3.6. Programmiermodul für serielle EEPROMS vom Typ 93C56N

Der Programmiermodul ist mit einem 68HC11 realisiert und wird über eine RS232-Schnittstelle von einem PC-Programm gesteuert. Er realisiert das Lesen und Schreiben von Speicherzellen des 93C56N und erzeugt alle dazu notwendigen Signalfolgen mit den vorgeschriebenen Zeitbedingungen. Diese Zeitbedingungen sind dabei völlig unabhängig von der effektiven Datenrate auf der RS232-Schnittstelle, wodurch keine unzulässigen Zustände im EEPROM entstehen können. Der Programmiermodul wurde mit einem ZWERG11A ausgetestet. Der folgende Schaltplan enthält nur die Funktionsgruppen des ZWERG11A, die für diese Anwendung benötigt werden.

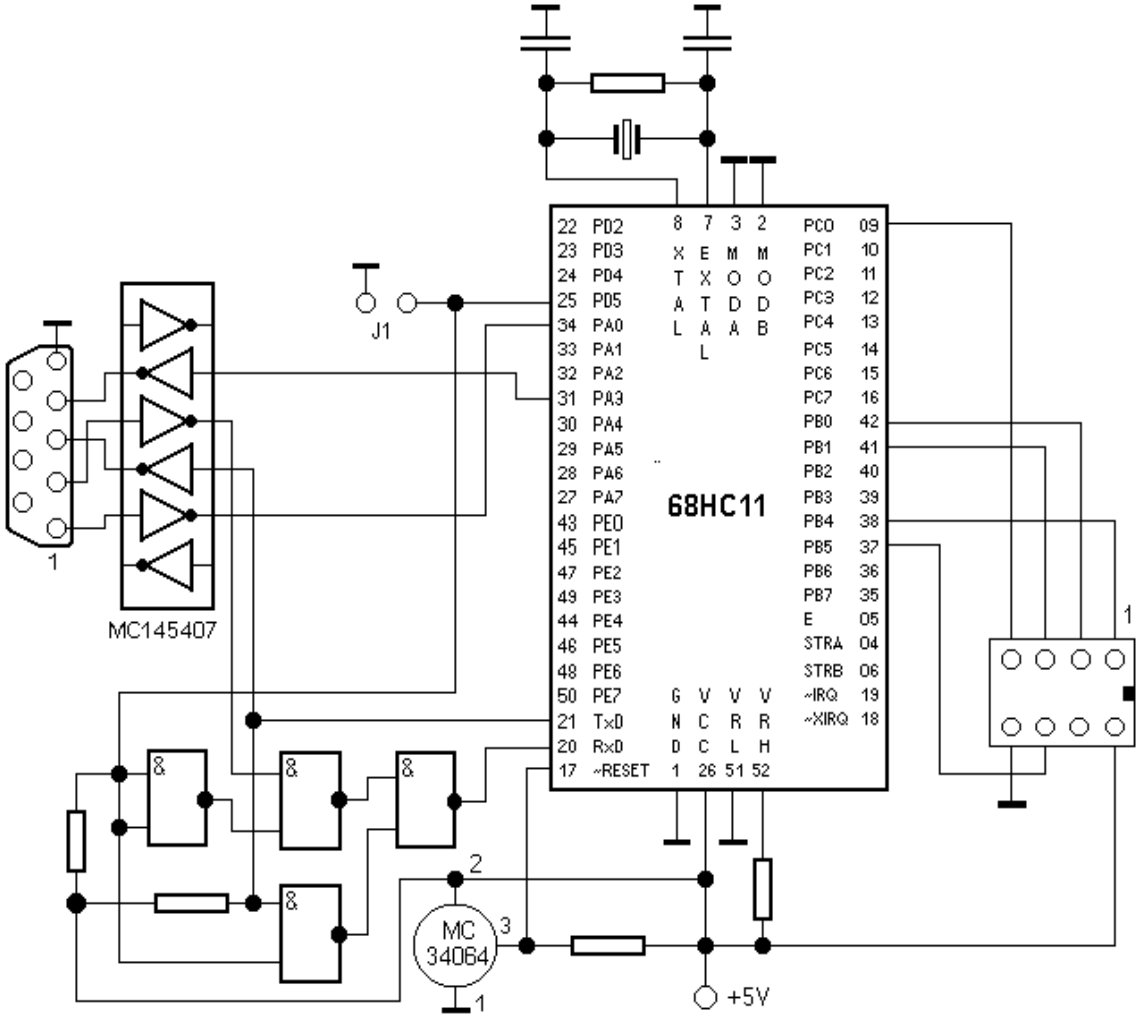

Bild 1: Programiemodul für serielle EEPROMs 93C56N

Ein Progamm für einen PC erlaubt das Lesen, Schreiben und Ändern von Speichezellen des EEPROM. Die Daten die im EEPROM gespeichert sind, können in einem File abgelegt werden oder es kann ein EEPROM mit den Daten aus einem File programmiert werden. Das Programm läuft unter DOS oder in einem DOS-Fenster unter Windows. Beim Aufruf muß als Parameter die Nummer der COM-Schnttstelle des PC angegeben werden, an der der Programmiermodul angeschlossen ist. Wird nichts angegeben, so nimmt das Programm die COM2-Schnittstelle. Das Programm stellt die COM-Schnittstelle selbständig in allen Übertragungsparametern passend zum Programmiermodul ein. Ist beim Start des Programms kein Programmiermodul angeschlossen oder nicht eingeschalten, so erscheint folgende Fehlermeldung.

```
Interface Fehler !!

Prommer meldet sich nicht !


Programmende mit ESC möglich
```

Fehlerausschrift, wenn kein Prommer angeschlossen oder bereit ist.

Es kann dann mit ESC abgebrochen oder bei einer anderen Taste ein neuer Synchronisationsversuch unternommen werden. Ist die Synchronisation erfolgreich, so erscheint folgendes Bild:

```
+------------------------------------------------------------------------------+
¦    EEPROMer für seriellen EEPROM 93C56N      (c) Scö   Dresden 25.10.1995    ¦
¦                                                                              ¦
¦       0 / 8   1 / 9   2 / A   3 / B   4 / C   5 / D   6 / E   7 / F        ¦
¦                                                                              ¦
¦  00   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  08   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  10   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  18   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  20   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  28   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  30   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    nur   ¦
¦  38   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  40   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    mit   ¦
¦  48   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  50   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF ANSI.SYS ¦
¦  58   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  60   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  68   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  70   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦  78   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF          ¦
¦          R - Read        G - Get                                  •         ¦
¦  EEPROM  W - Write  Disk                 ESC - Exit      Adresse  •         ¦
¦          L - Clear       P - Put                                  •         ¦
+------------------------------------------------------------------------------+
```

Bild 2: Bildschirmaufbau des PC-Programm

Mit den **Cuorsertasten** kann eine Adresse angewählt werden und durch Eingabe eines 4-stelligen Hexadezimalen Wertes verändert werden. Sind alle 4 Stellen eigegeben, so wird der Wert sofort in den EEPROM geschrieben. Mit den Kommandos Read, Write und Clear wird immer der vollständige Inhalt des EEPROM bearbeitet. Mit Read **(Taste R)** werden alle Adressen des EEPROM ausgelesen und anschliesend angezeigt. Das folgende Bild wird angezeigt, wenn ein EEPROM eingelesen wird, bei dem alle Bits auf 0 gesetzt sind. Oben links wird während des lesens oder Schreibens auf dem EEPROM immer die aktuelle Adresse Protokolliert, so das dort nach dem Einlesen aller Adressen **R7F** steht:

```
+------------------------------------------------------------------------------+
|    EEPROMer für seriellen EEPROM 93C56N      (c) Scö   Dresden 25.10.1995    |
|                                                                              |
| R7F   0 / 8   1 / 9   2 / A   3 / B   4 / C   5 / D   6 / E   7 / F          |
|                                                                              |
|  00   0000    0000    0000    0000    0000    0000    0000    0000           |
|  08   0000    0000    0000    0000    0000    0000    0000    0000           |
|  10   0000    0000    0000    0000    0000    0000    0000    0000           |
|  18   0000    0000    0000    0000    0000    0000    0000    0000           |
|  20   0000    0000    0000    0000    0000    0000    0000    0000           |
|  28   0000    0000    0000    0000    0000    0000    0000    0000           |
|  30   0000    0000    0000    0000    0000    0000    0000    0000    nur    |
|  38   0000    0000    0000    0000    0000    0000    0000    0000           |
|  40   0000    0000    0000    0000    0000    0000    0000    0000    mit    |
|  48   0000    0000    0000    0000    0000    0000    0000    0000           |
|  50   0000    0000    0000    0000    0000    0000    0000    0000 ANSI.SYS  |
|  58   0000    0000    0000    0000    0000    0000    0000    0000           |
|  60   0000    0000    0000    0000    0000    0000    0000    0000           |
|  68   0000    0000    0000    0000    0000    0000    0000    0000           |
|  70   0000    0000    0000    0000    0000    0000    0000    0000           |
|  78   0000    0000    0000    0000    0000    0000    0000    0000           |
|          R - Read         G - Get                                 •          |
|  EEPROM  W - Write  Disk                  ESC - Exit      Adresse  •         |
|          L - Clear        P - Put                                 •          |
+------------------------------------------------------------------------------+
```

