

!!! This site has syntax errors, the code is incorrect. !!!



Computer-Magazine

C.Häfele 2026



(XEROX Scanner Firmware Problem, you know ;)

David Kriesel and Xerox)

Booten mit SYSTEM: Accessories aus dem Ordner booten

GRUNDLAGEN

Mit der Hilfe von Ordnern ist eine übersichtliche Organisation von Daten und Programmen möglich. Das ermöglicht erst ein vernünftiges Arbeiten mit Festplatten. So sind z.B. alle Programme, die bei einem Systemstart automatisch ausgeführt werden sollen, in dem Auto-Ordner zusammengefaßt. Leider ist dies mit Accessories nicht möglich. Sie müssen im Wurzelverzeichnis des Boot-Laufwerks abgelegt sein. Das Programm System ermöglicht es jetzt, Accessories in einem Ordner SYSTEM.ACC zusammenzufassen. Als weitere Anwendung ermöglicht das Programm das Setzen des Environment-Strings für GEM.

System muß aus dem Auto-Ordner gestartet werden, damit es sich vor dem Laden und Starten der Accessories installieren kann. Wird es vom Desktop aus gestartet, erscheint eine Fehlermeldung. Die Installation des Programms läßt sich durch Drücken der Control-Taste beim Booten verhindern. Ist System erfolgreich installiert, werden nur Accessories aus dem Ordner SYSTEM.ACC geladen und gestartet. Die, die sich im Wurzelverzeichnis befinden, werden nicht geladen.

Environment-Strings

Das Laden aus dem "SYSTEM.ACC"-Ordner funktioniert bei Accessories mit RSC-Dateien leider nicht ohne weiteres. Das GEM sucht die RSC-Dateien nur im aktuellen Verzeichnis, was in diesem Fall dem Wurzelverzeichnis entspricht. Es sucht auch noch in jedem Ordner, der durch die Environment-Variable PATH angegeben ist. Leider setzt das GEM die Variable nur auf ,PATH=;?;' (? = Bootlaufwerk). Zum Laden der RSC-Dateien aus dem SYSTEM.ACC-Ordner müßte die Variable auf ,PATH=;?;?\SYSTEM.ACC' gesetzt werden. Die Variable sollte eigentlich ,PATH=;?;?\SYSTEM.ACC' heißen, aber durch einen TOS-Fehler muß nach ,PATH=' eineingesetzt werden. Nach dem Variablennamen PATH können dann weitere Pfade folgen, jeweils durch Semikolon getrennt.

Turbo C 2.0 ermöglicht es, z.B. mit der Environment-Variablen TC einen Pfad anzugeben, in dem sich dann die Help-Dateien der Assembler und der Linker befinden müssen. Eine weitere Anwendung ist, daß Angeben eines Pfades für das Clipboard. Dies geschieht mit der Environment-Variable CLIPBRD. Jedes Programm, das das Clipboard unterstützt, muß zuerst die Environment-Variable CLIP-BRD abfragen. Sollte sie vorhanden sein, darf das Programm sich nur auf diesen Pfad beziehen. Dieser Pfad sollte auch noch mit SCRP_WRITE (AES 81)

gesetzt werden. Befindet sich im angegebenen Pfad kein Clipboard-Ordner, muß man ihn noch anlegen. Nur wenn die Environment-Variable CLIPBRD nicht vorhanden ist, darf der Pfad des Clipboard mit SCRP_READ (AES 89) gelesen werden. Damit ist sichergestellt, daß sich alle Programme auf den gleichen Pfad beziehen, d.h. es würde sich nicht auf jedem Boot-Laufwerk ein Clipboard-Ordner befinden.

SYSTEM.INF

Wird System aus dem Auto-Ordner gestartet, lädt es die Datei .SYSTEM.INF1 nach, die auch im Auto-Ordner stehen muß. In dieser Datei sollten die Environment-Strings enthalten sein, wobei jeder String mit Return abgeschlossen sein muß. Die erste Variable sollte die PATH-Variable sein. Das Programm funktioniert auch ohne SYSTEM.INF; es werden dann aber nur die Pfad variablen für das Wurzelverzeichnis und den SYSTEM. ACC-Ordner gesetzt.

