Modul 1
C-Programmierung der Familie 8051
Einführung

Version 1.0 Dipl. Ing. Dr. Josef Humer

INHALTSVERZEICHNIS
MODUL 1
C-Programmierung der Familie 8051
Einführung

1 Mikrocontroller der Familie 8051

Als Mikrocontroller bezeichnet man Integrierte Schaltungen, die auf einem einzigen Chip ein vollständiges Mikrocomputersystem enthalten.

Mikrocontroller werden verwendet, um Regelungen und Steuerungen aufzubauen, um mechanische Aktoren zu bedienen, Sensoren einzulesen und Echtzeitberechnungen durchzuführen.

Aufbauend auf denn 8051, der von Intel entwickelt wurde, wurden zahlreiche Derivate von verschiedenen Firmen entwickelt. Als Vertreter solcher Derivate seien der Microcontroller 83C552 von Philips und der 80C517 von Siemens genannt. Beide sind leistungsfähige Erweiterungen des 8051. Sie enthalten den kompletten 8051, besitzen eine Menge von zusätzlichen Peripheriefunktionen und sind völlig softwarekompatibel.

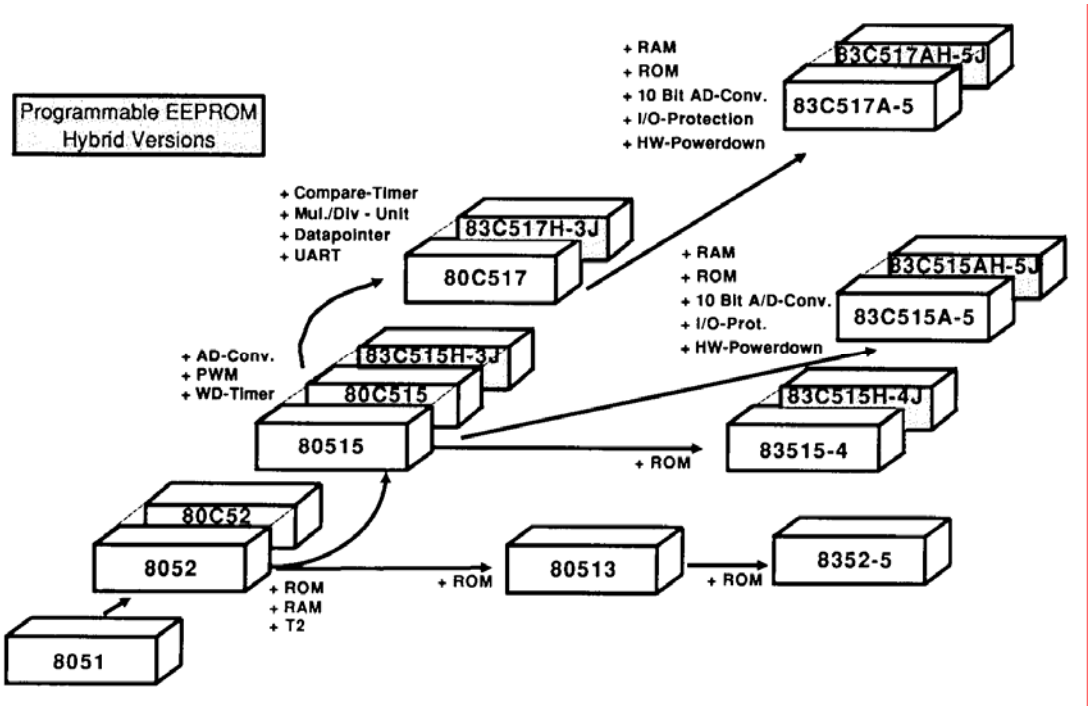


Abb. 1 Überblick der 8051-Familie von Siemens

1.1 Überblick über den 80C517/80C517A

- 8 Kbyte On-Chip-ROM beim 80C517
- 32 Kbyte On-Chip-ROM beim 83C517A-5
- ROM-lose Varianten sind auch erhältlich (80C537 und 80C517A)
- Aufwärtskompatibel zum 8051 und 80515
- 256 Byte On-Chip-RAM
- Zusätzliche 2 Kbyte On-Chip-RAM beim 80C517A
- Einzelbitverarbeitung
