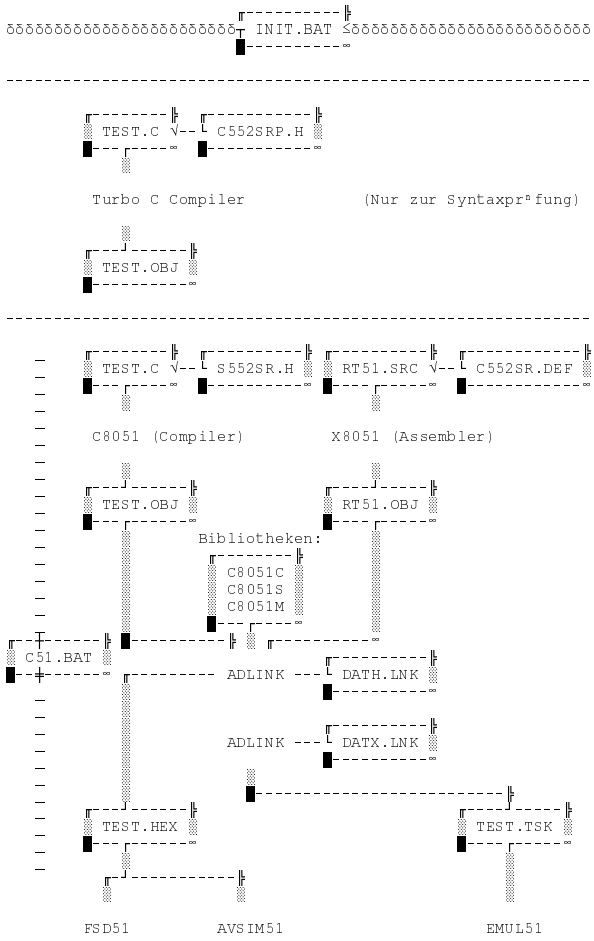
ENTWICKLUNGSSYSTEM FÜR 80C552

Gregor POKORNY,TGM,NT/EL,0SL90 TGM-DSK-169:\C*.*

TGM-interne Details sind kursiv gedruckt.

Ziel dieser Zusammenstellung war es, verschiedene, schon bestehende Softwarepakete, so miteinander zu verbinden, daß ein Entwickeln und Testen von C und Assemblerprogrammen für den 552 möglichst einfach und zweckmäßig wird. Die Programme im einzelnen: Compiler **C8051**, Assembler **X8051**, Linker **LINK**, Simulator **AVSIM51**, Debugger **FSD51**, Emulator **EMUL51**. Eine Anpassung an andere Mikocontrollertypen ist möglich.

Diagramm



Erläuterung zum Diagramm

Bevor man mit dem C-Compiler arbeiten kann, müssen am PC einige Dinge eingestellt werden. Diese Aufgabe erledigt `INIT.BAT`.

Das Diagramm zeigt die Entstehung eines Programms für den 552. Zuerst wird ein C-Programm geschrieben, in welches das File C552SR.H inkludiert werden muß, damit die Namen aller im 552 befindlichen Register im C-Programm bekannt sind. Dieses C-Programm schreibt man am besten im Turbo C++-Editor, da dieser Syntax durch Compilieren in ein Objektfile prüft, bevor eine Compilierung mit dem Compiler für den 552 durchgeführt wird. Diese Syntaxprüfung ist sinnvoll, weil der C-Compiler wesentlich schneller als der 552-Compiler ist und, weil manche Fehler vom 552-Compiler nicht erkannt werden, und die Gefahr besteht, daß das Programm dann nicht ordnungsgemäß läuft. Allerdings muß für diese Compilierung das File `C552SRP.H` inkludiert werden, da der Turbo-C-Compiler z.B. das Wort `NEAR` nicht versteht. (Dieses Wort sagt dem 552-Compiler, daß diese Variable im internen RAM abgelegt werden soll.) Wenn man vergißt diese Änderung wieder rückgängig zu machen bevor man den 552 Compiler startet, dann wird z.B. statt einer Ausgabe der `PWM` auf den entsprechenden Port, dieser Wert irgendwo im externen RAM abgelegt, wo er natürlich keine Wirkung hat. Beim Compilieren entsteht ein File mit der Erweiterung `.OBJ`.

