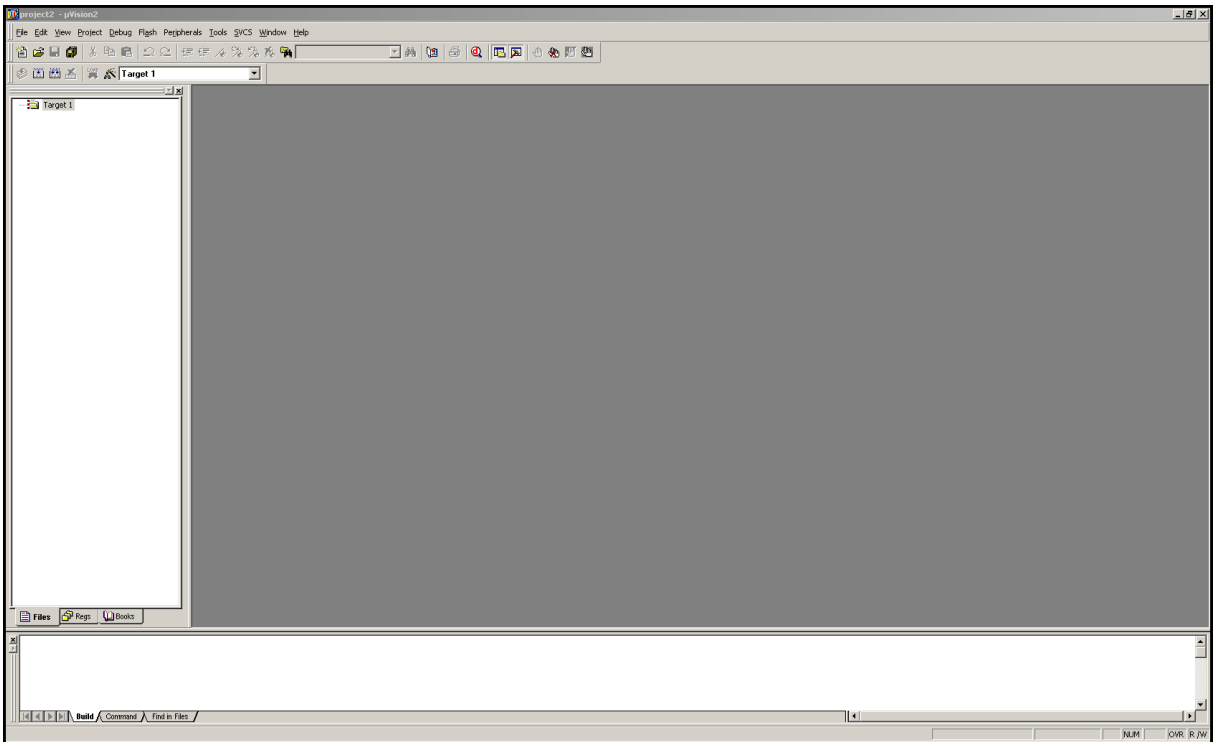
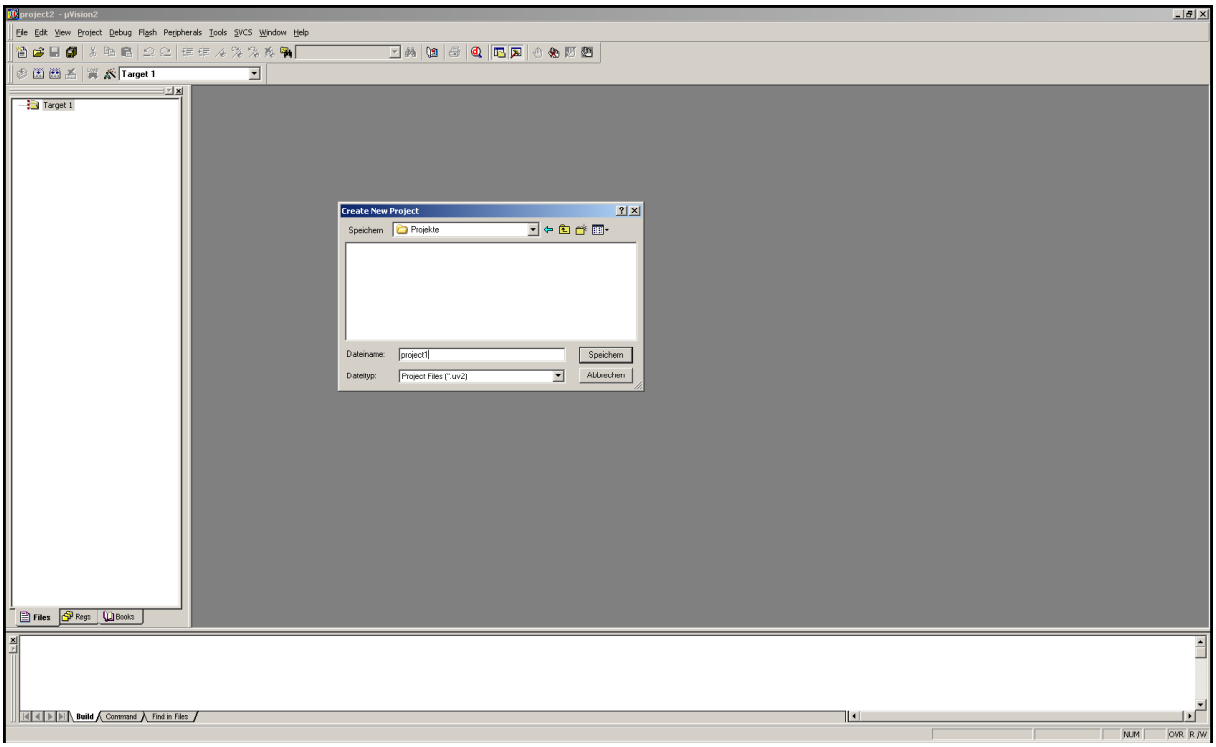
Bevor Sie Keil uVision starten, erstellen Sie sich einen Ordner im Verzeichnis C:/Keil/ wo Sie Ihre Projekte später ablegen. **In diesem Ordner kopieren Sie die Datei *0_template.a51* .**
Diese Datei ist sozusagen eine Vorlage, welche zu Beginn in das Projekt geladen wird.
Diese Datei finden Sie auf der Webseite zum downloaden.

