Application Note, V2.0, April 2007

AP32117

TC1130

TC1130 "Cookery-Book" for a "Hello world" application using Tasking and pls Tools

Microcontrollers

Never stop thinking

Edition 2008-07-17

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:	2007-04	V2.0
Previous Version:	none	

Page	Subjects (major changes since last revision)

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all?
Your feedback will help us to continuously improve the quality of this document.
Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com

Table of Contents **Page**

Note: Table of Contents see page 8.

Introduction:

This “Appnote” is an Infineon Hands-On-Training.
It will help inexperienced users to get an TC1130 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.

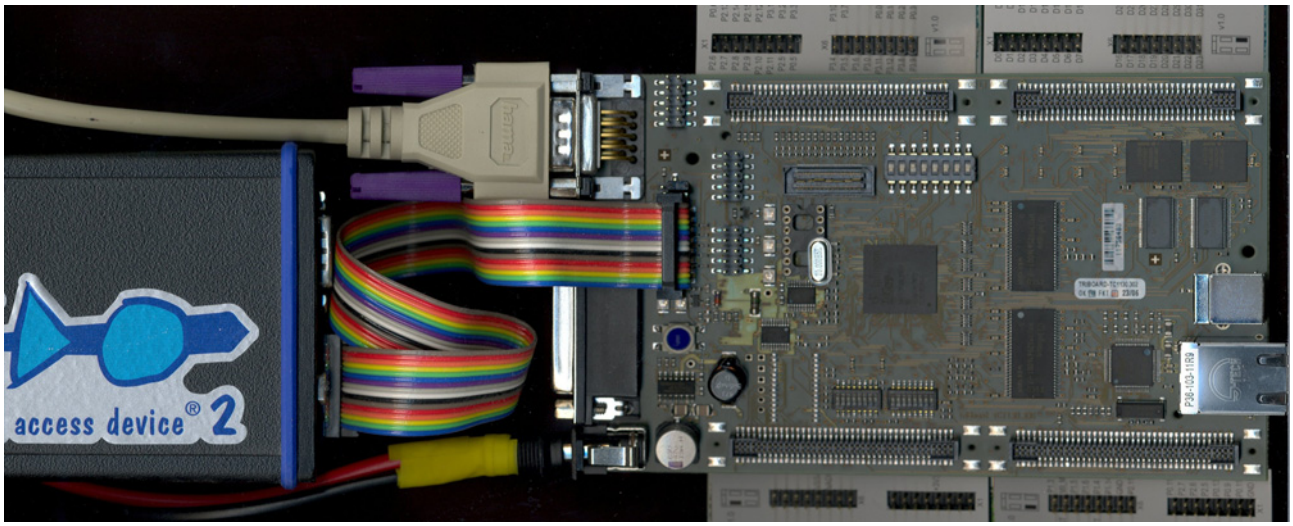
The purpose of this document is to gain know-how of the microcontroller and the tool-chain.
Additionally, the "hello-world-example" can easily be expanded to your needs.
You can connect either a part of - or your entire application to the Starter-Kit-Board.
You are also able to benchmark any of your 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_Code-Scratch-Pad-RAM (PMI_SPRAM)
Chapter 5: Program_Execution_From_OnBoardProgramFlash
Chapter 6: Program_Execution_From_OnBoardProgramFlash_Burst-Mode

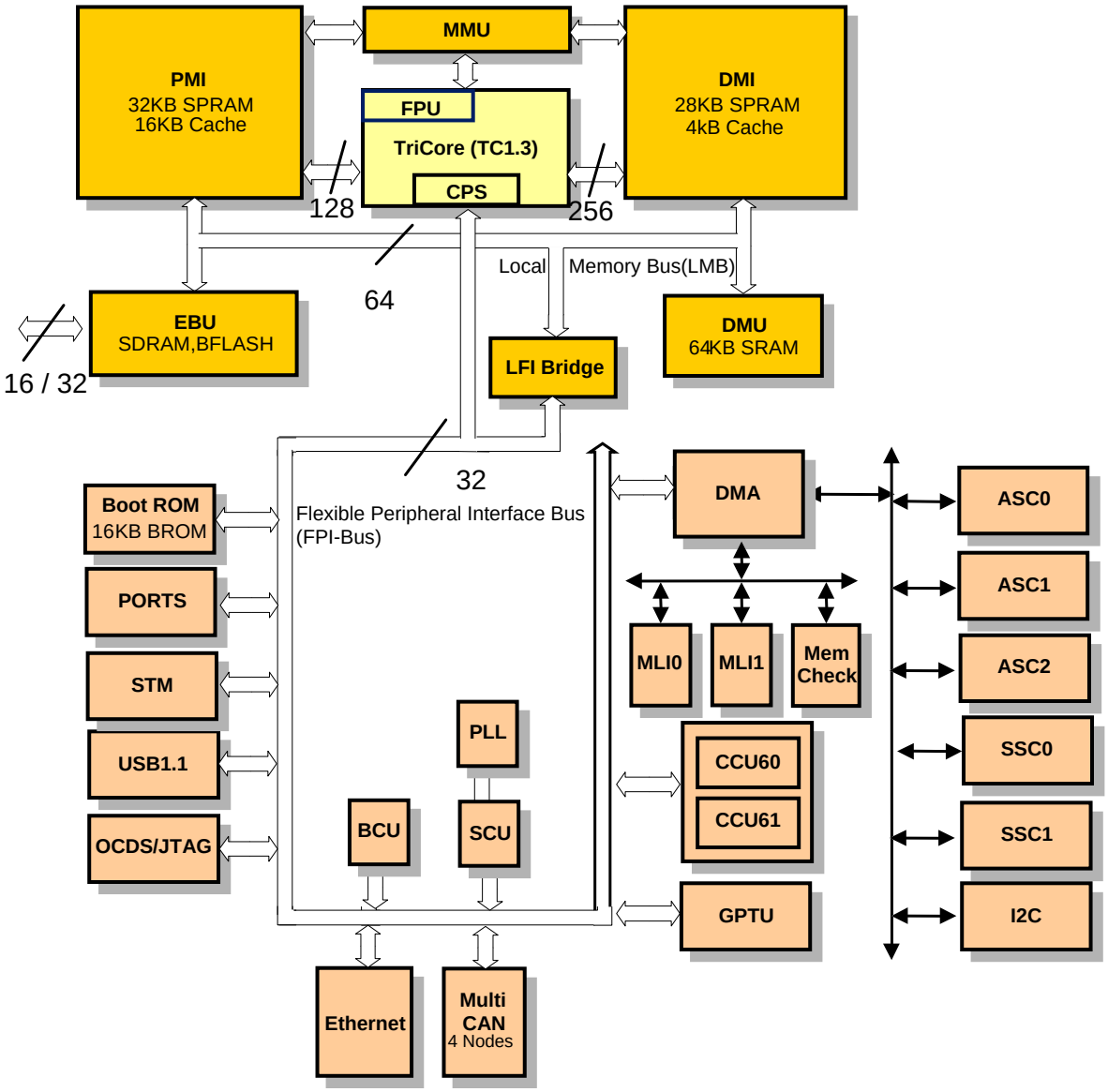
Note:
The style used in this document focuses on working through this material as fast and easily as possible. That means there are full screenshots instead of dialog-window-screenshots; extensive use of colours and page breaks; and listed source-code is not formatted to ease copy & paste.

Have fun and enjoy TriCore!

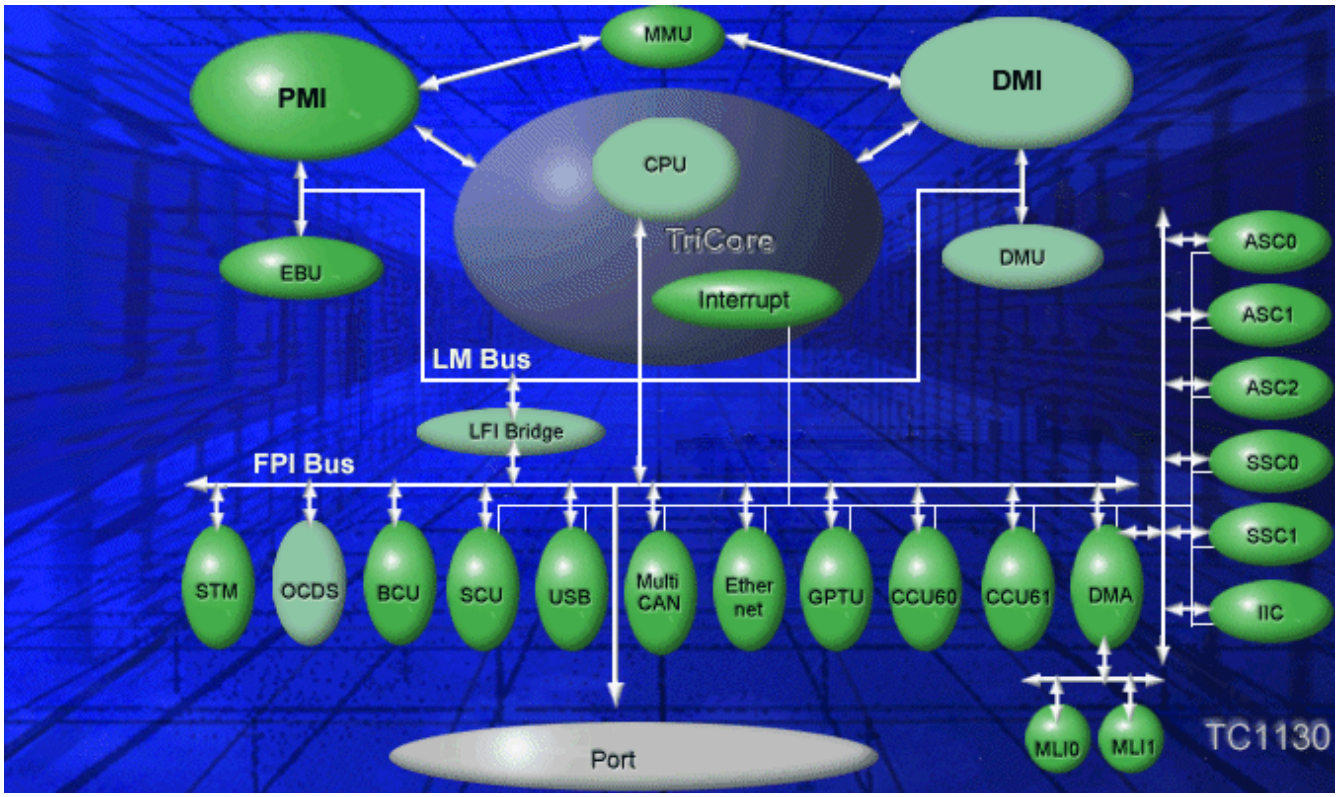
Programming Examples TC1130 Starter Kit



TC1130 Block Diagram (Source: Product Marketing)

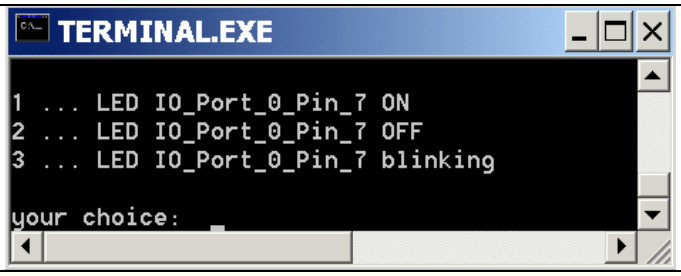


TC1130 Block Diagram (Source: code generator DAvE)



"Cookery-Book"

For your first programming example for the TC1130 Starter-Kit-Board:

	Your program: 
Chapter/ Step:	***** Recipes *****
1.)	**TC1130 Board** Power Supply, Jumper Setting, Serial cable to the notebook, pls-Debugger
2.)	**DAvE – program generator** DAvE installation (mothersystem) + DAvE Update-installation for TC1130 (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 into the 32 KBytes code scratchpad RAM. (PMI_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 into the 32 MBytes OnBoardFlash, using OnChipSRAM + OnBoardSDRAM)
6.) 	**Using the TASKING Development Tools (C/C++/EC++ Compiler)** Programming of your application with TASKING (Altium) tool chain (EDE) - v2.3r1 Locating programs into the 32 MBytes OnBoardFlash (Burst-Mode!), using OnChipSRAM (28 KBytes DMI_SPRAM + 64 KBytes DMU_SRAM) + 64 MBytes OnBoardSDRAM

Additional exercises

7.)	Time – Measurement (Using an oscilloscope / a logic analyzer)
8.)	Time – Measurement [Using the SystemTimer (STM)]

Feedback

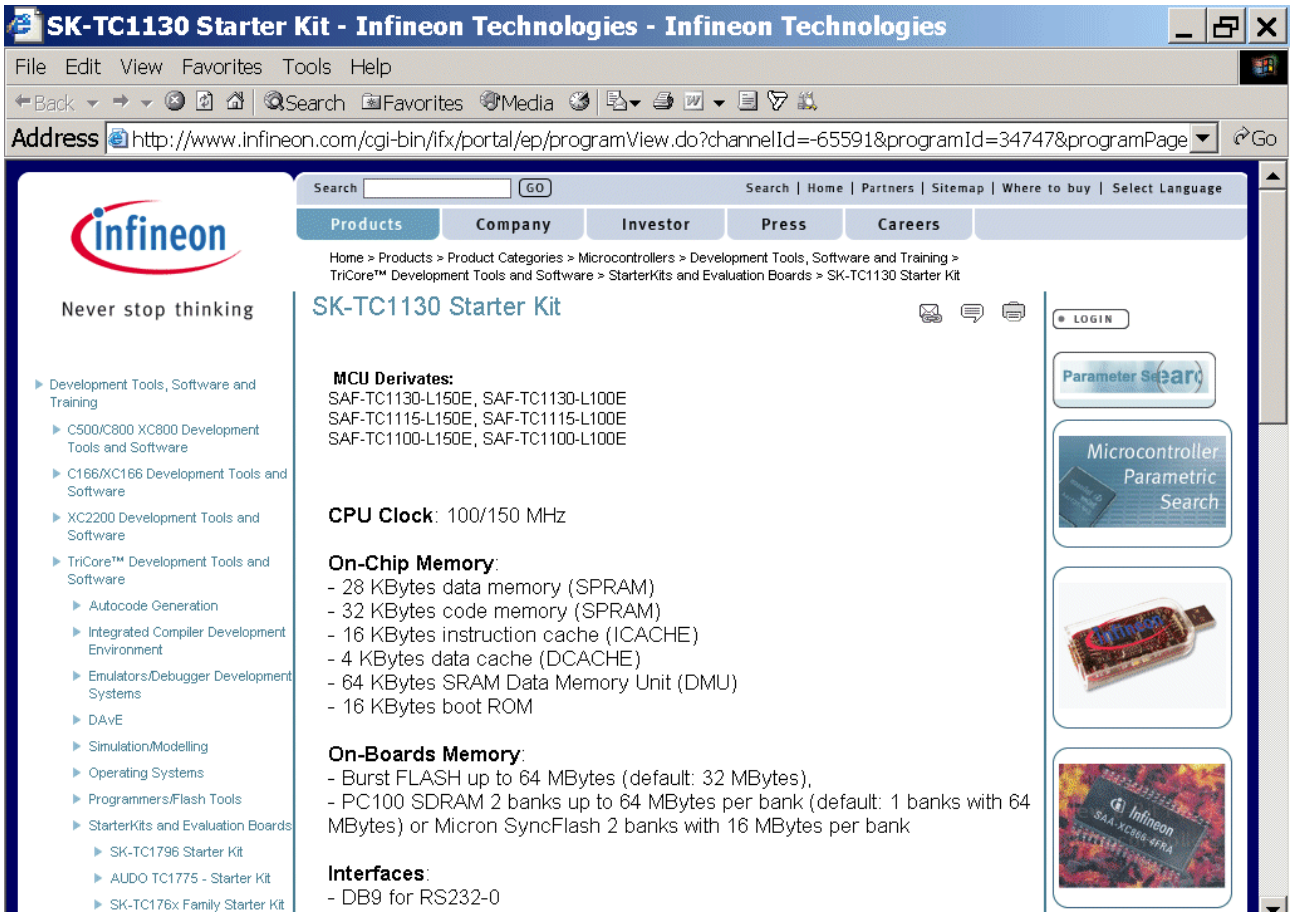
9.)	Feedback

1.) TC1130 Starter Kit Board:

Ordering information:

http://www.infineon.com/cgi-bin/ifx/portal/ep/programView.do?channelId=-65591&programId=34747&programPage=%2Fep%2Fprogram%2Finformation.jsp&pageTypeId=17099&BV

Screenshot of the TC1130 Starter-Kit Homepage:



- SK-TC1765 Starter Kit
- SK-TC1130 Starter Kit
- SK TC116x-Series Starter Kit
- Software Partners
- Software Downloads
- Training
- DAvE-Digital Application virtual Engineer
- DAvE Drive

- BERG10 for RS232-1
- two BERG10 for CAN-0/1 with Transceiver
- BERG16 for OCDS1
- SAMTEC QSH-030-01-F-D-A for OCDS2
- RJ45 Connector with LED's for Ethernet
- USB connector (type B) for USB connections
- DB25 On Board Wiggler for OCDS1
- four 80-pin connectors (male) with all I/O signals
- four 80-pin connectors (female) with all I/O signals

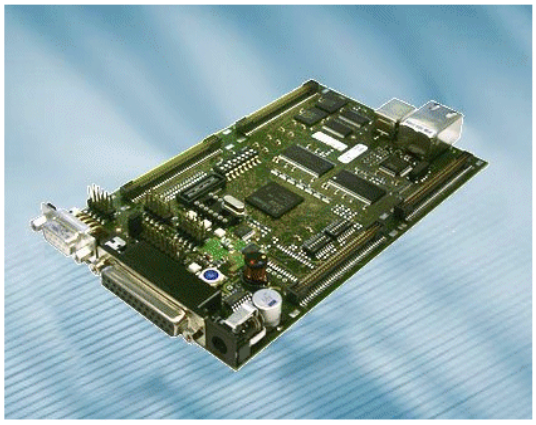
Includings:
- TC1130 User manuals (pdf),
- TriCore Architecture manual (pdf),
- TriBoard manual (pdf),
- Tools: Compiler, Debugger, Operating System from Infineon Tool Partners (Evaluation versions)
- Cable,
- 1 Extension Board,
- Power plug

Order Nr.: B158-H8359-X-X-7600

Price*: 400,- EUR

How to order?
Buy Online

or please contact your local distributor: **http://www.infineon.com/distribution.**



▽ **Documentation**

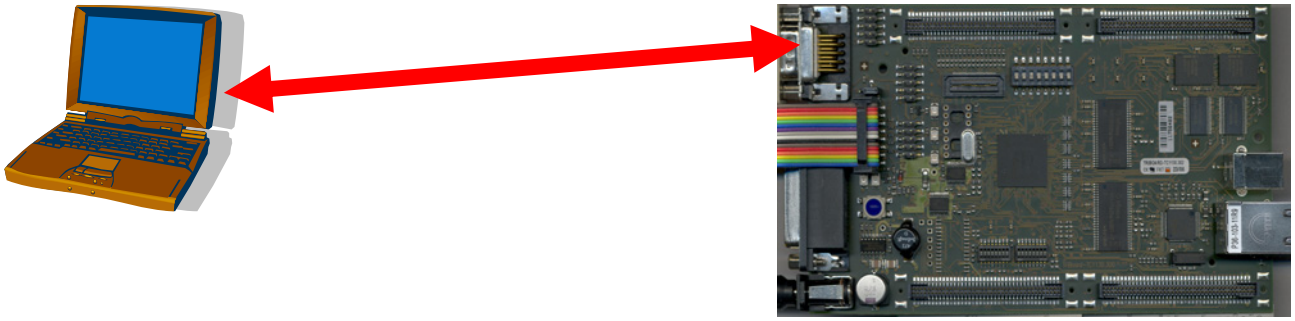
Description	File Name	Size	Date	Revision
TriBoard TC1130 - Hardware Manual TC1130-300	TriBoardManual-TC1130-V12.pdf	1.6 MB	1 Jul 2005	

Local intranet

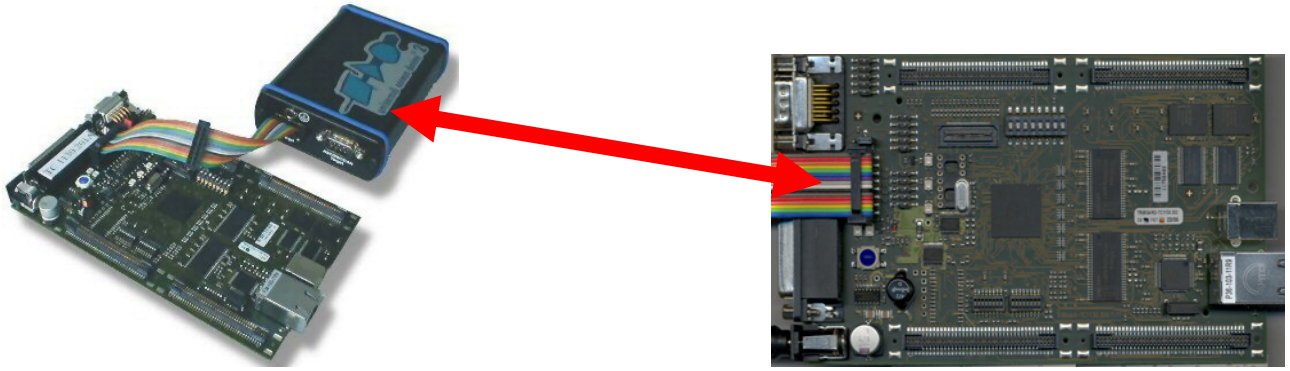
Connecting the TC1130-Board to the Environment:

Connect a Power Supply:
The TC1130 Board requires an external power supply.
A (un)regulated DC power supply from 5,5 to 60 Volts can be connected to the power connector.
500mA are sufficient for the TC1130 Starterkit.

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



Connect the pls-Debugger (used for: On-Board-Flash-Programming and Debugging):

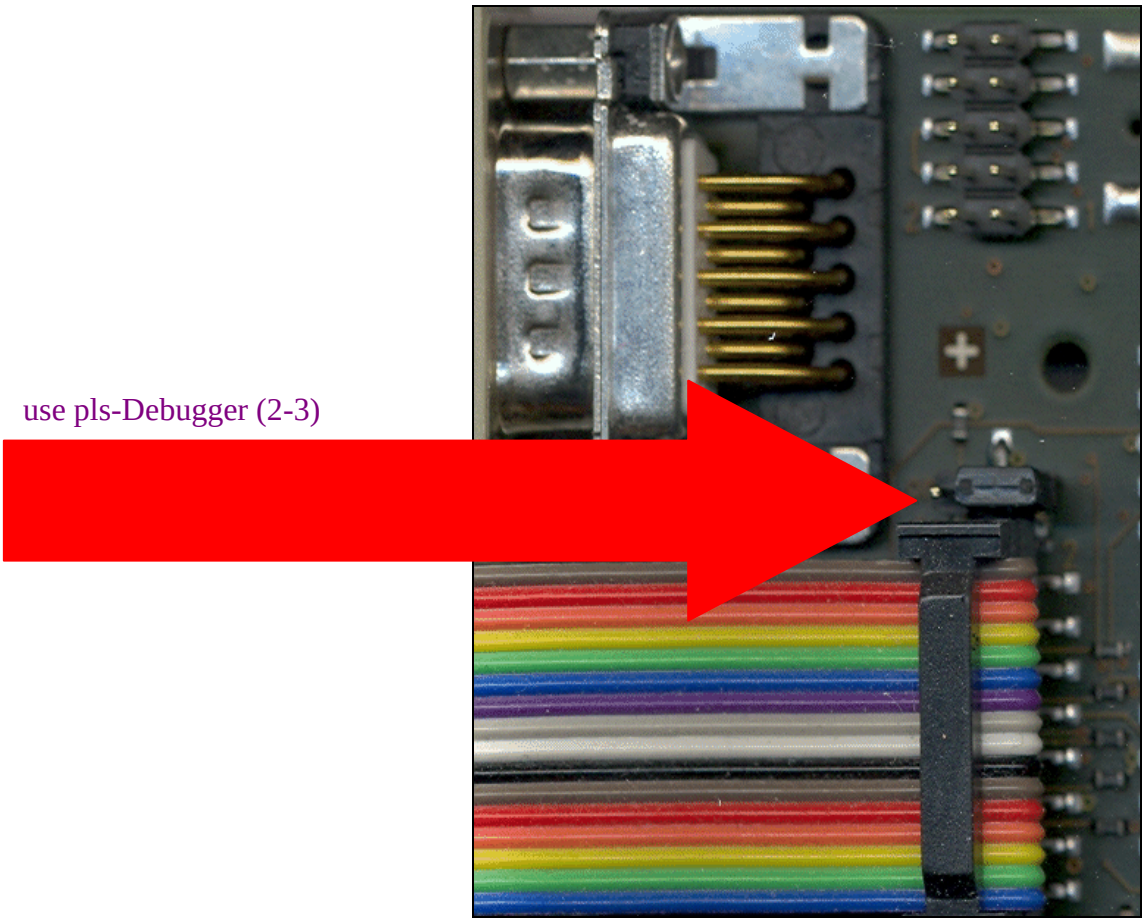


For further information, please refer to the TriBoard TC1130 V1.2, June 2004 Hardware Manual .

Jumper Settings (Jumper JP501):

Note:
We use the pls-Debugger in this document.

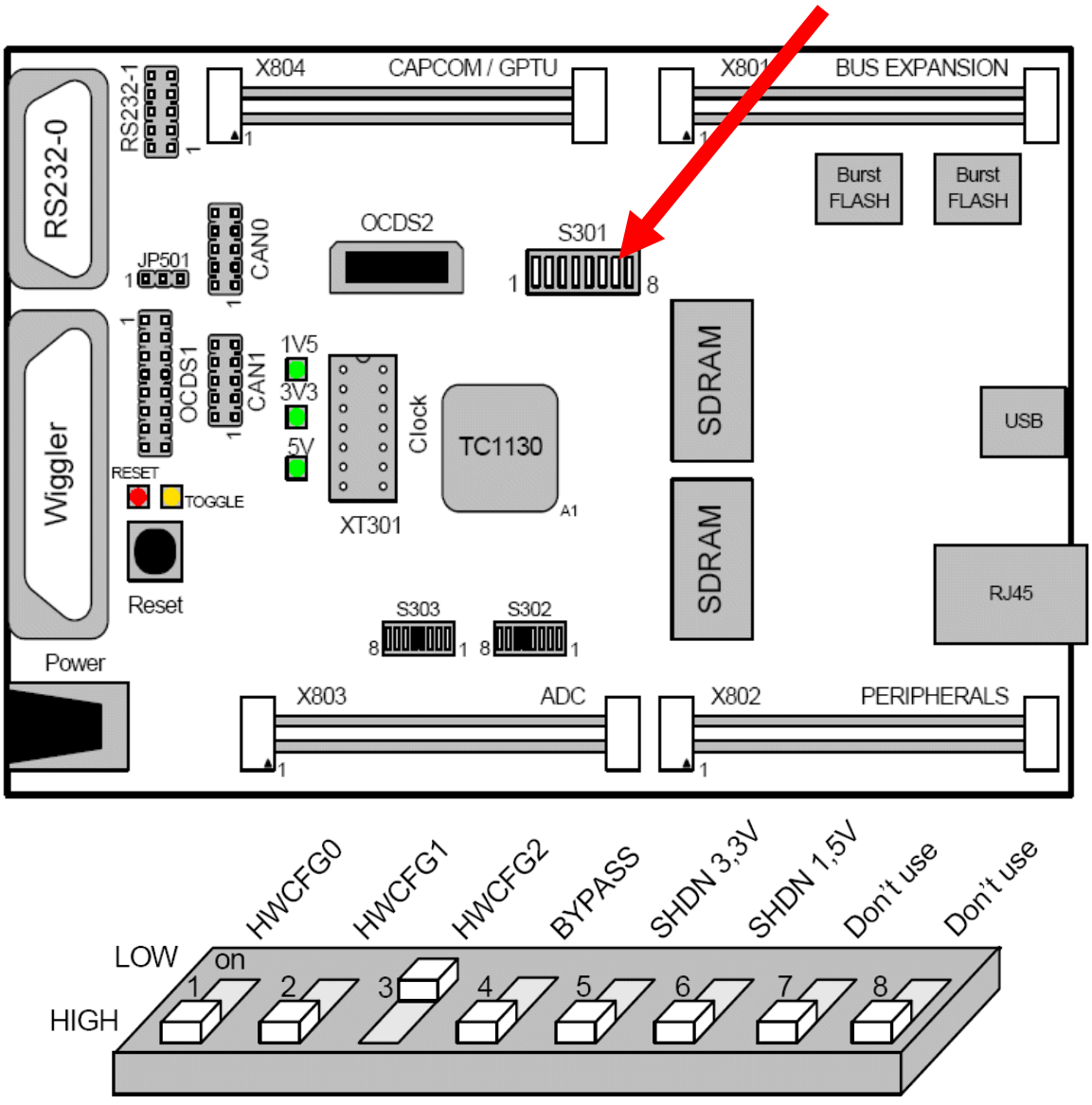
Jumper JP501
1-2 … Enable On Board Wiggler (use parallel-on-board-interface)
2-3 … Disable On Board Wiggler (use pls-Debugger)

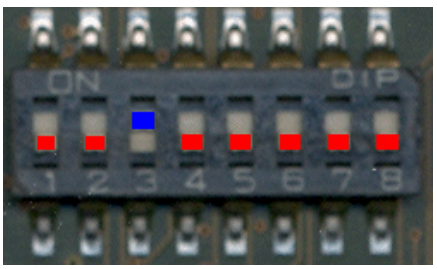


TC1130-Execution-Environment-SPRAM:

Note:
At the beginning we use the default configuration of Jumper S301.

Jumper Settings (HW-Configuration DIP-Switch S301)

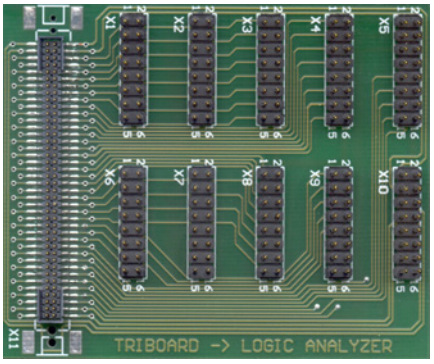




S301:
1, 2, 4, 5, 6, 7, 8 : OFF (default)
3 : ON (default)

Accessories for the TC1130 Starter Kit – Extension Boards

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



Ordering information:

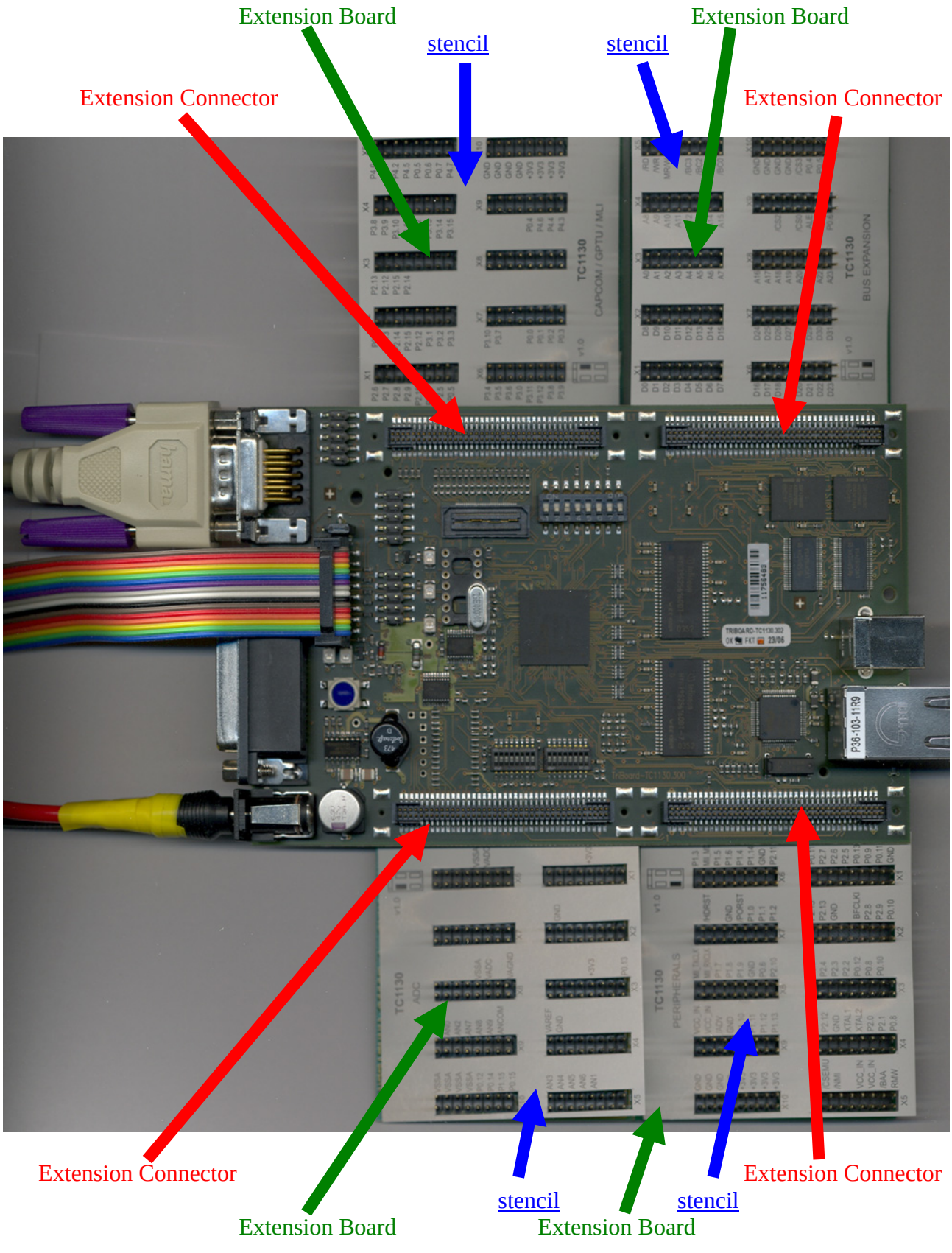
Name: TriBoard+XC16x-Adapter-Platine
Price: approximately € 32,- apiece, (4 required)

Purpose:
Extension Boards are used to measure the signals on the extension connectors (see next page) and/or to connect the Starter-Kit-Board with your application.

You can order them at:

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

Connecting the Extension Boards to the Starter-Kit / Using stencils



2.) DAvE – Installation for TC1130 microcontrollers:

Install DAvE:

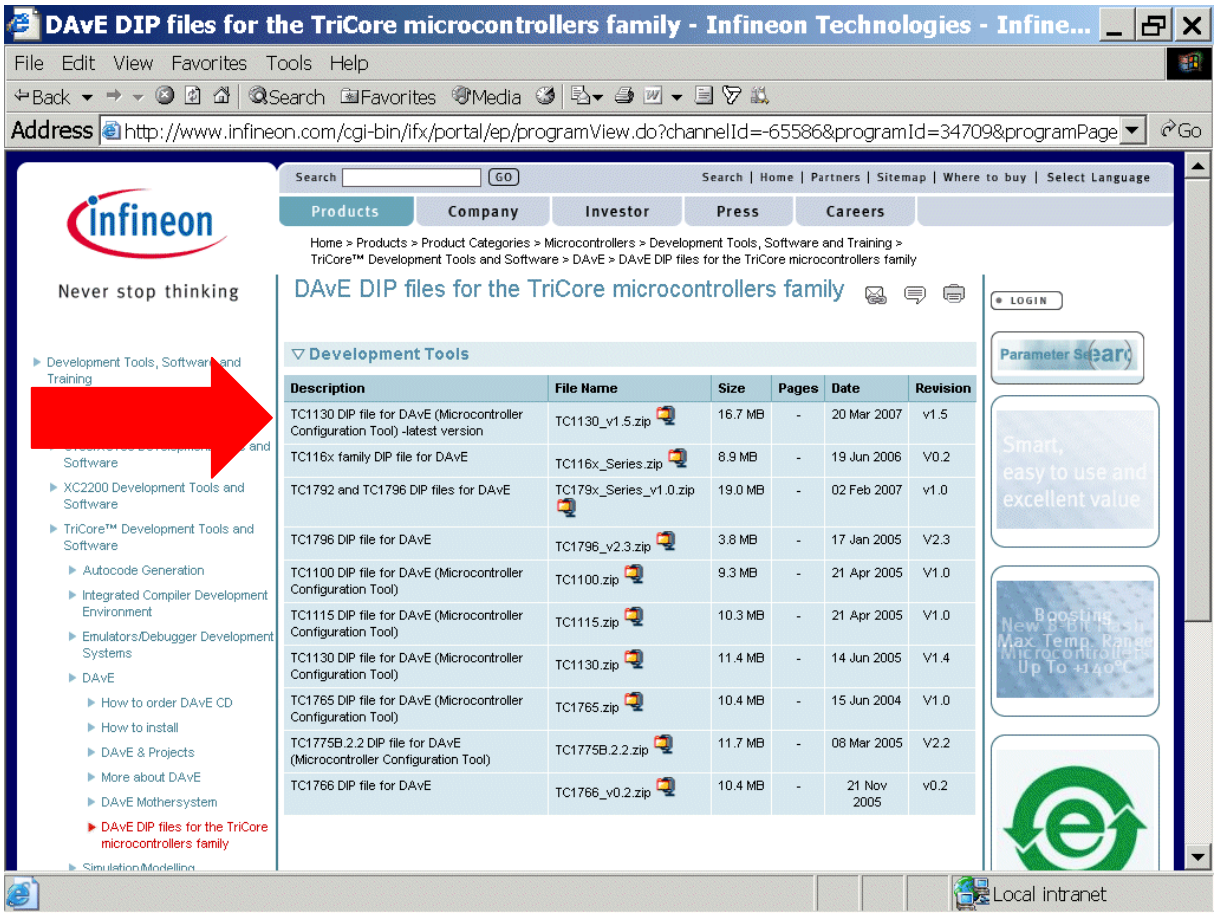
Download @ http://www.infineon.com/DAvE the DAvE-mothersystem setup.exe

Title	Date	Size
Tool Package		
DAvE - Mothersystem - latest version	05 Feb 2007	14.8 MB
DAvE - Mothersystem	04 Jul 2006	15.1 MB

and execute setup.exe to install DAvE .

Install the TC1130 microcontroller Update:

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



Unzip the zip-file TC1130_v1.5.zip and save "TC1130_v1.5.dip "
@ e.g. D:\DAvE\TC1130-2007-04-02\TC1130_v1.5.dip .

2.)

Start DAvE - (click 

)

3.)

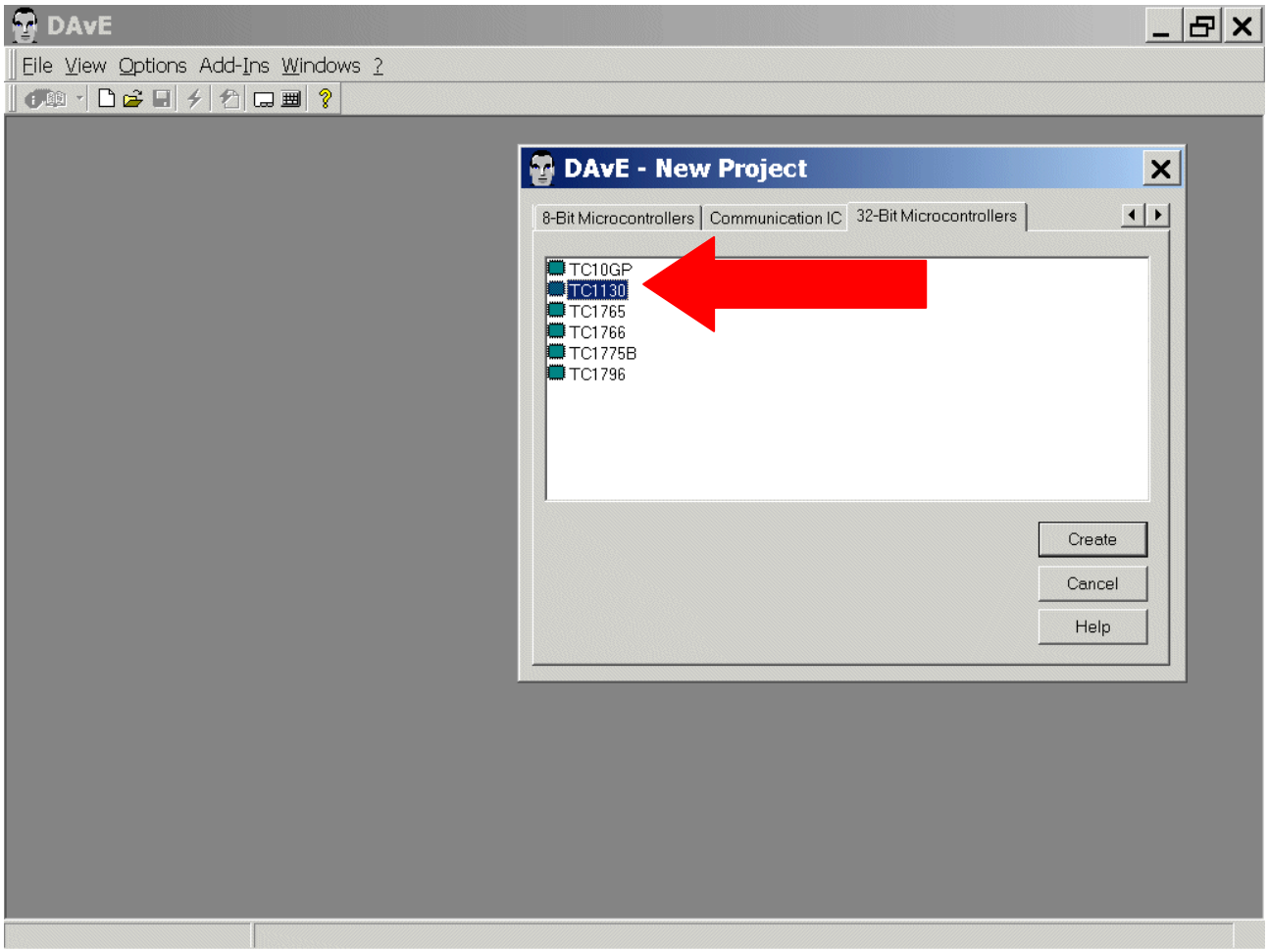
View
Setup Wizard
Default: • Installation
Forward>
Select: • I want to install products from the DAvE's web site
Forward>
Select: D:\DAvE\TC1130-2007-04-02
Forward>
Select: Available Products
click ✔ TC1130
Forward>
Install
End

4.) DAvE is now ready to generate code for the TC1130 microcontroller.

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

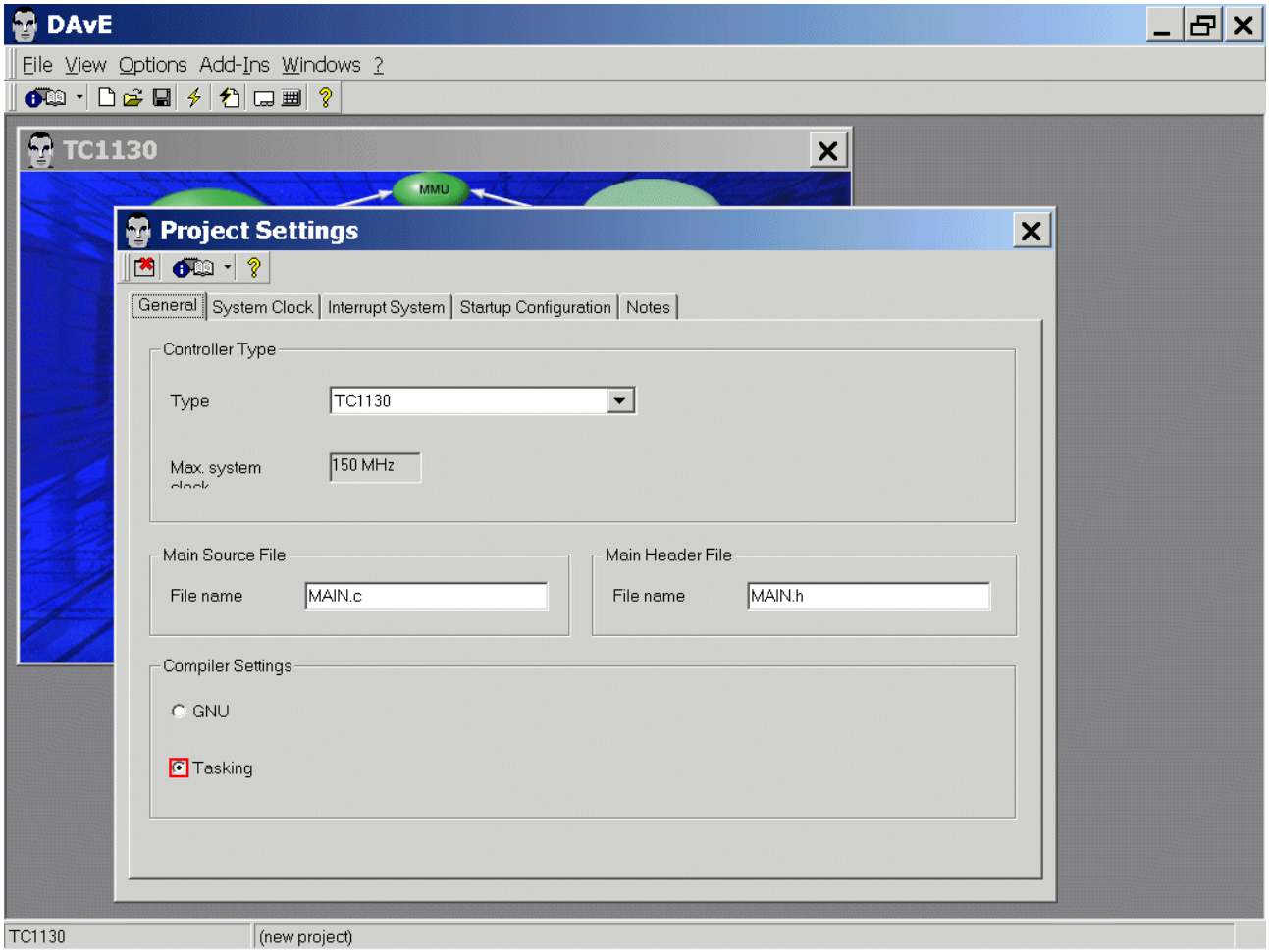
Start the program generator DAvE and select the TC1130 microcontroller:

File
New
32-Bit Microcontrollers
TC1130
Create



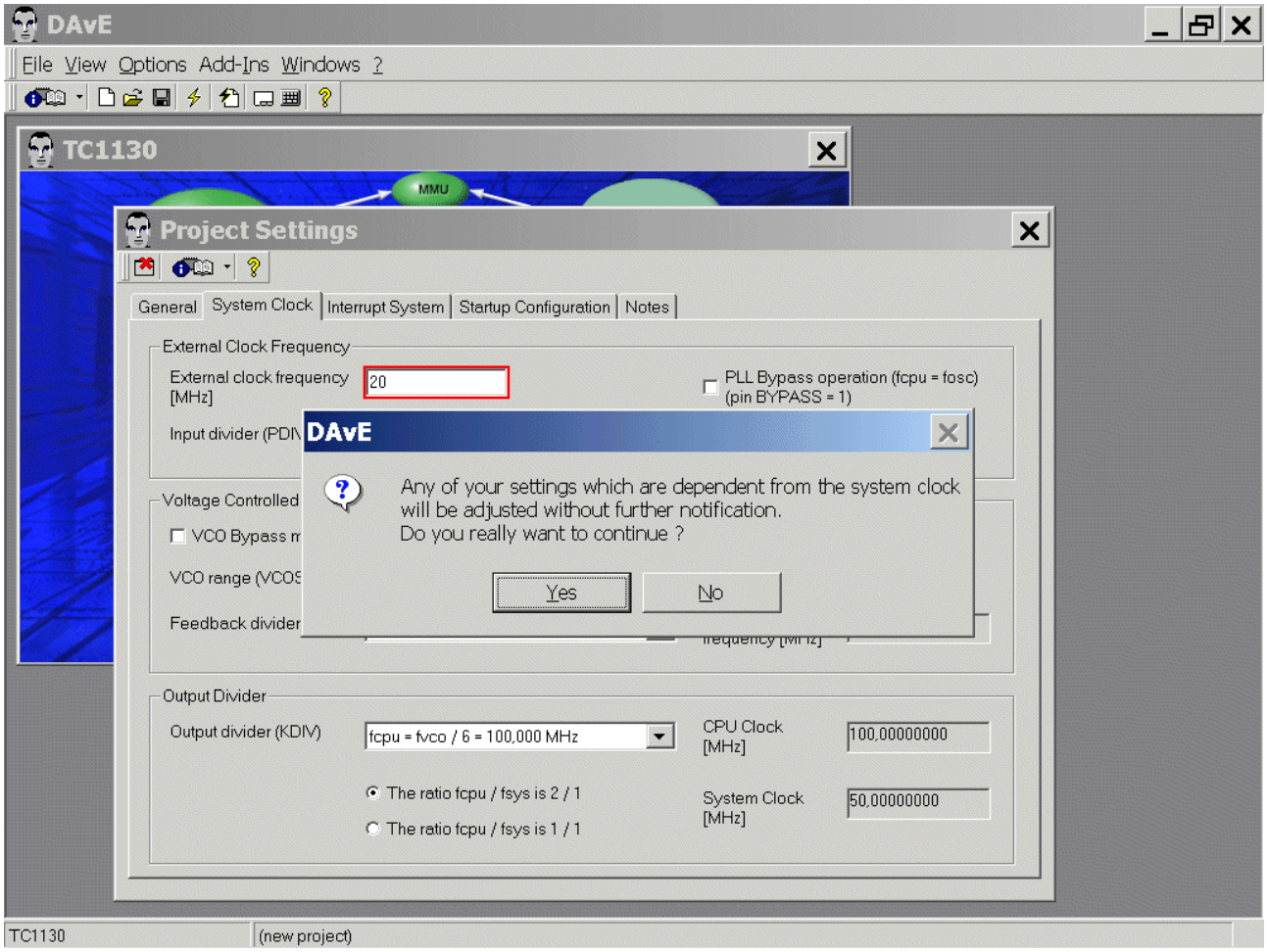
Choose the Project Settings as you can see in the Screenshots:

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



System Clock: CPU Clock will be 150 MHz:

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

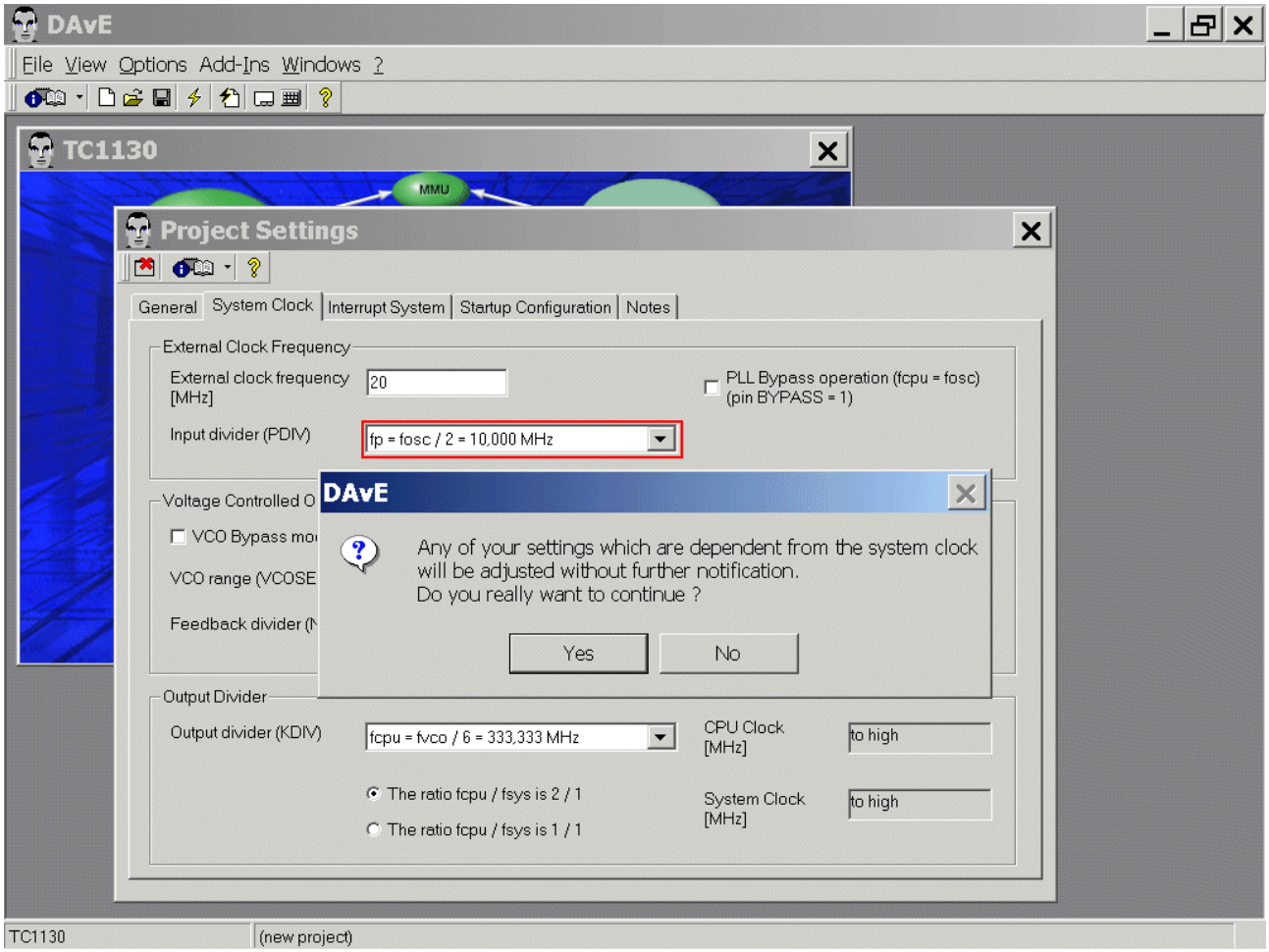


Yes

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

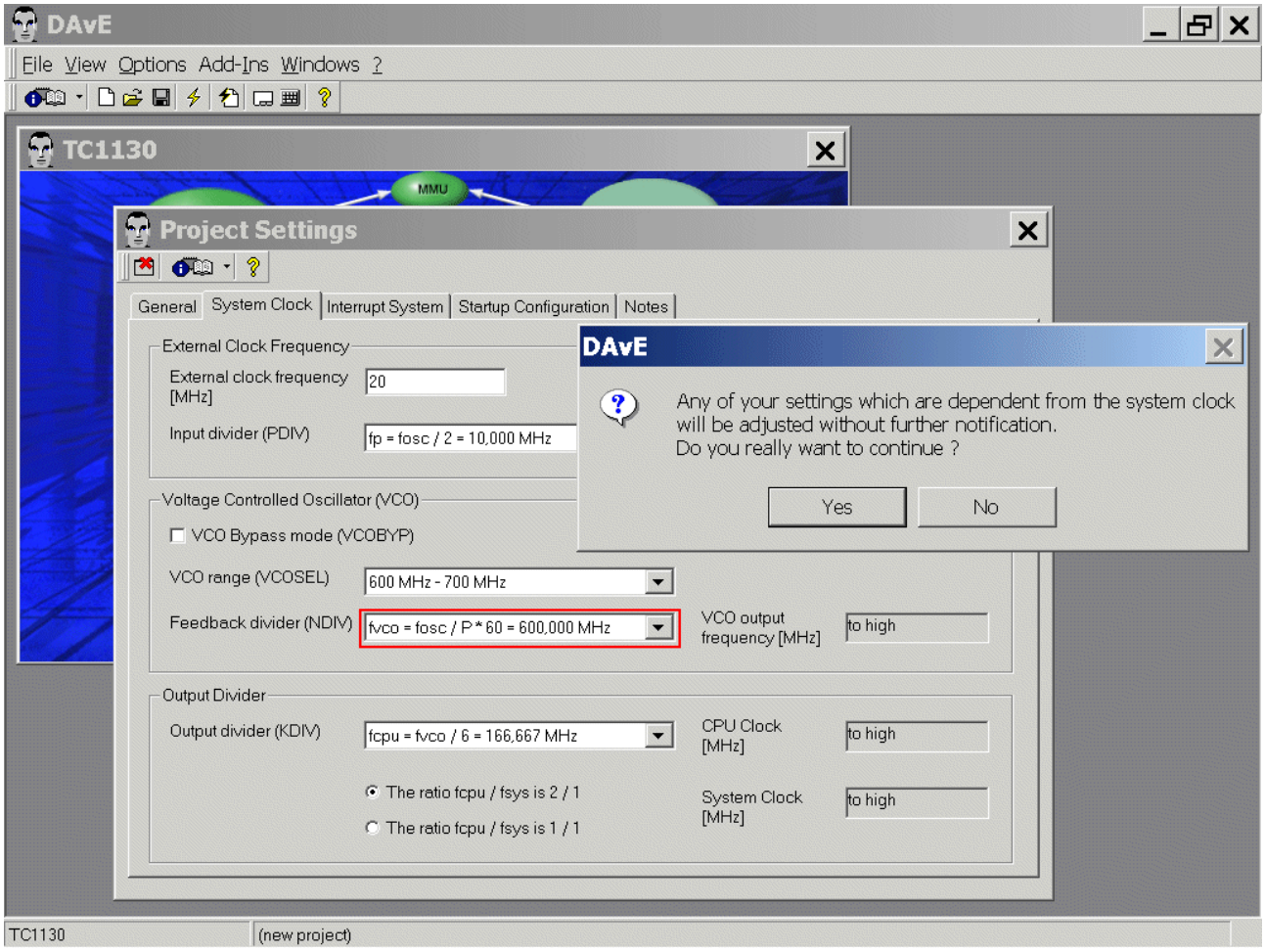
Note:
Validate each alpha numeric entry by pressing ENTER.

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



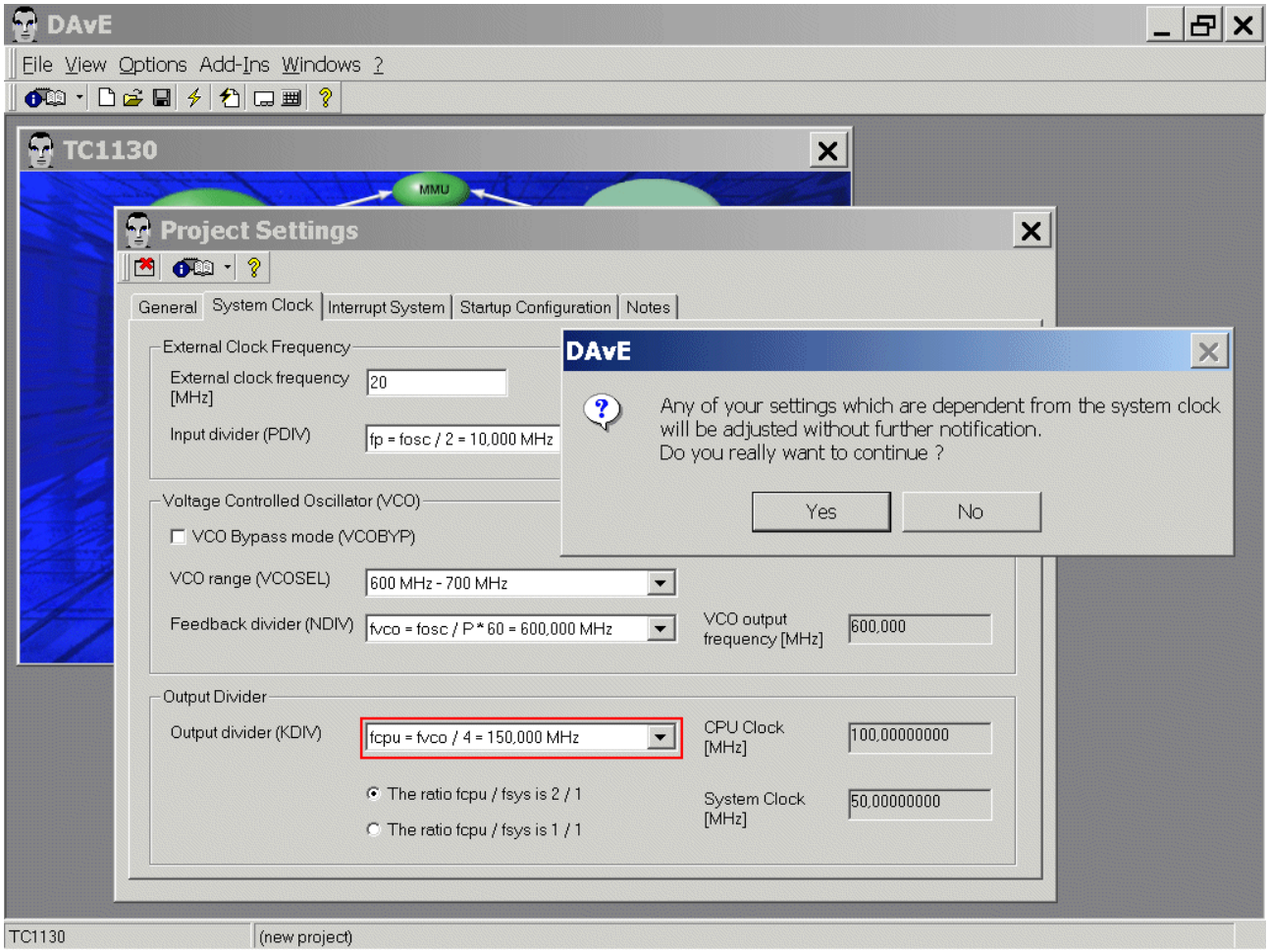
Yes

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



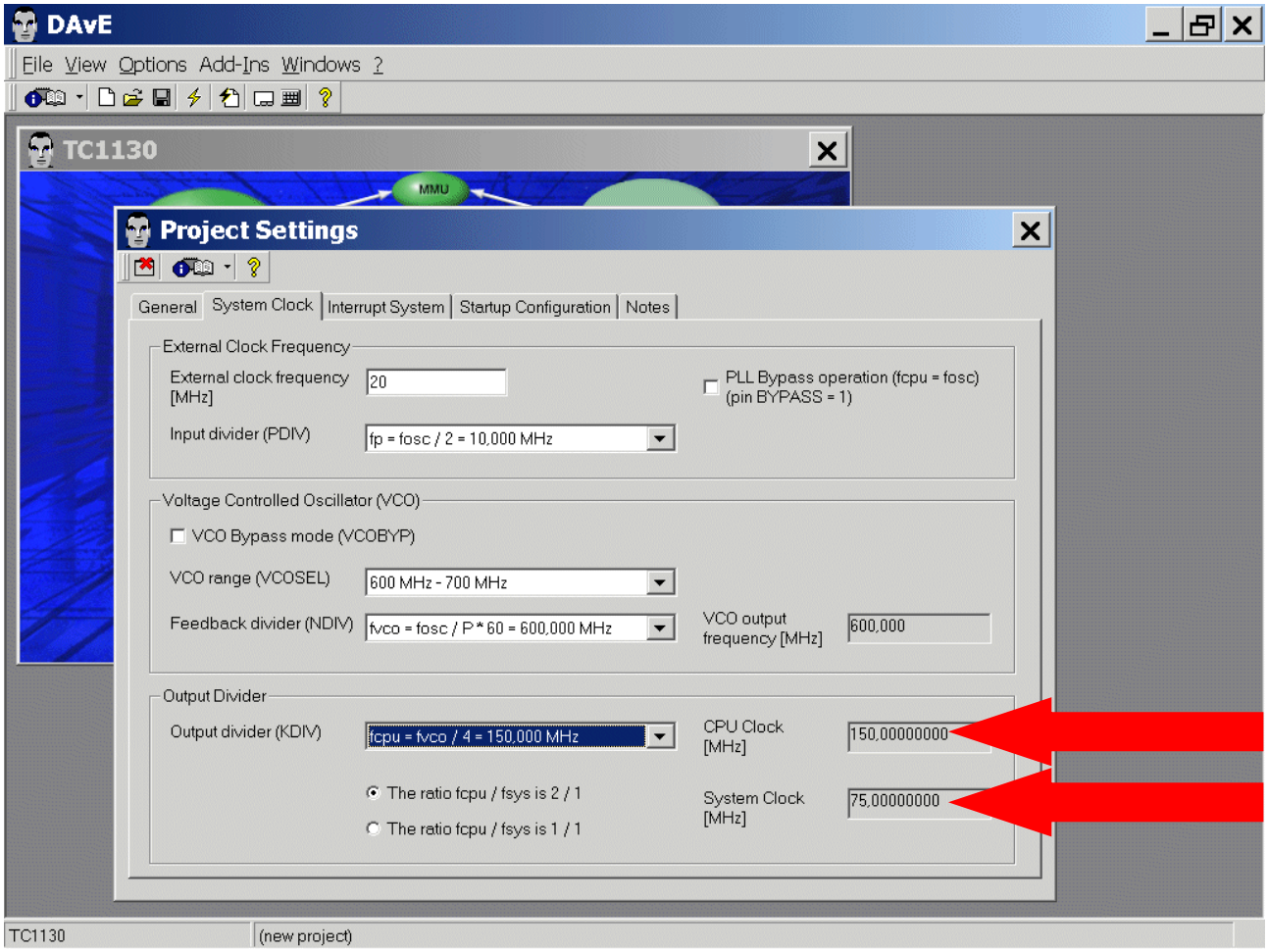
Yes

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

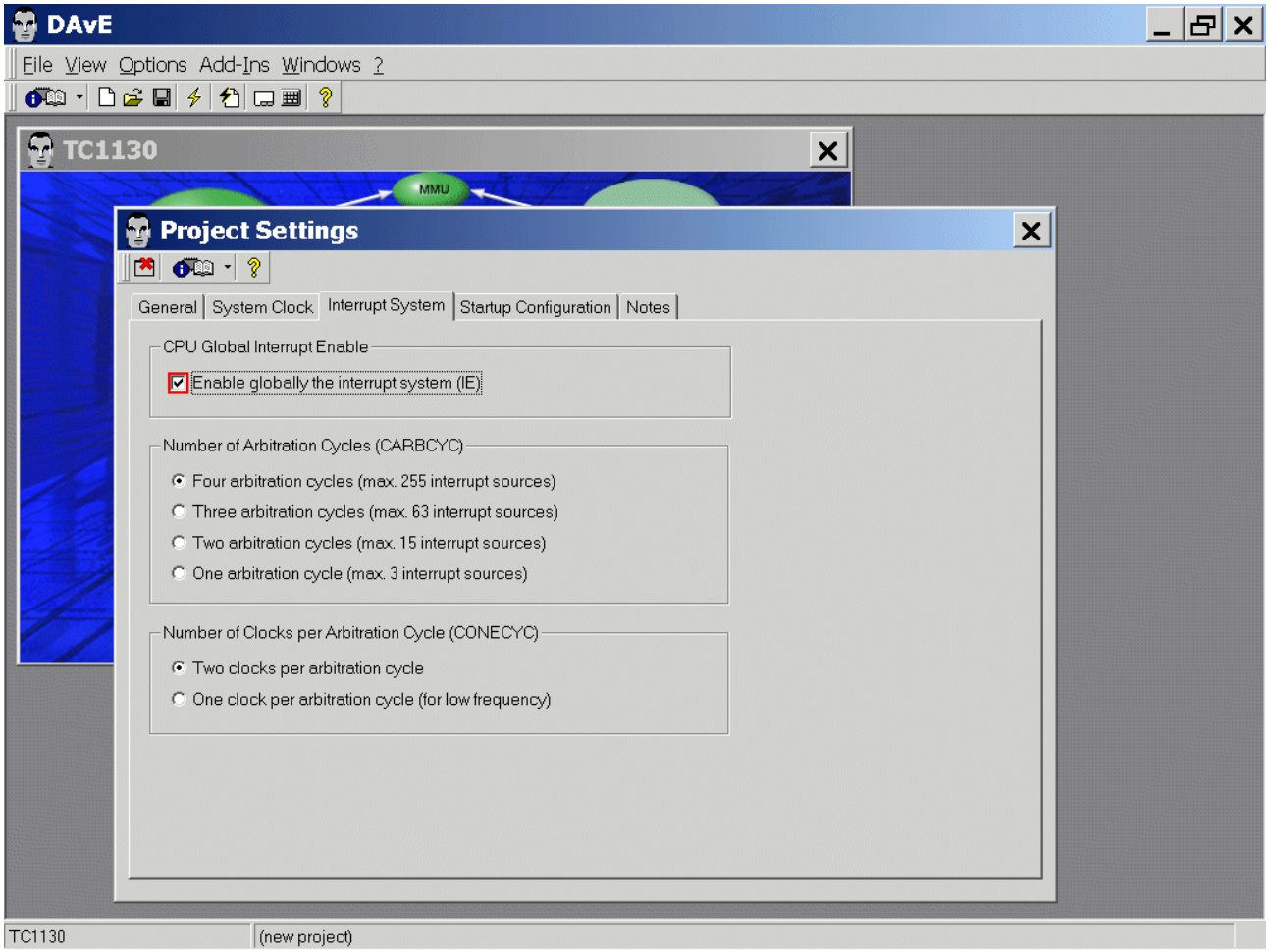


Yes

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)



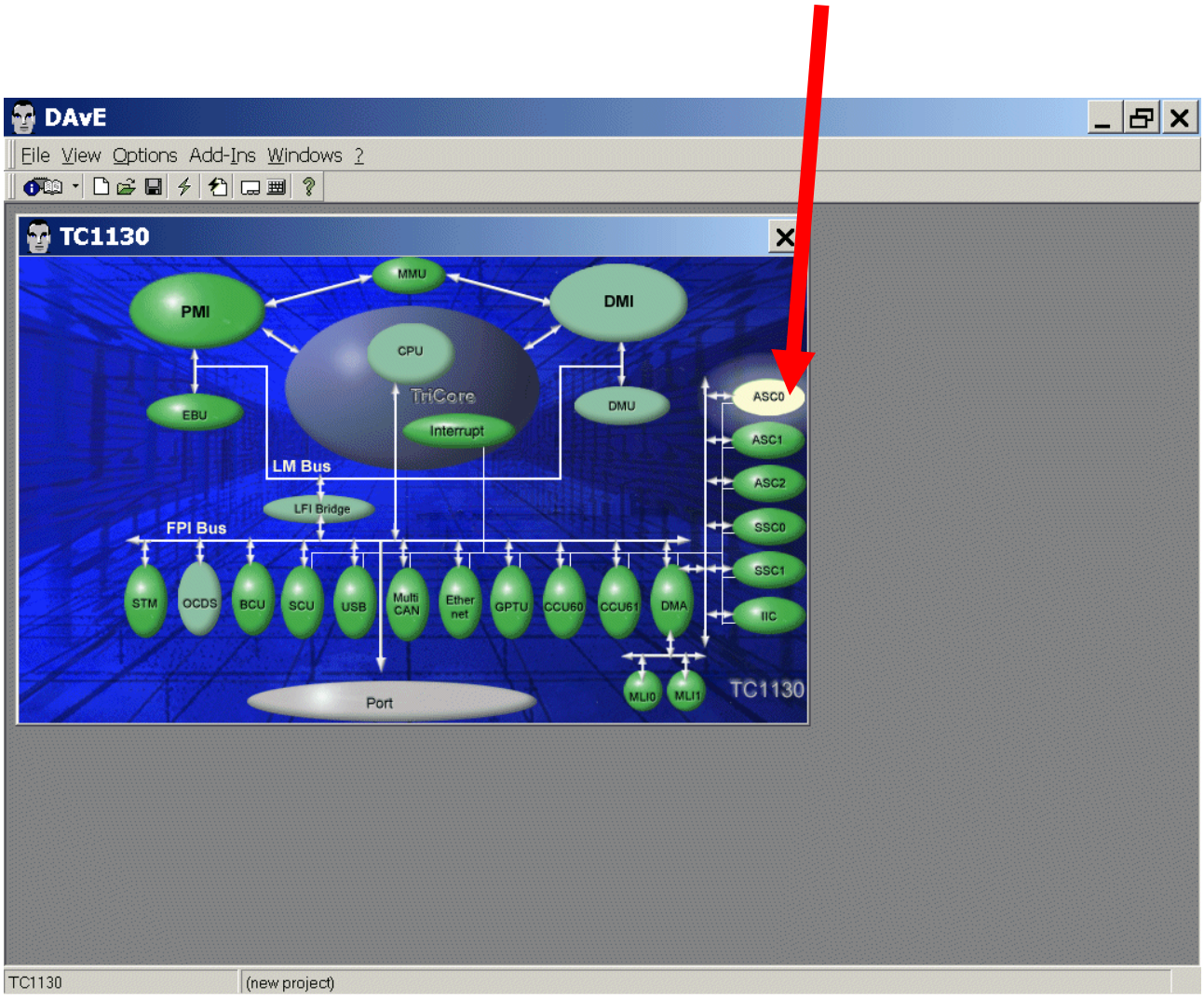
Startup Configuration: Hardware Booting Scheme: (do not care !!!)

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

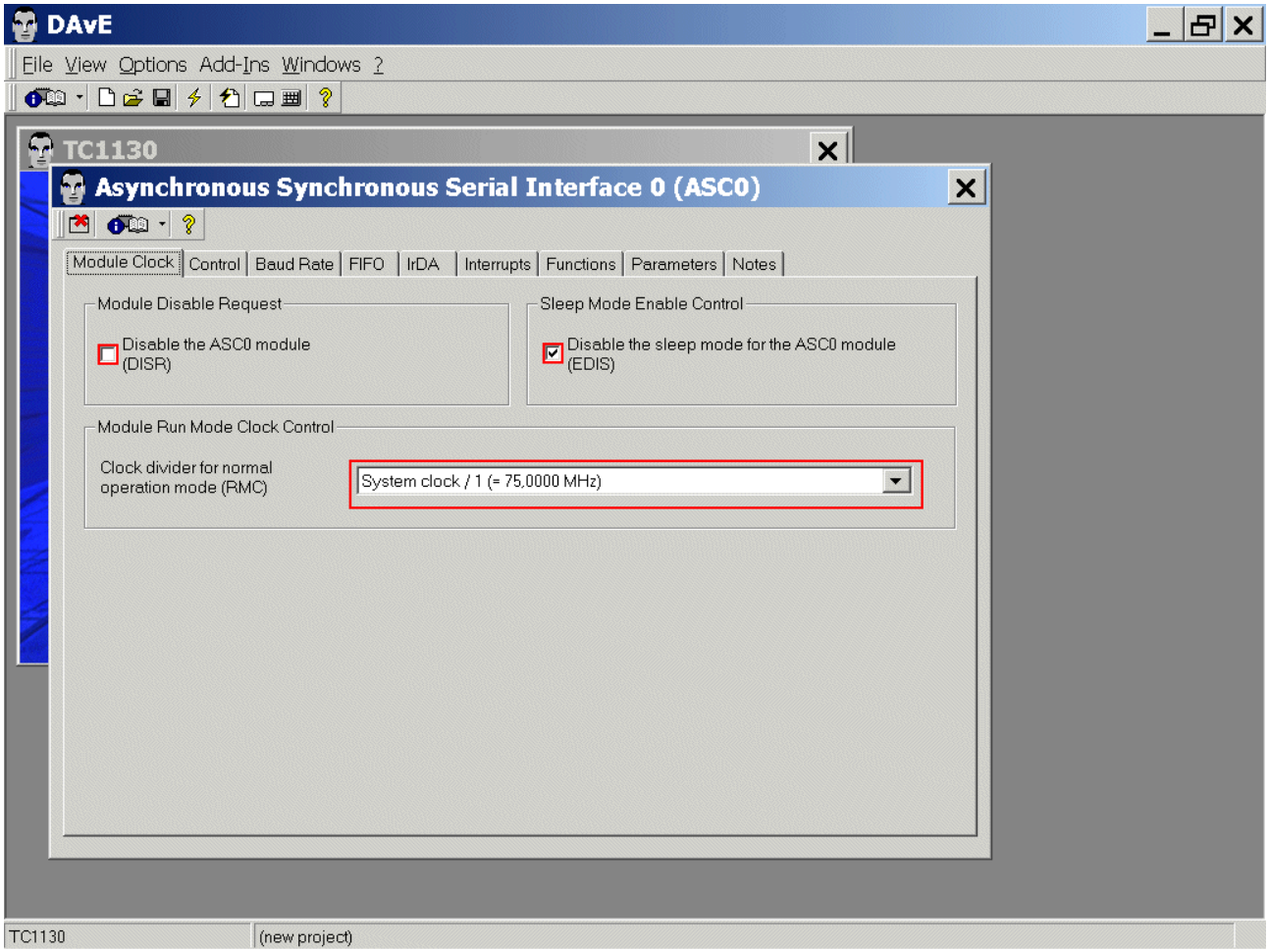
Exit this dialog now by clicking the close button:

Configuration of the ASC0:

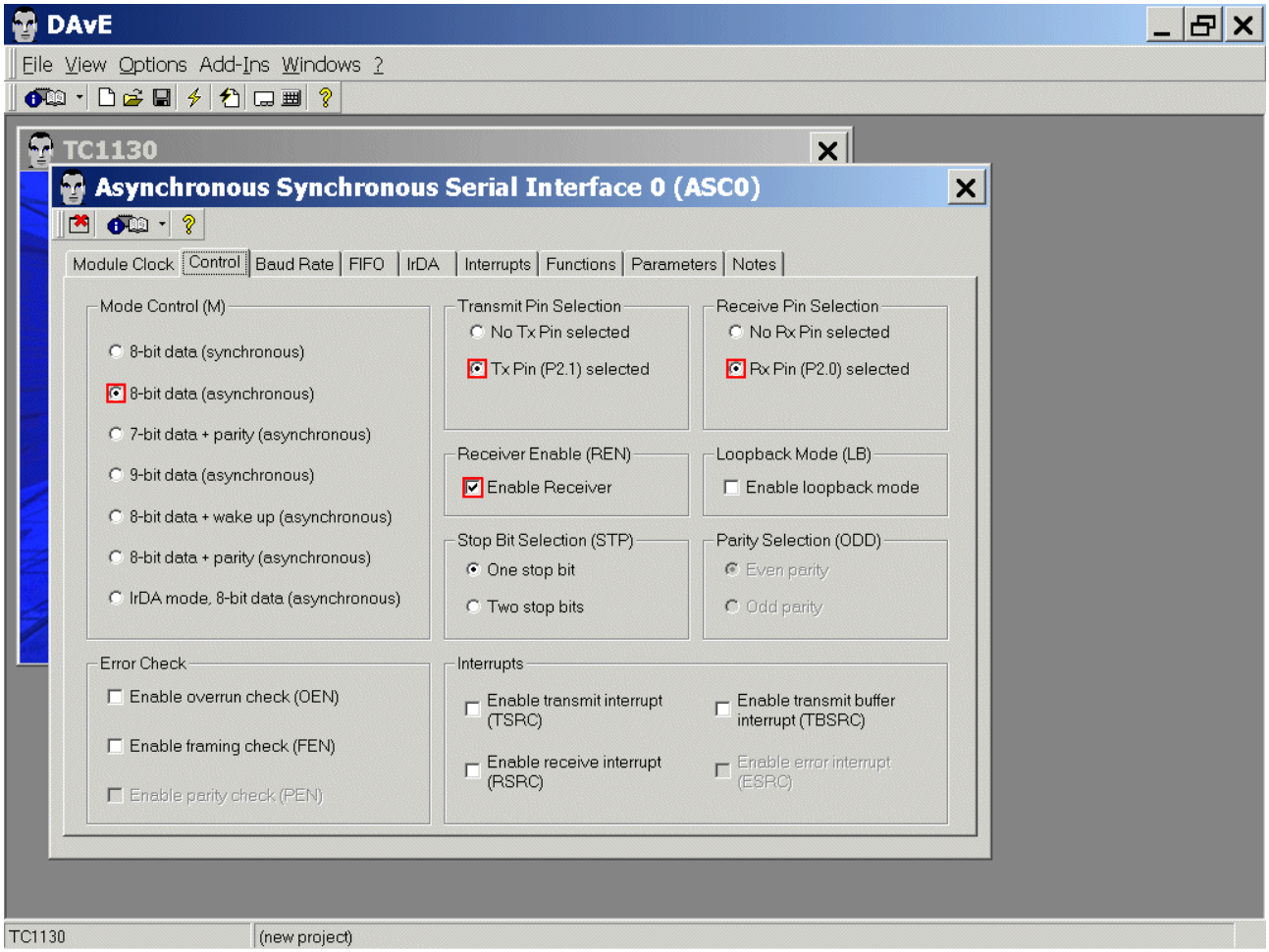
The configuration window can be opened by clicking the specific block/module.



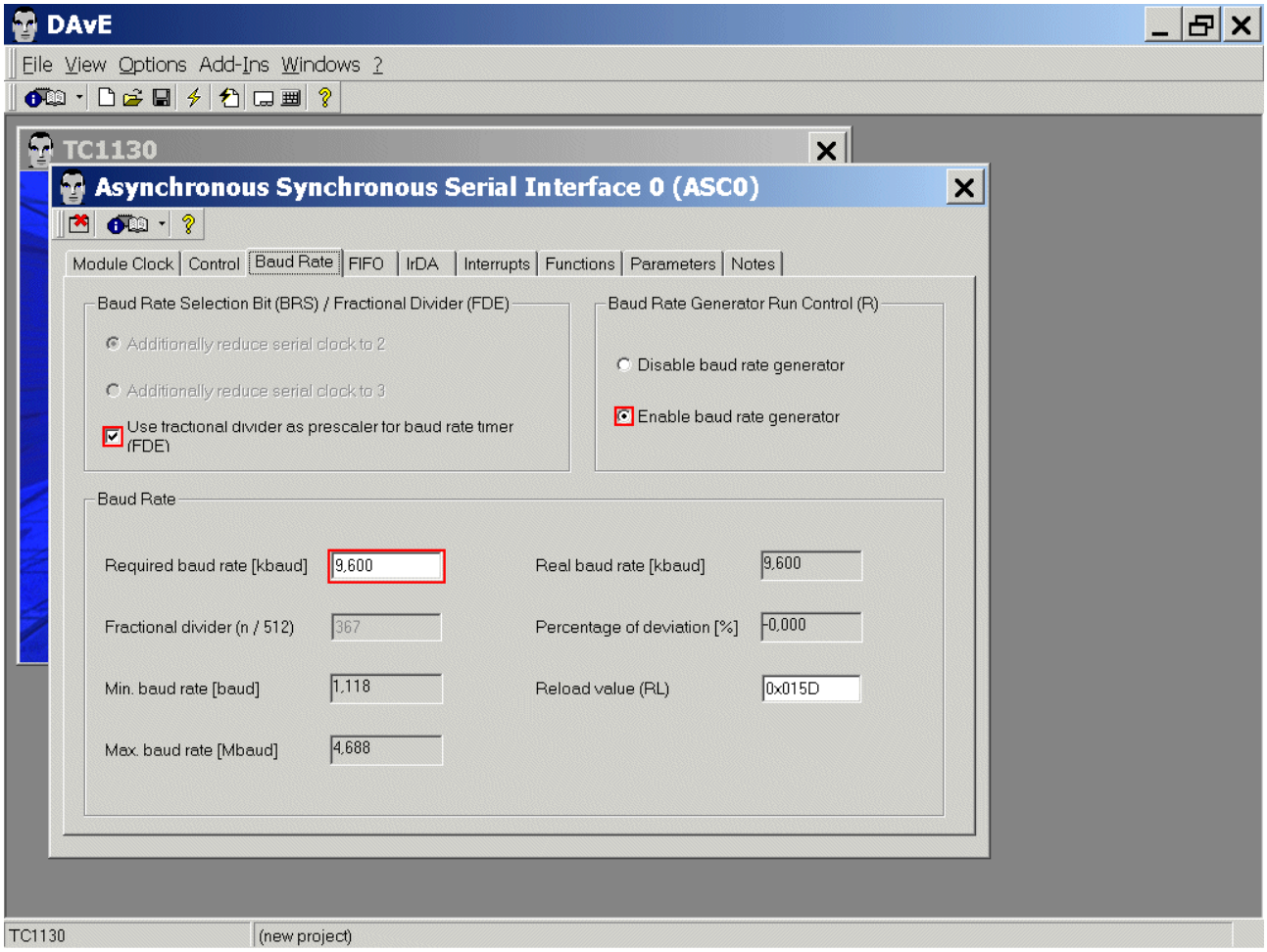
Module Clock: Module Disable Request: select/check ☐ Disable the ASC0 module
Module Clock: Module Run Mode Clock Control: choose System clock/1 (=75,0000 MHz)
Module Clock: Sleep Mode Enable Control: click ✓ Disable the sleep mode



Control: Mode Control (M): click ⊙ 8-bit data (asynchronous)
Control: Transmit Pin Selection: click ⊙ TxPin (P2.1) selected
Control: Receive Pin Selection: click ⊙ RxPin (P2.0) selected
Control: Receiver Enable: click ✓ Enable receiver (REN)

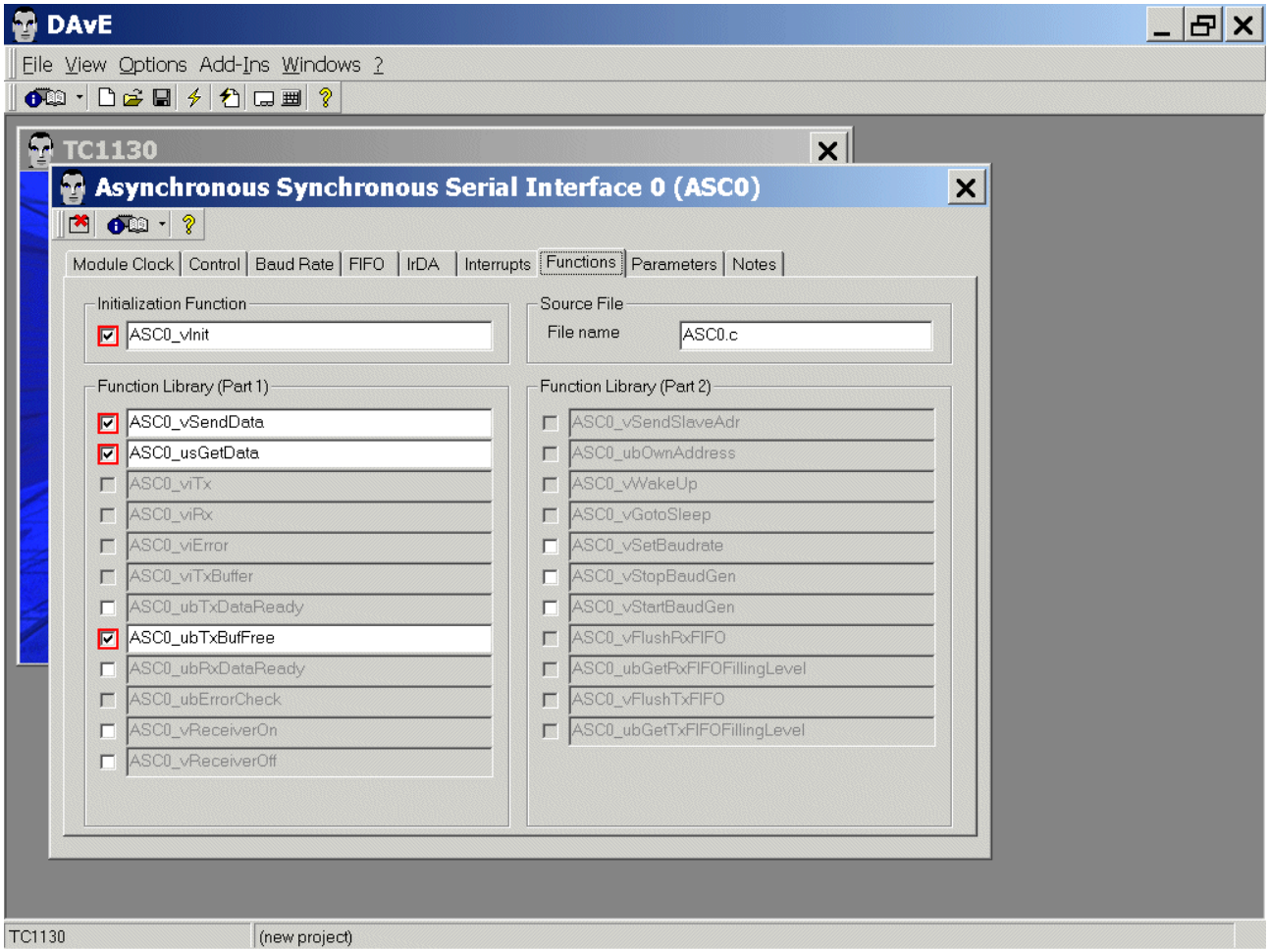


Baud Rate: Baud Rate: Required baud rate [kBaud] insert 9,600
Baud Rate: Baud Rate Selection Bit / Fractional Divider: click ✓ Use fractional divider
Baud Rate: Baud Rate Generator Run Control (R): click ✓ Enable baud rate generator



FIFO: (do nothing)
IrDA: (do nothing)
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

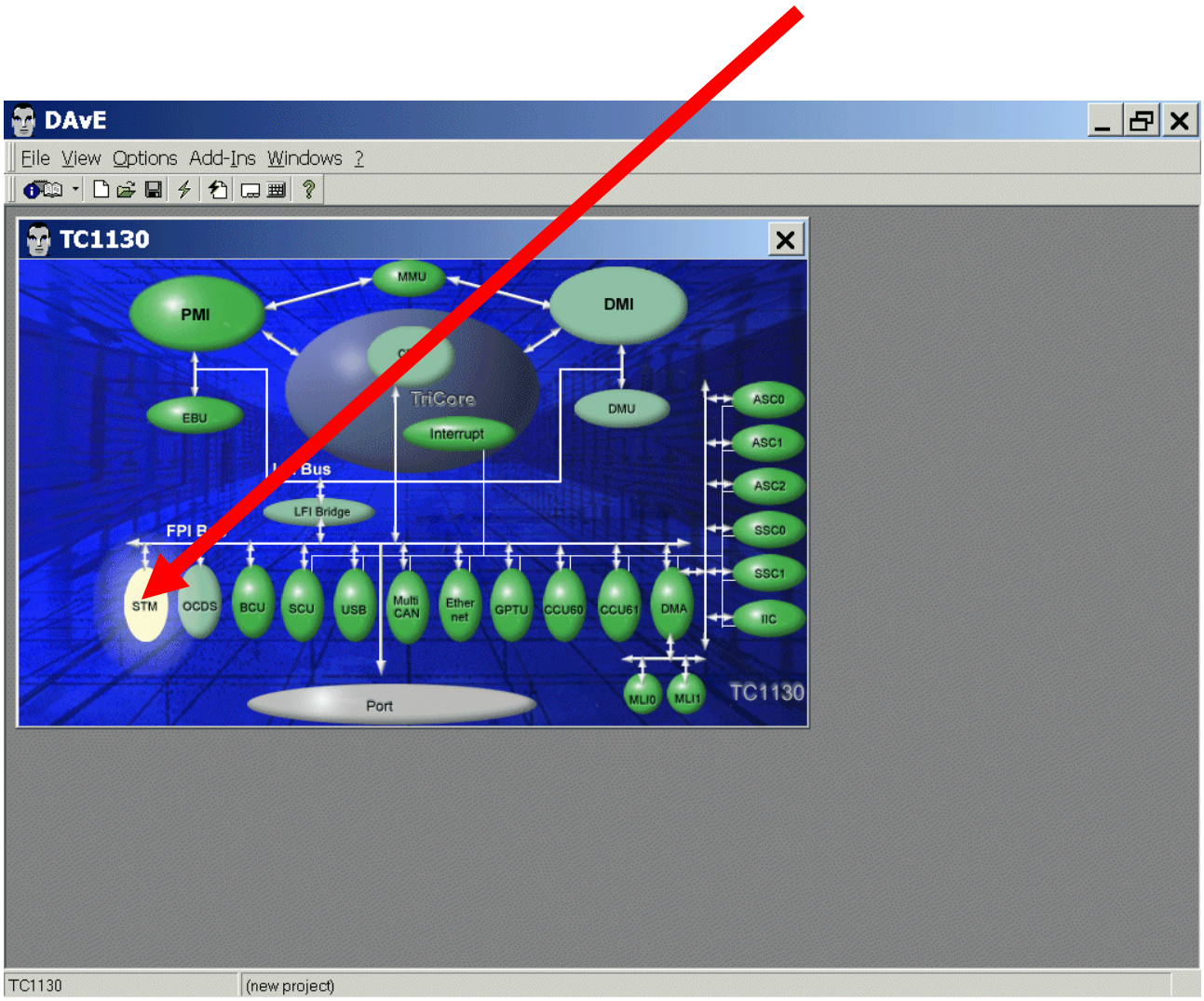


Parameters: (do nothing)
Notes: If you wish, you can insert your comments here.

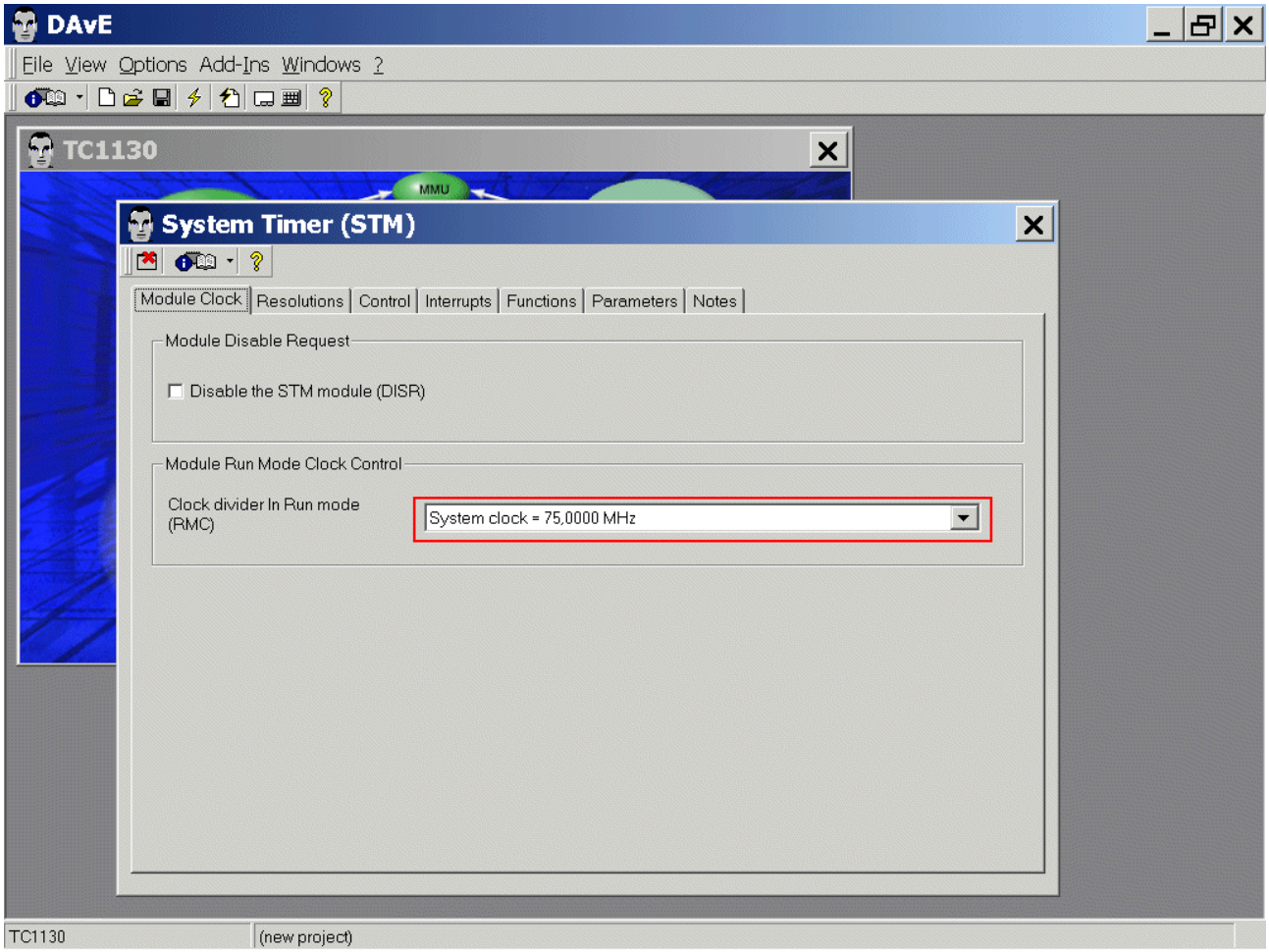
Exit this dialog now by clicking  the close button.

Configuration of the STM:

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

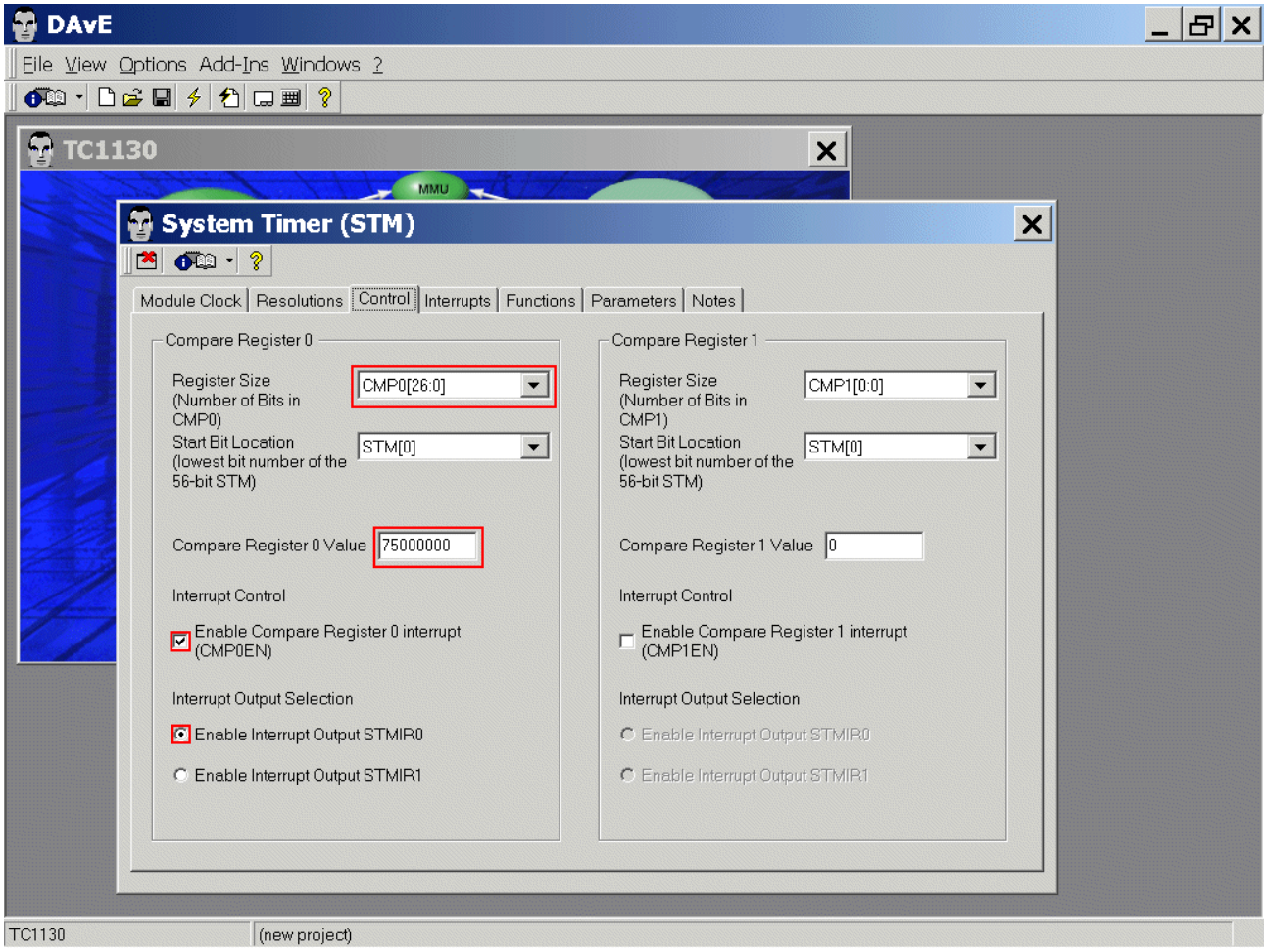


Module Clock: Module Run Mode Clock Control: select System clock = 75 MHz (= 13,3 ns)



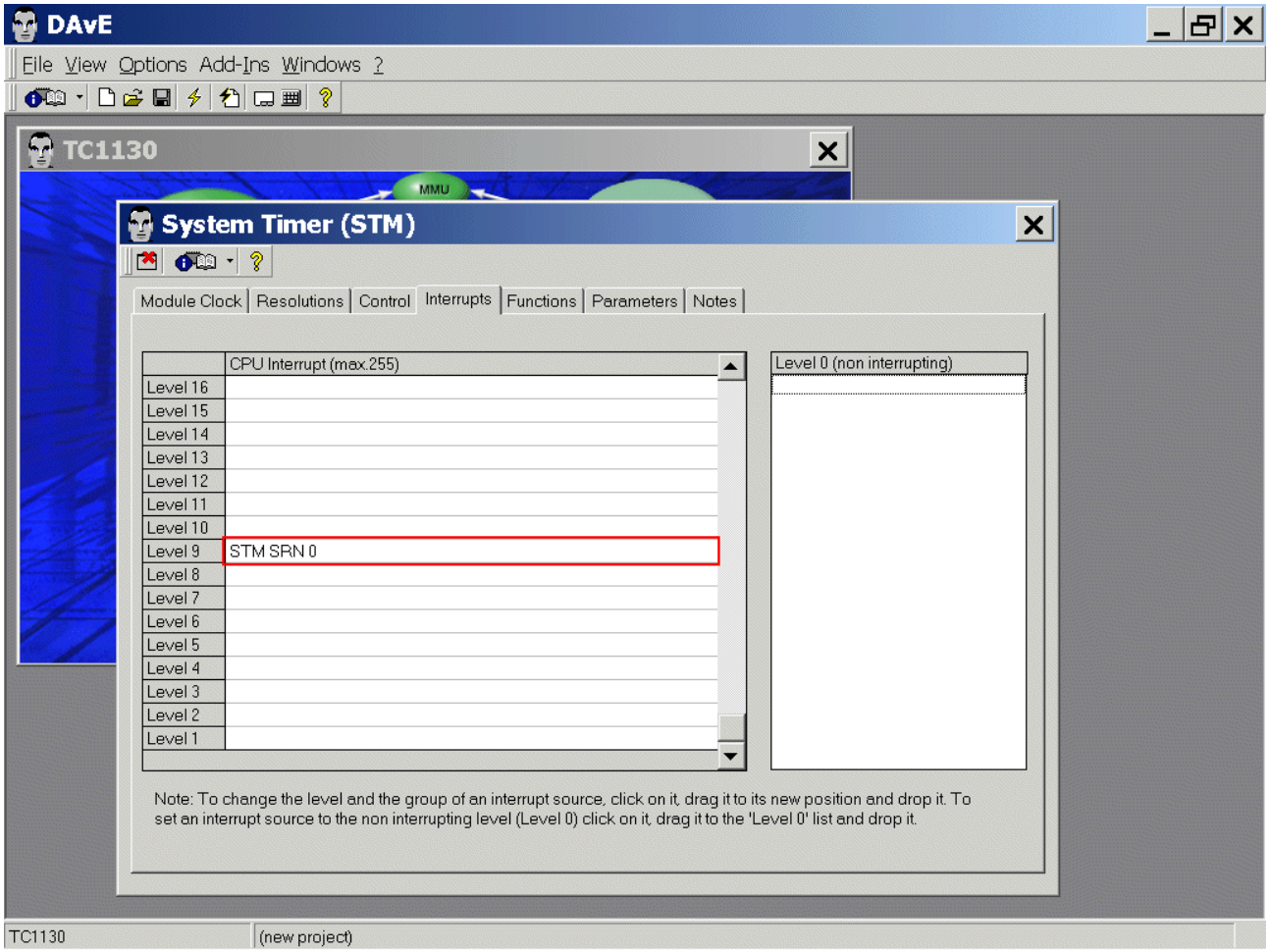
Resolutions: (do nothing)

Control: Compare Register 0: Register Size: select CMP0[26:0]
Control: Compare Register 0: Compare Register 0 Value: insert 75000000
Control: Compare Register 0: Interrupt Control: click ✓ Enable Compare Register 0 interrupt
Control: Compare Register 0: Interrupt Output Selection: click ⊙ Enable Interrupt Output

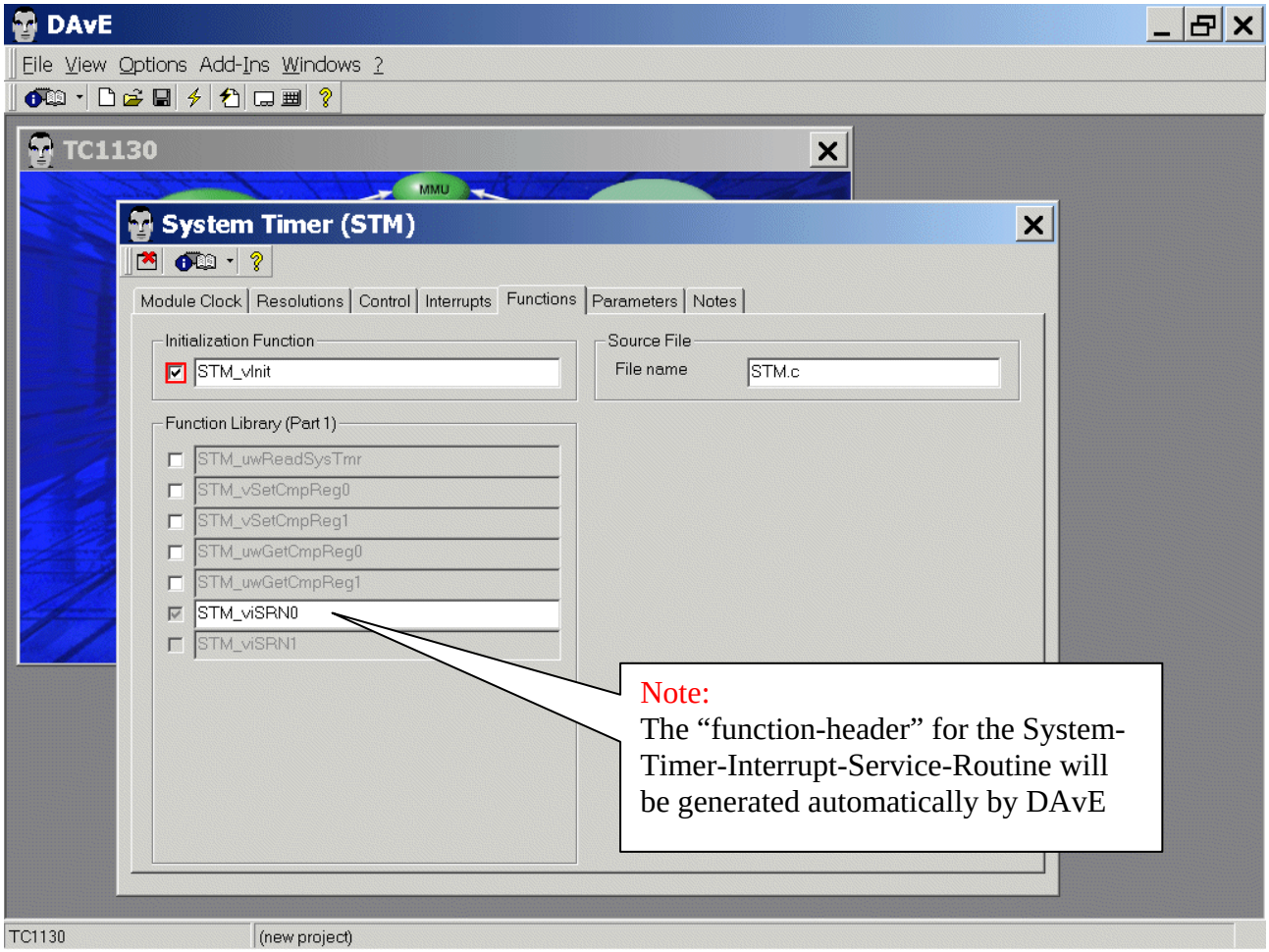


Note:
75.000.000 * 13,3333 ns = 1 s
There will be a System-Timer-Interrupt every 1 second.
The LED on IO_Port_0_Pin_7 will be blinking at a frequency of 1 second (done in the System-Timer-Interrupt-Service-Routine).

Interrupts: drag and drop STM SRN 0 from Level 0 (non interrupting) to Level 9



Functions: Initialization Function: click ✓ STM_vInit



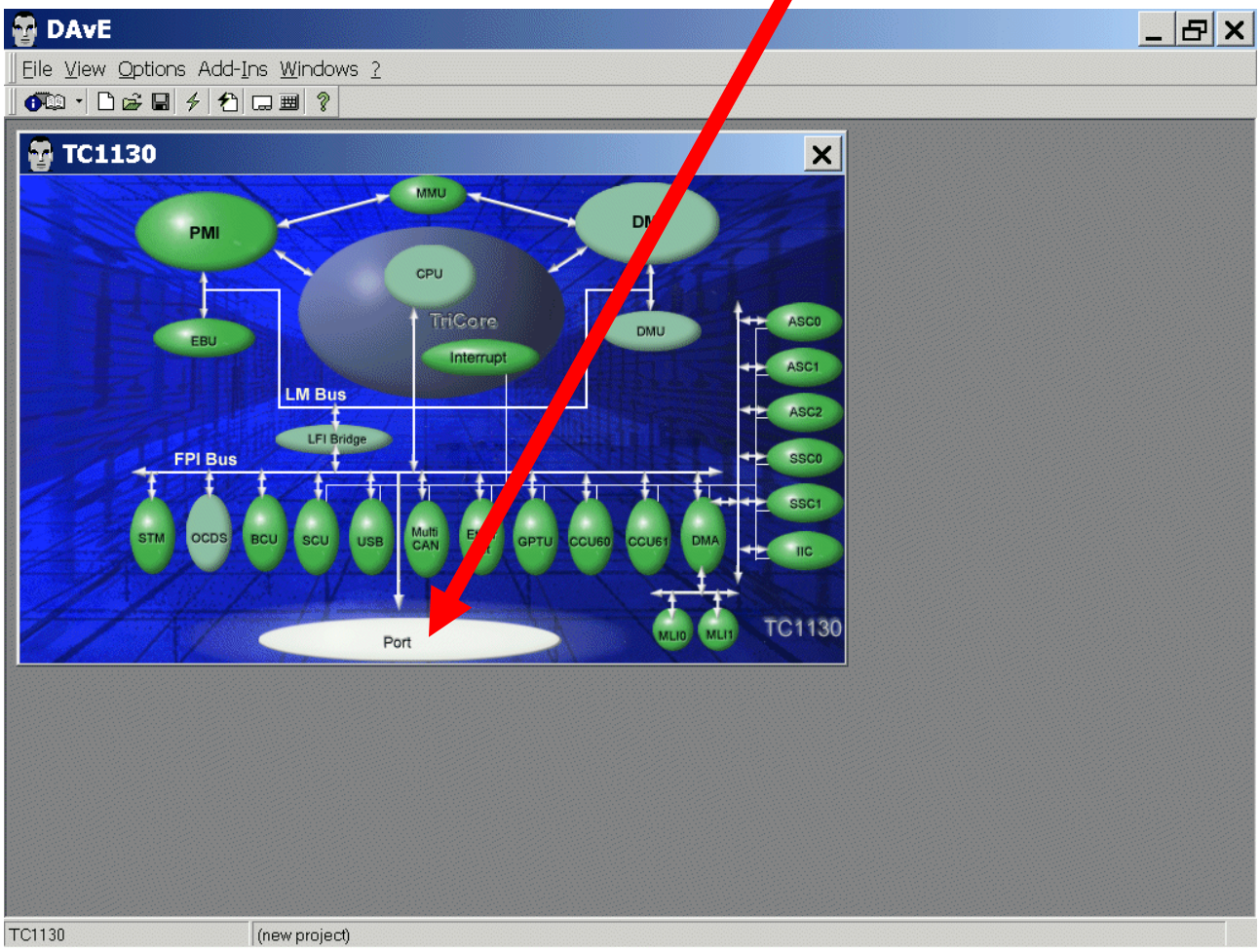
Parameters: (do nothing)

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

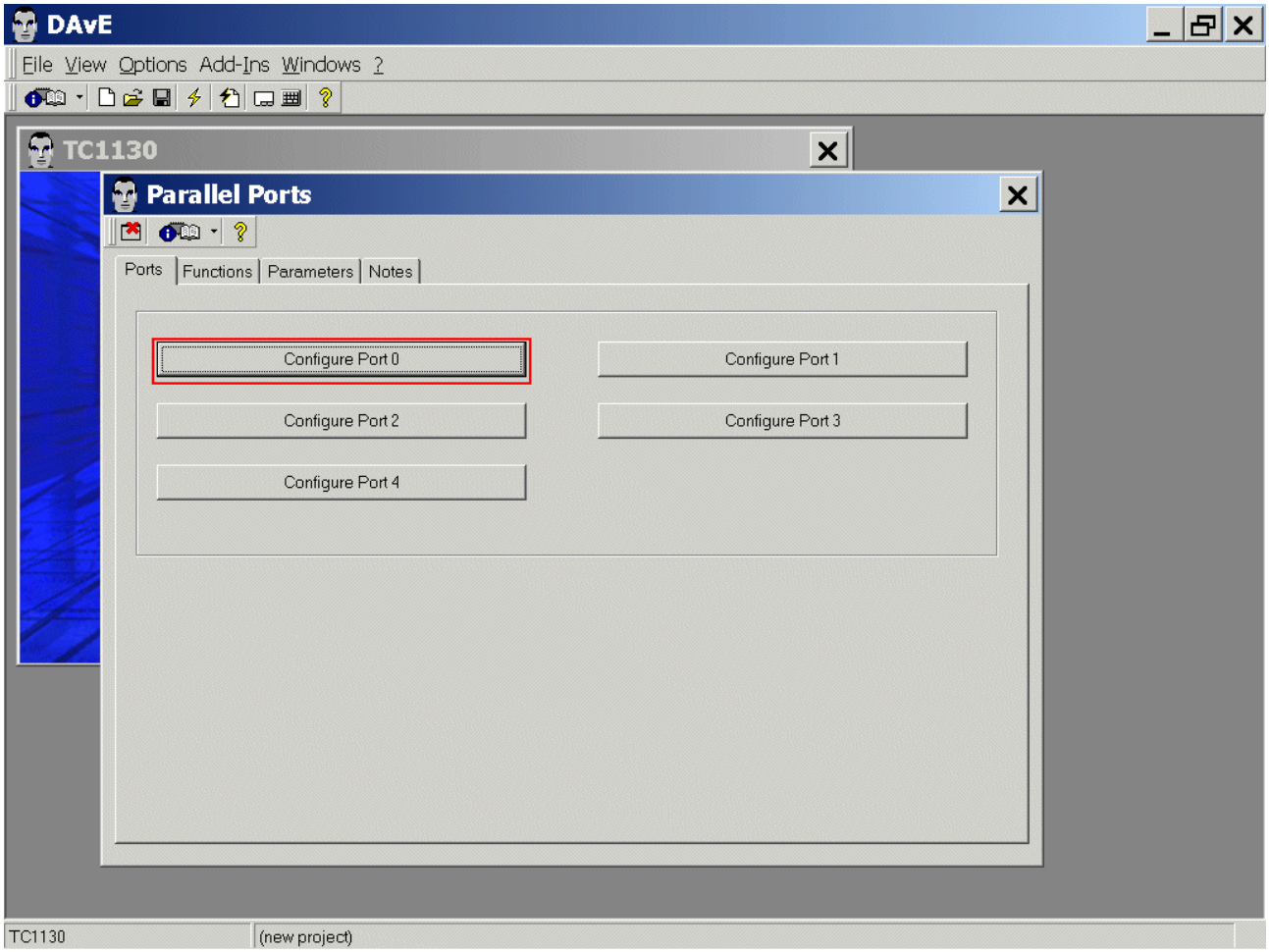
Exit this dialog now by clicking  the close button.

Port Configuration:

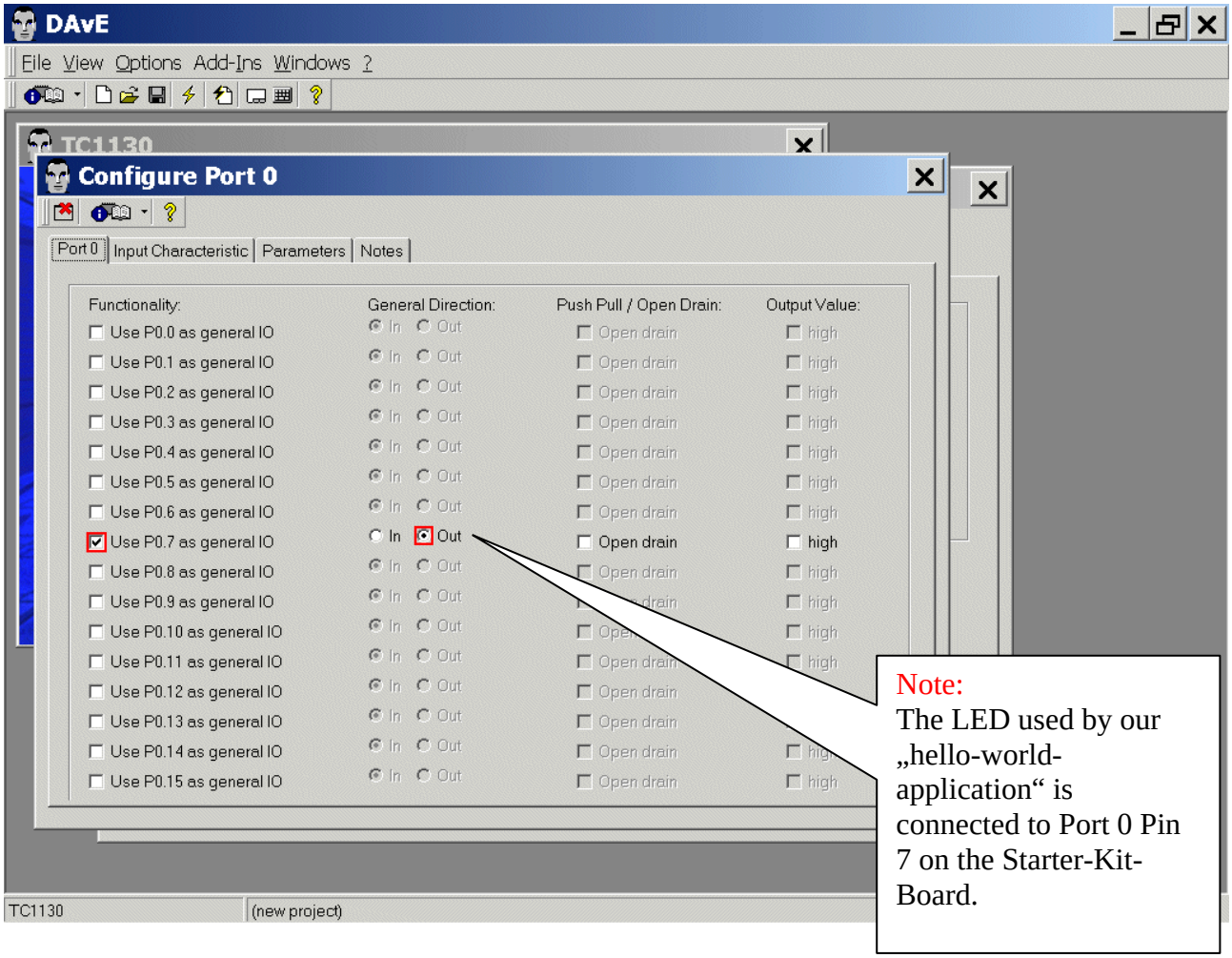
The configuration window can be opened by clicking the specific block/module.



Ports: click Configure Port 0



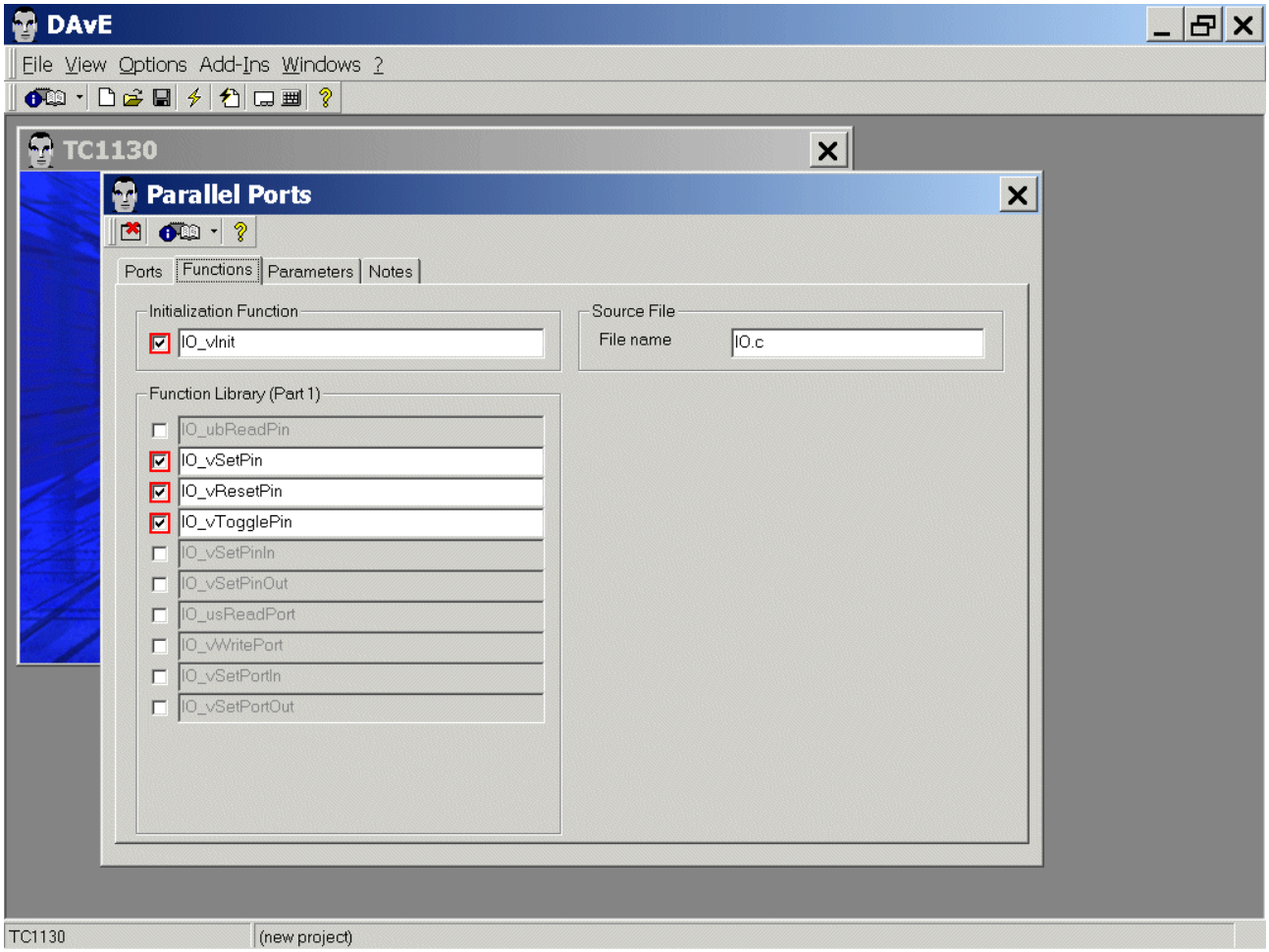
Ports: Configure Port 0
Port 0: Functionality: click ✓ Use P0.7 as general IO, General Direction: click ⊙ Out



Input 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



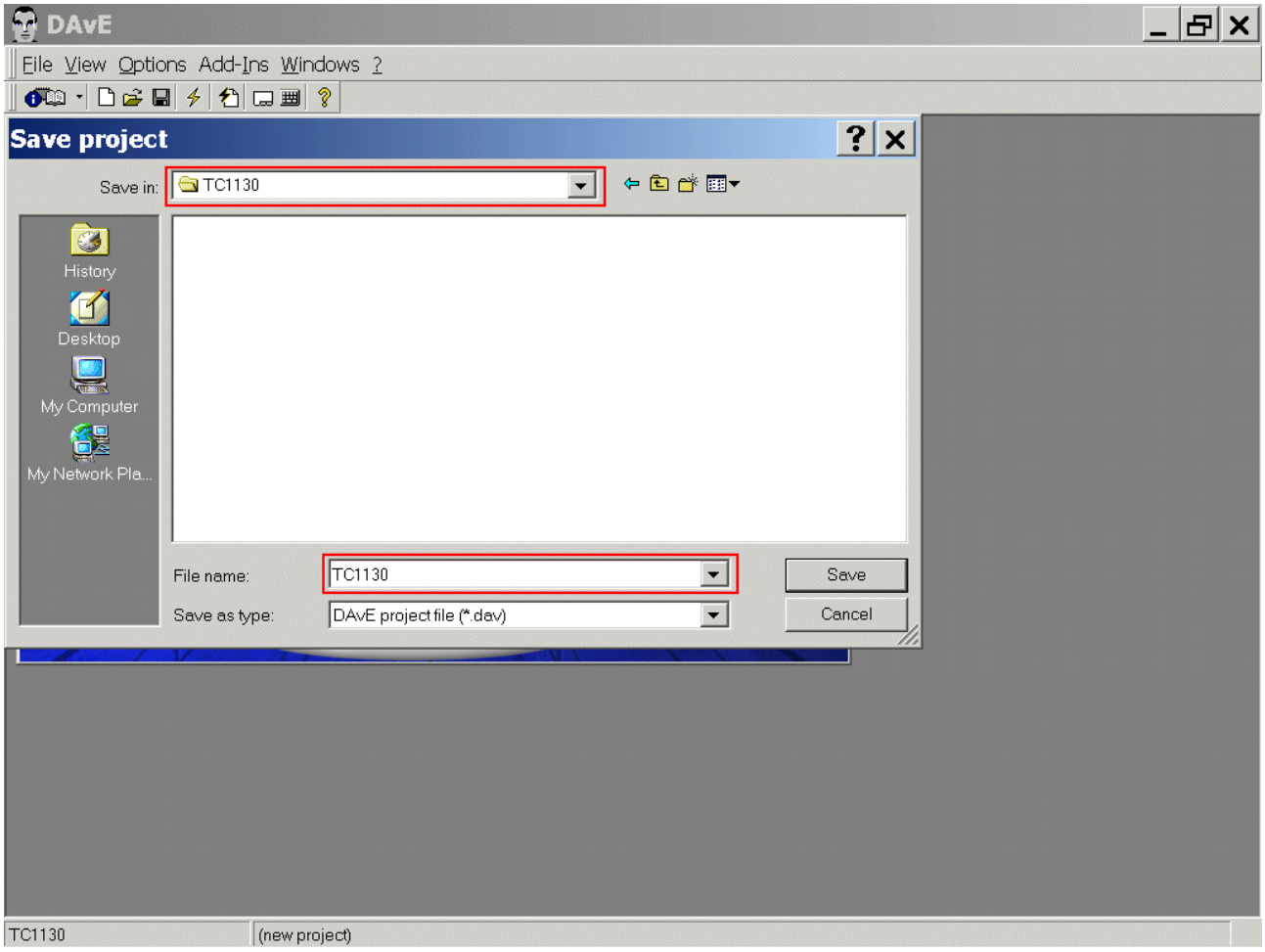
Parameters : (do nothing)
Notes: If you wish, you can insert your comments here.

Exit this dialog now by clicking  the close button.