Bild 3: Bildschirmanzeige nach einem Read-Kommando

Write **(Taste W)** schreibt die Daten, die auf dem Bildschirm angezeigt werden in den EEPROM und Clear **(Taste L)** löscht alle Adressen des EEPROM. Im gelöschten Zustand sind alle Bits auf allen Adressen des EEPROM gesetzt. Die auf dem Bildschirm angezeigten Daten können mit dem Kommando Put **(Taste P)** in einem File abgespeichert werden. Dazu wird nach dem Betätigen von P zur Eingabe eines Filenamen aufgefordert. Der Name muß den DOS-Konventionen entsprechen. Wird kein Laufwerk und Pfad mit angegeben, so speichert das Programm die Daten in das aktuelle Directory. Mit dem Kommando Get **(Taste G)** können Daten aus einem File eingelesen und angezeigt werden. Diese Daten können dann mit der Kommando Write in den EEPROM geschrieben werden. Eine Veränderung der Daten ist natürlich auch dann wie oben angegeben möglich. In den folenden beiden Bildern wird der Name eines zu ladenden Files eingegeben und im 2. Bild sind dann die geladenen Daten zu sehen.

```
+------------------------------------------------------------------------------+
|    EEPROMer für seriellen EEPROM 93C56N      (c) Scö   Dresden 25.10.1995    |
|                                                                              |
|       0 / 8   1 / 9   2 / A   3 / B   4 / C   5 / D   6 / E   7 / F          |
|                                                                              |
|  00   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  08   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  10   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  18   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  20   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  28   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  30   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    nur    |
|  38   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  40   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    mit    |
|  48   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  50   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF ANSI.SYS  |
|  58   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  60   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  68   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  70   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|  78   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF           |
|          R - Read         G - Get Name: orig.rom                  •          |
|  EEPROM  W - Write  Disk                  ESC - Exit      Adresse  •         |
|          L - Clear        P - Put                                 •          |
+------------------------------------------------------------------------------+
```

Bild 4: Laden des File "ORIG.ROM"

```
+-------------------------------------------------------------------------+
|   EEPROMer für seriellen EEPROM 93C56N      (c) Scö   Dresden 25.10.1995 |
|                                                                         |
|       0 / 8   1 / 9   2 / A   3 / B   4 / C   5 / D   6 / E   7 / F     |
|                                                                         |
|  00   A73F    AEA5    9289    8065    AB3C    A4A3    668B    8680      |
|  08   AD3C    9C9D    6284    7A80    A45D    B89F    6E8D    8480      |
|  10   5930    C899    6B88    7DC0    8B56    C8AE    B998    7D80      |
|  18   9B35    B9AC    A87E    7680    9548    AC99    6E8E    7383      |
|  20   A449    9992    4782    7D80    8F46    9A95    5984    8483      |
|  28   4C86    B971    6F68    7F80    6781    A494    5886    8380      |
|  30   4B4D    B793    5582    86B4    4A5D    BB90    5D7C    7FB6  nur |
|  38   8632    A1AD    538B    7F7F    6B67    BAA1    6D89    7780      |
|  40   9435    BF8E    718E    7383    0ECA    008A    5863    D966  mit |
|  48   8468    7F9E    12D2    C08A    A922    979C    6B84    7A65      |
|  50   0C4B    00A7    A92A    AFA2    6F71    C465    0C4B    808F ANSI.SYS |
|  58   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF      |
|  60   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF      |
|  68   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF      |
|  70   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF      |
|  78   FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    FFFF    AA00      |
|           R - Read        G - Get                                   •   |
|  EEPROM   W - Write   Disk                  ESC - Exit      Adresse   •   |
|           L - Clear       P - Put                                   •   |
+-------------------------------------------------------------------------+
```

Bild 5: Eingelesene Daten aus dem File ORIG.ROM

Das Programm wird durch Betätigen von **ESC**-Verlassen. Die Anzeige der Daten auf dem Bildschirm erfolgt mit ANSI-Sequenzen, deshalb muß in der CONFIG.SYS des PC der Treiber ANSI.SYS installiert sein! Ist der Treiber nicht installiert rollen die Daten nach oben weg und sind deshalb unleserlich. Ein Schaden kann dadurch aber nicht entstehen!

