COL256 Graphikprogramm

Die Beschreibung

COL 256 Farbgrafikprogramm (C) 1986 Kei Thomsen.

Inhaltsverzeichnis:

Befehlsverzeichnis:

Befehl:	Index:	Bedeutung:
ACLACDOT	33	Absolutes berechnen einer Punktadresse
ADRAW	29	Absolute Linie zur X2,Y2 Position
AMOVE	27	Absolutes merken der X1,Y1 Position
ANDDOT	9	AND-Verknüpfung mit einem Punkt
APOINT	31	Absolutes Setzen eines Punktes
CALCDOT	11	Berechnen einer Punktadresse
CALCWDOT	64	Berechnen einer Punktadresse im Window
CALCZEILE	12	Berechnen der Zeilenanfangsadresse
CHANGE	48	Vertauschen von Bildinhalten
CI	42	Zeichen von Tastatur einlesen
CIRCLE	38	Zeichnen eines Kreises
CLIPPEINFACH	26	Begrenzen der X,Y Coordinaten
CLPG	4	Löschen einer Bildseite
CLPGCOL	5	Löschen einer Bildseite mit Farbe
CLR	2	Löschen aller Bildseiten
CLRCOL	3	Löschen aller Bildseiten mit Farbe
CLRFIGUR	66	Löschen einer Figur
CLRSIGN	45	Löschen eines Zeichen
CLRWINDOW	57	Löschen eines Window
CO	44	Zeichenausgabe
COLINIT	0	Grafikkarte Initiaisieren und löschen
COPY	50	Copieren von Bildteilen
COXY	40	Zeichendarstellung auf X,Y Position
CRLF	82	Carriage Return und Line Feed ausgeben
CSTS	43	Tastatur abfragen
DEFWINDOW	59	Definieren eines Windowspeicher
DELAY	25	Verzögerung um 100ms * Faktor
DELTAXY	51	Coordinaten X1,Y1,X2,Y2 umrechnen
DISK	39	Ausgefüllten Kreis zeichnen
DRAWTO	15	Linie nach X2,Y2 zeichnen
FIGUR	67	Figur zeichen
FLAECHE	37	Ausgefülltes Rechteck
GETBACKGROUND	78	Hintergrundfarbe auslesen
GETBLAU	84	Blauwert lesen
GETCOL	10	Farbe eines Punkt auslesen
GETCOLOR	74	Aktuelle Punktefarbe lesen
GETCURXY	81	Cursorposition auslesen
GETGRAU	83	Grauwert lesen
GETGRUEN	85	Grünwert lesen
GETROT	86	Rotwert lesen
GETSETMODE	76	Aktuellen Setmode lesen
GETWDOT	63	Punkt aus einem Window lesen
GETWINDOW	58	Window aus Bildschirm copieren
GETXYXY	72	Lesen der X1,Y1,X2,Y2 Position
GETZCOLOR	80	Aktuelle Zeichenfarbe lesen
INIT	1	Grafikkarte Initiaisieren
INVERT	49	Invertieren eines Bildausschnittes
LINE	13	Linie zeichnen von X1,Y1 nach X2,Y2
LOAD	21	Laden einer Bildseite von Floppy

Befehlsverzeichnis:

Befehl:	Index:	Bedeutung:
MAXDELTA	53	Begrenzen der X,Y,dX,dY Coordinaten
MAXOVER	52	Begrenzen der X,Y Coordinaten
MOVETO	14	Merken der X1,Y1 Position
NEWPAGE	19	Anwahl der Schreib und Leseseite
PAINT	22	Ausfüllen von Flächen
PAINTCOL	23	Ausfüllen von Flächen bis Farbgrenze
POINT	35	Punkt setzen mit Modeauswahl
POLYGON	55	Linien zeichnen geschlossen
POLYLINE	54	Linien zeichnen offen
PUTFIGUR	68	Figur darstellen
RCALCDOT	34	Relatives berechnen einer Punktadresse
RDRAW	30	Relative Linie zur X2,Y2 Position
READ	46	Text einlesen
RECHTECK	36	Rechteck zeichnen
RMOVE	28	Relatives merken der X1,Y1 Position
RPOINT	32	Relatives setzen eines Punktes
SAVE	20	Speichern einer Bildseite auf Floppy
SCROLLTO	18	Softscroll auf eine Position
SCROLLX	16	Softscroll in X Richtung
SCROLLY	17	Softscroll in Y Richtung
SEARCH	87	Suchen und ersetzen einer Farbe
SETA4	70	Register A4 setzen
SETBACKGROUND	78	Hintergrundfarbe setzen
SETCOLOR	73	Aktuelle Punktfarbe setzen
SETCURXY	41	Cursor auf X,Y setzen
SETDOT	7	ODER-Verknüpfung mit einem Punkt
SETFDOT	6	Setzen eines Punkt
SETFIGUR	65	Figur darstellen
SETSETMODE	75	Aktuellen Setmode setzen
SETWDOT	62	Setzen eines Punktes im Window
SETWINDOW	56	Window darstellen
SETXYXY	71	Setzen der X1,Y1,X2,Y2 Position
SETZCOLOR	79	Aktuelle Zeichenfarbe setzen
SYNC	24	Syncronimpuls abfragen
WINDOW	60	Window darstellen
WINDOWOFF	61	Window löschen
WRITE	47	Text ausgeben
XORDDOT	8	EXOR-Verknüpfung mit einem Punkt
ZOOM	69	Ausschnittvergrösserung

Numerisches Befehlsverzeichnis:

Befehl:	Index:	Register: > Ausgabe
COLINIT	0	D0
INIT	1	D0
CLR	2	
CLRCOL	3	D0
CLPG	4	D1
CLPGCOL	5	D0 D1
SETFDOT	6	D0 D1 D2 >A0.L
SETDOT	7	D0 D1 D2 >A0.L
XORDOT	8	D0 D1 D2 >A0.L
ANDDOT	9	D0 D1 D2 >A0.L
GETCOL	10	D1 D2 >D0.B
CALCDOT	11	D1 D2 >A0.L
CALCZEILE	12	D2 >A0.L
LINE	13	X1 Y1 X2 Y2 COLOR
MOVETO	14	D1 D2
DRAWTO	15	D0 D1 D2
SCROLLX	16	D0
SCROLLY	17	D0
SCROLLTO	18	D1 D2
NEWPAGE	19	D0 D1
SAVE	20	D0 D1 A0
LOAD	21	D0 D1 A0
PAINT	22	D0 D1 D2
PAINTCOL	23	D0 D1 D2
SYNC	24	>D0.W
DELAY	25	D0
CLIPPEINFACH	26	D1 D2 >D1.W >D2.W
AMOVE	27	D1 D2
RMOVE	28	D1 D2
ADRAW	29	D0 D1 D2 D3
RDRAW	30	D0 D1 D2 D3
APOINT	31	D0 D1 D2 D3
RPOINT	32	D0 D1 D2 D3
ACALCDOT	33	D1 D2 >A0.L
RCALCDOT	34	D1 D2 >A0.L
POINT	35	D0 D1 D2 SETMODE (BACKGROUND) >A0.L
RECHTECK	36	D0 D1 D2 D3 D4
FLAECHE	37	D0 D1 D2 D3 D4
CIRCLE	38	D0 D1 D2 D3
DISK	39	D0 D1 D2 D3
COXY	40	D0 D1 D2 ZCOLOR
SETCURXY	41	D0 D1 D2 D3
CI	42	>D0.L
CSTS	43	>D0.B

Numerisches Befehlsverzeichnis:

Befehl:	Index:	Register: > Ausgabe
CO	44	D0
CLRSIGN	45	CURX CURY
READ	46	D0 D1 D2 D3 D4 A0 >D4.W >D5.B >A0.L
WRITE	47	D0 D1 D2 A0
CHANGE	48	D1 D2 D3 D4 D5 D6
INVERT	49	D0 D1 D2 D3 D4
COPY	50	D1 D2 D3 D4
DELTAXY	51	D1 D2 D3 D4 >D1.W >D2.W >D3.W >D4.W
MAXOVER	52	D1 D2 >CCR
MAXDELTA	53	D1 D2 D3 D4 >D1.W >D2.W >D3.W >D4.W
POLYLINE	54	D0 D3 A1
POLYGON	55	D0 D3 A1
SETWINDOW	56	D0 D1 D2 D3 D4 A1 A2
CLRWINDOW	57	D1 D2 D3 D4 A2
GETWINDOW	58	D1 D2 D3 D4 A1
DEFWINDOW	59	D0 D3 D4 A1 A2
WINDOW	60	D1 D2
WINDOWOFF	61	
SETWDOT	62	D0 D1 D2 >A0.L
GETWDOT	63	D1 D2 >D0.B >A0.L
CALCWDOT	64	D1 D2 >A0.L
SETFIGUR	65	D0 D1 D2 D3 A2
CLRFIGUR	66	
FIGUR	67	D0 D1 D2 D3 A0 A1
PUTFIGUR	68	D0 D1 D2 D3 A0 A1
ZOOM	69	D0 D1 D2 D3 D4
SETA4	70	>A4.L
SETXYXY	71	D1 D2 D3 D4
GETXYXY	72	>D1.W >D2.W >D3.W >D4.W
SETCOLOR	73	D0
GETCOLOR	74	>D0.B
SETSETMODE	75	D0
GETSETMODE	76	>D0.W
SETBACKGROUND	77	D0
GETBACKGROUND	78	>D0.B
SETZCOLOR	79	D0
GETZCOLOR	80	>D0.B
GETCURXY	81	>D0.B >D1.W >D2.W >D3.B
CRLF	82	
GETGRAU	83	D0 >D0.B
GETBLAU	84	D0 >D0.B
GETGRUEN	85	D0 >D0.B
GETROT	86	D0 >D0.B
SEARCH	87	D1 D2 D3 D4 D5 D6

Hardwareaufbau:

NDR - Klein - Computer

CPU 68008 / 68000 / 68020
COL 256
Farbmonitor oder ähnliches
min. 64Kbyte RAM frei
KEY
GDP64
FLO2 mit Floppy-Disk
Grundprogramm V4.2 / 4.3

Für dieses Programm ist ein zusammenhängender Speicher von mindestens 42 Kbyte nötig. Die Adressen der COL 256 sollten wie folgt liegen:

Port:	Adresse:	Jumper:						
		.	.	.	.	.	.	
		.	.	.	.	.	.	
I/O	$AC $AD $AE	nein	nein	ja	nein	ja	nein	(Schon auf Platine)
		.	.	.	.			
		.	.	.	.			
RAM	$dc000	nein	ja	nein	nein			

Für die Ramadresse kann aber auch jede andere genommen werden, jedoch muss dann das COL Programm neu angepasst werden. Ausserdem sollte man einen Standart einhalten, um mit anderen Programmen voll Kompatibel zu sein.

Aufbau der COL 256:
Bei dem Zusammenbau der COL 256 muss man besonders sorgfältig arbeiten und auch nur Marken IC's benutzen, da sich sonst unschöne Störungen ergeben können. Bei den Speichern sollte man sich, falls nicht schon vorhanden, die etwas teureren 64K*4 RAM's (4464 / 41254 etc.) kaufen. Diese haben den Vorteil, dass sich dann bis zu 4 Grafikseiten und sehr hochauflösende Grafiken darstellen lassen.

Bei der Auswahl des geeigneten Farbmonitor ist darauf zu achten, dass er verschiedene Eingänge besitzt. Denn die COL 256 hat als Ausgänge 1. einen Euro-Scart Stecker und 2. einen IBM Kompatiblen Stecker. Der Euro-Scart Stecker liefert als Ausgang drei Analogsignale, wodurch theoretisch unendlich viele Farben möglich sind. Bei der COL 256 sind es 256 Farben, welche über eine Farbtabelle auf 262144 erweiterbar sind. Diese drei Analogsignale bestehen aus den 3 Grundfarben ROT, GRÜN und BLAU, und den Vertikal und Horizontal Austastsignalen. Der IBM Stecker schränkt die Farbmöglichkeiten der COL 256 ein. Es lassen sich nur 16 Farben darstellen da die Ausgänge Digital sind. Dieses hat aber den Vorteil das sich in horizontaler Richtung die Auflösung verdoppelt. Das gesammte Bild wird also wesentlich feiner, was sich am besten bei senkrechten Linien beweisst.
Als Monitor kann ich den Grundig BGC 36 sehr empfehlen. Er hat ein sehr scharfes, farbenklares Bild und hat beide oben genannten Eingänge.

Anleitung zum Umgang mit der Befehlserweiterung:

Das COL 256 Farbgrafikprogramm wird auf einer Diskette geliefert und kann über Floppystart oder über Jogi DOS gestartet werden. Weitere Betriebssysteme wie CP/M 68k und Jados sind bereits in Arbeit. Beim ersten Laden des Programms muss die Diskette über Floppystart (4 im Grundprogramm) gestartet werden. Hier müssen nun diverse Eingaben vorgenommen werden.

```
Art der CPU 68008 = 1  68000 = 2  68020 = 4  ? [ ]
Frei verfügbarer Speicher (min. 42Kbyte)     ? [$xxxxxx]
```

Bei der Angabe des Prozessors, wird eines der 3 Programmodule ausgewählt. Dieses ist wichtig da sich die Speicherbelegung bei den Prozessoren unterschiedlich gestaltet.
Den frei verfügbaren Speicher sollte man dorthin auswählen, wo er dem Programmieren am wenigsten stört. Es ist allerdings darauf zu achten, dass der Speicher sich nicht mit dem Stack (A7) überschneidet. Es ist daher günstig, den Speicher an den Anfang der letzten 64Kbyte zu legen. (z.B. Ram von $20000-$5ffff , dann sollte der freie Speicher bei $50000 liegen, um keine Störungen zu erhalten).

Nach dieser Einstellung wird noch einmal abgefragt, ob die Eingaben richtig sind, um Fehler noch zu verbessern. Wenn dieses mit JA bestätigt wird, dann wird die Diskette umgeschrieben und ein Autostart mit Integriert. Mit dem Autostart lässt sich dann das COL Programm starten. Dabei wird dann der Bildschirm gelöscht und die COL 256 Initialisiert.

Um nun Programme mit dem COL 256 Farbgrafikprogramm zu schreiben, wird im normalen Texteditor ein Programm geschrieben wie sonst auch. Das Programm sollte allerdings nicht das A4 Register beinhalten, da dieses ein Zeiger für einen Speicher darstellt. Weiterhin darf das Programm keine Labels (Symbole) enthalten, die in der Befehlserweiterung genauso heissen. Ausnahmen bestehen allerdings wenn als Einsprünge Trap #6 benutzt wird. Hierbei wird dann keine Symboltabelle erzeugt.

Schreiben eines Programm:

1. Laden des COL 256 Farbgrafikprogramm von Diskette über Floppystart oder über Jogi DOS.

2. Schreiben des Programm im Texteditor.

Beispiel mit JSR

```
start:
 move.b #0,d0
 jsr colinit

 move.b #$03,d0
 move.w #100,d1
 move.w #100,d2
 jsr setfdot

rts
```

Beispiel mit TRAP #6

```
start:
 move.b #0,d0
 move.w #0,d7
 trap #6
 move.b #$03,d0
 move.w #100,d1
 move.w #100,d2
 move.w #6,d7
 trap #6
rts
```

Wie man deutlich sieht, kann man das Programm auf 2 verschiedene Arten schreiben. Einmal mit den einfachen JSR Befehlen, und zum anderen mit dem TRAP #6 Befehl. Das Programm mit dem JSR ist etwas Kürzer und auch etwas einfacher zu schreiben als das andere Programm. Der TRAP #6 hat den Nachteil, das er etwas länger in der Befehlsabarbeitung dauert, hat aber den Vorteil, dass die Programme als Daten auch auf anderen Systemen laufen, also Portabel sind. Die Übergabe des Index wird wie im Grundprogramm über D7 gemacht, wobei der Index mit der Nummer des Befehls übereinstimmt. (z.B. MOVETO = 14).