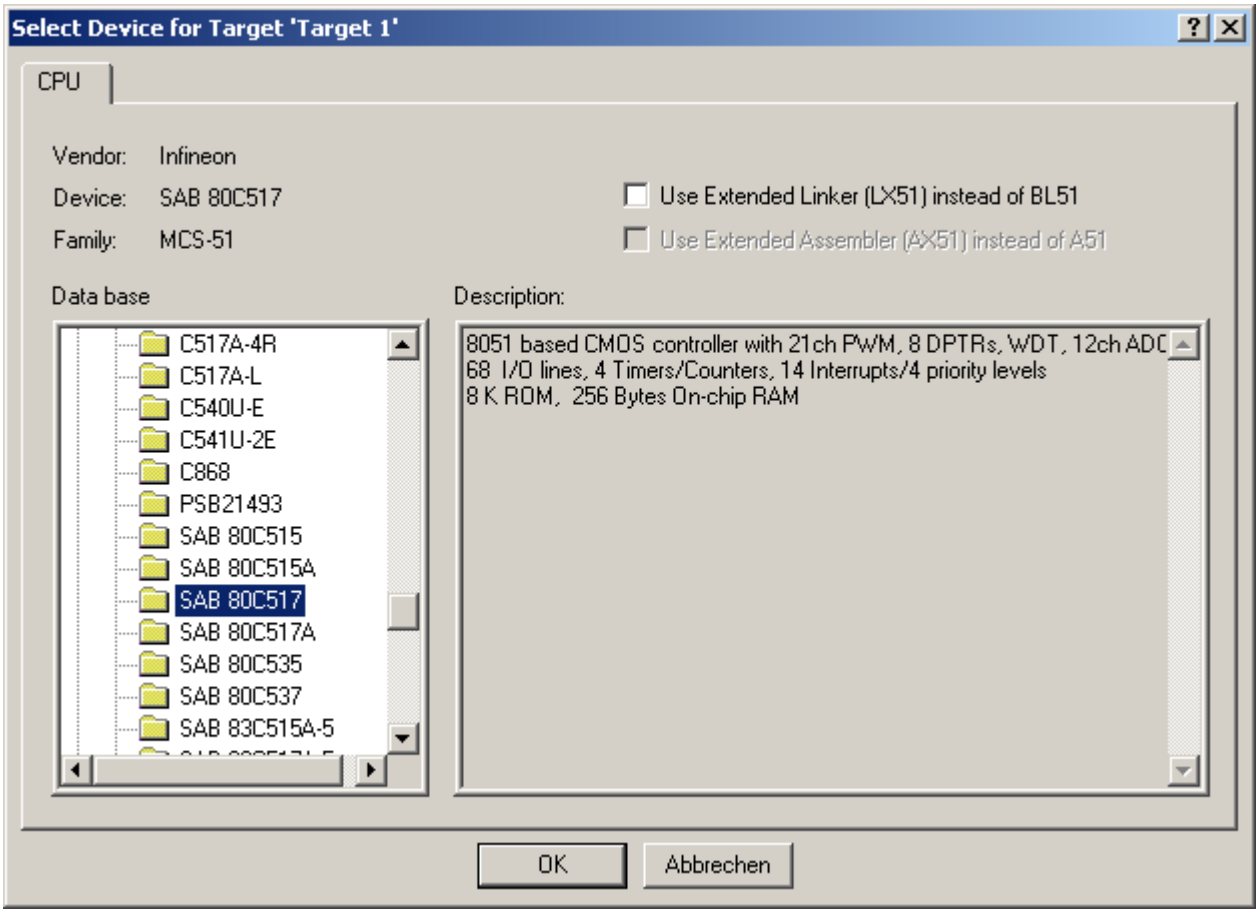
1. Keil starten. Sollten Sie keinen leeren Bildschirm haben, löschen Sie einfach die entsprechenden Dateien und Verzeichnisse.