3.6.1. PC-Steuerprogramm für den Programmiermodul

```
                ASSUME      CS:PGM, SS:STACK, DS:DATA, ES:ABSADR

DEF_PORT        EQU   1                           ;0=COM1, 1=COM2 ...
DEF_TO          EQU   2                           ;Timeoutkonstante
ABSSEG          EQU   0

ABSADR          SEGMENT AT ABSSEG

                ORG   400H

ADR_COM1        EQU   $
ADR_COM2        EQU   $+2
ADR_COM3        EQU   $+4
ADR_COM4        EQU   $+6

ADR_LPT1        EQU   $+8
ADR_LPT2        EQU   $+10
ADR_LPT3        EQU   $+12
ADR_LPT4        EQU   $+14

ANZ_PORT        EQU   $+16

                ORG   478H

TOK_LPT1        EQU   $
TOK_LPT2        EQU   $+1
TOK_LPT3        EQU   $+2
TOK_LPT4        EQU   $+3

TOK_COM1        EQU   $+4
TOK_COM2        EQU   $+5
```

```
TOK_COM3    EQU   $+6
TOK_COM4    EQU   $+7

ABSADR      ENDS

PGM         SEGMENT     PARA PUBLIC 'Programm'

START:      MOV   AX,seg DATA
            MOV   DS,AX                     ; Datensegment einstellen

; Parameter übernehmen

            CMP   BYTE ptr ES:[80H],0
            JNZ   PA_DA

; Keine Parameter

PA_FEHL:    MOV   AL,DEF_PORT
            JMP   NOPA

; Parameter ist vorhanden

PA_DA:      MOV   BX,82H                    ;ADR. 1. Parameter im PSP
            MOV   AL,ES:[BX]                ;Zeichen aus PSP lesen
            CMP   AL,'1'
            JC    PA_FEHL                   ;Parameterfehler
            CMP   AL,'4'+1
            JNC   PA_FEHL                   ;Parameterfehler
            SUB   AL,'1'
NOPA:       MOV   AH,0
            MOV   [PORT],AX

; Port Initialisieren

            MOV   BX,seg ABSADR
            MOV   ES,BX                     ; Extrasegment einstellen

            MOV   BX,offset TOK_COM1
            ADD   AX,BX
            XCHG  AX,BX
            MOV   BYTE ptr ES:[BX],DEF_TO   ;Timeout für Port

            MOV   DX,[PORT]
            MOV   AX,0E7H                   ;9600,N,2,8
            INT   14H

            MOV   BX,offset ADR_COM1
            MOV   DX,[PORT]
            ADD   DX,DX
            ADD   BX,DX
            MOV   DX,ES:[BX]
            ADD   DX,4                      ;Register mit DTR und RTS Bit
            MOV   [DTR_RTS],DX
            IN    AL,DX
            MOV   [REG4],AL

; Portadressen und TO-Konstanten in Anzeigestring eintragen

            MOV   BX,seg ABSADR
            MOV   ES,BX                     ; Extrasegment einstellen

            MOV   AX,WORD ptr ES:[ADR_COM1]
            MOV   SI,offset KA1
```

```
                CALL    OUADR_PU

                MOV     AX,WORD ptr ES:[ADR_COM2]
                MOV     SI,offset KA2
                CALL    OUADR_PU

                MOV     AX,WORD ptr ES:[ADR_COM3]
                MOV     SI,offset KA3
                CALL    OUADR_PU

                MOV     AX,WORD ptr ES:[ADR_COM4]
                MOV     SI,offset KA4
                CALL    OUADR_PU

                MOV     AL,BYTE ptr ES:[TOK_COM1]
                XOR     AH,AH
                CALL    HEXDEZ
                MOV     SI,offset KT1
                CALL    OUADR_PU

                MOV     AL,BYTE ptr ES:[TOK_COM2]
                XOR     AH,AH
                CALL    HEXDEZ
                MOV     SI,offset KT2
                CALL    OUADR_PU

                MOV     AL,BYTE ptr ES:[TOK_COM3]
                XOR     AH,AH
                CALL    HEXDEZ
                MOV     SI,offset KT3
                CALL    OUADR_PU

                MOV     AL,BYTE ptr ES:[TOK_COM4]
                XOR     AH,AH
                CALL    HEXDEZ
                MOV     SI,offset KT4
                CALL    OUADR_PU

; Interfaceprocedur syncronisieren !!!

NO_TO:          CALL    COM_IN
                TEST    AH,80H
                JZ      NO_TO                   ;Noch Zeichen im Puffer
                MOV     AL,'S'                  ;Status anfordern
                CALL    COM_OUT
                CALL    COM_IN
                CMP     AL,'?'
                JNZ     IN_IO
NOCH_0:         MOV     AL,'0'
                CALL    COM_OUT
                CALL    COM_IN
                TEST    AH,80H
                JNZ     NOCH_0
                JMP     NO_TO
IN_IO:          TEST    AH,80H
                JZ      TO_NO                   ;Noch Zeichen im Puffer

ERR_IF:         MOV     DX,offset IF_ERR
                MOV     AH,9
                INT     21H
                MOV     AH,7
                INT     21H
                CMP     AL,27                   ;ESC
```

```
                JNZ     NO_TO
                JMP     ENDE

TO_NO:          MOV     CX,10
IN_IOz:         CALL    COM_IN
                LOOP    IN_IOz                  ;Zeichen des Statuskommandos einlesen

; ************************************************************

MELDUNG:        MOV     DX,offset MELD
                MOV     AH,9
                INT     21H

; Anzeigezyklus für sEEPROM Inhalt
; ================================

ANZ_BYT:        MOV     DX,offset EEPROM
                MOV     AH,9
                INT     21H                     ;Anzeigen aller Informationen
                MOV     DI,0                    ;Adresszähler für Eingaben

NEW_ADR:        MOV     CX,4                    ;4 Zeichen pro ADR
                MOV     SI,0                    ;Zeichenposition in ADR (0...3)

; Beenden des PG durch betätigen von ESC

TAST:           MOV     AH,7
                INT     21H                     ;Warten auf Taste

                CMP     AL,1BH
                JZ      ENDE                    ;Programm beenden

                PUSH    DI
                MOV     BX,offset ADRTAB
                SHL     DI,1
                MOV     BX,[BX+DI]
                POP     DI

                OR      AL,AL                   ;CU_Tasten
                JZ      CU_TAST
                CMP     AL,'0'
                JC      ANZ_BYT
                CMP     AL,'9'+1
                JC      ZIFFER
                AND     AL,0DFH                 ;Klein --> Groß
                CMP     AL,'A'
                JC      ANZ_BYT
                CMP     AL,'F'+1
                JC      A__F                    ;Buchstaben A ... F
                CMP     AL,'R'
                JNZ     NO_R
                JMP     READ_ALL
NO_R:           CMP     AL,'W'
                JNZ     NO_W
                JMP     WRITE_ALL
NO_W:           CMP     AL,'L'
                JNZ     NO_L
                JMP     CLEAR_ALL
NO_L:           CMP     AL,'P'
                JNZ     NO_P
                JMP     PUT
NO_P:           CMP     AL,'G'
                JNZ     NO_G
```

```
                JMP     GET
NO_G:           CMP     AL,'Y'
                JZ      DTR_ON_OFF          ;DTR ein/aus
                CMP     AL,'Z'
                JZ      RTS_ON_OFF          ;RTS ein/aus
                JMP     ANZ_BYT

ENDE:

; Zurück zu DOS

                MOV     AX,4C00H
                INT     21H

DTR_ON_OFF:     MOV     AH,00000001B
                JMP     ON_OFF

RTS_ON_OFF:     MOV     AH,00000010B
ON_OFF:         MOV     DX,[DTR_RTS]
                IN      AL,DX
                XOR     AL,AH               ;Bit invertieren
                OUT     DX,AL
                JMP     ANZ_BYT             ;Anzeigen

CU_TAST:        MOV     AH,7
                INT     21H                 ;Warten auf Taste
                CMP     AL,4DH
                JZ      CUR                 ;CU nach rechts
                CMP     AL,4BH
                JZ      CUL                 ;CU nach links
                CMP     AL,48H
                JZ      CUU                 ;CU nach oben
                CMP     AL,50H
                JZ      CUD                 ;CU nach unten
                JMP     ANZ_BYT

TASTE:          JMP     TAST

ZIFFER:
A__F:           MOV     AH,2
                MOV     DL,AL
                INT     21H                 ;0 ... F anzeigen
                MOV     [BX+SI],AL
                INC     SI
                LOOP    TASTE
                MOV     AL,'W'
                CALL    COM_OUT
                PUSH    SI
                MOV     AX,DI
                CALL    OUADR               ;Adrese in ASCII auf PU_OUADR
                POP     SI
                MOV     AL,PU_OUADR+2
                CALL    COM_OUT
                MOV     AL,PU_OUADR+3
                CALL    COM_OUT
                CALL    ADR_OUT
                TEST    AH,80H
                JZ      CUR                 ;Noch Zeichen im Puffer
                JMP     ERR_IF

CUR:            INC     DI
                CMP     DI,80H
                JNZ     CU_SET
```

```
                MOV     DI,0

; CU auf neue Adresse setzen

CU_SET:         PUSH    DX
                PUSH    DI
                PUSH    CX
                MOV     CL,3
                MOV     DX,offset CUTAB
                SHL     DI,CL                   ;*8
                POP     CX
                ADD     DX,DI                   ;DX+ADR*8
                POP     DI
                ADD     DX,DI                   ;DX+ADR*9
                MOV     AH,9
                INT     21H
                POP     DX

                JMP     NEW_ADR


CUL:            DEC     DI
                JNS     CU_SET
                MOV     DI,7FH
                JMP     CU_SET

CUU:            SUB     DI,8
                JNS     CU_SET
                ADD     DI,80H
                JMP     CU_SET

CUD:            ADD     DI,8
                CMP     DI,80H
