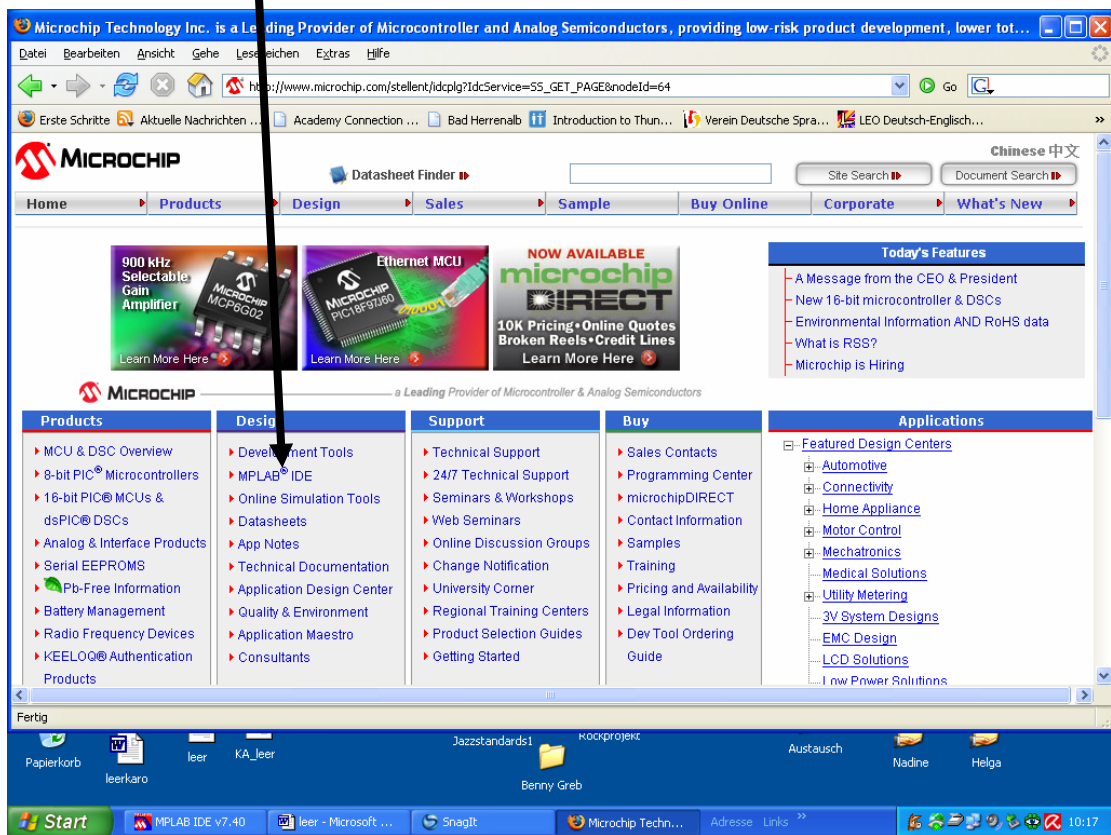
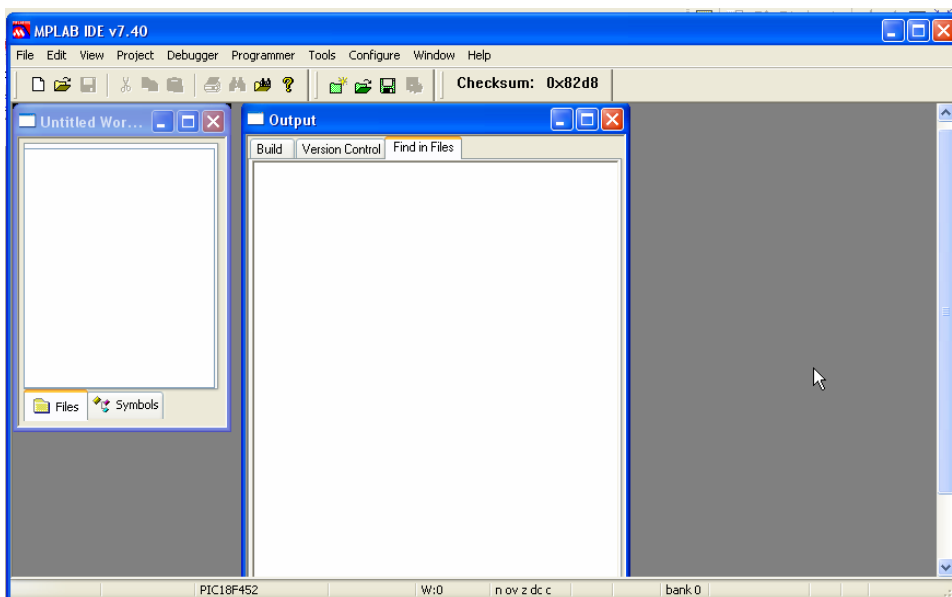


Installation von MPLab- 7.40 und C18 C Compiler

- 1.) → http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=64
MPLAB IDE (v7.40) downloaden IDE → Integrated Development Environment, die Integrierte Entwicklungsumgebung












- 2.) Nach Neustart kann man die IDE öffnen.



Scrollt man weiter nach unten, findet man die Downloads (pdf-Files)

Downloads

Title	Date Published	Size	D/L
MPLAB IDE User's Guide	8/4/2006 4:29:14 PM	3842 KB	
MPLAB® IDE Quick Chart	8/4/2006 4:30:48 PM	188 KB	
MPLAB IDE Quick Start Guide	8/4/2006 4:31:25 PM	985 KB	
MPASM/MPLINK User's Guide	8/18/2005 7:06:48 PM	2628 KB	
MPASM™/MPLINK™ PICmicro® Quick Chart	8/18/2005 7:06:48 PM	81 KB	
MPLAB C18 PIC18 Configuration Settings Addendum	5/23/2006 7:49:54 AM	3176 KB	
MPLAB® IDE User's Guide(Chinese)	8/18/2005 7:06:48 PM	4538 KB	
MPASM/MPLINK/MPLIB User's Guide (Chinese)	10/24/2005 10:20:57 PM	3579 KB	
Integrated Development Environment Tools Brochure	4/21/2006 9:14:43 AM	514 KB	

a) MPLAB IDE User's Guide

b) Evtl. C18 PIC 18 Configuration Settings. Wir fahren den PIC 18F2525.

MPLAB® IDE Quick Start Guide

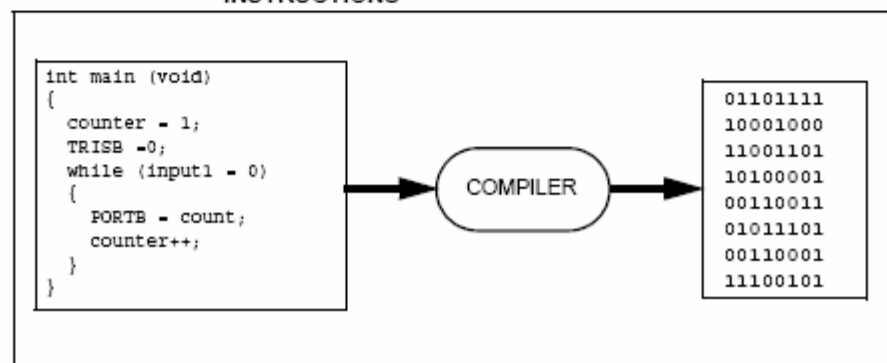
1.4 LANGUAGE TOOLS

Language tools are programs such as cross-assemblers and cross-compilers. Most people are familiar with language tools that run on a PC such as Visual Basic or C compilers. When using language tools for embedded systems, a "cross-assembler" or "cross-compiler" is used. These tools differ from typical compilers in that they run on a PC but produce code to run on another microprocessor, hence they "cross-compile" code for a microcontroller that uses an entirely different set of instructions from the PC.

The language tools also produce a debug file that MPLAB IDE uses to correlate the machine instructions and memory locations with the source code. This bit of integration allows the MPLAB IDE editor to set breakpoints, allows Watch windows to view variable contents, and lets you single step through the source code, watching the application execute.

Embedded system language tools also differ somewhat for compilers that run and execute on a PC because they must be very space conscious. The smaller the code produced, the better, because that allows the smallest possible memory for the target, which reduces cost. This means that techniques to optimize and enhance the code using machine specific knowledge are desirable. The size of programs for PCs typically extends into the megabytes for moderately complex programs. The size of simple embedded systems programs may be as small as a thousand bytes or less. A medium size embedded system might need 32K or 64K of code for relatively complex functions. Some embedded systems use megabytes of storage for large tables, user text messages or data logging.

FIGURE 1-7: A COMPILER CONVERTS SOURCE CODE INTO MACHINE INSTRUCTIONS



copyleft:munz

Ein **Cross-Compiler** bezeichnet einen Compiler, der auf einer Hostplattform H läuft und Objektdateien oder ein ausführbares Programm für eine andere Zielplattform T erzeugt.

