

Datei zur Dokumentation aller Future Operating System Systemfunktionen der Version .8 des OS. Sämtliche für den Anwender oder Programmierer wichtige OS Funktionen des A-ROMs werden erklärt und lokalisiert.

Alle OS Funktionen werden in folgender Form beschrieben:

1. Kurzbeschreibung: Die OS Funktion wird in einem Satz beschrieben.

2. Label: Mit diesem Label wird die OS Funktion im Source Code bezeichnet. In der mitgelieferten Label-Bibliothek (siehe Datei #EQU-API.DEU) mit den jeweils aktuellen ROM Adressen findet ebenfalls dieses Label Verwendung. Aus Gründen der Kompatibilität sollte man in eigenen Programmen alle OS- / System-Funktionen (und System-Variablen) stets mit diesen Labels ansprechen.

3. ROM-Nummer: Hier ist die logische ROM - Nummer des FutureOS ROMs angegeben, in dem die entsprechende OS Funktion zu finden ist. Manche OS Funktionen kommen, wegen ihrer Kürze, in mehreren ROMs vor, dann sind hier auch mehrere ROM Nummern angegeben. Da die ROM Nummer logisch zu verstehen ist, sollte man sie nicht mit dem physikalischen ROM Select gleichsetzen. Wie man die physikalische ROM Nummer eines der FutureOS ROMs ermittelt wird im Handbuch erklärt.

4. Startadresse: Gibt die Einsprung-Adresse der OS Funktion an. Falls diese in mehreren ROMs vorkommt, wird hier auch eine entsprechende Anzahl von Adressen angegeben.

5. Einsprungsbedingungen: Die Einsprungsbedingungen der OS Funktion werden beschrieben und es wird sowohl auf Registerinhalte als auch auf RAM-Variablen eingegangen.

6. Aussprungsbedingungen: Die Aussprungsbedingungen der OS-Funktion werden beschrieben und es wird sowohl auf Registerinhalte als auch auf RAM-Variablen eingegangen.

7. Manipuliert: Es werden alle manipulierten oder zerstörten Register und RAM-Variablen wiedergegeben. Manchmal werden auch manipulierte Pheripheriebausteine wiedergegeben.

8. Beschreibung: Es folgt eine vollständige Erklärung der Anwendung und Wirkungsweise der beschriebenen OS Funktion.

9. Bitte Beachten: Es werden einige wichtige Details kurz erklärt. Dies ist besonders wichtig, da alle OS Funktionen kompromißlos auf Höchstgeschwindigkeit getrimmt worden sind. Bei falscher Handhabung kann es zu Problemen mit der Systemstabilität kommen. Im ROM A befindet sich die Tastaturverwaltung, die Text- und Zeichen-Verwaltung, Teile der Festplattenverwaltung, die Druckerverwaltung. Funktionen für CPC-IDE, X-MASS, SYMBiFACE II und III, CPC-Booster. Das Dump und Porting-System und vieles mehr.

Die aktuelle Version dieser Datei finden Sie auch im Internet unter...

FutureOS Homepage: <http://www.FutureOS.de>

TESTE UND LESE DEN STATUS ALLER TASTEN

Kurzbeschreibung: Der Tastaturstatus wird getestet, falls irgendeine Taste gedrückt ist, wird ihr entsprechender ASCII Wert, unter Beachtung von SHIFT und CONTROL, bestimmt.

Label: H_ALLET

ROM-Nummer: A

Startadresse: &C02F

Einsprungsbedingungen: Die Tastaturübersetzungstabellen für Normal, Shift, Control und Shift+Control müssen intakt sein. D.h. ab TAST_N, TAST_S, TAST_C und TAST_SC müssen jeweils 80 Bytes mit Werten zwischen &00 und &FE folgen. &00, &FF sollten nicht verwendet werden.

Aussprungsbedingungen: A = ASCII Wert, falls entsprechende Taste gedrückt ist.
A = &FF falls keine Taste gedrückt wird.

Manipuliert: AF, BC, DE, HL und XL.
PIO Status, Port A und C. PSG Register 14, Latch Adress Register.

Beschreibung: Diese OS Funktion ermittelt ob eine Taste gedrückt ist. Falls keine Taste gedrückt ist kehrt sie mit &FF im Akku zurück. Falls aber gerade eine Taste gedrückt ist wird außerdem der Shift und Control Status der Tastatur überprüft.

Es werden vier Zustände unterschieden:

- weder Shift noch Control gedrückt,
- Shift allein gedrückt,
- Control allein gedrückt oder ...
- Shift und Control gemeinsam gedrückt.

Jeh nachdem wird aus einer von vier Tabellen gelesen. Da die Tastatur in einer 10 * 8 Matrix angeordnet ist enthält jede der vier Tabellen 80 Bytes. Der aus der Tabelle gelesene Wert wird im Akku zurückgeliefert (siehe #EQU-API.DEU).

Beispiel: Einlesen der ersten gedrückten Taste

```
LD  BC, (&FF01)
OUT (C),C           ; ROM A einblenden

SCHLEIFE CALL H_ALLET ; Tastatur einlesen,

INC A                ; Test ob A = &FF
JR  Z, SCHLEIFE      ; bis eine Taste gedrückt wurde
DEC A                ; Ausgleich des INC A
```

Der Akku ist nun mit dem ASCII Wert der gedrückten Taste geladen.

Bitte Beachten: Falls man einer Taste den ASCII Wert &FF zuordnet kann man nicht mehr auf Anhieb unterscheiden ob eine Taste mit dem Wert &FF oder gar keine Taste gedrückt wurde. Denn falls keine Taste gedrückt wurde wird im Akku der Wert &FF übergeben.

TESTE, LESE STATUS ALLER TASTEN, BEACHTE CAPS & EBENEN

Kurzbeschreibung: Der Tastaturstatus wird getestet, falls irgendeine Taste gedrückt ist, wird ihr entsprechender ASCII Wert, unter Beachtung von SHIFT, CONTROL und dem CAPS-LOCK Status bestimmt.

Label: H_XALLET (Erweiterung von H_ALLET s.o.)

ROM-Nummer: A

Startadresse: &C115

Einsprunghbedingungen: Die Tastaturübersetzungstabellen für Normal, Shift, Control und Shift+Control müssen intakt sein. D.h. ab TAST_N, TAST_S, TAST_C und TAST_SC müssen jeweils 80 Bytes mit Werten zwischen &00 und &FE folgen, &FF sollte nicht verwendet werden.

Die RAM-Variable CAPS muß einen korrekten Wert enthalten.

Aussprunghbedingungen:

A = ASCII Wert, falls entsprechende Taste gedrückt ist.

A = &00 und das Z-Flag ist gesetzt, falls CAPS gedrückt wurde.

A = &FF falls keine Taste gedrückt wird.

Manipuliert: AF, BC, DE, HL, XL und der PIO Status, Port A und C. PSG Register 14, Latch Adress Register. Außerdem wurde der Inhalt der RAM-Variable CAPS verändert, falls die CAPS Taste gedrückt wurde.

Beschreibung: Diese OS Funktion ist eine Erweiterung von H_ALLET. Siehe unbedingt dort!!

Bei gedrückter Taste wird ihr Wert im Akku übergeben. Wird gerade keine Taste gedrückt, dann enthält der Akku &FF (S-Flag gesetzt). Enthält der Akku &00, dann wurde CAPS gedrückt, und damit die Tastaturebene gewechselt.

Folgende Steuerfunktionen stehen zur Verfügung:

CAPS allein =====> schaltet GROSS-SCHREIBUNG dauerhaft ein.

CAPS + SHIFT =====> schaltet SHIFT-Ebene dauerhaft ein.

CAPS + CONTROL =====> schaltet CONTROL-Ebene dauerhaft ein.

CAPS + CONTROL + SHIFT ====> schaltet CONTROL+SHIFT-Ebene dauerhaft ein.

Drückt man dagegen eine "normale" Taste (nicht CAPS, CONTROL, SHIFT), dann wird ihr Wert, jeh nach Tastaturebene, im Akku übergeben. Drückt man zu dieser Taste zusätzlich SHIFT, CONTROL oder beide, dann wird diese Information mit der in CAPS enthaltenen Tastaturebene geXORt. Gleiches hebt sich auf!

Beispiel:

Ist die SHIFT Tastaturebene aktiv, und man drückt die Taste "s", dann wird im Akku der ASCII-Wert von "S" übergeben. Drückt man aber zusätzlich zu "s" noch SHIFT, dann wird im Akku auch "s" übergeben, da sich SHIFT XOR SHIFT aufhebt.

Der String-Editor STED benutzt diese OS Funktion, mit ihm läßt sich der Sachverhalt leicht nachvollziehen, z. B. beim Editieren eines Namens.

Bitte Beachten: Falls man einer Taste den ASCII Wert &00 oder &FF zuordnet kann man nicht mehr auf Anhieb unterscheiden ob eine Taste (mit dem Wert &00, &FF, CAPS) bzw. keine Taste gedrückt wurde.

TESTE OB IRGEND EINE TASTE GEDRÜCKT IST

Kurzbeschreibung: Diese OS Funktion testet ob irgendeine Taste gedrückt ist, oder nicht.

Label: TST_TS

ROM-Nummer: A

Startadresse: &C5FA

Einsprungsbedingungen: -

Aussprungsbedingungen:

Z-Flag gesetzt ==> Tastatur ist FREI, keine Taste gedrückt.

Z-Flag gelöscht ==> irgendeine Taste wird gerade gedrückt.

Manipuliert: AF, BC, DE, L und PIO & PSG

Beschreibung: Um zu testen ob überhaupt irgendeine Taste gedrückt ist, läßt sich diese Funktion verwenden. Dabei wird nicht auf einen Tastendruck gewartet, sondern der aktuelle Tastaturstatus ermittelt.

Beim Einsprung sind keine Parameter zu übergeben.

Nach dem Aussprung ist am Z Flag zu erkennen ob gerade eine Taste gedrückt ist. Ist das Z-Flag gesetzt (Z,1), dann ist die Tastatur frei, also wird momentan weder irgendeine Taste noch der Joystick betätigt. Wenn allerdings das Z-Flag rückgesetzt ist (NZ,0), so wird momentan mindestens eine Taste gedrückt. Um herauszufinden welche Taste dies ist, kann man z.B. H_ALLET benutzen.

Bitte Beachten: Diese OS Funktion wartet NICHT auf den Druck einer Taste, sondern liefert den aktuellen Tastatur-Status.

HOLE WERT VON &0 BIS &F UND GIB IHN AN DEN CRT

Kurzbeschreibung: Die Tastatur wird solange abgefragt, bis eine Taste von 0,1,2 bis 9, eine Taste von a,b,c bis f oder DEL gedrückt wird. Der Shift und Control Status wird dabei nicht beachtet. Ist ein Wert von &0 bis &F gelesen worden so wird dieser zusätzlich auf dem Bildschirm in Mode 2 ausgegeben, als ein Zeichen das dem Zahlenwert entspricht.

Label: A_F_0_9

ROM-Nummer: A

Startadresse: &C404

Einsprungsbedingungen: Die Cursorposition C_POS muß einen gültigen Wert haben, da sonst irgendwo ins RAM geschrieben wird. Mode 2 sollte aktiv sein, da das Zeichen in Mode 2 ausgegeben wird.

Aussprungsbedingungen: A = 0, 1, 2, ..., 9, &A, &B, ..., &F wenn die entsprechende Taste von 0..9 oder a..f gedrückt wurde. Falls die DEL Taste gedrückt wurde wird im Akku der Wert &FD geliefert.

Manipuliert: AF, BC, DE, HL und XL.
PIO Status, Port A und C. PSG Register 14, Latch Adress Register.

Beschreibung: Diese OS Funktion fragt die Tastatur ab, bis entweder die DEL Taste oder eine Taste mit der Beschriftung 0 bis 9 oder A bis F gedrückt wurde. Falls die DEL Taste gedrückt wurde liefert der Akku das Ergebniss &FD. Falls es eine Taste im Bereich von 0 bis 9 oder A bis F war so wird der entsprechende Wert in den Akku geladen.

Zusätzlich wird auf dem Bildschirm an der aktuellen Cursorposition das zugehörige ASCII Zeichen ausgegeben. So bekommt der Akku z.B. den Wert 8 wenn man die 8-Taste gedrückt hat, und es erscheint auf dem Bildschirm eine Acht. Drückt man z.B die C-Taste, dann wird der Akku auf &C gesetzt und zusätzlich der große Buchstabe C auf dem Bildschirm ausgegeben. Diese Funktion wird dazu verwendet um hexadezimale Werte von der Tastatur einzulesen. Diese werden gleichzeitig auf dem Bildschirm dargestellt. Es wird eine MODE 2 Ausgabefunktion verwendet.

Bitte Beachten: Die Funktion kehrt erst **nach** einem Tastendruck zurück. Wird sie aufgerufen, und man drückt danach keine Taste von 0 bis 9, A bis F oder DEL, dann kehrt diese Funktion auch nicht mehr zurück.

Das Zeichen das auf dem Bildschirm dargestellt wird wird mit der Modus 2 Zeichenausgabe ausgegeben. Diese OS Funktion sollte man also nur im Modus 2 verwenden.

WARTE BIS TASTATUR FREI IST

Kurzbeschreibung: Warten bis keine Taste mehr gedrückt wird.

Label: WART_TS

ROM-Nummer: A

Startadresse: &C4DB

Einsprungsbedingungen: -

Aussprungsbedingungen: kehrt erst zurück wenn keine Taste gedrückt ist.

Manipuliert: AF, BC, DE und L.
PIO Status, Port A und C. PSG Register 14, Latch Address Register.

Beschreibung: Diese OS Funktion testet die gesamte Tastatur, beide Joysticks und Mausanschluß (Joystick kompatible Mäuse) auf gedrückte Tasten. Eine Rückkehr erfolgt erst nachdem keine Taste mehr gedrückt wird. Vom analogen Joystick-Port werden nur die Feuerknöpfe getestet.

Verwendung findet diese OS Funktion z.B. dann, wenn man sicherstellen will, daß eine zuvor gedrückte Taste nun nicht mehr gedrückt wird.

Bitte Beachten: Wird irgendeine Taste ständig gedrückt, dann kehrt diese OS Funktion nicht mehr zurück, bis sich dies ändert. Einige Erweiterungen am Joystickport können u. U. einige Signale fest auf gedrückt legen, in diesem Fall sollte man diese OS Funktion nicht verwenden, da ja so eine Rückkehr nicht mehr möglich ist. Es sei denn durch das Entfernen jener Erweiterung. Die Maus aus dem Schneider-Magazin bzw. Computer Partner setzt z. B. einen Eingang des Joystickports fest auf gedrückt. Mit dieser Maus sollte man diese OS Funktion besser nicht benutzen.

WARTEN BIS DIE TASTATUR FREI- ODER DIE ZEIT ABGELAUFEN IST

Kurzbeschreibung: Diese OS Funktion dient dazu eine bestimmte Verzögerungszeit verstreichen zu lassen, solange irgendeine Taste gedrückt ist. Bei freigegebener Tastatur kehrt sie sofort zurück.

Label: XWART

ROM-Nummer: A

Startadresse: &C671

Einsprungsbedingungen: Es sind folgende RAM-Variablen zu setzen:

VZ_MOD	=	VerZögerungsmodus (0=Anfangs VZ /// 1=Folge VZ) normal auf 0.
VZ_AG	=	VerZögerung Anfang Generell (Wiederladewert) von &0000.&FFFF.
VZ_AA	=	VerZögerung Anfang Aktuelle Restzeit von &0000 bis &FFFF.
VZ_FG	=	VerZögerung Folge Generell (Wiederladewert) von &0000..&FFFF.
VZ_FA	=	VerZögerung Folge Aktuelle Restzeit von &0000 bis &FFFF.

Aussprungsbedingungen: OS Funktion kehrt erst dann zurück wenn **keine** Taste mehr gedrückt ist, oder wenn die vorgegebene Wartezeit abgelaufen ist.

Manipuliert: AF, BC, DE, HL

RAM-Variablen VZ_MOD, VZ_AA, VZ_FA und der Status der PIO, und des PSG.

Beschreibung: Diese OS Funktion stellt eine Kombination dar. Einerseits wartet sie darauf, daß die Tastatur freigegeben wird, andererseits kehrt sie spätestens dann zurück wenn die jeweilige Wartezeit abgelaufen ist. Bei der Wartezeit unterscheidet man zwischen Anfangsverzögerung und Folgeverzögerung. Dabei ist die Anfangsverzögerung die Zeit, die gewartet wird, wenn diese OS Funktion bei gedrückter Taste das erste Mal aufgerufen wird. Dagegen ist die Folgeverzögerung die Zeit die bei jeder weiteren Benutzung der OS Funktion gewartet wird, vorausgesetzt es ist noch immer irgendeine Taste gedrückt. Läßt man zwischendurch die Taste los, dann wird wieder mit der Anfangsverzögerung begonnen. Wird zu irgend einem Zeitpunkt keine Taste gedrückt, dann kehrt die OS Funktion sofort zurück.

Vor Aufruf dieser OS Funktion sind fünf Variablen mit Daten zu füllen, dies wird aber normalerweise vom OS erledigt. Man kann die Werte allerdings ändern. Siehe nächste Seite.

Die 8 Bit Variable VZ_MOD gibt den Verzögerungsmodus an. Normalerweise wird sie auf &00 gesetzt, also Anfangs VZ. Will man ohne spezielle Anfangs VZ arbeiten, dann setzt man sie auf einen Wert größer &00.

Die 16 Bit Variablen VZ_AG und VZ_AA (s.o.) können einen Wert von &0000 bis &FFFF annehmen, müssen aber auf den gleichen Wert gesetzt werden. In ihnen wird die Anfangsverzögerung gespeichert.

Auch die beiden 16 Bit Variablen VZ_FG und VZ_FA müssen den selben Wert zwischen &0000 und &FFFF beinhalten, sie geben die Folgeverzögerung an.

Standartmäßig sollte man die fünf Variablen folgendermaßen laden:

VZ_MOD = &00

VZ_AG = &1000

VZ_AA = &1000

VZ_FG = &0200

VZ_FA = &0200

Bitte Beachten: Die VZ_ Variablen werden vom OS gesetzt, man kann sie aber anpassen. Dies hat Einfluss auf die Reaktion der Tastatur.

TESTEN EINER TASTE ODER TASTENKOMBINATION

Kurzbeschreibung: Schneller Test einer bestimmten Taste oder mehrerer Tasten in einer Tastaturreihe der Tastaturmatrix.

Label: HOLE1TS

ROM-Nummer: A

Startadresse: &C561

Einsprungsbedingungen:

H = Nummer der Tastaturreihe von 0 bis 9, wobei H keinesfalls den Wert 63 = &3F übersteigen darf.

L = Maske für die zu testende Taste oder für die zu testende Tastenkombination innerhalb der Tastaturreihe, die durch H adressiert wird. Hierbei symbolisiert jedes gesetzte Bit eine zu testende Taste.

Aussprungsbedingungen: Der Akku hat den Wert &00 und das Zero Flag ist gesetzt wenn die entsprechende Taste, oder alle Tasten der Tastenkombination gedrückt waren. Ist der Akku ungleich &00 und das Zero Flag gelöscht, dann wurde entweder die Taste nicht gedrückt, oder es wurden nicht alle Tasten der Tastenkombination gedrückt.

Manipuliert: AF, BC, DE und H
PIO Status, Port A und C. PSG Register 14, Latch Adress Register.

Beschreibung: Diese Funktion ermöglicht es sehr schnell eine bestimmte Taste zu testen. Anhand des Akkus und des Zero Flags kann man erkennen ob die gewählte Taste zu diesem Zeitpunkt gedrückt wurde. Für die Verwendung dieser OS Funktion ist es erforderlich den Aufbau der Tastaturmatrix im CPC zu kennen. Denn man teilt der Funktion lediglich mit in welcher der 10 Reihen sich die Taste befindet (Z80 Register H) und welche der acht Tasten der entsprechenden Reihe getestet werden soll. Es ist durchaus möglich in dieser 8-Bit Maske mehr als nur ein Bit zu setzen. Es wird dann eine Kombination von bis zu acht Tasten abgefragt. Man erhält allerdings nur dann die Meldung "gedrückt", wenn alle Tasten gedrückt wurden.

Beispiel: Will man z.B. die Taste mit dem "S" testen, dann lädt man das Register H mit &07, da sich die Taste "S" in dieser Matrixreihe der Tastatur befindet. Die Taste S wird innerhalb der siebten Tastaturreihe durch das 5. Bit, also Bit 4 dargestellt man lädt also das Register L mit &10. &10 sieht im Binärsystem so aus: 0001 0000. Will man mit der Taste "S" gleichzeitig auch die Taste "D" testen, dann lädt man das Register L einfach mit dem Wert &30, was im Binärsystem für 0011 0000 steht. Es wird also für jede zu prüfende Taste ein Bit gesetzt.

