Unterstützung beim Test von 68HC11-Programmen

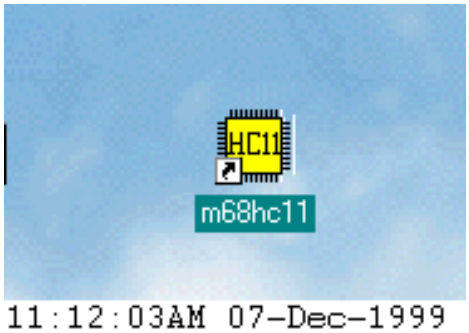
Im folgenden Programm wird die Bedingte Übersetzung zum Einbau von Testhilfen in ein Programm gezeigt, welches in seiner endgültigen Version in den 512 Byte des internen EEPROM eines 68HC11 unterzubringen ist. Dabei soll das Programm zum Test in den RAM eines ZWERG11plus mit dem Testmonitor ”MONI11m” geladen werden. Dieser Testmonitor stellt für die Interupts und für alle Anfangsladevorgänge die gleiche Umgebung wie im ”Spezial Bootstap Mode” zur Verfügung. Es besteht die Möglichkeit im Monitor Unterbrechungspunkte zu setzen und Speicher- oder Register-inhalte zu ändern. Für die Unterbrechungspunkte und für den Rücksprung in den Monitor wird der SWI-Befehl des 68HC11 verwendet. Der **SWI-Interupteinsprung** auf den Adressen **$00f4 - $00f6** darf dabei nicht durch den Stack oder andere Arbeitszellen überschrieben werden.

Die Startadresse aus der Systemvariablen ”start.11” wird dem Preprozessor mit dem Namen START_ übergeben und ist grün dargestellt. Die für den Testbetrieb notwendigen Einfügungen werden mit Includefiles vorgenommen und sind in roter Schrift dargestellt. Für den Test eines Programm im RAM des ZWERG11plus wurden die folgenden 6 Includefiles vorbereitet:

IF_stkC3.icl	- EEPROMstack = $C3 / Teststack = $1fff
IF_stkFF.icl	- EEPROMstack = $FF / Teststack = $1fff
IF_stack.icl	- EEPROMstack = stack / Teststack = $1fff („stack“ ist ein Symbol)
IF_lmon.icl	- Initialisierung des SCI und des MONI11m.
IF_eqmon.icl	- Unterproramm- und Datenadressen des MONI11m aus dem File Usr_rufe.icl. Die Benutzung wird im File usr_tst0.a gezeigt.
IF_mtst.icl	- Test der Eingabe eines Zeichens vom SCI und Neustart des MONI11m bei **ENTER** oder Sprung zu einem SWI-Befehl bei **ESC** und mögliche Fortsetzung des Programm nach Veränderungen im Speicher, in Registern oder nach dem Setzen eines Unterbrechungspunktes**.**

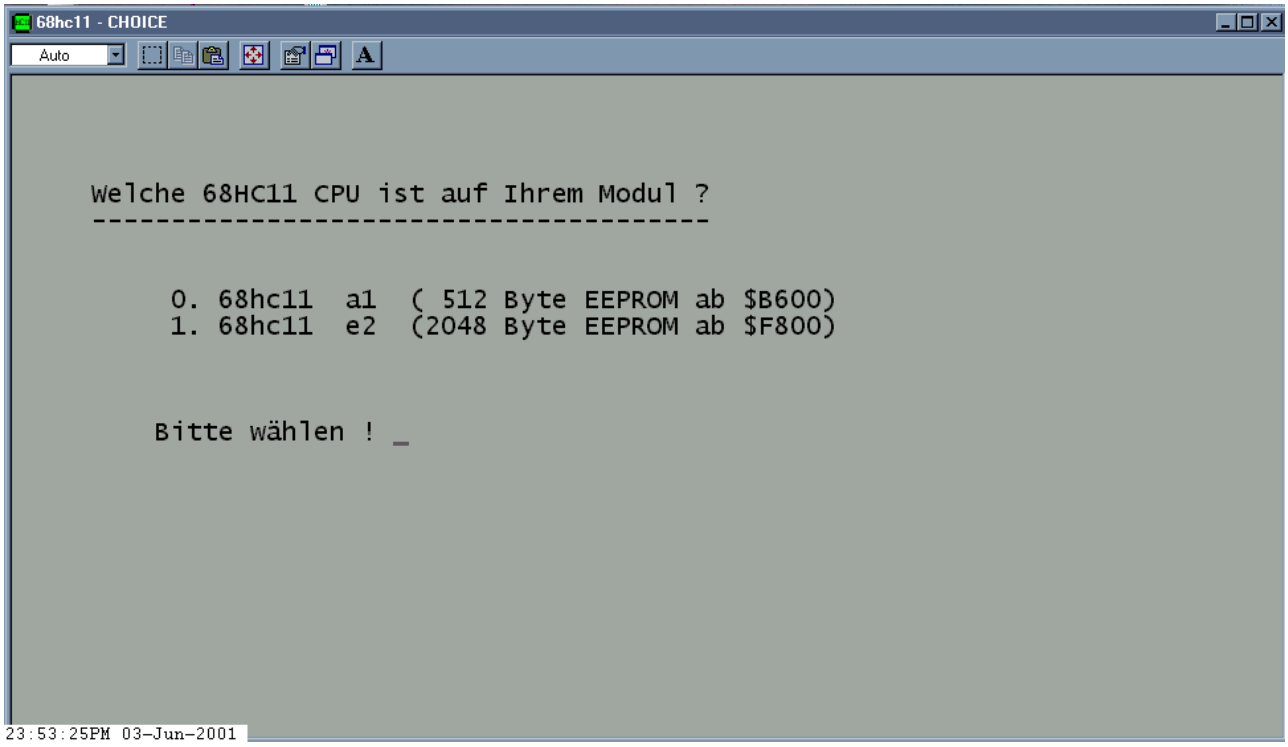
In diesen 6 Includefiles sind die Assemblerbefehle in IF-Anweisungen eingeschlossen, die alle zusätzlichen Test-Anweisungen nur dann übersetzen, wenn das Programm auf Adressse $2000 übersetzt wird. Dadurch muß an den Quellprogrammen nach einer Erfolgreichen Testung im RAM nichts mehr geändert werden um sie in den EEPROM des 68HC11 zu schreiben. Das Quellprogramm muß nur mit den verändeten Einstellungen für das Assemblersystem neu übersetzt werden. Diese Einstellungen werden von Shellscripten in Menue-Technik durchgeführt und sind damit leicht und fehlerfrei zu realisieren. Diese Software wird durch einen Doppelklick auf das folgende Icone gestartet.

Starticon auf dem Desktop eines PC:

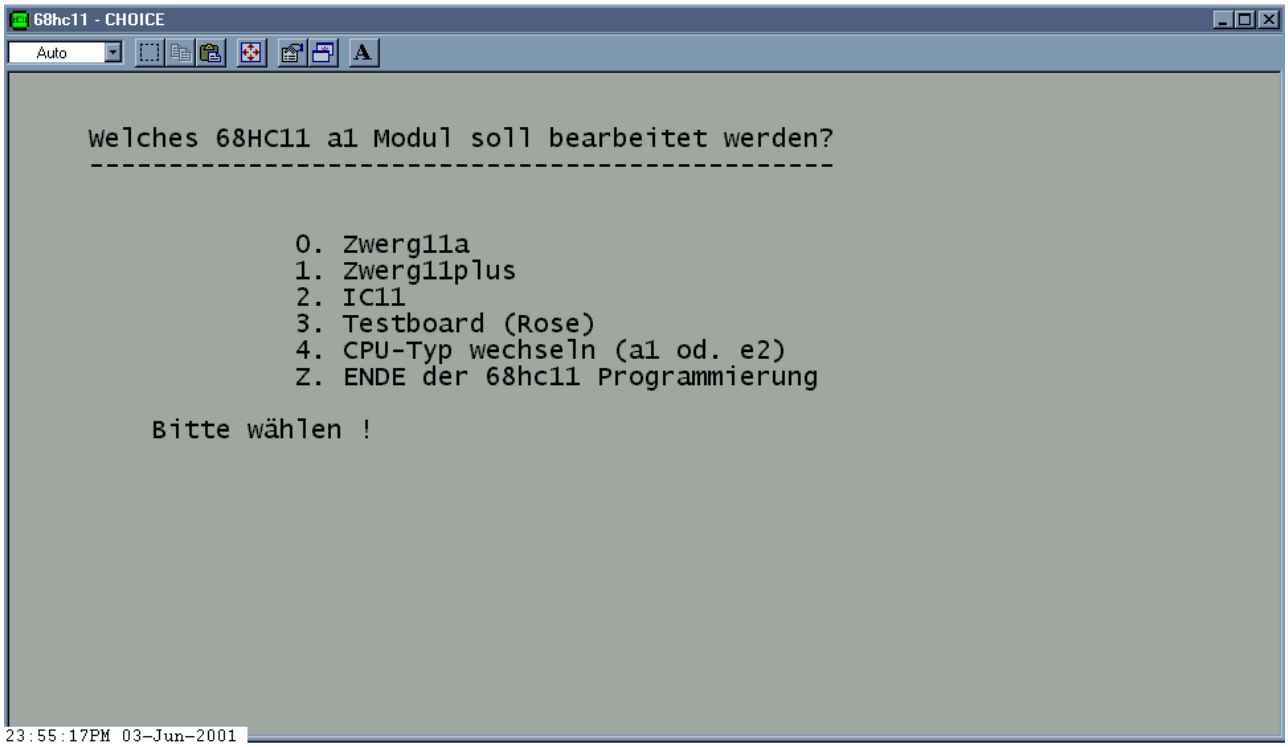


Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

1. Menu:

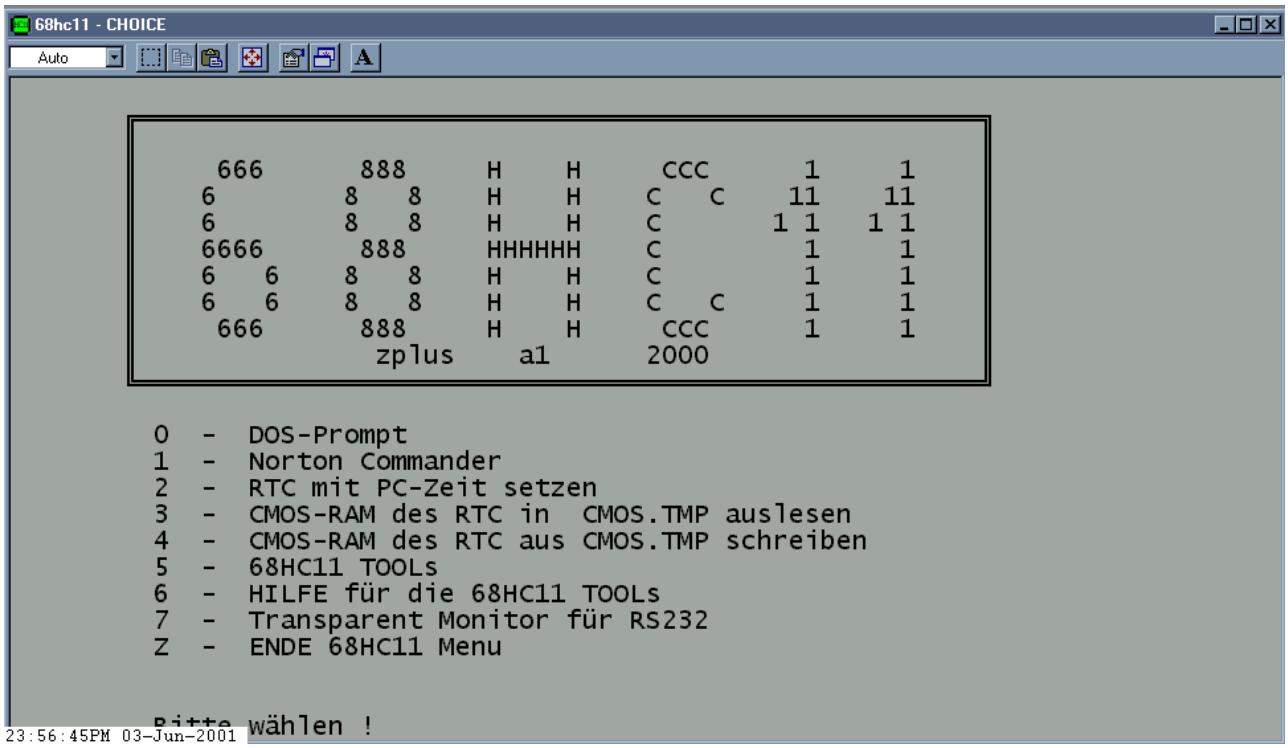


2. Menu nach der Eingabe von 0:

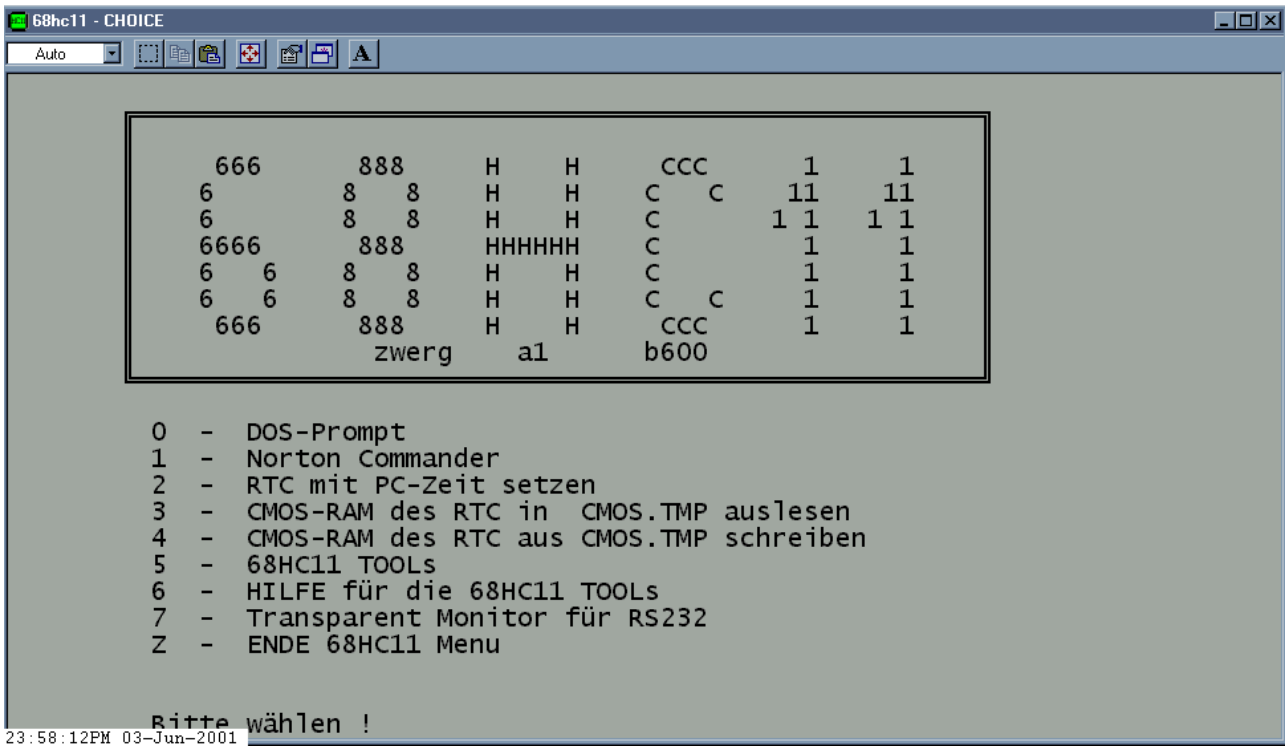


Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

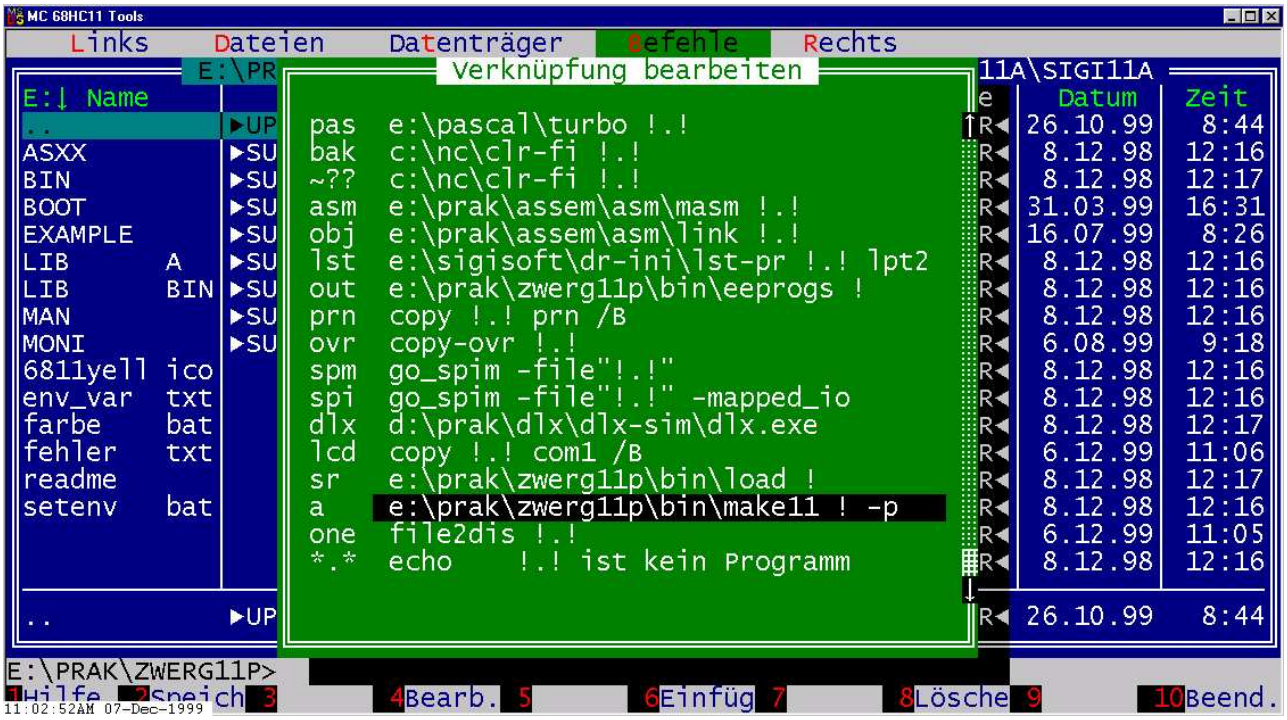
3. Menu nach der Eingabe von 1 im 2. Menu:



3. Menu nach der Eingabe von 0 im 2. Menu:



4. Mögliche Einstellungen für den Norton-Commander



Für die Filetypen *.out, *.a und *.sr sind Starts der 68HC11-Software vorbereitet

Programm mit eingebauten Testhilfen:

```
************************************************
*  Melodiegeneratorprogramm fuer den MC68HC11A1  *
*  - Entwurf  : Heiko Boettcher                  *
*               Meisenweg 1, 01589 Riesa         *
*  - Merkamle : 3 Oktaven Tonumfang mit Pausen   *
*               Notenlaengen von 1/16 bis 3/4    *
*               Tempo jedes Liedes einstellbar   *
*  - Erweitert: Dr. Siegmar Schöne               *
*  - neues    : Stacato abschaltbar, Frequenz-   *
*               bereich kann um 4 Oktaven ver-   *
*               schoben werden. 3 neue Songs.    *
*               2. Klingeltaster                 *
************************************************

*     Übersetzt auf Adresse $START_

#define     __     ***************

*__*** Konstanten ****

#include    <target.h>
#include    <IF_eqmon.icl>

otoc2       EQU   otoc2h            * Offset des toc2-Register zu REG

*__*** RAM-Belegung
            bss
            ORG   $0000
adr_nfrq    ds.w  1                 * Ablageplatz der Notenfrquenz
mode        ds.b  1                 * Bit 2 Stacato 10 F-Faktor
tempo       ds.b  1                 * Songtempo (Faktor für Notenlänge)
laenge      ds.w  1                 * Notenlänge
```

Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

```
wied        ds.b  1                 * Anzahl der Wiederholungen
re1         equ   *

*__*** Programm Stack

            ds.b  $db-re1           * Stack von re1 bis stack
stack       ds.b  1                 * oberste Stackadresse
adr_iroc2   ds.b  3                 * Einsprungadresse fuer OC2 Interrupts
*__                                 * muß auf $dc stehen !!!

            text
            ORG   PROG
*__*** Programminitialisierung ****
main        equ   *
*__         LDS   #stack            * Stackbereich ab RAM-Adresse $DB

#include    <IF_stack.icl>
#include    <IF_Imon.icl>           * SCI, IRQs, ... Initialisieren

            LDAA  #$7E              * OC2 Interruptadresse eintragen
            STAA  <adr_iroc2        * OPC vom JMP speichen!
            LDX   #frq_ir
            STX   <adr_iroc2+1      * ADR vom JMP speichern!
            LDX   hifrequenz        * Frequenz initialisieren
            STX   <adr_nfrq
            ldaa  #1
            staa  <mode
            LDX   #melanf           * Melodiezeiger initialisieren
            PSHX
            LDX   #REG
            BSET  $40,otmsk1,X      * OC2 Interrupts gestatten
            BSET  $40,otctl1,X      * TOGGLE - Funktion fuer OC2
            CLI                     * Freigabe maskierter Interrupts
*__*** Hauptprogramm ****
start       LDX   #REG              * Anfang des Steuerregisterblocks

inloop      equ   *

#include    <IF_mtst.icl>           * Rücksprung zum MONI11m

            brclr 2,oporta,x,gong   * Wohnungstür?
            brset 4,oporta,x,inloop * Haustür?

*__*** Klingel an der Haustür wurde betätigt

            PULX                    * Notenzeiger laden
            bsr   play
            PSHX                    * Notenzeiger retten
t_aus       ldd   hifrequenz
            std   <adr_nfrq         * Pausenfrequenz einstellen
            BRA   start             * Melodieende - Zurück zum Anfang

*__*** Klingel an der Wohnungstür wurde betätigt

gong        ldx   #T_gong
            ldaa  0,x
            staa  <wied
gong1       pshx
            bsr   play
            pulx
            dec   wied
            bne   gong1
            bra   t_aus
```

```
*__*** Unterprogramme ****

*__*** einen Song spielen

play        LDAA  1,X              * Tempo der Melodie laden
            STAA  <tempo
            INX
replay      INX
            LDY   #length          * Anfang der Längentabelle
            LDAA  <tempo           * Faktor für rel. Notenlänge
            LDAB  0,X              * relative Notenlänge ermitteln
            cmpb  #$E8             * Kommando von E8 ... EF
            bcs   no_kom
            stab  <mode
            clrb                   * 1/16 für Modeumschaltung
no_kom      ANDB  #$07             * nur Nr. der rel. Notenlänge
            ABY                    * Pos. rel. Notenlänge in IY
            LDAB  0,Y              * rel. Notenlänge in B laden
            MUL                    * = Faktor * rel. Notenlänge
            STD   <laenge          * = abs. Notenlänge
            LDY   #freqs-2         * Anfang der Frequenztabelle
            LDAB  0,X              * Notenfrequenz ermitteln
            cmpb  #$E8             * Kommando von E8 ... EF
            bcc   no_ton           * Frequenz nicht ändern!
            ANDB  #$F8             * nur Notennummer
            beq   hi_ff            * Pause
            LSRB
            LSRB                   * Frequenzen sind 16 Bitwerte
            ABY
            LDY   0,Y
            clra
            ldab  <mode
            andb  #3               * nur Faktor für Frequenz
            xgdy
            iny
mul_ff      dey
            beq   en_ff
            lsrd
            bra   mul_ff
hi_ff       ldd   0,Y
en_ff       STD   <adr_nfrq        * Frequenz einstellen
no_ton      BSR   nlgt             * Note spielen
            LDAA  #$F0             * Melodie - Blockende erreicht?
            CMPA  1,X
            BHI   replay           * sonst, nächste Note ausgeben
            BEQ   noblockend       * bei Blockende, Zeiger auf
            LDX   #melanf-1        * Blockanfang
noblockend  INX
            rts

*__*** UP - Note spielen **
nlgt        LDY   #REG
            LDD   <laenge
wait        BRCLR $40,otflg2,Y,wait * Note abzählen (warten auf RTI)
            BSET  $40,otflg2,Y
            SUBD  #10
            BGT   wait
            brclr 4,mode,nostc     * Stacato ist aus
            LDY   hifrequenz       * Ton aus
            STY   <adr_nfrq
nostc       LDY   #$0500           * kurze Pause zwischen 2 Noten
pause       DEY
            BNE   pause
            RTS                    * UP Ende
```

```
*__*** IR - Frequenz **
frq_ir      LDX   #REG              * Anfang des Steuerregisterblocks
            LDD   <adr_nfrq         * neuen Interrupt Zeitpunkt
            ADDD  otoc2,X           * bestimmen
            STD   otoc2,X           * ...
            BSET  $40,otflg1,X      * Interruptflag wieder löschen
            RTI                     * IR - Ende

endpg       equ   *

            DATA
            org   endpg

*__*** Daten ****

hifrequenz  DC.W  $0040             * Pausenfrequenz 31.25 kHz
*__               A    Ais  H    C0   Cis0 D0   Dis0
freqs       DC.W  9092,8582,8100,7645,7216,6811,6429
*__               E0   F0   Fis0 G0   Gis0 A0   Ais0
            DC.W  6068,5728,5406,5103,4816,4546,4291
*__               H0   C1   Cis1 D1   Dis1 E1   F1
            DC.W  4050,3823,3608,3406,3215,3034,2864
*__               Fis1 G1   Gis1 A1   Ais1 H1   C2
            DC.W  2703,2551,2408,2273,2145,2025,1911

*__ relative      1  3  1  3  1  3  1  3
*__ Notenlängen   -- -- -  -- -  -  -  -
*__ Tabelle       16 32 8  16 4  8  2  4

length      DC.B  10,15,20,30,40,60,80,120

*__         1.Byte      =     Anfangskennzeichen
*__         2.Byte      =     Faktor für relative Notenlänge
*__         3.  ...     =     Noten- und Notenlängen-nummern
*__   Bitaufteilung     =     nnnn nlll
*__         nnnnn       =     0 ... 28 = Pos. in Frequenzliste
*__                     =     29 in lll steht ein Kommando Zeitdauer = 1/16
*__                           Ton wie vorhergehender. Das höchste Bit von lll
*__                           schaltet Stacato ein/aus die unteren beiden Bit
*__                           sind Faktor für Frequenzen -1. 001 ist Default!
*__         lll         =     0 ... 7  = Pos. in Längentabelle


*__ Song für Wohnungstür

T_gong      dc.b  6                 * Anzahl der Wiederholungen
            dc.b  6                 * Tempo

            dc.b  $eb,$bf,2,$87,0,$e9
            dc.b  $f0

*__ Songs für Haustür

*__ Song 1 C-Dur Sonate KV545
melanf      DC.B  $F0,$18
            DC.B  $86,$A4,$BC,$7D,$80,$90,$84,$04,$CE,$BC,$E4,$BC
            DC.B  $AA,$A0,$A8,$A4

*__ Song 2 A-Dur Sonate KV331
            DC.B  $F0,$25
            DC.B  $8B,$90,$8A,$A4,$A2,$7B,$88,$7A,$94,$92,$6C,$7A
            DC.B  $8C,$92,$8C,$7A,$6C
```

```
*__ Song 3 Alla Turka KV331
            DC.B  $F0,$20
            DC.B  $6A,$60,$68,$82,$02,$82,$78,$80,$A2,$02,$A2,$98
            DC.B  $A0,$CA,$C0,$C8,$CA,$C0,$C8,$E5

*__ Song 4 Glory Glory Hallejulia
            dc.b  $f0,$25
            dc.b  $5B,$48,$40,$58,$80,$30,$44,$22,$80,$20,$6B,$78
            DC.B  $20,$78,$80,$68,$5C,$44,$58,$02,$48,$40,$58,$20
            DC.B  $30,$42,$30,$00,$20,$00,$80,$20,$32,$00,$30,$20
            DC.B  $00,$18,$00,$25

*__ Song 5
            dc.b  $F0,$25
            DC.B  $22,$32,$42,$20,$00,$22,$32,$42,$20,$00,$42,$4A
            DC.B  $5B,$00,$42,$4A,$5B,$00,$58,$68,$58,$48,$42,$20
            DC.B  $80,$58,$68,$58,$48,$40,$00,$22,$82,$5A,$83,$00
            DC.B  $82,$5A,$83

*__ Song 6
            dc.b  $F0,$25
            DC.B  $20,$40,$48,$5B,$EA,$00
            DC.B  $20,$40,$48,$5B,$E9,$00,$20,$40
            DC.B  $48,$5A,$42,$22,$40,$00,$40,$33,$00,$30,$40,$30
            DC.B  $22,$00,$80,$42,$58,$00,$58,$4B,$00,$48,$40,$48
            DC.B  $58,$00,$42,$22,$32,$25

melende     DC.B  $FF
*
* Ende von Melgen.a
```

Die verwendeten Includefiles werden auf den folgenden Seiten dargestellt:

1. Das File IF_eqmon.icl

```
*
* START_ wird dem PREPROCESSOR übergeben und ist gleich der
* Systemvariable start.11 die durch TARGET.BAT eingestellt wird!
*
#if         START_-2000

* Programm läuft nicht im RAM

#else
*
* Nur wenn Programm im RAM abgearbeitet wird
*
#include    <Usr_rufe.icl>          * Monitorrufe
*
#endif
*
```

2. Das File IF_stack.icl

```
*
* START_ wird dem PREPROCESSOR übergeben und ist gleich der
* Systemvariable start.11 die durch TARGET.BAT eingestellt wird!
*
#if         START_-2000

* Programm läuft nicht im RAM

            lds   #stack            * Marke stack aus Speichersegment
```

```
#else
*
* Nur wenn Programm im RAM abgearbeitet wird
*
            lds   #PROG-1
*
#endif
*
```

3. Das File IF_Imon.icl

```
*
* START_ wird dem PREPROCESSOR übergeben und ist gleich der
* Systemvariable start.11 die durch TARGET.BAT eingestellt wird!
*
#if         START_-2000

* Programm läuft nicht im RAM

#else
*
* Nur wenn Programm im RAM abgearbeitet wird
*
            jsr   inimon            * Initialisierung vom RS232, IRQs, ...
*
#endif
*
```

4. Das File IF_mtst.icl

```
*
* START_ wird dem PREPROCESSOR übergeben und ist gleich der
* Systemvariable start.11 die durch TARGET.BAT eingestellt wird!
*
#if         START_-2000

* Programm läuft nicht im RAM

#else
*
* Nur wenn Programm im RAM abgearbeitet wird
*
            jsr   intest            * Eingabe vorhanden?
            beq   weiter            * NEIN
            jsr   scin              * warten auf eine Eingabe
            cmpa  #$0d              * ENTER
            beq   moni11m
            cmpa  #$1b              * ESC
            bne   weiter
            swi                     * Rücksprung in MONI11m
            bra   weiter
moni11m     ldx   $fffe             * RESET-Vektor im Expandet Mode
            jmp   0,x
weiter      equ   *
*
#endif
*
```

5. Das File Usr_rufe.icl

```
*
*        Userrufe für den MONI11m
*
*        die Basisadresse der Datenzellen steht auf:
```

Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

```
*
baseadr     equ   $ffbe
*
*        Offsets zur Basisadresse der Datenzellen
*
word1       equ   0
word2       equ   2
word3       equ   4
now1        equ   6
asave       equ   7
upper       equ   8
echo        equ   9
fromptr     equ   10
combuf      equ   11
*
*        Kaltstart des Monitor
*
moni        equ   $ffbb
*
*        Unterprogramme aus dem MONI11m     (=!   steht für ungleich)
*
inimon      equ   $ffb8     SCI,Arbeiszellen,I-Vec.
scin        equ   $ffb5     Zeichen vom SCI
scout       equ   $ffb2     Zeichen zum SCI
inchar      equ   $ffaf     Zeichen vom SCI oder 0,x (fromptr=0 oder =!0)
*__                         mit oder ohne Echo       (echo=!0   oder =0 )
*__                         mit oder ohne Groß/Klein (upper=!0  oder =0 )
inbuf       equ   $ffac
outtext     equ   $ffa9
crlf        equ   $ffa6
space       equ   $ffa3
space2      equ   $ffa0
space4      equ   $ff9d
space5      equ   $ff9a
intest      equ   $ff97
out2hex     equ   $ff94
outx        equ   $ff91
outy        equ   $ff8e
getw1       equ   $ff8b
get2hex     equ   $ff88
getpar      equ   $ff85
skipblk     equ   $ff82
skipcha     equ   $ff7f
*
*        Ende Monitorrufe
```

Damit ergeben sich die folgenden Ausschnitte aus dem Listfile Melgen2.lst:

```
file: melgen2.a

   1:   ************************************************
   2:   *  Melodiegeneratorprogramm fuer den MC68HC11A1  *
   3:   *  - Entwurf  : Heiko Boettcher                  *
   4:   *               Meisenweg 1, 01589 Riesa         *
   5:   *  - Merkamle : 3 Oktaven Tonumfang mit Pausen   *
   6:   *               Notenlaengen von 1/16 bis 3/4    *
   7:   *               Tempo jedes Liedes einstellbar   *
   8:   *  - Erweitert: Dr. Siegmar Schöne               *
   9:   *  - neues    : Stacato abschaltbar, Frequenz-   *
  10:   *               bereich kann um 4 Oktaven ver-   *
  11:   *               schoben werden. 3 neue Songs.    *
  12:   *               2. Klingeltaster                 *
  13:   ************************************************
  14:
  15:   *      übersetzt auf Adresse $2000
  16:
  17:
  18:
```

Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

```
  19:  ******************** Konstanten ****
file: e:\prak\zwerg11p\lib.a\hc11.h

  16:       0000b600  EEPROM         equ     $b600   * eeprom area
  17:
  18:
  19:       00001000  REG            equ     $1000   * register base
  20:
  21:       00001000  porta          equ     REG+$00 * port a data reg.
  22:       00001002  pioc           equ     REG+$02 * parallel i/o control reg..

              • • •

140:       0000003f  oconfig        equ     $3f     * configuration control reg.
file: e:\prak\zwerg11p\lib.a\bit.h

   2:  * Austausch der binären Operanden durch hexadezimale Operanden
   3:
 278:                                list
file: e:\prak\zwerg11p\lib.a\IF_eqmon.icl

   1:  *
   2:  * 2000 wird dem PREPROCESSOR übergeben und ist gleich der
   3:  * Systemvariable start.11 die durch TARGET.BAT eingestellt wird!
   4:  *
   5:
   6:
   7:
   8:
   9:
  10:  *
  11:  * Nur wenn Programm im RAM abgearbeitet wird
  12:  *
file: e:\prak\zwerg11p\lib.a\Usr_rufe.icl

   1:  *
   2:  *         Userrufe für den MONI11m
   3:  *
   4:  *         die Basisadresse der Datenzellen steht auf:
   5:  *
   6:       0000ffbe  baseadr        equ     $ffbe
   7:  *
   8:  *         Offsets zur Basisadresse der Datenzellen
   9:  *
  10:       00000000  word1          equ     0
  11:       00000002  word2          equ     2
  12:       00000004  word3          equ     4
  13:       00000006  now1           equ     6
  14:       00000007  asave          equ     7
  15:       00000008  upper          equ     8
  16:       00000009  echo           equ     9
  17:       0000000a  fromptr        equ     10
  18:       0000000b  combuf         equ     11
  19:  *
  20:  *         Kaltstart des Monitor
  21:  *
  22:       0000ffbb  moni           equ     $ffbb
  23:  *
  24:  *         Unterprogramme aus dem MONI11m     (=!   steht f r ungleich)
  25:  *
  26:       0000ffb8  inimon         equ     $ffb8   SCI,Arbeiszellen,I-Vec.
  27:       0000ffb5  scin           equ     $ffb5   Zeichen vom SCI
  28:       0000ffb2  scout          equ     $ffb2   Zeichen zum SCI
  29:       0000ffaf  inchar         equ     $ffaf   Zeichen vom SCI oder 0,x (fromptr=0 oder =!0)
  30:  *****************                            mit oder ohne Echo       (echo=!0   oder =0 )
  31:  *****************                            mit oder ohne Groá/Klein (upper=!0  oder =0 )
  32:       0000ffac  inbuf          equ     $ffac
  33:       0000ffa9  outtext        equ     $ffa9
  34:       0000ffa6  crlf           equ     $ffa6
  35:       0000ffa3  space          equ     $ffa3
  36:       0000ffa0  space2         equ     $ffa0
  37:       0000ff9d  space4         equ     $ff9d
  38:       0000ff9a  space5         equ     $ff9a
  39:       0000ff97  intest         equ     $ff97
  40:       0000ff94  out2hex        equ     $ff94
  41:       0000ff91  outx           equ     $ff91
  42:       0000ff8e  outy           equ     $ff8e
```

Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

```
  43:        0000ff8b  getw1          equ     $ff8b
  44:        0000ff88  get2hex        equ     $ff88
  45:        0000ff85  getpar         equ     $ff85
  46:        0000ff82  skipblk        equ     $ff82
  47:        0000ff7f  skipcha        equ     $ff7f
  48:   *
  49:   *          Ende Monitorrufe
file: e:\prak\zwerg11p\lib.a\IF_eqmon.icl

  14:   *
  15:
  16:   *
file: melgen2.a

  24:        00000018  otoc2          EQU     otoc2h          * Offset des toc2-Register zu REG
  25:
  26:   ******************** RAM-Belegung
  27:                                 bss
  28:        00000000                 ORG     $0000
  29: 0000   00000001 adr_nfrq        ds.w    1               * Ablageplatz der Notenfrquenz
  30: 0002   00000001 mode            ds.b    1               * Bit 2 Stacato 10 F-Faktor
  31: 0003   00000001 tempo           ds.b    1               * Songtempo (Faktor f r Notenl„nge)
  32: 0004   00000001 laenge          ds.w    1               * Notenl„nge
  33: 0006   00000001 wied            ds.b    1               * Anzahl der Wiederholungen
  34:        00000007  re1            equ     *
  35:
  36:   ******************** Programm Stack
  37:
  38: 0007   000000d4                 ds.b    $db-re1         * Stack von re1 bis stack
  39: 00db   00000001 stack           ds.b    1               * oberste Stackadresse
  40: 00dc   00000003 adr_iroc2       ds.b    3               * Einsprungadresse fuer OC2 Interrupts
  41:   *****************                                     * muß auf $dc stehen !!!
  42:
  43:                                 text
  44:         00002000                ORG     $2000
  45:   ******************** Programminitialisierung ****
  46:         00002000 main           equ     *
  47:   *****************             LDS     #stack          * Stackbereich ab RAM-Adresse $DB
file: e:\prak\zwerg11p\lib.a\IF_stack.icl

   1:   *
   2:   * 2000 wird dem PREPROCESSOR übergeben und ist gleich der
   3:   * Systemvariable start.11 die durch TARGET.BAT eingestellt wird!
   4:   *
file: e:\prak\zwerg11p\lib.a\IF_stack.icl

  12:   *
  13:   * Nur wenn Programm im RAM abgearbeitet wird
  14:   *
  15: 2000  8e1fff                    lds     #$2000-1
  16:   *
  17:
  18:   *
file: e:\prak\zwerg11p\lib.a\IF_Imon.icl

   1:   *
   2:   * 2000 wird dem PREPROCESSOR übergeben und ist gleich der
   3:   * Systemvariable start.11 die durch TARGET.BAT eingestellt wird!
   4:   *
   5:
   6:
   7:
   8:
   9:
  10:   *
  11:   * Nur wenn Programm im RAM abgearbeitet wird
  12:   *
  13: 2003  bdffb8                    jsr     inimon          * Initialisierung vom RS232,IRQs, ...
  14:   *
  15:
  16:   *
file: melgen2.a

  52: 2006  867e                      LDAA    #$7E                    * OC2 Interruptadresse eintragen
  53: 2008  97dc                      STAA    <adr_iroc2              * OPC vom JMP speichen!
  54: 200a  ce20e8                    LDX     #frq_ir
  55: 200d  dfdd                      STX     <adr_iroc2+1            * ADR vom JMP speichern!
  56: 200f  fe20f5                    LDX     hifrequenz              * Frequenz initialisieren
```

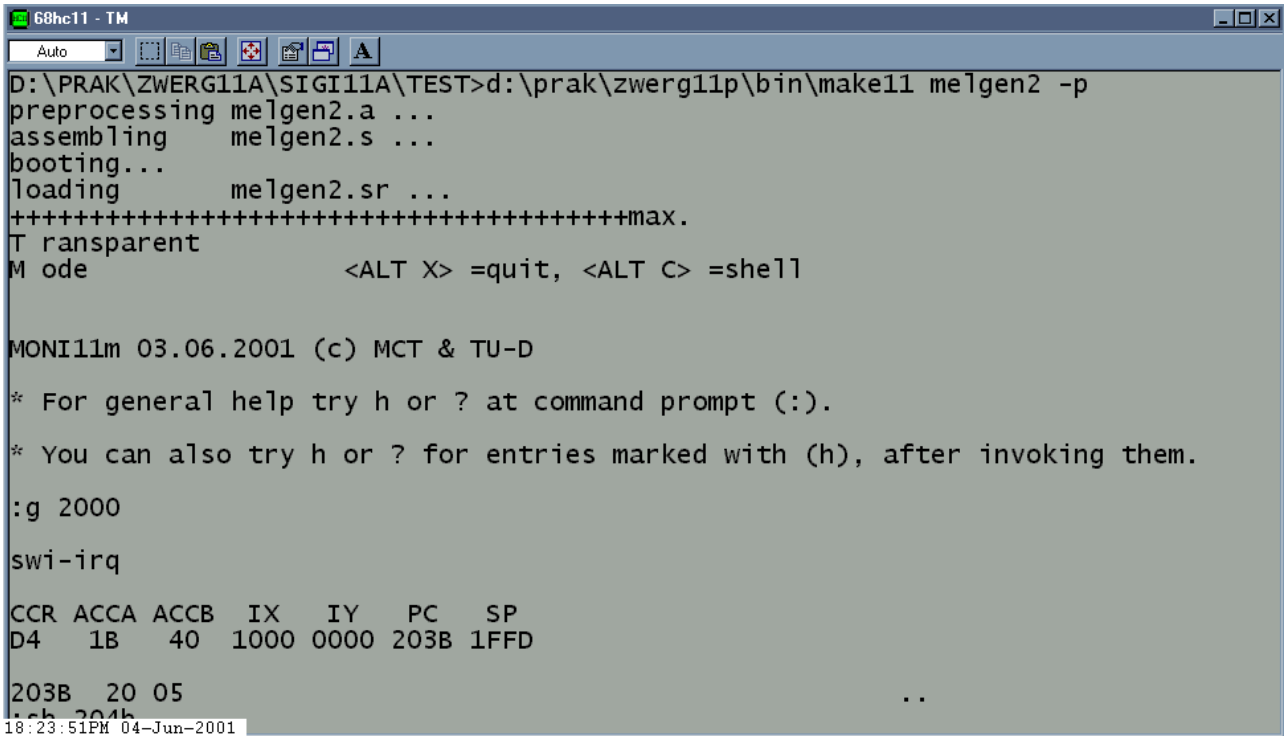
Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

```
  57: 2012  df00                    STX     <adr_nfrq
  58: 2014  8601                    ldaa    #1
  59: 2016  9702                    staa    <mode
  60: 2018  ce2140                  LDX     #melanf             * Melodiezeiger initialisieren
  61: 201b  3c                      PSHX
  62: 201c  ce1000                  LDX     #REG
  63: 201f  1c2240                  BSET    $40,otmsk1,X        * OC2 Interrupts gestatten
  64: 2022  1c2040                  BSET    $40,otctl1,X        * TOGGLE - Funktion fuer OC2
  65: 2025  0e                      CLI                         * Freigabe maskierter Interrupts
  66:   ******************** Hauptprogramm ****
  67: 2026  ce1000    start         LDX     #REG                * Anfang des Steuerregisterblocks
  68:
  69:         00002029 inloop       equ     *
file: e:\prak\zwerg11p\lib.a\IF_mtst.icl

   1:   *
   2:   * 2000 wird dem PREPROCESSOR übergeben und ist gleich der
   3:   * Systemvariable start.11 die durch TARGET.BAT eingestellt wird!
   4:   *
   5:
   6:
   7:
   8:
   9:
  10:   *
  11:   * Nur wenn Programm im RAM abgearbeitet wird
  12:   *
  13: 2029  bdff97                  jsr     intest          * Eingabe vorhanden?
  14: 202c  2714                    beq     weiter          * NEIN
  15: 202e  bdffb5                  jsr     scin            * warten auf eine Eingabe
  16: 2031  0f                      sei
  17: 2032  810d                    cmpa    #$0d            * ENTER
  18: 2034  2707                    beq     moni11m
  19: 2036  811b                    cmpa    #$1b            * ESC
  20: 2038  2608                    bne     weiter
  21: 203a  3f                      swi                     * Rücksprung in MONI11m
  22: 203b  2005                    bra     weiter
  23: 203d  fefffe    moni11m       ldx     $fffe           * RESET-Vektor im Expandet Mode
  24: 2040  6e00                    jmp     0,x
  25:         00002042 weiter       equ     *
  26: 2042  0e                      cli
  27:   *
  28:
  29:   *
file: melgen2.a

  73: 2043  1f00020f                brclr   2,oporta,x,gong     * Wohnungstür?
  74: 2047  1e0004de                brset   4,oporta,x,inloop   * Haustür?
  75:
  76:   ******************** Klingel an der Haustür wurde betätigt
  77:
  78: 204b  38                      PULX                        * Notenzeiger laden
  79: 204c  8d1a                    bsr     play
  80: 204e  3c                      PSHX                        * Notenzeiger retten
  81: 204f  fc20f5    t_aus         ldd     hifrequenz
  82: 2052  dd00                    std     <adr_nfrq           * Pausenfrequenz einstellen
  83: 2054  20d0                    BRA     start               * Melodieende - Zur ck zum Anfang
  84:
  85:   ******************** Klingel an der Wohnungstür wurde betätigt
  86:
  87: 2056  ce2137    gong          ldx     #T_gong
  88: 2059  a600                    ldaa    0,x
  89: 205b  9706                    staa    <wied
  90: 205d  3c        gong1         pshx
  91: 205e  8d08                    bsr     play
  92: 2060  38                      pulx
  93: 2061  7a0006                  dec     wied
  94: 2064  26f7                    bne     gong1
  95: 2066  20e7                    bra     t_aus
  96:
  97:   ******************** Unterprogramme ****
  98:
  99:   ******************** einen Song spielen
```