3. Übersetzen des Programm
Übersetzt wird das Programm dadurch, dass in der Bibliothek nun das Programm "COL ass " gestartet wird. Dabei wird dann der Bildschirm gelöscht, der TRAP #6 Einsprung festgelegt, eine Symboltabelle angelegt, welche alle Befehle enthält und dann das gesammte Programm Assembliert.
Weiterhin gibt es noch ein Bibliothekprogramm mit dem Namen "COL init". Hierbei wird nur die COL 256 Initialisiert, der Bildschirm gelöscht und der TRAP #6 eingetragen.

4. Starten des Programm
Nach dieser Aktion ist das Programm fertig und sollte nun, wenn keine Fehler angezeigt wurden, ablauffähig sein. Es kann nun gestartet werden. Zeigt der Monitor, nach dem Starten, jedoch nur ein flimmerndes Bild, dann sollte man noch einmal überprüfen ob die Adressen der COL 256 richtig eingestellt sind.

Installieren des Bootprogramm:

1. Es müssen folgende Dateien geladen werden:
   Boot Generator
   Boot 1
   Boot 2

2. Assemblieren des "Boot Generator".

3. Starten des "Boot Generator" mit "START".

4. Es wird Boot 1 und Boot 2 automatisch assembliert und auf Track 0 geschrieben.

Hardwareeigenschaften des CRT 6845.

Der CRT 6845 ist der Baustein, der den gesammten Speicher der COL 256 verwaltet. Er gibt automatisch die Speicherinhalte seriell auf den Bildschirm aus. Gleichzeitig kümmert er sich noch um den Refresh der Dynamischen Speicher und gibt Datenzugriffe von Aussen durch die CPU frei. Wenn nun ein Datenzugriff von der CPU während der Ausgabe eines Bytes an den Bildschirm oder während eines Refresh-Zyklus kommt, dann wird die WAIT Leitung aktiv "LOW" und die CPU muss kurz warten. Danach werden dann die Daten für die CPU freigegeben, und es kann ein Zugriff erfolgen. Weiterhin erzeugt der CRT alle nötigen Signale für die Bild-synchronisation (VS,HS). Über diese Siganle hinaus hat der CRT alle Steuersignale , die für den Betrieb dynamischer Speicher notwendig sind (RAS,CAS,Daten,MUX-Adressen). Ein Ausgang, der bei der COL 256 nicht beschaltet ist, ist der Cursor Ausgang. Dieser erzeugt ein Signal, welches man hardwaremässig dazu gebrauchen kann, einen bis zu 4 mal 4 grossen Cursor an beliebiger Stelle einzublenden. Hierbei wird der Speicher nicht einmal verändert. Als besonderen Bonus kann man dem CRT mitteilen, auf welcher Speicherzelle er mit der Bildschirmausgabe beginnen soll. Dadurch ist es möglich, mehrere Seiten anzusprechen und auch Bilder zu scrollen.

Register des CRT:
Der CRT besitzt 2 Register. Das eine wird dazu benutzt, um eines der 18 Internen Register auszuwählen und über das andere Register wird das zugehörige Datenwort übergeben.

Index	Registerart
0	horizontal total
1	horizontal displayed
2	horizontal sync position
3	sync width
4	vertikal total
5	vertikal total adjust
6	vertikal display
7	vertikal sync position
8	interlace mode and skew
9	max scan line adress
10	cursor start
11	cursor end
12	start adress (H)
13	start adress (L)
14	cursor (H)
15	cursor (L)
16	light pen (H)
17	light pen (L)

Genaueres in den Datenbüchern von Motorola.

Farbspectrum:
Die COL 256 kann 256 Farben darstellen. Dieses sind alles Mischfarben aus den 3 Grundfarben ROT, GRÜN und BLAU. Die Farben sind im Speicher folgendermassen organisiert:

Bit	7	6	5	4	3	2	1	0
Farbe	GRAU		BLAU		GRÜN		ROT	

Das höherwertige Bit ergibt eine grössere Helligkeit als das niedrigere.
Beispiel:

Bit	7	6	5	4	3	2	1	0	
Farbe									
Hellrot	0	0	0	0	0	0	1	1	$03
Hellgrün	0	0	0	0	1	1	0	0	$0c
Hellblau	0	0	1	1	0	0	0	0	$30
Grau	1	1	0	0	0	0	0	0	$c0
Dunkelrot	0	0	0	0	0	0	0	1	$01
Weiss	1	1	1	1	1	1	1	1	$ff
Gelb	0	0	0	0	1	1	1	1	$0f
Violett	0	0	1	1	0	0	1	1	$33

Die Grauwerte ergeben sich aus einer gleichmässigen Mischung von ROT, BLAU und GRÜN. Eine Kombination von Grau und Rot ergibt ein helles Rot, welches allerdings einen leichten Grauschimmer zeigt. Durch gekonnte Farbmischung lässt sich so ein kontrastreiches Bild aufbauen.
Es handelt sich hierbei um additive Farbmischung (Lichtmischung).

COLINIT
Initialisieren der COL 256 und löschen des Bildschirm.

Dieser Befehl sollte am Anfang eines jeden Programmes stehen. Hierbei wird der CRT auf seine Bildschirmgrösse eingestellt, eine Tabelle für die Speicheradressen entworfen und der gesammte Bildspeicher gelöscht. Gleichzeitig wird hiermit der Grafikmode bestimmt. Der Grafikmode sagt dem Programm um welche Auflösung es sich in vertikaler und horizontaler Richtung handelt. So z.B. kann ein Bild im Interlace Sync Mode betrieben werden, was besagt, dass das Bild aus 2 Halbbildern besteht. Dadurch lässt sich in vertikaler Richtung die Auflöschung verdoppeln, wobei das Bild aber etwas zu flimmern beginnt, da das Bild nun nicht mehr aus einem, sondern aus 2 Teilen besteht, die nacheinander ausgegeben werden, und somit die Wiederholfrequenz halbiert.

```
D0      Auswahl der Grafikauflösung
        0 = 256 * 256  50 Hz
        1 = 256 * 512  25 Hz (Interlace)
```

```
START:
 MOVE.W #0,D0           * Auflösung 256 * 256
 JSR COLINIT            * und aufrufen
 RTS

* Es wird die COL 256 Initialisiert und der Bildschirm gelöscht.

START:
 MOVE.W #0,D0
 MOVE.W #0,D7           * COLINIT anwählen
 TRAP #6
 RTS

* Das Gleiche nur mit TRAP #6 Aufruf anstatt JSR.
```

Der Hilfspeicher wird folgendermassen gesetzt:

```
Leseseite       $00
Schreibseite    $00
Cursorspeed     $00     (Cursor off)
Cursorx         $00
Cursory         $00
Color           $00
ZColor          $FF
Setmode         $00
```

INIT

Initialisieren der COL 256.

Dieser Befehl ist mit COLINIT gleich. Es wird nur der Bildschirm nicht gelöscht.

D0 Auswahl der Grafikauflösung
0 = 256 * 256 50 Hz
1 = 256 * 512 25 Hz (Interlace)

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR INIT                * und aufrufen
 RTS

* Es wird die COL 256 Initialisiert.

START:
 MOVE.W #0,D0
 MOVE.W #1,D7            * INIT anwählen
 TRAP #6
 RTS

* Das Gleiche nur mit TRAP #6 Aufruf anstatt JSR.
```

CLR
Löschen aller Bildseiten.

Der CLR Befehl löscht den gesammten Bildspeicher mit der Farbe Schwarz.
Dadurch wird auch der Bildschirm Schwarz.

Keine Register

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #0,D1            * Startkoordinaten auf 0,0 legen
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$03,D0          * Farbe = ROT
 MOVE.W #255,D1          * Endcoordinaten auf 255,255
 MOVE.W #255,D2
 JSR DRAWTO
 MOVE.L #20,D0           * Verzögerung um 2 Sekunden
 JSR DELAY
 JSR CLR                 * Bildschirm löschen
 RTS

* Es wird der Bildschirm gelöscht und eine Rote Linie von unten links
* nach oben rechts gezogen. Dann wartet der Computer 2 Sekunden und
* löscht anschliessend das Bild.
```

CLRCOL
Löschen aller Bildseiten mit einer Farbe

Der CLRCOL Befehl löscht den gesammten Bildspeicher mit der Farbe in D0.
Gleichzeitig wird die Hintergundfarbe (Background) mit D0 belegt.

D0 Farbe mit der der Bildschirm gelöscht wird.

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #$30,D0          * Farbe = BLAU
 JSR CLRCOL
 MOVE.W #0,D1            * Startkoordinaten auf 0,0 legen
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$03,D0          * Farbe = ROT
 MOVE.W #255,D1          * Endcoordinaten auf 255,255
 MOVE.W #255,D2
 JSR DRAWTO
 RTS

* Es wird der Bildschirm mit der Farbe Blau gelöscht und danach eine
* Rote Linie von unten links nach oben rechts gezogen.
```

CLPG
Löschen einer Bildseite.

Der CLPG Befehl löscht den Bildspeicher der Seite D1.
Gleichzeitig wird die Hintergundfarbe (Background) mit 0 belegt.
Es kann also eine unsichtbare Seite gelöscht werden, ohne das
auf dem Bildschirm Störungen entstehen.

D1 Seite [0-3] die gelöscht werden soll.

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie zeichnen
 MOVE.L #10,D0
 JSR DELAY               * 1 Sekunde verzögern
 MOVE.B #0,D1
 JSR CLPG                * Seite 0 löschen
RTS
```

CLPGCOL
Löschen einer Bildseite.

Der CLPGCOL Befehl löscht den Bildspeicher der Seite D1 mit der Farbe D0. Gleichzeitig wird die Hintergundfarbe (Background) mit D0 belegt. Es kann also eine unsichtbare Seite gelöscht werden, ohne das auf dem Bildschirm Störungen entstehen.

D0 Farbe mit der gelöscht wird.
D1 Seite [0-3] die gelöscht werden soll.

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.B #$0C,D0
 MOVE.B #0,D1
 JSR CLPGCOL             * Seite 0 mit Gruen löschen
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie zeichnen
 MOVE.L #10,D0
 JSR DELAY               * 1 Sekunde verzögern
 MOVE.B #0,D1
 JSR CLPG                * Seite 0 löschen
RTS
```

SETFDOT
Setzen eines Punktes.

Dieser Befehl setzt einen Punkt in der Farbe D0 an die X und Y Position. Ein Punkt wird extrem schnell gesetzt, wodurch sich Bilder schnell aufbauen lassen. Als Ausgabe wird die Punktadresse in A0 geliefert. Dadurch lassen sich Punkte wiederholt sehr schnell setzen.

D0	Farbe des Punktes	>A0	Adresse des Punktes
D1	X-Position		
D2	Y-Position		

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
LOOP:
 MOVE.W #255,D0
 JSR @RND                * Zufallszahl 0-255
 MOVE.W D0,D1            * X-Position
 MOVE.W #255,D0
 JSR @RND                * Zufallszahl 0-255
 MOVE.W D0,D2            * Y-Position
 MOVE.W #255,D0
 JSR @RND                * Farbe
 JSR SETFDOT             * Punkt setzen
 BRA LOOP                * Endlos wiederholen
```

SETDOT
ODER-Verknüpfung mit einem Punkt.

Ein Punkt an der X und Y Position wird mit D0 ODER verknüpfet.
Der Punkt wird extrem schnell gesetzt, wodurch sich fertige
Bilder schnell modifizieren lassen. Als Ausgabe wird die Punktadresse
in A0 geliefert. Dadurch lassen sich Punkte wiederholt sehr schnell
setzen.

Oder-Verknüpfung:
0 ODER 0 = 0
0 ODER 1 = 1
1 ODER 0 = 1
1 ODER 1 = 1

D0	Wert der Farbe	>A0	Adresse des Punktes
D1	X-Position		
D2	Y-Position		

```
START:
 MOVE.W #0,D0           * Auflösung 256 * 256
 JSR COLINIT
LOOP:
 MOVE.W #255,D0
 JSR @RND               * Zufallszahl 0-255
 MOVE.W D0,D1           * X-Position
 MOVE.W #255,D0
 JSR @RND               * Zufallszahl 0-255
 MOVE.W D0,D2           * Y-Position
 MOVE.W #255,D0
 JSR @RND               * Farbe
 JSR SETDOT             * Punkt setzen
 BRA LOOP               * Endlos wiederholen

* Im Gegensatz zum SETFDOT wird hierbei nach einer gewissen Zeit
* der Bildschirm Weiss.
```

XORDOT
Exklusiv ODER-Verknüpfung mit einem Punkt.

Ein Punkt an der X und Y Position wird mit D0 EXOR verknüpfet. Dieses entspricht einem Invertieren der Farben. So lassen sich Farben sehr schnell umdrehen. Als Ausgabe wird die Punktadresse in A0 geliefert. Dadurch lassen sich Punkte wiederholt sehr schnell setzen.

Invertieren der einzelnen Bits:
0 EXOR 0 = 0
1 EXOR 0 = 1
0 EXOR 1 = 1
1 EXOR 1 = 0

D0	Wert der Invertierung	>A0	Adresse des Punktes
D1	X-Position		
D2	Y-Position		

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.B #$0F,D0
 MOVE.W #128,D1
 MOVE.W #128,D2
 JSR SETFDOT
LOOP:
 MOVE.L #5,D0            * 1/2 Sekunde warten
 JSR DELAY
 MOVE.B #$FF,D0          * Jedes Bit invertieren
 MOVE.W #128,D1
 MOVE.W #128,D2
 JSR XORDOT
 BRA LOOP                * Unendlich wiederholen
```

ANDDOT

UND-Verknüpfung mit einem Punkt.

Ein Punkt an der X und Y Position wird mit D0 UND verknüpfet. Die Farbe dessen Bit = 0 ist wird gelöscht. Somit kann man aus Bildern Farben entfernen. Als Ausgabe wird die Punktadresse in A0 geliefert. Dadurch lassen sich Punkte wiederholt sehr schnell setzen.

Und-Verknüpfung:
0 UND 0 = 0
0 UND 1 = 0
1 UND 0 = 0
1 UND 1 = 1

D0	Wert der Farbe	>A0	Adresse des Punktes
D1	X-Position		
D2	Y-Position		

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.B #$FF,D0          * Farbe Weiss
 JSR CLRCOL
LOOP:
 MOVE.W #255,D0
 JSR @RND                * Zufallszahl 0-255
 MOVE.W D0,D1            * X-Position
 MOVE.W #255,D0
 JSR @RND                * Zufallszahl 0-255
 MOVE.W D0,D2            * Y-Position
 MOVE.W #255,D0
 JSR @RND                * Farbe
 JSR ANDDOT              * Punkt setzen
 BRA LOOP                * Endlos wiederholen

* Im Gegensatz zum SETDOT wird hierbei nach einer gewissen Zeit
* der Bildschirm Schwarz.
```

GETCOL
Farbe eines Punktes lesen.

Es wird die Farbe des Punktes an X und Y Position gelesen.
Mit diesem Befehl kann man den Bildschirm schnell auslesen.
Als Ausgabe wird die Punktadresse in A0 und Punktfarbe in D0
geliefert.