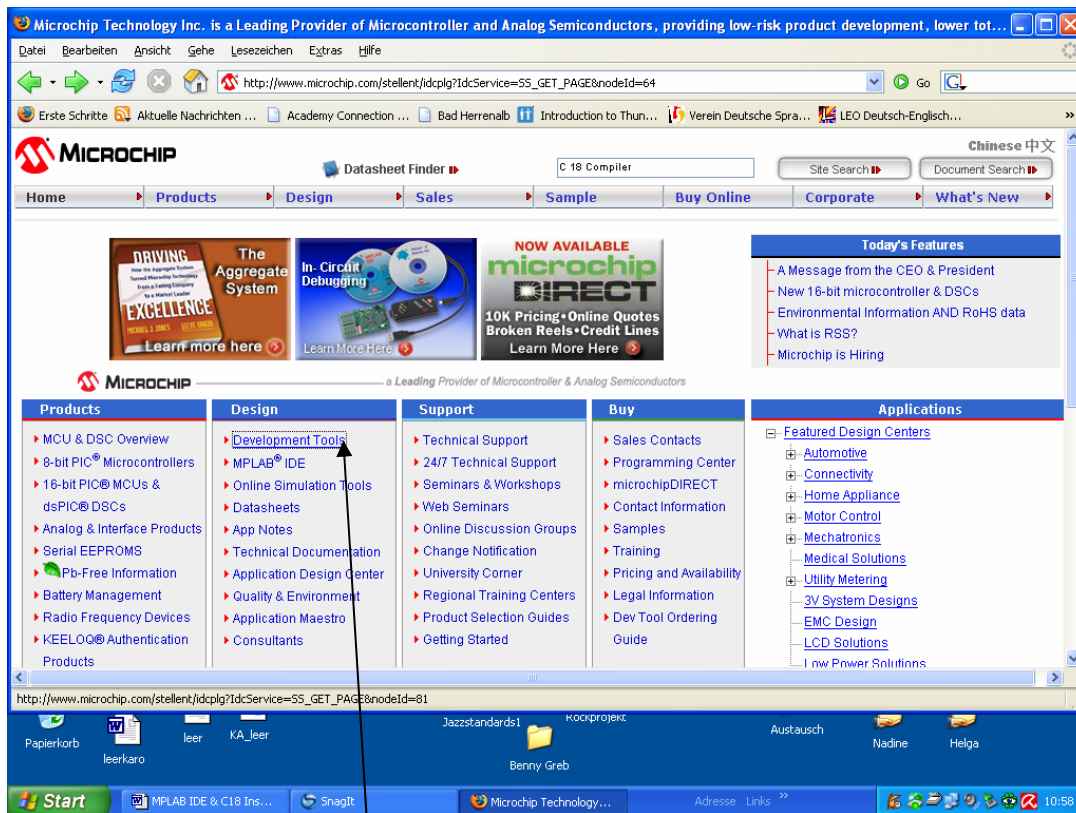
Cross-Compiler werden typischerweise benutzt, um ein Betriebssystem oder sich selbst auf eine neue Hardwareplattform zu portieren oder um Programme für eingebettete Systeme zu erstellen.

Populäre freie Compiler, die zum Teil auch für den Einsatz als Cross-Compiler geeignet sind, sind z.B. der GNU C Compiler GCC und der FreePascal-Compiler FPC.

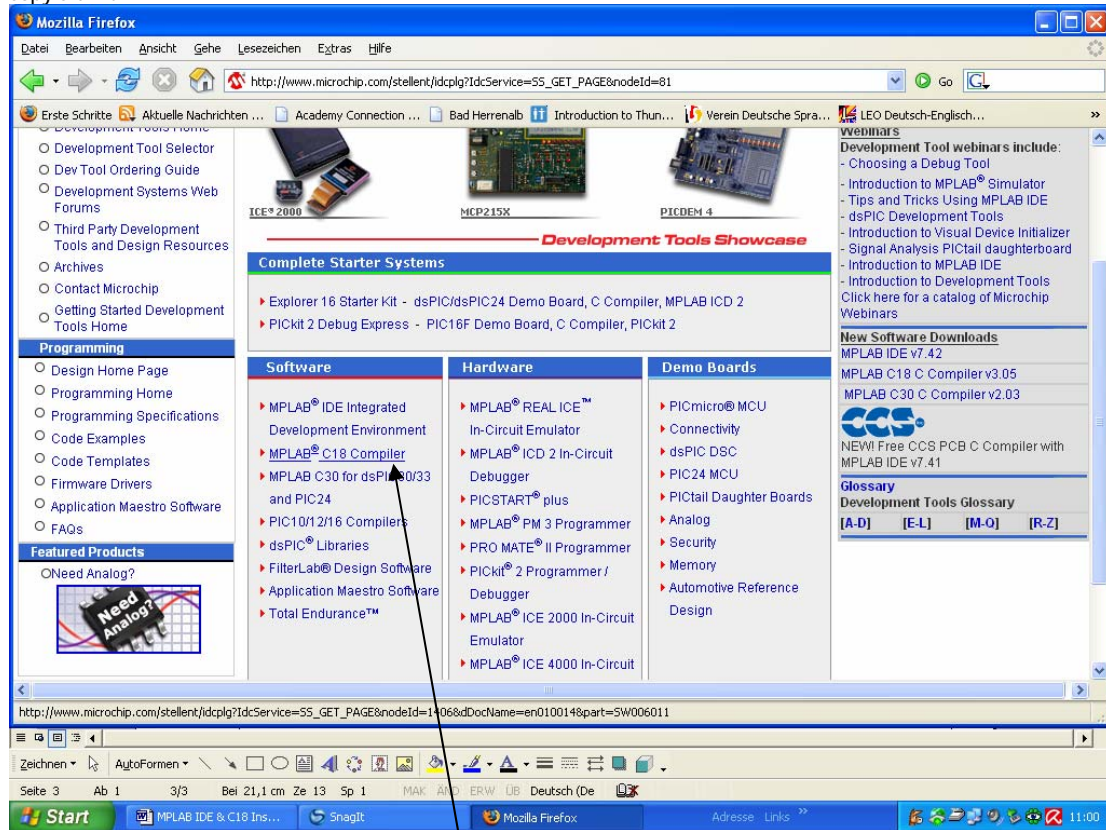
Für einige 8-Bit Zielsysteme (Intel 8051, Zilog Z80, Dallas 80C390, Motorola HC08, **Microchip PIC**) eignet sich der ebenfalls freie **Small Device C Compiler SDCC**.

<http://sdcc.sourceforge.net/> → http://sourceforge.net/project/showfiles.php?group_id=599

Wir bleiben aber bei Microchip und holen uns den C18 Compiler.



Über Development Tools (Entwicklungswerkzeuge) kommt man zum C18 Compiler Download.



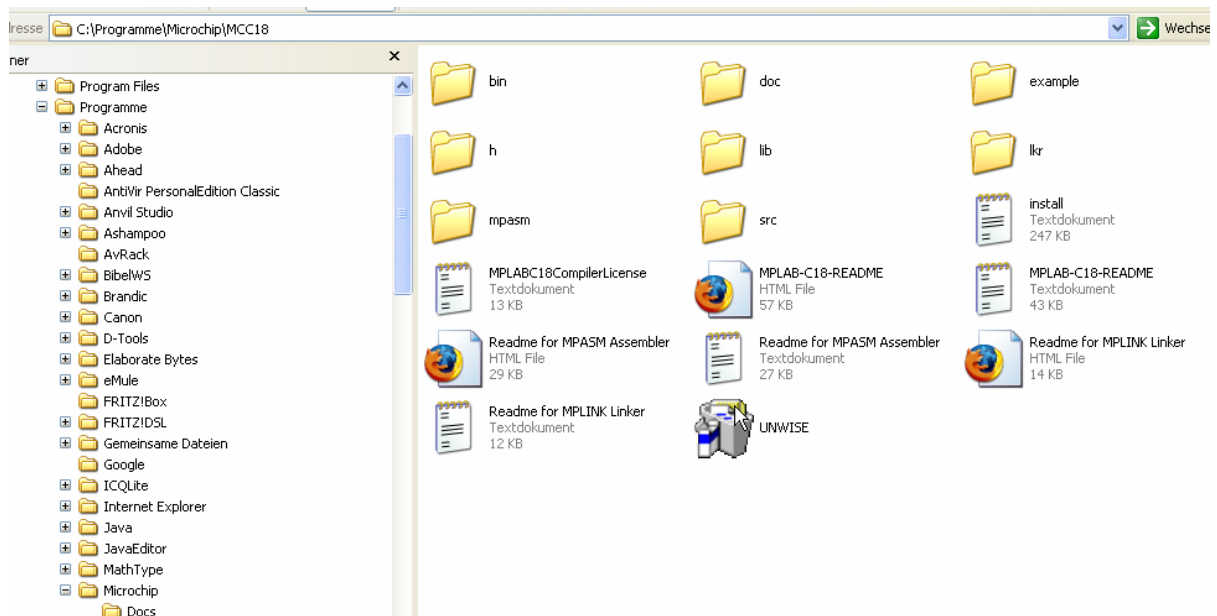
Downloads

Sign in required to download content.