Save the project:

File
Save

Save project: Save in: C:\TC1130 (create directory)
File name: TC1130

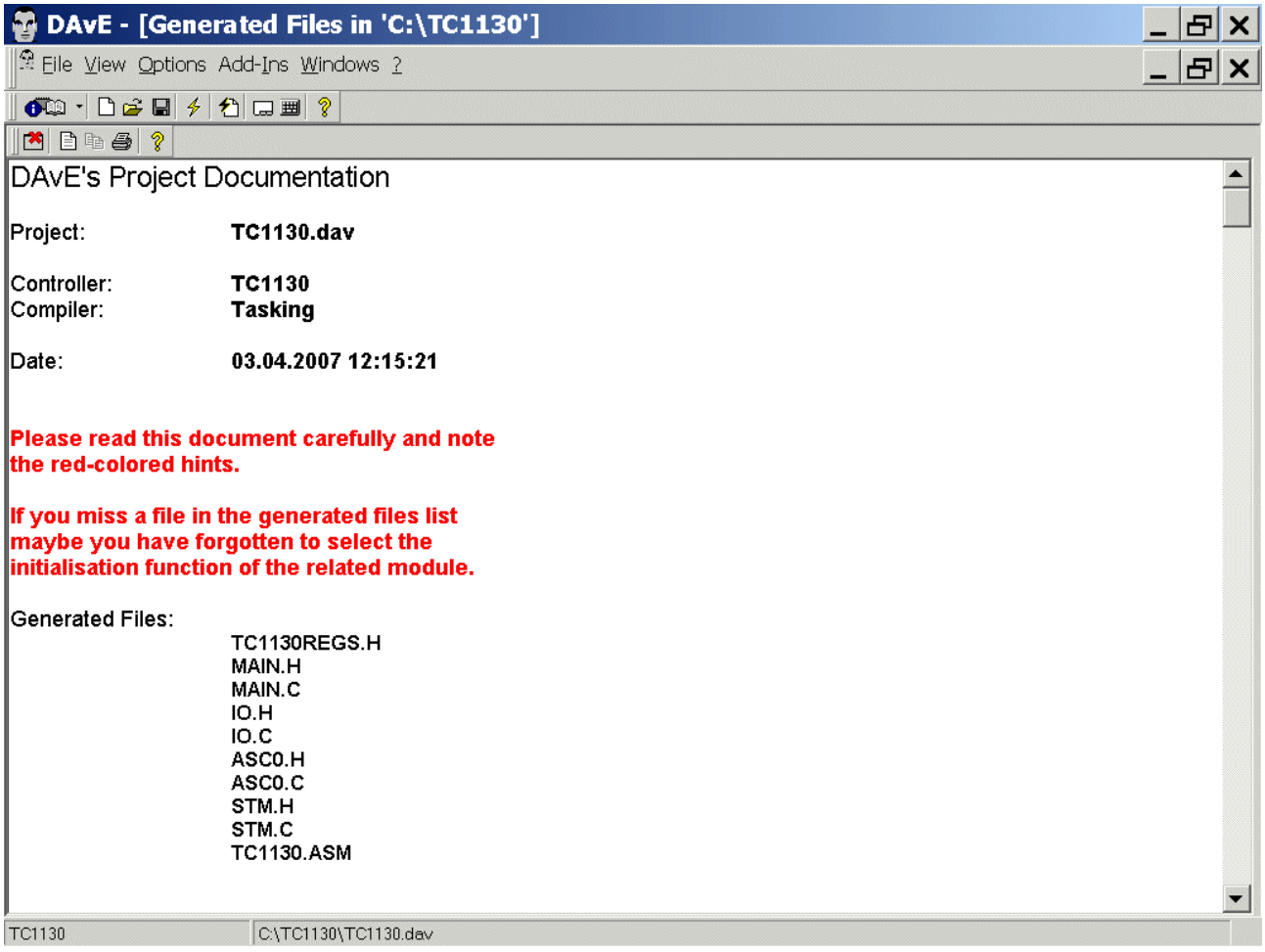


Save

Generate Code:

File Generate Code	or click 

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

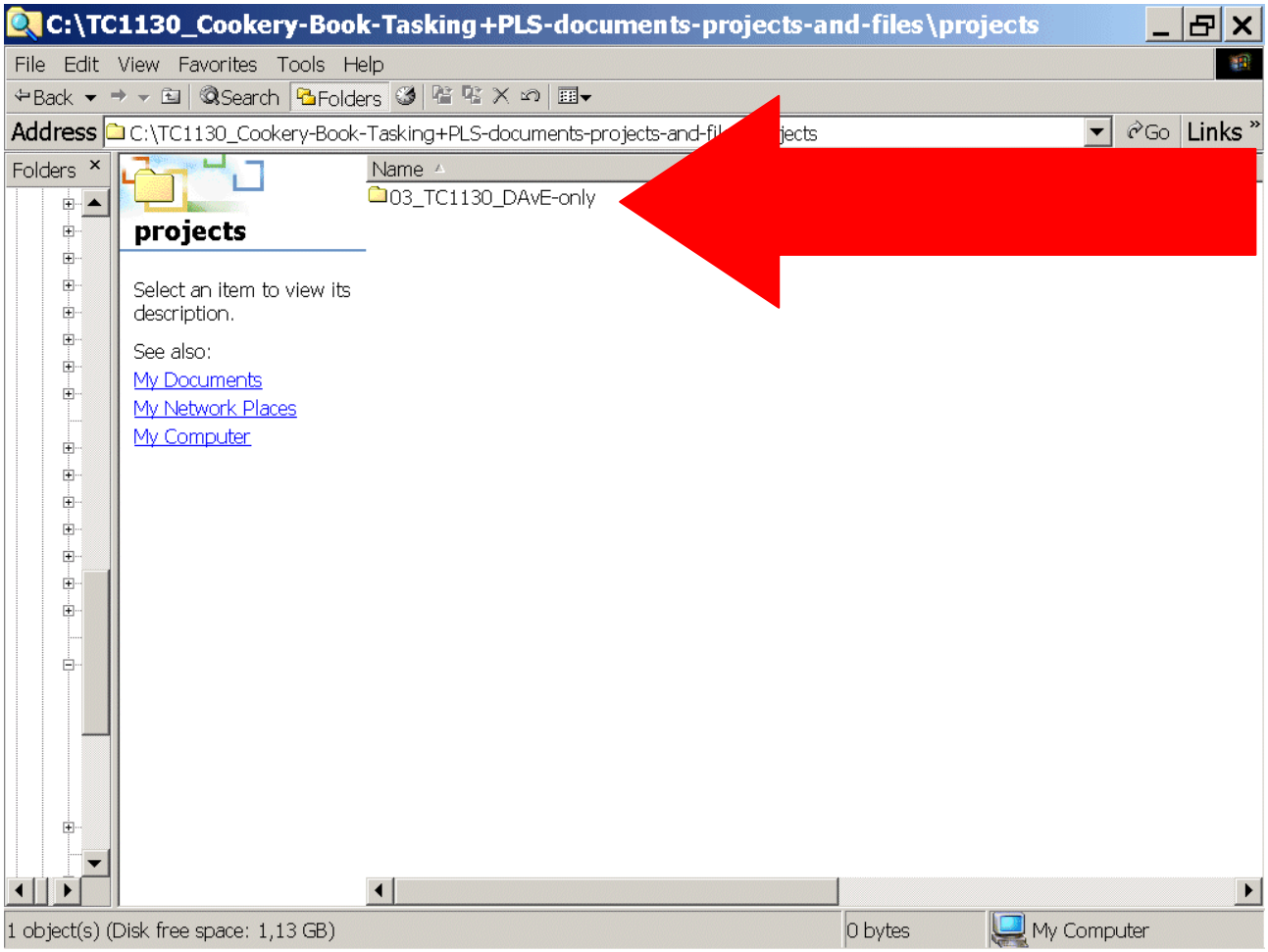


File
Exit

Save changes?

click Yes

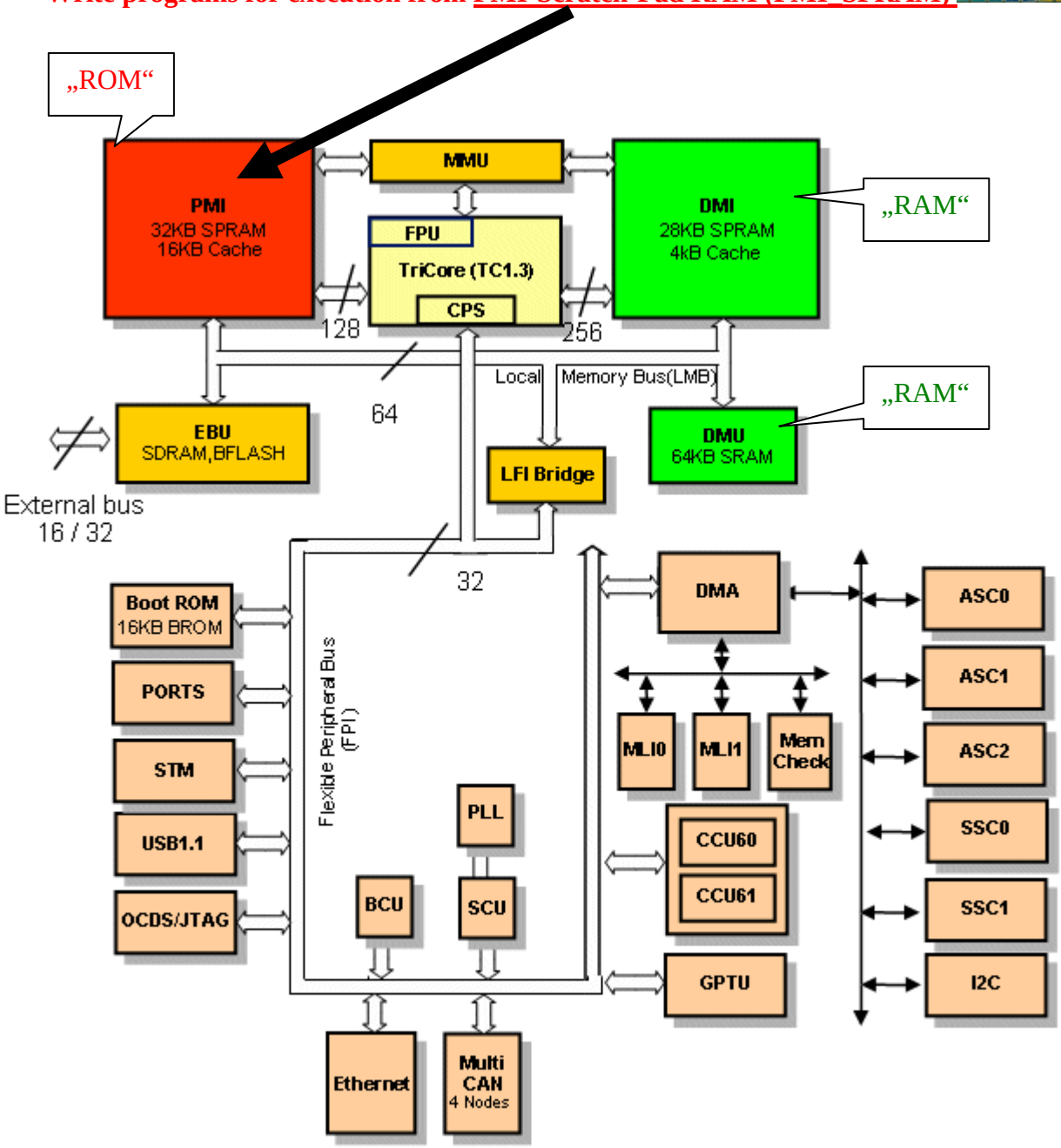
We recommend now to copy and store your project-directory “C:\TC1130” to “03_TC1130_DAvE-only”.



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

"ROM": Locating programs into the 32 KBytes code scratchpad RAM (PMI_SPRAM),
"RAM": Using OnChipSRAM (28 KBytes DMI_SPRAM + 64 KBytes DMU_SRAM)

Write programs for execution from PMI-Scratch-Pad RAM (PMI_SPRAM)



Note:

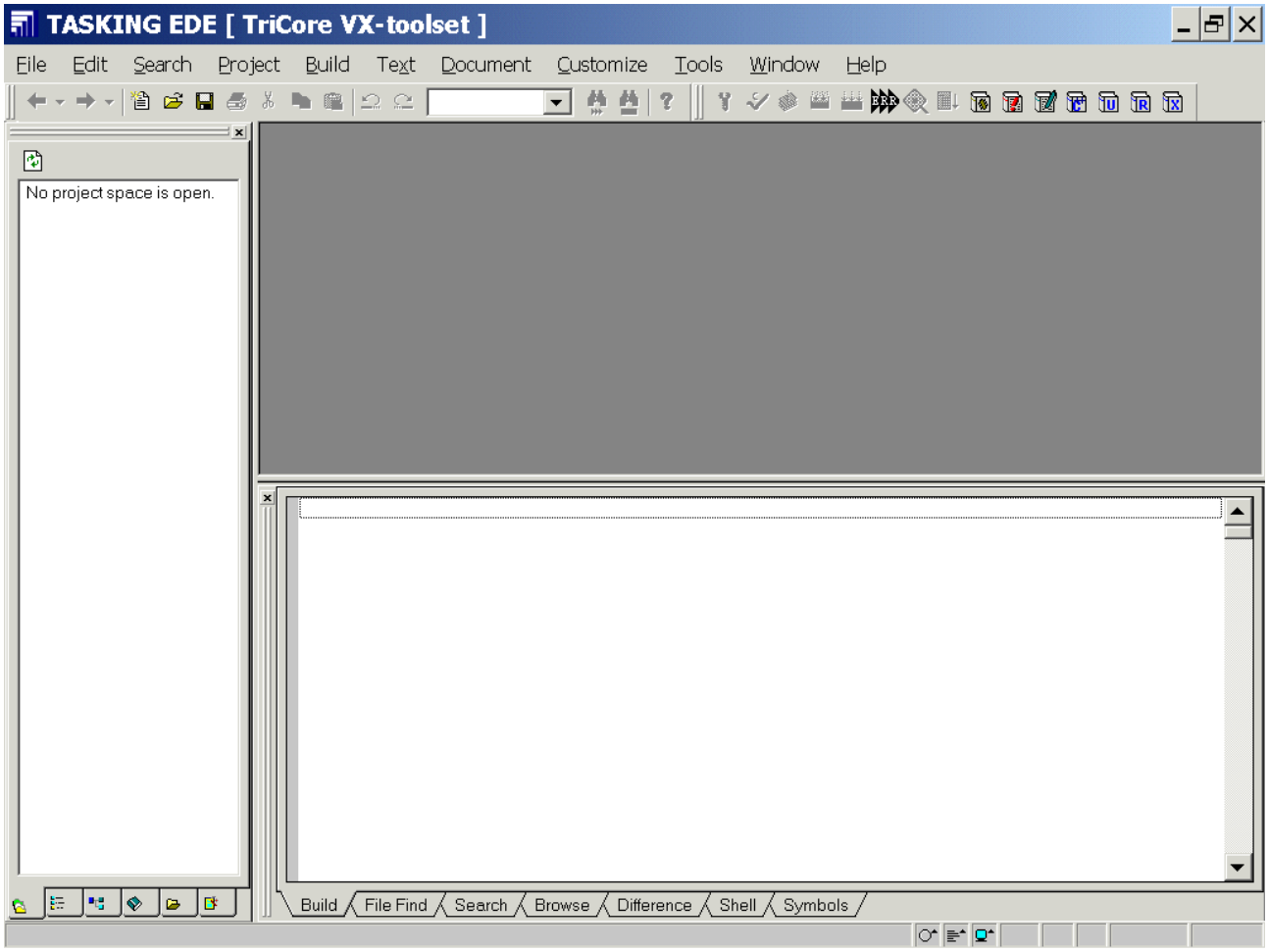
“ROM”:
Program, constant data and initialization-values for variables are located in „ROM-space“.

“RAM”:
Variables are located in “RAM-space”.

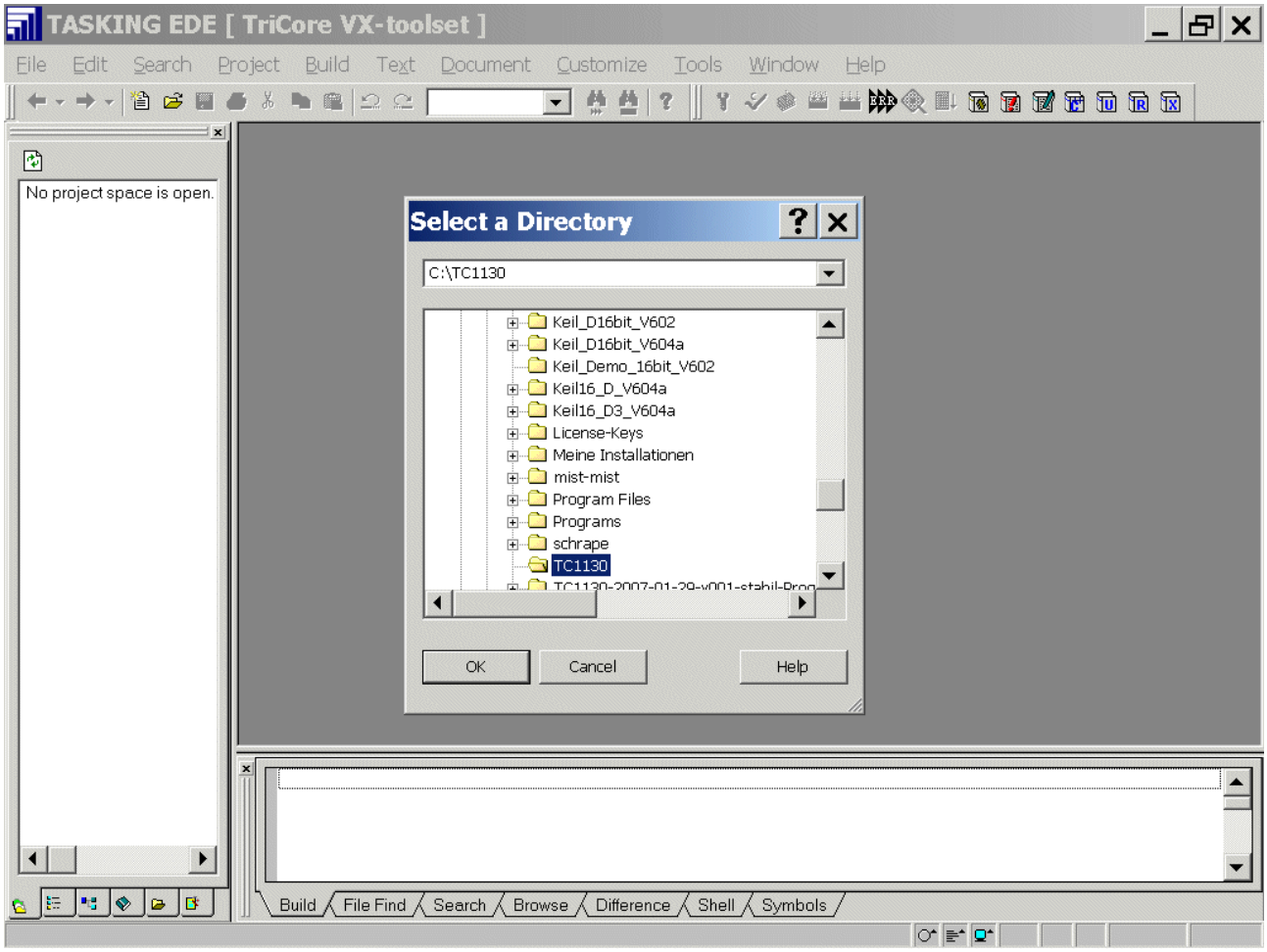
Install the Tasking Development Tools TriCore v2.3r1

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

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

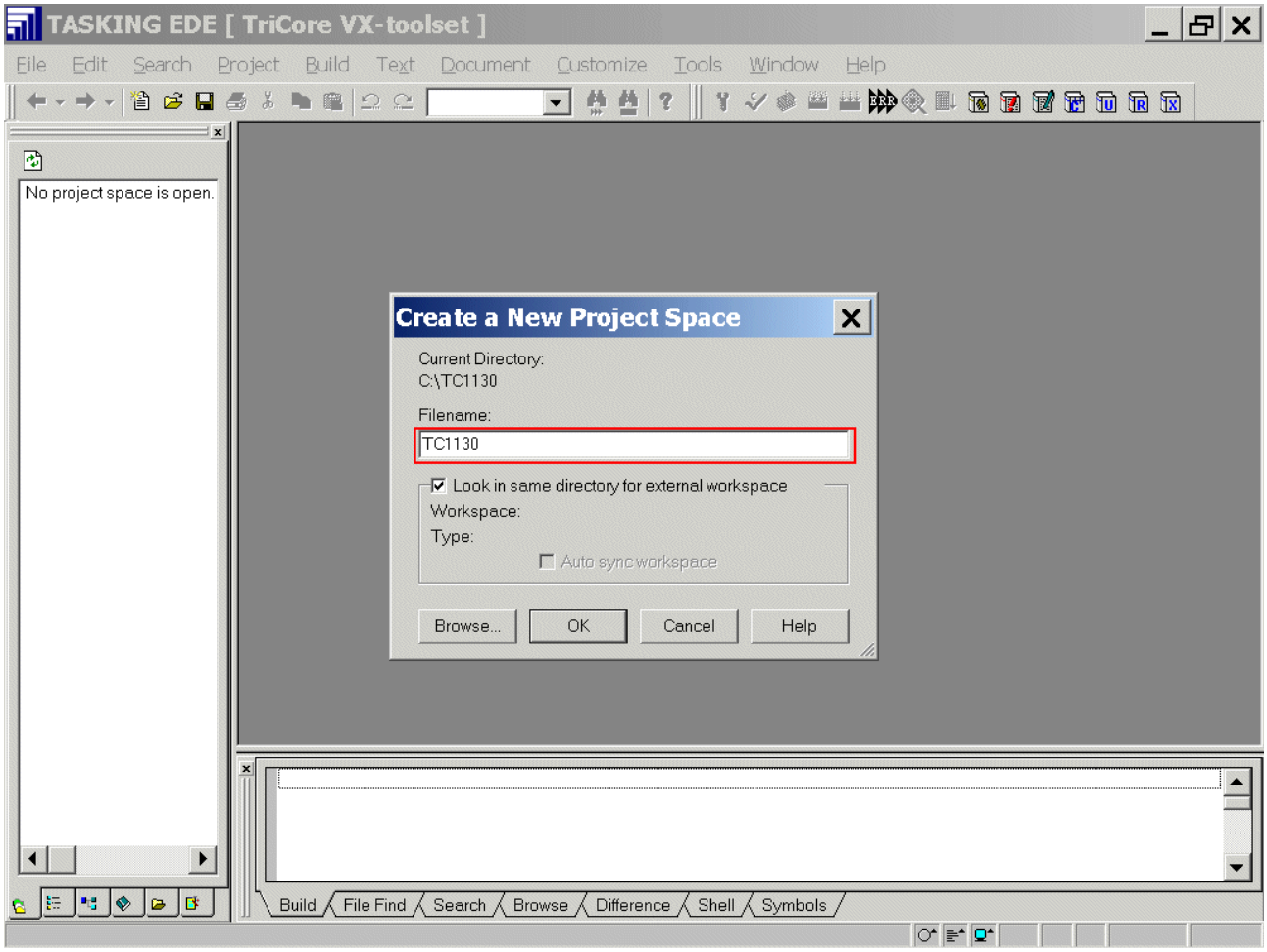


File
Change Directory
Choose C:\TC1130



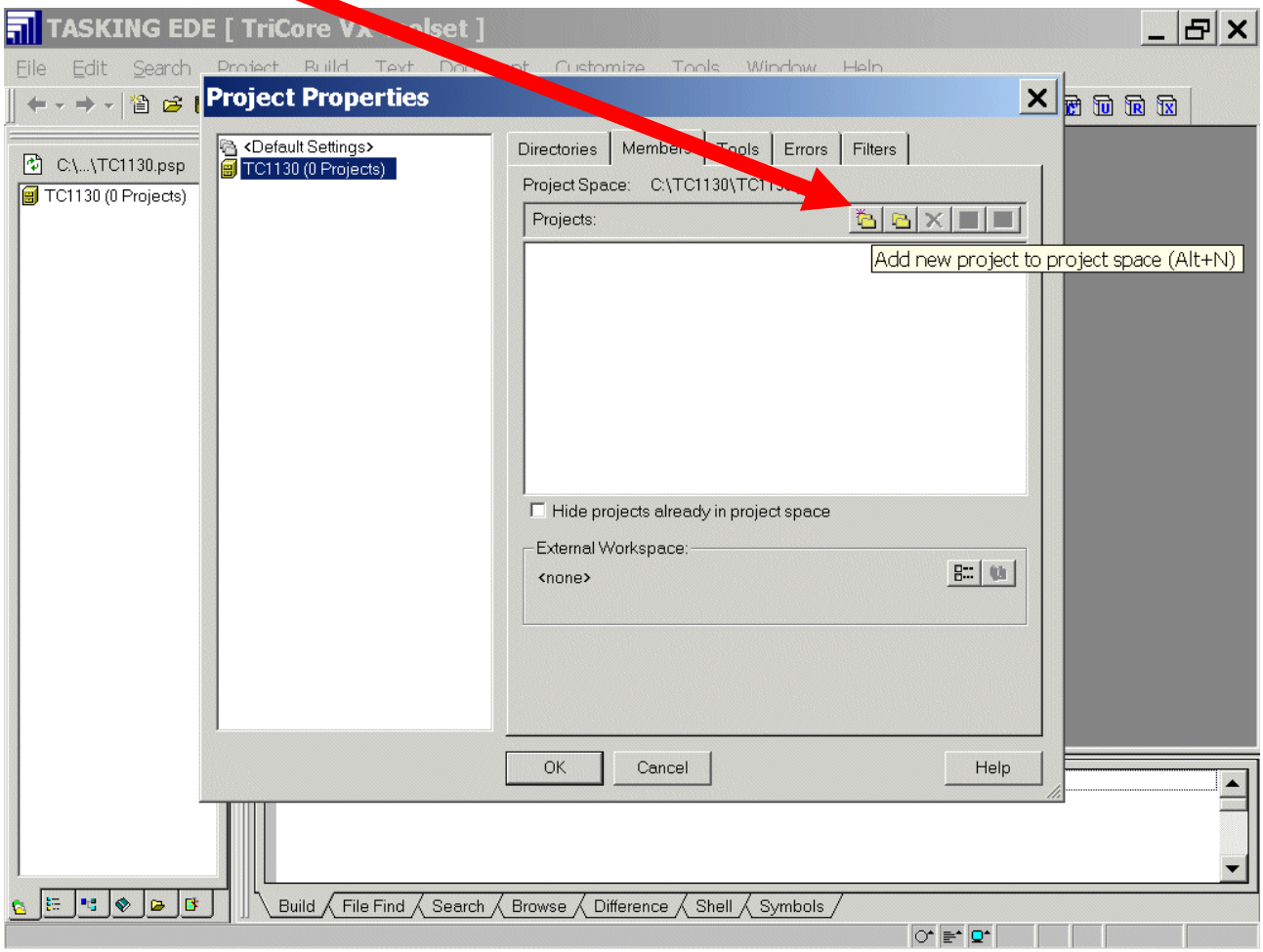
OK

File
New Project Space
Insert TC1130

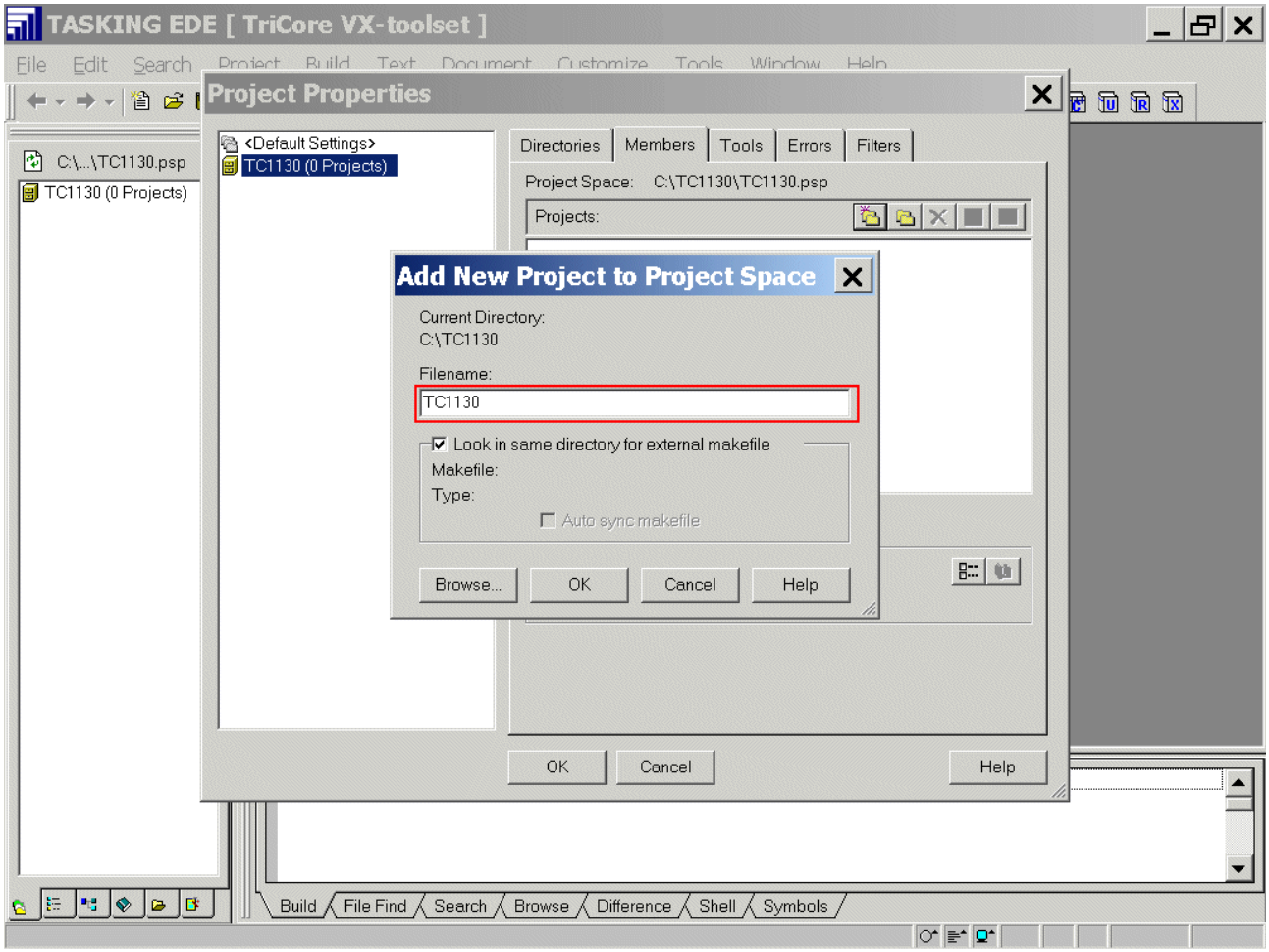


OK

Click: “Add new project to project space”

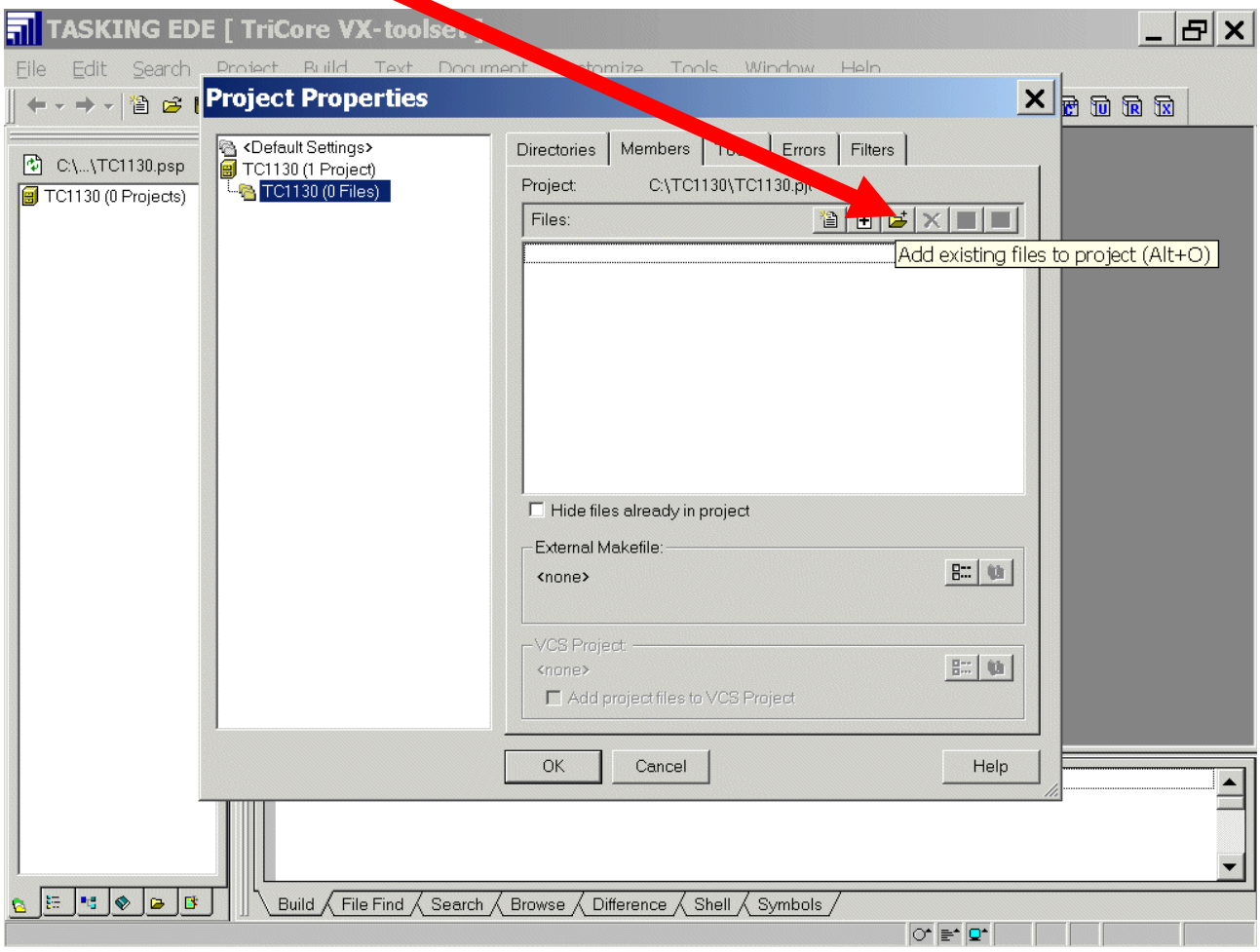


Insert TC1130

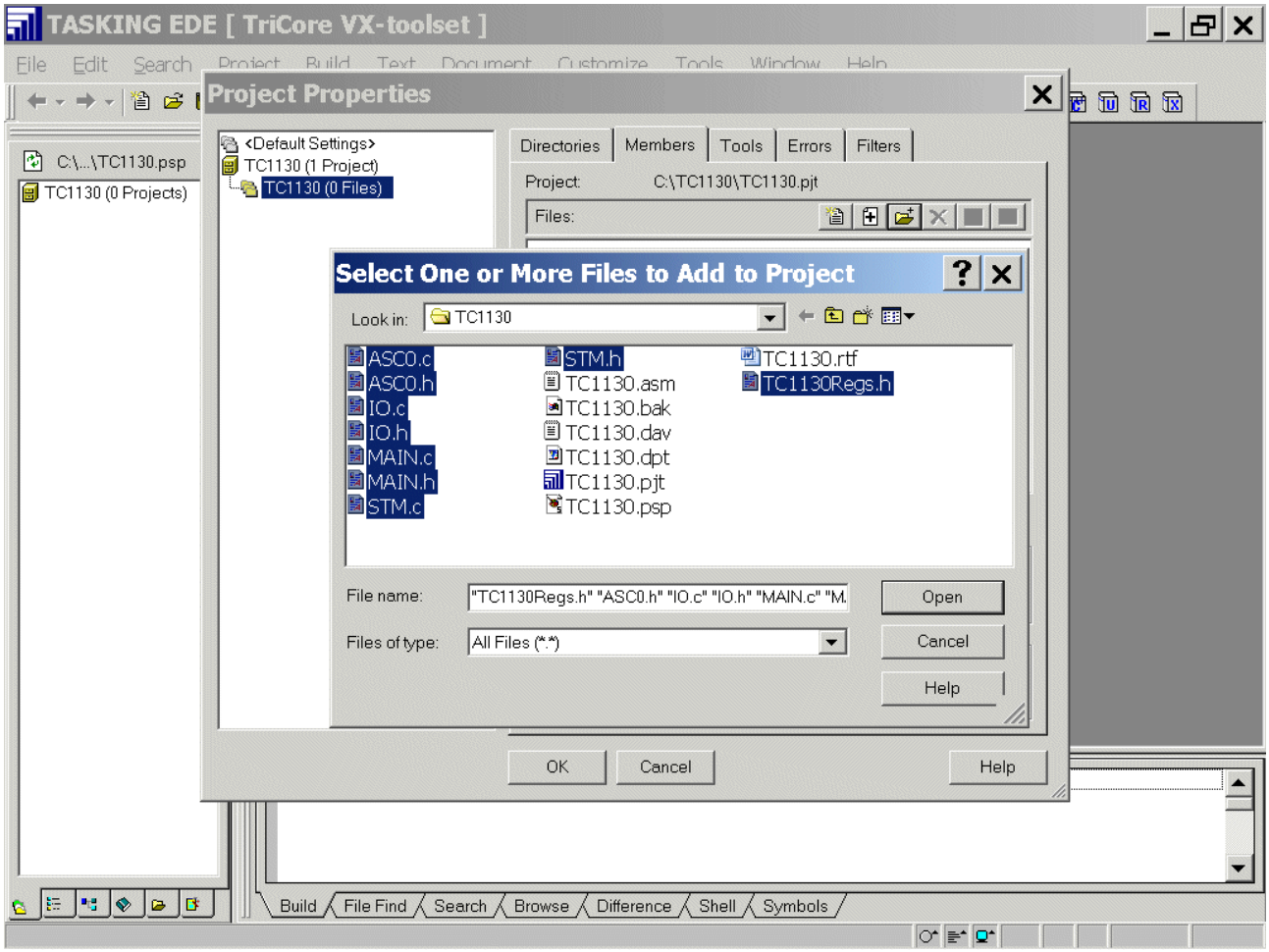


OK

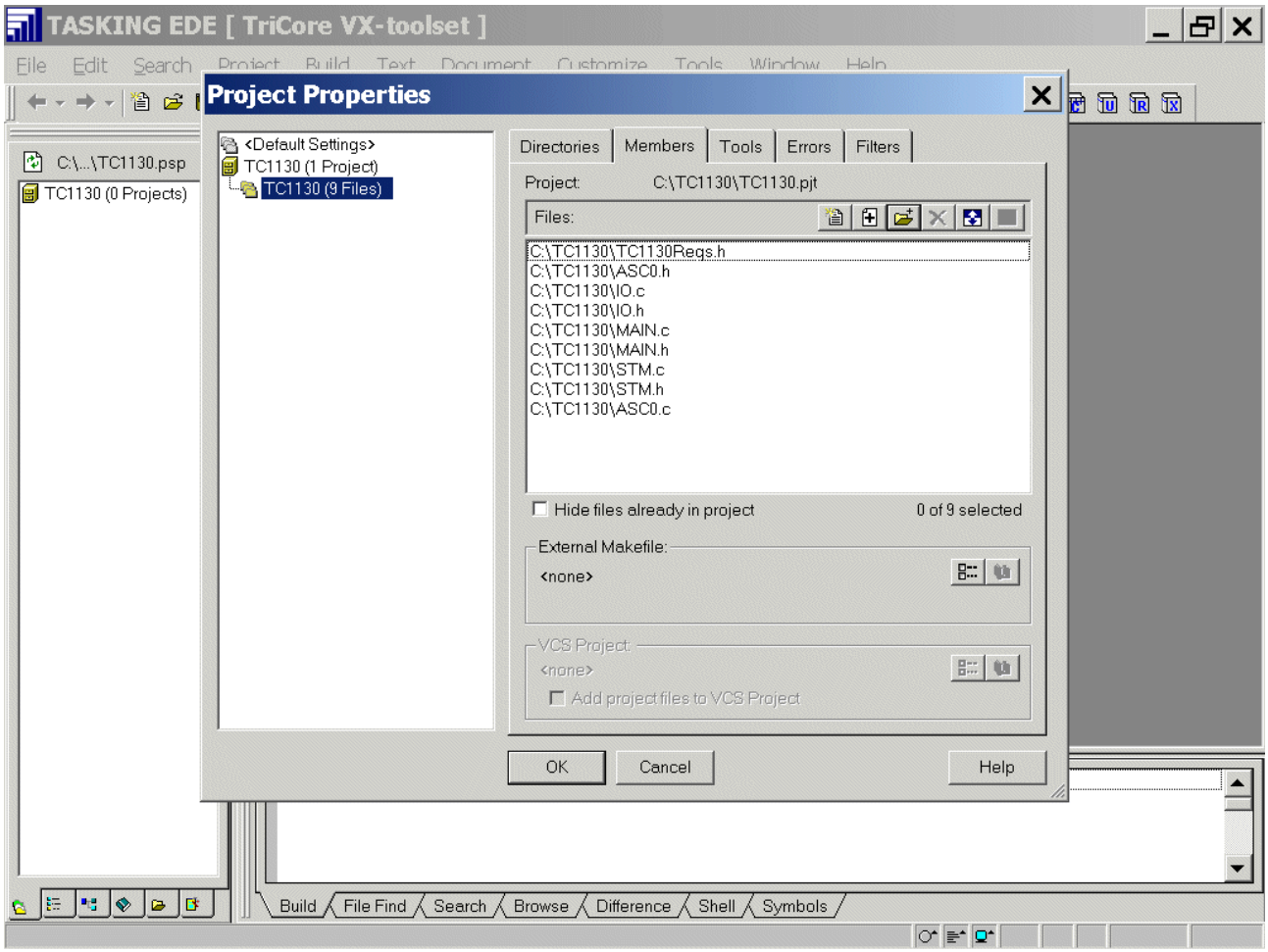
Click: “Add existing files to project”



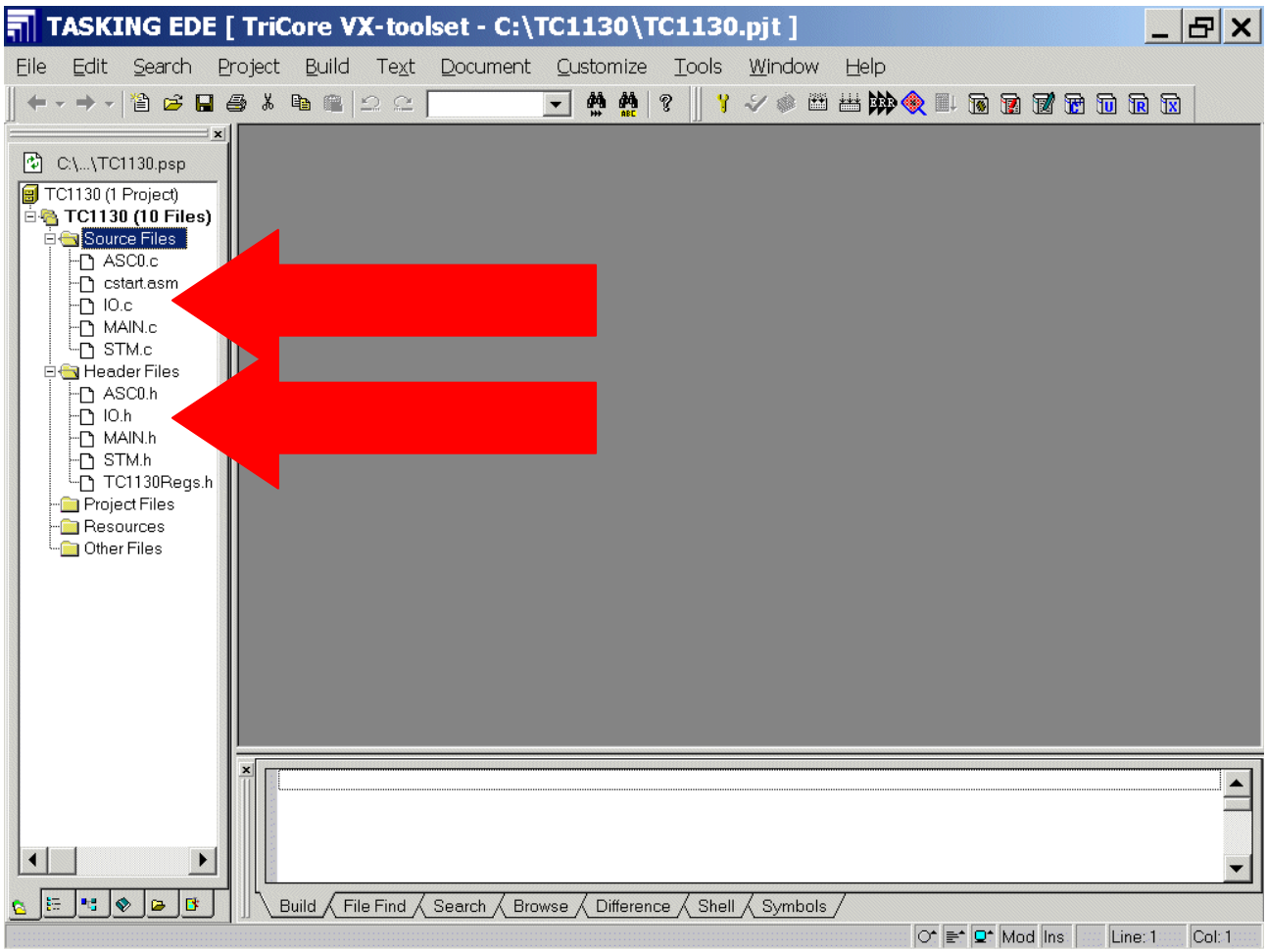
Select ASC0.c
Select ASC0.h
Select IO.c
Select IO.h
Select MAIN.c
Select MAIN.h
Select STM.c
Select STM.h
Select TC1130Regs.h



Open


TASKING EDE [TriCore VX-toolset]
File Edit Search Project Build Text Document Customize Tools Window Help
Project Properties
<Default Settings>
TC1130 (1 Project)
TC1130 (9 Files)
Directories Members Tools Errors Filters
Project: C:\TC1130\TC1130.pjt
Files:
C:\TC1130\TC1130Regs.h
C:\TC1130\ASC0.h
C:\TC1130\IO.c
C:\TC1130\IO.h
C:\TC1130\MAIN.c
C:\TC1130\MAIN.h
C:\TC1130\STM.c
C:\TC1130\STM.h
C:\TC1130\ASC0.c
Hide files already in project
0 of 9 selected
External Makefile:
<none>
VCS Project:
<none>
Add project files to VCS Project
OK
Cancel
Help
C:\...\TC1130.psp
TC1130 (0 Projects)
Build File Find Search Browse Difference Shell Symbols

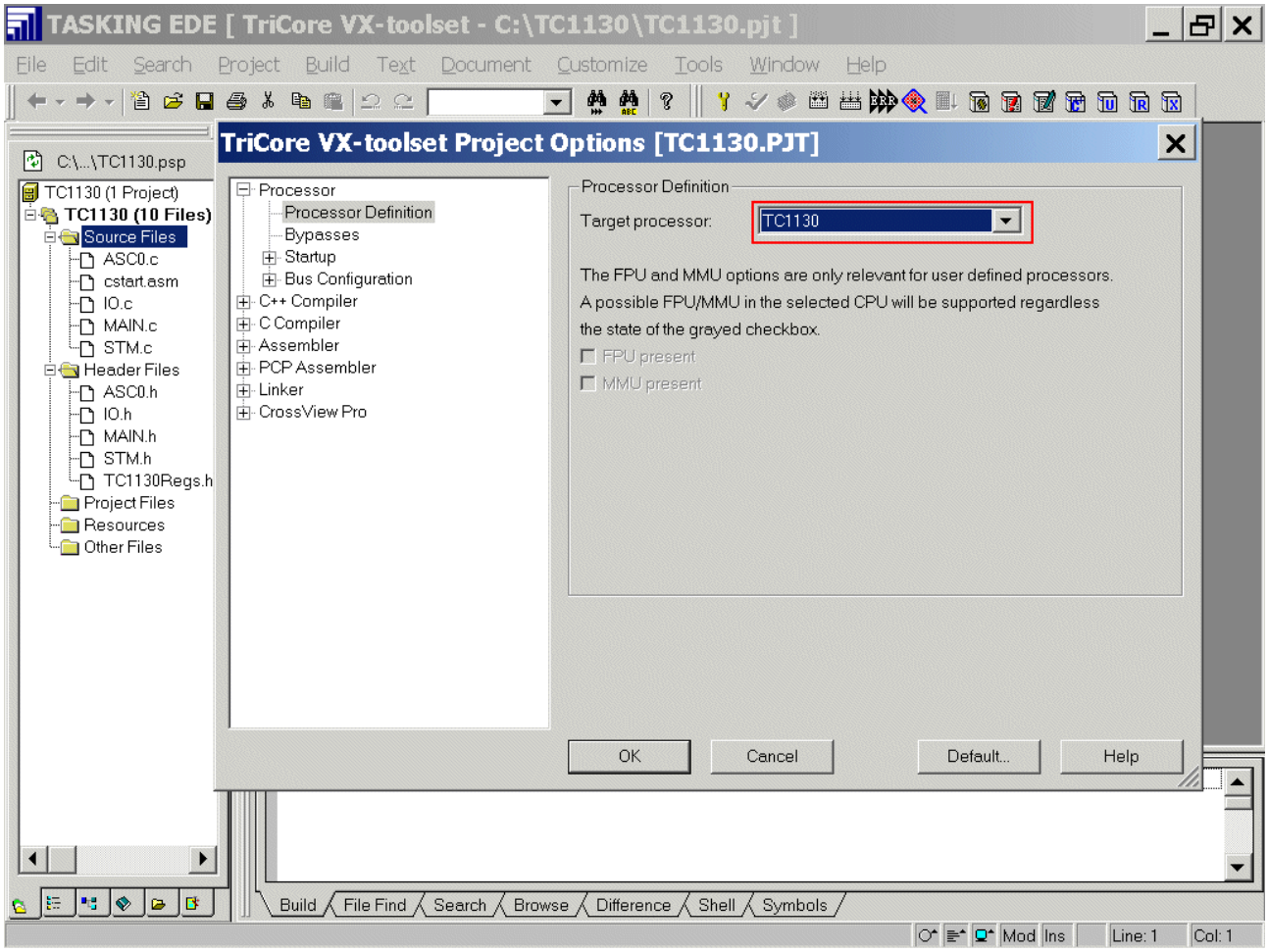
OK


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1130.psp
TC1130 (1 Project)
TC1130 (10 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
TC1130Regs.h
Project Files
Resources
Other Files
Build File Find Search Browse Difference Shell Symbols
Mod Ins Line: 1 Col: 1

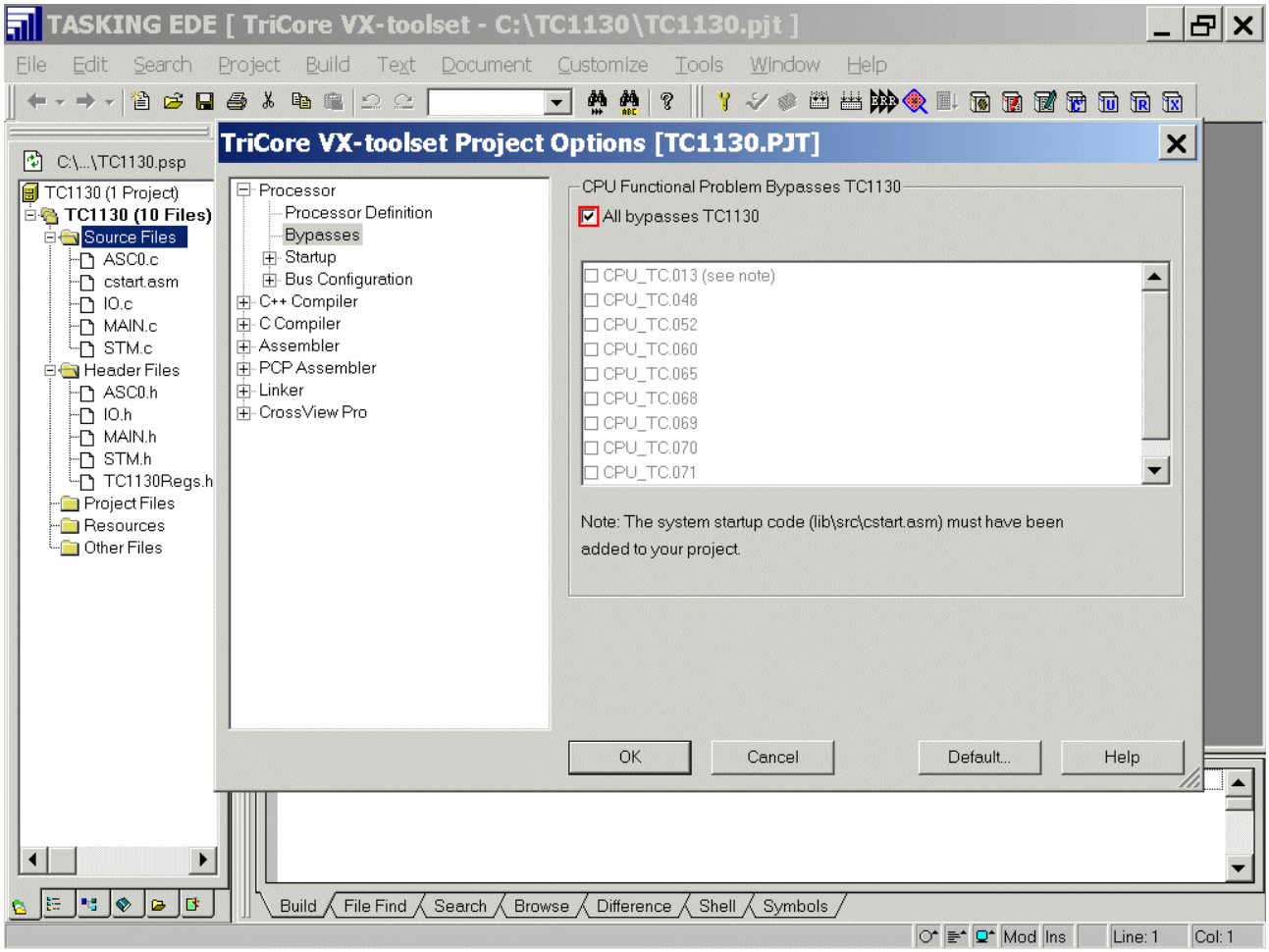
Configure Compiler, Assembler, Linker, Locater and Build – Control:

Project – Project Options

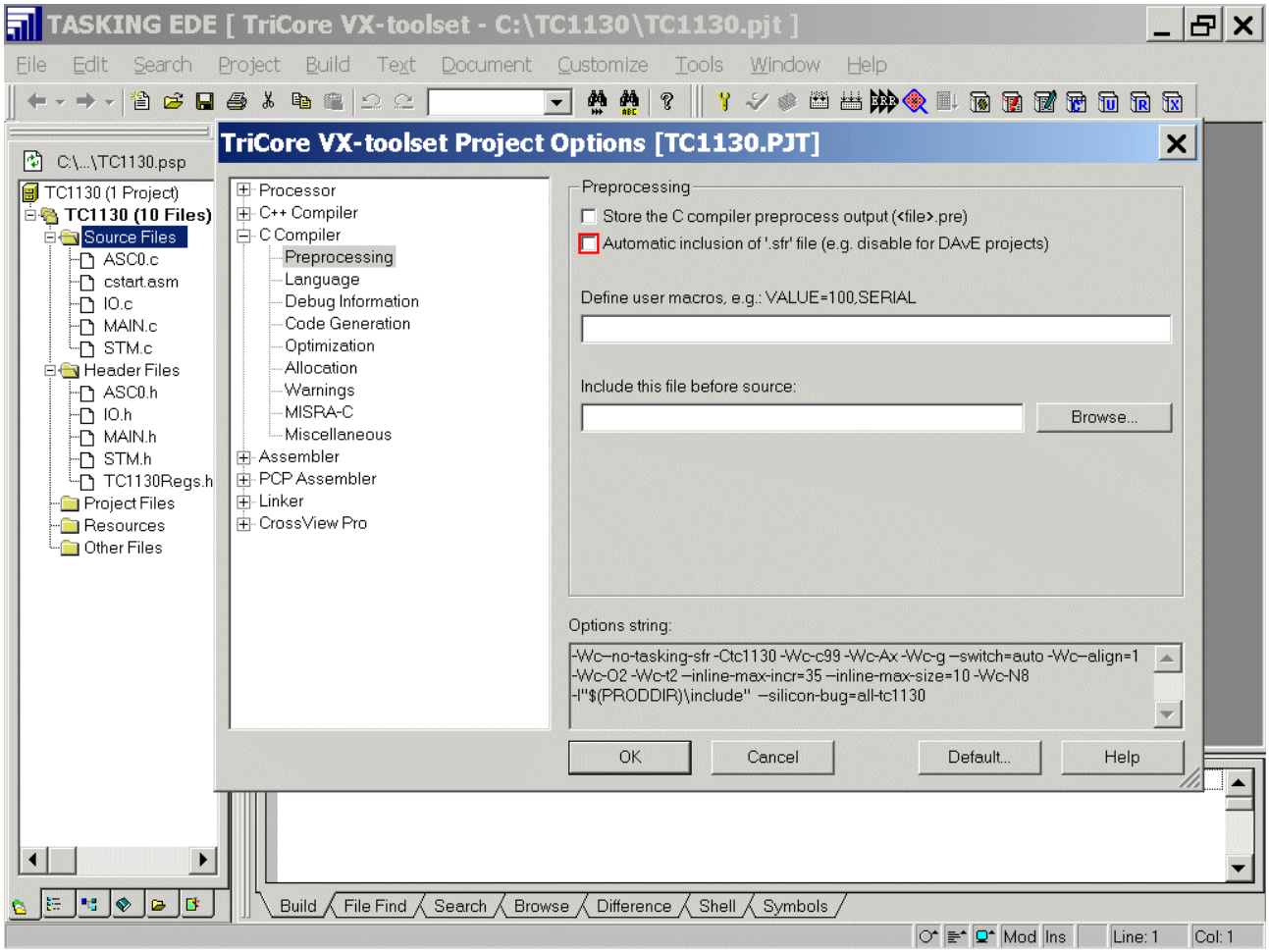
Processor: Processor Definition: Target processor: select TC1130



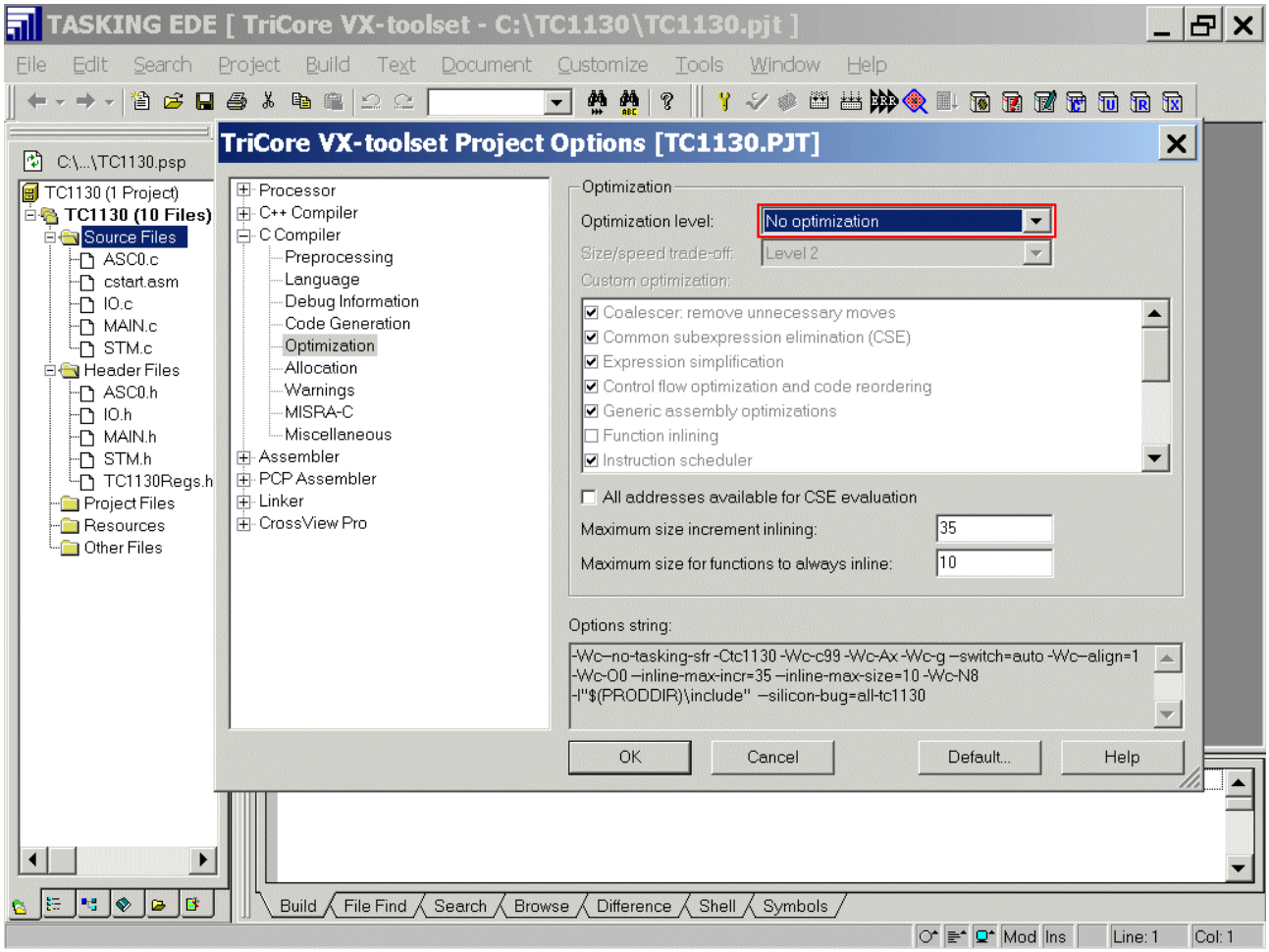
Processor: Bypasses: CPU Functional Problem Bypasses: click ✓ All bypasses TC1130



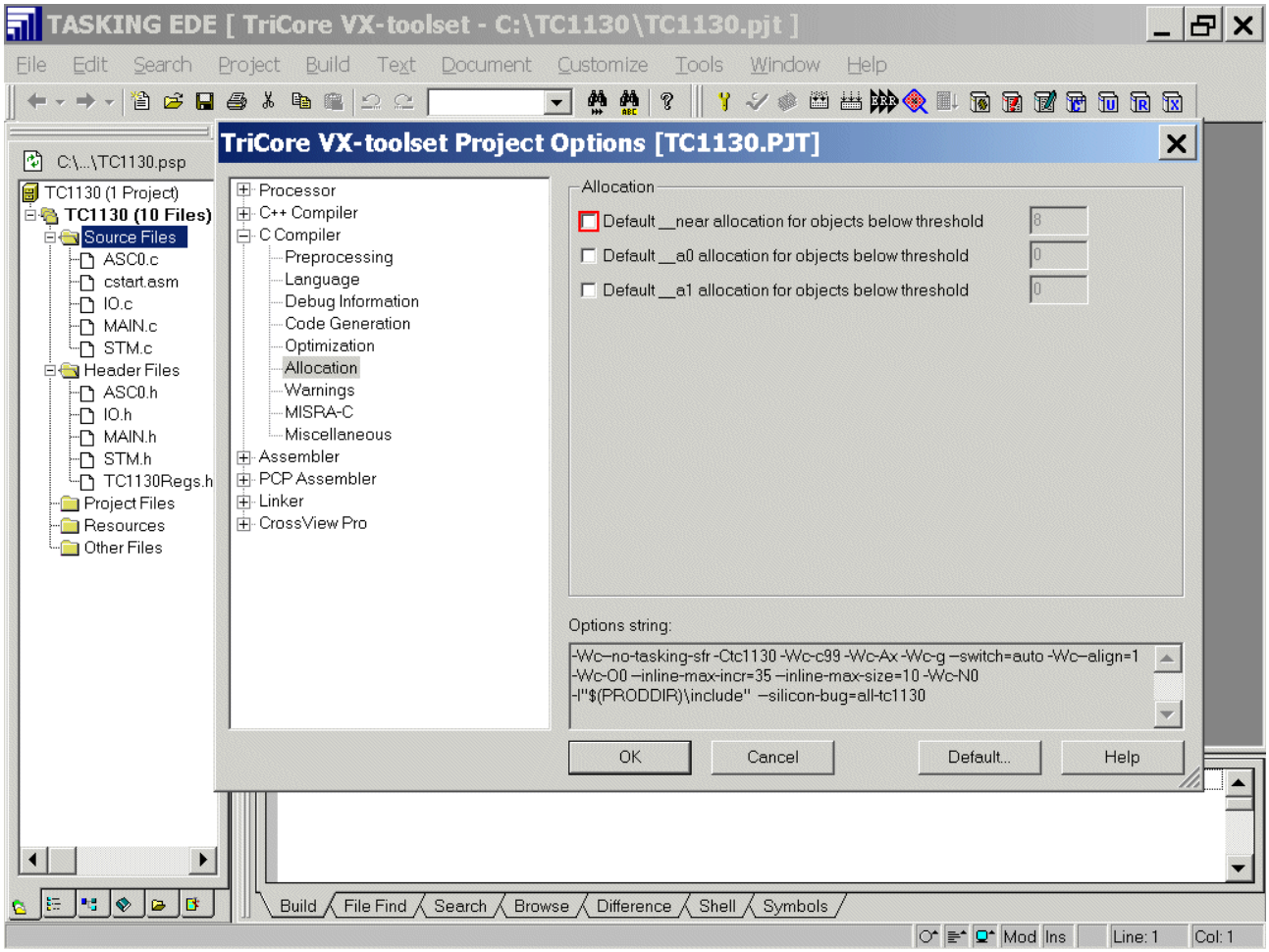
C Compiler: Preprocessing: click to deactivate ☐ Automatic inclusion of .sfr file



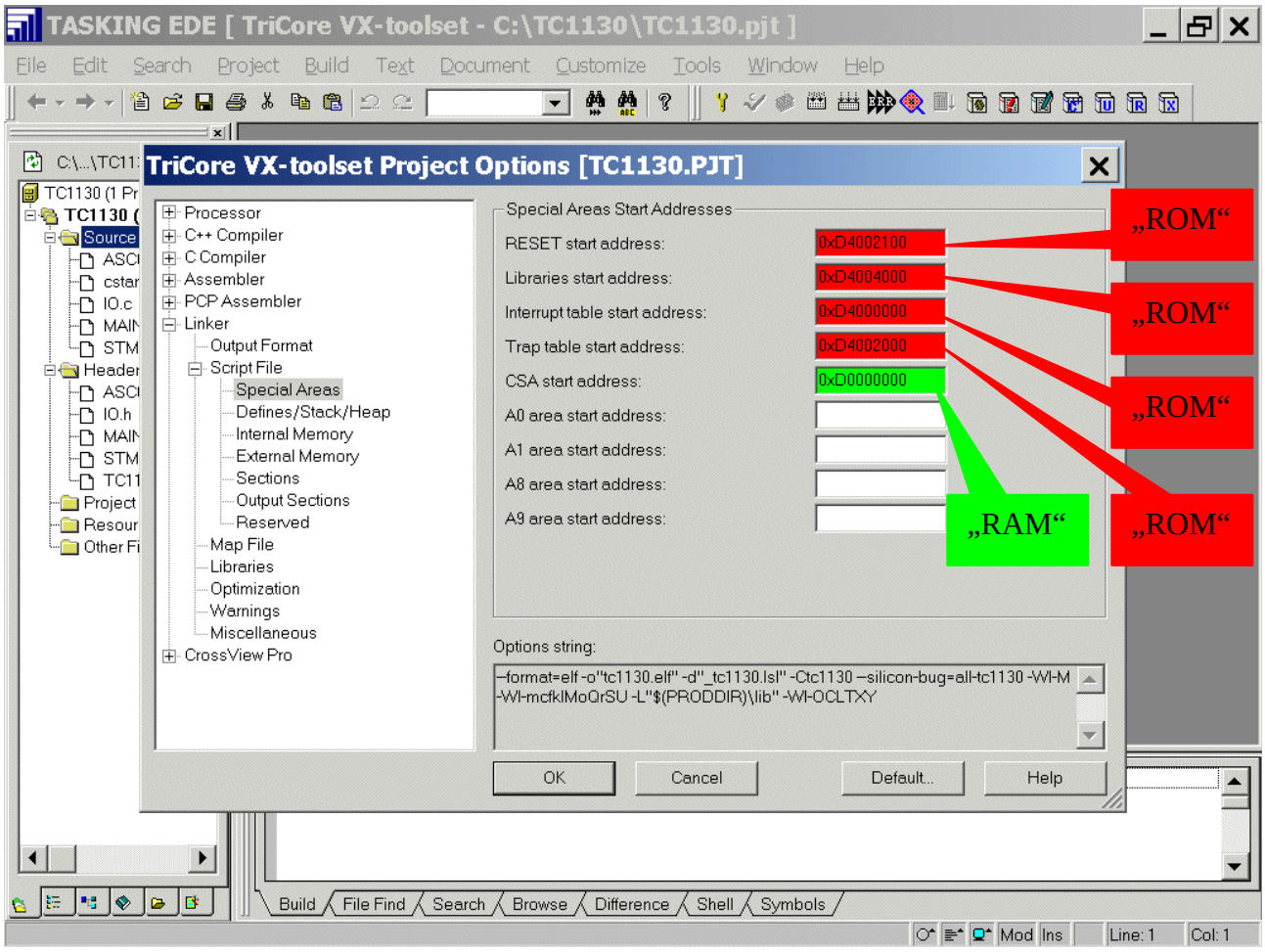
C Compiler: Optimization: Optimization level: select No optimization



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

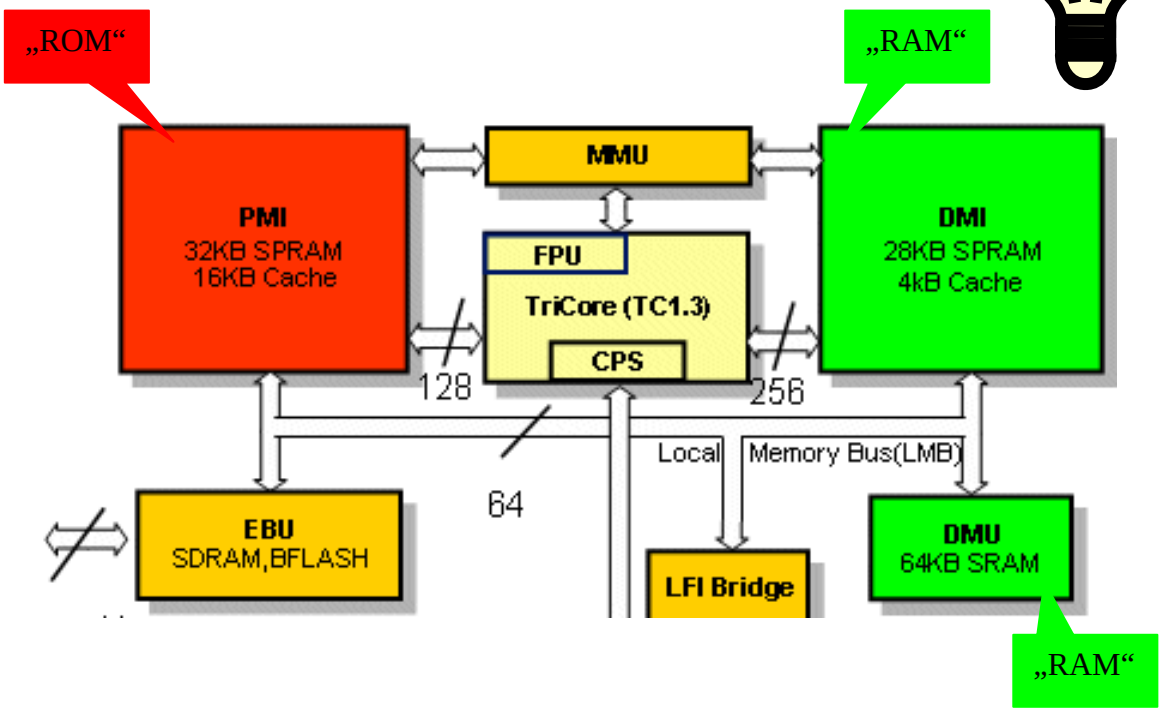


Linker: Script File: Special Areas: RESET start address: insert 0xD4002100 (PMI_SPRAM)
Linker: Script File: Special Areas: Libraries start address: insert 0xD4004000 (PMI_SPRAM)
Linker: Script File: Special Areas: Interrupt table start address: insert 0xD4000000 (PMI_SPRAM)
Linker: Script File: Special Areas: Trap table start address: insert 0xD4002000 (PMI_SPRAM)
Linker: Script File: Special Areas: CSA start address: insert/check 0xD0000000 (DMI_LDRAM)



Note:
DMI Local Data RAM = DMI_LDRAM = “DMI_SPRAM”

Additional Information:



Adobe Reader - [tc1130_um_v1.3_2004_11_sys.pdf]

File Edit View Document Tools Window Help

TC1130 (Vol. 1 of 2)
System Units

Memory Map of On-Chip Local Memories

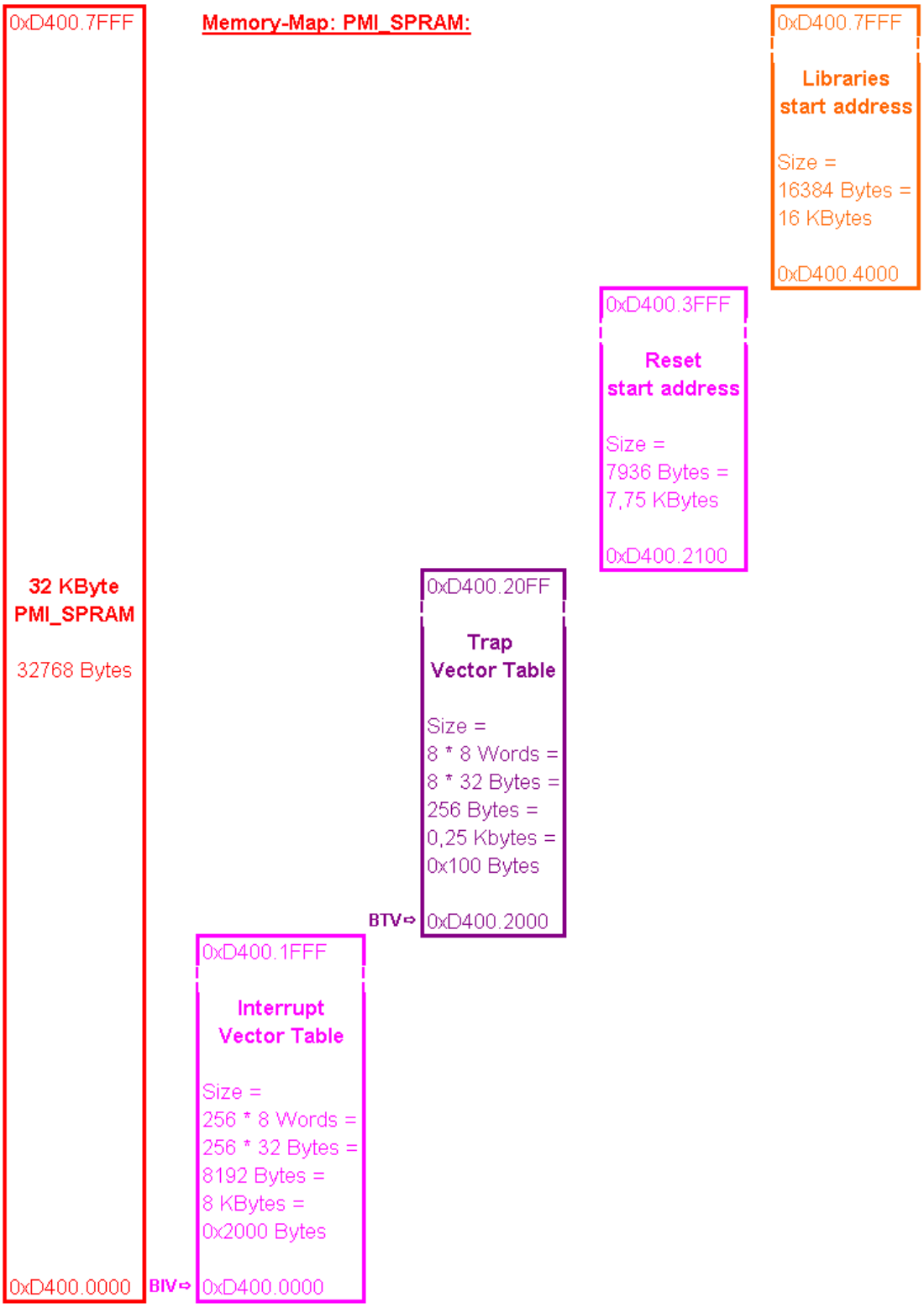
Table 7-1 TC1130 Block Address Map (cont'd)

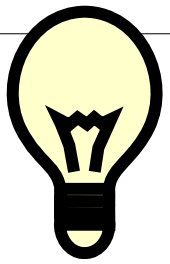
Segment	Address Range	Size	Description	DMI Acc.	PMI Acc.	
12	$C000\ 0000_H$ - $C000\ FFFF_H$	64 KB	DMU	via LMB	via LMB	cached
	$C001\ 0000_H$ - $CFFF\ FFFF_H$	≈ 256 MB	Reserved			
13	$D000\ 0000_H$ - $D000\ 6FFF_H$	28 KB	DMI Local Data RAM (LDRAM)	DMI local	via LMB	non-cached
	$D000\ 7000_H$ - $D3FF\ FFFF_H$	≈ 64 MB	Reserved			
	$D400\ 0000_H$ - $D400\ 7FFF_H$	32 KB	PMI Local Code Scratch Pad RAM (SPRAM)	via LMB	PMI local	cached
	$D400\ 8000_H$ - $D7FF\ FFFF_H$	≈ 64 MB	Reserved			

8,15 x 11,58 in

243 of 979

Additional Information:





Additional Information:

Interrupt Vector Table:

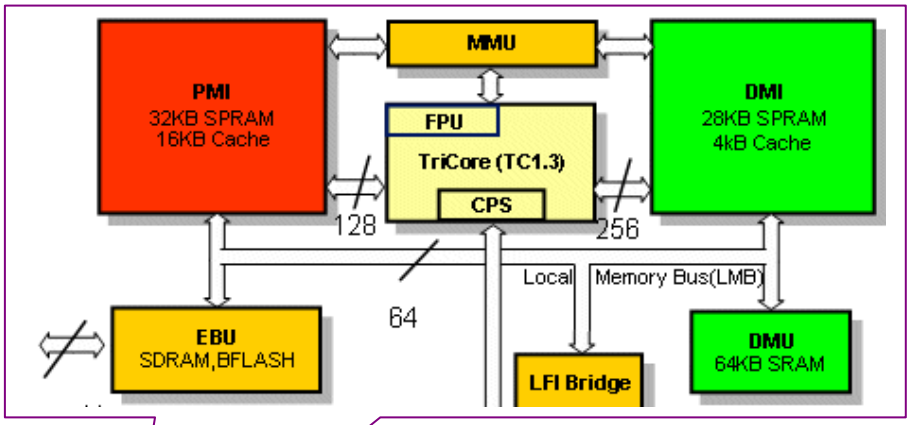
For the full range of 256 interrupt entries an alignment to an 8 KBytes boundary is required.

BTV = 0xD400.0000 = 3.556.769.792 / 8.192 = 434.176 ✓

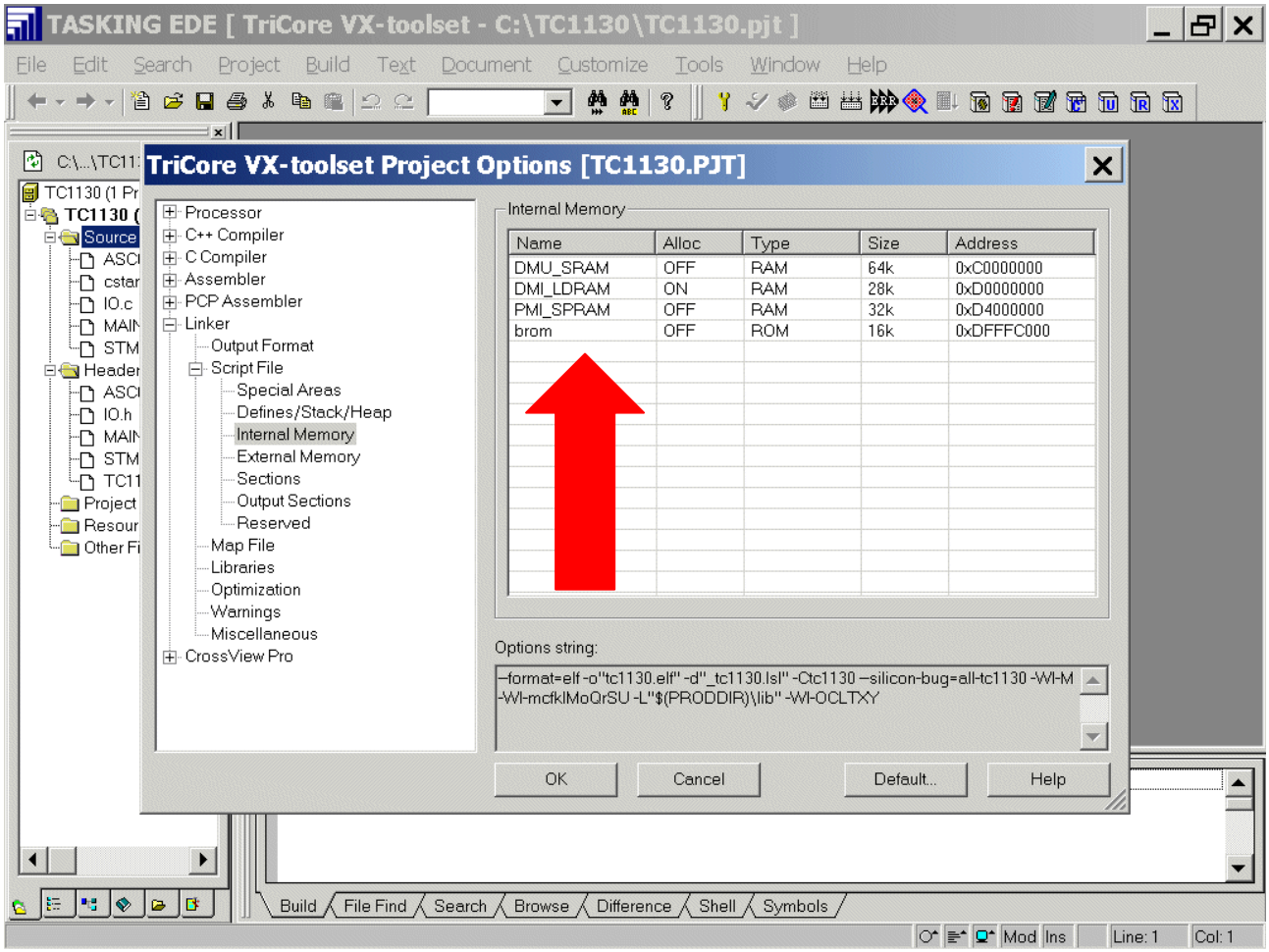
Trap Vector Table:

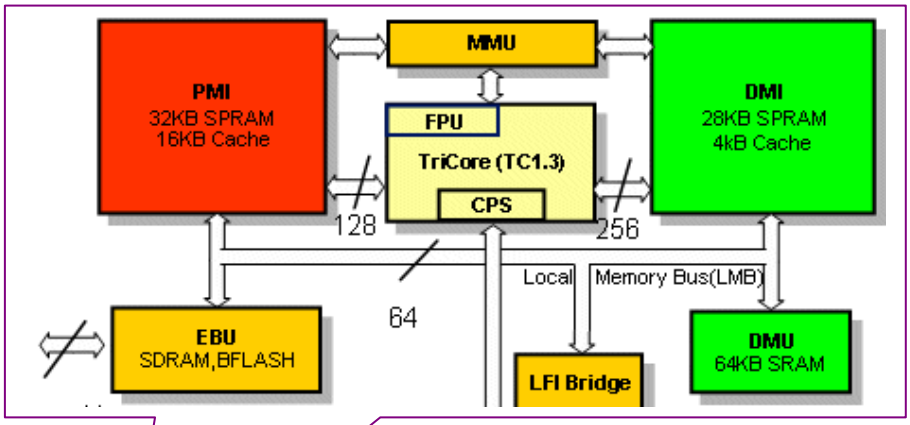
There are eight different trap classes, resulting in Trap Classes from 0 to 7. The contents of BTV should therefore be at least set to a 256 Byte boundary (8 Trap Classes * 8 word spacing).

BIV = 0xD400.2000 = 3.556.777.984 / 256 = 13.893.664 ✓

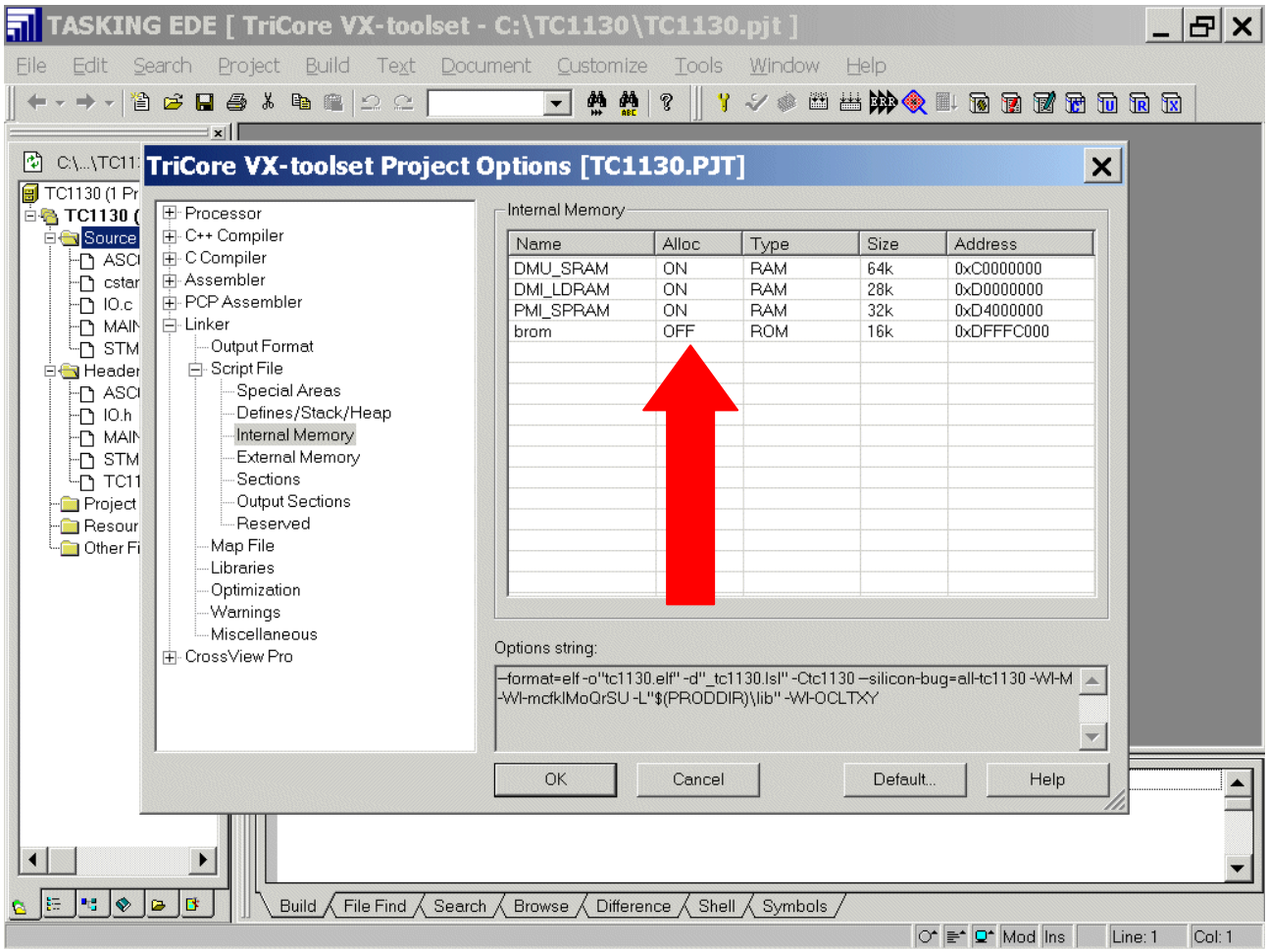


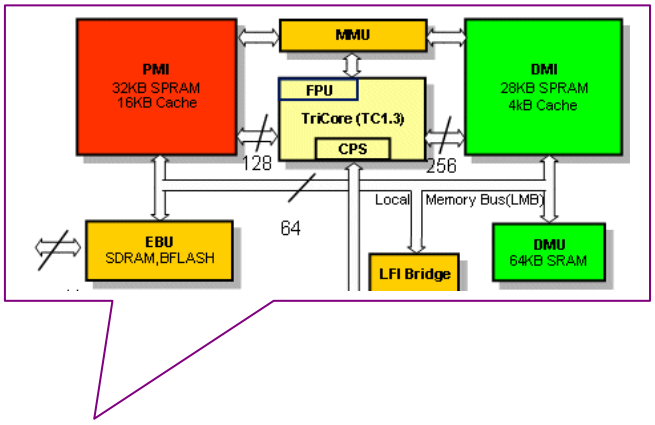
Linker: Script File: Internal Memory: change from lmbram to DMU_SRAM
Linker: Script File: Internal Memory: change from ldram to DMI_LDRAM
Linker: Script File: Internal Memory: change from spram to PMI_SPRAM



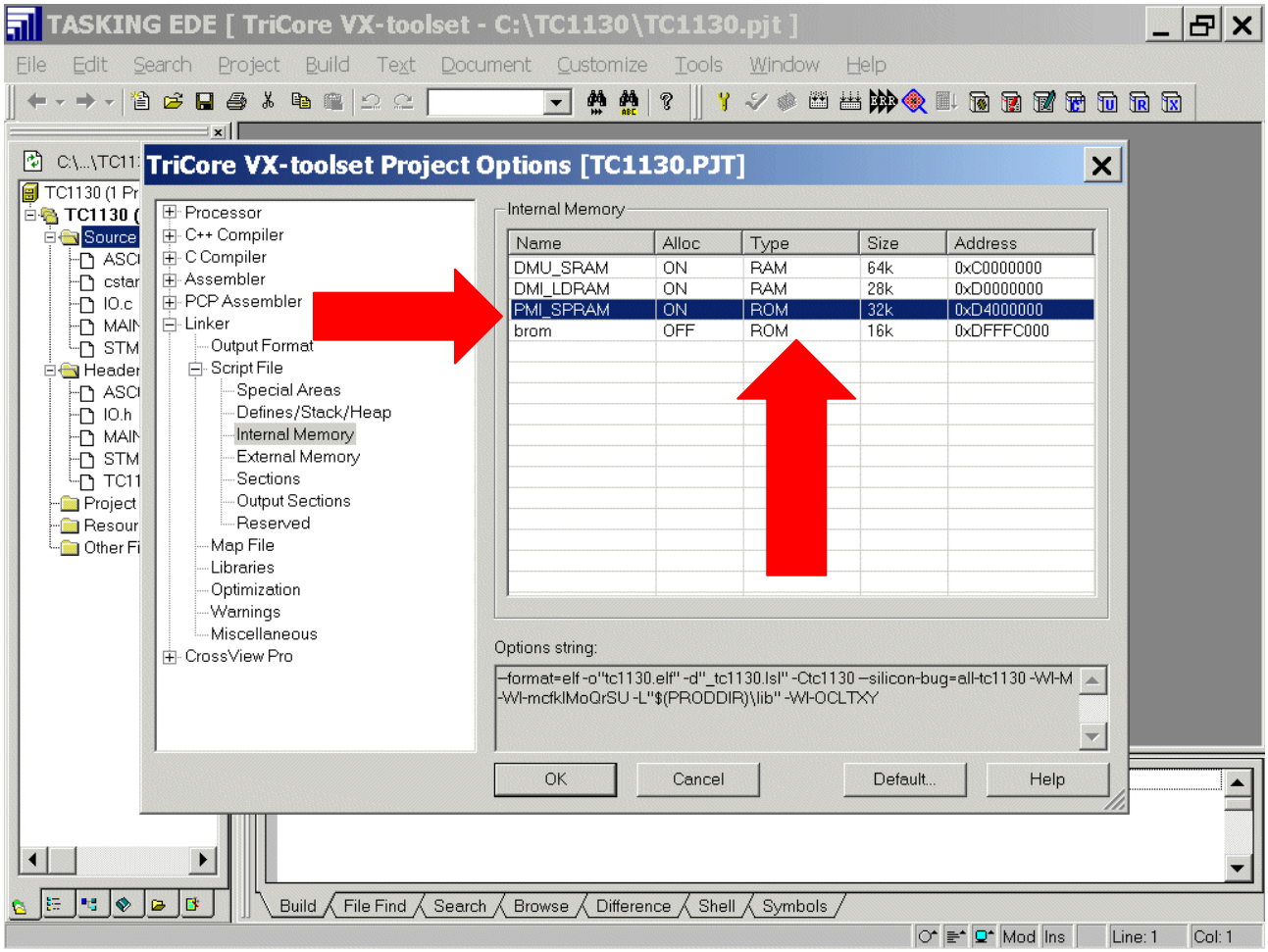


Linker: Script File: Internal Memory: DMU_SRAM: Alloc: select ON!!!
Linker: Script File: Internal Memory: PMI_SPRAM: Alloc: select ON!!!



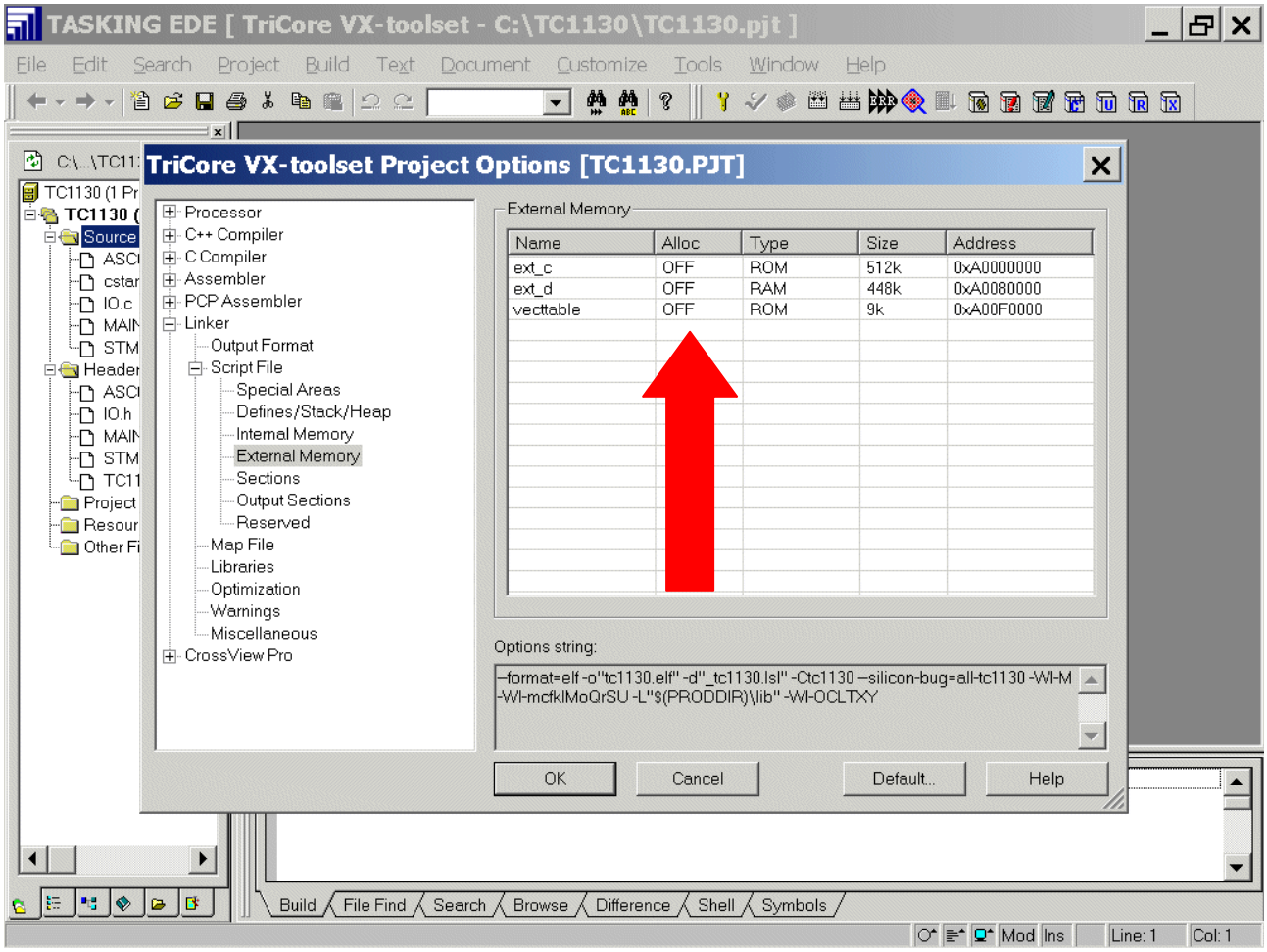


Linker: Script File: Internal Memory: PMI_SPRAM: Type: select ROM 

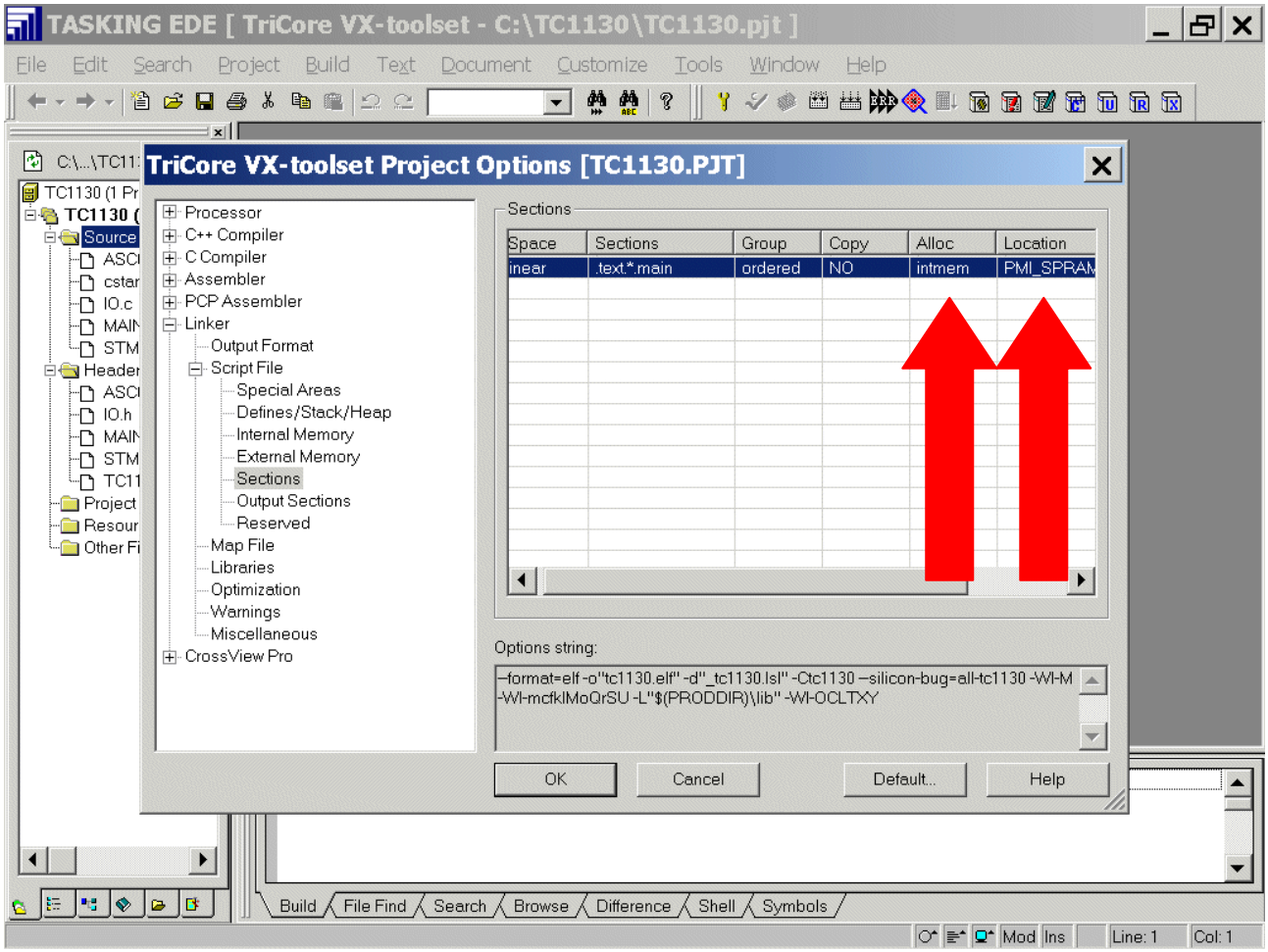


(Configuration of this dialog-window is now finished)

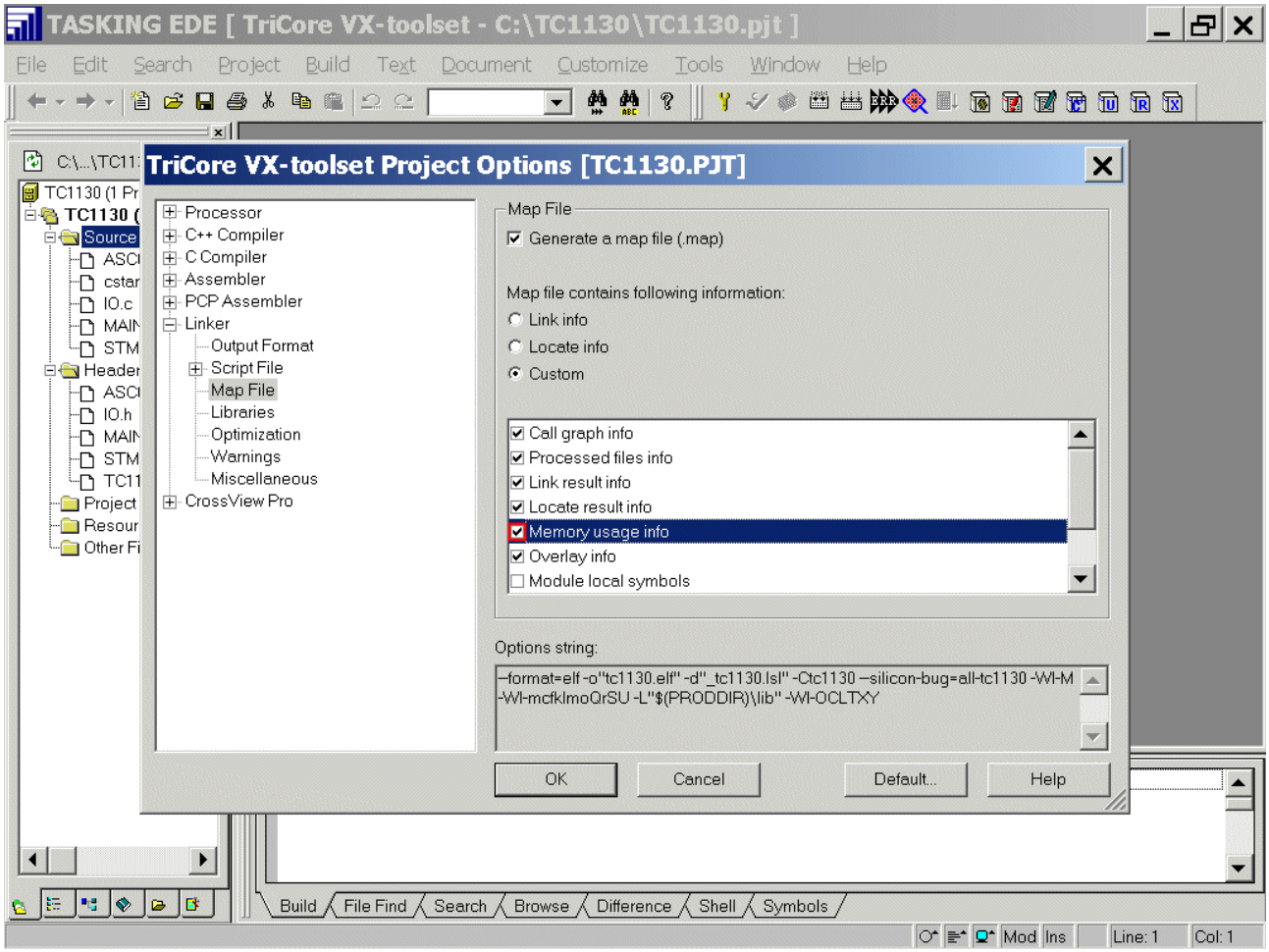
Linker: Script File: External Memory: ext_c: Alloc: select OFF
Linker: Script File: External Memory: ext_d: Alloc: select OFF
Linker: Script File: External Memory: vectable: Alloc: select OFF



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

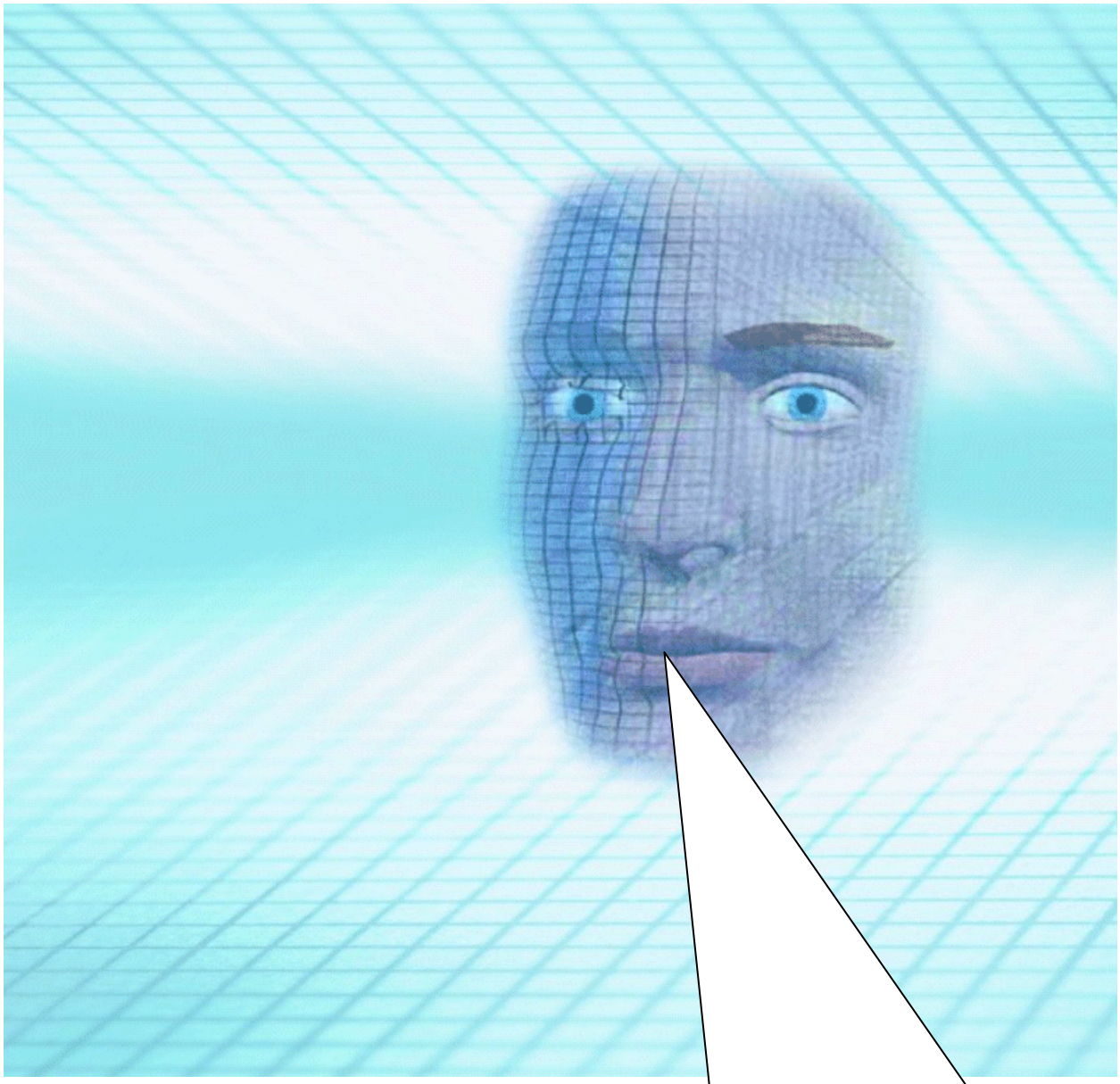


Linker: Map File: click ✓ Memory usage info



OK

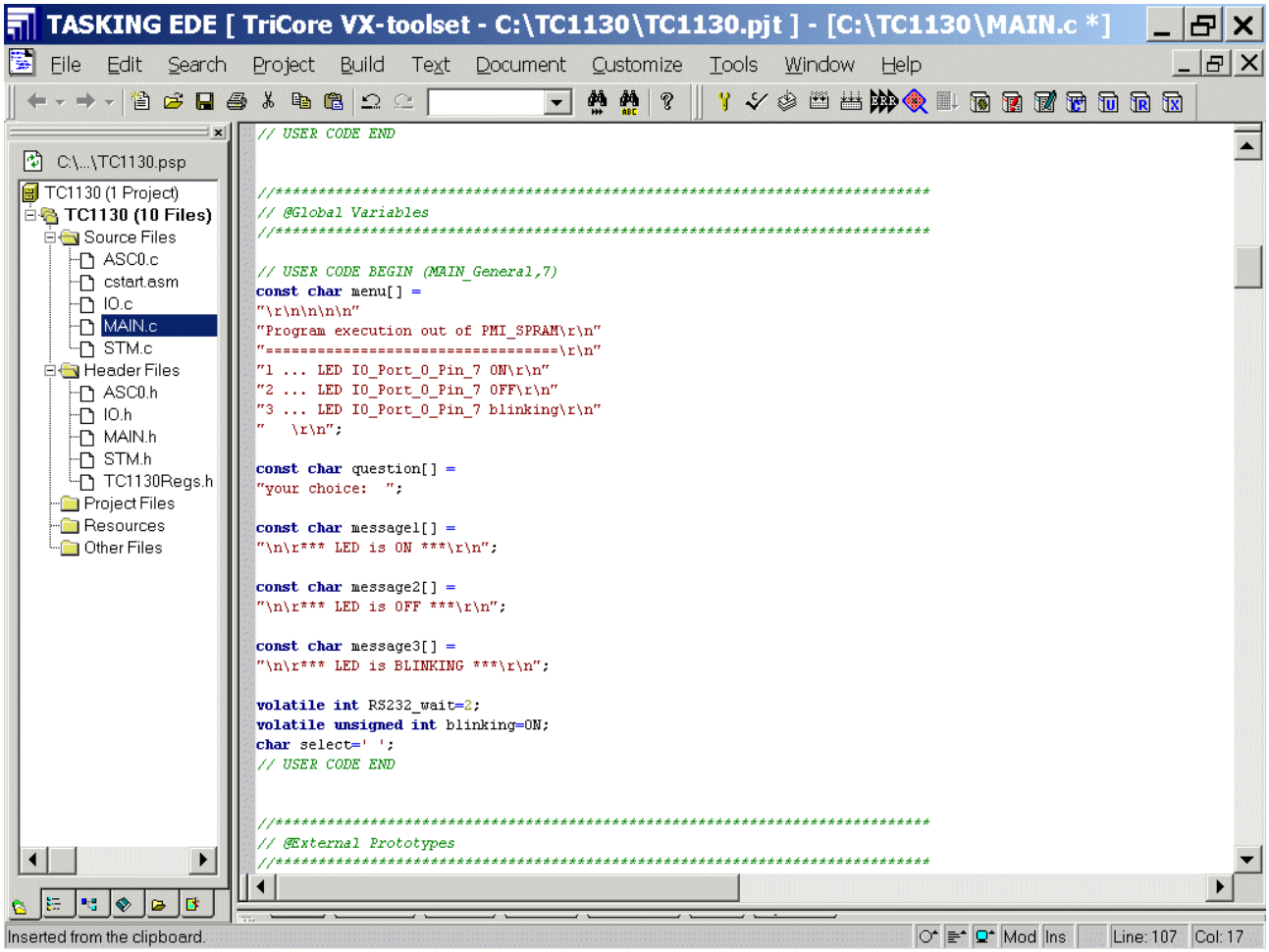
Insert your application specific program:



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!

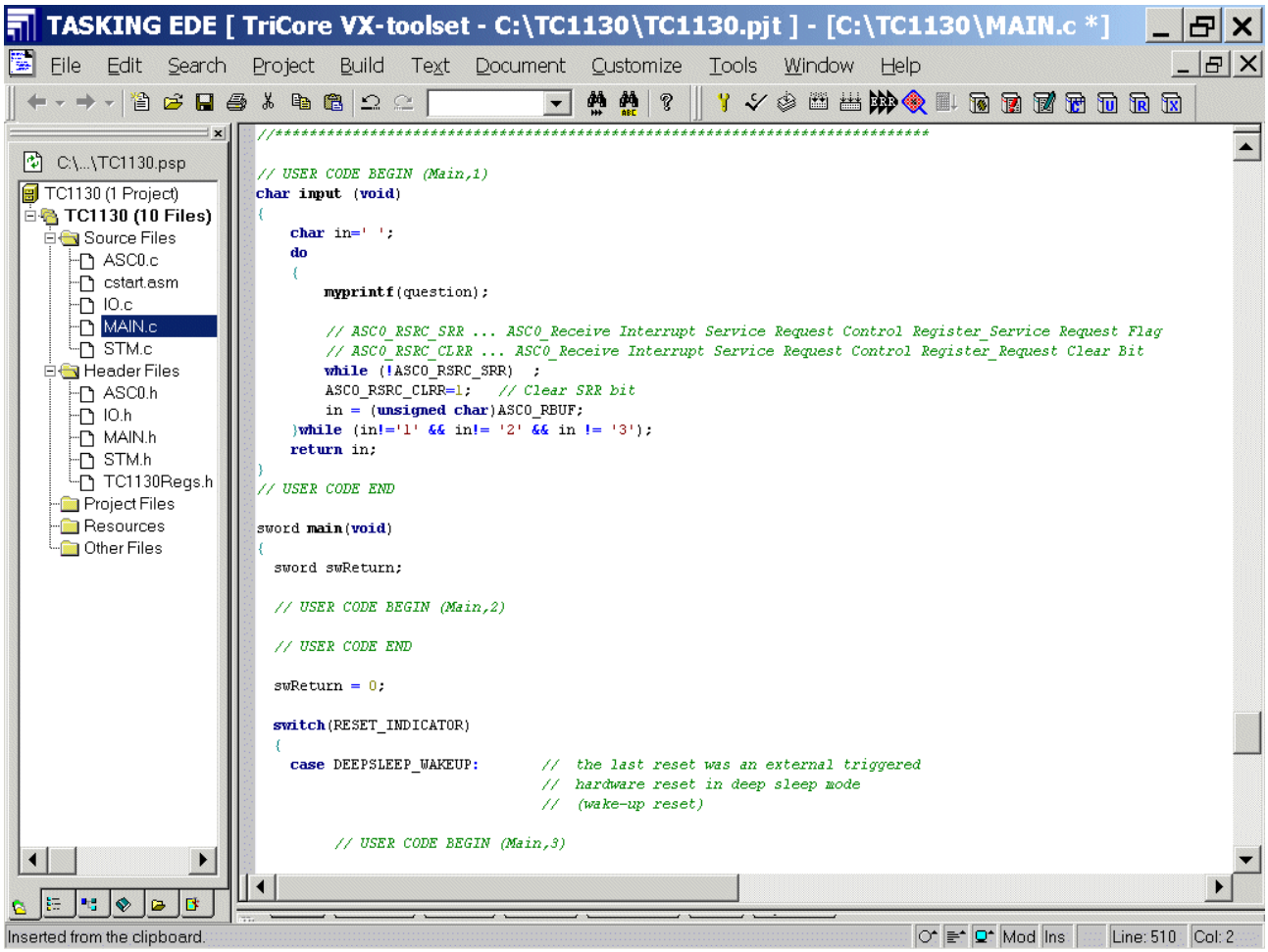
Double click: Main.c insert User Code (Global Variables):

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of PMI_SPRAM\r\n"
"==================================\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 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:\TC1130\TC1130.pjt] - [C:\TC1130\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1130.psp
TC1130 (1 Project)
TC1130 (10 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
TC1130Regs.h
Project Files
Resources
Other Files
// USER CODE END
//*****
// @Global Variables
//*****
// USER CODE BEGIN (MAIN_General,7)
const char menu[] =
"\r\n\n\n\n"
"Program execution out of PMI_SPRAM\r\n"
"==========\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 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
//*****
Inserted from the clipboard.
Mod Ins
Line: 107 Col: 17

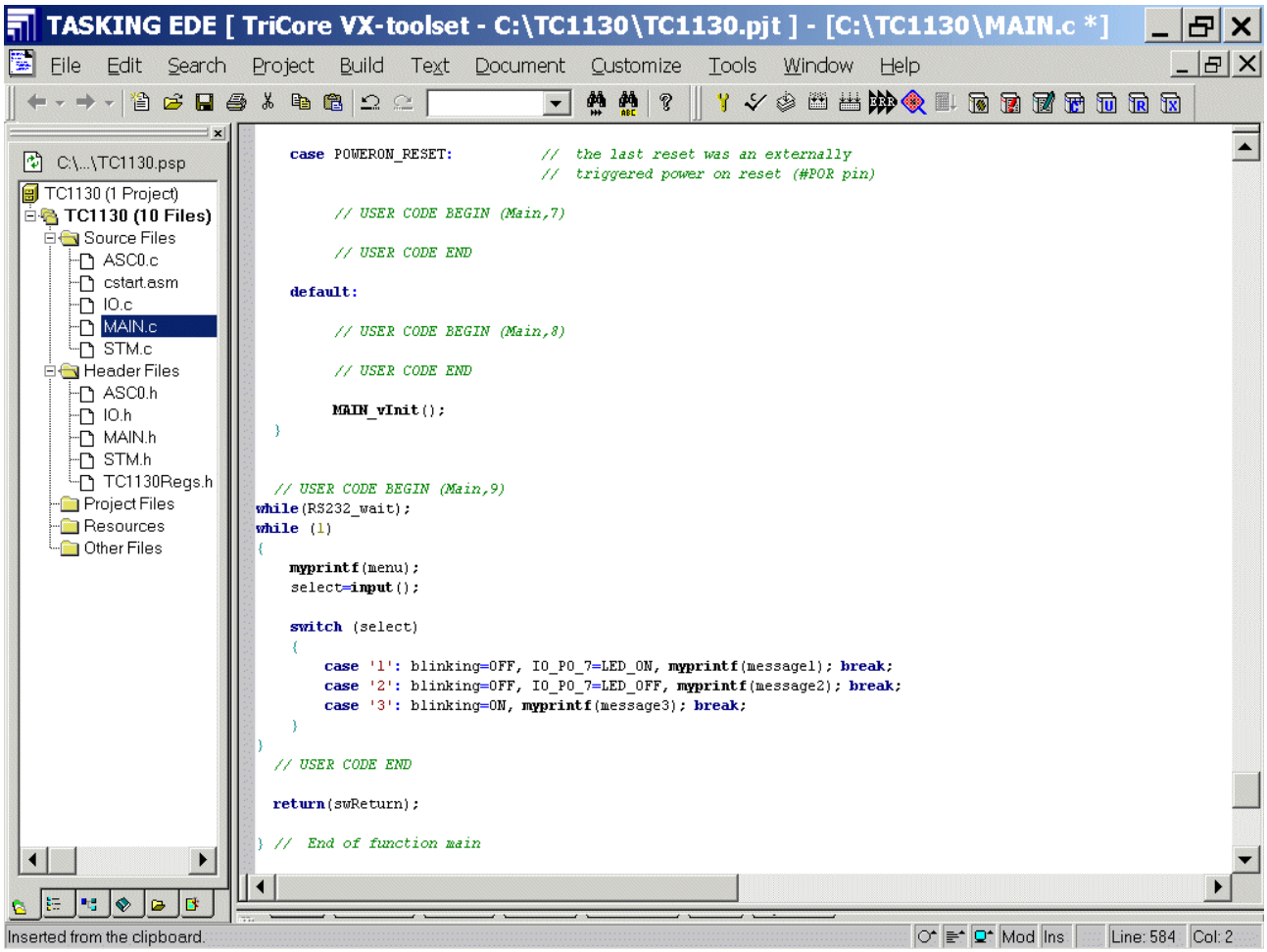
Double click: Main.c insert User Code [function: input()]:

```
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;
}
```



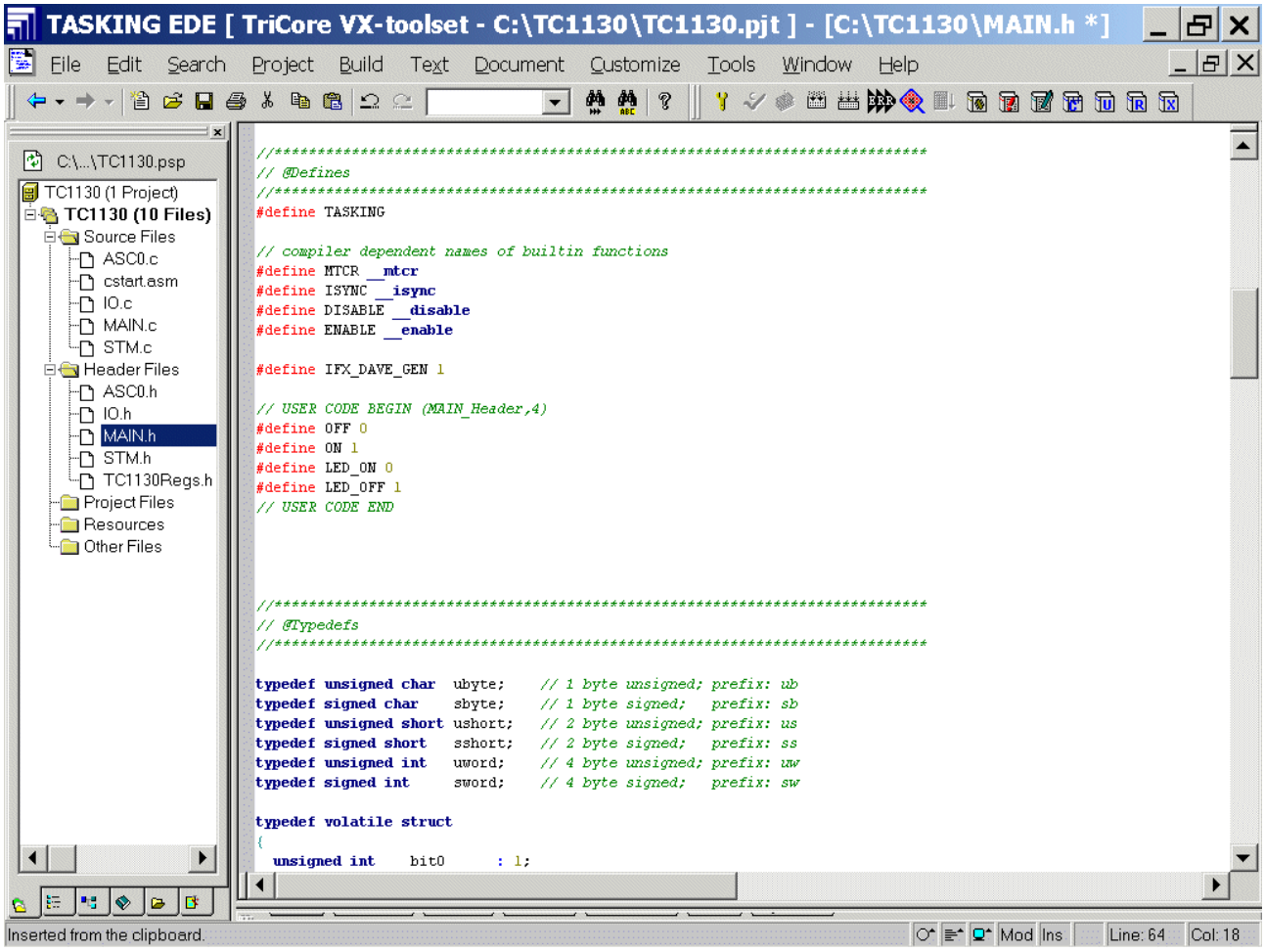
Double click: Main.c insert Code:

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

   switch (select)
   {
      case '1': blinking=OFF, IO_P0_7=LED_ON, myprintf(message1); break;
      case '2': blinking=OFF, IO_P0_7=LED_OFF, myprintf(message2); break;
      case '3': blinking=ON, myprintf(message3); break;
   }
}
```



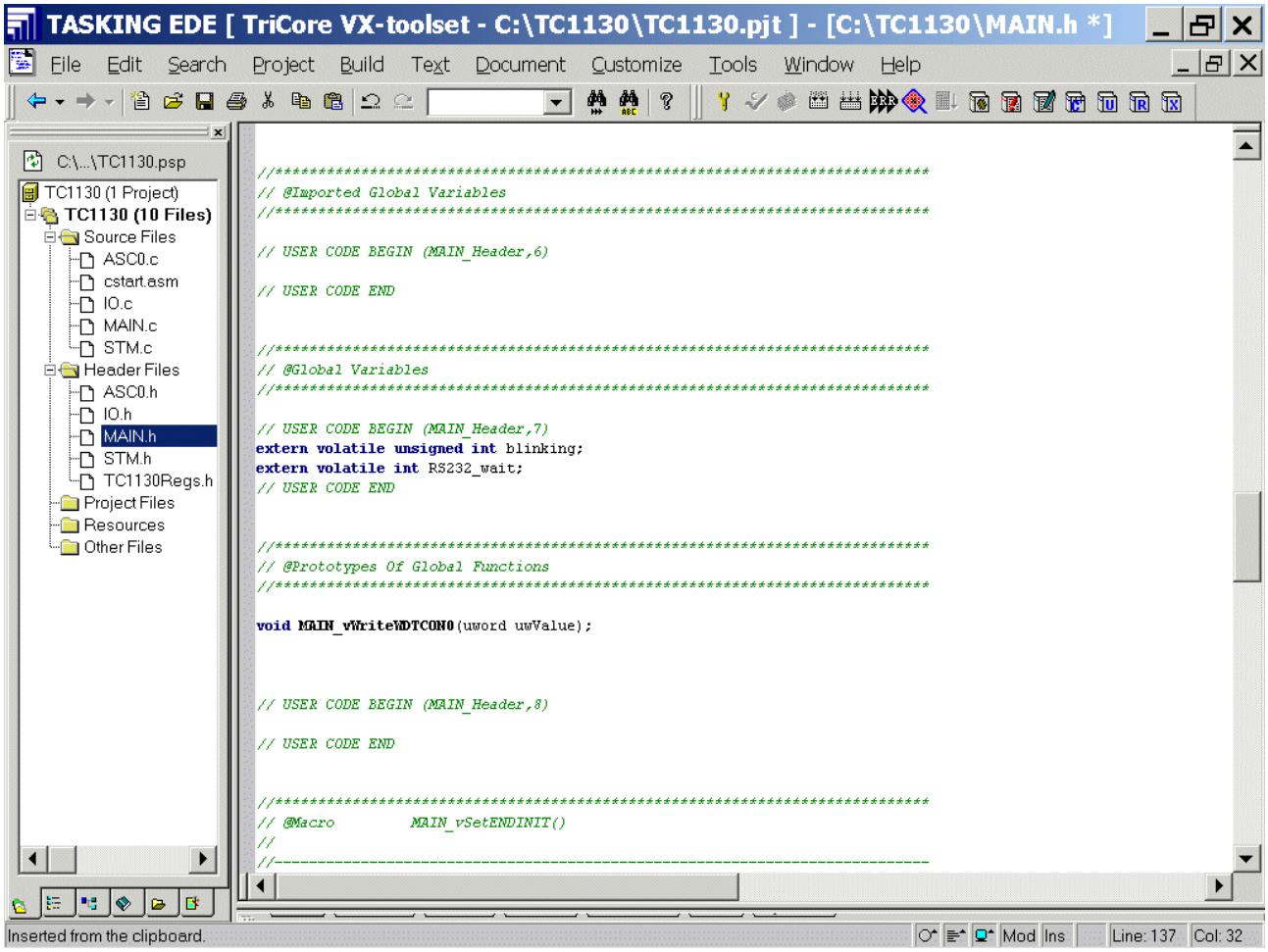
Double click: Main.h and insert the following Defines:

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



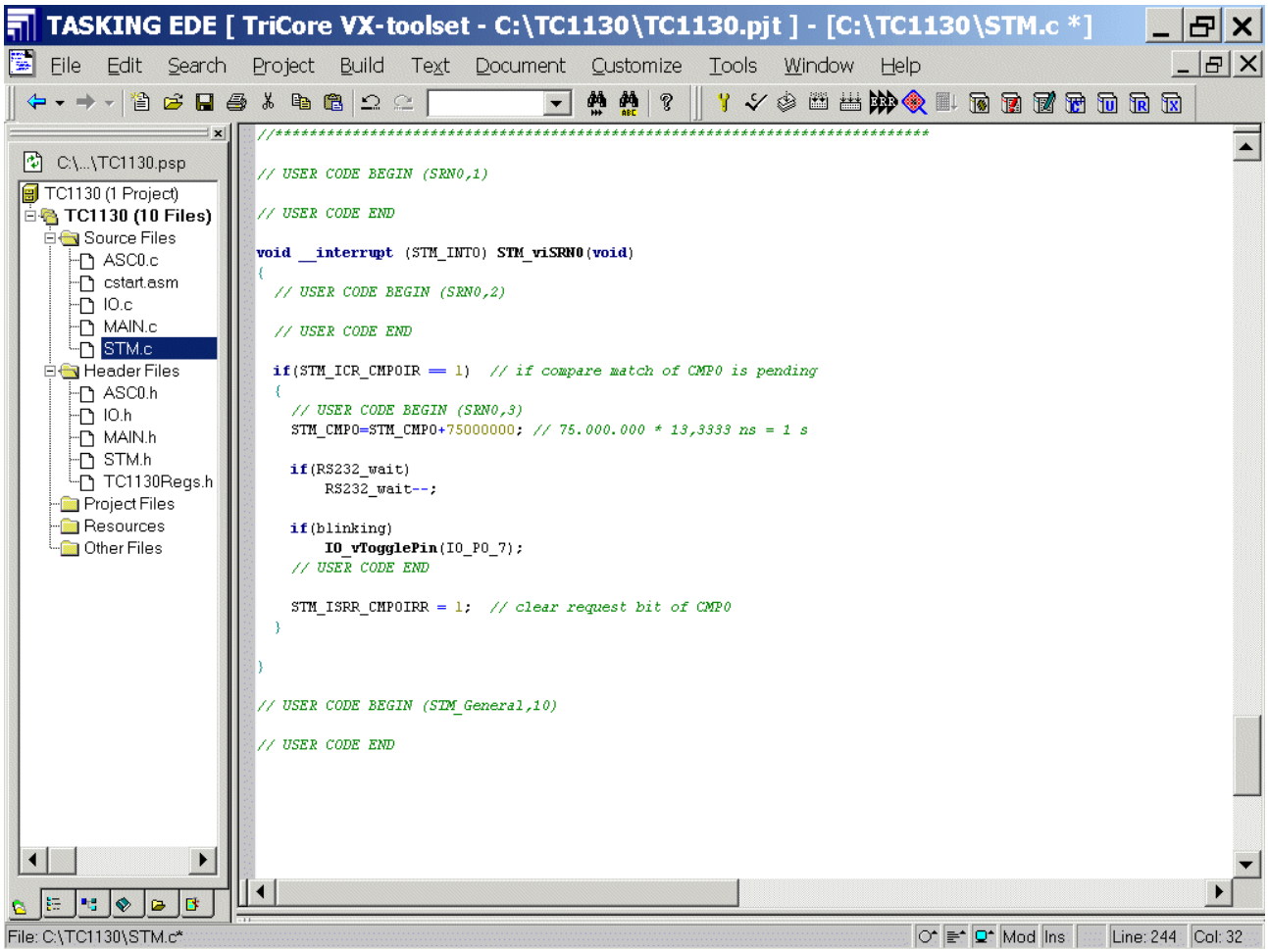
Double click: Main.h and insert Global Variables (Extern Declarations):

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



Double click: STM.c insert User Code for system-timer's interrupt-service-routine:

```
STM_CMP0=STM_CMP0+75000000; // 75.000.000 * 13,3333 ns = 1 s

if(RS232_wait)
   RS232_wait--;

if(blinking)
   IO_vTogglePin(IO_P0_7);
```

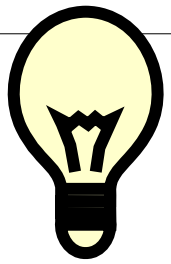


Note:
75.000.000 * 13,3333 ns = 1 s

To get an STM-interrupt every 1 second you must change the Compare-Value to "STM_CMP0+=75000000;", because there is no "reload-functionality"!

Warning:
If you forget to do so you will get the following undesired functionality:

1. Interrupt: 0b10001111000011010001100000 = 75000000 => 1 sec
2. Interrupt: 0b11000111100001101000110000000 = 209217728 => 2,789 sec
3. Interrupt : 0b101000111100001101000110000000 = 343435456 => 4,579 sec



Additional Information:

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

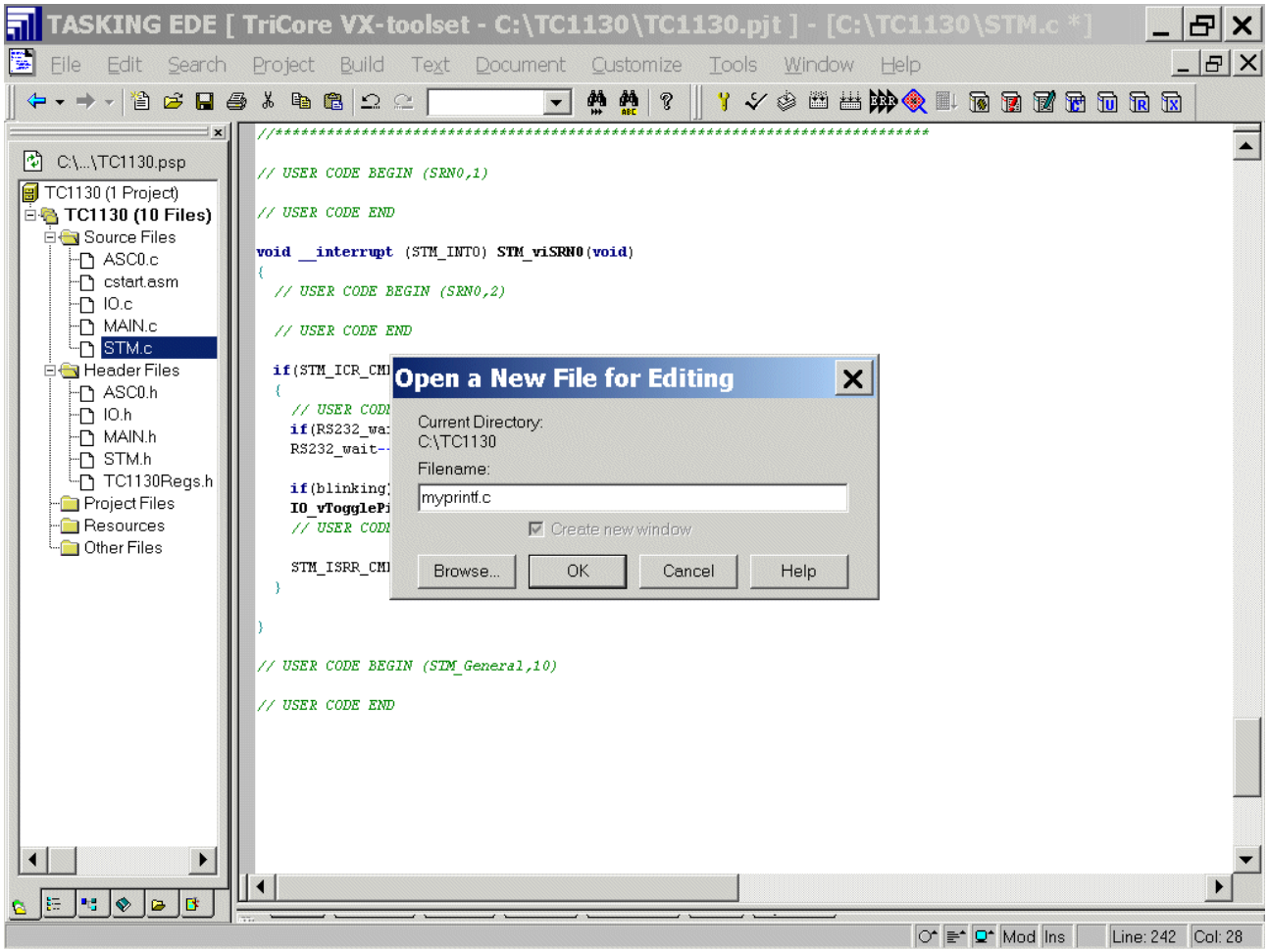
DESCRIPTION
Change request for a low-level I/O (_write function implementation) = example which does not use simulated I/O but uses the real serial = interface of the controller.

EXAMPLE

WORKAROUND

Note:
We use myprintf(); instead of printf(); for all 32-bit-Cookery-Books just to be independent from the compiler-vendor-realisation of low-level-I/O-software for printf().

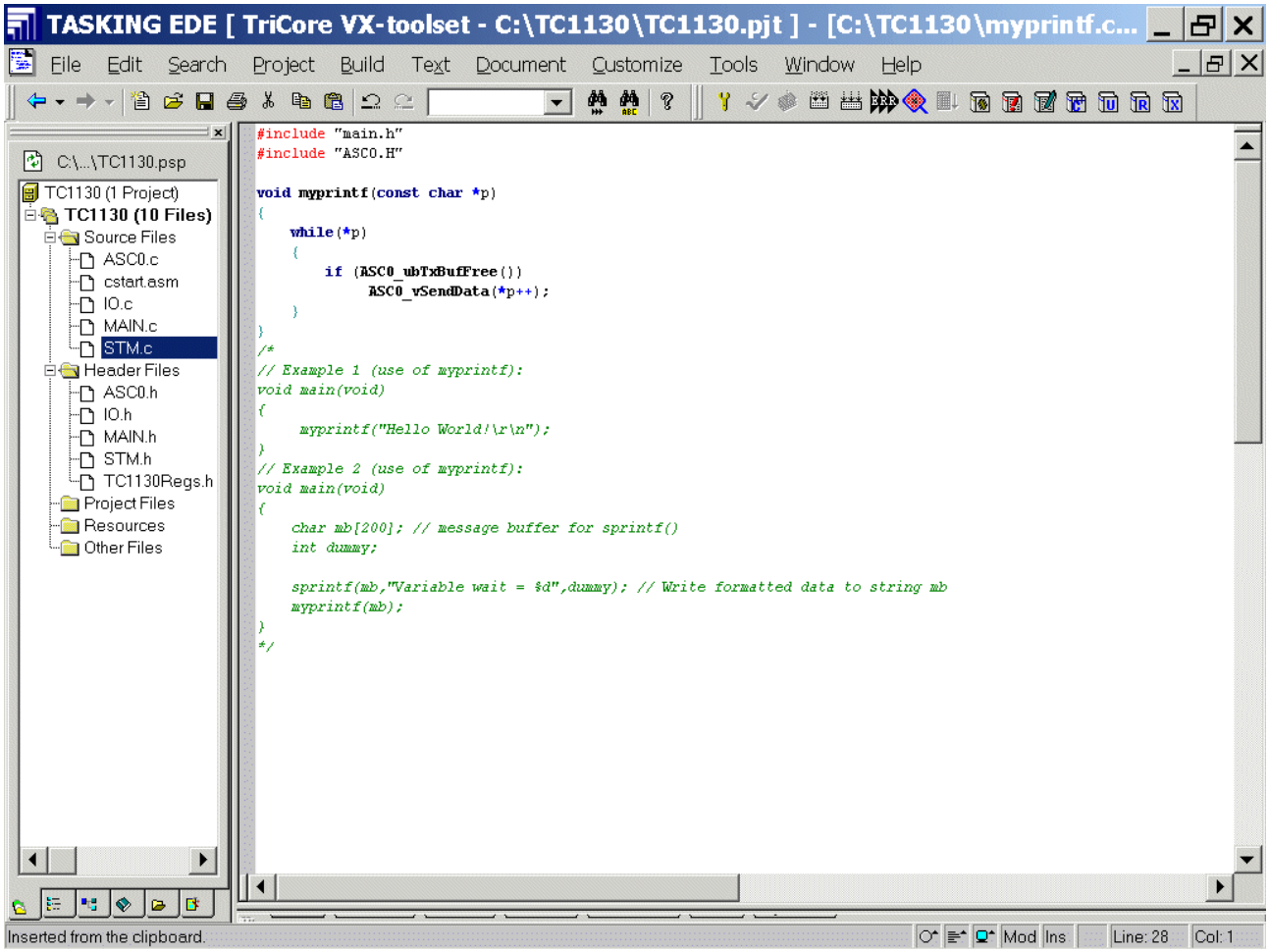
File – New
Insert myprintf.c



OK

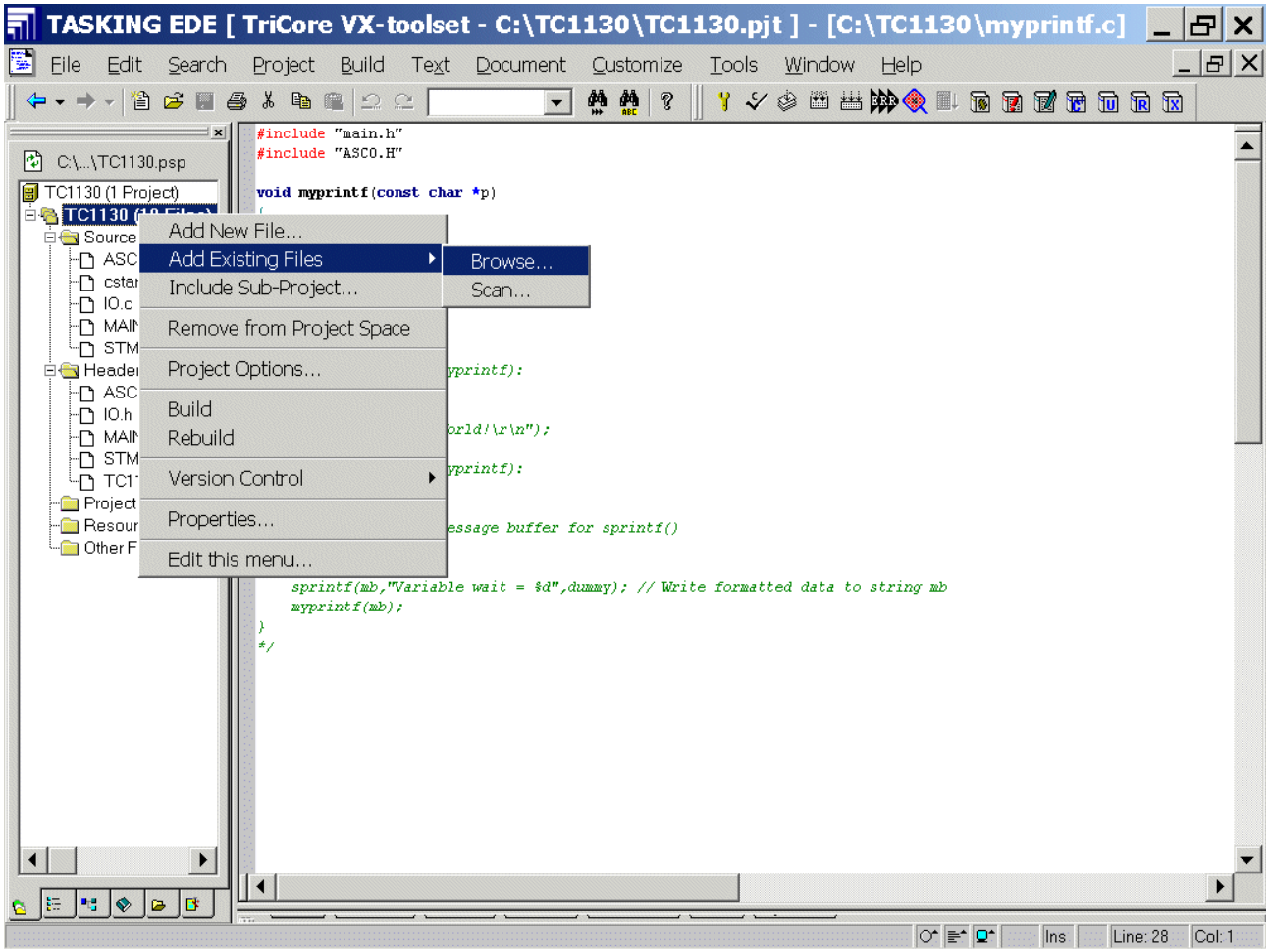
Insert User Code for myprintf():

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

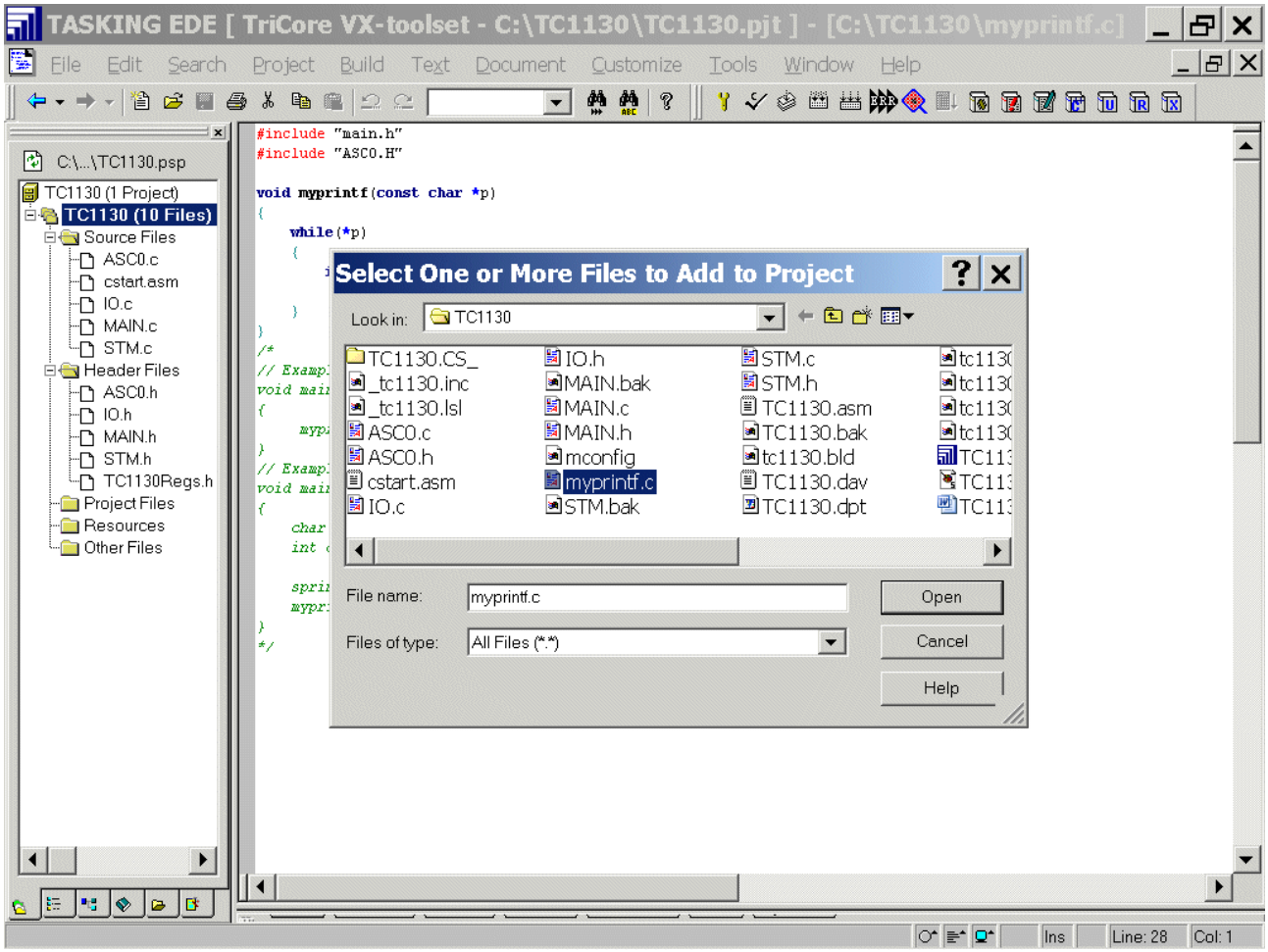

TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\myprintf.c...
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1130.psp
TC1130 (1 Project)
TC1130 (10 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
STM.c
Header Files
ASC0.h
IO.h
MAIN.h
STM.h
TC1130Regs.h
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

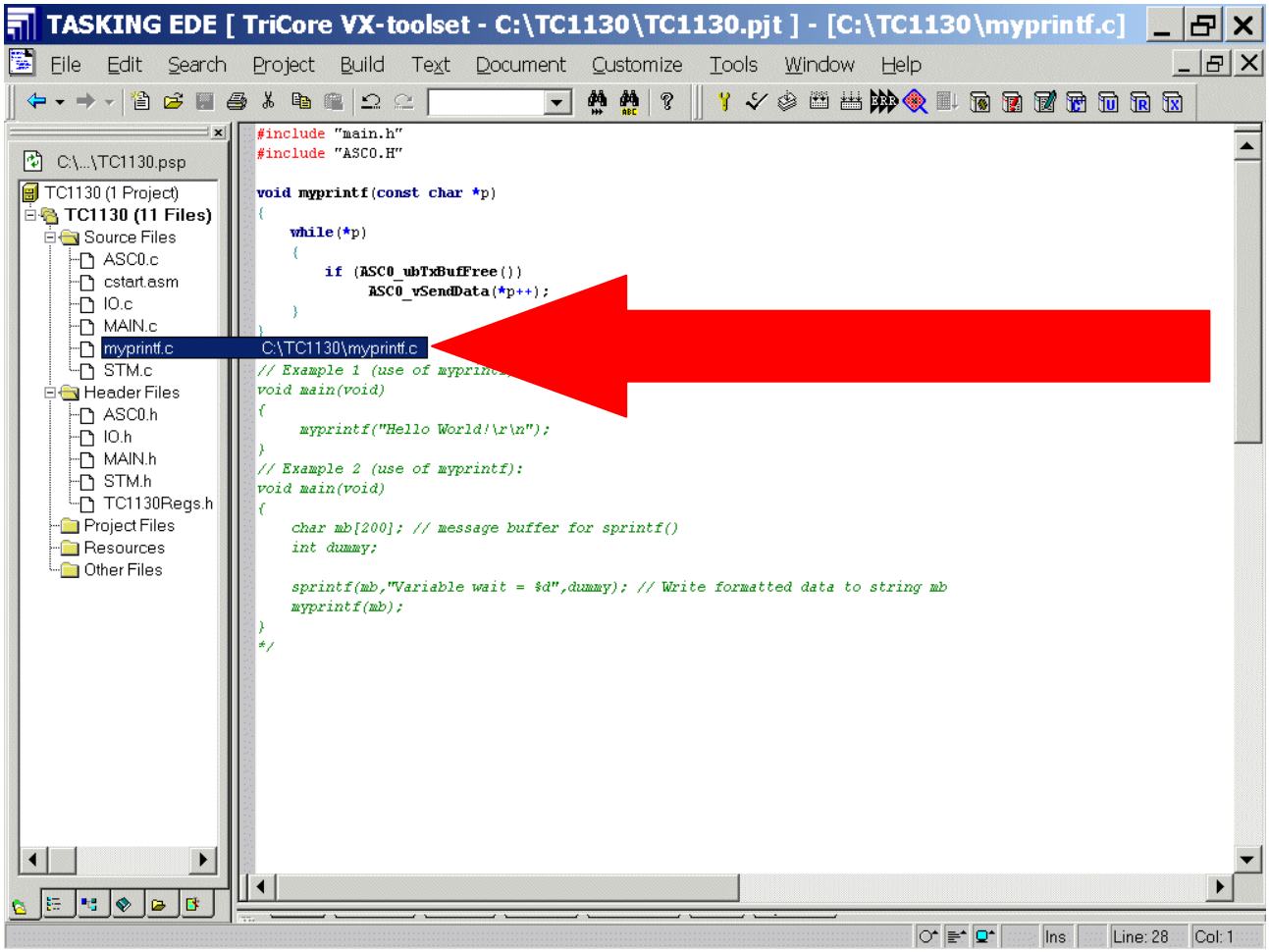
(Project Window File View) – TC1130 (Files) – right mouse button click – Add Existing Files – Browse



Select myprintf.c

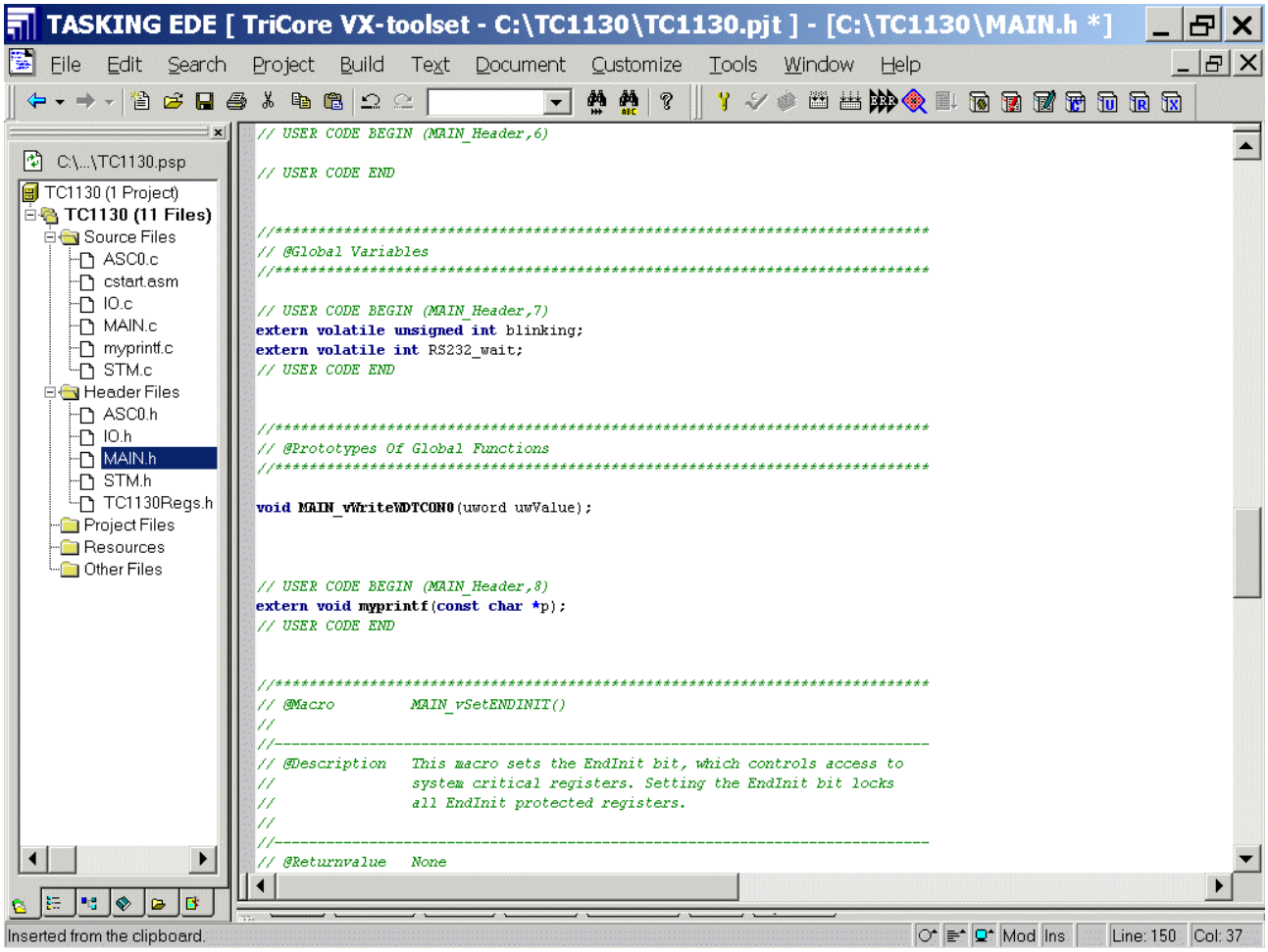


Open - OK


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\myprintf.c]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1130.psp
TC1130 (1 Project)
TC1130 (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
TC1130Regs.h
Project Files
Resources
Other Files
C:\TC1130\myprintf.c
#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);
}
*/
Ins
Line: 28
Col: 1

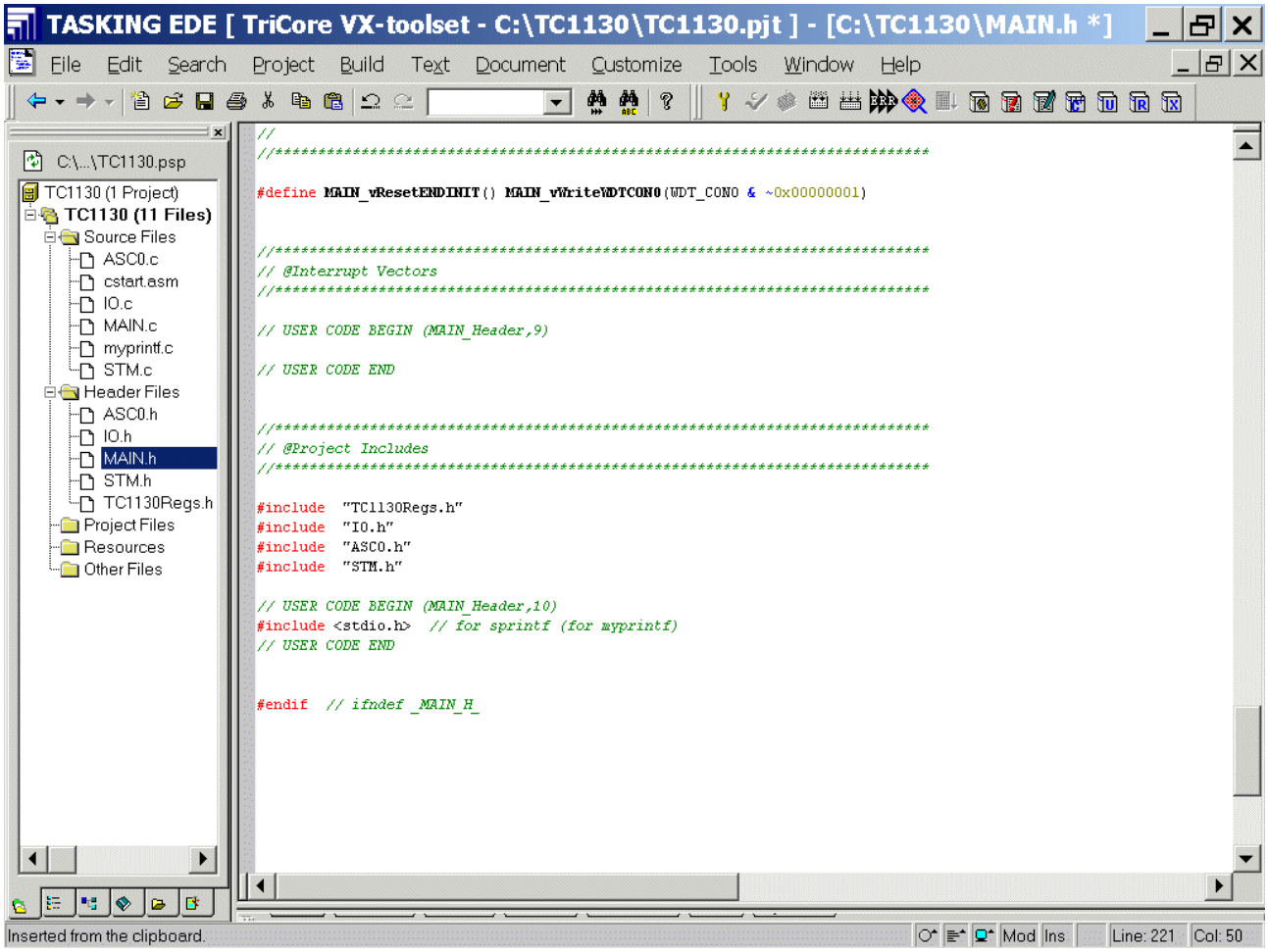
Double click: Main.h and insert Prototypes of Global Functions (Extern Declaration):

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



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

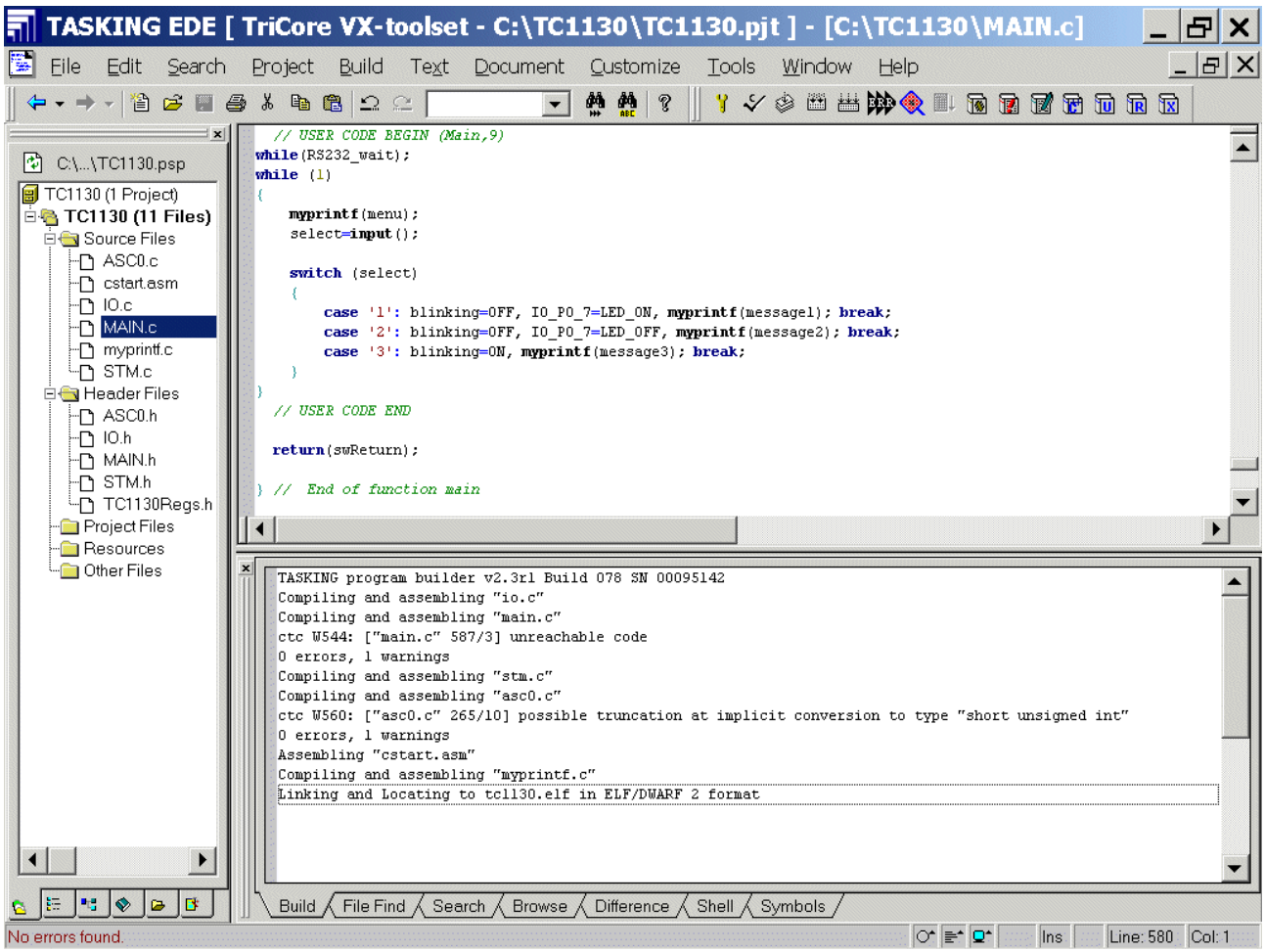
```
#include <stdio.h>     // for sprintf (for myprintf)
```



Generate your application program:

Build
Rebuild

or

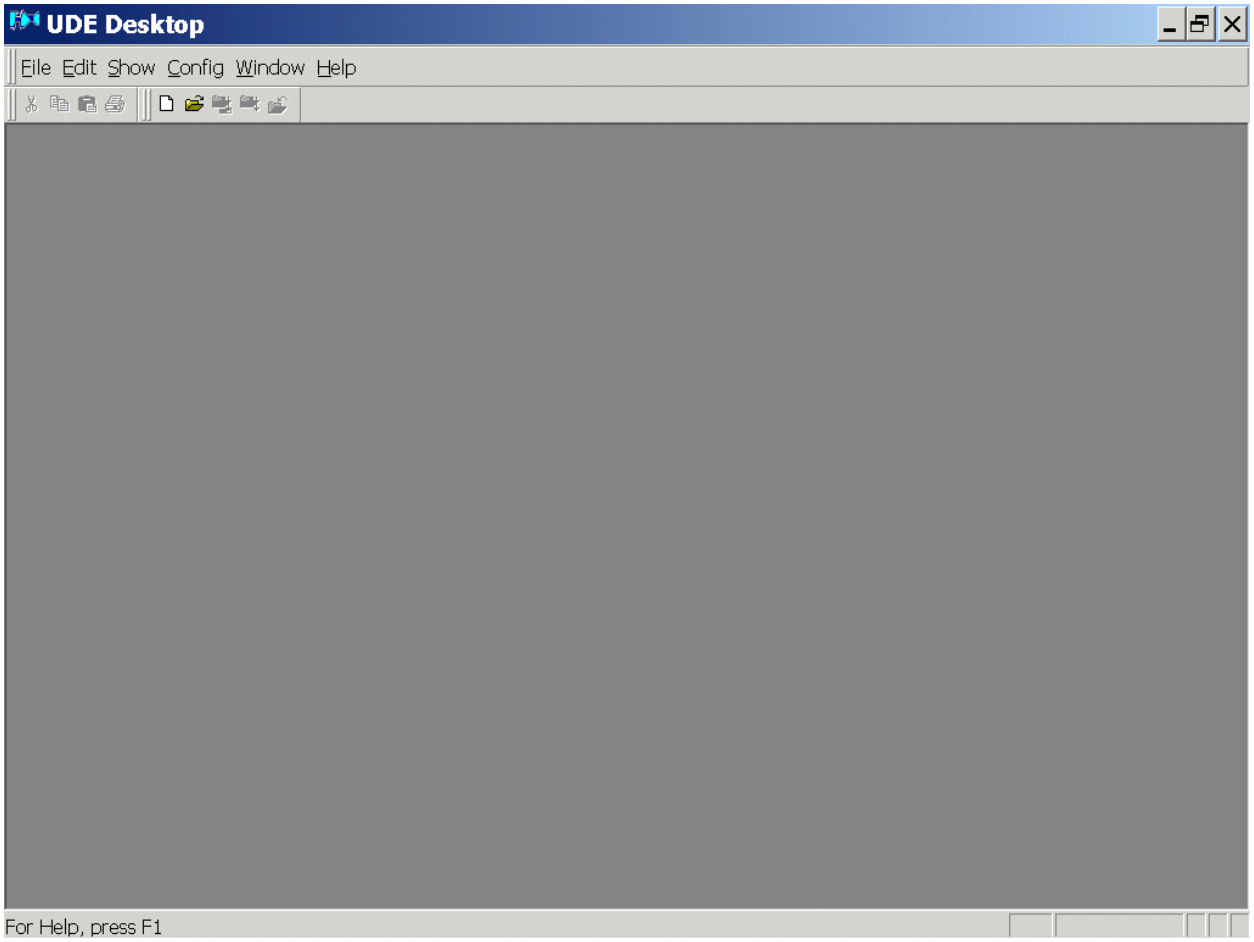


Now you can close both your project and Tasking EDE:

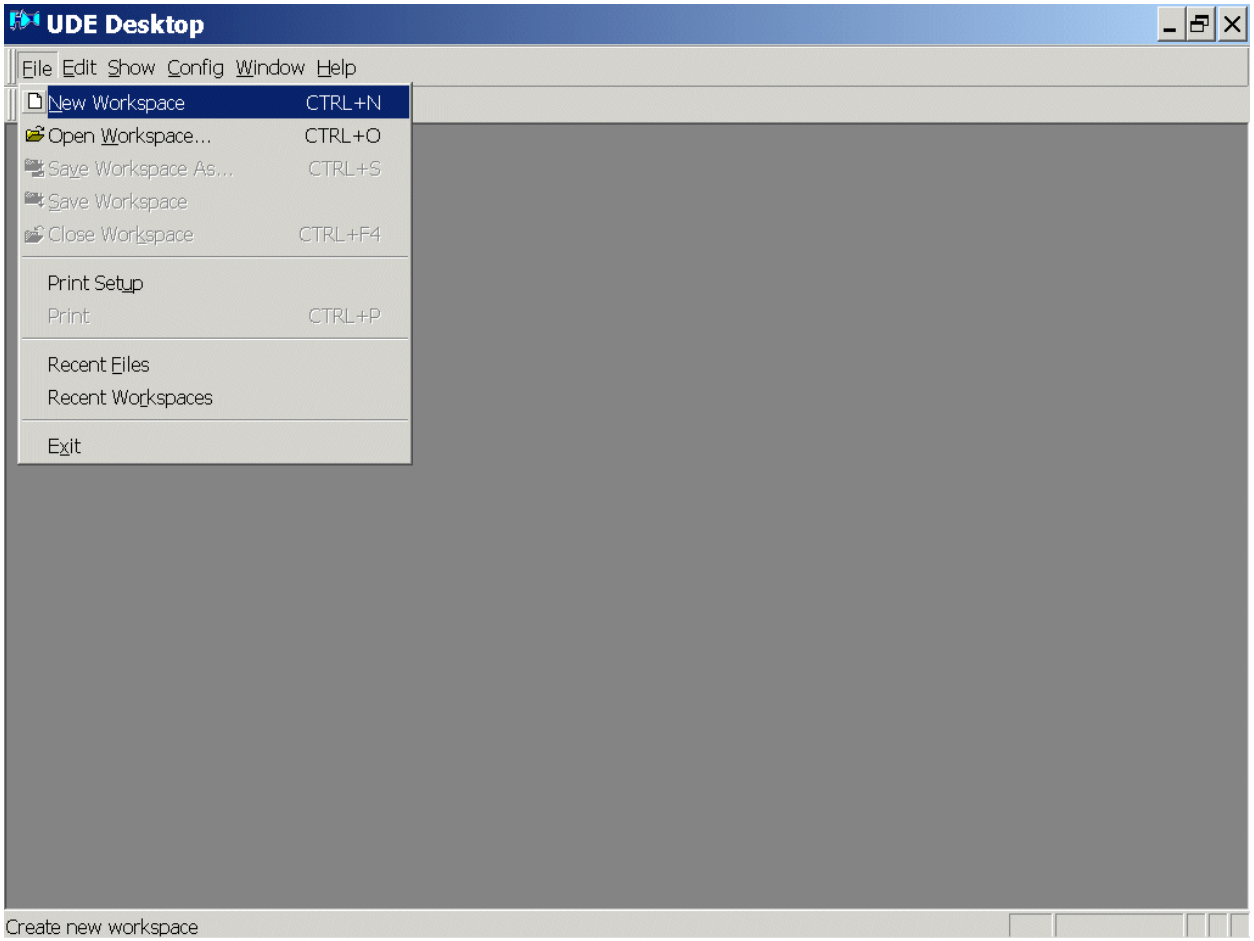
File - Close Project Space
File – Exit

Programming is now complete. You can now load and run your program
(Therefore we are going to use the pls Debugger):

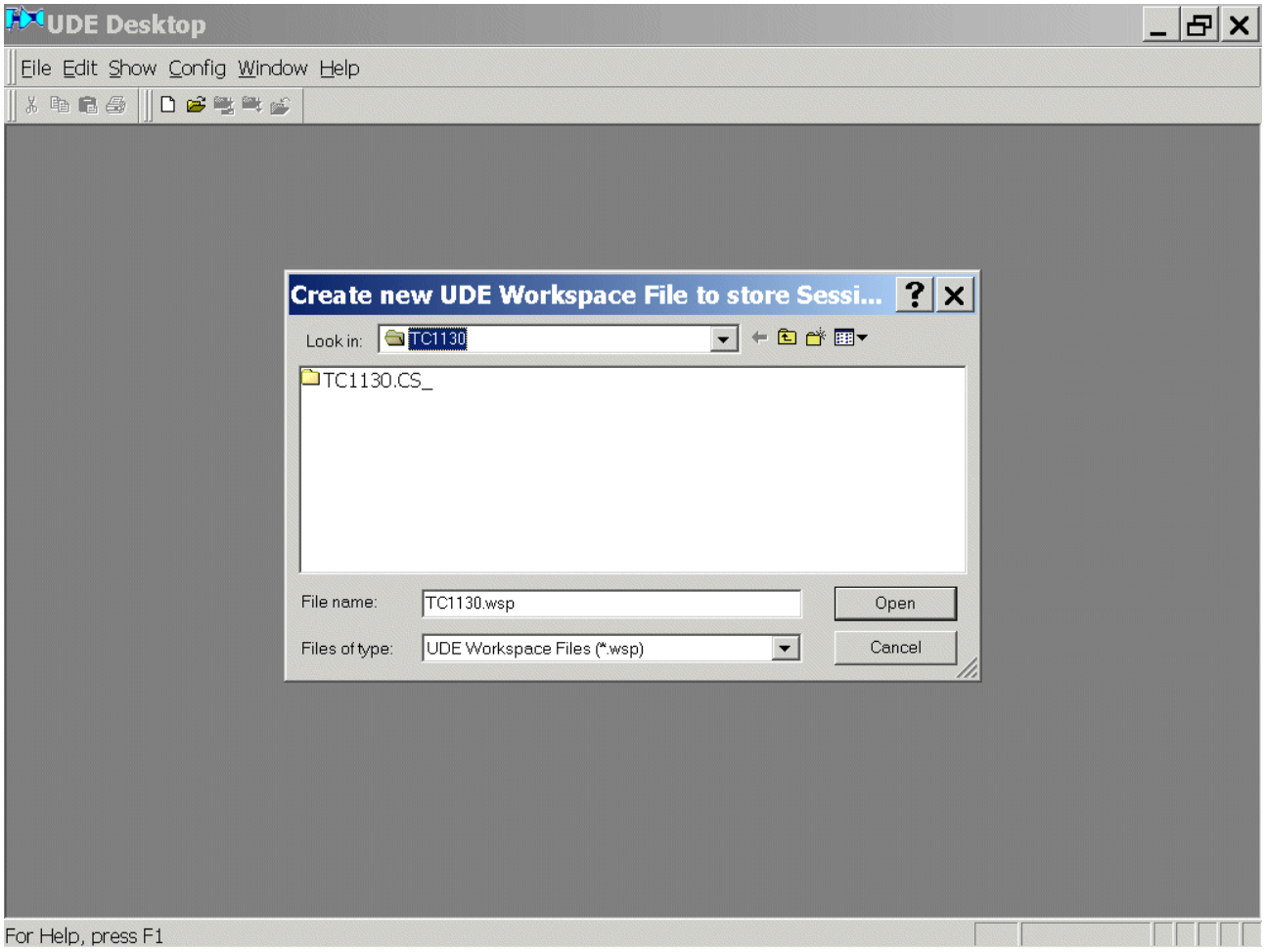
Start pls-Debugger



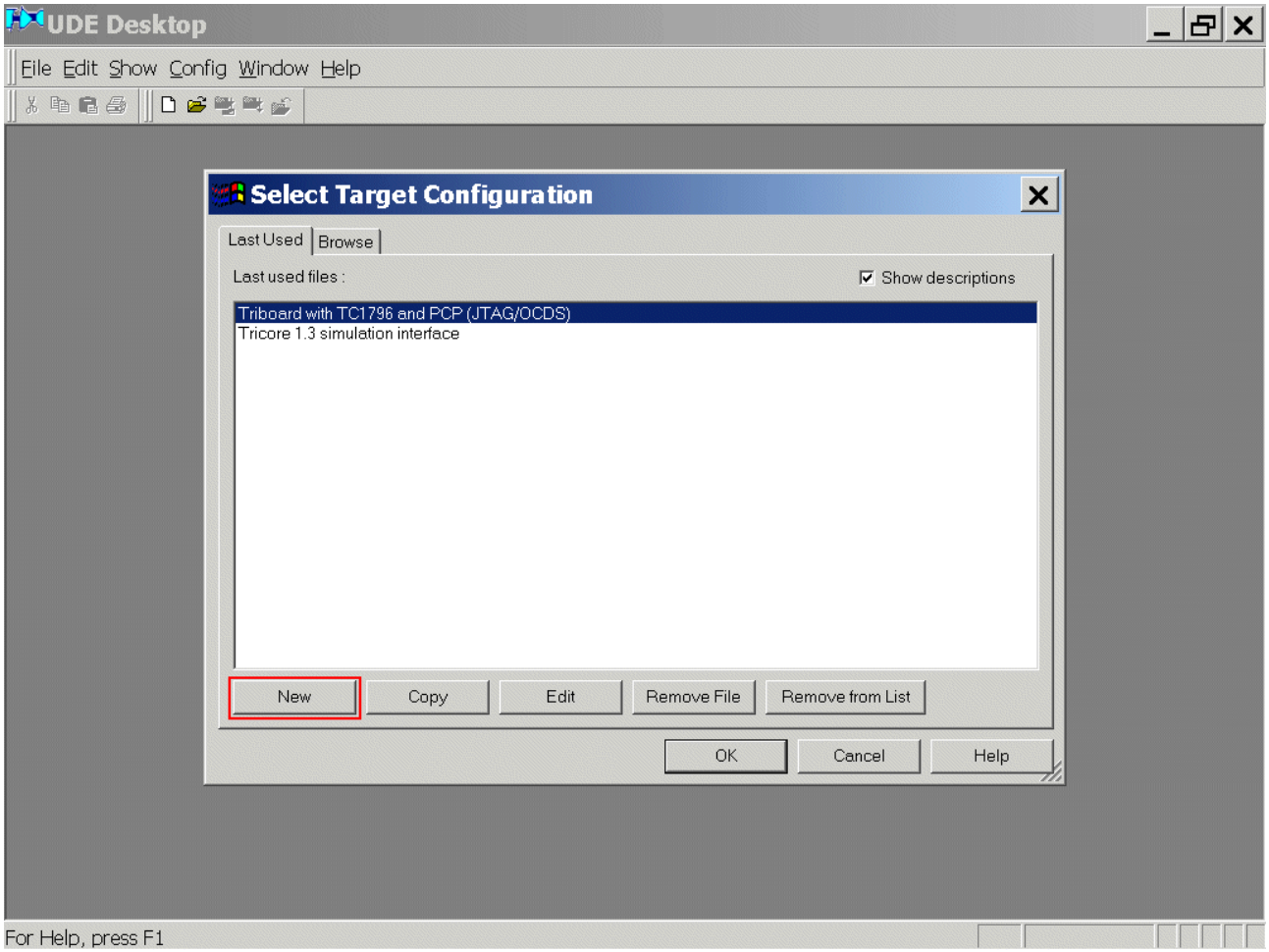
File – New Workspace



Look in: select C:\TC1130

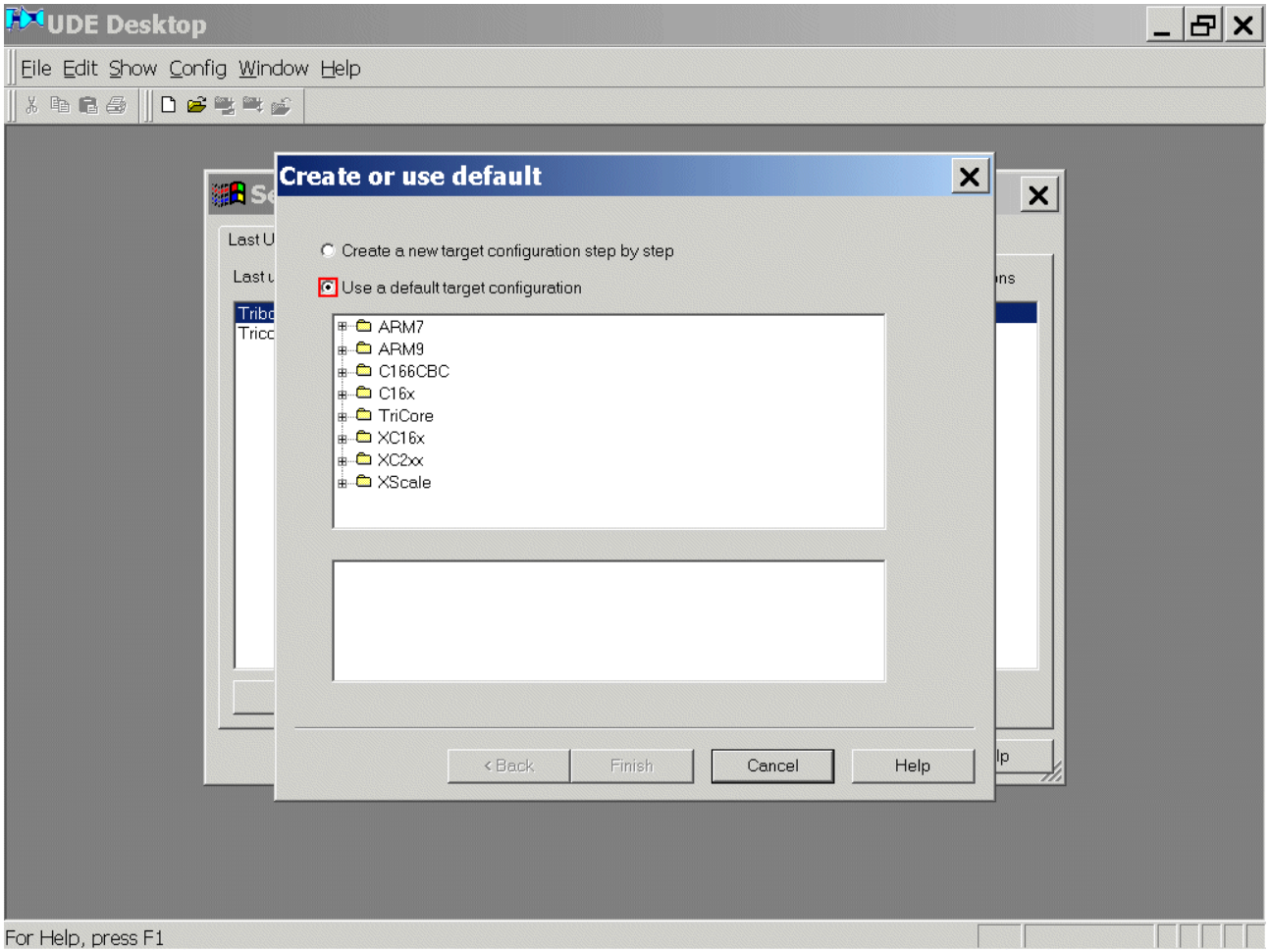


Open

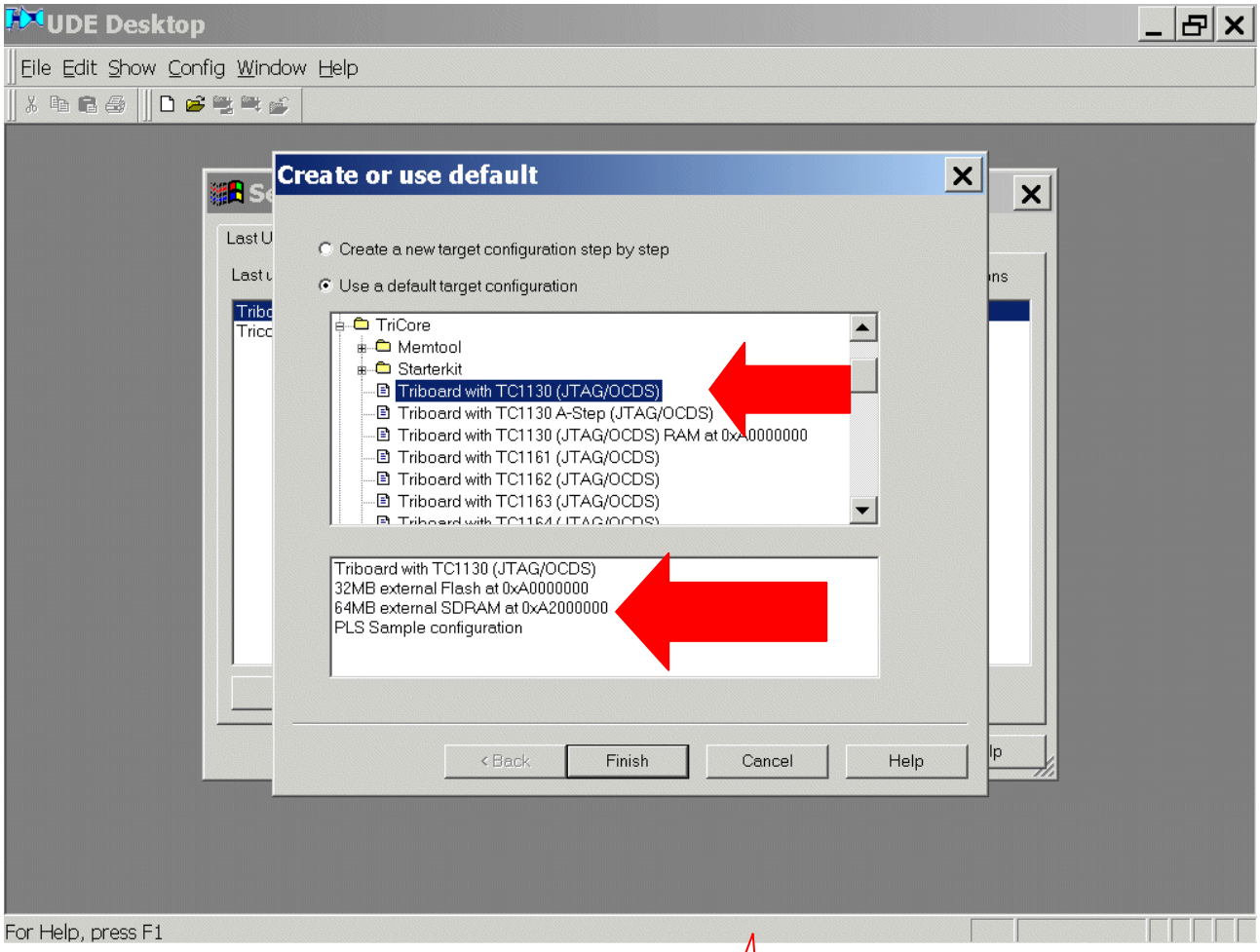

UDE Desktop
File Edit Show Config Window Help
Select Target Configuration
Last Used
Browse
Last used files :
Show descriptions
Triboard with TC1796 and PCP (JTAG/OCDS)
Tricore 1.3 simulation interface
New
Copy
Edit
Remove File
Remove from List
OK
Cancel
Help
For Help, press F1

New

Click ⊙ Use a default target configuration



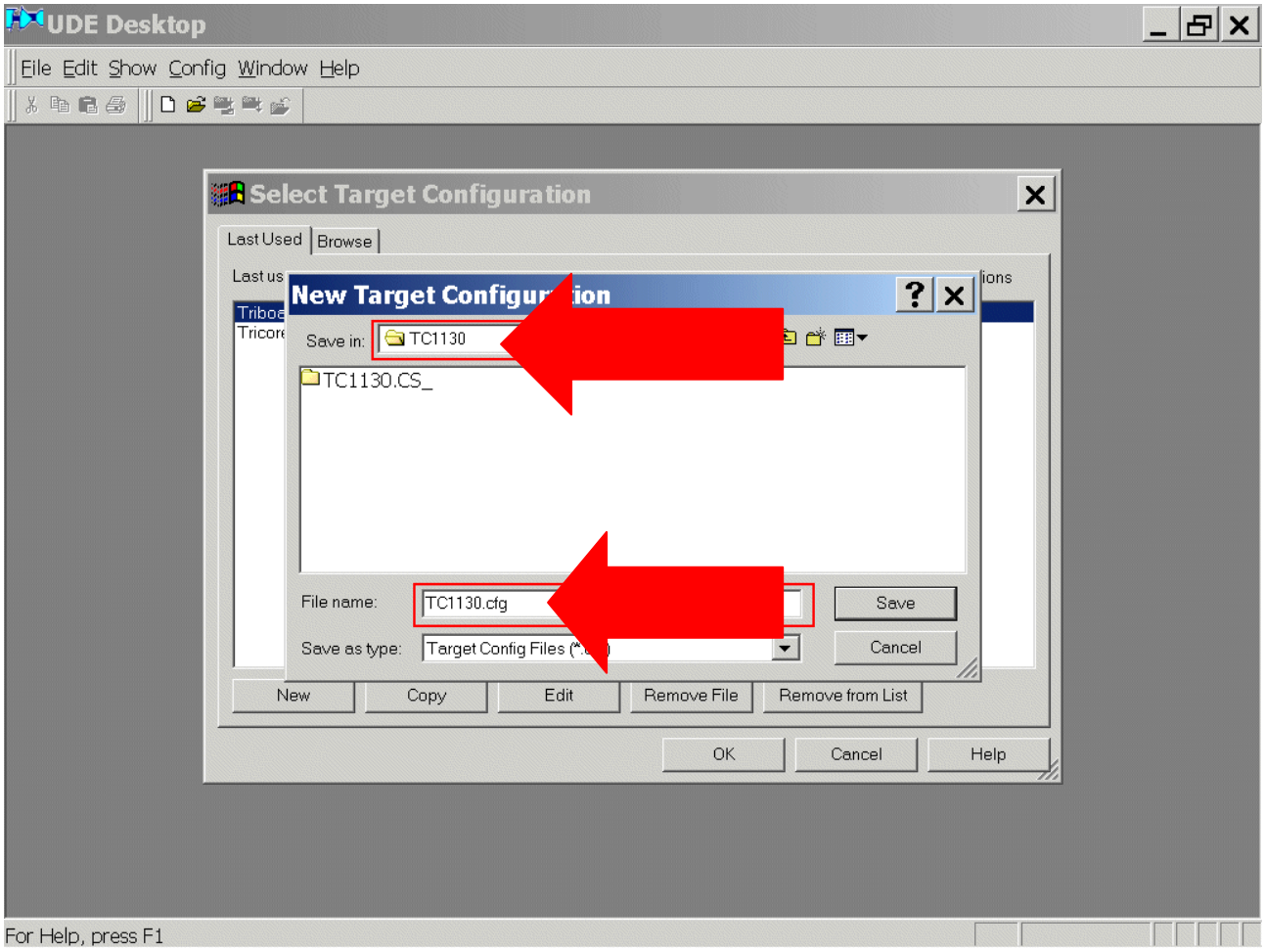
Select Triboard with TC1130 (JTAG/OCDS)



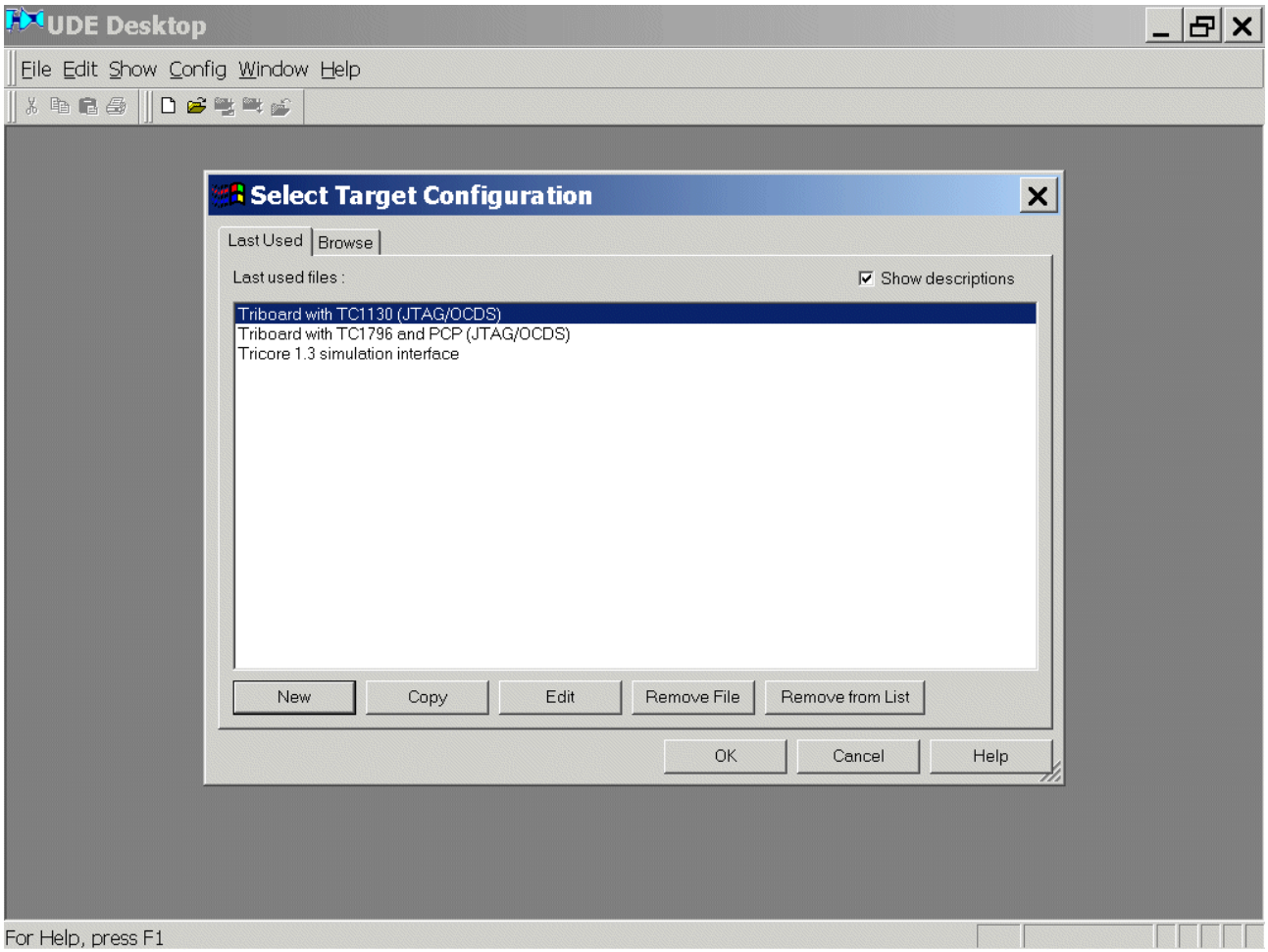
Finish

Note: The correct value for „64MB external SDRAM at 0xA2000000" will be 0xA4000000 later!

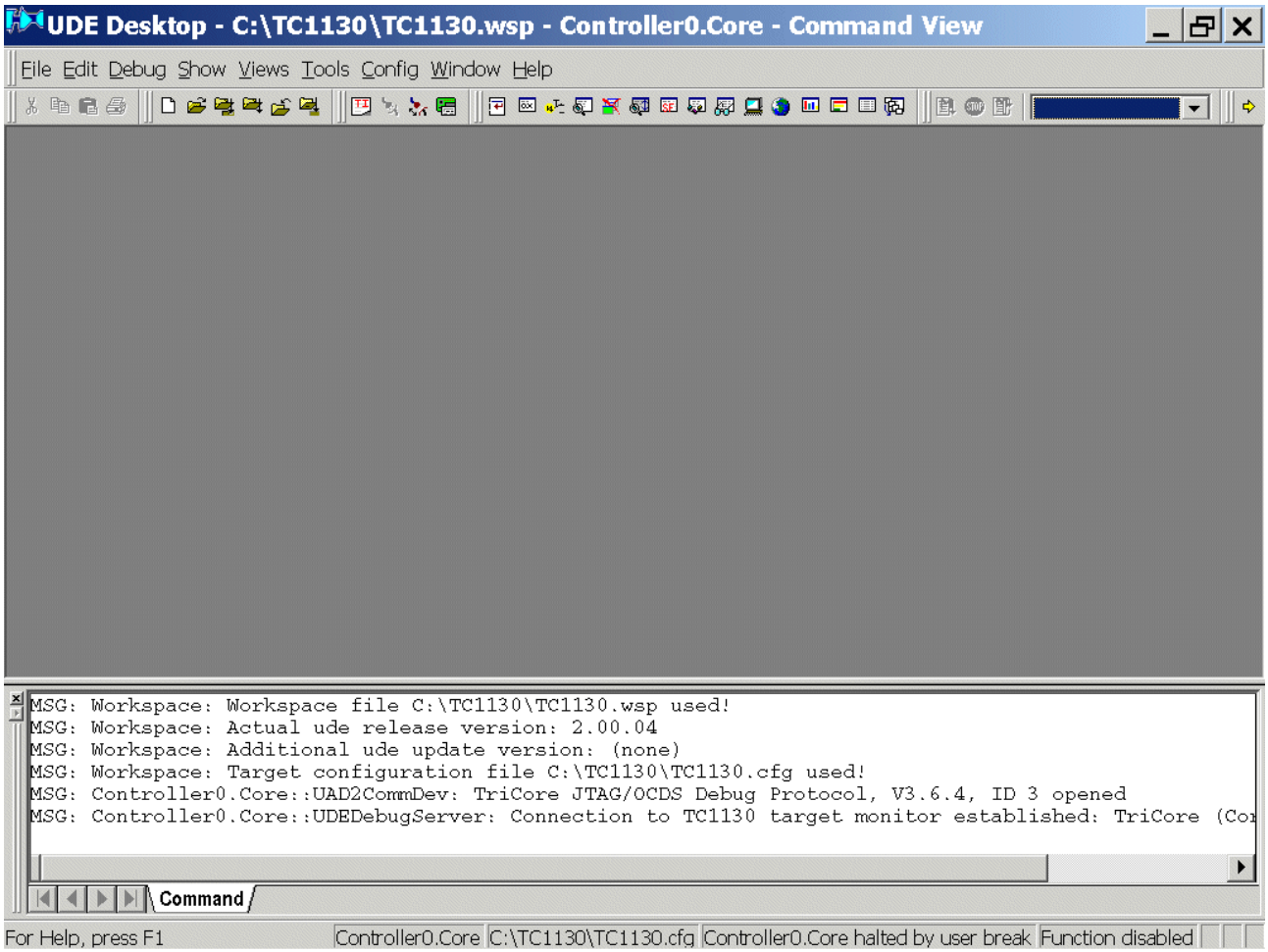
Save in: select C:\TC1130
File name: insert TC1130



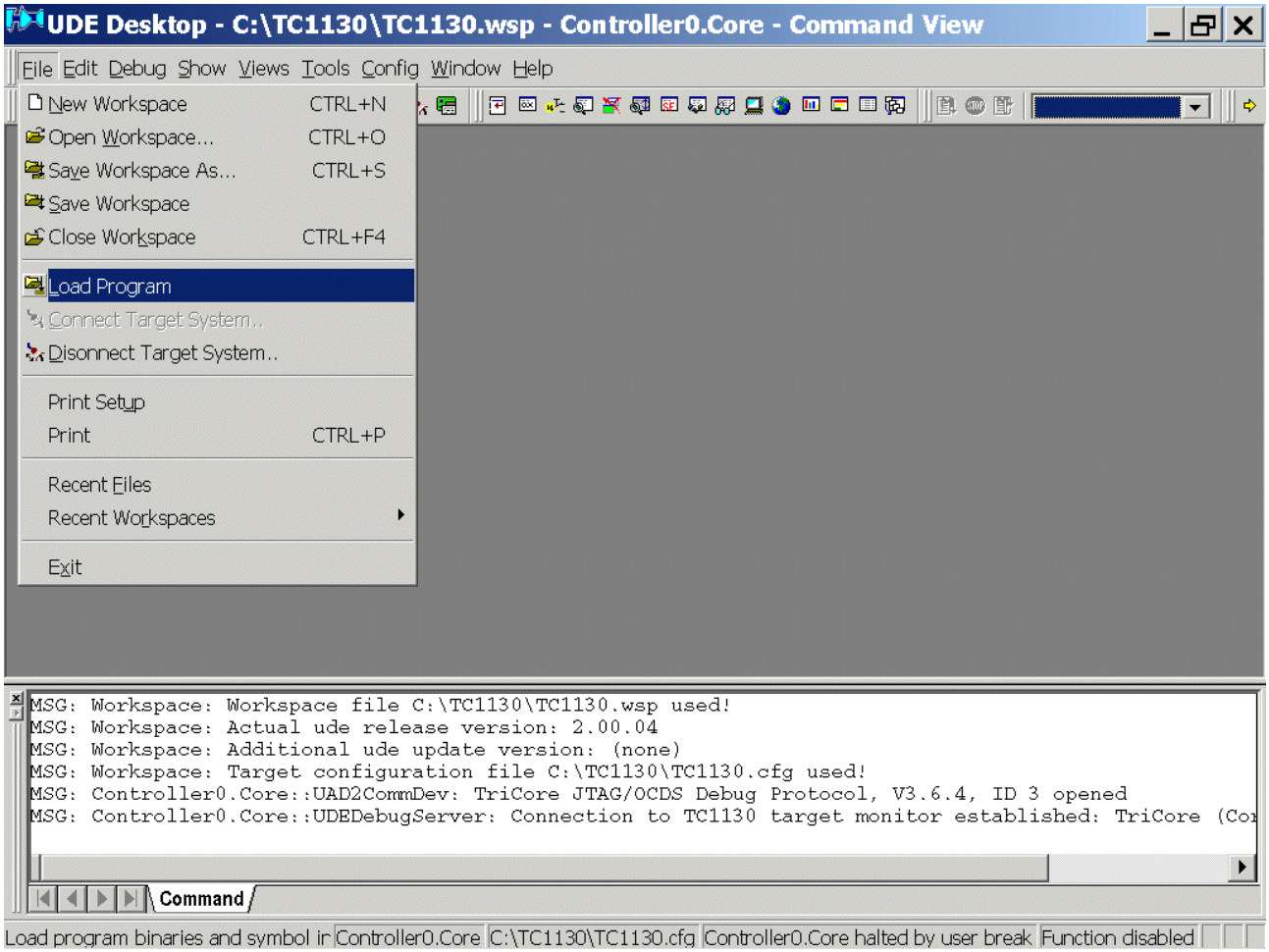
Save


UDE Desktop
File Edit Show Config Window Help
Select Target Configuration
Last Used
Browse
Last used files :
Show descriptions
Triboard with TC1130 (JTAG/OCDS)
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

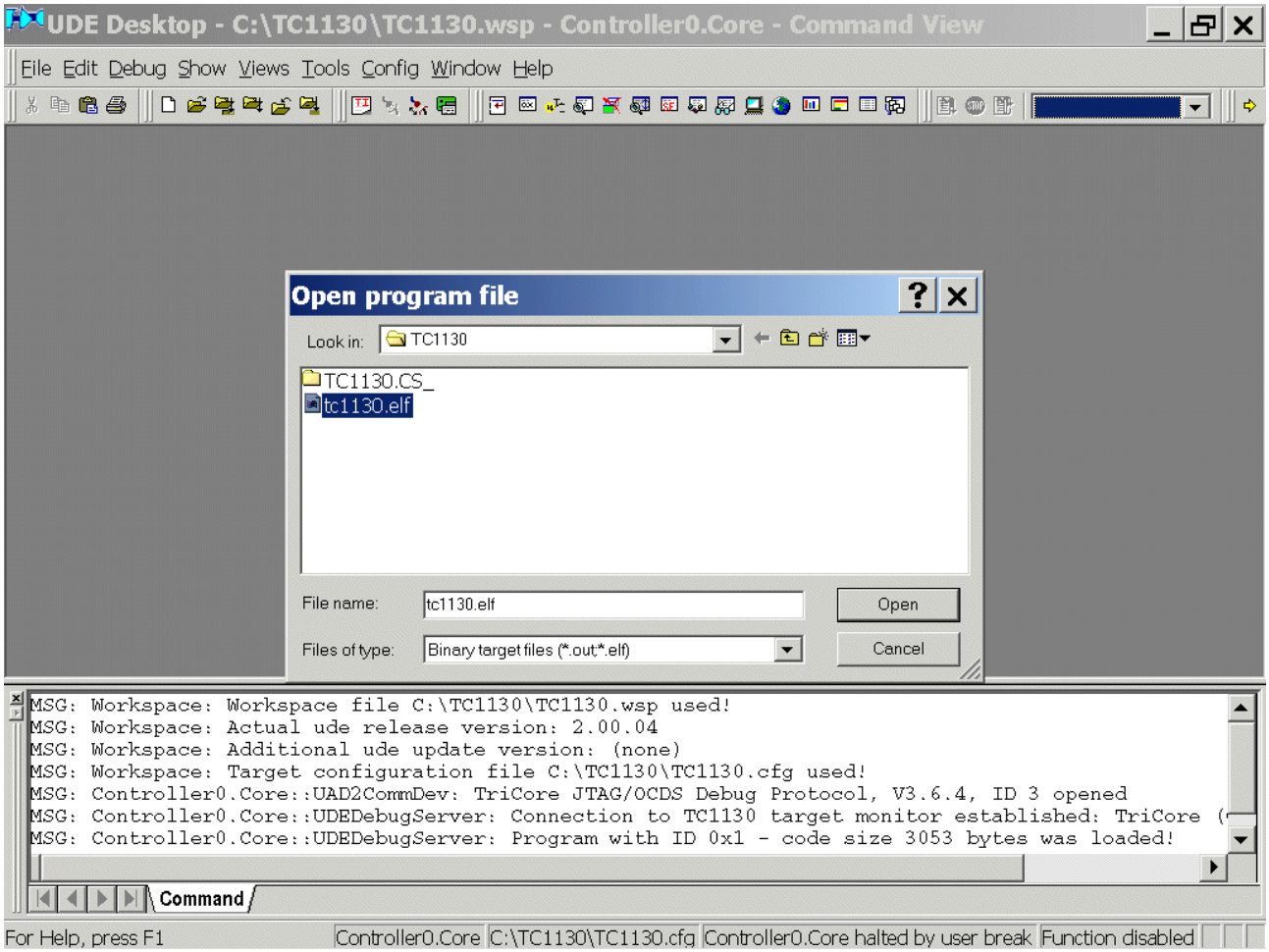
OK


UDE Desktop - C:\TC1130\TC1130.wsp - Controller0.Core - Command View
File Edit Debug Show Views Tools Config Window Help
MSG: Workspace: Workspace file C:\TC1130\TC1130.wsp used!
MSG: Workspace: Actual ude release version: 2.00.04
MSG: Workspace: Additional ude update version: (none)
MSG: Workspace: Target configuration file C:\TC1130\TC1130.cfg used!
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
MSG: Controller0.Core::UDEDebugServer: Connection to TC1130 target monitor established: TriCore (Co
Command
For Help, press F1
Controller0.Core
C:\TC1130\TC1130.cfg
Controller0.Core halted by user break
Function disabled

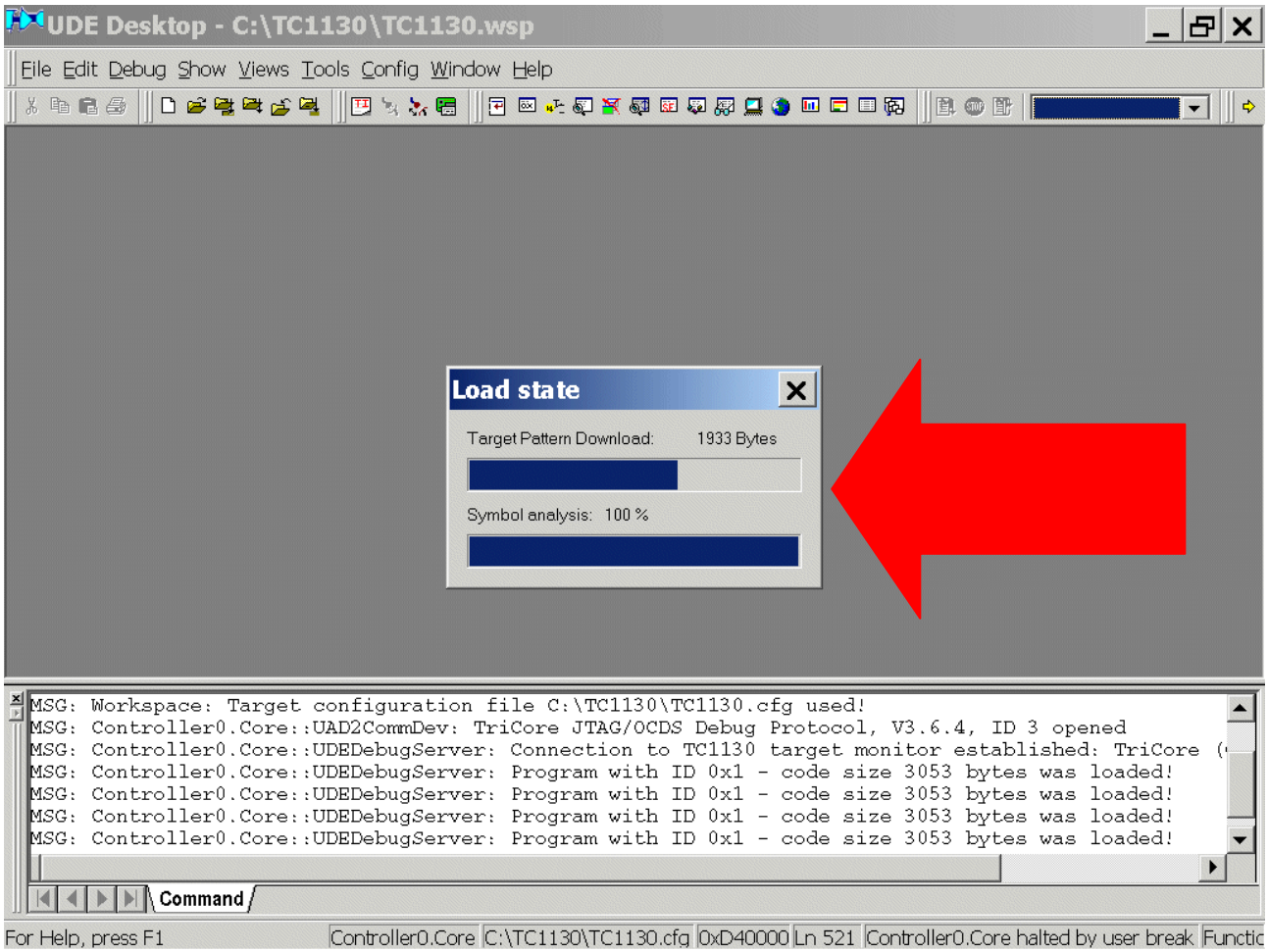
File – Load Program

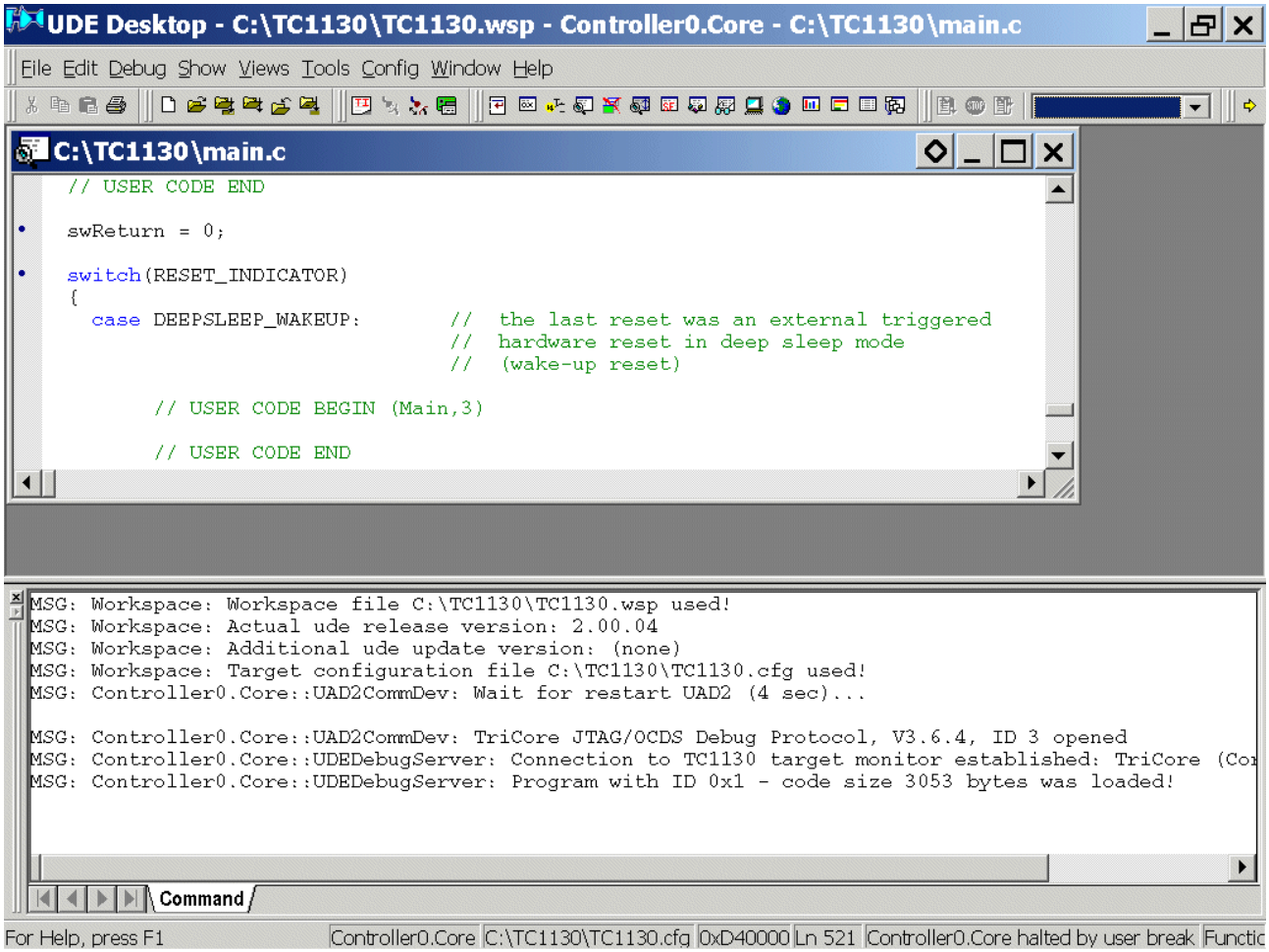


Select TC1130.elf

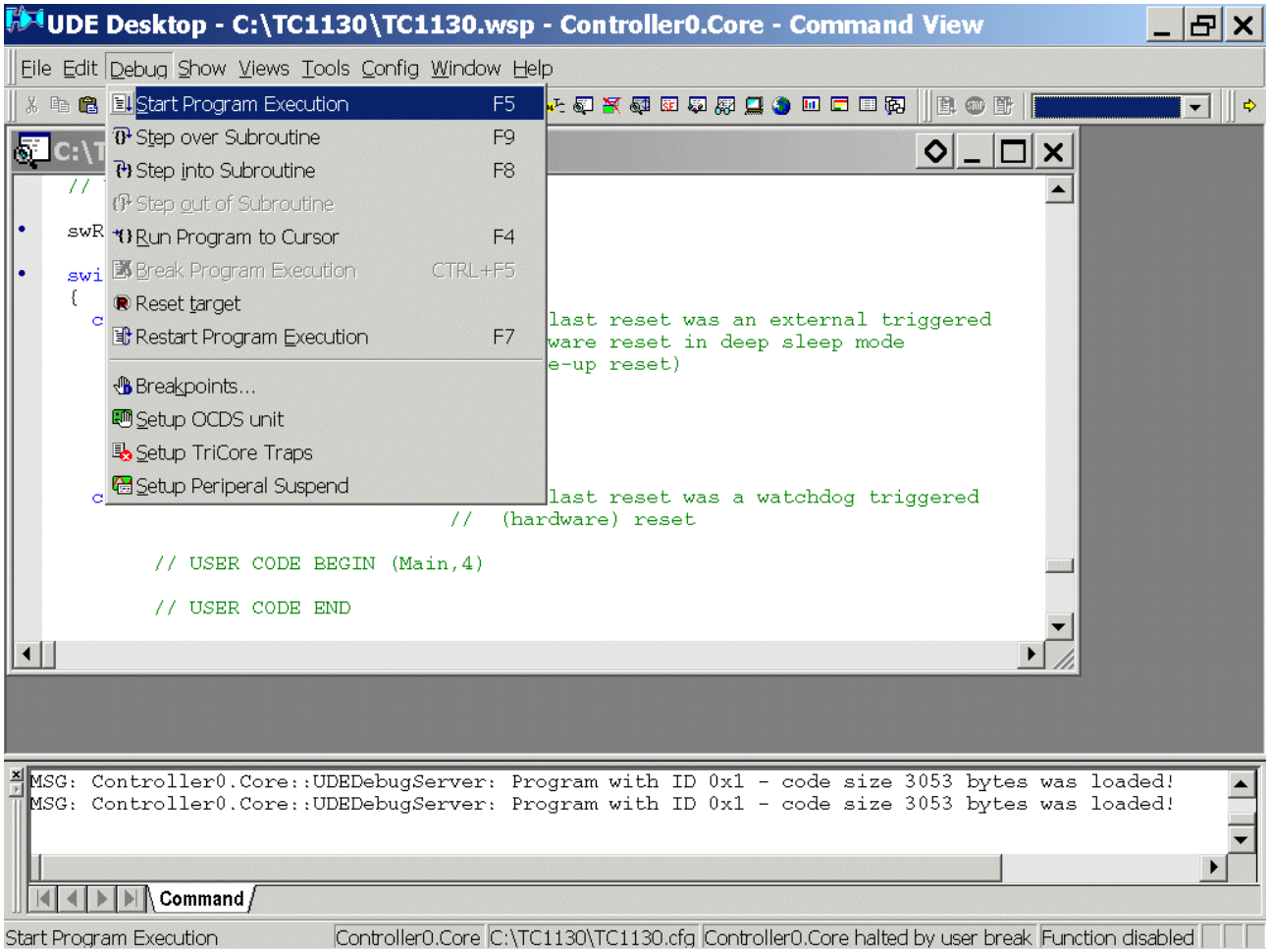


Open


UDE Desktop - C:\TC1130\TC1130.wsp
File Edit Debug Show Views Tools Config Window Help
Load state
Target Pattern Download: 1933 Bytes
Symbol analysis: 100 %
MSG: Workspace: Target configuration file C:\TC1130\TC1130.cfg used!
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
MSG: Controller0.Core::UDEDebugServer: Connection to TC1130 target monitor established: TriCore
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3053 bytes was loaded!
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3053 bytes was loaded!
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3053 bytes was loaded!
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3053 bytes was loaded!
Command
For Help, press F1
Controller0.Core
C:\TC1130\TC1130.cfg
0xD40000
Ln 521
Controller0.Core halted by user break

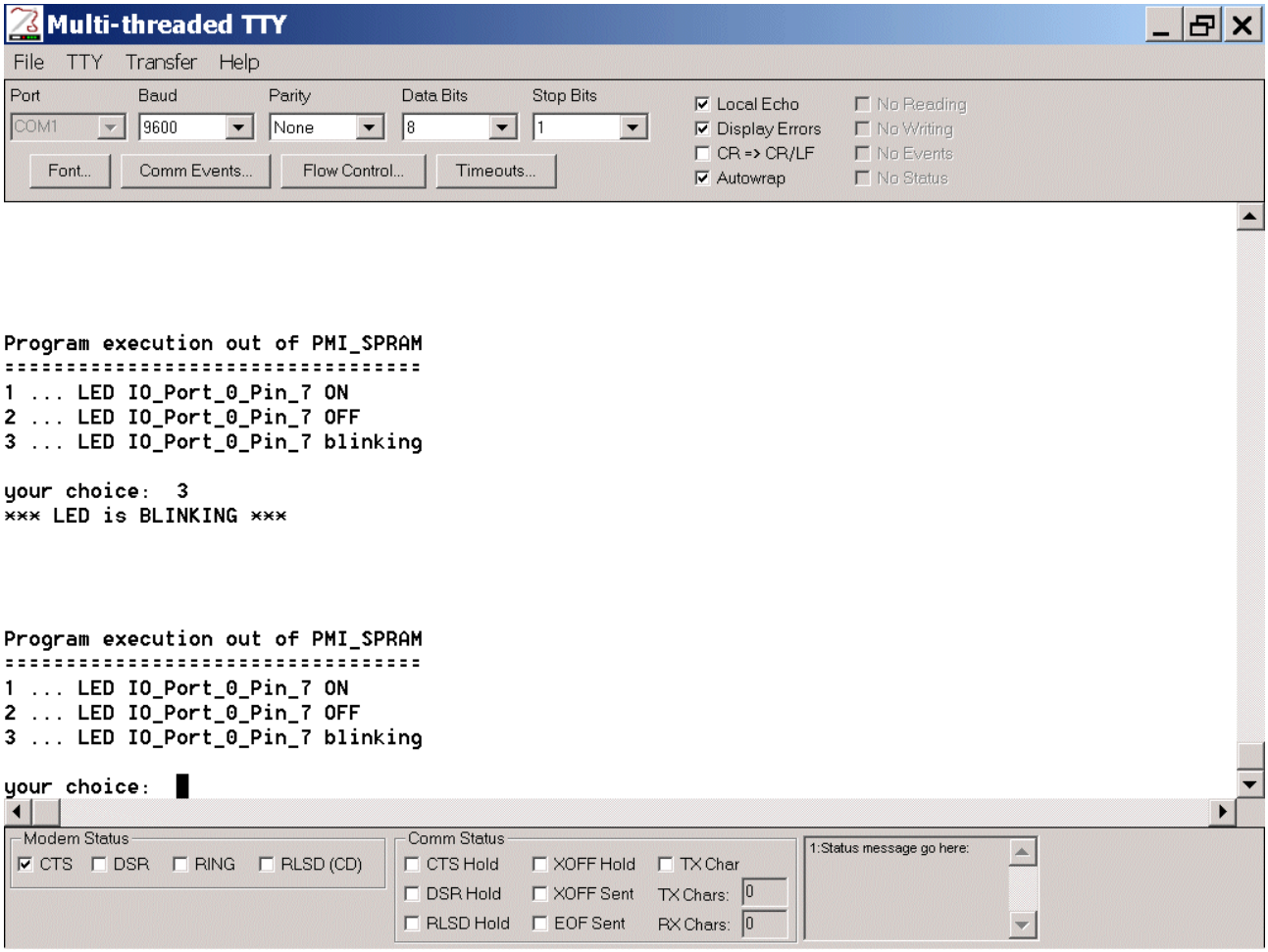

UDE Desktop - C:\TC1130\TC1130.wsp - Controller0.Core - C:\TC1130\main.c
File Edit Debug Show Views Tools Config Window Help
C:\TC1130\main.c
// USER CODE END
swReturn = 0;
switch(RESET_INDICATOR)
{
case DEEPSLEEP_WAKEUP: // the last reset was an external triggered
// hardware reset in deep sleep mode
// (wake-up reset)
// USER CODE BEGIN (Main,3)
// USER CODE END
MSG: Workspace: Workspace file C:\TC1130\TC1130.wsp used!
MSG: Workspace: Actual ude release version: 2.00.04
MSG: Workspace: Additional ude update version: (none)
MSG: Workspace: Target configuration file C:\TC1130\TC1130.cfg used!
MSG: Controller0.Core::UAD2CommDev: Wait for restart UAD2 (4 sec)...
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
MSG: Controller0.Core::UDEDebugServer: Connection to TC1130 target monitor established: TriCore (Co
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3053 bytes was loaded!
Command
For Help, press F1
Controller0.Core
C:\TC1130\TC1130.cfg
0xD40000
Ln 521
Controller0.Core halted by user break
Functic

Debug – Start Program Execution



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

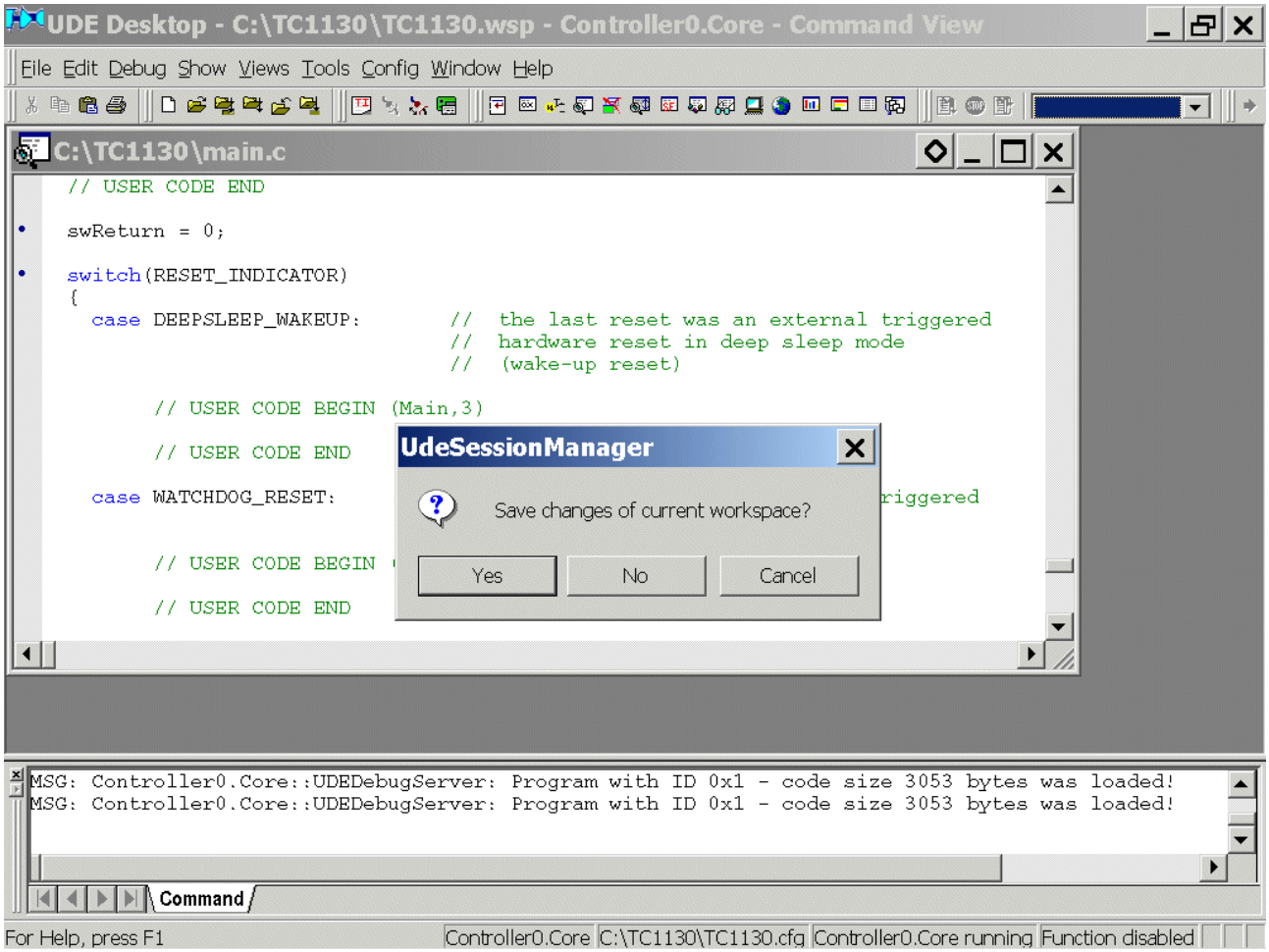
And see the result:



Note:

As an example for “any terminal-program” we use mttty.
MTTY can be downloaded @ http://www.freeware.de/software/DetailEN_MTTTY_19383.html

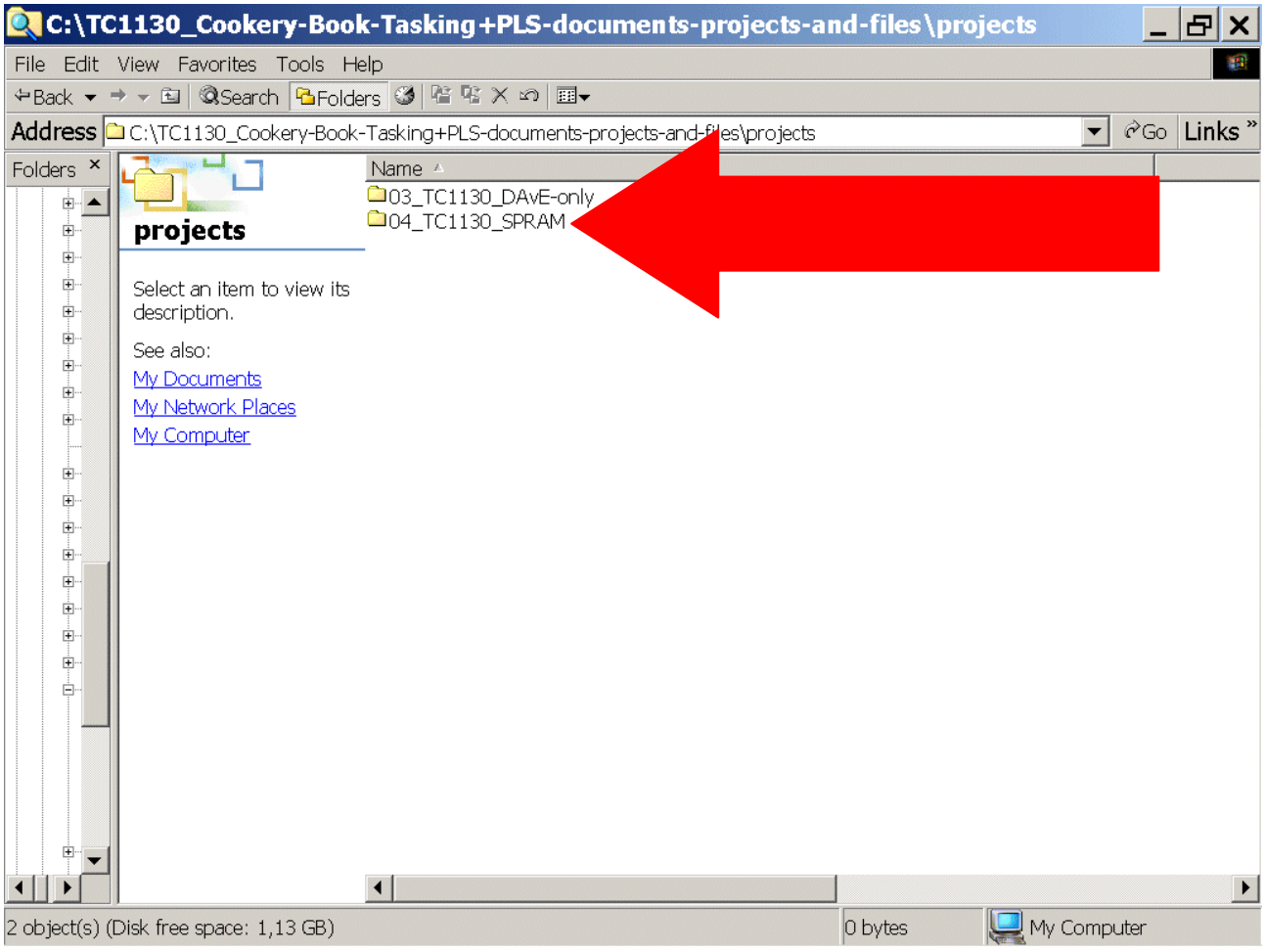
File – Close Workspace



Yes

File - Exit

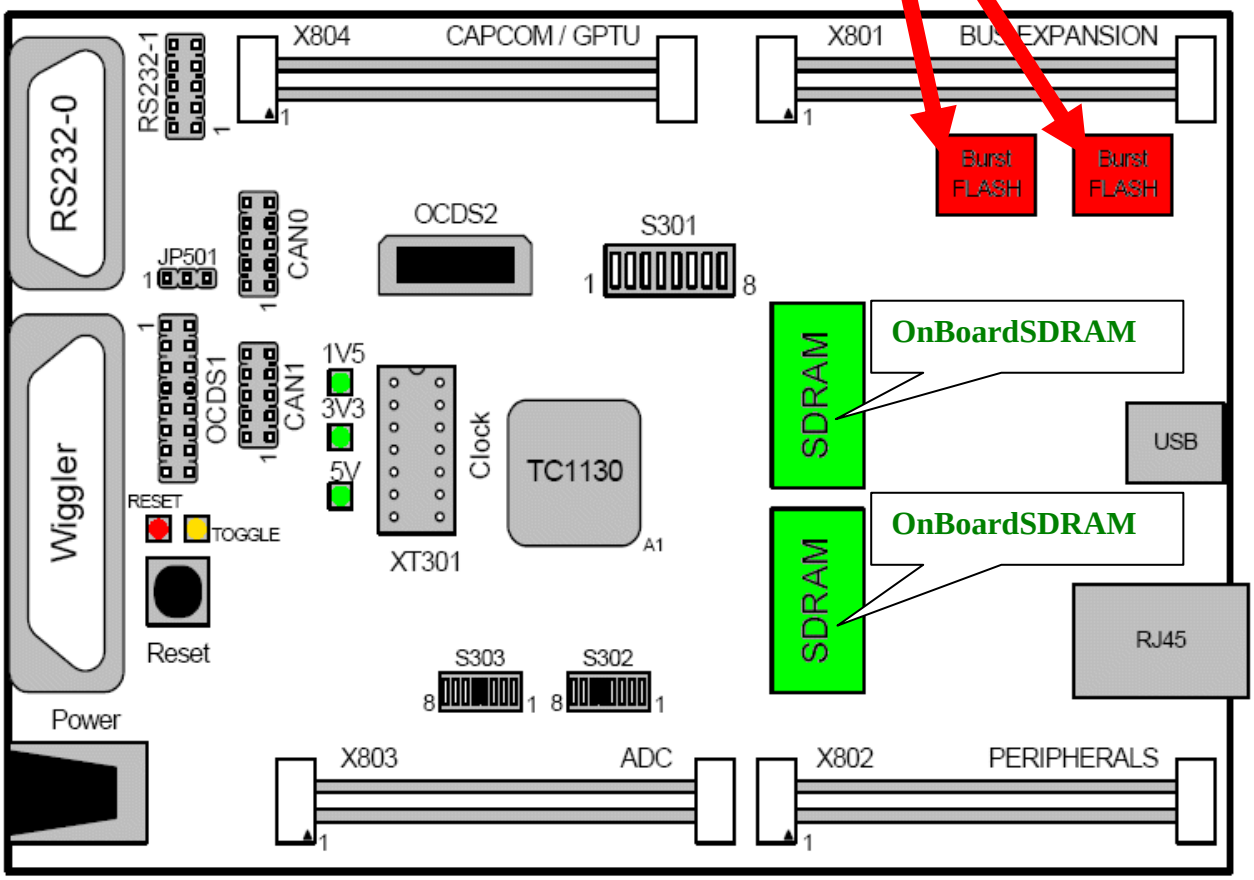
We recommend now to copy and store your project-directory “C:\TC1130” to “04_TC1130_SPRAM”:



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

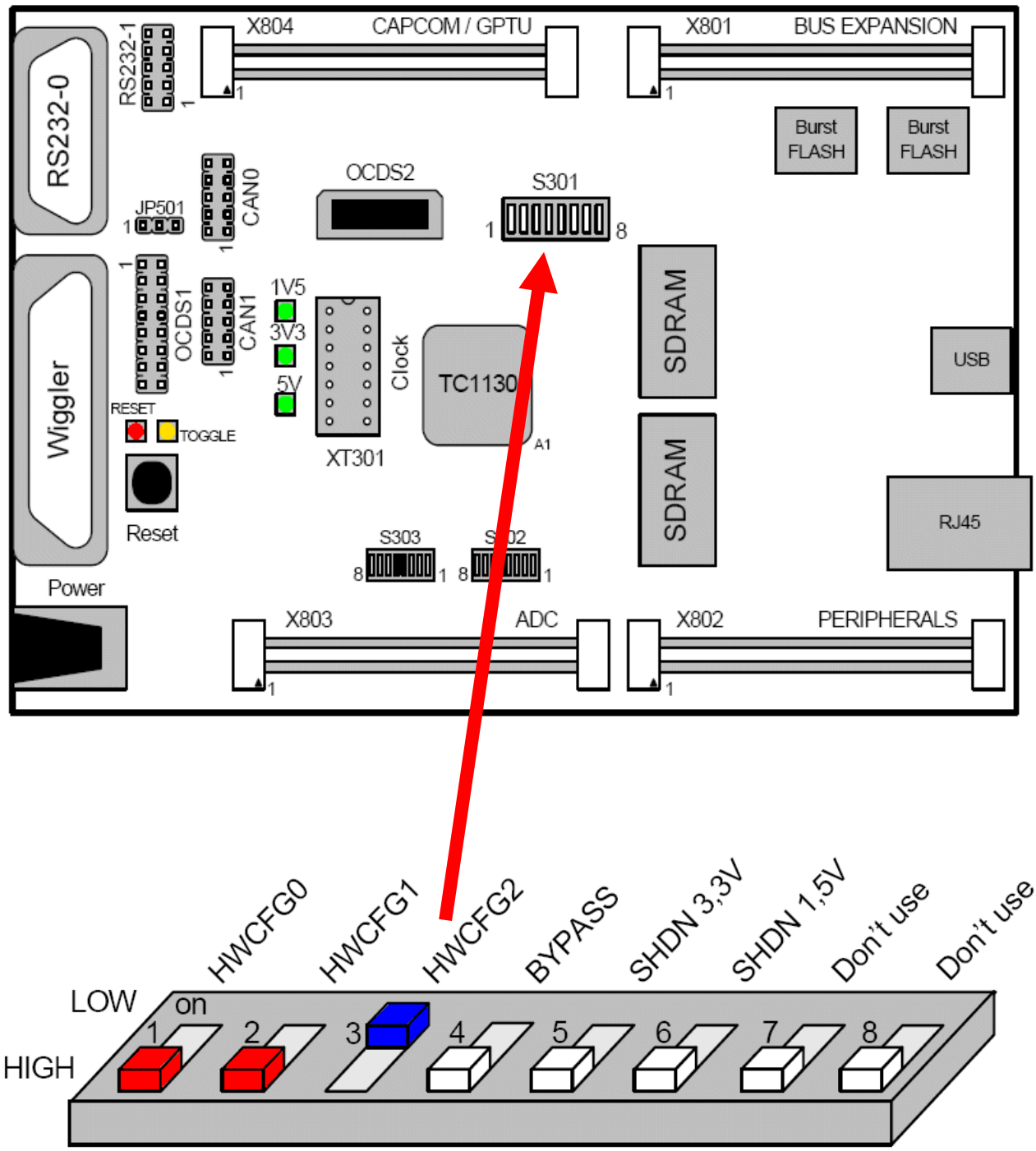
"ROM":
Locating programs into the Intel's 32 MBytes OnBoardFlash,
"RAM":
Using
OnChipSRAM (DMI_SPRAM+DMU_SRAM) + Infineon's 64 MBytes OnBoardSDRAM

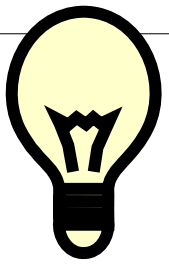
Write programs for execution from OnBoardFlash



TC1130-Execution-Environment: OnBoardFlash  :

Jumper S301 Settings (HW-Configuration DIP-Switch):





Additional Information:

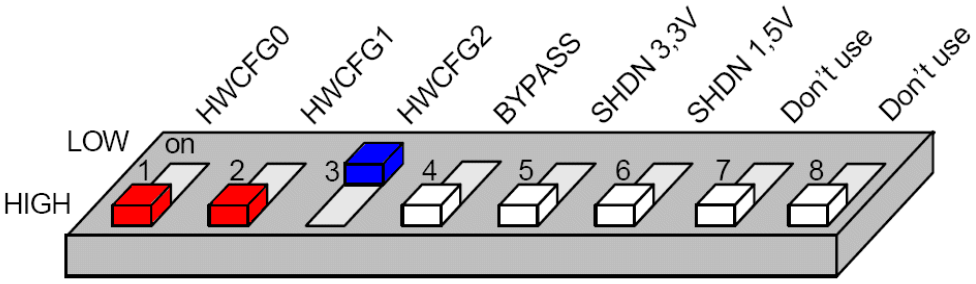
Adobe Reader - [TC1130 TriBoard Manual.pdf]

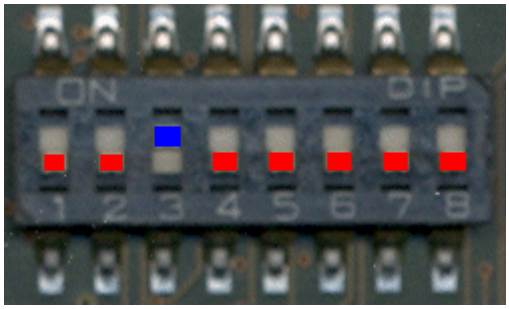
File Edit View Document Tools Window Help

/BRK_IN	CFG[2...0]	Type of Boot	PC Start value
1	000	Serial boot from ASC to PMI scratchpad, run loaded program	0xD4000000
1	001	Serial boot from CAN to PMI scratchpad, run loaded program	0xDFFFFFFC
1	010	Serial boot from SSC to PMI scratchpad, run loaded program	0xDFFFFFFC
1	011	External memory, EBU as master	0xA0000000
1	100	External memory, EBU as slave	0xA0000000
1	101	External memory, EBU as master	0xA0000000
1	110	PMI scratchpad	0xD4000000
1	111	reserved; don't use this combination	-
0	01x 1xx	reserved; don't use this combination	-
0	001	go to external emulator space	0xDE000000
0	000	put chip in tristate (deep sleep)	-

7,81 x 11,07 in

24 of 64





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

Additional Information:

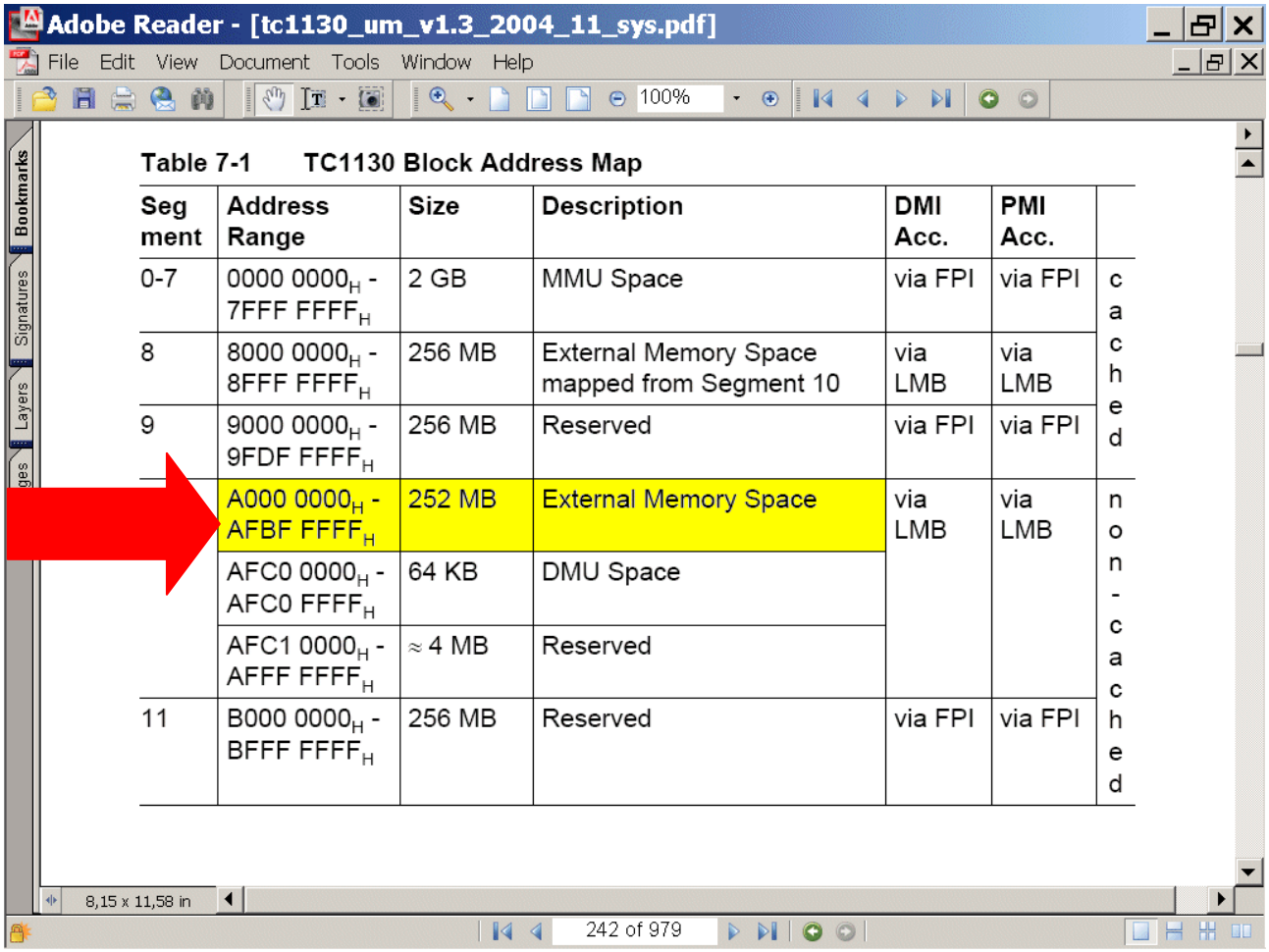
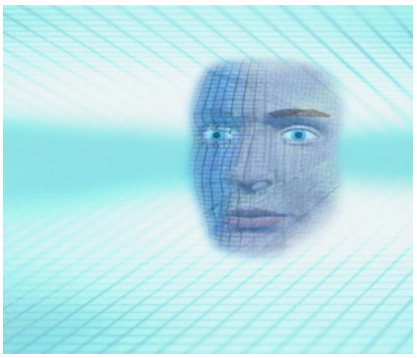


Table 7-1 TC1130 Block Address Map

Segment	Address Range	Size	Description	DMI Acc.	PMI Acc.	
0-7	0000 0000$_H$ - 7FFF FFFF$_H$	2 GB	MMU Space	via FPI	via FPI	cached
8	8000 0000$_H$ - 8FFF FFFF$_H$	256 MB	External Memory Space mapped from Segment 10	via LMB	via LMB	
9	9000 0000$_H$ - 9FDF FFFF$_H$	256 MB	Reserved	via FPI	via FPI	
	A000 0000$_H$ - AFBF FFFF$_H$	252 MB	External Memory Space	via LMB	via LMB	non-cached
	AFC0 0000$_H$ - AFC0 FFFF$_H$	64 KB	DMU Space			
	AFC1 0000$_H$ - AFFF FFFF$_H$	≈ 4 MB	Reserved			
11	B000 0000$_H$ - BFFF FFFF$_H$	256 MB	Reserved	via FPI	via FPI	

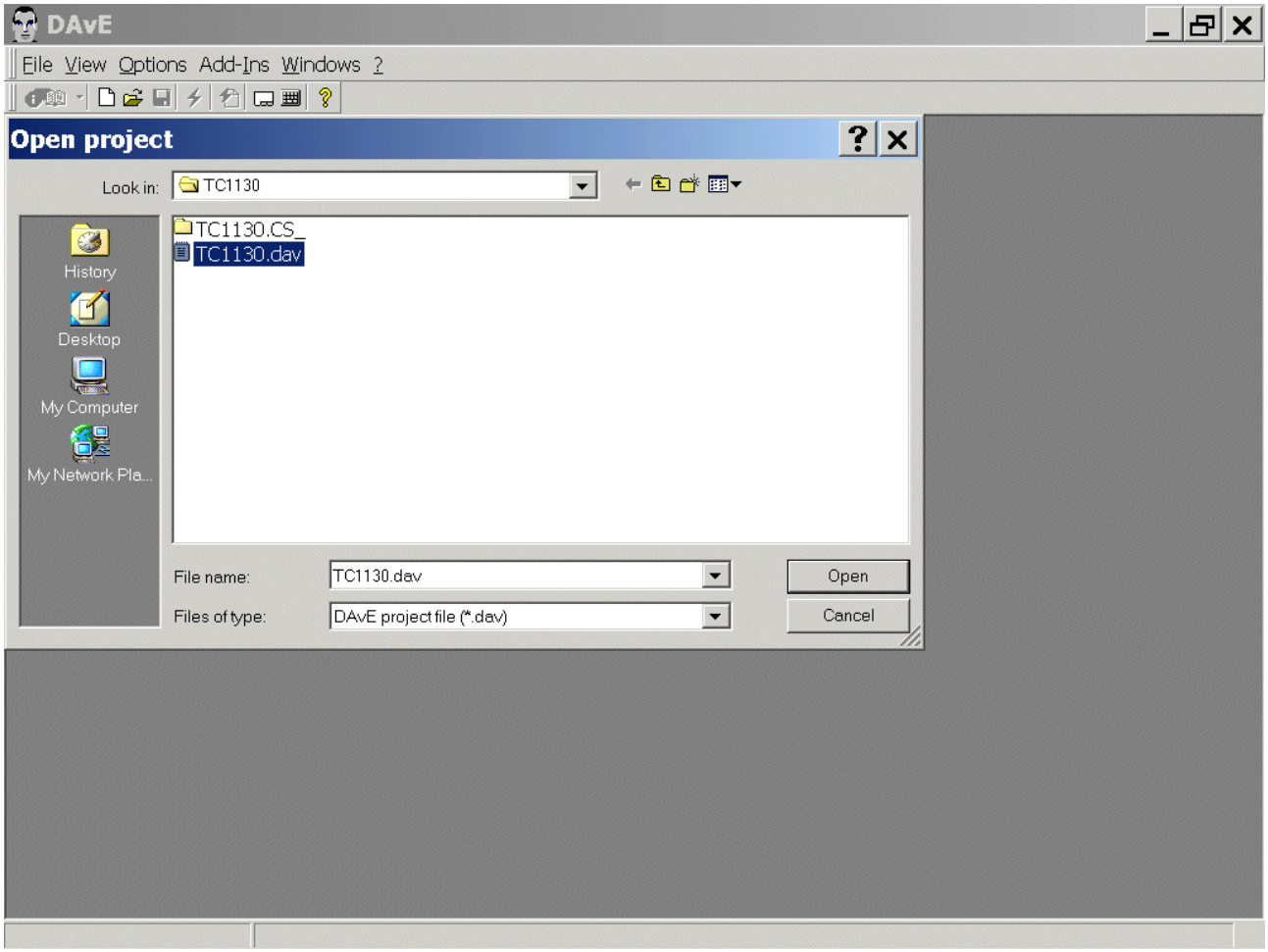
252 MBytes External Memory Space used for:
"ROM": 32 MBytes OnBoardFlash = 33.554.432 Bytes @ 0xA000.0000
"RAM": 64 MBytes OnBoardSDRAM = 67.108.864 Bytes @ 0xA400.0000

Memory Map: Used OnBoard Memories:

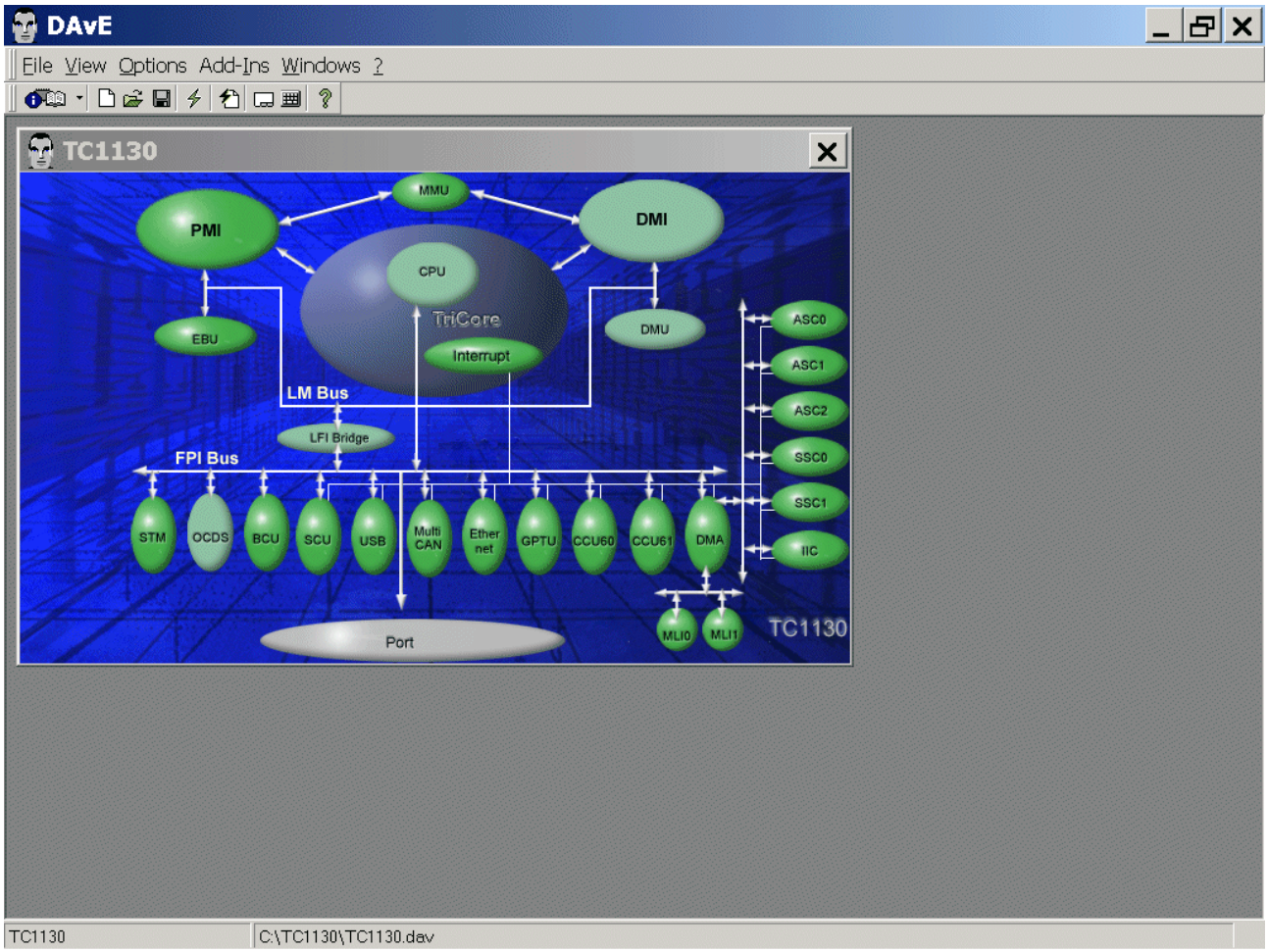


Start DAvE and open the TC1130 project:

File
Open
Look in: select C:\TC1130
File name: select TC1130.dav



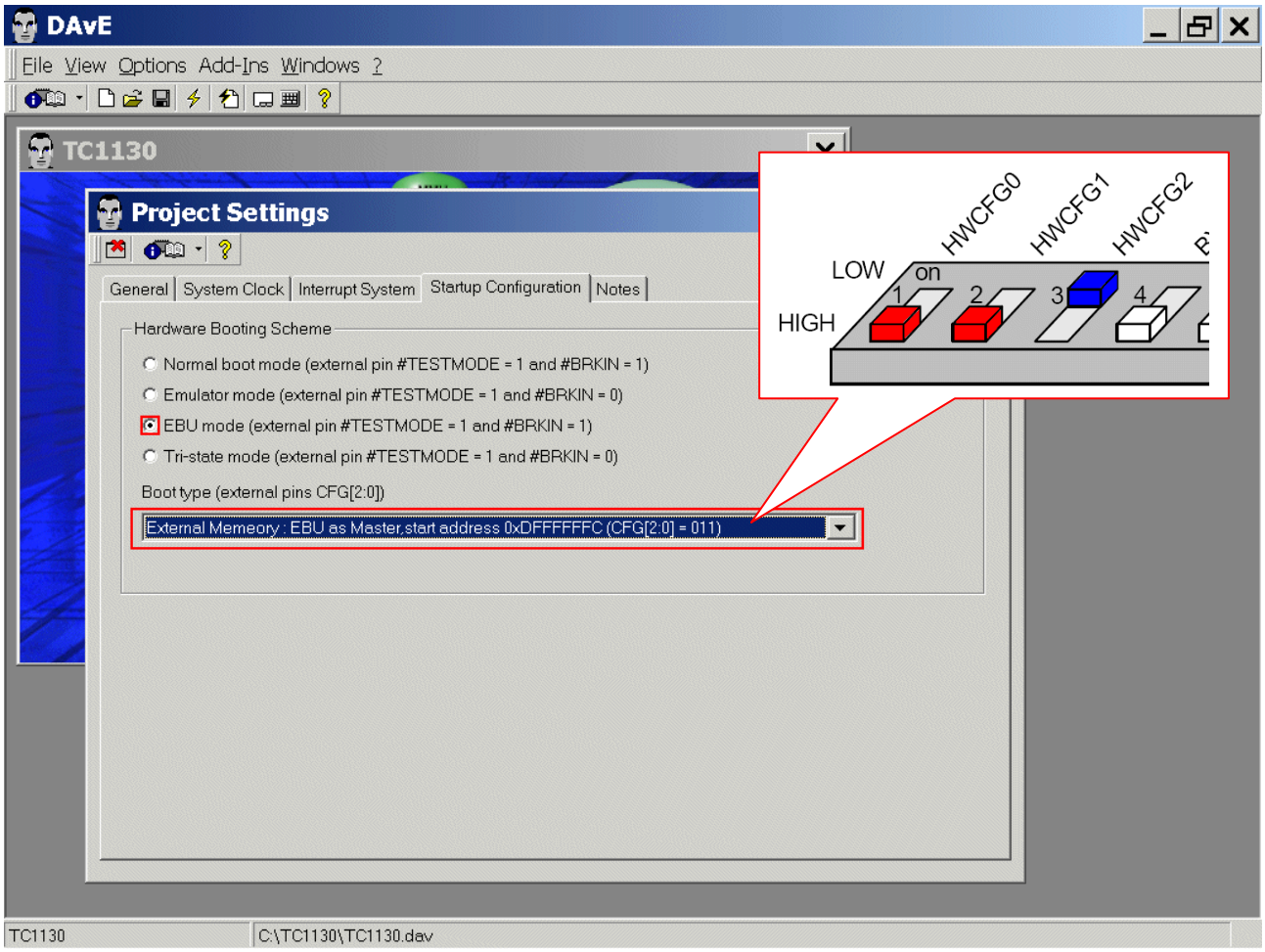
Open


DAvE
File View Options Add-Ins Windows ?
TC1130
MMU
PMI
DMI
CPU
TriCore
EBU
DMU
Interrupt
LM Bus
LFI Bridge
FPI Bus
ASC0
ASC1
ASC2
SSC0
SSC1
IIC
STM
OCDS
BCU
SCU
USB
Multi CAN
Ether net
GPTU
CCU60
CCU61
DMA
MLI0
MLI1
Port
TC1130
TC1130
C:\TC1130\TC1130.dav

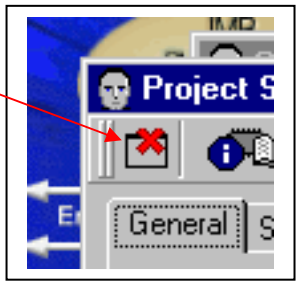
File – Project Settings

Startup Configuration: Hardware Booting Scheme: click ⊙ EBU mode

Startup Configuration: Hardware Booting Scheme: Boot type: select/check 011



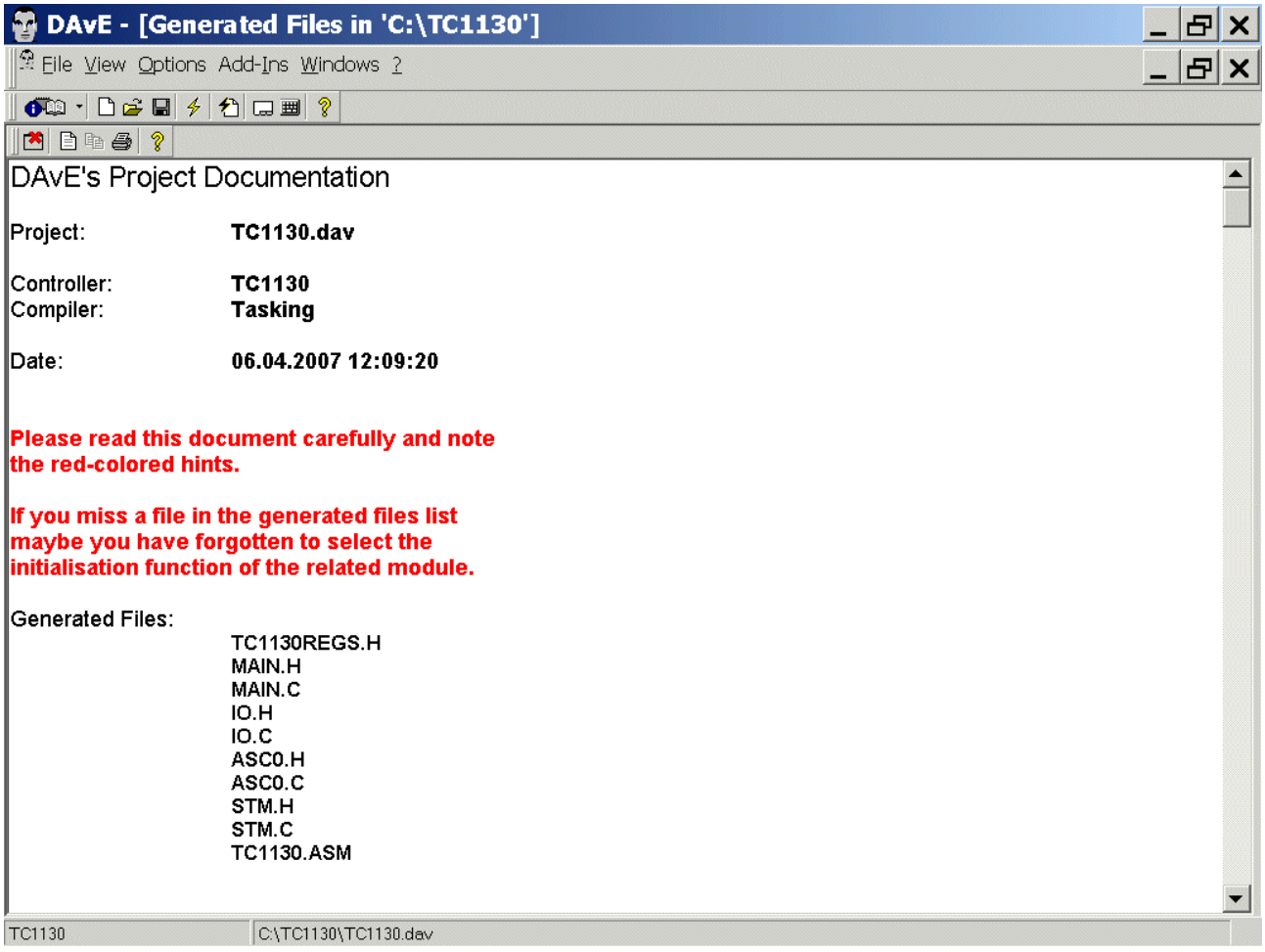
Exit this dialog now by clicking  the close button.



Generate Code:

File Generate Code	or click 

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



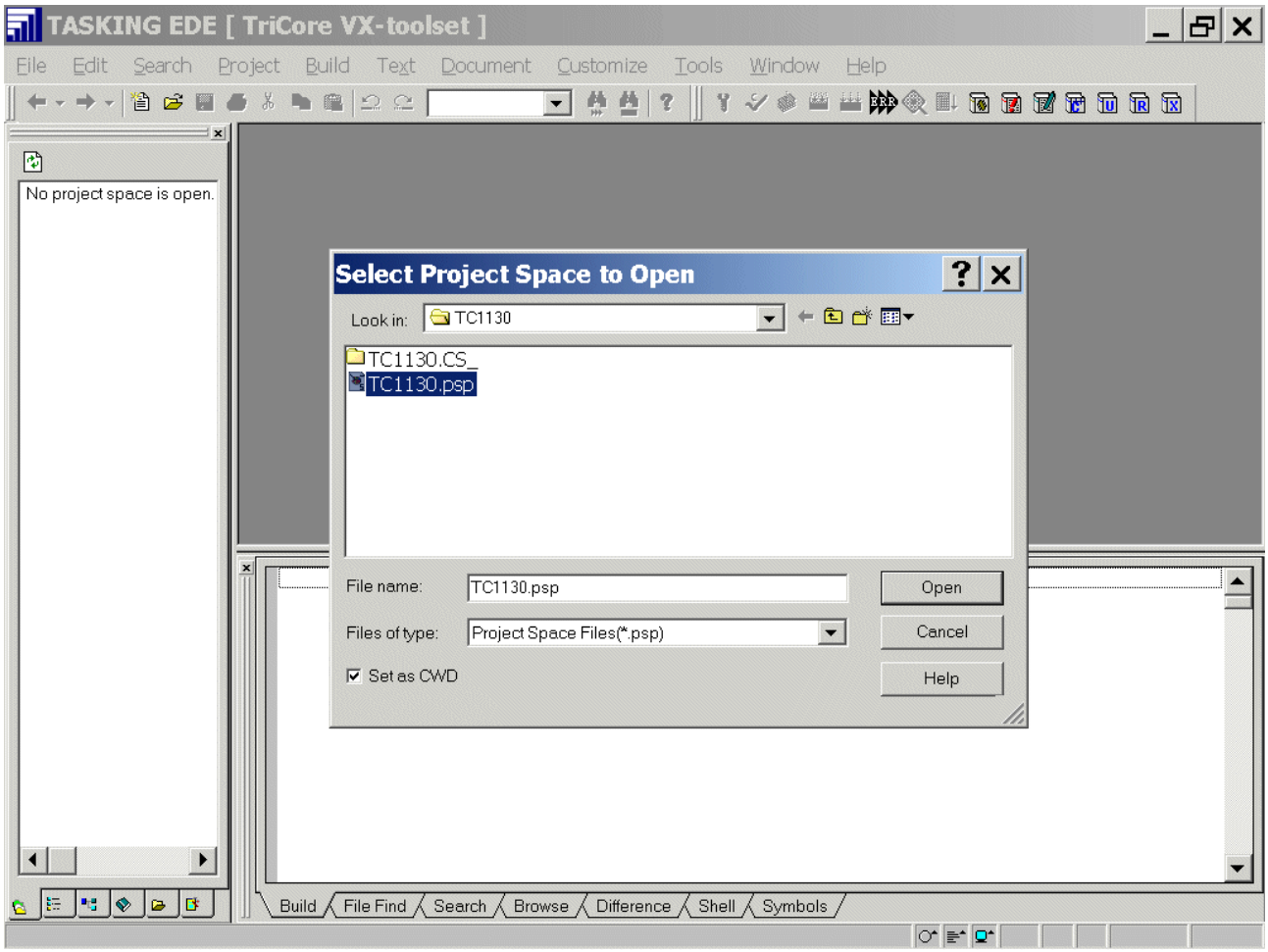
File
Exit
Save changes?

click Yes

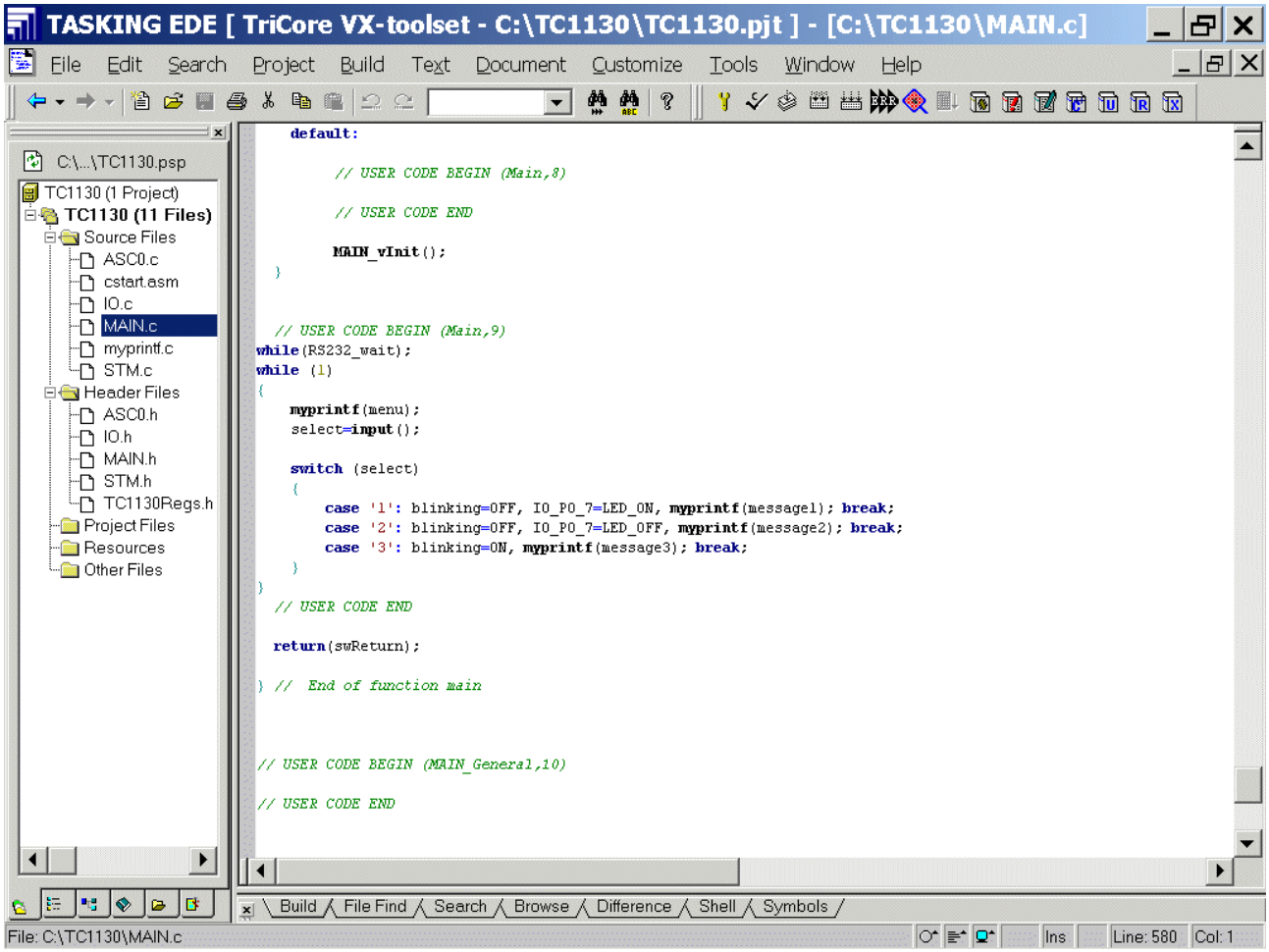
Start Tasking EDE and open the project:

File – Open Project Space

Look in: select C:\TC1130
File name: select TC1130.psp



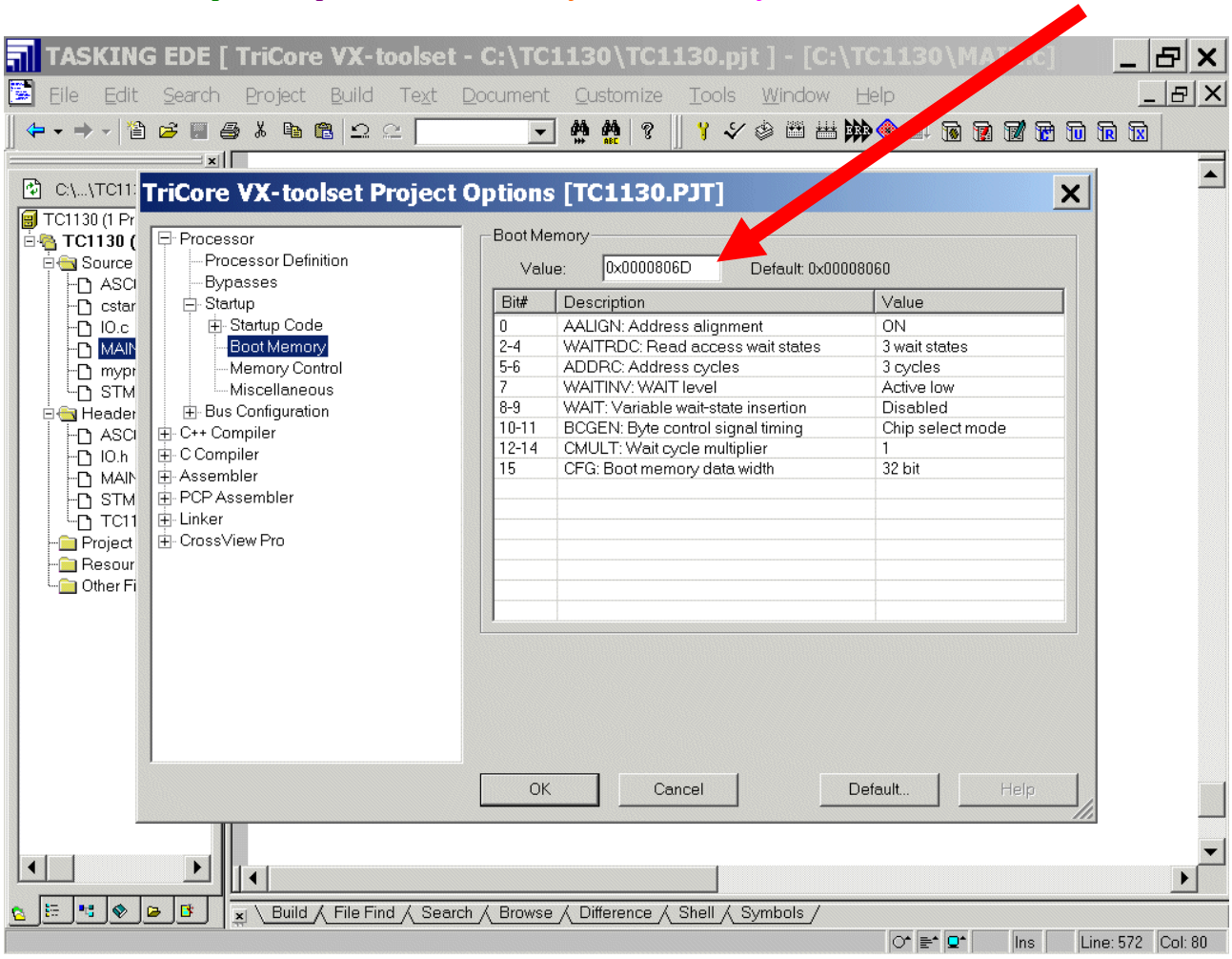
Open


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\MAIN.c]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1130.psp
TC1130 (1 Project)
TC1130 (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
TC1130Regs.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_P0_7=LED_ON, myprintf(message1); break;
case '2': blinking=OFF, IO_P0_7=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
Build File Find Search Browse Difference Shell Symbols
File: C:\TC1130\MAIN.c
Ins
Line: 580
Col: 1

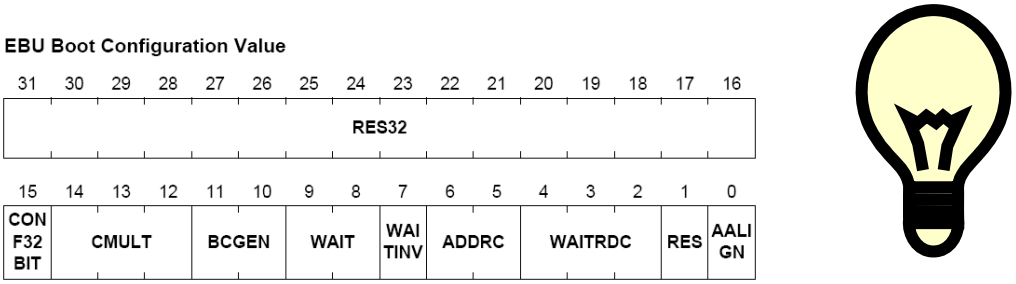
Configure Compiler, Assembler, Linker, Locater and Build – Control:

Project – Project Options

Processor: Startup: Startup Code: BootMemory: Boot Memory: Value: insert 0x0000806D



Note:

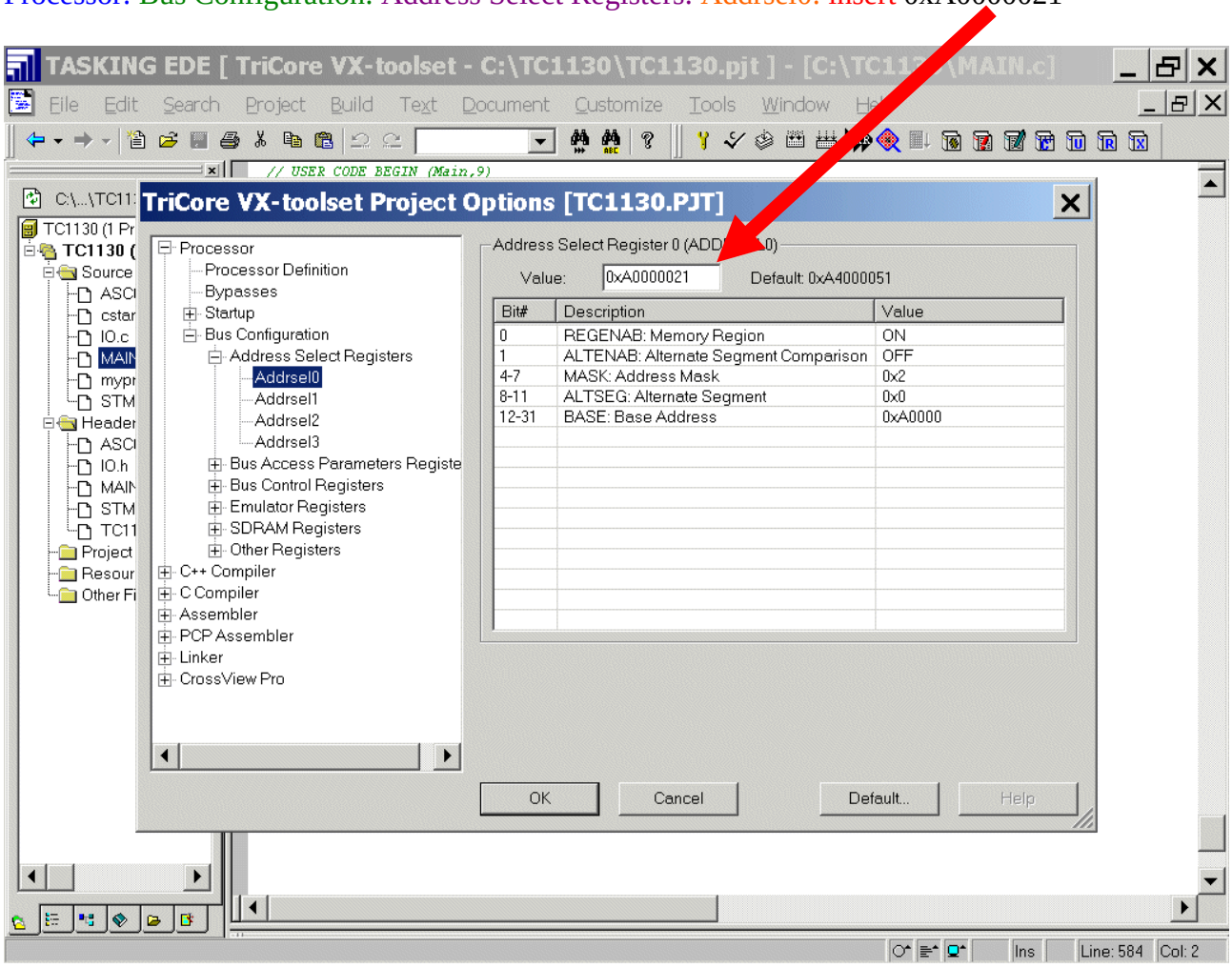


External Boot Memory Configuration Word:
If external boot is selected, the EBU will perform (exactly) one external bus read access to a specific address (0x000004 / 0x000010) of the memory device attached to CS0.
The result of this read access is used to configure the EBU.
To see “Troubleshooting” click here.

Table 14-11 EBU Address Regions, Registers, and Chip Selects

Address Region	Address Select Register	Bus Control Register	Bus Access Parameter Register	Chip Select
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

Processor: Bus Configuration: Address Select Registers: Addrsel0: insert 0xA0000021



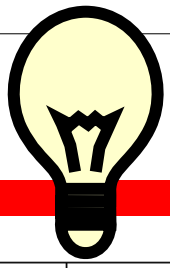


Table 14-11 EBU Address Regions, Registers, and Chip Selects

Address Region	Address Select Register	Bus Control Register	Bus Access Parameter Register	Chip Select
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

ADDRSEL[3:0]
EBU Address Select Register x

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		ALTENAB	REGENAB
rw				rw				rw				r		rw	rw

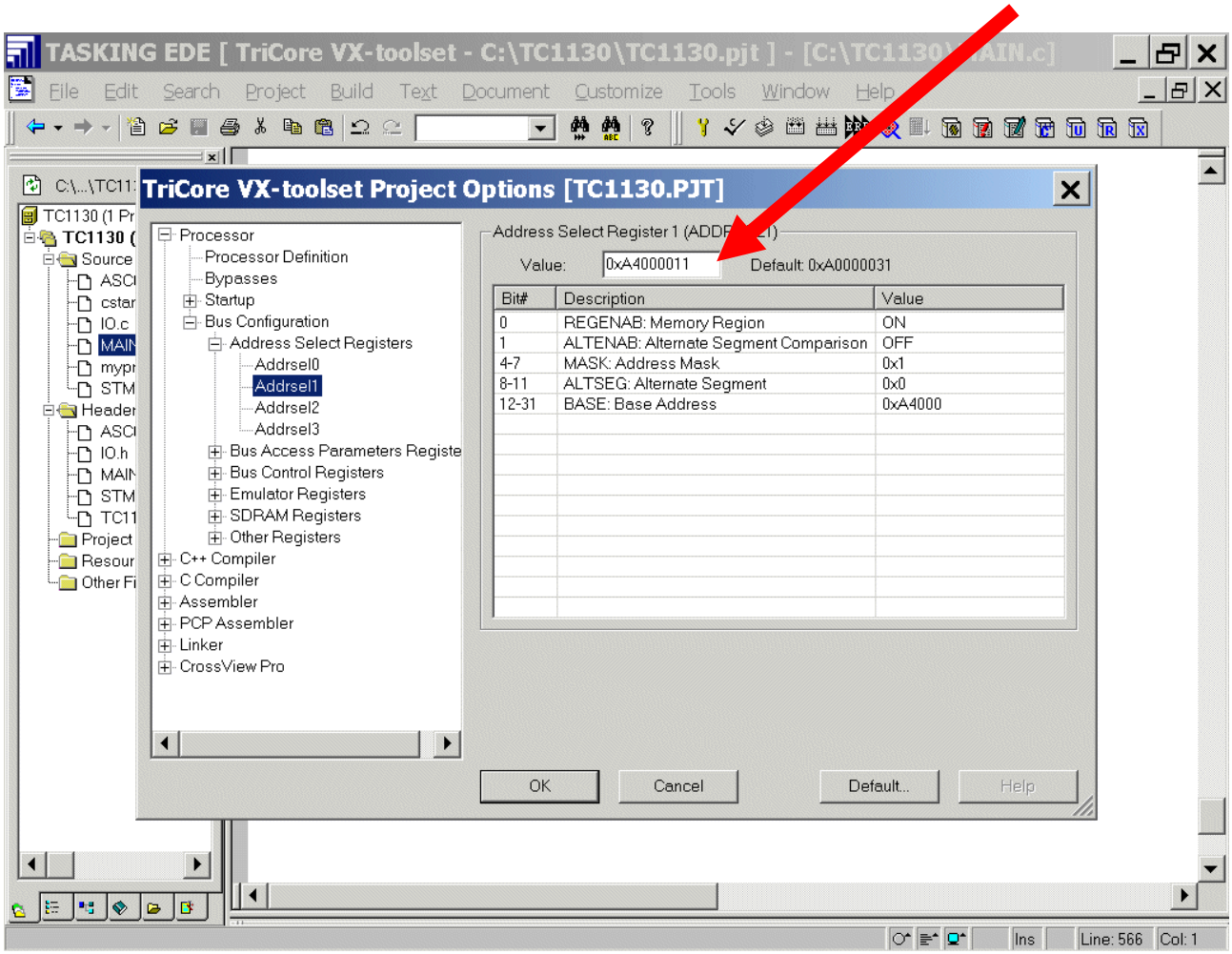
Note:
Memory Region Address Mask: Specifies the number of rightmost bits in the base address starting at bit 26, which should be included in the address comparison. Bits (31:27) will always be part of the comparison.

32 MBytes OnBoardFlash = 10000000000000000000000000 b
Base = A0000
Mask = 2

	1	2		= Mask
[31:27]	[26]	[25]	[24:0]	
		1	00000000000000000000000000	= 32 MBytes OnBoardFlash

Table 14-11 EBU Address Regions, Registers, and Chip Selects				
Address Region	**Address Select Register**	**Bus Control Register**	**Bus Access Parameter Register**	**Chip Select**
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

Processor: Bus Configuration: Address Select Registers: Addrsel1: insert 0xA4000011



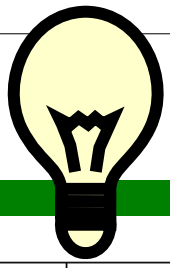


Table 14-11 EBU Address Regions, Registers, and Chip Selects

Address Region	Address Select Register	Bus Control Register	Bus Access Parameter Register	Chip Select
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

Note:
64 MBytes are an address-space between 0x0000.0000 and 0x0400.0000.
Therefore you should use a boundary of 0x0400.0000.

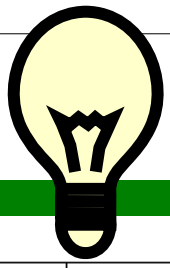


Table 14-11 EBU Address Regions, Registers, and Chip Selects

Address Region	Address Select Register	Bus Control Register	Bus Access Parameter Register	Chip Select
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

ADDRSEL[3:0]
EBU Address Select Register x **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		ALTENAB	REGENAB
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.

64 MBytes OnBoardSDRAM = 0x0400.0000 = 1000000000000000000000000 b
Base = A4000
Mask = 1

	1		= Mask
[31:27]	[26]	[25:0]	
	1	00000000000000000000000000	= 64 MBytes OnBoardSDRAM

Table 14-11 EBU Address Regions, Registers, and Chip Selects

Address Region	Address Select Register	Bus Control Register	Bus Access Parameter Register	Chip Select
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

Note: “User region 2” = NOT USED !!!

Processor: Bus Configuration: Address Select Registers: Addrsel2: insert 0xA2000030

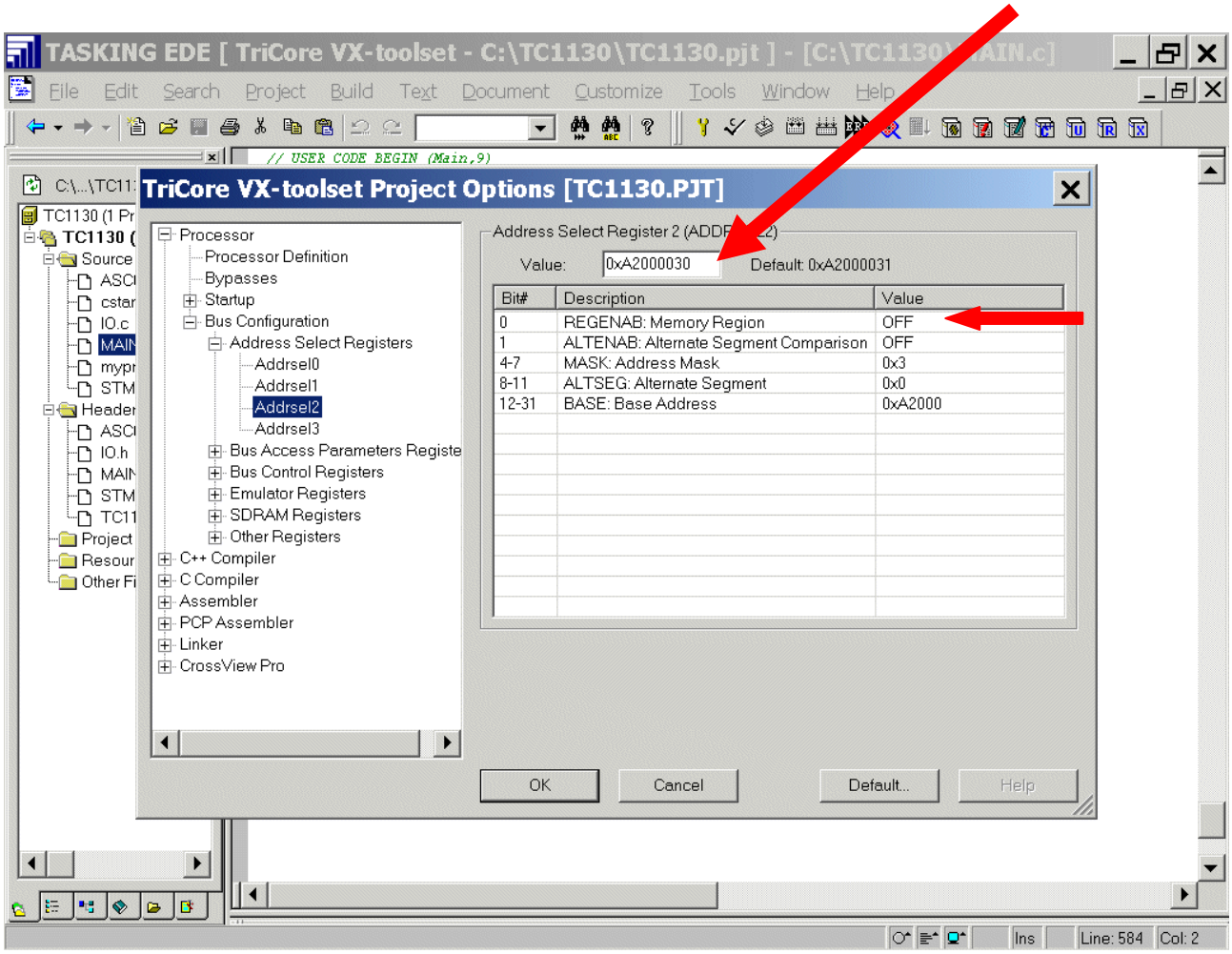


Table 14-11 EBU Address Regions, Registers, and Chip Selects

Address Region	Address Select Register	Bus Control Register	Bus Access Parameter Register	Chip Select
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

Processor: Bus Configuration: Bus Access Parameters Registers: Busap0: insert 0xC6DB0000

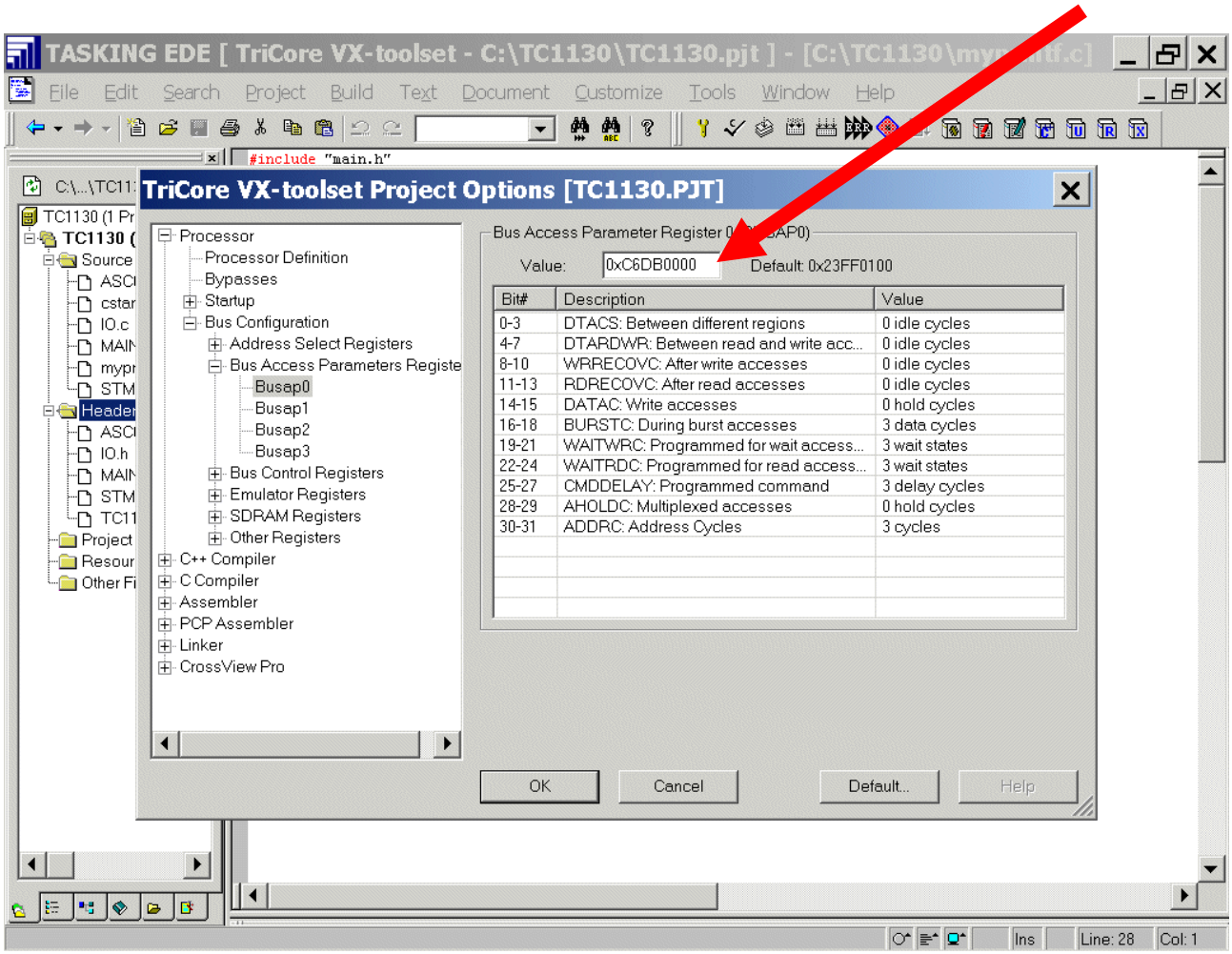


Table 14-11 EBU Address Regions, Registers, and Chip Selects				
Address Region	**Address Select Register**	**Bus Control Register**	**Bus Access Parameter Register**	**Chip Select**
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

Processor: Bus Configuration: Bus Access Parameters Registers: Busap1: insert 0x42000000

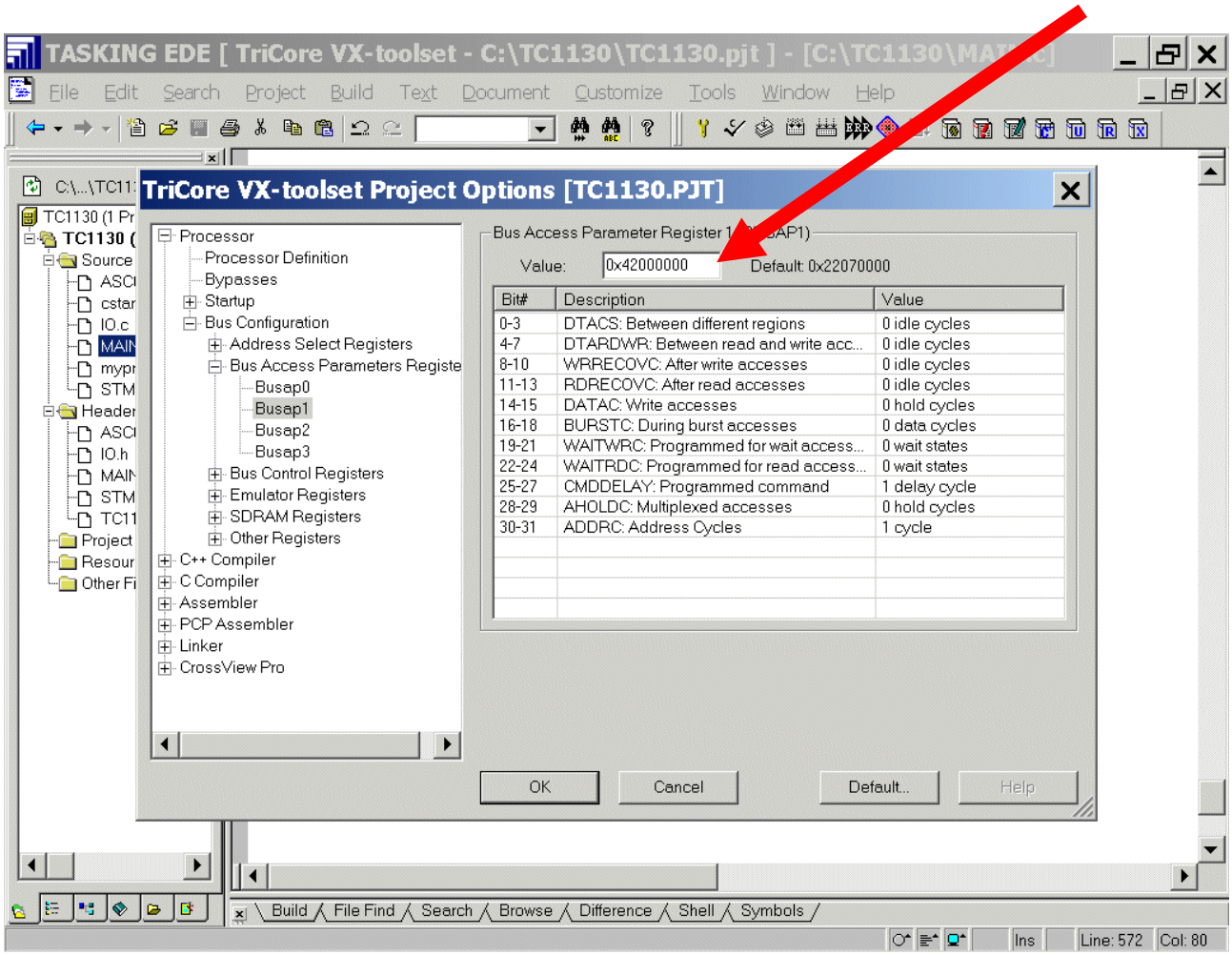


Table 14-11 EBU Address Regions, Registers, and Chip Selects

Address Region	Address Select Register	Bus Control Register	Bus Access Parameter Register	Chip Select
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

Processor: Bus Configuration: Bus Control Registers: Buscon0: insert 0x00922300

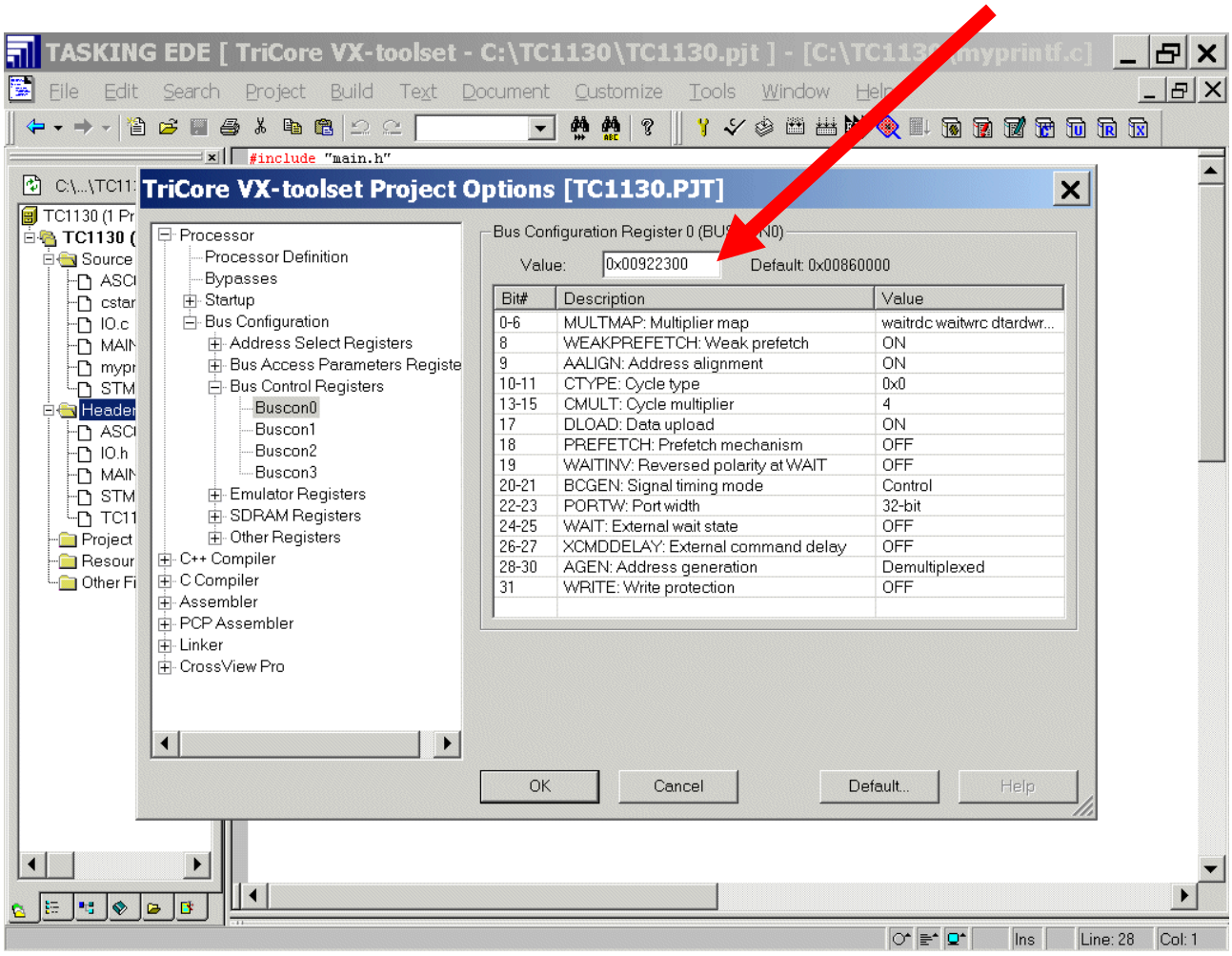
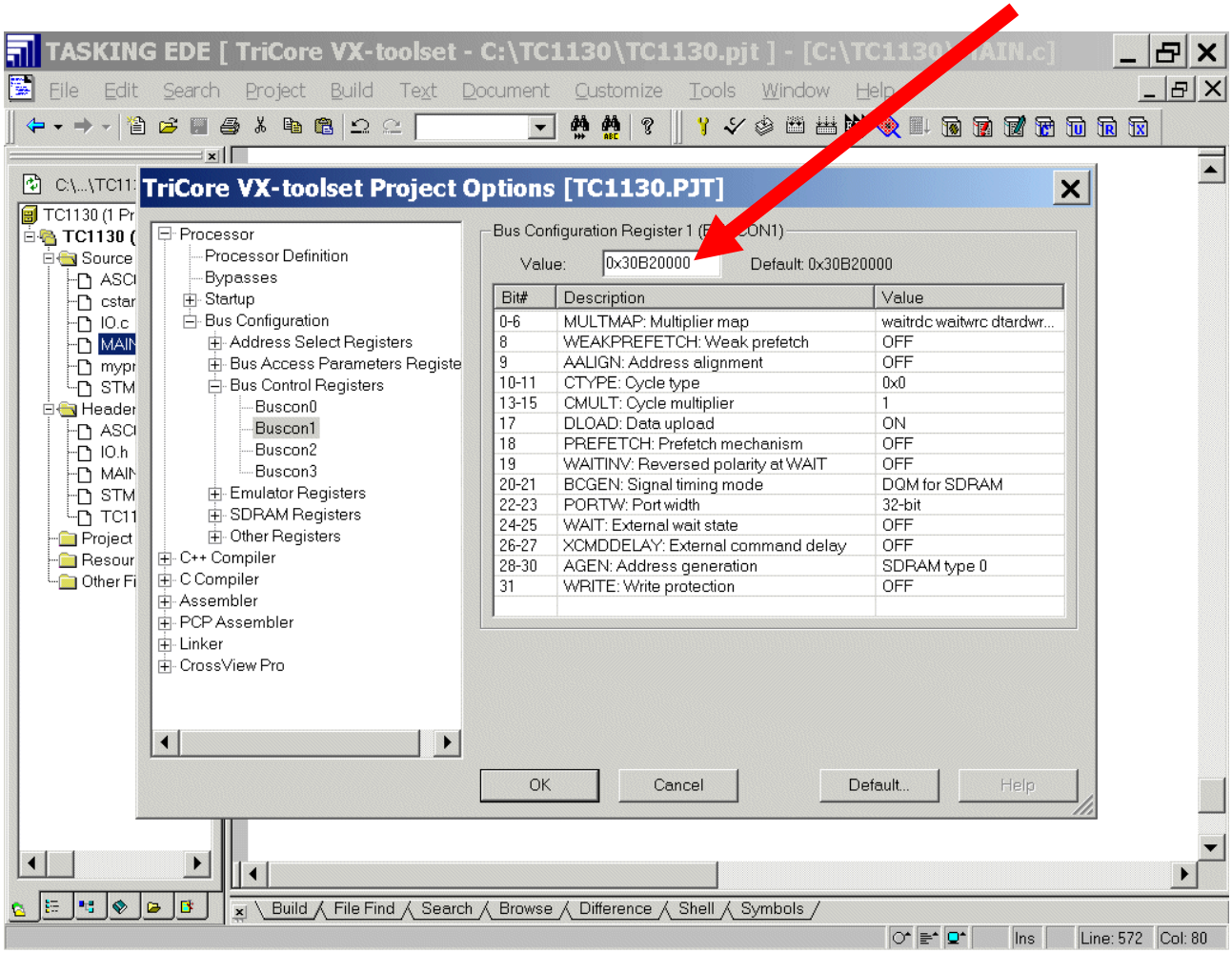
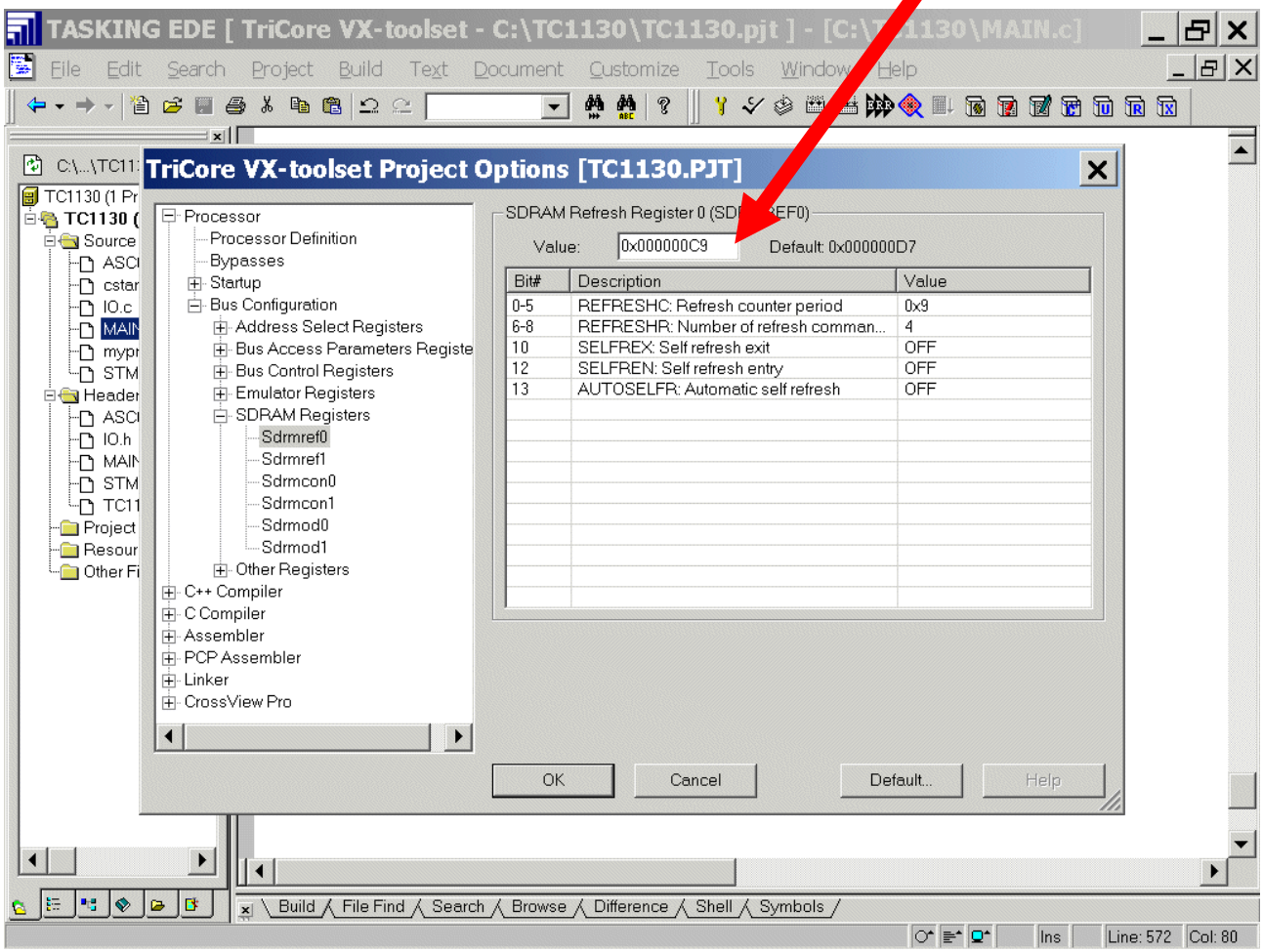


Table 14-11 EBU Address Regions, Registers, and Chip Selects				
Address Region	**Address Select Register**	**Bus Control Register**	**Bus Access Parameter Register**	**Chip Select**
User region 0	ADDRSEL0	BUSCON0	BUSAP0	$\overline{CS0}$
User region 1	ADDRSEL1	BUSCON1	BUSAP1	$\overline{CS1}$
User region 2	ADDRSEL2	BUSCON2	BUSAP2	$\overline{CS2}$
User region 3	ADDRSEL3	BUSCON3	BUSAP3	$\overline{CS3}$
Emulator region	EMUAS	EMUBC	EMUBAP	$\overline{CSEMU}$

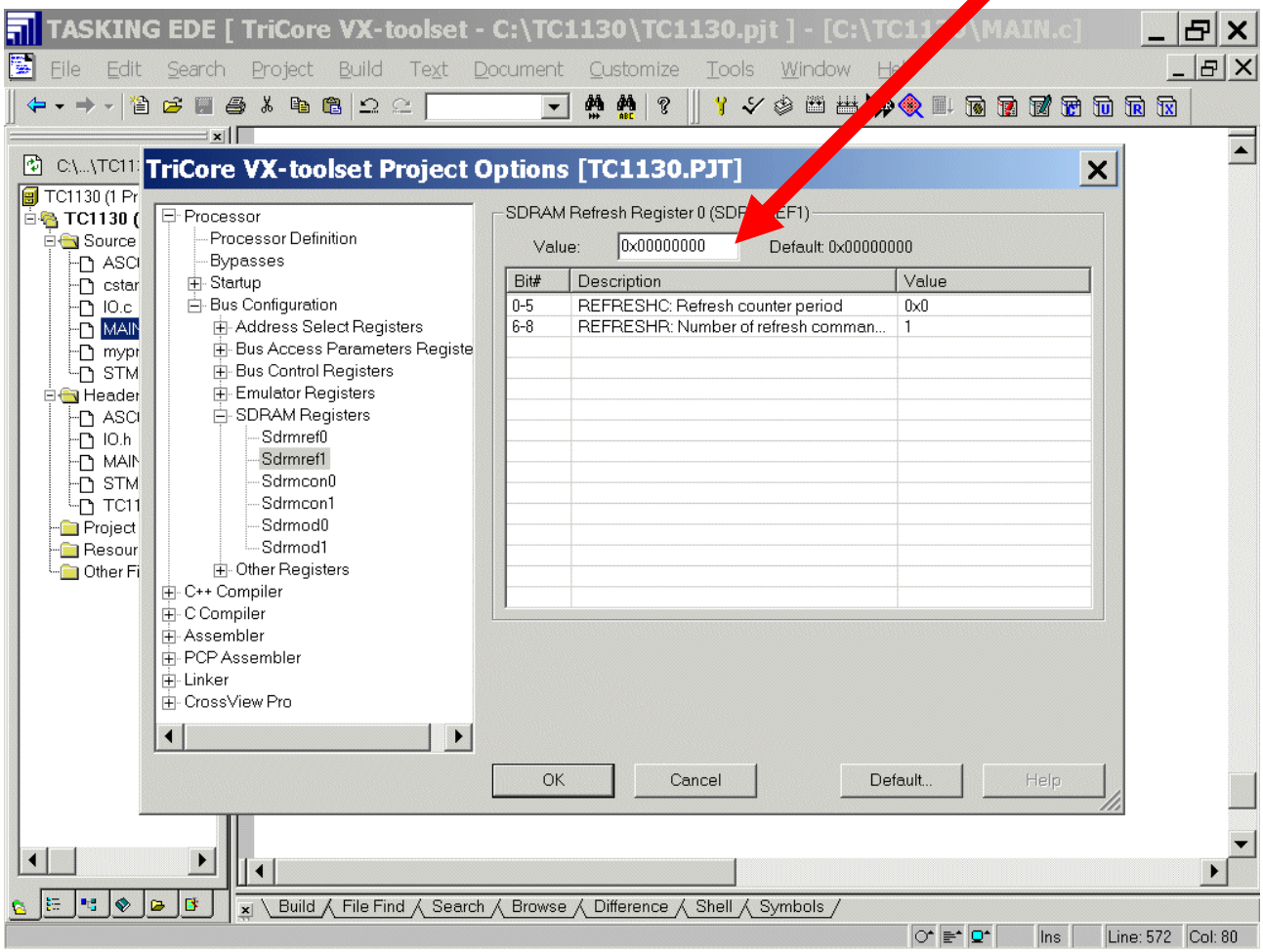
Processor: Bus Configuration: Bus Control Registers: Buscon1: check/insert 0x30B20000



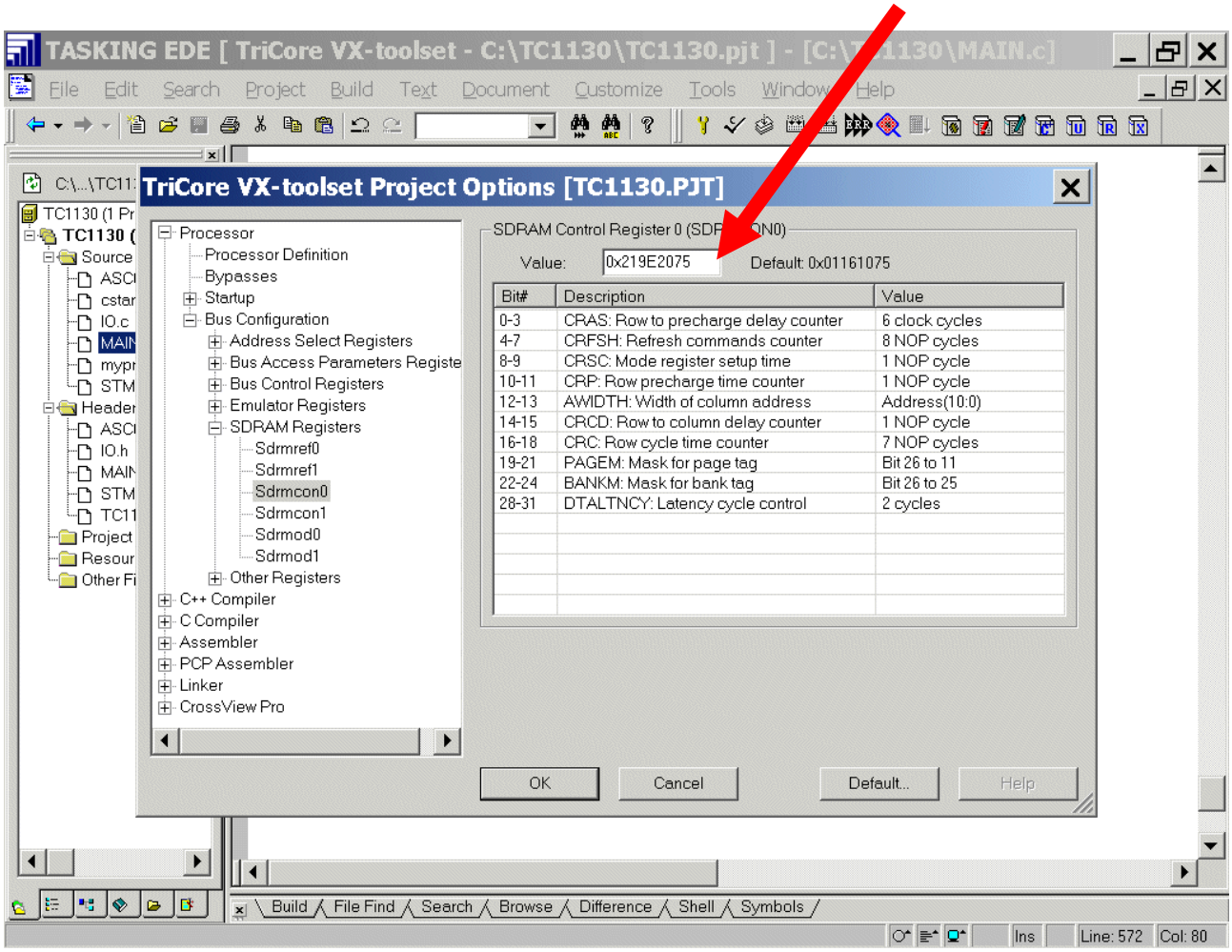
Processor: Bus Configuration: SDRAM Registers: Sdrmref0: insert 0x000000C9



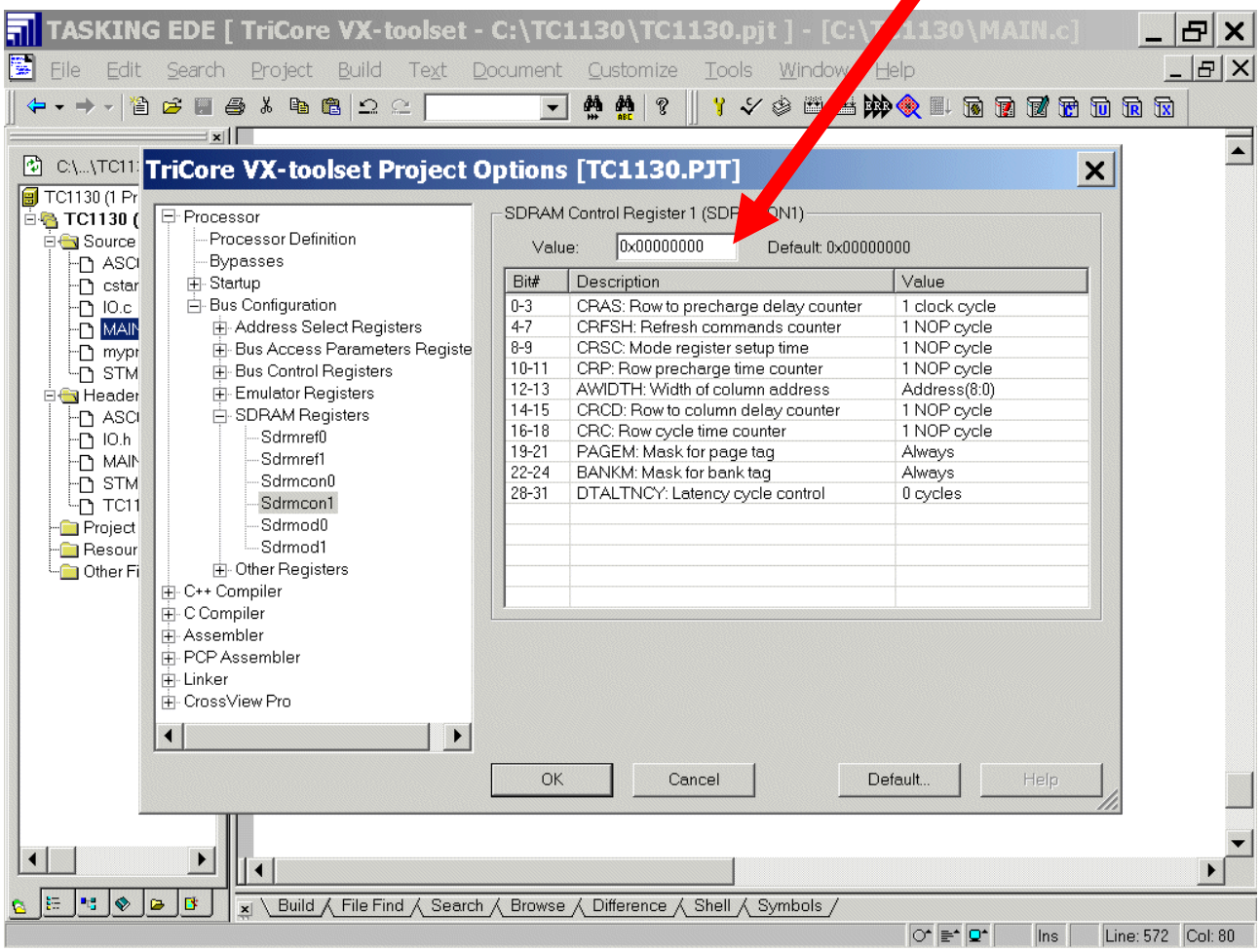
Processor: Bus Configuration: SDRAM Registers: Sdrmref1: insert/check 0x00000000



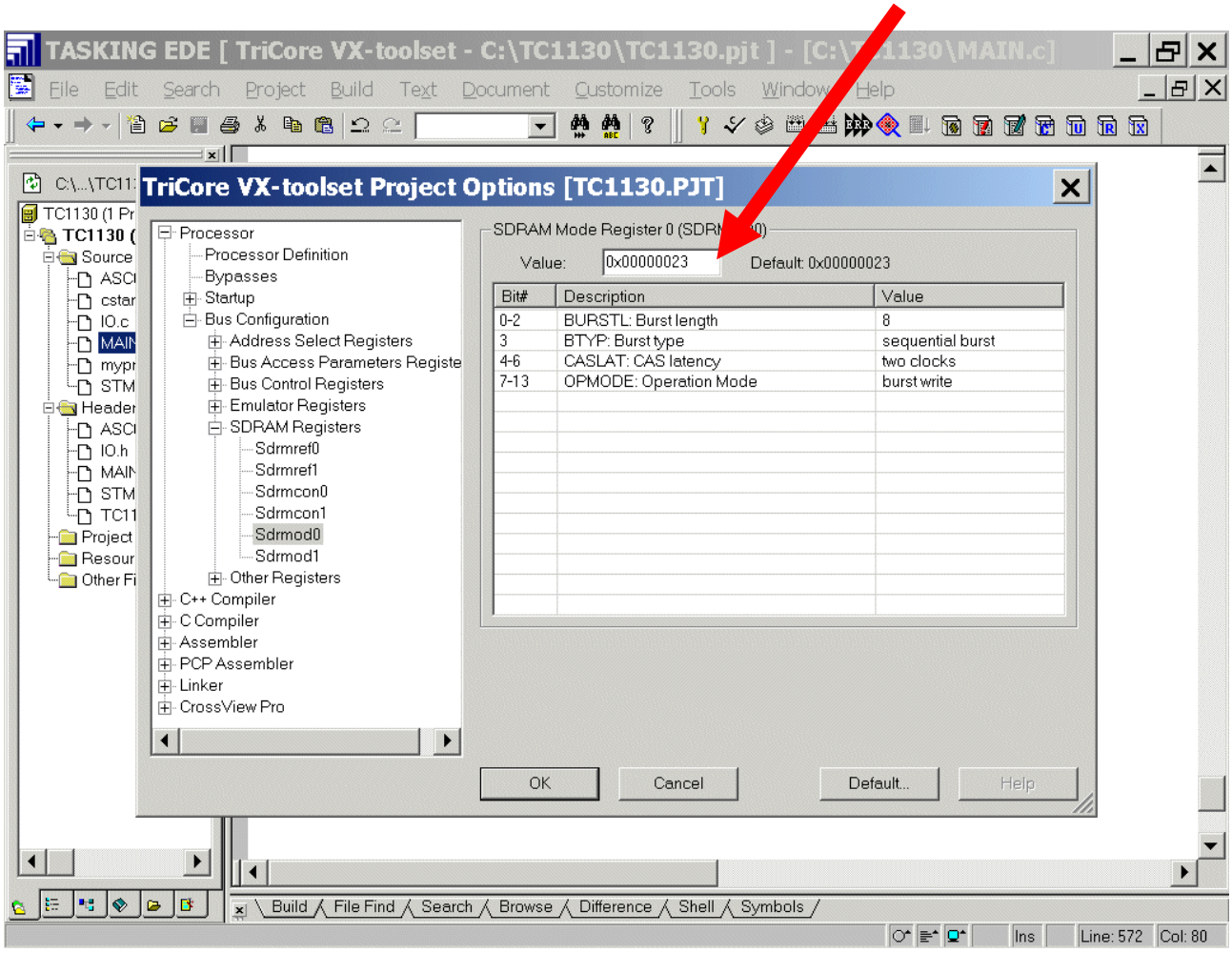
Processor: Bus Configuration: SDRAM Registers: Sdrmcon0: insert 0x219E2075



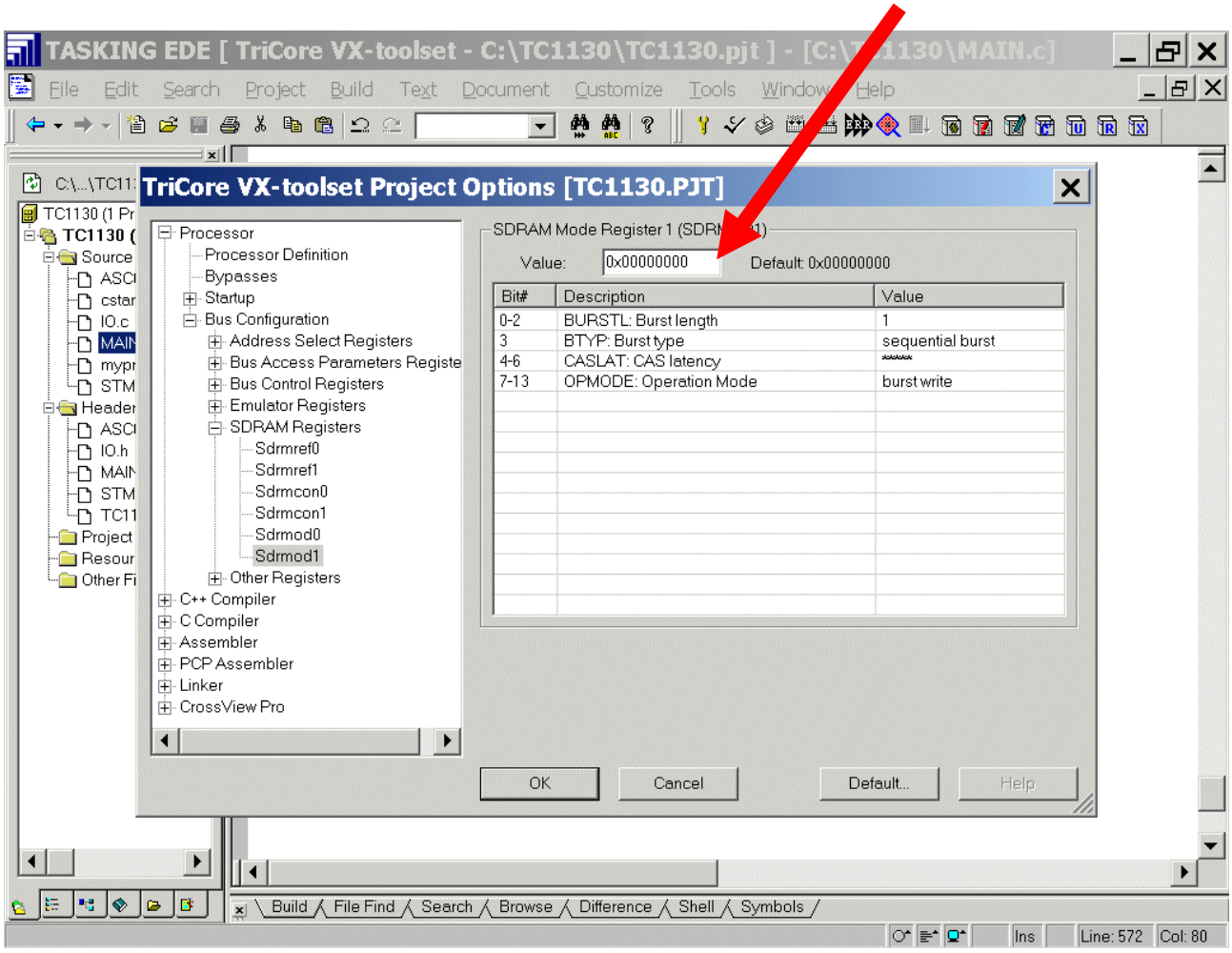
Processor: Bus Configuration: SDRAM Registers: Sdrmcon1: insert/check 0x00000000



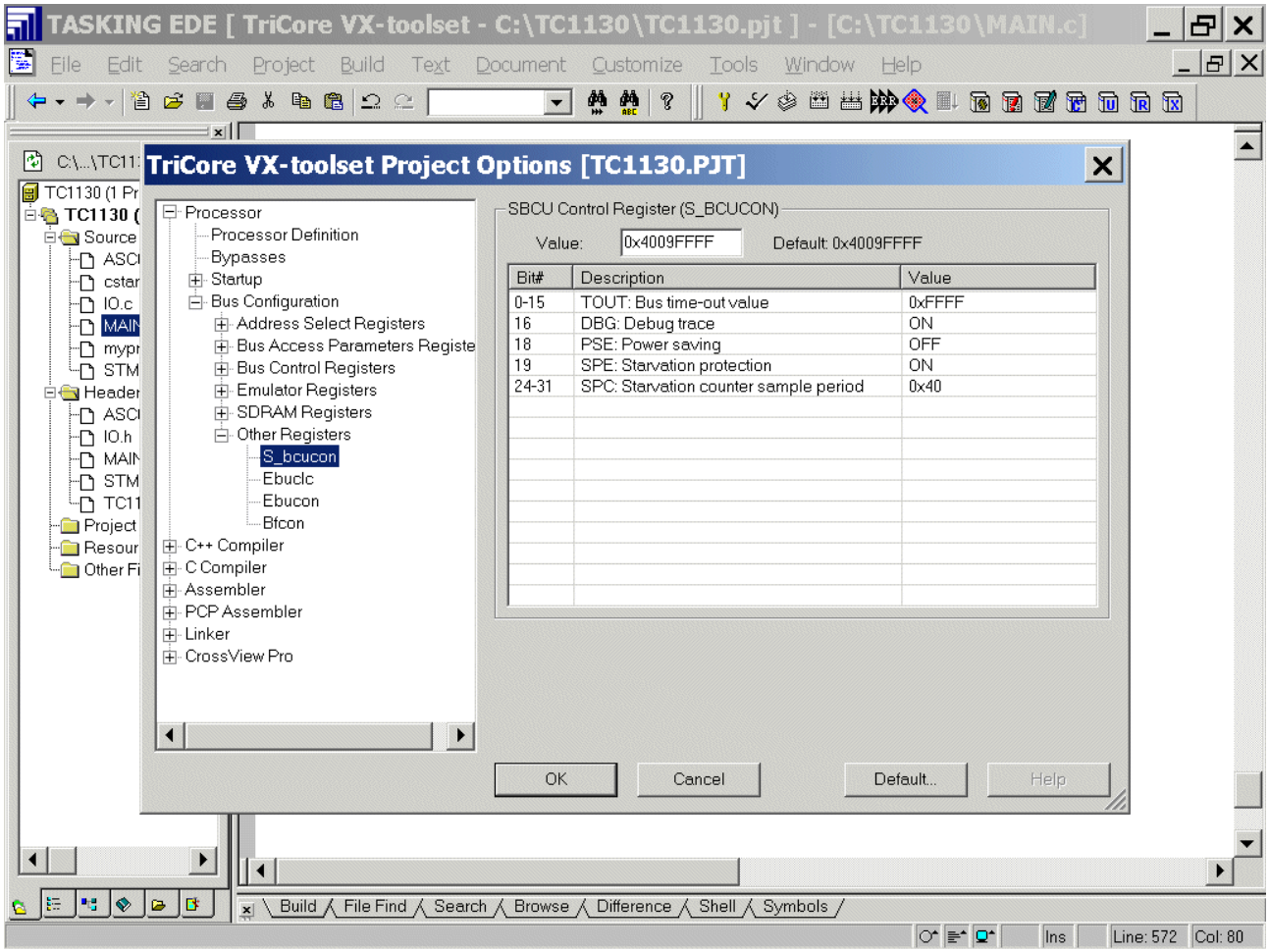
Processor: Bus Configuration: SDRAM Registers: Sdrmod0: insert/check 0x00000023



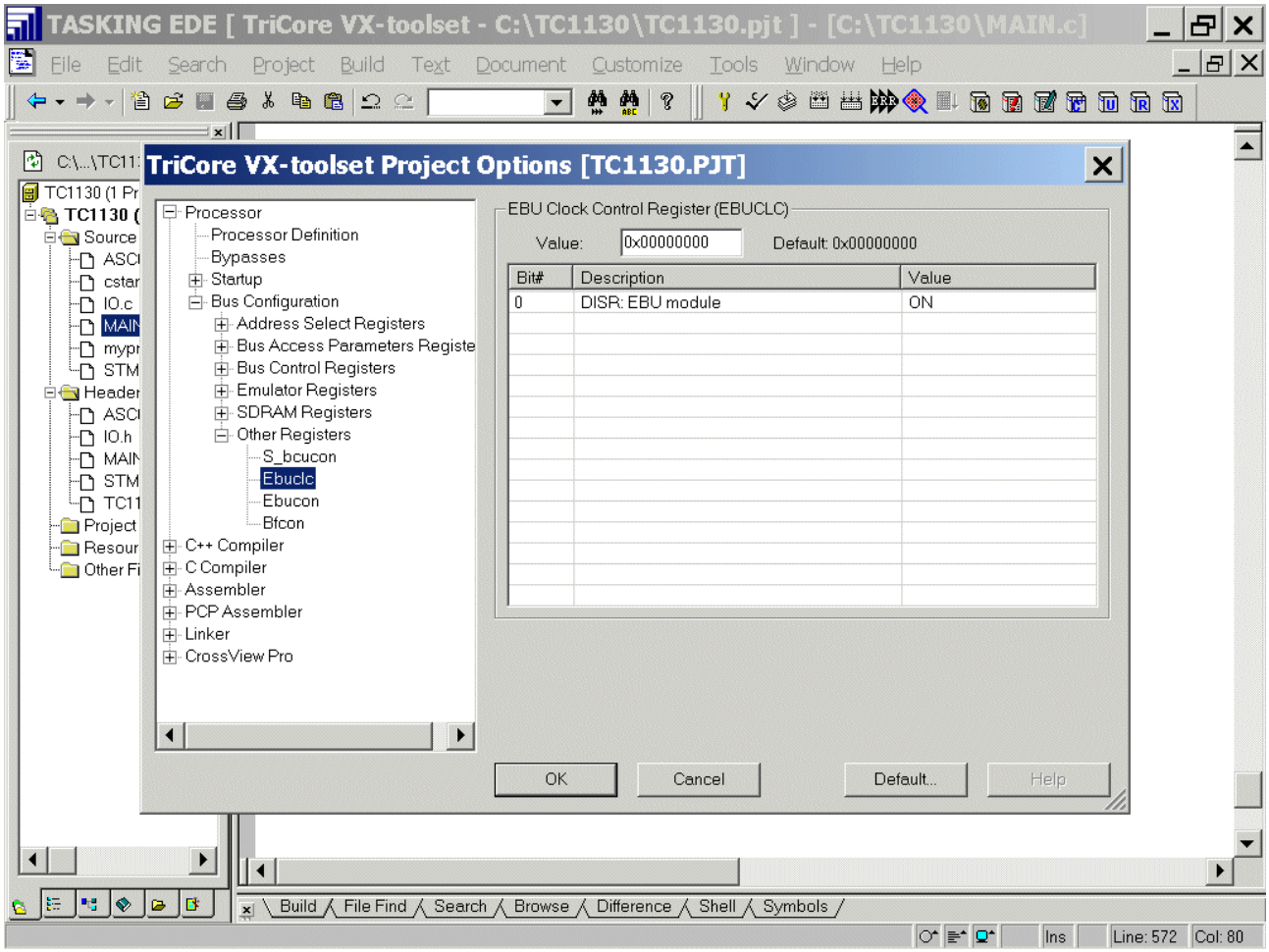
Processor: Bus Configuration: SDRAM Registers: Sdrmod1: insert/check 0x00000000



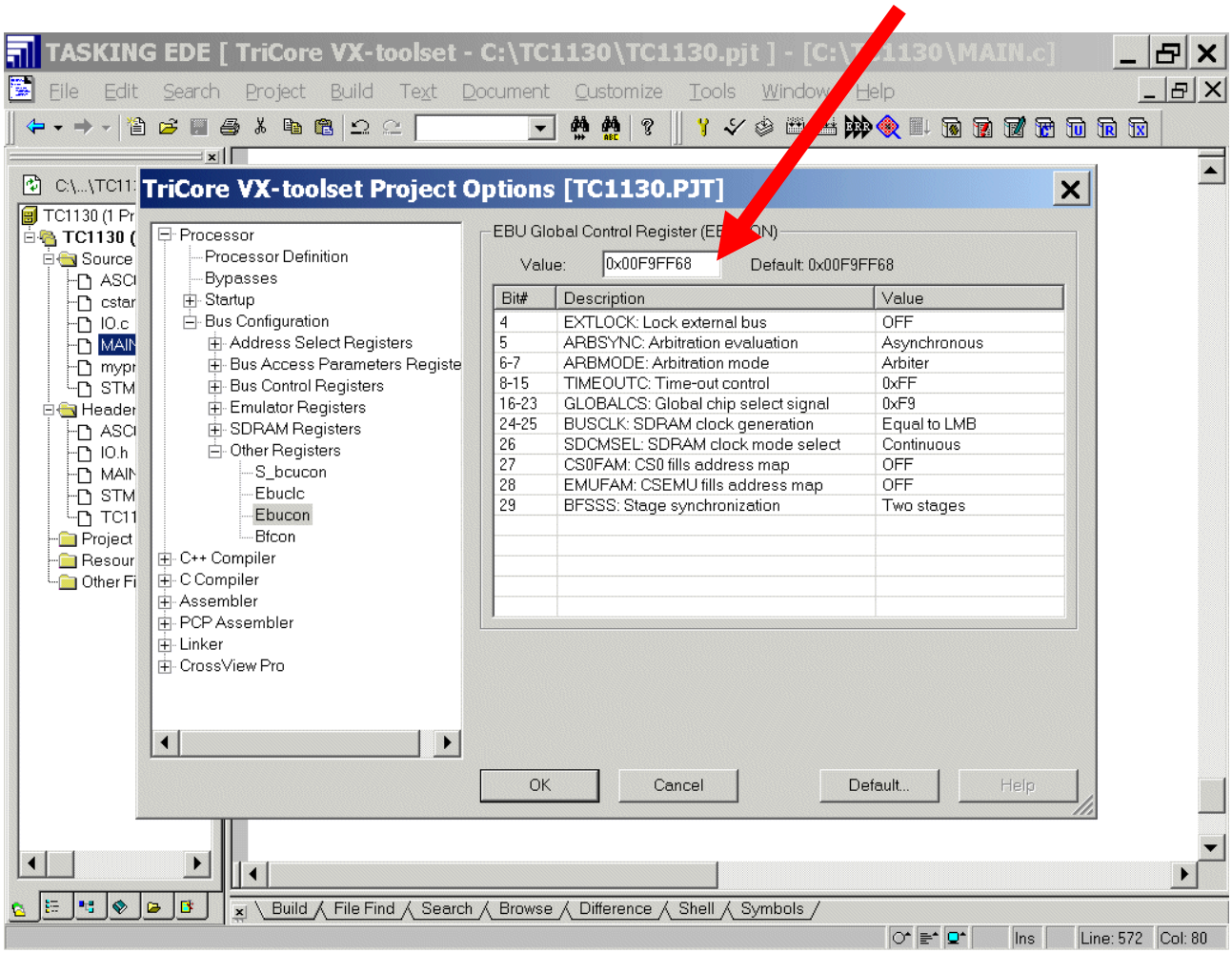
Processor: Bus Configuration: Other Registers: S_bcucon: (do nothing)



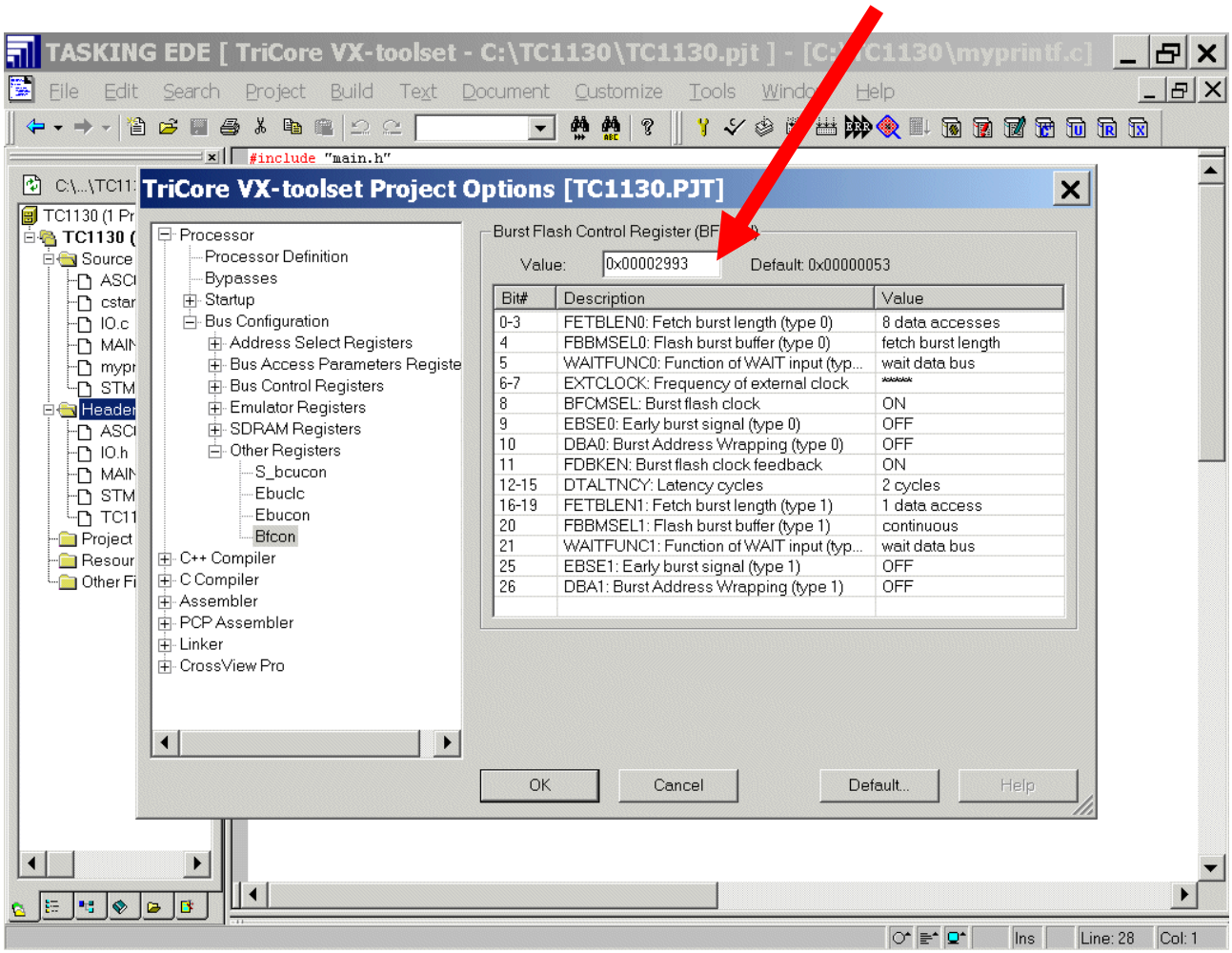
Processor: Bus Configuration: Other Registers: Ebuclc: (do nothing)

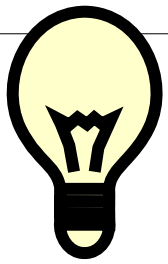


Processor: Bus Configuration: Other Registers: Ebucon: insert/check 0x00F9FF68



Processor: Bus Configuration: Other Registers: Bfcon: insert 0x00002993





Additional Information (remember):

Adobe Reader - [tc1130_um_v1.3_2004_11_sys.pdf]

File Edit View Document Tools Window Help

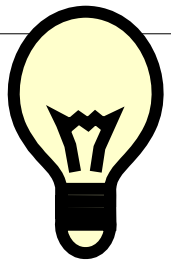
Table 7-1 TC1130 Block Address Map

Segment	Address Range	Size	Description	DMI Acc.	PMI Acc.	
0-7	0000 0000$_H$ - 7FFF FFFF$_H$	2 GB	MMU Space	via FPI	via FPI	cached
8	8000 0000$_H$ - 8FFF FFFF$_H$	256 MB	External Memory Space mapped from Segment 10	via LMB	via LMB	
9	9000 0000$_H$ - 9FDF FFFF$_H$	256 MB	Reserved	via FPI	via FPI	
	A000 0000$_H$ - AFBF FFFF$_H$	252 MB	External Memory Space	via LMB	via LMB	non-cached
	AFC0 0000$_H$ - AFC0 FFFF$_H$	64 KB	DMU Space			
	AFC1 0000$_H$ - AFFF FFFF$_H$	≈ 4 MB	Reserved			
11	B000 0000$_H$ - BFFF FFFF$_H$	256 MB	Reserved	via FPI	via FPI	

8,15 x 11,58 in

242 of 979

252 MBytes External Memory Space used for:
"ROM": 32 MBytes OnBoardFlash = 33.554.432 Bytes @ 0xA000.0000
"RAM": 64 MBytes OnBoardSDRAM= 67.108.864 Bytes @ 0xA400.0000



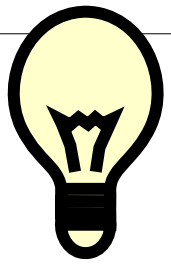
Additional Information:

To avoid a Linker/Locater-Problem (Function Calling Mode):