D1	X-Position	>A0	Adresse des Punktes
D2	Y-Position	>D0	Farbe des Punktes

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.B #$FF,D0          * Farbe Weiss
 JSR CLRCOL
 MOVE.W #128,D1
 MOVE.W #128,D2
 JSR GETCOL              * Farbe des Punktes 128,128 auslesen
RTS                      * Ergebnis steht in D0
```

CALCDOT
Berechen der Punktadresse.

Durch angeben der X und Y Position wird die Adresse des Punktes berechnet. Die Punktadresse wird in A0 geliefert.

D1	X-Position	>A0	Adresse des Punktes
D2	Y-Position		

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.B #$03,D0          * Farbe ROT
 JSR CLRCOL
 MOVE.W #128,D1
 MOVE.W #128,D2
 JSR CALCDOT             * Adresse des Punktes 128,128 berechnen
 MOVE.B #$01,(A0)        * Ergebnis steht in A0
LOOP:
 MOVE.L #5,D0
 JSR DELAY               * 1/2 Sekunde Verzögerung
 ROL.B #1,(A0)           * Schieben nach links (Farbänderung)
 BRA LOOP                * Unendlich wiederholen
```

CALCZEILE
Adresse des Zeilenanfangs berechnen.

Um sehr schnell eine gesammte Zeile zu beschreiben gibt es den CALCZEILE Befehl. Der Befehl errechnet die Adresse des Zeilenanfangs. Um nun einen bestimmten Bildpunkt in einer Zeile zu erreichen, wird nur ein einzelner Befehl benutzt:
MOVE.B D0,0(A0,D1.W)
Bei diesem Beispiel steht in D0 die Farbe und in D1 die X-Position. Hierdurch können Punkte in einer Zeile wesentlich schneller angesprochen werden, als wenn jeder Punkt über SETFDOT gesetzt wird.

D2	Y-Position	>A0	Adresse des Zeilenanfangs

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #255,D2          * in Zeile 255 beginnen
LOOP1:
 JSR CALCZEILE           * Ergebnis in A0
 MOVE.W #256-1,D1        * 256 Punkte zeichen
LOOP:
 MOVE.B D1,0(A0,D1.W)    * Punkt setzen
 DBRA D1,LOOP            * 1. Zeile zu ende Zeichnen
 DBRA D2,LOOP1           * 2. Höhe runterzählen
RTS
```

LINE
Ziehen einer Linie.

Um eine Linie zu zeichnen gibt es sehr viele Möglichkeiten.
Die in diesem Programm benutzte Routine benutzt den bekannten
Bresenham Algorithmus. Dieser Algorithmus ist relativ schnell
und ergibt eine Spiegelbildliche Linie. Andere sind zwar schneller,
ergeben aber nicht diese Genauigkeit beim Zeichnen.
Um nun mit dem LINE Befehl zu arbeiten, muss man ihm die
Anfangskoordinaten und die Endkoordinaten übergeben. Dieses geschieht
in dem Speicher X1,Y1 und X2,Y2. Der Speicher wird über den Befehl
SETXYXY gesetzt (siehe Seite 71). Weiterhin wird noch die Farbe
und der Setmode benötigt. Die Farbe wird im Speicher COLOR und
der Setmode in Speicher SETMODE übergeben.
(SETCOLOR siehe Seite 73)
(SETSETMODE siehe Seite 75)

```
X1       X-Startposition
Y1       Y-Startposition
X2       X-Endposition
Y2       Y-Endposition
COLOR    Farbe
SETMODE  Art des Punktesetzen
```

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 MOVE.W #255,D3
 MOVE.W #255,D4
 JSR SETXYXY             * Setzen von X1,Y1,X2,Y2
 MOVE.B #$30,D0
 JSR SETCOLOR            * Setzen der Farbe
 MOVE.W #0,D0
 JSR SETSETMODE          * Punkt fest setzen
 JSR LINE                * Linie ziehen
RTS
```

MOVETO
Merken der X und Y Position.

Diese Funktion erleichtert das Arbeiten mit dem LINE-Befehl.
Er ist mit dem Befehl aus dem Grundprogramm gleich und merkt
sich nur die Aktuelle X und Y Position. Es ergibt sich also
auf dem Bildschirm keine Ausgabe.

D1 X-Startposition
D2 Y-Startposition

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO              * Setzen des Anfangspunktes auf 0,0
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie zeichnen
RTS
```

DRAWTO
Merken der X und Y Position.

Diese Funktion erleichtert das Arbeiten mit dem LINE-Befehl. Er ist mit dem Befehl aus dem Grundprogramm gleich und zieht eine Linie von der durch MOVETO gemerkten Position. Die Farbe wird im Register D0 übergeben.

D0	Farbe der Linie
D1	X-Startposition
D2	Y-Startposition

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO              * Setzen des Anfangspunktes auf 0,0
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie Ziehen
RTS
```

SCROLLX

Scrollen des Bildschirm in X-Richtung.

Mit diesem Befehl ist ein Scrollen des Bildschirms in X-Richtung möglich, ohne den Speicher verschieben zu müssen. Dadurch ist ein Scrollen nebenbei sehr einfach und es lassen sich Spiele recht schnell bewegen. Es müssen aber 2 Einschränkungen gemacht werden.
1. Der Bildschirm lässt sich immer nur um 4 Bildpunkte verschieben, wodurch sich ein extrem feines Scrollen nur durch Speicherverschieben realisieren lässt. Das wird aber dadurch kompensiert, dass der Scroll-befehl sich die Anzahl der noch nicht gescrollten Punkte merkt und dann später ausführt, wenn wieder 4 Punkte zusämmen sind.
2. Zwischen dem linken und dem rechten Rand des Orginalbild ergibt sich beim Scrollen ein versatz um 4 Bildpunkte nach unten. Wieder ein Problem welches sich nur durch Speicherverschieben realisieren lässt.

D0 Anzahl Bildpunkte um die gescrollt werden soll
(Eine negative Zahl ergibt scrollen nach rechts)

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO              * Setzen des Anfangspunktes auf 0,0
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie ziehen
LOOP:
 BSR SYNC                * Vertikalen Syncron abwarten da sonst
 BEQ.S SYNC              * zu schnell.
 MOVE.W #4,D0
 JSR SCROLLX             * Scrollen um je 4 Punkte
 BRA LOOP                * Endlos wiederholen
```

SCROLLY
Scrollen des Bildschirm in Y-Richtung.

Mit diesem Befehl ist ein Scrollen des Bildschirms in Y-Richtung möglich, ohne den Speicher verschieben zu müssen. Dadurch ist ein Scrollen nebenbei sehr einfach und es lassen sich Spiele recht schnell bewegen. Es muss aber eine Einschränkung gemacht werden. Der Bildschirm lässt sich immer nur um 4 Bildpunkte verschieben, wodurch sich ein extrem feines Scrollen nur durch Speicherverschieben realisieren lässt. Das wird aber dadurch kompensiert, dass der Scrollbefehl sich die Anzahl der noch nicht gescrollten Punkte merkt und dann später ausführt, wenn wieder 4 Punkte zusammen sind.
Bei diesem Befehl zeigt sich der Vorteil der grossen Speicher wie auf Seite F beschrieben. Bei dem kleinen Speicherausbau ergibt sich ein Scrollen über eine Seite, wogegen beim grossen Speicherausbau der Bildschirm durch alle 4 Seiten scrollt.

D0 Anzahl Bildpunkte um die gescrollt werden soll
(Eine negative Zahl ergibt scrollen nach unten)

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO              * Setzen des Anfangspunktes auf 0,0
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie ziehen
LOOP:
 BSR SYNC                * Vertikalen Syncron abwarten da sonst
 BEQ.S SYNC              * zu schnell.
 MOVE.W #4,D0
 JSR SCROLLY             * Scrollen um je 4 Punkte
 BRA LOOP                * Endlos wiederholen
```

SCROLLTO

Scrollen des Bildschirm auf die X und Y-Position.

Es ist auch möglich den Bildschirm auf eine bestimmte X und Y Position zu setzen. So lässen sich z.B. Teile von anderen Grafikseiten einblenden. Es muss aber eine Einschränkung gemacht werden. Der Bildschirm lässt sich immer nur um 4 Bildpunkte verschieben, wodurch sich ein extrem feines Scrollen nur durch Speicherverschieben realisieren lässt.

Bei diesem Befehl zeigt sich der Vorteil der grossen Speicher wie auf Seite F beschrieben. Bei dem kleinen Speicherausbau ergibt sich ein Scrollen über eine Seite, wogegen beim grossen Speicherausbau der Bildschirm durch alle 4 Seiten scrollt.

D1 X-Position
D2 Y-Position

```
START:
 MOVE.W #0,D0            * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO              * Setzen des Anfangspunktes auf 0,0
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie ziehen
 MOVE.W #20,D1
 MOVE.W #40,D2
 JSR SCROLLTO            * Scrollen auf X = 20  Y = 40
RTS
```

NEWPAGE
Lese- und Schreibseite auswählen.

Die COL 256 besitzt bis zu 4 Grafikseiten. Um diese Seiten ansprechen zu können, muss man entweder den Scroll Befehl benutzen oder man kann auch den NEWPAGE Befehl benutzen. NEWPAGE hat den angenehmen Vorteil, dass man bestimmen kann auf welcher Seite gezeichnet wird und auf welcher Seite das Bild angezeigt wird. Man kann nun z.B. als Leseseite die Seite 0 nehmen und als Schreibseite die Seite 1. Dadurch wird das Bild unsichtbar auf einer Seite aufgebaut, während man sich das andere Bild betrachten kann. Dieses wird benötigt um flimmerfreie Grafiken zu erzeugen.

Dieser Befehl funktioniert nur mit dem Grossen Speicherausbau!

D0	Leseseite	[0-3]
D1	Schreibseite	[0-3]

```
START:
 MOVE.W #0,D0           * Auflösung 256 * 256
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO
 MOVE.L #10,D0
 JSR DELAY              * Verzögern um 1 Sekunde
 MOVE.B #0,D0           * Leseseite = 0
 MOVE.B #1,D1           * Schreibseite = 1
 JSR NEWPAGE
 MOVE.W #255,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$0C,D0
 MOVE.W #0,D1
 MOVE.W #255,D2
 JSR DRAWTO
 MOVE.B #1,D0           * Leseseite = 1
 MOVE.B #1,D1           * Schreibseite = 1
 JSR NEWPAGE
RTS
```

SAVE
Speichern einer Grafikseite auf Floppy unter Jogi DOS.

Da man ohne Probleme auf den Speicher der COL 256 zugreifen kann, besteht die Möglichkeit eine Grafikseite auf Floppy Disk zu speichern. Als Angaben werden nur die Bildseite, die Laufwerkseite und die Adresse des Namen, unter dem die Grafik gespeichert wird, benötigt. Als Betriebssystem wird Jogi DOS benutzt. Eine Grafikseite ist jeweils 64 Kbyte lang. Beim Speichern der grösseren Grafikseite (256 * 512) muss allerdings das Bild in 2 Teilen gespeichert werden.

```
D0      Grafikseite   [ 0-3 ]
D1      Laufwerk      [ 0-3 ]   + $80 für Rückseite
A0      Zeiger auf die Adresse des Namen
```

```
SPEICHERN:
 MOVE.B #0,D0           * Seite 0 Speichern
 MOVE.B #$81,D1         * Rückseite Laufwerk 1
 LEA NAME,A0            * Zeiger auf Namen
 JSR SAVE
RTS

NAME: DC.B 'Grafik test',0      * Name
DC.L 0,0                        * Auf 16 Buchstaben mit 0 auffüllen
```

LOAD
Laden einer Grafikseite von Floppy unter Jogi DOS.

Das Gegenstück zu SAVE ist der LOAD Befehl. Damit wird ein Bild geladen. Als Angaben werden nur die Bildseite, die Laufwerkseite und die Adresse des Namen, unter dem die Grafik gespeichert ist, benötigt. Als Betriebssystem wird Jogi DOS benutzt. Eine Grafikseite ist jeweils 64 Kbyte lang. Beim Laden der grösseren Grafikseite (256 * 512) muss allerdings das Bild in 2 Teilen geladen werden.

```
D0      Grafikseite    [ 0-3 ]
D1      Laufwerk       [ 0-3 ]   + $80 für Rückseite
A0      Zeiger auf die Adresse des Namen
```

```
LADEN:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #0,D0           * Seite 0 Speichern
 MOVE.B #$81,D1         * Rückseite Laufwerk 1
 LEA NAME,A0            * Zeiger auf Namen
 JSR LOAD
RTS

NAME: DC.B 'Grafik test',0      * Name
DC.L 0,0                        * Auf 16 Buchstaben mit 0 auffüllen
```

PAINT
Füllen einer Fläche mit einer Farbe.

Dieses ist der wohl aufwendigste Algorithmus im gesammten Programm. Es werden beliebige geschlossene Flächen mit der Farbe D0 gefüllt. Es wird nur ein Punkt innerhalb der Fläche angegeben. Dann füllt der Algorithmus die Zeile so lange aus, bis ein Punkt erreicht ist, der eine andere Farbe als der Startpunkt hat. Wenn dieses erfolgt ist, wird die nächste Zeile genommen, bis es keine weiteren ungesetzten Punkte mehr gibt.

D0	Farbe
D1	X-Positon
D2	Y-Position

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie von unten links nach oben rechts
 MOVE.B #$30,D0
 MOVE.W #128,D1          * Einen Punkt in der Fläche
 MOVE.W #0,D2
 JSR PAINT               * Eine Seite davon füllen
RTS
```

PAINTCOL

Füllen einer Fläche mit einer Farbe bis zur Farbgrenze.

Hier werden beliebige geschlossene Flächen mit der Farbe D0 gefüllt. Es wird nur ein Punkt innerhalb der Fläche angegeben. Dann füllt der Algorithmus die Zeile so lange aus, bis ein Punkt erreicht ist, der die Farbe hat, die in D0 steht. Wenn dieses erfolgt ist, wird die nächste Zeile genommen, bis es keine weiteren ungesetzten Punkte mehr gibt.

D0	Farbe
D1	X-Positon
D2	Y-Position

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie von unten links nach oben rechts
 MOVE.W #255,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$30,D0
 MOVE.W #0,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie von unten rechts nach oben links
 MOVE.B #$30,D0
 MOVE.W #128,D1          * Einen Punkt in der Fläche
 MOVE.W #0,D2
 JSR PAINTCOL            * Eine Seite davon füllen
RTS

* Es ist deutlich zu erkennen das nicht eine beliebige Fläche gefüllt
* wird, sondern nur die Fläche deren Begrenzung die Farbe D0 hat.
```

SYNC
Testen des Vertikalen Synchronimpuls

Damit ist es möglich den Vertikal-Synchronisationsimpuls der COL 256 abzufragen. Immer wenn ein Durchgang erfolgte, wird $FFFF im Register D0 übergeben, sonst der Wert 0. Hiermit lässt sich dann ein Bildschirmaufbau gleichmässig schnell und flimmerfrei aufbauen.