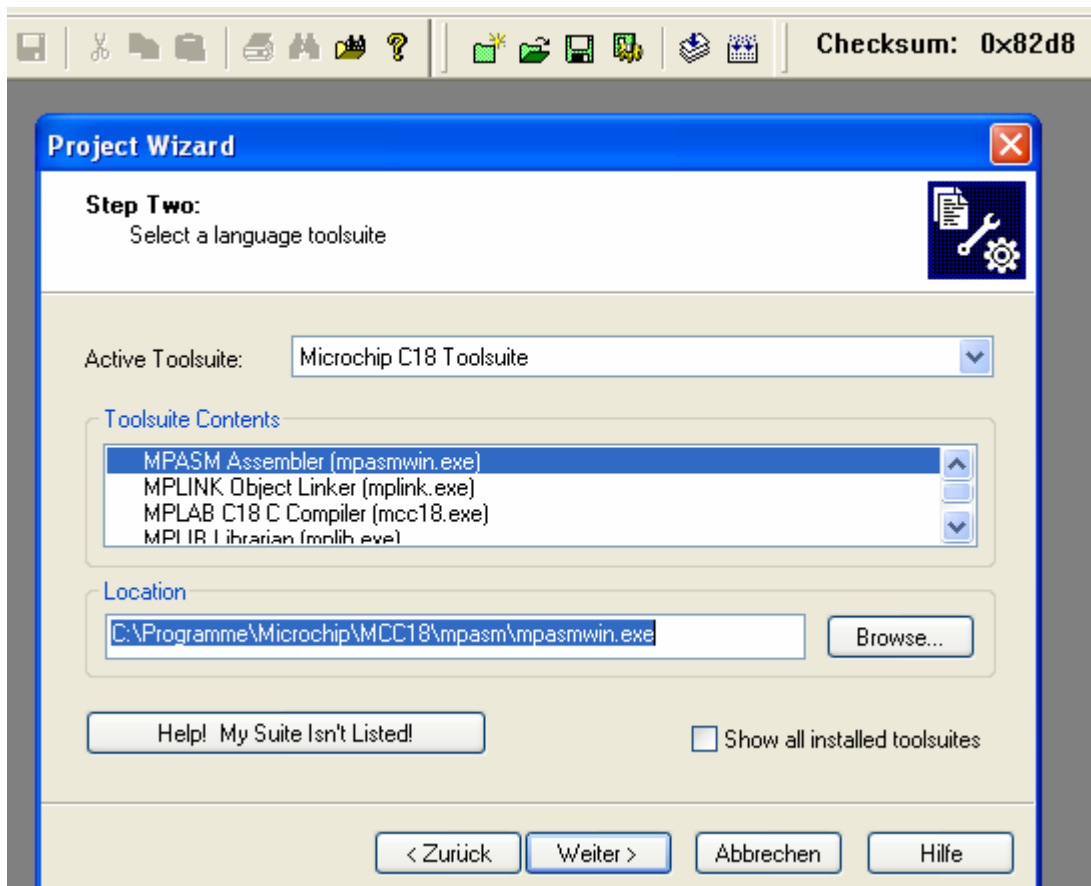
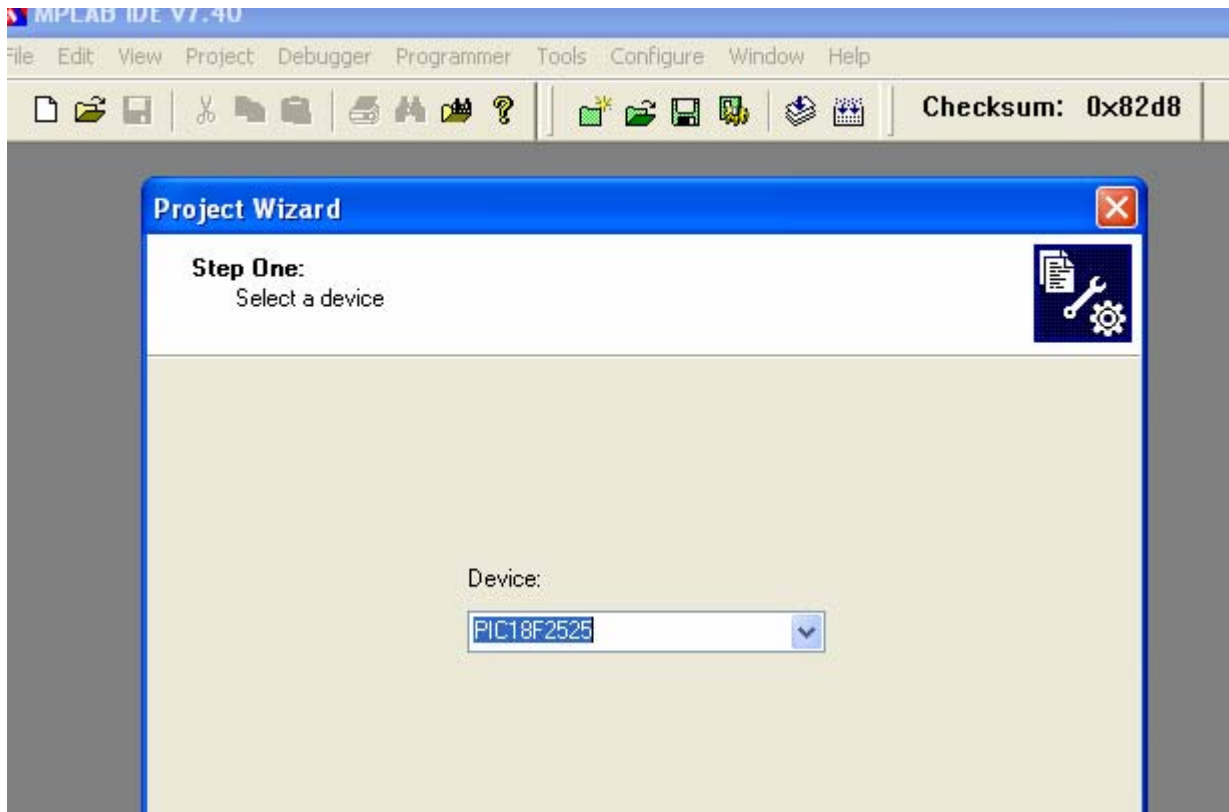
Title	Date Published	Size	D/L
MPLAB C18 v3.02 Student Edition	8/22/2006 10:13:34 AM	18292 KB	
MPLAB C18 v3.02 Student Edition with docs	8/22/2006 10:12:35 AM	23641 KB	
MPLAB C18 (v3.00) C Compiler Getting Started	10/3/2005 8:16:20 AM	3125 KB	

Wir nehmen MPLAB C18 Student Edition with docs und Getting started.

