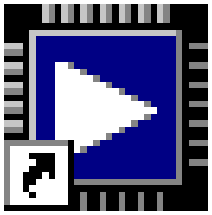
DSP

Assemblerprogrammierung

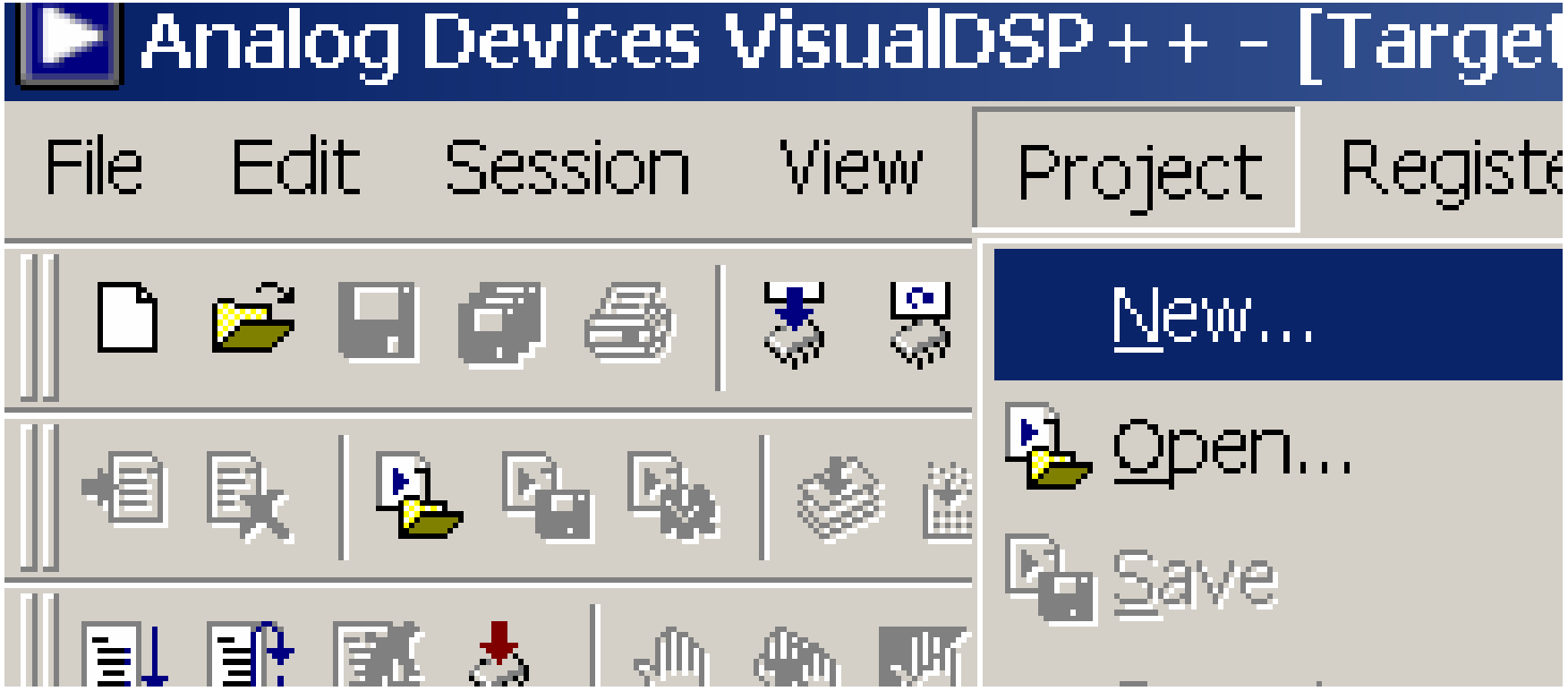
Entwicklungsablauf

Integrierte Entwicklungsumgebung starten

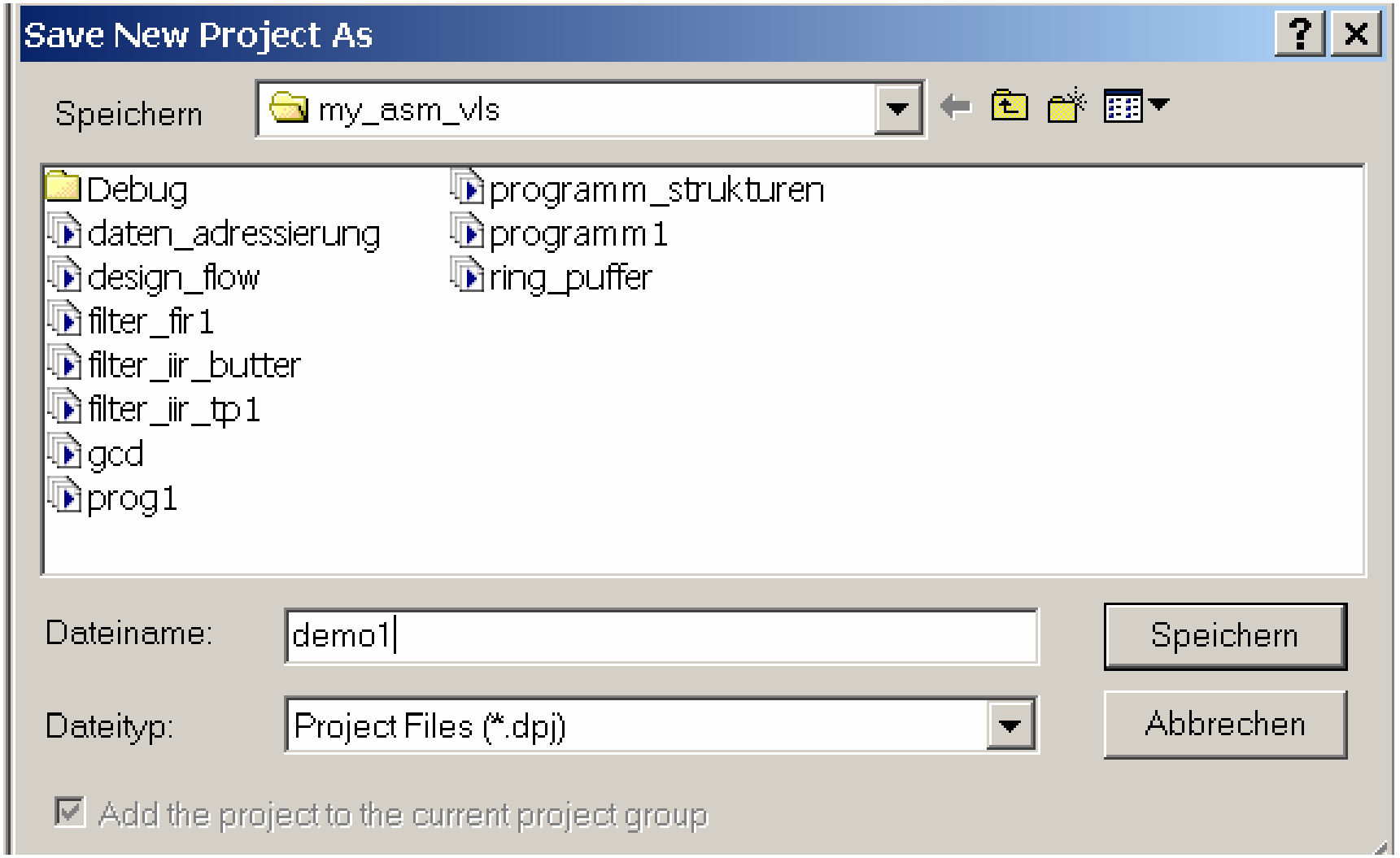


VisualDSP++ Environment.lnk

