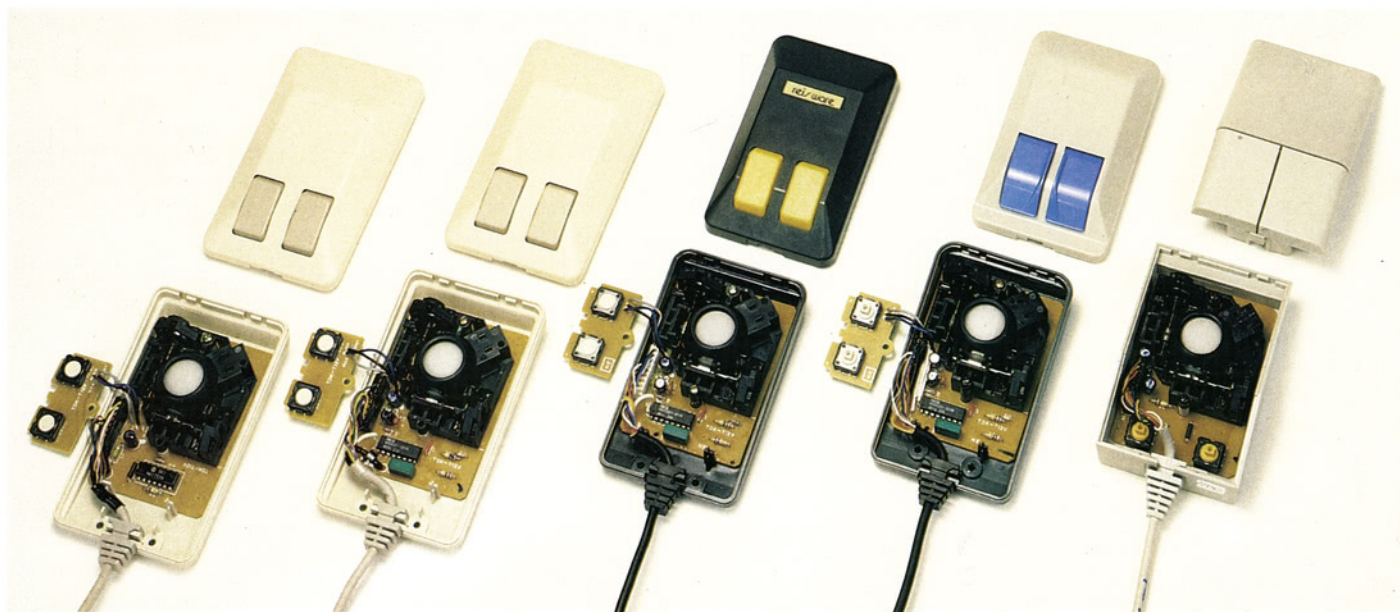




Die Mäuse breiten sich aus

Mäuse werden auch für den C 64 und C 128 immer interessanter. Worin unterscheiden sich diese Mäuse eigentlich und wie wird eine Maus überhaupt vom Programm abgefragt? Hier finden Sie Antworten darauf.

Für die großen Kollegen des C 64, wie den Amiga oder Atari, ist eine Maus unerlässlich. Die Benutzeroberfläche dieser Computer ist absolut auf die Bedienung mit diesem Eingabegerät abgestimmt. Aber auch für den C 64 werden Mäuse immer interessanter, nicht zuletzt durch das neue Betriebssystem Geos, das wir Ihnen in der Ausgabe 6/86 vorgestellt haben. Es lädt geradezu ein, eine Maus zu benutzen.

Commodore hat auch schon geraume Zeit angekündigt, eine Maus für den C 64 und C 128 auf den Markt zu bringen. Jetzt ist es endlich soweit. Laut Auskunft von Commodore Deutschland kann die Commodore-Maus seit Mitte August geliefert werden. Natürlich haben wir auch die Commodore-Maus (Bild 1) für Sie unter die Lupe genommen.

Die neue Commodore-Maus

Von außen sieht sie aus wie die Amiga-Maus. Da aber weder der Amiga-Typ am C 64 funktioniert, noch das C

64-Modell am Amiga, muß das Innenleben schon unterschiedlich sein. Also schraubten wir die neue Maus auseinander. Was dabei zum Vorschein kam, bringt sicher auch Sie zum Staunen (Bild 2). Eine verblüffende Ähnlichkeit mit der schon in der Ausgabe 3/86 vorgestellten NCE-Maus war nicht von der Hand zu weisen. Das macht natürlich neugierig auf das Innenleben anderer Mäuse.