Bitte Beachten: Beim Testen einer Tastenkombination ist zu beachten das alle Tasten dieser Kombination innerhalb EINER der 10 Matrixreihen der Tastatur liegen. Tasten verschiedener Matrixreihen können nicht auf einmal getestet werden.

KOMPLETTEN TASTATURSTATUS INS RAM LESEN

Kurzbeschreibung: Alle 80 Tasten werden eingelesen und im RAM in 10 Bytes abgespeichert.

Label: HOLETST

ROM-Nummer: A

Startadresse: &C585

Einsprungsbedingungen: HL = Zeiger auf einen 10 Bytes großen Datenbereich ab dem die Tastaturinformation abzulegen ist.

Aussprungsbedingungen: Die 10 Bytes ab HL sind mit den aktuellen Tastaturinformationen gefüllt. Achtung: Eine gedrückte Taste wird durch ein gelöscht Bit symbolisiert. Steht ein Bit auf 1 dann wurde die zugehörige Taste nicht gedrückt.

Manipuliert: AF, BC, DE und HL
PIO Status, Port A und C. PSG Register 14, Latch Adress Register.

Beschreibung: Diese OS Funktion liest die komplette Tastaturmatrix ins RAM. Da diese Matrix aus 10 Reihen zu acht Bit besteht werden 10 Bytes an RAM benötigt, um die Daten aufzunehmen. Jede Taste wird durch ein eigenes Bit symbolisiert. In die anfangs durch HL adressierte Speicherstelle wird die Tastaturreihe 0 eingelesen, in HL + 1 wird die Reihe Nr. 1 eingelesen, usw. bis schließlich in HL + 9 die neunte Reihe eingelesen wird.

Bitte Beachten: Da hier nur die Hardware eingelesen wird und die Daten nicht bearbeitet werden stellt hier ein gelöscht Bit eine gedrückte Taste dar und umgekehrt. Ist ein Bit also gesetzt, so wurde die entsprechende Taste NICHT gedrückt.

ABFRAGE DER CURSORTASTEN UND DER COPY-TASTE

Kurzbeschreibung: Diese OS Funktion gibt dem Anwender die Möglichkeit speziell die vier Cursor Tasten (= Pfeiltasten) und die Copy-Taste abzufragen.

Label: H_CURA

ROM-Nummer: A

Startadresse: &C250

Einsprungsbedingungen: -

Aussprungsbedingungen: Der Tastenstatus der fünf Tasten wird im Akku zurückgegeben. Die Bits haben folgende Bedeutung:

A = (nur Bits 0..4 relevant)

Bit 0: Pfeil AUF

Bit 1: Pfeil RECHTS

Bit 2: Pfeil AB

Bit 3: Pfeil LINKS

Bit 4: COPY

Manipuliert: AF, BC, DE, L und PIO, PSG

Beschreibung: Diese OS Funktion dient dazu gleichzeitig den Status von allen vier Cursor Tasten und der Copy-Taste einzulesen. Es gibt keine zu übergebenden Parameter, die PIO und der PSG werden entsprechend programmiert.

Diese Funktion übergibt den Tastaturstatus im Akku, dabei sind nur die unteren fünf Bits 0..4 relevant. Die Bits liefern von 0 bis 4 folgenden Tasten: AUF, RECHTS, AB, LINKS und COPY. Wobei ein gelöscht Bit eine gedrückte Taste symbolisiert.

Bitte beachten: Achtung, ein Null-Bit symbolisiert eine gedrückte Taste. Ist ein Bit gesetzt, dann ist die Taste NICHT gedrückt.

ABFRAGE DER CURSORTASTEN, COPY UND KLEINES ENTER

Kurzbeschreibung: Diese OS Funktion gibt Aufschluß über den Status der vier Cursorstasten, der Copy und der kleinen Enter-Taste.

Label: H_CCE

ROM-Nummer: A

Startadresse: &C284

Einsprungsbedingungen: -

Aussprungsbedingungen: A = Tastaturstatus (Bits 0..5 relevant)

Bit 0: Pfeil AUF

Bit 1: Pfeil AB

Bit 2: Pfeil LINKS

Bit 3: Pfeil RECHTS

Bit 4: COPY

Bit 5: kleines ENTER

Manipuliert: AF, BC, DE, L und PIO, PSG

Beschreibung: Aufgabe dieser OS Funktion ist es den Tastaturstatus von den vier Cursorstasten, der Copy-Taste und der kleinen Enter-Taste zu ermitteln. Die Information wird im Akku geliefert. Die Bits von 0 bis 5 haben folgende Bedeutung AUF, AB, LINKS, RECHTS, COPY und KLEINES ENTER. Ein gelöscht Bit symbolisiert eine gedrückte Taste.

Diese Funktion ist zum Joystick kompatibel, d.h. die Cursorstasten vertreten die vier Richtungen, in die man den Joystick bewegen kann. Die Copy Taste emuliert Feuer 0 und das kleine Enter vertritt Feuer 1.

Diese Abfrage und die Joystickabfrage mittels der OS Funktion H_JOY sind also bitkompatibel.

Bitte beachten: Vorsicht, ist im Akku ein Bit gelöscht, dann ist die entsprechende Taste gerade gedrückt. Wenn die Taste nicht betätigt wird, dann ist das zugehörige Bit gesetzt.

CURSORTASTEN, COPY UND ESC EINLESEN UND AUSWERTEN

Kurzbeschreibung: Die Cursortasten und die Tasten Copy und ESC werden eingelesen. Bei ESC erfolgt eine Spezialbehandlung. Sonst wird in A ein Byte übergeben aus dem sich der Tastaturstatus ablesen läßt.

Label: CUR_CPY

ROM-Nummer: A

Startadresse: &C6C0

Einsprungsbedingungen: REG16_1 (System RAM Variable) muß gleich dem SP Register sein. Oder REG16_1 muß einen neuen Wert für SP enthalten, der bei Druck von ESC in SP geladen werden kann.

Aussprungsbedingungen: Je nach gedrückter Taste wird im Akku ein Wert übergeben.

Cursor aufwärts	==>	A = &F0.
Cursor abwärts	==>	A = &F1.
Cursor links	==>	A = &F2.
Cursor rechts	==>	A = &F3.
Copy gedrückt	==>	A = &A4.

Falls ESC gedrückt wird, so wird der Wert aus REG16_1 in SP geladen und der Akku ist gelöscht.

Manipuliert: AF, BC, DE und SP falls ESC gedrückt wurde.

Beschreibung: Diese OS Funktion ließt den Tastaturstatus der vier Cussortasten, der Copy und der ESC Taste ein. Falls eine der vier Cursortasten oder die Copy Taste gedrückt wird, so wird der Akku mit einem der folgenden Bytes geladen.

Cursor aufwärts gedrückt: .. Der Akku bekommt den Wert: &F0.
Cursor abwärts gedrückt: ... Der Akku bekommt den Wert: &F1.
Cursor links gedrückt: Der Akku bekommt den Wert: &F2.
Cursor rechts gedrückt: Der Akku bekommt den Wert: &F3.
Copy gedrückt: Der Akku bekommt den Wert: &A4.

Wird allerdings die ESCape Taste gedrückt so verhält sich die OS Funktion vollkommen anders: Der Akku wird gelöscht und aus der RAM Variable REG16_1 wird ein 16 Bit Wort gelesen, und damit der Stackpointer SP geladen. Wird hier ein falscher Wert benutzt, so wird das System nach dem nächsten RET Befehl chancenlos abstürzen. Der Datentransfer von REG16_1 zu SP bietet aber folgende mächtige und vielseitige Möglichkeit. An irgendeiner Stelle des Programms wird der Wert des Stackpointers SP in die Variable REG16_1 geladen. Danach können nun mehrere CALL Befehle folgen, d.h. der Stack wird manipuliert. Wird nun beim Aufruf dieser OS Funktion die ESC Taste gedrückt, so wird der alte Wert des Stackpointers wieder hergestellt, und ein RET Befehl gegeben.

Diese OS Funktion kehrt also bei ESC dahin zurück wovon zuvor die Sicherung des Stacks durch einen Call ausgelöst wurde. Vor der Verwendung dieser OS Funktion sollte man damit einzeln experimentieren, und vor der Sicherung des Stackpointers (s.o.) muß mindestens ein CALL Befehl gegeben werden. Da bei ESC nach eben diesen CALL zurückgekehrt wird.

Bitte Beachten: Ist die RAM Variable REG16_1 nicht mit dem selben Wert wie das Register SP, oder einem anderem Wert der auf ein 2 Byte Wort innerhalb des Stacks zeigt, geladen, so wird dies zu einem Systemabsturz führen. Will man nur die Cursortasten und Copy testen, dann sollte man anstatt dieser OS Funktion die H_CURA OS Funktion verwenden. Trotzdem ist diese OS Funktion sehr vielseitig einsetzbar, wenn das Programm beim Druck von ESC an irgend einer anderen Stelle fortgesetzt werden soll.

Diese OS Funktion wird von der F_PORT Monitorfunktion benutzt.

ABFRAGE BEIDER DIGITAL-JOYSTICKS

Kurzbeschreibung: Der aktuelle Status beider digitalen Joysticks wird ermittelt und in je einem Register geliefert.

Label: H_JOY

ROM-Nummer: A

Startadresse: &C2CC

Einsprungsbedingungen: -

Aussprungsbedingungen: Der Status der beiden Joysticks wird in H und L zurückgegeben. Nur Bits 0..5 relevant.

L = Joystick 0

H = Joystick 1

Die Bits haben folgende Bedeutung, ein geleertes Bit entspricht einer gedrückten Taste.

Bit 0: Joy 0 bzw. 1 AUF

Bit 1: Joy 0 bzw. 1 AB

Bit 2: Joy 0 bzw. 1 LINKS

Bit 3: Joy 0 bzw. 1 RECHTS

Bit 4: Feuer 0

Bit 5: Feuer 1

Bit 6: Feuer 2 - nur bei CPC old Generation, oder bei Atari ST Maus.

Manipuliert: AF, BC, DE, HL und PIO, PSG

Beschreibung: Will man Informationen über den Status der Joysticks, dann tritt diese OS Funktion in Aktion. Es werden beide digitalen Joysticks abgefragt. Die Daten des Joystick 0 werden im L Register abgelegt, im H Register befinden sich dann die Daten des zweiten Joysticks. Beide Register haben den selben Aufbau, es sind jeweils nur die sechs unteren Bits relevant. Die Bits 0..5 haben folgende Bedeutung: AUF, AB, LINKS, RECHTS, FEUER 0 und FEUER 1. Wobei je ein GELÖSCHTES Bit eine GEDRÜCKTE Taste symbolisiert. Diese Funktion ist zur OS Funktion H_CCE bitkompatibel (siehe dort).

Bitte beachten: Ein gesetztes Bit symbolisiert eine NICHT gedrückte Taste. Die 3. Feuertaste, 7. Bit wird bei den alten CPCs mitgeliefert, sollte aber nicht verwendet werden, da die CPCs der Plus Serie, dank des ASIC Bausteins keinen solchen Anschluss mehr haben. Die Plus Besitzer können sich dieses Bit aber sehr einfach selber dazulöten, und sind dann kompatibel.

Abfrage von DIGITAL-JOYSTICKS, MULTIPLAY, CursorTasten, COPY & Enter

Kurzbeschreibung: Der aktuelle Status beider digitalen Joysticks, bzw. der beiden Joysticks am MultiPlay, bzw. der Cursortasten und Copy, sowie des kleinen Enter wird ermittelt und im Akku übergeben.

Label: H_JC

ROM-Nummer: A

Startadresse: &FE6B

Einsprungsbedingungen: -

Aussprungsbedingungen: Der Akku enthält die Tasten-Bits des ersten, des zweiten Joysticks, oder der Cursortasten mit Copy und dem kleinen Enter. Diese OS Funktion ist Bit-kompatibel zu H_CCE und zu H_JOY.

Ist ein MultiPlay angeschlossen, so werden beide Joysticks abgefragt.

Die Bits haben folgende Bedeutung, ein geleertes Bit entspricht einer gedrückten Taste / aktivierten Eingabe.

Bit 0: Aufwärts

Bit 1: Abwärts

Bit 2: Links

Bit 3: Rechts

Bit 4: Feuer 0

Bit 5: Feuer 1

Bit 6: Feuer 2 - nur bei CPC6128, Atari ST Maus oder MultiPlay

Manipuliert: AF, BC, DE, HL und PIO, PSG

Beschreibung: Mit dieser OS Funktion läßt sich der Status beider Joysticks am Computer, beider Joysticks am MultiPlay und der Status der Cursortasten mit Copy und dem kleinen Enter durch einen Aufruf ermitteln. Der Anwender kann also mit einem der vier möglichen digitalen Joysticks, oder mit den Cursortasten + Copy + Enter arbeiten. Diese OS Funktion fragt alle fünf Eingabemöglichkeiten ab.

Allerdings existiert in der Abfragereihenfolge eine Priorität: Oberste Priorität hat Joystick 0, dann kommt Joystick 1, dann Joysticks 0 und 1 am MultiPlay, dann die Cursortasten mit Copy und dem kleinen Enter. Die Steuersignale von Joystick 0 sind so z.B. denen von Joystick 1 und Joystick 0 und 1 des MultiPlay, sowie der Tastatur überlegen etc.

Der Eingabe-Status wird im Akku übergeben. Es sind jeweils nur die sieben unteren Bits relevant. Die Bits 0..6 haben folgende Bedeutung:

AUF, AB, LINKS, RECHTS, FEUER 0, 1 und 2. Wobei je ein GELÖSCHTES Bit eine GEDRÜCKTE Taste symbolisiert. Diese OS Funktion ist zu den OS Funktionen H_CCE und H_JOY bitkompatibel (siehe dort).

Bitte beachten: Ein gesetztes Bit symbolisiert eine NICHT gedrückte Taste. Die 3. Feuertaste, 7. Bit wird bei den alten CPCs mitgeliefert, sowie bei der MultiPlay Erweiterung, nicht aber beim 6128 Plus. Die Plus Serie hat keinen solchen Anschluss mehr, da an den Joystick-Anschlüssen eine Diode gespaart wurde. Plus User können sich dieses Bit aber sehr einfach selber dazulöten, und sind dann kompatibel.

TESTE OB MultiPlay FREI IST

Kurzbeschreibung: Diese OS Funktion tested ob das MultiPlay frei ist oder ob gerade eine Eingabe erfolgt. Es werden beide Ports abgefragt.

Label: TMPA

ROM-Nummer: A

Startadresse: &D9E6

Einsprungsbedingungen: -

Aussprungsbedingungen: Das Zero Flag gibt Aufschluss über den Status beider MultiPlay Joysticks.

Z-Flag gesetzt: Es ist entweder gar kein MultiPlay angeschlossen oder es erfolgt gerade keine Eingabe über einen der beiden Joysticks.

Z-Flag geleert: Es erfolgt gerade eine Eingabe über einen oder beide der MultiPlay Joysticks.

Register BC enthält den Wert &F991

Manipuliert: AF und BC.

Beschreibung: Diese OS Funktion erlaubt es den Status der beiden MultiPlay Joysticks abzufragen.

Zuerst wird das BIT 3 des Konfigurations-Bytes KF_MED abgefragt. Wenn dieses Bit geleert ist, dann ist kein MultiPlay angeschlossen und diese Funktion kehrt mit gesetztem Zero-Flag zurück.

Falls ein MultiPlay angeschlossen ist werden beide Joysticks einglesen und getestet. Falls gerade keine Eingabe erfolgt kehrt TMPA ebenfalls mit gesetztem Zero-Flag zurück.

Sollte das Zero-Flag jedoch geleert sein, so erfolgt gerade irgendeine Eingabe über einen oder beide Joysticks des MultiPlay.

Bitte beachten: Bei gesetztem Zero-Flag kann man nicht wissen ob entwedre gar kein MultiPlay angeschlossen ist oder ob gerade keine Eingabe erfolgt.

Es existiert keine gesonderte OS Funktion zum Einlesen der MultiPlay Joysticks, da man einfach nur die Ports einlesen muss:

Erster Joystick: LD BC,&F990 ;Port des 1. Joysticks
IN A, (C)

Zweiter Joystick: LD BC,&F991 ;Port des 2. Joysticks
IN A, (C)

Dabei symbolisiert ein gesetztes Bit eine gedrückte Taste / Richtung. (Bei Tastatur und Joysticks des CPC ist es genau umgekehrt, beim CPC symbolisiert ein gelöscht Bit eine gedrückte Taste).

BIT 0: Aufwärts

BIT 1: Abwärts

BIT 2: Links

BIT 3: Rechts

BIT 4: Erster Feuerknopf

BIT 5: Zweiter Feuerknopf

BIT 6: Dritter Feuerknopf

BIT 7: 0

ABFRAGE VON CONTROL UND SHIFT

Kurzbeschreibung: Die Tasten CONTROL und SHIFT werden auf ihren Status hin untersucht.

Label: H_CS

ROM-Nummer: A

Startadresse: &C2F7

Einsprungsbedingungen: -

Aussprungsbedingungen:

A = Reihe 2 der Tastaturmatrix

Bit 5: SHIFT

Bit 7: CONTROL

Manipuliert: AF, BC, DE, HL und PIO, PSG

Beschreibung: Um den Status der CONTROL und der SHIFT Taste zu bestimmen, ruft man diese OS Funktion auf. Sie liefert im Akku die gesammte Reihe 2 der Tastaturmatrix.

Die SHIFT-Taste wird durch Bit 5 repräsentiert, und das Bit 7 symbolisiert die CONTROL Taste. Die Bits 0..4 und Bit 6 symbolisieren andere Tasten, und sind somit unbestimmt.

Bitte beachten:

GELEERTES BIT ==> GEDRÜCKTE TASTE

GESETZTES BIT ==> TASTE NICHT GEDRÜCKT

STEUERCODE-TABELLEN INS RAM KOPIEREN

Kurzbeschreibung: Die Steuercode-Tabellen für Mode 1 und 2 werden ins RAM kopiert.

Label: CSTI

ROM-Nummer: A

Startadresse: &C71C

Einsprungsbedingungen: -

Aussprungsbedingungen: 64 Byte Steuercode-Tabellen ab &B800 (MODE 1) bzw. &B900 (MODE 2) wurden im RAM installiert.

Manipuliert: BC, DE, HL und &B800 .. &B83F , &B900 .. &B93F (je incl)

Beschreibung: Im FutureOS existieren für Mode 1 und für Mode 2 getrennte Zeichen-Ausgabe-OS Funktionen. Jeder Mode hat seine eigenen Steuerzeichen (siehe dort!). Für jedes Steuerzeichen existieren 2 Bytes im RAM, die auf die aufzuführende OS Funktion zeigen, so ist es sehr einfach eigene Steuerzeichen einzubinden (siehe gesonderte Doku!).

Ab &B800 bis incl. &B83F befinden sich die Sprungadressen der Mode 1 Zeichenausgabe. Analog befinden sich von &B900 bis &B93F die 32 Sprungadressen für Mode 2.

Als Steuerzeichen sind grundsätzlich die ersten 32 Zeichen definiert, jedes Steuerzeichen hat seine 2-Byte Zieladresse. So werden für jeden Mode 64 Bytes belegt.

Nun, diese OS Funktion macht nichts anderes als die vordefinierten Steuerzeichen-Tabellen aus dem ROM A ins RAM zu kopieren, somit sind den Steuerzeichen definierte Funktionen zugeordnet.

Hat man die Tabellen selber verändert, z.B. um eine Datei mit speziellem Format anzuschauen, dann kann diese OS Funktion die Tabellen wieder restaurieren.

Bitte Beachten: Diese OS Funktion überschreibt alle per Hand geänderten Änderungen in den Steuerzeichen-Tabellen.

AUSGABE EINES ZEICHENS IN MODE 1 MIT PEN 01

Kurzbeschreibung: Ein einzelnes Zeichen wird im Modus 1 an der aktuellen Cursorposition ausgegeben. Zuvor wird die Cursorposition um 2 erhöht. Es wird mit Pen 01 gezeichnet.

Label: PRI0BB

ROM-Nummer: A

Startadresse: &C7B3

Einsprunghbedingungen: Die Cursorposition C_POS muß einen gültigen Wert zwischen &BFFE und &C7FE haben.

Das auszugebende ASCII Zeichen steht in L (0..255).

Aussprunghbedingungen: Die Cursorposition wurde um 2 erhöht, d.h. man kann unmittelbar ein weiteres Zeichen ausgeben. Da in Modus 1 immer 2 Cursorpositionen addiert werden, kommen alle Zeichen die mit dieser OS Funktion ausgegeben werden auch wirklich hintereinander auf den Monitor, ohne Spaces wie man irrtümlicherweise annehmen könnte.

Manipuliert: AF, BC, DE, HL sowie die RAM Variable C_POS.