Das Setzen der Environment-Strings ist natürlich nur einem Programm möglich, da es nur einen Zeiger auf sie gibt. Sollte das erste Zeichen von SYSTEM.INF ein ,-' sein, setzt das Programm die Environment-Strings nicht. Damit ist es einem anderen Programm möglich, sie zu setzen. Die Environment-Strings dürfen höchstens eine Länge von 224 Bytes haben. Will man größere setzen, muß ENV_BUFFER auf einen größeren Wert im Listing gesetzt werden.

Programmbeschreibung

Bei einem Systemstart setzt das GEM mit Dsetpath (Gemdos 59) einen Pfad auf das Wurzelverzeichnis. Danach sucht es mit der Ffirst (Gemdos 78) mit der Suchmaske ,*.ACC' nach den Accessories im Wurzelverzeichnis. Das Setzen des Pfades und das Suchen geschieht zweimal hintereinander.

System hängt sich nun nach dem XBRA-Protokoll in den Gemdos-Trap (Trap #1) Systemvariable \$84 ein und wartet auf einen Ffirst-Aufruf. Hat der Aufruf die Suchmaske ,*.ACC' und kommt vom Desktop, wird die Rücksprungadresse durch die Adresse von MY_PATH ersetzt.

Die Routine MY_PATH setzt nun einen neuen Pfad (VSYSTEM.ACC) und sucht in diesem Ordner mit einem Ffirst-Aufruf (\SYSTEM.ACC*.ACC) nach Accessories. Danach springt es zu der original Routine zurück. Es reicht, wenn die ersten beiden Ffirst-Aufrufe abgefangen werden. Deshalb wird die Variable COUNTER bei jedem Aufruf um eins erniedrigt. Elat sie einen Wert von Null, springt die Routine NEW_GEMDOS immer den alten Gemdos-Vektor an.

Der Zeiger _run enthält die Adresse eines Zeigers auf die aktuelle Basepage. Mit ihrer Hilfe wird festgestellt, ob das Desktop das aktuelle Programm ist. Der Zeiger kbshift zeigt auf ein Byte, in dem der Tastaturstatus abgelegt ist. Durch den Tastaturstatus wird ein Drücken der Control-Taste abgefragt. Beide Zeiger sind ab TOS 1.2 in der SYSHDR-Struktur abgelegt.

```
Ffirst-Aufruf
move.w  attribs,-(sp)      ;Dateityp
pea     fspec              ;Zeiger auf Dateinamen
move.w  #$4e,-(sp)        ;Gemdos-Nummer
```

trap #1

Stack-Aufbau bei einem Ffirst-Aufruf im
User-Modus

User-Stack für 680xx:

AdresseParameter

USP \$4e

USP+2 fspec

USP+6 attribs

Supervisor-Stack für 68000:

AdresseParameter

SSP Statusregister

SSP+2 Rücksprungadresse

Supervisor-Stack ab 68010:

AdresseParameter

SSP Statusregister

SSP+2 Rücksprungadresse

SSP+6 Format-Code

Stack-Aufbau bei einem Ffirst-Aufruf im
Supervisor-Modus

Supervisor Stack für 68000:

AdresseParameter

SSP Statusregister

SSP+2 Rücksprungadresse

SSP+6 \$4e

SSP+8 fspec

SSP+12 attribs

Supervisor-Stack ab 68010:

AdresseParameter

SSP Statusregister

SSP+2 Rücksprungadresse

SSP+6 Format-Code

SSP+8 \$4e

SSP+10 fspec

SSP+14 attribs

Speicherbelegung

Der Programmteil von RESI_ANFANG bis RESI_ENDE bleibt resident im Speicher. Bei System sind das 744 Bytes, wobei die Basepage mit 256 Bytes schon eingerechnet ist. Das Programm ist so ausgelegt, daß die Installationsroutinen sich hinter dem residenten Teil des Programms befinden. Dadurch kann der Speicherplatz der Installationsroutinen nach erfolgreicher Installation dem Systemspeicher wieder zur Verfügung gestellt werden.