Nach dem Mäusetest in der März-Ausgabe kamen für den C 64/C 128 noch weitere Mäuse von Reissware und Schneider auf dem Markt. Die Firma Schneider hat übrigens nichts mit dem gleichnamigen Computer-Hersteller (CPC) zu tun. Nach dem Aufschrauben dieser Mäuse zeigte sich das gleiche Bild wie bei der NCE- und Commodore-Maus.

Um die Sache auf die Spitze zu treiben, zeigen wir Ihnen auch das Innenleben der Amiga- und Atari-Maus. Im Bild oben können Sie sich selbst einen Eindruck über die mehr oder weniger engen Verwandtschaftsverhält-

nisse unter all diesen Mäusen verschaffen. Von links nach rechts sehen Sie die Amiga-, Commodore-, Reissware-, NCE- und Atari-Maus. Auf die Ablichtung der Schneider-Maus haben wir verzichtet, da sie von außen und von innen absolut mit der Reissware-Maus identisch ist. Lediglich die goldfarbene Firmenbezeichnung fehlt.

Mäuse-Verwandtschaft

Die Ähnlichkeit der Mäuse reizt natürlich dazu, mehr über die Gründe dafür zu erfahren. Bei den Recherchen zeigte sich, daß die Mechaniken, die übrigens sehr gut sind, für all diese Mäuse von einem japanischen Großunternehmen hergestellt werden. Dieses Unternehmen verkauft seine Produkte nicht unter dem eigenen Namen, sondern nur an weitere Großabnehmer. Auch die Produktion der Anpassungsplatinen scheint in einer Hand zu liegen. Zumindest tragen die Platinen aller untersuchten Mäuse den Firmennamen »Mitsumi«. Bis auf

die schon in der Ausgabe 3/86 vorgestellte Rushware-Maus kommt demnach das Innenleben für alle Mäuse aus dem gleichen Haus. Nur die Schaltung ist auf den jeweiligen Computer angepaßt.

Was bedeutet das für unseren C 64? Als der C 64 entwickelt wurde, war an eine Maus für diesen Computer noch nicht zu denken. Also ging man davon aus, daß an dem Computer ein Joystick angeschlossen wird. Bis auf vereinzelte Ausnahmen sind ja auch heute noch alle Programme auf eine Bedienung mit dem Joystick abgestimmt. Diese Tatsache hat man bei der Entwicklung einer Maus für den C 64 berücksichtigt. Schauen Sie sich einmal die Belegung der Control-Ports vom C 64 und C 128 in der Tabelle 1 an. Das Bild 3 zeigt den Joystick-Port, wie er von außen am Computer zu sehen ist.

Da von fast keinem C 64-Programm Maus-Impulse, die eine bestimmte Wegstrecke singalisieren, verarbeitet werden, haben die Techniker sich bemüht, mit der Maus einen Joystick zu si-

mulieren. Praktisch bedeutet dies: Bei einer Mausebewegung aufwärts geht beispielsweise der Pin 1 am Mausstecker auf logisch »0«, ohne Impulse zu erzeugen. Lediglich bei extrem langsamer Mausebewegung entstehen noch Impulse. Gleiches geschieht bei Bewegungen in die anderen drei Richtungen. Wie die Kontakte bei den Anschluß-Steckern der C 64-Mäuse belegt sind, zeigt die Tabelle 2. Die Angaben gelten für die neue Commodore-Maus ebenso wie für die Mäuse von Reissware, Schneider und NCE.

Der Nachteil der Joystick-Simulation ist folgender: Auf dem Bildschirm werden nur horizontale, vertikale und diagonale Bewegungen (45°) wiedergegeben, wie man es vom Joystick her gewohnt ist. Die Bewegungsgeschwindigkeit auf dem Bildschirm entspricht auch nicht unbedingt der Mausebewegung. Von einer Maus, die einen Joystick simuliert, sind also auf dem Bildschirm nicht so präzise Bewegungsnachführungen zu erwarten, wie man es vom Amiga oder Atari ST her kennt. Selbst beim Betriebssystem »Geos« ist es leider nicht anders.