```
From: Altium Support [mailto:support.ap@altium.com]
Sent: Wednesday, January 31, 2007 11:34 AM
To: (IFAT S FAE)
Subject: Your case 00023116 has been updated

Hi,

Your case # 00023116: 23116 - relocation patch error [tricore] has been updated.

Please click on the link below to view this case in the SUPPORTcenter.
http://www.altium.com/supportcenter

The following comment was added to your case:

I order to explain your case let me start by repeating your error message:

   ltc E121: relocation patch error in "task1": relocation
   value 0xa11004b0 for relocation of type rel24 or abs24
   at offset 92 in section ".text.main.main" at address
   0xa0000188 is not a valid address in R_TRICORE_24REL.
   Hint: check the mapfile for a section that occupies this
   address.

I admit this is a bit of a mouthful, and it means:

   patch error at offset 92 of section .text.main.main

What I always do then is to enable list files and tick the "section directives"
CHECKBOX after you've done that. Since the offending section is called
.text.main.main  listfile main.lst  must be studied. Lookup section
.text.main.main and then offset 92 (0x5C). You'll end up at:

   005C 6Drrrrrr  2   30   276         call    sprintf

Since the linker says "patch error" it means that the call to sprintf is too big
to fit expression 'rrrrrr'. To make this work either the call must be changed or
the libraries must be located closer to the application code. Try locating the
libraries at 0xA1100000 from the same PAGE where you located the reset vector.
It will do magic.

Regards,
Altium Customer Support
```

you have to locate the target address of your code (e.g. RESET start address and Libraries start address) within the range **+/- 16 MByte** of any 256 MByte Segment [The Linker/Locater generates an error (see above) when the target address appears to be out of reach].

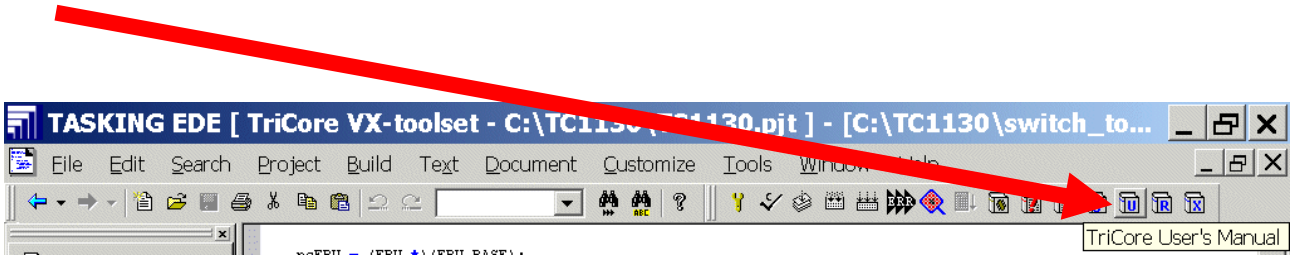
Note (Function Calling Mode Indirect**):**
The __indirect keyword can be used to force the less efficient indirect call [the target address may be any 32 bit address within the whole memory space (4 GByte)].

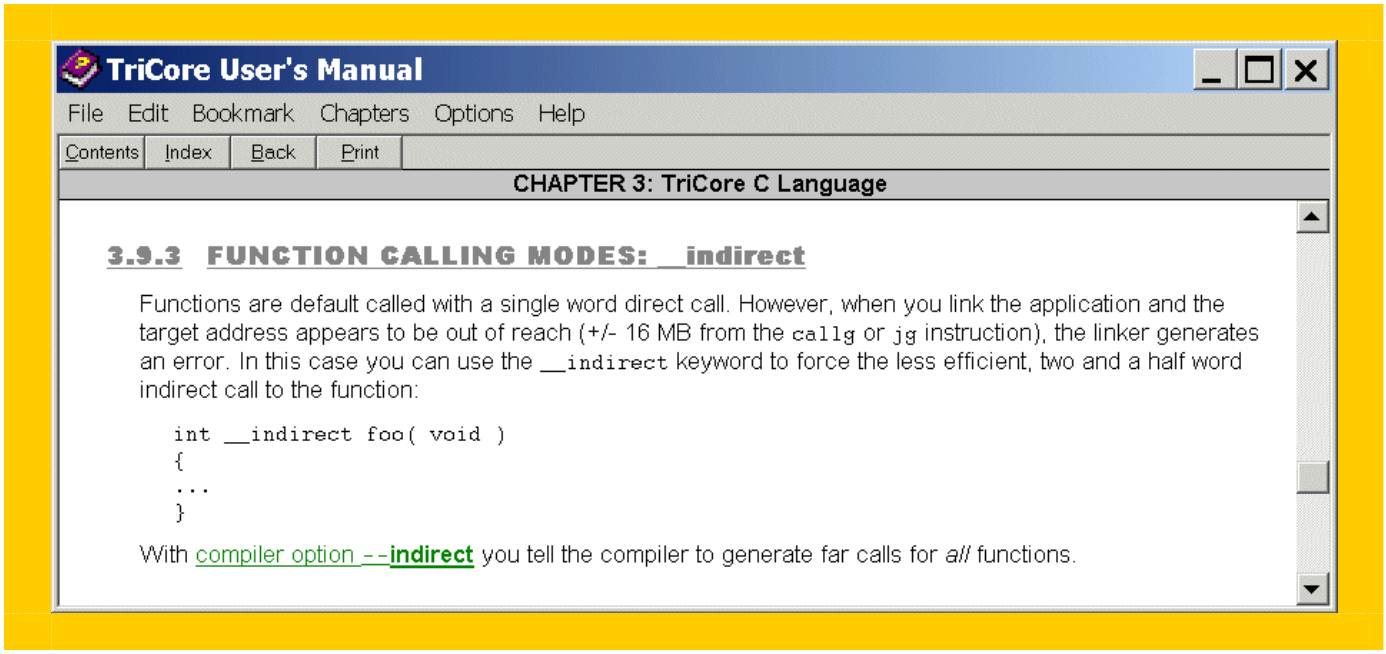


Additional Information:

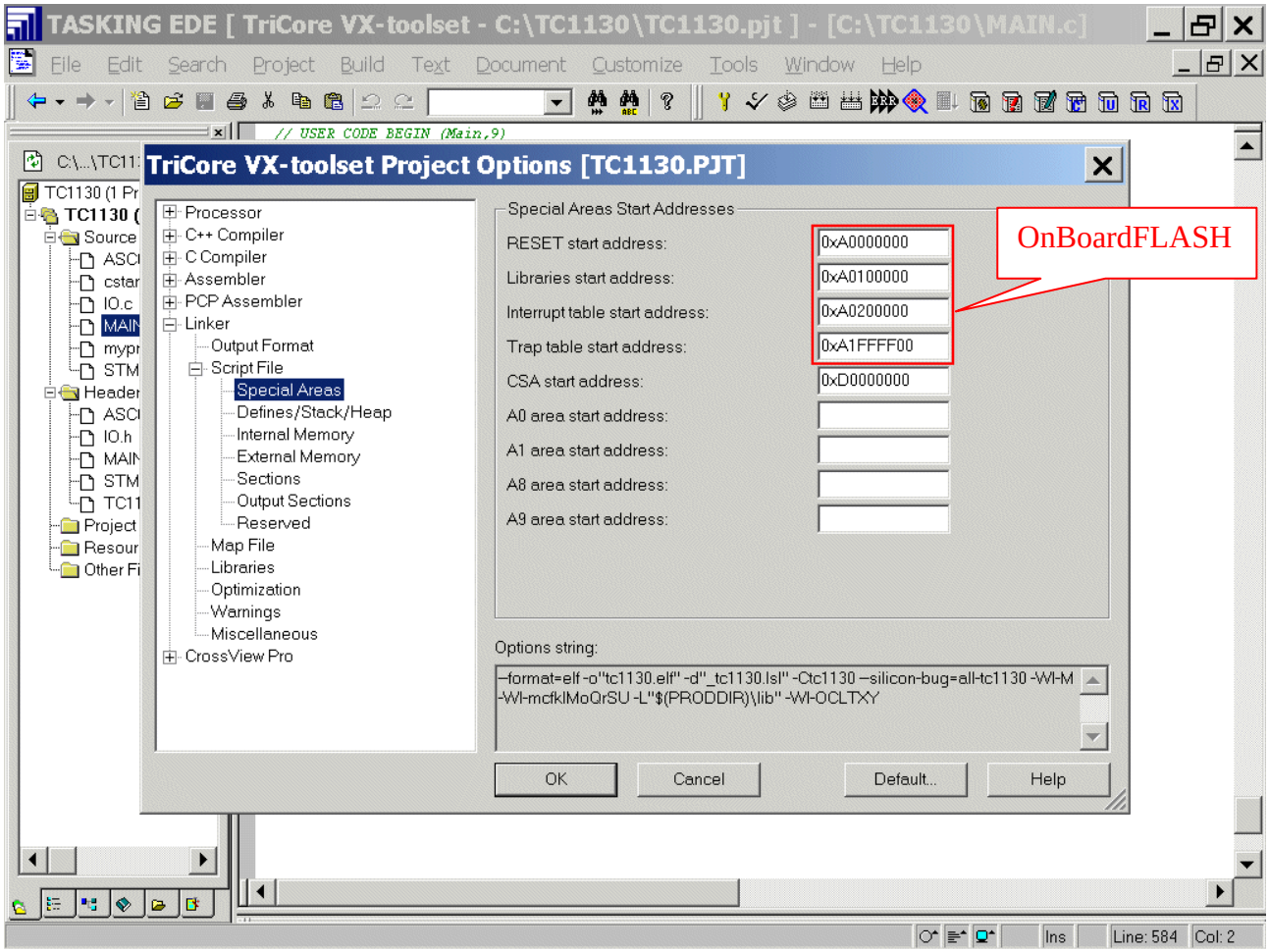
See also the Online-Manual:

Click TriCore User's Manual



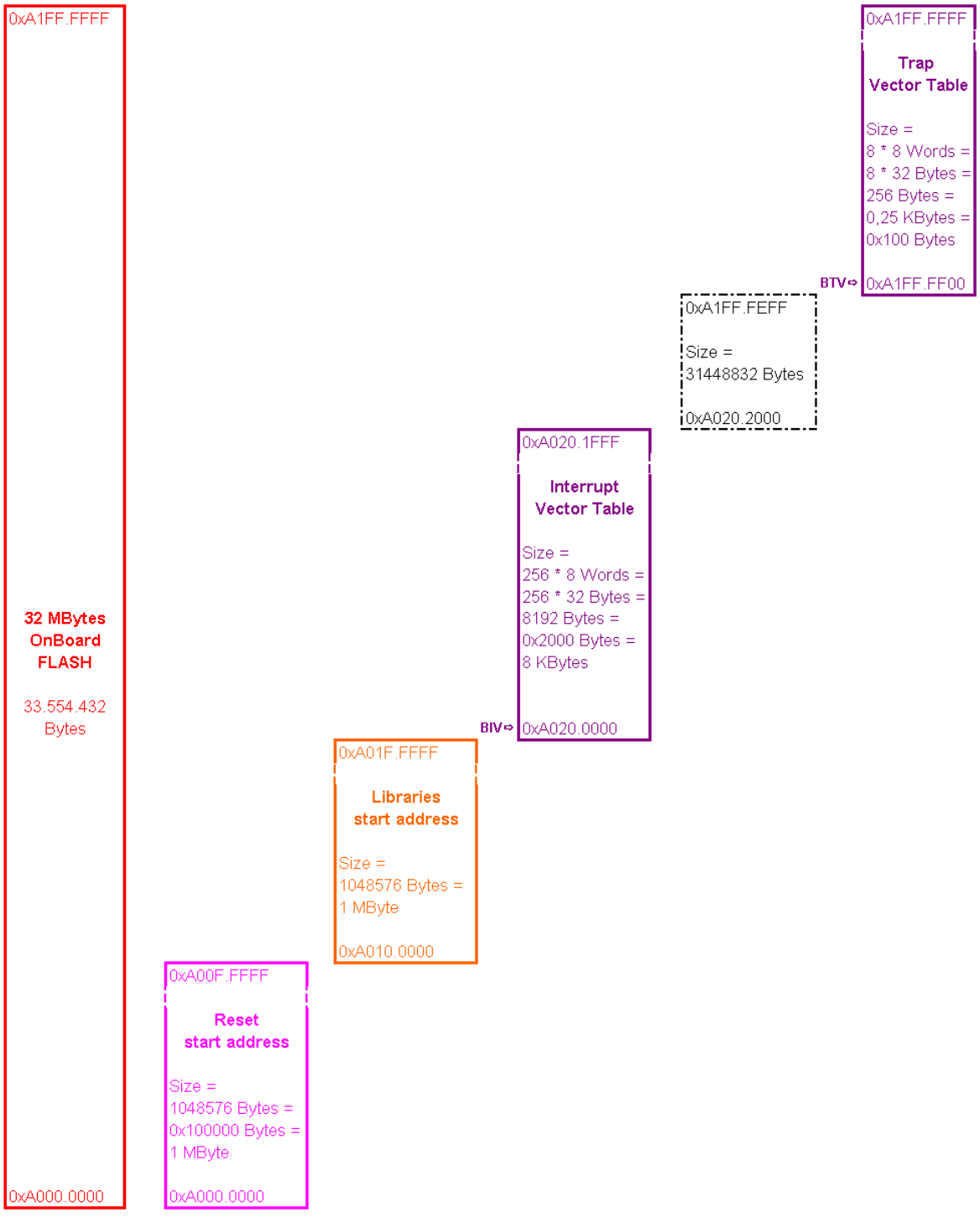


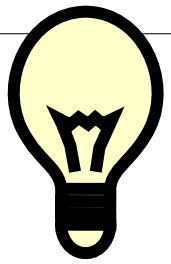
Linker: Script File: Special Areas: RESET start address: insert 0xA0000000 (OnBoardFLASH)
Linker: Script File: Special Areas: Libraries start address: insert 0xA0100000 (OnBoardFLASH)
Linker: Script File: Special Areas: Interrupt table start address: insert 0xA0200000 (OnBoardFLASH)
Linker: Script File: Special Areas: Trap table start address: insert 0xA1FFFF00 (OnBoardFLASH)



Additional Information:

Memory-Map OnBoard-Flash:

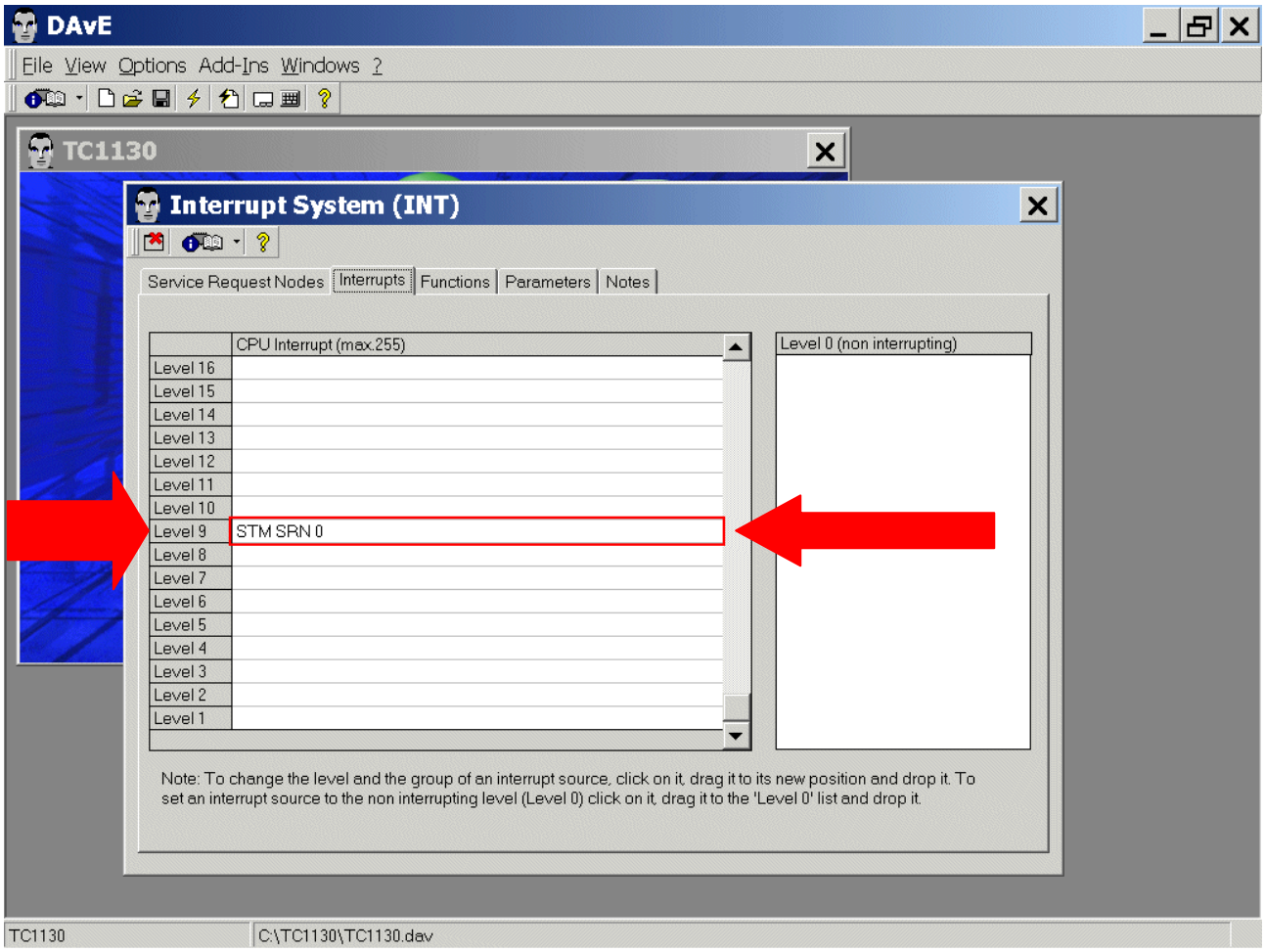


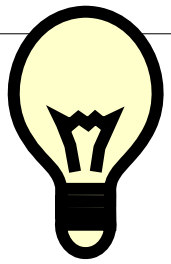


Additional Information:

Interrupt–Vector–Table:

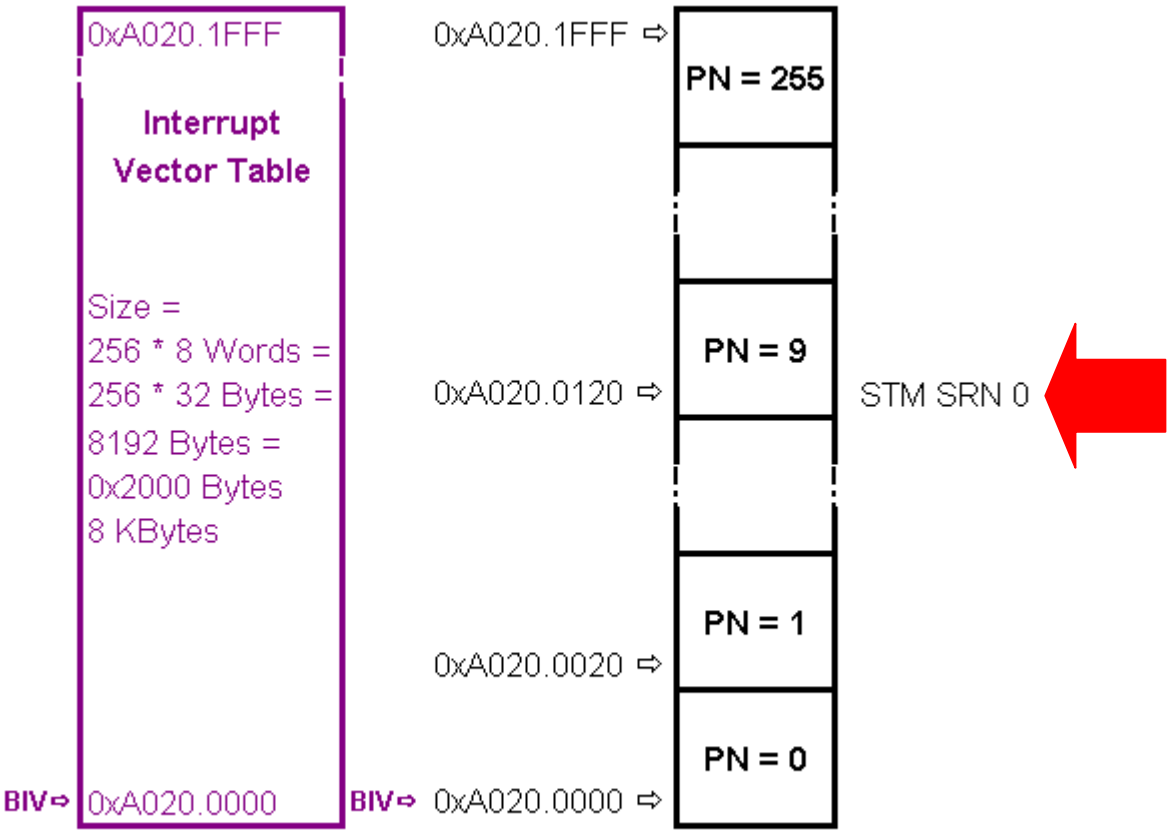
DAvE:





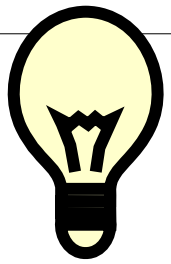
Additional Information:

Interrupt–Vector–Table:



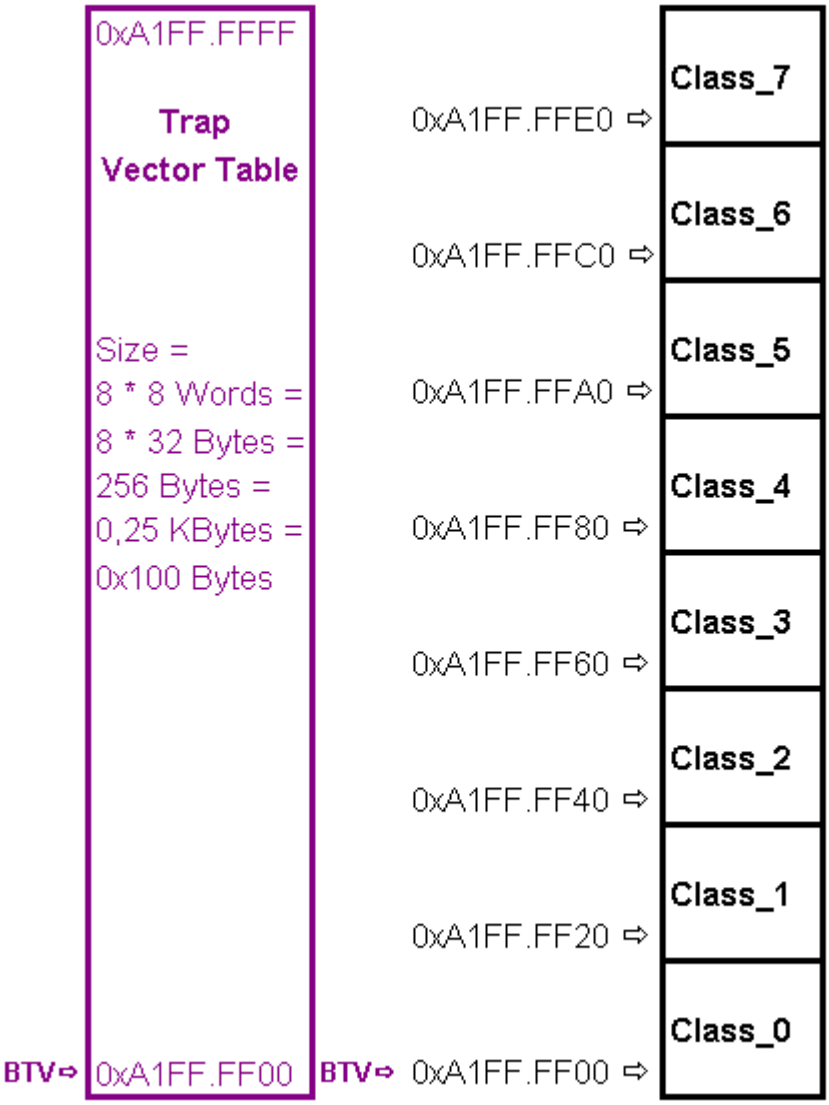
Note:
PN … Priority Number

Click here to see the Map File



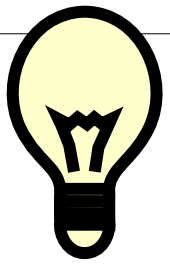
Additional Information:

TRAP–Vector–Table:



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

Click here to see the Map File



Additional Information:

Interrupt Vector Table:

For the full range of 256 interrupt entries an alignment to an 8 KBytes boundary is required.

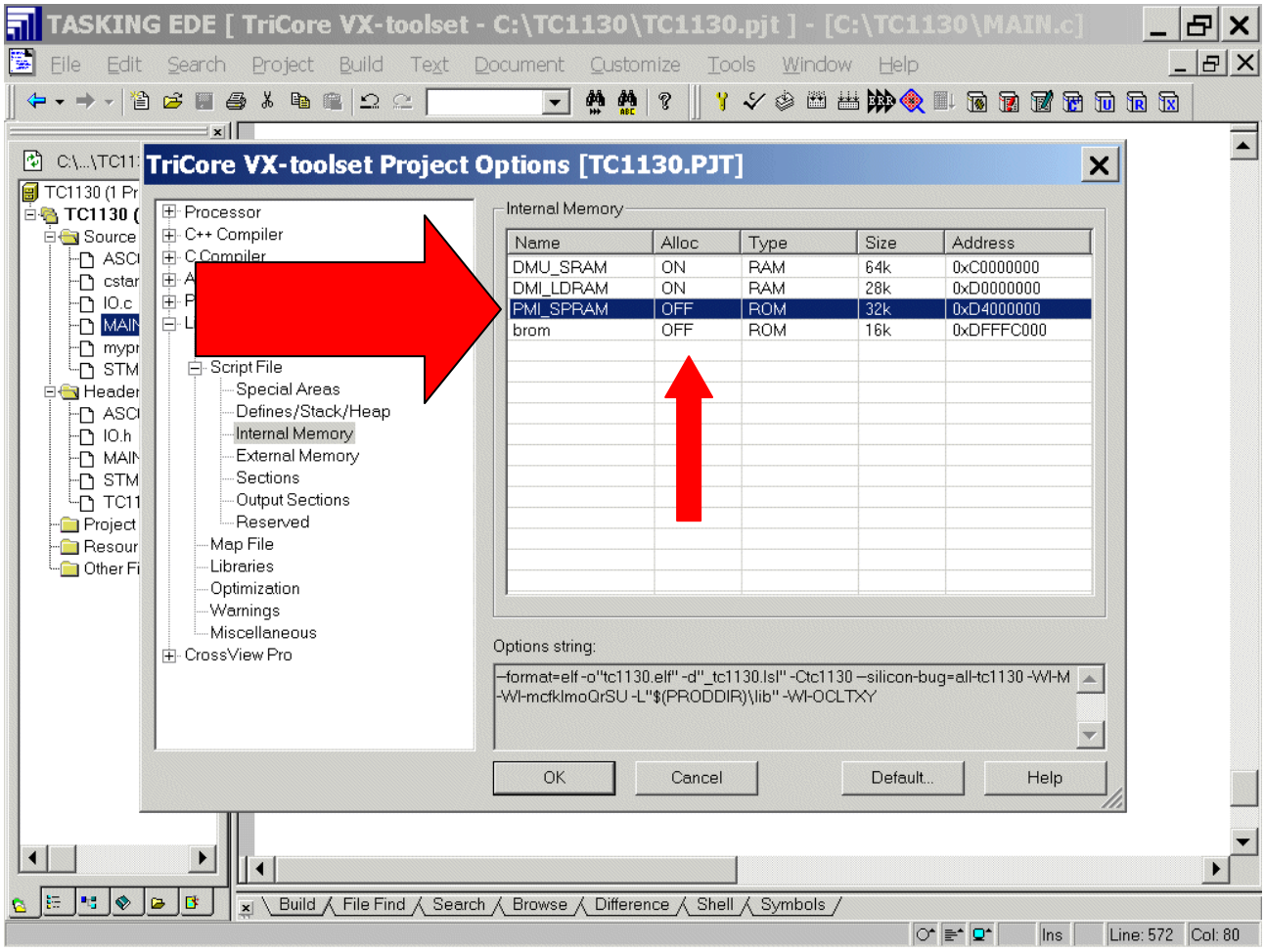
BTV = 0xA020.0000 = 2.686.451.712 / 8.192 = 327.936 ✓

Trap Vector Table:

There are eight different trap classes, resulting in Trap Classes from 0 to 7. The contents of BTV should therefore be at least set to a 256 Byte boundary (8 Trap Classes * 8 word spacing).

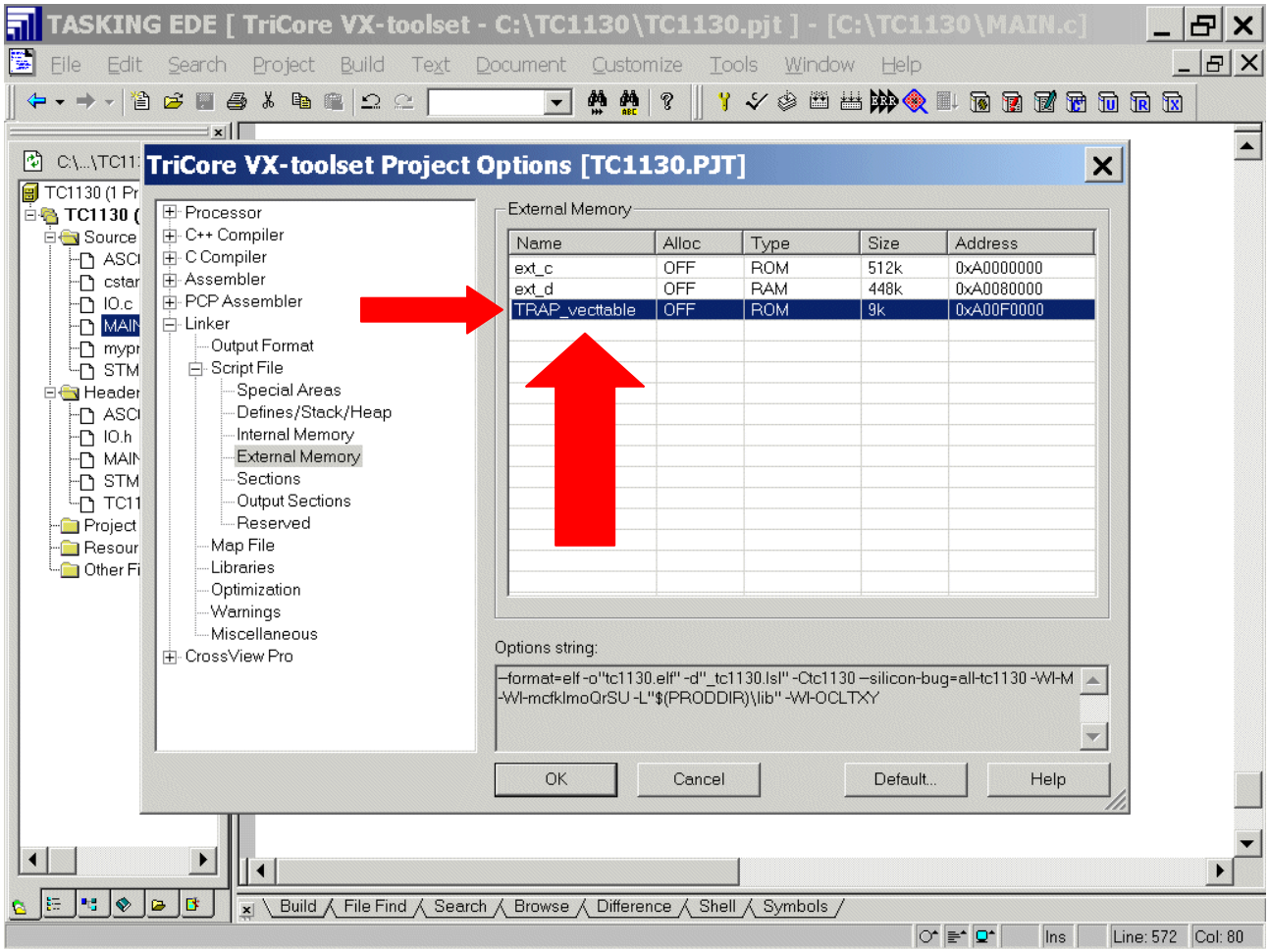
BIV = 0xA1FF.FF00 = 2.717.908.736 / 256 = 10.616.831 ✓

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

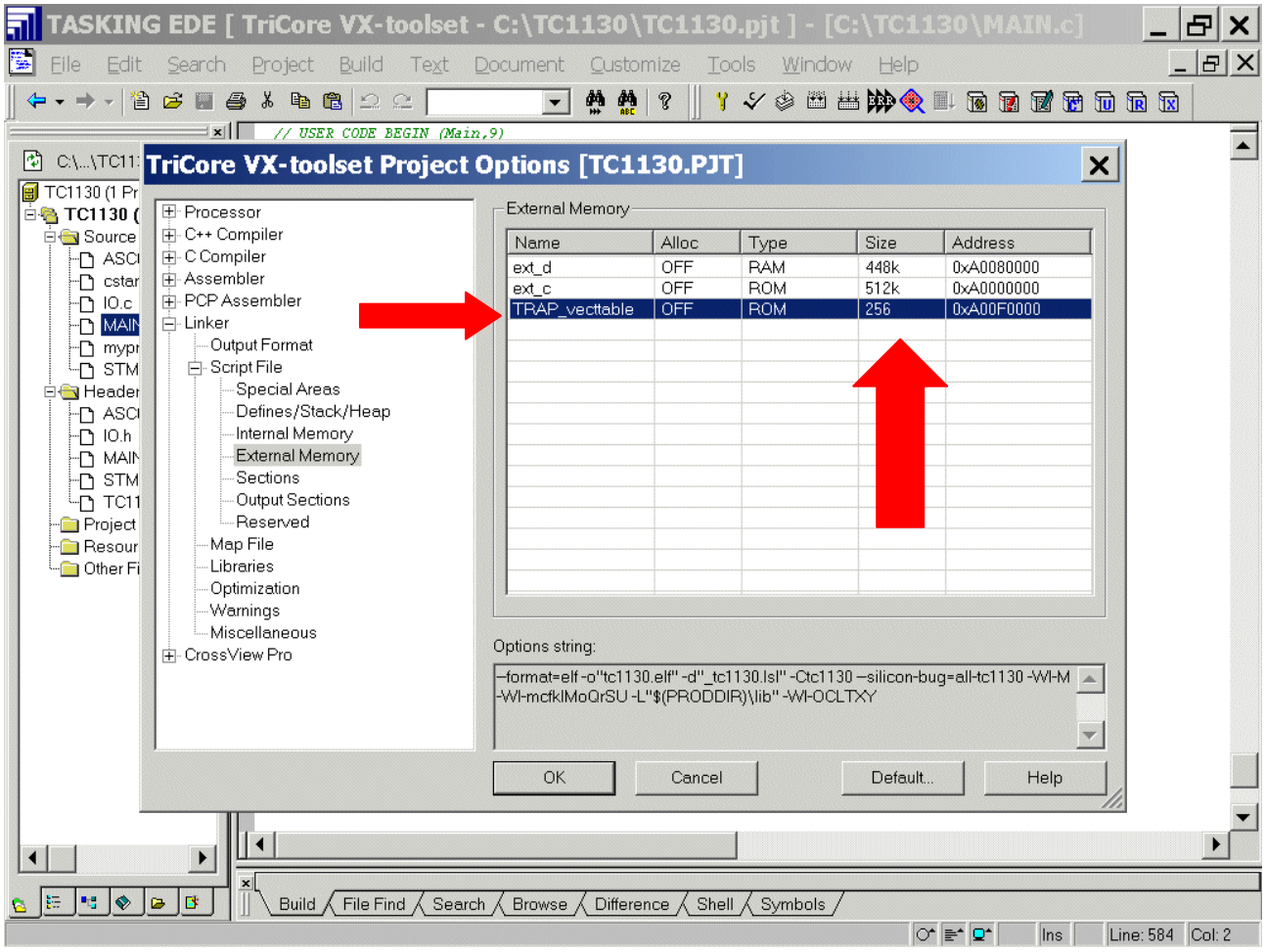


(Configuration of this dialog-window is now finished)

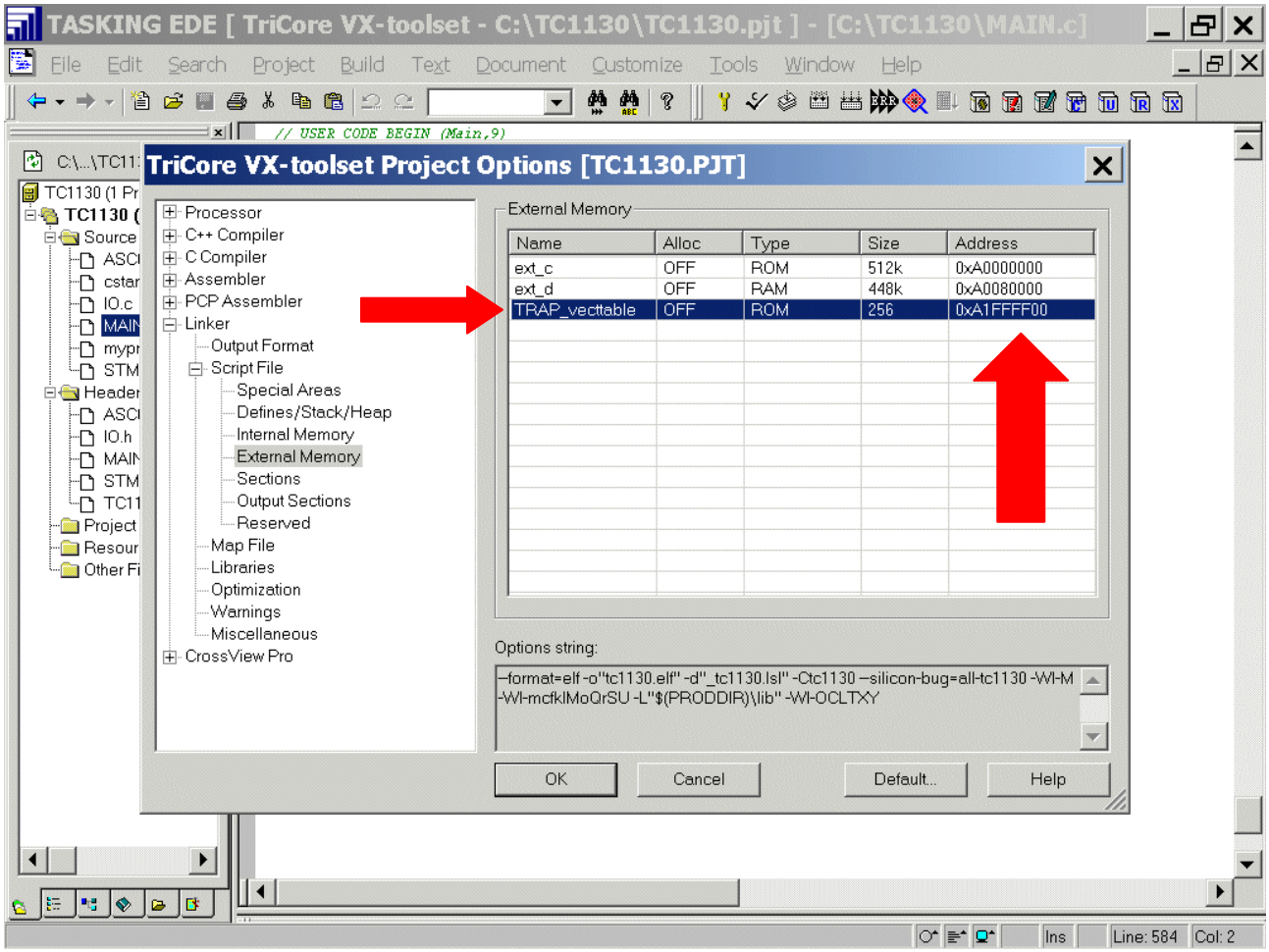
Linker: Script File: External Memory: Name: “vecttable” change to “TRAP_ vecttable”



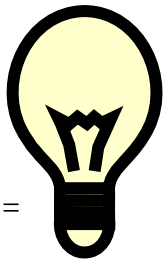
Linker: Script File: External Memory: Size (TRAP_vectable): change to 256



Linker: Script File: External Memory: Address (TRAP_vectable): change to 0xA1FFFF00



Note:

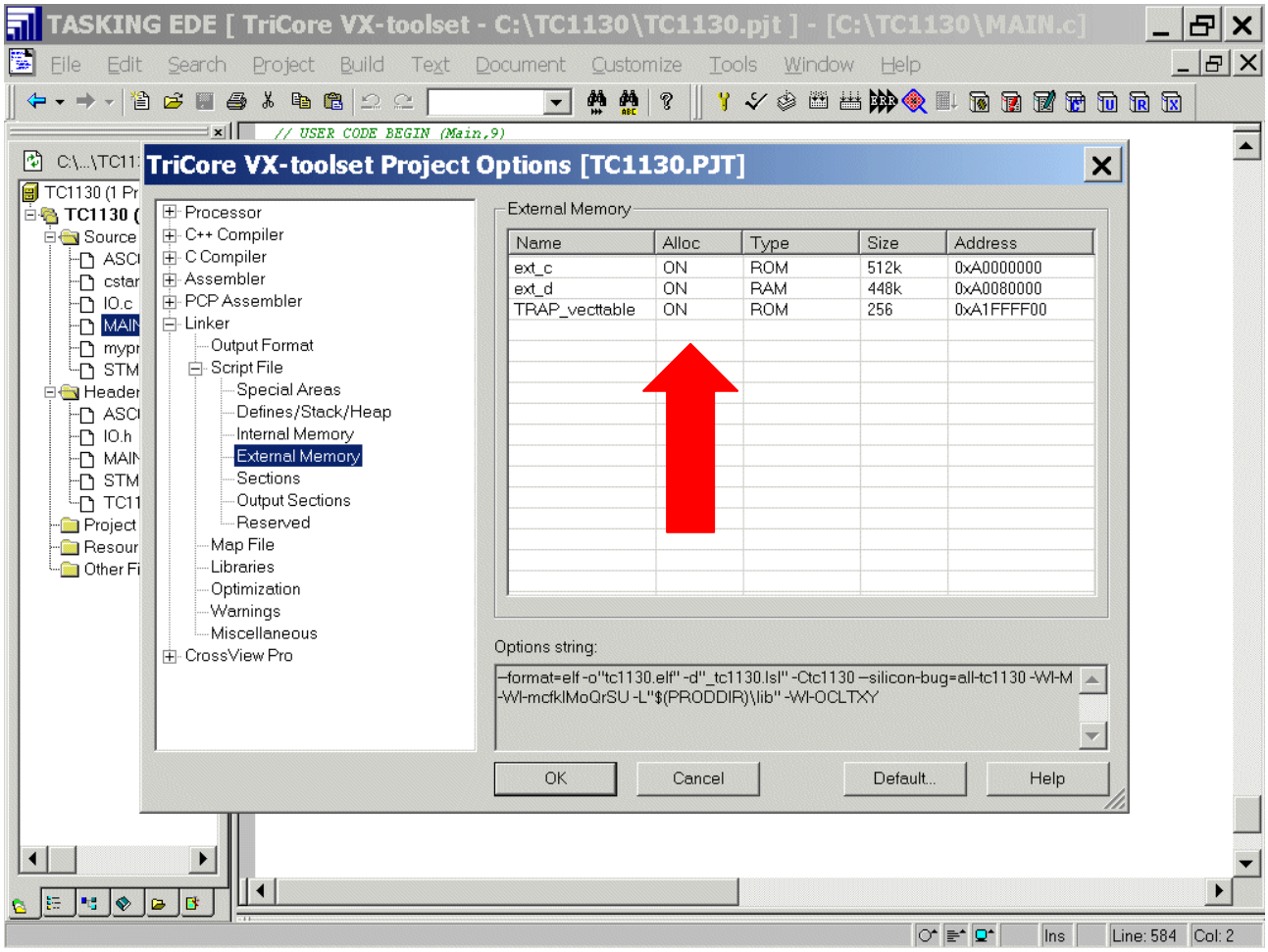


32 Mbytes OnBoardFlash
from A000.0000 to A1FF.FFFF – 256 Bytes “TRAP_ vecttable” =
from A000.0000 to 0xA1FFDFFF = 33.546.240 Bytes

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

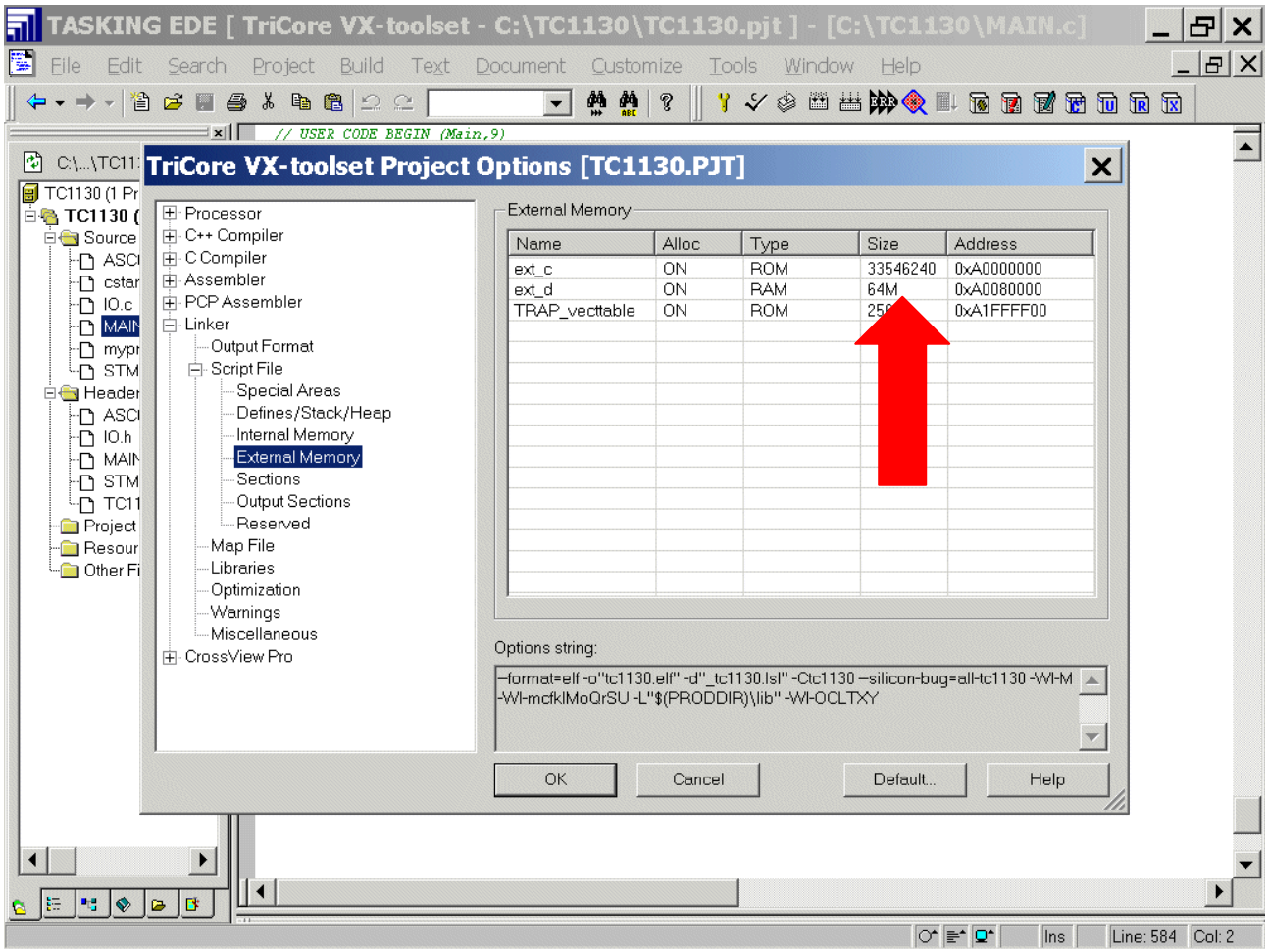
Linker: Script File: External Memory: Name=ext_d: Alloc: select “ON”

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

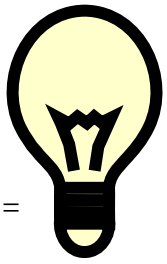


Linker: Script File: External Memory: Name=ext_c: Size: insert “33546240” [ROM]

Linker: Script File: External Memory: Name=ext_d: Size: insert “64M” [RAM]



Note:



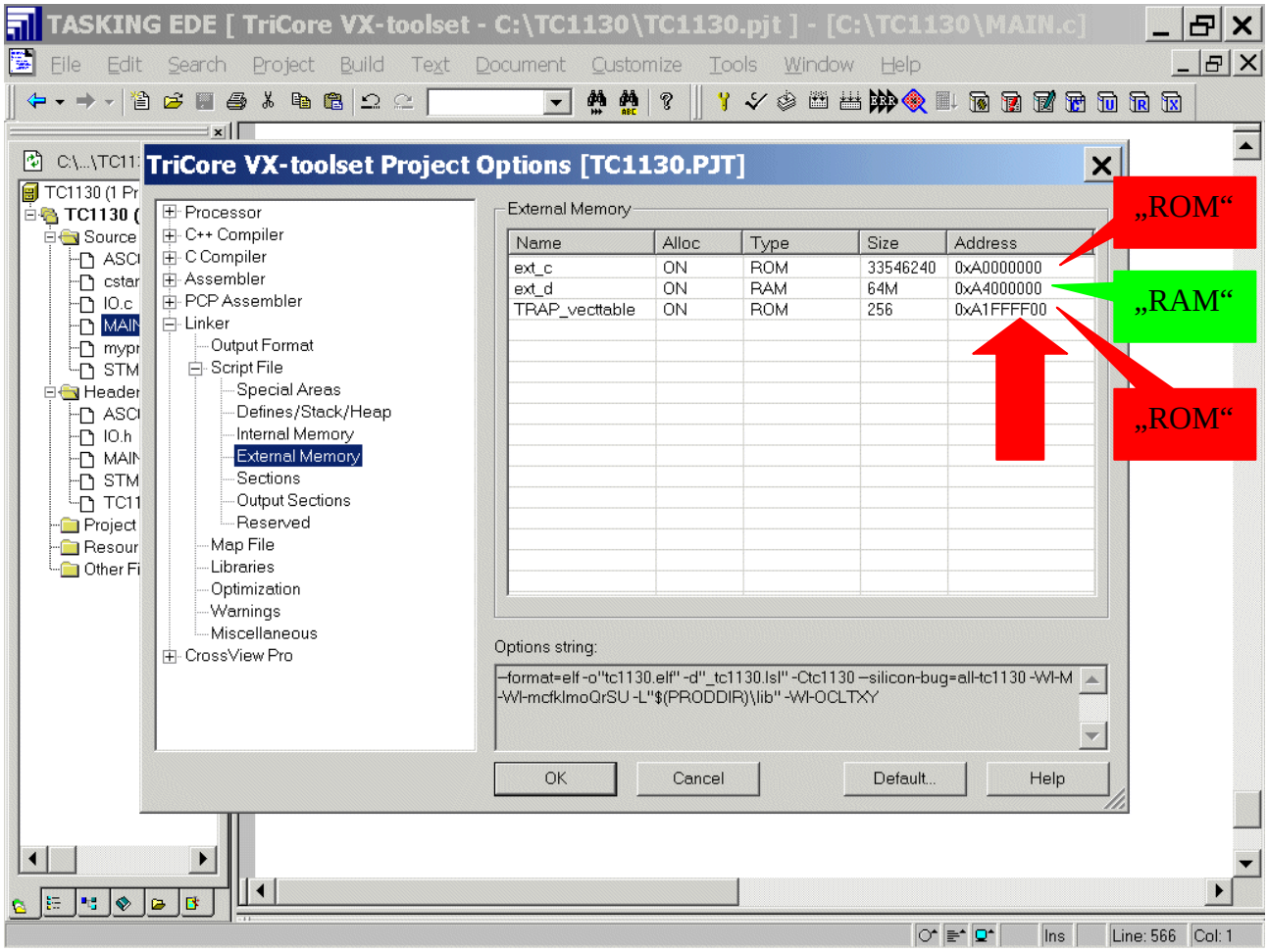
32 Mbytes OnBoardFlash
from A000.0000 to A1FF.FFFF – 256 Bytes “TRAP_ vecttable” =
from A000.0000 to 0xA1FFDFFF = 33.546.240 Bytes

Linker: Script File: External Memory:

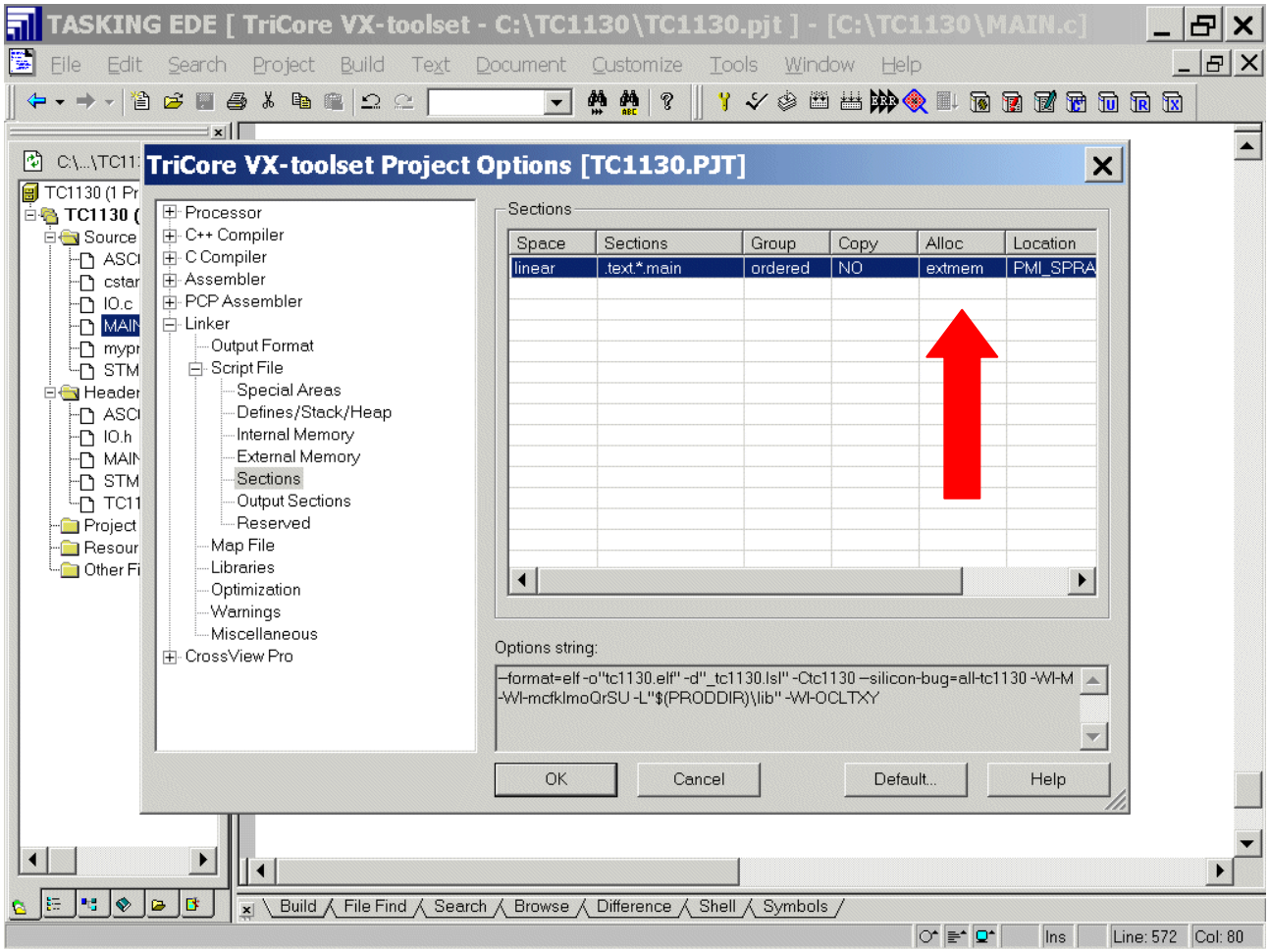

Name=ext_c: Address: check/insert 0xA0000000 (ROM)


Name=ext_d: Address: insert 0xA4000000 (RAM)

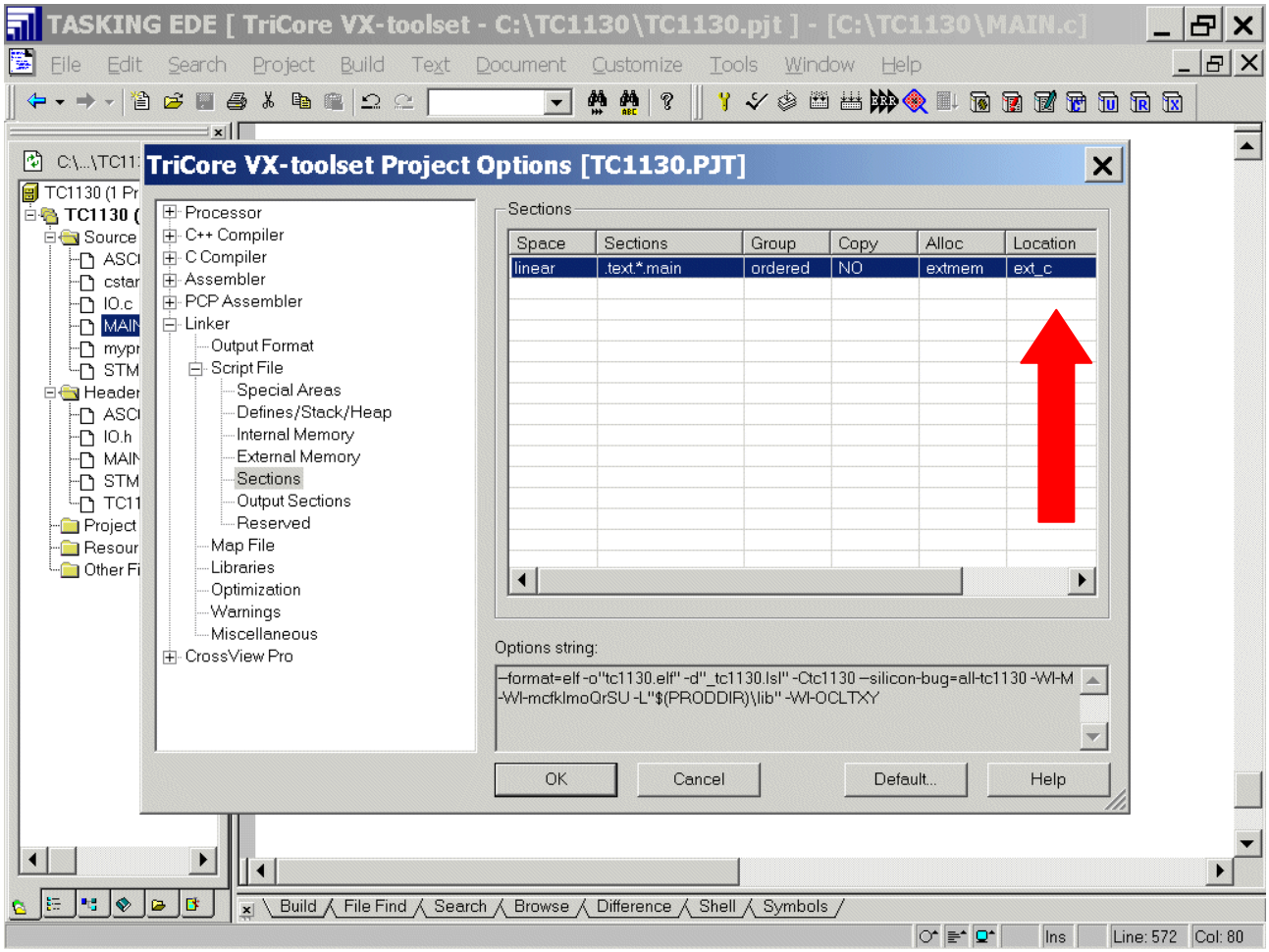

Name=TRAP_vecttable: Address: check/insert 0xA1FFFF00 (ROM)



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

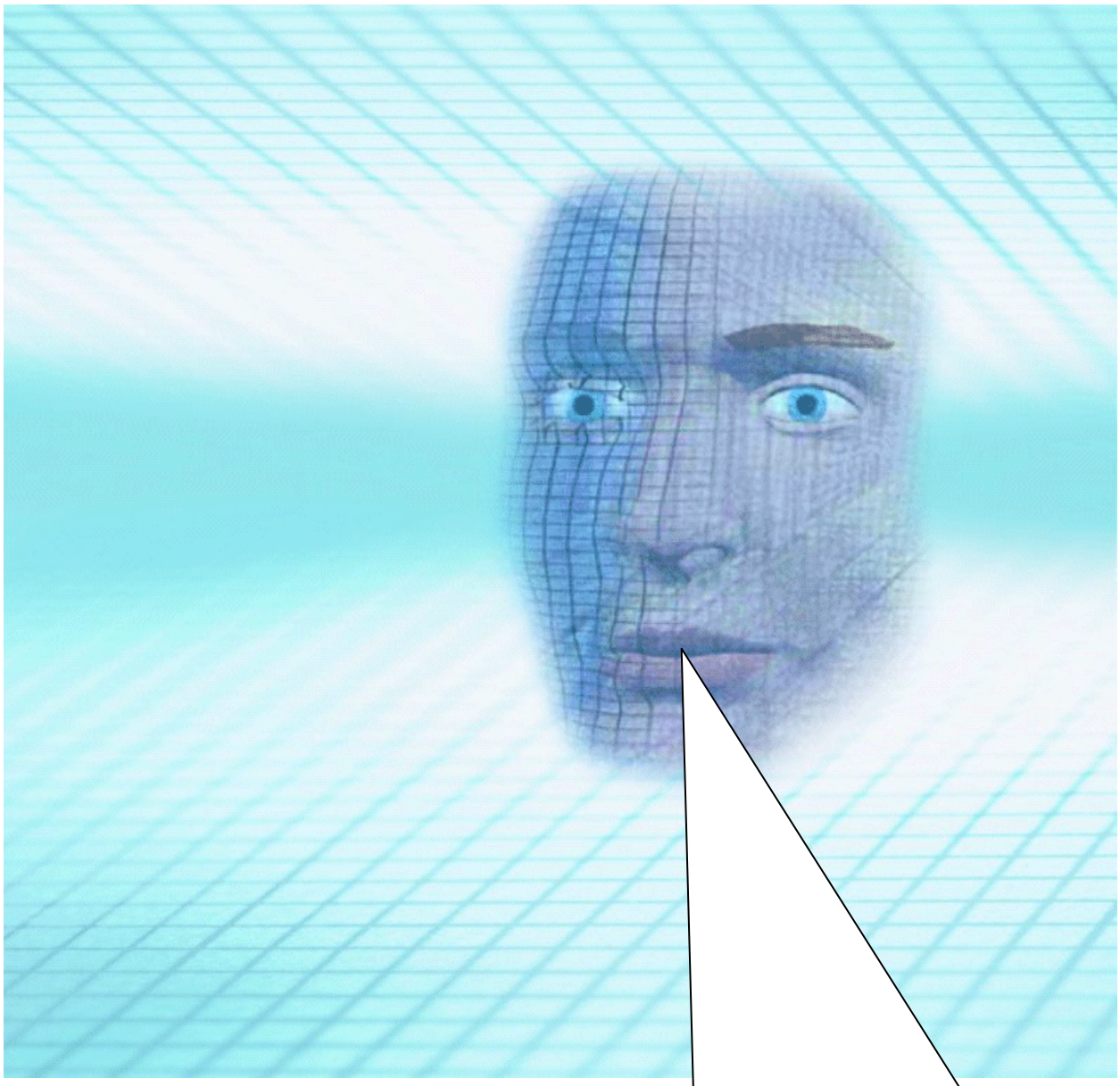


Linker: Script File: Sections: Space=linear: Location: insert “ext_c”



OK

Insert your application specific program:



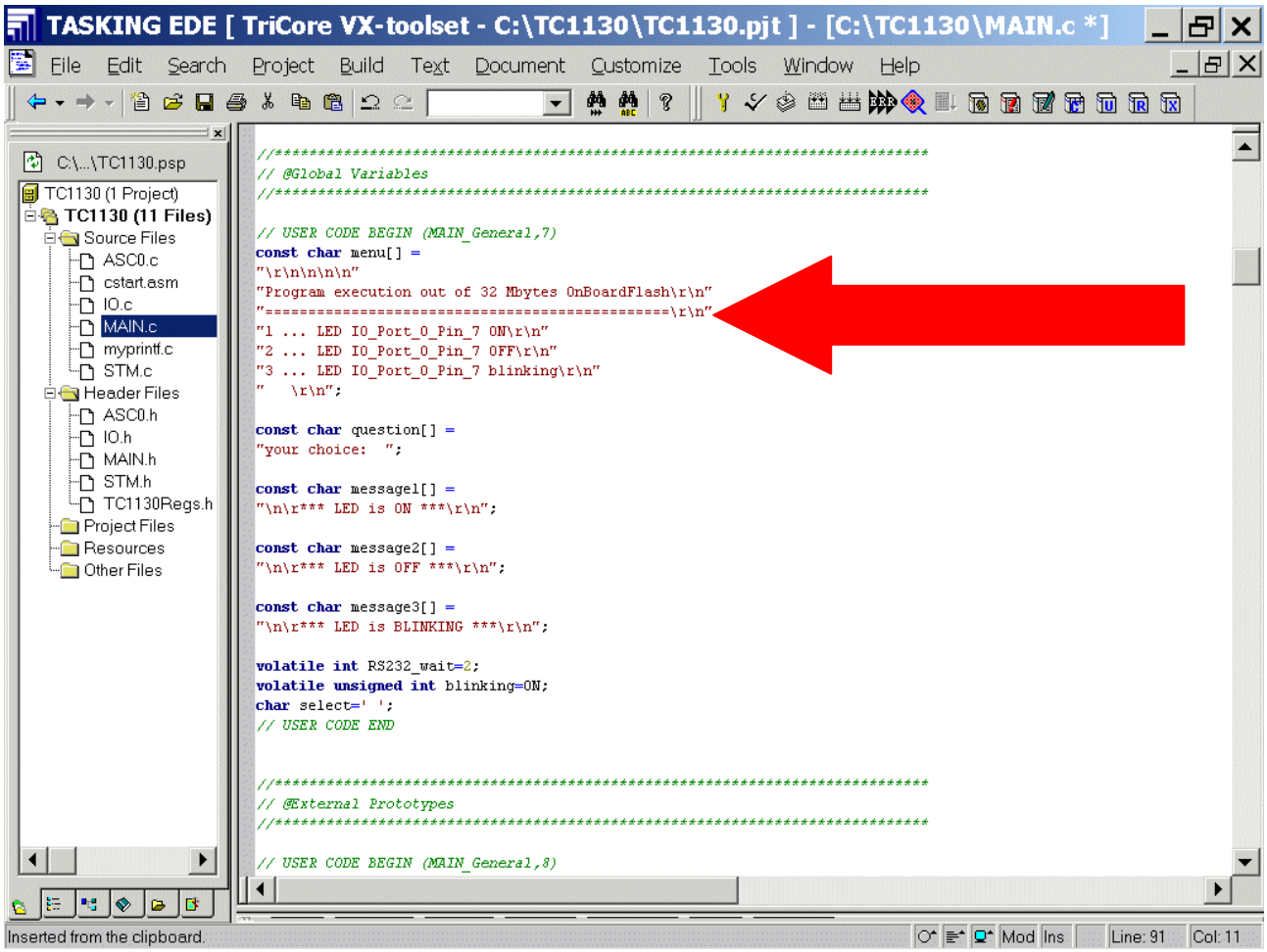
Note:
DAvE doesn't change code which is inserted between '// USER CODE BEGIN' and '// USER CODE END'. Therefore, whenever adding code to DAvE's generated code, write it between '// USER CODE BEGIN' and '// USER CODE END'.
If you wish to change DAvE´s generated code or add code outside these 'USER CODE' 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 PMI_SPRAM\r\n"
"==================================\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 blinking\r\n"
"   \r\n";
```

to

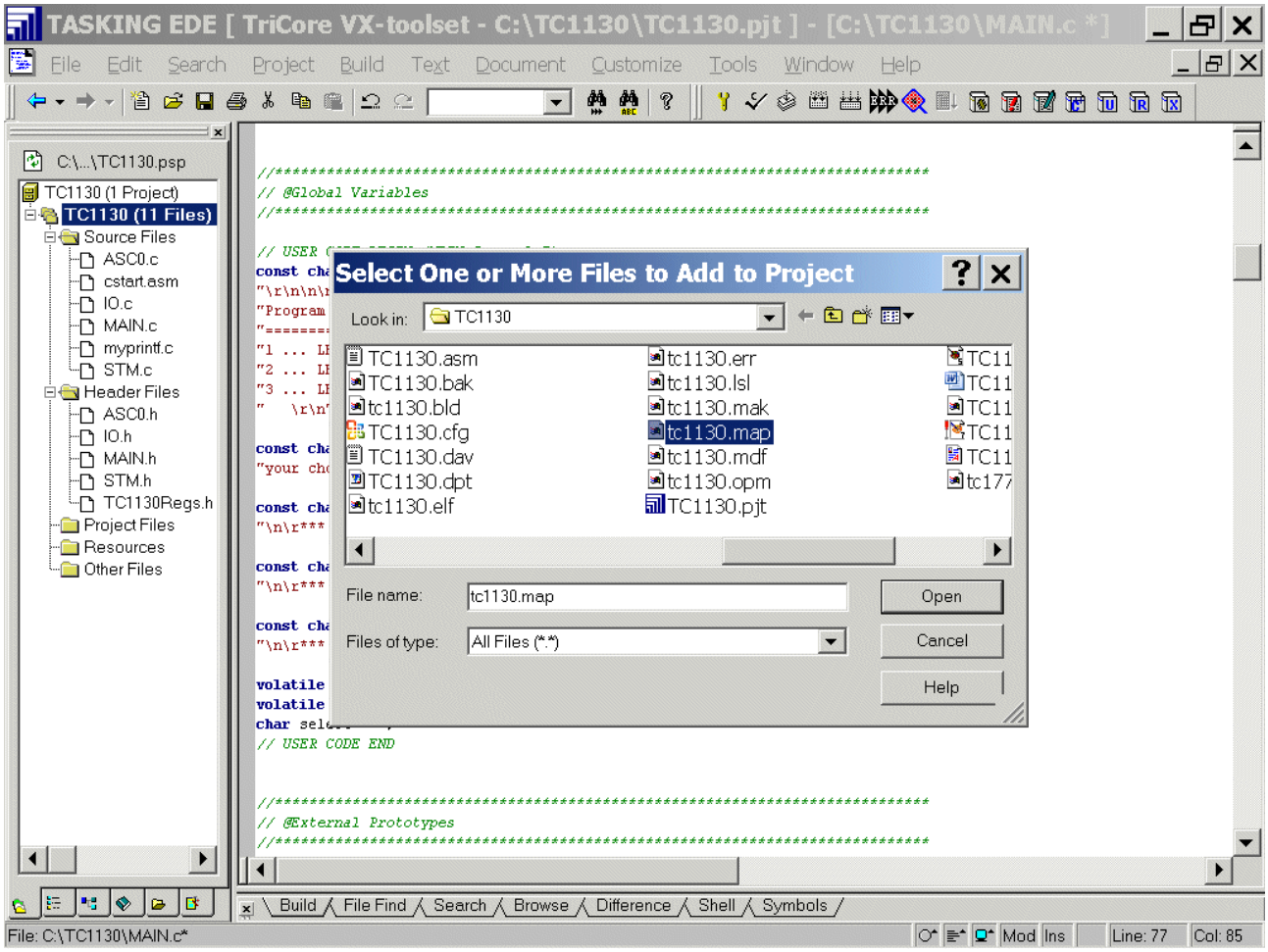
```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of 32 MBytes OnBoardFlash\r\n"
"================================================\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 blinking\r\n"
"   \r\n";
```