Beschreibung: Diese OS Funktion dient dazu um ein einzelnes Zeichen auf dem Bildschirm auszugeben. Es muß Bildschirmmodus 1 eingeschaltet werden. Das Zeichen wird mit Pen 01 ausgegeben. Hat man also die Farbpalette nicht verändert, dann erscheint das Zeichen in Blau auf dem Bildschirm. Die RAM Variable C_POS bestimmt an welchem Ort das Zeichen auszugeben ist. Das ein Mode 1 Zeichen im Video RAM 16 Bit breit ist wird auch die Cursorposition C_POS um 2 erhöht. Das hat zur Folge, daß man mit dieser OS Funktion, wenn man sie mehrmals hintereinander aufruft, eine Zeichenfolge ausgeben kann, bei der jedes Zeichen auf das andere folgt. Wenn man zwischendrin jeweils die Cursorposition um 1 erhöht, so hat man automatisch zwischen jedem Zeichen ein halbes SPACE, der Text ist dann nicht mehr so gedrängt.

Bitte Beachten: Die Grafikdaten der einzelnen Zeichen werden ab der Adresse &3800 gelesen. Hier beginnt im Lower ROM das sogenannte Zeichen- oder Charakter ROM, in dem von &3800 bis &3FFF die Grafikdaten für 256 Zeichen stehen. Bevor man ein Zeichen ausgiebt hat man selber dafür Sorge zu tragen daß das untere ROM eingeblendet ist. Ist dies nicht der Fall, dann werden alle Grafikinformatioren eines Zeichens aus dem RAM gelesen. Dies hat aber den Vorteil, daß man ab &3800 im RAM einen eigenen Zeichensatz ablegen kann.

Der Anwender hat vor Aufruf der OS Funktion unbedingt dafür zu sorgen, daß die RAM Variable C_POS einen gültigen Wert von &BFFE bis &C7FE hat. Ist dies nicht der Fall, dann werden die Grafikdaten irgendwo ins RAM geschrieben, und es ist u.U. ein Systemabsturz die Folge.

Diese Einzelzeichenausgabe existiert für die Pens 1..3 in Mode 1. Es können auch Zeichen ausgegeben werden, deren linke Hälfte eine andere Farbe als ihre rechte Hälfte hat. Auf der nächsten Seite sind alle OS Funktionen für die Mode 1 Einzel-Zeichen-Ausgabe aufgelistet.

EINZELZEICHENAUSGABE MODE 1 - EIN- ODER ZWEI-FARBIG

Label	Adresse	Pen linke Zeichenhälfte	Pen rechts Hälfte
PRI0GB	&C73B	Pen 10 (Gelb)	Pen 01 (Blau)
PRI0BB	&C7B3	Pen 01 (Blau)	Pen 01 (Blau)
PRI0GG	&C84B	Pen 10 (Gelb)	Pen 10 (Gelb)
PRI0RR	&C8E3	Pen 11 (Rot)	Pen 11 (Rot)

Die Werte 01, 10 und 11 für Pen sind binär zu verstehen.

Pen 01 bedeutet also Pen 1.

Pen 10 bedeutet also Pen 2.

Pen 11 bedeutet also Pen 3.

Der Zeichensatz ist ein 2 KB Bereich, er beginnt im Lower-ROM bei &3800. Will man einen eigenen Zeichensatz verwenden, dann kann man diesen also dort im RAM ablegen.

AUSGABE EINES EINZELNEN ZEICHENS IN MODE 2

Kurzbeschreibung: Ein einzelnes Zeichen wird in Mode 2 an der aktuellen Cursorposition ausgegeben. Die Cursorposition wird zuvor um eins erhöht.

Label: PR_2

ROM-Nummer: A

Startadresse: &C9BD

Einsprungsbedingungen: Die Cursorposition C_POS muss einen gültigen Wert zwischen &BFFF und &C7FF haben. Das auszugebende ASCII Zeichen steht in L, wobei L hier Werte von 0 bis 255 annehmen kann.

Aussprungsbedingungen: Ein einzelnes Zeichen wurde ausgegeben, die Cursorposition d.h. die Variable C_POS wurde um eins erhöht.

Manipuliert: AF, B, DE, HL und die Ram Variable C_POS

Beschreibung: Diese OS Funktion dient dazu im hochauflösendem Bildschirmmodus 2 ein einzelnes Zeichen auszugeben. Bevor allerdings das Zeichen ausgegeben wird, wird die Cursorposition um eins erhöht. Das Zeichen wird dann an die neue Position ausgegeben.

Bitte Beachten: In diesem Modus ist es möglich mit Zeichenattributen zu arbeiten. Man kann Zeichen mit folgenden Attributen ausgeben:

Label	!! Adresse	!! Attribut und Erklärung.
-----	!!-----	!!-----
PR_2	!! &C9BD	!! Normale Zeichenausgabe, kein Attribut
PR_2D	!! &D016	!! Durchgestrichene Zeichenausgabe (4. Zeile &FF)
PR_2I	!! &D064	!! Invertierte Zeichenausgabe (Pen <==> Paper)
PR_2K	!! &D0BA	!! Kursive Zeichenausgabe (Italics, schräg gestellt)
PR_2U	!! &D113	!! Unterstrichene Zeichenausgabe (8. Zeile &FF)

AUSGABE EINES STRINGS IN MODE 1 (OHNE STEUERZEICHEN)

Kurzbeschreibung: Eine Zeichenfolge von bis zu 65536 Zeichen wird auf dem Bildschirm dargestellt. Alle 256 Zeichen werden als grafisches Symbol dargestellt, auch Steuerzeichen. Es wird Pen 01 benutzt.

Label: STR_BB

ROM-Nummer: A

Startadresse: &CA1C

Einsprungsbedingungen: BC enthält die Anzahl der auszugebenden Zeichen plus 1. HL zeigt auf das erste auszugebende Zeichen. Die anderen Zeichen folgen speicheraufwärts.

Die Ram Variable C_POS, also die Cursorposition muß einen gültigen Wert im Bereich von &BFFE bis &C7FE enthalten. Ist kein eigener RAM Zeichensatz geladen, so sollte man das untere ROM einschalten, da es ab &3800 die Grafikdaten für alle 256 Zeichen enthält.

Aussprungsbedingungen: Die Cursorposition wurde um die Anzahl der ausgegebenen Zeichen erhöht. Man kann also anschließend weitere Zeichen ausgeben, die dann direkt am Ende des zuletzt ausgegebenen Strings angehängt werden.

Manipuliert: AF, BC, HL, AF', BC', DE', HL' und die Ramvariable C_POS.

Beschreibung: Diese OS Funktion dient dazu um eine Zeichenfolge auf dem Bildschirm darzustellen. Diese Zeichenfolge kann bis zu 64K lang sein.

Es können nur Zeichen im Bereich von 0 bis 255 ausgegeben werden, da für jedes Zeichen ein Byte verwendet wird. Alle 256 verschiedenen Zeichen werden als grafische Symbole ausgegeben, d.h. auch Steuerzeichen erscheinen am Bildschirm als grafisches Symbol. Um das Aussehen der Zeichen zu bestimmen, kann entweder das untere ROM eingeblendet werden, das ab &3800 die Grafikdaten aller 256 Zeichen enthält, oder man kann vor dem Aufruf der OS Funktion das untere ROM ausschalten und ab &3800 einen RAM Zeichensatz laden. Will man nun eine Reihe von Bytes auf dem Bildschirm darstellen, so lädt man das 16 Bit Register HL mit der Adresse des ersten Bytes. Alle anderen Bytes folgen darauf mit ansteigenden Adressen. Die Anzahl der Bytes, die ausgegeben werden sollen, schreibt man in das 16 Bit Register BC. Nun ruft man die OS Funktion durch einen CALL auf und die Zeichen erscheinen im Modus 1 auf dem Bildschirm. Es wird mit Pen 01 gezeichnet. Diese OS Funktion sollte nur im Bildschirmmodus 1 benutzt werden.

Bitte Beachten: Ist weder ein RAM Zeichensatz geladen, noch bei Aufruf der OS Funktion das untere ROM eingeschaltet, so kann das zu unvorhersehbaren Effekten auf dem Bildschirm führen, die zwar die Stabilität des Systems NICHT gefährden, aber zur Verwirrung des Users beitragen. Diese OS Funktion ist nur in Mode 1 zu benutzen, der Programmierer hat selber dafür Sorge zu tragen, daß Mode 1 aktiviert ist.

Diese OS Funktion zeichnet in Pen 1, es ist aber auch möglich Pen 2 oder Pen 3 zu verwenden, oder ein Zeichen in 2 Farben auszugeben. Wenn ein Zeichen in zwei Farben dargestellt wird, dann werden die linken vier Pixel in der einen und die rechten 4 Pixel in einer anderen Farbe dargestellt. Auf der folgenden Seite folgt dazu eine Aufschlüsselung.

Siehe auch unter 'AUSGEBE EINES ZEICHENS IN MODE 1' und der Folgeseite.

AUFSCHLÜSSELUNG DER EINZELNEN STRINGOS FUNKTIONEN:

! Label	! Addi.	! Pen links	! Pen rechts
! STR_GB	! &CAC6	! Pen 10	! Pen 01
! STR_BB	! &CA1C	! Pen 01	! Pen 01
! STR_GG	! &CB50	! Pen 10	! Pen 10
! STR_RR	! &CBFA	! Pen 11	! Pen 11

Die Bezeichnung der Pens von 01, 10 und 11 ist binär zu lesen, so ist Pen 01 also Pen 1, Pen 10 ist Pen 2 dezimal und Pen 11 ist dann im Dezimalsystem der Pen 3. Der Grund für diese binäre Bezeichnung liegt darin begründet wie die Grafikdaten im Videoram verteilt sind.

AUSGABE EINES MODE 2 STRINGS OHNE STEUERZEICHEN

Kurzbeschreibung: Eine Zeichenkette von bis zu 65536 Zeichen wird im Bildschirmmodus 2 ausgegeben. Steuerzeichen werden nicht beachtet.

Label: STR_2

ROM-Nummer: A

Startadresse: &CCE0

Einsprungsbedingungen: BC = Anzahl der auszugebende Bytes + 1, also die Länge der Zeichenkette plus 1.

HL = Start der Zeichenkette die auf einmal ausgegeben werden soll.

Die Cursorposition, sprich die RAM Variable C_POS muß einen gültigen Wert im Bereich von &BFFF bis &C7FF enthalten.

Ab &3800 muß ein RAM Zeichensatz vorhanden sein, oder der Programmierer muß vor Aufruf der OS Funktion das untere ROM einblenden.

Der Bildschirmmodus 2 sollte aktiviert sein.

Aussprungsbedingungen: Die Cursorposition wird um die Anzahl der ausgegebenen Bytes (aus Register BC) erhöht, so daß weitere Zeichen ausgegeben werden können.

Manipuliert: AF, BC, AF', BC', DE', HL' und die RAM Variable C_POS.

Beschreibung: Mit dieser OS Funktion ist es möglich im Mode 2 einen String mit beliebiger Länge bis 64K auszugeben. Alle 256 Zeichen, auch alle Steuerzeichen, werden als Grafik auf dem Bildschirm ausgegeben.

Bitte Beachten: Vor Aufruf der OS Funktion muß Mode 2 eingeschalten werden, die Cursorposition muss gesetzt werden, und ab &3800 muß ein Zeichensatz vorhanden sein.

Es ist möglich diese Stringausgabe mit verschiedenen Attributen durchzuführen, siehe auch 'EINZELZEICHENAUSGABE IN MODE 2'.

! Label	! Attribut des Strings	! Adresse
! STR_2	! kein zusätzliches Attribut.	! &CCE0
! STR_2D	! Alle Zeichen durchgestrichen.	! &D161
! STR_2I	! Alle Zeichen invertiert.	! &D1C2
! STR_2K	! Alle Zeichen kursiv.	! &D22A
! STR_2U	! Alle Zeichen unterstrichen.	! &D296

AUSGABE EINES STRINGS MIT STEUERZEICHEN IN MODE 1

Kurzbeschreibung: Eine Zeichenkette beliebiger Länge wird unter Beachtung aller Steuerzeichen in Mode 1 mit Pen 01 ausgegeben.

Label: TER_BB

ROM-Nummer: A

Startadresse: &CD4C

Einsprunghbedingungen: HL = Zeiger auf die Startadresse des auszugebenden Strings, wobei das Stringende durch ein Null Byte festgesetzt wird. Die Cursorposition C_POS muß einen gültigen Wert im Bereich von &BFFE bis &C7FE enthalten, ferner muß das untere ROM eingeschalten sein bzw. ein RAM Zeichensatz geladen sein.

Aussprunghbedingungen: -

Manipuliert: AF, DE, HL, AF', BC', DE', HL' und die Cursorposition.

Beschreibung: Mit dieser OS Funktion ist es möglich eine Zeichenkette in Mode 1 auszugeben. Es wird mit Pen 01 gezeichnet. Alle Steuerzeichen werden beachtet. Bei Aufruf der OS Funktion zeigt HL auf das erste Byte des Strings. Ab dieser Adresse werden alle Bytes gelesen, und entweder als Grafikzeichen ausgegeben (32...255) oder als Steuerzeichen interpretiert. Als Steuerzeichen werden alle Zeichen im Bereich von 0..31 benutzt. Das Ende der Zeichenkette wird durch ein Nullbyte, also Steuerzeichen 00 interpretiert. Die Grafikdaten werden ab &3800 gelesen, wobei die ersten 32 Zeichen wiegesagt nicht als Grafikzeichen ausgegeben werden, sondern als Steuerfunktion interpretiert werden.

Bitte Beachten: Die Cursorposition muß vor Aufruf der OS Funktion einen Wert im Bereich von &BFFE bis &C7FE bekommen. Der Programmierer hat dafür zu sorgen, ob ab &3800 ein RAM Zeichensatz geladen ist, oder ob das untere ROM eingeblendet ist.

Wie auch bei der Stringausgabe ohne Steuerzeichen können die auszugebenden Zeichen in Pen 1, 2, 3 oder zweifarbig gezeichnet werden.

```
!-----!  
! Label ! Addi. ! linker Pen ! rechter Pen !  
!-----!  
! TER_GB ! &CDF6 ! Pen 10 ! Pen 01 !  
!-----!  
! TER_BB ! &CD4C ! Pen 01 ! Pen 01 !  
!-----!  
! TER_GG ! &CE80 ! Pen 10 ! Pen 10 !  
!-----!  
! TER_RR ! &CF2A ! Pen 11 ! Pen 11 !  
!-----!
```

Wobei 01 für Pen 1 und 10 für Pen 2 steht. Und 11 steht für Pen 3.

AUSGABE EINES MODE 2 STRINGS MIT STEUERZEICHEN

Kurzbeschreibung: Eine Zeichenkette wird in Mode 2 und unter Beachtung aller Steuerzeichen ausgegeben.

Label: TERM_2

ROM-Nummer: A

Startadresse: &D48C

Einsprungsbedingungen: HL = erstes Byte der auszugebenden Zeichenkette.
Die Cursorposition C_POS muß auf einen Wert zwischen &BFFF und &C7FF gesetzt werden.
Ab &3800 muß ein Zeichensatz vorhanden sein.

Aussprungsbedingungen: -

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL' und die Cursorposition.

Beschreibung: Im Modus 2 wird eine Zeichenkette ausgegeben. Alle 32 Steuerzeichen (0..31) werden auch als solchen interpretiert. Bei Aufruf der OS Funktion muß das Doppelregister HL die Startadresse des Strings enthalten. Ab &3800 muß ein RAM Zeichensatz eingeblendet sein bzw. das untere ROM muß eingeblendet sein. Sobald die OS Funktion auf ein Nullbyte trifft wird die Textausgabe abgebrochen.

Bitte Beachten: Die Cursorposition, der ROM Status und der Zeichensatz müssen geprüft sein.

Im Modus 2 ist es möglich einen String mit verschiedenen Attributen auszugeben. Siehe dazu auch 'AUSGABE EINES ZEICHENS IN MODE 2'

Label	!	Addi.	!	Attribut des Strings.
TERM_2	!	&D48C	!	kein weiteres Attribut.
TERM_2D	!	&D2F7	!	Alle Zeichen durchgestrichen.
TERM_2I	!	&D358	!	Alle Zeichen invertiert.
TERM_2K	!	&D3C0	!	Alle Zeichen kursiv.
TERM_2U	!	&D42C	!	Alle Zeichen unterstrichen.

SETZE BILDSCHIRM AUF 64 SPALTEN UND 32 ZEILEN

Kurzbeschreibung: Das Format des Bildschirms wird auf 64 Spalten und 32 Zeilen gesetzt.

Label: S64X32

ROM-Nummer: A und D

Startadresse: &D5A8 (logisches ROM A) und &E6B5 (logisches ROM D)

Einsprungsbedingungen: -

Aussprungsbedingungen: MAX_CRX, MAX_CRY sind angeglichen.

Manipuliert: Flags, BC, DE, HL, CRTC Register 1, 2, 4, 6 und 7.

Beschreibung: Unabhängig vom bisherigen Bildschirmformat wird der Bildschirm auf 64 Mode 2 Spalten und 32 Zeilen gesetzt. Dabei wird aber weder der Bildschirmmodus geändert, noch der Bildschirm gelöscht.

SETZE BILDSCHIRM AUF 68 SPALTEN UND 30 ZEILEN

Kurzbeschreibung: Das Format des Bildschirms wird auf 68 Spalten und 30 Zeilen gesetzt.

Label: S68X30

ROM-Nummer: A und D

Startadresse: &D5DB

Einsprungsbedingungen: -

Aussprungsbedingungen: MAX_CRX, MAX_CRY sind angeglichen.

Manipuliert: Flags, BC, DE, CRTC Register 1, 2, 4, 6 und 7

Beschreibung: Das Bildschirmformat wird auf 68 Mode 2 Spalten und 32 Zeilen gesetzt.

SETZE BILDSCHIRM AUF 80 SPALTEN UND 25 ZEILEN

Kurzbeschreibung: Das Format des Bildschirms wird auf 80 Spalten und 25 Zeilen gesetzt.

Label: S80X25

ROM-Nummer: A und D

Startadresse: &D60E (logisches ROM A), &E6E7 (logisches ROM D)

Einsprunghbedingungen: -

Aussprunghbedingungen: MAX_CRX, MAX_CRY sind angeglichen.

Manipuliert: AF, BC, DE, CRTC Register 1, 2, 4, 6 und 7.

Beschreibung: Das Bildschirmformat wird auf 80 Mode 2 Zeichen und 25 Zeilen gesetzt.

Bitte Beachten: Der Bildschirmmodus wird nicht verändert.

Darstellung EINES SPEICHERAUSZUGES (DUMP) im 68 * 30 FORMAT MODE 2

Kurzbeschreibung: 480 Bytes Speicher werden in MODE 2 im Bildschirm-Format 68 Zeichen auf 30 Zeilen dargestellt. Die Ausgabe erfolgt sowohl in HEX als auch in ASCII.

Label: F_DUMP

ROM-Nummer: A

Startadresse: &FE77

Einsprunghbedingungen: HL = Startadresse des Dumps. Ab dieser Adresse werden 480 Bytes in HEX und FSCW / ASCII ausgegeben.

Vor Aufruf der OS Funktion ist der Bildschirm auf das 68 * 30 Format und Mode 2 zu schalten.

Aussprunghbedingungen: Das untere ROM wurde eingeblendet (Status &7F82)

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL' und die RAM Variablen REG08_0, REG16_0 und C_POS. Außerdem wurde der 2 KB Textbildschirm benutzt. Und das untere ROM wurde eingeblendet (Status &7F82).

Beschreibung: Diese OS Funktion dient dazu um 480 Bytes, aus dem gerade eingeblendeten Speicher, darzustellen.

Es wird Bildschirmmodus 2 vorausgesetzt. MODE 2 muß man vor Aufruf der OS Funktion selber aktivieren (die Befehle LD BC,&7F82 : OUT (C),C erledigen das). Und das Bildschirmformat muss auf 68 Zeichen pro Zeile und auf 30 Zeilen gesetzt werden (z.B. mittels OS Funktion S68X30). Das 16 Bit Register HL lädt man mit der Startadresse des Speicherblocks den man anzeigen will. Der Speicher wird folgendermaßen dargestellt.

In jeder Zeile steht ganz links eine 16 Bit Adresse die sich auf die nebenstehenden 16 Bytes bezieht. In jeder Zeile werden also 16 Bytes dargestellt, und zwar zuerst im Hexadezimalsystem, und dann in FSCW Code. Der FSCW Code entspricht hier dem ASCII Code. Diese OS Funktion ist ein Teil des Maschinensprache Monitors. Nach dem Ende dieser OS Funktion ist das untere ROM eingeblendet!!!

Achtung: Die aufrufende OS Funktion muss sich also oberhalb der Adresse &4000 befinden, sonst erfolgt ein Rücksprung ins untere ROM, also ein Crash!

