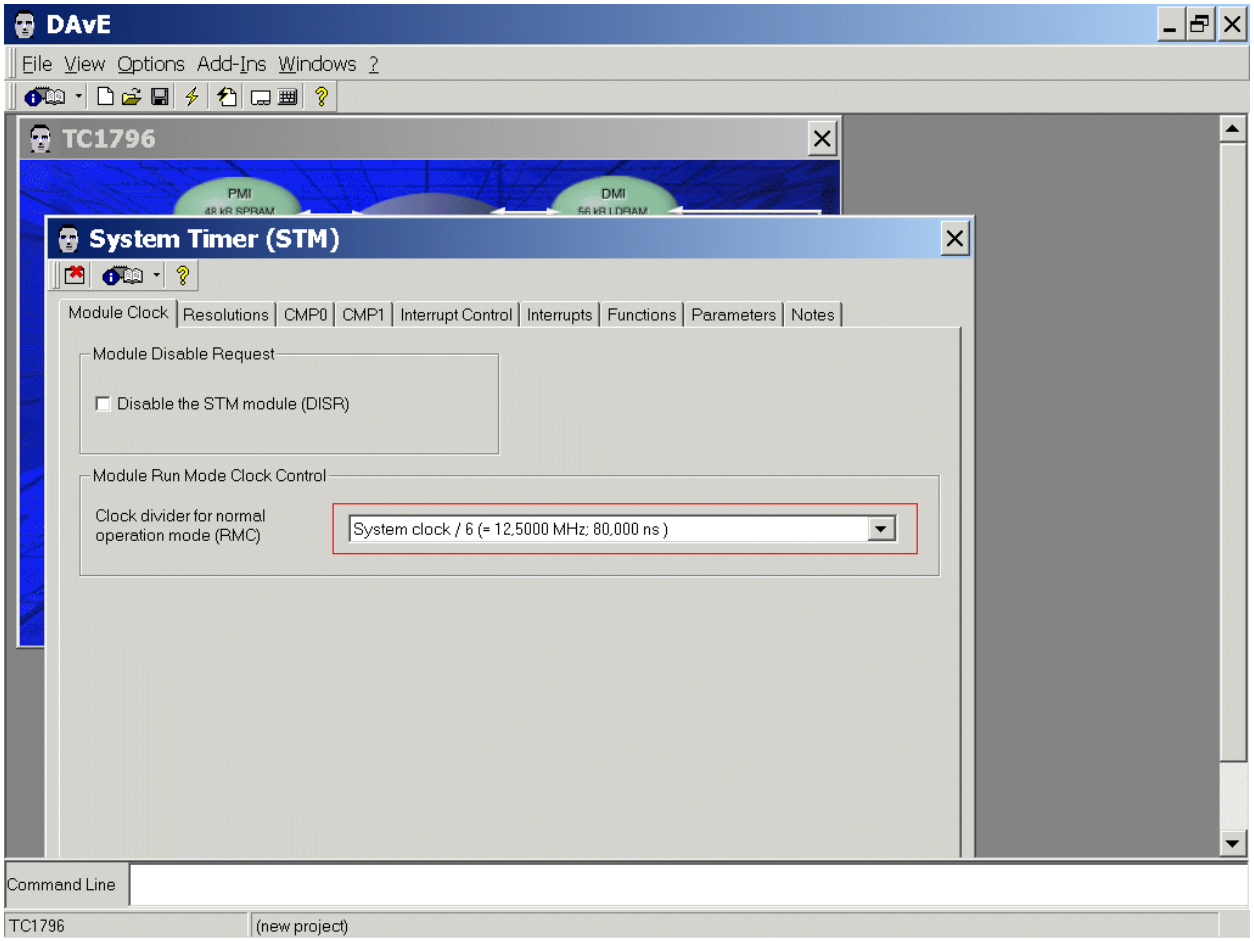


Configuration of the STM:

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

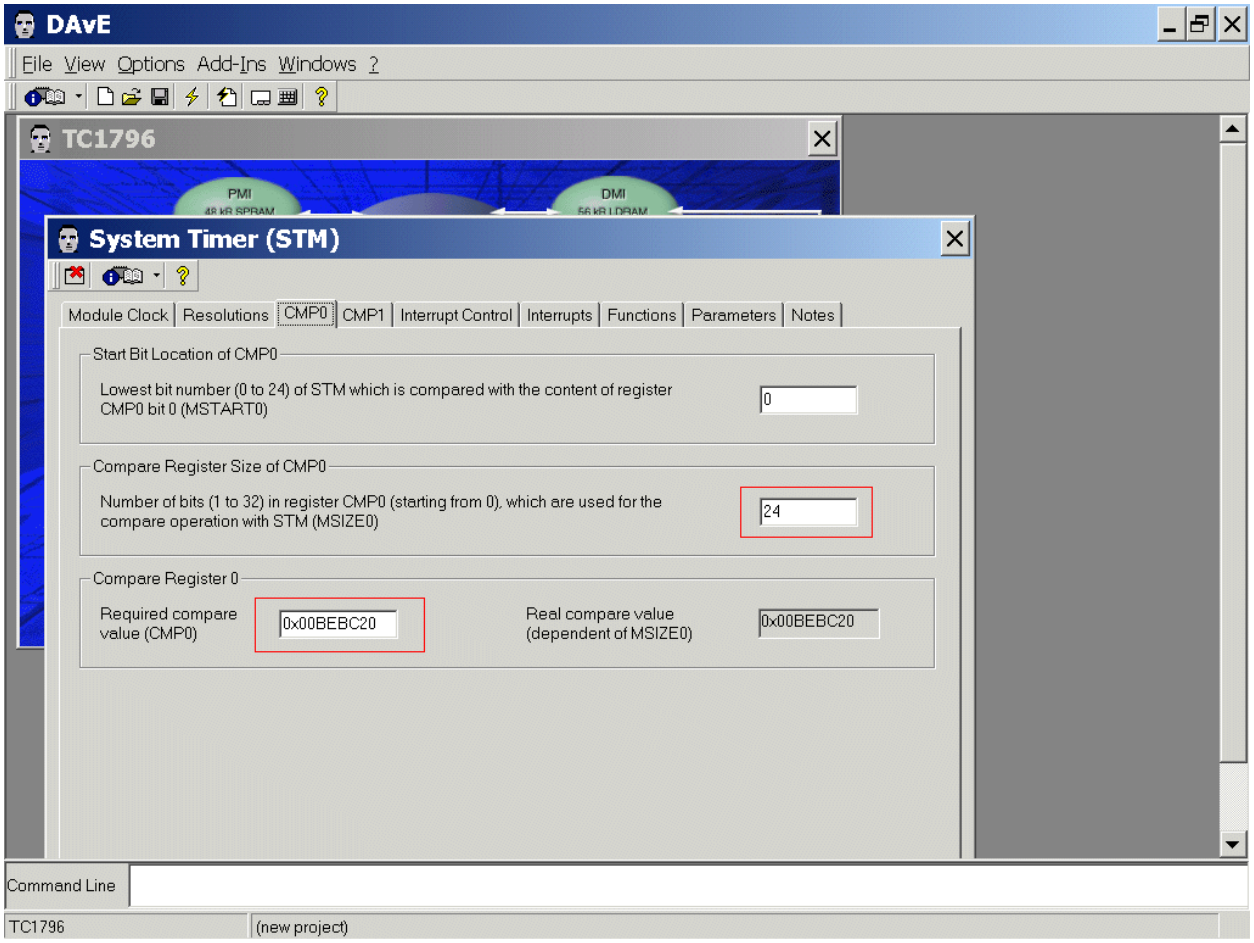


Module Clock: Module Run Mode Clock Control: select System clock / 6 (= 80 ns)



Resolutions: (do nothing)

CMP0: Compare Register Size of CMP0: Number of bits for compare: insert 24
CMP0: Compare Register 0: Required compare value (CMP0): insert 12500000

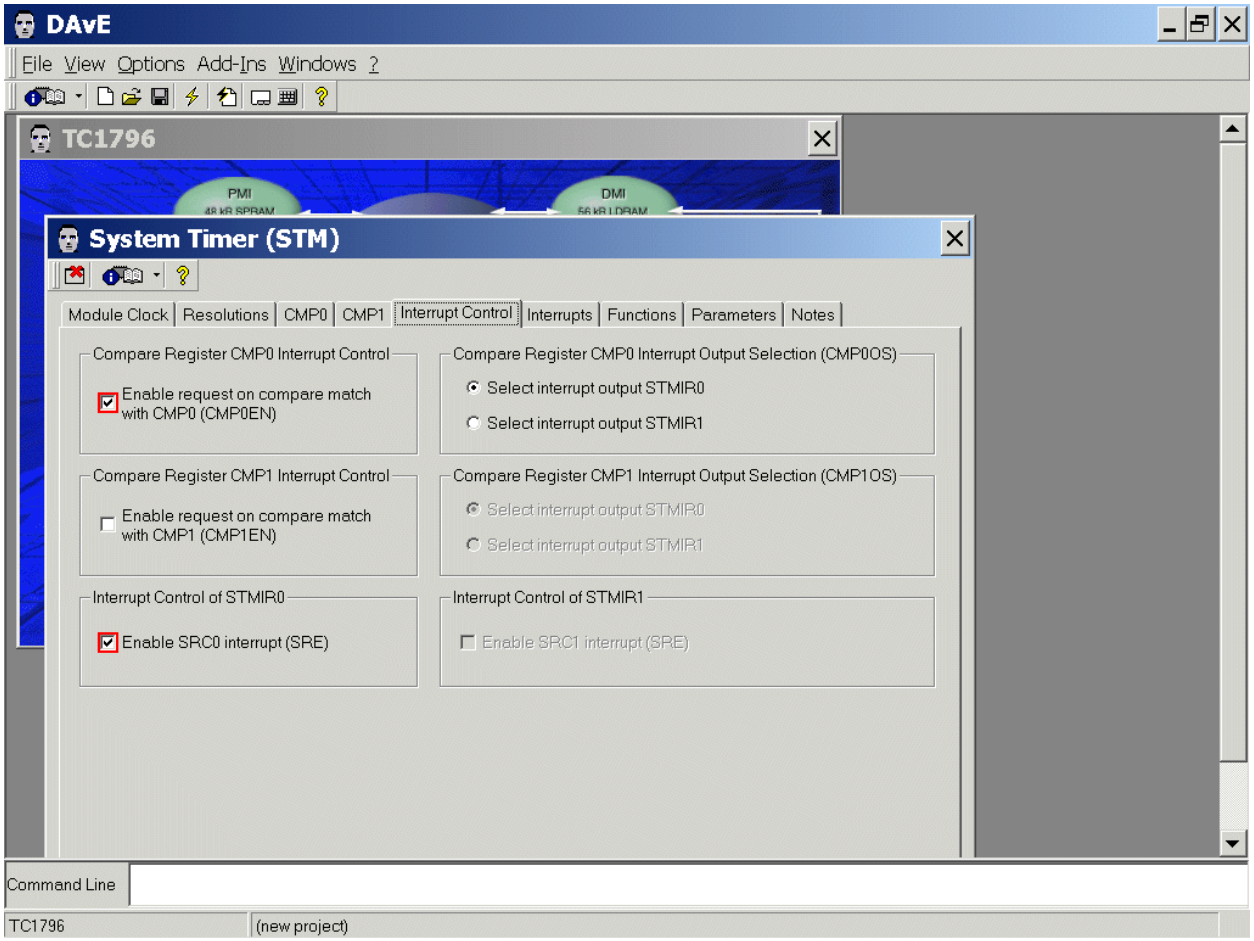


Note: 12500000 * 80 ns = 1 s

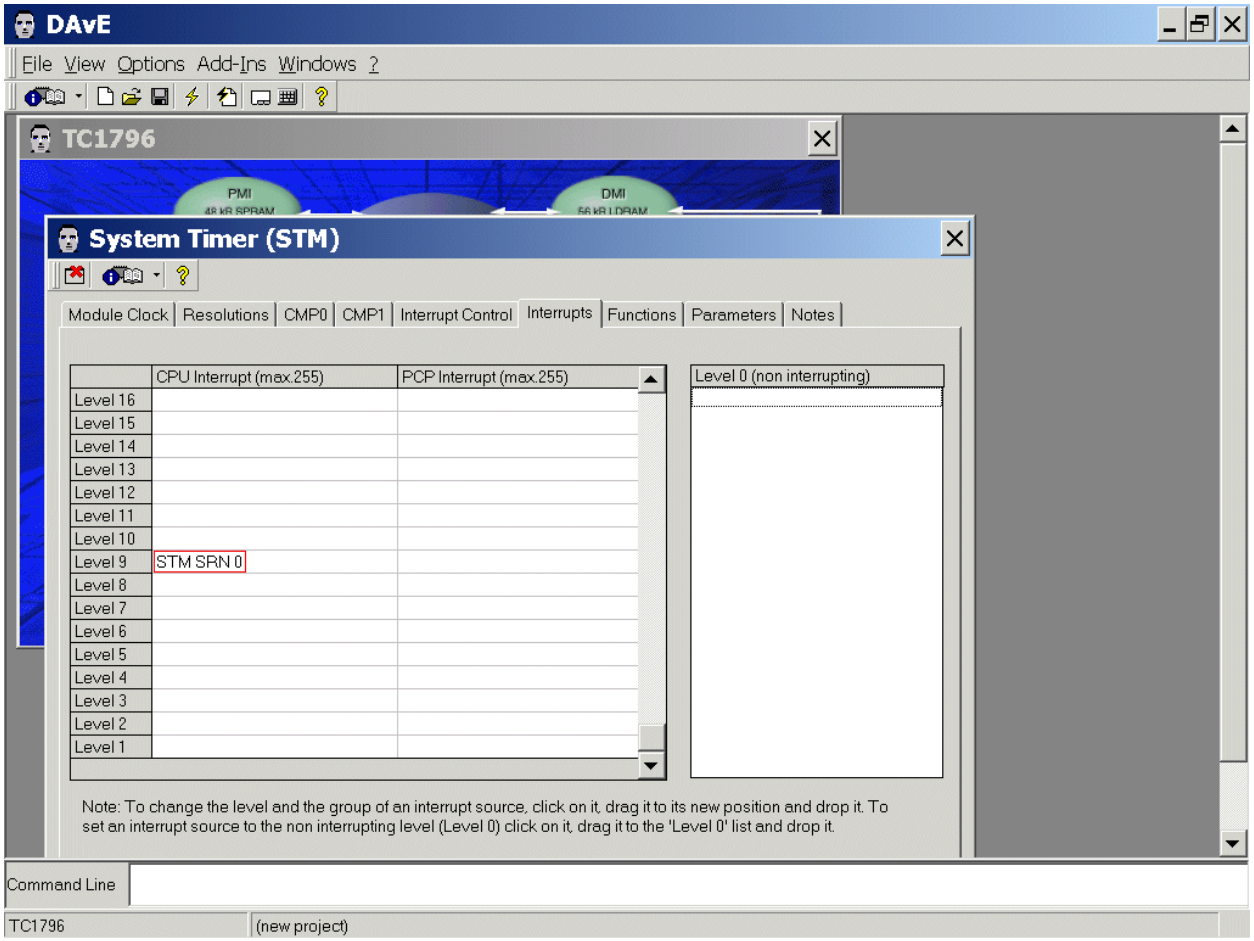
CMP1: (do nothing)

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

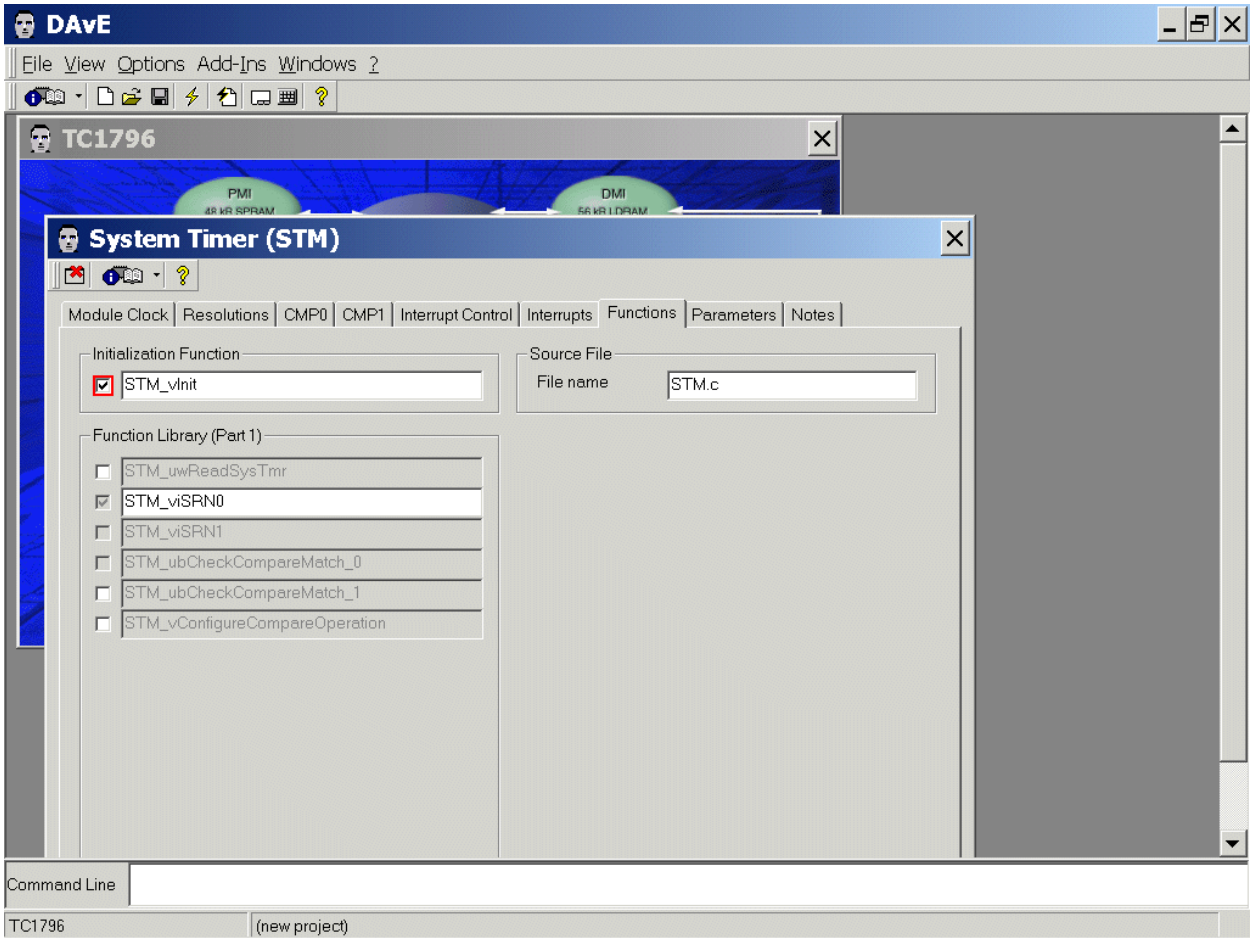
Interrupt Control: Interrupt Control of STMIR0: click ✓ Enable SRC0 interrupt



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



Functions: Initialization Function: click ✓ STM_vInit



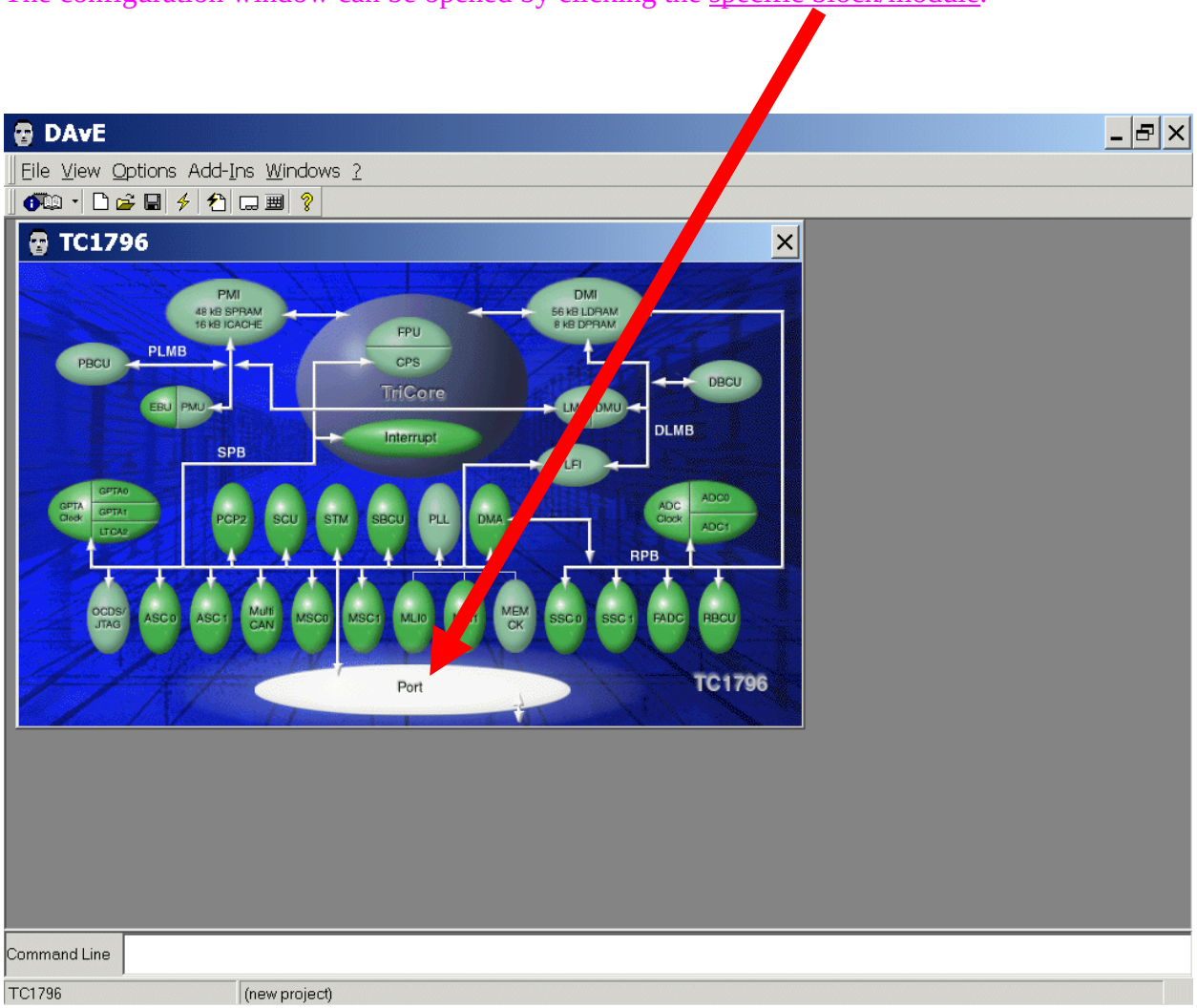
Parameters: (do nothing)

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

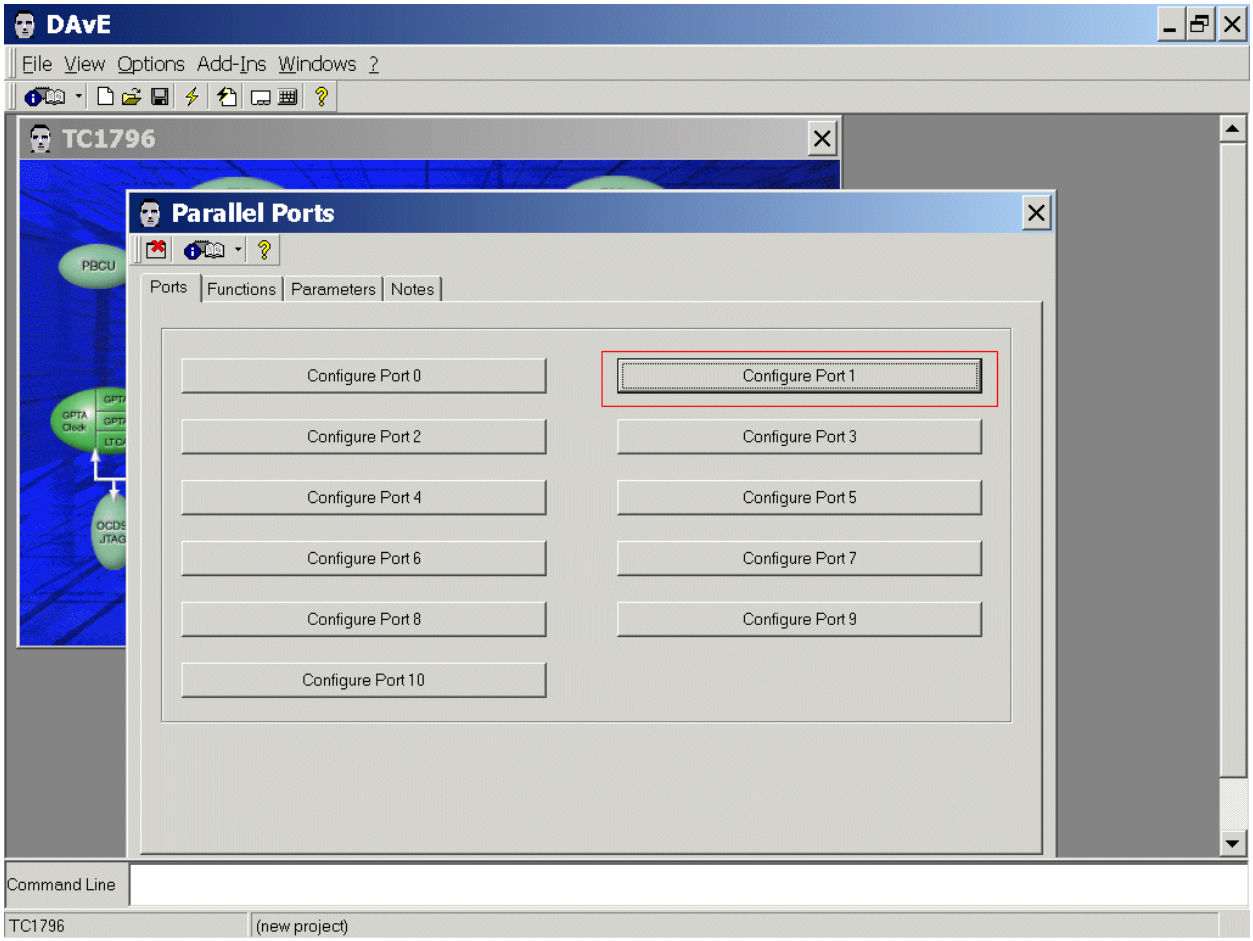
Exit this dialog now by clicking  the close button.

Port Configuration:

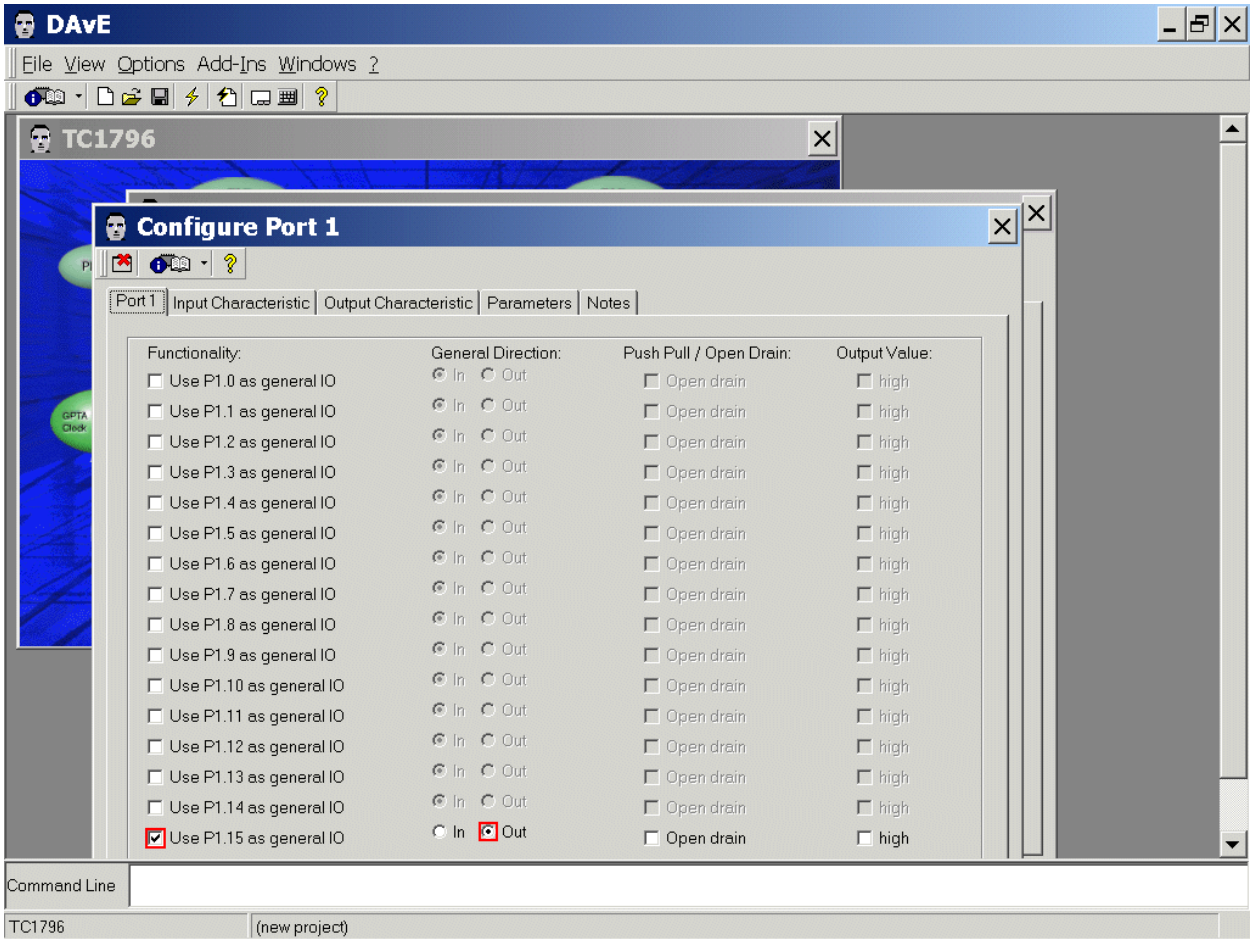
The configuration window can be opened by clicking the specific block/module.



Ports: click Configure Port 1



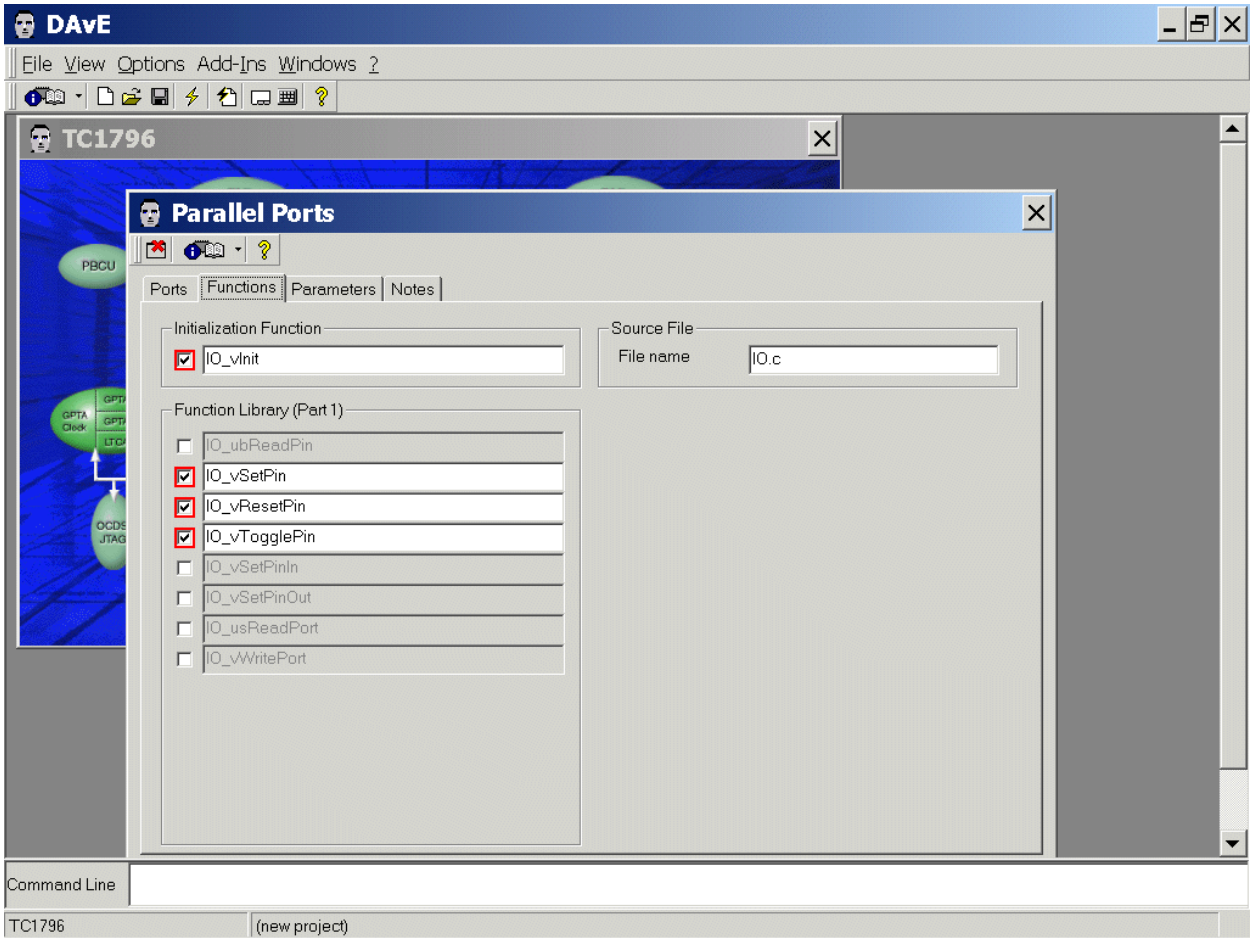
Ports: Configure Port 1
Port 1: Functionality: click ✓ Use P1.15 as general IO, General Direction: click ⊙ Out



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

Exit this dialog now by clicking  the close button.

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



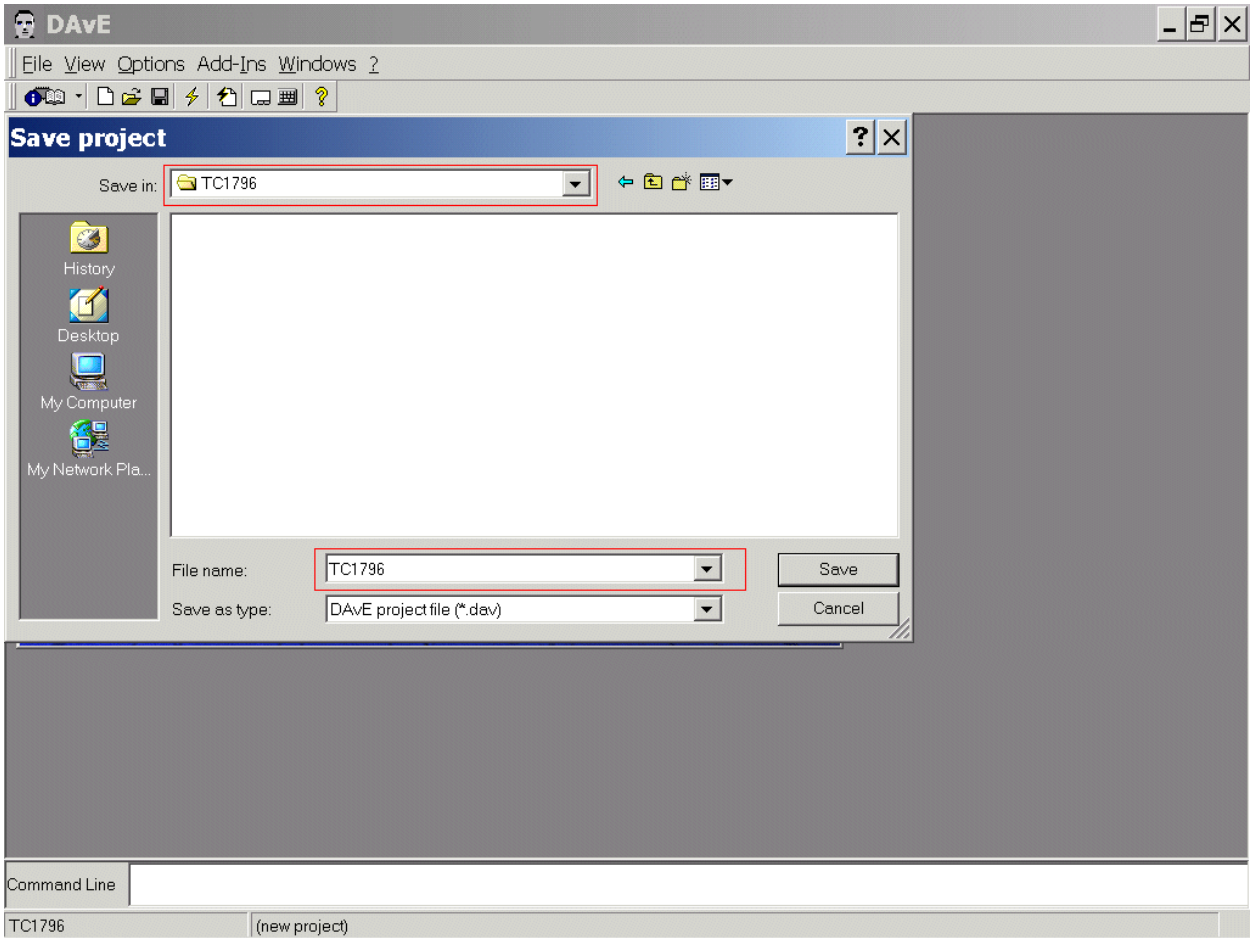
Parameters : (do nothing)
Notes: If you wish, you can insert your comments here.

Exit this dialog now by clicking the close button.

Save the project:

File
Save

Save project: Save in: C:\TC1796 (create directory)
File name: TC1796

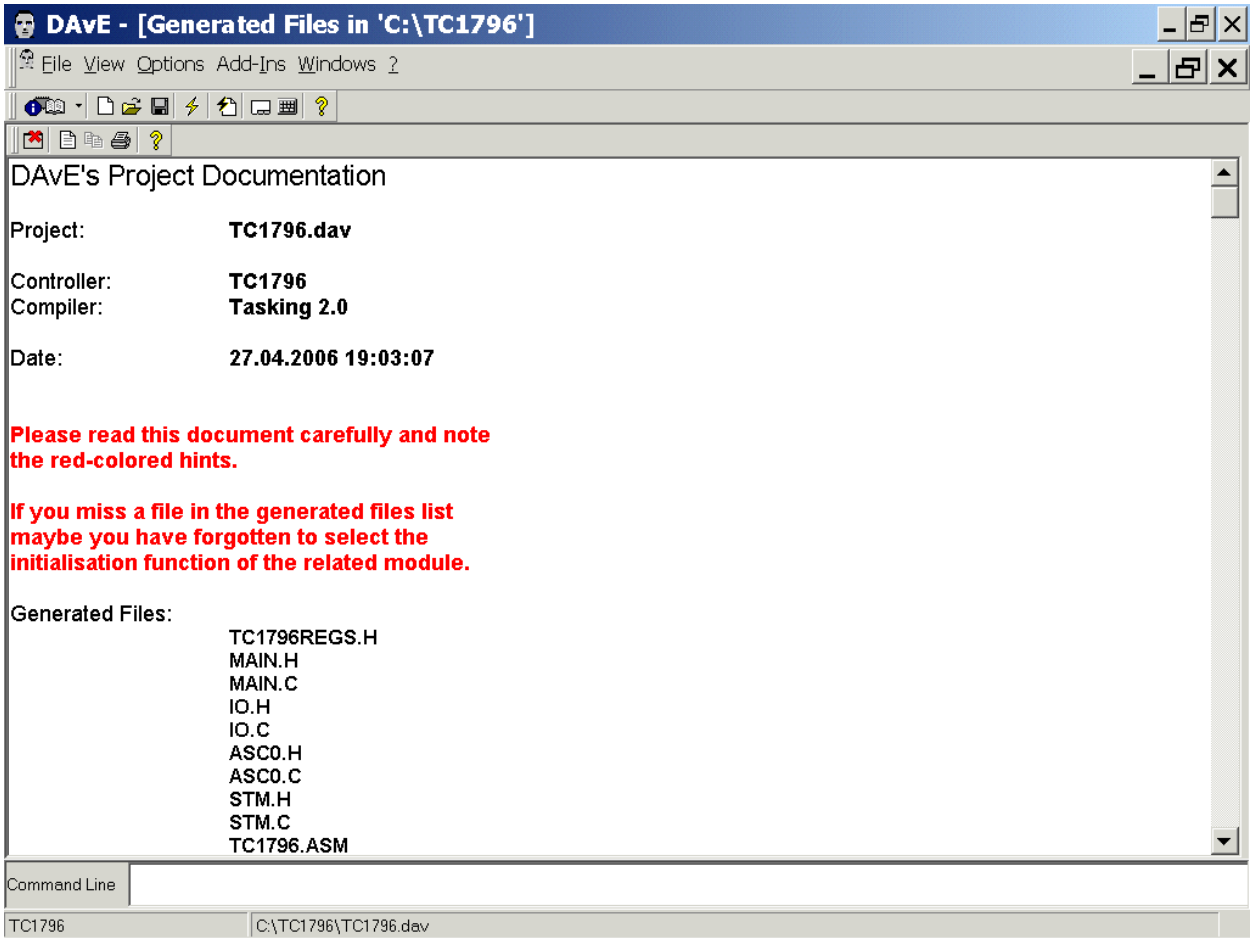


Save

Generate Code:

File Generate Code	or click 

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



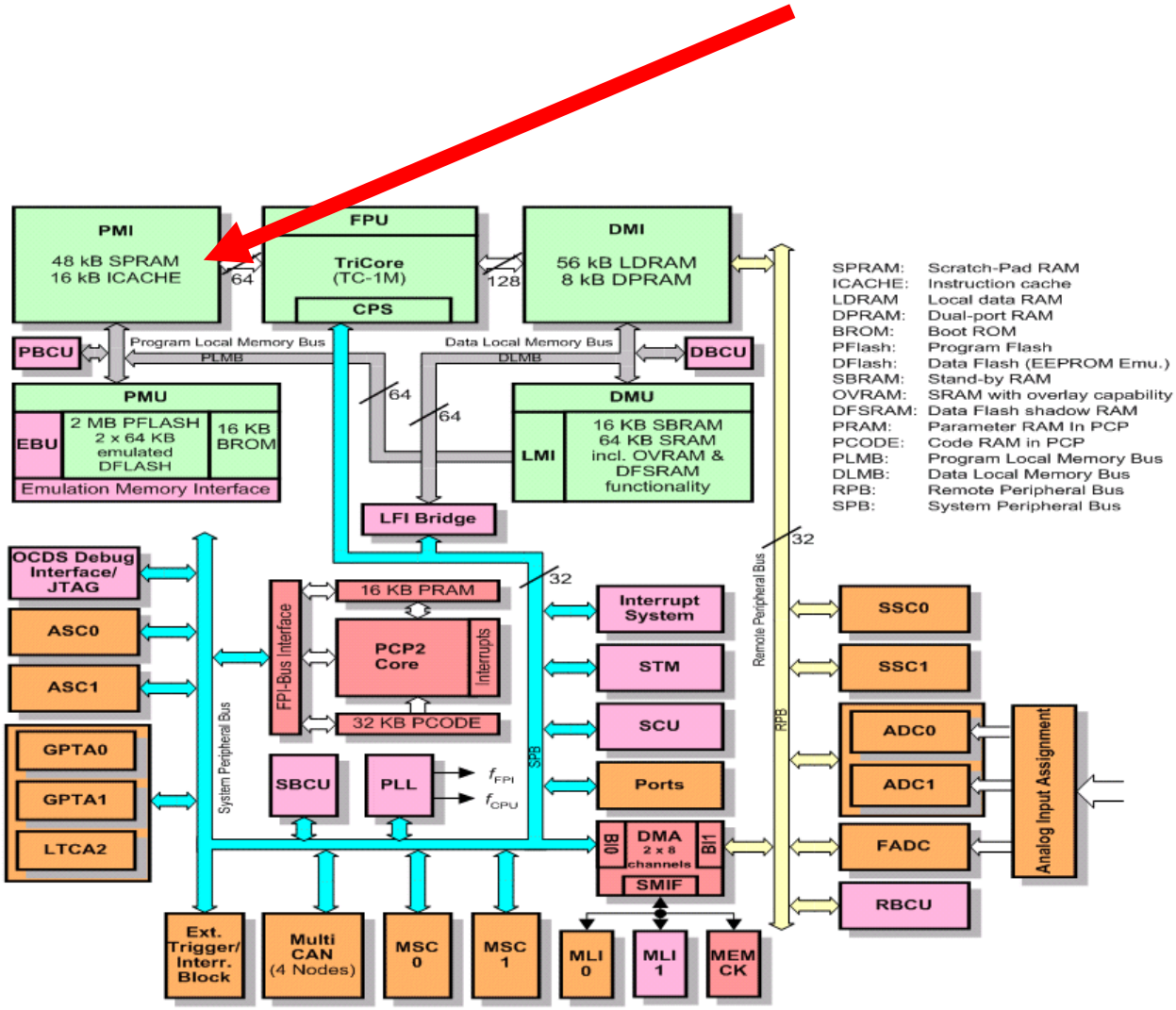
File
Exit

Save changes?

click Yes

4.) Using of the TASKING - EDE Development Tools:

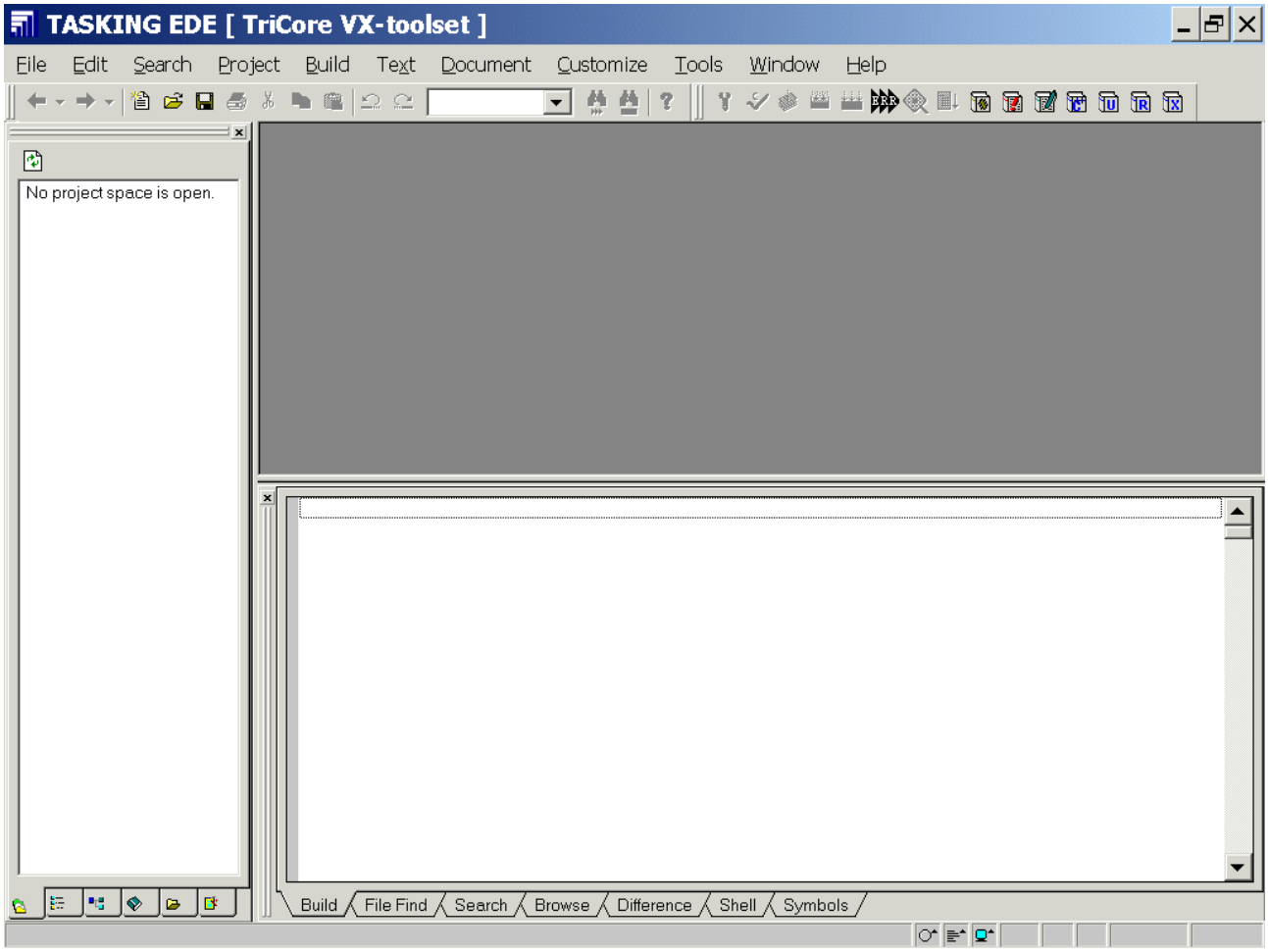
Write programs for execution from Scratch-Pad RAM (SPRAM) 



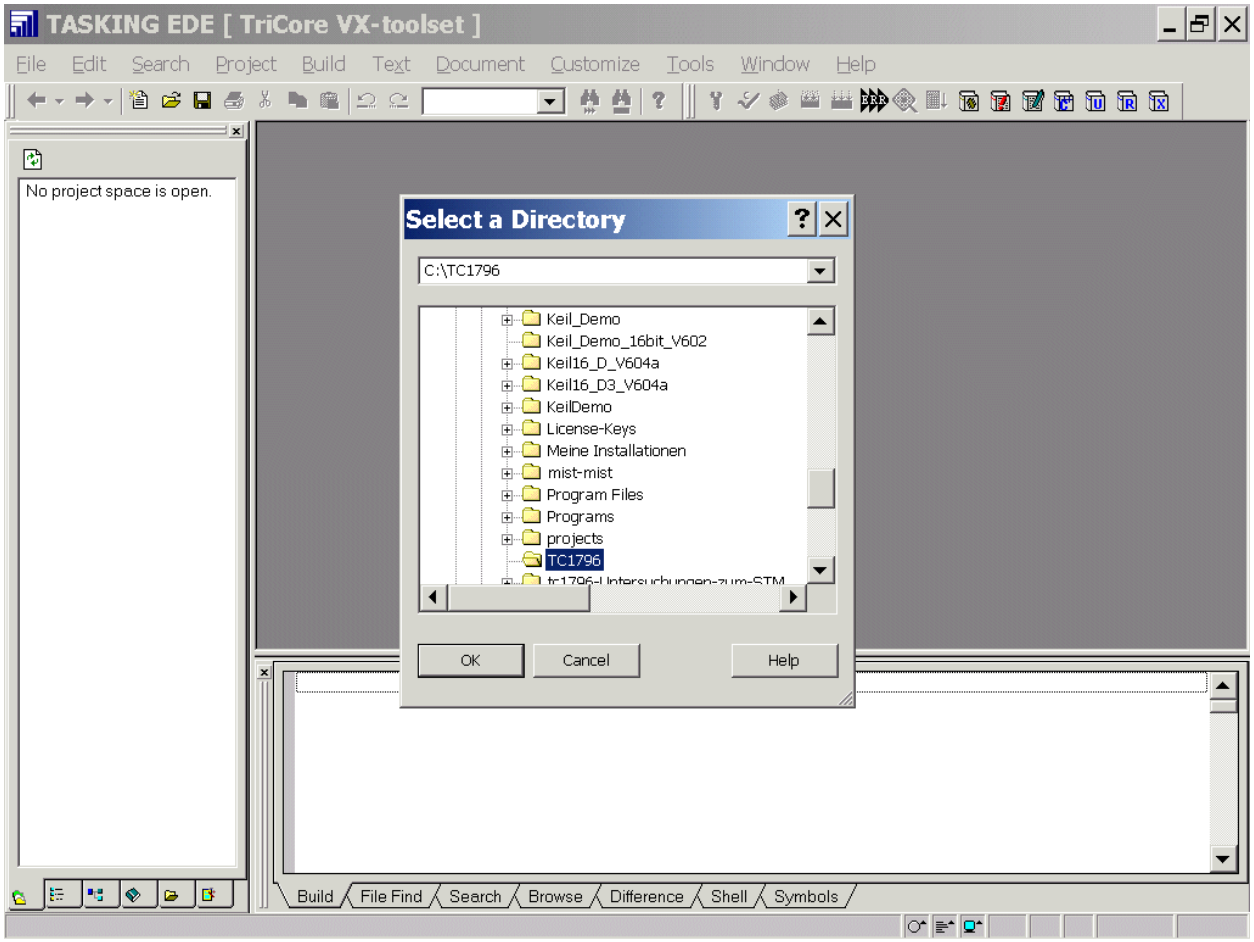
Install the Tasking Development Tools TriCore v2.3r1

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

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

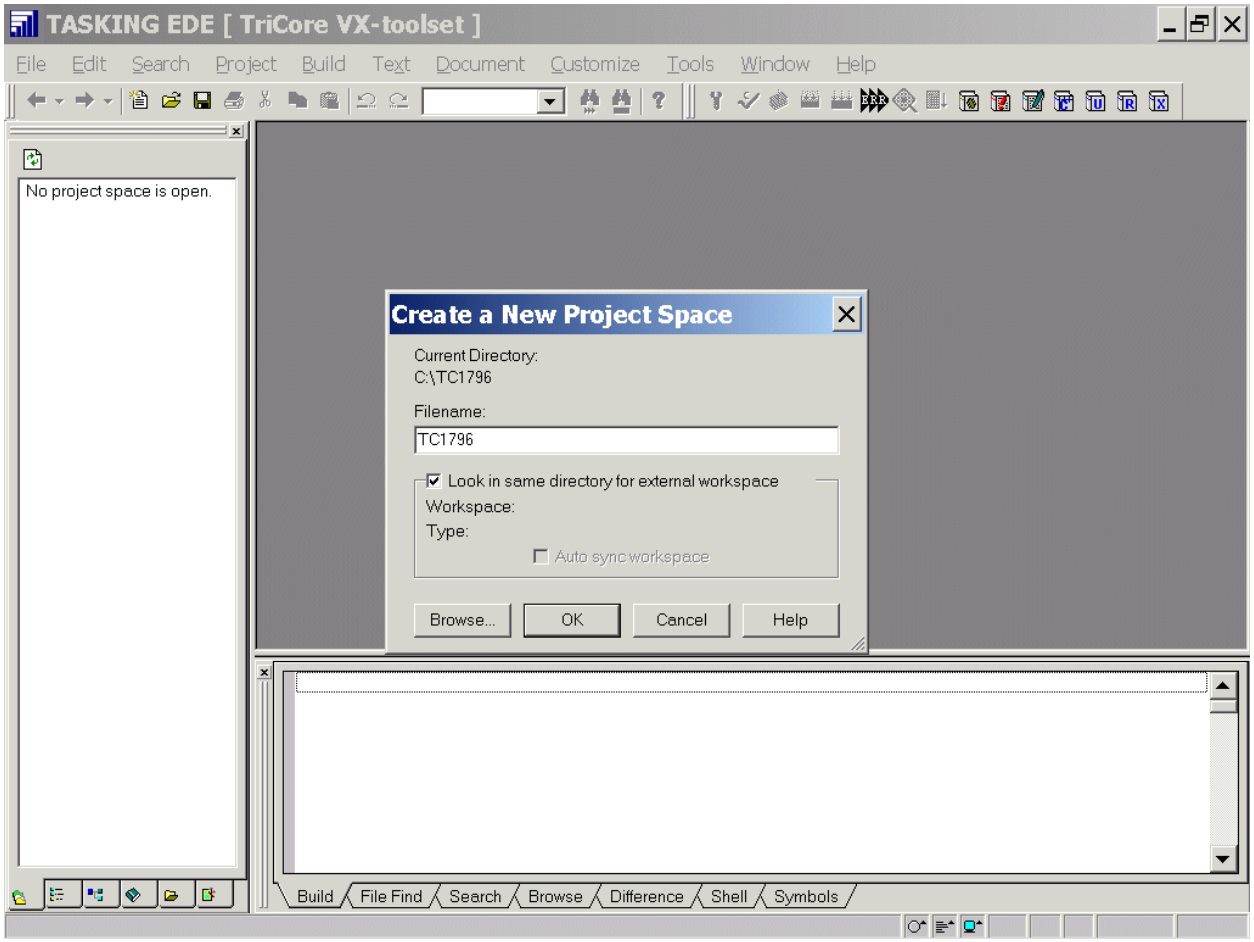


File
Change Directory
Choose C:\TC1796



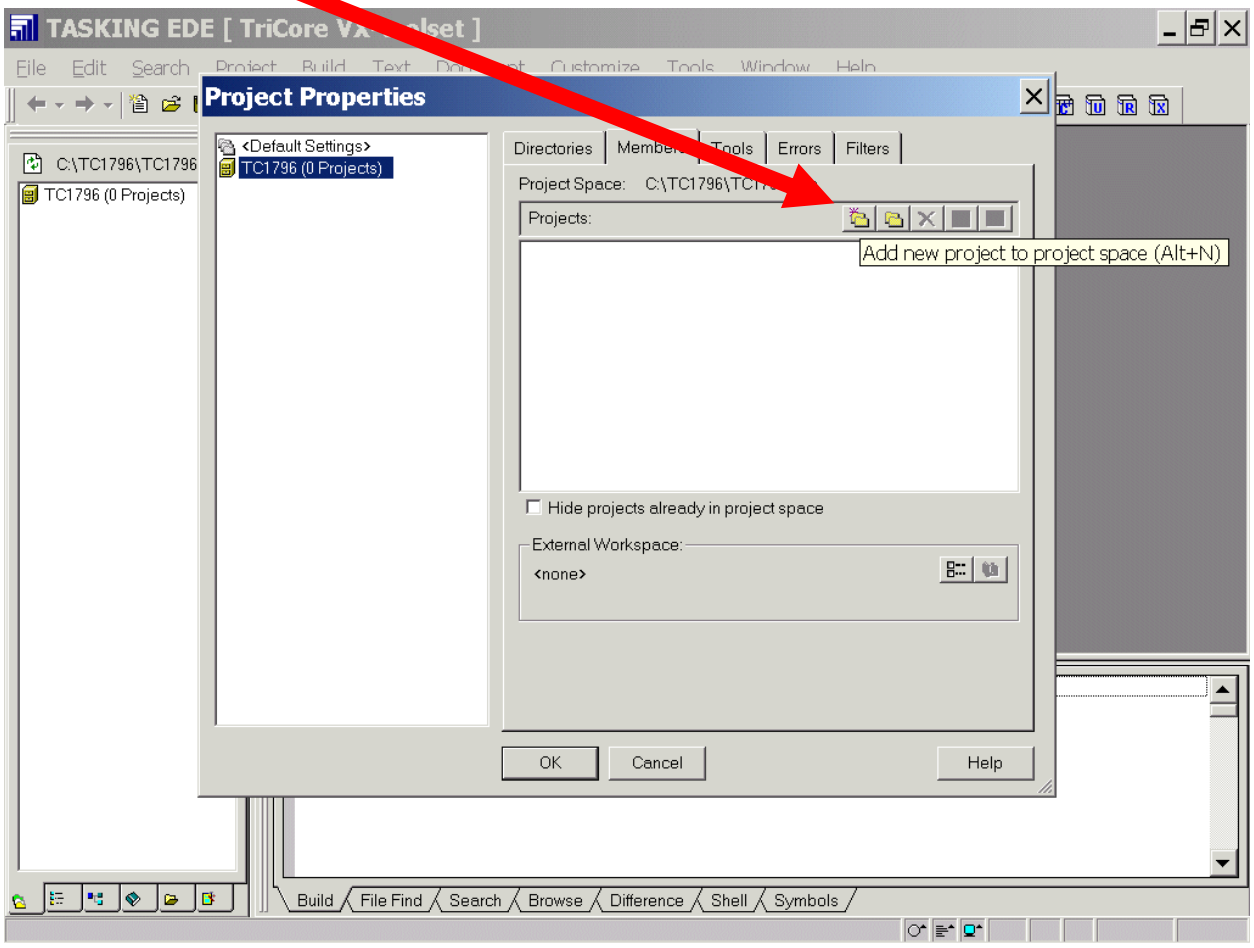
OK

File
New Project Space
Insert TC1796

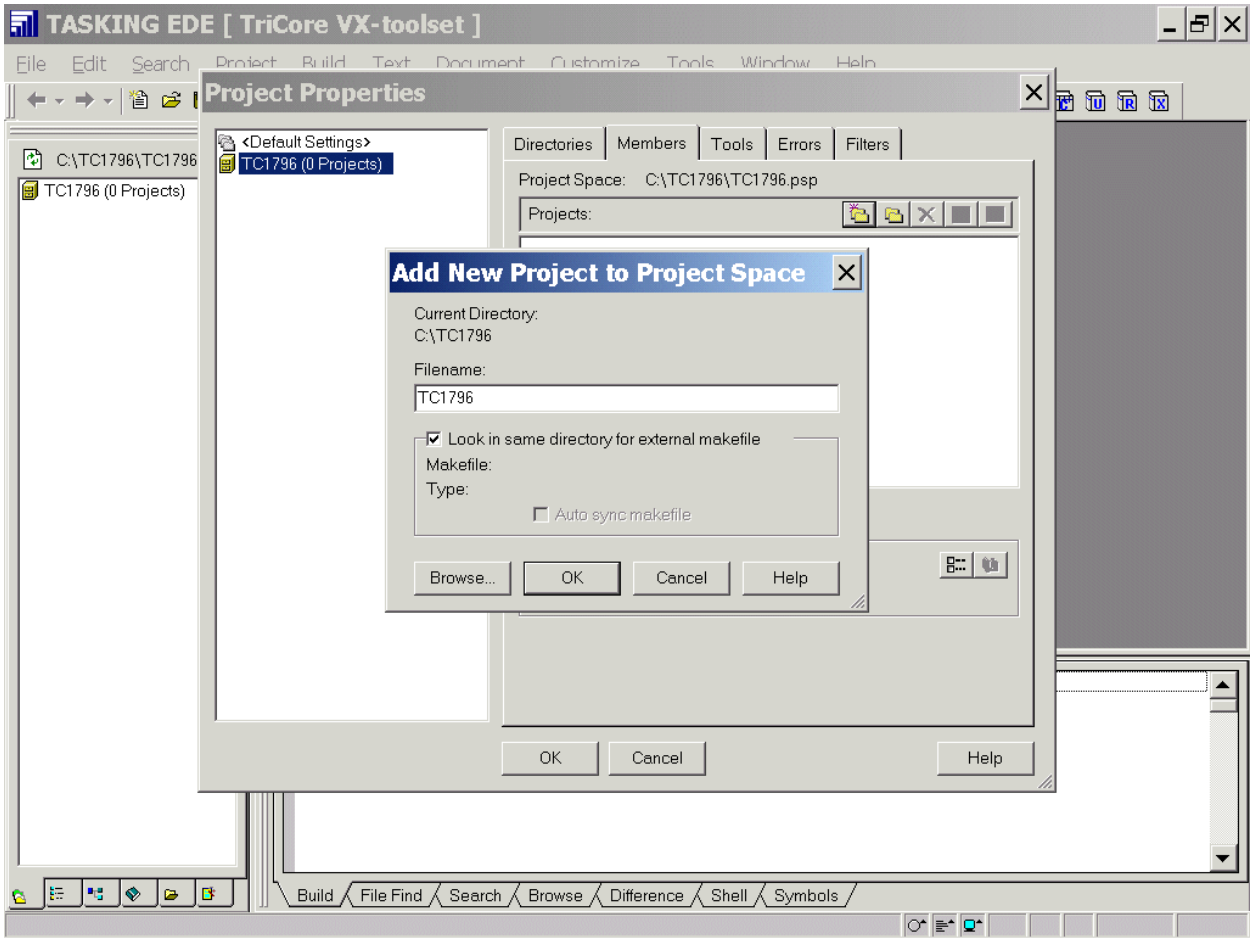


OK

Click: “Add new project to project space”

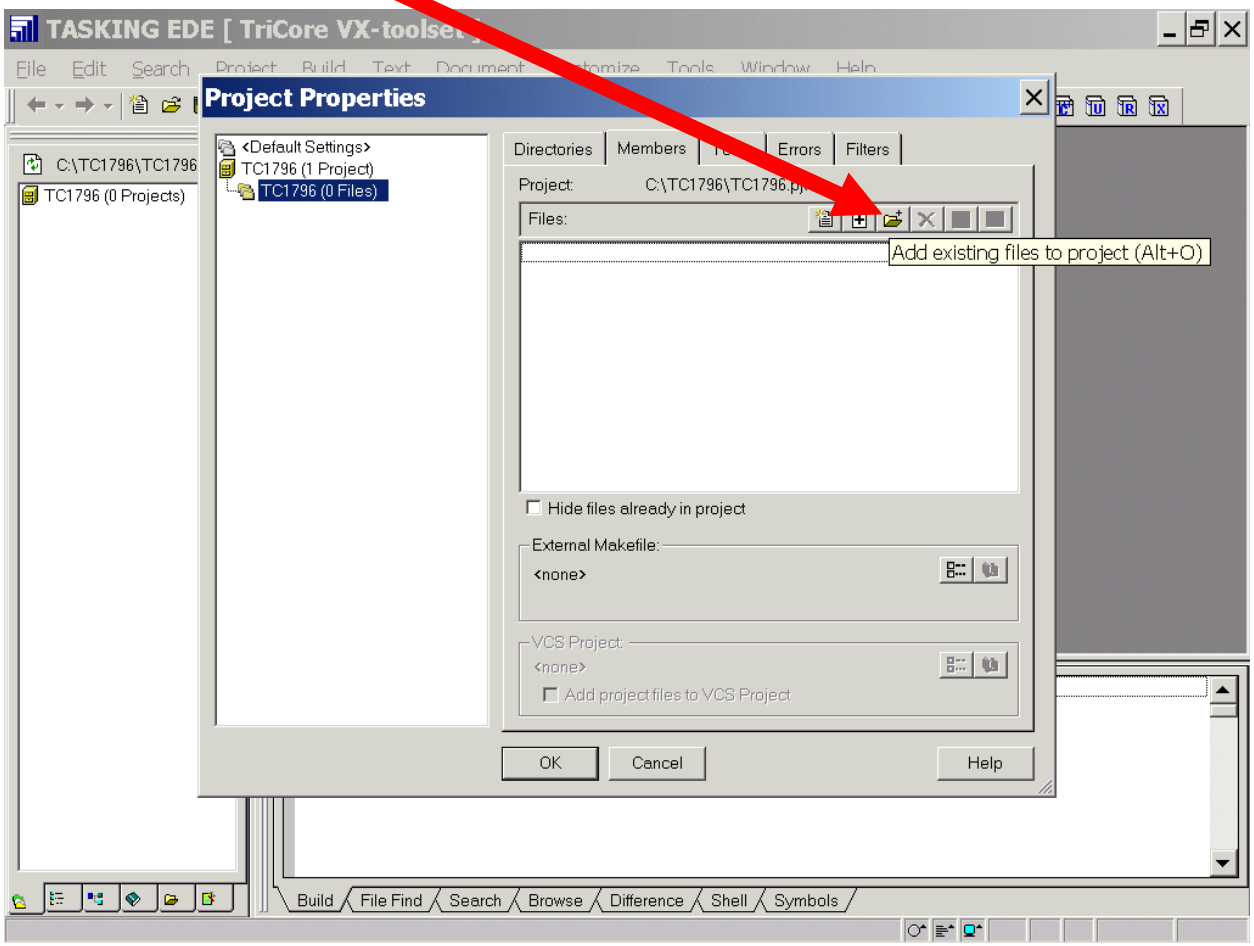


Insert TC1796

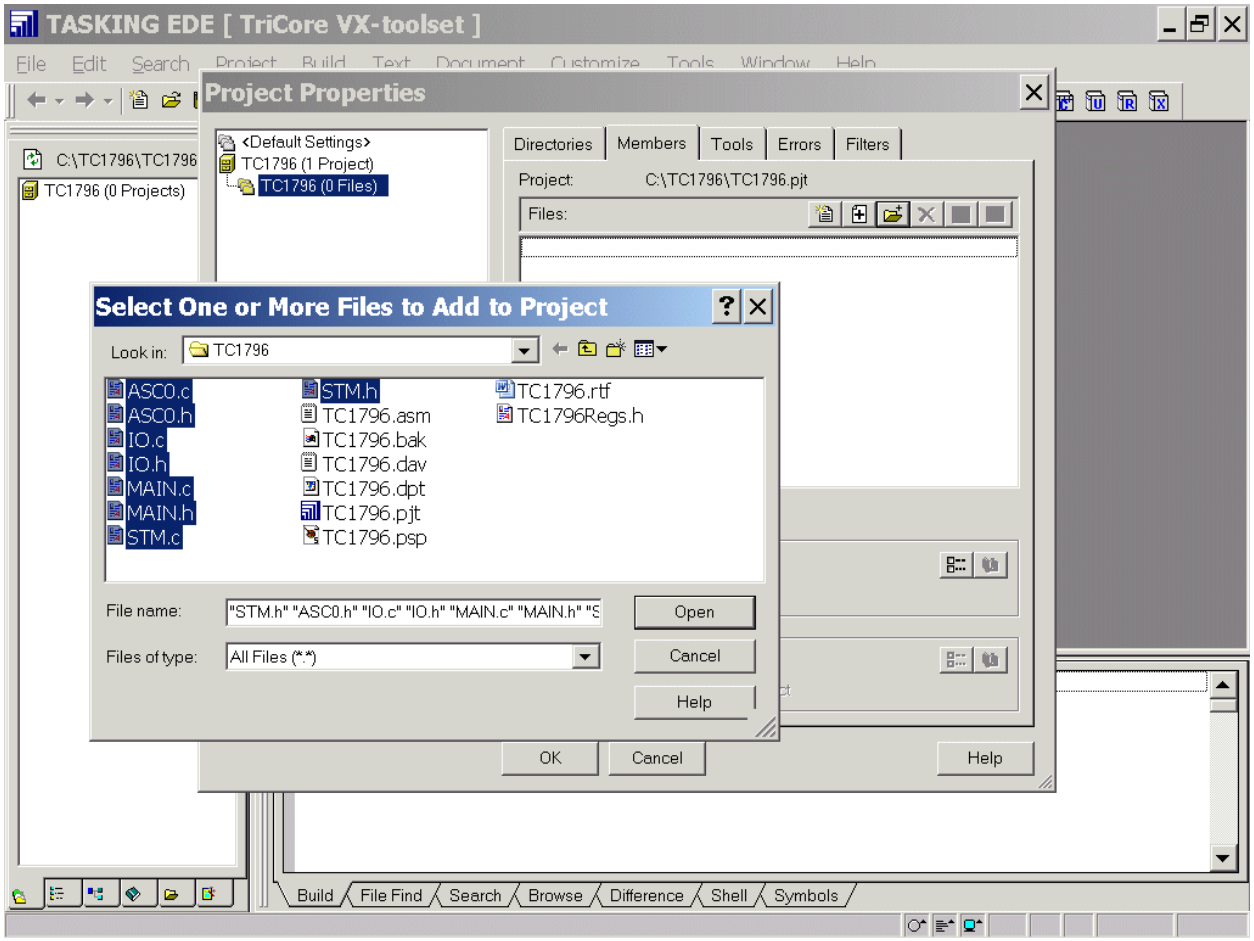


OK

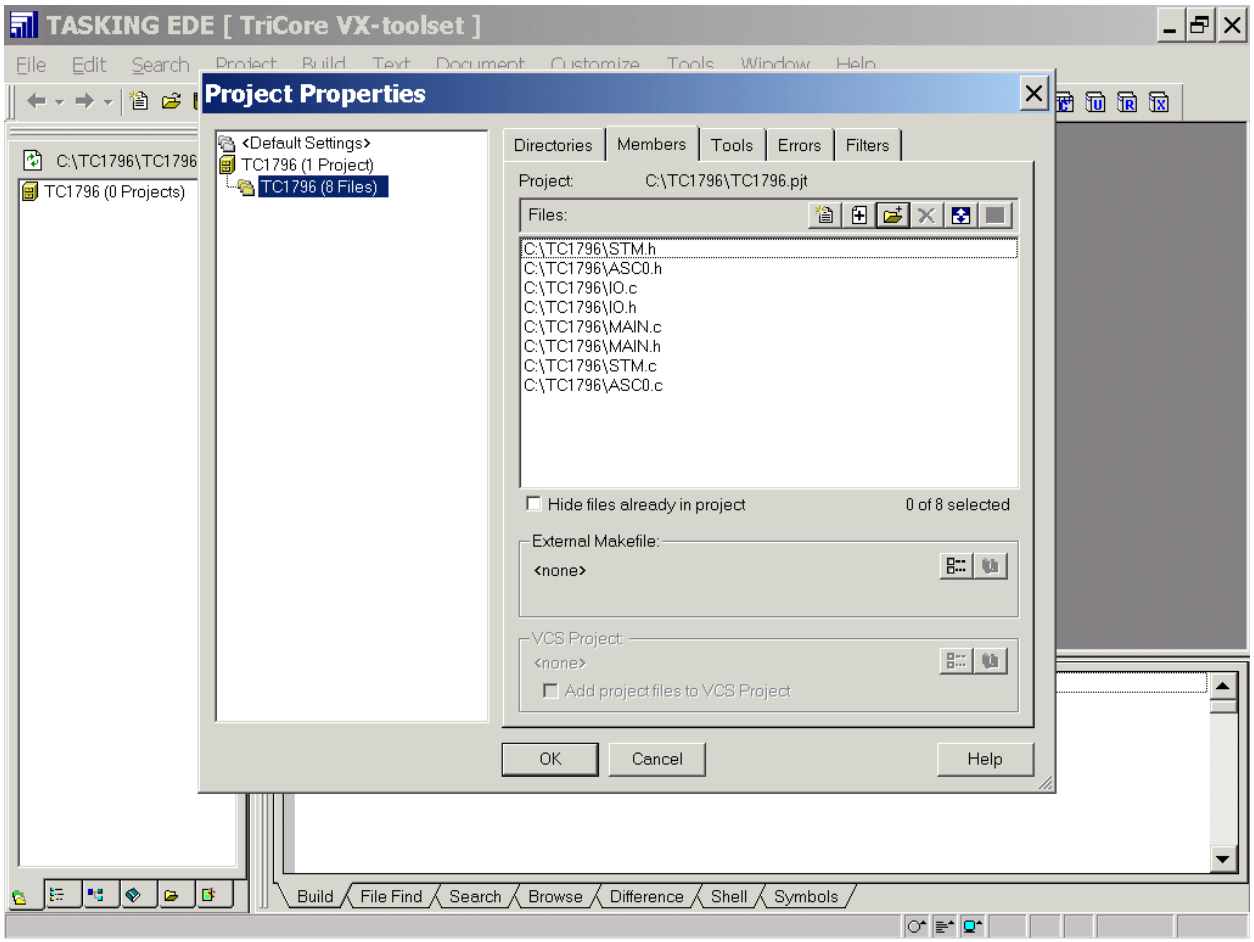
Click: “Add existing files to project”



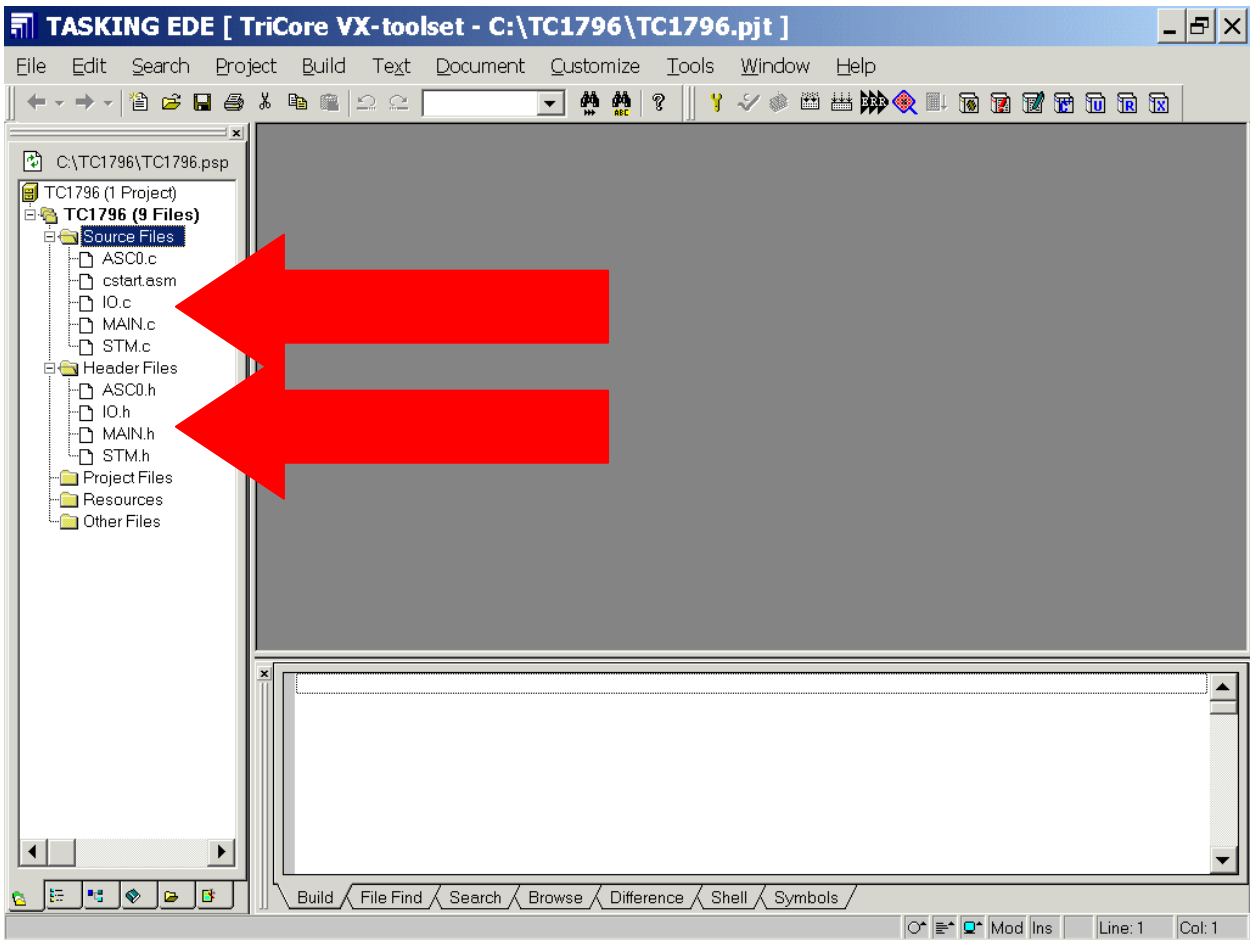
Select ASC0.c
Select ASC0.h
Select IO.c
Select IO.h
Select MAIN.c
Select MAIN.h
Select STM.c
Select STM.h



Open


TASKING EDE [TriCore VX-toolset]
File Edit Search Project Build Text Document Customize Tools Window Help
Project Properties
<Default Settings>
TC1796 (1 Project)
TC1796 (8 Files)
Directories Members Tools Errors Filters
Project: C:\TC1796\TC1796.pjt
Files:
C:\TC1796\STM.h
C:\TC1796\ASC0.h
C:\TC1796\IO.c
C:\TC1796\IO.h
C:\TC1796\MAIN.c
C:\TC1796\MAIN.h
C:\TC1796\STM.c
C:\TC1796\ASC0.c
Hide files already in project
0 of 8 selected
External Makefile:
<none>
VCS Project:
<none>
Add project files to VCS Project
OK
Cancel
Help
C:\TC1796\TC1796
TC1796 (0 Projects)
Build File Find Search Browse Difference Shell Symbols

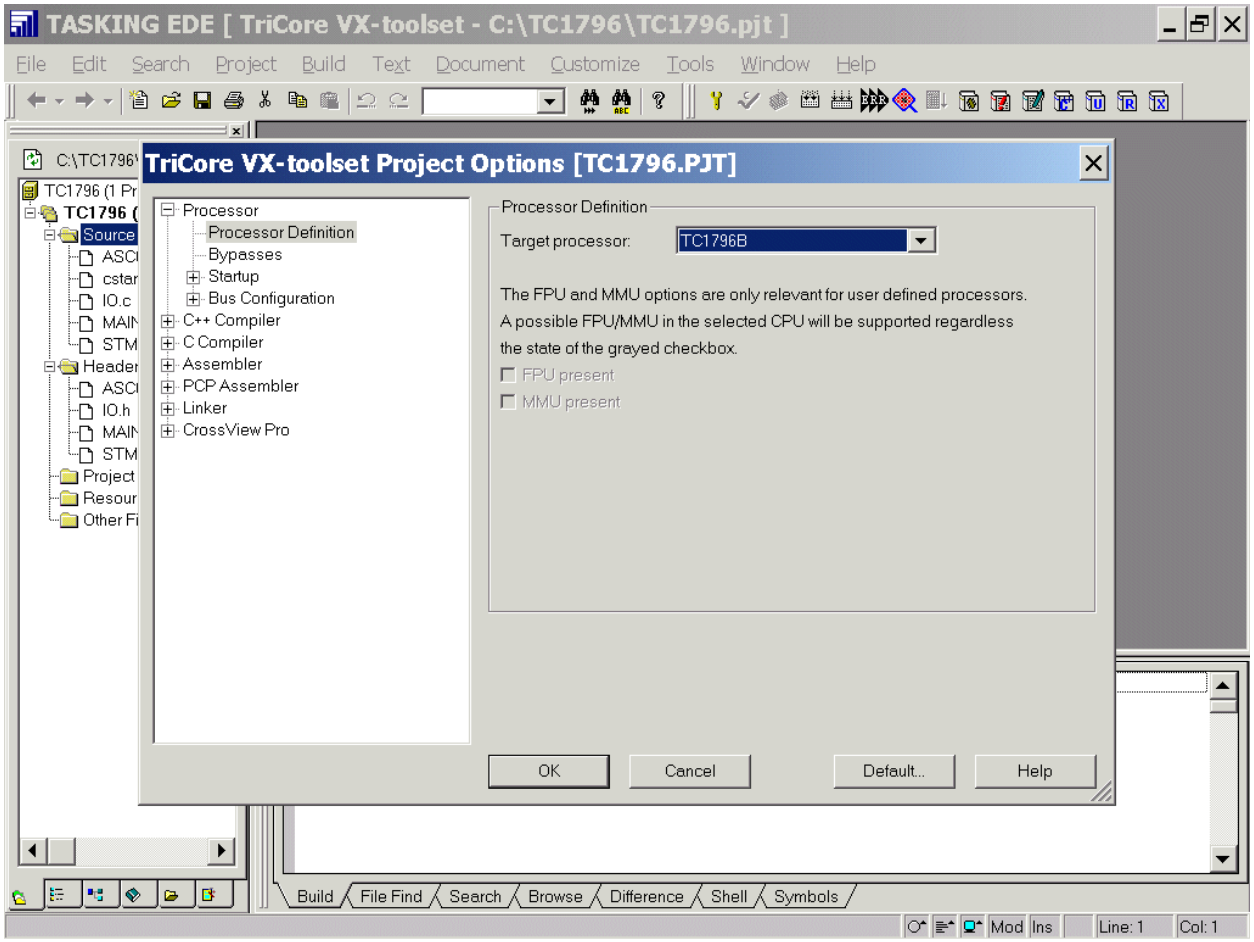
OK


TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1796\TC1796.psp
TC1796 (1 Project)
TC1796 (9 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
Project Files
Resources
Other Files
Build File Find Search Browse Difference Shell Symbols
Mod Ins Line: 1 Col: 1

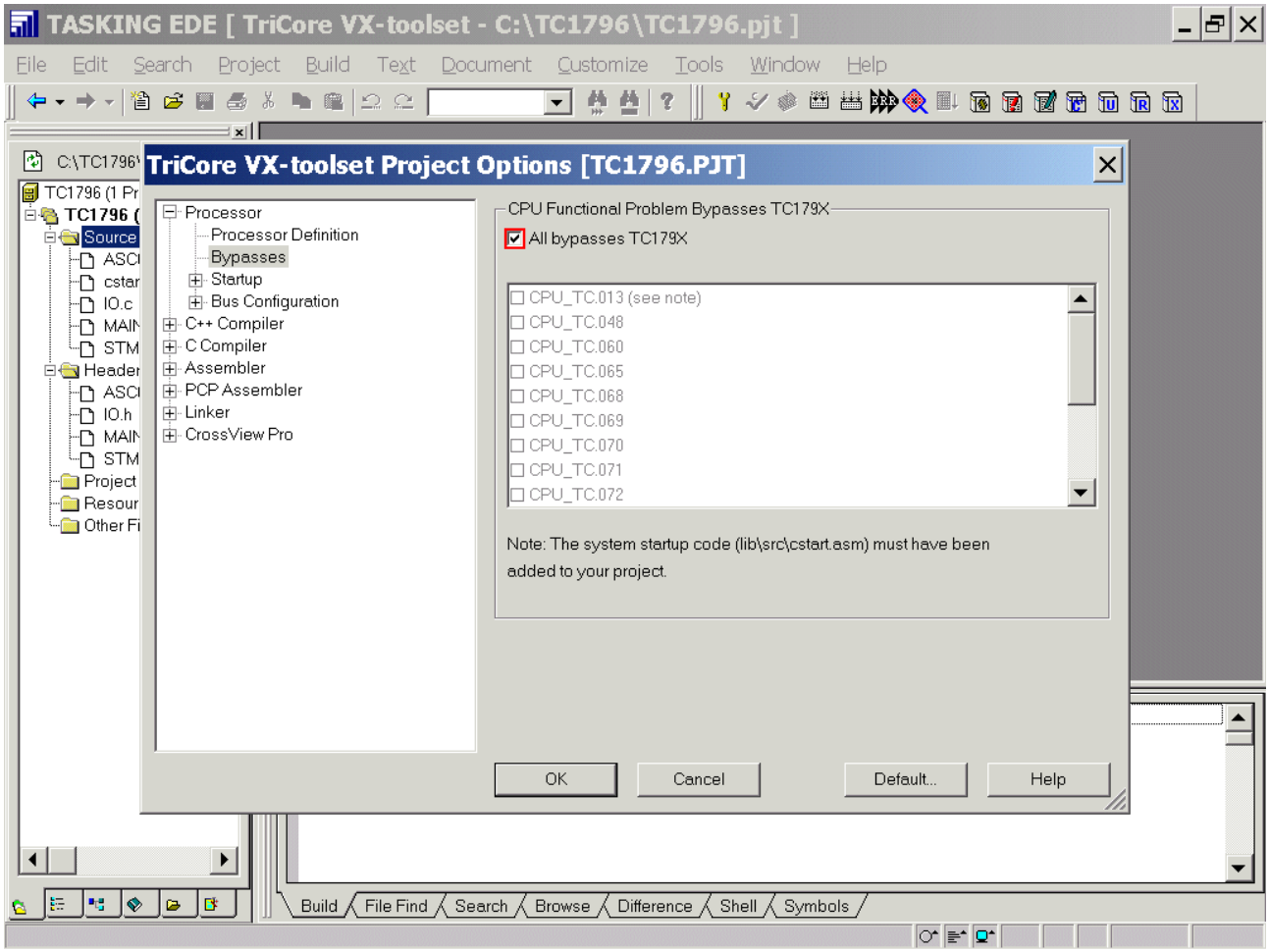
Configure Compiler, Assembler, Linker, Locater and Build – Control:

Project – Project Options

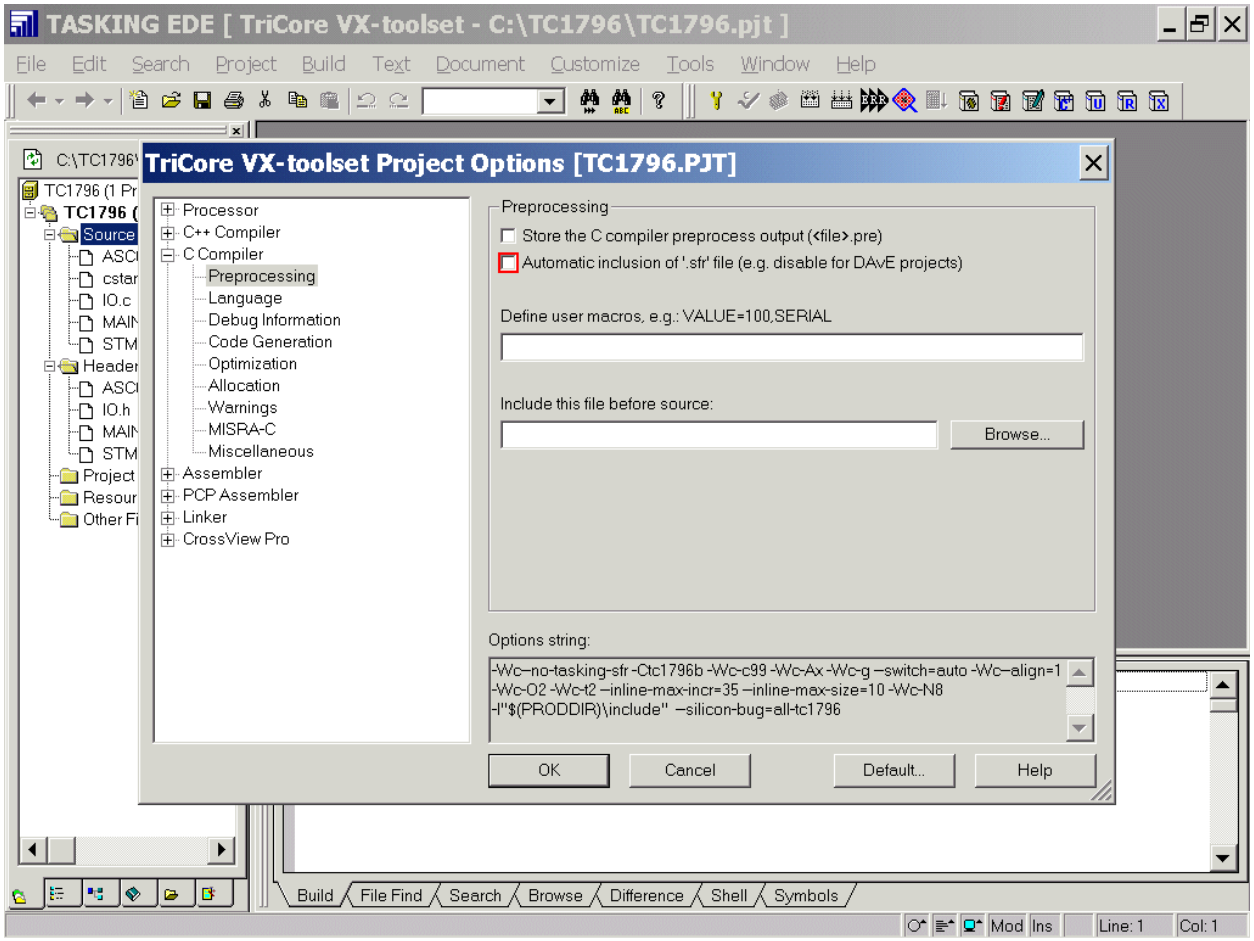
Processor: Processor Definition: Target processor: select TC1796B



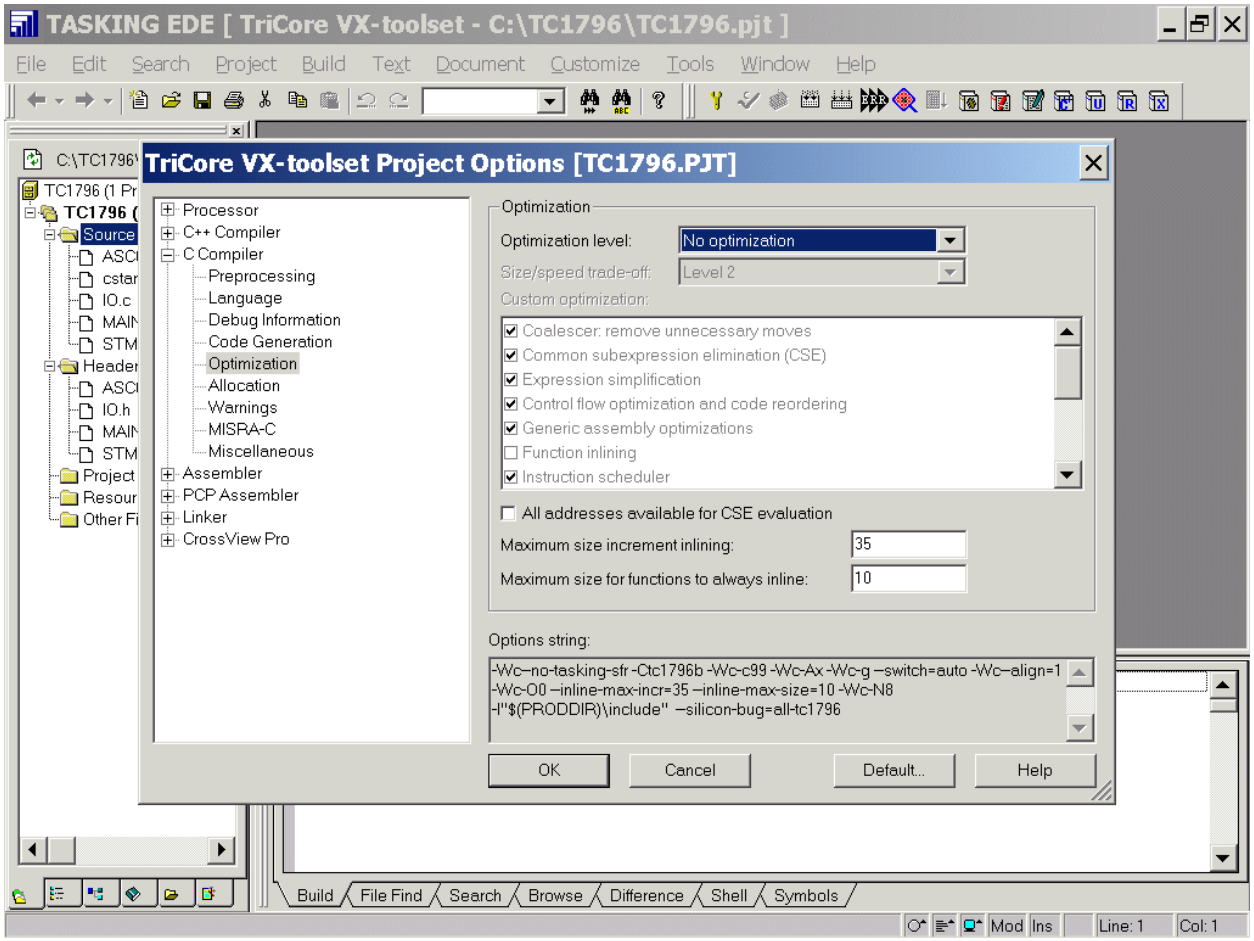
Processor: Bypasses: CPU Functional Problem Bypasses: click ✓ All bypasses TC179X



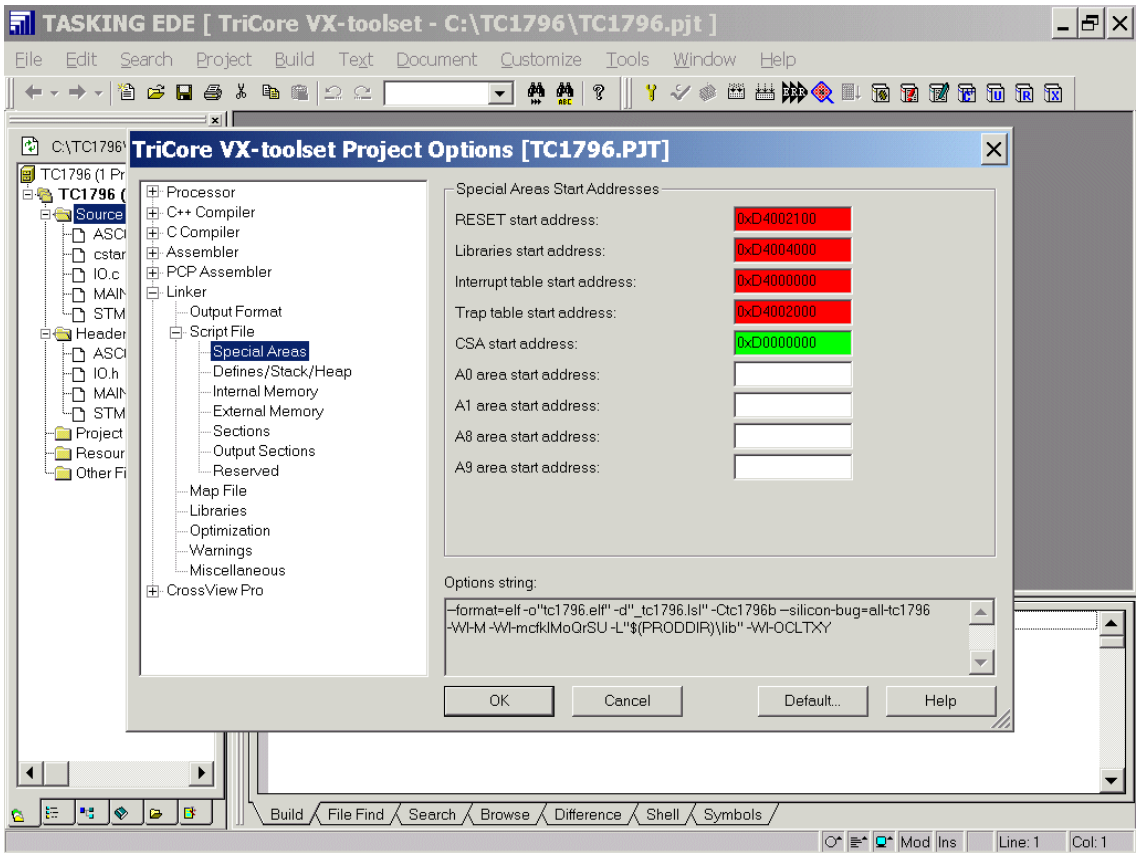
C Compiler: Preprocessing: deactivate ☐ Automatic inclusion of .sfr file

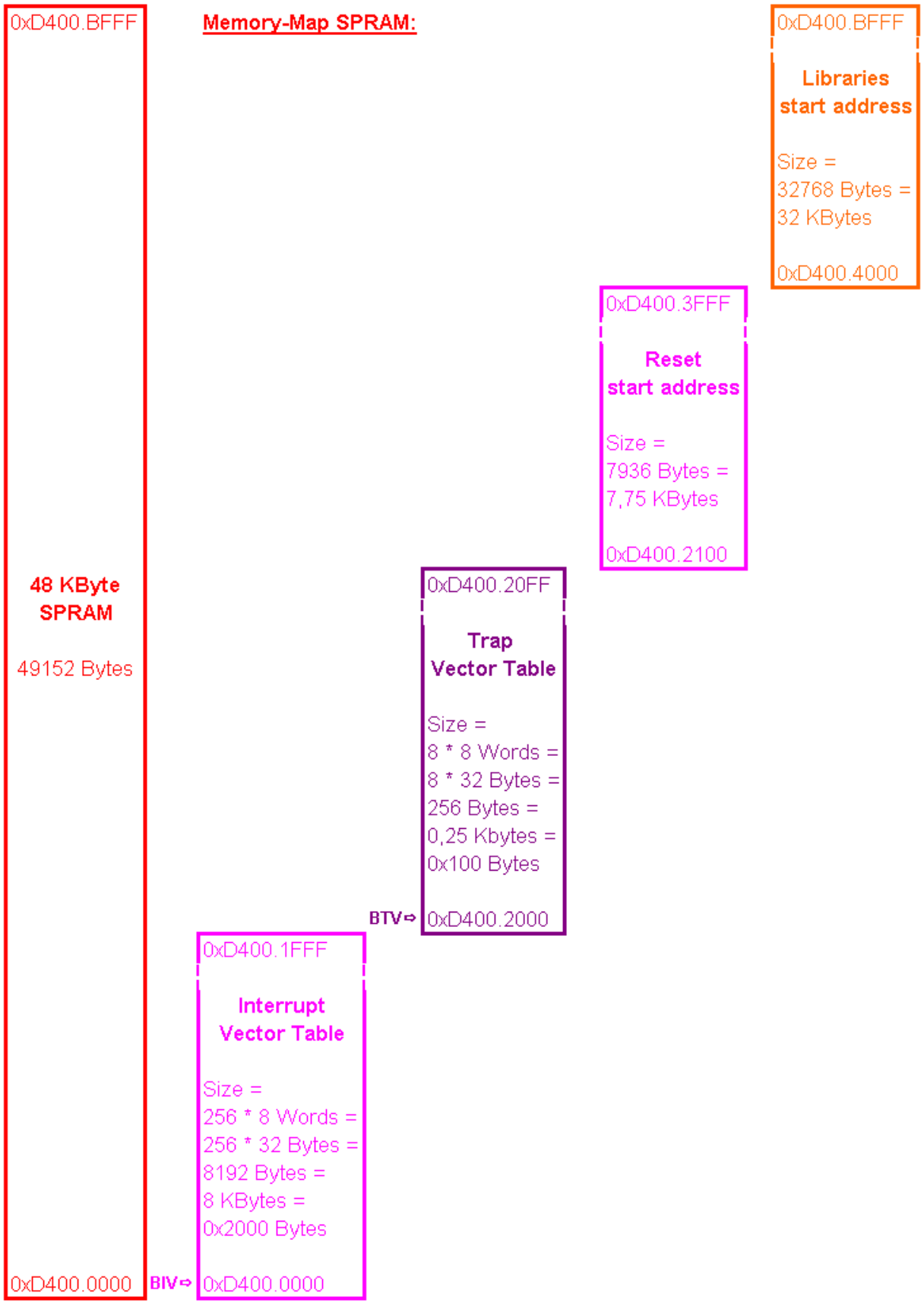


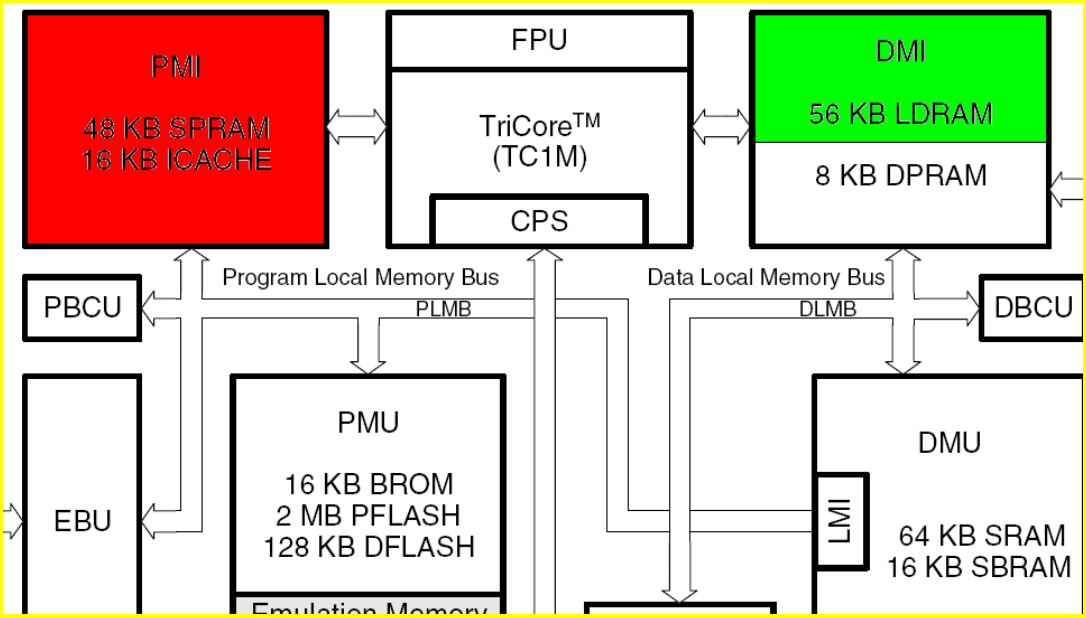
C Compiler: Optimization: Optimization level: select No optimization



Linker: Script File: Special Areas: RESET start address: insert 0xD4002100 (SPRAM)
Linker: Script File: Special Areas: Libraries start address: insert 0xD4004000 (SPRAM)
Linker: Script File: Special Areas: Interrupt table start address: insert 0xD4000000 (SPRAM)
Linker: Script File: Special Areas: Trap table start address: insert 0xD4002000 (SPRAM)
Linker: Script File: Special Areas: CSA start address: insert 0xD0000000 (LDRAM)




Memory-Map SPRAM:
0xD400.BFFF
48 KByte SPRAM
49152 Bytes
0xD400.0000
BIV⇨
0xD400.0000
Interrupt Vector Table
Size =
256 * 8 Words =
256 * 32 Bytes =
8192 Bytes =
8 KBytes =
0x2000 Bytes
0xD400.1FFF
BTV⇨
0xD400.2000
Trap Vector Table
Size =
8 * 8 Words =
8 * 32 Bytes =
256 Bytes =
0,25 Kbytes =
0x100 Bytes
0xD400.20FF
Reset start address
Size =
7936 Bytes =
7,75 KBytes
0xD400.2100
0xD400.3FFF
Libraries start address
Size =
32768 Bytes =
32 KBytes
0xD400.4000
0xD400.BFFF



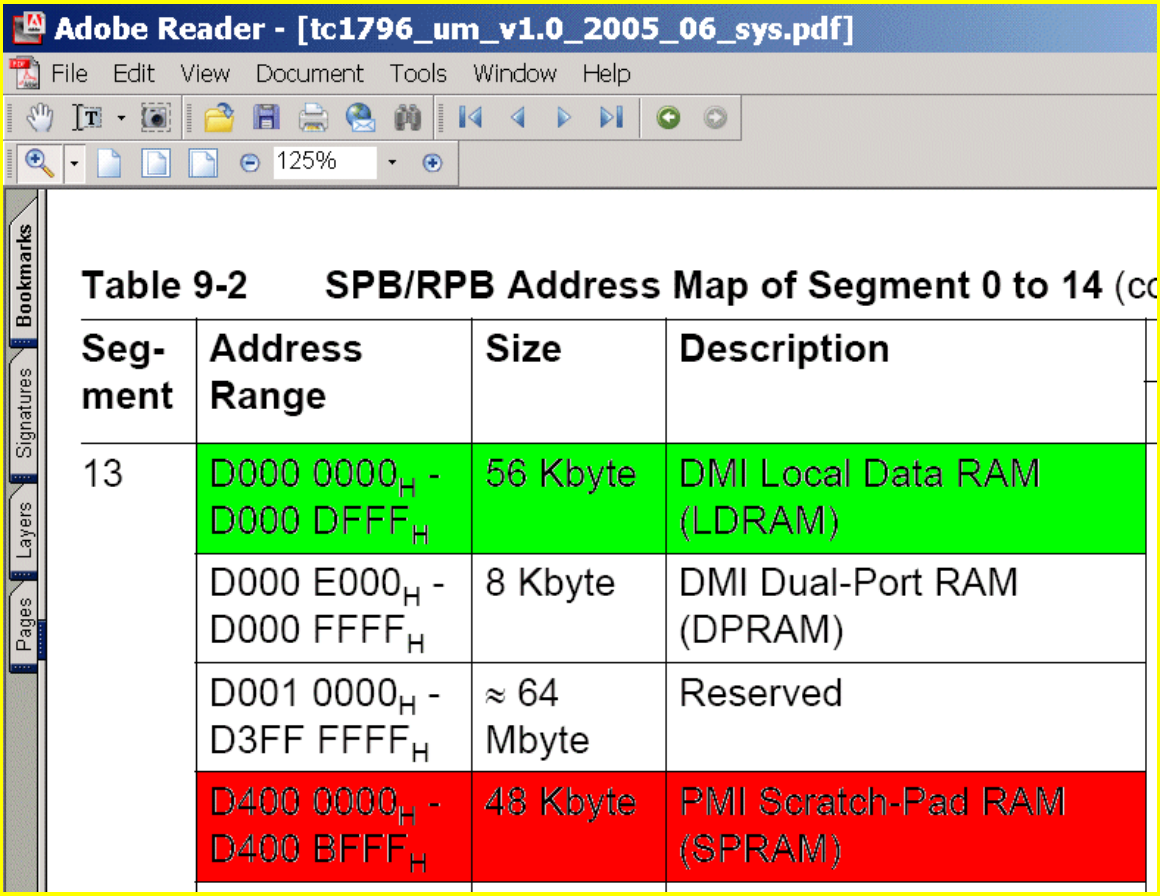
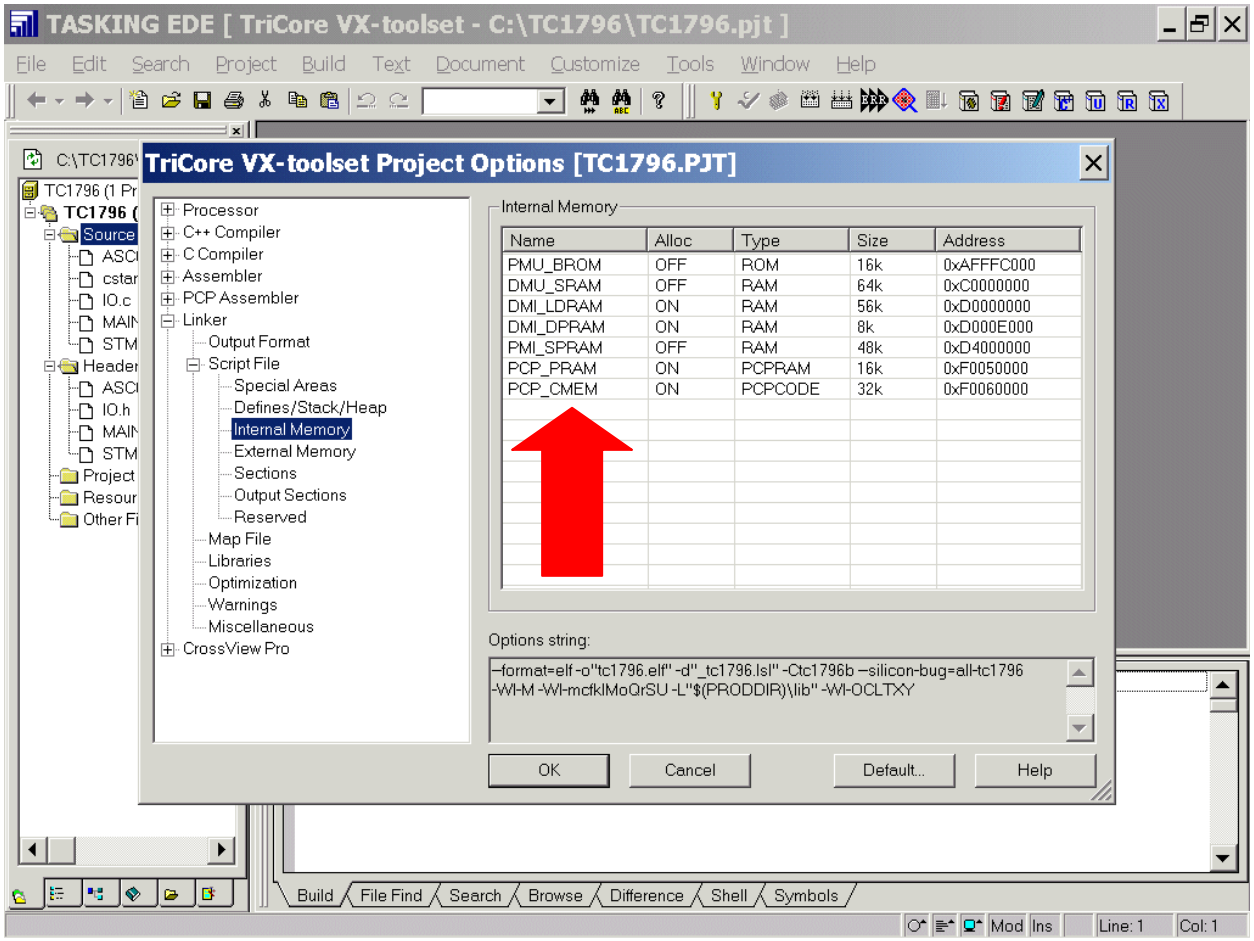
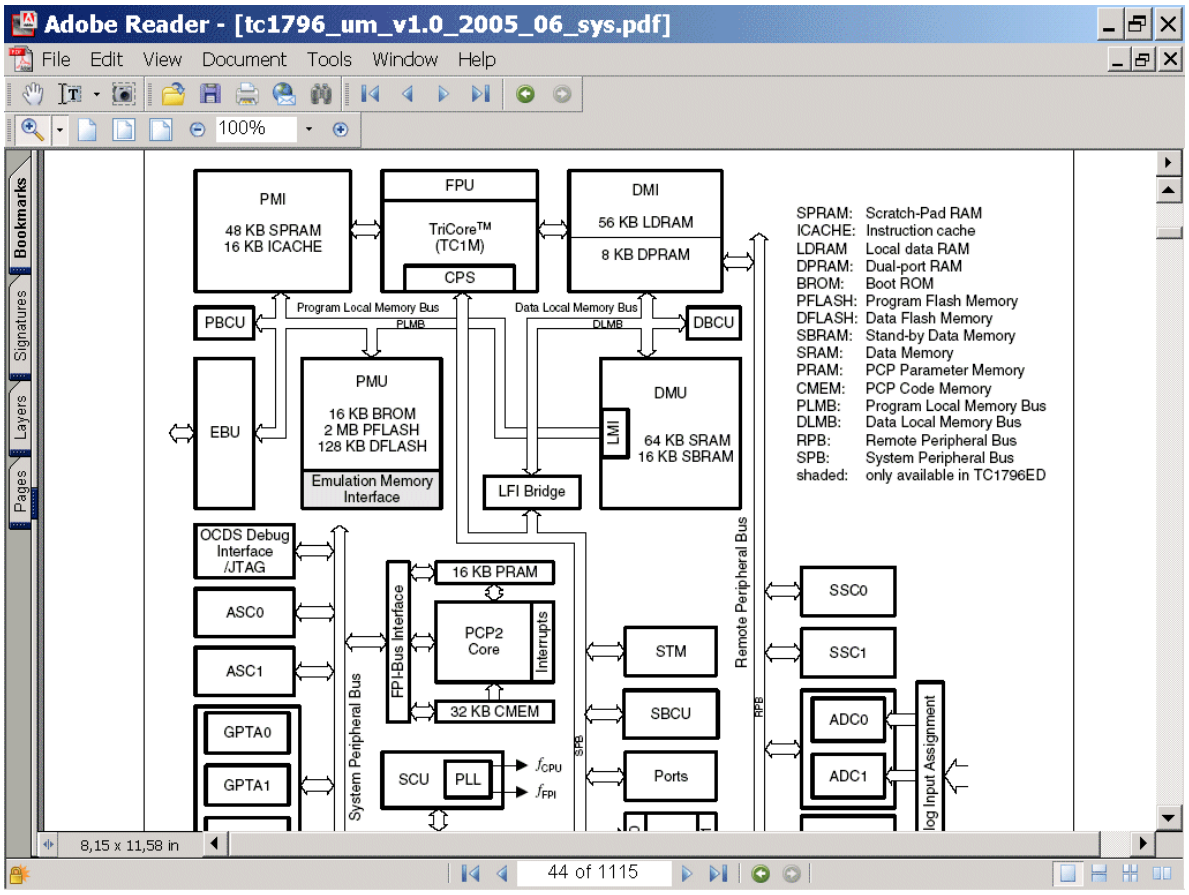


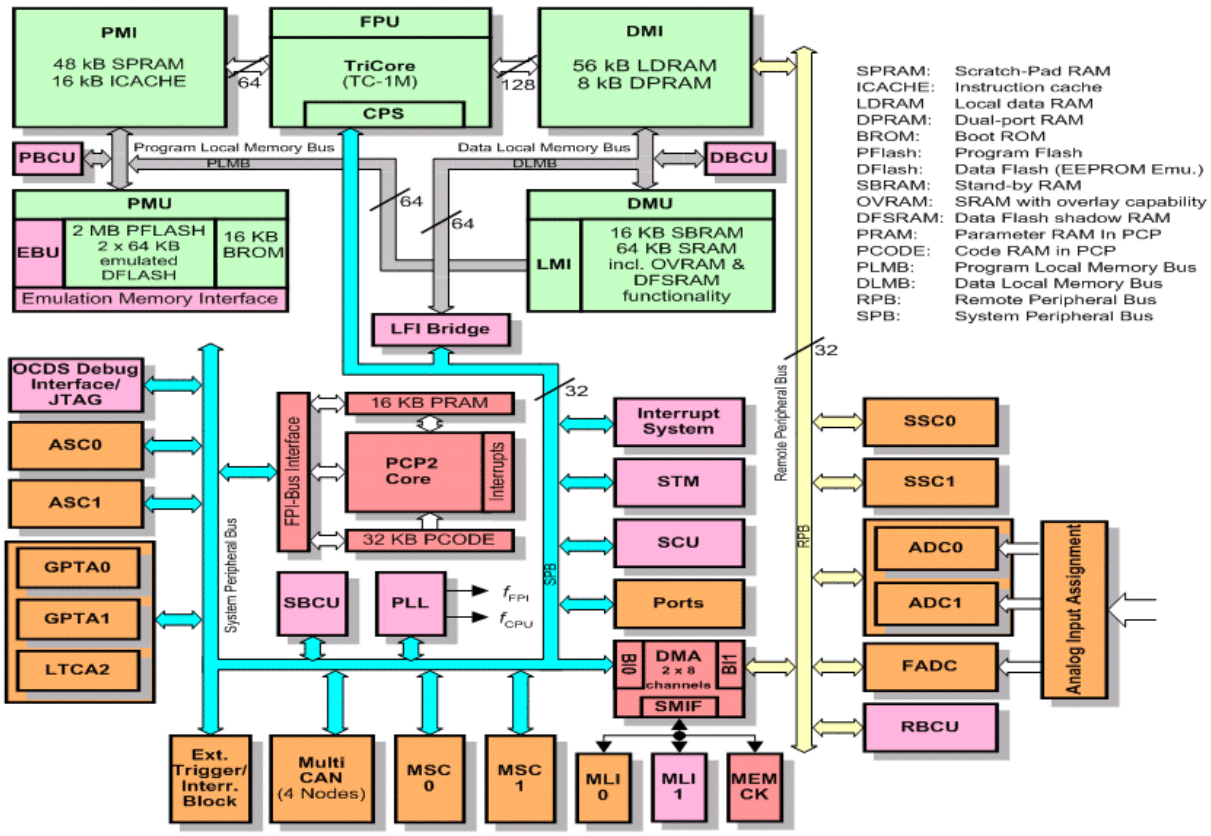
Table 9-2 SPB/RPB Address Map of Segment 0 to 14 (cc

Seg-ment	Address Range	Size	Description
13	$D000\ 0000_H$ - $D000\ DFFF_H$	56 Kbyte	DMI Local Data RAM (LDRAM)
	$D000\ E000_H$ - $D000\ FFFF_H$	8 Kbyte	DMI Dual-Port RAM (DPRAM)
	$D001\ 0000_H$ - $D3FF\ FFFF_H$	≈ 64 Mbyte	Reserved
	$D400\ 0000_H$ - $D400\ BFFF_H$	48 Kbyte	PMI Scratch-Pad RAM (SPRAM)

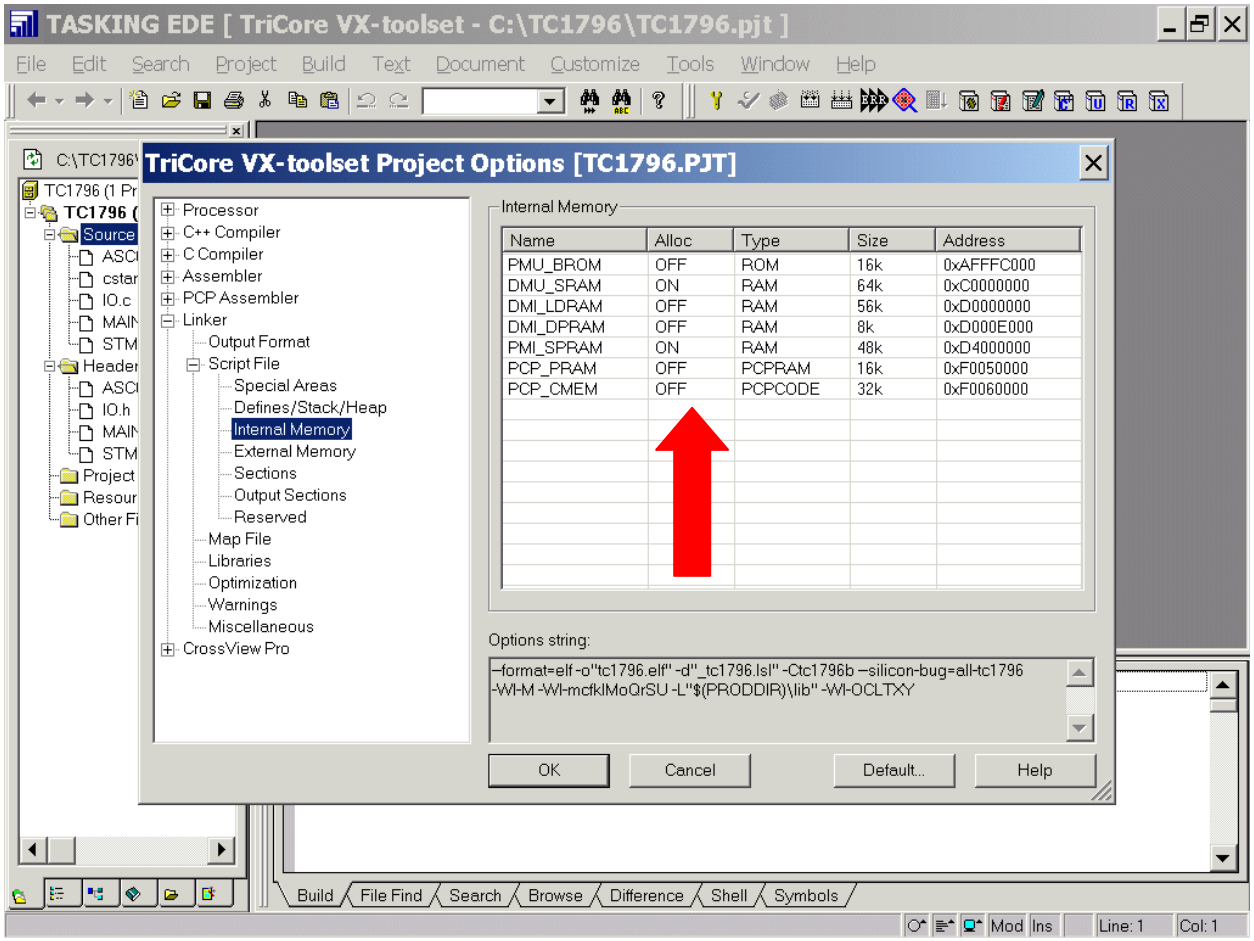
Linker: Script File: Internal Memory: change from brom to PMU_BROM
Linker: Script File: Internal Memory: change from lmbram to DMU_SRAM
Linker: Script File: Internal Memory: change from dsram to DMI_LDRAM
Linker: Script File: Internal Memory: change from dpram to DMI_DPRAM
Linker: Script File: Internal Memory: change from csram 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



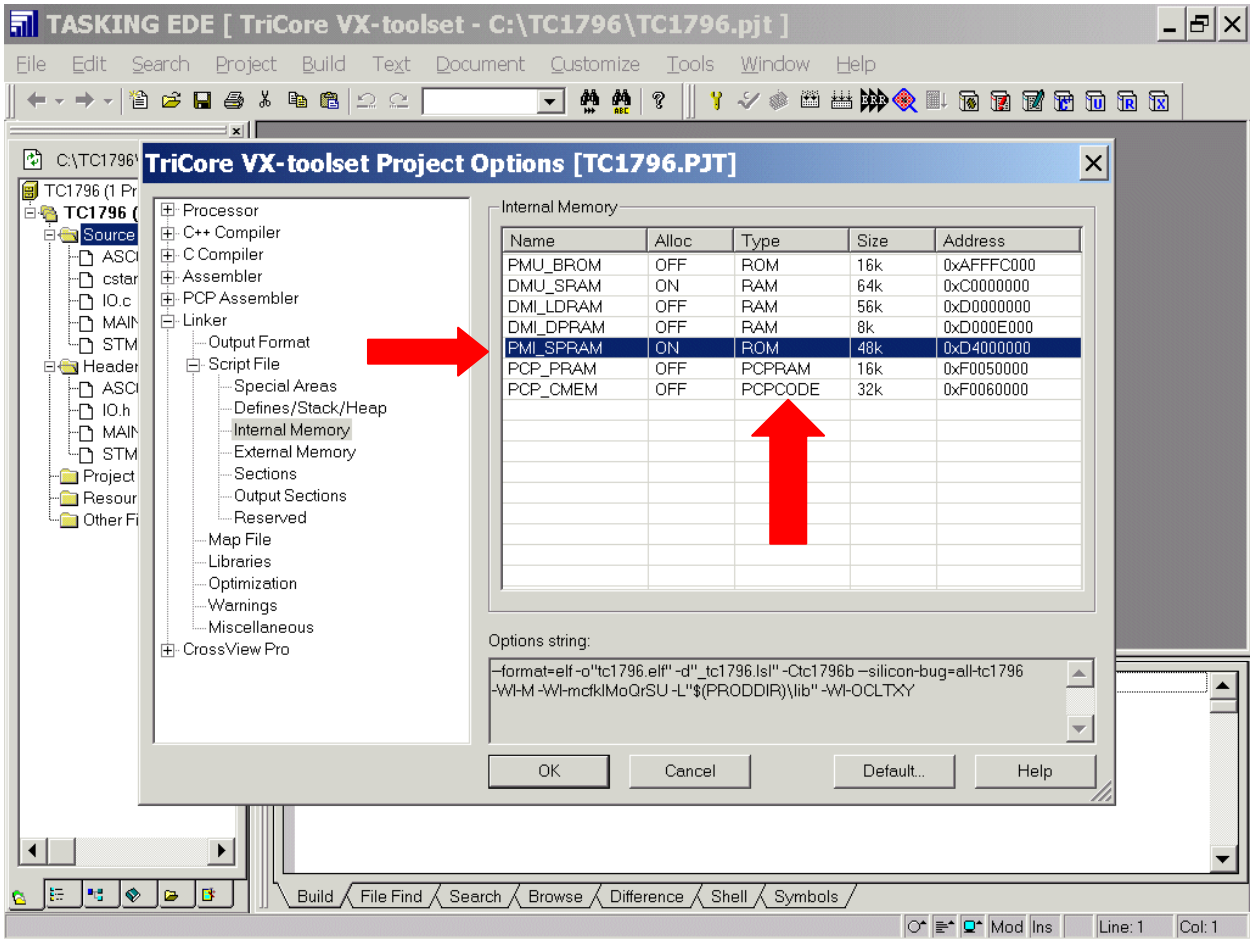

Adobe Reader - [tc1796_um_v1.0_2005_06_sys.pdf]
File Edit View Document Tools Window Help
100%
Bookmarks
Signatures
Layers
Pages
PMI
48 KB SPRAM
16 KB ICACHE
FPU
TriCore™
(TC1M)
CPS
DMI
56 KB LDRAM
8 KB DPRAM
SPRAM: Scratch-Pad RAM
ICACHE: Instruction cache
LDRAM Local data RAM
DPRAM: Dual-port RAM
BROM: Boot ROM
PFLASH: Program Flash Memory
DFLASH: Data Flash Memory
SBRAM: Stand-by Data Memory
SRAM: Data Memory
PRAM: PCP Parameter Memory
CMEM: PCP Code Memory
PLMB: Program Local Memory Bus
DLMB: Data Local Memory Bus
RPB: Remote Peripheral Bus
SPB: System Peripheral Bus
shaded: only available in TC1796ED
PBCU
Program Local Memory Bus
PLMB
Data Local Memory Bus
DLMB
DBCU
EBU
PMU
16 KB BROM
2 MB PFLASH
128 KB DFLASH
Emulation Memory Interface
DMU
LMI
64 KB SRAM
16 KB SBRAM
LFI Bridge
OCDS Debug Interface /JTAG
ASC0
ASC1
GPTA0
GPTA1
16 KB PRAM
PCP2 Core
Interrupts
32 KB CMEM
FPI-Bus Interface
System Peripheral Bus
SCU
PLL
STM
SBCU
Ports
SSC0
SSC1
ADC0
ADC1
Remote Peripheral Bus
RPB
SPB
log Input Assignment
8,15 x 11,58 in
44 of 1115


PMI
48 kB SPRAM
16 kB ICACHE
64
FPU
TriCore
(TC-1M)
CPS
128
DMI
56 kB LDRAM
8 kB DPRAM
SPRAM: Scratch-Pad RAM
ICACHE: Instruction cache
LDRAM Local data RAM
DPRAM: Dual-port RAM
BROM: Boot ROM
PFlash: Program Flash
DFlash: Data Flash (EEPROM Emu.)
SBRAM: Stand-by RAM
OVRAM: SRAM with overlay capability
DFSRAM: Data Flash shadow RAM
PRAM: Parameter RAM in PCP
PCODE: Code RAM in PCP
PLMB: Program Local Memory Bus
DLMB: Data Local Memory Bus
RPB: Remote Peripheral Bus
SPB: System Peripheral Bus
PBCU
Program Local Memory Bus
PLMB
Data Local Memory Bus
DLMB
DBCU
PMU
EBU
2 MB PFLASH
2 x 64 KB
emulated
DFLASH
16 KB
BROM
Emulation Memory Interface
64
64
DMU
LMI
16 KB SBRAM
64 KB SRAM
incl. OVRAM &
DFSRAM
functionality
LFI Bridge
32
OCDS Debug Interface/ JTAG
ASC0
ASC1
GPTA0
GPTA1
LTCA2
16 KB PRAM
PCP2 Core
Interrupts
32 KB PCODE
FPI-Bus Interface
System Peripheral Bus
SBCU
PLL
SPB
Interrupt System
STM
SCU
Ports
DMA
2 x 8
channels
SMIF
BIO
BII
Remote Peripheral Bus
RPB
SSC0
SSC1
ADC0
ADC1
FADC
RBCU
Analog Input Assignment
Ext. Trigger/ Interr. Block
Multi CAN (4 Nodes)
MSC 0
MSC 1
MLI 0
MLI 1
MEM CK

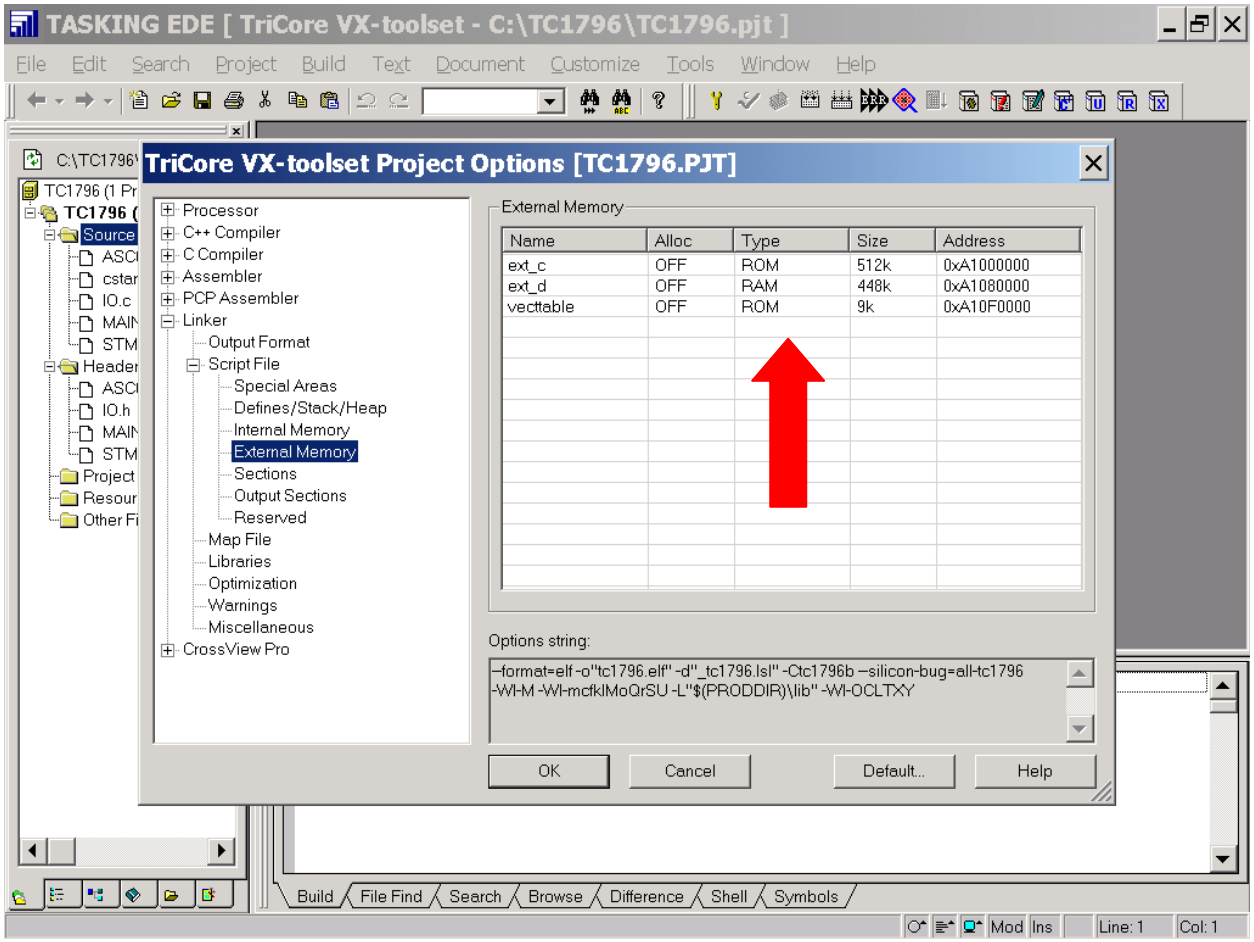
Linker: Script File: Internal Memory: PMU_BROM: Alloc: select OFF
Linker: Script File: Internal Memory: DMU_SRAM: Alloc: select ON!!!
Linker: Script File: Internal Memory: DMI_LDRAM: Alloc: select OFF
Linker: Script File: Internal Memory: DMI_DPRAM: Alloc: select OFF
Linker: Script File: Internal Memory: PMI_SPRAM: Alloc: select ON!!!
Linker: Script File: Internal Memory: PCP_PRAM: Alloc: select OFF
Linker: Script File: Internal Memory: PCP_CMEM: Alloc: select OFF



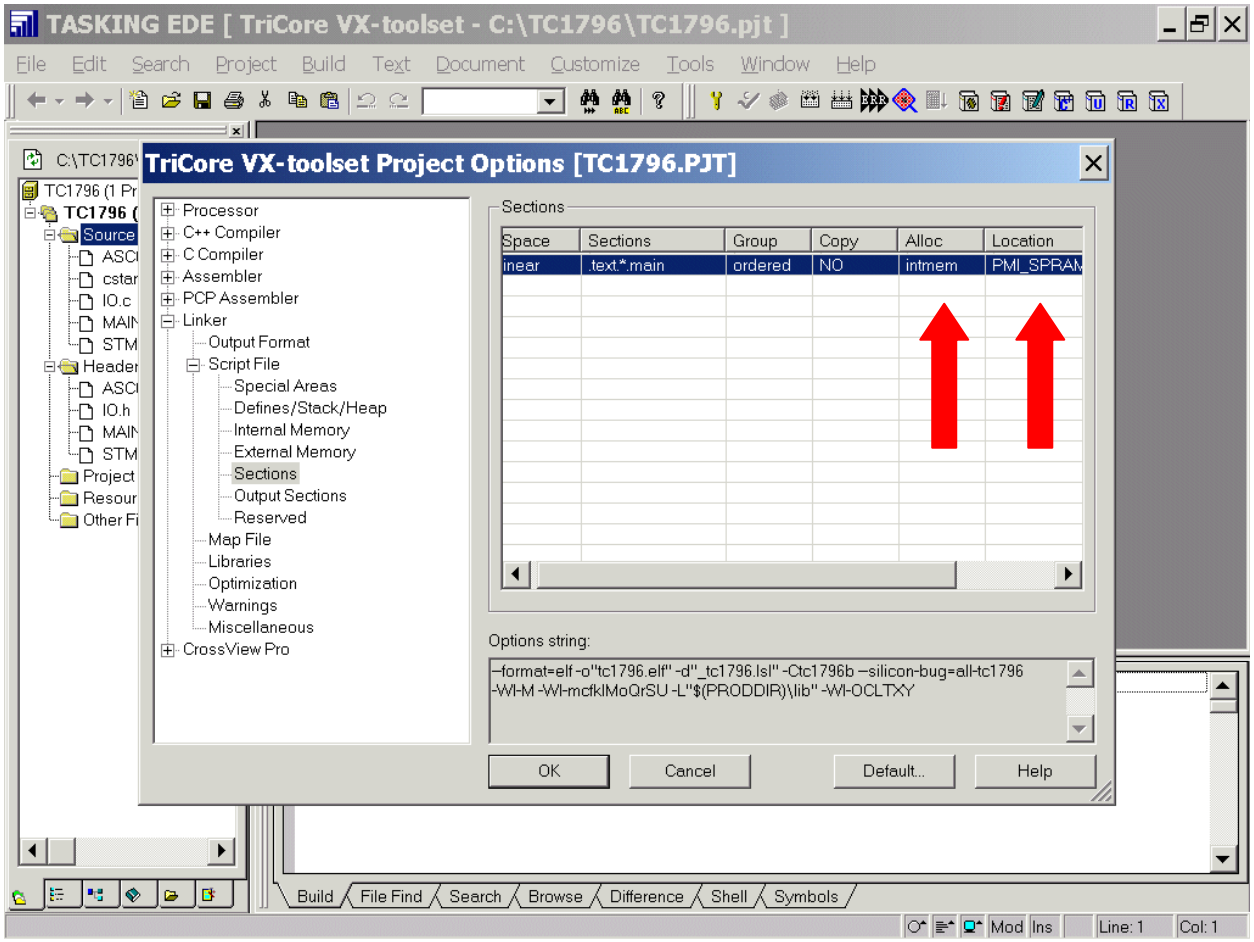
Linker: Script File: Internal Memory: PMI_SPRAM: Type: select ROM 



Linker: Script File: External Memory: ext_c: Alloc: select OFF
Linker: Script File: External Memory: ext_d: Alloc: select OFF
Linker: Script File: External Memory: vectable: Alloc: select OFF

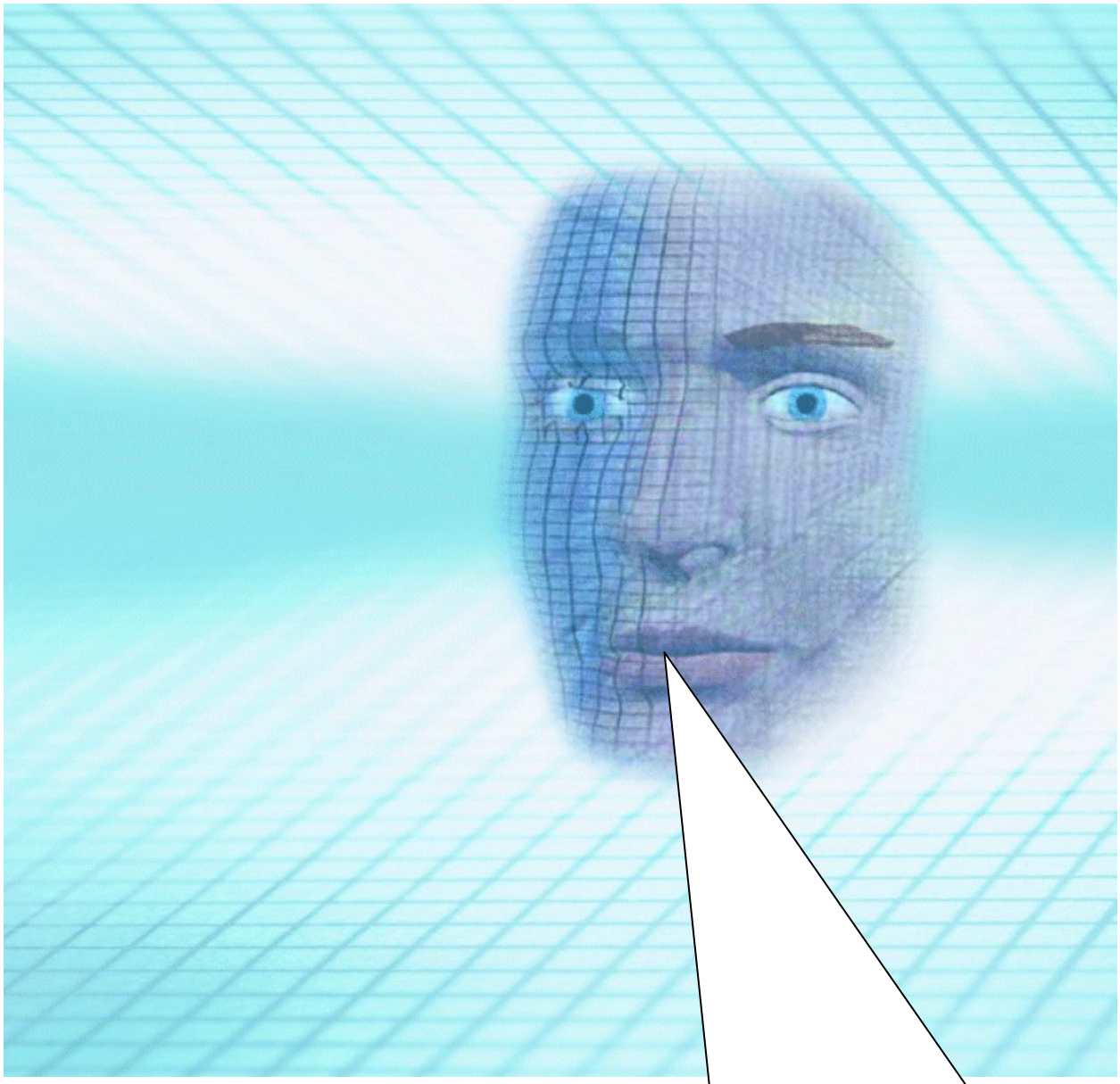


Linker: Script File: Sections: linear: Alloc: select intmem
Linker: Script File: Sections: linear: Location: insert PMI_SPRAM



OK

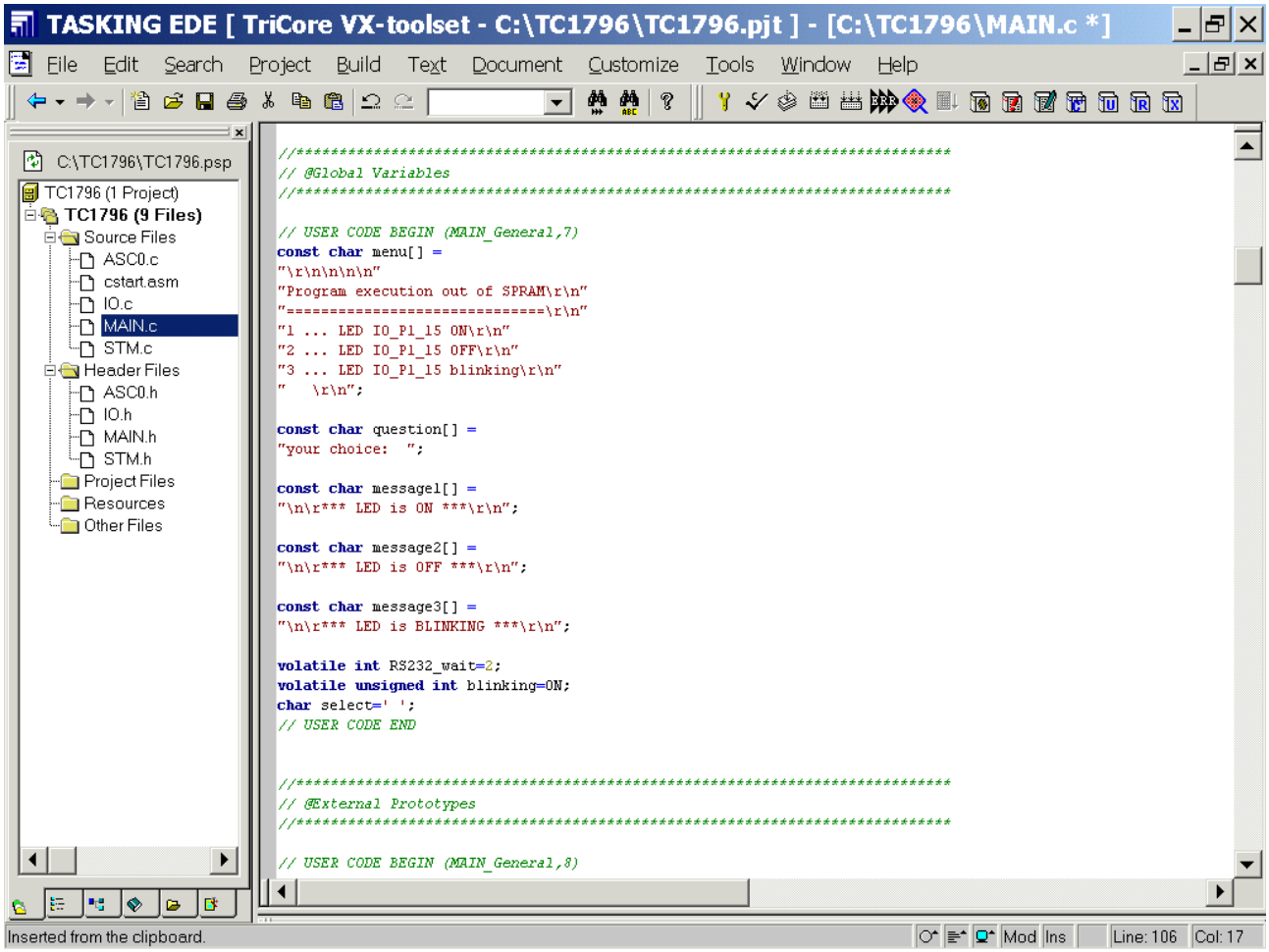
Insert your application specific program:



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!

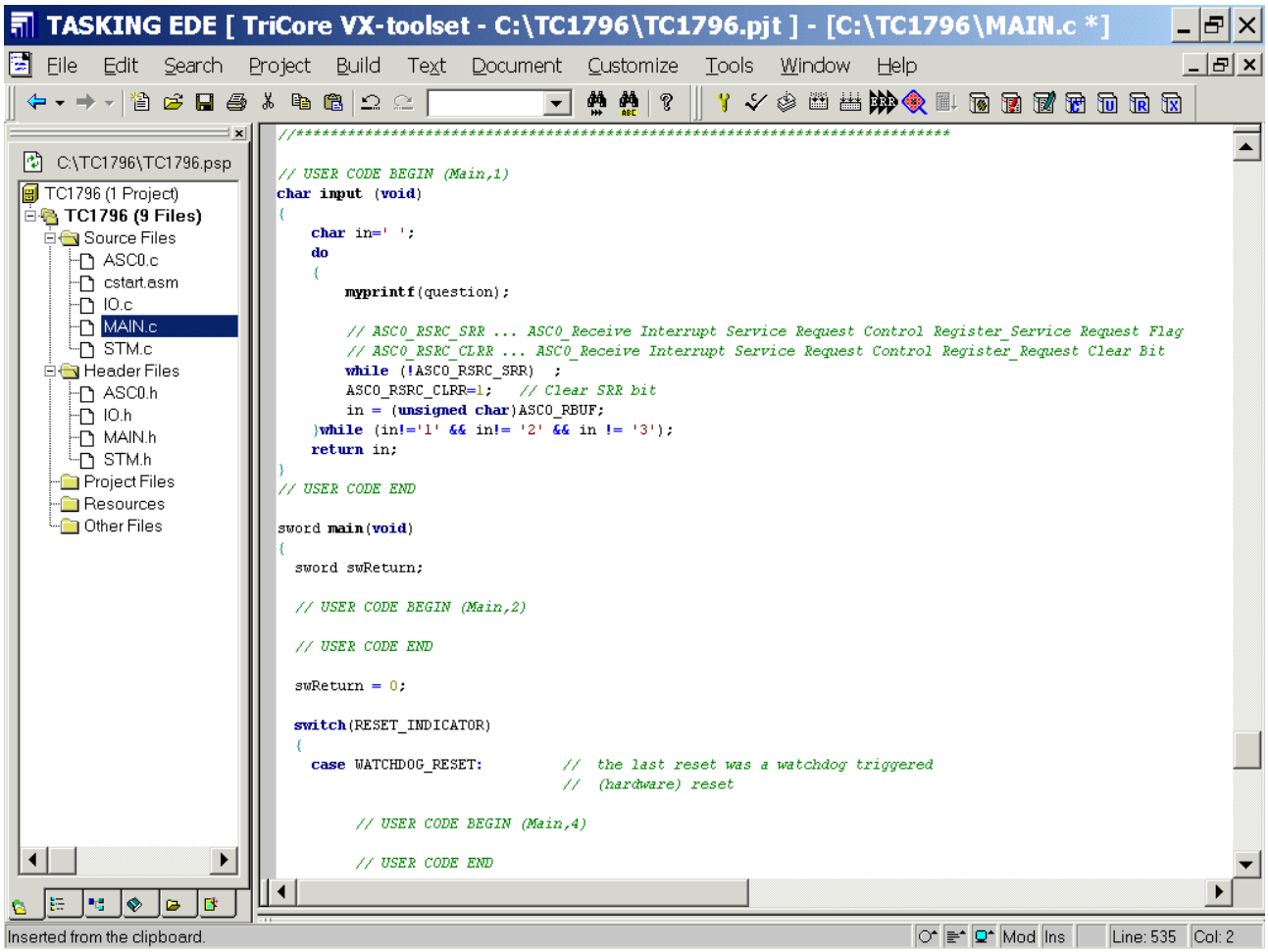
Double click: Main.c insert User Code (Global Variables):

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of SPRAM\r\n"
"==============================\r\n"
"1 ... LED IO_P1_15 ON\r\n"
"2 ... LED IO_P1_15 OFF\r\n"
"3 ... LED IO_P1_15 blinking\r\n"
"   \r\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:\TC1796\TC1796.pjt] - [C:\TC1796\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1796\TC1796.psp
TC1796 (1 Project)
TC1796 (9 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
Project Files
Resources
Other Files
//***
// @Global Variables
//***
// USER CODE BEGIN (MAIN_General,7)
const char menu[] =
"\r\n\n\n\n"
"Program execution out of SPRAM\r\n"
"==============================\r\n"
"1 ... LED IO_P1_15 ON\r\n"
"2 ... LED IO_P1_15 OFF\r\n"
"3 ... LED IO_P1_15 blinking\r\n"
" \r\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
//***
// @External Prototypes
//***
// USER CODE BEGIN (MAIN_General,8)
Inserted from the clipboard.
Mod Ins
Line: 106 Col: 17

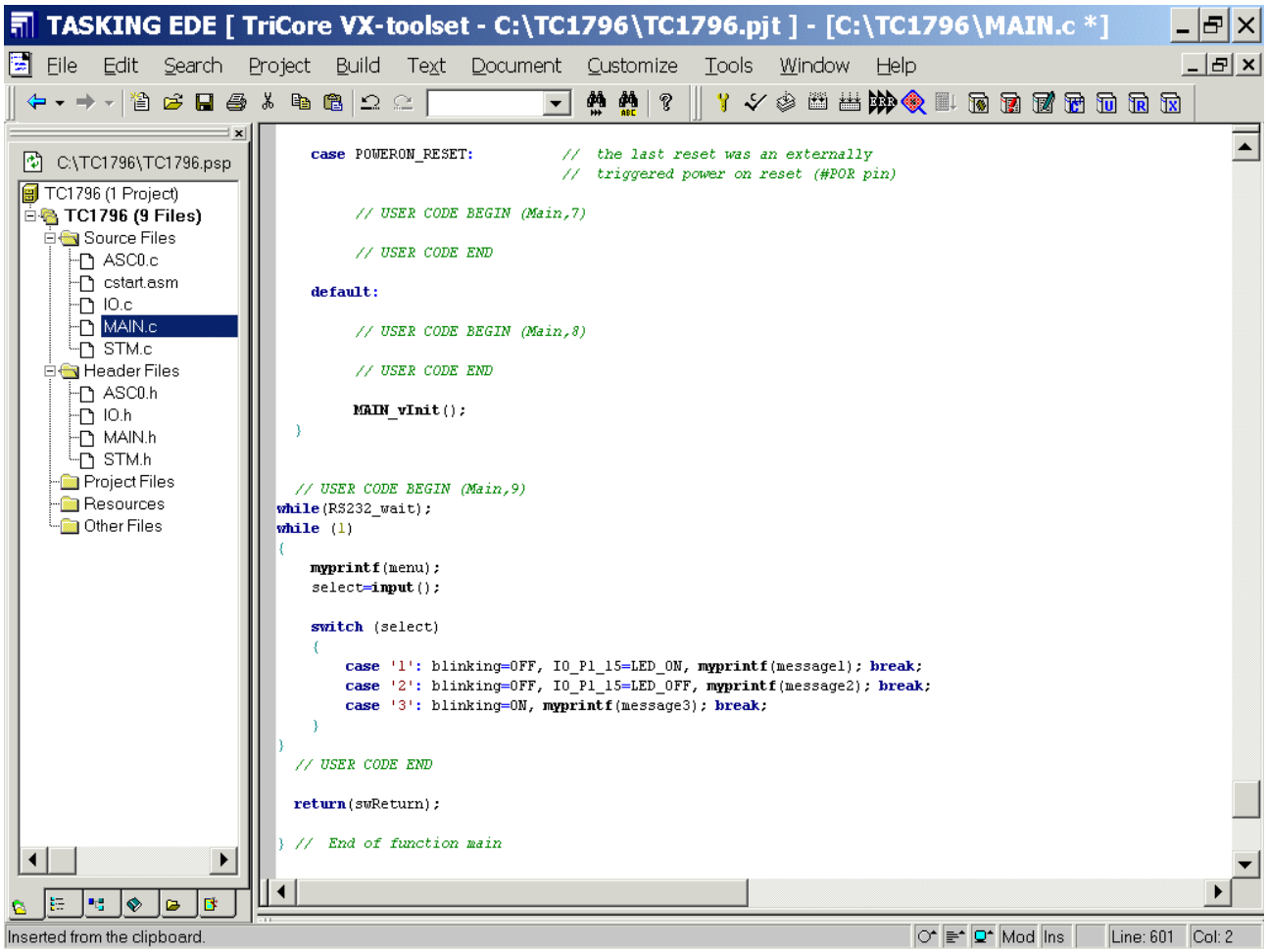
Double click: Main.c: insert User Code (function: input()):

```
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;
}
```



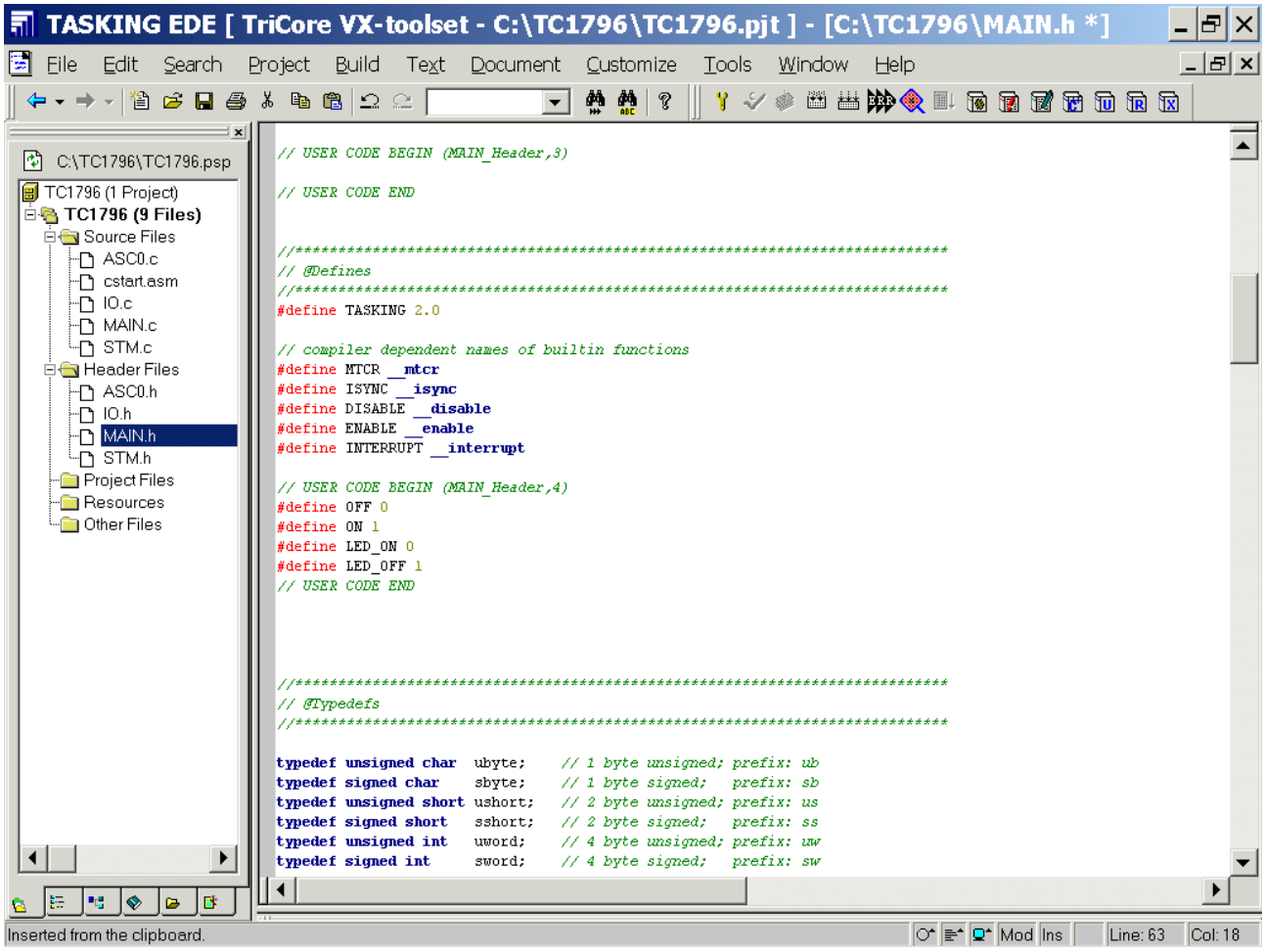
Double click: Main.c: insert User Code:

```
while(RS232_wait);
while (1)
{
   myprintf(menu);
   select=input();

   switch (select)
   {
      case '1': blinking=OFF, IO_P1_15=LED_ON, myprintf(message1); break;
      case '2': blinking=OFF, IO_P1_15=LED_OFF, myprintf(message2); break;
      case '3': blinking=ON, myprintf(message3); break;
   }
}
```



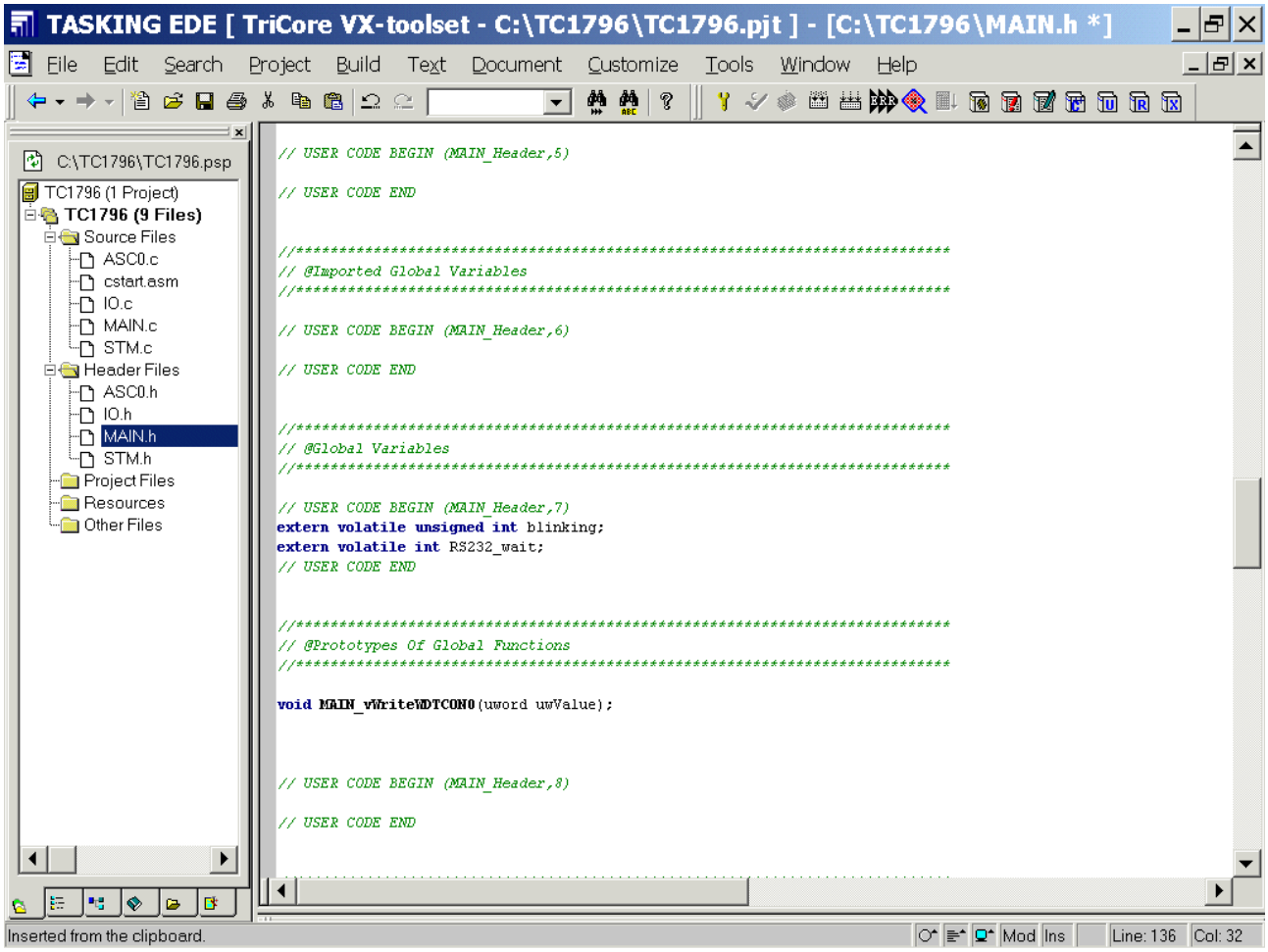
Double click: Main.h and insert the following Defines:

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



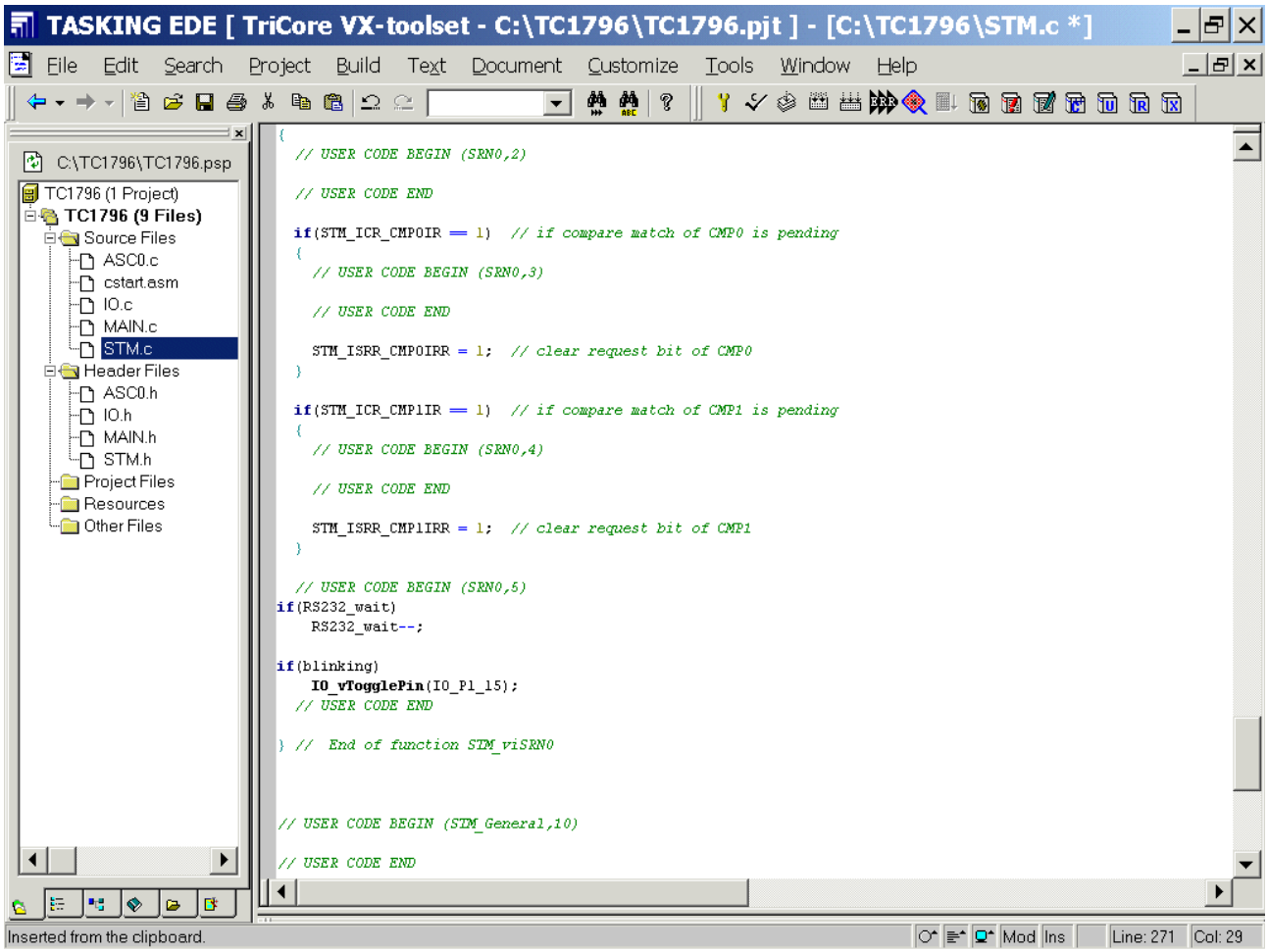
Double click: Main.h and insert Global Variables:

```
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+12500000; // 12.500.000 * 80 ns = 1 s

if(RS232_wait)
   RS232_wait--;

if(blinking)
   IO_vTogglePin(IO_P1_15);
```



Note:
12.500.000 * 80 ns = 1 s

To get an STM-interrupt every 1 second you must change the Compare-Value to "STM_CMP0+=12500000;", because there is no "reload-functionality"!