2. Neues Projekt anlegen, mit einem Projektnamen versehen und abspeichern

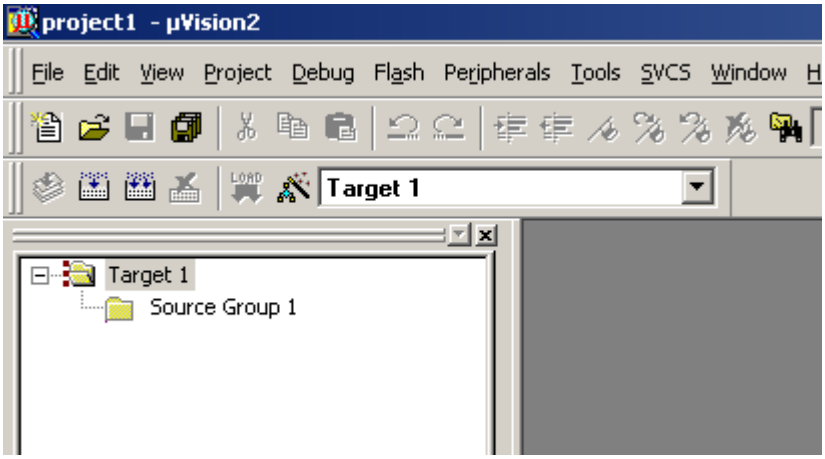


3. Es öffnet sich ein neues Fenster wo Sie den gewünschten Controller auswählen können. Wir wählen den Hersteller Infineon und den Typ SAB80C517.

