In die Geheimnisse der Floppy eingetaucht (Teil 7)

Dieser Teil zeigt wie das DOS codiert und physikalisch auf Diskette schreibt. Haben Sie gewußt, daß ein Block auf Diskette länger als 256 Byte ist?

Dieser Artikel bildet den Abschluß der festen Kursreihe. Ab jetzt soll der Floppy-Kurs in lockerer Reihenfolge fortgesetzt werden. Wir werden nun versuchen, auf die von Ihnen bevorzugten Themen einzugehen, um darüber dann in unregelmäßigen Abständen weitere Folgen zu bringen. Wir wollen Sie deshalb dazu anregen, uns Ihre Probleme und Sorgen mit der 1541 mitzuteilen.

An dieser Stelle soll aber noch ein wichtiger Sachverhalt besprochen werden, der bisher einfach unterschlagen wurde: die GCR-Codierung.

Was ist eine GCR-Codierung?

Wenn Sie den Floppy-Kurs schon länger verfolgen sind Ihnen bestimmt schon einige Ungereimtheiten aufgefallen, was den Direktzugriff auf die Diskette betrifft. Auch in der letzten Folge über das Formatieren waren zum Beispiel im Listing von S-Format einige Sprungbefehle, die nicht erklärt wurden.

Erinnern Sie sich noch an den Artikel, der sich das erste mal mit dem Schreiben von Daten auf die Diskette beschäftigte? Dort wurden, unter anderem, die SYNC-Markierungen auf der Diskette besprochen, die dem Disk-Controller als Positionsanzeiger dienen.

Ich schrieb damals, daß sich diese SYNC-Markierungen bei der 1541 aus fünf $FF-Bytes zusammensetzen, die hintereinander auf Diskette geschrieben werden. Was ist aber, wenn ein Datenblock geschrieben werden soll, der nur aus $FF-Bytes besteht? Eigentlich müßten dann diese Bytes als SYNC-Markierung wirksam werden und den gesamten Schreib- und Lesebetrieb stören. Wie die Praxis zeigt, tritt dieser Fehler nicht auf. Auch bei mehreren Blöcken aus $FF-Bytes kommt es zu keinen Komplikationen. Bei der Konstruktion der Floppy hat man sich nämlich eine Codierung der Daten einfallen lassen, die eine Eindeutigkeit der Daten schafft. Die Codierung heißt GCR, was nichts anderes als eine Abkürzung der englischen Wörter »Group Code Recording« ist.

Es stellt sich jetzt natürlich die Frage, was bei der GCR-Codierung passiert, damit eine Verwechselung zwischen SYNC- und Datenbytes unmöglich wird. Zur Beantwortung dieser Frage muß ein wenig intensiver auf das Lesen und Schreiben der Floppy eingegangen werden.

Was macht die GCR-Codierung

Das Lesen von Bytes durch den Lesekopfes steuert ein Timer des Disk-Controllers. Auf der Diskette selbst wird jedes »1«-Bit physikalisch durch einen Wechsel der Magnetisierungsrichtung dargestellt und ein »0«-Bit durch gleichbleibende Richtung der Magnetisierung. Bild 1 zeigt, was gemeint ist. Soll ein Byte von Diskette gelesen werden, so wartet der Disk-Controller einfach die Zeitspanne ab, die zum Lesen von acht Bits erforderlich ist. Innerhalb dieser Zeit liest der Schreib-/Lesekopf eine gewisse Folge von Magnetisierungswechseln und Nicht-Magnetisierungswechseln.

Dazu ein Beispiel: Auf der Diskette steht ein $55-Byte. $55 wird binär durch die Kombination %01010101 dargestellt. Der Tonkopf stellt also während der Lesezeit die folgenden Magnetisierungswechsel fest:

Magnetisierung wechselt nicht, wechselt, wechselt nicht, wechselt, wechselt nicht, wechselt, wechselt nicht, wechselt.

Das Erkennen eines Bits geschieht dabei völlig zeitgesteuert. Der Disk-Controller »weiß«, daß er zum Lesen eines Bits eine bestimmte Zeit warten muß. Danach gilt das Bit als gelesen, und es wird eine »1« oder eine »0« bereitgestellt, je nachdem, ob ein Magnetisierungswechsel stattgefunden hat oder nicht.

Praktisch könnte man das folgendermaßen beschreiben: Sie machen mit einem Freund eine Zeit von 10 Sekunden aus. Er hat dann die Aufgabe innerhalb dieser 10 Sekunden entweder zu pfeifen oder nicht. Danach warten Sie diese 10 Sekunden ab. Hat er während dieser Zeit gepfiffen, dann entspricht das einem Magnetisierungswechsel. Hat er innerhalb der 10 Sekunden nicht gepfiffen, bedeutet das ein »0«-Bit, also keinen Magnetisierungswechsel. Da eine Diskette im Laufwerk nicht absolut gleichmäßig gedreht werden kann, also Drehzahlschwankungen unterliegt, muß noch für eine Kompensation der mechanischen Fehler gesorgt werden. Dazu wird der Timer, der die abzuwartende Zeit für jedes Bit bestimmt, bei jedem Magnetisierungswechsel neu getriggert (gestellt). Ein »1«-Bit hat also neben seinem Informationsgehalt noch die wichtige Aufgabe, Laufwerksschwankungen auszugleichen, um Lesefehler zu verhindern. Aus diesem Grund darf es zum Beipiel nicht passieren, daß mehrere $00-Bytes hintereinander auf der Diskette stehen, da sonst zu lange keine Laufwerkskontrolle mehr stattfinden könnte.