• • •

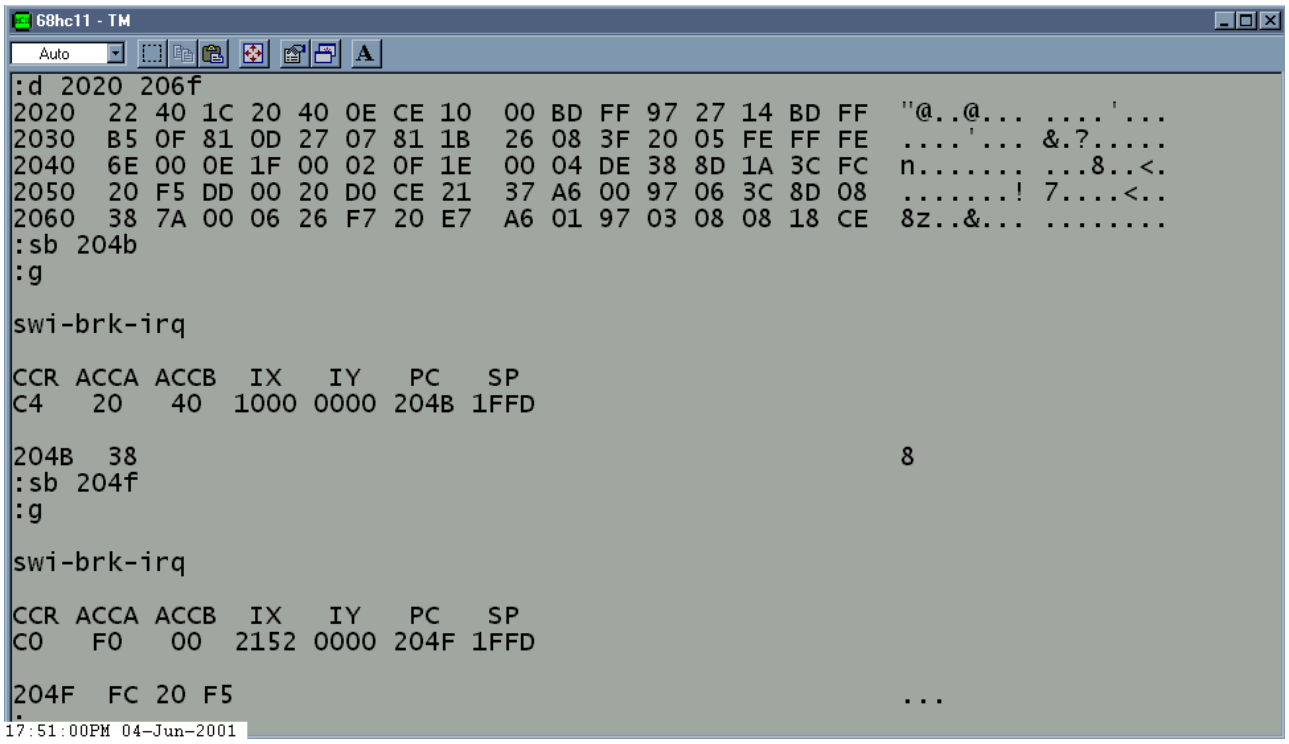
Ein solches Listfile entsteht automatisch beim start des Assemblersystems durch einen Doppelklick auf das Quellfile im Norton-Comander oder durch die Eingabe des Kommandos ”make11.bat melgen2 -p” am DOS-Prompt. Wenn die Übersetzung fehlerfrei war, wird der in dem File ”melfen2.sr” abgelegte Programmcode in den RAM ab Adresse $2000 gespeichert und bei fehlerfreier Übertragung auch gestartet. Am ZWERG11plus muß bei der Verwendung des MONI11m, der sich im Externen EEPROM befindet, nur der Jumper 3 geschlossen sein! Der Jumper 1 braucht nicht geschlossen zu sein. Da beim laden des Programms die Daten im Speicher von $0000 bis $00ff zerstört werden, muß vor einem 1. Rücksprung in den Monitor der Monitor neu initialisiert werden. Das erfolgt durch den Aufruf des Unterprogramm „inimon“ oder wie im Beispiel durch Betätigen der ENTER- Taste und dem damit erfolgten Neustart des MONI11m. Sobald sich der Monitor gemeldet hat ist nun ein Neustart des Programm ab Adresse $2000 mit dem Kommando ”g 2000” möglich. Wird nun am PC die Taste ESC betätigt, so verzweigt das Programm zu dem SWI Befehl und der Monitor meldet sich mit einem SWI-Interupt. Eine Fortsetzung des Programm ist nun einfach durch das Kommando ”g” möglich, da durch dem SWI alle CPU-Zustände gerettet wurden. Vorher können aber beliebige Veränderungen im Speicher, in den Registern oder das Setzen eines Unterbrechungspunktes erfolgen.

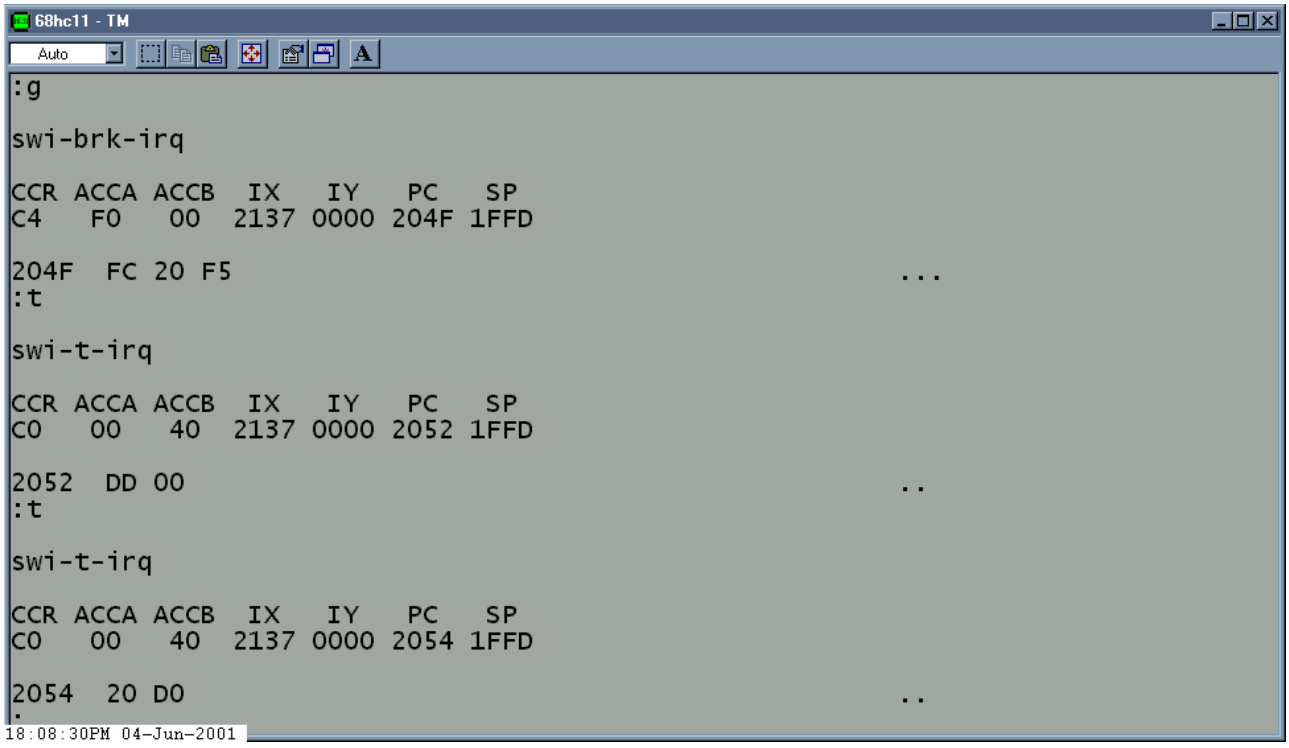
Das folgende Bild zeigt den Ablauf bis zur Betätigung von ENTER



Nun kann das Programm mit ”g 2000” wieder gestartet werden und nach der Betätigung von ESC wird der SWI Interrupt ausgeführt. Dabei werden alle Registerinhalte angezeigt. Dieser Softwareinterrupt auf der Adresse $203B ist auch noch auf dem oberen Bild zu sehen. Danach wird auf den folgenden Bildern der Speicherinhalt von $2020 bis $206f angezeigt und anschließend ein Breakpoint auf den Befehl ”**pulx**” auf der Adresse $204b gesetzt und das Programm fortgesetzt. Nach dem Erreichen dieses Breakpoint durch ein Betätigen des Haustürtasters wird dann ein neuer auf den Befehl auf der Adresse $204f ”**t_aus ldd hifrequenz**” gesetzt und das Programm erneut fortgesetzt. Beim erreichen dieses Breakpoint wird dann zuerst das Programm mit dem Kommando „g“ fortgesetzt. Durch betätigen des Tasters für die Wohnungstür wird dieser Breakpoint erneut erreicht. Diesmal wird 2 mal ein Einzelschritt mit dem Kommando „T“ ausgeführt. So läßt sich der Weg des Programms mit den Veränderungen in den Registern und im Speicher verfolgen. Mit dem Kommando ”CB” kann dann der Breakpoint wieder gelöscht und das Programm wieder gestartet werden. Dann ist ein Anhalten des Programms nur mit der eingebauten Testhilfe durch das Betätigen von ESC möglich.

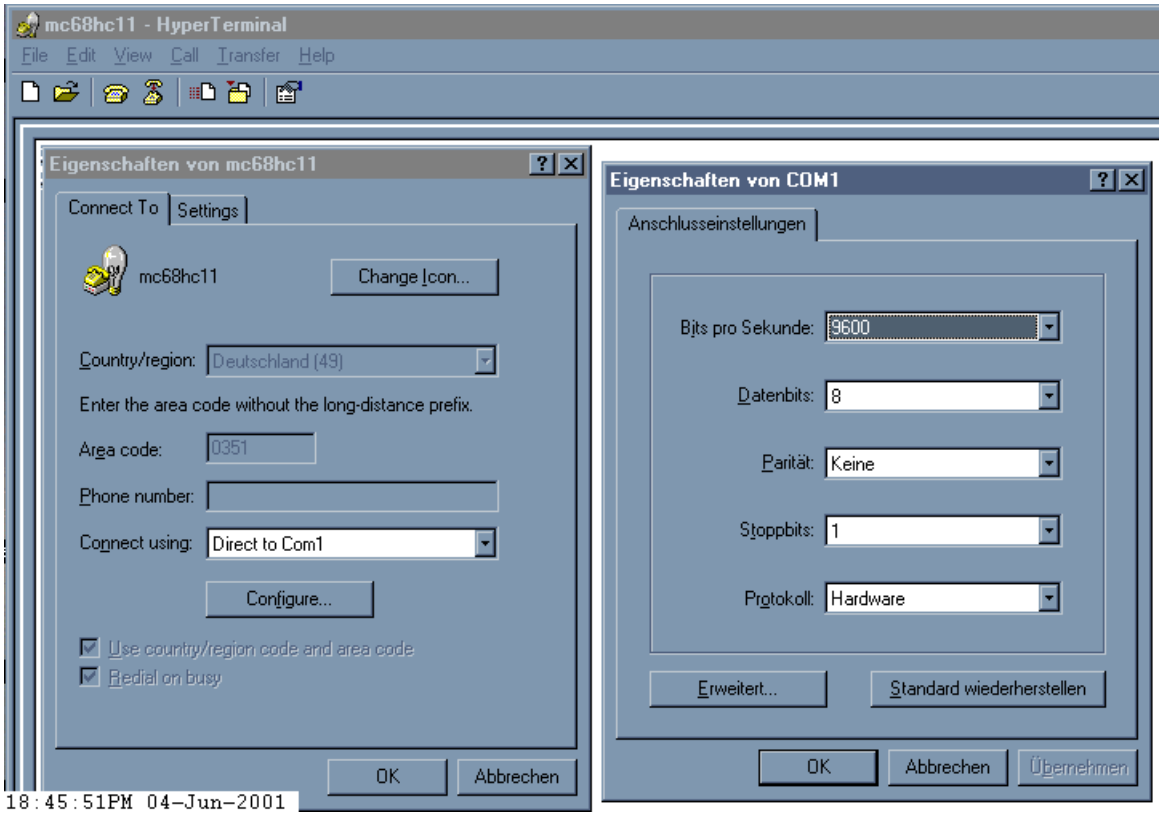
Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