- Versionen mit verschiedenen Taktgeschwindigkeiten erhältlich
- Externe Erweiterbarkeit von Programm- und Datenspeicher (je 64 Kbyte)
- On-Chip-A/D-Wandler (8 bit- Version beim 80C517, 10 bit- Version beim 80C517A) 12 Eingänge, wahlweise interner oder externer Start.
- Zwei 16-bit Timer, voll kompatibel zum 80(C)51.
- Universelle Compare/Capture-Einheit mit eigenen 16-bit Timern; unterschiedlich konfigurierbare 16-bit-Compare- und Capture-Register für zeitliche Auflösungen.
- Arithmetik-Einheit für 16-bit-Arithmetik (Multiplikation, Division, Schiebeoperationen)
- Integrierte Systemüberwachung: Programmierbarer 16-bit-Watchdog-Timer, Oszillator-Watchdog.
- Neun Eingabe-Ausgabe-Ports, unterschiedlich konfigurierbar
- Zwei unabhängige serielle Schnittstellen mit eigener Baudratenerzeugung
- Interrupt-System mit 14 (beim 80C517) bzw. 17 (beim 80C517A) Interrupt-Vektoren und 4 Prioritätsebenen.
- Betriebsarten zur Reduzierung der Stromaufnahme:
  - Slow-Down-Modus
  - Idle- Modus
  - Power-Down-Modus
- Gefertigt in CMOS-Technologie
- Unterschiedliche Gehäusevarianten erhältlich: PLCC84, PQFP 100

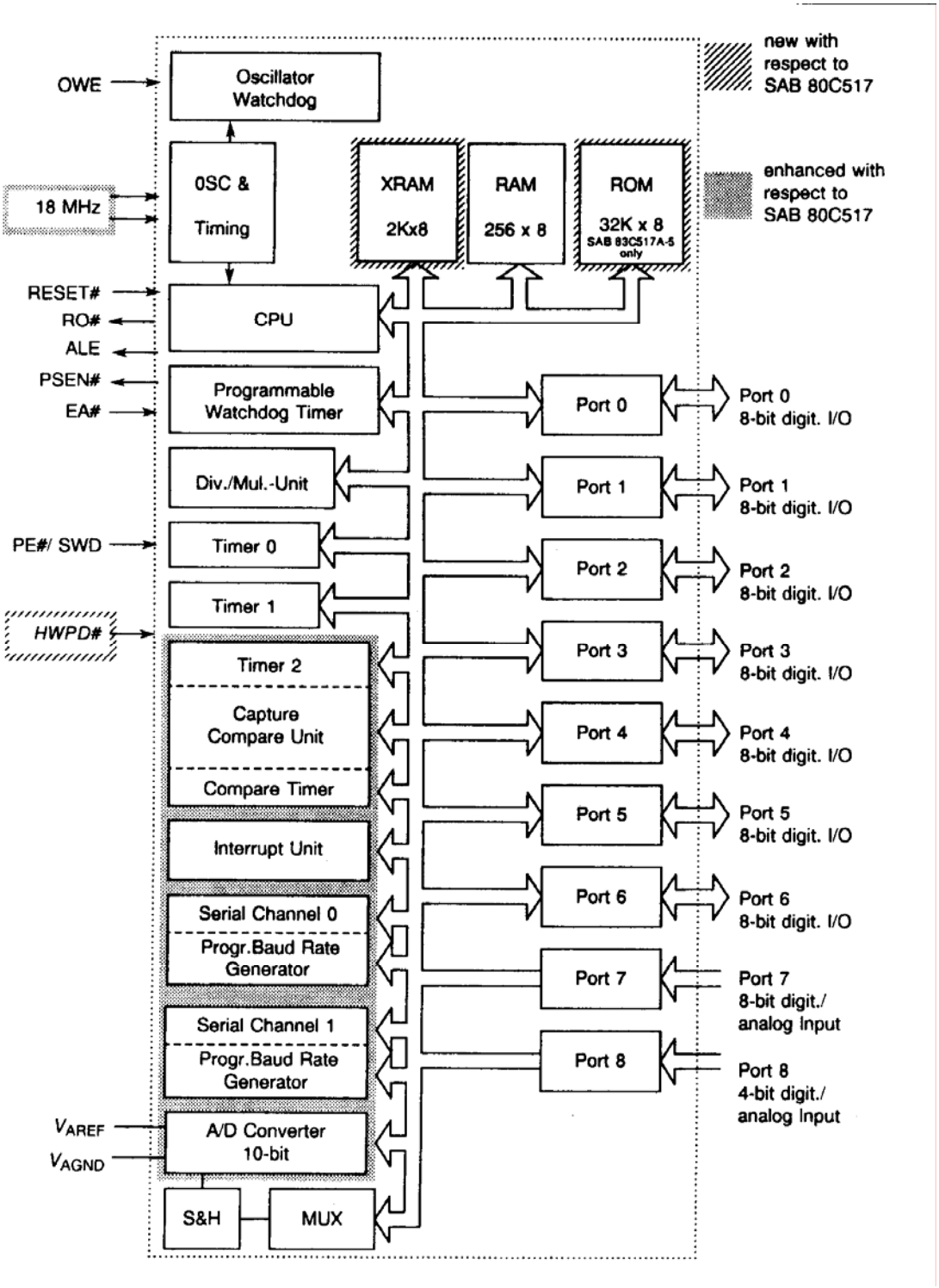


Abb. 2: Funktionsdiagramm des 80C517/80C517A

1.2 Speicherorganisation

1.2.1 Allgemeines

Der Speicherraum ist logisch und physikalisch in vier verschiedene Bereiche unterteilt.