Damit ein C-Programm am 552 ordnungsgemäß laufen kann, ist das Assemblerprogramm `RT51.SRC` notwendig, das einige Vorbereitungen trifft. In diesem Programm werden hauptsächlich Variablen definiert und initialisiert. Zum Definieren der Variablen wird das File `C552SR.DEF` in `RT51.SRC` eingebunden. Nach dem Assemblieren mit `X8051.EXE` entsteht ebenfalls ein File mit der Erweiterung `OBJ`. Alle diese `OBJ`-Files werden danach mit dem Linker verbunden. Zusätzlich können auch noch Programmbibliotheken eingebunden werden. Der Linker braucht dazu einige Informationen, die entweder händisch eingegeben, oder als File angebunden werden können. Hier können auch Optionen eingegeben werden. Diese sind unter anderem auch dafür verantwortlich, welches Format das Outputfile haben wird. Hier ist es vorgesehen, daß diese Informationen angebunden werden (`DATH.LNK` und `DATX.LNK`). Da für die verschiedenen Testmöglichkeiten verschiedene Outputfiles benötigt werden, muß der Linker zweimal gestartet werden. Einmal, um ein Outputfile im Intel-Hex-Format, und einmal, um ein binäres File zu erzeugen.

Der `AVSIM51` braucht zum Beispiel ein File im Intel-Hex-Format. Die genaue Bedienung des `AVSIM51` ist dem gleichnamigen Beitrag weiter hinten zu entnehmen. Der `AVSIM51` ist allerdings mit Vorsicht zu genießen, da sich manche Dinge zwar simulieren lassen, aber dann in Wirklichkeit doch nicht funktionieren. Leider war es mir nicht möglich einen solchen Fehler genau zu lokalisieren, da die C-Programme nach dem Compilieren nicht mehr gut nachvollziehbar sind.

Der `FSD51` verlangt auch ein File im Intel-Hex-Format. Außerdem ist zum Testen ein Prozessorprint nötig. Das hat den Vorteil, daß es sich dabei um keine Simulation handelt, hat aber den Nachteil, daß man sich auch öfter mit Hardwareproblemen herumschlagen muß.

Der `EMUL51` braucht ein File im BIN-Format. Er ist ein sehr komplexes Hard- und Softwarepaket, und bietet neben der besseren Emulation auch die Möglichkeit, die serielle Schnittstelle zu benutzen, was beim `FSD51` nur bedingt möglich ist, da er diese selbst zur Datenübertragung mit dem PC braucht. Außerdem können damit auch Programme ohne Prozessorprint getestet werden. Der einzige Nachteil ist, daß man eine umfangreiche, englische Anleitung zu studieren ist.

Installation

Die Programme sind samt Unterverzeichnissen mit `backup` *aus DOS 3.30 auf HD Disketten abgespeichert, und können mit* `restore a: c:/s` *wieder auf der Festplatte C: installiert werden. Es werden dann alle Verzeichnisse wiederhergestellt. Sollten die Programme auf einem anderen Laufwerk installiert werden, so muß diese Änderung in den Batchdateien berücksichtigt werden.*

Weiters muß die Datei SETPATH geschrieben werden und, z.B. im Verzeichnis `BAT`, abgelegt werden. In AUTOEXEC.BAT sollte dann auch statt der Angabe eines Pfades die Datei `SETPATH` aufgerufen werden. (Siehe auch Erläuterung zu `SETPATH` und Ausdruck von `SETPATH`). Außerdem muß ein virtuelles Laufwerk mit dem Namen `e:` installiert werden. Um C-Programme zweckmäßig editieren zu können, ist es auch zweckmäßig Turbo-C zu installieren.

Systemkonfiguration

```
C:\                          E:\
√-...                        ▒-C552
√-BAT
√-DOS
√-80552
▒ √-AVSIM51
▒ √-C552
▒ ▒ √-WINKLER
▒ ▒ ▒-2500AD
▒ ▒ ▒   √-BIN
▒ ▒ ▒   √-INCLUDE
▒ ▒ ▒   ▒-LIB
▒ √-FSD51
▒ ▒-EMUL51
...
```

```
C:\80552:                  INIT.BAT

C:\80552\AVSIM51:          AVSIM51.EXE, AVSIM51.HLP,
                           AVSIM51.OVR, AVSIM51.REG, GMENU.COM,
                           M.MNU, SIM.BAT, SIM.CMD, SIM.SYM

C:\80552\C552:             C51.BAT, C552SR.DEF, C552SR.H,
                           C552SRP.H, DATH.LNK, DATX.LNK,
                           RT51.OBJ, RT51.SRC, TEST.C, TEST.HEX,
                           TEST.MAP, TEST.OBJ, TEST.SYM,
                           TEST.TSK

C:\80552\C552\WINKLER:     AADC.C, AADCG.C, CADC.C, DADC.C,
                           FIRST.C, MADC.C, PIOLD.C, PIREG.C,
                           PIREG_L.C, PIREG_R.C, PIREG_RL.C,
                           RADC.C, TADC.C, TADCG.C, TESTADC.C,
                           TM2.C, TM2GUT.C, VADC.C

C:\80552\C552\2500AD\BIN:  ADLINK.EXE, C8051.EXE, C8051CG.EXE,
                           C8051P1.EXE, C8051PA.EXE, LIB.EXE,
                           MP.EXE, X8051.EXE.

C:\80552\C552\2500AD\INCLUDE:    C8051FI.SRC, C8051IFI.SRC,
                           C8051RT.SRC, C8051IIO.H, C8051IN.H,
                           C8051IO.H, C8051SR.H, ERROR.H,
                           MATH.H.

C:\80552\C552\2500AD\LIB:  C8051C.SRC, C8051IC.SRC, C8051C.LIB,
                           C8051IC.LIB, C8051IM.LIB,
                           C8051IP.LIB, C8051IS.LIB, C8051M.LIB,
                           C8051P.LIB, C8051S.LIB.

C:\80552\FSD51:            FSD51.EXE, GESCHW.M51, TEST.SET.

C:\80552\EMUL51:           80552.STR, 80552.SYM, EM.BAT,
                           EMUL31.HLP, EMUL31.NDX, EMUL31.EXE,
                           I751, PRIDOC.EXE, R751, SHOW51.COM,
                           TEST, TEST.A03, TEST.C, TEST.SRC,
                           TEST1.C.
```