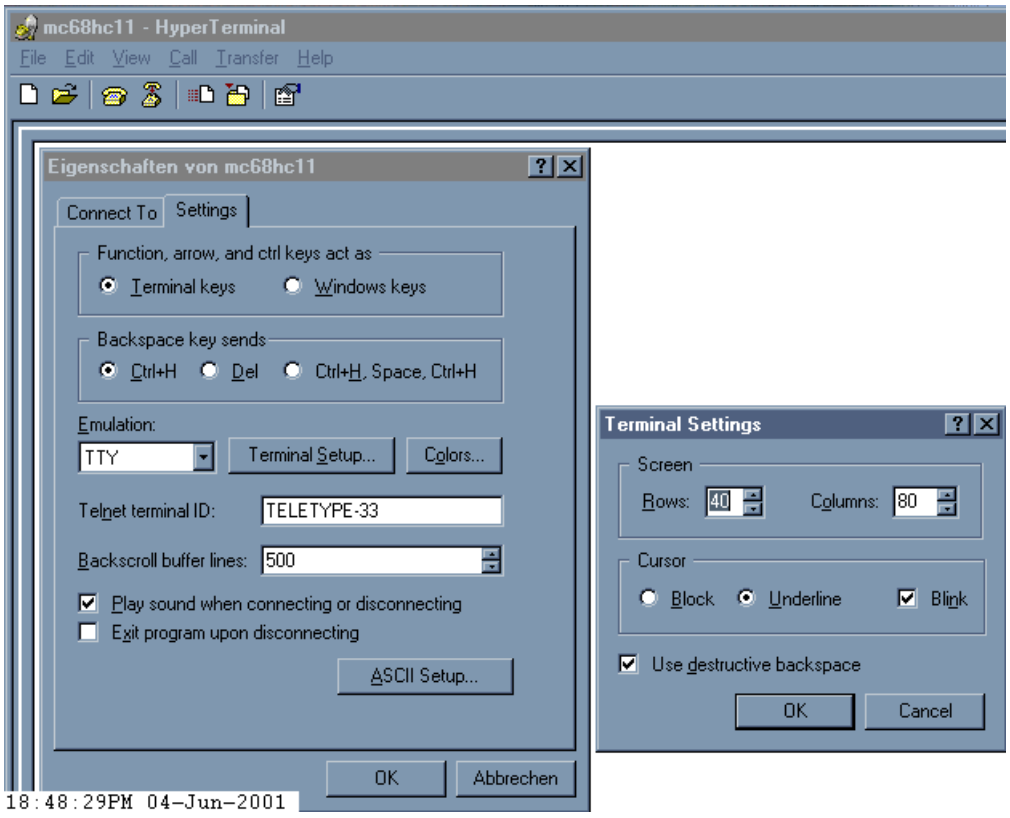




Diese Testung von 68HC11 Programmen kann aber auch mit dem Windowsprogramm ”Hyperterminal” durchgeführt werden. Die Einstellungen dazu sind im folgenden Bild zu sehen. Bei der Verwendung des ”Hyperterminal” sind der Jumper 1 und der Jumper 3 auf dem ZWERG11plus zu öffnen. Nach dem Einschalten des ZWERG11plus meldet sich dann der ”MONI11m”. Nach der Eingabe von ”h” werden alle Monitorkommandos angezeigt. Mit dem Kommando ”L” kann nun der Programmcode aus dem File ”melgen2.sr” geladen werden. Dazu muß nach der Eingabe des Kommandos ”L” im Menu ”Übertragung” der File ”melgen2.sr” als Textdatei gesendet werden. Bei einem erfolgreichen laden meldet sich dann der Monitor wieder mit einem ”ok” und hat in das PC-Register die Adresse aus dem Endeblock = $2000 geladen. Damit kann das Programm einfach durch die Eingabe des Kommandos ”g” gestartet werden. Eine Initialisierung des Monitors ist diesmal nicht notwendig, da der Monitor schon zum Laden des Programms benutzt wurde. Die weitere Testung erfolgt dann wieder wie oben bereits beschrieben.

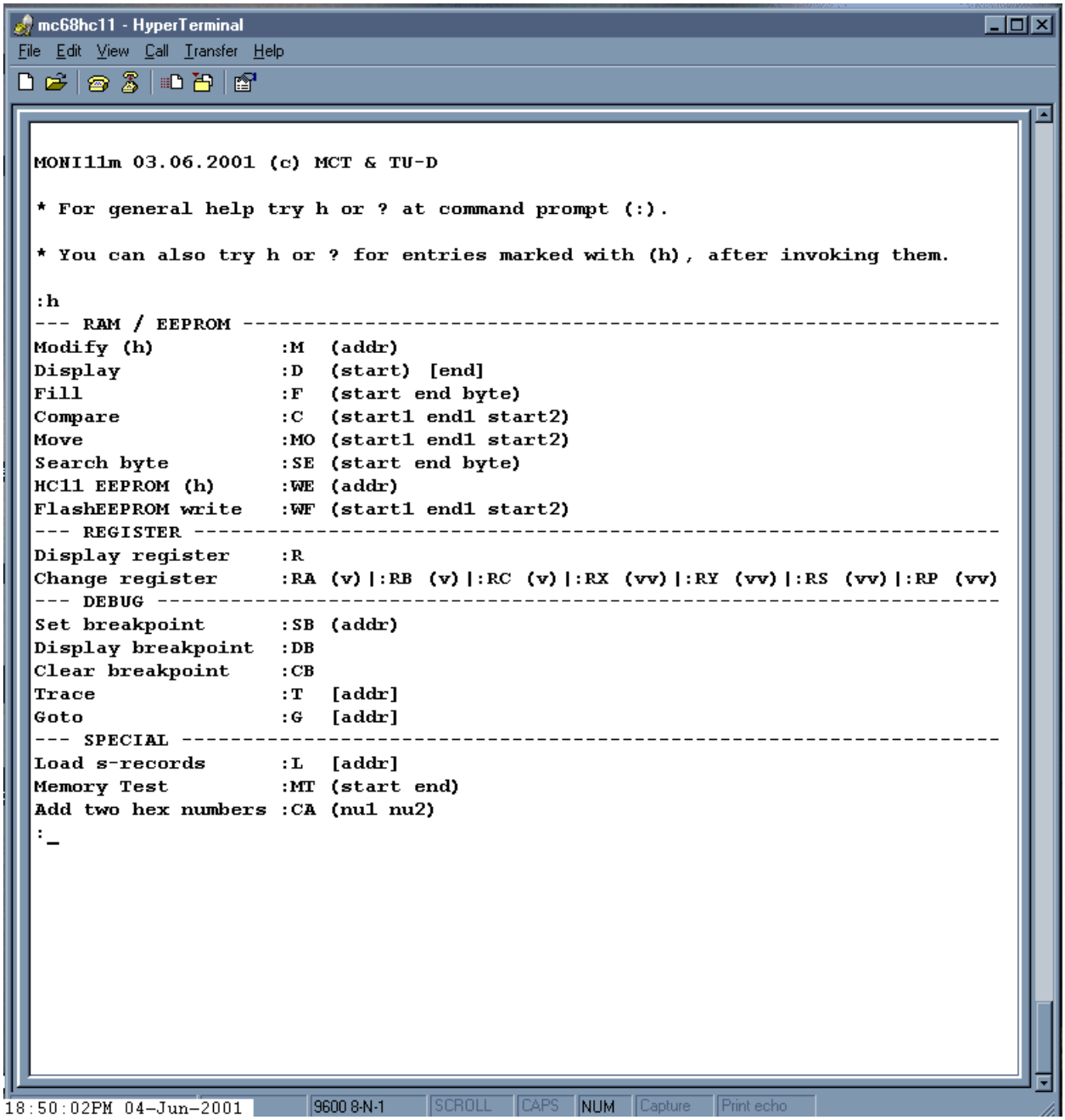
Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”



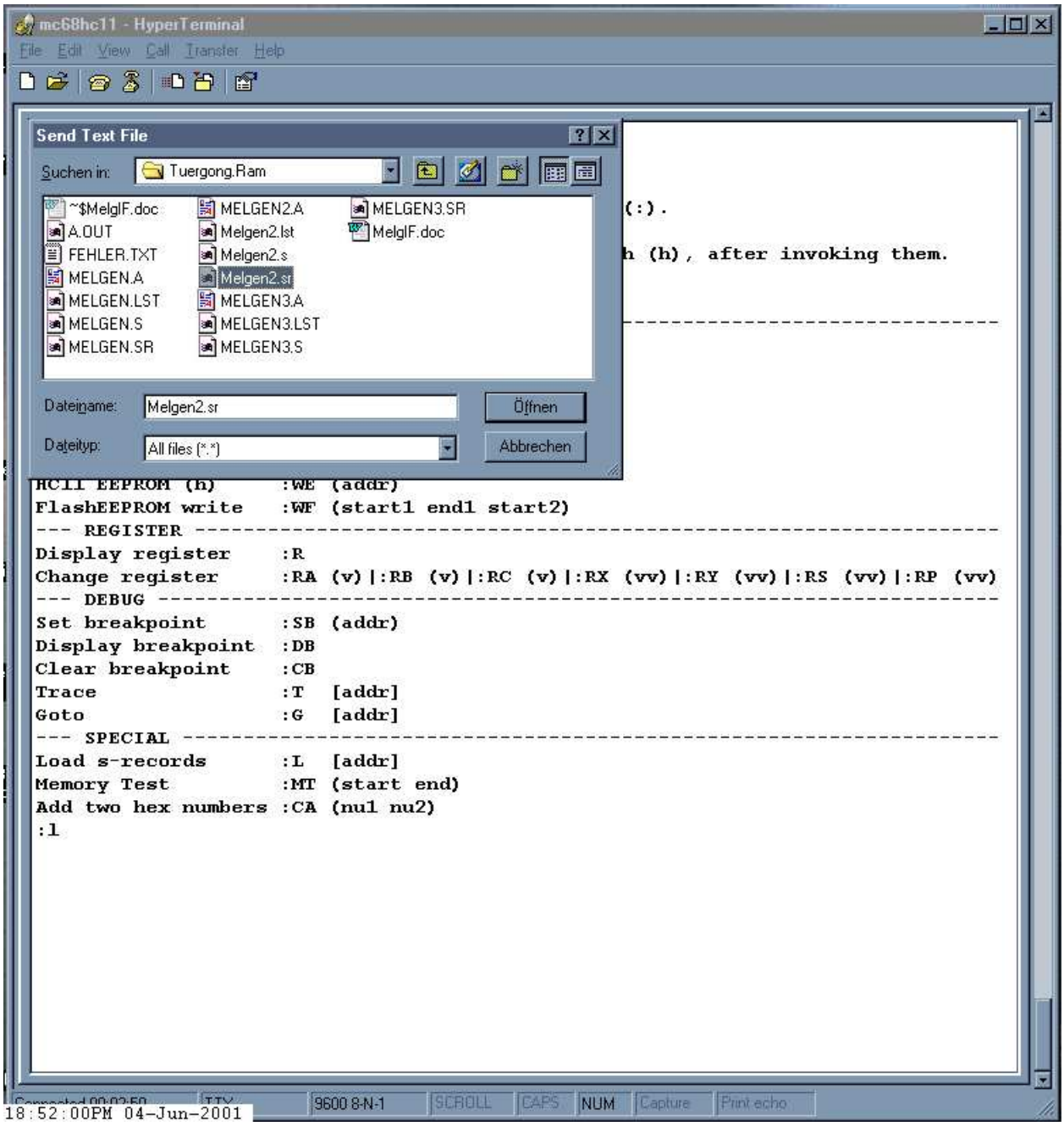


Die folgenden Bilder zeigen den Start des MONI11m mit seinem Hilfsmenu und das Laden von „melgen2.sr“.

Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

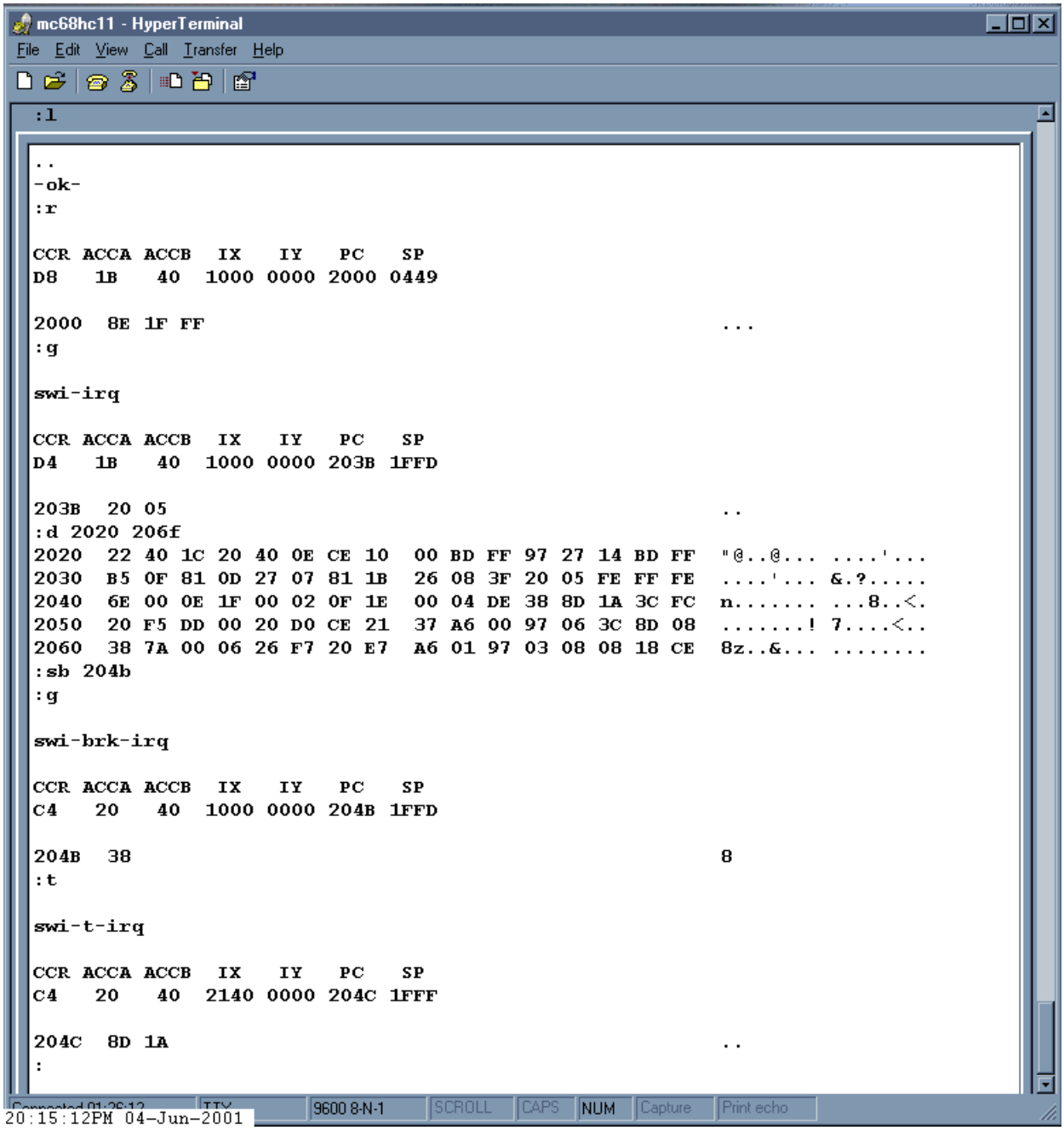


Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”



Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

Laden von „melgen2.sr“, Anzeige der Register, SWI-Interupt nach der Betätigung von ESC, Programmtestung durch setzen von Unterbrechungspunkten und Einzelschritt.



Im folgenden Programm wird die Benutzung der Unterprogramme des MONI11m gezeigt.

```
*
*  Programm zur Demonstration der USER-Rufe des MONI11m
*
*
#include     <target.h>
#include     <Usr_rufe.icl>
#include     "c:\datum.c"

             bss
             org    $0000

             ds.w   40
stack        equ    *
```

Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”

```
            text
            org    PROG

            lds    #stack

            jsr    inimon              * Initialisierung vom RS232, IRQs, ....

loop        ldx    #meld
            jsr    outtext             * Zeichenkette ausgeben

*       Register initialisieren

            ldd    #$aa55
            ldx    #$5aa5
            ldy    #$a55a

*       Register anzeigen

            jsr    crlf
            jsr    out2hex             * Akku A anzeigen
            jsr    space
            psha
            tba
            jsr    out2hex             * Akku B anzeigen
            pula
            jsr    space
            xgdx
            jsr    outx                * Akku D anzeigen
            xgdx
            jsr    space2
            jsr    outx                * IX anzeigen
            jsr    space2
            jsr    outy                * IY anzeigen
            jsr    crlf                * Neue Zeile

            ldx    #wait
            jsr    outtext             * Zeichenkette ausgeben

            jsr    scin                * Warten auf Eingabe

*       Inchar mit Echo und nur Großbuchstaben

            ldx    #inch1
            jsr    outtext             * Zeichenkette ausgeben

*       Mode für INCHAR einstellen

            ldx    baseadr
            inc    upper,x             * nur GROß einstellen
            inc    echo,x              * Echo freigeben

inzyk1      jsr    inchar
            jsr    scout               * 2. Echo!!
            cmpa   #$d                 * Ende bei ENTER
            bne    inzyk1


*       Inchar mit Echo und allen Buchstaben

            ldx    #inch2
            jsr    outtext             * Zeichenkette ausgeben

*       Mode für INCHAR einstellen

            ldx    baseadr
            clr    upper,x             * GROß und KLEIN einstellen

inzyk2      jsr    inchar
            jsr    scout               * 2. Echo!!
            cmpa   #$d                 * Ende bei ENTER
```

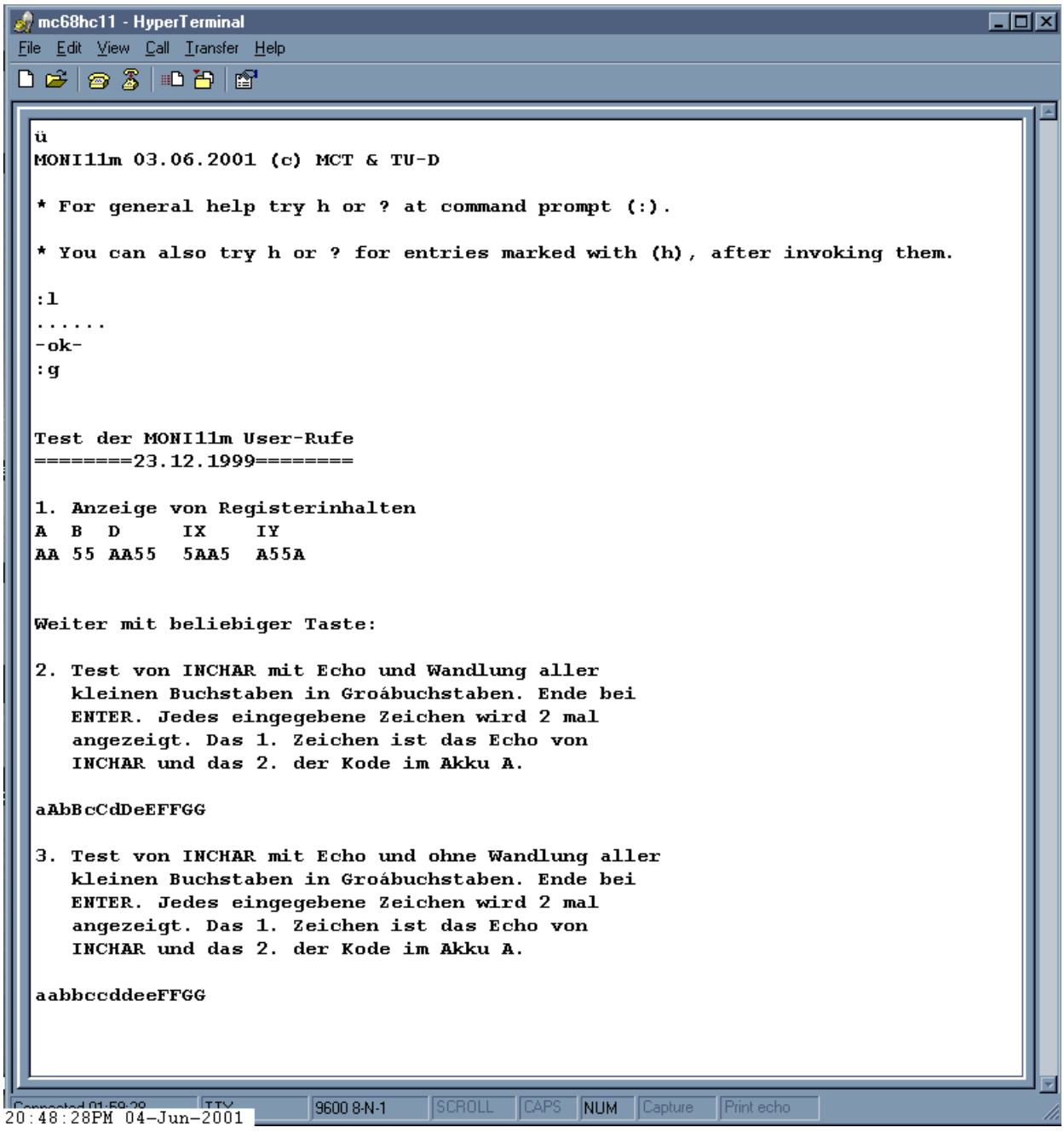
```
             bne    inzyk2
*
*         Test von GETW1
*
adrin        ldx    #inword
             jsr    outtext

             jsr    getw1
             bne    we_da

             ldx    #noin
             jsr    outtext
             bra    adrin

we_da        ldx    #wertda
             jsr    outtext

             ldx    baseadr
             ldx    word1,x
             jsr    outx
*
*         Test von GET2HEX
*
bytin        ldx    #byteein
             jsr    outtext

             ldx    baseadr
             inc    upper,x            * kleine Buchstaben in Groß wandeln

             jsr    get2hex
             bcc    byteda

             ldx    #errbyin
             jsr    outtext
             bra    bytin

byteda       ldx    #wertda
             jsr    outtext
             jsr    out2hex

             ldx    baseadr
             clr    upper,x
*
*         Test der Eingabe von mehreren Adressen mit gepufferter Eingabe
*
             ldx    #drei_adr
             jsr    outtext

             jsr    inbuf              * Eingabe in Puffer. Abschluß mit ENTER

             jsr    getpar

             ldx    #anz_adr
             jsr    outtext

             jsr    outy
             jsr    space
             ldx    baseadr
             ldy    word2,x
             jsr    outy
             jsr    space
             ldy    word1,x
             jsr    outy
             jsr    crlf

*****************************************************************

             jsr    crlf
```