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 not available for Tasking TriCore tools for the time being.
For the moment, Tasking has only got following “Change Request”:

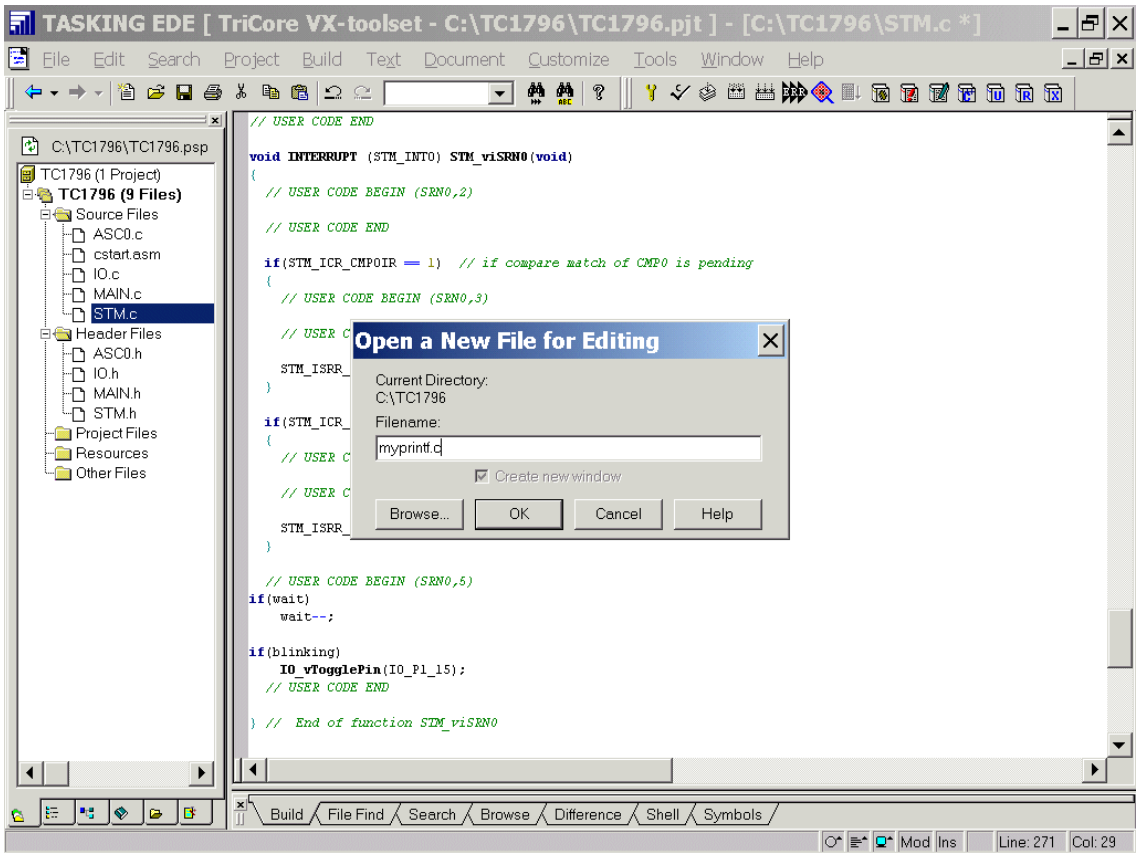
CR32186 CR: Example for _write function implementation using serial = interface

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

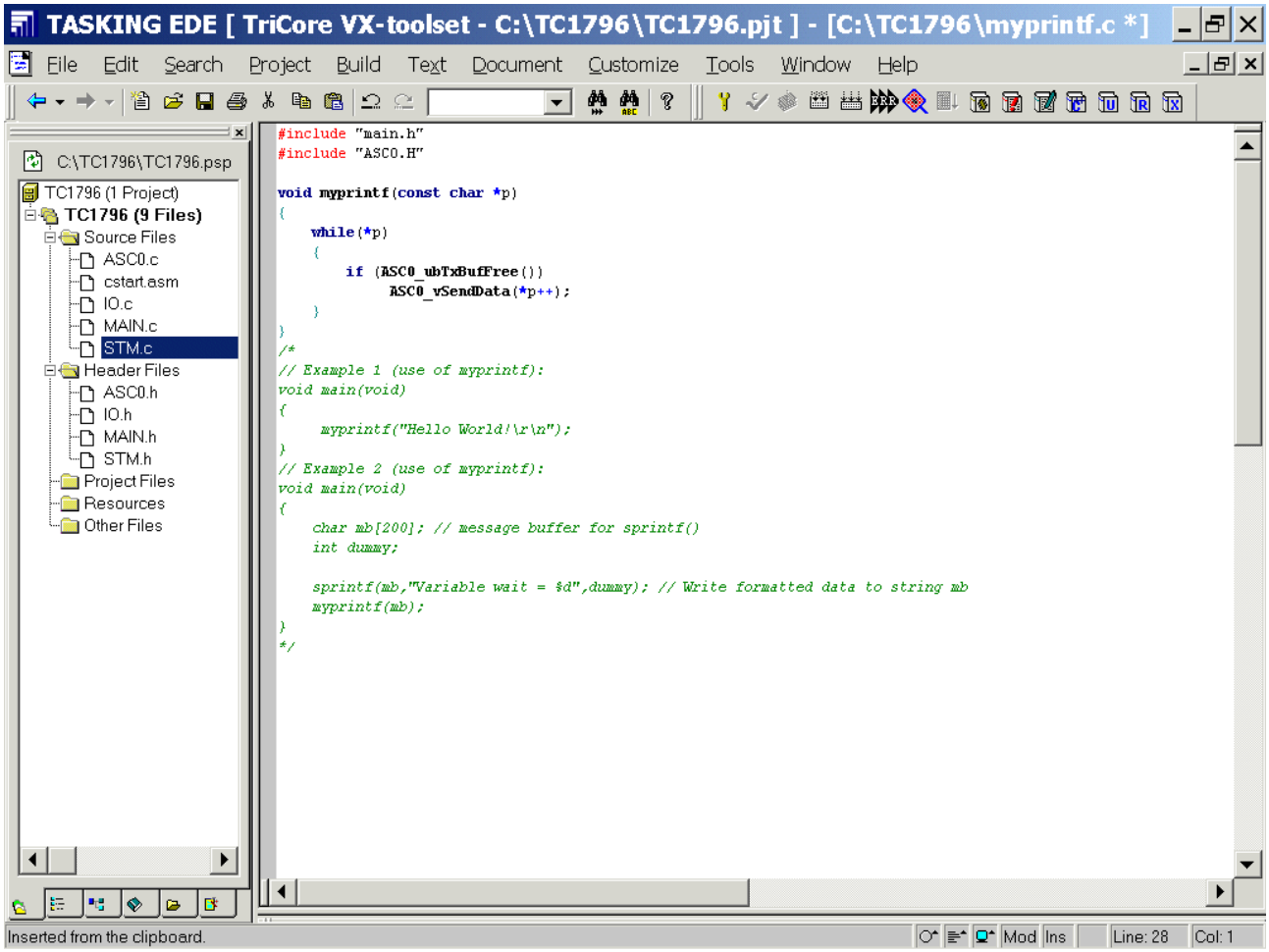
File – New
Insert myprintf.c



OK

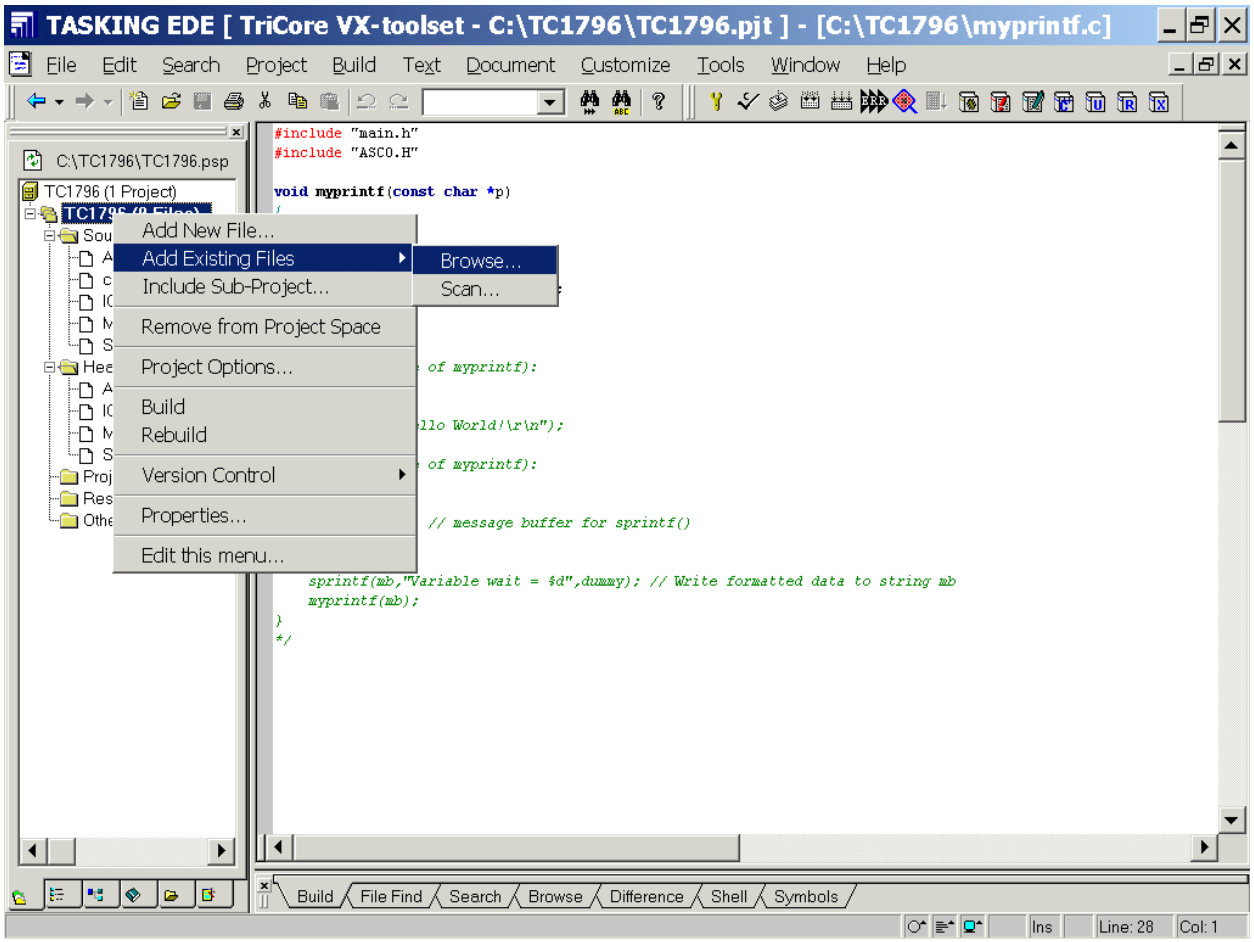
Insert User Code for myprintf():

```
#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);
}
*/
```

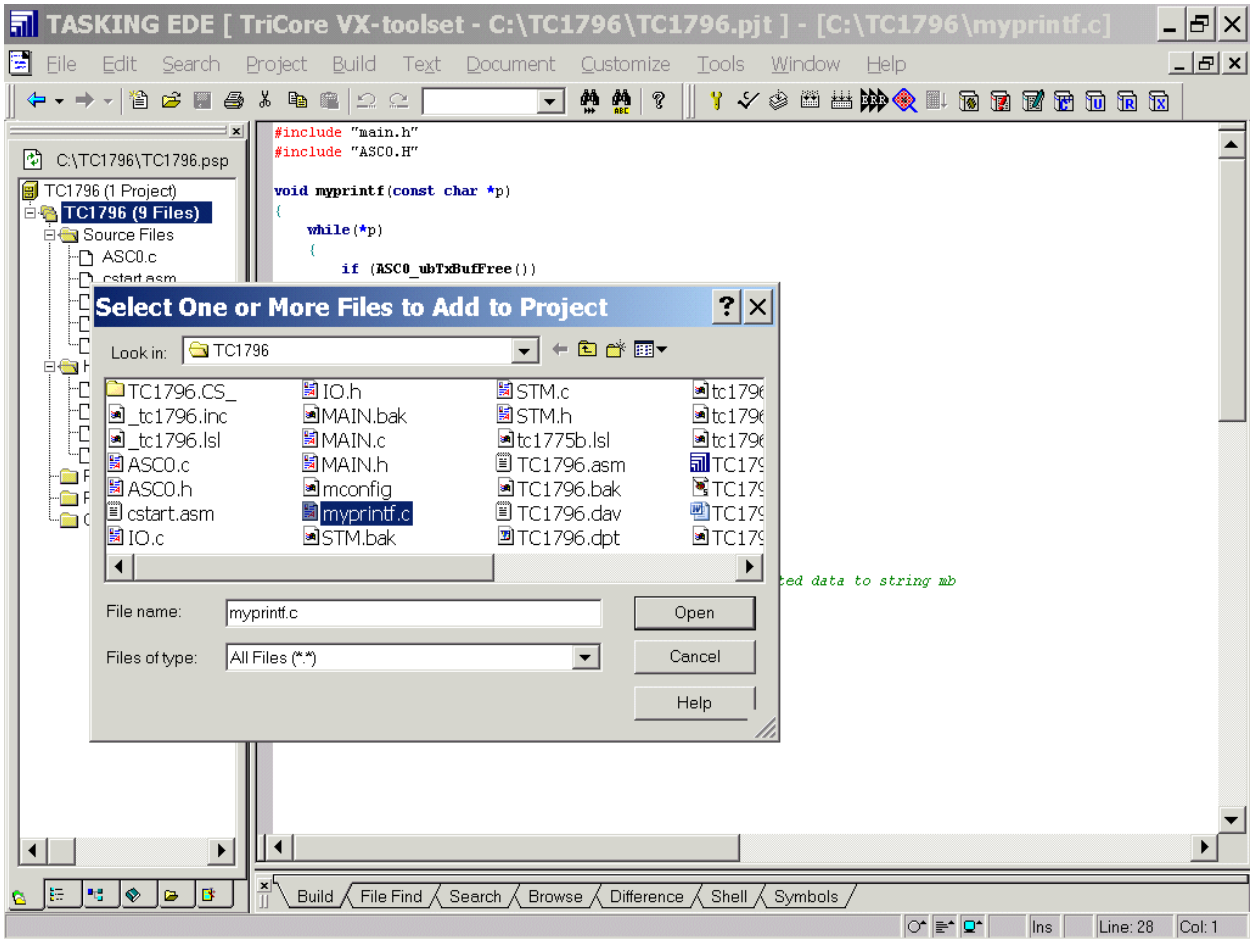

TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\myprintf.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1796\TC1796.psp
TC1796 (1 Project)
TC1796 (9 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
Project 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);
}
*/
Inserted from the clipboard.
Mod Ins Line: 28 Col: 1

File
Save all

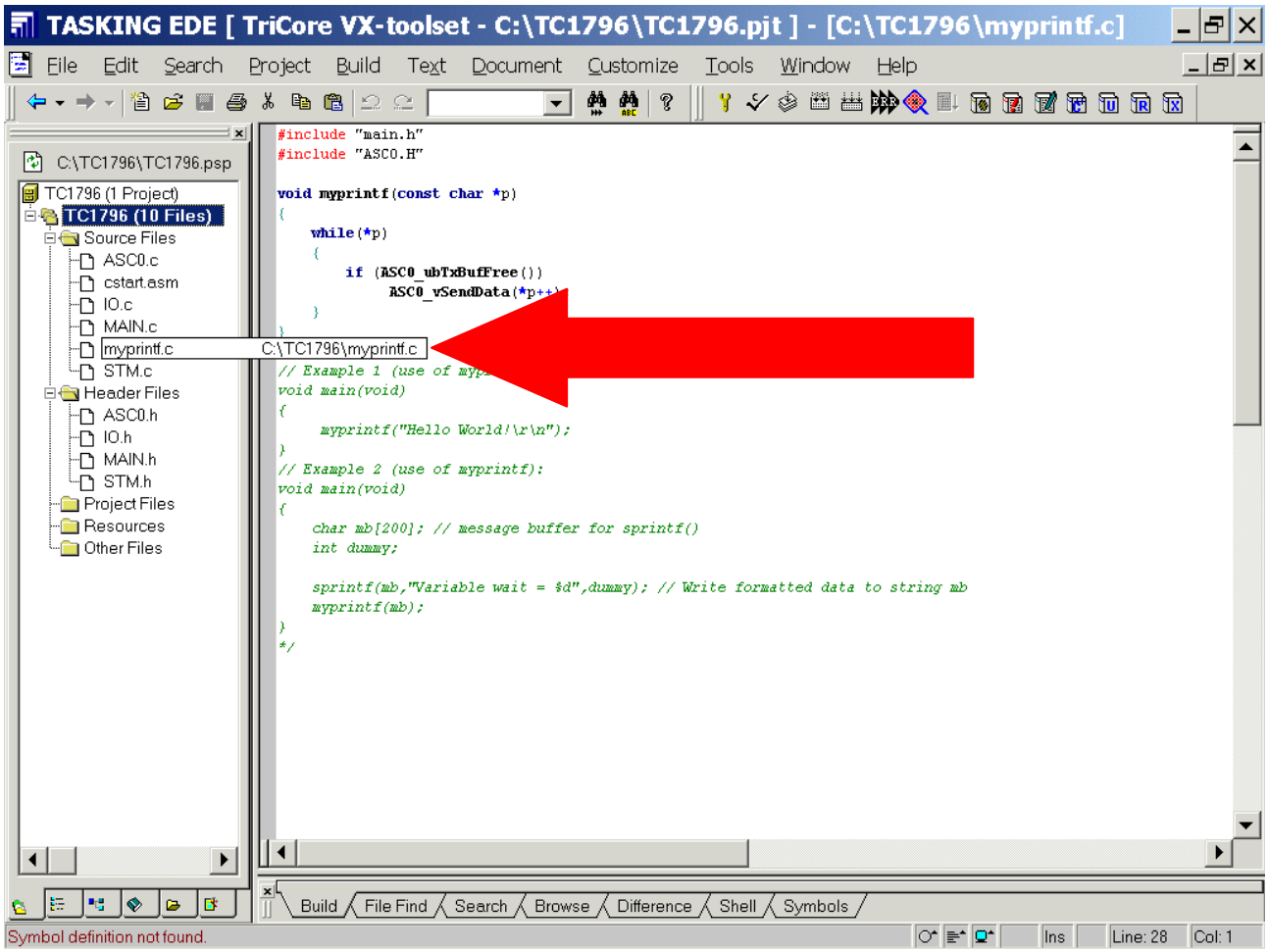
(Project Window File View) – TC1796 (Files) – right mouse button click – Add Existing Files – Browse



Select myprintf.c

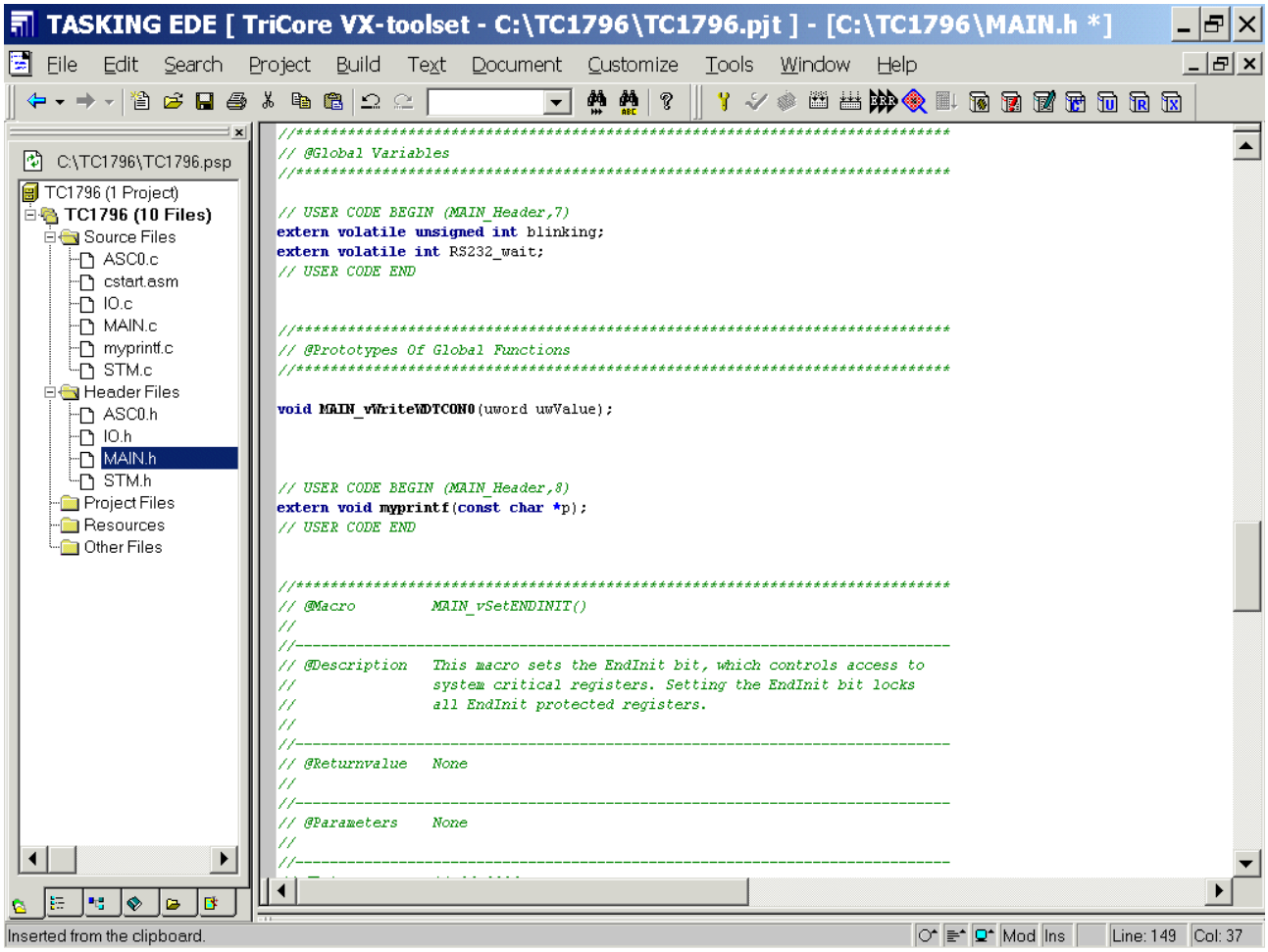


Open - OK


TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\myprintf.c]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1796\TC1796.psp
TC1796 (1 Project)
TC1796 (10 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
C:\TC1796\myprintf.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.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 myp
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
Symbol definition not found.
Ins
Line: 28
Col: 1

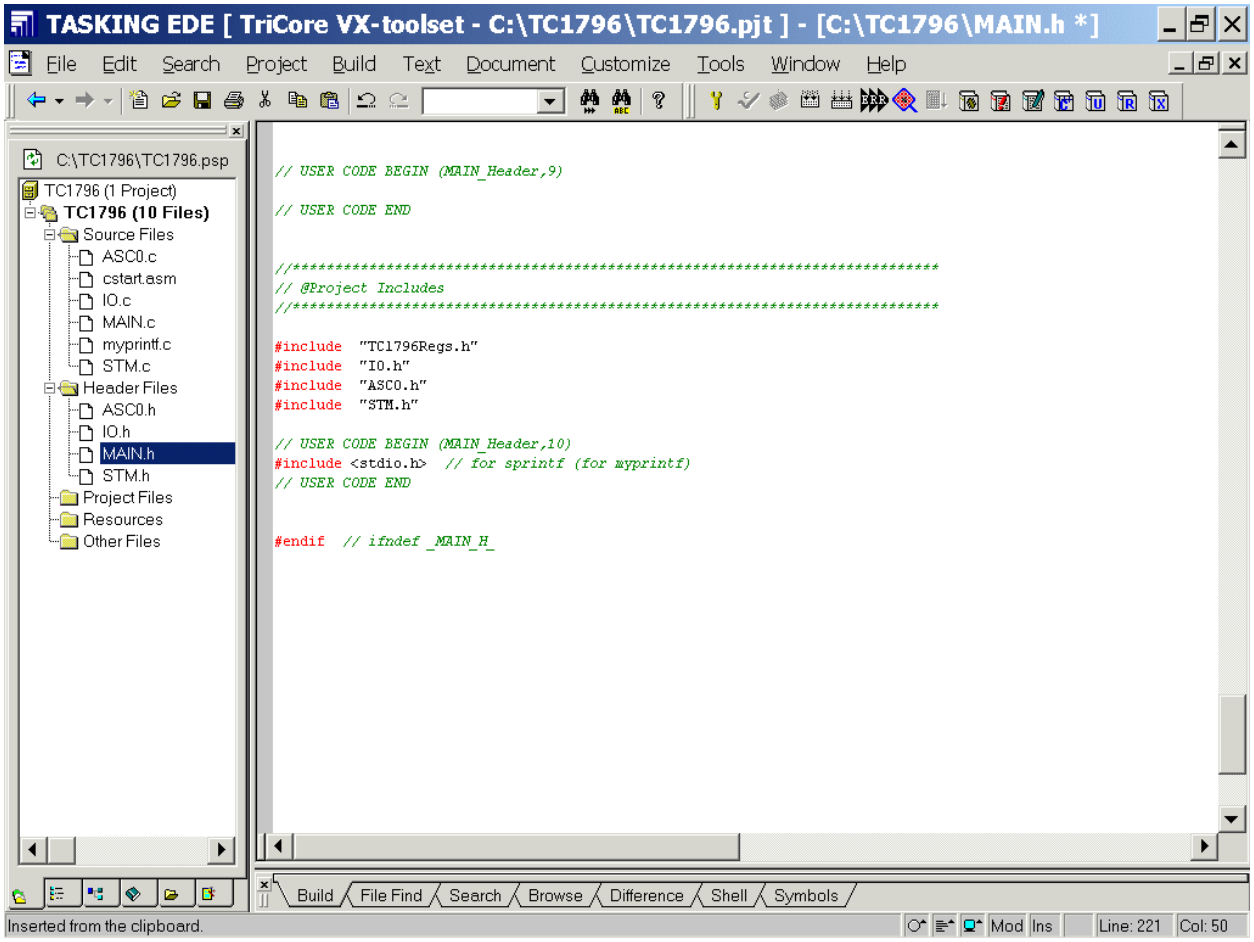
Double click: Main.h and insert Prototypes of Global Functions:

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



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

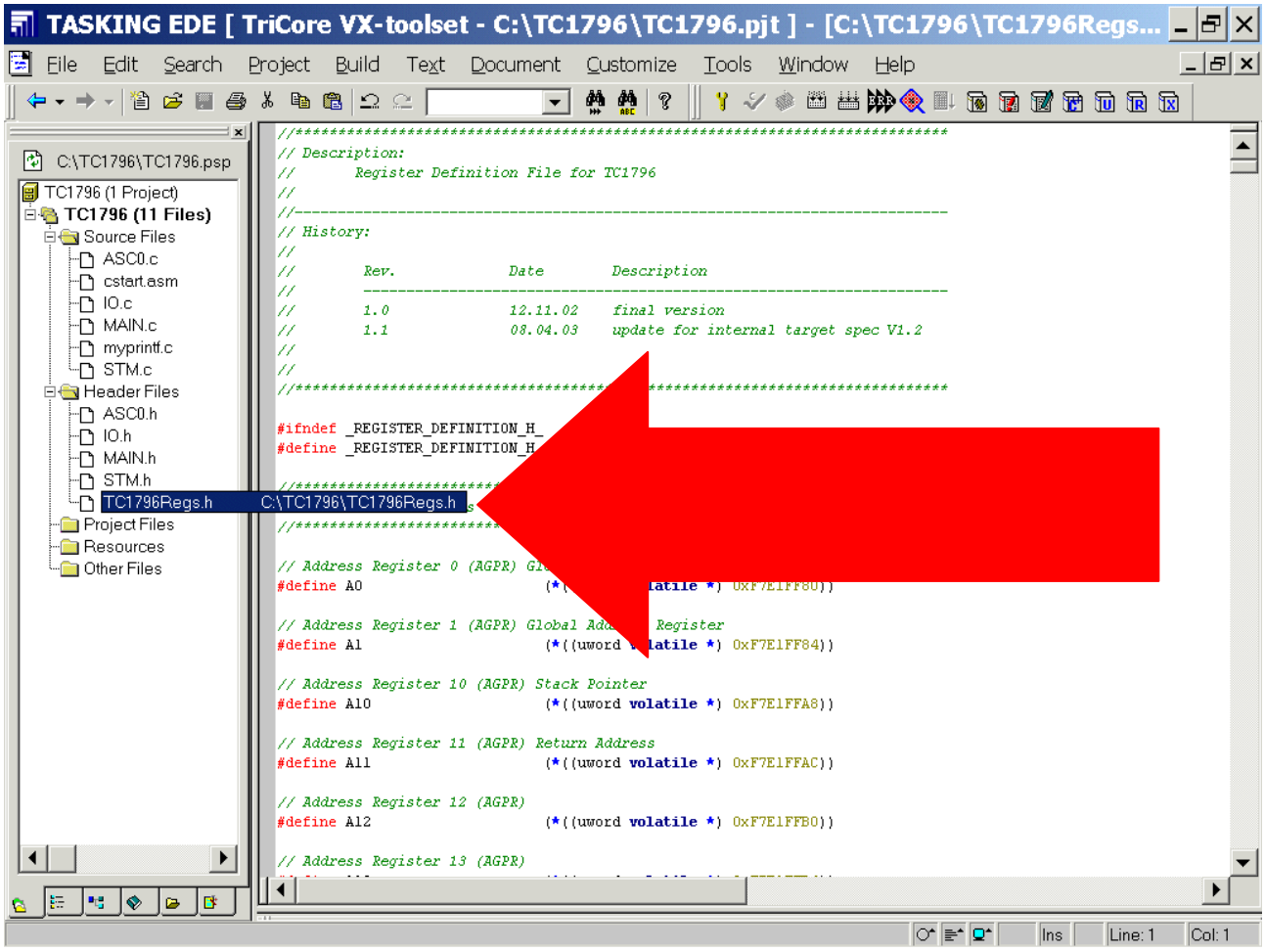
```
#include <stdio.h>     // for sprintf (for myprintf)
```



(Project Window File View) – TC1796 (Files) – right mouse button click – Add Existing Files – Browse

select TC1796Regs.h

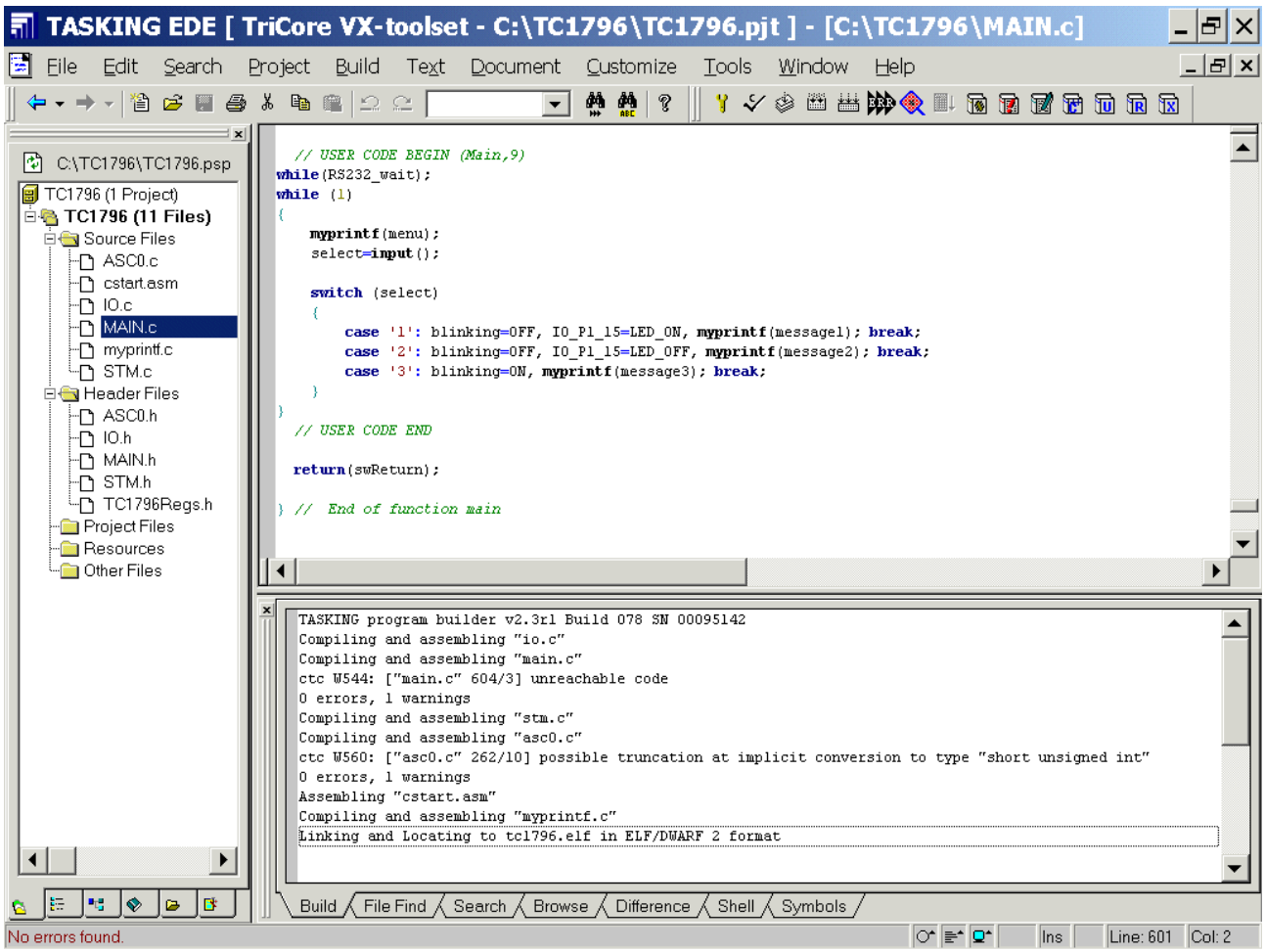
Open - OK



Generate your application program:

Build
Rebuild

or

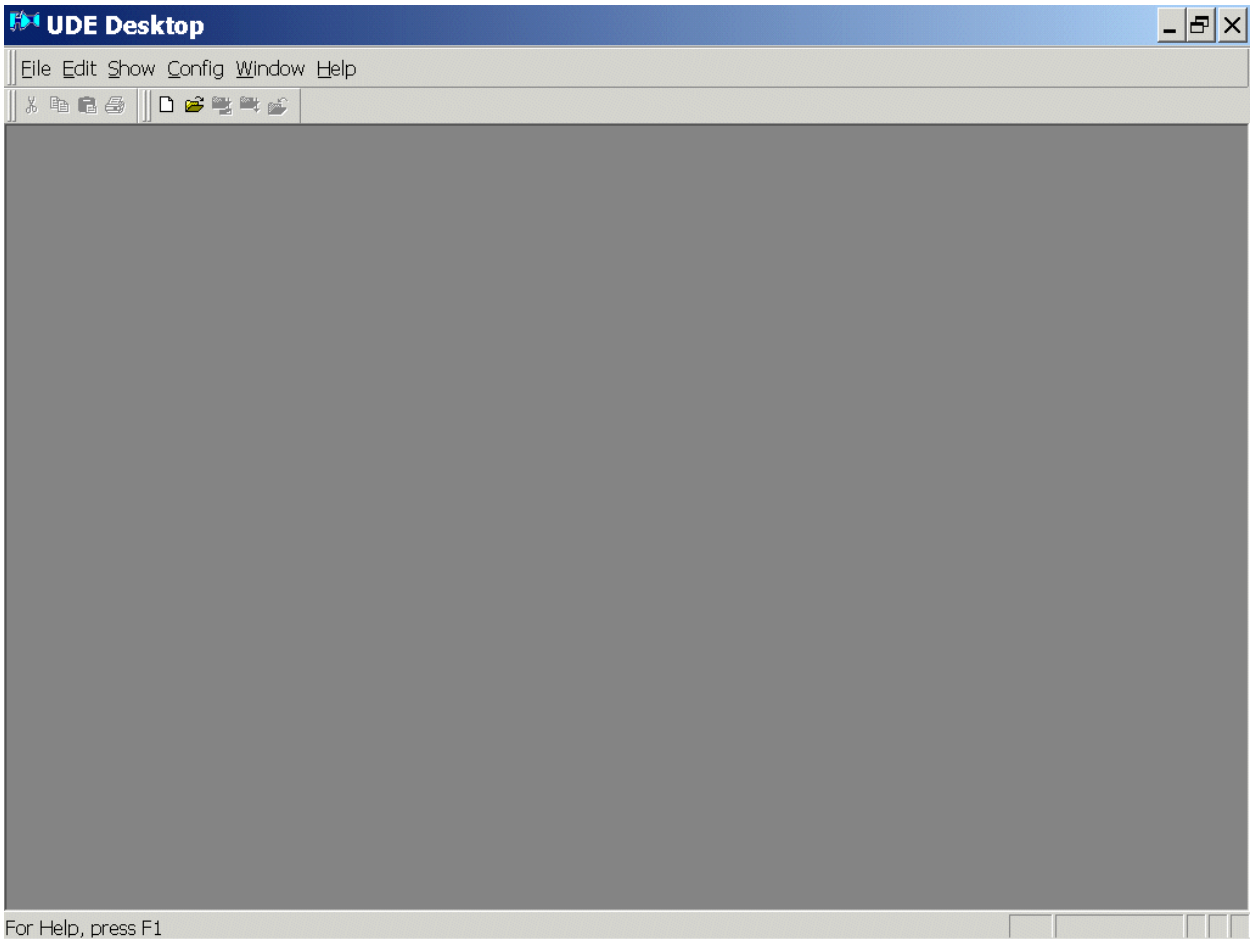


Now you can close your project and Tasking EDE:

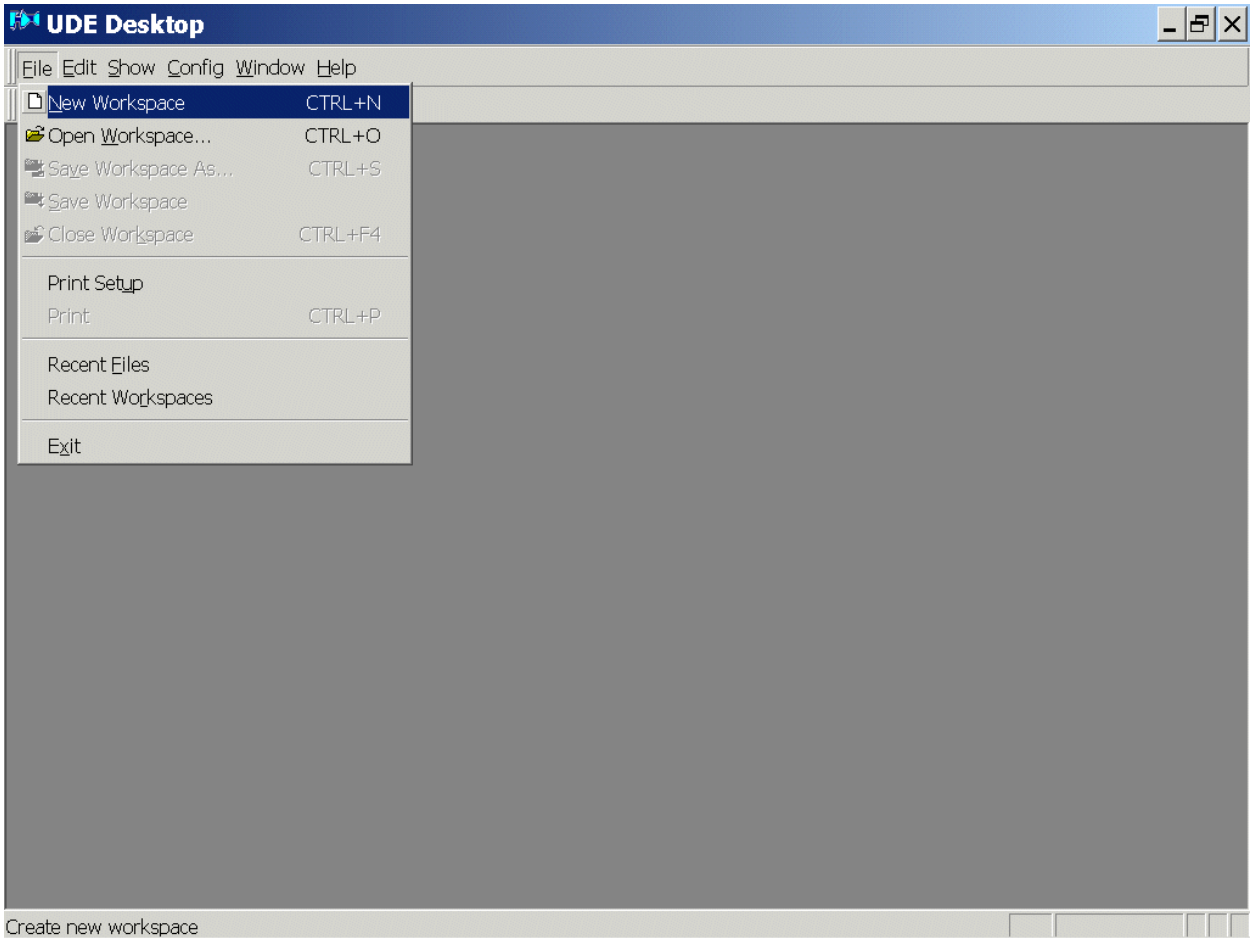
File - Close Project Space
File – Exit

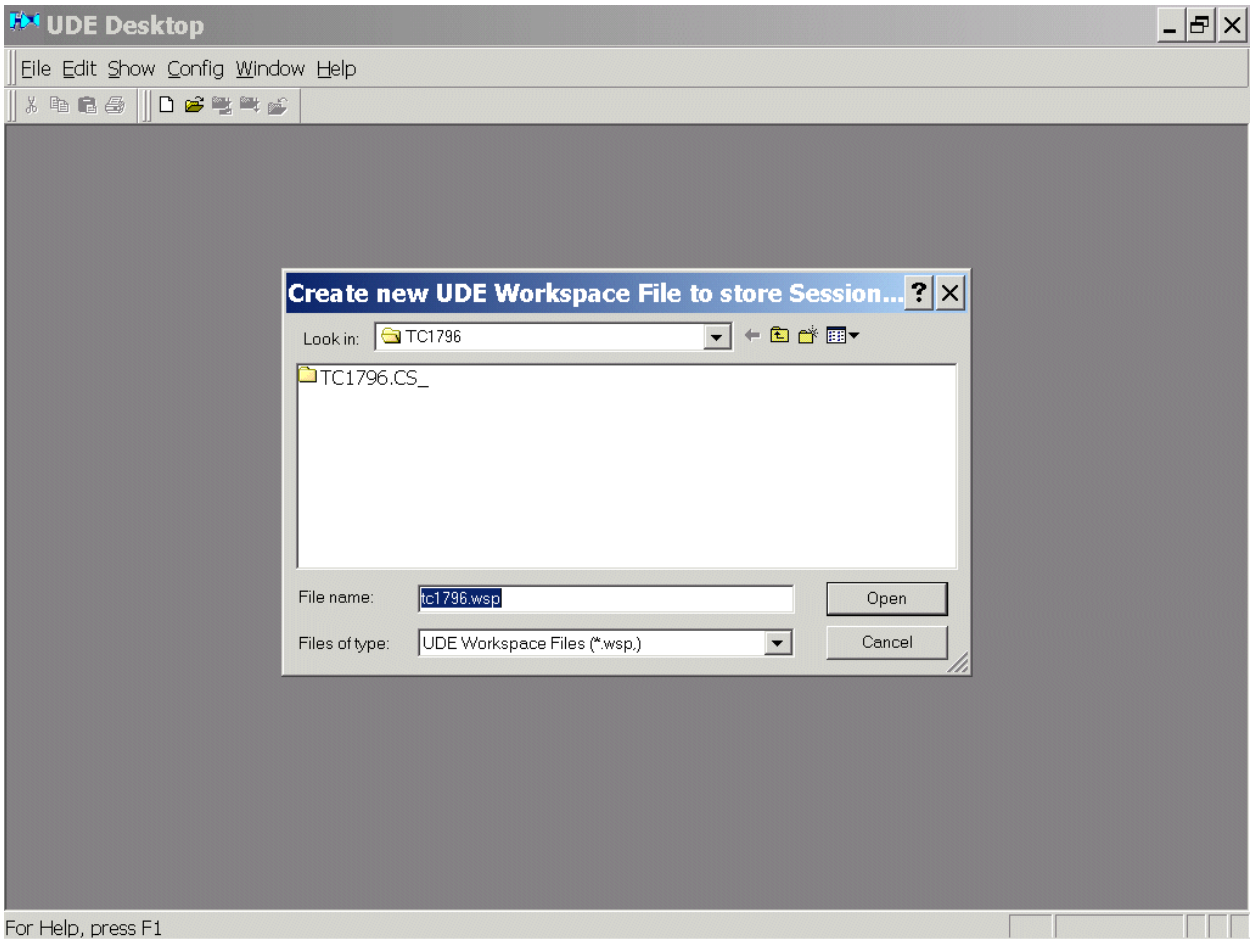
Programming is now complete. You can now load and run your program:

Start pls-Debugger



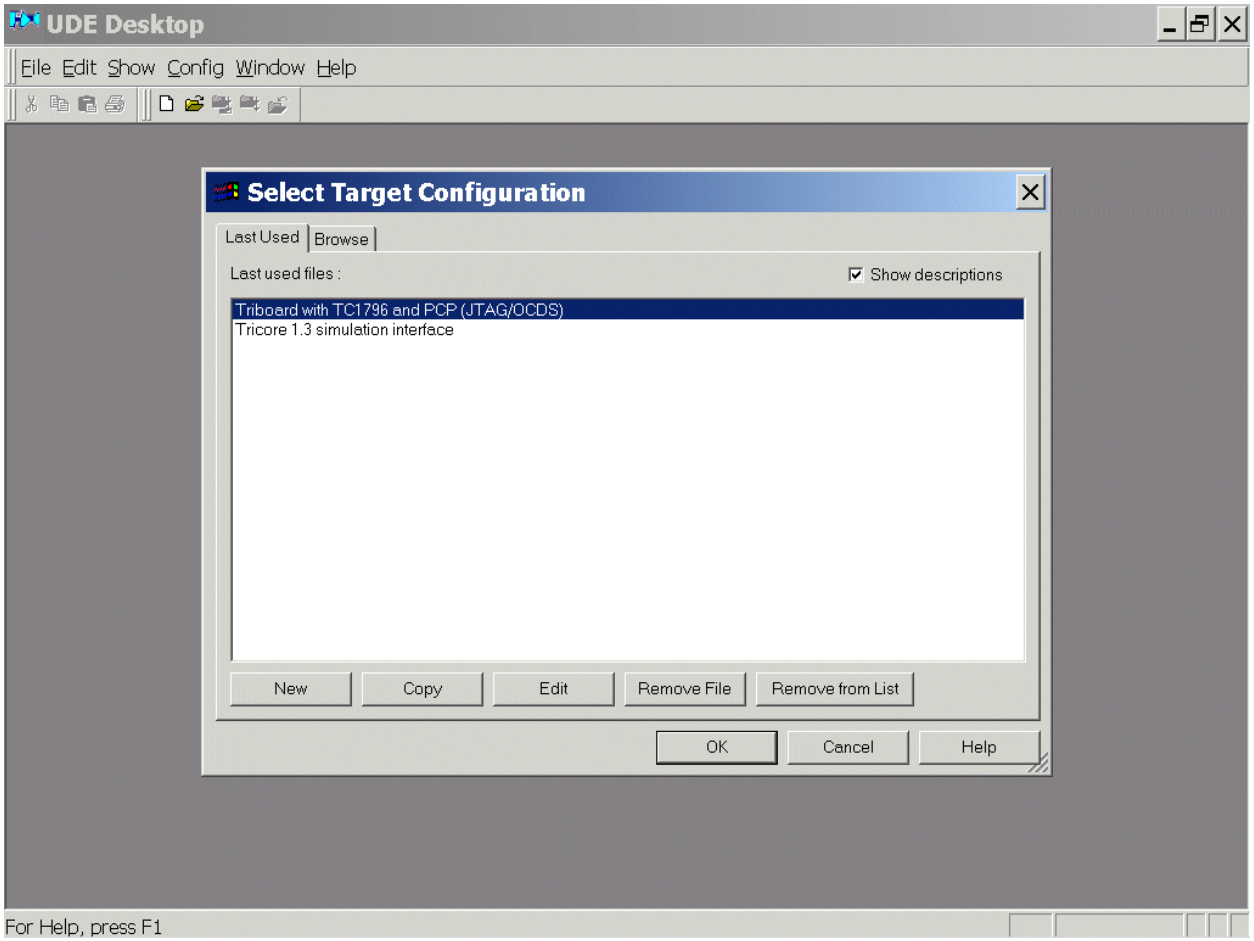
File – New Workspace




UDE Desktop
File Edit Show Config Window Help
Create new UDE Workspace File to store Session...
Look in:
TC1796
TC1796.CS_
File name:
tc1796.wsp
Open
Files of type:
UDE Workspace Files (*.wsp,)
Cancel
For Help, press F1

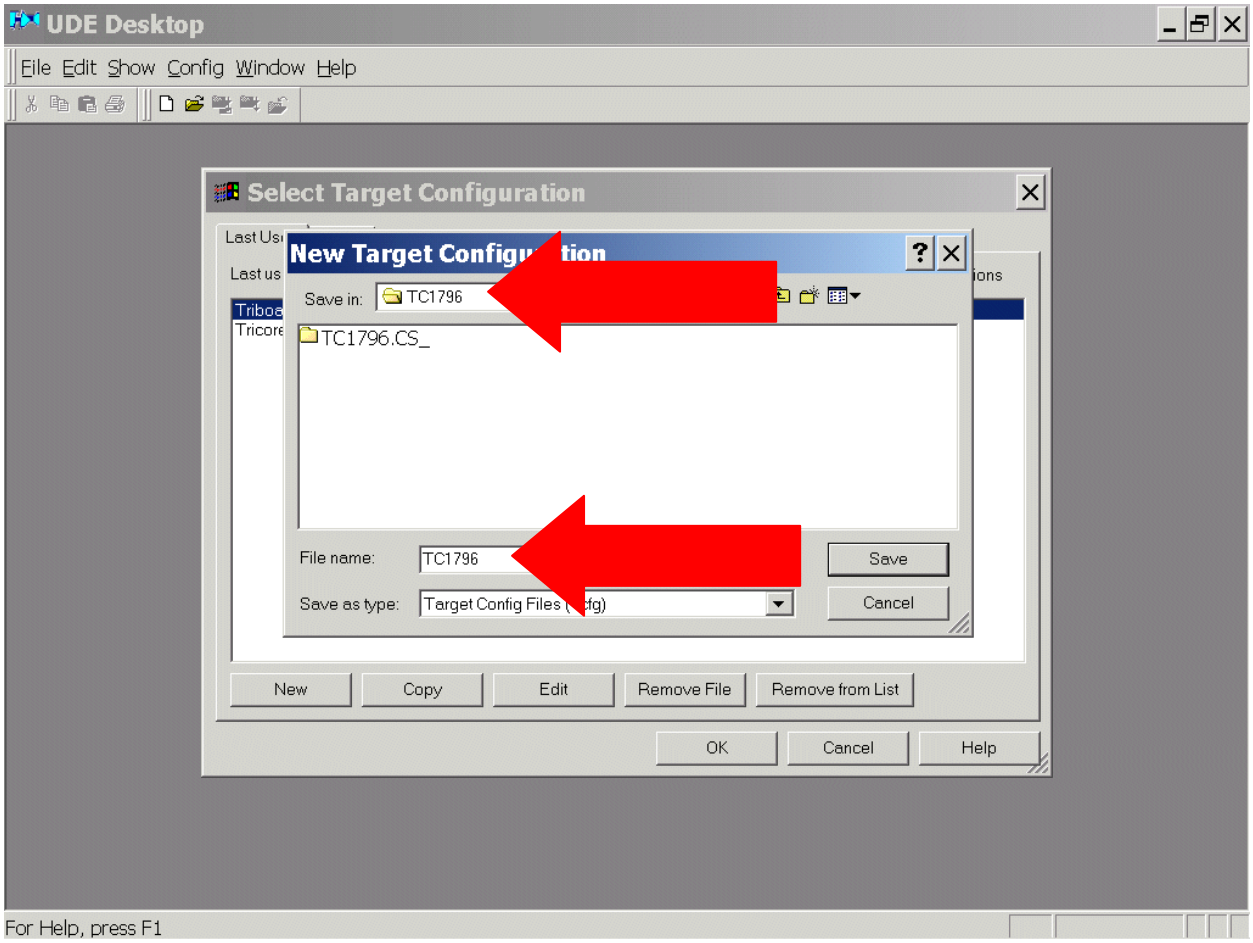
Open

Last used: Last used files: Triboard with TC1796 and PCP (JTAG/OCDS)

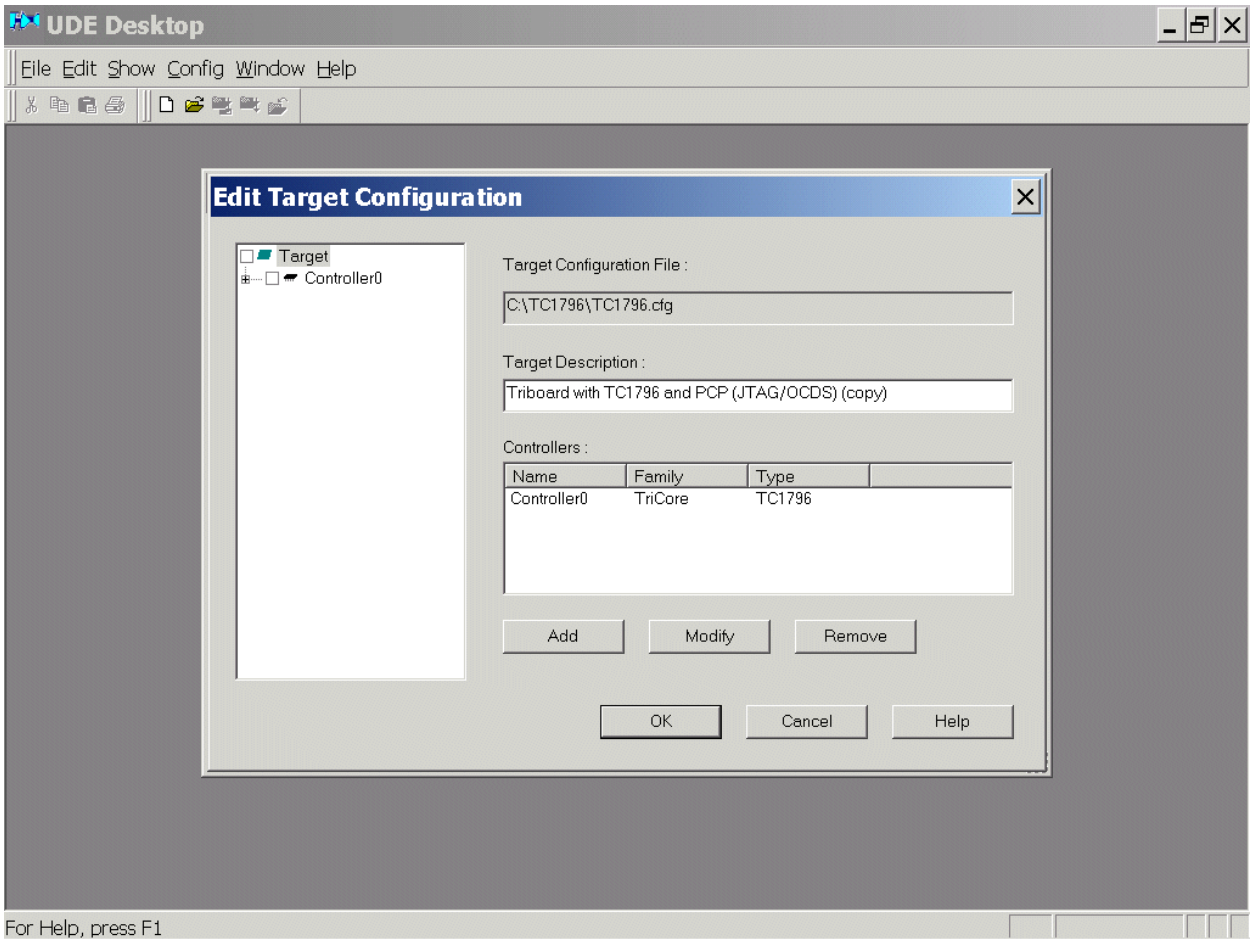


Copy

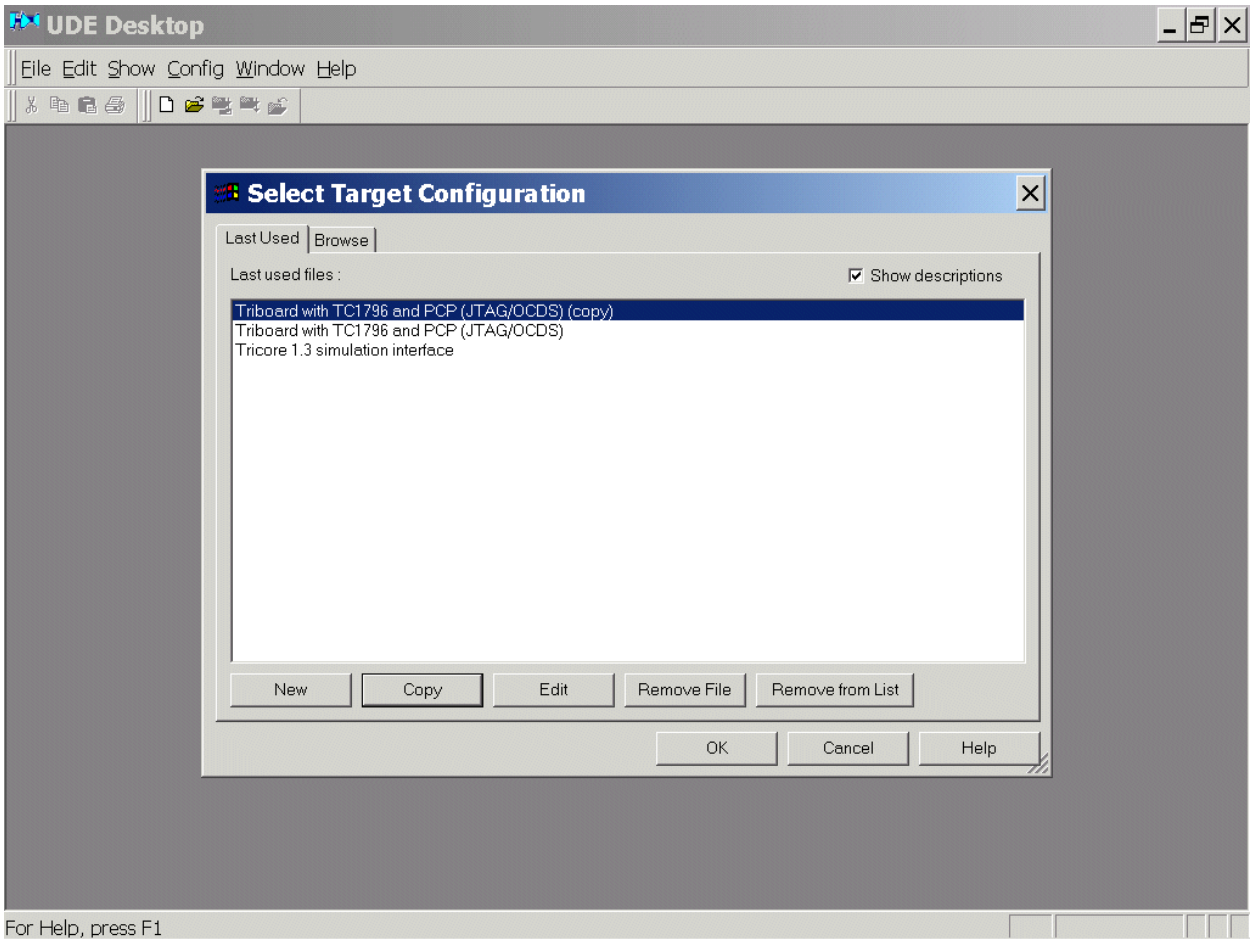
Save in: select C:\TC1796
File name: insert TC1796



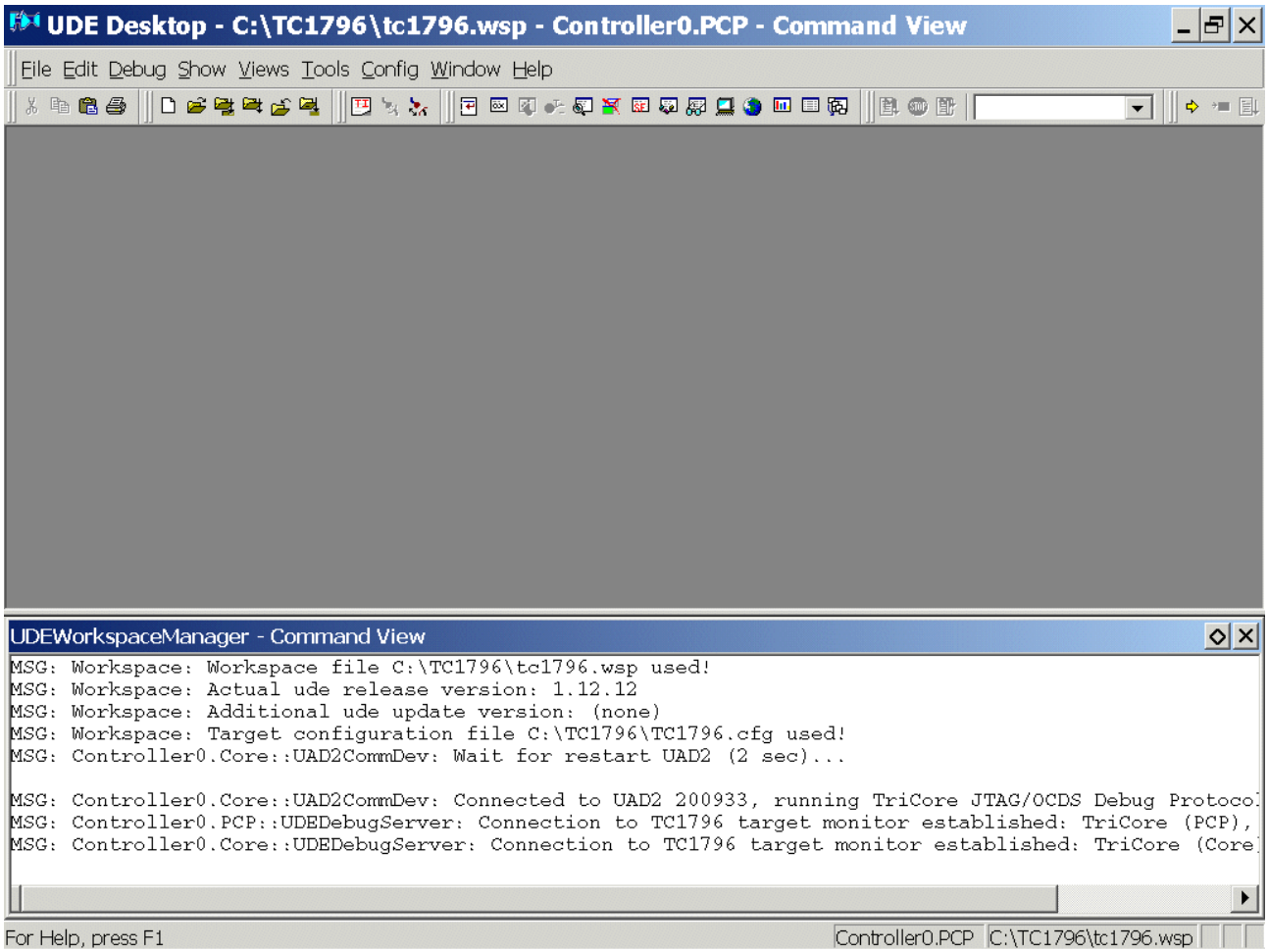
Save


UDE Desktop
File Edit Show Config Window Help
Edit Target Configuration
Target
Controller0
Target Configuration File :
C:\TC1796\TC1796.cfg
Target Description :
Triboard with TC1796 and PCP (JTAG/OCDS) (copy)
Controllers :
Name
Family
Type
Controller0
TriCore
TC1796
Add
Modify
Remove
OK
Cancel
Help
For Help, press F1

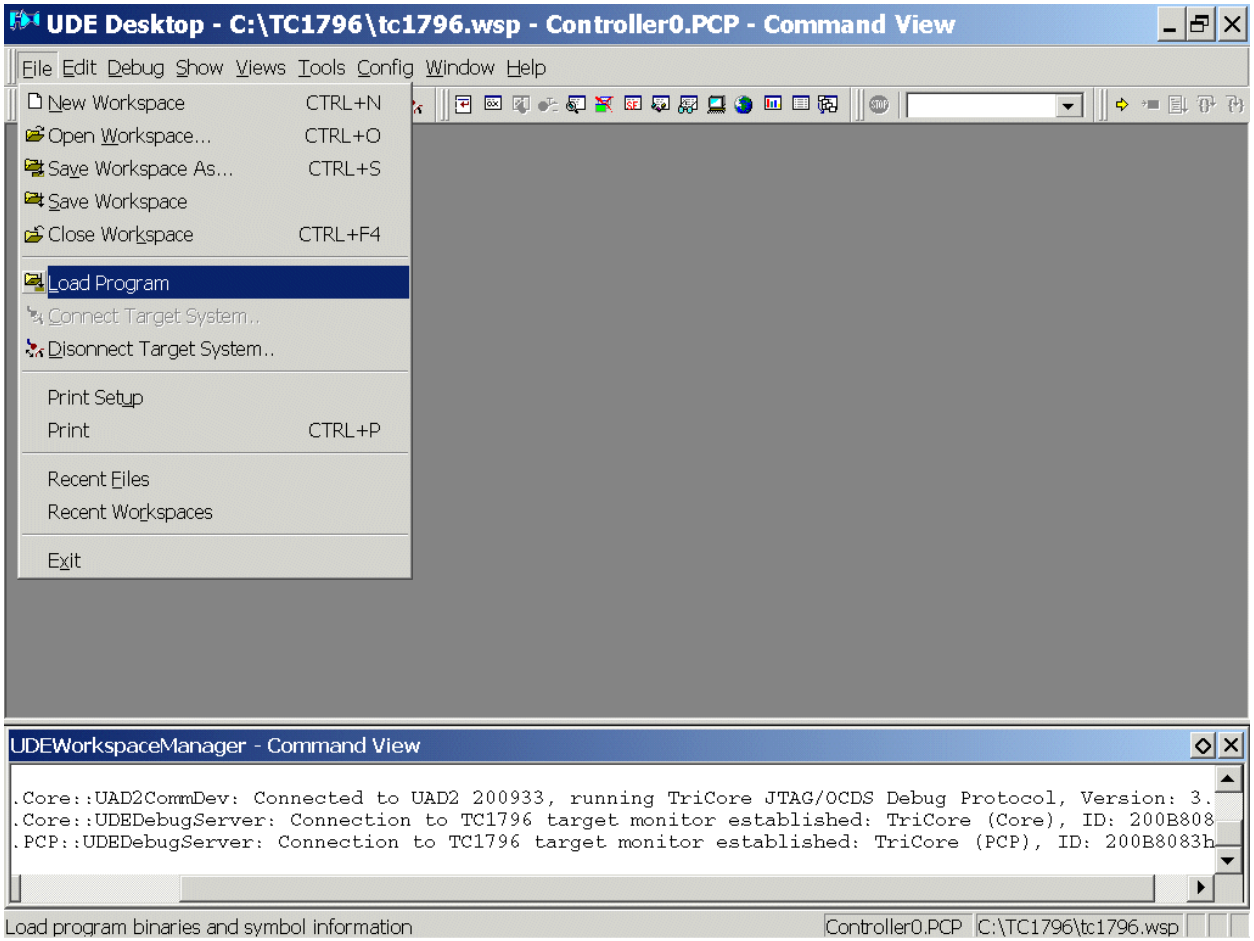
OK


UDE Desktop
File Edit Show Config Window Help
Select Target Configuration
Last Used
Browse
Last used files :
Show descriptions
Triboard with TC1796 and PCP (JTAG/OCDS) (copy)
Triboard with TC1796 and PCP (JTAG/OCDS)
Tricore 1.3 simulation interface
New
Copy
Edit
Remove File
Remove from List
OK
Cancel
Help
For Help, press F1

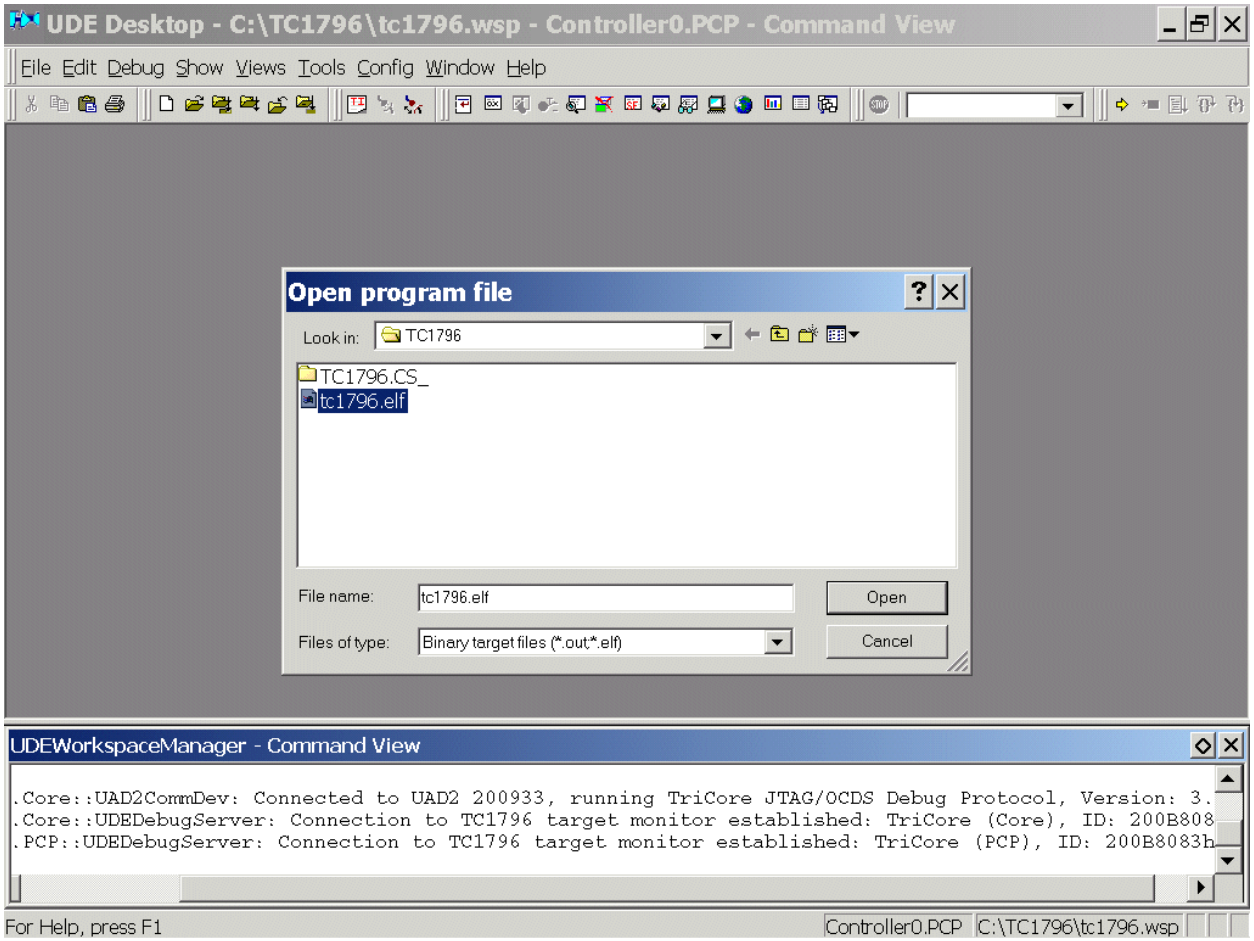
OK


UDE Desktop - C:\TC1796\tc1796.wsp - Controller0.PCP - Command View
File Edit Debug Show Views Tools Config Window Help
UDEWorkspaceManager - Command View
MSG: Workspace: Workspace file C:\TC1796\tc1796.wsp used!
MSG: Workspace: Actual ude release version: 1.12.12
MSG: Workspace: Additional ude update version: (none)
MSG: Workspace: Target configuration file C:\TC1796\TC1796.cfg used!
MSG: Controller0.Core::UAD2CommDev: Wait for restart UAD2 (2 sec)...
MSG: Controller0.Core::UAD2CommDev: Connected to UAD2 200933, running TriCore JTAG/OCDS Debug Protoco
MSG: Controller0.PCP::UDEDebugServer: Connection to TC1796 target monitor established: TriCore (PCP),
MSG: Controller0.Core::UDEDebugServer: Connection to TC1796 target monitor established: TriCore (Core
For Help, press F1
Controller0.PCP
C:\TC1796\tc1796.wsp

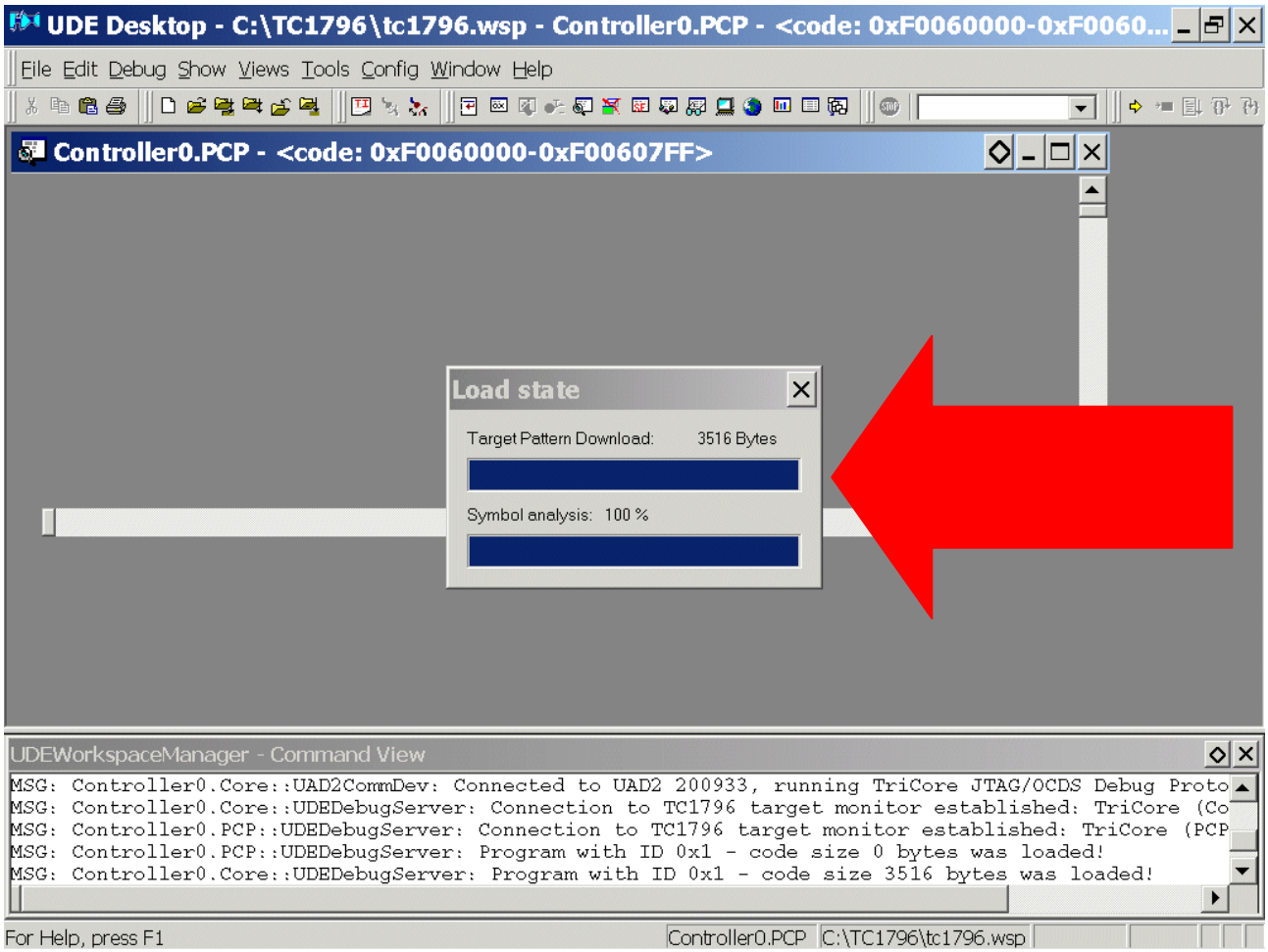
File – Load Program

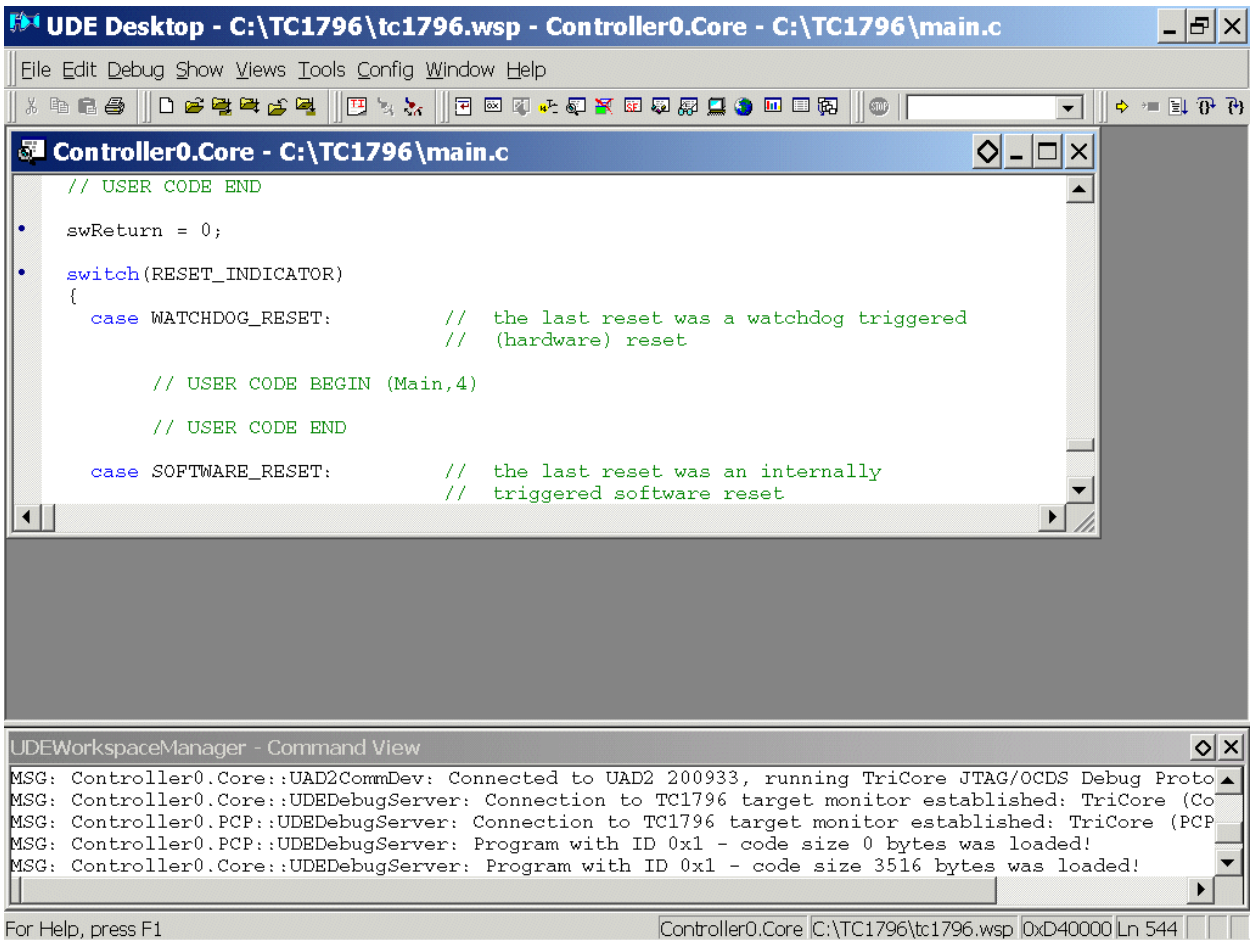


Select tc1796.elf

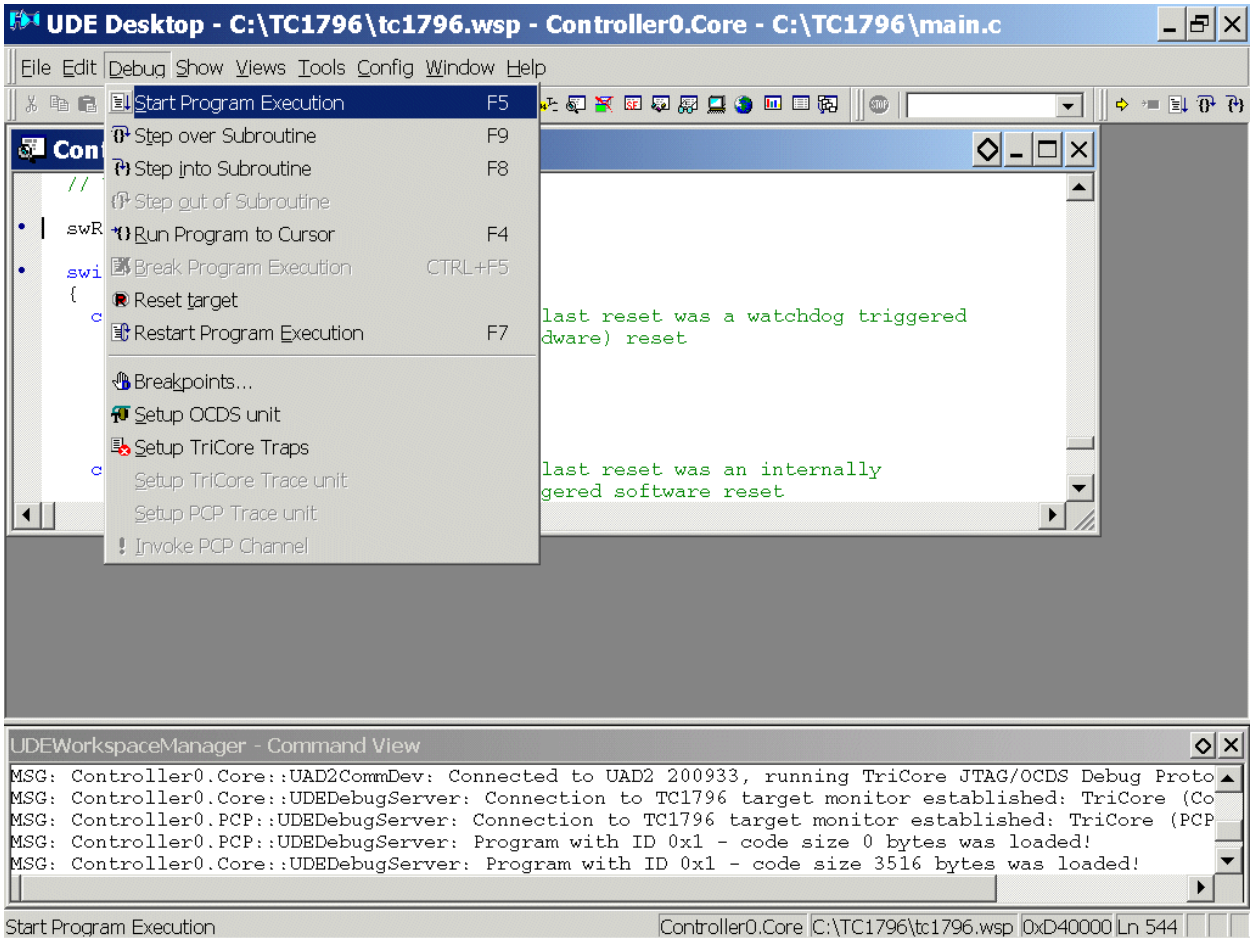


Open

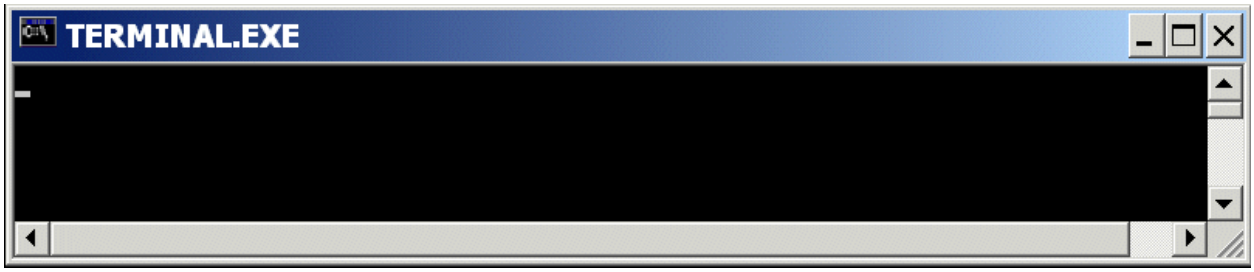

UDE Desktop - C:\TC1796\tc1796.wsp - Controller0.PCP - <code: 0xF0060000-0xF0060...
File Edit Debug Show Views Tools Config Window Help
Controller0.PCP - <code: 0xF0060000-0xF00607FF>
Load state
Target Pattern Download: 3516 Bytes
Symbol analysis: 100 %
UDEWorkspaceManager - Command View
MSG: Controller0.Core::UAD2CommDev: Connected to UAD2 200933, running TriCore JTAG/OCDS Debug Proto
MSG: Controller0.Core::UDEDebugServer: Connection to TC1796 target monitor established: TriCore (Co
MSG: Controller0.PCP::UDEDebugServer: Connection to TC1796 target monitor established: TriCore (PCP
MSG: Controller0.PCP::UDEDebugServer: Program with ID 0x1 - code size 0 bytes was loaded!
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3516 bytes was loaded!
For Help, press F1
Controller0.PCP
C:\TC1796\tc1796.wsp


UDE Desktop - C:\TC1796\tc1796.wsp - Controller0.Core - C:\TC1796\main.c
File Edit Debug Show Views Tools Config Window Help
Controller0.Core - C:\TC1796\main.c
// USER CODE END
swReturn = 0;
switch(RESET_INDICATOR)
{
case WATCHDOG_RESET: // the last reset was a watchdog triggered
// (hardware) reset
// USER CODE BEGIN (Main,4)
// USER CODE END
case SOFTWARE_RESET: // the last reset was an internally
// triggered software reset
UDEWorkspaceManager - Command View
MSG: Controller0.Core::UAD2CommDev: Connected to UAD2 200933, running TriCore JTAG/OCDS Debug Proto
MSG: Controller0.Core::UDEDebugServer: Connection to TC1796 target monitor established: TriCore (Co
MSG: Controller0.PCP::UDEDebugServer: Connection to TC1796 target monitor established: TriCore (PCP
MSG: Controller0.PCP::UDEDebugServer: Program with ID 0x1 - code size 0 bytes was loaded!
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3516 bytes was loaded!
For Help, press F1
Controller0.Core
C:\TC1796\tc1796.wsp
0xD40000
Ln 544

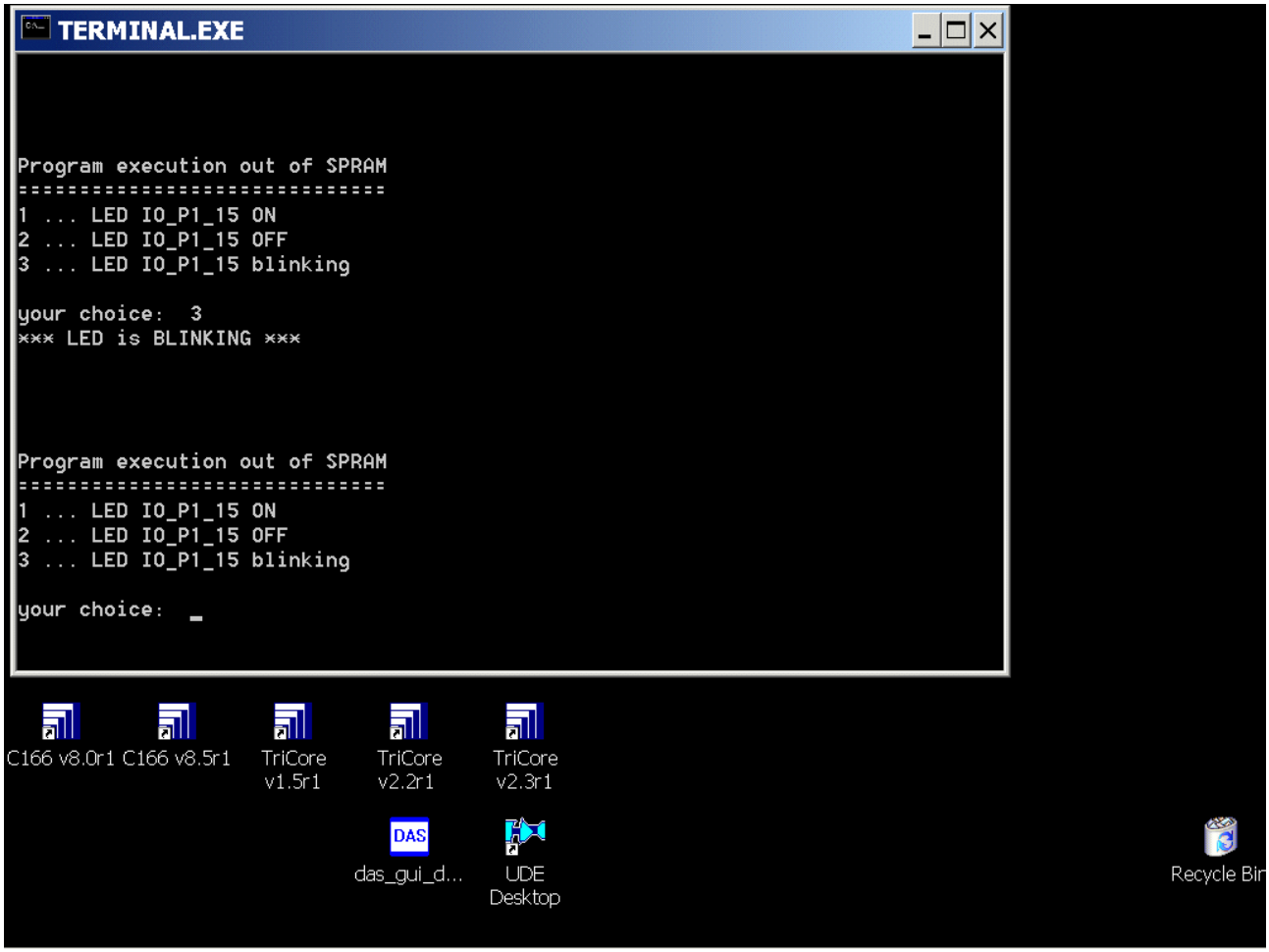
Debug – Start Program Execution



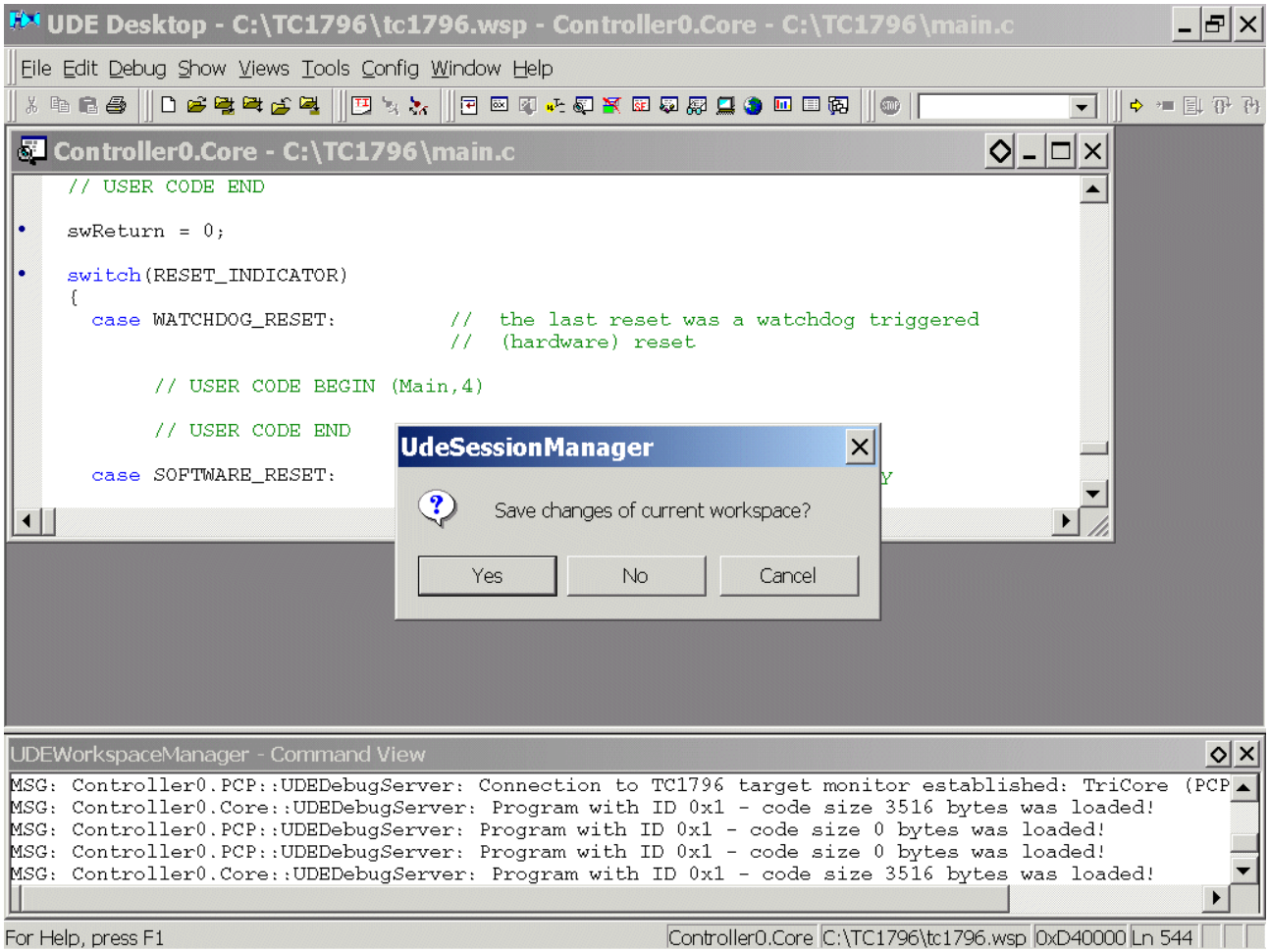
Execute any terminal-program
(9600 Baud, 8 bit Data, no Parity-Bit, 1 Stop-Bit, Xon/Xoff Protocol):



And see the result:



File – Close Workspace

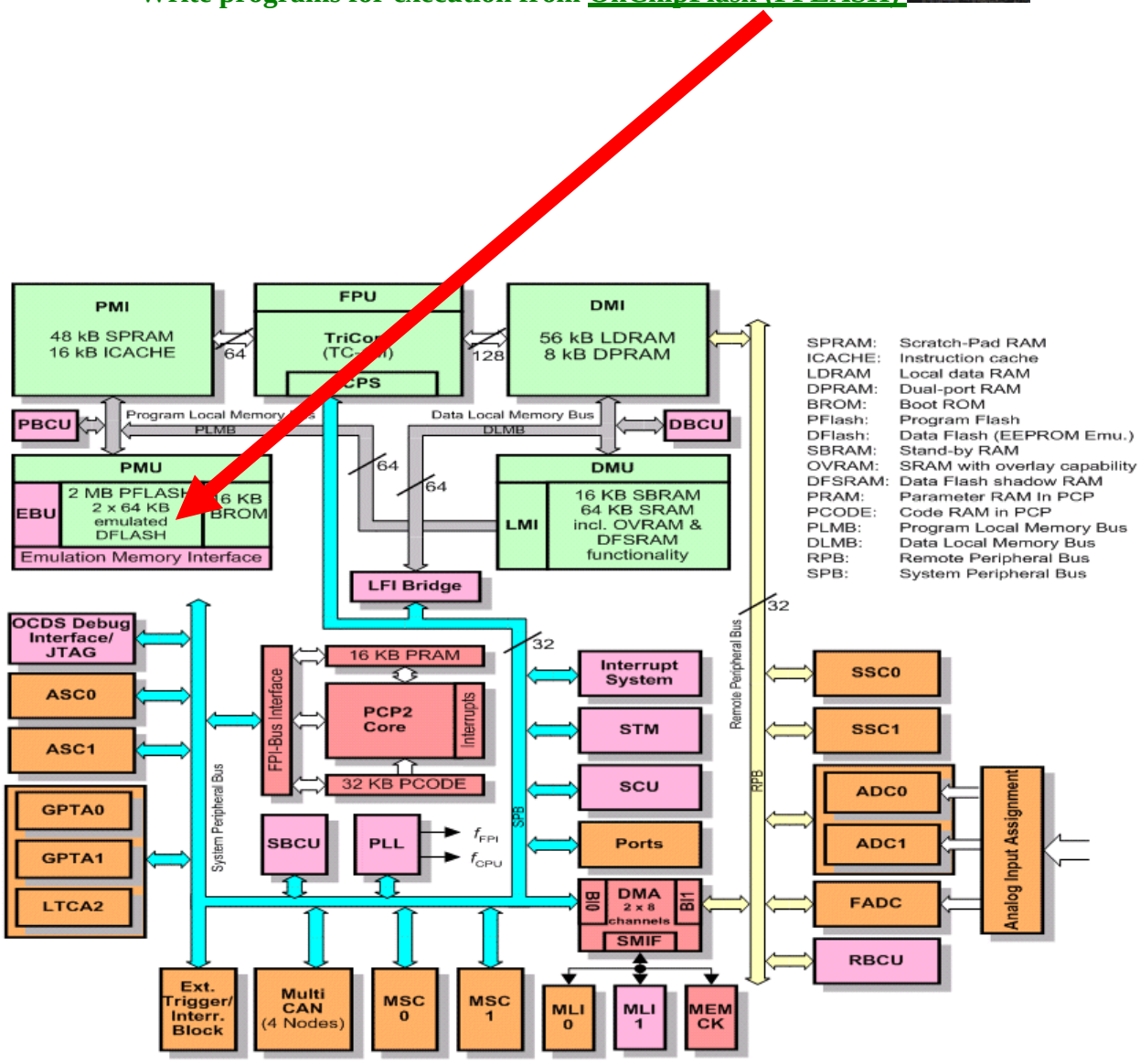


Yes

File - Exit

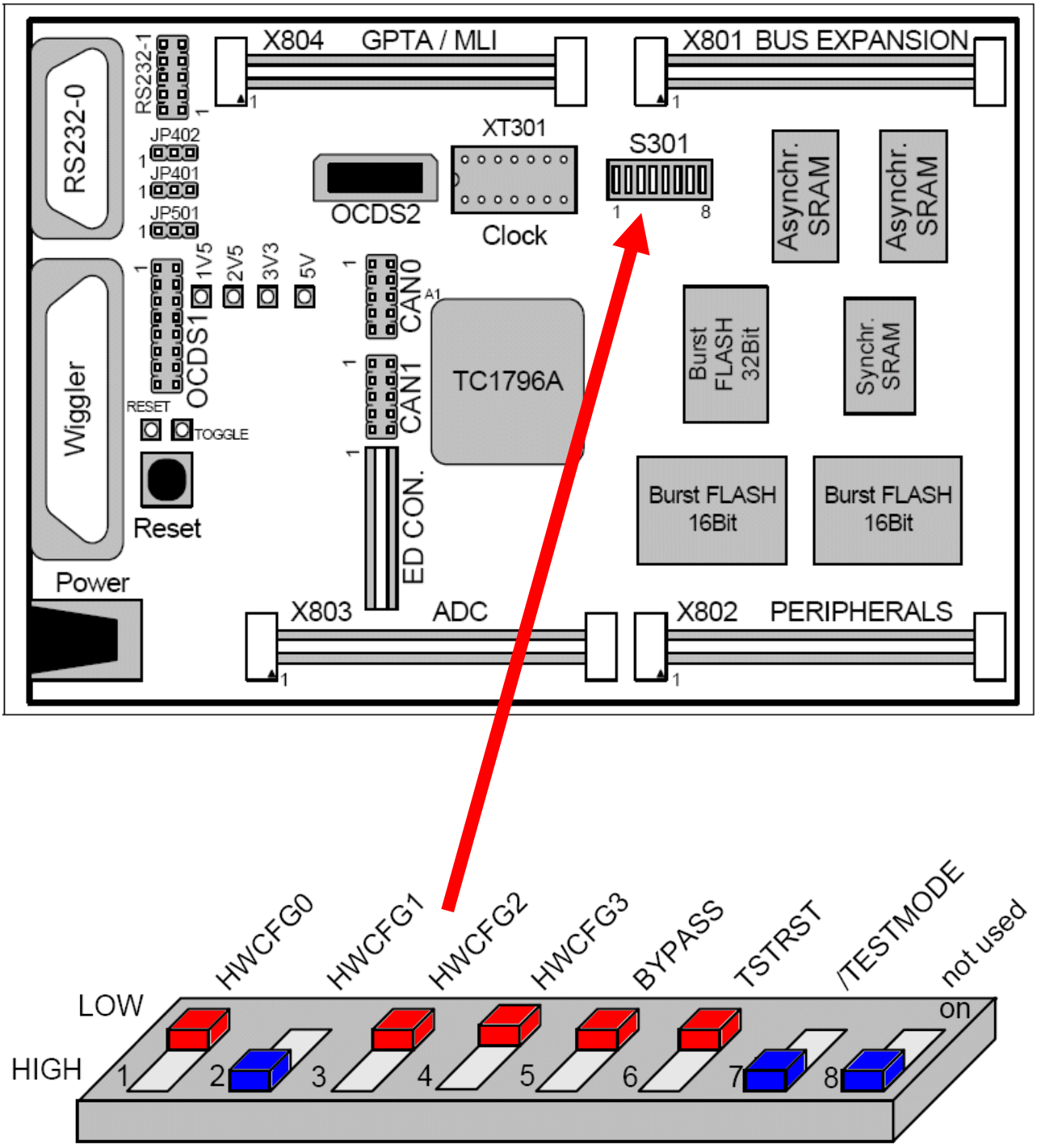
5.) Using of the TASKING - EDE Development Tools:

Write programs for execution from OnChipFlash (PFLASH) 



TC1796-Execution-Environment: OnChipFlash (PFLASH):

Jumper Settings (HW-Configuration DIP-Switch):



Adobe Reader - [TriBoardManual-TC179X-V32.pdf]

File Edit View Document Tools Window Help

Save a Copy Search Select 102% Help

Bookmarks Pages Attachments Comments

Note: 'x' represents the don't care state.