FSD51

Einleitung

Es handelt sich bei dieser Dokumentation um eine Kurzbeschreibung des `FSD51` zum Testen eines C-Programms. Da ein C-Programm nach dem Compilieren ohnedies als unlesbar zu bezeichnen ist, wird in diesem Fall der `FSD51` im Prinzip nur zum Starten des Programms verwendet. Wichtig ist auch noch, da es sich bei dieser Version des `FSD51` um eine spezielle Version handelt, die seinerzeit im Zusammenhang mit dem Robbie entwickelt wurde. Außerdem ist das dazugehörige EPROM nur auf dem alten 100-er-Print verwendbar, da einige Adreßleitungen ausgekreuzt sind. Da momentan noch kein anderer Print zu haben ist, wird eben dieser Print noch immer zu Testzwecken verwendet. Sollte ein neuer Prozessorprint fertiggestellt werden, sollte man sich auch eine neue Version des `FSD51` mit dem dazugehörigen EPROM besorgen.

Hardware

Die Hardware besteht aus dem Prozessorprint, und einem Pegelwandler für die serielle Schnittstelle. Der Prozessorprint muß mit 5V versorgt werden und über den Pegelwandler an eine der seriellen Schnittstellen des PC's angeschlossen werden.

Software

Der `FSD51` wird mit `FSD51 TEST.SET` gestartet. `<spc>` drücken, um den Begrüßungsbildschirm zu verlassen. Danach kommt der Hauptbildschirm, in dem immer ein Fenster heller erscheint. Ein Wechsel der Fenster ist mit `<home>` oder `<end>` möglich. Jetzt bestehen zwei Möglichkeiten: Entweder Der `FSD` übertrgt brav interne und externe Daten oder die Schnittstelle ist falsch eingestellt, und der FSD meldet sich mit `Empfänger nicht bereit` In diesem Fall kann mit `F3 Setup` die Schnittstelle geändert werden. Es empfiehlt sich auch das Setup gleich zu speichern, da sonst derselbe Fehler beim nächsten Mal wieder auftritt.

Laden eines Programms

Mit `F2 File`, und `File download` wählen. Dann erscheint der Name, der zuletzt eingegeben wurde, oder der Name, der im Setup abgespeichert war. Beim Speichern des Setups wird nämlich der Name, der hier gerade steht mitgespeichert, was den Vorteil hat, daß man beim Laden des Programms zumeist nur den Menüpunkt anwählt und `<Enter>` betätigt. Jetzt wird das Programm ins RAM des Prozessorprints geladen, und interne und externe Daten übertragen.

Starten des Programms

Zuerst muß der PC an die richtige Stelle gesetzt werden. Das geschieht im Fenster links oben. Die Adresse muß mit `<Enter>` bestätigt werden. Jetzt kann das Programm mit `F6` gestartet werden. Unterbrochen kann es jetzt nur noch mit einem Hardwarereset werden, da es vollkommen unabhängig vom PC läuft. Man könnte sogar die serielle Schnittstelle abstecken, ohne daß das Programm in seinem Ablauf beeinträchtigt würde.

Verlassen des FSD

Der FSD kann mit `F10` wieder verlassen werden. Das Programm am Prozessorprint läuft aber weiter.

Beschreibung der Files

INIT.BAT

```
Aufruf: INIT


call setpath
e:
md c552
cd c552
xcopy c:\80552\c552\*.* e:
set emul31=c:\80552\emul51
set include=e:\c552
set lib=c:\80552\c552\2500ad\lib
```

In diesem File werden Vorbereitungen für die Bearbeitung eines Programms getroffen. Voraussetzung für die Funktion ist, daß zuvor ein virtuelles Laufwerk `E:` installiert wurde. Zuerst wird mit der BAT-Datei `SETPATH` der Pfad wiederhergestellt, falls er, zum Beispiel durch einen Netzeinstieg, verändert worden ist. Dann wird auf `E:` ein Verzeichnis mit dem Namen `\C552` angelegt, in das alle zu bearbeitenden Files kopiert werden. Dieser Aufwand ist deshalb gerechtfertigt, weil der C-Compiler eine sehr langsame Angelegenheit ist, die damit etwas beschleunigt wird. Am Schluß werden noch drei Pfadangaben ins Enviroment geschrieben, damit Compiler und Linker wissen, wo sie Include- und Lib-Dateien suchen sollen, und damit der `EMUL51` funktioniert.