                JC      CU_SET
                SUB     DI,80H
                JMP     CU_SET

CLEAR_ALL:      MOV     AL,'C'
                CALL    COM_OUT
                CALL    ADR_IN
                TEST    AH,80H
                JZ      READ_ALL
                JMP     ERR_IF

READ_ALL:       MOV     AL,'R'
                CALL    COM_OUT
                MOV     AL,'0'
                CALL    COM_OUT
                MOV     AL,'0'
                CALL    COM_OUT

                MOV     CX,127                  ; Anzahl der Speicherplätze -1
                CALL    ADR_IN                  ; Adresse einlesen
                TEST    AH,80H
                JZ      R_ZYK                   ;Noch Zeichen im Puffer
                JMP     ERR_IF

R_ZYK:          MOV     AL,'+'
                CALL    COM_OUT
                CALL    ADR_IN
                LOOP    R_ZYK
                JMP     ANZ_BYT                 ;Anzeigen
```

```
WRITE_ALL:  MOV   DI,0                      ;Adresszähler
            MOV   AL,'W'
            CALL  COM_OUT
            MOV   AL,'0'
            CALL  COM_OUT
            MOV   AL,'0'
            CALL  COM_OUT

            MOV   CX,127                    ; Anzahl der Speicherplätze -1
            CALL  ADR_OUT                   ; Adresse SCHREIBEN
            TEST  AH,80H
            JZ    W_ZYK                     ;Noch Zeichen im Puffer
            JMP   ERR_IF

W_ZYK:      MOV   AL,'+'
            CALL  COM_OUT
            INC   DI                        ;Adresse +1
            CALL  ADR_OUT
            LOOP  W_ZYK
            JMP   ANZ_BYT                   ;Anzeigen


; Schreiben einer Datei

; Eröffnen der Datei mit dem einzugebenden Namen

PUT:        MOV   DX,offset PFNAME
            MOV   AH,9
            INT   21H                       ;Auforderung zur Eingabe anzeigen

            CALL  FNAME_IN

; O P E N - File

OPEN:       MOV   DX,offset PFADPUF
            MOV   AX,3D01H                  ;OPEN schreiben
            INT   21H
            JNC   Y_OPEN                    ;Datei existiert schon

; C R E A T E  - File

CREATE:     MOV   DX,offset PFADPUF
            MOV   AH,3CH
            MOV   CX,0                      ;Dateiatribute
            INT   21H                       ;CREATE
            JNC   Y_OPEN
            JMP   OPENERR

Y_OPEN:     MOV   FILENR,AX

WRITE:
            MOV   CX,FLEN                   ;Filelänge schreiben
            MOV   BX,FILENR
            MOV   DX,offset PUFFER
            MOV   AH,40H
            INT   21H
            JC    WRITEERR

; C L O S E  - Fileende

CLOSE:      MOV   BX,FILENR
            MOV   AH,3EH
            INT   21H                       ;CLOSE
```

```
                JNC     CLEND

; CLOSE - Fehler

CLOERR:         PUSH    AX
                MOV     DX,offset CLER
                MOV     AH,9
                INT     21H
                POP     AX
                CALL    FEHCD ;Fehlercode anzeigen

CLEND:          MOV     FILENR,-1

; Schreiben Ende

                JMP     MELDUNG

; CREATE - Fehler

OPENERR:        PUSH    AX
                MOV     DX,offset OPER
                MOV     AH,9
                INT     21H
                POP     AX
                CALL    FEHCD
                JMP     MELDUNG

; WRITE - Fehler

WRITEERR:       PUSH    AX
                MOV     DX,offset WRER
                MOV     AH,9
                INT     21H
                POP     AX
                CALL    FEHCD
                JMP     MELDUNG


; Lesen einer Datei

; Eröffnen der Datei mit dem einzugebenden Namen

GET:            MOV     DX,offset GFNAME
                MOV     AH,9
                INT     21H                     ;Auforderung zur Eingabe anzeigen

                CALL    FNAME_IN

; O P E N - File

                MOV     DX,offset PFADPUF
                MOV     AX,3D00H                ;OPEN lesen
                INT     21H
                JNC     Y_OPENr                 ;Datei existiert schon
                JMP     OPENERR
Y_OPENr:        MOV     FILENR,AX

                MOV     CX,FLEN                 ;Filelänge schreiben
                MOV     BX,FILENR
                MOV     DX,offset PUFFER
                MOV     AH,3FH                  ;Read
                INT     21H
                JC      READERR
```

```
; C L O S E  - Fileende

; READ Ende

                JMP     CLOSE

; R E A D - Fehler

READERR:        PUSH    AX
                MOV     DX,offset RDER
                MOV     AH,9
                INT     21H
                POP     AX
                CALL    FEHCD
                JMP     MELDUNG

; ************************************************************

;=================================================================

; Unterprogramme

; Zeichen von COM empfangen und in AL laden

COM_IN          PROC    NEAR
                PUSH    DX
                MOV     DX,[PORT]
                MOV     AH,2
                INT     14H
                POP     DX
                TEST    AH,80H
                JNZ     TO_IN

; Zeichen in AL an COM senden

COM_OUT         LABEL   NEAR
                PUSH    DX
                MOV     DX,[PORT]
                MOV     AH,1
                INT     14H
                POP     DX
TO_IN:          RET
COM_IN          ENDP

; aktuelle Adresse vom EEPROM empfangen

ADR_IN          PROC    NEAR
                PUSH    CX
                MOV     BP,offset AKT_ADR
                CALL    COM_IN                  ;R/W lesen
                TEST    AH,80H
                JNZ     TO_ADR
                MOV     DS:[BP],AL
                CALL    COM_IN                  ;Leerzeichen
                CALL    COM_IN                  ;high Tetrade der Adresse ASCII
                MOV     DS:[BP+1],AL
                AND     AL,7                    ;nur 128 Adressen
                MOV     CL,4
                SHL     AL,CL
                MOV     BL,AL
                CALL    COM_IN                  ;low Terade der Adresse ASCII
                MOV     DS:[BP+2],AL
```

```
                SUB     AL,'0'
                CMP     AL,9+1
                JC      ZIFF
                SUB     AL,7                    ;Buchstaben in Pseudotetraden A...F
ZIFF:           OR      BL,AL
                XOR     BH,BH                   ;BH=00
                SHL     BX,1
                MOV     SI,offset ADRTAB
                MOV     BX,[SI+BX]
                CALL    COM_IN                  ;Leerzeichen nach Adresse
                MOV     CX,4
                MOV     DI,0
D_INZY:         CALL    COM_IN
                MOV     [BX+DI],AL
                INC     DI
                LOOP    D_INZY
                CALL    COM_IN                  ;CR
                CALL    COM_IN                  ;LF
                MOV     DX,offset AKT_ADRA
                MOV     AH,9
                INT     21H
TO_ADR:         POP     CX
                RET
ADR_IN          ENDP

ADR_OUT         PROC    NEAR
                PUSH    BX
                PUSH    CX
                PUSH    DI
                MOV     BX,offset ADRTAB
                SHL     DI,1
                MOV     BX,[BX+DI]
                MOV     CX,4
D_OUTZY:        MOV     AL,[BX]
                CALL    COM_OUT
                INC     BX
                LOOP    D_OUTZY
                CALL    ADR_IN                  ;zurücklesen des Ergebnisses
                POP     DI
                POP     CX
                POP     BX
                RET
ADR_OUT         ENDP

; Umwandlung eines Dezimalwertes von 0 bis 65535
; in den HEX-Wert ohne Vorzeichen
; ----------------------------------------------
; IN:     BL= 00 bis 06  10000ter
;         AX= 0000 bis 9999
; OUT:  AX= 0000H bis 0FFFFH

DEZHEX          PROC    NEAR
                PUSH    BX
                PUSH    CX
                PUSH    DX
                MOV     DX,AX
                AND     AX,0FH                  ;Einer übernehmen
                CALL    DEHE3                   ;Schieben
                MOV     CX,10
                CALL    DEHE
                MOV     CX,100
                CALL    DEHE
                MOV     CX,1000
```

```
                CALL    DEHE
                MOV     CX,10000
                CALL    DEHE
                POP     DX
                POP     CX
                POP     BX
                RET
DEZHEX          ENDP

; BCD-Stelle zu AX in HEX addieren
; --------------------------------
; IN:     BL:DX noch nicht abgearbeitete BCD stellen
;         CX= Wertigkeit der niedrigsten BCD-Stelle in BL:DX
; OUT:  AX= HEX-Wert der abgearbeiteten BCD-Stellen

DEHE            PROC    NEAR
                PUSH    DX
                AND     DL,0FH                  ;niedrigste BCD-Stelle in DL
                JE      DEHE2
DEHE1:          ADD     AX,CX
                DEC     DL
                JNE     DEHE1
DEHE2:          POP     DX
DEHE3:          MOV     CL,4
                SHR     DX,CL
                SHL     BL,CL
                ADD     DH,BL
                MOV     BL,0
                RET
DEHE            ENDP

; 16 Bit Hexadezimal in Dezimalwert (BCD) umwandeln
; -------------------------------------------------
;  IN:     AX:= 16 Bit HEX ohne Vorzeichen
;  OUT: AX:= 0000 bis 9999 in BCD
;         BL:= 00 bis 06 10000 Stelle
HEXDEZ          PROC    NEAR
                PUSH    DX
                PUSH    CX
                MOV     BL,0
                MOV     CX,0