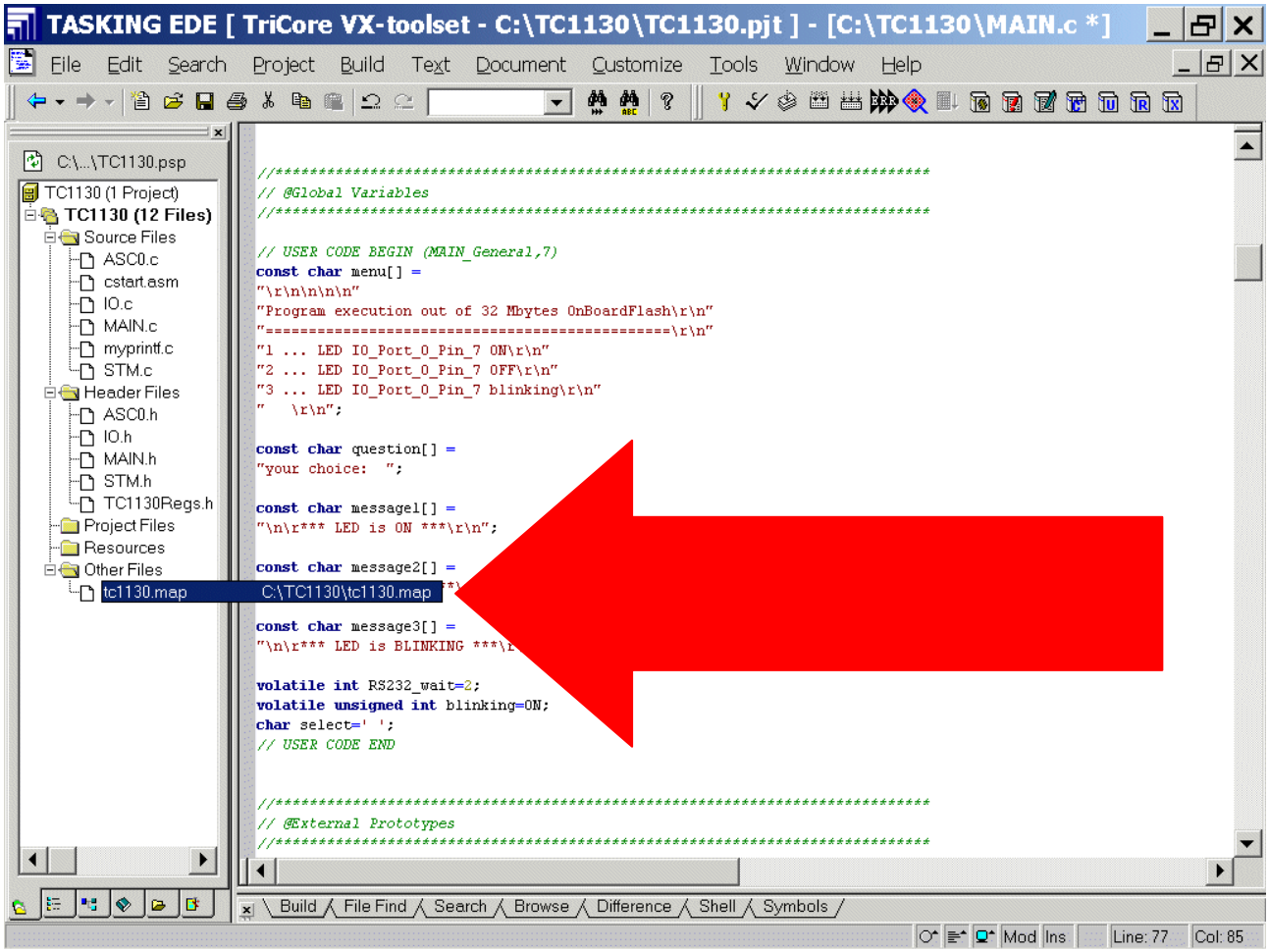
Insert Map File:

(Project Window File View) – TC1130 (Files) – right mouse button click – Add Existing Files – Browse

Select TC1130.map



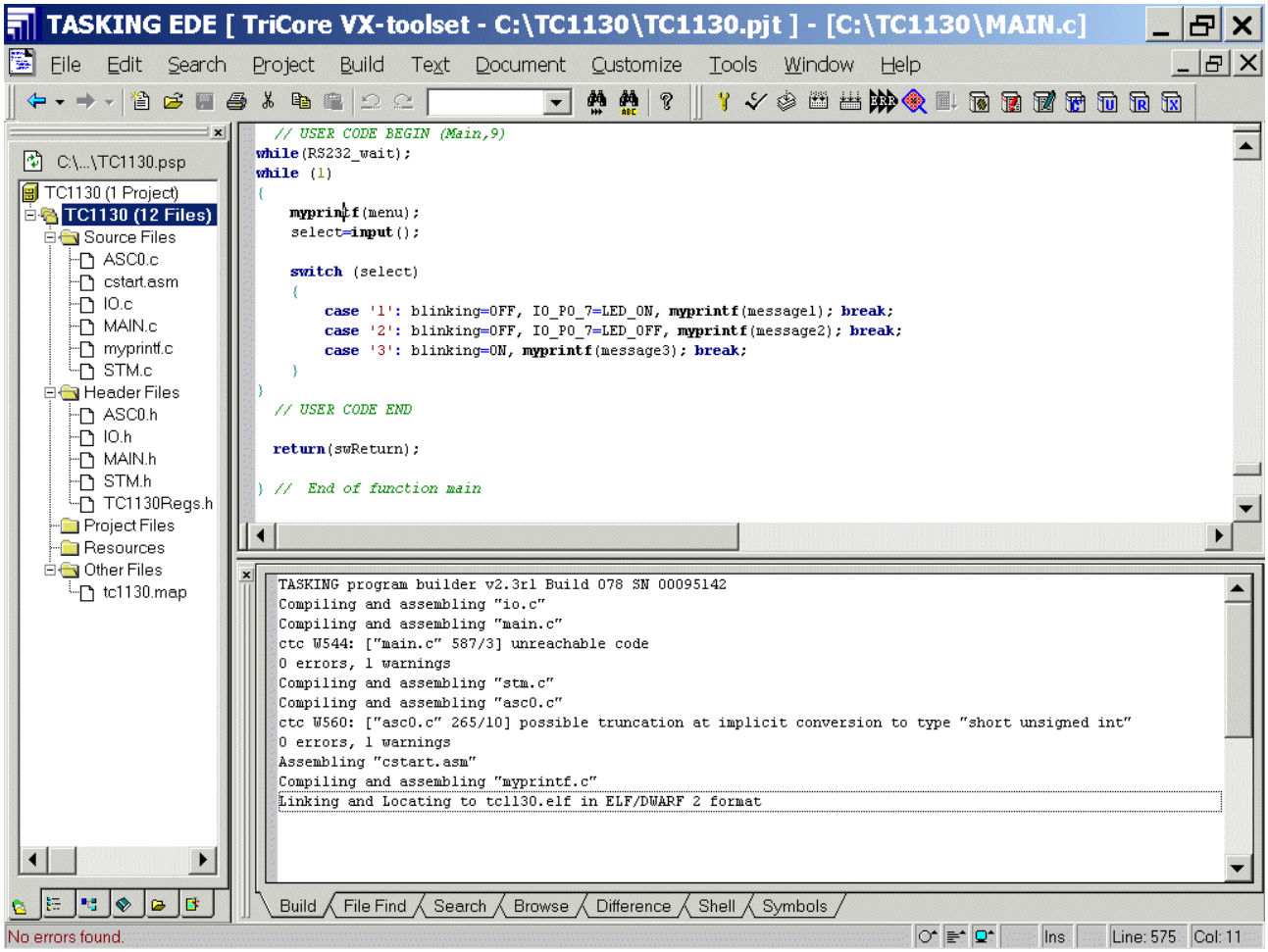
Open - OK


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1130.psp
TC1130 (1 Project)
TC1130 (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
TC1130Regs.h
Project Files
Resources
Other Files
tc1130.map
C:\TC1130\tc1130.map
// @Global Variables
// USER CODE BEGIN (MAIN_General,7)
const char menu[] =
"Program execution out of 32 Mbytes OnBoardFlash\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 blinking\r\n"
const char question[] =
"your choice: ";
const char message1[] =
"\n\r*** LED is ON ***\r\n";
const char message2[] =
const char message3[] =
"\n\r*** LED is BLINKING ***\r
volatile int RS232_wait=2;
volatile unsigned int blinking=ON;
char select=' ';
// USER CODE END
// @External Prototypes
Build File Find Search Browse Difference Shell Symbols
Mod Ins Line: 77 Col: 85

Generate your application program:

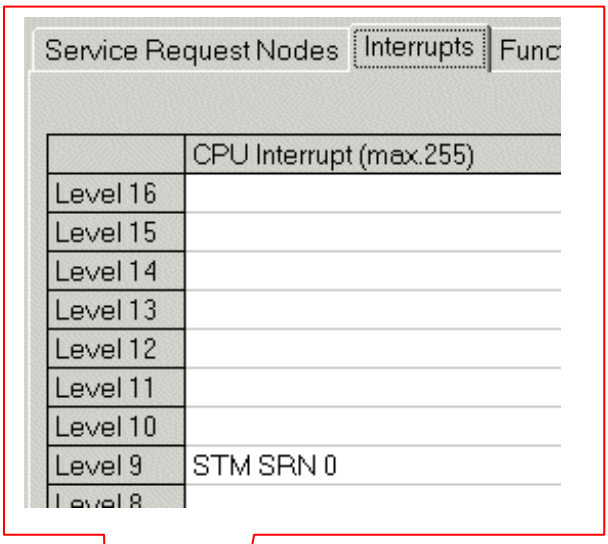
Build
Rebuild

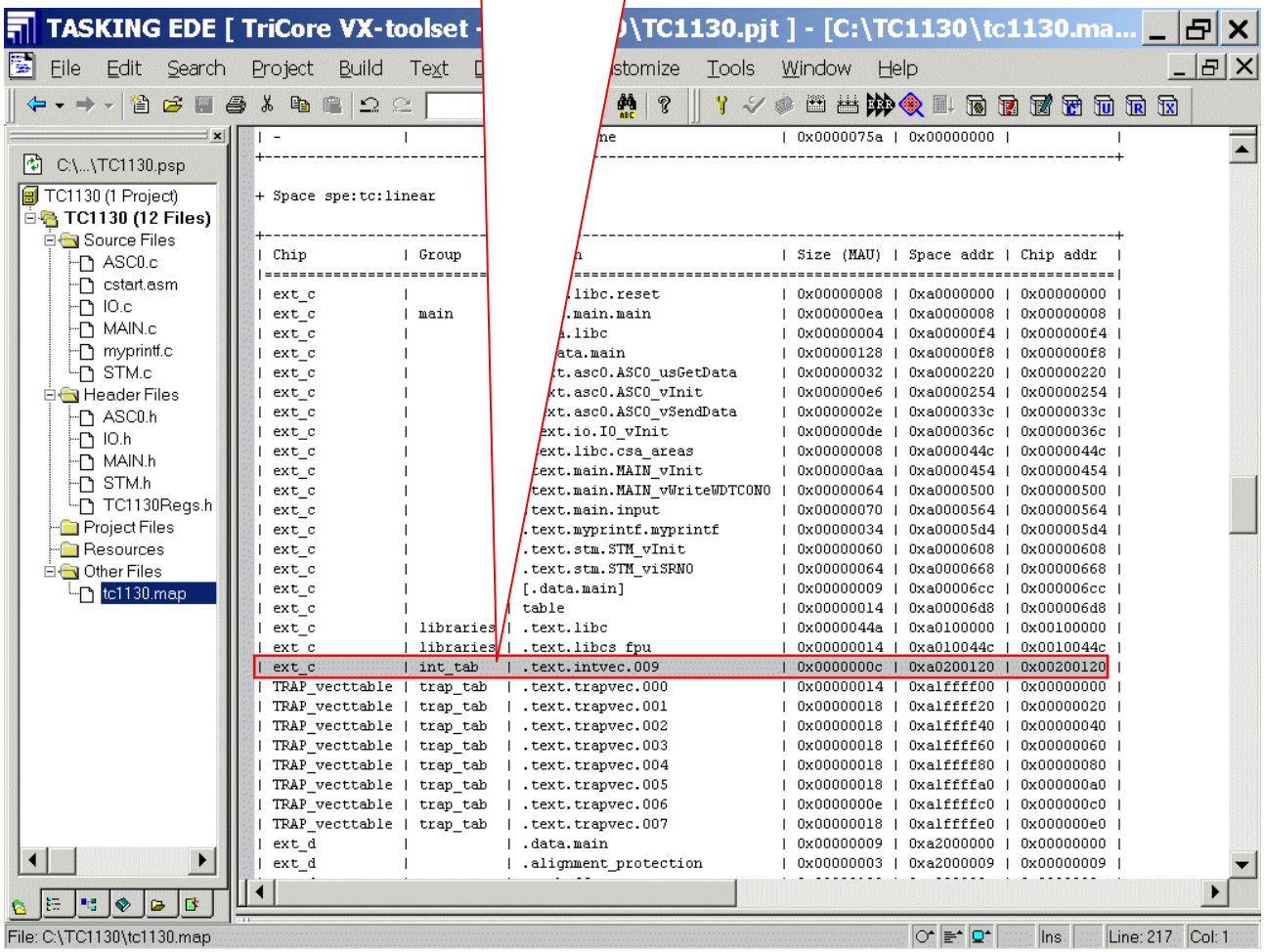
or



See Map File:

Interrupt Vector Table:

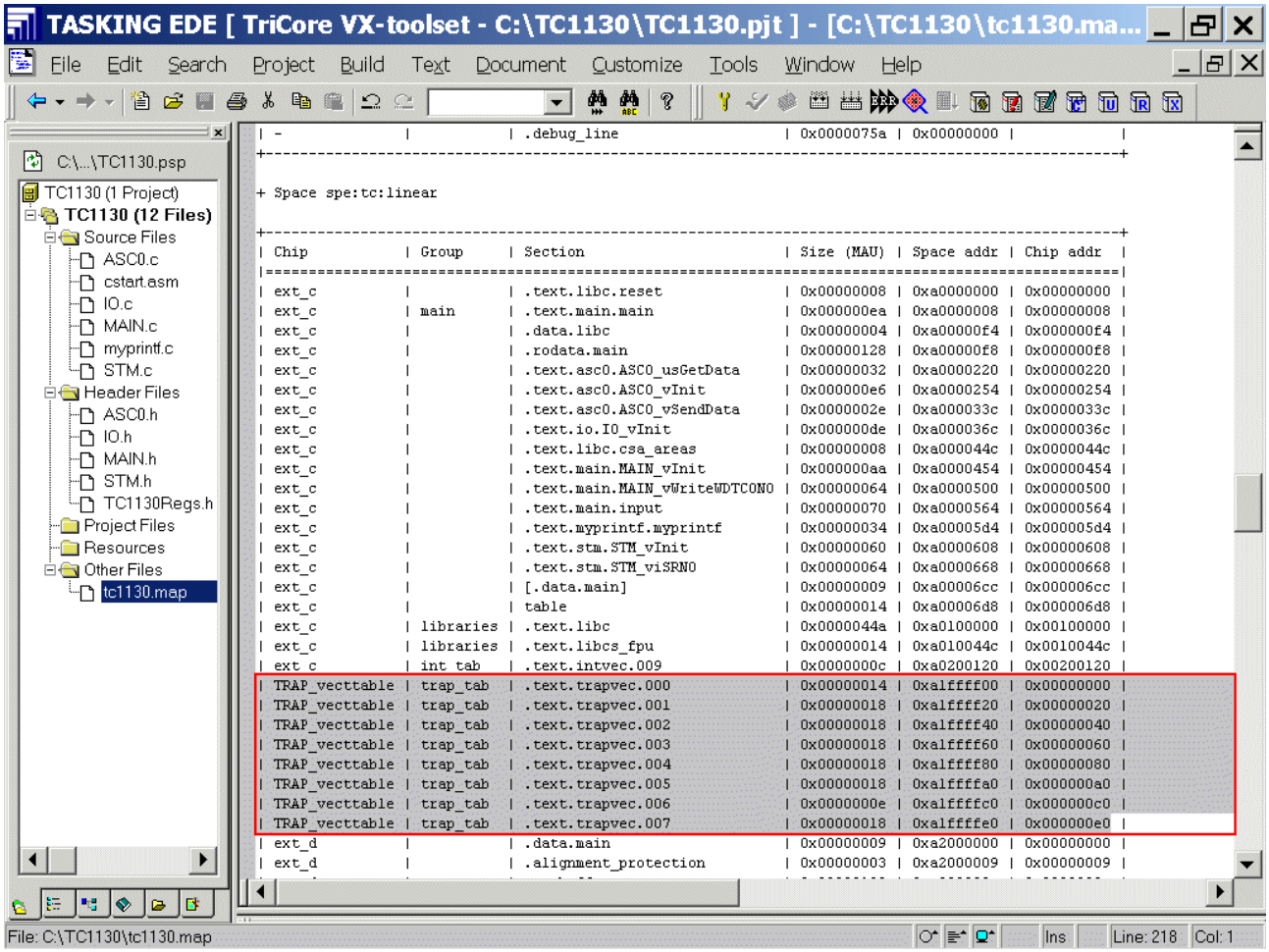




Click here to see Memory Map (Interrupt Vector Table).

See Map File:

Trap Vector Table:



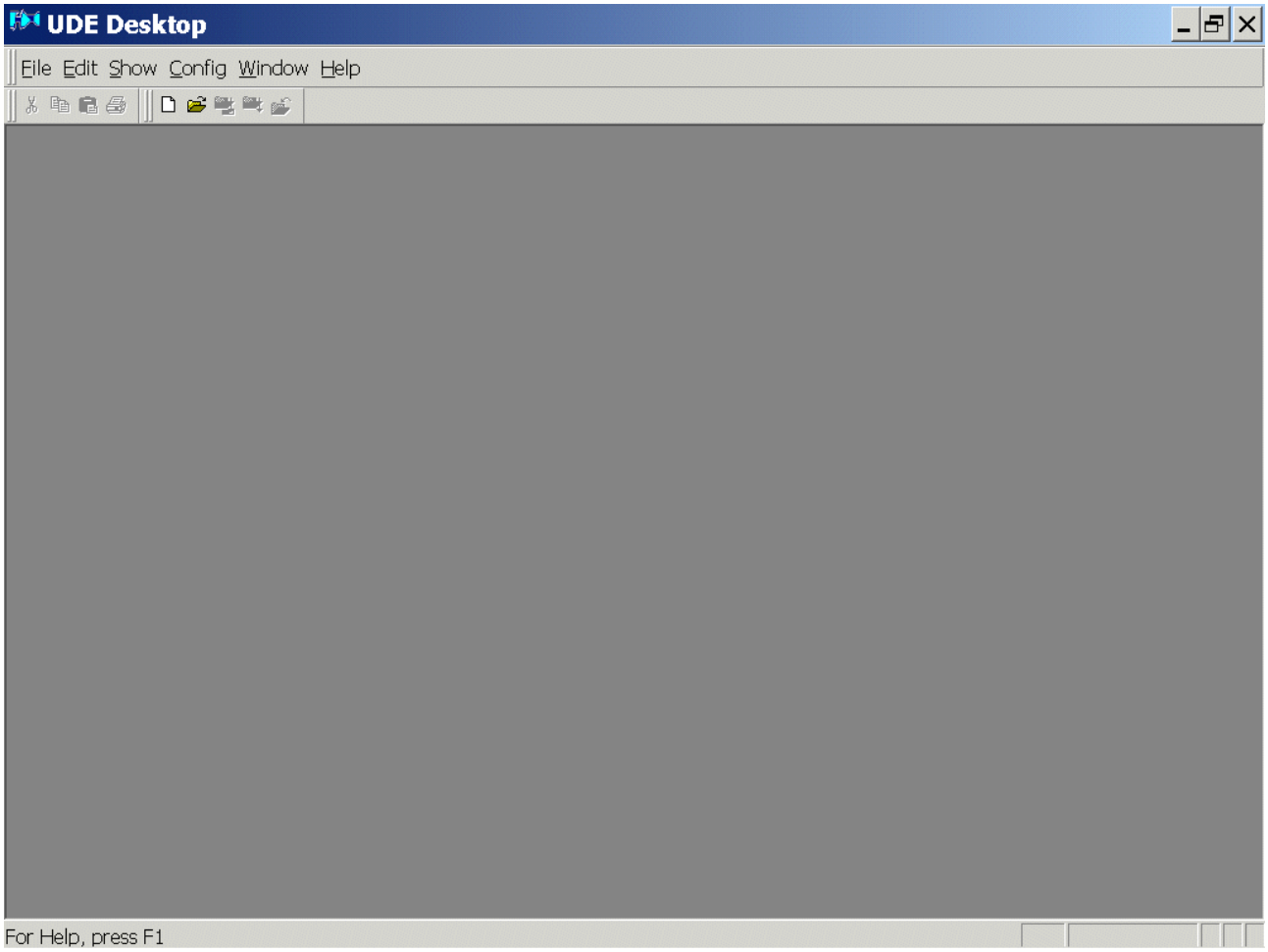
Click here to see Memory Map (Trap Vector Table).

Now you can close both your project and Tasking EDE:

File - Close Project Space
File – Exit

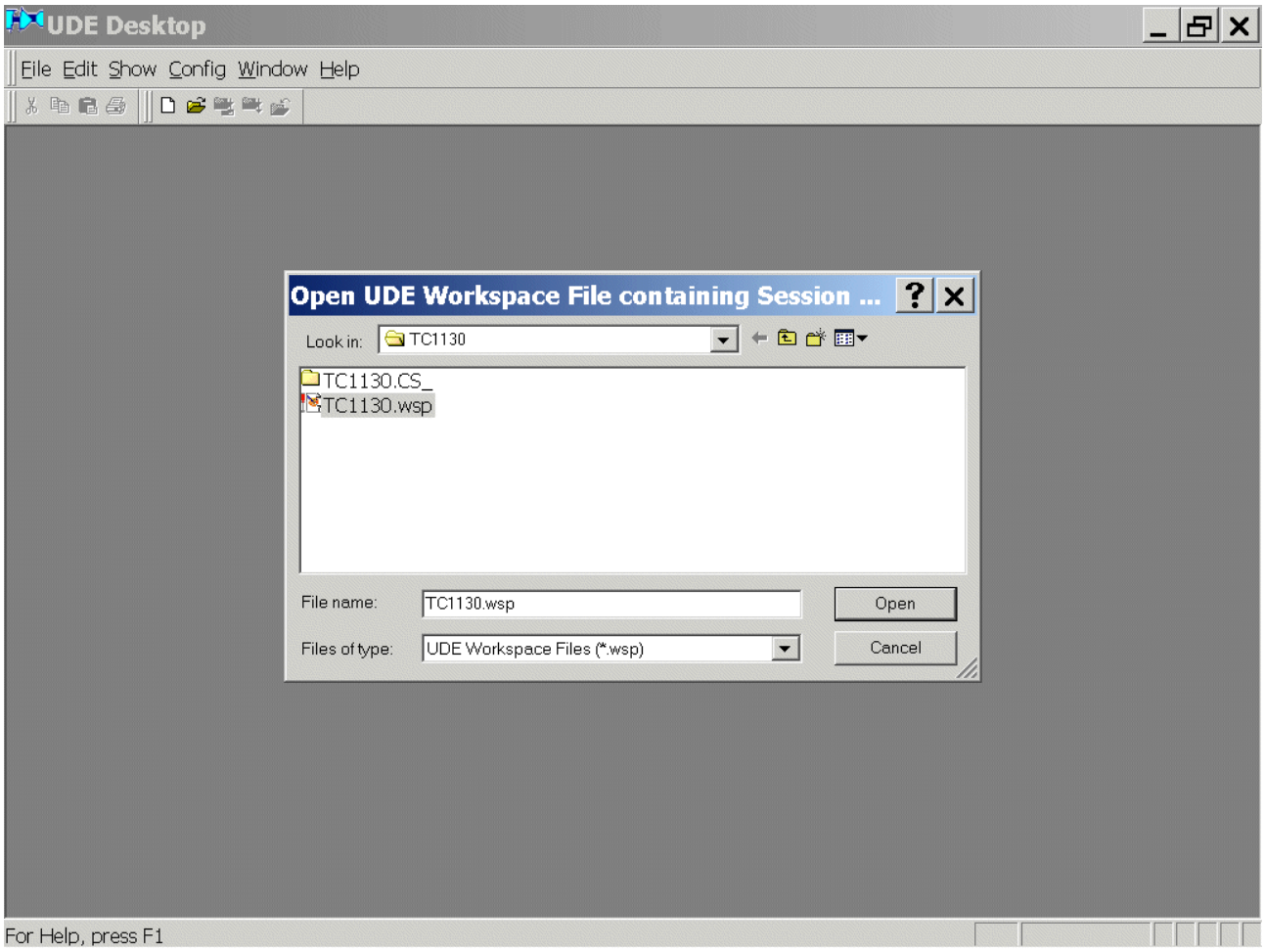
Programming is now complete. You can now load and run your program:

Start pls-Debugger

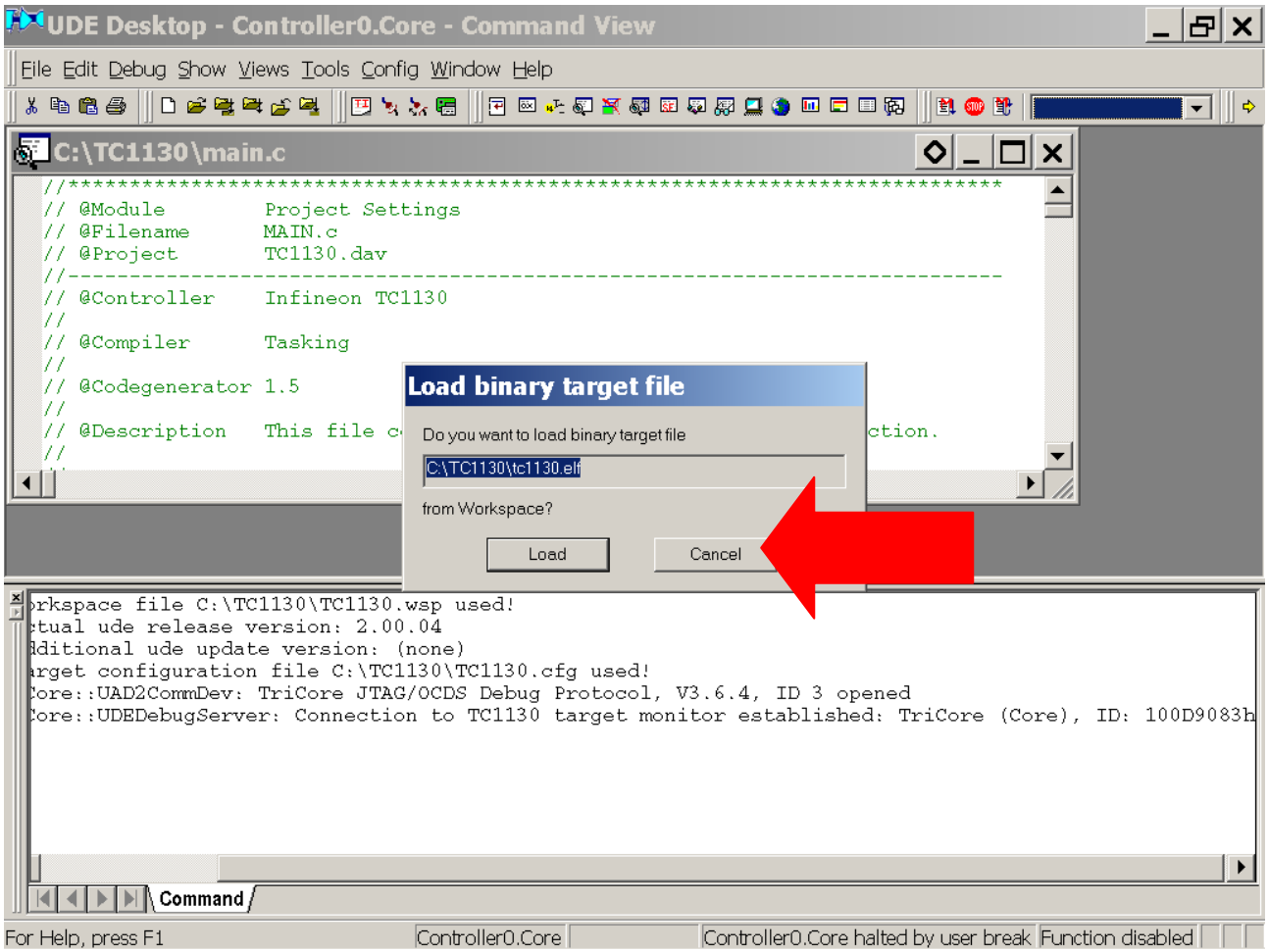


File – Open Workspace

Look in: select C:\TC1130
File name: select TC1130.wsp

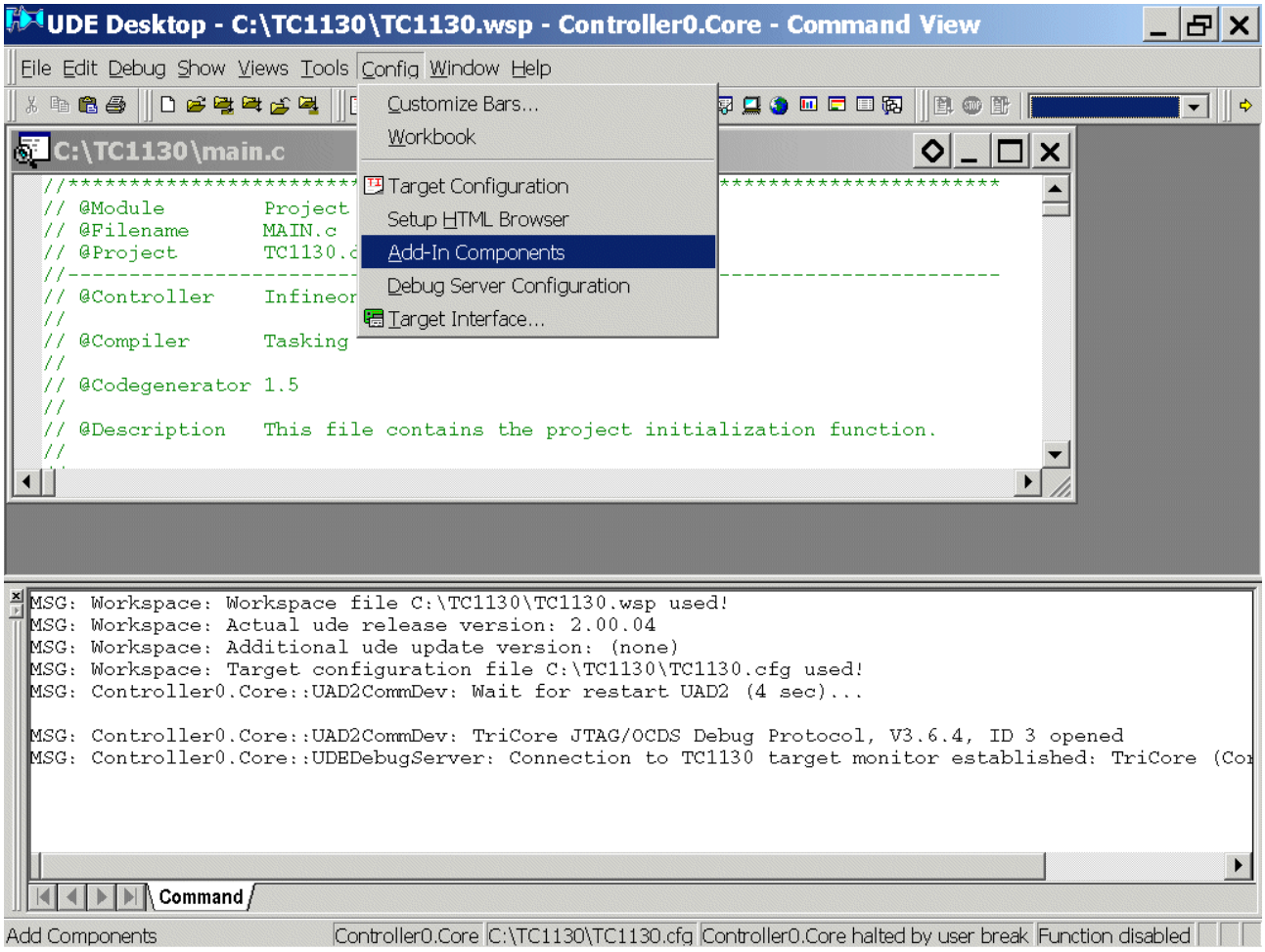


Open

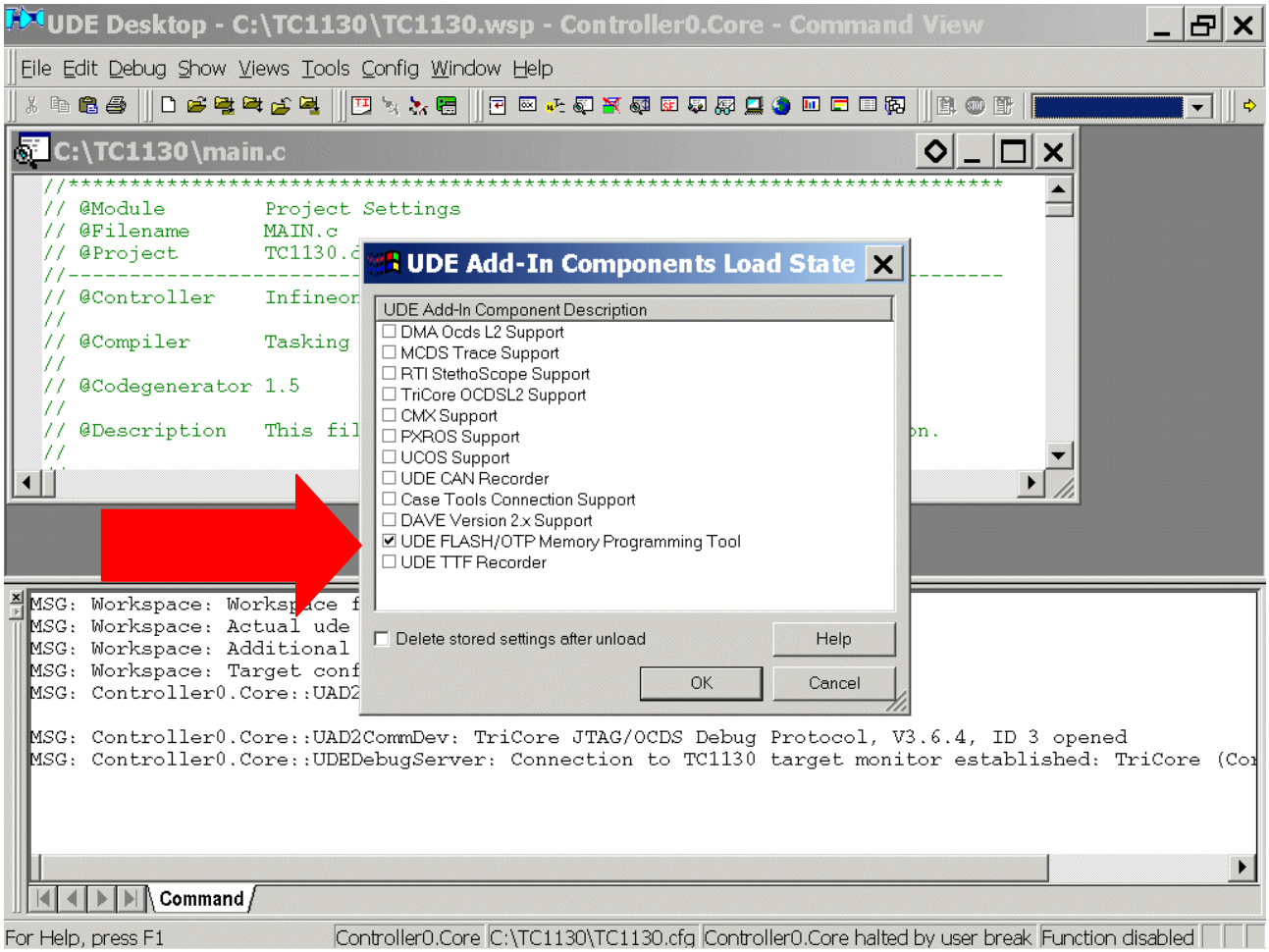

UDE Desktop - Controller0.Core - Command View
File Edit Debug Show Views Tools Config Window Help
C:\TC1130\main.c
// @Module Project Settings
// @Filename MAIN.c
// @Project TC1130.dav
// @Controller Infineon TC1130
// @Compiler Tasking
// @Codegenerator 1.5
// @Description This file c
ction.
Load binary target file
Do you want to load binary target file
C:\TC1130\tc1130.elf
from Workspace?
Load
Cancel
rkspace file C:\TC1130\TC1130.wsp used!
tual ude release version: 2.00.04
ditional ude update version: (none)
rget configuration file C:\TC1130\TC1130.cfg used!
ore::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
ore::UDEDebugServer: Connection to TC1130 target monitor established: TriCore (Core), ID: 100D9083h
Command
For Help, press F1
Controller0.Core
Controller0.Core halted by user break
Function disabled

Cancel

Config – Add-In Components

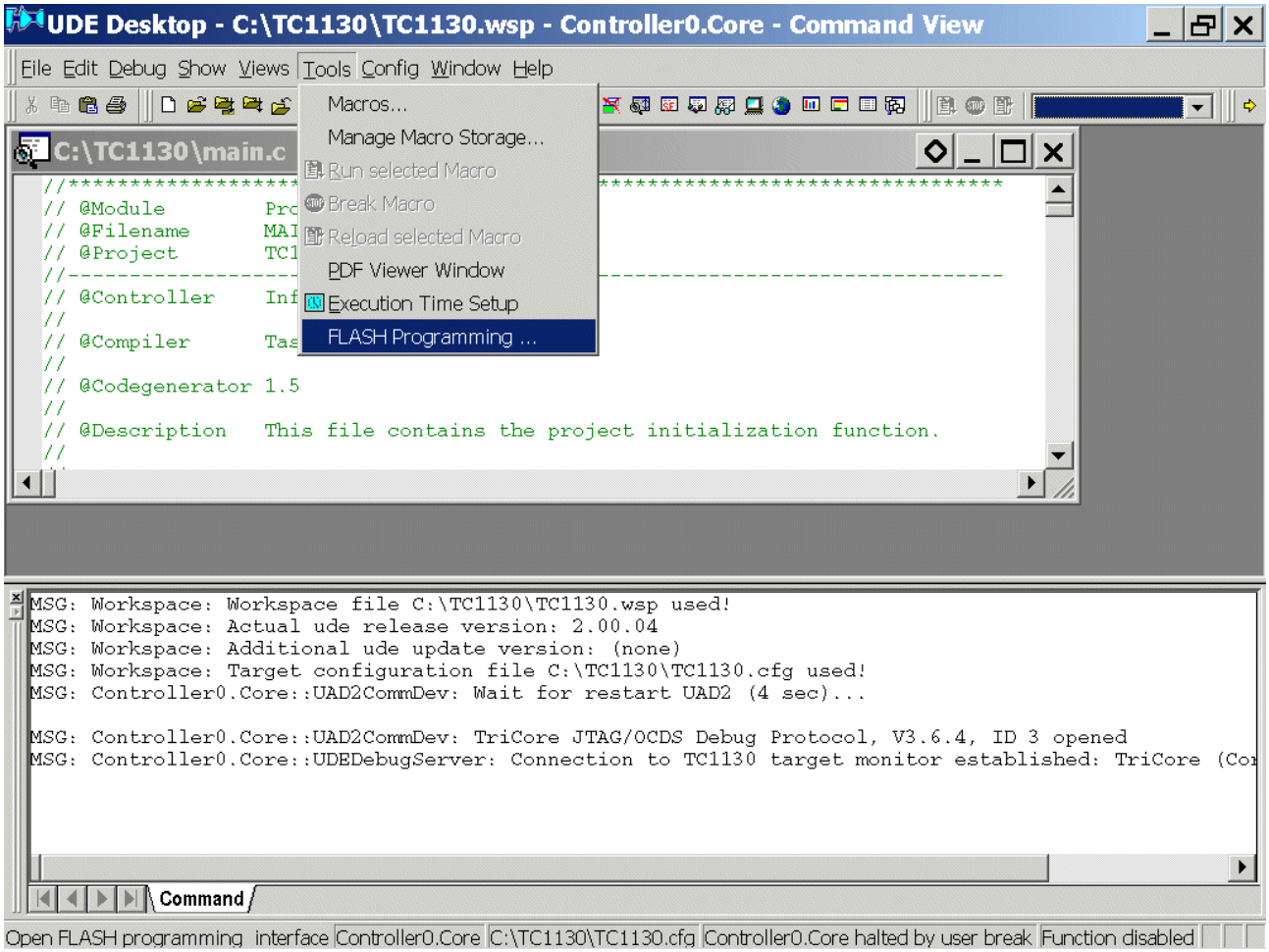


Click ✓ UDE FLASH/OTP Memory Programming Tool

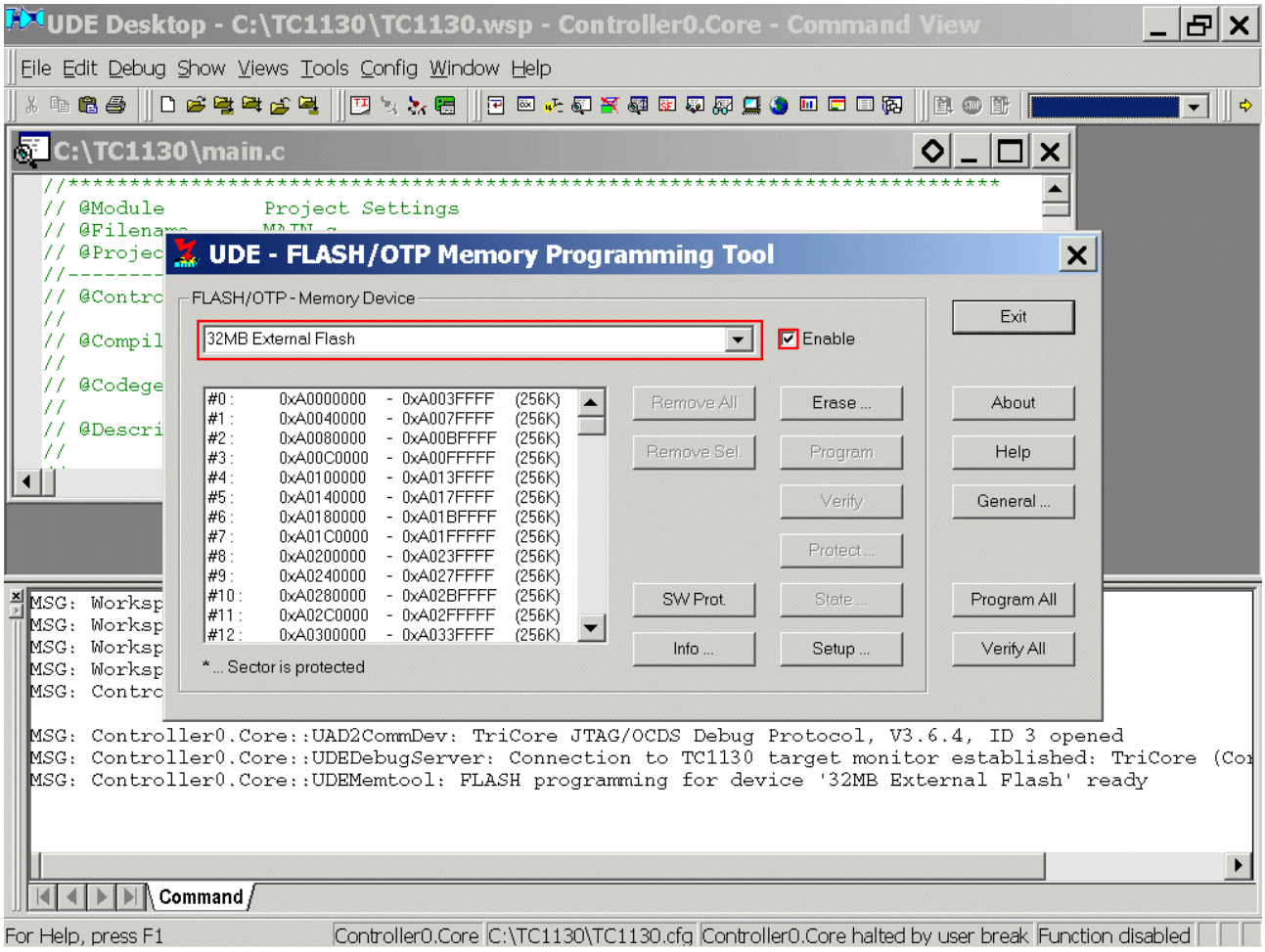


OK

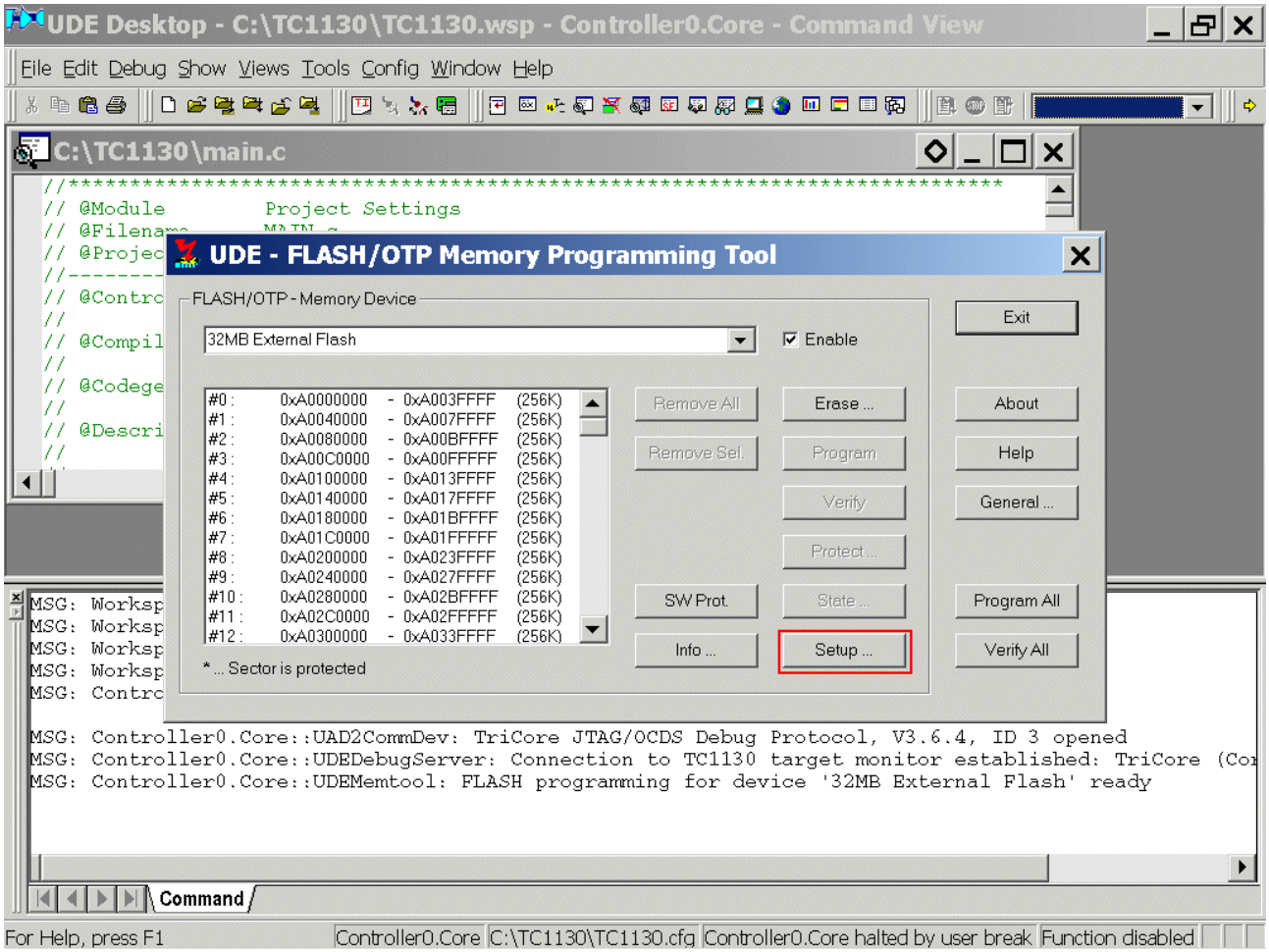
Tools – FLASH Programming …



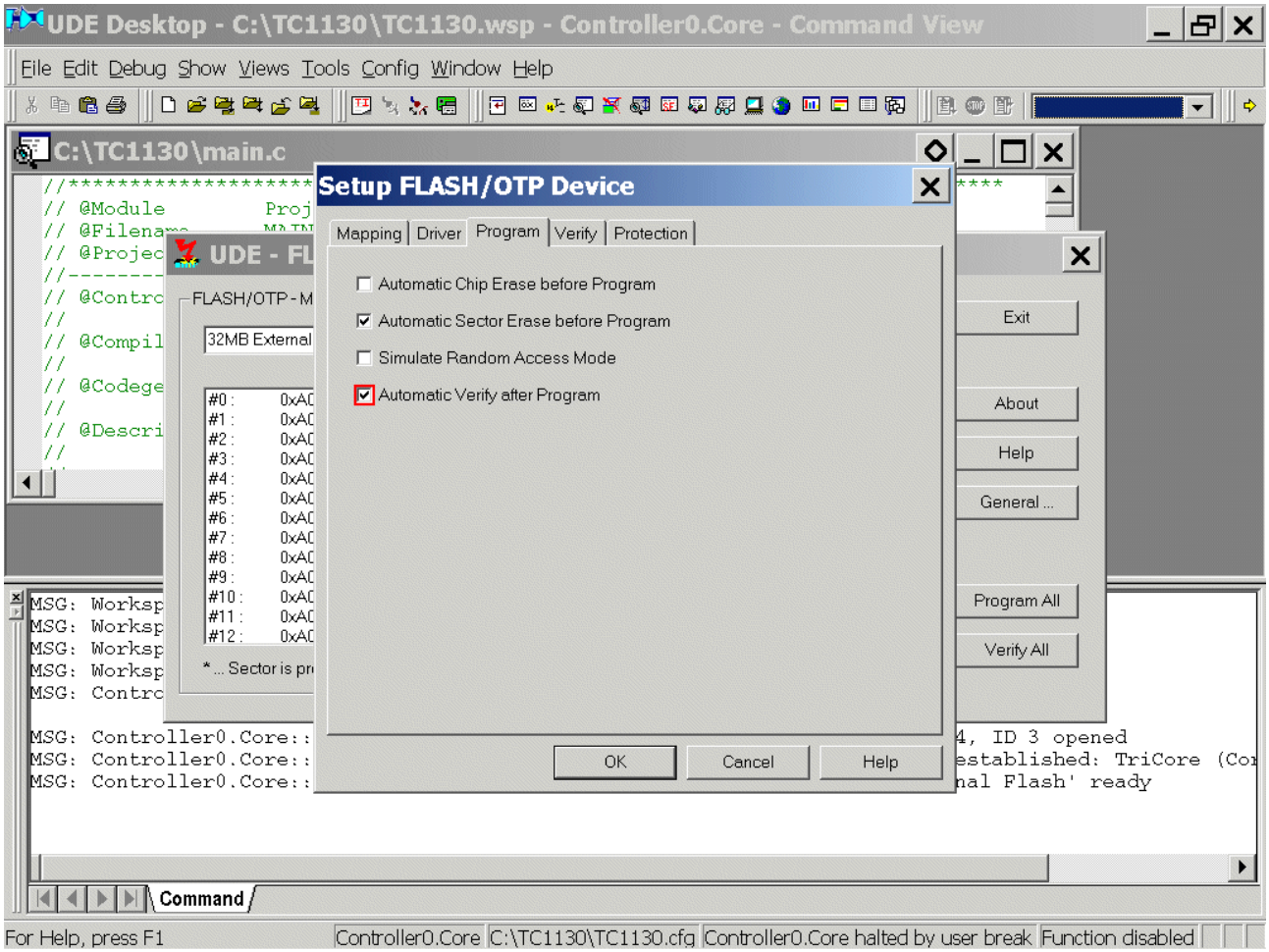
FLASH/OTP – Memory Device: select 32MB External Flash (not ready)
FLASH/OTP – Memory Device: click ✓ Enable



click Setup …

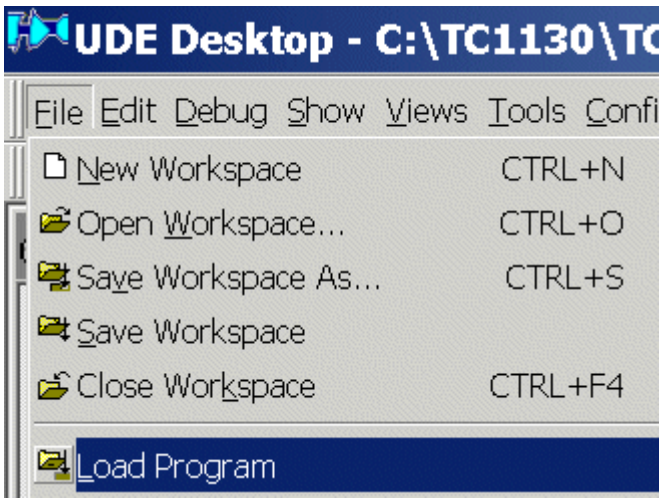

UDE Desktop - C:\TC1130\TC1130.wsp - Controller0.Core - Command View
File Edit Debug Show Views Tools Config Window Help
C:\TC1130\main.c
UDE - FLASH/OTP Memory Programming Tool
FLASH/OTP - Memory Device
32MB External Flash
Enable
Exit
#0 : 0xA0000000 - 0xA003FFFF (256K)
#1 : 0xA0040000 - 0xA007FFFF (256K)
#2 : 0xA0080000 - 0xA00BFFFF (256K)
#3 : 0xA00C0000 - 0xA00FFFFF (256K)
#4 : 0xA0100000 - 0xA013FFFF (256K)
#5 : 0xA0140000 - 0xA017FFFF (256K)
#6 : 0xA0180000 - 0xA01BFFFF (256K)
#7 : 0xA01C0000 - 0xA01FFFFF (256K)
#8 : 0xA0200000 - 0xA023FFFF (256K)
#9 : 0xA0240000 - 0xA027FFFF (256K)
#10 : 0xA0280000 - 0xA02BFFFF (256K)
#11 : 0xA02C0000 - 0xA02FFFFF (256K)
#12 : 0xA0300000 - 0xA033FFFF (256K)
* ... Sector is protected
Remove All
Erase ...
About
Remove Sel.
Program
Help
Verify
General ...
Protect ...
SW Prot.
State ...
Program All
Info ...
Setup ...
Verify All
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
MSG: Controller0.Core::UDEDebugServer: Connection to TC1130 target monitor established: TriCore (Co
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '32MB External Flash' ready
Command
For Help, press F1
Controller0.Core
C:\TC1130\TC1130.cfg
Controller0.Core halted by user break
Function disabled

Program: click ✓ Automatic Verify after Program

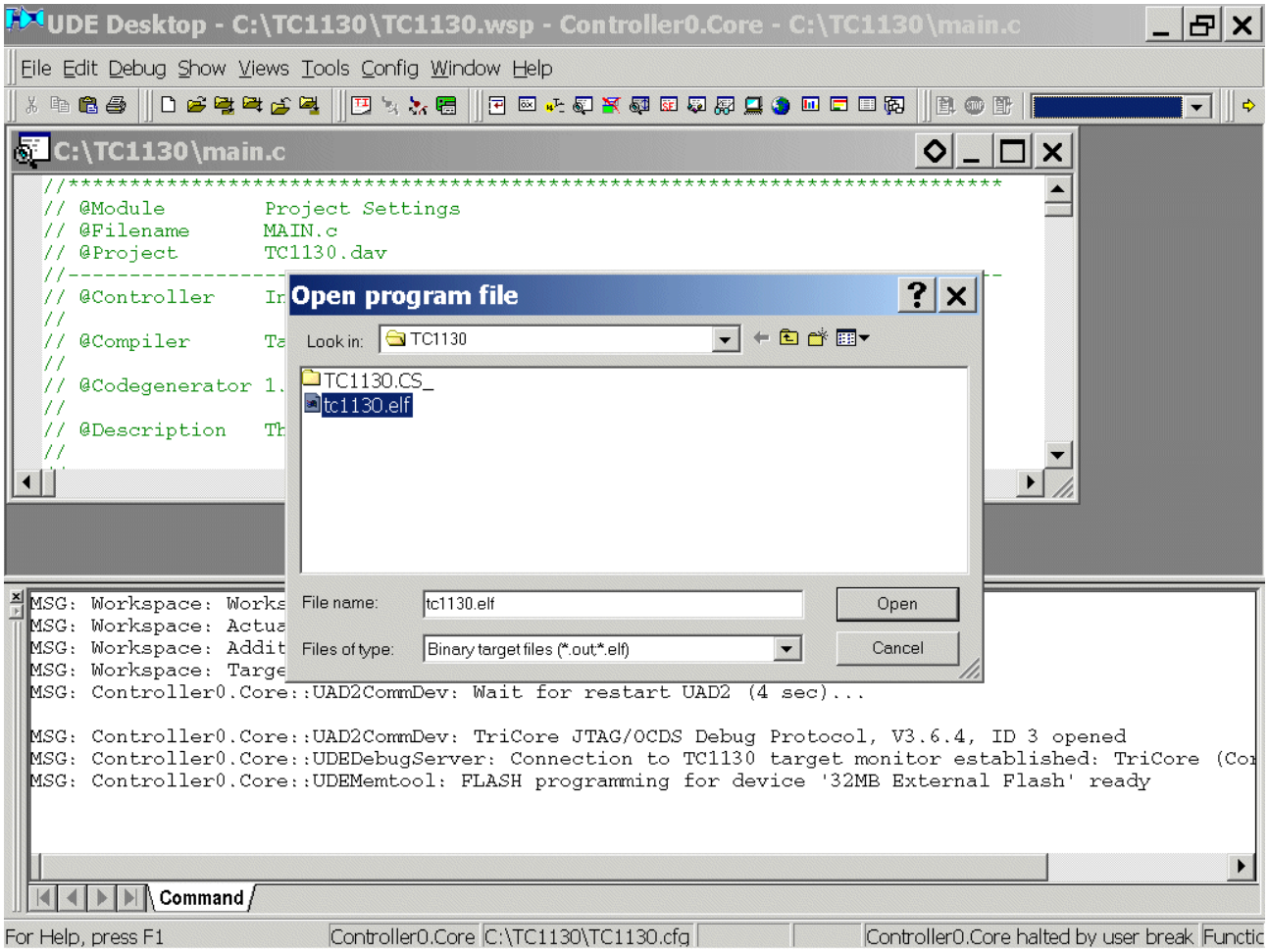


OK
Exit

File – Load Program

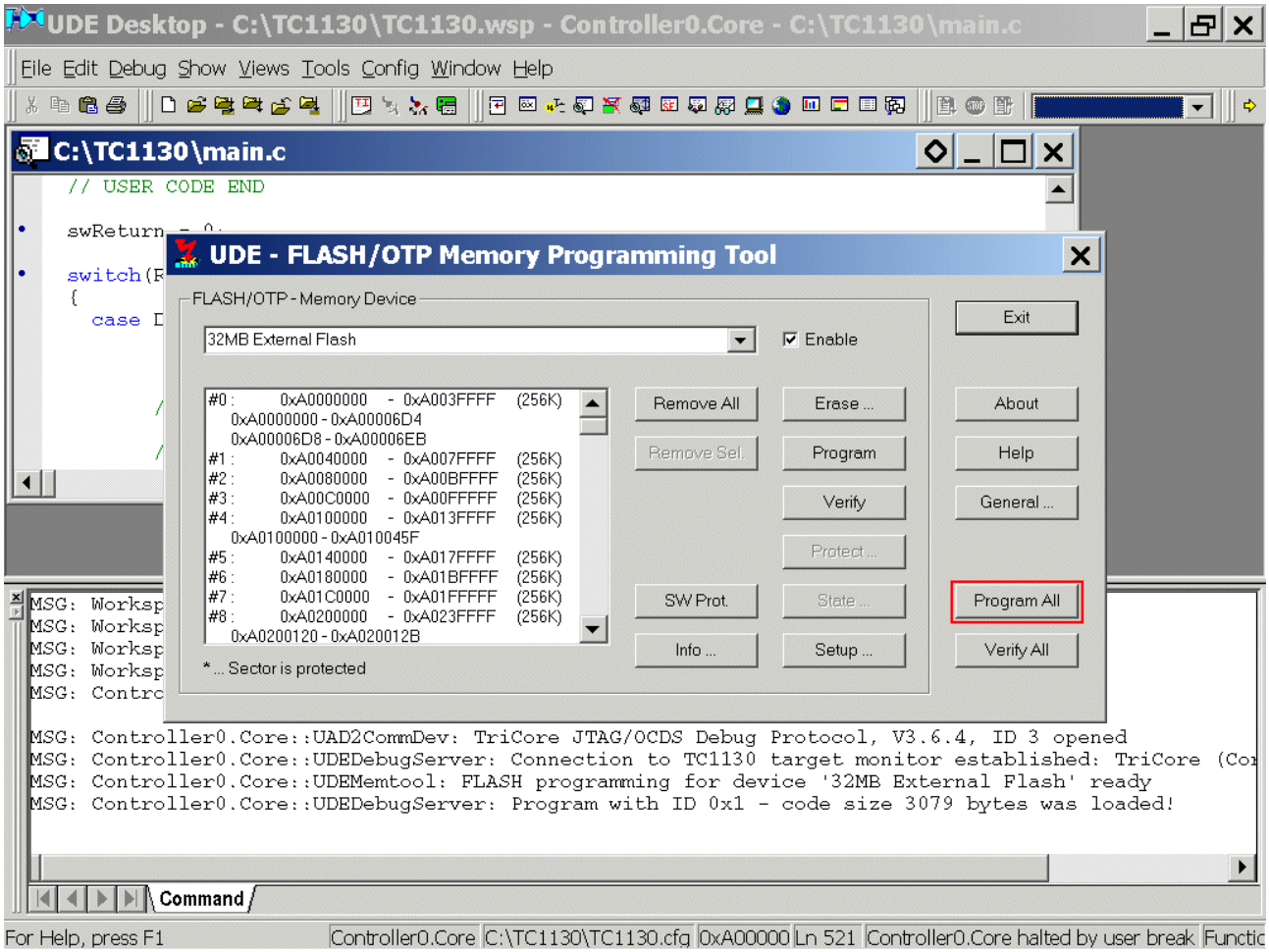


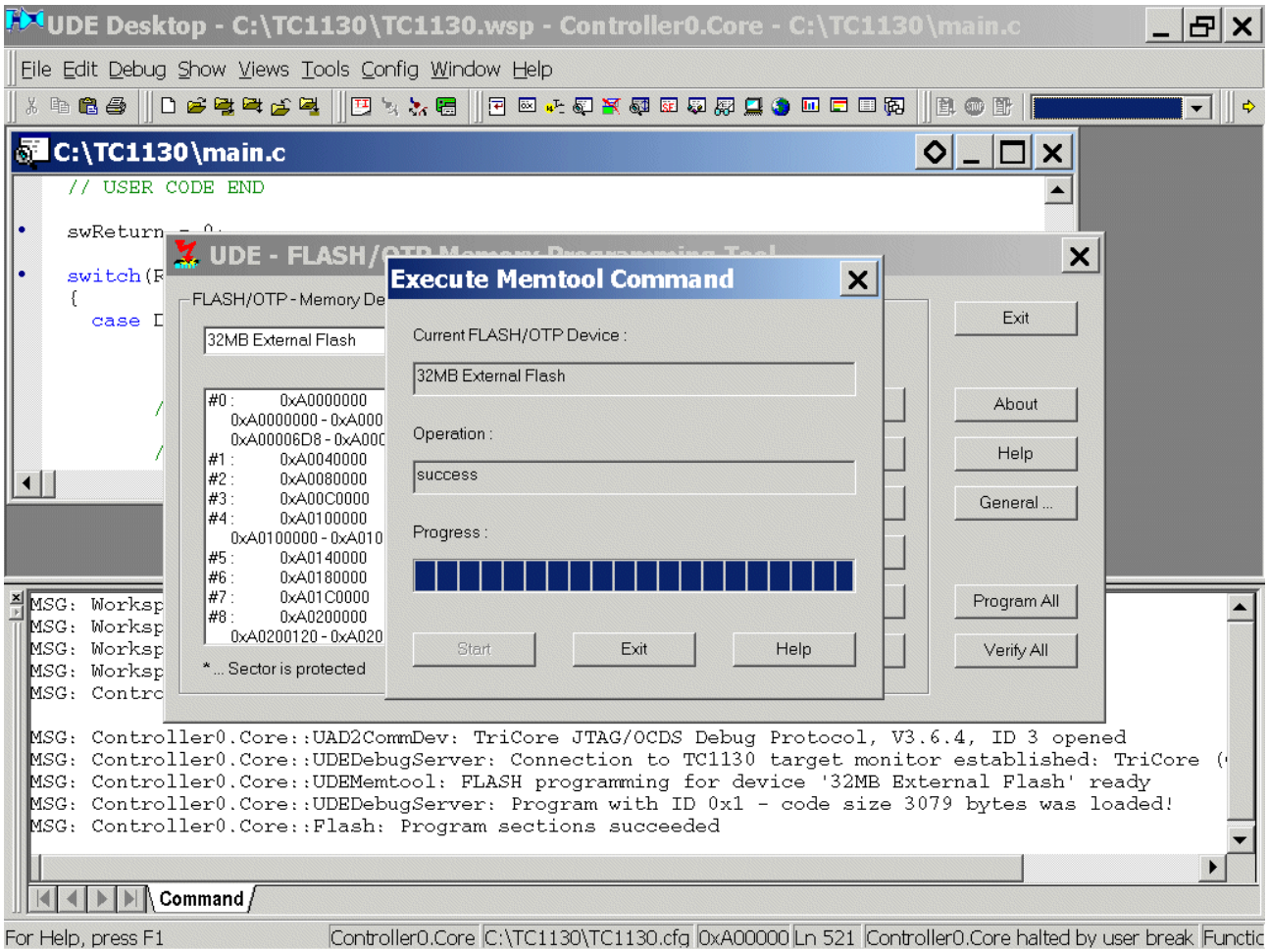
Look in: select TC1130
File name: select TC1130.elf



Open

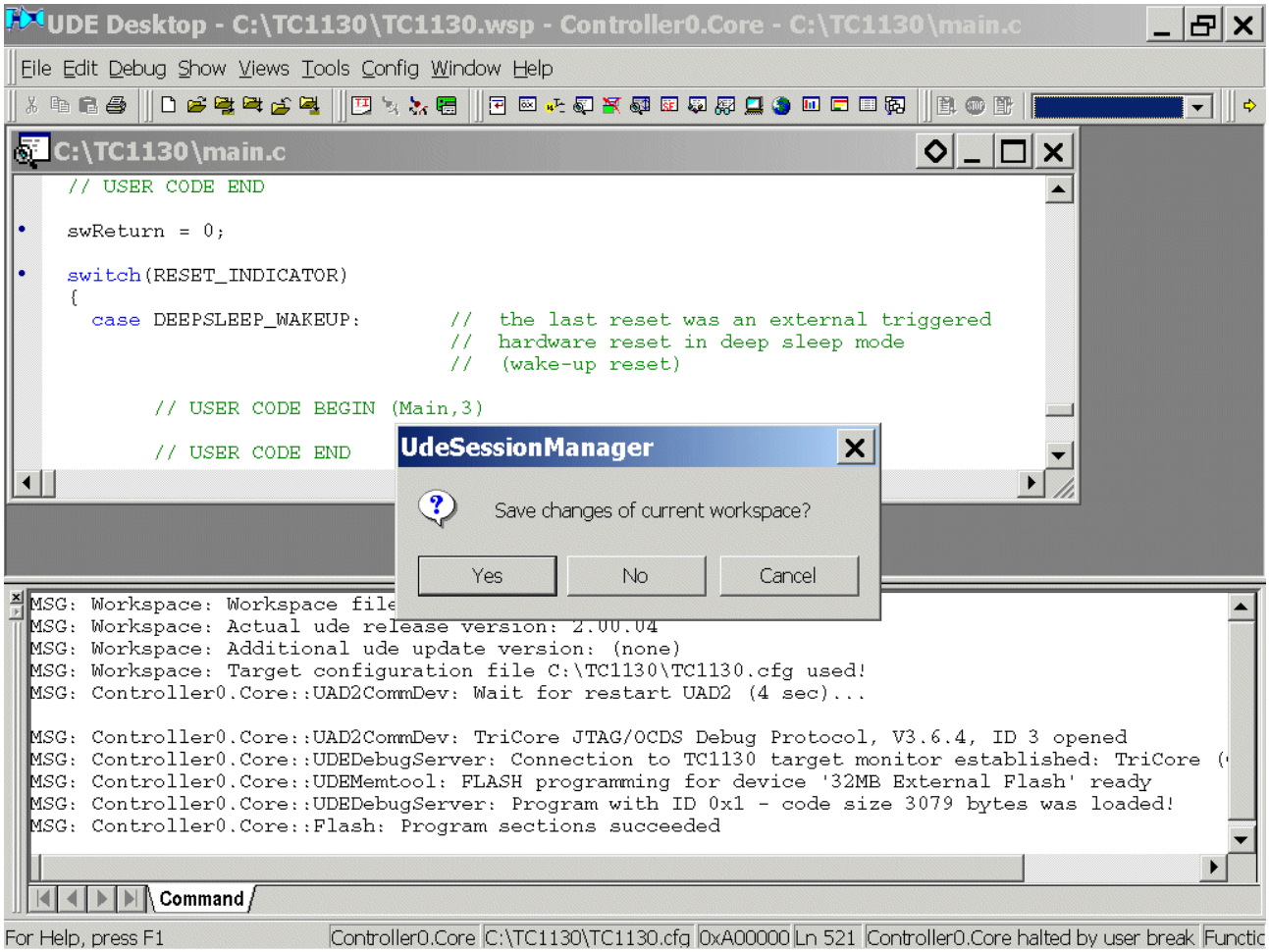
Click Program All




UDE Desktop - C:\TC1130\TC1130.wsp - Controller0.Core - C:\TC1130\main.c
File Edit Debug Show Views Tools Config Window Help
C:\TC1130\main.c
// USER CODE END
swReturn = 0;
switch(R
{
case D
UDE - FLASH/OTP Memory Programming Tool
FLASH/OTP - Memory De
32MB External Flash
#0 : 0xA0000000
0xA0000000 - 0xA000
0xA00006D8 - 0xA00
#1 : 0xA0040000
#2 : 0xA0080000
#3 : 0xA00C0000
#4 : 0xA0100000
0xA0100000 - 0xA010
#5 : 0xA0140000
#6 : 0xA0180000
#7 : 0xA01C0000
#8 : 0xA0200000
0xA0200120 - 0xA020
* ... Sector is protected
Execute Memtool Command
Current FLASH/OTP Device :
32MB External Flash
Operation :
success
Progress :
Start
Exit
Help
Exit
About
Help
General ...
Program All
Verify All
MSG: Worksp
MSG: Worksp
MSG: Worksp
MSG: Worksp
MSG: Contro
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
MSG: Controller0.Core::UDEDebugServer: Connection to TC1130 target monitor established: TriCore (
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '32MB External Flash' ready
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3079 bytes was loaded!
MSG: Controller0.Core::Flash: Program sections succeeded
Command
For Help, press F1
Controller0.Core
C:\TC1130\TC1130.cfg
0xA00000
Ln 521
Controller0.Core halted by user break
Functic

Exit
Exit

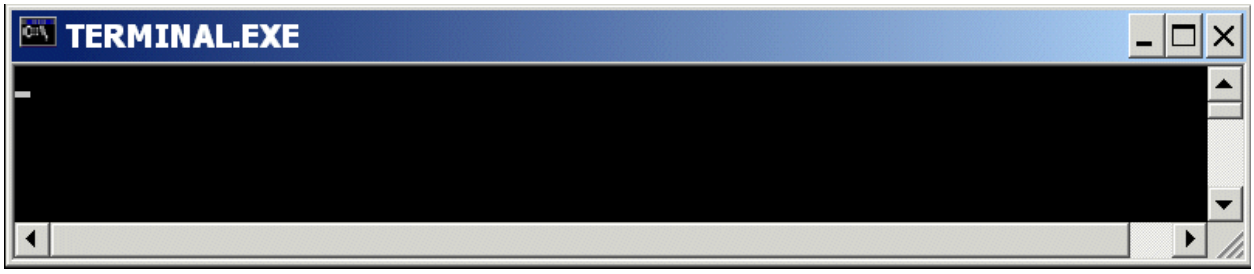
File – Close Workspace



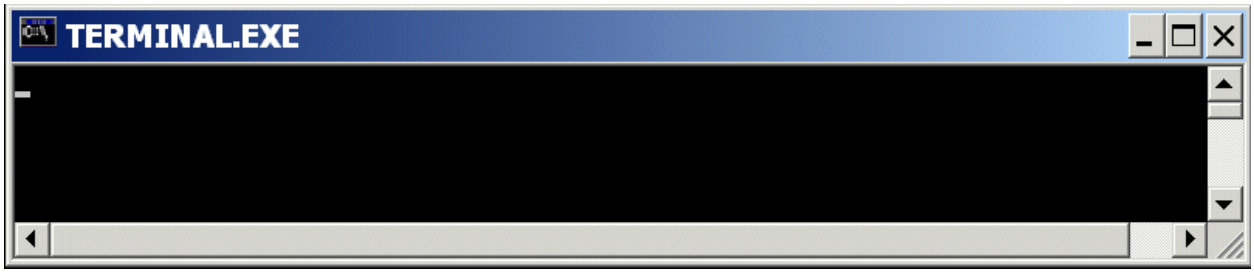
Yes

File – Exit

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

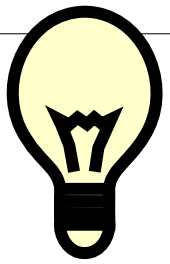


Power-ON the Board and see the result:



Conclusion:

The application runs with the Debugger!
The application does NOT run after “Power-ON” !!!



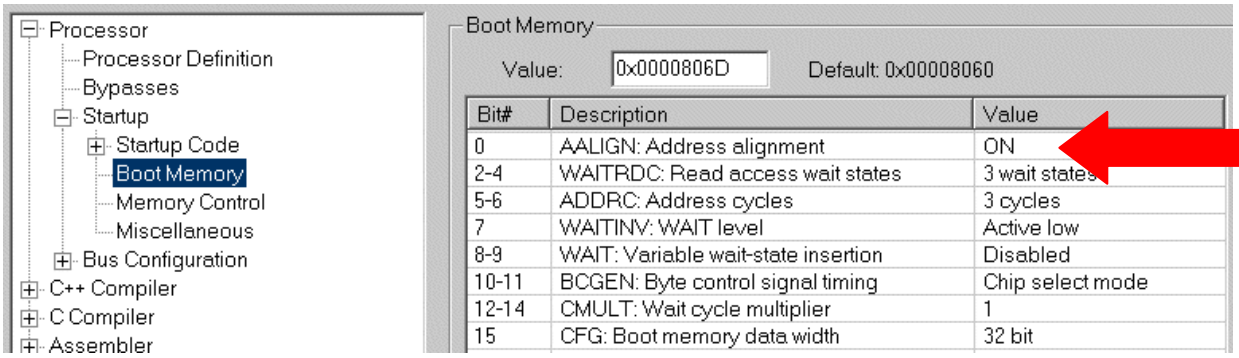
Troubleshooting:

The application runs with the Debugger!
The application does NOT run after “Power-ON” !!!

➔ The only difference between “Debugger” and “Power-ON” program-execution is that the debugger does NOT read the “external-boot-memory-configuration-word”.
Therefore we take a closer look at this item:

External Boot Memory Configuration Word:
If external boot is selected, the EBU will perform (exactly) one external bus-read-access to a specific address (0x000004) of the memory device attached to CS0.
The result of this read-access is used during “Power-ON” to configure the EBU.
To see the configuration value click here.

➔ Because of the 32-bit-alignment we need the “external-boot-memory-configuration-word” at address 0xA0000010.



EBU Boot Configuration Value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES32															

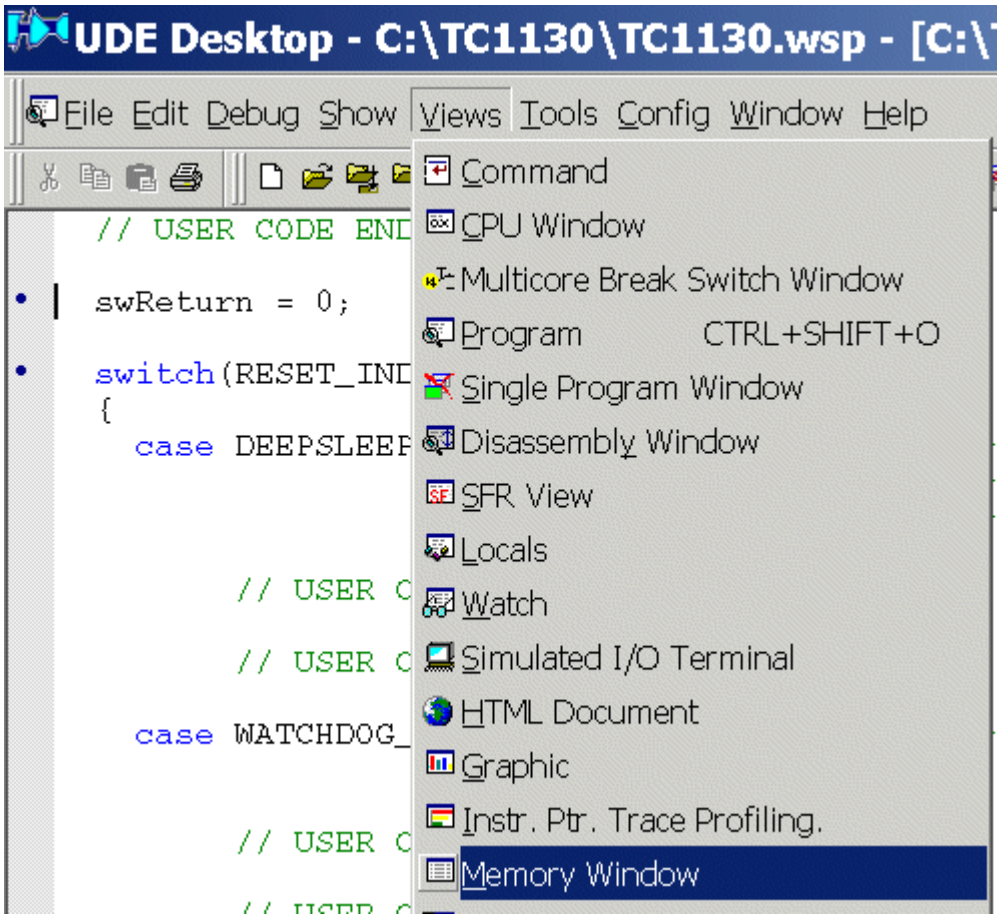
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONF32BIT	CMULT			BCGEN		WAIT		WAITINV	ADDRC		WAITRDC			RES	AALIGN

Field	Bits	Description
AALIGN	0	Address Alignment Loaded into BUSCON0 AALIGN (see Section 14.12.3).

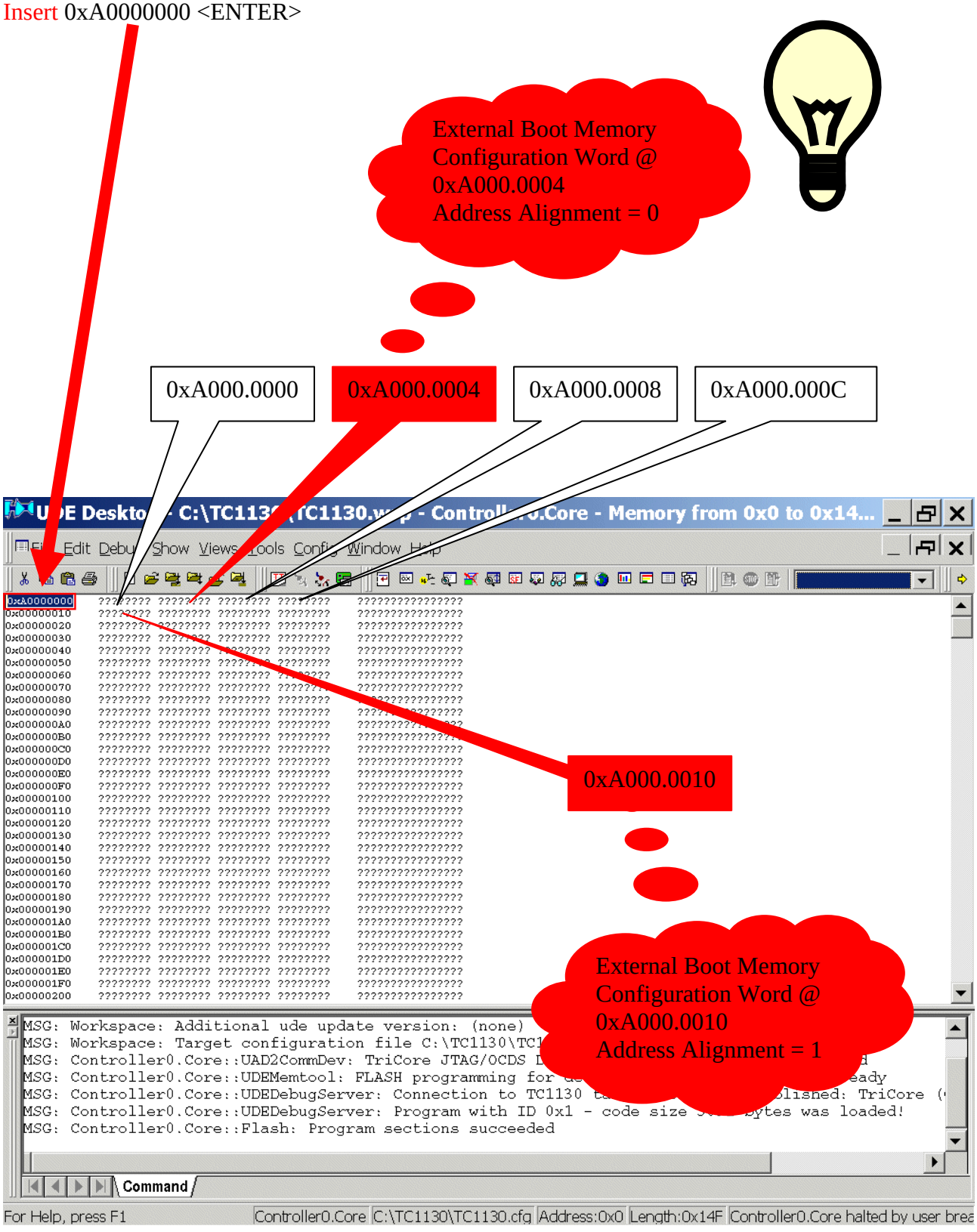
“external-boot-memory-configuration-word” at address 0xA0000010
(0xA0000004 << 2 = 0xA0000010 , due to 32 bit alignment)

To check this we use the pls-debugger:

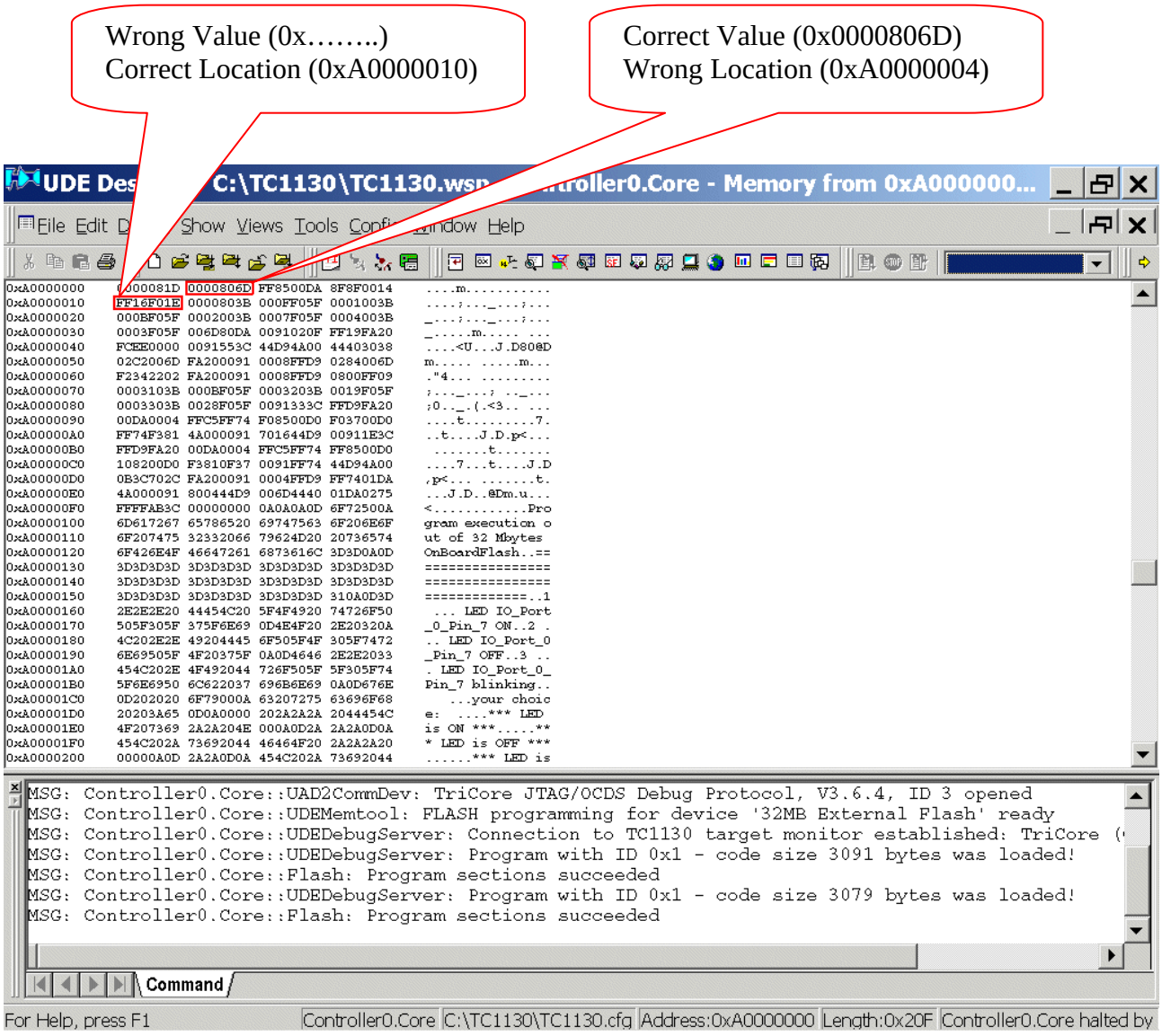
Views – Memory Window



Insert 0xA0000000 <ENTER>



As you can see:
The “external-boot-memory-configuration-word” value (0x0000806D) is NOT @ 0xA0000010 - it is still @ 0xA0000004



➔ We consider this as a Compiler/EDE-Error and therefore we change the cstart.asm file!

Start Tasking EDE and open the project:

File – Open Project Space

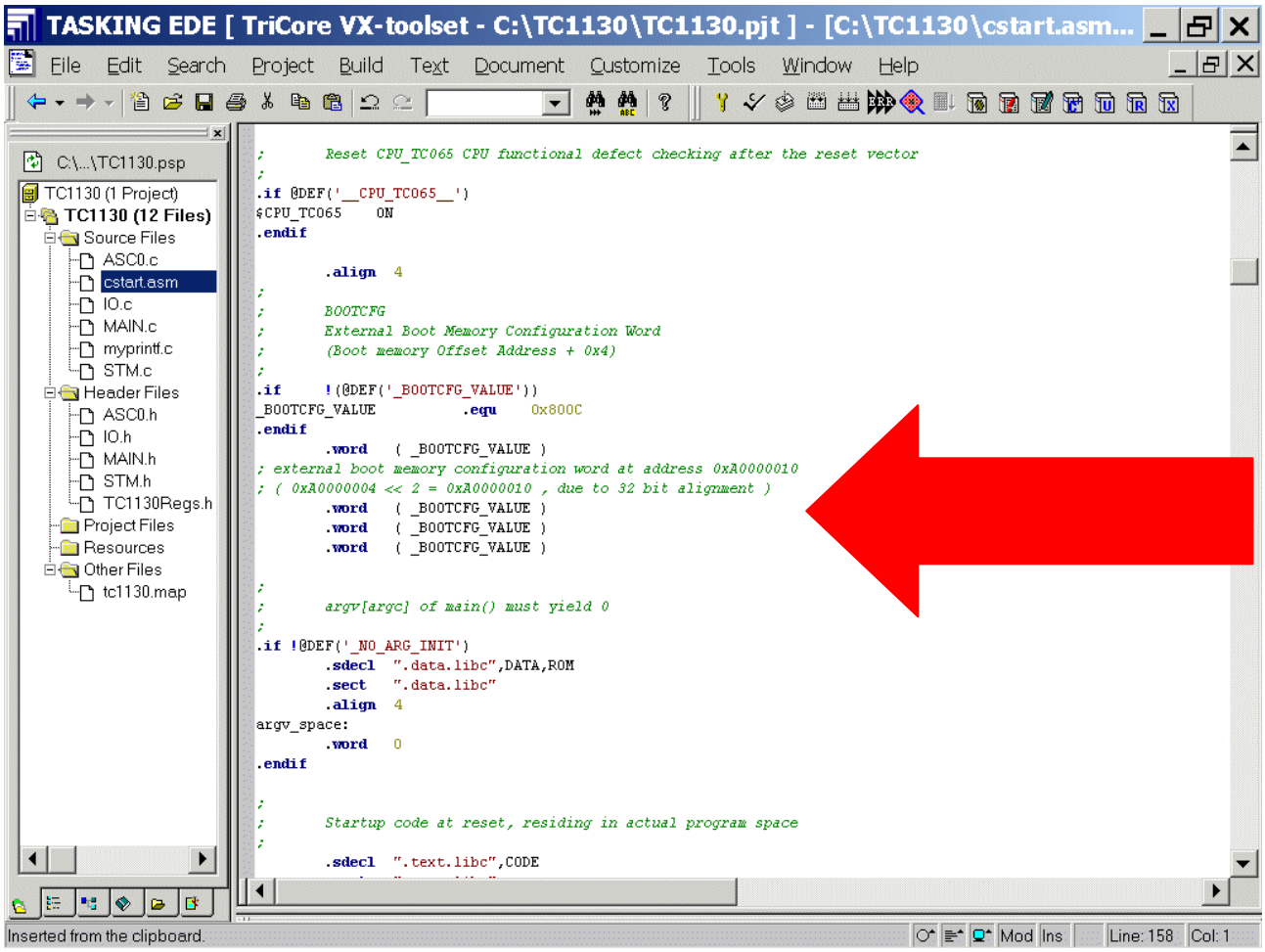
Look in: select C:\TC1130
File name: select TC1130.psp

Open

Insert your application specific program:

Double click: cstart.asm and insert code:

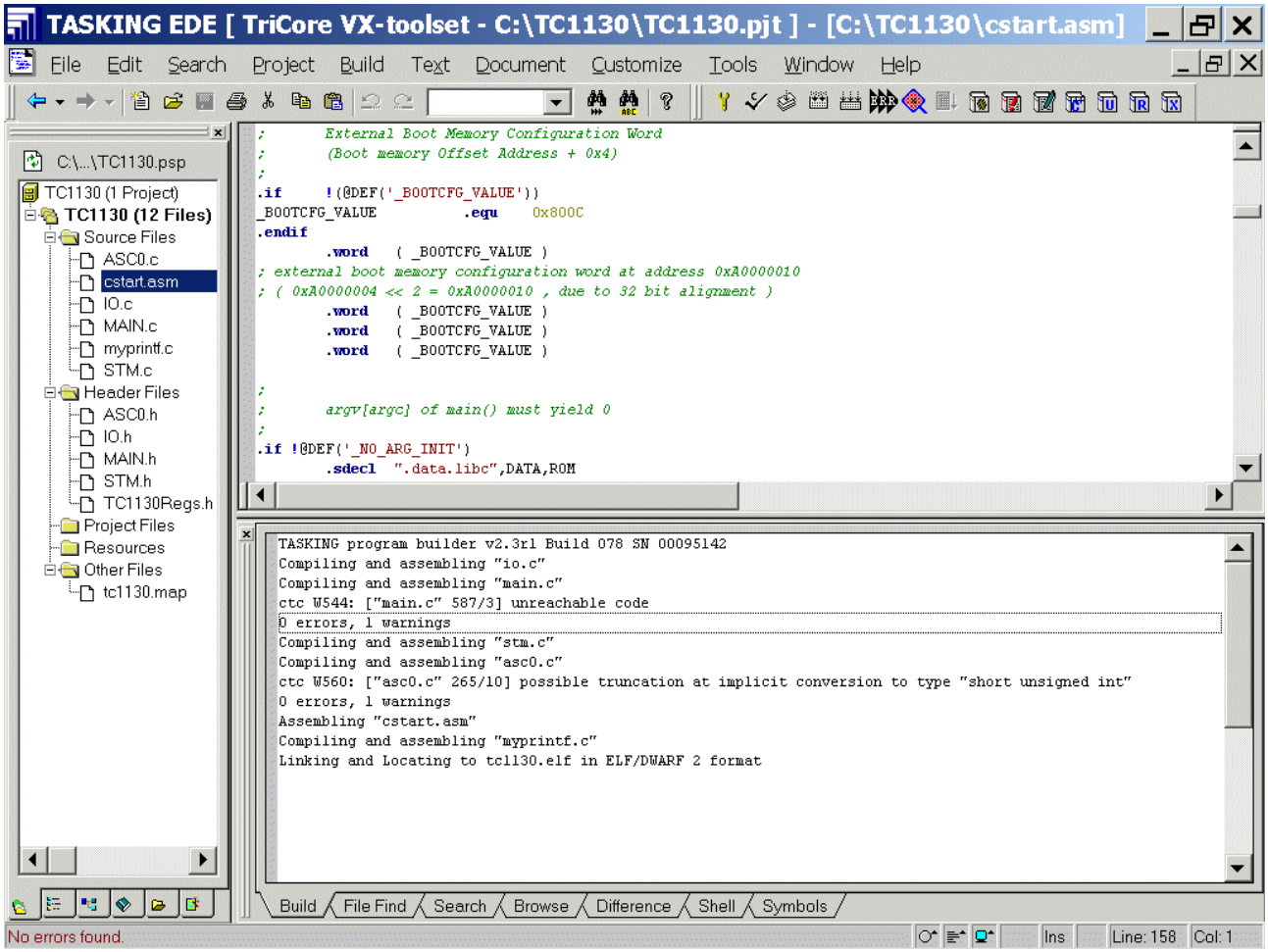
```
; external boot memory configuration word at address 0xA0000010
; ( 0xA0000004 << 2 = 0xA0000010 , due to 32 bit alignment )
        .word   ( _BOOTCFG_VALUE )
        .word   ( _BOOTCFG_VALUE )
        .word   ( _BOOTCFG_VALUE )
```



Generate your application program:

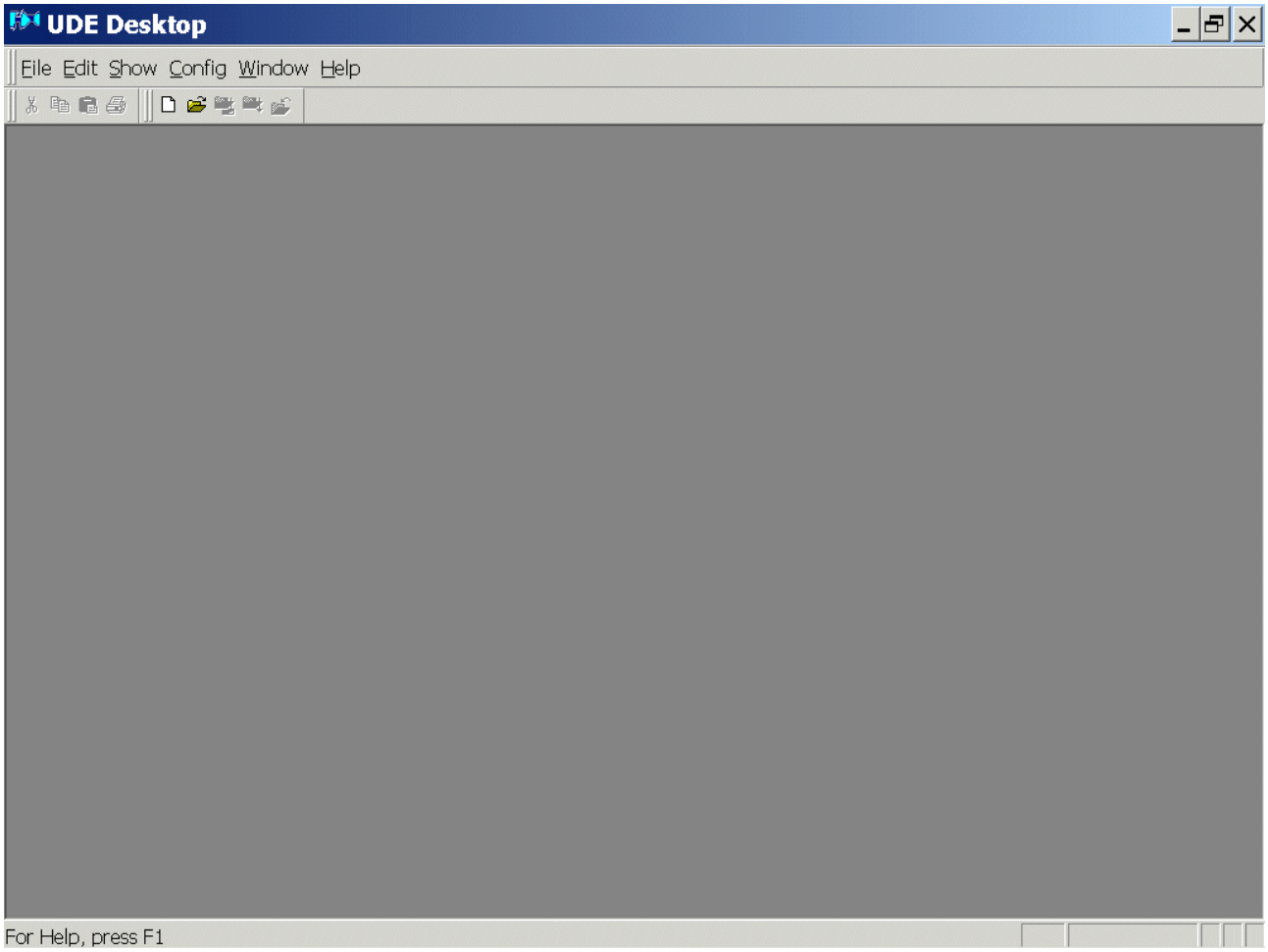
Build
Rebuild

or



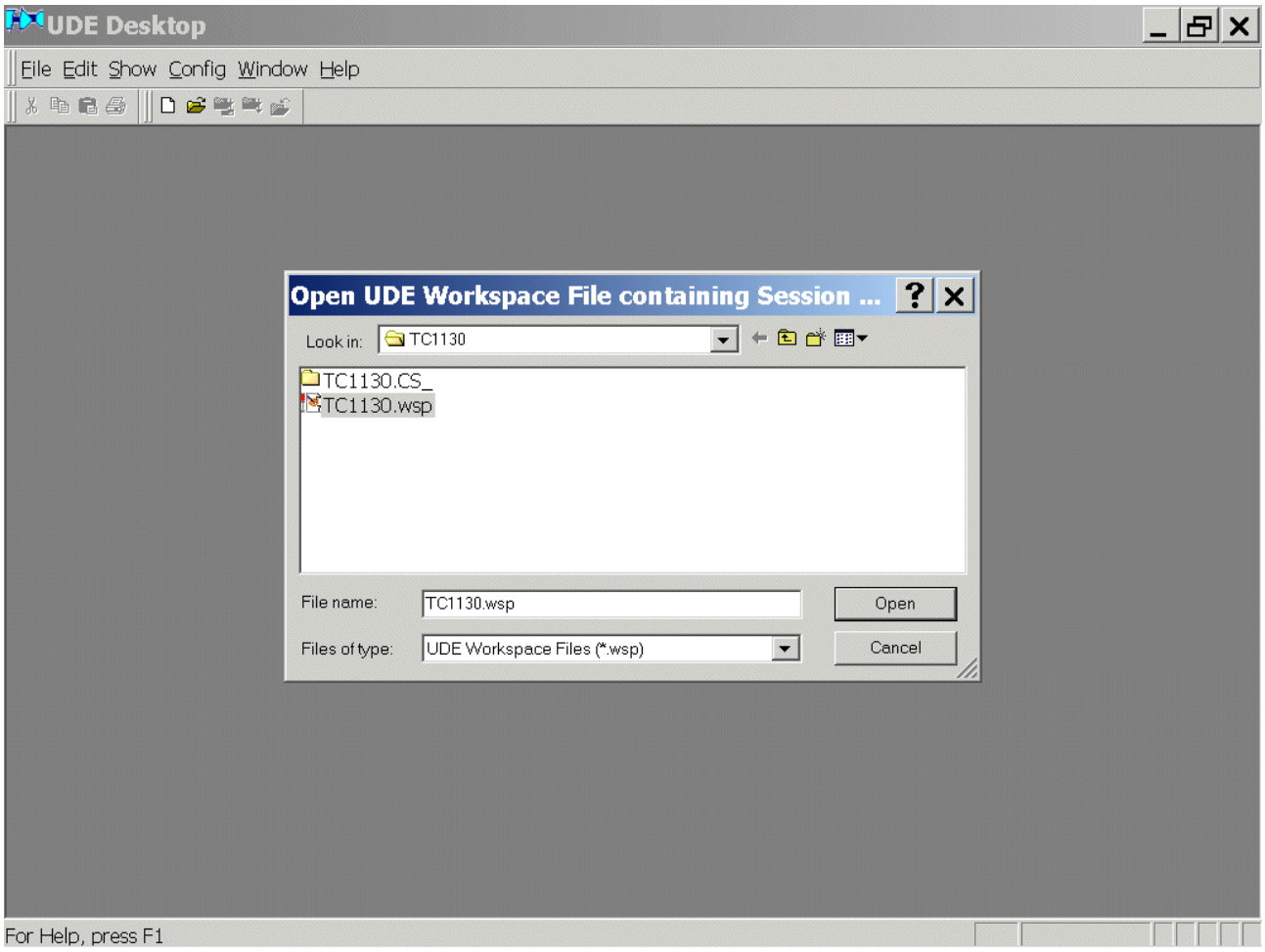
Programming and Troubleshooting is now complete. You can now load and run your program:

Start pls-Debugger

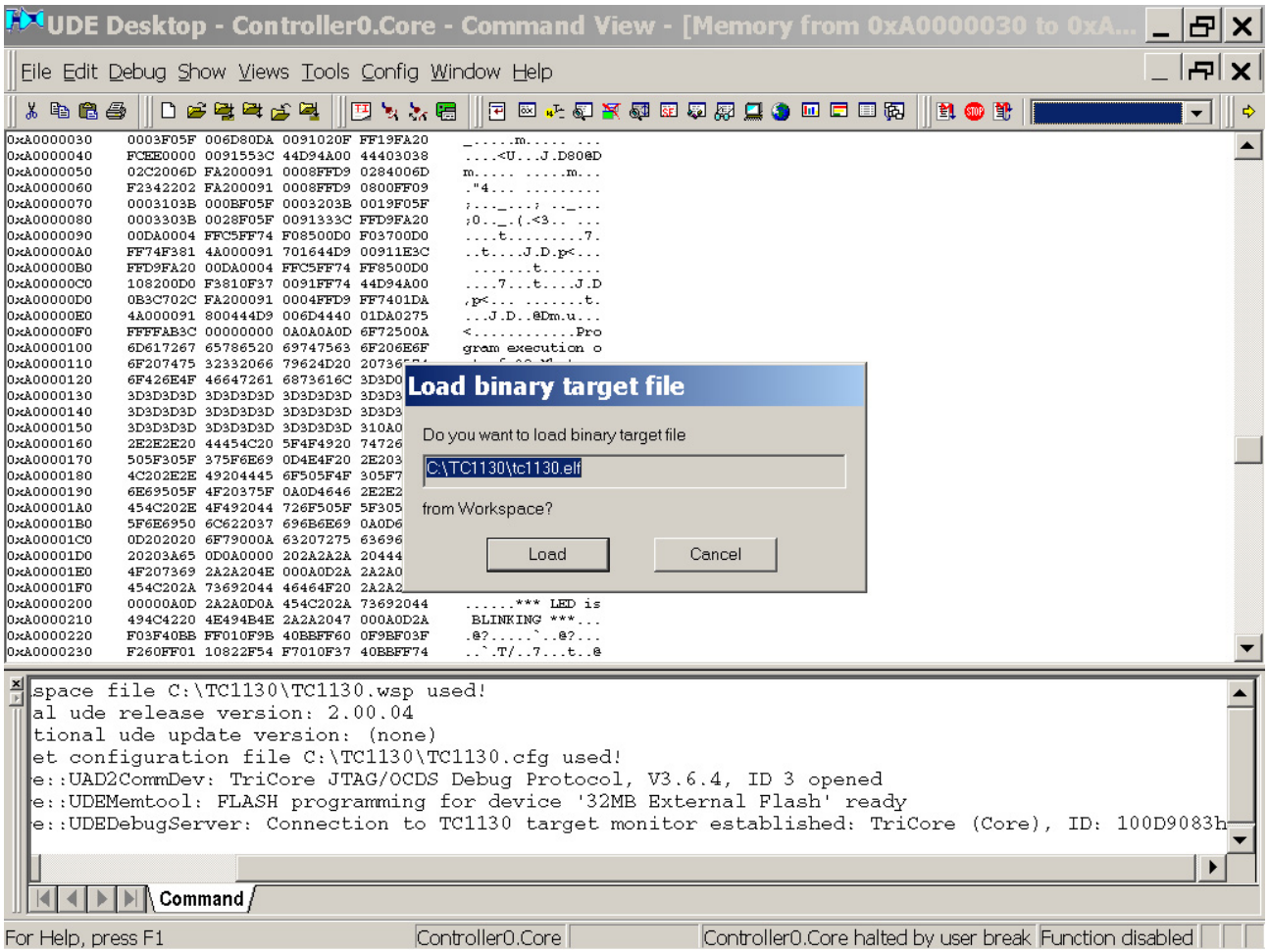


File – Open Workspace

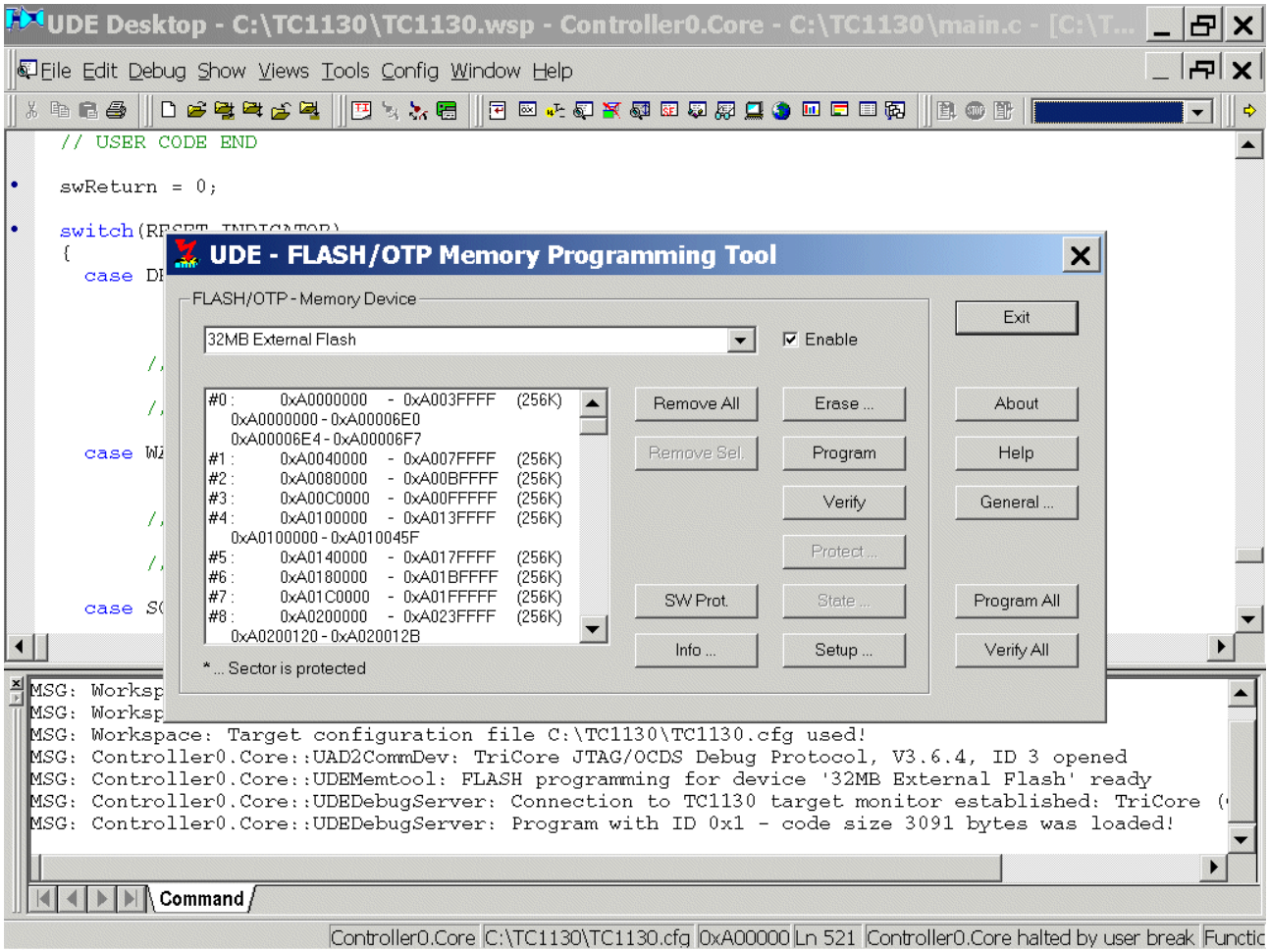
Look in: select C:\TC1130
File name: select TC1130.wsp



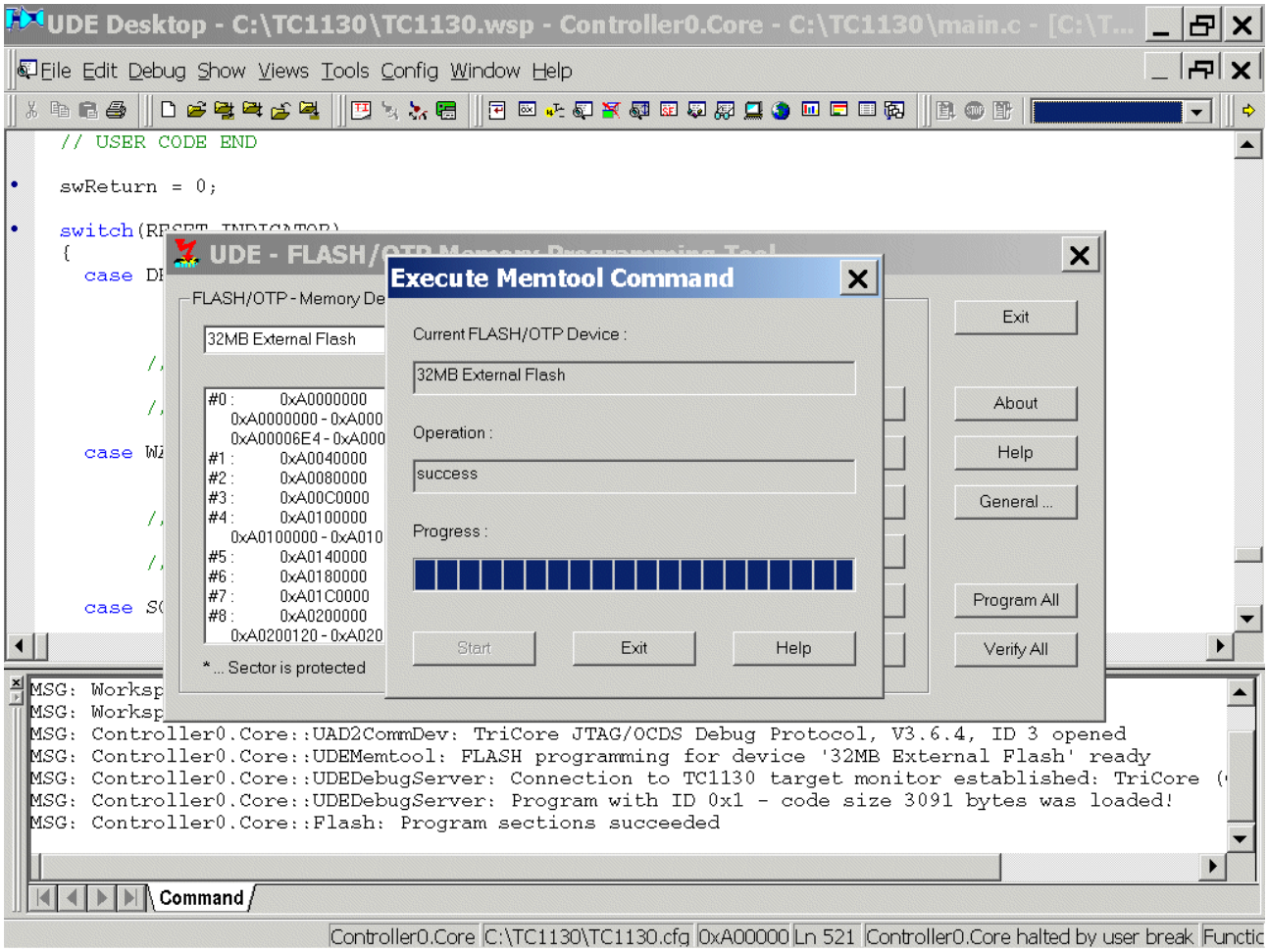
Open


UDE Desktop - Controller0.Core - Command View - [Memory from 0xA0000030 to 0xA...
File Edit Debug Show Views Tools Config Window Help
Load binary target file
Do you want to load binary target file
C:\TC1130\tc1130.elf
from Workspace?
Load
Cancel
space file C:\TC1130\TC1130.wsp used!
al ude release version: 2.00.04
tional ude update version: (none)
et configuration file C:\TC1130\TC1130.cfg used!
e::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
e::UDEMemtool: FLASH programming for device '32MB External Flash' ready
e::UDEDebugServer: Connection to TC1130 target monitor established: TriCore (Core), ID: 100D9083h
Command
For Help, press F1
Controller0.Core
Controller0.Core halted by user break
Function disabled

Load


UDE Desktop - C:\TC1130\TC1130.wsp - Controller0.Core - C:\TC1130\main.c - [C:\T...
File Edit Debug Show Views Tools Config Window Help
// USER CODE END
swReturn = 0;
switch(RESET_INDICATOR)
UDE - FLASH/OTP Memory Programming Tool
FLASH/OTP - Memory Device
32MB External Flash
Enable
#0 : 0xA0000000 - 0xA003FFFF (256K)
0xA0000000 - 0xA00006E0
0xA00006E4 - 0xA00006F7
#1 : 0xA0040000 - 0xA007FFFF (256K)
#2 : 0xA0080000 - 0xA00BFFFF (256K)
#3 : 0xA00C0000 - 0xA00FFFFF (256K)
#4 : 0xA0100000 - 0xA013FFFF (256K)
0xA0100000 - 0xA010045F
#5 : 0xA0140000 - 0xA017FFFF (256K)
#6 : 0xA0180000 - 0xA01BFFFF (256K)
#7 : 0xA01C0000 - 0xA01FFFFF (256K)
#8 : 0xA0200000 - 0xA023FFFF (256K)
0xA0200120 - 0xA020012B
* ... Sector is protected
Remove All
Remove Sel.
Erase ...
Program
Verify
Protect ...
SW Prot.
State ...
Info ...
Setup ...
Exit
About
Help
General ...
Program All
Verify All
MSG: Workspace: Target configuration file C:\TC1130\TC1130.cfg used!
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '32MB External Flash' ready
MSG: Controller0.Core::UDEDebugServer: Connection to TC1130 target monitor established: TriCore
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3091 bytes was loaded!
Command
Controller0.Core C:\TC1130\TC1130.cfg 0xA00000 Ln 521 Controller0.Core halted by user break

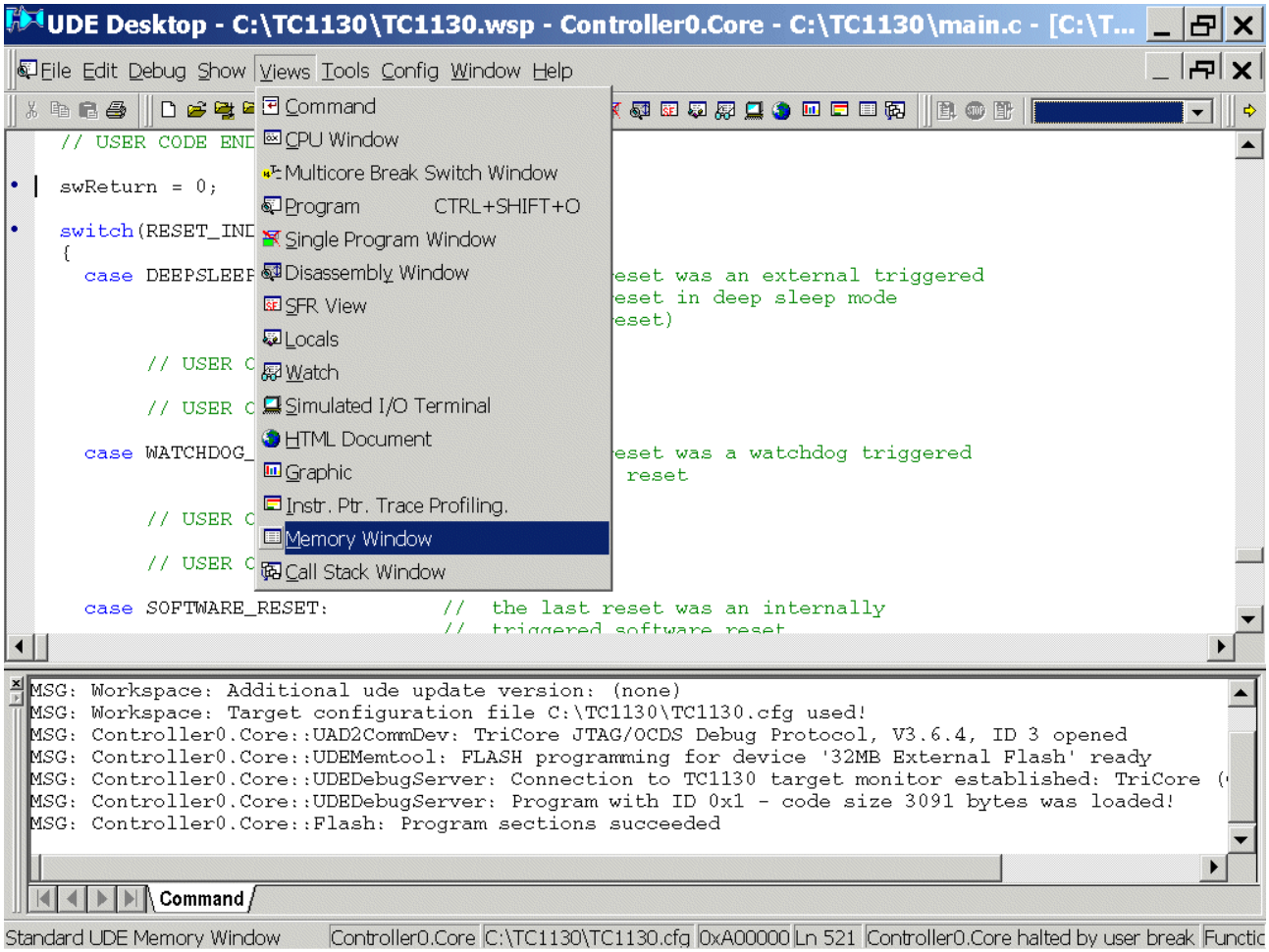
Program All


UDE Desktop - C:\TC1130\TC1130.wsp - Controller0.Core - C:\TC1130\main.c - [C:\T...
File Edit Debug Show Views Tools Config Window Help
// USER CODE END
swReturn = 0;
switch(RESET_INDICATOR)
UDE - FLASH/OTP Memory Programming Tool
FLASH/OTP - Memory De
32MB External Flash
#0 : 0xA0000000
0xA0000000 - 0xA000
0xA00006E4 - 0xA000
#1 : 0xA0040000
#2 : 0xA0080000
#3 : 0xA00C0000
#4 : 0xA0100000
0xA0100000 - 0xA010
#5 : 0xA0140000
#6 : 0xA0180000
#7 : 0xA01C0000
#8 : 0xA0200000
0xA0200120 - 0xA020
* ... Sector is protected
Execute Memtool Command
Current FLASH/OTP Device :
32MB External Flash
Operation :
success
Progress :
Start
Exit
Help
Exit
About
Help
General ...
Program All
Verify All
MSG: Worksp
MSG: Worksp
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '32MB External Flash' ready
MSG: Controller0.Core::UDEDebugServer: Connection to TC1130 target monitor established: TriCore (
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 3091 bytes was loaded!
MSG: Controller0.Core::Flash: Program sections succeeded
Command
Controller0.Core
C:\TC1130\TC1130.cfg
0xA00000
Ln 521
Controller0.Core halted by user break

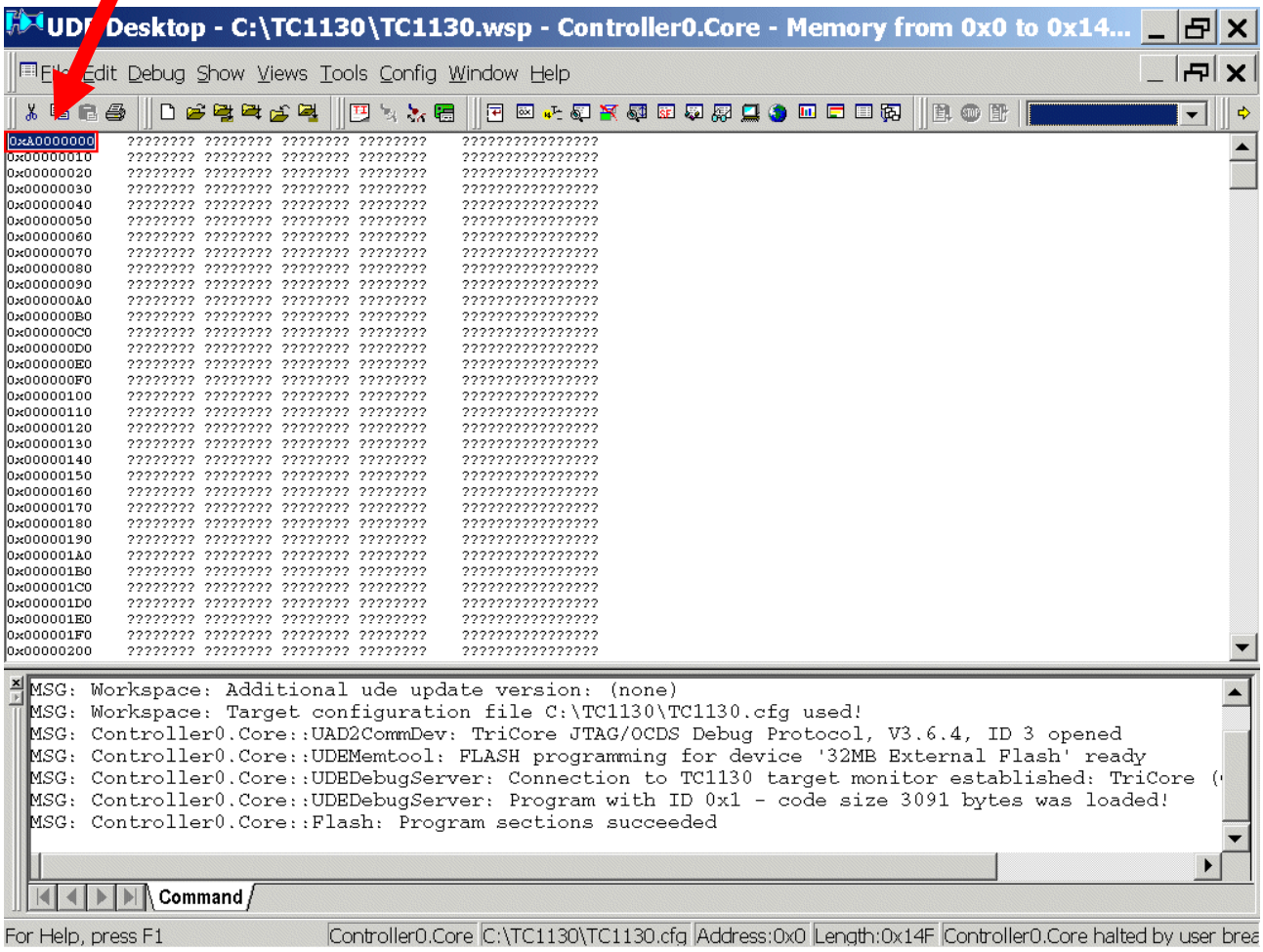
Exit

Exit

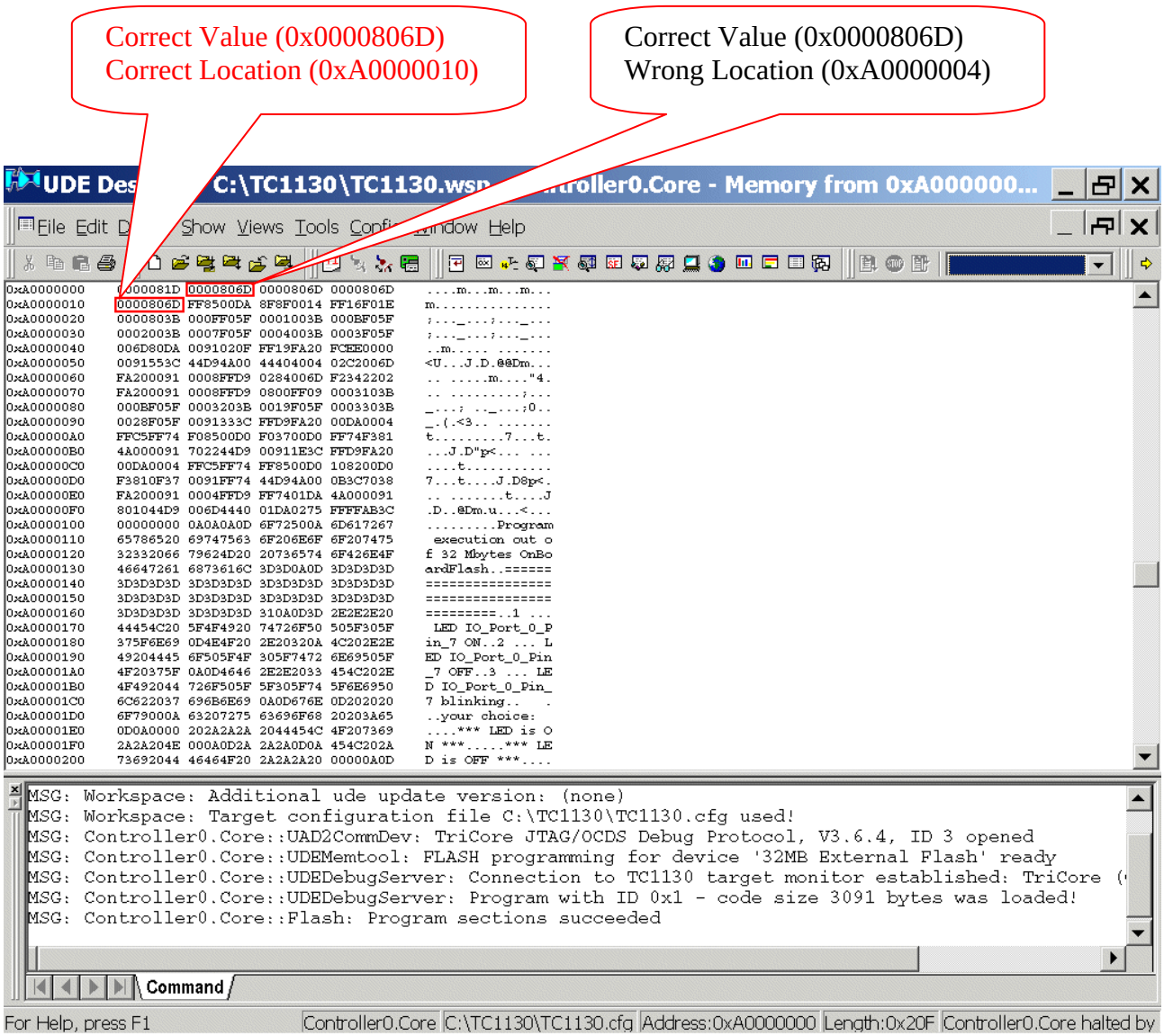
Views – Memory Window



Insert 0xA0000000 <ENTER>

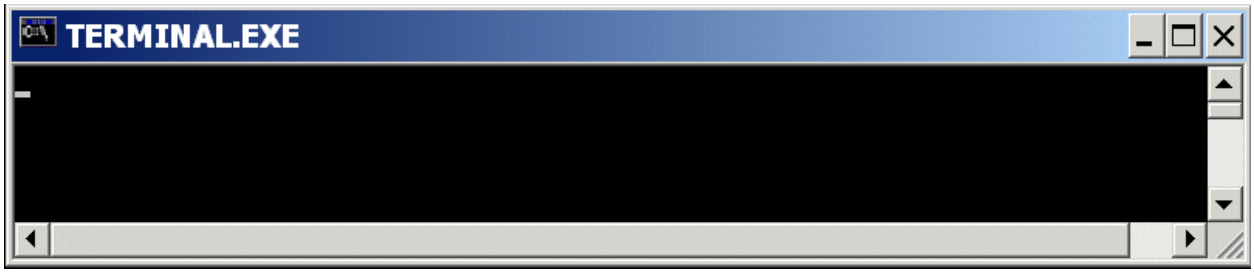


As you can see:
The “external-boot-memory-configuration-word” value (0x0000806D) is now @ 0xA0000010:

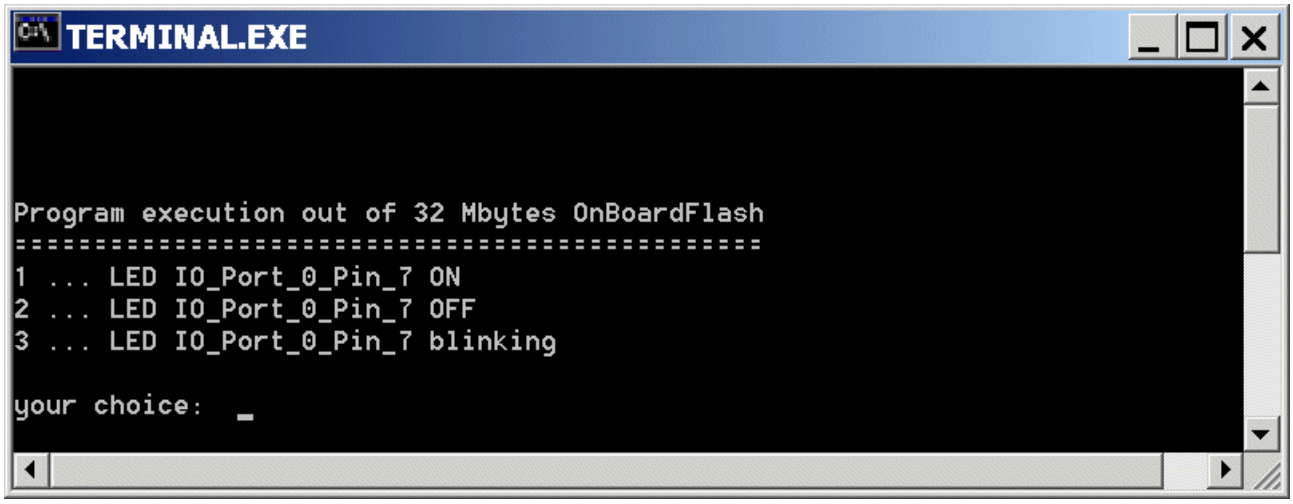


➔ The “external-boot-memory-configuration-word” value (0x0000806D) is now @ 0xA0000010 !