>D0 Synchronimpuls abfrage
$0000 = keine VS
$FFFF = VS war dar

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
LOOP:
 JSR SYNC              * Warten bis Synchronsignal
 BEQ.S LOOP            * Kurzer Sprung schneller
 MOVE.B #$0F,D0        * Punkt setzen
 JSR SETFDOT
 ADDQ.W #1,D1
 ADDQ.W #1,D2          * Diagonale Linie
 CMP.W #256,D1
 BNE LOOP
RTS

* Achtung bei TRAP #6:
* LOOP:
* MOVE.W #24,D7
* TRAP #6
* TST.W D0
* BEQ LOOP
```

DLEAY
Verzögerung um jeweils 100ms.

Um eine bestimmte Zeit zwischen 2 Abläufen verstreichen zu lassen, benutzt man oft Verzögerungsschleifen, welche nicht all zu genau sind. Dieser DELAY ist aber relativ genau, weil er 5 mal den Synchronimpuls der COL 256 abfragt und somit genau 5*20ms wartet. In D0 wird die Dauer des Delay eingegeben, wobei 1 = 100ms sind. Folgerichtig ist 10 = 1 sec. Verzögerung.
WICHTIG: in D0 muss ein Langwort stehen !

D0.L Verzögerung * 100ms

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
LOOP:
 MOVE.L #1,D0            * Warten bis Synchronsignal
 JSR DELAY               * Kurzer Sprung schneller
 MOVE.B #$0F,D0          * Punkt setzen
 JSR SETFDOT
 ADDQ.W #1,D1
 ADDQ.W #1,D2            * Diagonale Linie
 CMP.W #256,D1
 BNE LOOP
RTS
```

CLIPPEINFACH

Begrenzen der X und Y Koordinaten.

Um keinen Bildschirmüberlauf an den Rändern zu erhalten, wird dieser Befehl benutzt. Nach der Eingabe der X und Y Koordinaten in D1 und D2, werden diese Koordinaten gegen die Ränder getestet und falls nötig gekürzt. Dabei ist darauf zu achten, dass die gekürzten Werte nur die Minimal- oder Maximalwerte annehmen.

D1	X-Koordinate	>D1	gekürzte X-Koordinate
D2	Y-Koordinate	>D2	gekürzte Y-Koordinate

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #1000,D1        * Eindeutiger Überlauf
 MOVE.W #1000,D2        * Eindeutiger Überlauf
 JSR CLIPPEINFACH
 MOVE.B #$30,D0
 JSR SETFDOT
RTS
```

AMOVE
Absoluter Moveto.

Der AMOVE Befehl entspricht dem MOVETO Befehl und ist nur aus Kompatiblitätsgründen zu anderen Computern gewählt. Das A hierbei ist die Abkürzung für Absolut. Das heisst, dass die Koordinaten auf den 0,0 Punkt bezogen sind.

D1 X-Position
D2 Y-Position

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1            * Absolut 0,0
 MOVE.W #0,D2
 JSR AMOVE
 MOVE.B #$33,D0
 MOVE.W #255,D1          * Linie Absolut zu 255,255 ziehen
 MOVE.W #255,D2
 MOVE.W #0,D3            * Fest setzen
 JSR ADRAW
RTS
```

RMOVE
Relativer Moveto.

Der RMOVE Befehl ist ein MOVETO Befehl der nicht auf den 0,0 Punkt bezogen ist, sondern auf den letzten angesprochenen Bildpunkt.
Das R steht hier also für Relativ, wodurch die Werte in D1 und D2 auf den letzten angesprochenen Punkt aufaddiert (bzw. subtrahiert) werden.

D1 X-Position relativ zur alten X-Position
D2 Y-Position relativ zur alten Y-Position

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR AMOVE
 MOVE.B #$33,D0
 MOVE.W #100,D1
 MOVE.W #100,D2
 MOVE.W #0,D3            * Fest setzen
 JSR ADRAW
 MOVE.W #20,D1           * 20 Punkte nach rechts
 MOVE.W #-20,D2          * 20 Punkte nach unten
 JSR RMOVE
 MOVE.B #$FF,D0
 MOVE.W #20,D1           * 20 Punkte nach rechts
 MOVE.W #10,D2           * 10 Punkte nach oben
 MOVE.W #0,D3            * Fest setzen
 JSR RDRAW
RTS
```

ADRAW
Absoluter Drawto.

Der ADRAW Befehl entspricht dem DRAWTO Befehl und ist nur aus Kompatiblitätsgründen zu anderen Computern gewählt. Das A hierbei ist die Abkürzung für Absolut. Däs heisst, dass die Koordinaten auf den 0,0 Punkt bezogen sind. Es wird eine Linie, von der durch Moveto bestimmten Koordinaten, zu den in D1 und D2 angegebenen Koordinaten gezogen.
Mit D3 wird der Setmode bestimmt.

D0 Farbe
D1 X-Position
D2 Y-Position
D3 Setmode (siehe Seite 75)

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1          * Absolut 0,0
 MOVE.W #0,D2
 JSR AMOVE
 MOVE.B #$33,D0
 MOVE.W #255,D1        * Linie Absolut zu 255,255 ziehen
 MOVE.W #255,D2
 MOVE.W #0,D3          * Fest setzen
 JSR ADRAW
RTS
```

RDRAW
Relativer Moveto.

Der RDRAW Befehl ist ein DRAWTO Befehl, der nicht auf den 0,0 Punkt bezogen ist, sondern auf den letzten gesetzten Bildpunkt.
Das R steht hier also für Relativ, wodurch die Werte in D1 und D2 auf den letzten angesprochenen Punkt aufaddiert (bzw. subtrahiert) werden.
Es ergibt sich also eine Linie, von der durch Moveto bestimmten Koordinaten, zu dem relativ dazu liegenden Punkt.
Der Setmode bestimmt die Art der Linie.

D0	Farbe
D1	X-Position relativ zur alten X-Position
D2	Y-Position relativ zur alten Y-Position
D3	Setmode (siehe Seite 75)

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR AMOVE
 MOVE.B #$33,D0
 MOVE.W #100,D1
 MOVE.W #100,D2
 MOVE.W #0,D3            * Fest setzen
 JSR ADRAW
 MOVE.W #20,D1           * 20 Punkte nach rechts
 MOVE.W #-20,D2          * 20 Punkte nach unten
 JSR RMOVE
 MOVE.B #$FF,D0
 MOVE.W #20,D1           * 20 Punkte nach rechts
 MOVE.W #10,D2           * 10 Punkte nach oben
 MOVE.W #0,D3            * Fest setzen
 JSR RDRAW
RTS
```

APOINT
Absoluten Punkt setzen.

Der APOINT Befehl ist ein erweiterter Punktsetz-Befehl, mit dem man auswählen kann, wie der Punkt geändert werden soll. In D0 wird der Wert der Änderung übergeben.
Genaue Beschreibung siehe unten!

D0	Farbe (Wert der Änderung)	>D0	Ergebnis der Auswertung
D1	X-Position	>A0	Adresse des Punkt
D2	Y-Position	>CCR	In Abhängigkeit der
D3	Setmode		Abfrage gesetzt.

Setmode: (D3) (Alle Erklärungen beziehen sich auf den angesprochenen Bildpunkt in D1 D2)

0 setzen der Farbe D0	1 setzen mit Hintergrundfarbe
2 auslesen der Punktes nach D0	3 ODER-Verknüpfung mit D0
4 AND-Verknüpfung mit D0	5 EXOR-Verknüpfung mit D0
6 addieren mit D0	7 subtrahieren mit D0
8 prüfe Bit D0 und invertiere	9 prüfe Bit D0 und lösche
10 prüfe Bit D0 und setze	11 prüfe Bit D0
12 Punkt mit D0 vergleichen	13 negiere Punkt
14 NOT des Punkt	15 testen ob der Punkt = 0
16 logisches schieben links um D0	17 logisches schieben rechts
18 rotiere links um D0	19 rortiere rechts
20 arithmetrisches schieben links um D0	21 arith. schieben rechts
22 setze Punkt im Window	23 lese Punkt aus Window
24 Berechne Punkt im Window	

Zur genaueren Beschreibung jetzt noch die entsprechenden Assembler Befehle

0 MOVE.B D0,PUNKT	1 MOVE.B HINTERGUND,PUNKT
2 MOVE.B PUNKT,D0	3 OR.B D0,PUNKT
4 AND.B D0,PUNKT	5 EOR.B D0,PUNKT
6 ADD.B D0,PUNKT	7 SUB.B D0,PUNKT
8 BCHG.B D0,PUNKT	9 BCLR.B D0,PUNKT
10 BSET.B D0,PUNKT	11 BTST.B D0,PUNKT
12 CMP.B PUNKT,D0	13 NEG.B PUNKT
14 NOT.B PUNKT	15 TST.B PUNKT
16 LSL.B D0,PUNKT	17 LSR.B D0,PUNKT
18 ROL.B D0,PUNKT	19 ROR.B D0,PUNKT
20 ASL.B D0,PUNKT	21 ASR.B D0,PUNKT
22 BSR SETWDOT	23 BSR GETWDOT
24 BSR CALCWDOT	

RPOINT
Relativen Punkt setzen.

Der RPOINT Befehl ist ein erweiterter Punktsetz-Befehl, mit dem man auswählen kann, wie der Punkt geändert werden soll. In D0 wird der Wert der Änderung übergeben. Die Koordinaten sind relativ zu den des letzten Punktes.
Genaue Beschreibung siehe unten!

D0	Farbe (Wert der Änderung)	>D0	Ergebnis der Auswertung
D1	X-Position relativ	>A0	Adresse des Punkt
D2	Y-Position relativ	>CCR	In Abhängigkeit der
D3	Setmode		Abfrage gesetzt.

Setmode: (D3) (Alle Erklärungen beziehen sich auf den angesprochenen Bildpunkt in D1 D2)

0 setzen der Farbe D0	1 setzen mit Hintergrundfarbe
2 auslesen der Punktes nach D0	3 ODER-Verknüpfung mit D0
4 AND-Verknüpfung mit D0	5 EXOR-Verknüpfung mit D0
6 addieren mit D0	7 subtrahieren mit D0
8 prüfe Bit D0 und invertiere	9 prüfe Bit D0 und lösche
10 prüfe Bit D0 und setze	11 prüfe Bit D0
12 Punkt mit D0 vergleichen	13 negiere Punkt
14 NOT des Punkt	15 testen ob der Punkt = 0
16 logisches schieben links um D0	17 logisches schieben rechts
18 rotiere links um D0	19 rortiere rechts
20 arithmetrisches schieben links um D0	21 arith. schieben rechts
22 setze Punkt im Window	23 lese Punkt aus Window
24 Berechne Punkt im Window	

Zur genaueren Beschreibung jetzt noch die entsprechenden Assembler Befehle

0 MOVE.B D0,PUNKT	1 MOVE.B HINTERGUND,PUNKT
2 MOVE.B PUNKT,D0	3 OR.B D0,PUNKT
4 AND.B D0,PUNKT	5 EOR.B D0,PUNKT
6 ADD.B D0,PUNKT	7 SUB.B D0,PUNKT
8 BCHG.B D0,PUNKT	9 BCLR.B D0,PUNKT
10 BSET.B D0,PUNKT	11 BTST.B D0,PUNKT
12 CMP.B PUNKT,D0	13 NEG.B PUNKT
14 NOT.B PUNKT	15 TST.B PUNKT
16 LSL.B D0,PUNKT	17 LSR.B D0,PUNKT
18 ROL.B D0,PUNKT	19 ROR.B D0,PUNKT
20 ASL.B D0,PUNKT	21 ASR.B D0,PUNKT
22 BSR SETWDOT	23 BSR GETWDOT
24 BSR CALCWDOT	

ACALCDOT
Absoluter Drawto.

Der ACALCDOT Befehl entspricht dem CALCDOT Befehl und ist nur aus Kompatiblitätsgründen zu anderen Computern gewählt. Das A hierbei ist die Abkürzung für Absolut. Das heisst, dass die Koordinaten auf den 0,0 Punkt bezogen sind. Es wird also die Adresse des Punktes berechnet, auf die die Koordinaten D1 und D2 zeigen. Das Ergebnis steht in A0.

D1	X-Position	>A0	Adresse des Punktes
D2	Y-Position		

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #128,D1
 MOVE.W #128,D2          * Adresse der Koordinate 128,128 bestimmen
 JSR ACALCDOT
 MOVE.W #255,D3
LOOP:
 JSR SYNC                * Einige Zeit warten, sonst zu schnell
 BEQ.S LOOP
 MOVE.B D3,(A0)          * Setzen
 DBRA D3,LOOP
RTS
```

RCALCDOT
Absoluter Drawto.

Der RCALCDOT Befehl entspricht dem CALCDOT Befehl, aber die Koordinaten sind relativ zu den, des zu letzt angesprochenen Punktes.
Es wird also die Adresse des Punktes berechnet, der sich um D1 in X-Richtung und um D2 in Y-Richtung von dem alten Punkt befindet. Das Ergebnis steht in A0.

D1	X-Position relativ	>A0	Adresse des Punktes
D2	Y-Position relativ		

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #20,D1
 MOVE.W #10,D2
 JSR AMOVE              * Absoluten Move nach 20,10
 MOVE.W #128,D1
 MOVE.W #128,D2         * Adresse der Koordinate 128+20,128+10 bestimmen
 JSR RCALCDOT
 MOVE.W #255,D3
LOOP:
 JSR SYNC               * Einige Zeit warten, sonst zu schnell
 BEQ.S LOOP
 MOVE.B D3,(A0)         * Setzen
 DBRA D3,LOOP
RTS
```

POINT
Punkt setzen.

Der POINT Befehl ist ein erweiterter Punktsetz-Befehl, mit dem man auswählen kann, wie der Punkt geändert werden soll. In D0 wird der Wert der Änderung übergeben.
Genaue Beschreibung siehe unten!

D0	Farbe (Wert der Änderung)	>D0	Ergebnis der Auswertung
D1	X-Position	>A0	Adresse des Punkt
D2	Y-Position	>CCR	In Abhängigkeit der
D3	Setmode		Abfrage gesetzt.

Setmode: (D3) (Alle Erklärungen beziehen sich auf den angesprochenen Bildpunkt in D1 D2)

0 setzen der Farbe D0	1 setzen mit Hintergrundfarbe
2 auslesen der Punktes nach D0	3 ODER-Verknüpfung mit D0
4 AND-Verknüpfung mit D0	5 EXOR-Verknüpfung mit D0
6 addieren mit D0	7 subtrahieren mit D0
8 prüfe Bit D0 und invertiere	9 prüfe Bit D0 und lösche
10 prüfe Bit D0 und setze	11 prüfe Bit D0
12 Punkt mit D0 vergleichen	13 negiere Punkt
14 NOT des Punkt	15 testen ob der Punkt = 0
16 logisches schieben links um D0	17 logisches schieben rechts
18 rotiere links um D0	19 rortiere rechts
20 arithmetrisches schieben links um D0	21 arith. schieben rechts
22 setze Punkt im Window	23 lese Punkt aus Window
24 Berechne Punkt im Window	

Zur genaueren Beschreibung jetzt noch die entsprechenden Assembler Befehle

0 MOVE.B D0,PUNKT	1 MOVE.B HINTERGUND,PUNKT
2 MOVE.B PUNKT,D0	3 OR.B D0,PUNKT
4 AND.B D0,PUNKT	5 EOR.B D0,PUNKT
6 ADD.B D0,PUNKT	7 SUB.B D0,PUNKT
8 BCHG.B D0,PUNKT	9 BCLR.B D0,PUNKT
10 BSET.B D0,PUNKT	11 BTST.B D0,PUNKT
12 CMP.B PUNKT,D0	13 NEG.B PUNKT
14 NOT.B PUNKT	15 TST.B PUNKT
16 LSL.B D0,PUNKT	17 LSR.B D0,PUNKT
18 ROL.B D0,PUNKT	19 ROR.B D0,PUNKT
20 ASL.B D0,PUNKT	21 ASR.B D0,PUNKT
22 BSR SETWDOT	23 BSR GETWDOT
24 BSR CALCWDOT	

RECHTECK

Zeichnen eines Rechteck.

Mit dem RECHTECK Befehl kann man sehr schnell ein Rechteck mit einer beliebigen grösse zeichnen. Dadurch spart man sich das sonst recht umfangreiche Erstellen eines Rahmen.
In D0 wird die Farbe übergeben, in D1 und D2 die Startkoordinaten und in D3 und D4 die Endkoordinaten.

D0	Farbe
D1	X-Anfangsposition
D2	Y-Anfangsposition
D3	X-Endposition
D4	Y-Endposition

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$FF,D0
 MOVE.W #10,D1
 MOVE.W #10,D2
 MOVE.W #200,D3
 MOVE.W #50,D4
 JSR RECHTECK
RTS
```

FLAECHE
Zeichnen eines ausgefüllten Rechteck.