Eine Ausnahme bildet nur die NCE-Maus im Zusammenhang mit dem eigenen Grafik-Programm (siehe auch Ausgabe 3/86, Seite 29). Aber darauf kommen wir später noch einmal zurück. An dieser Stelle wollen wir zunächst einmal festhalten, daß sich die C 64-Mäuse von Commodore, Reissware, Schneider und mit Einschränkungen auch die NCE-Maus im inneren Aufbau und im Verhalten gegenüber vorhandener Software absolut gleichen.

Aber warum reagieren die Amiga- und Atari-Maus an ihren Computern so präzise? Nun, die Ursache ist eine andere Anpassungsschaltung und eine darauf abgestimmte Software. Die Mäuse erzeugen in horizontaler und vertikaler Richtung jeweils zwei Rechtecksignale, die gegeneinander phasenverschoben sind. Durch die Richtung der Phasenverschiebung kann der Computer ermitteln, ob die Bewegungsrichtung positiv oder negativ ist. Als positiv kön-



Bild 1.
Die neue Commodore-Maus für den C 64

nen beispielsweise die Bewegungsrichtungen aufwärts und rechts definiert werden, negativ wären dann die Bewegungen abwärts und links. Allerdings bleibt diese Definition dem Programmierer überlassen.

Wenn Sie für Ihre Software eigene Mausroutinen entwickeln wollen, dann finden Sie als wichtige Grundlage die Impulse und Schaltfunktionen dieser beiden Mäuse in den Tabellen 3 (Amiga-Maus) und 4 (Atari-Maus).

Aber kommen wir zurück zu den Mäusen, die einen

Joystick simulieren. Wie läßt sich die Bewegung dieser Mäuse über den Control-Port abfragen?

Maus-Programme

Dazu einige Grundlagen. Verwaltet werden die Control-Ports vom ersten der beiden Complex Interface Adapter (CIA). Er belegt die Register \$DC00 bis \$DCFF (dezimal 56320 bis 56575). Die vier Bewegungsrichtungen für den Control-Port 2 werden in den ersten vier

Bits (Bit 0 bis Bit 3) des Registers \$DC00 (dezimal 56320) registriert. Wenn Sie eine der Mäuse, die mit Joystick-Simulation arbeiten, in den Port einstecken und die Maus nicht bewegen, dann enthalten alle vier Bits logisch »1«. Bewegen Sie die Maus, dann wird das entsprechende Bit von der Maus auf 0 gesetzt. Dabei steht das Bit 0 für aufwärts, Bit 1 für abwärts, Bit 2 für links und Bit 3 für rechts. Das vierte Bit in diesem Register geht übrigens auf 0, wenn Sie die linke Maustaste drücken. Das nachfolgende Register \$DC01 (dezimal 56321) verwaltet in gleicher Weise den Control-Port 1.

In Basic können Sie ein einzelnes Bit mit Hilfe der logischen Verknüpfung AND überprüfen. Sie lesen das entsprechende Register mit PEEK aus und verknüpfen es mit dem Wert des gewünschten Bits (1 = Bit 0, 2 = Bit 1, 4 = Bit 2, 8 = Bit 3, 16 = Bit 4 etc.). Ist das Ergebnis gleich dem Wert, dann enthält das entsprechende Bit eine »1«. Da dieses Verfahren absolut identisch mit der Joystick-Abfrage ist, die in dieser Ausgabe im Artikel »Joysticks und Trackballs — mehr als nur ein Spielzeug« ausführlich beschrieben wird, gehen wir hier nicht weiter auf die Basic-Programmierung ein. Eine Cursor-Bewegung auf dem Bildschirm, die einigermaßen der Mausebewegung entsprechen soll, läßt sich ohnehin nicht in Basic durchführen. Für solche zeitkritischen Vorgänge muß schon in Maschinensprache programmiert werden.

Eine sinnvolle Methode ist die interruptgesteuerte Abfrage. Eine solche Routine wollen wir Ihnen vorstellen (Listing 1). Damit Sie das Maschinenprogramm besser verstehen können, haben wir auch den dokumentierten Source-Code (Listing 2) abgedruckt. Alle notwendigen Erklärungen sind im Listing 2 enthalten.

