

M16C Installation und Test

Ziel

Erfolgreiche Installation der M16C Entwicklungsumgebung und Sicherstellen der Funktionsfähigkeit mit einem einfachen Testprogramm.

Umfeld

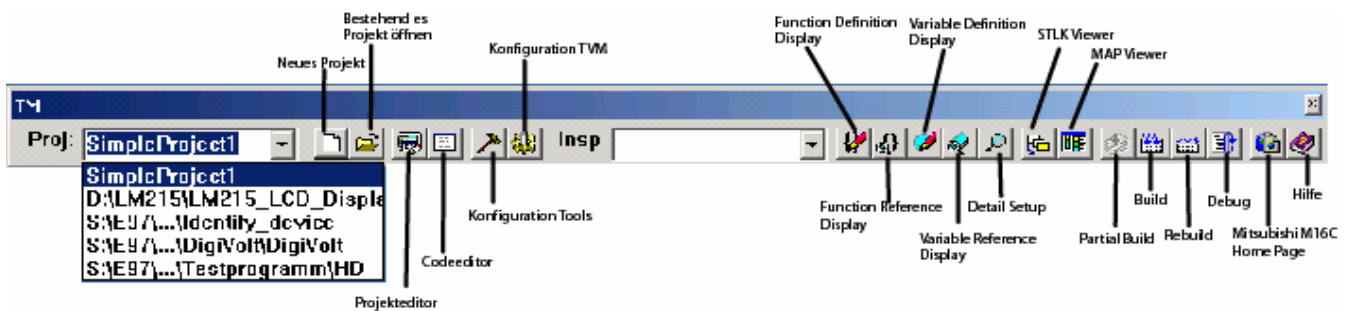
Die Embedded-System Programmierung wird M16C62-Evaluationsboard der HTA-BE durchgeführt. Es ist ähnlich dem Mitsubishi MSA0654, hat aber mehr Periferie zum Experimentieren.

Installation

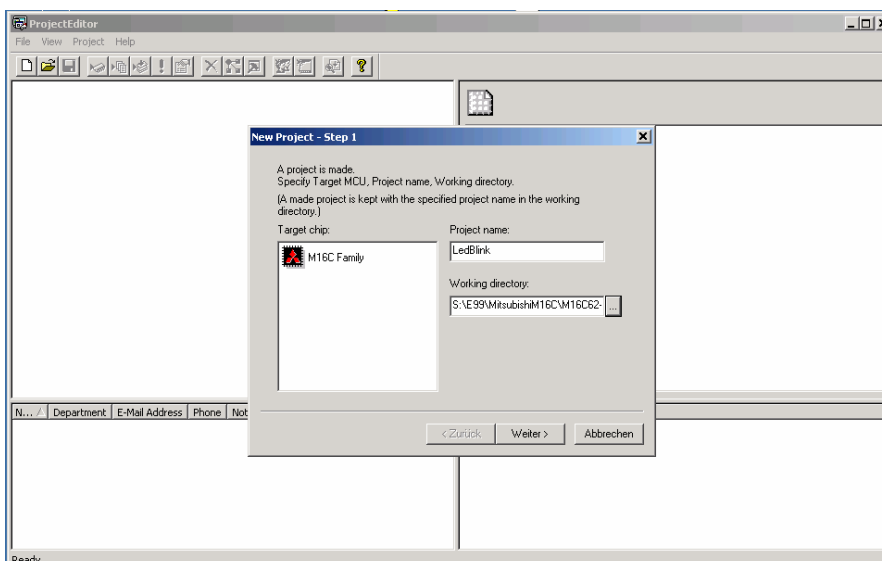
Ist gemäss der Einführung und Manual zu Evaluationsboard vorzunehmen. Auf den 8 Laborrechnern ist die Entwicklungssoftware bereits installiert.