Bitte Beachten: Die OS Funktion benutzt den Textbildschirm, d. h. alle zuvor darin enthaltenen Daten sind verloren. Der Programmierer hat vor Aufruf der OS Funktion selber den Mode 2 und das 68 * 30 Format zu aktivieren. Diese OS Funktion darf nur von einer Adresse größer als &4000 aus aufgerufen werden, da sie vor dem Rücksprung das untere ROM einblendet und somit ein Rücksprung ins RAM unterhalb von &4000 unmöglich ist.

16 BIT EIN- UND AUSGABE ALLER PORTADRESSEN DES CPC

Kurzbeschreibung: Anzeigen und editieren aller Ports des CPC.

Label: F_PORT

ROM-Nummer: A

Startadresse: &FE7A

Einsprunghbedingungen: –

Aussprunghbedingungen: Die vom Anwender veränderten Werte wurden an die entsprechenden I/O Ports geschickt.

Beide ROM's sind eingeschalten, Mode 2 und das 64 * 32 Format sind aktiviert.

Manipuliert: AF, BC, DE, HL, IX, YH, AF', BC', DE', HL' und die RAM Variablen REG16_0, REG16_1 und C_POS.

Der RAM / ROM Status der Gate Arrays und der Bildschirmmodus sind verändert. Das Bildschirmformat wird auf 64 Spalten und 32 Zeilen gesetzt.

Beschreibung: Diese OS Funktion ist ein Teil des Maschinensprache Monitors. Da vor Aufruf der OS Funktion im RAM das CUR_INV Programm vorhanden sein muß sollte man diese OS Funktion mit Vorsicht benutzen. Sie dient dazu, an die verschiedensten 16 Bit I/O Adressen den CPC Werte zu versenden, oder diese einzulesen. Die OS Funktion wird mit den Cursortasten und Copy bedient. Mit den Cursortasten wählt man die gewünschte Ein- oder Ausgabeadresse. Will man einen Wert einlesen, so fährt man den Cursor auf die gewünschte Adresse und betätigt Copy sofort wird der Wert des Ports eingelesen und auf dem Bildschirm ausgegeben. Will man einen Wert ausgeben, so fährt man zuerst mit dem Cursor die entsprechende Ausgabeadresse an, dann drückt man Copy, nun tippt man einen hexadezimalen Wert im Bereich von 00 bis FF ein und drückt anschließend Return. Es gibt auch die Möglichkeit das High und Lowbyte einer beliebigen I/O Adresse einzugeben. Will man die OS Funktion werlaßen, so wählt man eine Ausgabefunktion an, und drückt die ESC Taste anstatt einen Wert einzugeben.

Bitte Beachten: Es kann zu Systemabstürzen führen, wenn man an bestimmt Ports Werte ausgiebt, oder von bestimmten Ports Werte einließt. Nur der Assembler-Profi sollte dieses imens mächtige Instrument bedienen. Der Anfänger könnte erheblichen Schaden anrichten. Diese OS Funktion arbeitet mit abgeschalteten Interrupts.

EINGABEFUNKTION FÜR 8 BIT WERTE MIT DARSTELLUNG AUF BILDSCHIRM

Kurzbeschreibung: Eingabe-Funktion für Werte zwischen 0 und 255.

Label: RB_8 bzw. RBB_8

ROM-Nummer: A

Startadresse: &FE3B (RB_8) bzw. &FE35 (RBB_8)

Einsprungsbedingungen: Die Cursorposition C_POS (RB_8) bzw. ein Zeichen vor der Cursorposition C_POS-1 (RBB_8) wird als Eingabestelle auf dem Bildschirm benutzt.

Aussprungsbedingungen: A enthält den vom Anwender eingegebenen Wert und das Carry Flag ist auf 1 gesetzt.

Kehrt die OS Funktion mit gelöschttem Carry-Flag(NC) zurück, dann wurde die Eingabe mittels ESC abgebrochen.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, IY, PIO, PSG, und die System RAM Variable C_POS, die die Cursorposition enthält.

Beschreibung: Diese beiden OS Funktionen stellen Eingabemöglichkeiten für 8 Bit Werte (0-255) in MODE 2 dar. Dabei wird entweder die aktuelle Cursorposition benutzt (OS Funktion RB_8) oder es wird eine Stelle weiter links benutzt (OS Funktion RBB_8). Im zweiten Fall wird also das zu letzt eingegeben Zeichen überschrieben, allerdings nur im Dezimal-System. Dadurch kann ein eventuelles "&" Zeichen für die Eingabe von hexadezimalen Werten automatisch überschrieben werden.

Beide OS Funktionen RB_8 und RBB_8 springen entweder zu BIT8_IN oder B8DIN, abhängig davon ob FutureOS im Dezimal- oder Hexadezimal-System arbeitet. Bitte siehe auch dort!

Die Eingabe wird entweder mit RETURN abgeschlossen oder mit ESC abgebrochen. Bei Abbruch (ESC) kehrt die OS Funktion mit gelöschttem Carry Flag zurück. Wurde die Eingabe mit RETURN beendet, dann ist das Carry Flag gesetzt.

Bitte Beachten: Die OS Funktion kehrt erst dann zurück, wenn entweder ein RETURN bzw. ein ESC vom Anwender eingegeben wurde.

Bitte benutzen sie diese OS Funktion nur im Bildschirm MODE 2.

EingabeFunktion für 8 BIT WERTE (HEX) mit DARSTELLUNG auf Bildschirm

Kurzbeschreibung: Eingabe-Funktion für hexadezimale 8 Bit Werte. Dabei bricht die ESC Taste die Eingabe ab.

Label: BIT8_IN

ROM-Nummer: A

Startadresse: &C31A

Einsprunghbedingungen: Die Cursorposition C_POS wird als Eingabestelle auf dem Bildschirm benutzt.

Aussprunghbedingungen: A = der vom Anwender eingegebene 8 Bit Wert.

Kehrt die OS Funktion mit gelöschtem Carry-Flag(NC) zurück, dann wurde die Eingabe mittels ESC abgebrochen.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IY, PIO, PSG und die Variable C_POS, die die Cursorposition darstellt.

Beschreibung: Diese OS Funktion stellt eine Eingabemöglichkeit für 8 Bit Werte in MODE 2 dar. Es wird die aktuelle Cursorposition benutzt.

Die Eingabe findet im hexadezimalen Zahlensystem statt. Nach Aufruf der OS Funktion werden vom Anwender zwei Werte verlangt. Zuerst die höherwertigen vier Bit, dann die niederwertigen vier Bit. Es können Werte d.h. Tasten im Bereich von 0 bis 9 und a bis f. benutzt werden.

Shift und Control werden nicht abgefragt. Mit DEL kann man den zuvor eingegebenen Wert löschen. Die Eingabe wird mit RETURN abgeschlossen.

Abgesehen davon ist es möglich die Eingabe mit Hilfe der ESC Taste abubrechen, die OS Funktion kehrt dann mit gelöschtem Carry Flag zurück. Wurde die Eingabe mit RETURN beendet, dann ist das Carry Flag gesetzt. Die ESC Taste ist allerdings wirkungslos, wenn schon beide Werte eingegeben sind, dann ist zuerst einmal DEL zu drücken und anschließend die ESC Taste.

Bitte Beachten: Die OS Funktion kehrt erst dann zurück, wenn entweder zwei Werte und ein RETURN bzw. ein ESC vom Anwender eingegeben wurden.

Andernfalls wartet das System bis in die Unendlichkeit oder bis zum nächsten Strohmausfall.

Bitte benutzen sie diese OS Funktion nur im Bildschirm MODUS 2.

Diese OS Funktion wird von der F_PORT Monitorfunktion benutzt.

Eingabefunktion für 8 BIT WERTE (DEZ) mit DARSTELLUNG auf Bildschirm

Kurzbeschreibung: Inputfunktion für dezimale 8 Bit Werte. Dabei bricht die ESC Taste die Eingabe ab.

Label: B8DIN

ROM-Nummer: A

Startadresse: &FE41

Einsprungsbedingungen: Die Cursorposition C_POS wird als Eingabestelle auf dem Bildschirm benutzt.

Aussprungsbedingungen: A = der vom Anwender eingegebene 8 Bit Wert.

Kehrt die OS Funktion mit gelöschtem Carry-Flag(NC) zurück, dann wurde die Eingabe mittels ESC abgebrochen.

Manipuliert: AF, BC, DE, HL, IX, IY, BC', DE', HL', PIO, PSG und die Variable C_POS, die die Cursorposition darstellt.

Beschreibung: Diese OS Funktion stellt eine Eingabemöglichkeit für 8 Bit Werte in MODE 2 dar. Es wird die aktuelle Cursorposition benutzt.

Die Eingabe findet im dezimalen Zahlensystem statt. Nach Aufruf der OS Funktion werden vom Anwender drei Werte verlangt. Zuerst Hunderter, dann Zehner, dann Einer. Es können Werte d. h. Tasten im Bereich von 0 bis 9 benutzt werden. Shift und Control werden nicht abgefragt. Mit DEL kann man den zuvor eingegebenen Wert löschen. Die Eingabe wird mit RETURN abgeschlossen.

Abgesehen davon ist es möglich die Eingabe mit Hilfe der ESC Taste abubrechen, die OS Funktion kehrt dann mit gelöschtem Carry Flag zurück. Wurde die Eingabe mit RETURN beendet, dann ist das Carry Flag gesetzt.

Bitte Beachten: Die OS Funktion kehrt erst dann zurück, wenn entweder RETURN bzw. ESC vom Anwender gedrückt wurden.

Bitte benutzen sie diese OS Funktion nur im Bildschirm MODUS 2.

Eingabefunktion für 16 BIT WERTE mit DARSTELLUNG auf BILDSCHIRM

Kurzbeschreibung: Eingabe-Funktion für Werte zwischen 0 und 65535.

Label: RB_16 bzw. RBB_16

ROM-Nummer: A

Startadresse: &FE3E (RB_16) bzw. &FE38 (RBB_16)

Einsprungsbedingungen: Die Cursorposition C_POS(RB_16) bzw. ein Zeichen vor der Cursorposition C_POS-1 (RBB_16) wird als Eingabestelle auf dem Bildschirm benutzt.

Aussprungsbedingungen: DE' enthält den vom Anwender eingegebenen 16 Bit Wert und das Carry Flag ist auf 1 gesetzt.

Kehrt die OS Funktion mit gelöschttem Carry-Flag(NC) zurück, dann wurde die Eingabe mittels ESC abgebrochen.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, IY, PIO, PSG, und die System RAM Variable C_POS, die die Cursorposition enthält.

Beschreibung: Diese beiden OS Funktionen stellen Eingabemöglichkeiten für 16 Bit Werte (0-65535) in MODE 2 dar. Dabei wird entweder die aktuelle Cursorposition benutzt (OS Funktion RB_16) oder es wird eine Stelle weiter links benutzt (OS Funktion RBB_16). Im zweiten Fall wird also das zu letzt eingegeben Zeichen überschrieben, allerdings nur im Dezimal-System. Dadurch kann ein eventuelles "&" Zeichen für die Eingabe von hexadezimalen Werten automatisch überschrieben werden.

Beide OS Funktionen RB_16 und RBB_16 springen entweder zu B16IN oder B16DIN, abhängig davon ob FutureOS im Dezimal- oder Hexadezimal-System arbeitet. Bitte siehe auch dort!

Die Eingabe wird entweder mit RETURN abgeschlossen oder mit ESC abgebrochen. Bei Abbruch (ESC) kehrt die OS Funktion mit gelöschttem Carry Flag zurück. Wurde die Eingabe mit RETURN beendet, dann ist das Carry Flag gesetzt.

Bitte Beachten: Die OS Funktion kehrt erst dann zurück, wenn entweder ein RETURN bzw. ein ESC vom Anwender eingegeben wurde.

Bitte benutzen sie diese OS Funktion nur im Bildschirm MODUS 2.

Eingabefunktion für 16 BIT WERTE (HEX) mit Darstellung auf Bildschirm

Kurzbeschreibung: Eingabe-Funktion für hexadezimale 16 Bit Werte.

Dabei bricht die ESC Taste die Eingabe ab.

Label: B16IN

ROM-Nummer: A

Startadresse: &C38A

Einsprungsbedingungen: Die Cursorposition C_POS wird als Eingabestelle auf dem Bildschirm benutzt.

Aussprungsbedingungen: DE' = der vom Anwender eingegebene 16 Bit Wert.

Kehrt die OS Funktion mit gelöschttem Carry-Flag(NC) zurück, dann wurde die Eingabe mittels ESC abgebrochen.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL' und die Cursorposition C_POS.

Beschreibung: Diese OS Funktion stellt eine Eingabemöglichkeit für 16 Bit Werte in MODE 2 dar. Es wird die aktuelle Cursorposition benutzt.

Die Eingabe findet im hexadezimalen Zahlensystem statt. Nach Aufruf der OS Funktion werden vom Anwender vier Werte verlangt. Zuerst die höherwertigen, dann die niederwertigen Bytes. Es können Werte d.h. Tasten im Bereich von 0 bis 9 und a bis f. benutzt werden. Shift und Control werden nicht abgefragt. Mit DEL kann man den zuvor eingegebenen Wert löschen. Die Eingabe wird mit RETURN abgeschlossen.

Abgesehen davon ist es möglich die Eingabe mit Hilfe der ESC Taste abubrechen, die OS Funktion kehrt dann mit gelöschttem Carry Flag zurück. Wurde die Eingabe mit RETURN beendet, dann ist das Carry Flag gesetzt. Die ESC Taste ist allerdings wirkungslos, wenn schon alle vier Werte eingegeben sind, dann ist zuerst einmal DEL zu drücken und anschließend die ESC Taste.

Bitte Beachten: Die OS Funktion kehrt erst dann zurück, wenn entweder vier Werte und ein RETURN bzw. ein ESC vom Anwender eingegeben wurden.

Ein anderer Ausstieg ist nicht möglich.

Bitte benutzen sie diese OS Funktion nur im Bildschirm MODUS 2.

Diese OS Funktion wird vom FutureOS Monitor benutzt.

Eingabefunktion für 16 BIT WERTE (DEZ) mit Darstellung auf Bildschirm

Kurzbeschreibung: Eingabe-Funktion für dezimale 16 Bit Werte. Dabei bricht die ESC Taste die Eingabe ab.

Label: B16DIN

ROM-Nummer: A

Startadresse: &FE44

Einsprunghbedingungen: Die Cursorposition C_POS wird als Eingabestelle auf dem Bildschirm benutzt.

Aussprunghbedingungen: DE' = der vom Anwender eingegebene 16 Bit Wert.

Kehrt die OS Funktion mit gelöschttem Carry-Flag(NC) zurück, dann wurde die Eingabe mittels ESC abgebrochen.

Manipuliert: AF, BC, DE, HL, BC', DE', HL', IX, IY, PIO, PSG und die Cursorposition C_POS.

Beschreibung: Diese OS Funktion stellt eine Eingabemöglichkeit für 16 Bit Werte in MODE 2 dar. Es wird die aktuelle Cursorposition benutzt.

Die Eingabe findet im dezimalen Zahlensystem statt. Nach Aufruf der OS Funktion werden vom Anwender fünf Werte verlangt: Zehntausender, Tausender, Hunderter, Zehner und Einer. Es können Werte d. h. Tasten im Bereich von 0 bis 9 benutzt werden. Shift und Control werden nicht abgefragt. Mit DEL kann man den zuvor eingegebenen Wert löschen. Die Eingabe wird mit RETURN abgeschlossen.

Abgesehen davon ist es möglich die Eingabe mit Hilfe der ESC Taste abubrechen, die OS Funktion kehrt dann mit gelöschttem Carry Flag zurück. Wurde die Eingabe mit RETURN beendet, dann ist das Carry Flag gesetzt.

Bitte Beachten: Die OS Funktion kehrt erst dann zurück, wenn entweder RETURN bzw. ein ESC vom Anwender eingegeben wurden. Ein anderer Ausstieg ist nicht möglich.

Bitte benutzen sie diese OS Funktion nur im Bildschirm MODUS 2.

UMWANDLUNG EINES BYTES IN ZWEI ASCII ZEICHEN

Kurzbeschreibung: Eine einzelnes Byte wird in zwei ASCII Zeichen umgewandelt. Diese kann man später auf dem Bildschirm ausgeben, dadurch erscheint das Byte hexadezimal mit zwei Stellen.

Label: N_2_2C

ROM-Nummer: A

Startadresse: &D58A

Einsprungsbedingungen: A = enthält das umzuwandelnde Quellbyte.

DE = Zeiger auf einen zwei Bytes langen Bereich in dem die beiden zu generierenden ASCII Zeichen abgelegt werden sollen.

Aussprungsbedingungen: DE wurde um 1 erhöht. In (DE) steht das untere, in (DE-1) das höhere Nibble des Quellbytes.

Der Akku enthält das niederwertige Nibble.

Manipuliert: AF, BC und DE wurde um 1 erhöht.

Beschreibung: Will man ein beliebiges Byte von &00 bis &FF auf dem Bildschirm im hexadezimalen System darstellen, so ist es zuerst in zwei ASCII Zeichen umzuwandeln. Genau dies macht diese OS Funktion, dabei wird in die durch DE adressierte Speicherstelle das obere Nibble des Quellbytes geschrieben. In die darauffolgende Adresse wird das untere Nibble geschrieben. Die zwei generierten ASCII Zeichen können sofort als String ausgegeben werden, und es erscheint auf dem Bildschirm das Quellbyte, für jedermann lesbar. Diese OS Funktion gibt aber selber keine Zeichen auf dem Bildschirm aus.

Bitte Beachten: Wird DE vor Aufruf der OS Funktion nicht mit einem gültigem Wert geladen, so werden irgendwo ins RAM zwei Bytes geschrieben. Dies kann unabsehbare Folgen nach sich ziehen.

ANZEIGEN & EDITIEREN EINES STRINGS

Kurzbeschreibung: Mittels STED läßt sich ein String anzeigen und editieren, oder neu generieren.

Label: STED

ROM-Nummer: A

Startadresse: &FE7D

Einsprungsbedingungen:

B = Obergrenze, 1. Zeichen OBERHALB des legalen Bereichs, von &01 bis &FF.

C = Untergrenze, kleinstes legales Zeichen, von &00 bis &FF.

DE = Zeiger auf 1. Byte des zu editierenden Strings, von &0000 bis &FFFF.

HL = Zeiger auf Byte NACH dem Stringende, von &0000 bis &FFFF.

Aussprungsbedingungen: String wurde durch den Anwender editiert.

A = &03 (=0000 0011) ==> Abbruch durch ESC!

A = &0E (=0000 1110) ==> Eingabe korrekt beendet.

Außerdem wurden der Hauptspeicher (RAM-Konfiguration &C0) und beide ROMs eingeblendet. Bildschirm Mode 2 wurde aktiv geschaltet.

Manipuliert: AF, BC, DE, HL, BC', DE', HL', IX, IY, RAM- und ROM-Status sowie PIO und PSG.

Beschreibung: Mittels STED wird ein bestehender String editiert, oder ein neuer generiert. Die Stringlänge ist nur vom Bildschirmformat abhängig. STED läuft in MODE 2.

Vor Aufruf der OS Funktion übergibt man in DE die Adresse des ersten Bytes des Strings, in HL wird die Adresse des Bytes übergeben, daß oberhalb des Strings folgt. Die Stringlänge entspricht also HL - DE.

Außerdem kann man der OS Funktion angeben welcher Zeichenbereich überhaupt zulässig ist. In B wird der Wert des ASCII Zeichens angegeben, daß oberhalb des legalen Bereichs liegt. In C wird das erste legale Zeichen angegeben. Beispiel: B = &80 und C = &20, dann dürfen alle ASCII Zeichen von &20 bis &7F verwendet werden.

Während des Editierens kann der Anwender durch die vier Pfeiltasten und DEL den Cursor bewegen. Der Editiervorgang wird durch RETURN oder ENTER abgeschlossen.

Man kann die Editierung auch durch ESC abbrechen. Beim Aussprung wird in A das Byte &0E (o.k.) bzw. &03 (ESC) übergeben. Druch folgende Zeilen kann dies getestet werden.

Beispiel:

```
LD BC,&8020      ;Zeichen von &20 bis &7F sind legal
LD DC,&8400      ;Editieren von &8400
LD HL,&8420 + 1  ;bis &8420
CALL STED       ;String editieren
RRCA            ;Bit 0 ins Carry
JR C,ESC        ;Bit 0 ist "1", Editierung durch ESC abgebrochen.
```

Bitte Beachten: Der String darf nicht länger sein, als dies das Bildschirmformat zuläßt. Bei einem Bildschirmformat von 64 Zeichen auf 32 Zeilen darf der String nicht länger als knapp 2KB sein. Dieses Unterprogramm ist nur in MODE 2 lauffähig.

Bei Aufruf sollte HL größer DE sein.

INVERTIEREN VON 2 MODE 2 ZEICHEN (MODE 1 CURSOR)

Kurzbeschreibung: Zwei aufeinander folgende Mode 2 Zeichen werden invertiert.