                MOV     DX,10000
                CALL    HEDE
                MOV     DX,1000
                CALL    HEDE
                MOV     DX,100
                CALL    HEDE
                MOV     DX,10
                CALL    HEDE
                MOV     BH,AL                   ;Einer AL nach BH
                CALL    HEDE3
                MOV     AX,CX
                POP     CX
                POP     DX
                RET
HEXDEZ          ENDP

; Bestimmung 10er, 100ter, 1000ter und 10000ter
; ---------------------------------------------
;  IN:     AX:=restliche HEX-Wert
;         DX:=(1. 10000, 2. 1000, 3. 100, 4. 10)
;  OUT: BL:CX = BCD-Wert fuer abgezogene 10000, 1000, 100 und 10
```

```
HEDE        PROC  NEAR
            PUSH  BX
            MOV   BH,-1
HEDE1:      INC   BH
            SUB   AX,DX
            JNC   HEDE1                ;solange abziehen bis negativ
            ADD   AX,DX                ;Rest in AX einstellen, BH=BCD-Stelle
HEDE2:      MOV   BL,CH
            MOV   DX,CX
            MOV   CL,4
            SHR   BL,CL
            SHL   DX,CL
            MOV   CX,DX
            ADD   CL,BH
            MOV   DL,BL
            POP   BX
            MOV   BL,DL
            RET

HEDE3:      PUSH  BX
            JMP   HEDE2
HEDE        ENDP

; Adresse in AX in ASCII-String wandeln

OUADR       PROC  NEAR
            MOV   SI,offset PU_OUADR
OUADR_PU:   PUSH  AX
            MOV   AL,AH
            CALL  NEAR ptr OUBYTE
            POP   AX

OUBYTE:     PUSH  AX
            PUSH  CX
            MOV   CL,4
            SAR   AL,CL
            CALL  NEAR ptr OUHB
            POP   CX
            POP   AX

OUHB:       AND   AL,0FH
            ADD   AL,'0'
            CMP   AL,'9'+1
            JC    ZIF
            ADD   AL,7

ZIF:        PUSH  AX
            MOV   [SI],AL
            INC   SI
            POP   AX
            RET
OUADR       ENDP

; Anzeigen des Fehlercode

FEHCD       PROC  NEAR
            PUSH  AX
            PUSH  DX
            AND   AL,0FH
            ADD   AL,'0'
            MOV   DL,AL
            MOV   AH,2
            INT   21H                  ;Fehlercode Anzeigen
```

```
; W A I T  zum lesen

                PUSH    BX
                PUSH    CX
                MOV     BX,5                    ;5 mal BEL
T0:             MOV     DL,7                    ;BEL
                MOV     AH,2
                INT     21H
                MOV     CX,3000
T1:             LOOP    T1
                DEC     BX
                JNZ     T0
                POP     CX
                POP     BX
                POP     DX
                POP     AX
                RET
FEHCD           ENDP

FNAME_IN        PROC    NEAR
                MOV     AH,0AH
                MOV     DX,offset PFADPU
                INT     21H                     ;gepufferte Tastatureingabe

                MOV     AL,PFADPU + 1           ;Anzahl der eingegebenen Zeichen in AL
                MOV     DX,offset PFADPUF       ;Position 1. Zeichen
                MOV     AH,0
                ADD     AX,DX                   ;Position von CR (=0DH)
                MOV     SI,AX
                MOV     AL,0
                MOV     [SI],AL                 ;Puffer fuer Open mit 0 beenden

; Umwandlung der kleinen Buchstaben im Puffer in große Buchstaben

                MOV     SI,DX
KLGR:           MOV     AL,[SI]                 ;Zeichen aus Puffer lesen
                CMP     AL,0
                JZ      NA_IO                   ;alles in große Buchstaben gewandelt
                CMP     AL,'a'
                JC      NOKL
                CMP     AL,'z'+1
                JNC     NOKL
                AND     AL,0DFH
                MOV     [SI],AL                 ;Zeichen in Puffer schreiben
NOKL:           INC     SI
                JMP     KLGR

NA_IO:          RET
FNAME_IN        ENDP

PGM             ENDS

DATA            SEGMENT     WORD PUBLIC 'Daten_und_Variablen'

PORT            DW      -1
DTR_RTS         DW      -1                      ;Adr. 8250 Reg. 4
REG4            DB      -1                      ;Inhalt Reg. 4 beim Start

PU_OUADR        DB      '0000   '


MELD            DB      27,'[2J'
```

```
                DB      '+------------------------------------------------------------------------+'
                DB      10,13,'¦  EEPROMer für seriellen EEPROM 93C56N     (c) Scö   Dresden '
                INCLUDE         C:\DATUM.ASM
                DB      ' ¦'
                DB      10,13,'¦                                                                ¦'
                DB      10,13,'¦    0 / 8        1 / 9        2 / A        3 / B        4 / C        5 / D        6 / E        7 / F          ¦'
                DB      10,13,'¦                                                                ¦'
                DB      10,13,'¦ 00 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 08 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 10 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 18 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 20 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 28 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 30 ....         ....         ....         ....         ....         ....         ....         ....  nur  ¦'
                DB      10,13,'¦ 38 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 40 ....         ....         ....         ....         ....         ....         ....         ....  mit  ¦'
                DB      10,13,'¦ 48 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 50 ....         ....         ....         ....         ....         ....         ....         .... ANSI.SYS ¦'
                DB      10,13,'¦ 58 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 60 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 68 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 70 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦ 78 ....         ....         ....         ....         ....         ....         ....         ....           ¦'
                DB      10,13,'¦       R - Read      G - Get                                                    '
                DB      24,'  ¦'
                DB      10,13,'¦ EEPROM  W - Write         Disk                          ESC - Exit   Adresse      '
                DB      29,'  ¦'
                DB      10,13,'¦       L - Clear      P - Put                                                    '
                DB      25,'  ¦'
                DB      10,13,'+------------------------------------------------------------------------+'

                DB      '$'

KS0             DB      27,'[8;17H'
KA1             DB      '....    '
KA2             DB      '....    '
KA3             DB      '....    '
KA4             DB      '....'

                DB      27,'[9;17H'
KT1             DB      '....    '
KT2             DB      '....    '
KT3             DB      '....    '
KT4             DB      '....'

EEPROM          EQU     $
PUFFER          EQU     $

                DB      27,'[06;09H'
ADR00           DB      'FFFF    '
ADR01           DB      'FFFF    '
ADR02           DB      'FFFF    '
ADR03           DB      'FFFF    '
ADR04           DB      'FFFF    '
ADR05           DB      'FFFF    '
ADR06           DB      'FFFF    '
ADR07           DB      'FFFF'

                DB      27,'[07;09H'
ADR08           DB      'FFFF    '
ADR09           DB      'FFFF    '
ADR0A           DB      'FFFF    '
ADR0B           DB      'FFFF    '
```

```
ADR0C       DB      'FFFF   '
ADR0D       DB      'FFFF   '
ADR0E       DB      'FFFF   '
ADR0F       DB      'FFFF'

            DB      27,'[08;09H'
ADR10       DB      'FFFF   '
ADR11       DB      'FFFF   '
ADR12       DB      'FFFF   '
ADR13       DB      'FFFF   '
ADR14       DB      'FFFF   '
ADR15       DB      'FFFF   '
ADR16       DB      'FFFF   '
ADR17       DB      'FFFF'

            DB      27,'[09;09H'
ADR18       DB      'FFFF   '
ADR19       DB      'FFFF   '
ADR1A       DB      'FFFF   '
ADR1B       DB      'FFFF   '
ADR1C       DB      'FFFF   '
ADR1D       DB      'FFFF   '
ADR1E       DB      'FFFF   '
ADR1F       DB      'FFFF'

            DB      27,'[10;09H'
ADR20       DB      'FFFF   '
ADR21       DB      'FFFF   '
ADR22       DB      'FFFF   '
ADR23       DB      'FFFF   '
ADR24       DB      'FFFF   '
ADR25       DB      'FFFF   '
ADR26       DB      'FFFF   '
ADR27       DB      'FFFF'

            DB      27,'[11;09H'
ADR28       DB      'FFFF   '
ADR29       DB      'FFFF   '
ADR2A       DB      'FFFF   '
ADR2B       DB      'FFFF   '
ADR2C       DB      'FFFF   '
ADR2D       DB      'FFFF   '
ADR2E       DB      'FFFF   '
ADR2F       DB      'FFFF'

            DB      27,'[12;09H'
ADR30       DB      'FFFF   '
ADR31       DB      'FFFF   '
ADR32       DB      'FFFF   '
ADR33       DB      'FFFF   '
ADR34       DB      'FFFF   '
ADR35       DB      'FFFF   '
ADR36       DB      'FFFF   '
ADR37       DB      'FFFF'

            DB      27,'[13;09H'
ADR38       DB      'FFFF   '
ADR39       DB      'FFFF   '
ADR3A       DB      'FFFF   '
ADR3B       DB      'FFFF   '
ADR3C       DB      'FFFF   '
ADR3D       DB      'FFFF   '
ADR3E       DB      'FFFF   '
```

```
ADR3F       DB      'FFFF'

            DB      27,'[14;09H'
ADR40       DB      'FFFF   '
ADR41       DB      'FFFF   '
ADR42       DB      'FFFF   '