Aber auch zuviele »1«-Bits sind nicht gestattet, da mehr als acht »1«-Bits ein SYNC-Signal auslösen.

Aus den genannten Gründen werden alle Daten, die auf die Diskette geschrieben werden, vorher GCR-codiert. Mit dieser Codierung wird ausgeschlossen, daß mehr als acht »1«-Bits und mehr als zwei »0«-Bits direkt hintereinander auf die Diskette geschrieben werden und so die Schreib- und Lese-Elektronik durcheinanderbringen.

Einzig und allein die SYNC-Markierungen (mehr als acht »1«-Bits) werden vom DOS (Disk Operating System, Controller) uncodiert auf die Diskette geschrieben.

Es gibt zwei Schreibarten

Man kann also zwischen zwei Schreibarten auf Diskette unterscheiden:

1) Schreiben von Markierungen. Hier werden fünf $FF-Bytes direkt hintereinander auf die Diskette geschrieben, um eine SYNC-Markierung zu bilden, die der Orientierung dient.

2) Schreiben von Daten.

In diesem Modus werden Byte-Inhalte codiert, um sich von den Markierungen zu unterscheiden.

Sehen Sie sich jetzt einmal Tabelle 1 an, die Umwandlungstabelle für die Konvertierung Binär nach GCR und umgekehrt.

Wie Sie unschwer erkennen können, handelt es sich beim GCR-Code um einen 5-Bit-Code. Jedes 4-Bit-Nibble, das Sie umwandeln, wird zu einem 5-Bit-GCR-Nibble. Ein Byte, das vorher aus 8 Bits bestand wird also durch die Codierung 10 Bits lang. Allgemein nimmt die Länge der codierten Daten, um den Faktor 5/4 zu. Deshalb ist die Handhabung der GCR-Bytes nicht ganz einfach. Wandeln Sie doch einmal zwei Bytes in den GCR-Code um. Als Ergebnis erhalten Sie »zweieinhalb« Bytes, die sicherlich schwer zu behandeln sind.

Bei der GCR-Codierung geht man aus diesem Grund einen ganz einfachen Weg, um keine Format-Probleme zu bekommen: Es werden jeweils immer 4 Bytes gleichzeitig umgewandelt. Als Ergebnis erhält man 5 vollständige Bytes, die sich ohne Probleme weiterverarbeitet werden können.

Lassen Sie mich das einmal an einem Beipiel erläutern:

Nehmen wir einmal an, wir hätten vier Bytes mit dem Wert $FF. Eine Kombination also, die nicht direkt auf die Diskette geschrieben werden darf.

Wir wandeln diese vier Hex-Bytes nun in die entsprechenden fünf GCR-Bytes um, indem wir in Tabelle 1 nachsehen, was die entsprechenden GCR-Äquivalente dieser Bytes sind. Wir kommen zu folgendem Ergebnis:

HEX	BINÄR	GCR-Code
$FF	1111 1111	10101 10101
$FF	1111 1111	10101 10101
$FF	1111 1111	10101 10101
$FF	1111 1111	10101 10101

Die binär dargestellten GCR-Werte müssen wir jetzt nur noch zu fünf Bytes zusammenfassen, um auf folgendes Ergebnis zu kommen:

1010 + 1101 = AD
(1010 1+101 01)
0110 + 1011 = 6B
0101 + 1010 = 5A
1101 + 0110 = D6
1011 + 0101 = B5

Vier $FF-Bytes werden also bei der GCR-Codierung in die fünf Bytes $AD, $6B, $5A, $D6 und $B5 umgewandelt, und Sie können sich jetzt leicht davon überzeugen, daß diese fünf Bytes für den Disk-Controller absolut ungefährlich und unkritisch

sind, und daß sie die vorgeschriebenen Normen (nicht mehr als zwei »0«-Bytes und nicht mehr als acht »1«-Bytes) erfüllen.

Um Ihnen die Umwandlung der Bytes zu erleichtern, habe ich diesem Kurs zwei Programmlistings beigefügt. Listing 1 enthält ein Programm, das Ihnen vier Hex-Bytes in fünf GCR-Bytes umwandelt. In Listing 2 sehen Sie ein Programm abgedruckt, das die GCR-Codierung wieder rückgängig macht. Hier werden fünf GCR-Bytes in vier Hex-Bytes zurückverwandelt, wobei Sie mit unerlaubten Bitkombinationen vorsichtig sein sollten. Kann ein Byte nicht zurückverwandelt werden, so haben Sie eine unerlaubte GCR-Bitkombination, die sich im Ergebnis dadurch äußert, daß entsprechende Nibbles fehlen. Sie erhalten dann unter Umständen nur »halbe« Bytes.

Die Floppy hält übrigens für diesen Fall eine Fehlermeldung bereit, einen »24, READ ERROR«.