Label: CUR_INV

ROM-Nummer: A

Startadresse: &D4EC

Einsprungsbedingungen: C_POS ==> Zeichen-Pos. ab der invertiert wird.

Aussprungsbedingungen: Zwei Mode 2 Zeichen invertiert. Außerdem wurden die ersten 64K eingeblendet, RAM-Konfiguration &C0. Auch wurden beide ROMs eingeblendet. Es wurde Bildschirm Mode 2 aktiviert.

Manipuliert: AF, BC, HL, 16 Bytes V-RAM und der RAM/ROM-Status.

Beschreibung: Diese OS Funktion dient dazu einen Cursor darzustellen, oder bessergesagt zu invertieren. Dabei handelt es entweder um ein Mode 1 Zeichen oder um zwei aufeinanderfolgende Mode 2 Zeichen (16 Bit).

Durch einen zweiten Aufruf dieser OS Funktion wird ihr Effekt kompensiert, da die betroffenen Bits nur invertiert werden.

Die OS Funktion ist extrem schnell, blendet aber beim Rücksprung die RAM-Konfiguration &C0 ein, also die ersten 64K. Außerdem werden beide ROMs (das obere und das untere) eingeblendet. Es ist Mode 2 aktiv.

Bitte Beachten: Es werden die ersten 64K eingeblendet.

INVERTIEREN VON 1 MODE 2 ZEICHEN (MODE 2 CURSOR)

Kurzbeschreibung: Ein Mode 2 Zeichen wird invertiert.

Label: CUR_IV2

ROM-Nummer: A

Startadresse: &D544

Einsprungbedingungen: C_POS ==> Zeichen-Pos. ab der invertiert wird.

Aussprungbedingungen: Ein Mode 2 Zeichen invertiert. Außerdem wurden die ersten 64K eingeblendet (RAM-Konfiguration &C0) und beide ROMs eingeblendet. Es wurde Bildschirm Mode 2 aktiviert.

Manipuliert: AF, BC, HL, 8 Bytes V-RAM und der RAM/ROM-Status.

Beschreibung: Diese OS Funktion dient dazu einen Cursor darzustellen, oder bessergesagt zu invertieren. Dabei handelt es um ein Mode 2 Zeichen (8 Bit breit).

Durch einen zweiten Aufruf dieser OS Funktion wird ihr Effekt kompensiert, da die betroffenen Bits nur invertiert werden.

Die OS Funktion ist extrem schnell, blendet aber beim Rücksprung die RAM-Konfiguration &C0 ein, also die ersten 64K. Außerdem werden beide ROMs (das obere und das untere) eingeblendet. Es ist Mode 2 aktiv.

Bitte Beachten: Es werden die Standard 64K (Konfig &C0) eingeblendet.

KOPFZEILE/HEADER EINER FUTURE-OS DATEI ANZEIGEN

Kurzbeschreibung: Die Kopfzeile bzw. das Icon eines 128 Byte Headers einer FutureOS Datei wird angezeigt.

Label: SHED

ROM-Nummer: A

Startadresse: &FE71

Einsprungsbedingungen: Die RAM-Variable C_POS bestimmt wo die Daten angezeigt werden sollen. Der Bildschirm ist auf MODE 1 und 32*32 Zeichen zu schalten, u.U. ist der Bildschirm zu löschen.

Ein FutureOS Dateihheader muß ab &B400 im RAM liegen.

Aussprungsbedingungen: Kopfzeile, Semigrafik- oder Grafik Icon wurde auf dem Bildschirm dargestellt.

Manipuliert: AF, BC, HL, IX, IY und das Video-RAM ab C_POS.

Beschreibung: Unter AmsDOS hat jede NICHT-ASCII Datei einen 128 Bytes Dateihheader, dieser Dateikopf ist unter FutureOS stark erweitert worden. Der Dateikopf gibt zusätzliche Informationen über den Inhalt einer Datei.

Diese OS Funktion gibt lediglich diese FutureOS spezifischen Zusatz-Informationen am Bildschirm aus. Siehe Einsprung-Bedingungen!

Dabei handelt es sich entweder um eine Kopfzeile, ein Semigrafik-Icon oder eine echtes Grafikicon. Siehe dazu auch gesonderte Information.

Bitte Beachten: Ein FutureOS Dateihheader muß ab &B400 im RAM liegen.
Vor Aufruf der OS Funktion sollte MODE 1 eingestellt werden, und der Bildschirm auf 32 mal 32 MODE 1 Zeichen eingestellt werden.

ECHTZEITUHR-VERWALTUNG INS RAM KOPIEREN

Kurzbeschreibung: Kopiere Code der Echtzeituhr-Verwaltung ins RAM.

Label: DOBIN

ROM-Nummer: A

Startadresse: &FE6E

Einsprungsbedingungen: -

Aussprungsbedingungen: Der Smart-Code wurde ab LUHR, RDUK und KOAS ins RAM kopiert. Die Smart-Watch kann nun abgefragt werden.

Manipuliert: BC, DE, HL und die Flags.

Beschreibung: Die Dobbertin bzw. dxs Echtzeituhr wird am CPC wie ein ROM angeschlossen. Um Daten von ihr lesen zu können, muß die von der Uhr belegte ROM-Nummer eingeblendet werden. Ist die Uhr eingeblendet, dann ist das FutureOS natürlich ausgeblendet. Deshalb sind OS Funktionen erforderlich die zuerst die Uhr einblenden, die Zeit-Daten lesen, und dann wieder das Future OS einblenden.

Für die Echtzeituhr der M4 Erweiterung verhält es sich ähnlich, auch hier wird Zeit und Datum aus M4 ROM gelesen.

OS Funktionen zum lesen oder schreiben von Zeit und Datum müssen also im RAM vorhanden sein, und dort müssen sie zuerst mal hinkopiert werden. Dies ist die Aufgabe von DOBIN. DOBIN kopiert die entsprechenden OS Funktionen ins RAM. Normalerweise wird DOBIN nach jedem Systemstart aufgerufen, die Uhrverwaltung sollte sich also immer im RAM befinden.

Sollte ein Programm aber das entsprechende RAM überschreiben, was tunlichst zu vermeiden ist, dann können die Funktionen zur Verwaltung von Echtzeituhren wieder hergestellt werden – eben mit DOBIN.

Um die Daten der Echtzeit-Uhr zu lesen verwendet man die OS Funktion LUHR, sie liest Zeit und Datum in die dafür vorgesehenen Bytes ab UHR_00.

Sollte eine M4 Karte angeschlossen sein, so wird die M4 Echtzeituhr benutzt, selbst wenn die Dobbertin / dxs RTC zusätzlich angeschlossen ist.

Bitte Beachten: Es werden einige OS Funktionen ins RAM kopiert. Die Uhr wird jedoch mittels LUHR gelesen.

ASCII-ZEICHEN 8-FACH GEZOOMT DARSTELLEN

Kurzbeschreibung: Ein ASCII-Zeichen wird, in X- und Y- Richtung 8-fach gezoomt, dargestellt.

Label: PR2GR

ROM-Nummer: A

Startadresse: &FE68

Einsprungsbedingungen: L = Zeichen von &00 bis &FF, es muß Mode 2 aktiv sein, RAM-Variablen C_POS und MAX_CRX müssen korrekte Werte enthalten.

Aussprungsbedingungen: 8-fach gezoomtes Zeichen ab C_POS dargestellt.

Manipuliert: AF, BC, DE, HL, BC', HL', YL und das V-RAM.

Beschreibung: Diese OS Funktion dient dazu in Mode 2 ein Zeichen 8-fach gezoomt darzustellen. Das Zeichen wird ab C_POS ausgegeben, C_POS selbst bleibt aber erhalten. Rechts bzw. unterhalb von C_POS sollten noch acht Spalten bzw. Zeilen frei sein. Das Bildschirmformat (Spalten

* Zeilen) kann beliebig gewählt sein, aber die korrekte Spalten-Zahl muß in der RAM-Variable MAX_CRX enthalten sein.

ACHT BIT BREITE BIT-MATRIX DARSTELLEN

Kurzbeschreibung: Eine acht Bit breite, und beliebig hohe Bit-Matrix wird dargestellt. Siehe auch OS Funktion PR2GR, Prinzip ist ähnlich.

Label: PR2GS

ROM-Nummer: A

Startadresse: &FE65

Einsprungsbedingungen: HL = QuellAdr. Matrix-Daten, es muß Mode 2 aktiv sein, RAM-Variablen C_POS und MAX_CRX müssen korrekte Werte enthalten.

YL = Anzahl der Zeilen der Matrix = Anz. Quellbytes ab HL.

Aussprungsbedingungen: Bit-Matrix ab C_POS dargestellt.

Manipuliert: AF, BC, DE, HL, BC', HL', YL und das V-RAM.

Beschreibung: Diese OS Funktion ähnelt im Prinzip obiger OS Funktion PR2GR, nur wird diesmal kein Zeichen dargestellt, sondern eine Bit-Matrix.

Der Unterschied liegt darin, daß ein Zeichen immer 8 Bit breit und hoch ist, wogegen die Bit-Matrix in diesem Falle zwar ebenfalls acht Bit breit ist, aber fast beliebig hoch.

Beim Aufruf wird die Anzahl der Zeilen, also die Höhe der Matrix in YL übergeben, setzt man sie z.B. auf 10, dann ist es möglich mit dieser OS Funktion die komplette Tastaturmatrix darzustellen, siehe auch Programm "ZeigeTasten".

Beim Aufruf zeigt HL auf das erste Quellbyte, diese Bytes werden von oben nach unten, untereinander auf dem Bildschirm dargestellt. Da jedes Byte bekanntlich acht Bit hat, ist die Matrix auch immer acht Bit breit.

Bitte Beachten: Die Anzahl der Matrix-Zeilen darf nicht so groß sein, dass die Matrix nach unten über den Bildrand reicht! Die Matrix wird zwar ab C_POS dargestellt, C_POS bleibt jedoch erhalten.

DRUCKEN EINES 7-BIT ZEICHENS

Kurzbeschreibung: Ein 7-Bit Wert (&00 ... &7F / 0 ... 127) wird an den Drucker geschickt. Alle CPCs.

Label: DRZ7

ROM-Nummer: A

Startadresse: &D67B

Einsprunghbedingungen: A = zu druckernder Wert &00..&7F // 0..127.
PIO Port B (Adresse &F5??) muß auf Eingabe stehen (Drucker-Status).

Aussprunghbedingungen: Das Zero Flag gibt Aufschluss über den Erfolg der Aktion.
Z-Flag gesetzt (Z) ==> Zeichen wurde korrekt an Drucker geschickt.
Z-Flag gelöscht (NZ) ==> Drucker war NICHT bereit!

Manipuliert: F, BC und der Druckerport &EF??.

Beschreibung: DRZ7 ist eine OS Funktion, die es erlaubt auf allen CPCs einen 7 Bit Wert an den Drucker zu schicken. Der Akku enthält bei Aufruf den zu druckenden Wert, sein 8. Bit wird einfach ignoriert.

Wurde der Wert korrekt an den Drucker geschickt, dann kehrt die OS Funktion mit gesetztem Zero Flag zurück. Ist das Zero-Flag dagegen gelöscht, dann war der Drucker nicht zur übernahme eines Bytes bereit.

Bitte Beachten: PIO Port B muss auf Eingabe stehen, denn sonst kann der Status des Druckers nicht überprüft werden. Dies ist aber normalerweise der Fall. DIES GILT BEI ALLEN DRUCK-OS FUNKTIONEN!

WARTEN BIS DRUCKER BEREIT IST

Kurzbeschreibung: Diese OS Funktion wartet bis der Drucker bereit ist, das erste/nächste Zeichen zu empfangen.

Label: XW_DR

ROM-Nummer: A

Startadresse: &D70D

Einsprungsbedingungen: -

Aussprungsbedingungen: Drucker ist bereit ein Zeichen zu empfangen.

Manipuliert: AF, BC und PIO Status (Port B ist Eingang).

Beschreibung: Will man ein Zeichen an den Drucker schicken, dann sollte dieser auch dazu bereit sein. Diese OS Funktion XW_DR wartet nun darauf, dass der Drucker Bereitschaft meldet. Dies ist der Fall, wenn Bit 6 der PIO Port B gelöscht ist.

XW_DR kehrt erst zurück, wenn der Drucker bereit ist. Anschließend kann man ein Zeichen an den Drucker schicken.

Bitte Beachten: Achtung, ist der Drucker z.B. ausgeschaltet, dann kehrt XW_DR NICHT mehr zurück, da ja auf Bereitschaft gewartet wird.

DRUCKEN EINES 8-BIT ZEICHENS (NUR CPC PLUS)

Kurzbeschreibung: Ein 8-Bit Wert wird an den Drucker geschickt. Plus!

Label: DRZP8

ROM-Nummer: A

Startadresse: &D657

Einsprungbedingungen: A = zu druckernder Wert &00 - &FF / 0 - 255.
PIO Port B (Adresse &F5??) muß auf Eingabe stehen (Drucker-Status).

Aussprungbedingungen: Das Zero Flag gibt Aufschluss über den Erfolg der Aktion.
Z-Flag gesetzt (Z) ==> Zeichen wurde korrekt an Drucker geschickt.
Z-Flag gelöscht (NZ) ==> Drucker war NICHT bereit!

Manipuliert: F, BC und der Druckerport &EF??.

Beschreibung: DRZP8 ist eine OS Funktion, die es erlaubt auf den Plus CPCs einen 8 Bit Wert an den Drucker zu schicken. Der Akku enthält bei Aufruf den zu druckenden Wert.

Wurde der Wert korrekt an den Drucker geschickt, dann kehrt die OS Funktion mit gesetztem Zero Flag zurück. Ist das Zero-Flag dagegen gelöscht, dann war der Drucker nicht zur übernahme eines Bytes bereit.

Bitte Beachten: PIO Port B muss auf Eingabe stehen.

DRUCKEN EINES 8-BIT ZEICHENS (NUR CPC OLD GENERATION)

Kurzbeschreibung: Ein 8-Bit Wert wird an den Drucker geschickt. oG!

Label: DRZO8

ROM-Nummer: A

Startadresse: &D66D

Einsprungsbedingungen: A = zu druckernder Wert &00 - &FF / 0 - 255.
PIO Port B (Adresse &F5??) muß auf Eingabe stehen (Drucker-Status).

Aussprungsbedingungen: Das Zero Flag gibt Aufschluss über den Erfolg der Aktion.
Z-Flag gesetzt (Z) ==> Zeichen wurde korrekt an Drucker geschickt.
Z-Flag gelöscht (NZ) ==> Drucker war NICHT bereit!

Manipuliert: F, BC und der Druckerport &EF??.

Beschreibung: DRZO8 ist eine OS Funktion, die es erlaubt auf den CPCs alter Generation (CPCs oG), also 464/664/6128 einen 8 Bit Wert an den Drucker zu schicken. Dies setzt aber voraus, das der 8-Bit-Patch in den CPC installiert wurde. Siehe Konfig-Bytes. Der Akku enthält bei Aufruf den zu druckenden Wert.

Wurde der Wert korrekt an den Drucker geschickt, dann kehrt die OS Funktion mit gesetztem Zero Flag zurück. Ist das Zero-Flag dagegen gelöscht, dann war der Drucker nicht zur Übernahme eines Bytes bereit.

Bitte Beachten: PIO Port B muss auf Eingabe stehen.

Achtung: Diese OS Funktion setzt den PIO-8-Bit-Patch voraus!

DRUCKEN EINES STRINGS (7/8 BIT - CPC PLUS/OG)

Kurzbeschreibung: Diese drei OS Funktionen dienen dem Ausdruck von Strings.

Labels:

- o DRS7 : Drucken eines 7-Bit Strings auf allen CPCs.
- o DRSP8 : Drucken eines 8-Bit Strings auf den PLUS CPCs.
- o DRSO8 : Drucken eines 8-Bit Strings auf den CPCs 464/664/6128 (oG).

ROM-Nummer: A

Startadresse: &D6F0 (DRS7) // &D692 (DRSP8) // &D6C5 (DRSO8)

Einsprungsbedingungen: HL = Zeiger auf String-Anfang. &00 = End-Byte.

Aussprungsbedingungen: Ein Rücksprung erfolgt erst nach dem Ausdruck aller Zeichen des Strings.

Manipuliert: AF, BC, DE, HL und der Drucker-Port.

Beschreibung: Will man mehrere Zeichen ausdrucken, dann kann man eine dieser OS Funktionen benutzen.

Jeh nach dem ob man 7- oder 8- Bit Werte ausdrucken will, jeh nach dem ob man eine CPC Plus oder einen CPC der alten Generation benutzt, kann man einen anderen Einsprung wählen.

Als Programm kann man in den Konfig-Bytes nachschlagen um welchen CPC es sich handelt, und ob dieser auch mit einer 8-Bit Drucker-Schnittstelle ausgerüstet ist.

Beim Aufruf dieser OS Funktion muß HL auf den Anfang der auszudruckenden Zeichenkette zeigen. Dieser String wird durch ein &00 Byte beendet.

Die OS Funktion kehrt erst nach Ausdruck aller Zeichen des Strings zurück.

Man sollte von Aufruf den Status des Druckers testen. Dies geschieht z.B. folgendermaßen:

```
TEST_DRUCKER LD BC,&F782 ;PIO Steuerregister
```

```
OUT (C),C ;Port B als Eingang schalten.
```

```
LD B,&F5 ;PIO Port B
```

```
IN A,(C) ;Drucker-Status lesen.
```

```
BIT 6,A ;Status testen.
```

```
JR NZ,DRUCKER_NICHT_BEREIT
```

;oder ...

```
JR Z,DRUCKER_IST_BEREIT
```

Bitte Beachten: Der Drucker muß bei Aufruf einer dieser OS Funktionen unbedingt bereit sein, denn sonst erfolgt KEIN Rücksprung.

HD20 - ERMITTLE DIE ANZAHL FREIER EINTRÄGE EINER PARTITION

Kurzbeschreibung: Diese OS Funktion ermittelt die Anzahl der freien IHV-Einträge einer HD-Partition.

Label: EFED_HD

ROM-Nummer: A

Startadresse: &D71B

Einsprungsbedingungen: REG08_1 = Partition 8...11 (entspricht I...L). Das 32er IHV der Partition muß eingelesen sein.

Aussprungsbedingungen: BC = &0020 // HL = &8000
DE = Anzahl der freien IHV/DIR Einträge der Partition aus REG08_1.
Das 32er IHV/DIR der Partition ist ab &4000 eingeblendet.

Manipuliert: AF, BC, DE, HL und der RAM-Status.

Beschreibung: Diese OS Funktion berechnet die Anzahl der freien Einträge des Inhaltsverzeichnisses (IHV/DIR) einer Festplatten-Partition.

Beim Aufruf wird die Nummer der Partition in der RAM-Variable REG08_1 übergeben, die Partitionen I, J, K und L werden dabei durch die Nummern 8, 9, 10 und 11 symbolisiert, NICHT durch 0 bis 3 wie bei den HD - LadeOS Funktionen.

Die OS Funktion berechnet die Anzahl freier Einträge und übergibt sie in DE. Dabei kann DE Werte zwischen &0000 und &0200 annehmen, denn ein 16KB IHV stellt 512 Einträge zur Verfügung.

Bitte Beachten: Die HD muß aktiv sein und ihr IHV muß korrekt eingelesen worden sein.

HD20 - BLOCKBELEGUNGSTABELLE EINER PARTITION GENERIEREN

Kurzbeschreibung: Die Blockbelegungstabelle einer HD - Partition wird generiert.

Label: BBTG_HD

ROM-Nummer: A

Startadresse: &D73F

Einsprungsbedingungen: REG08_1 = Partition 8...11 (entspricht I...L).
Das 32er IHV der Partition muß eingelesen sein.

Aussprungsbedingungen: Blockbelegungstabelle ab &B000 im RAM.
Das 32er IHV/DIR der Partition ist ab &4000 eingeblendet.

Manipuliert: AF, BC, DE, HL, HL', IX, der RAM-Status und der RAM-Bereich von &B000 bis &B7FF.

Beschreibung: Diese OS Funktion berechnet die BBT (=BlockBelegungsTabelle) einer Festplatten-Partition. Die BBT wird ab &B000 angelegt, und reicht normalerweise (5 MB Partition) bis &B50D, maximal bis &B7FF.

Die ersten vier Blöcke (0 bis 3) sind für das IHV/DIR der Partition reserviert. Sucht man freie Blöcke, so sollte man mit der Suche ab Block &0004 = 4 beginnen. In dieser BBT steht das Block-Byte für Block &0000 in Adresse &B000, das für Block &0001 in &B001 usw. Das letzte Block-Byte für Block &050D steht dann in Adresse &B50D.

Ein belegter Block wird dabei durch ein &FF Block-Byte symbolisiert, ein freier Block durch ein &00 Block-Byte.