Mit Hilfe des kurzen Testprogramms (Listing 3) können Sie einen Cursor mit der Maus über den Bildschirm bewegen. Bevor Sie allerdings dieses Programm mit RUN starten, müssen Sie noch das Maschinenpro-

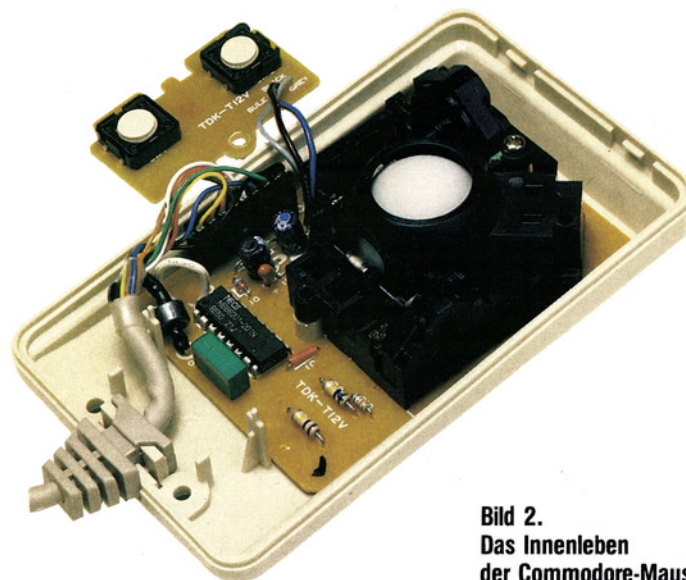


Bild 2.
Das Innenleben der Commodore-Maus

gramm initialisieren und dabei die Parameter für den Bildschirmausschnitt und die Bewegungsgeschwindigkeit übergeben. Geben Sie dazu bitte folgendes im Direktmodus ein:

SYS 49152,39,24,2

Die beiden Zahlen hinter 49152 legen die rechte und obere Bewegungsgrenze für den Cursor fest. In unserem Fall wird der ganze Bildschirm in Anspruch genom-

men. Die letzte Zahl bestimmt die Bewegungsgeschwindigkeit. Bei größeren Werten wird die Bewegung auf dem Bildschirm langsamer.

Nachdem Sie die Parameter durch Drücken der Taste <RETURN> übergeben haben, können Sie das Basic-Programm starten, die Maus am Control-Port 2 anschließen und den Cursor mit der Maus über den Bildschirm

bewegen. Für eine Maus mit Joystick-Simulation funktioniert dies erstaunlich gut. Experimentieren Sie doch mal mit den Parametern für Bildausschnitt und Bewegungsgeschwindigkeit.

Letztlich bleiben aber bei allen Mäusen mit Joystick-Simulation noch Wünsche nach präziser und schneller Bewegungsnachführung offen. Besonders macht sich dies beim Umgang mit Gra-