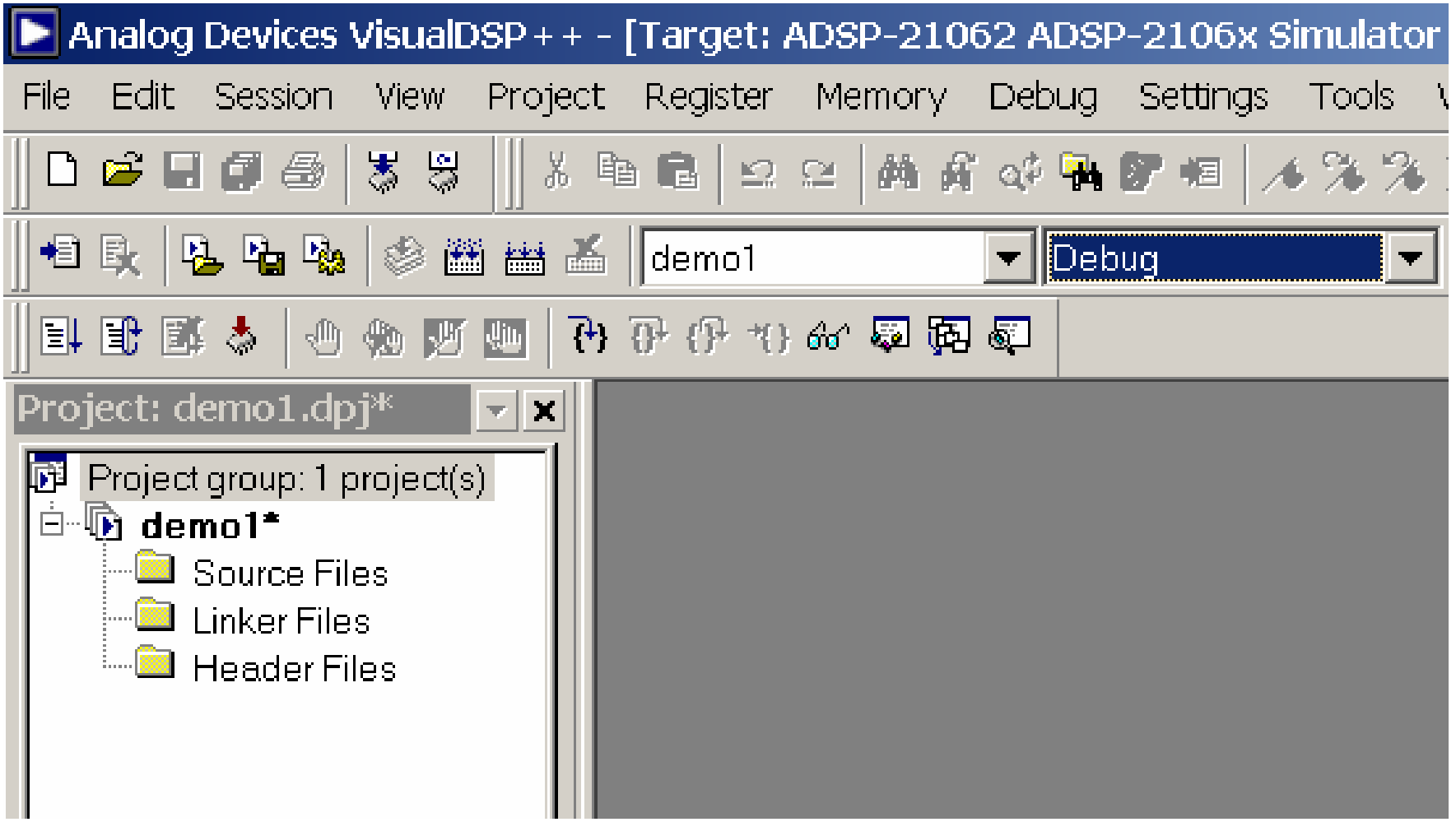
Neues Projekt erstellen



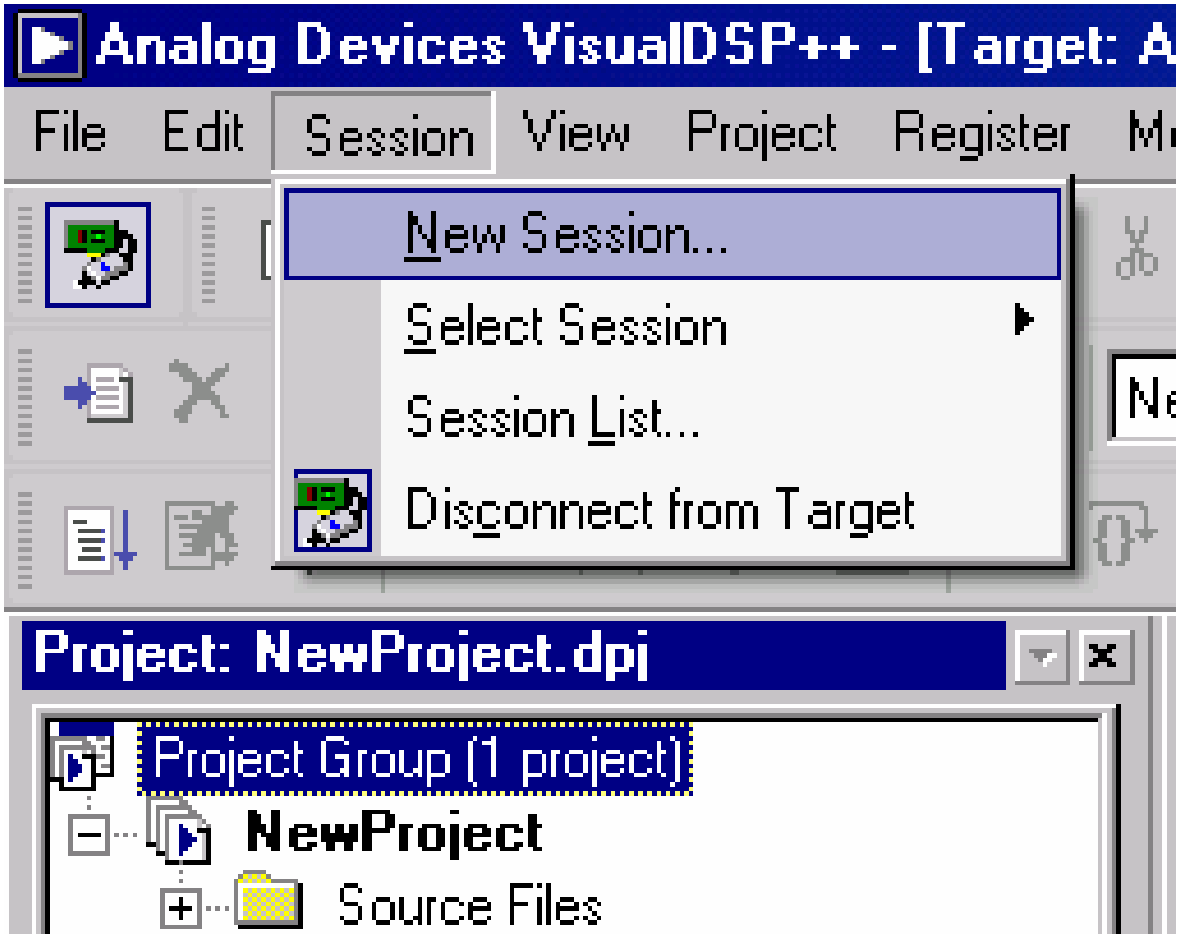
Neues Projekt erstellen



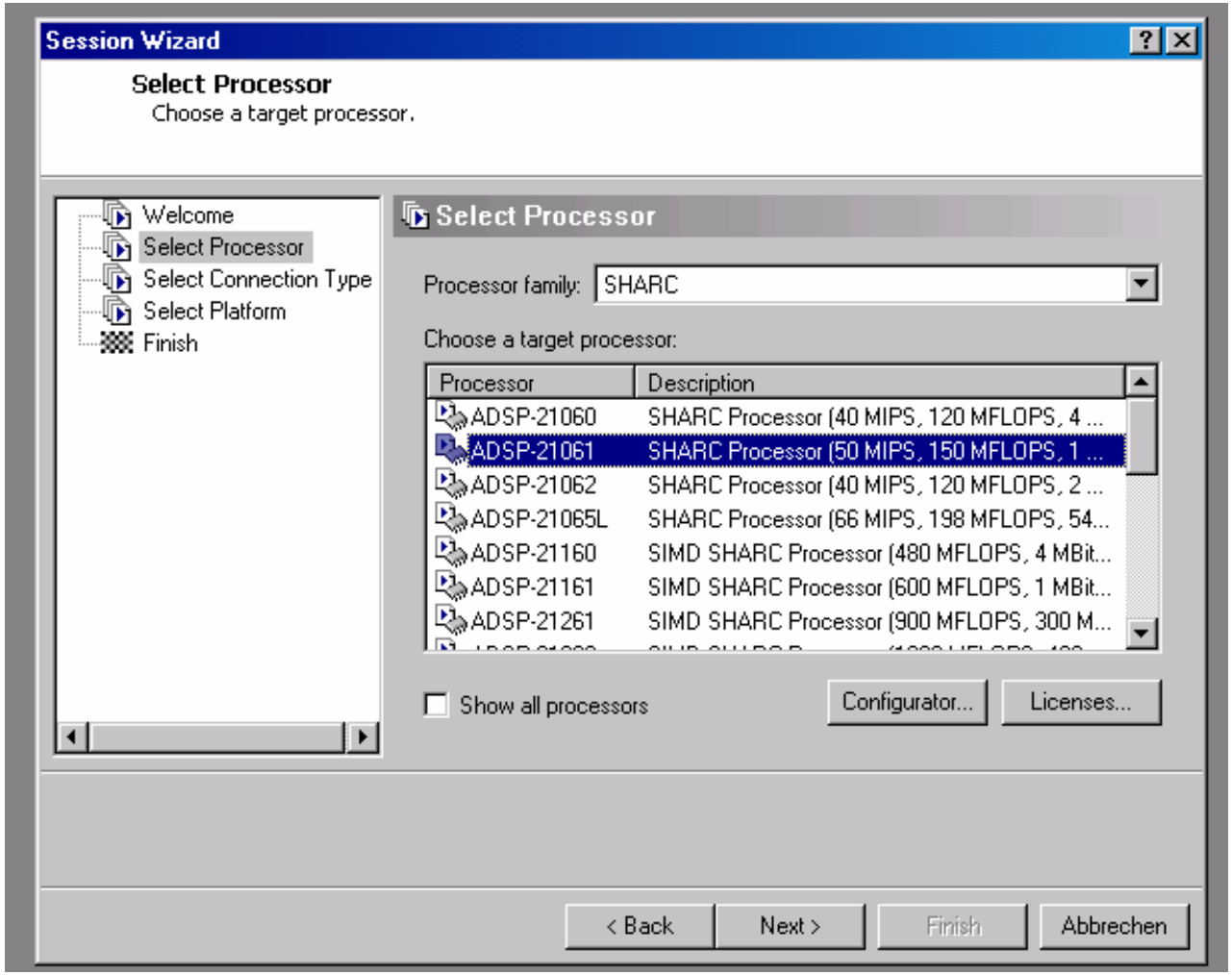
Neues Projekt erstellen



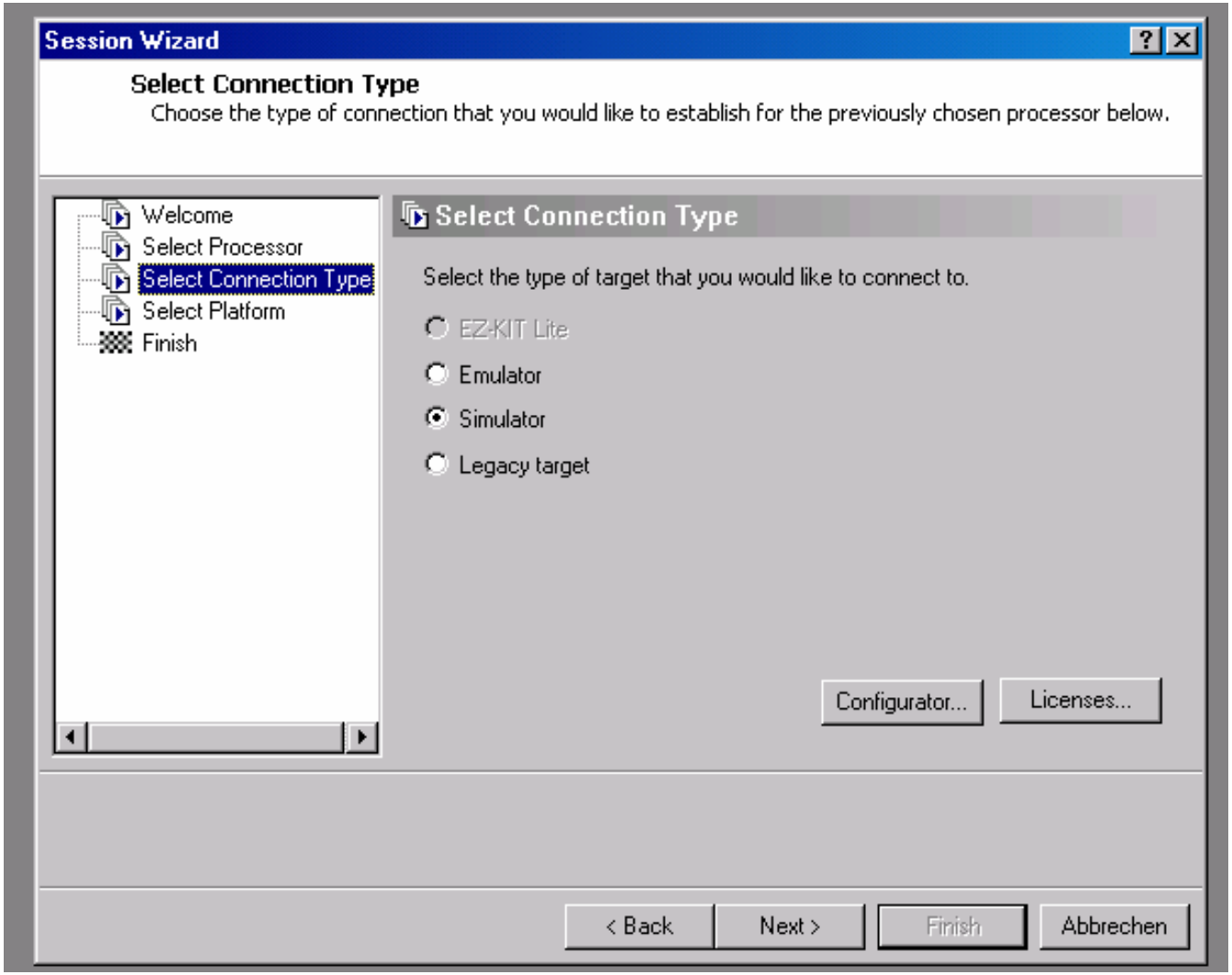
Session auswählen oder definieren