Damit Sie auch in Maschinensprache in der Lage sind, Hex-GCR-Konvertierungen durchzuführen, ist noch ein weiteres Listing (Listing 3) beigefügt. Dieses enthält die Originalroutinen des DOS zur Umwandlung von Hex-Bytes in GCR-Bytes und umgekehrt.

$F6D0: Dieses Programm holt vier Hex-Bytes aus den Speicherstellen $52 bis $55 und wandelt diese Bytes in die fünf entsprechenden GCR-Werte um. Diese fünf Bytes werden anschließend im Puffer der Adresse $30/31 (L,H) mit dem Pufferzeiger in $34 abgelegt.

Pufferadresse und Pufferzeiger müssen dabei vor Aufruf dieser Routine übergeben werden.

$F78F: Diese Routine wandelt einen gesamten Puffer, dessen Adresse in $30/31 (L,H) stehen muß, in GCR-Werte um und speichert diese in den Ausweichpuffer sowie den ursprünglichen Puffer zurück. Der Pufferinhalt vergrößert sich durch diese Umwandlung von 256 auf 324 Bytes.

$F7E6: Diese Routine wandelt fünf GCR-Bytes aus einem Puffer (dessen Adresse in $30/31 (L,H) und dessen Pufferzeiger in $34 steht) wieder in vier Hex-Bytes zurück, wobei diese dann in der Zeropage von $52 bis $55 abgespeichert werden.

$F8E0: Diese Routine decodiert einen gesamten GCR-Pufferinhalt in die ursprüngliche Form und legt diese 256 Bytes dann im Puffer mit der Adresse $30/31 (L,H) ab. Die vorherigen 324 GCR-Bytes müssen im gleichen Puffer und im Ausweichpuffer ($01BB bis $01FF) stehen.

Die Anwendungen dieser Routinen sind äußerst vielfältig. So können Sie diese Programme zum Beispiel für einen Disk-Monitor verwenden, in dem man zwischen der Anzeige von GCR-Bytes und der Anzeige von normalen Hex-Bytes hin- und herschalten kann. Die einzigen Änderungen, die Sie dazu machen müssen, bestehen in der Umrechnung der Adressen für die Speicherbereiche im Computer und der Angabe neuer Parameter als Puffer- und Zeropagebereiche. Ihrer Phantasie, was die Möglichkeiten des Monitors angeht, sind außer dem Speicherplatz im Computer keine Grenzen gesetzt.

Bis zu 365 Byte in einem Block

Durch die Verwendung der GCR-Codierung ergeben sich noch Konsequenzen. Wie sieht es beispielsweise in den Puffern der Floppy aus, wenn ein Puffer mit einem vollständigen Datenblock (also 256 Bytes) gefüllt wurde und dieser aufgezeichnet werden soll? Für dieses Problem hat der Controller einen speziellen Ausweichpuffer. Der Puffer hat eine Größe von 68 Bytes und befindet sich im Bereich von $01BB bis $01FF.

Wird nun ein Datenblock in Puffer 1 ($0400-$04FF) codiert, so werden die ersten 68 GCR-Bytes in den Ausweichpuffer übernommen. Die restlichen Bytes stehen in Puffer 1.

Aus den 256 Bytes an Information macht das DOS durch die Konvertierung also 324 Bytes, die einen gesamten Datenblock darstellen (inklusive Prüfsumme). Natürlich werden auch die Parameter im Datenblockheader (ID, Track, Sektor, Prüfsumme und Kennzeichen) vor dem Schreiben auf die Diskette in GCR-Bytes umgewandelt, wobei der Blockheader dann mit den zwei Lückenbytes auf eine Länge von zehn GCR-Bytes anwächst, da der Header aus ursprünglich acht Hex-Werten besteht.

Zusammenfassend besteht ein Sektor auf der Diskette aus den fünf Bytes der ersten SYNC-Markierung; danach folgen die zehn Bytes des Blockheaders. Vor der SYNC-Markierung des Datenblocks folgen jedoch noch neun $55-Bytes, die der GCR-Norm entsprechen und direkt auf die Diskette geschrieben werden. Sie dienen als Pufferlücke, in der dem Disk-Controller Zeit bleibt, zwischen Schreiben und Lesen umzuschalten.

Nach den fünf Bytes der SYNC-Markierung folgen die 324 Bytes des Datenblocks inklusive dessen Prüfsumme und anschließend noch die Lücke zwischen zwei Sektoren, die erfahrungsgemäß zwischen acht und zwölf Bytes lang ist. Wie Sie sehen hat also so ein Sektor auf der Diskette die stattliche Länge von 361 bis 365 Bytes.

Jetzt werden Ihnen bestimmt auch ein paar zweifelhafte JSR-Befehle in der letzten Folge des Floppy-Kurses klar: bei dem Formatiersystem in Ausgabe 5/1985, wird einmal ein Befehl JSR $FE30 und an anderer Stelle ein Befehl JSR $F78F ausgeführt. Diese Adressen sind die Einsprünge der Codier-Routinen.