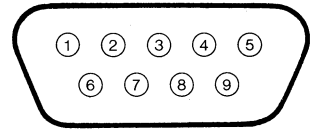


Bild 3. Der Control-Port am C 64 und C 128 von außen gesehen

```
Name : maus-irq c64      c000 c100
c000 : 20 fd ae 20 8a ad 20 9b 9c
c008 : bc a5 65 a6 64 8d 42 c0 02
c010 : 8e 43 c0 20 f1 b7 8e 44 14
c018 : c0 20 f1 b7 8e 45 c0 8e 8f
c020 : 46 c0 a2 04 a9 00 95 fa d6
c028 : ca 10 fb a9 47 a0 c0 78 9c
c030 : 8d 14 03 8c 15 03 58 a9 38
c038 : b9 a0 c0 8d 11 03 8c 12 a3
c040 : 03 60 00 00 00 00 00 ce 11
c048 : 46 c0 d0 6a ad 45 c0 8d 93
c050 : 46 c0 ad 00 dc 4a b0 0b 5b
c058 : e6 fc ae 44 c0 e4 fc b0 79
c060 : 02 86 fc 4a b0 06 a6 fc fe
c068 : f0 02 c6 fc 4a b0 0c a6 52
c070 : fa d0 06 a6 fb f0 04 c6 0e
c078 : fb c6 fa 4a b0 1a e6 fa 4c
c080 : d0 02 e6 fb ac 43 c0 ae d0
c088 : 42 c0 c4 fb 90 06 d0 08 67
c090 : e4 fa b0 04 86 fa 84 fb e8
c098 : 4a b0 04 a9 01 85 fd a9 f8
c0a0 : 80 8d 00 dc a2 00 e8 d0 f2
c0a8 : fd ad 19 d4 10 03 e8 86 27
c0b0 : fe a9 7f 8d 00 dc 4c 31 8f
c0b8 : ea 20 9b bc a5 64 d0 0d 0c
c0c0 : a6 65 f0 0c ca f0 17 ca fd
c0c8 : f0 22 ca f0 26 4c 48 b2 e5
c0d0 : a5 fa 85 63 a5 fb 85 62 d5
c0d8 : a2 90 38 4c 49 bc a9 00 7b
c0e0 : 85 62 a5 fc 85 63 a2 90 be
c0e8 : 38 4c 44 bc a5 fd 46 fd 4e
c0f0 : 4c 3c bc a5 fe 46 fe 4c f5
c0f8 : 3c bc 00 00 00 00 00 93
```

Listing 1. Eine interruptgesteuerte Mausabfrage

Pin	Signal	Bemerkung
1	Joy 0	logisch 0 = aufwärts
2	Joy 1	logisch 0 = abwärts
3	Joy 2	logisch 0 = links
4	Joy 3	logisch 0 = rechts
5	Pot Y	Paddle
6	Button	logisch 0 = Feuerknopf
7	+5V	Betriebsspannung
8	GND	Masse
9	Pot X	Paddle

Tabelle 1. Der Control-Port am C 64 und C 128

Pin	Signal	Bemerkung
1	aufwärts	logisch 0 = aufwärts
2	abwärts	logisch 0 = abwärts
3	links	logisch 0 = links
4	rechts	logisch 0 = rechts
5	NC	nicht angeschlossen
6	linke Taste	logisch 0 = Taste gedrückt
7	+5V	Betriebsspannung
8	GND	Masse
9	rechte Taste	logisch 0 = Taste gedrückt

Tabelle 2. Kontaktbelegung bei den Mäusen von Commodore, Reisware, Schneider und NCE

PIN	aufwärts	abwärts	links	rechts
1	Impulse	Impulse		
2			Impulse	Impulse
3	Impulse	Impulse		
4			Impulse	Impulse
5	nicht angeschlossen			
6	linke Taste; logisch 0 = Taste gedrückt			
7	+5 V			
8	GND (Masse)			
9	rechte Taste; logisch 0 = Taste gedrückt			

Tabelle 3. Kontaktbelegung bei der Amiga-Maus

PIN	aufwärts	abwärts	links	rechts
1			Impulse	Impulse
2			Impulse	Impulse
3	Impulse	Impulse		
4	Impulse	Impulse		
5	nicht angeschlossen			
6	linke Taste; logisch 0 = Taste gedrückt			
7	+5 V			
8	GND (Masse)			
9	rechte Taste; logisch 0 = Taste gedrückt			

Tabelle 4. Kontaktbelegung bei der Atari-Maus

fikprogrammen und Geos bemerkbar.

Erstaunlich ist allerdings die Genauigkeit der NCE-Maus im Zusammenhang mit dem mitgelieferten Grafikprogramm. Der Grund dafür ist ein eigener Maus-Modus, der trotz aller Ähnlichkeit gegenüber den Mäusen von Commodore, Reissware und Schneider besteht. Ob eine solche Präzision auch mit anderen Mäusen am C 64 und

C 128 möglich ist, dazu erfahren Sie in einer der nächsten Ausgaben mehr.

(Ingolf Kübler/kn)

Info: Commodore Deutschland, Lyoner Str. 38, 6000 Frankfurt/Main 71
Atari Corp. (Deutschland), Frankfurter Str. 89-91, 6096 Raunheim
Nordphon Computer Electronic (NCE), Fliederbogen 1, 2399 Tarp
Reis GmbH (Reissware), Bergstr. 80, 5584 Bullay
Schneider Elektronik, Siegenstr. 124, 4600 Dortmund 15

Beschreibung:

Diese Routine testet bei jedem k-ten Interrupt ob die Maus (Port 2) bewegt wird und aktualisiert die Koordinaten.

