

# "Cookery-book" for the XC164CS

Wilhelm Brezovits

## Introduction

„Cookery-book“ for your first programming example for the XC164CS Step AA Starterkit-Board in 9 Steps:

## Recipes

### I.) Application

```
1 ... LED P9_P4 ON
2 ... LED P9_P4 OFF
3 ... LED P9_P4 blinking
your choice: _
```

XC16x Board

Power Supply

the correct Jumper Setting (please refer to the XC16Board Hardware Manual) serial cable to the notebook

### II.) DAVe - program generator

- you need a installed version of the DAVe 2 CD ROM
- you need the DAVe Update for XC16x microcontrollers (XC164CS\_v21.DIP)

### III.) Using DAVe

microcontroller initialization for your programming example (either:)

#### IV.)

using the KEIL Development Tools (C/EC++ Compiler) Programming of your application (e10k) with KEIL tool chain (µVision2) (or:)

#### V.)

using the TASKING Development Tools (C/C++/EC++ Compiler) Programming of your application (e10t) with TASKING (Altium) tool chain (EDE)

### VI.) MEMTOOL Installation

You need this Tool for the programming of the On-ChipFlash in the XC164CS.

### VII.) Using MEMTOOL

- for KEIL: programming of e10k.h86 into the OnChipFlash memory
- for TASKING: programming of e10t.hex into the OnChipFlash memory

### VIII.) HyperTerminal - Settings

using a terminal program (settings)

### IX.) Using HyperTerminal

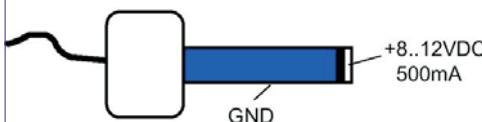
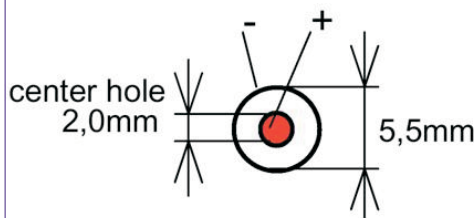
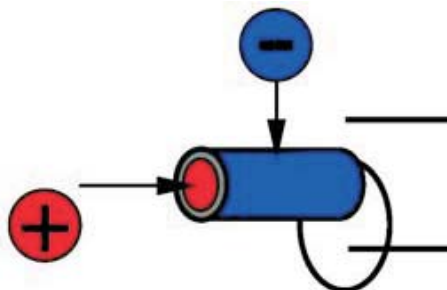
Screenshots of your running application

## I.) XC16x Board

Please refer to the XC16Board Hardware Manual.

### 1.) Power Supply

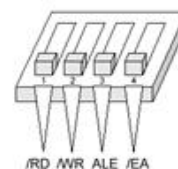
The XC16x Board requires an external



power supply.

A regulated DC power supply from 9 to 12 Volts can be connected to the power

### Dip Switch S106



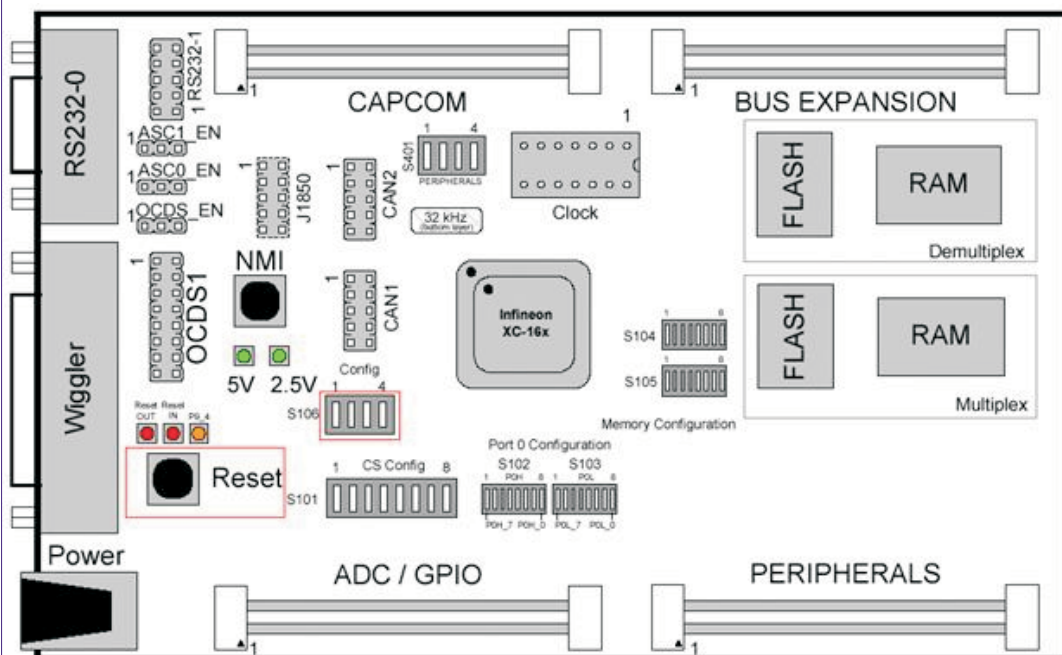
ALE='0'	/EA='1'	
	RD='0'	RD='1'
	Standard Boot (bootstrap loader)	Standard internal start

connector.

### 2.) Bootstrap Loader ⇔ Standard Internal Start

## II.) DAVe - Update Installation for XC16x microcontrollers

- 1.) Copy the files XC161CJ\_v21.DIP and XC164CS\_v21.DIP under the installation path of DAVe (C:\Programme\DAVe\).
- 2.) Start DAVe
- 3.)

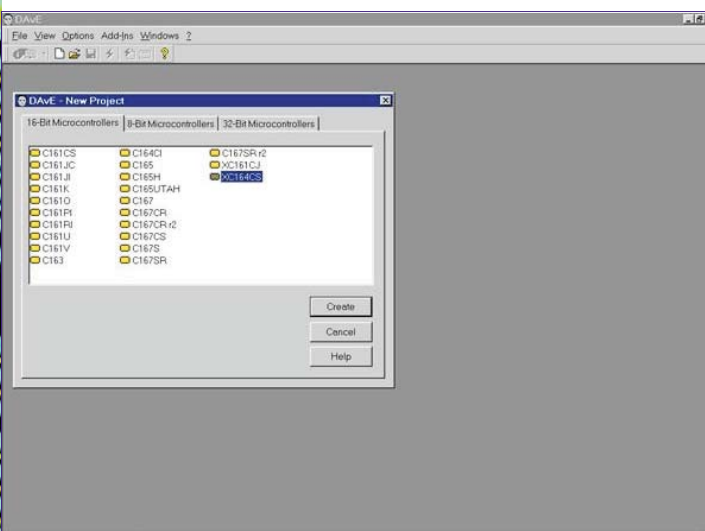


- View
  - Setup Wizard
  - Default: \* Installation
  - Forward>
  - Select: \* I want to install products from the DAVe's web site
  - Forward>
  - Select: C:\Programme\DAVe
  - Forward>
  - Select: Available Products
  - ✓ XC161CJ
  - ✓ XC164CS
  - Forward>
  - Install
  - End
- DAVe is now ready to generate code for XC161CJ and XC164CS microcontrollers.

### III.) DAVe - Initialization after Power-On

#### 1. Step: Start the programm generator DAVe and select the XC164CS microcontroller

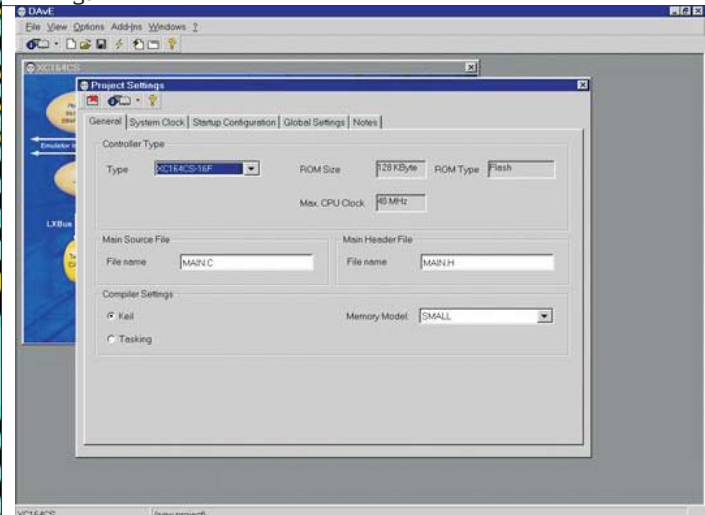
File;



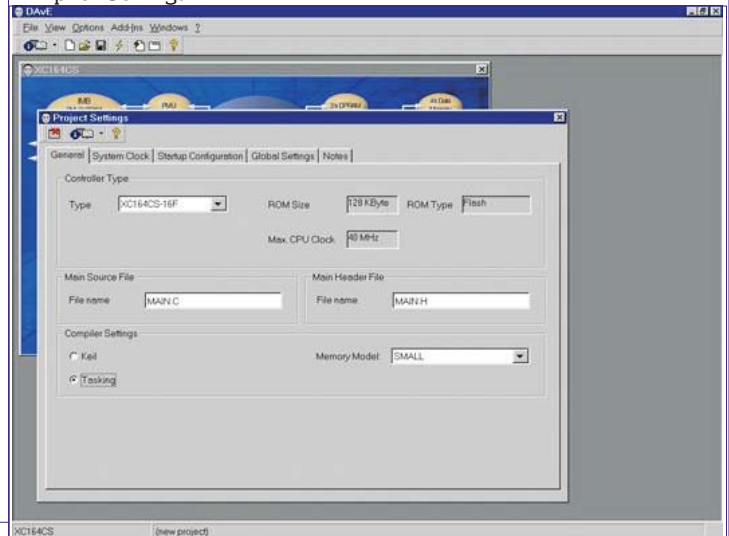
New;  
16-Bit Microcontrollers;  
XC164CS;  
Create:

#### 2. Step - Choose the Project Settings as you can see in the Screenshots

General: For the KEIL Compiler choose Keil in the Compiler Settings



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



System Clock: - do not change configuration, CPU Clock will be 20 Mhz

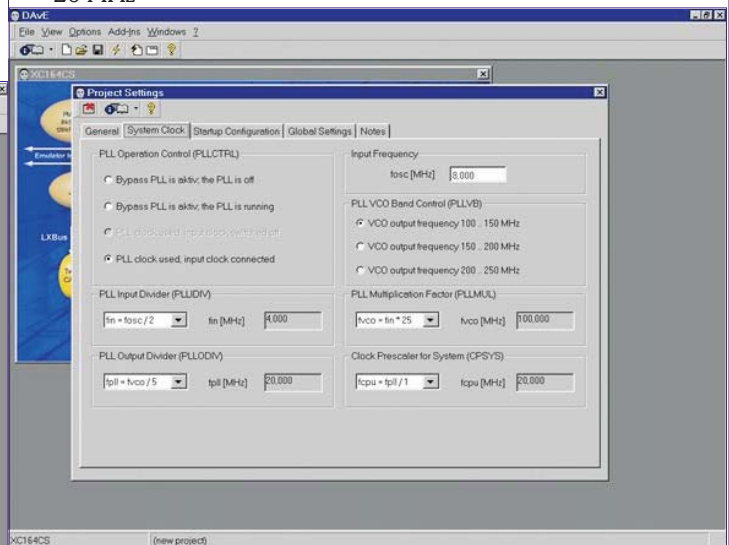
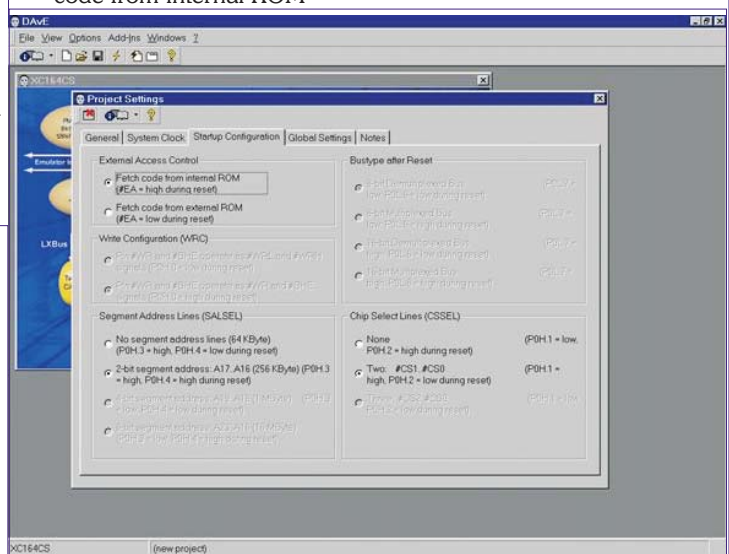