SETPATH.BAT

```
PATH=C:\BAT;C:\DOS;D:\ASM51;C:\80552\c552\2500AD\BIN;C:\TOOLS;
C:\TOOLS\NU;c:\tc\bin
```

Diese Datei setzt den Pfad. Sie wird von `INIT.BAT` aufgerufen und ist so gedacht, daß sie ganz allgemein verwendet wird, z.B. auch von AUTOEXEC.BAT. Wichtig für diese Software ist nur, daß dabei der Pfad `C:\80552\C552\2500AD\BIN` vorkommt. `SETPATH.BAT` ist dafür nur ein Beispiel.

C51.BAT

```
Aufruf: C51 test


e:
cd\c552
del %1.obj
c8051 %1
adlink dath
adlink datx
xcopy e:\c552\%1.* c:\80552\c552
```

Diese Datei löscht zuerst das OBJ-File, sonst würden bei nicht erfolgreicher Kompilierung die alten Files zusammengelinkt, was man dann leicht übersieht. Dann wird der Compiler gestartet (und hinterher zweimal der Linker), um ein HEX-File (für `FSD51` und `AVSIM51`) und ein TSK-File (für `EMUL51`) zu erzeugen. Am Schluß werden noch alle Dateien, die mit dem Projekt zu tun haben auf die Festplatte zurückkopiert.

TEST.C

```
#include "c552sr.h"

extern struct special_function_bits SFB;
   int a=1;

main()
{
  while (1)
  {
  SFB.P1_5 = ~SFB.P1_5;
  }
}
```

`TEST.C` ist das eigentliche C-Programm (siehe Diagramm-Beschreibung). Damit man die Special-Function-Register bitadressieren kann, muß das `struct special_function_bits SFB` extern deklariert werden. Danach steht das eigentliche C-Programm für den 552. Das hier abgedruckte Programm invertiert nur ein Portbit, was beim Testen sehr schön am Oszi zu beobachten ist, und in jedem Programm zum Testen zweckmäßig ist.

TEST.HEX

```
:02A22500000136
:108C0000EE24FCFEEF34FFFF79007401C39970037A
:108C1000028C25E430950104F420E004C295800222
:108C2000D295028C08F5F0EE2404FEEF3400FFE547
:028C3000F02230
:0FA2270001A238A23800000000A338A3380000BD
:03A0000002A103B7
:03A0530002A20165
:03A0730002A21036
:10A1000002A18E7E7F7FFF743675F0A290A236C3C7
:10A110009582F8E5F09583F94860387436C3943633
:10A12000FA74A294A2FBEA2436FAEB34A2FB7C0078
:10A130007D808A828B83E493A3AA82AB838C828DF9
:10A1400083F0A3AC82AD83E8C39401F8E99400F9ED
:10A150004870DF743875F0A490A236C39582F8E594
:10A16000F09583F948600FE4F0A3E8C39401F8E99F
:10A170009400F94870F190A236747FC39400FE7485
:0EA18000FF9400FFF0A3EEF0128C0002A0008E
:10A18E0022C0D0C0E0E58124FDC0F0C082C083C0F3
:10A19E0000C001C002C003C004C005C006C007F85D
:10A1AE0090A236E0C0E0A3E0C0E0EEC39400FEF063
:10A1BE00EF9400FF90A236F0E6C0E008E6C0E02281
:10A1CE00EE2400FEEF3400FF90A236D0F0D0E0F087
:10A1DE00A3E5F0F0D007D006D005D004D003D0020E
:10A1EE00D001D000D083D082D0F0D0E0D0D0158175
:10A1FE00158132C0D0C0E053C5EF432480D0E0D0EB
:10A20E00D032C0D0C0E053CB7F53EAEF43C50843F2
:07A21E002401D0E0D0D03292
:02A00003A000BB
:00000001FF
```

Dieses File entsteht, beim Linken, und dient als Inputfile für den `AVSIM51` und den `FSD51`. Am Anfang jeder Zeile steht immer ein Doppelpunkt, danach die Anzahl der Bytes, danach die Adresse, und dann die Daten. Dieses Wissen kann vor allem bei der Suche von Link-errors praktisch sein.