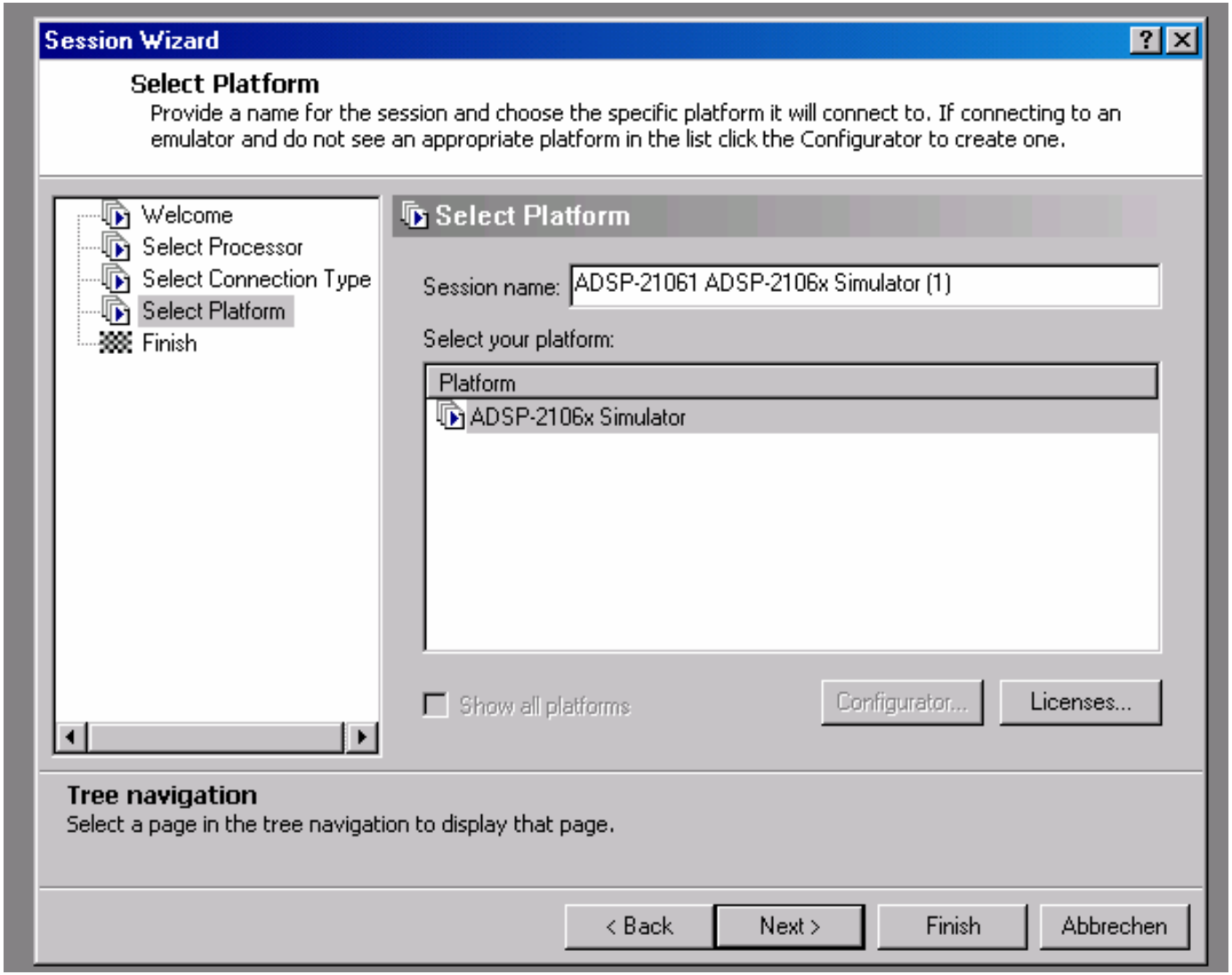
Session definieren



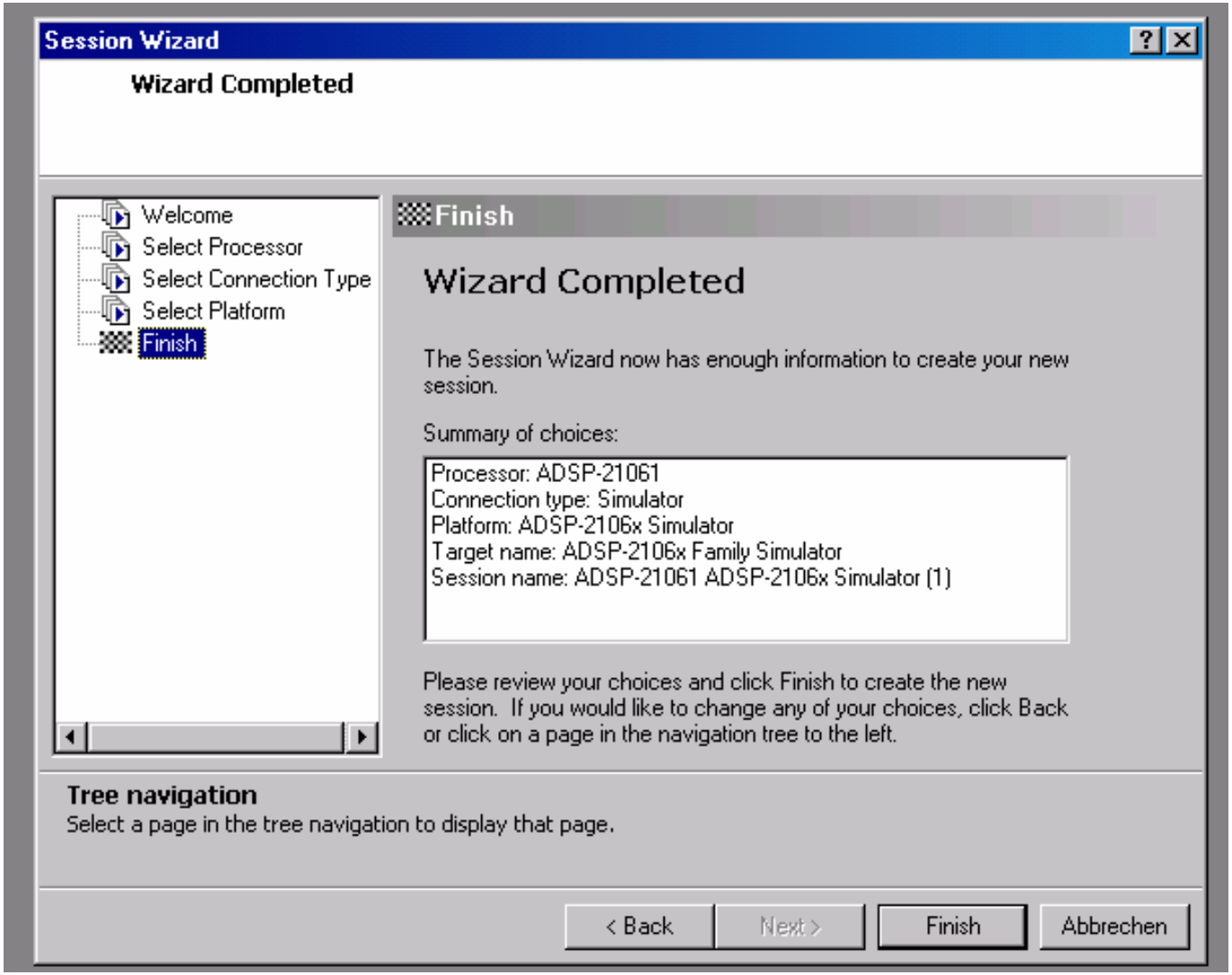
Session definieren
Simulator wählen



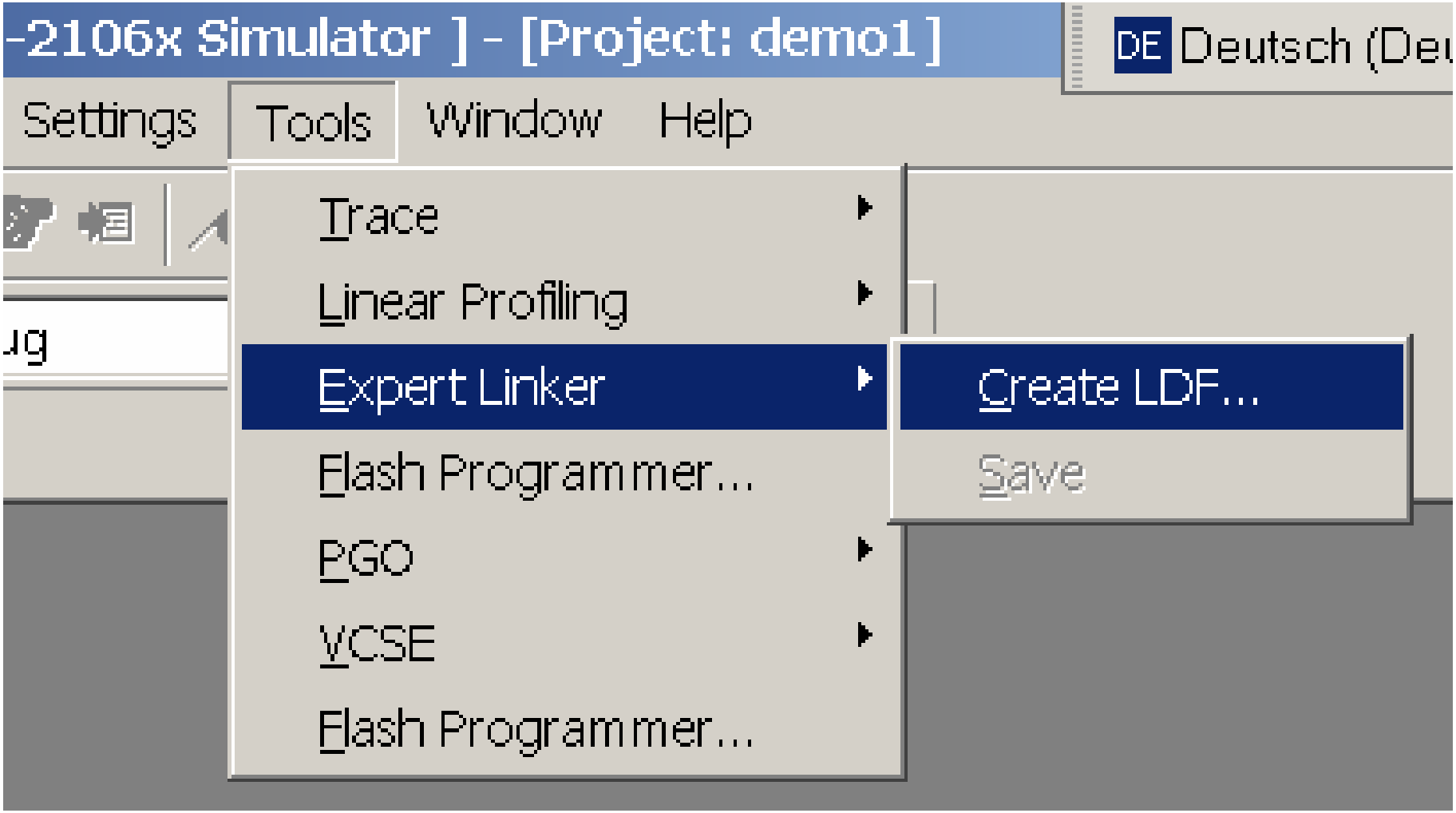
Session definieren



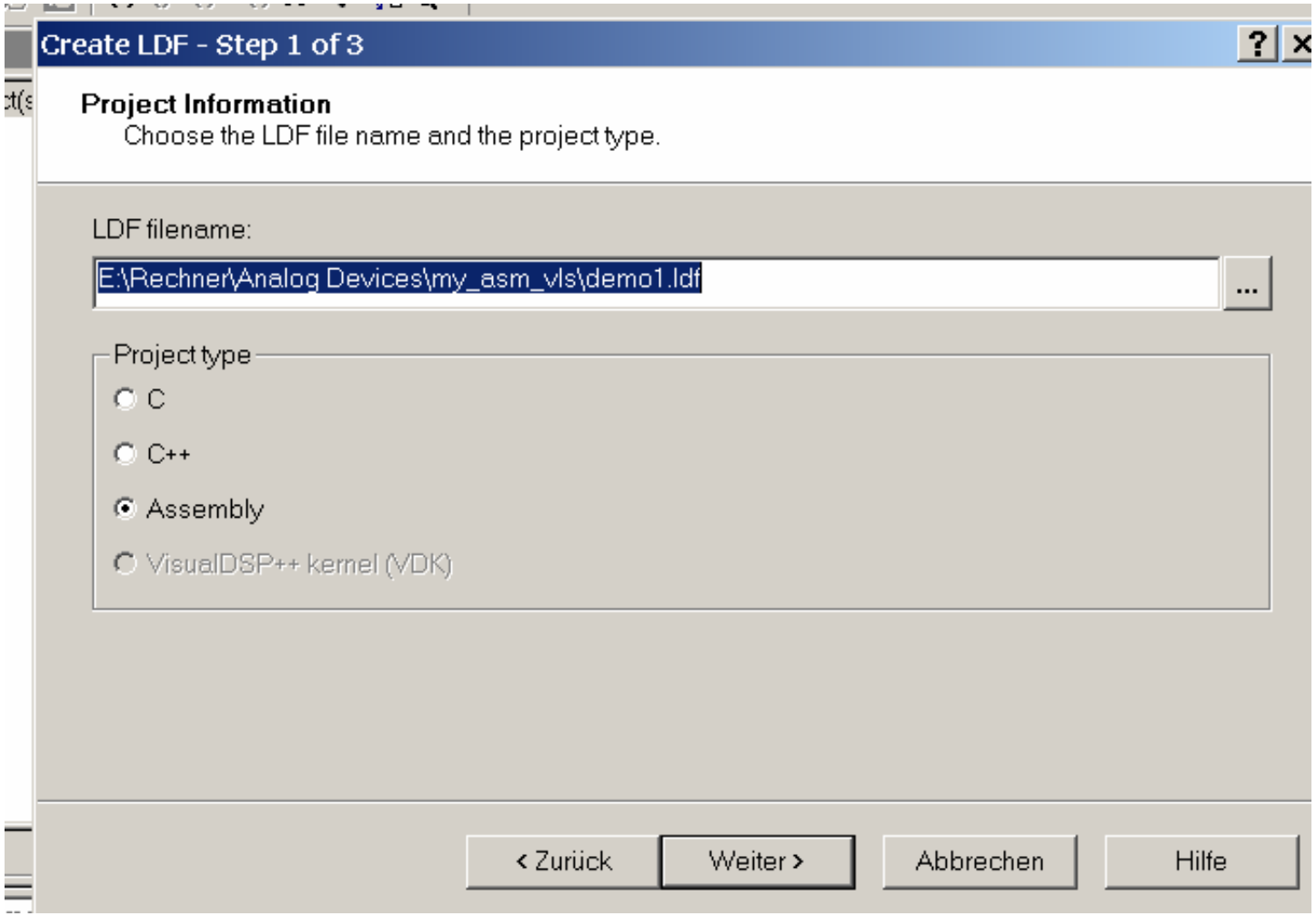