Bild 4

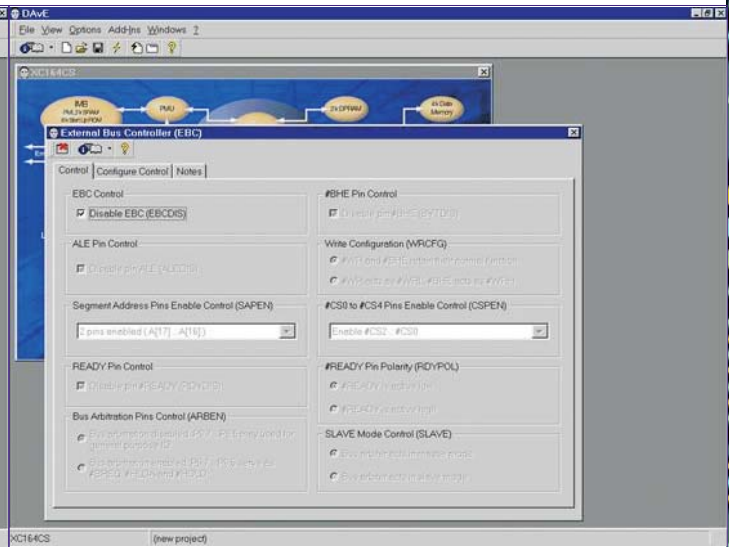
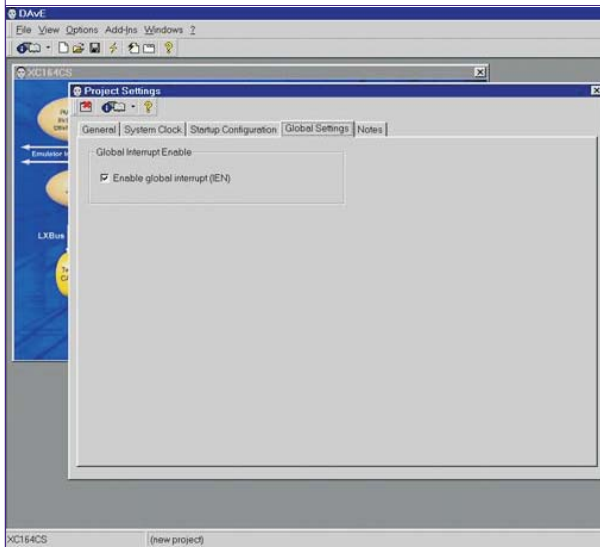
Startup Configuration: External Access Control: Select: Fetch code from internal ROM



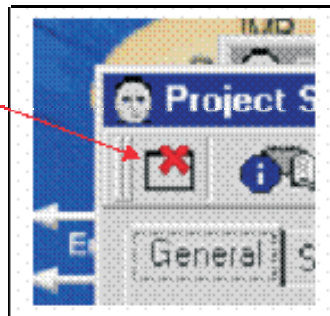


Global Settings: do not change configuration

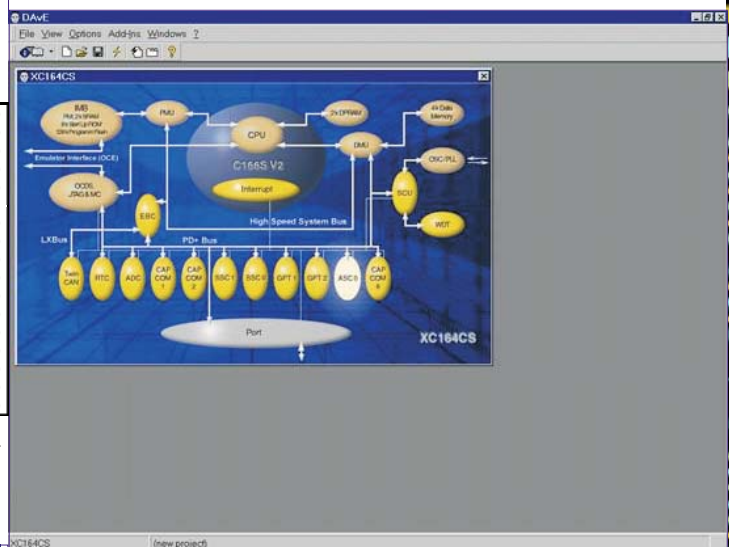
Exit this dialog now by clicking X in the close button



Notes: if you wish, you can insert here your comments  
Exit this dialog now by clicking X in the close button:

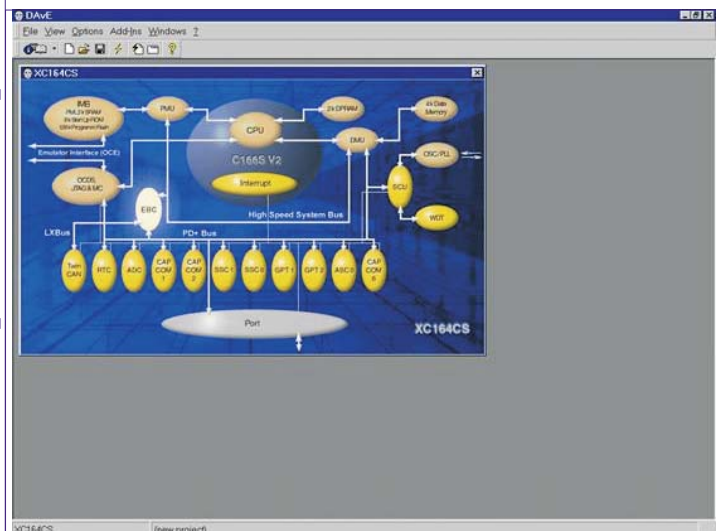


#### 4. Step - Configuration of the ASC0

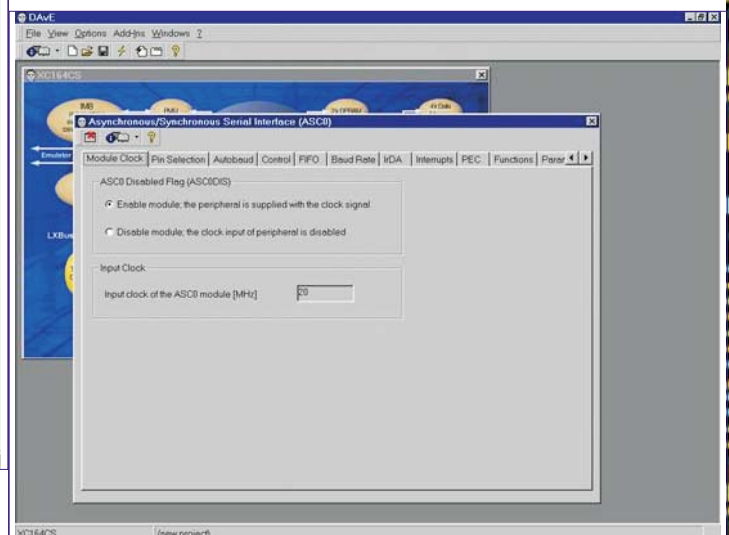


#### 3. Step: Configuration of the EBC: (configure as you can see in the screenshots)

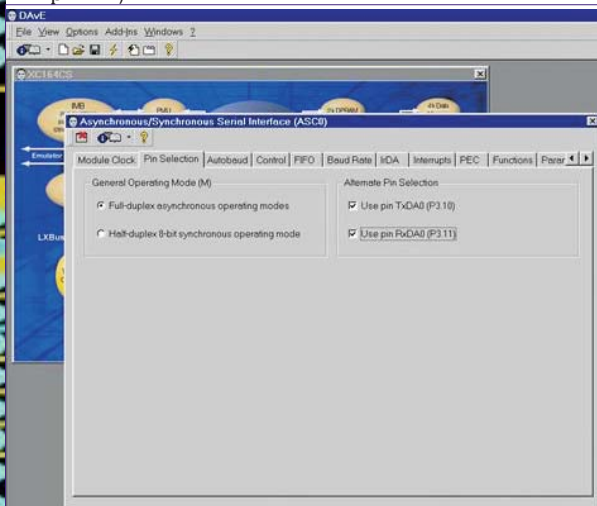
EBC: Control: click Disable EBC (EBDCDIS)



Module Clock: (do nothing)

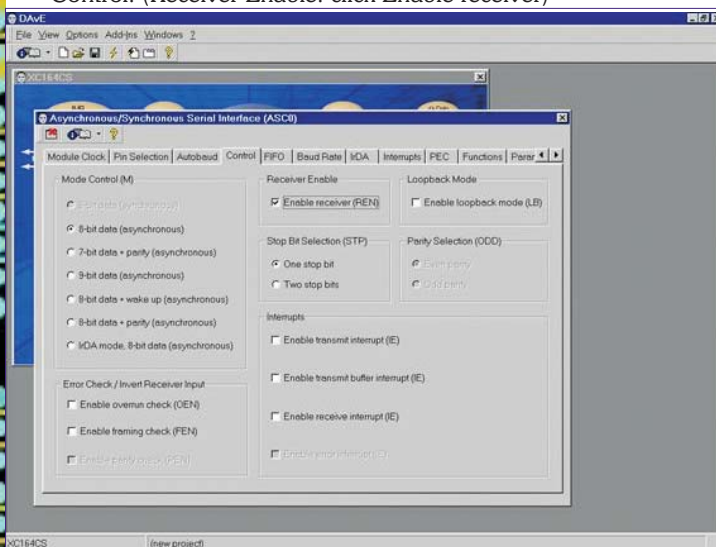


Pin Selection: (Alternate Pin Selection: click Use pin Tx, click Use pin Rx)



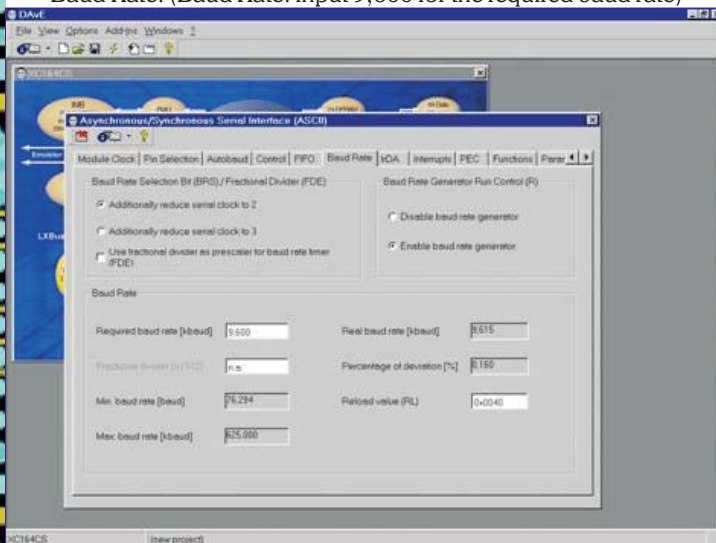
Autobaud: (do nothing)

Control: (Receiver Enable: click Enable receiver)



FIFO: (do nothing)

Baud Rate: (Baud Rate: input 9,600 for the required baud rate)

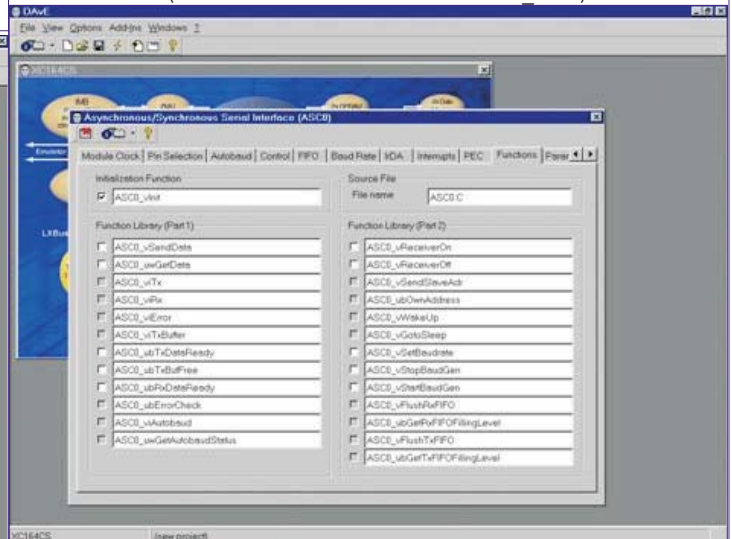


IrDA: (do nothing)

Interrupts: (do nothing)

PEC: (do nothing)

Functions: (Initialization Function: click ASC0\_vInit)

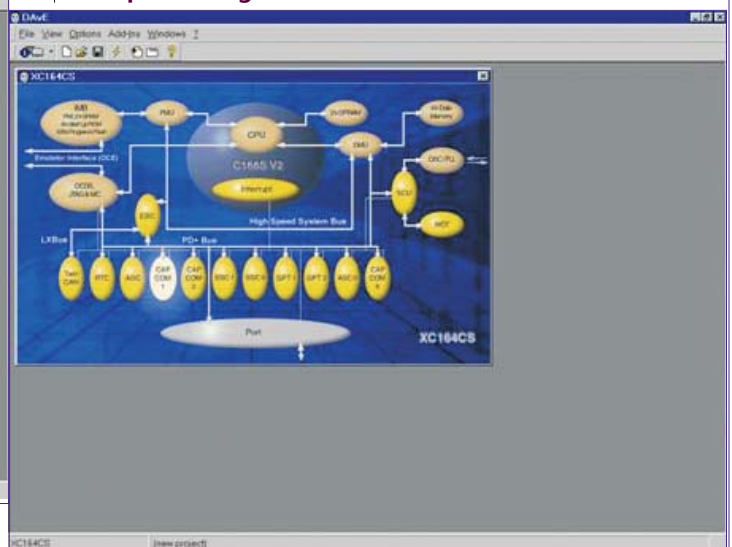


Parameters: (do nothing)

Notes: (if you wish, you can insert here your comments)