C552SR.H

```
/*********************************************************/
/* 8051 C COMPILER SPECIAL FUNCTION REGISTER DECLARATIONS */
/*********************************************************/

.asm
.chip 8051
.endasm
extern  near  char  ACC; /* power control       */
extern  near  char  B;   /* power control       */
extern  near  char  SP;  /* power control       */
extern  near  char  DPL; /* power control       */
extern  near  char  DPH; /* power control       */
extern  near  char  PSW; /* program status word */
extern  near  char  P0;    /* port 0               */
extern  near  char  P1;    /* port 1               */
extern  near  char  P2;    /* port 2               */
extern  near  char  P3;    /* port 3               */
extern  near  char  P4;    /* port 4               */
extern  near  char  P5;    /* port 5               */
extern  near  char  ADCON; /* ADC control          */
extern  near  char  ADCH;  /* ADC high             */
extern  near  char  S0CON; /* serial control       */
extern  near  char  S0BUF; /* serial buffer        */
extern  near  char  TMOD;  /* timer mode           */
extern  near  char  TCON;  /* timer control        */
extern  near  char  PCON;  /* power control        */
extern  near  char  TM2CON;/* timer 2 control      */
extern  near  char  CTCON; /* capture control      */
extern  near  char  S1CON; /* serial 1 contorl     */
extern  near  char  S1STA; /* serial 1 status      */
extern  near  char  S1DAT; /* serial 1 data        */
extern  near  char  S1ADR; /* serial 1 address     */
extern  near  char  TL0;   /* timer 0 low byte     */
extern  near  char  TH0;   /* timer 0 high byte    */
extern  near  char  TL1;   /* timer 1 low byte     */
extern  near  char  TH1;   /* timer 1 high byte    */
extern  near  char  TML2;  /* timer 2 low byte     */
extern  near  char  TMH2;  /* timer 2 high byte    */
extern  near  char  T3;    /* timer 3              */
extern  near  char  IP0;  /* interrupt priority control */
extern  near  char  IP1;  /* interrupt priority control */
extern  near  char  IEN0; /* interrupt enable control   */
extern  near  char  IEN1; /* interrupt enable control   */
extern  near  char  TM2IR;/* timer 2 inerrupt flags     */
extern  near  char  STE;  /* set enable                 */
extern  near  char  RTE;  /* reset/toggle enable        */
extern  near  char  PWMP; /* PWM prescaler              */
extern  near  char  PWM0; /* PWM 0                      */
extern  near  char  PWM1; /* PWM 1                      */
```

```
extern  near  char  SIGNAL;
extern  near  char  T2_OLD;

/************************************************/
/* 8051 C COMPILER SPECIAL FUNCTION BIT DECLARATIONS */
/************************************************/

near struct special_function_bits
{                        /*   Bit Name    Bit Value */

/* P0  (Port 0)                          */
unsigned int P0_0   : 1; /* bit 0   80h */
unsigned int P0_1   : 1; /* bit 1   81h */
unsigned int P0_2   : 1; /* bit 2   82h */
unsigned int P0_3   : 1; /* bit 3   83h */
unsigned int P0_4   : 1; /* bit 4   84h */
unsigned int P0_5   : 1; /* bit 5   85h */
unsigned int P0_6   : 1; /* bit 6   86h */
unsigned int P0_7   : 1; /* bit 7   87h */

/* TCON  (Timer Control)                                  */
unsigned int IT0: 1; /* input INT0/ transition activ.  88h */
unsigned int IE0: 1; /* ext interr.request flag 0  89h     */
unsigned int IT1: 1; /* input INT1/ transition activ.  8ah */
unsigned int IE1: 1; /* ext interr.request flag 1  8bh     */
unsigned int TR0: 1; /* counter 0 enable/disable      8ch  */
unsigned int TF0: 1; /* counter 0 interr.req & overfl 8dh  */
unsigned int TR1: 1; /* counter 1 enable/disable      8eh  */
unsigned int TF1: 1; /* counter 1 interr.req & overfl 8fh  */

/* P1  (Port 1)                          */
unsigned int P1_0: 1; /* bit 0  90h */
unsigned int P1_1: 1; /* bit 1  91h */
unsigned int P1_2: 1; /* bit 2  92h */
unsigned int P1_3: 1; /* bit 3  93h */
unsigned int P1_4: 1; /* bit 4  94h */
unsigned int P1_5: 1; /* bit 5  95h */
unsigned int P1_6: 1; /* bit 6  96h */
unsigned int P1_7: 1; /* bit 7  97h */

/* SCON   (Serial Port Control)                           */
unsigned int RI :1; /* reception complete int.flag    98h */
unsigned int TI :1; /* transmission complete int.flag 99h */
unsigned int RB8:1; /* received data bit 8            9ah */
unsigned int TB8:1; /* transmitter data bit 8         9bh */
unsigned int REN:1; /* receiver enable                9ch */
unsigned int SM2:1; /* conditional receiver enable    9dh */
unsigned int SM1:1; /* serial port operation mode lsb 9eh */
unsigned int SM0:1; /* serial port operation mode msb 9fh */

/* P2   (Port 2)                         */
unsigned int P2_0: 1; /* bit 0 a0h */
unsigned int P2_1: 1; /* bit 1 a1h */
unsigned int P2_2: 1; /* bit 2 a2h */
unsigned int P2_3: 1; /* bit 3 a3h */
unsigned int P2_4: 1; /* bit 4 a4h */
unsigned int P2_5: 1; /* bit 5 a5h */
unsigned int P2_6: 1; /* bit 6 a6h */
unsigned int P2_7: 1; /* bit 7 a7h */

/* IE   (Interrupt Enable)                                */
unsigned int EX0: 1;  /* external request 0       a8h */
unsigned int ET0: 1;  /* internal timer/counter 0 a9h */
unsigned int EX1: 1;  /* external request 1       aah */
unsigned int ET1: 1;  /* internal timer/counter 1 abh */
unsigned int ES : 1;  /* internal serial port     ach */
unsigned int IE_5: 1; /* reserved             adh     */
unsigned int IE_6: 1; /* reserved             aeh     */
unsigned int EA  : 1; /* enable all           afh     */

/* P3   (Port 3)                            */
unsigned int P3_0: 1; /* bit 0        b0h */
unsigned int P3_1: 1; /* bit 1        b1h */
unsigned int P3_2: 1; /* bit 2        b2h */
unsigned int P3_3: 1; /* bit 3        b3h */
unsigned int P3_4: 1; /* bit 4        b4h */
unsigned int P3_5: 1; /* bit 5        b5h */
unsigned int P3_6: 1; /* bit 6        b6h */
unsigned int P3_7: 1; /* bit 7        b7h */

/* IP   (Interrupt Priority)                              */
unsigned int PX0: 1;  /* external request 0       b8h */
unsigned int PT0: 1;  /* internal timer/counter 0 b9h */
unsigned int PX1: 1;  /* external request 1       bah */
unsigned int PT1: 1;  /* internal timer/counter 1 bbh */
unsigned int PS : 1;  /* internal serial port     bch */
unsigned int IP_5: 1; /* reserved                 bdh */
unsigned int IP_6: 1; /* reserved                 beh */
unsigned int IP_7: 1; /* reserved                 bfh */

/*      (Unimplemented)                  */
unsigned int C0H: 1; /* unimplemented  c0h */
unsigned int C1H: 1; /* unimplemented  c1h */
unsigned int C2H: 1; /* unimplemented  c2h */
unsigned int C3H: 1; /* unimplemented  c3h */
unsigned int C4H: 1; /* unimplemented  c4h */
unsigned int C5H: 1; /* unimplemented  c5h */
unsigned int C6H: 1; /* unimplemented  c6h */
unsigned int C7H: 1; /* unimplemented  c7h */

/*      (Unimplemented)                  */
```