Session definieren- Ergebnis



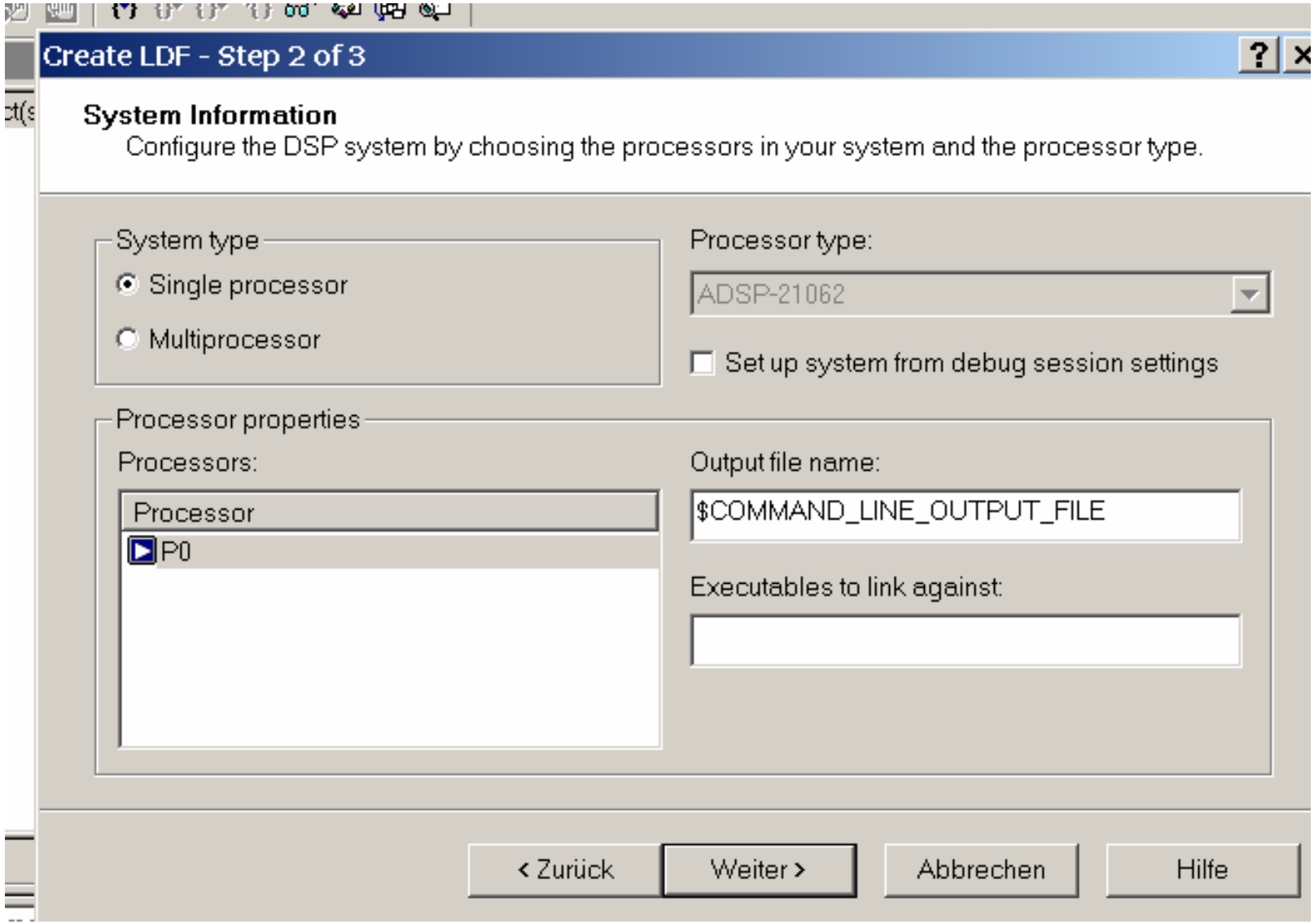
Link-File erstellen



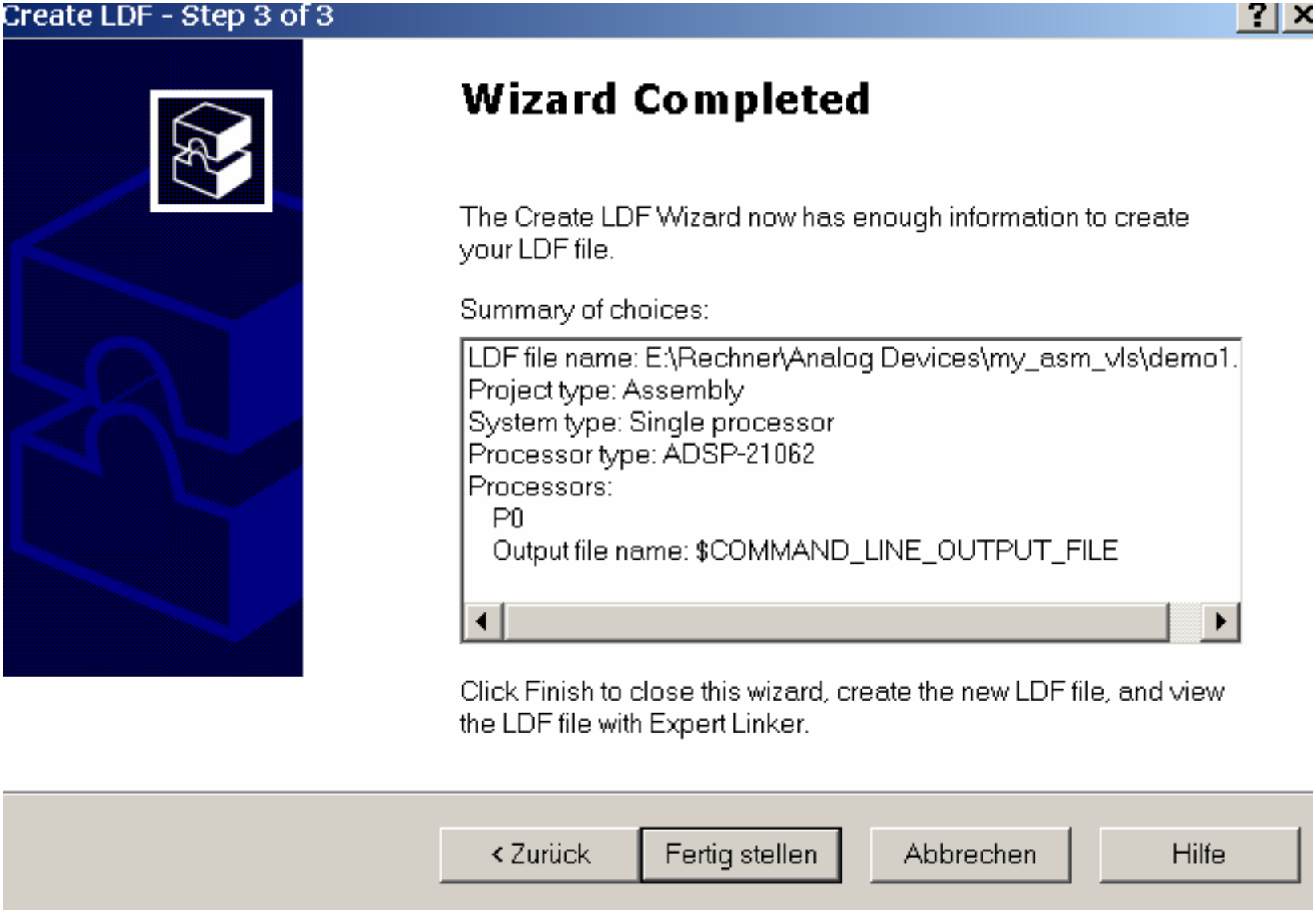
Link-File erstellen
Assembly wählen



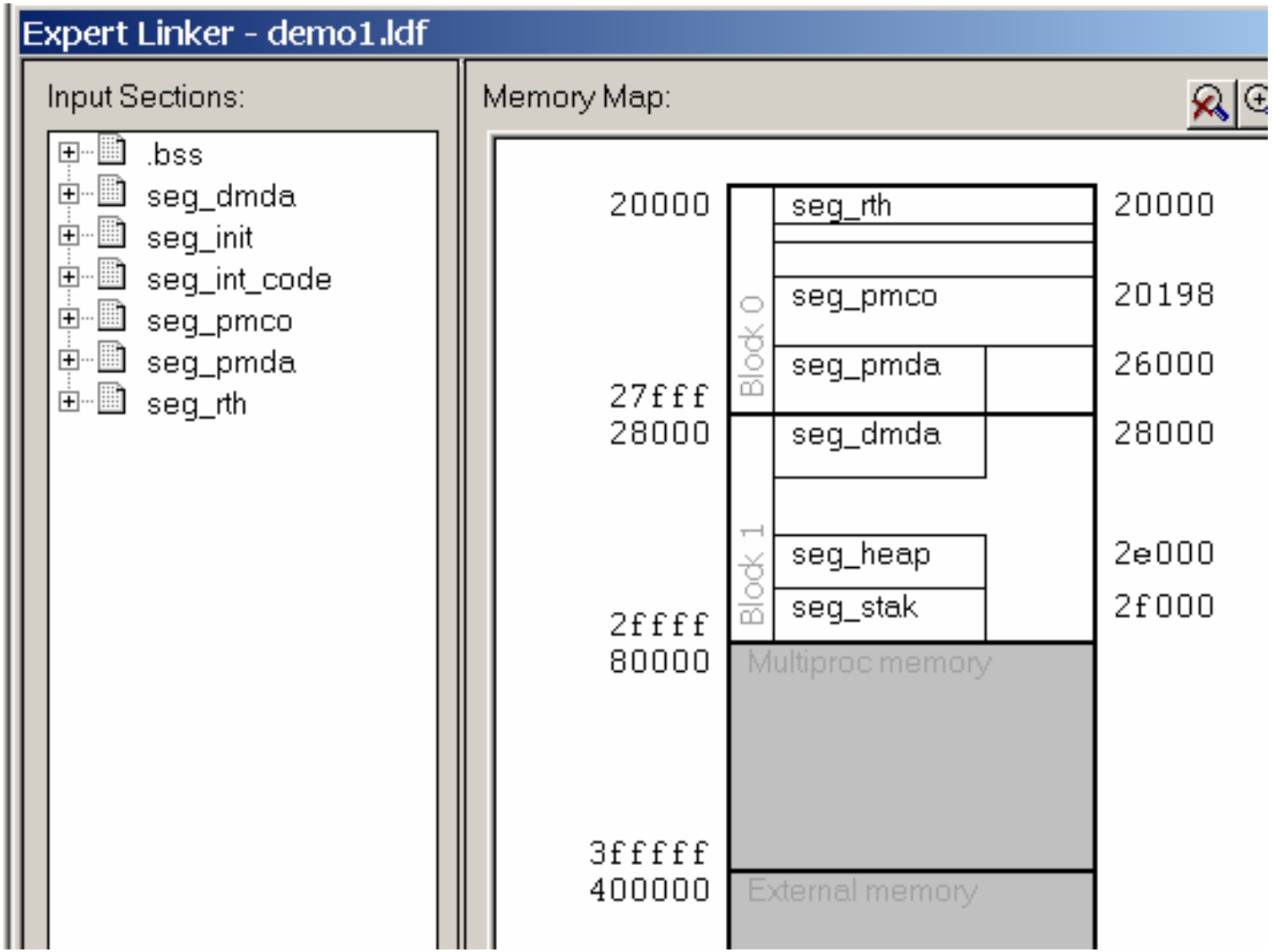
Link-File erstellen



Link-File erstellen