Note: The signal /BRK_IN will be set by the Debugger via OCDS-Interface.

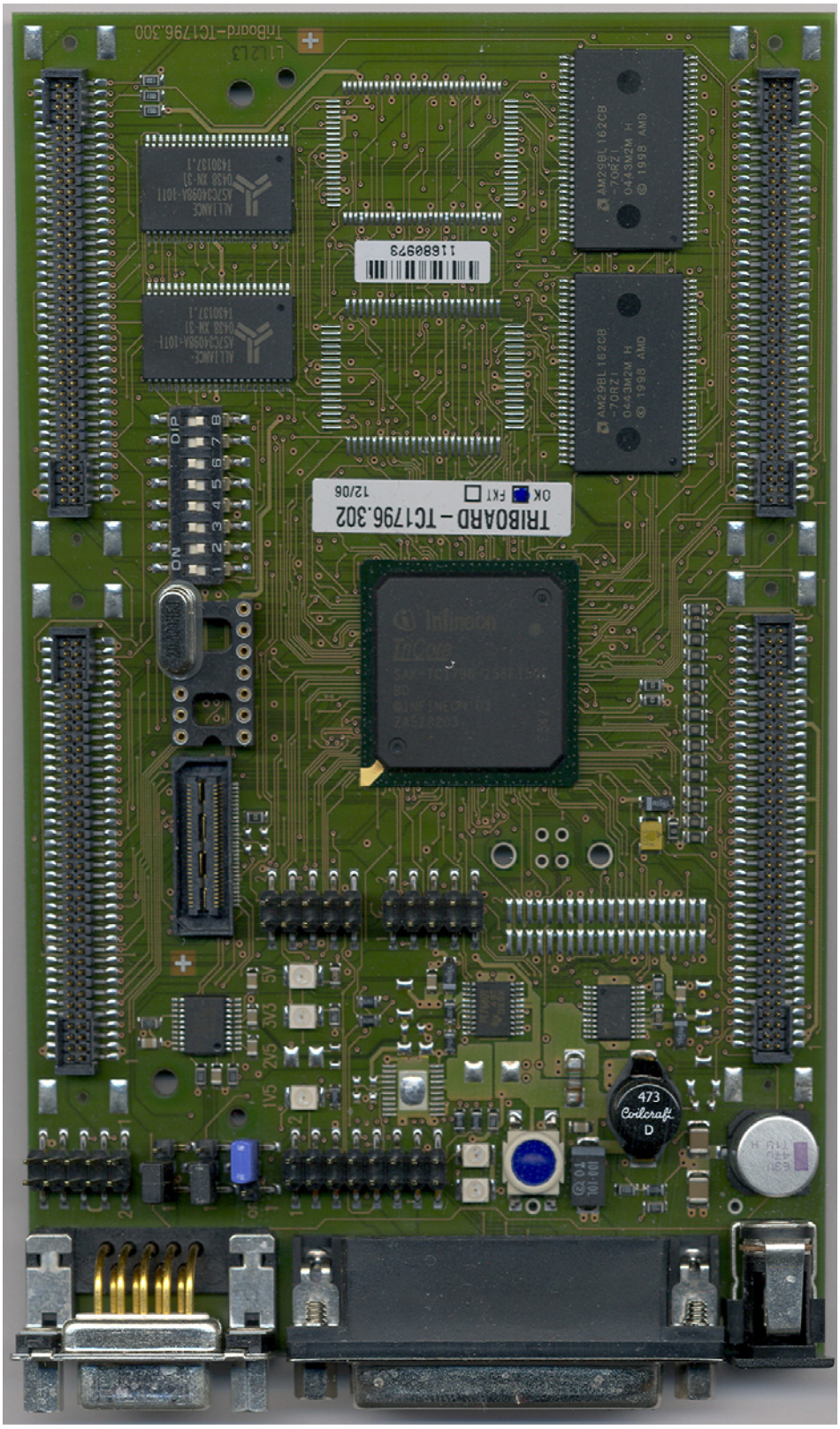
/BRK_IN	HWCFG[3...0]	Type of Boot	PC Start value
1	0000	Serial boot from ASC to PMI scratchpad, run loaded program	0xD4000000
1	0001	Serial boot from CAN to PMI scratchpad, run loaded program	0xD4000000
1	0010	Start from internal flash	0xA0000000
1	0011	Alternate Bootmode from internal flash	from Header or 0xD4000000
1	0100	External memory, EBU as master	0xA1000000
1	0101	Alternate Bootmode from External memory, EBU as master	from Header or 0xD4000000
1	0110	External memory, EBU as slave	0xA1000000
1	0111	Alternate Bootmode from External memory, EBU as slave	from Header or 0xD4000000
1	1XXX	reserved; don't use this combination	-
0	1000	go to external emulator space	0xDE000000
0	0000	put chip in tristate (deep sleep)	-
0	all others	reserved; don't use this combination	-

22 of 62

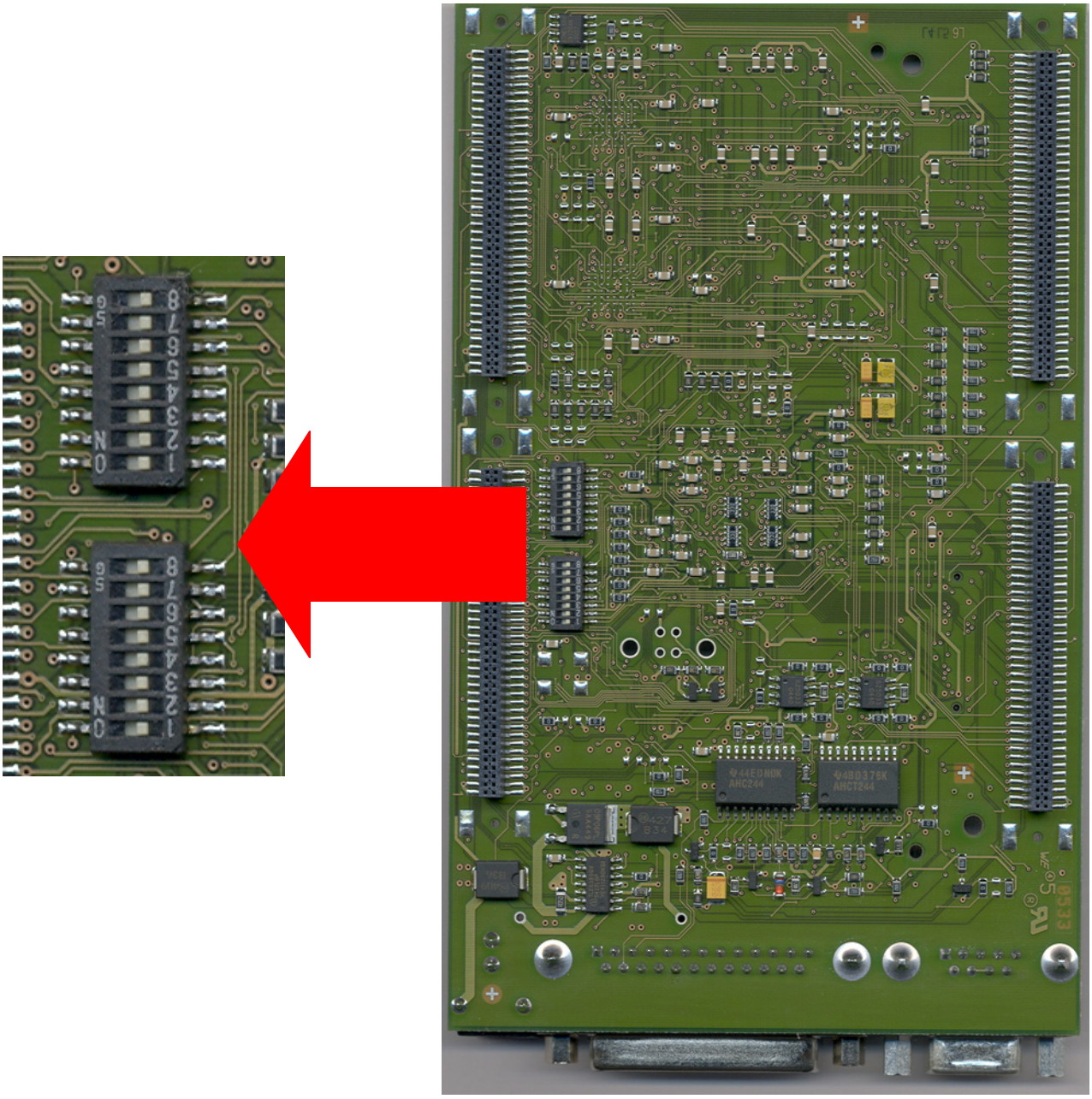
S301:
1, 3, 4, 5, 6 : ON
2, 7, 8 : OFF

TC1796-Execution-Environment: OnChipFlash (PFLASH):

Jumper Settings (HW-Configuration DIP-Switch):



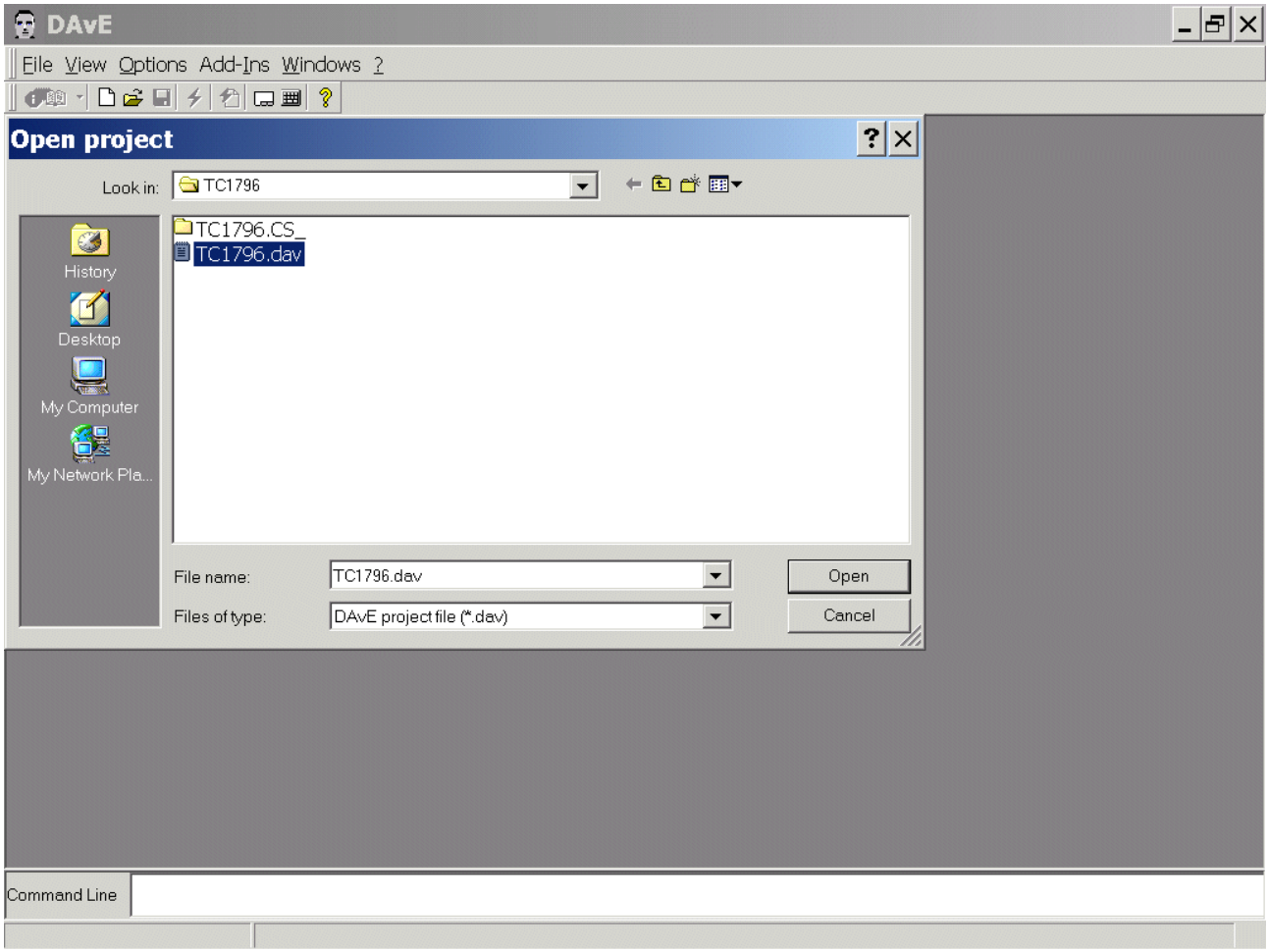
Bottom side: all little DIPs: OFF



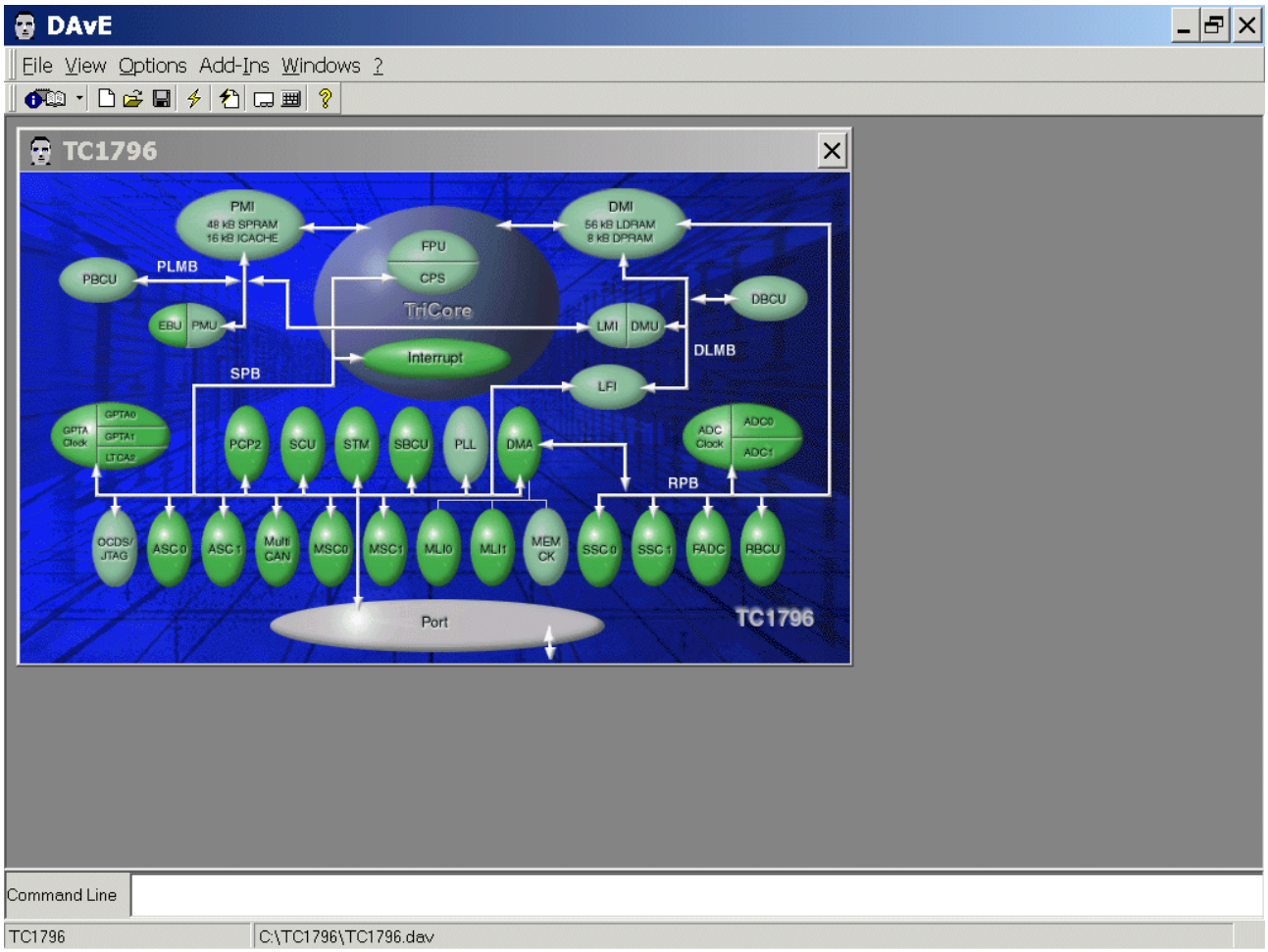
Note:
The little DIPs on the bottom side are for internal purpose.
Please, guarantee that they are all set to “OFF”.

Start DAvE and open the TC1796 project:

File
Open
Look in: select C:\TC1796
File name: select TC1796.dav

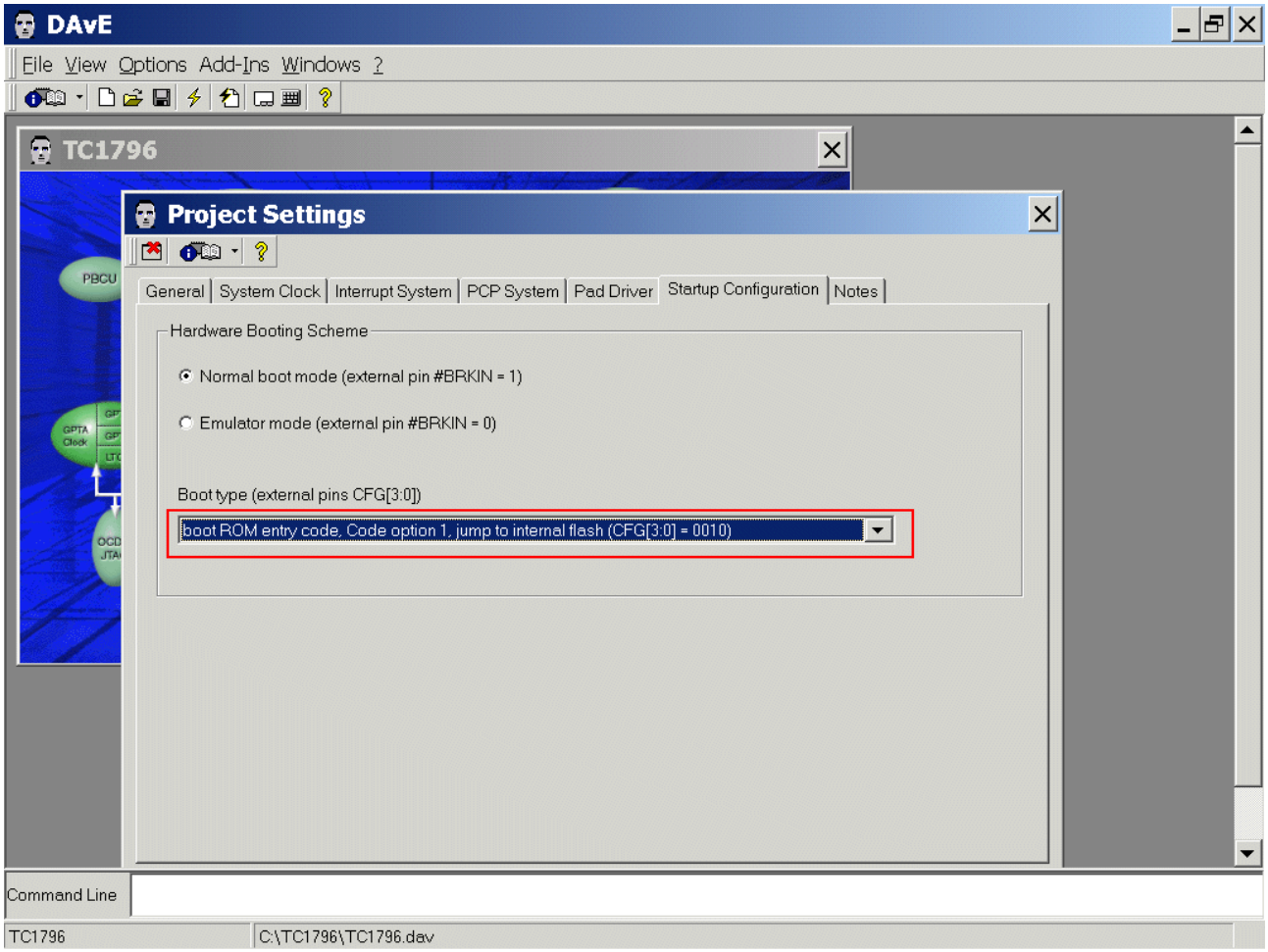


Open

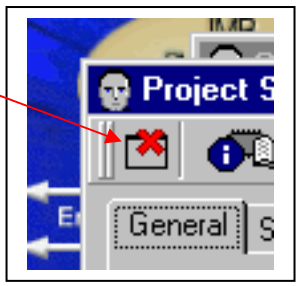

DAvE
File View Options Add-Ins Windows ?
TC1796
PMI
48 kB SPRAM
16 kB ICACHE
DMI
56 kB LDRAM
8 kB DPRAM
FPU
CPS
TriCore
PBCU
PLMB
EBU
PMU
LMI
DMU
DBCU
DLMB
Interrupt
SPB
LFI
GPTA Clock
GPTA0
GPTA1
LTCA2
PCP2
SCU
STM
SBCU
PLL
DMA
ADC Clock
ADC0
ADC1
RPB
OCDS/ JTAG
ASC0
ASC1
Multi CAN
MSC0
MSC1
MLI0
MLI1
MEM CK
SSC0
SSC1
FADC
RBCU
Port
TC1796
Command Line
TC1796
C:\TC1796\TC1796.dav

File – Project Settings

Startup Configuration: Boot type (CFG[3:0]) select 0010



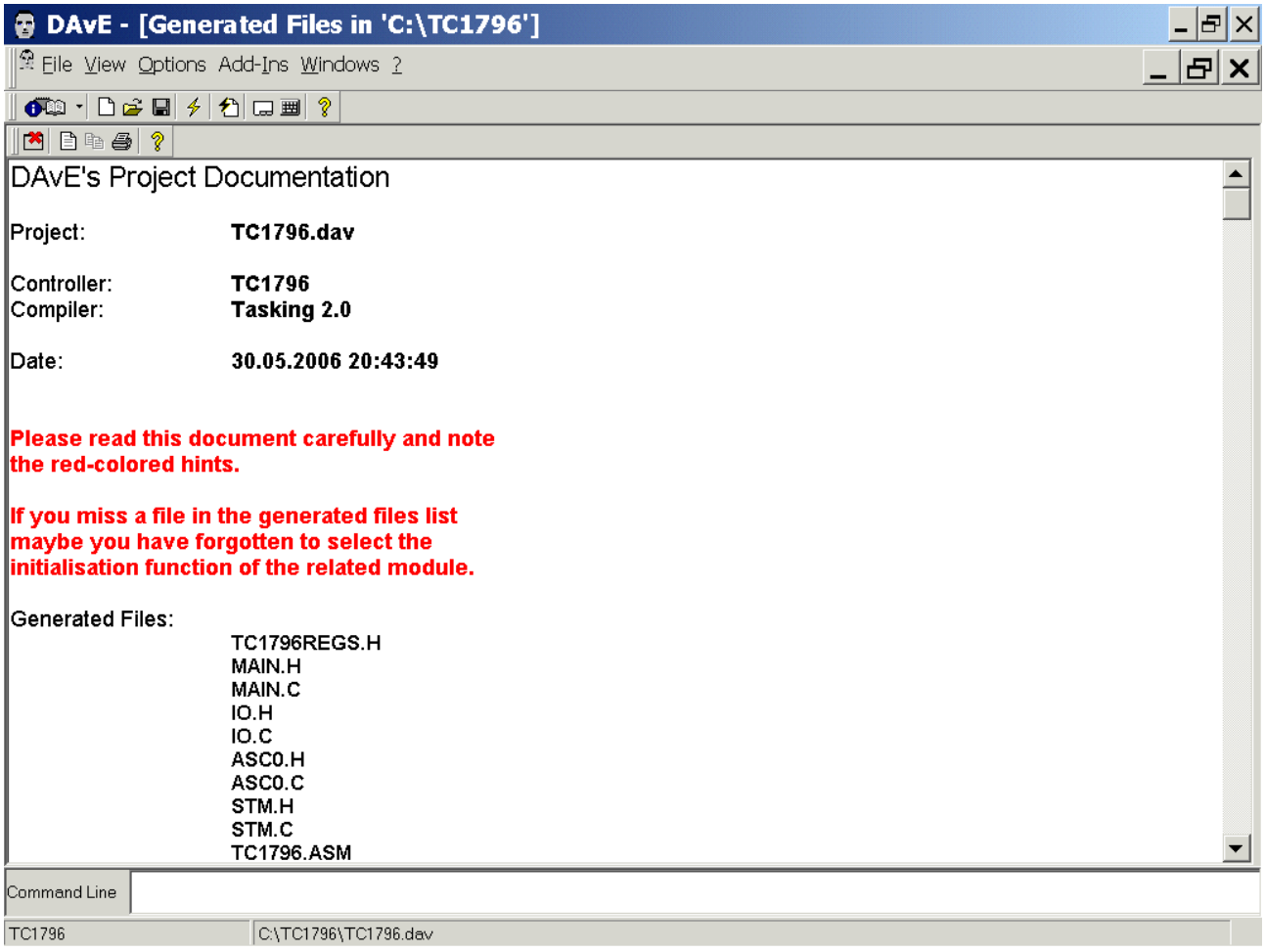
Exit this dialog now by clicking  the close button:



Generate Code:

File Generate Code	or click 

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



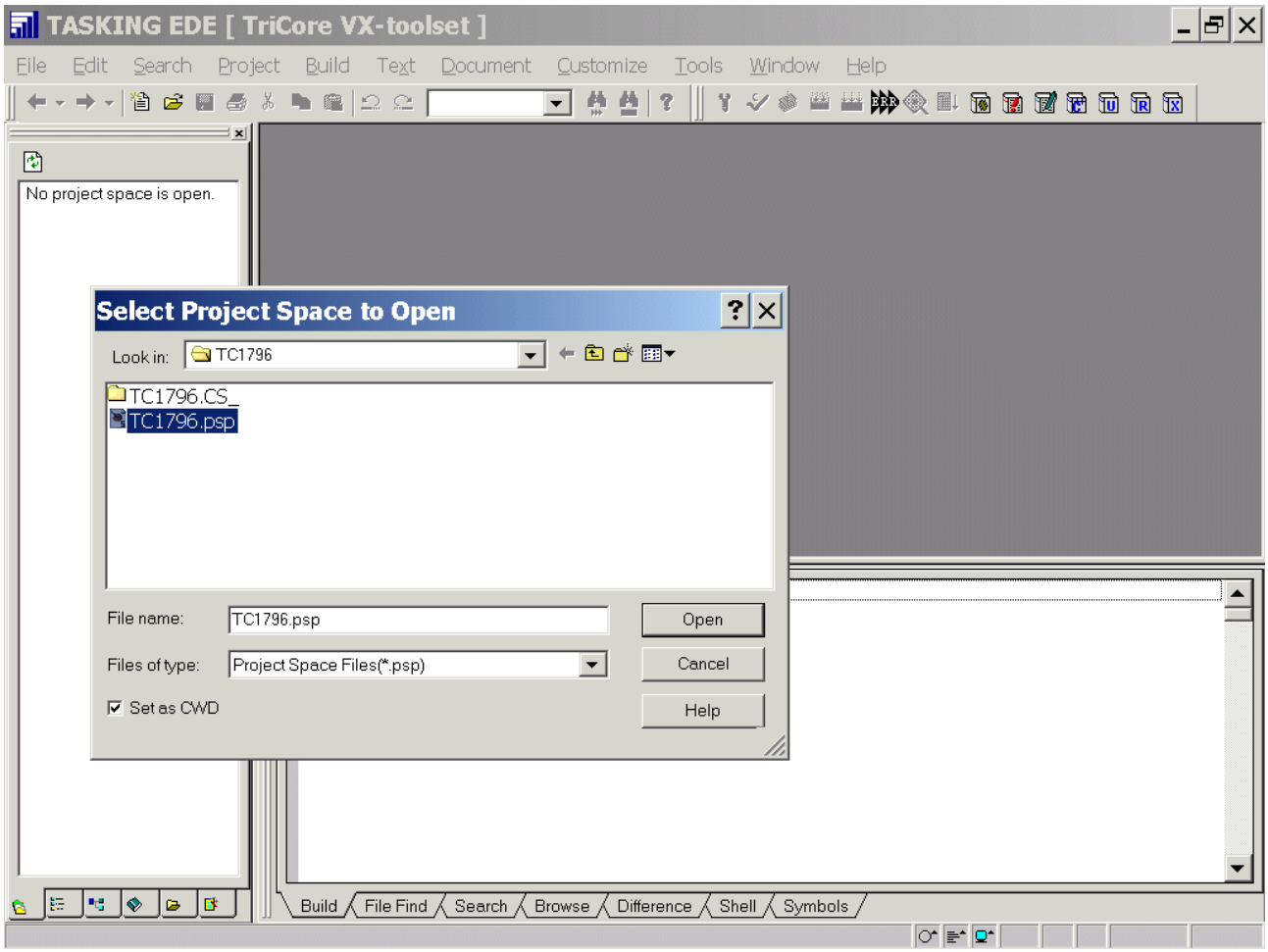
File
Exit
Save changes?

click Yes

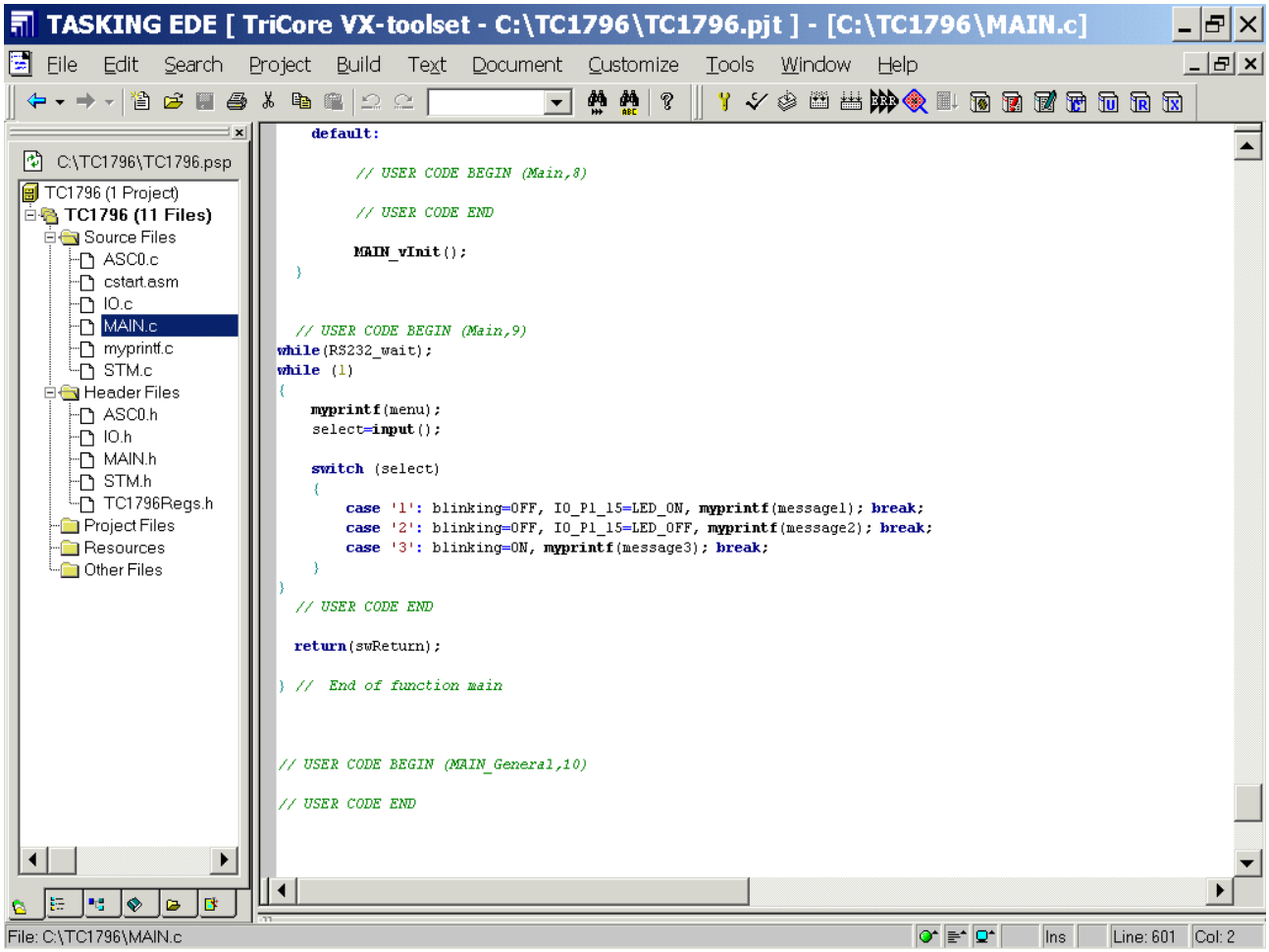
Start Tasking EDE and open the project:

File – Open Project Space

Look in: select C:\TC1796
File name: select TC1796.psp



Open


TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\MAIN.c]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1796\TC1796.psp
TC1796 (1 Project)
TC1796 (11 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
TC1796Regs.h
Project Files
Resources
Other Files
default:
// USER CODE BEGIN (Main,8)
// USER CODE END
MAIN_vInit();
}
// USER CODE BEGIN (Main,9)
while(RS232_wait);
while (1)
{
myprintf(menu);
select=input();
switch (select)
{
case '1': blinking=OFF, IO_P1_15=LED_ON, myprintf(message1); break;
case '2': blinking=OFF, IO_P1_15=LED_OFF, myprintf(message2); break;
case '3': blinking=ON, myprintf(message3); break;
}
}
// USER CODE END
return(swReturn);
} // End of function main
// USER CODE BEGIN (MAIN_General,10)
// USER CODE END
File: C:\TC1796\MAIN.c
Ins
Line: 601
Col: 2

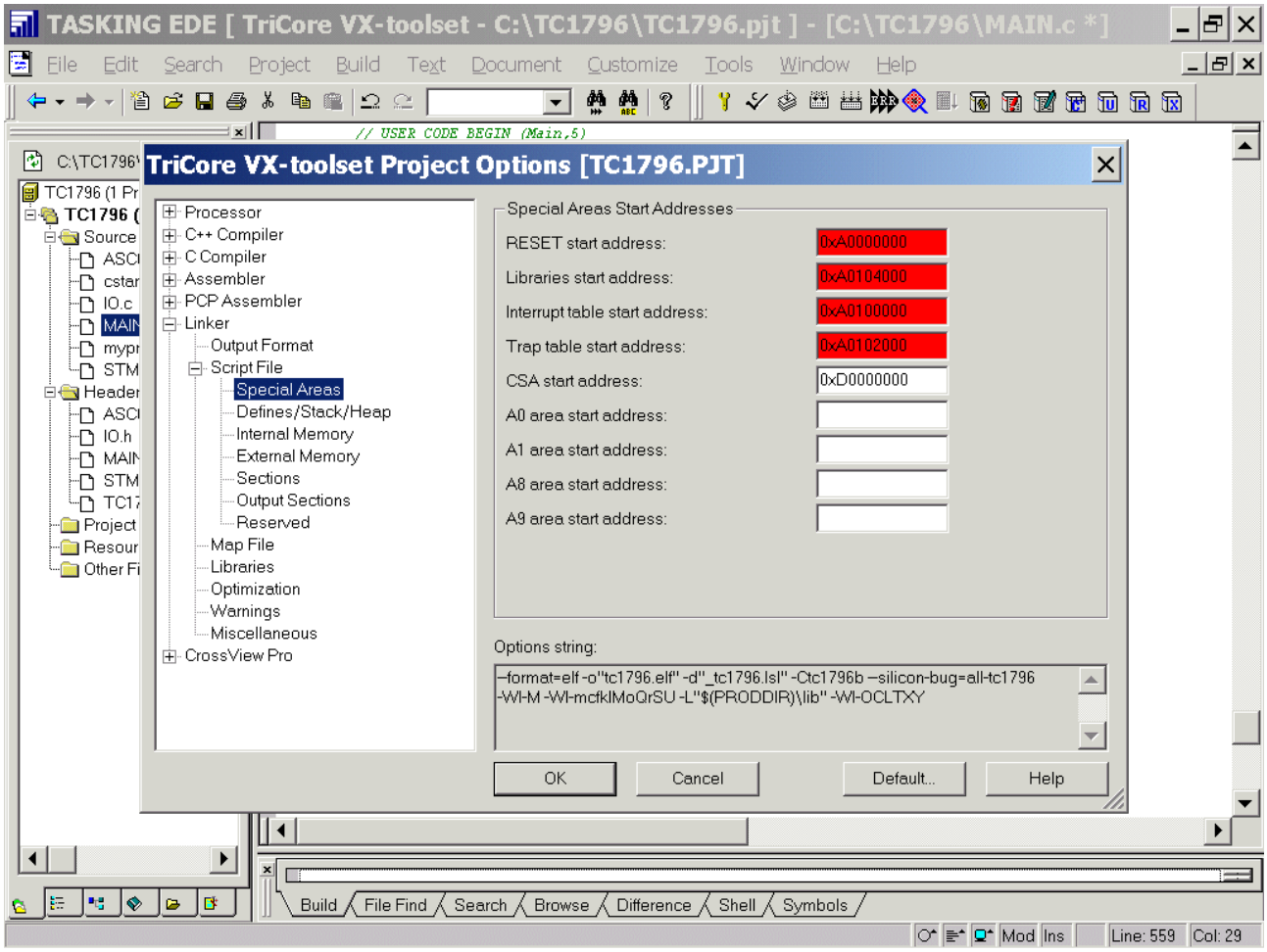
Configure Compiler, Assembler, Linker, Locater and Build – Control:

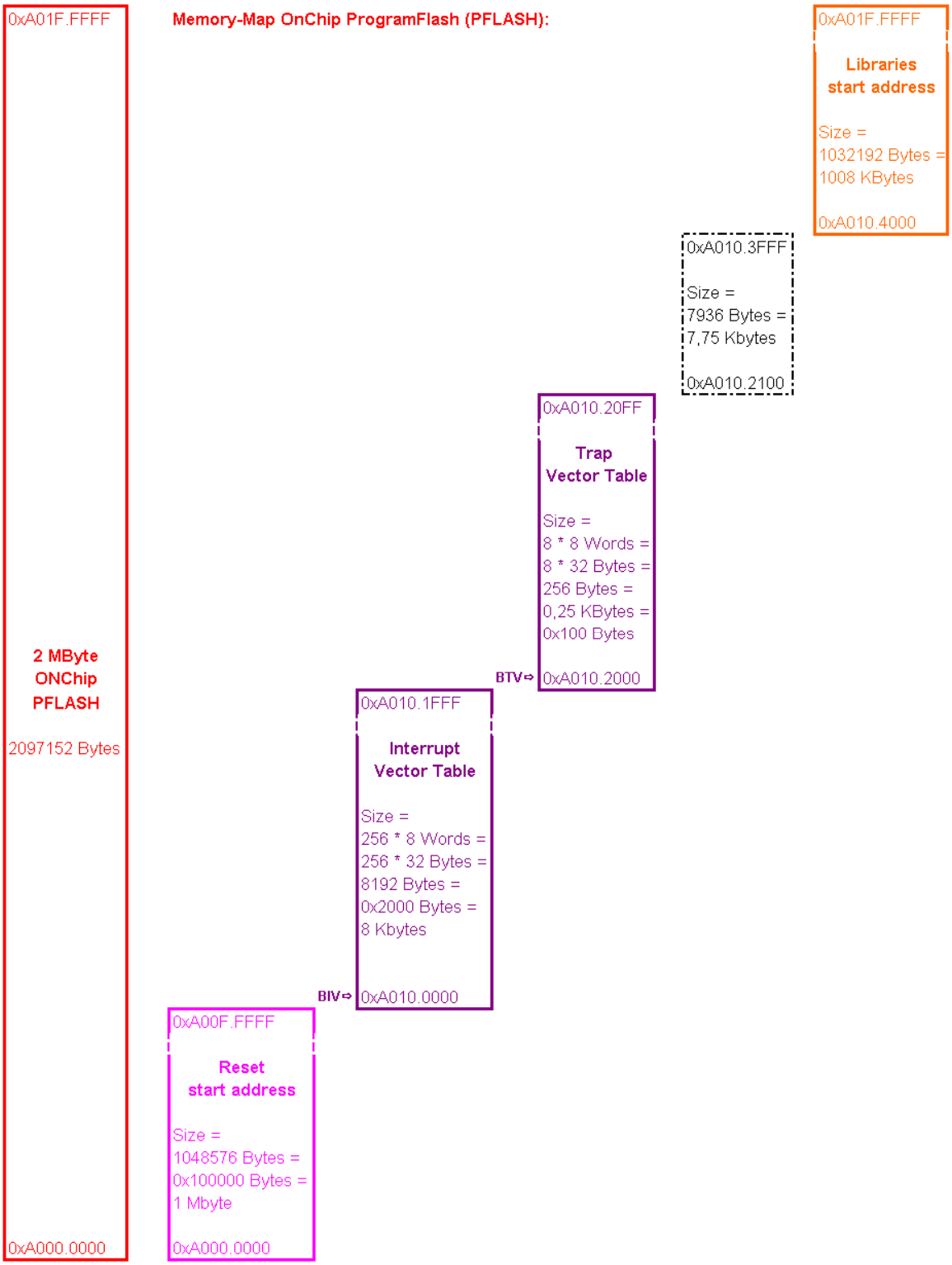
Project – Project Options

Table 9-2 SPB/RPB Address Map of Segment 0 to 14 (cont'd)

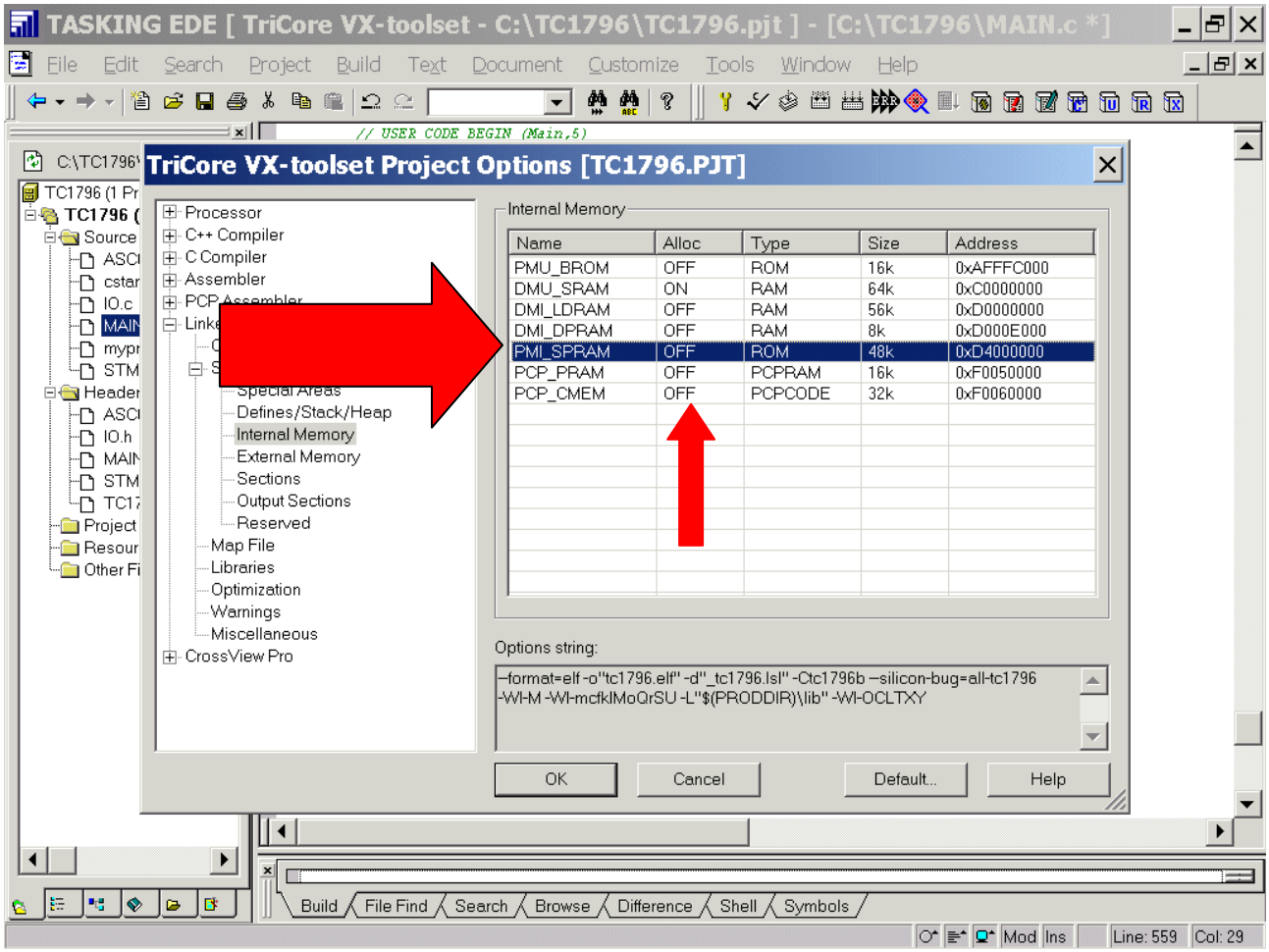
Seg-ment	Address Range	Size	Description	Access Type	
				Read	Write
10	A000 0000$_H$ - A01F FFFF$_H$	2 Mbyte	Program Flash (PFLASH)	access	access[1)]

Linker: Script File: Special Areas: RESET start address: insert 0xA0000000 (PFLASH)
Linker: Script File: Special Areas: Libraries start address: insert 0xA0104000 (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)

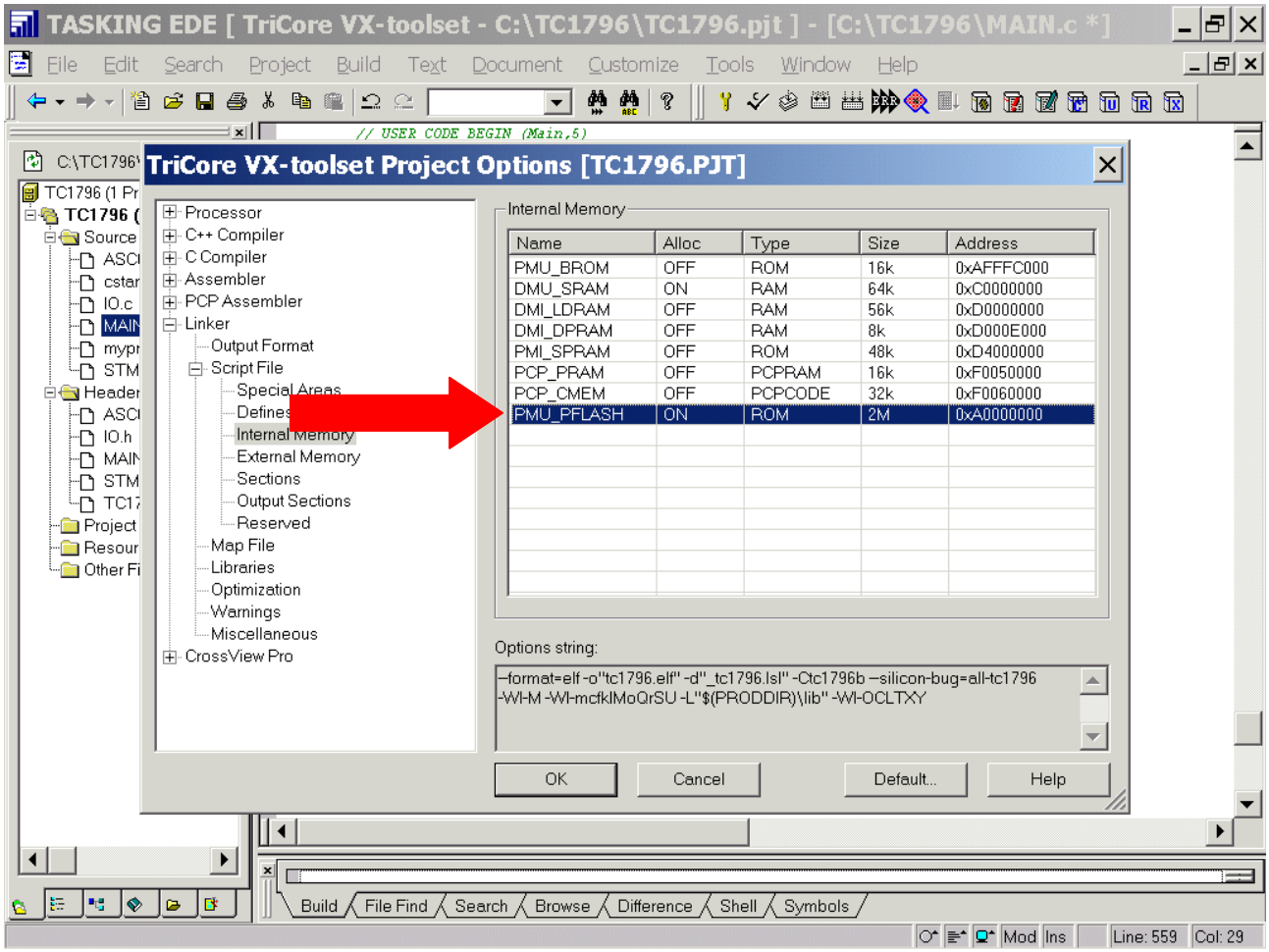



Memory-Map OnChip ProgramFlash (PFLASH):
0xA01F.FFFF
2 MByte
ONChip
PFLASH
2097152 Bytes
0xA000.0000
0xA00F.FFFF
Reset
start address
Size =
1048576 Bytes =
0x100000 Bytes =
1 Mbyte
0xA000.0000
BIV⇨
0xA010.0000
0xA010.1FFF
Interrupt
Vector Table
Size =
256 * 8 Words =
256 * 32 Bytes =
8192 Bytes =
0x2000 Bytes =
8 Kbytes
BTV⇨
0xA010.2000
0xA010.20FF
Trap
Vector Table
Size =
8 * 8 Words =
8 * 32 Bytes =
256 Bytes =
0,25 KBytes =
0x100 Bytes
0xA010.2100
0xA010.3FFF
Size =
7936 Bytes =
7,75 Kbytes
0xA010.4000
0xA01F.FFFF
Libraries
start address
Size =
1032192 Bytes =
1008 KBytes

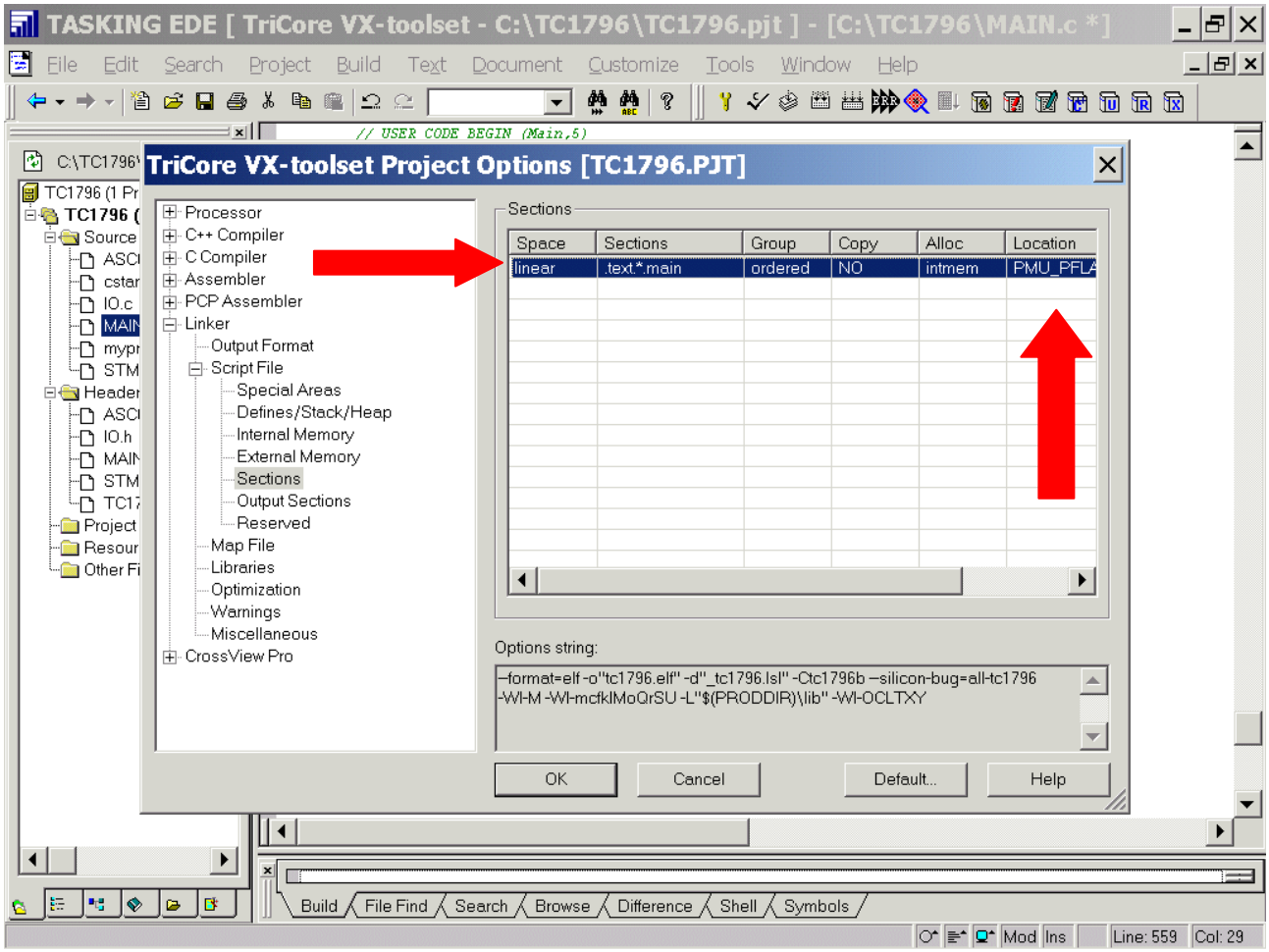
Linker: Script File: Internal Memory: PMI_SPRAM: Alloc: select OFF



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 2M
Linker: Script File: Internal Memory: Address insert 0xA0000000

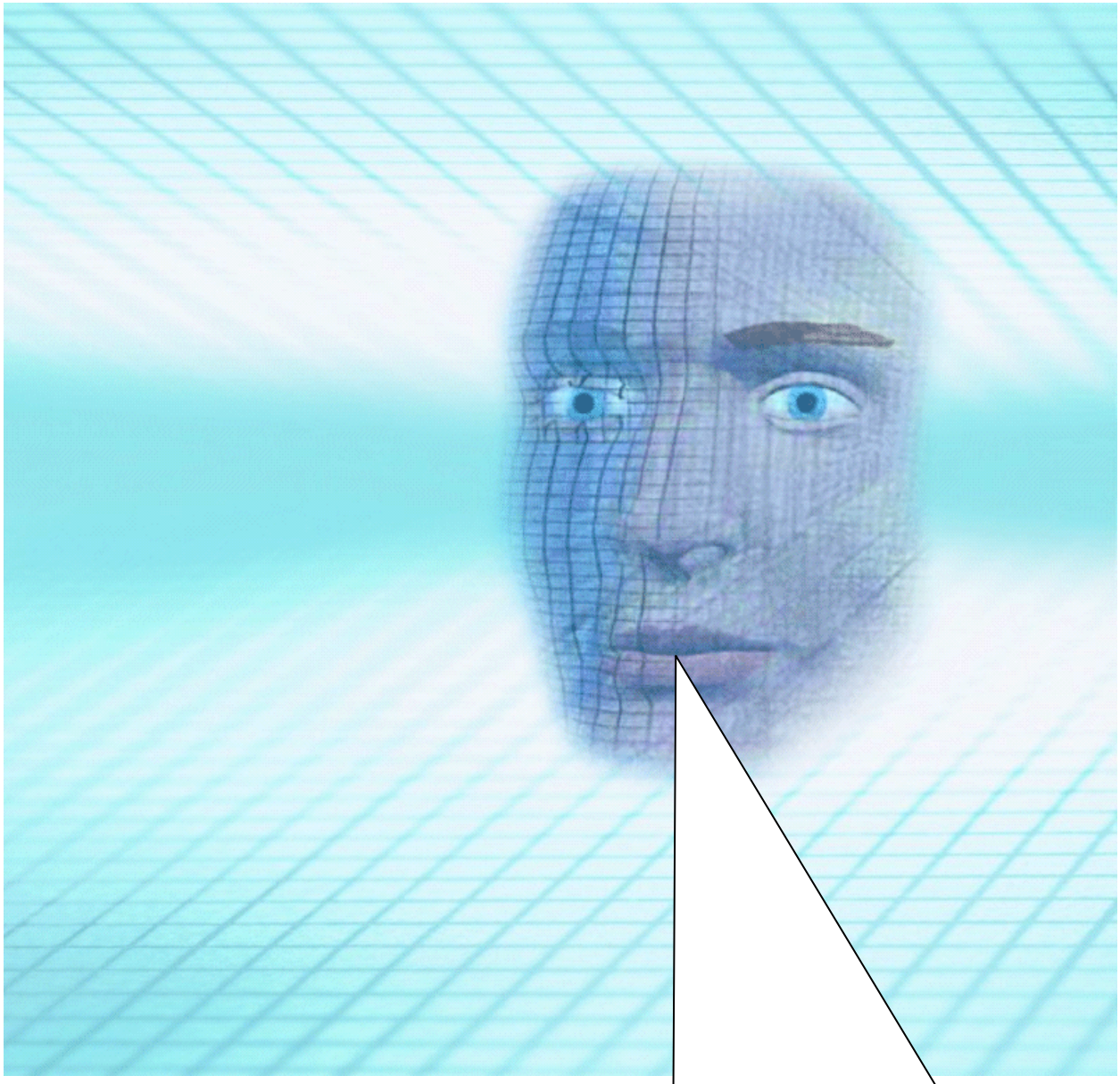


Linker: Script File: Sections: linear: Location: insert PMU_PFLASH



OK

Insert your application specific program:



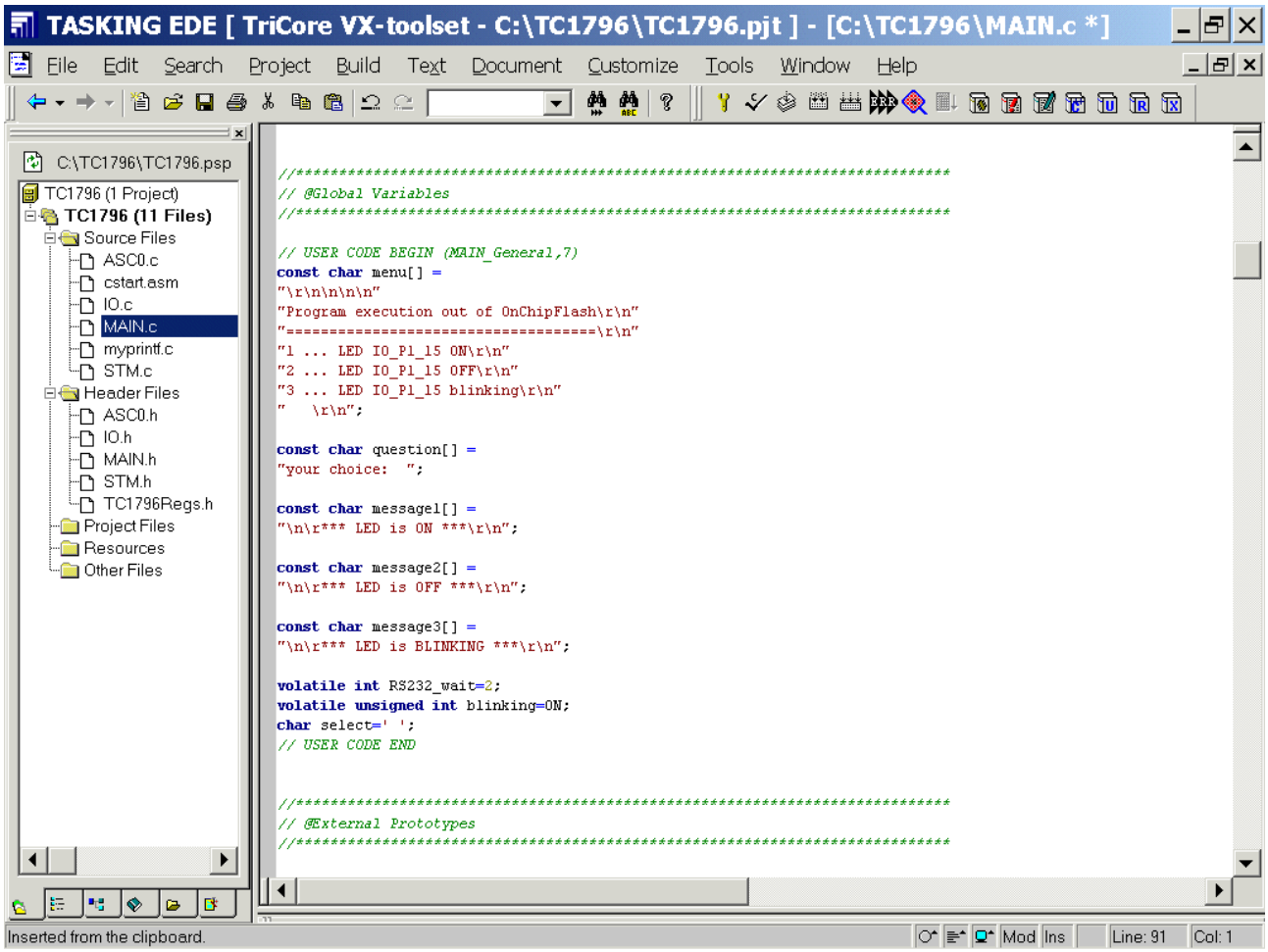
Note:
DAvE doesn’t change code which is inserted between ‘// USER CODE BEGIN’ and ‘// USER CODE END’. Therefore, whenever adding code to DAvE’s generated code, write it between ‘// USER CODE BEGIN’ and ‘// USER CODE END’.
If you wish to change DAvE´s generated code or add code outside these ‘USER CODE’ sections you will have to insert/modify your changes each time after letting DAvE regenerate code!