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

### 5. Step: - configure Timer T0 in CAPCOM 1



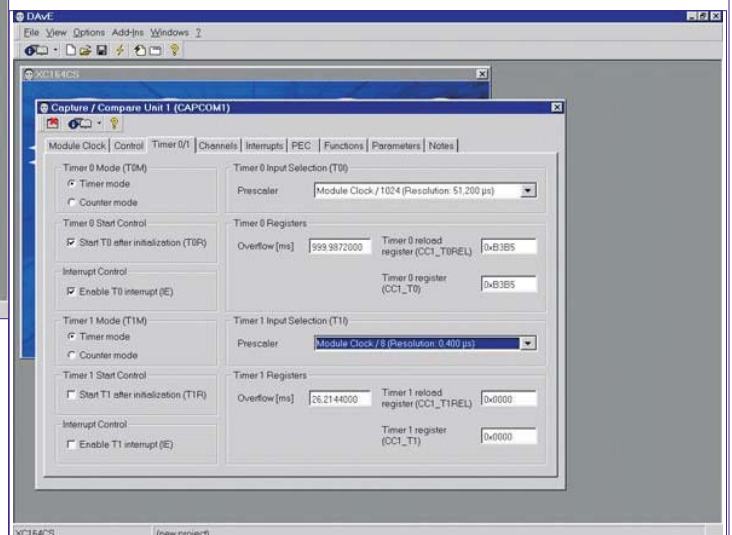
Module Clock: (do nothing)

Control: (do nothing)

Timer 0/1: (Timer 0 Start Control: click Start T0 after initialization (TOR))

Timer 0/1: (Interrupt Control click Enable T0 interrupt (IE))

Timer 0/1: (Timer 0 Input Selection (T0I): Prescaler: choose Module Clock/1024)

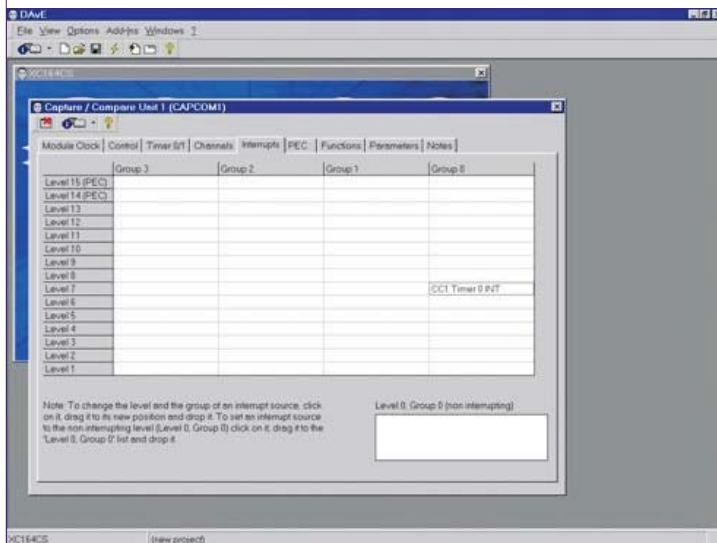




Timer 0/1: (Timer 0 Registers: Timer 0 reload register: input 0xB3B5 for 1 second)

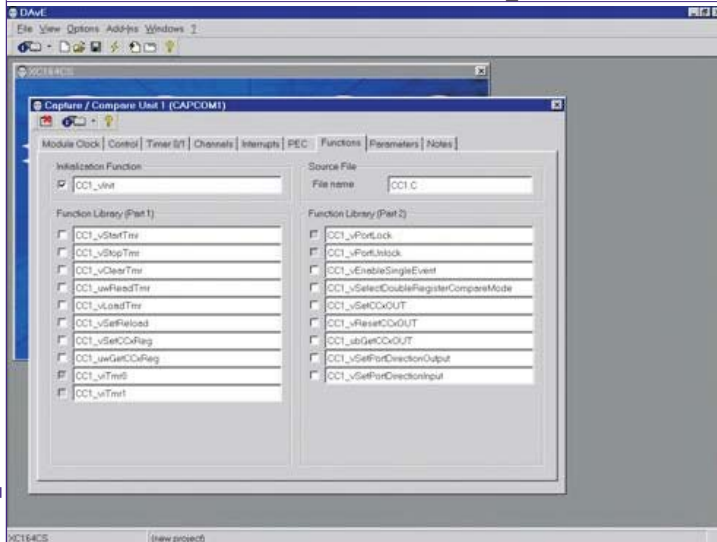
Channels: (do nothing)

Interrupts: (drag and drop the CC1 Timer 0 INT to Interrupt Level 7, Group 0)



PEC: (do nothing)

Functions: (Initialization Function: click CC1\_vInit)

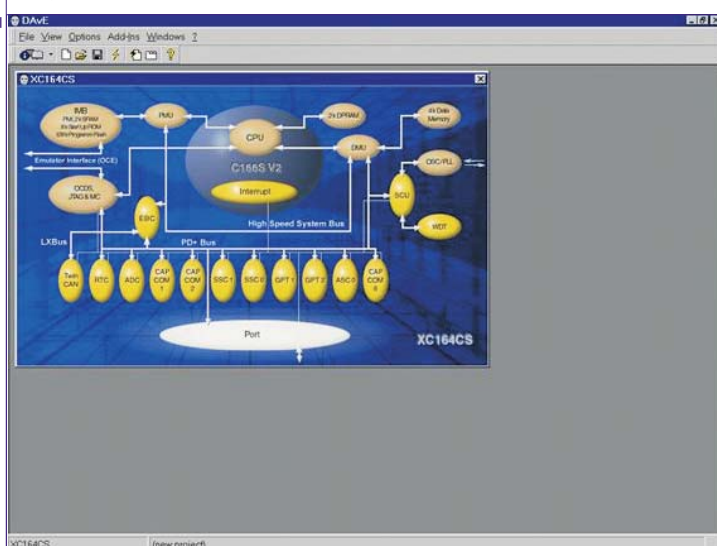


Parameters: (do nothing)

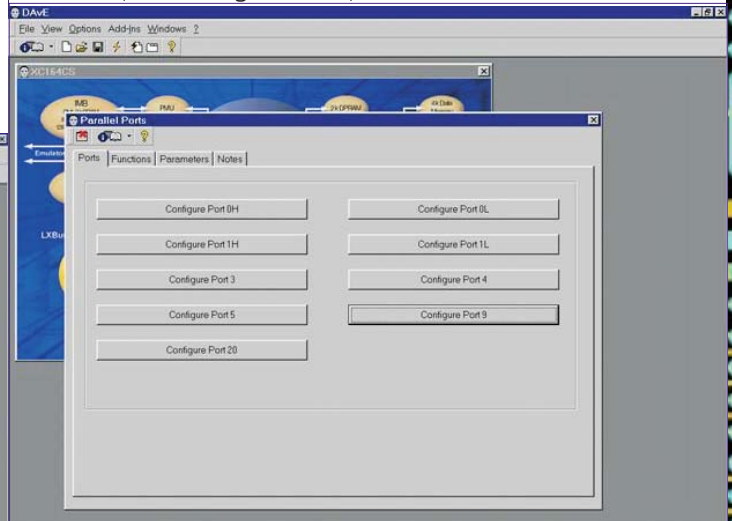
Notes: (if you wish, you can insert here your comments)

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

## 6. Step: - configure Port 9 Pin 4 to Output

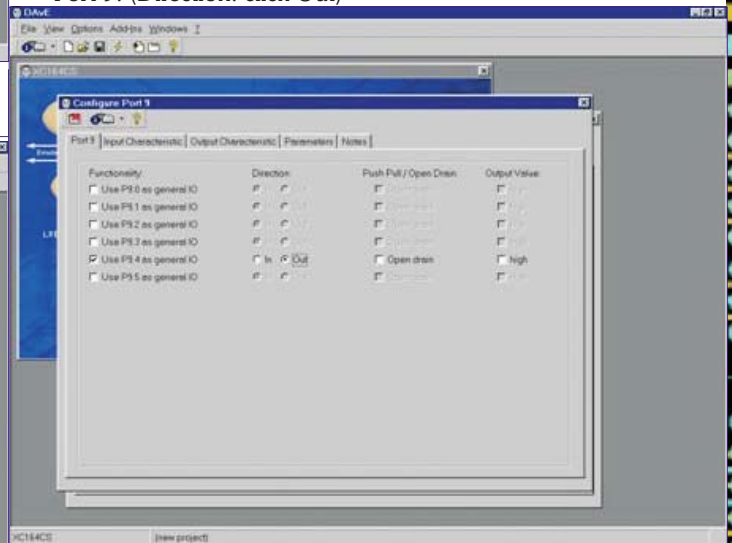


Ports: (click Configure Port 9)



Port 9: (Functionality: click Use P9.4 as general IO)

Port 9: (Direction: click Out)



Input Characteristic: (do nothing)

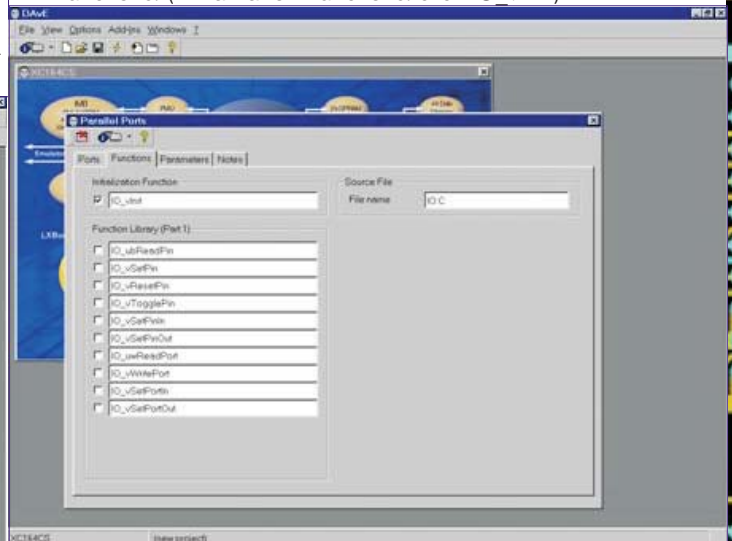
Output Characteristic: (do nothing)

Parameters: (do nothing)

Notes: (if you wish, you can insert here your comments)

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

Functions: (Initialization Functions: click IO\_vInit)



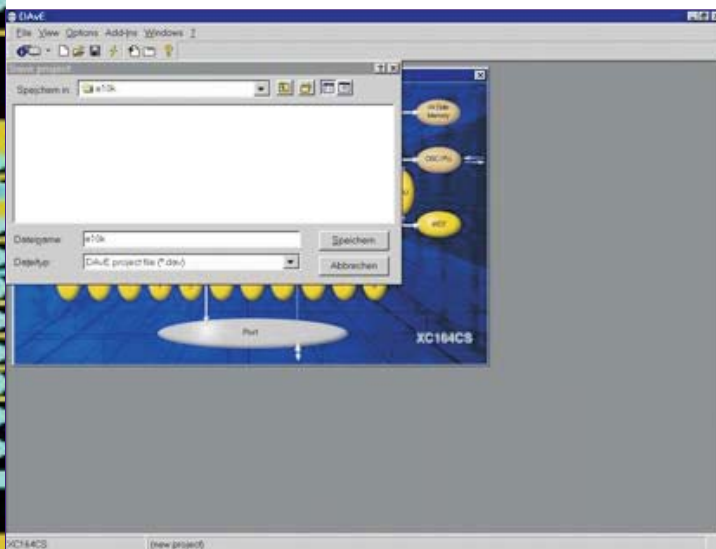
Parameters: (do nothing)

Notes: (if you wish, you can insert here your comments)

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

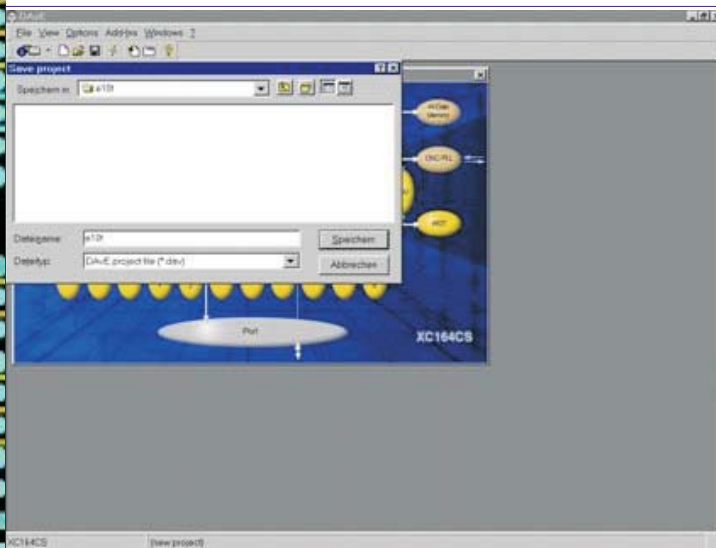
**7. Step – Save the project**

File Save

**For the KEIL Development Tools**

Location: C:\e10k

Filename: e10k

**For the TASKING Development Tools**

Location: C:\e10t

Filename: e10t

Save

**8. Step – Generate Code**

File

Generate Code or 

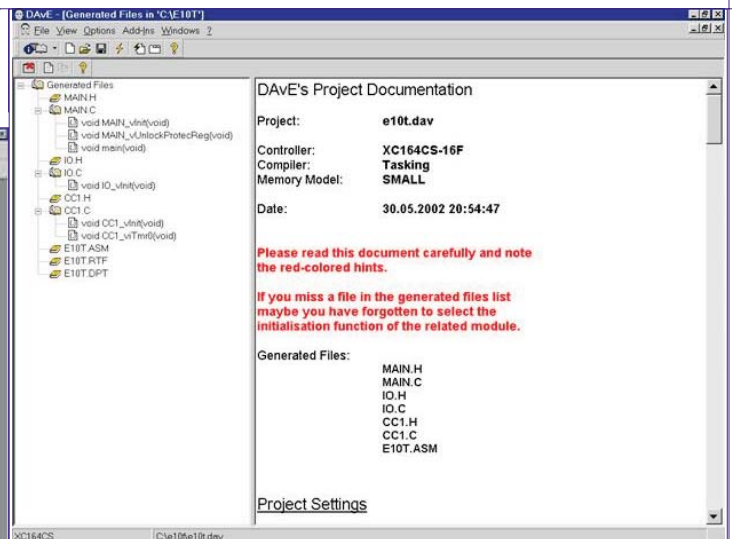
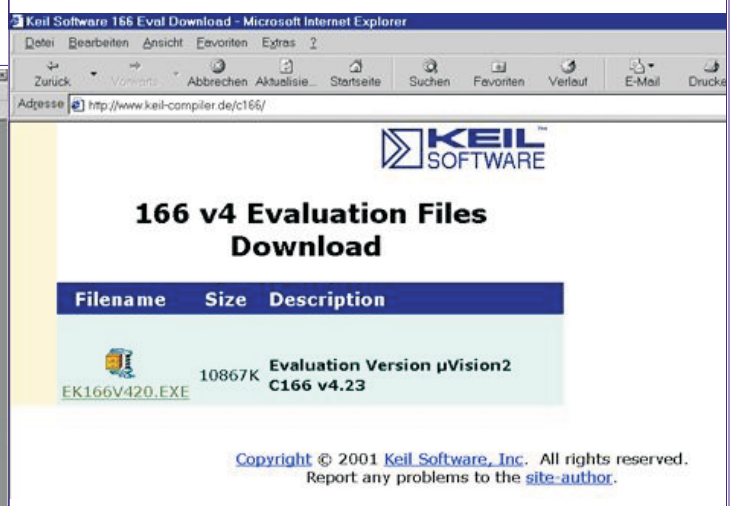
DAvE will show you all the files he has generated (File Viewer opens automatically).