Link-File Ergebnis

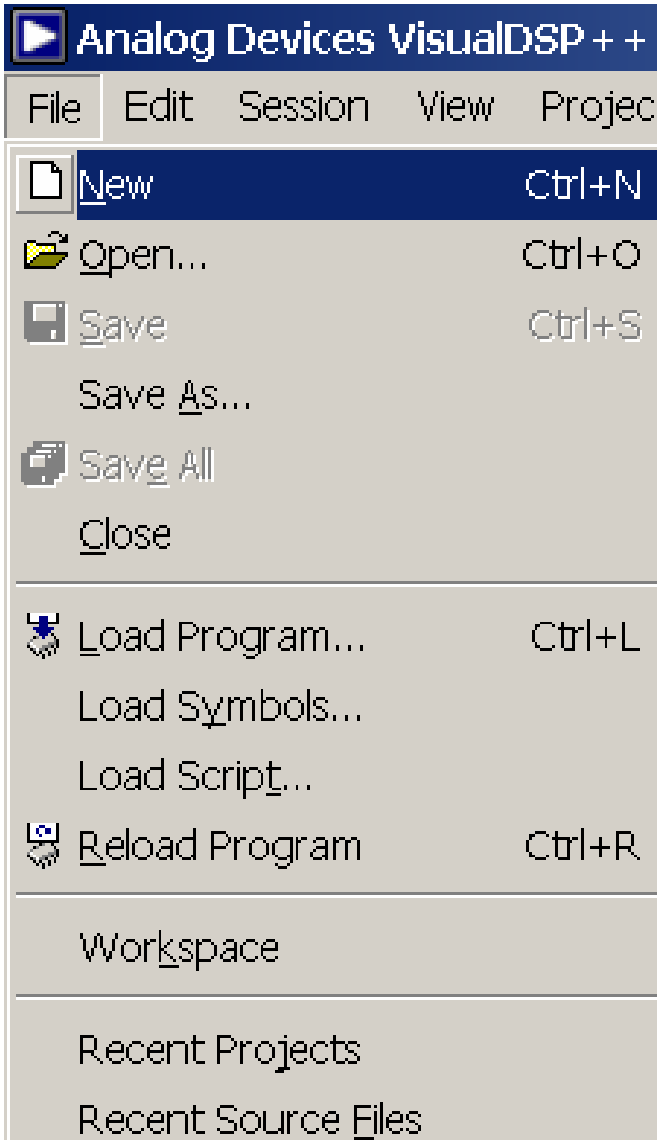


Link-File Architekturbeschreibung

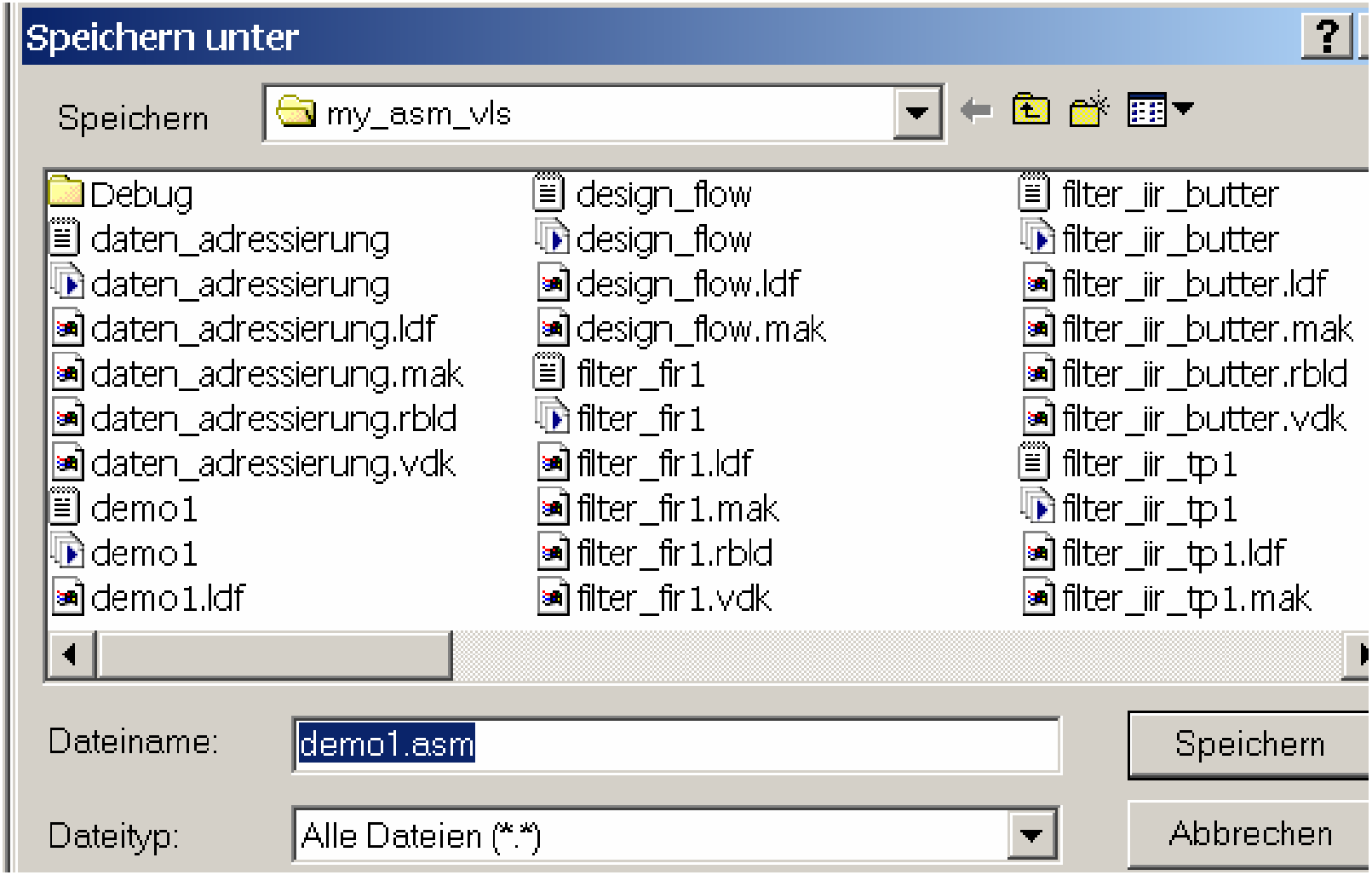
```
MEMORY
{
    seg_rth  { TYPE(PM RAM) START(0x00020000) END(0x000200ff) WIDTH(48) }
    seg_init { TYPE(PM RAM) START(0x00020100) END(0x0002010f) WIDTH(48) }
    seg_int_code { TYPE(PM RAM) START(0x00020110) END(0x00020197)
      WIDTH(48) }
    seg_pmco { TYPE(PM RAM) START(0x00020198) END(0x00023fff) WIDTH(48) }
    seg_pmda { TYPE(PM RAM) START(0x00026000) END(0x00027fff) WIDTH(32) }

     seg_dmda { TYPE(DM RAM) START(0x00028000) END(0x0002bfff) WIDTH(32)
     }
    seg_heap { TYPE(DM RAM) START(0x0002e000) END(0x0002efff) WIDTH(32) }
    seg_stak { TYPE(DM RAM) START(0x0002f000) END(0x0002ffff) WIDTH(32) }
}
```