C552SR.DEF

```
*********************************************************
* 8051 C COMPILER SPECIAL FUNCTION REGISTER DEFINITIONS *
*********************************************************

spec_funct_regs:  .section   reg, offset 0, ref_only

          .globals  on

_ACC:          .reg    e0h   ;accumulator
_B:            .reg    f0h   ;B register
_PSW:          .reg    d0h   ;program status word
_SP:           .reg    81h   ;stack pointer
_DPL:          .reg    82h   ;data pointer low
_DPH:          .reg    83h   ;data pointer high
```

```
unsigned int C8H: 1; /* unimplemented c8h */
unsigned int C9H: 1; /* unimplemented c9h */
unsigned int CAH: 1; /* unimplemented cah */
unsigned int CBH: 1; /* unimplemented cbh */
unsigned int CCH: 1; /* unimplemented cch */
unsigned int CDH: 1; /* unimplemented cdh */
unsigned int CEH: 1; /* unimplemented ceh */
unsigned int CFH: 1; /* unimplemented cfh */

/* PSW  (Processor Status Register)                */

unsigned int P    : 1; /* parity                d0h */
unsigned int PSW_1: 1; /* reserved              d1h */
unsigned int OV   : 1; /* overflow              d2h */
unsigned int RS0  : 1; /* register bank select 0 d3h */
unsigned int RS1  : 1; /* register bank select 1 d4h */
unsigned int F0   : 1; /* user flag 0           d5h */
unsigned int AC   : 1; /* auxiliary carry       d6h */
unsigned int CY   : 1; /* carry                 d7h */

/*      (Unimplemented)                  */

unsigned int D8H: 1; /* unimplemented         d8h */
unsigned int D9H: 1; /* unimplemented         d9h */
unsigned int DAH: 1; /* unimplemented         dah */
unsigned int DBH: 1; /* unimplemented         dbh */
unsigned int DCH: 1; /* unimplemented         dch */
unsigned int DDH: 1; /* unimplemented         ddh */
unsigned int DEH: 1; /* unimplemented         deh */
unsigned int DFH: 1; /* unimplemented         dfh */

/* ACC   (Accumulator)                    */

unsigned int ACC_0: 1; /* accumulator bit 0 e0h */
unsigned int ACC_1: 1; /* accumulator bit 1 e1h */
unsigned int ACC_2: 1; /* accumulator bit 2 e2h */
unsigned int ACC_3: 1; /* accumulator bit 3 e3h */
unsigned int ACC_4: 1; /* accumulator bit 4 e4h */
unsigned int ACC_5: 1; /* accumulator bit 5 e5h */
unsigned int ACC_6: 1; /* accumulator bit 6 e6h */
unsigned int ACC_7: 1; /* accumulator bit 7 e7h */

/*      (Unimplemented)                  */

unsigned int E8H: 1; /* unimplemented e8h */
unsigned int E9H: 1; /* unimplemented e9h */
unsigned int EAH: 1; /* unimplemented eah */
unsigned int EBH: 1; /* unimplemented ebh */
unsigned int ECH: 1; /* unimplemented ech */
unsigned int EDH: 1; /* unimplemented edh */
unsigned int EEH: 1; /* unimplemented eeh */
unsigned int EFH: 1; /* unimplemented efh */

/* B   (B Register)                       */

unsigned int B_0: 1; /* unimplemented f0h */
unsigned int B_1: 1; /* unimplemented f1h */
unsigned int B_2: 1; /* unimplemented f2h */
unsigned int B_3: 1; /* unimplemented f3h */
unsigned int B_4: 1; /* unimplemented f4h */
unsigned int B_5: 1; /* unimplemented f5h */
unsigned int B_6: 1; /* unimplemented f6h */
unsigned int B_7: 1; /* unimplemented f7h */

/*      (Unimplemented)                  */
unsigned int F8H: 1; /* unimplemented f8h */
unsigned int F9H: 1; /* unimplemented f9h */
unsigned int FAH: 1; /* unimplemented fah */
unsigned int FBH: 1; /* unimplemented fbh */
unsigned int FCH: 1; /* unimplemented fch */
unsigned int FDH: 1; /* unimplemented fdh */
unsigned int FEH: 1; /* unimplemented feh */
unsigned int FFH: 1; /* unimplemented ffh */

};
```