Beispiel: LED Blinker

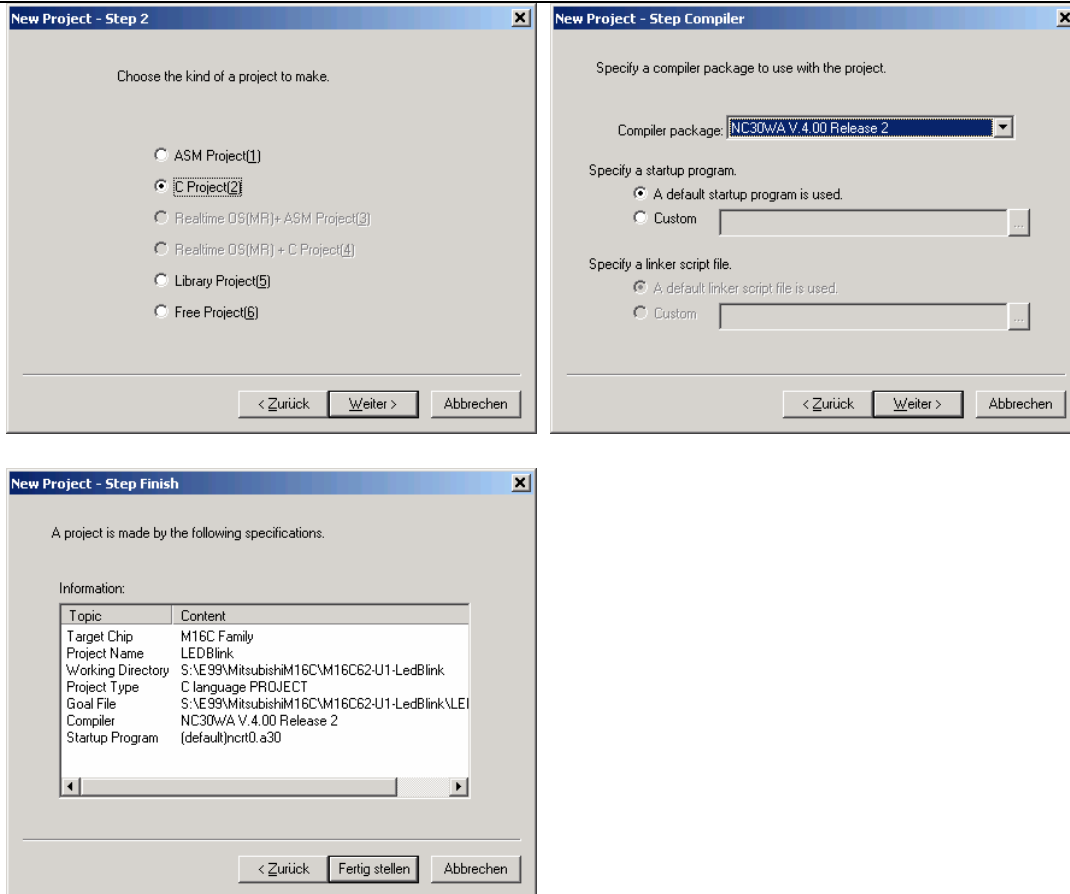
1. Projektmanager TM über Icon auf dem Desktop starten.
In der Listbox links erscheinen die jeweils letztgeöffneten Projekte:



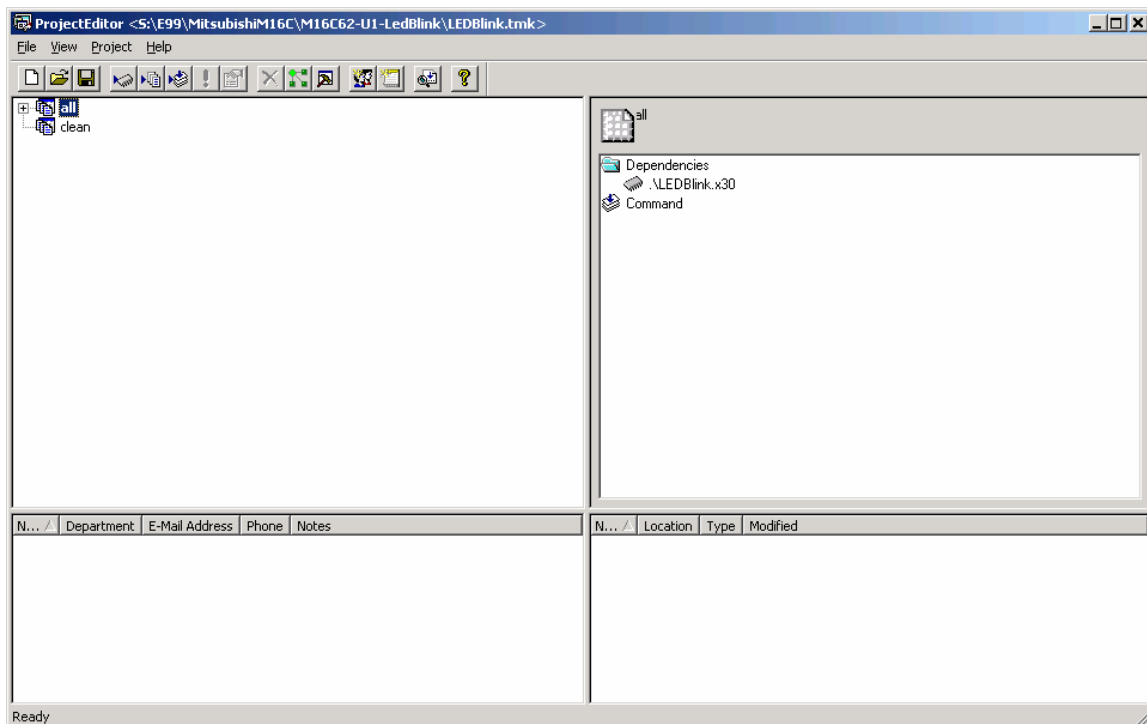
2. Neues Projekt definieren.
Dazu zuerst Verzeichnis für die neuen Files erstellen. Nachher Projektnamen und Arbeitsverzeichnis eingeben. Nicht vergessen auf das M16C Family Icon zu klicken:



Mit weiter die restlichen Konfigurationseinstellungen bestätigen:



Nach „Fertig stellen“ ist das Projekt konfiguriert:



Weiterführende Informationen zur Konfiguration und Arbeit entnehme man dem ausführlichen Handbuch zum TM-Projektmanager.