Double click: Main.c and change Global Variable menu
from

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of SPRAM\r\n"
"==============================\r\n"
"1 ... LED IO_P1_15 ON\r\n"
"2 ... LED IO_P1_15 OFF\r\n"
"3 ... LED IO_P1_15 blinking\r\n"
"   \r\n";
```

to

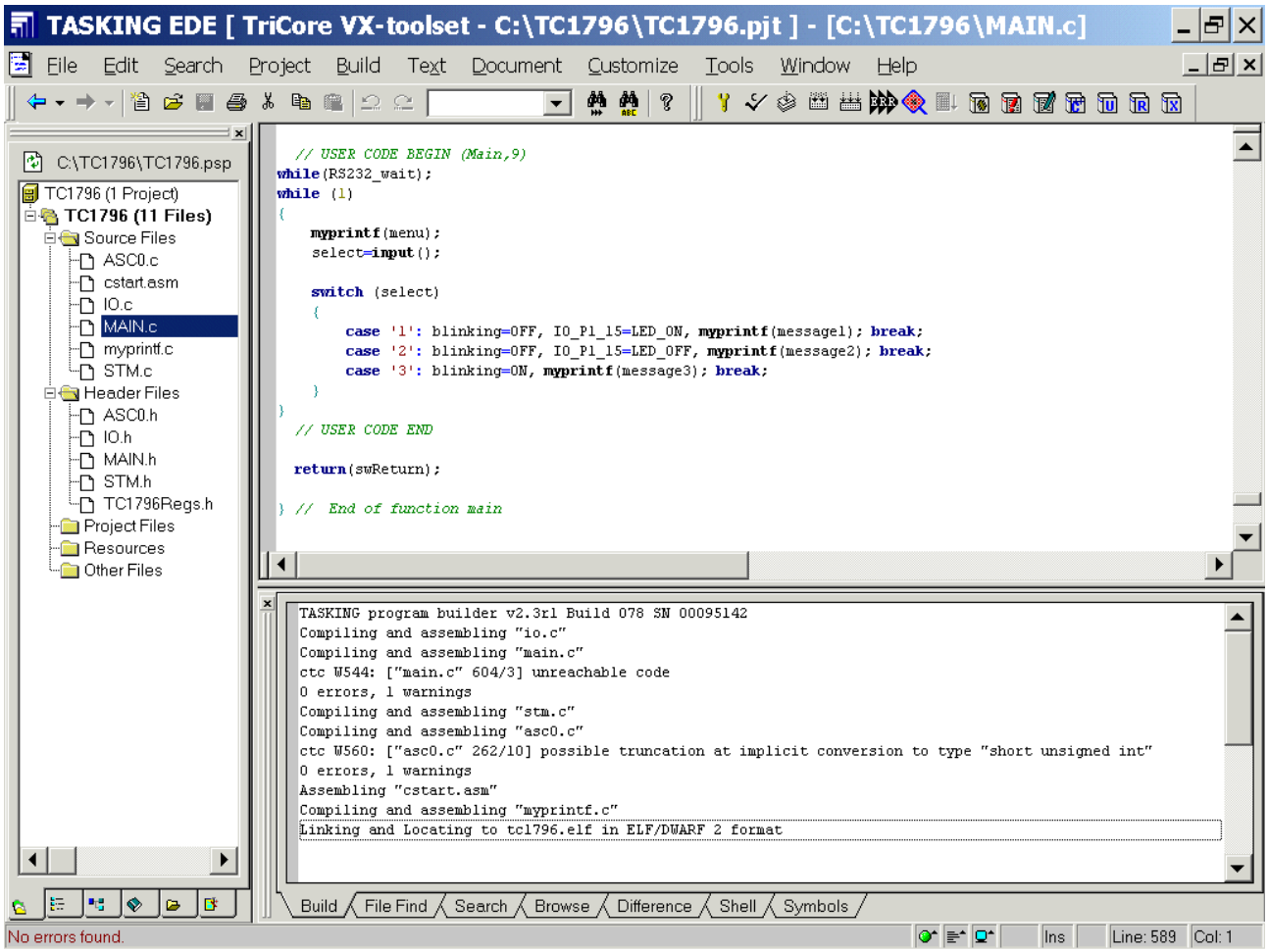
```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of OnChipFlash\r\n"
"====================================\r\n"
"1 ... LED IO_P1_15 ON\r\n"
"2 ... LED IO_P1_15 OFF\r\n"
"3 ... LED IO_P1_15 blinking\r\n"
"   \r\n";
```



Generate your application program:

Build
Rebuild

or

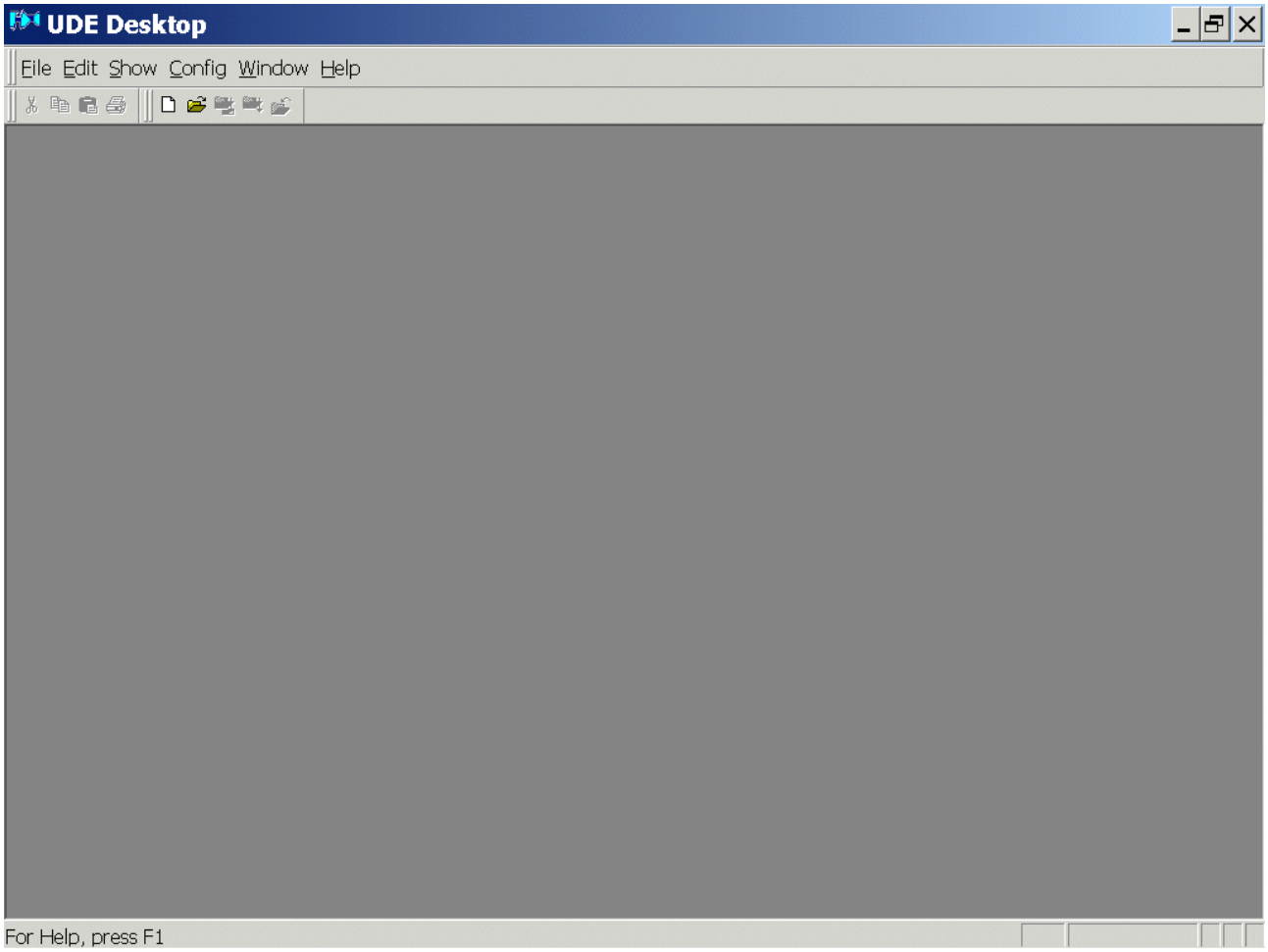


Now you can close your project and Tasking EDE:

File - Close Project Space
File – Exit

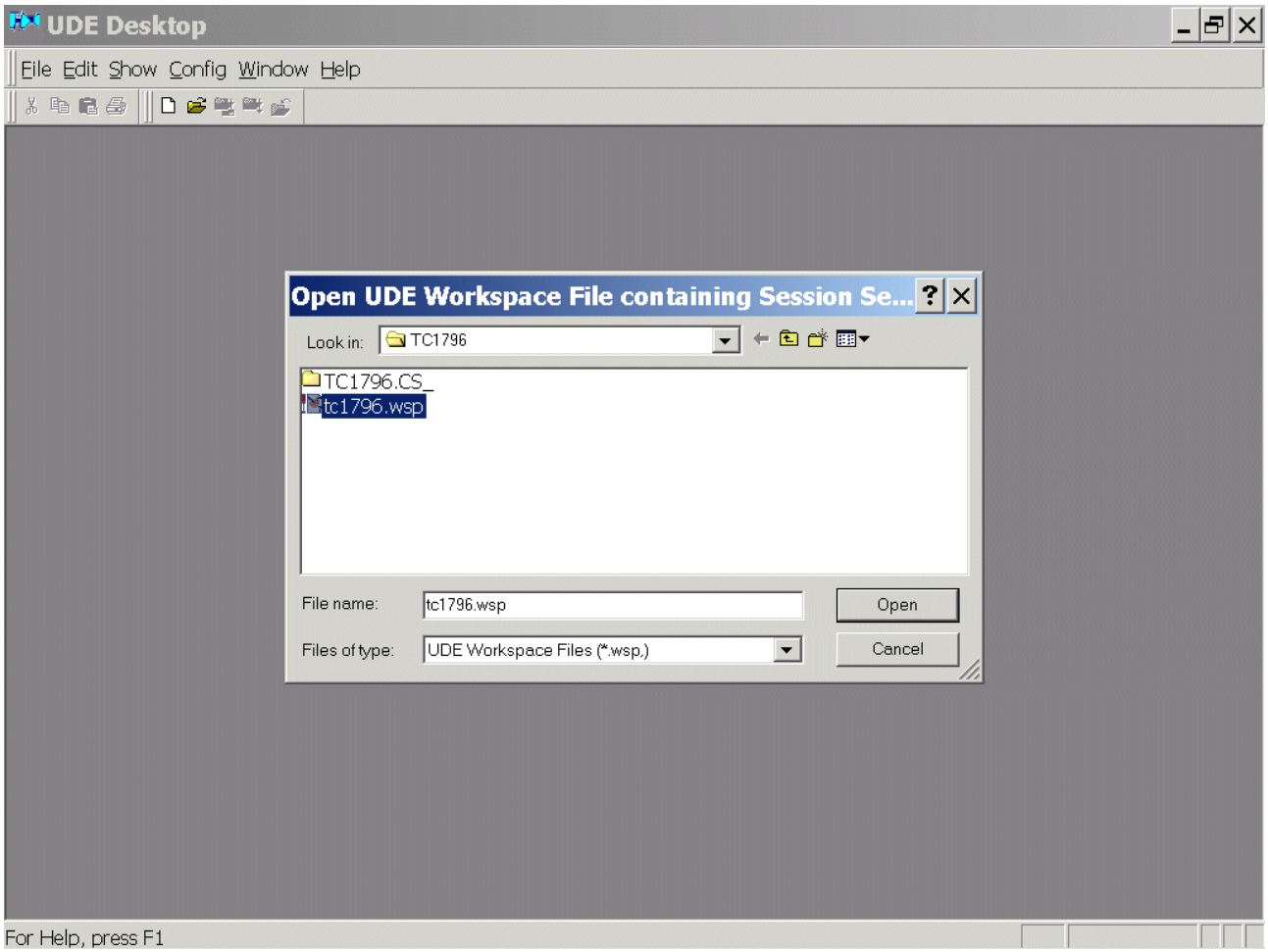
Programming is now complete. You can now load and run your program:

Start pls-Debugger



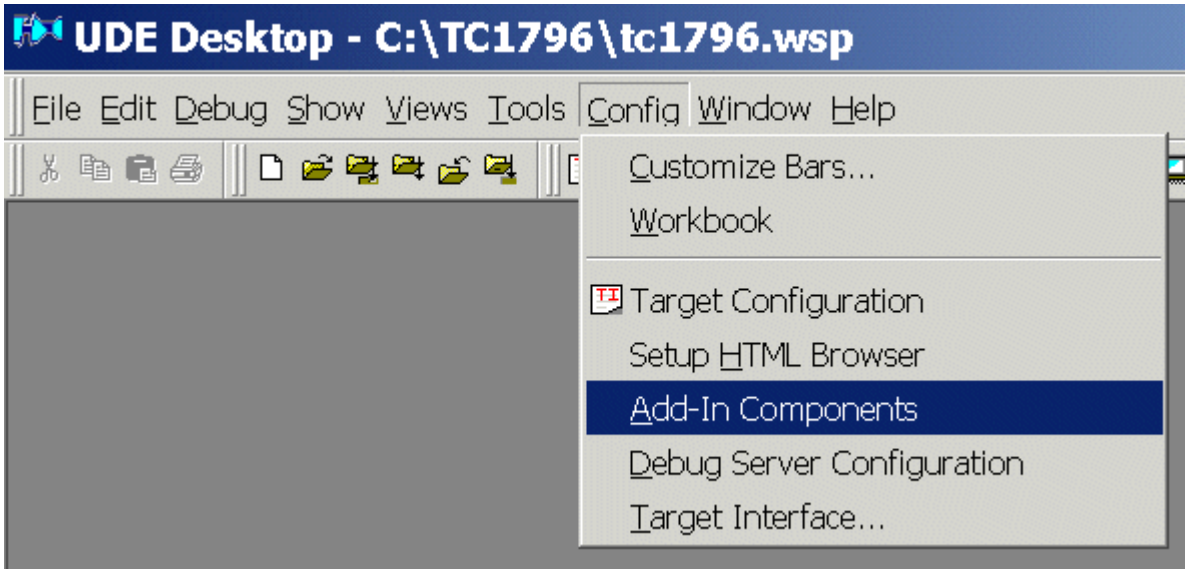
File – Open Workspace

Look in: select C:\TC1796
File name: select tc1796.wsp

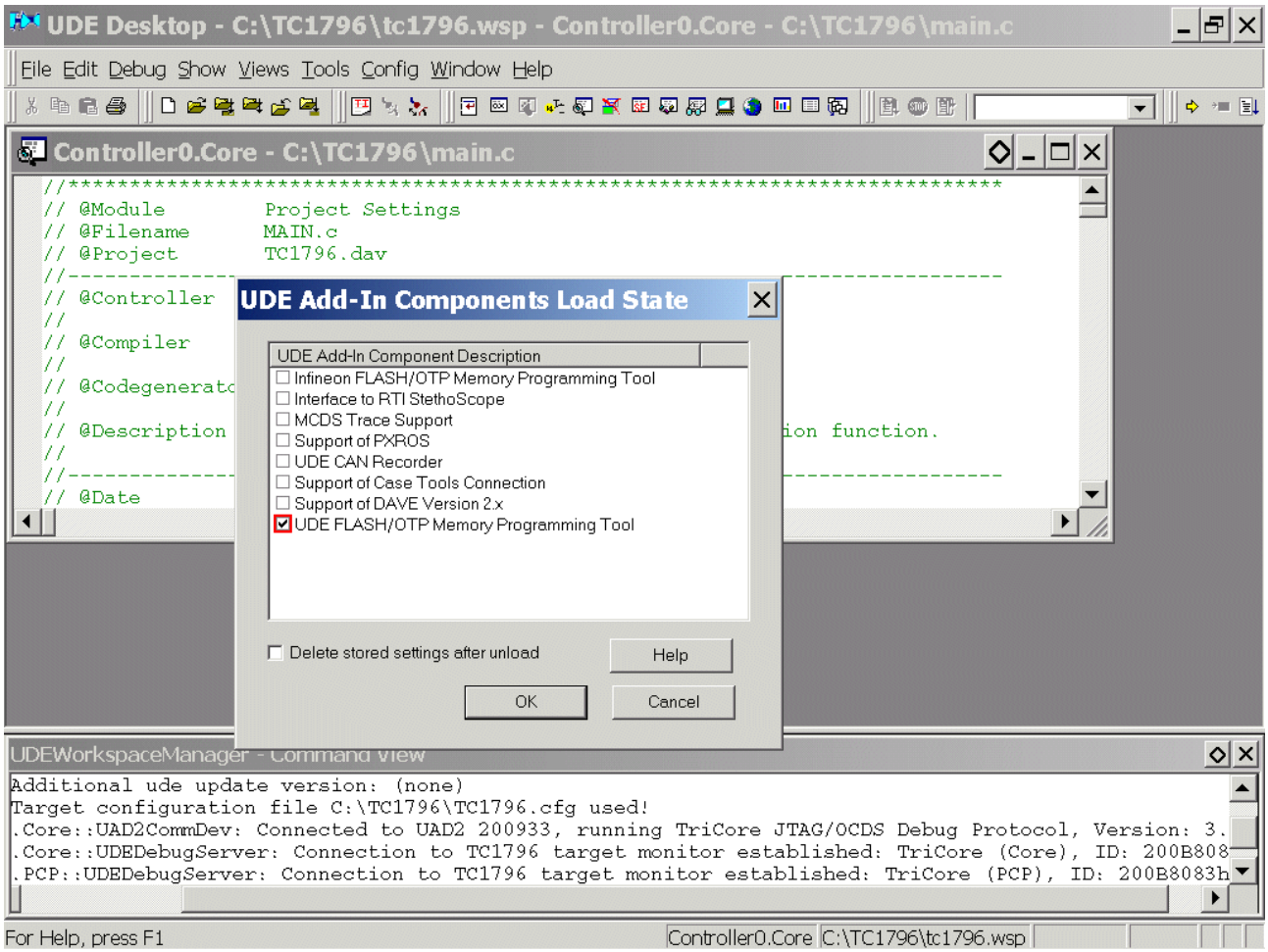


Open
Cancel

Config – Add-In Components

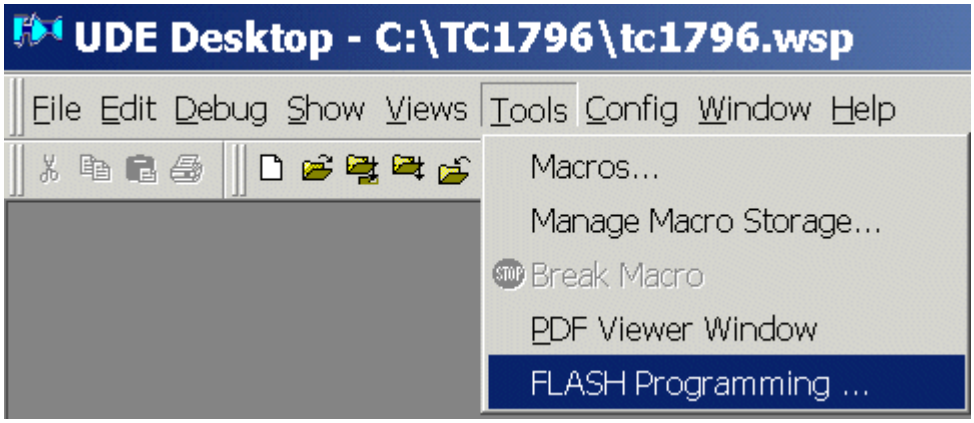


UDE Add-In Component Description click ✓ UDE FLASH/OTP Memory Programming Tool

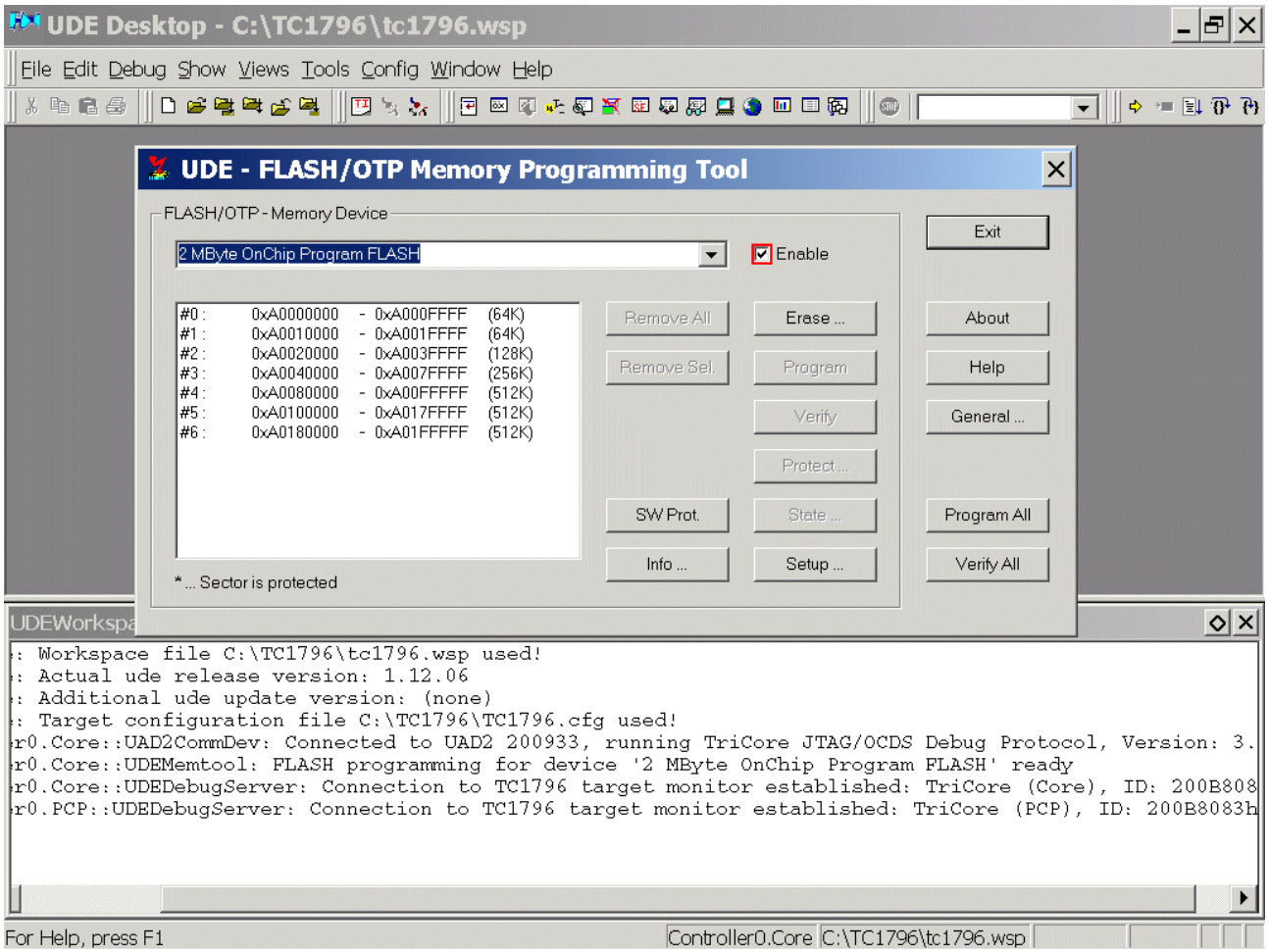


OK

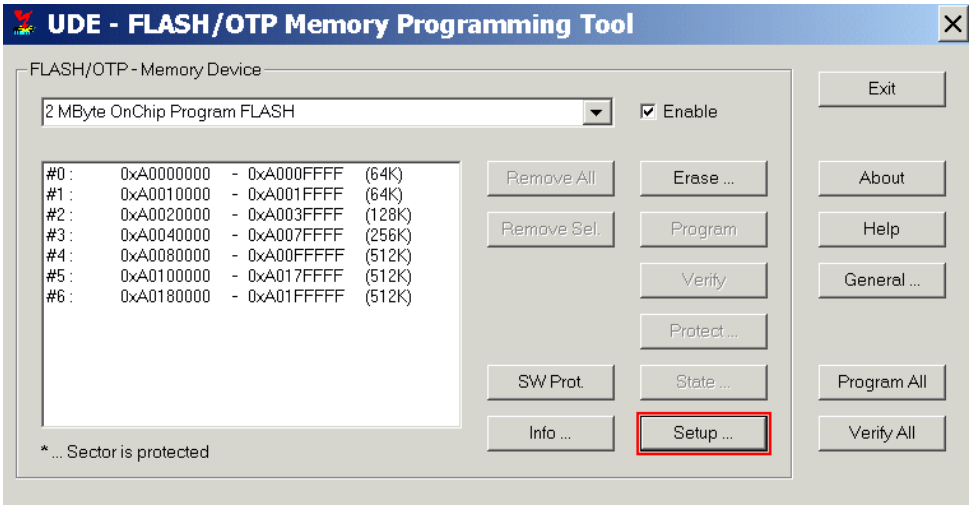
Tools – FLASH Programming …



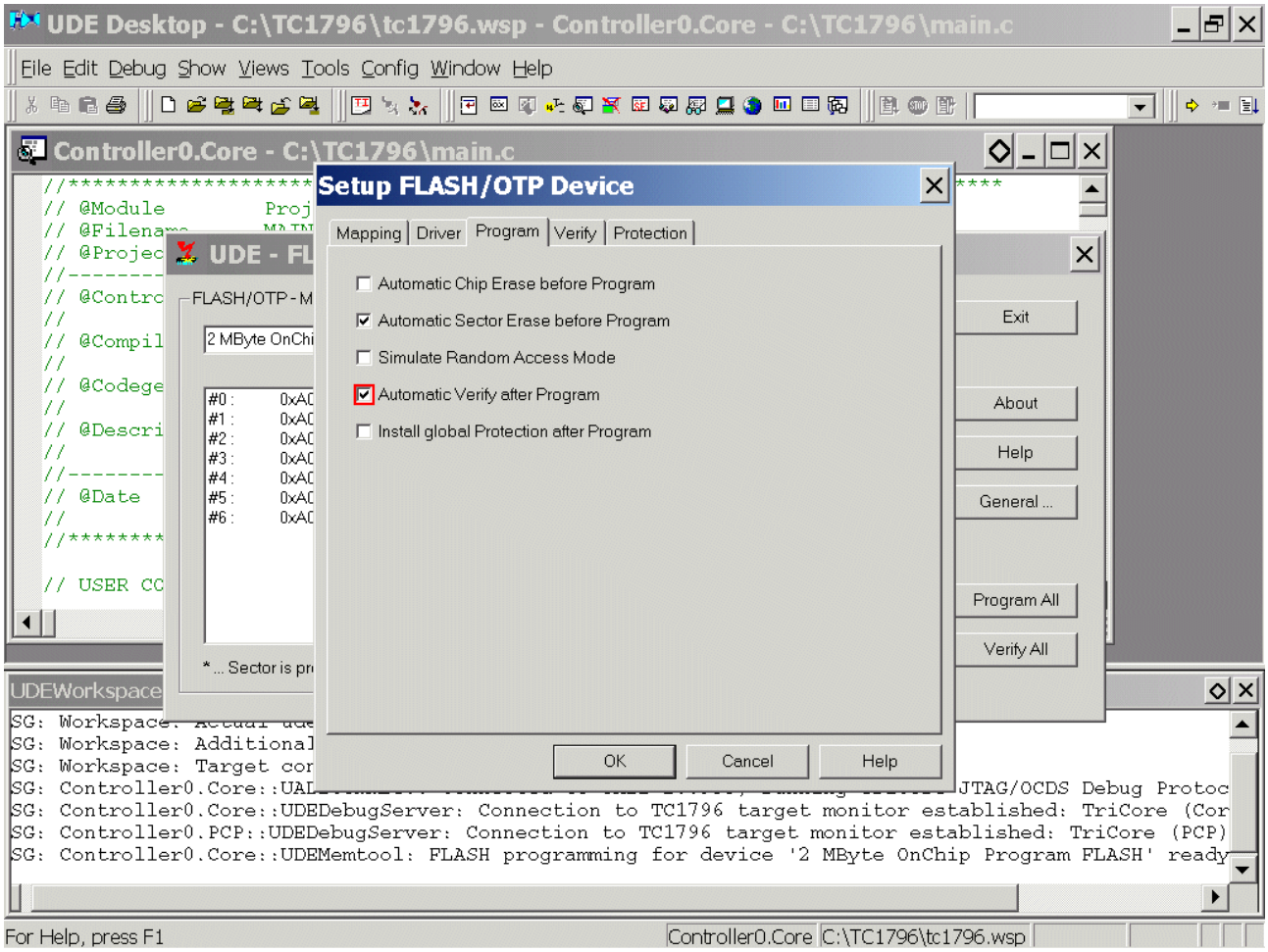
FLASH/OTP – Memory Device: select 2 MByte OnChip Program FLASH
FLASH/OTP – Memory Device: click ✓ Enable



Click Setup …

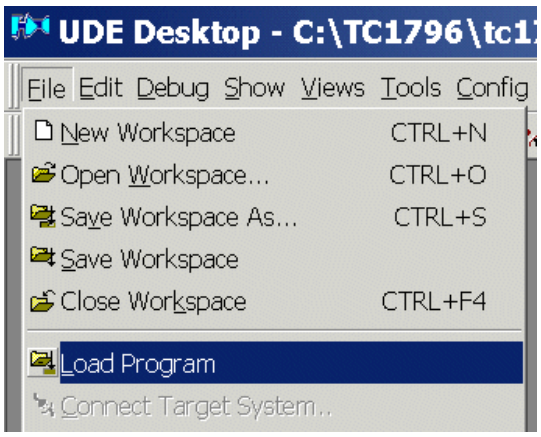


Program: click ✓ Automatic Verify after Program

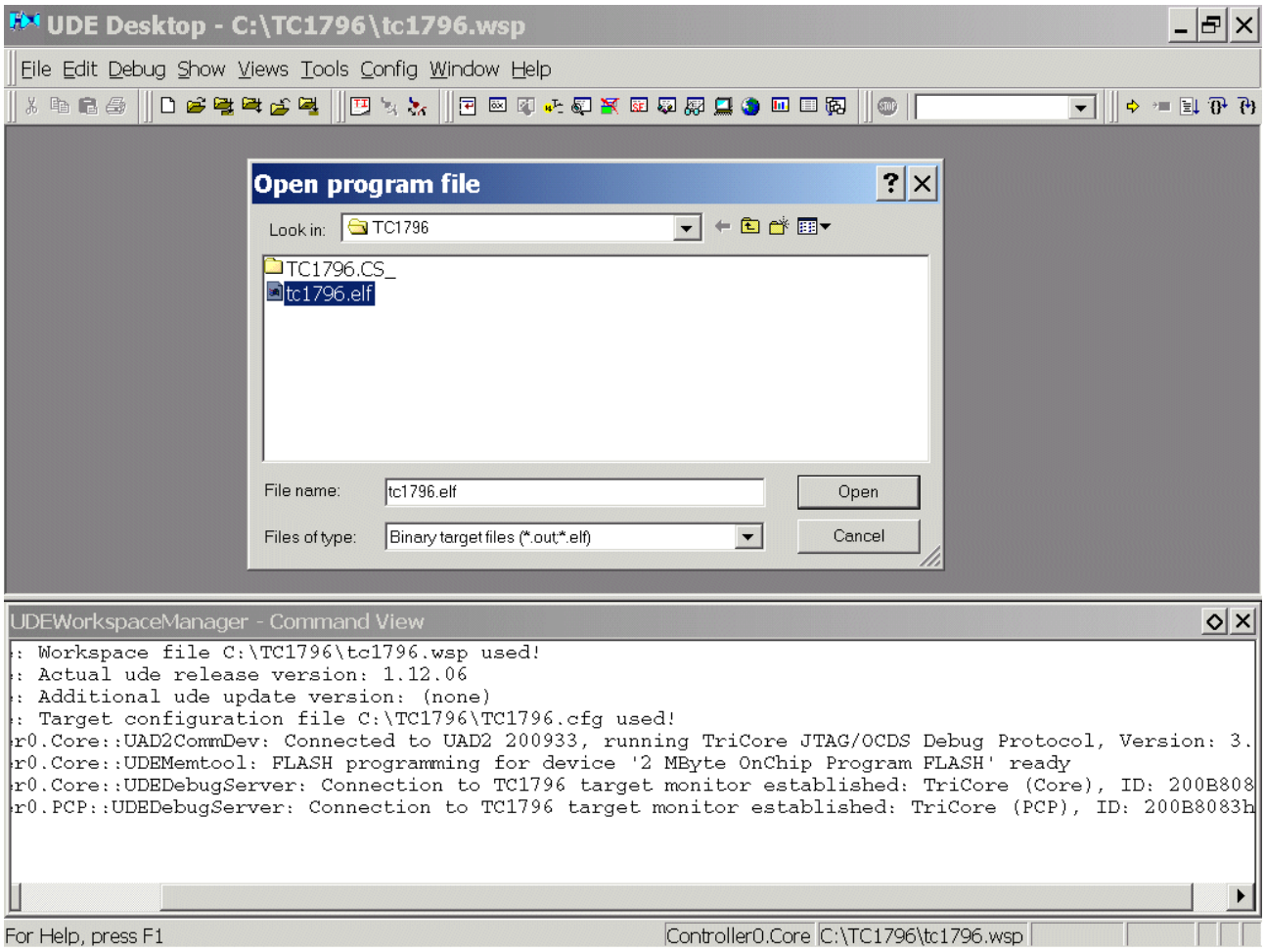


OK
Exit

File – Load Program

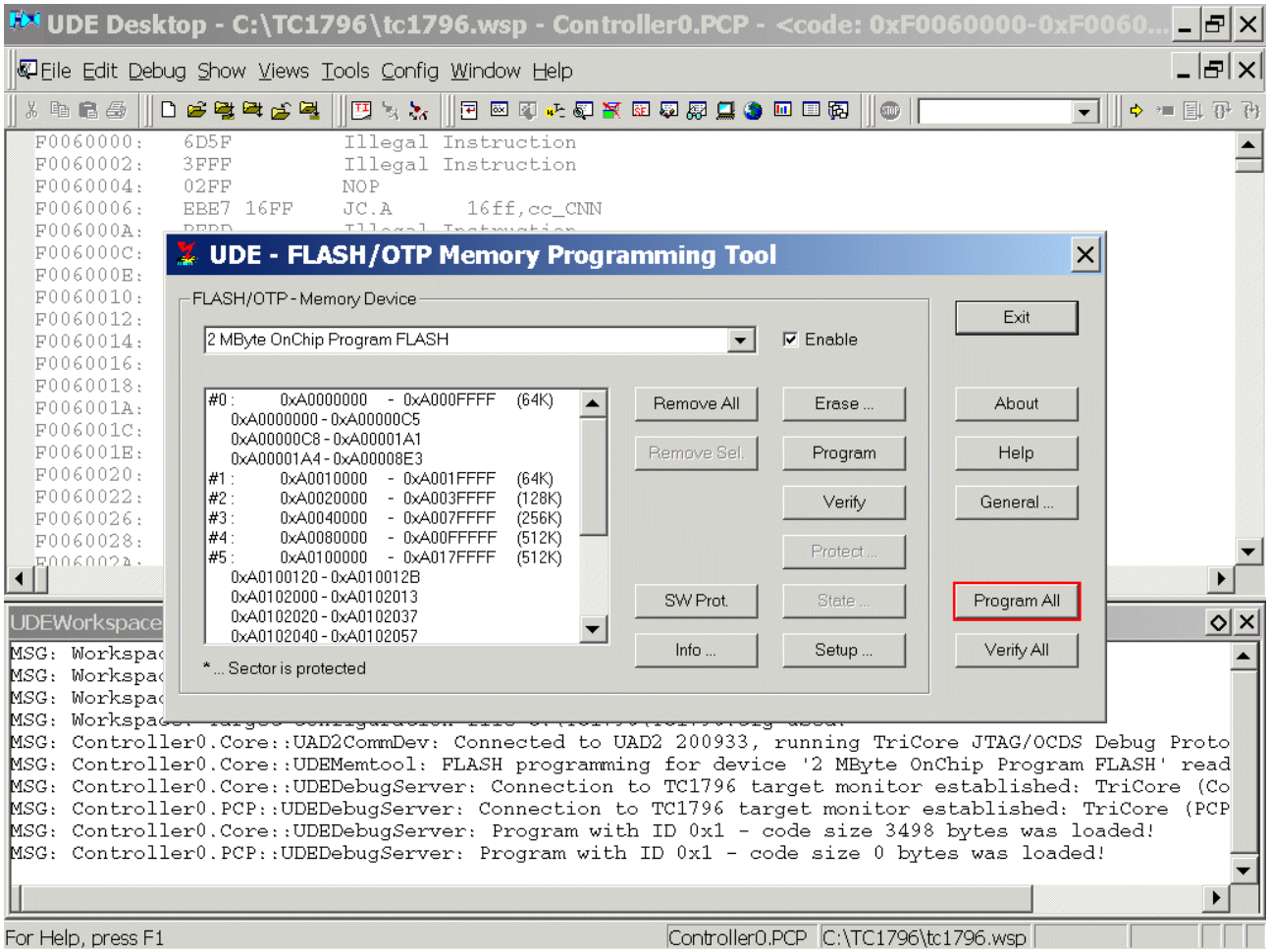


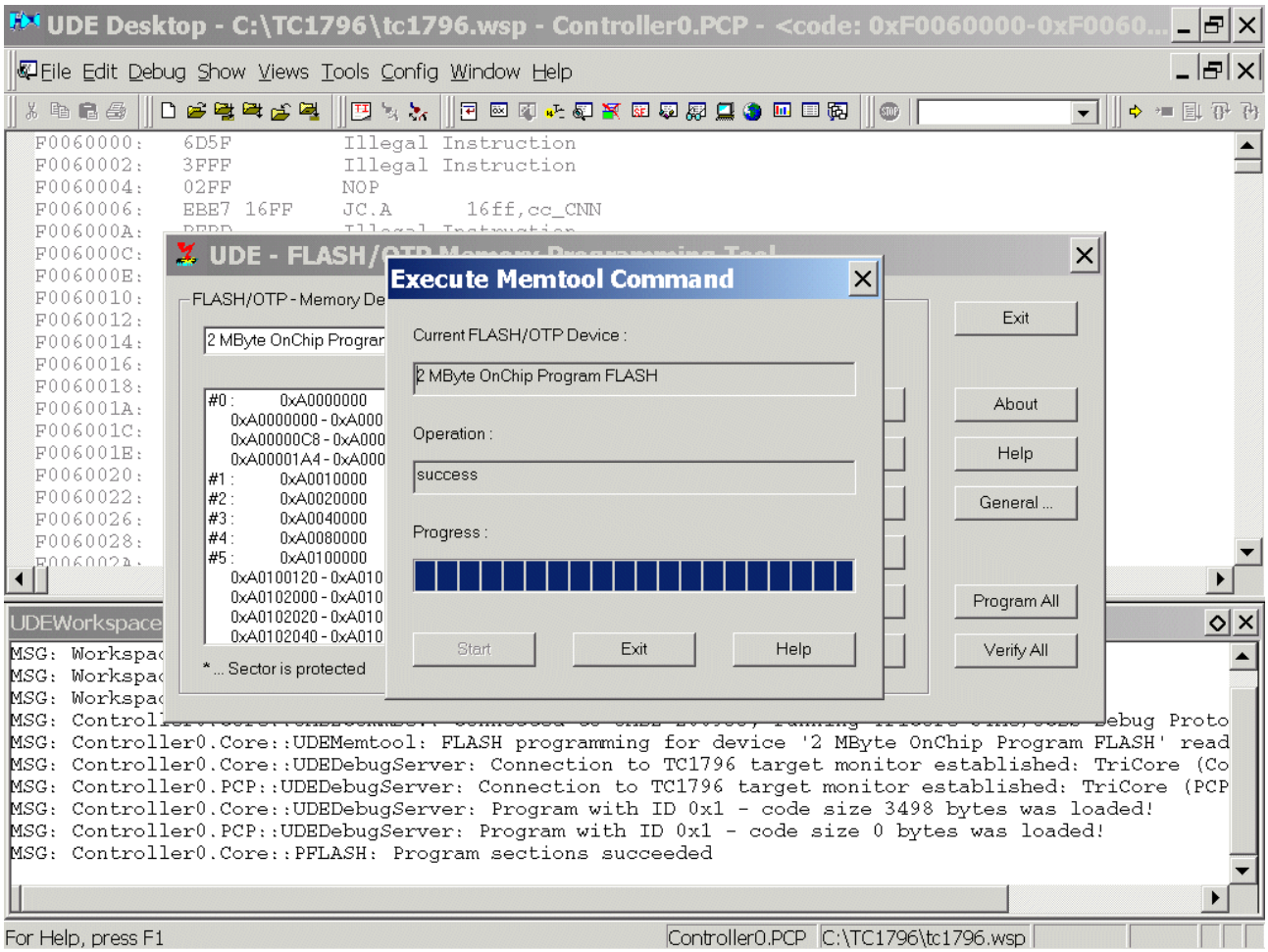
Look in: select TC1796
File name: select tc1796.elf



Open

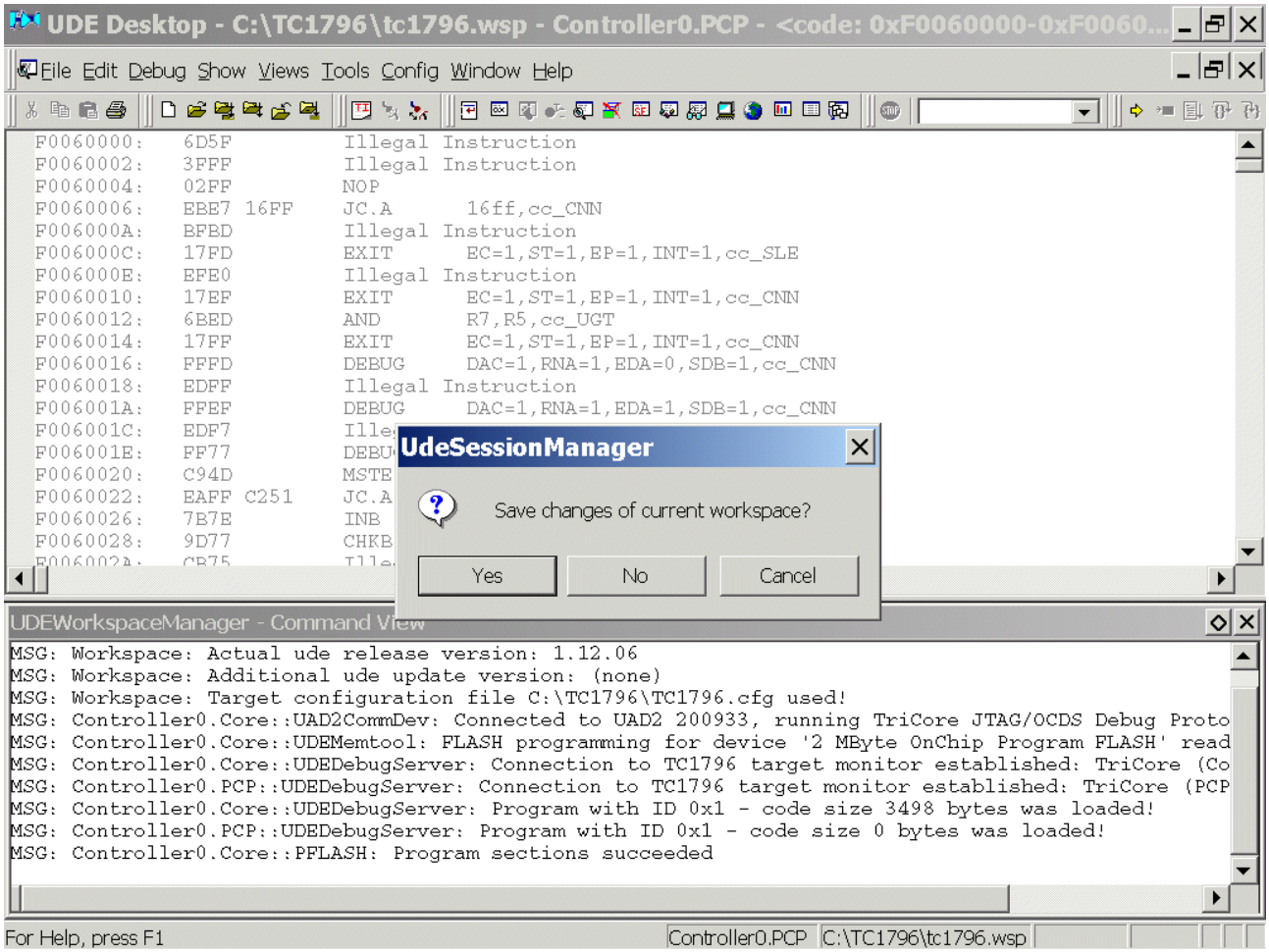
Click Program All




UDE Desktop - C:\TC1796\tc1796.wsp - Controller0.PCP - <code: 0xF0060000-0xF0060...
File Edit Debug Show Views Tools Config Window Help
F0060000: 6D5F Illegal Instruction
F0060002: 3FFF Illegal Instruction
F0060004: 02FF NOP
F0060006: EBE7 16FF JC.A 16ff,cc_CNN
UDE - FLASH/OTP Memory Programming Tool
Execute Memtool Command
Current FLASH/OTP Device :
2 MByte OnChip Program FLASH
Operation :
success
Progress :
Start
Exit
Help
Exit
About
Help
General ...
Program All
Verify All
* ... Sector is protected
UDEWorkspace
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '2 MByte OnChip Program FLASH' read
MSG: Controller0.Core::UDEDebugServer: Connection to TC1796 target monitor established: TriCore (Co
MSG: Controller0.PCP::UDEDebugServer: Connection to TC1796 target monitor established: TriCore (PCP
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3498 bytes was loaded!
MSG: Controller0.PCP::UDEDebugServer: Program with ID 0x1 - code size 0 bytes was loaded!
MSG: Controller0.Core::PFLASH: Program sections succeeded
For Help, press F1
Controller0.PCP
C:\TC1796\tc1796.wsp

Exit
Exit

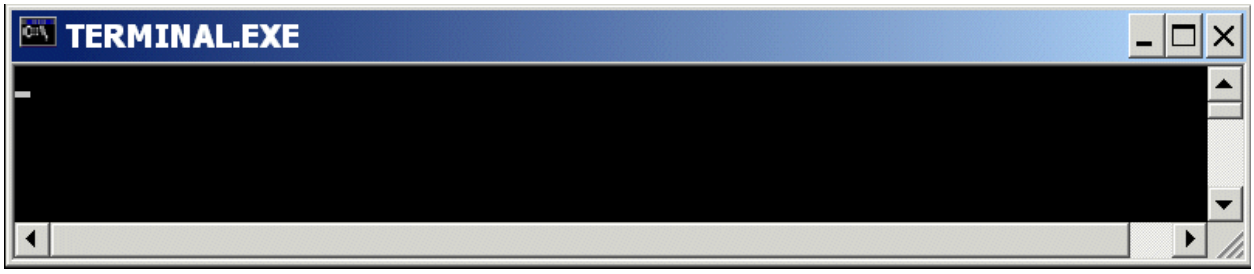
File – Close Workspace



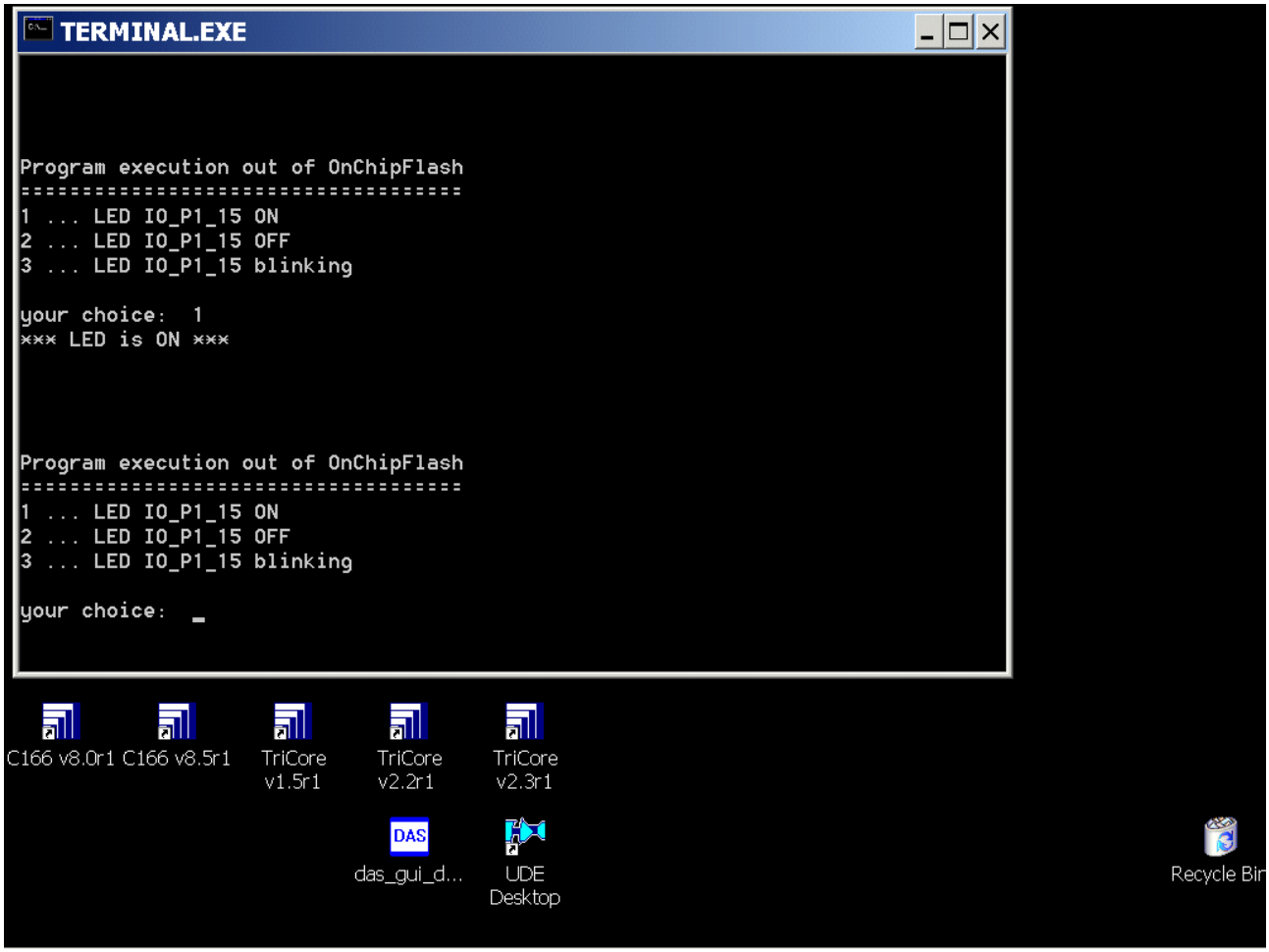
Yes

File – Exit

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

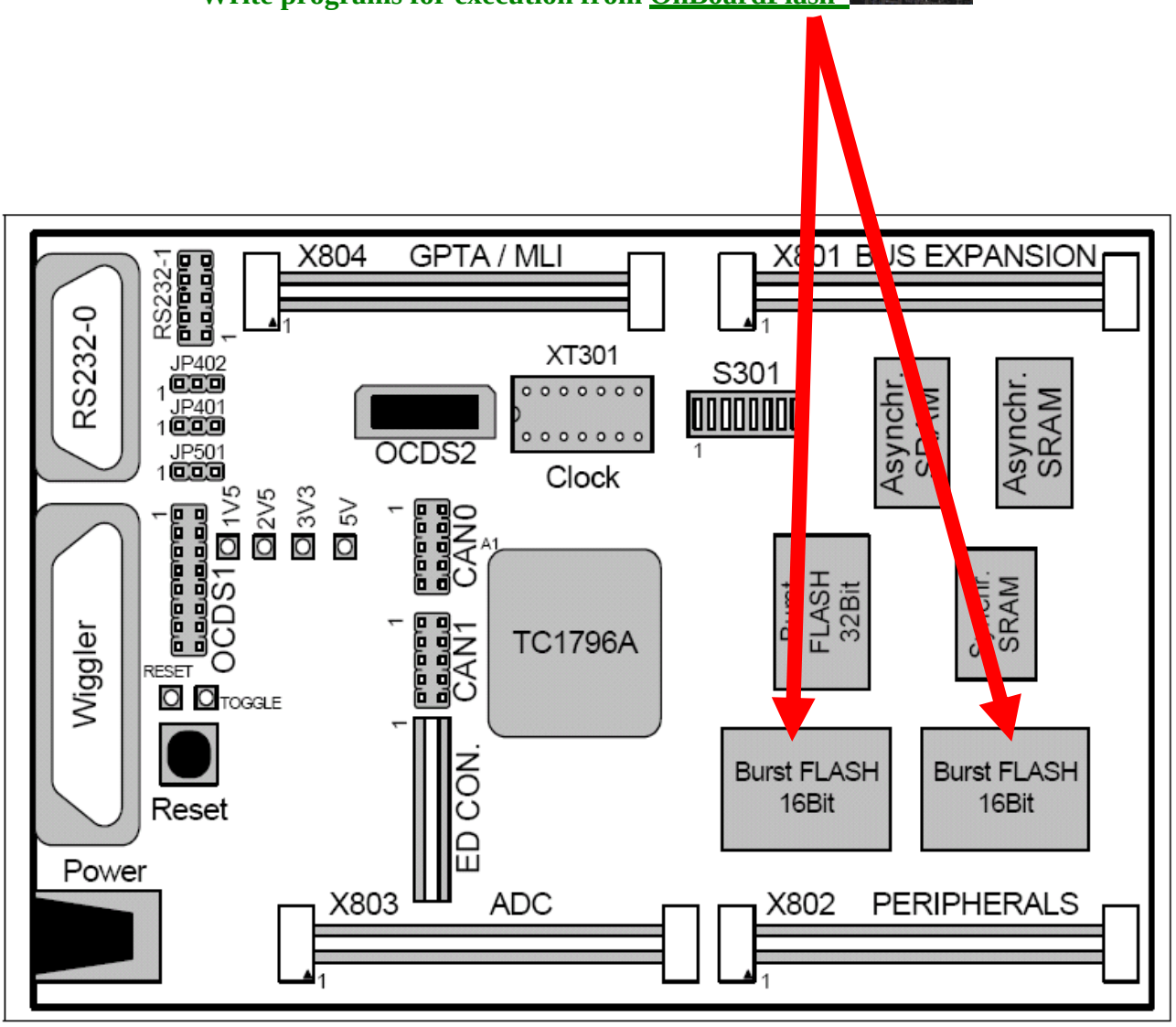


Power-On the Board and see the result:



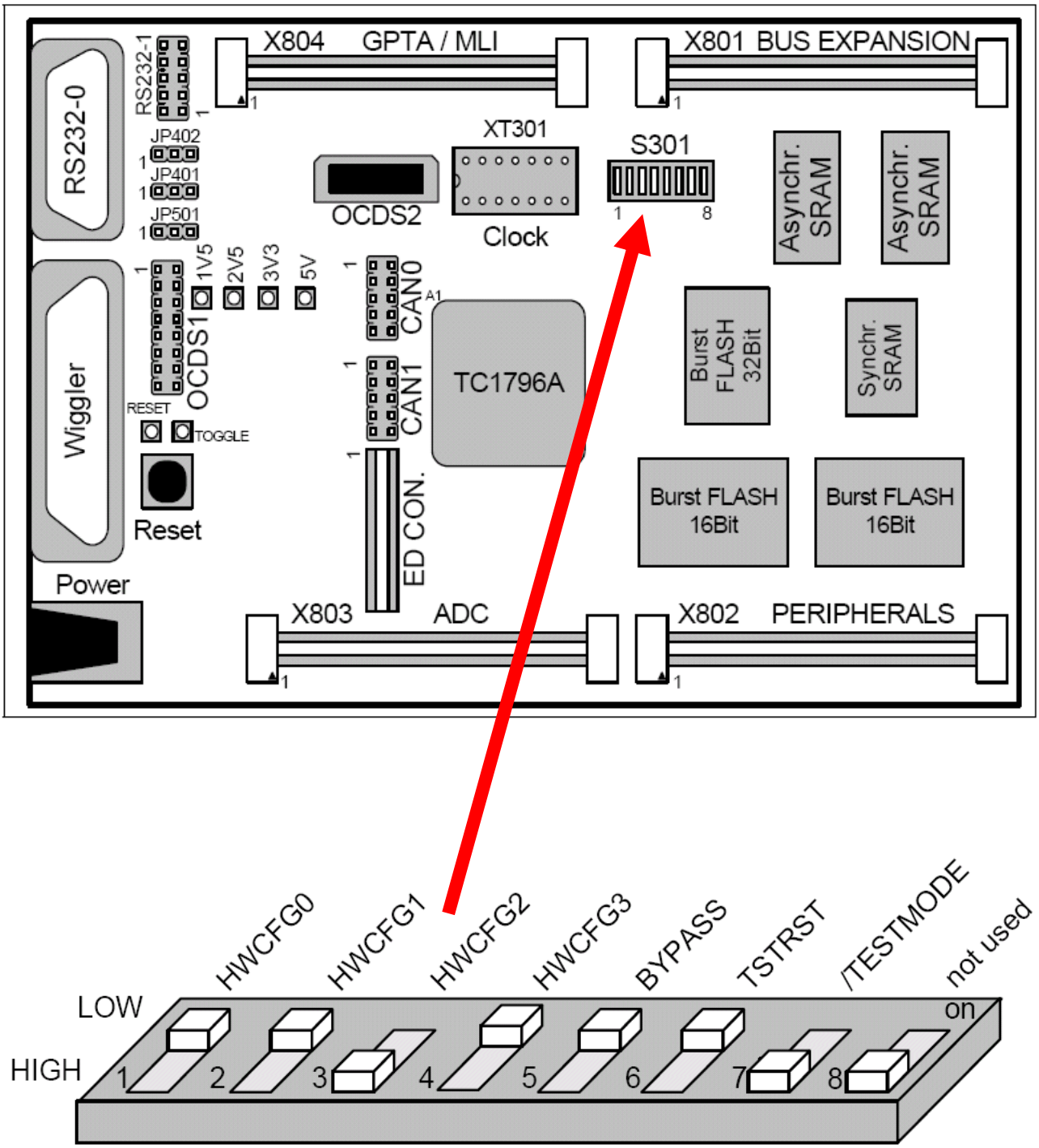
6.) Using of the TASKING - EDE Development Tools:

Write programs for execution from OnBoardFlash



TC1796-Execution-Environment: OnBoardFlash  :

Jumper Settings (HW-Configuration DIP-Switch):



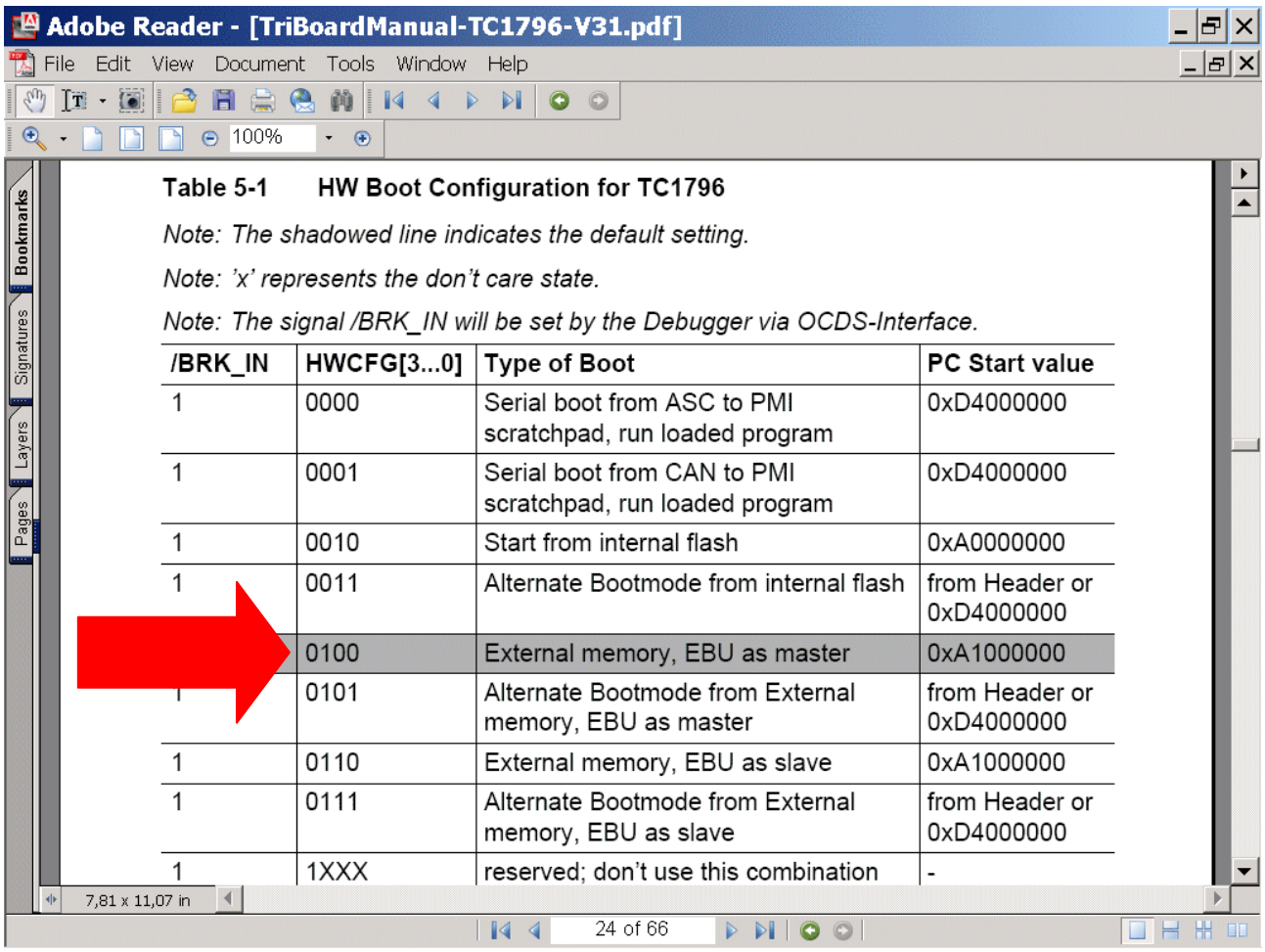


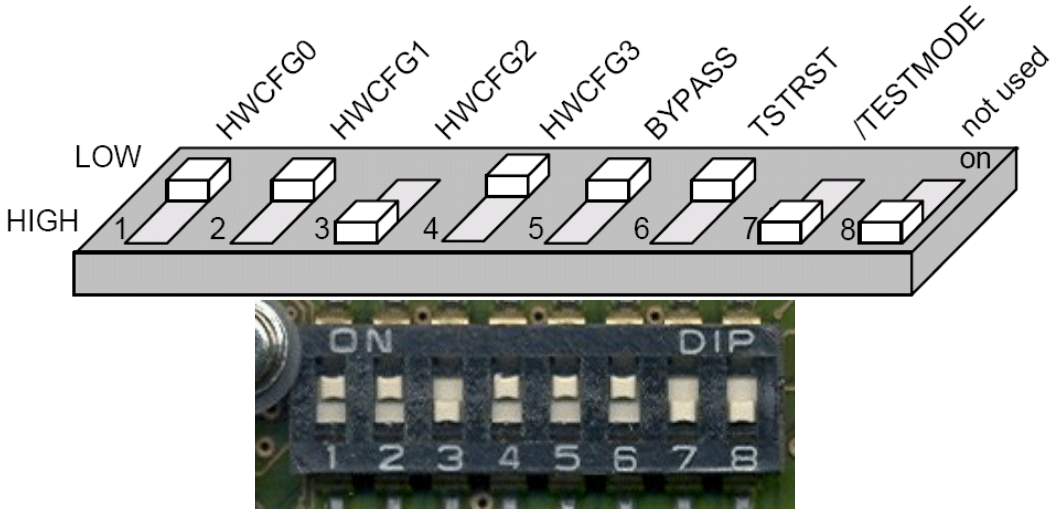
Table 5-1 HW Boot Configuration for TC1796

Note: The shadowed line indicates the default setting.

Note: 'x' represents the don't care state.

Note: The signal /BRK_IN will be set by the Debugger via OCDS-Interface.

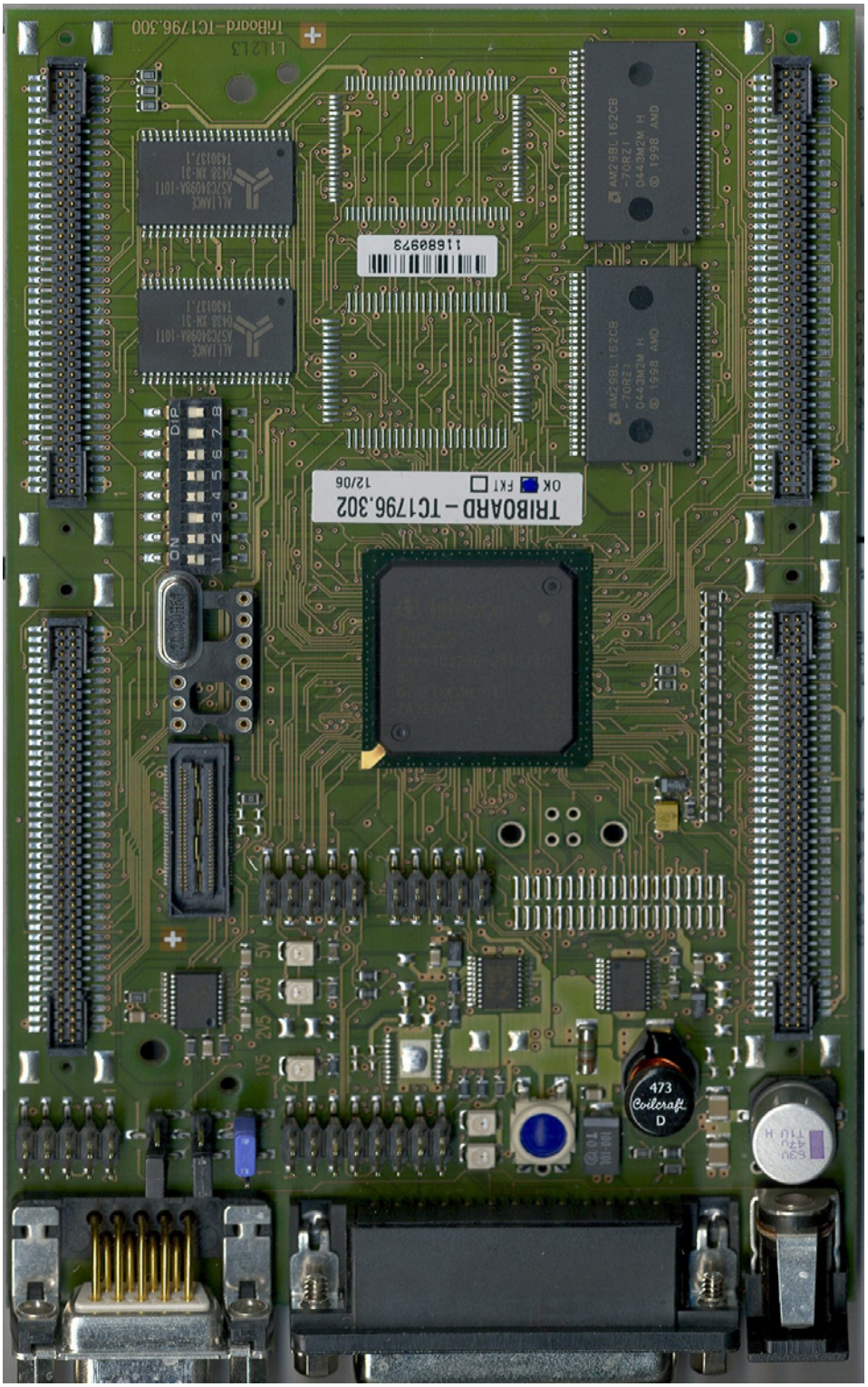
/BRK_IN	HWCFG[3...0]	Type of Boot	PC Start value
1	0000	Serial boot from ASC to PMI scratchpad, run loaded program	0xD4000000
1	0001	Serial boot from CAN to PMI scratchpad, run loaded program	0xD4000000
1	0010	Start from internal flash	0xA0000000
1	0011	Alternate Bootmode from internal flash	from Header or 0xD4000000
1	0100	External memory, EBU as master	0xA1000000
1	0101	Alternate Bootmode from External memory, EBU as master	from Header or 0xD4000000
1	0110	External memory, EBU as slave	0xA1000000
1	0111	Alternate Bootmode from External memory, EBU as slave	from Header or 0xD4000000
1	1XXX	reserved; don't use this combination	-



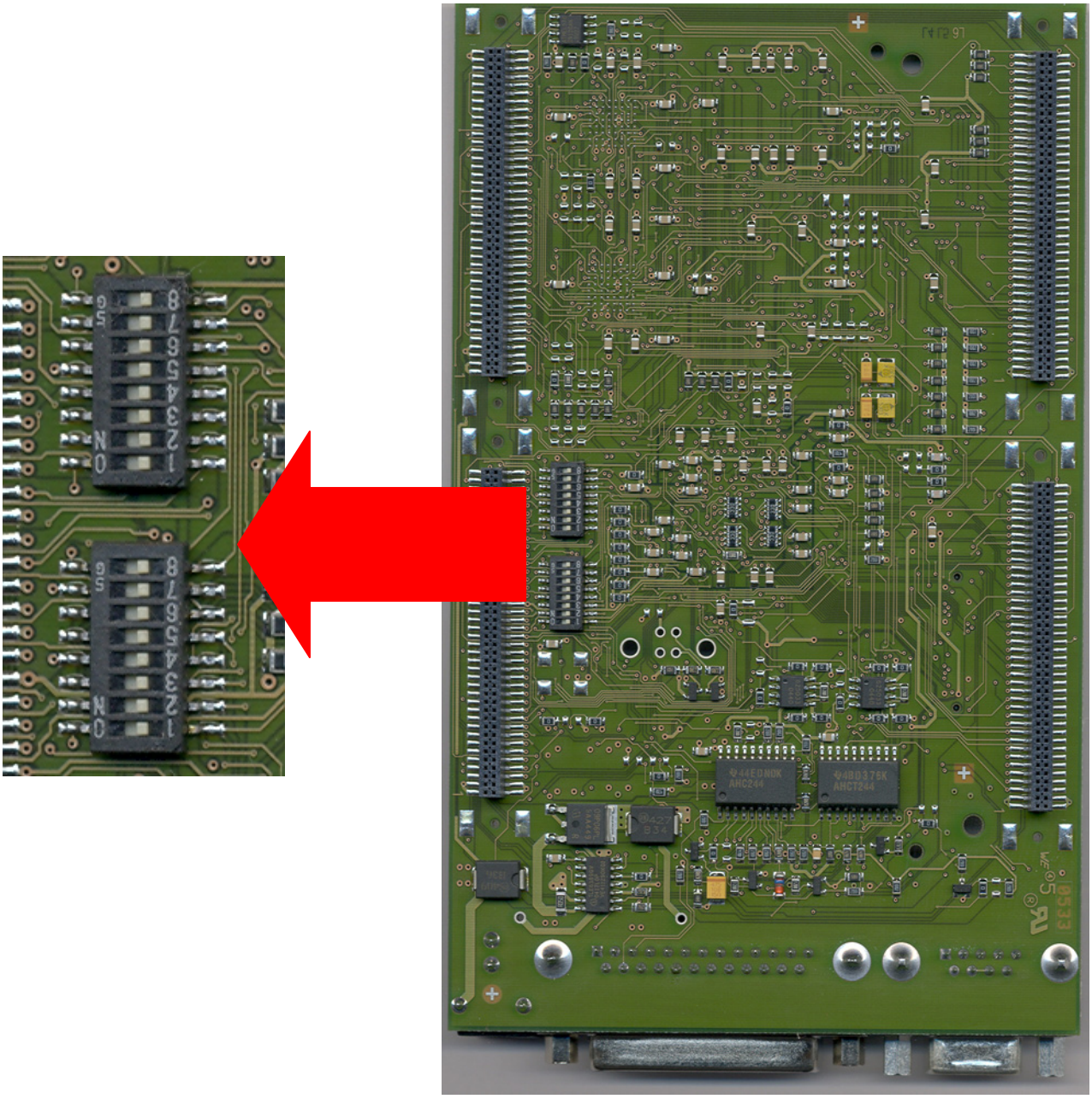
S301:
1, 2, 4, 5, 6 : ON
3, 7, 8 : OFF

TC1796-Execution-Environment: OnBoardFlash  :

Jumper Settings (HW-Configuration DIP-Switch):



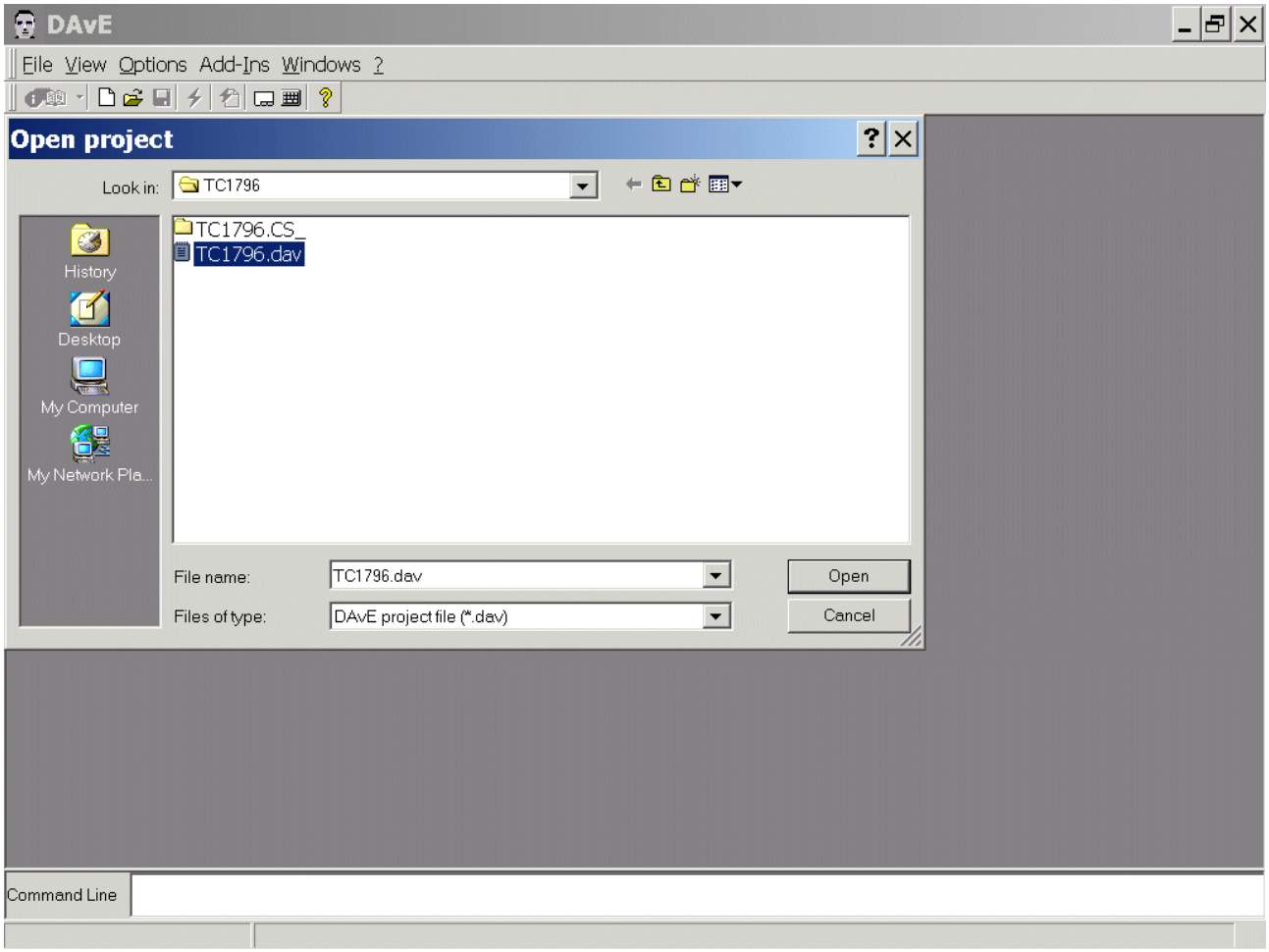
Bottom side: all little DIPs: OFF



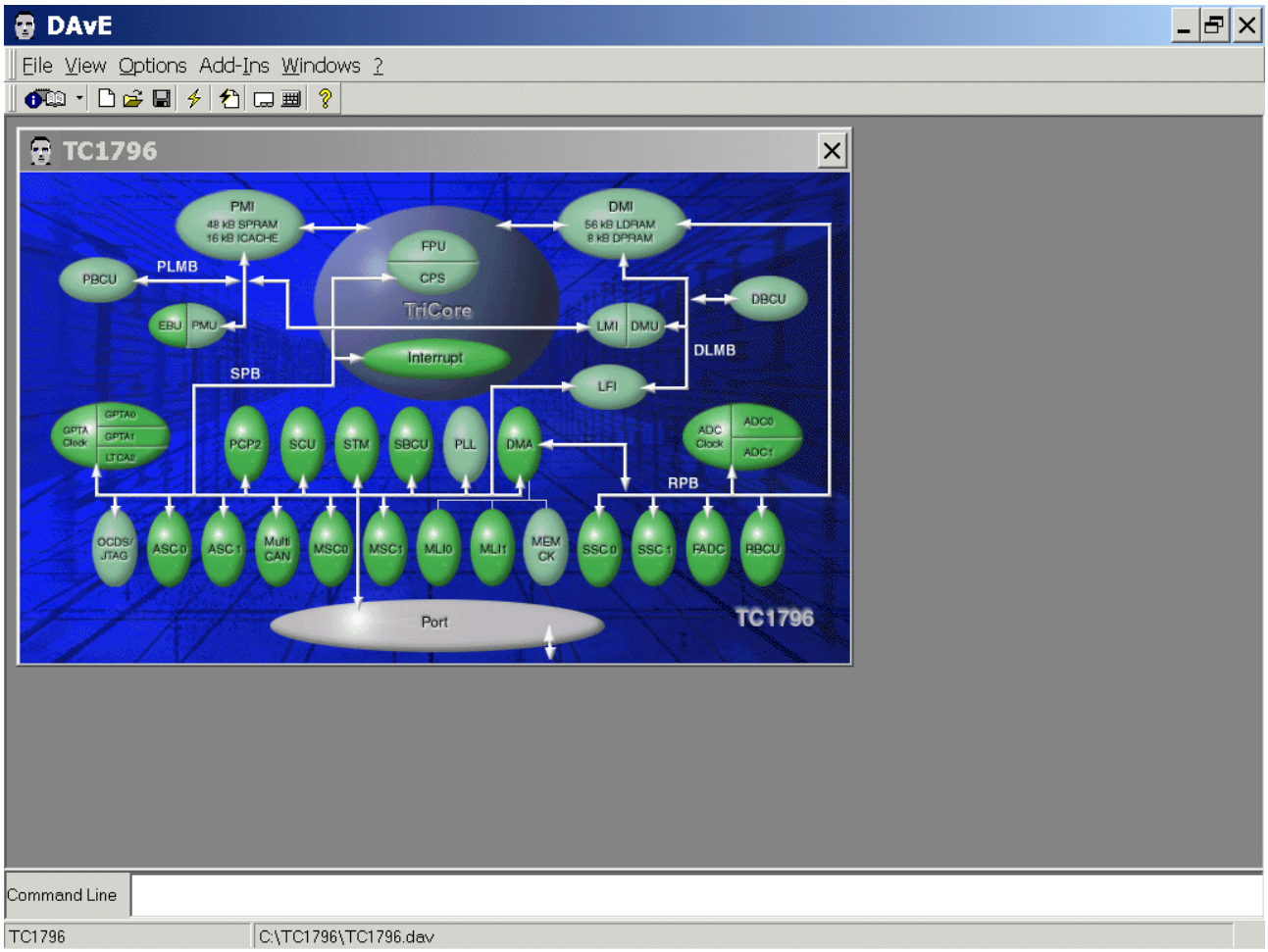
Note:
The little DIPs on the bottom side are for internal purpose.
Please, guarantee that they are all set to “OFF”.

Start DAvE and open the TC1796 project:

File
Open
Look in: select C:\TC1796
File name: select TC1796.dav

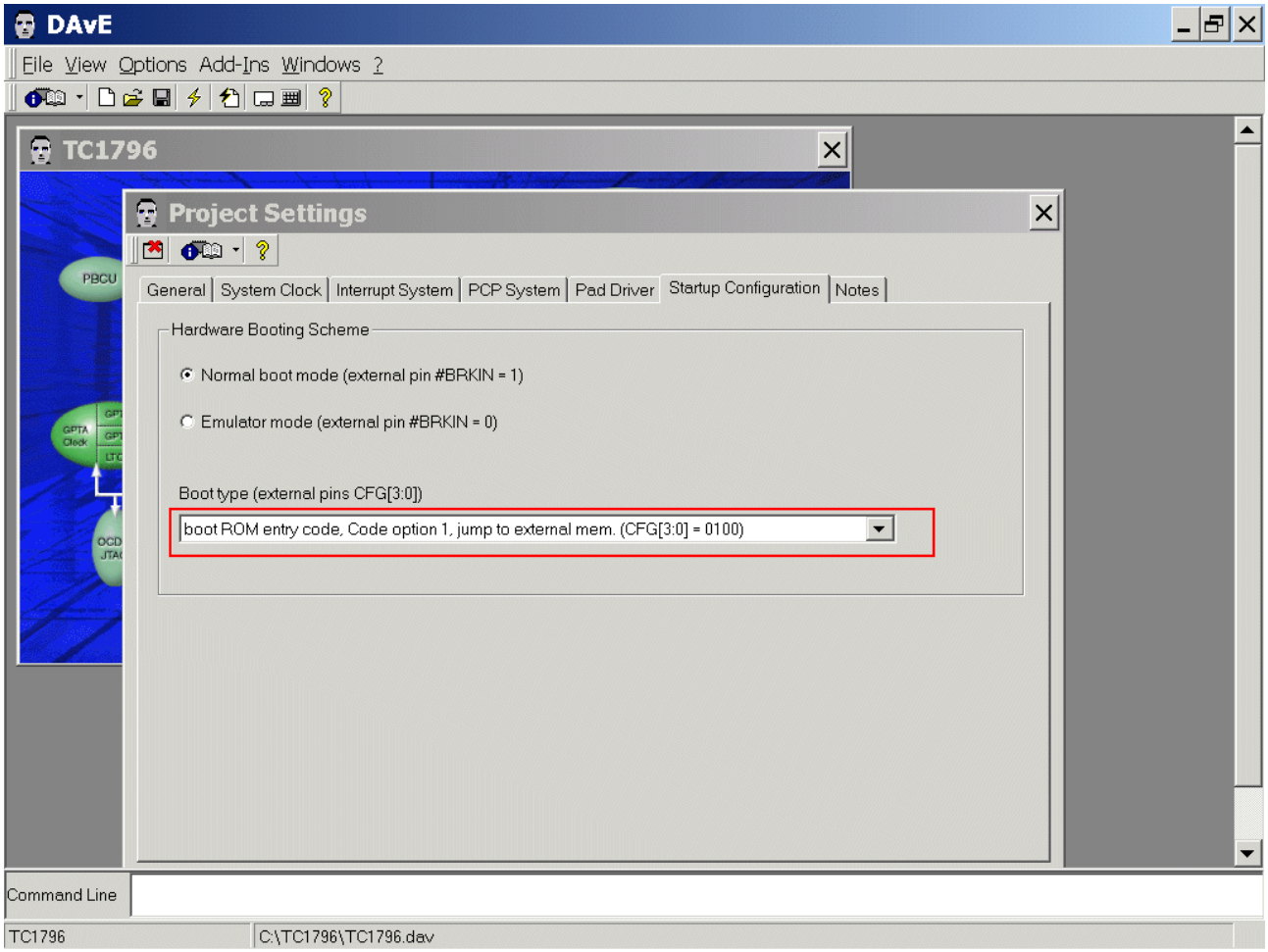


Open

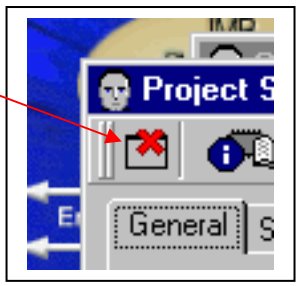

DAvE
File View Options Add-Ins Windows ?
TC1796
PMI
48 kB SPRAM
16 kB ICACHE
DMI
56 kB LDRAM
8 kB DPRAM
FPU
CPS
TriCore
PBCU
PLMB
EBU
PMU
LMI
DMU
DBCU
DLMB
Interrupt
SPB
LFI
GPTA Clock
GPTA0
GPTA1
LTCA2
PCP2
SCU
STM
SBCU
PLL
DMA
ADC Clock
ADC0
ADC1
RPB
OCDS/ JTAG
ASC0
ASC1
Multi CAN
MSC0
MSC1
MLI0
MLI1
MEM CK
SSC0
SSC1
FADC
RBCU
Port
TC1796
Command Line
TC1796
C:\TC1796\TC1796.dav

File – Project Settings

Startup Configuration: Boot type (CFG[3:0]) select 0100



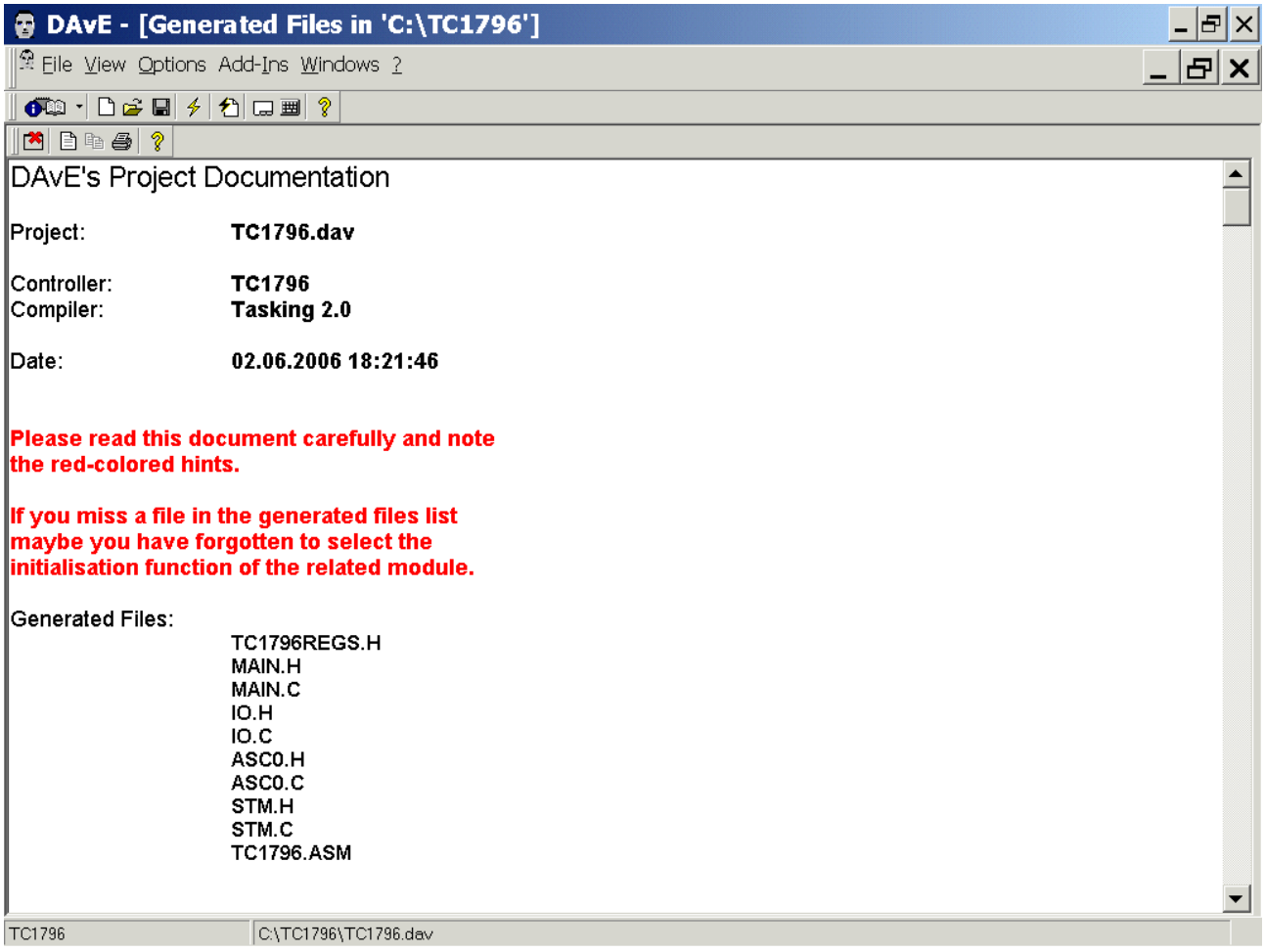
Exit this dialog now by clicking  the close button:



Generate Code:

File Generate Code	or click 

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



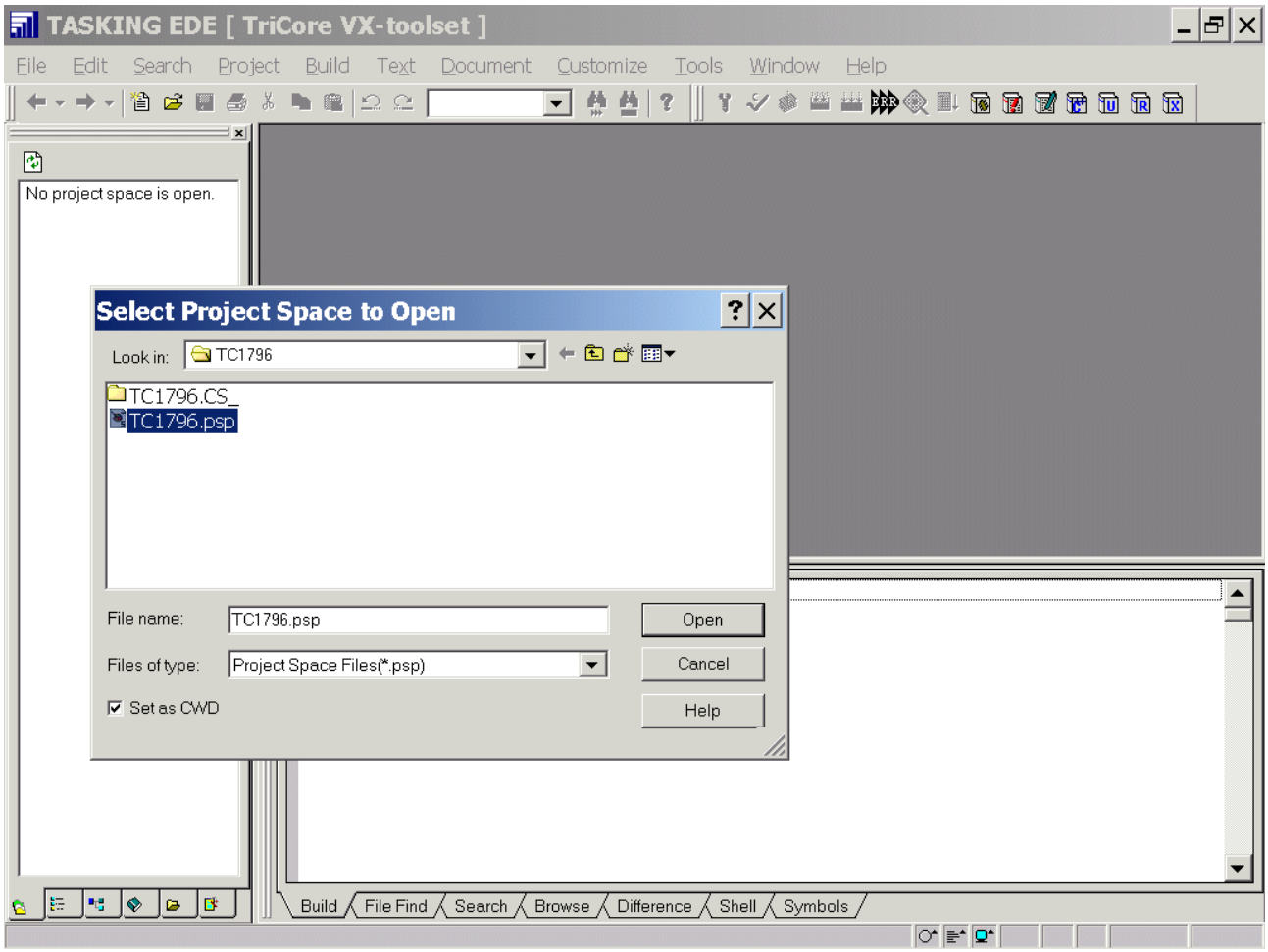
File
Exit
Save changes?

click Yes

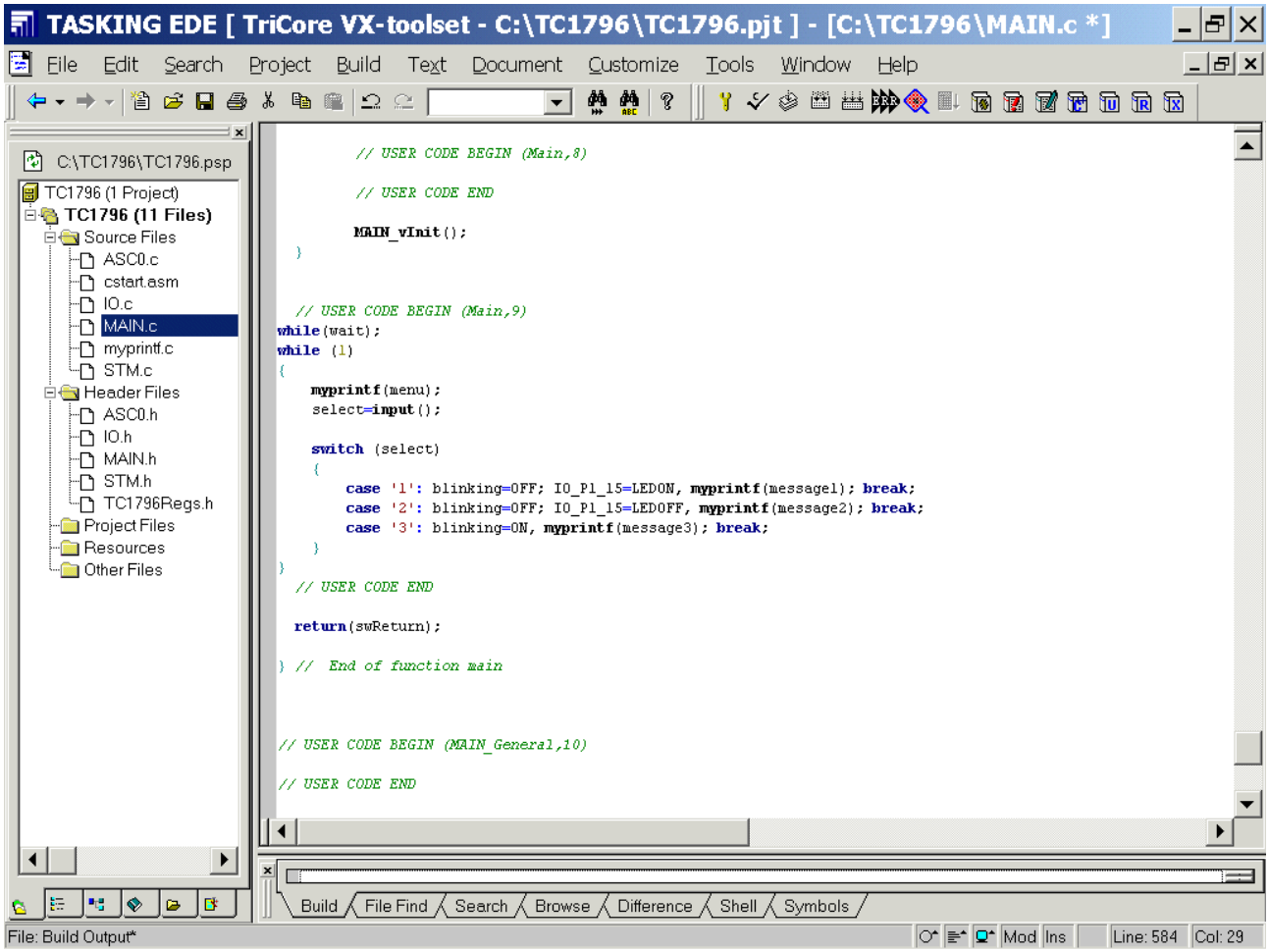
Start Tasking EDE and open the project:

File – Open Project Space

Look in: select C:\TC1796
File name: select TC1796.psp



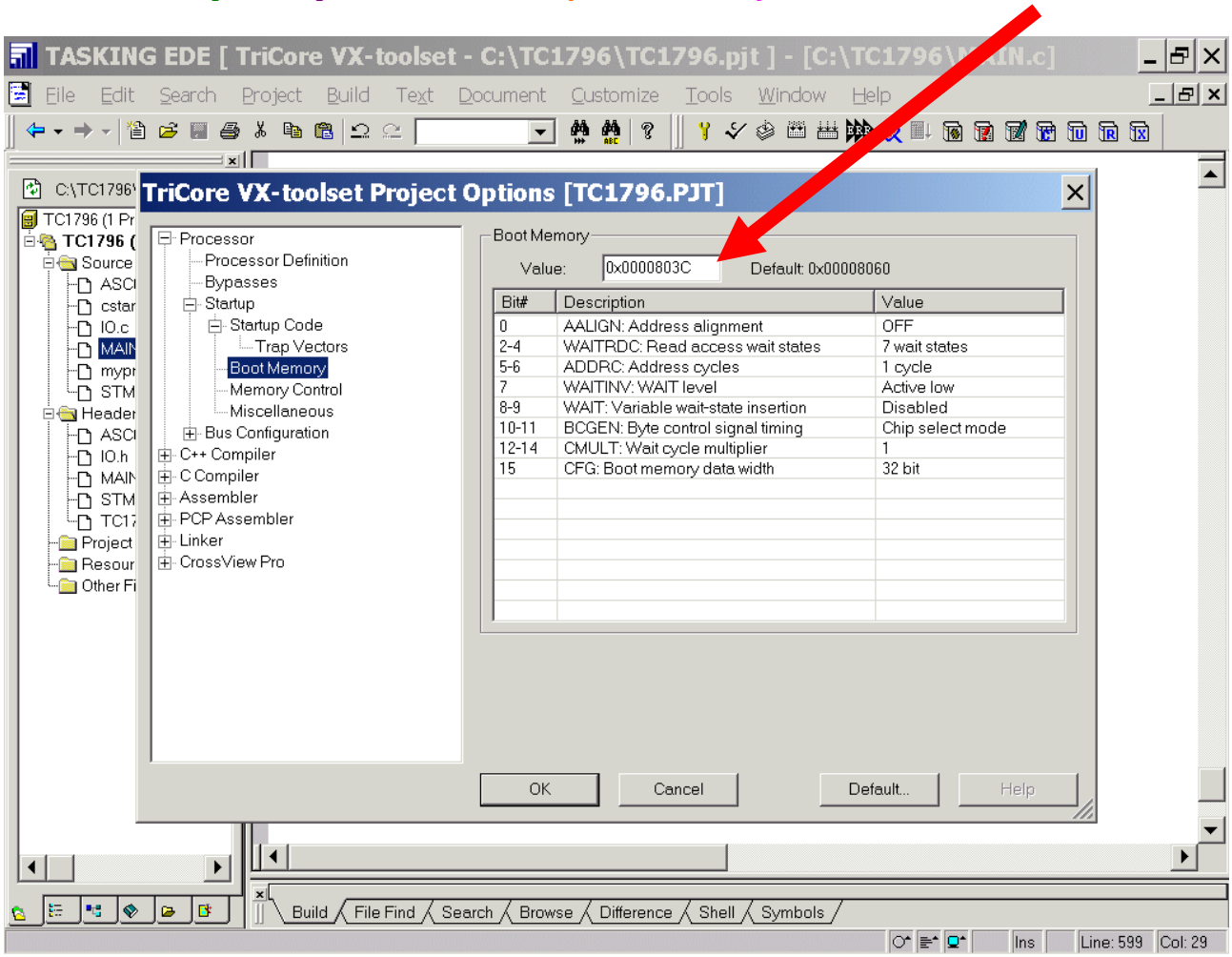
Open


TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1796\TC1796.psp
TC1796 (1 Project)
TC1796 (11 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
TC1796Regs.h
Project Files
Resources
Other Files
// USER CODE BEGIN (Main,8)
// USER CODE END
MAIN_vInit();
// USER CODE BEGIN (Main,9)
while(wait);
while (1)
{
myprintf(menu);
select=input();
switch (select)
{
case '1': blinking=OFF; IO_P1_15=LED0N, myprintf(message1); break;
case '2': blinking=OFF; IO_P1_15=LED0FF, myprintf(message2); break;
case '3': blinking=ON, myprintf(message3); break;
}
}
// USER CODE END
return(swReturn);
} // End of function main
// USER CODE BEGIN (MAIN_General,10)
// USER CODE END
Build File Find Search Browse Difference Shell Symbols
File: Build Output*
Mod Ins
Line: 584 Col: 29

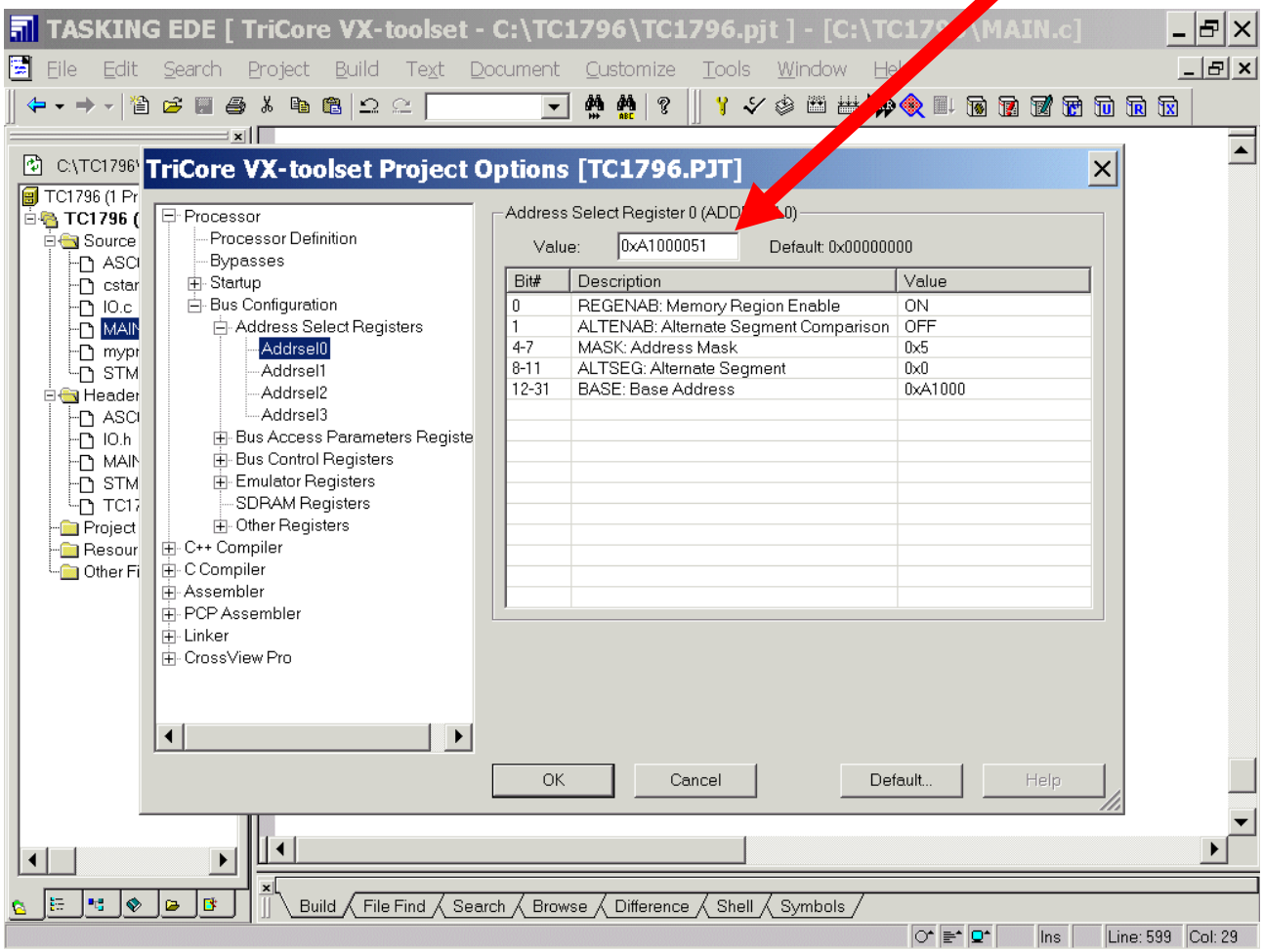
Configure Compiler, Assembler, Linker, Locater and Build – Control:

Project – Project Options

Processor: Startup: Startup Code: BootMemory: Boot Memory: Value: insert 0x0000803C



Processor: Bus Configuration: Address Select Registers: Addrsel0: insert 0xA1000051



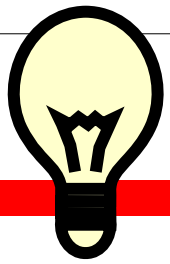


Table 13-10 EBU Address Regions, Registers and Chip Selects

Region	Associated Chip Select	Address Select Registers	Bus Configuration Registers	Bus Access Parameters Registers
Region 0	$\overline{CS0}$	EBU_ADDRSEL0	EBU_BUSCON0	EBU_BUSAP0
Region 1	$\overline{CS1}$	EBU_ADDRSEL1	EBU_BUSCON1	EBU_BUSAP1
Region 2	$\overline{CS2}$	EBU_ADDRSEL2	EBU_BUSCON2	EBU_BUSAP2
Region 3	$\overline{CS3}$	EBU_ADDRSEL3	EBU_BUSCON3	EBU_BUSAP3
Emulator region	$\overline{CSEMU}$	EBU_EMUAS	EBU_EMUBC	EBU_EMUBAP

EBU_ADDRSELx (x = 0-3)
EBU Address Select Register **ADDRSEL[3:1] - Reset Value: 0000 0000_H**
ADDRSEL0 - Reset Value (internal boot): $0000\ 0000_H$
ADDRSEL0 - Reset Value (external boot): $A000\ 0001_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASE															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE				ALTSEG				MASK				0		ALT EN AB	REG EN AB
rw				rw				rw				r		rw	rw

Note:
Memory Region Address Mask: Specifies the number of rightmost bits in the base address starting at bit 26, which should be included in the address comparison. Bits (31:27) will always be part of the comparison.

4 MBytes OnBoardFlash = 10000000000000000000 b
Base = A1000
Mask = 5

	1	2	3	4	5		= Mask
[31:27]	[26]	[25]	[24]	[23]	[22]	[21:0]	
					1	00000000000000000000	= 4 MBytes OnBoardFlash

Processor: Bus Configuration: Address Select Registers: Addrsel1: insert 0xA2000071

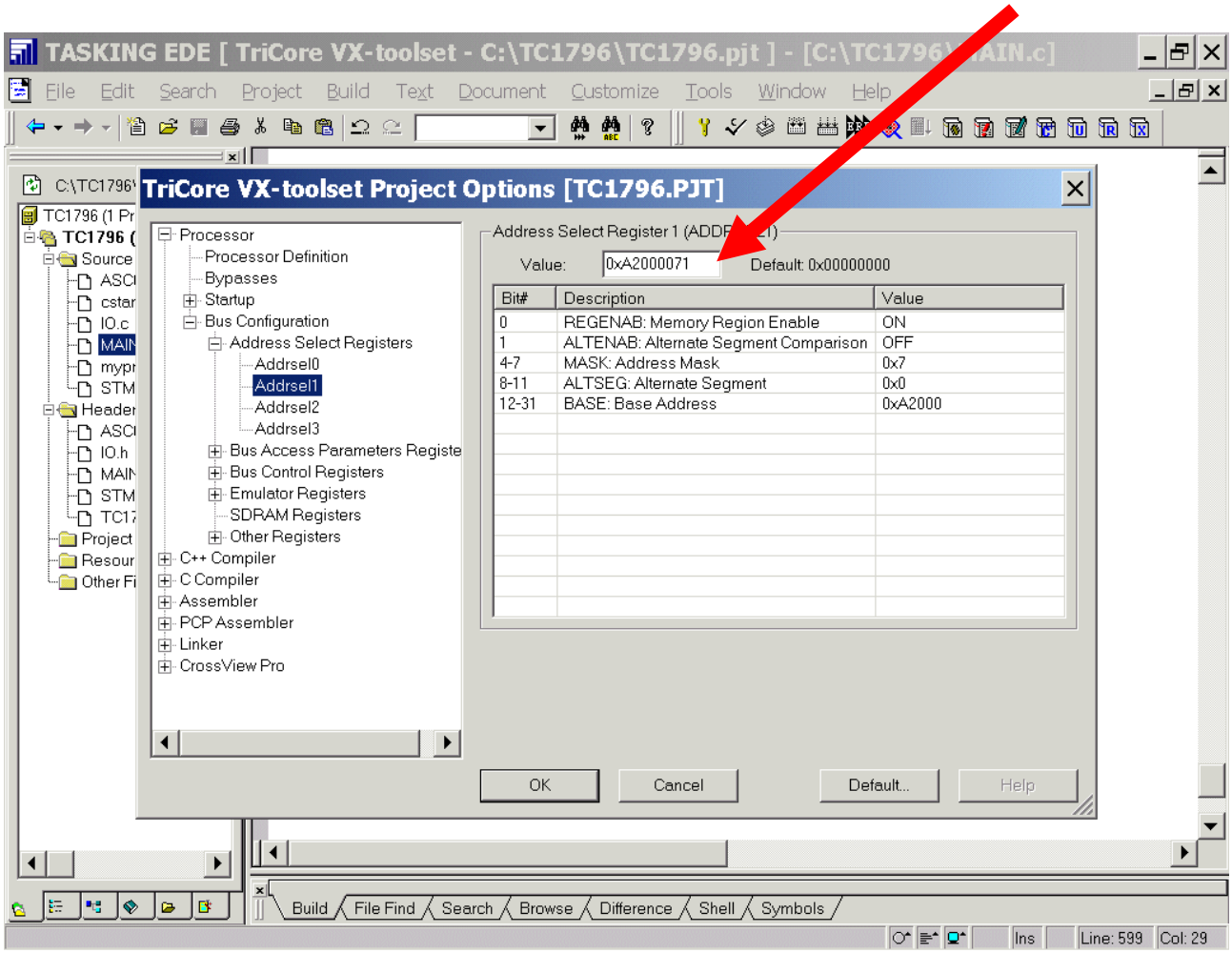


Table 13-10 EBU Address Regions, Registers and Chip Selects

Region	Associated Chip Select	Address Select Registers	Bus Configuration Registers	Bus Access Parameters Registers
Region 0	$\overline{CS0}$	EBU_ADDRSEL0	EBU_BUSCON0	EBU_BUSAP0
Region 1	$\overline{CS1}$	EBU_ADDRSEL1	EBU_BUSCON1	EBU_BUSAP1
Region 2	$\overline{CS2}$	EBU_ADDRSEL2	EBU_BUSCON2	EBU_BUSAP2
Region 3	$\overline{CS3}$	EBU_ADDRSEL3	EBU_BUSCON3	EBU_BUSAP3
Emulator region	$\overline{CSEMU}$	EBU_EMUAS	EBU_EMUBC	EBU_EMUBAP

EBU_ADDRSELx (x = 0-3)
EBU Address Select Register **ADDRSEL[3:1] - Reset Value: 0000 0000$_H$**
ADDRSEL0 - Reset Value (internal boot): 0000 0000$_H$
ADDRSEL0 - Reset Value (external boot): A000 0001$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASE															
rw															

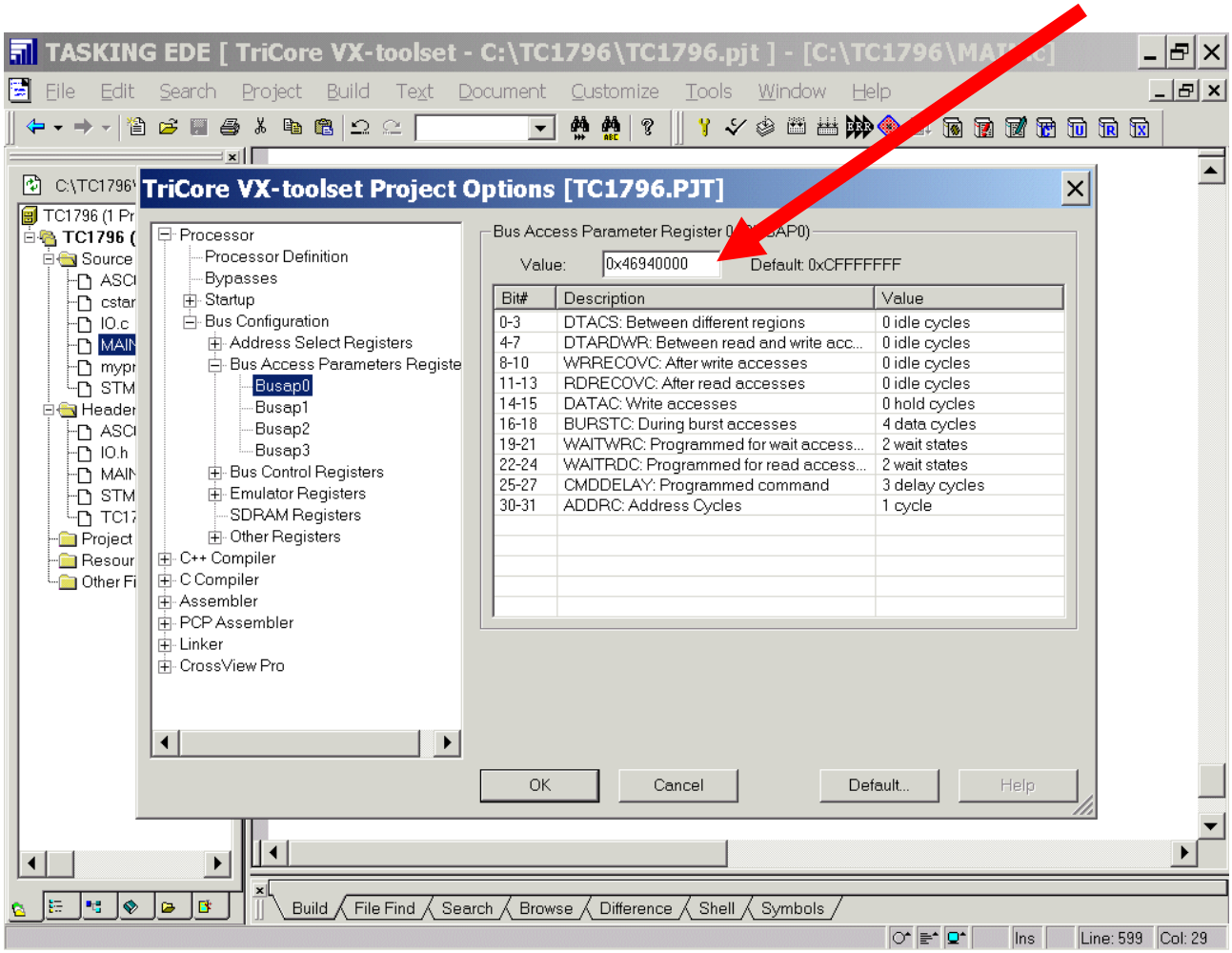
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE				ALTSEG				MASK				0		ALT EN AB	REG EN AB
rw				rw				rw				r		rw	rw

Note:
Memory Region Address Mask: Specifies the number of rightmost bits in the base address starting at bit 26, which should be included in the address comparison. Bits (31:27) will always be part of the comparison.

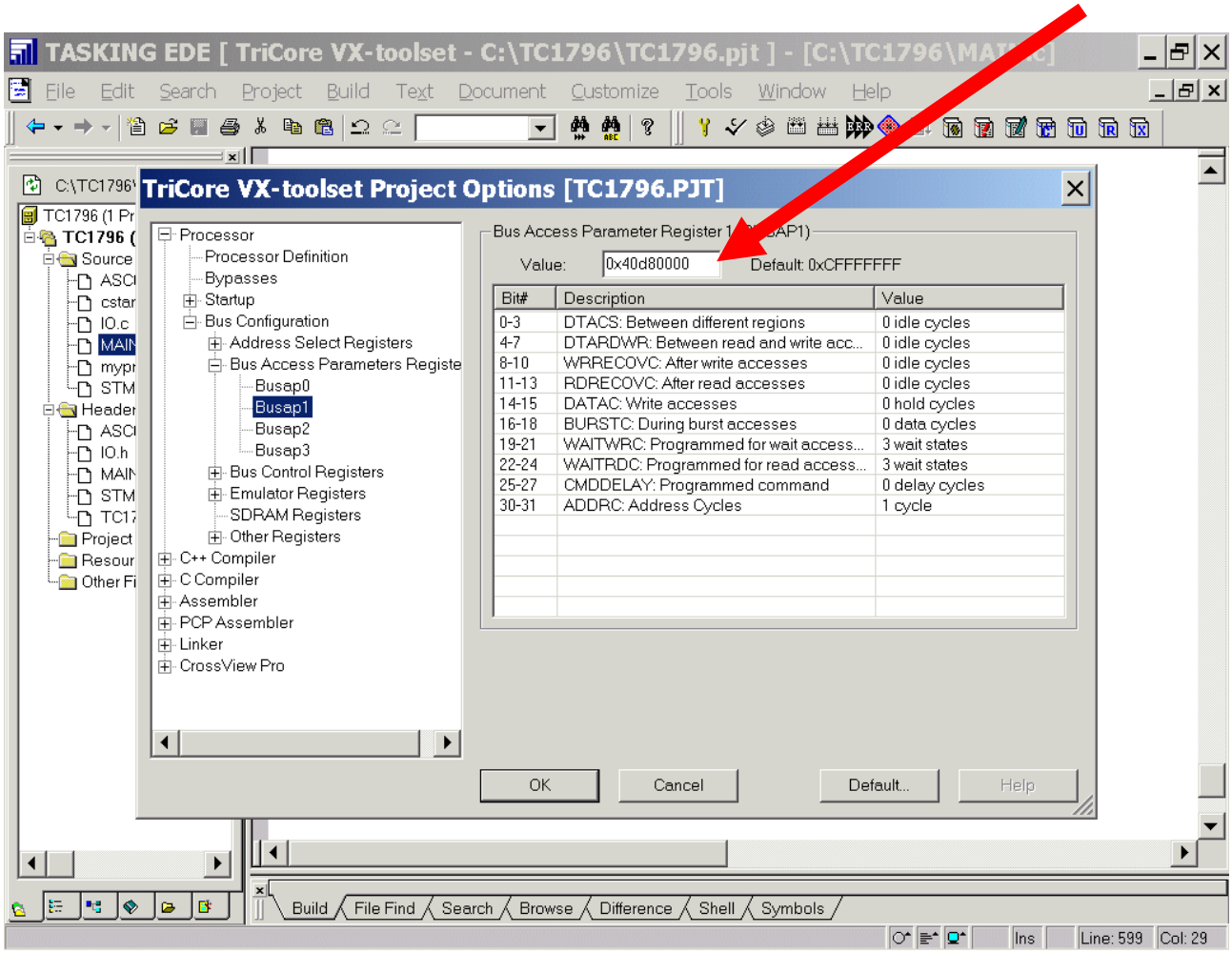
1 MBytes OnBoardFlash = 10000000000000000000 b
Base = A2000
Mask = 7

	1	2	3	4	5	6	7		= Mask
[31:27]	[26]	[25]	[24]	[23]	[22]	[21]	[20]	[19:0]	
							1	00000000000000000000	= 1 MBytes OnBoardFlash

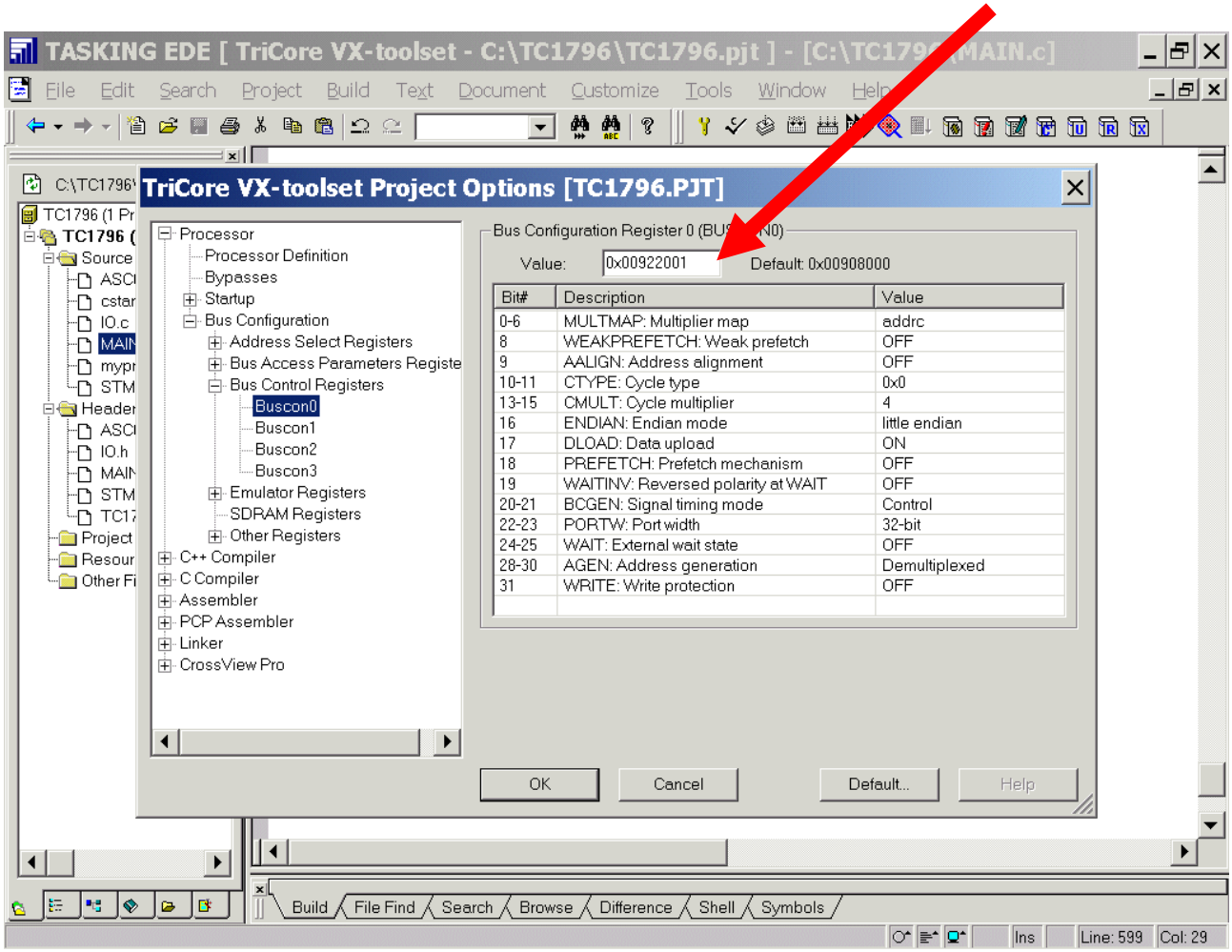
Processor: Bus Configuration: Bus Access Parameters Registers: Busap0: insert 0x46940000



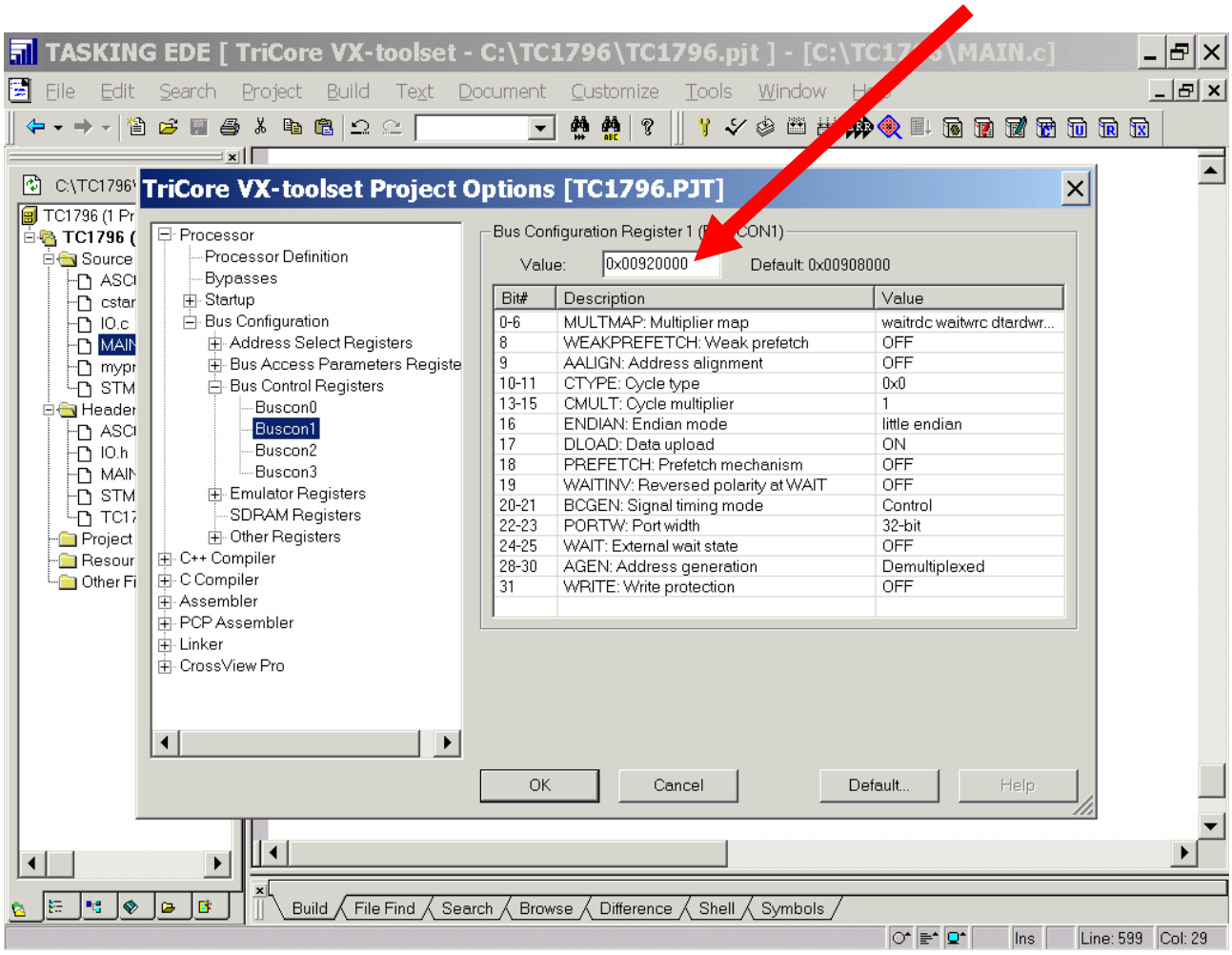
Processor: Bus Configuration: Bus Access Parameters Registers: Busap1: insert 0x40d80000



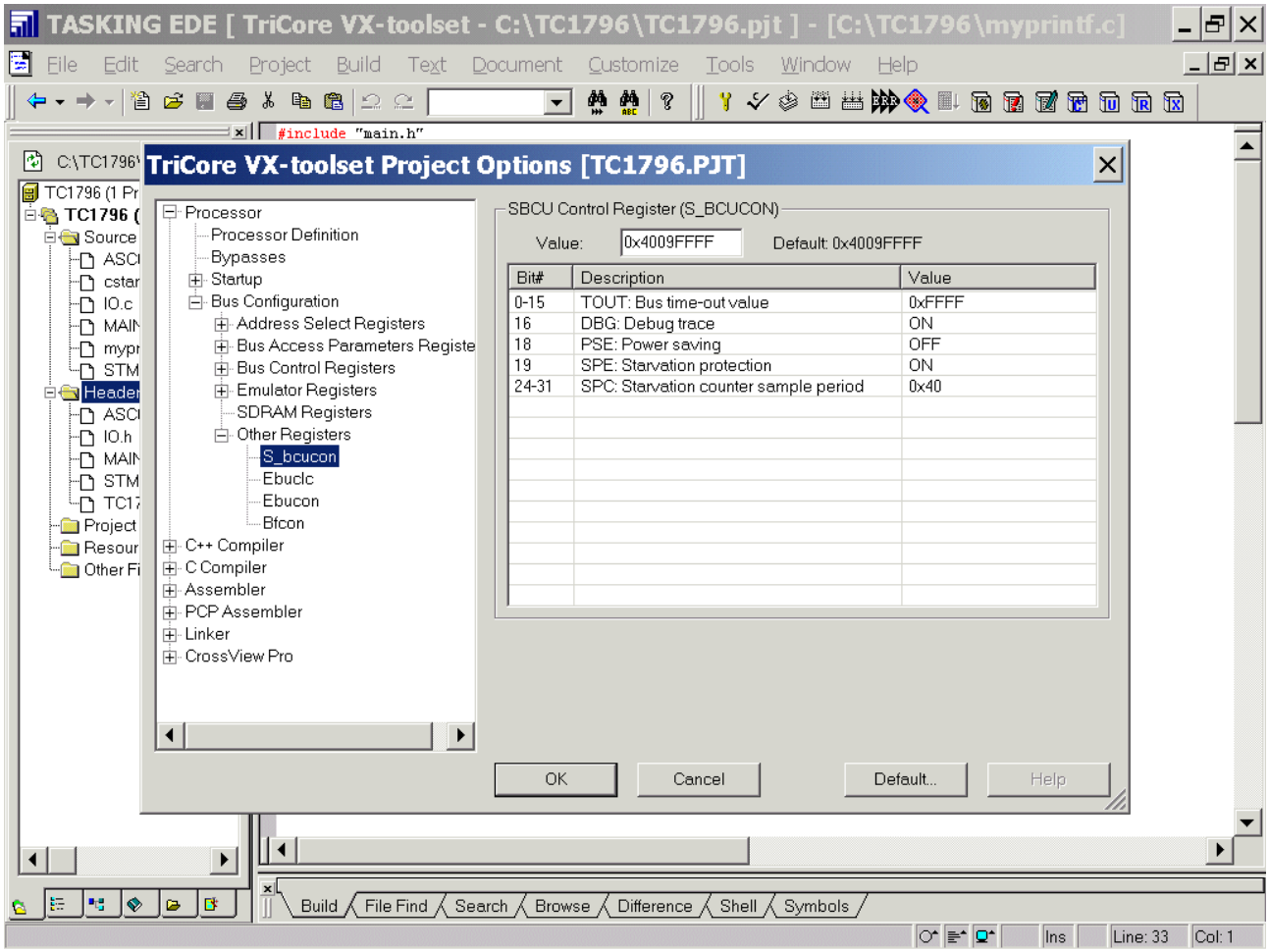
Processor: Bus Configuration: Bus Control Registers: Buscon0: insert 0x00922001



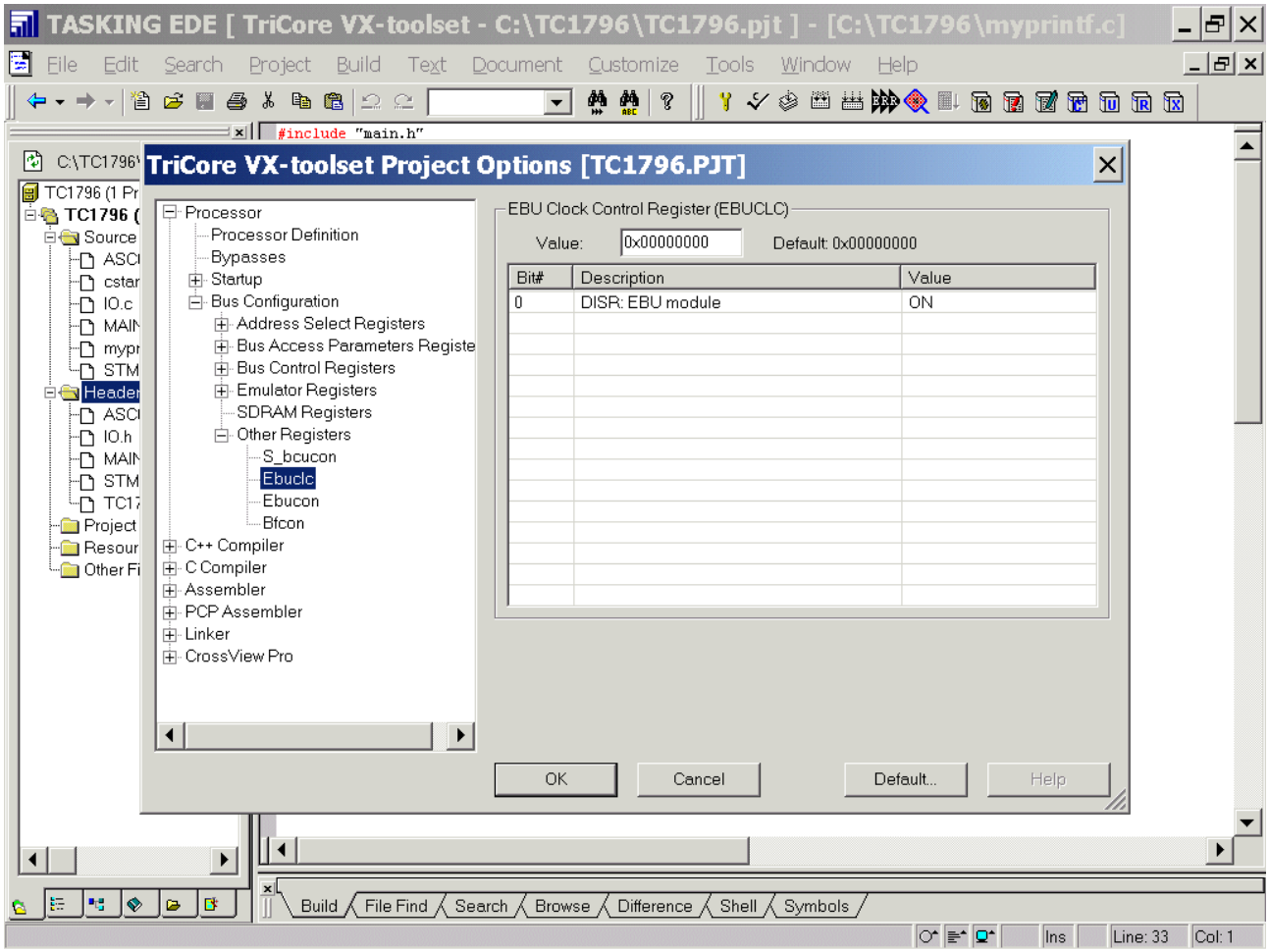
Processor: Bus Configuration: Bus Control Registers: Buscon1: insert 0x00920000



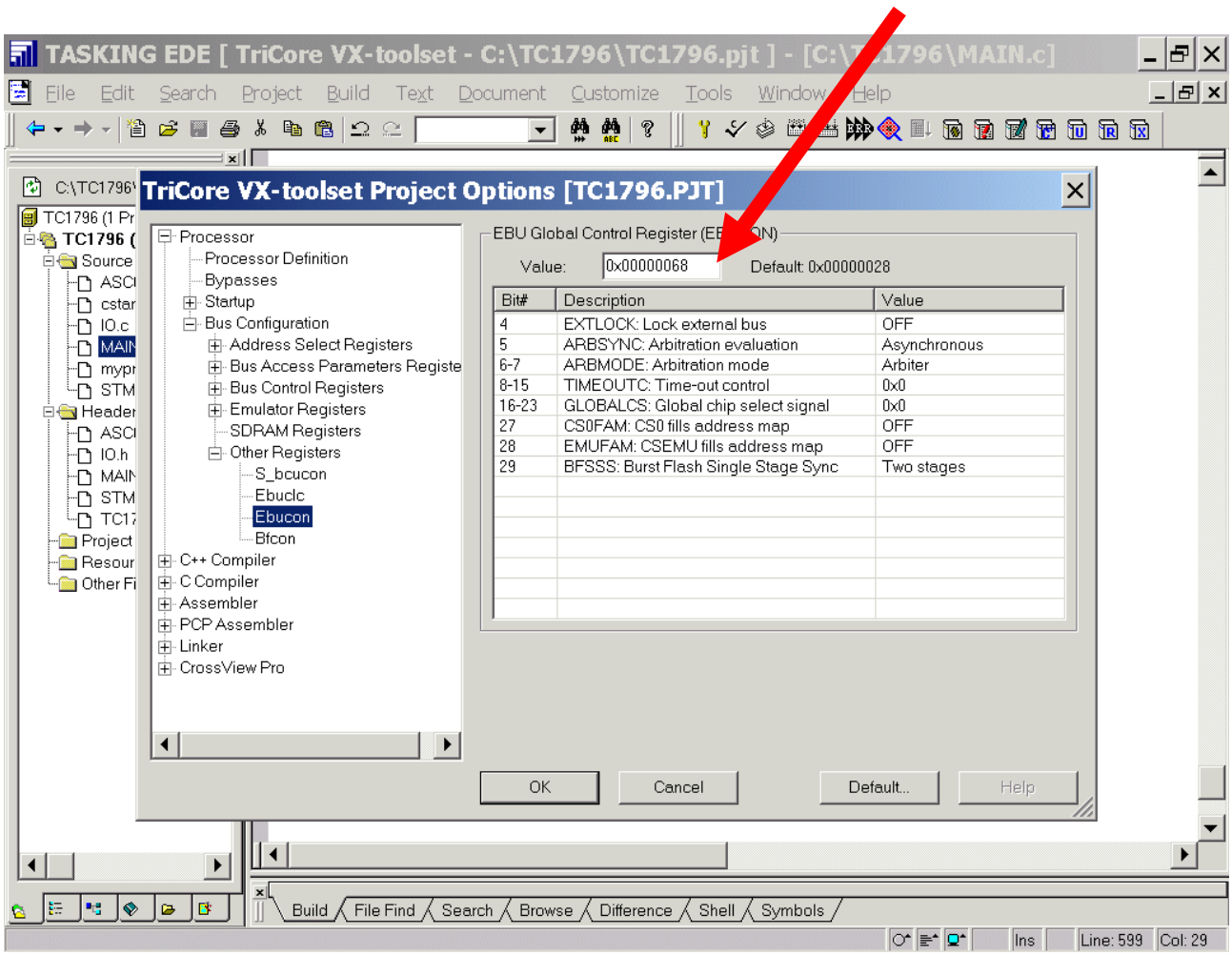
Processor: Bus Configuration: Other Registers: S_bcucon: (do nothing)



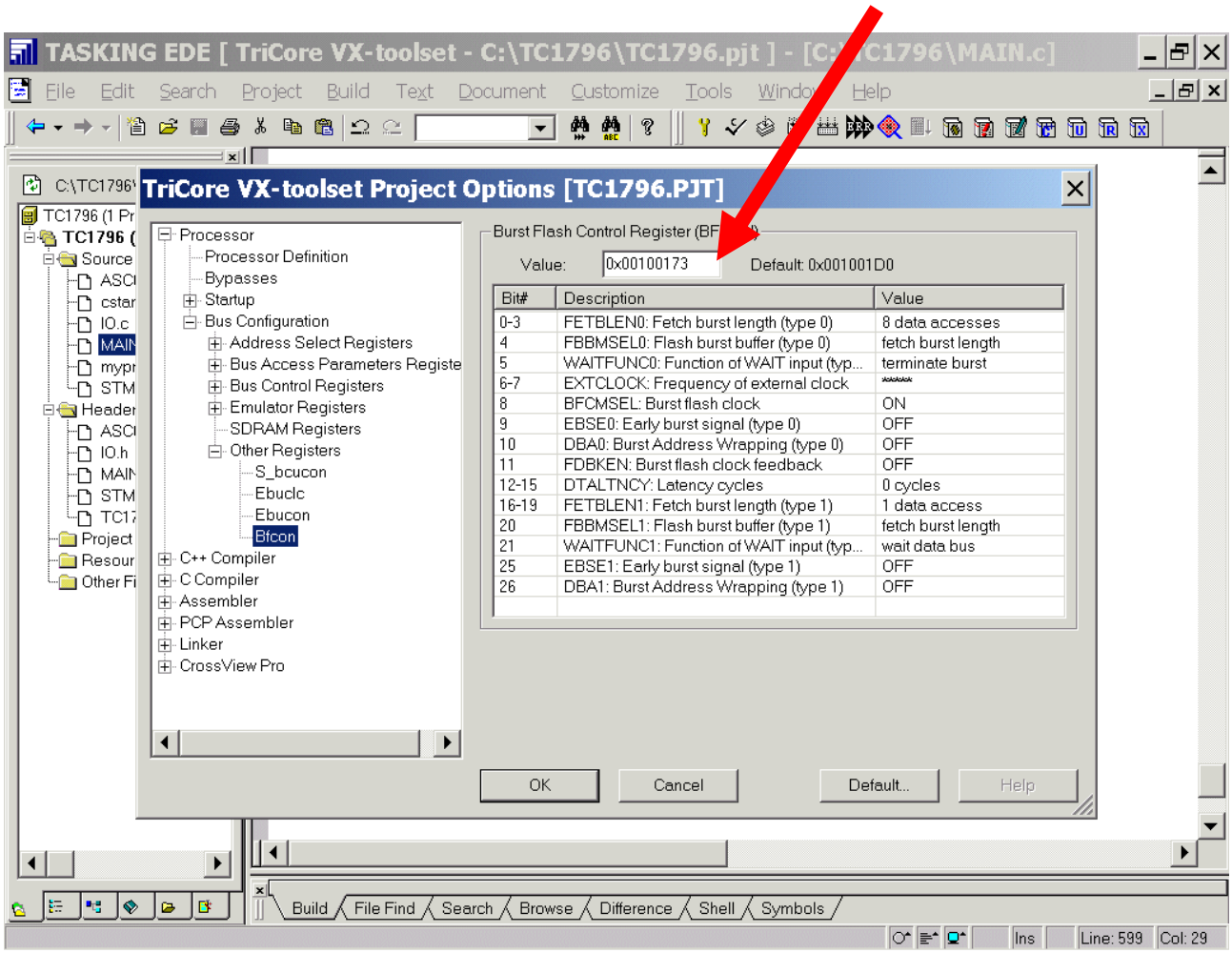
Processor: Bus Configuration: Other Registers: Ebuclc: (do nothing)



Processor: Bus Configuration: Other Registers: Ebucon: insert 0x00000068



Processor: Bus Configuration: Other Registers: Bfcon: insert 0x00100173



C Compiler: Allocation: deactivate ☐ Default __near allocation for objects below threshold

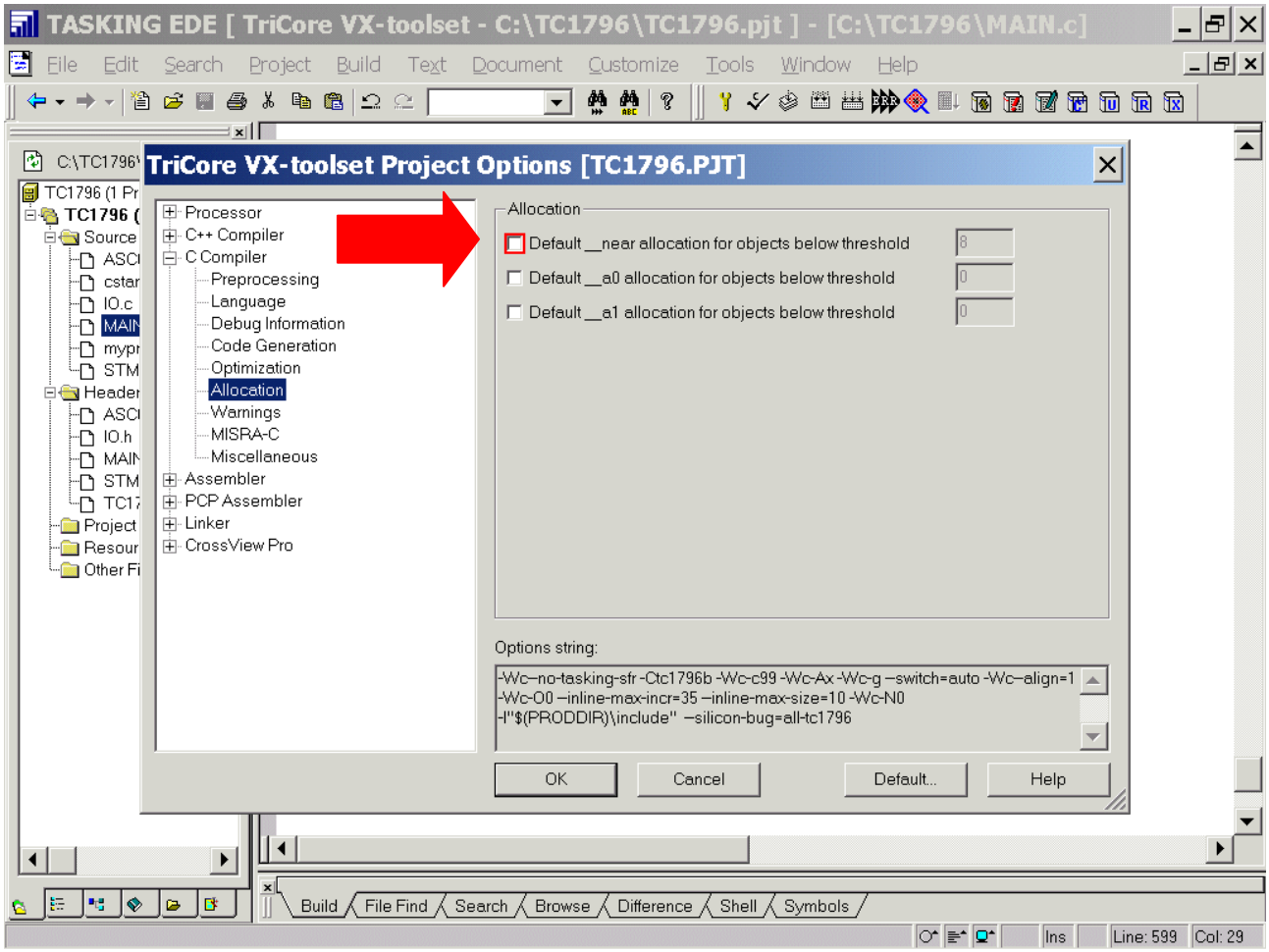
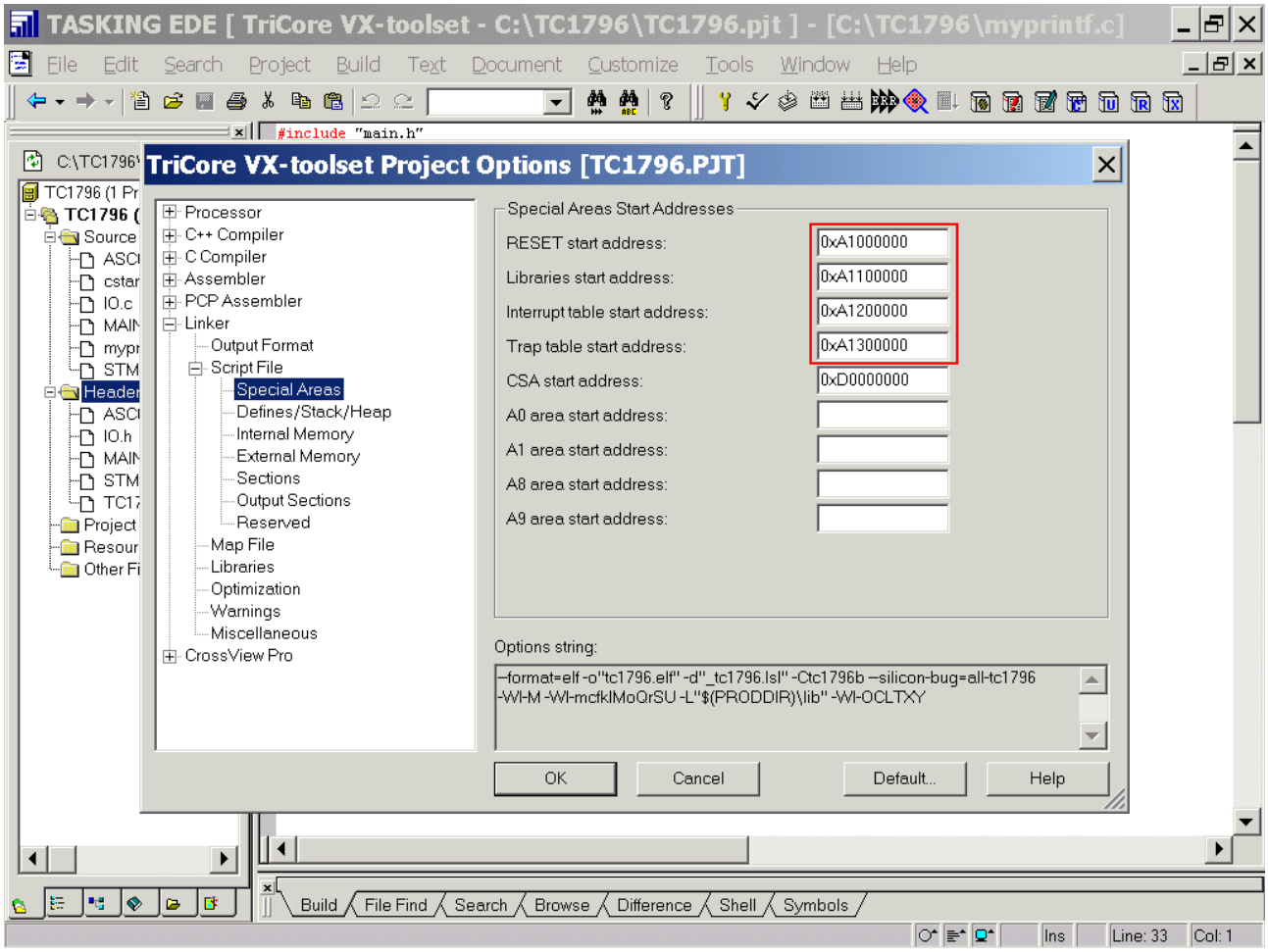


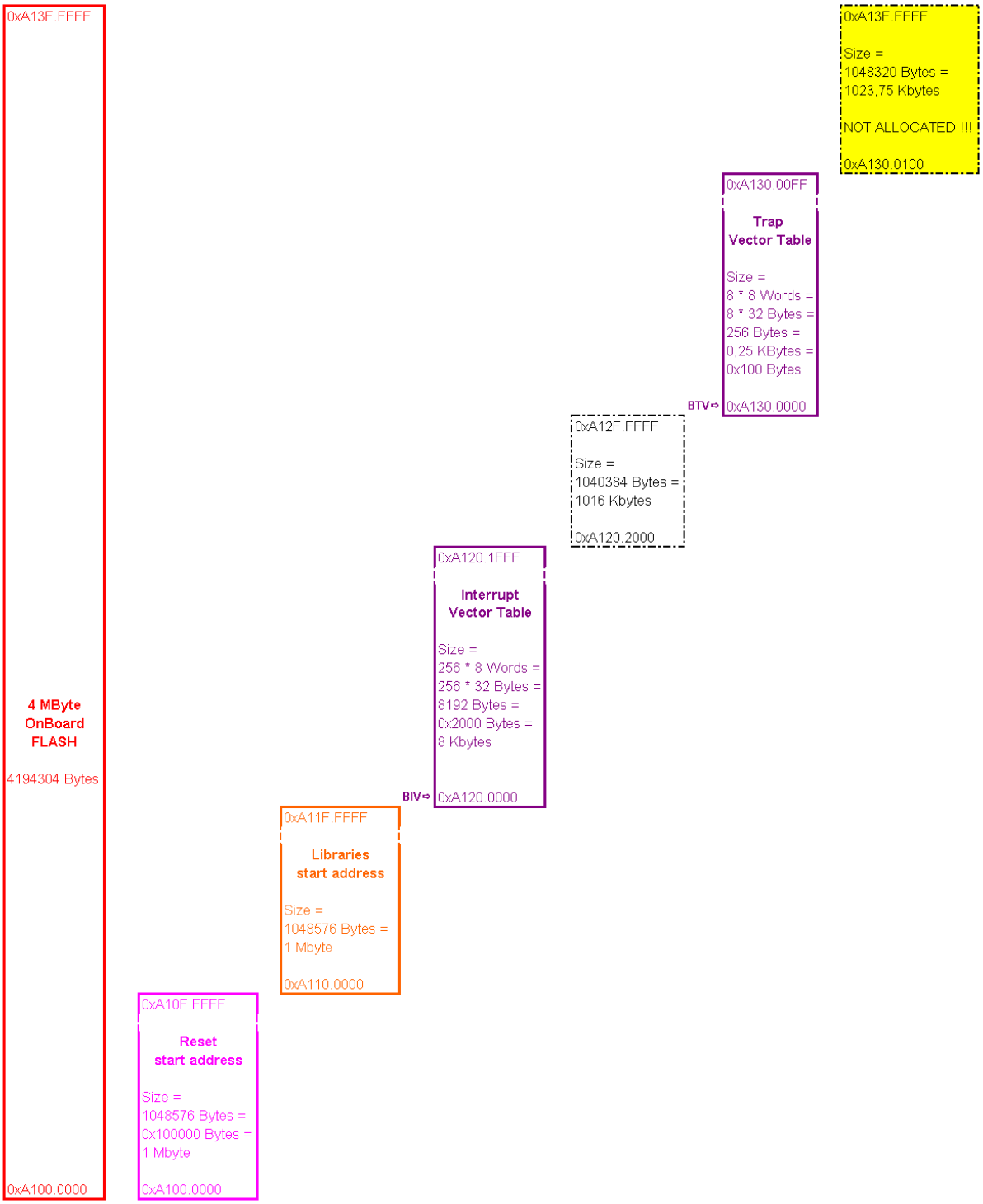
Table 9-2 SPB/RPB Address Map of Segment 0 to 14 (cont'd)

Seg-ment	Address Range	Size	Description	Access Type	
				Read	Write
10	$A000\ 0000_H$ - $A01F\ FFFF_H$	2 Mbyte	Program Flash (PFLASH)	access	access[1)]
	$A020\ 0000_H$ - $A07F\ FFFF_H$	6 Mbyte	Reserved	PLMBBE & DLMBBE & SPBBE	PLMBBE
	$A080\ 0000_H$ - $AFDF\ FFFF_H$	246 Mbyte	External EBU space	EBU access	EBU access

Linker: Script File: Special Areas: RESET start address: insert 0xA1000000 (OnBoardFLASH)
Linker: Script File: Special Areas: Libraries start address: insert 0xA1100000 (OnBoardFLASH)
Linker: Script File: Special Areas: Interrupt table start address: insert 0xA1200000 (OnBoardFLASH)
Linker: Script File: Special Areas: Trap table start address: insert 0xA1300000 (OnBoardFLASH)

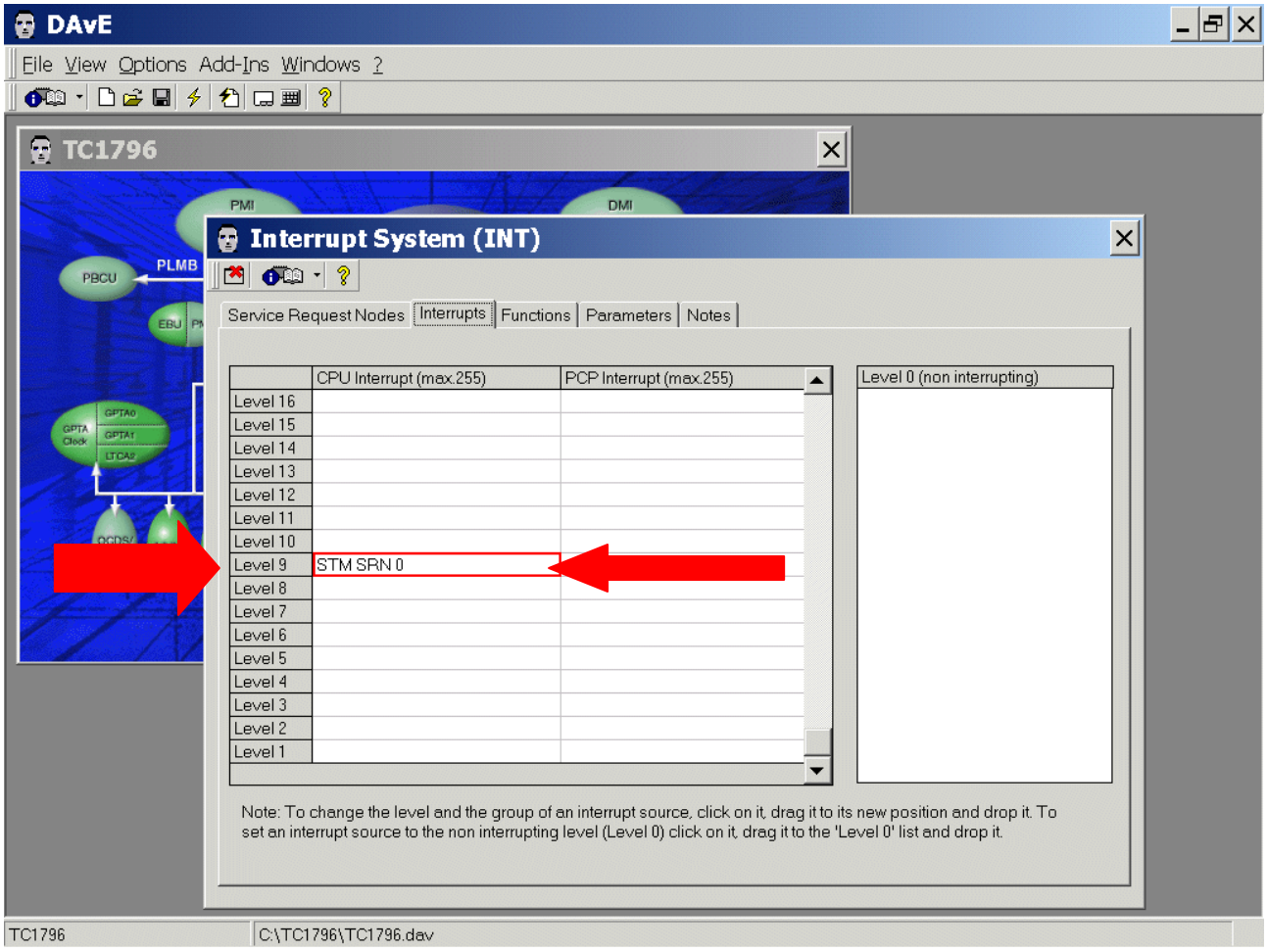

TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\myprintf.c]
File Edit Search Project Build Text Document Customize Tools Window Help
#include "main.h"
TriCore VX-toolset Project Options [TC1796.PJT]
Processor
C++ Compiler
C Compiler
Assembler
PCP Assembler
Linker
Output Format
Script File
Special Areas
Defines/Stack/Heap
Internal Memory
External Memory
Sections
Output Sections
Reserved
Map File
Libraries
Optimization
Warnings
Miscellaneous
CrossView Pro
Special Areas Start Addresses
RESET start address: 0xA1000000
Libraries start address: 0xA1100000
Interrupt table start address: 0xA1200000
Trap table start address: 0xA1300000
CSA start address: 0xD0000000
A0 area start address:
A1 area start address:
A8 area start address:
A9 area start address:
Options string:
--format=elf -o"tc1796.elf" -d"_tc1796.lsl" -Ctc1796b --silicon-bug=all-tc1796
-Wl-M -Wl-mcfklMoQrSU -L"$(PRODDIR)\lib" -Wl-OCLTXY
OK
Cancel
Default...
Help
Build
File Find
Search
Browse
Difference
Shell
Symbols
Ins
Line: 33
Col: 1

Memory-Map OnBoard-Flash:

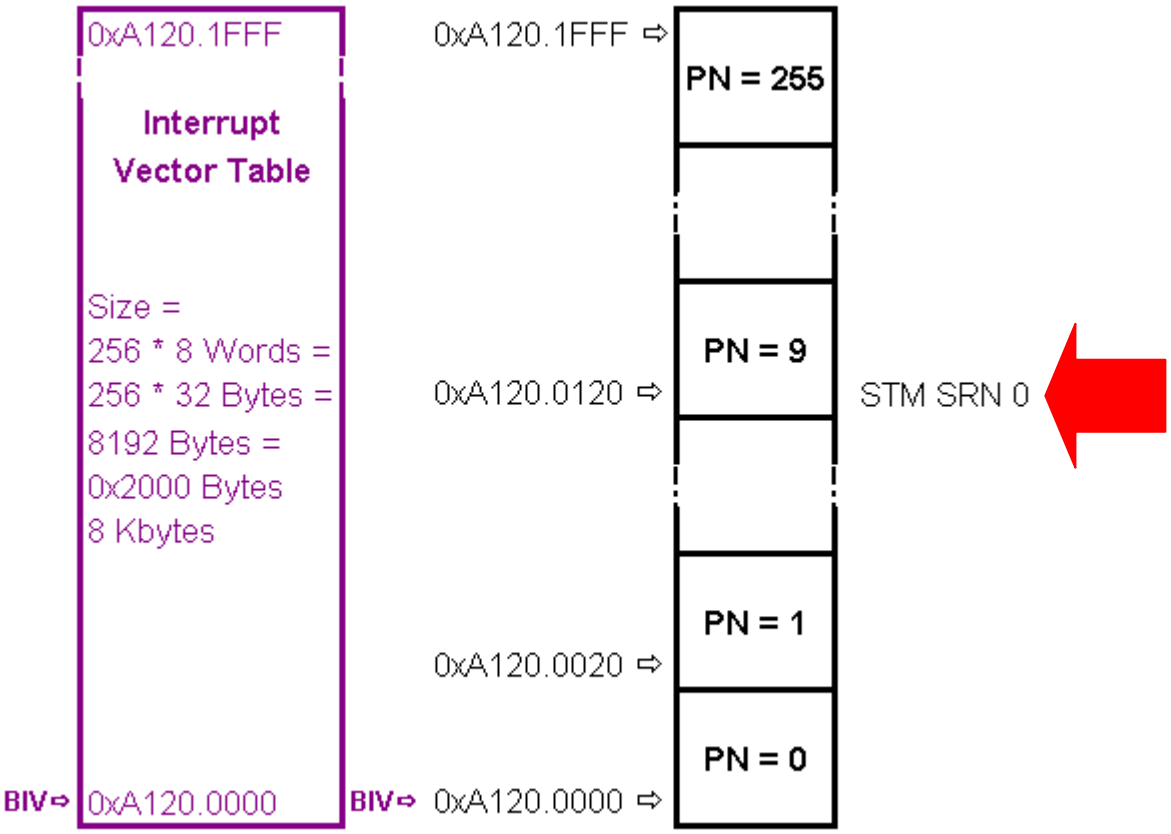


Interrupt–Vector–Table:

DAvE:



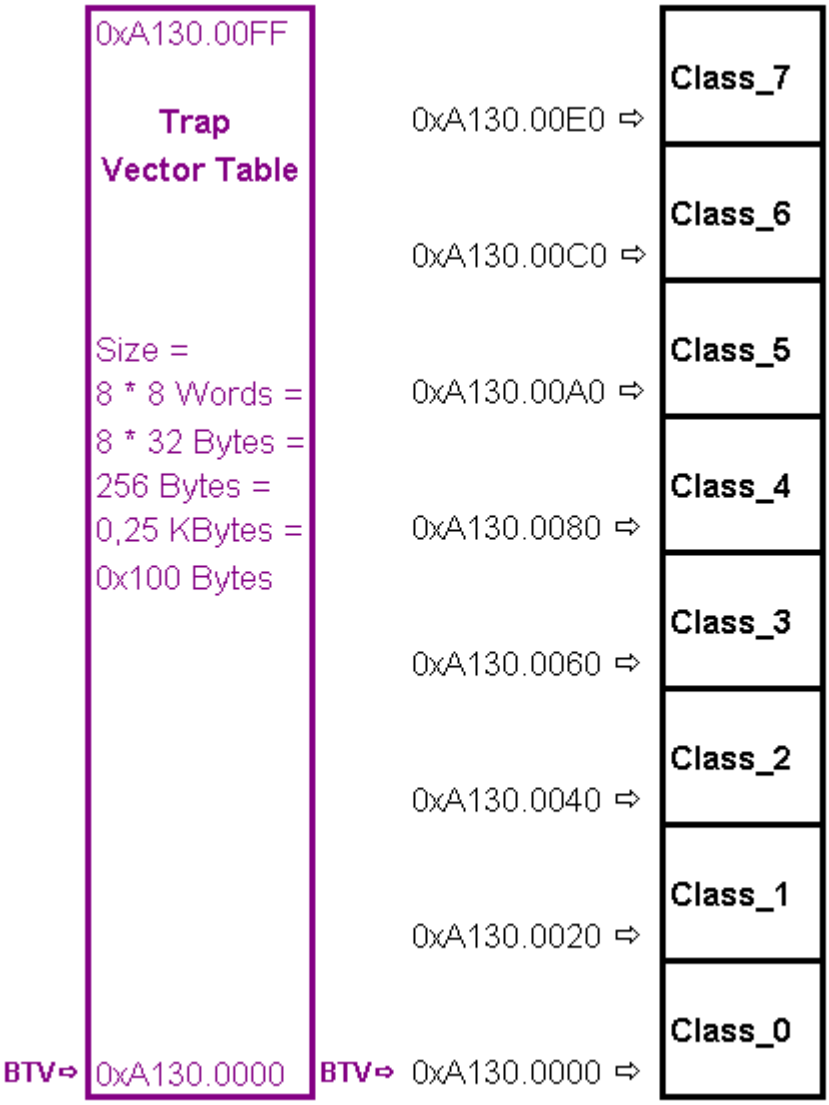
Interrupt–Vector–Table:



Note:
PN … Priority Number

Click here to see the Map File

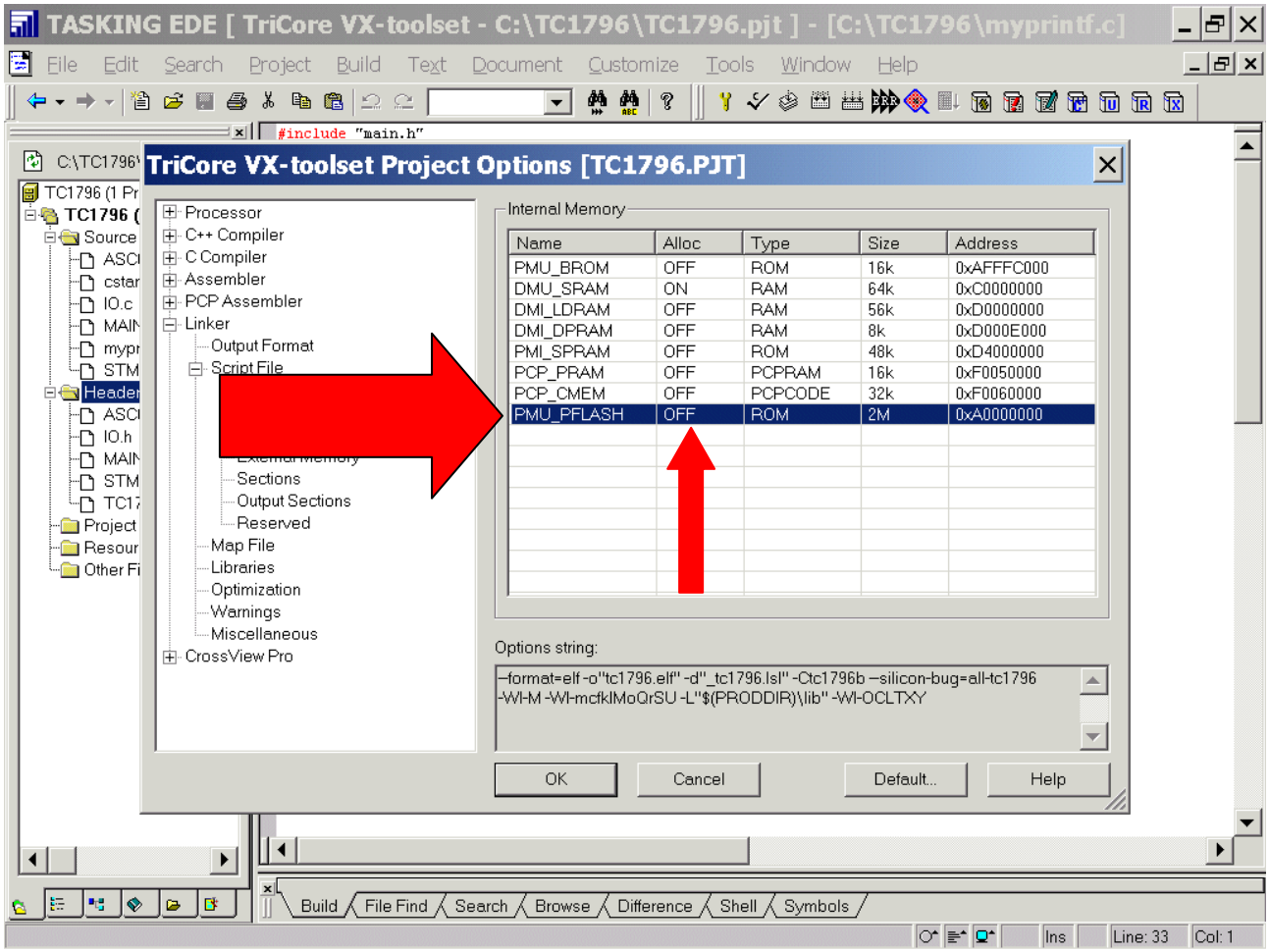
TRAP–Vector–Table:



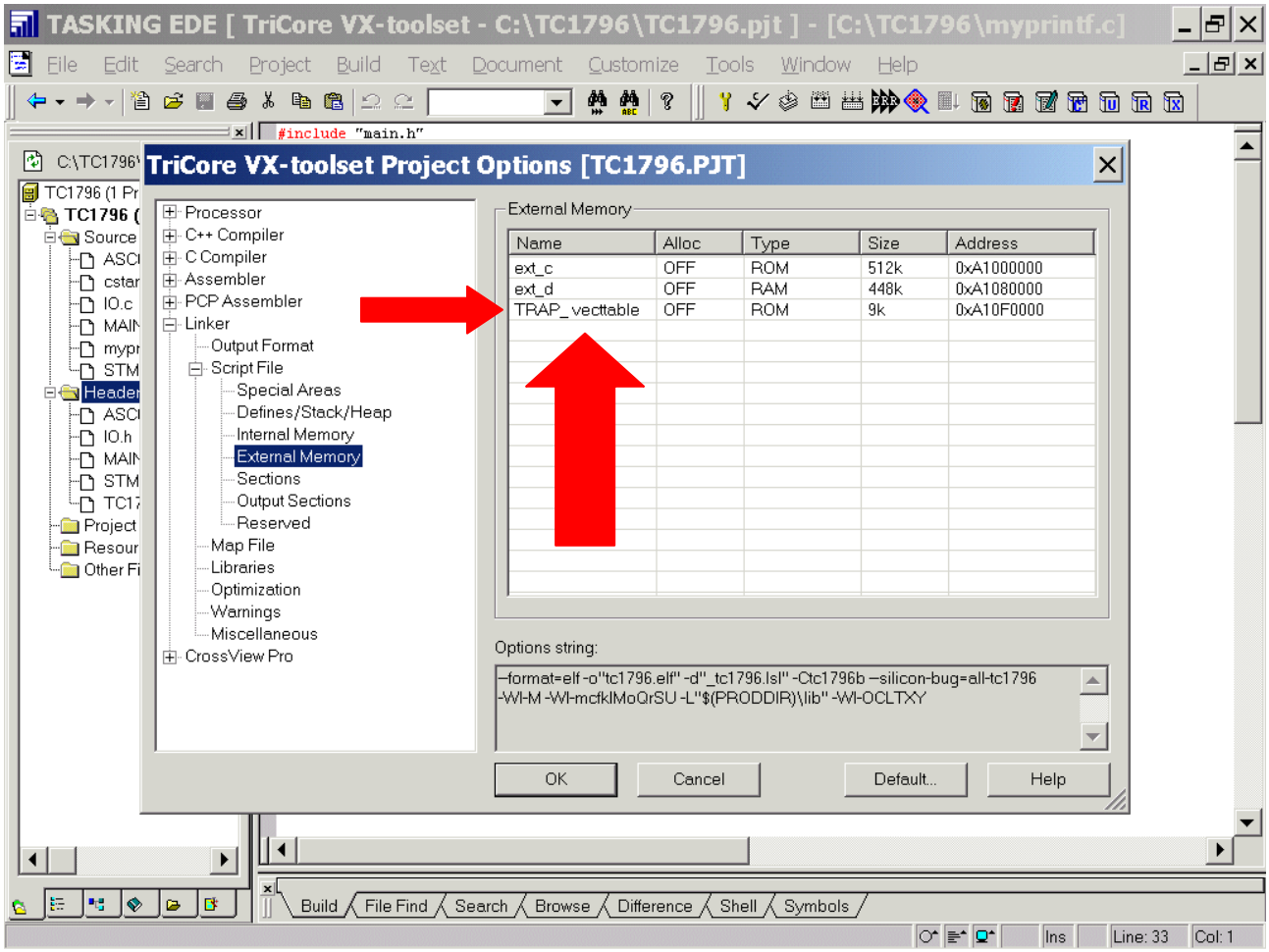
Note:
1 Word = 32 Bit
1 Word = 4 Bytes
8 Words= 32 Bytes

Click here to see the Map File

Linker: Script File: Internal Memory: PMU_PFLASH: Alloc: select OFF



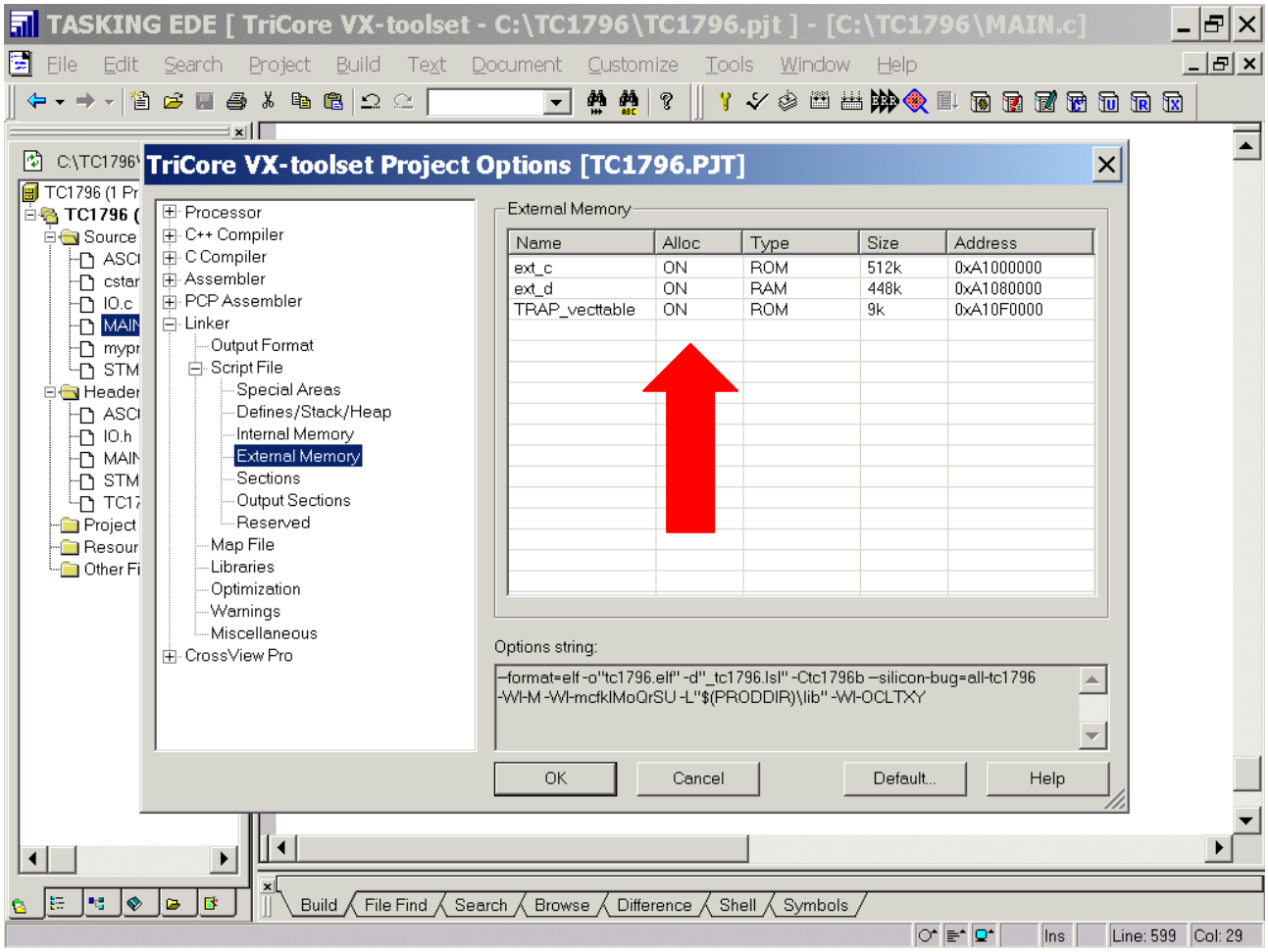
Linker: Script File: External Memory: Name: “vecttable” change to “TRAP_ vecttable”



Linker: Script File: External Memory: Name=ext_c: Alloc: select “ON”

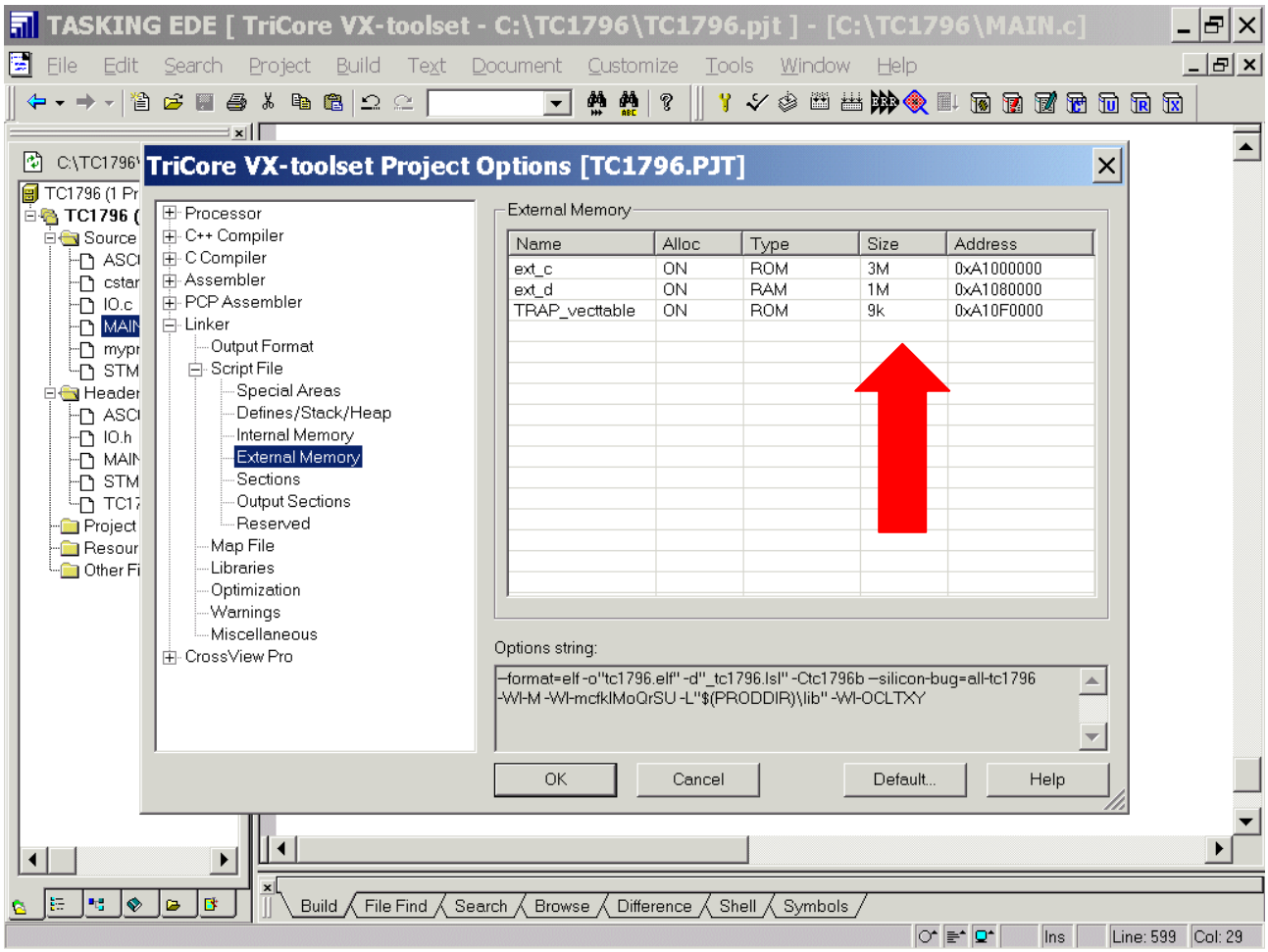
Linker: Script File: External Memory: Name=ext_d: Alloc: select “ON”

Linker: Script File: External Memory: Name=TRAP_vecttable: Alloc: select “ON”




Linker: Script File: External Memory: Name=ext_c: Size: insert “3M”


Linker: Script File: External Memory: Name=ext_d: Size: insert “1M”

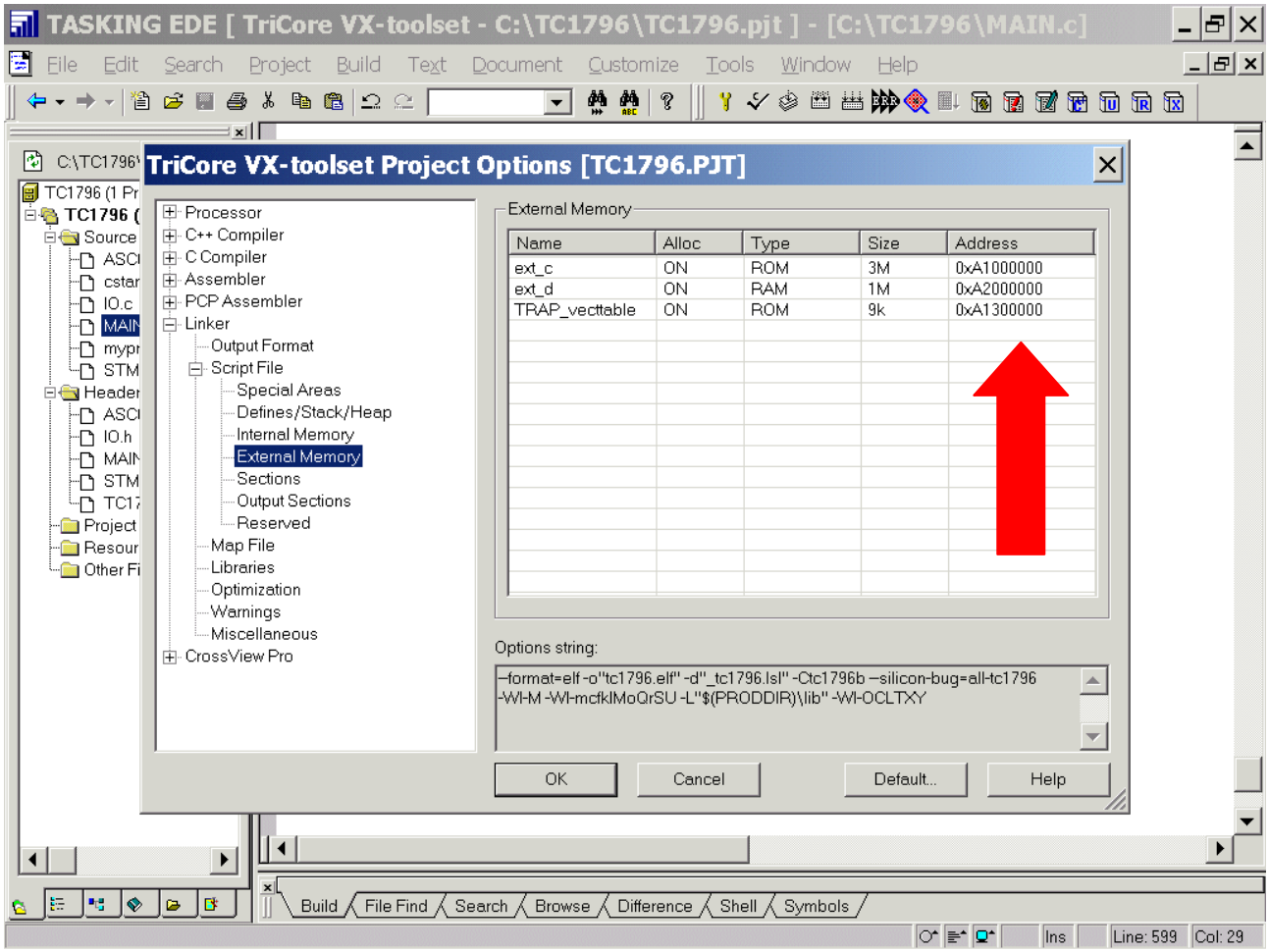


Linker: Script File: External Memory:

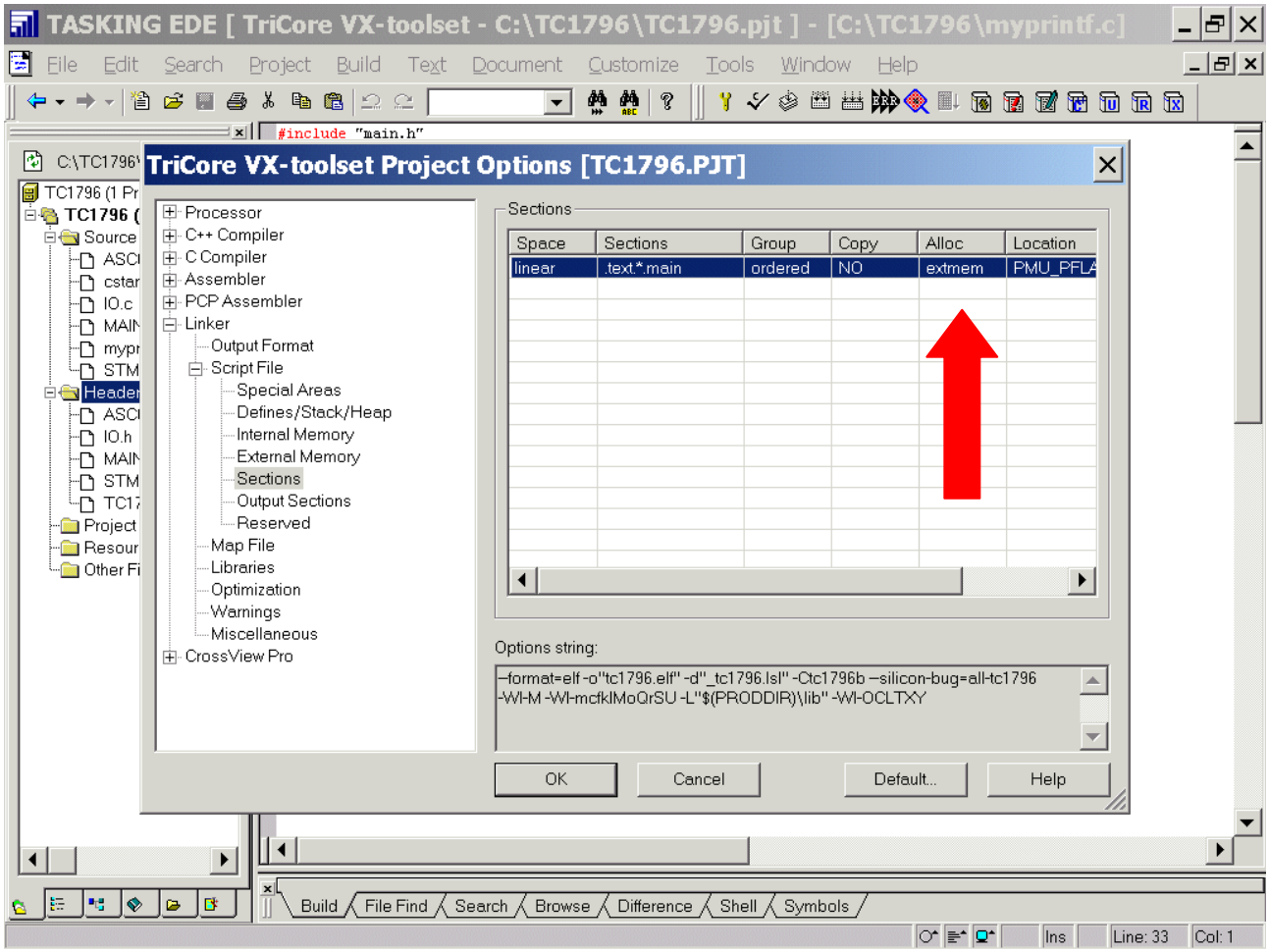
Name=ext_c: Address: insert 0xA1000000

Name=ext_d: Address: insert 0xA2000000

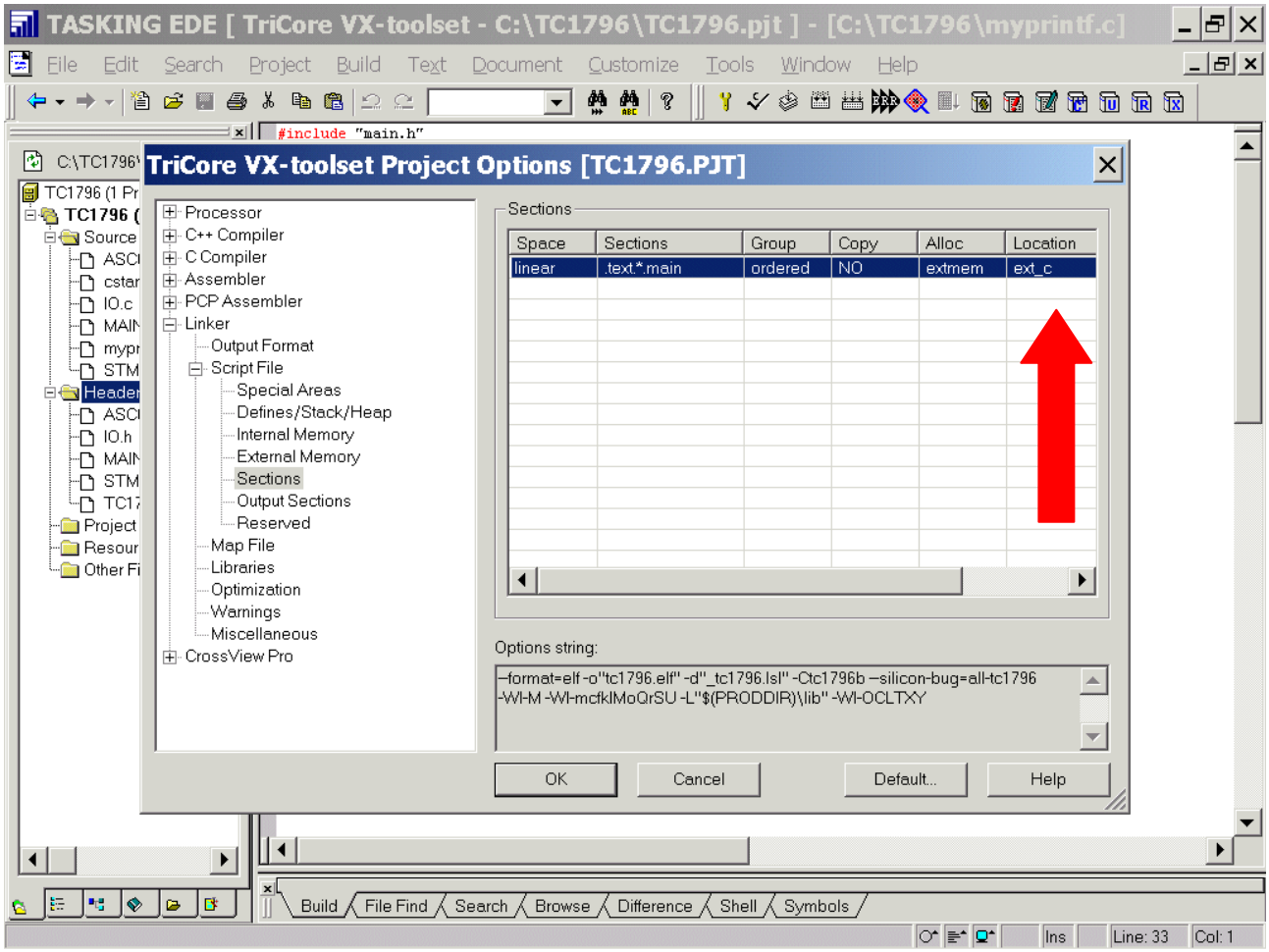
Name=TRAP_vecttable: Address: insert 0xA1300000



Linker: Script File: Sections: Space=linear: Alloc: select “extmem”

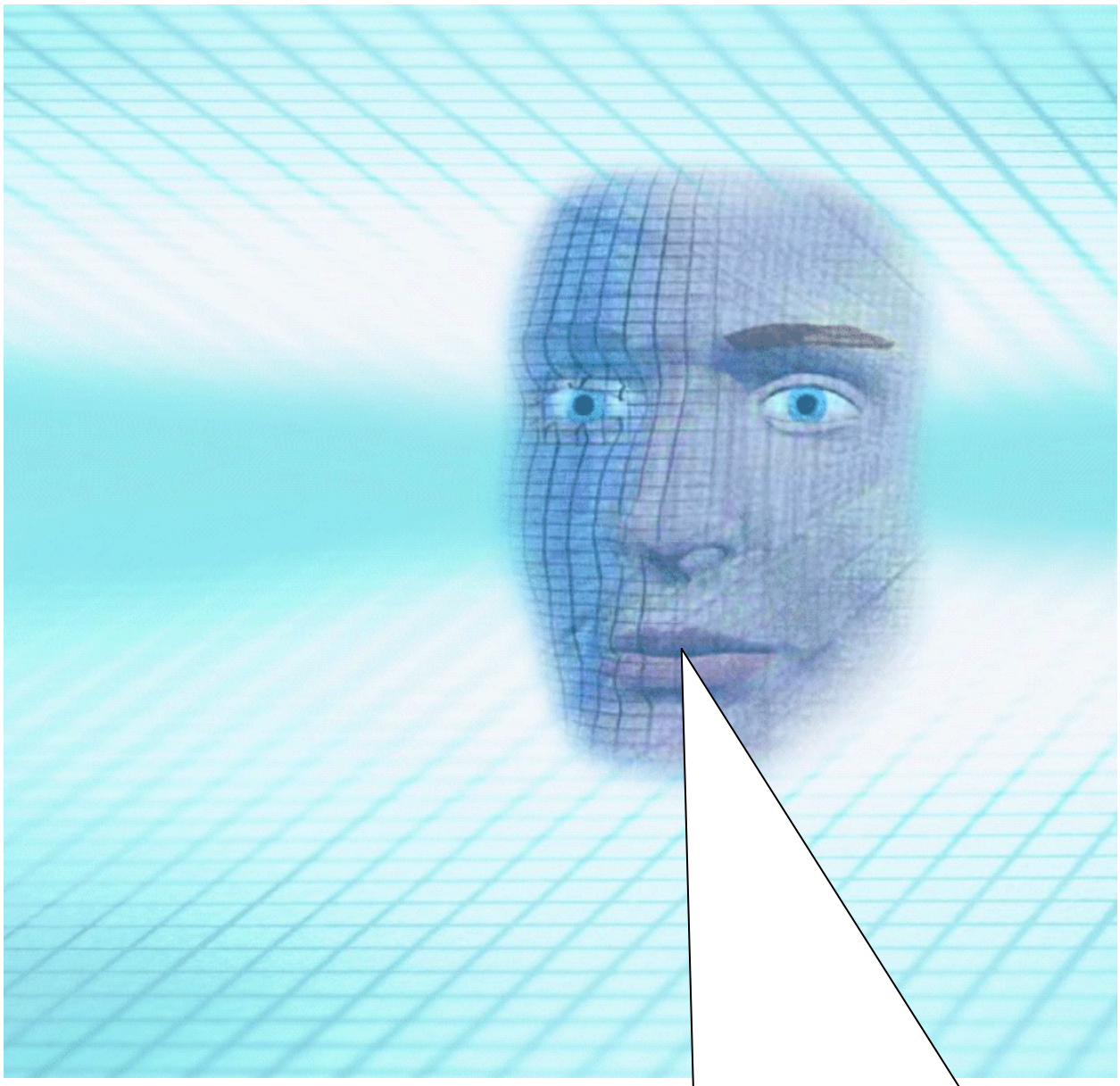


Linker: Script File: Sections: Space=linear: Location: insert “ext_c”



OK

Insert your application specific program:



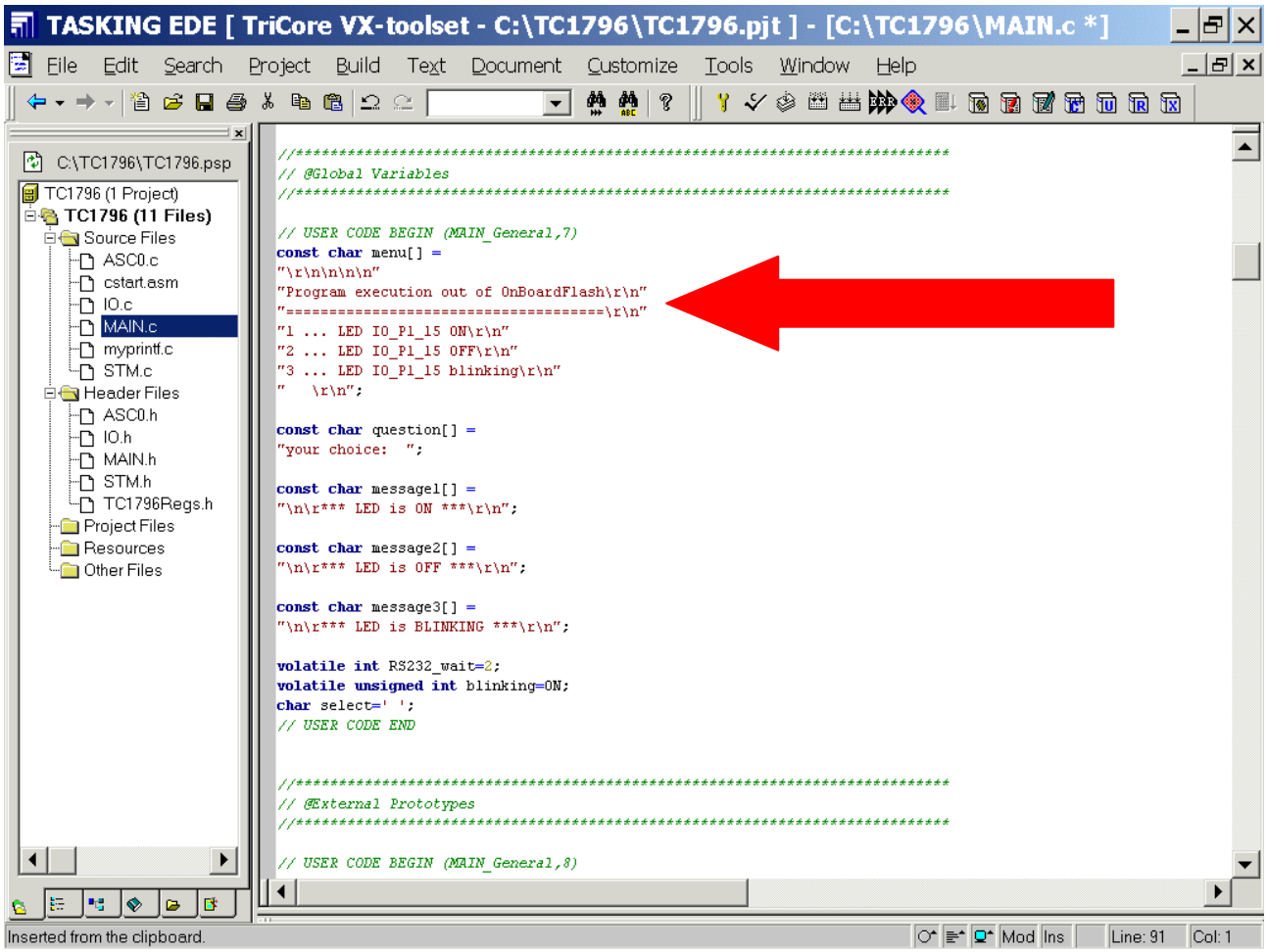
Note:
DAvE doesn't change code which is inserted between '// USER CODE BEGIN' and '// USER CODE END'. Therefore, whenever adding code to DAvE's generated code, write it between '// USER CODE BEGIN' and '// USER CODE END'.
If you wish to change DAvE´s generated code or add code outside these 'USER CODE' sections you will have to insert/modify your changes each time after letting DAvE regenerate code!

Double click: Main.c and change Global Variable menu
from

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of OnChipFlash\r\n"
"====================================\r\n"
"1 ... LED IO_P1_15 ON\r\n"
"2 ... LED IO_P1_15 OFF\r\n"
"3 ... LED IO_P1_15 blinking\r\n"
"   \r\n";
```