Beim Aufruf wird die Nummer der Partition in der RAM-Variable REG08_1 übergeben, die Partitionen I, J, K und L werden dabei durch die Nummern 8, 9, 10 und 11 symbolisiert, NICHT durch 0 bis 3 wie bei den HD - LadeOS Funktionen.

Nach dem Rücksprung steht die BBT ab &B000 zur Verfügung, ihr folgen &00 Bytes bis zur Adresse &B7FF. Das 32er IHV der Partition bleibt zwischen &4000 und &7FFF eingeblendet.

Bitte Beachten: Die HD muß aktiv sein und ihr IHV muß korrekt eingelesen worden sein.

HD20 - INHALTSVERZEICHNISSEINTRAG EINER DATEI GENERIEREN

Kurzbeschreibung: Generierung des Inhaltsverzeichniseintrages einer auf HD zu sichernden Datei.

Label: EGEN_HD

ROM-Nummer: A

Startadresse: &D7A7

Einsprungsbedingungen: 32er IHV/DIR muß ab &4000 eingeblendet sein.

BC = Blockanzahl der Datei, deren DIR-Eintrag generiert werden soll.

REG16_8 = User(1), Name(8), Erweiterung(3) der Datei. 12 Bytes.

Ab &B000 muß eine BBT für die HD Partition vorhanden sein.

Aussprungsbedingungen: Die Datei hat im IHV/DIR ihren Eintrag erhalten.

Das 32er DIR/IHV der Partition ist nun unsortiert.

Manipuliert: AF, BC, E, BC', DE', HL', XL und das IHV ist unsortiert.

Beschreibung: Will man eine Datei auf HD speichern, dann muß man sie auch im IHV(=Inhaltsverzeichnis) mit einem oder mehreren Einträgen vermerken. Erst danach kann man die Speicher-Table mittels HD_TAB generieren, und die eigentlichen Daten mit SHD4 bzw. SHD6 speichern.

Diese OS Funktion dient nun dazu einer zu speichernden Datei einen oder mehrere Einträge im IHV zuzuweisen. Dazu sind folgende Einsprungsbedingungen nötig:

- o Das IHV, in das die Datei integriert werden soll, muß zwischen &4000 und &7FFF eingeblendet sein (Dies erledigt z.B. BBTG_HD).

- o Die BBT (Block-Belegung-Tabelle) dieser Partition (durch BBTG_HD generiert) muß ab &B000 im RAM stehen.

- o Das Doppelregister BC enthält die Blockanzahl der Datei, deren IHV-Einträge (Extents) generiert werden sollen.

- o Ab der RAM-Variable REG16_8 stehen 12 Bytes im RAM, die User(1Byte), Name(8) und Datei-Typ(3) der Datei enthalten.

Nach dem Aufruf der OS Funktion wurde für die Datei ein Eintrag im IHV generiert.

Für jeweils 32 KB Dateilänge wird je ein Eintrag im IHV belegt, denn die Dobbartin HD hat eine Blockgröße von 4 KB und 16 Bit Blocknummern.

Das IHV ist nun unsortiert. Zum sortieren kann man folgendermaßen vorgehen: ROM C einblenden, dann SSB0, ISWS und RRB0 aufrufen.

Bei der Generierung der Einträge wird die Record-Anzahl immer auf die nächste 4 KB Grenze aufgerundet. Eine 17 KB Datei belegt auf HD also 20 KB. Siehe auch 4 KB Blockgröße.

Bitte Beachten: Beim Rücksprung ist der 32er IHV der Partition unsortiert. Man sollte deshalb SSB0, ISWS, RRB0 (alle ROM C) aufrufen.

(TEILWEISE) SICHERN einer DATEI aus dem KURZ-ZEIT-SPIECHER auf HD20

Kurzbeschreibung: Eine Datei wird (teilweise) aus dem KZS auf die Festplatte gesichert.

Label: TSDHD (anfangs) und TDEHD (bei Dateirest)

ROM-Nummer: A

Startadresse: &D88D (TSDHD), &D87E (TSEHD)

Einsprungsbedingungen:

a. TSDHD:

A = Partition, auf die gespeichert werden soll: 8..11 (nur HD20 !!).

REG_BC1 = Dateilänge in KB.

REG16_8 = User(1), Name(8) und Extension(3) der Datei (12 Bytes).

b. TSEHD: Die RAM Variablen REG_AF1, REG_DE1, REG_HL1, REG_IX und REG16_8 bis REG16_8+12 müssen seit TSDHD erhalten worden sein.

Aussprungsbedingungen: Der Akku liefert Informationen über den Erfolg der Operation:

A = &FF ==> Die Datei wurde komplett auf HD geschrieben. Gut so!

A = &F0 ==> Die Datei wurde nur TEILWEISE auf HD geschrieben, es muß nochmals nachgeladen werden, der Rest wird mit TSEHD gesichert.

A = &00 ==> Es ist überhaupt kein DIR eingelesen.

A = &01 ==> Die gewählte Partition ist nicht markiert.

A = &02 ==> Die Datei hat eine Länge von 0 KB. Also auch Abbruch.

A = &04 ==> Im Ziel-DIR sind zu wenig Einträge frei.

A = &05 ==> Es sind keine KZS im E-RAM belegt/sicherbar.

Bei TSDHD & TSEHD manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', IX, IY, RAM Variablen REG08_0,1, REG_DE1(low), REG_PC(low).

Nur bei TSDHD manipuliert: RAM-Variablen TURBO_A bis M, REG16_0 bis 9, REG32_1,2, REG_AF1, REG_HL1, REG_IX, das RAM von &B000 bis &BFFF.

Außerdem wurden die RAM-Konfiguration und die Inhaltsverzeichnisse verändert.

Beschreibung: Diese OS Funktion ist das Festplatten-Gegenstück zu TEILA. Während TEILA eine Datei teilweise lädt, dienen TSDHD / TSEHD dazu die geladene Datei wieder abzuspeichern, und zwar auf HD. Um auf Diskette Teil-zu-speichern würde man TEISI / TEISK nutzen. Das Speichern geschieht hierbei ganz oder in Stückchen.

Zuerst springt man nach TSDHD. TSDHD generiert einen Eintrag im Inhaltsverzeichniss und speichert alle KZS(Kurz-Zeit-Speicher) auf Diskette. Ein Fehler wird im Akku übergeben.

Kehrt TSDHD mit dem Byte &FF im Akku zurück, dann wurde die Datei komplett gesichert. Enthält der Akku jedoch den Wert &F0, so wurden zwar alle KZS gespeichert, aber die Datei ist länger als das zur Verfügung stehende RAM. Einige RAM-Variablen sind nun zu erhalten s.o.

Es muß also erst der Dateirest mit TEILB nachgeladen werden, dann kann dieser Rest mit TSEHD auf HD geschrieben werden. Dieser Vorgang kann sich mehrmals wiederholen, wenn die Datei lang oder das E-RAM gering ist.

Das Inhaltsverzeichnis wird zwar mit den korrekten neuen Einträgen versorgt, jedoch wird es nicht auf HD geschrieben. Dies hat z.B. dann Sinn, wenn mehrere Dateien kopiert werden sollen.

Bitte Beachten: Das neue DIR muß noch mittels z.B. SIDIR gesichert werden. Vor Aufruf der OS Funktion kann es sinnvoll sein, einen weiteren Block (unterhalb des DIR Puffers) als DIR Puffer zu vermerken. Da neue Einträge generiert werden, steigt der RAM Verbrauch im DIR Puffer.

ACHTUNG: TSDHD und TSEHD sind NUR für HD zuständig. Die entsprechenden OS Funktionen, um auf Disk Teil-zu-speichern sind in ROM C enthalten.

TESTE OB DAS SYMBiFACE II VORHANDEN IST

Kurzbeschreibung: Diese Funktion ermittelt ob ein SYMBiFACE II (nicht CPC-IDE oder X-MASS) angeschlossen ist.

Label: T_SF

ROM-Nummer: A

Startadresse: &FE4D

Einsprungsbedingungen: -

Aussprungsbedingungen: Z-Flag gesetzt -> SYMBiFACE II angeschlossen Zero gelöscht --> SYMBiFACE II nicht vorhanden.

Manipuliert: AF, BC und RTC-Millennium auf 20 gesetzt

Beschreibung: Diese Funktion wird dazu verwendet, um zu testen ob ein SYMBiFACE II angeschlossen ist. Je nachdem, ob ein SF2 angeschlossen ist oder nicht, wird das Zero-Flag beeinflusst. Wenn ein SF2 vorhanden ist, so wird das Zero Flag gesetzt, andernfalls gelöscht.

Diese Funktion unterscheidet zwischen CPC-IDE bzw. X-MASS einerseits und SYMBiFACE II andererseits. Es wird die Existenz der Echtzeituhr (RTC) getestet. Ein CPC-IDE bzw. X-MASS wird nicht erkannt, und das Zero-Flag wird gelöscht.

Bitte Beachten: Die CPC-IDE / X-MASS Erweiterungen werden nicht erkannt. Diese Funktion setzt das Millennium-Byte der SF2-RTC auf 20.

TESTE OB DAS SYMBiFACE III VORHANDEN IST

Kurzbeschreibung: Funktion tested ob das SYMBiFACE III vorhanden ist.

Label: T_SF3

ROM-Nummer: A

Startadresse: &D875

Einsprungbedingungen: -

Aussprungbedingungen: Z-Flag gesetzt -> SYMBiFACE III angeschlossen
 Zero gelöscht --> SYMBiFACE III nicht vorhanden

Manipuliert: AF und BC

Beschreibung: Diese Funktion wird dazu verwendet, um zu testen ob ein SYMBiFACE III angeschlossen ist. Je nachdem, ob ein SF3 angeschlossen ist oder nicht, wird das Zero-Flag beeinflusst. Ist ein SF3 vorhanden, so wird das Zero Flag gesetzt, andernfalls gelöscht.

Diese OS Funktion benutzt die Echo-Funktion des SF3 um zu testen, ob es angeschlossen ist.

Bitte Beachten: -

SCHREIBE ZEIT / DATUM IN DIE RTC DES SYMBIFACE II

Kurzbeschreibung: Diese OS Funktion beschreibt die RTC des SYMBiFACE II mit Zeit- und Datums-Daten.

Label: S_SFRT

ROM-Nummer: A

Startadresse: &FE50

Einsprungsbedingungen: HL = 7 Bytes Daten (Zeit, Datum; im BCD-Format) Abfolge der 7 Bytes: Sekunde, Minute, Stunde, Wochentag, Tag, Monat, Jahr

Aussprungsbedingungen: Zeit-Daten in SYMBiFACE II-RTC geschrieben.

Manipuliert: AF, BC, DE, HL und SYMBiFACE II-RTC.

Beschreibung: S_SFRT dient dazu die Echtzeituhr des SYMBiFACE II mit Zeit & Datum zu beschreiben. In HL wird die Adresse eines Datenblocks übergeben, der folgenden Aufbau hat:

- Sekunde &00 - &59
- Minute &00 - &59
- Stunde &00 - &23
- Wochentag &01 - &07 (1 = Montag)
- Tag &00 - &31
- Monat &01 - &12
- Jahr &00 - &99

Das Datenformat dieser 7 Bytes ist BCD, diese 7 Bytes können an beliebiger Stelle im Speicher lokalisiert sein. Das FutureOS legt diese 7 Bytes in dieser Form ab UHR_SEK in den System-Variablen ab. Will man die Uhr per Hand stellen, so kann dieser Bereich ebenfalls verwendet werden.

Bitte Beachten: Es wird nur die RTC des SYMBiFACE II beschrieben, andere angeschlossene RTCs werden nicht beeinflusst.

SCHREIBE ZEIT / DATUM IN DIE RTC DES SYMBIFACE III

Kurzbeschreibung: Diese OS Funktion beschreibt die RTC des SYMBiFACE III mit Zeit- und Datums-Daten.

Label: S_SF3RTC

ROM-Nummer: A

Startadresse: &FE2C

Einsprungsbedingungen: Ab der System-Variable UHR_SEK müssen sich gültige Zeit- und Datums-Daten befinden. Es wird das BCD-Format verwendet.

Aussprungsbedingungen: Zeit und Datum in SYMBiFACE III RTC geschrieben.

Manipuliert: AF, BC, DE, HL und SYMBiFACE III RTC.

Beschreibung: Die OS Funktion S_SF3RTC dient dazu die Echtzeituhr des SYMBiFACE III mit Zeit & Datum zu beschreiben.

Zeit und Datum werden dafür aus den System-Variablen ab UHR_SEK gelesen. Der Datenblock hat folgenden Aufbau:

- Sekunde &00 - &59
- Minute &00 - &59
- Stunde &00 - &23
- Wochentag &01 - &07 - ungenutzt
- Tag &00 - &31
- Monat &01 - &12
- Jahr &00 - &99

Das Datenformat dieser sieben Bytes ist BCD.

Bitte Beachten: Es wird nur die RTC des SYMBiFACE III beschrieben, andere eventuell angeschlossene RTCs werden nicht beeinflusst.

SCHREIBE ZEIT / DATUM in die RTC des LAMBDA SPEAK III oder FS

Kurzbeschreibung: Diese OS Funktion beschreibt die RTC des LambdaSpeak III bzw. FS mit Daten für Zeit und Datum.

Label: S_LS3RTC

ROM-Nummer: A

Startadresse: &FE23

Einsprungsbedingungen: Ab der System-Variable UHR_SEK müssen sich gültige Zeit- und Datums-Daten befinden. Es wird das BCD-Format verwendet.

Aussprungsbedingungen: Zeit und Datum wurden in LambdaSpeak III/FS RTC geschrieben.

Manipuliert: AF, BC, DE, HL und LambdaSpeak III/FS RTC.

Beschreibung: Die OS Funktion S_LS3RTC dient dazu die Echtzeituhr des LambdaSpeak III/FS mit Zeit & Datum zu beschreiben.

Zeit und Datum werden dafür aus den System-Variablen ab UHR_SEK gelesen. Der Datenblock hat folgenden Aufbau:

- Sekunde &00 - &59
- Minute &00 - &59
- Stunde &00 - &23
- Wochentag &00 - &06
- Tag &00 - &31
- Monat &01 - &12
- Jahr &00 - &99

Das Datenformat dieser sieben Bytes in den OS Variablen ist BCD.

Bitte Beachten: Es wird lediglich die RTC des LambdaSpeak III/FS beschrieben, andere eventuell angeschlossene RTCs werden nicht beeinflusst.

SCHREIBE ZEIT / DATUM in die RTC der NOVA Erweiterung

Kurzbeschreibung: Diese OS Funktion beschreibt die RTC der Nova Karte mit Daten für Zeit und Datum.

Label: S_NRTC

ROM-Nummer: A

Startadresse: &FCCA

Einsprungsbedingungen: Ab der System-Variable UHR_SEK müssen sich gültige Zeit- und Datums-Daten befinden. Es wird das BCD-Format verwendet.

Aussprungsbedingungen: Zeit und Datum wurden in Nova RTC geschrieben
Bei gesetztem Carry wurden die Daten geschrieben.
Bei geleertem Carry wurde keine Nova Karte gefunden.

Manipuliert: AF, BC, DE, HL und die Nova RTC

Beschreibung: Die OS Funktion S_NRTC dient dazu die Echtzeituhr der Nova Karte mit Zeit & Datum zu beschreiben.

Zeit und Datum werden dafür aus den System-Variablen ab UHR_SEK gelesen. Der Datenblock hat folgenden Aufbau:

- Sekunde	&00 - &59
- Minute	&00 - &59
- Stunde	&00 - &23
- Wochentag	&00 - &06
- Tag	&00 - &31
- Monat	&01 - &12
- Jahr	&00 - &99

Das Datenformat dieser sieben Bytes in den OS Variablen ist BCD.

Bitte Beachten: Es wird lediglich die RTC der Nova beschrieben, andere eventuell angeschlossene RTCs werden nicht beeinflusst.

LESE ZEIT UND DATUM DER SYMBiFACE II-RTC INS RAM

Kurzbeschreibung: Die Daten (z.B. Zeit und Datum) der Echtzeituhr des SYMBiFACE II werden ins RAM gelesen.

Label: R_SFRT

ROM-Nummer: A

Startadresse: &FE53

Einsprungsbedingungen: HL = Zeiger auf 10 Bytes Puffer im RAM

Aussprungsbedingungen: Puffer (ab voriger Adresse in HL) mit RTC-Daten gefüllt. Die Abfolge: Jahr, Monat, Tag, Wochentag, Alarm Stunde, aktuelle Stunde, Alarm Minute, aktuelle Minute, Alarm Sekunde, aktuelle Sekunde.

Carry geleert --> Daten ordnungsgemäß gelesen!

Carry gesetzt --> Ein Fehler ist aufgetreten, z.B. ein Uhr-Update.

Manipuliert: AF, BC, D, HL und 10 Bytes ab altem HL im RAM.

Beschreibung: R_SFRT liest die Daten der Echtzeituhr des SYMBiFACE II ins RAM. Die Zieladresse wird vor Aufruf der Funktion in HL übergeben.

Nach dem Aufruf dieser OS Funktion stehen die 10 Byte RTC-Daten ab der zuvor in HL übergebenen Adresse im RAM. Die Abfolge ist die folgende:

- Jahr	&00 - &99
- Monat	&01 - &12
- Tag	&01 - &31
- Wochentag	&01 - &07 (1 = Montag)
* Alarm Stunde	&00 - &23
- Aktuelle Stunde	&00 - &23
* Alarm Minute	&00 - &59
- Aktuelle Minute	&00 - &59
* Alarm Sekunde	&00 - &59
- Aktuelle Sekunde	&00 - &59

Die Daten werden ebenfalls wieder im BCD Format übergeben. Siehe OS Funktion zuvor.

Der Erfolg dieser OS Funktionen läßt sich am Carry-Flag erkennen. Ist das Carry-Flag nach dem Rücksprung der Funktion geleert wurden alle Daten ordnungsgemäß gelesen. Ist das Carry nach dem Rücksprung allerdings gesetzt, so ist ein Fehler aufgetreten, in den meisten Fällen wird von der Echtzeituhr gerade ein 'internes Update' durchgeführt. In diesem Fall ist es sinnvoll die Funktion einfach noch ein mal aufzurufen.

Bitte Beachten: Nach der Ausführung der OS Funktion bitte eventuelle Fehler durch das Carry-Flag analysieren.

LESE ZEIT UND DATUM DER SYMBiFACE III RTC INS RAM

Kurzbeschreibung: Die Daten (Zeit und Datum) der Echtzeituhr des SYMBiFACE III werden ins System-RAM (ab UHR_SEK) gelesen.

Label: R_SF3RTC

ROM-Nummer: A

Startadresse: &FE2F

Einsprungsbedingungen: SF3 sollte vorhanden sein

Aussprungsbedingungen: Die sieben System-Variablen ab UHR_SEK wurden mit Daten der SF3 RTC gefüllt.

Manipuliert: AF, BC, DE, HL und sieben Bytes ab UHR_SEK.

Beschreibung: Die OS Funktion R_SF3RTC liest die Daten der Echtzeituhr des SYMBiFACE III ins System-RAM, und zwar ab der System-Variable UHR_SEK (plus folgende Zeit-/Datums-Bytes).

Nach dem Aufruf dieser OS Funktion stehen die sieben Bytes RTC-Daten ab UHR_SEK im System-RAM:

- Aktuelle Sekunde	&00 - &59
- Aktuelle Minute	&00 - &59
- Aktuelle Stunde	&00 - &23
- Wochentag	&01 - &07 - ungenutzt!
- Tag	&01 - &31
- Monat	&01 - &12
- Jahr	&00 - &99

Die Daten werden ebenfalls wieder im BCD Format übergeben. Siehe OS Funktion zuvor.

Bitte Beachten: -

LESE ZEIT und DATUM der LAMBDA SPEAK III bzw. FS RTC ins RAM

Kurzbeschreibung: Die Daten (Zeit und Datum) der Echtzeituhr des LambdaSpeak III / FS werden ins System-RAM (ab UHR_SEK) gelesen.

Label: R_LS3RTC

ROM-Nummer: A

Startadresse: &FE26

Einsprungsbedingungen: LambdaSpeak III oder FS mit RTC vorhanden

Aussprungsbedingungen: Die sieben System-Variablen ab UHR_SEK wurden mit Daten der LS3/LFS RTC gefüllt.

Manipuliert: AF, BC, DE, HL und sieben Bytes ab UHR_SEK.

Beschreibung: Die OS Funktion R_SF3RTC liest die Daten der Echtzeituhr des LambdaSpeak III/FS ins System-RAM, und zwar ab der System-Variable UHR_SEK (plus folgende Zeit-/Datums-Bytes).