Mit dem FLAECHE Befehl kann man sehr schnell ein ausgefülltes Rechteck mit einer beliebigen grösse zeichnen. Mit dieser Hilfe ist z.B. der Zoom Befehl entstanden.
In D0 wird die Farbe übergeben, in D1 und D2 die Startkoordinaten und in D3 und D4 die Endkoordinaten.

D0	Farbe
D1	X-Anfangsposition
D2	Y-Anfangsposition
D3	X-Endposition
D4	Y-Endposition

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$3C,D0
 MOVE.W #10,D1
 MOVE.W #10,D2
 MOVE.W #200,D3
 MOVE.W #50,D4
 JSR FLAECHE
RTS
```

CIRCLE
Zeichnen eines Kreises.

Mit dem CIRCLE Befehl kann man sehr schnell einen Kreis mit einer beliebigen grösse zeichnen.
In D0 wird die Farbe übergeben, in D1 und D2 die Mittelpunktkoordinaten und in D3 der Radius.

D0	Farbe
D1	X-Mittelpunkt
D2	Y-Mittelpunkt
D3	Radius

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$3C,D0
 MOVE.W #128,D1
 MOVE.W #128,D2
 MOVE.W #100,D3
 JSR CIRCLE
RTS
```

DISK

Zeichnen eines ausgefüllten Kreises.

Mit dem DISK Befehl kann man sehr schnell einen ausgefüllten Kreis mit einer beliebigen grösse zeichnen.
In D0 wird die Farbe übergeben, in D1 und D2 die Mittelpunktkoordinaten und in D3 der Radius.

D0	Farbe
D1	X-Mittelpunkt
D2	Y-Mittelpunkt
D3	Radius

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$04,D0
 MOVE.W #128,D1
 MOVE.W #128,D2
 MOVE.W #100,D3
 JSR DISK
RTS
```

COXY
Setzen eines Zeichen an der X und Y Position.

Um ein Zeichen an einer bestimmten X und Y Position darzustellen, muss in diesem Befehl in D0 das Zeichen, in D1 die X und in D2 die Y Position übergeben werden. Die Farbe holt sich der Befehl aus dem ZCOLOR Speicher. Dieser wird über SETZCOLOR angesprochen.
Steuerzeichen werden ignoriert und es wird auch die X und Y Koordinate nicht hochgezählt. Dieses muss immer einzeln gemacht werden.
Die Zeichen sind 7 Punkte breit und 9 Punkte hoch. Bei Zeichen deren unterster Punkt wie beim kleinen "g" etwas tiefer stehen, wird das Zeichen automatisch um 3 Punkte tiefgestellt.

D0	Zeichen
D1	X-Position
D2	Y-Position
ZCOLOR	Zeichenfarbe

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0
 JSR SETZCOLOR          * Farbe auf ROT
 MOVE.B #'A',D0         * Zeichen "A"
 MOVE.W #128,D1         * Position 128,128
 MOVE.W #128,D2
 JSR COXY
 MOVE.B #'a',D0         * Zeichen "a"
 MOVE.W #100,D1         * Position 100,100
 MOVE.W #100,D2
 JSR COXY
RTS
```

SETCURXY

Positionieren des Cursor und Blinkrate einstellen.

Um bei Texteingabe und Textausgabe auch einen Cursor zu erhalten, muss dieser erst einmal Initialisiert werden. Mit D0 wird die Cursorfarbe und damit auch die Zeichenfarbe gesetzt. Mit D1 und D2 werden die Cursorkoordinaten gesetzt. Diese sind in einem Bereich von 0 bis 31 in X-Richtung und in einem Bereich von 0 bis 20 in Y-Richtung erlaubt. Mit D3 wird die Art des Cursor bestimmt. Wenn in D3 = $00 steht, dann wird der Cursor ausgeschaltet. Wenn in D3 = $FF steht, dann wird der Cursor dauernd eingeschaltet. Alle anderen Werte dazwischen ($01-$FE) geben die Blinkdauer an. Die Blinkdauer hängt von dem Synchronimpuls ab. Eine $01 im Register D3 bedeutet also, dass bei jedem Synchronimpuls der Cursor einmal wechselt. Bei $0a blinkt er also 2,5 mal in der Sekunde (alle 200ms ein wechsel). Man muss also nur 20ms * 2 * D3 nehmen, um die Blinkfrequenz zu errechnen.

```
D0      Farbe
D1      X-Position      [ 0-31 ]
D2      Y-Position      [ 0-22 ]
D3      Cursor Blinkfrequenz  $00 = aus  $01-$FE = anzahl Sync's
        $FF = dauernd an
```

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0          * Farbe Rot
 MOVE.W #10,D1           * X-Position
 MOVE.W #10,D2           * Y-Position
 MOVE.W #10,D3           * Blinkfrequenz = 20ms * 2 * 10 = 0,4 sec = 2,5 Hz
 JSR SETCURXY
 MOVE.L #30,D0
 JSR DELAY               * Einige Zeit blinken lassen
RTS
```

CI
Einlesen eines Zeichen von Tastatur.

Dieser CI Befehl entspricht dem @CI Befehl aus dem Grundprogramm, und wurde aus Kompatibilitäts-Gründen verwendet.
Es wird solange gewartet, bis eine Taste gedrückt wird, die dann auch prommt im Register D0 zur Verfügung steht.

>D0.L gedrückte Taste

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0          * Farbe Rot
 MOVE.W #0,D1            * X-Position
 MOVE.W #20,D2           * Y-Position
 MOVE.W #10,D3           * Blinkfrequenz = 20ms * 2 * 10 = 0,4 sec = 2,5 Hz
 JSR SETCURXY
LOOP:
 JSR CI                  * Zeichen lesen nach D0
 JSR CO                  * Zeichen ausgeben auf Bildschirm
 BRA LOOP
```

CSTS
Testen ob ein Zeichen eingegeben wurde.

Dieser CSTS Befehl entspricht dem @CSTS Befehl aus dem Grundprogramm, und wurde aus Kompatibilitäts-Gründen verwendet.
Es wird getestet ob eine Taste gedrückt wurde und danach das Register D0 gesetzt. Wenn keine Taste gedrückt wurde, enthält D0 = $00, sonst einen anderen Wert (meist schon das Zeichen selbst).
Achtung: Dieser Befehl gibt das Eingaberegister auf der KEY nicht Frei!

>D0.B $00 = keine Taste gedrückt

```
START:
 MOVE.W #0,D0
 JSR COLINIT
LOOP:
 MOVE.W #255,D0
 JSR @RND              * X-Position
 MOVE.W D0,D1
 MOVE.W #255,D0
 JSR @RND              * Y-Position
 MOVE.W D0,D2
 MOVE.W #255,D0
 JSR @RND              * Farbe
 JSR SETFDOT           * Punkt setzen
 JSR CSTS              * Testen ob Taste gedrückt
 TST.B D0
 BEQ LOOP              * Nein dann wiederholen
RTS
```

CO
Setzen eines Zeichen.

Beim dieser Art von Zeichenausgabe wird das Zeichen in einer bestimmten Spalte und Zeile dargestellt. Die Farbe, Spalte und Zeile wird durch den SETCURXY Befehl bestimmt. Wenn nun ein Zeichen ausgegeben wird, wird automatisch die Spalte (X-Position) um Eins erhöht. Dadurch kann man einen Text zusammenhängend ausgeben. Wenn der rechte Rand erreicht ist, dann wird das nächste Zeichen in der Zeile darunter am Zeilenanfang gesetzt. Wenn der Text nach unten herausgeht, dann wird er am Oberen Rand weitergeführt. Es ergibt sich also ein geschlossener Bildschirm mit 32 * 21 Zeichen. Steuerbefehle werden unten erklärt.

DO Zeichen

Steuerzeichen: ^ = CTRL

^H ($08) Backspace Cursor links	Ein Zeichen nach links. Wenn der Cursor dabei am linken Rand angekommen ist, dann wird er um eine Zeile nach oben, an den rechten Rand versetzt.
^I ($09) Hor. Tab Cursor rechts	Ein Zeichen nach rechts. Wenn der Cursor dabei am rechten Rand angekommen ist, dann wird er um eine Zeile nach unten, an den linken Rand versetzt.
^J ($0a) Line Feed Cursor abwärts	Der Cursor wird um eine Zeile nach unten versetzt. Wenn er dabei am unteren Rand ankommt, wird er an den oberen Rand versetzt.
^K ($0b) Vert. Tab Cursor aufwärts	Der Cursor wird um eine Zeile nach oben versetzt. Wenn er dabei am oberen Rand ankommt, wird er an den unteren Rand versetzt.
^L ($0c) Cursor rechts	Wie ^I ($09)
^M ($0d) Return Carriage Return	Der Cursor wird um eine Zeile nach unten, an den linken Rand versetzt.
^V ($16) Line Feed Cursor abwärts	Wie ^J ($0a)
^Z ($1a) Clear Bild löschen	Der Bildschirm wird gelöscht und der Cursor wird an den oberen linken Rand gesetzt.
^^ ($1e) Home Cursor Home	Der Cursor wird in an den oberen linken Rand versetzt.

CLRSIGN
Löschen eines Zeichen.

Es wird ein Zeichen an der Aktuellen CURX und CURY Position gelöscht.
Das Zeichen wird mit der Hintergrundfarbe überschrieben.

CURX X-Position des Zeichen
CURY Y-Position des Zeichen

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0          * Farbe Rot
 MOVE.W #0,D1            * X-Position
 MOVE.W #20,D2           * Y-Position
 MOVE.W #10,D3           * Blinkfrequenz = 20ms * 2 * 10 = 0,4 sec = 2,5 Hz
 JSR SETCURXY
 MOVE.W #'A',D0
 JSR CO                  * Zeichen ausgeben auf Bildschirm
 MOVE.L #20,D0           * 2 Sekunden Warten
 JSR DELAY
 MOVE.B #$03,D0          * Farbe Rot
 MOVE.W #0,D1            * X-Position
 MOVE.W #20,D2           * Y-Position
 MOVE.W #10,D3           * Blinkfrequenz = 20ms * 2 * 10 = 0,4 sec = 2,5 Hz
 JSR SETCURXY
 JSR CLRSIGN             * Zeichen wieder löschen
RTS
```

READ
Lesen eines Textes von Tastatur.

Dieser Befehl ist ähnlich dem @READ aus dem Grundprogramm. Es wird ein Text über die Tastatur eingelesen und gleichzeitig, optional in einem Rahmen, angezeigt. Im Register D0 wird die Textfarbe, in D1 die X-Position (0-31 wie bei SETCURXY) und in D2 die Y-Position (0-20 wie bei SETCURXY) übergeben. In D3 muss die Anzahl der maximal einzugebenen Zeichen stehen und D4 besagt ob ein Rahmen gezeichnet werden soll oder nicht (D4 = $FF mit Rahmen $00 ohne Rahmen). Weiterhin muss in A0 die Adresse stehen, wohin der eingegebene Text gespeichert werden soll.
Als Ausgabe erhält man dann in D4.W die Anzahl der eingegebenen Zeichen, und in D5.B das letzte Zeichen. Ausserdem steht in A0.L der Zeiger auf das Ende des Textes.

D0	Farbe des Textes	>D4.W	Anzahl der Zeichen
D1	X-Position [0-31]	>D5.B	Letztes Zeichen
D2	Y-Position [0-20]	>A0.L	Zeiger auf Textende
D3	Anzahl der Zeichen		
D4	Rahmen $FF = ja $00 = nein		
A0	Zeiger auf Textspeicher		

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$30,D0          * Farbe
 MOVE.W #0,D1            * X-Position
 MOVE.W #0,D2            * Y-Position
 MOVE.W #20,D3           * 20 Zeichen
 MOVE.B #$FF,D4          * mit Rahmen
 LEA TEXT,A0             * Speicher
 JSR READ
RTS

TEXT: DS.B 20            * 20 Zeichen freihalten
```

WRITE
Schreiben eines Textes auf Bildschirm.

Dieser Befehl ist ähnlich dem @WRITE aus dem Grundprogramm. Es wird ein Text auf dem Bildschirm ausgegeben. In D0 steht die Textfarbe und in D1 wird die X-Position (0-31 wie bei SETCURXY) und in D2 die Y-Position (0-20 wie bei SETCURXY) übergeben.
Weiterhin muss in A0 die Adresse stehen wo der Text gespeichert ist.

D0	Farbe des Textes
D1	X-Position [0-31]
D2	Y-Position [0-20]
A0	Zeiger auf Textspeicher

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$AA,D0          * Farbe
 MOVE.W #0,D1            * X-Position
 MOVE.W #0,D2            * Y-Position
 LEA TEXT,A0             * Speicher
 JSR WRITE
RTS

TEXT: DC.B 'Textspeicher',0
DS 0
```

CHANGE
Vertauschen von Bildschirminhalten.

Mit diesem Befehl kann man den Inhalt von 2 Bildseiten vertauschen.
Es wird ein Rechteck aus der Bildschirmseite D5 mit demselben
Rechteck aus der Bildschirmseite D6 vertauscht. Die Koordinaten
für das Rechteck werden in D1 D2 D3 D4 bestimmt. D1 und D2 geben die
Startkoordinaten und D3 und D4 die Endkoordinaten an.
Dadurch ist es möglich, wie durch ein Fenster, auf eine andere Bildseite
zu schauen. Man kann somit feste Texte in eine Grafik ein und ausblenden.

ACHTUNG dieser Befehl funktioniert nur mit dem grossen Speicheraufbau!

D1	X-Position Start
D2	Y-Position Start
D3	X-Position Ende
D4	Y-Position Ende
D5	Seite 1
D6	Seite 2

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0
 MOVE.B #0,D1            * Seite 0
 JSR CLPGCOL
 MOVE.B #$30,D0
 MOVE.B #1,D1            * Seite 1
 JSR CLPGCOL
 MOVE.L #10,D0
 JSR DELAY
 MOVE.W #60,D1
 MOVE.W #60,D2
 MOVE.W #180,D3
 MOVE.W #180,D4          * Ausschnitt 60,60 bis 180,180
 MOVE.W #0,D5
 MOVE.W #1,D6            * von Seite 0 nach Seite 1
 JSR CHANGE
 MOVE.L #10,D0
 JSR DELAY
 JSR CHANGE              * Wieder zurück tauschen
RTS
```

INVERT

Invertieren des Bildschirminhalt.

Mit diesem Befehl kann man den Inhalt einer Bildseite invertieren. Es wird ein Rechteck aus der aktuellen Bidschirmseite mit den Koordinaten D1 D2 bis D3 D4 invertiert. Dabei wird jedes Bit einzeln invertiert, das heisst, aus einer 1 wird ein 0 und aus einer 0 wird eine 1. Dadurch wird z.B. ein blaues Bild in ein Gelbes umgefärbt. Weiterhin wird in D0 der Mode übergeben. Der Mode bestimmt ob die Farbe Schwarz ($00) mit invertiert werden soll oder nicht.
Mode: $00 = Schwarz ($00) nicht invertieren.
$01 = Schwarz ($00) mit invertieren.