ADR43       DB      'FFFF   '
ADR44       DB      'FFFF   '
ADR45       DB      'FFFF   '
ADR46       DB      'FFFF   '
ADR47       DB      'FFFF'

            DB      27,'[15;09H'
ADR48       DB      'FFFF   '
ADR49       DB      'FFFF   '
ADR4A       DB      'FFFF   '
ADR4B       DB      'FFFF   '
ADR4C       DB      'FFFF   '
ADR4D       DB      'FFFF   '
ADR4E       DB      'FFFF   '
ADR4F       DB      'FFFF'

            DB      27,'[16;09H'
ADR50       DB      'FFFF   '
ADR51       DB      'FFFF   '
ADR52       DB      'FFFF   '
ADR53       DB      'FFFF   '
ADR54       DB      'FFFF   '
ADR55       DB      'FFFF   '
ADR56       DB      'FFFF   '
ADR57       DB      'FFFF'

            DB      27,'[17;09H'
ADR58       DB      'FFFF   '
ADR59       DB      'FFFF   '
ADR5A       DB      'FFFF   '
ADR5B       DB      'FFFF   '
ADR5C       DB      'FFFF   '
ADR5D       DB      'FFFF   '
ADR5E       DB      'FFFF   '
ADR5F       DB      'FFFF'

            DB      27,'[18;09H'
ADR60       DB      'FFFF   '
ADR61       DB      'FFFF   '
ADR62       DB      'FFFF   '
ADR63       DB      'FFFF   '
ADR64       DB      'FFFF   '
ADR65       DB      'FFFF   '
ADR66       DB      'FFFF   '
ADR67       DB      'FFFF'

            DB      27,'[19;09H'
ADR68       DB      'FFFF   '
ADR69       DB      'FFFF   '
ADR6A       DB      'FFFF   '
ADR6B       DB      'FFFF   '
ADR6C       DB      'FFFF   '
ADR6D       DB      'FFFF   '
ADR6E       DB      'FFFF   '
ADR6F       DB      'FFFF'

            DB      27,'[20;09H'
```

```
ADR70       DB      'FFFF   '
ADR71       DB      'FFFF   '
ADR72       DB      'FFFF   '
ADR73       DB      'FFFF   '
ADR74       DB      'FFFF   '
ADR75       DB      'FFFF   '
ADR76       DB      'FFFF   '
ADR77       DB      'FFFF'

            DB      27,'[21;09H'
ADR78       DB      'FFFF   '
ADR79       DB      'FFFF   '
ADR7A       DB      'FFFF   '
ADR7B       DB      'FFFF   '
ADR7C       DB      'FFFF   '
ADR7D       DB      'FFFF   '
ADR7E       DB      'FFFF   '
ADR7F       DB      'FFFF'

            DB      27,'[6;9H$'

FLEN        EQU     $-PUFFER

IN_PUF      DW      4 DUP(-1)

ADRTAB      DW      ADR00
            DW      ADR01
            DW      ADR02
            DW      ADR03
            DW      ADR04
            DW      ADR05
            DW      ADR06
            DW      ADR07
            DW      ADR08
            DW      ADR09
            DW      ADR0A
            DW      ADR0B
            DW      ADR0C
            DW      ADR0D
            DW      ADR0E
            DW      ADR0F

            DW      ADR10
            DW      ADR11
            DW      ADR12
            DW      ADR13
            DW      ADR14
            DW      ADR15
            DW      ADR16
            DW      ADR17
            DW      ADR18
            DW      ADR19
            DW      ADR1A
            DW      ADR1B
            DW      ADR1C
            DW      ADR1D
            DW      ADR1E
            DW      ADR1F

            DW      ADR20
            DW      ADR21
            DW      ADR22
            DW      ADR23
```

```
DW      ADR24
DW      ADR25
DW      ADR26
DW      ADR27
DW      ADR28
DW      ADR29
DW      ADR2A
DW      ADR2B
DW      ADR2C
DW      ADR2D
DW      ADR2E
DW      ADR2F

DW      ADR30
DW      ADR31
DW      ADR32
DW      ADR33
DW      ADR34
DW      ADR35
DW      ADR36
DW      ADR37
DW      ADR38
DW      ADR39
DW      ADR3A
DW      ADR3B
DW      ADR3C
DW      ADR3D
DW      ADR3E
DW      ADR3F

DW      ADR40
DW      ADR41
DW      ADR42
DW      ADR43
DW      ADR44
DW      ADR45
DW      ADR46
DW      ADR47
DW      ADR48
DW      ADR49
DW      ADR4A
DW      ADR4B
DW      ADR4C
DW      ADR4D
DW      ADR4E
DW      ADR4F

DW      ADR50
DW      ADR51
DW      ADR52
DW      ADR53
DW      ADR54
DW      ADR55
DW      ADR56
DW      ADR57
DW      ADR58
DW      ADR59
DW      ADR5A
DW      ADR5B
DW      ADR5C
DW      ADR5D
DW      ADR5E
DW      ADR5F
```

```
        DW      ADR60
        DW      ADR61
        DW      ADR62
        DW      ADR63
        DW      ADR64
        DW      ADR65
        DW      ADR66
        DW      ADR67
        DW      ADR68
        DW      ADR69
        DW      ADR6A
        DW      ADR6B
        DW      ADR6C
        DW      ADR6D
        DW      ADR6E
        DW      ADR6F

        DW      ADR70
        DW      ADR71
        DW      ADR72
        DW      ADR73
        DW      ADR74
        DW      ADR75
        DW      ADR76
        DW      ADR77
        DW      ADR78
        DW      ADR79
        DW      ADR7A
        DW      ADR7B
        DW      ADR7C
        DW      ADR7D
        DW      ADR7E
        DW      ADR7F

CUTAB   DB      27,'[06;09H$'
        DB      27,'[06;17H$'
        DB      27,'[06;25H$'
        DB      27,'[06;33H$'
        DB      27,'[06;41H$'
        DB      27,'[06;49H$'
        DB      27,'[06;57H$'
        DB      27,'[06;65H$'

        DB      27,'[07;09H$'
        DB      27,'[07;17H$'
        DB      27,'[07;25H$'
        DB      27,'[07;33H$'
        DB      27,'[07;41H$'
        DB      27,'[07;49H$'
        DB      27,'[07;57H$'
        DB      27,'[07;65H$'

        DB      27,'[08;09H$'
        DB      27,'[08;17H$'
        DB      27,'[08;25H$'
        DB      27,'[08;33H$'
        DB      27,'[08;41H$'
        DB      27,'[08;49H$'
        DB      27,'[08;57H$'
        DB      27,'[08;65H$'

        DB      27,'[09;09H$'
```

```
DB      27,'[09;17H$'
DB      27,'[09;25H$'
DB      27,'[09;33H$'
DB      27,'[09;41H$'
DB      27,'[09;49H$'
DB      27,'[09;57H$'
DB      27,'[09;65H$'

DB      27,'[10;09H$'
DB      27,'[10;17H$'
DB      27,'[10;25H$'
DB      27,'[10;33H$'
DB      27,'[10;41H$'
DB      27,'[10;49H$'
DB      27,'[10;57H$'
DB      27,'[10;65H$'

DB      27,'[11;09H$'
DB      27,'[11;17H$'
DB      27,'[11;25H$'
DB      27,'[11;33H$'
DB      27,'[11;41H$'
DB      27,'[11;49H$'
DB      27,'[11;57H$'
DB      27,'[11;65H$'

DB      27,'[12;09H$'
DB      27,'[12;17H$'
DB      27,'[12;25H$'
DB      27,'[12;33H$'
DB      27,'[12;41H$'
DB      27,'[12;49H$'
DB      27,'[12;57H$'
DB      27,'[12;65H$'

DB      27,'[13;09H$'
DB      27,'[13;17H$'
DB      27,'[13;25H$'
DB      27,'[13;33H$'
DB      27,'[13;41H$'
DB      27,'[13;49H$'
DB      27,'[13;57H$'
DB      27,'[13;65H$'

DB      27,'[14;09H$'
DB      27,'[14;17H$'
DB      27,'[14;25H$'
DB      27,'[14;33H$'
DB      27,'[14;41H$'
DB      27,'[14;49H$'
DB      27,'[14;57H$'
DB      27,'[14;65H$'

DB      27,'[15;09H$'
DB      27,'[15;17H$'
DB      27,'[15;25H$'
DB      27,'[15;33H$'
DB      27,'[15;41H$'
DB      27,'[15;49H$'
DB      27,'[15;57H$'
DB      27,'[15;65H$'

DB      27,'[16;09H$'
```

```
                DB      27,'[16;17H$'
                DB      27,'[16;25H$'
                DB      27,'[16;33H$'
                DB      27,'[16;41H$'
                DB      27,'[16;49H$'
                DB      27,'[16;57H$'
                DB      27,'[16;65H$'

                DB      27,'[17;09H$'
                DB      27,'[17;17H$'
                DB      27,'[17;25H$'
                DB      27,'[17;33H$'
                DB      27,'[17;41H$'
                DB      27,'[17;49H$'
                DB      27,'[17;57H$'
                DB      27,'[17;65H$'

                DB      27,'[18;09H$'
                DB      27,'[18;17H$'
                DB      27,'[18;25H$'
                DB      27,'[18;33H$'
                DB      27,'[18;41H$'
                DB      27,'[18;49H$'
                DB      27,'[18;57H$'
                DB      27,'[18;65H$'

                DB      27,'[19;09H$'
                DB      27,'[19;17H$'
                DB      27,'[19;25H$'
                DB      27,'[19;33H$'
                DB      27,'[19;41H$'
                DB      27,'[19;49H$'
                DB      27,'[19;57H$'
                DB      27,'[19;65H$'

                DB      27,'[20;09H$'
                DB      27,'[20;17H$'
                DB      27,'[20;25H$'
                DB      27,'[20;33H$'
                DB      27,'[20;41H$'
                DB      27,'[20;49H$'
                DB      27,'[20;57H$'
                DB      27,'[20;65H$'

                DB      27,'[21;09H$'
                DB      27,'[21;17H$'
                DB      27,'[21;25H$'
                DB      27,'[21;33H$'
                DB      27,'[21;41H$'
                DB      27,'[21;49H$'
                DB      27,'[21;57H$'
                DB      27,'[21;65H$'


AKT_ADRA        DB      27,'[4;3H'
AKT_ADR         DB      'R00$'


PFADPU          DB      40
                DB      0
PFADPUF         DB      40 DUP(0)               ;Pfad fuer File

FILENR          DW      -1                      ;Nr. des Händlers
```

```
; Fehlerausschriften

OPER        DB      27,'[3;25H',7,'*** OPEN-Fehler $'

WRER        DB      27,'[3;25H',7,'*** WRITE-Fehler $'

RDER        DB      27,'[3;25H',7,'*** READ-Fehler $'

CLER        DB      27,'[3;25H',7,'*** CLOSE-Fehler $'

; Aufforderung zur Nameneingabe

PFNAME      DB      27,'[24;38HName: $'
GFNAME      DB      27,'[22;38HName: $'

IF_ERR      DB      27,'[2J',10,7                   ;CLR-SCRN
            DB      'Interface Fehler !!',10,10,13,7
            DB      'Prommer meldet sich nicht !',10,10,10,13
            DB      'Programmende mit ESC möglich$'

DATA        ENDS

STACK       SEGMENT     WORD STACK 'STK'

            DW      100 DUP(-1)

STACK       ENDS

            END     START
```