- bis zu 64 Kbyte Programmspeicher adressierbar.
- bis zu 64 Kbyte externer Datenspeicher adressierbar.
- 256 Byte interner Datenspeicher
- ein 128 Byte großer Bereich für Special-Function-Register (SFR)

1.2.2 Programmspeicheraufteilung

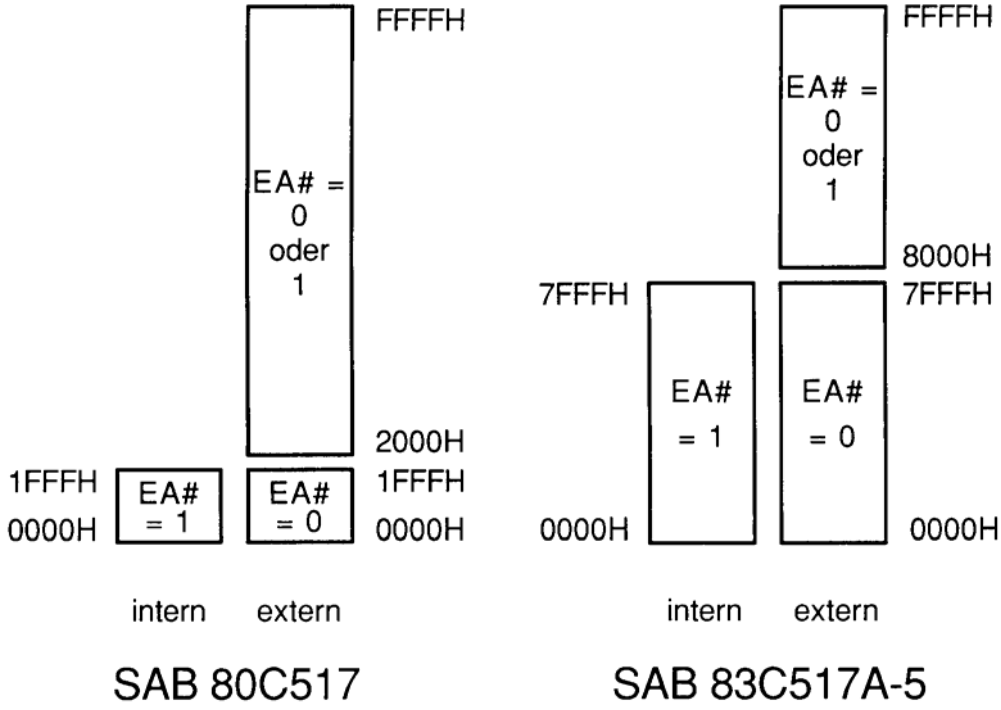


Abb. 3: Programmspeicheraufteilung

Im Bereich von 03H bis 93H liegen beim 80C517 des Programmspeichers die Einsprungadressen für die Interrupt-Routinen. Wenn einzelne Interrupts nicht benutzt werden, können aber diese Bereiche selbstverständlich für normalen Code verwendet werden.

1.2.3 Interner Datenspeicher

Speicherbereich	Adresse	Adressierungsart
Untere 128 Byte RAM	00H bis 7FH	direkt/indirekt
Obere 128 Byte RAM	80H bis 0FFH	indirekt
SFR	80H bis 0FFH	direkt