D0	Mode
D1	X-Position Start
D2	Y-Position Start
D3	X-Position Ende
D4	Y-Position Ende

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$30,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO
 MOVE.B #$00,D0          * Schwarz ignorieren
 MOVE.W #0,D1
 MOVE.W #0,D2
 MOVE.W #128,D3
 MOVE.W #128,D4          * Ausschnitt 0,0 bis 128,128
 JSR INVERT
 MOVE.B #$01,D0          * Schwarz mit invertieren
 MOVE.W #255,D1
 MOVE.W #255,D2
 MOVE.W #128,D3
 MOVE.W #128,D4          * Ausschnitt 128,128 bis 255,255
 JSR INVERT
RTS
```

COPY

Copieren von Bildschirmteilen.

Mit diesem Befehl kann man einen Teil einer Bildseite an eine andere Stelle copieren. Es wird das Rechteck mit den Koordinaten D1 D2 bis D3 D4 nach D5 D6 hin copiert.
Weiterhin wird in D0 der Mode übergeben. Der Mode bestimmt ob die Farbe Schwarz ($00) mit copiert werden soll oder nicht. Dieses ist wichtig, wenn man nicht immer den Rahmen mit sehen möchte.
Mode: $00 = Schwarz ($00) nicht copieren.
$01 = Schwarz ($00) mit copieren.

D0	Mode
D1	X-Position Start
D2	Y-Position Start
D3	X-Position Ende
D4	Y-Position Ende
D5	X-Position Ziel
D6	Y-Position Ziel

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$30,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO
 MOVE.B #$00,D0          * Schwarz ignorieren
 MOVE.W #0,D1
 MOVE.W #0,D2
 MOVE.W #80,D3
 MOVE.W #80,D4           * Ausschnitt 0,0 bis 80,80
 MOVE.W #10,D5
 MOVE.W #100,D6          * Nach 10,100 copieren
 JSR COPY
RTS
```

DELTAXY
Umrechnen von Koordinaten in delta Werte.

Der DELTAXY Befehl wird benutzt um aus 2 absoluten Koordinaten die minimal und die delta Werte zu erhalten. Als Eingabe erhält der Befehl die Anfangs- und Endposition einer Figur (z.B. Rechteck) in D1 D2 und D3 D4. Daraus errechnet sich der Computer die kleinsten Koordinaten (die am dichtesten an 0,0 liegen) und die Differenz (delta) zu den Endkoordinaten. Z.B. würde die Eingabe D1 = 10 D2 = 80 D3 = 40 D4 = 10 als Ergebnis D1 = 10 D2 = 10 D3 = 30 D4 = 70 ergeben. (kleinstes X u. Y) (delta X u. Y)

D1	X-Position Start	>D1	Kleinste X-Koordinate
D2	Y-Position Start	>D2	Kleinste Y-Koordinate
D3	X-Position Ende	>D3	Delta X
D4	Y-Position Ende	>D4	Delta Y

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #120,D1          * Als Start annehemen
 MOVE.W #200,D2
 MOVE.W #200,D3          * Als Ende annehmen
 MOVE.W #120,D4
 JSR DELTAXY
 JSR AMOVE               * Startposition
 MOVE.B #$30,D0
 MOVE.W D3,D1
 MOVE.W D4,D2
 MOVE.W #0,D3            * Setmode fest setzen
 JSR RDRAW               * Endposition
RTS
```

MAXOVER
Testen der Koordinaten gegen die Maximale.

Der MAXOVER Befehl wird benutzt, um festzustellen, ob die Koordinaten D1 und D2 sich innerhalb des Bildschirms befinden. Hiernach werden dann auch die Flags gesetzt. Da bei diesem Befehl keine Register verändert werden, aber die Flags, funktioiert dieser Befehl nicht mit dem TRAP #6.

D1	X-Position	>CCR	Z = 1 dann ausserhalb
D2	Y-Position		Z = 0 dann innerhalb

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #260,D1          * Als zu gross annehmen
 MOVE.W #200,D2
 JSR MAXOVER
 BEQ ENDE
 MOVE.B #$03,D0
 JSR SETFDOT
ENDE:
RTS
```

MAXDELTA

X+XDelta und Y+YDelta werden gegen die Maximale getestet und gekürzt.

Der MAXDELTA Befehl wird benutzt um festzustellen, ob die Koordinaten D1 + D3 und D2 + D4 innerhalb des Bildschirms befinden. Sind diese zu gross oder zu klein, dann werden sie auf die Maximale abgeschnitten. Es wird also das Rechteck so abgeschnitten, dass es auf den Bildschrim passt.

D1	X-Position	>D1	X-Position
D2	Y-Position	>D2	Y-Position
D3	Delta X	>D3	Delta X
D4	Delta Y	>D4	Delta Y

```
START:
 MOVE.W #-10,D1          * Als als zu klein annehmen
 MOVE.W #-10,D2
 MOVE.W #400,D3          * Delta Werte auch zu gross
 MOVE.W #400,D4
 JSR MAXDELTA            * Dann abschneiden lassen
RTS
```

POLYLINE

Zeichnen mehrerer Linien aus einer Tabelle.

Um eine grosse Anzahl Linien zu Zeichnen benutzt man den POLYLINE Befehl. Hierbei wird nur mit A1 auf eine Tabelle gezeigt, die die gesammten X und Y Koordinaten gespeichert hat. Weiterhin wird in D0 die Farbe und in D3 die Anzahl der Koordinaten übergeben.
Die Koordinaten sind für X und Y immer in Wortgrösse angegeben.

D0	Farbe
D3	Anzahl der Koordinaten
A1	Zeiger auf die Koordinatentabelle

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$0C,D0          * Grün
 MOVE.W #9,D3            * 9 Koordinaten
 LEA TAB,A1
 JSR POLYLINE
RTS

TAB:
 DC.W 10,10 ,10,120 ,120,240 ,240,120 ,10,120 ,240,10 ,10,10 ,240,120 ,240,10
```

POLYGON

Zeichnen mehrerer Linien aus einer Tabelle in sich geschlossen.

Dieser Befehl ist dem POLYLINE gleich, zieht allerdings ein von der letzten Koordinate eine Linie zur ersten Koordinate. Es ergibt sich also ein geschlosserner Körper.
Die Koordinaten sind für X und Y immer in Wortgrösse angegeben.

D0	Farbe
D3	Anzahl der Koordinaten
A1	Zeiger auf die Koordinatentabelle

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$0C,D0          * Grün
 MOVE.W #4,D3            * 4 Koordinaten
 LEA TAB,A1
 JSR POLYGON             * Geschlossener Körper
RTS

TAB:
 DC.W 10,10 ,10,200 ,200,200 ,200,10    * Eindeutig nicht geschlossen
```

SETWINDOW

Setzen eines Window auf den Bildschirm.

Der SETWINDOW Befehl setzt ein Window beliebiger Grösse auf den Bildschirm. Der Speicher des Window wird im Benutzerspeicher definiert. Dabei wird der Speicher in Bytewerten, die den Farben entsprechen, aufwärts Zählend organisiert. Das erste Byte entspricht dem Punkt unten links in dem Window, das zweite Byte dem rechts daneben, und das letzte Byte dem Punkt oben rechts. Es können alle verfügbaren Farben in einem Window benutzt werden. Die X und Y Position wird in D1 und D2 bestimmt, die Grösse in X und Y Richtung wird in D3 und D4 bestimmt, die Adresse des Windowspeicher wird im Register A1 und die Adresse des Zwischenspeicher für den alten Bildschirmausschnitt wird in A2 übergeben. Zusätzlich wird im Register D0 der Mode übergeben. Dieser bestimmt ob das Window vor dem Bild ($00 d.h. Schwarz wird ignoriert), fest ($01 d.h. Schwarz wird mitgezeichnet) oder hinter dem Bild dargestellt wird ($02 nur wo Schwarz ist)

```
D0      Mode  $00 = Schwarz ignorieren  $01 = Schwarz mit zeichnen
        $02 = In Hintergrund zeichnen
D1      X-Position
D2      Y-Position
D3      X-Grösse
D4      Y-Grösse
A1      Zeiger auf Windowspeicher
A2      Zeiger auf Zwischenspeicher
```

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$00,D0          * Vor dem Bild zeichnen
 MOVE.W #128,D1
 MOVE.W #128,D2
 MOVE.W #5,D3            * 5 * 5 Punkte gross
 MOVE.W #5,D4
 LEA TAB,A1
 LEA ZWISCHEN,A2
 JSR SETWINDOW
RTS

TAB:
DC.B $03,$00,$C0,$00,$0C
DC.B $00,$03,$C0,$0C,$00
DC.B $30,$30,$FF,$30,$30
DC.B $00,$0C,$C0,$03,$00
DC.B $0C,$00,$C0,$00,$03
DS 0
ZWISCHEN: DS.B 5*5      * für 25 Byte frei halten
```

CLRWINDOW
Löschen eines Window auf den Bildschirm.

Der CLRWINDOW Befehl löscht ein Window beliebiger Grösse auf den Bildschirm. Die X und Y Position wird in D1 und D2 bestimmt, die Grösse in X und Y Richtung wird in D3 und D4 bestimmt und die Adresse des Zwischenspeicher für den alten Bildausschnitt wird in A2 übergeben. Hierbei wird also nur der alte Bildschirminhalt wieder aus dem Zwischenspeicher in das Bildram geschrieben.
Beim Löschen von mehreren Windows sollte man darauf aufpassen, dass man zuerst das letzte Window wieder löscht, da es sonst zu Überlappungen einiger Windows kommen kann.

D1	X-Position
D2	Y-Position
D3	X-Grösse
D4	Y-Grösse
A2	Zeiger auf Zwischenspeicher

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$00,D0          * Vor dem Bild zeichnen
 MOVE.W #128,D1
 MOVE.W #128,D2
 MOVE.W #5,D3            * 5 * 5 Punkte gross
 MOVE.W #5,D4
 LEA TAB,A1
 LEA ZWISCHEN,A2
 JSR SETWINDOW
 MOVE.L #20,D0
 JSR DELAY
 JSR CLRWINDOW           * Die alten Werte sind noch gespeichert
RTS                      * in D1 - D4 und A2

TAB:
DC.B $03,$00,$C0,$00,$0C
DC.B $00,$03,$C0,$0C,$00
DC.B $30,$30,$FF,$30,$30
DC.B $00,$0C,$C0,$03,$00
DC.B $0C,$00,$C0,$00,$03
DS 0
ZWISCHEN: DS.B 5*5       * für 25 Byte frei halten
```

GETWINDOW
Lesen eines Window vom Bildschirm.

Der GETWINDOW Befehl liest ein Window beliebiger Grösse aus dem Bildschirmspeicher. Die X und Y Position wird in D1 und D2 bestimmt, die Grösse in X und Y Richtung wird in D3 und D4 bestimmt und die Adresse des Windowspeicher, wo das Bild abgelegt wird, wird in A1 übergeben. Hierbei wird also der Bildschirminhalt ausgelesen und in dem Windowspeicher geschrieben. Dieses ist sinnvoll, wenn man später diesen Bildabschnitt noch einmal wieder haben möchte, oder wenn der Bildteil sehr oft auf den Bildschirm gesetzt werden soll.

D1	X-Position
D2	Y-Position
D3	X-Grösse
D4	Y-Grösse
A1	Zeiger auf Windowspeicher

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO
 MOVE.W #128,D1          * Position 128,128
 MOVE.W #128,D2
 MOVE.W #10,D3           * Grösse 10 * 10
 MOVE.W #10,D4
 LEA WINDOW1,A1          * Windowspeicher 1
 JSR GETWINDOW
RTS

WINDOW1: DS.B 10*10      * für 100 Byte frei halten
```

DEFWINDOW
Definieren eines Window.

Der DETWINDOW Befehl definiert ein Window beliebiger Grösse. Er legt den Windowspeicher und den Zwischenspeicher an und definiert dabei gleich die Umrechnungsformel für die Punktesetzroutine im Windowspeicher (SETWDOT GETWDOT CALCWDOT).
In D0 wird der Mode gesetzt. Dieser bestimmt ob das Window vor dem Bild ($00 Schwarz ignorieren), fest ($01 Schwarz mit zeichnen) oder hinter dem Bild ($02 nur auf Schwarz zeichnen) dargestellt wird. In D3 und D4 wird die Grösse des Window in X und Y Richtung definiert. Weiterhin wird in Register A1 der Windowspeicher und in A2 der Zwischenspeicher organisiert. Die Speicher werden nicht gelöscht!

D0	Mode $00 = Schwarz ignorieren $01 = Schwarz mit zeichnen
	$02 = nur auf Schwarz zeichnen
D3	X-Grösse
D4	Y-Grösse
A1	Zeiger auf Windowspeicher
A2	Zeiger auf Zwischenspeicher

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0
 MOVE.W #5,D3
 MOVE.W #5,D4
 LEA TAB,A1
 LEA ZWISCHEN,A2
 JSR DEFWINDOW
 MOVE.W #128,D1          * Position 128,128
 MOVE.W #128,D2
 JSR WINDOW              * Darstellen
RTS

TAB:
DC.B $03,$00,$C0,$00,$0C
DC.B $00,$03,$C0,$0C,$00
DC.B $30,$30,$FF,$30,$30
DC.B $00,$0C,$C0,$03,$00
DC.B $0C,$00,$C0,$00,$03
DS 0
ZWISCHEN: DS.B 5*5       * für 25 Byte frei halten
```

WINDOW
Darstellen eines Window.

Der WINDOW Befehl stellt das durch DEFWINDOW definierte Window auf dem Bildschirm dar. Hierbei werden nur die X und Y Koordinaten des Window in D1 und D2 übergeben. Wenn das Window vorher schon einmal dargestellt wurde, dann wird zuvor das alte Window gelöscht bevor das neue dargestellt wird. Dadurch können bewegte Grafiken extrem einfach hergestellt werden.

D1 X-Position
D2 Y-Position

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0
 MOVE.W #5,D3
 MOVE.W #5,D4
 LEA TAB,A1
 LEA ZWISCHEN,A2
 JSR DEFWINDOW
 MOVE.W #0,D1            * Start Position 0,0
 MOVE.W #0,D2
LOOP:
 JSR SYNC                * Sonst zu schnell
 BEQ.S LOOP
 JSR WINDOW              * Darstellen
 ADDQ.W #1,D1
 ADDQ.W #1,D2
 CMP.W #250,D1           * Maximal bis 250,250
 BNE.S LOOP
RTS

TAB:
DC.B $03,$00,$C0,$00,$0C
DC.B $00,$03,$C0,$0C,$00
DC.B $30,$30,$FF,$30,$30
DC.B $00,$0C,$C0,$03,$00
DC.B $0C,$00,$C0,$00,$03
DS 0
ZWISCHEN: DS.B 5*5      * für 25 Byte frei halten
```

WINDOWOFF
Löschen des alten Window.

Der WINDOWOFF Befehl löscht das durch WINDOW dargestellte Window.
Hier werden keine Register angegeben, da die alte Window Position
noch gespeichert ist.

Keine Register

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0
 MOVE.W #5,D3
 MOVE.W #5,D4
 LEA TAB,A1
 LEA ZWISCHEN,A2
 JSR DEFWINDOW
 MOVE.W #0,D1            * Start Position 0,0
 MOVE.W #0,D2
LOOP:
 JSR SYNC                * Sonst zu schnell
 BEQ.S LOOP
 JSR WINDOW              * Darstellen
 ADDQ.W #1,D1
 ADDQ.W #1,D2
 CMP.W #150,D1           * Maximal bis 150,150
 BNE.S LOOP
 JSR WINDOWOFF
RTS

TAB:
DC.B $03,$00,$C0,$00,$0C
DC.B $00,$03,$C0,$0C,$00
DC.B $30,$30,$FF,$30,$30
DC.B $00,$0C,$C0,$03,$00
DC.B $0C,$00,$C0,$00,$03
DS 0
ZWISCHEN: DS.B 5*5      * für 25 Byte frei halten
```

SETWDOT
Punkt setzen im Windowspeicher.