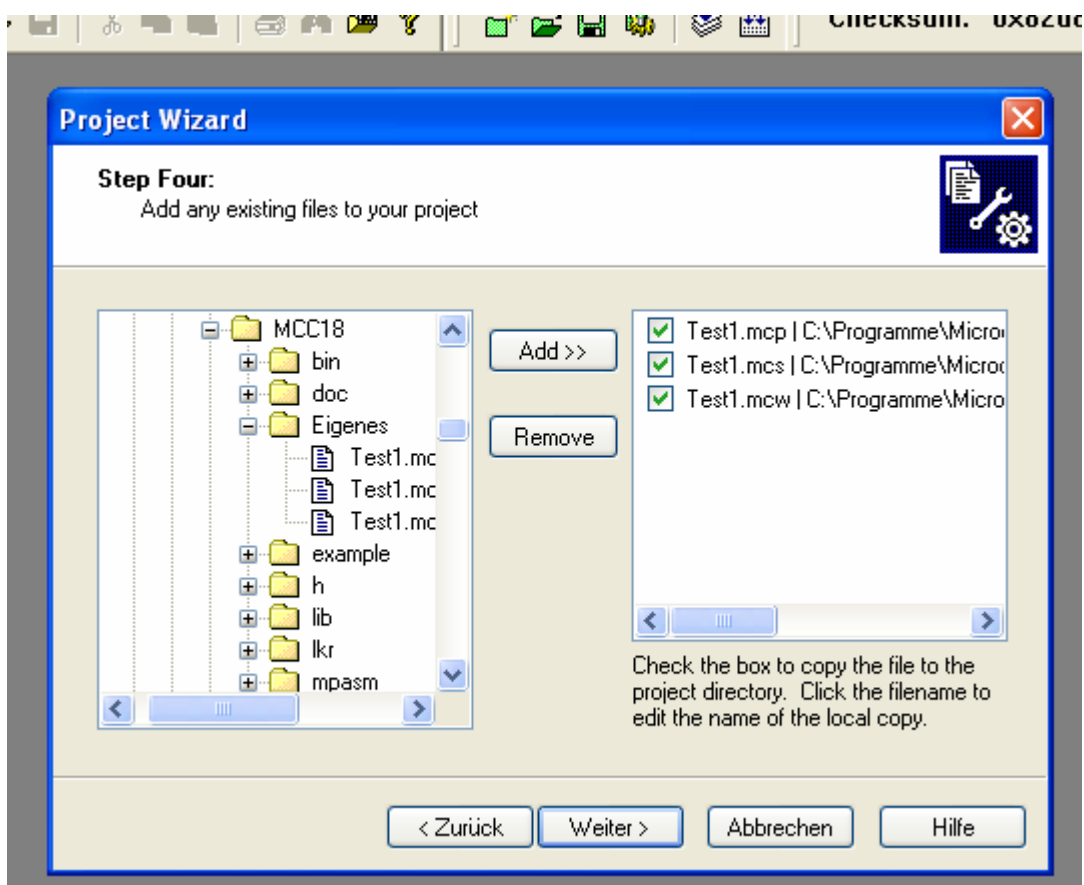
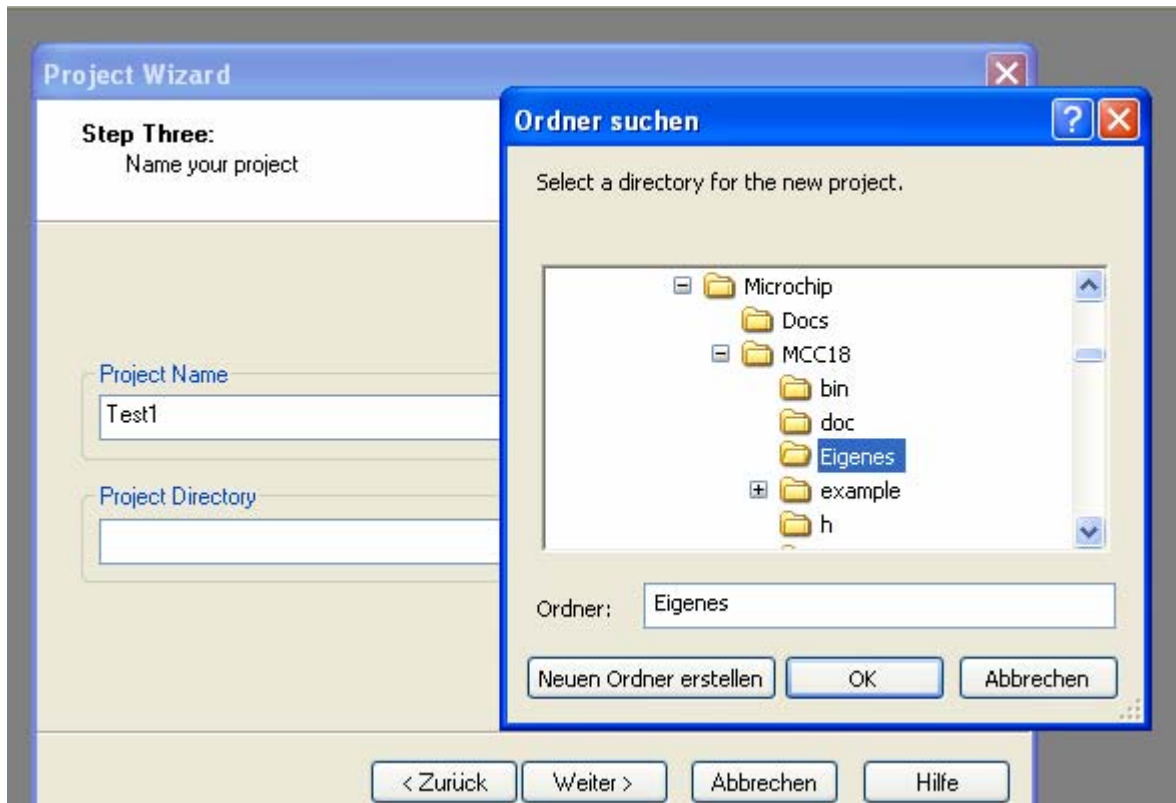
Und installieren in das folgende Verzeichnis (Directory). C:\Programme\Microchip\MCC18.



copyleft:munz
Project Wizzard:

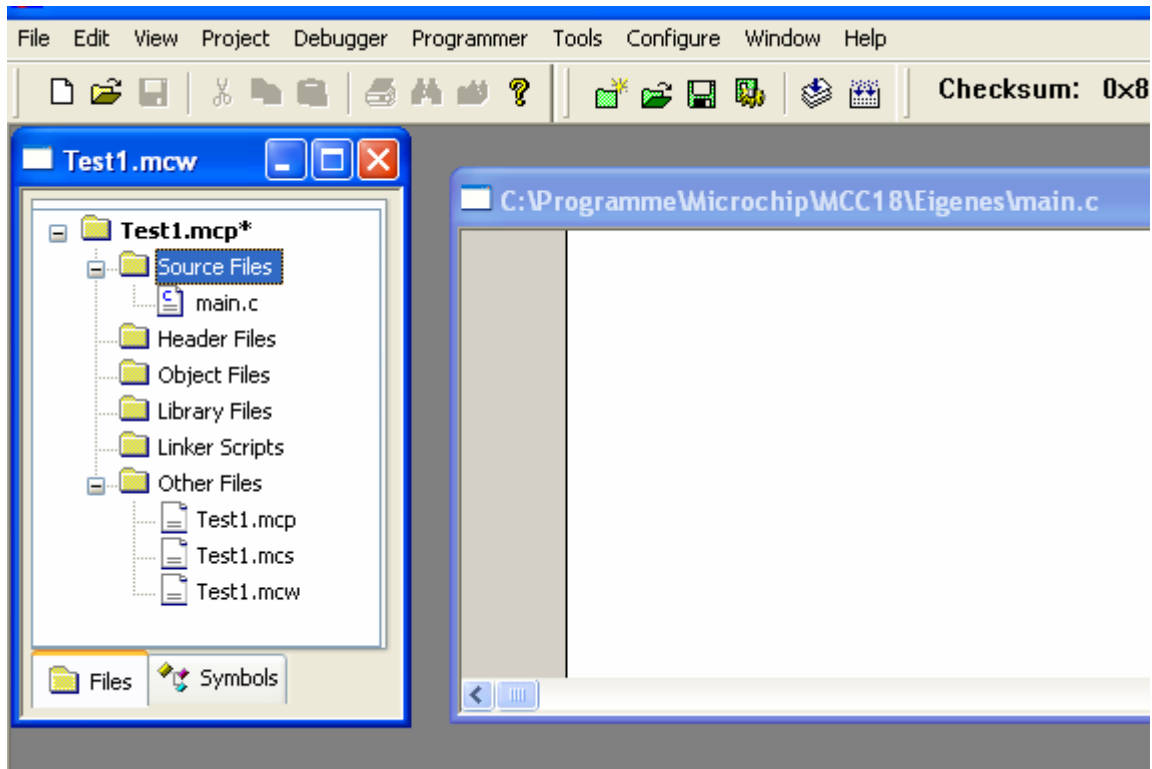


Neuen Ordner erstellen „Eigenes“. Projektname z.B. Test1

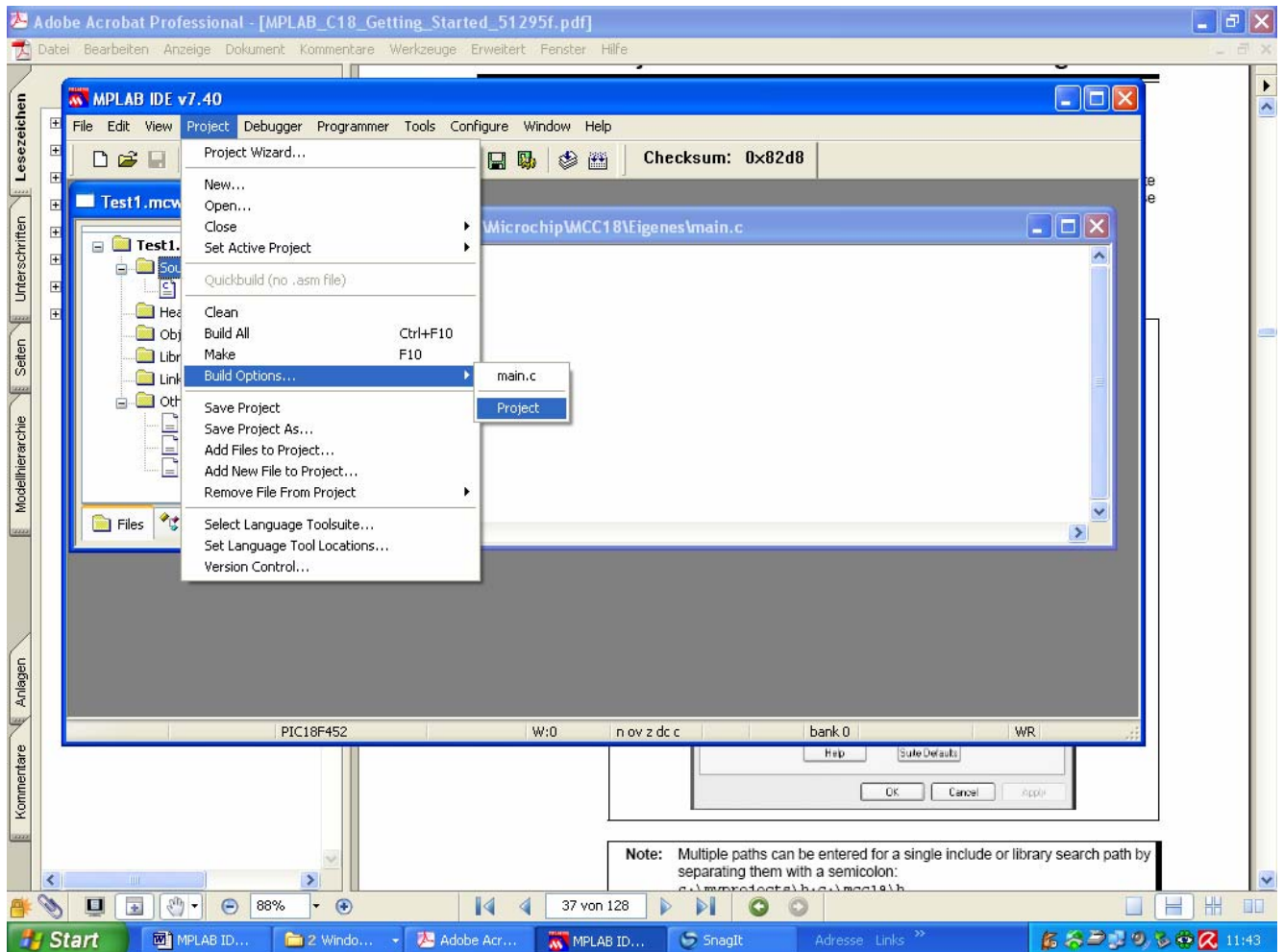


copyleft:munz

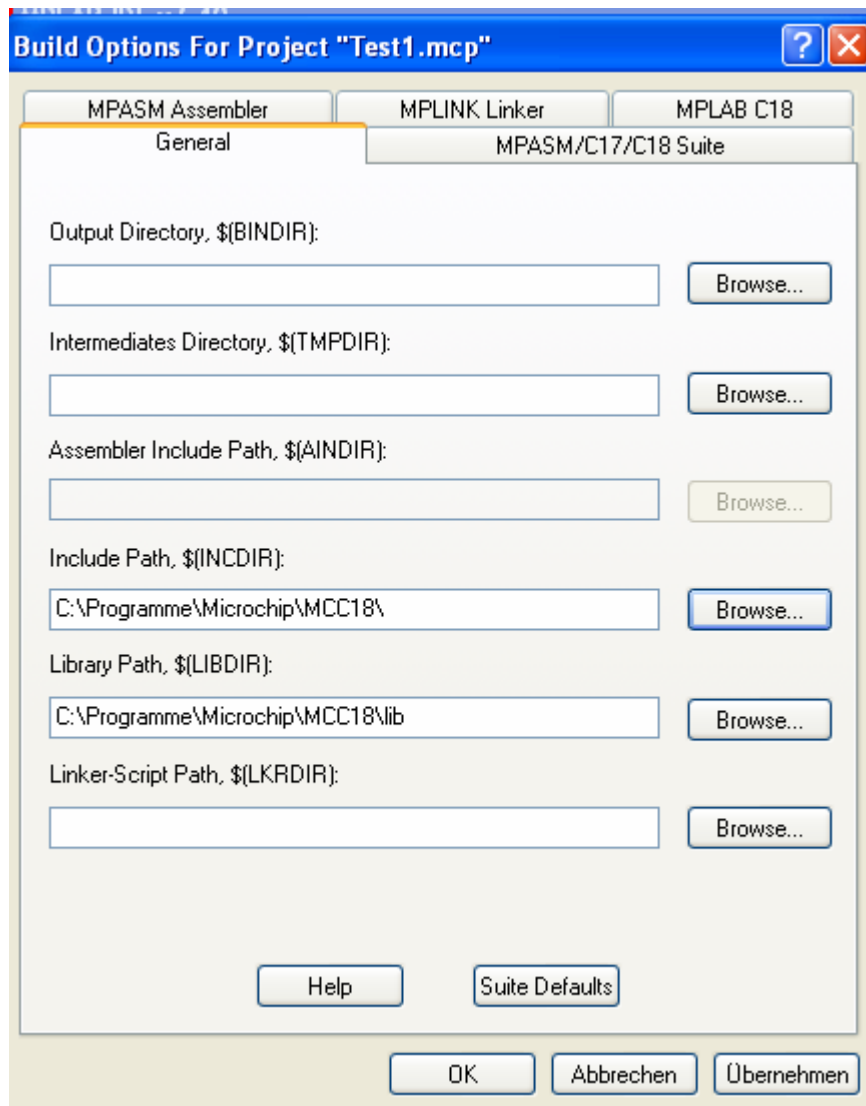
Neues File (neue Datei) erstellen und hinzufügen. → main.c



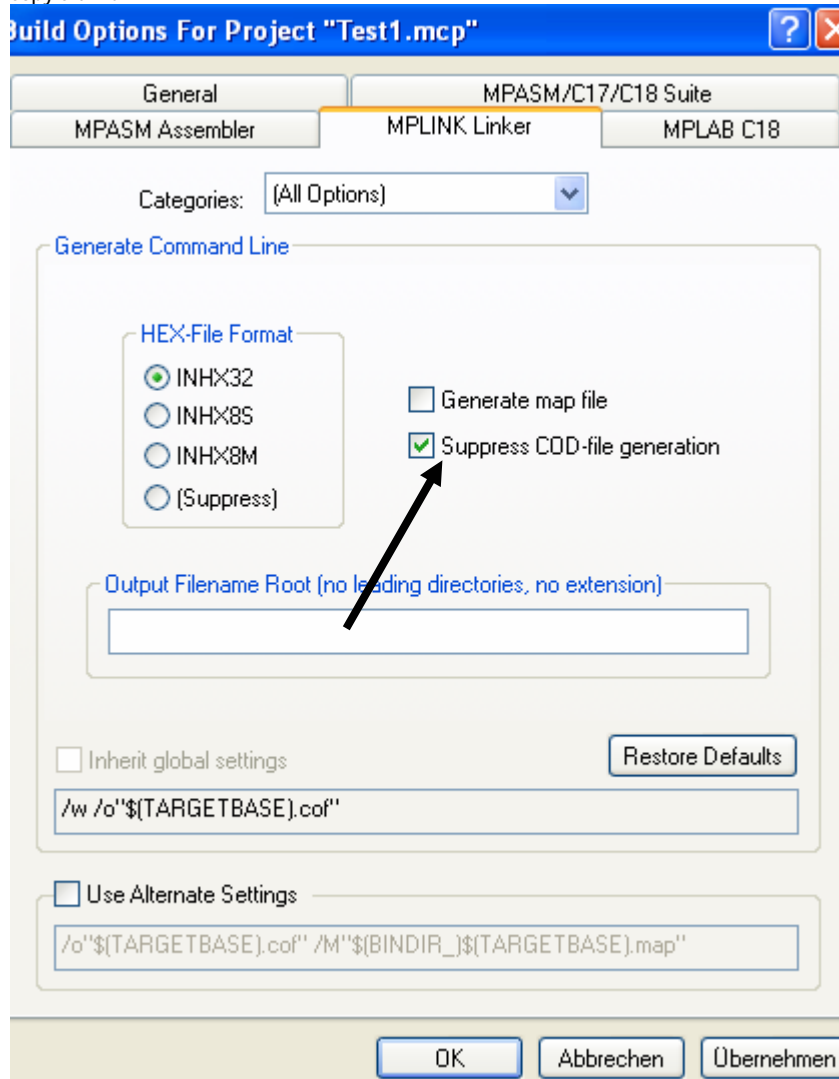
Project Build Options wählen.



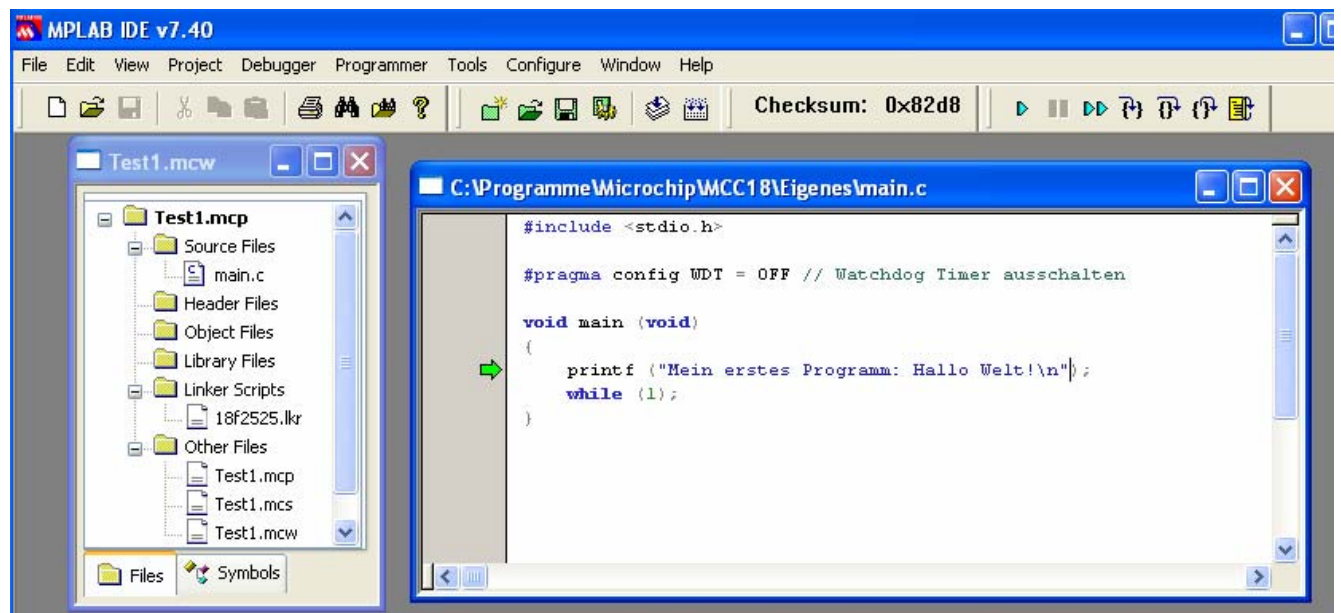
copyleft:munz
Pfade einstellen



Im selben Fenster:

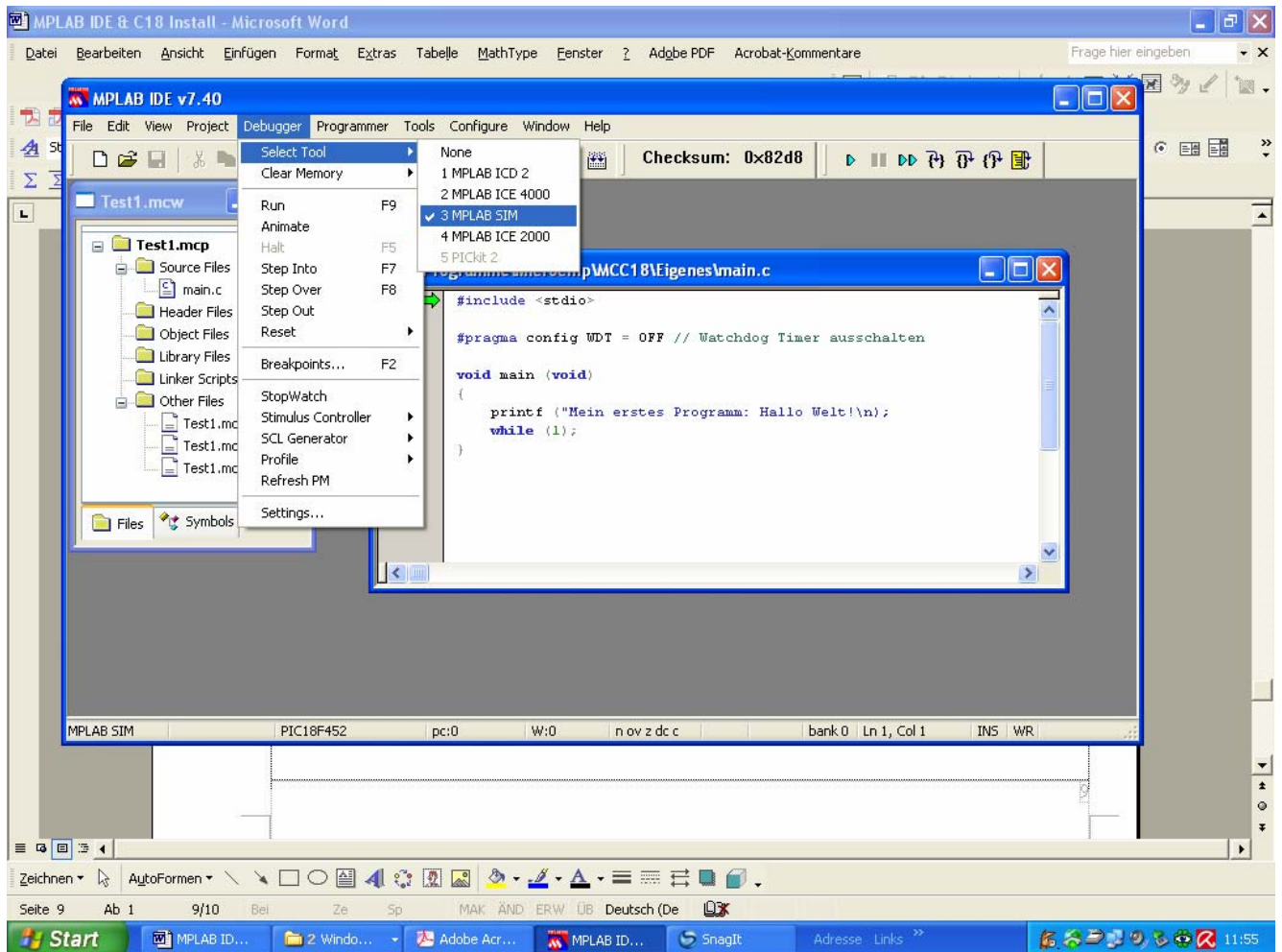


main erstellen: File new → speichern im Projekt Ordner → rechte Maustaste auf Source Files hinzufügen „main.c“.