to

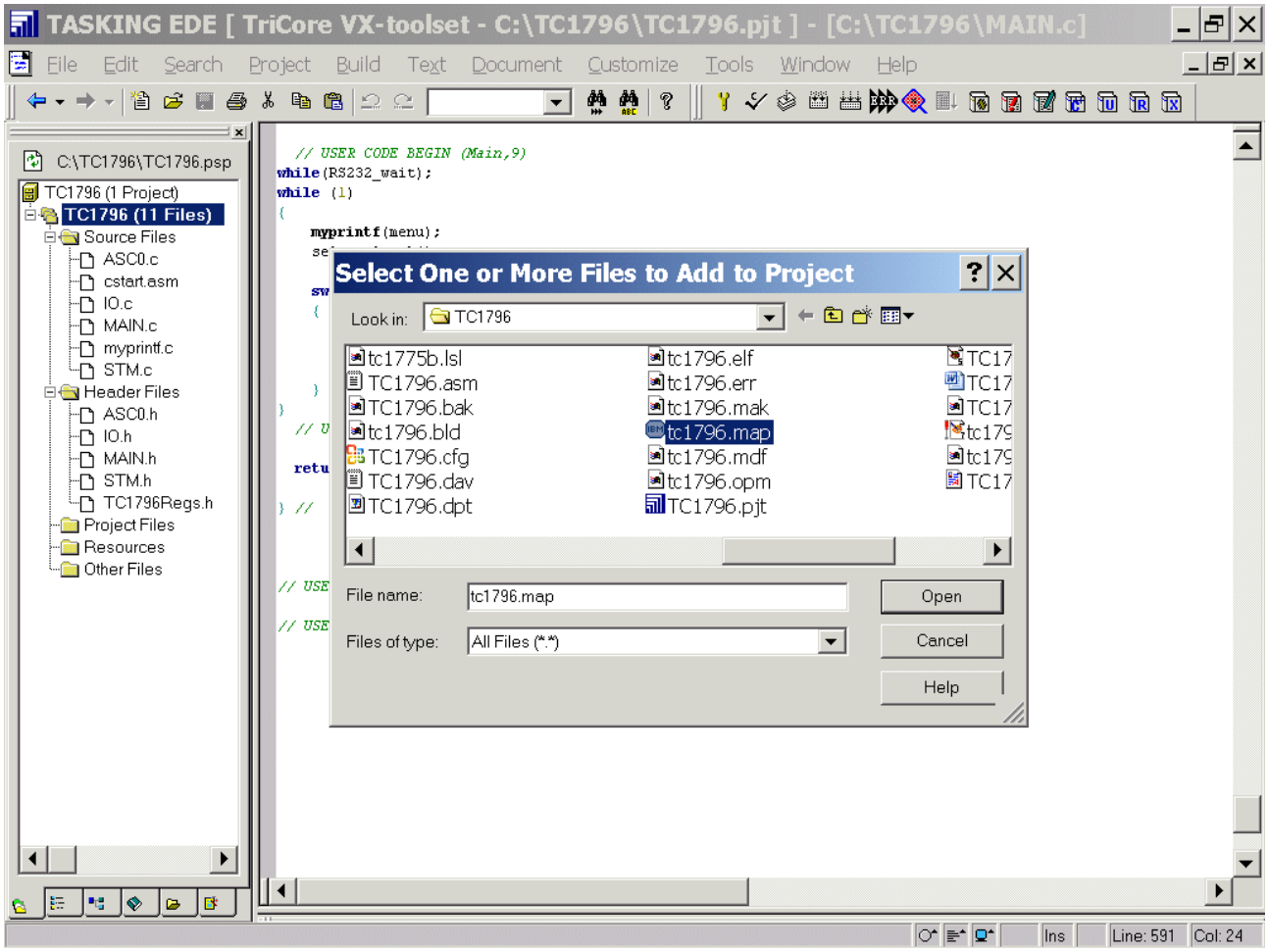
```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of OnBoardFlash\r\n"
"====================================\r\n"
"1 ... LED IO_P1_15 ON\r\n"
"2 ... LED IO_P1_15 OFF\r\n"
"3 ... LED IO_P1_15 blinking\r\n"
"   \r\n";
```