Mit dem SETWDOT wird ein Punkt im Windowspeicher gesetzt. Dadurch lassen sich die Windows schon vor dem Darstellen verändern.
In D0 steht die Farbe des Punktes und in D1 und D2 die X und Y Koordinaten. Als Ausgabe erhält man in A0.L die Adresse des Punktes innerhalb des Windowspeicher.

D0	Farbe	>A0.L	Adresse des Punktes
D1	X-Position		
D2	Y-Position		

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0
 MOVE.W #5,D3
 MOVE.W #5,D4
 LEA TAB,A1
 LEA ZWISCHEN,A2
 JSR DEFWINDOW
 MOVE.B #$FF,D0
 MOVE.W #3,D1
 MOVE.W #3,D2
 JSR SETWDOT            * Punkt setzen an 3,3
 MOVE.W #128,D1         * Start Position 128,128
 MOVE.W #128,D2
 JSR WINDOW             * Darstellen
RTS

TAB:
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DS 0
ZWISCHEN: DS.B 5*5      * für 25 Byte frei halten
```

GETWDOT
Punkt lesen aus Windowspeicher.

Mit dem GETWDOT wird ein Punkt aus dem Windowspeicher gelesen. Dadurch lassen sich die Windows direkt auslesen und verändern. Eine Anwendung ist z.B. ob das Window mit einem Bildpunkt einer bestimmten Farbe zusammengestossen ist. Damit kann man dann sehr einfach bewegte Spiele schreiben.
Als Eingabe wird in D1 und D2 die X und Y Position im Window angegeben. Als Ausgabe erhält man dann in D0 die Farbe des Punktes und in A0.L die Adresse des Punktes.

D1	X-Position	>D0.B	Farbe
D2	Y-Position	>A0.L	Adresse des Punktes

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0
 MOVE.W #5,D3
 MOVE.W #5,D4
 LEA TAB,A1
 LEA ZWISCHEN,A2
 JSR DEFWINDOW
 MOVE.W #3,D1
 MOVE.W #3,D2
 JSR GETWDOT            * Punkt lesen an 3,3
 CMP.B #$FF,D0
 BEQ ENDE               * Wenn Punkt 3,3 = $FF dann Ende
 MOVE.W #128,D1         * Start Position 128,128
 MOVE.W #128,D2
 JSR WINDOW             * Darstellen
ENDE:
RTS

TAB:
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$FF,$00
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DS 0
ZWISCHEN: DS.B 5*5      * für 25 Byte frei halten
```

CALCWDOT

Punktadresse im Windowspeicher berechnen.

Mit dem CALCWDOT wird ein Punkt im Windowspeicher berechnet. Dadurch lassen sich die Windows direkt auslesen und verändern. Eine Anwendung ist z.B. das Window schnell zu Invertieren.
Als Eingabe wird in D1 und D2 die X und Y Position im Window angegeben. Als Ausgabe erhält man dann in A0.L die Adresse des Punktes.

D1	X-Position	>A0.L	Adresse des Punktes
D2	Y-Position		

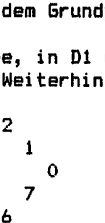
```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0
 MOVE.W #5,D3
 MOVE.W #5,D4
 LEA TAB,A1
 LEA ZWISCHEN,A2
 JSR DEFWINDOW
 MOVE.W #3,D1
 MOVE.W #3,D2
 JSR CALCWDOT          * Punkt berechnen an 3,3
 EORI.B #$FF,(A0)      * Punkt invertieren
 MOVE.W #128,D1        * Start Position 128,128
 MOVE.W #128,D2
 JSR WINDOW            * Darstellen
RTS

TAB:
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DC.B $00,$00,$00,$00,$00
DS 0
ZWISCHEN: DS.B 5*5     * für 25 Byte frei halten
```

SETFIGUR
Figur auf Bildschirm darstellen.

Der SETFIGUR Befehl setzt eine Figur beliebiger grösse auf den Bildschirm. Dieser Befehl zeichnet eine Figur aus einer Vektortabelle. Ist also ähnlich dem @FIGUR aus dem Grundprogramm. Jedoch wird hierbei die Figur fest dargestellt.
Als Eingabe wird in D0 die Farbe, in D1 und D2 die X und Y Position und in D3 die Grösse übergeben. Weiterhin wird in A2 die Adresse der Figurtabelle übergeben.

Richtungscode:
8 Schreibstift heben
9 Schreibstift senken
10 Ende der Tabelle



D0	Farbe der Figur
D1	X-Position
D2	Y-Position
D3	Grösse
A2	Zeiger auf Tabelle

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0          * Farbe ROT
 MOVE.W #128,D1          * X = 128
 MOVE.W #128,D2          * Y = 128
 MOVE.W #10,D3           * Grösse = 10
 LEA TAB,A2
 JSR SETFIGUR
RTS

TAB:
DC.B 0,0,0,0,1,2,2,4,4,0,0,2,2,3,4,4
DC.B 8,6,9,6,4,2,0,8,2,9,4,4,5,6,6,6,6,7,10
```

CLRFIGUR
Figur vom Bildschirm löschen.

Hierbei wird die durch PUTFIGUR gesetzte Figur vom Bildschirm gelöscht.
Es werden keine Parameter übergeben, da diese noch gespeichert sind.

Keine Register

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0          * ROT
 MOVE.W #128,D1          * X = 128
 MOVE.W #128,D2          * Y = 128
 MOVE.W #10,D3           * Grösse
 LEA ZWISCHEN,A1
 LEA TAB,A0
 JSR PUTFIGUR            * Beim ersten setzen PUTFIGUR
 MOVE.L #20,D0
 JSR DELAY
 JSR CLRFIGUR
RTS

TAB:
DC.B 0,0,0,0,1,2,2,4,4,0,0,2,2,3,4,4
DC.B 8,6,9,6,4,2,0,8,2,9,4,4,5,6,6,6,6,7,10
DS 0
ZWISCHEN:
DS.B 400        * 400 Byte freihalten (ca. Grösse * Anzahl Vektoren)
```

FIGUR
Figur auf Bildschirm darstellen.

Der FIGUR Befehl setzt eine Figur beliebiger grösse auf den Bildschirm. Dieser Befehl zeichnet eine Figur aus einer Vektortabelle. Ist also ähnlich dem @FIGUR aus dem Grundprogramm. Die Figur lässt sich frei auf dem Bildschirm bewegen.
Als Eingabe wird in D0 die Farbe, in D1 und D2 die X und Y Position und in D3 die Grösse übergeben. Weiterhin wird in A0 die Adresse der Figurtabelle übergeben und A1 ein Zeiger auf einen Zwischenspeicher. Dieses ist Wichtig, da der alte Bildinhalt zuerst ausgelesen wird. ACHTUNG beim ersten Darstellen muss PUTFIGUR benutzt werden!

Richtungscode:

```
8 Schreibstift heben          3  2  1
9 Schreibstift senken       4        0
10 Ende der Tabelle           5  7  6
```

D0	Farbe der Figur
D1	X-Position
D2	Y-Position
D3	Grösse
A0	Zeiger auf Tabelle
A1	Zeiger auf Zwischenspeicher

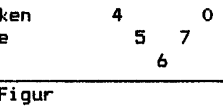
```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0
 MOVE.W #10,D1
 MOVE.W #10,D2
 MOVE.W #2,D3
 LEA TAB,A0              * Zeiger auf Tabelle
 LEA ZWISCHEN,A1         * Zeiger auf Zwischenspeicher
 JSR PUTFIGUR            * Erstes Darstellen
 MOVE.W #200,D6
LOOP:
 JSR SYNC
 BEQ.S LOOP
 ADDQ.W #1,D1
 ADDQ.W #1,D2
 JSR FIGUR
 DBRA D6,LOOP
RTS

TAB:
DC.B 0,0,0,0,1,2,2,4,4,0,0,2,2,3,4,4
DC.B 8,6,9,6,4,2,0,8,2,9,4,4,5,6,6,6,6,7,10
DS 0
ZWISCHEN: DS.B 400      * Gross genug
```

PUTFIGUR
Figur auf Bildschirm darstellen.

Der PUTFIGUR Befehl setzt eine Figur beliebiger grösse auf den Bildschirm. Dieser Befehl zeichnet eine Figur aus einer Vektortabelle. Ist also ähnlich dem @FIGUR aus dem Grundprogramm. Dieser Befehl muss als erstes vor jedem neuen FIGUR Befehl stehen.
Als Eingabe wird in D0 die Farbe, in D1 und D2 die X und Y Position und in D3 die Grösse übergeben. Weiterhin wird in A0 die Adresse der Figurtabelle übergeben und A1 ein Zeiger auf einen Zwischenspeicher. Dieses ist Wichtig, da der alte Bildinhalt zuerst ausgelesen wird. ACHTUNG beim ersten Darstellen muss PUTFIGUR benutzt werden!

Richtungscode:
8 Schreibstift heben
9 Schreibstift senken
10 Ende der Tabelle



D0	Farbe der Figur
D1	X-Position
D2	Y-Position
D3	Grösse
A0	Zeiger auf Tabelle
A1	Zeiger auf Zwischenspeicher

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0
 MOVE.W #10,D1
 MOVE.W #10,D2
 MOVE.W #2,D3
 LEA TAB,A0              * Zeiger auf Tabelle
 LEA ZWISCHEN,A1         * Zeiger auf Zwischenspeicher
 JSR PUTFIGUR            * Erstes Darstellen
 MOVE.W #200,D6
LOOP:
 JSR SYNC
 BEQ.S LOOP
 ADDQ.W #1,D1
 ADDQ.W #1,D2
 JSR FIGUR
 DBRA D6,LOOP
RTS

TAB:
DC.B 0,0,0,0,1,2,2,4,4,0,0,2,2,3,4,4
DC.B 8,6,9,6,4,2,0,8,2,9,4,4,5,6,6,6,6,7,10
DS 0
ZWISCHEN: DS.B 400      * Gross genug
```

ZOOM
Ausschnittvergrösserung.

Der ZOOM Befehl vergrössert Bildausschnitte bis zu einer beliebigen Grösse. Als Koordinaten wird in D0 der Vergrösserungsfactor, in D1 und D2 die Anfangs und in D3 und D4 die Endkoordinaten eingegeben.
Die Vergrösserung bezieht sich immer auf die Ecke die am weitesten unten links liegt. Eine Vergrösserung findet also immer nach oben rechts statt.

D0	Vergrösserungsfactor
D1	X-Anfangs Position
D2	Y-Anfangs Position
D3	X-End Position
D4	Y-End Position

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.B #$03,D0
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Linie zeichnen
 MOVE.W #5,D0
 MOVE.W #10,D1           * Rechteck von 10,10 bis 20,20
 MOVE.W #10,D2           * Um Factor 5 Vergrössern
 MOVE.W #20,D3
 MOVE.W #20,D4
 JSR ZOOM
RTS
```

SETA4
Setzten des A4 Register auf Hilfsspeicher.

In diesem Programm wird ein bestimmter Speicherbereich für Interne Register benutzt. Dieses Register muss bei ansprechen der Befehle einen bestimmten Wert besitzen, da sonst die Grafik nicht richtig oder gar nicht funtioiert. Bei TRAP #6 Einsprüngen ist dieses nicht Notwendig, da das Register A4 automatisch neu gesetzt wird.
Wenn sonst also das A4 Register benutzt wird, MUSS vor dem Aufruf eines Befehls dieser Befehl gegeben werden.
Weiterhin wird der Befehl dafür verwendet, um bestimmte Interne Register direkt zu manipulieren.

>A4.L Adresse des Hilfsspeicher

```
START:
 JSR SETA4              * Setzen des A4 Register
RTS
```

SETXYXY

Setzten von X und Y Koordinaten im Hilfsspeicher.

Mit diesem Befehl werden die Register X1 Y1 X2 und Y2 belegt.
Dieses ist für den LINE Befehl notwendig. Zusätzlich wird damit
die Koordinaten für die relativen Zeichenbefehle verändert.
D1 wird nach X1, D2 wird nach Y1, D3 wird nach X2, D4 wird nach Y2
transferiert. X1 und Y1 sind gleichzeitig die Koordinaten für die
Relativen Zeichenbefehle.

D1	X-Position 1
D2	Y-Position 1
D3	X-Position 2
D4	Y-Position 2

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #10,D1          * X1 = 10  Y1 = 10  X2 = 200  Y2 = 200
 MOVE.W #10,D2
 MOVE.W #200,D3
 MOVE.W #200,D4
 JSR SETXYXY
 MOVE.W #$03,D0         * Farbe = Rot
 JSR SETCOLOR
 JSR LINE               * Linie von X1 Y1 nach X2 Y2
RTS
```

GETXYXY

Lesen von X und Y Koordinaten aus dem Hilfsspeicher.

Mit diesem Befehl werden die Register X1 Y1 X2 und Y2 ausgelesen.
Damit kann man dann z.B. die Aktuellen Koordinaten auslesen.
X1 wird nach D1, Y1 wird nach D2, X2 wird nach D3, Y2 wird nach D4
transferiert. X1 und Y1 sind gleichzeitig die Koordinaten für die
Relativen Zeichenbefehle.

>D1	X-Position 1
>D2	Y-Position 1
>D3	X-Position 2
>D4	Y-Position 2

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #10,D1          * X1 = 10  Y1 = 10  X2 = 200  Y2 = 200
 MOVE.W #10,D2
 MOVE.W #200,D3
 MOVE.W #200,D4
 JSR SETXYXY
 MOVE.W #$03,D0         * Farbe = Rot
 JSR SETCOLOR
 JSR LINE               * Linie von X1 Y1 nach X2 Y2
 JSR GETXYXY            * Und wieder auslesen
                        * D1 und D2 enthält jetzt die letzten Koordinaten
                        * (200,200)
 MOVE.W #200,D3         * Endposition neu Laden
 MOVE.W #10,D4
 JSR SETXYXY
 JSR LINE
RTS
```

*SETCOLOR
Setzen der Punktefarbe.

Mit diesem Befehl werden das Farbregister verändert. Damit ist es möglich die Zeichenfarbe zu verändern.
In D0 steht die neue Farbe.

D0 Farbe

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #10,D1          * X1 = 10  Y1 = 10  X2 = 200  Y2 = 200
 MOVE.W #10,D2
 MOVE.W #200,D3
 MOVE.W #200,D4
 JSR SETXYXY
 MOVE.W #$03,D0         * Farbe = Rot
 JSR SETCOLOR
 JSR LINE               * Linie von X1 Y1 nach X2 Y2
RTS
```

GETCOLOR
Lesen der Aktuellen Punktefarbe.

Mit diesem Befehl wird das Farbregister nach D0 gelesen. Damit ist es möglich die Zeichenfarbe zu speichern, um sie später wieder zu gebrauchen.

>D0.B Farbe

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #10,D1          * X1 = 10  Y1 = 10  X2 = 200  Y2 = 200
 MOVE.W #10,D2
 MOVE.W #200,D3
 MOVE.W #200,D4
 JSR SETXYXY
 MOVE.W #$03,D0         * Farbe = Rot
 JSR SETCOLOR
 JSR LINE               * Linie von X1 Y1 nach X2 Y2
 JSR GETXYXY            * Und wieder auslesen
                        * D1 und D2 enthält jetzt die letzten Koordinaten
                        * (200,200)
 MOVE.W #200,D3         * Endposition neu Laden
 MOVE.W #10,D4
 JSR SETXYXY
 JSR GETCOLOR
 EORI.B #$FF,D0
 JSR SETCOLOR
 JSR LINE
RTS
```

SETSETMODE
Setzen des Setmode.