File

Exit

Save changes?

click Yes

**IV.) Using of the KEIL - µVision 2 Development Tools****1. Step: – Install the Tool chain - here you can download the Keil Development Tools**<http://www.keil-compiler.de/c166/>

Execute ek166v423.exe and choose C:\KeilDemo for the installation path

**2. Step: Start Keil µVision and open the DAVE Project**

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

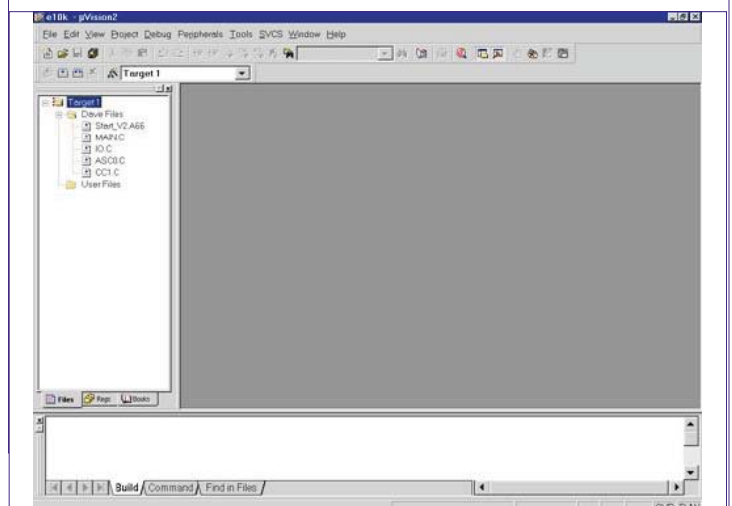
Project - Open Project

choose: C:\e10k

choose: File type: DAVE Project Files

choose: e10k.dpt

Open



**3. Step: include the DAVE Header Files**

mouse position: Files window: Dave Files: click right mouse button

Add Files to Group 'Dave Files'

input filename \*.h

mark Asc0.h Cc1.h Io.h Main.h

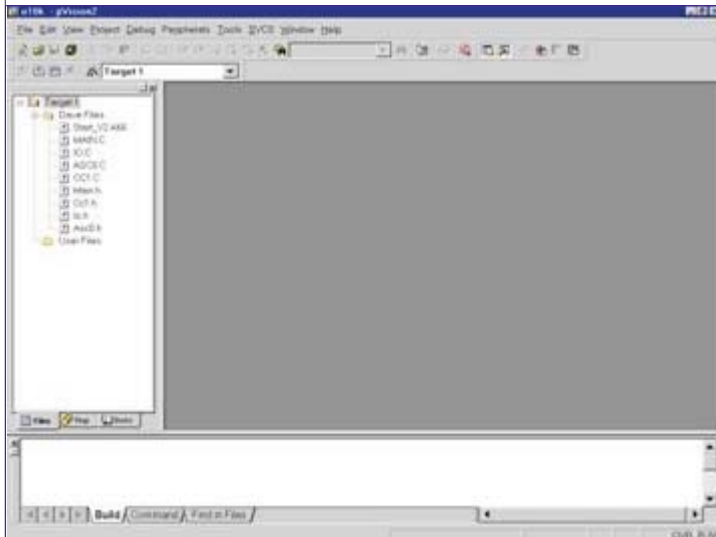
Add

do this 4 times:

Type: choose: Custom file

OK

Close

**4. Step: include the ConfPLL+Syscon3.c File**

mouse position: Files window: User Files: click right mouse button

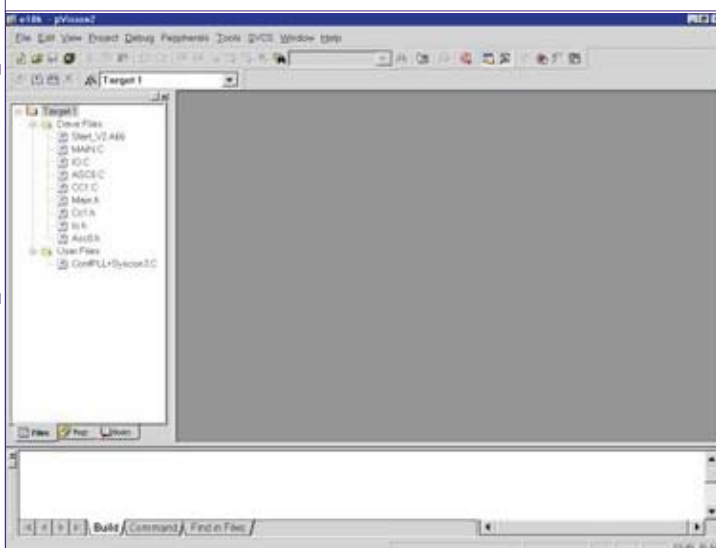
Add Files to Group 'User Files'

input filename \*.c

mark ConfPLL+Syscon3.c

Add

Close

**5. Step: Configure the Options for the Header Files:**

right mouse button click Main.h

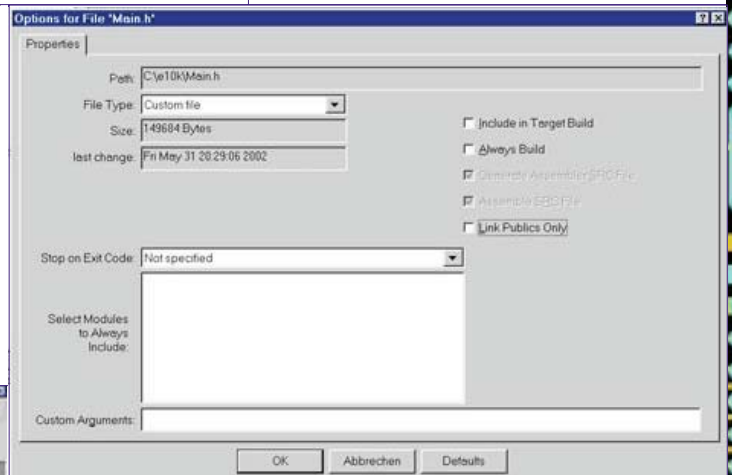
Options for File Main.h

click ? Include in Target Build

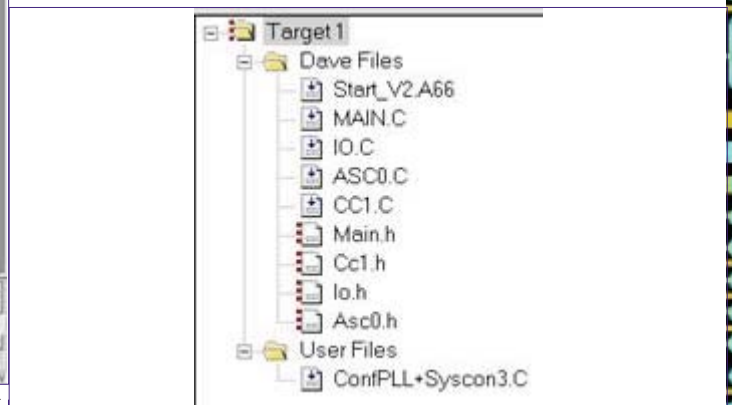
click ? Always Build

click ? Link Publics Only

OK



repeat this for the header files Cc1.h Io.h Asc0.h

**6. Step: Configure Compiler, Assembler, Linker, Locator, Hex-Converter and Build - Control**

right mouse button (Files window) click Target1

Options for Target 'Target1'

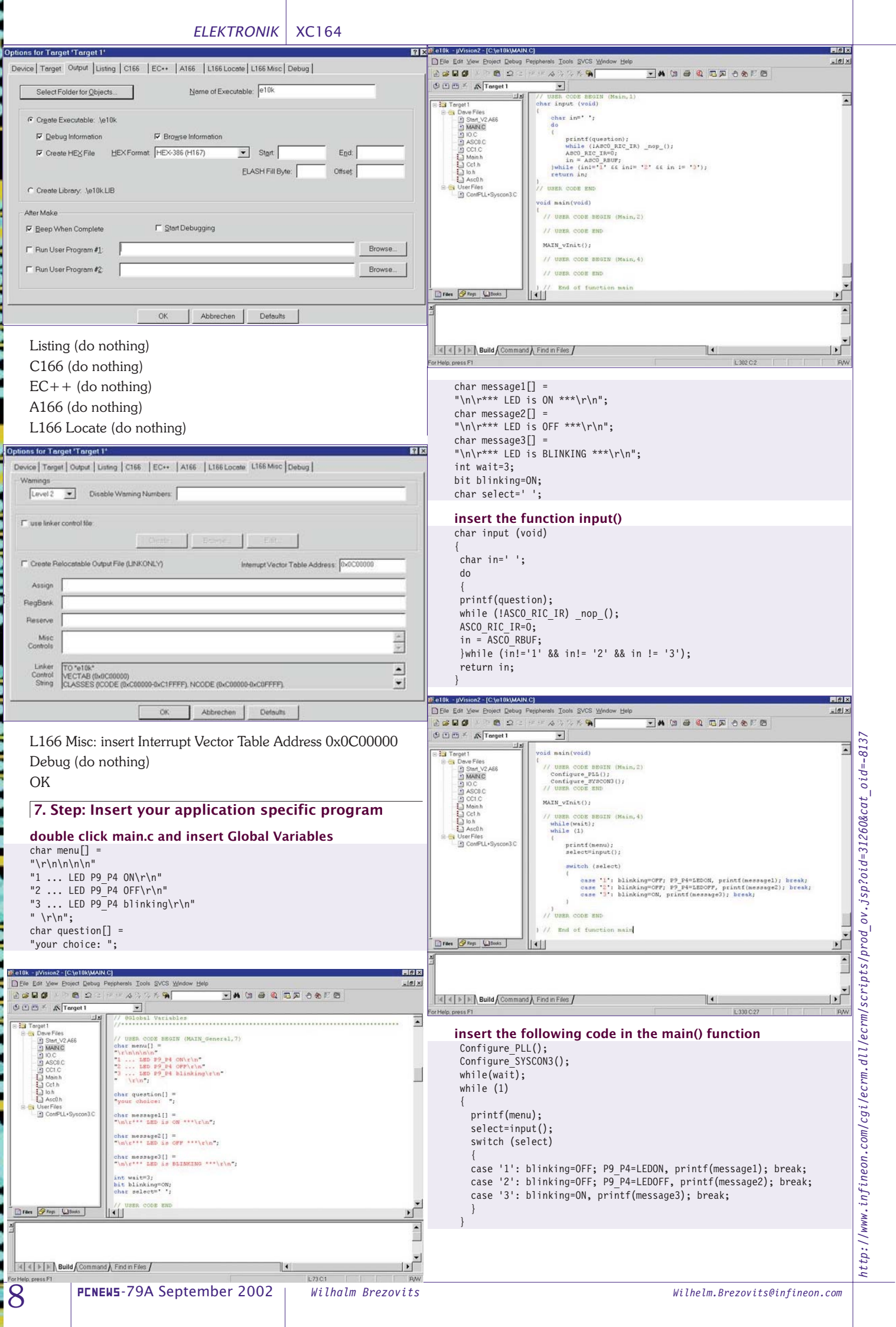
click ? Use On-chip XRAM

External Memory #1: select ROM, insert Start: 0x0C00000, insert Size: 0x20000



Output: click ? Create HEX File, choose HEX Format HEX-386(H167)





Options for Target 'Target 1'

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

Select Folder for Objects... Name of Executable: e10k

☒ Create Executable: \e10k

☒ Debug Information ☒ Browse Information

☒ Create HEX File HEX Format: HEX-386 (H167) Start: End:

FLASH Fill Byte: Offset:

☐ Create Library: \e10k.LIB

After Make


☒ Keep When Complete ☐ Start Debugging

☐ Run User Program #1: Browse...