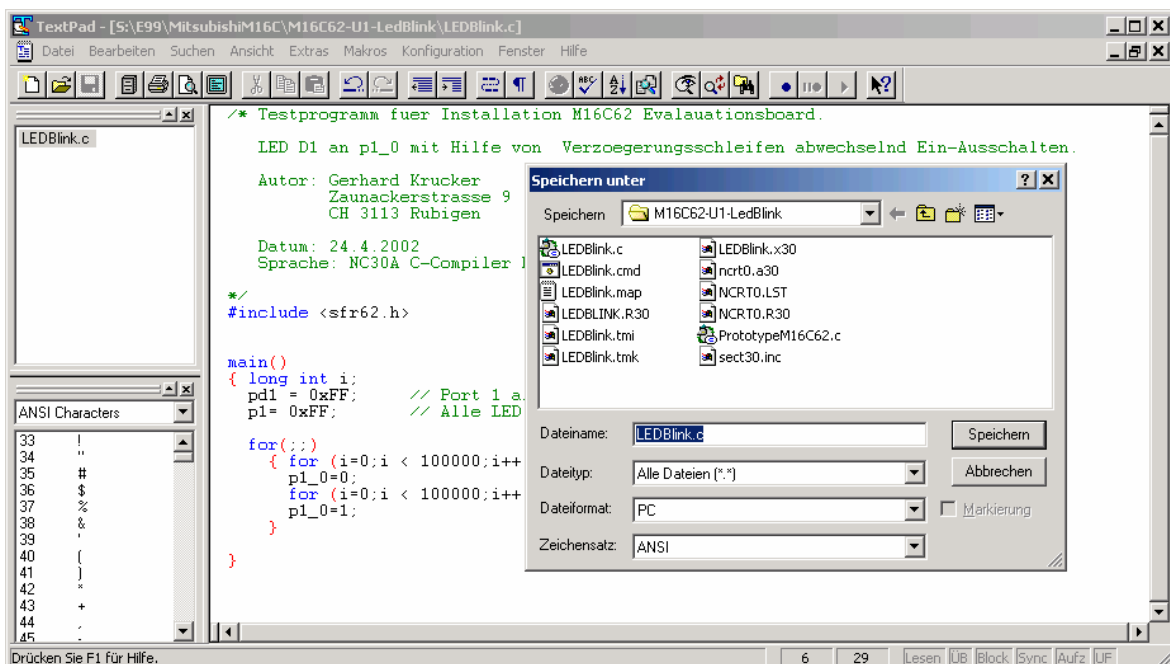
3. Include- und Startup-Files:
SECT30.INC und NCRT0.A30 werden automatisch in das Arbeitsverzeichnis zum Projekt kopiert. Ev. kontrollieren, dass für die Hardware richtige Version verwendet wird.
4. Eingeben des Sourcecodes.
Dazu Editor in der TM-Leiste starten und Code eingeben. Grundlage wäre PrototypeM16C62.C mit den Ergänzungen:

```
/* Testprogramm fuer Installation M16C62 Evaluationsboard.
LED D1 an p1_0 mit Hilfe von Verzoeigerungsschleifen abwechselnd Ein-Ausschalten.
Autor: Gerhard Krucker
Zaunackerstrasse 9
CH 3113 Rubigen
Datum: 24.4.2002
Sprache: NC30A C-Compiler Mitsubishi 4.00r2
*/
#include <sfr62.h>

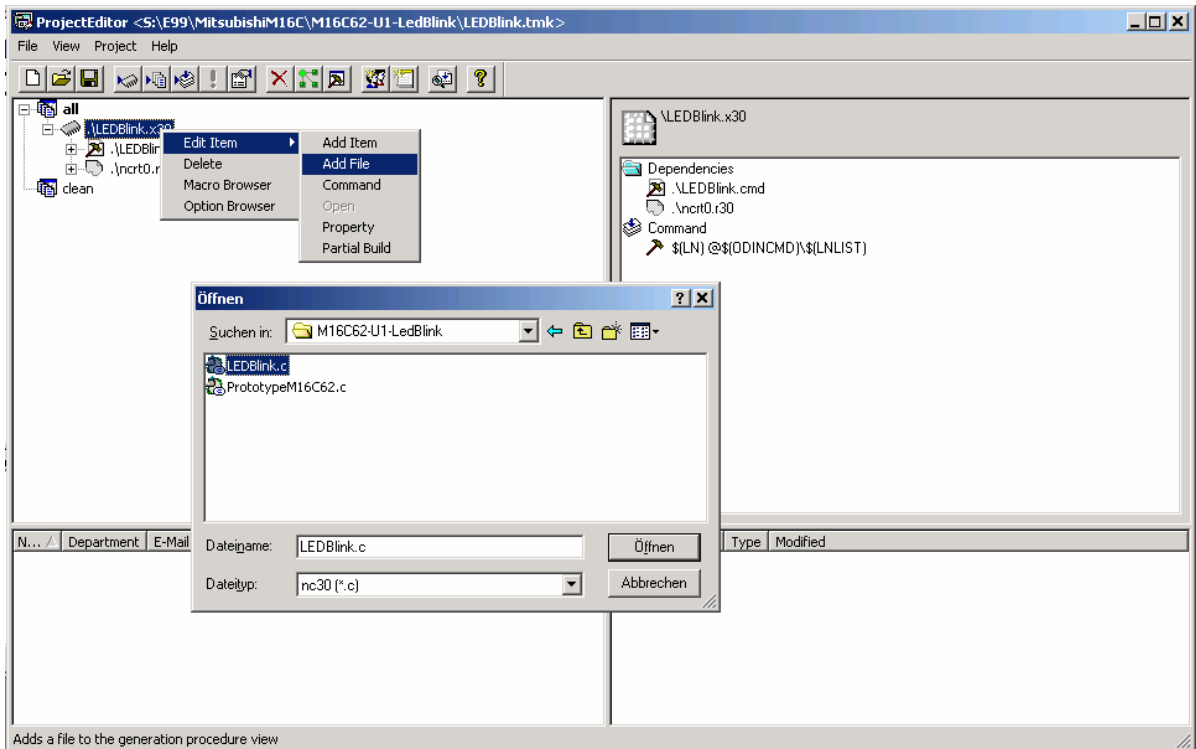
main()
{ long int i;
  pd1 = 0xFF; // Port 1 alles Ausgaenge
  p1 = 0xFF; // Alle LED D1..D8 aus

  for(;;)
  { for (i=0;i < 100000;i++); // Warten
    p1_0=0; // LED D1 ein
    for (i=0;i < 100000;i++); // Warten
    p1_0=1; // LED D1 aus
  }
}
```