Betriebsarten

Die 680xx-CPU hat zwei Betriebsarten, zum einen den Supervisor-, und zum anderen den User-Modus. Jede der beiden Betriebsarten hat einen eigenen Stapelzeiger. Will man feststellen, welche Funktion bei einem Gemdos-Aufruf vorliegt, muß man zuerst herausfinden, in welchem Betriebsmodus der Gemdos-Aufruf gemacht wurde. Dazu muß das Statusregister vom Stack geholt werden. Nun wird Bit 13 (Supervisor-Flag) des Statusregisters überprüft. Ist dies gesetzt, wurde der Aufruf aus dem Supervisor-Modus ausgeführt. Hat man jetzt den Betriebsmodus festgestellt, lädt man den entsprechenden Stapelzeiger. Da die Parameter vor jedem Gemdos-Aufruf auf dem Stack abgelegt werden, kann man nun die Parameter verändern. Dabei muß beachtet werden, daß die Parameter auf dem Stack jetzt in umgekehrter Reihenfolge vorliegen.

Bei den Nachfolgern der 68000-CPU wird ein zusätzliches Wort im Supervisor-Modus bei einem Gemdos-Aufruf auf dem Stack abgelegt, der Format Code. Damit System auch auf diesen Prozessoren läuft, wird in diesem Fall ein Offset von zwei Bytes bei einem Gemdos-Aufruf im Supervisor-Modus dazuaddiert.

```
PATH=;?;?\SYSTEM.ACC;D:\1ST_WORD;E:\RESOURCE TC=D:\TC_2\HELP  
CLIPBRD=D:\SYSTEM\CLIPBRD
```

Beispiel für SYSTEM.INF

```
'!' -> kein Environment-String setzen  
,?' -> Platzhalter für Boot-Laufwerk
```

Sonderzeichen für SYSTEM.INF

Um den Prozessor festzustellen, sucht das Programm das Cookie „_CPU“. Die Systemvariable _p_cookies (\$5a0) enthält den Zeiger auf die Cookies. Sollte kein Zeiger vorhanden sein, nimmt das Programm an, daß eine 68000-CPU vorhanden ist.

Environment setzen

Das AES wird wie jedes andere Programm auch vom Gemdos mit der Pexec-Funktion (Gemdos 75) gestartet. Dabei springt es durch den exec_os-Vektor (Systemvariable \$4fe). System installiert sich nach dem XBRA-Protokoll in diesen Vektor und wartet darauf, daß das AES vom Gemdos gestartet wird. Sollen die Environment-Strings von System gesetzt werden, wartet das Programm solange, bis der exec_os-Vektor angesprungen wird. Dann holt es sich die Adresse der Basepage. Der Zeiger auf den Environment hat einen Offset von \$2c auf die Basepage. Jetzt überschreibt System diesen Zeiger mit der Adresse von NEW_ENV. Damit sind die neuen Environment-Strings gesetzt.

System wurde auf einem ST Computer unter TOS 1.4 entwickelt, müßte aber mit jeder TOS-Version laufen die den Gemdos-Aufruf Ffirst zum Suchen der Acc benutzt. Das Programm sollte auch mit jeder 680xx CPU arbeiten. Als Assembler wurde der Makroassembler von Turbo C 2.0 verwendet.

Ralf Stacks

Literatur:

Jan Bolt: „Environment Strings“,
ST Computer 9/91

Julian Reschke: „Environment Variablen“,
ST Magazin 6/90

Eric Böhnisch: „Clipboard“,
ST Computer 7/8/90

Friedei van Megen: „Trashcan ST“,
ST Computer 12/91

Jankowski/Reschke/Rabich:
„Atari ST Profibuch“, Sybex Verlag

```

; * _____ *
; *   System           *
; *   by Ralf Stachs   *
; * (c) 1992 MAXON Computer *
; *-----*

#####
;# Größe des Environment Speichers #
#####

ENV_BUFFER equ 200

;TRAPS
;*****
GEMDOS equ 1
BIOS     equ 13
XBIOS equ 14

        TEXT
;Anfang des resident gehaltenen Speicher
RESI_ANFANG:

;zuerst System installieren
        jmp INSTALL

#####
;# Neue Gemdosroutine #
#####
;XBRA Protokoll der neuen Gemdosroutine
        dc.b "XBRA" ;XBRA Protokoll
        dc.b "RS23" ;eigene Erkennung
OLD_GEMDOS: dc.l 0      ;alter Vektor

NEW_GEMDOS:
;nach zweitem Ffirst aufruf NEW_GEMDOS ausgehängt
        tst.b COUNTER      ;Counter gleich 0
        beq ENDE           ;ja, dann ende

;Stack bestimmen
        move.l usp,a0      ;Zeiger auf Userstack (USP) holen

;Gemdos Aufruf aus Supervisor Modus?
        move.w (sp),d0      ;Statusreg. von Stack hol
        btst #13,d0        ;Supervisormodus ?
        beq USER          ;nein User Modus

;Zeiger auf Gemdosaufruf (SSP)
        move.l a7,a0        ;Stackpointer (SSP) holen
        move.w OFFSET,d0    ;Offset ab 68010
        lea 6(a0,d0.w),a0    ;Zeiger auf Gemdosaufruf

;Gemdosnummer holen
USER: move.w (a0)+,d0        ;Gemdosnummer holen
        cmp.w #78,d0        ;Ffirst Aufruf
        bne ENDE           ;Nein

;Suchname = '*.ACC'
        move.l (a0),a0      ;Zeiger auf Suchnamen
        cmp.b #'\',(a0)+    ;Suchname '\'

```

```

bne ENDE                ;nein dann ende
cmp.b #'*', (a0)+        ;Suchname '*'
bne ENDE                ;nein dann ende
cmp.b #'.', (a0)+        ;Suchname
bne ENDE                ;nein dann ende
cmp.b #'A', (a0)+        ;Suchname 'A'
bne ENDE                ;nein dann ende
cmp.b #'C', (a0)+        ;Suchname 'C'
bne ENDE                ;nein dann ende
cmp.b #'C', (a0)+        ;Suchname 'C'
bne ENDE                ;nein dann ende

```

;Gemdosaufruf Ffirst mit richtigem Suchname liegt vor

,*****

;Desktop aktuelles PRG ?

```

move.l  A_RUN,a0        ;Adresse auf Zeiger der aktuellen Basepage
move.l  (a0),a0          ;Zeiger auf Basepage
tst.l  $(a0)             ;Länge des Programmcode = 0
bne ENDE                ;nein, nicht Desktop

```

;Rücksprungadresse merken

```

move.l  2(a7),a0         ;Rücksprungadresse vom SSP holen
move.l  a0,RETURN        ;und sichern

```

;und neue Rücksprungadresse setzen

```

move.l  #MY_PATH,2(a7) ;Rücksprung zu eigener Routine

```

;Aufrufe Zählen

```

sub.b  #1,COUNTER        ;Counter herunterzählen

```

;alten Gemdosvektor anspringen

```

ENDE: move.l  OLD_GEMDOS,a0 ;alten Gemdosvektor laden
      jmp (a0)              ;und anspringen

```

;Eigene Routine nach Ffirst von GEM

,*****

MY_PATH:movem.1 d1-d7/a0-a6,-(sp) ;Register retten

;neuen Pfad setzen "\SYSTEM.ACC"

```

move.l  #PATH,-(sp)      ;Pfad "\SYSTEM.ACC"
move.w  #59,-(sp)        ;Dsetpath aufrufen
trap #GEMDOS
addq.l  #6,sp

```

;Directory nach ACC absuchen

;ersetzt den Original Ffirst (GEM) Aufruf

```

move.w  #0,-(sp)         ;normaler Dateityp
pea PATH_2               ;Pfad "\SYSTEM.ACC\*.ACC"
move.w  #78,-(sp)        ;Ffirst aufrufen
trap #GEMDOS
addq.l  #8,sp

```

```

movem.1 (sp)+,d1-d7/a0-a6 ;Register zurückschreiben

```

;zur alten Ffirst Routine (GEM) springen

```

move.l  RETURN,a0        ;alte Rücksprungadresse
jmp (a0)                 ;anspringen