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

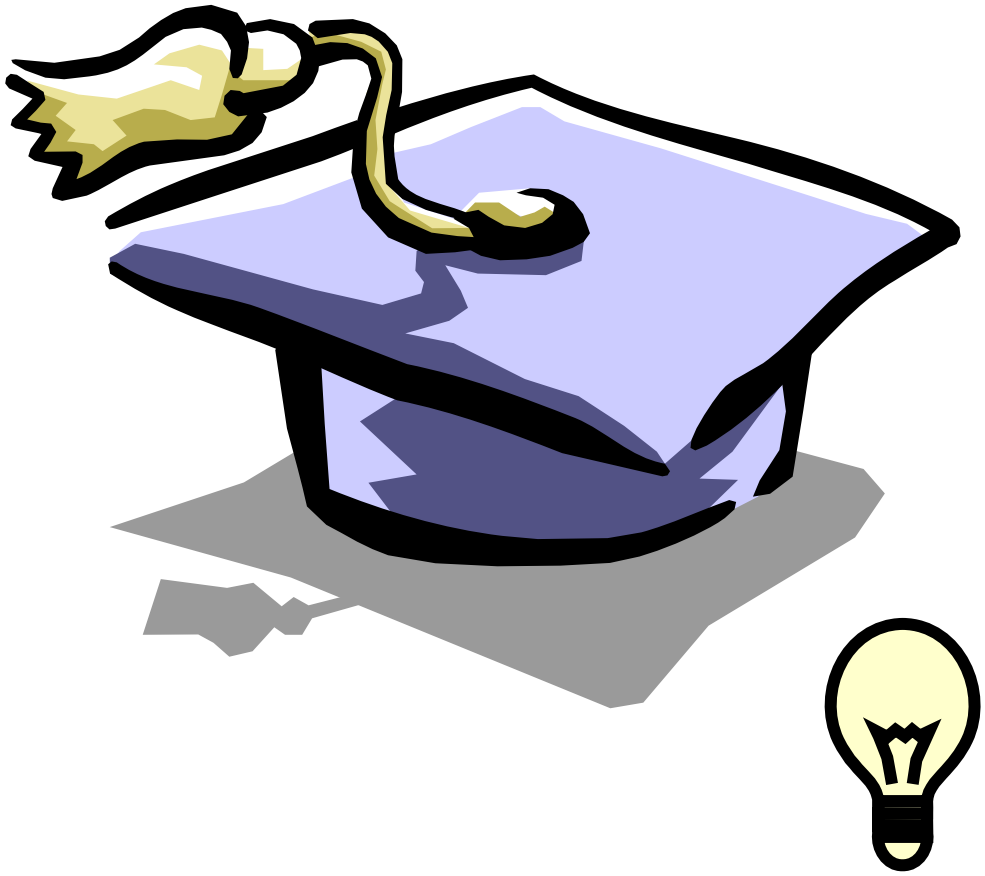


Power-On the Board and see the result:



Conclusion:

The application runs with the Debugger!
The application runs after Power-ON !!!



We learnt:

1.) The debugger cares for the correct initialization of the EBU.
2.) If we start after "Power-On" without the debugger (Power-on) we have to take care of the settings ourselfs.
3.) We have to change the start-up-file (cstart.asm) to correct the External-Boot-Memory-Configuration-Word.
4.) How to create a workaround.
5.) How to use the pls-debugger for debugging (memory-view).

OnBoardSDRAM-Pattern-Test:

Start Tasking EDE and open the project:

File – Open Project Space

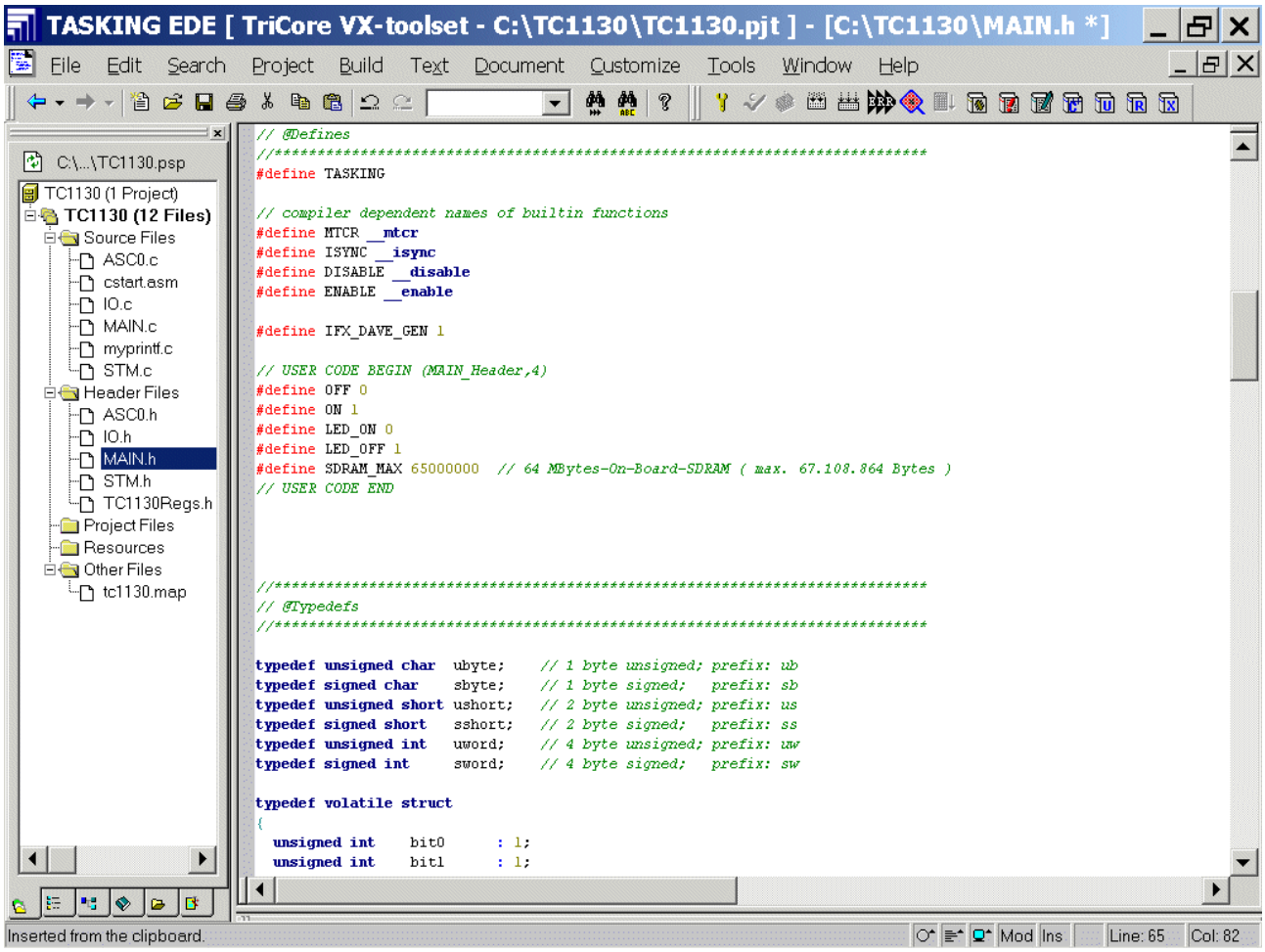
Look in: select C:\TC1130
File name: select TC1130.psp

Open

Insert your application specific program:

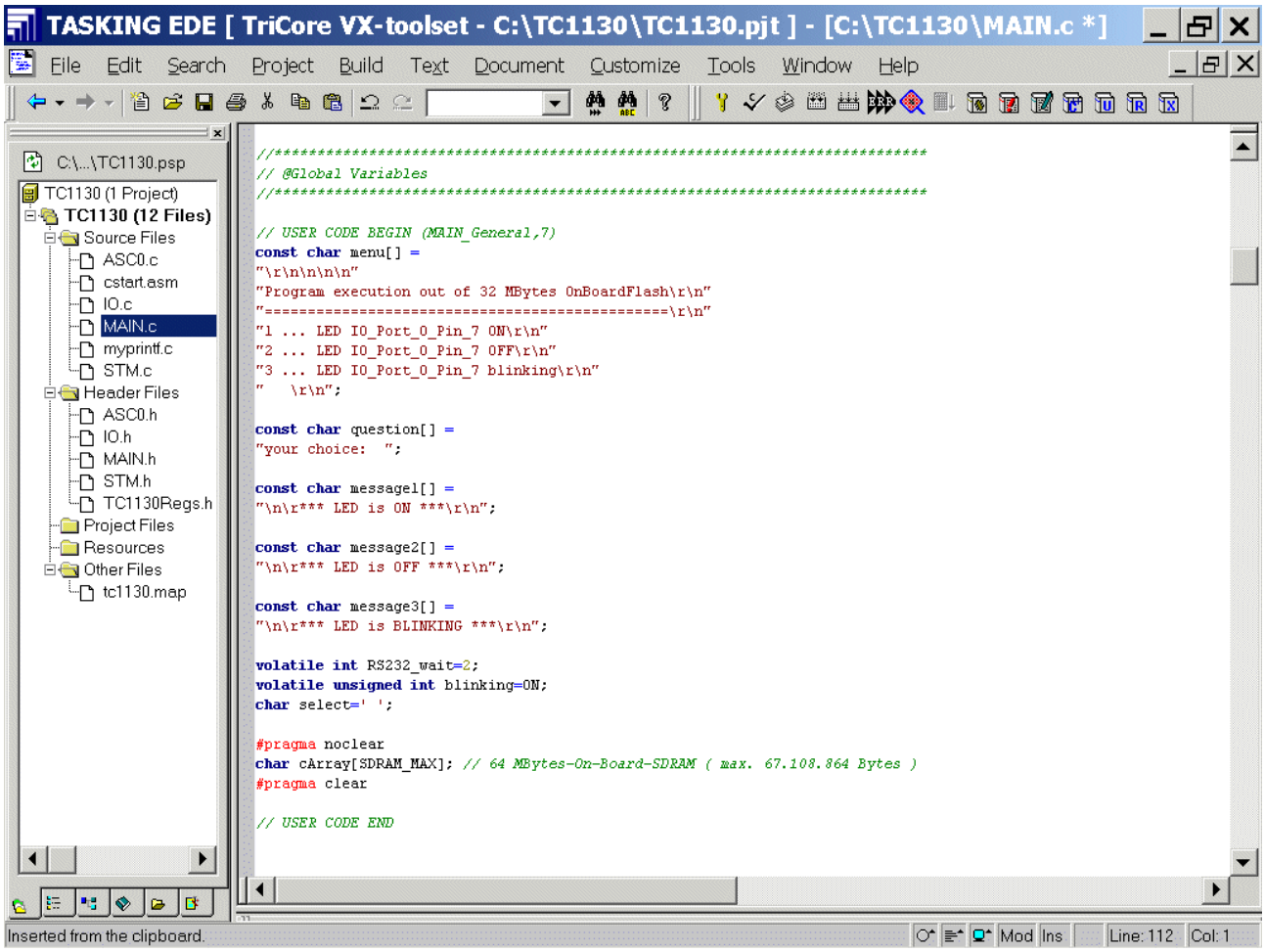
Double click: Main.h and insert the following Defines:

```
#define SDRAM_MAX 65000000  // 64 MBytes-On-Board-SDRAM ( max. 67.108.864 Bytes )
```



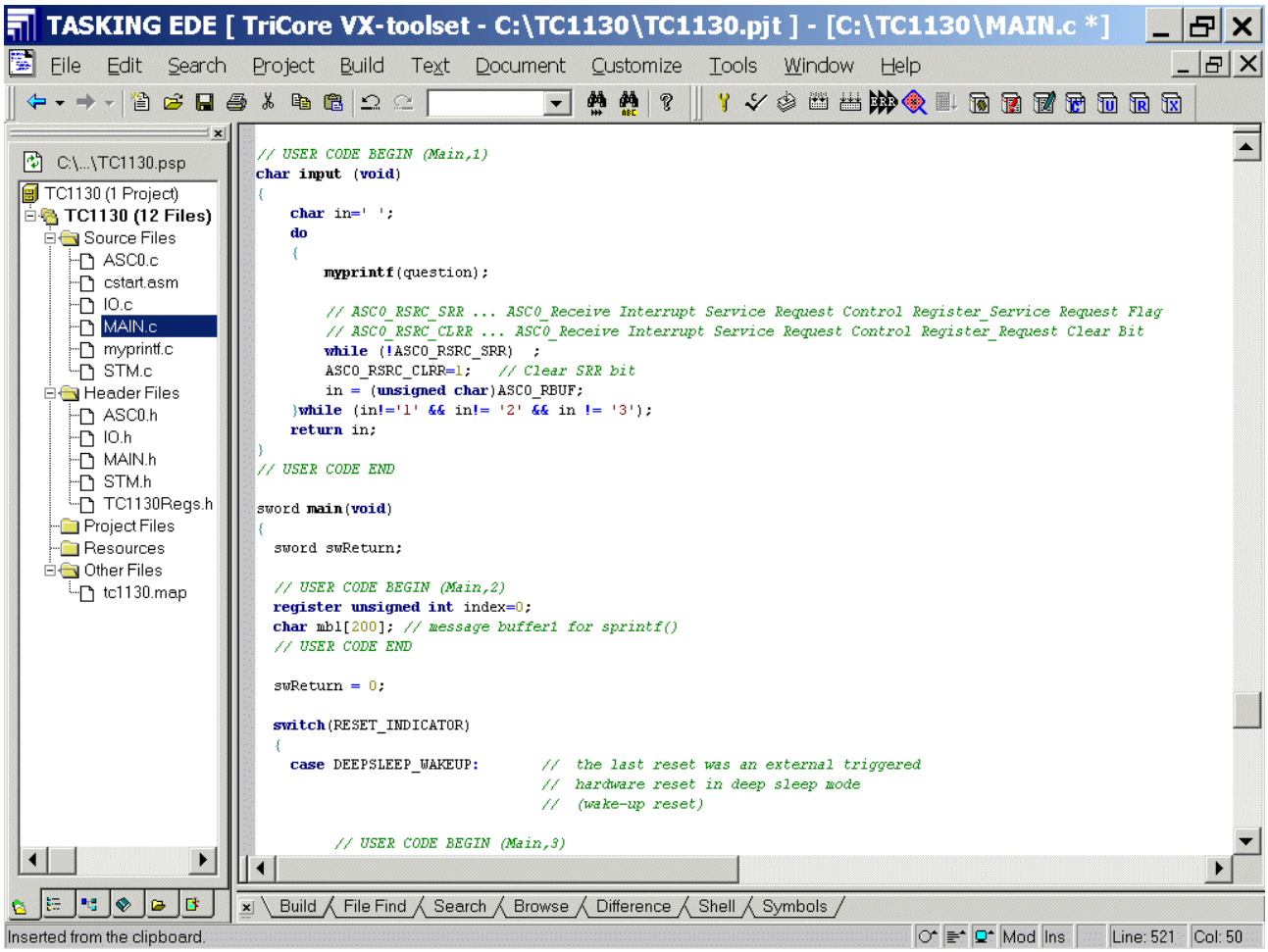
Double click: MAIN.C and insert Global Variables:

```
#pragma noclear
char cArray[SDRAM_MAX]; // 64 MBytes-On-Board-SDRAM ( max. 67.108.864 Bytes )
#pragma clear
```



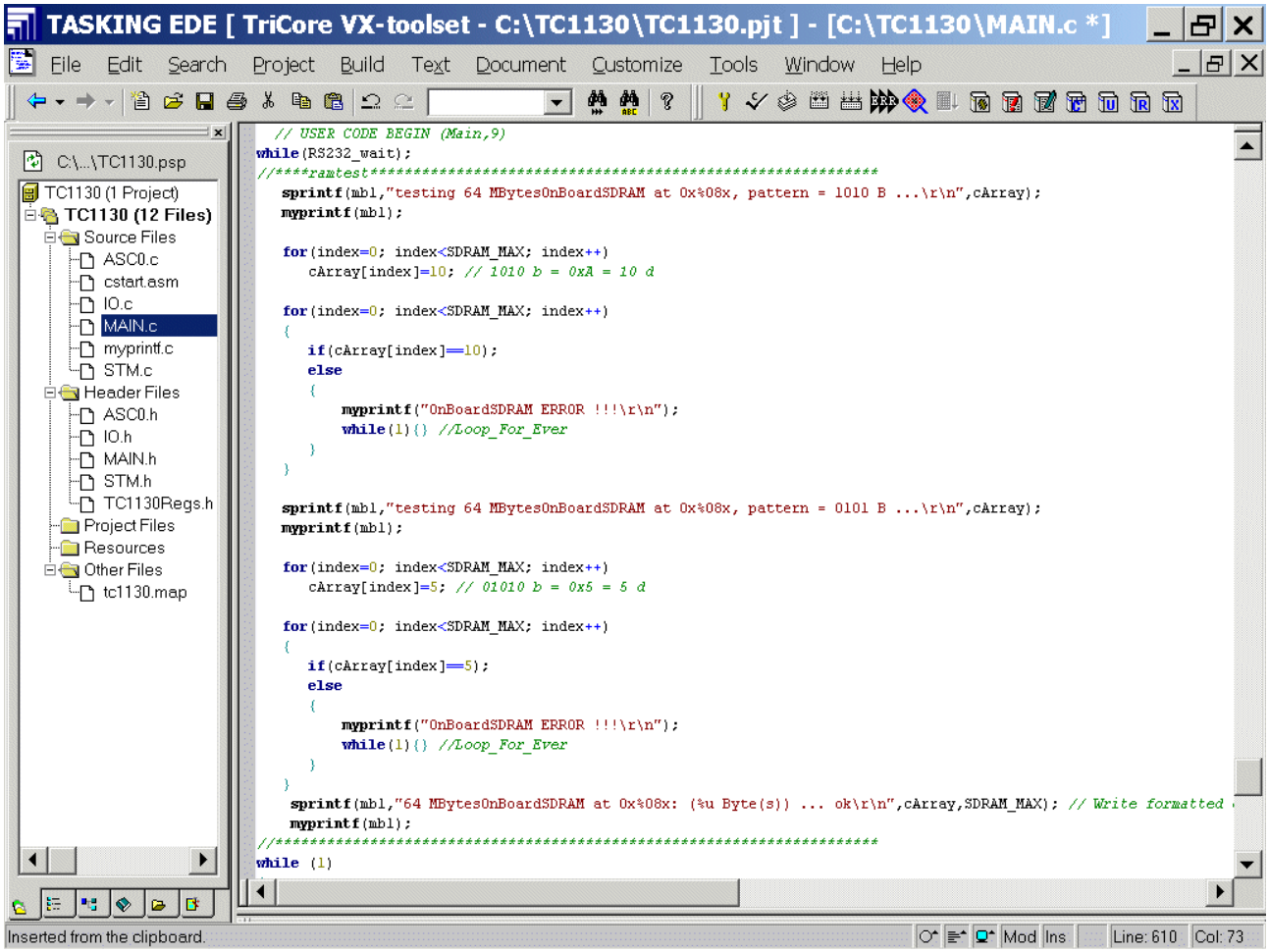
Double click: MAIN.C and insert the following code in the main function:

```
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 64 MBytesOnBoardSDRAM at 0x%08x, pattern = 1010 B
...\r\n",cArray);
    myprintf(mb1);

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

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

    sprintf(mb1,"testing 64 MBytesOnBoardSDRAM at 0x%08x, pattern = 0101 B
...\r\n",cArray);
    myprintf(mb1);

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

    for(index=0; index<SDRAM_MAX; index++)
    {
         if(cArray[index]==5);
         else
         {
               myprintf("OnBoardSDRAM ERROR !!!\r\n");
               while(1){} //Loop_For_Ever
         }
    }
       sprintf(mb1,"64 MBytesOnBoardSDRAM at 0x%08x: (%u Byte(s)) ...
ok\r\n",cArray,SDRAM_MAX); // Write formatted data to string mb
       myprintf(mb1);
//************************************************************************
```



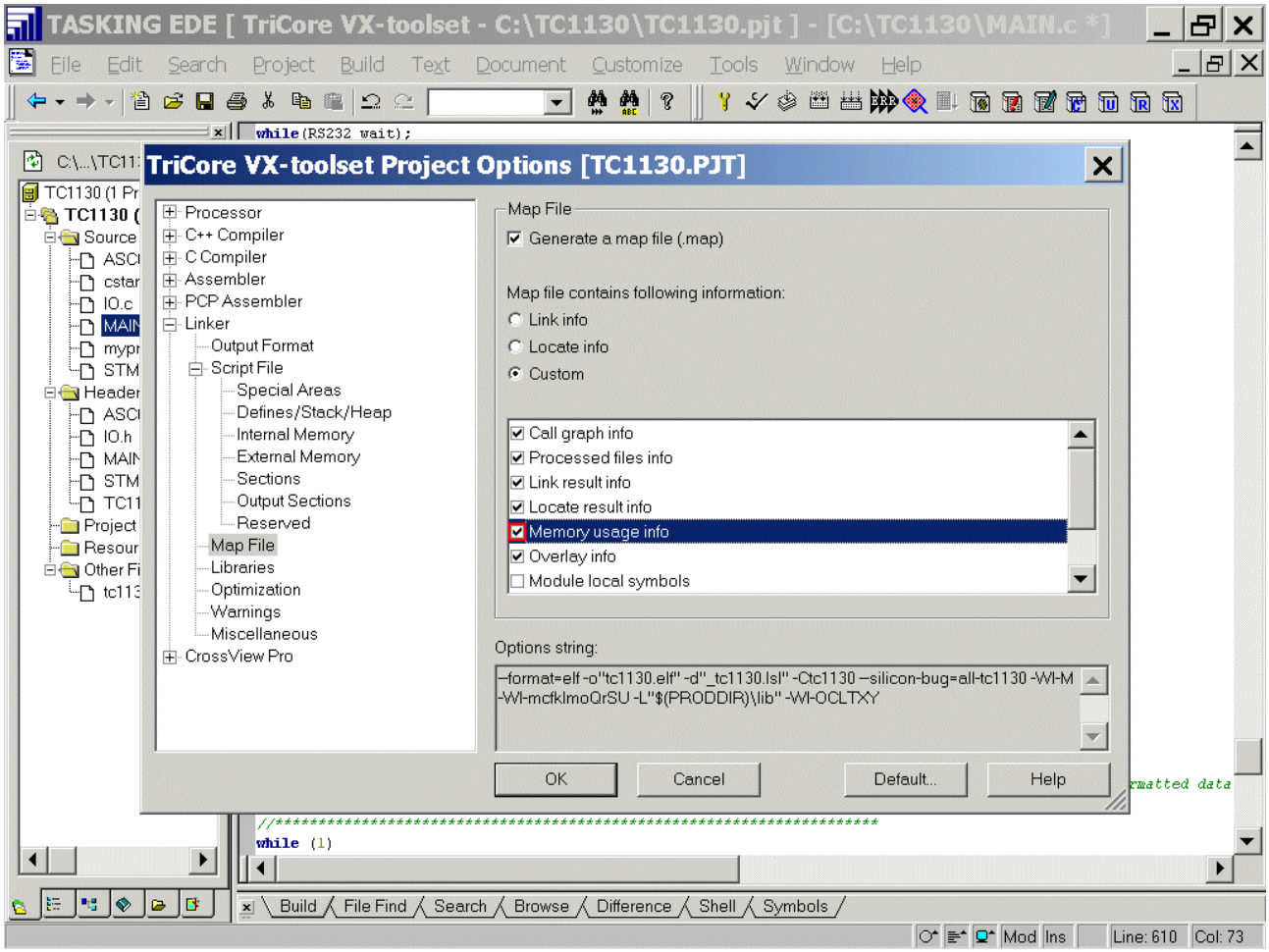
Note:

Testing 65.000.000 Bytes take about 11 minutes 13 seconds.
Therefore we suggest to make the array (cArray) smaller
(e.g. #define SDRAM_MAX 100).

Configure Linker – Control:

Project – Project Options

Linker: Map File: check/click ✓ Memory usage info

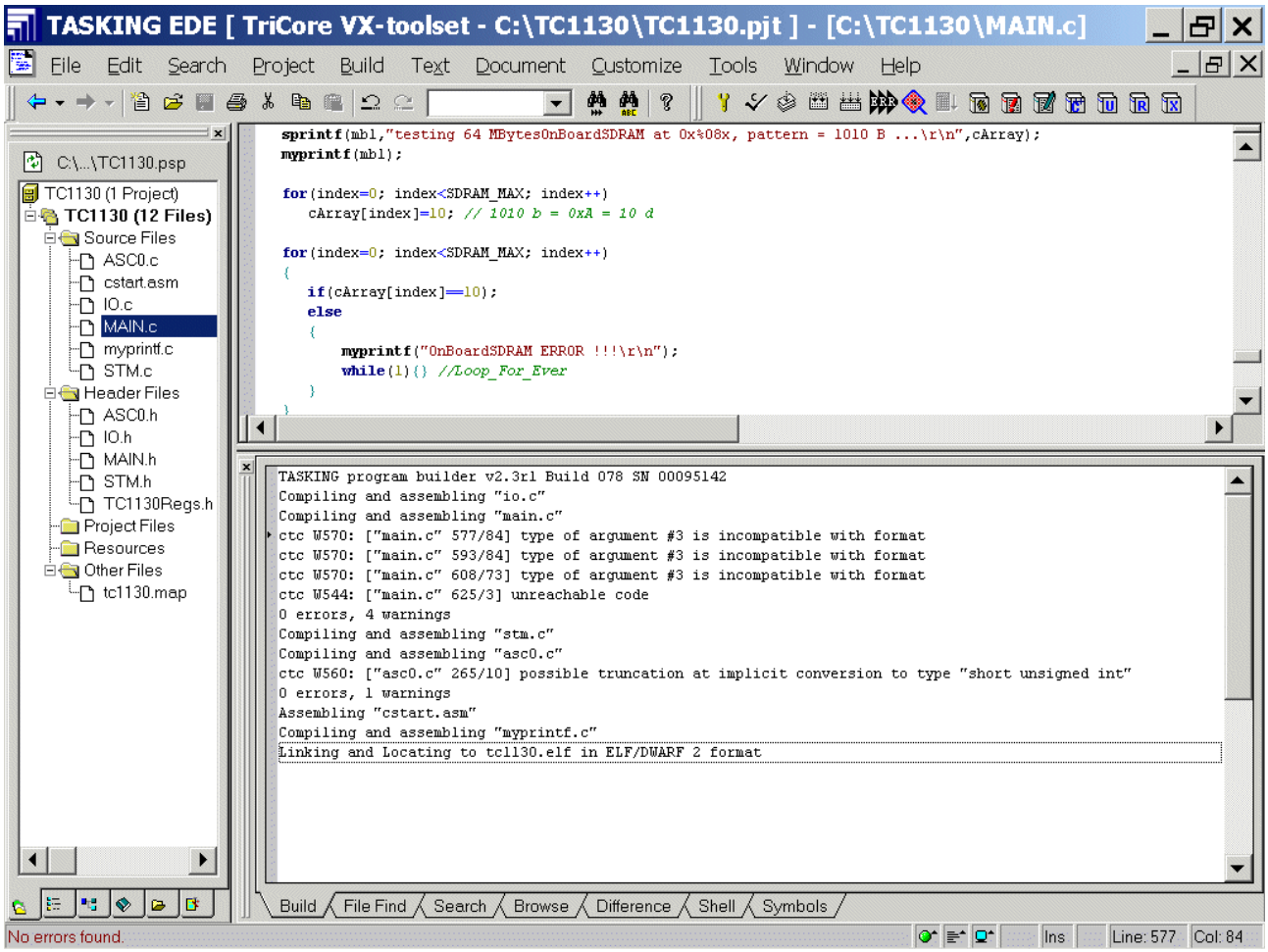


OK

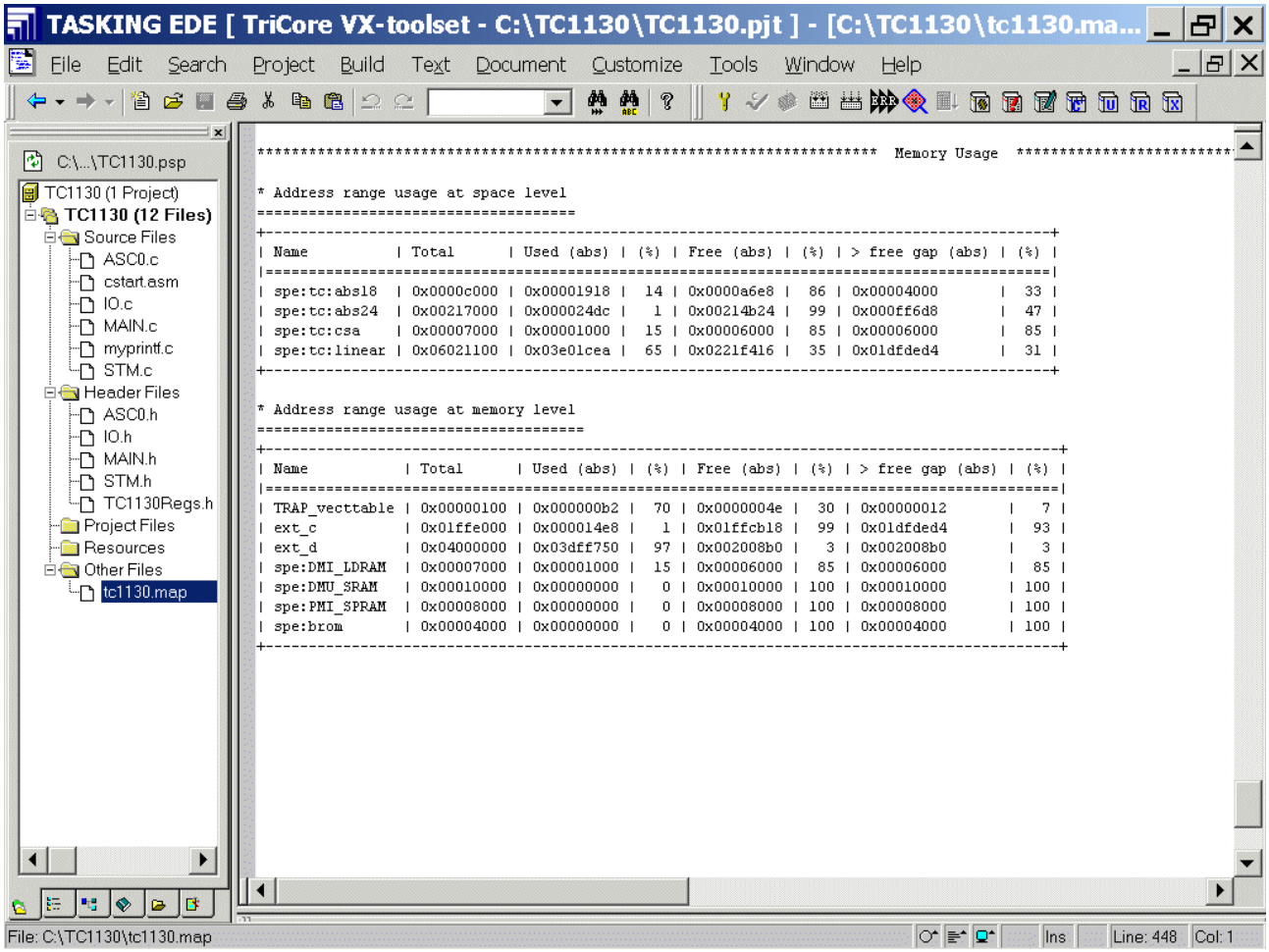
Generate your application program:

Build
Rebuild

or



See the Memory Usage:



Now you can close both 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:\TC1130
File name: select TC1130.wsp

Open
Cancel

File – Load Program

Look in: select TC1130
File name: select TC1130.elf

Open

click Program All

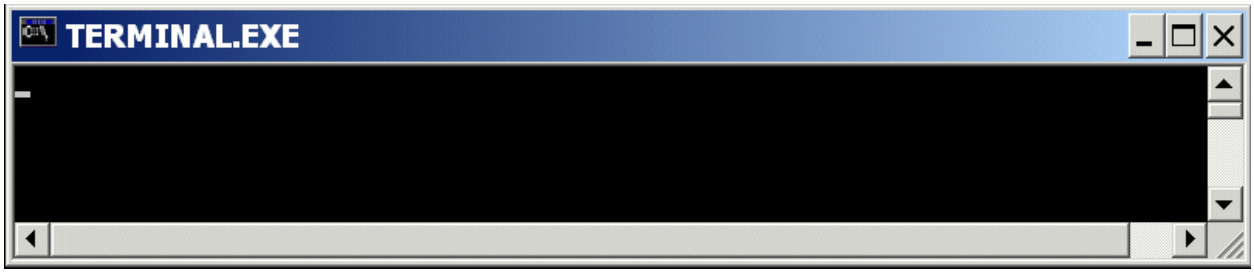
Exit
Exit

File – Close Workspace

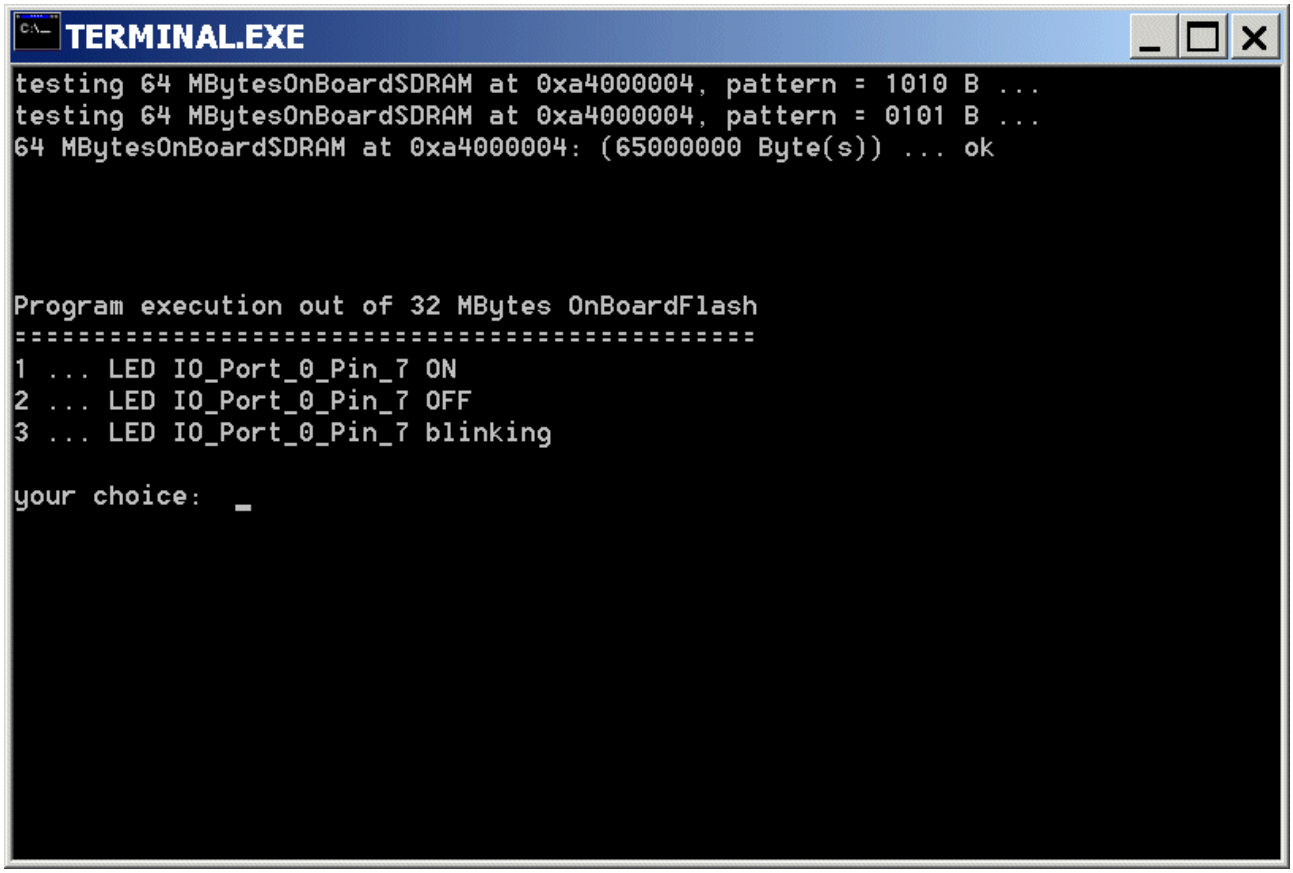
Yes

File – Exit

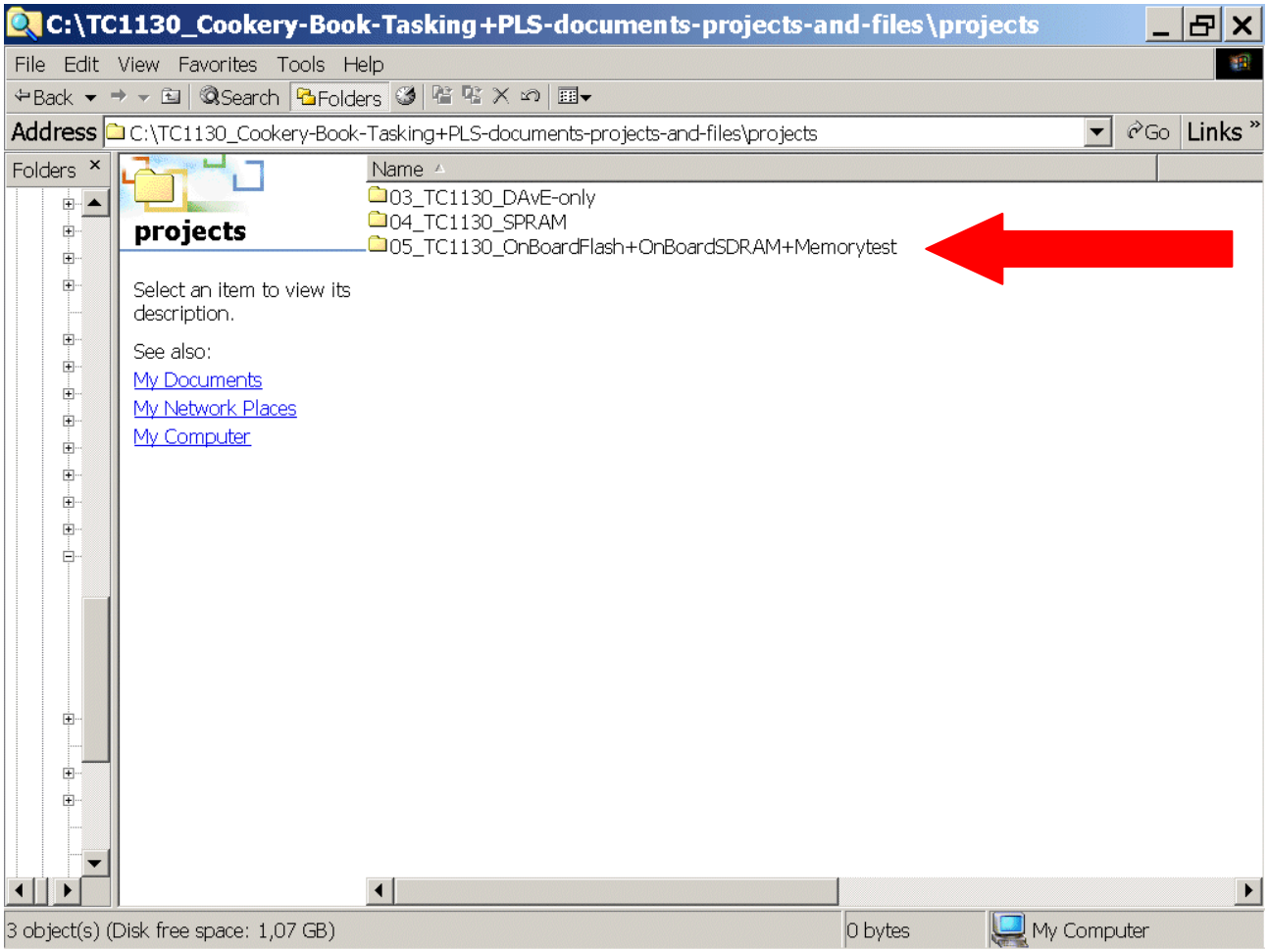
Execute any terminal-program
(9600 Baud, 8 bit Data, no Parity-Bit, 1 Stop-Bit, Xon/Xoff Protocol):



Power-On the Board and see the result:



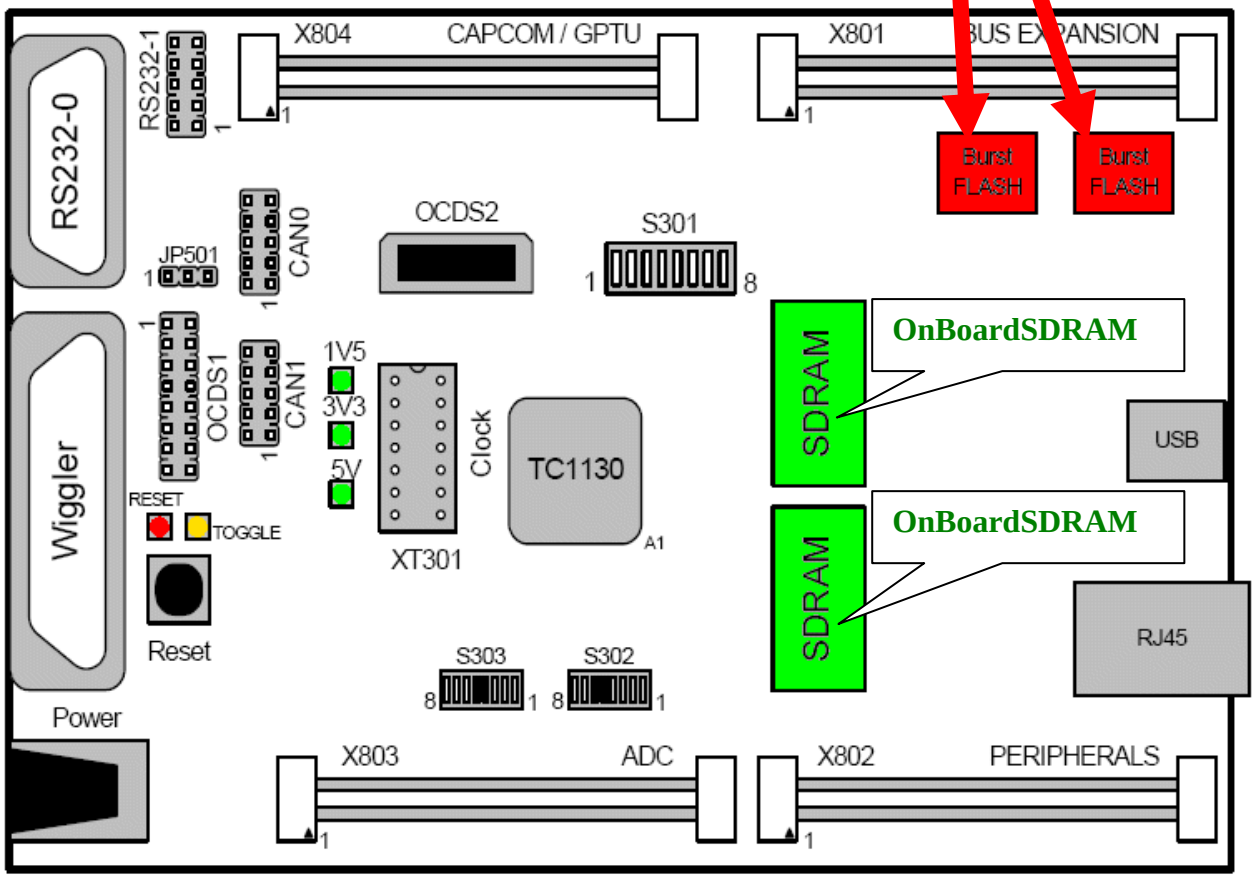
We recommend now to copy and store your project-directory “C:\TC1130” to “05_TC1130_OnBoardFlash+OnBoardSDRAM+Memorytest”:



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

Write programs

for execution from Intel's 32 Mbytes OnBoardFlash (BURST-Mode)



Start Tasking EDE and open the project:

File – Open Project Space

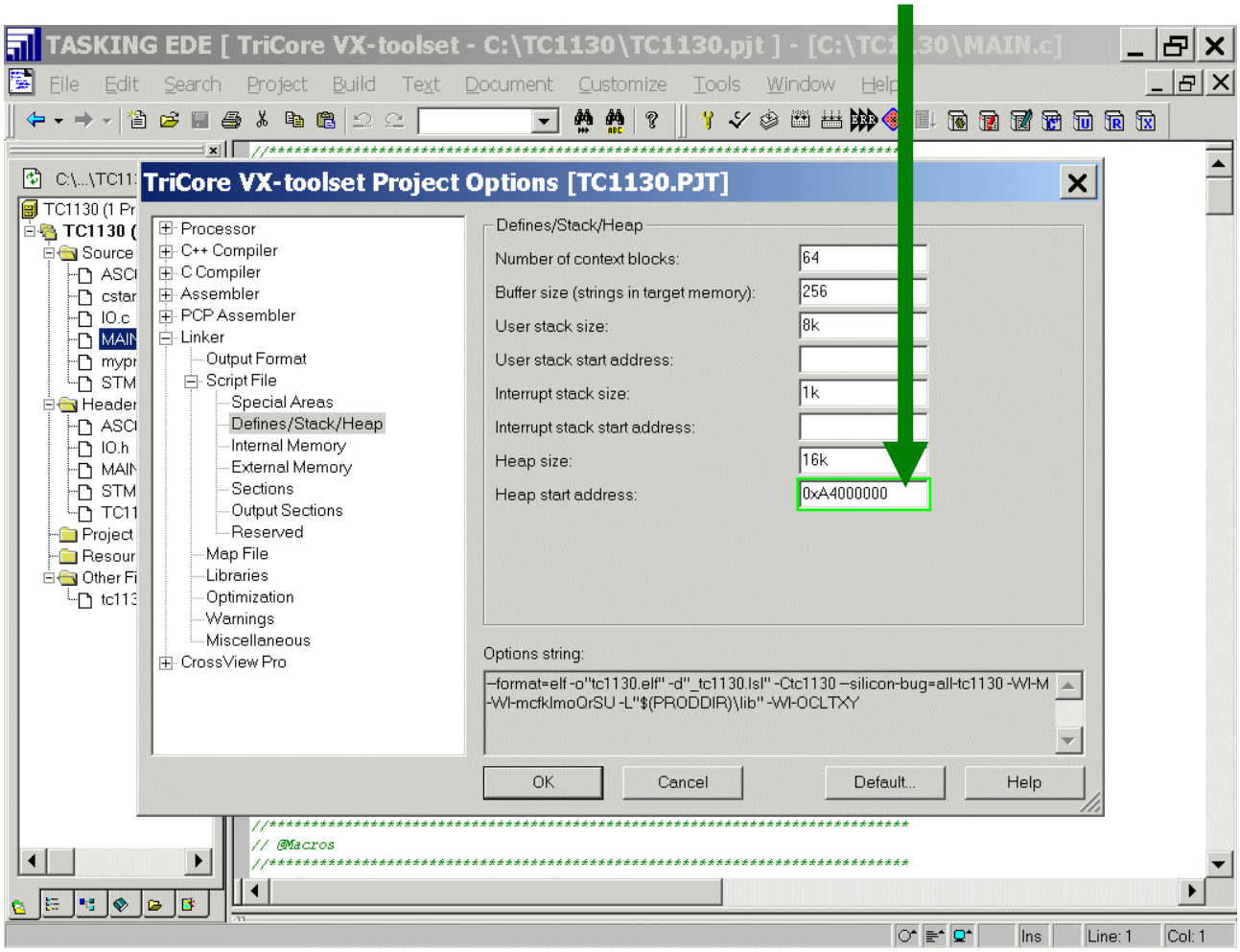
Look in: select C:\TC1130
File name: select TC1130.psp

Open

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

Project – Project Options

Linker: Script File: Defines/Stack/Heap: Heap start address: insert 0xA4000000



OK

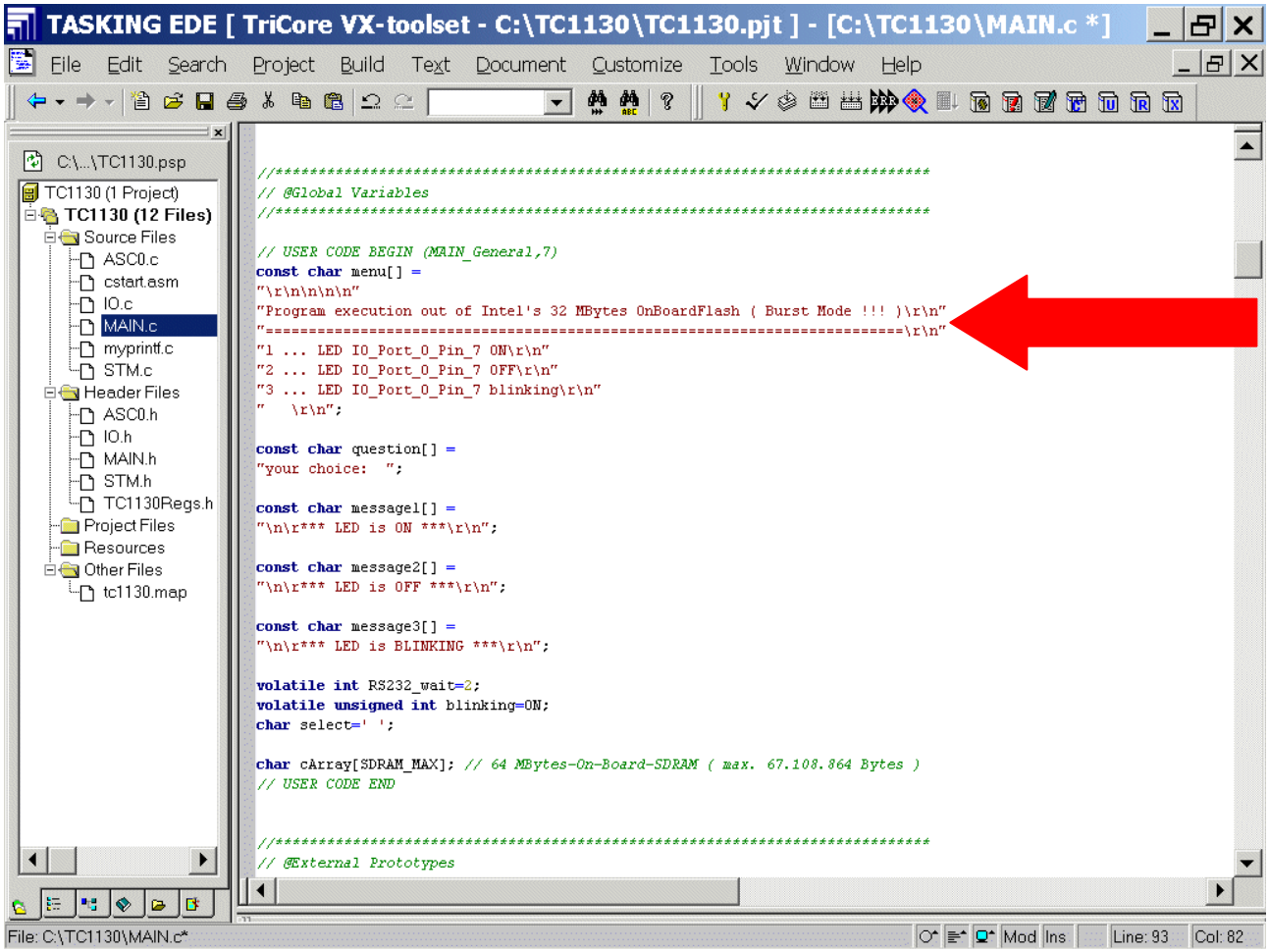
Insert your application specific program:

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

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of 32 MBytes OnBoardFlash\r\n"
"================================================\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 blinking\r\n"
"   \r\n";
```

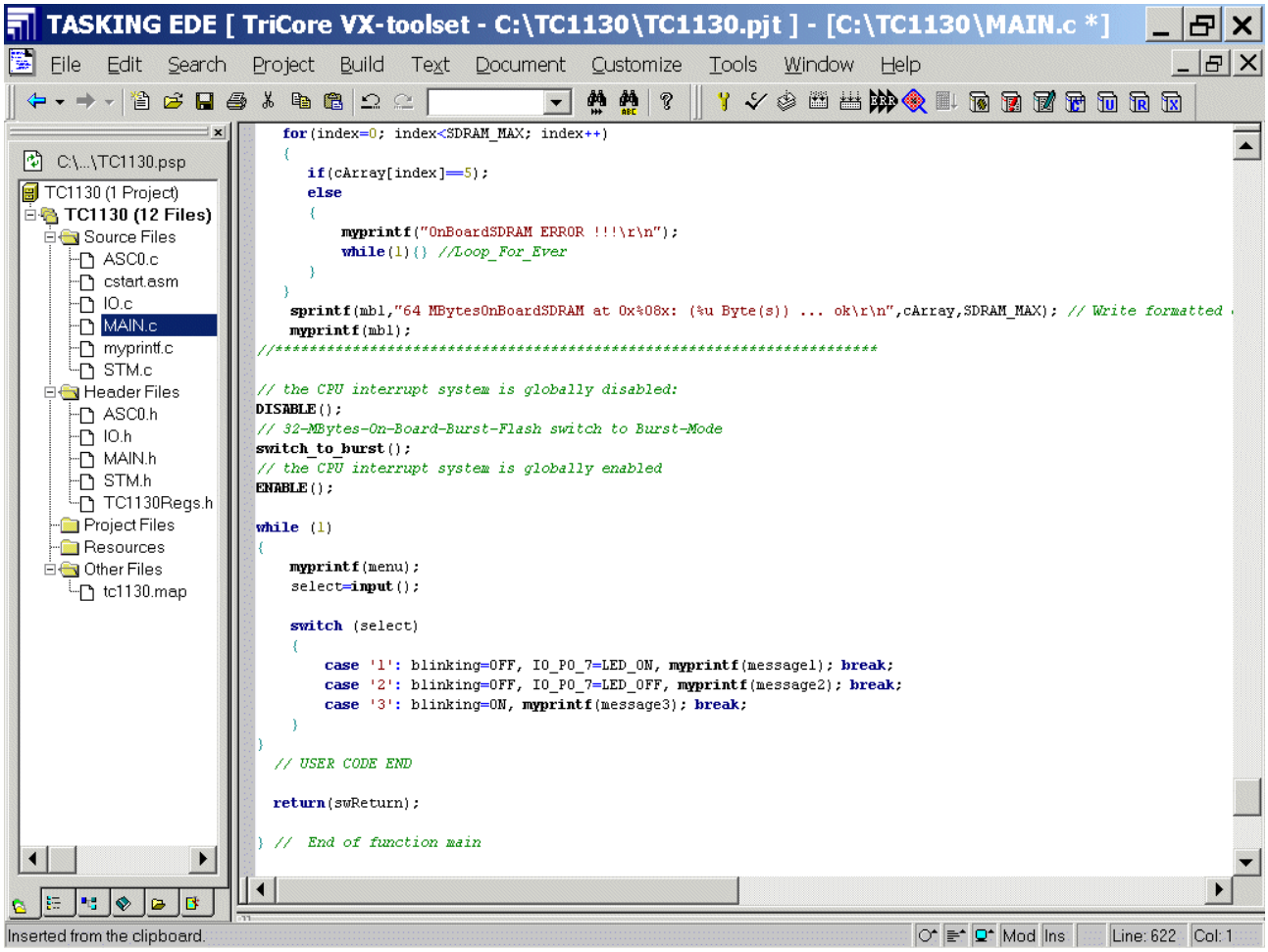
to

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of Intel's 32 MBytes OnBoardFlash ( Burst
Mode !!! )\r\n"
"================================================================
========\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 blinking\r\n"
"   \r\n";
```


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1130.psp
TC1130 (1 Project)
TC1130 (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
TC1130Regs.h
Project Files
Resources
Other Files
tc1130.map
//***
// @Global Variables
//***
// USER CODE BEGIN (MAIN_General,7)
const char menu[] =
"\r\n\n\n\n"
"Program execution out of Intel's 32 MBytes OnBoardFlash (Burst Mode !!!)\r\n"
"==\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 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[SDRAM_MAX]; // 64 MBytes-On-Board-SDRAM (max. 67.108.864 Bytes)
// USER CODE END
//***
// @External Prototypes
File: C:\TC1130\MAIN.c*
Mod Ins
Line: 93
Col: 82

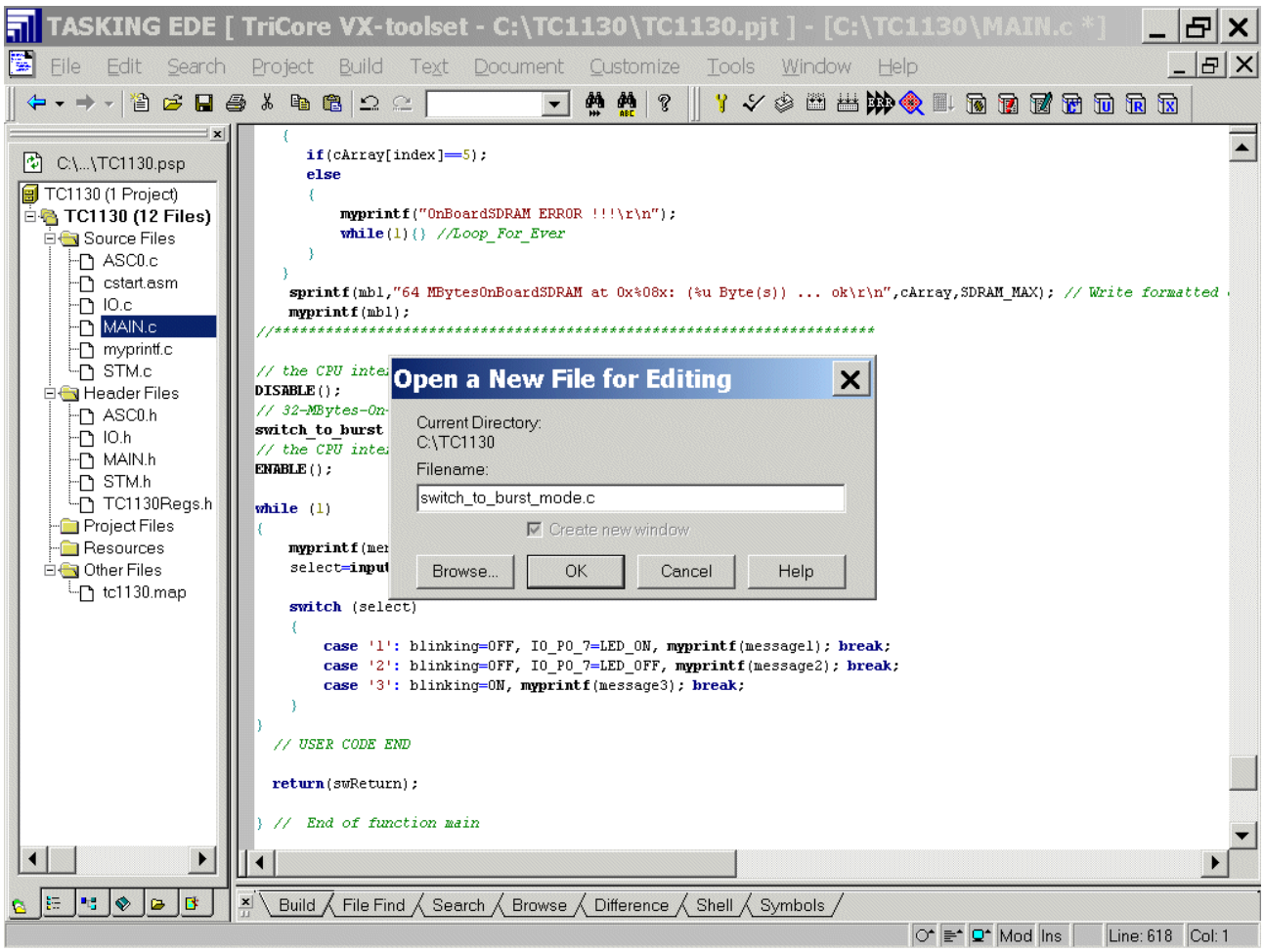
Double click: Main.c: insert application-specific-program:

```
// the CPU interrupt system is globally disabled:
DISABLE();
// 32-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

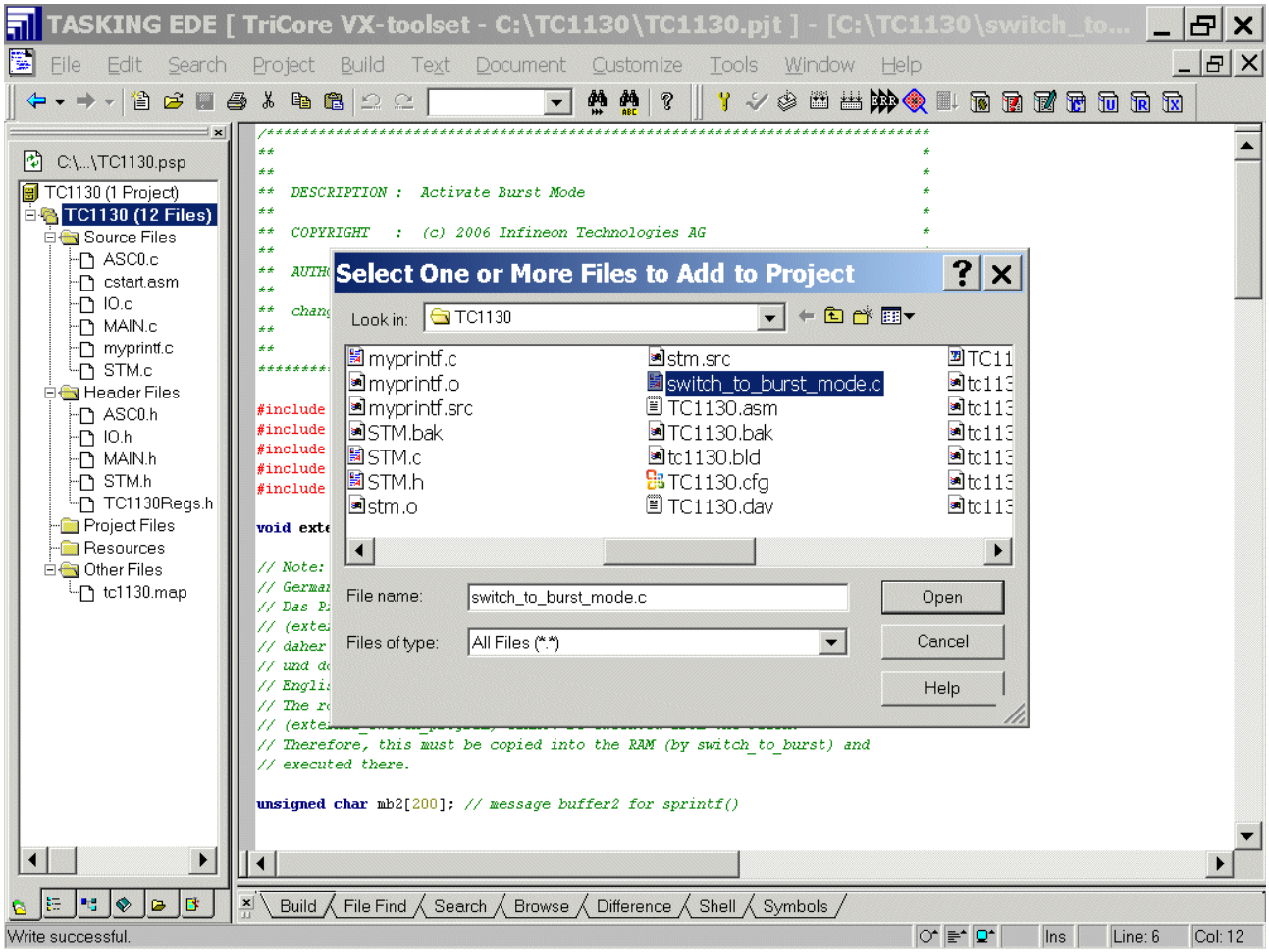
Insert code:

```
/*****************************************************************************
**                                                                          *
**                                                                          *
**  DESCRIPTION :  Activate Burst Mode                                      *
**                                                                          *
**  COPYRIGHT   :  (c) 2006 Infineon Technologies AG                        *
**                                                                          *
**  AUTHOR  :  Holger Dienst (AIM MC ATV AE)                                *
**                                                                          *
**  changed/used by :  Wilhelm Brezovits, April/May 2007                    *
**                                                                          *
**                                                                          *
                                  *
*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "MAIN.h"
#include "ebu.h"
#include "types.h"

void external_switch_program(void);

// Note:
// The routine which switches the Flash
// (external_switch_program) cannot be executed from the Flash.
// Therefore, this must be copied into the RAM/heap (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 mode on the flash
      puiAdr = (u_int32 *)(uiCS0_Base + (0x28C2<<2));
      *puiAdr = 0x00600060;

      puiAdr = (u_int32 *)(uiCS0_Base + (0x28C2<<2));
      *puiAdr = 0x00030003;

      // setting BUSCON0
      // setting AGEN to BURST0 and WAIT to synchronous wait
      psEBU->BUSCON0 &= ~(EBUAGEN_MSK | EBUWAIT_MSK);
      psEBU->BUSCON0 |= EBUAGEN_BURST0 | EBUWAIT_SYN;
      // setting BUSAP0
      // setting WAITRDC to 1 (more waitstates are inserted via WAIT signal)
      psEBU->BUSAP0 &= ~(EBUWAITRDC_MSK);
      psEBU->BUSAP0 |= 0x00400000;
      // */
      __isync();
      __dsync();
      // we make a dummy read, to make absolutly sure that our EBU settings are
done, if we return
```

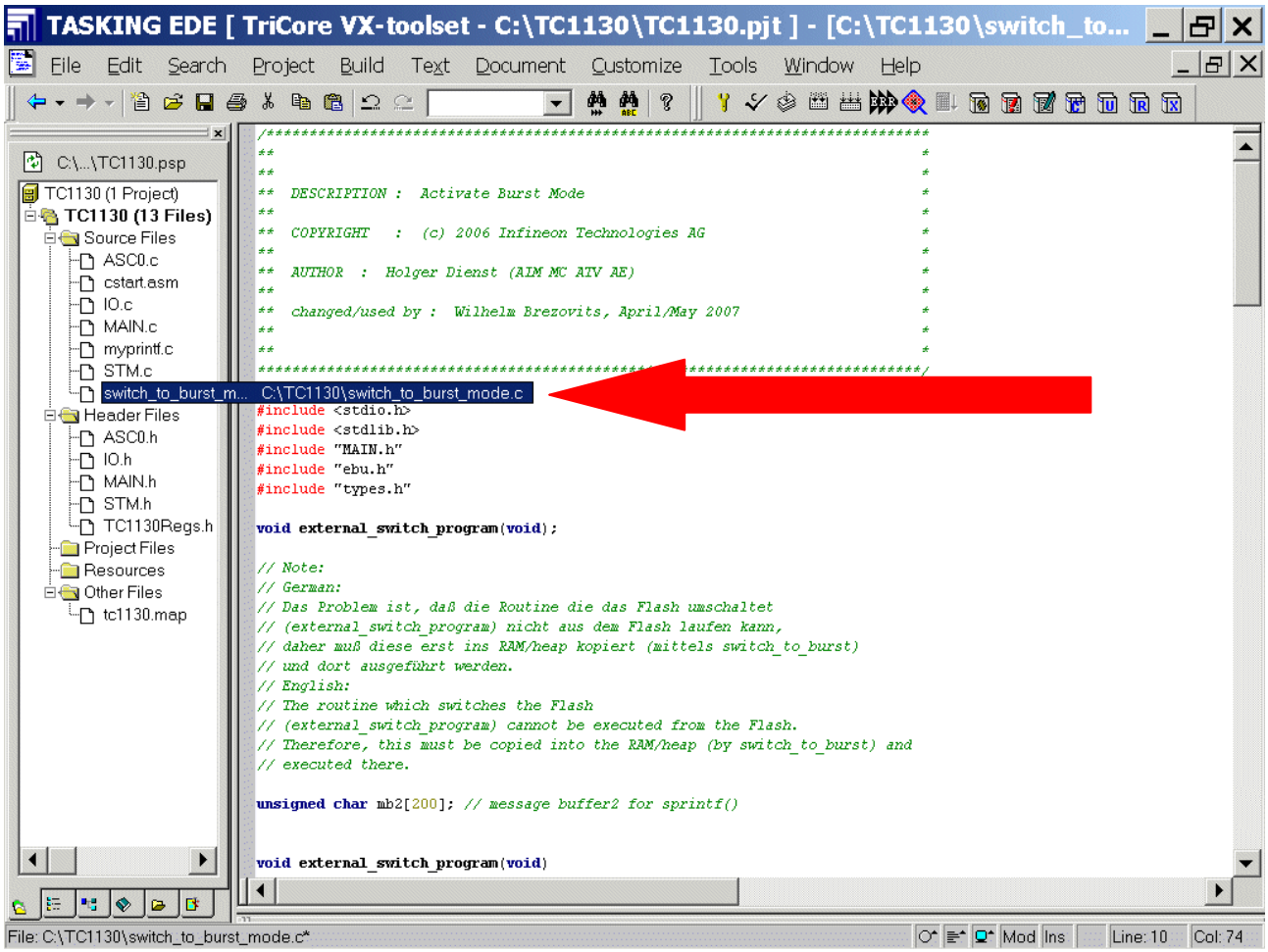
```
      uiCS0_Base = psEBU->BUSAP0;
}

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 0x%08x (heap)
allocated\r\n",byte_count,puiDataHeap); //Write formatted data to string mb2
        myprintf(mb2);
        byte_count=0;

        // first copy all codes to external sdram
        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");

        ((void(*)(void))puiDestination)(); // jump to external ram (heap)

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

File – Save

(Project Window File View) – TC1130 (Files) – right mouse button click – Add Existing Files – Browse

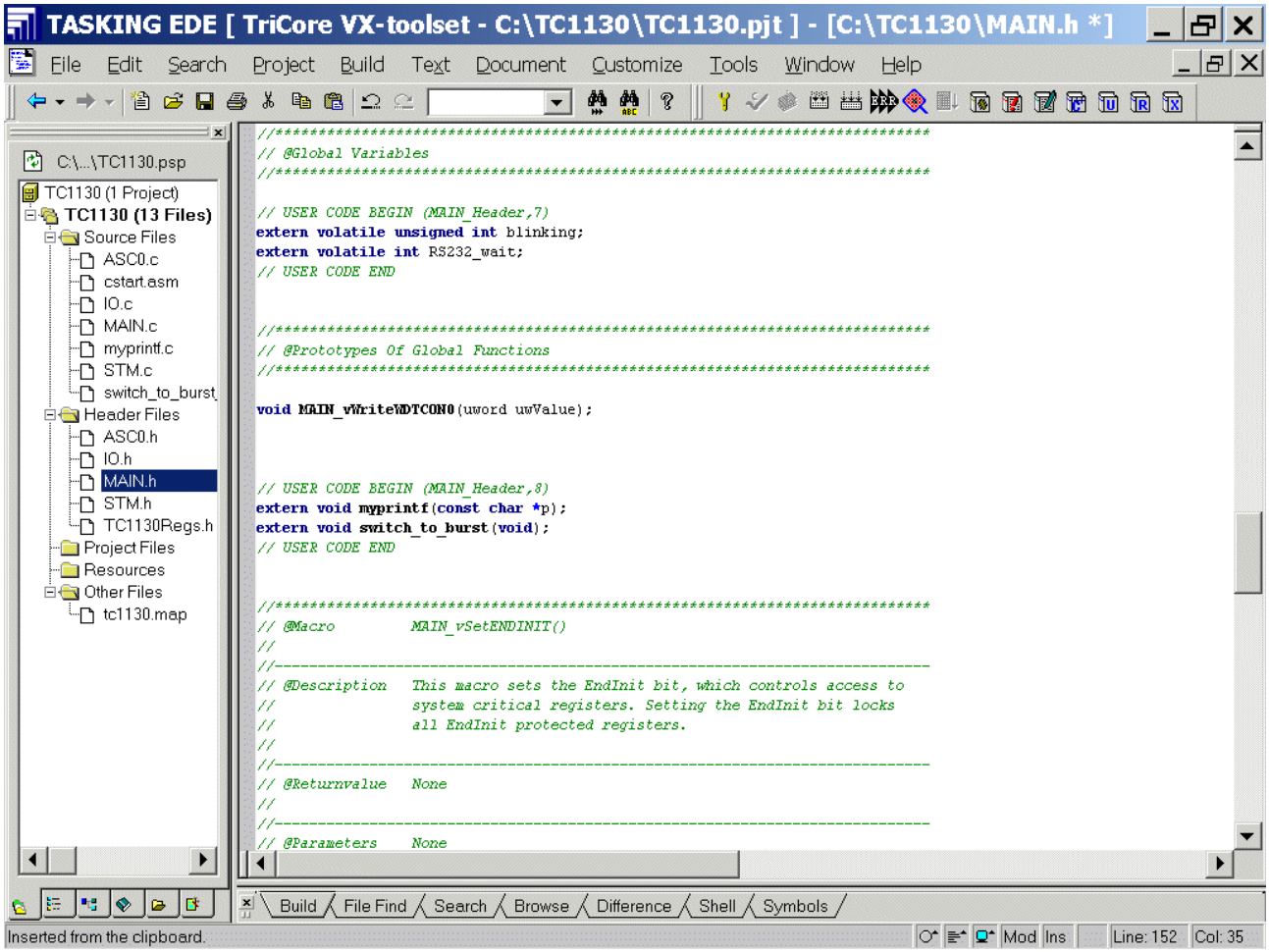


Select switch_to_burst_mode.c
Open
OK


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\switch_to...
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1130.psp
TC1130 (1 Project)
TC1130 (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
TC1130Regs.h
Project Files
Resources
Other Files
tc1130.map
C:\TC1130\switch_to_burst_mode.c
** DESCRIPTION : Activate Burst Mode
** COPYRIGHT : (c) 2006 Infineon Technologies AG
** AUTHOR : Holger Dienst (AIM MC ATV AE)
** changed/used by : Wilhelm Brezovits, April/May 2007
#include <stdio.h>
#include <stdlib.h>
#include "MAIN.h"
#include "ebu.h"
#include "types.h"
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/heap 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/heap (by switch_to_burst) and
// executed there.
unsigned char mb2[200]; // message buffer2 for sprintf()
void external_switch_program(void)
File: C:\TC1130\switch_to_burst_mode.c*
Mod Ins
Line: 10
Col: 74

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 TC1130                    *
*                                                                            *
*  COPYRIGHT   :  (c) 2002 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 (R) */
        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[7];  /* Reserved */
        volatile unsigned int SDRMREF0;      /* SDRAM Type 0 Refresh Control */
        volatile unsigned int RESERVED4[1];  /* Reserved */
        volatile unsigned int SDRMREF1;      /* SDRAM Type 1 Refresh Control */
        volatile unsigned int RESERVED5[1];  /* Reserved */
        volatile unsigned int SDRMCON0;      /* SDRAM Type 0 Configuration */
        volatile unsigned int RESERVED6[1];  /* Reserved */
        volatile unsigned int SDRMCON1;      /* SDRAM Type 1 Configuration */
        volatile unsigned int RESERVED7[1];  /* Reserved */
        volatile unsigned int SDRMOD0;       /* SDRAM Type 0 Mode */
        volatile unsigned int RESERVED8[1];  /* Reserved */
        volatile unsigned int SDRMOD1;       /* SDRAM Type 1 Mode */
        volatile unsigned int RESERVED9[1];  /* Reserved */
        volatile unsigned int SDRSTAT0;      /* SDRAM Type 0 Status */
        volatile unsigned int RESERVED10[1]; /* Reserved */
        volatile unsigned int SDRSTAT1;      /* SDRAM Type 1 Status */
        volatile unsigned int RESERVED11[1]; /* Reserved */
        volatile unsigned int ADDSEL0;       /* Address Select Register 0 (32) */
        volatile unsigned int RESERVED12[1]; /* Reserved */
        volatile unsigned int ADDSEL1;       /* Address Select Register 1 (32) */
        volatile unsigned int RESERVED13[1]; /* Reserved */
        volatile unsigned int ADDSEL2;       /* Address Select Register 2 (32) */
        volatile unsigned int RESERVED14[1]; /* Reserved */
        volatile unsigned int ADDSEL3;       /* Address Select Register 3 (32) */
        volatile unsigned int RESERVED15[9]; /* Reserved */
        volatile unsigned int BUSCON0;       /* Bus Configuration Register 0 (32) */
        volatile unsigned int RESERVED16[1]; /* Reserved */
        volatile unsigned int BUSCON1;       /* Bus Configuration Register 1 (32) */
        volatile unsigned int RESERVED17[1]; /* Reserved */
        volatile unsigned int BUSCON2;       /* Bus Configuration Register 2 (32) */
        volatile unsigned int RESERVED18[1]; /* Reserved */
        volatile unsigned int BUSCON3;       /* Bus Configuration Register 3 (32) */
        volatile unsigned int RESERVED19[9]; /* Reserved */
```

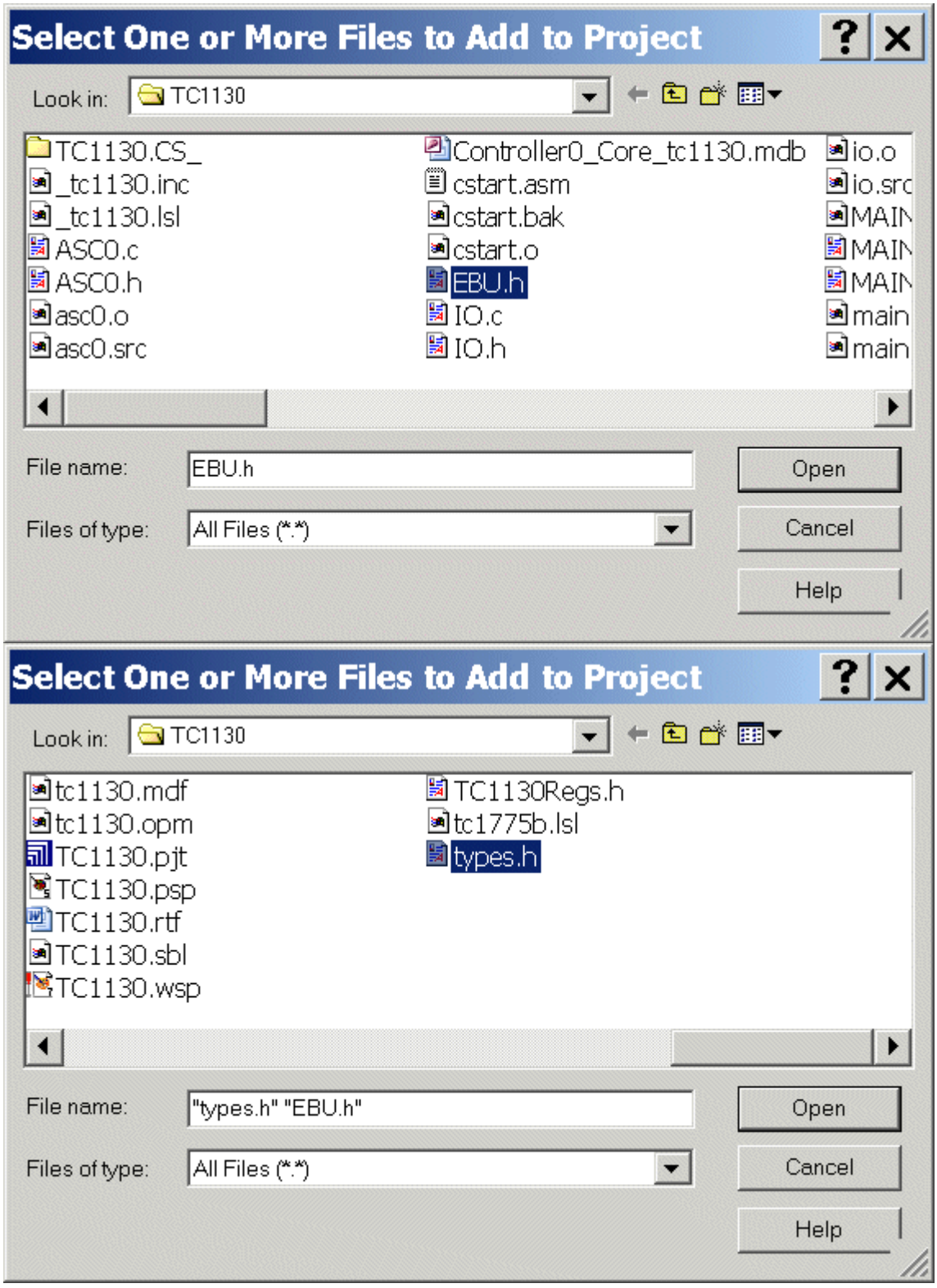
```
        volatile unsigned int BUSAP0;        /* Bus access parameter Register 0 (32) */
        volatile unsigned int RESERVED20[1]; /* Reserved */
        volatile unsigned int BUSAP1;        /* Bus access parameter Register 1 (32) */
        volatile unsigned int RESERVED21[1]; /* Reserved */
        volatile unsigned int BUSAP2;        /* Bus access parameter Register 2 (32) */
        volatile unsigned int RESERVED22[1]; /* Reserved */
        volatile unsigned int BUSAP3;        /* Bus access parameter Register 3 (32) */
        volatile unsigned int RESERVED23[17];/* Reserved */
        volatile unsigned int EMUAS;         /* Emulator Memory Address Select Register (32) */
        volatile unsigned int RESERVED24[1]; /* Reserved */
        volatile unsigned int EMUBC;         /* Emulator Memory Bus Configuration Register (32) */
        volatile unsigned int RESERVED25[1]; /* Reserved */
        volatile unsigned int EMUBAP;        /* Emulator Memory access parameter (32) */
        volatile unsigned int RESERVED26[1]; /* Reserved */
        volatile unsigned int EMUOVL;        /* Overlay memory chip-select generation (32) */
        volatile unsigned int RESERVED27[5]; /* Reserved */
        volatile unsigned int USERCON;       /* Test/Control Configuration Register (32) */

} EBU;

/* Global Control Register */
#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 EBUBUSCLK_MSK       0x03000000      /* EBU bus clock */
#define EBUBUSCLK_1         0x00000000      /* EBU bus clock = LMB clock */
#define EBUBUSCLK_2         0x01000000      /* EBU bus clock = LMB clock/2 */
#define EBUBUSCLK_4         0x02000000      /* EBU bus clock = LMB clock/4 */
#define EBUSDCMSEL          0x04000000      /* EBU SDRAM clock gated */
#define EBUCS0FAM           0x08000000      /* EBU CS0 Fills Address Map */
#define EBUEMUFAM           0x10000000      /* EBU CSemu Fills Address Map */
#define EBUBFSSS            0x20000000      /* EBU Burst Flash Single Stage Synchronisation */

/* 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 */

/* SDRAM Refresh Register */
#define EBUREFRESHC_MSK     0x0000003F      /* Refresh Counter Period Value */
#define EBUREFRESHR_MSK     0x000001C0      /* Number of Refresh Commands */
#define EBUSELFREXST        0x00000200      /* Self refresh exit status */
#define EBUSELFREX          0x00000400      /* Self refresh exit */
#define EBUSELFRENST        0x00000800      /* Self refresh entry status */
#define EBUSELFREN          0x00001000      /* Self refresh entry */
#define EBUAUTOSELFR        0x00002000      /* Automatic Self Refresh (only SDRMREF0) */

/* SDRAM Configuration */
#define EBUCRAS_MSK         0x0000000F      /* Row Activate to Precharge Cycle Counter */
#define EBUCRFSH_MSK        0x000000F0      /* Refresh Commands Counter Value */
#define EBUCRSC_MSK         0x00000300      /* Mode Register Setup Time Counter Value */
#define EBUCRP_MSK          0x00000C00      /* Row Precharge Time Counter Value */
```

```
#define EBUAWIDTH_MSK       0x00003000      /* Starting Index of Address Lines in SDRAM Row Address
*/
#define EBUCRCD_MSK         0x0000C000      /* Row-to-Column Delay Counter Value */
#define EBUCRC_MSK          0x00070000      /* Row Cycle Time Counter Value */
#define EBUPAGEM_MSK        0x00380000      /* Mask for page tag */
#define EBUBANKM_MSK        0x01C00000      /* Mask for bank tag */
#define EBUDTALTNCY_MSK     0xF0000000      /* Mask Latency Cylce Control */

/* SDRAM Mode */
#define EBUBURSTL_MSK       0x00000007      /* SDRAM burst length mask */
#define EBUBURSTL_1         0x00000000      /* SDRAM burst length 1 */
#define EBUBURSTL_8         0x00000003      /* SDRAM burst length 8 */
#define EBUCASLAT_MSK       0x00000070      /* CAS-Latency Mask */
#define EBUCASLAT_2         0x00000020      /* CAS-Latency 2 */
#define EBUCASLAT_3         0x00000030      /* CAS-Latency 3 */

/* SDRAM Status Register */
#define EBUREFERR           0x00000001      /* SDRAM Refresh Error Flag */
#define EBUSDRMBUSY         0x00000002      /* SDRAM Busy Flag */

/* 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 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 EBUBCGEN_DQMM       0x00300000      /* DQM 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 EBUAGEN_MSK         0x70000000      /* Address Generation Control */
#define EBUAGEN_DEMUX       0x00000000      /* Demultiplex address */
#define EBUAGEN_MUX         0x10000000      /* Multiplex address */
#define EBUAGEN_BURST0      0x20000000      /* Burst Flash Type 0 access */
#define EBUAGEN_SDRAM0      0x30000000      /* SDRAM Type 0 access */
#define EBUAGEN_SDRAM1      0x40000000      /* SDRAM Type 1 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 */
```

```
#endif  /*  EBU H */
```

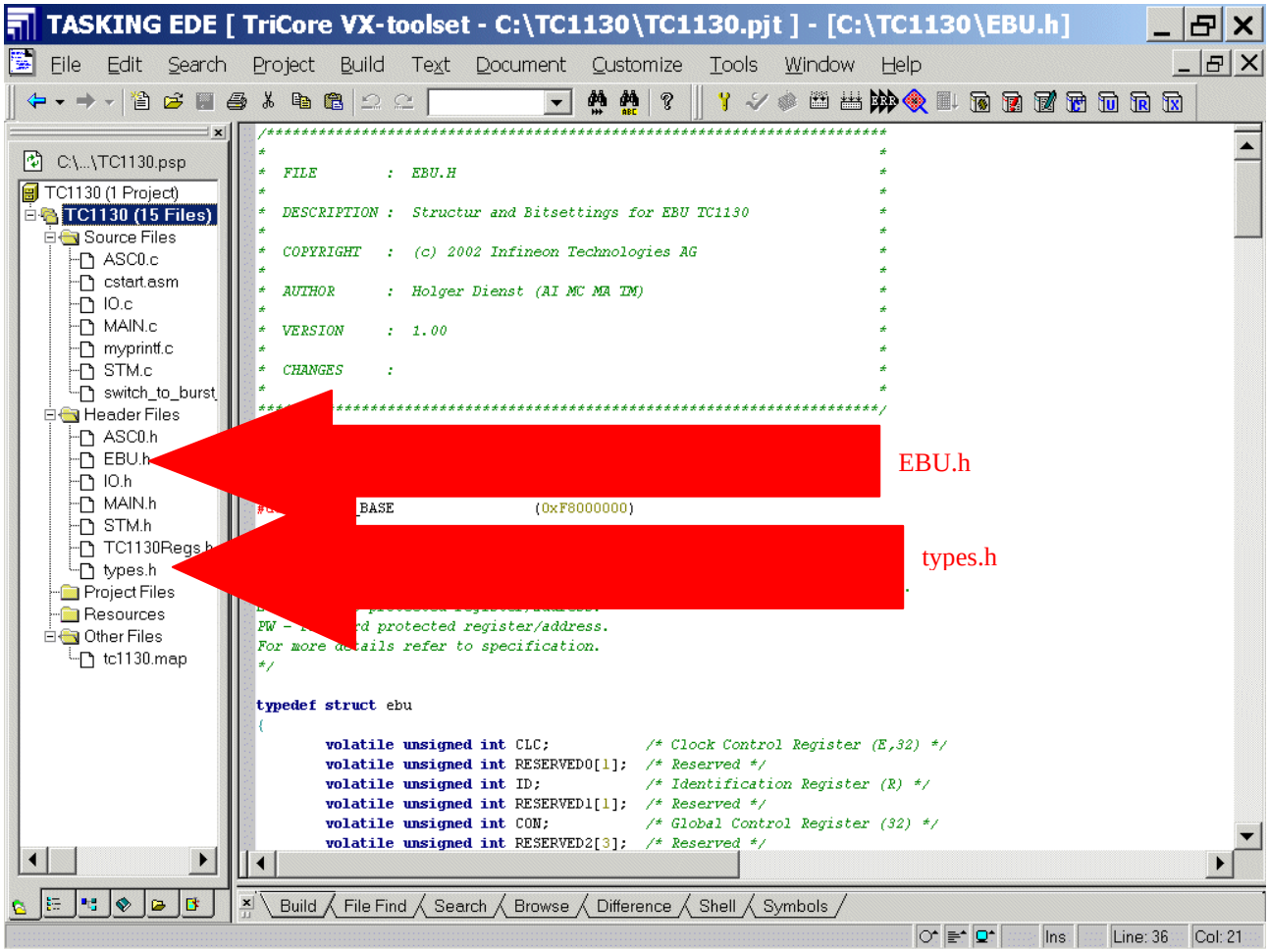
File – Save

(Project Window File View) – TC1130 (Files) – right mouse button click – Add Existing Files – Browse



Select EBU.h
Select types.h

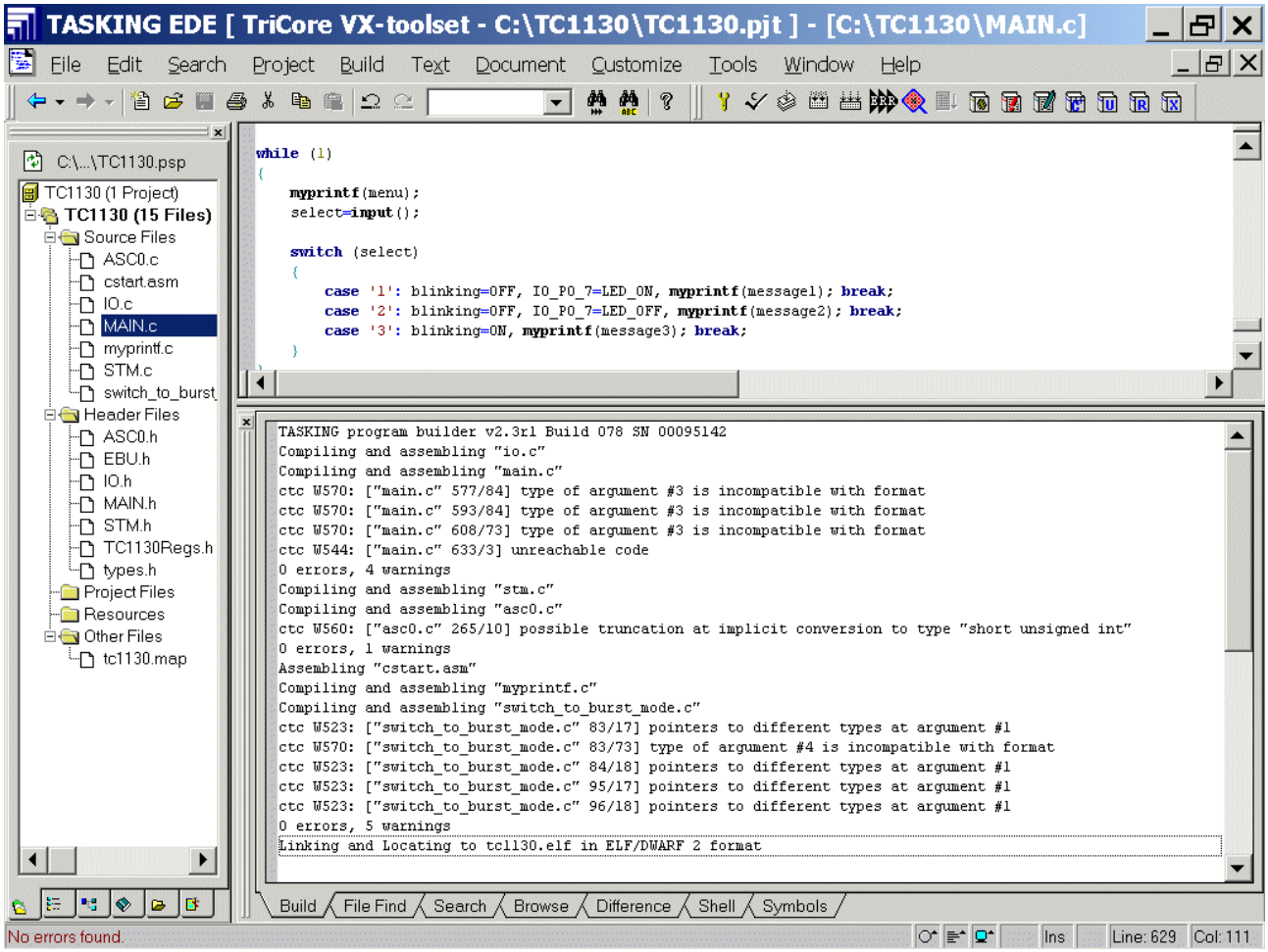
Open
OK


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\EBU.h]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\TC1130.psp
TC1130 (1 Project)
TC1130 (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
TC1130Regs.h
types.h
Project Files
Resources
Other Files
tc1130.map
EBU.h
types.h
Build File Find Search Browse Difference Shell Symbols
Ins Line: 36 Col: 21

Generate your application program:

Build
Rebuild

or



Now you can close both 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:\TC1130
File name: select TC1130.wsp

Open
Cancel

File – Load Program

Look in: select TC1130
File name: select TC1130.elf

Open

Click Program All

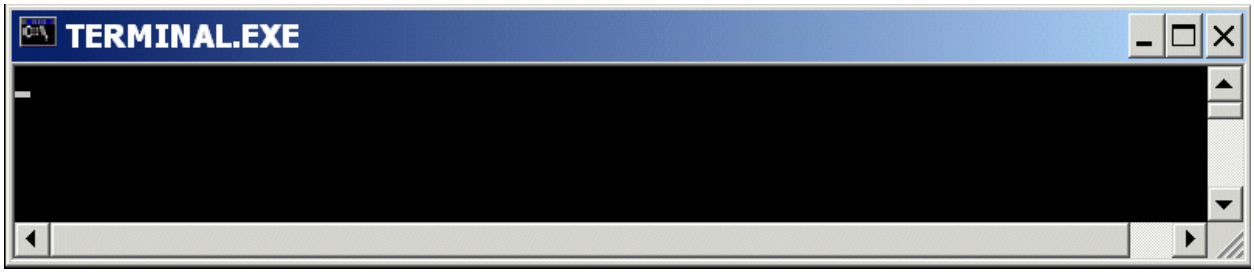
Exit
Exit

File – Close Workspace

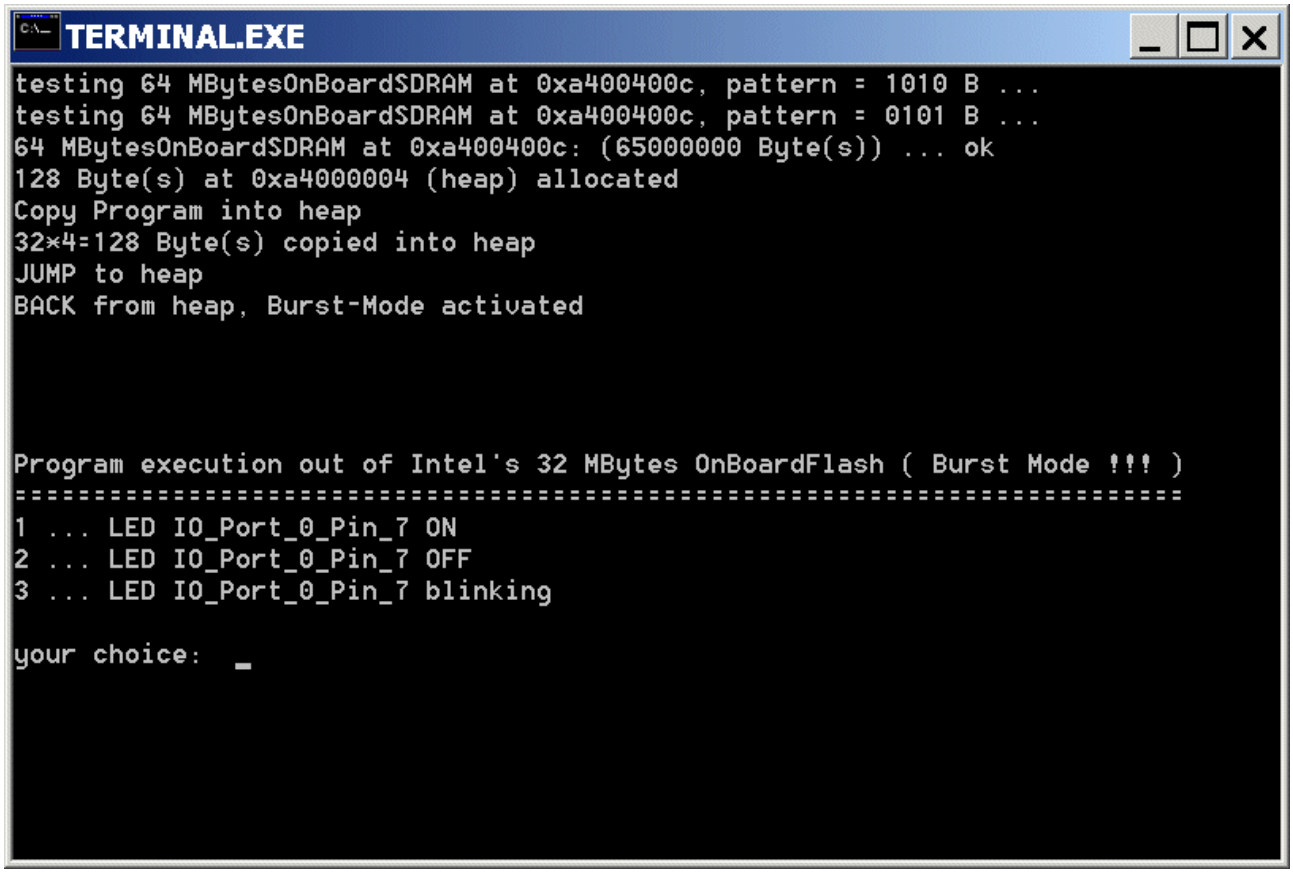
Yes

File – Exit

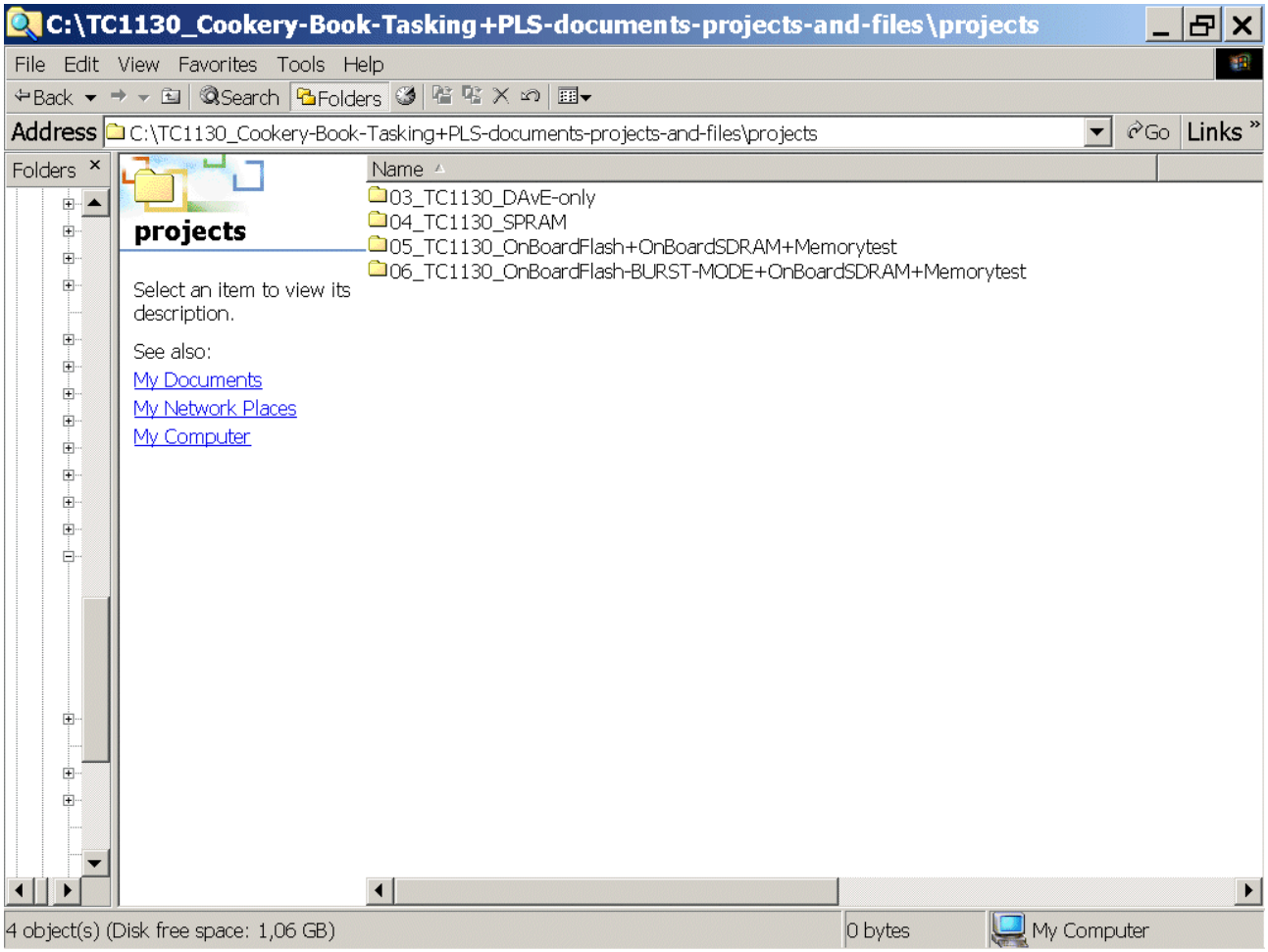
Execute any terminal-program
(9600 Baud, 8 bit Data, no Parity-Bit, 1 Stop-Bit, Xon/Xoff Protocol):



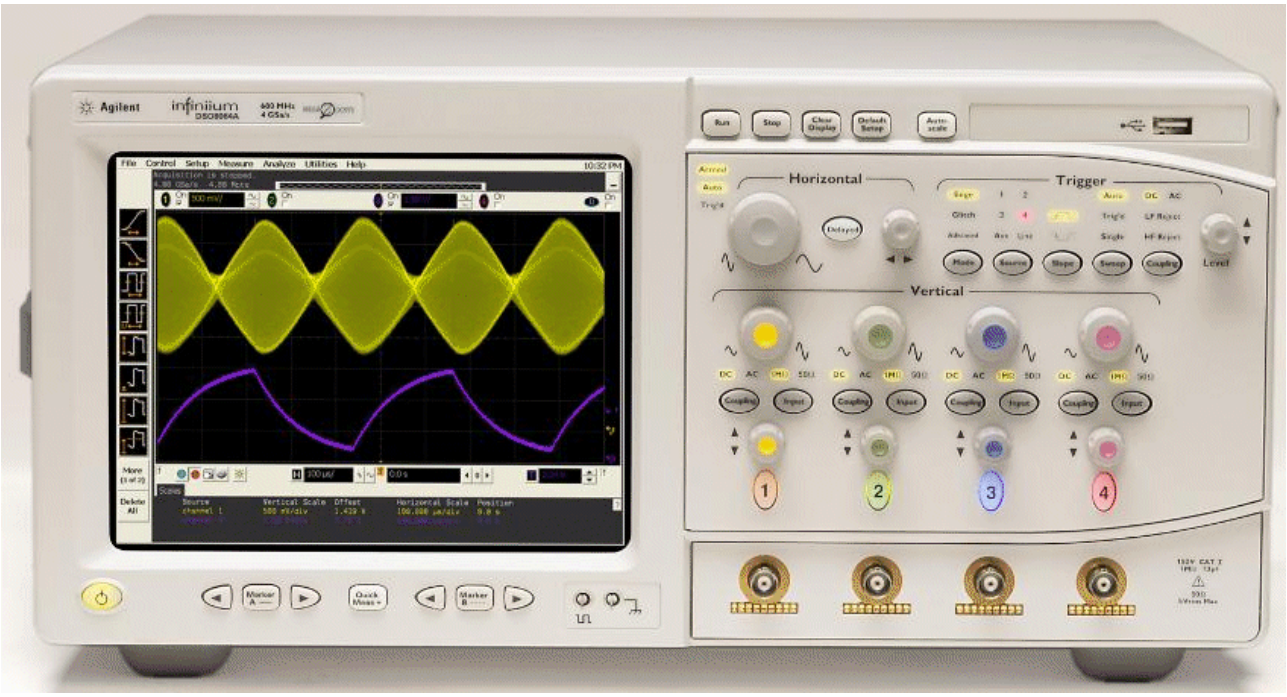
Power-On the Board and see the result:



We recommend now to copy and store your project-directory “C:\TC1130” to “06_TC1130_OnBoardFlash-BURST-MODE+OnBoardSDRAM+Memorytest”:



7.) Time – Measurement (Using an oscilloscope / a logic analyzer):



Insert application specific program ("time-measurement", see below) into the following programming examples:

Chapter 4: Program_Execution_From_PMI_Scratch-Pad-RAM (PMI_SPRAM)
Chapter 5: Program_Execution_From_OnBoardProgramFlash
Chapter 6: 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 Local Variables into "void main (void)"

```
register unsigned int i = 0;
register 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_P0_7);
  // time-consuming, dummy operations
  for (i=0;i<=100000;i++)
  {
    for (j=2;j<=5;j++)
     {
       __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_Port_0_Pin_7” is a “value” for the speed of the application:

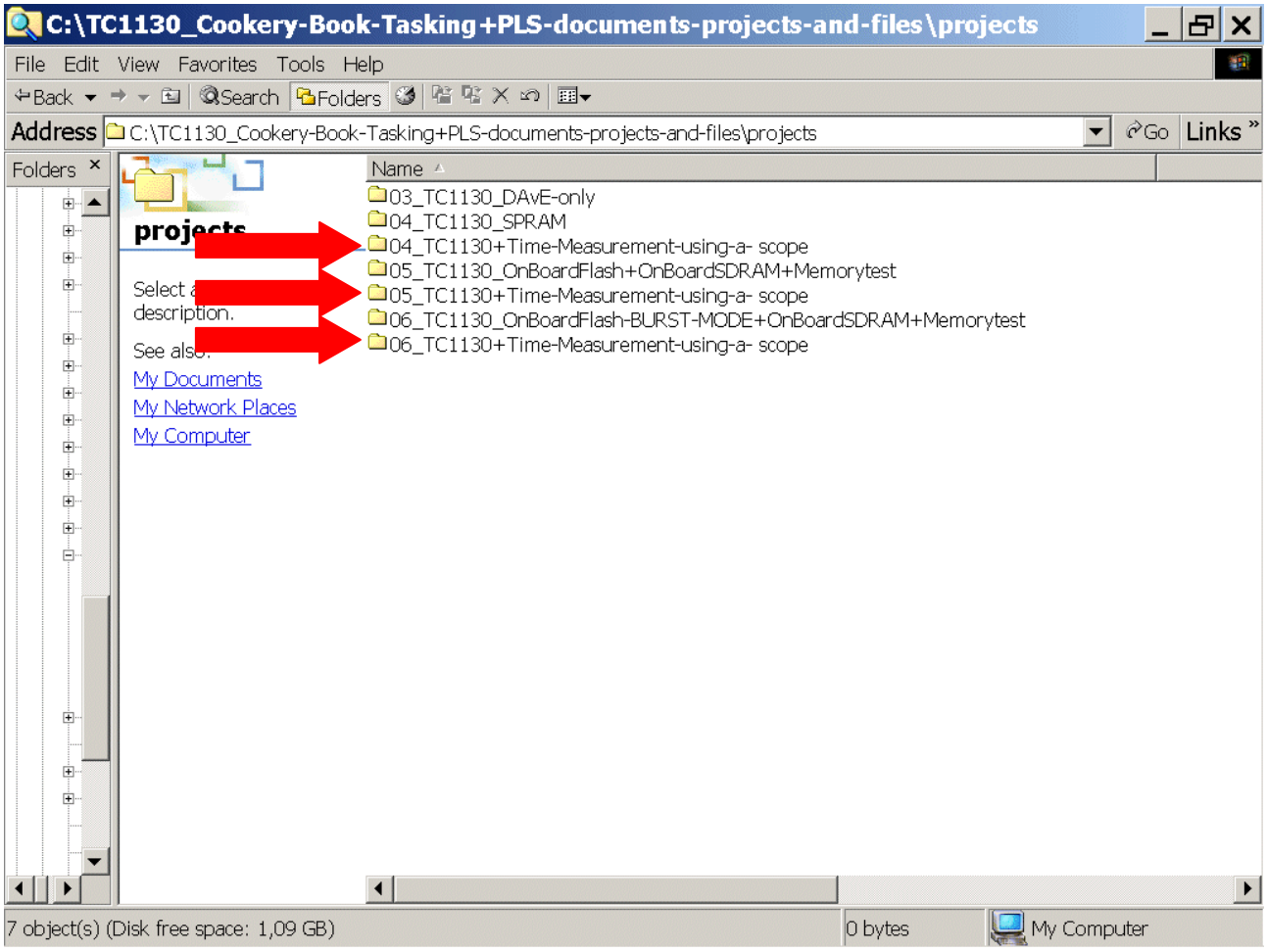
Program execution sorted by speed:

Program execution out of	Time [s] *1	Frequency [Hz] *2
PMI_SPRAM	0,104	9,60
OnBoardFlash (Burst Mode)	1,086	0,92
OnBoardFlash	2,265	0,44

Speed

*1 … lower is better
*2 … higher is better

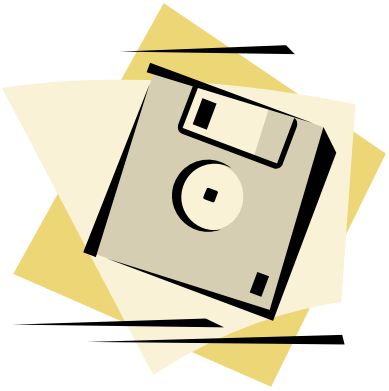
We recommend now to copy and store your project-directories “C:\TC1130” to
“04_TC1130+Time-Measurement-using-a- scope” and
“05_TC1130+Time-Measurement-using-a- scope” and
“06_TC1130+Time-Measurement-using-a- scope”:



8.) Time – Measurement [Using the SystemTimer (STM)]

8.1.) Creating a Software-Clock Using the STM-Interrupt

Note:
We are going to use the “Software-Clock Using the STM-Interrupt” to check everything concerning “time-measurement” in chapter 8.2.