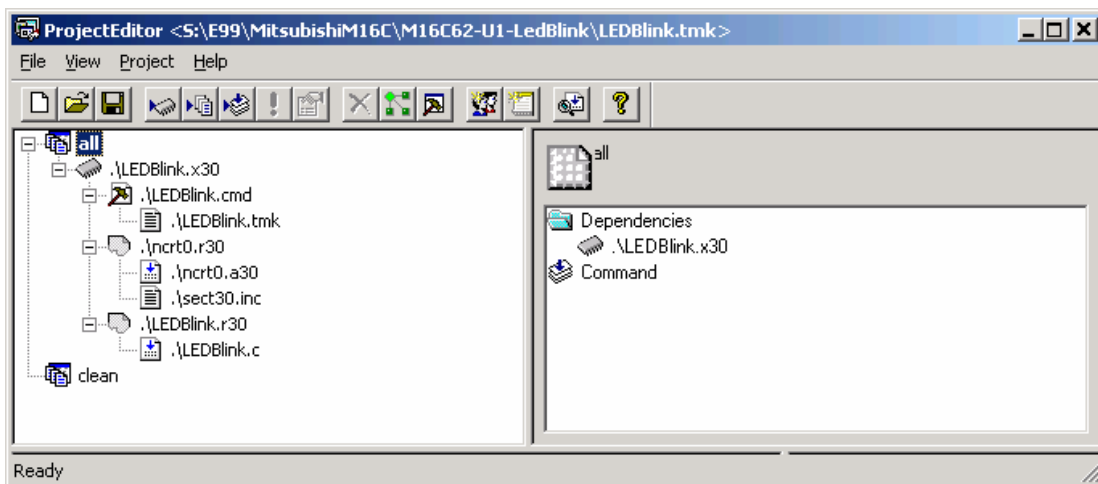
Module mit sinnvollem Namen und mit Extension .C speichern. Der Quelltext wird nach erstmaligem Speichern kontextsensitiv eingefärbt.