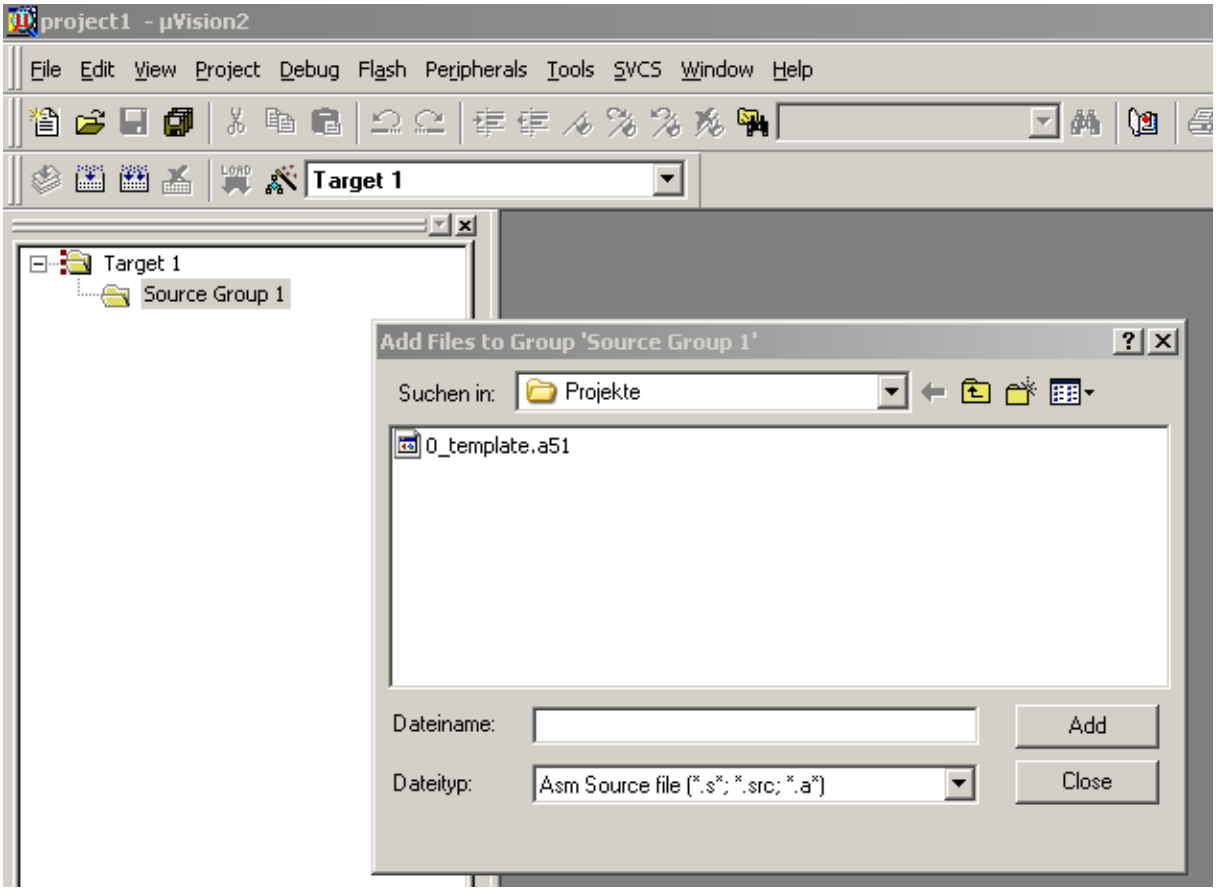


4. Ein neues Fenster wird geöffnet und wir wählen Nein.

5. Unter Target 1 ist nun ein neuer Ordner mit dem Namen Source Group 1 entstanden.

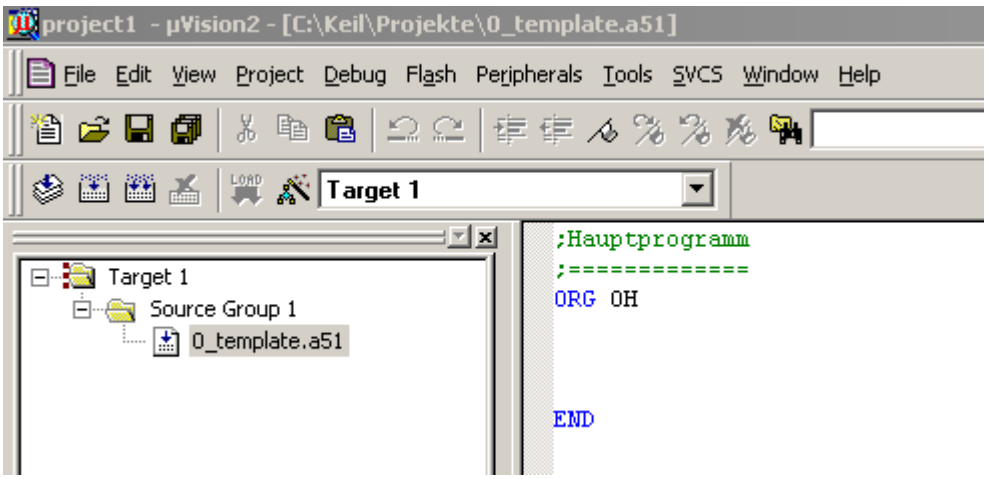


6. Auf Source Group 1 mit der rechten Maustaste klicken und Add Files to Group „Source Group 1“ auswählen. Als Dateityp wählen Sie Asm Source file und und fügen mit Add die Datei *0_template.a51* hinzu. Nach hinzufügen der Datei schließen Sie das Fenster mit Close.



Sie können diese Startdatei (z.B. als leere Datei) auch ganz einfach selbst mit einem Texteditor erstellen. Z.B. *startdatei.a51*

Unter Source Group 1 finden Sie nun die Datei. Mit einem Doppelklick darauf wird die Datei in den Editor geladen und wir können jetzt unser erstes Assemblerprogramm schreiben.