3.6.2. Steuerprogramm des Programmiermodul

```
*           P r o g r a m m i e r f a s s u n g
*           ========================
*
* 1K Typen mit der ORGanisation 128*8 oder 64*16
*   MSM16811   AK93C46    HY93C46    NMC9346    CAT93C46
*   LX93C46    93C46      S-2914R    BR93C46
*   MSM16911   ER5911     CAT59C11   S-2911R
* 2K Typen mit der ORGanisation 256*8 oder 128*16
*   MSM16812   AK93C56    CAT35C102  S-2924R    BR93C56
*   MSM16912   ER5912     CAT35C202  S-2921R
*
*   St.:          __________          St.:
*                |          |
*   1.01    GND -| 05    04 |- DO    1.11
*   1.06    ORG -| 06    03 |- DI    1.09
*           NC  -| 07    02 |- SK    1.08
*   1.03    Vcc -| 08    01 |- CS    1.05
*                |____oo____|
*
* Vcc    - 5 Volt Stromversorgung
* ORG    - Umschaltung der ORGanisation
*          = 0   -->  8*128 oder  8*256
*          = 1   --> 16*64  oder 16*128
* CS     - Chipselekt für den Baustein
* SK     - Takt für serielle Datensignale
* DI     - serieller Dateneingang
* DO     - serieller Datenausgang
```

```
* GND   - Masse (Steckverbinder 1.01 1.02 und 1.26)
* Vcc   - 5 Volt Betriebsspannung
*
* Belegung der Signale an den Ports des 68HC11
* ====================================
*
* Port   Signal          Stecker        sEEPROM
* ------------------------------------------------------------------------
* PA0    DCD             2.09
* PA3    DTR             2.06
*
* PB0    SK              1.08           02
* PB1    DI              1.09           03
* PB2                    1.20
* PB3                    1.21
* PB4    CS              1.05           01
* PB5    ORG             1.06           06
* PB6                    1.07
* PB7                    1.22
*
* PC0    DO              1.11           04
* PC1                    1.12
* PC2                    1.13
* PC3                    1.14
* PC4                    1.15
* PC5                    1.16
* PC6                    1.17
* PC7                    1.18
*
* PD0    RxD             2.04 * kann über LOOP mit TxD verbunden sein!
* PD1    TxD             2.05
*

    00000010   d_len          equ    16
    00000008   a_len          equ    8
    0000b600   EEPROM         equ    $b600          eeprom area


    00001000   REG            equ    $1000          register base

    00001000   porta          equ    REG+$00        port a data reg.
    00001002   pioc           equ    REG+$02        parallel i/o control reg.
    00001003   portc          equ    REG+$03        port c data reg.
    00001004   portb          equ    REG+$04        port b data reg.
    00001005   portcl         equ    REG+$05        port c latched data reg.
    00001007   ddrc           equ    REG+$07        port c data direction reg.
    00001008   portd          equ    REG+$08        port d data reg.
    00001009   ddrd           equ    REG+$09        port d data direction reg.
    0000100a   porte          equ    REG+$0a        port e data reg.
    0000100b   cforc          equ    REG+$0b        timer compare force reg.
    0000100c   oc1m           equ    REG+$0c        output compare 1 mask reg.
    0000100d   oc1d           equ    REG+$0d        output compare 1 data reg.
    0000100e   tcnt           equ    REG+$0e        timer count reg. 16 bit
    0000100e   tcnth          equ    REG+$0e        timer count reg. high
    0000100f   tcntl          equ    REG+$0f        timer count reg. low
    00001010   tic1h          equ    REG+$10        timer input capture reg. high
    00001011   tic1l          equ    REG+$11        tic1 low
    00001012   tic2h          equ    REG+$12        tic2 high
    00001013   tic2l          equ    REG+$13        tic2 low
    00001014   tic3h          equ    REG+$14        tic3 high
    00001015   tic3l          equ    REG+$15        tic3 low
    00001016   toc1h          equ    REG+$16        timer output compare reg. high
    00001017   toc1l          equ    REG+$17        toc1 low
```

```
00001018 toc2h    equ REG+$18 toc2 high
00001019 toc2l    equ REG+$19 toc2 low
0000101a toc3h    equ REG+$1a toc3 high
0000101b toc3l    equ REG+$1b toc3 low
0000101c toc4h    equ REG+$1c toc4 high
0000101d toc4l    equ REG+$1d toc4 low
0000101e toc5h    equ REG+$1e toc5 high
0000101f toc5l    equ REG+$1f toc5 low
00001020 tctl1    equ REG+$20 timer control reg. 1
00001021 tctl2    equ REG+$21 timer control reg. 2
00001022 tmsk1    equ REG+$22 main timer interrupt mask reg.1
00001023 tflg1    equ REG+$23 main timer interrupt flag reg.1
00001024 tmsk2    equ REG+$24 misk. timer interrupt mask reg.2
00001025 tflg2    equ REG+$25 misc. timer interrupt flag
00001026 pactl    equ REG+$26 pulse accumulator control reg.
00001027 pacnt    equ REG+$27 pulse accumulator count reg.
00001028 spcr     equ REG+$28 spi control reg.
00001029 spsr     equ REG+$29 spi status reg.
0000102a spdr     equ REG+$2a spi data reg.
0000102b baud     equ REG+$2b sci baud rate control reg.
0000102c sccr1    equ REG+$2c sci control reg. 1
0000102d sccr2    equ REG+$2d sci control reg. 2
0000102e scsr     equ REG+$2e sci status reg.
0000102f scdr     equ REG+$2f sci data reg.
00001030 adctl    equ REG+$30 a/d control/status reg.
00001031 adr1     equ REG+$31 a/d result reg. 1
00001032 adr2     equ REG+$32 a/d result reg. 2
00001033 adr3     equ REG+$33 a/d result reg. 3
00001034 adr4     equ REG+$34 a/d result reg. 4
00001035 bprot    equ REG+$35 block protect reg.
00001039 option   equ REG+$39 system configuration options
0000103a coprst   equ REG+$3a arm/reset cop timer circuitry
0000103b pprog    equ REG+$3b eeprom programming reg.
0000103c hprio    equ REG+$3c highest priority interrupt and misc.
0000103d init     equ REG+$3d ram and i/o mapping reg.
0000103e test1    equ REG+$3e factory test reg.
0000103f config   equ REG+$3f configuration control reg.

00000000 oporta   equ $00 port a data reg.
00000002 opioc    equ $02 parallel i/o control reg.
00000003 oportc   equ $03 port c data reg.
00000004 oportb   equ $04 port b data reg.
00000005 oportcl  equ $05 port c latched data reg.
00000007 oddrc    equ $07 port c data direction reg.
00000008 oportd   equ $08 port d data reg.
00000009 oddrd    equ $09 port d data direction reg.
0000000a oporte   equ $0a port e data reg.
0000000b ocforc   equ $0b timer compare force reg.
0000000c ooc1m    equ $0c output compare 1 mask reg.
0000000d ooc1d    equ $0d output compare 1 data reg.
0000000e otcnth   equ $0e timer count reg. high
0000000f otcntl   equ $0f timer count reg. low
00000010 otic1h   equ $10 timer input capture reg. high
00000011 otic1l   equ $11 tic1 low
00000012 otic2h   equ $12 tic2 high
00000013 otic2l   equ $13 tic2 low
00000014 otic3h   equ $14 tic3 high
00000015 otic3l   equ $15 tic3 low
00000016 otoc1h   equ $16 timer output compare reg. high
00000017 otoc1l   equ $17 toc1 low
00000018 otoc2h   equ $18 toc2 high
00000019 otoc2l   equ $19 toc2 low
0000001a otoc3h   equ $1a toc3 high
```

```
    0000001b    otoc3l          equ     $1b             toc3 low
    0000001c    otoc4h          equ     $1c             toc4 high
    0000001d    otoc4l          equ     $1d             toc4 low
    0000001e    otoc5h          equ     $1e             toc5 high
    0000001f    otoc5l          equ     $1f             toc5 low
    00000020    otctl1          equ     $20             timer control reg. 1
    00000021    otctl2          equ     $21             timer control reg. 2
    00000022    otmsk1          equ     $22             main timer interrupt mask reg.1
    00000023    otflg1          equ     $23             main timer interrupt flag reg.1
    00000024    otmsk2          equ     $24             misk. timer interrupt mask reg.2
    00000025    otflg2          equ     $25             misc. timer interrupt flag
    00000026    opactl          equ     $26             pulse accumulator control reg.
    00000027    opacnt          equ     $27             pulse accumulator count reg.
    00000028    ospcr           equ     $28             spi control reg.
    00000029    ospsr           equ     $29             spi status reg.
    0000002a    ospdr           equ     $2a             spi data reg.
    0000002b    obaud           equ     $2b             sci baud rate control reg.
    0000002c    osccr1          equ     $2c             sci control reg. 1
    0000002d    osccr2          equ     $2d             sci control reg. 2
    0000002e    oscsr           equ     $2e             sci status reg.
    0000002f    oscdr           equ     $2f             sci data reg.
    00000030    oadctl          equ     $30             a/d control/status reg.
    00000031    oadr1           equ     $31             a/d result reg. 1
    00000032    oadr2           equ     $32             a/d result reg. 2
    00000033    oadr3           equ     $33             a/d result reg. 3
    00000034    oadr4           equ     $34             a/d result reg. 4
    00000035    obprot          equ     $35             block protect reg.
    00000039    ooption         equ     $39             system configuration options
    0000003a    ocoprst         equ     $3a             arm/reset cop timer circuitry
    0000003b    opprog          equ     $3b             eeprom programming reg.
    0000003c    ohprio          equ     $3c             highest priority interrupt and misc.
    0000003d    oinit           equ     $3d             ram and i/o mapping reg.
    0000003e    otest1          equ     $3e             factory test reg.
    0000003f    oconfig         equ     $3f             configuration control reg.

                                bss
    00000000                    org     $0000

*                               ds.b    60
*               stack           ds.b    1
    00000001    anz_sh          ds.b    1               *Zähler für Schiebetakte
0001  00000001r_w               ds.b    1               *lesen oder schreiben
0002  00000001f_s               ds.b    1               *Freigabe/Sperre für W und C
0003  00000001f_s_old           ds.b    1
0004  00000001adr               ds.w    1               *Adresse
0006  00000001dat               ds.w    1               *daten


                                text
0000b600                        org     EEPROM

0000b600        start           equ     *
*                               lds     stack
b600  8620                      ldaa    #$20
b602  b71009                    staa    ddrd            *ZWERG11A TxD-RxD Schleife öffnen
b605  ce1000                    ldx     #REG

****** init sci (no register save)
*
b608  8630      sciini          ldaa    #$30
b60a  b7102b                    staa    baud            * 9600 baud      *
b60d  1c0008                    bset    8,oporta,x      * DTR Busy set *
b610  860c                      ldaa    #$0c
```

```
b612  b7102d                staa    sccr2           * enable sci      *

b615  1c0420                bset    $20,oportb,x    * ORG aktiv
b618  7f0004                clr     adr
b61b  7f0005                clr     adr+1
b61e  7f0006                clr     dat
b621  7f0007                clr     dat+1
b624  8652                  ldaa    #"R"
b626  9701                  staa    <r_w
b628  7f0002                clr     f_s             * W und C gesperrt !
b62b  7f0003                clr     f_s_old

b62e  bdb7a5    zyk         jsr     sciget          * Kommando einlesen R W + - . S
b631  18de04                ldy     <adr
b634  812b                  cmpa    #"+"
b636  2733                  beq     iadr            * Adresse + 1
b638  812d                  cmpa    #"-"
b63a  272b                  beq     dadr            * Adresse - 1
b63c  812e                  cmpa    #"."
b63e  273f                  beq     adrout          * gleiche Adresse

b640  84df                  anda    #$df            * klein in gro
b642  8152                  cmpa    #"R"
b644  2730                  beq     adrin           * Lesen; Adresse folgt
b646  8157                  cmpa    #"W"
b648  272c                  beq     adrin           * Schreiben; Adresse folgt
*                           cmpa    #"L"
*                           beq     adrin           * Löschen; Adresse folgt
b64a  8143                  cmpa    #"C"
b64c  2603                  bne     noclr
b64e  7eb6ec                jmp     clear           * alles löschen
b651  8153      noclr       cmpa    #"S"
b653  2609                  bne     komfeh          * falches Kommando
b655  7f0002                clr     f_s
b658  bdb748                jsr     fg_sp
b65b  7eb702                jmp     status

b65e  8d02      komfeh      bsr     err
b660  20cc                  bra     zyk

*               ** Fehler Anzeigen

b662  863f      err         ldaa    #"?"
b664  7eb79a                jmp     sciput


*               ** ADR - 1

b667  1809      dadr        dey
b669  2002                  bra     idadr

*               ** ADR + 1

b66b  1808      iadr        iny
b66d  18df04    idadr       sty     <adr
b670  200d                  bra     adrout

*               ** Adresse enthält nicht nur 0 ... 9 und A ... F

b672  8dee      aderr       bsr     err
b674  2002                  bra     adi1

*               ** Adresse laden
```

```
b676  9701      adrin           staa    <r_w                    * Kommando merken
     0000b678   adi1            equ     *
*                               jsr     get2h
*                               blo     aderr                   * Fehler in ADR
*                               staa    <adr
b678  bdb7d4                    jsr     get2h
b67b  25f5                      blo     aderr                   * Fehler in ADR
b67d  9705                      staa    <adr+1

*               ** Adresse an sEEPROM für Kommando R, W, oder L senden

     0000b67f   adrout          equ     *

*               ** löschen, lesen oder schreiben ?

b67f  9601                      ldaa    <r_w
*                               cmpa    #"L"
*                               beq     loeadr
b681  8157                      cmpa    #"W"
b683  270c                      beq     write
b685  7f0002                    clr     f_s
b688  bdb748                    jsr     fg_sp
b68b  2034                      bra     read

*               ** Ein Byte schreiben !!!

b68d  8dd3      d_err           bsr     err                     * Fehler in Daten
b68f  2005                      bra     wr1

b691  9702      write           staa    <f_s
b693  bdb748                    jsr     fg_sp