5. Quelltextmodul dem Projekt zufügen.
Dazu im Projektmanager über Add Item oder Add File das File zufügen:

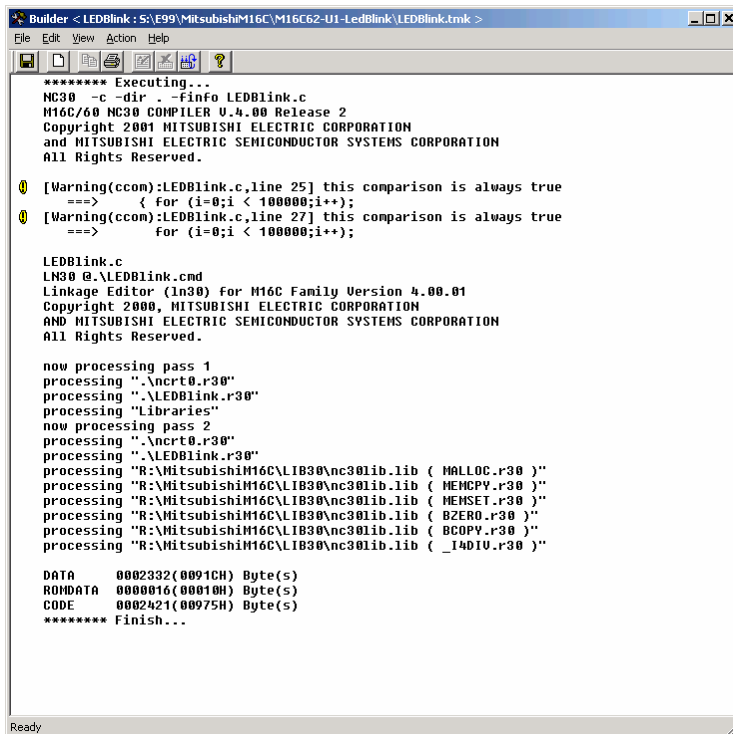


Kontrollieren, dass das zugefügte File auch wirklich im Projekt liegt: LEDBlink.tmk und NCRT0.r30 und LEDBlink.r30 müssen alle unterhalb LEDBlink.x30 liegen.



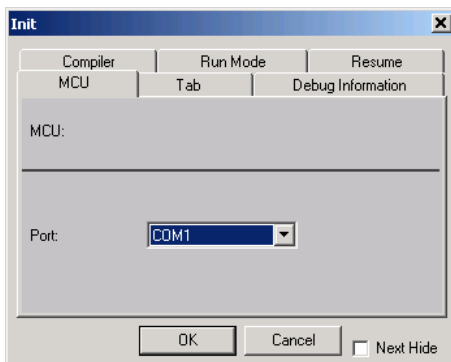
6. Kompilieren /Builden.

Start des Compilers über die Toolbar des TM Projektmanagers. Ein neues Fenster zeigt den Kompilationserfolg. Erstmalig wird auch der Startup-Code assembliert.