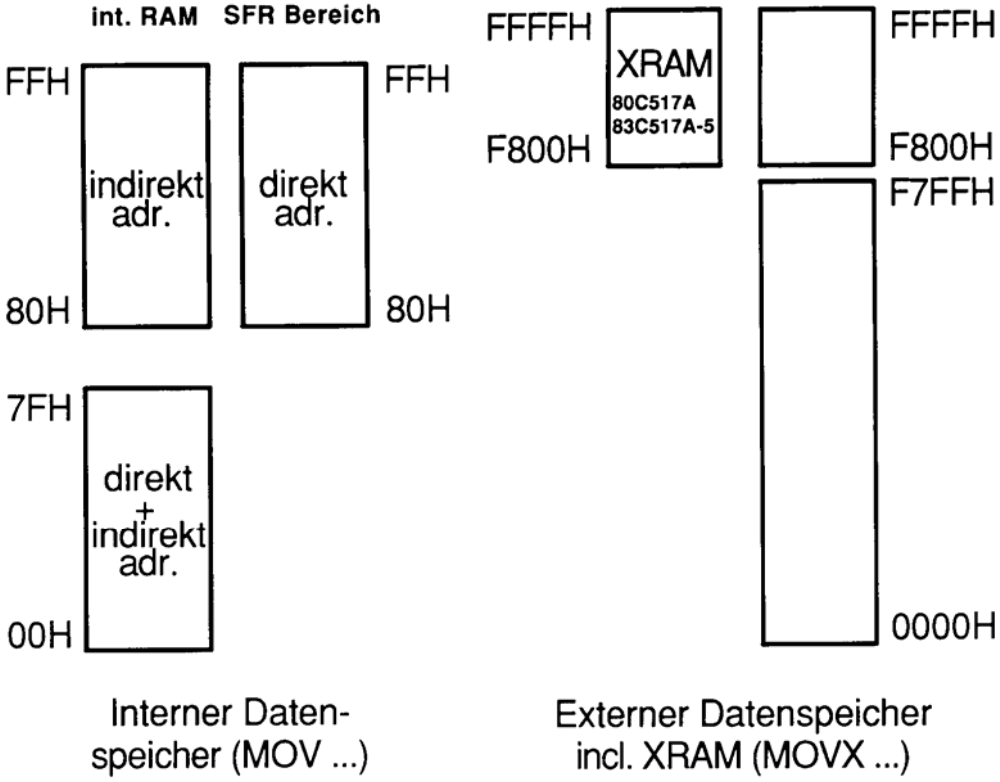


Abb. 4: Aufteilung des internen Datenspeichers

2 Die Code-Data-Struktur des 8051

Alle Microcontroller der Familie 8051 verfügen über getrennte Adressierbereiche für den Programmspeicher und den Datenspeicher. Der externe Programmspeicher und der externe Datenspeicher kann bis zu 64 kByte umfassen.
Der Zugriff auf den exteren Programmspeicher wird über das Signal PSEN gesteuert. Für den Zugriff auf den Datenspeicher stehen die Signale RD und WR zur Verfügung. Die Struktur eines solchen Speichersystems wird Harvard Struktur genannt.

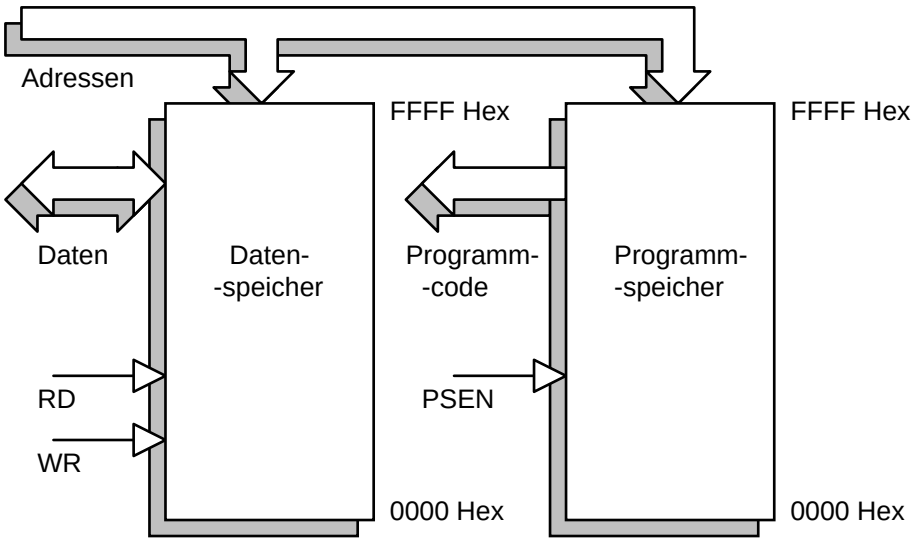


Abb. 5: Harvard Struktur

Die Harvard Struktur wird während des normalen Betriebs des Mikrocontrollers verwendet.