Vielleicht kommt Ihnen auch noch einmal die Herstellung eines Killertracks in Erinnerung. Hier wird ein gesamter Track direkt mit $FF-Bytes vollgeschrieben und stellt so eine »Riesen-SYNC-Markierung« dar. Da eine solche Bitfolge jedoch unzulässig ist, kommt die Lese-und Schreibelektronik der Floppy völlig aus dem Konzept; der Controller »stürzt ab«.

Wenn Sie noch mehr über Ihre 1541, über schnelle Kopierprogramme und Kopierschutz-Methoden erfahren oder ein gut dokumentiertes DOS-Listing haben wollen, dann sollten Sie einmal in das M&T Floppy-Buch schauen.(Karsten Schramm/hm)

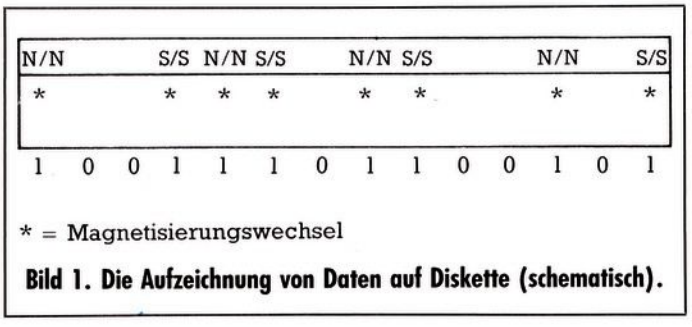

Bild 1. Die Aufzeichnung von Daten auf Diskette (schematisch).

```
10 REM PROGRAMM ZUR KONVERTIERUNG                <242>
20 REM VON FUENF GCR-BYTES IN DIE                <003>
30 REM VIER ENTSPRECHENDEN                       <249>
40 REM HEX-AEQUIVALENTE                          <077>
50 REM                                           <193>
60 REM                                           <203>
70 REM                                           <213>
80 REM (W) 1985 BY KARSTEN SCHRAMM               <028>
90 REM                                           <233>
100 A$="0123456789ABCDEF":DIM G$(15):E$=""       <220>
110 G$(0)="01010"                                <066>
120 G$(1)="01011"                                <078>
130 G$(2)="10010"                                <088>
140 G$(3)="10011"                                <100>
150 G$(4)="01110"                                <111>
160 G$(5)="01111"                                <123>
170 G$(6)="10110"                                <133>
180 G$(7)="10111"                                <145>
190 G$(8)="01001"                                <154>
200 G$(9)="11001"                                <166>
210 G$(10)="11010"                               <216>
220 G$(11)="11011"                               <228>
230 G$(12)="01101"                               <238>
240 G$(13)="11101"                               <250>
250 G$(14)="11110"                               <005>
260 G$(15)="10101"                               <016>
270 PRINT"{CLR}GCR - HEX - KONVERTIERUNG":
    PRINT                                        <096>
280 PRINT:PRINT"GEBEN SIE JETZT 5 GCR-BYTE
    S EIN":PRINT                                 <016>
290 INPUT"{2DOWN}";H$:GC$=""                     <147>
300 X$="":FOR X=1 TO LEN(H$)                     <185>
310 IF MID$(H$,X,1)<>" "THEN X$=X$+MID$(H$
    ,X,1)                                        <053>
320 NEXT                                         <195>
330 H$=X$                                        <229>
340 FOR X=1 TO 10                                <022>
350 X$=MID$(H$,X,1)                              <245>
360 XX=VAL(X$):IF XX=0 AND X$<>"0"THEN XX=
    ASC(X$)-55                                   <104>
370 FOR Y=0 TO 3                                 <006>
380 YY=INT(XX/2↑(3-Y)):XX=XX-YY*2↑(3-Y)          <105>
390 IF YY THEN GC$=GC$+"1":GOTO 410              <240>
400 GC$=GC$+"0"                                  <189>
410 NEXT Y,X                                     <250>
420 HC$="":FOR X=1 TO 8                          <028>
430 X$=MID$(GC$,X*5-4,5)                         <075>
440 FOR Y=0 TO 15                                <127>
450 IF X$<>G$(Y)THEN NEXT Y                      <197>
460 :                                            <007>
470 HC$=HC$+MID$(A$,Y+1,1)                       <206>
480 IF INT(X/2)=X/2 THEN HC$=HC$+" "             <055>
490 NEXT X                                       <197>
500 PRINT:PRINT:PRINT"HEX: ";HC$                 <129>
```