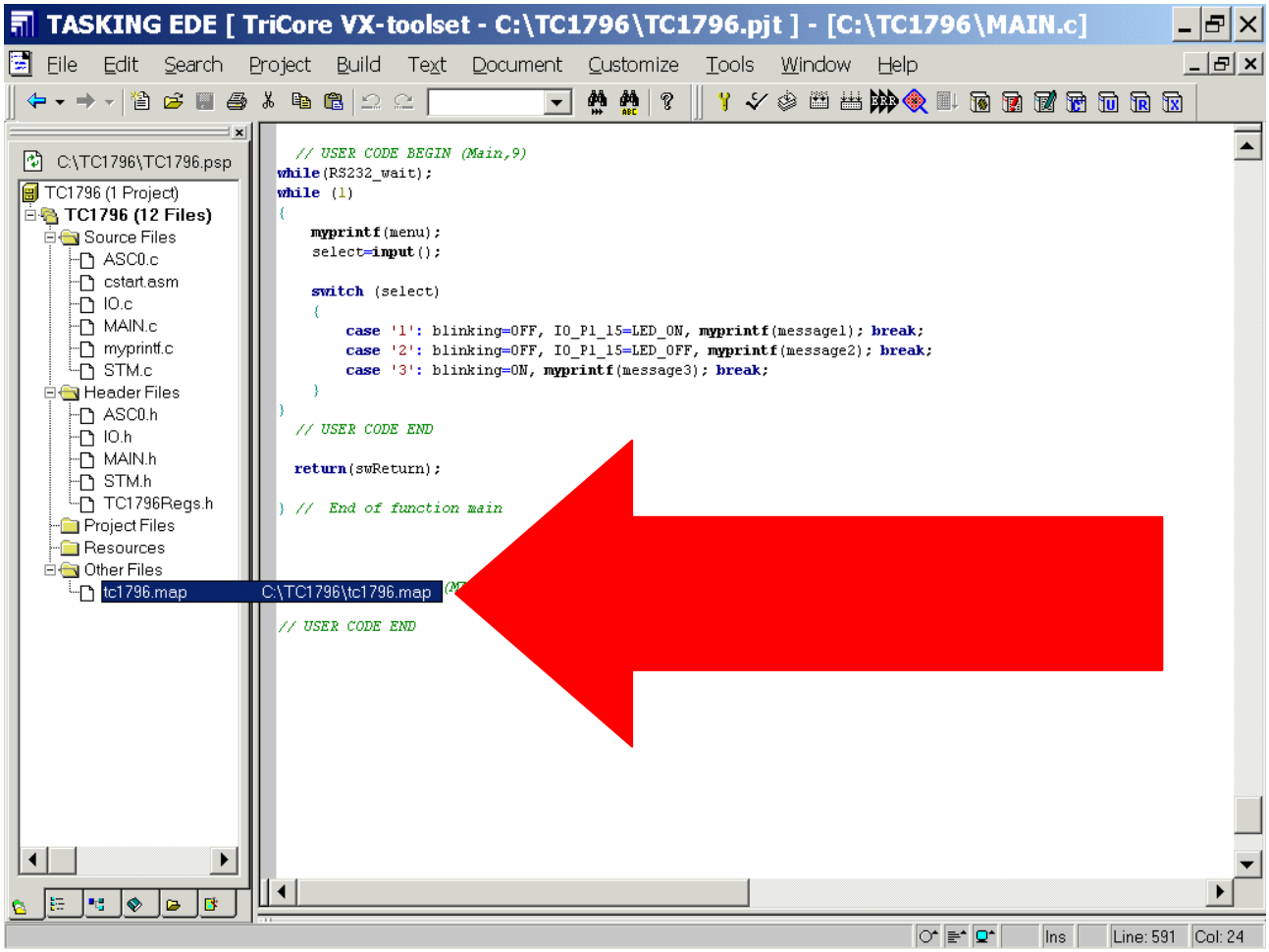
Insert Map File:

(Project Window File View) – TC1796 (Files) – right mouse button click – Add Existing Files – Browse

Select TC1796.map



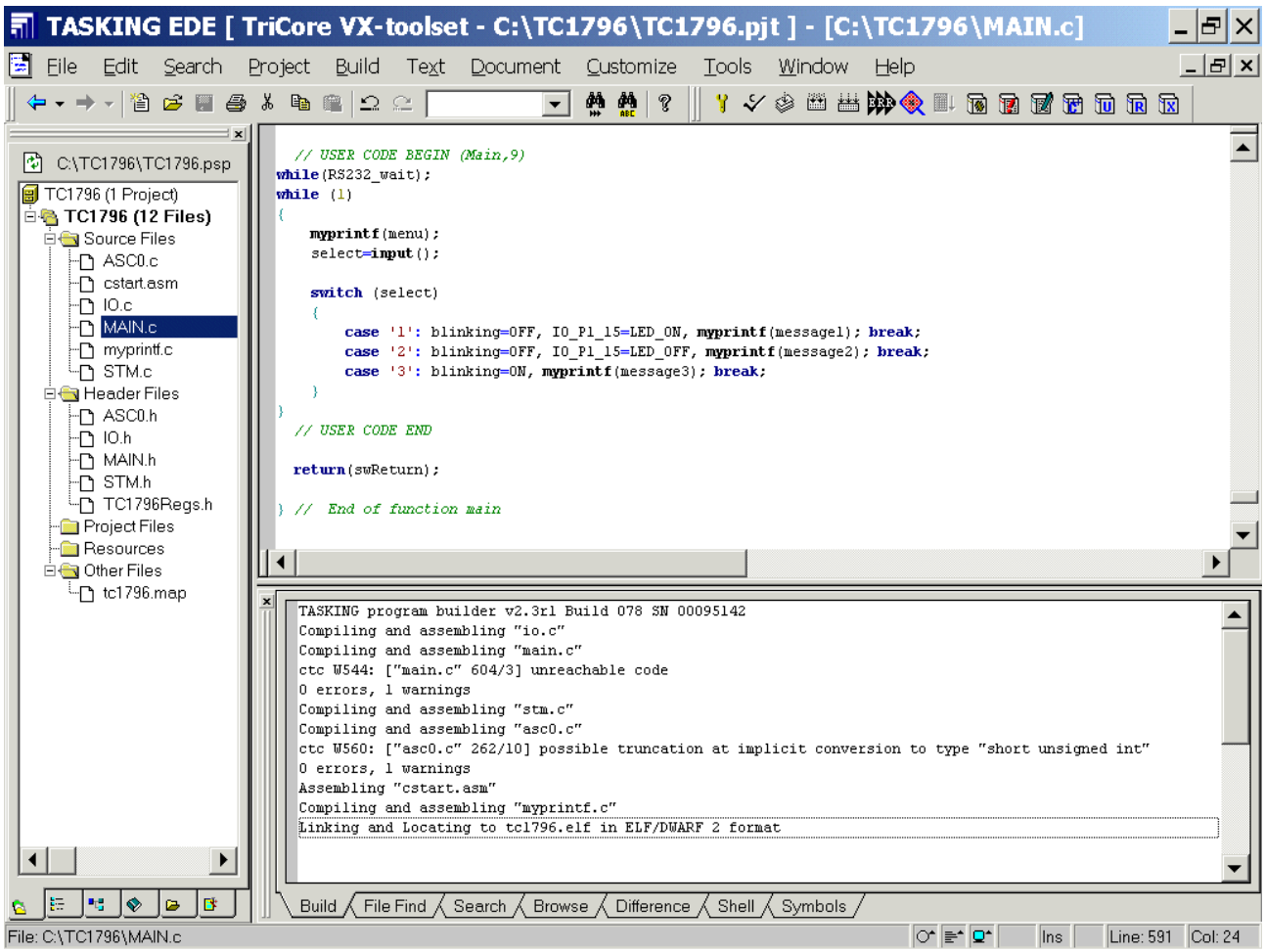
Open - OK


TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\MAIN.c]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1796\TC1796.psp
TC1796 (1 Project)
TC1796 (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
TC1796Regs.h
Project Files
Resources
Other Files
tc1796.map
C:\TC1796\tc1796.map
// USER CODE BEGIN (Main,9)
while(RS232_wait);
while (1)
{
myprintf(menu);
select=input();
switch (select)
{
case '1': blinking=OFF, IO_P1_15=LED_ON, myprintf(message1); break;
case '2': blinking=OFF, IO_P1_15=LED_OFF, myprintf(message2); break;
case '3': blinking=ON, myprintf(message3); break;
}
}
// USER CODE END
return(swReturn);
} // End of function main
// USER CODE END
Ins
Line: 591
Col: 24

Generate your application program:

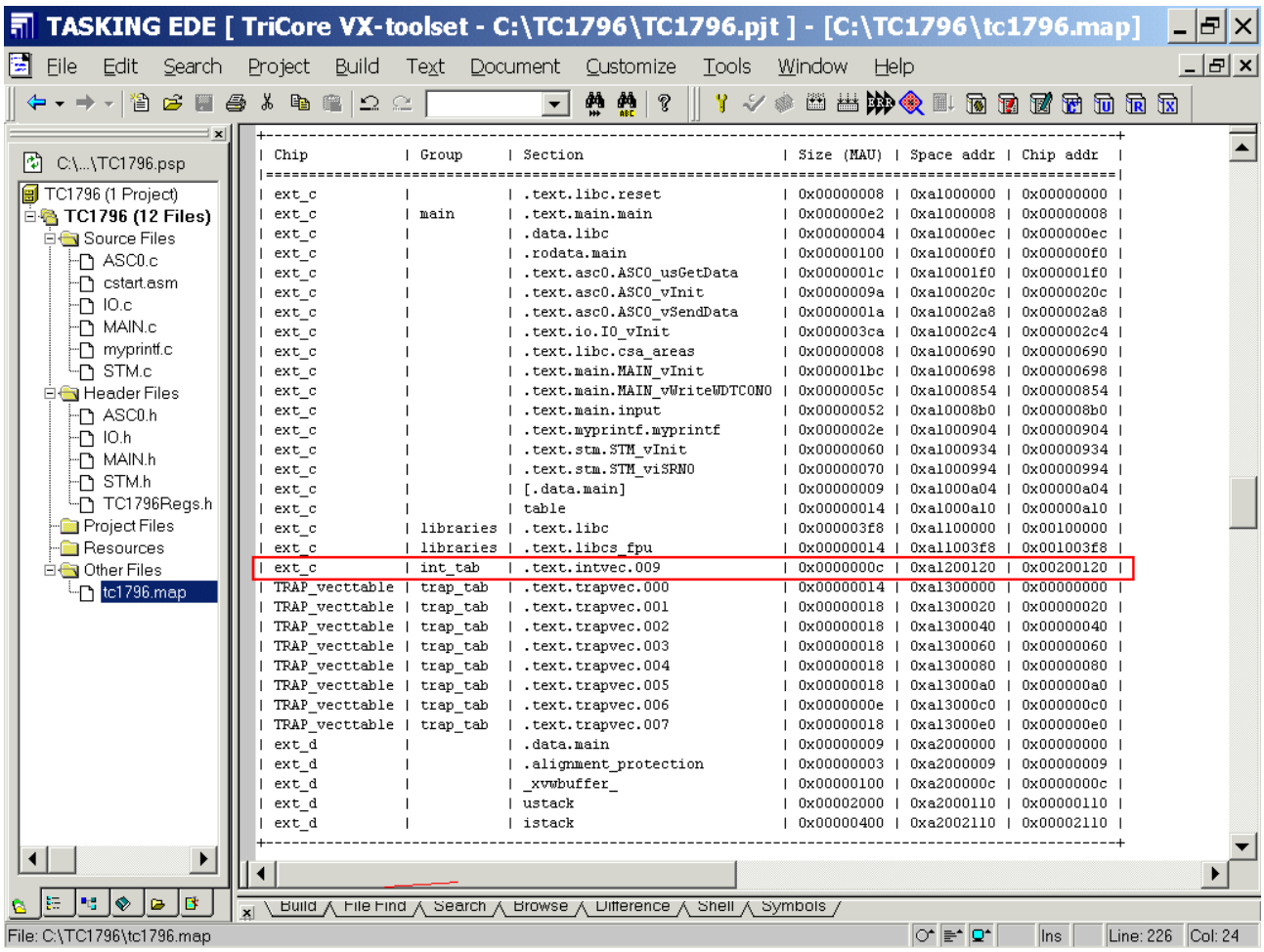
Build
Rebuild

or



See Map File:

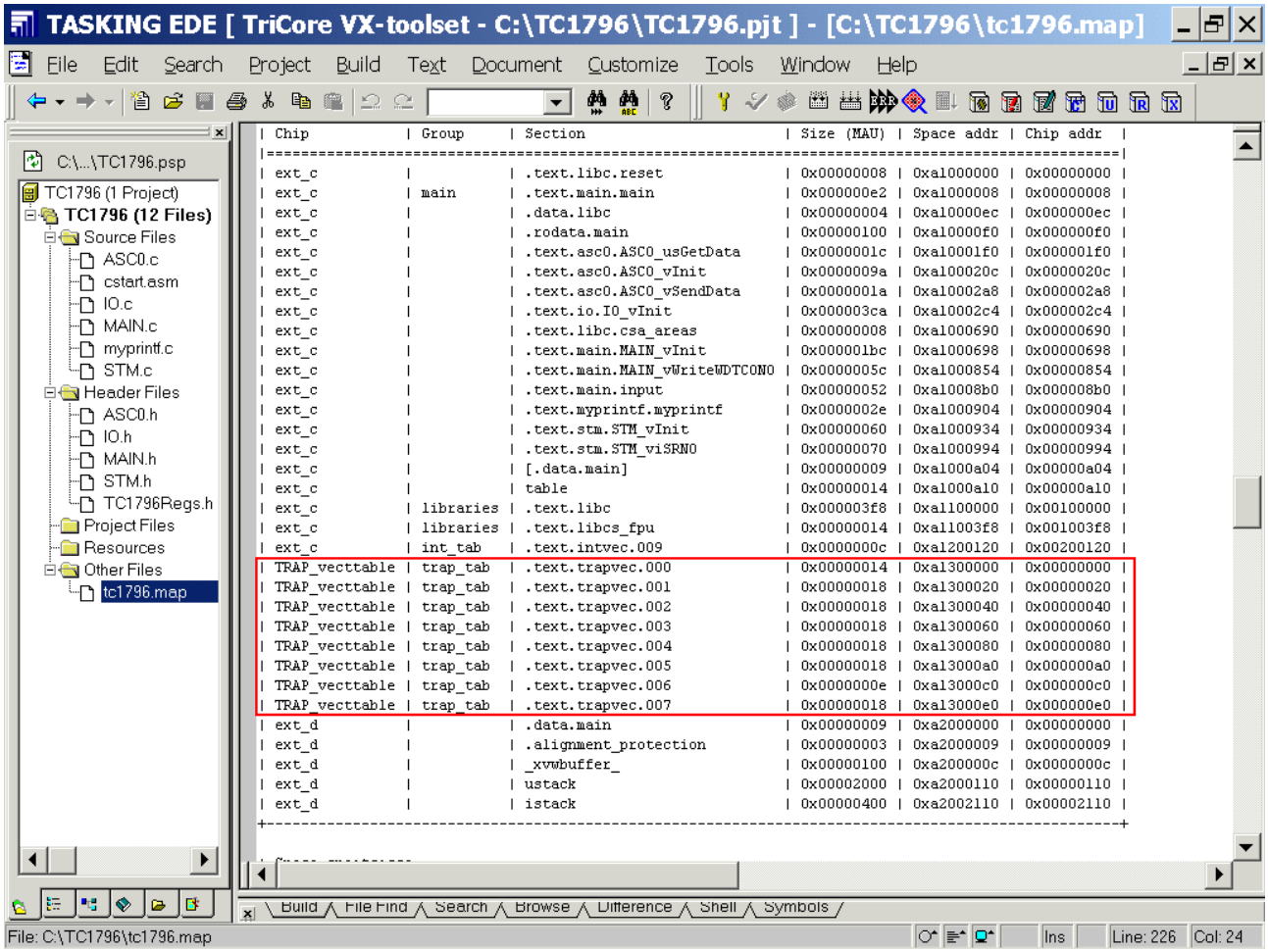
Interrupt Vector Table:



Chip	Group	Section	Size (MAU)	Space addr	Chip addr
ext_c		.text.libc.reset	0x00000008	0xa1000000	0x00000000
ext_c	main	.text.main.main	0x000000e2	0xa1000008	0x00000008
ext_c		.data.libc	0x00000004	0xa10000ec	0x000000ec
ext_c		.rodata.main	0x00000100	0xa10000f0	0x000000f0
ext_c		.text.asc0.ASC0_usGetData	0x0000001c	0xa10001f0	0x000001f0
ext_c		.text.asc0.ASC0_vInit	0x0000009a	0xa100020c	0x0000020c
ext_c		.text.asc0.ASC0_vSendData	0x0000001a	0xa10002a8	0x000002a8
ext_c		.text.io.IO_vInit	0x000003ca	0xa10002c4	0x000002c4
ext_c		.text.libc.csa_areas	0x00000008	0xa1000690	0x00000690
ext_c		.text.main.MAIN_vInit	0x000001bc	0xa1000698	0x00000698
ext_c		.text.main.MAIN_vWriteWDTCON0	0x0000005c	0xa1000854	0x00000854
ext_c		.text.main.input	0x00000052	0xa10008b0	0x000008b0
ext_c		.text.myprintf.myprintf	0x0000002e	0xa1000904	0x00000904
ext_c		.text.stm.STM_vInit	0x00000060	0xa1000934	0x00000934
ext_c		.text.stm.STM_viSRN0	0x00000070	0xa1000994	0x00000994
ext_c		[.data.main]	0x00000009	0xa1000a04	0x00000a04
ext_c		table	0x00000014	0xa1000a10	0x00000a10
ext_c	libraries	.text.libc	0x000003f8	0xa1100000	0x00100000
ext_c	libraries	.text.libcs_fpu	0x00000014	0xa11003f8	0x001003f8
ext_c	int_tab	.text.intvec.009	0x0000000c	0xa1200120	0x00200120
TRAP_vecttable	trap_tab	.text.trapvec.000	0x00000014	0xa1300000	0x00000000
TRAP_vecttable	trap_tab	.text.trapvec.001	0x00000018	0xa1300020	0x00000020
TRAP_vecttable	trap_tab	.text.trapvec.002	0x00000018	0xa1300040	0x00000040
TRAP_vecttable	trap_tab	.text.trapvec.003	0x00000018	0xa1300060	0x00000060
TRAP_vecttable	trap_tab	.text.trapvec.004	0x00000018	0xa1300080	0x00000080
TRAP_vecttable	trap_tab	.text.trapvec.005	0x00000018	0xa13000a0	0x000000a0
TRAP_vecttable	trap_tab	.text.trapvec.006	0x0000000e	0xa13000c0	0x000000c0
TRAP_vecttable	trap_tab	.text.trapvec.007	0x00000018	0xa13000e0	0x000000e0
ext_d		.data.main	0x00000009	0xa2000000	0x00000000
ext_d		.alignment_protection	0x00000003	0xa2000009	0x00000009
ext_d		_xvwbuffer_	0x00000100	0xa200000c	0x0000000c
ext_d		ustack	0x00002000	0xa2000110	0x00000110
ext_d		istack	0x00000400	0xa2002110	0x00002110

Click here to see Memory Map

Trap Vector Table:



Chip	Group	Section	Size (MAU)	Space addr	Chip addr
ext_c		.text.libc.reset	0x00000008	0xa1000000	0x00000000
ext_c	main	.text.main.main	0x000000e2	0xa1000008	0x00000008
ext_c		.data.libc	0x00000004	0xa10000ec	0x000000ec
ext_c		.rodata.main	0x00000100	0xa10000f0	0x000000f0
ext_c		.text.asc0.ASC0_usGetData	0x0000001c	0xa1001f0	0x000001f0
ext_c		.text.asc0.ASC0_vInit	0x0000009a	0xa100020c	0x0000020c
ext_c		.text.asc0.ASC0_vSendData	0x0000001a	0xa10002a8	0x000002a8
ext_c		.text.io.IO_vInit	0x000003ca	0xa10002c4	0x000002c4
ext_c		.text.libc.csa_areas	0x00000008	0xa1000690	0x00000690
ext_c		.text.main.MAIN_vInit	0x000001bc	0xa1000698	0x00000698
ext_c		.text.main.MAIN_vWriteWDTCON0	0x0000005c	0xa1000854	0x00000854
ext_c		.text.main.input	0x00000052	0xa10008b0	0x000008b0
ext_c		.text.myprintf.myprintf	0x0000002e	0xa1000904	0x00000904
ext_c		.text.stm.STM_vInit	0x00000060	0xa1000934	0x00000934
ext_c		.text.stm.STM_viSRN0	0x00000070	0xa1000994	0x00000994
ext_c		[.data.main]	0x00000009	0xa1000a04	0x00000a04
ext_c		table	0x00000014	0xa1000a10	0x00000a10
ext_c	libraries	.text.libc	0x000003f8	0xa1100000	0x00100000
ext_c	libraries	.text.libcs_fpu	0x00000014	0xa1003f8	0x001003f8
ext_c	int_tab	.text.intvec.009	0x0000000c	0xa1200120	0x00200120
TRAP_vecttable	trap_tab	.text.trapvec.000	0x00000014	0xa1300000	0x00000000
TRAP_vecttable	trap_tab	.text.trapvec.001	0x00000018	0xa1300020	0x00000020
TRAP_vecttable	trap_tab	.text.trapvec.002	0x00000018	0xa1300040	0x00000040
TRAP_vecttable	trap_tab	.text.trapvec.003	0x00000018	0xa1300060	0x00000060
TRAP_vecttable	trap_tab	.text.trapvec.004	0x00000018	0xa1300080	0x00000080
TRAP_vecttable	trap_tab	.text.trapvec.005	0x00000018	0xa13000a0	0x000000a0
TRAP_vecttable	trap_tab	.text.trapvec.006	0x0000000e	0xa13000c0	0x000000c0
TRAP_vecttable	trap_tab	.text.trapvec.007	0x00000018	0xa13000e0	0x000000e0
ext_d		.data.main	0x00000009	0xa2000000	0x00000000
ext_d		.alignment_protection	0x00000003	0xa2000009	0x00000009
ext_d		_xvwbuffer_	0x00000100	0xa200000c	0x0000000c
ext_d		ustack	0x00002000	0xa2000110	0x00000110
ext_d		istack	0x00000400	0xa2002110	0x00002110

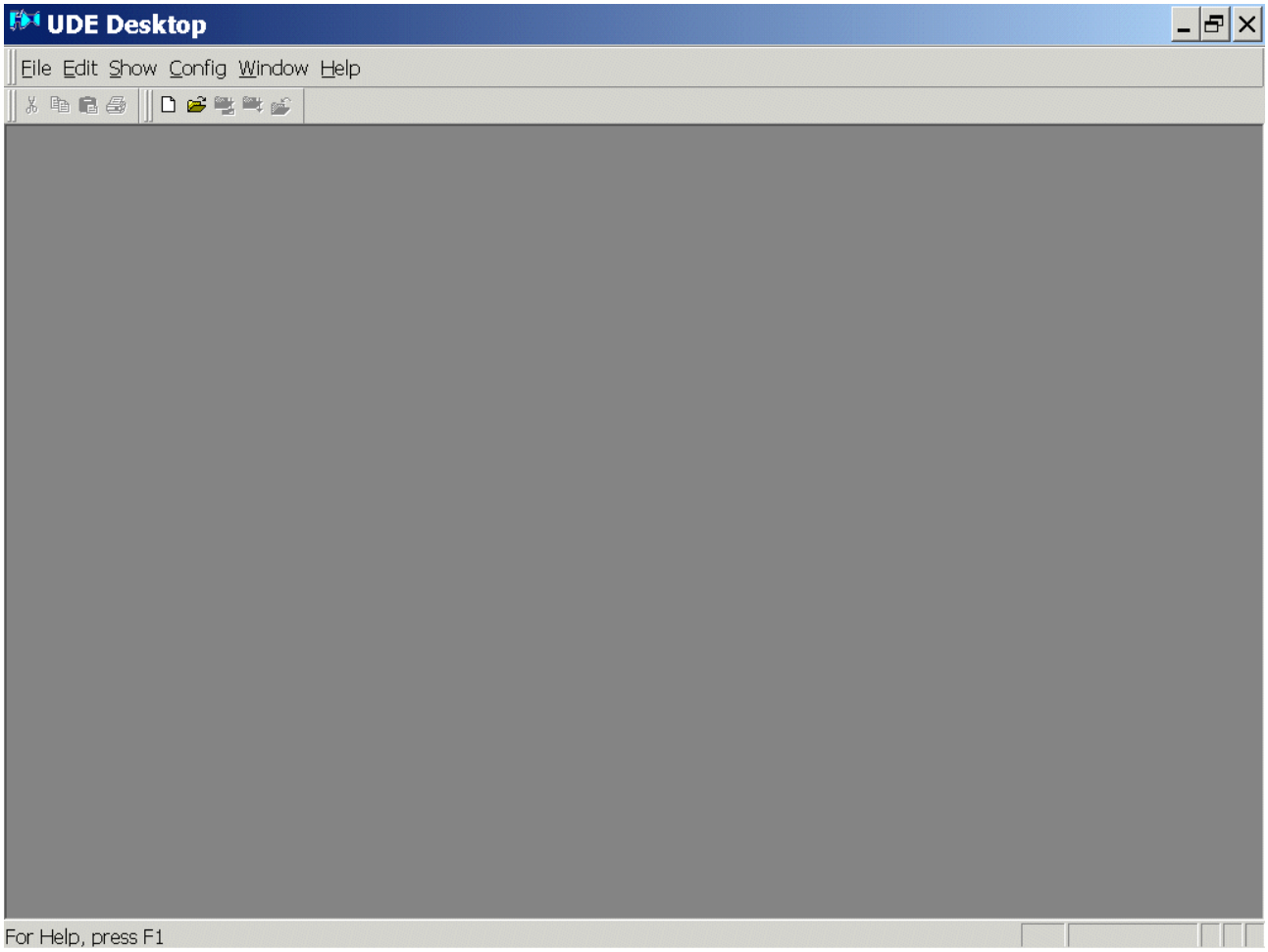
Click here to see Memory Map

Now you can close your project and Tasking EDE:

File - Close Project Space
File – Exit

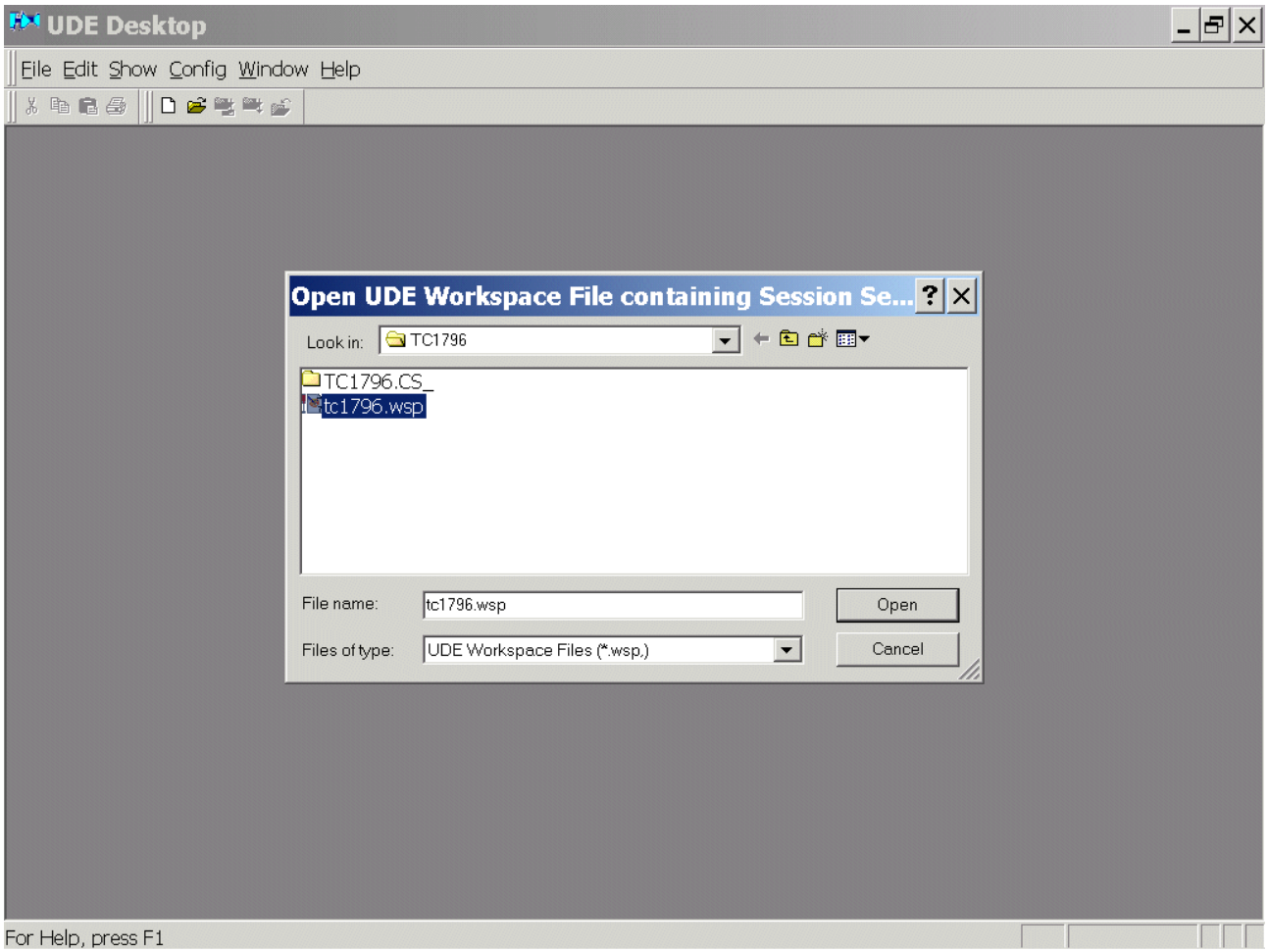
Programming is now complete. You can now load and run your program:

Start pls-Debugger



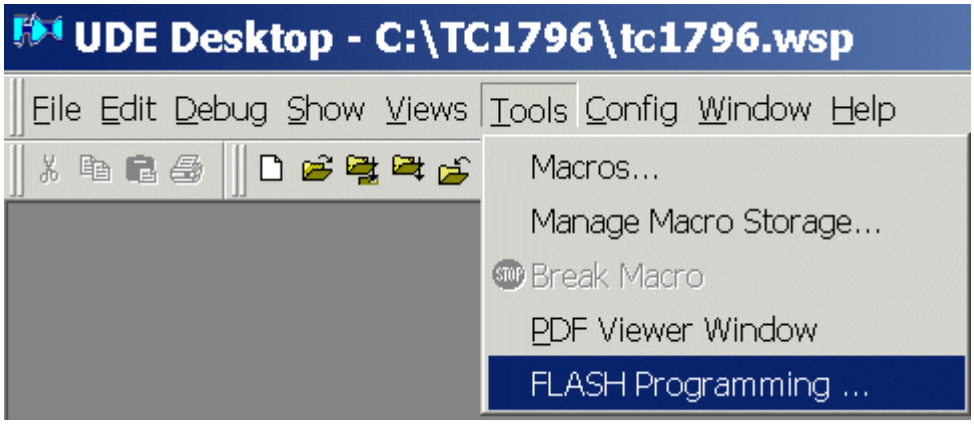
File – Open Workspace

Look in: select C:\TC1796
File name: select tc1796.wsp

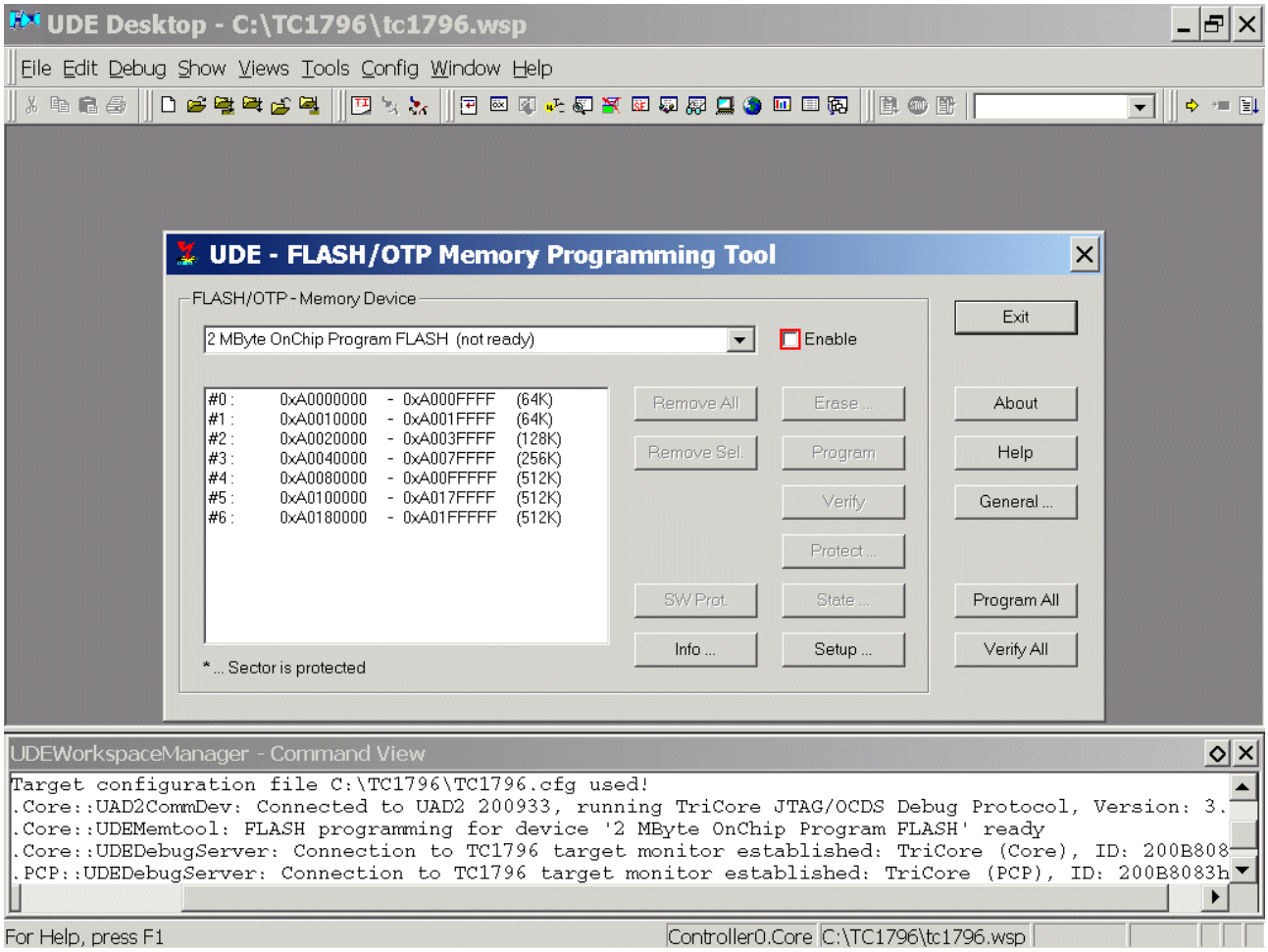


Open
Cancel

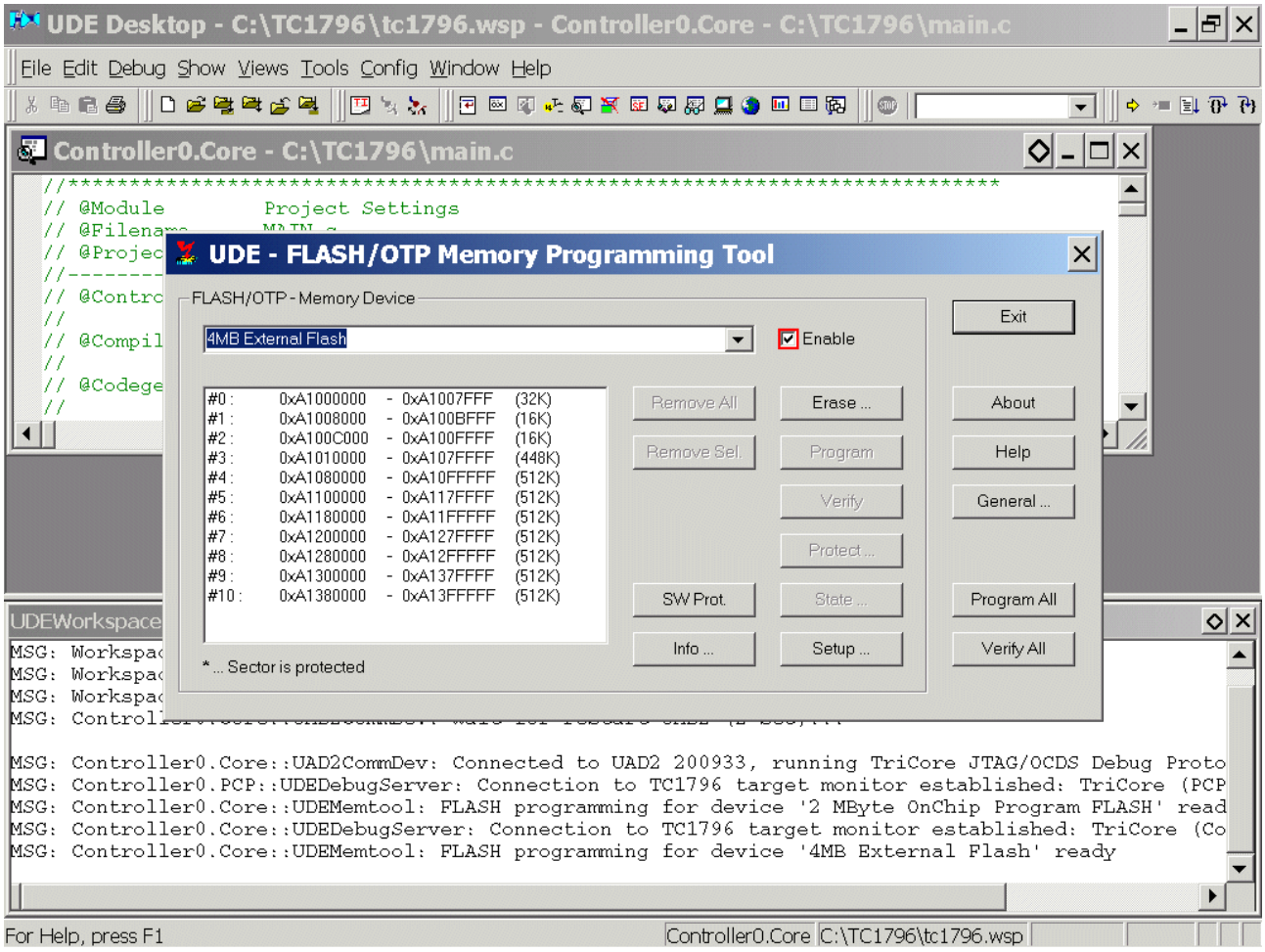
Tools – FLASH Programming …



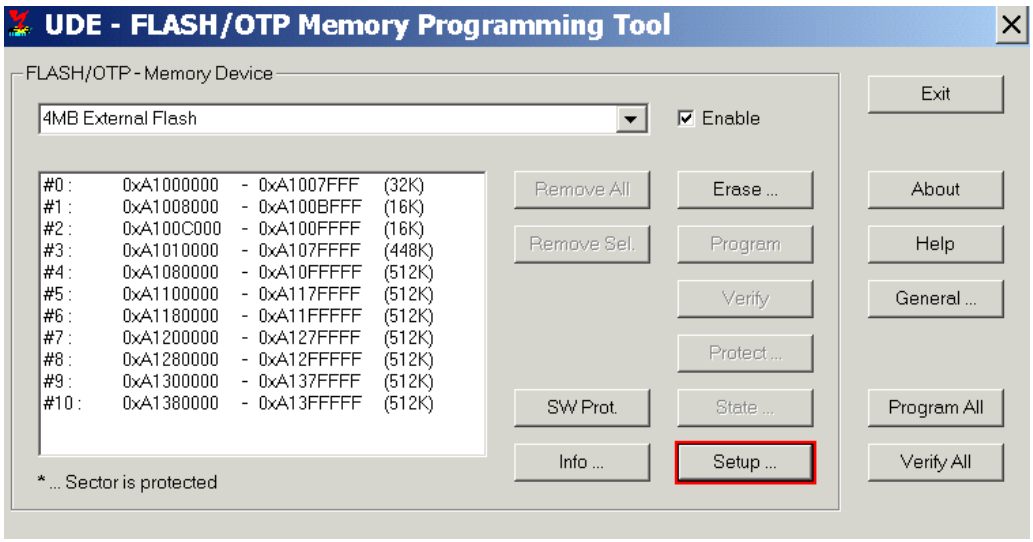
FLASH/OTP – Memory Device: select 2 MByte OnChip Program FLASH:
FLASH/OTP – Memory Device: unclick to deactivate ☐ Enable



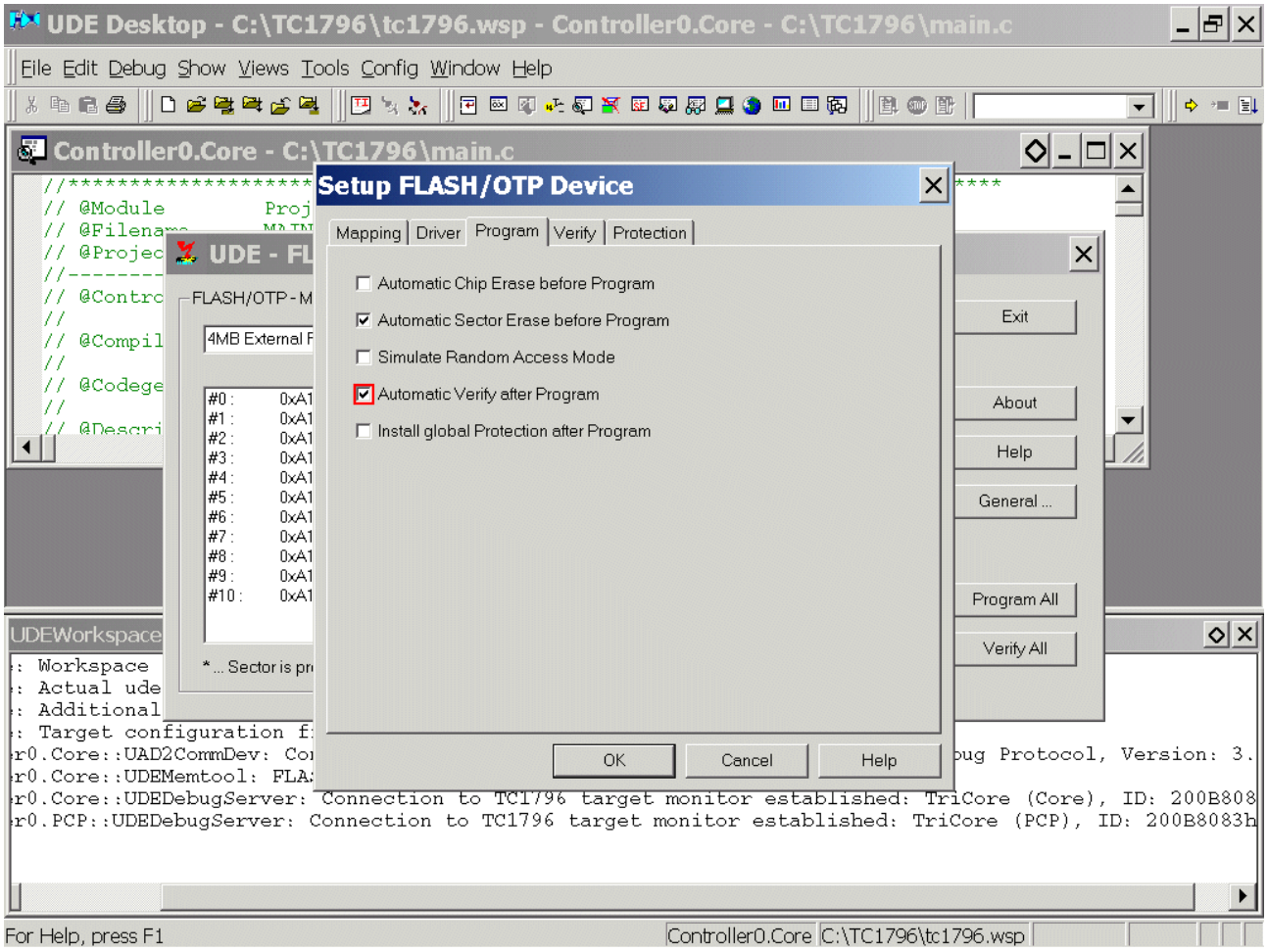
FLASH/OTP – Memory Device: select 4 MB External FLASH
FLASH/OTP – Memory Device: click ✓ Enable



click Setup …

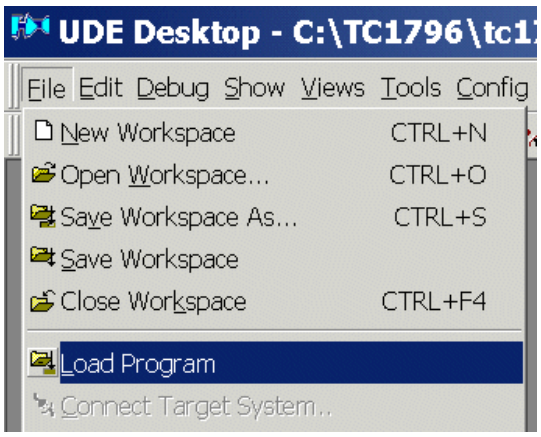


Program: click ✓ Automatic Verify after Program

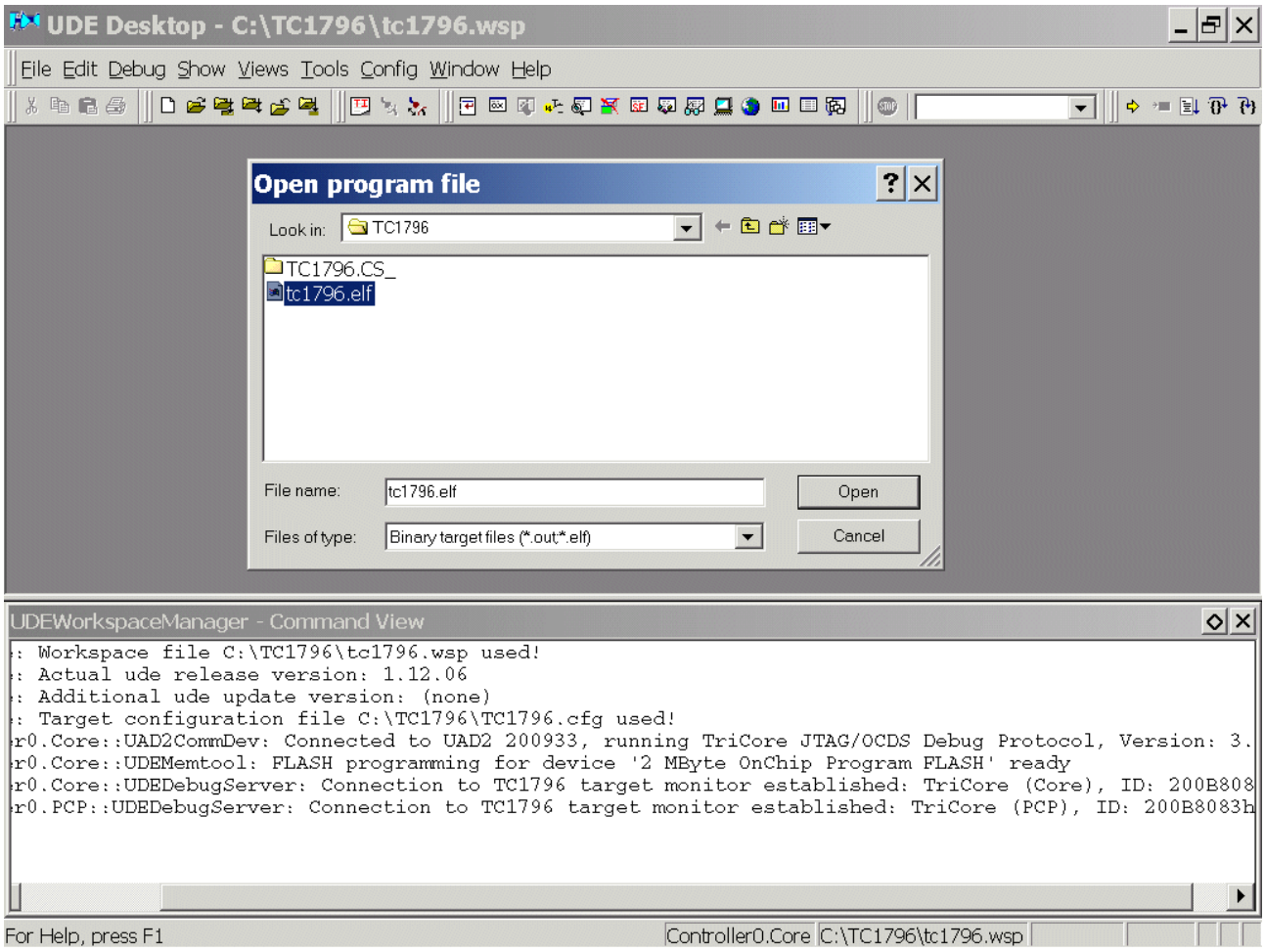


OK
Exit

File – Load Program

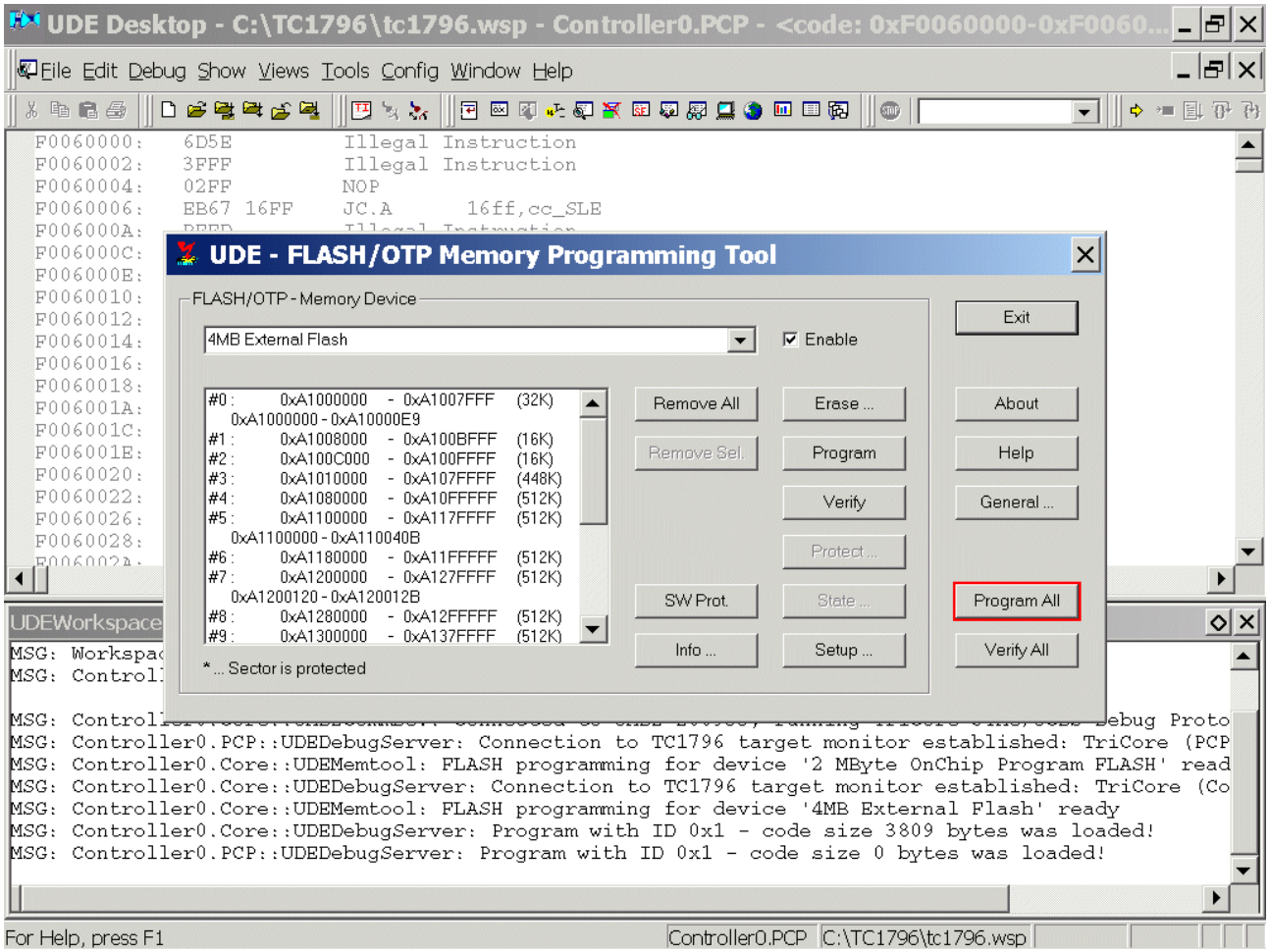


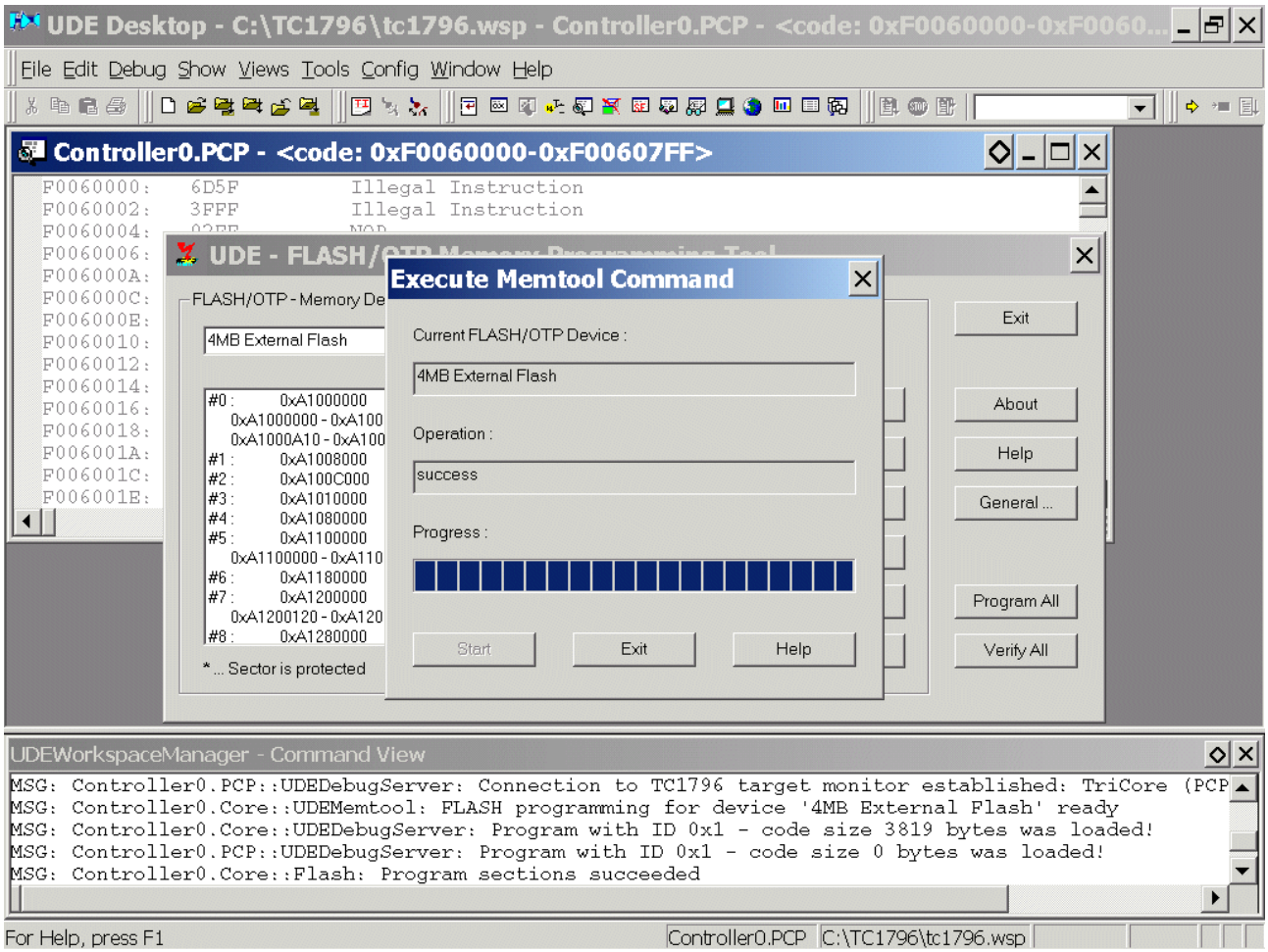
Look in: select TC1796
File name: select tc1796.elf



Open

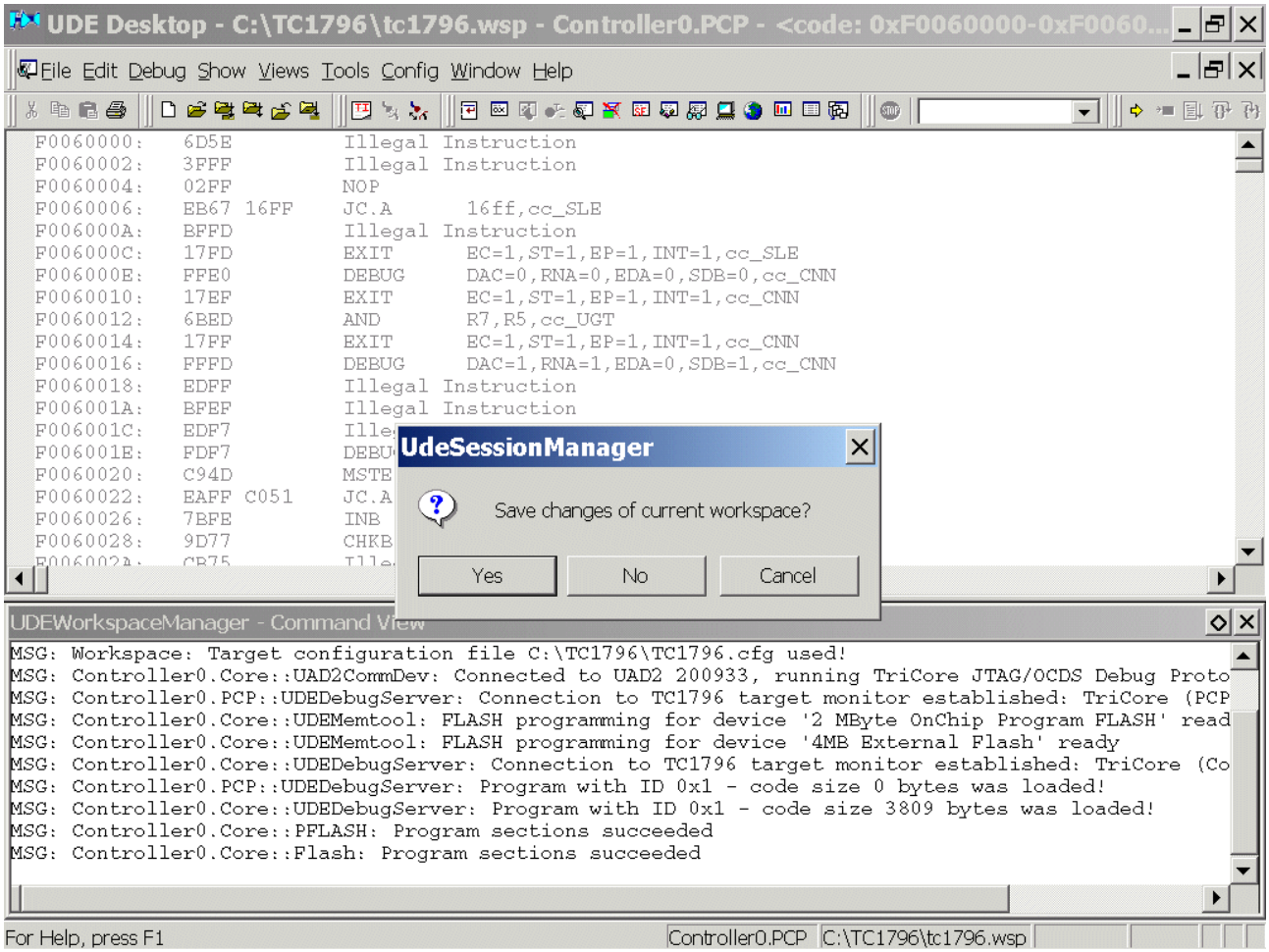
Click Program All




UDE Desktop - C:\TC1796\tc1796.wsp - Controller0.PCP - <code: 0xF0060000-0xF0060...
File Edit Debug Show Views Tools Config Window Help
Controller0.PCP - <code: 0xF0060000-0xF00607FF>
F0060000: 6D5F Illegal Instruction
F0060002: 3FFF Illegal Instruction
UDE - FLASH/OTP Memory Programming Tool
Execute Memtool Command
Current FLASH/OTP Device :
4MB External Flash
Operation :
success
Progress :
Start
Exit
Help
FLASH/OTP - Memory De
4MB External Flash
#0 : 0xA1000000
0xA1000000 - 0xA100
0xA1000A10 - 0xA100
#1 : 0xA1008000
#2 : 0xA100C000
#3 : 0xA1010000
#4 : 0xA1080000
#5 : 0xA1100000
0xA1100000 - 0xA110
#6 : 0xA1180000
#7 : 0xA1200000
0xA1200120 - 0xA120
#8 : 0xA1280000
* ... Sector is protected
Exit
About
Help
General ...
Program All
Verify All
UDEWorkspaceManager - Command View
MSG: Controller0.PCP::UDEDebugServer: Connection to TC1796 target monitor established: TriCore (PCP
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '4MB External Flash' ready
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3819 bytes was loaded!
MSG: Controller0.PCP::UDEDebugServer: Program with ID 0x1 - code size 0 bytes was loaded!
MSG: Controller0.Core::Flash: Program sections succeeded
For Help, press F1
Controller0.PCP
C:\TC1796\tc1796.wsp

Exit
Exit

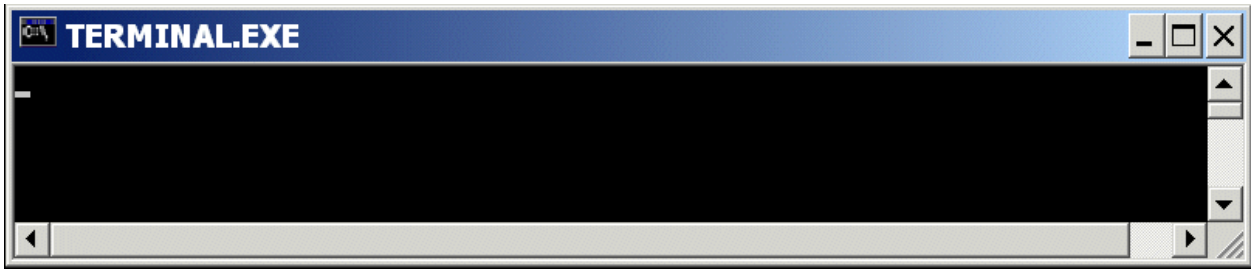
File – Close Workspace



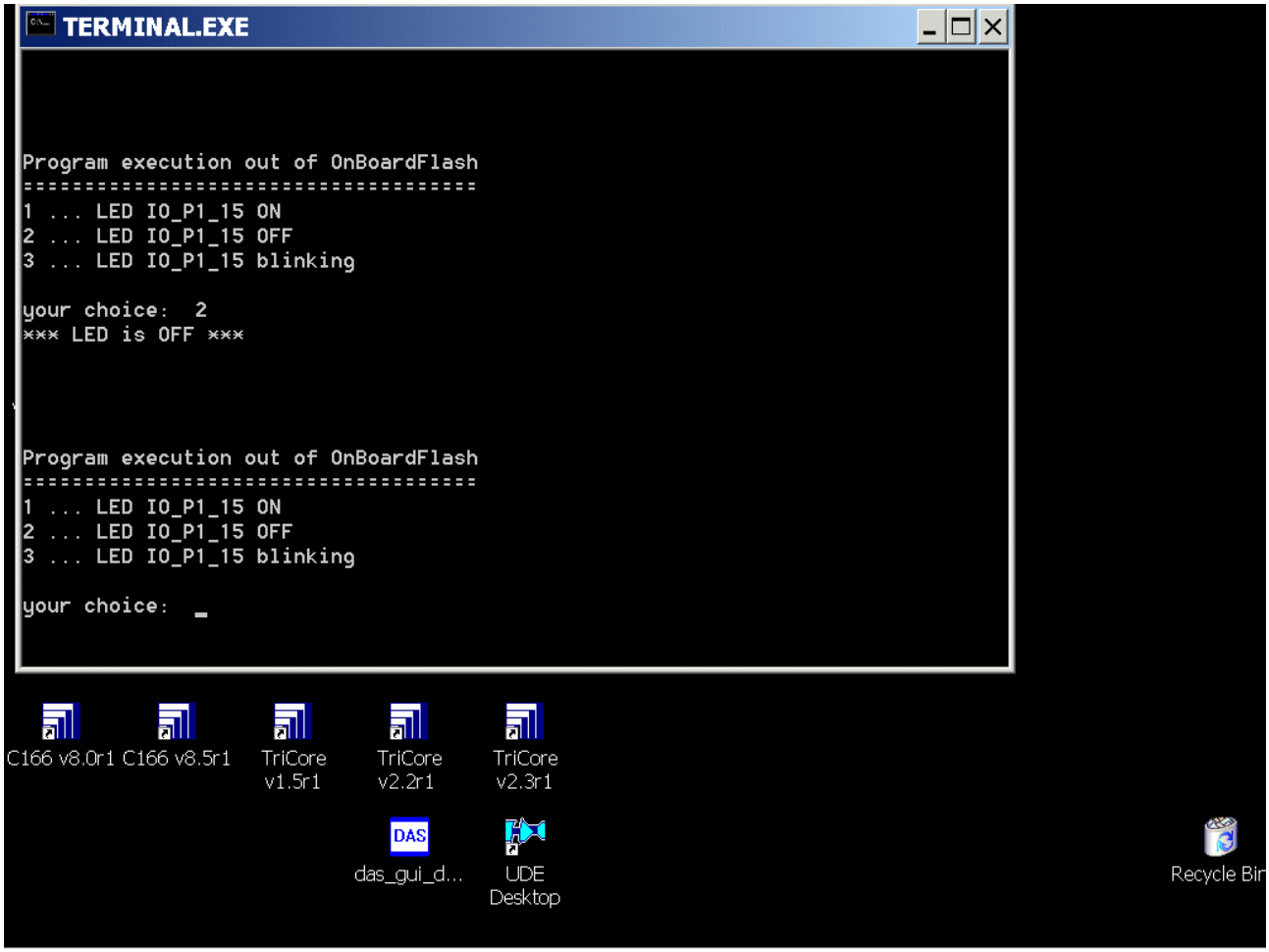
Yes

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:



OnBoardSRAM Test:

Start Tasking EDE and open the project:

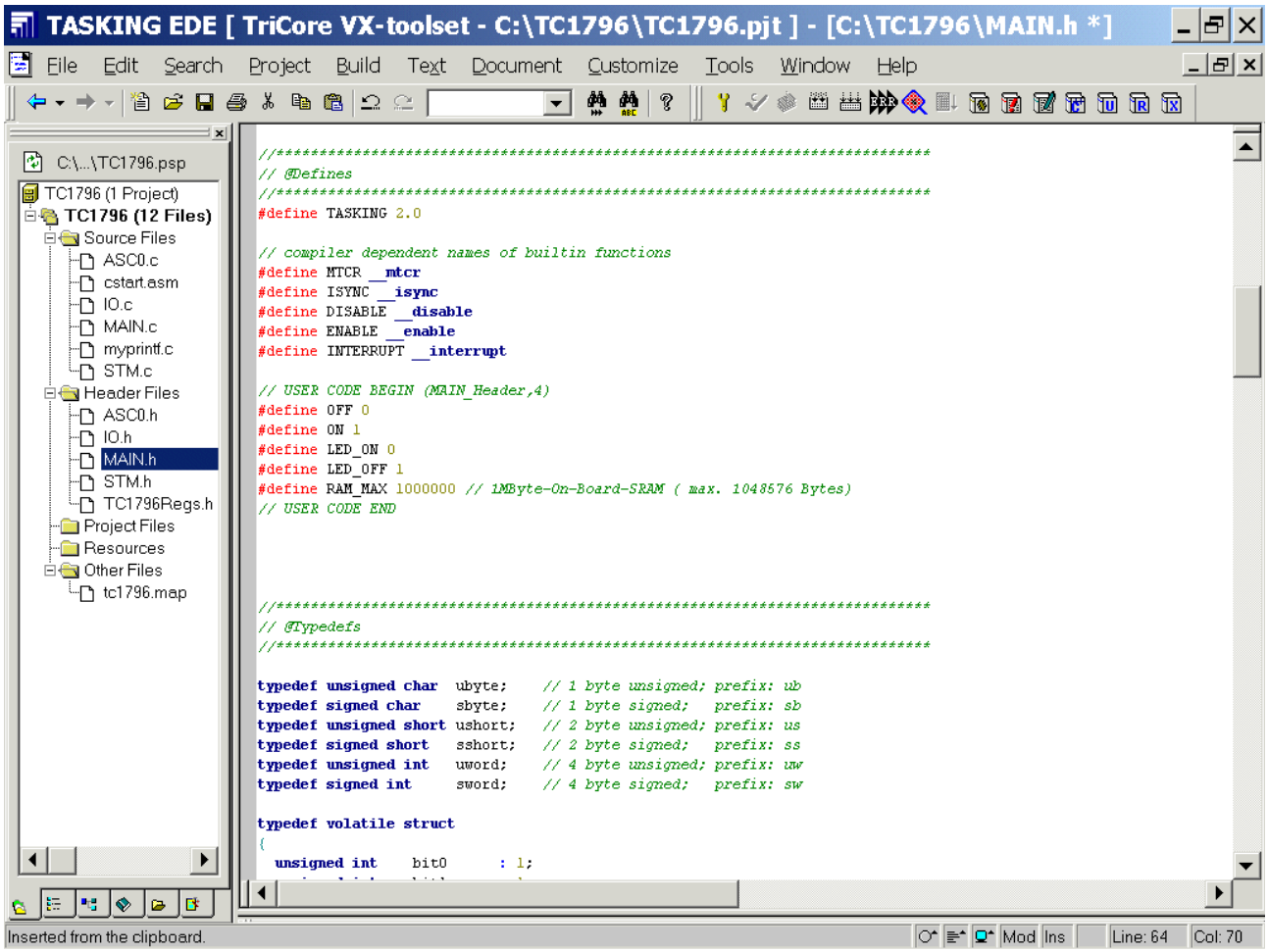
File – Open Project Space

Look in: select C:\TC1796
File name: select TC1796.psp

Insert your application specific program:

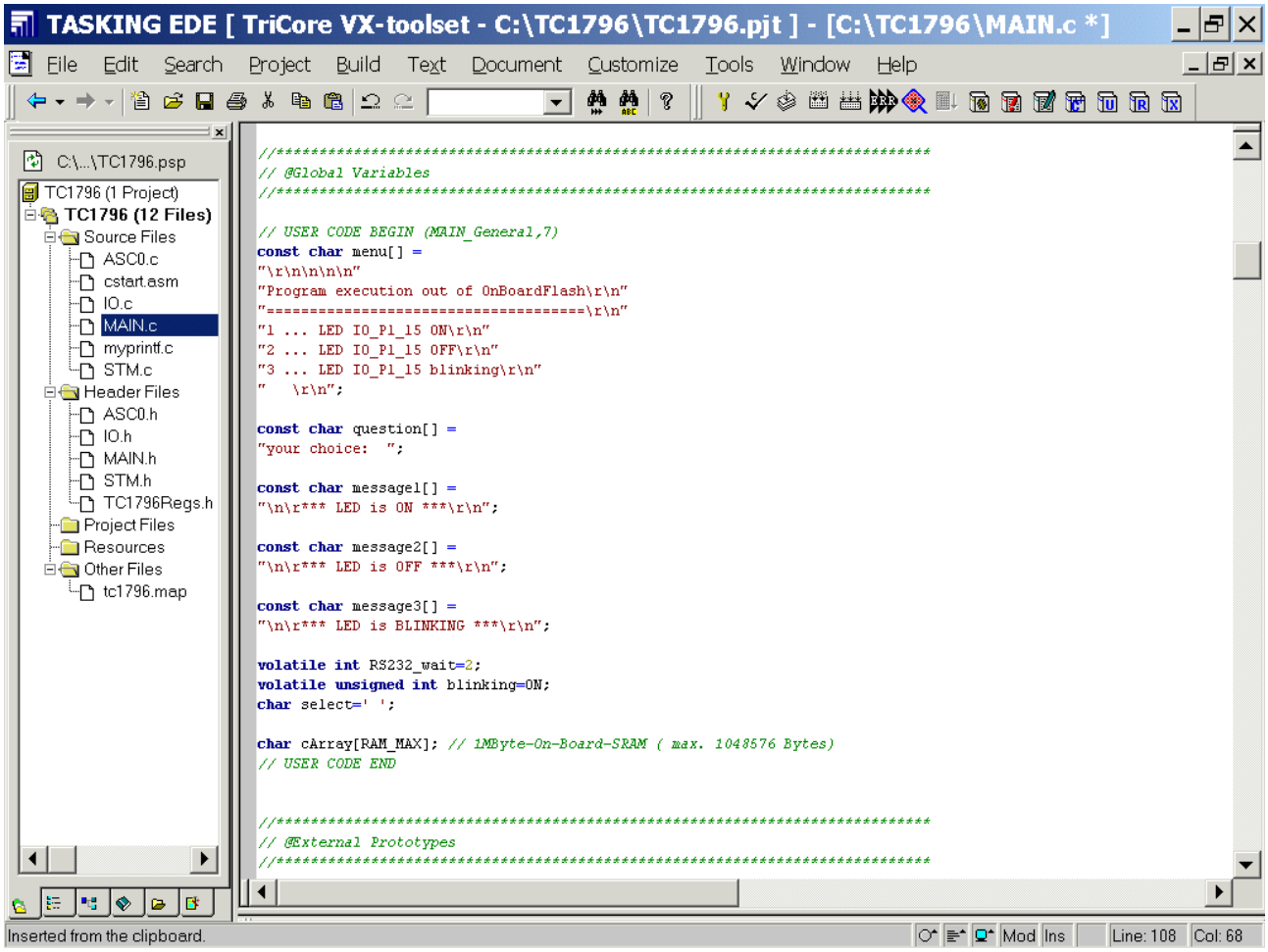
Double click: Main.h and insert the following Defines:

```
#define RAM_MAX 1000000 // 1MByte-On-Board-SRAM ( max. 1048576 Bytes)
```



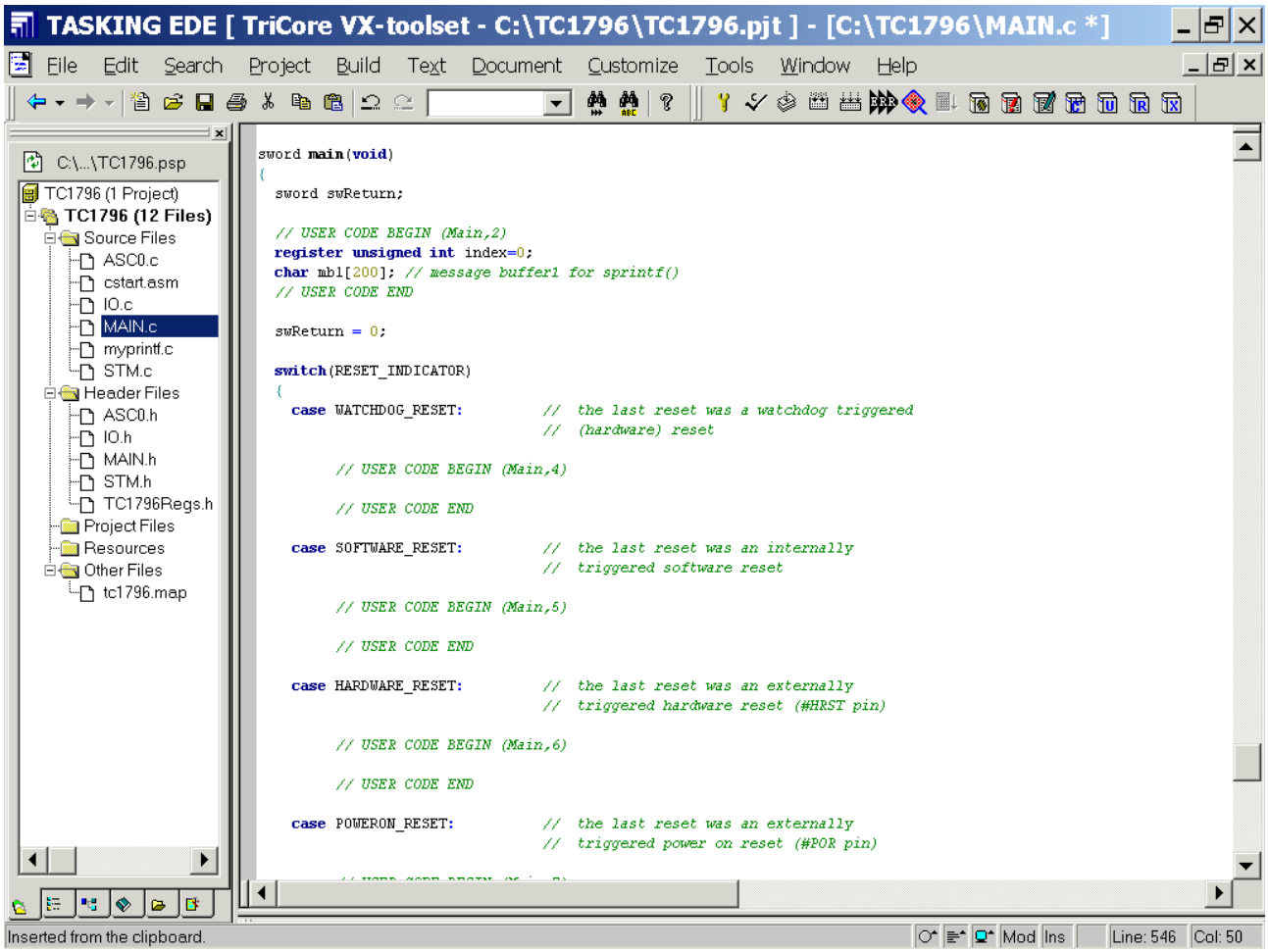
Double click: MAIN.C and insert Global Variables:

```
char cArray[RAM_MAX]; // 1MByte-On-Board-SRAM ( max. 1048576 Bytes)
```



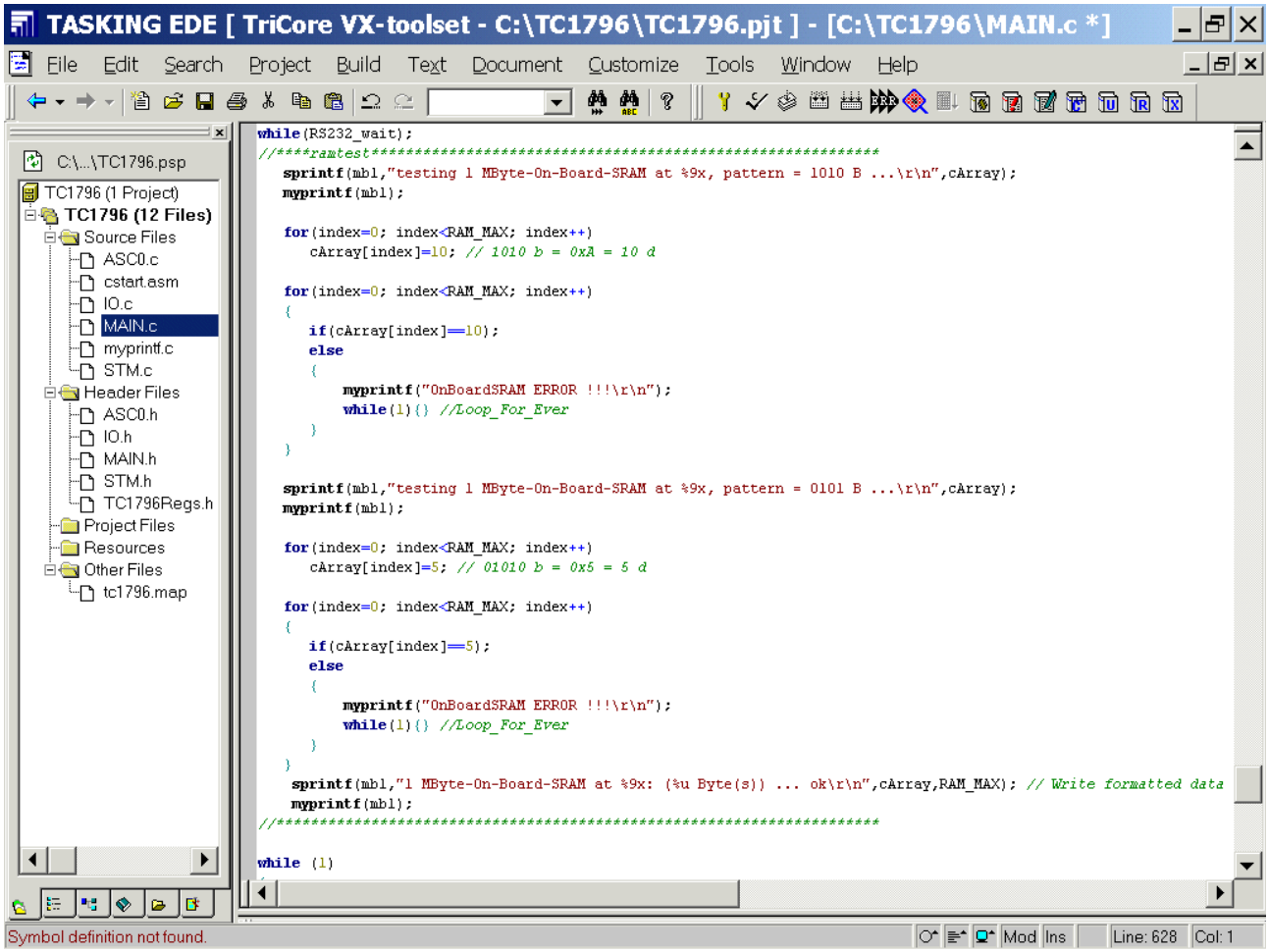
Double click: MAIN.C and insert the following code in the main function:

```
 register unsigned int index=0;
char mb1[200]; // message buffer1 for sprintf()
```



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

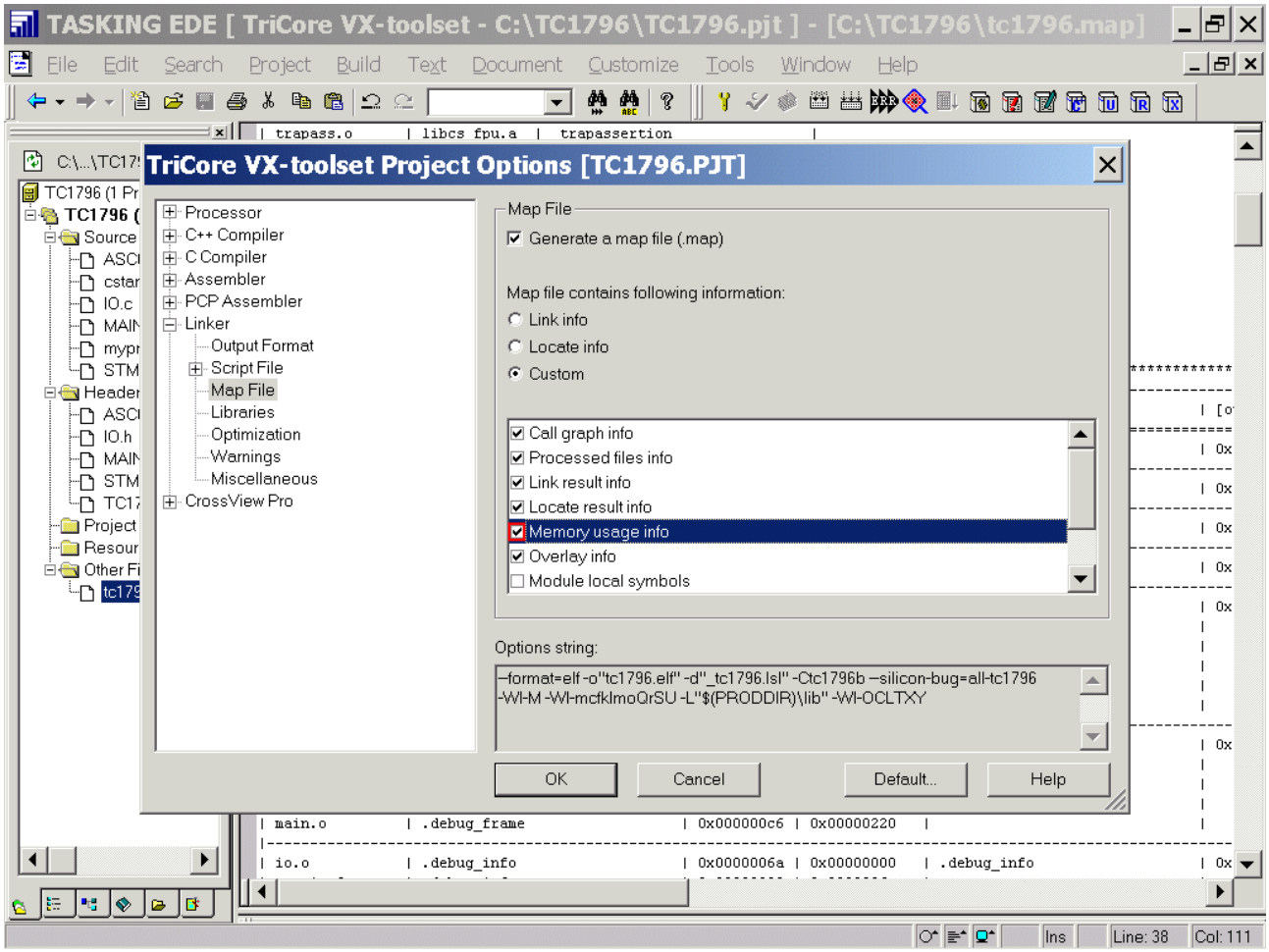
```
 //****ramtest*************************************************************
   sprintf(mb1,"testing 1 MByte-On-Board-SRAM at %9x, pattern = 1010 B
...\r\n",cArray);
   myprintf(mb1);

   for(index=0; index<RAM_MAX; index++)
        cArray[index]=10; // 1010 b = 0xA = 10 d

   for(index=0; index<RAM_MAX; index++)
   {
        if(cArray[index]==10);
        else
        {
              myprintf("OnBoardSRAM ERROR !!!\r\n");
              while(1){} //Loop_For_Ever
        }
   }

   sprintf(mb1,"testing 1 MByte-On-Board-SRAM at %9x, pattern = 0101 B
...\r\n",cArray);
   myprintf(mb1);

   for(index=0; index<RAM_MAX; index++)
        cArray[index]=5; // 01010 b = 0x5 = 5 d

   for(index=0; index<RAM_MAX; index++)
   {
        if(cArray[index]==5);
        else
        {
              myprintf("OnBoardSRAM ERROR !!!\r\n");
              while(1){} //Loop_For_Ever
        }
   }
      sprintf(mb1,"1 MByte-On-Board-SRAM at %9x: (%u Byte(s)) ...
ok\r\n",cArray,RAM_MAX); // Write formatted data to string mb
      myprintf(mb1);
//************************************************************************
```


TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1796.psp
TC1796 (1 Project)
TC1796 (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
TC1796Regs.h
Project Files
Resources
Other Files
tc1796.map
while(RS232_wait);
//****ramtest***
sprintf(mbl,"testing 1 MByte-On-Board-SRAM at %9x, pattern = 1010 B ...\r\n",cArray);
myprintf(mbl);
for(index=0; index<RAM_MAX; index++)
cArray[index]=10; // 1010 b = 0xA = 10 d
for(index=0; index<RAM_MAX; index++)
{
if(cArray[index]==10);
else
{
myprintf("OnBoardSRAM ERROR !!!\r\n");
while(1){} //Loop_For_Ever
}
}
sprintf(mbl,"testing 1 MByte-On-Board-SRAM at %9x, pattern = 0101 B ...\r\n",cArray);
myprintf(mbl);
for(index=0; index<RAM_MAX; index++)
cArray[index]=5; // 01010 b = 0x5 = 5 d
for(index=0; index<RAM_MAX; index++)
{
if(cArray[index]==5);
else
{
myprintf("OnBoardSRAM ERROR !!!\r\n");
while(1){} //Loop_For_Ever
}
}
sprintf(mbl,"1 MByte-On-Board-SRAM at %9x: (%u Byte(s)) ... ok\r\n",cArray,RAM_MAX); // Write formatted data
myprintf(mbl);
//**
while (1)
Symbol definition not found.
Mod Ins Line: 628 Col: 1

Configure Linker – Control:

Project – Project Options

Linker: Map File: click ✓ Memory usage info

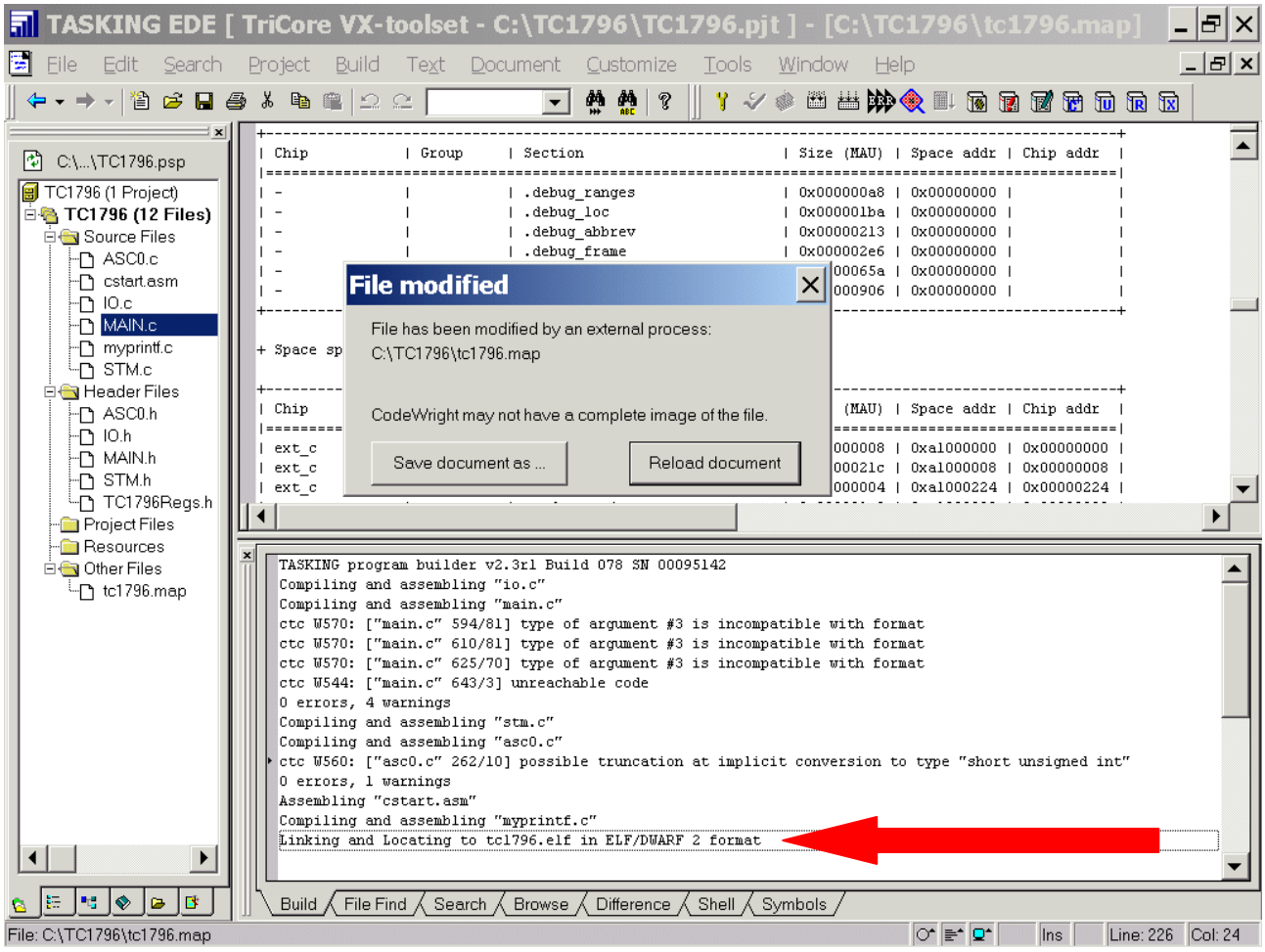


OK

Generate your application program:

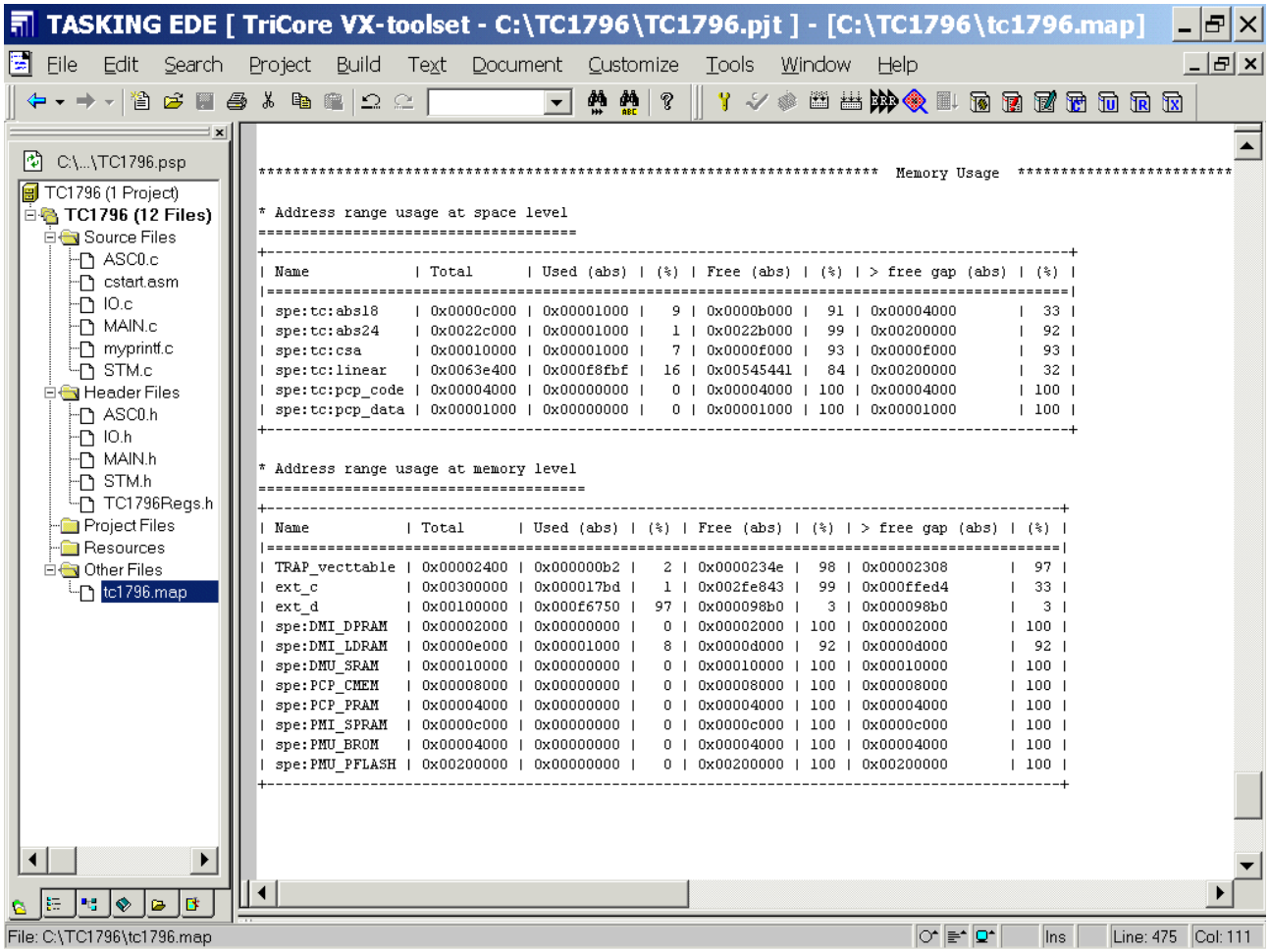
Build
Rebuild

or



Click Reload document

See the Memory Usage:



Now you can close your project and Tasking EDE:

File - Close Project Space
File – Exit

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

Start pls-Debugger

File – Open Workspace

Look in: select C:\TC1796
File name: select tc1796.wsp

Open
Cancel

File – Load Program

Look in: select TC1796
File name: select tc1796.elf

Open

click Program All

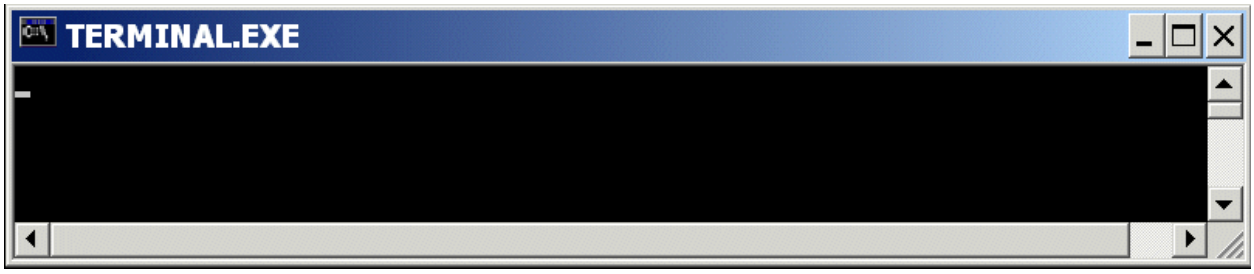
Exit
Exit

File – Close Workspace

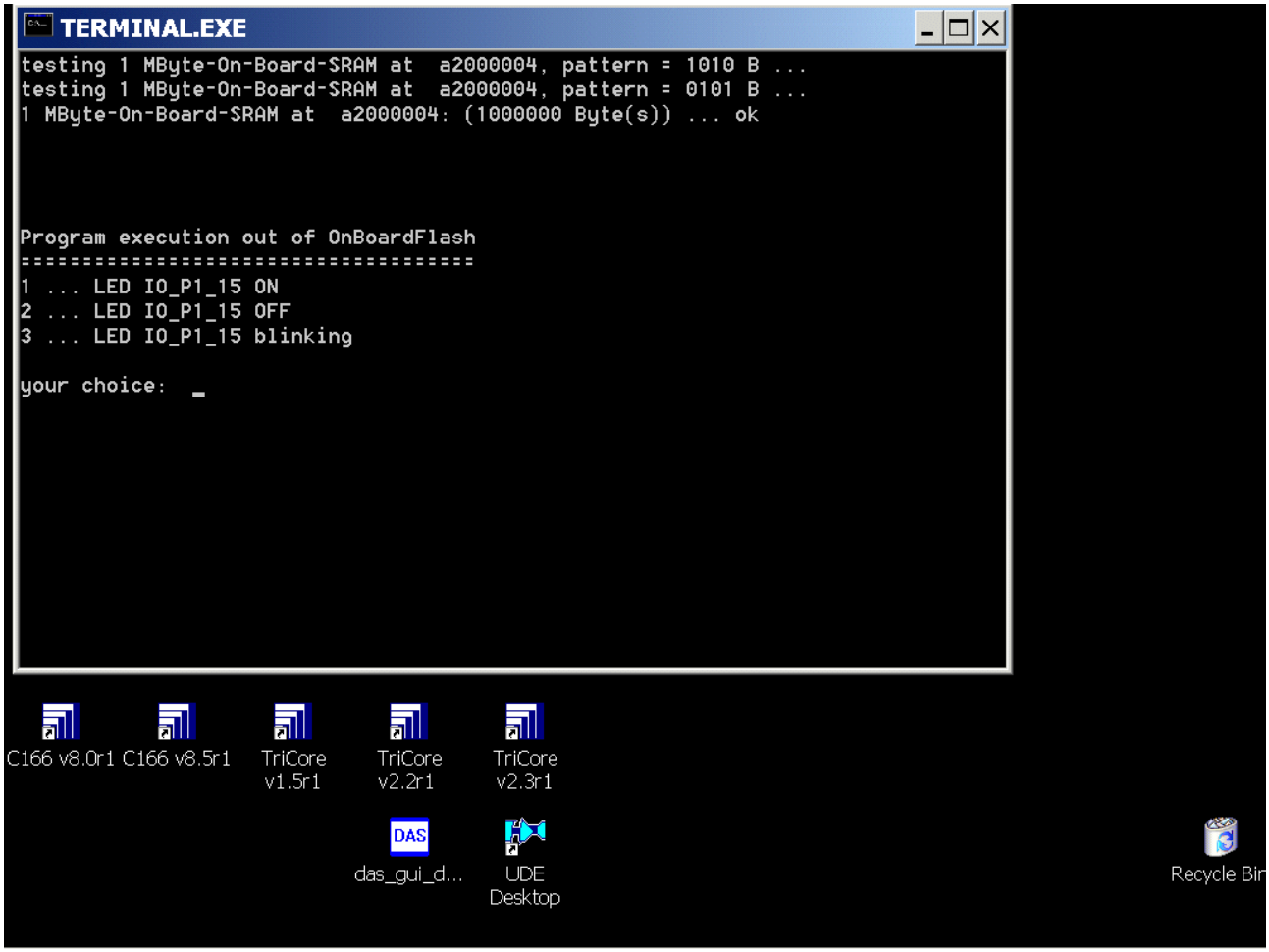
Yes

File – Exit

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

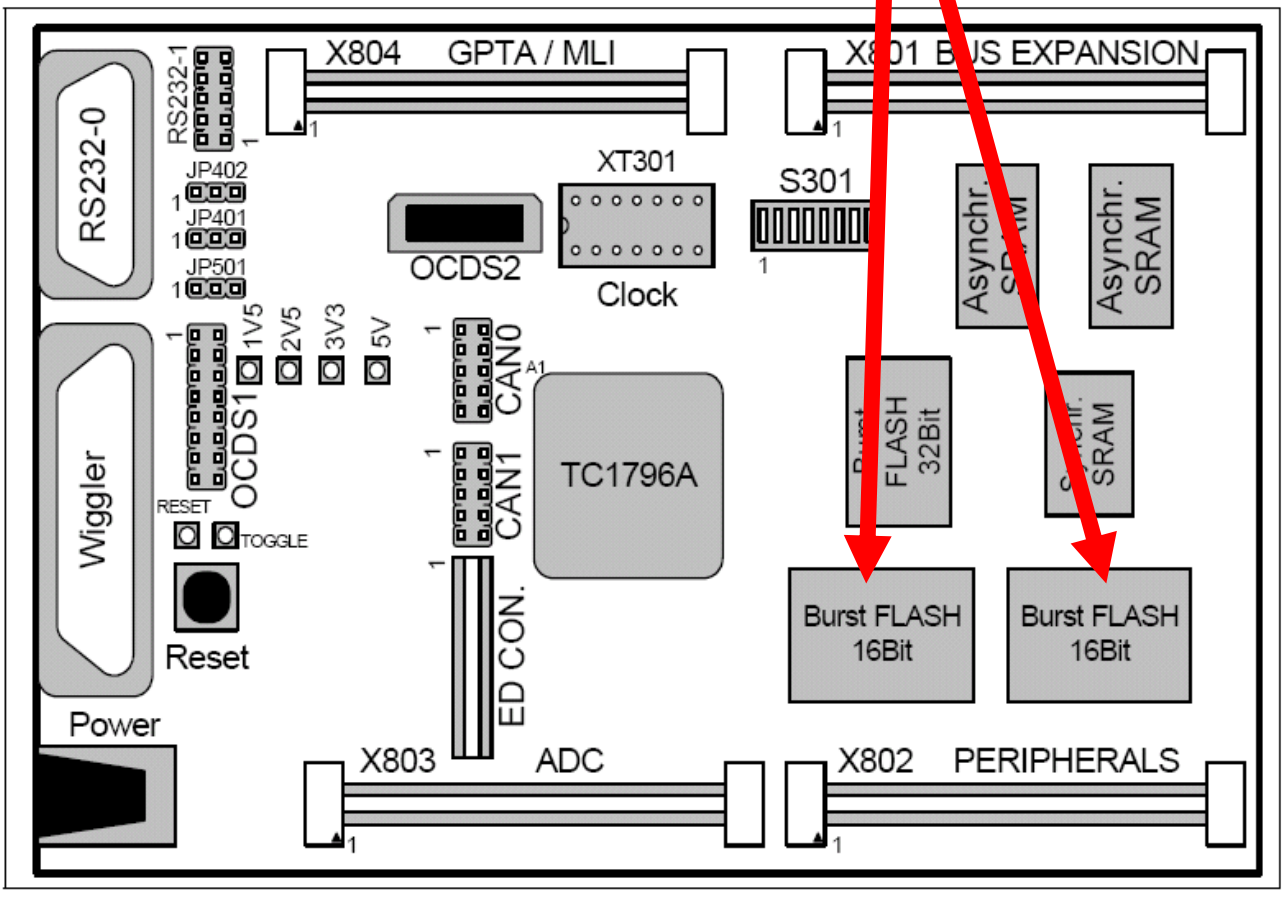


Power-On the Board and see the result:



7.) Using of the TASKING - EDE Development Tools:

Write programs for execution from OnBoardFlash (BURST-Mode)



Start Tasking EDE and open the project:

File – Open Project Space

Look in: select C:\TC1796
File name: select TC1796.psp

Open

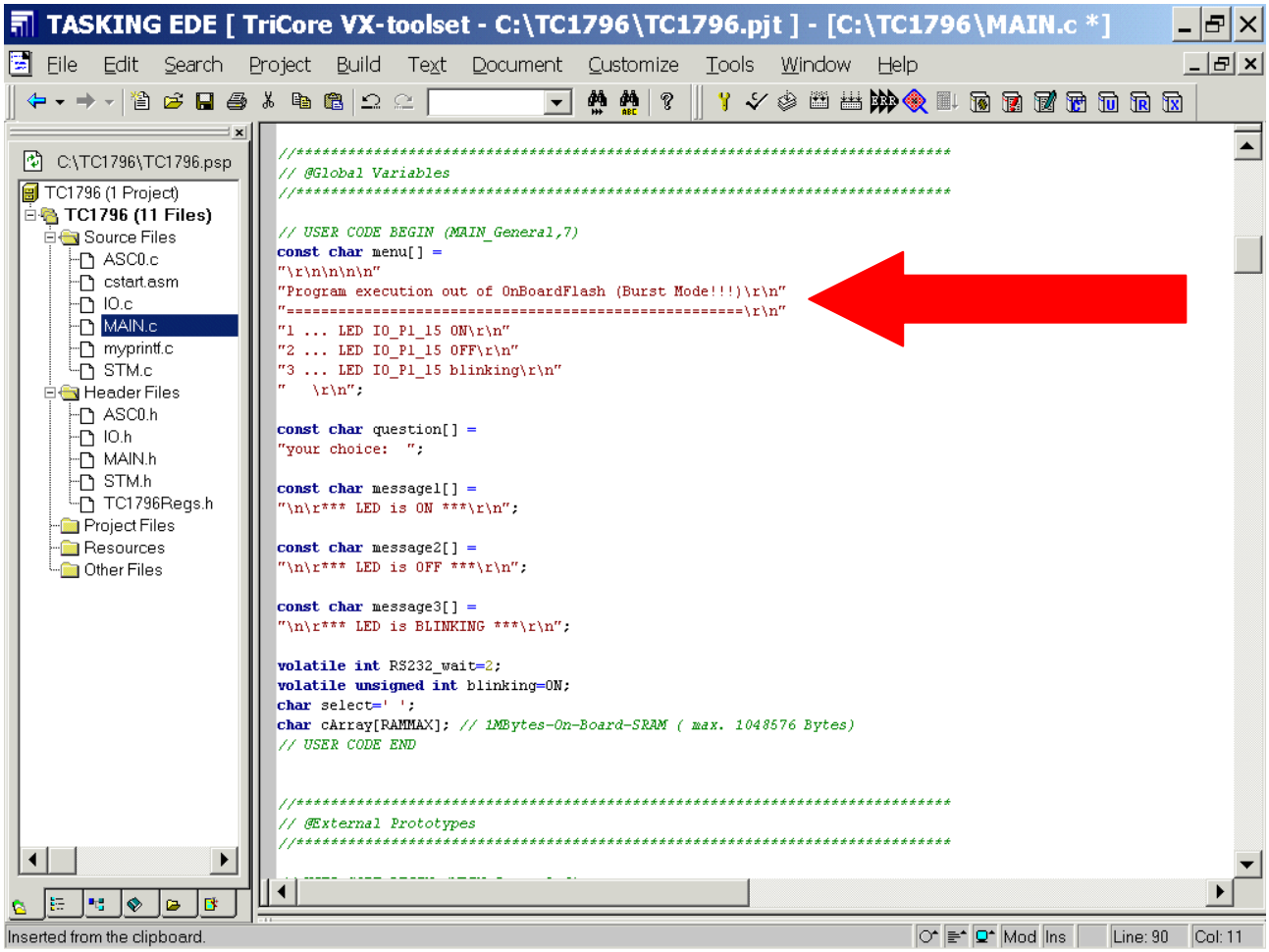
Insert your application specific program:

Double click: Main.c and change Global Variable menu
from

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of OnBoardFlash\r\n"
"=====================================\r\n"
"1 ... LED IO_P1_15 ON\r\n"
"2 ... LED IO_P1_15 OFF\r\n"
"3 ... LED IO_P1_15 blinking\r\n"
"   \r\n";
```

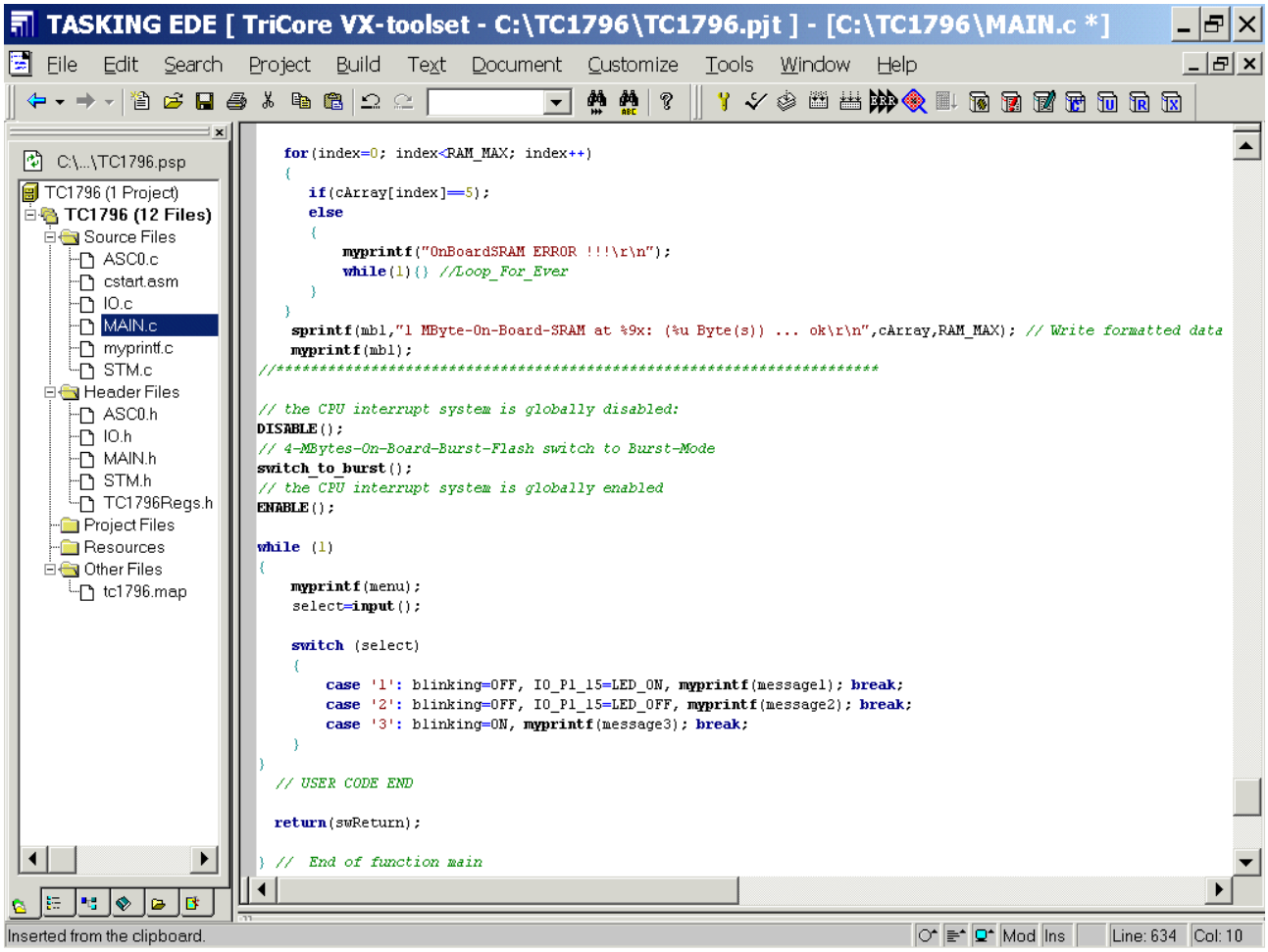
to

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of OnBoardFlash (Burst Mode!!!)\r\n"
"=====================================================\r\n"
"1 ... LED IO_P1_15 ON\r\n"
"2 ... LED IO_P1_15 OFF\r\n"
"3 ... LED IO_P1_15 blinking\r\n"
"   \r\n";
```


TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1796\TC1796.psp
TC1796 (1 Project)
TC1796 (11 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
TC1796Regs.h
Project Files
Resources
Other Files
// USER CODE BEGIN (MAIN_General,7)
const char menu[] =
"\r\n\n\n\n"
"Program execution out of OnBoardFlash (Burst Mode!!!)\r\n"
"==========\r\n"
"1 ... LED IO_P1_15 ON\r\n"
"2 ... LED IO_P1_15 OFF\r\n"
"3 ... LED IO_P1_15 blinking\r\n"
" \r\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=' ';
char cArray[RAMMAX]; // 1MBytes-On-Board-SRAM (max. 1048576 Bytes)
// USER CODE END
// @External Prototypes
Inserted from the clipboard.
Mod Ins Line: 90 Col: 11

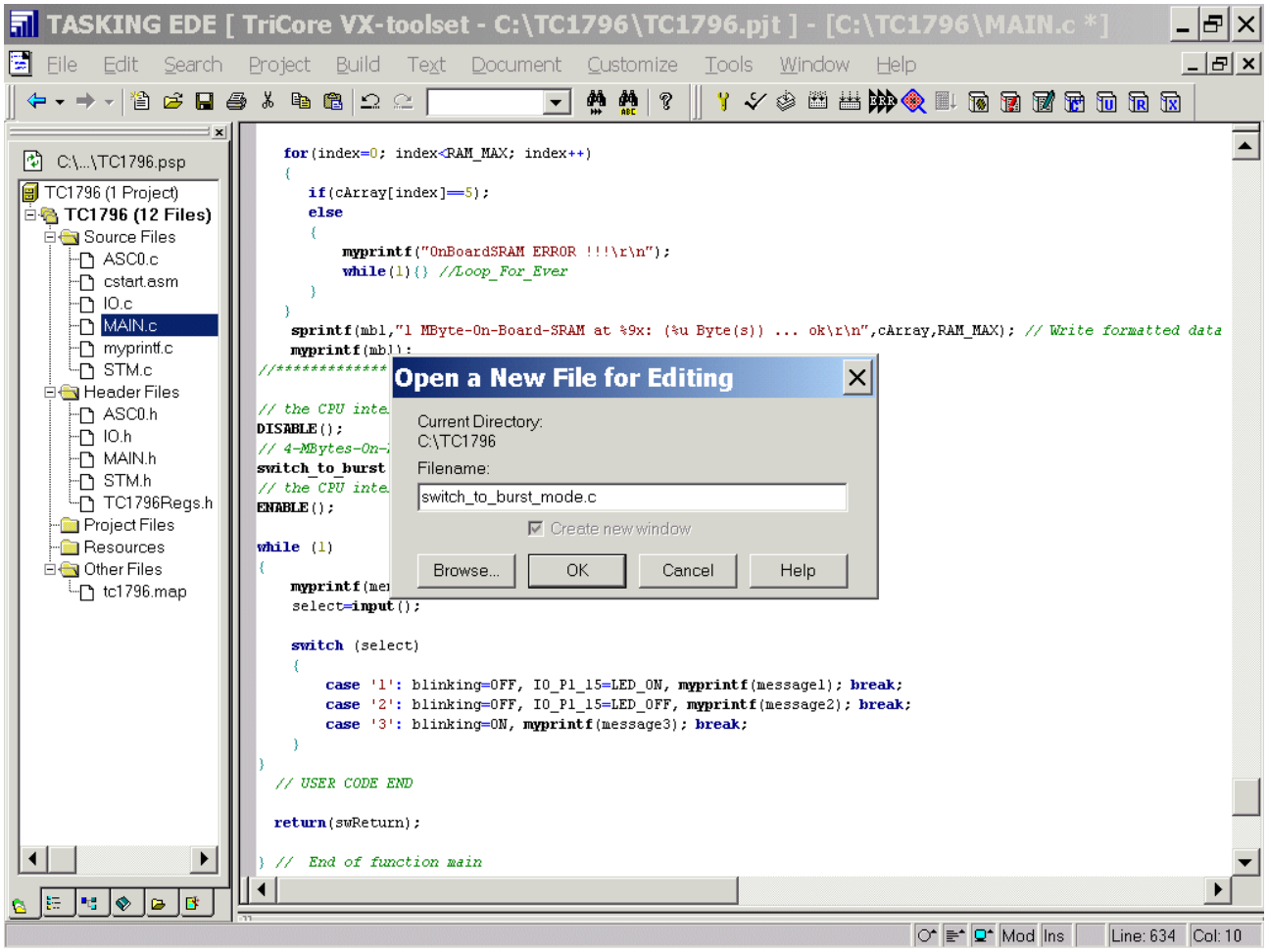
Double click: Main.c: insert application-specific-program:

```
// the CPU interrupt system is globally disabled:
DISABLE();
// 4-MBytes-On-Board-Burst-Flash switch to Burst-Mode
switch_to_burst();
// the CPU interrupt system is globally enabled
ENABLE();
```



File – New

Insert switch_to_burst_mode.c



OK

Insert code:

```
/*****************************************************************************
**                                                                          *
**                                                                          *
**  COPYRIGHT:       (c) 2003 Infineon Technologies                        *
**                                                                          *
**  AUTHOR:          Holger Dienst (AI MC MA TM)                           *
**                                                                          *
**  changed/used by: Wilhelm Brezovits, June 2006                          *
**                                                                          *
**  DESCRIPTION:     activate Burst Mode                                   *
**                                                                          *
**                                                                          *
*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "MAIN.h"
#include "ebu.h"
#include "types.h"
#define SETUP_ADR       0x0555 * 4
#define UNLOCK_ADR      0x02AA * 4
#define AMDSET_UP       0x00AA00AA
#define AMDUNLOCK       0x00550055
#define AMDRESET        0x00F000F0
#define AMDBURST        0x00C000C0
#define AMDBURST_ENABLE 0x00010001

void external_switch_program(void);

// Note:
// German:
// Das Problem ist, daß die Routine die das Flash umschaltet
// (external_switch_program) nicht aus dem Flash laufen kann,
// daher muß diese erst ins RAM kopiert (mittels switch_to_burst)
// und dort ausgeführt werden.
// English:
// The routine which switches the Flash
// (external_switch_program) cannot be executed from the Flash.
// Therefore, this must be copied into the RAM (by switch_to_burst) and
// executed there.

unsigned char mb2[200]; // message buffer2 for sprintf()

void external_switch_program(void)
{
        EBU             *psEBU;
        u_int32 *puiAdr;
        u_int32 uiCS0_Base;

        psEBU = (EBU *)(EBU_BASE);

        uiCS0_Base = psEBU->ADDSEL0;

        uiCS0_Base &= EBUBASE_MSK;

        // enable burst on the flash
        puiAdr = (u_int32 *)(uiCS0_Base + SETUP_ADR);
        *puiAdr = AMDSET_UP;

        puiAdr = (u_int32 *)(uiCS0_Base + UNLOCK_ADR);
        *puiAdr = AMDUNLOCK;

        puiAdr = (u_int32 *)(uiCS0_Base + SETUP_ADR);
        *puiAdr = AMDBURST;

        puiAdr = (u_int32 *)(uiCS0_Base);
        *puiAdr = AMDBURST_ENABLE;

        *puiAdr = AMDRESET;

        // setting BUSCON0
        psEBU->BUSCON0 &= ~EBUAGEN_MSK;
        psEBU->BUSCON0 |= EBUAGEN_BURST0;

        // setting BFCON
        psEBU->BFCON = EBUFETBLEN0_8 | EBUFBBMSEL0 | EBUWAITFUNC0 | EBUEXTCLK_4 | EBUDBA0;

        // get BFCON back for synchronizing
        uiCS0_Base = psEBU->BFCON;

          isync();
```

```
        __dsync();
}

void switch_to_burst(void)
{
       u_int32 *puiSource;
       u_int32 *puiDestination;
       u_int32 *puiDataHeap;
       unsigned int byte_count=(u_int32)&switch_to_burst-(u_int32)&external_switch_program;

       puiDataHeap = malloc((u_int32)&switch_to_burst-(u_int32)&external_switch_program);
       if( puiDataHeap == NULL )
       {
          myprintf("ERROR: Insufficient memory (heap) available !!!\r\n");
          while(1){} //Loop_For_Ever
       }
       else
       {
          puiDestination = puiDataHeap;

          sprintf(mb2,"%u Byte(s) at %9x (heap) allocated\r\n",byte_count,puiDataHeap); //Write
formatted data to string mb2
          myprintf(mb2);
          byte_count=0;

          // first copy all codes to external sram
          myprintf("Copy Program into heap\r\n");
          for(puiSource = (u_int32 *)&external_switch_program;
             puiSource < (u_int32 *)(&switch_to_burst); puiSource++)
          {
             *puiDestination = *puiSource;
             puiDestination++;
             byte_count++;
          }
          sprintf(mb2,"%u*4=%u Byte(s) copied into heap\r\n",byte_count,byte_count*4);
          myprintf(mb2);

          puiDestination = puiDataHeap;

          myprintf("JUMP to heap\r\n");
          // jump to external ram

          // __asm("calli\t%0"::"a"(puiDestination));  // (possibility 1/3) WRONG !!! FAULTY !!!
          // a2 is passed to the instruction template.
          // a2 is part of the lower context and therefore not saved/restored.
          // The compiler assumes that a2 remains constant as it is an input parameter.
          // The technical reasoning for this is that otherwise all input parameters require a
          // backup prior to passing them to the instruction template.
          // The trick now is to tell the compiler that there is a possibility that the input
          // parameter might change because you after all know what the instruction template is
          // doing. One way would be the following:

          // __asm("calli\t%0":"=a"(puiDestination):"0"(puiDestination)); // (possibility 2/3)
OK !!!

          // But another would be to just do the whole thing in C:

          ((void(*)(void))puiDestination)(); // (possibility 3/3) OK !!!

          free(puiDataHeap);
          myprintf("BACK from heap, Burst-Mode activated\r\n");
       }
}
```

Note:

// jump to external ram
__asm("calli\t%0"::"a"(puiDestination));

Syntax see: TriCore User's Manual, CHAPTER 3, TriCore C Language (next page):

TriCore User's Manual

File Edit Bookmark Chapters Options Help

Contents Index Back Print

CHAPTER 3: TriCore C Language

3.6 USING ASSEMBLY IN THE C SOURCE: __asm()

With the `__asm()` keyword you can use assembly instructions in the C source and pass C variables as operands to the assembly code. Be aware that C modules that contain assembly are not portable and harder to compile in other environments.

The compiler does not interpret assembly blocks but passes the assembly code to the assembly source file. Possible errors can only be detected by the assembler.

General syntax of the `__asm` keyword

```
__asm( "instruction_template"
       [ : output_param_list
       [ : input_param_list
       [ : register_save_list]]] );
```

instruction_template	Assembly instructions that may contain parameters from the input list or output list in the form: **%***parm_nr*
%*parm_nr*[**.***regnum*]	Parameter number in the range 0 .. 9. With the optional **.***regnum* you can access an individual register from a register pair or register quad. For example, with register pair `d0/d1`, `.0` selects register `d0`.
output_param_list	[[**"=**[**&**]*constraint_char***"(***C_expression***)**],...]
input_param_list	[[**"***constraint_char***"(***C_expression***)**],...]
&	Says that an output operand is written to before the inputs are read, so this output must not be the same register as any input.
constraint_char	Constraint character: the type of register to be used for the *C_expression*. (see table 3-6)
C_expression	Any C expression. For output parameters it must be an *lvalue*, that is, something that is legal to have on the left side of an assignment.
register_save_list	[[**"***register_name***"**],...]
register_name	Name of the register you want to reserve.

Typical example: multiplying two C variables using assembly

```
int a,b,result;

void main( void )
{
    __asm("mul\t%0,%1,%2" : "=d"(result) : "d"(a), "d"(b) );
}
```

generated code:

```
ld.w    d15,a
ld.w    d0,b
mul     d15,d15,d0
st.w    result,d15
```

`%0` corresponds to the first C variable, `%1` corresponds to the second and so on. The escape sequence `\t` generates a tab.

Specifying registers for C variables

With a *constraint character* you specify the register *type* for a parameter. In the example above, the `d` is used to force the use of data registers for the parameters `a`, `b` and `result`.

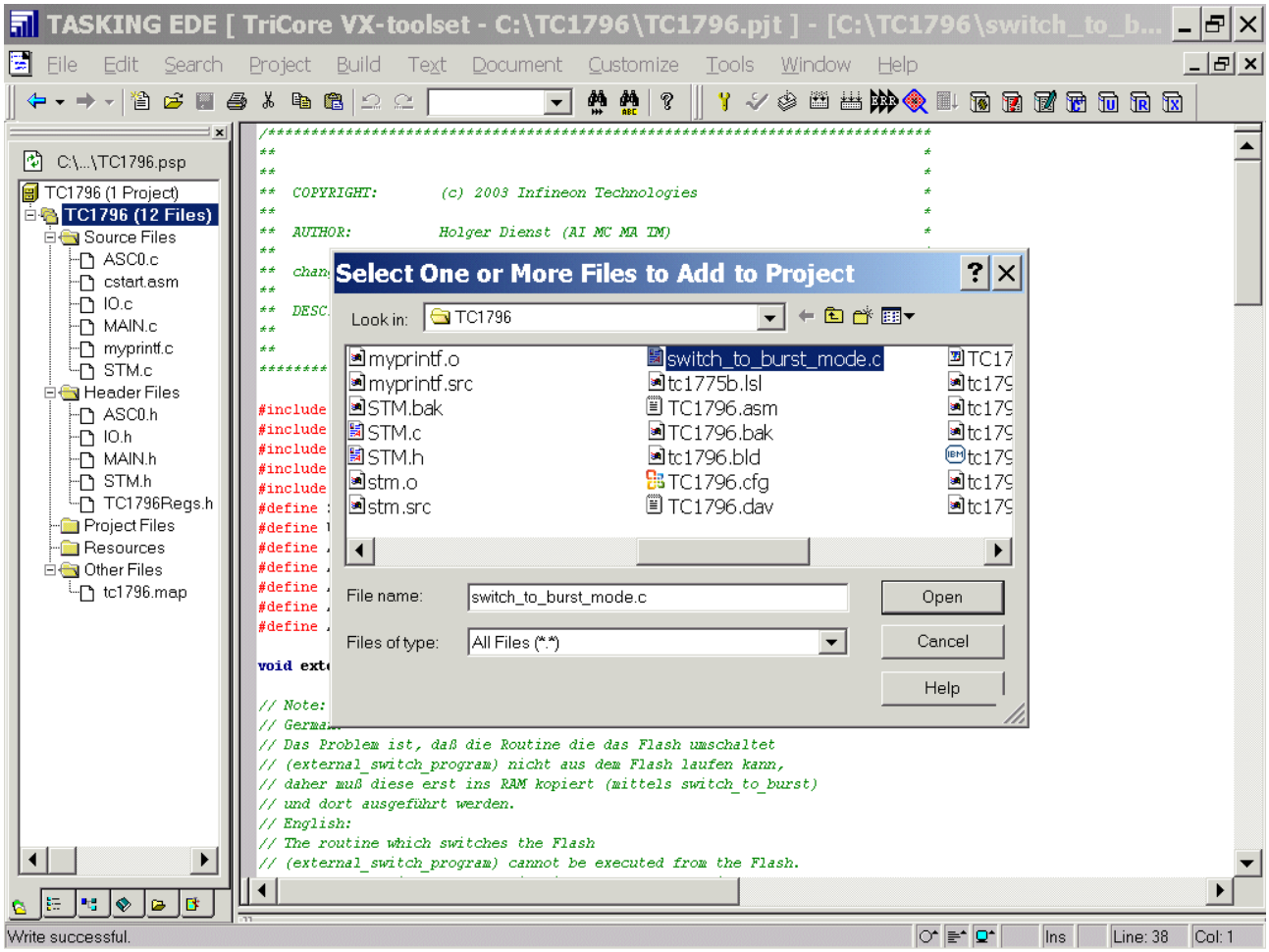
You can reserve the registers that are used in the assembly instructions, either in the parameter lists or in the reserved register list (*register_save_list*). The compiler takes account of these lists, so no unnecessary register saves and restores are placed around the inline assembly instructions.

Constraint character	Type	Operand	Remark
a	Address register	a0 .. a15	
d	Data register	d0 .. d15	
e	Data register pair	e0,e2,..,e14	e0 = pair d0/d1, e2 = d2/d3, ...
m	Memory	*variable*	Stack or memory operand
number	Type of operand it is associated with	same as %*number*	Indicates that %*number* and *number* are the same register.

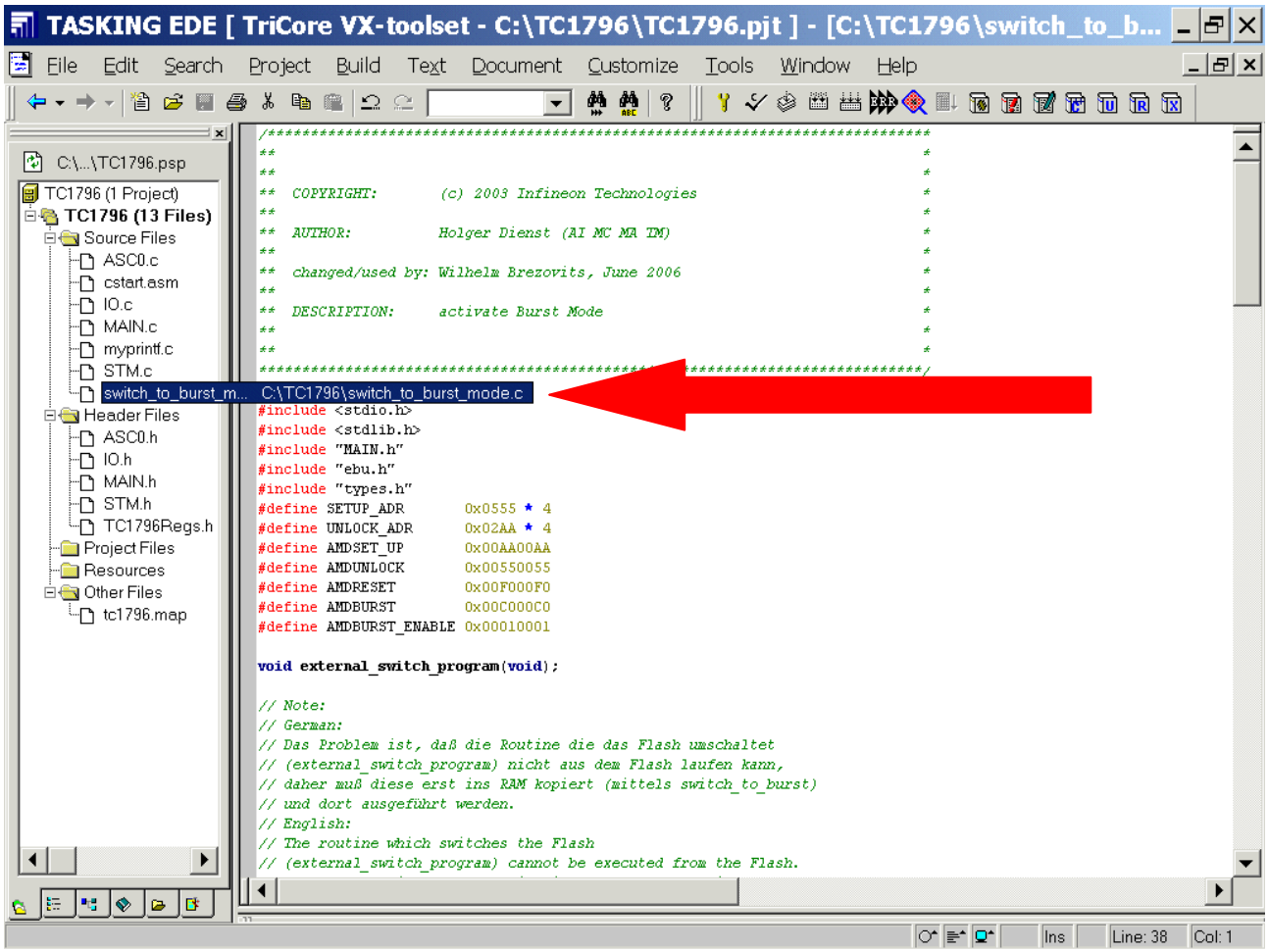
Table 3-6: Available input/output operand constraints

File – Save

(Project Window File View) – TC1796 (Files) – right mouse button click – Add Existing Files – Browse

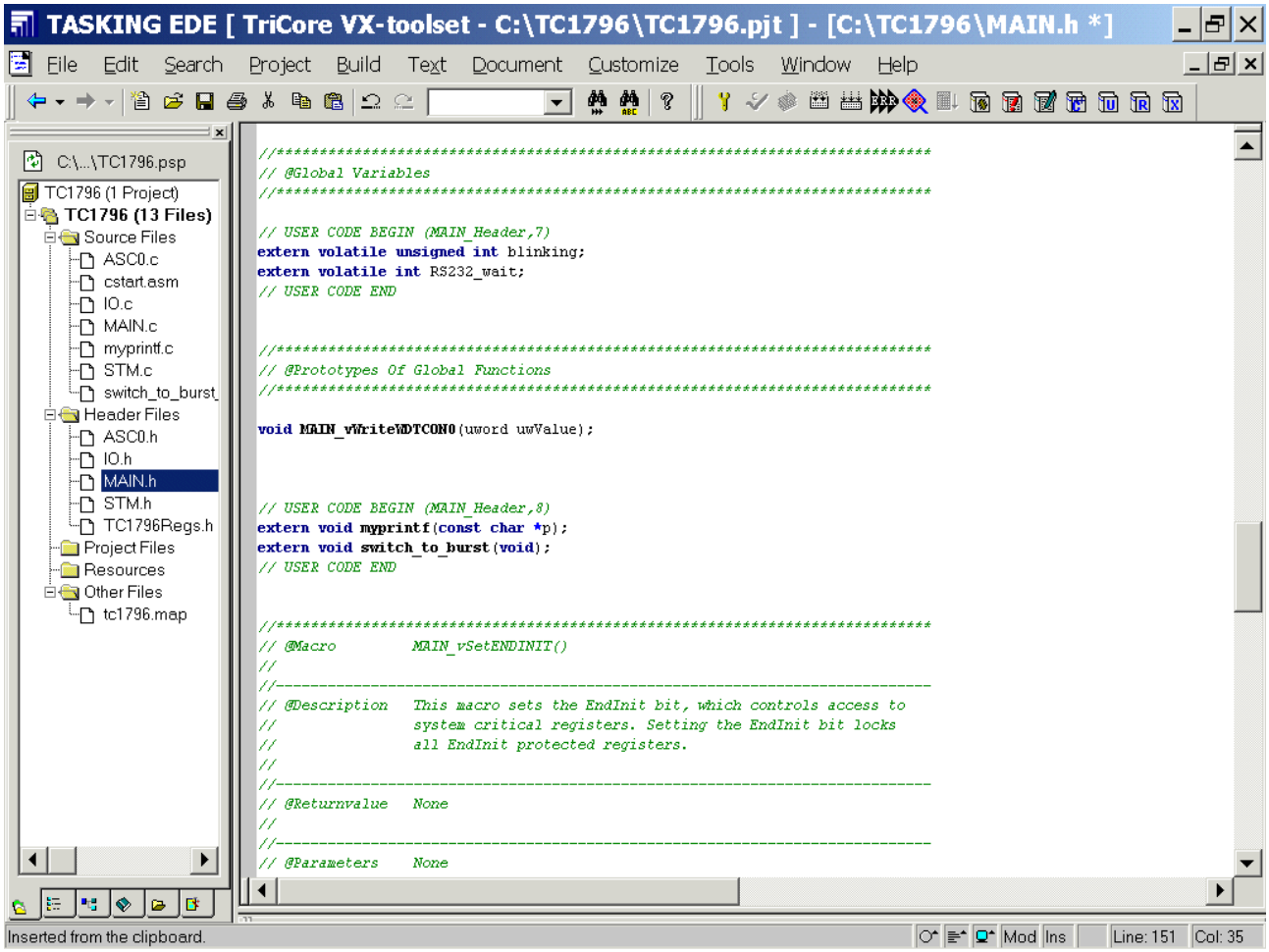


Select switch_to_burst_mode.c
Open
OK


TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\switch_to_b...
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1796.psp
TC1796 (1 Project)
TC1796 (13 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
STM.c
switch_to_burst_m...
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
TC1796Regs.h
Project Files
Resources
Other Files
tc1796.map
C:\TC1796\switch_to_burst_mode.c
/***
** *
** *
** COPYRIGHT: (c) 2003 Infineon Technologies *
** *
** AUTHOR: Holger Dienst (AI MC MA TM) *
** *
** changed/used by: Wilhelm Brezovits, June 2006 *
** *
** DESCRIPTION: activate Burst Mode *
** *
** *
***/
#include <stdio.h>
#include <stdlib.h>
#include "MAIN.h"
#include "ebu.h"
#include "types.h"
#define SETUP_ADR 0x0555 * 4
#define UNLOCK_ADR 0x02AA * 4
#define AMDSET_UP 0x00AA00AA
#define AMDUNLOCK 0x00550055
#define AMDRESET 0x00F000F0
#define AMDBURST 0x00C000C0
#define AMDBURST_ENABLE 0x00010001
void external_switch_program(void);
// Note:
// German:
// Das Problem ist, daß die Routine die das Flash umschaltet
// (external_switch_program) nicht aus dem Flash laufen kann,
// daher muß diese erst ins RAM kopiert (mittels switch_to_burst)
// und dort ausgeführt werden.
// English:
// The routine which switches the Flash
// (external_switch_program) cannot be executed from the Flash.
Ins
Line: 38
Col: 1

Double click: Main.h and insert Prototype of Global Function:

```
extern void switch to burst(void);
```



File – New

Insert types.h

OK

Insert code:

```
/*****************************************************************************
*                                                                            *
*  FILE        :  TYPES.H                                                    *
*                                                                            *
*  DESCRIPTION :  Setting names for types                                    *
*                                                                            *
*  COPYRIGHT   :  (c) 1999 Infineon Technologies                             *
*                                                                            *
*  AUTHOR      :  Holger Dienst (AI MC AE)                                   *
*                                                                            *
*****************************************************************************/

#ifndef _TYPES_H
#define _TYPES_H

typedef unsigned int u_int32;
typedef int int32;
typedef unsigned short u_int16;
typedef short int16;
typedef unsigned char u_int8;
typedef char int8;

#endif  /*  TYPES H */
```

File – Save

File – New

Insert EBU.h

OK

Insert code:

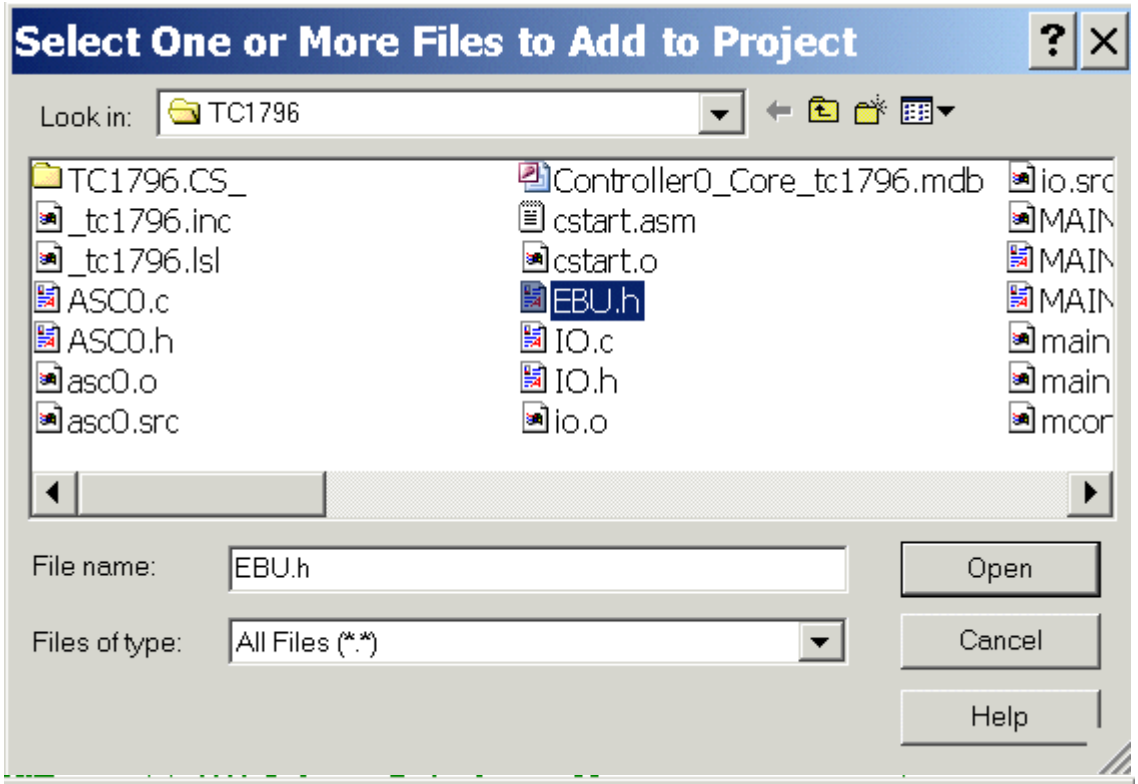
```
/******************************************************************************
*                                                                             *
*  FILE        :  EBU.H                                                       *
*                                                                             *
*  DESCRIPTION :  Structur and Bitsettings for EBU TC1796                     *
*                                                                             *
*  COPYRIGHT   :  (c) 2003 Infineon Technologies AG                           *
*                                                                             *
*  AUTHOR      :  Holger Dienst (AI MC MA TM)                                    *
*                                                                             *
*  VERSION     :  1.00                                                        *
*                                                                             *
*  CHANGES     :                                                              *
*                                                                             *
******************************************************************************/

#ifndef _EBU_H
#define _EBU_H

#define EBU_BASE                (0xF8000000)

/* Access Mode:
R  - Read-only register.
32 - Only 32-bit word accesses are permitted to that register/address range.
E  - Endinit protected register/address.
PW - Password protected register/address.
For more details refer to specification.
*/

typedef struct ebu
{
        volatile unsigned int CLC;           /* Clock Control Register (E) (32) */
        volatile unsigned int RESERVED0[1];  /* Reserved */
        volatile unsigned int ID;            /* Identification Register (32) */
        volatile unsigned int RESERVED1[1];  /* Reserved */
        volatile unsigned int CON;           /* Global Control Register (32) */
        volatile unsigned int RESERVED2[3];  /* Reserved */
        volatile unsigned int BFCON;         /* Burst Flash Control Register (32) */
        volatile unsigned int RESERVED3[23]; /* Reserved */
        volatile unsigned int ADDSEL0;       /* Address Select Register 0 (32) */
        volatile unsigned int RESERVED4[1];  /* Reserved */
        volatile unsigned int ADDSEL1;       /* Address Select Register 1 (32) */
        volatile unsigned int RESERVED5[1];  /* Reserved */
        volatile unsigned int ADDSEL2;       /* Address Select Register 2 (32) */
        volatile unsigned int RESERVED6[1];  /* Reserved */
        volatile unsigned int ADDSEL3;       /* Address Select Register 3 (32) */
        volatile unsigned int RESERVED7[9];  /* Reserved */
        volatile unsigned int BUSCON0;       /* Bus Configuration Register 0 (32) */
        volatile unsigned int RESERVED8[1];  /* Reserved */
        volatile unsigned int BUSCON1;       /* Bus Configuration Register 1 (32) */
        volatile unsigned int RESERVED9[1];  /* Reserved */
        volatile unsigned int BUSCON2;       /* Bus Configuration Register 2 (32) */
        volatile unsigned int RESERVED10[1]; /* Reserved */
        volatile unsigned int BUSCON3;       /* Bus Configuration Register 3 (32) */
        volatile unsigned int RESERVED11[9]; /* Reserved */
        volatile unsigned int BUSAP0;        /* Bus access parameter Register 0 (32) */
        volatile unsigned int RESERVED12[1]; /* Reserved */
        volatile unsigned int BUSAP1;        /* Bus access parameter Register 1 (32) */
        volatile unsigned int RESERVED13[1]; /* Reserved */
        volatile unsigned int BUSAP2;        /* Bus access parameter Register 2 (32) */
        volatile unsigned int RESERVED14[1]; /* Reserved */
        volatile unsigned int BUSAP3;        /* Bus access parameter Register 3 (32) */
        volatile unsigned int RESERVED15[17];/* Reserved */
        volatile unsigned int EMUAS;         /* Emulator Memory Address Select Register (32) */
        volatile unsigned int RESERVED16[1]; /* Reserved */
        volatile unsigned int EMUBC;         /* Emulator Memory Bus Configuration Register (32) */
        volatile unsigned int RESERVED17[1]; /* Reserved */
        volatile unsigned int EMUBAP;        /* Emulator Memory access parameter (32) */
        volatile unsigned int RESERVED18[1]; /* Reserved */
        volatile unsigned int EMUOVL;        /* Overlay memory chip-select generation (32) */
        volatile unsigned int RESERVED19[5]; /* Reserved */
```

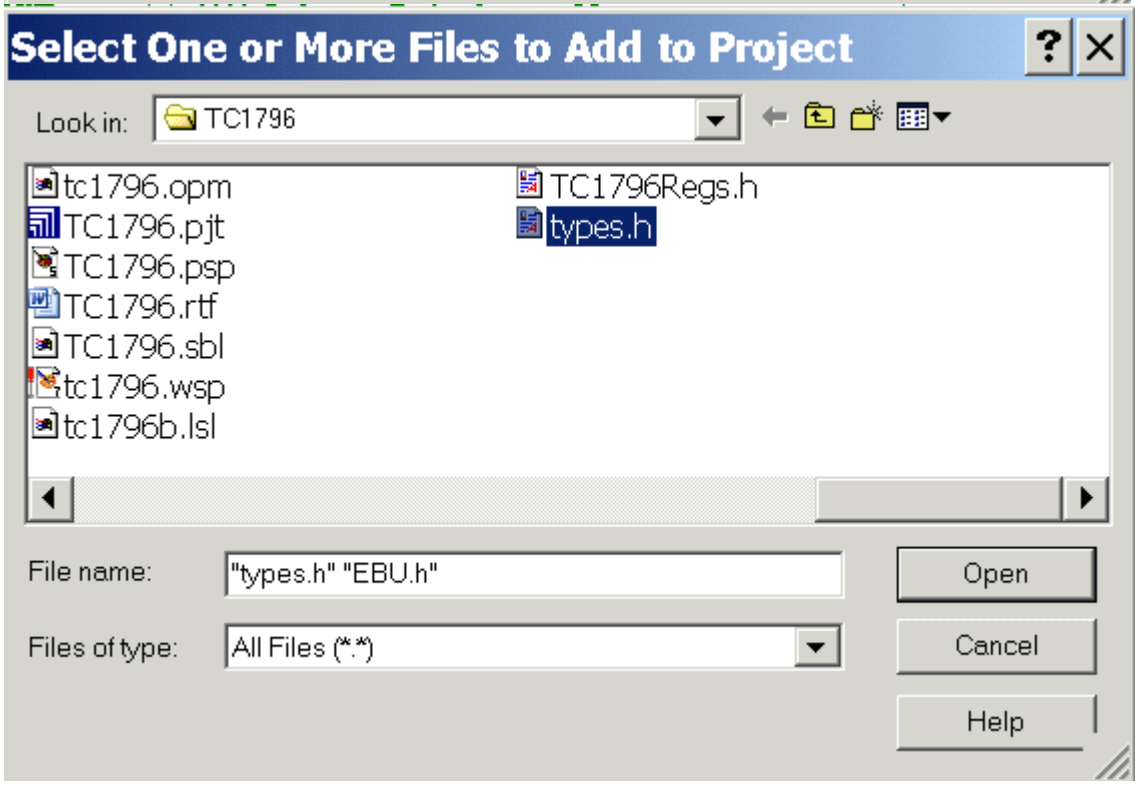
```
        volatile unsigned int USERCON;       /* Test/Control Configuration Register (32) */
} EBU;

/* Global Control Register */
#define EBUEXTRECON         0x00000002      /* EBU External Address Extension Reconfiguration Control
*/
#define EBUEXTSVM           0x00000004      /* EBU External Supervisor Mode Access Control */
#define EBUEXTACC           0x00000008      /* EBU External Access to FPI Control */
#define EBUEXTLOCK          0x00000010      /* EBU External Bus Lock Control */
#define EBUARBSYNC          0x00000020      /* EBU Arbitration Inputs Evaluation Control */
#define EBUARBMODE_MSK      0x000000C0      /* EBU Arbitration Strategy */
#define EBUARBMODE_EBUDIS   0x00000000      /* EBU is Disabled */
#define EBUARBMODE_EXTM     0x00000040      /* EBU is External Master */
#define EBUARBMODE_EXTS     0x00000080      /* EBU is External Slave */
#define EBUARBMODE_NOARB    0x000000C0      /* EBU Arbitration is Disabled */
#define EBUTIMOUTC_MSK      0x0000FF00      /* EBU Time Out Control */
#define EBUGLOBAL_CS_MSK    0x00FF0000      /* EBU chip select global mask */
#define EBUGLOBAL_CS0       0x00010000      /* EBU chip select global 0 */
#define EBUGLOBAL_CS1       0x00020000      /* EBU chip select global 1 */
#define EBUGLOBAL_CS2       0x00040000      /* EBU chip select global 2 */
#define EBUGLOBAL_CS3       0x00080000      /* EBU chip select global 3 */
#define EBUCS0FAM           0x08000000      /* EBU CS0 Fills Address Map */
#define EBUEMUFAM           0x10000000      /* EBU CSemu Fills Address Map */

/* Burst Flash Control Register */
#define EBUFETBLEN0_MSK 0x0000000F     /* Fetch burst length Type 0 */
#define EBUFETBLEN0_1   0x00000000      /* Fetch burst length 1 data access Type 0 */
#define EBUFETBLEN0_2   0x00000001      /* Fetch burst length 2 data access Type 0 */
#define EBUFETBLEN0_4   0x00000002      /* Fetch burst length 4 data access Type 0 */
#define EBUFETBLEN0_8   0x00000003      /* Fetch burst length 8 data access Type 0 */
#define EBUFBBMSEL0     0x00000010      /* Flash burst buffer mode select Type 0 */
#define EBUWAITFUNC0    0x00000020      /* Operation of /WAIT input Type 0 */
#define EBUEXTCLK_MSK   0x000000C0      /* BFCLK external clock */
#define EBUEXTCLK_1     0x00000000      /* BFCLK = LMBCLK */
#define EBUEXTCLK_2     0x00000040      /* BFCLK = LMBCLK/2 */
#define EBUEXTCLK_3     0x00000080      /* BFCLK = LMBCLK/3 */
#define EBUEXTCLK_4     0x000000C0      /* BFCLK = LMBCLK/4 */
#define EBUBFCMSEL      0x00000100      /* BFCLK only present during burst access */
#define EBUEBSE0        0x00000200      /* ADV and BAA not 1/2LMBCLK delayed Type 0 */
#define EBUDBA0         0x00000400      /* Disable Burst Address Wrapping Type 0 */
#define EBUFDBKEN       0x00000800      /* Burst Clock Feedback Enable */
#define EBUDTALTNCY     0x0000F000      /* Latency Cycle Control */
#define EBUFETBLEN1_MSK 0x000F0000     /* Fetch burst length Type 1 */
#define EBUFETBLEN1_1   0x00000000      /* Fetch burst length 1 data access Type 1 */
#define EBUFETBLEN1_2   0x00010000      /* Fetch burst length 2 data access Type 1 */
#define EBUFETBLEN1_4   0x00020000      /* Fetch burst length 4 data access Type 1 */
#define EBUFETBLEN1_8   0x00030000      /* Fetch burst length 8 data access Type 1 */
#define EBUFBBMSEL1     0x00100000      /* Flash burst buffer mode select Type 1 */
#define EBUWAITFUNC1    0x00200000      /* Operation of /WAIT input Type 1 */
#define EBUEBSE1        0x02000000      /* ADV and BAA not 1/2LMBCLK delayed Type 1 */
#define EBUDBA1         0x04000000      /* Disable Burst Address Wrapping Type 1 */

/* Address Select Register */
#define EBUREGENAB          0x00000001      /* Memory Region Enable Control */
#define EBUALTENAB          0x00000002      /* Altseg is always compared with LMB address */
#define EBUMASK_MSK         0x000000F0      /* Memory Region Address Mask */
#define EBUALTSEG_MSK       0x00000F00      /* Memory Region Alternate Segment */
#define EBUBASE_MSK         0xFFFFF000      /* Memory Region Base Address */

/* Bus Configuration Register */
#define EBUCMULTMAP         0x0000007F      /* Multiplier Map */
#define EBUCMULTMAP_ADDRC   0x00000001      /* Multiplier Map addrc */
#define EBUCMULTMAP_AHOLDC 0x00000002      /* Multiplier Map aholdc */
#define EBUCMULTMAP_CMDDELAY  0x00000004   /* Multiplier Map cmddelay */
#define EBUCMULTMAP_BURSTC 0x00000008      /* Multiplier Map burstc */
#define EBUCMULTMAP_DATAC   0x00000010      /* Multiplier Map datac */
#define EBUCMULTMAP_RDRECOVC  0x00000020   /* Multiplier Map rdrecovc */
#define EBUCMULTMAP_WRRECOVC  0x00000040   /* Multiplier Map wrrecovc */
#define EBUWEAK_PREFETCH    0x00000100      /* Code prefetch can be aborted by data access */
#define EBUAALIGN           0x00000200      /* Address Alignment */
#define EBUCTYPE_MSK        0x00000C00      /* Cycle Typ */
#define EBUCTYPE_NOMUX      0x00000000      /* Cycle Typ is non-multiplexed */
#define EBUCMULT_MSK        0x0000E000      /* Cycle Multiplier Control */
#define EBUCMULT_CM1        0x00000000      /* Cycle Multiplier 1 */
#define EBUCMULT_CM4        0x00002000      /* Cycle Multiplier 4 */
#define EBUCMULT_CM8        0x00004000      /* Cycle Multiplier 8 */
#define EBUCMULT_CM16       0x00006000      /* Cycle Multiplier 16 */
#define EBUCMULT_CM32       0x00008000      /* Cycle Multiplier 32 */
#define EBUENDIAN           0x00010000      /* Big Endian mode */
#define EBUDLOAD            0x00020000      /* Data access always fed from external bus */
#define EBUPREFETCH         0x00040000      /* Code access always uses prefetch buffer */
#define EBUWAITINV          0x00080000      /* #WAIT Input is active High */
#define EBUBCGEN_MSK        0x00300000      /* Byte Control Signal Timing Mode */
#define EBUBCGEN_CSM        0x00000000      /* Chipselect Mode */
#define EBUBCGEN_CM         0x00100000      /* Control Mode */
```

```
#define EBUBCGEN_WEM        0x00200000      /* Write Enable Mode */
#define EBUPORTW_MSK        0x00C00000      /* External Device Data Width Control */
#define EBUPORTW_16         0x00400000      /* 16 Bit */
#define EBUPORTW_32         0x00800000      /* 32 Bit */
#define EBUWAIT_MSK         0x03000000      /* External Waitstate Control */
#define EBUWAIT_DIS         0x00000000      /* External Waitstate Control OFF */
#define EBUWAIT_ASYN        0x01000000      /* Asynchronous Input at #WAIT */
#define EBUWAIT_SYN         0x02000000      /* (Early for Burst Flash Devices) Synchronous Input at
#WAIT  */
#define EBUWAIT_NAND        0x03000000      /* NAND Flash Mode (not valid for Burst Flash Devices) */
#define EBUXCMDDELAY_OFF    0x0000000       /* External Command Delay Control OFF */
#define EBUAGEN_MSK         0x70000000      /* Address Generation Control */
#define EBUAGEN_DEMUX       0x00000000      /* Demultiplex address */
#define EBUAGEN_BURST0      0x20000000      /* Burst Flash Type 0 access */
#define EBUAGEN_BURST1      0x50000000      /* Burst Flash Type 1 access */
#define EBUWRITE            0x80000000      /* Write Protection */

/* Bus Access Parameter Register */
#define EBUDTACS_MSK        0x0000000F      /* Recovery cycles between different regions */
#define EBUDTARDWR_MSK      0x000000F0      /* Recovery cycles between read and write accesses */
#define EBUWRRECOVC_MSK     0x00000700      /* Recovery Cycles after write accesses */
#define EBURDRECOVC_MSK     0x00003800      /* Recovery Cycles after read accesses */
#define EBUDATAC_MSK        0x0000C000      /* Data hold Cycles for write accesses */
#define EBUBURSTC_MSK       0x00070000      /* Data Cycles during burst accesses */
#define EBUWAITWRC_MSK      0x00380000      /* Programmed Waitstates for Write Access */
#define EBUWAITRDC_MSK      0x01C00000      /* Programmed Waitstates for Read Access */
#define EBUCMDDELAY_MSK     0x0E000000      /* Programmed command delay cycles */
#define EBUHOLDC_MSK        0x30000000      /* Address Hold Cycles for multiplexes accesses */
#define EBUADDRC_MSK        0xC0000000      /* Address Cycles */

/* Emulator Memory Control Register */
#define EBUOVL0             0x00000001      /* Overlay Memory Control for Region 0 */
#define EBUOVL1             0x00000002      /* Overlay Memory Control for Region 1 */
#define EBUOVL2             0x00000004      /* Overlay Memory Control for Region 2 */
#define EBUOVL3             0x00000008      /* Overlay Memory Control for Region 3 */

/* Test/Control Configuration Register */
#define EBUDIP              0x00000001      /* Disable Internal Pipelining */

/* External Access Configuration Register */
#define EBUAEXT0_MSK        0x000003FF      /* Address Extension Bits A[31..22] for External Access
with A[23..22] = 00 */
#define EBUAEXT2_MSK        0x000FFC00      /* Address Extension Bits A[31..22] for External Access
with A[23..22] = 10 */
#define EBUAEXT3_MSK        0x3FF00000      /* Address Extension Bits A[31..22] for External Access
with A[23..22] = 11 */
#define EBUFPILOCK          0x80000000      /* Gain Ownership of FPI Bus and Lock It */

#endif  /*  EBU_H */
```

File – Save

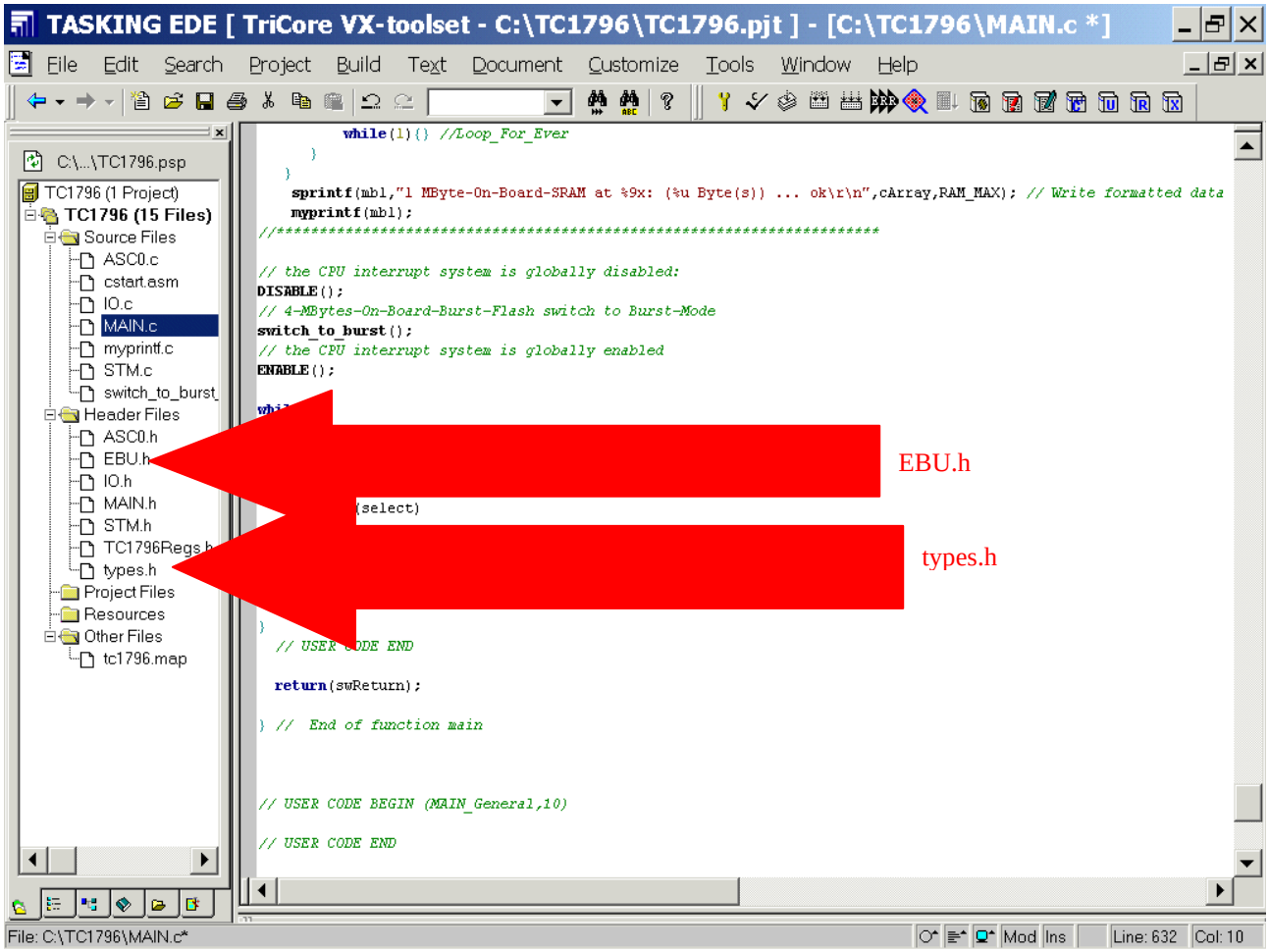
(Project Window File View) – TC1796 (Files) – right mouse button click – Add Existing Files – Browse





Select EBU.h
Select types.h

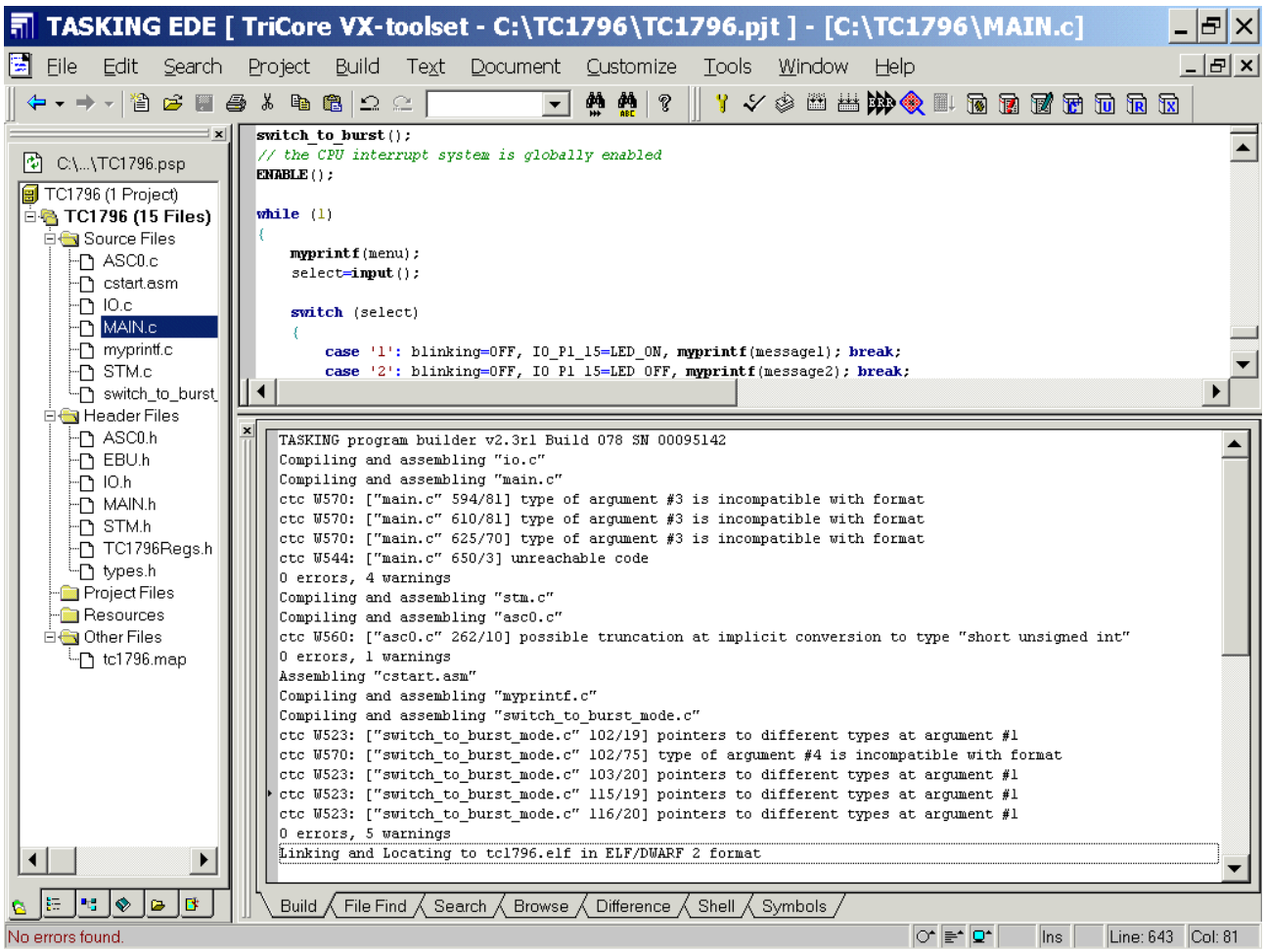
Open
OK


TASKING EDE [TriCore VX-toolset - C:\TC1796\TC1796.pjt] - [C:\TC1796\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1796.psp
TC1796 (1 Project)
TC1796 (15 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
STM.c
switch_to_burst_
Header Files
ASC0.h
EBU.h
IO.h
MAIN.h
STM.h
TC1796Regs.h
types.h
Project Files
Resources
Other Files
tc1796.map
while(1){} //Loop_For_Ever
sprintf(mbl,"1 MByte-On-Board-SRAM at %9x: (%u Byte(s)) ... ok\r\n",cArray,RAM_MAX); // Write formatted data
myprintf(mbl);
//*****
// the CPU interrupt system is globally disabled:
DISABLE();
// 4-MBytes-On-Board-Burst-Flash switch to Burst-Mode
switch_to_burst();
// the CPU interrupt system is globally enabled
ENABLE();
(select)
// USER CODE END
return(swReturn);
} // End of function main
// USER CODE BEGIN (MAIN_General,10)
// USER CODE END
EBU.h
types.h
File: C:\TC1796\MAIN.c*
Mod Ins
Line: 632
Col: 10

Generate your application program:

Build
Rebuild

or



Now you can close your project and Tasking EDE:

File - Close Project Space
File – Exit

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

Start pls-Debugger

File – Open Workspace

Look in: select C:\TC1796
File name: select tc1796.wsp

Open
Cancel

File – Load Program

Look in: select TC1796
File name: select tc1796.elf

Open

Click Program All

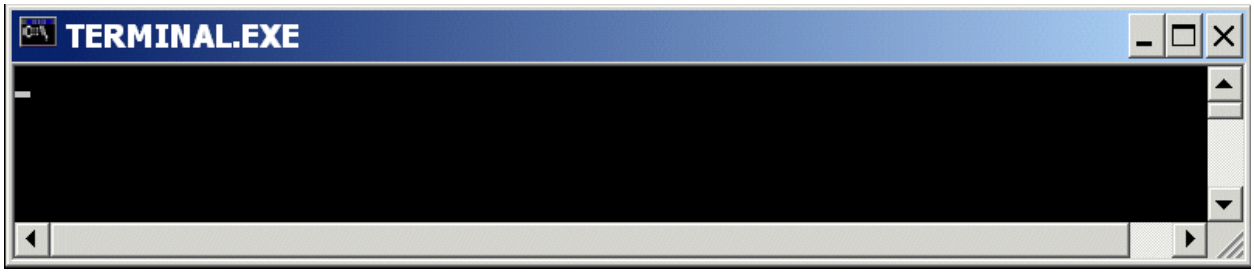
Exit
Exit

File – Close Workspace

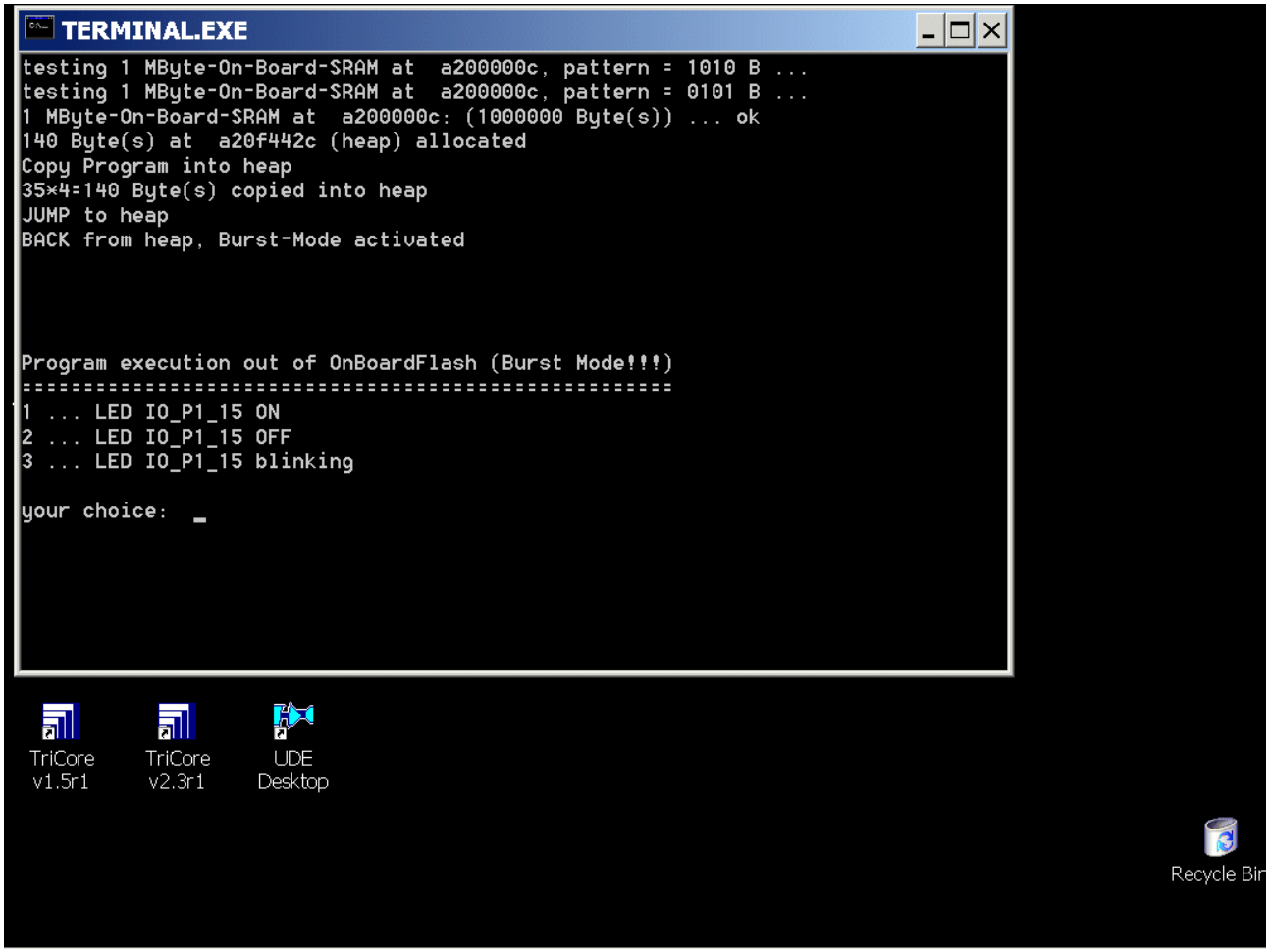
Yes

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:



8.) Time – Measurement:

Insert application specific program (time measurement):

For programming example
Chapter 4: Program_Execution_From_Scratch-Pad-RAM_SPRAM
Chapter 5: Program_Execution_From_OnChipProgramFlash
Chapter 6: Program_Execution_From_OnBoardProgramFlash
Chapter 7: Program_Execution_From_OnBoardProgramFlash_Burst-Mode

Double click: Main.c and change Global Variable menu
from

```
volatile unsigned int blinking=ON;
```

to

```
volatile unsigned int blinking=OFF;
```

Double click: Main.c and insert Global Variables

```
volatile unsigned int i = 0;
volatile unsigned int j = 0;
```

Double click: Main.c and insert the following endless loop (before “while (1)”):

```
//  - the CPU interrupt system is globally disabled
DISABLE();

while(1) // endless loop
{
  IO_vTogglePin(IO_P1_15);
  // time-consuming, dummy operations
  for (i=0;i<=100000;i++)
  {
    __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
    __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
    j=i;
    __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
    __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
  }
  __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
  __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
  __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
  __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
  __nop(); __nop(); __nop(); __nop(); __nop(); __nop(); __nop();
}
```

Result:

The frequency/time of the toggling pin `IO_P1_15` is a measurement for the speed of the application:

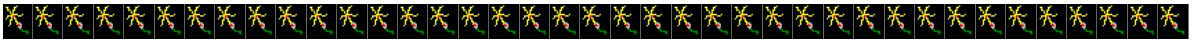
Program execution sorted by speed:

Program execution out of	Time [s]	Frequency [Hz] *1
SPRAM	0,0787	12,7
OnChipFlash	0,0907	11,0
OnBoardFlash (Burst Mode)	0,3734	2,7
OnBoardFlash	0,7854	1,3

Speed

*1 … higher is better

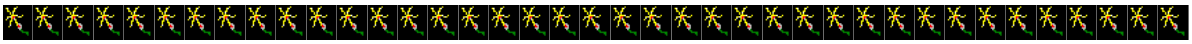
9.) Feedback (TC1796): Your opinion, suggestions and/or criticisms



Contact Details (this section may remain empty 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