Der Eintrag ORG OH wäre hier gar nicht notwendig. Wichtig ist aber der Befehl END.

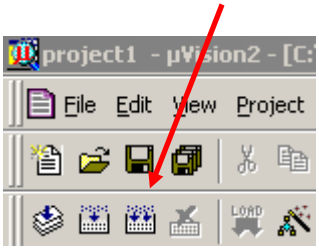
7. Das erste Assemblerprogramm

Aufgabe:
1.) 21h in den Akku laden
2.) Den Inhalt vom Akku auf die Speicherzelle 35h übertragen
3.) Den Inhalt von Speicherzelle 35h in Speicherzelle 36h speichern

```
;Hauptprogramm
;==============
ORG 0H

mov A,#21h      ;Die Zahl 21h in den Akku laden
mov 35h,A       ;Den Inhalt vom Akku auf SZ 35h übertragen
mov 36h,35h     ;Speichere den Inhalt von 35h auf 36h

END
```

Nun müssen wir das Programm ÜBERSETZEN und LINKEN!

Direkt über das Menü oder über Project/ Rebuild all target files



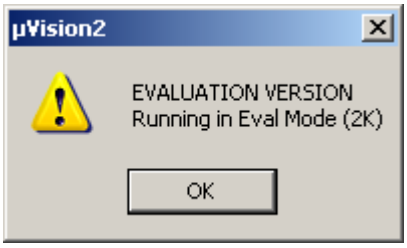
Im Output Window sehen Sie, dass keine fehler bzw. Warnungen gefunden wurden. Der Assemble- und Linkvorgang war erfolgreich.

```
Build target 'Target 1'
assembling 0_template.a51...
linking...
Program Size: data=8.0 xdata=0 code=7
"project1" - 0 Error(s), 0 Warning(s).
```

Wir starten den Simulator mit dem Debugger! Der Debugger ermöglicht ein schrittweises Ausführen des Programmcodes.

START / STOP

In der Evaluation Version erhalten Sie jetzt die Meldung, dass Sie max 2kByte Daten verarbeiten können. Dies reicht aber am Anfang aus!



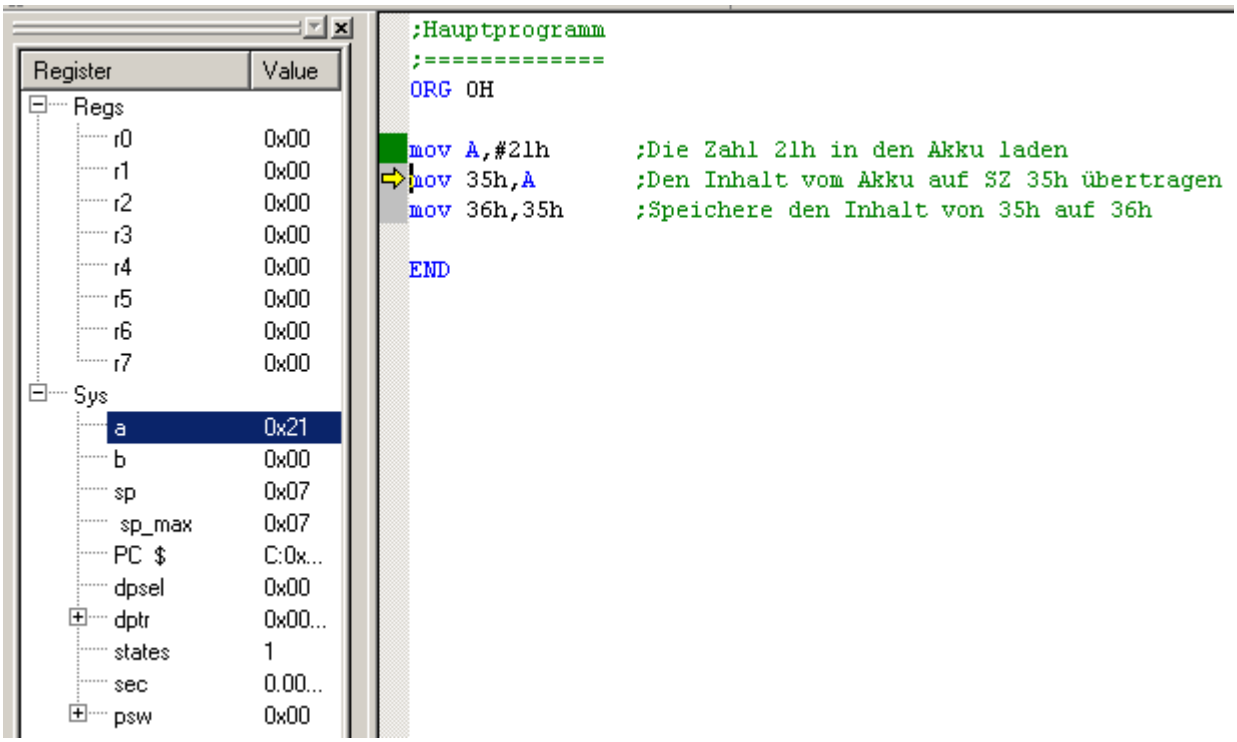
Wenn Sie den Debugger starten, setzt sich der Programmzeiger an den Anfang des 1. Befehles.
Im linken Registerfenster stehen die Register und die derzeitigen Werte.

```
;Hauptprogramm
;=============
ORG 0H

mov A,#21h      ;Die Zahl 21h in den Akku laden
mov 35h,A       ;Den Inhalt vom Akku auf SZ 35h übertragen
mov 36h,35h     ;Speichere den Inhalt von 35h auf 36h

END
```