```

,#####

;# setzen der Environment Strings #

,#####

```

;XBRA Protokoll für
    dc.b "XBRA"          ;XBRA Protokoll
    dc.b "RS23"          ;eigene Erkennung
OLD_EXEC_OS:dc.1 0      ;alter Vektor

NEW_EXEC_OS:
    move.l 4(sp),a0      ;Adresse BASEPAGE
    move.l #NEW_ENV,$2c(a0) ;neues Environment setzen

    move.l OLD_EXEC_OS,a0 ;alten Vektor
    jmp (a0)            ;anspringen

;Variablen und Flags
;*****
;OFFSET -> ab 68010 ein offset von 2 für Formate-Code
;RETURN -> Rücksprungadresse der Ffirst Routine (GEM)
;A_RUN -> Adresse von Zeiger auf aktueller Basepage
;COUNTER -> Anzahl der Aufrufe bis NEW_GEMDOS ausgehängt wird
OFFSET: dc.w      0
RETURN: dc.l      0
A_RUN:   dc.l      0
COUNTER: dc.b 2
         dc.b 0

    EVEN
    ;Pfad zum laden der ACC
PATH:   dc.b "\SYSTEM.ACC",0

    EVEN
    ;Pfad für Ffirst
PATH_2: dc.b      "\SYSTEM.ACC\*.ACC",0

    EVEN
    ;neue Environment Strings
NEW_ENV: dc.b "PATH=/?.:??:\SYSTEM.ACC",0,0
         ;Voreinstellung ohne SYSTEM.INF
         ds.b ENV_BUFFER ;200 Bytes für Environment
         dc.b -1,0       ;Ende des Buffers (-1)

;Ende des resident gehaltenen Speicher      ^

RESI_ENDE:

;#####
;# Installierung von System #
;#####
    TEXT
    EVEN

;gesamt-PRG Speicher belegen
INSTALL:
    move.l sp,a6      ;Adresse BASEPAGE
    lea    USTACK,sp  ;neuer Stack

    move.l 4(a6),a6    ;Speicher belegen
    move.l $c(a6),a4
    adda.l $14(a6),a4
    adda.l $1c(a6),a4

    pea    256(a4)
    pea    (a6)

```

```

clr.w    -(sp)
move.w   #74,-(sp)      ;Mshrink aufrufen

trap     #GEMDOS
lea      12(sp),sp

```

;Start aus Autoordner ? (AES anmelden)

```

lea contrl,a0          ;Adresse contrl nach a0
move.w #10,(a0)+       ;Opcode
clr.w (a0)+            ;einträge int_in
move.w #1,(a0)         ;einträge int_out
clr.w (a0)+            ;einträge addr_in
clr.w (a0)             ;einträge addr_out

```

```

move.l #aes_data,d1 ;Adresse AES-Array
move.w #$c8,d0      ;AES_Aufruf
trap #2

```

```

tst.w aes_global      ;starten aus AUTO-Ordner
beq SP_20             ;ja, keine ap_version

```

;vom desktop aus gestartet

;SYSTEM nur aus Auto Ordner starten

```

pea STRING_4
jmp ERROR

```

#####

;<# internes Setzen der Environment Strings #

#####

;aktuelles Laufwerk für SYSTEM.INF lesen

```

SP_20:   move.w #$19,-(sp) ;Dgetdrv aufrufen
trap #GEMDOS
addq.l #2,sp

```

```

add.b #"A",d0          ;Laufwerksbuchstabe berechnen
move.b d0,FNAME        ;Laufwerksbuchstabe in Pfad setzen
move.b d0,NEW_ENV+6    ;Voreinstellung für Environment Variable Path
move.b d0,NEW_ENV+10   ;Voreinstellung für Environment Variable Path

```

;SYSTEM.INF öffnen (?:\AUTO\SYSTEM.INF)

```

move.w #0,-(sp)        ;nur lesen
pea FNAME              ;Pfad mit Dateinamen
move.w #$3d,-(sp)      ;Fopen aufrufen
trap #GEMDOS
addq.l #8,sp

```

```

tst.w d0               ;SYSTEM.INF nicht vorhanden
bmi SP_21              ;ja, Rückgabewert negativ
move.w d0,FILE_HANDLE ;Handle merken

```

;SYSTEM.INF lesen

```

lea NEW_ENV,a6         ;Adresse der Environment Strings

```

```

SP_23:   tst.b (a6)      ;ende des Buffer (negativ)
bpl SP_28              ;nein