Nach dem Aufruf dieser OS Funktion stehen die sieben Bytes RTC-Daten ab UHR_SEK im System-RAM:

- Aktuelle Sekunde	&00 - &59
- Aktuelle Minute	&00 - &59
- Aktuelle Stunde	&00 - &23
- Wochentag	&00 - &06
- Tag	&01 - &31
- Monat	&01 - &12
- Jahr	&00 - &99

Die Daten werden ebenfalls wieder im BCD Format übergeben. Siehe OS Funktionen zuvor.

Bitte Beachten: Ein LambdaSpeak III bzw. FS mit RTC sollte vorhanden sein.

LESE ZEIT und DATUM der NOVA RTC ins RAM ab UHR_SEK

Kurzbeschreibung: Die Daten (Zeit und Datum) der Echtzeituhr der Nova Erweiterung werden ins System-RAM (ab UHR_SEK) gelesen.

Label: R_NRTC

ROM-Nummer: A

Startadresse: &FCB6

Einsprungsbedingungen: NOVA mit RTC vorhanden

Aussprungsbedingungen: Die sieben System-Variablen ab UHR_SEK wurden mit Daten der NOVA RTC gefüllt.

Bei gesetztem Carry wurden die Daten gelesen.

Bei geleertem Carry wurde keine Nova Karte gefunden.

Manipuliert: AF, BC, DE, HL und sieben Bytes ab UHR_SEK

Beschreibung: Die OS Funktion R_NRTC liest die Daten der Echtzeituhr der Nova Erweiterung ins System-RAM, und zwar ab der System-Variable UHR_SEK (plus folgende Zeit-/Datums-Bytes).

Nach dem Aufruf dieser OS Funktion stehen die sieben Bytes RTC-Daten ab UHR_SEK im System-RAM:

- Aktuelle Sekunde	&00 - &59
- Aktuelle Minute	&00 - &59
- Aktuelle Stunde	&00 - &23
- Wochentag	&00 - &06
- Tag	&01 - &31
- Monat	&01 - &12
- Jahr	&00 - &99

Die Daten werden ebenfalls wieder im BCD Format übergeben. Siehe OS Funktionen zuvor.

Bitte Beachten: Eine Nova Karte mit RTC sollte vorhanden sein.

Daten der SYMBiFACE II RTC in FutureOS RAM-Variablen transferieren

Kurzbeschreibung: Die zuvor mittels R_SFRT gelesenen RTC-Daten werden in das FutureOS Format konvertiert und ab UHR_SEK abgelegt.

Label: K_SF20

ROM-Nummer: A

Startadresse: &FE56

Einsprungsbedingungen: HL zeigt auf 10 Byte RTC-Daten. Deren Abfolge ist: Jahr, Monat, Tag, Wochentag, Alarm Stunde, aktuelle Stunde, Alarm Minute, aktuelle Minute, Alarm Sekunde, aktuelle Sekunde.

Aussprungsbedingungen: 7 Zeit / Datums-Bytes wurden ab UHR_SEK abgelegt

Manipuliert: F, BC, DE und HL

Beschreibung: Werden die Daten der SYMBiFACE II Echtzeituhr mittels R_SFRT in 10 Bytes RAM gelesen, so entsprechen sie in ihrer Abfolge nicht den Zeit-Bytes des FutureOS. Die Funktion konvertiert die zuvor eingelesenen Bytes in das FutureOS Zeit- und Datums-Format. Vor dem Aufruf dieser OS Funktion ist ihr in HL mitzuteilen ab welcher Adresse die zuvor gelesenen 10 Bytes stehen. Nach der Rückkehr dieser OS Funktion sind dann 7 Bytes (Uhrzeit, Wochentag, Datum) ab UHR_SEK in den FutureOS System-Variablen abgelegt.

Bitte Beachten: Die Alarm-Stunde, -Minute, und -Sekunde wird nicht konvertiert.

Die ECHTZEITUHR des SYMBiFACE II wird auf BCD-Kodierung geschalten

Kurzbeschreibung: Die Zeitverwaltung der SYMBiFACE II RTC wird auf BCD-Kodierung geschalten.

Label: RTC2BC

ROM-Nummer: A

Startadresse: &FE59

Einsprungsbedingungen: -

Aussprungsbedingungen: Zero-Flag gesetzt ==> BCD war bereits aktiviert Zero-Flag geleert ==> RTC jetzt auf BCD gestellt

Manipuliert: AF, BC, DE, HL und die SF-RTC

Beschreibung: Die Echtzeituhr des SYMBiFACE II wird auf BCD-Kodierung gestellt. Die RTC beherrscht sowohl BCD- als auch Binär-Kodierung, da BCD aber einfacher und schneller zu handhaben ist wird es unter FutureOS verwendet. Außerdem verwenden die Echtzeituhren anderer Hersteller (z.B. Dobbertin) ebenfalls standardmäßig BCD-Kodierung.

Unter FutureOS werden die zentralen Zeit-Daten in den System-Variablen (ab UHR_00) dementsprechend ebenfalls im BCD Format verwaltet.

Bitte Beachten: Das Umstellen der SYMBiFACE II RTC führt u.U. bei nicht-FutureOS-Programmen zu Problemen, wenn diese nur mit Binär-Kodierung arbeiten können.

ALARM-ZEIT IN DIE RTC DES SYMBIFACE II SCHREIBEN

Kurzbeschreibung: Die Alarmzeit (Stunde, Minute, Sekunde) der RTC des SYMBiFACE II wird gesetzt.

Label: S_SFWZ

ROM-Nummer: A

Startadresse: &FE4A

Einsprungsbedingungen: HL = Zeiger auf drei Bytes Alarm Daten im BCD Format. Aufbau: Sekunde (&00-&59), Minute (&00-&59), Stunde (&00-&23).

Aussprungsbedingungen: Alarm-Zeit der SYMBiFACE II RTC gesetzt.

Manipuliert: AF, BC, DE, HL und SF-RTC (auf BCD gesetzt!)

Beschreibung: Die interne Alarm-Zeit der Echtzeituhr des SYMBiFACE II wird gesetzt. Bei Aufruf der OS Funktion muß HL einen Zeiger auf einen drei Byte-Datenblock enthalten, der den folgenden Aufbau hat:

- Sekunde &00-&59
- Minute &00-&59
- Stunde &00-&23

Die drei Alarm-Zeit Bytes werden wie üblich im BCD Format übergeben.

Bitte Beachten: Das setzen der Alarm-Zeit in der SF-RTC führt nicht automatisch zum Auslösen eines Weck-Alarms unter FutureOS, dazu muß die entsprechende Weck-Zeit im Turbo-Desk eingestellt werden.

KONVERTIERE DATEN EINER PROPORTIONALEN MAUS AUF JOYSTICK

Kurzbeschreibung: Diese OS Funktion liest die Positionsdaten einer aktiven proportionalen Maus ein und wandelt sie in ein zum Joystick kompatibles Format um

Label: M_CON

ROM-Nummer: A

Startadresse: &FD32

Einsprungsbedingungen: -

Aussprungsbedingungen: Register A enthält die Aktions-Bits. Ein leeres Bit entspricht einer Feuertaste oder Richtung.

A = %1, Feuer 2, Feuer 1, Feuer 0, Rechts, Links, Unten, Oben (0=AN)

Manipuliert: AF, BC, DE, H und IY

Beschreibung: Die OS Funktion M_CON dient zur Abfrage der angeschlossenen proportionalen Mäuse. Es werden die PS/2 Maus des SYMBiFACE II, die USB Maus des SYMBiFACE III und die Mäuse des MultiPlay überprüft. Diese Daten werden im Joystick Bit-kompatiblen Format übergeben.

Falls keine Maus angeschlosse ist bzw. keine Maus bewegt wurde, so kehrt diese OS Funktion mit dem Wert &FF in Register A zurück.

Ansonsten haben die einzelnen Bits die selbe Bedeutung wie bei den OS Funktionen H_JC, H_CCE und H_JOY.

Bit 0: Aufwärts

Bit 1: Abwärts

Bit 2: Links

Bit 3: Rechts

Bit 4: Feuer 0

Bit 5: Feuer 1

Bit 6: Feuer 2

Bit 7: 1 (immer auf 1 gesetzt)

Bitte Beachten: -

ABFRAGE EINER PROPORTIONALEN MAUS

Kurzbeschreibung: Diese OS Funktion liest die Positionsdaten einer angeschlossenen und betriebsbereiten proportionalen Maus ein.

Label: R_ALM

ROM-Nummer: A

Startadresse: &FC6E

Einsprungsbedingungen: -

Aussprungsbedingungen: Das Zero-Flag informiert über den Erfolg
Zero-Flag gesetzt: Keine Maus vorhanden oder sie wurde nicht bewegt
Zero-Flag geleert: Erfolg! Siehe Daten in folgenden Registern:

D = Feuer-Tasten %000, Back, Fwd, Feuer 2, Feuer 1, Feuer 0 (1=AN)

E = Musrad &01-&1F (positiv) oder &FF-&E0 (negativ)

YL = X-Koordinate
= %00000001 (+1) bis %00011111 - Maus nach rechts bewegt
= %11111111 (-1) bis %11100000 - Maus nach links bewegt

YH = Y-Koordinate
= %00000001 (+1) bis %00100000 - Maus runter
= %11111111 (-1) bis %11100000 - Maus rauf

Manipuliert: AF, BC, DE, H und IY

Beschreibung: Die OS Funktion R_ALM dient zur Abfrage der angeschlossenen proportionalen Mäuse. Es werden die PS/2 Maus des SYMBiFACE II, die USB Maus des SYMBiFACE III und die Mäuse des MultiPlay überprüft.

Falls keine Maus angeschlosse ist bzw. keine Maus bewegt wurde, so kehrt diese OS Funktion mit gesetzten Zero-Flag zurück. Ist das Zero Flag jedoch gelöscht, so können die relevanten Daten aus den Registern D, E, YL und YH gelesen werden.

Die übergebenen delta-X und delta-Y Werte können direkt zur aktuellen Position addiert werden.

Siehe dazu auch die nachfolgenden OS Funktionen.

Bitte Beachten: -

ABFRAGE DER SYMBiFACE II PS/2 MAUS

Kurzbeschreibung: Diese OS Funktion liest die Positionsdaten der PS/2 Maus des SYMBiFACE II ein.

Label: R_PS2

ROM-Nummer: A

Startadresse: &FE32

Einsprungsbedingungen: -

Aussprungsbedingungen: Das Zero-Flag informiert über den Erfolg
Zero-Flag gesetzt: Keine Maus vorhanden oder sie wurde nicht bewegt
Zero-Flag geleert: Erfolg! Siehe Daten in folgenden Registern:

D = Feuer-Tasten %000, Back, Fwd, Feuer 2, Feuer 1, Feuer 0 (1=AN)

E = Mauseisen &01-&1F (positiv) oder &FF-&E0 (negativ)

YL = X-Koordinate
= %000 00001 (+1) bis %000 11111 (+31) - Maus nach rechts bewegt
= %111 11111 (-1) bis %111 00000 (-31) - Maus nach links bewegt

YH = Y-Koordinate
= %000 00001 (+1) bis %001 00000 (+32) - Maus runter
= %111 11111 (-1) bis %111 00000 (-31) - Maus rauf

Manipuliert: AF, BC, DE, H und IY

Beschreibung: Die OS Funktion R_PS2 dient zur Abfrage der PS/2 Maus des SYMBiFACE II.
Falls keine PS/2 Maus angeschlossen ist wird in den Doppelregistern DE und IY jeweils der Wert &0000 zurückgegeben. Dies ist auch der Fall falls gar kein SF2 angeschlossen ist.

Falls die SF2 PS2 Maus erfolgreich abgefragt wird, dann werden in Register D die fünf möglichen Feuerknöpfe in den Bits 4-0 angegeben. In Register E wird der Versatz des Mauseisens übermittelt. Und Register IY (low) und IY (high) enthalten die X- bzw. Y-Koordinate respektive. Negative Werte werden im 2er Komplement angegeben, so wird -1 z.B. als 11111111 (binär) angegeben. Dadurch kann der übergebene Wert sofort zur aktuellen Koordinate addiert werden.

Bitte Beachten: Mit T_SF kann geprüft werden, ob das SF2 und damit die PS2 Maus angeschlossen sind.

Die PS2 Maus muss vor dem Einschalten des SF2 (und anschließend des CPC) angeschlossen werden um benutzt werden zu können. Das SF2 beherrscht in diesem Fall kein "hot plug and play".

ABFRAGE DER SYMBiFACE III USB MAUS

Kurzbeschreibung: Diese OS Funktion liest die Positionsdaten der USB Maus des SYMBiFACE III ein.

Label: R_USB3

ROM-Nummer: A

Startadresse: &FE29

Einsprungsbedingungen: -

Aussprungsbedingungen: Das Zero-Flag informiert über den Erfolg
Zero-Flag gesetzt: Keine Maus vorhanden oder sie wurde nicht bewegt
Zero-Flag geleert: Erfolg! Siehe Daten in folgenden Registern:

D = Feuer-Tasten %000, Back, Fwd, Feuer 2, Feuer 1, Feuer 0 (1=AN)

E = Mausekranz 0 bis 100 (positiv) oder -1 bis -100 (negativ)

YL = X-Koordinate
= %000 00001 (+1) bis %011 11111 (+100) - Maus nach rechts bewegt
= %111 11111 (-1) bis %100 00000 (-100) - Maus nach links bewegt

YH = Y-Koordinate
= %000 00001 (+1) bis %010 00000 (+100) - Maus runter
= %111 11111 (-1) bis %110 00000 (-100) - Maus rauf

Manipuliert: AF, BC, DE und IY

Beschreibung: Die OS Funktion R_USB3 dient zur Abfrage der USB Maus des SYMBiFACE III. Falls keine USB Maus angeschlossen ist wird in den Doppelregistern DE und IY jeweils der Wert &0000 zurückgegeben. Dies ist auch der Fall falls gar kein SF3 angeschlossen ist. Falls die SF3 USB Maus erfolgreich abgefragt wird, dann werden in Register D die fünf möglichen Feuerknöpfe in den Bits 4-0 angegeben.

In Register E wird der Versatz des Mausekranzes übermittelt. Und Register IY (low) und IY (high) enthalten die X- bzw. Y-Koordinate respektive.

Negative Werte werden im 2er Komplement angegeben, so wird -1 z.B. als 11111111 (binär) angegeben. Dadurch kann der übergebene Wert sofort zur aktuellen Koordinate addiert werden.

Bitte Beachten: Mit T_SF3 kann geprüft werden, ob das SF3 und damit die USB Maus angeschlossen sind.

Die USB Maus sollte vor dem Einschalten des SF3 (und anschließend des CPC) angeschlossen werden.

ABFRAGE DER BEIDEN MultiPlay MÄUSE

Kurzbeschreibung: Diese drei OS Funktionen dienen der Abfrage der beiden Mäuse, die an das MultiPlay (MP) angeschlossen werden können.

Labels: R_MPM, R_MPM1 und R_MPM2

ROM-Nummer: A

Startadresse: &FD5B(R_MPM), &FC7C(R_MPM1) und &FC99(R_MPM2)

Einsprungsbedingungen: Bei Verwendung von R_MPM1 und R_MPM2 sollte zuvor getestet werden, ob eine Maus an Port A oder B selektiert wurde.

Aussprungsbedingungen: Das Zero-Flag informiert über den Erfolg
Zero-Flag gesetzt: Keine Maus vorhanden oder sie wurde nicht bewegt
Zero-Flag geleert: Erfolg! Siehe Daten in folgenden Registern:

D = Feuer-Tasten %00000, Feuer 2, Feuer 1, Feuer 0 (1=AN)

E = Mauseisrad, immer &00 (aus Kompatibilität zu R_PS2 und R_USB3)

YL = X-Koordinate
= %0000 0001 (+1) bis %0000 1111 (+15) - Maus nach rechts bewegt
= %1111 1111 (-1) bis %1111 0000 (-15) - Maus nach links bewegt

YH = Y-Koordinate
= %0000 0001 (+1) bis %0000 1111 (+15) - Maus runter
= %1111 1111 (-1) bis %1111 0000 (-15) - Maus rauf

Manipuliert: AF, BC, DE und IY. Und bei R_MPM auch Register L

Beschreibung: Die OS Funktionen R_MPM, R_MPM1 und R_MPM2 dienen zur Abfrage der beiden Maus-Ports des MultiPlay. Dabei fragt R_MPM beide Mäuse ab und liefert die Daten der ersten Maus, die bewegt wurde.

R_MPM1 und R_MPM2 fragen respektive die Maus an Port 1 bzw. an Port 2 des MP ab. Diese beiden OS Funktionen darf man nur aufrufen, wenn auch ein MP angeschlossen wurde und die Ports auf Maus-Betrieb gestellt sind. Um das zu prüfen bitte Bit 3 in OS Variable KF_MED und die Bits 1 und 0 in OS ROM A Variable KOBYP testen (siehe Handbuch). Nur die OS Funktion R_MPM testet selbst ob eine Maus angeschlossen ist und bewegt wurde. Nach dem Rücksprung einer der drei OS Funktionen gibt das Zero-Flag an, ob die Maus bewegt wurde. Ist das Zero-Flag gesetzt, so ist entweder keine Maus bewegt worden oder keiner der MP-Ports wurde auf 'Maus' gestellt (siehe KOBYP in ROM A, siehe Handbuch). Wurde eine Maus erfolgreich abgefragt, dann werden in Register D die drei möglichen Feuerknöpfe in den Bits 2-0 angegeben. In Register E steht der Wert &00. Und zwar aus Gründen der Kompatibilität zu den OS Funktionen R_PS2 und R_USB3, die in E die Daten des Mauseisrads angeben.

Register IY (low) und IY (high) enthalten die Veränderung der X- bzw. Y-Koordinate respektive.

Negative Werte werden im 2er Komplement angegeben, so wird -1 z.B. als 11111111 (binär) angegeben. Dadurch kann der übergebene Wert sofort zur aktuellen Koordinate addiert werden.

Bitte Beachten: Nur OS Funktion R_MPM prüft, ob sowohl MP als auch eine Maus an einem Port des MP angeschlossen ist.

DEKOMPRIMIERUNG EINER ADVANCED OCP ART STUDIO *.SCR DATEI

Kurzbeschreibung: Diese beiden OS Funktionen dienen dazu eine komprimierte Bilddatei des Advanced OCP Art Studios zu dekomprimieren.

Label: OCPC0 und OCPXX

ROM-Nummer: A

Startadresse: &C18C (OCPC0) und &C18F (OCPXX)

Einsprungsbedingungen: HL = Quelladresse des komprimierten Bildes, das zuvor ins RAM geladen wurde.

Bei OCPXX muss DE zusätzlich mit einer Zieladresse geladen werden, ab der das entkomprimierte Bild (16 KB) abgelegt wird. Bei OCPC0 wird das Bild automatisch ab &C000 abgelegt.

Aussprungsbedingungen: Zero-Flag gesetzt --> 16KB entkomprimiertes Bild liegt ab &C000 (bei OCPC0) bzw. ab Adresse, die zuvor in DE geladen wurde, im Speicher.

Zero-Flag gelöscht --> Es ist ein Fehler aufgetreten!

Manipuliert: AF, BC, DE, HL, IX, IYL und RAM (entkomprimiertes Bild)

Beschreibung: Diese beiden OS Funktionen dienen dazu eine Bild – Datei (*.SCR), die vom Zeichenprogramm Advanced OCP Art Studio komprimiert wurde wieder zu entkomprimieren.

Dazu muss man das komprimiert Bild zuerst ins RAM laden. Die Adresse, an die das Bild geladen wurde, wird nun an OCPC0 bzw. OCPXX übergeben.

Nach dem Aufruf von OCPC0 befindet sich das entkomprimierte Bild ab der Adresse &C000 (bis &FFFF) im RAM.

Soll das Bild an eine andere Adresse im RAM entkomprimiert werden, so ist OS Funktion OCPXX zu verwenden. Vor dem Aufruf von OCPXX muss allerdings Register DE mit der entsprechenden Zieladresse geladen werden.

Die 16 KB Grafik-Daten des entkomprimierten Bildes werden nach Aufruf der OS Funktion im RAM abgelegt, exakt ab der Stelle, die zuvor mittels DE spezifiziert wurde.

Bitte Beachten: Wird OCPXX anstelle von OCPC0 verwendet, so ist DE mit einer Zieladresse zu laden, aber der Platz für 16 KB Grafik-Daten vorhanden ist.

SCHNELLE 8 DURCH 8 BIT DIVISION

Kurzbeschreibung: Diese OS Funktion dividiert 8 Bit durch 8 Bit.

Label: DIV88

ROM-Nummer: A

Startadresse: &C1DE

Einsprungsbedingungen: D = Dividend (die Zahl die geteilt wird)
E = Divisor (die Zahl mit der "D" geteilt wird)

Aussprungsbedingungen: D = Quotient (D/E)
A = Rest (unteilbarer Rest, der übrig bleibt)

Manipuliert: AF, D

Beschreibung: Diese OS Funktion erlaubt es einen 8 Bit Wert sehr schnell durch einen anderen 8 Bit Wert zu teilen (Integer Division).