Bedingte Übersetzung in Abhängigkeit von der Zieladresse für "melgen2.a"

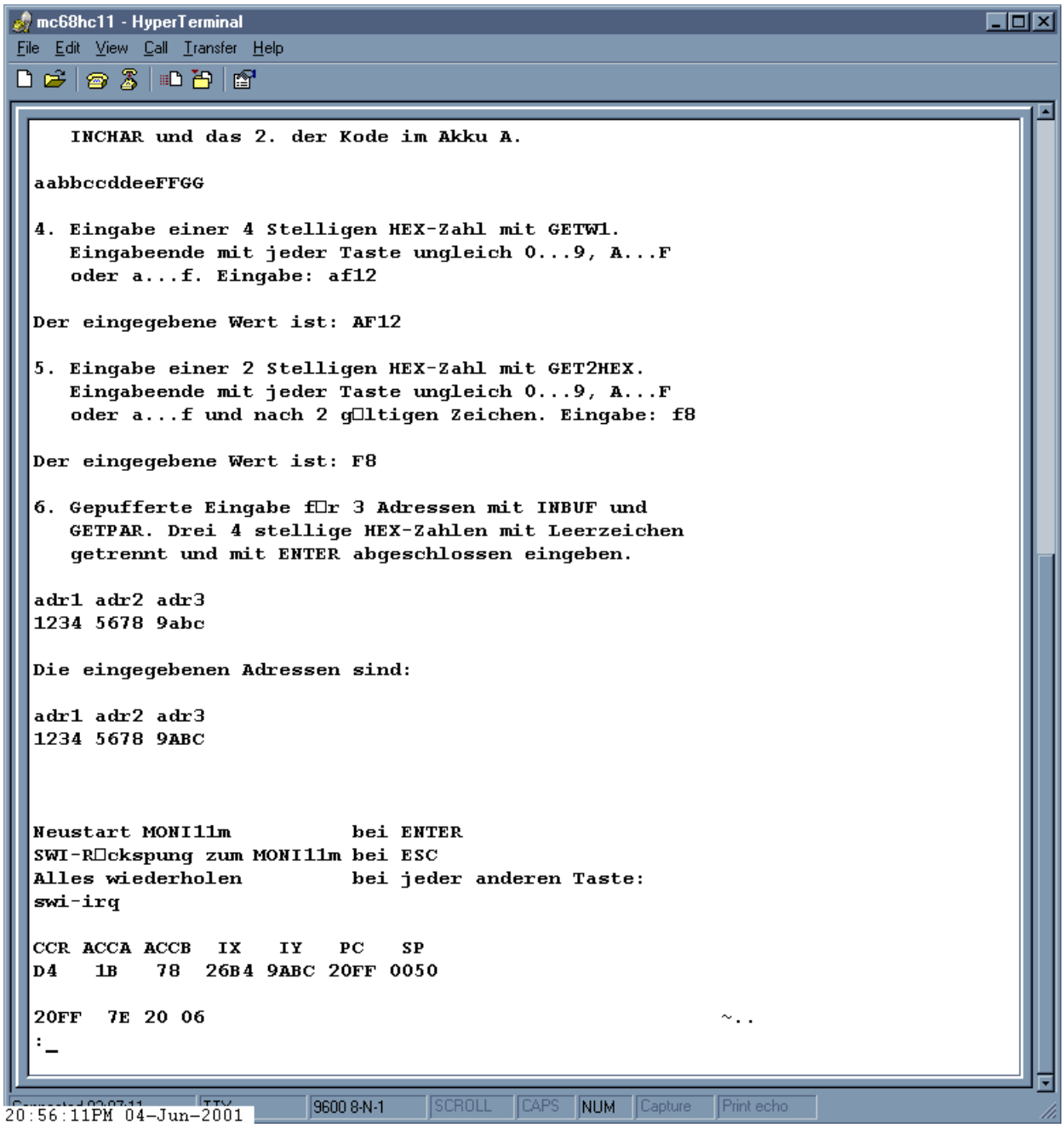
```
            ldx   #cont_mon
            jsr   outtext           * Zeichenkette ausgeben

nochar      jsr   intest
            beq   nochar            * Kein Zeichen eingegeben

            jsr   scin              * Zeichen abholen

            cmpa  #$0d              * ENTER
            beq   moni11m
            cmpa  #$1b              * ESC
            bne   loopw
            swi                     * Rücksprung in MONI11m
loopw       jmp   loop

moni11m     jmp   moni              * Kaltstart des MONI11m

endpg       equ   *

            data
            org   endpg


meld        dcc   "\r\n\n"
            dcc   "Test der MONI11m User-Rufe\r\n"
            dcc   "========"
            dcc   DATUM
            dcc   "========\r\n\n"
            dcc   "1. Anzeige von Registerinhalten\r\n"
            dcc   "A  B  D     IX    IY\00"

wait        dcc   "\r\n\n"
            dcc   "Weiter mit beliebiger Taste: \00"

inch1       dcc   "\r\n\n"
            dcc   "2. Test von INCHAR mit Echo und Wandlung aller\r\n"
            dcc   "   kleinen Buchstaben in Großbuchstaben. Ende bei\r\n"
            dcc   "   ENTER. Jedes eingegebene Zeichen wird 2 mal\r\n"
            dcc   "   angezeigt. Das 1. Zeichen ist das Echo von\r\n"
            dcc   "   INCHAR und das 2. der Kode im Akku A.\r\n\n\00"

inch2       dcc   "\r\n\n"
            dcc   "3. Test von INCHAR mit Echo und ohne Wandlung aller\r\n"
            dcc   "   kleinen Buchstaben in Großbuchstaben. Ende bei\r\n"
            dcc   "   ENTER. Jedes eingegebene Zeichen wird 2 mal\r\n"
            dcc   "   angezeigt. Das 1. Zeichen ist das Echo von\r\n"
            dcc   "   INCHAR und das 2. der Kode im Akku A.\r\n\n\00"

inword      dcc   "\r\n\n"
            dcc   "4. Eingabe einer 4 Stelligen HEX-Zahl mit GETW1.\r\n"
            dcc   "   Eingabeende mit jeder Taste ungleich 0...9, A...F\r\n"
            dcc   "   oder a...f. Eingabe: \00"

noin        dcc   "\r\n\n"
            dcc   "Es wurde kein Wert eingegeben! Die Eingabe wird wiederholt!\00"

wertda      dcc   "\r\n\n"
            dcc   "Der eingegebene Wert ist: \00"

byteein     dcc   "\r\n\n"
            dcc   "5. Eingabe einer 2 Stelligen HEX-Zahl mit GET2HEX.\r\n"
            dcc   "   Eingabeende mit jeder Taste ungleich 0...9, A...F\r\n"
            dcc   "   oder a...f und nach 2 gültigen Zeichen. Eingabe: \00"

errbyin     dcc   "\r\n\n"
            dcc   "Es wurde ein falsches Zeichen eingegeben!\r\n"
            dcc   "Die Eingabe wird wiederholt!\00"

drei_adr    dcc   "\r\n\n"
            dcc   "6. Gepufferte Eingabe für 3 Adressen mit INBUF und\r\n"
```

```
              dcc    "   GETPAR. Drei 4 stellige HEX-Zahlen mit Leerzeichen\r\n"
              dcc    "   getrennt und mit ENTER abgeschlossen eingeben.\r\n\n"
              dcc    "adr1 adr2 adr3\r\n\00"

anz_adr       dcc    "\r\n\n"
              dcc    "Die eingegebenen Adressen sind:\r\n\n"
              dcc    "adr1 adr2 adr3\r\n\00"

cont_mon      dcc    "\r\n\n"
              dcc    "Neustart MONI11m          bei ENTER\r\n"
              dcc    "SWI-Rückspung zum MONI11m bei ESC\r\n"
              dcc    "Alles wiederholen         bei jeder anderen Taste: \00"
*         Ende
```

Bei der Abarbeitung des Programms entsteht dann folgendes Protokoll:

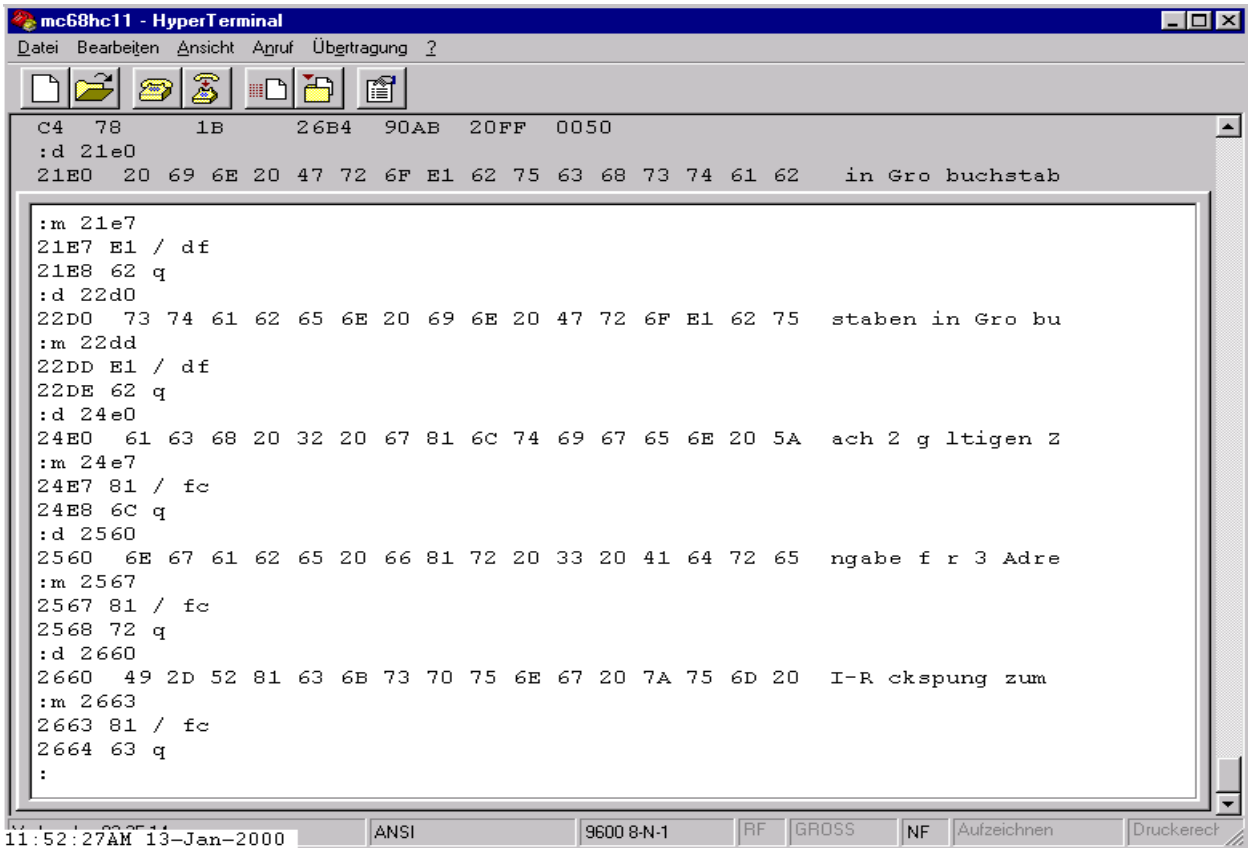


Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”



Änderung der Umlaute vom DOS nach ANSI.

Bedingte Übersetzung in Abhängigkeit von der Zieladresse für ”melgen2.a”



Nun werden die Umlaute auch richtig angezeigt!