In diesem File werden die Variablen, die mit dem 552 zu tun haben deklariert. Wenn Namen im 552-Handbuch etwas anders sind, dann empfiehlt es sich hier nachzusehen.

C552SRP.H

Dieses File ist dem C552SR.H sehr ähnlich, nur so abgeändert, daß es der Turbo-C-Compiler versteht (siehe Diagramm-Beschreibung). Diese Datei ist hier nicht abgedruckt, da sie inhaltlich praktisch gleich der Datei C552SR.H ist mit folgenden Abweichungen: Es fehlen die drei mit einem Punkt beginnenden Zeilen, zu Beginn von C552SR.H; es fehlt das Schlüsselwort `near`. Die Datei ist aber auf TGM-DSK-169 enthalten.

```
_P0:           .reg    80h   ;port 0
_P1:           .reg    90h   ;port 1
_P2:           .reg    a0h   ;port 2
_P3:           .reg    b0h   ;port 3
_P4:           .reg    c0h   ;port 4
_P5:           .reg    c4h   ;port 5
_ADCON:        .reg    c5h   ;ADC control
_ADCH:         .reg    c6h   ;ADC high

_S0CON:        .reg    98h   ;serial control
_S0BUF:        .reg    99h   ;serial buffer
_TMOD:         .reg    89h   ;timer mode
_TCON:         .reg    88h   ;timer control
_PCON:         .reg    87h   ;power control
_TM2CON:       .reg    eah   ;timer 2 control
_CTCON:        .reg    ebh   ;capture control

_S1CON:        .reg    d8h   ;serial 1 control
_S1STA:        .reg    d9h   ;serial 1 status
```

```
_S1DAT:       .reg    dah   ;serial 1 data
_S1ADR:       .reg    dbh   ;serial 1 address

_TL0:         .reg    8ah   ;timer 0 low byte
_TH0:         .reg    8ch   ;timer 0 high byte
_TL1:         .reg    8bh   ;timer 1 low byte
_TH1:         .reg    8dh   ;timer 1 high byte
_TML2:        .reg    ech   ;timer 2 low byte
_TMH2:        .reg    edh   ;timer 2 high byte
_T3:          .reg    ffh   ;timer 3
_STE:         .reg    eeh   ;set enable
_RTE:         .reg    efh   ;reset/toggle enable

_IP0:         .reg    b8h   ;interrupt priority control
_IP1:         .reg    f8h   ;interrupt priority control
_IEN0:        .reg    a8h   ;interrupt enable control
_IEN1:        .reg    e8h   ;interrupt enable control
_TM2IR:       .reg    cbh   ;timer 2 interrupt flags

_PWMP:        .reg    feh   ;PWM prescaler
_PWM0:        .reg    fch   ;PWM 0
_PWM1:        .reg    fdh   ;PWM 1

_SIGNAL:      .reg    24h
_T2_OLD:      .reg    25h

_SFB:         .reg    80h   ;special function bits
                            ;starting address

          .globals  off

          .ends

; User bit structures located in the internal RAM may
; be defined here, or in another file that is included
; in 'c8051rt.src'. Creating a user defined file will keep
; the additions from being overwritten if the compiler is
; reinstalled. If another file is created, be sure and copy
; the 'section' directive shown below.

; The actual structure template can be defined in
; 'c8051sr.h', or in a user defined include file. The
; structure 'special_functions_bits' can be used as an
; example. Be sure and put an underbar in front of the name
; that will be used when accessing the structure in C.

; For example, if a structure consisting of 5 bits is to be
; defined and placed at the top of the internal memory, the
; following could be used: (The semi-colons must be removed
; for the statements to take effect)

;user_defined_bits:  .section     bit, offset 7bh, ref_only

;          .globals  on

;_FIVE_BITS:     .ds     5

;          .globals  off

;          .ends

; Note that the section is linked to start at 7bh by the
; 'offset' modifier on the section definition statement. This
; number should be changed as additional bit structures are
; defined.  Alternatively, the 'offset' modifier can be
; removed and the section can be located using an 'org'.

          .end
```