Der Setmode bestimmt die Art wie ein Punkt verändert werden soll.
So ist es möglich, recht einfach Bilder zu modifizieren.
In D0 wird der neue Setmode übergeben.

D0 Setmode

Setmode: (D0)

0 setzen der Farbe D0	1 setzen mit Hintergrundfarbe
2 auslesen der Punktes nach D0	3 ODER-Verknüpfung mit D0
4 AND-Verknüpfung mit D0	5 EXOR-Verknüpfung mit D0
6 addieren mit D0	7 subtrahieren mit D0
8 prüfe Bit D0 und invertiere	9 prüfe Bit D0 und lösche
10 prüfe Bit D0 und setze	11 prüfe Bit D0
12 Punkt mit D0 vergleichen	13 negiere Punkt
14 NOT des Punkt	15 testen ob der Punkt = 0
16 logisches schieben links um D0	17 logisches schieben rechts
18 rotiere links um D0	19 rortiere rechts
20 arithmetrisches schieben links um D0	21 arith. schieben rechts
22 setze Punkt im Window	23 lese Punkt aus Window
24 Berechne Punkt im Window	

Zur genaueren Beschreibung jetzt noch die entsprechenden Assembler Befehle

0 MOVE.B D0,PUNKT	1 MOVE.B HINTERGUND,PUNKT
2 MOVE.B PUNKT,D0	3 OR.B D0,PUNKT
4 AND.B D0,PUNKT	5 EOR.B D0,PUNKT
6 ADD.B D0,PUNKT	7 SUB.B D0,PUNKT
8 BCHG.B D0,PUNKT	9 BCLR.B D0,PUNKT
10 BSET.B D0,PUNKT	11 BTST.B D0,PUNKT
12 CMP.B PUNKT,D0	13 NEG.B PUNKT
14 NOT.B PUNKT	15 TST.B PUNKT
16 LSL.B D0,PUNKT	17 LSR.B D0,PUNKT
18 ROL.B D0,PUNKT	19 ROR.B D0,PUNKT
20 ASL.B D0,PUNKT	21 ASR.B D0,PUNKT
22 BSR SETWDOT	23 BSR GETWDOT
24 BSR CALCWDOT	

GETSETMODE
Lesen des Setmode.

Der Setmode bestimmt die Art wie ein Punkt verändert werden soll.
Dieser kann nun mit dem GETSETMODE Befehl ausgelesen werden.
In D0 wird der aktuelle Setmode übergeben.

>D0 Setmode

Setmode: (D0)

0 setzen der Farbe D0	1 setzen mit Hintergrundfarbe
2 auslesen der Punktes nach D0	3 ODER-Verknüpfung mit D0
4 AND-Verknüpfung mit D0	5 EXOR-Verknüpfung mit D0
6 addieren mit D0	7 subtrahieren mit D0
8 prüfe Bit D0 und invertiere	9 prüfe Bit D0 und lösche
10 prüfe Bit D0 und setze	11 prüfe Bit D0
12 Punkt mit D0 vergleichen	13 negiere Punkt
14 NOT des Punkt	15 testen ob der Punkt = 0
16 logisches schieben links um D0	17 logisches schieben rechts
18 rotiere links um D0	19 rortiere rechts
20 arithmetrisches schieben links um D0	21 arith. schieben rechts
22 setze Punkt im Window	23 lese Punkt aus Window
24 Berechne Punkt im Window	

Zur genaueren Beschreibung jetzt noch die entsprechenden Assembler Befehle

0 MOVE.B D0,PUNKT	1 MOVE.B HINTERGUND,PUNKT
2 MOVE.B PUNKT,D0	3 OR.B D0,PUNKT
4 AND.B D0,PUNKT	5 EOR.B D0,PUNKT
6 ADD.B D0,PUNKT	7 SUB.B D0,PUNKT
8 BCHG.B D0,PUNKT	9 BCLR.B D0,PUNKT
10 BSET.B D0,PUNKT	11 BTST.B D0,PUNKT
12 CMP.B PUNKT,D0	13 NEG.B PUNKT
14 NOT.B PUNKT	15 TST.B PUNKT
16 LSL.B D0,PUNKT	17 LSR.B D0,PUNKT
18 ROL.B D0,PUNKT	19 ROR.B D0,PUNKT
20 ASL.B D0,PUNKT	21 ASR.B D0,PUNKT
22 BSR SETWDOT	23 BSR GETWDOT
24 BSR CALCWDOT	

SETBACKGROUND
Setzen der Hintergrundfarbe.

Mit diesem Befehl kann man die gespeicherte Hintergrundfarbe verändern. Dadurch kann man beim setzten von Punkten oder Linien einen Festen Farbwert anwählen, ohne immer auf D0 zu achten.

D0 Hintergrundfarbe

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0
 JSR SETBACKGROUND       * Neue Hintergrundfarbe ROT
 MOVE.W #1,D0
 JSR SETSETMODE          * Setzen mit Hintergrundfarbe
 CLR D0                  * Als beweis D0 löschen
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.W #255,D1
 MOVE.W #255,D2
 JSR DRAWTO              * Nicht auf D0 achten
RTS
```

GETBACKGROUND
Lesen der Hintergrundfarbe.

Mit diesem Befehl kann man die gespeicherte Hintergrundfarbe auslesen. Dadurch ist es möglich zu Testen ob ein Bildpunkt der Hintergrundfarbe oder einer anderen entspricht.

>D0.B Hintergrundfarbe

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #$03,D0
 JSR CLRCOL              * Löschen mit Farbe ROT
 JSR GETBACKGROUND       * Lesen der Hintergrundfarbe (ROT)
RTS
```

SETZCOLOR
Setzten der Zeichenfarbe.

Mit diesem Befehl wird die aktuelle Zeichenfarbe verändert.
Dadurch ist es möglich einen Bunten Text darzustellen.

D0 Zeichenfarbe

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0          * Zeichenfarbe Dunkelrot
 MOVE.B D0,D6            * Speichern
 MOVE.W #0,D1            * X = 0
 MOVE.W #18,D2           * Y = 18
 MOVE.W #$00,D3          * Cursor aus
 JSR SETCURXY
LOOP:
 ADDQ.B #1,D6            * Eine farbe höher
 MOVE.B D6,D0
 JSR SETZCOLOR           * Farbe setzen
 JSR CI                  * Zeichen einlesen
 JSR CO                  * Zeichen ausgeben
 BRA LOOP
```

GETZCOLOR
Lesen der Zeichenfarbe.

Mit diesem Befehl wird die aktuelle Zeichenfarbe auslesen.

>D0.B Zeichenfarbe

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0          * Zeichenfarbe Dunkelrot
 MOVE.W #0,D1            * X = 0
 MOVE.W #18,D2           * Y = 18
 MOVE.W #$00,D3          * Cursor aus
 JSR SETCURXY
LOOP:
 JSR GETZCOLOR           * Alte Farbe lesen
 ADDQ.B #1,D0            * Eine farbe höher
 JSR SETZCOLOR           * Farbe setzen
 JSR CI                  * Zeichen einlesen
 JSR CO                  * Zeichen ausgeben
 BRA LOOP
```

GETCURXY
Lesen der Aktuellen Cursorposition.

Dieses ist der gegenteilige Befehl zum SETCURXY. Es wird nach D0 die Zeichenfarbe gelesen, nach D1 und D2 die X und Y Position und nach D3 der Cursormode.
Cursormode:
$00 = Cursor dauernd aus
$01-$FE = Cursor blinkzeit * 20ms
$FF = Cursor dauernd an

>D0.B	Zeichenfarbe
>D1.W	X-Position
>D2.W	Y-Position
>D3.B	Cursormode

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0          * Zeichenfarbe Dunkelrot
 MOVE.W #0,D1            * X = 0
 MOVE.W #18,D2           * Y = 18
 MOVE.W #$10,D3          * Cursor Frequenz
 JSR SETCURXY
 MOVE.B #'A',D0
 JSR CO
 MOVE.B #'a',D0
 JSR CO
 JSR GETCURXY            * Aktuelle Position lesen
 ADD0.W #1,D1            * 1 Zeichen links
 ADD0.W #1,D2            * 1 Zeichen rauf
 JSR SETCURXY
 MOVE.B #'B',D0
 JSR CO
 MOVE.B #'b',D0
 JSR CO
RTS
```

CRLF
Carriage Return ausgeben.

Es wird der Code Carriage Return ausgegeben. Dieses entspricht dem $0D beim CO Befehl. Dadurch wandert der Cursor eine Zeile nach unten, an den linken Randes.

Keine Register

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$01,D0          * Zeichenfarbe Dunkelrot
 MOVE.W #0,D1            * X = 0
 MOVE.W #18,D2           * Y = 18
 MOVE.W #$10,D3          * Cursor Frequenz
 JSR SETCURXY
 MOVE.B #'a',D0
 JSR CO
 JSR CRLF                * Return ausgeben
 MOVE.B #'b',D0
 JSR CO
 JSR CRLF                * Return ausgeben
 MOVE.B #'c',D0
 JSR CO
RTS
```

GETGRAU
Lesen eines Grauwertes.

Mit der COL 256 kann man 16 verschiedene Graustufen darstellen. Allerdings ist es etwas schwierig, wenn man diese direkt angeben möchte, da dieses ein Zusammenspiel aller Farben ist. Dafür gibt es den GETGRAU Befehl. Als Eingabe gibt man die Graustufe, die man haben möchte, zwischen 0 und 15 an, und als Ausgabe erhält man dann den Ausgabecode für die COL 256. Damit ist es sehr einfach, ohne grossen Aufwand, eine Graustufe zu erhalten.

D0	Graustufe	>D0.B	Farbwert

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #15,D7
LOOP:
 MOVE.W D7,D0
 JSR GETGRAU
 MOVE.W #0,D1
 MOVE.W D7,D2            * Von unten links
 JSR MOVETO
 MOVE.W #255,D1
 MOVE.W D7,D2
 ADD.W #255-11,D2        * Nach oben rechts
 JSR DRAWTO
 DBRA D7,LOOP
RTS
```

GETBLAU
Lesen eines Blauwertes.

Mit der COL 256 kann man 12 verschiedene Blaustufen darstellen. Allerdings ist es etwas schwierig, wenn man diese direkt angeben möchte, da dieses ein Zusammenspiel aller Farben ist. Dafür gibt es den GETBLAU Befehl. Als Eingabe gibt man die Blaustufe, die man haben möchte, zwischen 0 und 11 an, und als Ausgabe erhält man dann den Ausgabecode für die COL 256. Damit ist es sehr einfach, ohne grossen Aufwand, eine Blaustufe zu erhalten.

D0	Blaustufe	>D0.B	Farbwert

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #11,D7
LOOP:
 MOVE.W D7,D0
 JSR GETBLAU
 MOVE.W #0,D1
 MOVE.W D7,D2            * Von unten links
 JSR MOVETO
 MOVE.W #255,D1
 MOVE.W D7,D2
 ADD.W #255-11,D2        * Nach oben rechts
 JSR DRAWTO
 DBRA D7,LOOP
RTS
```

GETGRUEN
Lesen eines Gruenwertes.

Mit der COL 256 kann man 12 verschiedene Gruenstufen darstellen. Allerdings ist es etwas schwierig, wenn man diese direkt angeben möchte, da dieses ein Zusammenspiel aller Farben ist. Dafür gibt es den GETGRUEN Befehl. Als Eingabe gibt man die Gruenstufe, die man haben möchte, zwischen 0 und 11 an, und als Ausgabe erhält man dann den Ausgabecode für die COL 256. Damit ist es sehr einfach, ohne grossen Aufwand, eine Gruenstufe zu erhalten.

D0	Gruenstufe	>D0.B	Farbwert

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #11,D7
LOOP:
 MOVE.W D7,D0
 JSR GETGRUEN
 MOVE.W #0,D1
 MOVE.W D7,D2            * Von unten links
 JSR MOVETO
 MOVE.W #255,D1
 MOVE.W D7,D2
 ADD.W #255-11,D2        * Nach oben rechts
 JSR DRAWTO
 DBRA D7,LOOP
RTS
```

GETROT
Lesen eines Rotwertes.

Mit der COL 256 kann man 12 verschiedene Rotstufen darstellen. Allerdings ist es etwas schwierig, wenn man diese direkt angeben möchte, da dieses ein Zusammenspiel aller Farben ist. Dafür gibt es den GETROT Befehl. Als Eingabe gibt man die Rotstufe, die man haben möchte, zwischen 0 und 11 an, und als Ausgabe erhält man dann den Ausgabecode für die COL 256. Damit ist es sehr einfach, ohne grossen Aufwand, eine Rotstufe zu erhalten.

D0	Rotstufe	>D0.B	Farbwert

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.W #11,D7
LOOP:
 MOVE.W D7,D0
 JSR GETROT
 MOVE.W #0,D1
 MOVE.W D7,D2            * Von unten links
 JSR MOVETO
 MOVE.W #255,D1
 MOVE.W D7,D2
 ADD.W #255-11,D2        * Nach oben rechts
 JSR DRAWTO
 DBRA D7,LOOP
RTS
```

SEARCH

Vertauschen von Farben.

Mit diesem Befehl kann man eine Farbe eines Rechteckes verändern.
Es wird ein Rechteck mit den Koordinaten D1 D2 bis D3 D4 umgefärbt.
Dabei wird die Farbe die in D5 steht gesucht und durch die Farbe die
in D6 steht vertauscht. Dadurch erhält man ein schnelles ändern einzelner
Farben, ohne gleich den Bildschirm neu aufzubauen.

D1	X-Position Start
D2	Y-Position Start
D3	X-Position Ende
D4	Y-Position Ende
D5	Suchfarbe
D6	Neue Farbe

```
START:
 MOVE.W #0,D0
 JSR COLINIT
 MOVE.B #$03,D0
 JSR CLRCOL
 MOVE.W #0,D1
 MOVE.W #0,D2
 JSR MOVETO
 MOVE.W #255,D1
 MOVE.W #255,D2
 MOVE.B #$30,D0
 JSR DRAWTO
 MOVE.W #60,D1
 MOVE.W #60,D2
 MOVE.W #180,D3
 MOVE.W #180,D4          * Ausschnitt 60,60 bis 180,180
 MOVE.W #$03,D5          * Farbe ROT suchen
 MOVE.W #$0C,D6          * Durch Gruen ersetzen
 JSR SEARCH
RTS
```

Graf Elektronik Systeme GmbH
Magnusstraße 13 · Postfach 1610
8960 Kempten (Allgäu)
Telefon: (08 31) 62 11
Teletex: 831804 = GRAF
Telex: 17 831804 = GRAF
Datentelefon: (08 31) 6 93 30

Filiale Hamburg
Ehrenbergstraße 56
2000 Hamburg 50
Telefon: (0 40) 38 81 51

Filiale München:
Georgenstraße 61
8000 München 40
Telefon: (0 89) 2 71 58 58

Öffnungszeiten der Filialen:
Montag – Freitag
10.00 – 12.00 Uhr, 13.00 – 18.00 Uhr
Samstag 10.00 – 14.00 Uhr

Verkauf:
Computervilla
Ludwigstraße 18 b
(bei Möbel-Krügel)
8960 Kempten-Sankt Mang

Öffnungszeiten:
Montag – Freitag
10.00 – 12.00 Uhr, 13.00 – 18.00 Uhr
langer Samstag 10.00 – 14.00 Uhr