```

;Fehlermeldung ausgeben wenn ende des Puffer erreicht

```

pea STRING_5
move.w #9,-(sp)
trap #GEMDOS
addq.l #6,sp

```

```
;auf Taste warten
    move.w #2,-(sp)      ;von Tastatur
    move.w #2,-(sp)      ;Bconin
    trap #BIOS
    addq.l #4,sp
    bra SP_22            ; SYSTEM. INF schließen

SP_28:    move.l a6,-(sp) ;Adresse des Buffers
    move.l #1,-(sp)      ;Anzahl der Bytes
    move.w FILE_HANDLE,-(sp)
    move.w #$3f,-(sp)    ;Fread
    trap #GEMDOS
    lea $c(sp),sp

    tst.l d0             ;Ende von SYSTEM.INF
    beq SP_22            ;ja, kein Zeichen gelesen

;Daten von SYSTEM.INF in Environment schreiben
    move.b (a6),d0       ;Zeichen holen

    cmp.b #'?',d0        ;Bootlaufwerk ?
    bne SP_26            ;nein, dann weiter
    move.b FNAME,(a6)     ;ja, Bootlaufwerk eintragen

SP_26:    cmp.b #13,d0    ;Return
    beq SP_23            ;ja, Zeichen überschreiben

    cmp.b #10,d0         ;Line Feed
    bne SP_24            ;nein, Buffer plus 1
    move.b #0,(a6)       ;ja, dann ende des Strings

SP_24:    add.l #1,a6      ;Adresse des Buffers plus 1
    bra SP_23            ;nächstes Zeichen lesen

;Ende von SYSTEM.INF
SP_22:    move.b #0,(a6)   ;Doppel null als ende

;SYSTEM.INF schließen
    move.w FILE_HANDLE,-(sp)
    move.w #$3E,-(sp)     ;Fclose aufrufen
    trap #GEMDOS
    addq.l #4,sp

;zur Installierung in Supervisor-Modus
;*****

SP_21:    lea INIT_GEMDOS,a0 ;Adresse von INIT_GEMDOS nach a0
    pea (a0)
    move.w #$26,-(sp)     ;Supexec aufrufen
    trap #XBIOS
    addq.l #6,sp

;Installierung abgeschlossen und String ausgeben
    pea STRING_2
    move.w #9,-(sp)
    trap #GEMDOS
    addq.l #6,sp

;neue Gemdos-Routine resident im Speicher halten
;von RESI_ENDE bis RESI_ANFANG
```

```
clr.w -(sp)
pea RESI_ENDE-RESI_ANFANG+256
move.w #49,-(sp)      ;Ptermres aufrufen
trap #GEMDOS

; #####
; neue GEMDOS-Routine im Supervisor-Modus installieren
; #####
;Adresse von Kbshift ermitteln
;und bei gedrückter Control Taste abbrechen
INIT_GEMDOS:
    move.l #$e1b,a1      ;vorgabe für TOS 1.0 (Kbshift)
    move.l #$602c,A_RUN ;vorgabe für TOS 1.0 (Basepage)
    move.l $4f2,a0       ;sysbase a0 = Anfangsadresse des Betriebssystem

    cmp.w #$0100,2(a0)   ;TOS 1.0
    beq SP_9            ;ja

    move.l 36(a0)/a1      ;(a1) Adresse von Kbshift
    move.l 40(a0),A_RUN ;Adresse auf Zeiger der aktuellen Basepage

SP_9:    btst.b #2,(a1)   ;Control Taste gedrückt
    beq SP_12           ;nein, dann weiter

;Installieren durch drücken der Control Taste abgebrochen
    pea STRING_3
    jmp ERROR

;System schon installiert ?
;*****
SP_12:    move.l $84,a0    ;Adresse des Gemdosvektor nach a0

SP_1:    cmp.l #"RS23",-8(a0) ;System schon vorhanden
    beq SP_2            ;ja, System dann ende

    cmp.l #"XBRA",-12(a0) ;XBRA Kennung
    bne SP_4            ;nein, dann System installieren

    move.l -4(a0),a1      ;Adresse der nächsten Gemdosvektors
    move.l a1,a0          ;von a1 nach a0
    bra SP_1            ;weiter

;System war schon installiert
;*****
;Meldung System schon installiert
SP_2:    pea STRING_1