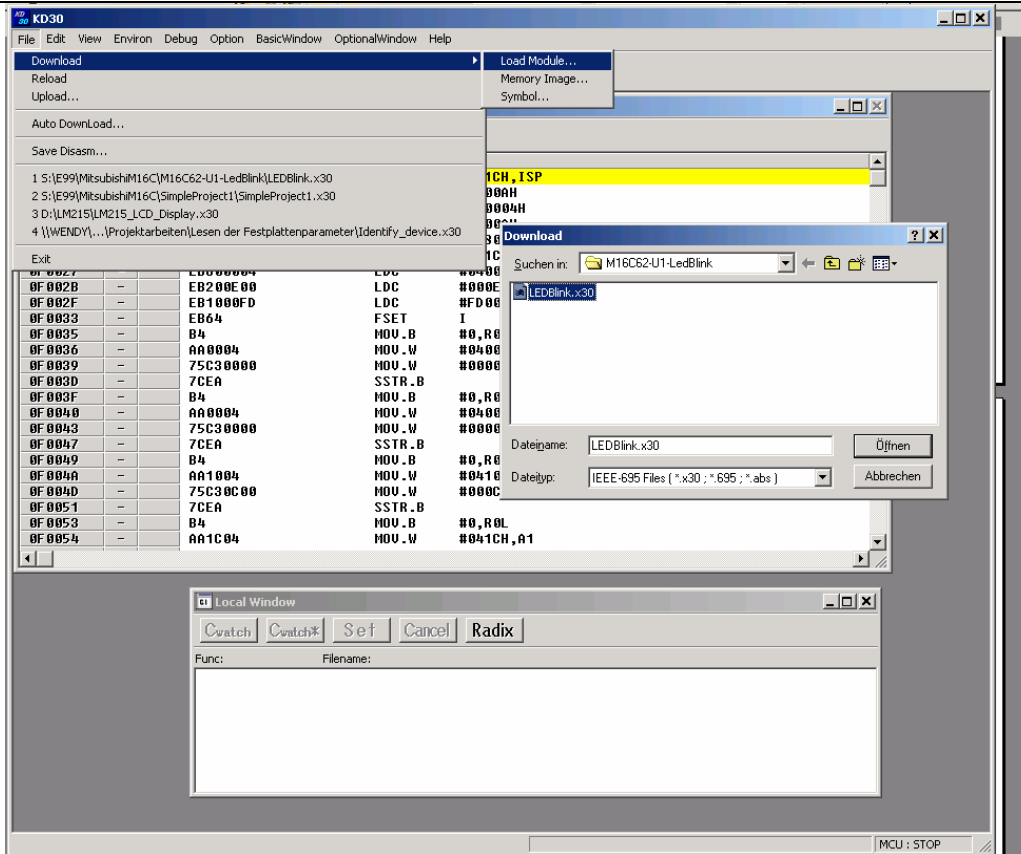


7. Test mit KD30 Debugger:

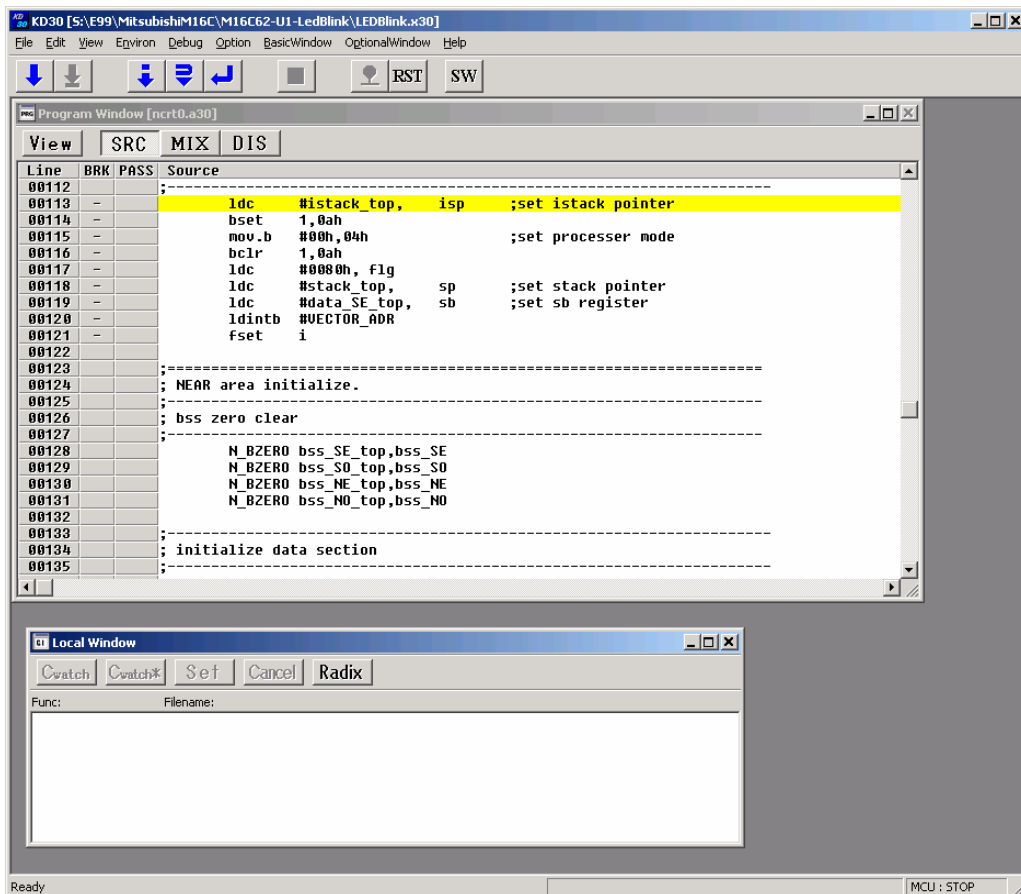
Nach erfolgreicher Kompilation kann der Debugger aktiviert werden. Das Evaluationsboard anschliessen und einschalten. Debugger aus der TM-Leiste starten. Es erscheint ein Konfigurationsdialog. Die Schnittstellendefinition kontrollieren und starten:



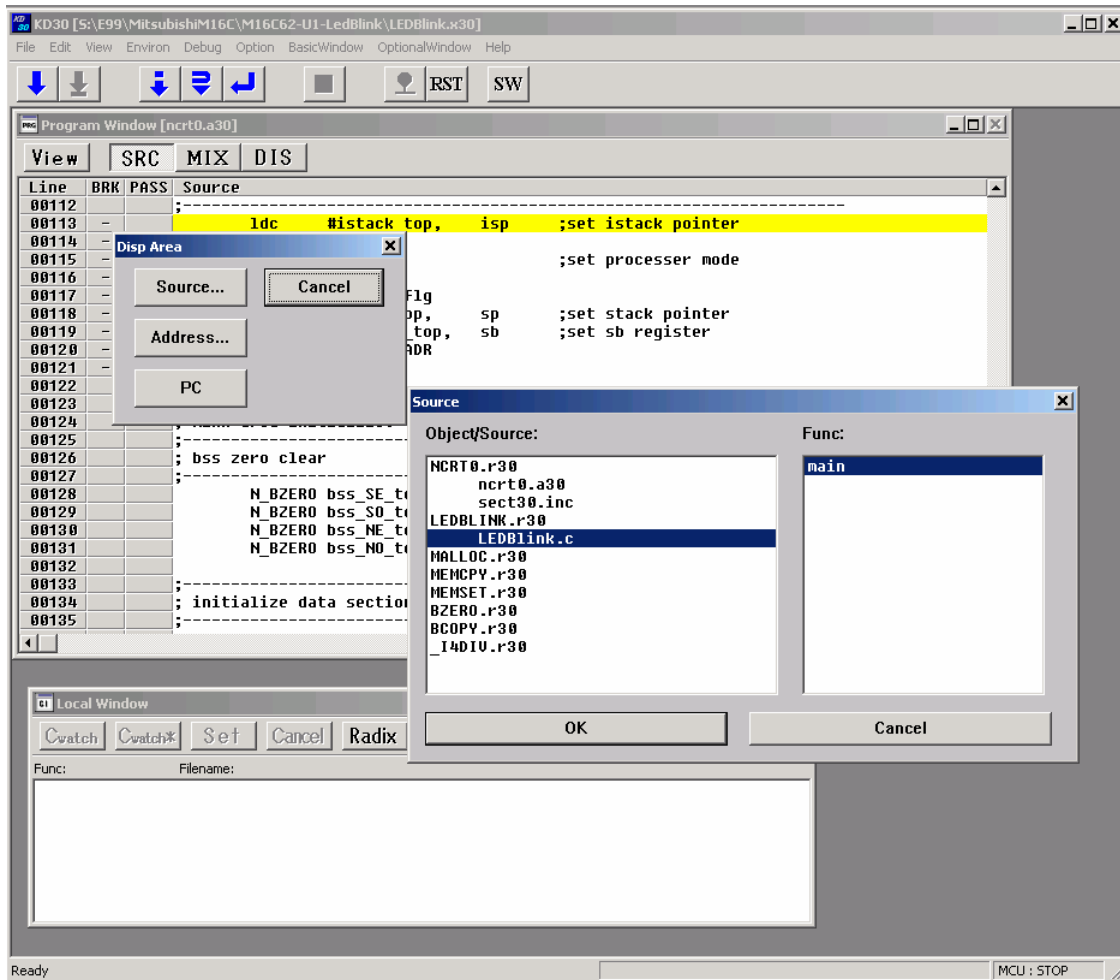
Das Debuggerfenster wird geöffnet. Es zeigt den jeweils letztgeladenen Code an. Den Knopf RST im Debugger-Menü 1x drücken und Codemodul mit Extension *.x30 laden:



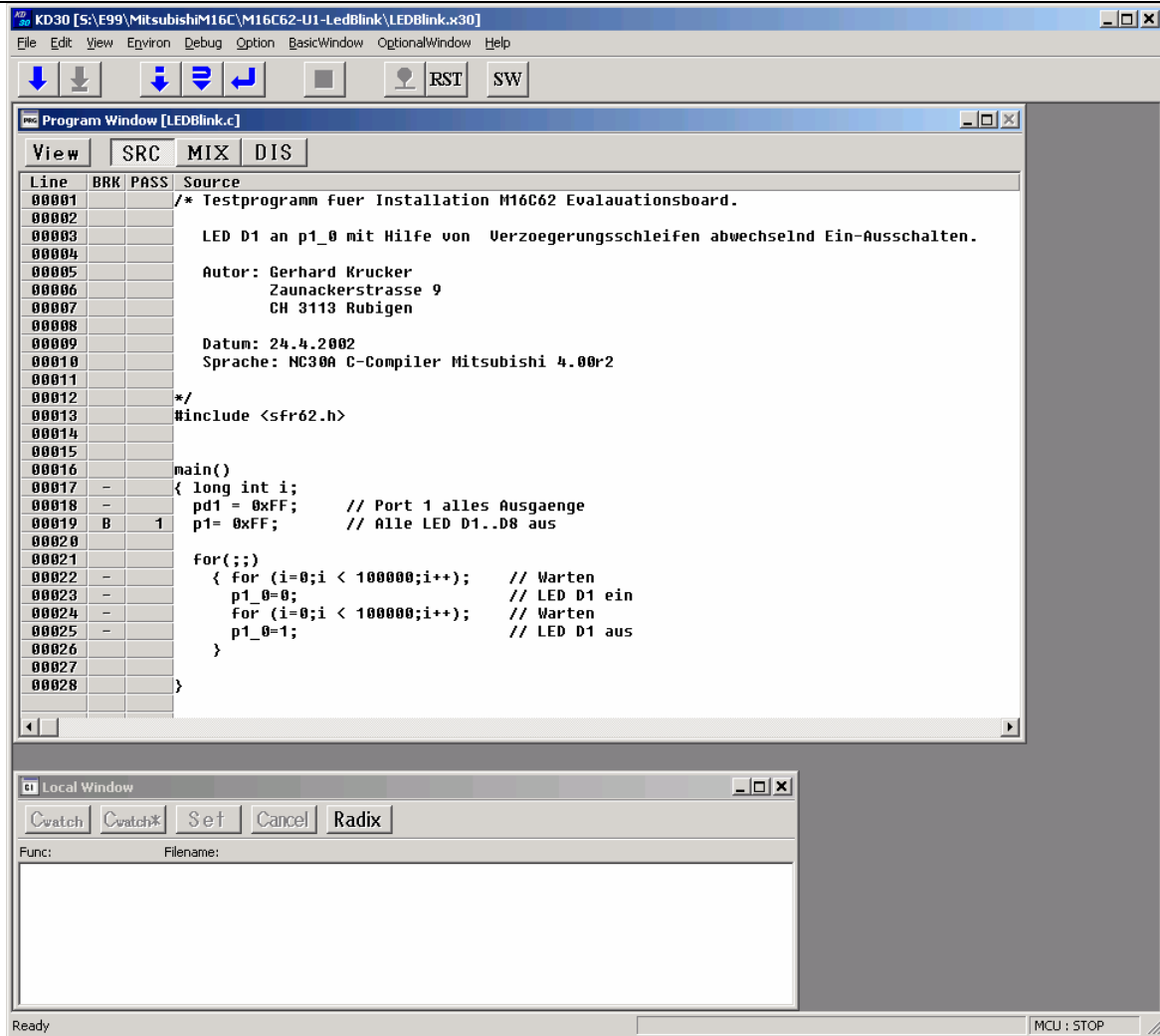
Nach dem Laden zeigt das Fenster den Startup-Code:



Das eigentliche Hauptprogramm kann über View/Source eingestellt werden:



Wir sehen nachher das eigentliche Hauptprogramm, wie auscodiert. In der Spalte BRK können über Mausclick maximal 2 Haltpunkte gesetzt werden.



8. Starten des Programmes. Über die Pfeiltaste links in der Toolbar des Debuggers

Aufgaben

1. Installation der SW und Aufbau der HW gemäss Manual. Speisung auf 12V Einstellen anschliessen. Kabel für Debug-Schnittstelle einstecken und an der seriellen Schnittstelle am PC anschliessen.
2. Erstellen eines neuen Projektes mit TM, das den LED-Blinker nach vorherigem Beispiel beinhaltet.
3. Austesten, die LED-Blinkfrequenz beträgt ca. 2Hz.
4. Erweitern des Programmes so, dass ein Lauflicht über die LED D1..D8 entsteht.

Lösung zum Laufflicht mit direkten C-Statements und _asm()-Code:

```

Line Address BRK PASS Objcode Label Source/Mnemonic
00006
00007 Autor: Gerhard Krucker
00008 Zaunackerstrasse 9
00009 CH 3113 Rubigen
00010
00011 Datum: 24.4.2002
00012 Sprache: NC30A C-Compiler Mitsubishi 4.00r2
00013
00014 */
00015 #include <sfr62.h>
00016
00017 main()
00018 { long int i;
00019 ENTER #05H
00020 unsigned char ledPattern;
00021
00022 pd1 = 0xFF; // Port 1 alles Ausgaenge
00023 MOV.B #FFH,03E3H
00024 p1= 0xFF; // Alle LED D1..D8 aus
00025 MOV.B #FFH,03E1H
00026
00027 ledPattern = 0xfe; // Muster 00000001
00028 MOV.B #FEH,-1H[FB]
00029
00030 for(;;)
00031 { for (i=0;i < 100000;i++); // Warten
00032 MOV.W #0H,-5H[FB]
00033 MOV.W #0H,-3H[FB]
00034 CMP.W #1H,-3H[FB]
00035 JGT F0156H
00036 JLT F014EH
00037 CMP.W #86A0H,-5H[FB]
00038 JGEU F0156H
00039 ADD.W #1H,-5H[FB]
00040 ADCF.W -3H[FB]
00041 JNP.B F013EH
00042 p1=ledPattern; // Ausgabe auf die LED D0..D7
00043 MOV.B -1H[FB],03E1H
00044 // Loesung mit Assembler Statements
00045 // Referenz: _asm Zugriff: MC Compiler Users Manual S. B.2.7
00046 // Prozessorbefehle: 6020ESM, S.131
00047 _asm ("MOV.B $$[FB],R0L",ledPattern); // Variablenwert in R0L laden
00048 MOV.B -1H[FB],R0L
00049 _asm ("ROT.L #1,R0L"); // Rotieren nach links um eine Stelle
00050 ROT.L #1H,R0L
00051 _asm ("MOV.B R0L,$$[FB]",ledPattern); // Rotierten Wert in Variable zurueck speichern
00052 MOV.B R0L,-1H[FB]
00053 JNP.B F0138H
00054
00055 // Loesung mit direkten C-Statements
00056 // ledPattern=ledPattern << 1 | 1; // Muster weiterschieben
00057 // if (ledPattern==0xFF) ledPattern = 0xFE; // 1x Durchlaufen Muster 00000001
00058 }
    
```

Ablauf Laufflicht

Ports initialisieren
Muster 000'0001 laden
Do forever
warten
Muster auf Port1 ausgeben
Muster um eine Stelle linksrotieren