☐ Run User Program #2: Browse...

OK Abbrechen Defaults

Listing (do nothing)  
C166 (do nothing)  
EC++ (do nothing)  
A166 (do nothing)  
L166 Locate (do nothing)



The screenshot shows the 'Options for Target "Target 1"' dialog box. The 'Device' tab is selected, showing a list of targets: C166, EC++, A166, L166 Locate, L166 Misc, and Debug. The 'Warnings' section has 'Level 2' selected and 'Disable Warning Numbers' checked. The 'use linker control file' checkbox is unchecked, with 'Create...', 'Browse...', and 'Edit...' buttons below it. The 'Create Relocatable Output File (LINKONLY)' checkbox is also unchecked, with an 'Interrupt Vector Table Address' field set to '0xC00000'. The 'Assign' section has empty fields for 'RegBank' and 'Reserved'. The 'Misc Controls' section has a 'Linker Control String' field containing 'TO "e10k" VECTAB (0xC00000) CLASSES (CODE (0xC00000-0xC1FFFF), NCODE (0xC00000-0xC0FFFF))'. At the bottom are 'OK', 'Abbrechen', and 'Defaults' buttons.

```
L166 Misc: insert Interrupt Vector Table Address 0x0C00000
Debug (do nothing)
OK
```

## 7. Step: Insert your application specific program

**double click main.c and insert Global Variables**

```
char menu[] =
"\r\n\r\n\r\n\r\n"
"1 ... LED P9_P4 ON\r\n"
"2 ... LED P9_P4 OFF\r\n"
"3 ... LED P9_P4 blinking\r\n"
" \r\n";
char question[] =
"your choice: ";
```

The screenshot shows the Keil uVision IDE interface. The top menu bar includes File, Edit, View, Project, Debug, Peripherals, Tools, RVDS, Window, and Help. Below the menu is a toolbar with various icons for file operations and debugging. The left pane displays the project structure for 'Target1', which includes a 'Devive Files' folder containing 'Start\_V2.A66', 'MAIN.C', and several C files (I/O.C, ASCII.C, OCI.C, Main.h, Ccl.h, Ioh, Asc0.h) under a 'User Files' folder. The main window shows the code for 'MAIN.C'. The code includes a comment '// Global Variables', a preprocessor directive '#include "MAIN.h"', and a function 'char menu()' that displays a menu with three options: 'LED P9\_P4 ON', 'LED P9\_P4 OFF', and 'LED P9\_P4 BLINKING'. It also includes a 'char question()' function for user input, a loop that processes the user's choice, and a 'main()' function that calls 'question()', 'menu()', and a 'wait3()' function. The code is written in C and uses standard formatting for comments and indentation.

```
// Global Variables
.....

// USER CODE BEGIN (MAIN_general,7)
char menu() =
{
    "\n\n\n\n\n"
    "1 ... LED P9_P4 ON\n\n"
    "2 ... LED P9_P4 OFF\n\n"
    "3 ... LED P9_P4 BLINKING\n\n"
    "    \n\n";

char question[] =
"your choice: ";

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

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

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

int wait3;
bit blinking=ON;
char select= ' ';

// USER CODE END
```

The screenshot shows the Keil uVision2 IDE interface. The title bar indicates the project is 'pVision2 - pVision2 [C:\e10k\MAIN.C]'. The menu bar includes File, Edit, View, Project, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations and debugging. The left-hand pane displays the 'Target1' project tree, showing a hierarchy of files: Start\_V2.A66, MAIN.C, I/O.C, ASC0.C, CCI.C, Main.h, Ccl.h, Ioh, Asc0.h, and a 'User Files' folder containing 'ConfPL\*Syscon3.C'. The main workspace displays the source code for 'MAIN.C'. The code includes a function 'char input(void)' that prints a question and reads a character from the user, and a 'main(void)' function that calls 'input()' and 'MAIN\_vInit()'. The code is annotated with 'USER CODE BEGIN' and 'USER CODE END' markers. The status bar at the bottom shows 'Build / Command' and 'Find in Files'.

```

// USER CODE BEGIN (Main,1)
char input (void)
{
    char in=' ';
    do
    {
        printf(question);
        while (!ASC0_RIC_IR) _nop();
        ASC0_RIC_IR=0;
        in = ASC0_RBUP;
        while (in=='1' && in=='2' && in != '3');
        return in;
    }
    // USER CODE END

void main(void)
{
    // USER CODE BEGIN (Main,2)

    // USER CODE END

    MAIN_vInit();

    // USER CODE BEGIN (Main,4)

    // USER CODE END

} // End of function main

```

```
char message1[] =
"\n\r*** LED is ON ***\r\n";
char message2[] =
"\n\r*** LED is OFF ***\r\n";
char message3[] =
"\n\r*** LED is BLINKING ***\r\n";
int wait=3;
bit blinking=ON;
char select=' ';
```

insert the function input()

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

The screenshot shows the Keil uVision2 IDE interface. The title bar indicates the project is 'e10k - uVision2 - [C:\e10k\MAIN.C]'. The menu bar includes File, Edit, View, Project, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar contains various icons for file operations and debugging. The 'Target 1' dropdown menu is set to 'Target 1'.

The left pane displays the project structure for 'Target 1':

- Target 1
  - Device Files
    - Start\_V2\_A66
    - MAIN.C
    - IO.C
    - ASCO.C
    - CCI.C
    - Mem.h
    - Ccl.h
    - Io.h
    - Ascl.h
  - User Files
    - ConfPLL+Syscon3.C

The main editor window shows the code for 'main.c':

```
void main(void)
{
    // USER CODE BEGIN (Main,2)
    Configure_PLL1();
    Configure_SYSCON3();
    // USER CODE END

    MAIN_vInit();

    // USER CODE BEGIN (Main,4)
    while(wait);
    while (1)
    {
        printf(menu);
        select=input();

        switch (select)
        {
            case '1': blinking=OFF; P5_P4=LEDON; printf(message1); break;
            case '2': blinking=OFF; P5_P4=LEDOFF; printf(message2); break;
            case '3': blinking=ON; printf(message3); break;
        }
    }
    // USER CODE END
} // End of function main
```

The bottom status bar shows 'Build' and 'Command' buttons, and the 'Find in Files' search bar.

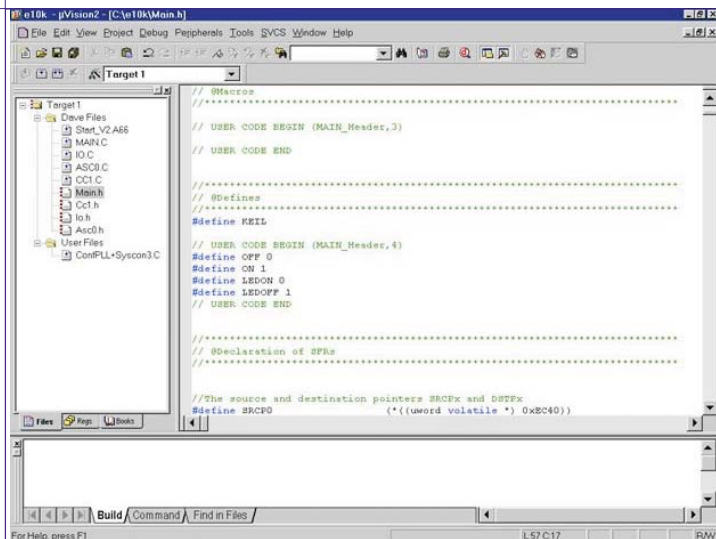
**insert the following code in the main() function**

```

Configure_PLL();
Configure_SYSCON3();
while(wait);
while (1)
{
    printf(menu);
    select=input();
    switch (select)
    {
        case '1': blinking=OFF; P9_P4=LEDON, printf(message1); break;
        case '2': blinking=OFF; P9_P4=LEDOFF, printf(message2); break;
        case '3': blinking=ON, printf(message3); break;
    }
}

```





double click main.h and insert the following Defines

```

#define OFF 0
#define ON 1
#define LEDON 0
#define LEDOFF 1

```

insert Global Variables

```

extern bit blinking;
extern int wait;

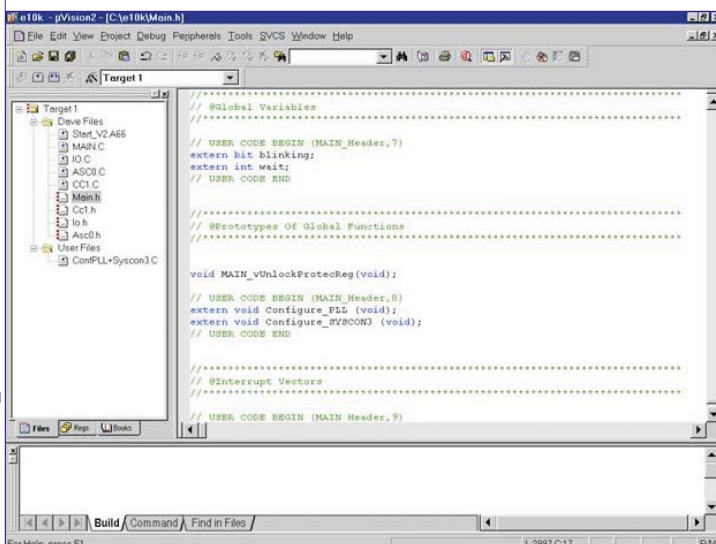
```

insert Prototypes

```

extern void Configure_PLL (void);
extern void Configure_SYSCON3 (void);

```

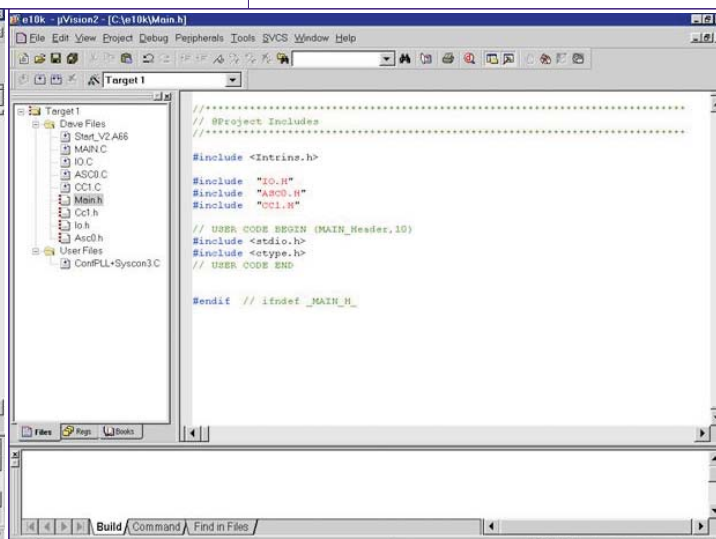


insert Includes (for printf() )

```

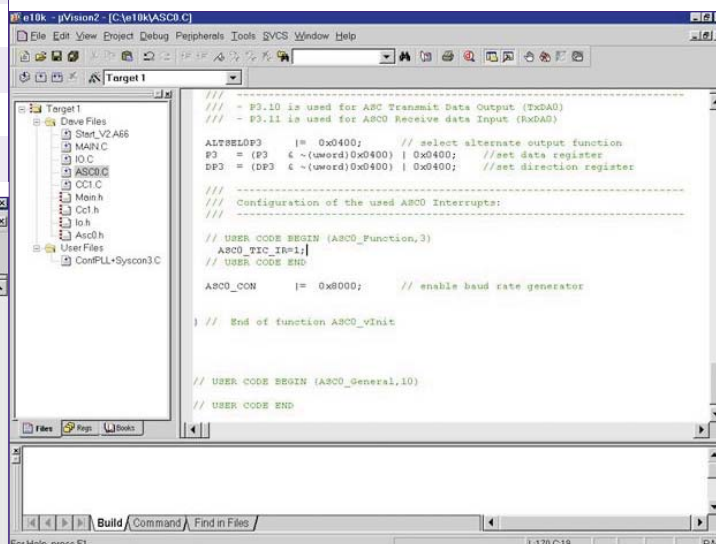
#include <stdio.h>
#include <ctype.h>

```



double click ASC0.C

insert in the ASC0\_vinit() function: (to start printf() )  
ASC0\_TIC\_IR=1;

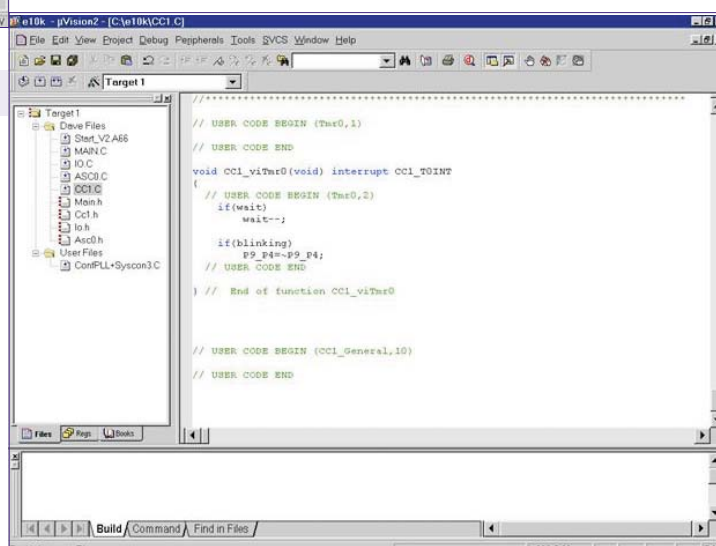


double click CCI.C  
insert code for T0 interrupt service routine

```

if(wait)
    wait--;
if(blinking)
    P9_P4=~P9_P4;

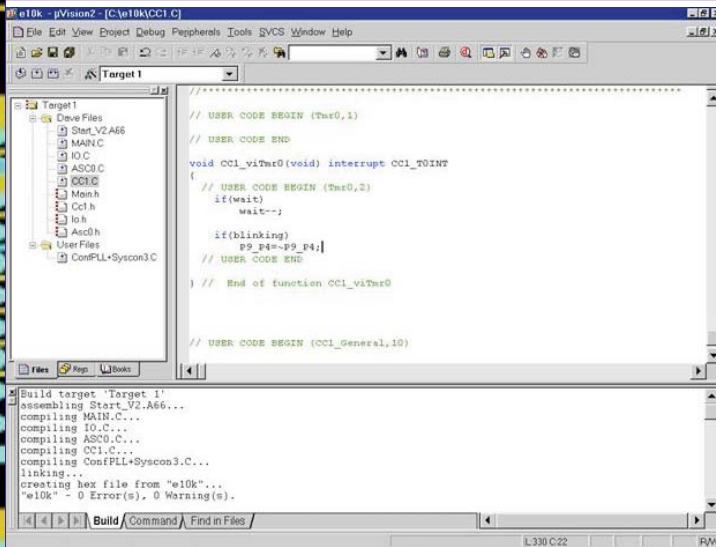
```



**8. Step: Generate your application program – generate the hex file for Memtool:**

Project

Rebuild all target files



Now you can close your project and µVision 2:

Project

Close Project

File

Exit

Now you can program the hex output file e10k.H86 with Mem-tool in the OnChipFlash (Program Memory) of the XC164CS microcontroller.