Listing 1. Umwandlung von Daten in GCR-Bytes

```
10 REM PROGRAMM ZUR KONVERTIERUNG                <242>
20 REM VON VIER HEXBYTES IN DIE                  <161>
30 REM FUENF ENTSPRECHENDEN                      <055>
40 REM GCR-AEQUIVALENTE                          <068>
50 REM                                           <193>
60 REM                                           <203>
70 REM                                           <213>
80 REM (W) 1985 BY KARSTEN SCHRAMM               <028>
90 REM                                           <233>
100 A$="0123456789ABCDEF":DIM G$(15):E$=""       <220>
110 G$(0)="01010"                                <066>
120 G$(1)="01011"                                <078>
130 G$(2)="10010"                                <088>
140 G$(3)="10011"                                <100>
150 G$(4)="01110"                                <111>
160 G$(5)="01111"                                <123>
170 G$(6)="10110"                                <133>
180 G$(7)="10111"                                <145>
190 G$(8)="01001"                                <154>
200 G$(9)="11001"                                <166>
210 G$(10)="11010"                               <216>
220 G$(11)="11011"                               <228>
230 G$(12)="01101"                               <238>
240 G$(13)="11101"                               <250>
250 G$(14)="11110"                               <005>
260 G$(15)="10101"                               <016>
270 PRINT"{CLR}HEX - GCR - KONVERTIERUNG":
    PRINT                                        <096>
280 PRINT:PRINT"GEBEN SIE JETZT 4 HEXBYTES
     EIN":PRINT                                  <235>
290 PRINT"Z.B. ED 34 27 58":INPUT"{2DOWN}"
    ;H$:GC$=""                                   <104>
300 GOSUB 470:FOR X=1 TO 4                       <035>
310 H1$=MID$(H$,X*2-1,1):H2$=MID$(H$,X*2,1
    )                                            <120>
320 H1=VAL(H1$):H2=VAL(H2$)                      <057>
330 IF H1=0 AND H1$<>"0"THEN H1=ASC(H1$)-5
    5                                            <240>
340 IF H2=0 AND H2$<>"0"THEN H2=ASC(H2$)-5
    5                                            <254>
350 GC$=GC$+G$(H1)+G$(H2)                        <044>
360 NEXT X                                       <067>
370 FOR X=1 TO 10                                <052>
380 B=0:B$=MID$(GC$,X*4-3,4)                     <094>
390 FOR Y=0 TO 3                                 <026>
400 IF MID$(B$,Y+1,1)="1"THEN B=B+2↑(3-Y)        <112>
410 NEXT Y                                       <118>
420 E$=E$+MID$(A$,B+1,1)                         <249>
430 IF X/2=INT(X/2)THEN E$=E$+" "                <121>
440 NEXT X                                       <147>
450 PRINT:PRINT:PRINT"GCR: ";E$                  <000>
460 END                                          <077>
470 X$="":FOR X=1 TO LEN(H$)                     <099>
480 IF MID$(H$,X,1)<>" "THEN X$=X$+MID$(H$
    ,X,1)                                        <223>
490 NEXT                                         <109>
500 H$=X$:RETURN                                 <087>
```

Listing 2. Umwandlung von GCR- in Daten-Bytes

Hexa-dezimal	Binär	GCR
$0	0000	01010
$1	0001	01011
$2	0010	10010
$3	0011	10011
$4	0100	01110
$5	0101	01111
$6	0110	10110
$7	0111	10111
$8	1000	01001
$9	1001	11001
$A	1010	11010
$B	1011	11011
$C	1100	01101
$D	1101	11101
$E	1110	11110
$F	1111	10101