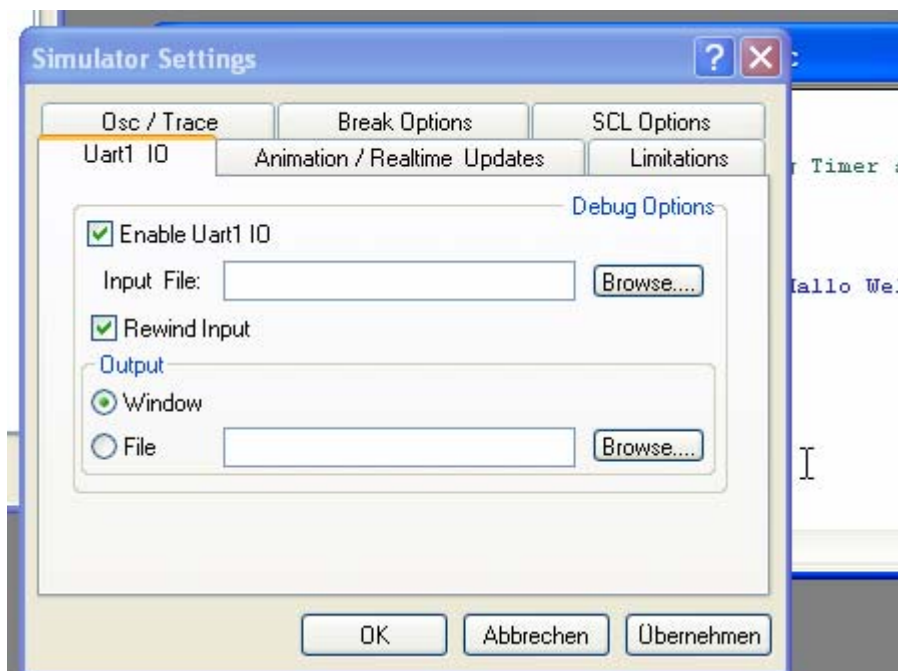


copyleft:munz

→ Debugger – Select Tool → MPSIM

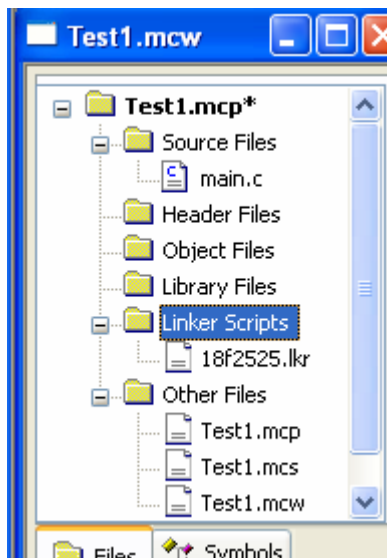
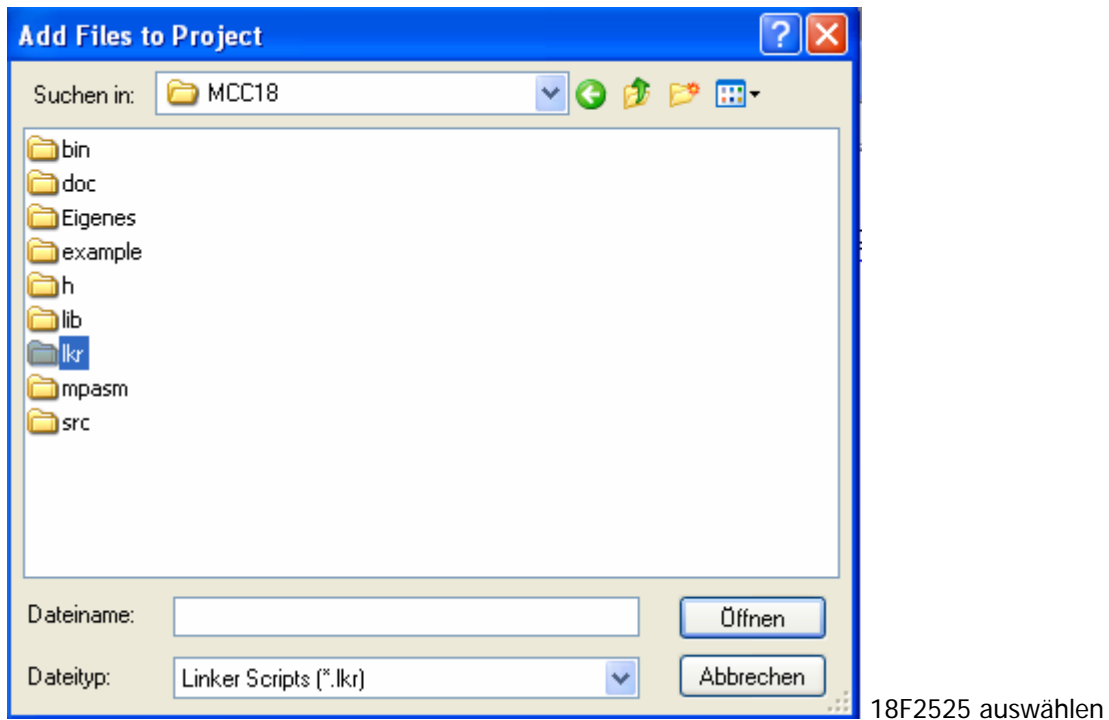
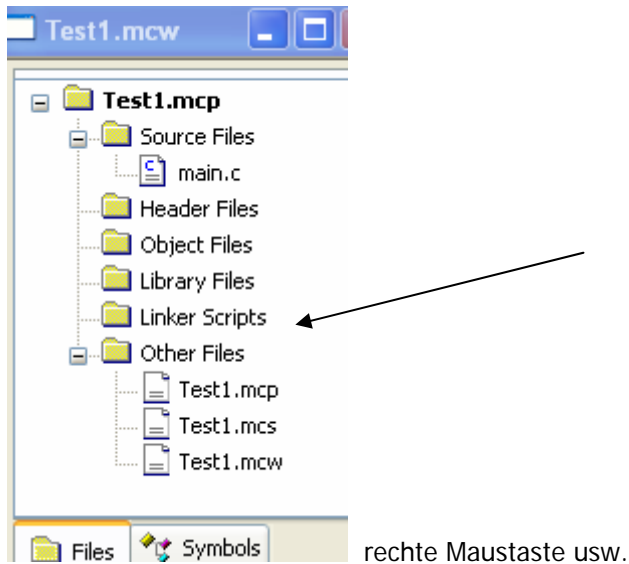


Debugger>Settings → Uart1 IO



copyleft:munz

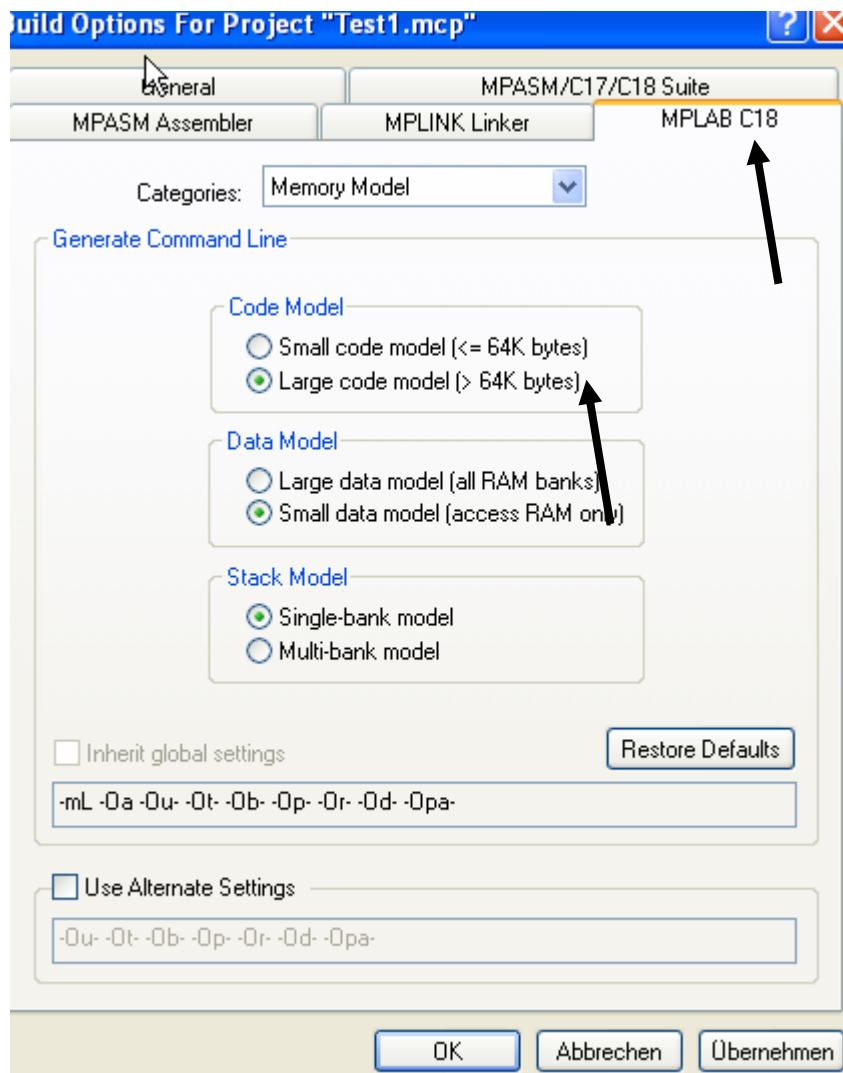
Im Projektfenster → Linkerscript hinzufügen (für später)



copyleft:munz

Memory Model auswählen: Wir verwenden die stdio.h und müssen daher das large code model auswählen.

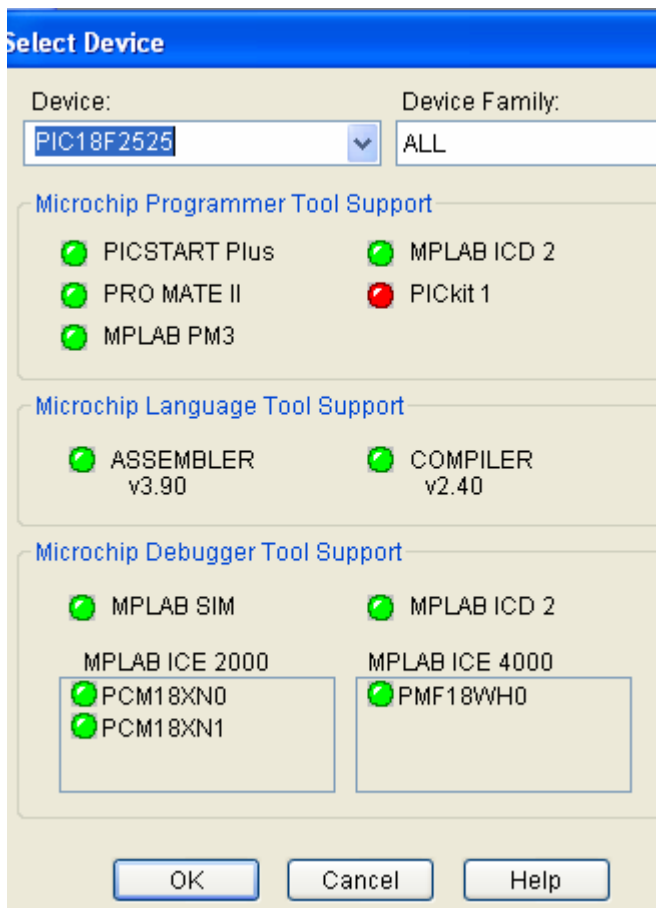
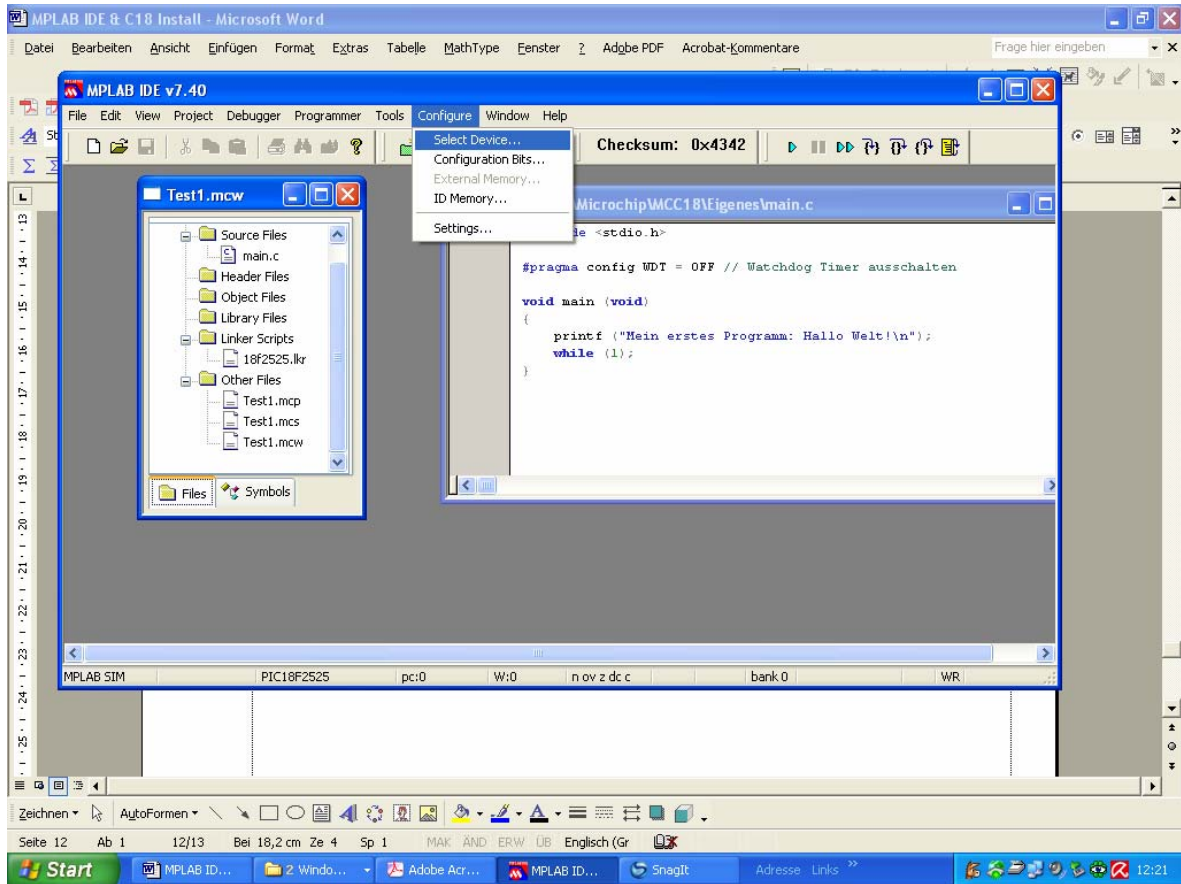
→ Project>Build Options/Project> MPLAB C18 Register