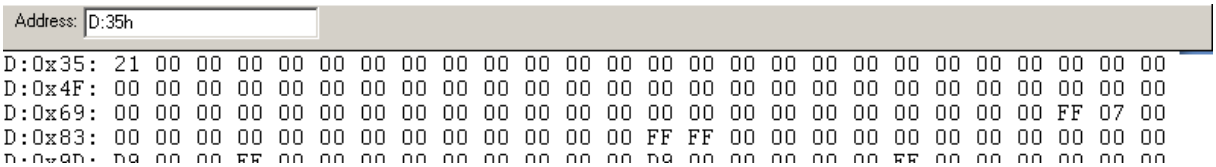
Beachten Sie das Registe a (a = Akku)!
Der Wert ist 00.

Mit F10 wird diese Zeile abgearbeitet und Sie springen zur nächsten Zeile.

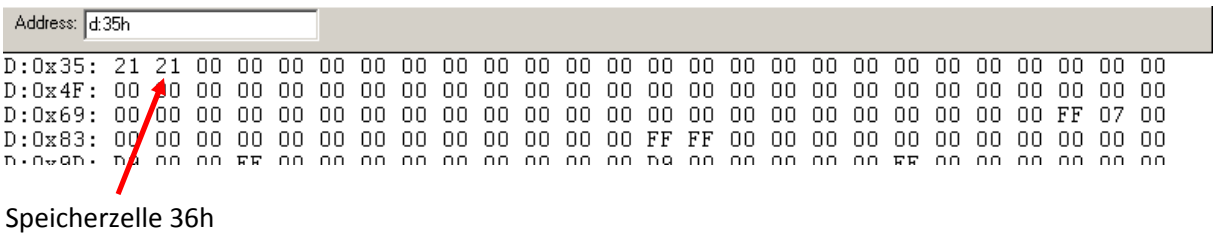
Im linken Registerfenster steht nun die Zahl 21h im Akku.



Weiter mit F10. Mit mov 35h, A wird nun der Inhalt des Akku an die Speicherzelle 35h geschrieben.
Nach Abarbeiten des Befehls müsste dort der Wert 21h stehen.
In der nächsten Programmzeile wird der Inhalt vom Akku in die Speichezelle 35h geschrieben.
Dies können Sie überprüfen, wenn Sie im Adressfenster D:35h eingeben.
D steht für die direkte Adressierung des internen RAM-Speichers.



Wenn das Fenster bei Ihnen nicht aktiv ist, können Sie es über View/Memory Window aktivieren.
Im folgenden Schritt wird der Inhalt von Speicherzelle 35h in die Speicherzelle 36h geschrieben.



Speicherzelle 36h

Specifier	Description
B	Bit–addressable RAM memory (**BIT**).
C	Code memory (**CODE**).
CO	Memory range for constants (251 **CONST**).
D	Internal directly-addressable RAM memory of the 8051 (**DATA**).
EB	Extended bit-addressable RAM memory (251 **EBIT**).
ED	Extended data RAM memory (251 **EDATA**).
HC	Huge memory range for constants (251 **HCONST**).
I	Internal indirectly-addressable RAM memory of the 8051 (**IDATA**).
X	External RAM memory (**XDATA**).