Die Koordinaten werden mit Hilfe der usr-Funktion ausgelesen:

```
usr(0) 0 < x-Koordinate (= wählbare rechte Grenze) ($fa/$fb)
usr(1) 0 < y-Koordinate (= wählbare obere Grenze) ($fc)
usr(2) 1 = linke Taste gedrückt, 0 sonst ($fd)
usr(3) 1 = rechte Taste gedrückt, 0 sonst ($fe)

Man kann die Koordinate vorbesetzen, indem man die gewünschten Werte nach der Initialisierung in die Register $fa-$fc poke't.
```

Initialisiert wird die Routine mit

```
sys 49152, <rechte Grenze>, <obere Grenze>, <k>
```

```
wobei <rechte Grenze> = [1,...,3191]
      <obere Grenze> = [1,...,1991]
      <k> = Decrement für Maus-Interrupt [1,...,15]

Mit diesem Parameter läßt sich die Genauigkeit der Maus beeinflussen.
```

Belegter Speicherplatz:

```
$00fa-$00fe : Koordinaten und Flags
$c000-$c041 : Initialisierung
$c042-$c046 : Variablen
$c047-$c0b8 : IRQ-Routine
$c0b9-$c0f9 : usr-Funktion
```

Programm:

Initialisierung

```
0c000 20 fd ae jsr $ae fd Test auf Komma
0c003 20 8a ad jsr $ad 8a rechte Grenze lesen
0c006 20 9b bc jsr $bc 9b
0c009 a5 65 ldx $65 und merken
0c00b a6 64 ldx $64
0c00d 8d 42 c0 sta $c042
0c010 8e 43 c0 stx $c043
0c013 20 f1 b7 jsr $b7 f1 Test auf Komma und obere Grenze lesen
0c016 8e 44 c0 stx $c044 und merken
0c019 20 f1 b7 jsr $b7 f1 Test auf Komma und IRQ-Decrement lesen
0c01c 8e 45 c0 stx $c045 und merken
0c01f 8e 46 c0 stx $c046
0c022 a2 04 ldx $04 Koordinaten löschen
0c024 a9 00 ldx $00
0c026 95 fa sta $fa,x
0c028 ca dex
0c029 10 fb bpl $c026
0c029 10 fb bpl $c026
0c02b a9 47 ldx $47 IRQ-Zeiger setzen
0c02d a0 c0 ldy $c0
0c02f 78 sei
0c030 8d 14 03 sta $0314
0c033 8c 15 03 sty $0315
0c036 58 cli
0c037 a9 b9 ldx $b9 usr-Funktion-Zeiger setzen
0c039 a0 c0 ldy $c0
0c03b 8d 11 03 sta $0311
0c03e 8c 12 03 sty $0312
0c041 60 rts
```

Variablen

```
0c042 rechte Grenze low-Byte
0c043 rechte Grenze high-Byte
0c044 obere Grenze
0c045 IRQ-Decrement
0c046 IRQ-Zähler
```