OK

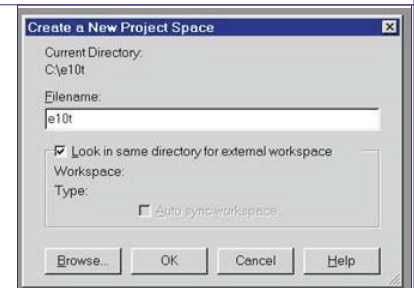
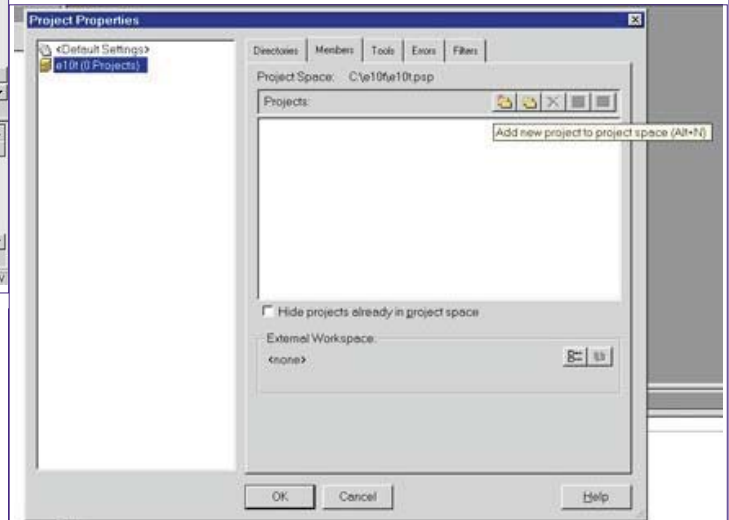
Project

Project Space

New

choose e10t

OK

**Add new project to project space**

Filename: insert e10t

**V.) Using of the TASKING - EDE Development Tools**

1. Step: – Install the Tool chain - here you can download the Tasking Development Tools:

[http://www.tasking.com/products/C166-ST10/demo\\_req.html](http://www.tasking.com/products/C166-ST10/demo_req.html)



Download dc166-75p-zip

Extract with WinZip

Execute setup.exe and choose C:\dc166 for the installation path

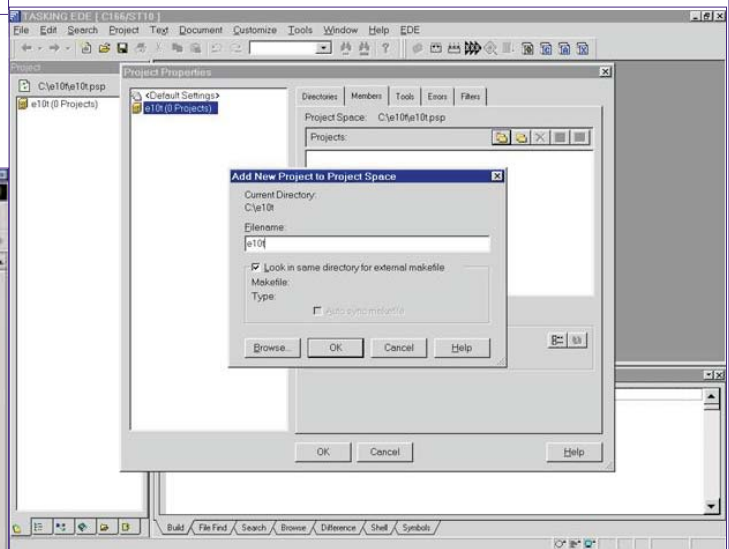
**2. Step: Start Tasking EDE, select directory and include the DAVE Files**

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

File

Change Directory

choose C:\e10t

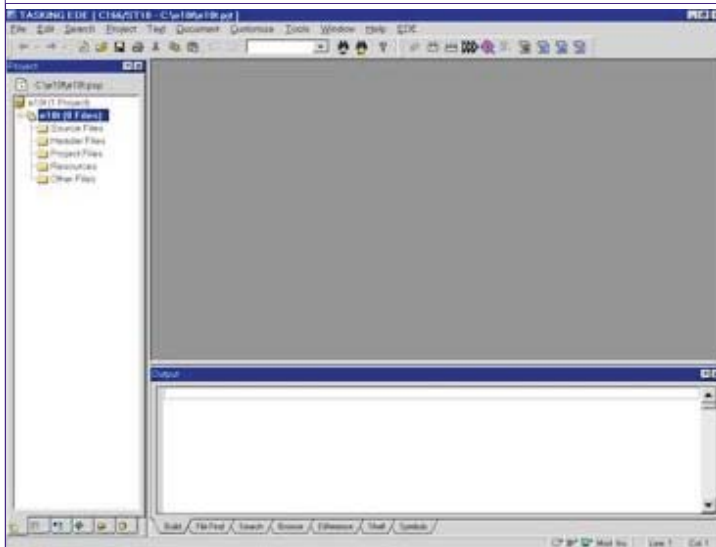


OK



OK

EDE - Project Options - CPU; CPU type: choose XC164CS OK



(File View window) right mouse button click e10t (0Files)

Add Existing Files

Browse

insert File-name (Dateiname) \*.asm

select Cstartx2.asm

Open (Öffnen)

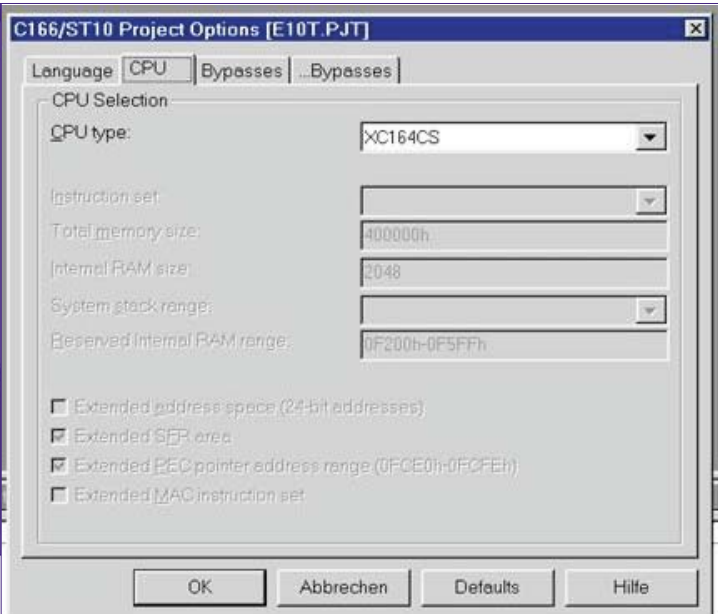
OK

repeat this with:

\*.c ( Asc0.c CC1.c Io.c Main.c Serio.c )

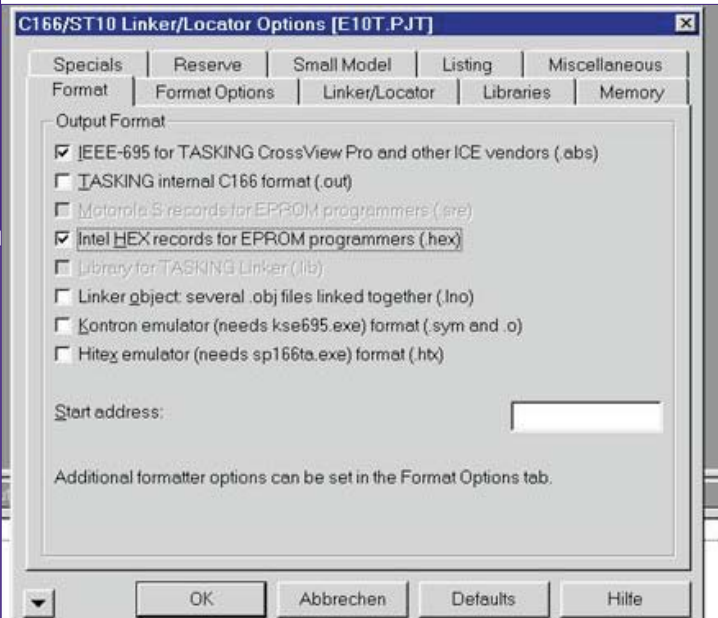
and

\*.h ( Asc0.h CC1.h Io.h Main.h Serio.h )



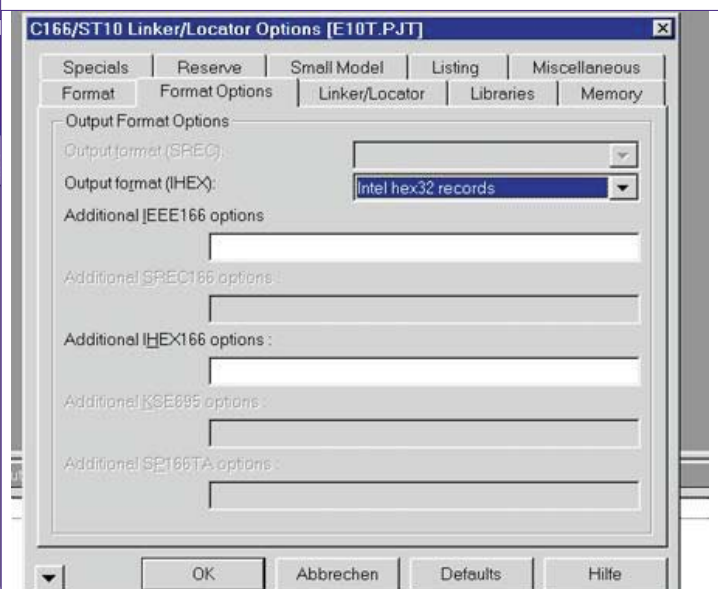
EDE - Linker/Locator Options - Format

✓ Intel HEX records for EPROM programmers (.hex)



EDE - Linker/Locator Options – Format Options

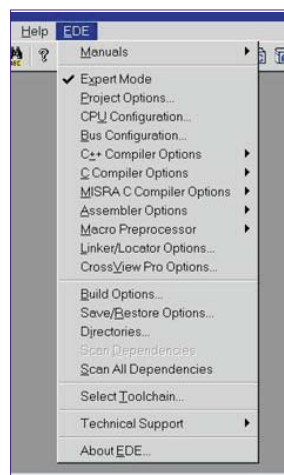
Output format (IHEX): choose Intel hex32 records