Zur Programmentwicklung wird jedoch eine andere Speicherstruktur verwendet. Sie wird Neumann Struktur genannt. Bei der Neumann Struktur liegt der Datenspeicher und der Programmspeicher in einem gemeinsamen Adressbereich. Dazu muß das Signal RD, welches aus dem RAM liest, und das Signal PSEN, welches aus dem ROM liest, verknüpft werden. Die UND Verknüpfung der beiden Signale ergibt ein Lesesignal für den gesamten Speicher.
Nötig ist diese Zusammenschaltung aus folgendem Grund: Während der Programmentwicklung läuft im ROM ein Monitorprogramm. Dieses kommuniziert über die serielle Schnittstelle mit dem PC. Auf dem PC wird das Anwenderprogramm geschrieben, compiliert, gelinkt und in ein INTEL Hex Format gebracht. Das Anwenderprogramm wird mit Hilfe des Monitors in das RAM des 8051 geladen und

vom Monitorprogramm gestartet. Das Programm kann bequem getestet und auf Fehlfunktionen überprüft werden.

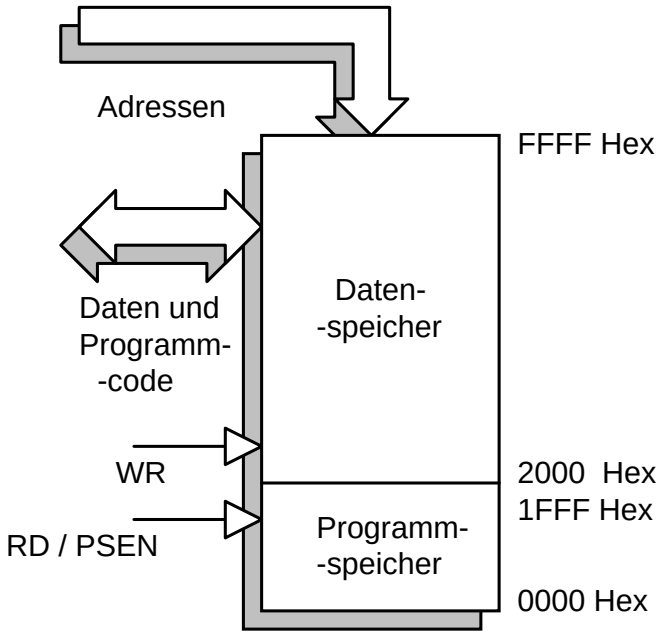


Abb. 6: Neumannstruktur

Erst wenn das Programm einwandfrei funktioniert, wird es in ein EPROM gebrannt. Dann wird wieder die Harvard Stuktur mit getrenntem Code- und Datenbereich verwendet.

3 Das Monitorprogramm MON51

3.1 Allgemeines

Das Monitorprogramm MONITOR-51 ist für den Betrieb auf Mikrocontoller-baugruppen mit einem 8051 Prozessor zugeschnitten. Der MONITOR-51 führt die Kommunikation mit dem Benutzer über die serielle Schnittstelle durch.

Folgende Kommandos stehen dem Benützer zur Verfügung:

- Anzeigen der verschiedenen Speicherbereiche in Hexadezimal und ASCII Darstellung.
- Interaktives Ändern der Speicherinhalte
- Anzeigen und Ändern der Registerinhalte und der "Special Function Register".
- Initialisieren der verschiedenen Speicherbereiche mit einem konstanten Wert.
- Anzeigen des Code Bereichs in 8051 Mnemonics
- Inline Assembler
- Real Time Go mit bis zu 10 festen und einem temporären Breakpoint
- Verwalten von bis zu 10 festen Breakpoints für Real Time Go
- Einzelschritt-Abarbeiten von Programmen mit der Möglichkeit, Unterprogramme als einen Einzelschritt auszuführen.
- Down-load und Up-load für Intel Hex Dateien.
- Kommando Menü (= Help Funktion)

3.2 Die wichtigsten Kommandos in Kurzform

Das Monitorprogramm wird durch die Eingabe C:> MON51 gestartet. Nach dem Start meldet sich das Programm folgendermaßen:

```
C:\> MON51
MS-DOS MON51 V1.02
(C) Franklin Software Inc./KEIL ELEKTRONIK GmbH  1991

INSTALLED FOR PC/XT/AT (COM LINE 1) USING HARDWARE INTERRUPT SERVICE
BAUDRATE: 9600 (DEFAULT)
*** MONITOR MODE ***
#
```

Kommt die Meldung ****** MONITOR MODE ****** und in der nächsten Zeile das Zeichen ***#,*** so weiß man, das die Verbindung erfolgreich aufgebaut wurde und das Monitorprogramm und der 8051 betriebsbereit sind.