IRQ

```
0c047 ce 46 c0 dec $c046 IRQ-Zähler erniedrigen
0c04a d0 6a bne $c046 wenn größer als null dann normalen IRQ
0c04c ad 45 c0 ldx $c045 IRQ-Zähler setzen
0c04f 8d 46 c0 sta $c046
0c052 ad 00 dc ldx $dc00 CIA Port A lesen
0c055 4a lsr Maus nach oben?
0c056 b0 0b bcs $c053 wenn nein dann Sprung
0c058 e6 fc inc $fc y-Koordinate erhöhen
0c05a ae 44 c0 ldx $c044 mit oberer Grenze vergleichen
```

```
0c05d e4 fc cpx $fc
0c05f b0 02 bcs $c063
0c061 86 fc stx $fc wenn größer dann y-Koordinate = obere Grenze
0c063 4a lsr Maus nach unten
0c064 b0 06 bcs $c06c wenn nein dann Sprung
0c066 a6 fc ldx $fc y-Koordinate bereits null?
0c068 f0 02 beq $c06c
0c06a c6 fc dec $fc wenn nein dann erniedrigen
0c06c 4a lsr Maus nach links?
0c06d b0 0c bcs $c07b wenn nein dann Sprung
0c06f a6 fa ldx $fa wenn x-Koordinate größer als null
0c071 d0 06 bne $c079 dann decrementieren
0c073 a6 fb ldx $fb
0c075 f0 04 beq $c07b
0c077 c6 fb dec $fb
0c079 c6 fa dec $fa
0c07b 4a lsr
0c07c b0 1a bcs $c098 Maus nach rechts?
0c07e e6 fa inc $fa wenn nein dann Sprung
0c080 d0 02 bne $c084 incrementieren
0c082 e6 fb inc $fb
0c084 ac 43 c0 ldy $c043 wenn x-Koordinate größer als rechte Grenze
0c087 ae 42 c0 ldx $c042 dann auf rechte Grenze setzen
0c08a c4 fb cpy $fb
0c08c 90 06 bcc $c094
0c08e d0 08 bne $c098
0c090 e4 fa cpx $fa
0c092 b0 04 bcs $c098
0c094 86 fa stx $fa
0c096 84 fb sty $fb
0c098 4a lsr linke Maus-Taste gedrückt?
0c099 b0 04 bcs $c09f wenn nein dann Sprung
0c09b a9 01 ldx $01 Flag setzen
0c09d 85 fd sta $fd
0c09f a9 80 ldx $80 rechte Maus-Taste gedrückt?
0c0a1 8d 00 dc sta $dc00
0c0a4 a2 00 ldx $00
0c0a6 e8 inx
0c0a7 d0 fd bne $c0ab
0c0a9 ad 19 d4 ldx $d419
0c0ac 10 03 bpl $c0b1
0c0ae e8 inx wenn ja dann Flag setzen
0c0af 86 fe stx $fe
0c0b1 a9 7f ldx $7f
0c0b3 8d 00 dc sta $dc00
0c0b6 4c 31 ea jmp $ea31 reguläre IRQ-Routine fortsetzen
```

usr-Funktion

```
0c0b9 20 9b bc jsr $bc 9b Index holen
0c0bc a5 64 ldx $64 wenn Index > 255 dann Fehler
0c0be d0 0d bne $c0cd
0c0c0 a6 65 ldx $65
0c0c2 f0 0c beq $c0d0 Index = 0?
0c0c4 ca dex
0c0c5 f0 17 beq $c0de Index = 1?
0c0c7 ca dex
0c0c8 f0 22 beq $c0ec Index = 2?
0c0ca ca dex
0c0cb f0 26 beq $c0f3 Index = 3?
0c0cd 4c 48 b2 jmp $b248 sonst 'illegal quality error' ausgeben
```

usr(0)

```
0c0d0 a5 fa ldx $fa x-Koordinate lesen
0c0d2 85 63 sta $63
0c0d4 a5 fb ldx $fb
0c0d6 85 62 sta $62
0c0d8 a2 90 ldx $90
0c0da 38 sec
0c0db 4c 49 bc jmp $bc49 ins FAC bringen und usr beenden
```

usr(1)

```
0c0de a9 00 ldx $00 y-Koordinate lesen
0c0e0 85 62 sta $62
0c0e2 a5 fc ldx $fc
0c0e4 85 63 sta $63
0c0e6 a2 90 ldx $90
0c0e8 38 sec
0c0e9 4c 44 bc jmp $bc44 ins FAC bringen und usr beenden
```

usr(2)

```
0c0ec a5 fd ldx $fd Flag für linke Taste lesen
0c0ee 46 fd lsr $fd Flag löschen
0c0f0 4c 3c bc jmp $bc3c ins FAC bringen und usr beenden
```

usr(3)

```
0c0f3 a5 fe ldx $fe Flag lesen
0c0f5 46 fe lsr $fe Flag löschen
0c0f7 4c 3c bc jmp $bc3c ins FAC bringen und usr beenden
```

Listing 2. Source-Code (Schluß)

```
10 U=1024 <179>
20 POKE U,32 <208>
30 U=1984+USR(0)-USR(1)*40 <133>
40 POKE U,160 <189>
50 GOTO 20 <236>
```

© 64'er