We recommend now to copy and store
“06_TC1130_OnBoardFlash-BURST-MODE+OnBoardSDRAM+Memorytest”
to your working project-directory “C:\TC1130”:

Start Tasking EDE and open the project:

File – Open Project Space

Look in: select C:\TC1130
File name: select TC1130.psp

Open

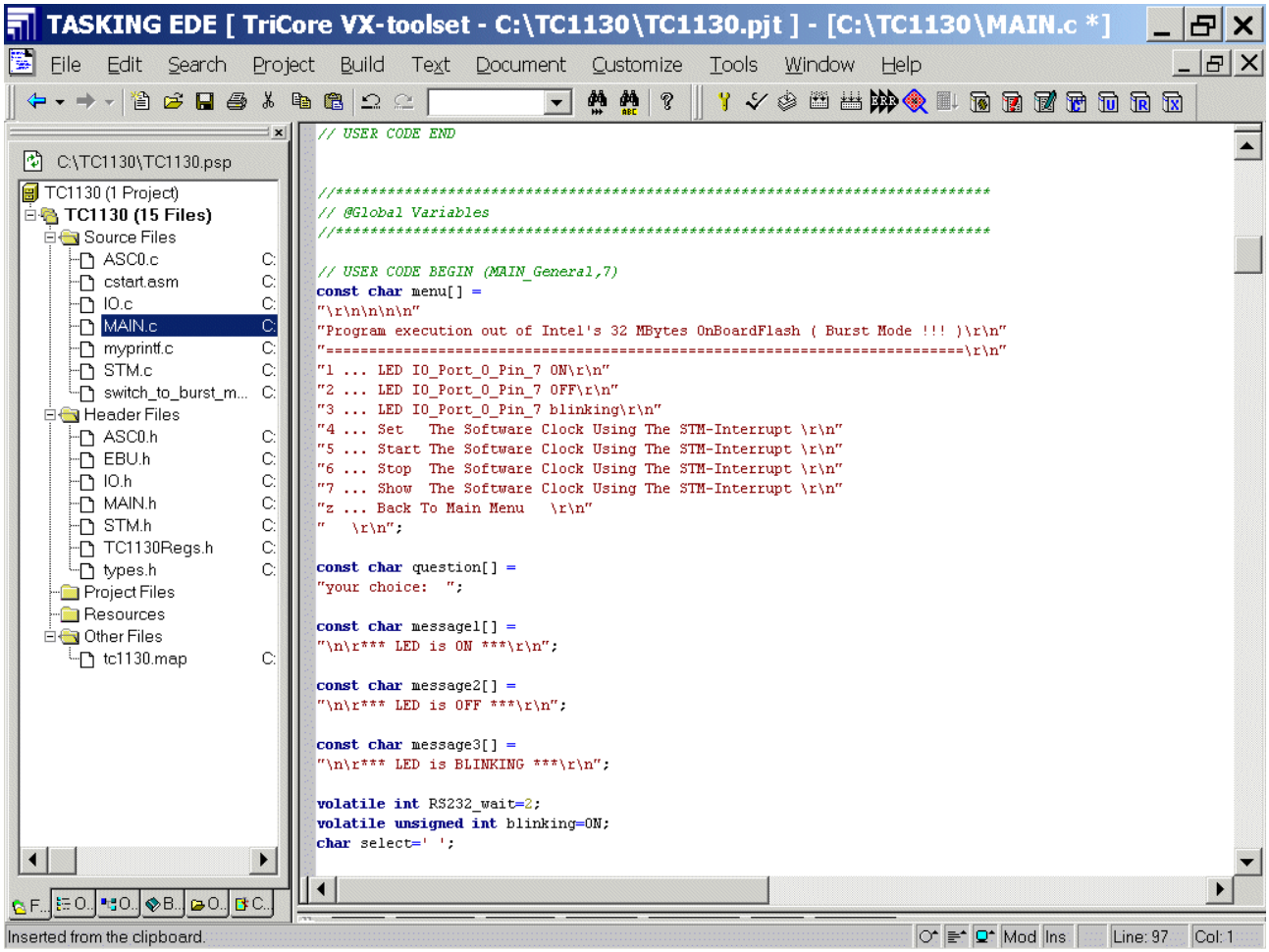
Insert your application specific program:

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

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of Intel's 32 MBytes OnBoardFlash ( Burst
Mode !!! )\r\n"
"================================================================
========\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 blinking\r\n"
"   \r\n";
```

to

```
const char menu[] =
"\r\n\n\n\n"
"Program execution out of Intel's 32 MBytes OnBoardFlash ( Burst
Mode !!! )\r\n"
"================================================================
========\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 blinking\r\n"
"4 ... Set   The Software Clock Using The STM-Interrupt \r\n"
"5 ... Start The Software Clock Using The STM-Interrupt \r\n"
"6 ... Stop  The Software Clock Using The STM-Interrupt \r\n"
"7 ... Show  The Software Clock Using The STM-Interrupt \r\n"
"z ... Back To Main Menu   \r\n"
"   \r\n";
```

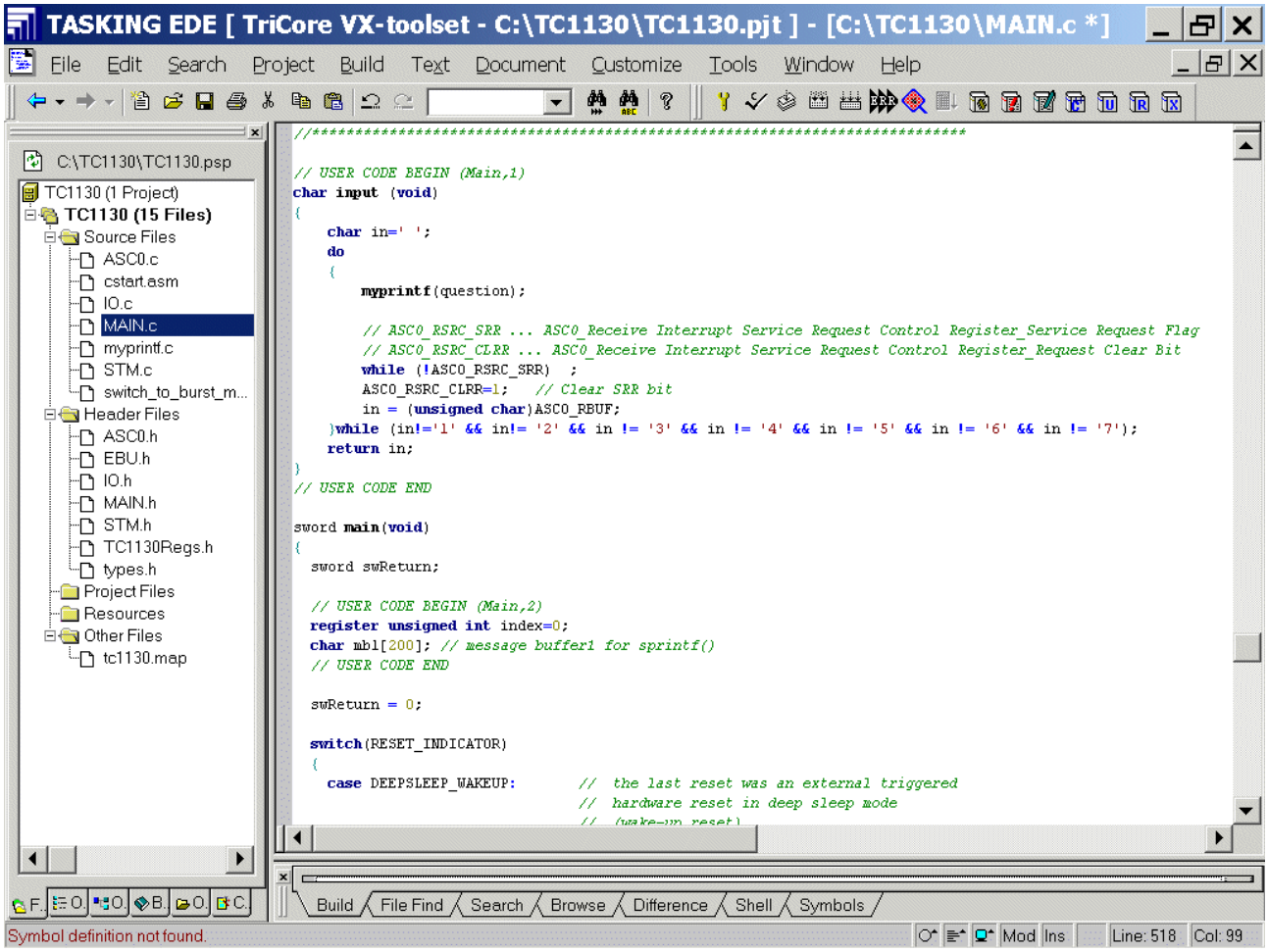

TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1130\TC1130.psp
TC1130 (1 Project)
TC1130 (15 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
STM.c
switch_to_burst_m...
Header Files
ASC0.h
EBU.h
IO.h
MAIN.h
STM.h
TC1130Regs.h
types.h
Project Files
Resources
Other Files
tc1130.map
// USER CODE END
//***
// @Global Variables
//***
// USER CODE BEGIN (MAIN_General,7)
const char menu[] =
"\r\n\n\n\n"
"Program execution out of Intel's 32 MBytes OnBoardFlash (Burst Mode !!!)\r\n"
"==\r\n"
"1 ... LED IO_Port_0_Pin_7 ON\r\n"
"2 ... LED IO_Port_0_Pin_7 OFF\r\n"
"3 ... LED IO_Port_0_Pin_7 blinking\r\n"
"4 ... Set The Software Clock Using The STM-Interrupt \r\n"
"5 ... Start The Software Clock Using The STM-Interrupt \r\n"
"6 ... Stop The Software Clock Using The STM-Interrupt \r\n"
"7 ... Show The Software Clock Using The STM-Interrupt \r\n"
"z ... Back To Main Menu \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=' ';
Inserted from the clipboard.
Mod Ins Line: 97 Col: 1

Double click: Main.c and change Code [“char input (void)” - Function]
From

```
}while (in!='1' && in!= '2' && in != '3');
```

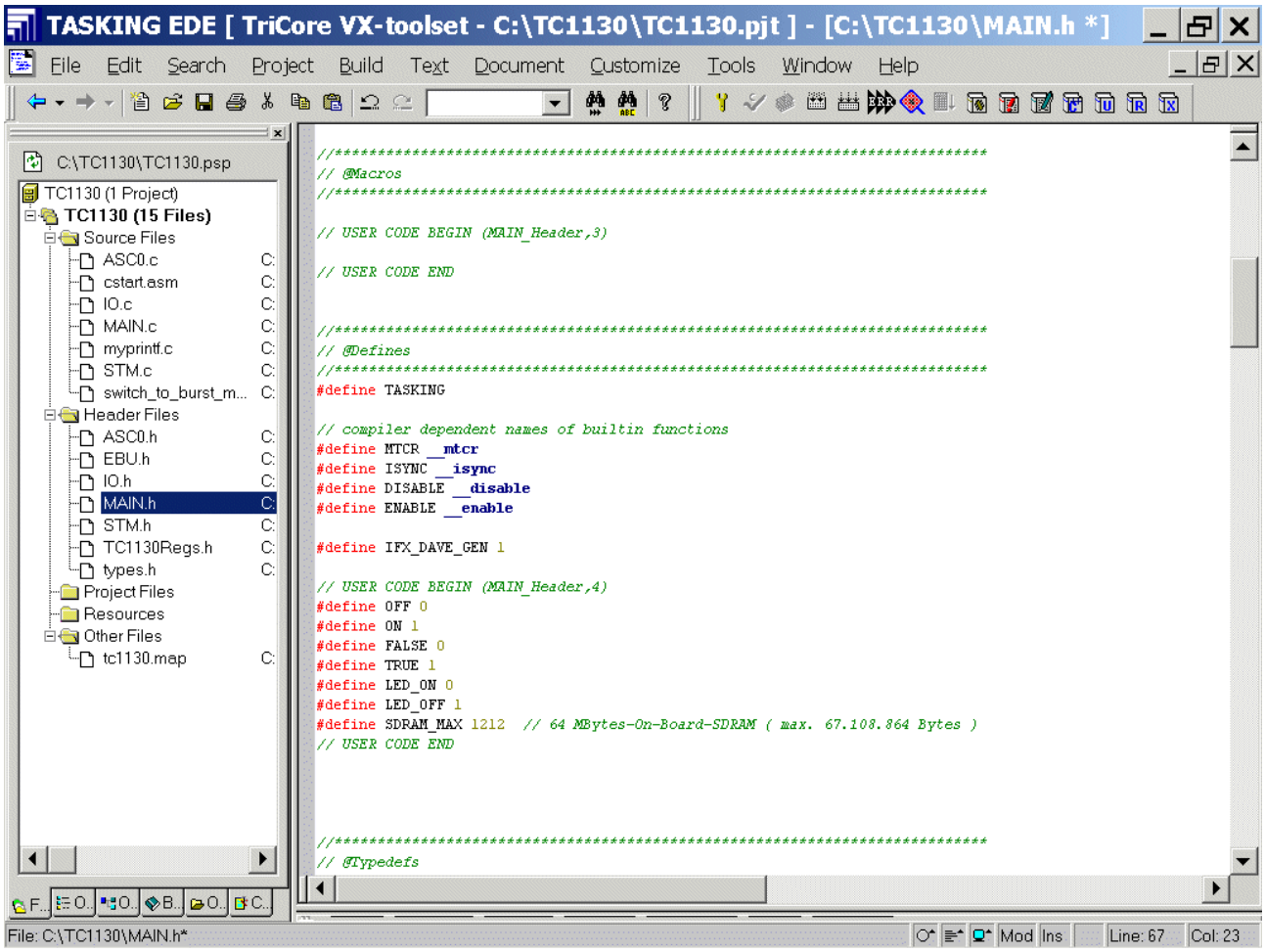
To

```
}while (in!='1' && in!= '2' && in != '3' && in != '4' && in != '5'
&& in != '6' && in != '7');
```



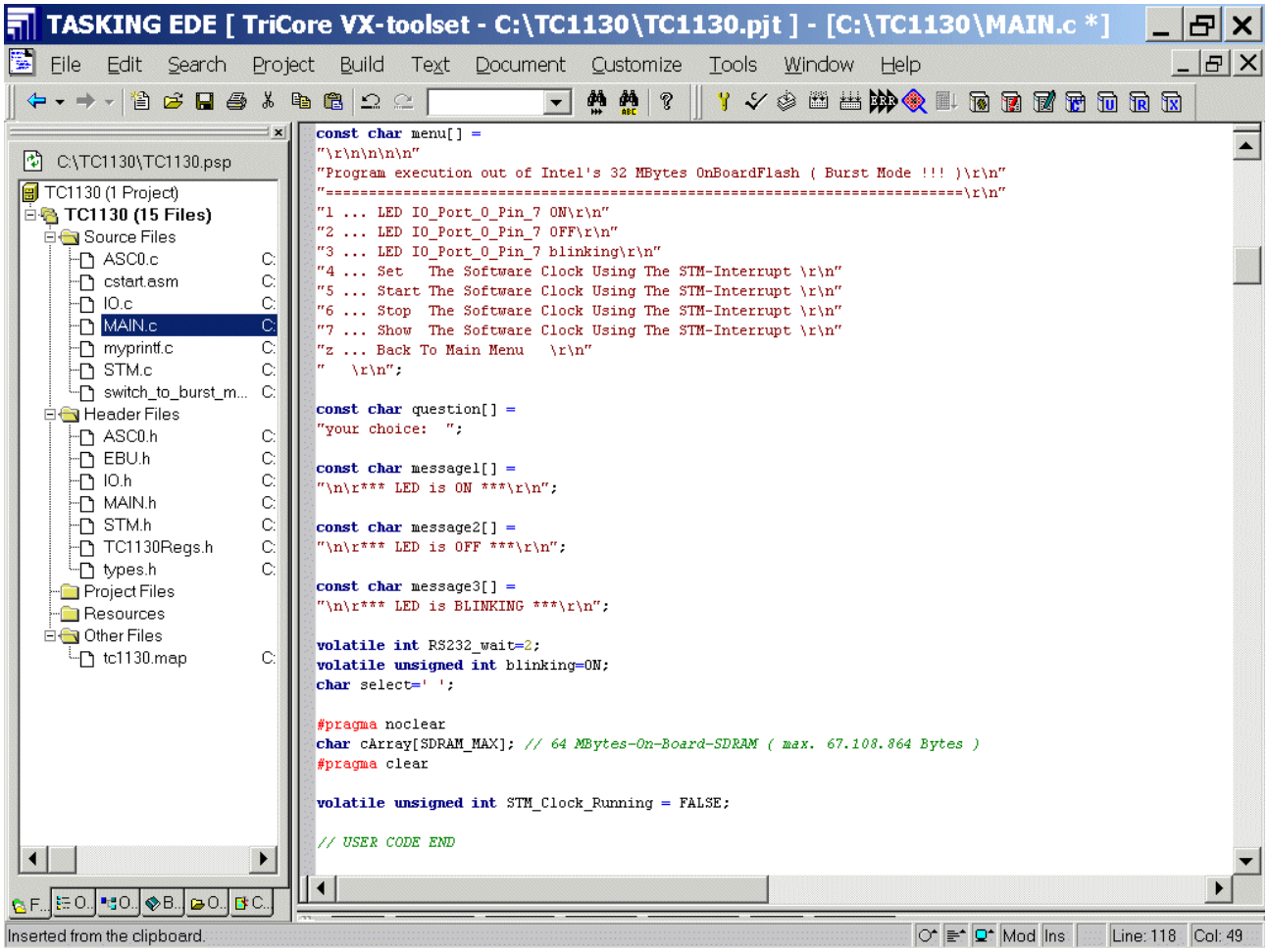
Double click: Main.h and insert the following Defines:

```
#define FALSE 0
#define TRUE 1
```



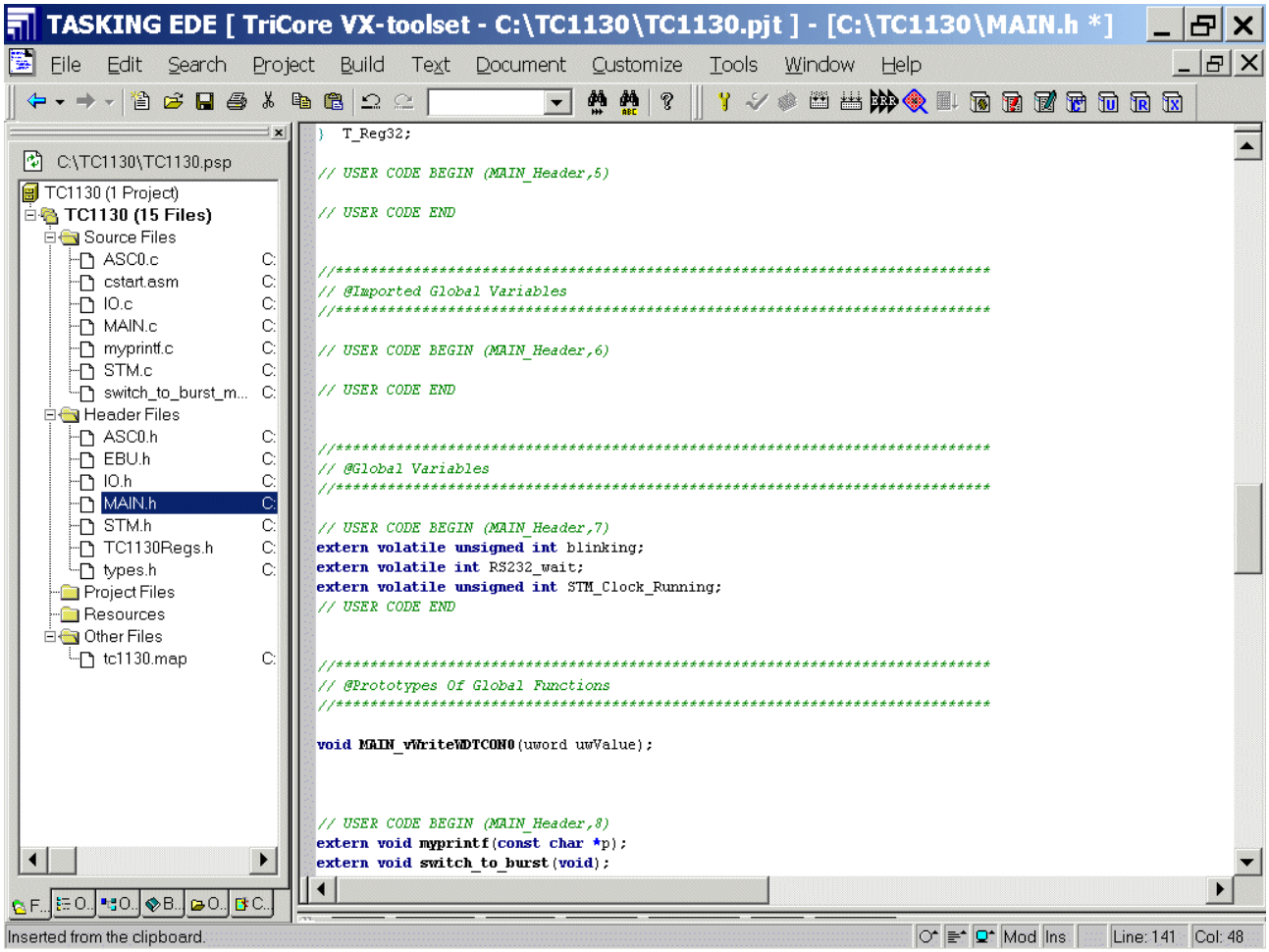
Double click: Main.c insert User Code (Global Variables):

```
volatile unsigned int STM Clock Running = FALSE;
```



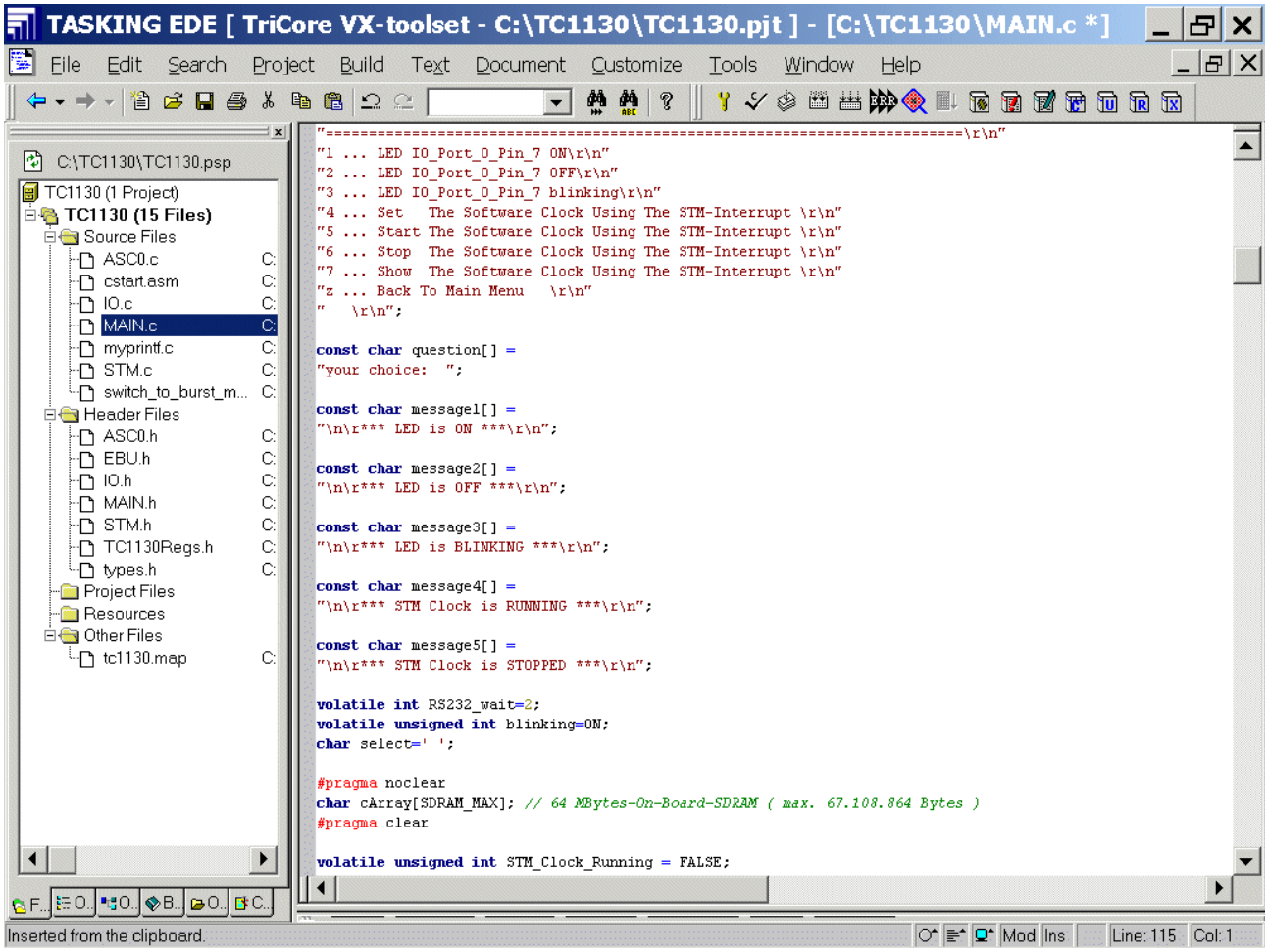
Double click: Main.h and insert Global Variables (Extern Declarations):

```
extern volatile unsigned int STM Clock Running;
```



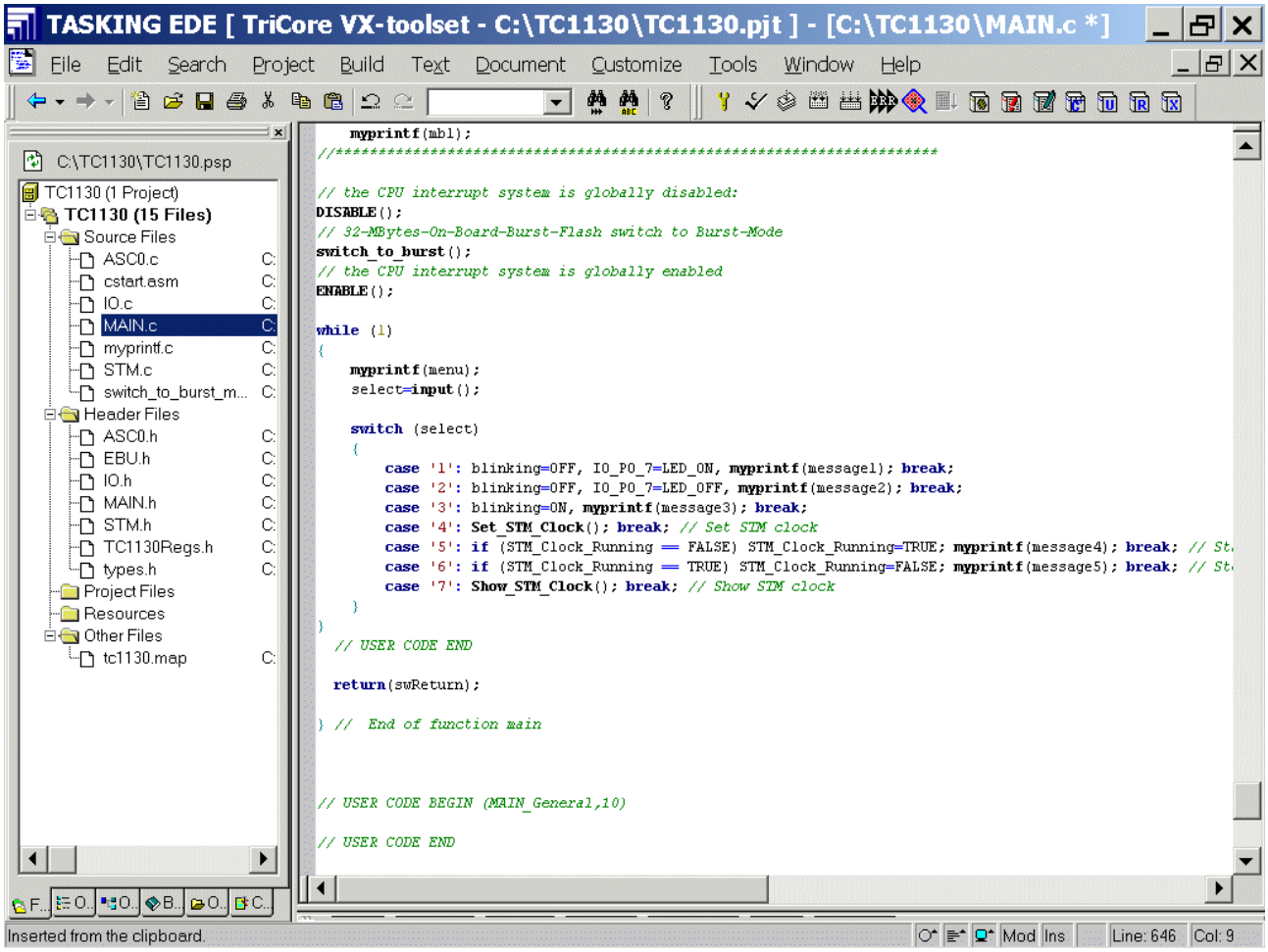
Double click: Main.c insert User Code (Global Variables):

```
const char message4[] =
"\n\r*** STM Clock is RUNNING ***\r\n";

const char message5[] =
"\n\r*** STM Clock is STOPPED ***\r\n";
```



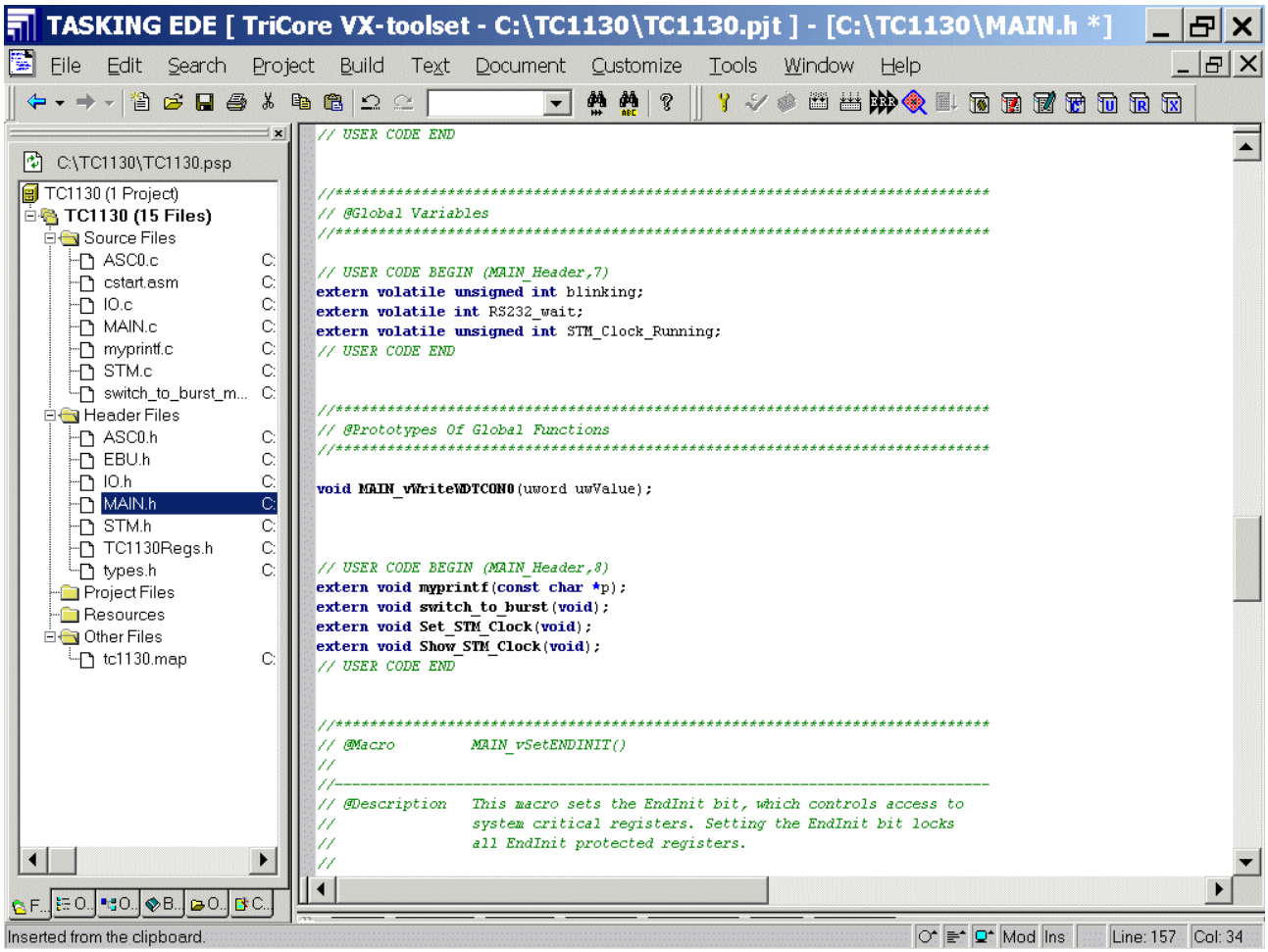
Double click: Main.c insert User Code [void main (void) - Function]:

```
          case '4': Set_STM_Clock(); break; // Set STM clock
          case '5': if (STM_Clock_Running == FALSE)
STM_Clock_Running=TRUE; myprintf(message4); break; // Start STM
clock
          case '6': if (STM_Clock_Running == TRUE)
STM_Clock_Running=FALSE; myprintf(message5); break; // Stop STM
clock
          case '7': Show_STM_Clock(); break; // Show STM clock
```



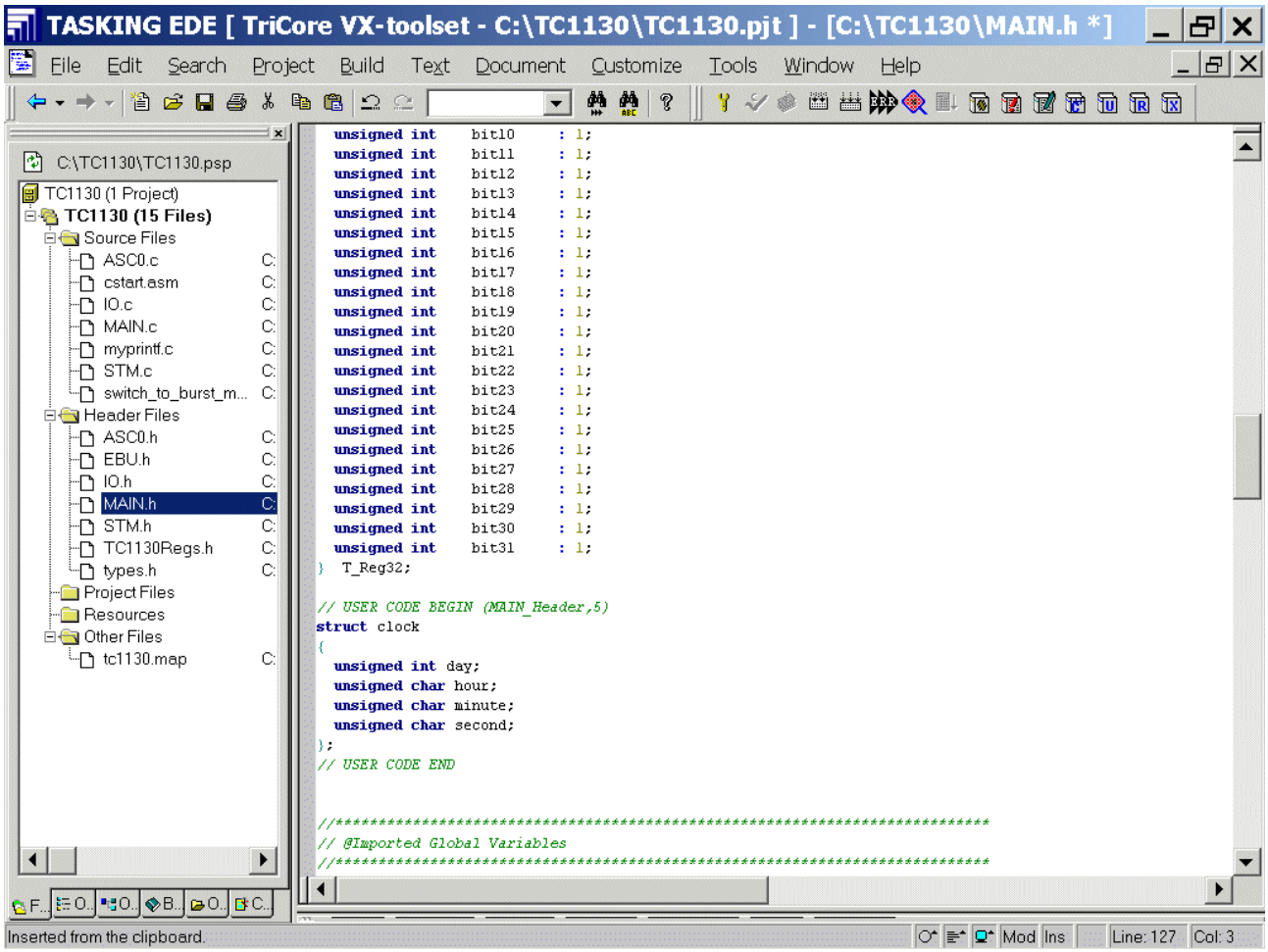
Double click: Main.h and insert Prototypes of Global Functions (Extern Declaration):

```
extern void Set_STM_Clock(void);
extern void Show_STM_Clock(void);
```



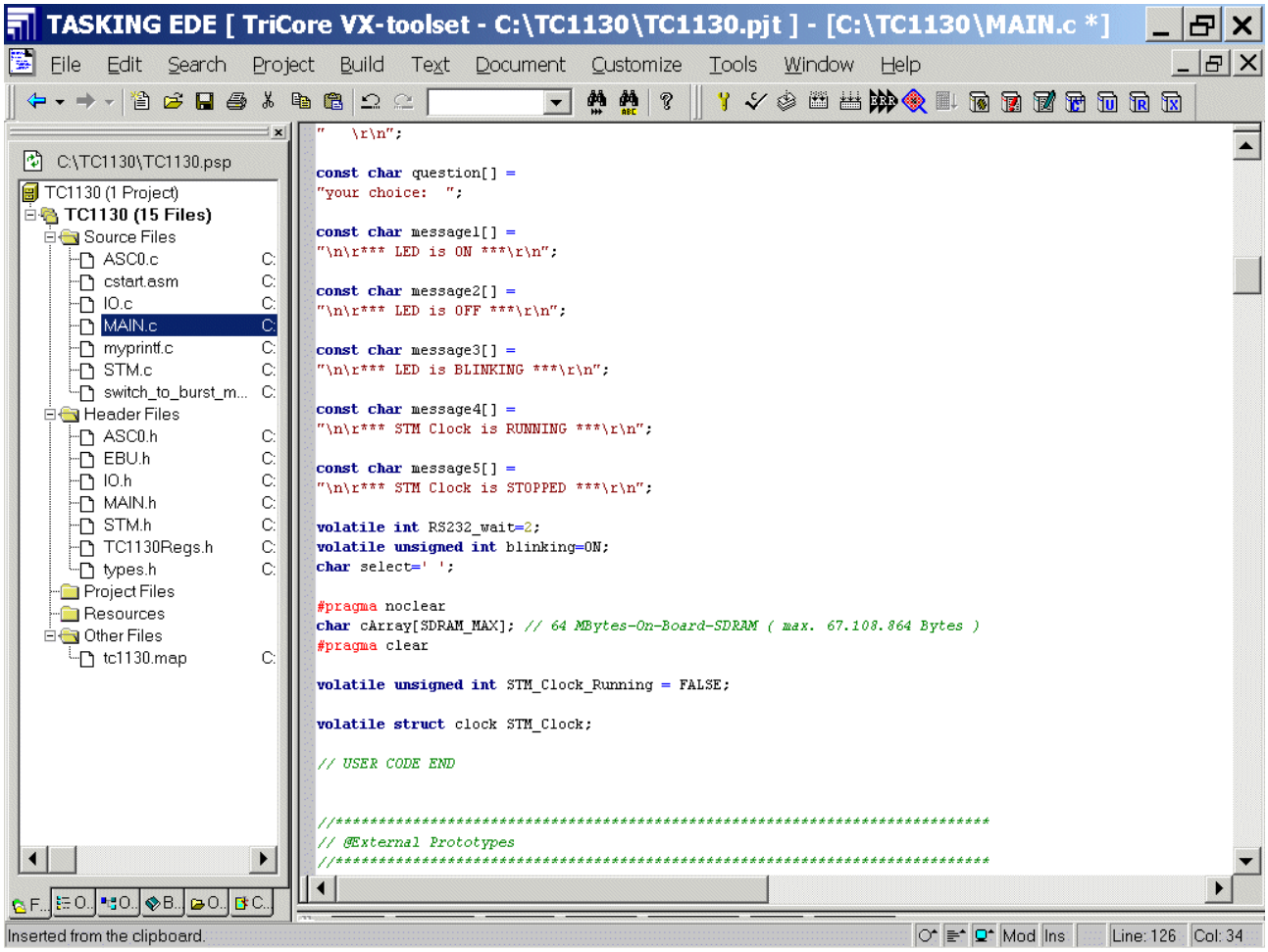
Double click: Main.h and insert Typedefs (struct Definition):

```
struct clock
{
  unsigned int day;
  unsigned char hour;
  unsigned char minute;
  unsigned char second;
};
```



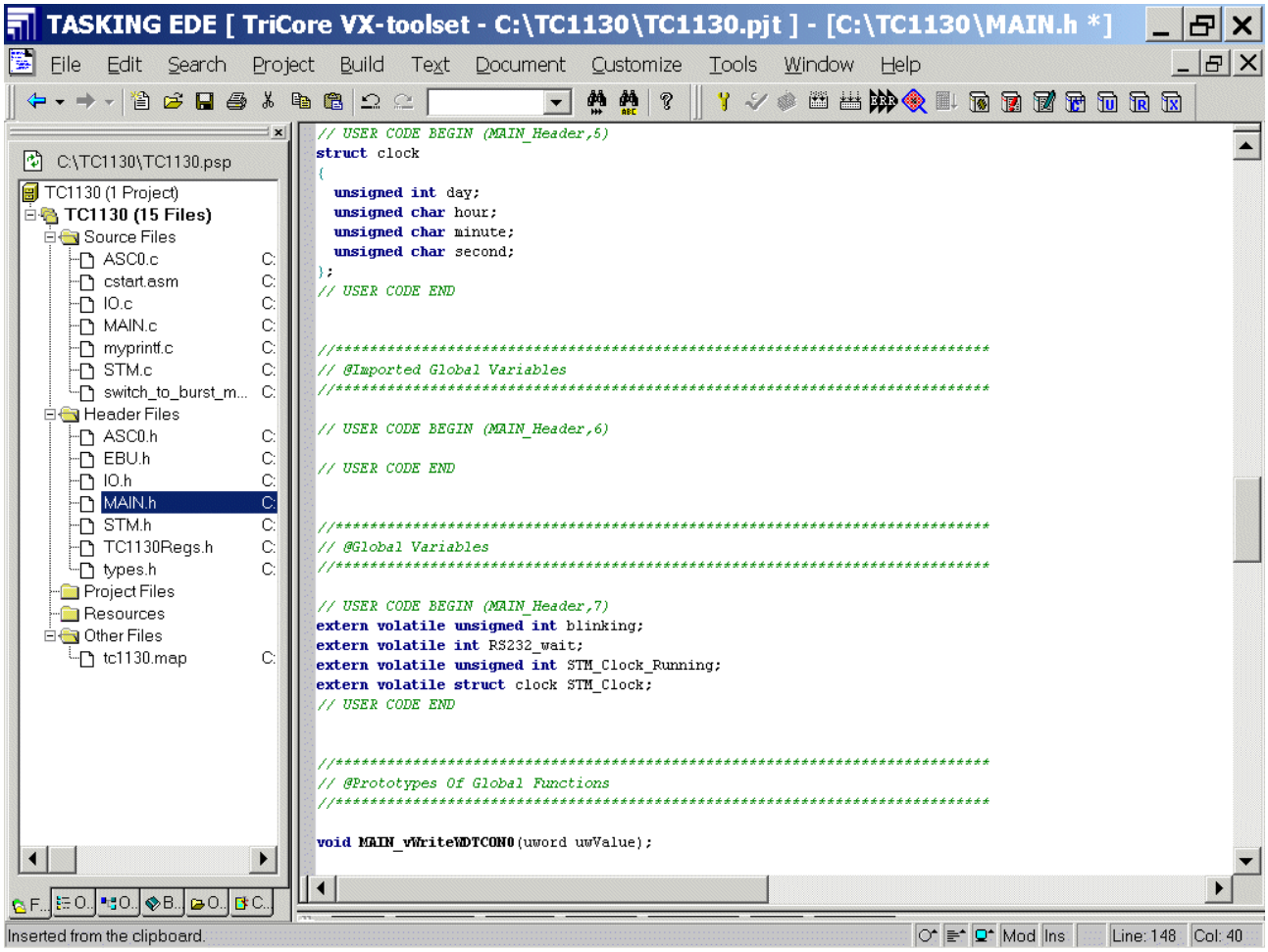
Double click: Main.c insert User Code (Global Variables):

```
volatile struct clock STM_Clock;
```

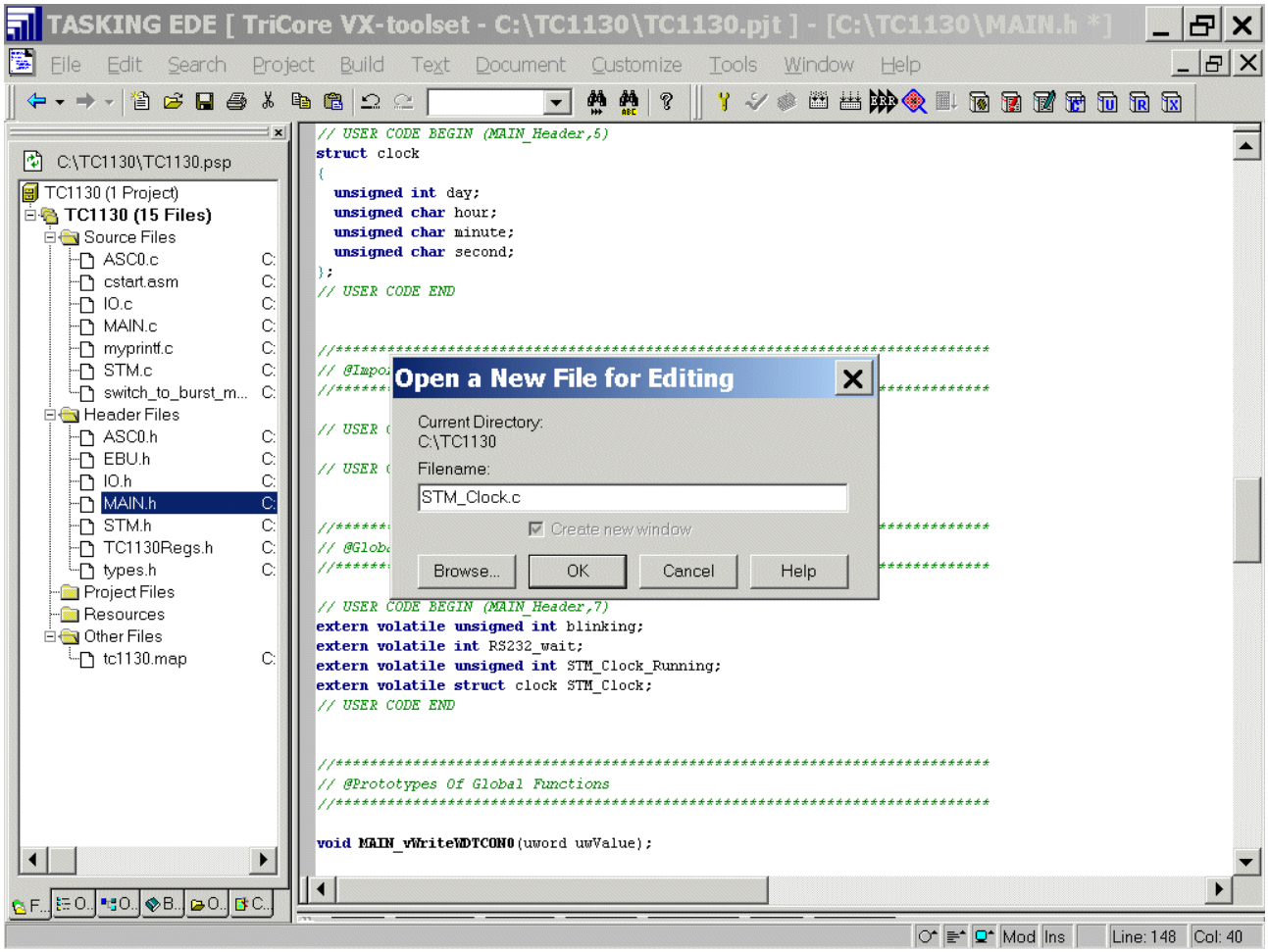


Double click: Main.h and insert Extern Declaration:

```
extern volatile struct clock STM_Clock;
```



File – New
Insert STM_Clock.c



OK

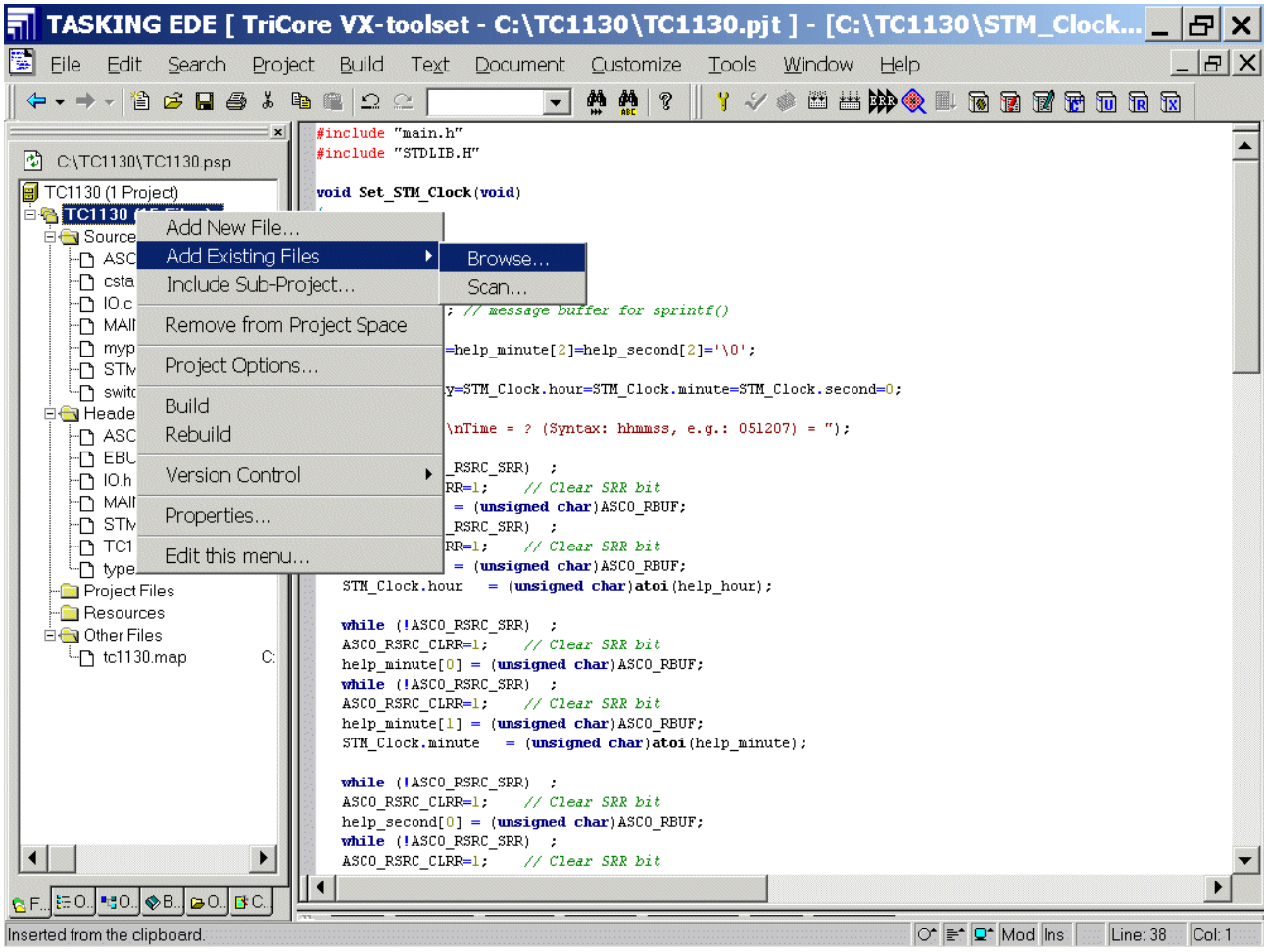
Insert User Code:

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

void Set_STM_Clock(void)
{
   char in=' ';
   char help_hour[3];
   char help_minute[3];
   char help_second[3];
   char mb[200]; // message buffer for sprintf()

   help_hour[2]=help_minute[2]=help_second[2]='\0';


STM_Clock.day=STM_Clock.hour=STM_Clock.minute=STM_Clock.second=0;

   myprintf("\r\nTime = ? (Syntax: hhmmss, e.g.: 051207) = ");

   while (!ASC0_RSRC_SRR)  ;
   ASC0_RSRC_CLRR=1;     // Clear SRR bit
   help_hour[0] = (unsigned char)ASC0_RBUF;
   while (!ASC0_RSRC_SRR)  ;
   ASC0_RSRC_CLRR=1;     // Clear SRR bit
   help_hour[1] = (unsigned char)ASC0_RBUF;
   STM_Clock.hour   = (unsigned char)atoi(help_hour);

   while (!ASC0_RSRC_SRR)  ;
   ASC0_RSRC_CLRR=1;     // Clear SRR bit
   help_minute[0] = (unsigned char)ASC0_RBUF;
   while (!ASC0_RSRC_SRR)  ;
   ASC0_RSRC_CLRR=1;     // Clear SRR bit
   help_minute[1] = (unsigned char)ASC0_RBUF;
   STM_Clock.minute   = (unsigned char)atoi(help_minute);

   while (!ASC0_RSRC_SRR)  ;
   ASC0_RSRC_CLRR=1;     // Clear SRR bit
   help_second[0] = (unsigned char)ASC0_RBUF;
   while (!ASC0_RSRC_SRR)  ;
   ASC0_RSRC_CLRR=1;     // Clear SRR bit
   help_second[1] = (unsigned char)ASC0_RBUF;
   STM_Clock.second   = (unsigned char)atoi(help_second);

   sprintf(mb,"\r\n*** Time = %02u:%02u:%02u
***\r\n",STM_Clock.hour,STM_Clock.minute,STM_Clock.second);
   myprintf(mb);
}

void Show_STM_Clock(void)
{
   char mb[200]; // message buffer for sprintf()
   unsigned char in;

     myprintf("\r\n\n***** TIME *****\r\n");
     do
```

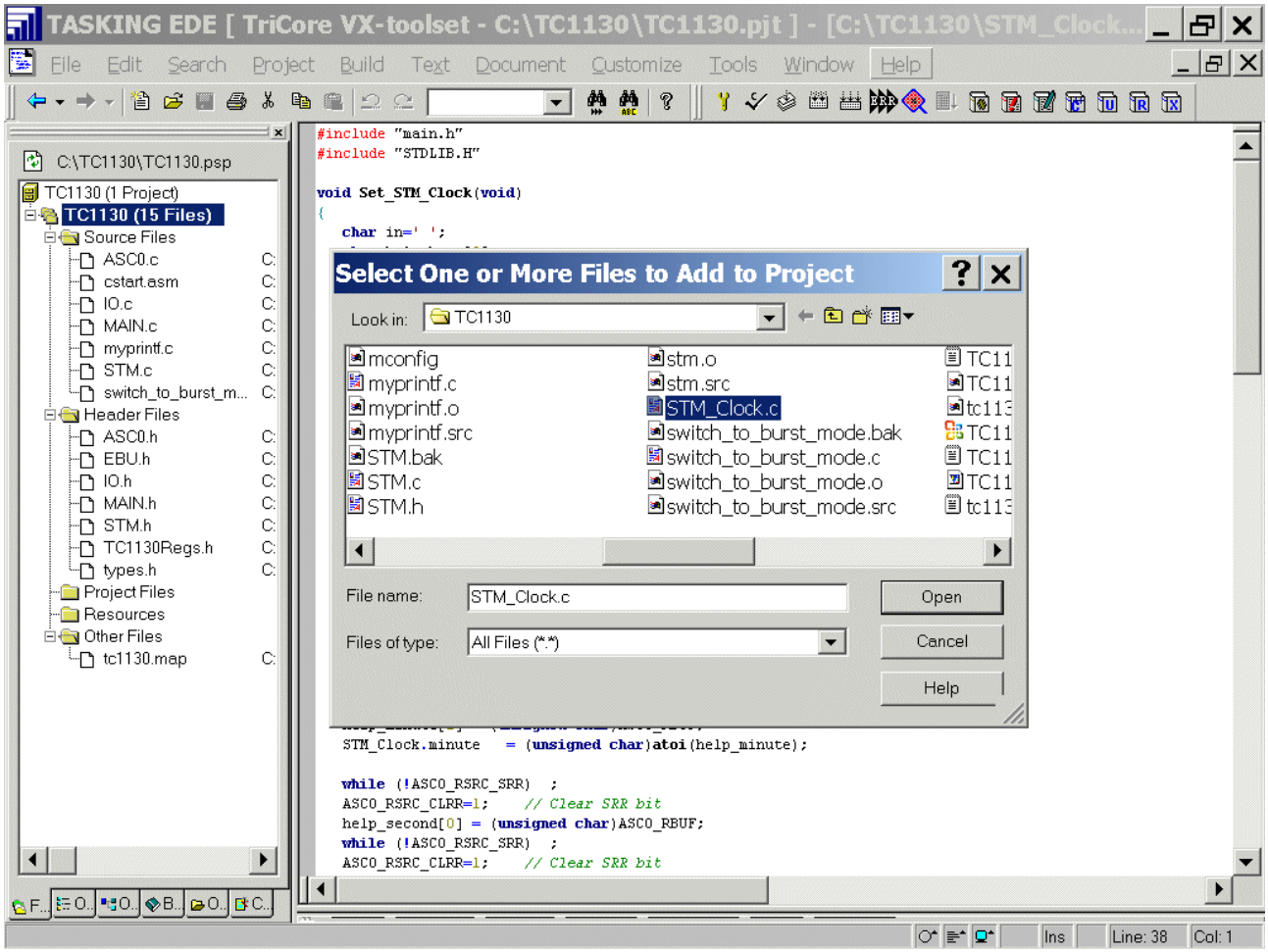
```
      {
        sprintf(mb,"*** %02u:%02u:%02u
***\r",STM_Clock.hour,STM_Clock.minute,STM_Clock.second);
        myprintf(mb);

        in = (unsigned char)ASC0_RBUF;
     }while (in!='z' && in!= 'Z');
}
```

File
Save all

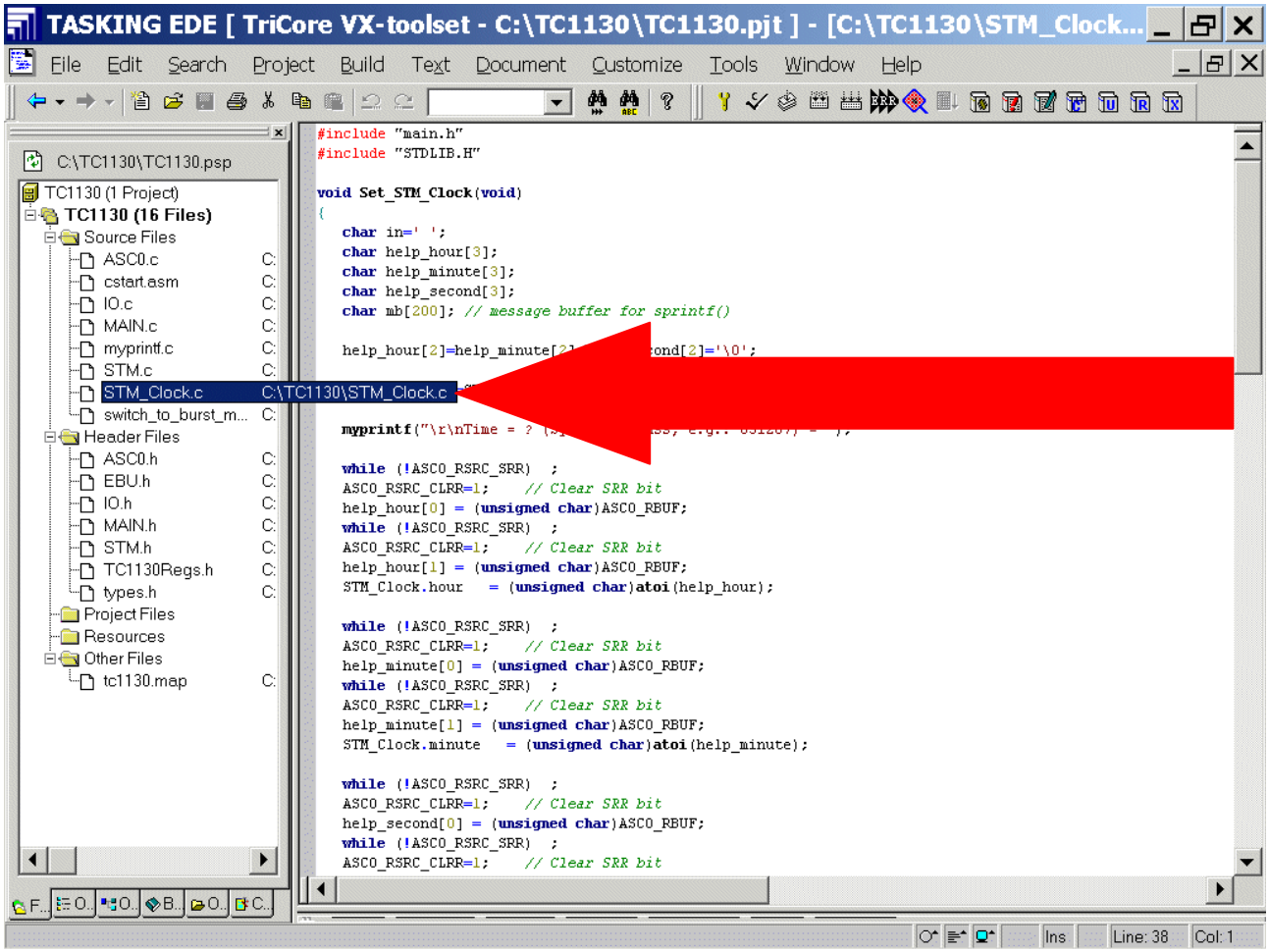
(Project Window File View) – TC1130 (Files) – right mouse button click – Add Existing Files – Browse



Select STM_Clock.c

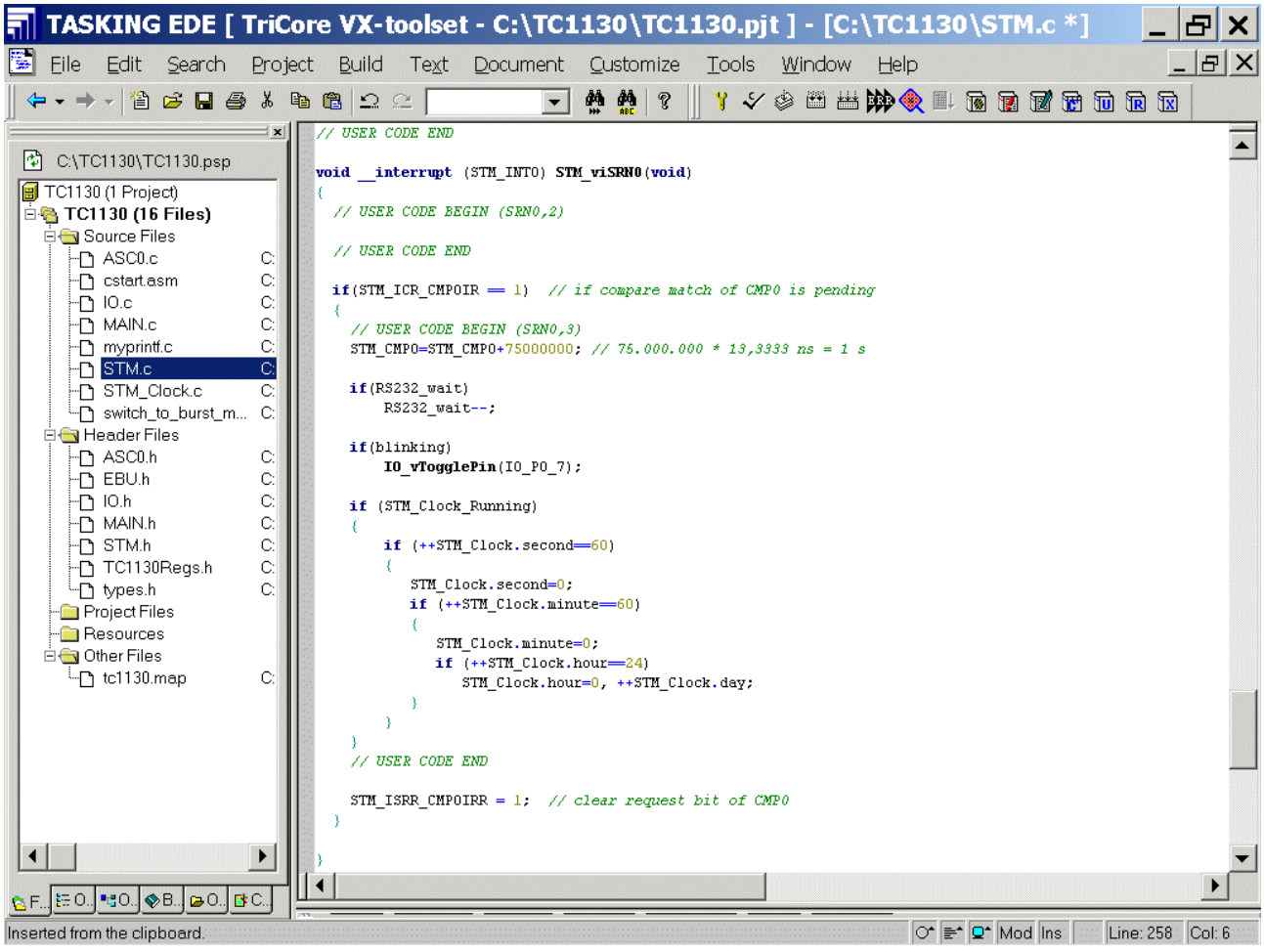


Open - OK


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\STM_Clock...
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1130\TC1130.psp
TC1130 (1 Project)
TC1130 (16 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
STM.c
STM_Clock.c
C:\TC1130\STM_Clock.c
switch_to_burst_m...
Header Files
ASC0.h
EBU.h
IO.h
MAIN.h
STM.h
TC1130Regs.h
types.h
Project Files
Resources
Other Files
tc1130.map
#include "main.h"
#include "STDLIB.H"
void Set_STM_Clock(void)
{
char in=' ';
char help_hour[3];
char help_minute[3];
char help_second[3];
char mb[200]; // message buffer for sprintf()
help_hour[2]=help_minute[2]
ond[2]='\0';
myprintf("\r\nTime = ?
while (!ASC0_RSRC_SRR) ;
ASC0_RSRC_CLRR=1; // Clear SRR bit
help_hour[0] = (unsigned char)ASC0_RBUF;
while (!ASC0_RSRC_SRR) ;
ASC0_RSRC_CLRR=1; // Clear SRR bit
help_hour[1] = (unsigned char)ASC0_RBUF;
STM_Clock.hour = (unsigned char)atoi(help_hour);
while (!ASC0_RSRC_SRR) ;
ASC0_RSRC_CLRR=1; // Clear SRR bit
help_minute[0] = (unsigned char)ASC0_RBUF;
while (!ASC0_RSRC_SRR) ;
ASC0_RSRC_CLRR=1; // Clear SRR bit
help_minute[1] = (unsigned char)ASC0_RBUF;
STM_Clock.minute = (unsigned char)atoi(help_minute);
while (!ASC0_RSRC_SRR) ;
ASC0_RSRC_CLRR=1; // Clear SRR bit
help_second[0] = (unsigned char)ASC0_RBUF;
while (!ASC0_RSRC_SRR) ;
ASC0_RSRC_CLRR=1; // Clear SRR bit
Ins
Line: 38
Col: 1

Double click: STM.c insert User Code for system-timer's interrupt-service-routine:

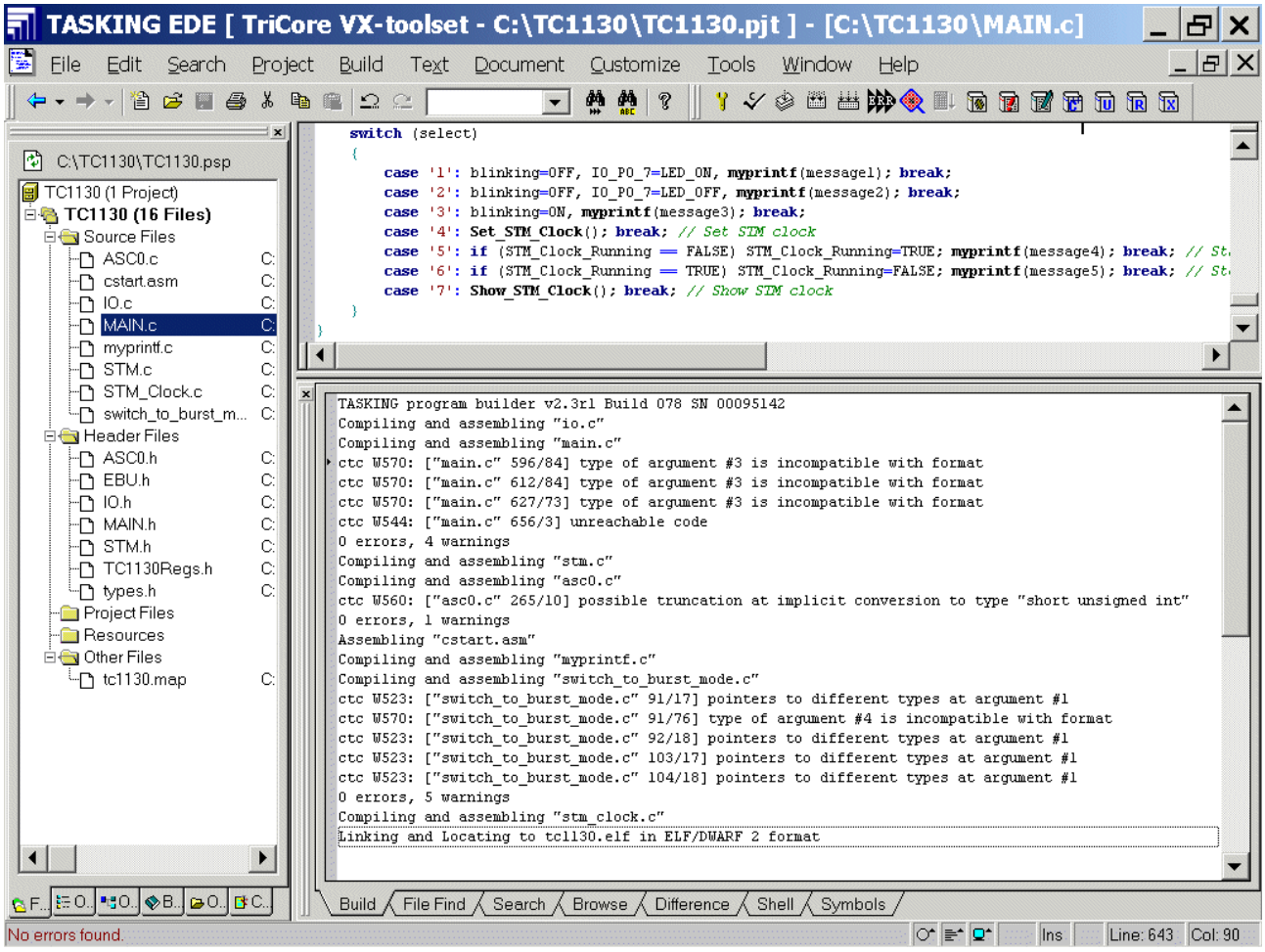
```
    if (STM_Clock_Running)
     {
        if (++STM_Clock.second==60)
        {
           STM_Clock.second=0;
           if (++STM_Clock.minute==60)
           {
              STM_Clock.minute=0;
              if (++STM_Clock.hour==24)
                 STM_Clock.hour=0, ++STM_Clock.day;
           }
        }
    }
```



Generate your application program:

Build
Rebuild

or



Now you can close both 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

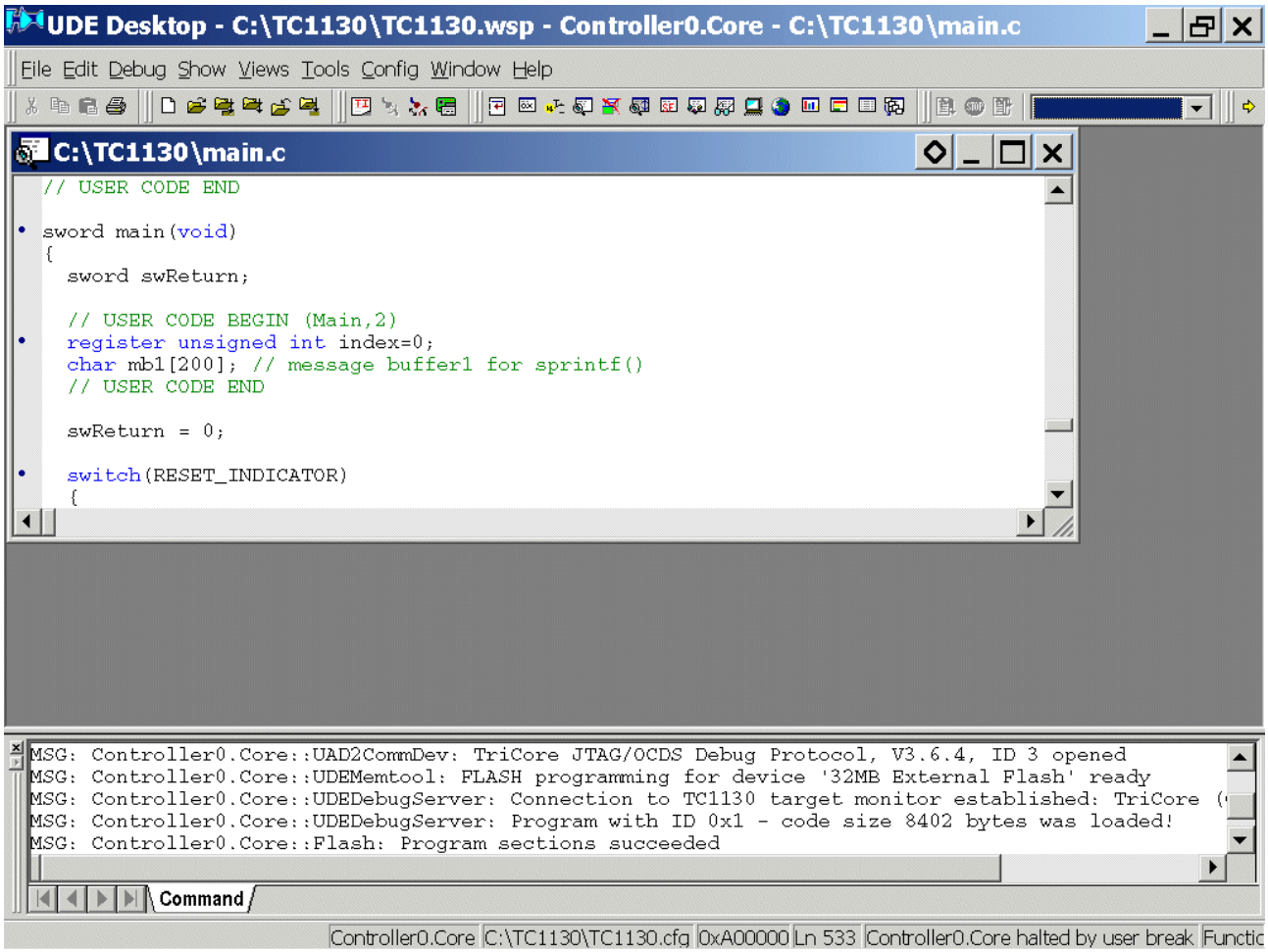
Look in: select C:\TC1130
File name: select TC1130.wsp

Open

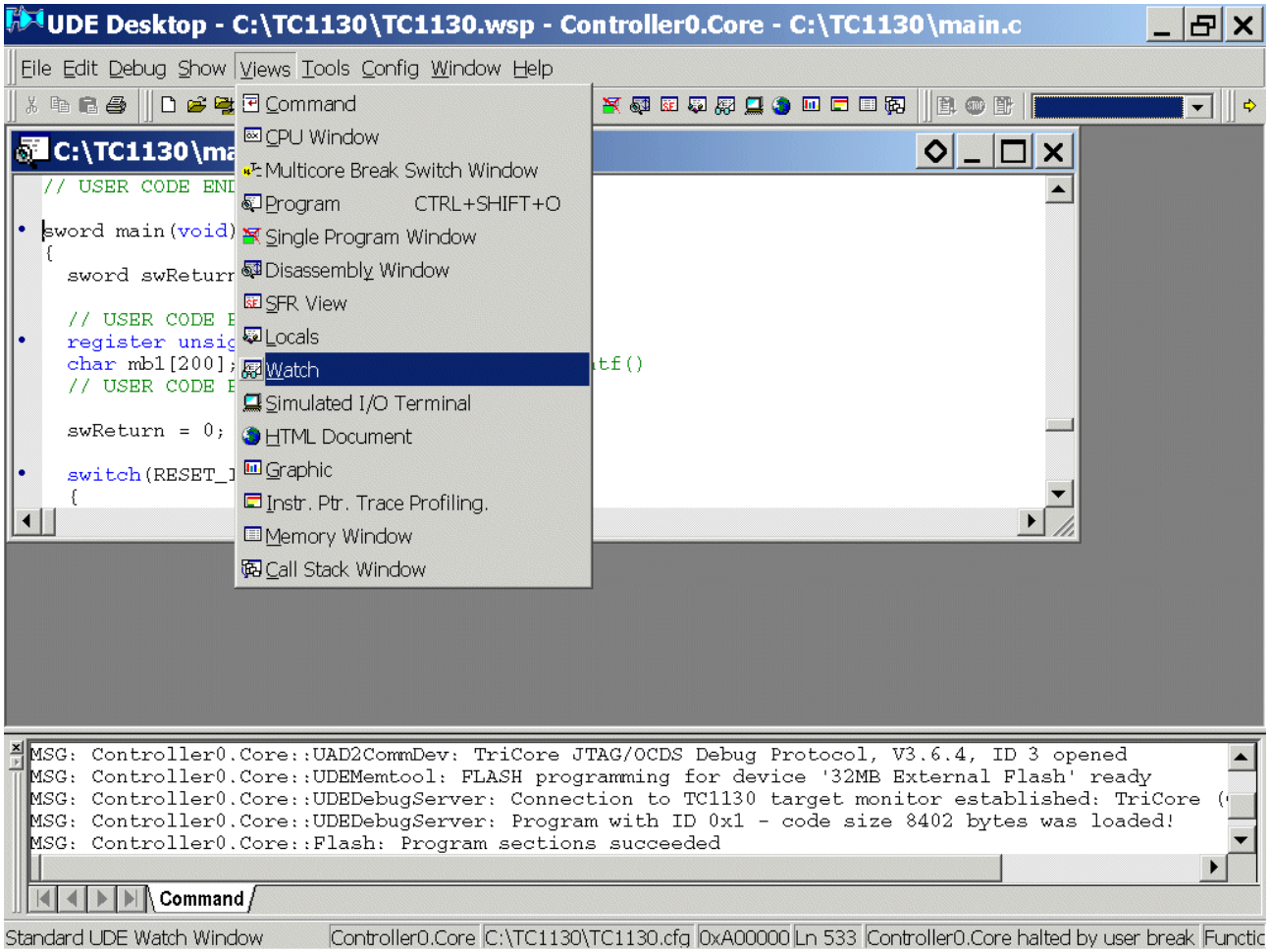
Load

Click Program All

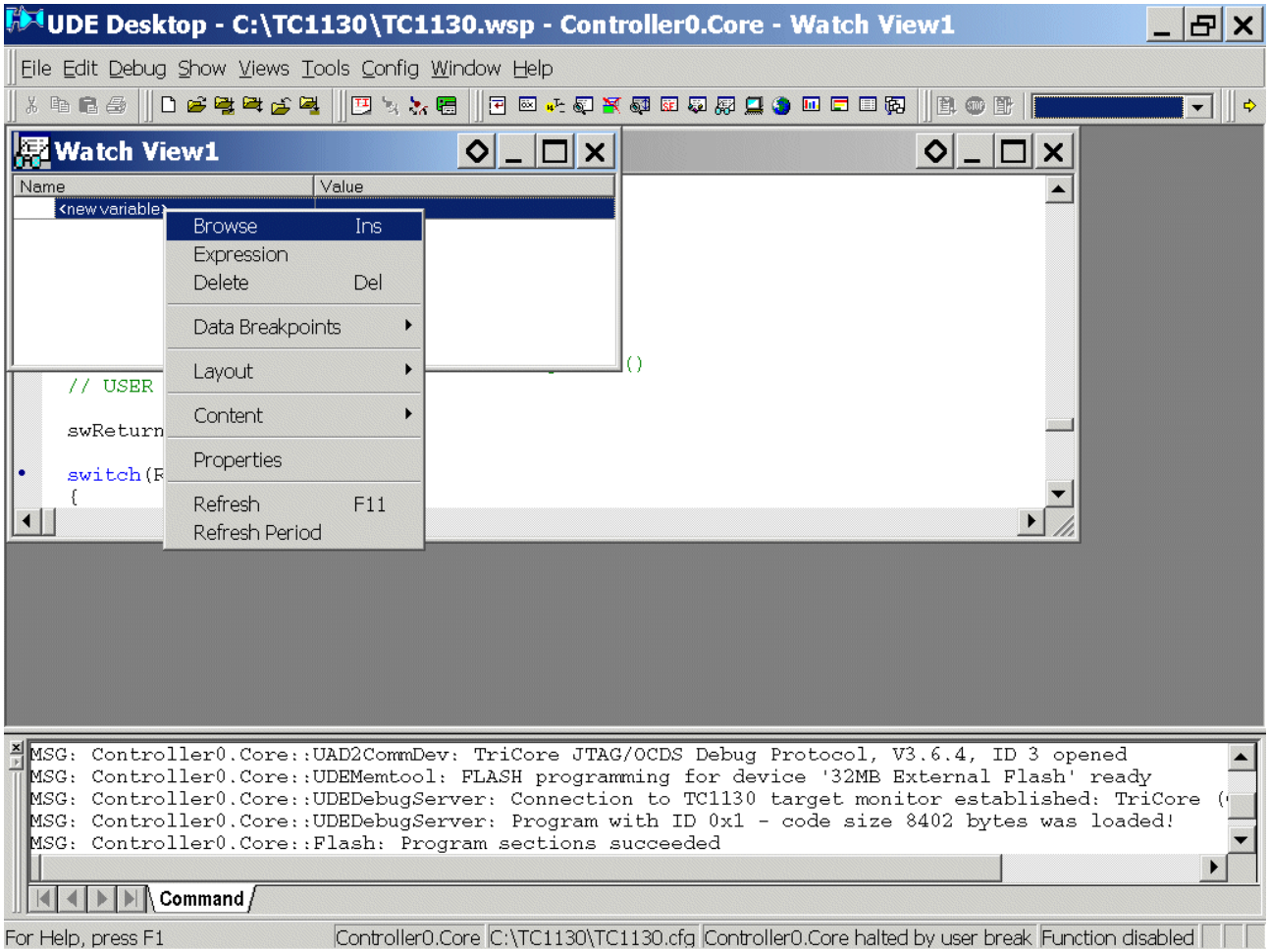
Exit
Exit


UDE Desktop - C:\TC1130\TC1130.wsp - Controller0.Core - C:\TC1130\main.c
File Edit Debug Show Views Tools Config Window Help
C:\TC1130\main.c
// USER CODE END
sword main(void)
{
sword swReturn;
// USER CODE BEGIN (Main,2)
register unsigned int index=0;
char mbl[200]; // message buffer1 for sprintf()
// USER CODE END
swReturn = 0;
switch(RESET_INDICATOR)
{
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '32MB External Flash' ready
MSG: Controller0.Core::UDEDebugServer: Connection to TC1130 target monitor established: TriCore
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 8402 bytes was loaded!
MSG: Controller0.Core::Flash: Program sections succeeded
Command
Controller0.Core
C:\TC1130\TC1130.cfg
0xA00000
Ln 533
Controller0.Core halted by user break

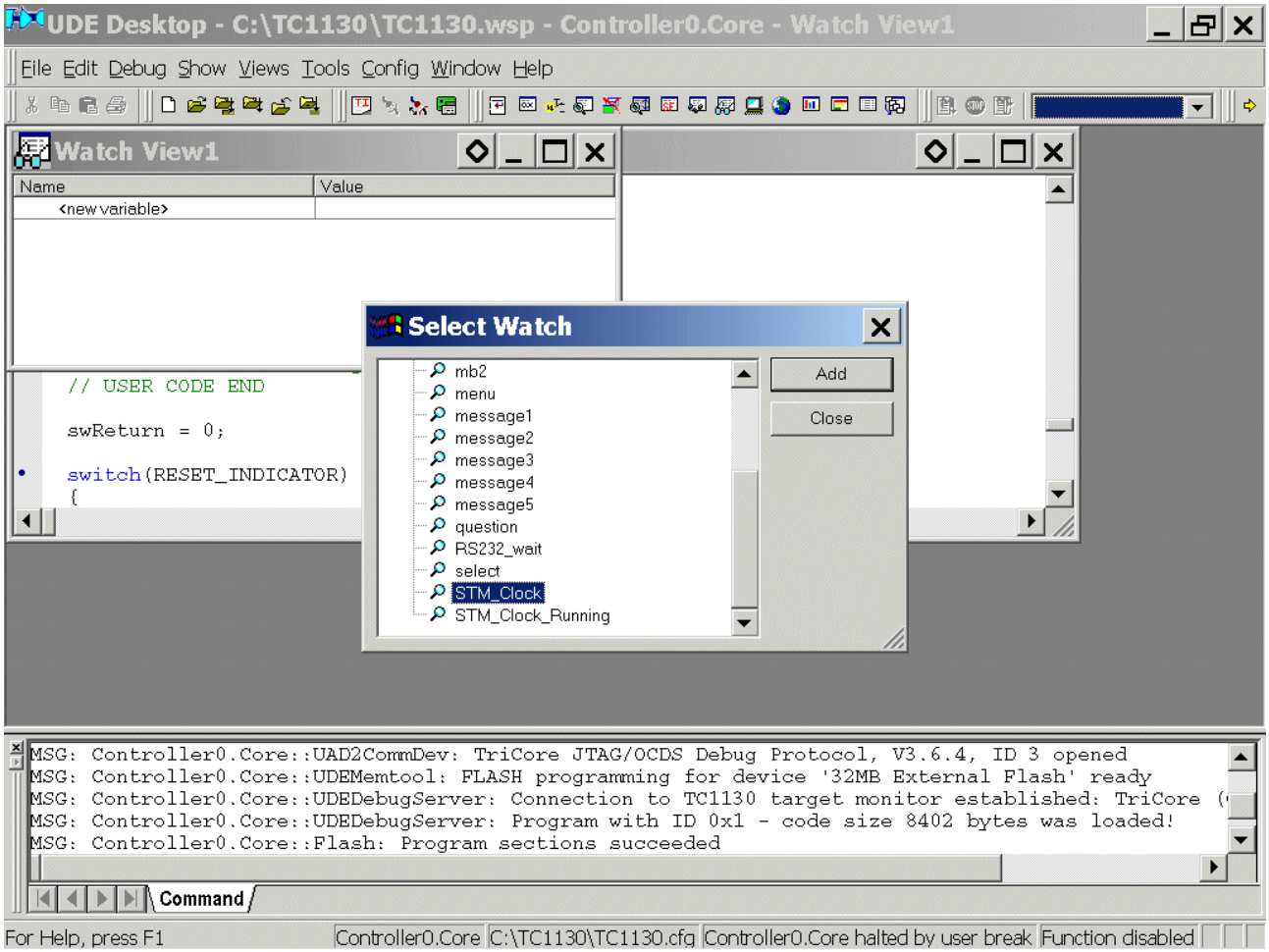
Views – Watch


UDE Desktop - C:\TC1130\TC1130.wsp - Controller0.Core - C:\TC1130\main.c
File Edit Debug Show Views Tools Config Window Help
Command
CPU Window
Multicore Break Switch Window
Program CTRL+SHIFT+O
Single Program Window
Disassembly Window
SFR View
Locals
Watch
Simulated I/O Terminal
HTML Document
Graphic
Instr. Ptr. Trace Profiling.
Memory Window
Call Stack Window
MSG: Controller0.Core::UAD2CommDev: TriCore JTAG/OCDS Debug Protocol, V3.6.4, ID 3 opened
MSG: Controller0.Core::UDEMemtool: FLASH programming for device '32MB External Flash' ready
MSG: Controller0.Core::UDEDebugServer: Connection to TC1130 target monitor established: TriCore
MSG: Controller0.Core::UDEDebugServer: Program with ID 0x1 - code size 8402 bytes was loaded!
MSG: Controller0.Core::Flash: Program sections succeeded
Command
Standard UDE Watch Window
Controller0.Core
C:\TC1130\TC1130.cfg
0xA00000
Ln 533
Controller0.Core halted by user break

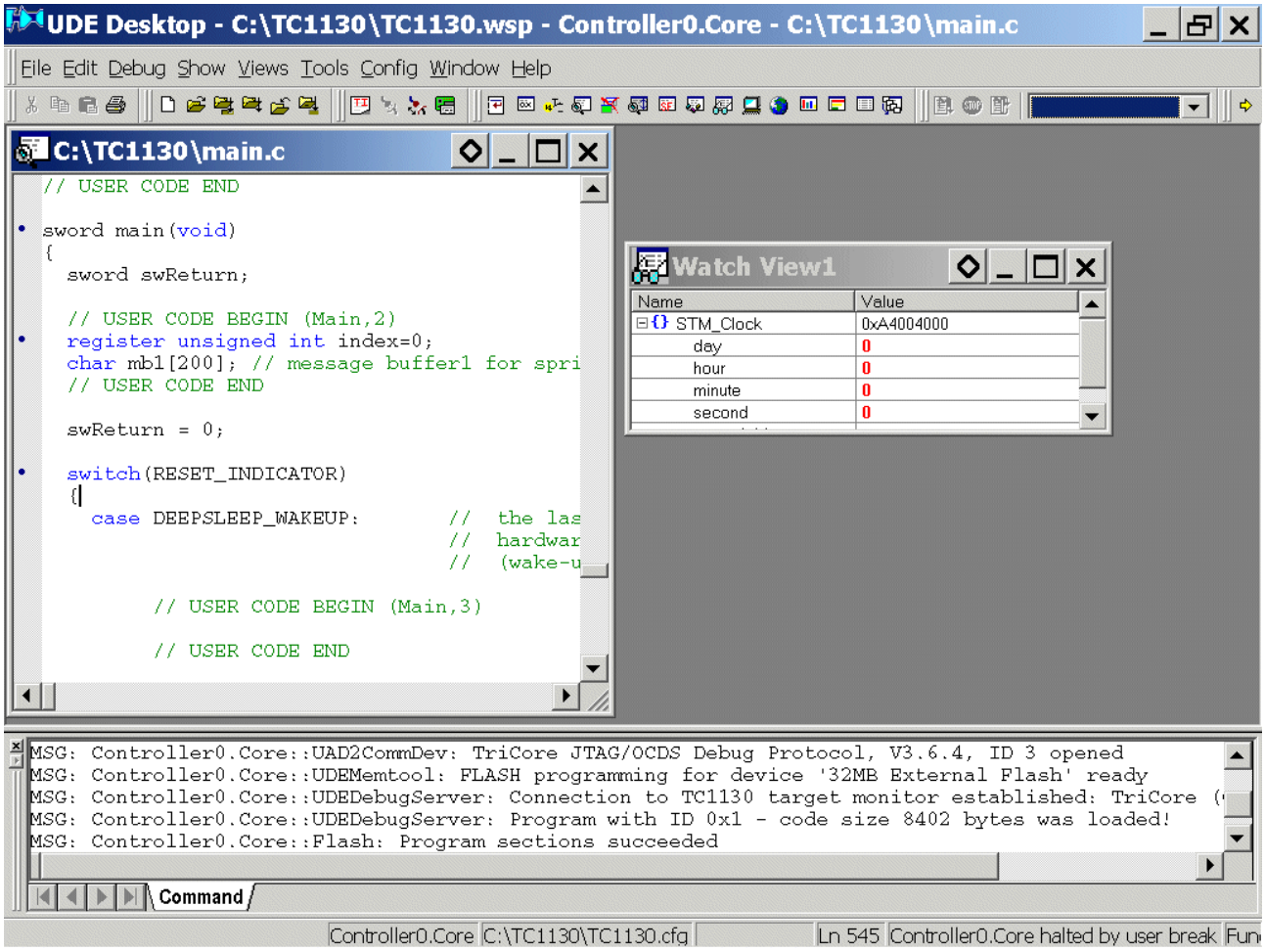
Right mouse button click at “new variable” select Browse



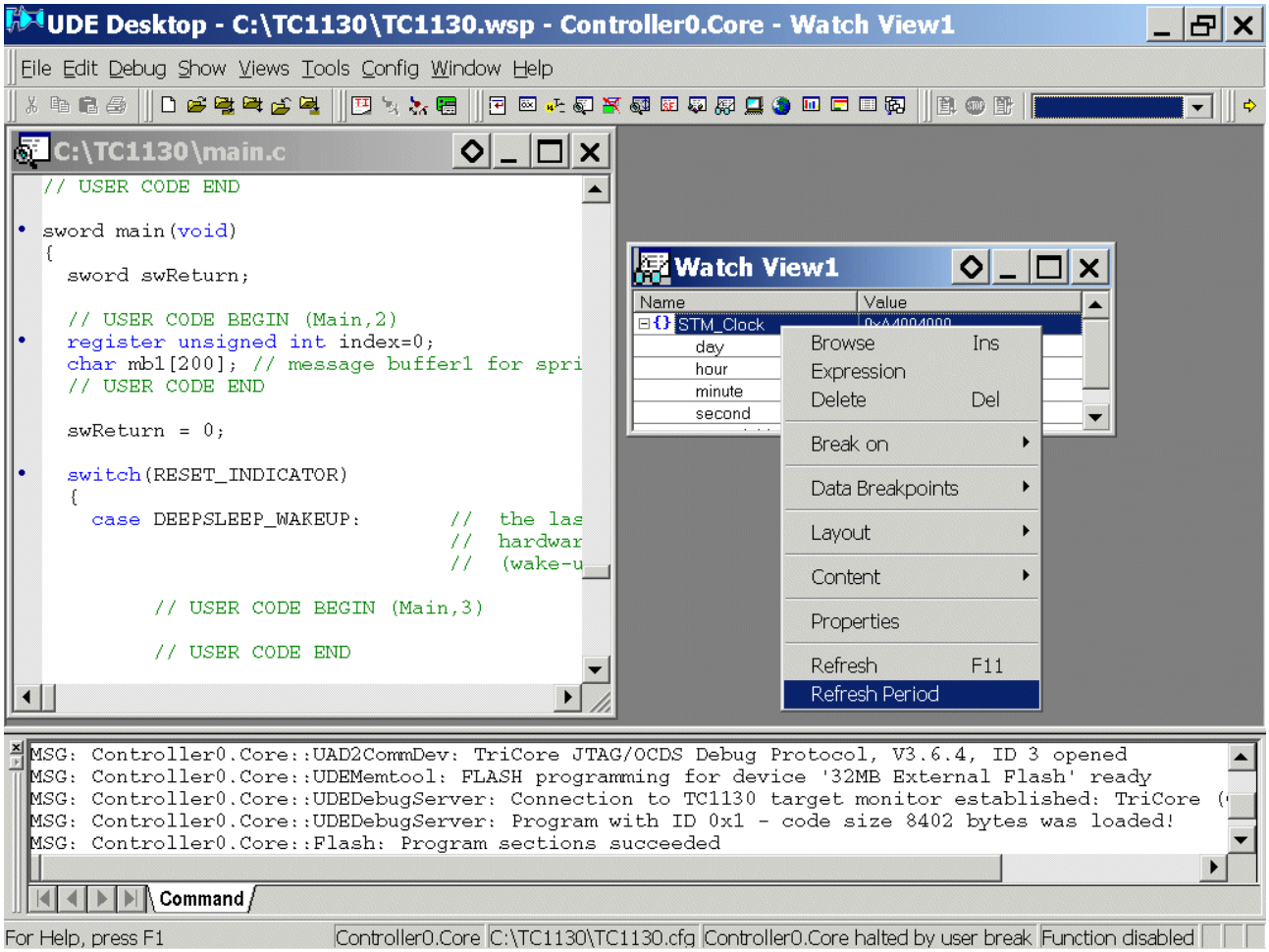
Select “STM_Clock”



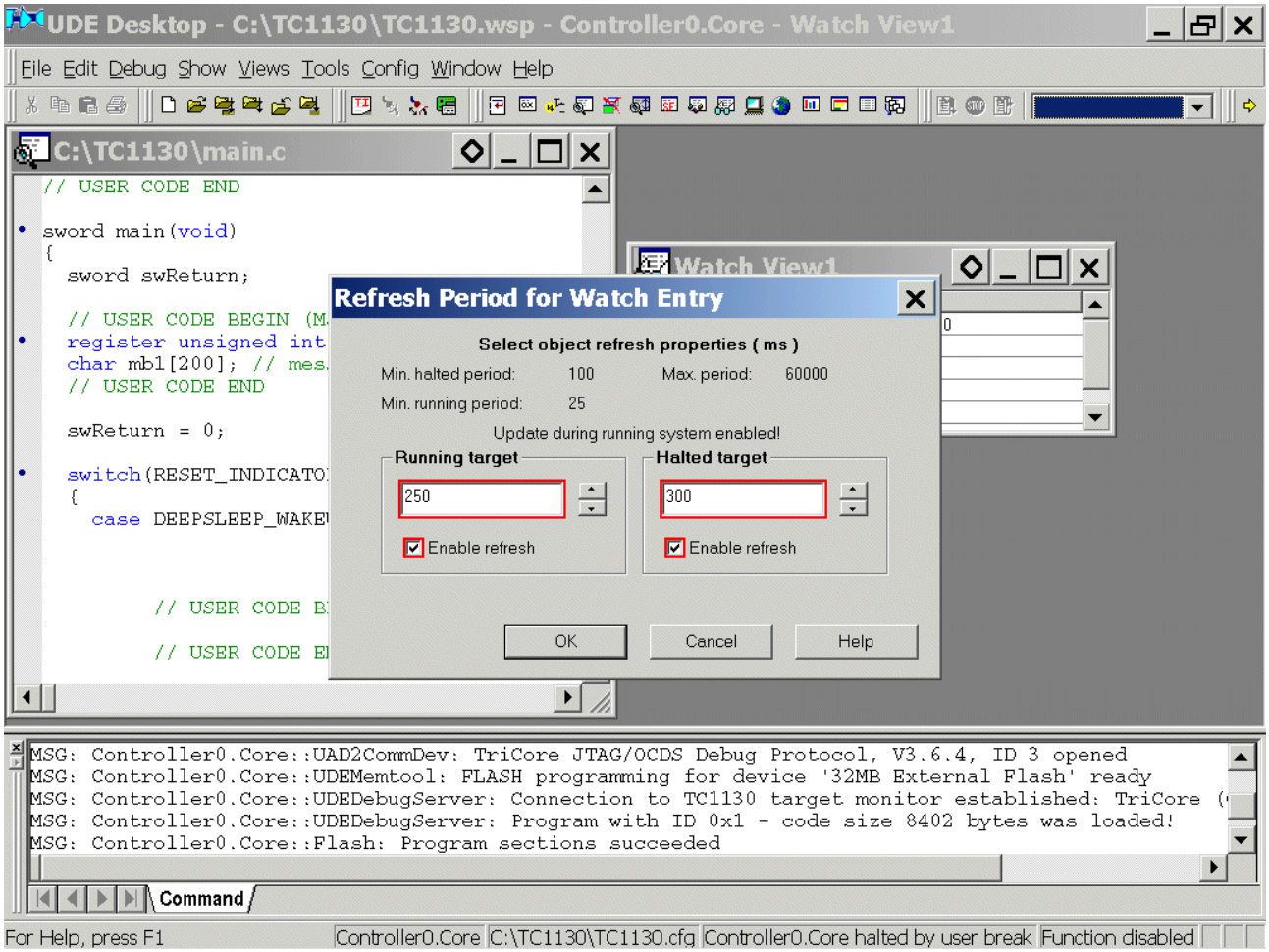
Click Add
Click Close



Right mouse button click at “STM_Clock” select Refresh Period

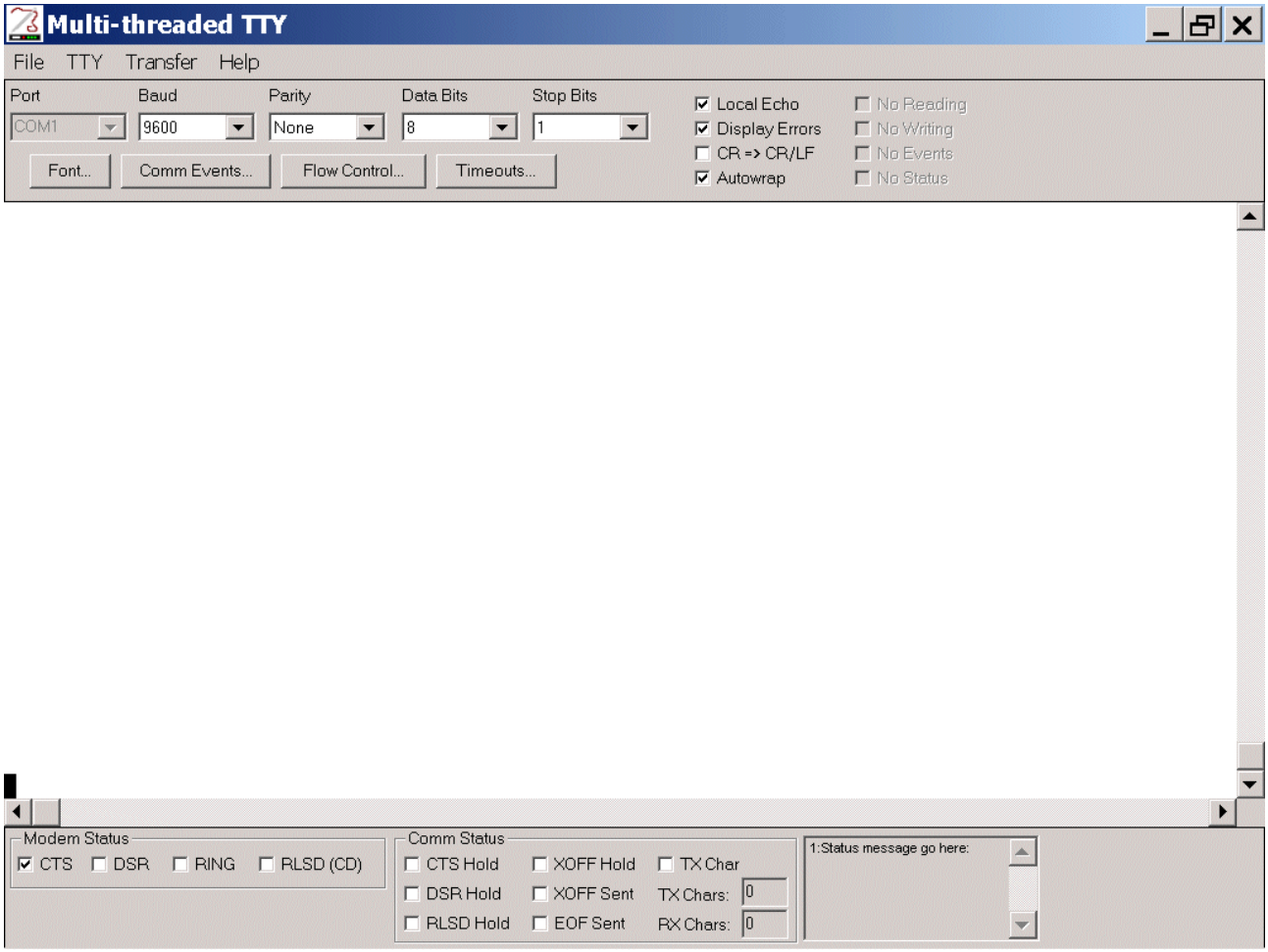


Running target: click ✓ Enable refresh
Running target: select 250
Halted target: click ✓ Enable refresh
Halted target: select 300

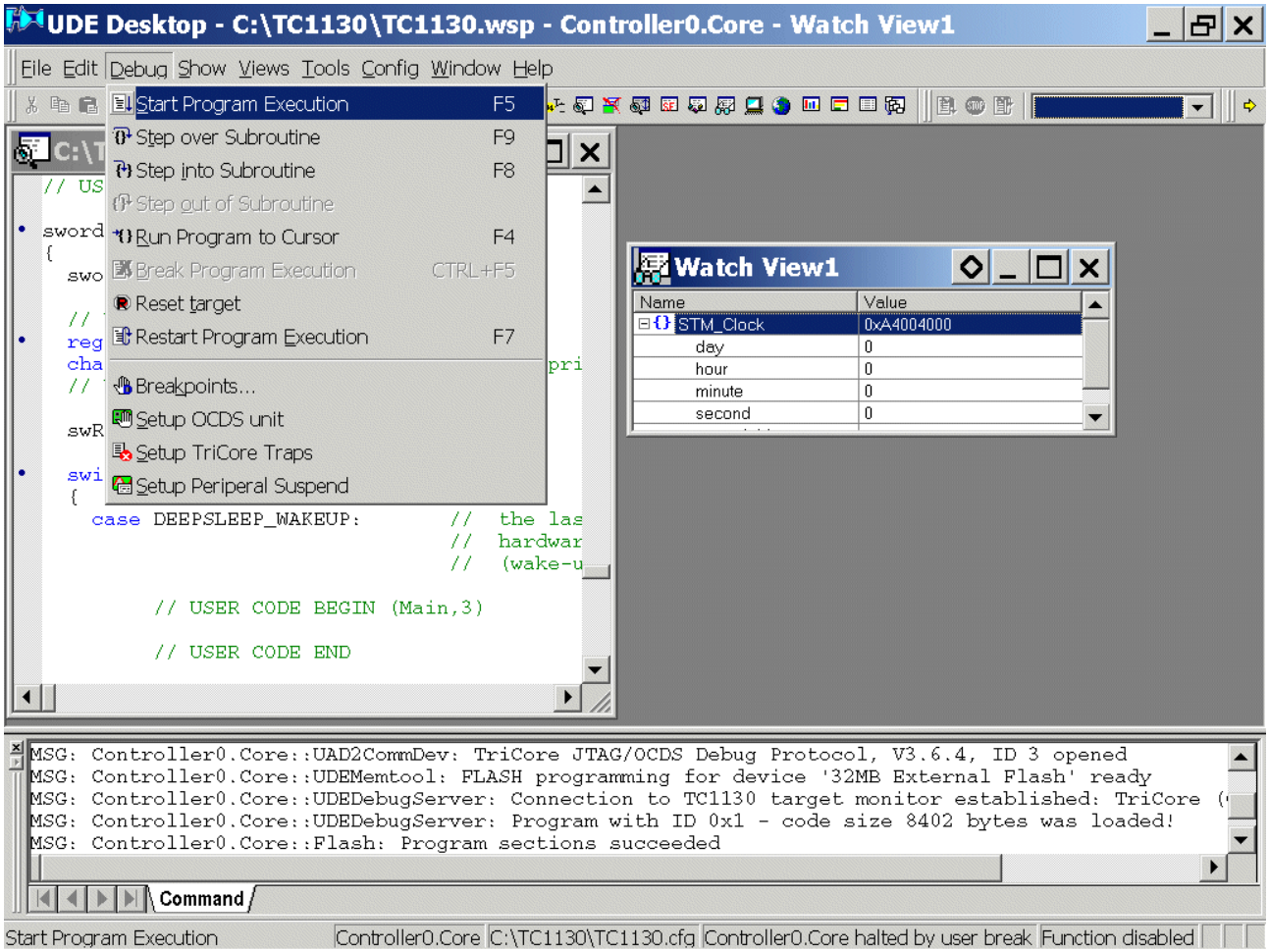


OK

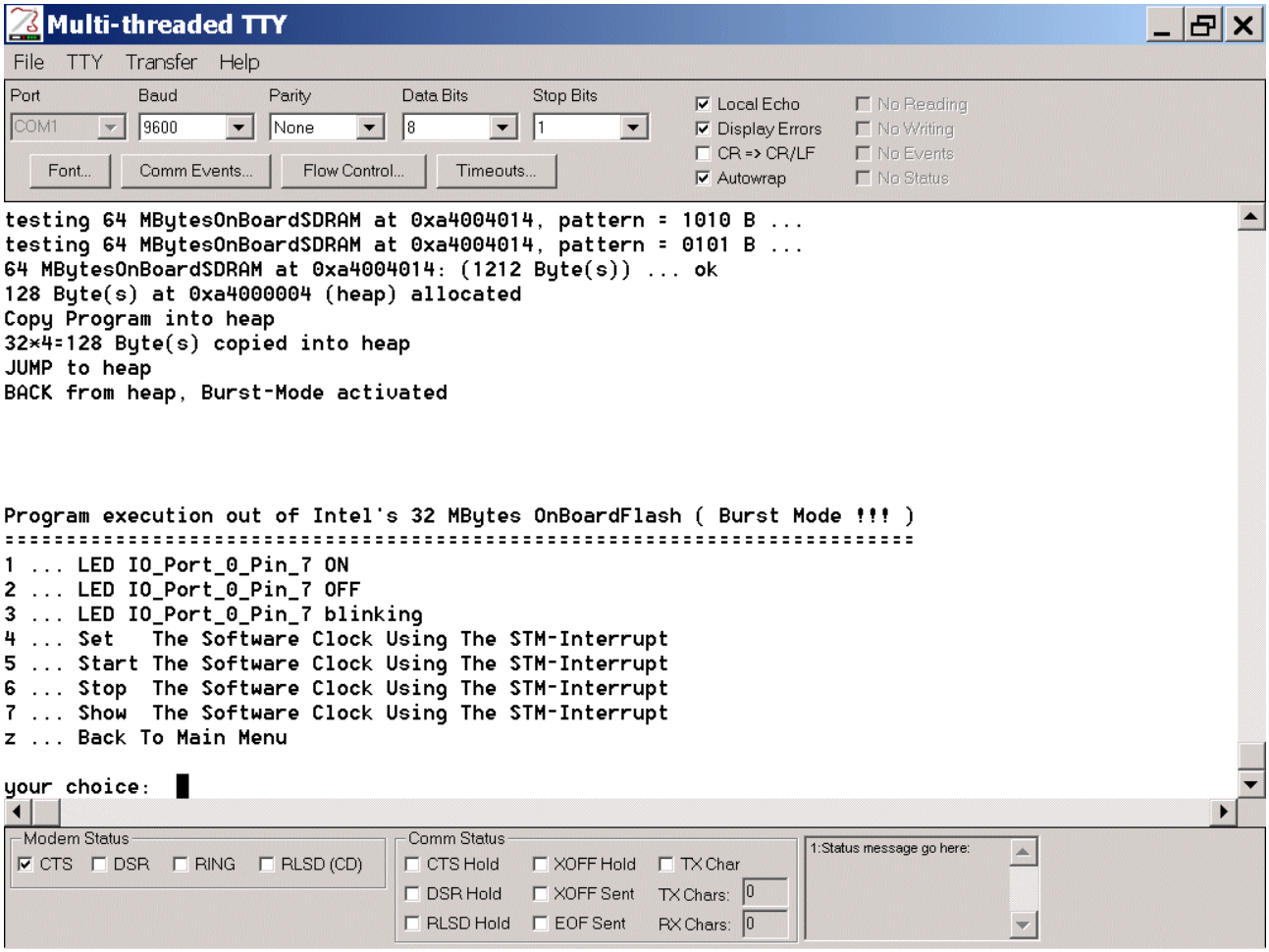
Execute any terminal-program
(9600 Baud, 8 bit Data, no Parity-Bit, 1 Stop-Bit, Xon/Xoff Protocol):



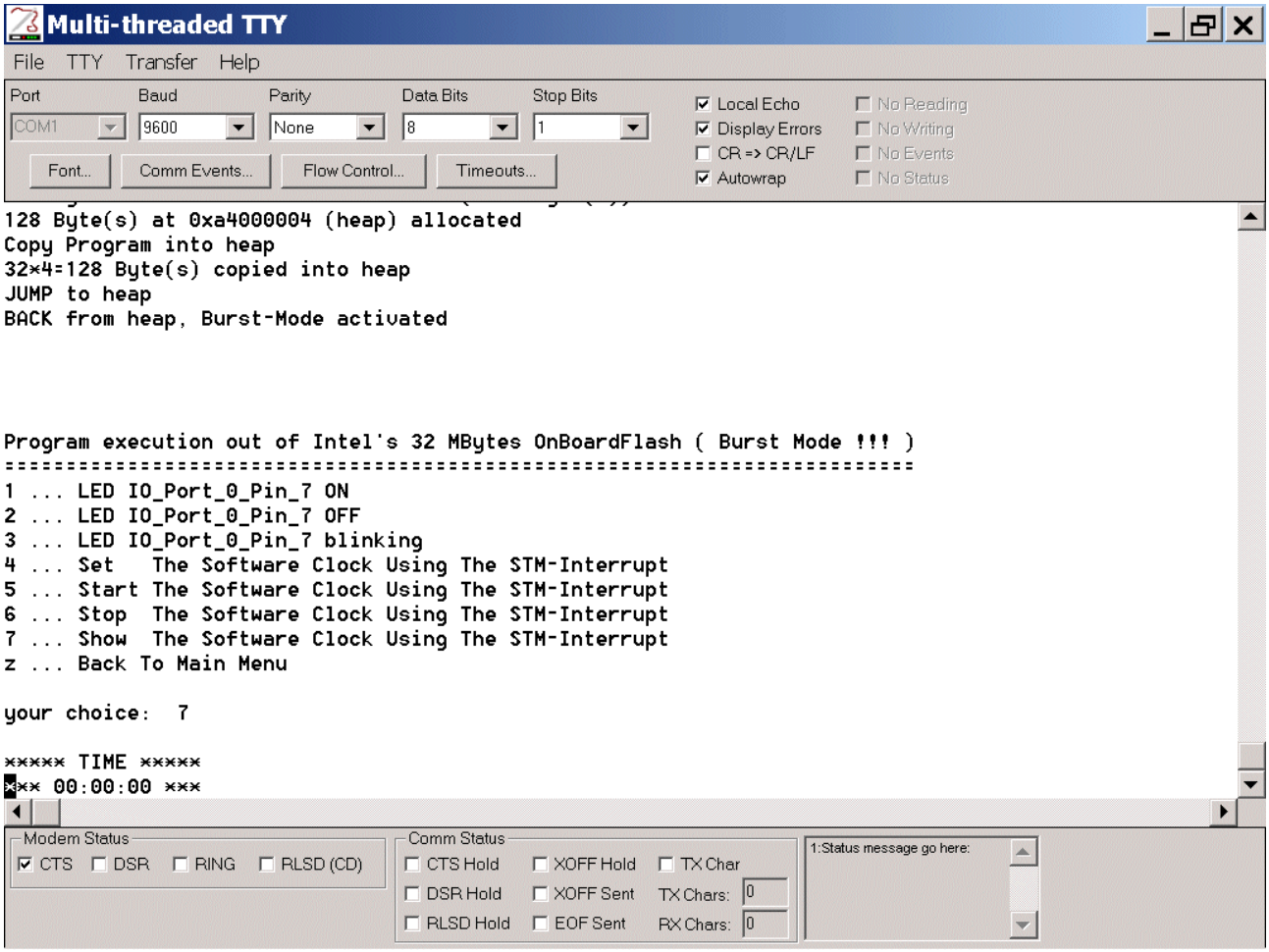
Debug – Start Program Execution



And see the result:

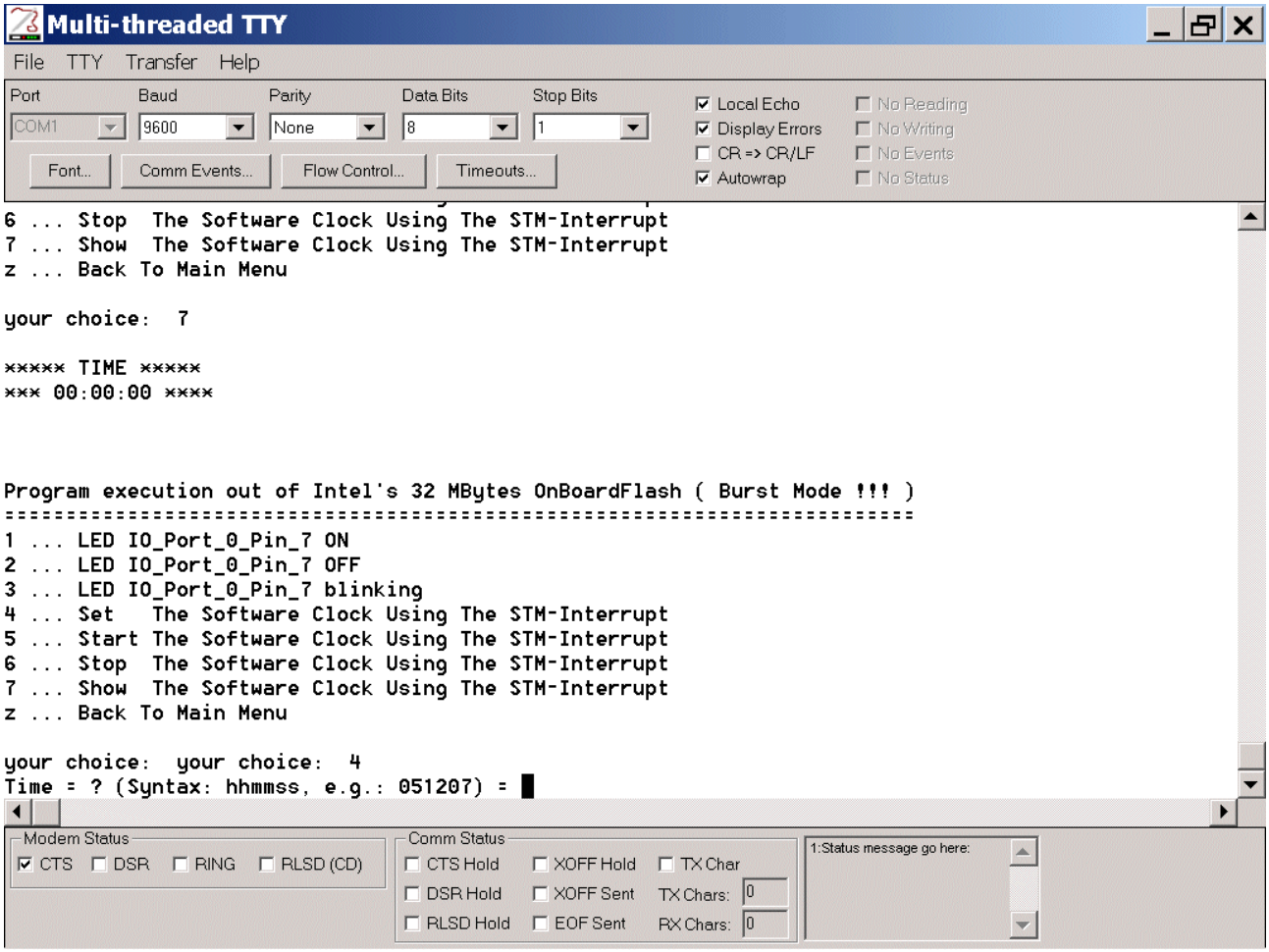


Insert 7

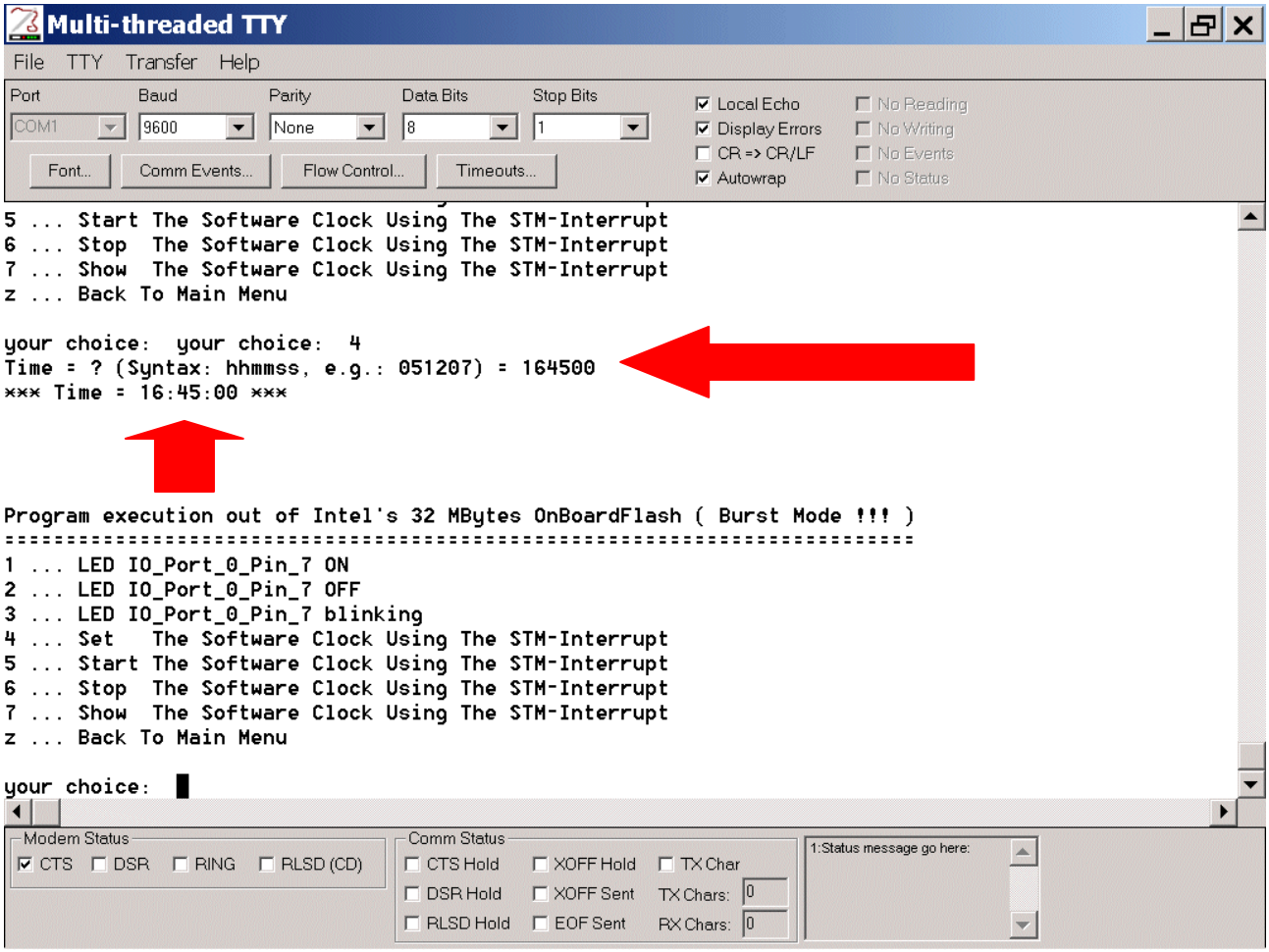


Insert z

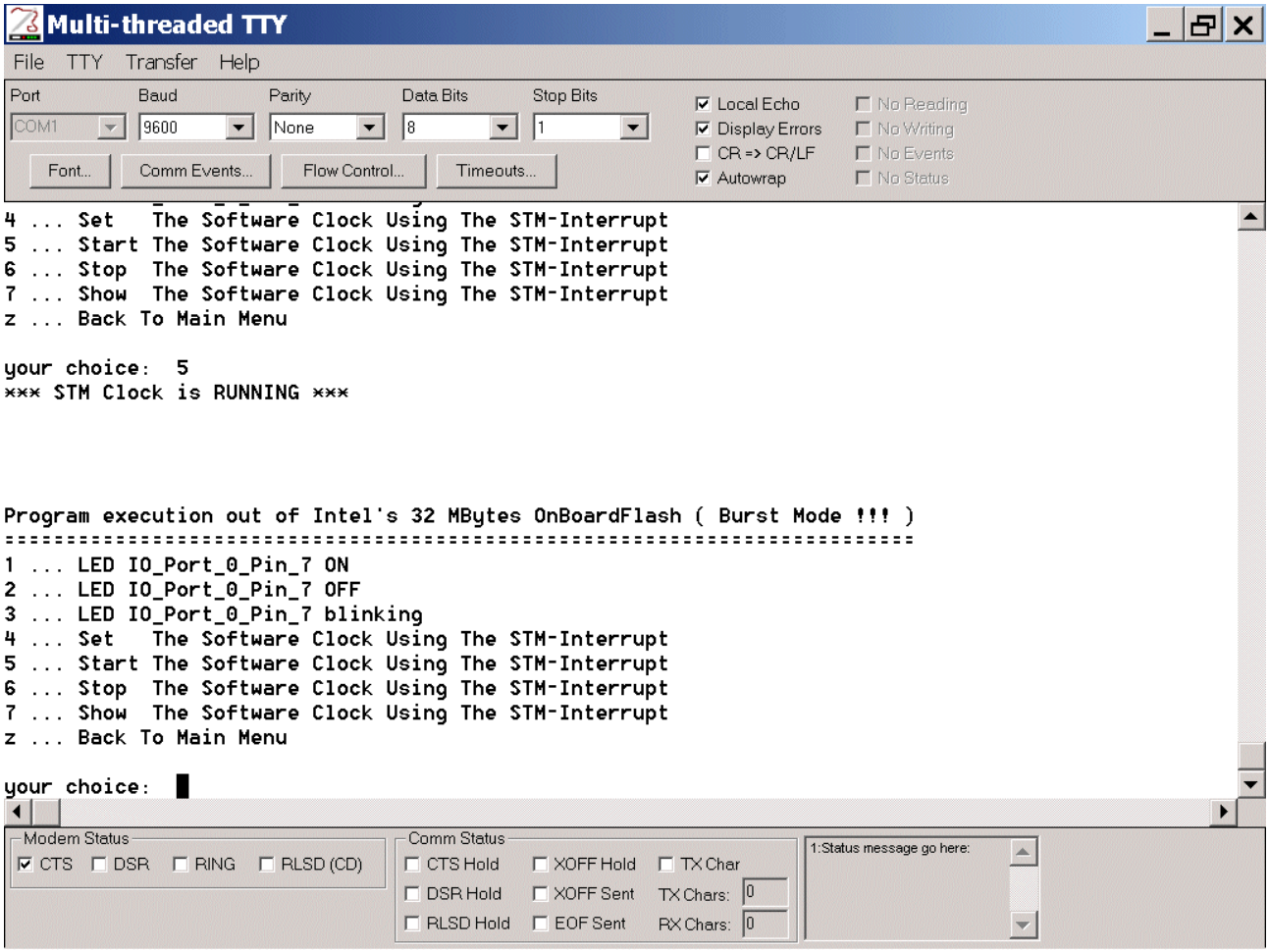
Insert 4


Multi-threaded TTY
File TTY Transfer Help
Port Baud Parity Data Bits Stop Bits
COM1 9600 None 8 1
Local Echo
Display Errors
CR => CR/LF
Autowrap
No Reading
No Writing
No Events
No Status
Font... Comm Events... Flow Control... Timeouts...
6 ... Stop The Software Clock Using The STM-Interrupt
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice: 7
***** TIME *****
*** 00:00:00 ****
Program execution out of Intel's 32 MBytes OnBoardFlash (Burst Mode !!!)
1 ... LED IO_Port_0_Pin_7 ON
2 ... LED IO_Port_0_Pin_7 OFF
3 ... LED IO_Port_0_Pin_7 blinking
4 ... Set The Software Clock Using The STM-Interrupt
5 ... Start The Software Clock Using The STM-Interrupt
6 ... Stop The Software Clock Using The STM-Interrupt
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice: your choice: 4
Time = ? (Syntax: hhmmss, e.g.: 051207) =
Modem Status
CTS DSR RING RLSD (CD)
Comm Status
CTS Hold XOFF Hold TX Char
DSR Hold XOFF Sent TX Chars: 0
RLSD Hold EOF Sent RX Chars: 0
1:Status message go here:

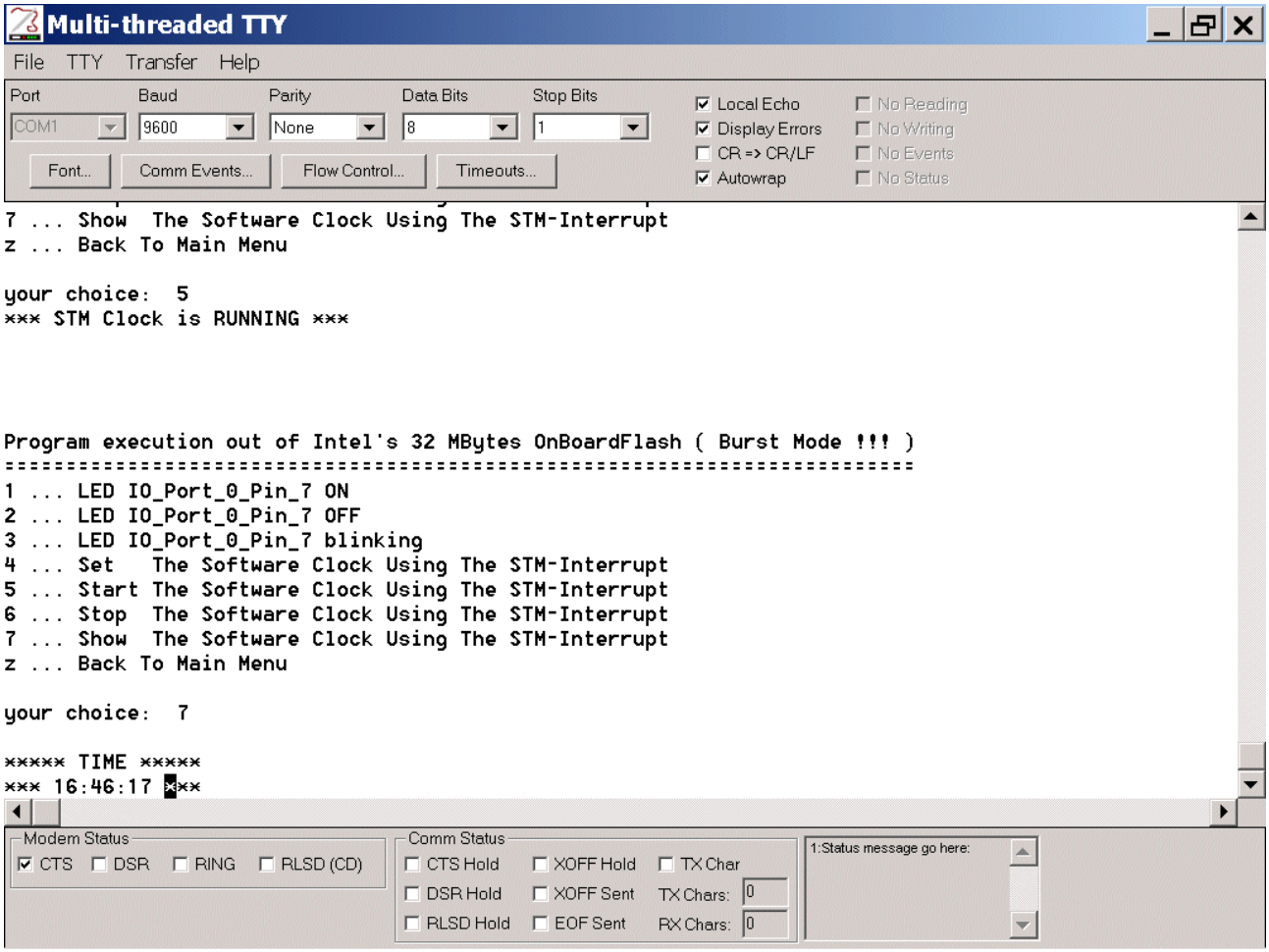
Insert 164500



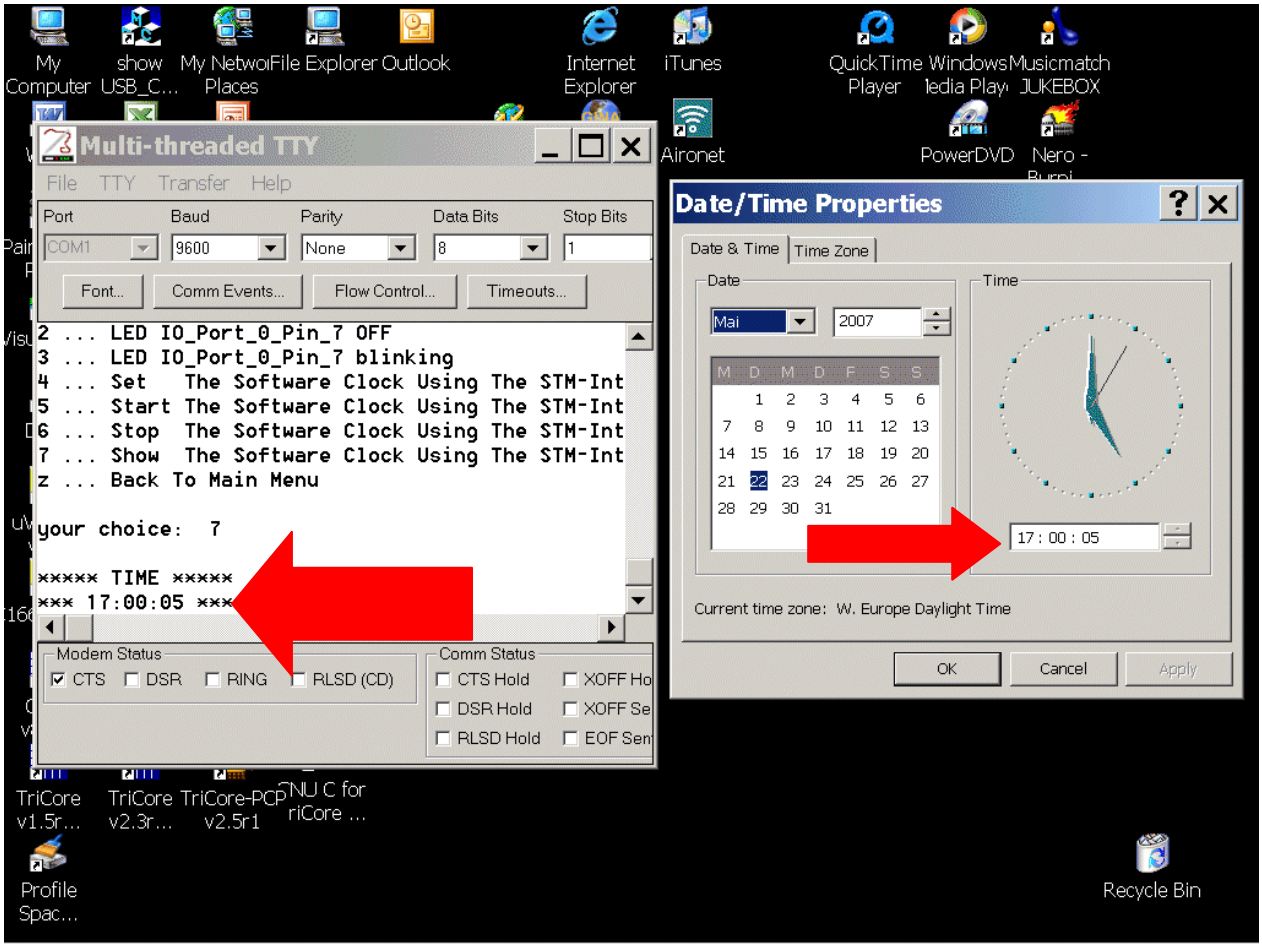
@ 16:45:00 insert 5



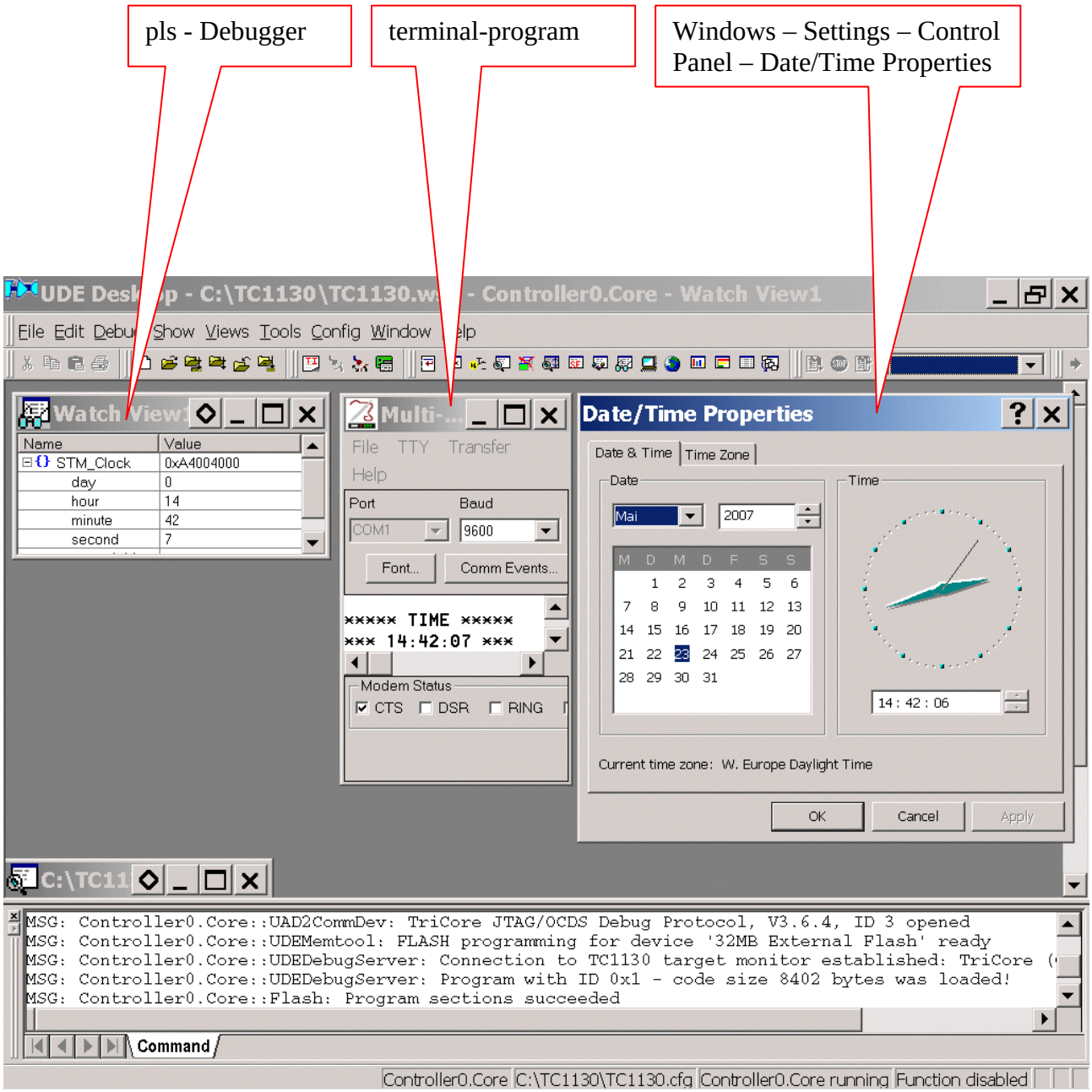
Insert 7


Multi-threaded TTY
File TTY Transfer Help
Port Baud Parity Data Bits Stop Bits
COM1 9600 None 8 1
Local Echo
Display Errors
CR => CR/LF
Autowrap
No Reading
No Writing
No Events
No Status
Font... Comm Events... Flow Control... Timeouts...
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice: 5
*** STM Clock is RUNNING ***
Program execution out of Intel's 32 MBytes OnBoardFlash (Burst Mode !!!)
1 ... LED IO_Port_0_Pin_7 ON
2 ... LED IO_Port_0_Pin_7 OFF
3 ... LED IO_Port_0_Pin_7 blinking
4 ... Set The Software Clock Using The STM-Interrupt
5 ... Start The Software Clock Using The STM-Interrupt
6 ... Stop The Software Clock Using The STM-Interrupt
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice: 7
***** TIME *****
*** 16:46:17 ***
Modem Status
CTS DSR RING RLSD (CD)
Comm Status
CTS Hold XOFF Hold TX Char
DSR Hold XOFF Sent TX Chars: 0
RLSD Hold EOF Sent RX Chars: 0
1:Status message go here:

Compare with your “computer time”:



After 1 day (Compare with your “computer time” and with your debugger):

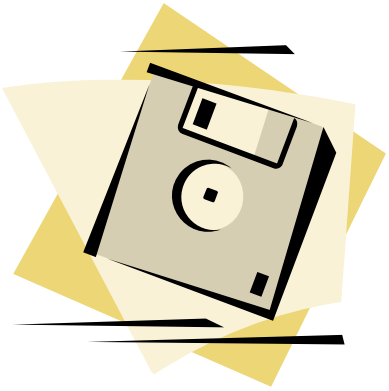


Close the pls-Debugger:

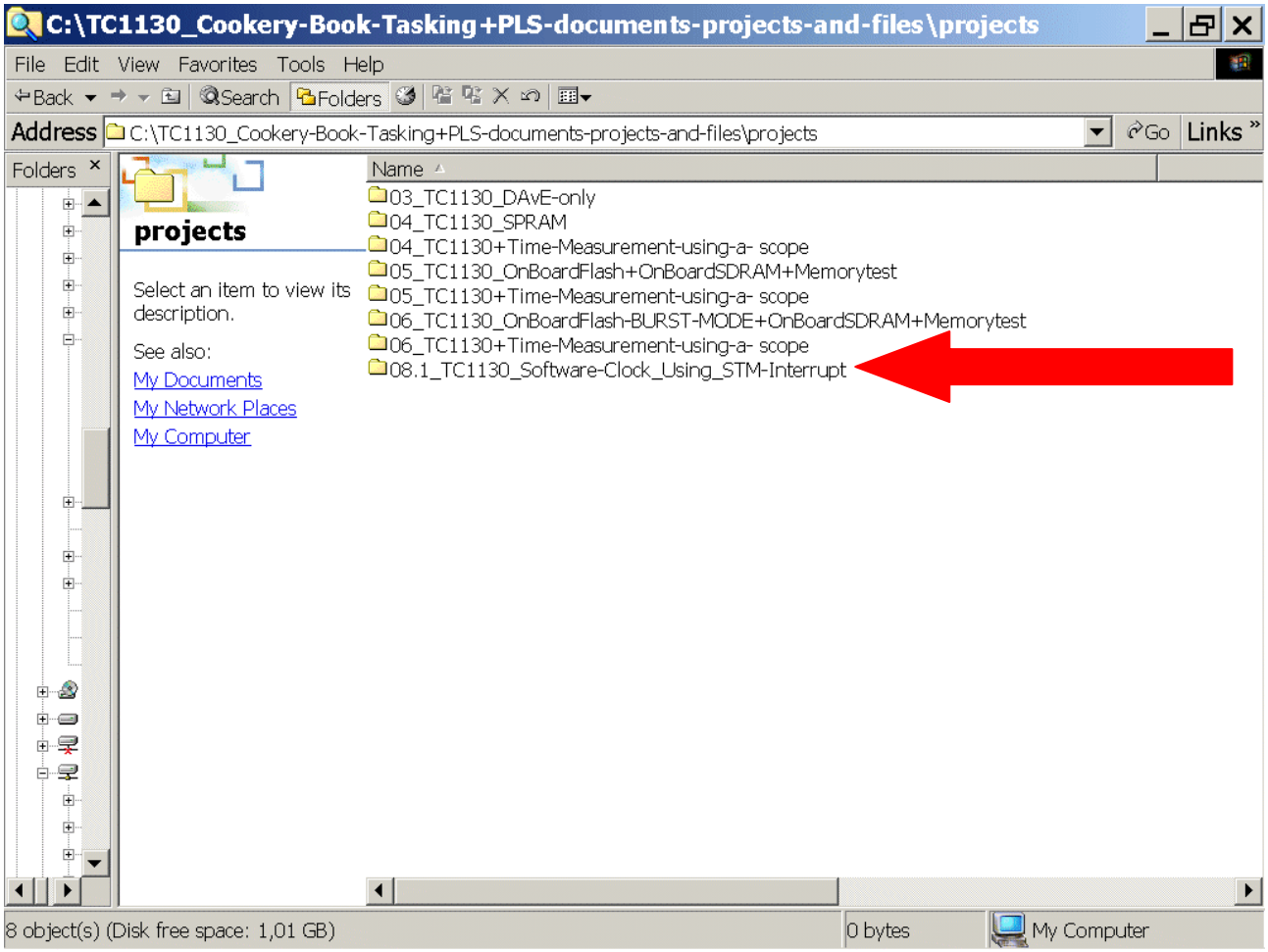
File – Close Workspace

Yes

File – Exit



We recommend now to copy and store your project-directory “C:\TC1130” to “08.1_TC1130_Software-Clock_Using_STM-Interrupt”:



8.2.) Time-Lag-Measurement Using the STM

Note:
In this chapter we are going to use the STM for Time-Lag-Measurement (“long-time” and “short time” lag-measurement).