Tabelle 1. Umrechnungstabelle für Binär-GCR-Umwandlung

Listing 3. Die DOS-Routinen zur GCR-Codierung

```
Umwandlung von 4 Daten-Bytes
in 5 GCR-Bytes
f6d0 a9 00      lda  #$00
f6d2 85 57      sta  $57
f6d4 85 5a      sta  $5a
;    Pufferzeiger holen
f6d6 a4 34      ldy  $34
;    erstes Byte holen
f6d8 a5 52      lda  $52
f6da 29 f0      and  #$f0
f6dc 4a         lsr
f6dd 4a         lsr
f6de 4a         lsr
f6df 4a         lsr
f6e0 aa         tax
;    und anhand der Tabelle
;    umwandeln
f6e1 bd 7f f7   lda  $f77f,x
f6e4 0a         asl
f6e5 0a         asl
f6e6 0a         asl
f6e7 85 56      sta  $56
f6e9 a5 52      lda  $52
f6eb 29 0f      and  #$0f
f6ed aa         tax
f6ee bd 7f f7   lda  $f77f,x
f6f1 6a         ror
f6f2 66 57      ror  $57
f6f4 6a         ror
f6f5 66 57      ror  $57
f6f7 29 07      and  #$07
f6f9 05 56      ora  $56
;    Byte in Puffer schreiben
f6fb 91 30      sta  ($30),y
f6fd c8         iny
;    zweites Byte holen
f6fe a5 53      lda  $53
f700 29 f0      and  #$f0
f702 4a         lsr
f703 4a         lsr
f704 4a         lsr
f705 4a         lsr
f706 aa         tax
f707 bd 7f f7   lda  $f77f,x
;    und codieren
f70a 0a         asl
f70b 05 57      ora  $57
f70d 85 57      sta  $57
f70f a5 53      lda  $53
f711 29 0f      and  #$0f
f713 aa         tax
f714 bd 7f f7   lda  $f77f,x
f717 2a         rol
f718 2a         rol
f719 2a         rol
f71a 2a         rol
f71b 85 58      sta  $58
f71d 2a         rol
f71e 29 01      and  #$01
f720 05 57      ora  $57
;    in Puffer schreiben
f722 91 30      sta  ($30),y
f724 c8         iny
;    drittes Byte holen
f725 a5 54      lda  $54
f727 29 f0      and  #$f0
f729 4a         lsr
f72a 4a         lsr
f72b 4a         lsr
f72c 4a         lsr
f72d aa         tax
;    codieren und in
f72e bd 7f f7   lda  $f77f,x
f731 18         clc
f732 6a         ror
f733 05 58      ora  $58
;    Puffer schreiben
f735 91 30      sta  ($30),y
f737 c8         iny
f738 6a         ror
f739 29 80      and  #$80
f73b 85 59      sta  $59
f73d a5 54      lda  $54
f73f 29 0f      and  #$0f
f741 aa         tax
f742 bd 7f f7   lda  $f77f,x
f745 0a         asl
f746 0a         asl
f747 29 7c      and  #$7c
f749 05 59      ora  $59
f74b 85 59      sta  $59
;    viertes Byte holen
f74d a5 55      lda  $55
f74f 29 f0      and  #$f0
f751 4a         lsr
f752 4a         lsr
f753 4a         lsr
f754 4a         lsr
f755 aa         tax
;    codieren und in
f756 bd 7f f7   lda  $f77f,x
f759 6a         ror
f75a 66 5a      ror  $5a
f75c 6a         ror
f75d 66 5a      ror  $5a
f75f 6a         ror
f760 66 5a      ror  $5a
f762 29 03      and  #$03
f764 05 59      ora  $59
;    Puffer schreiben
f766 91 30      sta  ($30),y
f768 c8         iny
f769 d0 04      bne  $f76f
f76b a5 2f      lda  $2f
f76d 85 31      sta  $31
f76f a5 55      lda  $55
f771 29 0f      and  #$0f
f773 aa         tax
f774 bd 7f f7   lda  $f77f,x
f777 05 5a      ora  $5a
:    Überlaufbyte ebenfalls
;    in Puffer schreiben
f779 91 30      sta  ($30),y
f77b c8         iny
;    Pufferzeiger merken
f77c 84 34      sty  $34
f77e 60         rts
;    Tabelle für die
;    Umwandlung
;    von HEX nach GCR-
     Werten
f77f 0a 0b 12 13 0e 0f 16 17
f787 09 19 1a 1b 0d 1d 1e 15
;    Pufferdaten codieren
     Pufferadresse Lo
f78f a9 00      lda  #$00
f791 85 30      sta  $30
f793 85 2e      sta  $2e
;    Pufferzeiger 2 setzen
f795 85 36      sta  $36
;    Pufferzeiger für
f797 a9 bb      lda  #$bb
;    Ausweichpuffer setzen
f799 85 34      sta  $34
f79b 85 50      sta  $50
;    Pufferadresse Hi
f79d a5 31      lda  $31
;    merken
f79f 85 2f      sta  $2f
;    Pufferadresse auf
f7a1 a9 01      lda  #$01
;    Ausweichpuffer
f7a3 85 31      sta  $31
;    Datenblockkennz. $07
;    für Kodierung
;    vorbereiten
f7a5 a5 47      lda  $47
f7a7 85 52      sta  $52
f7a9 a4 36      ldy  $36
;    Bytes aus normalem
;    Puffer
f7ab b1 2e      lda  ($2e),y
;    in Speicher für
;    Codierung
f7ad 85 53      sta  $53
f7af c8         iny
f7b0 b1 2e      lda  ($2e),y
f7b2 85 54      sta  $54
f7b4 c8         iny
f7b5 b1 2e      lda  ($2e),y
f7b7 85 55      sta  $55
f7b9 c8         iny
;    Pufferzeiger retten
f7ba 84 36      sty  $36
;    Bytes kodieren und in
;    Ausweichpuffer
;    schreiben
f7bc 20 d0 f6   jsr  $f6d0
f7bf a4 36      ldy  $36
;    weitere Bytes aus
;    Puffer in
f7c1 b1 2e      lda  ($2e),y
;    Speicher für Codierung
f7c3 85 52      sta  $52
f7c5 c8         iny
f7c6 f0 11      beq  $f7d9
f7c8 b1 2e      lda  ($2e),y
f7ca 85 53      sta  $53
f7cc c8         iny
f7cd b1 2e      lda  ($2e),y
```

```
f7cf  85 54     sta  $54
f7d1  c8        iny
f7d2  b1 2e     lda  ($2e),y
f7d4  85 55     sta  $55
f7d6  c8        iny
;     und ggf. codieren
f7d7  d0 e1     bne  $f7ba
;     Prüfsumme Datenblock
f7d9  a5 3a     lda  $3a
;     ebenfalls codieren und
;     in Puffer schreiben
f7db  85 53     sta  $53
f7dd  a9 00     lda  #$00
f7df  85 54     sta  $54
;     Leerbytes zum Auffüllen
f7e1  85 55     sta  $55
;     ebenfalls codieren
f7e3  4c d0 f6  jmp  $f6d0
```