Übernehmen nicht vergessen

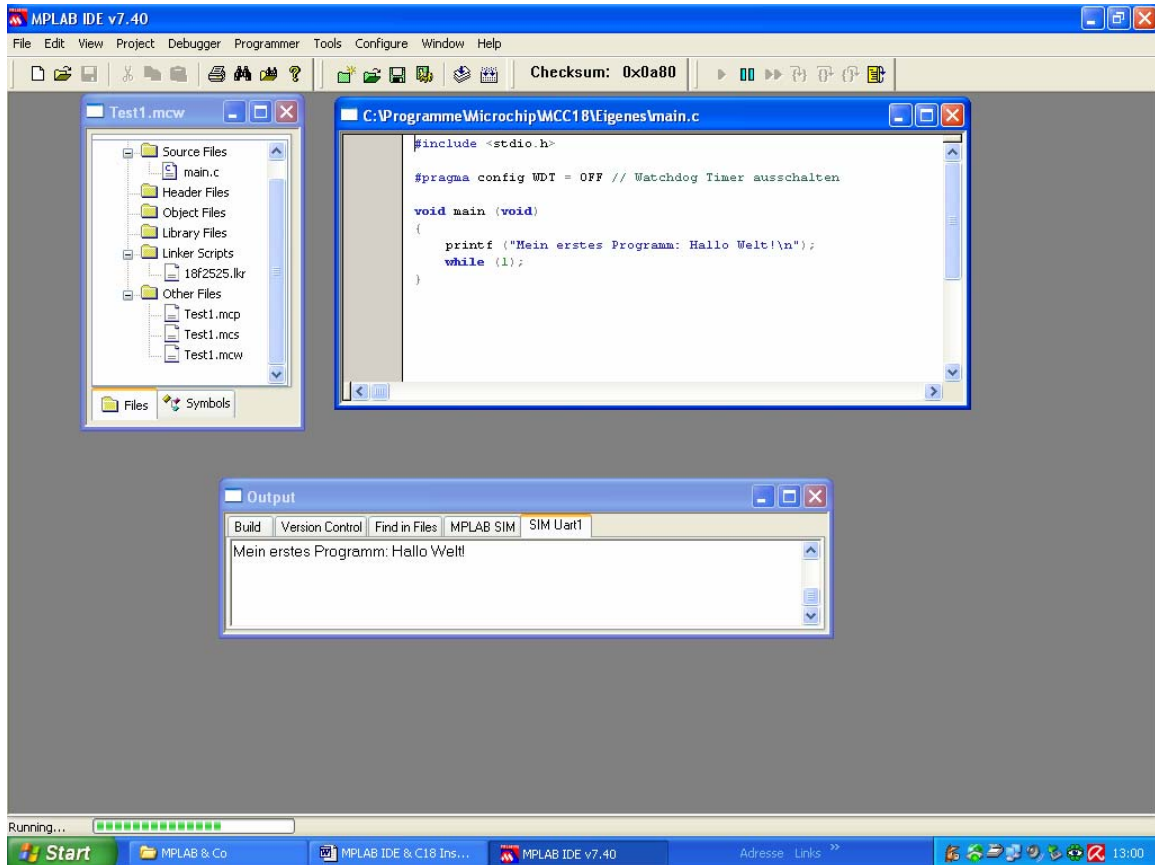
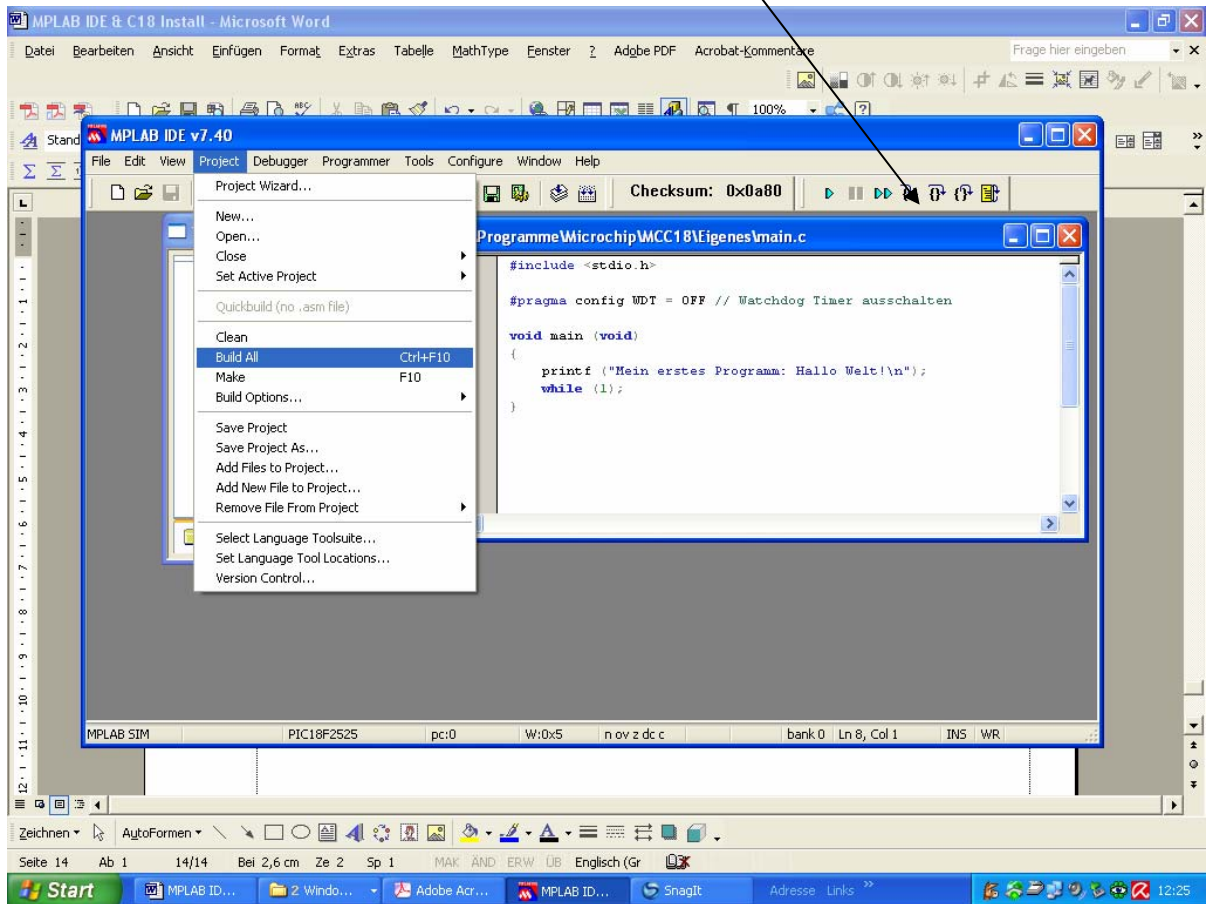
copyleft:munz

Unter Configure Select Device → kann man den richtigen PIC auswählen.



copyleft:munz

Unter Project → Build all compilieren. → Run Icon benutzen. Es wird blau.



Und schon ist unser erstes C – Programm fertig.!