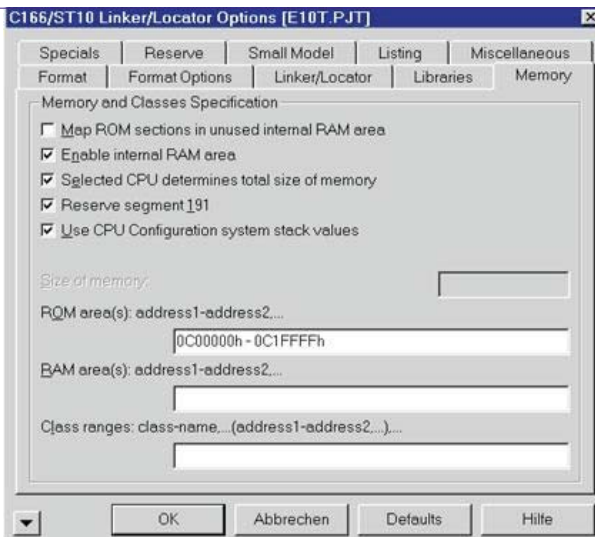
### 3. Step: Configure Compiler, Assembler, Linker, Locator, Hex-Converter and Build – Control

click EDE

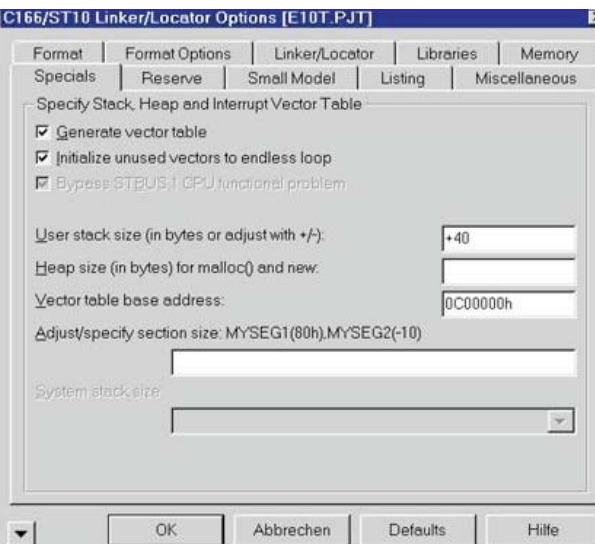
✓ Expert Mode



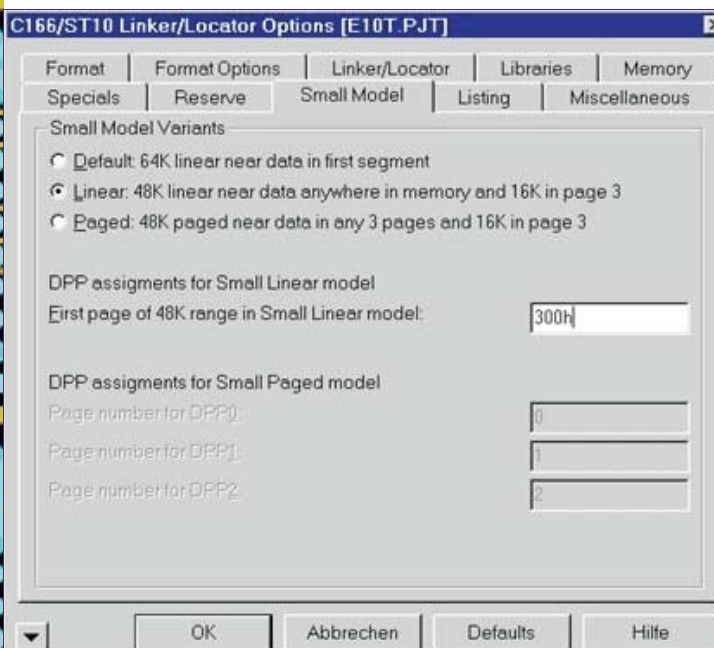
EDE - Linker/Locator Options - Memory  
Rom area(s) – input 0C00000h – 0C1FFFFh



EDE - Linker/Locator Options – Specials  
insert Vector table base address 0C00000h



EDE - Linker/Locator Options – Small Model  
click ? Linear: 48K  
insert First page of 48K range in Small Linear model: 300h

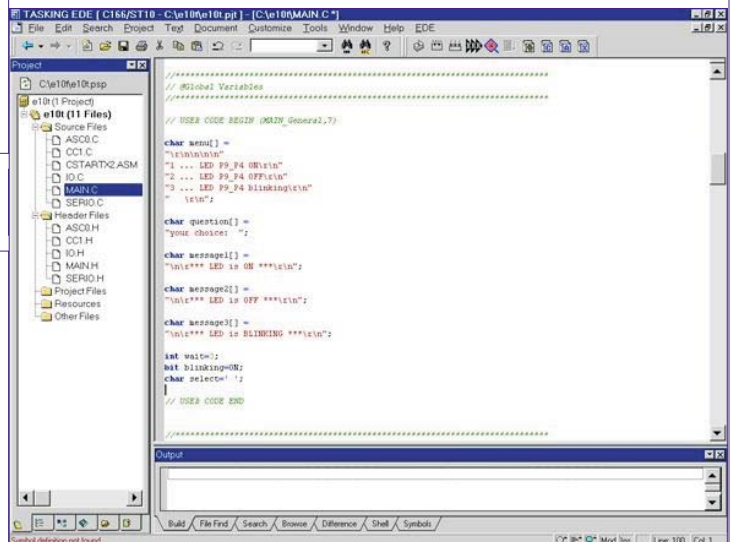


#### 4. Step: Insert your application specific program:

##### double click main.c: insert Global Variables

```
char menu[] =
"\r\n\r\n\r\n\r\n"
"1 ... LED P9_P4 ON\r\n\r\n"
"2 ... LED P9_P4 OFF\r\n\r\n"
"3 ... LED P9_P4 blinking\r\n\r\n"
"\r\n";

char question[] =
"your choice: ";
char message1[] =
"\n\r*** LED is ON ***\r\n";
char message2[] =
"\n\r*** LED is OFF ***\r\n";
char message3[] =
"\n\r*** LED is BLINKING ***\r\n";
int wait=3;
bit blinking=0N;
char select=' ';
```



##### main.c: insert User Code

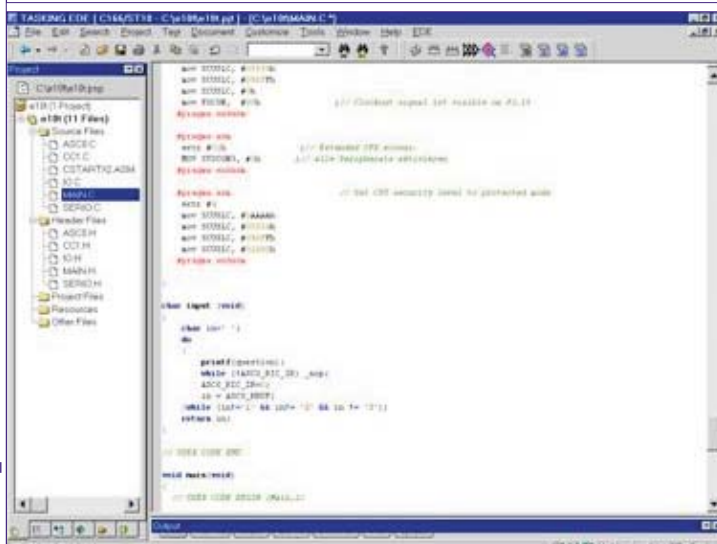
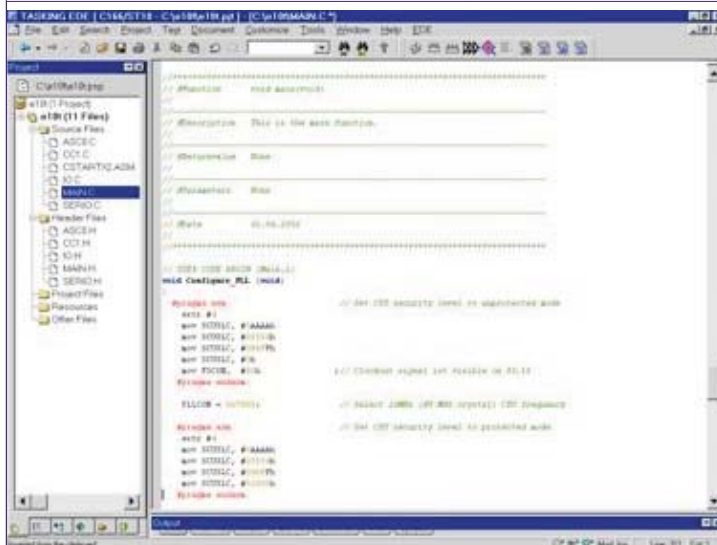
```
void Configure_PLL (void)
{
#pragma asm // Set CPU security level to unprotected mode
extr #4
mov SCUSLC, #0AAAAh
mov SCUSLC, #05554h
mov SCUSLC, #096FFh
mov SCUSLC, #0h
mov FOCON, #80h ;// Clockout signal ist visible on P3.15
#pragma endasm
PLLCON = 0x7889; // Select 20MHz (@8 MHz crystal) CPU frequency
#pragma asm // Set CPU security level to protected mode
extr #4
mov SCUSLC, #0AAAAh
mov SCUSLC, #05554h
mov SCUSLC, #096FFh
mov SCUSLC, #01800h
#pragma endasm
}

void Configure_SYSCON3 (void)
{
#pragma asm // Set CPU security level to unprotected mode
extr #4
mov SCUSLC, #0AAAAh
mov SCUSLC, #05554h
mov SCUSLC, #096FFh
mov SCUSLC, #0h
mov FOCON, #80h ;// Clockout signal ist visible on P3.15
#pragma endasm
#pragma asm
extr #01h ;// Extended SFR access.
MOV SYSCON3, #0h ;// alle Peripherals aktivieren
#pragma endasm
#pragma asm // Set CPU security level to protected mode
extr #4
mov SCUSLC, #0AAAAh
mov SCUSLC, #05554h
mov SCUSLC, #096FFh
mov SCUSLC, #01800h
#pragma endasm
}

char input (void)
{
char in=' ';
```

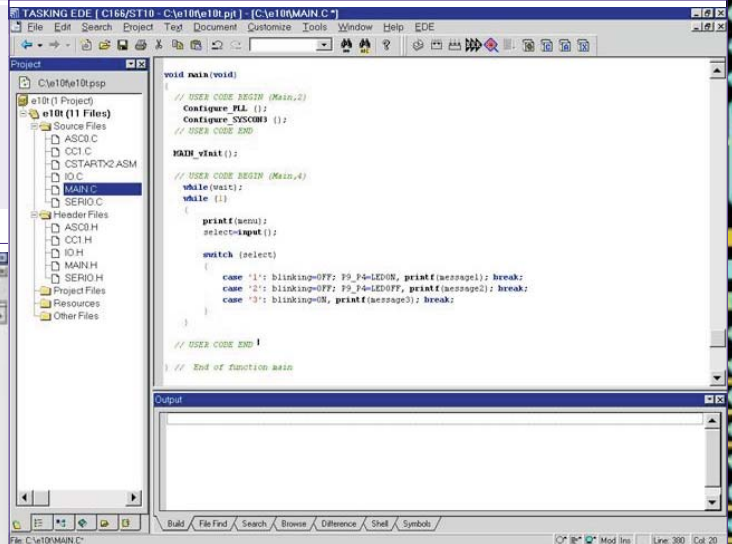


```
do
{
printf(question);
while (!ASCO_RIC_IR) _nop;
ASCO_RIC_IR=0;
in = ASCO_RBUF;
}while (in!='1' && in!='2' && in!='3');
return in;
}
```



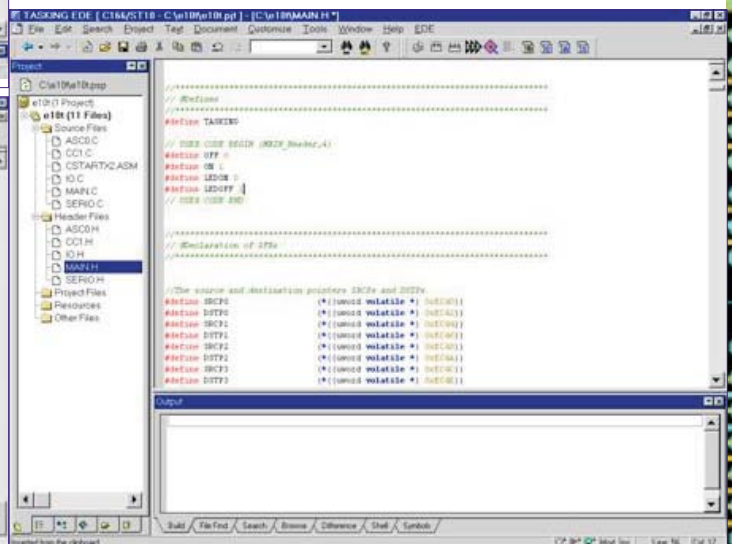
### insert the following code in the main function

```
Configure_PLL ();
Configure_SYSCON3 ();
while(wait);
while (1)
{
printf(menu);
select=input();
switch (select)
{
case '1': blinking=OFF; P9_P4=LEDON, printf(message1); break;
case '2': blinking=OFF; P9_P4=LEDOFF, printf(message2); break;
case '3': blinking=ON, printf(message3); break;
}
}
```



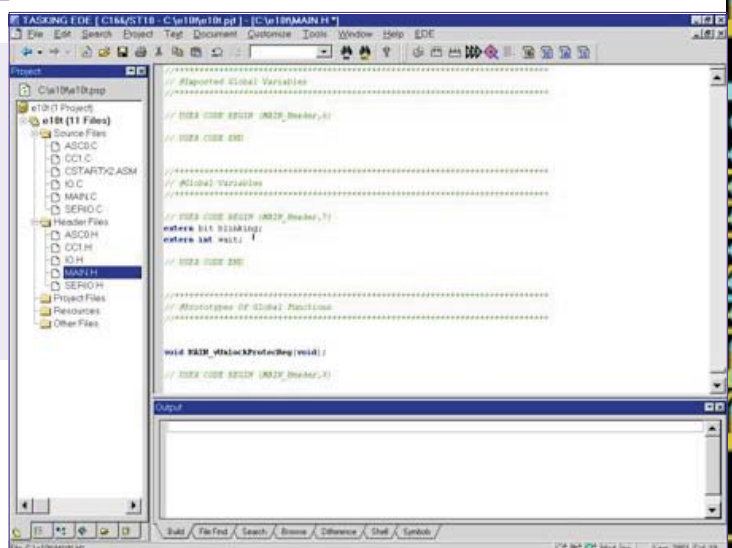
### double click Main.h and insert the following Defines:

```
#define OFF 0
#define ON 1
#define LEDON 0
#define LEDOFF 1
```