In diesem File werden die Register des 552 definiert. Es wird in das Assemblerfile RT51.SRC beim Assemblieren eingebunden.

DATH.LNK

```
test
8c00
_
rt51
0
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
test
c8051c
c8051s
c8051m
_
h
```

DATX.LNK

```
test
8c00
_
rt51
0
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
_
test
c8051c
c8051s
c8051m
_
x
```

DATH.LNK ist jenes File, das man dem Linker übergeben muß, damit er ein HEX-File erzeugt. Achtung! Es kann passieren, daß sich die Anzahl der einzugebenden Parameter ändert. Das passiert zum Beispiel, wenn man keine globalen Variablen im C-Programm definiert. Dann funktioniert das Linken nicht, solange man das LNK-File nicht ändert. Weiters muß bei Veränderung des Namens des C-Programms auch hier die Änderung vorgenommen werden. Zweckmäßiger ist es aber den Namen TEST.C beizubehalten und immer das Programm, das man momentan bearbeitet, in TEST.C umzubenenen.

DAT.LNK funktioniert genauso wie DATH.LNK, nur erzeugt der Linker dann kein HEX-, sondern ein TSK-File.

RT51.SRC

Nur auf Disk TGM-DSK-169. RT51.SRC ist das Assemblerfile, das vor dem C-Programm abgearbeitet wird (siehe Diagrammbeschreibung). Bemerkenswert ist dabei: Auf Page 2 unten wird C552SR.DEF eingebunden. Hier werden einige Variablen definiert. Weiters auf Page 3, mitte steht die Zeile `.org 0a000h`. Hier steht der Offset der Startadresse. Nachdem beim Linken für die Startadresse aber 0 angegeben wird, ist das gleichzeitig die Startadresse. Wirklich gestartet wird das Programm aber auf A103h. Den genauen Grund habe ich nicht erforscht. Es funktioniert einfach so. Weiters ist zu erwähnen, daß es sehr zweckmäßig ist, am Ende des RT51.SRC Interruptservice-Routinen zu schreiben, wie es Prof. Winkler schon gemacht hat (siehe Ende des Files). Achtung! Man darf danach nicht vergessen RT51.SRC neu zu assemblieren.

SIM.BAT

```
gmenu m
avsim51 bflsim.cmd
```

SIM.BAT ist die Datei, die man zum Starten des AVSIM51 verwendet. Beim AVSIM51 besteht die Möglichkeit, daß man Tasten, die man im Programm betätigen müßte, beim Aufruf als Buchstabenkombination anhängt. Weiters besteht im Programm die Möglichkeit ein File zu laden, das einige Tasten betätigt. Im SIM.BAT wird jetzt eine trickreiche Kombination dieser beiden Möglichkeiten angewendet, indem mit der Tastenkombination `bflsim.cmd` mit dem `"b"` der Prozessor ausgewählt wird, und danach das File `sim.cmd` nachgeladen wird, welches weitere Tasten betätigt.

SIM.CMD

```
lssim.sym
lpe:\c552\test.hex
a103
```

In diesem File sind die Tasten abgespeichert, die der AVSIM51 nach seinem Aufruf selbständig betätigen soll. In diesem Fall wird damit das HEX-File geladen und der PC auf `a103h` gestellt.

SIM.SYM

```
SG CODE  C:0000 FFFF
SG XDATA X:0000 FFFF
```

Aus diesem File bezieht der AVSIM51 Symbole und die Speicherkonfiguration.