Kann keine Verbindung zum 8051 aufgebaut werden, meldet sich das Monitorprogramm mit folgender Meldung:

```
C:\>mon51
MS-DOS MON51 V1.02
(C) Franklin Software Inc./KEIL ELEKTRONIK GmbH  1991

INSTALLED FOR PC/XT/AT (COM LINE 1) USING HARDWARE INTERRUPT SERVICE
BAUDRATE: 9600 (DEFAULT)
*** TERMINAL MODE ***
```

Die Meldung ****** TERMINAL MODE ****** zeigt an, das keine Verbindung zum 8051 System aufgebaut werden konnte. Außerdem wird in der nächsten Zeile das Zeichen # nicht angezeigt. In dieser Betriebsart dient das Monitorprogramm nur zur Kommunikation über die serielle Schnittstelle, es können Daten ausgeschrieben und empfangen werden. Nicht möglich ist das Senden von Anwenderprogrammen sowie das Lesen und Verändern der Speicherbereiche des 8051.

Das Programm wird durch Drücken der Taste *F1* beendet. Nach dem Drücken der Taste ***F1*** erscheint am oberen Bildschirmrand die Frage:
EXIT MONPC (y or [n]) ?
Durch Bestätigung mit der Taste *y* wird das Programm verlassen und man kehrt in die DOS Umgebung zurück.

Um ein Programm (z.B. das Programm test.hex) zu laden, wird folgender Befehl benötigt:
 # load test.hex <cr>
.....
Die Punkte zeigen an, daß das Programm in den Speicher des 8051 geladen wird. Anschließend meldet sich das Programm wieder mit dem Zeichen #.

Nachdem das Programm geladen wurde, wird es gestartet. Dazu wird der Befehl *g* (für *go*) und der Angabe der Startadresse gestartet:
 # g8000 <cr>

Das Programm läuft dann bis zu seinem Ende durch und erst dann meldet sich der Monitor wieder mit der Meldung *#*.

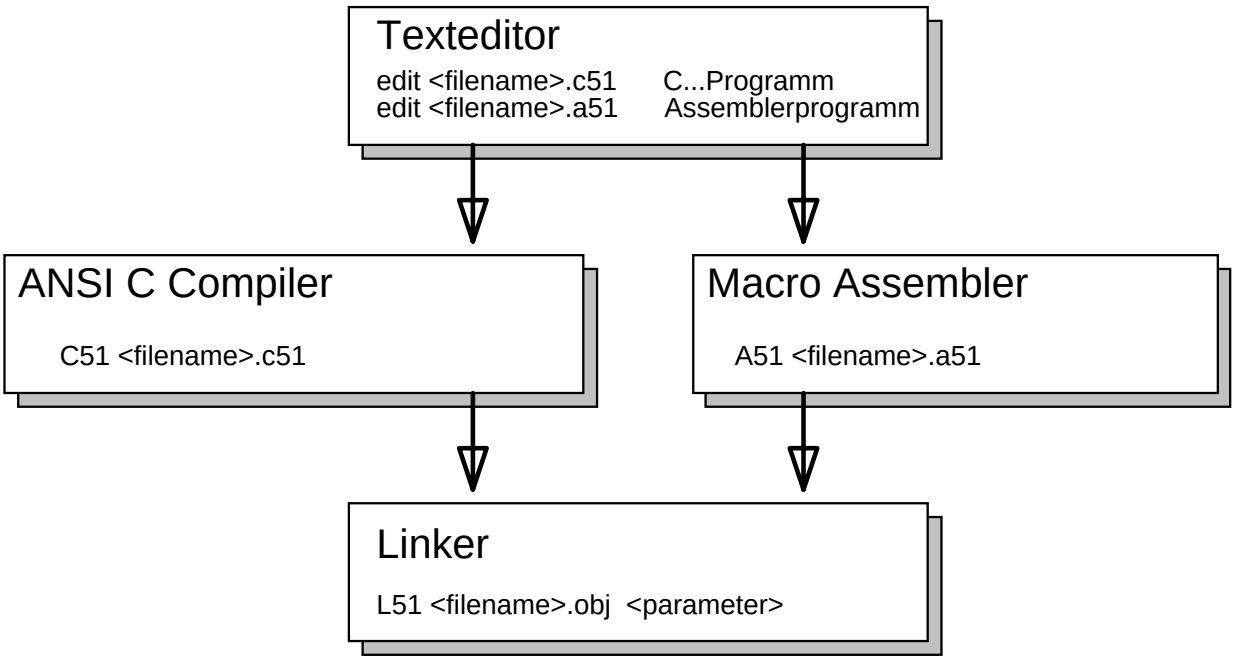
🕮 Übung 1:

Starten Sie das Monitorprogramm mit dem Befehl *MON51* .
Laden Sie das Programm TEST.HEX in den Speicher des 80517.(*# load TEST.HEX*)
Starten sie das Programm. (*# g8000*)

4 C-Programme für den 8051

4.1 Allgemeines

Zur Programmierung der Mikrocontroller der Familie 8051 steht ein C-Compiler der Firma Keil zur Verfügung.
Der Ablauf bei der Erstellung eines C-Programms sieht folgendermaßen aus:



Mit dem Texteditor wird der C Source Code eingegeben und mit dem C Compiler übersetzt. Dieser liefert einen absoluten Oject Modul für das Anlaufen des Programms an der Adresse 0 und einige verschiebbare (relocatable) Programm Module, denen erst beim Link Vorgang absolute Adressen zugewiesen werden. Beim Link Vorgang werden dann auch die mit dem C Compiler mitgelieferten Library Module, die von den Funktionsaufrufen angesprochen werden, in den Maschinencode miteingebunden.
Der Linker liefert den lauffähigen Code, der für die Datenübertragung mittels des Monitorprogramms noch mit einem Object Hex Konvertierungsprogramm in ASCII umgewandelt wird.

4.2 Zugriff auf Special Function Register

Das erste Programm soll die acht Leuchtdioden auf der Platine aktivieren Die Leuchtdioden sind über einen invertierenden Treiber (74HC540) an den Port 4 des 80517(A) angeschlossen. Das Programm soll folgende Aufgaben erfüllen:
Zuerst soll der Port 4 rückgesetzt werden (alle Leitungen an Port 4 sind auf L, dies Bedeutet, alle 8 Leuchtdioden sind dunkel). Anschließend soll ein bestimmtes Bitmuster am Port 4 ausgegeben werden. Das Bitmuster soll abwechselnd 01010101 und 10101010 lauten, dies entspricht den beiden HexZahlen **0x55** und **0xAA.**
Die im Hauptprogramm verwendete Bezeichnung P4 bezeichnet den Port 4 des Microcontrollers. Die zugehörige Datendefinition ist in der Datei **REG517.H** (**REG517A.H**) abgespeichert und lautet ***sfr P1 = 0xE8*** . Die Abkürzung ***sfr*** steht für ***s****pecial* ***f****unction* ***r****egister* und bezeichnet einen Datenypen zum Zugriff auf ein special function register des 8051. Die Zeile ***sfr P1 = 0xE8*** sagt aus , daß das special function register, über welches man auf Port 4 zugreifen kann, im Programm **P4** heißt und intern im Speicher des 8051 die Adresse E8**Hex (0xE8)** hat.

Der C Source Code wird mit dem Texteditor von MS DOS eingegeben. Dieser wird mit dem Befehl ***C:***\>***edit*** gestartet.

```
/*************************************************************/
/*                          PORT.C                           */
/*            Erstes Testprogramm für MC - Board             */
/*************************************************************/

/****   Steueranweisungen an den Compiler       ****/

/****   Angabe der Include Dateien              ****/
#include <reg517.h>

/****   Prototypen der Funktionen               ****/
void warte(unsigned int msec);

/****   Funktionen                              ****/
void warte(unsigned int msec)

{ data char i;
  for (msec; msec!=0; msec--) for (i=0; i<83; i++);
}

/*************************************************/
/****   Hauptprogramm                           ****/
/*************************************************/

main()
{
        P4 = 0x00;
        warte (1000);
        P4 = 0xAA;
        warte (1000);
        P4 = 0x55;
        warte (1000);
        P4 = 0xAA;
        warte (1000);
        P4 = 0x00;
}
```

Der eingegebene Code wird unter dem Namen ***PORT.C*** abgespeichert und der Editor beendet.

Die Compilierung des C-Source Codes erfolgt mit dem Aufruf:

C51 PORT.C

Der Compiler erzeugt die Dateien ***PORT.LST*** und ***PORT.OBJ***.
Als nächstes müssen jetzt für ein lauffähiges Programm die einzelnen Programm Module mit dem Linker zusammengefügt werden. Der Aufruf des Linkers erfolgt mit:

L51 PORT.OBJ,START.OBJ CODE(8000H) XDATA(0C000H) RAMSIZE(256)

START.OBJ ist ein Modul, welches nötig ist, um einwandfreie C Programme erzeugen zu können. Es löscht das interne RAM und bereitet die STACK-Verwaltung für Hochsprachenprogramme vor.
Vom Linker werden die Dateien ***PORT*** und ***PORT.M51*** erzeugt. ***PORT*** ist das lauffähige Programm, ***PORT.M51*** ist die Linkmap. Es empfiehlt sich, für den Linkeraufruf eine Batchdatei ***LINK.BAT*** zu schreiben, da die Anweisung meist sehr lang ist.

Mit ***OH51 PORT***

wird der Code in Standard Intel Hex Format gebracht (***PORT.HEX***), das für das Monitorprogramm nötig ist.
Geladen und gestartet wird das Programm mit Hilfe des Monitorprogramms, wie schon in der Übung 1 beschrieben wurde.