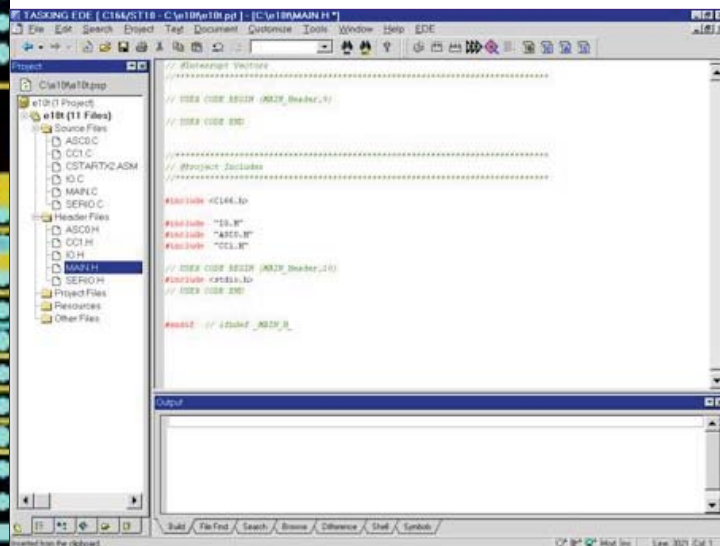
### insert Global Variables

```
extern bit blinking;
extern int wait;
```

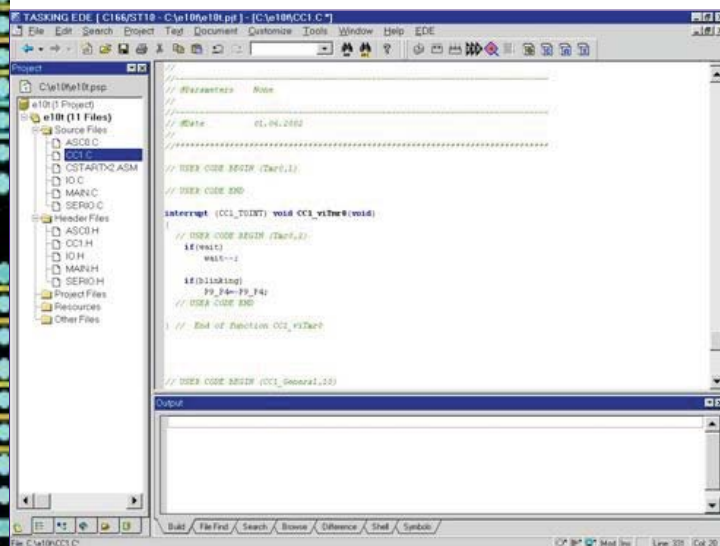


**insert Includes (for printf() )**

#include &lt;stdio.h&gt;

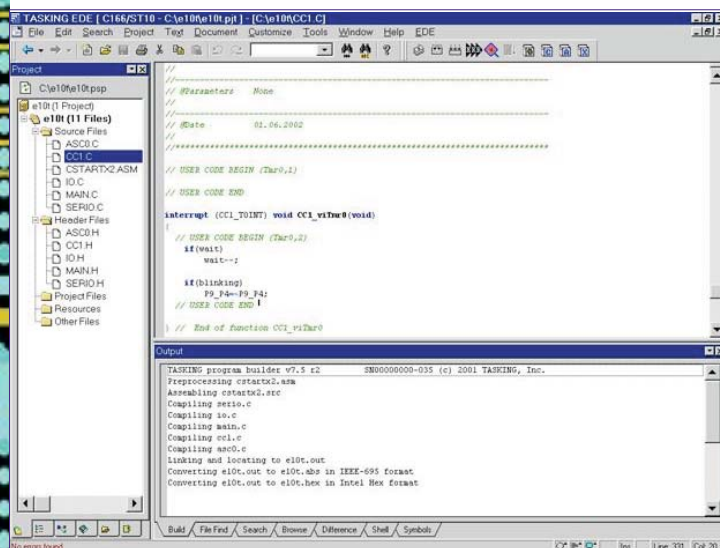
**double click CC1.C****insert code for T0 interrupt service routine**

```
if(wait)
wait--;
if(blinking)
P9_P4--P9_P4;
```

**5. Step: Generate your application program - generate the hex file for Memtool**

Project

Rebuild



Now you can close your project and Tasking EDE:

Project

Project Space

Close

File

Exit

Now you can program the hex output file e10t.HEX with Memtool in the OnChipFlash (Program Memory) of the XC164CS microcontroller.

**VI.) MEMTOOL Installation**

The following procedure will install/prepare Memtool for the programming of the XC16 family:

1. Execute file infineon\_memtool\_3\_2\_5.exe and choose Destination Location (default directory) C:\Programme\Memtool .
2. Start Memtool (use the Windows Start Menu)
3. Select under Menu Target\Change device XC161CJ-16FF/XC164CS-16FF STEP AA or device XC161CJ-16FF/XC164CS-16FF STEP AB.

Your system is now ready to program the OnChipFlash of the XC16 family.

**VII.) Using MEMTOOL**

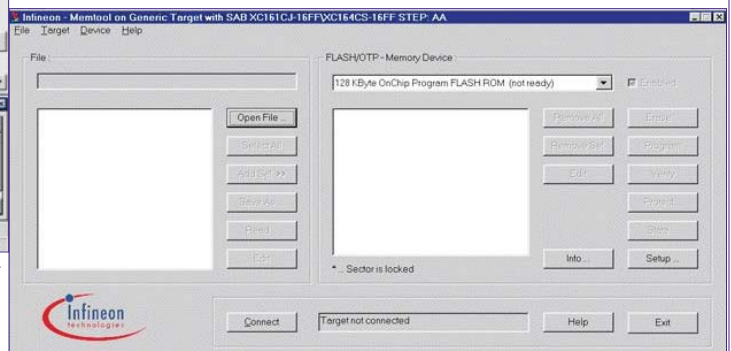
Program e10k.h86 (KEIL) or e10t.hex (TASKING) into the On-ChipFlash in the XC164:

Start MEMTOOL

Assign Power to your target board

Set Dip Switch S106 to Standard Boot (bootstrap loader)

Press Reset Key

**Erase the ONChipFlash**

Connect

Erase

Select All

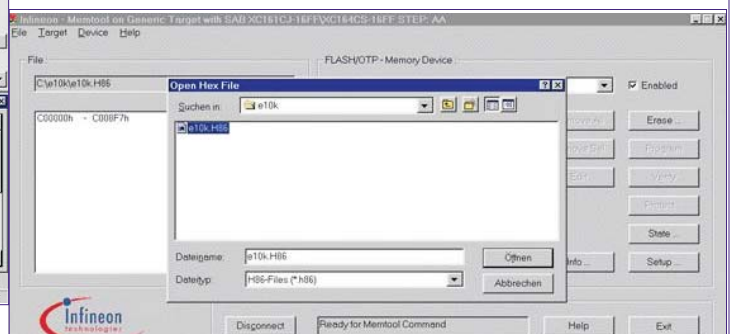
Start

Exit

**Load HEX File****for KEIL Compiler**

Open File

Dateityp (File type): H86-Files (\*.h86)





Select C:\e10k.h86

Öffnen (Open)

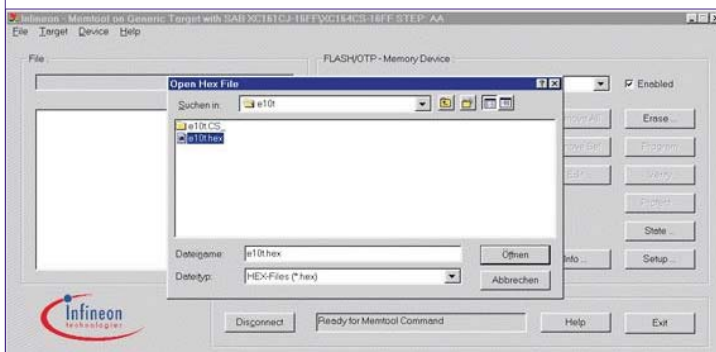
**for TASKING Compiler**

Open File

Dateityp (File type): HEX-Files (\*.hex)

Select C:\e10t.hex

Öffnen (Open)



**Program selected hex-file into the OnChipFlash:**

Select All

Add Sel >>

Program

Exit

**Exit MEMTOOL**

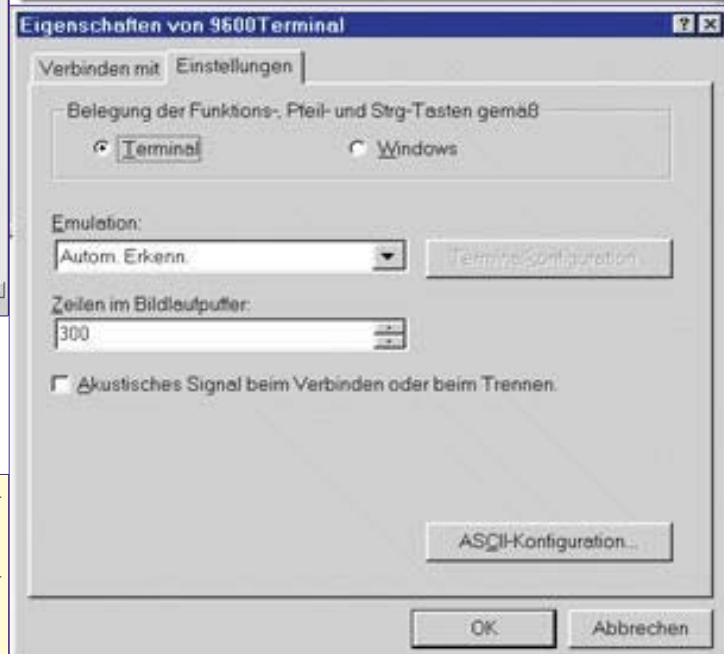
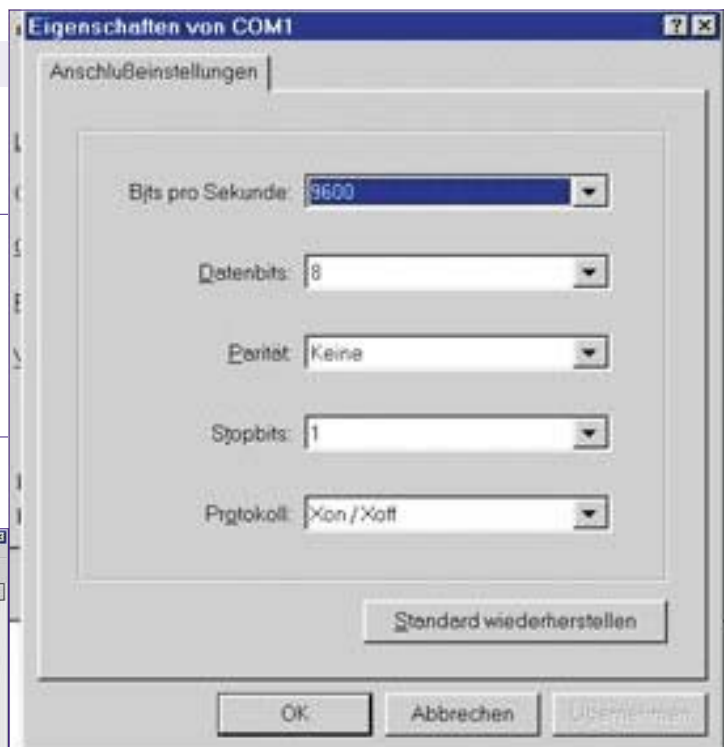
Disconnect

Exit

Set Dip Switch S106 to Standard Boot internal start

Press Reset Key

Your application is running!



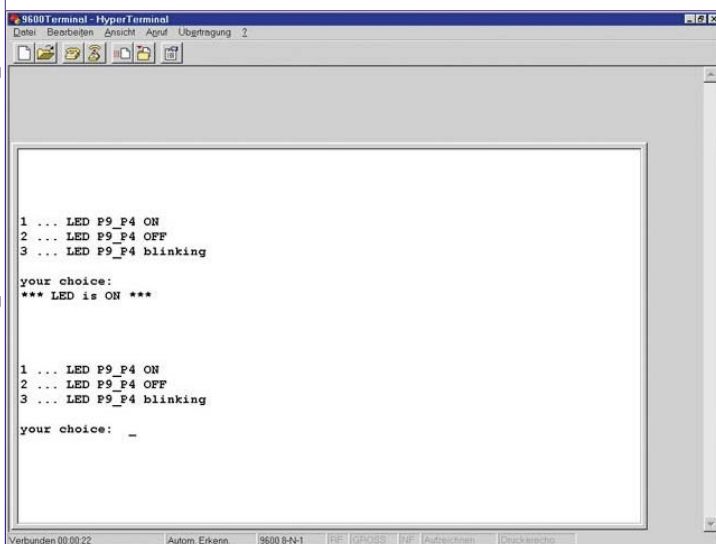
## VIII.) You need a Terminal Program; Settings

For your programming example you will need a terminal program.

You can use the Windows Utility Hyper Terminal. Please setup the configuration as shown in the following three pictures (9600 Baud, 8 bit Data, no Paritybit, 1 Stopbit, Xon/Xoff Protocol):

## IX.) Using a Terminal program

Screenshots of your running application:



**Starterkit**

<http://www.spacetools.com/tools4/space/688.htm>

**Starterkit-Bestellung**

<http://www.mtm.at/>