Umwandlung von 5 GCR-Bytes in 4 Daten-Bytes

```
;     Pufferzeiger holen
f7e6  a4 34     ldy  $34
;     Byte aus Puffer holen
f7e8  b1 30     lda  ($30),y
f7ea  29 f8     and  #$f8
f7ec  4a        lsr
f7ed  4a        lsr
f7ee  4a        lsr
f7ef  85 56     sta  $56
;     und decodieren;
;     zwischen-
;     speichern für späteres
f7f1  b1 30     lda  ($30),y
f7f3  29 07     and  #$07
f7f5  0a        asl
f7f6  0a        asl
;     zusammenschieben
f7f7  85 57     sta  $57
f7f9  c8        iny
f7fa  d0 06     bne  $f802
f7fc  a5 4e     lda  $4e
f7fe  85 31     sta  $31
f800  a4 4f     ldy  $4f
;     nächstes Byte holen
f802  b1 30     lda  ($30),y
f804  29 c0     and  #$c0
f806  2a        rol
f807  2a        rol
f808  2a        rol
f809  05 57     ora  $57
f80b  85 57     sta  $57
;     codieren und
f80d  b1 30     lda  ($30),y
f80f  29 3e     and  #$3e
f811  4a        lsr
;     zwischenspeichern
f812  85 58     sta  $58
f814  b1 30     lda  ($30),y
f816  29 01     and  #$01
f818  0a        asl
f819  0a        asl
f81a  0a        asl
f81b  0a        asl
f81c  85 59     sta  $59
f81e  c8        iny
f81f  b1 30     lda  ($30),y
f821  29 f0     and  #$f0
f823  4a        lsr
f824  4a        lsr
f825  4a        lsr
f826  4a        lsr
f827  05 59     ora  $59
f829  85 59     sta  $59
f82b  b1 30     lda  ($30),y
f82d  29 0f     and  #$0f
f82f  0a        asl
f830  85 5a     sta  $5a
f832  c8        iny
f833  b1 30     lda  ($30),y
f835  29 80     and  #$80
f837  18        clc
f838  2a        rol
f839  2a        rol
f83a  29 01     and  #$01
f83c  05 5a     ora  $5a
f83e  85 5a     sta  $5a
f840  b1 30     lda  ($30),y
f842  29 7c     and  #$7c
f844  4a        lsr
f845  4a        lsr
f846  85 5b     sta  $5b
f848  b1 30     lda  ($30),y
f84a  29 03     and  #$03
f84c  0a        asl
f84d  0a        asl
f84e  0a        asl
f84f  85 5c     sta  $5c
f851  c8        iny
f852  d0 06     bne  $f85a
f854  a5 4e     lda  $4e
f856  85 31     sta  $31
f858  a4 4f     ldy  $4f
f85a  b1 30     lda  ($30),y
f85c  29 e0     and  #$e0
f85e  2a        rol
f85f  2a        rol
f860  2a        rol
f861  2a        rol
f862  05 5c     ora  $5c
f864  85 5c     sta  $5c
f866  b1 30     lda  ($30),y
f868  29 1f     and  #$1f
f86a  85 5d     sta  $5d
f86c  c8        iny
f86d  84 34     sty  $34
;     Bytes aus
;     Zwischenspeicher
f86f  a6 56     ldx  $56
;     holen, gemäß Tabelle
;     übersetzen und ab $52
;     bis $55
f871  bd a0 f8  lda  $f8a0,x
f874  a6 57     ldx  $57
f876  1d c0 f8  ora  $f8c0,x
;     speichern für Übergabe
;     1. Byte
f879  85 52     sta  $52
f87b  a6 58     ldx  $58
f87d  bd a0 f8  lda  $f8a0,x
f880  a6 59     ldx  $59
f882  1d c0 f8  ora  $f8c0,x
;     2. Byte
f885  85 53     sta  $53
f887  a6 5a     ldx  $5a
f889  bd a0 f8  lda  $f8a0,x
f88c  a6 5b     ldx  $5b
f88e  1d c0 f8  ora  $f8c0,x
;     3. Byte
f891  85 54     sta  $54
f893  a6 5c     ldx  $5c
f895  bd a0 f8  lda  $f8a0,x
f898  a6 5d     ldx  $5d
f89a  1d c0 f8  ora  $f8c0,x
;     4. Byte
f89d  85 55     sta  $55
f89f  60        rts
;     Tabelle dient der
;     Decodierung der GCR-
;     Bytes in die entspr.
;     Hex-Werte.
;     Die $FF-Bytes geben
;     die Positionen der
;     illegalen Bitkombina-
;     tionen an. Sie veran-
;     lassen ggf. einen
;     »24, READ ERROR«.
f8a0  ff ff ff ff ff ff ff ff
f8a8  ff 80 00 10 ff c0 40 50
f8b0  ff ff 20 30 ff f0 60 70
f8b8  ff 90 a0 b0 ff d0 e0 ff
f8c0  ff ff ff ff ff ff ff ff
f8c8  ff 08 00 01 ff 0c 04 05
f8d0  ff ff 02 03 ff 0f 06 07
f8d8  ff 09 0a 0b ff 0d 0e ff
```