In D wird der Dividend und in E der Divisor übergeben. Die OS Funktion teilt den Inhalt von Register D durch den Wert aus Register E.

Nach Aufruf der OS Funktion steht der Quotient der Division (D/E) in Register D. Sollte ein Rest übrig bleiben, so wird dieser in Register A (Akkumulator) übergeben.

Bitte Beachten: Es handelt sich um eine ganzzahlige Division. Auf dem CPC existiert wohl keine schnellere Möglichkeit 8 Bit durch 8 Bit zu teilen.

EDITIEREN DES SPEICHERS IM HEXADEZIMAL-MODUS

Kurzbeschreibung: Ein Speicherbereich von 01E0 Bytes (der bereits angezeigt wird) wird im Hexadezimal-Modus editiert.

Label: EDIT

ROM-Nummer: A

Startadresse: &FE5C

Einsprungsbedingungen:

- Register IY enthält die Adresse ab der editiert werden soll.
- Die RAM Variable MON_ROM enthält entweder &82 oder &86, je nachdem ob das untere ROM oder RAM während des Editierens eingeblendet werden soll (zwischen &0000 und &3FFF).
- Die RAM Variable MON_RAM enthält &C0 oder &C4, &C5 etc. je nachdem ob die ersten 64 KB oder Erweiterungs-RAM während des Editierens eingeblendet sein soll.
- Der zu editierende Speicherbereich wird bereits angezeigt (F_DUMP).

Aussprungsbedingungen: Speicher wurde editiert

Manipuliert: AF, BC, HL, AF', IX, PIO, PSG

Beschreibung: Beschränkt einsetzbare EDITier-Funktion des Speichers.

Bitte Beachten: --- weitere Informationen folgen ---

- CODE Beispiel auf nächster Seite -

Beispiel: Editieren des Speichers mit F_DUMP und EDIT

```
CALL S68X30 ;Setze Bildschirm auf 68 Zeichen und 30 Zeilen!

LD HL,&4000:LD (REG16_1),HL ;Startadresse ab der editiert wird

ED_L0 LD HL,(REG16_1):CALL F_DUMP ;DUMP ROM (in E-RAM) - 480 Bytes

CALL XWART ;Warten bis die Tastatur freigegeben ist

ED_L1 LD HL,&0804:CALL HOLE1TS:JP Z,ED_EX ;ESC!

CALL H_JC
RRCA:JR NC,ED_UP ;Auf
RRCA:JR NC,ED_DN ;Ab
RRCA:JR NC,ED_UP ;Links
RRCA:JR NC,ED_DN ;Rechts
RRCA:JR NC,ED_ED ;Feuer 0 (Editieren)
RRCA:JR NC,ED_EX ;Feuer 1 (Beenden)
JR ED_L1

ED_UP LD HL,(REG16_1):LD DE,&01E0:SBC HL,DE ;Auf / Links

LD A,H:CP A,&40:JR NC,EDUPO ;HL >= &4000
LD HL,&7E20

EDUPO LD (REG16_1),HL:JR ED_L0

ED_DN LD HL,(REG16_1):LD DE,&01E0:ADD HL,DE ;Ab / Rechts

LD A,H:CP A,&80:JR C,EDUPO ;HL < &7FFF
LD HL,&4000:JR EDUPO

ED_ED ;Feuer 0

LD IY,(REG16_1):CALL EDIT ;EDIT ROM (in E-RAM) - 480 Bytes
JR ED_L0

ED_EX RET ;Feuer 1 beendet
```

ABSCHALTEN ALLER KLANG-KANÄLE DES PSG

Kurzbeschreibung: Jegliche Klangproduktion durch den PSG wird sofort abgeschalten.

Label: MAUS

ROM-Nummer: A

Startadresse: &FE5F

Einsprunghbedingungen: -

Aussprunghbedingungen: Musik-Ausgabe ausgeschalten

Manipuliert: AF, BC, DE (= &3F07) und PIO, PSG

Beschreibung: MAUS (Musik AUS) dient dazu die komplette Klang-Erzeugung durch den PSG auszuschalten. Alle drei Kanäle werden ausgeschalten und die Lautstärke auf Null gesetzt.

Bitte Beachten: -

SENDEN EINES DATEN-BYTES AN EIN PSG REGISTER

Kurzbeschreibung: Ein Daten-Byte wird in ein PSG-Register geschrieben.

Label: S_PSG

ROM-Nummer: A

Startadresse: &FE62

Einsprungsbedingungen: D = Daten-Byte für den PSG
E = Nummer des PSG-Registers (Ziel)

Aussprungsbedingungen: Byte aus D wurde in PSG-Register aus E geschrieben.

Manipuliert: AF, BC, PIO und PSG

Beschreibung: Um ein beliebiges Byte in ein beliebiges Register des PSG zu schreiben wird S_PSG benutzt. Dabei können alle Register des PSG von &00 bis &0F verwendet werden.

Bitte Beachten: Nach dem Rücksprung wurde das PIO Status-Register manipuliert. Weiterhin wurde die Zeilen-Nummer der Tastatur-Matrix auf Null gesetzt und ein eventuell laufender Kassettenrekorder angehalten. Denn an den PIO Port C wurde zuletzt &00 geschickt.

ERMITTLE FREIEN 16 KB BLOCK IM ERWEITERUNGS-RAM

Kurzbeschreibung: Ein freier 16 KB Block wird in den ersten 512 KB E-RAM gesucht und in den System-Variablen als "genutzt" verbucht.

Label: EFER

ROM-Nummer: A

Startadresse: &D9C1

Einsprungsbedingungen: -

Aussprungsbedingungen: Der Akku A enthält die physikalische Nummer des freien 16 KB E-RAMs (von &C4 bis &FF) das gefunden wurde. Oder ...

Enthält der Akku den Wert 0 und ist das Zero-Flag gesetzt, so wurde KEIN freies 16 KB E-RAM gefunden.

Wurde ein freies E-RAM gefunden (physikalische Nr. in A), so enthält das Registerpaar HL einen Zeiger auf die zugehörige System-Variable XRAM_C4 bis XRAM_FF. Diese XRAM_?? Variable wurde bereits mit dem Wert &81 beschrieben, und das E-RAM dadurch als belegt markiert.

Manipuliert: AF, BC, HL, und eine der Variablen XRAM_C4 bis XRAM_FF.

Beschreibung: Die OS Funktion EFER dient dazu ein freies 16 KB großes Erweiterungs-RAM zu ermitteln. Dabei werden nur die ersten 512 KB durchsucht (physikalisch &7FC4 bis &7FFF).

Nach Aufruf von EFER wird im Akku die physikalische Nummer (RAM select) des freien 16 KB Blocks übergeben (&C4 bis &FF).

Ist das Zero-Flag gesetzt (und der Akku = &00), so wurde kein freies E-RAM gefunden.

Neben der physikalischen E-RAM Nummer (in A) wird im Register HL ein Zeiger auf die zu dem E-RAM gehörende XRAM Variable übergeben. Mit Hilfe dieses Zeigers ist es möglich dieses E-RAM nach Gebrauch wieder freizugeben. Dies geschieht durch schreiben des Wertes &01.

Die entsprechende XRAM Variable wird von EFER automatisch mit &81 beschreiben, damit ist das E-RAM als belegt markiert und kann nicht doppelt verwendet werden.

Ein Beispiel:

```
CALL EFER          ;ermittle freies E-RAM
JR    Z,FEHLER     ;Kein E-RAM frei !!!
PUSH HL           ;Zeiger auf Variable XRAM_?? sichern

LD    B,&7F
OUT   (C),A        ;E-RAM zwischen &4000 und &7FFF einblenden

... .. . ;hier kann mit dem E-RAM gearbeitet werden

POP HL            ;Zeiger auf Variable XRAM_?? holen
LD    (HL),&01     ;E-RAM nach Gebrauch wieder freigeben

...
```

Bitte Beachten: Die System-Variablen XRAM_C4 bis XRAM_FF müssen beim Einsprung korrekte Werte enthalten (dies ist normalerweise der Fall).

Um ein auf diese Art belegtes E-RAM wieder freizugeben ist einfach die von EFER in HL übergebene Variable/Adresse mit &01 zu beschreiben.

XRAM-Variable in physikalische E-RAM Auswahl umrechnen

Kurzbeschreibung: Ein Zeiger auf eine der 32 XRAM-Variablen wird in die zugehörige Erweiterungs-RAM Auswahl umgerechnet.

Label: XR2ER

ROM-Nummer: A

Startadresse: &C022

Einsprungsbedingungen: HL = Zeiger auf OS Variable XRAM_C4...FF.

Aussprungsbedingungen: Der Akku A enthält die physikalische Nummer des zugehörigen 16 KB E-RAM Blocks, also die Hardware E-RAM Auswahl von &C4 bis &FF.

Manipuliert: AF und H.

Beschreibung: Für die Verwaltung der ersten 512 KB Erweiterungs-RAM (E-RAM) stellt das FutureOS die XRAM-Variablen zur Verfügung. Sie tragen die Bezeichnungen XRAM_C4, XRAM_C5 ... XRAM_FF. Jede der 32 Variablen gibt Aufschluß über die Verwendung des zugehörigen 16 KB großen E-RAM Blocks. Diese Blöcke werden physikalisch über Port &7Fxx mittels der Bytes &C4, &C5 ... &FF ausgewählt. Ein Beispiel:

```
LD    BC, &7FC4    ;Auswahl des ersten E-RAMs (&C4) über den Port &7Fxx
OUT   (C), C        ;Das E-RAM im Bereich von &4000 bis &7FFF einblenden
```

Die OS Funktion XR2ER dient nun dazu eine der XRAM Variablen in die entsprechende Hardware E-RAM Auswahl umzurechnen. Dabei muss HL beim Einsprung auf die XRAM-Variable zeigen. Nach dem Aussprung enthält der Akku den Wert (&C4...&FF) für die Hardware E-RAM Auswahl.

Bitte Beachten: Nach dem Rücksprung dieser OS Funktion ist es möglich das berechnete E-RAM direkt einzublenden:

```
LD    B, &7F        ;Gate Array - E-RAM Banking
OUT   (C), A        ;E-RAM einblenden (zwischen &4000 und &7FFF)
```

LESE DATUM UND ZEIT VON EINER ECHTZEITUHR

Kurzbeschreibung: Datum und Zeit werden von einer angeschlossenen Echtzeituhr / Real-Time-Clock (RTC) gelesen.

Label: R_RTC

ROM-Nummer: A

Startadresse: &FE47

Einsprungsbedingungen: Es sollte eine RTC vorhanden sein.

Aussprungsbedingungen: Datum und Zeit wurden in die UHR_?? Variablen gelesen. Deren voriger Inhalt wurde in die AUH_?? Variablen kopiert.

Manipuliert: AF, BC, DE, HL die System-Variablen UHR_00 bis UHR_JAR und AUH_00 bis AUH_JAR. Ausserdem der RAM Bereich von &BE02 bis &BE0B.

Beschreibung: Für den CPC existieren verschiedene Echtzeituhren (RTCs) um den Computer mit aktuellen Zeit-Daten zu versorgen.

Die OS Funktion R_RTC prüft nun ob eine RTC von Dobbartin / dxs, M4, LambdaSpeak III / FS oder die des SYMBIFACE II oder III angeschlossen ist. Falls ja, dann werden Zeit und Datum in die System-Variablen UHR_00 bis UHR_JAR eingelesen (siehe Datei #OS-VAR.DEU). Zeit und Datum werden dabei vorteilhaft im BCD Format abgelegt.

Die alten Werte für Zeit und Datum werden vor dem Einlesen in die die System-Variablen AUH_00 bis AUH_JAR kopiert.

Bitte Beachten: Bei der Verwendung dieser OS Funktion solle entweder eine Dobbartin RTC (bzw. der Nachbau von dxs), eine M4 Karte, eine LambdaSpeak III/FS RTC, ein SF-II oder SF-III mit RTC vorhanden sein.

SCHREIBE DATUM UND ZEIT IN ALLE ANGESCHLOSSENEN ECHTZEITUHREN

Kurzbeschreibung: Datum und Zeit werden in alle angeschlossenen Echtzeituhren / Real-Time-Clocks (RTCs) geschrieben.

Label: S_RTC

ROM-Nummer: A

Startadresse: &FD7D

Einsprungsbedingungen: Es sollte mindestens eine RTC vorhanden sein.

Aussprungsbedingungen: Datum und Zeit wurden von den UHR_?? Variablen gelesen und in alle angeschlossenen RTCs geschrieben.

Manipuliert: AF, BC, DE, HL, BC', DE', HL' und alle angeschlossenen RTCs.

Beschreibung: Für den CPC existieren verschiedene Echtzeituhren (RTCs) um den Computer mit aktuellen Zeit-Daten zu versorgen.

Die OS Funktion S_RTC schreibt aktualisierte Daten von Zeit und Datum in alle momentan vorhandenen RTCs. Dabei werden die RTCs von Dobbartin / dxs, LambdaSpeak III / FS, SYMBIFACE II oder III berücksichtigt.

Dabei werden Zeit und Datum von den OS System-Variablen UHR_00 bis UHR_JAR gelesen (siehe Datei #OS-VAR.DEU). Zeit und Datum sind in diesen Variablen stets im vorteilhaften BCD Format abgelegt.

Die alten Werte für Zeit und Datum werden vor dem Einlesen in die die System-Variablen AUH_00 bis AUH_JAR kopiert.

Bitte Beachten: Bei der Verwendung dieser OS Funktion solle entweder eine Dobbartin RTC (bzw. der Nachbau von dxs), eine LambdaSpeak III/FS RTC, ein SF-II oder SF-III mit RTC vorhanden sein.

GEBE TEXT AUF BILDSCHIRM (IN MODE 2) UND VIA LAMBDA SPEAK AUS

Kurzbeschreibung: Text wird auf dem Bildschirm in MODE 2 ausgegeben und parallel dazu mit dem LambdaSpeak Sprach-Synthesizer gesprochen.

Label: LTERM_2

ROM-Nummer: A

Startadresse: &FD7A

Einsprunghbedingungen: Der Bildschirm-Modus 2 sollte aktiv sein.

Aussprunghbedingungen: Der Text wurde am Bildschirm ausgegeben und vom LambdaSpeak gesprochen.

Manipuliert: AF, BC, DE, HL, AF', BC', DE', HL', die Cursor Position C_POS und eine eventuelle vorherige Sprachausgabe wurde abgebrochen.

Beschreibung: Diese OS Funktion LTERM_2 arbeitet so wie die TERM_2 Textausgabefunktion, aber zusätzlich wird der Text auch an den LambdaSpeak Sprachsynthesizer geschickt und gesprochen. Sollte aktuell bereits ein Text gesprochen werden, so wird seine Ausgabe abgebrochen.

Der gesprochene Text enthält dabei keine Kontroll-Kodes, und seine Länge ist auf 255 Bytes limitiert.

Sollte die Sprachausgabe abgeschaltet sein (siehe OS Variable KOBYA in ROM A), so wird kein Text gesprochen.

Für die Textausgabe sollte der Bildschirm Modus 2 aktiv sein.

Bitte Beachten: Die Text Ausgabe erfolgt für MODE 2, also sollte MODE 2 auch aktiv sein. Die maximale Textlänge beträgt 255 Bytes. Die Sprachausgabe muss erlaubt sein (siehe Variable KOBYA in ROM A).

WANDLE BINÄR-ZAHL IN BCD-ZAHL UM

Kurzbeschreibung: Eine reguläre binäre Zahl (0-99) wird in eine Binär-Codierte-Dezimal (BCD) Zahl umgewandelt.

Label: BIN2BCD (bzw. BIN2BCE, wenn Register E = 10)

ROM-Nummer: A

Startadresse: &FD34 (bzw. &FD36)

Einsprungsbedingungen: D = Binärzahl von 0 bis 99
E = &0A (E = 10, bei Aufruf von BIN2BCE)

Aussprungsbedingungen: A = BCD-Zahl (&00...&99)
E = &0A (Register E ist mit 10 geladen)

Manipuliert: AF und DE

Beschreibung: Diese OS Funktion(en) wandeln eine Binärzahl in eine BCD Zahl um. BCD Zahlen lassen sich einfacher am Bildschirm darstellen. Im FutureOS sind z.B. die Uhrzeit und das Datum in BCD Zahlen abgelegt, siehe Variablen UHR_SEK bis UHR_JAR. Auch diverse Echtzeituhren nutzen das BCD Format (z.B. die RTC von Dobbertin, dxs bzw. SYMBiFACE II).

Beispiel: Das Register D wird mit einer Binärzahl von 0 bis 99 geladen und anschließend wird die OS Funktion BIN2BCD aufgerufen. Nach dem Rücksprung enthält das Register A die Zahl im BCD Format und Register E wurde mit dem Wert &0A = 10 geladen.

Will man diese Funktion mehrmals verwenden, so kann man ab dem zweiten Aufruf das Label BIN2BCE verwenden (denn Register E enthält bereits den Wert 10. E dient hier als Divisor).

```
LD    D,&12          ;D = &12 = 18 = Binärzahl
CALL  BIN2BCD        ;Binärzahl auf BCD konvertieren
LD    (XXXX),A       ;A = &18, BCD Zahl ins RAM schreiben

LD    D,&2D           ;D = 45
CALL  BIN2BCE        ;Binärzahl auf BCD,
                    ;hier Label BIN2BCE verwendbar

LD    (XXXX),A       ;A = &45, BCD Zahl in RAM
```

Bitte Beachten: Binär-Zahlen größer 99 können nicht im BCD Format dargestellt werden. Bitte nur Zahlen von 0 bis 99 verwenden.

Ruft man diese OS Funktion mit dem zweiten Label (BIN2BCE) auf, so ist die Funktion um 2 µs schneller, jedoch muss das Register E mit 10 geladen sein. Dies ist nach erstmaligem Aufruf von BIN2BCD der Fall.

WANDLE BCD-ZAHL IN BINÄR-ZAHL UM

Kurzbeschreibung: Eine Binär-Codierte-Dezimal (BCD) Zahl wird in eine reguläre binäre Zahl (0-99 / &00-&63) umgewandelt.

Label: BCD2BIN

ROM-Nummer: A

Startadresse: &FD4D

Einsprungsbedingungen: A = BCD-Zahl von &00 bis &99

Aussprungsbedingungen: A = Binärzahl von 0 bis 99 (= &00 bis &63)

Manipuliert: AF und DE

Beschreibung: Diese OS Funktion wandelt eine binär codierte Dezimal (BCD) Zahl in eine Binärzahl um. BCD Zahlen lassen sich einfacher am Bildschirm darstellen. Im FutureOS sind z.B. die Uhrzeit und das Datum in BCD Zahlen abgelegt, siehe Variablen UHR_SEK bis UHR_JAR. Auch diverse Echtzeituhren nutzen das BCD Format (z.B. die RTC von Dobbertin, dxs bzw. SYMBiFACE II).

Beispiel: Das Register A wird mit einer BCD Zahl von &00 bis &99 geladen und anschließend wird die OS Funktion BCD2BIN aufgerufen. Nach dem Rücksprung enthält das Register A die Zahl im Binärzahl (= Byte) Format.

Beispiel in Assembler:

```
LD    A,&18          ;A = &18, BCD Zahl
CALL  BCD2BIN        ;BCD-Zahl in Binärzahl konvertieren
LD    (XXXX),A       ;A = &12 = 18, binäre Zahl ins RAM schreiben
```

Bitte Beachten: Binär-Zahlen größer 99 können nicht im BCD Format dargestellt werden. Bitte nur Zahlen von &00 bis &99 konvertieren.

WARTEN BIS DAS SYMBiFACE III BEREIT IST

Kurzbeschreibung: Warten bis das SYMBiFACE III Bereitschaft meldet.

Label: W_SF3

ROM-Nummer: A

Startadresse: &FD43

Einsprungsbedingungen: Ein SF3 sollte vorhanden sein (siehe T_SF3)

Aussprungsbedingungen: Das SF3 ist jetzt bereit Kommandos zu empfangen
Register BC = &FD41 = SF3 Status-Register
Zero-Flag gesetzt: SF3 meldet 'alles bestens'

Manipuliert: AF und BC

Beschreibung: Diese OS Funktion wartet so lange bis das SYMBiFACE III Bereitschaft meldet. Dabei liefert Register A den Wert des Status-Registers des SF3: &00 (Zero Flag gesetzt) = alles in Ordnung. Ist jedoch Register A = &02, dann ist ein Fehler aufgetreten.

Außerdem wird BC mit &FD41 geladen, das ist die Portadresse des SF3 Kommando-/Status-Registers.

Bitte Beachten: Diese OS Funktion sollte nur verwendet werden, wenn auch tatsächlich ein SF3 angeschlossen ist. Dies kann man mit der OS Funktion **T_SF3** einmal testen.

Die neueste Version dieser Datei kann aus dem Internet bezogen werden:

<http://www.FutureOS.de>