    jmp ERROR

;System installieren
;*****
SP_4:    move.l $84,OLD_GEMDOS ;alten Gemdosvektor sichern
    move.l #NEW_GEMDOS,$84    ;und neuen Gemdosvektor setzen

    cmp.b #'-',NEW_ENV        ;Environment setzen
    beq SP_8                  ;nein, erstes Zeichen dann nicht setzen

    move.l $4fe,OLD_EXEC_OS ;alten exec_os Vektor sichern
    move.l #NEW_EXEC_OS,$4fe ;und neuen exec_os Vektor setzen

;Prozessor feststellen
SP_8:    move.l $5a0,d0      ;_p_cookies laden Zeiger vorhanden
```

```
        beq _68000          ;nein, dann 68000
        move.l d0,a0        ;Zeiger auf Cookies in a0

SP_31:   move.l (a0)+,d1 ;Cookie ID-Code in d1
        beq _68000          ;das war der letzte dann 68000
        cmp.l #"_CPU",d1    ;CPU Cookie suchen
        beq SP_30           ;gefunden
        add.l #4,a0         ;Cookie Wert überspringen
        bra SP_31           ;weiter suchen

;CPU Cookie gefunden
SP_30:   tst.l (a0)          ;Cookie Wert testen
        beq _68000          ;Cookie Wert = 0 dann 68000

        move.w #2,OFFSET    ;ab 68010 offset von 2

_68000: rts

;#####
;# Fehlermeldung ausgeben und auf Taste warten #
;#####
ERROR:   move.w #9,-(sp)
        trap #GEMDOS
        addq.l #6,sp

;auf Taste warten
        move.w #2,-(sp)      ;von Tastatur
        move.w #2,-(sp)      ;Bconin
        trap #BIOS
        addq.l #4,sp

;PRG beenden
        clr.w -(sp)
        trap #GEMDOS

;#####
;# Datensegment #
;#####
        DATA
        ;FILE_HANDLE -> Handle von SYSTEM.INF
FILE_HANDLE: dc.w 0

        EVEN
        ;Pfad von SYSTEM.INF
FNAME:    dc.b "A:\AUTO\SYSTEM.INF",0

        EVEN
STRING_1: dc.b 13,10,"*****"
        dc.b 13,10,"*** System ist schon installiert "
        dc.b 13,10,"*** Taste drücken"
        dc.b 13,10,"*****",13,10,0

        EVEN
STRING_2: dc.b 27,"p"
        dc.b 13,10,"+*****+"
        dc.b 13,10,"+ System 1.0 +"
        dc.b 13,10,"+      +"
        dc.b 13,10,"+ Ralf Stachs +"
        dc.b 13,10,"+ ST Computer +"
        dc.b 13,10,"+*****+"
        dc.b 13,10,27,"q",0
```

```
    EVEN
STRING_3:    dc.b 27,"p"
            dc.b 13,10,"*****"
            dc.b 13,10,"*** System wird nicht installiert "
            dc.b 13,10,"*** Taste drücken"
            dc.b 13,10,"*****",13,10
            dc.b 27,"q",0

    EVEN
STRING_4:    dc.b 27,"E"
            dc.b 13,10,"*** SYSTEM.PRG nur aus dem Auto Ordner starten"
            dc.b 13,10,"*** Taste drücken",13,10,0

    EVEN
STRING_5:    dc.b 27,"p"
            dc.b 13,10,"*****"
            dc.b 13,10,"*** Environment Speicher voll max. 224 Zeichen"
            dc.b 13,10,"*** Taste drücken"
            dc.b 13,10,"*****",13,10
            dc.b 27,"q",0


    EVEN
aes_data:    dc.1 contrl
            dc.1 aes_global
            dc.1 init_in
            dc.l init_out
            dc.1 addr_in
            dc.1 addr_out


;#####
;# BSS-Segment #
;#####

    EVEN
    BSS

aes_global: ds.w 15
contrl:     ds.w 10
init_in:    ds.w 128
init_out:   ds.w 128
addr_in:    ds.1 128
addr_out:   ds.l 128


            ds.b 256
sUSTACK:    ds.w 0
```

Ralf Stacks