Pufferinhalt decodieren

```
;     Pufferzeiger auf Null
f8e0  a9 00     lda  #$00
;     setzen
f8e2  85 34     sta  $34
f8e4  85 2e     sta  $2e
;     2. Pufferzeiger
f8e6  85 36     sta  $36
;     Adresse für
;     Ausweichpuffer
f8e8  a9 01     lda  #$01
f8ea  85 4e     sta  $4e
;     nach $4e/4f
f8ec  a9 ba     lda  #$ba
f8ee  85 4f     sta  $4f
;     Pufferadresse Hi retten
f8f0  a5 31     lda  $31
f8f2  85 2f     sta  $2f
;     Bytes decodieren
f8f4  20 e6 f7  jsr  $f7e6
;     1. Byte als Kennzeichen
f8f7  a5 52     lda  $52
;     für Datenblock
f8f9  85 38     sta  $38
f8fb  a4 36     ldy  $36
;     restliche Bytes in
;     Puffer
f8fd  a5 53     lda  $53
f8ff  91 2e     sta  ($2e),y
f901  c8        iny
f902  a5 54     lda  $54
f904  91 2e     sta  ($2e),y
f906  c8        iny
f907  a5 55     lda  $55
f909  91 2e     sta  ($2e),y
f90b  c8        iny
f90c  84 36     sty  $36
;     nächste Kolonne
;     codieren
f90e  20 e6 f7  jsr  $f7e6
f911  a4 36     ldy  $36
f913  a5 52     lda  $52
;     Bytes in Puffer
;     schreiben
f915  91 2e     sta  ($2e),y
f917  c8        iny
f918  f0 11     beq  $f92b
f91a  a5 53     lda  $53
f91c  91 2e     sta  ($2e),y
f91e  c8        iny
f91f  a5 54     lda  $54
f921  91 2e     sta  ($2e),y
f923  c8        iny
f924  a5 55     lda  $55
f926  91 2e     sta  ($2e),y
f928  c8        iny
;     restliche Bytes ggf.
;     codieren
f929  d0 e1     bne  $f90c
;     Prüfsumme Datenblock
;     übernehmen
f92b  a5 53     lda  $53
f92d  85 3a     sta  $3a
;     Pufferadresse Hi
;     zurückholen
f92f  a5 2f     lda  $2f
f931  85 31     sta  $31
f933  60        rts
```

Listing 3. Die DOS-Routinen zur GCR-Codierung (Schluß)

Hier gibt's Clubs

Luxemburg

Commodore Club Luxemburg (C.C.L.), 1 Rue des Muguets, L 5970 Itzig/Luxemburg, Clubmagazin, Extrainfos, Musikteil, Kummertelefon;

Holland

Copy Tronics, Postbus 700, 7400 AS Deventer Nederland, Clubzeitschrift, Clubtreffen;

Österreich

CHIPSI-Computer-Club, Jasomirgottstr. 3, Mezzanin, 1010 Wien, Programm- und Literaturtausch, Programmierkurse, begünstigter Bezug von Computern, praktische Programmiertips, Clubzeitung, Clubtreffen;

Wiener Hard- u. Software Computerclub, René Fürst, Herbortgasse 22, 1110 Wien, Softwareerstellung und Tausch, Hardwareunterstützung, Beitrag 150 Schilling pro Quartal, Beitrittsgebühr 1x150 Schilling, Tel. (0222) 74391 54, (0222) 61 26 83

Bund Technischer Amateure, Stromstr. 36-38/29/1, 1200 Wien, Clubtreffen, Grundlagen, Tips und Tricks, PRG-Bibliothek, Softwaretausch;

COMUSCLUS, Computer User Club Salzburg, A-5033 Salzburg, Postfach 1 33, Clubtreffen, Clubstammtisch, monatliche Zeitschrift, Buch- und Softwarebibliothek, Arbeitskreise, Programmierkurse, auch für Nichtmitglieder, Sammelbestellungen;

Ruf-Commodore-User-Club/Salzburg, A-5020 Salzburg, Haunspergstraße 21, Tel. (0662) 75770, Herr Wiedermann

Ruf-Commodore-User-Club/Graz, Grazbachgasse 60, Tel. (03 16) 70 02 66, Herr Hiess

Ruf-Commodore-User-Club/Klagenfurth, A-9020 Klagenfurt, Sponheimer Straße 9-11, Tel. 04222) 55105, Herr Krassnitzer

Clubtreffen (wöchentlich), Softwarebibliothek, Hardwareangebot, Programmierkurse etc.

IKC, Internationaler Kommunikationsclub, Postfach 18, CH-6312 Steinhausen

Info gegen 3 DM, Clubzeitschrift, Beratertelefon, DFÜ, Sektionen, Kurse, Listingkorrekturservice, Meisterschaften, etc. Aktiv-, Passiv- und Ehremitglieder sowie Gönner.

Schweiz

COBEC Computer-Benutzer-Club Wohlen, Postfach 10, CH-5613 Hilfikon AG, Schweiz, Clubtreffen, Softwarebibliothek, diverse Hardware, Kurse;

C 64-Anwenderclub Zürich (ACZ), Postfach 194, 8029 Zürich, Clubtreffen, Softwarebibliothek, Hardware, Modem, Assembler- und Basic-Kurse, Jahresbeitrag 50 Franken;

C 64 Club, J. Reich, Am Glattbogen 170, 8050 Zürich
für Anfänger, Clubtreffen, Zeitschrift (geplant), Kurse, Seminare, Softwaretausch, vergünstigter Bezug von Soft- und Hardware, Beitrag 30 Franken

Syntax Error C 64, Postfach 253, CH-9004 St. Gallen, Clubtreffen, Mailbox, DFÜ, Softwarebibliothek, Hard- und Softwareentwicklung.