🕮 Übung 2:

Geben Sie das Programm PORT.C ein, erzeugen Sie ein lauffähiges Programm und laden Sie es in das µP-Board und starten Sie es.
Programmieren Sie anschließend als Erweiterung eine Endlosschleife mit Hilfe der *while(....)* Anweisung, damit das Programm bis zum Drücken der Taste RESET läuft.

4.3 Zugriff auf einzelne Bits

Mit dem eben geschriebenen Programm wird immer auf alle acht Leitungen des Ports gleichzeitig zugegriffen. Der 8051 hat jedoch special funktion register, bei denen auf jedes Bit einzeln zugegriffen werden kann. Dazu gehört auch der Port 4. Welche Register sonst noch bit-adressierbar sind, ist dem Datenbuch des jeweiligen Controllers zu entnehmen.
Zum Zugriff auf ein Bit eines bit-adressierbaren sfr wird die Datendefinition ***sbit*** (= ***s****pecial* ***bit***) verwendet. Wenn man in der Datei REG517.H nachsieht, wird man sehen, daß diese Datenvereinbarungen noch nicht in der Datei enthalten sind. Sie müssen daher entweder in einer selbst zu schreibenden Header Datei oder in der eigentlichen Anwenderprogrammdatei hinzugefügt werden. Die Datenvereinbarungen für Port 1 lauten:

```
/****   Datenvereinbarung der special bits             ****/
sbit P4_0 = 0xE8;
sbit P4_1 = 0xE9;
sbit P4_2 = 0xEA;
sbit P4_3 = 0xEB;
sbit P4_4 = 0xEC;
sbit P4_5 = 0xED;
sbit P4_6 = 0xEE;
sbit P4_7 = 0xEF;
```

🕮 Übung 3:

Schreiben Sie ein Programm, welches ein Lauflicht mit den acht Leuchtdioden erzeugt. Verwenden Sie dazu die Datenvereinbarungen der einzelnen Bits von Port 4, wie oben gezeigt. Das Programm soll außerdem in einer Endlosschleife laufen.

5 Programmbeispiel 1

```
/* ********************************************** */
/* Programm: port.c                               */
/* Bitweiser Zugriff auf die LEDs an Port4 ******** */

/* Preprozessoranweisungen */
#include <REG517A.H>

/* Variablendefinitionen */
sbit LED0=0xE8;
sbit LED1=0xE9;
sbit LED2=0xEA;
sbit LED3=0xEB;
sbit LED4=0xEC;
sbit LED5=0xED;
sbit LED6=0xEE;
sbit LED7=0xEF;


/* Funtionen */
void warte(msec)
unsigned int msec;
{
 data unsigned char i;
 for (msec; msec!=0; msec--)
   for (i=0; i<255; i++);
}

/* Hauptprogramm */
main()
{
   while(1)
   {
      P4 = 0;
      warte(1000);
      LED0=1;
      warte(1000);
      LED1=1;
      warte(1000);
      LED2=1;
      warte(1000);
      LED3=1;
      warte(1000);
      LED4=1;
      warte(1000);
      LED5=1;
      warte(1000);
      LED6=1;
      warte(1000);
      LED7=1;
      warte(1000);

   }
}
```

6 Programmbeispiel 2

```
/* ********************************************** */
/* Programm: port1.c  ***************************** */
/* Zähleranzeige mit LED an Port1 ***************** */
/* ********************************************** */

/* Preprozessoranweisungen */

#include <reg517.h>

/* Variablendefinitionen */

/* Funtionen */
void warte(msec)
unsigned int msec;
{
  data unsigned char i;
     for (msec; msec!=0; msec--)
        for (i=0; i<255; i++);
}

/* Hauptprogramm */
main()
{
  while(1)
  {
     P1 = 0;
     warte(1000);
     while (P1 < 0xFF)
     {
        P1++;
        warte(500);
     }
  }
}
```

7 Programmbeispiel 3

```
/* ****************************************** */
/* ** Programm: port2.c                       */
/* Einfaches Lauflicht an Port1 ************* */
/* ****************************************** */

#pragma debug pagelength (54)
#include <stdio.h>
#include <reg517.h>

/* Funtionen */
void warte(msec)
unsigned int msec;
{
 data unsigned char i;
 for (msec; msec!=0; msec--)
   for (i=0; i<255; i++);
}
/* Hauptprogramm */
main()
{
   while(1)
   {
      P1 = 0x01;
      warte(1000);
      while (P1 < 0x7F)
      {
         P1<<=1;
         warte(1000);
      }
      while (P1 > 0x02)
      {
         P1>>=1;
         warte(1000);
      }
   }
}
```