b696  bdb7d4    wr1             jsr     get2h                   * zu schreibendes high Byte laden
b699  25f2                      blo     d_err
b69b  9706                      staa    <dat
b69d  bdb7d4                    jsr     get2h
b6a0  25eb                      blo     d_err
b6a2  9707                      staa    <dat+1

b6a4  bdb76c                    jsr     adr_vor
b6a7  8aa0                      oraa    #$a0                    *Schreibbefehl einblenden
b6a9  1c0410                    bset    $10,oportb,x            *CS ein
b6ac  bdb72c                    jsr     sh_out
b6af  8610                      ldaa    #d_len
b6b1  b70000                    staa    anz_sh
b6b4  dc06                      ldd     <dat
b6b6  bdb737                    jsr     sh_outd
b6b9  bdb779                    jsr     by_ry
b6bc  1d0410                    bclr    $10,oportb,x            *CS aus
b6bf  2000                      bra     read

*               ** Eine Adresse löschen !!!

     0000b6c1   loeadr          equ     *
*                               staa    <f_s
*                               jsr     fg_sp
*                               jsr     adr_vor
*                               oraa    #$e0                    *Löschbefehl einblenden
*                               bset    $10,oportb,x            *CS ein
*                               bsr     sh_out
*                               jsr     by_ry
*                               bclr    $10,oportb,x            *CS aus
```

```
*                ** Ein Byte lesen

     0000b6c1  read          equ     *
b6c1  bdb76c                 jsr     adr_vor
b6c4  8ac0                   oraa    #$c0            *Lesebefehl einblenden
b6c6  1c0410                 bset    $10,oportb,x    *CS ein
b6c9  8d61                   bsr     sh_out

*                ** 2 Byte einlesen nach dat
:
b6cb  c611       sh_in       ldab    #d_len+1
b6cd  d700                   stab    <anz_sh
b6cf  4f                     clra
b6d0  5f                     clrb
b6d1  1c0401     sh_in1      bset    1,oportb,x      *SK ein
b6d4  1d0401                 bclr    1,oportb,x      *SK aus
b6d7  dd06                   std     <dat
b6d9  7a0000                 dec     anz_sh
b6dc  2709                   beq     end_shin
b6de  05                     lsld                    *höchste Bit nach CY
b6df  1f0301ee               brclr   1,oportc,x,sh_in1
b6e3  ca01                   orab    #1              *DO ist 1
b6e5  20ea                   bra     sh_in1

b6e7  1d0410     end_shin    bclr    $10,oportb,x    *CS aus
b6ea  2016                   bra     status

*                ** Alles Löschen

     0000b6ec  clear         equ     *
b6ec  9702                   staa    <f_s
b6ee  8d58                   bsr     fg_sp
b6f0  8610                   ldaa    #16
b6f2  9700                   staa    <anz_sh
b6f4  cc9000                 ldd     #$9000
b6f7  1c0410                 bset    $10,oportb,x    *CS ein
b6fa  8d30                   bsr     sh_out
b6fc  bdb779                 jsr     by_ry
b6ff  1d0410                 bclr    $10,oportb,x    *CS aus

*                ** Systeminformationen anzeigen

     0000b702  status        equ     *

b702  9601                   ldaa    <r_w
b704  bdb79a                 jsr     sciput

b707  8620                   ldaa    #" "
b709  bdb79a                 jsr     sciput

*                            ldd     <adr
*                            jsr     put2h           * High anzeigen
*                            tba
b70c  9605                   ldaa    <adr+1
b70e  bdb7e5                 jsr     put2h           * Low anzeigen

b711  8620                   ldaa    #" "
b713  bdb79a                 jsr     sciput

b716  dc06       anzda       ldd     <dat
b718  bdb7e5                 jsr     put2h           * Daten high anzeigen
b71b  17                     tba
b71c  bdb7e5                 jsr     put2h           * Daten low  anzeigen
```

```
b71f  860d                    ldaa    #13
b721  bdb79a                  jsr     sciput          * CR
b724  860a                    ldaa    #10
b726  bdb79a                  jsr     sciput          * LF
b729  7eb62e                  jmp     zyk


*               ** Unterprogramme

*               ** Inhalt von D an DI senden! Anzahl der Bit in "anz_sh"

b72c  1c0401    sh_out        bset    1,oportb,x      *SK ein
b72f  1d0401                  bclr    1,oportb,x      *SK aus
b732  7a0000                  dec     anz_sh
b735  270d                    beq     end_sh
b737  05        sh_outd       lsld                    *höchste Bit nach CY
b738  2405                    bcc     bit0
b73a  1c0402                  bset    2,oportb,x      *DI ein
b73d  20ed                    bra     sh_out
b73f  1d0402    bit0          bclr    2,oportb,x      *DI aus
b742  20e8                    bra     sh_out
b744  1d0402    end_sh        bclr    2,oportb,x      *DI aus
b747  39                      rts

*               ** Freigabe oder Sperren von W, L und C

b748  36        fg_sp         psha
b749  37                      pshb
b74a  9602                    ldaa    <f_s
b74c  9103                    cmpa    <f_s_old
b74e  2719                    beq     gl_old
b750  9703                    staa    <f_s_old
b752  8610                    ldaa    #16
b754  9700                    staa    <anz_sh
b756  1c0410                  bset    $10,oportb,x    *CS ein
b759  cc8000                  ldd     #$8000
b75c  7d0002                  tst     f_s
b75f  2703                    beq     sperr
b761  cc9800                  ldd     #$9800
b764  8dc6      sperr         bsr     sh_out
b766  1d0410                  bclr    $10,oportb,x    *CS aus
b769  33        gl_old        pulb
b76a  32                      pula
b76b  39                      rts

*               ** Adresse für sh_out aufbereiten

b76c  860c      adr_vor       ldaa    #a_len+4
b76e  9700                    staa    <anz_sh
b770  dc04                    ldd     <adr
b772  4f                      clra                    *Adresse ist maximal 255
b773  05                      lsld
b774  05                      lsld
b775  05                      lsld
b776  05                      lsld
b777  05                      lsld
b778  39                      rts

*               ** Warten auf READY

b779  1d0410    by_ry         bclr    $10,oportb,x    *CS aus
b77c  1c0401                  bset    1,oportb,x      *SK ein
```

```
b77f  1d0401                    bclr    1,oportb,x              *SK aus
b782  1c0410                    bset    $10,oportb,x            *CS ein
b785  1c0401    by_ry1          bset    1,oportb,x              *SK ein
b788  1d0401                    bclr    1,oportb,x              *SK aus
b78b  1e0301f6                  brset   1,oportc,x,by_ry1
b78f  1c0401    by_ry2          bset    1,oportb,x              *SK ein
b792  1d0401                    bclr    1,oportb,x              *SK aus
b795  1f0301f6                  brclr   1,oportc,x,by_ry2
b799  39                        rts

************** SCI-DTR module ***********************************************************
*                                                                                        *
* Date          Version What                                                  Who        *
* 13.07.95      1.00    create                                                 he       *
* ---------------------------------------------------------------------------------------*
*                                 - Notes -                                              *
*                                                                                        *
* Dieses Modul stellt Funktionen für die serielle Schnittstelle (SCI) des                *
* HC11 mit DTR-Protokoll zur Verfügung.                                                  *
* ---------------------------------------------------------------------------------------*
*                                 - Usage -                                              *
*                                                                                        *
* include in application                                                                 *
* ---------------------------------------------------------------------------------------*
* Copyright (C) MCT Paul & Scherer Mikrocomputertechnik GmbH Berlin                      *
* DTR-Protokoll und Indx für Register von Dr. Siegmar Schöne                            *
*****************************************************************************************

****** init sci (no register save)
*
*                sciini  ldaa    #$30
*                        staa    baud                    * 9600 baud                    *
*                        bset    8,oporta,x              * DTR Busy set                 *
*                        ldaa    #$0c
*                        staa    sccr2                   * enable sci                   *
*                        rts

****** put char in ACCA to sci if DCD=0
*
     0000b79a   sciput  equ     *
b79a  1f0001fc  sc1     brclr   1,oporta,x,sc1          * DCD = 0 ? NEIN
b79e  1f2e80f8          brclr   $80,oscsr,x,sc1         * transmit buf empty? no
b7a2  b7102f            staa    scdr                    * write to buf                 *
*        rts

****** get char from sci to ACCA
*
b7a5  1e2e2005sciget    brset   $20,oscsr,x,scig1       * receive buf full? yes        *
b7a9  1d0008            bclr    8,oporta,x              * DTR not Busy set             *
b7ac  20f7              bra     sciget
b7ae  1c0008    scig1   bset    8,oporta,x              * DTR Busy set                 *
b7b1  b6102f            ldaa    scdr                    * read from buf                *
b7b4  39                rts

************** input module ************************************************************
*                                                                                        *
* Date          Version What                                                  Who        *
* 25.10.91      1.00    create                                                 he       *
* ---------------------------------------------------------------------------------------*
*                                 - Notes -                                              *
*                                                                                        *
* Dieses Modul stellt Eingabe-Funktionen zur Verfügung.                                  *
* ---------------------------------------------------------------------------------------*
```

```
*                                   - Usage -                                          *
*                                                                                      *
* include in application                                                               *
* ------------------------------------------------------------------------------------ *
* Copyright (C) MCT Paul & Scherer Mikrocomputertechnik GmbH Berlin                    *
****************************************************************************************

****** get 1hex, get 2hex
*  return ACCA  = 1hex/2hex (C set for non-hex)
*
b7b5  bdb7a5    get1h   jsr     sciget
b7b8  8030              suba    #"0"
b7ba  2b14              bmi     get11
b7bc  8109              cmpa    #9
b7be  2312              bls     get12
b7c0  8007              suba    #7
b7c2  2b0c              bmi     get11
b7c4  810f              cmpa    #$0f
b7c6  230a              bls     get12
b7c8  8020              suba    #32
b7ca  2b04              bmi     get11
b7cc  810f              cmpa    #$0f
b7ce  2302              bls     get12
b7d0  0d        get11   sec
b7d1  39                rts

b7d2  0c        get12   clc
b7d3  39                rts


b7d4  37        get2h   pshb
b7d5  8dde              bsr     get1h
b7d7  250a              bcs     get22
b7d9  16                tab
b7da  8dd9              bsr     get1h
b7dc  2505              bcs     get22
b7de  58                aslb
b7df  58                aslb
b7e0  58                aslb
b7e1  58                aslb
b7e2  1b                aba
b7e3  33        get22   pulb
b7e4  39                rts

************** output module ************************************************************
*                                                                                      *
* Date          Version What                                                  Who      *
* 25.10.91      1.00    create                                                 he      *
* ------------------------------------------------------------------------------------ *
*                                   - Notes -                                          *
*                                                                                      *
* Dieses Modul stellt Ausgabe-Funktionen zur Verfügung.                                *
* ACHTUNG:                                                                             *
*  Allen Ausgabe-Funktionen muß die Adresse der zu benutzenden low-level               *
*  Funktion zur Ausgabe eines einzelnen Zeichens in IY übergeben werden.               *
* ------------------------------------------------------------------------------------ *
*                                   - Usage -                                          *
*                                                                                      *
* include in application                                                               *
* ------------------------------------------------------------------------------------ *
* Copyright (C) MCT Paul & Scherer Mikrocomputertechnik GmbH Berlin                    *
****************************************************************************************
```

```
****** put 1hex, put2hex
*       ACCA      = 1hex, 2hex
*       IY= low-level putc
*
b7e5  36          put2h   psha
b7e6  44                  lsra
b7e7  44                  lsra
b7e8  44                  lsra
b7e9  44                  lsra
b7ea  8d01                bsr     put1h
b7ec  32                  pula

b7ed  36          put1h   psha
b7ee  840f                anda    #$0f
b7f0  8b30                adda    #"0"
b7f2  8139                cmpa    #"9"
b7f4  2302                bls     put11
b7f6  8b07                adda    #7
b7f8  bdb79a      put11   jsr     sciput
b7fb  32                  pula
b7fc  39                  rts

*                 * Ende sEEPROM
```