8.2.1.) Time-Lag-Measurement Using the STM (example: “long-time-lag-measurement”)

Start Tasking EDE and open the project:

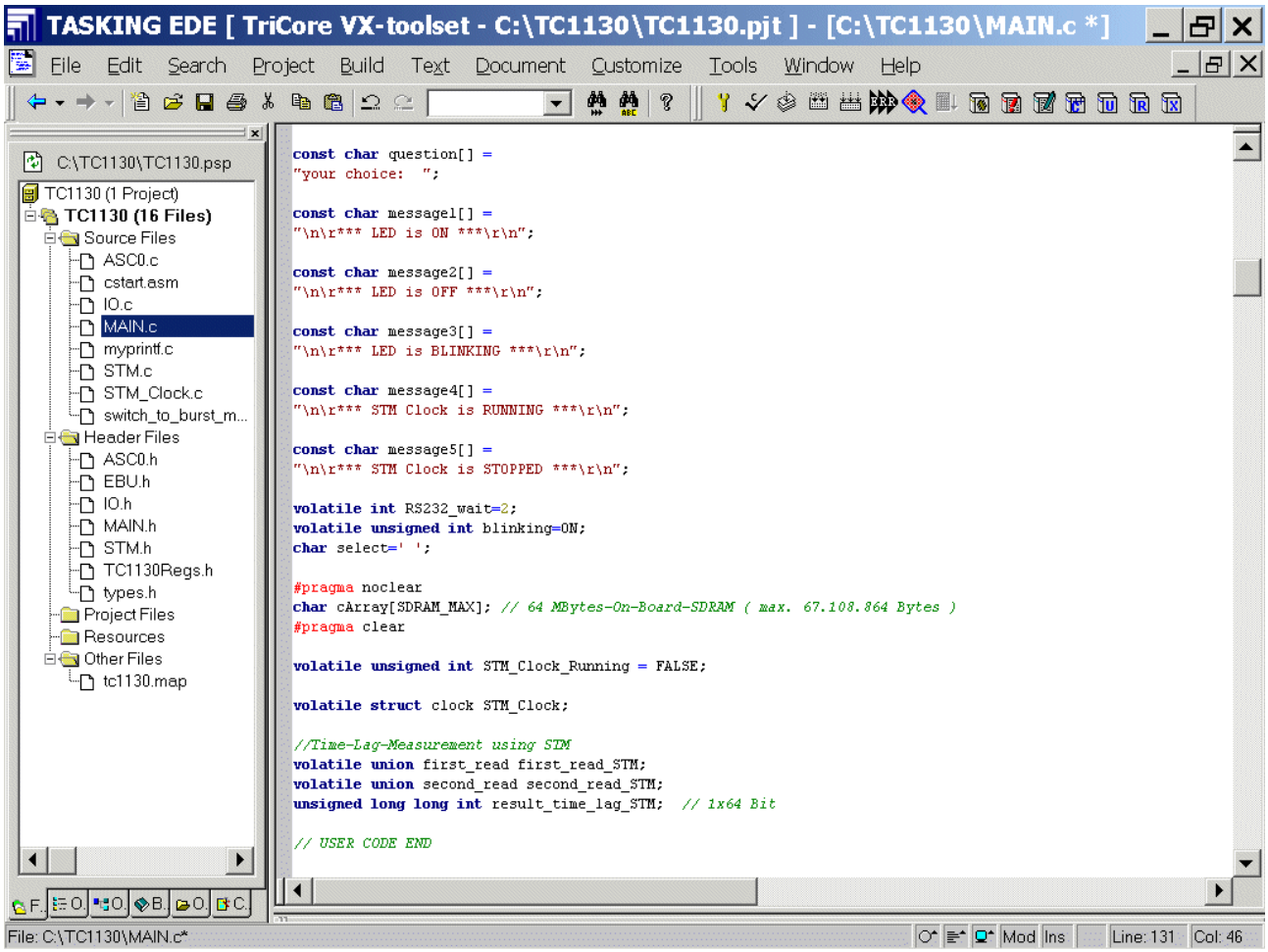
File – Open Project Space

Look in: select C:\TC1130
File name: select TC1130.psp

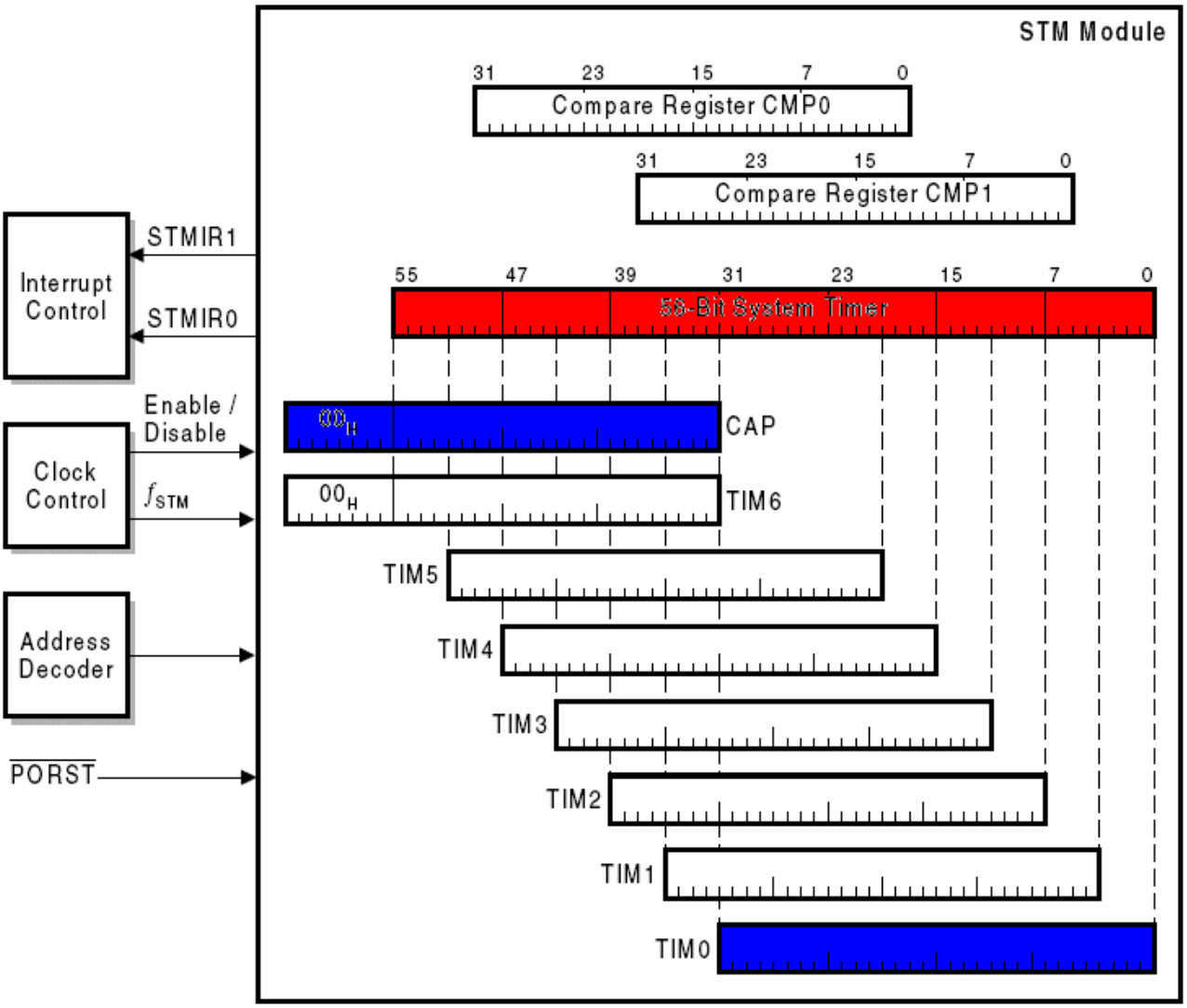
Open

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

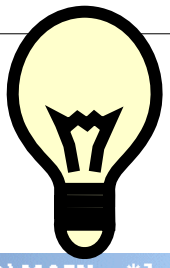
```
//Time-Lag-Measurement using STM
volatile union first_read first_read_STM;
volatile union second_read second_read_STM;
unsigned long long int result_time_lag_STM;  // 1x64 Bit
```



Additional Information: System Timer (STM):

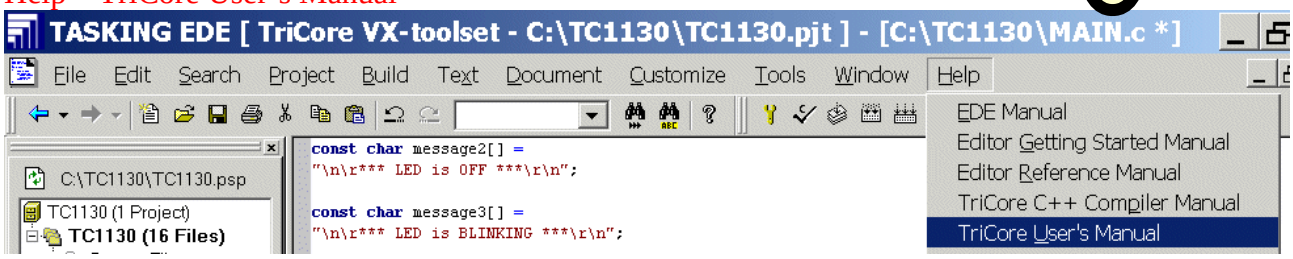


Note:
Because the STM is 56 bits wide, it is not possible to read its entire contents with one instruction. It must be read with two load instructions. Since the timer would continue to count between the two load operations, there is a chance that the two values read may not be consistent (due to possible overflow from the low part of the timer to the high part between the two read operations). To enable synchronous and consistent reading of the STM contents, a capture register (CAP), is implemented. It latches the contents of the high part of the STM each time one of the registers TIM0 to TIM5 is read. Thus, it holds the upper value of the timer at exactly the same time when the lower part is read. The second read operation then reads the contents of the CAP for the complete timer value.



Additional Information: fundamental data types:

Help – TriCore User's Manual

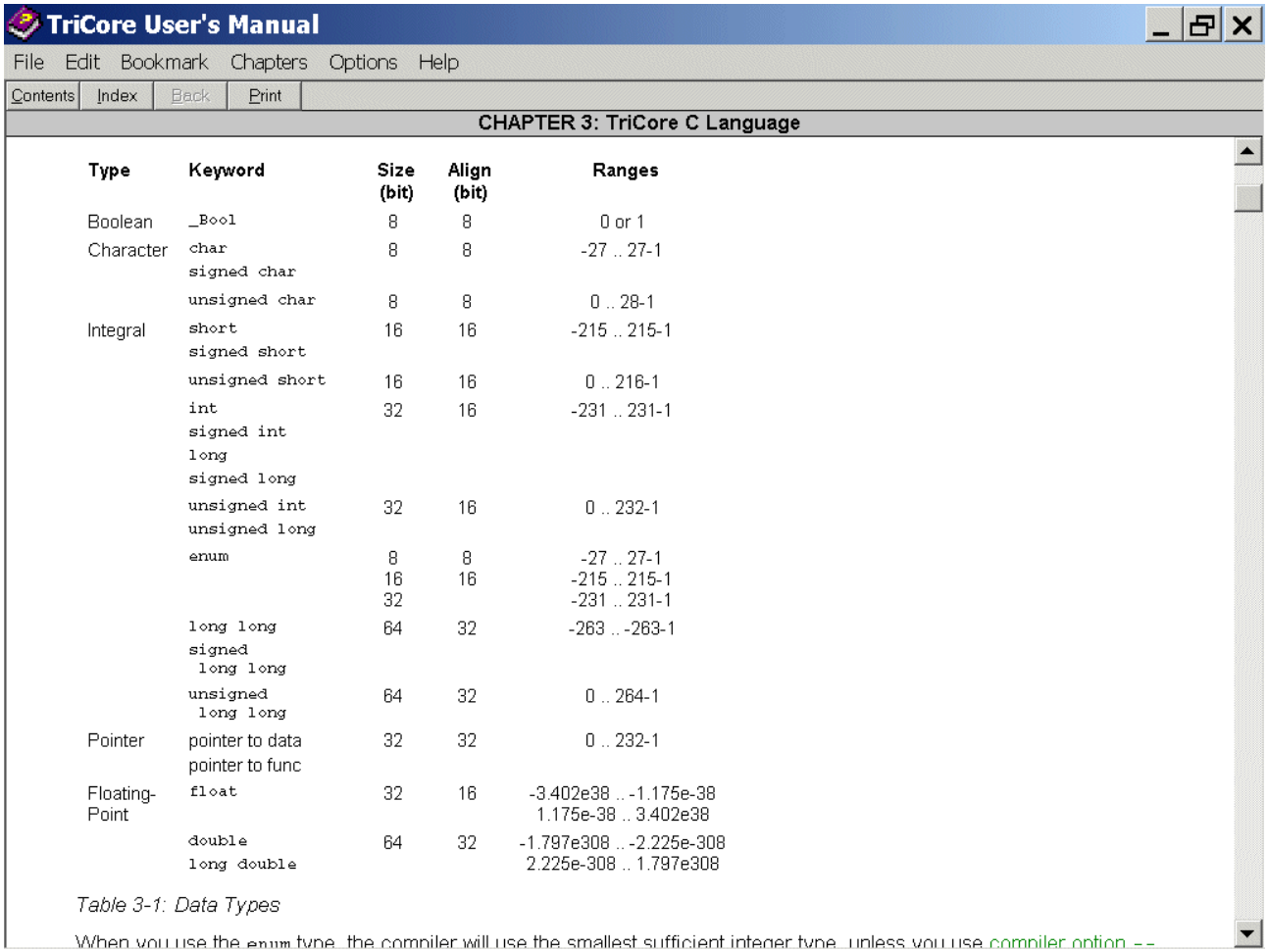


Note:

The TriCore architecture defines the following fundamental data types:

- An 8-bit byte
- A 16-bit short
- A 32-bit word
- A 64-bit double word

The next picture shows the mapping between these fundamental data types and the C language data type:



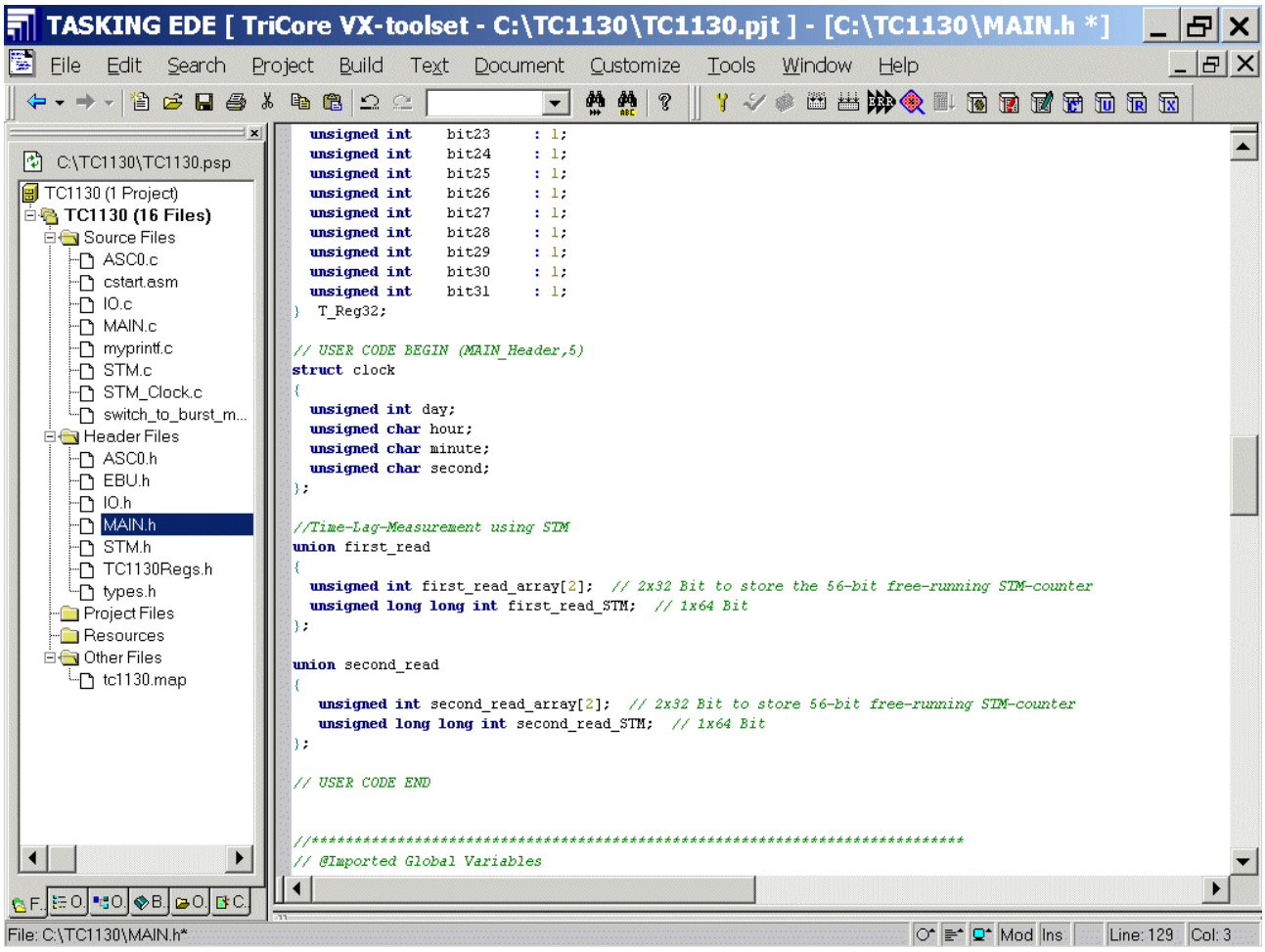
Type	Keyword	Size (bit)	Align (bit)	Ranges
Boolean	_Bool	8	8	0 or 1
Character	char signed char	8	8	-27 .. 27-1
	unsigned char	8	8	0 .. 28-1
Integral	short signed short	16	16	-215 .. 215-1
	unsigned short	16	16	0 .. 216-1
	int signed int long signed long	32	16	-231 .. 231-1
	unsigned int unsigned long	32	16	0 .. 232-1
	enum	8 16 32	8 16	-27 .. 27-1 -215 .. 215-1 -231 .. 231-1
	long long signed long long	64	32	-263 .. -263-1
	unsigned long long	64	32	0 .. 264-1
Pointer	pointer to data pointer to func	32	32	0 .. 232-1
Floating-Point	float	32	16	-3.402e38 .. -1.175e-38 1.175e-38 .. 3.402e38
	double long double	64	32	-1.797e308 .. -2.225e-308 2.225e-308 .. 1.797e308

Table 3-1: Data Types

When you use the enum type, the compiler will use the smallest sufficient integer type, unless you use compiler option --

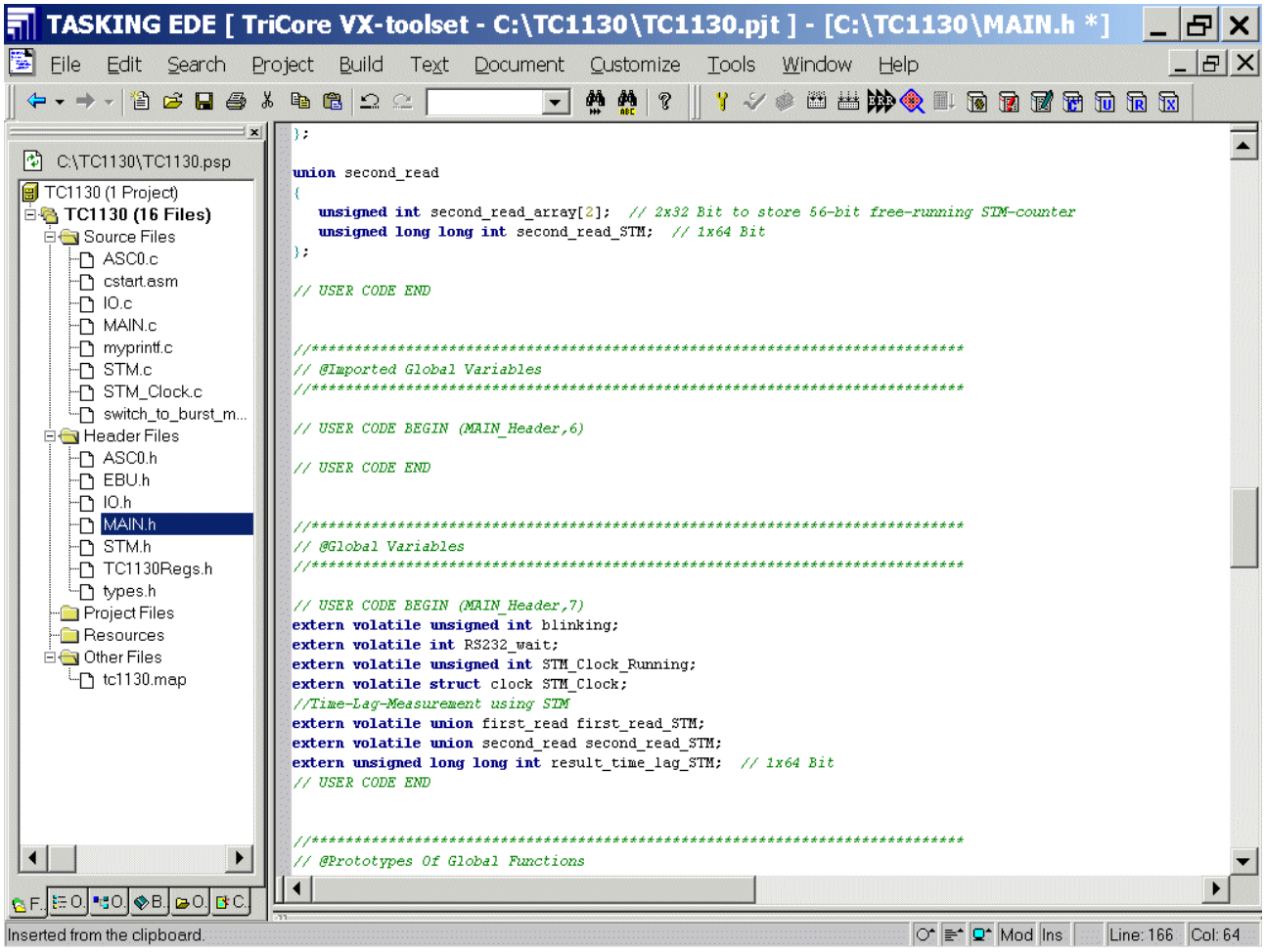
Double click: Main.h and insert Typedefs:

```
//Time-Lag-Measurement using STM
union first_read
{
 unsigned int first_read_array[2];  // 2x32 Bit to store the 56-bit free-running STM-counter
 unsigned long long int first_read_STM;  // 1x64 Bit
};

union second_read
{
  unsigned int second_read_array[2];  // 2x32 Bit to store 56-bit free-running STM-counter
  unsigned long long int second_read_STM;  // 1x64 Bit
};
```



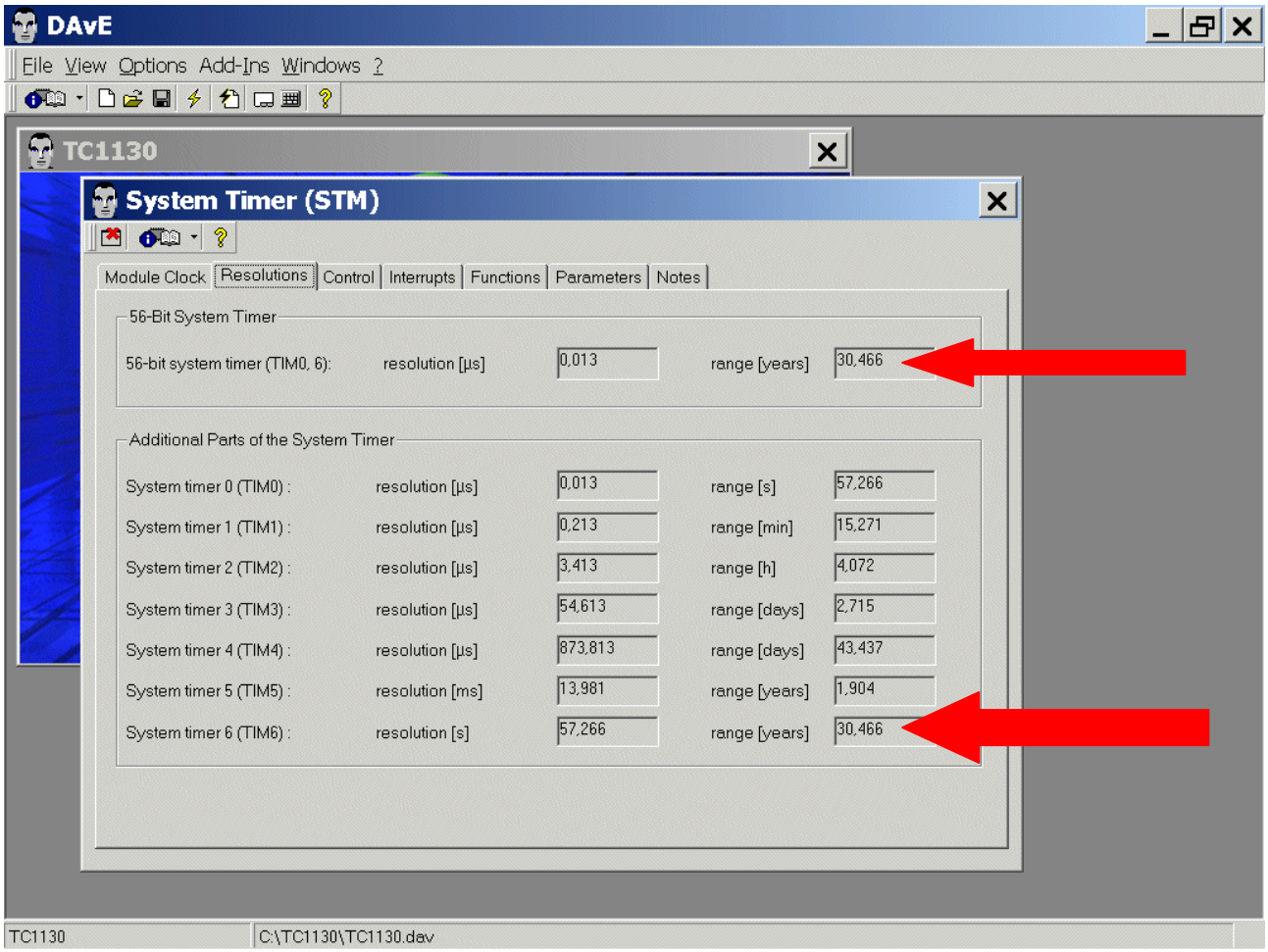
Double click: Main.h and insert Prototypes of Global Variables (Extern Declaration):

```
//Time-Lag-Measurement using STM
extern volatile union first_read first_read_STM;
extern volatile union second_read second_read_STM;
extern unsigned long long int result_time_lag_STM;  // 1x64 Bit
```

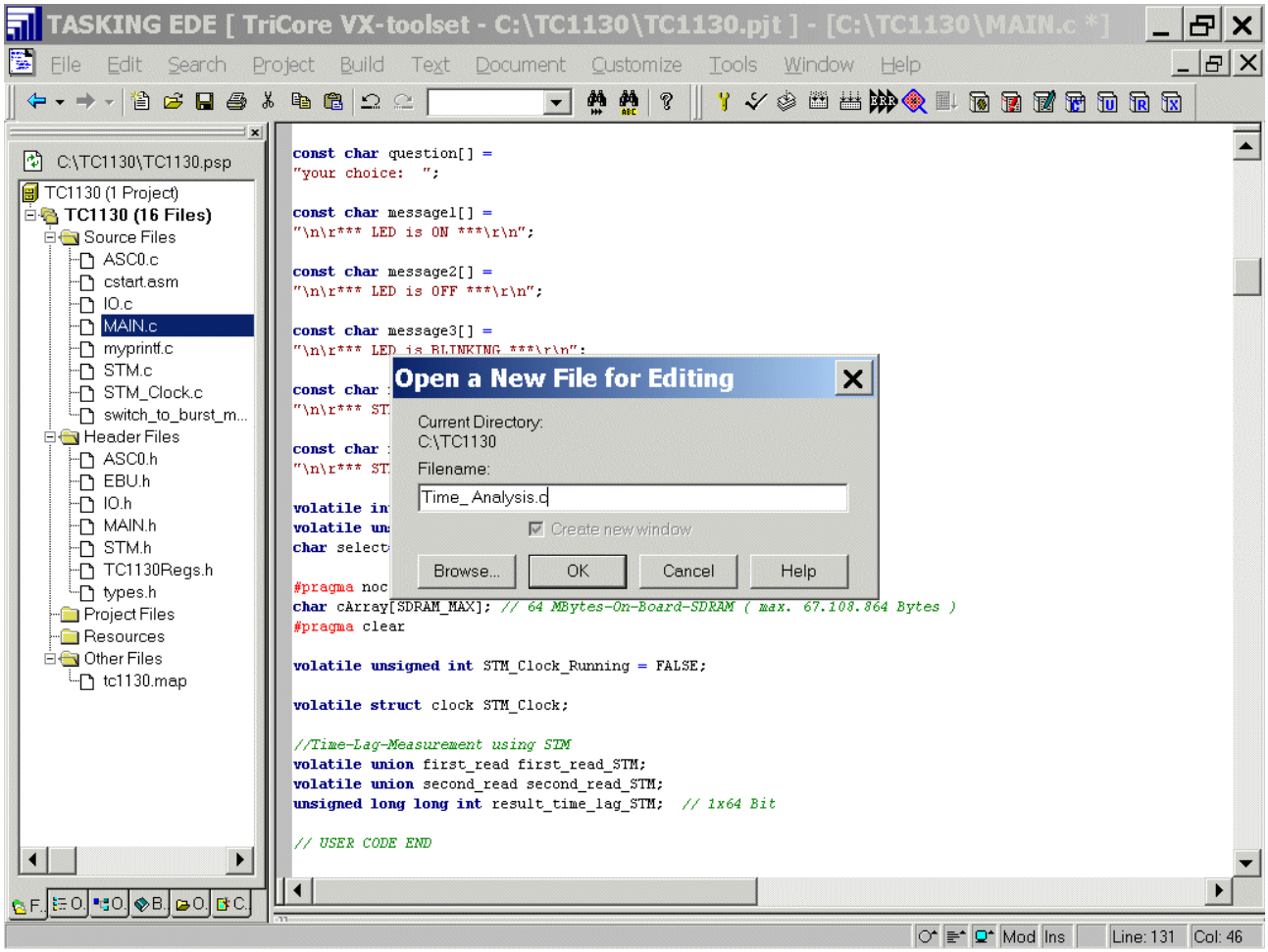


Note:
Using the 56 Bit width System-Timer (STM) with a resolution of 13,333 ns there will be an STM-Timer-Overflow / STM-Time-Range of over 30 years (see DAvE screenshot below).
Therefore we will NOT care about the STM-Overflow!

See Dave:



File – New
Insert Time_ Analysis.c



OK

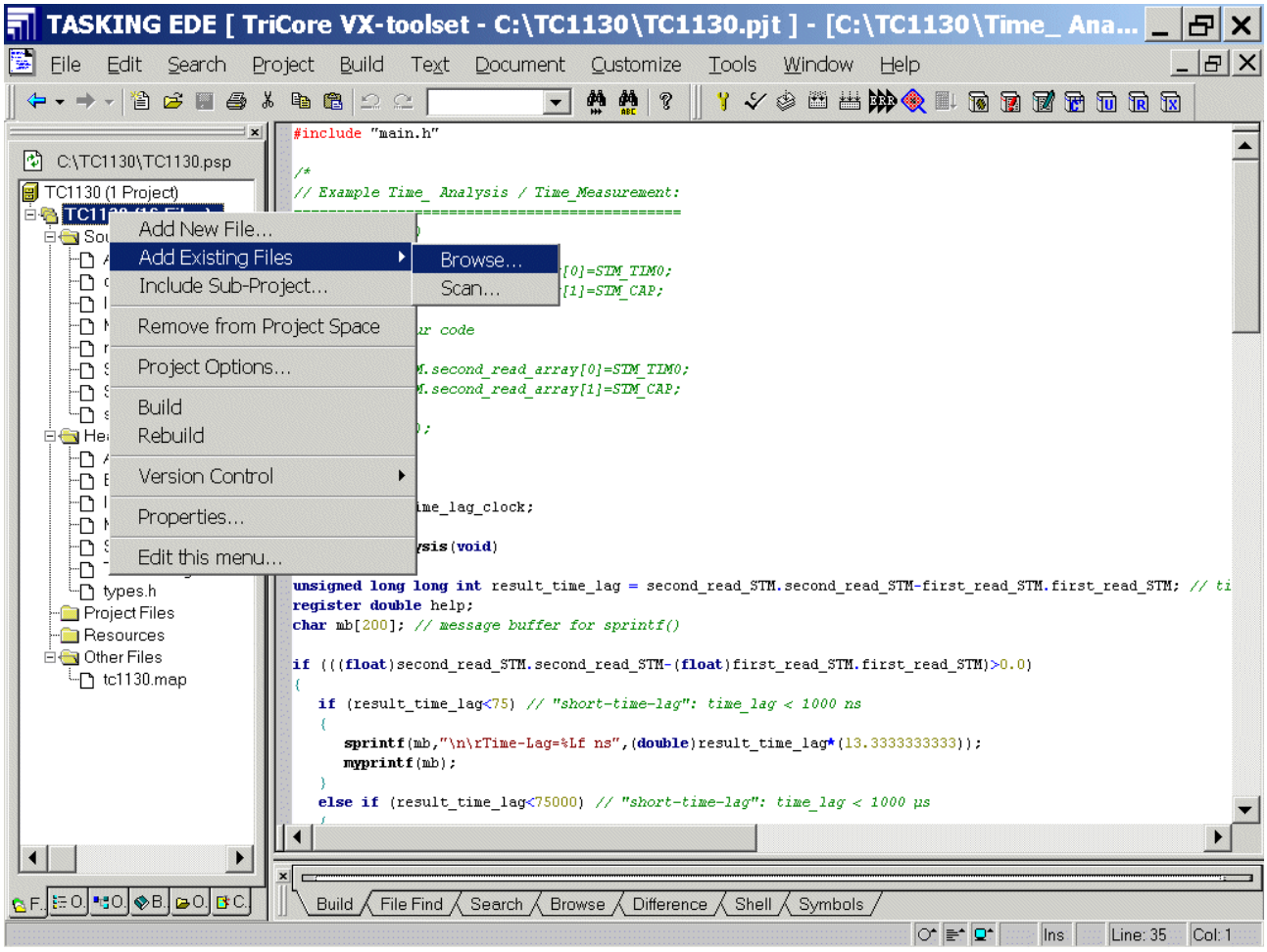
Insert User Code:

```
#include "main.h"

/*
// Example Time_ Analysis / Time_Measurement:
==========================================
void main(void)
{
first_read_STM.first_read_array[0]=STM_TIM0;
first_read_STM.first_read_array[1]=STM_CAP;

__nop(); // your code

second_read_STM.second_read_array[0]=STM_TIM0;
second_read_STM.second_read_array[1]=STM_CAP;

Time_Analysis();
}
*/

struct clock time_lag_clock;

void Time_Analysis(void)
{
unsigned long long int result_time_lag = second_read_STM.second_read_STM-
first_read_STM.first_read_STM; // time-lag in STM-Timer-Ticks
register double help;
char mb[200]; // message buffer for sprintf()

if (((float)second_read_STM.second_read_STM-(float)first_read_STM.first_read_STM)>0.0)
{
  if (result_time_lag<75) // "short-time-lag": time_lag < 1000 ns
  {
    sprintf(mb,"\n\rTime-Lag=%Lf ns",(double)result_time_lag*(13.3333333333));
    myprintf(mb);
  }
  else if (result_time_lag<75000) // "short-time-lag": time_lag < 1000 µs
  {
    sprintf(mb,"\n\rTime-Lag=%Lf µs",(double)result_time_lag*(13.3333333333*0.001));
    myprintf(mb);
  }
  else if (result_time_lag<75000000) // "short-time-lag": time_lag < 1000 ms
  {
    sprintf(mb,"\n\rTime-Lag=%Lf
ms",(double)result_time_lag*(13.3333333333*0.001*0.001));
    myprintf(mb);
  }
  else // "long-time-lag": time_lag >= 1000 ms
```

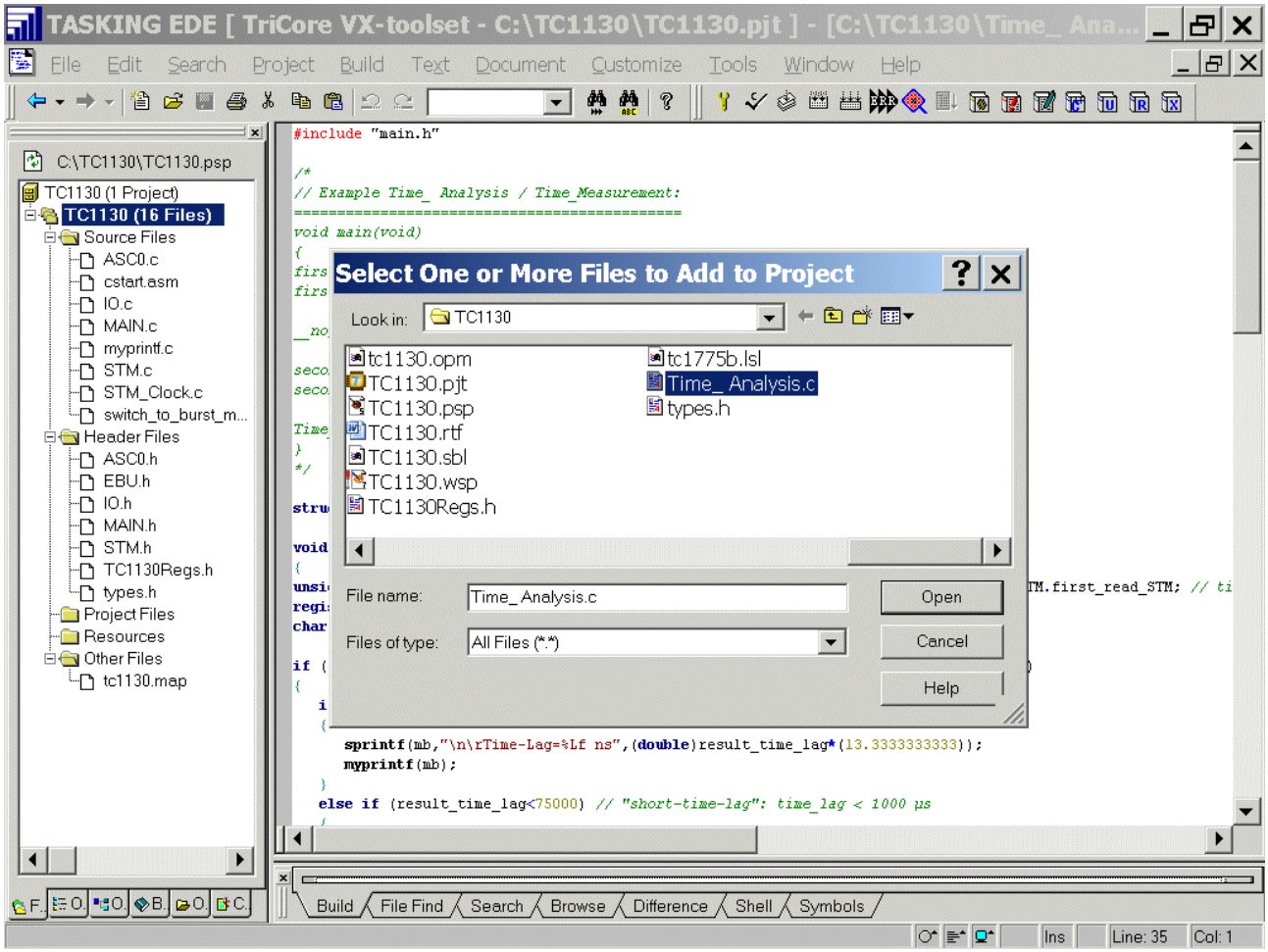
```
  {

time_lag_clock.day=time_lag_clock.hour=time_lag_clock.minute=time_lag_clock.second=0;
    help=(double)result_time_lag*(13.3333333333*0.001*0.001*0.001);
    do
    {
      ++time_lag_clock.second,help=help-1.0;
      if (time_lag_clock.second==60)
      {
        time_lag_clock.second=0;
        if (++time_lag_clock.minute==60)
        {
          time_lag_clock.minute=0;
          if (++time_lag_clock.hour==24)
            time_lag_clock.hour=0, ++time_lag_clock.day;
        }
      }
    }while (help>0.0);
    myprintf("\r\n\n*** TIME-LAG ***\r\n");
    sprintf(mb,"*** %02u:%02u:%02u
***\r",time_lag_clock.hour,time_lag_clock.minute,time_lag_clock.second);
    myprintf(mb);
  }
}
}
```

File
Save All

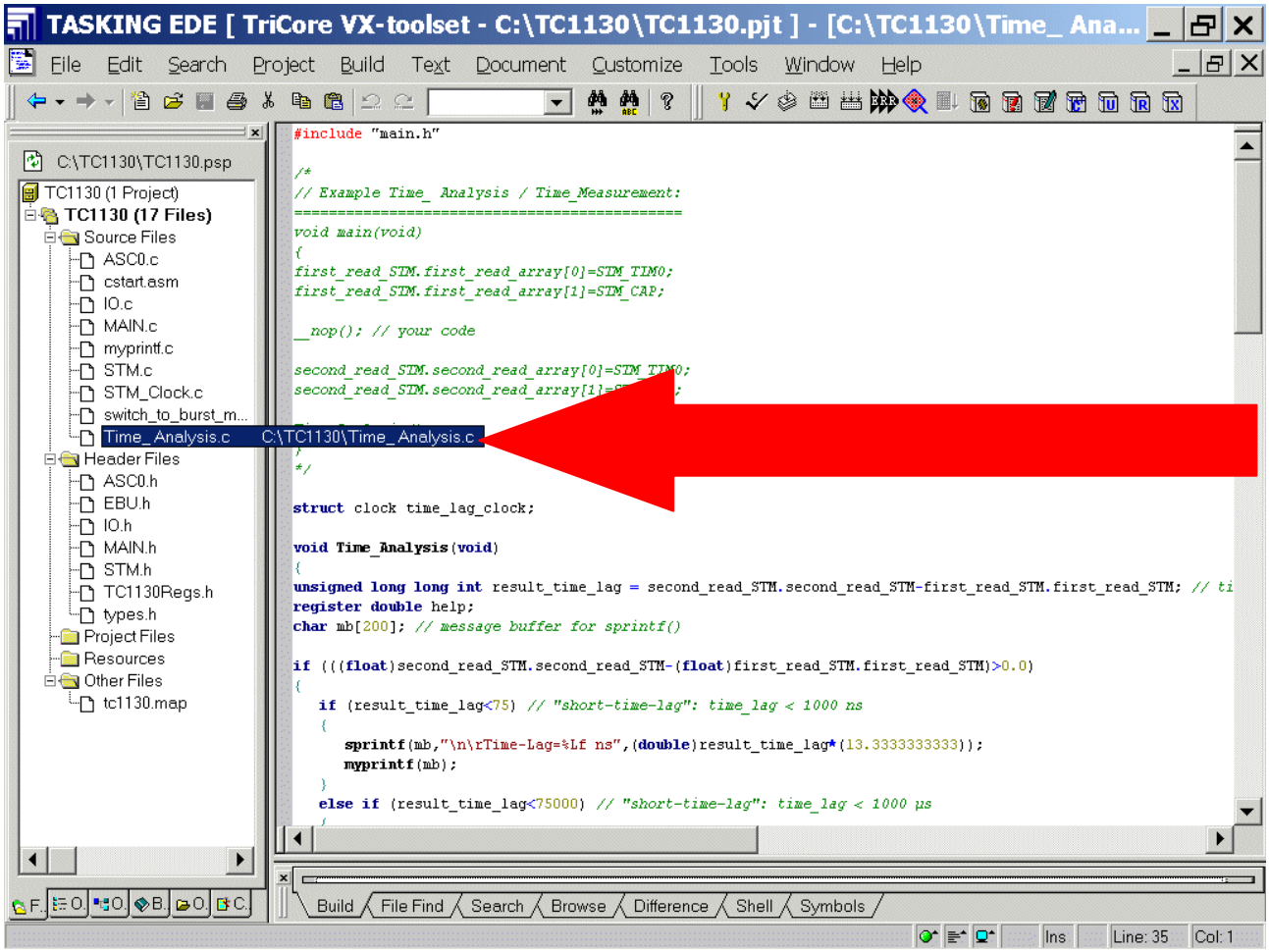
(Project Window File View) – TC1130 (Files) – right mouse button click – Add Existing Files – Browse



Select Time_ Analysis.c

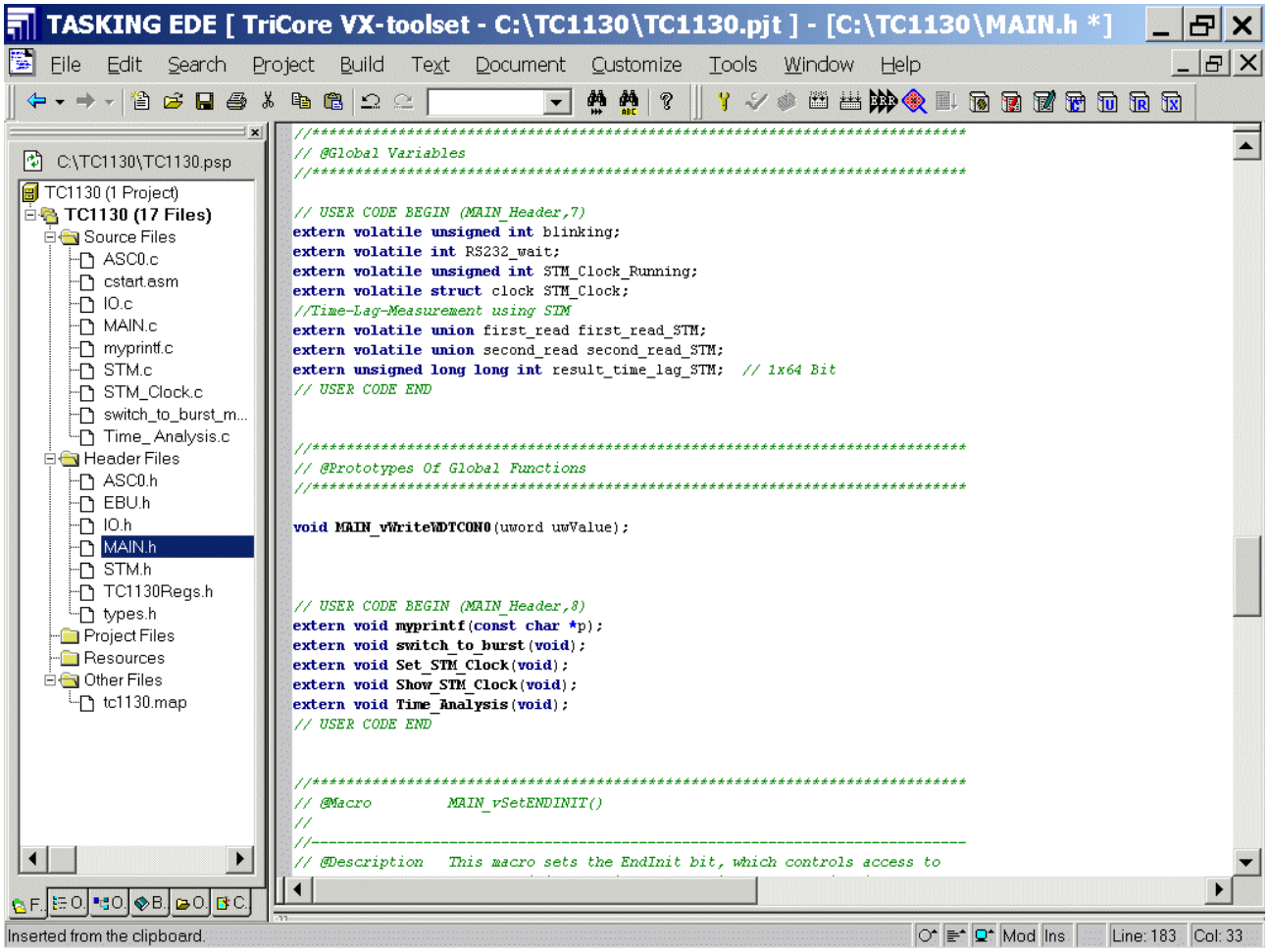


Open - OK


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\Time_ Ana...
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1130\TC1130.psp
TC1130 (1 Project)
TC1130 (17 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
STM.c
STM_Clock.c
switch_to_burst_m...
Time_ Analysis.c C:\TC1130\Time_Analysis.c
Header Files
ASC0.h
EBU.h
IO.h
MAIN.h
STM.h
TC1130Regs.h
types.h
Project Files
Resources
Other Files
tc1130.map
#include "main.h"
/*
// Example Time_ Analysis / Time_Measurement:
void main(void)
first_read_STM.first_read_array[0]=STM_TIM0;
first_read_STM.first_read_array[1]=STM_CAP;
__nop(); // your code
second_read_STM.second_read_array[0]=STM_TIM0;
*/
struct clock time_lag_clock;
void Time_Analysis(void)
unsigned long long int result_time_lag = second_read_STM.second_read_STM-first_read_STM.first_read_STM; // ti
register double help;
char mb[200]; // message buffer for sprintf()
if (((float)second_read_STM.second_read_STM-(float)first_read_STM.first_read_STM)>0.0)
if (result_time_lag<75) // "short-time-lag": time_lag < 1000 ns
sprintf(mb,"\n\rTime-Lag=%Lf ns",(double)result_time_lag*(13.3333333333));
myprintf(mb);
else if (result_time_lag<75000) // "short-time-lag": time_lag < 1000 µs
Build File Find Search Browse Difference Shell Symbols
Ins Line: 35 Col: 1

Double click: Main.h and insert Prototypes of Global Functions (Extern Declaration):

```
extern void Time_Analysis(void);
```

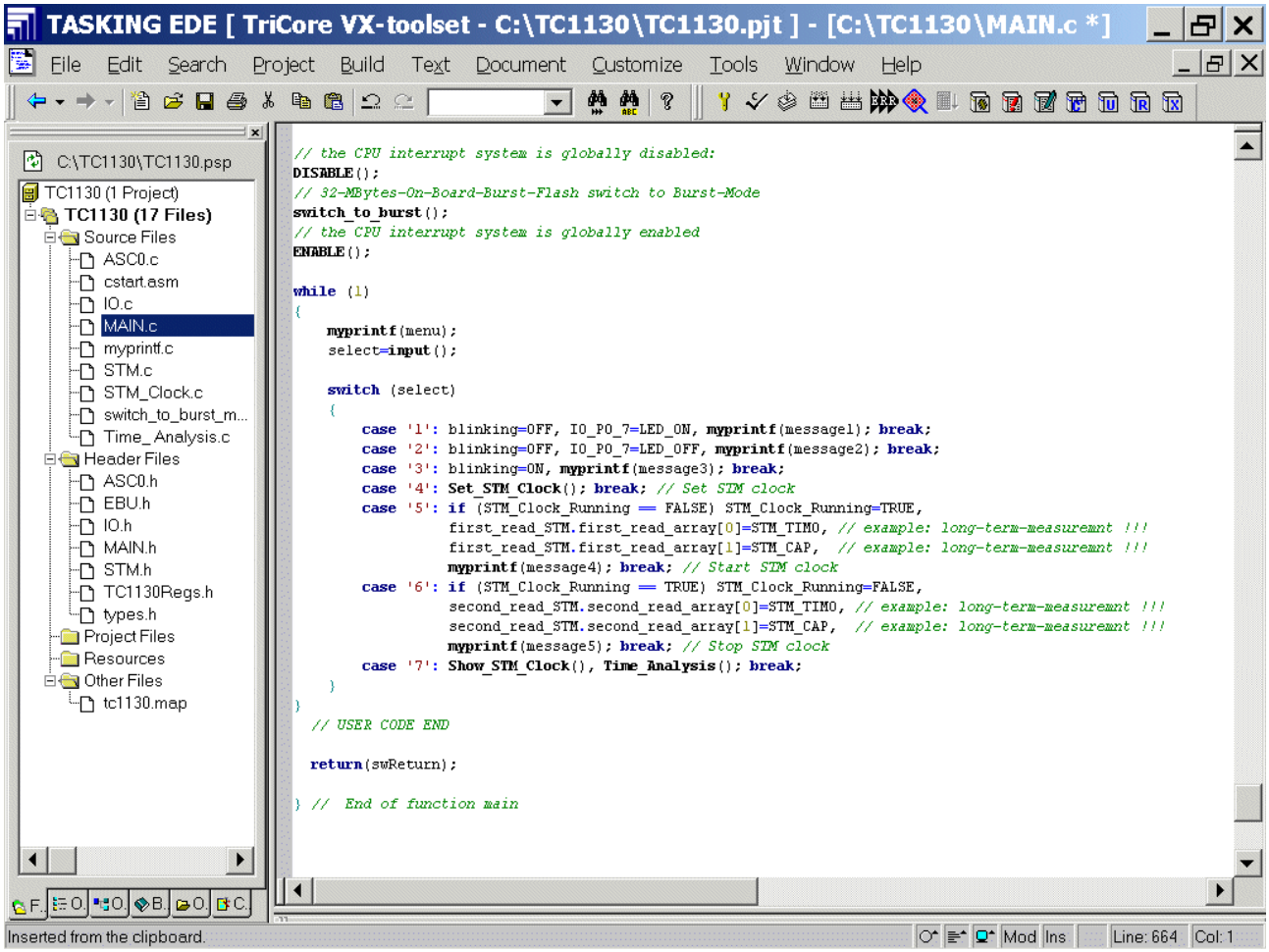


Double click: Main.c and change Code [“void main (void)” - Function]
from

```
    switch (select)
    {
        case '1': blinking=OFF, IO_P0_7=LED_ON,
myprintf(message1); break;
        case '2': blinking=OFF, IO_P0_7=LED_OFF,
myprintf(message2); break;
        case '3': blinking=ON, myprintf(message3); break;
        case '4': Set_STM_Clock(); break; // Set STM clock
        case '5': if (STM_Clock_Running == FALSE)
STM_Clock_Running=TRUE; myprintf(message4); break; // Start STM
clock
        case '6': if (STM_Clock_Running == TRUE)
STM_Clock_Running=FALSE; myprintf(message5); break; // Stop STM
clock
        case '7': Show_STM_Clock(); break; // Show STM clock
    }
```

to

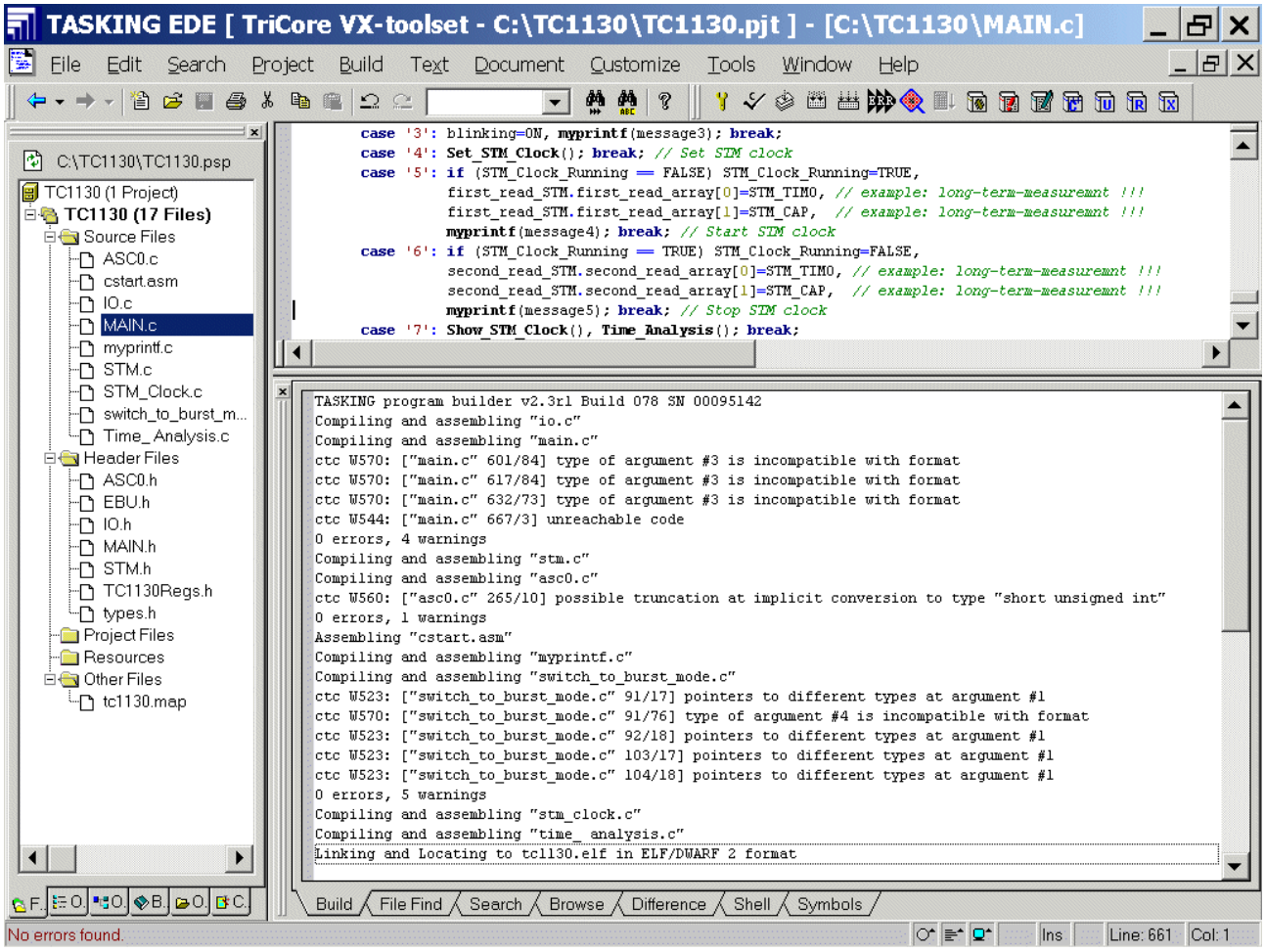
```
    switch (select)
    {
        case '1': blinking=OFF, IO P0 7=LED ON, myprintf(message1);
break;
        case '2': blinking=OFF, IO_P0_7=LED_OFF,
myprintf(message2); break;
        case '3': blinking=ON, myprintf(message3); break;
          case '4': Set_STM_Clock(); break; // Set STM clock
          case '5': if (STM_Clock_Running == FALSE)
STM_Clock_Running=TRUE,
                    first_read_STM.first_read_array[0]=STM_TIM0, //
example: long-term-measuremnt !!!
                  first_read_STM.first_read_array[1]=STM_CAP,  //
example: long-term-measuremnt !!!
                    myprintf(message4); break; // Start STM clock
          case '6': if (STM_Clock_Running == TRUE)
STM_Clock_Running=FALSE,

second_read_STM.second_read_array[0]=STM_TIM0, // example: long-
term-measuremnt !!!
                  second_read_STM.second_read_array[1]=STM_CAP,  //
example: long-term-measuremnt !!!
                    myprintf(message5); break; // Stop STM clock
          case '7': Show_STM_Clock(), Time_Analysis(); break;
    }
```


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1130\TC1130.psp
TC1130 (1 Project)
TC1130 (17 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
STM.c
STM_Clock.c
switch_to_burst_m...
Time_ Analysis.c
Header Files
ASC0.h
EBU.h
IO.h
MAIN.h
STM.h
TC1130Regs.h
types.h
Project Files
Resources
Other Files
tc1130.map
// the CPU interrupt system is globally disabled:
DISABLE();
// 32-MBytes-On-Board-Burst-Flash switch to Burst-Mode
switch_to_burst();
// the CPU interrupt system is globally enabled
ENABLE();
while (1)
{
myprintf(menu);
select=input();
switch (select)
{
case '1': blinking=OFF, IO_P0_7=LED_ON, myprintf(message1); break;
case '2': blinking=OFF, IO_P0_7=LED_OFF, myprintf(message2); break;
case '3': blinking=ON, myprintf(message3); break;
case '4': Set_STM_Clock(); break; // Set STM clock
case '5': if (STM_Clock_Running == FALSE) STM_Clock_Running=TRUE,
first_read_STM.first_read_array[0]=STM_TIM0, // example: long-term-measuremnt !!!
first_read_STM.first_read_array[1]=STM_CAP, // example: long-term-measuremnt !!!
myprintf(message4); break; // Start STM clock
case '6': if (STM_Clock_Running == TRUE) STM_Clock_Running=FALSE,
second_read_STM.second_read_array[0]=STM_TIM0, // example: long-term-measuremnt !!!
second_read_STM.second_read_array[1]=STM_CAP, // example: long-term-measuremnt !!!
myprintf(message5); break; // Stop STM clock
case '7': Show_STM_Clock(), Time_Analysis(); break;
}
}
// USER CODE END
return(swReturn);
} // End of function main
Inserted from the clipboard.
Mod Ins
Line: 664 Col: 1

Generate your application program:

Build
Rebuild

or



Now you can close both 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:\TC1130
File name: select TC1130.wsp

Open
Cancel

File – Load Program

Look in: select TC1130
File name: select TC1130.elf

Open

Click Program All

Exit
Exit

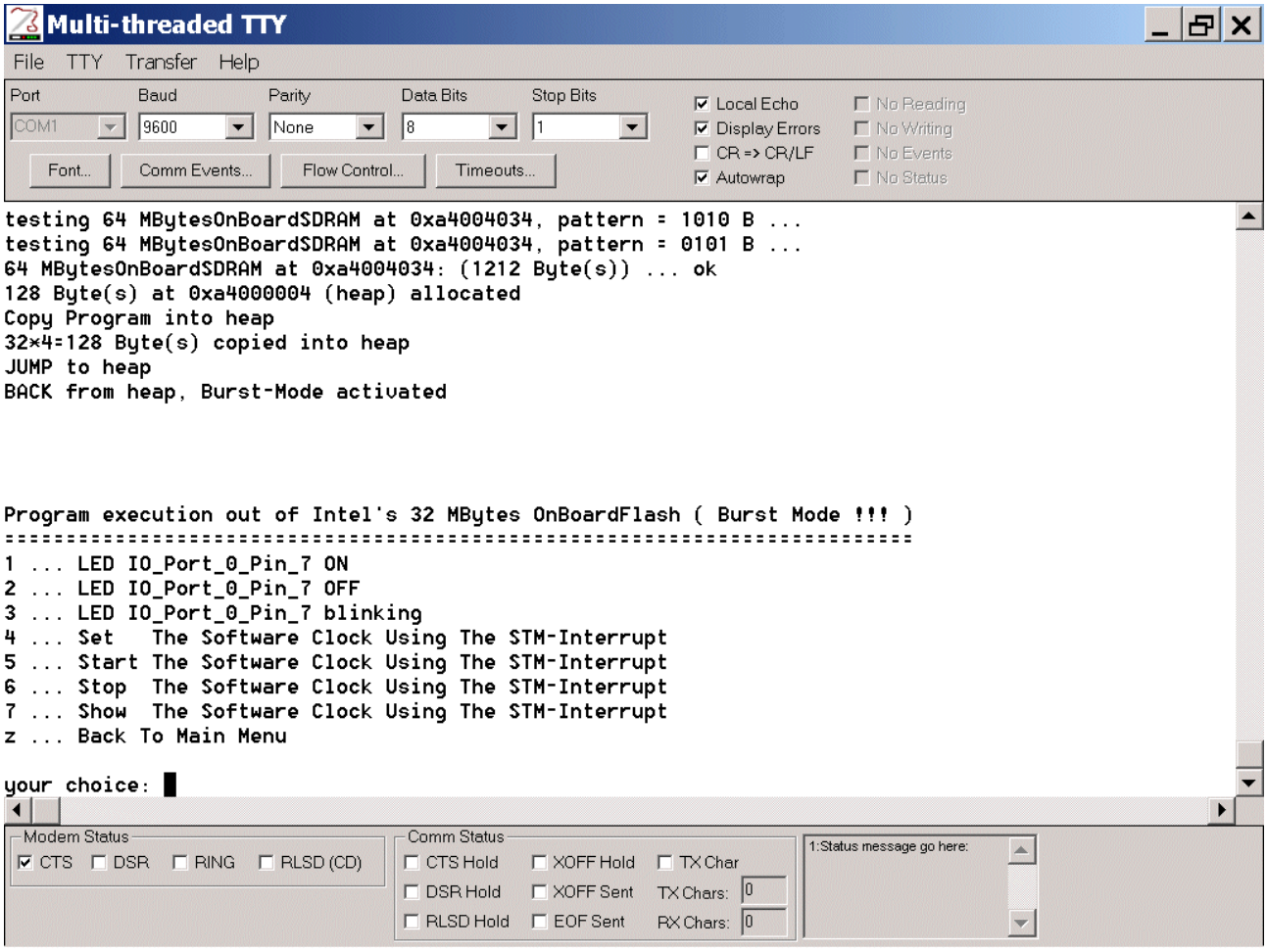
File – Close Workspace

Yes

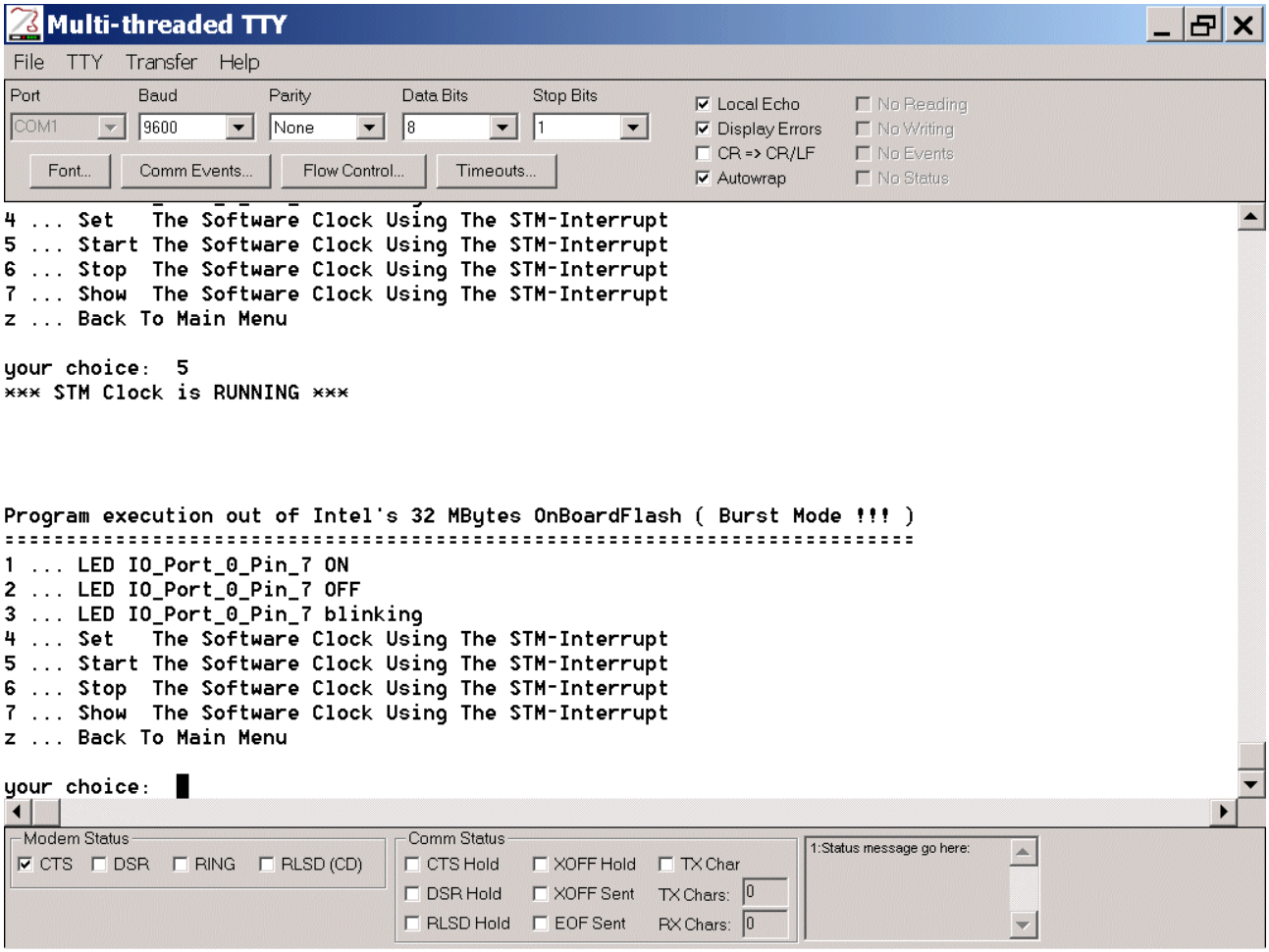
File – Exit

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

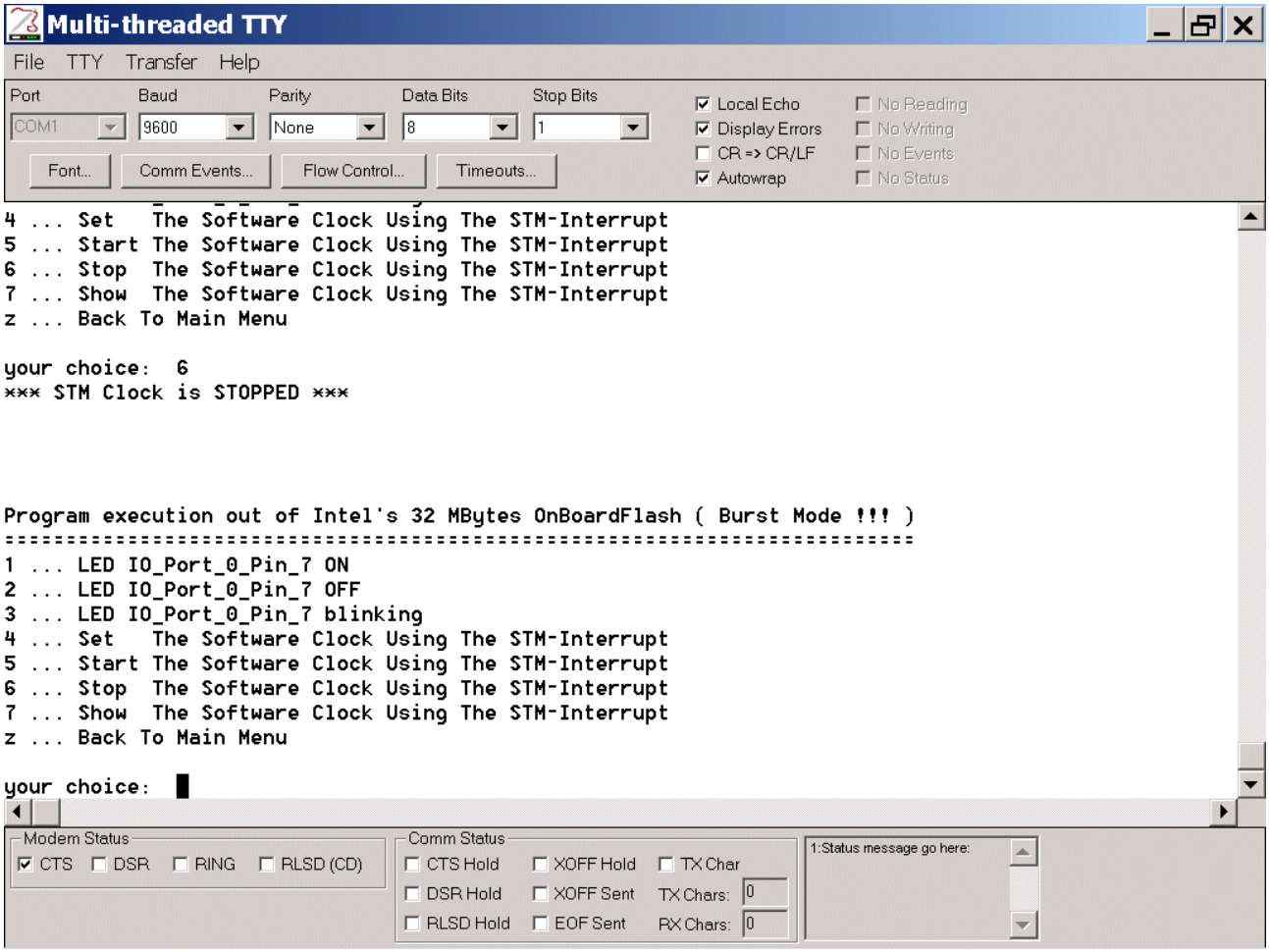
Power-On the Board and see the result:



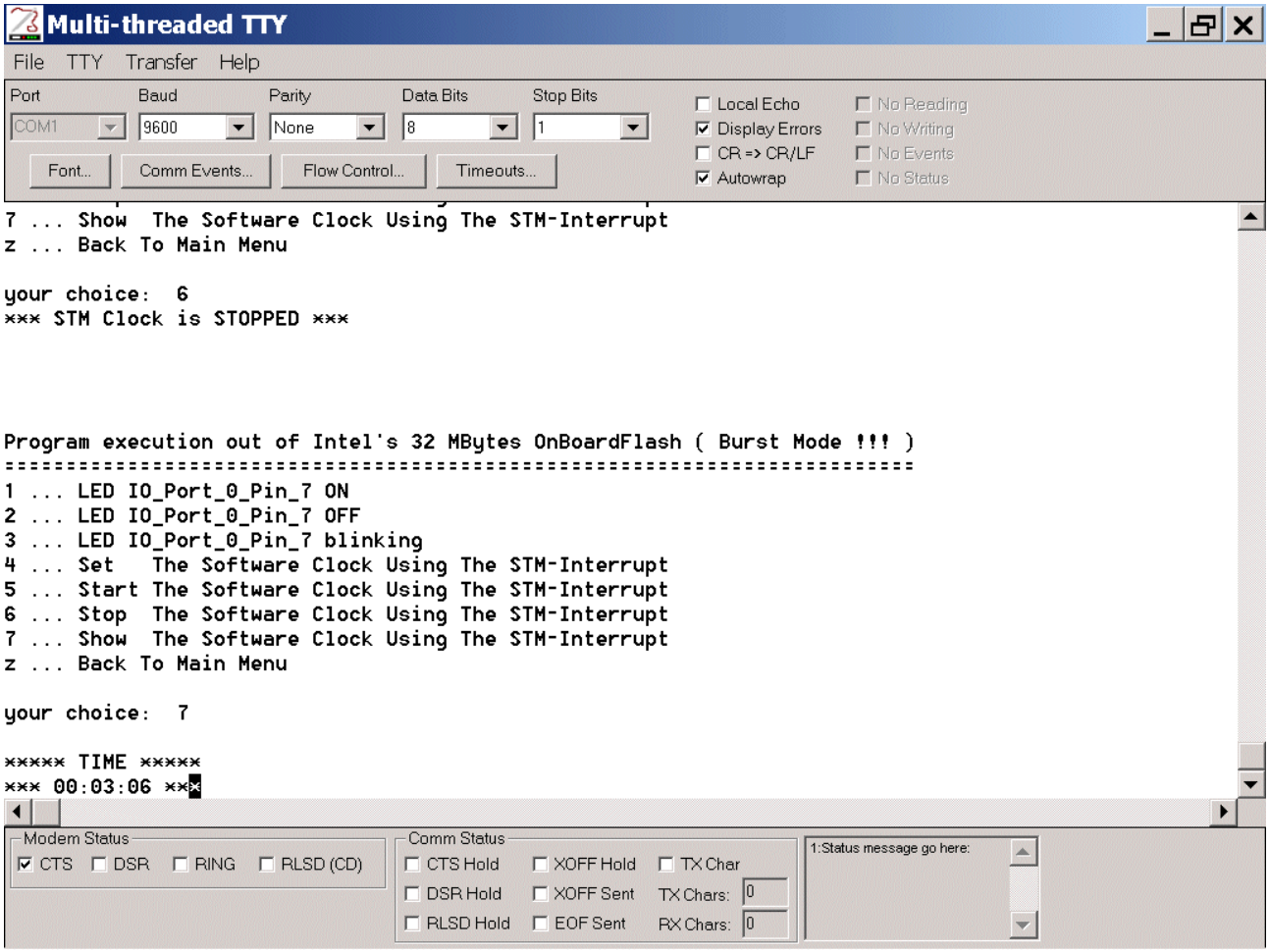
Insert 5


Multi-threaded TTY
File TTY Transfer Help
Port Baud Parity Data Bits Stop Bits
COM1 9600 None 8 1
Local Echo
Display Errors
CR => CR/LF
Autowrap
No Reading
No Writing
No Events
No Status
Font... Comm Events... Flow Control... Timeouts...
4 ... Set The Software Clock Using The STM-Interrupt
5 ... Start The Software Clock Using The STM-Interrupt
6 ... Stop The Software Clock Using The STM-Interrupt
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice: 5
*** STM Clock is RUNNING ***
Program execution out of Intel's 32 MBytes OnBoardFlash (Burst Mode !!!)
1 ... LED IO_Port_0_Pin_7 ON
2 ... LED IO_Port_0_Pin_7 OFF
3 ... LED IO_Port_0_Pin_7 blinking
4 ... Set The Software Clock Using The STM-Interrupt
5 ... Start The Software Clock Using The STM-Interrupt
6 ... Stop The Software Clock Using The STM-Interrupt
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice:
Modem Status
CTS DSR RING RLSD (CD)
Comm Status
CTS Hold XOFF Hold TX Char
DSR Hold XOFF Sent TX Chars: 0
RLSD Hold EOF Sent RX Chars: 0
1:Status message go here:

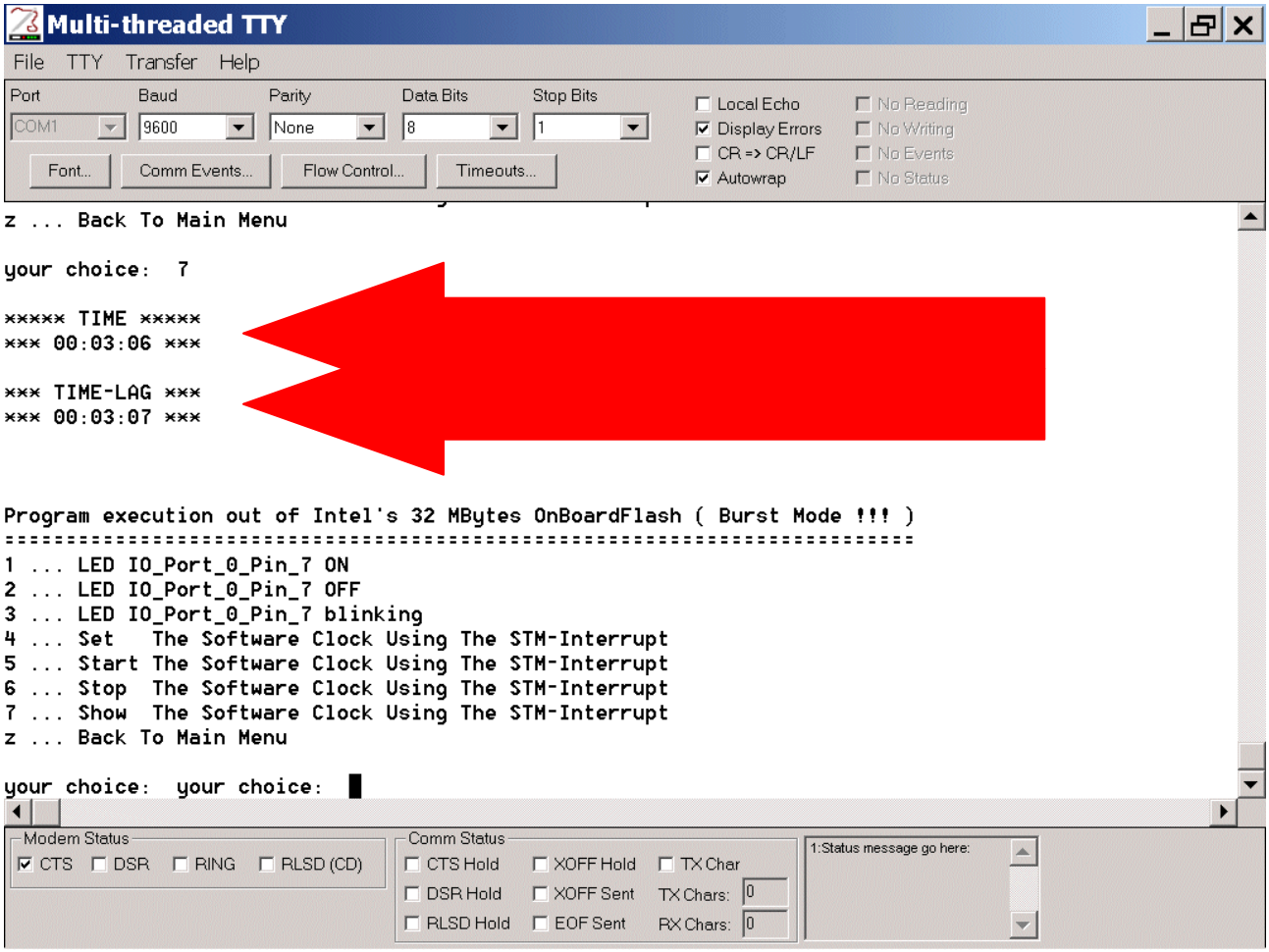
Insert 6

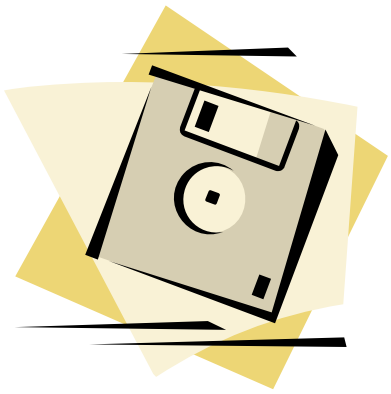

Multi-threaded TTY
File TTY Transfer Help
Port Baud Parity Data Bits Stop Bits
COM1 9600 None 8 1
Local Echo
Display Errors
CR => CR/LF
Autowrap
No Reading
No Writing
No Events
No Status
Font... Comm Events... Flow Control... Timeouts...
4 ... Set The Software Clock Using The STM-Interrupt
5 ... Start The Software Clock Using The STM-Interrupt
6 ... Stop The Software Clock Using The STM-Interrupt
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice: 6
*** STM Clock is STOPPED ***
Program execution out of Intel's 32 MBytes OnBoardFlash (Burst Mode !!!)
1 ... LED IO_Port_0_Pin_7 ON
2 ... LED IO_Port_0_Pin_7 OFF
3 ... LED IO_Port_0_Pin_7 blinking
4 ... Set The Software Clock Using The STM-Interrupt
5 ... Start The Software Clock Using The STM-Interrupt
6 ... Stop The Software Clock Using The STM-Interrupt
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice:
Modem Status
CTS DSR RING RLSD (CD)
Comm Status
CTS Hold XOFF Hold TX Char
DSR Hold XOFF Sent TX Chars: 0
RLSD Hold EOF Sent RX Chars: 0
1:Status message go here:

Insert 7

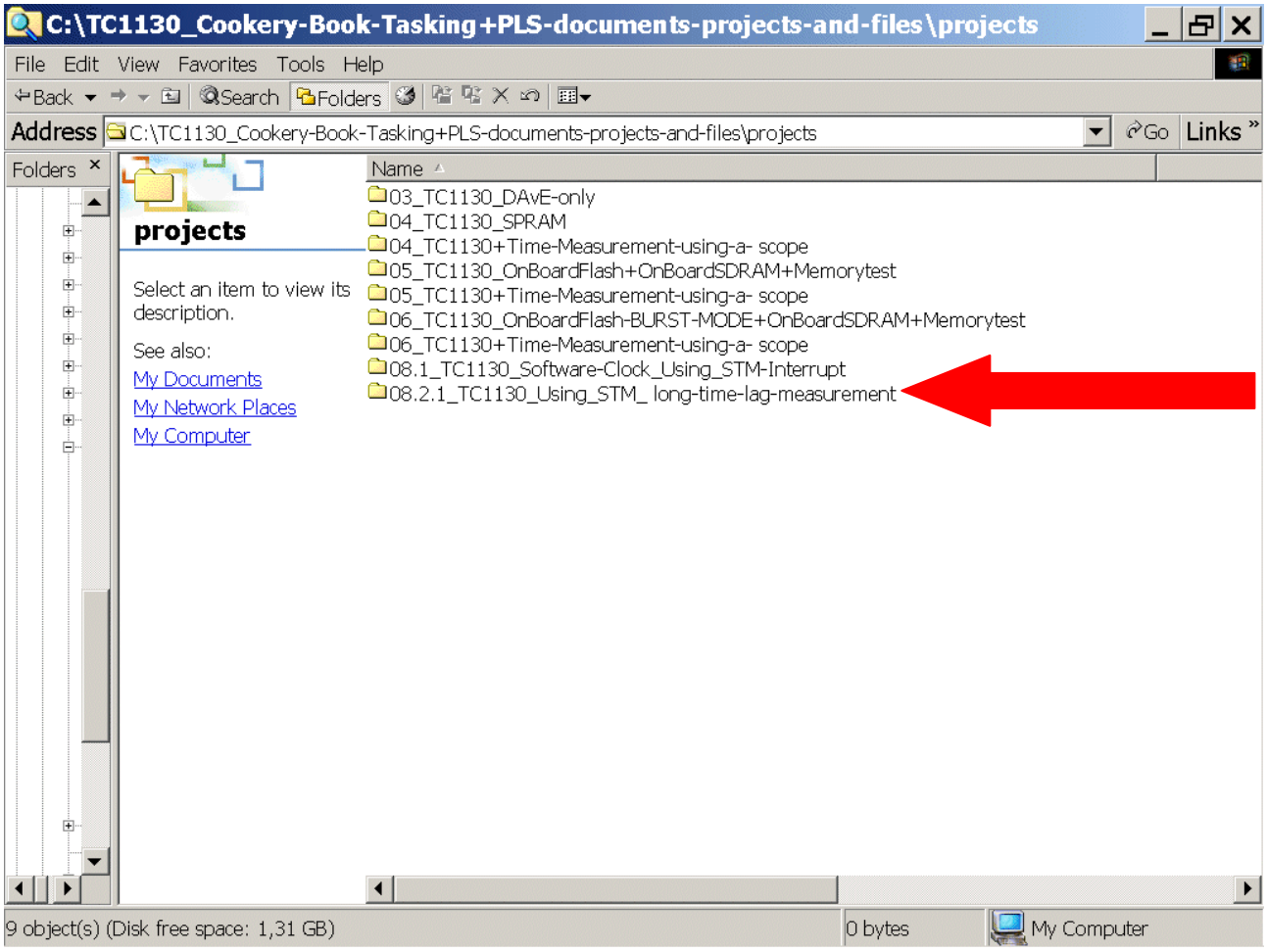

Multi-threaded TTY
File TTY Transfer Help
Port Baud Parity Data Bits Stop Bits
COM1 9600 None 8 1
Font... Comm Events... Flow Control... Timeouts...
Local Echo
Display Errors
CR => CR/LF
Autowrap
No Reading
No Writing
No Events
No Status
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice: 6
*** STM Clock is STOPPED ***
Program execution out of Intel's 32 MBytes OnBoardFlash (Burst Mode !!!)
1 ... LED IO_Port_0_Pin_7 ON
2 ... LED IO_Port_0_Pin_7 OFF
3 ... LED IO_Port_0_Pin_7 blinking
4 ... Set The Software Clock Using The STM-Interrupt
5 ... Start The Software Clock Using The STM-Interrupt
6 ... Stop The Software Clock Using The STM-Interrupt
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice: 7
***** TIME *****
*** 00:03:06 ***
Modem Status
CTS DSR RING RLSD (CD)
Comm Status
CTS Hold XOFF Hold TX Char
DSR Hold XOFF Sent TX Chars: 0
RLSD Hold EOF Sent RX Chars: 0
1:Status message go here:

Insert z


Multi-threaded TTY
File TTY Transfer Help
Port Baud Parity Data Bits Stop Bits
COM1 9600 None 8 1
Local Echo
Display Errors
CR => CR/LF
Autowrap
No Reading
No Writing
No Events
No Status
Font... Comm Events... Flow Control... Timeouts...
z ... Back To Main Menu
your choice: 7
***** TIME *****
*** 00:03:06 ***
*** TIME-LAG ***
*** 00:03:07 ***
Program execution out of Intel's 32 MBytes OnBoardFlash (Burst Mode !!!)
1 ... LED IO_Port_0_Pin_7 ON
2 ... LED IO_Port_0_Pin_7 OFF
3 ... LED IO_Port_0_Pin_7 blinking
4 ... Set The Software Clock Using The STM-Interrupt
5 ... Start The Software Clock Using The STM-Interrupt
6 ... Stop The Software Clock Using The STM-Interrupt
7 ... Show The Software Clock Using The STM-Interrupt
z ... Back To Main Menu
your choice: your choice:
Modem Status
CTS DSR RING RLSD (CD)
Comm Status
CTS Hold XOFF Hold TX Char
DSR Hold XOFF Sent TX Chars: 0
RLSD Hold EOF Sent RX Chars: 0
1:Status message go here:



We recommend now to copy and store your project-directory “C:\TC1130” to “08.2.1_TC1130_Using_STM_ long-time-lag-measurement”:



8.2.2.) Time-Lag-Measurement Using the STM (example: “short-time-lag-measurement”)

Start Tasking EDE and open the project:

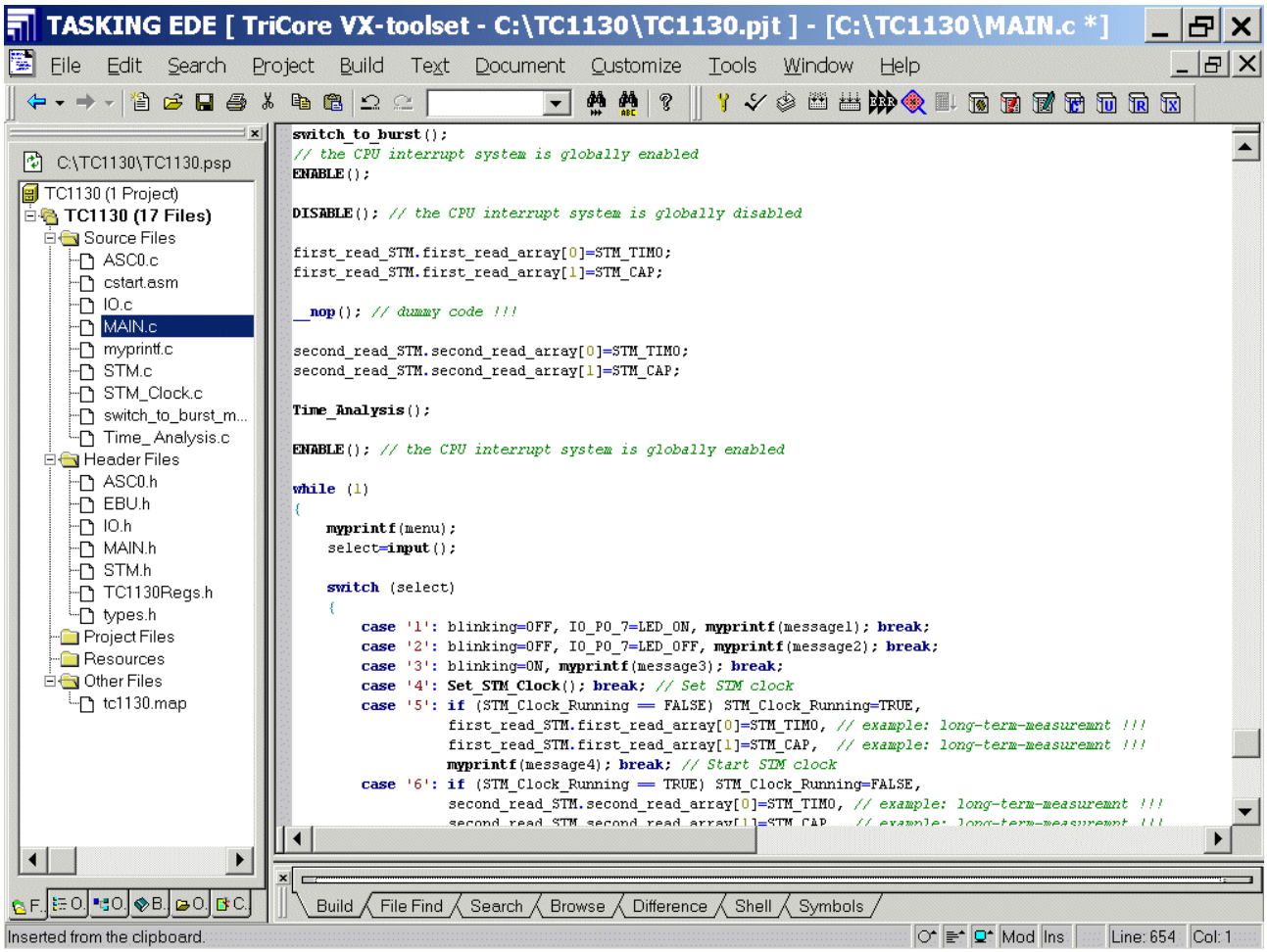
File – Open Project Space

Look in: select C:\TC1130
File name: select TC1130.psp

Open

Double click: Main.c insert Code [before ”while(1) {;}”]:

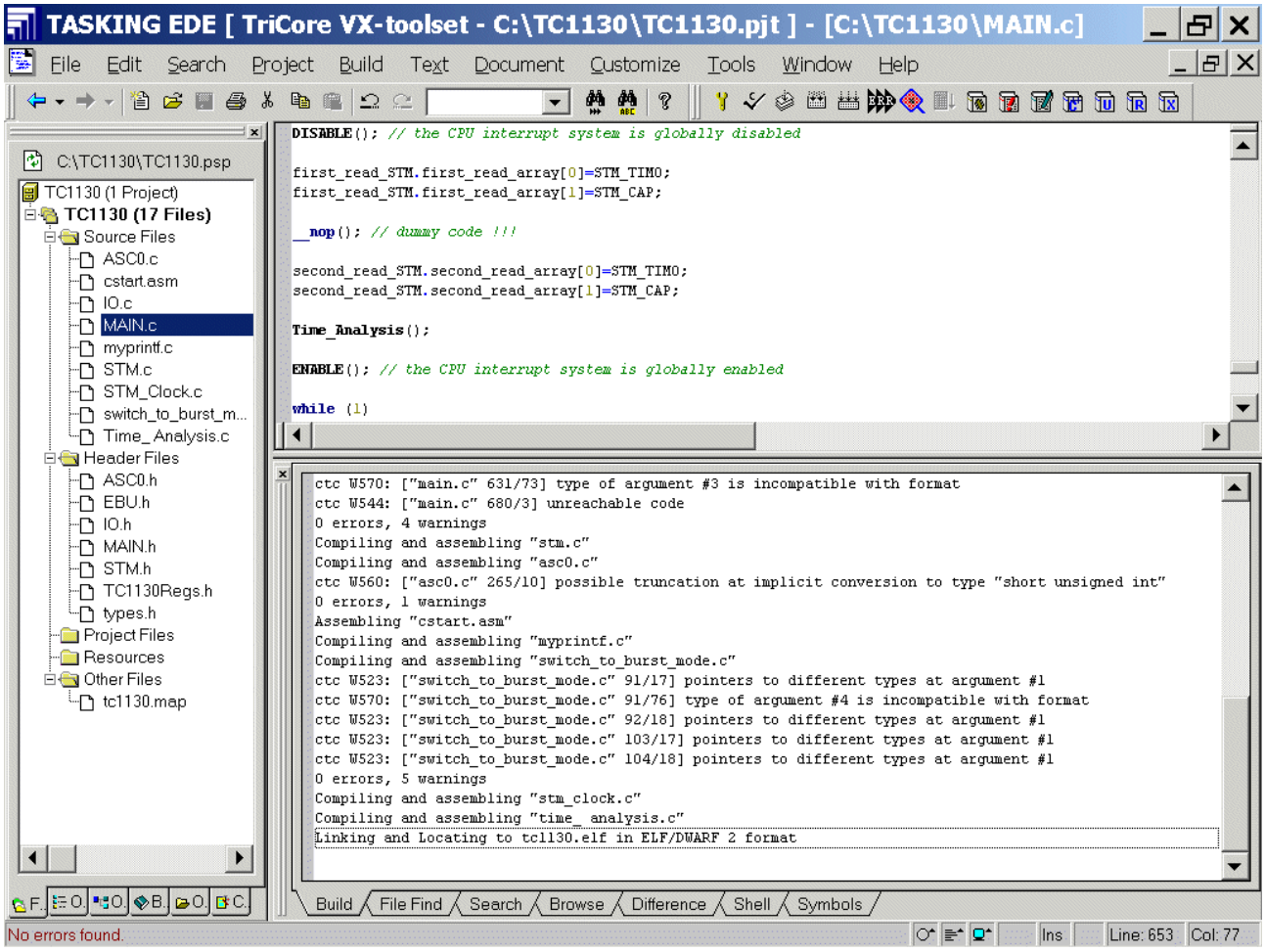
```
DISABLE(); // the CPU interrupt system is globally disabled

first_read_STM.first_read_array[0]=STM_TIM0;
first_read_STM.first_read_array[1]=STM_CAP;

__nop(); // dummy code !!!

second_read_STM.second_read_array[0]=STM_TIM0;
second_read_STM.second_read_array[1]=STM_CAP;

Time_Analysis();

ENABLE(); // the CPU interrupt system is globally enabled
```



Generate your application program:

Build
Rebuild

or



Now you can close both 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:\TC1130
File name: select TC1130.wsp

Open
Cancel

File – Load Program

Look in: select TC1130
File name: select TC1130.elf

Open

Click Program All

Exit
Exit

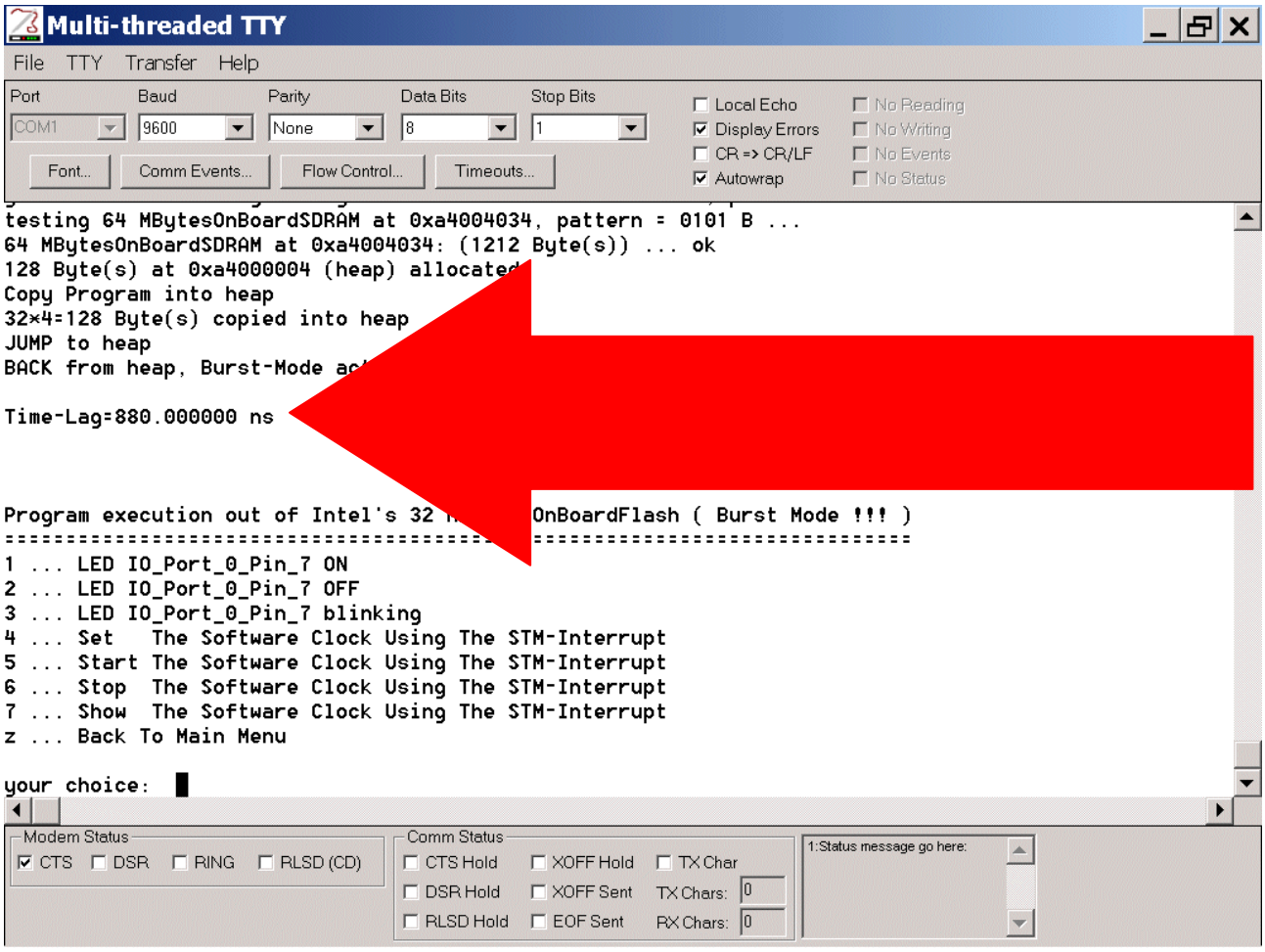
File – Close Workspace

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:



Start Tasking EDE and open the project:

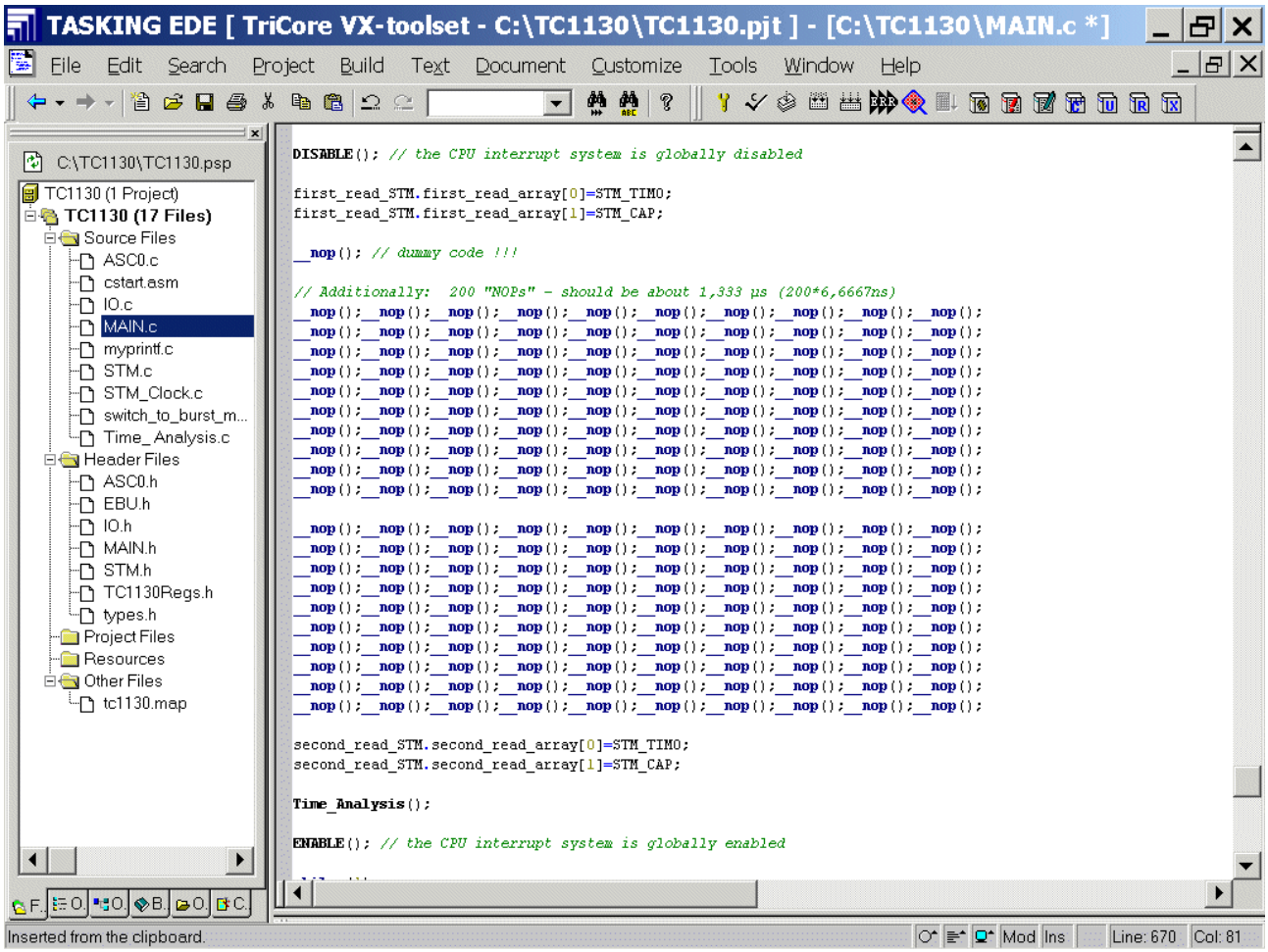
File – Open Project Space

Look in: select C:\TC1130
File name: select TC1130.psp

Open

Double click: Main.c insert Code:

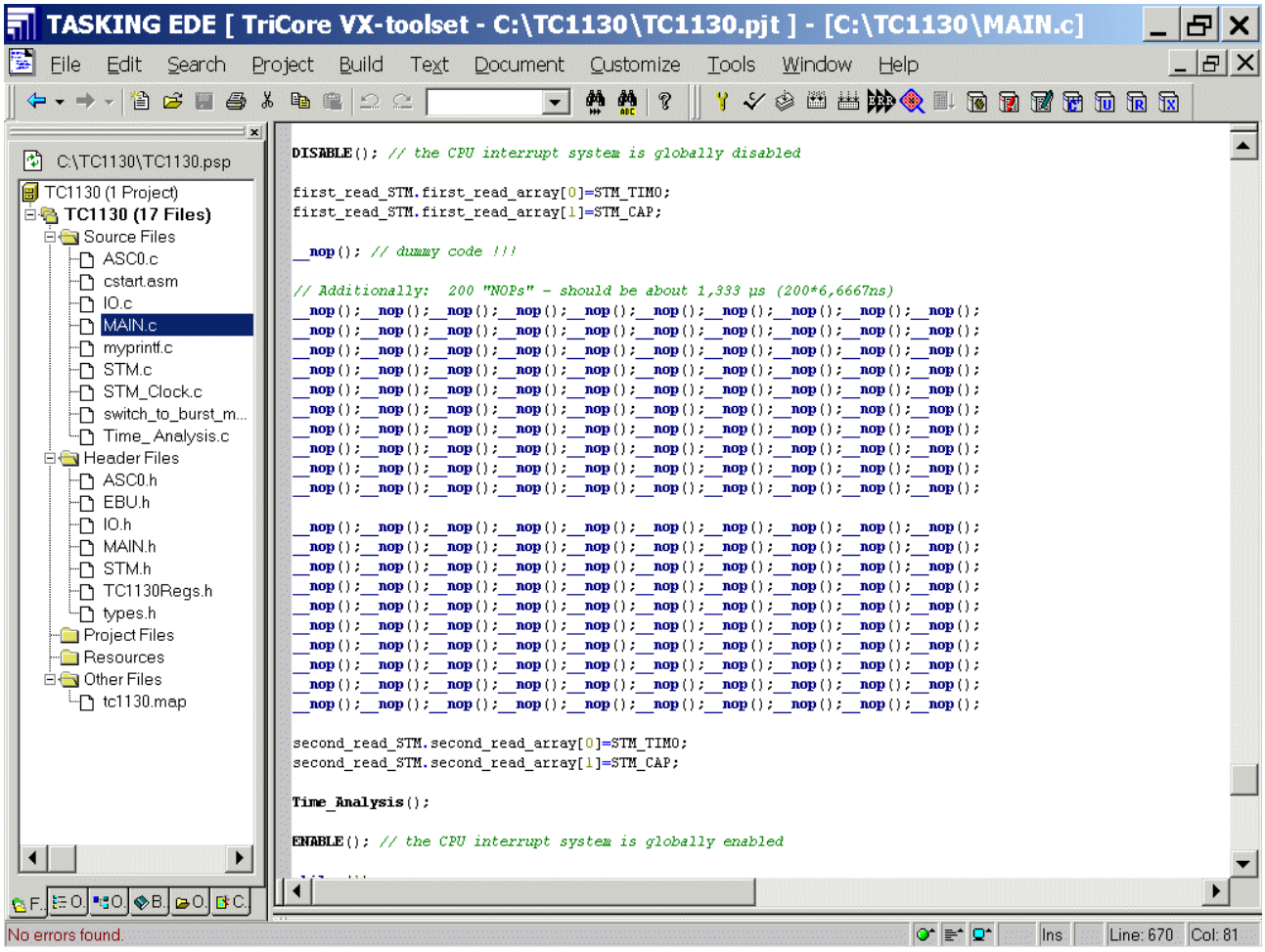
```
// Additionally:  200 “NOPs” - should be about 1,333 µs (200*6,6667ns)
__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();
__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__nop();__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();__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();__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();__nop();__nop();__nop();__nop();
```


TASKING EDE [TriCore VX-toolset - C:\TC1130\TC1130.pjt] - [C:\TC1130\MAIN.c *]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\TC1130\TC1130.psp
TC1130 (1 Project)
TC1130 (17 Files)
Source Files
ASC0.c
cstart.asm
IO.c
MAIN.c
myprintf.c
STM.c
STM_Clock.c
switch_to_burst_m...
Time_Analysis.c
Header Files
ASC0.h
EBU.h
IO.h
MAIN.h
STM.h
TC1130Regs.h
types.h
Project Files
Resources
Other Files
tc1130.map
DISABLE(); // the CPU interrupt system is globally disabled
first_read_STM.first_read_array[0]=STM_TIM0;
first_read_STM.first_read_array[1]=STM_CAP;
__nop(); // dummy code !!!
// Additionally: 200 "NOPs" - should be about 1,333 µs (200*6,6667ns)
second_read_STM.second_read_array[0]=STM_TIM0;
second_read_STM.second_read_array[1]=STM_CAP;
Time_Analysis();
ENABLE(); // the CPU interrupt system is globally enabled
Inserted from the clipboard.
Mod Ins
Line: 670 Col: 81

Generate your application program:

Build
Rebuild

or



Now you can close both 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:\TC1130
File name: select TC1130.wsp

Open
Cancel

File – Load Program

Look in: select TC1130
File name: select TC1130.elf

Open

Click Program All

Exit
Exit

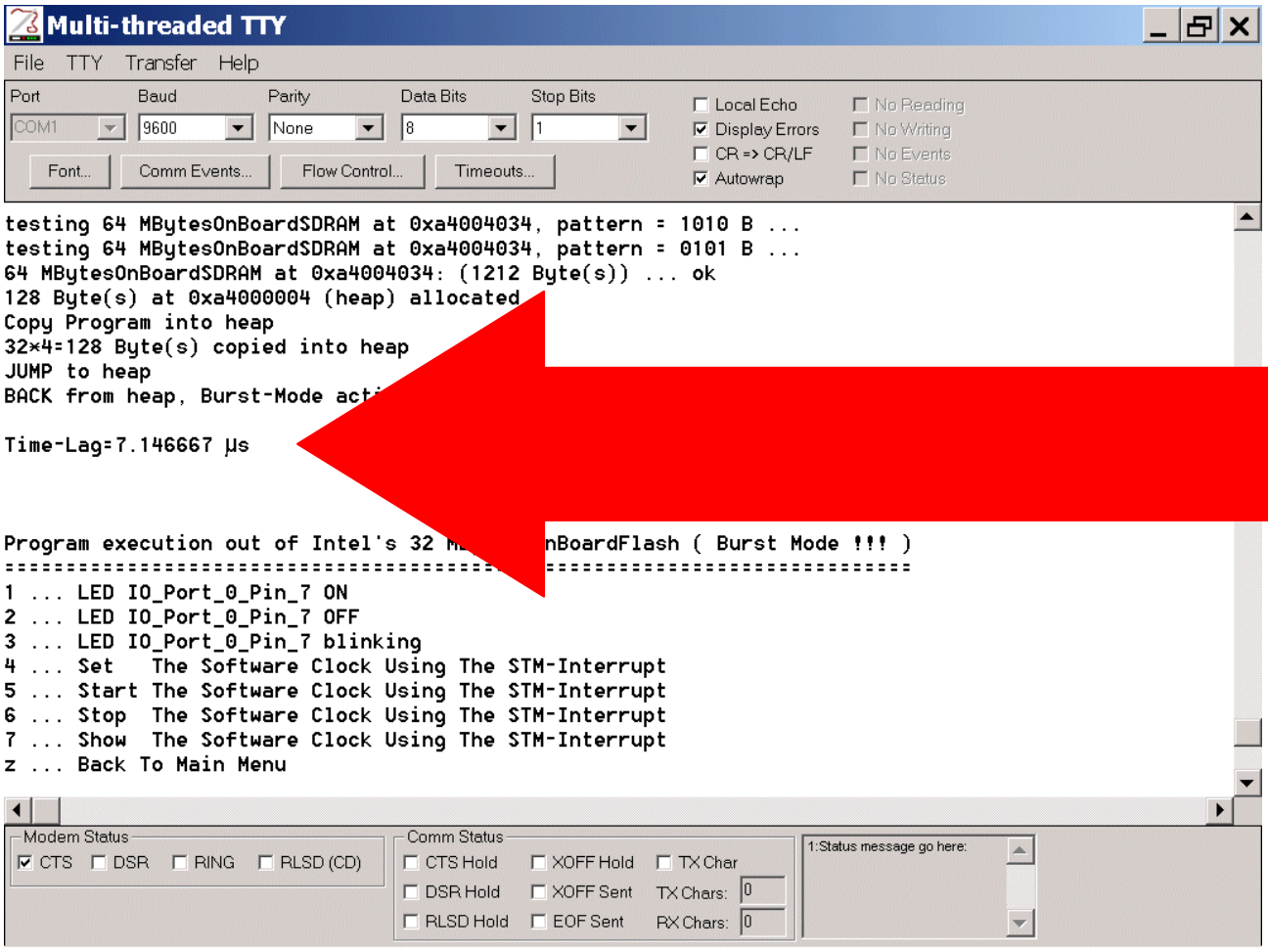
File – Close Workspace

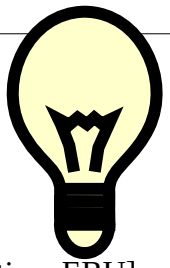
Yes

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:





Conclusion:

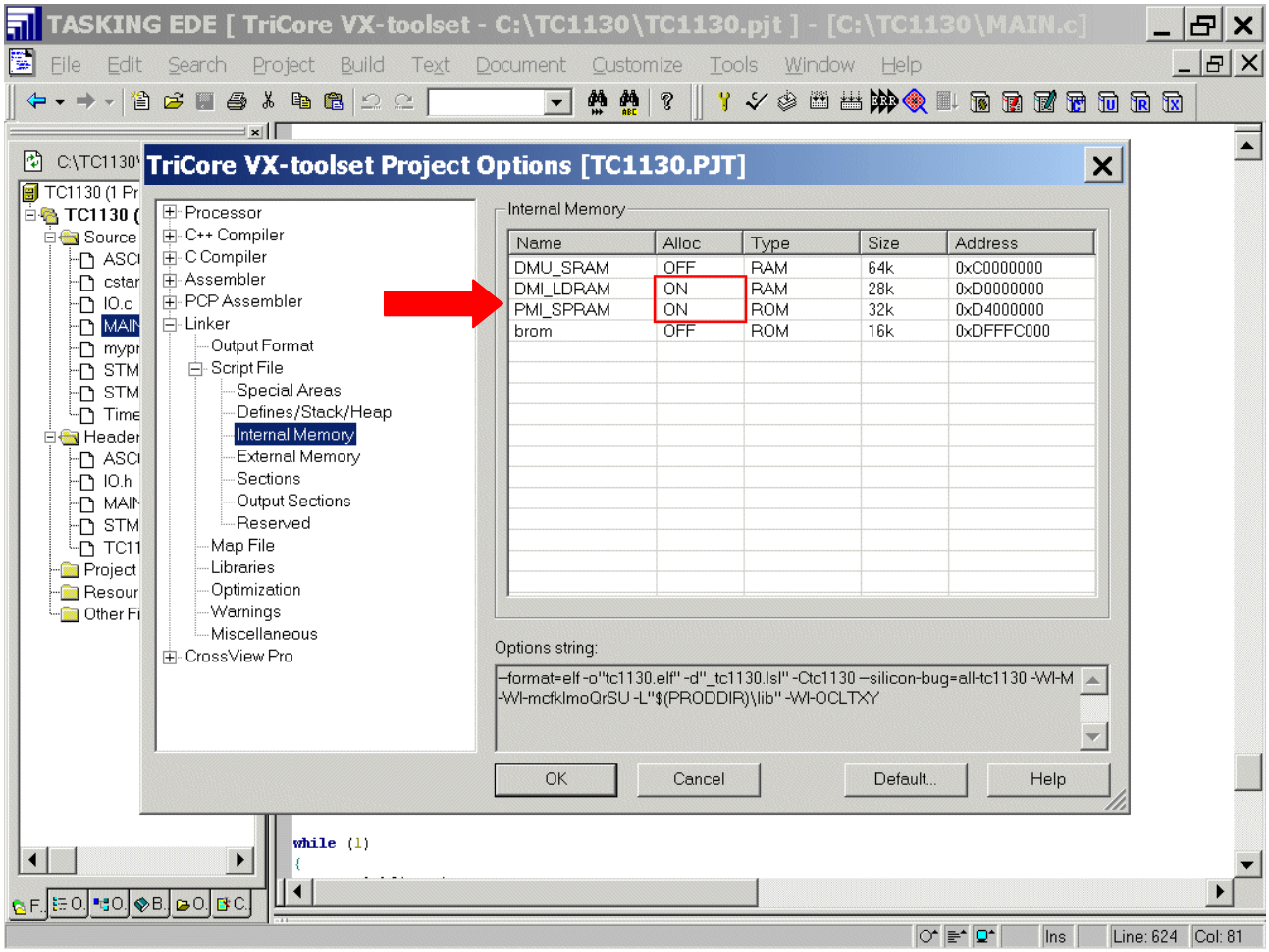
We expected 1,33 µs for additional 200 NOPs (200 * 6.667 ns).
We got 7,146 µs – 880 ns = 6,3 µs instead of 1,33 µs [Cause: LMB Bus Arbitration, EBU].

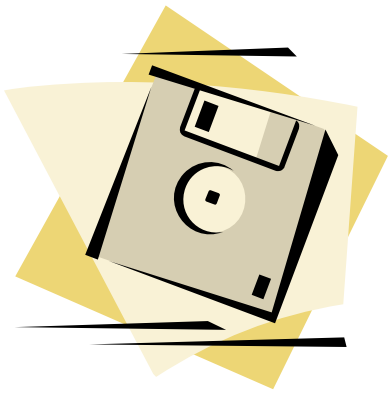
Just to satisfy curiosity:
Is it possibly to measure even CPU clocks (150 MHz -> 6,6667 ns) ?

We used the programming example "04_TC1130_SPRAM", deactivated all memories - except PMI_SPRAM and DMI_SPRAM, added software from chapter 8.1. upwards and got the following values:
1,546667 µs – 213 ns = 1,3337 µs – THIS IS EXACTLY THE TIME WE CALCULATED FOR 200 NOPs (1,3337 µs / 200 = 6,67 ns = ONE CPU CLOCK).

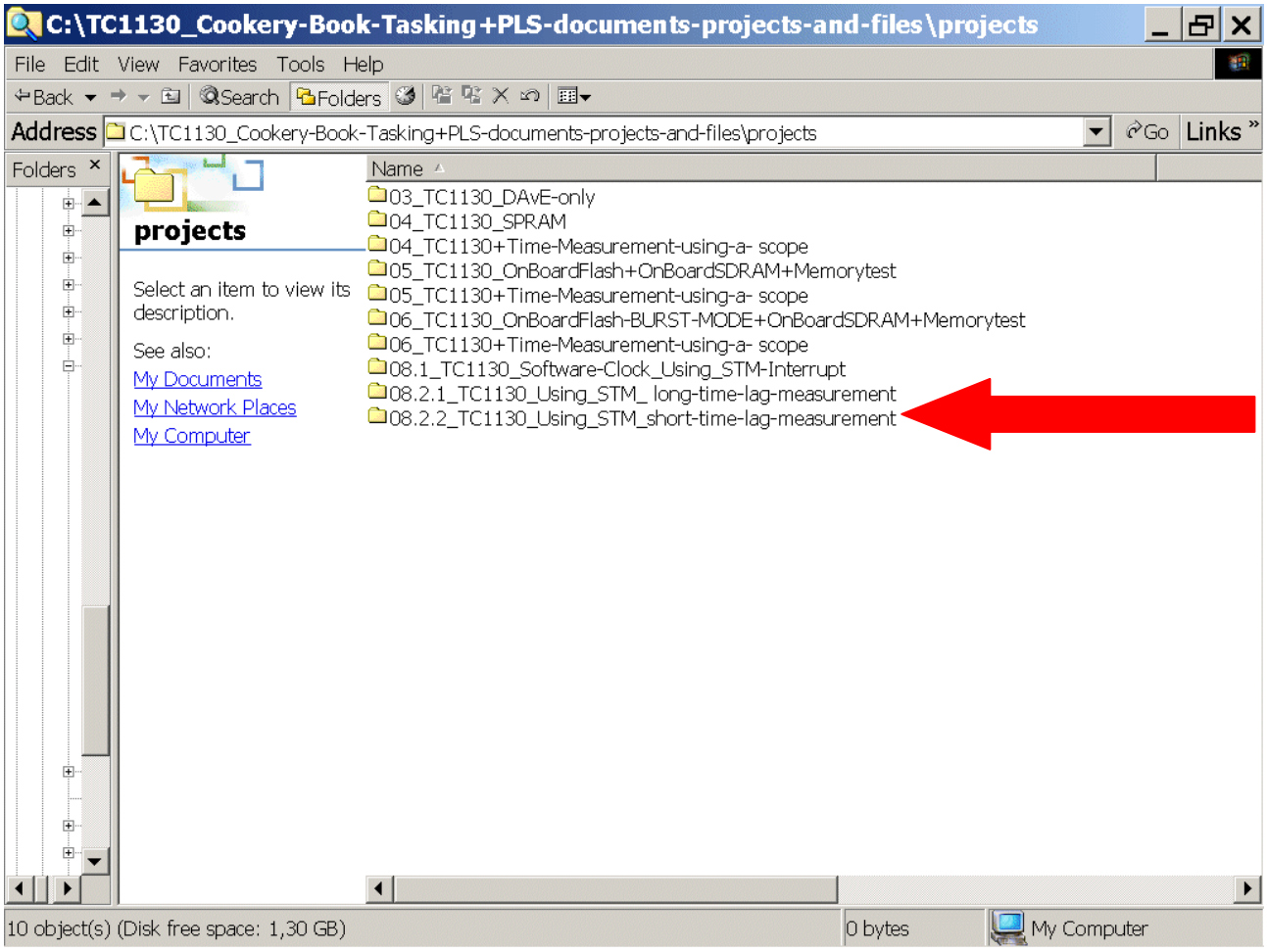
We recommend to store this additional programming example as "08.2.2_TC1130_Using_STM_short-time-lag-measurement_SPRAM-ONLY".

What we have learnt:
LMB-Bus-Arbitration and External-Memory-Access via EBU - which both is necessary for Program-Execution out-of On-Board-Flash takes longer than execution out of internal memory.

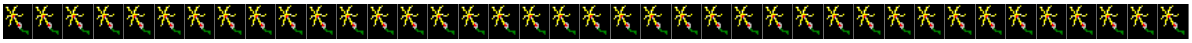




We recommend now to copy and store your project-directory “C:\TC1130” to “08.2.2_TC1130_Using_STM_short-time-lag-measurement”:



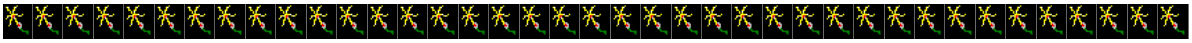
9.) Feedback (TC1130): Your opinion, suggestions and/or criticisms



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