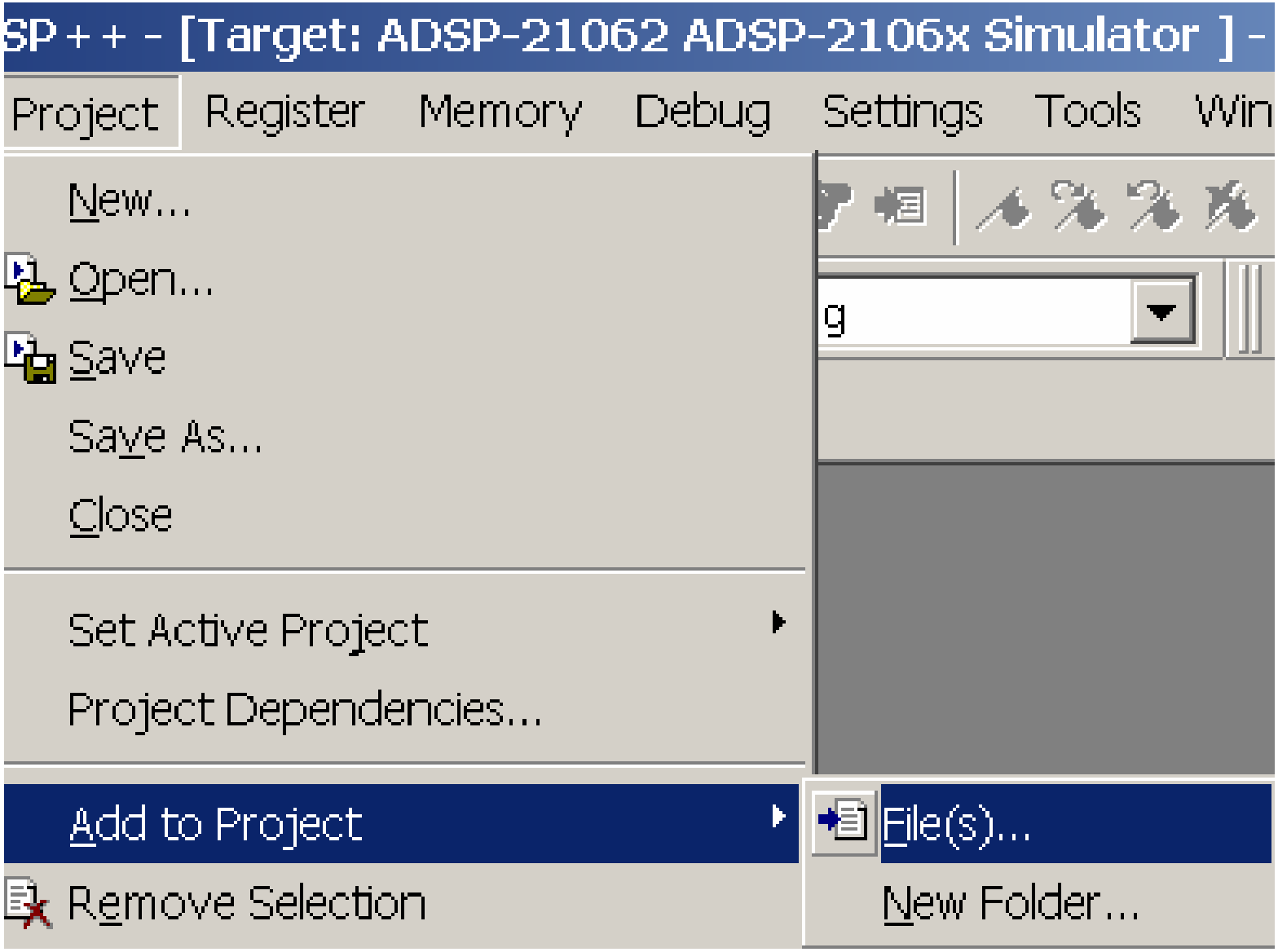
Assemblerfile erzeugen



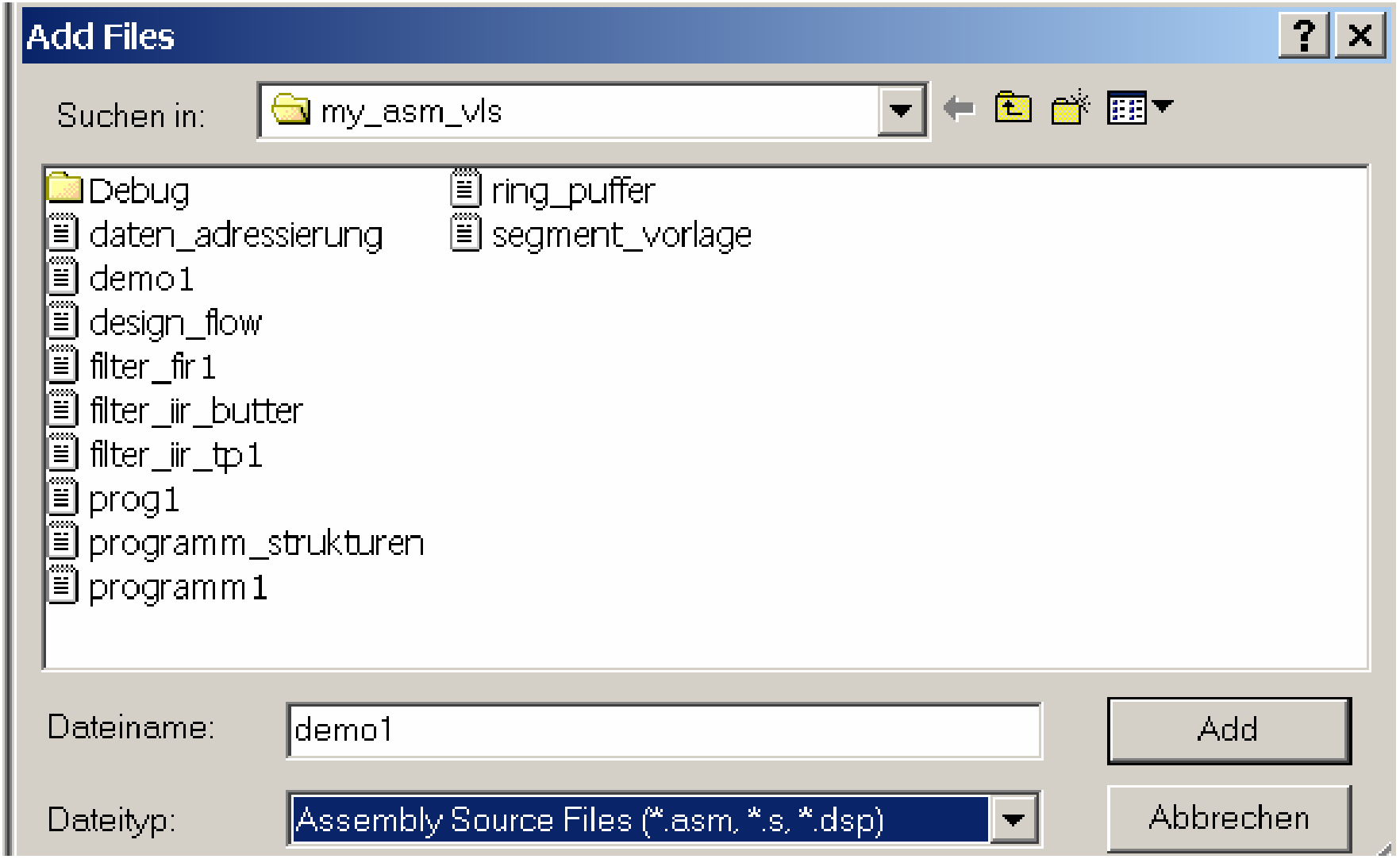
Assemblerfile erzeugen



Assemblerfile erzeugen



Assemblerfile erzeugen



Sections aus Link-File

/* 1. Projekt "demo1"
FUNKTION: Segmente aus Link-File übernehmen*/

.SECTION /dm seg_dmda;

.SECTION /pm seg_pmda;

.SECTION /pm seg_rth;

.SECTION /pm seg_pmco;

Template-File für Sections anlegen

```
/* Programmsegment für den Programmstart */
.SECTION        /pm seg_rth;

.SECTION        /pm seg_init;

/* Programmsegment für das Anwendungsprogramm */
.SECTION        /pm seg_pmco;

/* Datensegment Programmspeicher */
.SECTION        /pm       seg_pmda;

/* Datensegment Datenspeicher */
.SECTION        /dm       seg_dmda;
```

Variablendeklaration

```
/* Datensegment im Programmspeicher */
/* Beispiele für Variablendefinition */
.SECTION      /pm  seg_pmda;
.VAR       pmint1=1234;
.VAR       pmint2;
.VAR       pmfloat1=12.3456;
.VAR       pmfloat2=0.123e5;
.VAR       pmfixp1=0x40000000;
.VAR       pmfixp2=0xC0000000;
```

Variablendeklaration

```
/* Datensegment Datenspeicher */
/* Beispiele für Variablendefinition */
.SECTION /dm   seg_dmda;
.VAR dmint1=1234;
.VAR dmint2;
.VAR dmfloat2=0.123e5;
.VAR dmfixp1=0x40000000;
/* Beispiel für die Definition von Vektoren */
.VAR dmvek1[4];
.VAR dmvek2[]=11,22,33,44,55,66,77,88;
/* Datenvektor mit Hilfe eines Files initialisieren */
.VAR dmvek3[]="daten.dat";
```

Variablendeklaration

```
/* Variablendeklaration */
.SECTION /dm seg_dmda;
.VAR dminput[2]=0,1;
.VAR dmfloat=1.234e-1;
.VAR dmvek[6];

.SECTION /pm seg_pmda;
.VAR pmfloat = -1.234e2;
.VAR pmvek[6]=1,2,3,4,5,6;
```

Variablendeklaration

```
.VAR buf1=0x1234, buf2=0x5678, ...;
    // Define two initialized buffers
.VAR buf2=0x1234, 0x5678, ...;
    // Define two initialized words
.VAR samples[] = {10, 11, 12, 13, 14};
    // Declare and initialize an implicit-length buffer
    // since there are five values; this has the same
  effect
    // as samples[5].
    // Initialization values for implicit-size buffer must be
    // in curly brackets.
```

Variablendeklaration

```
.VAR Ins, Outs, Remains;
      // Declare three uninitialized variables
.VAR samples[100] = "inits.dat";
      // Declare a 100-location buffer and initialize it
      // with the contents of the inits.dat file;
.VAR taps=100;
      // Declare a variable and initialize the variable
      // to 100
.VAR twiddles[10] = "phase.dat";
      // Declare a 10-location buffer and load the buffer
      // with the contents of the phase.dat file
```

1. Programm

```
/* Section beginnt bei 0x20000
   PC steht nach RESET auf 0x20005
*/
.SECTION /pm seg_rth;
        nop;
        nop;
        nop;
        nop;
        nop;
        jump start;
```

Variablendeklaration

```
/* Register mit immediate Adressierung
   beschreiben */
.SECTION /pm seg_pmco;
start: R0=1;
       R1=2;
       F3=1.234e2;
       F4=-2.345e-1;
       idle;
```